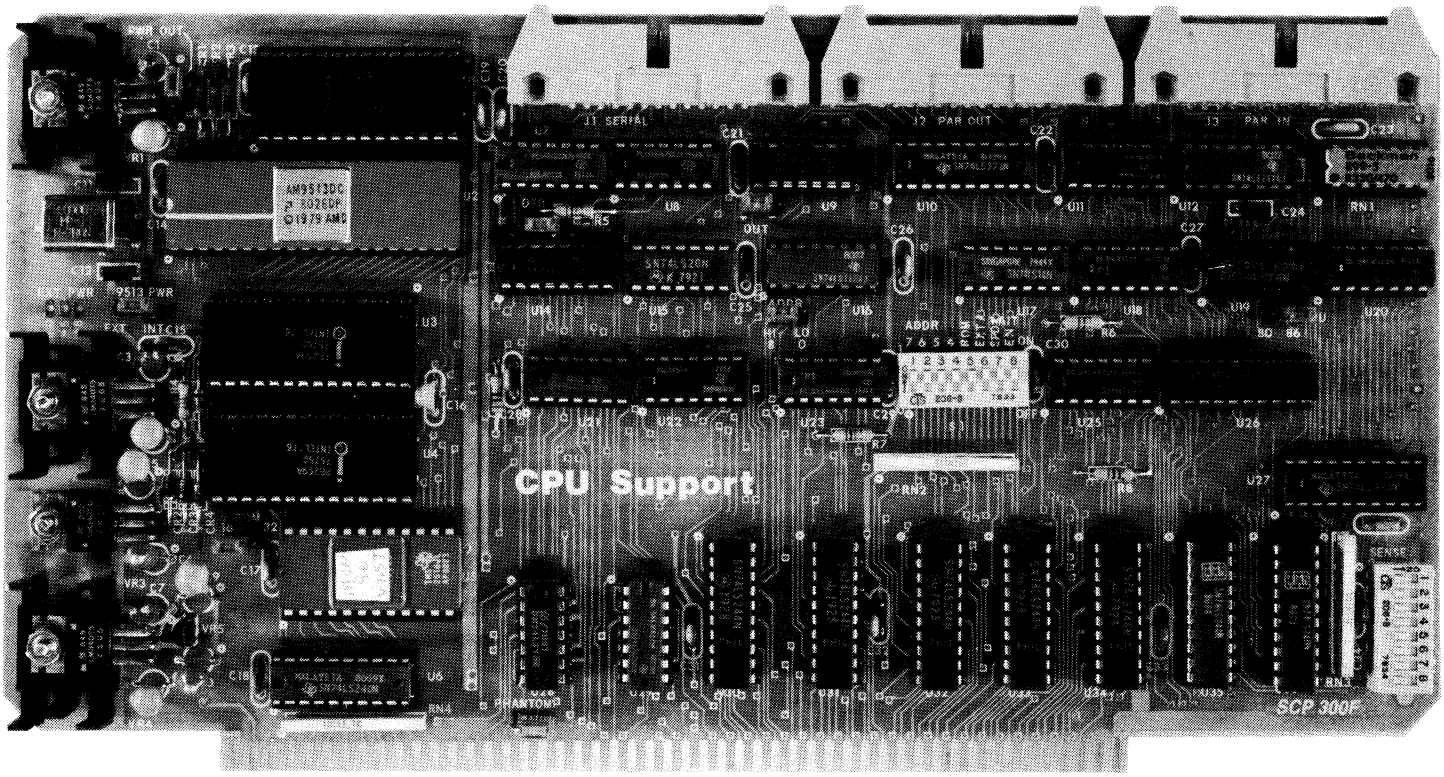
Instruction Manual

Model SCP300F



CPU Support Board

Seattle Computer Products, Inc.
1114 Industry Drive, Seattle, WA. 98188
(206) 575-1830

Rev. F

TABLE OF CONTENTS

Features

This card provides a collection of functions important to a computer system. Included are:

1. A ROM monitor (optional). The ROM may be either an Intel-type 2716 or an Intel-type 2732. It resides near the end of memory, as determined by either the usual 16 address lines, or by 20 address lines (extended addressing). It can drive PHANTOM to allow a memory board to overlap, and it can be disabled with an output command.

2. A serial communication port. Includes RS-232 drivers and full handshaking. Software-controlled baud rate generator allows almost any conceivable data rate.

3. A parallel input port. Latched and with full handshaking, this port can provide +5V or +8V to power an external device (such as a keyboard).

4. A parallel output port. Full handshaking is provided with an output that may be statically latched and driving at all times or configured for 3-state output which allows bidirectional operation.

5. A vectored interrupt controller. Provides 15 level of vectored interrupts, expandable to 64 through "slave" controllers on other cards (such as our Multiport Serial card). Includes complete interrupt support for 8080, 8085, Z80, and 8086 microprocessors.

6. Two general-purpose timer/counters. Each is 16 bits wide. One may be combined with the time-of-day clock to count days.

7. A time-of-of day clock. Time of day is kept in 24-hour format to 0.01 second. Power to run the clock may be provided from an external source, such as a battery, to keep the clock running when computer power is off. If time of day is not needed, the clock may be used as two general-purpose 16-bit timers.

8. A "sense switch" input port. Simply allows reading the setting of an 8-position DIP switch for the purpose of setting software options or configuration information.

Each of these functions are controlled by a block of 16 I/O ports. The upper 4 bits of the 8-bit I/O address are compared with a switch setting to enable the board as a whole; the lower 4 bits select the particular function on the board.

Summary of Switches and Jumpers

S1

Positions 1, 2, 3, and 4 of S1 set the highest 4 bits of the I/O port address (the low 4 bits select a particular function on the board). Position 1 sets A7, position 2 sets A6, 3 sets A5, and 4 sets A4. CLOSED represents a one, OPEN a zero.

Position 5 enables or disables the ROM socket. Closed enables the ROM, open disables it.

Position 6 controls extended addressing of the ROM. When closed 20 address bits are used and the ROM resides at either FF000H or FF800H depending on Jumper ADD and Jumper ROM. When open, only 16 address bits are used, and the ROM resides at F000H or F800H.

Positions 7 and 8 control the wait state generator. To disable any wait states, open position 8 (position 7 doesn't matter). To always insert one wait state for both memory and I/O,

close position 8 and open position 7. In systems in which pin 98 of the S-100 bus indicates processor speed, such as the SCP 8086 CPU board, closing both positions 7 and 8 will enable a wait state only when the processor is at high speed. Generally, wait states are not needed in 8080 and Z80 systems; SCP recommends a wait state only when used with our 8086 CPU at 8 MHz.

S2

The 8 positions of S2 can be read by the CPU by inputting from port BASE+15. (BASE is the I/O address set on S1.) Positions 1 through 8 are read in bits 0 through 7, respectively. CLOSED will be read as a one, OPEN as a zero. The SCP 8086 Monitor will automatically boot the disk immediately after baud rate selection if bit 0 is a one.

JUMPER PWR OUT - Four pins of the parallel input connector J3 are used to provide power to the input device. Jumpering PWR OUT to +8 selects unregulated 8V directly from the S-100 bus as the power supply; PWR OUT to +5 selects regulated 5V at 500mA.

JUMPER DTR - Serial output data is enabled only when Data Terminal Ready, pin 20 of the DB-25, is >3V. Should this handshaking line be left open, Jumper DTR controls whether or not the serial tranmitter will be allowed to operate: jumper DTR to "+" to enable the transmitter when DTR is open (for 3-wire communication); jumper DTR to "-" to disable it (system would wait until all connections are complete and DTR is no longer open).

JUMPER OUT - Jumpering OUT to "+" causes parallel output data to remain driving at all times. OUT to "-" causes output data to go high impedance when ACK is low; data is enabled when ACK is high or open.

JUMPER ADDR - Jumpering ADDR to HI sets the ROM address for 2716 type ROMs to F800 hex or FF800 hex, depending on the extended address enable switch, S2-6. ADDR to LO sets the ROM address to F000 hex or FF000 hex. If a 2732 type ROM is used, The jumpering block should be set so that none of the pins are connected. This can be accomplished by setting one hole in the block over an end pin and having the other hole hang out in space. A 2732 always resides at F000 hex through FFFF hex or FF000 hex through FFFFF hex.

JUMPER ROM - Jumpering ROM to 16 selects an Intel-type (single supply) 2716 (including the TMS2516) EPROM for the ROM. The address of the ROM is selected using jumper ADDR as above. Jumpering ROM to 32 selectes an Intel-type 2732 EPROM. If the 2732 type EPROM is used, none of the pins on jumper ADDR should be connected together.

JUMPER CPU - Jumpering CPU to 80 selects the three byte 8080 type cycle and jumpering CPU to 86 selects the two byte 8086 type cycle. The CPU jumper can also be placed so that it doesn't connect any of the pins together. In this position the CPU Support card does not take over the address or status bus during interrupt acknowledge and assumes the CPU will provide correct status during all bytes of the interrupt acknowledge sequence. This mode is the same as the older Rev. C board. Note that interrupt priority cascading using the address bus isn't possible in this mode and boards such as the SCP Multiport Serial card can't slave their interrupt controllers to the CPU Support card.

JUMPER PHANTOM - Jumpering PHANTOM to "+" enables the CPU Support card to drive PHANTOM active whenever the ROM is selected. This is useful to create a "hole" in a RAM board for the ROM. If PHANTOM to "-" is connected, the CPU Support card doesn't drive PHANTOM when the ROM is active.

JUMPER 9513 PWR - selects the power source for the 9513 timer chip. The INT position should be used if no external power is available. In this position the 9513 gets its power from the S-100 bus and time-of-day information is lost when the power is turned off. The EXT position should be used if an external non-interruptible power supply is connected to the EXT PWR connecter to power the 9513.

EXT PWR CONNECTOR - An external +8 volt unregulated or +5 volt regulated supply can be connected to the EXT PWR connector to provide a non-interruptible supply for the 9513.

Using the Support Card with the SCP 8086 Monitor

All switches and jumpers on the board are properly configured as shipped. To check or reset them after a change:

All 8 positions of switch S1 (in the middle of the board) should be CLOSED. This selects the I/O address BASE to be F0 hex, enables the ROM, enables extended addressing, and inserts a wait state only if an 8 MHz 8086 CPU is used.

All 8 positions of switch S2 (lower right corner) should be OPEN to select the monitor or position one could be CLOSED to select auto-boot.

Jumper PWR OUT selects what type of voltage (+5 regulated or +8 unregulated) is supplied to the parallel input connector to power a keyboard or other device. The setting of PWR OUT is irrelevent to the operation of the 8086 Monitor.

Jumper DTR should be in the "+" position to allow the monitor to run with a three wire cable to the terminal.

Jumper OUT affects the operation of the parallel output port and is irrelevent to the operation of the 8086 Monitor.

Jumper ADDR should be in the HI position to put the ROM at address FF800 hex.

Jumper ROM should be in the 16 position to use the 2716 type EPROM in which the 8086 Monitor resides.

Jumper CPU should be in the 86 position to select the two byte 8086 type interrupt acknowledge cycle.

Jumper PHANTOM selects whether or not the CPU Support card drives PHANTOM when the ROM is enabled. The position of this jumper is irrelevant to the operation of the 8086 Monitor but does depend on the configuration of your system.

Jumper 9513 PWR should be in the INT position if internal (supplied from bus) power is used to power the 9513 timer chip.

Connector EXT PWR is the external power connector for the 9513 timer chip and has no connections to it unless an external power source is used.

Connect a cable to J1 (the leftmost connector at the top of the board); the other end should have a DB-25 connector (normally female). Plug an RS-232 serial terminal into that end. The hardware is now all set; refer to the Monitor manual to use the software.

Programming and Using the CPU Support Card

I/O Ports

The CPU Support card requires a total of twelve I/O ports for communication with the CPU. These twelve ports can be set on any sixteen port boundary called the BASE. The BASE is selected using the first four positions of S1. These first four positions correspond to A7-A4 from left to right. Turning a switch on (closed) selects a one and turning it off (open) selects a zero.

S1-1	S1-2	S1-3	S1-4	I/O port address BASE
OFF	OFF	OFF	OFF	00 hex
OFF	OFF	OFF	ON	10 hex
OFF	OFF	ON	OFF	20 hex
OFF	OFF	ON	ON	30 hex
OFF	ON	OFF	OFF	40 hex
OFF	ON	OFF	ON	50 hex
OFF	ON	ON	OFF	60 hex
OFF	ON	ON	ON	70 hex
ON	OFF	OFF	OFF	80 hex
ON	OFF	OFF	ON	90 hex
ON	OFF	ON	OFF	A0 hex
ON	OFF	ON	ON	B0 hex
ON	ON	OFF	OFF	C0 hex
ON	ON	OFF	ON	D0 hex
ON	ON	ON	OFF	E0 hex
ON	ON	ON	ON	F0 hex

The I/O ports are assigned as follows:

BASE+0 & BASE+1, input & output. Master programmable interrupt controller.
BASE+2 & BASE+3, input & output. Slave programmable interupt controller.
BASE+4, input & output. System timing controller data port.
BASE+5, input & output. System timing controller status & control port.
BASE+6, input & output. Serial data port.
BASE+7, input & output. Serial I/O status & control port.
BASE+12, input & output. Parallel I/O data port.
BASE+13, input. Parallel I/O status and serial DCD.
BASE+14, input & output. EPROM disable port.
BASE+15, input. Sense switch.

Any other ports within the BASE that are not used are free for use by other devices. These are BASE+8, 9, 10, 11, 13 output, and 15 output.

Interrupt Controller

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. One of the two is 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 second 8259A on board is a Slave controller, which handles the seven vectored interrupt sources on the board plus the one non-cascadable VI line from the bus. In the table below, "IR" stands for "interrupt request".

Master IR0 - VI0 cascadable
Master IR1 - not available (used by Slave)
Master IR2 - VI2 cascadable
Master IR3 - VI3 "
Master IR4 - VI4 "
Master IR5 - VI5 "
Master IR6 - VI6 "
Master IR7 - VI7 "
Slave IR0 - OUT2 from 9513 timer
Slave IR1 - RxRDY from serial input
Slave IR2 - RxRDY from parallel input
Slave IR3 - VI1 (vectored interrupt line from S-100 bus, NOT cascadable)
Slave IR4 - OUT3 from 9513 timer
Slave IR5 - TxRDY from serial output
Slave IR6 - TxRDY from parallel output
Slave IR7 - OUT4 from 9513 timer

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

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 falls under five main categories: vectoring, priorities, triggering, status, and cascading. Each of these categories use various modes and commands. This section will explain the operation of these modes and commands. For clarity of explanation, however, the actual programming of the 8259A isn't covered in this section but 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 of an 8080 system differs from that of an 8086 system. Thus, the 8259A must be intially programmed in either a 8080 or 8086 mode of operation to insure 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 will output to the data bus the opcode for a CALL instruction and the address of the desired routine. This is in response to a 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 effects operation in the cascade mode, which will be covered later.

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 intruction which 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 will automatically insert 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 to be programmed by the user.

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 is placed on the data bus during the third INTA pulse.

8086 MODE

Upon interrupt in the 8086 mode, the 8259A will output 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. As in the 8080 mode, this set-up includes priority resolution and cascade mode operations which will be covered later. Unlike the 8080 mode, no CALL opcode is placed on the data bus.

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 5 bits of the interrupt vector byte are user definable. The lower 3 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. All of them are programmable, that is, they may be changed dynamically under software control. With these modes and commands, many possibilities are conceivable, giving the user 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.

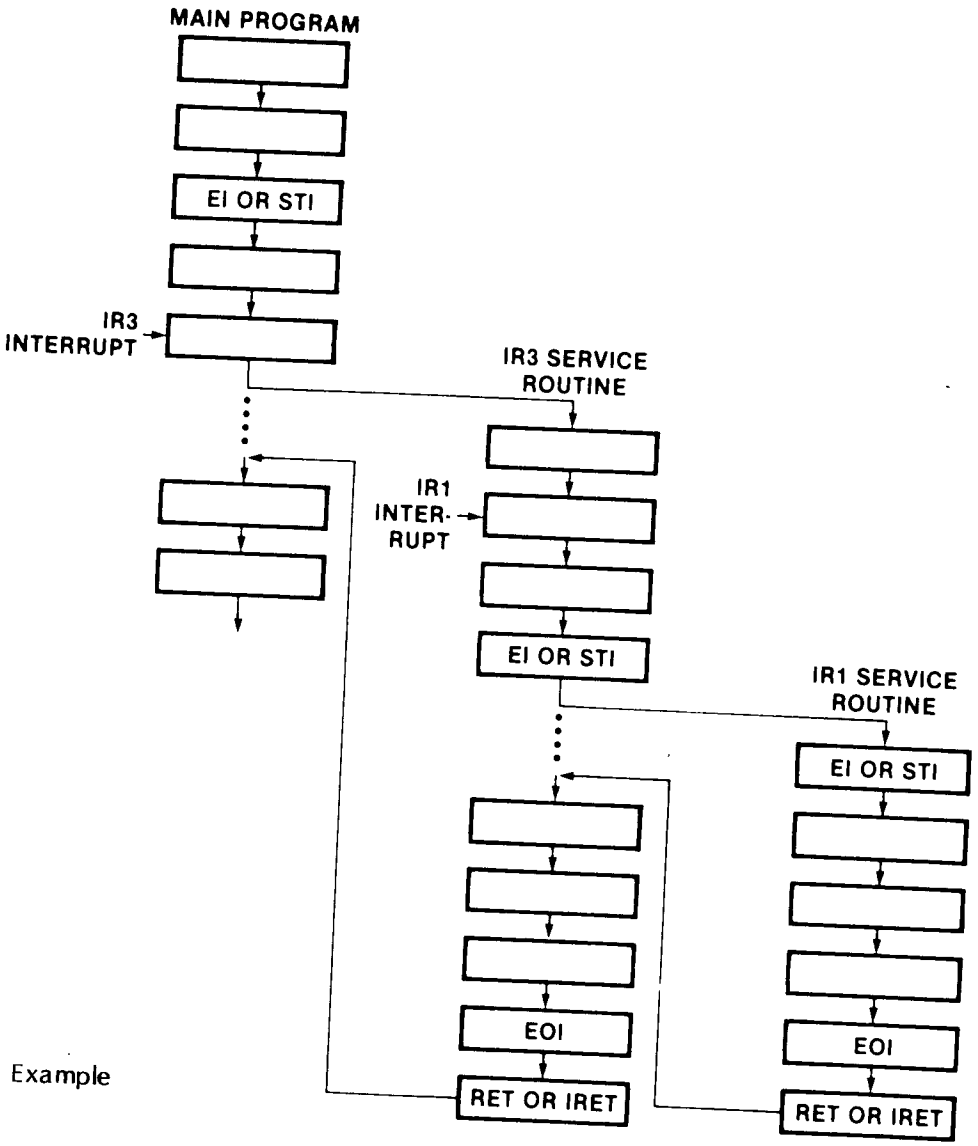
Unless otherwise programmed, the fully nested mode is entered by default upon intialization. At this time, IR0 is assigned the highest priority through IR7 the lowest. The fully nested mode, however, is not confined to this IR structure alone. Once past initialization, other IR inputs can be assigned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

Further explanation of the fully nested mode, in this section, is linked with information of general 8259A interrupt operations. This is done to ease explanation to the user in both areas.

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. EOI's will be explained in greater detail shortly.

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. A higher priority request, though, 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 aren't 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, however, they are mentioned now so the user can be aware of them. As long as these program conditions aren't inacted, 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 for the user. These are: the non-specific EOI command, the specific EOI command, and the automatic EOI Mode. Selection of which EOI to use is dependent upon the interrupt operations the user wishes 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 isn't necessary as in the "Specitic 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.

Using a special mask mode.

These conditions are covered in more detail in their own sections, but are listed here for the users reference.

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 isn't 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 obvious 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.

However, 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's 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 perfect 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't 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

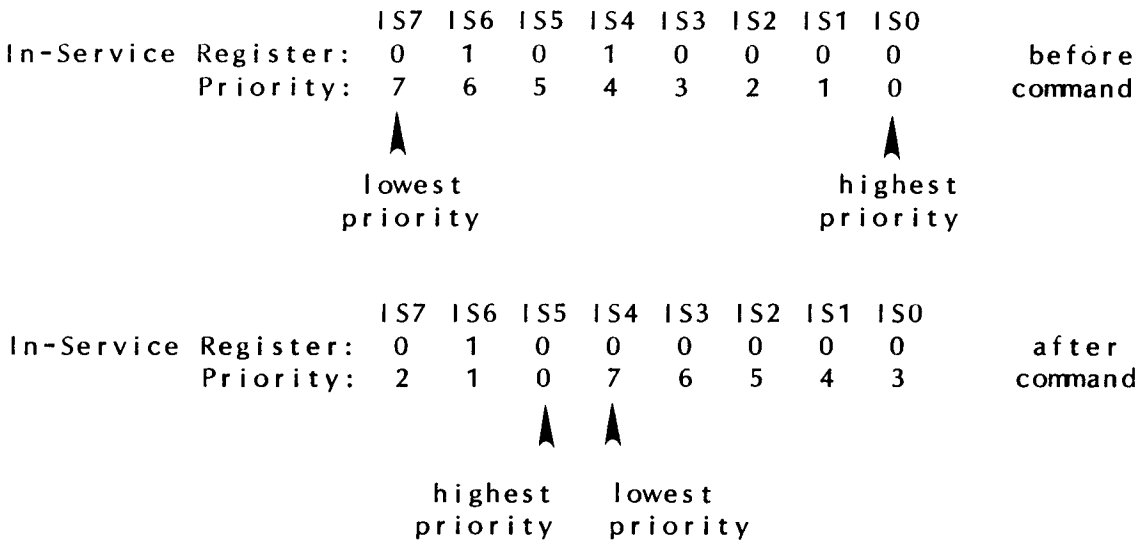
peripheral is serviced again. This is accomplished by automatically assigning a peripheral the lowest priority after being serviced. Thus, in worst case, the device would have to wait until all other devices are serviced before being serviced again.

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's 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

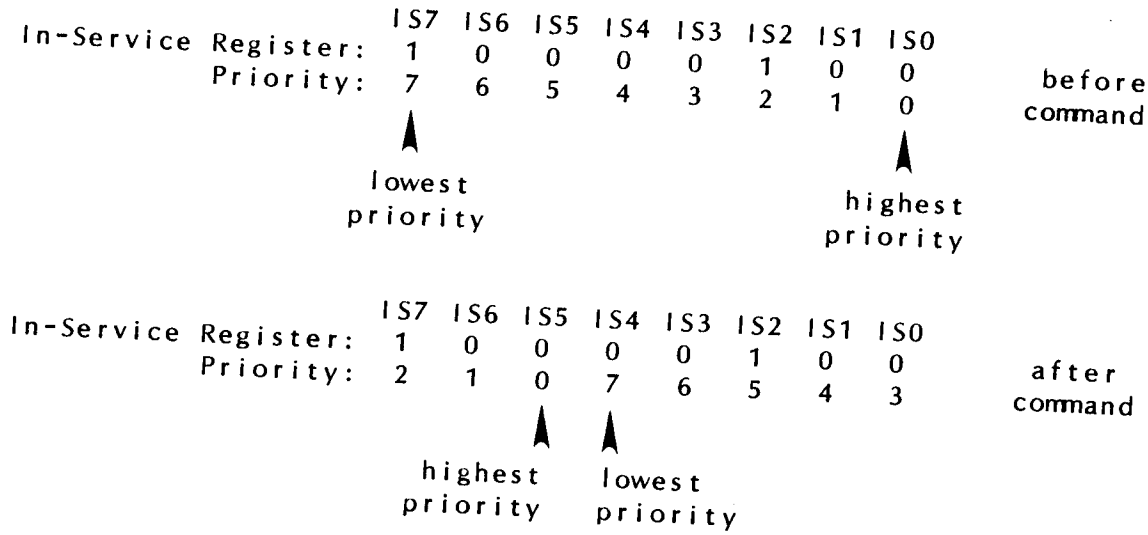
Specific rotation gives the user versatile capabilities in interrupt controlled operations. It serves in those applications in which a specific device's interrupt priority must be altered. As opposed to

automatic rotation which automatically sets priorities, specific rotation is completely user controlled. That is, the user selects 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 the programmer 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 won't 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.



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 shouldn't 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.

If it is not necessary to change IR priorities prior to the end of an interrupt routine, then this command is advantageous. 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. One example is when a portion of a main routine wishes only to be interrupted by specific interrupts. Another might be disabling higher priority interrupts for a portion of a lower priority service routine. The possibilities are many.

When an interrupt occurs while its IMR bit is set, it isn't necessarily forgotten. For, as stated earlier, 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.

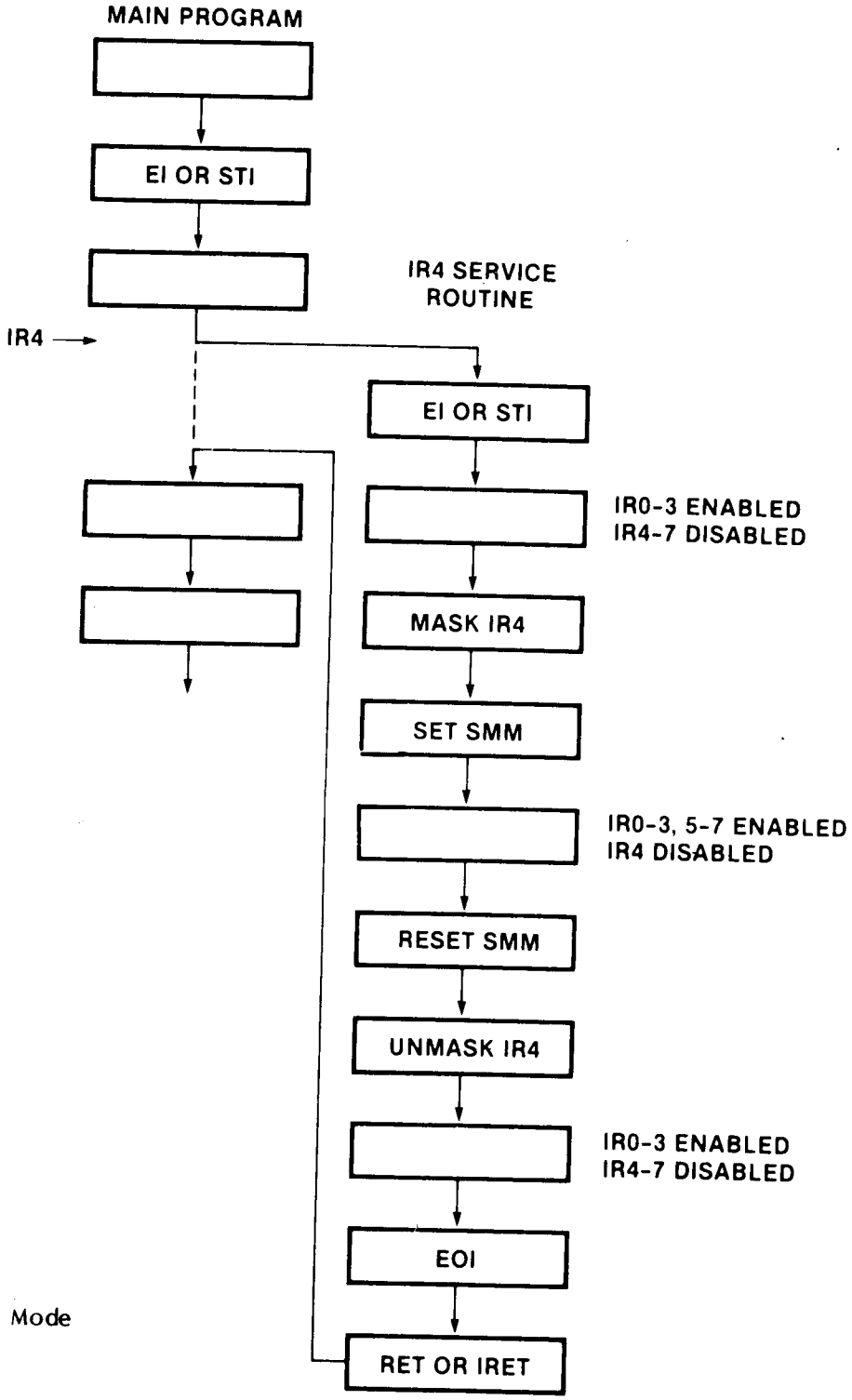
Special Mask Mode

In various cases, it may be desirable to enable interrupts of a lower priority than the routine in service. Or, in other words, 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?

Well, one method could be using an EOI command before the actual completion of a routine in service. But beware, 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 can't be used when in the special mask mode. This is because a non-specific won't 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 Figure 15, but, to avoid any confusion it's best to use the specific EOI whenever using the special mask mode.

It must be remembered 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 gives the user the capability for 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 providing, 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's 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 won't generate further interrupts on this level. The user needn't 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 isn't 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 factor.

INTERRUPT STATUS

By means of software control, the user 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 the user's program on the present status of the 8259A. This can be a versatile tool in the decision making process of a service routine, giving the user 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 which are being serviced.
IMR (Interrupt Mask Register)	Specifies all interrupt levels that are masked.

To read the contents of the IRR or ISR, the user 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's 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 addresses 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 throughout. 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 isn't 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 isn't 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 8259A's 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 thru 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 will place 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 isn't initialized as a slave receives an interrupt request, the three CAS lines won't be activated, thus staying 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 won't 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 won't 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 the user wishes 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 insure 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 aren't any other interrupts from the slave in service and an EOI command can be sent to the master. If the ISR isn't all zeros, an EOI command shouldn't 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

Programming the 8259A interrupt controllers is accomplished by using 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 which are covered shortly. 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. Note that 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.

	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	uPM

ICW1 & ICW2

ADI: The ADI bit is used to specify the address interval for the 8080 mode. If a four byte address interval is to be used, ADI must equal 1. For an eight byte address interval, ADI must equal zero. 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 four-byte interval, then the 8259A will automatically insert 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 eight-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 will be ignored when in the 8086 mode. Computing the actual memory address of the interrupt is done by multiplying the interrupt type (i.e. the interrupt vector) by four. Thus T3 corresponds to A5, T4 to A6, etc.

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 one 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, if a 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 isn't selected; thus an EOI command must be used during a service routine.

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 needn't 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.

	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".

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 won't 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 won't 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 isn't 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 isn't 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.

Timer

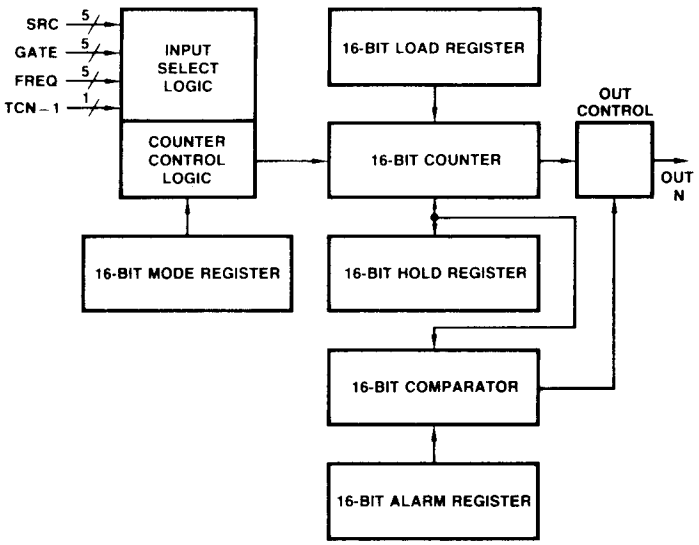
The CPU Support card has five programmable counters one of which is dedicated to use as a baud rate generator for the serial I/O port. The other four are general purpose counters two of which can be used as a Time Of Day counter. A 9513 System Timing Controller chip is used to implement these counters.

Communication with the 9513 is through two ports:

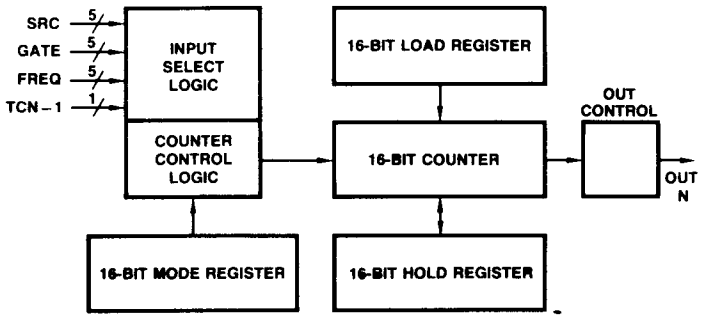
Port	Function
BASE+4	Data port
BASE+5	Status/Control port

The logic group which makes up each counter contains a sixteen 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. Detailed descriptions of their various elements are presented in the pages that follow.



Counter Logic Groups 1 and 2.



Counter Logic Groups 3, 4, and 5.

The counters and their output pins have the following definitions:

Counter	Function	Output connection
1	General purpose or Time Of Day	not used
2	General purpose or Time Of Day	Slave Interrupt Request 0
3	General purpose	Slave Interrupt Request 4
4	General purpose	Slave Interrupt Request 7
5	Serial baud rate generator	Serial Tx & Rx clocks

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 software selectable.

FREQUENCY SOURCES

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

source	binary		BCD	
	freq	period	freq	period
F1	4MHz	250nS	4MHz	250nS
F2	250kHz	4uS	400kHz	2.5uS
F3	15.625kHz	64uS	40kHz	25uS
F4	976.5625Hz	1.024mS	4kHz	250uS
F5	61.03515625Hz	16.384mS	400Hz	2.5mS

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

There is a four 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 ten or sixteen, the FOUT divider can divide by anything from one to sixteen. 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 won't 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.

Communication with the 9513 is through two I/O ports.

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

9513 REGISTERS

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/write	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 & 2 only	BASE+4

COMMAND REGISTER

The command register isn't really a register since the definitions of many of the bits change depending on the command. Its really just a port to which commands are sent.

Commands:

C7	C6	C5	C4	C3	C2	C1	C0	
0	0	0	E1	E0	G2	G1	G0	Load data pointer register with E & 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 & 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 (N = 1,2,3,4,5)
1	1	1	0	1	N2	N1	N0	Set output bit N (N = 1,2,3,4,5)
1	1	1	1	0	N2	N1	N0	Step 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 section on the data pointer register for more information.

Six of the commands provide direct software control over the counting process. Each of these commands contains a five 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 three 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.

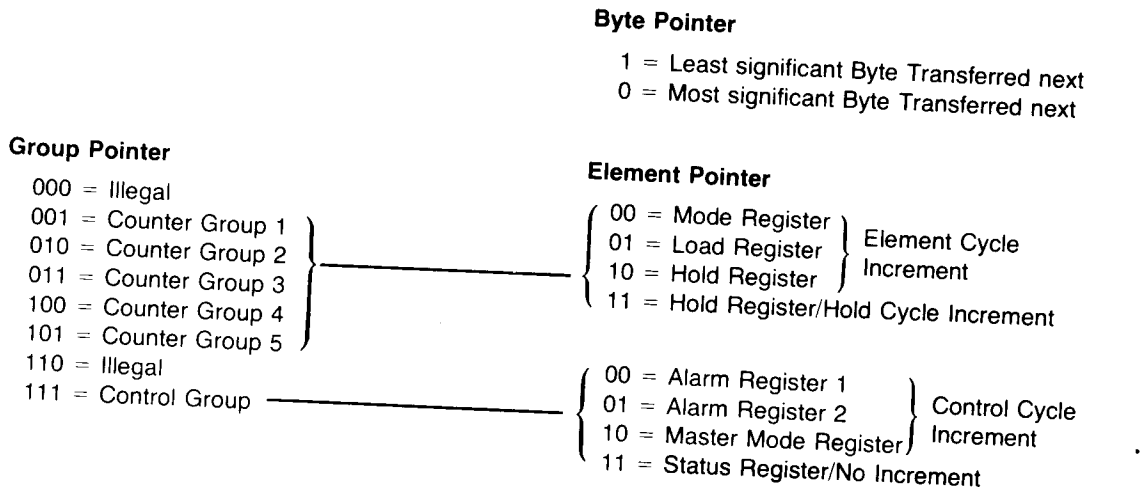
Set/Clear MM14 - disable/enable data pointer sequencing (respectively).

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

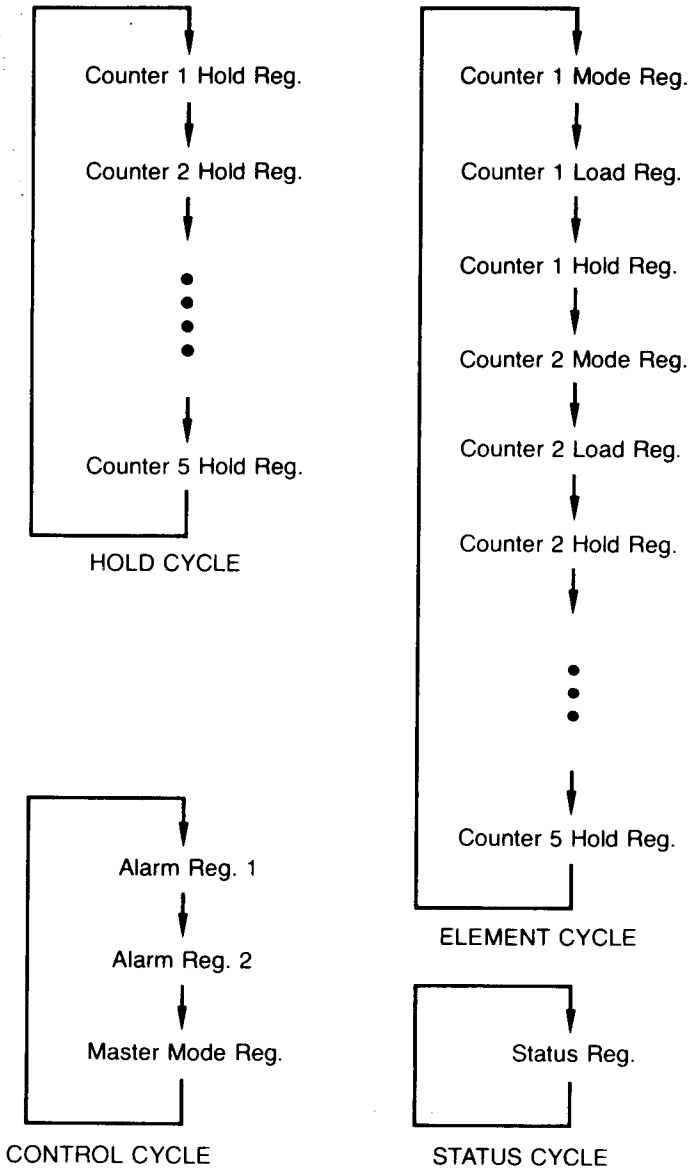
The Data Pointer register is used to establish which of the 16 bit read/write registers the CPU will communicate with through the data port. The Data Pointer register has some automatic sequencing features to allow the user to load or scan through the various registers without the need to update the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



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.



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 sixteen 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 aren't covered by the individual counter mode registers.

MM15 MM14 MM13 MM12 MM11 MM10 MM9 MM8 MM7 MM6 MM5 MM4 MM3 MM2 MM1 MM0

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 register.

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 aren't connected to anything. See the 9513 data sheet at the back of this manual 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 & comparators.

MM2 - compare 1 enable. Controls counter 1's comparator. See the section on Alarm registers & comparators.

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

LOAD REGISTER

This sixteen 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's accessed as an element of a counter group.

HOLD REGISTER

This sixteen 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's accessed as an element of a counter group.

COUNTER MODE REGISTER

This sixteen bit read/write register controls the operation of the counter. Each counter has one and it's accessed as an element of a counter group.

CM15 CM14 CM13 CM12 CM11 CM10 CM9 CM8 CM7 CM6 CM5 CM4 CM3 CM2 CM1 CM0

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 repetitively
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
  101 = active low TC pulse

CM13 - CM15 & CM7 are used to control gating of the counters. Gating is primarily used with five "GATE" inputs to the 9513 chip which aren't connected to anything on the CPU Support card. For information on the GATE inputs and the various gating modes the user should consult the 9513 data sheet appended to this manual.

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, 3, and 4 are 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 till 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 aren't 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 is the 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 isn't 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.

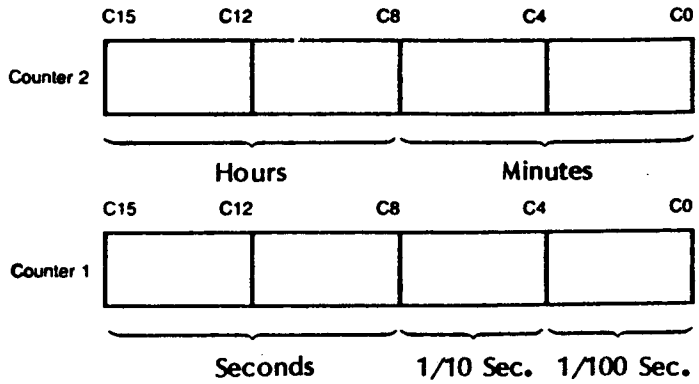
The comparator output can be active high or low. The selection is made by choosing either the active high or active low TC pulse output mode in the Counter Mode Register. This only selects the output polarity, the actual output source is the comparator.

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 isn't 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 & MM1. When these two bits are 00 counters 1 & 2 operate normally. When these bits are 11 counters 1 & 2 operate in the time of day mode. Time of day mode enables special counting modes in counters 1 & 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 one

should be FOUT, 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 thirty two bit comparison is required to activate OUT2. The time of day at which the interrupt is desired is loaded into alarm registers 1 & 2. When that time arrives OUT2 will go active if both comparators are enabled and an interrupt request will be sent to the 8259A.



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.

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.

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. Step one is to initialize counters 1 & 2 to a value of all zeros. This is done by setting load registers 1 & 2 to 0000 hex and transfering their contents into counters 1 & 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.

Step two involves setting Load registers 1 & 2 to the desired time and transfering this into the counters. The user must ensure 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.

The third step of initialization is to set load registers 1 and 2 to all 0s to insure that the counters roll over to the correct time at Terminal Count.

The fourth step is to start the counters at the desired time by writing the "Arm counters 1 & 2" command (23 hex) to the command register (BASE+5).

READING THE CLOCK

The user 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 & 2" command (A3 hex) to the command port (BASE+5). This causes the contents of counters 1 & 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 & 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 won't 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 PWR 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 won't happen too often and won't 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.

Serial Input/Output

The CPU Support card has a serial I/O device for RS-232 communication. The heart of the serial I/O device is an 8251A USART. One of the 9513's five timers is used for baud rate generation. The 8251A occupies two of the CPU Support's sixteen I/O ports.

BASE+6 is the data port.
BASE+7 is the control/status port.

Counter 5 of the 9513 system timing controller is used for baud rate generation. Counter Mode Register 5 should be set for:

No gating.
Count source = F1.
Count repetitively.
TC toggle output mode.
Count down.
Binary count.

Any part of the Counter Mode Register not mentioned can be set as desired. For 19,200 baud the "reload from Load or Hold" mode should be used with the Load register set to 0006 hex and the Hold register set to 0007 hex. This enables the counter to divide F1 (4MHz) by 6.5. The toggled output mode divides by 2 giving an output frequency of 307,692Hz (= 19,200 X 16). For the other standard baud rates the "reload from Load" mode should be used. Load register values for different baud rates are given below.

Baud rate	Load register
9600	000D hex
4800	001A hex
2400	0034 hex
1800	0045 hex
1200	0068 hex
600	00D0 hex
300	01A1 hex
150	0341 hex
134.5	03A1 hex
110	0470 hex

In general, Load register = (125000 decimal)/(baud rate). The baud rate and result are in decimal. The result should either be converted to binary or BCD division should be used.

An example of programming the baud rate generator using automatic baud rate selection is given in 8086 assembly language in the Seattle Computer Products 8086 Monitor manual starting on page 12. For further information on programming the 9513 see the TIMER section in this manual.

Before the 8251A can be used it must be initialized by sending the desired operating parameters to the control/status port (BASE+7). 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

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

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 DIFINITION

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

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 didn't detect a valid stop bit after the data and parity bits.

OE indicates overrun error. If this bit is high, the CPU didn't read the character received before the last character.

PE indicates parity error. If this bit is high, parity of the last character didn't 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.

THE DTR INPUT

The DTR input can't 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. Jumper DTR is provided on the board to pull the DTR line high or low through a resistor. If the jumper is installed DTR to -, DTR is pulled to -12 volts so that a terminal must be connected to J1 and pull DTR high before the transmitter will be enabled. If DTR to + is installed, DTR is pulled to +12 volts so that a simple three wire cable between the CPU support card and the terminal is all that is required.

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 & 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 aren't properly set because it would do two random address reads. If two bytes are allowed a register PUSH & 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's a good idea to make the software 8MHz compatible so it won't 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 doesn't 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.

Serial pinouts:

DB25S		abbr.	name	J1
1	AA	CG	Chassis Ground	1
2	BA	TxD	Transmitter Data	3
3	BB	RxD	Receiver Data	5
4	CA	RTS	Request To Send	7
5	CB	CTS	Clear To Send	9
6	CC	DSR	Data Set Ready	11
7	AB	SG	Signal Ground	13
8				15
9				17
10				19
11				21
12				23
13				25
14				2
15				4
16				6
17				8
18				10
19				12
20	CD	DTR	Data Terminal Ready	14
21				16
22				18
23				20
24				22
25				24
			no connection	26

Parallel Input/Output

The CPU Support card has both a parallel input and parallel output port. Both of these are latched and have complete handshaking capability. There are separate connectors for input and output, both designed for use with a ribbon cable with alternating signal and ground wires. The input port has passive termination (330 ohms to +5 volts and 470 ohms to ground) on all input lines and the output port has the same termination on the ACK input.

PARALLEL INPUT

The parallel input port communicates with the CPU through two I/O ports:

BASE+12 (input) Read data
BASE+13 (input) Read status

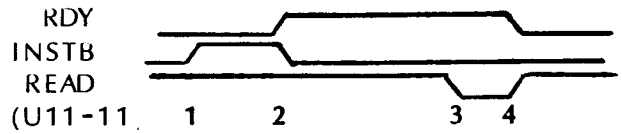
Input from BASE+12 reads the data in the input latch and sets RDY high (inactive). Input from BASE+13 reads the parallel input (& output) status:

Bit 1 INRDY (INput ReaDY)
Bit 3 INSTB (INput STroBe)

INRDY = 0 indicates data is not available.
INRDY = 1 indicates data is available.
INSTB = 0 indicates the data is latched.
INSTB = 1 indicates the latch is open.

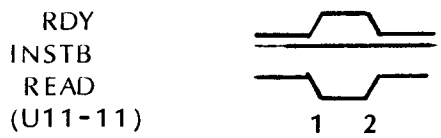
The parallel input port has two handshaking lines RDY and INSTB. RDY is an is an active low output which when active indicates the input port can accept data. INSTB is an active low input which opens the latch when inactive and closes the latch when active. Typical input cycles are given below:

Input mode 1, standard mode with complete handshaking.



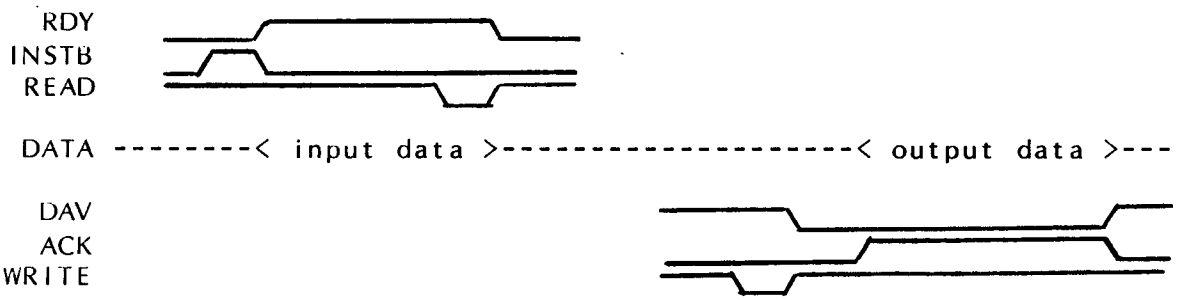
1> INSTB goes high putting opening latch.
2> INSTB goes low latching data.
RDY goes high indicating the input port is not ready to receive data.
The INRDY status bit goes high at this time indicating data is ready.
3> READ (U11 pin 11) goes low as the CPU reads the data.
4> READ goes high at the end of the read cycle.
RDY goes low indicating the port is ready to accept new data.

Input mode 2, transparent mode with no handshaking. In this mode INSTB is tied high or left floating with termination. The CPU can at anytime read the data on the input lines, the data isn't latched. The output signal RDY goes inactive during the read.



1> CPU starts reading the data. RDY goes high indicating CPU is reading.
2> CPU finishes reading. RDY goes low indicating CPU isn't reading.

Input/output mode 3, bidirectional I/O. In this mode the input and output lines are tied together for communication with another bidirectional device. Note that jumper OUT must be in position OUT to - to avoid data bus conflicts. Some type of external harness must be provided to connect the data lines together. The handshaking in this mode is identical to that of the standard I/O modes.



INPUT CONNECTIONS

Connections to the parallel input port are made through connector J3. J3 is designed to be connected to a 25 pin ribbon cable terminated with a DB25S connector (available from Seattle Computer Products). The pinout is as follows:

J3	function	DB25S
1	ground	1
3	ground	2
5	ground	3
7	ground	4
9	ground	5
11	ground	6
13	ground	7
15	ground	8
17	ground	9
19	ground	10
21	ground	11
23	power	12
25	power	13
2	DI0	14
4	DI1	15
6	DI2	16
8	DI3	17
10	DI4	18
12	DI5	19
14	DI6	20
16	DI7	21
18	RDY	22
20	INSTB	23
22	power	24
24	power	25
26	no connection	

The four power pins 12, 13, 24 and 25 provide either +5 volts regulated or +8 volts unregulated from pins 1 and 51 of the S-100 bus for powering an external device such as a keyboard. Selection between the two power supplies is accomplished through the use of jumper PWR OUT. PWR OUT to 8 selects +8 volts unregulated and PWR OUT to 5 selects +5 volts regulated. The source of the +5 volt supply is a 7805 type voltage regulator on board. This regulator also supplies power to the passive termination pack. The current available at these pins depends primarily on the input voltage to the regulator from pins 1 and 51 of the S-100 bus but also on ambient temperature and how many terminations are used (see the section on termination). A table giving maximum output current for two cases is given:

Vin	Tamb	Iout max
12V	50 C	490mA
10V	40 C	836mA

These current ratings can be increased by 9mA per terminator not used.

PARALLEL OUTPUT

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

BASE+12 (output) write data
BASE+13 (input) read status

Output to BASE+12 loads data into the output latch and sets DAV low (active).
Input from BASE+13 reads the parallel output (& input) status:

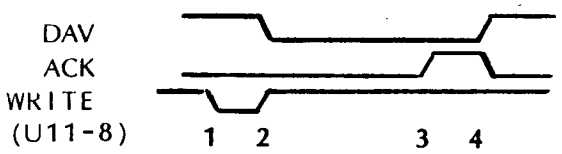
Bit 0 OUTRDY (OUTput ReaDY)
Bit 2 ACK (ACKnoledge)

OUTRDY = 1 indicates the previous data has been read and the the output port is ready to accept new data.
ACK = 0 while the external device reads the data.

The parallel output port has two handshaking lines DAV and ACK. DAV is an active low output line which when active indicates data is available for reading. ACK is an active low input which when inactive indicates the external device is reading the data. The high to low transition of ACK turns off DAV.

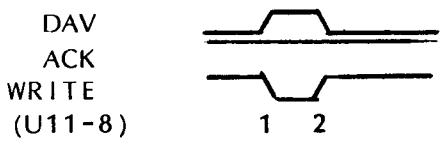
Jumper OUT selects how the parallel port's output drivers are enabled. If OUT to + is connected the drivers are always on and driving. If OUT to - is connected the drivers are only on when the external device is reading the data (when ACK is high) or when ACK is floating but terminated.

Output modes.
Mode 1, standard mode with complete handshaking.



1> WRITE goes low as the CPU writes data to the output port.
2> WRITE goes high as the CPU finishes writing.
DAV goes low indicating data is available
3> ACK goes high as the external device reads the data. If jumper OUT to - is installed the output drivers go on.
4> ACK goes low as the external device finishes reading. If jumper OUT to - is installed the output drivers go off.
DAV goes high indicating data is no longer available.

Mode 2, direct output with no handshaking. In this mode ACK is tied high or left floating with termination. The CPU can send out data at any time and it appears at the output connector J2. The output signal DAV pulses high during write in case this is needed to strobe data into the external device.



1> DAV goes high as CPU starts to write.
2> DAV goes low as CPU finishes writing.

Input/output mode 3, bidirectional I/O. This is a combination of input and output modes 1. See input/output mode 3 in the input mode section for a complete description of this mode.

OUTPUT CONNECTIONS

Connections to the parallel output port are made through connector J2. J2 is designed to be connected to a 25 pin ribbon cable terminated with a DP25P connector (available from Seattle Computer Products). The pinout is as follows:

DB25P	function	J2
1	ground	1
2	ground	3
3	ground	5
4	ground	7
5	ground	9
6	ground	11
7	ground	13
8	ground	15
9	ground	17
10	ground	19
11	ground	21
12		23
13		25
14	DO0	2
15	DO1	4
16	DO2	6
17	DO3	8
18	DO4	10
19	DO5	12
20	DO6	14
21	DO7	16
22	ACK	18
23	DAV	20
24		22
25		24
no connection		26

TERMINATION

All inputs to the parallel input and output ports (DI0-DI7, INSTB and ACK) have passive termination to properly terminate the lines and reduce reflections. This termination consists of a pair of resistors for each input, 330 ohms to +5 volts and 470 ohms to ground. This provides an impedance of 194 ohms and an open circuit voltage of 2.94 volts. Since the open circuit voltage is greater than the TTL threshold VIH of 2.0 volts either totem pole or open collector drivers can be used on these inputs. The low drive requirement is 0.8 volts at 11.4 mA. If the device to be connected to the port can't meet this requirement the termination can be removed from each input individually by removing the termination network RN1, bending out the pin in question, and replacing the network. If no termination is required on all inputs the whole termination network RN1 can simply be removed. With termination installed a six meter (twenty foot) cable can be used. Without termination, the cable should be as short as possible.

Input	RN1 pin
DI0	8
DI1	9
DI2	10
DI3	4
DI4	2
DI5	1
DI6	13
DI7	12
INSTB	3
ACK	11

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 uA (source)
VIL = 0.8 volts @ 0.4 mA (sink)

OUTPUT DRIVE CAPABILITY

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

Sense Switch

The CPU Support card has an eight bit sense switch. The switch positions can be read by the CPU on port BASE+15.

Open switch = 0
Closed switch = 1

Bit 0 is the top switch, bit 7 is the bottom switch.

Note: Seattle Computer Products may use these sense switches (if BASE=F0 hex) for future software options starting with bit 0 (top switch) and working toword bit 7 (bottom switch) as more bits are needed. The user wishing to use these switches should start with bit 7 to insure software compatibility for as long as possible. Currently bit 0 is used by our 8086 Monitor to select automatic disk boot.

Eprom Socket

The CPU Support card socket U5 is for either an Intel-type 2716 (single supply) EPROM (including the TMS2516) or an Intel-type 2732 EPROM.

Jumper ROM selects the EPROM type. ROM to 16 selects a 2716 or ROM to 32 selects a 2732. If the 2732 type EPROM is used none of the pins of the ADDR jumper should be connected. One of the two holes in the jumper block should be put over an end pin with the other hole hanging out in space.

Switch 1 position 6 enables or disables the extended addressing. With extended addressing enabled A16-A19 must all be high to select the EPROM. With extended addressing disabled A16-A19 have no effect on the EPROM. Jumper ADDR selects A11 = 0 or A11 = 1 for the 2716 type ERPOM or is not connected as described above for the 2732 type EPROM.

S1-6	Jumper ADDR	2716 address
OFF	ADDR-LO	XF000 hex - XF7FF hex
OFF	ADDR-HI	XF800 hex - XFFFF hex
ON	ADDR-LO	FF000 hex - FF7FF hex
ON	ADDR-HI	FF800 hex - FFFFF hex

S1-6	2732 address
OFF	XF000 hex - XFFFF hex
ON	FF000 hex - FFFFF hex

In 8086 systems S1-6 should be on and jumper ADDR should be connected ADDR-HI if a 2716 type EPROM is used to put the EPROM at the top of memory for the 8086 power on jump to FFFF0 hex.

In 8080/8085/Z80 systems S1-6 should be off because these systems do not provide A16-A19. Jumper ADDR selects EPROM at F000 hex or F800 hex if a 2716 type EPROM is used.

Switch 1 position 5 enables or disables the EPROM.

S1-5	EPROM
OFF	disabled
ON	enabled

The EPROM can also be disabled by accessing port BASE+14, any access at all, input or output, will turn the EPROM off. (The EPROM can only be turned back on by activating RESET). In this way the EPROM can be disabled under software control. This feature is primarilly used to remove the EPROM from the computer's memory space after using it to boot up an operating system. During the time the EPROM is on, RAM boards which reside at the same location as the EPROM can be disabled by using bank select boards which "come up" disabled or by enabling the PHANTOM driver as described below and using RAM boards which disable when PHANTOM is active. (Only the board which resides at the same address as the EPROM needs to use PHANTOM).

The PHANTOM jumper selects whether or not the S-100 PHANTOM line is active when the EPROM is on.

PHANTOM jumper	PHANTOM
+	active when EPROM is addressed
-	not driven by CPU Support card

Theory of Operation

The functions of this card are implemented primarily with MOS LSI microprocessor support chips, with TTL used for parallel I/O and S-100 bus interface. Most of the circuitry is very straightforward and requires no explanation, so only selected portions of the circuit are discussed below.

Wait State Generator

When a wait state is to be generated, the circuit simply gates inverted pSYNC from the CPU onto the RDY line with a driver wired for open-collector operation. The use of pSYNC to generate a wait state may not work with all CPU cards, although it should according to the IEEE S-100 standard. It does work with the SCP 8086 CPU card, which when run at 8 MHz is the only card we know of that needs the wait state for I/O.

Pulse Stretcher

This circuit is formed primarily by U16, pins 7 - 14, and U18. Its purpose is to stretch the RD (read), WR (write), and INTA (interrupt acknowledge) pulses to meet the minimum pulse width requirements of the MOS LSI parts.

When pSTVAL* goes high during pSYNC, one of the flip-flops of U16 is cleared. This allows inverted copies of sINP, sOUT, and sINTA to be gated through as the control lines RD, WR, and INTA, respectively. One of these control lines stays active until the trailing edge of either pWR* or pDBIN clocks the flip-flop and gates it off. The "trailing edge" means the rising edge of pWR* or the falling edge of pDBIN. Thus the control pulse terminates just about the time it would have if pDBIN or pWR* had been used, but it starts considerably earlier.

Control Line Pull-up Resistors

Three 10k resistors pull up the RD and WR internal control lines as well as the CS line to the 9513. The purpose is to ensure these three lines remain high, and the 9513 inactive, when the board as a whole is powered off but the 9513 is provided with external power to keep its time-of-day counters running.

U18 must not be LS (Low-power Schottky) to ensure RD and WR will be pulled high. The 74LS32 clamps down its outputs when power is removed; either 7432 or 74S32 is OK in this position.

Bus Takeover During Interrupt Acknowledge

When normal 8080 or Z80 CPUs are used with a multi-byte INTA (interrupt acknowledge) sequence, such as the "CALL" instruction jam of the 8259 interrupt controller, they do not continue to indicate INTA status for the second and third bytes of the sequence. It would be possible to design an interrupt controller which counts off the three cycles whether INTA status was present or not, but this only solves half the problem. Since INTA status is not present, other devices on the S-100 bus, such as memory boards, may be fooled into thinking they are being addressed. If such a device starts driving the bus (since pDBIN will be active), the vector from the interrupt controller will never reach the CPU.

The complete solution has been implemented on the CPU Support card. During the second and third INTA cycles, the Support card takes over the status bus from the CPU by asserting SDSB* and driving INTA status. At the same time, the address bus is taken over so that the interrupt controller cascade information may be transmitted on A0-A2.

The two D flip-flops of U19 count each INTA cycle on the trailing (falling) edge of pDBIN. Since it is possible for the 8080 and Z80 to grant a Hold Acknowedge during the INTA sequence, pHLDA is gated with pDBIN to ensure that DMA cycles are not mistaken for INTA cycles. A jumper selects whether the Support card will take over the bus for one cycle (8086 mode) or two (8080 mode). The RC network at U19 pin 12 insures that INTA status is still present at the "D" input when pDBIN clocks the flip-flop on the first INTA cycle, since some CPU cards remove it a bit early.

Repair Service

Repair service is available at the factory for any product manufactured by Seattle Computer Products. For items under warranty, there is no charge for this service. For repair of this board when it is no longer under warranty, return the board to the factory address shown below along with a check for $35. If the repair and return shipping can be accomplished for the amount, the work will be done and the board returned to you along with a check for any overpayment. If the cost of repair exceeds this amount, you will be notified before any work is done. Normal factory repair time is 48 to 72 hours. Ship all returned boards to:

Seattle Computer Products
1114 Industry Drive
Seattle, Washington 98188

WARRANTEE AND WARRANTY PERIOD

The Seattle Computer Products (hereinafter referred to as SCP) warranty for this product extends to the original purchaser and all subsequent owners of the product for a period of one year 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.

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 the SCP retains the right in all cases to examine the defective board or other products prior to the items 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 than 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 consequencial damange the user may suffer because the products 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 toreturn the product or component to SCP, also return a written explaination of the difficulty encountered along with your name, address and phone number. Package the items in a crushproof container with adequate packing material to prevent damage and ship prepaid to:

Seattle Computer Products
1114 Industry Drive
Seattle, Washington 98188