SEATTLE COMPUTER
1114 Industry Dr. Seattle WA 98188 206/575-1830

CPU Support Board

Model 301

For the S-100 Bus

SEATTLE COMPUTER
1114 Industry Dr. Seattle WA 98188 206/575-1830

CPU Support Board

Model 301

For the S-100 Bus

If you have comments about this product or this manual, please complete the Comment Form at the back of the manual and return it to Seattle Computer Products, 1114 Industry Drive, Seattle, WA 98188.

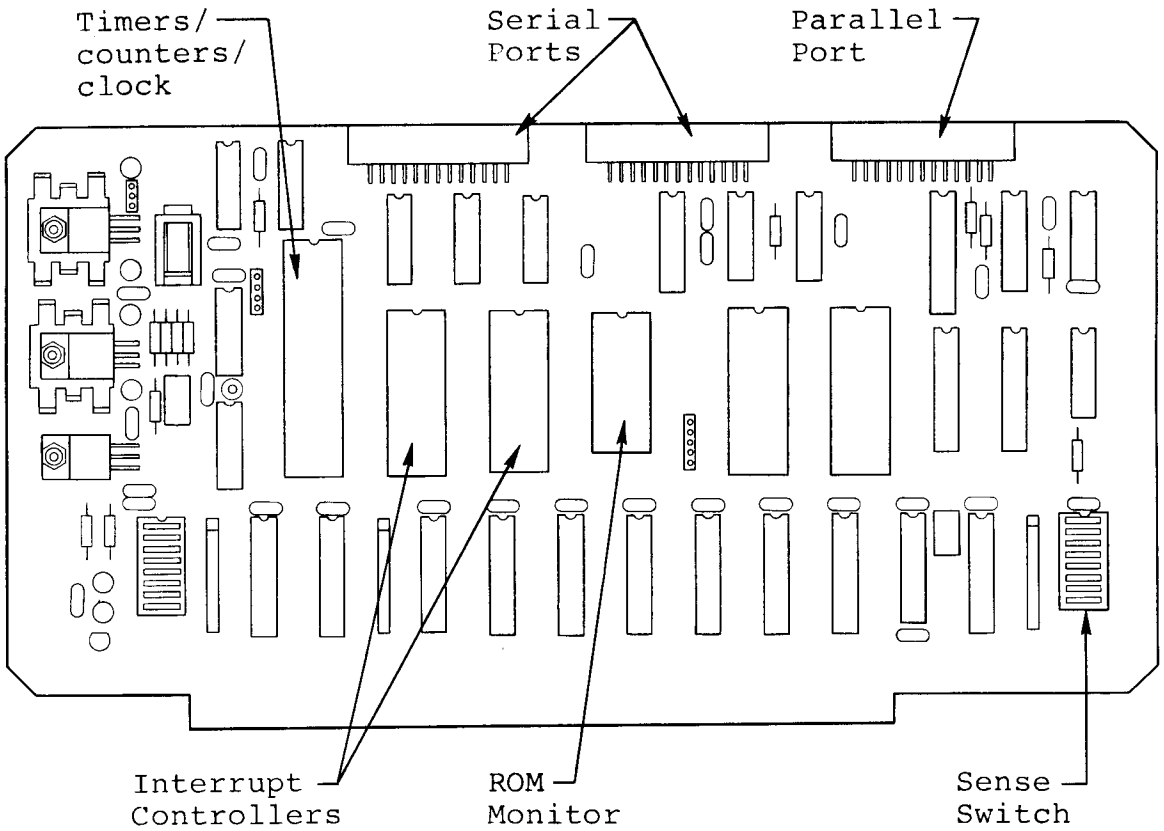
Part Number 900-000899

CONTENTS

CHAPTER 1. INTRODUCTION

In S-100 systems, the Seattle Computer CPU Support Card is an ideal companion board for Seattle Computer's 8086 CPU card. It can also be used with CPU cards from other manufacturers.

Since it combines several useful capabilities on a single S-100 card, system designers can save considerable amounts of design time.



The features of the CPU Support board are listed on the next page.

- Contains EPROM monitor for 8086. May be ordered without EPROM for other CPUs. Monitor is configured for Seattle Computer Disk Master or Tarbell double-density floppy disk controller.

- Two serial communication ports. RS-232 drivers with full handshaking. Independently programmable baud rate generator for each port. Sixteen baud rates from 50 to 19200 baud.

- A Centronics-compatible parallel port.

- A vectored interrupt controller. Provides 15 levels of vectored interrupts, expandable to 64 levels through slave controllers on other cards (such as Seattle Computer Products' Multiport Serial card). Includes complete interrupt support for 8080, 8085, Z80, and 8086 microprocessors.

- Five 16-bit timers. Two may be combined for time-of-day; the others may be used to count days, time events, etc. Totally software programmable.

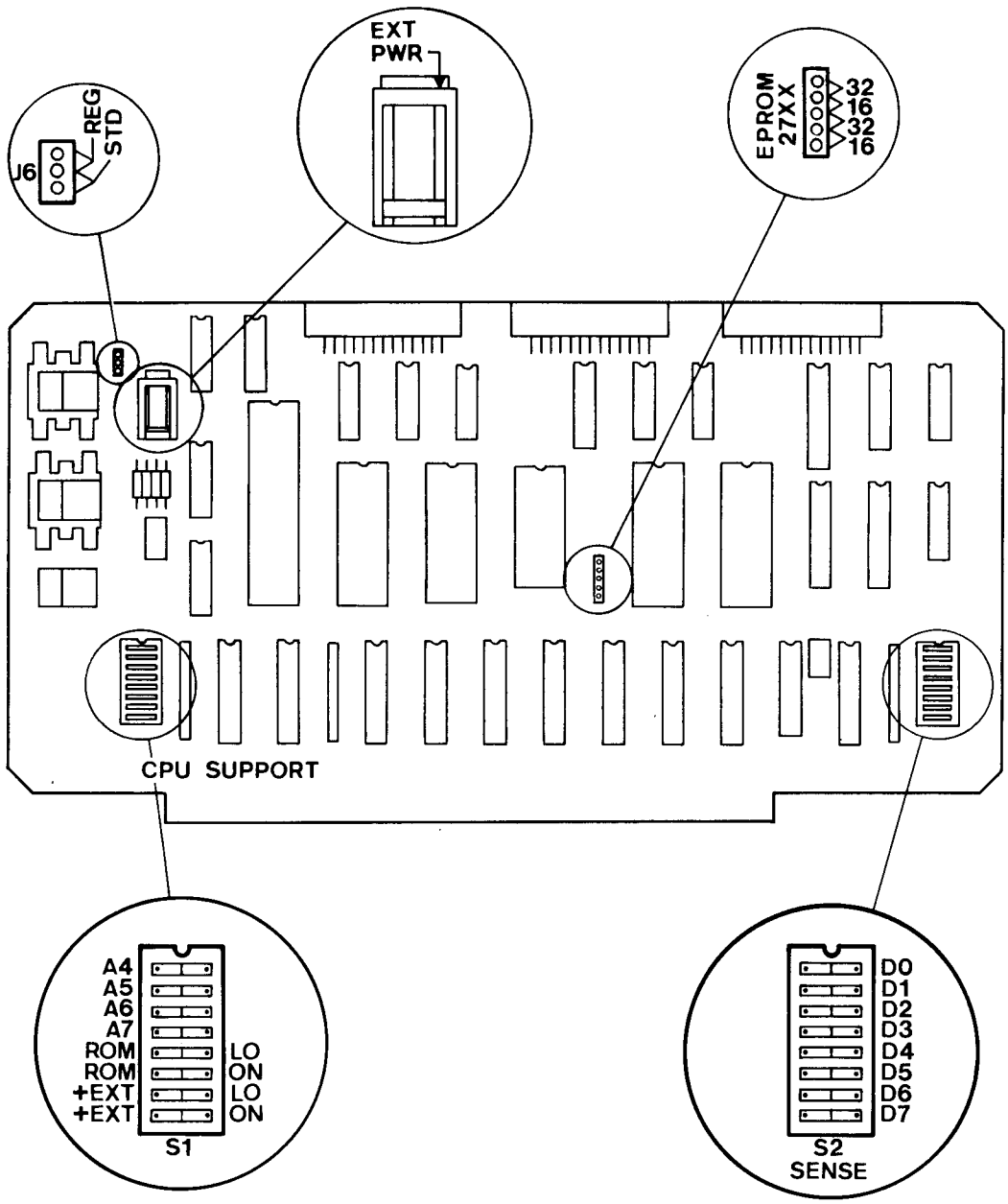
- A time-of day clock. Time of day is kept in 24-hour (military) format. Power to run the clock may be provided from an external source. May also be used as two general purpose 16-bit timers.

- A sense switch input port. Allows reading of 8-position DIP switch for setting software options.

- IEEE-696 Standard (S-100) compatible.

CHAPTER 2. SWITCH AND JUMPER SETTINGS

The locations of the switches and jumpers are shown below.



S1 SWITCH

A4, A5, A6, A7

Switch positions A4, A5, A6, and A7 set the highest four bits of the I/O port address. CLOSED represents a one, OPEN a zero. For details, see Chapter 3.

ROM (ON)

ROM (ON), switch position 6, enables or disables the ROM socket.

CLOSED Enables ROM. This is the factory setting.

OPEN Disables ROM

ROM (LO), EXT(LO), EXT

These three switches are used in combination to select the starting EPROM address. The type of EPROM (2716 or 2732) is chosen by the EPROM jumper (described below).

ROM (LO) controls extended addressing of the ROM, EXT (ON) enables or disables extended addressing for the ROM, and EXT (LO) selects whether the extended address bits are low or high. Settings for the 2732 and the 2716 EPROM are shown below.

Switch 5, 7, and 8 Settings for 2732 EPROM

ROM (LO) (Switch 5)	EXT (LO) (Switch 7)	EXT (Switch 8)	Starting and Ending Addresses of ROM
CLOSED	x	OPEN	xF000-xFFFF
CLOSED[1]	OPEN[1]	CLOSED[1]	FF000-xFFFF
CLOSED	CLOSED	CLOSED	0F000-xFFFF

NOTES: x = don't care
1 = Factory setting

Switch 5, 7, and 8 Settings for 2716 EPROM

ROM (LO) (Switch 5)	EXT (LO) (Switch 7)	EXT (Switch 8)	Starting and Ending Addresses of ROM
OPEN	x	OPEN	xF800-xF7FF
OPEN	OPEN	CLOSED	FF800-xF7FF
OPEN	CLOSED	CLOSED	0F800-xF7FF
NOTES: x = don't care			

S2 SWITCH (WITH 8086 MONITOR INSTALLED)

The CPU Support board comes from the factory with the 8086 Monitor installed in the ROM socket. With the 8086 Monitor installed, these are the functions of the S2 switch.

D0 (AUTO BOOT)

CLOSED The system will boot after reset once the baud rate has been determined. This is the factory setting.

OPEN The system will begin running the monitor program. The > prompt will appear on the screen expecting a monitor command. To boot the system, press a capital B followed by RETURN.

D1 (LARGE/SMALL DISK)

CLOSED For booting with a 5-1/4-inch disk in drive A.

OPEN For booting with an 8-inch disk in drive A or from a hard disk. This is the factory setting.

D2 (HARD/FLOPPY DISK)

CLOSED For booting with a hard disk.

OPEN For booting with a floppy disk in drive A. This is the factory setting.

D3 AND D4 (BAUD RATE)

Positions D3 and D4 can be set to correspond to the speed of the terminal.

Switch Position		Baud Rate
4	5	
OPEN*	OPEN*	Auto baud rate
CLOSED	OPEN	19200 baud rate
OPEN	CLOSED	9600 baud rate
CLOSED	CLOSED	300 baud rate
*This is the factory setting.		

When auto baud rate is selected, you just press the RETURN key several times after the booting the system. The monitor will select a baud rate of 110, 150, 300, 1200, 9600, or 19200.

D5, D6, D7

These positions are not used and should be OPEN.

S2 SWITCH (WITHOUT 8086 MONITOR INSTALLED)

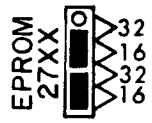
The eight positions of S2 can be read by the CPU by inputting from port BASE+15. (See Chapter 3 for details.) If you are not using the SCP 8086 Monitor EPROM, switch S2 settings have the following meanings.

Bit	Switch Position
D0	1
D1	2
D2	3
D3	4
D4	5
D5	6
D6	7
D7	8

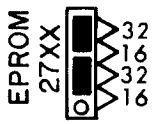
EPROM 27XX JUMPER

This jumper (J4) selects the type of ROM to be used.

To select an Intel-type (single supply) 2716 EPROM (including the TMS2516), place the two blue shunts as shown below:



To select an Intel-type 2732 EPROM, place the two blue shunts as shown below. (This is the factory setting.)



REG/STD JUMPER

This jumper selects whether the CPU Support card is running in a Gazelle Model II with a regulated (+5, +12V) S-100 bus or in a standard S-100 bus system.

REG CPU Support card is running in a Gazelle Model II without external power to the 9513 timer chip.

STD CPU Support card is running in a standard S-100 bus system and/or power is to be supplied to the 9513 timer chip through an external power supply.

EXT PWR CONNECTOR

To provide a constant power supply for the 9513 timer chip and to maintain current time even with system power off, you may attach an external +8 volt unregulated supply to the EXT PWR connector. If you attach an external power supply to the EXT PWR connector, be sure to set the REG/STD jumper to STD.

An example of this type of power supply is Radio Shack Model No. 273-1651. A plug only for the EXT PWR connector is a Radio Shack Model No. 274-286 plug.

CHAPTER 3. I/O PORTS

The CPU Support board requires twelve I/O ports for communication with the CPU. These twelve ports can be set on any 16-port boundary called the BASE. The BASE is selected using the first four switches of S1. These switches are labeled A7-A4 and correspond to address lines A7-A4 on the bus.

A7	A6	A5	A4	I/O Port Address BASE
0	0	0	0	00 hex
0	0	0	1	10 hex (factory setting)
0	0	1	0	20 hex
0	0	1	1	30 hex
0	1	0	0	40 hex
0	1	0	1	50 hex
0	1	1	0	60 hex
0	1	1	1	70 hex
1	0	0	0	80 hex
1	0	0	1	90 hex
1	0	1	0	A0 hex
1	0	1	1	B0 hex
1	1	0	0	C0 hex
1	1	0	1	D0 hex
1	1	1	0	E0 hex
1	1	1	1	F0 hex

NOTE: OPEN = 0; CLOSED = 1

The I/O ports are assigned as shown on the following page.

I/O Port	Input/Output	Use
BASE+0 and BASE+1	Input and output	Master programmable interrupt controller
BASE+2 and BASE+3	Input and output	Slave programmable interupt controller
BASE+4	Input and output	System timing controller data
BASE+5	Input and output	System timing controller status and control port
BASE+6	Input and output	Serial port 1, data
BASE+7	Input and output	Serial port 1, status and control
BASE+8	Input and output	Serial port 2, data
BASE+9	Input and output	Serial port 2, status/control
BASE+10	Output only	Serial port 1, baud rate
BASE+11	Output only	Serial port 2, baud rate
BASE+12	Output only	Parallel port, data/control
BASE+13	Input only	Parallel port, status
BASE+14	Input and output	Enable/disable EPROM
BASE+15*	Input only	Read 8-bit S2 switch

*Seattle Computer Products uses switch S2 (if BASE=F0 hex) for selecting software options starting with bit 0 (top switch) and working toward bit 7 (bottom switch) as more bits are needed. If you use these switches, you should start with bit 7 to ensure software compatibility for as long as possible.

CHAPTER 4. INTERRUPT CONTROLLERS

The CPU Support card can handle vectored interrupts from seven sources on the card and eight sources from the bus (VI0 - VI7). In addition, seven of the eight VI lines from the bus are cascadable to eight levels each, providing a total of sixty-four different interrupt sources.

The CPU Support card uses a pair of 8259A programmable interrupt controllers to handle the interrupt requests:

- The master controller, which has eight inputs, each of which can be connected to a slave controller to expand each single input from the master to eight from each slave.
- The slave controller, which handles the seven vectored interrupt sources on the board plus the one non-cascadable VI line from the bus.

Hardware Interrupt Vectors

Vector No.	Destination
Master IR0	VI0 (cascadable)
Master IR1	Not available (used by slave)
Master IR2	VI2 (cascadable)
Master IR3	VI3 (cascadable)
Master IR4	VI4 (cascadable)
Master IR5	VI5 (cascadable)
Master IR6	VI6 (cascadable)
Master IR7	VI7 (cascadable)
Slave IR0	OUT2 from 9513 timer
Slave IR1	RxRDY from serial port 1
Slave IR2	RxRDY from serial port 2
Slave IR3	VI1 (vectored interrupt line from S-100 bus, NOT cascadable)
Slave IR4	OUT4 from 9513 timer
Slave IR5	TxRDY from serial port 1
Slave IR6	TxRDY from serial port 2
Slave IR7	TxRDY from parallel port

NOTE: IR = Interrupt request

Each 8259A interrupt controller requires two of the CPU Support board's sixteen available I/O ports for communication with the processor.

I/O Port	Interrupt Controller
BASE+0	Master 8259A (A0 = 0)
BASE+1	Master 8259A (A0 = 1)
BASE+2	Slave 8259A (A0 = 0)
BASE+3	Slave 8259A (A0 = 1)

OPERATION OF THE INTERRUPT CONTROLLERS

Interrupt operation of the 8259A can be described in five main areas:

- Vectoring
- Priorities
- Triggering
- Status
- Cascading

Within each of these areas, there are various modes and commands. This section will explain the operation of these modes and commands. Programming of the 8259A is covered in "Programming the Interrupt Controllers".

Interrupt Vectoring

Each IR input of the 8259A has an individual interrupt-vector address in memory associated with it. Designation of each address depends upon the initial programming of the 8259A. The interrupt sequence and addressing on an 8080 system differs from that of an 8086 system. Thus, the 8259A must be initially programmed in either an 8080 or 8086 mode of operation to ensure the correct interrupt vectoring.

8080 Mode

This mode applies to 8080, 8085, and Z80 microprocessors. After an interrupt request in the 8080 mode, the 8259A outputs to the data bus the opcode for a CALL instruction and the address of the desired routine. This is in response to the sequence of three INTA (interrupt acknowledge) pulses issued by the CPU after the 8259A has activated the INT line of the S-100 bus.

The first INTA pulse to the 8259A enables the CALL opcode CD hex onto the data bus. It also resolves IR priorities and affects operation in the cascade mode (see the "Cascade Mode" section for more details).

During the second and third INTA pulses, the 8259A conveys a 16-bit interrupt-vector address to the 8080. The interrupt-vector addresses for all eight levels are selected when initially programming the 8259A. However, only one address is needed for programming. Interrupt-vector addresses of IR0 - IR7 are automatically set at equally spaced intervals based on the one programmed address. Address intervals are user-definable to 4 or 8 bytes apart.

If the service routine for a device is short, it may be possible to fit the entire routine within an 8-byte interval. Usually, though, the service routines require more than 8 bytes. So, a 4-byte interval is used to store a Jump instruction that directs the CPU to the appropriate routine. The 8-byte interval maintains compatibility with current 8080 Restart (RST) instruction software, while the 4-byte interval is best for a compact jump table. If the 4-byte interval is selected, then the 8259A automatically inserts bits A0-A4. This leaves A5-A15 to be programmed by the user. If the 8-byte interval is selected, the 8259A will automatically insert bits A0-A5. This leaves only A6-A15 for you to program.

The LSB of the interrupt-vector address is placed on the data bus during the second INTA pulse. The MSB of the interrupt-vector address isplaced on the data bus during the third INTA pulse.

8086 Mode

Upon interrupt in the 8086 mode, the 8259A outputs a single interrupt-vector byte to the data bus. This is in response to two INTA (interrupt acknowledge) pulses issued by the 8086 after the 8259A has activated the INT line of the S-100 bus:

The first INTA pulse is used only for set-up purposes internal to the 8259A. This set-up includes priority resolution and cascade mode operations which will be covered later.

The second INTA pulse is used to enable the single interrupt-vector byte onto the data bus. The 8086 uses this interrupt-vector byte to select one of 256 interrupt types in 8086 memory. Interrupt type selection for all eight IR levels is made when intially programming the 8259A. However, reference to only one interrupt type is needed for

programming. The upper five bits of the interrupt vector byte are user definable. The lower three bits are automatically inserted by the 8259A depending upon the IR level. Contents of the interrupt-vector byte for 8086-type selection is put on the data bus during the second INTA pulse.

Interrupt Priorities

A variety of modes and commands are available for controlling interrupt priorities of the 8259A:

- Fully nested mode
- End of interrupt
- Automatic rotation - equal priority
- Specific rotation - specific priority
- Interrupt masking

All these modes and commands are programmable, that is, they may be changed dynamically under software control. These modes and commands provide enough versatility for almost any interrupt controlled-application.

Fully Nested Mode

The fully nested mode of operation is a general purpose priority mode. This mode supports a multilevel interrupt structure in which priority order of all eight IR inputs are arranged from highest to lowest.

Unless otherwise programmed, the fully nested mode is entered by default upon intialization. At this time, IR0 is assigned the highest priority through IR7 the lowest. The fully nested mode, however, is not confined to this IR structure alone. Once past initialization, other IR inputs can be assigned highest priority also, keeping the multilevel-interrupt structure of the fully nested mode.

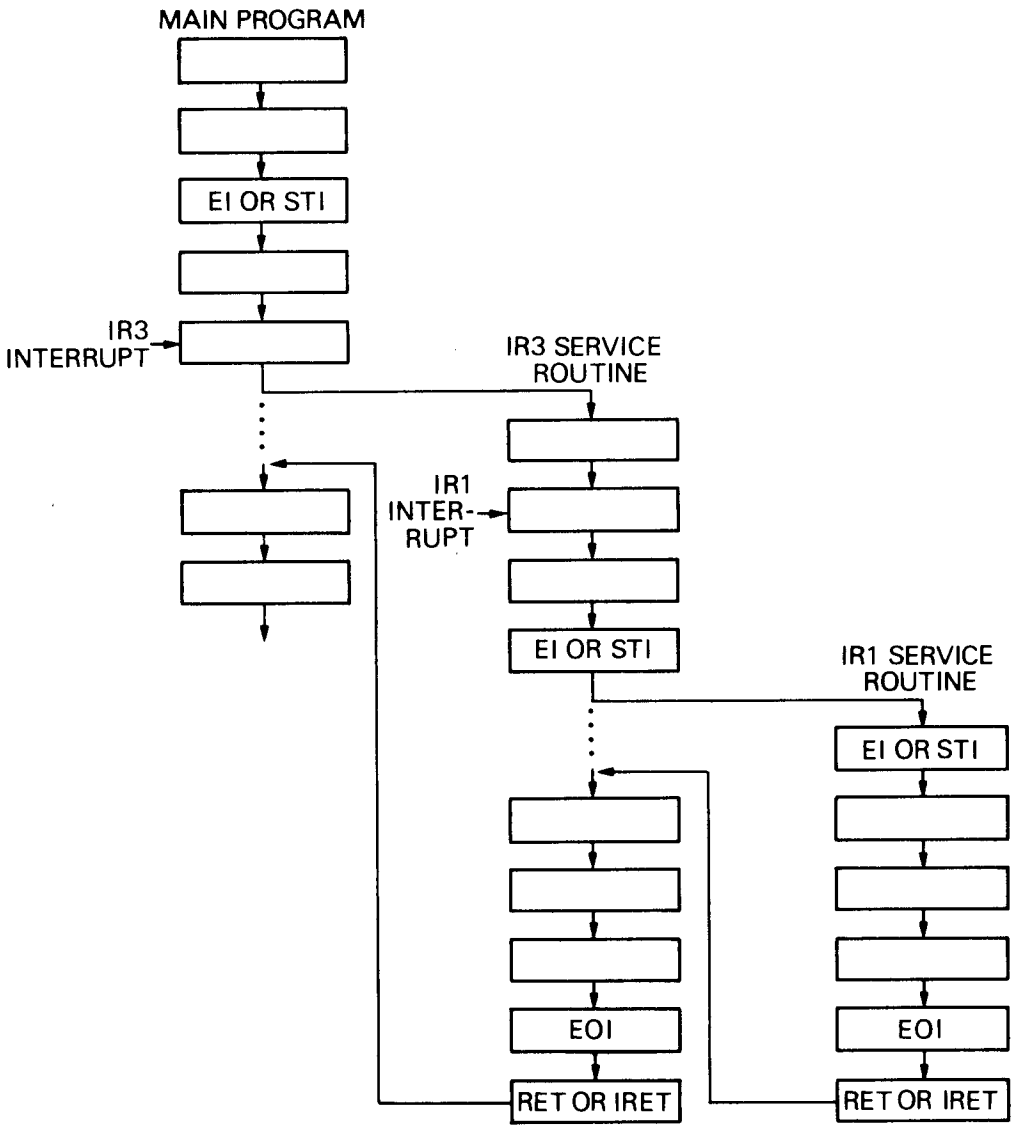
In general, when an interrupt is acknowledged, the highest priority request is determined from the IRR (Interrupt Request Register). The interrupt vector is then placed on the data bus. In addition, the corresponding bit in the ISR (In-Service Register) is set to designate the routine in service. This ISR bit remains set until an EOI (End-Of-Interrupt) command is issued to the 8259A. See the "End-of-Interrupt" section for more details.)

In the fully nested mode, while an ISR bit is set, all further requests of the same or lower priority are inhibited from generating an interrupt to the microprocessor. However,

a higher priority request can generate an interrupt, thus vectoring program execution to its service routine. Interrupts are only acknowledged, however, if the microprocessor has previously executed an Enable Interrupts instruction. This is because the interrupt request pin on the microprocessor gets disabled automatically after acknowledgement of any interrupt.

The assembly language instruction used to enable interrupts is EI. Interrupts can be disabled by using DI. When a routine is completed a Return instruction is executed: RET for 8080 and IRET for 8086.

The figure below illustrates the correct usage of interrupt related instructions and the interaction of interrupt levels in the fully nested mode.



Fully Nested Mode Example

Assuming IR0 has the highest priority and IR7 the lowest, the sequence is as follows. During the main program, IR3 makes a request. Since interrupts are enabled, the microprocessor is vectored to the IR3 service routine. During the IR3 routine, IR1 asserts a request. Since IR1 has higher priority than IR3, an interrupt is generated.

However, it is not acknowledged because the microprocessor disabled interrupts in response to the IR3 interrupt. The IR1 interrupt is not acknowledged until the Enabled Interrupts instruction is executed. Thus the IR3 routine has a protected section of code over which no interrupts (except non-maskable) are allowed. The IR1 routine has no such protected section since an Enable Interrupts instruction is the first one in its service routine. Note that in this example the IR1 request must stay active until it is acknowledged. This is covered in more depth in the "Interrupt Triggering" section.

What is happening to the ISR register? While in the main program, no ISR bits are set since there are not any interrupts in service. When the IR3 interrupt is acknowledged, the ISR3 bit is set. When the IR1 interrupt is acknowledged, both the ISR1 and the ISR3 bits are set, indicating that neither routine is complete. At this time, only IR0 could generate an interrupt since it is the only input with a higher priority than those previously in service.

To terminate the IR1 routine, the routine must inform the 8259A that it is complete by resetting its ISR bit. It does this by executing an EOI command. A Return instruction then transfers execution back to the IR3 routine. This allows IR0-IR2 to interrupt the IR3 routine again, since ISR3 is the highest ISR bit set.

No further interrupts occur in the example so the EOI command resets ISR3 and the Return instruction causes the main program to resume at its pre-interrupt location, ending the example.

A single 8259A is essentially always in the fully nested mode unless certain programming conditions disturb it. The following programming conditions can cause the 8259A to go out of the high to low priority structure of the fully nested mode.

- The Automatic EOI mode
- The Special Mask mode
- A slave with a master not in the special fully nested mode

These modes will be covered in more detail later. As long as these program conditions are not present, the fully nested mode remains undisturbed.

End of Interrupt

Upon completion of an interrupt service routine the 8259A needs to be notified so its ISR can be updated. This is done to keep track of which interrupt levels are in the process of being serviced and their relative priorities. Three different End-Of-Interrupt (EOI) formats are available:

- The Non-Specific EOI command
- The Specific EOI command
- The Automatic EOI Mode

Selection of which EOI to use depends on the interrupt operations you wish to perform.

Non-Specific EOI Command. A Non-Specific EOI command sent from the microprocessor lets the 8259A know when a service routine has been completed, without specification of its exact interrupt level. The 8259A automatically determines the interrupt level and resets the correct bit in the ISR.

To take advantage of the Non-Specific EOI the 8259A must be in a mode of operation in which it can predetermine in-service routine levels. For this reason the Non-Specific EOI command should only be used when the most recent level acknowledged and serviced is always the highest priority level. When the 8259A receives a Non-Specific EOI command, it simply resets the highest priority ISR bit, thus confirming to the 8259A that the highest priority routine of the routines in service is finished.

The main advantage of using the Non-Specific EOI command is that IR level specification is not necessary as in the Specific EOI Command, covered shortly. However, special consideration should be taken when deciding to use the non-Specific EOI . Here are two program conditions in which it is best not used:

- Using the Set Priority command within an interrupt service routine (see the "Set Priority Command" section)
- Using a Special Mask mode (see "Special Mask Mode" section)

Specific EOI Command. A Specific EOI command sent from the microprocessor lets the 8259A know when a service routine of a particular interrupt level is completed. Unlike a non-Specific EOI command, which automatically resets the highest priority ISR bit, a Specific EOI command specifies an exact ISR bit to be reset. One of the eight IR levels of the 8259A can be specified in the command.

The reason the Specific EOI command is needed is to reset the ISR bit of a completed service routine whenever the 8259A is not able to automatically determine it. An example of this type of situation might be if the priorities of the interrupt levels were changed during an interrupt routine (Specific Rotation). In this case, if any other routines were in service at the same time, a Non-Specific EOI might reset the wrong ISR bit.

Thus the Specific EOI command is the best bet in this case, or for that matter, any time in which confusion of interrupt priorities may exist. The Specific EOI command can be used in all conditions of 8259A operation, including those that prohibit Non-Specific EOI command usage.

Automatic EOI Mode. When programmed in the Automatic EOI mode, the microprocessor no longer needs to issue a command to notify the 8259A if it has completed an interrupt routine. The 8259A accomplishes this by performing a Non-Specific EOI automatically at the trailing edge of the last INTA pulse (third pulse in 8080; second in 8086).

The advantage of the Automatic EOI mode over the other EOI command is that no command has to be issued. In general, this simplifies programming and lowers code requirements within interrupt routines.

NOTE

> When the 8259A interrupt controller is used in the slave mode, the automatic end-of-interrupt mode does not work. The only solution is to send an end-of-interrupt command to the 8259A in the interrupt routine. The automatic end-of-interrupt mode does work if the 8259A is used as a master.

Special consideration should be taken when deciding to use the Automatic EOI mode because it disturbs the fully nested mode. In the Automatic EOI mode the ISR bit of a routine in service is reset right after it is acknowledged, thus leaving no designation in the ISR that a service routine is being executed. If any interrupt request occurs during this time (and interrupts are enabled) it will get serviced regardless of its priority, low or high.

The problem of over nesting is when an IR input keeps interrupting its own routine, resulting in unnecessary stack pushes which could fill the stack in a worst case condition. This is not usually a desired form of operation!

So what good is the Automatic EOI mode with problems like those just covered? Well, again, like the EOIs, selection is dependent upon the application. If interrupts are controlled at a predetermined rate, so as not to cause the problems mentioned above, the Automatic EOI mode works perfectly just the way it is. However, if interrupts happen sporadically at an indeterminate rate, the Automatic EOI mode should only be used under the following guideline:

> When using the Automatic EOI mode with an indeterminate interrupt rate, the microprocessor should keep its interrupt request input disabled during execution of service routines.

By doing this, higher priority interrupt levels will be serviced only after the completion of a routine in service. This guideline restores the fully nested structure in regards to the IRR; however, a routine in-service can not be interrupted.

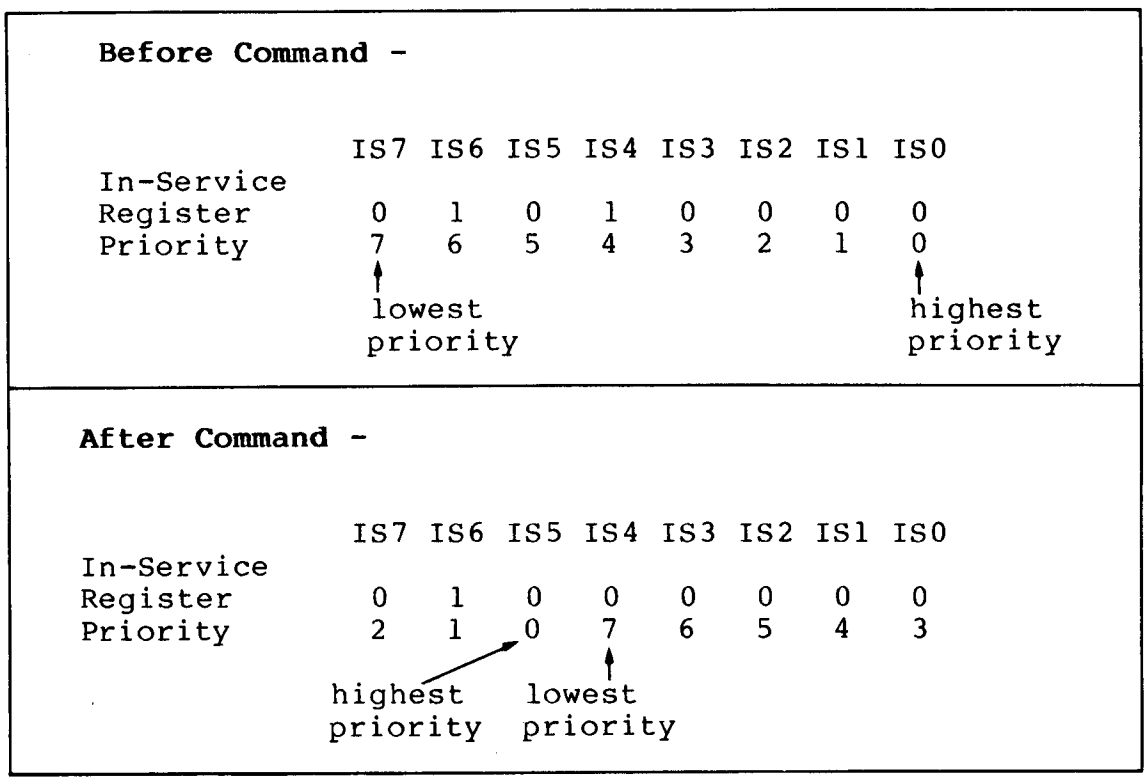
Automatic Rotation - Equal Priority

Automatic rotation of priorities serves in applications where the interrupting devices are of equal priority, such as communications channels. The concept is that once a peripheral is serviced, all other equal priority peripherals should be given a chance to be serviced before the original peripheral is serviced again. This is accomplished by automatically assigning a peripheral the lowest priority after being serviced. Thus, in worst case, the device would have to wait until all other devices are serviced before being serviced again.

There are two methods of accomplishing automatic rotation. One is used in conjunction with the Non-Specific EOI, Non-Specific EOI with Priority Rotation command. The other is used with the Automatic EOI mode, Rotate in Automatic EOI mode.

Non-Specific EOI with Priority Rotation Command. When the Non-Specific EOI with Priority Rotation command is issued, the highest ISR bit is reset as in a normal Non-Specific EOI command. After it is reset though, the corresponding IR level is assigned lowest priority. Other IR priorities rotate to conform to the fully nested mode based on the newly assigned low priority.

The figures below show how the Non-Specific EOI with Priority Rotation command effects the interrupt priorities. Let's assume the IR priorities were assigned with IR0 the highest and IR7 the lowest, as in the top figure. IR6 and IR4 are already in service but neither is completed. Being the higher priority routine, IR4 is necessarily the routine being executed. During the IR4 routine a Non-Specific EOI with Priority Rotation command is executed. When this happens, bit 4 in the ISR is reset. IR4 then becomes the lowest priority and IR5 becomes the highest as in the lower figure.



Example of Non-Specific EOI with Priority Rotation

Automatic EOI with Priority Rotation. The Automatic EOI with Priority Rotation mode works much like the Rotate on Non-Specific EOI command. The main difference is that priority rotation is done automatically after the last INTA pulse of an interrupt request.

To enter or exit this mode, Enable Rotation at Automatic EOI and Disable Rotation at Automatic EOI commands are provided. After that, no commands are needed as with the normal Automatic EOI mode. However, it must be remembered, when using any form of the Automatic EOI mode, special consideration should be taken. Thus, the guideline for the Automatic EOI mode also stands for the Automatic EOI with Priority Rotation mode.

Specific Rotation - Specific Priority

Specific Rotation gives you versatility in interrupt controlled operations. It serves in those applications in which a specific device's interrupt priority must be altered. As opposed to Automatic Rotation which automatically sets priorities, Specific Rotation is completely user controlled. That is, you select which interrupt level is to receive lowest or highest priority. This can be done during the main program or within interrupt routines. Two Specific Rotation commands are available to the user, the Set Priority command and the Specific EOI with Priority Rotation command.

Set Priority Command. The Set Priority command allows you to assign an IR level the lowest priority. All other interrupt levels will conform to fully nested mode based on newly assigned low priority.

An example of how the Set Priority command works is shown in the figures below. These figures show the status of ISR and the relative priorities of the interrupt levels before and after the Set Priority command. Two interrupt routines are shown to be in service in the top figure. Since IR2 is the highest priority, it is necessarily the routine being executed.

During the IR2 routine, priorities are altered so that IR5 is the highest. This is done simply by issuing the set priority command to the 8259A. In this case, the command specifies IR4 as being the lowest priority. The result of this Set Priority command is shown in the bottom figure.

Even though IR7 now has higher priority than IR2, it will not be acknowledged until the IR2 routine is finished (via EOI). This is because priorities are only resolved upon an interrupt request or an interrupt acknowledge sequence. If a higher priority request occurs during the IR2 routine, then priorities are resolved and the highest will be acknowledged.

Before Command -

	IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
In-Service Register	1	0	0	0	0	1	0	0
Priority	7	6	5	4	3	2	1	0
	↑ lowest priority							↑ highest priority

After Command -

	IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
In-Service Register:	1	0	0	0	0	1	0	0
Priority:	2	1	0	7	6	5	4	3
			↗ highest priority	↑ lowest priority				

Example of the Set Priority Command

When completing a service routine in which the set priority command is used, the correct EOI must be issued. The non-Specific EOI command should not be used in the same routine as a Set Priority command. This is because the Non-Specific EOI command resets the highest ISR bit, which, when using the Set Priority command, is not always the most recent routine in service.

The Automatic EOI mode, on the other hand, can be used with the Set Priority command. This is because it automatically performs a Non-Specific EOI before the Set Priority command can be issued. The Specific EOI command is the best bet in most cases when using the Set Priority command within a routine. By resetting the specific ISR bit of a routine being completed, confusion is eliminated.

Specific EOI With Priority Rotation Command. The Specific EOI with Priority Rotation command is literally a combination of the Set Priority command and the Specific EOI command. Like the Set Priority command, a specified IR level is assigned lowest priority. Like the Specific EOI command, a specified level will be reset in the ISR. Thus the Specific EOI with Priority Rotation command accomplishes both tasks in only one command.
This command is advantageous when it is not necessary to change IR priorities before the end of an interrupt routine. Since an EOI command must be executed anyway (unless in the Automatic EOI mode), why not do both at the same time?

Interrupt Masking

Disabling or enabling interrupts can be done by other means than just controlling the microprocessor's interrupt request pin. The 8259A has an IMR (Interrupt Mask Register) which enhances interrupt control capabilities. Rather than all interrupts being disabled or enabled at the same time, the IMR allows individual IR masking. The IMR is an 8-bit register; bits 0-7 directly correspond to IR0-IR7. Any IR input can be masked by writing to the IMR and setting the appropriate bit. Likewise, any IR input can be enabled by clearing the correct IMR bit.

There are various uses for masking off individual IR inputs. For example, when a portion of a main routine is only to be interrupted by specific interruptsor for disabling higher priority interrupts for a portion of a lower priority service routine.

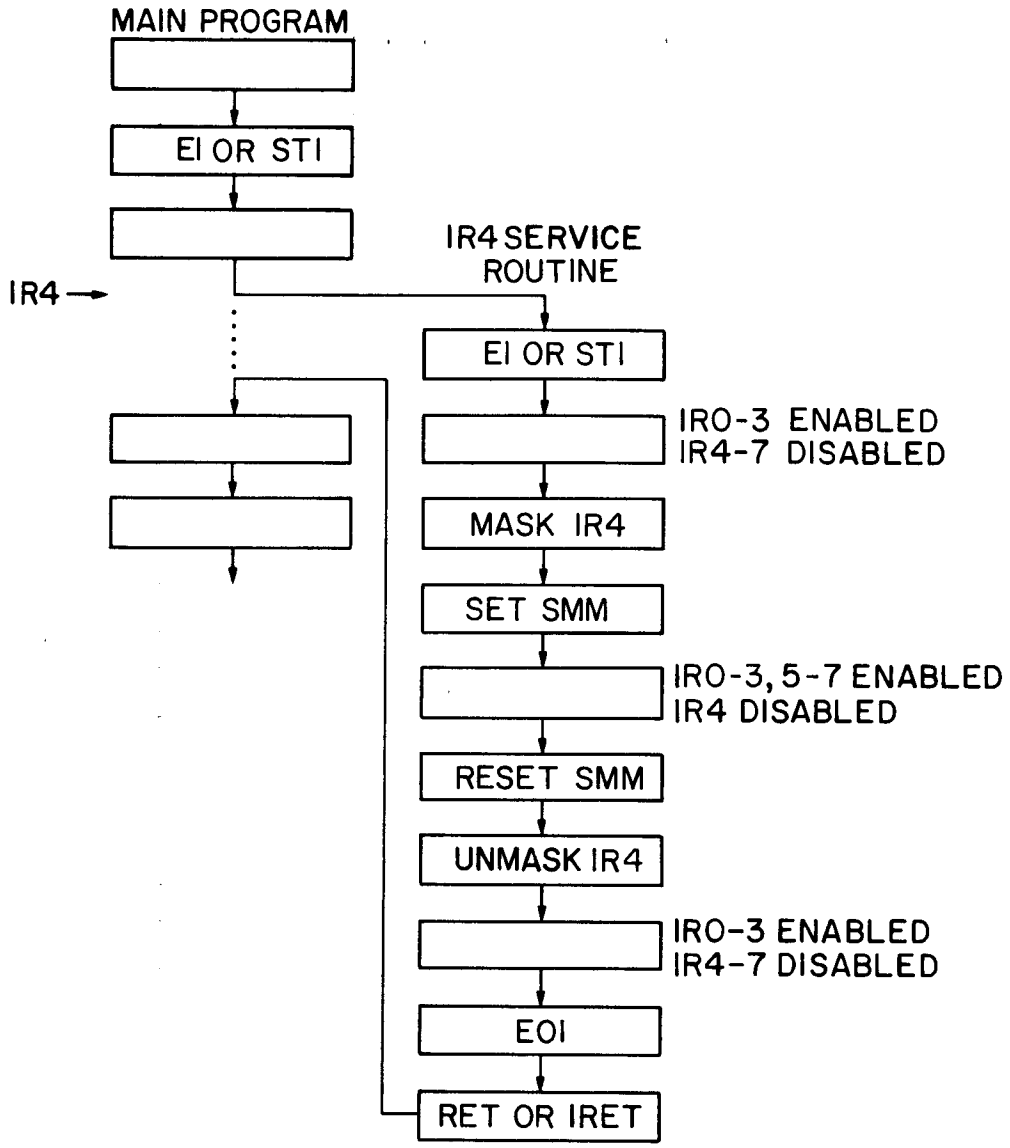
When an interrupt occurs while its IMR bit is set, it is not necessarily forgotten. For the IMR acts only on the output of IRR. Even with an IR input masked it is still possible to set the IRR. Thus, when resetting an IMR, if its IRR bit is set it will then generate an interrupt. This is providing, of course, that other priority factors are taken into consideration and the IR request remains active. If the IR request is removed before the IMR is reset, no interrupt will be acknowledged.

Special Mask Mode. Sometimes you may wish to enable interrupts of a lower priority than the routine in service. That is, you may want to allow lower priority devices to generate interrupts. However, in the fully nested mode, all IR levels of priority below the routine in service are inhibited. So what can be done to enable them?

One method could be using an EOI command before the actual completion of a routine in service. But doing this may cause an over nesting problem, similar to in the Automatic EOI mode. In addition, resetting an ISR bit is irreversible by software control, so lower priority IR levels could only be later disabled by setting the IMR.

A much better solution is the Special Mask mode. Working in conjunction with the IMR, the Special Mask mode enables interrupts from all levels except the level in service. This is done by masking the level that is in service and then issuing the Special Mask mode command. Once the special mask mode is set, it remains in effect until reset.

The figure below shows how to enable lower priority interrupts by using the Special Mask Mode (SMM). Assume that IR0 has highest priority when the main program is interrupted by IR4. In the IR4 service routine an enable interrupt instruction is executed. This only allows higher priority interrupt requests to interrupt IR4 in the normal fully nested mode. Further in the IR4 routine, bit 4 of the IMR is masked and the Special Mask mode is entered. Priority operation is no longer in the fully nested mode. All interrupt levels are enabled except for IR4. To leave the Special Mask mode, the sequence is executed in reverse.



Example of Special Mask Mode

Precautions must be taken when exiting an interrupt service routine which has used the Special Mask mode. A Non-Specific EOI command cannot be used when in the Special Mask mode. This is because a Non-Specific will not clear an ISR bit of an interrupt which is masked when in the Special Mask mode. In fact, the bit will appear invisible. If the special mask mode is cleared before an EOI command is issued a non-Specific EOI command can be used. This could be the case in the example shown in the previous figure, but to avoid any confusion it is best to use the Specific EOI whenever using the Special Mask mode.

Remember that the Special Mask mode applies to all masked levels when set. Take, for instance, IR1 interrupting IR4 in the previous example. If this happened while in the Special Mask mode, and the IR1 routine masked itself, all interrupts would be enabled except IR1 and IR4 which are masked.

Interrupt Triggering

There are two classical ways of sensing an active interrupt request: a level sensitive input or an edge sensitive input. The 8259A lets you use either method with the edge triggered mode and the level triggered mode. Selection of one of these interrupt triggering methods is done during the programmed initialization of the 8259A.

Level Triggered Mode

When in the level triggered mode the 8259A will recognize any active level on an IR input as an interrupt request. Note that an active level is high at the IR pin of the 8259A. The eight VI interrupt requests from the bus go through inverting buffers before reaching the 8259As so that the active level on the S-100 VI lines is low. If the IR input remains active after an EOI command has been issued (resetting its ISR bit), another interrupt will be generated. This is assuming of course, the processor INT pin is enabled.

Unless repetitous interrupt generation is desired, the IR input must be brought to an inactive state before an EOI command is issued in its service routine. Note that the request on the IR input must remain until after the falling edge of the first INTA pulse. If on any IR input, the request goes inactive before the first INTA pulse, the 8259A will respond as if IR7 was active. In any design in which there is a possibility of this happening, the IR7 default feature can be used as a safeguard. This can be accomplished by using the IR7 routine as a clean-up routine which might recheck the 8259A status or merely return program execution to its pre-interrupt location.

Depending upon the particular design and application, the level triggered mode has a number of uses. For one, it provides for repetitious interrupt generation. This is useful in cases when a service routine needs to be continually executed until the interrupt request goes inactive.

Caution should be taken when using the Automatic EOI mode and the level triggered mode together. Since in the Automatic EOI mode an EOI is automatically performed at the end of the interrupt acknowledge sequence, if the processor enables interrupts while an IR input is still active, an interrupt will occur immediately. To avoid this situation interrupts should be kept disabled until the end of the service routine or until the IR input goes inactive.

Edge Triggered Mode

When in the edge triggered mode, the 8259A will only recognize interrupts if generated by an inactive to active transition on an IR input. Note that active and inactive are as defined above in the section on the Level Triggered Mode.

The edge triggered mode incorporates an edge lockout method of operation. This means that after the rising edge of an interrupt request and the acknowledgement of the request, the active level of the IR input will not generate further interrupts on this level. You need not worry about quickly removing the request after acknowledgement in fear of generating further interrupts as might be the case in the level triggered mode. Before another interrupt can be generated the IR input must return to the inactive state.

Like the level triggered mode, in the edge triggered mode the request on the IR input must remain active until after the falling edge of the first INTA pulse for that particular interrupt. Unlike the level triggered mode, though, after the interrupt request is acknowledged its IRR latch is disarmed. Only after IR input goes inactive will the IRR latch again become armed, making it ready to receive another interrupt request (in the level triggered mode, the IRR latch is always armed). Note that the IR7 default feature mentioned in the level triggered mode section also works for the edge triggered mode.

Depending upon the particular design and application, the edge triggered mode has various uses. Because of its edge lockout operation, it is best used in those applications where repetitious interrupt generation is not desired. It is also very useful in systems where the interrupt request is a pulse (this should be in the form of a negative pulse to the 8259A).

Another possible advantage is that it can be used with the Automatic EOI mode without the cautions in the level

triggered mode. Overall, in most cases, the edge triggered mode simplifies operation for the user, since the duration of the interrupt request at a positive level is not usually a factor.

Interrupt Status

By means of software control, you can interrogate the status of the 8259A. This allows the reading of the internal interrupt registers, which may prove useful for interrupt control during service routines. It also provides for a modified status poll method of device monitoring, by using the poll command. This makes the status of the internal IR inputs available to the user via software control. The poll command offers an alternative to the interrupt vector method, especially for those cases when more than 64 interrupts are needed.

Reading Interrupt Registers

The contents of each 8-bit interrupt register, IRR, ISR, and IMR, can be read to update your program on the present status of the 8259A. This can be a versatile tool in the decision making process of a service routine, giving you more control over interrupt operations. Before delving into the actual process of reading the registers, let's briefly review their general descriptions:

- IRR (Interrupt Request Register) - Specifies all interrupt levels requesting service
- ISR (Interrupt Service Register) - Specifies all interrupt levels that are being serviced
- IMR (Interrupt Mask Register) - Specifies all interrupt levels that are being masked

To read the contents of the IRR or ISR, you must first issue the appropriate read register command (read IRR or read ISR) to the 8259A. Then by applying a RD pulse to the 8259A (an input instruction), the contents of the desired register can be acquired. There is no need to issue a read register command every time the IRR or ISR is to be read. Once a read register command is received by the 8259A, it remembers which register has been selected unless a Poll command is issued.

Thus, all that is necessary to read the contents of the same register more than once is the RD pulse and the correct addressing (A0=0, explained in "Programming the Interrupt Controllers"). Upon initialization, the selection of registers defaults to the IRR. Some caution should be taken when using the read register command in a system that supports several levels of interrupts.

If the higher priority routine causes an interrupt between the read register command and the actual input of the register command and the actual input of the register contents, there is no guarantee that the same register will be selected when it returns. Thus it is best in such cases to disable interrupts during the operation.

Reading the contents of the IMR is different than reading the IRR or ISR. A read register command is not necessary when reading the IMR. This is because the IMR can be addressed directly for both reading and writing. Thus all that the 8259A requires for reading the IMR is a RD pulse and the correct addressing (A0=1, explained in "Programming the Interrupt Controllers").

Poll Command

There are two methods of servicing peripherals: status polling and interrupt servicing. For most applications the interrupt service method is best. This is because it requires the least amount of CPU time, thus increasing systems throughput. However, for certain applications, the status poll method may be desirable.

For this reason, the 8259A supports polling operations with the Poll command. As opposed to the conventional method of polling, the Poll command offers improved device servicing and increased throughput. Rather than having the processor poll each peripheral in order to find the actual device requiring service, the processor polls the 8259A. This allows the use of all the previously mentioned priority modes and commands. Additionally, both polled and interrupt methods can be used within the same program.

To use the Poll command the processor must first have its interrupt request pin disabled. Once the Poll command is issued, the 8259A will treat the next RD pulse issued to it (an input instruction) as an interrupt acknowledge. It will then set the appropriate bit in the ISR, if there was an interrupt request, and enable a special word onto the data bus. This word shows whether an interrupt request has occured and the highest priority level requesting service.

The figure below shows the contents of the Poll word which is read by the processor. Bits 0-2 convey the binary code of the highest priority level requesting service. Bit 7 designates whether or not an interrupt request is present. If an interrupt request is present, bit 7 will equal 1. If there is not an interrupt request at all, bit 7 will equal 0 and bits 0-2 will be set to ones. Service to the requesting

device is achieved by software decoding the Poll word and branching to the appropriate service routine. Each time the 8259A is to be polled, the Poll command must be written before reading the Poll word.

	D7	D6	D5	D4	D3	D2	D1	D0
Poll word:	I	-	-	-	-	W2	W1	W0

I = 1 if an interrupt occurred

W0 - W2 = binary code of highest priority level requesting service

The Poll command is useful in various situations. For instance, it's a good alternative when memory is very limited, because an interrupt-vector table is not needed. Another use for the Poll command is when more than 64 interrupt levels are needed (64 is the limit when cascading 8259As). The only limit of interrupts using the Poll command is the number of 8259As that can be addressed in a particular system.

For those cases when the 8259A is using the Poll command only and not the interrupt method, each 8259A must still receive an intialization sequence. This must be done even though the interrupt vector features of the 8259A are not used. In this case, the interrupt vector specified in the initialization sequence could be a fake.

Interrupt Cascading

As mentioned earlier, more than one 8259A can be used to expand the priority interrupt scheme to up to 64 levels without additional hardware. This method for expanded interrupt capability is called cascading. The 8259A supports cascading operations with the cascade mode. Additionally, the special fully nested mode is available for increased flexibility when cascading 8259As in certain applications.

Cascade Mode

When programmed in the cascade mode, basic operations consists of one 8259A acting as a master to the others which are serving as slaves. On the CPU Support card, the 8259A at BASE+0 and BASE+1 acts as a master to the 8259A at BASE+2 and BASE+3 which serves as a slave. In addition, seven other 8259As can be slaved to the master 8259A using seven of the S-100 VI lines.

A specific hardware set-up is required to establish operation in the cascade mode. The INT output pin of each slave is connected to an IR input pin of the master. The slave 8259A on the CPU support card has its INT output connected to the master's IR1 input. Inputs IR0 and IR2 - IR7 of the master are connected to VI0 and VI2 - VI7 respectively. Any additional 8259As in the system (on the SCP Multiport Serial card for example) should have their INT outputs connected through open-collector inverting buffers to one of the seven VI lines which are cascadable (VI0 and VI2 - VI7). The eighth VI line, VI1, is NOT cascadable but instead is connected to IR3 of the slave 8259A on the CPU Support card. The master drives three CAS lines to indicate which of the up to eight slaves should drive the data bus with an interrupt vector during interrupt acknowledge.

The CPU Support card takes over the S-100 address bus during interrupt acknowledge and puts the three CAS lines on the lowest three address lines. Any additional slaves in the system should simply have their three CAS inputs connected to A0 - A2 to receive the proper cascade address. The SP/EN output pin is used to enable data bus drivers to drive the interrupt vector from the 8259A onto the S-100 DI bus. All other pins are connected as in normal operation.

Besides hardware set-up requirements, all 8259As must be software programmed to work in the cascade mode. Programming the cascade mode is done during the initialization of each 8259A. The 8259A that is selected as master must receive specifications during its initialization as to which of its IR inputs are connected to a slave's INT pin.

Each slave 8259A, on the other hand, must be designated during its initialization with an ID (0 through 7) corresponding to which of the master's IR inputs its INT pin is connected to. This is all necessary so the three CAS lines from the master will be able to address each individual slave.

Note that as in normal operation, each 8259A must also be initialized to give IR inputs a unique interrupt vector. More detail on the necessary programming of the cascade mode is explained in "Programming the Interrupt Controllers".

Now, with background information on both hardware and software for the cascade mode, let's go over the sequence of events that occur during a valid interrupt request from a slave. Suppose a slave IR input has received an interrupt request. Assuming this request is higher priority than other requests and in-service levels on the slave, the slave's INT pin is driven high. This signals the master of the request by causing an interrupt request on a designated IR pin of the master. Again, assuming that this request to the master

is higher priority than other master requests and in-service levels (possibly from other slaves), the master's INT pin is pulled high, interrupting the processor.

The interrupt acknowledge sequence appears to the processor the same as the non-cascading interrupt acknowledge sequence; however, it's different among the 8259As. The first INTA pulse is used by all the 8259As for internal set-up purposes, and, if in the 8080 mode, the master places the CALL opcode on the data bus. The first INTA pulse also signals the master to place the requesting slave's ID code on the CAS lines and hence on A0-A2 of the S-100 bus. This turns control over to the slave for the rest of the interrupt acknowledge sequence, placing the appropriate pre-programmed interrupt vector on the data bus, completing the interrupt request.

During the interrupt acknowledge sequence, the corresponding ISR bit of both the master and the slave get set. This means two EOI commands must be issued (if not in the Automatic EOI mode), one for the master and one for the slave.

Special consideration should be taken when mixed interrupt requests are assigned to a master 8259A; that is, when some of the master's IR inputs are used for slave interrupt requests and some are used for individual interrupt requests. In this type of structure, the master's IR0 must not be used for a slave. This is because when an IR input that is not initialized as a slave receives an interrupt request, the three CAS lines are not be activated and stay in the default condition addressing for IR0 (slave IR0).

If a slave is connected to the master's IR0 when a non-slave interrupt occurs on another master IR input, erroneous conditions may result. Thus IR0 should be the last choice when assigning slaves to IR inputs.

Special Fully Nested Mode

Depending on the application, changes in the nested structure of the cascade mode may be desired. This is because the nested structure of a slave 8259A differs from that of the normal fully nested mode. In the cascade mode, if a slave receives a higher priority interrupt request than one which is in service (through the same slave), it will not be recognised by the master. This is because the master's ISR bit is set, ignoring all requests of equal or lower priority. Thus, in this case, the higher priority slave interrupt will not be serviced until after the master's ISR bit is reset by an EOI command. This is most likely after the completion of the lower priority routine.

If you wish to have a truly fully nested structure within a slave 8259A, the special fully nested mode should be used. The special fully nested mode is programmed in the master

only. This is done during the master's initialization. In this mode the master will ignore only those interrupt requests of lower priority than the set ISR bit and will respond to all requests of equal or higher priority. Thus if a slave receives a higher priority request than one which is in service, it will be recognized.

To ensure proper interrupt operation when using the special fully nested mode, the software must determine if any other slave interrupts are still in service before issuing an EOI command to the master. This is done by resetting the appropriate slave ISR bit with an EOI and then reading its ISR. If the ISR contains all zeros, there are not any other interrupts from the slave in service and an EOI command can be sent to the master. If the ISR is not all zeros, an EOI command should not be sent to the master.

Clearing the master's ISR bit with an EOI command while there are still slave interrupts in service would allow lower priority interrupts to be recognized at the master.

PROGRAMMING THE INTERRUPT CONTROLLERS

The 8259A interrupt controllers are programmed with two types of command words: Initialization Command Words (ICWs) and Operational Command Words (OCWs). All the modes and commands explained in the previous section, "Operation of the Interrupt Controllers", are programmable using the ICWs and OCWs.

The ICWs are issued from the processor in a sequential format and are used to set up the 8259As in an initial state of operation. The OCWs are issued as needed to vary and control 8259A operation.

Both ICWs and OCWs are sent by the processor to the 8259A via output commands from the processor. The 8259A distinguishes between the different ICWs and OCWs by the state of its A0 pin, the sequence they're issued in (ICWs only), and some dedicated bits among the ICWs and OCWs.

The state of the A0 pin is defined by which port the ICW or OCW is sent to. A0 = 0 is BASE+0 for the master and BASE+2 for the slave. A0 = 1 is BASE+1 for the master and BASE+3 for the slave. Those bits which are dedicated are indicated so by fixed values (0 or 1) in the corresponding ICW or OCW programming formats.

Note

When issuing either ICWs or OCWs, the interrupts should be disabled at the processor (using the DI function of the CPU).

Initialization Command Words

Before normal operation can begin, each 8259A in the system (including the two on the CPU Support card and any others cascaded over the S-100 VI lines) must be initialized by a sequence of four ICWs. Once initialized, if any programming changes within the ICWs are to be made, all four ICWs must be reprogrammed in sequence.

Certain internal set up conditions occur automatically within the 8259As after the first ICW has been issued. These are:

1. The In-Service Register and Interrupt Mask Register are both cleared. (Clearing the IMR enables all eight interrupt request inputs).

2. The Special Mask mode is reset.

3. Rotation at Automatic End Of Interrupt is disabled.

4. The Interrupt Request Register is selected for the read register command.

5. The fully nested mode is entered with an initial priority assignment of IR0 highest through IR7 lowest.

6. The edge sense latch of each IR input is cleared thus requiring an active transition to generate an interrupt if the edge triggered input mode is chosen in ICW1.

The ICW programming format below shows bit designations. A complete description of each bit follows the table.

Bit	D7	D6	D5	D4	D3	D2	D1	D0
ICW1 A0=0	A7	A6	A5	1	LTIM	ADI	0	1
ICW2 A0=1	A15/T7	A14/T6	A13/T5	A12/T4	A11/T3	A10	A9	A8
ICW3 (Master) A0=1	S7	S6	S5	S4	S3	S2	1	S0
ICW3 (Slave) A0=1	0	0	0	0	0	ID2	ID1	ID0
ICW4 A0=1	0	0	0	SNFM	1	M/S	AEOI	µPM

ICW1 and ICW2

ADI: The ADI bit is used to specify the address interval for the 8080 mode. If a 4-byte address interval is to be used, ADI must equal 1. For an 8-byte address interval, ADI must equal 0. The state of ADI is ignored when the 8086 mode is selected.

LTIM: The LTIM bit is used to select between the two interrupt request input triggering modes. If LTIM = 1, the level triggered input mode is selected. If LTIM = 0, the edge triggered input mode is selected.

A5-A15: The A5-A15 bits are used to select the interrupt vector address when in the 8080 mode. There are two programming formats that can be used to do this. Which one is implemented depends upon the selected address interval (ADI). If ADI is set for the 4-byte interval, then the 8259A automatically inserts A0-A4 (A0-A1 = 0, A2-A4 = interrupt request number). Thus A5-A15 must be user-selected by programming the A5-A15 bits with the desired address. If aDI is set for the 8-byte interval, then A0-A5 are automatically inserted (A0-A2 = 0, A3-A5 = interrupt request number). This leaves A6-A15 to be selected by programming the A6-A15 bits with the desired address. The state programmed for A5 is ignored in the latter format.

T3-T7: The T3-T7 bits are used to select the interrupt type when the 8086 mode is used. The programming of T3-T7 selects the upper five bits of the interrupt type (interrupt vector). The lower three bits are automatically inserted corresponding to the interrupt request level causing the interrupt. The state of bits A5-A10 is ignored when in the 8086 mode. Computing the actual memory address of the interrupt is done by multiplying the interrupt type (that is, the interrupt vector) by four.

ICW3

S0-S7: If the 8259A being initialized is the master, then S0-S7 in ICW3 define which interrupt request inputs have slaves on them. A 1 designates a slave, a 0 means no slave (i.e. a normal interrupt request input). Bit 1 must always be set to 1 since IR1 is the CPU Support's slave 8259A. Any other slaves which exist in the system using the S-100 VI lines to connect to the master must have a bit of the master's ICW3 set to 1 corresponding to which VI line is used. For example if the Seattle Computer Products Multiport Serial card (model SCP-400) is connected to VI5, then S5 in the master's ICW3 must

be set to 1 to indicate to the master that the VI5 interrupt request input has a slave on it.

ID0-ID2: If the 8259A being initialized is a slave, then ID0-ID2 in ICW3 identify which of the master's interrupt request lines the slave is connected to. The slave 8259A on the CPU Support card is hard-wired to interrupt request 1 so 01 hex should always be used to program ICW3 in that particular 8259A. 05 hex would be used for ICW3 in the slave 8259A in the above example with the serial card.

ICW4

uPM: The uPM bit allows for selection of either the 8080 or 8086 mode. If set as a 1, the 8086 mode is selected; it set to 0, the 8080 mode is selected. Be sure the CPU jumper is properly set to match the uPM bit definition of which processor type is used.

AEOI: The AEOI bit is used to select the automatic end of interrupt mode. If AEOI = 1, the automatic end of interrupt mode is selected. If AEOI = 0, it is not selected; thus an EOI command must be used during a service routine.

NOTE:

When the 8259A interrupt controller is used in the slave mode, the automatic end-of-interrupt mode does not work. The only solution is to send an end-of-interrupt command to the 8259A in the interrupt routine. The automatic end-of-interrupt does work if the 8259A is used as a master.

M/S: The M/S bit defines whether the 8259A is the master or a slave. When M/S is set to a 1, the 8259A operates as the master; when M/S is 0, it operates as a slave.

SFNM: The SFNM bit designates selection of the special fully nested mode. Only the master should be programmed in the special fully nested mode to assure a truly fully nested structure among the slave interrupt request inputs. If SFNM is set to 1, the special fully nested mode is selected; if SFNM is 0, it is not selected.

Operational Command Words

Three OCWs are available for programming various modes and commands. Unlike the ICWs, OCWs need not be in any type of sequential order. Rather, they are issued by the processor as needed within a program.

The OCW programming format below shows the bit designation for each OCW. With the OCW format as reference, the functions of each OCW will be explained individually.

Bit	D7	D6	D5	D4	D3	D2	D1	D0
OCW1 A0=1	M7	M6	M5	M4	M3	M2	M1	M0
OCW2 A0=0	R	SL	EOI	0	0	L2	L1	L0
OCW3 A0=0	0	ESMM	SMM	0	1	P	RR	RIS

OCW1

OCW1 is used solely for 8259A masking operations. It provides a direct link to the Interrupt Mask Register. The processor can write to or read from the Interrupt Mask Register via OCW1. The OCW1 bit definition is as follows:

M0-M7: The M0-M7 bits are used to control the masking of the interrupt request inputs. If an M bit is set to a 1, it will mask the corresponding interrupt request input. A 0 clears the mask, thus enabling the interrupt request input. These bits convey the same meaning when being read by the processor for status update.

Note that to change the value of a single mask bit it is simplest to read the current mask, force the desired bit high or low using OR or AND instructions, and write the new mask back.

OCW2

OCW2 is used for end of interrupt, automatic rotation, and specific rotation operations. Associated commands and modes of these operations (with exception of AEOI initialization), are selected using the bits of OCW2 in a combined fashion as

explained below. A complete explanation of the different modes and commands is given in the previous section "Operation of the Interrupt Controllers."

Bits			Function
R	SL	EOI	
0	0	0	Disable Rotation at Automatic End of Interrupt (AEOI).
0	0	1	Non-Specific End of Interrupt.
0	1	0	No operation.
0	1	1	Specific End of Interrupt. L0-L2 is the ISR bit to reset.
1	0	0	Enable rotation at Automatic End of Interrupt (AEOI).
1	0	1	Non-Specific End of Interrupt with Priority Rotation.
1	1	0	Set priority using L0-L2.
1	1	1	Specific End of Interrupt with Priority Rotation.

R: The R bit is used to control all 8259A rotation operations. If the R bit is set to a 1, a form of priority rotation will be executed depending on the state of the SL and EOI bits. If R is 0, rotation will not be executed.

SL: The SL bit is used to select a specific level for a given operation. If SL is set to a 1, the L0-L2 bits are enabled. The operation selected by the EOI and R bits will be executed on the specified interrupt level. If SL is 0, the L0-L2 bits are disabled.

EOI: The EOI bit is used for all end of interrupt commands (not automatic end of interrupt mode). If set to a 1, a form of an end of interrupt command will be executed depending on the state of the SL and R bits. If EOI is 0, an end of interrupt command will not be executed.

L0-L2: The L0-L2 bits are used to designate an interrupt level (0-7) to be acted upon for the operation selectedby the EOI, SL, and R bits of OCW2. The level designated will either be used to reset a specific Interrupt Service Register (ISR) bit or to set a specific priority. The L0-L2 bits are enabled or disabled by the SL bit.

OCW3

OCW3 is used to issue various modes and commands to the 8259As. There are two main categories of operation associated with OCW3: interrupt status and interrupt masking. Bit definition of OCW3 is as follows:

ESMM: The ESMM bit is used to enable or disable the effect of the SMM bit. If ESMM is set to a 1, SMM is enabled. If ESMM is 0, SMM is disabled. This bit is useful to prevent interference of mode and command selections in OCW3.

SMM: The SMM bit is used to set the Special Mask mode. If SMM is set to a 1, the Special Mask mode is enabled. If it is 0, it is disabled. The state of the SMM bit is only honored if it is enabled by the ESMM bit.

P: The P bit is used to issue the Poll command. If P is set to a 1, the Poll command is issued. If it is 0, the Poll command is not issued. The Poll command will override a read register command if set simultaneously.

RR: The RR bit is used to execute the read register command. If RR is set to a 1, the read register command is issued and the state of RIS determines the register to be read. If RR is 0, the read register command is not issued.

RIS: The RIS bit is used to select the ISR or IRR for the read register command. If RIS is set to a 1, the In-Service Register is selected. If RIS is 0, the Interrupt Request Register is selected. The state of the RIS bit is only honored if the RR bit is a 1.

CHAPTER 5. TIMER

The CPU Support card has five programmable counters. One counter is dedicated for use as a baud rate generator for the serial I/O port. The other four are general purpose counters two of which can be combined for use as a time-of-day counter. A 9513 System Timing Controller chip is used to implement these counters.

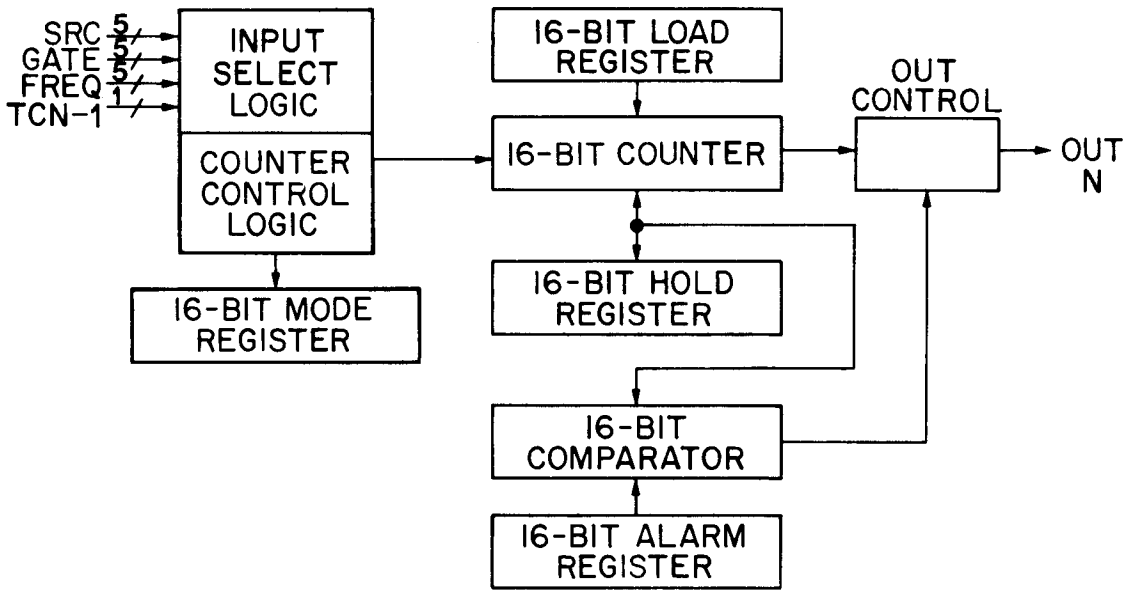
For more details on the 9513 chip, see the Advanced Micro Devices Am9513 System Timing Controller Data Sheet.

Communication with the 9513 is through two ports:

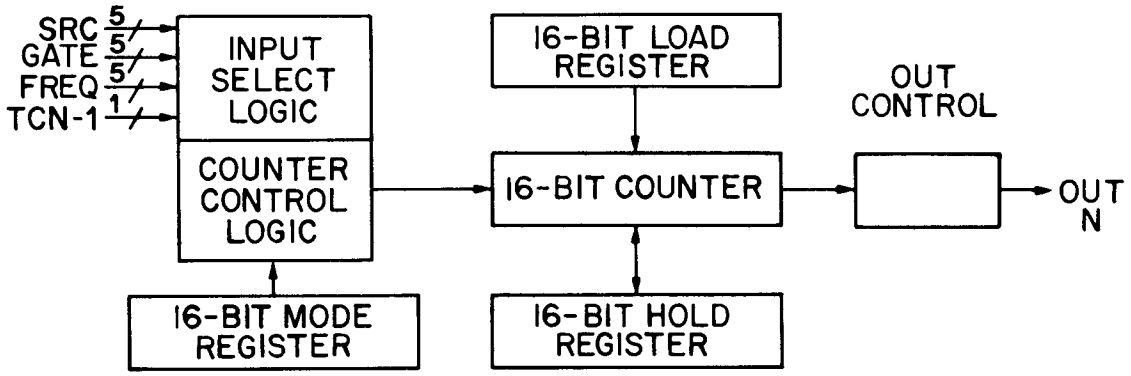
Port	Function
BASE+4	Data port
BASE+5	Status/Control port

The logic group that makes up each counter contains a 16-bit counter, three associated registers which control the counter and transfer data in and out of it, and an output pin which has different functions for each of the five counter logic groups.

Below are block diagrams of the counter logic groups.



Block Diagram of Counter Logic Groups 1 and 2



Block Diagram of Counter Logic Groups 3, 4, and 5

The counters and their output pins are defined below:

Counter	Function	Output connection
1	General purpose or time of day	Not connected
2	General purpose or time of day	Slave Interrupt Request 0
3	General purpose	Not connected
4	General purpose	Slave Interrupt Request 7
5	General purpose	Not connected

The clock input to each counter can be connected to one of seven different frequency sources. The clock polarity can be active high or low and counting can be in BCD or binary, all software selectable. The outputs can be either pulse or level (toggled) outputs and are software selectable.

FREQUENCY SOURCES

The 9513 has its own independant 4MHz oscillator which is the basic frequency source. Following the oscillator is a 16-bit frequency scaler with taps every four bits. The scaler can be programmed to count in BCD or binary.

	Binary		BCD	
Source	Frequency	Period	Frequency	Period
F1	4MHz	250nS	4MHz	250nS
F2	250kHz	4µS	400kHz	2.5µS
F3	15.625kHz	64µS	40kHz	25µS
F4	976.5625Hz	1.024mS	4kHz	250µS
F5	61.03515625Hz	16.384mS	400Hz	2.5mS

BCD division usually results in more useful frequencies and periods than binary division.

There is a 4-bit counter called the FOUT divider which can can be connected to any of the five scaler outputs. Unlike the main frequency scaler which can only divide by 10 or 16, the FOUT divider can divide by anything from 1 to 16. The output of the FOUT divider is one of the seven frequency sources.

The seventh source for each counter input is an output from the previous counter called TCN-1. This allows concatenation of the counters if more than sixteen bits are required.

TERMINAL COUNT

Terminal Count (TC) represents the period in time that the counter reaches an equivalent value of zero. Since the counter is always reloaded from its LOAD or HOLD register it will not actually reach zero unless the LOAD or HOLD register contains zero. TC is active during the first clock that the counter contains the value loaded from the LOAD or HOLD register. TCN-1 is the terminal count from the previous counter (N-1) and provides a way of cascading the counters internally. Counter 1 wraps around to counter 5 so that for counter 1, TCN-1 comes from counter 5.

9513 REGISTERS

The 9513 registers are shown below:

Register	Type	Port
Command	8 bits, write only	BASE+5
Data pointer	6 bits, write only	BASE+5
Status	8 bits, read only	BASE+5
Master mode	16 bits, read/writ	BASE+4
Load	16 bits, read/write, one for each counter	BASE+4
Hold	16 bits, read/write, one for each counter	BASE+4
Counter mode	16 bits, read/write, one for each counter	BASE+4
Alarm	16 bits, read/write, counters 1 and 2 only	BASE+4

Command Register

The command register is not really a register since the definitions of many of the bits change depending on the command. It is really just a port to which commands are sent.

C7	C6	C5	C4	C3	C2	C1	C0	Definition
0	0	0	E1	E0	G2	G1	G0	Load data pointer register with E and G fields
0	0	1	S5	S4	S3	S2	S1	Arm all counters for which the S bit is 1
0	1	0	S5	S4	S3	S2	S1	Load selected counters from LOAD or HOLD
0	1	1	S5	S4	S3	S2	S1	Load and arm all selected counters
1	0	0	S5	S4	S3	S2	S1	Disarm and save all selected counters
1	0	1	S5	S4	S3	S2	S1	Save all selected counters into the HOLD register
1	1	0	S5	S4	S3	S2	S1	Disarm all selected counters
1	1	1	0	0	N2	N1	N0	Clear output bit (N = 1,2,3,4,5)
1	1	1	0	1	N2	N1	N0	Set output bit (N = 1,2,3,4,5)
1	1	1	1	0	N2	N1	N0	Set counter N (N = 1,2,3,4,5)
1	1	1	0	0	0	0	0	Clear MM14 (enable data pointer sequencing)
1	1	1	0	1	0	0	0	Set MM14 (disable data pointer sequencing)
1	1	1	1	1	1	1	1	Master reset

The data pointer register selects which of the 16 bit read/write registers the CPU communicates with through the data port (BASE+4). It is loaded using one of the commands. See the "Data Pointer Register" section for more information.

Six of the commands provide direct software control over the counting process. Each of these commands contains a 5-bit S field. Each bit in the S field corresponds to one of the five counters (S1 = counter 1, etc.). A 1 in an S bit causes the command to be performed on the corresponding counter. If a counter's S bit is zero, no operation is performed.

Arm - The Arm command enables the selected counters to count.

Load - The Load command causes the selected counters to be loaded from their LOAD or HOLD register depending on the counter mode register.

Disarm - The Disarm command disables counting in the selected counters. A disarmed counter can still be loaded, stepped, or saved.

Save - The Save command transfers the contents of a counter to its hold register where it can be read by the CPU. This transfer takes place without disturbing the count process. The save command is the only way to read the value of the counter because the counter is not directly accessable by the CPU.

The following three commands can only affect one counter at a time. The desired counter is selected using the N field where N0-N2 form a 3-bit binary number. N must be in the range 1 - 5 for these commands.

Set output - This command activates the selected output pin in the mode selected by the counter mode register. It could be active high, active low, or disabled.

Clear output - This command deactivates the selected output pin in the mode selected by the counter mode register. It could be inactive low, inactive high, or disabled.

Step counter - This command steps the selected counter up or down once, the direction depending on the counter mode register.

The following two commands set or clear a bit of the Master Mode register without having to load the data pointer register to access the Master Mode register.

Set/Clear MM14 - Disable/enable data pointer sequencing (respectively).

Master reset - On reset all five counters are disarmed, 0B00 hex is loaded into each Counter Mode register, and 0000 hex is loaded into the Master Mode register. This results in each counter being configured to count down in binary on the positive-going edge of the F1 frequency source with no repetition. The counter outputs are off with a low impedance to ground. The Master Mode register is cleared to configure the 9513 for binary division of the internal oscillator, the FOUT divider dividing by sixteen, Time Of Day mode and comparators 1 and 2 disabled, and Data Pointer sequencing enabled.

Reset will clear the Load and Hold registers for each counter but will not change either the counter contents or the Data Pointer register.

Data Pointer Register

The Data Pointer register is used to establish which of the 16 bit read/write registers the CPU will communicate with through the data port. The Data Pointer register has some automatic sequencing features to allow you to load or scan through the various registers without the need to update the Data Pointer for each register.

Bit 14 of the Master Mode register (MM14) controls whether the automatic sequencing is enabled or not. The Data Pointer has three parts, the group pointer (G), the element pointer (E), and the byte pointer (B). The group pointer selects one of the five counter groups or the control group. The element pointer selects an element of the group. The byte pointer selects the least or most significant byte of the sixteen bit word.

Whenever the command to load the Data Pointer register is given the byte pointer is set to 1 indicating the least significant byte is expected. The byte pointer is toggled after each data transfer. The E and G fields are sequenced every other transfer to allow both bytes of the sixteen bit registers to be transfered. Since the byte pointer is set when the Data Pointer register is loaded the least significant byte is always transfered first.

Byte Pointer

1 = Least significant byte transferred next
0 = Most significant byte transferred next

Group Pointer

000 = Illegal
001 = Counter Group 1
010 = Counter Group 2
011 = Counter Group 3
100 = Counter Group 4
101 = Counter Group 5
110 = Illegal
111 = Control Group

Element Pointer

00 = Mode Register
01 = Load Register
10 = Hold Register
11 = Hold Register/ Hold Cycle Increment

(Element Cycle Increment)

00 = Alarm Register 1
01 = Alarm Register 2
10 = Master Mode Register
11 = Status Register/ No Increment

(Control Cycle Increment)

Data Pointer Register

The register addressed by the Data Pointer is read at the time the Data Pointer is set, with its contents being placed in the output buffer register. This happens

1) whenever the Data Pointer is set by a command to BASE+5;

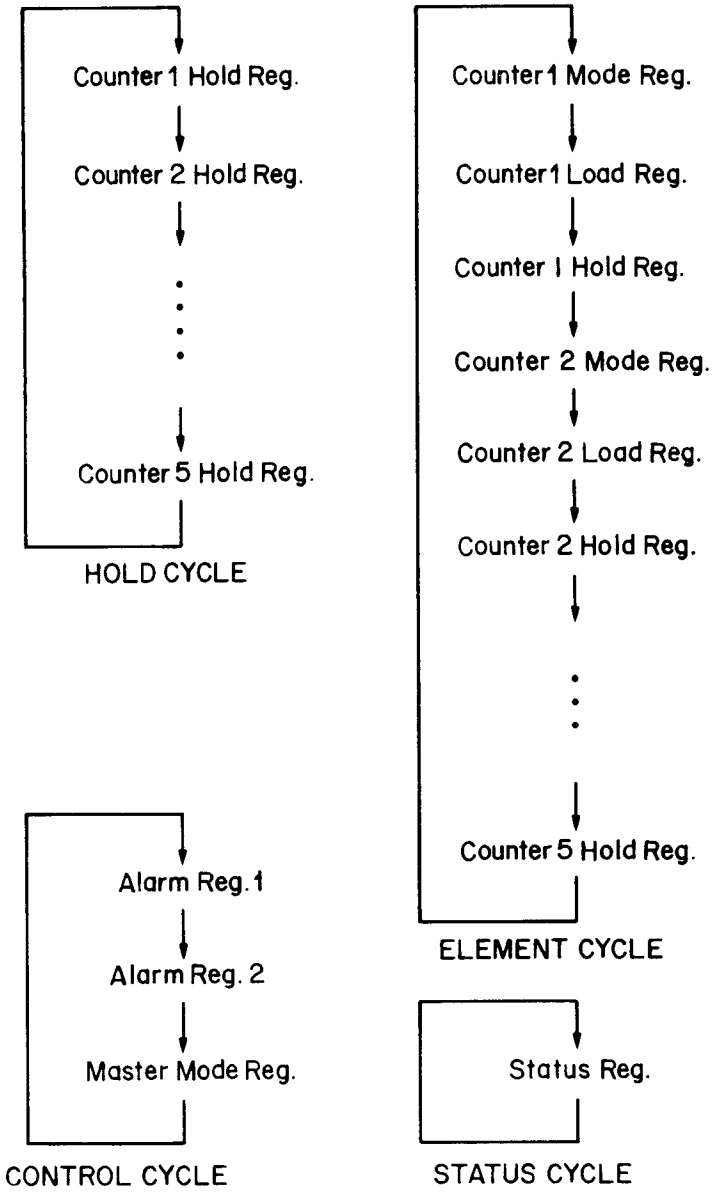
   or

2) whenever the data port at BASE+4 is read or written since this toggles the byte pointer.

If the contents of a register are changed after the data pointer is set to point to that register, the correct data will not be read. For example, setting the Data Pointer to Hold Register 1, then saving Counter 1 into the Hold Register will result in reading erroneous data. The data will be correct if the data pointer is set **after** the save command is given.

Another problem can occur if interrupts are possible when the 9513 is being read or written. The interrupt could occur between setting the data pointer register and reading the data, or even between transferring low and high data bytes. If the interrupt service routine then also communicates with the 9513, it may change the Data Pointer. Moral: disable interrupts while communicating with the 9513.

The four Data Pointer Sequencing loops are shown in the following figure.



Data Pointer Sequencing Loops

Status Register

The 8-bit read-only status register is read with every input from the control/status port. It contains the status of the five output pins and the byte pointer (B) of the data pointer register. The status register may also be read as part of the control group.

SR7	SR6	SR5	SR4	SR3	SR2	SR1	SR0
1	1	OUT5	OUT4	OUT3	OUT2	OUT1	B

The values read for the output pins are internal values taken after the polarity select logic but before three state buffer circuitry.

Master Mode Register

The 16-bit read/write Master Mode register is accessed through the data port as part of the control group. It is used to control internal activities which are not covered by the individual counter mode registers.

The Master Mode register contains bits MM15-MM0 and is summarized below:

```
MM15 Scaler control
   0 = Binary division
   1 = BCD division
MM14 Data pointer control
   0 = enable sequencing
   1 = disable sequencing
MM13 = 0
MM12 = 0
MM8-MM11 FOUT divider
   0000 = divide by 16
   0001 = divide by 1
   0010 = divide by 2
             .
             .
             .
   1110 = divide by 14
   1111 = divide by 15

MM4-MM7 FOUT source
   0000 = F1
   1011 = F1
   1100 = F2
   1101 = F3
   1110 = F4
   1111 = F5
MM3 Compare 2 enable
   0 = disabled
   1 = enabled
MM2 Compare 1 enable
   0 = disabled
   1 = enabled
MM0-MM1 Time-of-day
        enable
   00 = time-of-day
        disabled
   11 = time-of-day
        enabled
```

MM15: Scaler control. Selects whether the internal oscillator frequency scaler counts in binary or BCD.

MM14: Data pointer control. Selects whether the data pointer sequences or stays pointing to a particular
MM13: This bit should always be zero.

MM12: This bit should always be zero.

MM8-MM11: FOUT divider. Chooses what the FOUT source frequency is divided by before being presented to the FOUT pin. Divisors from 1 through 16 are available.

MM4-MM7: FOUT source. Selects one of the five outputs of the internal oscillator scaler for input to the FOUT divider. Only those binary combinations given in the table should be used. The others are from input pins which are not connected to anything. See the 9513 data sheet in Appendix B for more information. Note that F1 can be selected with two different combinations.

MM3: Compare 2 enable. Controls counter 2's comparator. See the section on "Alarm Registers."

MM2: Compare 1 enable. Controls counter 1's comparator. See the "Alarm Registers" section.

MM0-MM1: Time Of Day mode. These two bits enable or disable the Time Of Day mode in counters 1 and 2.

Load Register

This 16-bit read/write register is used to load the counter when Terminal Count occurs or when a load command is given. Each counter has one and it is accessed as an element of a counter group.

Hold Register

This 16-bit read/write register is used to save the contents of the count when a save command is received or it can be used to load the counter on alternate Terminal Counts alternating with the LOAD register. Each counter has one and it is accessed as an element of a counter group.

Counter Mode Register

This 16-bit read/write register controls the operation of the counter. Each counter has this register and it is accessed as an element of a counter group.

The Counter Mode register contains bits CM15-CM0 and is summarized below:

```
CM13-CM15 = 000
CM12 Count source polarity
     0 = Count on rising edge
     1 = Count on falling edge
CM8-CM11 Count source
     0000 = TCN-1
     0001 = FOUT
     1011 = F1
     1100 = F2
     1101 = F3
     1110 = F4
     1111 = F5
CM7 = 0
CM6 Reload source
     0 = Reload from load
     1 = Reload from load or hold

CM5 Repetition
     0 = Count once
     1 = Count repeatedly
CM4 Count base
     0 = Binary count
     1 = BCD count
CM3 Count direction
     0 = Count down
     1 = Count up
CM0-CM2 Output
     control
     000 = Inactive,
           output low
     001 = Active high
           TC pulse
     010 = TC Toggle
```

CM13 - CM15 and CM7: Used to control gating of the counters. Gating is primarily used with five GATE inputs to the 9513 chip which are not connected to anything on the CPU Support card. For information on the GATE inputs and the various gating modes, consult the AM9513 Data Sheet.

CM12: Count source polarity. Selects which edge of the clock the counter counts on. Rising edge should be used for concatenating counters using TCN-1 source.

CM8 - CM11: Count source. Selects one of seven sources. See the section on frequency sources.

CM7: See CM13 - CM15.

CM6: Reload source. If the load register is chosen for the reload source, the counter will be reloaded from the load register on Terminal Count. If the reload source is load or hold the source alternates between load and hold on Terminal Count. Very fine duty cycle control can be acheived in this way using toggled outputs.

CM5: Repetition. If count once is selected the counter will count one full cycle then disarm. A cycle can consist of one or two Terminal Count cycles depending on CM6. If the reload from Load mode is chosen there will be only one Terminal Count cycle. If the the reload from Load or Hold mode is chosen there will be two Terminal Count cycles. The first will count down to Terminal Count, reload from either Load or Hold, count down to Terminal Count again and reload from the other of the two. If count repetitively is chosen, the counter will automatically reload at Terminal Count and keep counting.

CM4: Count base. Either binary or BCD counting can be selected.

CM3: Count direction. Counting down is handy for frequency dividers while counting up is good for timers.

CM0 - CM2: Output control. These bits control the action of the output pin for each counter. The inactive mode simply turns the output off so that a logic zero is present at the output pin. The TC toggle mode toggles the output each time the counter reaches Terminal Count. In active high TC pulse mode the output is normally low and pulses high during Terminal Count. In active low TC pulse mode the output is normally high and pulses low during Terminal Count.

OUT2 and 4: Connected to the Slave 8259A programmable interrupt controller. In general the TC toggle mode is the most useful for interrupt generation since the output will stay active until it is turned off (or till the next TC pulse). The procedure is simply to turn off the output, load and arm the counter, and turn off the counter output after the interrupt has occured.

This mode works equally well with either the level or edge triggered input modes of the 8259A interrupt controllers. If the timer is set to count repeatedly to generate interrupts at a certain frequency then turning the output off after each interrupt is all that is required to initialize for the next cycle. The output will toggle high at the next TC pulse.

If interrupts are not used the inactive output mode should be used or the interrupt should be masked at the Slave 8259A interrupt controller. If the interrupt request is masked at the 8259A but the counter output is active then the logic level of the output pin can be read using the status port.

OUT5: Baud rate generator for the serial port. The TC toggle mode should be chosen to keep the duty cycle close to 50%.

Alarm Registers

Only counters 1 and 2 have these devices. When a comparator for a counter is enabled, the counter's output is driven by the comparator. The output will be active whenever the count is equal to the contents of Alarm register. Counter 1's output is not connected to anything but the Alarm register and comparator can still be used because the output level can be read through the status port. When counter 2's comparator is enabled its output interrupt request pin is driven by the comparator. The operation of comparator 2 is changed under Time Of Day mode. See the section on "Time Of Day" below.

The comparator output can be active high or low. The selection is made by choosing either the active high or active low TC pulse output mode in the Counter Mode Register. This only selects the output polarity, the actual output source is the comparator.

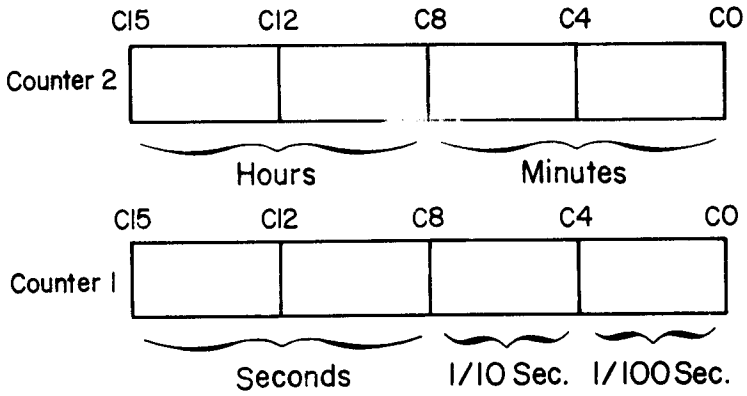
If fast counting rates are used the length of the interrupt request output pulse from comparator 2 could be quite short. If the interrupts are disabled at the CPU or a higher priority interrupt request is in service the Alarm interrupt might be missed. One solution to this problem is to use the active low output mode for the counter and the Edge Triggered Input Mode with the Slave 8259A interrupt controller. The interrupt request will now happen at the trailing (positive going) edge of the output pulse which happens one clock period after the time set in the Alarm register. To correct this, set the Alarm register to interrupt one count earlier than when the interrupt is actually desired. If the CPU is certain to get the interrupt then the method described above is not necessary. Use the active high output and set the actual desired interrupt count in the Alarm register.

Time of Day

Time of day counting is controlled by master mode register bits MM0 and MM1. When these two bits are 00, counters 1 and 2 operate normally. When these bits are 11, counters 1 and 2 operate in the time-of-day mode. Time-of-day mode enables special counting modes in counters 1 and 2 which count hours, minutes, seconds, tenths of seconds, and hundredths of seconds rather than the standard BCD or binary.

FOUT should be programmed to provide 100Hz, the count source for counter 1 should be FOUT, and the count source for counter 2 should be TCN-1 so that the minutes and hours counter in counter 2 toggles from the seconds output from counter one. A complete list of the proper master mode register and counter mode register bit requirements for time of day operation is given below.

The comparators also operate differently in time of day mode. Counter 2's comparator is gated by counter 1's comparator so that a full 32-bit comparison is required to activate OUT2. The time of day at which the interrupt is desired is loaded into alarm registers 1 and 2. When that time arrives OUT2 will go active if both comparators are enabled and an interrupt request will be sent to the 8259A.



Time-of-Day Mode

Example: Master Mode Register for Time Of Day.

Scaler uses BCD division.
FOUT divider divides by four.
FOUT source is F5.
Time Of Day enabled.

A typical master mode might be: 84F3 hex.
This enables data pointer sequencing and disables the comparators in addition to those functions mentioned above.

Example: Counter Mode Register 1 for Time Of Day

Count source is FOUT.
Reload from Load.
Count repetitively.
Use BCD counting.
Count up.

A typical Counter 1 Mode might be: 0138 hex.
This counts on rising edges and sets the output inactive.

Example: Counter Mode Register 2 for Time Of Day

Count on rising edge.
Count source is TCN-1.
CM3 - CM6 are the same as for Counter Mode Register 1.

A typical Counter 2 mode might be: 0038 hex.
This sets the output inactive (no interrupts generated).

SETTING THE CLOCK

There are four steps involved in setting the time of day counters to the correct time of day:

1. Initialize counters 1 and 2 to a value of all zeros. This is done by setting load registers 1 and 2 to 0000 hex and transfering their contents into counters 1 and 2. This conditions the time of day count circuitry and must follow the setting of the Master Mode register and Counter Mode registers. Note that the Master Mode register contents may be changed after this providing MM0 and MM1 are not altered; if these are changed, the Time of Day circuitry must be reconditioned.

2. Set Load registers 1 and 2 to the desired time and transfer this into the counters. Make sure that the time loaded does not set any of the decades to an illogical value. In particular, no decade should be set to A hex through F hex; the tens of minutes and tens of seconds should be between 0 and 5 inclusive; the hours should be between 0 and 23 decimal; and the hundredths of seconds decade should be 0. Note that this is a 24 hour clock so that PM times must have a 12 hour bias added.

3. Set load registers 1 and 2 to all 0s to insure that the counters roll over to the correct time at Terminal Count.

4. Start the counters at the desired time by writing the Arm counters 1 and 2 command (23 hex) to the command register (BASE+5).

READING THE CLOCK

You may read the current time whenever desired since the reading operation does not affect the counter operation. The time is read by first issuing the Save counters 1 and 2 command (A3 hex) to the command port (BASE+5). This causes the contents of counters 1 and 2 to be transfered into the hold register.

The value read from counter 2 may be in error if counter 1 had just rolled over but the Terminal Count pulse had not yet incremented counter two. To detect and correct this problem examine the contents of hold register one. If the count is zero then the value in hold register two may be in error. The command to Save counter 2 (A2 hex) should be sent to the command register (BASE+5). By the time this command is issued, counter 2's value will have settled.

SETTING THE ALARM

When both comparators 1 and 2 and Time Of Day are enabled, the operation of comparator 2 is conditioned by comparator 1 so that a full-32 bit compare must be true before the output goes active. If comparator 1 is disabled, then comparator 2 operates in its standard fashion and only checks the hours and minutes. As described in the section on the Alarm registers and comparators, the interrupt request will only be a short pulse (ten milliseconds) if both comparators are enabled due to the action of the comparators.

If there is a possiblity that the interrupts will be disabled during the time the Alarm interrupt request occurs then the active low output mode should be used with Counter Mode register 2 and the Edge Triggered Input Mode should be used with the Slave 8259A interrupt controller. Since the interrupt request now occurs on the trailing edge of the pulse it will be one one-hundredth of a second late. If that is a problem, set the alarm time for one one-hundredth of a second earlier than the actual time the interrupt is desired.

When changing the alarm register values, the output of counter 2 should be deactivated or should be masked at the Slave 8259A interrupt controller because spurrious interrupts may be generated as the alarm register values change.

NON-INTERRUPTIBLE POWER SUPPLY

If the time of day clock is used it is desirable to have a non-interruptable power supply for the 9513 so that the time of day will not have to be reset each time the power is turned on. Provision is made on the CPU Support card for doing this.
The simplest arrangement is to have a non-switched line-powered supply to provide power for the 9513. The supply must provide either +5 volts regulated at 200 mA or +8 volts unregulated at 200 mA and should be connected to the EXT PWR connector using GND and +5 or +8 as appropriate. The 9513 PWR jumper should be set to EXT since power is coming from an external source.

Note that if the power fails the time-of-day will be lost. Hopefully that will not happen too often and will not be a problem. If it is, some sort of battery back-up should be used. If the battery supplies +5 volts then the 9513 PWR jumper may be connected to INT and the +8 pin of the EXT PWR connector will supply +8 volts from the S-100 bus when power is on to charge the battery. Due to the special regulator circuit used, no power will flow back into the on-board regulator from the external +5 volt source and the computer will supply power when it is on. The battery should not be directly connected to the +5 pin of the EXT PWR connector but instead should use a diode to keep current from flowing from the on-board regulator into the battery when power is on. The 9513 PWR jumper must not be connected to INT if the external supply provides +8 volts since the +8 pin is directly connected to the S-100 +8 volt supply.

Special procedures should be followed to prevent the power going on and off and reinitialization of the 9513 from disturbing the Time Of Day counters. Whenever the CPU is finished accessing the 9513 the data pointer register should be set to read the status register. This prevents glitches on the RD, WR, and CS lines as power goes off and on from disturbing the other registers. When power is turned on, the software which initializes the 9513 must not disable the Time Of Day mode in the Master Mode Register.

CHAPTER 6. SERIAL INPUT/OUTPUT

The CPU Support card has two serial I/O devices for RS-232 communication. The heart of the serial I/O device are two 8251A UARTs. The 8251As occupy six of the CPU Support board's sixteen I/O ports.

Port	Function
BASE+6	Data port for serial port 1
BASE+7	Control/status port for serial port 1
BASE+8	Data port for serial port 2
BASE+9	Status/control port for serial port 2
BASE+10	Baud rate port for serial port 1
BASE+11	Baud rate port for serial port 2

The baud rate is determined by the low four bits of the byte sent to the baud rate port.

Bits 0-3 (hex)	Baud rate
0	50
1	75
2	110
3	134.5
4	150
5	300
6	600
7	1200
8	1800
9	2000
A	2400
B	3600
C	4800
D	7200
E	9600
F	19200

Before an 8251A can be used, it must be initialized by sending the desired operating parameters to the control/status port (BASE+7 for serial port; BASE+9 for serial port 2). A total of four bytes are required; the first two reset the 8251A so that it can accept the Mode and Command instructions. The first write to the control/status

port after reset programs the Mode instruction. All subsequent writes to the control/status port program Command instructions.

Byte 1 - 10110111 (B7 hex)
Byte 2 - 01110111 (77 hex)
Byte 3 - Mode instruction.
Byte 4 - Command instruction.

MODE INSTRUCTION DEFINITION

The Mode instruction is summarized below:

M7	M6	M5	M4	M3	M2	M1	M0
S1	S0	PE/O	PEN	L1	L0	B1	B0

M6-M7 S0-S1 number of transmitter stop bits
- 00 = Invalid
- 01 = 1 stop bit
- 10 = 1.5 stop bits
- 11 = 2 stop bits

M5 PE/O Parity Even/Odd
- 0 = Odd parity
- 1 = Even parity

M4 PEN Parity Enable
- 0 = No parity bit
- 1 = Parity bit enabled

M2-M3 L0-L1 character length (number of data bits)
- 00 = 5 bits
- 01 = 6 bits
- 10 = 7 bits
- 11 = 8 bits

M0-M1 B0-B1 baud rate factor
- 00 = Sync mode, see an 8251A data sheet.
- 01 = 1 X
- 10 = 16 X
- 11 = 64 X

S0-S1: Programs the number of stop bits sent by the transmitter. The receiver never requires more than one stop bit.

PE/O: Selects even or odd parity.

PEN: Selects whether or not a parity bit is transmitted by the transmitter and expected by the receiver. If parity is enabled, even or odd parity is selected by PE/O. If no parity is selected, PE/O has no effect.

L0-L1: Selects the number of data bits transmitted by the transmitter and expected by the receiver. The start, stop, and parity bit (if any) are not counted in the number of data bits.

B0-B1: The baud rate factor selects how many clocks per transmitted or received bit. 16 X is almost universally used but 64 X provides a simple way to switch between two baud rates a factor of four apart (1200 and 300 for example). The baud rate divisors given in the table of Load register values assume 16 X baud rate factor. 1 X should only be used for transmission because slight differences in baud rate between two devices can cause serious reception errors if 1 X is used. B0-B1 = 00 selects the synchronous mode of operation, see an 8251A data sheet for more information.

COMMAND INSTRUCTION DEFINITION

The Command instruction is summarized below:

C7	C6	C5	C4	C3	C2	C1	C0
X	IR	DSR	ER	SBRK	RxE	CTS	TxE

C7: X = don't care.
This bit is used with the sync mode. See an 8251A data sheet.

C6: IR Internal Reset
0 = No operation
1 = Reset the 8251A.

C5: DSR Data Set Ready
0 = DSR line inactive (-12 volts)
1 = DSR line active (+12 volts)

C4: ER Error Reset
0 = No operation
1 = Reset error flags (PE, OE, FE)

C3: SBRK Send BReaK
0 = Normal operation
1 = Send break (continuous space [+12 volts] on RxD line)

C2: RxE Receiver Enable
0 = disable receiver
1 = enable receiver

C1: CTS Clear To Send
0 = CTS line inactive (-12 volts)
1 = CTS line active (+12 volts)

C0: TxE Transmitter Enable
0 = disable transmitter
1 = enable transmitter

IR = 1 puts the 8251A into an "idle" mode waiting for a new Mode instruction. After reset the 8251A must be reinitialized before it can be used.

DSR: Controls the level on the DSR output line. The 8251A data sheet defines this bit as RTS. Since the CPU Support card was designed to be used in a computer (the

equivalent of a modem from the terminal's point of view) the RTS line is an input. Therefore this output was redefined as DSR and connected to the DSR line.

ER: Resets the three error flags PE, OE, and FE.

SBRK: Sends a continuous space level (+12 volts) on the RxD (Receiver Data) line. Note that "receiver" is from the terminal's point of view. In spite of the name RxD, this is an output line.

RxE = 0 turns off the receiver. In the off state the RxRDY status bit is inhibited as well as the RxRDY Interupt Request to the Slave 8259A.

CTS: Controls the level on the CTS output line. The 8251A data sheet defines this bit as DTR. Since the CPU Support card was designed to be used in a computer (the equivalent of a modem from the terminal's point of view) the DTR line is an input. Therefore this output was redefined as CTS and connected to the CTS line.

TxE = 0 turns off the transmitter. In the off state the TxRDY Interupt Request to the Slave 8259A is inhibited immediately but the transmitter will still accept a character to fill its buffer although it will not transmit it until the transmitter is turned back on.

STATUS READ DEFINITION

The status of the 8251A can be read from the control/status port (BASE+7 for serial port 1 and BASE+2 for serial port 2). The various bits indicate the condition on the receiver, transmitter, and error conditions.

The Status word is summarized below.

S7	S6	S5	S4	S3	S2	S1	S0
RTS	BD	FE	OE	PE	TxE	RxRDY	TxRDY

S7 RTS Request To Send
0 = RTS line is inactive (-12 volts)
1 = RTS line is active (+12 volts)
S6 BD Break Detect
0 = Receiver is receiving normal data
1 = Receiver is receiving a break level (continuous space, +12 volts)
S5 FE Framing Error
0 = No error
1 = A valid stop bit was not received after the last character
S4 OE Overrun Error
0 = No error
1 = The last character overran the previous one.
S3 PE Parity Error
0 = No error
1 = The last character received had incorrect parity.
S2 TxE Transmitter Empty
0 = Transmitter not empty
1 = Transmitter has sent all characters out
S1 RxRDY Receiver ReaDY
0 = No character has been received
1 = Receiver has a character to read
S0 TxRDY Transmitter ReaDY
0 = Transmitter is not ready to accept a character
1 = Transmitter can accept a character

RTS: Indicates the level on the RTS input line. The 8251A data sheet defines this bit as DSR. Since the CPU Support card was designed to be used in a computer (the equivalent of a modem from the terminal's point of view) the DSR line is an output. Therefore this input was redefined as RTS and connected to the RTS line.

BD: Is high when the 8251A detects a continuous break level (+12 volts) on TxD. Note that "transmitter" is from the terminal's point of view. In spite of the name TxD, this is an input line.

FE: Indicates framing error. If this bit is high, the receiver did not detect a valid stop bit after the data and parity bits.

OE: Indicates overrun error. If this bit is high, the CPU did not read the character received before the last character.

PE: Indicates parity error. If this bit is high, parity of the last character did not match its parity bit.

TxE = 1 indicates the transmitter is empty. It has sent all the data given to it.

RxRDY = 1 indicates the receiver has a character to read.

TxRDY = 1 indicates the transmitter is ready to accept a character for transmission.

DTR INPUT

The DTR input cannot be read as part of the status as the RTS input can. Instead, the DTR input has direct control over the operation of the transmitter. It has the same effect as the Transmitter Enable bit of the Command instruction.

When this line goes inactive (-12 volts) the transmitter will stop transmitting after it transmits the contents of its buffer. The transmitter will accept a character to fill its buffer although it will not transmit it till DTR goes active once more. The 8251A data sheet defines this input as CTS. Since the CPU Support card was designed to be used in a computer (the equivalent of a modem from the terminal's point of view) the CTS line is an output. Therefore this input was redefined as DTR and connected to the DTR line.

WRITE RESTRICTIONS

1. Data may only be written to the 8251A when TxRDY = 1.

2. Mode and Command instructions require 4uS recovery time between writes. This could be a problem because inline code can possibly output faster than this.

```
MOV     AL,0B7H       ; 4 cycles
OUTB    BASE+7        ; 10 cycles      14 cycles total
```

For example, the MOV and OUTB instructions shown require 14 clock cycles for execution minus one cycle for the write pulse width gives a total time between writes of 13 cycles or 1.625uS for an 8MHz 8086 with 16 bit memory. We therefore need to add 2.375uS or 19 more clocks to meet the requirement.

Any instruction requiring 19 or more clocks could be placed between writes to provide the delay. The only one byte instruction which meets this requirement is CMPB which takes 22 clocks. This instruction could cause problems if the SI, DI, DS, and ES registers are not properly set because it would do two random address reads. If two bytes are allowed a register PUSH and POP takes 19 clocks but requires a valid stack.

Possibly the least offensive instruction is an AAD instruction. This requires two bytes and only modifies the AX register. AAD requires 60 clocks which is a bit excessive but does meet the requirement. With a 4MHz 8086 only 3 clocks are required. A NOP fills the requirement nicely. If you are ever planning to upgrade to an 8MHz 8086 it is a good idea to make the software 8MHz compatible so it will not have to be modified in the future. Possibly the best solution is to use a loop to initialize:

```
            MOV       CX,4
            MOV       SI,INITABLE
INITLOOP:   SEG       CS
            LODB
            OUTB      BASE+7
            LOOP      INITLOOP
```

The actual loop takes 41 clocks (5.125uS @ 8MHz) which solves the write recovery time requirement. The whole routine requires only 16 bytes including the table compared to 22 bytes for inline code with two-byte delays.

A 4MHZ Z80 does not require any delays because 18 clocks satisfy the requirements (including two for the write pulse). Inline requires 18 clocks, seven for the load and eleven for the output. OTIR requires 21 per loop.

I/O CONNECTIONS

A ribbon cable with appropriate connectors is supplied to connect between J1 on the CPU Support card and the back panel of the computer. The orientation of the connector to J1 is such that the two triangular marks line up. The back panel connector is a standard DB25S type.

Computer Back Panel DB25S	Name	CPU Support Board J1
1	Chassis Ground (CG)	1
2	Transmit Data (TxD)	3
3	Receive Data (RxD)	5
4	Request To Send (RTS)	7
5	Clear To Send (CTS)	9
6	Data Set Ready (DSR)	11
7	Signal Ground (SG)	13
20	Data Terminal Ready (DTR)	14
	no connection	26

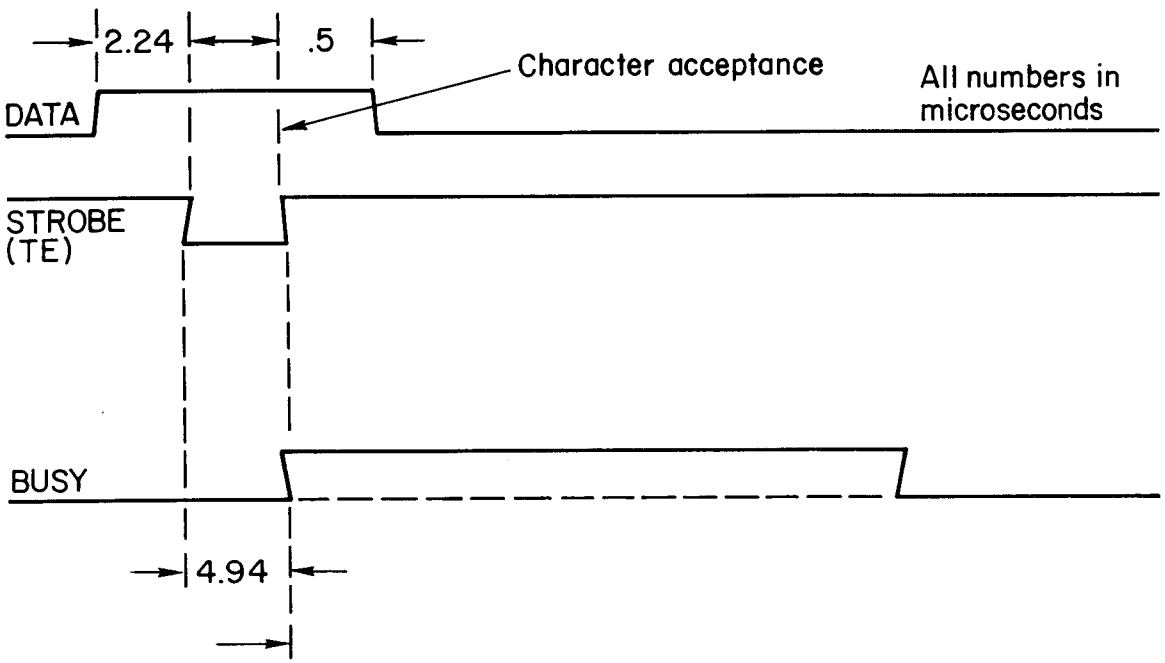
7. PARALLEL OUTPUT

The parallel output port communicates with the CPU through two ports:

Port	Function
BASE+12 (output) BASE+13 (input)	Output (parallel port data/control) Input (parallel port status)

TIMING

A timing diagram is shown below:



OUTPUT CONNECTIONS

Connections to the parallel output port are made through connector J3. J3 is designed to be connected to a 25-pin ribbon cable terminated with a DB25P connector.

Parallel Output

Pin assignments are shown below:

CPU Support Board J3	Function	Computer Back Panel DB25P
1	STROBE	1
2	Ground	14
3	DATA 0	2
4	Ground	15
5	DATA 1	3
6	Ground	16
7	DATA 2	4
8	Ground	17
9	DATA 3	5
10	Ground	23
11	DATA 4	6
12	Ground	24
13	DATA 5	7
14	Ground	25
15	DATA 6	8
16	Ground	18
17	DATA 7	9
18	Ground	19
19	Ground	10
20	Ground	20
21	BUSY	11
22	Ground	21
23	PAPER OUT	12
24	Ground	22
25	Ground	13
26	Ground	

TERMINATION

Input Drive Requirements - Terminated

VIH = 2.0 volts, no current requirement due to termination.
VIL = 0.8 volts @ 11.4 mA (sink)

Input Drive Requirements - Not Terminated

VIH = 2.0 volts @ 20 µA (source)
VIL = 0.8 volts @ 0.4 mA (sink)

Output Drive Capability

730VOH = 2.4 volts @ 2.6 mA (source)
VOL = 0.5 volts @ 24 mA (sink)
= 0.4 volts @ 12 mA (sink)

APPENDIX B. SPECIFICATIONS

IEEE-696 Standard (S-100)	Fully meets standard except as follows: This board requires status lines to be valid earlier than the IEEE Standard requires. This characteristic may cause problems with some Z80 systems. The board may be modified so it does not require early status.
EPROM Address	F000 hex or F800 hex in systems with 16-bit addressing; FF000 hex in systems with 20-bit addressing. Switch and pin-shunt selectable.
Timers	Five 16-bit programmable timers, two of which may be combined for time-of-day, three for general purpose, or all five for general purpose.
Interrupt Controller	Two 8259A chips. Both 8080 mode (8080, 8085, Z80 - use three byte call interrupt acknowledge) and 8086 mode. With programmable priorities and priority rotation, masking. Seven interrupts generated on board; accepts eight from bus.
I/O Parallel Port	8-bit Centronics-compatible parallel output port.
I/O Serial Port	Uses 8251A type chips. Two RS-232 serial ports with individually programmable baud rates, from 50 to 19200 baud. For console I/O port, 8086 Monitor provides autoselection of 19200, 9600, 1200, 300, 150, or 110 baud.
Noise Margins	All signal inputs to the board have minimum of 0.4V hysteresis at 25°C.

Specifications

Power Requirements	+8V at 1.0 Amp (including timer chip), +16V at less than 0.1 Amp, -16V at less than 0.1 Amp. Timer chip has separate power connections which may be either +8 or +5V at 0.2 Amp.
Reliability	This product has been in the field for approximately two years. Failure rates have been running less than 2% per year.
Operating Environment	0°C to 70°C.

Cut along line

Comment Form

Use this form to comment on this product and to report errors in the manual.

System Description:

CPU: Manufacturer _____ Model No. _____

Memory: Manufacturer _____ Model No. _____

Memory Size _____

Disk Controller: Manufacturer _____ Model No. _____

I/O Interfaces: Manufacturer _____ Model No. _____

Other _____

_____

_____

Comments:

Name _____ Company _____

Street _____

City _____ State _____ Zip _____

Phone _____ Date _____

Cut along line

Fold and tape

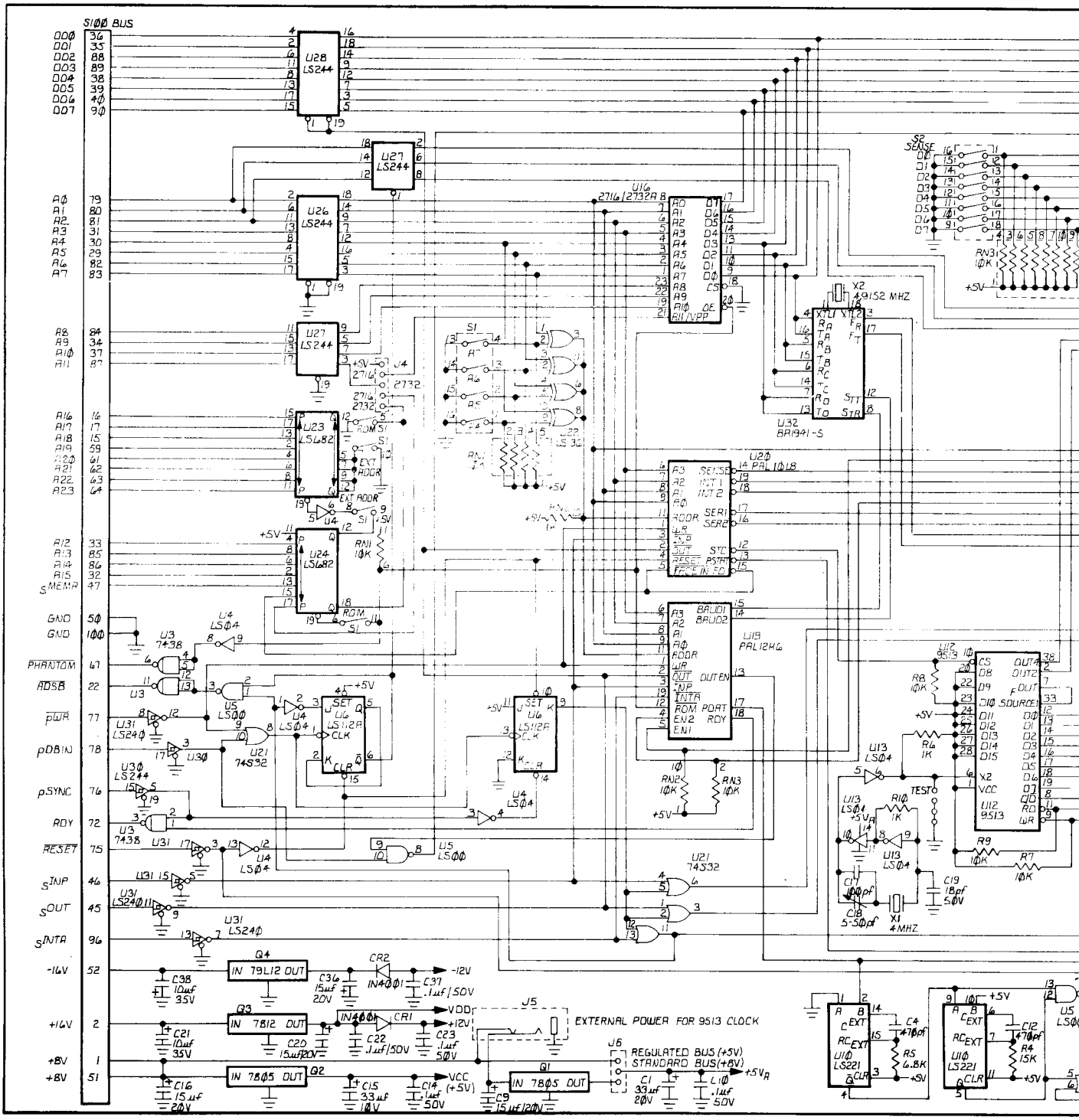
Place stamp here

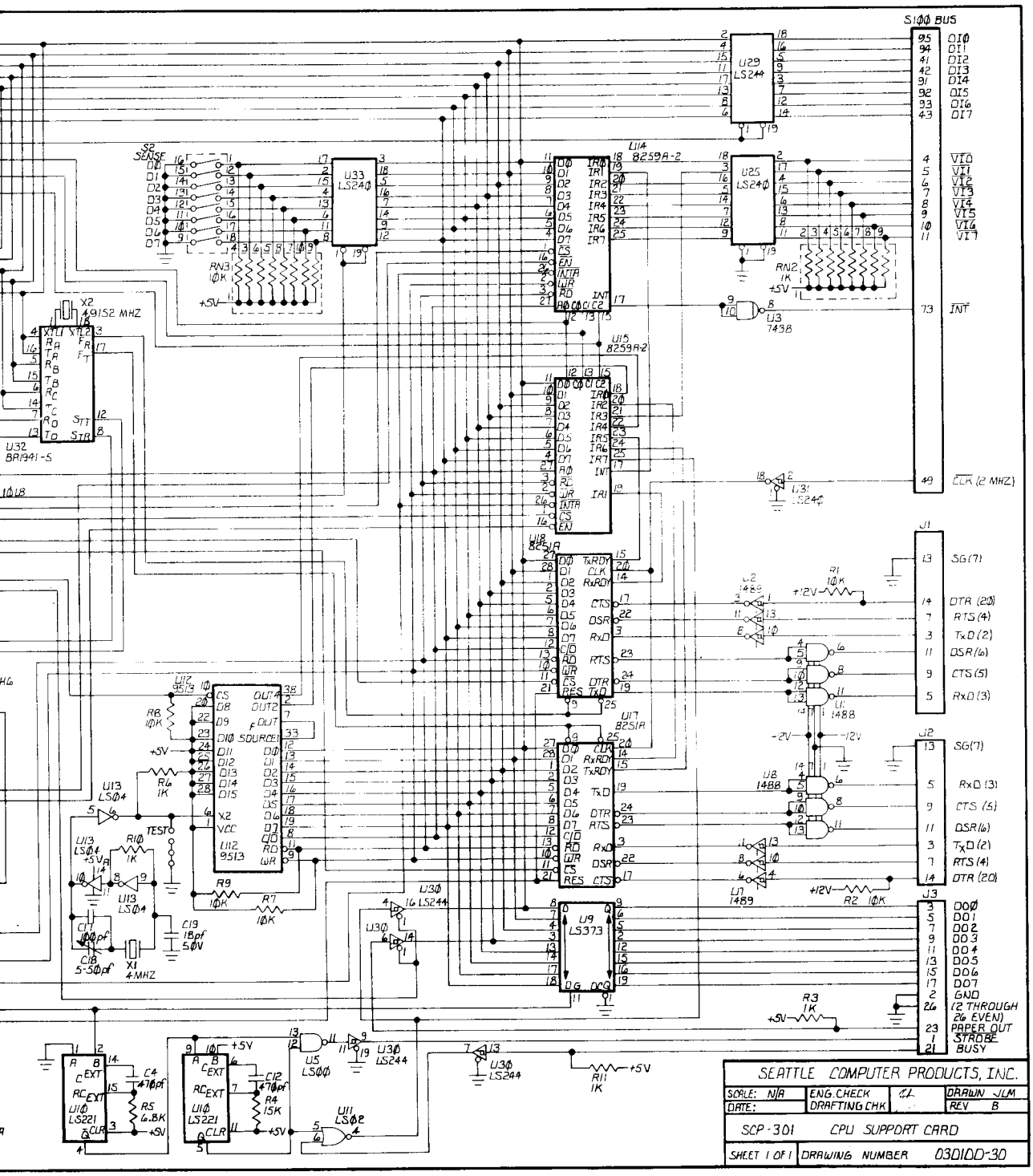
Seattle Computer Products
1114 Industry Drive
Seattle, WA 98188

Attention: Hardware Support

Fold and tape

APPENDIX A. CPU SUPPORT CARD SCHEMATIC




S100 BUS
U29 LS244
U25 LS240
U33 LS240
U14 8259A-2
U15 8259A-2
U18 8251A
U17 8251A
U12 9513
U9 LS373
U30 LS244
U31 LS240
U32 BR1941-S
X2 4.9152 MHZ
X1 4 MHZ
U13 LS04
U5 LS00
U11 LS02
U10 LS221
U7 1489
U8 1488
U1 1488
U2
U3
SENSE
TEST
SCP-301 CPU SUPPORT CARD
SEATTLE COMPUTER PRODUCTS, INC.
SHEET 1 OF 1
DRAWING NUMBER 03010D-30

90-DAY LIMITED WARRANTY

WARRANTY AND WARRANTY PERIOD

When sold by Seattle Computer Products (hereinafter referred to as SCP) or through an authorized SCP dealer, this product is warranted to the original purchaser and all subsequent owners of the product for a period of 90 days from the time the product is first sold at retail and for such additional time as the product may be out of the owner's possession for the purpose of receiving warranty service at the factory. When sold to the end-user by an OEM, the warranty terms vary. Consult your OEM for specific warranty coverage.

WARRANTY COVERAGE

This product is warranted to be free from defects of material and workmanship and to perform within its specifications as detailed in the instruction or operating manual during the period of the warranty. This warranty does not cover damage and is void if the product has been damaged by neglect, accident, unreasonable use, improper repair, or other causes not arising out of defects in material or workmanship.

WARRANTY PERFORMANCE

During the warranty period, SCP will repair or replace defective boards or products or components of boards or products upon written notice that a defect exists. Certain high value parts may have to be returned to SCP prior to replacement. Other components will be replaced without the part having to be returned to the factory with the exception that SCP retains the right in all cases to examine the defective board or other products prior to the item's replacement under the warranty. In the event the return of the board, product, or component is requested by SCP under this warranty, the owner shall ship the item prepaid to the SCP factory. SCP will pay for shipment of replacement items back to the owner. All repairs or replacements under this warranty will be performed by SCP within five working days of receipt of notice of defect or return of components as called for under this warranty.

WARRANTY DISCLAIMERS

While high reliability was a major design factor for this product and care was used in its manufacture, no certainty can be achieved that any particular product will operate correctly for any specific time. No representation is made by SCP that this product will not fail in normal use. Because of the inability to guarantee 100% reliability, SCP shall not be liable for any consequential damage the user may suffer because the product fails to function reliably 100% of the time. Any implied warranties arising from the sale of this product are limited in duration to the warranty period defined above.

LEGAL REMEDIES

This warranty gives the purchaser specific legal rights. He may have additional rights which vary from state to state.

SHIPPING INSTRUCTIONS

In the event it becomes necessary to return the product or component to SCP:

1. Telephone (206) 575-1830 to obtain an RMA number. No products will accepted by SCP without an RMA number.

2. Package the product in a crushproof container with adequate packing material to prevent damage.

3. Return the product along with proof of purchase and a written explanation of the difficulty encountered along with your name, address, and phone number. Send the product to Seattle Computer Products, 1114 Industry Drive, Seattle WA 98188.