altair™ 8800b

TURNKEY COMPUTER DOCUMENTATION

mits
a subsidiary of Pertec Computer Corporation

altair 8800b

TURNKEY COMPUTER DOCUMENTATION

1st Printing, July, 1977

mits
a subsidiary of Pertec Computer Corporation
2450 Alamo S.E. /Albuquerque, New Mexico 87106

TABLE OF CONTENTS

1. INTRODUCTION

1-1. Description

The Altair 8800b Turnkey computer is housed in a standard Altair system case and contains the following elements:

-Power supply and motherboard assembly

-CPU board with the 8080A microprocessor

-Turnkey Module with memory and I/O circuitry

-Front panel board

The heart of the computer is the Central Processing Unit (CPU) board which holds the 8080A microprocessor and its associated circuitry. The 8080A performs all the logical and arithmetic computations for the system. It also supplies control and status signals for the other system components. Other circuitry on the CPU board provides clock signals and synchronization functions.

The Turnkey module is a general support board for the Turnkey system which includes memory, I/O and AUTO-START control. The memory section has 1K bytes of random access memory (RAM) and positions for up to 1K bytes of read-only memory (ROM). Random access memory stores information that can be read, written or changed at will. RAM is volatile, however, and information is lost when power is interrupted. Programmable Read-only memory (PROM) is non-volatile. Information in PROM is always present whether the power is on or not. Thus, PROM can store programs and data which must be permanently retained. The computer cannot write information into PROM, however. A special PROM programmer must be used to do this, although factory-programmed PROMs are available for some widely used functions.

The serial Input/Output channel (SIO) connects the parallel data bus in the computer to serial input/output devices, such as Teletypes, CRT terminals, modems, etc. The SIO may be configured to accomodate a variety of terminal types and speeds to match virtually any serial I/O arrangement.

The AUTO-START feature is the key to the Altair 8800b Turnkey computer's ease of operation. When the computer's power is turned on or the START switch is actuated, the AUTO-START logic forces the computer to execute the instruction at a pre-selected address in PROM. The address could be the beginning of a series of instructions to load a program from an I/O device, the start of a monitor program or a dedicated application program.

This means that the Monitor program is ready to receive commands. At this point, the user may enter data into memory or cause control to jump to another program (e.g. a loader). During the execution of any program, actuating the START switch causes control to return to the Monitor. The prompt period is printed and the Monitor is again ready to receive commands.

For more information on the Monitor, its commands and use, see the Turnkey PROM Monitor manual.

Other PROMs, ROMs or non-volatile memory may be installed at the AUTO-START address (although only type 1702A PROMs may be installed on the Turnkey Module board). The action of the computer when the power is turned on depends upon the program in that memory. For example, if the Multi-Boot Loader PROM is installed to load Altair BASIC, the first message printed on the terminal is BASIC's "MEMORY SIZE" initialization question which appears after BASIC is loaded. For more information, see the documentation for the program in use.

ALTAIR 8800b TURNKEY COMPUTER

SECTION II
THEORY OF OPERATION

2. THEORY OF OPERATION

2-1. General

A generalized block diagram of a computer system is shown in Figure 2-1. The Control, Processor, Memory and I/O are arranged so that instructions in Memory cause the Control to direct the Processor to access and manipulate Memory data. The Control also directs the Processor to arrange for Input and Output of data.

The system in the diagram is stored-program computer. Its actions are directed by instructions that are stored in memory. The computer has the ability to change the order in which its instruction are executed and to modify the instructions themselves. This accounts, in part, for the great flexibility of stored program computers.

As the diagram shows, the Control and Processor elements are the heart of the system. All the other elements of the system communicate with and are controlled by the Control and Processor. In many computers, including the Altair 8800b Turnkey computer, the Control and Processor elements are combined in one unit, the Central Processing Unit. Moreover, in the Turnkey system, some of the memory and I/O functions are combined into the Turnkey Module.

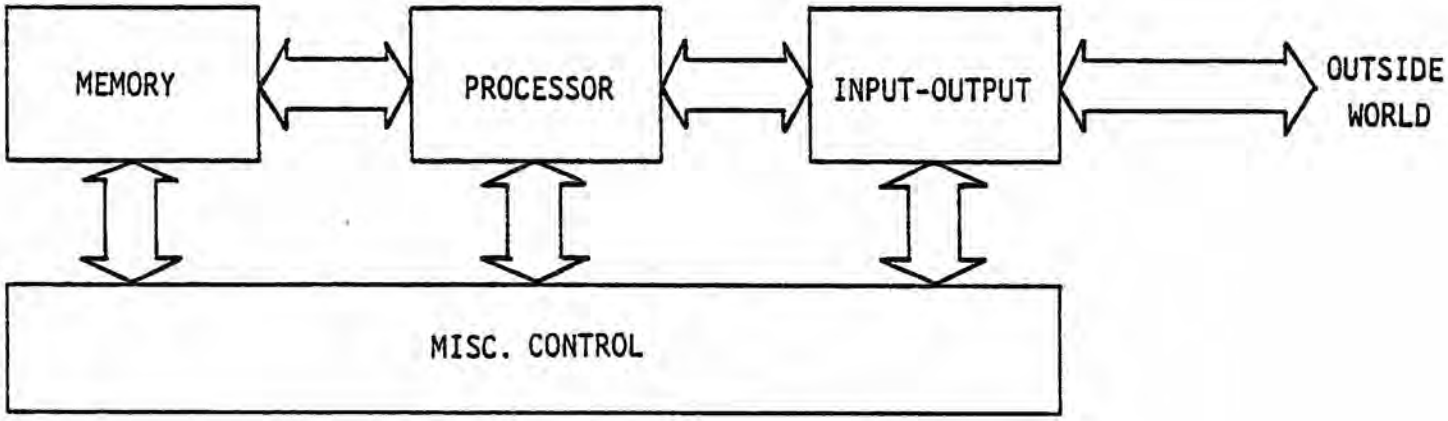


Figure 2-1
Typical Computer System
Block Diagram

Figure 2-2 is a diagram of the Altair 8800b Turnkey computer system showing how the functional parts are distributed among the physical units of the system.

One important feature of the Turnkey system's design is that it is based upon a bus. The Altair bus is a collection of conducting paths (lines) which distribute signals to all of the system's components. The pins of each board are connected directly to the corresponding pins of every other board. As a result, the system is easily expandable, since every board has access to all of the data, addresses, status and control information in the system.

The bus is located physically on a 100 conductor printed circuit motherboard. The printed circuit card edge connectors on the motherboard provide mechanical support for the system's circuit boards, as well as electrical connection to the bus.

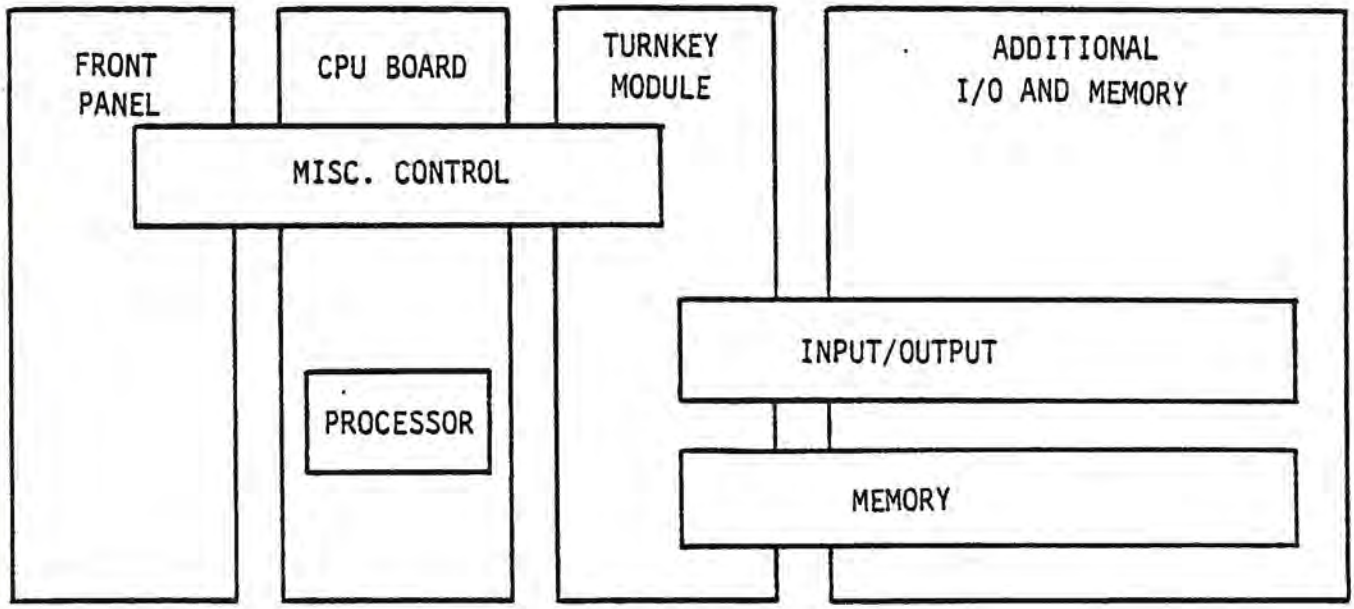


Figure 2-2
Altair 8800b Turnkey System
Function Distribution

The bus has 16 Address lines, 8 Data-In lines carrying data to the CPU, 8 Data-Out lines carrying data from the CPU and lines carrying pulses that indicate the status of the system and serve various control functions. A list of the conductors on the Altair system bus is in the Appendix.

The activity on the bus corresponds to six machine cycles; Memory Read and Write, I/O In and Out, Interrupt and Halt. The functions of the various bus lines depend on what cycle is currently in progress.

In a Memory cycle, the address of the memory location to be read or written is carried on the address bus. The Data-In bus carries data or instruction codes from memory to the CPU during a Memory Read. The Data-Out bus carries data from the CPU during a Memory Write. In an I/O cycle, the Address bus carries the address of the I/O port through which the data transfer is to take place. The port address is only eight bits long, so it is carried both in the high and low order bytes of the 16 bit Address bus. The Data-In bus carries information from the port to the CPU during an Input cycle. Similarly, the Data out bus carries information from the CPU to the port in an Output cycle.

The Interrupt cycle is provided so that peripheral devices can gain access to the bus. In the absence of interrupts, the CPU can be programmed to check its peripheral devices periodically and service them as they have information to transfer. An input device, for example, waits until the CPU signals that it is ready for input. This arrangement is simple, but it is often inefficient, since the CPU is required to poll the I/O devices whether or not there is information to be transferred.

In an interrupt driven system, on the other hand, the I/O device signals the CPU when it is ready to transfer data. This signal is called an Interrupt Request. If the CPU Interrupt Enable bit is set to 1, the CPU acknowledges the Interrupt Request. Otherwise, the request is ignored. In an Interrupt cycle, the CPU fetches an instruction which causes the computer to interrupt the execution of its current program and begin executing another program (called an interrupt service routine) at a special location in memory. The computer also stores the location of its current instruction so that it can return to the place where it left off when the interrupt service routine is completed. The interrupt scheme allows the computer to work on a program until a peripheral device has information to transfer. The CPU can then accept the information and take the necessary action without losing track of the program in progress.

A special machine cycle is provided for direct memory access. During the Hold state, the CPU is effectively disconnected from the bus. This allows a direct memory access device, if it is used, to take control of the bus and transfer information directly to and from memory. For more information on machine cycles, see the Intel 8080 machine cycles, see the Intel 8080 Microcomputer System User's Manual (abbreviated IMSUM), section 2, pp 3-11.

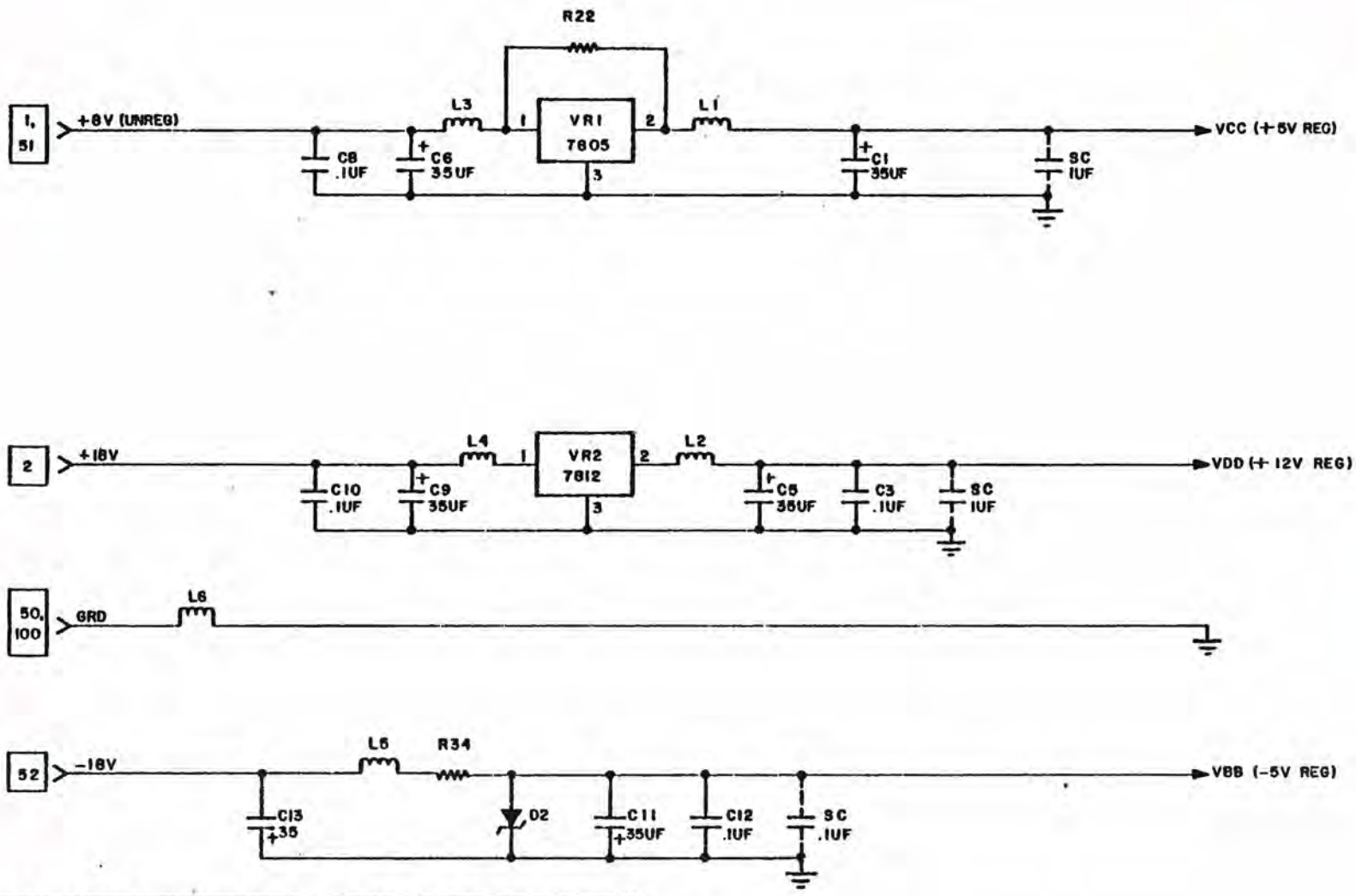
2-2. The CPU Board

NOTE

In the following descriptions, the names of signals appear in capital letters. Signal names which are barred ($\overline{\text{PRESET}}$, for example) are active LOW. That is, they are LOW (Ø volts) when activated and HIGH (+5 volts) otherwise. All other signals are active HIGH.

The address and data lines represent binary 1 as HIGH and binary Ø as LOW.

On the Turnkey module, the sense switches and the address switches for RAM, PROM and AUTO-START allow manual entry of data and addresses. The two states of each switch -on and off- correspond to the two states of a binary digit - 1 and 0. The bits represented by the sets of switches can, therefore, be treated by the computer in the same way as any other data or address bits. Moving a switch in the direction of the arrow next to each set of switches puts the switch in the 1 position.



REF DESIG	TYPE	VCC	GRD	OTHER	REF DESIG	TYPE	VCC	GRD	OTHER
					M	8080A	20	2	
G,B	74LS04	14	7		J,X,R,V, N,U,P	74366 OR 8T98	16	8	
C	74LS13 OR 74LS20	14	7		K	8212	24	12	
S,Y	74LS14	14	7		D,E	8216	16	8	
					F	8224	16	8	VDD = 9
P,W	74367	16	8		A	4009	1	8	VDD = 16

Figure 2-3b.
CPU Board Power Regulators

The CPU board contains the 8080A microprocessor, its timing and auxiliary control circuits. It also contains buffers and line drivers for the system bus. The heart of the CPU (and the whole computer system) is the 8080A microprocessor. All of the other circuitry on the board serves to support the microprocessor and connect it to the rest of the system.

A. The Microprocessor. The 8080A is IC M on the schematic, Figure 2-3a. For a complete description of the internal organization and timing of the 8080A, see the IMSUM, Chapter 2.

B. Clock and Synchronization. The 8224 clock generator (IC F on the schematic) provides the two-phase, 2 MHz clock for the 8080A at the required 0 and 12 volt levels. The master timing reference for the 8224 is an external 18 MHz crystal. The system timing frequency can be changed by replacing the crystal.

The clock generator also synchronizes the 8080A's READY and RESET inputs and provides a status strobe signal ($\overline{STSTB}$) that is used to load the 8212 status latch. The READY signal is the logical product (AND) of bus signals XRDY, PRDY, XRDY2 and FRDY. Normally, only PRDY is used for memory synchronization; the others are set to logical 1 by pullup resistors. The bus signal $\overline{PRESET}$ is filtered, lengthened, shaped and synchronized by the 8224 to generate the RESET input for the 8080A. The $\overline{STSTB}$ signal is generated as soon as the status information is available on the 8080A data lines.

C. The Status Latch. At the beginning of each bus cycle, status information is presented momentarily on the 8080A data lines. This information is stored in the 8212 latch so that the data lines may be used for other information later. The 8212 is IC K on the schematic. For a complete description of the status information and its timing, see page 2-6 of IMSUM. The 8212 is described on page 5-101 of the same manual.

D. Buffers.

1) Bus splitters. Information is communicated to and from the 8080A through an eight-bit, bidirectional data bus. This bus is buffered by two 8216 bidirectional bus drivers to form a TTL compatible, bidirectional bus on the CPU board. The direction of the 8216's is controlled by PDBIN.

At the other end of the CPU bus are tri-state buffers that split the bidirectional data bus into Data-In and Data-Out busses which are brought out to the system bus. The buffers shown in zones B-3 and B-4 transfer data from the system Data-In bus to the bidirectional bus when both PDBIN and DIG1 are HIGH. (DIG1 is normally HIGH unless the bidirectional bus is being accessed through connector P3. When either PDBIN or DIG1 are LOW, the buffers are put in a high impedance state and no information is transferred through them. Similarly, the buffers in zones C-2 and C-3 transfer data from the CPU bus to the Data-Out system bus when the signal $\overline{\text{DO DSBL}}$ is HIGH.

If no information is present on the Data-In bus, all eight lines are pulled HIGH by resistors on the CPU board. When a interrupt is acknowledged, this forces a RST 7 instruction onto the bus if the Vector Interrupt Board is not used.

2) Input buffers. All input lines to the CPU board are buffered by TTL circuitry. Unused lines are pulled HIGH by resistors which also allow any one of several boards to pull any input line LOW. A line is normally pulled LOW either by an open collector driver or a tri-state driver, with the condition that no more than one tri-state driver may be enabled at one time.

Examples of these input lines are $\overline{\text{PINT}}$ and PHOLD in zone D-8, the READY lines (PRDY, etc.) in zone A-7, PRESET in zone A-6, and the driver disable lines throughout the schematic.

3) Output buffers. All outputs from the CPU board are buffered to drive 30 standard TTL loads. The lines are collected into four groups, Address lines, Data Out lines, Status lines and a Control group made up of PWR, PDBIN, PWAIT, PSYNC, PHLDA and PINTE. Each group of lines may be disabled as a group by signals from the bus.

2-3. The Turnkey Module

The Turnkey Module contains 1024 bytes each of RAM and PROM, a serial I/O channel, AUTO-START logic, sense switch, the front panel board and logic and miscellaneous logic. The Turnkey Module schematic is shown in Figures 2-4 a, b and c.

A. RAM and PROM. The RAM and PROM memories and their control logic are shown in the schematic diagrams, Figures 2-4 a and c.

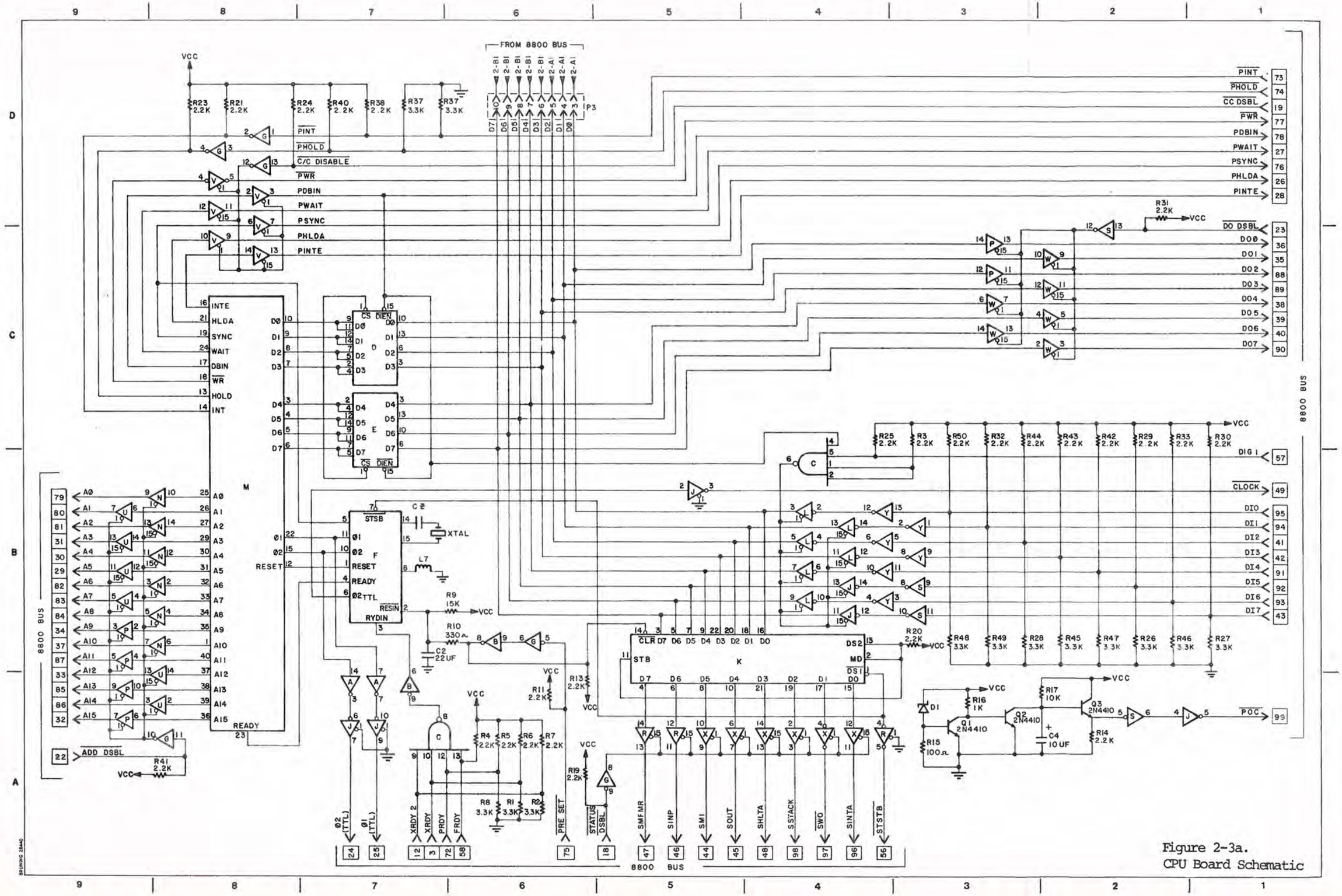


Figure 2-3a.
CPU Board Schematic

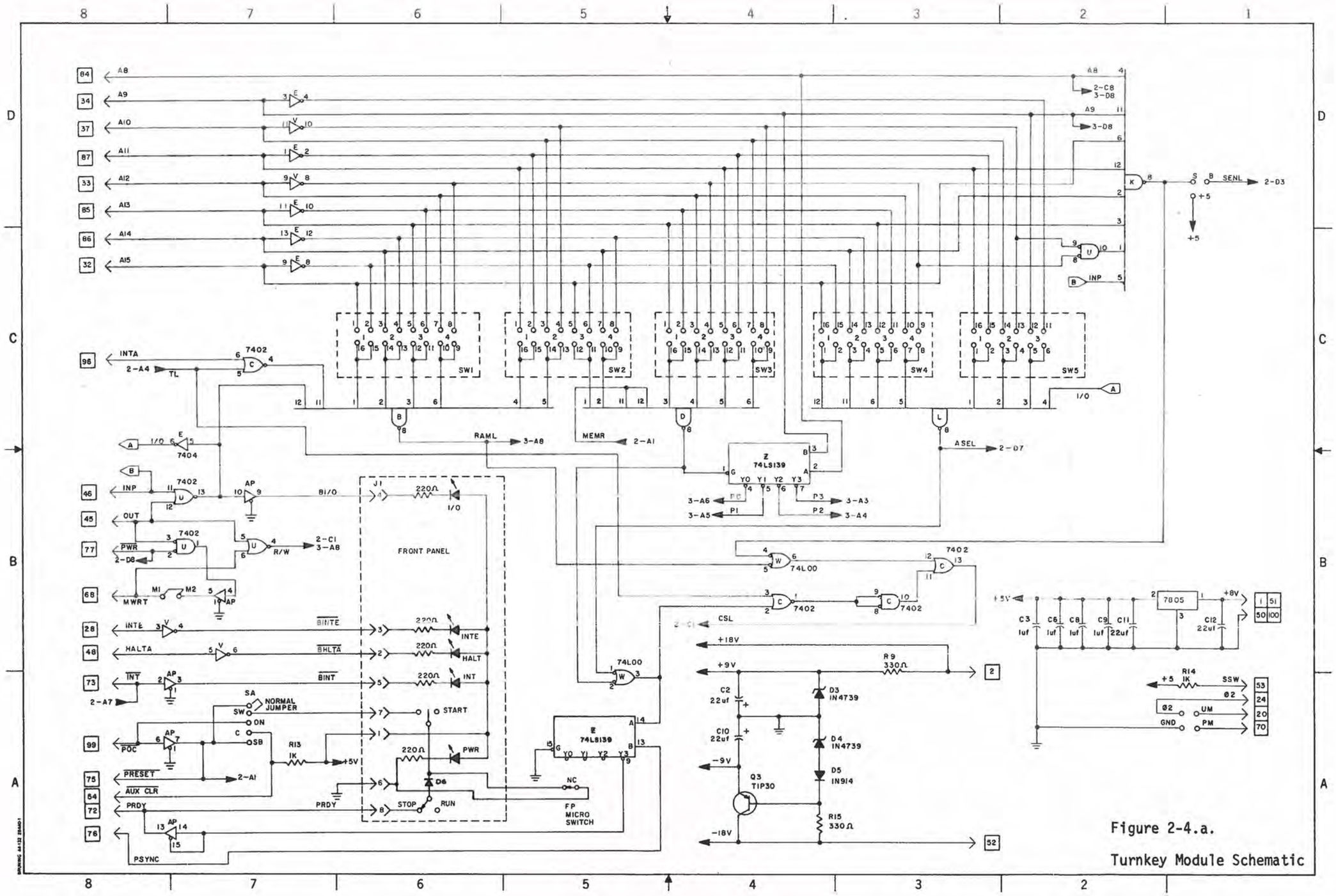


Figure 2-4.a.

Turnkey Module Schematic

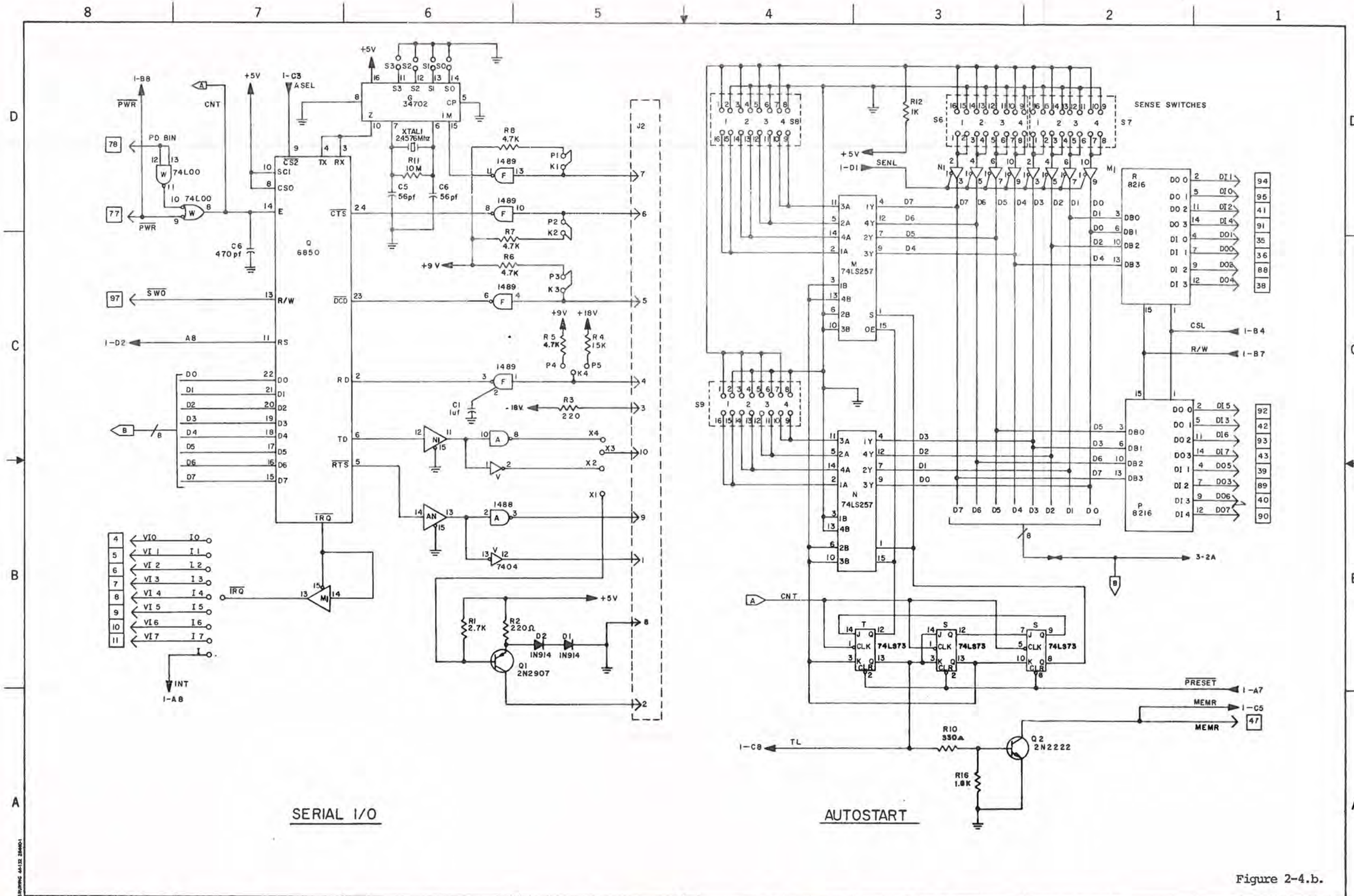

SERIAL I/O
AUTOSTART
SENSE SWITCHES
G 34702
Q 6850
XTAL1 24576MHz
74LS257
74LS73
R 8216
P 8216
2N2907
2N2222
1N914
7404
1488
1489
74LOO

Figure 2-4.b.

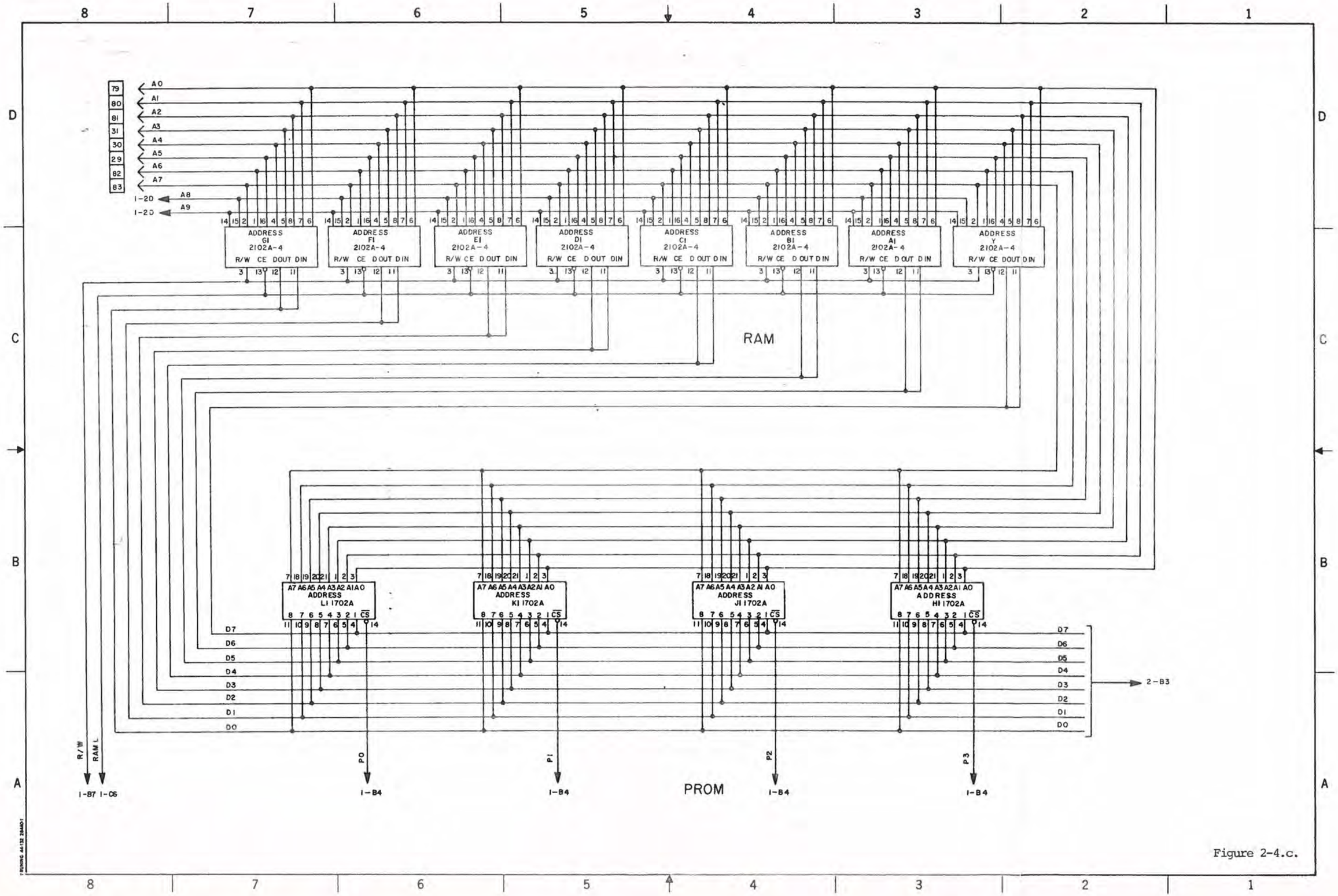

A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
ADDRESS G1 2102A-4
ADDRESS F1 2102A-4
ADDRESS E1 2102A-4
ADDRESS D1 2102A-4
ADDRESS C1 2102A-4
ADDRESS B1 2102A-4
ADDRESS A1 2102A-4
ADDRESS Y 2102A-4
R/W CE DOUT DIN
RAM
ADDRESS L1 1702A
ADDRESS K1 1702A
ADDRESS J1 1702A
ADDRESS H1 1702A
A7 A6 A5 A4 A3 A2 A1 A0
D7
D6
D5
D4
D3
D2
D1
D0
2-B3
R/W
RAM L
1-87
1-C6
P0
P1
P2
P3
1-B4
PROM

Figure 2-4.c.

1) The incoming PROM address is compared with the starting PROM address by the NAND gate IC D. If the incoming address is in the 1K block selected for PROM, the output of IC D, active LOW, enables the PROM address decoder, IC Za, which selects one of the PROM ICs. Up to four PROM chips may be installed. The output of IC D is also combined with the output of the I/O address detector in a NOR gate, the output of which causes the CPU to execute a WAIT state and enables the data bus interface, IC's P and R. The selected PROM drives the bidirectional data bus and the interfaces P and R put the selected data on the Data-In system bus.

2) An address of a byte in RAM is detected by IC B. The output of IC B is active if a start sequence is not in progress and the current machine cycle is not an I/O cycle or an interrupt. Therefore, the only time a RAM address is detected is when the machine cycle is a memory cycle or a Halt cycle. The output of IC B enables the RAM ICs and the data bus interfaces P and R. The direction of the data bus is controlled by the MWRT pulse described in paragraph 2-3F. When RAM is selected, data at the address is put on the bidirectional bus. If the cycle is a memory read, the cycle is complete. If it is a write cycle, the direction of the data bus interfaces is momentarily reversed, overdriving the RAM output drivers. Pulsing the RAM write inputs then causes the data on the bus to be written in the addressed location.

B. AUTO-START. The AUTO-START logic causes the computer to jump to the address designated by the AUTO-START switches when the power is turned on or the START switch is actuated. Figure 2-5 shows the arrangement of the AUTO-START logic. The switches represent the variable byte in a JMP instruction. The JMP instruction is generated by a multiplexer, ICs M and N, which is controlled by flip-flops Ta, Sa and Sb. The flip-flops are cleared by PRESET, a bus signal derived from POC or generated by the START switch on the front panel. Subsequent PDBIN pulses cause the flip-flops to change from one state to the next as shown in the sequence diagram, Figure 2-6. The pulses generated by the flip-flops cause the multiplexer to choose one of three possible bytes; 303 octal, 000 octal or byte designated by the AUTO-START switches.

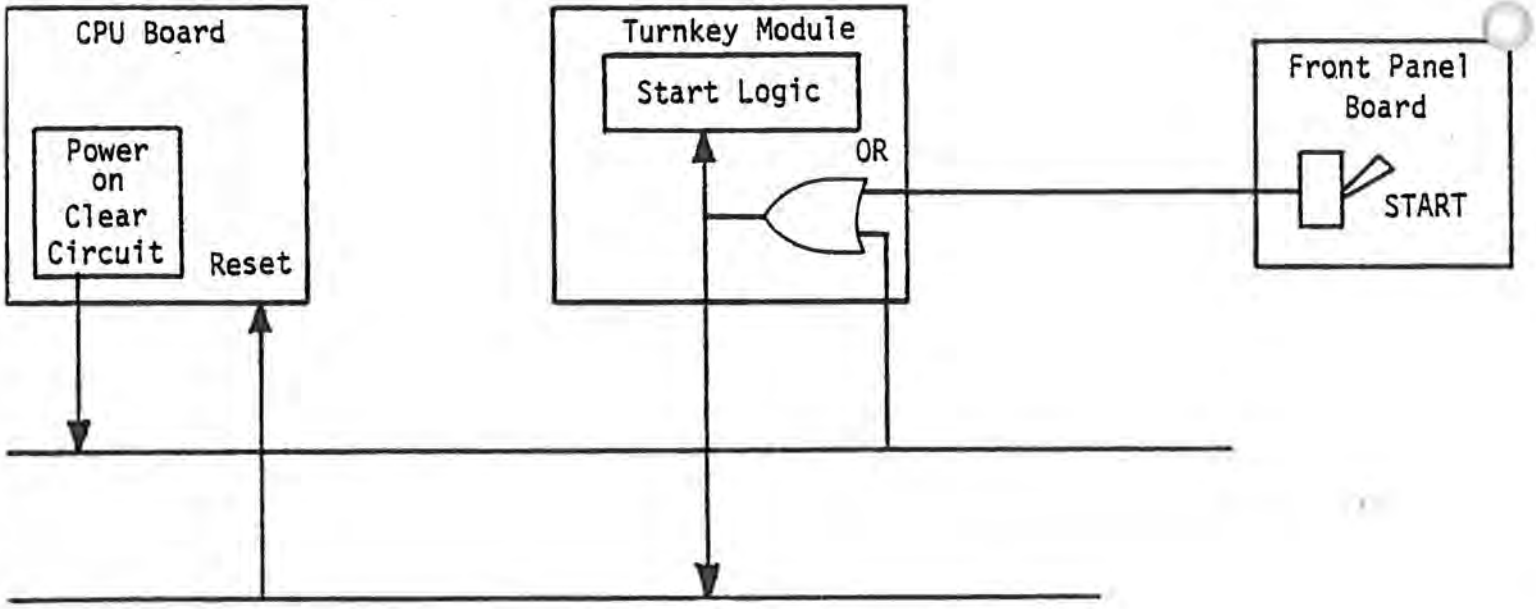


Figure 2-5
AUTO-START Block Diagram

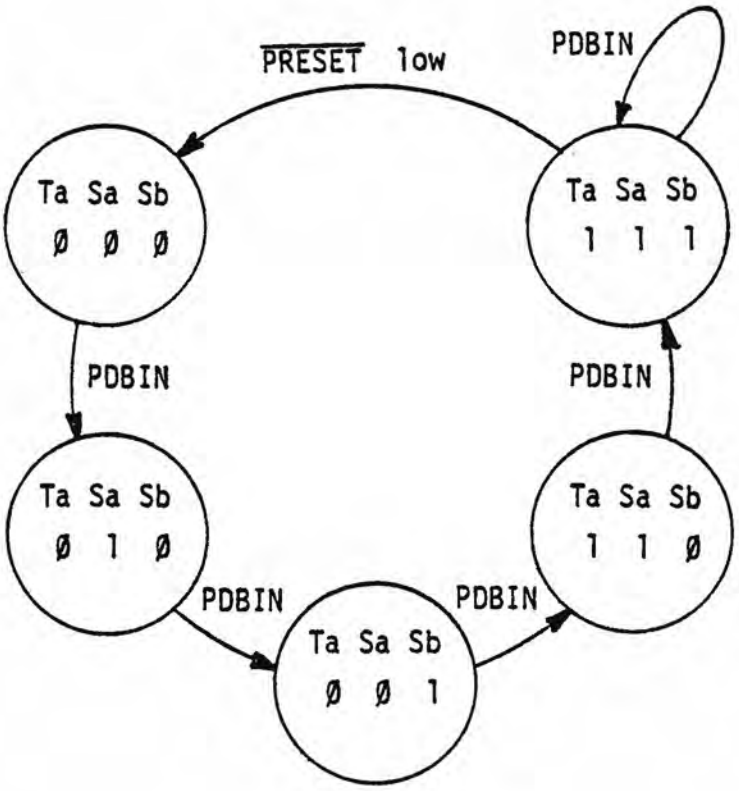


Figure 2-6
AUTO-START Logic
Control State Diagram

These bytes are placed on the bidirectional data bus in sequence and become the AUTO-START JMP instruction. Table 2-A shows the sequence of events in the AUTO-START procedure.

Table 2-A

Signal In	Control State	Function
PRESET	000	Mux. outputs enabled, Bus interface enabled, 3Ø3 octal put on bus.
PDBIN	Ø1Ø	ØØØ put on bus.
PDBIN	ØØ1	Byte in address switches put on the bus.
PDBIN	11Ø	First byte of PROM program
PDBIN	111	Next byte of PROM program
.	.	.
.	.	.
.	.	.

During the three bytes of the JUMP instruction, the HIGH Q output of flip-flop Ta is used to drive transistor Q2 to hold MEMR low and keep memory data off the bus. Once the JUMP instruction is complete, Q goes LOW and memory instructions can be fetched.

C. Sense Switches. The sense switch circuitry is shown in Figure 2-4·a and b. IC K (Figure 2-4a) detects I/O port address 255 decimal (377 octal), which is reserved for the sense switches. The output, active LOW, enables the tri-state buffers connected to the sense switches (Figure 2-4b) putting the bits represented by the switch positions on the bidirectional bus. IC K also enables the bus interface, so the sense switch bits are placed on the Data-In system bus.

D. Serial I/O Channel (SIO). The SIO schematic is shown in Figure 2-4b. The heart of the SIO is the 685Ø Asychronous Communications Interface Adapter. The ACIA contains all the Status and Control registers discussed in section 3-2 and most of the timing and control circuitry. The bit rate is generated by a 34702 integrated circuit whose timing

reference is a 2.4576 MHz crystal. Jumpers are provided to set the output rate of the 34702 at 1, 16 or 64 times the required bit rate.

Input signals are received by RS232 receivers that may be made compatible with TTL or current loop signals by means of jumper-selected pullup resistors. TTL or RS232 outputs may be selected by jumpers or by internal cable wiring.

E. Front Panel. The front panel logic is contained on the Turnkey Module board. The indicators and switches are connected to the Turnkey Module by a cable and Molex connectors. The schematic diagram for the front panel circuitry is in Figure 2-4a.

The bus signals PHLTA, PINTE and $\overline{\text{PINT}}$ are buffered to drive the indicators HALT, INTE and INT, respectively. The I/O indicator is driven by the logical sum (OR) of the INP and OUT signals. The POWER indicator monitors the +5 volt supply on the Turnkey Module.

The bus signal PRDY is grounded when the RUN/STOP switch is in the STOP position. PRESET is grounded momentarily by the START switch which, in turn, initiates the AUTO-START sequence. Contact bounce is filtered out on the CPU board to generate a reset signal suitable for the CPU.

F. Miscellaneous Signals. Several miscellaneous signals are handled by the rest of the circuitry on the Turnkey Module. Some are optional and may be selected by jumpers.

1. MWRT is generated if the jumper from M1 to M2 is installed. In the full front panel version, MWRT is generated by the front panel logic and is used by memory boards of write functions.
2. PROT and UNPROT are used on the standard model of the Altair 8800b for memory protect and unprotect functions. As supplied, the Turnkey Module grounds PROT and pulses UNPROT with phase 2 of the clock to unprotect all memory as it is accessed. This feature may be disabled by removing the jumpers from 2 to UM and from G to PM.

3. AUX CLR is normally pulled HIGH by a resistor on the Turnkey Modlue, but insertion of optional jumpers as described in section 3-4b allows the signal to be used.

G. Power. All power used on the Turnkey Module and Front Panel boards comes from the +18 volt, -18 volt and +8 volt lines on the motherboard. The Turnkey Module's power regulator circuitry is shown in Figure 2-4a. The +5 volt supply is derived from the +8 volt line by an IC regulator. The +9 volt supply comes from the +18 volt line through a zener regulator and the -0 volt supply is derived by a transistor-zener regulator from the -18 volt line.

Figure 2-6 is the schematic for the power circuits in the 88ØØb case. The +18 and -18 volt supplies are pre-regulated. The +8 volt supply is not regulated, but it is adjusted by the taps on the secondary of the power transformer. See Section 3-5.

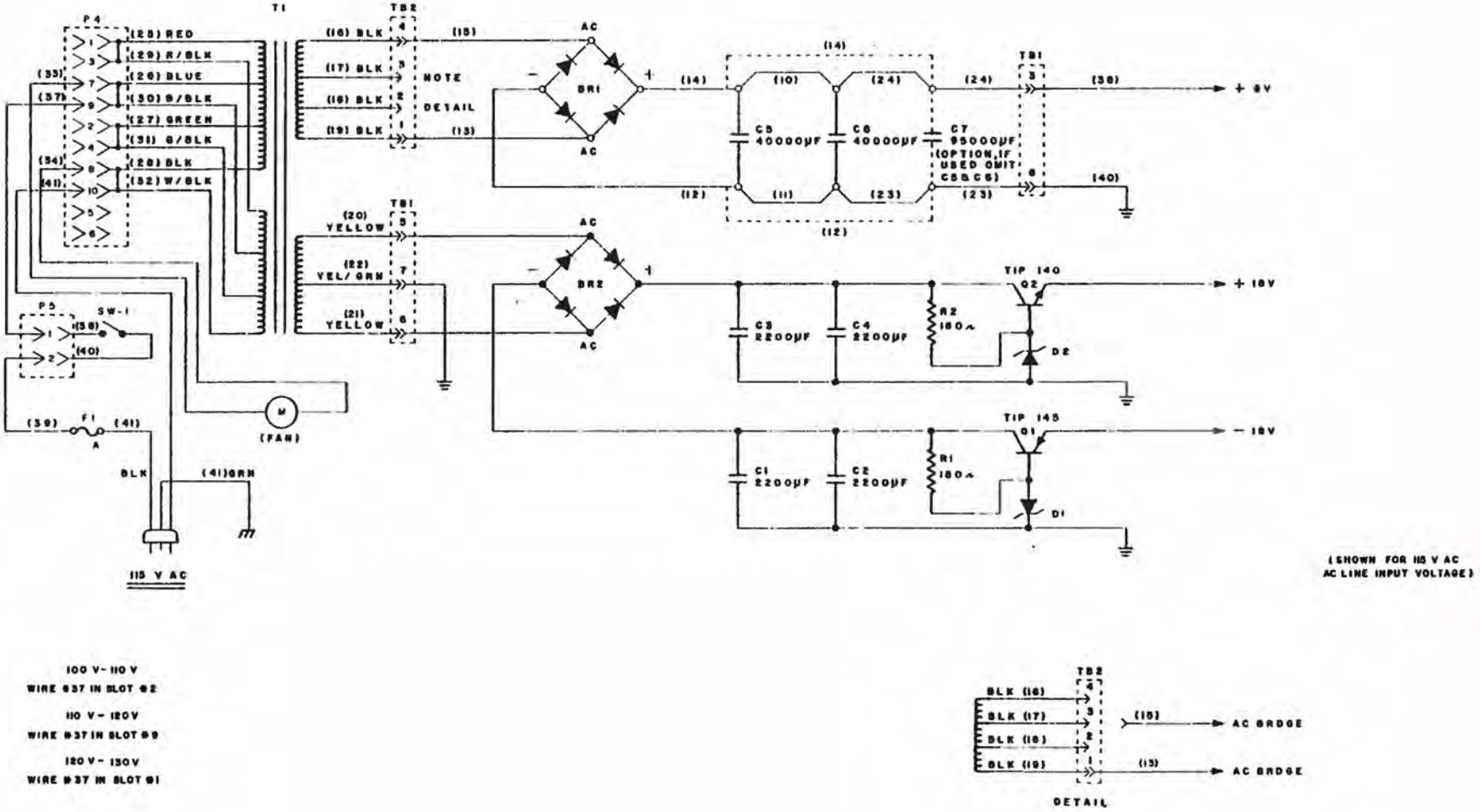


Figure 2-7
Power Supply Schematic

ALTAIR 8800b TURNKEY COMPUTER

SECTION III
ADVANCED OPERATION

3. ADVANCED OPERATION

3-1. The Front Panel

The Front Panel contains the functional switches and status indicators for controlling and monitoring the computer. A list of the switches and indicators along with their functions follows:

Power switch	Turns power on and off. Key-lock switch
POWER indicator	Indicates that power is on. Monitors the +5 volt supply on the Turnkey Module.
RUN/STOP switch	Causes the computer to run the program in memory when placed in the RUN position. Execution stops in the middle of the next machine cycle after the switch is moved to the STOP position. Moving the switch back to RUN continues execution at the point where it was stopped.
START switch	When actuated, stops execution. When released, starts execution at the START address selected by switches on the Turnkey Module. If the switch is actuated when the RUN/STOP switch is in the STOP position, the computer stops in the middle of the START sequence. Moving RUN/STOP to RUN continues the START sequence.
HALT indicator	Indicates that the computer is stopped either because a HALT instruction has been executed or because the PRESET line is LOW.
I/O indicator	Indicates data transfer to or from an I/O port.
INT indicator	Indicates that an interrupt is being requested.
INTE indicator	Indicates that the Interrupt Enable bit is set in the CPU and that the computer may be interrupted. When the INTE indicator is off, interrupt requests have no effect.

3-2. The Turnkey Module

A. Memory. The Turnkey Module contains 1K bytes each of RAM and PROM. The starting address of each 1K block is selected by 6 switches on the Turnkey Module board (Figure 3-1). In operation, the most significant 6 bits of the incoming address are compared with the settings of the switches. If they match, the remaining 1Ø bits are decoded to select the proper byte in that block.

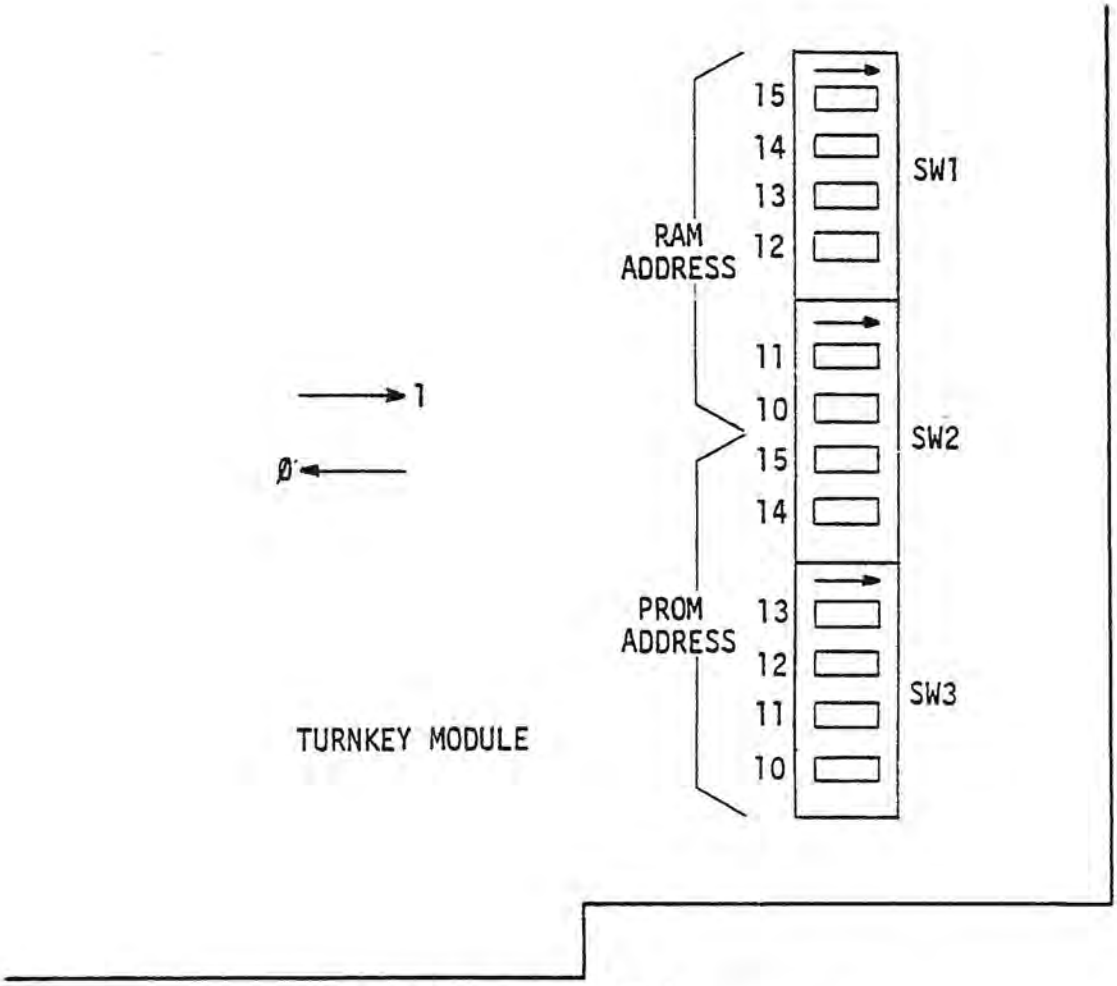


Figure 3-1
RAM and PROM Address Switches

To set the RAM and PROM address switches, first select the starting addresses for each block and convert them to binary. The starting addresses must be integral multiples of 1K (1024) so the low-order (rightmost) 10 bits of the addresses are zeros. The RAM and PROM addresses are zeros. The RAM and PROM addresses cannot be the same, nor may they overlap the addresses of any other memory in the system.

One starting address switch corresponds to each of the 6 high-order bits of the starting address (the most significant bit is bit 15). If the bit is one, the corresponding switch is moved in the direction of the arrow silkscreened on the board. If the bit is zero, the switch is moved in the opposite direction.

B. Sense Switches. There are eight sense switches on the Turnkey Module representing one byte of data. This byte may be read by program instructions and used as data or to select options in the program.

Sense switch settings are described in the documentation for the software products (such as Altair BASIC) that use the switches. Moving the Turnkey Module sense switches in the direction of the silk-screened arrow next to the switches is equivalent to moving a front panel switch up.

To allow use of the sense switches, there must be a jumper between points S and B on the Turnkey Module board as shown in Figure 3-2. If the Turnkey Module is being used in a standard model Altair 88ØØb computer, the front panel switches override the Turnkey Module sense switches. The sense switches may be read by the following instruction:

Assembly	IN 255
Machine (octal)	333 377

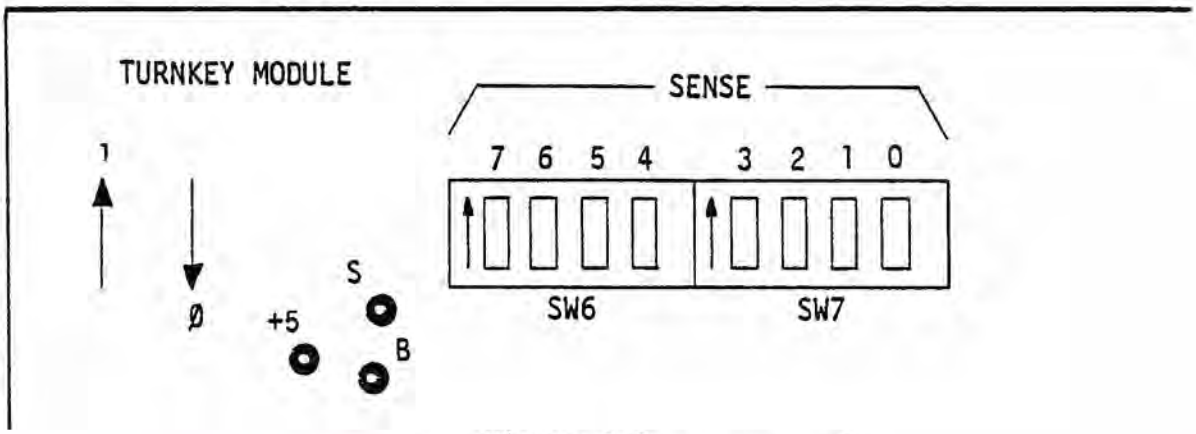


Figure 3-2
Sense Switches and Jumper Pads

C. SIO Section.

1) The Turnkey Module SIO channel appears to the CPU as two of the 256 possible I/O ports (see IMSUM, Section 3, pp. 8 -10). One of the ports is used for data transfer and the other for channel status and control information. Table 3-A is a summary description of the SIO ports.

The high-order seven bits of the I/O address are compared with the SIO address switch settings. If they match, the channel is enabled. The least significant bit selects the Data (bit zero=1) or Status/ Control (bit zero=0) port. To set the switches, convert the desired address to binary. Move each switch in the direction of the silk-screened arrow to represent one and in the opposite direction for zero. Switch 7 represents the most significant bit.

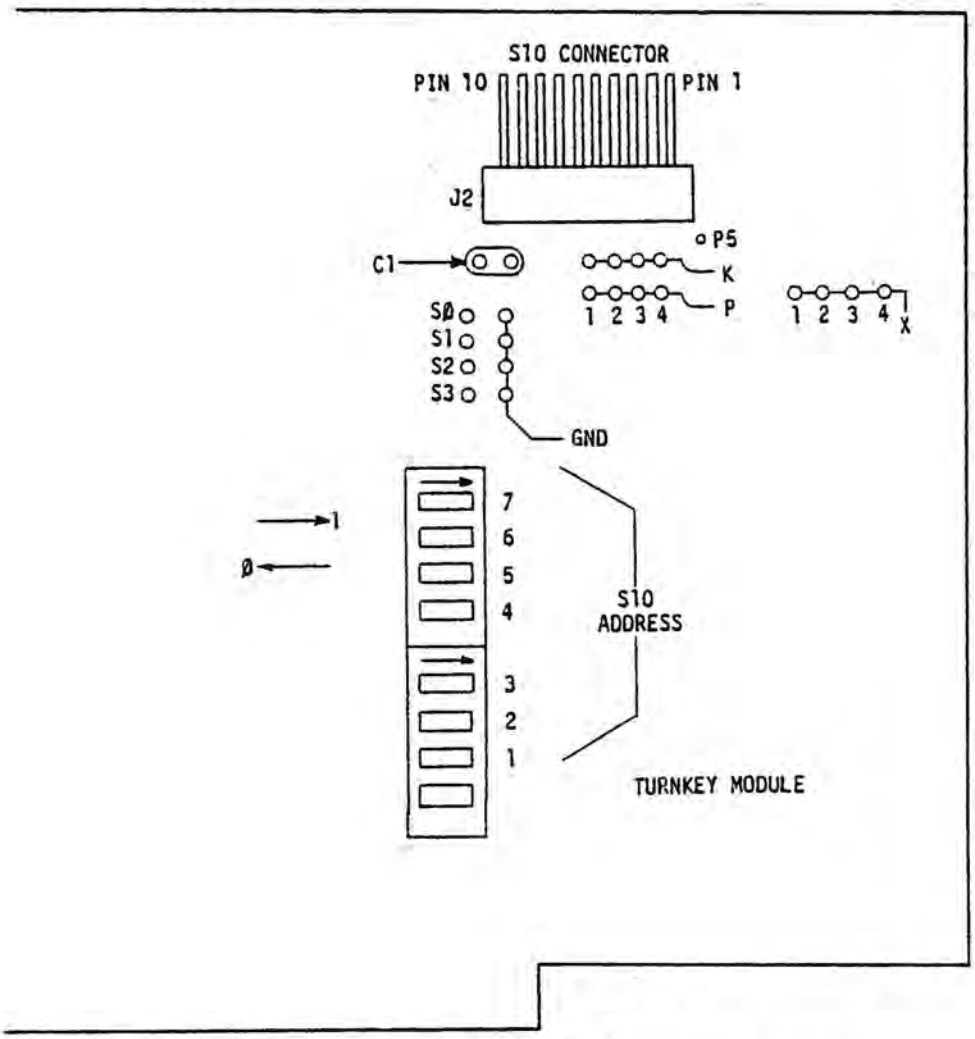


Figure 3-3
SIO Address Switches and Jumper Pads

Table 3-A

Address	Function when read during an Input Operation	Function when loaded during an Output Operation
S_7 S_6 S_5 S_4 S_3 S_2 S_1 1	Receive Data Buffer (Data is stripped of parity)	Transmit Data Buffer
S_7 S_6 S_5 S_4 S_3 S_2 S_1 0	Status	Control
S_1 through S_7 refer to the binary representation of the positions of the SIO address select switches		

2) The Status Register. When the Status/Control port is read during an input operation, the Status Register contains information on the status of the Transmit Data Register, the Receive Data Register and error logic, and the peripheral/modem status inputs. The function of each bit of the status register is given in the table below.

a. Receive Data Register Full (RDRF), Bit Ø. Receive Data Register Full indicates that received data has been transferred to the Receive Data Register. RDRF is cleared after the CPU reads the Receive Data Register or by a master reset. The cleared or empty state indicates that the contents of the Receive Data Register are not current. RDRF also indicates empty if Data Carrier Detect is LOW.

b. Transmit Data Register Empty (TDRE), Bit 1. The Transmit Data Register Empty bit set to 1 indicates that the Transmit Data Register contents have been transferred and that new data may be entered. A zero indicates that the register is full and that transmission of a new character has not begun since the last write data command.

c. $\overline{\text{Data Carrier Detect}}$ ($\overline{\text{DCD}}$), Bit 2. When the DCD input from a modem goes LOW to indicate that a carrier is not present, the $\overline{\text{Data Carrier Detect}}$ bit is set to 1. This setting causes an Interrupt Request to be generated if the Receive Interrupt Enable bit is set. After the DCD input returns HIGH, $\overline{\text{DCD}}$ remains one until it is reset, either by reading first the Status Register and then the Data Register, or by a master reset. If the DCD input remains LOW after the Status and Data Registers have been read or a master reset occurs, the $\overline{\text{DCD}}$ bit remains the inverse of the DCD input.

d. $\overline{\text{Clear-to-Send}}$ ($\overline{\text{CTS}}$), Bit 3. The $\overline{\text{Clear-to-Send}}$ bit is the inverse of the Clear-to-Send input from a modem. Thus, zero in $\overline{\text{CTS}}$ indicates that there is a Clear-to-Send from the modem. When the Clear-to-Send signal is LOW, the Transmit Data Register Empty bit is inhibited and the $\overline{\text{CTS}}$ status bit is set to one. Master reset does not affect the $\overline{\text{CTS}}$ bit.

e. Framing Error (FE), bit 4. A framing error occurs when the received character is improperly framed by a start and a stop bit. It is detected by the absence of the first stop bit. This indicates a synchronization error, faulty transmission or a break condition. The Framing Error flag, FE, is set during the receive data transfer time. Therefore, the error indicator is present throughout the time that the associated character is available.

f. Receiver Overrun (OVRN), bit 5. OVRN is an error flag that indicates that one or more characters in the data stream were lost. That is, a character or a number of characters were received but not read from the Receive Data Register (RDR) before subsequent characters were received. The overrun condition begins at the midpoint of the last bit of the second character received without the RDR having been read. The OVRN bit is not set in the Status Register until the valid character prior to the overrun has been read. The RDRF bit remains set until OVRN is reset. The OVRN bit is reset when data is read from the Receive Data Register or by the master reset. Character synchronization is maintained during the overrun condition.

g. Parity Error (PE), bit 6. The Parity Error flag indicates that the number of ones in the character does not agree with the preselected parity. Odd parity is defined as the condition in which the total number of ones in the character is odd. Even parity means the number of ones is even. The parity error indication is present as long as the data character is in the RDR. If no parity is selected, then both the transmitter parity generator output and the receiver parity check results are inhibited.

h. Interrupt Request (IRQ), bit 7. IRQ indicates the state of the IRQ signal. Any interrupt condition, with its applicable enable, is indicated in this status bit. Anytime the IRQ signal is LOW, the IRQ bit is one to indicate the interrupt or service request status. Section 2-2.C.4. shows how to jumper the IRQ signal to an interrupt line on the bus.

3) The Control Register. When the Status/Control port is loaded during an output operation, the Control Register contains information which controls the functions of the receiver and transmitter, interrupt enables and the Request-to-Send peripheral/modem control input. The Control Register bits and their functions are shown below.

a. Counter Divide Select, bits Ø and 1. The Counter Divide Select Bits determine the clock divide ratios used in both the transmitter and receiver sections of the SIO. Additionally, these bits are used to provide a master reset for SIO which clears the Status Register (except for external conditions on the CTS and DCD lines) and initializes both the receiver and transmitter. Master reset does not affect other Control Register bits. Note that after power-on or a power failure/restart, these bits must be set HIGH to

reset the SIO. After reset, the clock divide ratio may be selected. The counter select bits provide for the following clock divide ratios:

CR1	CRØ	Ratio
Ø	Ø	1 (synchronized)
Ø	1	16 (normal)
1	Ø	64 (slow)
1	1	master reset

Section 3-2.C.5. describes the use of these bits to select the baud rate.

b. Word Select, bits 2, 3 and 4. The Word Select Bits are used to specify word length, parity and the number of stop bits in each character. Word length, Parity Select, and Stop bit changes are not buffered and therefore become effective immediately. The encoding format is as follows:

CR4	CR3	CR2	Function
Ø	Ø	Ø	7 bits, even parity, 2 stop bits
Ø	Ø	1	7 bits, odd parity, 2 stop bits
Ø	1	Ø	7 bits, even parity, 1 stop bit
Ø	1	1	7 bits, odd parity, 1 stop bit
1	Ø	Ø	8 bits, two stop bits
1	Ø	1	8 bits, one stop bit
1	1	Ø	8 bits, even parity, one stop bit
1	1	1	8 bits, odd parity, one stop bit

c. Transmitter Control, bits 5 and 6. Two Transmitter Control Bits provide for control of the interrupt from the Transmit Data Register Empty condition, the Request-to-Send output, and the transmission of a break level (space). The setup of interrupt jumpers are shown in Section 3-2.C.4. The following coding is used:

CR6	CR5	Function
Ø	Ø	RTS = HIGH, Transmitting Interrupt disabled
Ø	1	RTS = HIGH, Transmitting Interrupt enabled
1	Ø	RTS = LOW, Transmitting Interrupt disabled
1	1	RTS = HIGH, transmits a break level on the transmit data output, Transmitting disabled.

d. Receive Interrupt Enable, bit 7. Receive Data Register Full, Overrun and Data Carrier Detect interrupt requests are enabled by a 1 in the Receive Interrupt Enable Bit. The setup of interrupt jumpers is shown in Section 3-2.C.4.

4) Interrupts. The SIO generates an interrupt signal that may be connected to an interrupt line on the bus by means of jumpers. The jumper pads are shown in Figure 3-4.

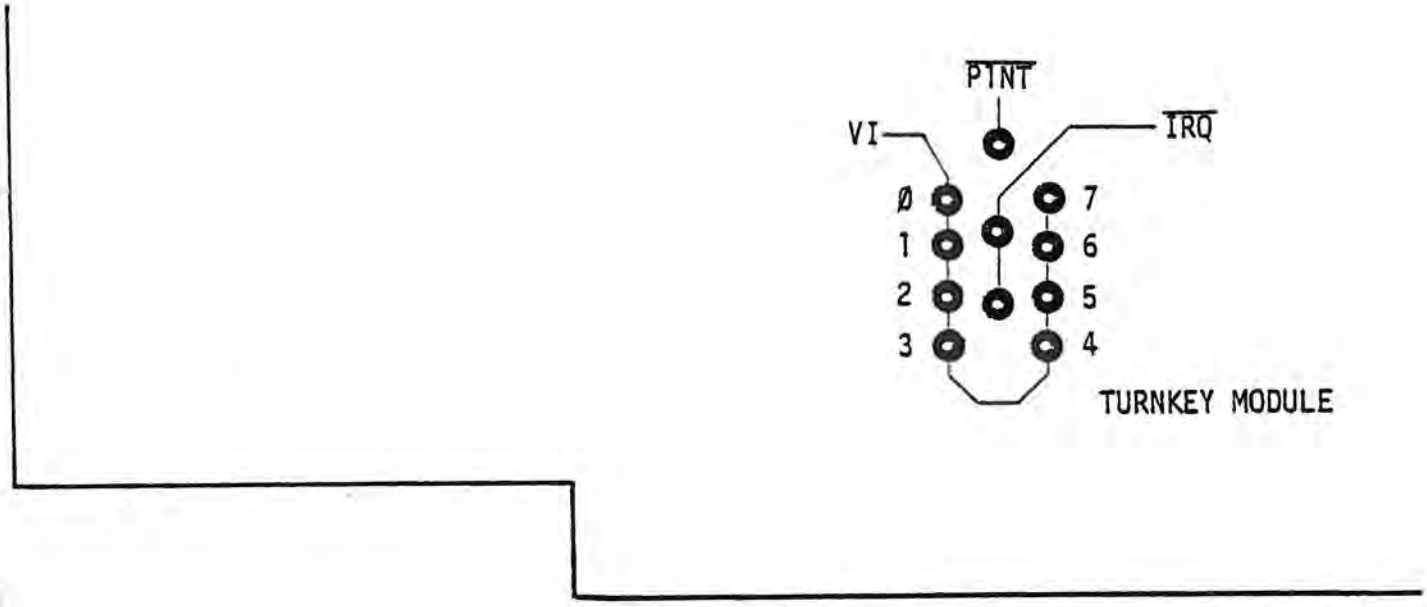


Figure 3-4
SIO Interrupt Jumper Pads

a. No jumper is installed if the SIO is not required to interrupt the CPU. Software can detect Status Register bits in this case and direct I/O operations without interrupts.

b. In systems not using the Vector Interrupt board, the signal IRQ may be connected to the $\overline{\text{PINT}}$ line. Then, if an SIO interrupt occurs, $\overline{\text{PINT}}$ will be pulled LOW until the condition that caused the interrupt no longer exists. If the Interrupt Enable bit in the CPU is set, then a RST 7 instruction is forced into the instruction sequence. Interrupt signals from other I/O circuits may also be connected to the $\overline{\text{PINT}}$ line if their outputs are effectively driven by open collector drivers. All Altair I/O boards currently manufactured and supported have open collector interrupt signal drivers. See the appropriate I/O board manual for more information.

c. In systems using the Vector Interrupt board, IRQ may be connected through a jumper to any pad marked VIØ through VI7. These represent the 8 interrupt priority levels. Interrupt signals from different boards may be connected to the same priority level if they are all driven by open collector drivers. See the Vector Interrupt board manual for further information.

5) Bit Rate Selection. In this section, 'bit rate' is defined as the maximum rate of level changes on the data signal line. Since the SIO is an asynchronous device, the bit rate determines the rate at which the bits within each character are received or sent, but not, in general, the average rate at which characters are handled.

The bit rate is selected by jumpers and by Control Register bits 1 and Ø. Table 3-B shows the resultant bit rate for every usable combination of jumpers and control bits. The jumpers SØ, S1, S2 and S3 are shown in Figure 3-3. If CRØ and CR1 are both zero and the external rate is not selected, then the SIO may only be used for transmission. Otherwise, the SIO may be used for both receiving and transmitting at the selected rate. If a rate above 3ØØ bits per second is selected, capacitor C1 should be removed.

If the external clock and the 1 counter are selected, then the data must be synchronized with the clock. The transmitted data line changes levels within one microsecond of the LOW to HIGH clock transition. Output jitter is about 5ØØ ns. The SIO will sample the receive data line within 1 microsecond after the HIGH to LOW clock transition.

Table 3-B

Data Transmission Rates						
Jumpers (X means installed)				CR0 = 1 CR1 = 0 (normal; 16 counter selected)	CR0 = 0 CR1 = 1 (slow; 64 counter selected)	CR0 = 0 CR1 = 0 (sychronous data; 1 counter selected)
S3	S2	S1	S0			
-	-	-	-	110	27.5	1760
-	-	-	X	150	37.5	2400
-	-	X	-	300	75	4800
-	-	X	X	2400	600	38400
-	X	-	-	1200	300	19200
-	X	-	X	1800	450	28800
-	X	X	-	4800	1200	76800
-	X	X	X	9600	2400	153600
X	-	-	-	2400	600	38400
X	-	-	X	600	150	9600
X	-	X	-	200	50	3200
X	-	X	X	134.5	33.375	2152
X	X	-	-	75	18.75	1200
X	X	-	X	50	12.5	800
X	X	X	-	External 16 rate	External 64 rate	External rate
X	X	X	X	(36,000 max)	(9000 max)	(400,000 max)

6) Signal Types. The SIO can be configured to interface with peripheral equipment using 20 mA loop, RS232C and TTL signals. These signal options are selected by jumpers and internal cable connections. Table 3-C shows the I/O signal types. The locations of the jumpers are shown in Figure 3-3. The cable runs from the I/O connector on the Turnkey Module to the rear panel. The rear panel connector is the industry standard 25-pin data communications connector.

Table 3-C

Signal Type	From	To	Notes
TTY Compatible	X1	X2	
	K4	P5	
	K3	P3	
	K2	P2	
RS232 Compatible	X3	X4	
	K3	P3	Put in only if DCD signal is not used.
	K2	P2	Put in only if CTS signal is not used
TTL Compatible (3.2 mA max load 16 mA min drive)	X2	X3	
	K4	P4	
	K3	P3	
	K2	P2	
	K1	P1	Not needed if external clock is not used.

Note: TTL inputs are two unit loads; input is actually "TTL compatible."

Table 3-D shows the connections of the internal cable.

D. Interfacing. This section describes interfacing of the SIO through the 25 pin rear panel connector to modems, RS 232 terminals or current loop terminals. The terminal's instruction book should be consulted for the proper choice of signal types.

Table 3-D

From Molex		To 25 Pin Connector		
Pin Number	Function	Female TTY Cable	Male RS232 Cable	Female TTL Cable
1	TTL RTS	6		4
2	TTY XMIT	3		
3	TTY REC	4		
4	A11 REC	5	2	2
5	DCD		8	8
6	CTS		5	5
7	XTERNAL CLOCK		15	15
8	GND	2	7	7
9	RS232 RTS		4	
10	RS232 + TTL XMIT		3	3

1) Connection to a modem is through a male to female extension cable. The board and internal cable are set up for RS-232 I/O.

2) To connect to a Teletype, first unplug the Teletype, then loosen the three thumb-screws in the back and remove the roll of paper, the Mode Switch knob and the face plate. Remove the four screws under the nameplate and the small screw on the reader cover. The cover can now be removed.

The interconnection between the computer's SIO and the Teletype is shown in Figure 3-5. Connection is made to terminal strip 1514111, which is at the right rear of the Teletype.

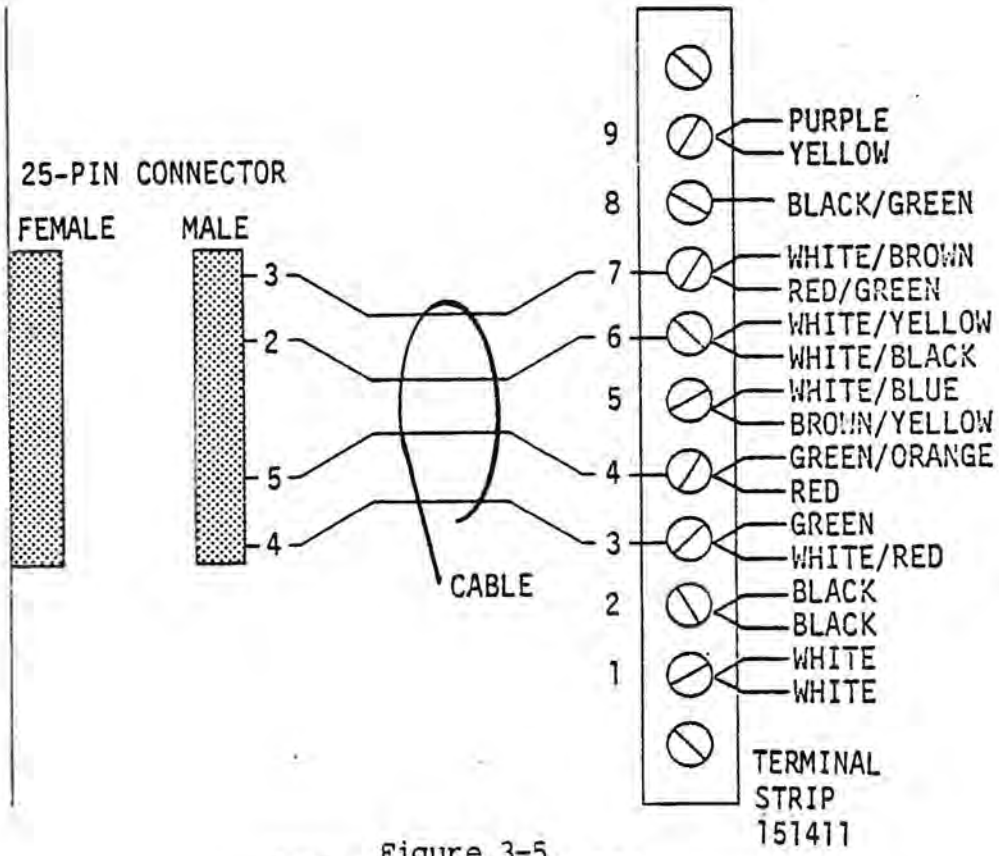


Figure 3-5
20mA Current Loop (TTY)
I/O Connections

While the cover is off, check that the Teletype is wired for 20mA, full duplex operation. The unit is set for full duplex if the Brown/Yellow wire is on terminal 5 and not on 3, and the White/Blue wire is on terminal 5 and not on 4 of terminal strip 1514111. The receiver current level is set to 20 mA if the Purple wire is connected to terminal 9 and not to 8. The correct connections are shown in Figure 3-5. The current source resistance for local mode should be connected as 1450 ohms. This resistor is on the right hand side of the Teletype, halfway back.

After the connections and modifications are made, replace the cover, faceplate, knobs, paper roll and screws.

Be sure that the Turnkey Module and the internal cable have been set up for TTY I/O.

3) For RS-232 input/output, the cable is wired as shown in Figure 3-6. Note that the cable is symmetrical. The SIO of the Turnkey Module and the internal cable must be set up for RS-232 I/O.

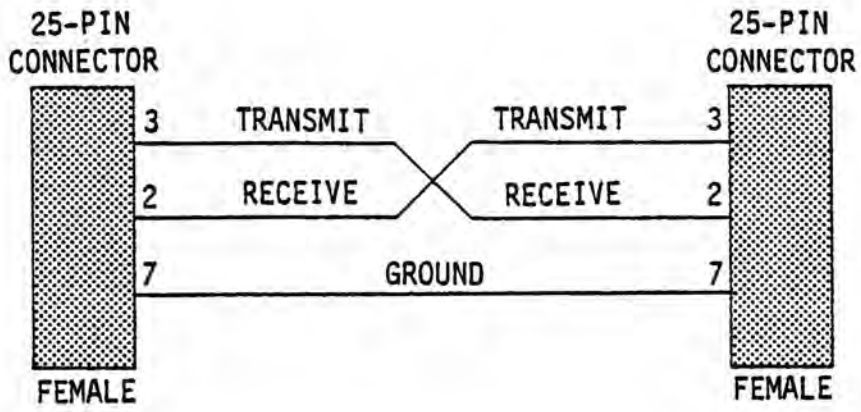


Figure 3-6
RS-232 I/O Connections

3-3. AUTO-START

When power is turned on, or when the START switch is released, the start sequence logic forces the CPU to begin executing instructions at an address selected by a set of switches on the Turnkey Module. The switches are shown in Figure 3-7.

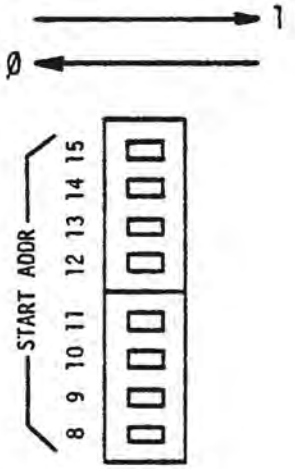


Figure 3-7
AUTO-START Address Switches

The AUTO-START switches are set in the same manner as the RAM and PROM address switches (see section 3-1). The AUTO-START address is the address of the first location of a routine in PROM. The address must be an integral multiple of 256, so the low order eight bits must be zeros. The eight AUTO-START switches correspond to the high-order eight bits of the AUTO-START address. Bit 8 is the least significant bit.

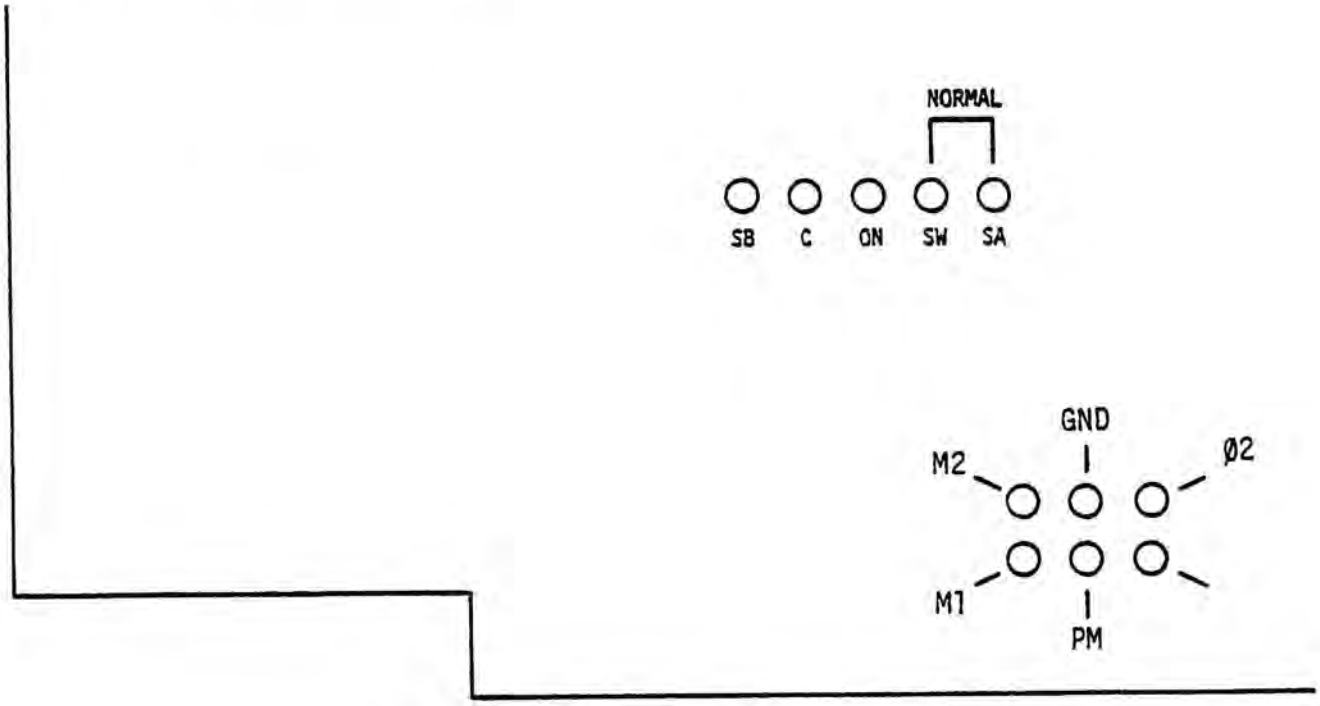
If the 88ØØb Turnkey Monitor PROM is installed, the AUTO-START address must be 176ØØ octal. Therefore, all the AUTO-START switches except switch 9 must be in the "1" position.

3-4. Miscellaneous Options

A. Use of Turnkey Module with Front Panel Model. The Turnkey Module may be used in the standard (full front panel) model of the Altair 88ØØb computer. To do this, the following jumpers must be removed from the Turnkey Module board:

M1 to M2
PM to GND
Ø2 to UM

Figure 3-8 shows these jumpers.



Section 3-8
Miscellaneous Option Jumper Pads

Table 3-E

Configuration	Jumpers	Cause	Effect
Normal	SA to SW	Power On	$\overline{\text{POC}}$ pulse START
		START switch	START
Alternate #1	SA to SW SB to C	Power On	$\overline{\text{POC}}$ pulse START $\overline{\text{AUX CLR}}$ pulse
		START switch	START $\overline{\text{AUX CLR}}$ pulse
Alternate #2	ON to SW	Power On	$\overline{\text{POC}}$ pulse START
		START switch	$\overline{\text{POC}}$ pulse START
Alternate #3	ON to SW SB to C	Power On	$\overline{\text{POC}}$ pulse START $\overline{\text{AUX CLR}}$ pulse
		START switch	$\overline{\text{POC}}$ pulse START $\overline{\text{AUX CLR}}$ pulse
Alternate #4	SA to SW ON to C	Power On	$\overline{\text{POC}}$ pulse START $\overline{\text{AUX CLR}}$ pulse
		START switch	START

B. POC and AUX CLR options. As supplied, the computer generates the POC pulse when the power is turned on, but does not generate the AUX CLR pulse. This may be changed by jumpers between the pads marked C, ON, SA, SB and SW. These pads are shown in figure 3-8. Table 3-E shows the options available. The AUX CLR pulse is generated by a panel switch on the standard model of the 88ØØb computer and may be used by peripheral devices.

3-5. The Power Supply

A. Adjusting for differing loads. The +18 and -18 volt supplies are pre-regulated, but the +8 volt supply must be adjusted for differing loads by moving the tap on the power transformer secondary. The transformer secondary taps are shown in Figure 3-9. The correct secondary tap

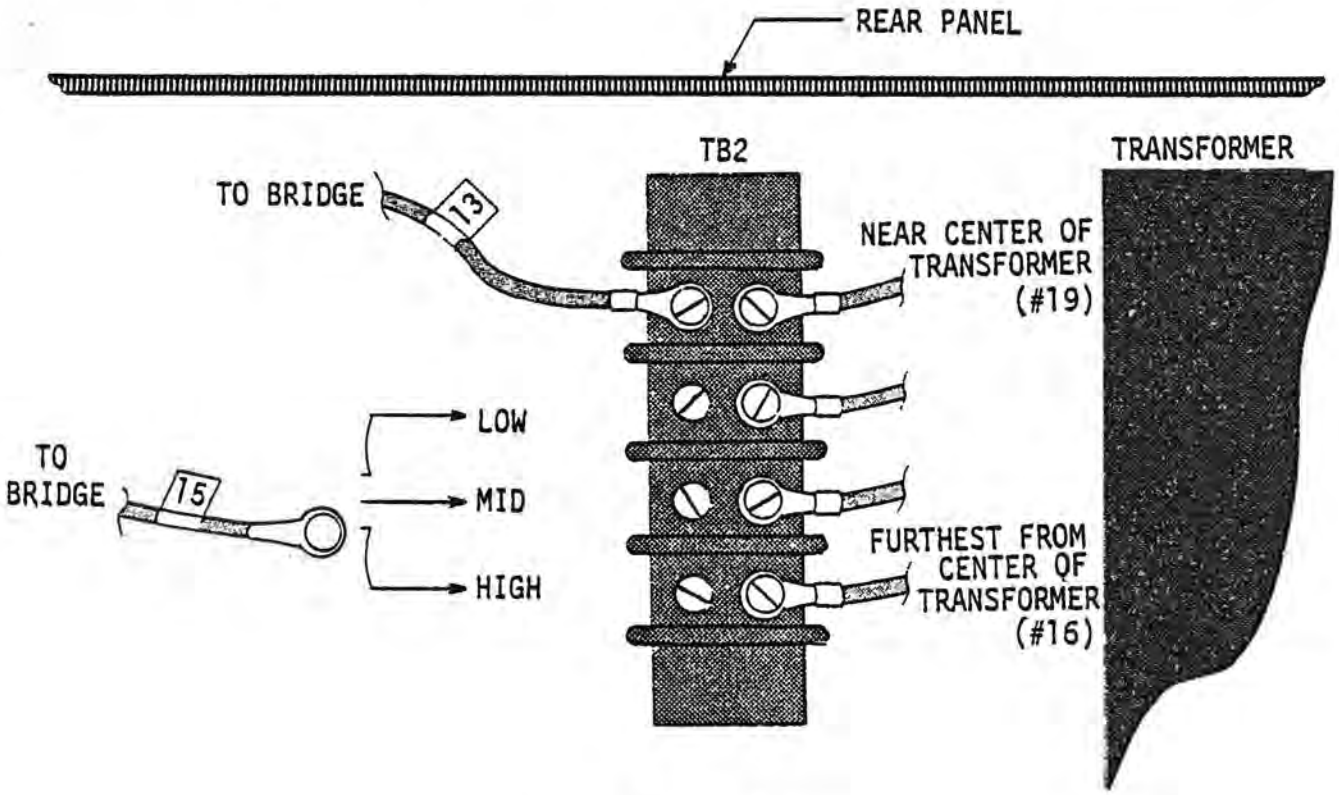


Figure 3-9
Power Transformer Secondary Taps

is determined experimentally. The procedure for adjusting the tap is as follows:

1. Disconnect the computer from its A.C. supply.
2. Connect wire 15 from the bridge rectifier to one of the secondary taps on the transformer as shown below:

Number of Boards in the Computer	Transformer Tap
Ø-6	low
7-12	middle
13-18	high

3. Connect the A.C. supply and measure the voltage between pins 1 and 5Ø on the motherboard.
4. If the measured voltage is greater than 9 volts, the next lower tap should be used if it is available. If the voltage is less than 7.5 volts, the next higher tap should be used. Be sure to disconnect the power cord before moving the transformer tap connection.

B. Power Supply capacity. The power supplies can power most systems that can be accommodated in the case. Only an unusually large system will tax the resources of the power supplies.

The A.C. line into the power supply is furnished with a 3 amp, slow-blow fuse. It should be replaced as necessary only with an identical fuse.

ALTAIR 8800b TURNKEY COMPUTER

SECTION IV
TROUBLESHOOTING

4. TROUBLESHOOTING HINTS

4-1. Introduction

This section is not an extensive troubleshooting guide, but rather contains guidelines that can save the troubleshooter time. Knowledge of electronics and of the contents of Sections 2 and 3 of this manual is required for troubleshooting.

A. Equipment. An oscilloscope with 30 MHz or greater bandwidth is normally required to troubleshoot this unit. A voltage meter may be required in some measurements. In most cases, either a logic analyzer or a full panel version of the Altair 8800b computer is also needed. If the Turnkey Module is used with the front panel, remove jumpers as described in paragraph 3-9. To display both address lines and data lines on the logic analyzer, clock on the falling edge of IC W pin 8 on the Turnkey Module. To view data going to the Data Out bus as well as the Data In bus, make connections to the bi-direction bus on the CPU board.

B. Troubleshooting Optional Boards. Troubleshooting optional memory and I/O boards is covered in the manuals for those boards. The directions in these manuals often assume that a full front panel is used, presenting test loops to be executed in the single step mode. If a full front panel is not used, use a logic analyzer and a test program instead.

C. Trouble Follows Change. An error in system change, jumper installation, repair or board modification can cause trouble. If a system change has been made, be sure that all Turnkey Module jumpers are installed as shown in Section 3. If boards were added, be sure that the +8 volt bus has the correct voltage. Check that all boards and connectors are well seated and mated correctly. If the trouble follows jumper installation, repair or board modification, check for solder bridges and poor solder connections. If the trouble follows repair, check that the IC pins were inserted into the sockets properly.

4-2. Preliminary Considerations

A great deal of time can be saved by some preliminary checks.

A. First, check that all connectors and boards are well seated, that the unit is plugged in and that the fuse is not open.

B. Corner the problem by removing boards from the unit and by swapping boards with good ones, if possible. Use this method with caution; a board with a damaged bus driver may look OK when a good board is removed from the system, because the load is reduced. If a board fails catastrophically, several boards may be damaged along with cabling and power wiring. In this case, each board should be tested in another system.

C. Check for incorrect voltage and noise on the regulated voltages of every board plugged into the motherboard. The tolerances for the regulated voltages generated on the CPU board and the Turnkey Module are tabulated below.

Board	Voltage	Tolerance
CPU	+12	5%
	+5	5%
	-5	5%
Turnkey Module	+5	5%
	+9	10%
	-9	-1.Ø volt
		+Ø.5 volt

D. Check the clock on the CPU board. This may be monitored at IC J pin 3. The correct frequency is 2.Ø MHz.

4-3. CPU

The CPU board requires less troubleshooting than the circuits on the Turnkey Module. When troubleshooting the CPU, monitor TTL buffered signals rather than MOS driven signals as much as possible, to avoid loading the 8Ø8ØA IC. When monitoring the 8Ø8ØA outputs, use a low capacitance XIØ probe. See IMSUM (section 5, pp. 13-19, 163-166, 1-4) for nominal signals and timing measurements for the 8Ø8ØA, 8216 and 8224 ICs.

4-4. Turnkey Module

A. AUTO-START. AUTO-START should be one of the first functions tested if there are no other clues to the trouble. AUTO-START is tested with a front panel as shown in the table below. Depressing the RESET or STEP switch should produce the displays shown in the table.

Press Switch	Address Indicators	Data Indicators	Status Indicators
RESET	Ø	3Ø3	M1
STEP	1	ØØØ	none
STEP	2	setting of start address switches	none
STEP	start address	-----	M1,MEMR

If a logic analyzer is used instead of a front panel, depress the START switch on the Turnkey front panel and monitor at running speed.

B. PROM. If AUTO-START is operating properly and the computer still does not start, there may be a problem with the PROM circuitry. Most PROM problems can be found by stepping through the AUTO-START sequence and then through the program, monitoring key signals on the Turnkey Module. It should be possible to read every PROM location from the front panel. Problems may be masked when stepping through PROM, though, by the logic which generates WAIT states. The most common PROM problems are bent PROM IC pins, loss of -9 volts, bad PROM ICs, trouble in the wait state logic and incorrect address setting.

C. RAM. Correctly functioning RAM can be read from and written into from the front panel. The most common RAM problems are incorrect setting of the starting address switches and bad RAM ICs.

D. Sense Switches. Because software uses the sense switches to make decisions, trouble with the sense switches may look like trouble elsewhere. The front panel sense switches on the Altair 88ØØb computer override the sense switches on the Turnkey Module even if the logic on the Turnkey Module does not work correctly. So, if the trouble disappears when the front panel is used, check the sense switches as well as the logic that generates the MWRITE pulse.

E. Serial I/O Channel. Watch out, the trouble may not be here. A Teletype may run open if the SIO is not initialized because of trouble in the PROM circuitry, the logic for the sense switches, the AUTO-START logic or the CPU. Trouble with the sense switches may also cause the SIO channel to seem to be malfunctioning in other ways.

1) To check out the SIO, monitor a character echo loop, either by stepping through it using a front panel, by using a logic analyzer or, in a limited way, by using a different SIO channel to control a debug routine. As the status register is input, verify the correctness of each bit. Note that DCD should be Ø. Then, by examining the operation of the loop, decide if the problem is in the transmitter or the receiver.

2) The output of the baud rate generator can be checked at pin 1Ø of IC G. It should be a square wave with a frequency 16 times the selected baud rate. For verification on the oscilloscope, the periods for some popular baud rates are shown below:

Baud Rate	Clock Period
11Ø	568 µs
3ØØ	2Ø8 µs
12ØØ	52 µs

3) The most common sources of SIO problems are incorrect jumper installation, incorrect cabling, trouble in the wait state logic, damaged MOS ICs and catastrophic damage due to high voltages on the I/O lines.

ALTAIR 8800b TURNKEY COMPUTER

SECTION V
ASSEMBLY

5-1. Introduction and Preparation For Assembly

This section contains instructions for circuit and mechanical construction of the Altair 8800b Turnkey Computer. It includes hints for assembly, detailed component installation instructions, and printed circuit board and mainframe assembly instructions.

A. Assembly Hints and Necessary Supplies

Before beginning to assemble your 8800b Turnkey it is important to read the "MITS Kits Assembly Hints" booklet included with your kit. Carefully read the section concerning soldering. Most problems occur as a result of poor soldering techniques. Be sure to use the correct type of soldering iron. A 25-30 watt iron with a chisel tip is essential for all printed circuit assemblies. An Unger 776 with a 7155 tip is recommended. These can be purchased at most electronic supply stores. The assembly hints booklet will also provide a list of other necessary tools.

> NOTE
>
> IMPORTANT WARNINGS ARE INCLUDED IN THE ASSEMBLY HINTS BOOKLET. READ THEM CAREFULLY BEFORE BEGINNING ANY WORK ON THE UNIT. FAILURE TO HEED ANY OF THESE SPECIFIC WARNINGS COULD RESULT IN A DAMAGED UNIT AND/OR VOIDED WARRANTY.

B. Parts List

Check the contents of the kit against Appendix C (Parts List) to make sure all necessary components are contained in the plastic envelopes; do not open the envelope until the component is needed in an assembly step.

It is important to assemble the kit as per instructions in this manual and to follow the instructions in the order in which they are presented. Always complete each section before going on to the next.

This manual includes two aids to help organize the assembly process: (1) boxed off parts identification lists in boxes with spaces provided to check off components as they are installed; (2) reproductions of board silkscreens showing previously installed components (black), components currently being installed (screened) and components yet to be installed (white). Figure 5-1 is an example of a typical silkscreen.

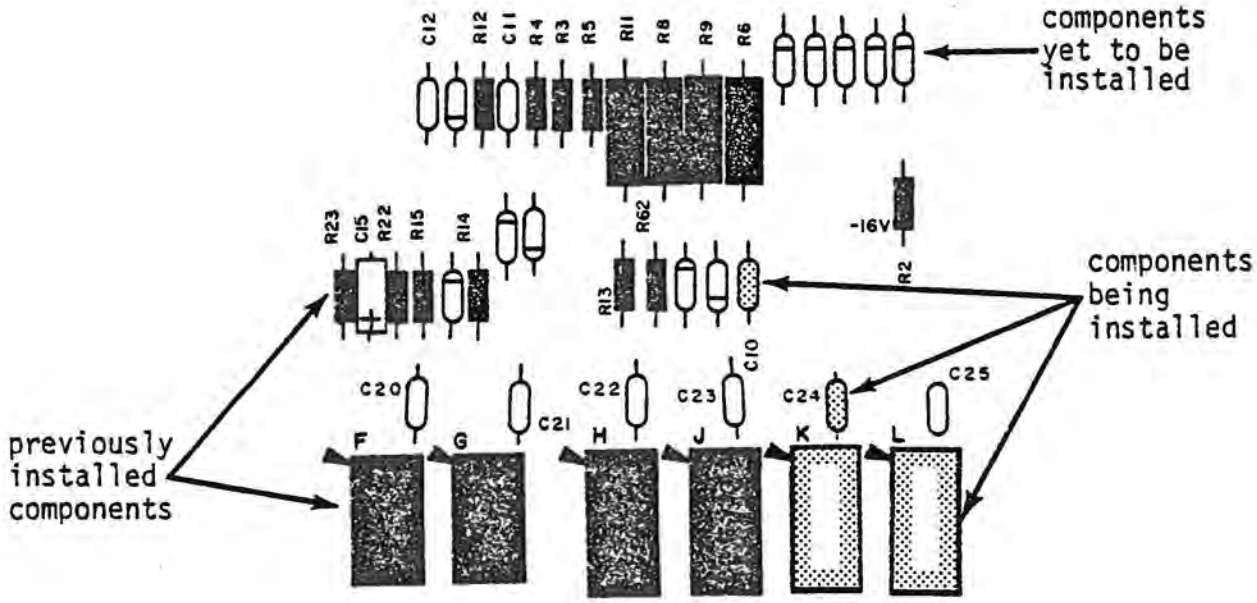


Figure 5-1 Typical Silk Screen

C. Printed Circuit Board Inspection

It is recommended that all PC boards be carefully inspected before beginning assembly. Look for "etching bridges" depicted in Figure 5-2 or "etching opens" as illustrated in Figure 5-3. The "open" may also appear as a "hair-line" cut.

A thorough visual inspection, preferably with a magnifying glass and strong light, will eliminate one area in which problems may occur. Should the board fail to operate properly after assembly, troubleshooting efforts may then be concentrated elsewhere.

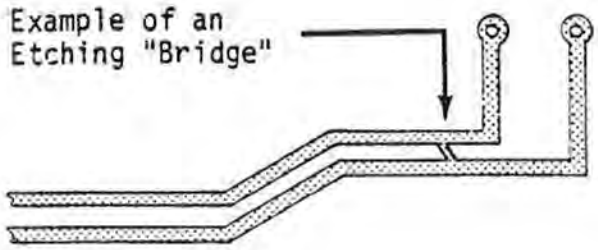


Figure 5-2
Example of an Etching "Bridge"

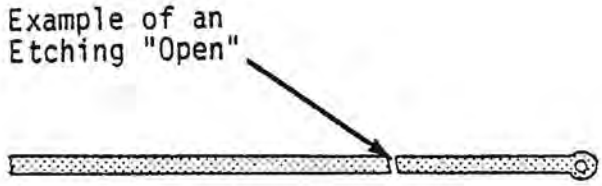


Figure 5-3
Example of an Etching "Open"

5-2. Component Installation Instructions

This section describes the correct procedures which must be followed when installing various components (resistors, diodes, capacitors, etc.) included in the kit. Read these instructions carefully. Subsections within the manual (i.e. CPU board assembly) will not describe installation of each component. The reader will be referred to the section describing the installation of a specific component. Failure to properly install components may cause permanent damage to the component and other parts of the unit; should such damage occur the warranty will be voided.

Specific instructions or procedures of a less general nature will be included within the assembly text itself. Read each set of assembly instructions before beginning the actual construction. Do not continue until every step of assembly is fully understood. This will prevent potential problems and possible failures on the board.

A. Resistor Installation Instructions

Resistors have four (or possibly five) color-coded bands as represented in the chart below. The fourth band is gold or silver and indicates the tolerance. NOTE: In assembling a MITS kit, you need only be concerned with the three bands of color to the one side of the gold or silver (tolerance) band. These three bands denote the resistor's value in ohms. The first two bands correspond to the first two digits of the resistor's value and the third band represents a multiplier.

For example: a resistor with red, violet, yellow and silver bands has a value of 270,000 ohms and a tolerance of 10%. By looking at the following chart, note that red is 2 and violet 7. Multiplying 27 by the yellow multiplier band (10,000) indicates 270,000 ohm (270K) resistor. The silver band denotes the 10% tolerance. Use this process to choose the correct resistor specified in the manual.

Use the following procedure to install resistors on the boards. Make sure the colored bands on each resistor match the colors called for in the list of Resistor Values and Color Codes given in the assembly instructions.

1. Using needle-nose pliers, bend the leads of the resistor at right angles to match their respective holes on the PC board.

2. Install the resistor into the correct holes on the silk-screened side of the PC board.
3. Holding the resistor in place with one hand, turn the board over and bend the two leads slightly outward.
4. Solder the leads to the foil patter on the back side of the board; then clip off any excess lead lengths.

RESISTOR COLOR CODES		
COLOR	BANDS 1&2	3rd BAND (Multiplier)
Black	0	1
Brown	1	10^2
Red	2	10^3
Orange	3	10^4
Yellow	4	10^5
Green	5	10^6
Blue	6	10^7
Violet	7	10^8
Gray	8	10^9
White	9	

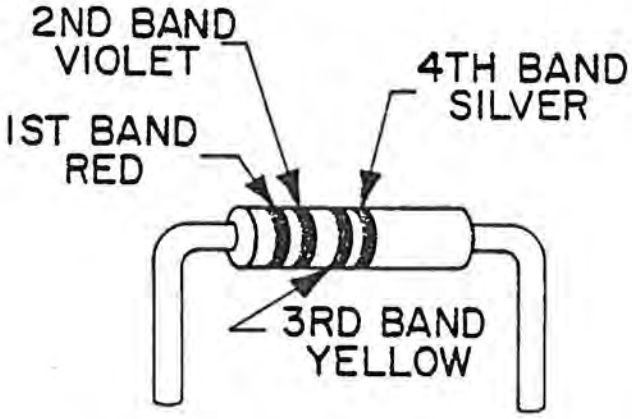


Figure 5-4 Resistor Color Code Example

B. Capacitor Installation Instructions

1) <u>Polarity must be noted on electrolytic capacitors before they are installed.</u>

The electrolytic capacitors contained in your kit may have one of three types of polarity markings. (Figure 5-5 illustrates two types of capacitors.) To determine the correct orientation, look for the following:

One type has plus (+) signs on the positive end; another has a band or a groove around the positive side in addition to the plus signs. The third type has an arrow on it; in the tip of the arrow there is a negative (-) sign. The capacitor must be oriented so the arrow points to the negative side.

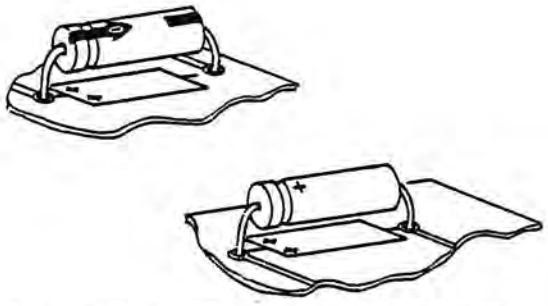


Figure 5-5 Electrolytic Capacitors

Install the electrolytic capacitors using the following procedure. Make sure you have the correct capacitor value before installing each one.

a. Bend the two leads of the capacitor at right angles to conform to their respective holes on the board. Insert the capacitor into the holes on the silk-screened side of the board, aligning the positive side with the "+" signs printed on the board.

b. Holding the capacitor in place, turn the board over and bend the two leads slightly outward. Solder the leads to the foil (bottom) side of the board and clip off any excess lead lengths.

2) Epoxy Dipped Tantalum, Epoxy Dipped Ceramic, and Ceramic Disk Capacitors

Polarity must be noted on epoxy dipped tantalum capacitors before they are installed.

There are two types of epoxy dipped tantalum capacitors contained in the kit. The first type is blue on the positive side. The second type is marked with "+" signs on the positive side. Both types of epoxy dipped tantalum capacitors are shown in Figures 5-6 and 5-7.

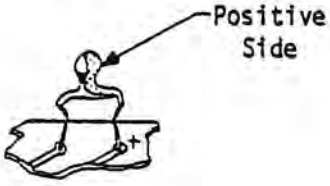


Figure 5-6
Epoxy Dipped Tantalum Capacitor

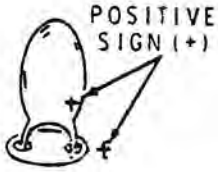


Figure 5-7
Epoxy Dipped Tantalum Capacitor

The epoxy dipped ceramic capacitors and the ceramic disk capacitors are non-polarized.

These two types of capacitors are shown in Figure 5-8.

Figure 5-8 Epoxy Dipped Ceramic Capacitor

Install these 4 types of capacitors using the following procedure. Make sure you have the correct capacitor value before installing each one.

a. Bend the two capacitor leads to conform to their respective holes on the board.
b. Insert the capacitor into the correct holes from the silk-screened side of the board. Holding the capacitor in place, turn the board over and bend the two leads slightly outward.
c. Solder the two leads to the foil (bottom) side of the board and clip off any excess lead lengths.

C. Diode Installation Instruction

Use the following procedure to install diodes onto the board. Refer to the list of Diode Part Numbers included for each board to make sure you install the correct diode each time.

> NOTE
>
> DIODES ARE MARKED WITH A BAND ON ONE END INDICATING THE CATHODE END. EACH DIODE MUST BE INSTALLED SO THAT THE END WITH THE BAND IS ORIENTED TOWARDS THE BAND PRINTED ON THE PC BOARD. FAILURE TO ORIENT THE DIODES CORRECTLY MAY RESULT IN PERMANENT DAMAGE TO YOUR UNIT.

NOTE: MITS part numbers are stamped on most diodes.

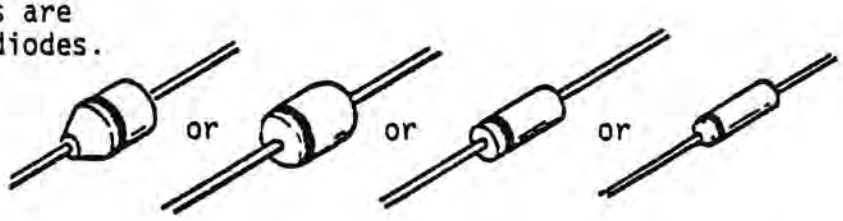


Figure 5-9 Diodes

1. Bend the leads of the diode at right angles to match their respective holes on the board.
2. Insert the diode into the correct holes on the silkscreen, making sure the cathode end is properly oriented. Turn the board over and bend the leads slightly outward.
3. Solder the two leads to the foil pattern on the back side of the board; then clip off any excess lead lengths.

D. Transistor Installation Instructions

To install transistors, use the following instructions.

> NOTE
>
> ALWAYS CHECK THE PART NUMBER OF EACH TRANSISTOR BEFORE YOU INSTALL IT. (SEE LISTING OF TRANSISTOR PART NUMBERS FOR EACH BOARD.) SOME TRANSISTORS LOOK IDENTICAL BUT DIFFER IN ELECTRICAL CHARACTERISTICS. IF YOU HAVE RECEIVED SUBSTITUTE PART NUMBERS FOR THE TRANSISTORS IN YOUR KIT, CHECK THE TRANSISTOR IDENTIFICATION CHART WHICH FOLLOWS THESE INSTRUCTIONS TO BE SURE YOU MAKE THE CORRECT SUBSTITUTIONS.

1. After the correct transistor has been selected and the leads have been properly oriented, insert the transistor into the holes on the silk-screened side of the board.
2. Holding the transistor in place, turn the board over and bend the three leads slightly outward.
3. Solder the leads to the foil pattern on the back side of the board; then clip off any excess lead lengths.

NOTE

ALWAYS MAKE SURE THE TRANSISTOR IS ORIENTED SO THAT THE EMITTER LEAD IS INSTALLED IN THE HOLE ON THE PC BOARD LABELED WITH AN "E." TO DETERMINE WHICH LEAD IS THE EMITTER LEAD, REFER TO THE TRANSISTOR IDENTIFICATION CHART.

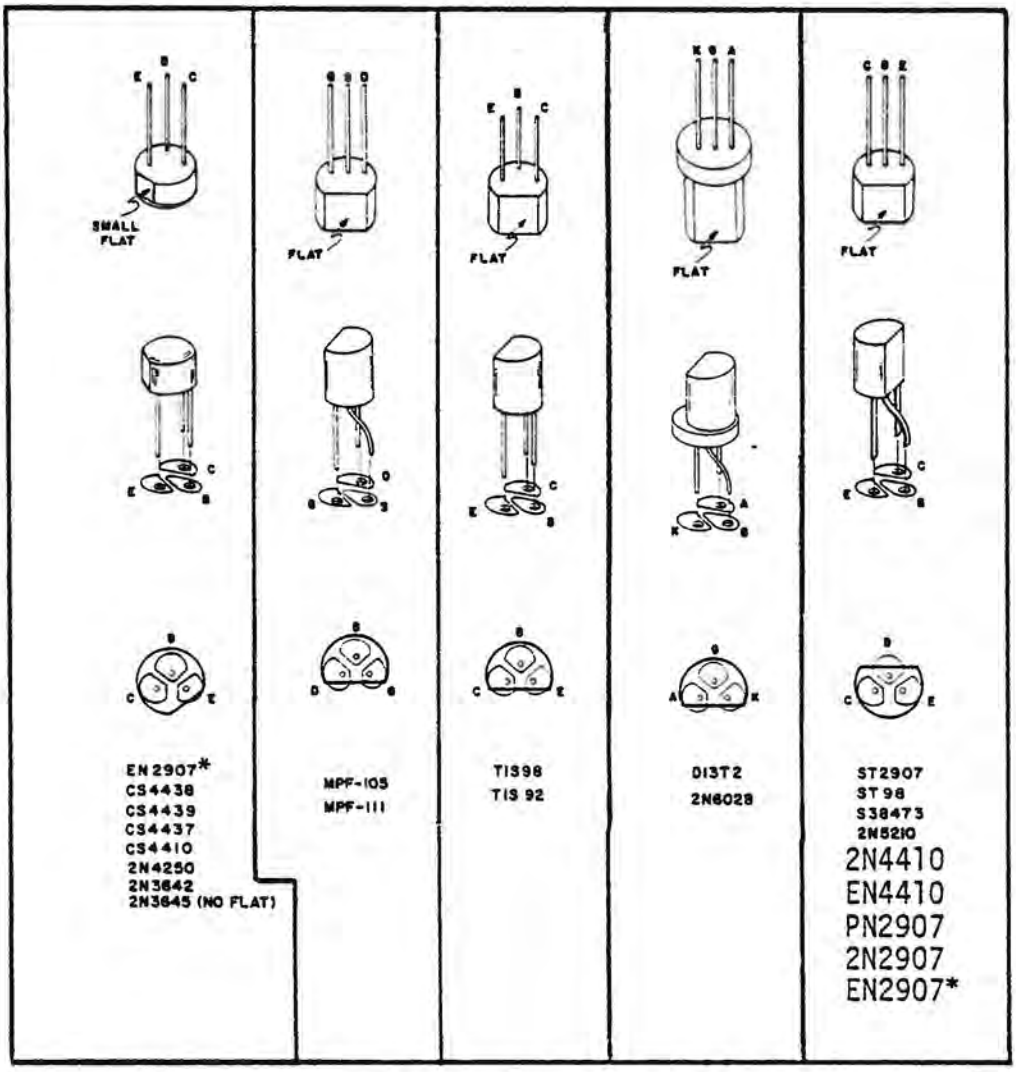


Figure 5-10 Transistor Identification Chart

E. IC Installation Instructions

All ICs must be oriented so that the notched end is toward the end with the arrowhead printed on the PC board. Pin 1 of the IC should correspond with the pad marked with the arrowhead. If the IC does not have a notch on one end, refer to the IC Identification Chart to identify Pin 1.

To prepare ICs for installation:

All ICs are damaged easily and should be handled carefully--especially static-sensitive MOS ICs. Always try to hold the IC by the ends, touching the pins as little as possible. When you remove the IC from its holder, CAREFULLY straighten any bent pins using needle-nose pliers. All pins should be evenly spaced and should be aligned in a straight line, perpendicular to the body of the IC itself.

1) Installing ICs without sockets

a. Orient the IC so that Pin 1 conicides with the arrowhead on the PC board.

b. Align the pins on one side of the IC so that just the tips are inserted into the proper holes on the board.

c. Lower the other side of the IC into place. If the pins do not go into their holes right away, rock the IC back, exerting a little inward pressure, and try again. Be patient. The tip of a small screwdriver may be used to help guide the pins into place. When the tips of all the pins have been started into their holes, push the IC into the board the rest of the way. Tape the IC to the board with a piece of masking tape.

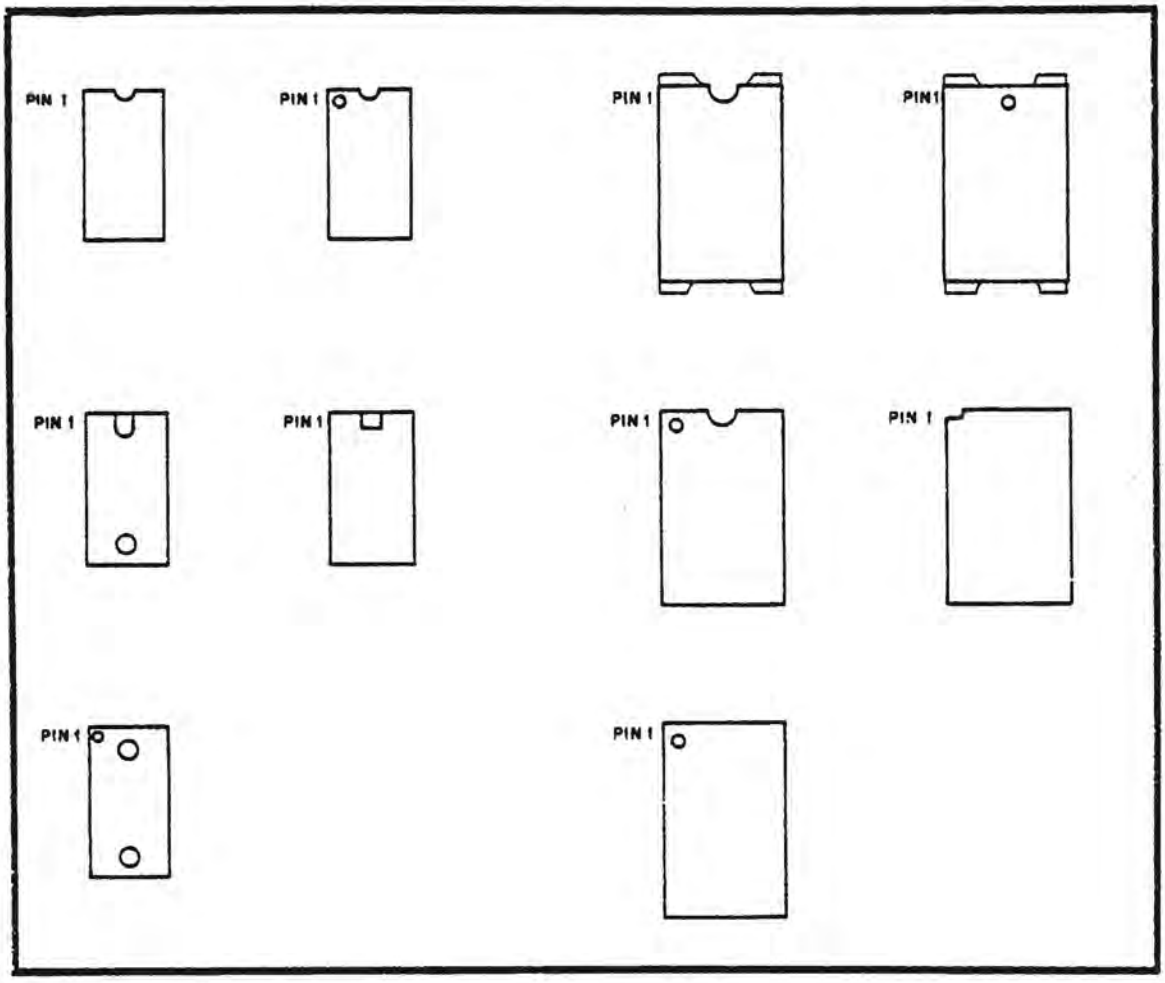
d. Turn the board over and solder each pin to the foil pattern on the back side of the board. Be sure to solder each pin and be careful not to leave any solder bridges. Remove the masking tape.

WARNING
MAKE SURE NONE OF THE PINS HAVE BEEN PUSHED UNDERNEATH THE IC DURING INSERTION.

2) Installing ICs with sockets

Referring to the drawing below, set the IC socket into the designated holes on the board and secure it with a piece of masking tape.

NOTE
THE IC SOCKET HAS A NOTCH IN ONE CORNER. THIS IS THE PIN 1 MARKING. ORIENT THE SOCKET WITH THE NOTCH CORRESPONDING TO THE PIN 1 ARROW SILKSCREENED ON THE BOARD.



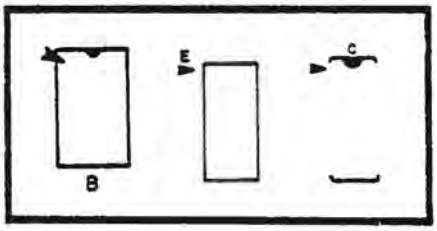


Figure 5-11 IC Installation Chart

INTEGRATED CIRCUITS (ICs) CAN COME WITH ANY ONE OF, OR A COMBINATION OF, SEVERAL DIFFERENT MARKINGS. THESE MARKINGS ARE VERY IMPORTANT IN DETERMINING THE CORRECT ORIENTATION FOR THE ICs WHEN THEY ARE PLACED ON THE PRINTED CIRCUIT BOARDS. REFER TO FIGURE 5-11 TO LOCATE PIN 1 OF IC'S THEN USE THIS INFORMATION IN CONJUNCTION WITH THE INFORMATION BELOW TO PROPERLY ORIENT EACH IC FOR INSTALLATION.

WARNING: INCORRECTLY ORIENTED IC's MAY CAUSE PERMANENT DAMAGE!

THE DRAWING ON THE LEFT INDICATES VARIOUS METHODS USED TO SHOW THE POSITION OF IC's ON THE PRINTED CIRCUIT BOARDS. THESE ARE SILK-SCREENED DIRECTLY ON THE BOARD. THE ARROWHEAD INDICATES THE POSITION FOR PIN 1 WHEN THE IC IS INSTALLED.

a. Turn the board over and solder each pin to the foil pattern on the back side of the board. Be sure to solder each pin and be careful not to leave any solder bridges. Remove the masking tape.

b. Orient the IC over the socket so that Pin 1 coincides with the arrowhead on the PC board.

c. Align the pins on one side of the socket so that just the tips are inserted into the holes.

d. Lower the other side of the IC into place. If the pins do not go into their holes right away, rock the IC back, exerting a little inward pressure, and try again. Be patient. When the tips of all the pins have been started into their holes, push the IC into the socket the rest of the way.

F. MOS IC Special Handling Precautions

This kit contains several MOS IC (Metal Oxide Semiconductor) integrated circuits. These IC's are very sensitive to static electricity and transient voltages. In order to prevent damaging these components, read over the following precautions and adhere to them as closely as possible. FAILURE TO DO SO MAY RESULT IN PERMANENT DAMAGE TO THE IC.

1. All equipment (soldering iron, tools, solder, etc.) should be at the same potential as the PC board, the assembler, the work surface and the IC itself along with its container. This can be accomplished by continuous physical contact with the work surface, the components and everything else involved in the operation.
2. When handling the IC, develop the habit of first touching the conductive container in which it is stored before touching the IC itself.
3. If the IC has to be moved from one container to another, touch both containers before doing so.
4. Do not wear clothing which will build up static charges. Wear clothing made of cotton, rather than wool or synthetic fibers.
5. Always touch the PC board before touching the IC to the board. Try to maintain this contact as much as possible while installing the IC.
6. Handle the IC by the edges. Avoid touching the pins themselves as much as possible.
7. Dry air moving over plastic can result in the development of a significant static charge. Avoid placing the IC near any such area or object.
8. In general, never touch anything to the IC that you have not touched first while touching both it and the IC itself.

5-3. CPU Board Assembly

The CPU (Central Processing Unit) Board is the control center of the Altair 8800b Turnkey. It controls the flow of data, and performs computations. In addition, it retrieves and executes program instructions stored in memory and interprets stored program information. The CPU controls data input and output from all I/O ports.

The following instructions describe the assembly of the CPU Board. Be sure to check off the appropriate box after installing each component and heed all the warnings and/or hints.

A. CPU Diode Installation

The CPU board has two diodes, D1 and D2 (Bag 4). Figure 5-12 defines the location. These diodes have different values; be sure to insert them in the correct location. Install the diodes according to Diode Installation Instructions given in section 5-2E. Be sure to orient the cathode (banded) end of the diode with the banded marking on the board.

Diodes

( )	D1	IN473Ø
( )	D2	IN4733

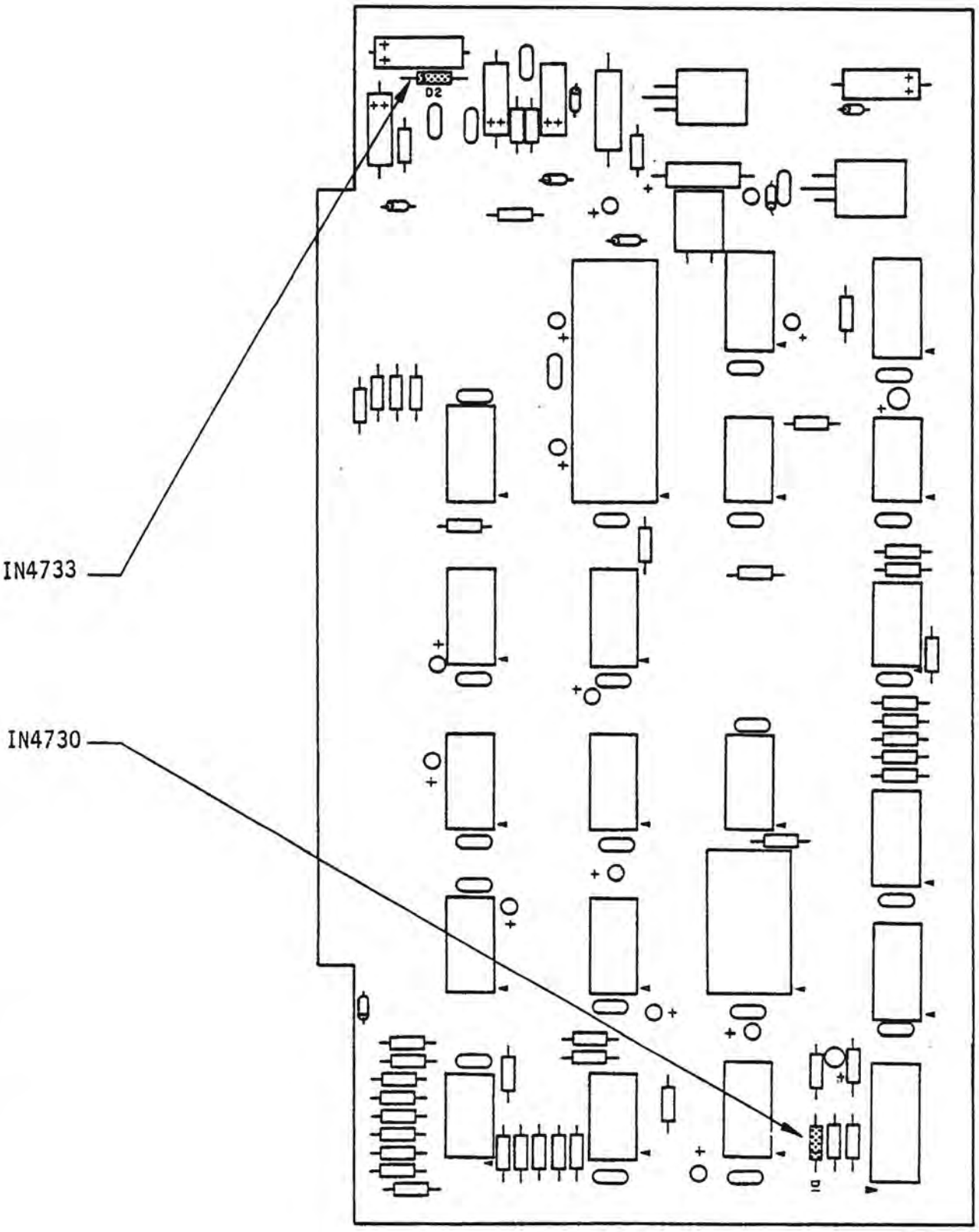


Figure 5-12 Diode Installation

B. CPU Resistor Installation

There are 46 resistors to be installed on the CPU board (Bag 4). Install the resistors as per instructions in section 5-2A. Check each resistor or group of resistors carefully before installing them to make sure the correct value resistor is installed in the correct location. It is easiest to install a small group of resistors (those with the same value) simultaneously. Turn the board over and solder the leads in place. Clip off resistor leads immediately. A small forest of unclipped resistor leeds is difficult to work around and may result in poor soldering results.

NOTE

SAVE CLIPPED RESISTOR LEADS. THESE WILL BE USED FOR FERRITE BEAD INSTALLATION (SECTION 5-3E).

Resistors	Value
( ) R3-7 ( ) R11 ( ) R13 ( ) R14 ( ) R19 ( ) R2Ø ( ) R24 ( ) R25 ( ) R28-33 ( ) R38 ( ) R40-43 ( ) R5Ø	2.2K ohm 1/2w or 1/4w red, red, red
( ) R1 ( ) R2 ( ) R8 ( ) R26 ( ) R27 ( ) R37 ( ) R39 ( ) R44-49	3.3K ohm 1/2w or 1/4w orange, orange, red

( ) R9	15K ohm 1/2w or 1/4w brown, green, orange
( ) R15	1ØØ ohm 1/2 or 1/4w brown, black, brown
( ) R16	1K ohm 1/2w or 1/4w brown, black, red
( ) R34	62Ø ohm 1/2w blue, red, brown
( ) R1Ø	33Ø ohm 1/2w or 1/4w orange, orange, brown
( ) R21 ( ) R23	47Ø ohm 1/2w or 1/4w yellow, violet, brown
( ) R17	1ØK ohm 1/2w or 1/4w brown, black, orange
( ) R22	1Ø ohm 2w brown, black, black

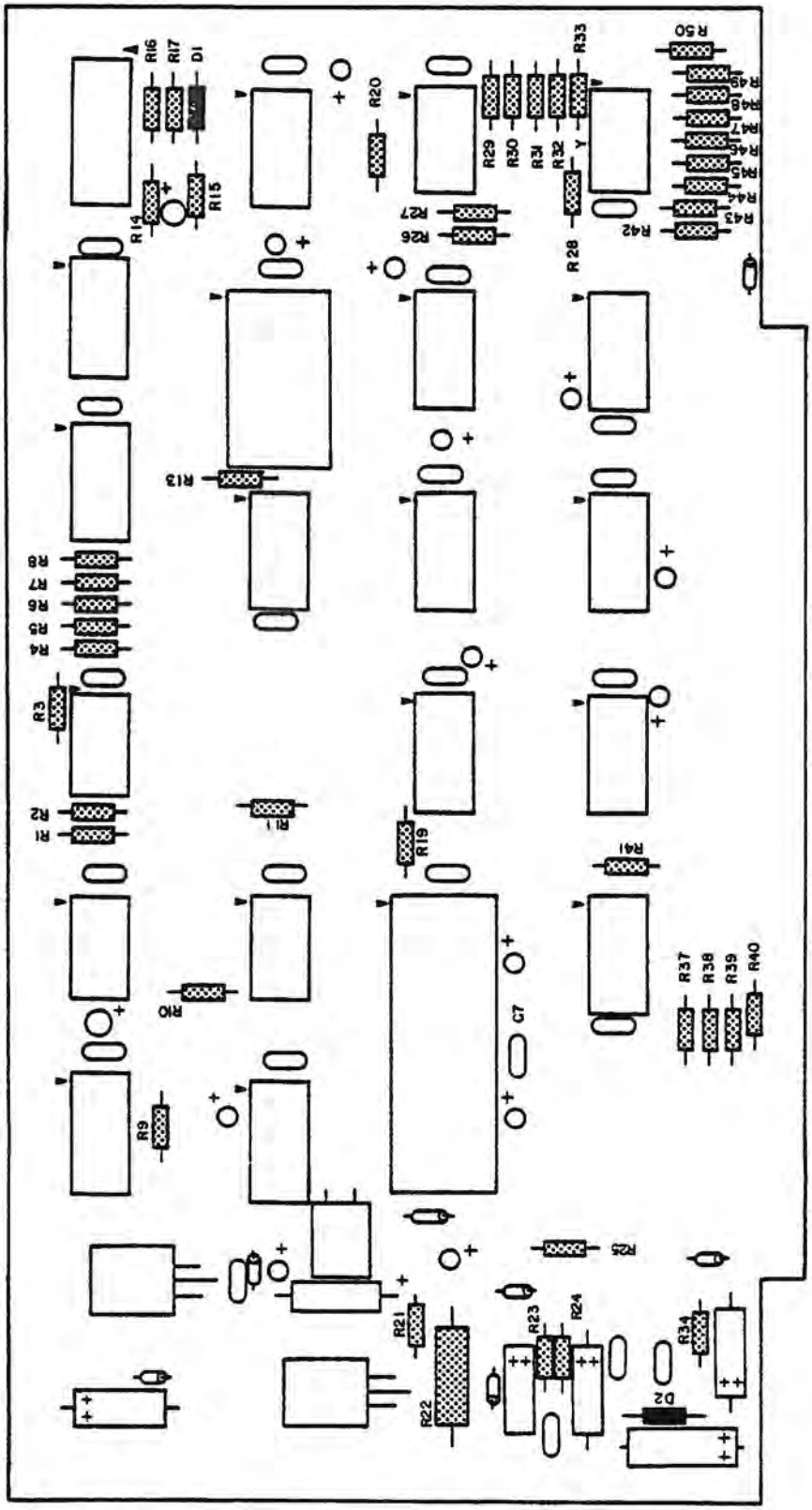


Figure 5-13 Resistor Installation

C. CPU Integrated Circuit Installation

There are 20 integrated circuits (Bags 1 and 2) which will be installed in sockets (Bag 8) on the CPU board. Howevever, only 17 will be installed at this time. IC's A, K, and M are heat and static sensitive MOS IC's and will be inserted in the final installation procedure.

Install the smallest sockets first proceeding to the 24 pin socket. Follow the instructions in section 5-2 for IC socket installation.

The following table lists all ICs, part designations and acceptable substitutions.

IC	Designation
( ) D,E	8216
( ) F	8224
( ) N	8224
( ) P	74367
( ) R	74367
( ) U	74367
( ) V	74367
( ) W	74367
( ) X	74367
( ) S	74LS14 or 74LSØ4
( ) Y	74LS14 or 74LSØ4
( ) C	74LS14 or 74LS2Ø
( ) B	74LSØ4
( ) G	74LSØ4
( ) L	8T98 or 8098 or 74368
( ) J	8T98 or 8098 or 74368

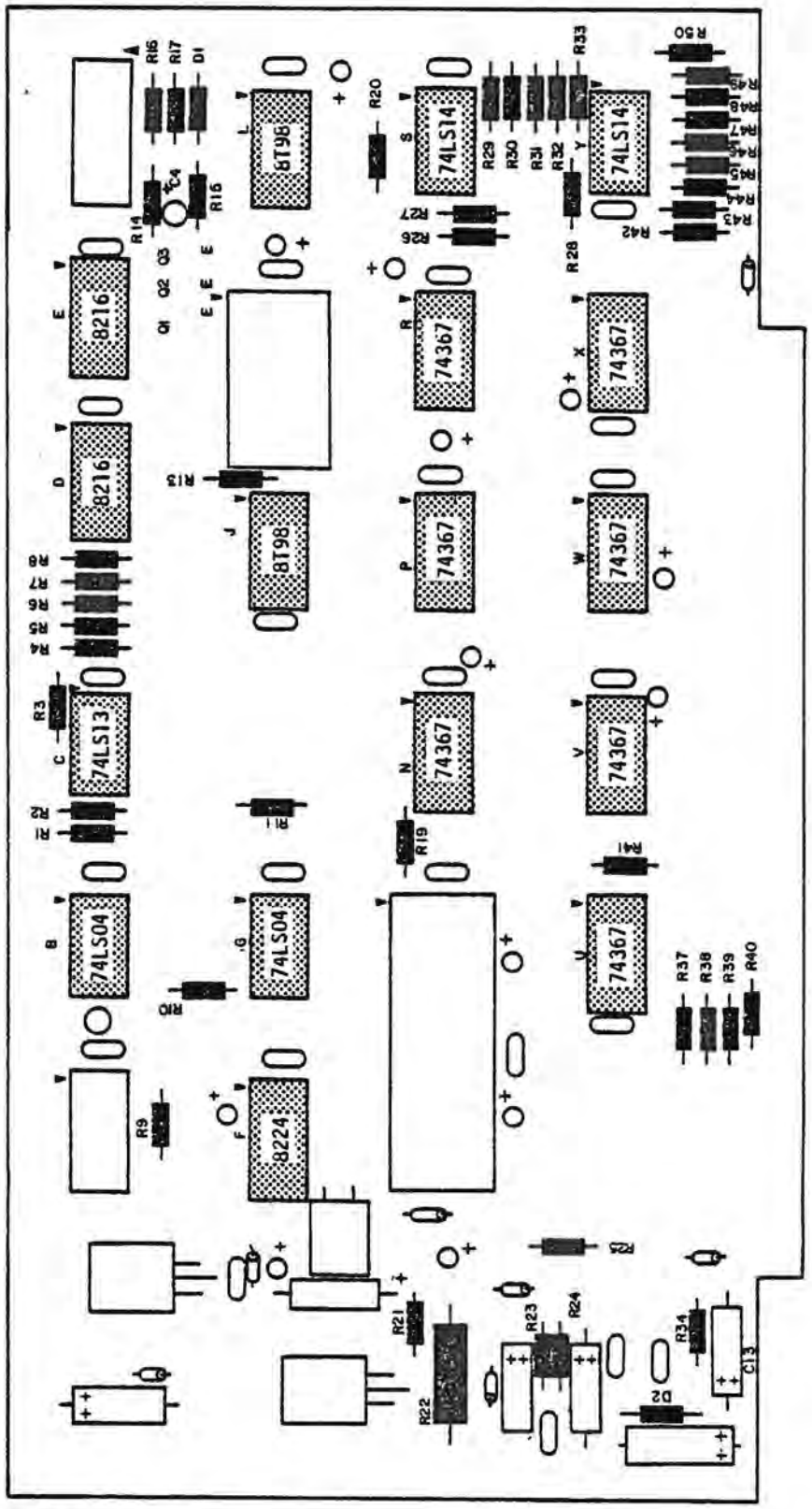


Figure 5-14 Integrated Circuit Installation

D. Voltage Regulator/Heat Sink Installation

The following instructions describe the installation of the heat sink and voltage regulators, VR 1 and 2 (Bag 7). Check the part number of the voltage regulator to be sure each is installed in its respective place on the board. They are not interchangeable.

Use heat sink grease between the heat sink and the regulator. Apply the grease to all metal surfaces which are in contact. Figure 5-15 defines the location of the voltage regulatory heat sink.

1. Set the regulator in place on the silk-screened side of the board. Make a small mark on each lead at the point directly above the hole in the board.
2. Use needle-nose pliers to carefully bend each lead at a right angle to fit into the proper hole on the board.
3. As shown in Figure 5-16, install the regulator on the heat sink and the heat sink on the silk-screened side of the board. Some kits contain a mica insulating strip (not illustrated). Place this insulator between the voltage regulator and heat sink.
   Secure the entire unit to the board with a screw (#6-32 x 3/8"), nut (#6-32) and lockwasher (#6) included in the kit for this board. If the heat sink included in your kit has 2 screw holes, be sure to orient the voltage regulator over the hole which will result in the entire voltage regulator being located on the heat sink. The voltage regulator should not protrude from the heat sink.
4. Solder the voltage regulator leads to the bottom of the board, taking care not to leave any solder bridges.
5. Clip off excess lead lengths.

Voltage Regulator	Part Number
( ) VR1	78Ø5
( ) VR2	7812

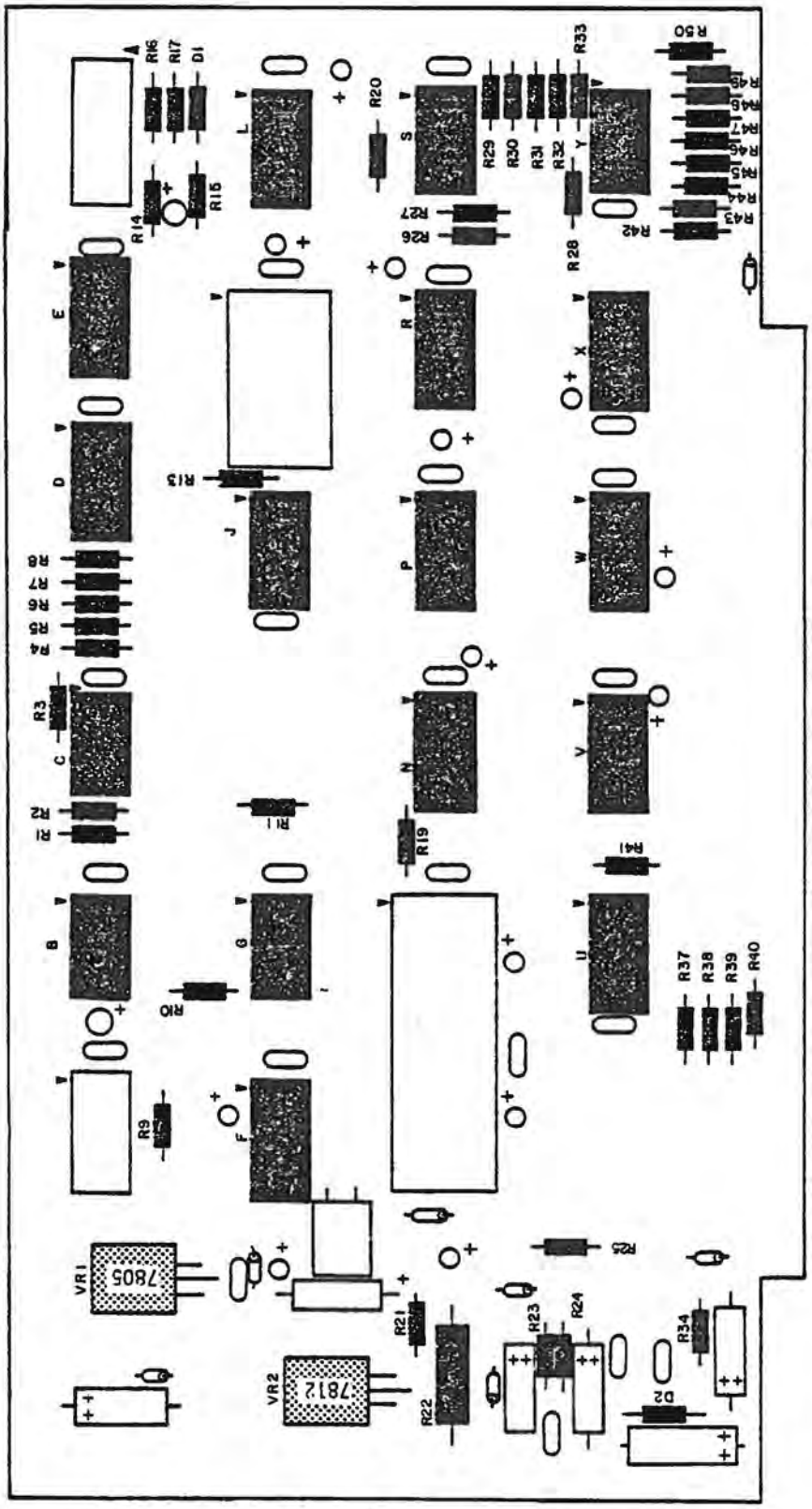


Figure 5-15 Voltage Regulator/Heat Sink Installation

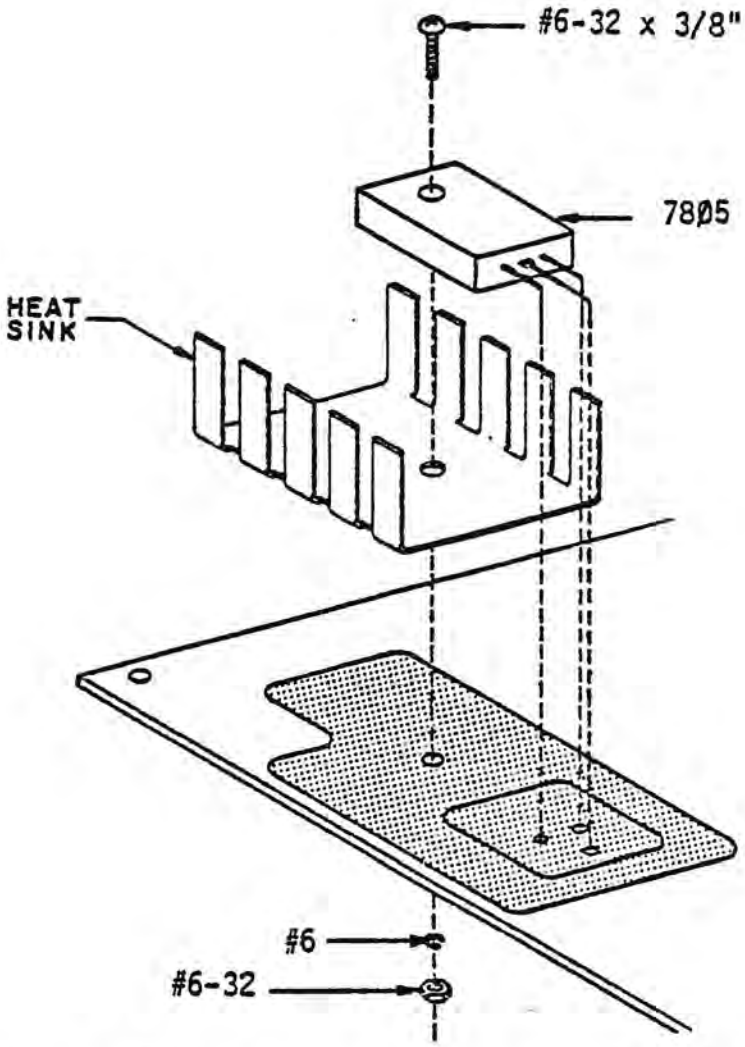


Figure 5-16 Voltage Regulator/Heat Sink Orientation

E. Ferrite Bead Installation

Use the following instructions to install the 7 ferrite beads L1-L7 (Bag 7). The ferrite beads are installed using clipped resistor leads which were saved from the installation of the resistors. Take care that small pieces of leftover solder do not become mixed with excess resistor leads.

a. Cut seven, 1-inch lead lengths.

b. Insert the lead through the bead, and bend the ends so they conform to the designated holes of the CPU board.

c. Insert the leads into the board and solder into place.

d. Clip excess lead lengths.

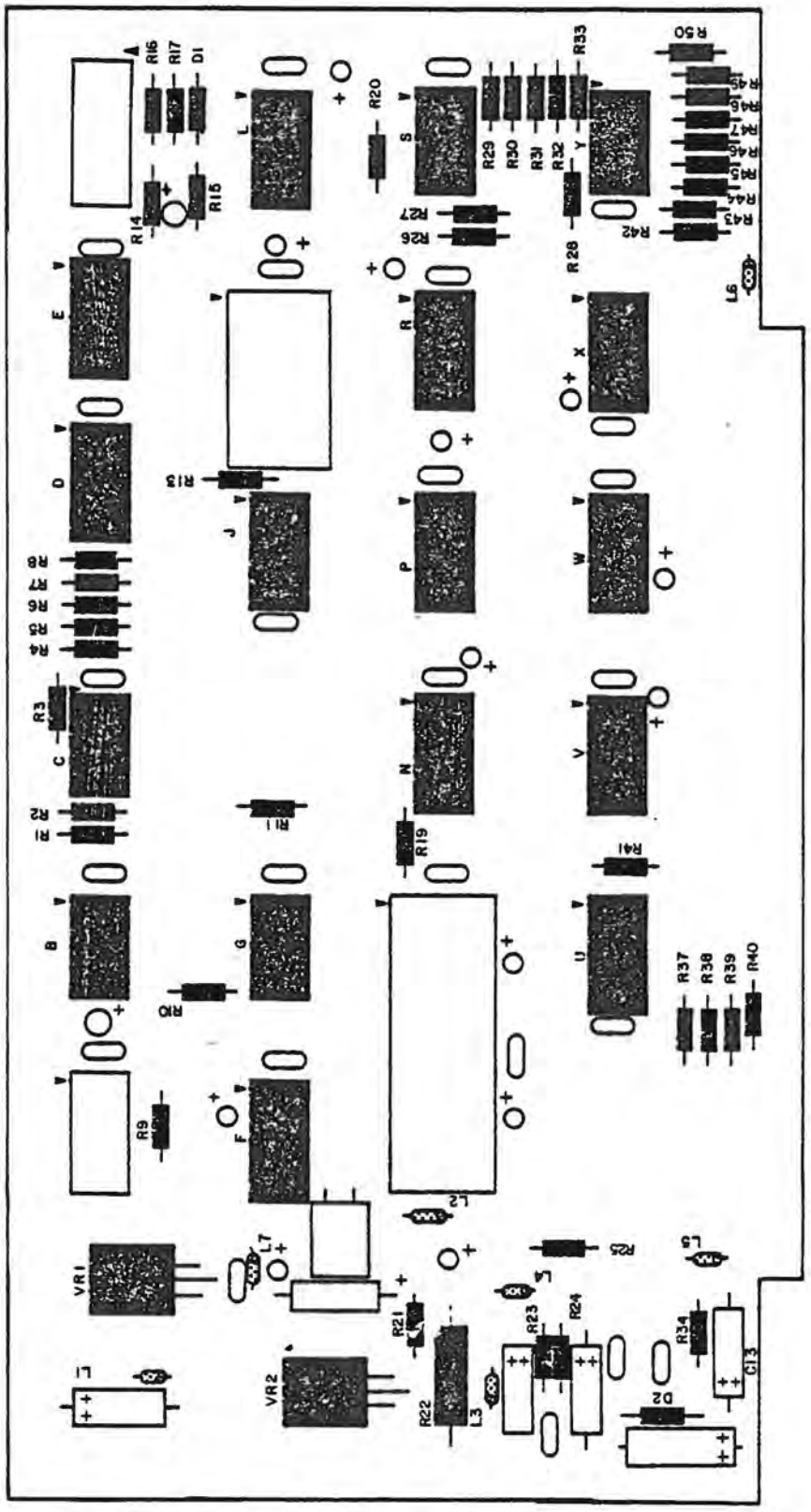


Figure 5-17 Ferrite Bead Installation

F. Suppressor Capacitor Installation

There are two types of suppressor capacitors to be installed on the CPU board.

1) Epoxy dipped tantalum capacitors (Bag 3)

These capacitors are marked for polarity and must be installed to correlate with the polarity markings on the silk-screened side of the board.

a) According to supply variations, some capacitors are marked with a plus (+) sign to designate polarity. The plus sign may be quite small, so it might be desirable to use a hand lens to distinguish polarity.

b) Other capacitors may be blue or red on one side to denote positive polarity.

c) When inserting dipped tantalum capacitors, leave a small amount of lead above the board. Since the location of these capacitors is not designated by a letter/numeral code, it is easiest to mark off each suppressor capacitor on Figure 5-18 as it is installed.

NOTE
DO NOT GO ACROSS THE BOARD INSERTING TANTALUM DIPPED SUPPRESSOR CAPACITORS IN ANY ROUND MARKING WITH A POLARITY SYMBOL. TWO MARKINGS (C2 AND C4) ARE FOR CAPACITORS OF A DIFFERENT VALUE. THESE WILL BE INSERTED IN SUBSECTION G.

2) Ceramic Disk Capacitors (Bag 6)

Capacitors C8 and C12 are not used for noise suppression. However, installing these ceramic disk capacitors along with the other ceramic disk capacitors will reduce the number of construction steps.

These capacitors have no polarity markings, and need no polarity orientation.

It is easiest to install and then solder one row of capacitors at a time. Ceramic disk capacitor leads are often quite long and installing all of them at once can be an awkward task.

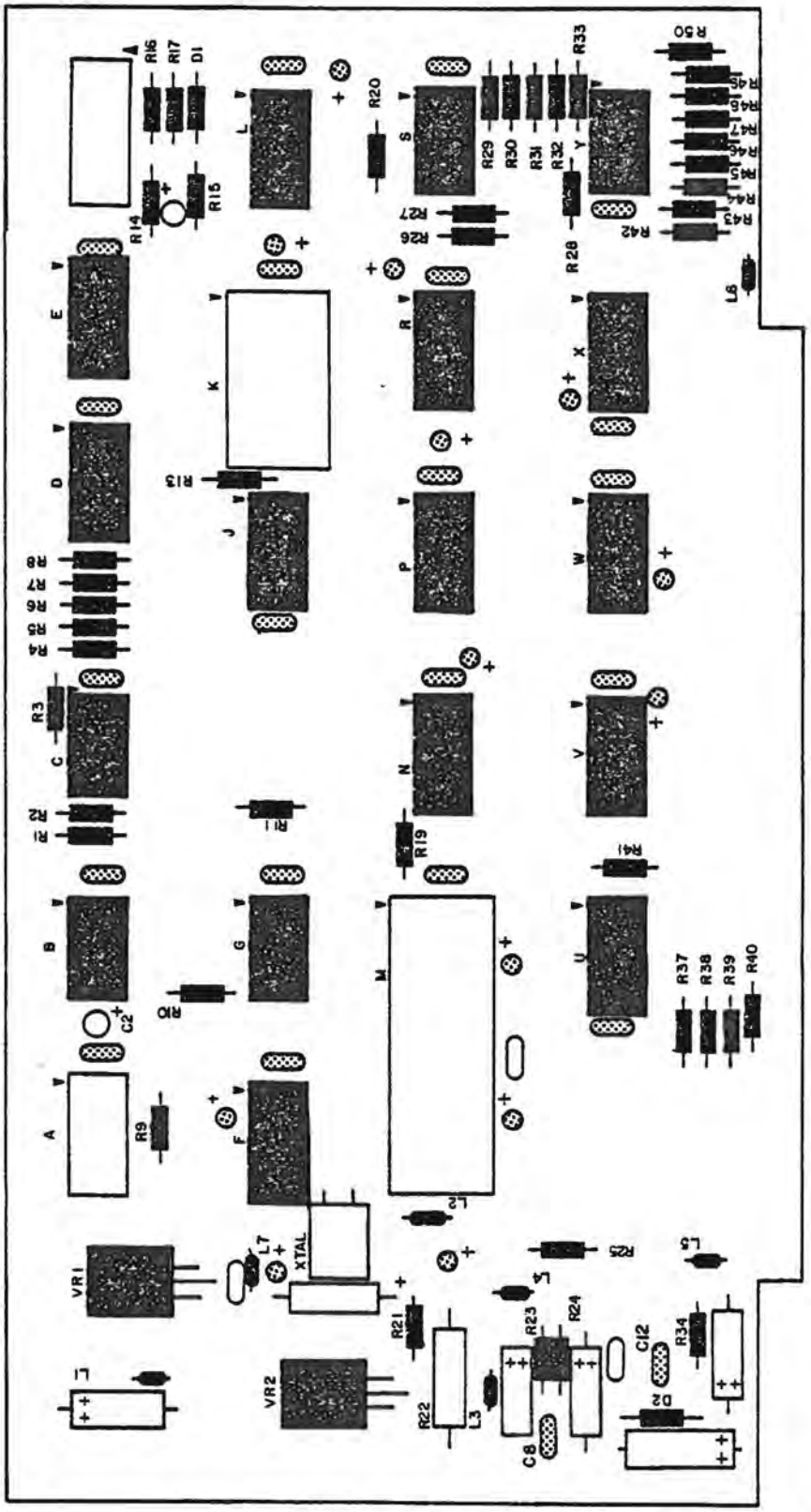


Figure 5-18 Suppressor & Ceramic Disk Capacitor Installation

Suppressor Capacitor	Values
( ) 13 dipped tantalum	1uf, 35V
( ) 12 ceramic disk	1uf, 12V

Ceramic Disk Capacitors	Values
( ) C8 ( ) C12	.1uf, 12v-16v, ceramic disk

G. Capacitor Installation

There are 2 dipped tantalum capacitors, 6 electrolytic capacitors and 3 dipped ceramic capacitors (square in shape) to be installed on the CPU Board. Install each capacitor according to the instructions in section 5-2.

Be sure to install the dipped tantalum and electrolytic capacitors with the positive lead in its corresponding positive hole (+) on the board.

The remaining 2 dipped tantalum capacitors, while similar in appearance, are of different values. The value is written on the capacitor in very small numbers. Be sure to insert the capacitor in the proper position and orient the leads for proper polarity.

> NOTE
>
> When inserting Electrolytic Capacitor C13, be sure to insert leads in the holes outside the square silk-screened on the board.

The leads for Dipped Ceramic capacitor C3 should be in the leftmost of the two adjoining holes. Be especially careful when inserting this component that the Dipped ceramic capacitor leads and the voltage regulator leads do not touch or that a solder bridge does not join the two components.

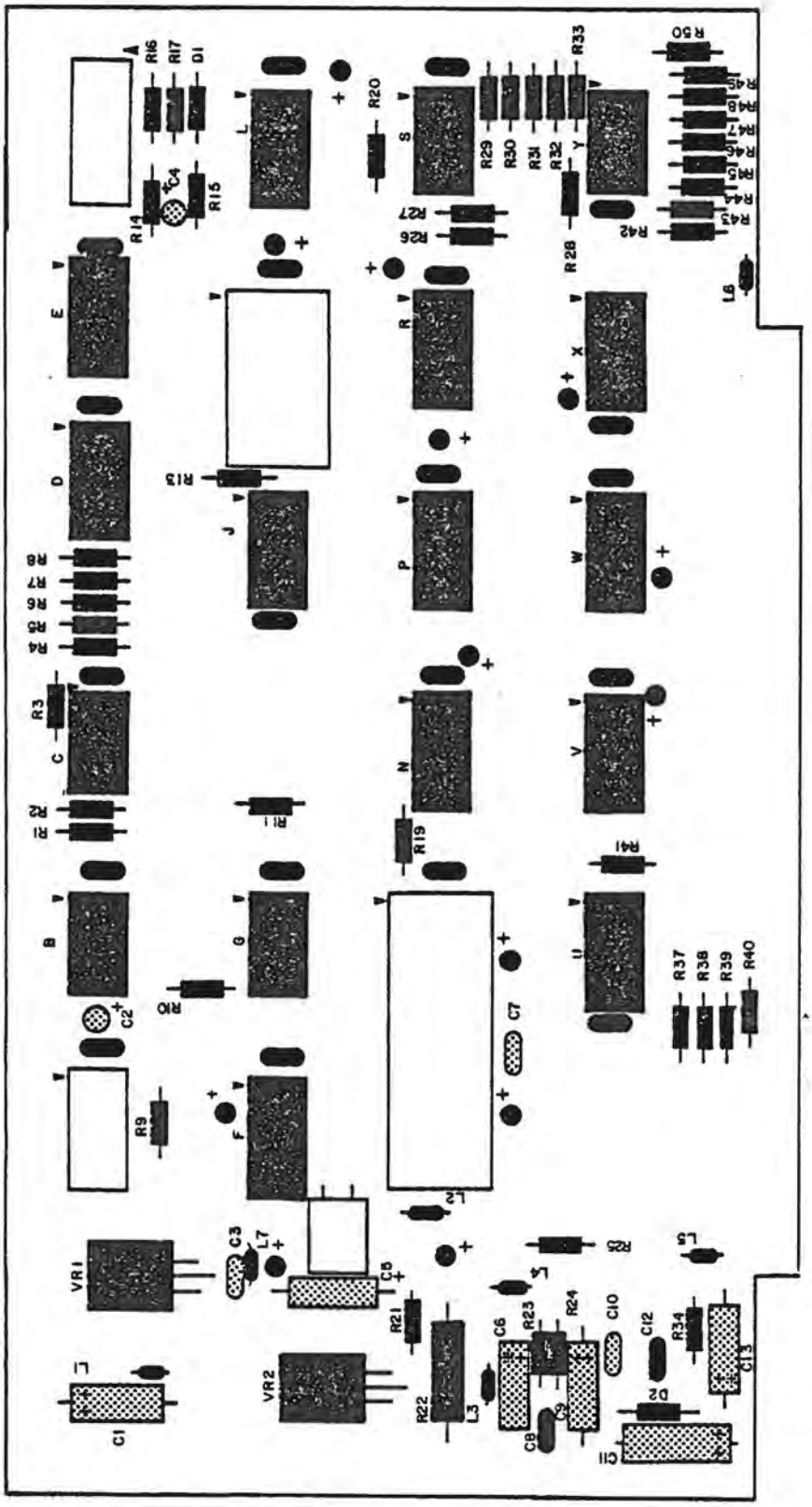


Figure 5-19 Capacitor Installation

Capacitor	Value
( ) C1	33 uf, 16V
( ) C5, C6 ( ) C11	Electrolytic
( ) C2	22 uf, 16v, dipped tantalum
( ) C3 ( ) C7 ( ) C10	.1uf, 50v dipped ceramic
( ) C4	10 uf, 16v dipped tantalum
( ) C9 ( ) C13	10 uf, 25v electrolytic

H. Transistor Installation

Instructions for transistor installation can be found in section 5-2. Transistor Q1-Q3 (Bag 4) should be carefully installed following those instructions.

Be sure the emitter lead is affixed in the hole marked E. The transistor identification chart in section 5-2 identifies the emitter lead.

Transistor	Part Number
( ) Q1	2N4410 or
( ) Q2 ( ) Q3	CS4410

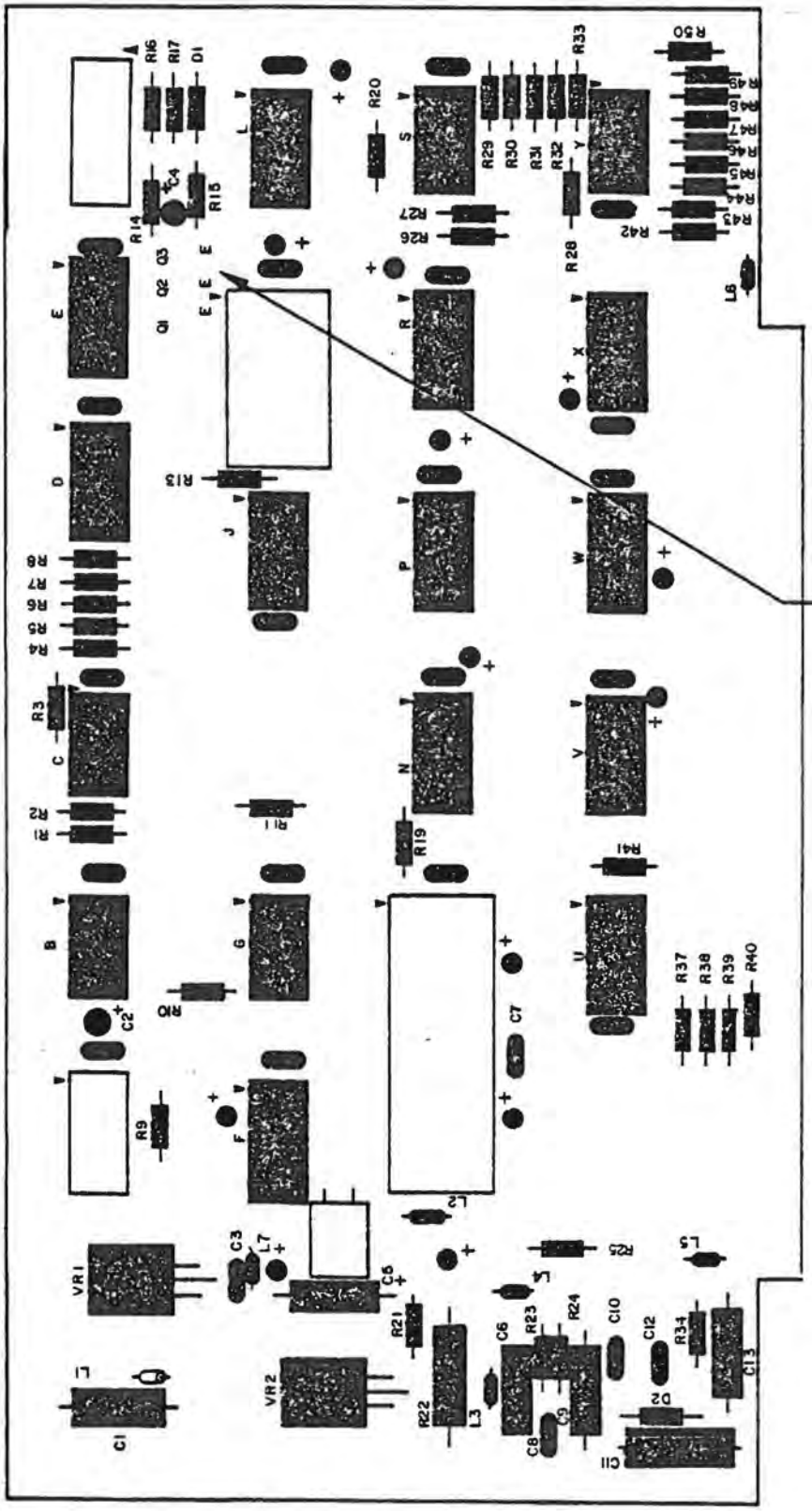


Figure 5-20 Transistor Installation

I. Male Connector Installation

Some kits may arrive with the 10-pin male connector previously installed on the CPU board. If it has been installed, ignore the installation instruction.

The following instructions define the installation of the male connector, P 3 Bag 7.

1. Orient the connector as shown in Figure 5-21. The bent pins should point toward the top of the board.
2. Insert the short pins into the 10 designated holes on the silk-screened side of the board.
3. Solder each pin to the back of the board. Be careful not to leave any solder bridges.
4. The arrow on the illustration below points to Pin #1. After the male connector has been installed on the board, clip off Pin #2 of the connector. This is done for purposes of keying.

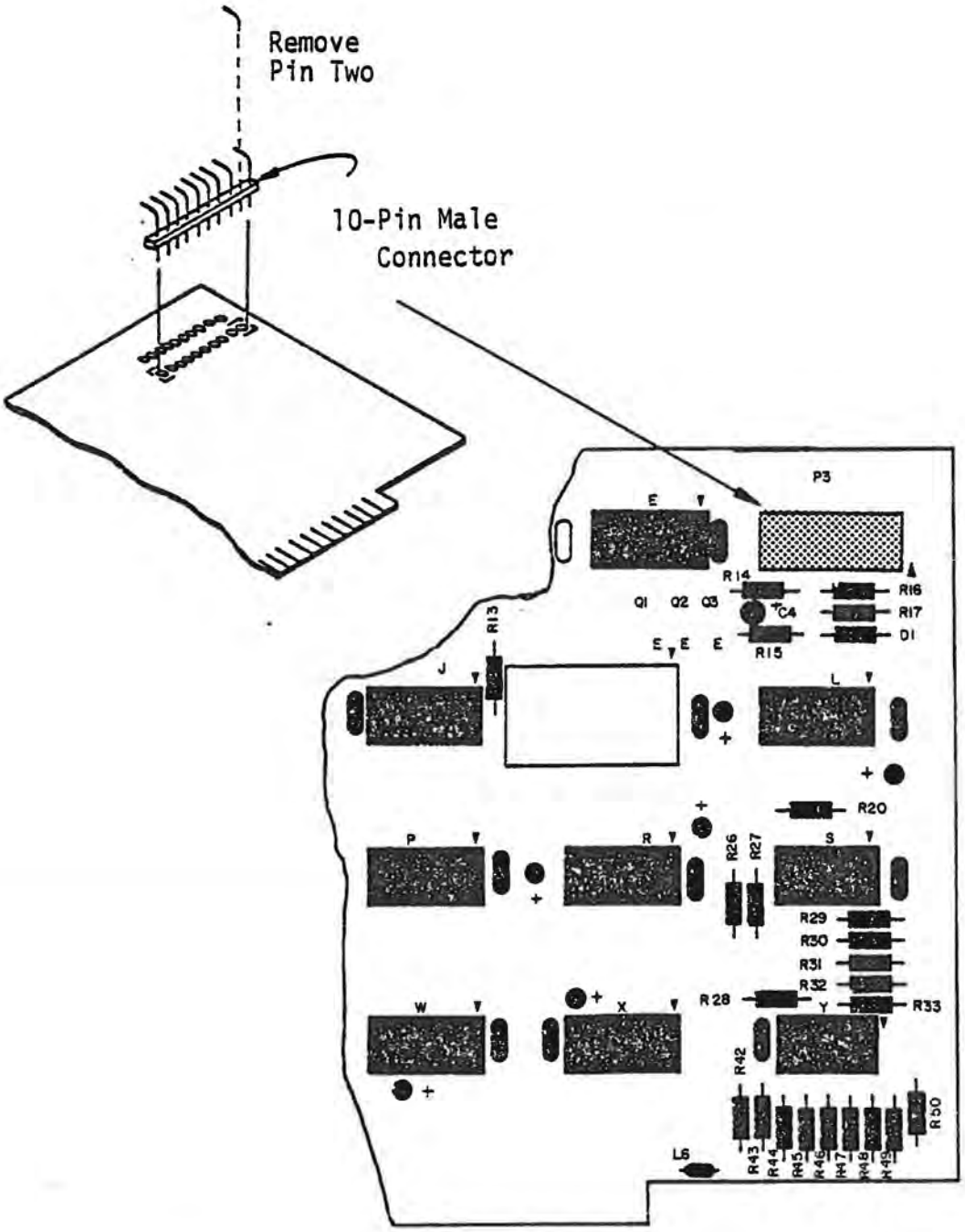


Figure 5-21 Male Connector Installation

J. Crystal Installation Instruction

There is one 18.00000MHz crystal (labeled XTAL) (Bag 7) which should be installed according to the following instructions.

NOTE
Do not allow crystal case to come in contact with any traces on the CPU Board.

1. As illustrated in Figure 5-22, set the crystal in place on the silk-screened side of the CPU board, with identification markings facing up within the square labeled XTAL. Align both leads with their respective holes on the board.

NOTE
Be sure to insert crystal leads in the holes closest to the square marked XTAL.

2. Bend each lead with needle-nose pliers to conform to its respective hole on the board. The crystal must rest flat on the board.

3. Carefully solder both leads to the bottom of the board, and clip excess lead length.

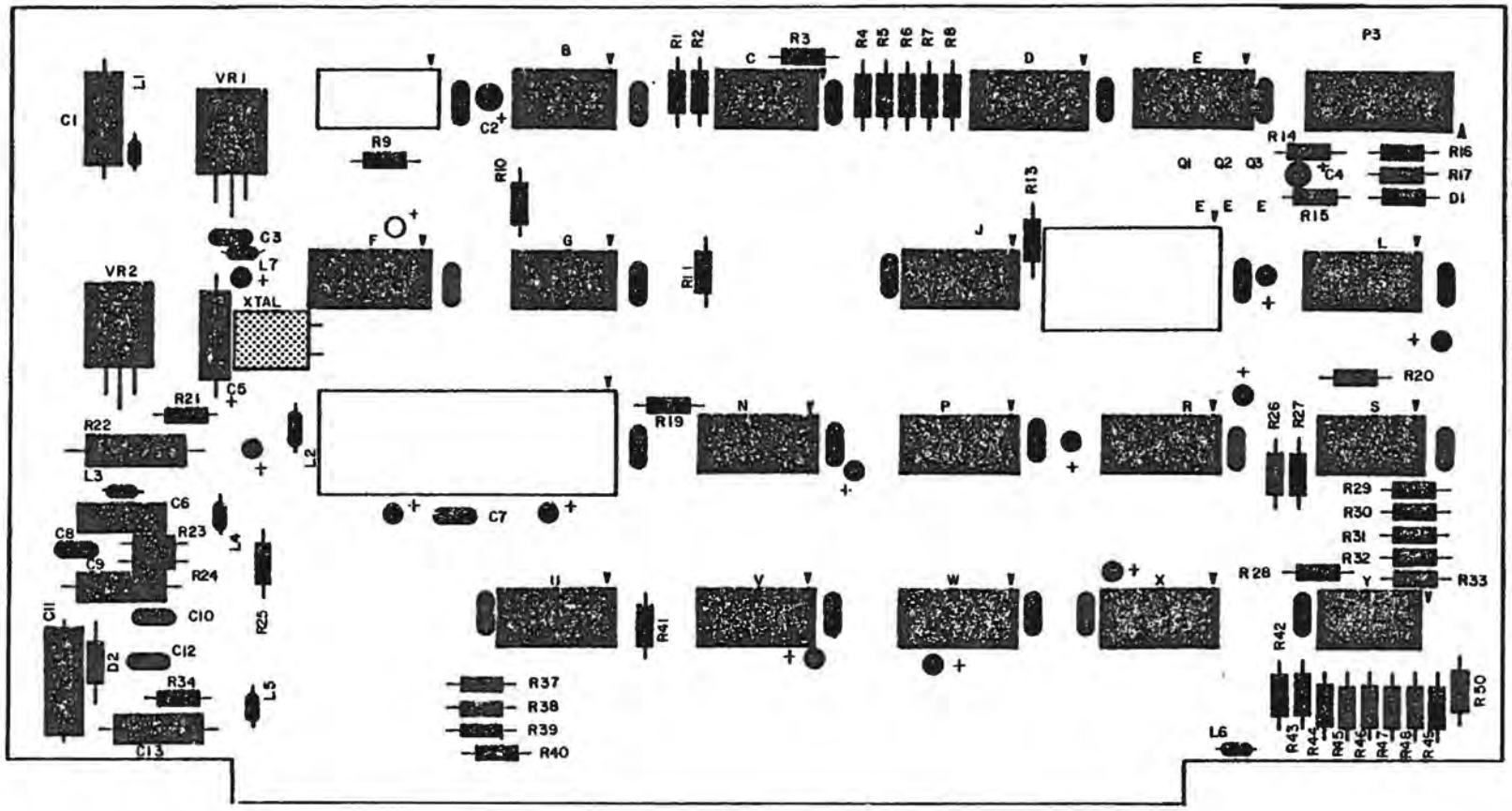


Figure 5-22 Crystal Installation

K. Static Sensitive MOS IC Installation Instructions

The final components to be installed on the CPU board are the static sensitive MOS Integrated Circuits (IC's A and M) and IC K. These IC's should be installed in their respective sockets on the board.

> NOTE
>
> IC's A AND M ARE MOS STATIC SENSITIVE IC's. IT IS MOST IMPORTANT TO FOLLOW THE SPECIAL INSTRUCTIONS IN SECTION 5-2 FOR HANDLING MOS IC's. FAILURE TO FOLLOW THESE PRECAUTIONS MAY RESULT IN A DAMAGED COMPONENT.

Silk-screen Designation	IC Part Designation	Socket Size
( ) K	8212	24-pin
( ) M	8080	40-pin
( ) A	4009	none

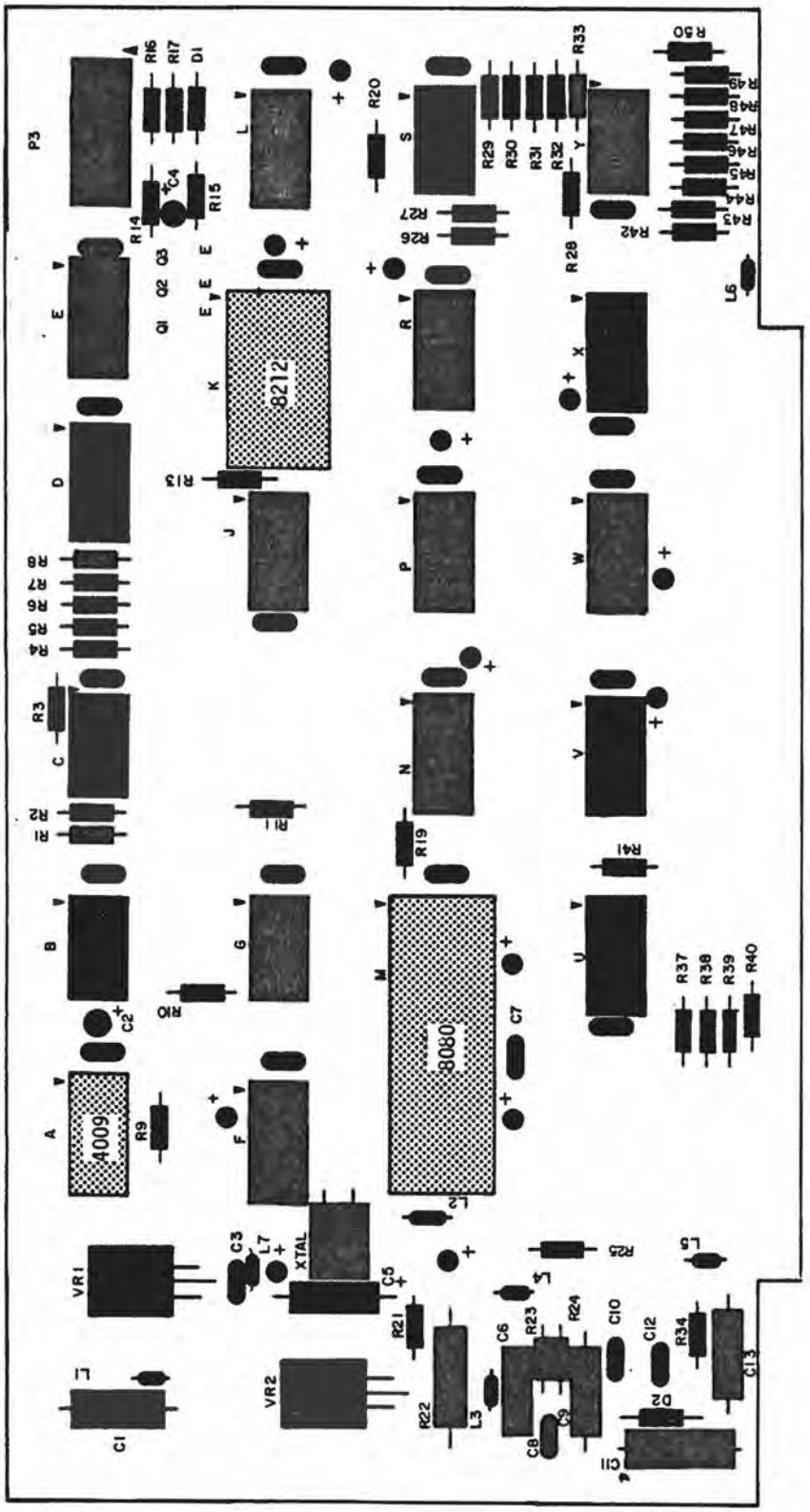


Figure 5-23 Static Sensitive MOC IC Installation

5-4. Turnkey Module

A. Description

The Turnkey Module board contains a serial I/O channel, 1K of RAM memory, and provisions for 1K of PROM memory.

B. Diode Installation

Referring to the diode installation instructions in section 5-2, install the three diodes (Bag 3) in the designated locations on the board.

NOTE
IT IS VERY IMPORTANT TO ORIENT THE CATHODE END (MARKED WITH A DARK BAND) OF THE DIODE OVER THE BANDED MARKING ON THE BOARD.

Diodes	Part Number
( ) D1	
( ) D2	IN914
( ) D5	

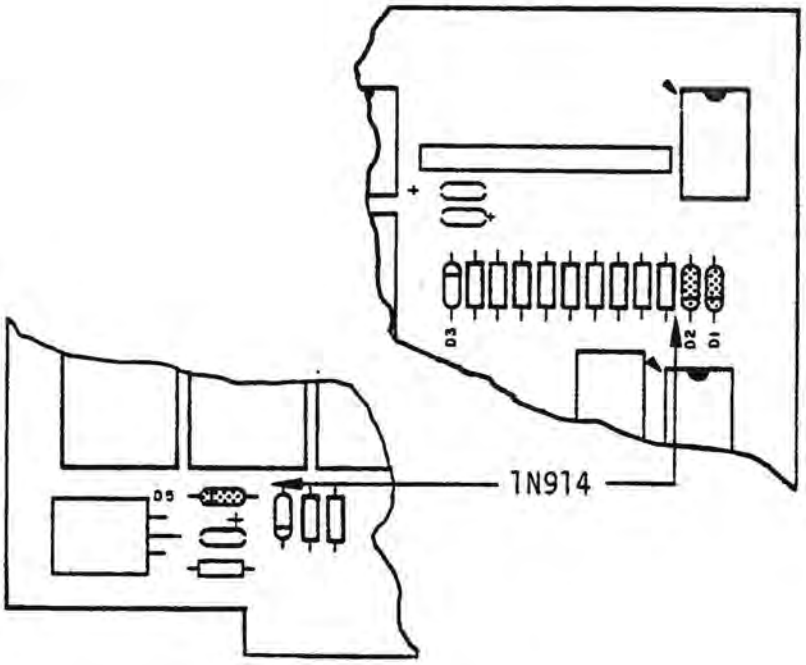


Figure 5-24 Diode Installation

C. Zener Diode Installation

Bag 3 contains 2 Zener diodes. Again, follow the diode installation instructions (section 5-2) for installing these components. The part number is printed in minute numerals on the diode. A hand lens will help distinguish the Zener diodes should they accidently become mixed with the other diodes. It is also necessary to orient the cathode (banded) end of these diodes over the banded markings on the board.

Zener Diodes

( ) D3 } IN4739
( ) D4 }

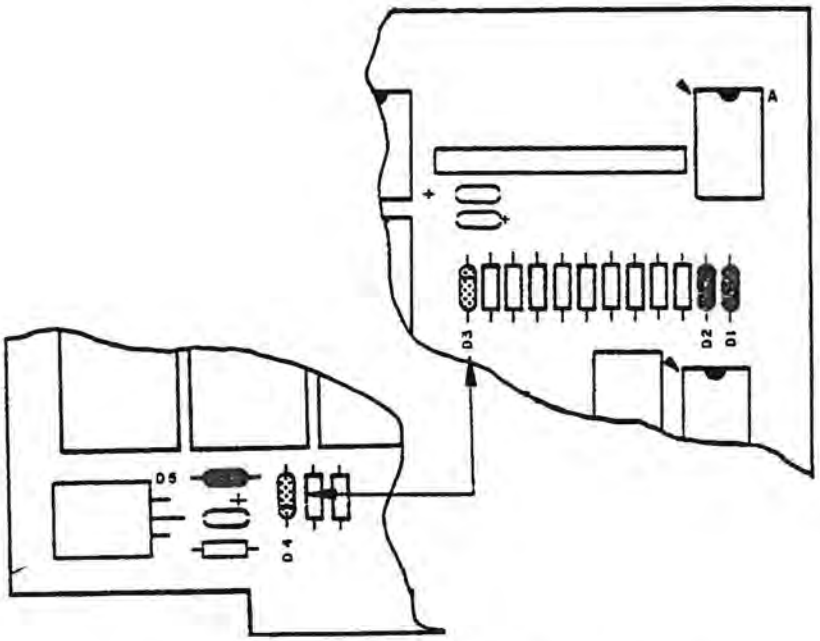


Figure 5-25 Zener Diode Installation

D. Resistor Installation

There are 16 resistors (Bag 2) on the Turnkey Module board. The box below notes the value and location. Follow the resistor installation instructions in section 5-2.

Resistor	Values
( ) R1	3K ohm 1/4w 5% (orange, black, red)
( ) R2 ( ) R3	220 ohm 1/4w 5% (red, red, brown)
( ) R4	1.5K ohm 1/4w 5% (brown, green, red)
( ) R5 ( ) R6 ( ) R7 ( ) R8	4.7K ohm 1/4w 5% (yellow, purple, red)
( ) R9 ( ) R10 ( ) R15	330 ohm 1/4w 5% (orange, orange, brown)
( ) R11	10 meg ohm 1/2w 5% (brown, black, blue)
( ) R12 ( ) R13 ( ) R14	1K ohm 1/4w 5% (brown, black, red
( ) R16	1.8K ohm 1/4w 5% (brown, silver, red)

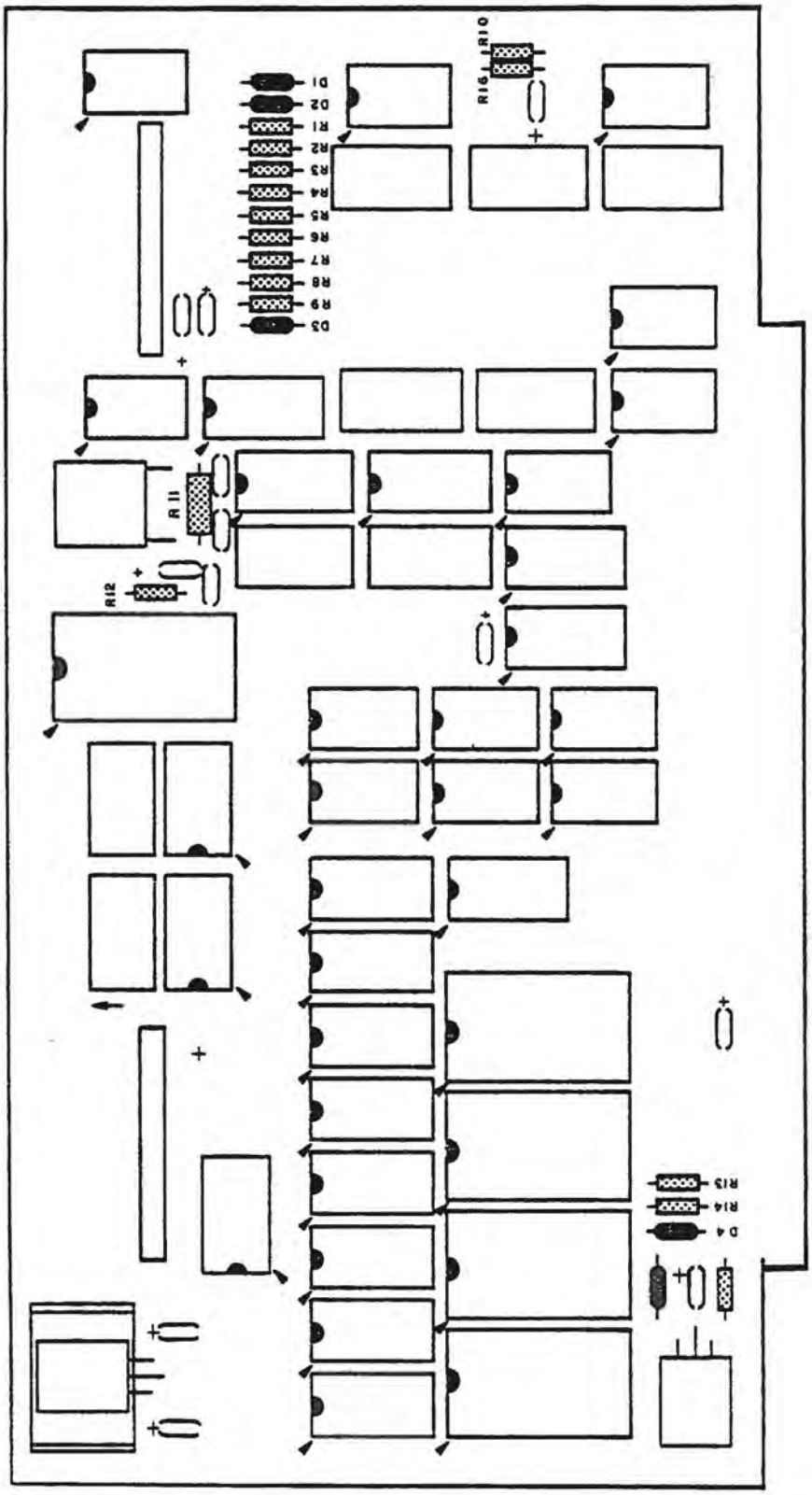


Figure 5-26 Resistor Installation

E. Capacitor Description

Twelve (12) capacitors (Bag 3) are installed at this point on the Turnkey Module board. There are three (3) ceramic disk capacitors and nine (9) epoxy dipped tantalum capacitors.

Save some clipped capacitor leads; these will be used as jumper connections in the next assembly procedure.

1) Ceramic Dipped Disk Capacitor Installation Instructions
   The three (3) ceramic disk capacitors have no polarity orientation, however, the capacitors are of different values. The value is marked on each capacitor.

Insert the ceramic dipped disk capacitors into the designated locations on the board.

Ceramic Dipped Disk Capacitors (Figure 5-27a)

	Value
( ) C4	56 pf 20v
( ) C5	
( ) C7	470 pf 1Kv

The capacitor marked 47Ø is to be inserted in the space marked C7.

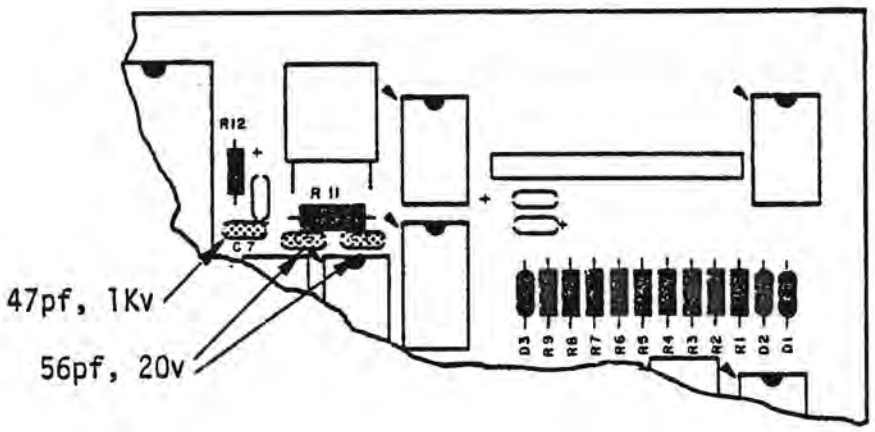


Figure 5-27a Ceramic Dipped Disk Capacitor Installation

2) Epoxy Dipped Tantalum Capacitors Installation

The nine (9) epoxy dipped tantalum capacitors have a polarity marking (+) over one lead. These capacitors must be oriented with the positive marking on the board.

NOTE
DUE TO SUPPLY VARIATIONS, SOME CAPACITORS MAY BE BLUE OR RED ON ONE SIDE TO NOTE POLARITY.

Following the instructions for Tantalum dipped capacitor installation, insert the capacitor in the designated spaces.

NOTE
THERE ARE FIVE (5) 35V CAPACITORS AND FOUR (4) 16V CAPACITORS. BE SURE TO INSERT THEM IN THE PROPER LOCATION AS DEFINED BY THE TABLE, AND FIGURE 5-27b.

Epoxy Dipped Tantalum Capacitors (Figure 5-27b)

( ) C1 ( ) C3 ( ) C6 ( ) C8 ( ) C9	1MF 35V
( ) C2 ( ) C10 ( ) C12 ( ) C13	22 MF 16V

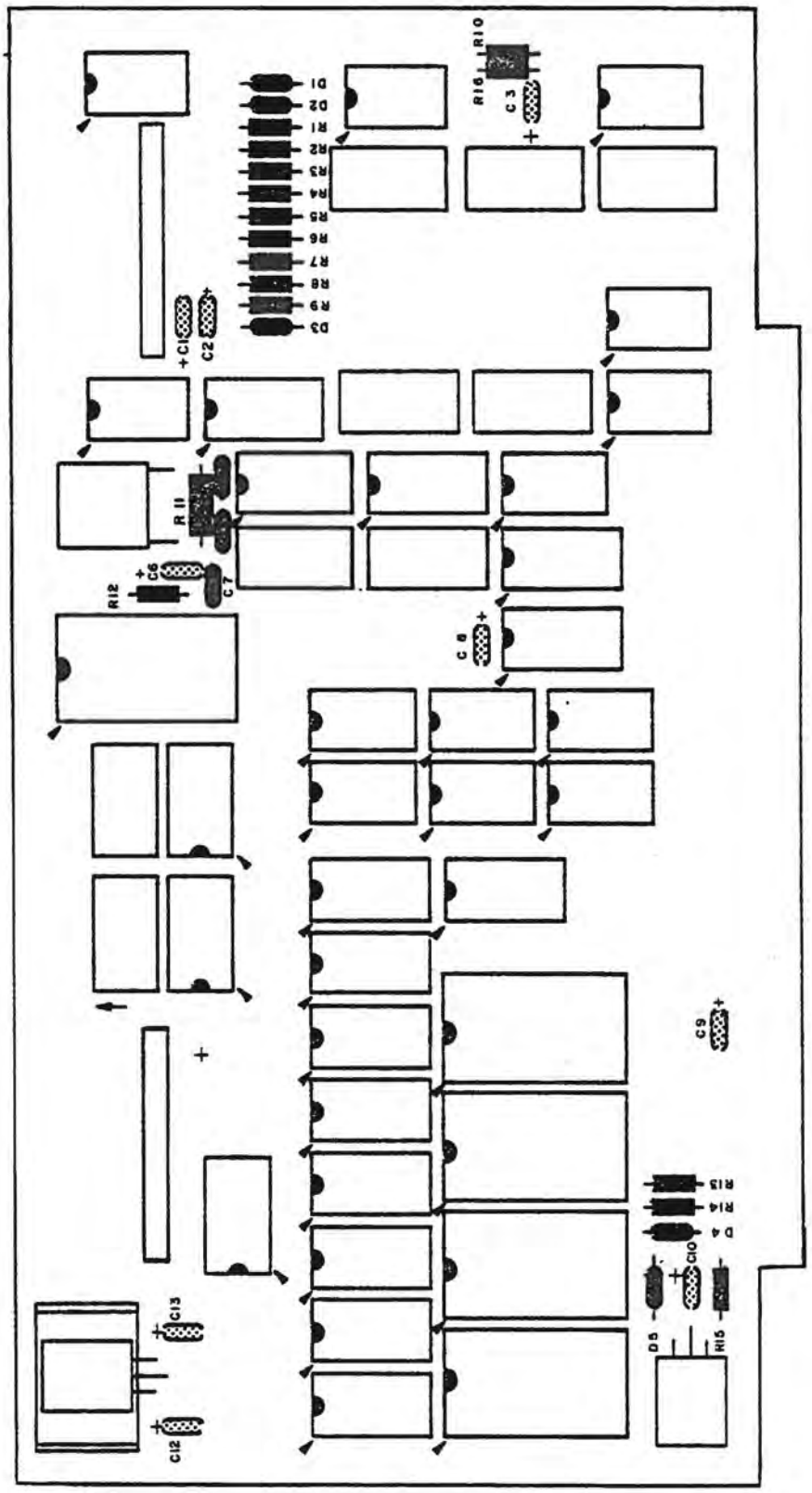


Figure 5-27b Epoxy Dipped Tantalum Capacitor Installation

F. Jumper Connections Installation

There are several jumper connections which may be installed on the Turnkey Module Board. The number and locations of these jumper connections depends upon the various options and interface capabilities the user wishes to utilize. Section 3 of this document (Advanced Operation) provides a detailed description of jumper connections to be installed to provide the desired optional capabilities. In addition, Section 3 also provides illustrations and defines the location of the pads to be jumpered. If jumper connections are needed on your board, insert them at this time according to the following instructions.

1. Cut a dipped capacitor lead length of one inch.
2. Using needle-nosed pliers bend the lead into an arch. (See Figure 5-28).
3. Insert each end of the lead into the holes of the pad specified in the description in Section 3.
4. Solder each end of the lead in place.
5. Clip off excess lead.

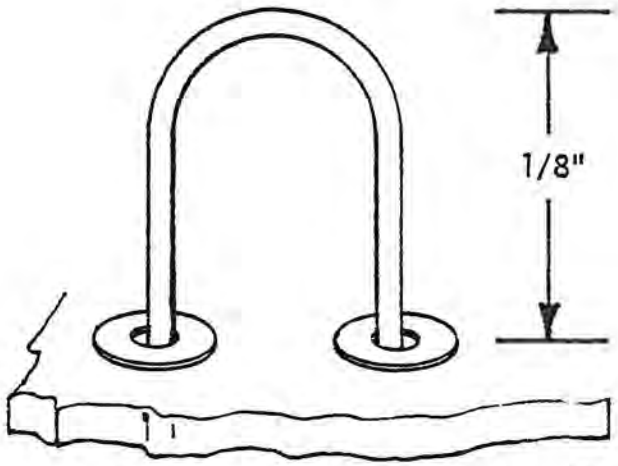


Figure 5-28 Jumper Connection Installation

8800b-T
August, 1977

G. Transistor Installation

The Turnkey Module board has two (2) transistors which are installed at this point in the assembly process. The transistors are contained in bag 3.

Instructions for transistor installation are provided in section 5-2. In addition there are illustrations which depict the installation of the various types of transistors.

It is most important to insert the emitter lead in correct hole in the board.

NOTE
ACCORDING TO SUPPLY VARIATIONS, SOME KITS MAY CONTAIN A METAL TRANSISTOR. THIS TYPE OF TRANSISTOR IS NOT DISPLAYED ON THE TRANSISTOR IDENTIFICATION CHART. FIGURE 5-30 ILLUSTRATES THIS TYPE OF TRANSISTOR.

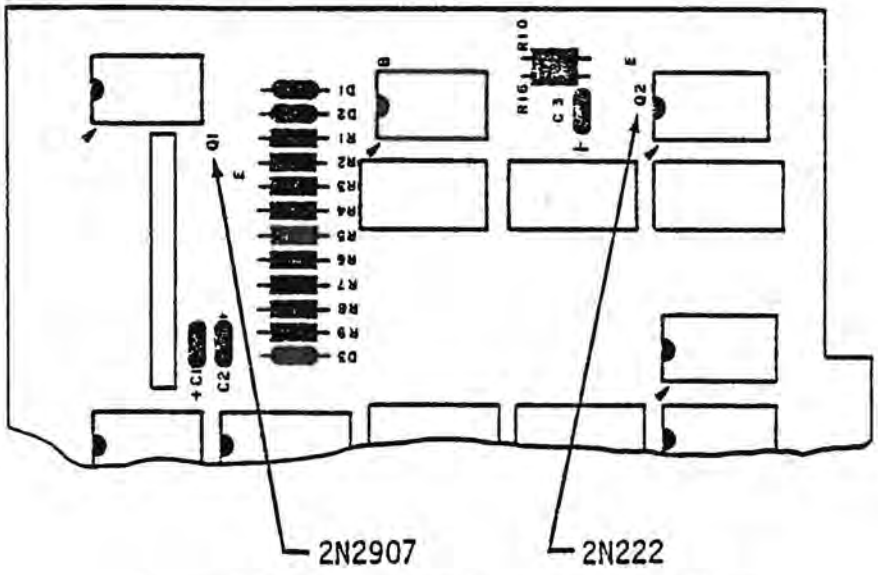


Figure 5-29 Transister Installation

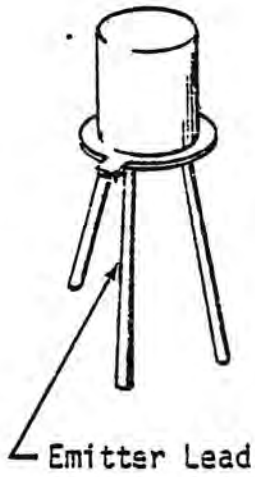


Figure 5-30
Metal Transister

Transistors

	Part Number
( ) Q1	2N2907
( ) Q2	2N222

H. Voltage Regulator/Heat Sink Installation

The Turnkey Module has two (2) voltage regulators (Bag 1) one of which (78Ø5) is installed over a heat sink.

1) Use heat sink grease when installing the 78Ø5. Apply grease to all surfaces (i.e. bottom of voltage regulator, bottom of heat sink, mica insulator).
   1. Set the regulator (78Ø5) in place on the silk-screened side of board.
   2. Mark each lead to correspond with the proper holes in which the leads will be inserted on the board.
   3. Using needle nose-pliers, bend each lead at a right angle on each mark.

NOTE
ACCORDING TO SUPPLY VARIATIONS, SOME KITS MAY INCLUDE A MICA INSULATING STRIP. THIS IS INSERTED BETWEEN THE VOLTAGE REGULATOR AND HEAT SINK.

   4. Referring to Figure 5-31b set the voltage regulator on the heat sink, inserting the leads in the proper holes on the board. Secure both in place with a #6-32 x 3/8" screw, a #6-32 nut, and a #6 lockwasher.
   5. Turn the board over and solder the leads in place, taking care to leave no solder bridges. Clip off excess lead length.

Voltage Regulator/Heat Sink

( ) Voltage Regulator 78Ø5

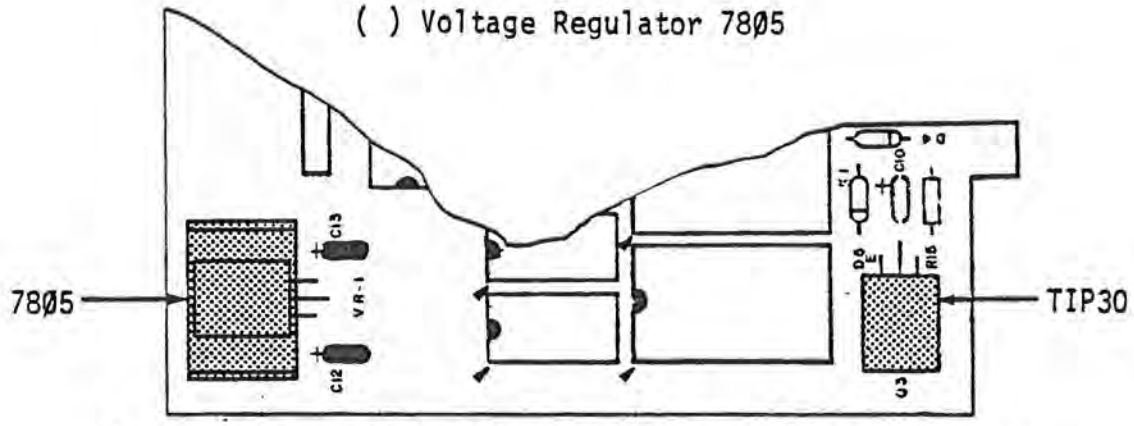


Figure 5-31a. Voltage Regulator/Heat Sink Installation

2) Voltage regulator TIP-30 is merely soldered into place on the board.

1. Place the voltage regulator in the proper position on the board.
2. Make a mark on each lead to correspond with the proper hole on the board.
3. Using needle-nose pliers, bend each lead at right angles on the mark to correspond to the correct holes on the board.
4. Turn the board over and solder each lead in place.

Voltage Regulator

( ) Voltage Regulator TIP 3Ø

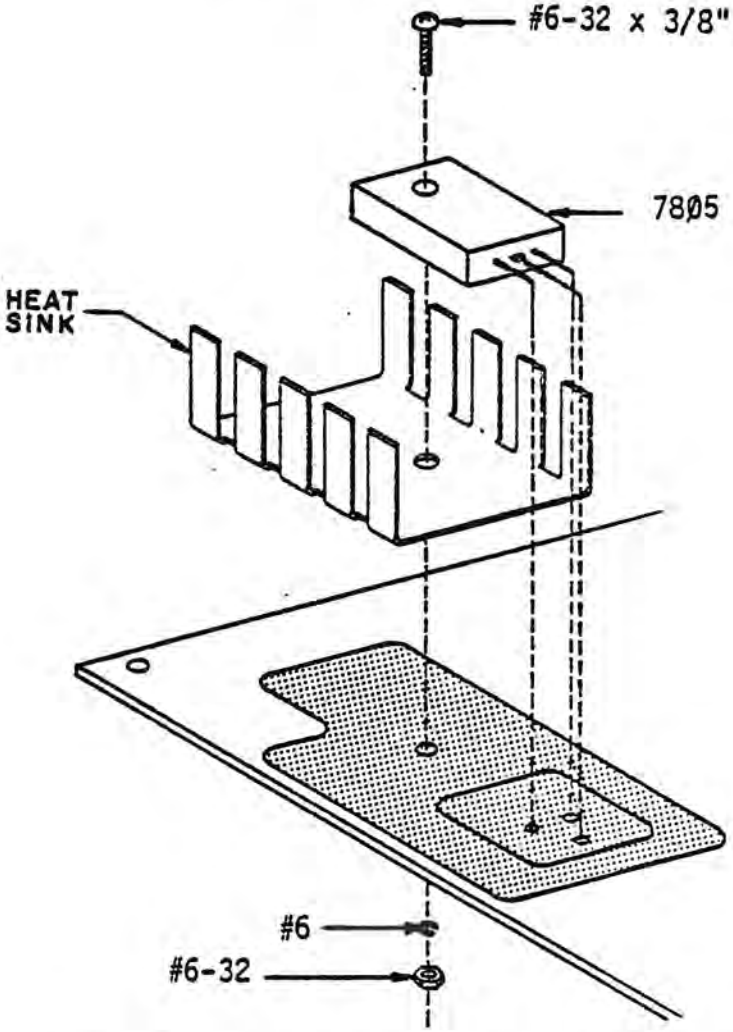


Figure 5-31b. Voltage Regulator/Heat Sink Orientation

I. Integrated Circuit Description

There are 31 Integrated Circuits on the Turnkey module board. Nine of these IC's are static sensitive MOS chips. All IC's on the turnkey module board are installed in sockets. The MOS chips are not inserted in the sockets at this time. These will be the last components installed on the board.

J. IC Socket Installation

1) Insert the thirteen 14-pin sockets (Bag 5) in the specified locations on the board. Follow the instructions in section 5-2 for installing IC Sockets.

Take care not to leave any solder bridges between pins. This is a common cause of problems.

14 Pin Sockets

( ) A	( ) C
( ) B	( ) U
( ) D	( ) W
( ) E	( ) T
( ) F	( ) S
( ) K	( ) V
( ) L	

2) The 16-pin IC sockets (Bag 5) are installed at this point following the instructions in 5-1.

16 Pin Sockets

( ) N1	( ) A1
( ) M1	( ) Y
( ) G1	( ) Z
( ) F1	( ) M
( ) D1	( ) N
( ) C1	
( ) B1	

3) The last four components to be installed on the Turnkey module are the 24-pin sockets. The ICs for these sockets are not included in the kit since they are optional PROM ICs to expand the capabilities of the Altair 8800b Turnkey Computer. These PROMs include:

   a. PROM monitor
   b. Disk Bootloader PROM for loading programs, DOS and BASIC from disk.
   c. Multi-Bootloader PROM for loading programs or BASIC from paper tape.
   d. Other PROM options which can be programmed by users with special equipment for specific applications.

24 pin sockets

( ) L1
( ) K1
( ) J1
( ) H1

K. IC Installation

Carefully insert the IC's (Bag 1) in the sockets specified in Figure 5-32. Follow the instructions in section 5-2E for insertion of IC's in sockets. The table will further define where specific IC's should be inserted.

The IC designation is printed on each IC.

IC	Designation
( ) N1	74367
( ) M1	74367
( ) P1	74367
( ) V	74Ø4
( ) S	74LS73
( ) W	74ØØ
( ) T	74LS73
( ) C	74Ø2N
( ) U	74Ø2N
( ) R	8216
( ) P	8216
( ) M	74LS257
( ) N	74LS257
( ) L	743Ø
( ) F	75189
( ) M	47Ø2PC
( ) A	75188
( ) B	743Ø
( ) D	743Ø

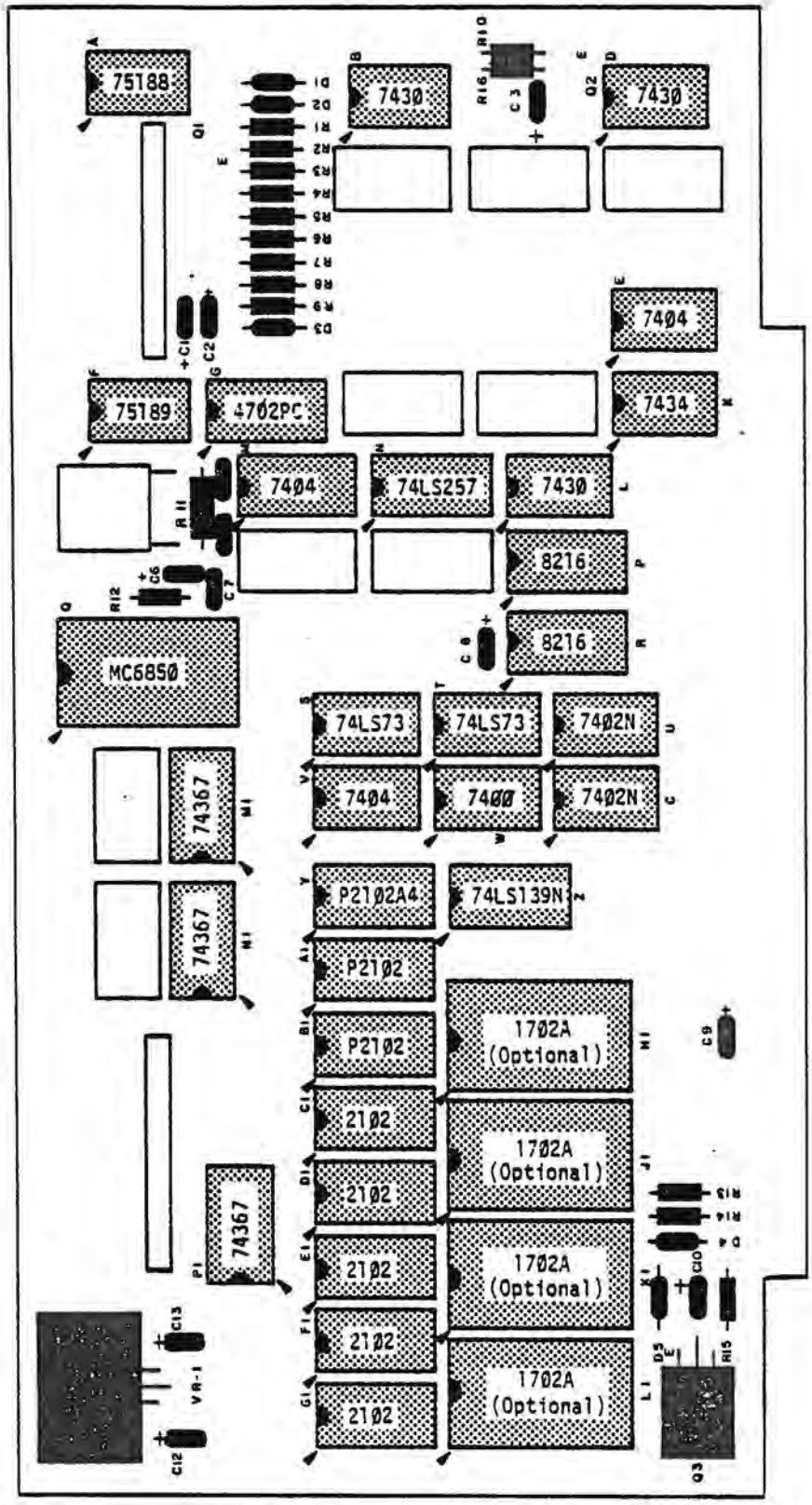


Figure 5-32 IC Installation

L. DIP Switch Installation

The nine (9) DIP (Dual Inline Package) switches are installed at this point in the assembly procedure. All the switches are identical; it is their placement on the board that determines their function.

Switches SW-6 and SW-7 are sense switches. Software, via the switches determine which I/O devices will be used. Switches SW-8 and SW-9 are the AUTO-START address switches. The SIO port address is established through switches SW-4 and SW-5. Switches SW-1 through SW-3 are memory address switches. Numbering for each individual switch of these components is silk-screened on the board. The numbering of these switches corresponds to the bits of the starting addresses of the RAM and PROM memory blocks.

For complete information regarding the addressing of the switches refer to Section 3-2, pages 24-26, the theory section of this document.

The DIP switches are installed in much the same manner as integrated circuits.

1. Remove the switches from Bag 5. Insert the pins carefully in the holes within the designated squares. Begin at the top of the board with switches SW-6 and SW-7 and work down the board inserting the switches.
2. Turn the board over and solder all pins in place. Do not leave any solder bridges.

DIP Switch	Part Designation
( ) SW6	CTS2Ø6
( ) SW7	
( ) SW8	
( ) SW9	CTS2Ø6
( ) SW4	
( ) SW3	
( ) SW1	
( ) SW2	

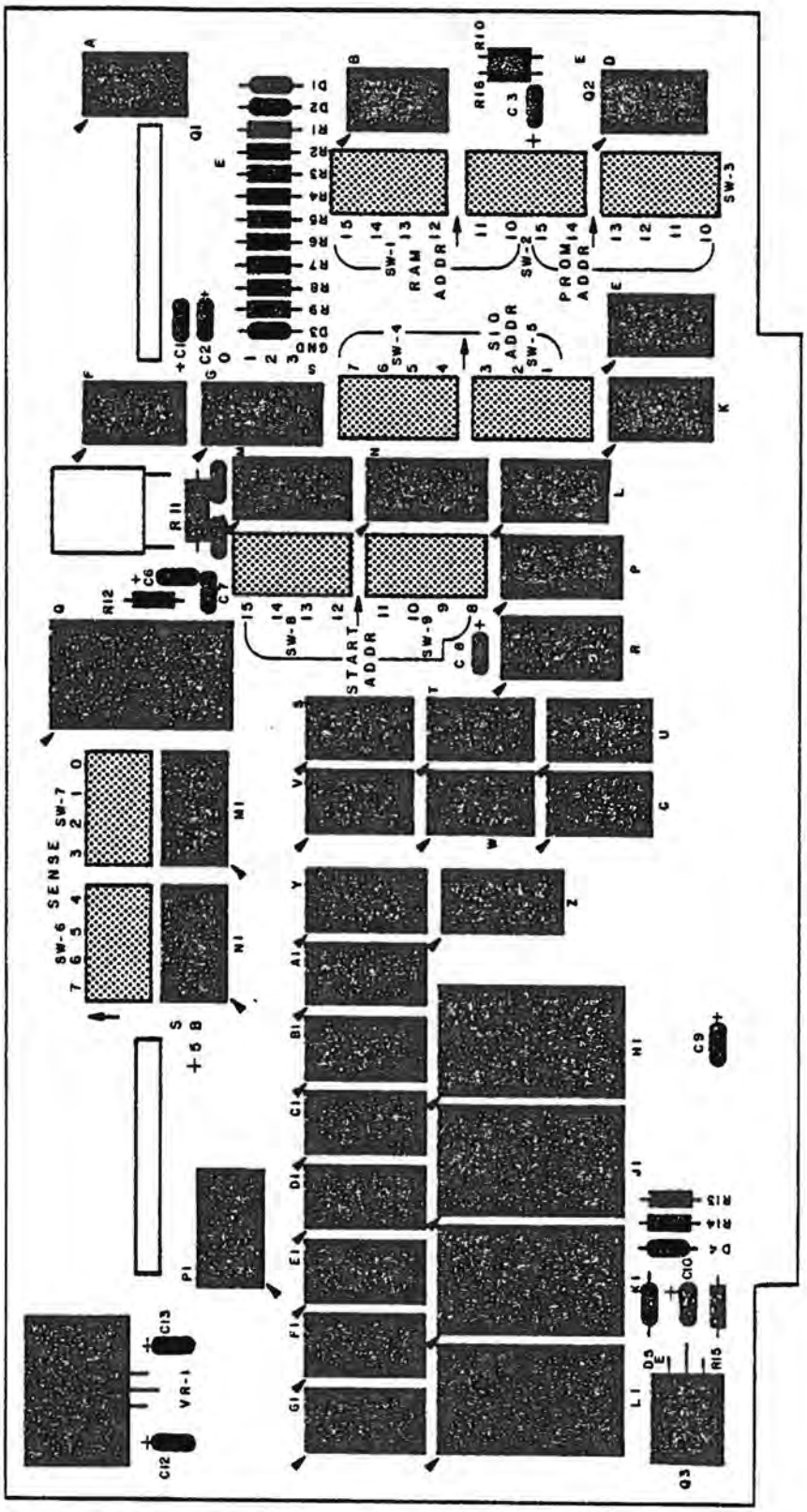


Figure 5-33 DIP Switch Installation

M. Crystal Description

There is one 2.4576 MHz crystal (Bag 4) to be installed on the Turnkey Module Board. It is imperative that the crystal case does not come in contact with any of the bands on the board. Such contact may cause permanent damage to the crystal and/or board.

1. Insert crystal leads in designated holes on silkscreened side of the board.
2. Bend leads, so that the crystal rests flat within the designated square on the board.
3. Turn the board over and solder the leads in place.
4. Clip off excess lead length.

Crystal	Part Number
( ) XTAL-1	2.4576

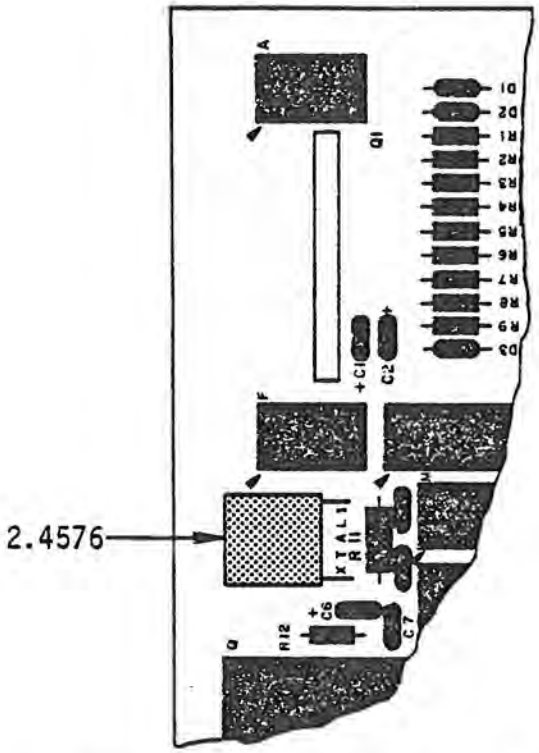


Figure 5-34 Crystal Installation

N. Male Connector

The final components to be installed on the Turnkey Module Board are two 10-pin male connectors (Bag 1).

1. Orient the connector as shown in the illustration with long, curved pins pointed toward the top of the board.
2. Insert the short pins in the designated holes.
3. Solder each pin in place on the bottom of the board, taking care to leave no solder bridges.
4. Clip off excess lead lengths.

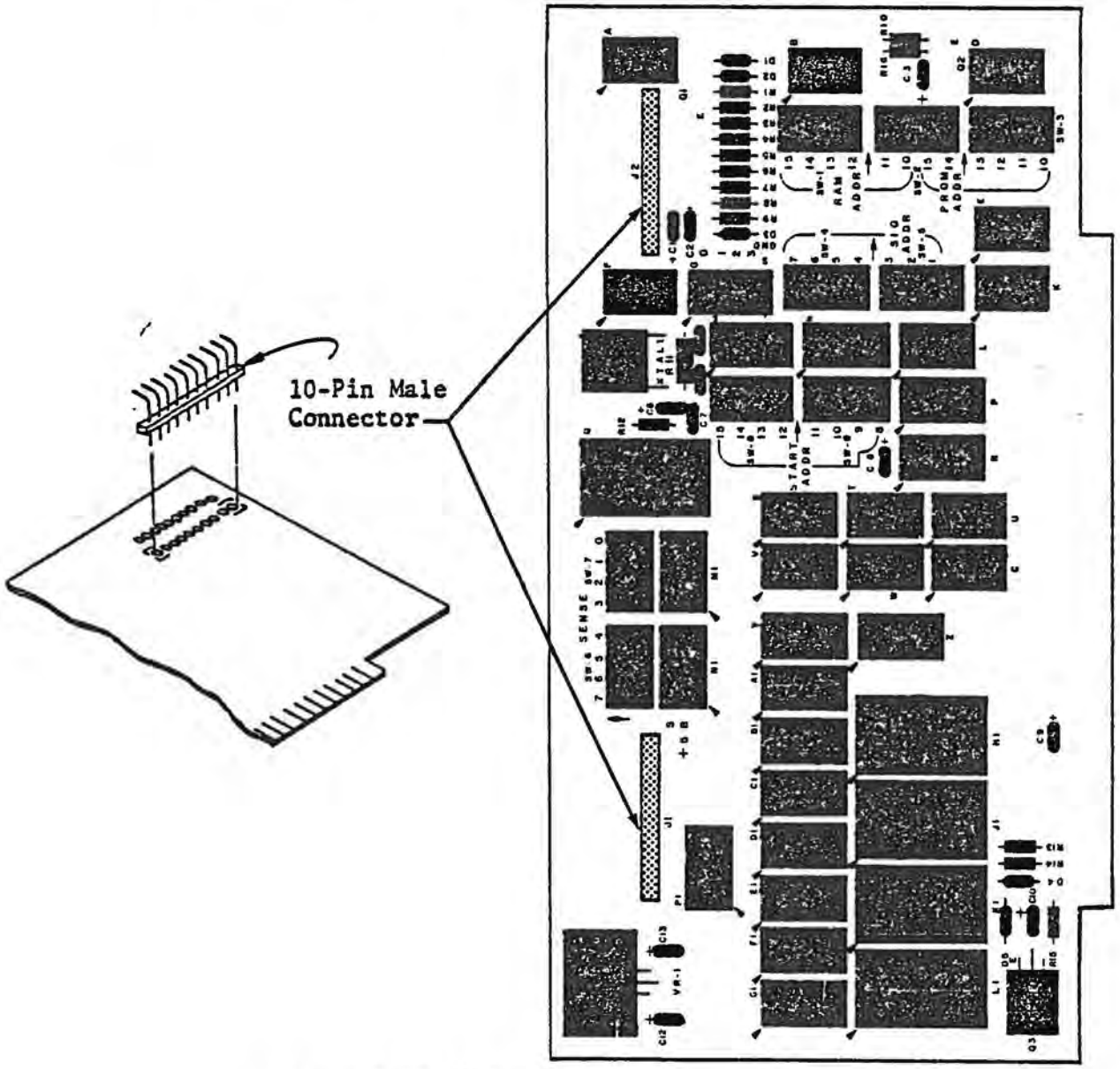


Figure 5-35 Male Connector Installation

5-5. Front Panel Display/Control Board

This small board consists of five (5) LEDs (Light Emitting Diodes), two (2) switches, five (5) resistors and one (1) diode. It is attached to the Turnkey Module by a flat cable. The numerals in the upper right corner are engineering description numbers. The number in the box designated "S.N." is the serial number of the Display/Control board.

A. Resistor Installation Instructions

Install the five (5) resistors (Bag 2) according to the directions in Section 5-2, in the specified locations on the board. (See Figure 5-36)

Resistor	
( ) R17 through R21	220 ohm (red, red, brown) 1/4W

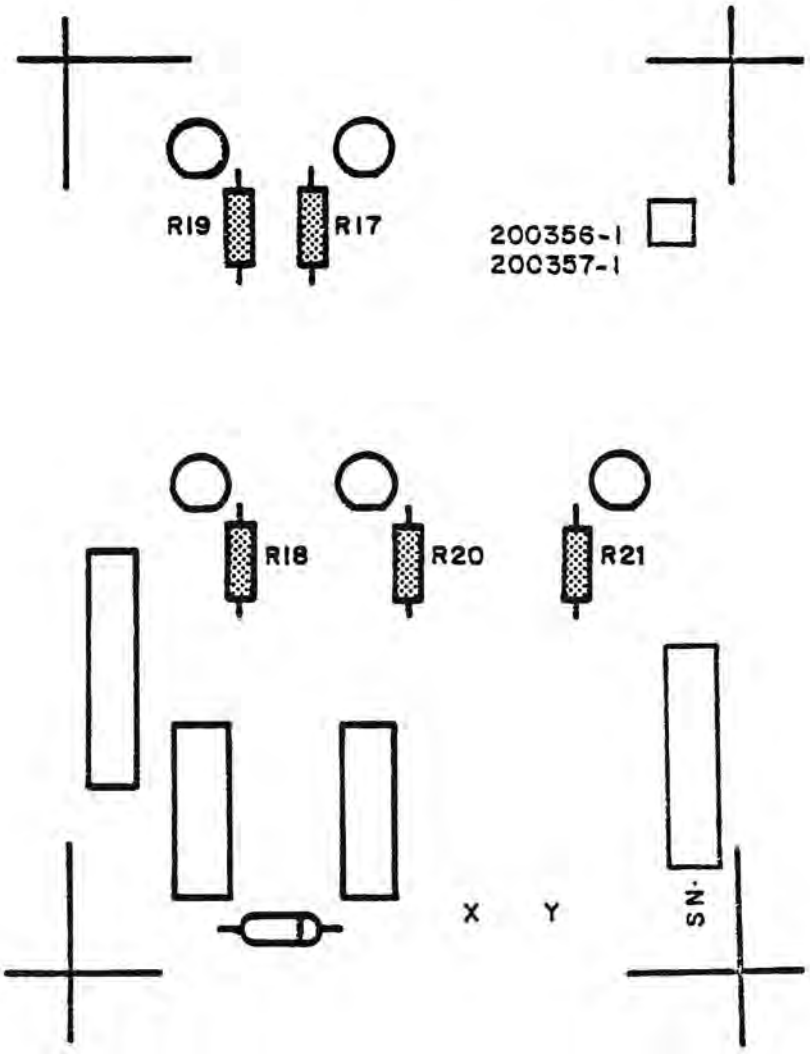


Figure 5-36 Resistor Installation

B. Diode Installation Instructions

Figure 5-37 defines the location of the single diode (Bag 3) to be installed on the D/C board. Follow the instructions in Section 5-2 for installing this component. Be sure the banding on the diode corresponds with the banding on the board.

Diode	Part Number
( ) D6	1N914

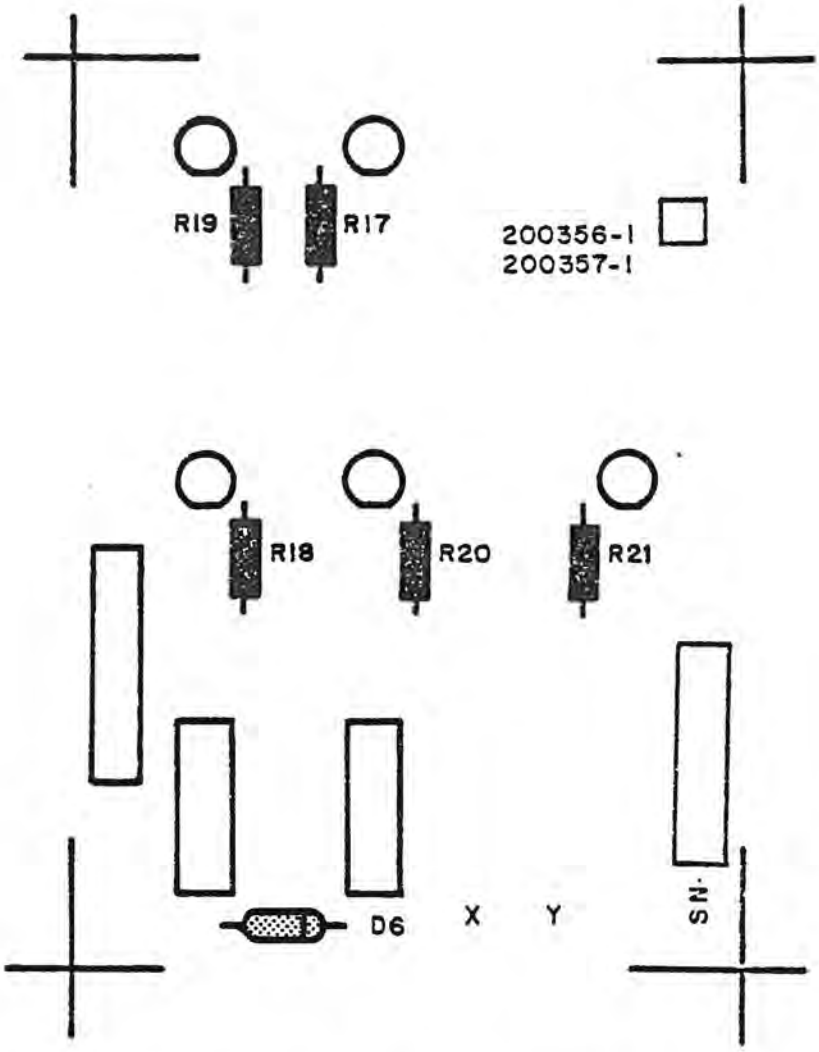


Figure 5-37 Diode Installation

C. Ribbon Cable Installation

A flat eight (8) stranded wire ribbon cable connects the D/C board to the Turnkey Module board. This cable will be installed on the D/C board at this point in the assembly process.

1. Carefully strip about 1/2 inch of insulation off each individual wire from one end of the flat cable. Strip 1/8 inch from the other end. Apply a thin coating of solder to the exposed portion of each wire.
2. Insert a 1/2 inch stripped wire in each hole on the silk-screened side of the board (as indicated in Figure 5-38a and 5-38b). Push the wire in until the insulation touches the board.

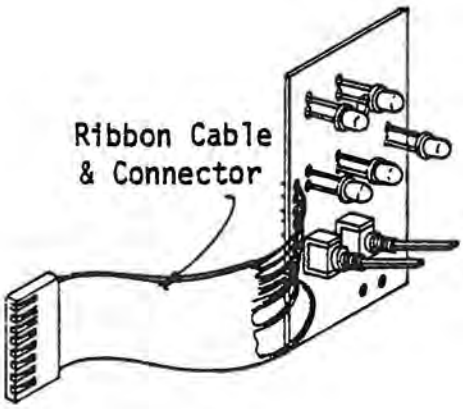


Figure 5-38a
Ribbon Cable Orientation

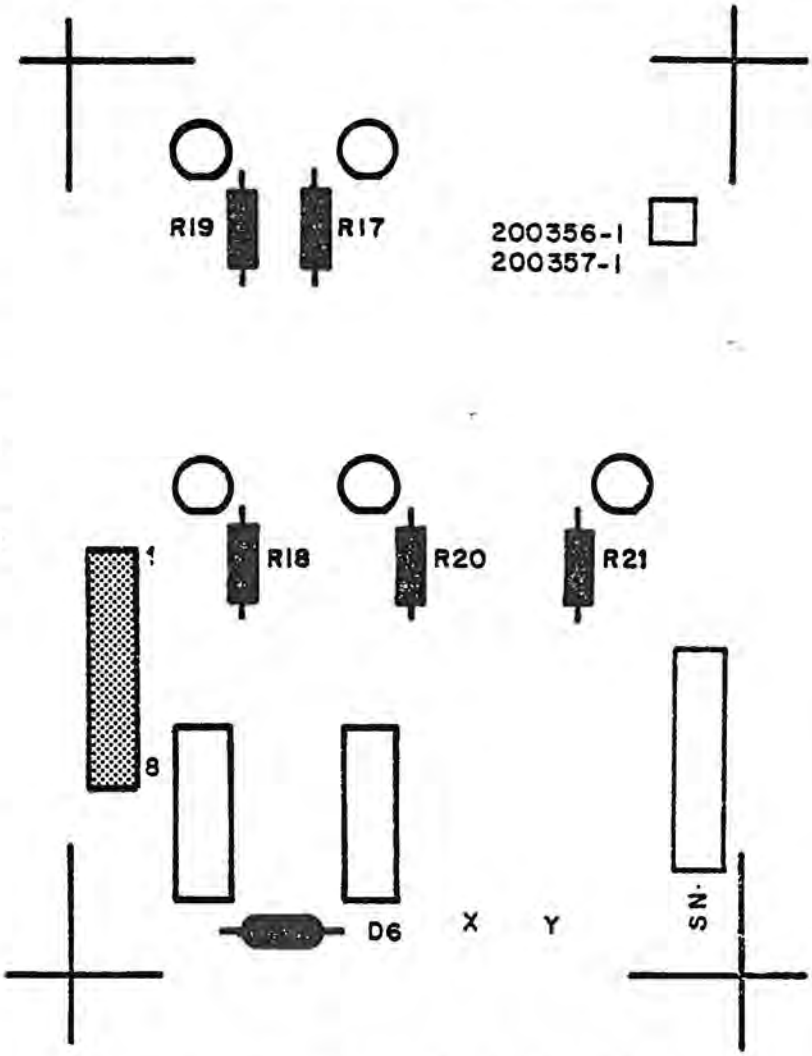


Figure 5-38b Ribbon Cable Installation

3. Turn the board over and solder each wire in place. Be very careful not to leave any solder bridges. Clip off excess lengths of wire.
4. Using Figure 5-39 (a and b) as a guide, install a terminal pin (Bag 2) on each wire of the flat cable. Crimp the pin with needle-nose pliers to attach it to the wire.

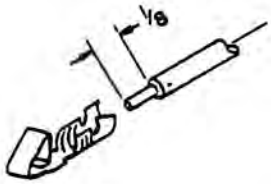


Figure 5-39a
Terminal Pin & Wire

Figure 5-39b
Crimping Terminal Pin

5. Bag 2 contains a female connector housing in which the terminal pins will be inserted. This housing is marked with tiny numerals over slots one (1) and ten (10) (as indicated in Figure 5-39c). Insert the wire from hole one (1) (as marked on the board) into slot one (1) of the female connector. Proceed through the wires in numerical order (i.e. wire 2 to slot 2). Slots nine (9) and ten (10) are not used. Make sure each connector "clicks" into place.

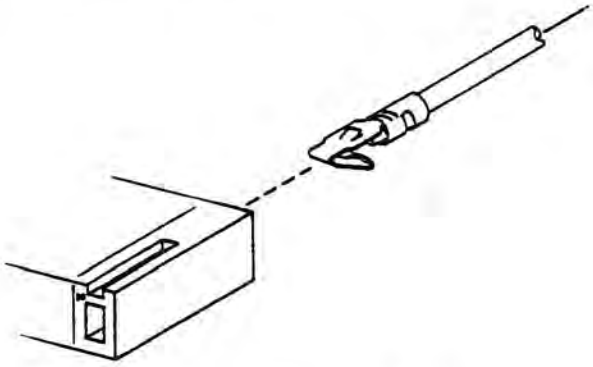


Figure 5-39c Terminal Pin Insertion

D. LED Installation

Bag 1 contains the five (5) LEDs which will be installed on the Turnkey D/C board. The LEDs must be inserted with the cathode lead in the correct location.

Notice in Figure 5-40a the LEDs are represented as circles with a flattened side. The base of the LEDs are also circular with a single flattened surface. This should correspond with the flattened marking on the board.

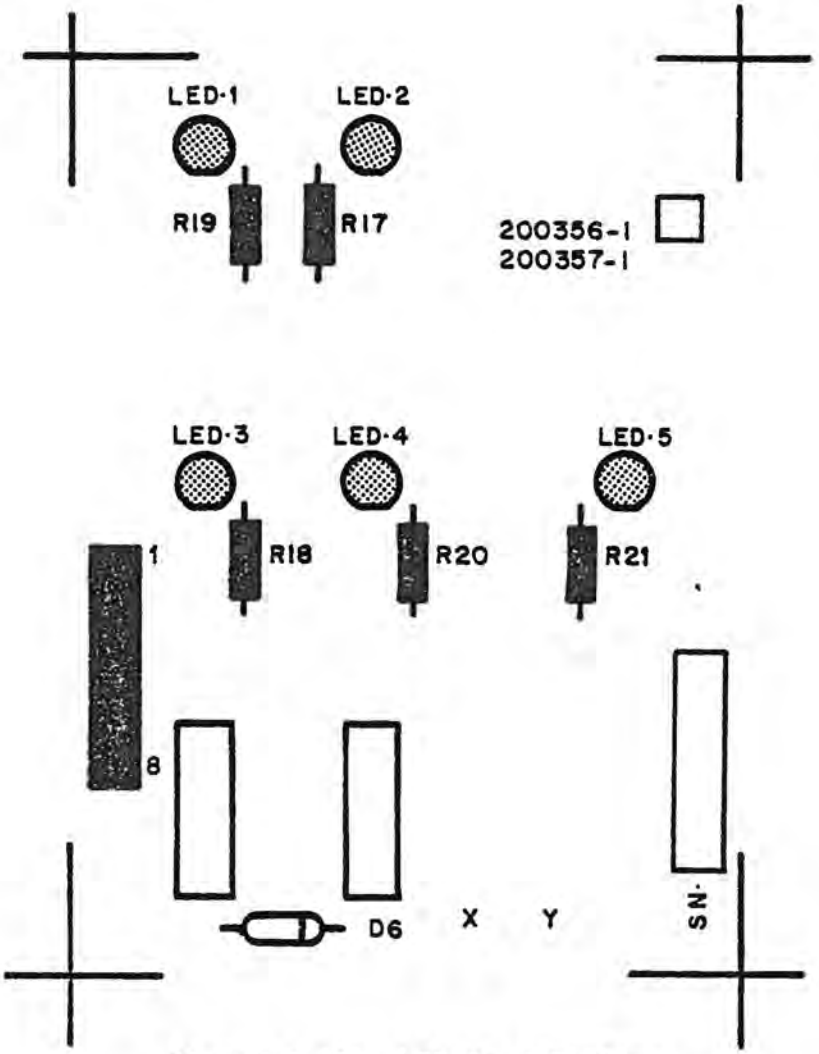


Figure 5-40a LED Installation

CAUTION
LEDs are heat sensitive. When soldering, heat quickly and for the shortest time possible to avoid damaging this component.

1. Insert the LEDs, taking care to match the flattened side of the LED to the flattened marking on the silkscreened side of the board. The leads of the LED should project above the board approximately 1/2 inch. (Figure 5-40b describes the process).
2. Solder the LEDs in place. Be sure there are no solder bridges between leads. It is not necessary to clip the leads.

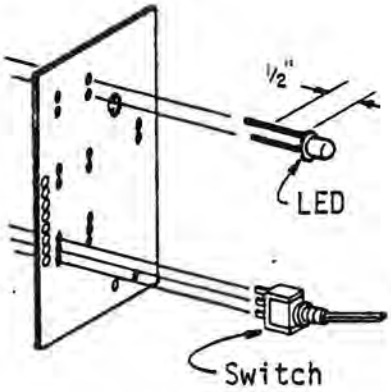


Figure 5-40b LED & Switch Orientation

E. Display/Control Switch Installation

The final components to be installed on the Turnkey Display/Control board are two switches; a two position Run/Stop switch which runs the program in memory or stops execution and a momentary contact switch (Start) which controls the AUTO-START logic. (See Section 3-1, page 23 of this document for more detailed information.)

1. Insert the two position switch (Bag 1, MITS part number 101879) in the area labelled "Stop." This switch need not be oriented in a specific direction; simply insert the three leads in the three holes in the "Stop" square. The washers etc. in this bag will be used in a later assembly procedure.

2. Insert the momentary contact switch in the three holes in the square marked "Start." This switch also need not be oriented in a specific direction.
3. Turn the board over and solder each lead in place. Do not cut excess leads.

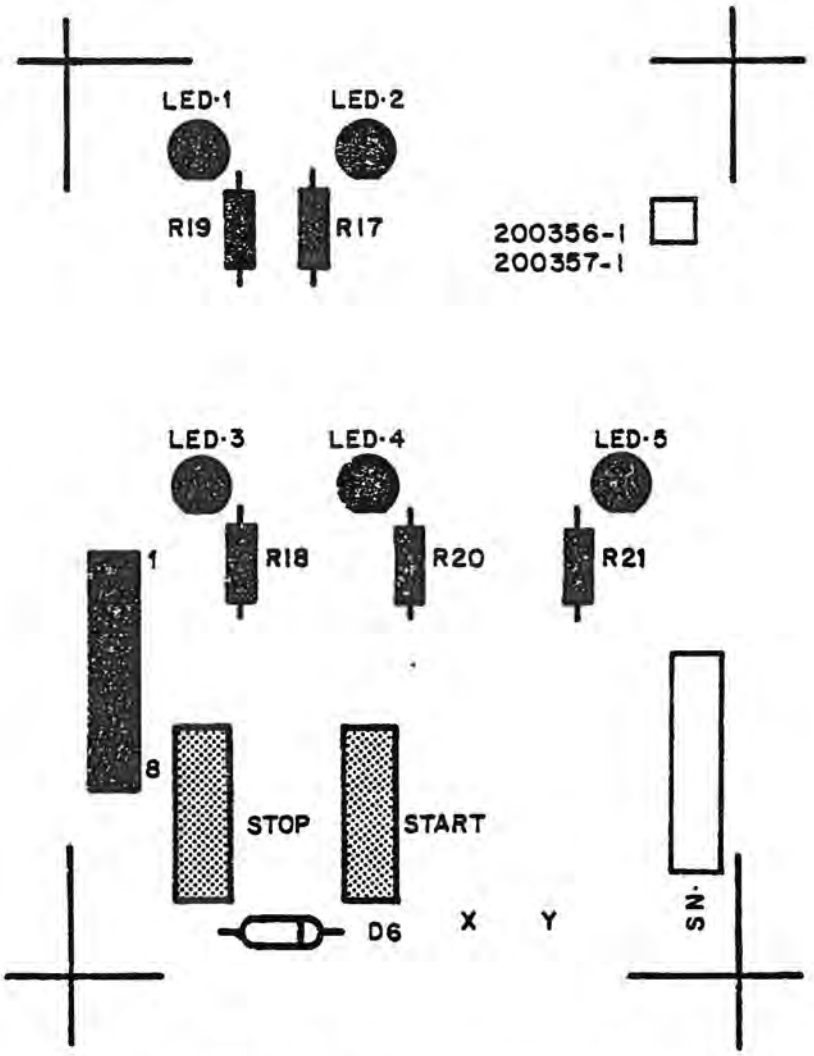


Figure 5-41 Display/Control Switch Installation

5-6. Cross Member Subassembly

A. Electrolytic Capacitor Installation Instruction

There are four large electrolytic capacitors C1-4 (Bag 3) which will be installed over four open areas on the board (Fig. 5-42). According to supply variations most capacitors will be of the variety which has an arrow with a negative (-) marking in it. Orient this arrow toward the negative polarity designation on the silkscreened side of the board.

1. Insert all the capacitors according to the instructions provided in Section 5-2. Do not try to push the capacitors all the way down into the open areas.
2. Solder leads in place on the back of the board.
3. Clip off excess lead lengths.
4. With these large components, it may be necessary to touch up the soldering around the lead on the front of the board.

Capacitor	Value
( ) C1	22- if. 25v electrolytic
( ) C2	
( ) C3	
( ) C4	

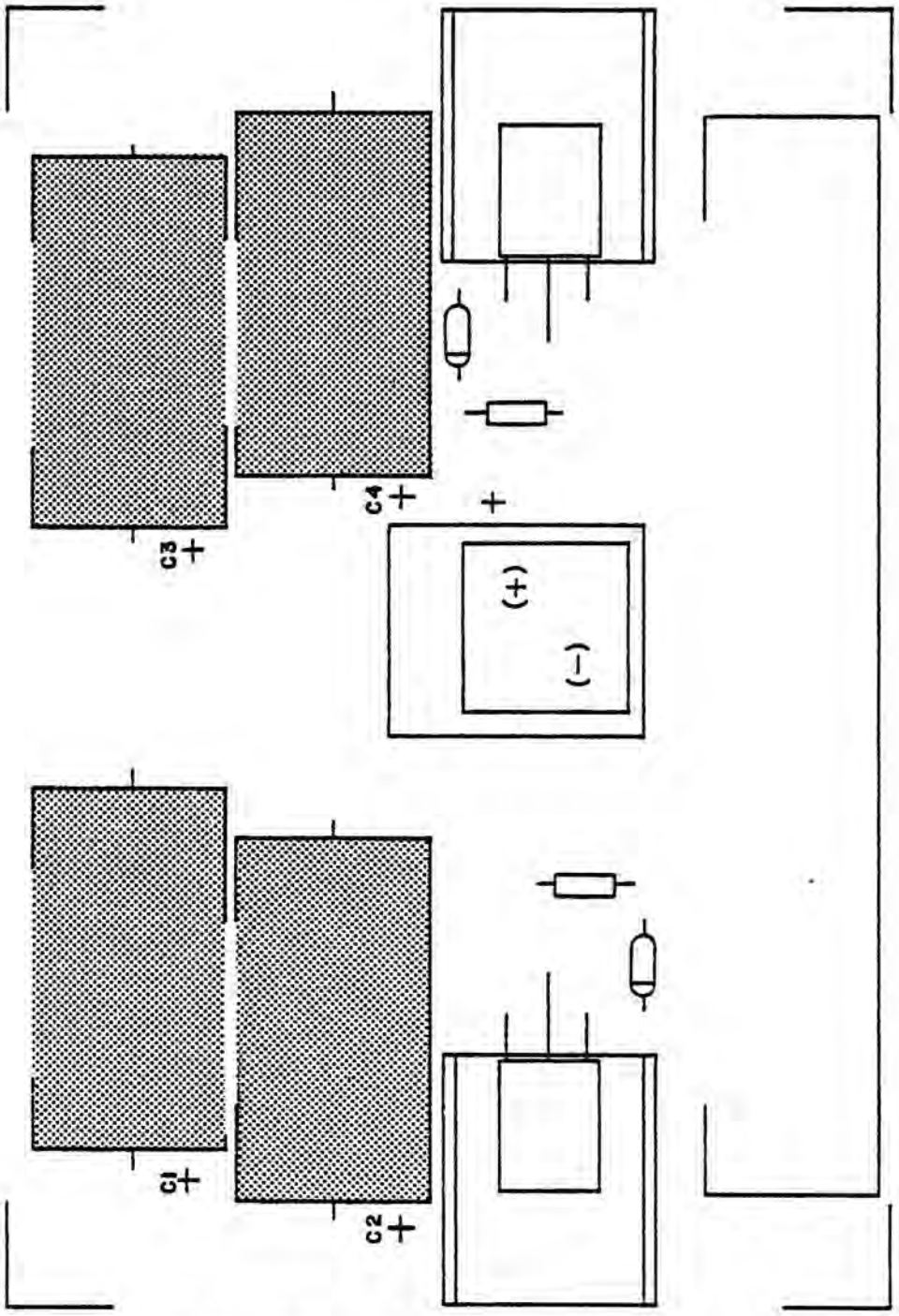


Figure 5-42. Electrolytic Capacitor Installation

B. Diode Installation

Install the two diodes (Bag 1) on the Power Supply Board according to the directions in section 5-2.

Be sure to place the banded end of the diode over the banded marking on the board.

Diode	Part #
( ) D1	IN4746
( ) D2	

C. Resistor Installation

Following the instructions in section 5-2, install the two resistors R1 and R2 (Bag 1) on the power supply board.

Resistor	Value
( ) R1	18Ø ohm 1/2 w
( ) R2	(brown, grey, brown)

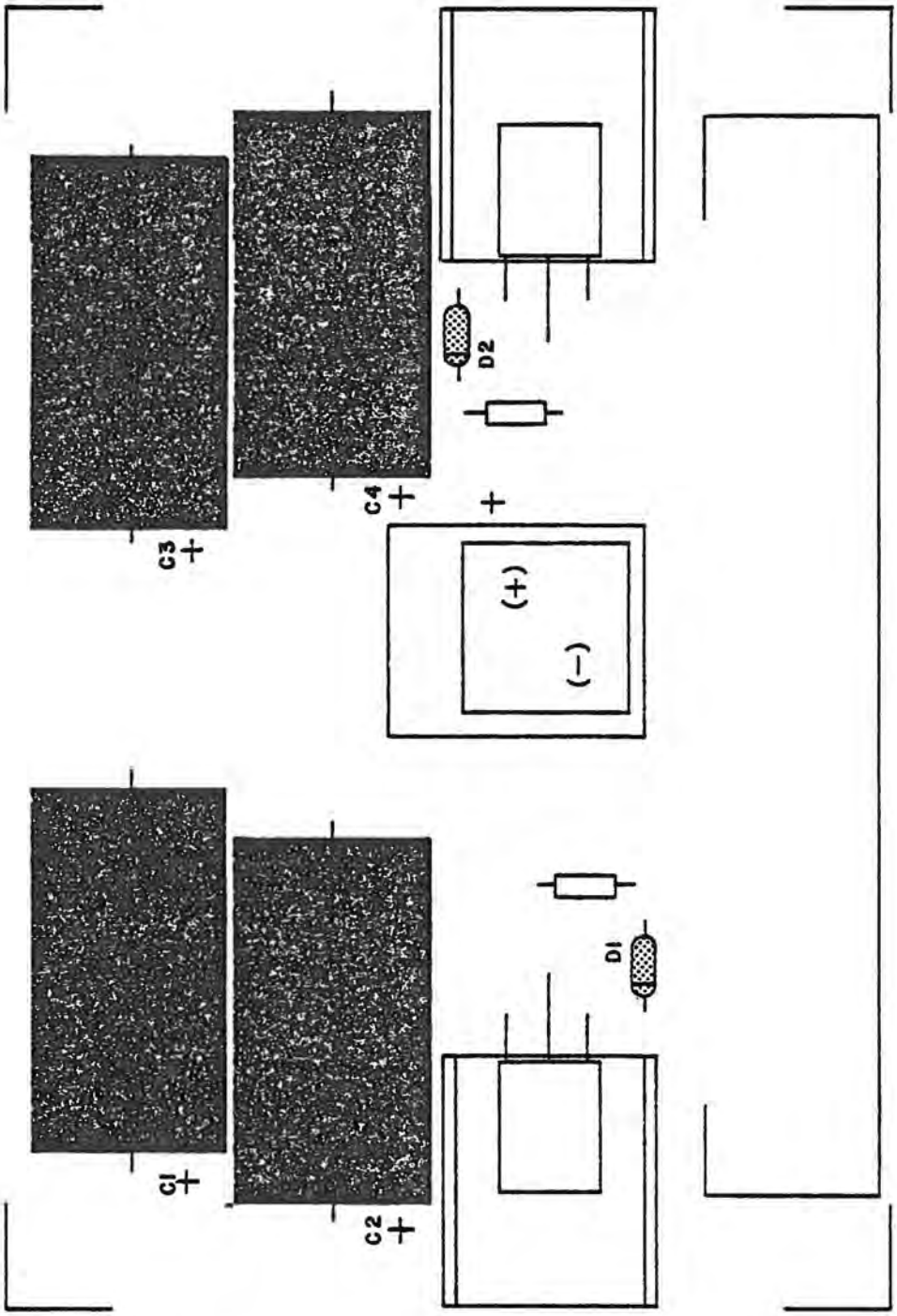


Figure 5-43 Diode Installation

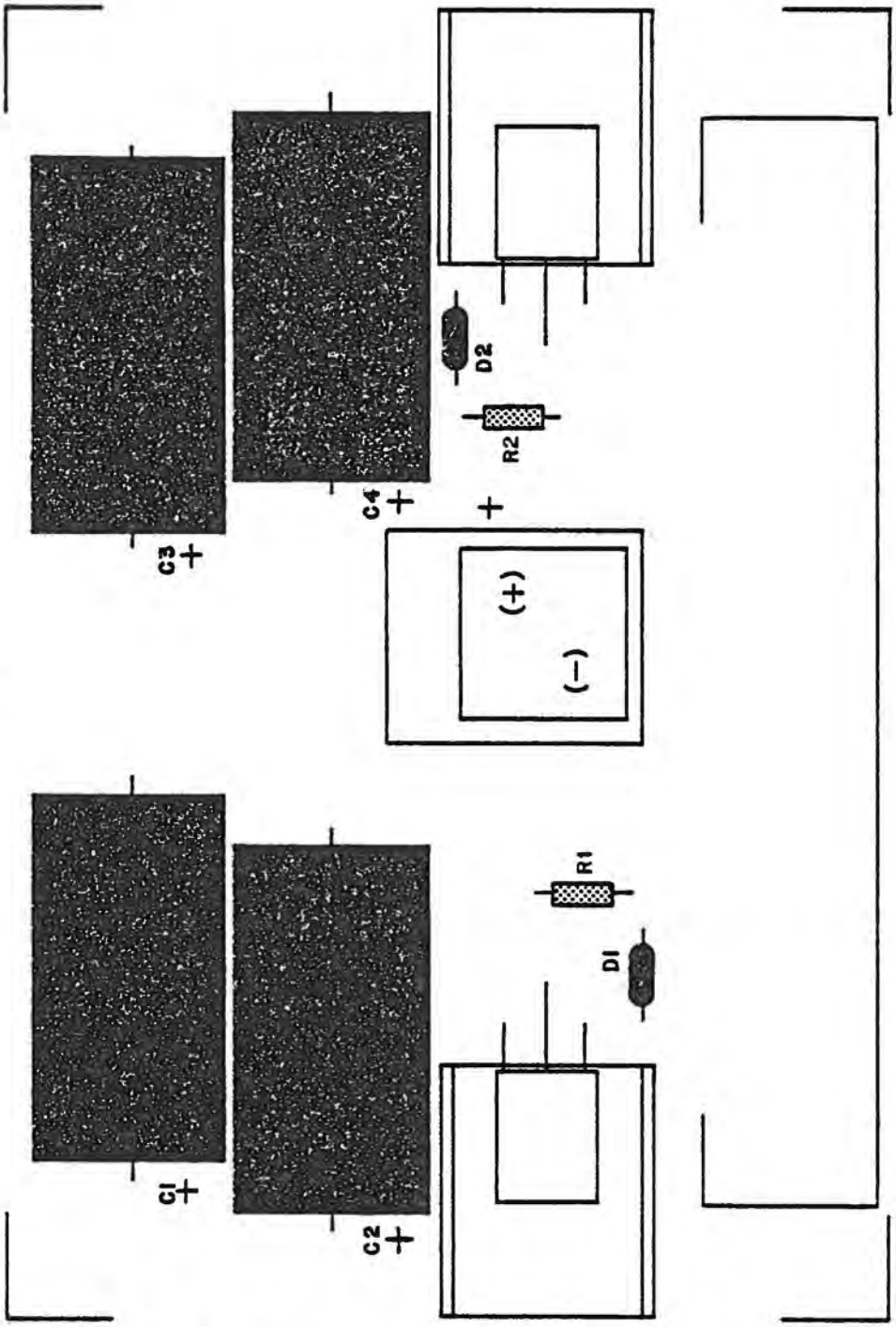


Figure 5-44 Resistor Installation

D. Transistor Installation

There are two transistors Q1 and Q2 (Bag 1) which are installed with heat sinks and mica insulators on the Power Supply Board.

Use heat sink grease when installing this component. Apply the grease to all surfaces where contact occurs, (i.e. bottom of the transistor, mica insulator, heat sink).

NOTE
ACCORDING TO SUPPLY VARIATIONS, SOME KITS MAY CONTAIN EITHER TWO #6-32 x 3/8" NYLON SCREWS OR TWO #4-40 x 3/8" METAL SCREWS TO BE USED FOR SECURING THE TRANSISTORS AND HEAT SINK TO THE BOARD. IF YOUR KIT CONTAINS METAL SCREWS, NYLON SHOULDER WASHERS (ALSO CONTAINED IN THE KIT) MUST BE USED ALONG WITH THE SCREWS.

The following instructions define how to install transistors Q1 and Q2. Install one transistor at a time thereby avoiding the possibility of confusing them.

1. Set transistor Q1 in place on silk-screened side of the board and mark each lead at the point where it will be bent to insert in the designated hole.
2. Using needle-nosed pliers bend each lead at right angles to conform to the proper place on the board.
3. Coat the contact surfaces of components with heat sink grease. Insert screw through transistor and mica insulator.
4. Secure the component in place with a #6 lockwasher and #6-32 nut. Do not tighten down the screw at this point.
5. Turn the board over and solder the transistor leads in place. Clip excess lead lengths.
6. Tighten down screw.

   Repeat this procedure for transistor Q2.

Transistor	Designation
( ) Q1	TIP 145 or TIP 146
( ) Q2	TIP 140 or TIP 141

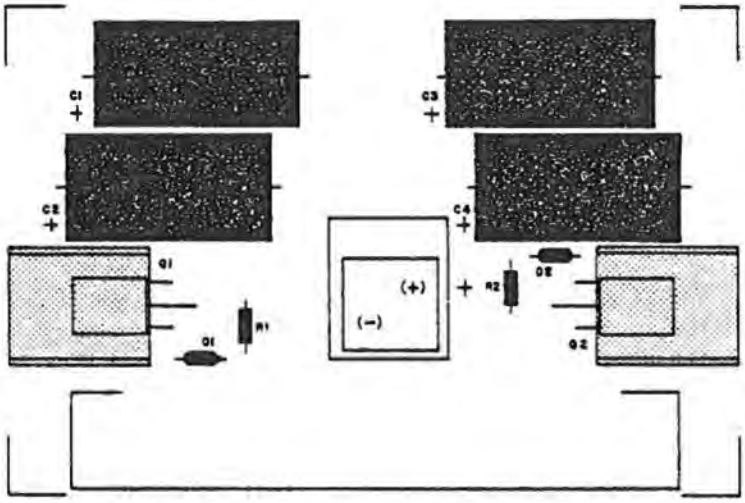


Figure 5-45a Transistor Installation

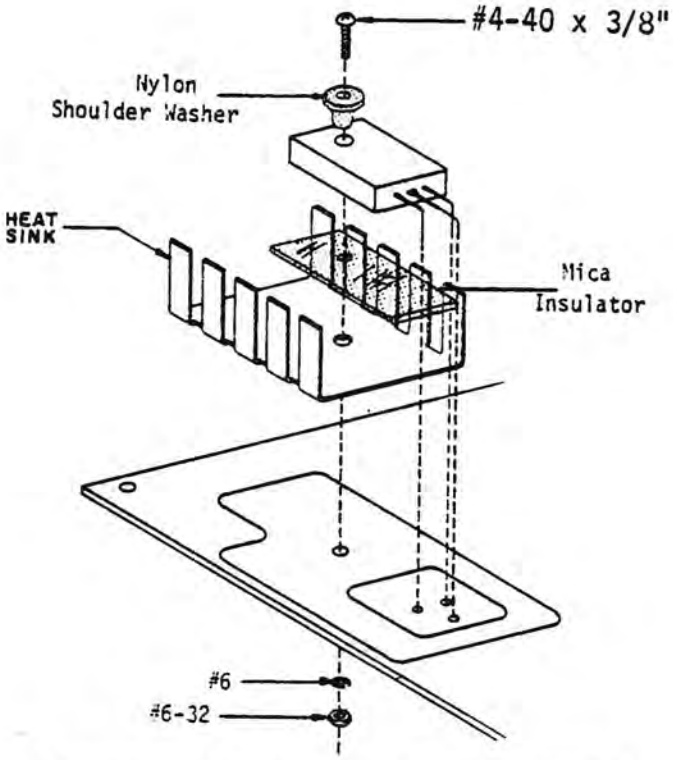


Figure 5-45b Transistor Orientation

E. Bridge Rectifier Installation

The bridge rectifier BR-1 (Bag 1) is installed in the center of the Power Supply Board (Figure 5-46a). Apply heat sink grease to all contact surfaces when installing this component.

NOTE
IT IS ESSENTIAL FOR THE BRIDGE RECTIFIER TO BE CORRECTLY ORIENTED WITH THE POSITIVE (+) LEAD CORRESPONDING TO THE POSITIVE (+) MARKING ON THE BOARD. ACCORDING TO SUPPLY VARIATIONS SOME RECTIFIERS MAY BE MARKED WITH A POSITIVE (PLUS) SIGN OVER THE POSITIVE LEAD: OTHERS MAY BE MARKED WITH A RED DOT OVER THE POSITIVE LEAD.

1. Apply heat sink grease to contact surfaces of heat sink and bridge rectifier.
2. Using the illustration in figure 5-46b as a guide, attach the heat sink to the bridge rectifier with a #6 x 32 x 1/2" screw and a #6 hex nut. The heat sink may have two holes. Insert the screw through the hole furthest from the edge of the heat sink which will result in the heat sink resting within the designated square on the board.

NOTE
ATTACHING THE HEX NUT TO THE SCREW IS A DIFFICULT TASK. BE SURE THE POSITIVE LEAD OF THE RECTIFIER IS PROPERLY ORIENTED, AND THE RECTIFIER IS POSITIONED SQUARELY IN THE HEAT SINK. HOLD THE RECTIFIER WITH ONE HAND WHILE TIGHTENING DOWN THE SCREW TO MAKE SURE THE RECTIFIER DOESN'T SHIFT ON THE HEAT SINK.

3. Insert rectifier leads in the proper holes on the board until the legs of the heat sink touch the board.
4. Holding the heat sink/rectifier unit with one hand, turn the board over and solder the rectifier leads in place. Occasionally solder will not adhere to the bridge rectifier leads. Should this occur gently run a file over hte leads. Reinsert the rectifier and try again.
5. Clip off excess lead lengths.

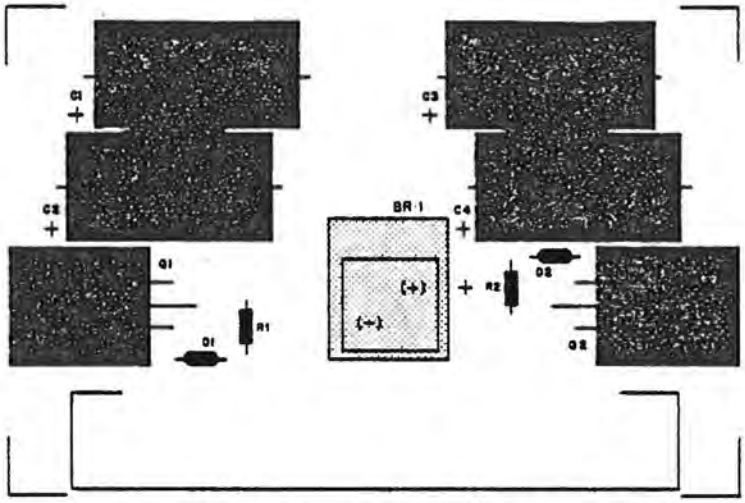


Figure 5-46a Bridge Rectifier Installation

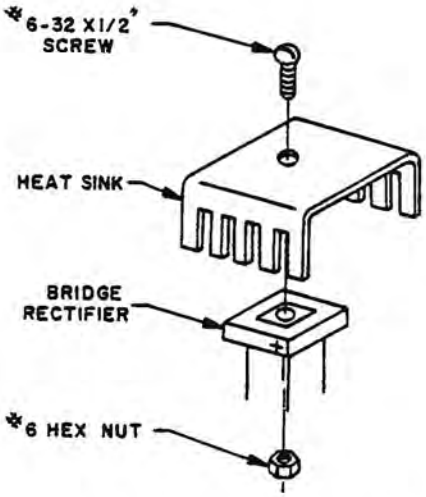


Figure 5-46b Bridge Rectifier Orientation

F. Terminal Block Installation

Bag 2 contains one terminal block, TB1, which should be attached to the board according to the following instructions.

1. Secure the terminal block to the power supply board with four #6-32 x 9/16" or #6-32 x 5/8" screws, four #6 hex nuts and four lockwashers.

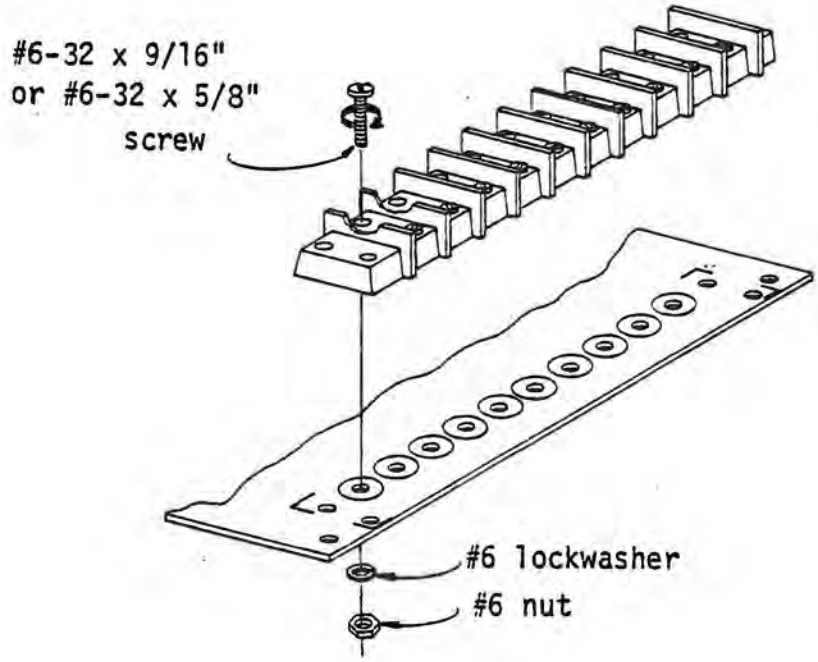


Figure 5-47a

2. Remove five screws from the top row of the terminal block (marked -18, AC, AC, GRD, +18). These are illustrated in Figure 5-47b.

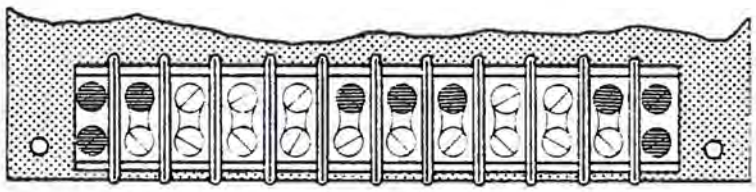


Figure 5-47b

3. Replace these screws with five #6-32 x 9/16" or 6-32 x 5/8" and 5 lockwashers and #6 hex nuts.

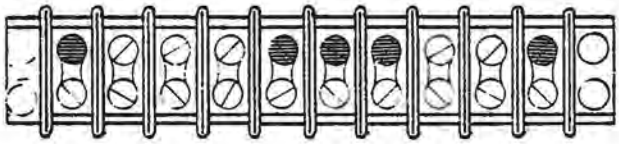


Figure 5-47c

4. Install 1 shorting link between screws of terminals 7 and 8. (Remove screws, insert link, replace screws, threading them through the links.)

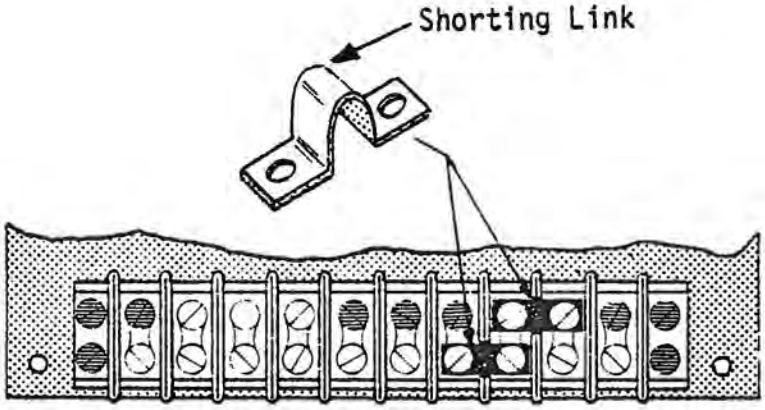


Figure 5-47d

G. Power Supply Board Mounting Description

The power Supply Board is mounted in five holes on the cross member at the back of the mainframe.

Disassemble the mainframe by removing the blue and grey dress panels. This will expose the cross member.

1. Grounding the Power Supply Board

   Before mounting the Power Supply Board, it is necessary to make a ground connection between terminal #9 on the terminal block and the lower right mounting screw on the cross member. The following instructions define this procedure.

   a. Bag 2 contains a small length of metal braid, and two solder lugs. Thread each end of the metal braid through the small holes in the solder lugs. Twist the braid to taper it so it will fit in the holes.

   b. Bend the braid back on itself about 1/8 inch. Solder the braid to the solder lug. It is very important to completely fill the hole in the solder lug with solder.

2) Power Supply Board Mounting Instructions

   Five 3/4" threaded spacers (Bag 5) and ten #6-32 x 3/8" screws (Bag 5) are used in this procedure.

NOTE
OCCASIONALLY THE CROSS MEMBER IS MOUNTED INCORRECTLY ON THE MAINFRAME. CHECK TO BE SURE THE SCREW HOLE PATTERN CORRESPONDS TO THE DIAGRAM (FIGURE 5-48).

1. Insert one screw into each mounting hole on the corners of the power supply board. Insert the forward right corner screw through the solder lug and then into the board.
2. Put a spacer (Bag 5) on each screw, and tighten it down.

3. Set the Power Supply Board on the cross-member. The large capacitors should be positioned toward the rear of the mainframe, Figure 5-48.
4. Attach the board to the cross-member by inserting another screw up thru the spacer. Do not tighten each screw down all the way. After all screws have been loosely inserted, then tighten them down.

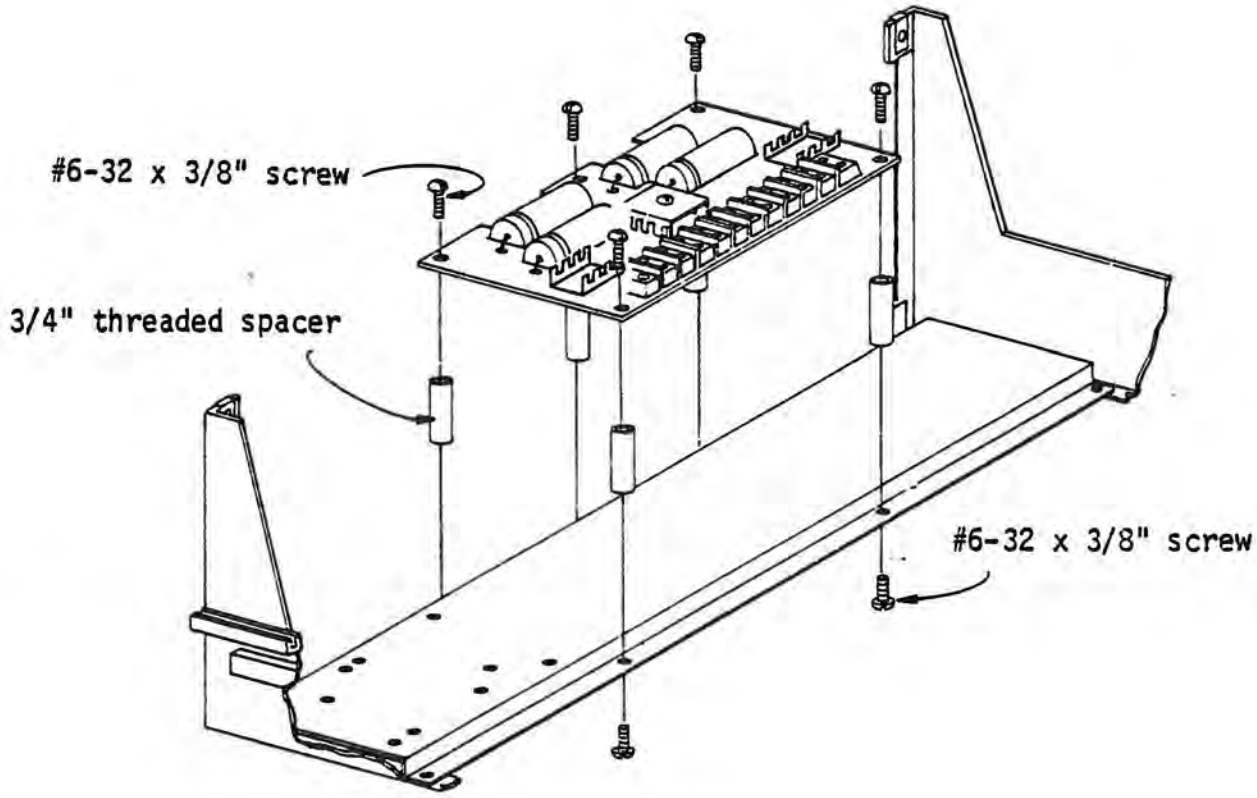


Figure 5-48 Power Supply Board Mounting Orientation

3) 100,000 MF 15V Capacitor Installation

There is one large capacitor which is mounted on the cross-member with a clamp. Illustration 5-49 shows the way in which the capacitor is attached to the cross-member.

1. Insert the large capacitor in the clamp. The feet of the clamp should not be above the bottom of the capacitor. The capacitor must sit flat on the cross-member. Close the clamp with a #6-32 x 5/8" screw, #6 lock washer and a #6 x 32 hex nut.
2. Place the clamp and capacitor on the cross-member, aligning the mounting holes. (It is at this point you will discover if the cross-member is installed incorrectly as the capacitor must be mounted on the left of the Power Supply Board.)
3. Secure the clamp and capacitor to the cross-member using three (3) #6-32 x 3/8" screws and three (3) #6-32 nuts.

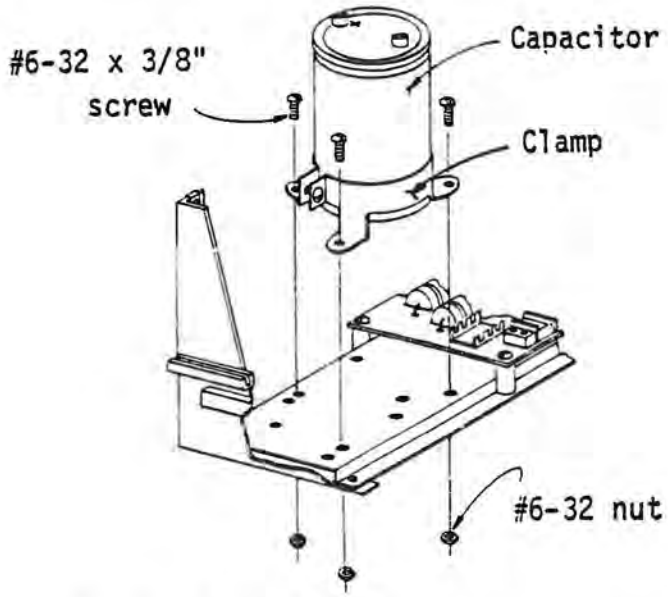


Figure 5-49 Capacitor Installation

5-7. Back Panel Assembly Description and Procedural Instructions

> CAUTION
>
> BE EXTRA CAREFUL WHEN WIRING THE VARIOUS COMPONENTS. INCORRECT WIRING CAN HAVE DISASTROUS RESULTS.

A. Terminal Ends

There are five different sizes of terminal ends which are used when wiring the back panel. Figure 5-50 defines the various terminal end types, gauge size of wire to be used and screw size which are used for each terminal end.

Refer to Figure 5-50 each time a terminal end size is specified in the assembly instructions.

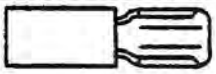
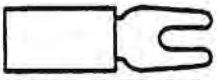
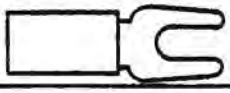
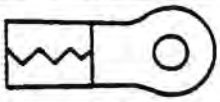
SIZE	BAG #	TERMINAL END	WIRE GAUGE	SCREW SIZE
A	2		12-10	slip on
B	2		22-18	#6 screw
C	2		12-10	#6 screw
D	2		12-10	#10 screw
E	2		12-10	#10 screw

Figure 5-50 Terminal End Sizes

B. Wire Preparation Instructions

Prepare all wire according to the following instructions before using it in any assembly procedure.

1. Cut the length of wire specified in the assembly instructions.
2. Strip 1/8" to 1/4" of insulation off the ends.
3. Apply a thin coating of solder over the entire exposed wire end.

C. Attachment of Terminal Ends to Wires

Most of the wire connections in the Back Panel Assembly Instructions call for attaching a terminal end to a wire and mounting it to the proper terminal. This procedure is detailed below:

1) Terminal Sizes A-D

a. Insert the exposed (stripped) portion of wire which has been prepared according to the instructions in the previous subsection into the terminal end (size specified in assembly instructions).

b. Heat the wire and terminal end with the soldering iron. Apply solder to the heated wire, allowing solder to flow until there is a solid connection with no open spaces.

If the insulator on the end of the terminal loosens during the soldering process, be sure to push it back into place when the soldering process is complete.

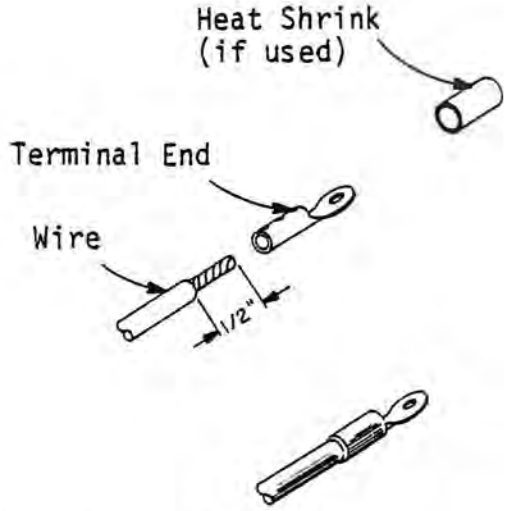


Figure 5-51 Terminal End Attachment

NOTE

BE SURE TO HOLD SIZE A TERMINAL ENDS IN A VERTICAL POSITION WITH THE WIRE DOWN WHILE SOLDERING TO PREVENT SOLDER FROM FLOWING ONTO THE SLIP-ON TABS.

2. Terminals with End Size E

   Size E terminal ends do not have insulators, and therefore must be insulated with heat shrink tubing. The procedure for attaching E size terminals ends varies slightly, as follows:

   1. Set the E Size terminal end on the work surface and heat it with a soldering iron until it is hot enough to allow solder to flow.
   2. Insert the exposed portion of a wire you have prepared into the terminal end and apply solder until there is a solid connection.
   3. After the wire has been soldered in place and the joint has cooled, cut a 1-inch piece of heat shrink tubing and place it over the terminal end. Use a heat gun, if available, or a match to shrink the tubing.

CAUTION

TERMINAL ENDS BECOME EXTREMELY HOT DURING SOLDERING. ALLOW FIVE MINUTES COOLING TIME AFTER SOLDERING BEFORE TOUCHING THE TERMINAL ENDS.

D. Connector Pins and Connector Sockets

Some of the wire connections in the back panel assembly instructions call for connector pins and connector sockets housed in a plastic plug. The general procedure for preparing these plugs is detailed below:

1. Insert the exposed portion of a wire that you have prepared into a connector pin or connector socket as shown in Figure 5-52a.
2. Crimp the lower portion of the pin or socket around the wire insulation. Solder the center portion of the pin or socket to the exposed portion of the wire.

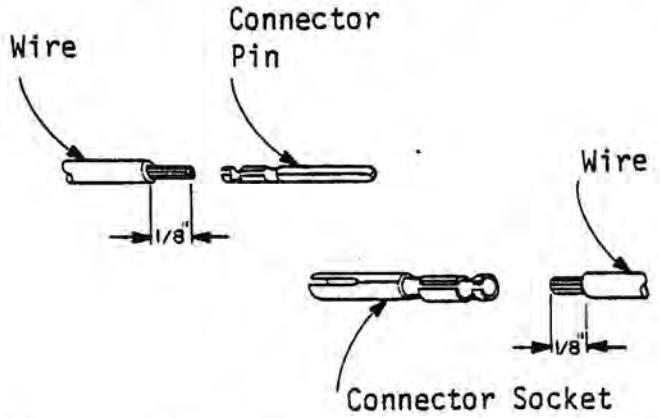


Figure 5-52a Connector Pin and Connector Socket Wire Installation

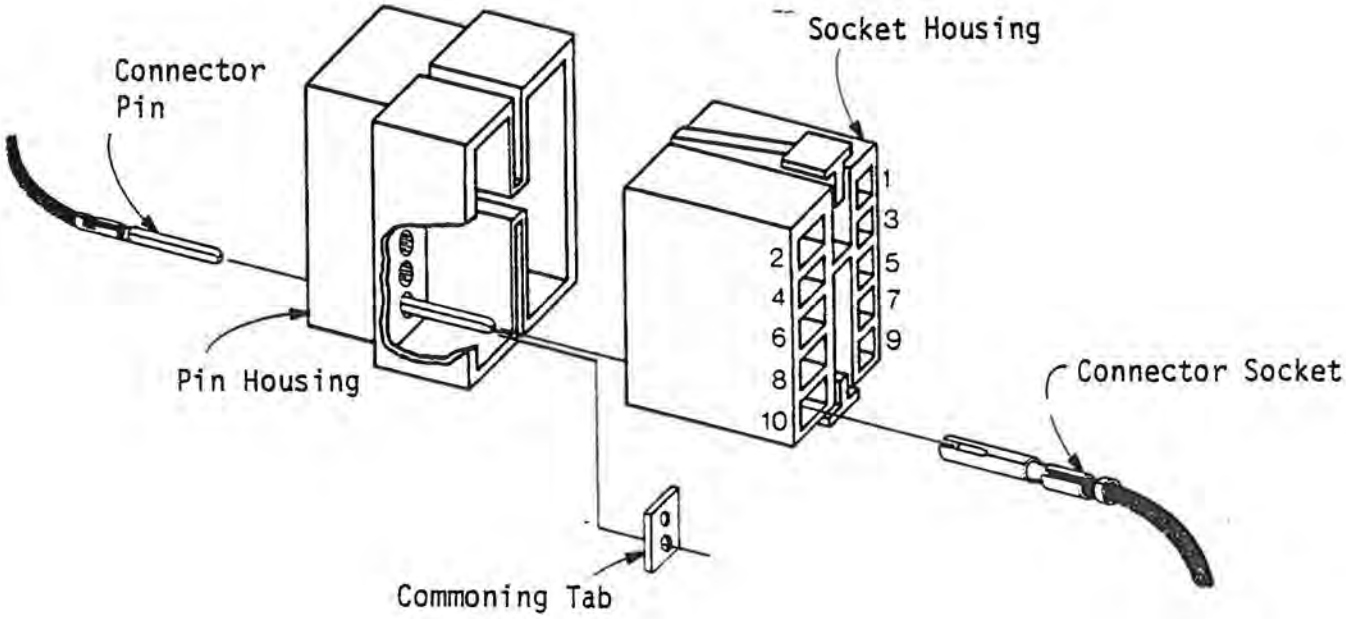


Figure 5-52b Pin and Socket Housing Assembly

3. Insert the pins and sockets into their respective housings as shown in Figure 5-52b.
4. Commoning tabs are put into the pin housing over pins that must be shorted together. Push the commoning tabs all the way to the base of the pin housing, using the tip of a small screwdriver

CAUTION
MAKE SURE THE COMMONING TABS DO NOT COME IN CONTACT WITH EACH OTHER.

E. Capacitor Wiring

It is necessary to wire the large capacitor before beginning any other back panel assembly procedures. Follow the wiring instructions very carefully. Wiring the capacitor improperly can result in destruction of the capacitor.

1. Cut two lengths of 10 gauge (white) wire nine inches long.
2. Attach a size C terminal end to one end of each wire and a size D terminal to the other end of each wire.
3. Attach the size D terminal ends to the capacitor with the screws provided.

CAUTION
STEPS 4 AND 5 ARE CRITICAL; FOLLOW THE INSTRUCTIONS CAREFULLY.

4. Attach the wire leading from the positive (+) pole of the capacitor to terminal #3 (8V) on the power supply board. (See wiring diagram Figure 5-63)
5. Attach the wire leading from the negative pole on the capacitor to terminal #8 of the terminal block on the power supply board. (See wiring diagram.)

WARNING
DO NOT REVERSE THIS WIRING!! (I.E.: NEGATIVE TO #3, AND POSITIVE TO GROUND.)

F. Bridge Rectifier Installation

The following instructions describe the installation of the mounting and wiring of the back panel bridge rectifier. Figure 5-55 further defines the installation. Do not use excessive force when screwing down the bridge rectifier.

1. Attach the bridge rectifier to the back panel, over the gold colored square, with a #6-32 x 3/4 inch screw and a #6-32 nut, flat washer and lockerwasher. The negative sign (-) should be oriented in the upper right corner.
2. Cut two 4-inch and two 19-inch lengths of 10 gauge (white) wire.
3. Attach an A size terminal to one end of each wire. Attach a D size terminal to the other end of each wire.
4. Attach the A size terminal ends to the bridge rectifier terminals. Using masking tape, label the 4-inch wires 13 and 15, and the 19-inch wires 14 and 12. (See Wiring Diagram, Figure 5-63.) With the front of the computer facing you, attach wire 12 to terminal #9 of the lower terminal block. Attach wire 13 to terminal #1 of the upper terminal block. Attach wire 14 to terminal #4 of the lower terminal block. Attach wire 15 to terminal #2 of the upper terminal block.

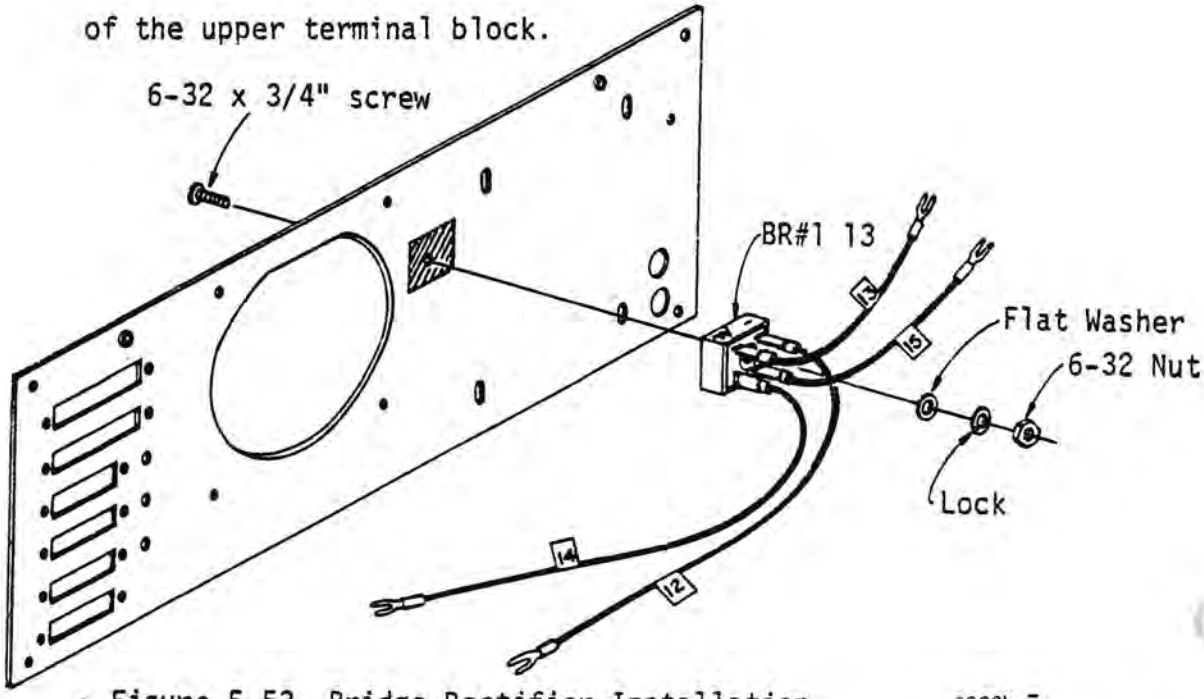


Figure 5-53 Bridge Rectifier Installation

G. Fan Installation

The kit includes one cooling fan and screen with mounting screws and snapon nuts. The fan is to be wired and then attached to the back panel. Be sure to label the wires with masking tape.

According to supply variations some kits may contain a fan with a plug. The wires will be inserted in the plug rather than being soldered to terminals.

1. Cut two 20-inch lengths of 22 gauge (black) wire. Strip and prepare the ends. Label (with masking tape) each wire 'fan'.
2. Slip a two inch length of heat shrink tube over each wire. Insert a prepared wire end into each prong hole of the plug connection; in the following manner bend the stripped end into a right angle, insert the wire into the prong hole and bend the wire back on itself. Solder into place. Slip shrink tube down and heat to shrink.
3. Mount the fan and screen (Figure 5-54) to the back panel. The air flow arrow should point toward what will be the interior of the computer. The plug should be in the lower right corner. Use 4 #6-32 x 5/8" screw and 4 #6 snap-on nuts.
4. Attach a connector socket to the end of each wire. Solder into place.

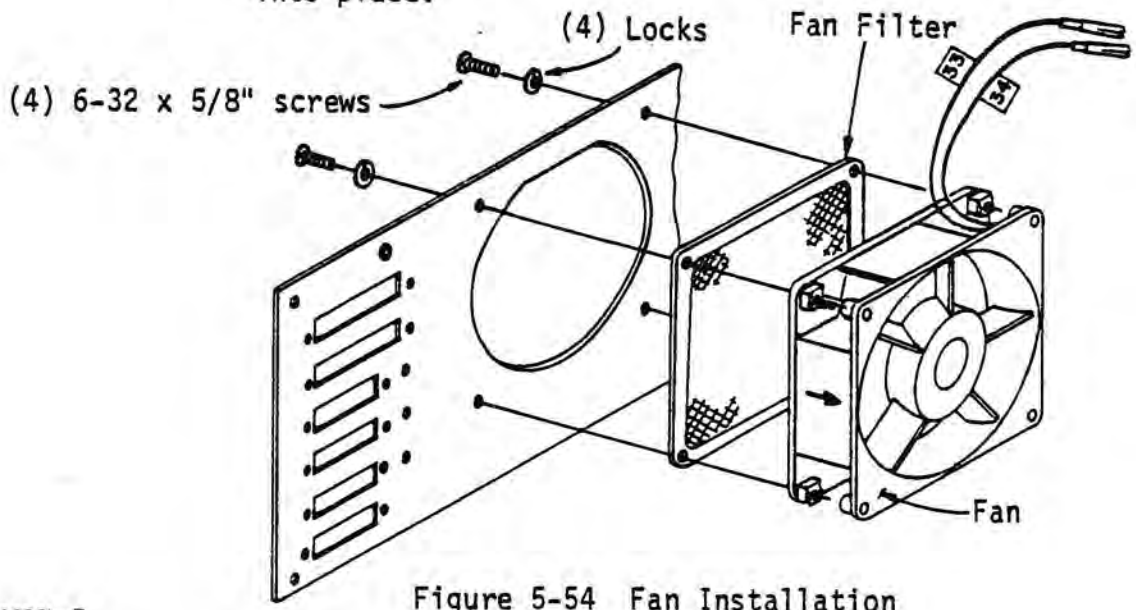


Figure 5-54 Fan Installation

H. Fuse Holder Installation

The fuse holder (Bag 2) is inserted in the top hole of the pair of large round holes in the lower right corner of the back panel. The fuse will be inserted in the fuse holder in one of the final assembly procedures.

NOTE
WHEN INSTALLING THE FUSE HOLDER, TAKE CARE NOT TO TIGHTEN THE FUSE HOLDER DOWN TOO TIGHTLY AGAINST THE NUT SINCE IT MAY CAUSE THE PLASTIC HOLDER TO CRACK.

1. Secure the fuse holder (as illustrated in Figure 5-55) with the rubber washer and nut.

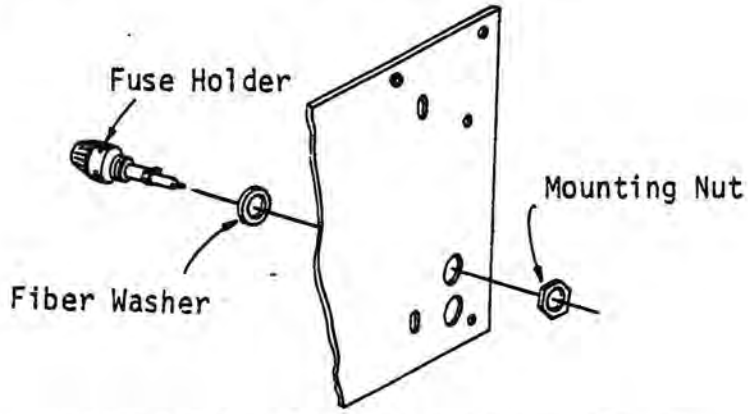


Figure 5-55 Fuse Holder Installation

2. Cut a length of 18 gauge (black) wire 40 inches long, and prepare the ends.

NOTE
HEAT SHRINK TUBING CAN BE USED IN THIS INSTALLATION PROCEDURE IF THE USER HAS ACCESS TO A HEAT GUN. SLIP THE TUBING OVER THE WIRE BEFORE CONTINUING.

3. Attach the wire to the terminal on the side of the fuse holder and solder into place.
4. Attach a connector pin to the end of the 40-inch length of wire and solder into place. Label this wire #39.

H. AC Power Cord Installation

1. Carefully strip seven inches of casing off the end of the AC power cord to expose the three wires inside.
2. Cut the exposed black wire to a length of two inches, and the exposed green wire to a length of five inches. Do not cut the white wire.
3. Bag 2 contains a strain relief. Slip it over the cord, and push it down until it is situated about 1/2 inch from the stripped end of the casing (below the exposed wires) (see Figure 5-56).
4. Snap the strain relief in place in the hole on the rear panel just above the fuse holder (see Figure 5-56).
5. Slip shrink tube over the black wire. Solder the black wire to the terminal on the fuse holder. Insulate with shrink tube.
6. Attach a solder lug to the green wire.
7. Attach a connector socket to the white wire.

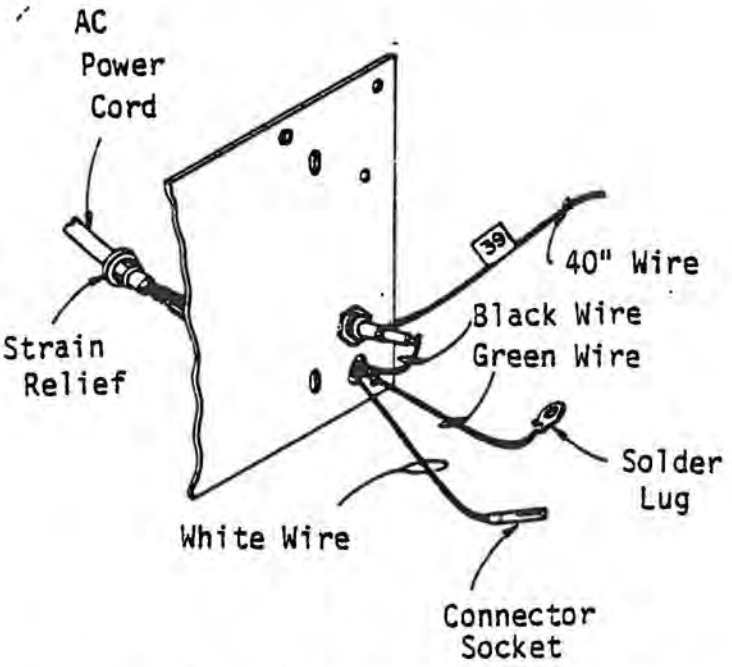


Figure 5-56 AC Power Cord Installation

5-8. Transformer Sub Assembly

The installation of the transformer is divided into three processes; secondary wiring, primary wiring, and transformer mounting. The attachment of terminals, pins, etc. will follow the previously defined instructions.

> NOTE
>
> The following wiring instructions are for 110 volts AC only. To modify the Altair 8800b Turnkey to operate with 230 volts, see Appendix B of this document. Any modifications of this nature must be done at this point in the assembly procedure.

A. Transformer Secondary Wiring Instructions

1. Orient the transformer with the secondary side (4 large black wires*) facing you as illustrated in Figure 5-57. Remove the two (2) top bolts and nuts, and use them to mount the pair of "L" brackets (Bag 2). See Figure 5-57.

   > *According to supply variations, some transformers may have four (4) large white or other color wires.

2. Slip shrink tubing over the four black wires. After preparing the wire ends, attach an "E" size terminal to each one. Label the wires 16, 17, 18 and 19. The top wire of the group of black wires is number 16.

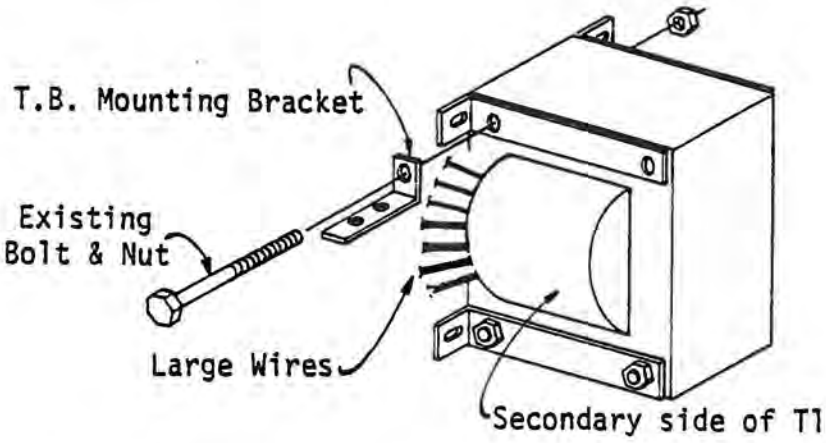


Figure 5-57 Transformer Mounting Bracket

3. Prepare the ends of the remaining three secondary wires and attach a "B" size terminal to each one. Label the yellow wires 20 and 21. Label the yellow/green wire 22 (Figure 5-58).
4. Bend each "E" size terminal end at a right angle as shown in Figure 5-58. Bend the terminal at right angles away from the wire hole. (If the terminal is bent over the wire hole, it will not fit on the terminal block.) Attach the four terminals to the terminal block (TB2) as illustrated in Figure 5-58. Use the screws provided on the terminal block.
5. Mount the terminal block to the "L" brackets on the transformer with four (4) #6-32 x 3/4 inch screws, four (4) #6-32 nuts and four (4) #6 lockwashers (see Figure 5-59).

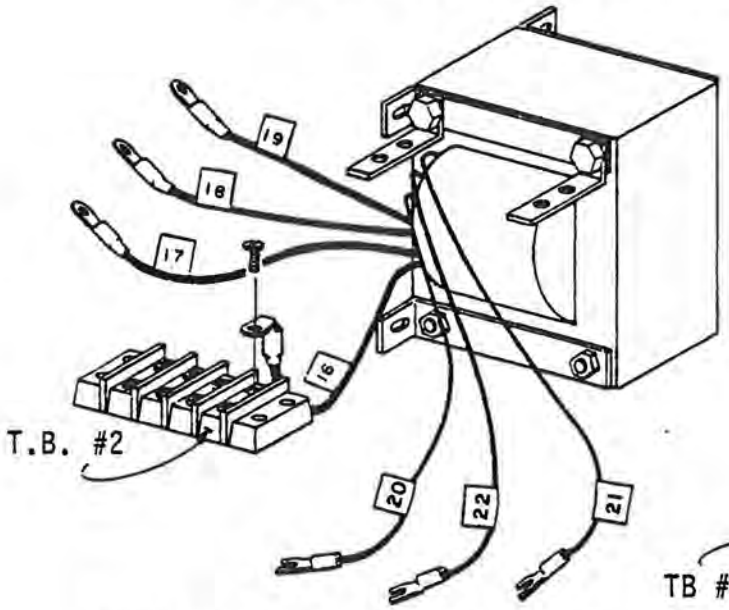


Figure 5-58 Wire and Terminal Attachment to Terminal Block

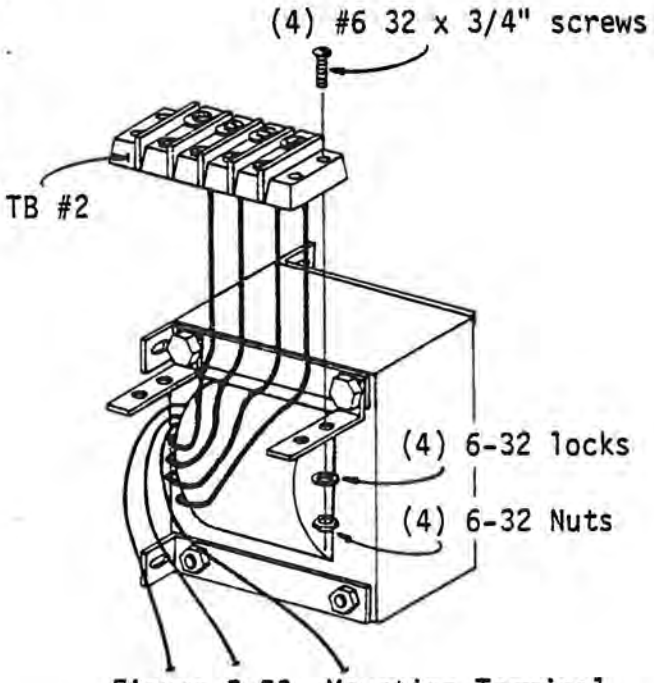


Figure 5-59 Mounting Terminal Block on Transformer

B. Transformer Primary Wiring

A two-part 10-pin plug (Bag 4) will be used to connect the primary wiring with the 110 volt source.

(1) Pin Housing

a. Attach a connector pin to the end of each primary wire of the transformer.

b. Insert the pins into the proper location in the pin housing (Bag 4) as specified by the table from the next page. Make sure each pin "clicks" into place. If a pin is inserted properly, it should not come out when pulled. Occasionally it may be necessary to push a pin into place with needle-nose pliers.

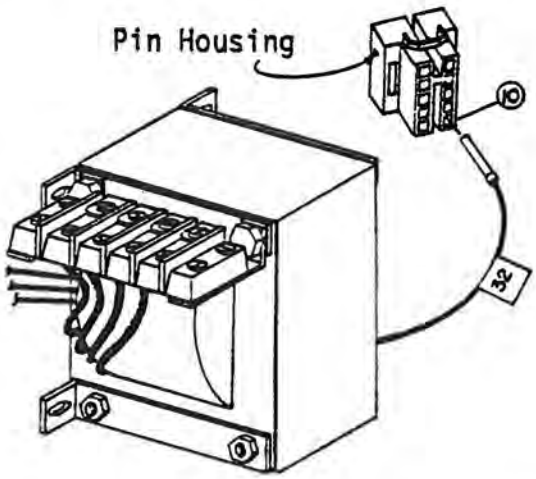


Figure 5-60 Pin Insertion Into Pin Housing

NOTE
FIGURE 5-52b DEFINES THE PIN MARKING. THE NUMBERING CAN ALSO BE FOUND PRINTED IN SMALL NUMERALS INSIDE THE PIN HOUSING.

Wiring Diagram Designation	Transformer Wire Color (Primary Side)	P4 Pin Housing Slot Number
25	Red/Black	3
26	Blue/Black	9
27	Green/Black	4
28	White/Black	10
29	Red	1
30	Blue	7
31	Green	2
32	Black	8

c. Place a two-circuit commoning tab (following instructions in Section 5-1) over the following pairs of pins:

2 and 4 1 and 3
7 and 9 8 and 10

NOTE

THE COMMONING TABS SHOULD BE INSERTED WITH THE SHADED OR DARK SIDE UP.

(2) Socket Housing

a. Insert the black fan wires into slots 7 and 8 of the socket housing. (This is not to be confused with the pin housing of the previous step.

b. Cut a 40-inch length of 22-18 (black) gauge wire, and attach a connector socket to each end. Label this wire 37.

c. Insert one connector socket of wire 37 into slot 9 of the socket housing (Figure 5-61).

d. Insert the connector socket on the end of the white AC power cord wire into slot 10 of the socket housing. Figure 5-52b describes the "mating" process of a pin housing and socket housing.

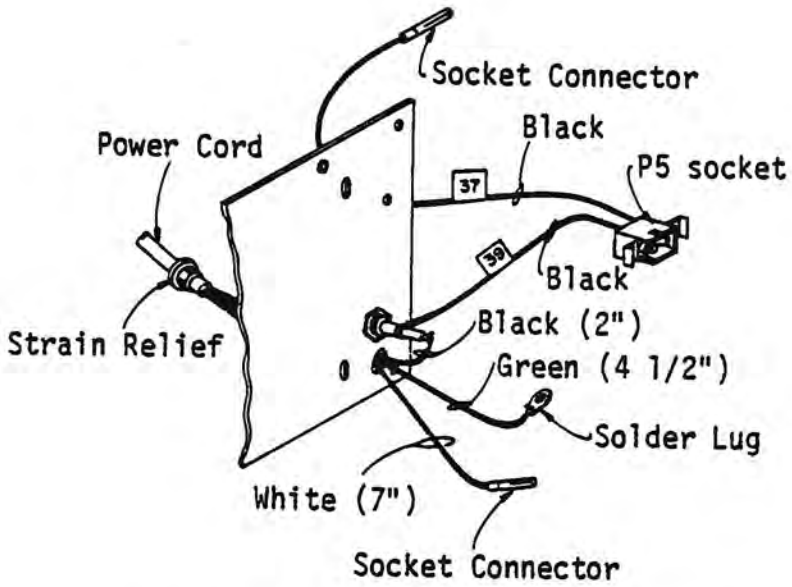


Figure 5-61 Connector Socket to Socket Housing

C. Mounting the Transformer to the Back Panel

1. Mount the transformer on the back panel as shown in Figure 5-62. Secure to the back panel with the four (4) #10-32 x 1/2 inch screws, four (4) nuts, four (4) flat washers and four (4) #10 lockwashers.

HINT
To secure the transformer to the back panel, turn the transformer on its side and place the back panel on top of it. Insert the screws and nuts as illustrated in Figure 5-62. Do not tighten the screws; positioning adjustments will be necessary later.

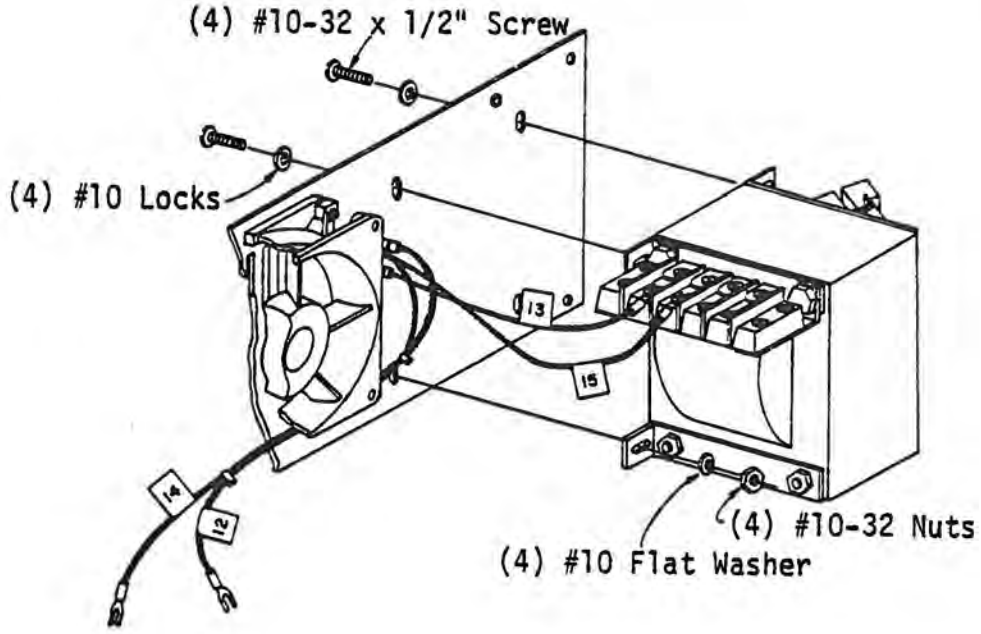


Figure 5-62 Mounting Transformer to Back Panel

2. Attach wires 13 and 15 from the bridge rectifier to terminals 1 and 2 of the terminal block (TB2) as shown in the wiring diagram (Figure 5-63).

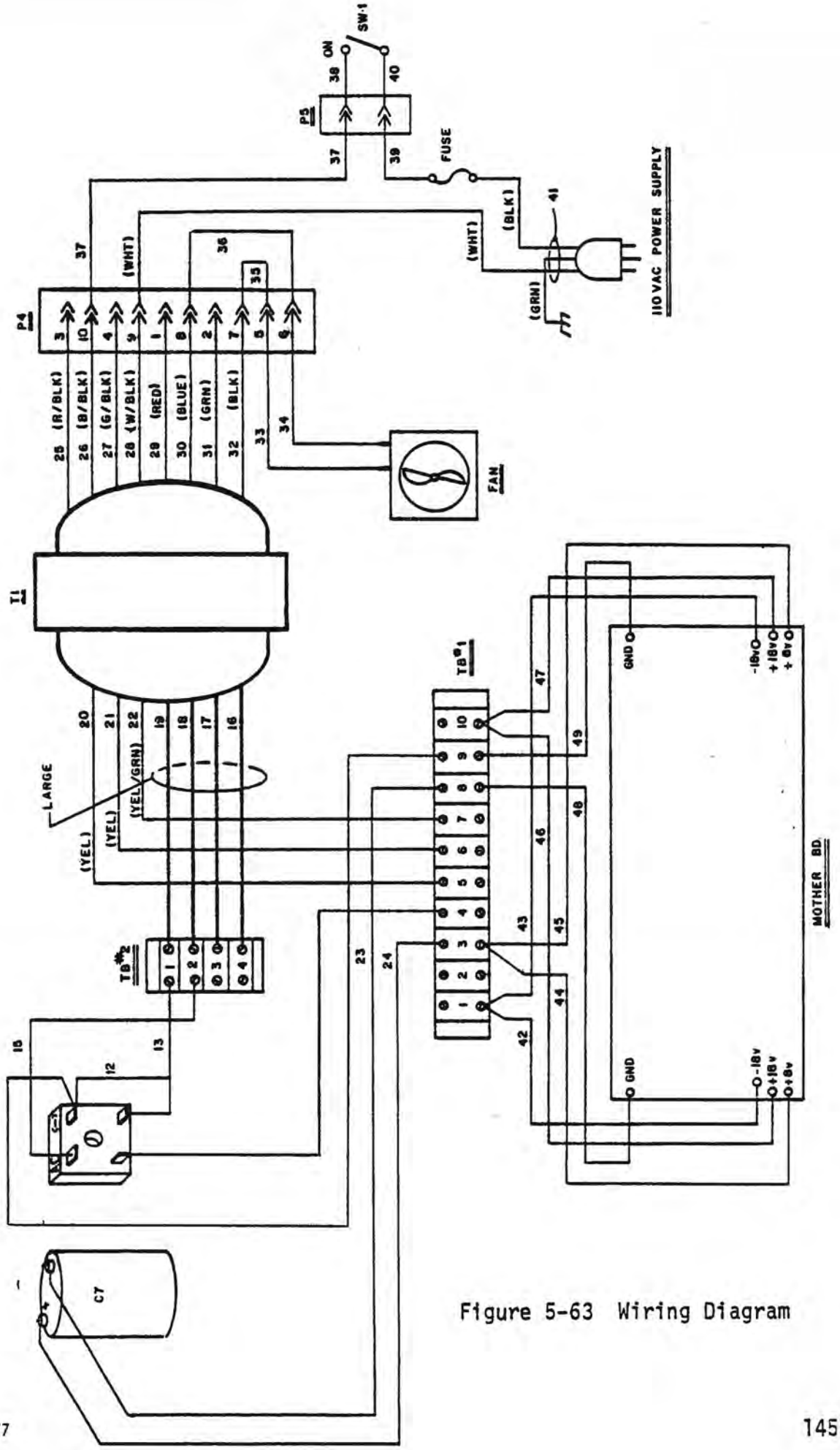


Figure 5-63 Wiring Diagram

D. Mounting the Back Panel to the Mainframe

1. Mount the back panel on the mainframe as illustrated in Figure 5-64 using the original mainframe mounting screws.

SPECIAL NOTE

When positioning the back panel for mounting, take care that:

a. no wires are caught under the transformer

b. the two 19-inch white bridge rectifier wires, the 40-inch fuse wire (#39), and the 40-inch connector plug wire (#37) run under the fan

c. the wires from the fan go under the fan and under (toward the back panel) the transformer. Be sure the transformer does not rest on the fan wires.

d. the electrolytic capacitors on the power supply board are not dislodged

e. the transformer rests solidly on the cross member

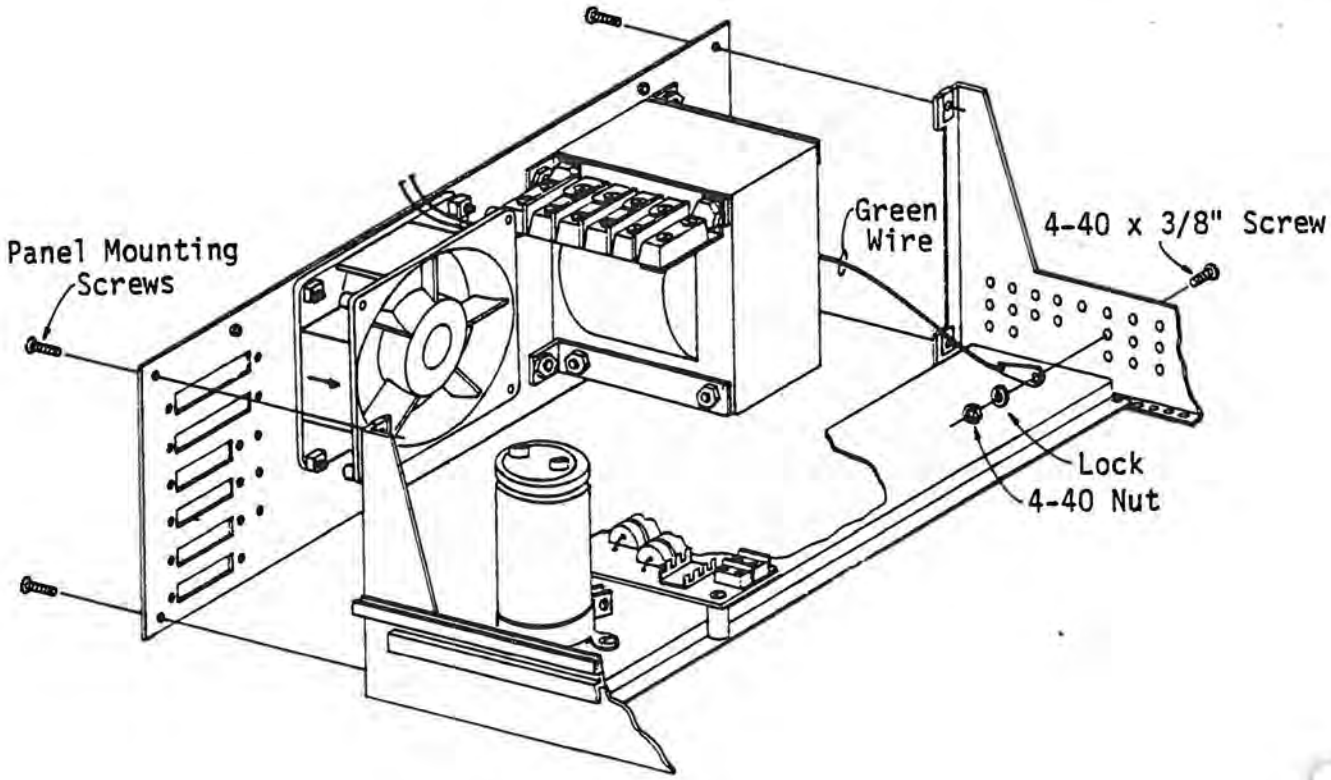


Figure 5-64 Mounting Back Panel to Mainframe

HINT

Mounting the back panel requires care due to the concentrated weight and positioning of the transformer. Should the back panel and mainframe screw holes seem misaligned, loosen the transformer screws nearly all the way. This will allow some "play" in the back panel which should make the mounting easier. Inserting the screws in alternate holes (i.e. 1 top left, 2 lower right) will also make screw hole alignment easier.

2. Attach the solder lug on the green AC ground wire to any hole on the side of the mainframe using a #6-32 x 1/4" screw, a #6-32 nut, and a #6 lockwasher.
3. Connect the three secondary wires from the transformer to Terminal Block #1 as follows:

   Wire #20 (yellow) to slot #5, TB #1
   Wire #21 (yellow) to slot #6, TB #1
   Wire #22 (yellow/green) to slot #7, TB #1

5-9. 18 Slot Motherboard Assembly

A. 18 Slot Motherboard Description

The Altair 8800b Turnkey kit contains an 18 slot motherboard which has two (2) edge connectors. Additional edge connectors can be added to accommodate more boards and expand capabilities.

The cable clamps, mainframe cross rails, edge connectors and card guides are installed on the motherboard. The motherboard is then wired and mounted on the mainframe.

B. Edge Connector Installation Instructions

Set the motherboard in front of you, foil side down with the single row of small holes closest to you (see Figure 5-65).

1. Remove the 100 pin edge connectors from bag 7 (CPU board assembly) and bag 4 (Turnkey Module board assembly). Align the pins by gently running a screwdriver along them.
2. Insert the edge connectors in the location specified in Figure 5-65. These are the two rightmost locations on the board.

NOTE
BE VERY CAREFUL WHEN INSERTING THE EDGE CONNECTORS SO THAT ALL 1ØØ PINS PASS THROUGH THE BOARD AND NO PINS ARE BENT BENEATH THE EDGE CONNECTOR.

3. Solder each pin in place. This is a rather tedious procedure, but one that must be done very carefully. Visually inspect the board to make sure there are no solder bridges.

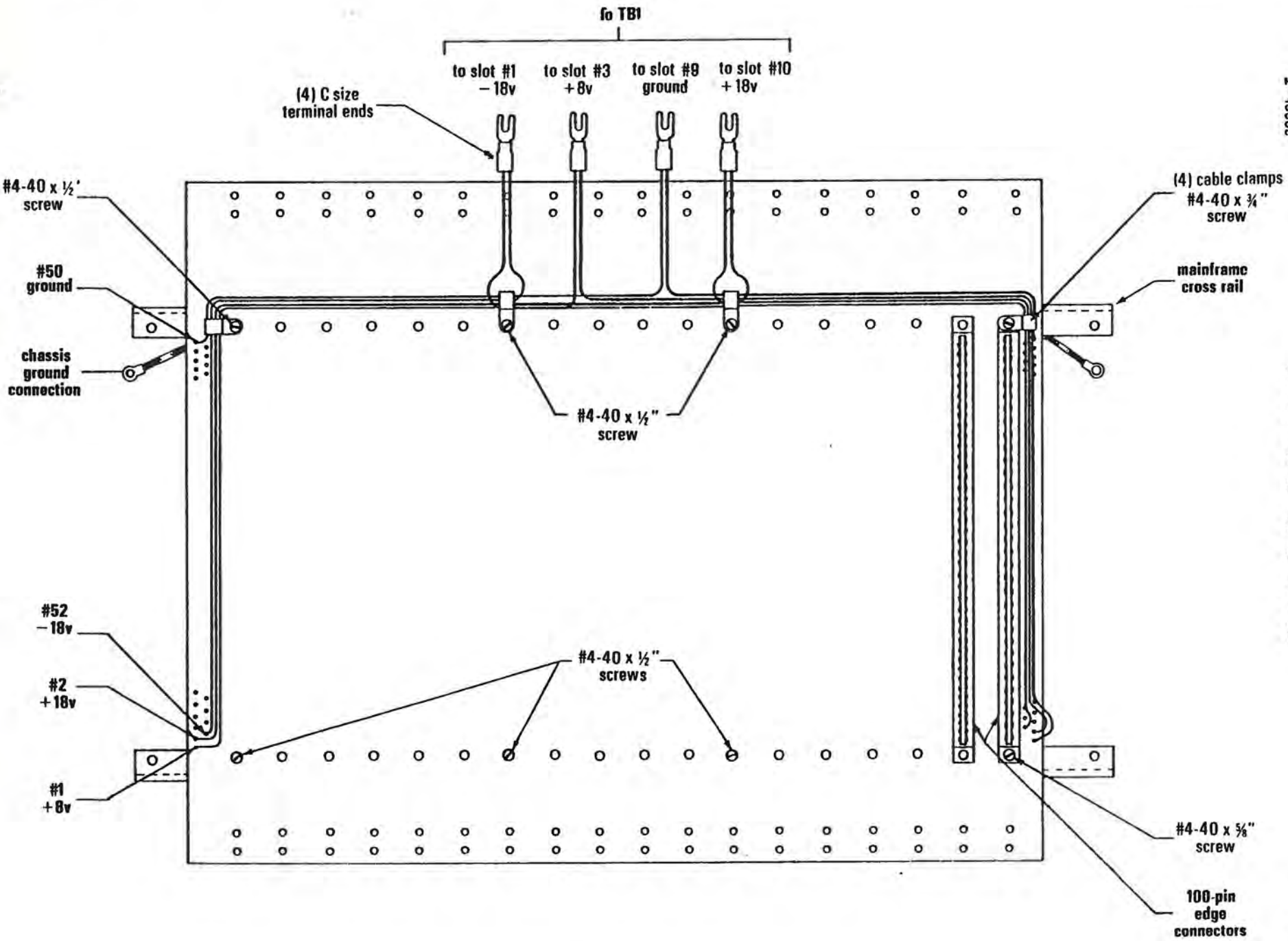


Figure 5-65 Motherboard

C. Bus Wire Connection

Study Figure 5-65 before beginning the bus wiring procedures. Note that the two outside rows of holes on both sides will have four wire connections. These are the +18v, -18v, +8v and ground lines to the power supply from the bus. These connections are made by inserting a prepared wire end from the top of the motherboard and soldering it to the foil (bottom side of the board). Holes #1 and #50 are marked on each side of the foil side of the motherboard. The next row of holes begins with #51.

Complete the wire connections according to the following instructions.

1. Cut 22-18 gauge (black) wire into the following lengths:
   Six 20-inch lengths
   Two 14-inch lengths
2. Install one 20-inch wire in hole #1 (+8) as indicated in Figure 5-65.
3. Install one 20-inch wire into hole #2 (+18v) as indicated in Figure 5-65.
4. Install one 20-inch wire into hole #52 (-18v) as indicated in Figure 5-65.
5. Install one 14-inch wire into hole #50 (ground) as specified in Figure 5-65.
6. Repeat this wiring on the opposite side (top) of the motherboard and solder all wire connections into place.

D. Motherboard/Cross Rail Mounting Instructions

1. Attach four cable clamps to four #4-40 x 3/4" screws. Run the bus wires through these cable clamps. Mount the motherboard to the cross rails using these screws.
2. Insert the remaining four screws in the areas specified in Figure 5-65.
3. Match the four pairs of bus wires specified in figure 5-65. Attach a size C terminal end to each pair. Carefully check the wires to be sure they are correctly paired as indicated in the following table.

   -18v with -18v
   + 8v with + 8v
   ground with ground
   +18v with +18v
4. Mount the card guides on both sides of the connectors as illustrated in Figure 5-66.

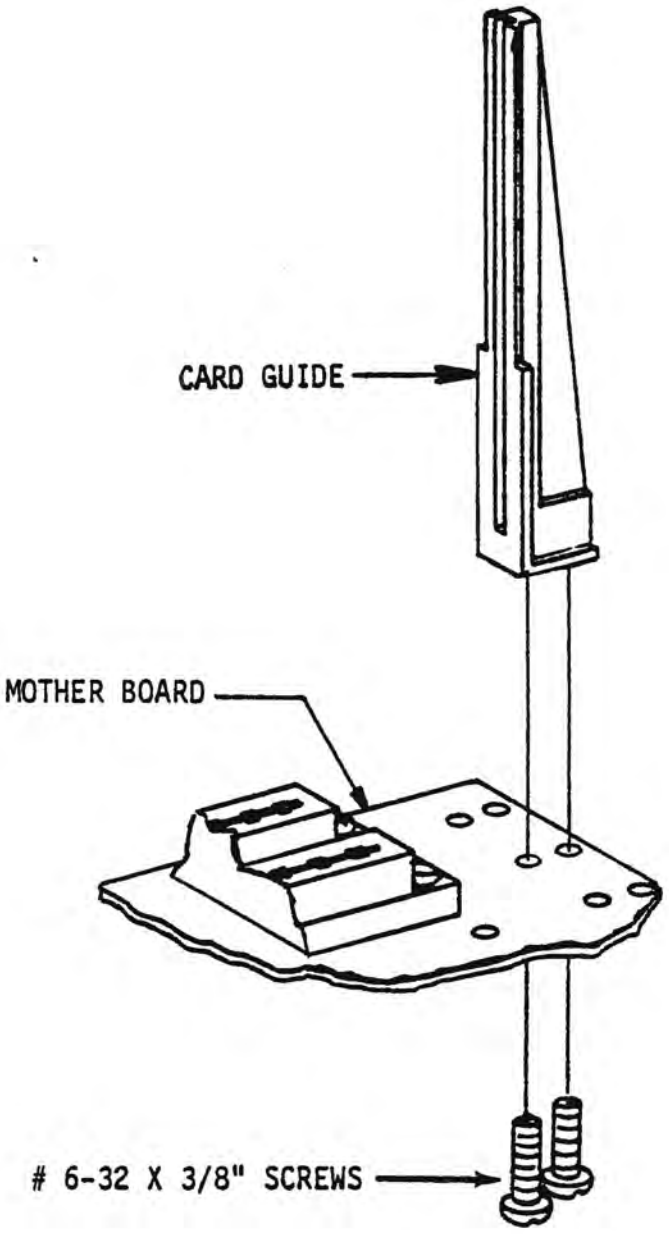


Figure 5-66 Card Guide Mounting

E. Chassis Ground Connection

To insure a good ground connection between the motherboard and the chassis, two ground cables are run from the ground land on the foil (bottom) side of the motherboard to the side rails of the mainframe. Figure 5-67 specifies the area in which the ground connections are made.

NOTE
IT IS NECESSARY TO SCRAPE THE SOLDER MASK OFF THE GROUND LAND BEFORE ATTEMPTING TO SOLDER THE GROUND CABLE TO THE LAND. USE A SCREWDRIVER TO CAREFULLY SCRAPE THE GREEN COATING TO EXPOSE THE FOIL.

1. Cut two 6-inch pieces of wire board.
2. Attach a solder lug to one end of each piece by twisting the end of the wire braid and inserting it into the smaller hole of the solder lug. Solder the braid to the lug taking care to completely fill the hole with solder.
3. On both sides of the motherboard:
   Place the braid on the ground land along side the cross rail so that the lug and about three inches of braid hang over the side, as shown in Figure 5-67. Solder the remaining three inches to the ground land. It may be helpful to first "tack" the braid in place with small amounts of solder and then, using the flat of the soldering iron to heat the braid, make a solid solder connection over the entire three inches. Make sure there are no solder bridges to the adjacent lands on the board. The lugs will be attached to the side rails of the mainframe after the motherboard has been installed.

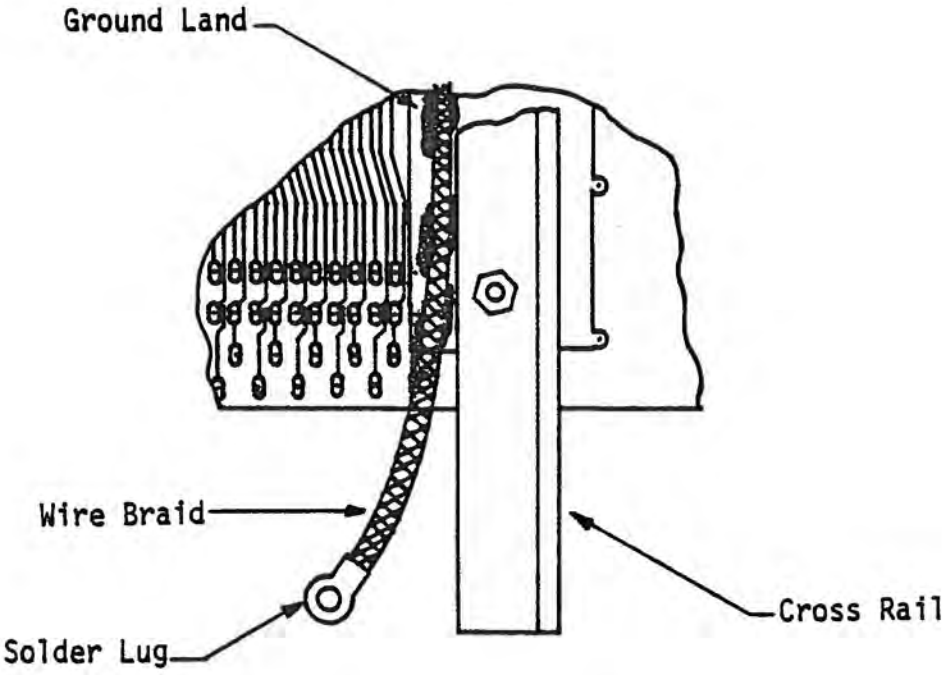


Figure 5-67 Chassis Ground Connection

F. Mounting the Motherboard in the Mainframe

1. Attach four #6-32 x 3/8" threaded spacers to the end holes in the cross rails, using #6-32 x 1/4" screws. Place the motherboard/cross rail assembly in the chassis so that the spacers at the front of the assembly align with the 8th hole (from the front) of the chassis side members. Secure the assembly to the chassis with four #6-32 x 1/4" screws.
2. Connect the four terminal ends on the bus wires to the terminal block (TB1) on the Power Supply Board as follows (see wiring diagram, Figure 5-65).

Voltage	Bus Connection	TB1 Connection
-18v	holes #52	slot #1
+8v	holes #1	slot #3
ground	holes #50	slot #9
+18v	holes #2	slot #10

Check for continuity between each bus connection and its respective terminal block connection.

3. To assure a good ground connection, rub the alodine coating off the chassis side member with steel wool. On each side of the board, connect the chassis ground wire from the motherboard to one of the holes on the chassis side member. Secure with a #6-32 x 3/8" screw and a #6-32 nut.

5-10. Front Panel Assembly

A. Front Panel Assembly Description

The Display/Control Board, micro switches and key switch are mounted at this point in the assembly process. These components are first installed on the subpanel; the subpanel is then mounted on the mainframe, followed by the dress panel.

> NOTE
>
> Before beginning the assembly processes described below, remove the subpanel from the mainframe and save the screws.

B. Micro Switch Wiring Description

Bag 3 contains a pair of micro switches. These are wired with 22-18 gauge (black) wire. Terminal and pin connections are insulated with heat shrink tubing. Carefully note the location on Figures 5-68 and 5-69 of the terminals which are to be wired.

1. Micro Switch Wiring

   a. Cut four lengths of 22 gauge (black) wire in the following lengths:

      2 6 inch lengths

      2 3 inch lengths

      Strip approximately 1/8 inch from the end of each wire and prepare the ends.

   b. Cut six lengths of shrink tube 3/4" long. Slip two lengths over each 6 inch wire and the remaining 2 pieces over each of smaller lengths.

Micro Switch A

a. Attach a connector pin to one end of both 6 inch wires.
b. Using Figure 5-68 as a guide, attach the other end of the 6 inch wire to micro switch terminals 1 and 2 (specified in the illustration). Solder into place.
c. Slip heat shrink tubing down over the terminals and connector pins and heat to shrink.
d. Insert a connector pin into each slot of the plastic pin housing (bag 2). The wiring need not be from a specific terminal to a specific slot in the pin housing.

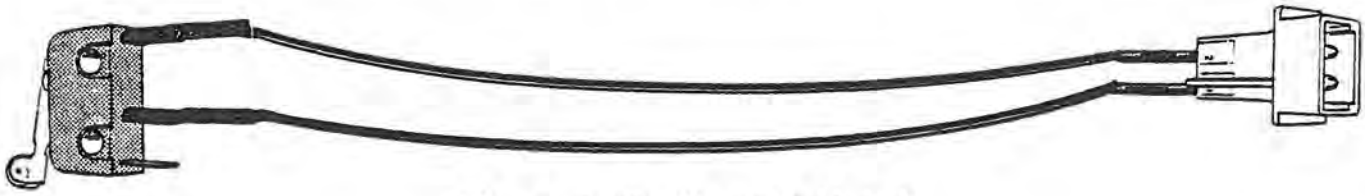


Figure 5-68 Micro Switch A

Micro Switch B

a. Attach a 3 inch wire to terminals 1 and 3 of micro switch B as specified in Figure 5-69.
b. Slip heat shrink tubing down over terminals and heat.

Figure 5-69 Micro Switch B

2. Micro Switch Installation Instructions

a. Using the two small 2-5 x 3/4 CS screws and #2 nuts attach micro switch A to the subpanel as illustrated in Figure 5-7Ø. (Micro switch A is inserted below the largest hole on the subpanel.) It is easiest to tighten the screw, if a nut driver is used to hold the nut.

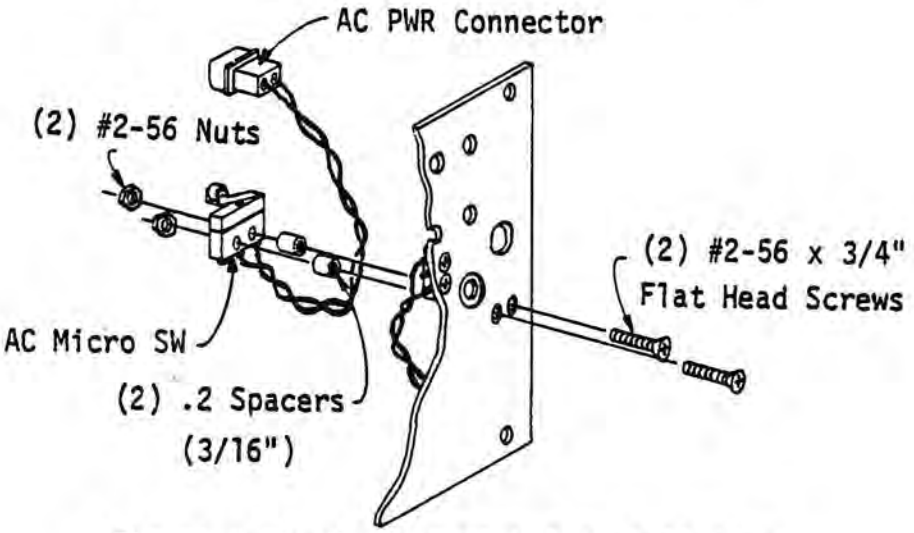


Figure 5-70 Micro Switch A Installation

b. Follow the same procedure defined in step 1 to install micro switch B. Figure 5-71 specifies the location of this switch.

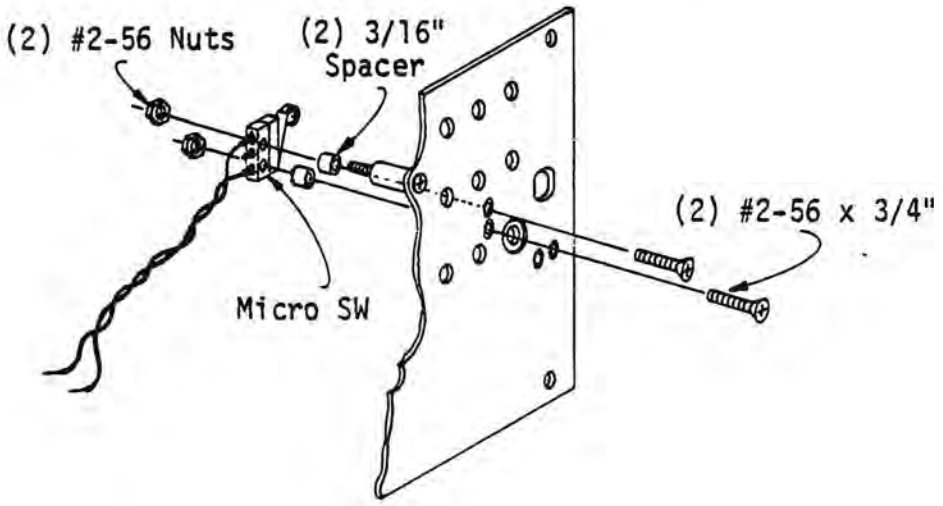


Figure 5-71 Micro Switch B Installation

C. Display Control Board Installation Instructions

1. Insert the wire from one terminal of micro switch B into pad X on the front of the Display/Control Board. Insert the wire from the other micro switch terminal into pad Y on the front Display/Control Board. Solder both into place. It is not critical for specific wires to go to specific pads.
2. Slip a #2 nut (Bag 4) over each switch of the Display/Control Board (see Figure 5-72) and tighten down with nut driver.
3. Attach the Display/Control Board to the subpanel with a #6-32 x 11" screw, 1.87 spacer (bag 4) and #6 x 32 nut. Figure 5-72 illustrates this assembly process.

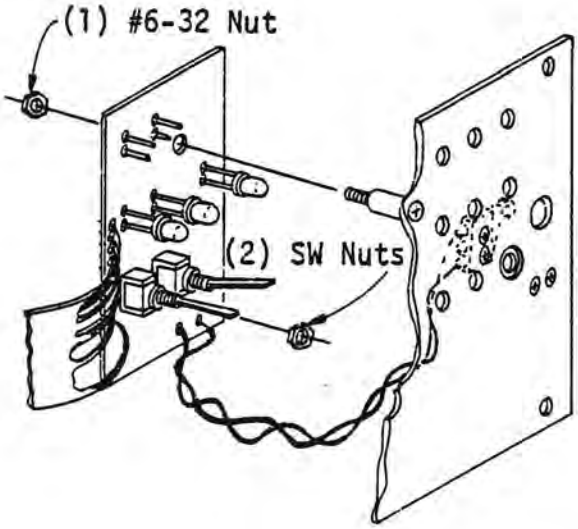


Figure 5-72 Display Control Board Installation

D. Key Switch Installation Description

Before installing the key switch study Figure 5-73 to determine the proper order and precise orientation of the hardware used for installation of this component. If the cam and 180° stop plate are not positioned correctly the key switch will not make proper contact with the microswitches.

1. Mount the subpanel on the mainframe.
2. Insert the key switch in the dress panel as illustrated in Figure 5-73.

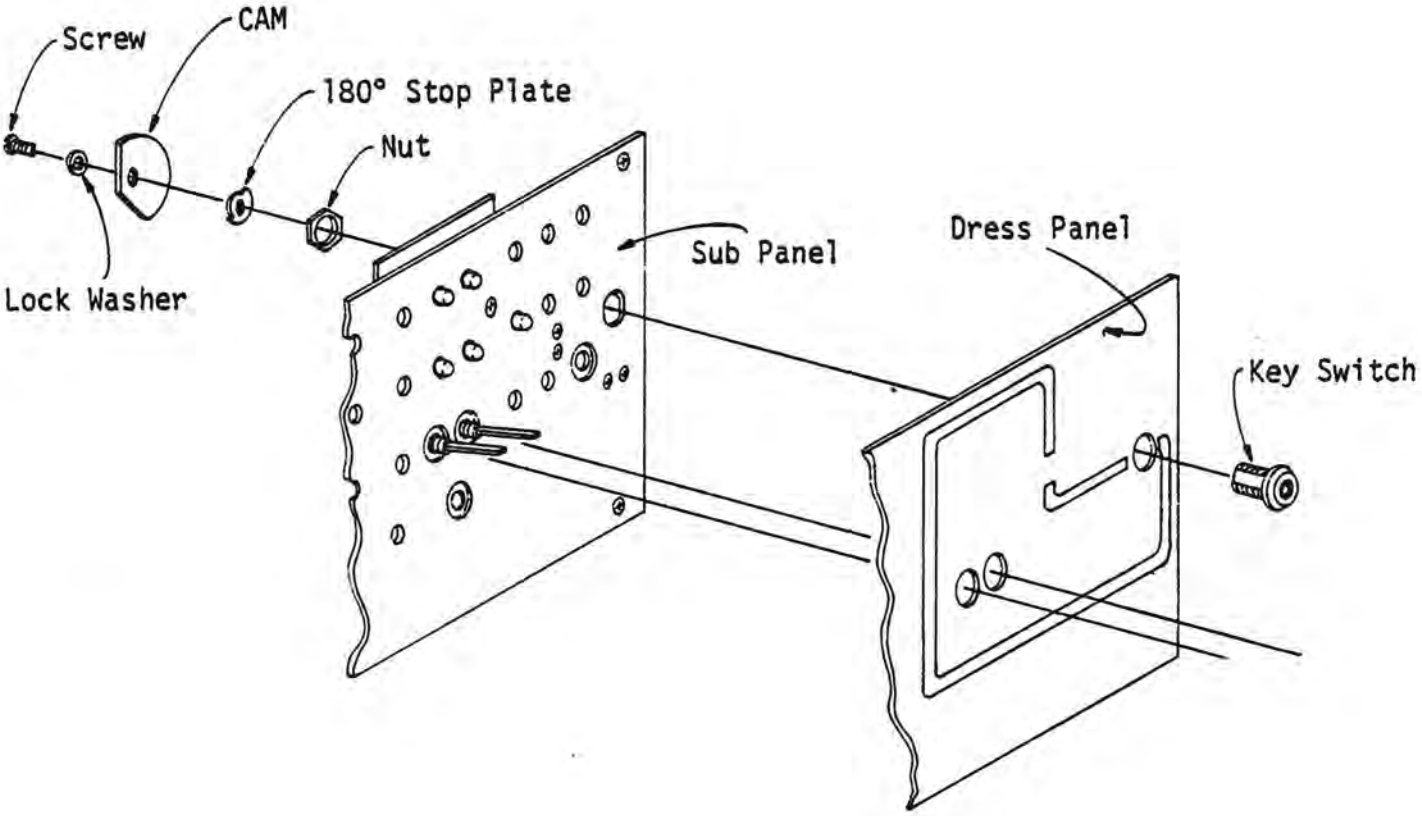


Figure 5-73 Key Switch Installation

NOTE

BE SURE THE KEY SWITCH IS INSTALLED WITH THE ROUND MARKING (ON THE THREADED PORTION OF THE SWITCH) UP.

3. Mount the dress panel (with key switch) on the subpanel as illustrated in Figure 5-73. Install the remainder of the hardware (180° stop plate, cam, star, lock washer, screw). Figure 5-74 is a side view of the assembled hardware, keyswitch, subpanel and dress panel.

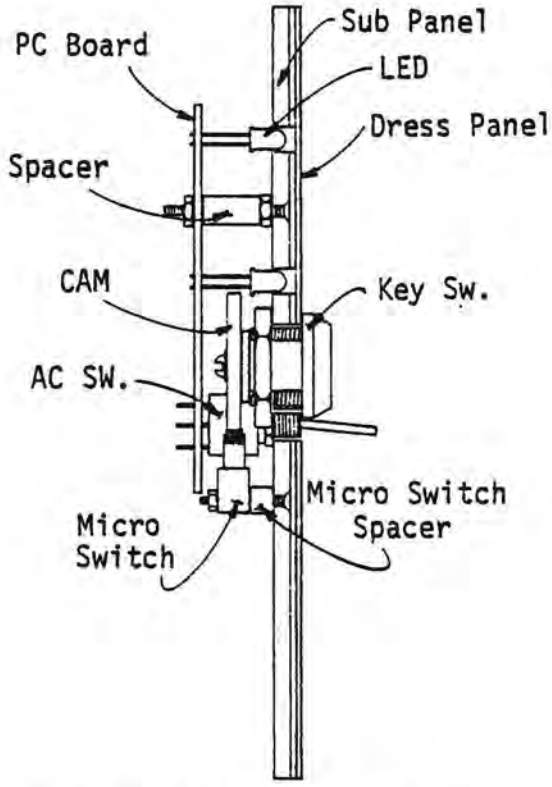


Figure 5-74 Dress Panel Mounting on the Subpanel

5-11. Interface Conductor Cable Wiring Description

The cables which interface the Turnkey module board with peripheral devices are wired to molex connectors. Wiring Instructions for TTY and 232 interface are provided. For a detailed description see section 3 (Advanced Operation, page 35 of this document.)

A. TTY Interface Conductor Cable Wiring Instruction

1. Strip 1 3/4" of the grey casing of the conductor cable (Turnkey module (Bag 5). This exposes the colored wires.
2. Strip 1/8" from the following wires, prepare the ends and attach a connector socket.

orange	green
red	black
yellow	

3. Insert the connector socket into the molex connector (Turnkey module Bag 5) slots specified below.

orange	1
red	2
yellow	3
green	4
black	8

Clip off the excess wires at the casing.

4. Slip two (2) 1 inch lengths of heat shrink tubing over the cable.
5. Strip 1/8 inch from the other end of the same wires on the cable. Prepare the ends.
6. Insert the stripped wires into the holes (specified in the table) of the 25 pin female DBS connector (Bag 5). Solder into place. The holesnumbers are marked on the plastic portion of the connector just above the hole.

6	orange
3	red
4	yellow
5	green
2	black

Clip off the excess wires at the casing.

7. Slip the heat shrink tube down to insulate exposed wires and heat to shrink.

B. RS-232 Interface Cable Wiring Instructions.

Follow the same step by step procedures defined in the previous subsection for stripping and inserting the cable wires into the female molex connector and female DBS. The following tables define the location of each wire.

Female Molex Connector

green	7
brown	5
yellow	9
orange	4
black	8
red	10

DBS Female Connector

green	5
brown	8
yellow	4
orange	2
black	7
red	3

C. Attach the female molex connector from the Display/Control board to male connector J-1.

D. Attach the female molex connector from the interface conductor cable to male connector J-2. Insert the other female connector in one of the rectangular openings in the back panel.

E. Fuse Insertion

Insert the 3 amp SLO-BLOW fuse (Bag 2 Power Supply Board) into the fuse holder.

Your Altair 8800b Turnkey is now completed. See section 1, page 3 of this document for instructions concerning the operation of this computer.

APPENDICES

8800BT P/S Bd.
Parts List
July 1977

APPENDIX A

Bag 1

1	KBH25005/Bridge Rectifier	100735
1	KBPC802 (TJ118-0) Bridge Rectifier	100733
1	TIP 140 or 141 with Mica Insulator and Washer	102819
1	TIP 145 or 146 with Mica Insulator and Washer	102820
2	IN4746	100726
2	180 Ohm 1/2w.	101998
3	Heat Sinks (Large)	101870

Bag 2

1	Terminal Block (150-4)	101627
1	Terminal Block (141-10)	101868
2	Jumpers (141)	101651
1	Fuse Holder	101813
2	T.B. Brackets	101652
1	Strain Relief	101719
4	Rubber Feet	101751
4	Ring Terminal, #10 Wire -#10 Bolt	101642
6	Spade Terminal, #10 Wire - #10 bolt	101643
6	Spade Terminal, #10 Wire -#6 Bolt	101644
4	Quik Dis. 10 Wire - 1/4" Tab	101645
4	Spade Terminal, 18 Wire - 6 Bolt	101646
1	Fuse - 3 amp SLO-BLOW	101772

Bag 3

4	2200MF 25v	100375

Bag 4

1	Plug, Mate-N-Lok, ckt.	101635
1	Receptacle, Mate-N-Lok, 10 ckt.	101636
8	Pin, Mate-N-Lok	101639
6	Socket, Mate-N-Lok	101640
4	Commoning Tab	101641
1	Plug, Mate-N-Lok, 2 ckt.	101637
1	Receptacle, Mate-N-Lok, 2 ckt.	101638

Bag 5

1	6-32 x 1/2" Screw	100918
19	6-32 x 3/8" Screw	100925
5	6-32 x 5/8" Screw	100916
9	6-32 x 9/16" Screw	100956
4	6-32 x 3/4" Screw	100935
1	8-32 x 1" Screw	100927
4	10-32 x 1/2" Screw	100958
13	6-32 x 1/4" Screw	100917
19	6-32 Nut	100933
4	#6 Snap On for Fan	
1	8-32 Nut	100929
4	10-32 Nut	100962
2	6-32 x 3/8" Nylon Screw	100959
2	6-32 Nylon Nut	100960
17	#6 Lockwasher	100942
1	#8 Lockwasher	100945
4	#10 Lockwasher	100963
5	3/4" 6-32 Spacer (Threaded)	101626
4	#6 Flat Washer	100943
1	#8 Flat Washer	100939
4	#10 Flat Washer	100961
4	3/8" 6-32 Spacer (Threaded)	101863

Misc.

1	Fan Filter	101757
1	95000Uf Capacitor	100391
1	P.C. Board	100202
1	Power Cord - 3 Wire	101742
1	Fan	101869
1	Transformer (Packed in Separate Box)	102616
20'	#18 Stranded Wire	103090
8'	#12 Stranded Wire	103092
10"	grn. braid	103064
8	lugs	101801
4	3/16" Cable Clamps	103023
20	Tie Wraps	103037
6"	3/8" Heat Shrink	103073
6"	1/2" Heat Shrink	103071
6"	1/4" Heat Shrink	103072

8800 BT FRONT PANEL
KITS PARTS LIST
JULY 1977

Bag 1

5	RL-21 Led	100702
1	10 Pin Female Conn	101768
1	SPDT STI-1 Switch	101879
1	MOM ST1-3 Switch	101880

Bag 2

8	Terminal Pin	101769
5	220 ohm 1/4w 5%	101901
1	IN4148 Diode	100705
1	Pin Housing Plastic	101637
2	Pin For Housing	101639

Bag 3

1	10" 8 Cond. Cable	103093
2	F.P. Micro Switch	102327
1	Key Lock Switch	102324
1	Key Lock Cam	100976

Bag 4

4	2-56 x 3/4" CS screw	100967
1	6-32 x 1" CS screw	100964
4	#2 Nut	100931
1	#6 Nut	100933
1	#6 Lockwasher	100942
1	.6 Plastic Spacer	101824
4	.187 Spacer	100968
3"	1/8" Heat Shrink	103068

MISC.

1	Front Panel PC Bd.	100219
1	Turnkey Dress Panel	100551
1	Mother Bd. 18 slot	100193

8800BT Module
Parts List
July 1977

Bag 1

1	7805 Regulator	101074
1	TIP 30 Regulator **	102821
1	1488 I.C. *	101112
1	1489 I.C. *	101113
8	2102A I.C. *	101107
1	34702 I.C. *	101099
1	6850 I.C. *	101098
1	74L00 I.C.	101080
2	7402 I.C.	101021
2	7404 I.C.	101022
4	7430 I.C.	101059
2	74LS73 I.C.	101119
1	74LS139 I.C.	101181
2	74LS257 I.C.	101182
3	7436 I.C.	101040
2	8216 I.C. *	101141

Bag 2

2	220 ohm 1/4w. res.	101901
3	440 ohm 1/4w. res.	101911
3	1K 1/4w. 5% res.	101903
1	1. 5K 1/4w. 5% res.	101917
1	1. 8K 1/4w. 5% res.	101980
1	3K 1/4w. 5% res.	101981
4	4. 7K 1/4w. 5% res.	101912
1	10 Meg 1/2w. 5% res.	102079

Bag 3

2	56pf 20v capacitor	100317
1	470pf 1KV cap disc	100316
5	1mf 35v cap epoxy	100308
4	22mf 16v cap tant	100395
3	IN914 Diode	100705
2	IN4739 Diode	100706
1	2N2222 Transistor	102822
1	EN2907 Transistor	102804

* DO NOT TOUCH PINS
** INCLUDE WASHER AND INSULATOR

Bag 4

1	6-32 x 3/8" Screw	100925
1	#6 Lockwasher	100942
1	6-32 Nut	100933
1	2.4576 Mhz xytl.	101741
2	10 Pin rt. angle	101812
18"	12 cond. cable	103058
4"	3/8" Shrink Tub.	103073
1	DB25S 25 Pin Conn.	102112
1	10 Pin Female Molex	101768
7	Terminal Pin Large	101769
1	Heat Sink Small	101667

Bag 5

11	16 Pin Sockets	102103
2	14 Pin Sockets	102102
5	24 Pin Sockets	102105
9	Dip Switch	102321

Misc.

1	Turnkey Module P.C. Bd.	100218

8800BT CPU BD.
PARTS LIST
JULY 1977

Bag 1

1	8080*	101070
1	8212*	101071
1	4009*	101143
1	24 Pin Socket	102105
1	40 Pin Socket	102106

Bag 2

2	8216 *	101141
1	8224*	101125
7	74367	101040
2	74368	101045
2	74LS14	101123
1	74LS13	101124
2	74LS04	101042
1	7805	101074
1	7812	101085

Bag 3

24	2.2K 1/2w.	101945

Bag 4

13	3.3K 1/2w.	102085
1	15K 1/2w.	102083
1	1K 1/2w.	101928
1	620 ohm 1/2w.	102095
1	330 ohm 1/2w.	101926
2	470 ohm 1/4w.	101902
1	10K 1/2w.	101932
1	100 ohm 1/2w.	101924
1	IN4733 5v.	100721
1	IN4730 3.9v.	100734
3	CS4410 or 2N4410	102806
1	10 ohm 2w	101960

Bag 5

22	.1mf 12V	100348

Bag 6

3	.1mf 50v	10038-
13	1mf 35v	100308
4	33 uf 16v	100326
2	10 uf 25v	100352
1	10mf 16v	100394
1	22mf 16v	100395

Bag 7

1	Small 10 Pin Rt. Angle Conn.	101798
1	100 Pin Edge Conn.	101864
2	Card Guides	101714
7	Ferrite Beads	101876
1	18 MHZ Crystal	101877
2	Heat Sink (Large)	101870
6	6-32 x 3/8 Screw	100925
2	6-32 Nut	100933
2	#6 Lockwasher	100942
2	4-40 x 5/8" Screw	100904
2	4-40 Nut	100932
2	#4 Lockwasher	100941

Misc.

1	PC Board 1	100198

*DO NOT TOUCH PINS.

APPENDIX B
OPERATING FROM 230 VOLTS

To allow operation of the 8800b Turnkey computer on 230 volts AC (nominal), make the following changes to the power supply:

1. Change the wiring of the P4 connector pin housing (Figure 1) according to Table A. Place a commoning tab over pins 8 and 10 and one over pins 7 and 9. (Note that no wire is connected to pin 7. Pin 7 must be installed, however, to hold the commoning tab. The exposed end of pin 7 should be covered with a small piece of heat-shrinkable tubing or equivalent instulation.)

Table A

Transformer Wire Color	P4 Pin Housing Slot Number
(Primary Side)	
Red	3
Blue	1
Green	2
Black	10
Red/Black	4
Blue/Black	8
Green/Black	6
White/Black	9

2. Change the wiring of the P4 socket housing according to the following table:

Table B

Wire	P4 Socket Housing Slot Number
Black AC input (Wire #37 from switch connector)	1
White AC input	9
Fan	8 7

CAUTION

The fan supplied with the 8800b Turnkey computer is designed for use with 115 VAC. It is connected across half the transformer primary for 230 VAC operation. To protect the fan motor, make sure this connection is correct.

3. To adjust the power supply for high or low line voltage, move the black AC wire from slot number 1 of the P4 socket housing to one of the following positions.

For low line voltage (215 VAC), use slot 2 of the P4 socket housing.

For high line voltage (245 VAC), use slot 3 of the P4 socket housing.

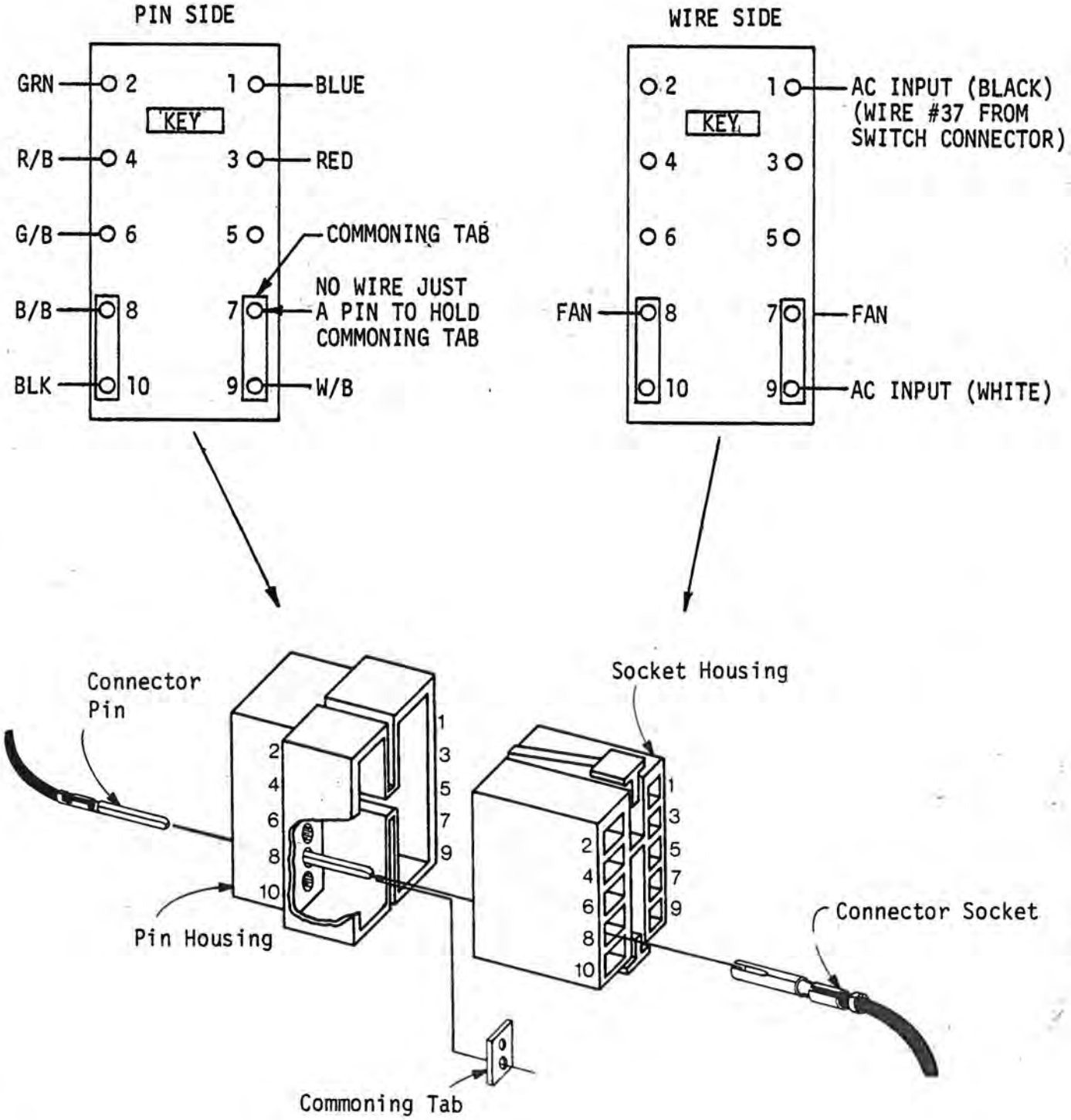


Figure 1
Connector P4

APPENDIX
ALTAIR 88ØØb BUS ASSIGNMENTS

The Altair 88ØØb system bus has 1ØØ lines. These are arranged 5Ø on each side of the plug-in boards. The following general rules apply to the Altair 88ØØb bus.

SYMBOLS a "P" prefix denotes a processor command/control signal. "S" denotes a processor status signal.

LOADING All inputs to a card are loaded with a maximum of one TTL low power load except for the Turnkey Module.

LEVEL All bus signals except those for the power supply are TTL compatible. Signals whose names are barred (DO DSBL, for example) are active low (Ø volts). All others are active high (+5 volts).

In the listing below, those signal names accompanied by * are ineffective or not used in the Altair 88ØØb Turnkey computer.

Number	Symbol	Name	Function
1	+8V	+8 volts	Unregulated input to 5 volt regulators
2	+18V	+18 volts	Positive unregulated voltage
3	XRDY	External Ready	For special applications: pulling this line low causes the processor to enter a wait state and allows the status of the normal ready line (PRDY) to be examined.
4	VI Ø	Vectored Interrupt	Line Ø
5	VI 1	Vectored Interrupt	Line 1
6	VI 2	Vectored Interrupt	Line 2
7	VI 3	Vectored Interrupt	Line 3
8	VI 4	Vectored Interrupt	Line 4
9	VI 5	Vectored Interrupt	Line 5
1Ø	VI 6	Vectored Interrupt	Line 6
11	VI 7	Vectored Interrupt	Line 7
12	XRDY2	Extra Ready line	For special applications.
13 to 17		To be assigned	

18	$\overline{\text{STA DSB}}$	$\overline{\text{STATUS DISABLE}}$	Puts buffers for the 8 status lines in their high-impetance tri-state. In this state, no information can be transferred.
19	C/C DSB	$\overline{\text{COMMAND/CONTROL DISABLE}}$	Puts the buffers for the 6 command/ control lines in their high-impedance tri-state.
*20	UNPROT	UNPROTECT	Input to the memory protect flip-flop.
*21	SS	SINGLE STEP	Indicates that the computer is in the process of performing a single step.
22	$\overline{\text{ADD DSB}}$	$\overline{\text{ADDRESS DISABLE}}$	Puts the buffers for the 16 address lines in their high-impedance tri-state.
23	$\overline{\text{DO DSBL}}$	$\overline{\text{DATA OUT DISABLE}}$	Puts the buffers for the 8 data out lines in their high-impedance tri-state.
24	Ø2	Phase 2 clock	
25	Ø1	Phase 1 clock	
26	PHLDA	Hold Acknowledge	Processor output signal which appears in response to the HOLD signal indicates that the data and address buffers will go to the high-impedance tri-state.
27	PWAIT	WAIT	Processor output indicates that the processor is in the WAIT state.
28	PINTE	Interrupt Enable	Indicates interrupts are enabled; displays contents of the CPU interrupt flip-flop. This flip-flop may be set or reset by the EI or DI instructions. When reset, it prevents the CPU from acknowledging interrupt requests.
29	A 5	Address Line 5	
3Ø	A 4	Address Line 4	
31	A3	Address Line 3	
32	A 15	Address Line 15	

33	A 12	Address Line 12	
34	A 9	Address Line 9	
35	DO 1	Data Out Line 1	
36	DO Ø	Data Out Line Ø	
37	A 1Ø	Address Line 1Ø	
38	DO 4	Data Out Line 4	
39	DO 4	Data Out Line 5	
4Ø	DO 6	Data Out Line 6	
41	DI 2	Data In Line 2	
42	DI 3	Data In Line 3	
43	DI 7	Data In Line 7	
44	SM1	M1	Status output that indicates that the processor is in the fetch cycle for the first byte of an instruction.
45	SOUT	OUT	Indicates that the address bus contains the address of an output device and the data bus will contain the output data when PWR is active.
46	SINP	INP	Indicates that the address bus contains the address of an input device
47	SMEMR	MEMR	Indicates that the data bus will carry memory read data.
48	SHLTA	HLTA	Acknowledges a HALT instruction.
49	$\overline{\text{CLOCK}}$	$\overline{\text{Clock}}$	Inverted output of the 2MHz oscillator that drives the clock.
5Ø	GND	Ground	
51	+8V	+8 volts	
52	-18V	-18 volts	
*53	$\overline{\text{SSW DSB}}$	$\overline{\text{SENSE SWITCH DISABLE}}$	Disables the data input buffers so that the inputs from the sense switches may be strobed onto the bidirectional data bus at the processor.
54	$\overline{\text{EXT CLR}}$	$\overline{\text{EXTERNAL CLEAR}}$	Clear signal for I/O devices.
55	RTC	Real Time Clock	
56	$\overline{\text{STSTB}}$	$\overline{\text{STATUS STROBE}}$	See section 2-2.

*57	DIG1	DATA INPUT GATA #1	Front panel control line.
*58	FRDY	Front Panel READY	
59 to 67		To be assigned	
68	MWRT	MEMORY WRITE	Indicates that the current data on the Data-Out bus is to be written into the memory location currently on the address bus.
69	$\overline{PS}$	$\overline{PROTECT\ STATUS}$	Indicates the status of the memory protect flip-flop on the memory board currently being addressed.
*7Ø	PROT	PROTECT	Input to the memory protect flip-flop on memory board currently being addressed.
*71	RUN	RUN	Indicates that the RUN/STOP flip-flop is reset.
72	PRDY	READY	Input that controls the run state of the processor. If the line is pulled low the processor will enter a WAIT state until it is released.
73	$\overline{PINT}$	$\overline{INTERRUPT\ REQUEST}$	The processor recognizes a request on this line at the end of the current instruction or while halted. If the processor is in the HOLD state or the Interrupt Enable flip-flop is reset it will not honor the request.
74	$\overline{PHOLD}$	$\overline{HOLD}$	Requests the processor to enter the HOLD state. This allows an external device to gain control of the bus as soon as the processor has completed its current machine cycle.
75	$\overline{PRESET}$	$\overline{RESET}$	While activated, the contents of the program counter are cleared and the instruction register is set to Ø.

76	PSYNC	SYNC	Provides a signal to indicate the beginning of each machine cycle.
77	$\overline{\text{PWR}}$	$\overline{\text{WRITE}}$	Used for memory write or I/O control. Data on the data bus is stable while PWR is active.
78	PDBIN	DATA BUS IN	Indicates to external devices that the data bus is in input mode.
79	AØ	Address Line Ø	
8Ø	A1	Address Line 1	
81	A2	Address Line 2	
82	A6	Address Line 6	
83	A7	Address Line 7	
84	A8	Address Line 8	
85	A13	Address Line 13	
86	A14	Address Line 14	
87	A11	Address Line 11	
88	DO 2	Data Out Line 2	
89	DO 3	Data Out Line 3	
9Ø	DO 7	Data Out Line 7	
91	DI 4	Data In Line 4	
92	DI 5	Data In Line 5	
93	DI 6	Data In Line 6	
94	DI 1	Data In Line 1	
95	DI Ø	Data In Line Ø	
96	SINTA	INTA	Acknowledge signal for interrupt request.
97	$\overline{\text{SWO}}$	WRITE OUT	Indicates that the operation in the current machine cycle is a WRITE memory or output function.
98	SSTACK	STACK	Indicates that the address bus holds the push down stack address from the Stack Pointer.
99	$\overline{\text{POC}}$	$\overline{\text{Power-On-Clear}}$	
100	GND	Ground	

INDEX

INTEL 8080 MICROCOMPUTER SYSTEM USER'S MANUAL

CHAPTER 1 THE FUNCTIONS OF A COMPUTER

This chapter introduces certain basic computer concepts. It provides background information and definitions which will be useful in later chapters of this manual. Those already familiar with computers may skip this material, at their option.

A TYPICAL COMPUTER SYSTEM

A typical digital computer consists of:

a) A central processor unit (CPU)
b) A memory
c) Input/output (I/O) ports

The memory serves as a place to store **Instructions**, the coded pieces of information that direct the activities of the CPU, and **Data**, the coded pieces of information that are processed by the CPU. A group of logically related instructions stored in memory is referred to as a **Program**. The CPU "reads" each instruction from memory in a logically determined sequence, and uses it to initiate processing actions. If the program sequence is coherent and logical, processing the program will produce intelligible and useful results.

The memory is also used to store the data to be manipulated, as well as the instructions that direct that manipulation. The program must be organized such that the CPU does not read a non-instruction word when it expects to see an instruction. The CPU can rapidly access any data stored in memory; but often the memory is not large enough to store the entire data bank required for a particular application. The problem can be resolved by providing the computer with one or more **Input Ports**. The CPU can address these ports and input the data contained there. The addition of input ports enables the computer to receive information from external equipment (such as a paper tape reader or floppy disk) at high rates of speed and in large volumes.

A computer also requires one or more **Output Ports** that permit the CPU to communicate the result of its processing to the outside world. The output may go to a display, for use by a human operator, to a peripheral device that produces "hard-copy," such as a line-printer, to a peripheral storage device, such as a floppy disk unit, or the output may constitute process control signals that direct the operations of another system, such as an automated assembly line. Like input ports, output ports are addressable. The input and output ports together permit the processor to communicate with the outside world.

The CPU unifies the system. It controls the functions performed by the other components. The CPU must be able to fetch instructions from memory, decode their binary contents and execute them. It must also be able to reference memory and I/O ports as necessary in the execution of instructions. In addition, the CPU should be able to recognize and respond to certain external control signals, such as INTERRUPT and WAIT requests. The functional units within a CPU that enable it to perform these functions are described below.

THE ARCHITECTURE OF A CPU

A typical central processor unit (CPU) consists of the following interconnected functional units:

- Registers
- Arithmetic/Logic Unit (ALU)
- Control Circuitry

Registers are temporary storage units within the CPU. Some registers, such as the program counter and instruction register, have dedicated uses. Other registers, such as the accumulator, are for more general purpose use.

Accumulator:

The accumulator usually stores one of the operands to be manipulated by the ALU. A typical instruction might direct the ALU to add the contents of some other register to the contents of the accumulator and store the result in the accumulator itself. In general, the accumulator is both a source (operand) and a destination (result) register.

Often a CPU will include a number of additional general purpose registers that can be used to store operands or intermediate data. The availability of general purpose

registers eliminates the need to "shuffle" intermediate results back and forth between memory and the accumulator, thus improving processing speed and efficiency.

Program Counter (Jumps, Subroutines and the Stack):

The instructions that make up a program are stored in the system's memory. The central processor references the contents of memory, in order to determine what action is appropriate. This means that the processor must know which location contains the next instruction.

Each of the locations in memory is numbered, to distinguish it from all other locations in memory. The number which identifies a memory location is called its **Address.**

The processor maintains a counter which contains the address of the next program instruction. This register is called the **Program Counter.** The processor updates the program counter bv adding "1" to the counter each time it fetches an instruction, so that the program counter is always current (pointing to the next instruction).

The programmer therefore stores his instructions in numerically adjacent addresses, so that the lower addresses contain the first instructions to be executed and the higher addresses contain later instructions. The only time the programmer may violate this sequential rule is when an instruction in one section of memory is a **Jump** instruction to another section of memory.

A jump instruction contains the address of the instruction which is to follow it. The next instruction may be stored in any memory location, as long as the programmed jump specifies the correct address. During the execution of a jump instruction, the processor replaces the contents of its program counter with the address embodied in the Jump. Thus, the logical continuity of the program is maintained.

A special kind of program jump occurs when the stored program "**Calls**" a subroutine. In this kind of jump, the processor is required to "remember" the contents of the program counter at the time that the jump occurs. This enables the processor to resume execution of the main program when it is finished with the last instruction of the subroutine.

A **Subroutine** is a program within a program. Usually it is a general-purpose set of instructions that must be executed repeatedly in the course of a main program. Routines which calculate the square, the sine, or the logarithm of a program variable are good examples of functions often written as subroutines. Other examples might be programs designed for inputting or outputting data to a particular peripheral device.

The processor has a special way of handling subroutines, in order to insure an orderly return to the main program. When the processor receives a Call instruction, it increments the Program Counter and stores the counter's contents in a reserved memory area known as the **Stack.** The Stack thus saves the address of the instruction to be executed after the subroutine is completed. Then the processor loads the address specified in the Call into its Program Counter. The next instruction fetched will therefore be the first step of the subroutine.

The last instruction in any subroutine is a **Return.** Such an instruction need specify no address. When the processor fetches a Return instruction, it simply replaces the current contents of the Program Counter with the address on the top of the stack. This causes the processor to resume execution of the calling program at the point immediately following the original Call Instruction.

Subroutines are often **Nested;** that is, one subroutine will sometimes call a second subroutine. The second may call a third, and so on. This is perfectly acceptable, as long as the processor has enough capacity to store the necessary return addresses, and the logical provision for doing so. In other words, the maximum depth of nesting is determined by the depth of the stack itself. If the stack has space for storing three return addresses, then three levels of subroutines may be accommodated.

Processors have different ways of maintaining stacks. Some have facilities for the storage of return addresses built into the processor itself. Other processors use a reserved area of external memory as the stack and simply maintain a **Pointer** register which contains the address of the most recent stack entry. The external stack allows virtually unlimited subroutine nesting. In addition, if the processor provides instructions that cause the contents of the accumulator and other general purpose registers to be "pushed" onto the stack or "popped" off the stack via the address stored in the stack pointer, multi-level interrupt processing (described later in this chapter) is possible. The status of the processor (i.e., the contents of all the registers) can be saved in the stack when an interrupt is accepted and then restored after the interrupt has been serviced. This ability to save the processor's status at any given time is possible even if an interrupt service routine, itself, is interrupted.

Instruction Register and Decoder:

Every computer has a **Word Length** that is characteristic of that machine. A computer's word length is usually determined by the size of its internal storage elements and interconnecting paths (referred to as **Busses**); for example, a computer whose registers and busses can store and transfer 8 bits of information has a characteristic word length of 8-bits and is referred to as an 8-bit parallel processor. An eight-bit parallel processor generally finds it most efficient to deal with eight-bit binary fields, and the memory associated with such a processor is therefore organized to store eight bits in each addressable memory location. Data and instructions are stored in memory as eight-bit binary numbers, or as numbers that are integral multiples of eight bits: 16 bits, 24 bits, and so on. This characteristic eight-bit field is often referred to as a **Byte.**

Each operation that the processor can perform is identified by a unique byte of data known as an **Instruction**

Code or **Operation Code**. An eight-bit word used as an instruction code can distinguish between 256 alternative actions, more than adequate for most processors.

The processor fetches an instruction in two distinct operations. First, the processor transmits the address in its Program Counter to the memory. Then the memory returns the addressed byte to the processor. The CPU stores this instruction byte in a register known as the **Instruction Register**, and uses it to direct activities during the remainder of the instruction execution.

The mechanism by which the processor translates an instruction code into specific processing actions requires more elaboration than we can here afford. The concept, however, should be intuitively clear to any logic designer. The eight bits stored in the instruction register can be decoded and used to selectively activate one of a number of output lines, in this case up to 256 lines. Each line represents a set of activities associated with execution of a particular instruction code. The enabled line can be combined with selected timing pulses, to develop electrical signals that can then be used to initiate specific actions. This translation of code into action is performed by the **Instruction Decoder** and by the associated control circuitry.

An eight-bit instruction code is often sufficient to specify a particular processing action. There are times, however, when execution of the instruction requires more information than eight bits can convey.

One example of this is when the instruction references a memory location. The basic instruction code identifies the operation to be performed, but cannot specify the object address as well. In a case like this, a two- or three-byte instruction must be used. Successive instruction bytes are stored in sequentially adjacent memory locations, and the processor performs two or three fetches in succession to obtain the full instruction. The first byte retrieved from memory is placed in the processor's instruction register, and subsequent bytes are placed in temporary storage; the processor then proceeds with the execution phase. Such an instruction is referred to as **Variable Length**.

Address Register(s):

A CPU may use a register or register-pair to hold the address of a memory location that is to be accessed for data. If the address register is **Programmable**, (i.e., if there are instructions that allow the programmer to alter the contents of the register) the program can "build" an address in the address register prior to executing a **Memory Reference** instruction (i.e., an instruction that reads data from memory, writes data to memory or operates on data stored in memory).

Arithmetic/Logic Unit (ALU):

All processors contain an arithmetic/logic unit, which is often referred to simply as the **ALU**. The ALU, as its name implies, is that portion of the CPU hardware which performs the arithmetic and logical operations on the binary data.

The ALU must contain an **Adder** which is capable of combining the contents of two registers in accordance with the logic of binary arithmetic. This provision permits the processor to perform arithmetic manipulations on the data it obtains from memory and from its other inputs.

Using only the basic adder a capable programmer can write routines which will subtract, multiply and divide, giving the machine complete arithmetic capabilities. In practice, however, most ALUs provide other built-in functions, including hardware subtraction, boolean logic operations, and shift capabilities.

The ALU contains **Flag Bits** which specify certain conditions that arise in the course of arithmetic and logical manipulations. Flags typically include **Carry, Zero, Sign,** and **Parity**. It is possible to program jumps which are conditionally dependent on the status of one or more flags. Thus, for example, the program may be designed to jump to a special routine if the carry bit is set following an addition instruction.

Control Circuitry:

The control circuitry is the primary functional unit within a CPU. Using clock inputs, the control circuitry maintains the proper sequence of events required for any processing task. After an instruction is fetched and decoded, the control circuitry issues the appropriate signals (to units both internal and external to the CPU) for initiating the proper processing action. Often the control circuitry will be capable of responding to external signals, such as an interrupt or wait request. An **Interrupt** request will cause the control circuitry to temporarily interrupt main program execution, jump to a special routine to service the interrupting device, then automatically return to the main program. A **Wait** request is often issued by a memory or I/O element that operates slower than the CPU. The control circuitry will idle the CPU until the memory or I/O port is ready with the data.

COMPUTER OPERATIONS

There are certain operations that are basic to almost any computer. A sound understanding of these basic operations is a necessary prerequisite to examining the specific operations of a particular computer.

Timing:

The activities of the central processor are cyclical. The processor fetches an instruction, performs the operations required, fetches the next instruction, and so on. This orderly sequence of events requires precise timing, and the CPU therefore requires a free running oscillator clock which furnishes the reference for all processor actions. The combined fetch and execution of a single instruction is referred to as an **Instruction Cycle**. The portion of a cycle identified

with a clearly defined activity is called a **State**. And the interval between pulses of the timing oscillator is referred to as a **Clock Period**. As a general rule, one or more clock periods are necessary for the completion of a state, and there are several states in a cycle.

Instruction Fetch:

The first state(s) of any instruction cycle will be dedicated to fetching the next instruction. The CPU issues a read signal and the contents of the program counter are sent to memory, which responds by returning the next instruction word. The first byte of the instruction is placed in the instruction register. If the instruction consists of more than one byte, additional states are required to fetch each byte of the instruction. When the entire instruction is present in the CPU, the program counter is incremented (in preparation for the next instruction fetch) and the instruction is decoded. The operation specified in the instruction will be executed in the remaining states of the instruction cycle. The instruction may call for a memory read or write, an input or output and/or an internal CPU operation, such as a register-to-register transfer or an add-registers operation.

Memory Read:

An instruction **fetch** is merely a special memory read operation that brings the instruction to the CPU's instruction register. The instruction fetched may then call for data to be read from memory into the CPU. The CPU again issues a read signal and sends the proper memory address; memory responds by returning the requested word. The data received is placed in the accumulator or one of the other general purpose registers (not the instruction register).

Memory Write:

A memory write operation is similar to a read except for the direction of data flow. The CPU issues a write signal, sends the proper memory address, then sends the data word to be written into the addressed memory location.

Wait (memory synchronization):

As previously stated, the activities of the processor are timed by a master clock oscillator. The clock period determines the timing of all processing activity.

The speed of the processing cycle, however, is limited by the memory's **Access Time**. Once the processor has sent a read address to memory, it cannot proceed until the memory has had time to respond. Most memories are capable of responding much faster than the processing cycle requires. A few, however, cannot supply the addressed byte within the minimum time established by the processor's clock.

Therefore a processor should contain a synchronization provision, which permits the memory to request a **Wait state**. When the memory receives a read or write enable signal, it places a request signal on the processor's READY line, causing the CPU to idle temporarily. After the memory has had time to respond, it frees the processor's READY line, and the instruction cycle proceeds.

Input/Output:

Input and Output operations are similar to memory read and write operations with the exception that a peripheral I/O device is addressed instead of a memory location. The CPU issues the appropriate input or output control signal, sends the proper device address and either receives the data being input or sends the data to be output.

Data can be input/output in either parallel or serial form. All data within a digital computer is represented in binary coded form. A binary data word consists of a group of bits; each bit is either a one or a zero. **Parallel** I/O consists of transferring all bits in the word at the same time, one bit per line. **Serial** I/O consists of transferring one bit at a time on a single line. Naturally serial I/O is much slower, but it requires considerably less hardware than does parallel I/O.

Interrupts:

Interrupt provisions are included on many central processors, as a means of improving the processor's efficiency. Consider the case of a computer that is processing a large volume of data, portions of which are to be output to a printer. The CPU can output a byte of data within a single machine cycle but it may take the printer the equivalent of many machine cycles to actually print the character specified by the data byte. The CPU could then remain idle waiting until the printer can accept the next data byte. If an interrupt capability is implemented on the computer, the CPU can output a data byte then return to data processing. When the printer is ready to accept the next data byte, it can request an interrupt. When the CPU acknowledges the interrupt, it suspends main program execution and automatically branches to a routine that will output the next data byte. After the byte is output, the CPU continues with main program execution. Note that this is, in principle, quite similar to a subroutine call, except that the jump is initiated externally rather than by the program.

More complex interrupt structures are possible, in which several interrupting devices share the same processor but have different priority levels. Interruptive processing is an important feature that enables maximum untilization of a processor's capacity for high system throughput.

Hold:

Another important feature that improves the throughput of a processor is the **Hold**. The hold provision enables **Direct Memory Access** (DMA) operations.

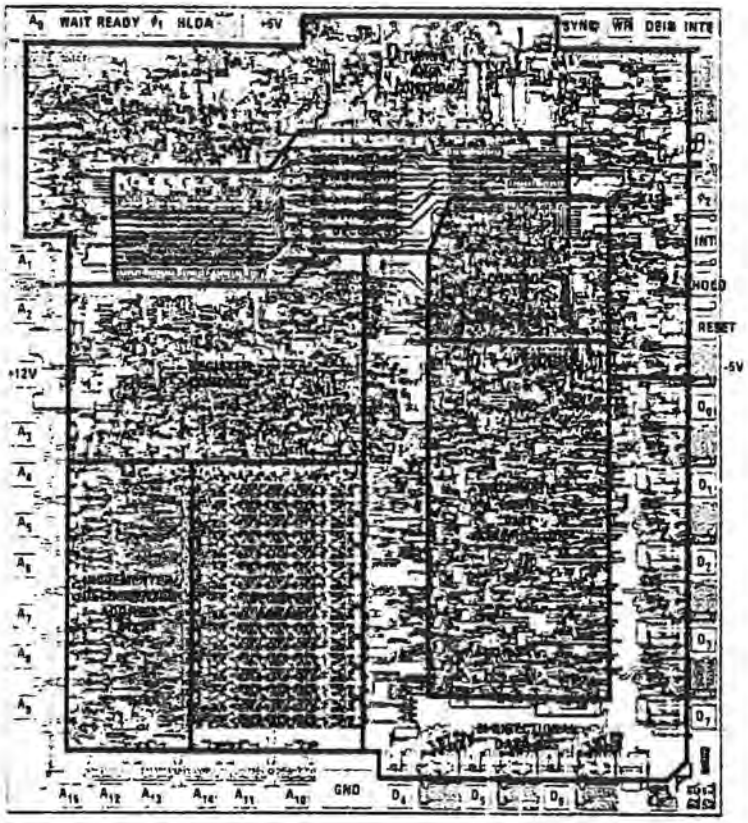
In ordinary input and output operations, the processor itself supervises the entire data transfer. Information to be placed in memory is transferred from the input device to the processor, and then from the processor to the designated memory location. In similar fashion, information that goes

from memory to output devices goes by way of the processor.

Some peripheral devices, however, are capable of transferring information to and from memory much faster than the processor itself can accomplish the transfer. If any appreciable quantity of data must be transferred to or from such a device, then **system throughput** will be increased by having the device accomplish the transfer directly. The processor must temporarily suspend its operation during such a transfer, to prevent conflicts that would arise if processor and peripheral device attempted to access memory simultaneously. It is for this reason that a **hold** provision is included on some processors.

CHAPTER 2
THE 8080 CENTRAL PROCESSOR UNIT

The 8080 is a complete 8-bit parallel, central processor unit (CPU) for use in general purpose digital computer systems. It is fabricated on a single LSI chip (see Figure 2-1), using Intel's n-channel silicon gate MOS process. The 8080 transfers data and internal state information via an 8-bit, bidirectional 3-state Data Bus (D_0-D_7). Memory and peripheral device addresses are transmitted over a separate 16-bit 3-state Address Bus (A_0-A_{15}). Six timing and control outputs (SYNC, DBIN, WAIT, $\overline{WR}$, HLDA and INTE) emanate from the 8080, while four control inputs (READY, HOLD, INT and RESET), four power inputs (+12v, +5v, -5v, and GND) and two clock inputs (ϕ_1 and ϕ_2) are accepted by the 8080.



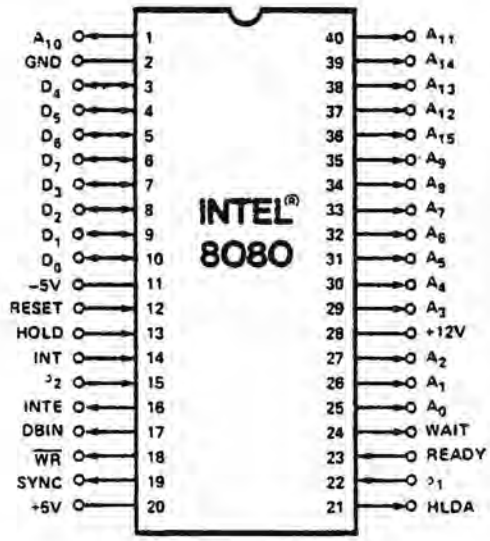


Figure 2-1. 8080 Photomicrograph With Pin Designations

ARCHITECTURE OF THE 8080 CPU

The 8080 CPU consists of the following functional units:

- Register array and address logic
- Arithmetic and logic unit (ALU)
- Instruction register and control section
- Bi-directional, 3-state data bus buffer

Figure 2-2 illustrates the functional blocks within the 8080 CPU.

Registers:

The register section consists of a static RAM array organized into six 16-bit registers:

- Program counter (PC)
- Stack pointer (SP)
- Six 8-bit general purpose registers arranged in pairs, referred to as B,C; D,E; and H,L
- A temporary register pair called W,Z

The program counter maintains the memory address of the current program instruction and is incremented automatically during every instruction fetch. The stack pointer maintains the address of the next available stack location in memory. The stack pointer can be initialized to use any portion of read-write memory as a stack. The stack pointer is decremented when data is "pushed" onto the stack and incremented when data is "popped" off the stack (i.e., the stack grows "downward").

The six general purpose registers can be used either as single registers (8-bit) or as register pairs (16-bit). The temporary register pair, W,Z, is not program addressable and is only used for the internal execution of instructions.

Eight-bit data bytes can be transferred between the internal bus and the register array via the register-select multiplexer. Sixteen-bit transfers can proceed between the register array and the address latch or the incrementer/decrementer circuit. The address latch receives data from any of the three register pairs and drives the 16 address output buffers (A_0-A_{15}), as well as the incrementer/decrementer circuit. The incrementer/decrementer circuit receives data from the address latch and sends it to the register array. The 16-bit data can be incremented or decremented or simply transferred between registers.

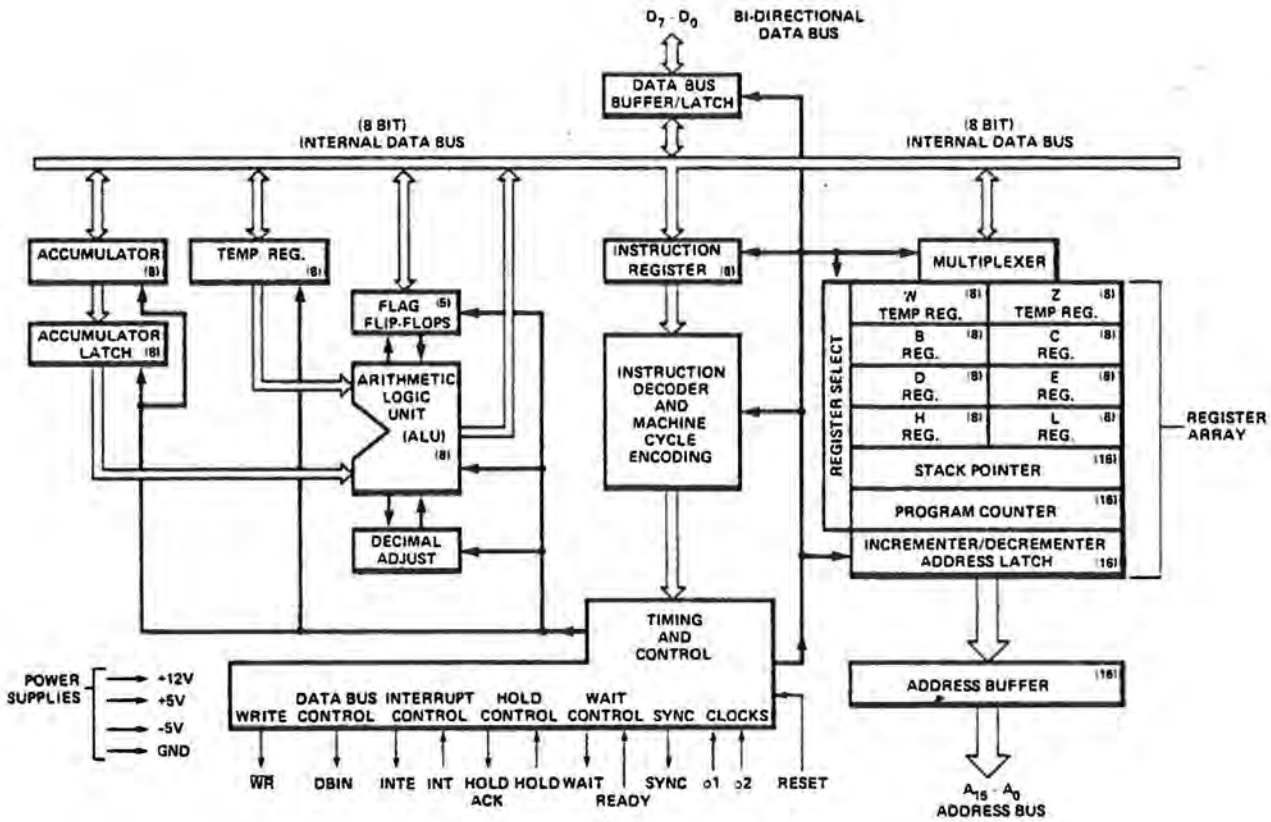


Figure 2-2. 8080 CPU Functional Block Diagram

Arithmetic and Logic Unit (ALU):

The ALU contains the following registers:

- An 8-bit accumulator
- An 8-bit temporary accumulator (ACT)
- A 5-bit flag register: zero, carry, sign, parity and auxiliary carry
- An 8-bit temporary register (TMP)

Arithmetic, logical and rotate operations are performed in the ALU. The ALU is fed by the temporary register (TMP) and the temporary accumulator (ACT) and carry flip-flop. The result of the operation can be transferred to the internal bus or to the accumulator; the ALU also feeds the flag register.

The temporary register (TMP) receives information from the internal bus and can send all or portions of it to the ALU, the flag register and the internal bus.

The accumulator (ACC) can be loaded from the ALU and the internal bus and can transfer data to the temporary accumulator (ACT) and the internal bus. The contents of the accumulator (ACC) and the auxiliary carry flip-flop can be tested for decimal correction during the execution of the DAA instruction (see Chapter 4).

Instruction Register and Control:

During an instruction fetch, the first byte of an instruction (containing the OP code) is transferred from the internal bus to the 8-bit instruction register.

The contents of the instruction register are, in turn, available to the instruction decoder. The output of the decoder, combined with various timing signals, provides the control signals for the register array, ALU and data buffer blocks. In addition, the outputs from the instruction decoder and external control signals feed the timing and state control section which generates the state and cycle timing signals.

Data Bus Buffer:

This 8-bit bidirectional 3-state buffer is used to isolate the CPU's internal bus from the external data bus (D_0 through D_7). In the output mode, the internal bus content is loaded into an 8-bit latch that, in turn, drives the data bus output buffers. The output buffers are switched off during input or non-transfer operations.

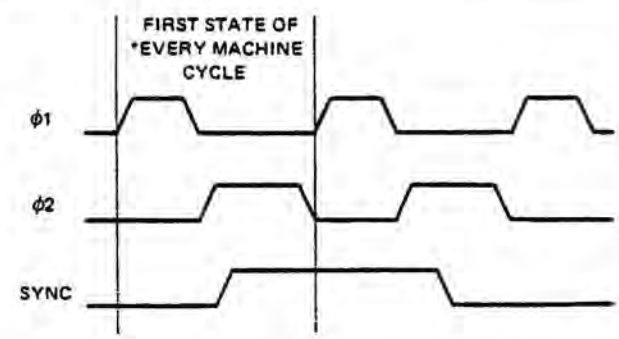
During the input mode, data from the external data bus is transferred to the internal bus. The internal bus is precharged at the beginning of each internal state, except for the transfer state (T_3—described later in this chapter).

THE PROCESSOR CYCLE

An **instruction cycle** is defined as the time required to fetch and execute an instruction. During the fetch, a selected instruction (one, two or three bytes) is extracted from memory and deposited in the CPU's instruction register. During the execution phase, the instruction is decoded and translated into specific processing activities.

Every instruction cycle consists of one, two, three, four or five machine cycles. A **machine cycle** is required each time the CPU accesses memory or an I/O port. The fetch portion of an instruction cycle requires one machine cycle for each byte to be fetched. The duration of the execution portion of the instruction cycle depends on the kind of instruction that has been fetched. Some instructions do not require any machine cycles other than those necessary to fetch the instruction; other instructions, however, require additional machine cycles to write or read data to/from memory or I/O devices. The DAD instruction is an exception in that it requires two additional machine cycles to complete an internal register-pair add (see Chapter 4).

Each machine cycle consists of three, four or five states. A state is the smallest unit of processing activity and is defined as the interval between two successive positive-going transitions of the ϕ_1 driven clock pulse. The 8080 is driven by a two-phase clock oscillator. All processing activities are referred to the period of this clock. The two non-overlapping clock pulses, labeled ϕ_1 and ϕ_2, are furnished by external circuitry. It is the ϕ_1 clock pulse which divides each machine cycle into states. Timing logic within the 8080 uses the clock inputs to produce a SYNC pulse, which identifies the beginning of every machine cycle. The SYNC pulse is triggered by the low-to-high transition of ϕ_2, as shown in Figure 2-3.



*SYNC DOES NOT OCCUR IN THE SECOND AND THIRD MACHINE CYCLES OF A DAD INSTRUCTION SINCE THESE MACHINE CYCLES ARE USED FOR AN INTERNAL REGISTER-PAIR ADD.

Figure 2-3. ϕ_1, ϕ_2 And SYNC Timing

There are three exceptions to the defined duration of a state. They are the WAIT state, the hold (HLDA) state and the halt (HLTA) state, described later in this chapter. Because the WAIT, the HLDA, and the HLTA states depend upon external events, they are by their nature of indeterminate length. Even these exceptional states, however, must

be synchronized with the pulses of the driving clock. Thus, the duration of all states are integral multiples of the clock period.

To summarize then, each **clock period** marks a **state**; three to five states constitute a machine cycle; and one to five **machine cycles** comprise an **instruction cycle**. A full instruction cycle requires anywhere from four to eighteen states for its completion, depending on the kind of instruction involved.

Machine Cycle Identification:

With the exception of the DAD instruction, there is just one consideration that determines how many machine cycles are required in any given instruction cycle: the number of times that the processor must reference a memory address or an addressable peripheral device, in order to fetch and execute the instruction. Like many processors, the 8080 is so constructed that it can transmit only one address per machine cycle. Thus, if the fetch and execution of an instruction requires two memory references, then the instruction cycle associated with that instruction consists of two machine cycles. If five such references are called for, then the instruction cycle contains five machine cycles.

Every instruction cycle has at least one reference to memory, during which the instruction is fetched. An instruction cycle must always have a fetch, even if the execution of the instruction requires no further references to memory. The first machine cycle in every instruction cycle is therefore a FETCH. Beyond that, there are no fast rules. It depends on the kind of instruction that is fetched.

Consider some examples. The add-register (ADD r) instruction is an instruction that requires only a single machine cycle (FETCH) for its completion. In this one-byte instruction, the contents of one of the CPU's six general purpose registers is added to the existing contents of the accumulator. Since all the information necessary to execute the command is contained in the eight bits of the instruction code, only one memory reference is necessary. Three states are used to extract the instruction from memory, and one additional state is used to accomplish the desired addition. The entire instruction cycle thus requires only one machine cycle that consists of four states, or four periods of the external clock.

Suppose now, however, that we wish to add the contents of a specific memory location to the existing contents of the accumulator (ADD M). Although this is quite similar in principle to the example just cited, several additional steps will be used. An extra machine cycle will be used, in order to address the desired memory location.

The actual sequence is as follows. First the processor extracts from memory the one-byte instruction word addressed by its program counter. This takes three states. The eight-bit instruction word obtained during the FETCH machine cycle is deposited in the CPU's instruction register and used to direct activities during the remainder of the instruction cycle. Next, the processor sends out, as an address, the contents of its H and L registers. The eight-bit data word returned during this MEMORY READ machine cycle is placed in a temporary register inside the 8080 CPU. By now three more clock periods (states) have elapsed. In the seventh and final state, the contents of the temporary register are added to those of the accumulator. Two machine cycles, consisting of seven states in all, complete the "ADD M" instruction cycle.

At the opposite extreme is the save H and L registers (SHLD) instruction, which requires five machine cycles. During an "SHLD" instruction cycle, the contents of the processor's H and L registers are deposited in two sequentially adjacent memory locations; the destination is indicated by two address bytes which are stored in the two memory locations immediately following the operation code byte. The following sequence of events occurs:

(1) A FETCH machine cycle, consisting of four states. During the first three states of this machine cycle, the processor fetches the instruction indicated by its program counter. The program counter is then incremented. The fourth state is used for internal instruction decoding.

(2) A MEMORY READ machine cycle, consisting of three states. During this machine cycle, the byte indicated by the program counter is read from memory and placed in the processor's Z register. The program counter is incremented again.

(3) Another MEMORY READ machine cycle, consisting of three states, in which the byte indicated by the processor's program counter is read from memory and placed in the W register. The program counter is incremented, in anticipation of the next instruction fetch.

(4) A MEMORY WRITE machine cycle, of three states, in which the contents of the L register are transferred to the memory location pointed to by the present contents of the W and Z registers. The state following the transfer is used to increment the W,Z register pair so that it indicates the next memory location to receive data.

(5) A MEMORY WRITE machine cycle, of three states, in which the contents of the H register are transferred to the new memory location pointed to by the W,Z register pair.

In summary, the "SHLD" instruction cycle contains five machine cycles and takes 16 states to execute.

Most instructions fall somewhere between the extremes typified by the "ADD r" and the "SHLD" instructions. The input (INP) and the output (OUT) instructions, for example, require three machine cycles: a FETCH, to obtain the instruction; a MEMORY READ, to obtain the address of the object peripheral; and an INPUT or an OUTPUT machine cycle, to complete the transfer.

While no one instruction cycle will consist of more then five machine cycles, the following ten different types of machine cycles may occur within an instruction cycle:

(1) FETCH (M1)
(2) MEMORY READ
(3) MEMORY WRITE
(4) STACK READ
(5) STACK WRITE
(6) INPUT
(7) OUTPUT
(8) INTERRUPT
(9) HALT
(10) HALT • INTERRUPT

The machine cycles that actually do occur in a particular instruction cycle depend upon the kind of instruction, with the overriding stipulation that the first machine cycle in any instruction cycle is always a FETCH.

The processor identifies the machine cycle in progress by transmitting an eight-bit status word during the first state of every machine cycle. Updated status information is presented on the 8080's data lines (D_0-D_7), during the SYNC interval. This data should be saved in latches, and used to develop control signals for external circuitry. Table 2-1 shows how the positive-true status information is distributed on the processor's data bus.

Status signals are provided principally for the control of external circuitry. Simplicity of interface, rather than machine cycle identification, dictates the logical definition of individual status bits. You will therefore observe that certain processor machine cycles are uniquely identified by a single status bit, but that others are not. The M_1 status bit (D_6), for example, unambiguously identifies a FETCH machine cycle. A STACK READ, on the other hand, is indicated by the coincidence of STACK and MEMR signals. Machine cycle identification data is also valuable in the test and de-bugging phases of system development. Table 2-1 lists the status bit outputs for each type of machine cycle.

State Transition Sequence:

Every machine cycle within an instruction cycle consists of three to five active states (referred to as T_1, T_2, T_3, T_4, T_5 or T_W). The actual number of states depends upon the instruction being executed, and on the particular machine cycle within the greater instruction cycle. The state transition diagram in Figure 2-4 shows how the 8080 proceeds from state to state in the course of a machine cycle. The diagram also shows how the READY, HOLD, and INTERRUPT lines are sampled during the machine cycle, and how the conditions on these lines may modify the basic transition sequence. In the present discussion, we are concerned only with the basic sequence and with the READY function. The HOLD and INTERRUPT functions will be discussed later.

The 8080 CPU does not directly indicate its internal state by transmitting a "state control" output during each state; instead, the 8080 supplies direct control output (INTE, HLDA, DBIN, $\overline{WR}$ and WAIT) for use by external circuitry.

Recall that the 8080 passes through at least three states in every machine cycle, with each state defined by successive low-to-high transitions of the ϕ_1 clock. Figure 2-5 shows the timing relationships in a typical FETCH machine cycle. Events that occur in each state are referenced to transitions of the ϕ_1 and ϕ_2 clock pulses.

The SYNC signal identifies the first state (T_1) in every machine cycle. As shown in Figure 2-5, the SYNC signal is related to the leading edge of the ϕ_2 clock. There is a delay (t_{DC}) between the low-to-high transition of ϕ_2 and the positive-going edge of the SYNC pulse. There also is a corresponding delay (also t_{DC}) between the next ϕ_2 pulse and the falling edge of the SYNC signal. Status information is displayed on D_0-D_7 during the same ϕ_2 to ϕ_2 interval. Switching of the status signals is likewise controlled by ϕ_2.

The rising edge of ϕ_2 during T_1 also loads the processor's address lines (A_0-A15). These lines become stable within a brief delay (t_{DA}) of the ϕ_2 clocking pulse, and they remain stable until the first ϕ_2 pulse after state T_3. This gives the processor ample time to read the data returned from memory.

Once the processor has sent an address to memory, there is an opportunity for the memory to request a WAIT. This it does by pulling the processor's READY line low, prior to the "Ready set-up" interval (t_{RS}) which occurs during the ϕ_2 pulse within state T_2 or T_W. As long as the READY line remains low, the processor will idle, giving the memory time to respond to the addressed data request. Refer to Figure 2-5.

The processor responds to a wait request by entering an alternative state (T_W) at the end of T_2, rather than proceeding directly to the T_3 state. Entry into the T_W state is indicated by a WAIT signal from the processor, acknowledging the memory's request. A low-to-high transition on the WAIT line is triggered by the rising edge of the ϕ_1 clock and occurs within a brief delay (t_{DC}) of the actual entry into the T_W state.

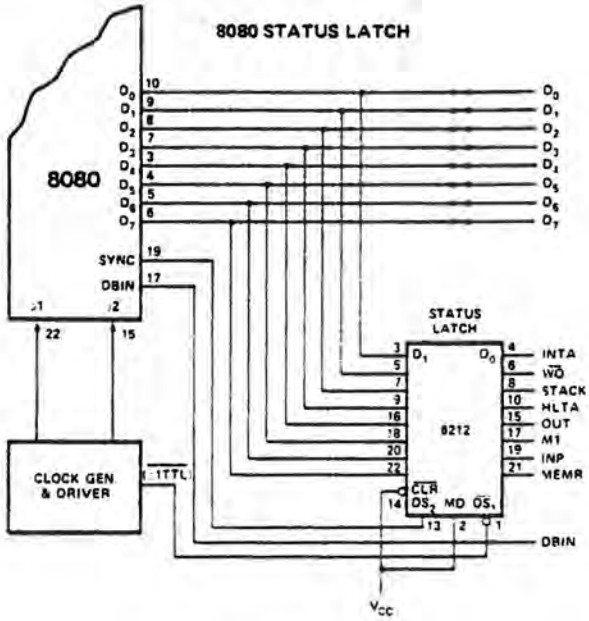
A wait period may be of indefinite duration. The processor remains in the waiting condition until its READY line again goes high. A READY indication **must** precede the falling edge of the ϕ_2 clock by a specified interval (t_{RS}), in order to guarantee an exit from the T_W state. The cycle may then proceed, beginning with the rising edge of the next ϕ_1 clock. A WAIT interval will therefore consist of an integral number of T_W states and will always be a multiple of the clock period.

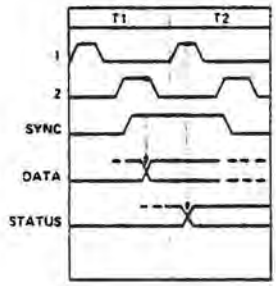
Instructions for the 8080 require from one to five machine cycles for complete execution. The 8080 sends out 8 bit of status information on the data bus at the beginning of each machine cycle (during SYNC time). The following table defines the status information.

STATUS INFORMATION DEFINITION

Symbols	Data Bus Bit	Definition
INTA*	D_0	Acknowledge signal for INTERRUPT request. Signal should be used to gate a restart instruction onto the data bus when DBIN is active.
$\overline{WO}$	D_1	Indicates that the operation in the current machine cycle will be a WRITE memory or OUTPUT function ($\overline{WO}$ = 0). Otherwise, a READ memory or INPUT operation will be executed.
STACK	D_2	Indicates that the address bus holds the pushdown stack address from the Stack Pointer.
HLTA	D_3	Acknowledge signal for HALT instruction.
OUT	D_4	Indicates that the address bus contains the address of an output device and the data bus will contain the output data when $\overline{WR}$ is active.
M_1	D_5	Provides a signal to indicate that the CPU is in the fetch cycle for the first byte of an instruction.
INP*	D_6	Indicates that the address bus contains the address of an input device and the input data should be placed on the data bus when DBIN is active.
MEMR*	D_7	Designates that the data bus will be used for memory read data.

*These three status bits can be used to control the flow of data onto the 8080 data bus.





STATUS WORD CHART

TYPE OF MACHINE CYCLE

DATA BUS BIT	STATUS INFORMATION	INSTRUCTION FETCH	MEMORY READ	MEMORY WRITE	STACK READ	STACK WRITE	INPUT READ	OUTPUT WRITE	INTERRUPT ACKNOWLEDGE	HALT ACKNOWLEDGE	INTERRUPT ACKNOWLEDGE WHILE HALT
		①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩
D_0	INTA	0	0	0	0	0	0	0	1	0	1
D_1	$\overline{WO}$	1	1	0	1	0	1	0	1	1	1
D_2	STACK	0	0	0	1	1	0	0	0	0	0
D_3	HLTA	0	0	0	0	0	0	0	0	1	1
D_4	OUT	0	0	0	0	0	0	1	0	0	0
D_5	M_1	1	0	0	0	0	0	0	1	0	1
D_6	INP	0	0	0	0	0	1	0	0	0	0
D_7	MEMR	1	1	0	1	0	0	0	0	1	0

Ⓝ STATUS WORD

Table 2-1. 8080 Status Bit Definitions

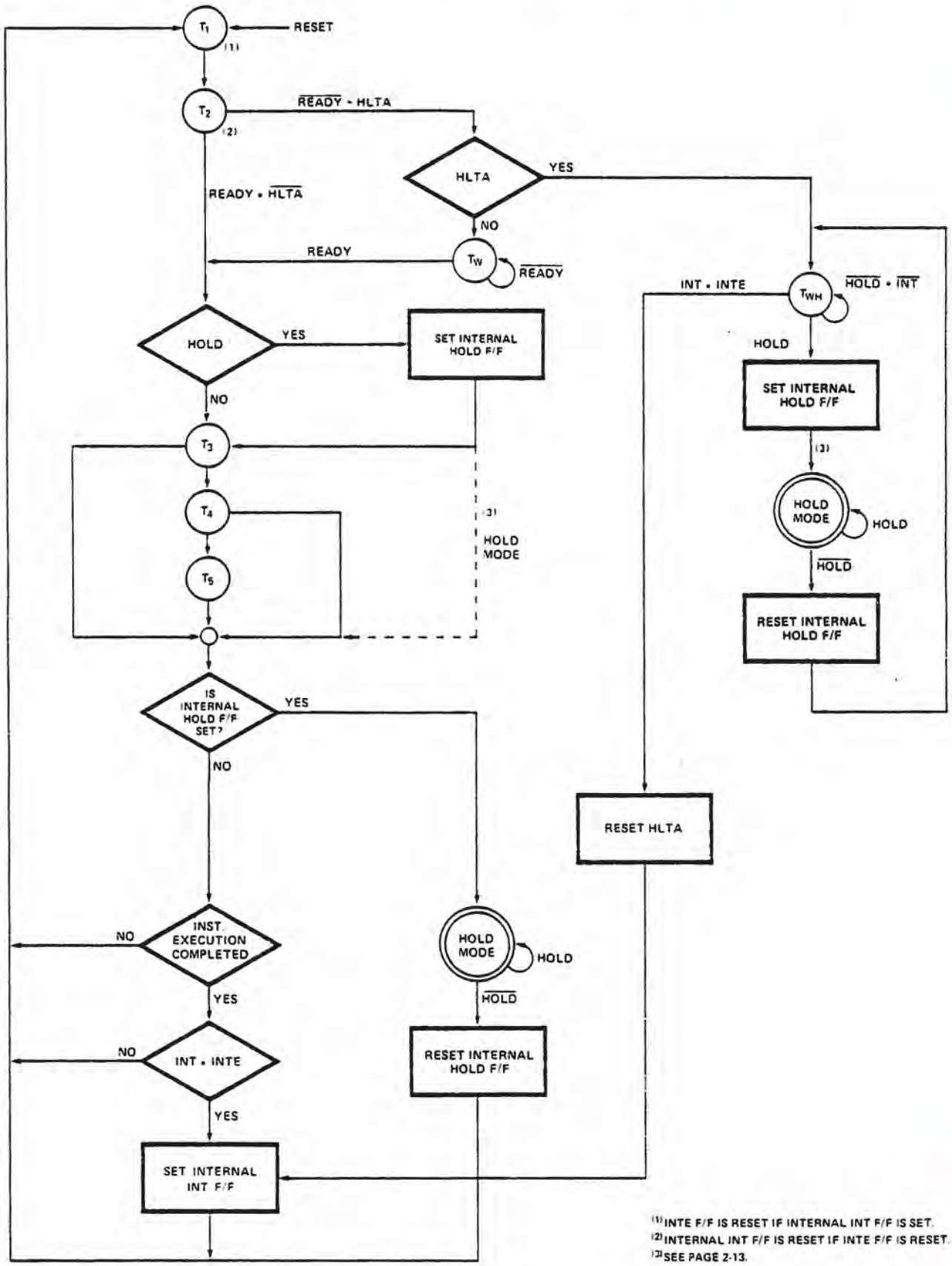


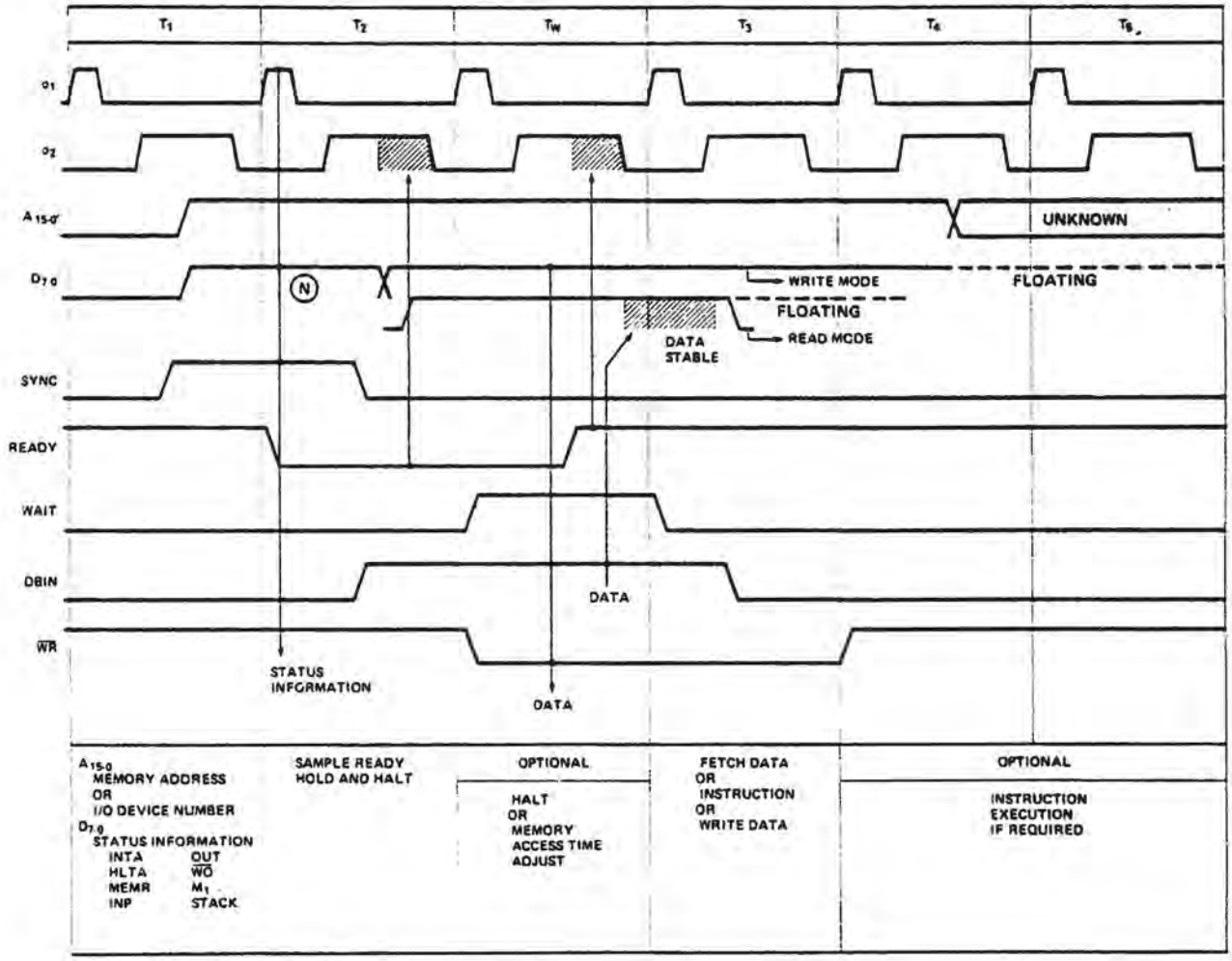
Figure 2-4. CPU State Transition Diagram

The events that take place during the T_3 state are determined by the kind of machine cycle in progress. In a FETCH machine cycle, the processor interprets the data on its data bus as an instruction. During a MEMORY READ or a STACK READ, data on this bus is interpreted as a data word. The processor outputs data on this bus during a MEMORY WRITE machine cycle. During I/O operations, the processor may either transmit or receive data, depending on whether an OUTPUT or an INPUT operation is involved.

Figure 2-6 illustrates the timing that is characteristic of a data input operation. As shown, the low-to-high transition of ϕ_2 during T_2 clears status information from the processor's data lines, preparing these lines for the receipt of incoming data. The data presented to the processor must have stabilized prior to both the "ϕ_1–data set-up" interval (t_{DS1}), that precedes the falling edge of the ϕ_1 pulse defining state T_3, and the "ϕ_2–data set-up" interval (t_{DS2}), that precedes the rising edge of ϕ_2 in state T_3. This same data must remain stable during the "data hold" interval (t_{DH}) that occurs following the rising edge of the ϕ_2 pulse. Data placed on these lines by memory or by other external devices will be sampled during T_3.

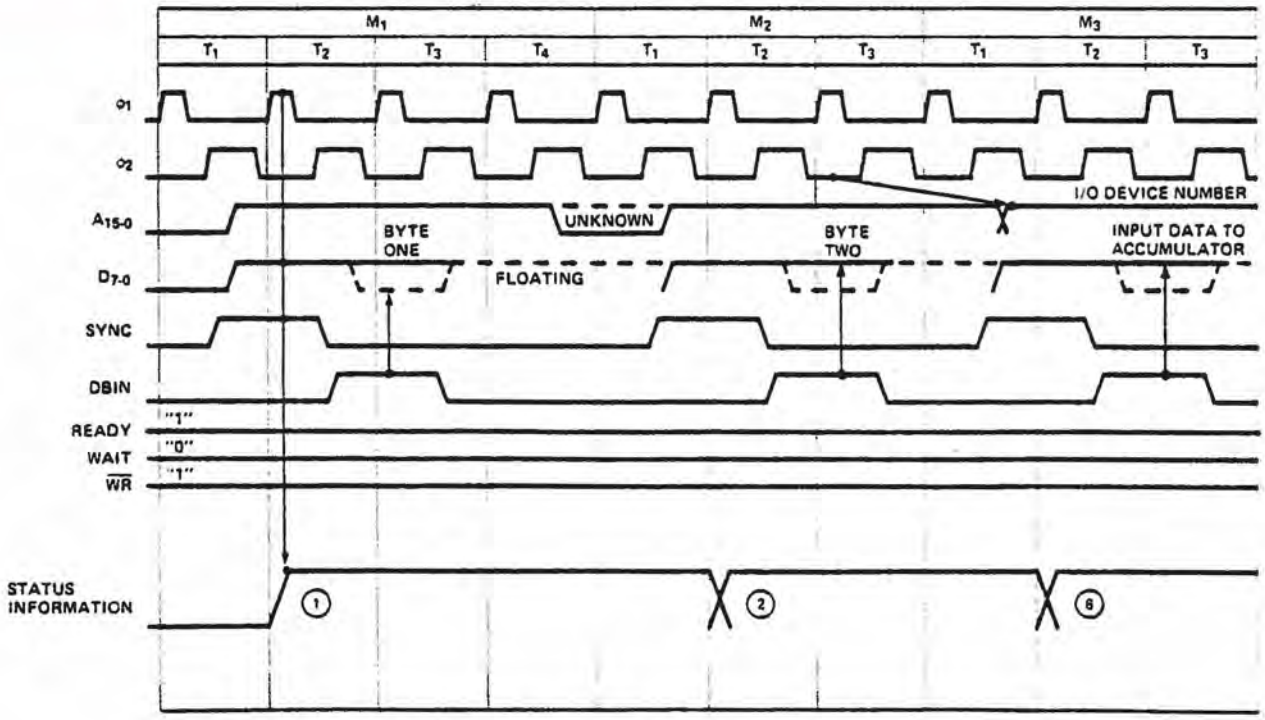
During the input of data to the processor, the 8080 generates a DBIN signal which should be used externally to enable the transfer. Machine cycles in which DBIN is available include: FETCH, MEMORY READ, STACK READ, and INTERRUPT. DBIN is initiated by the rising edge of ϕ_2 during state T2 and terminated by the corresponding edge of ϕ_2 during T_3. Any T_W phases intervening between T_2 and T_3 will therefore extend DBIN by one or more clock periods.

Figure 2-7 shows the timing of a machine cycle in which the processor outputs data. Output data may be destined either for memory or for peripherals. The rising edge of ϕ_2 within state T_2 clears status information from the CPU's data lines, and loads in the data which is to be output to external devices. This substitution takes place within the



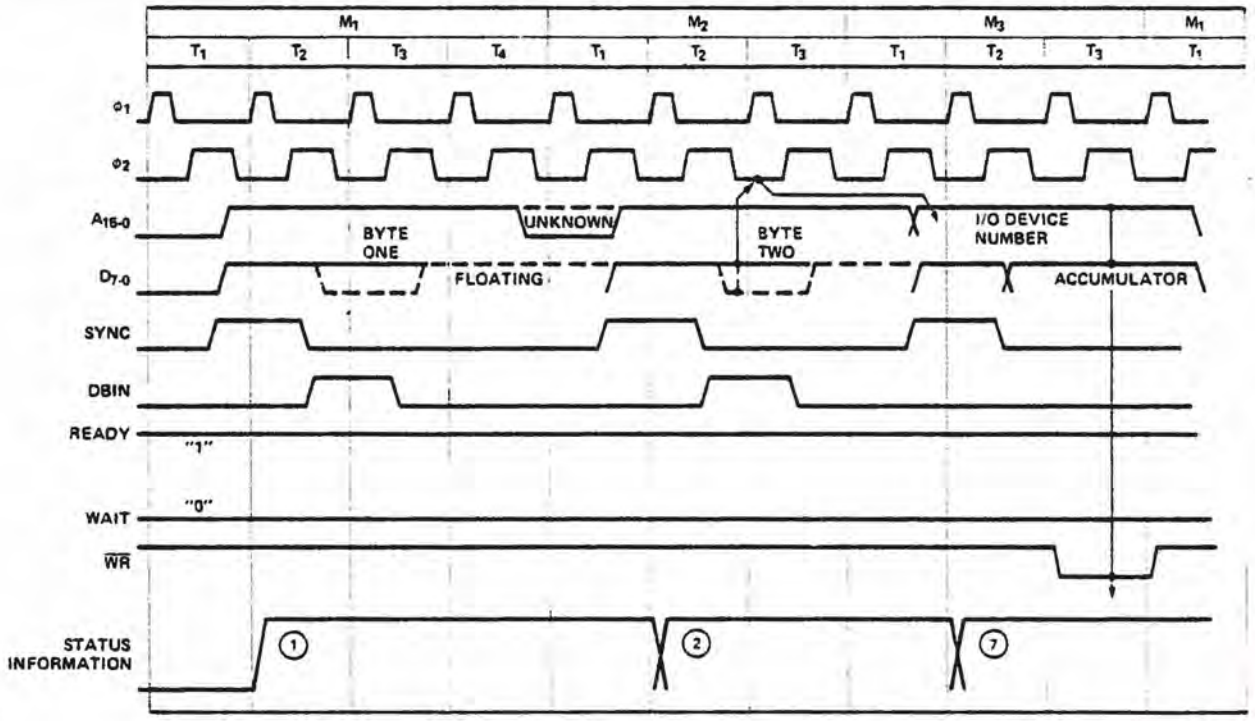
NOTE: (N) Refer to Status Word Chart on Page 2-6.

Figure 2-5. Basic 8080 Instruction Cycle



NOTE: (N) Refer to Status Word Chart on Page 2-6.

Figure 2-6. Input Instruction Cycle



NOTE: (N) Refer to Status Word Chart on Page 2-6.

Figure 2-7. Output Instruction Cycle

"data output delay" interval (t_{DD}) following the ϕ_2 clock's leading edge. Data on the bus remains stable throughout the remainder of the machine cycle, until replaced by updated status information in the subsequent T_1 state. Observe that a READY signal is necessary for completion of an OUTPUT machine cycle. Unless such an indication is present, the processor enters the T_W state, following the T_2 state. Data on the output lines remains stable in the interim, and the processing cycle will not proceed until the READY line again goes high.

The 8080 CPU generates a $\overline{WR}$ output for the synchronization of external transfers, during those machine cycles in which the processor outputs data. These include MEMORY WRITE, STACK WRITE, and OUTPUT. The negative-going leading edge of $\overline{WR}$ is referenced to the rising edge of the first ϕ_1 clock pulse following T_2, and occurs within a brief delay (t_{DC}) of that event. $\overline{WR}$ remains low until re-triggered by the leading edge of ϕ_1 during the state following T_3. Note that any T_W states intervening between T_2 and T_3 of the output machine cycle will necessarily extend $\overline{WR}$, in much the same way that DBIN is affected during data input operations.

All processor machine cycles consist of at least three states: T_1, T_2, and T_3 as just described. If the processor has to wait for a response from the peripheral or memory with which it is communicating, then the machine cycle may also contain one or more T_W states. During the three basic states, data is transferred to or from the processor.

After the T_3 state, however, it becomes difficult to generalize. T_4 and T_5 states are available, if the execution of a particular instruction requires them. But not all machine cycles make use of these states. It depends upon the kind of instruction being executed, and on the particular machine cycle within the instruction cycle. The processor will terminate any machine cycle as soon as its processing activities are completed, rather than proceeding through the T_4 and T_5 states every time. Thus the 8080 may exit a machine cycle following the T_3, the T_4, or the T_5 state and proceed directly to the T_1 state of the next machine cycle.

STATE	ASSOCIATED ACTIVITIES
T_1	A memory address or I/O device number is placed on the Address Bus (A_{15-0}); status information is placed on Data Bus (D_{7-0}).
T_2	The CPU samples the READY and HOLD inputs and checks for halt instruction.
TW (optional)	Processor enters wait state if READY is low or if HALT instruction has been executed.
T3	An instruction byte (FETCH machine cycle), data byte (MEMORY READ, STACK READ) or interrupt instruction (INTERRUPT machine cycle) is input to the CPU from the Data Bus; or a data byte (MEMORY WRITE, STACK WRITE or OUTPUT machine cycle) is output onto the data bus.
T4 T5 (optional)	States T_4 and T_5 are available if the execution of a particular instruction requires them; if not, the CPU may skip one or both of them. T_4 and T_5 are only used for internal processor operations.

Table 2-2. State Definitions

INTERRUPT SEQUENCES

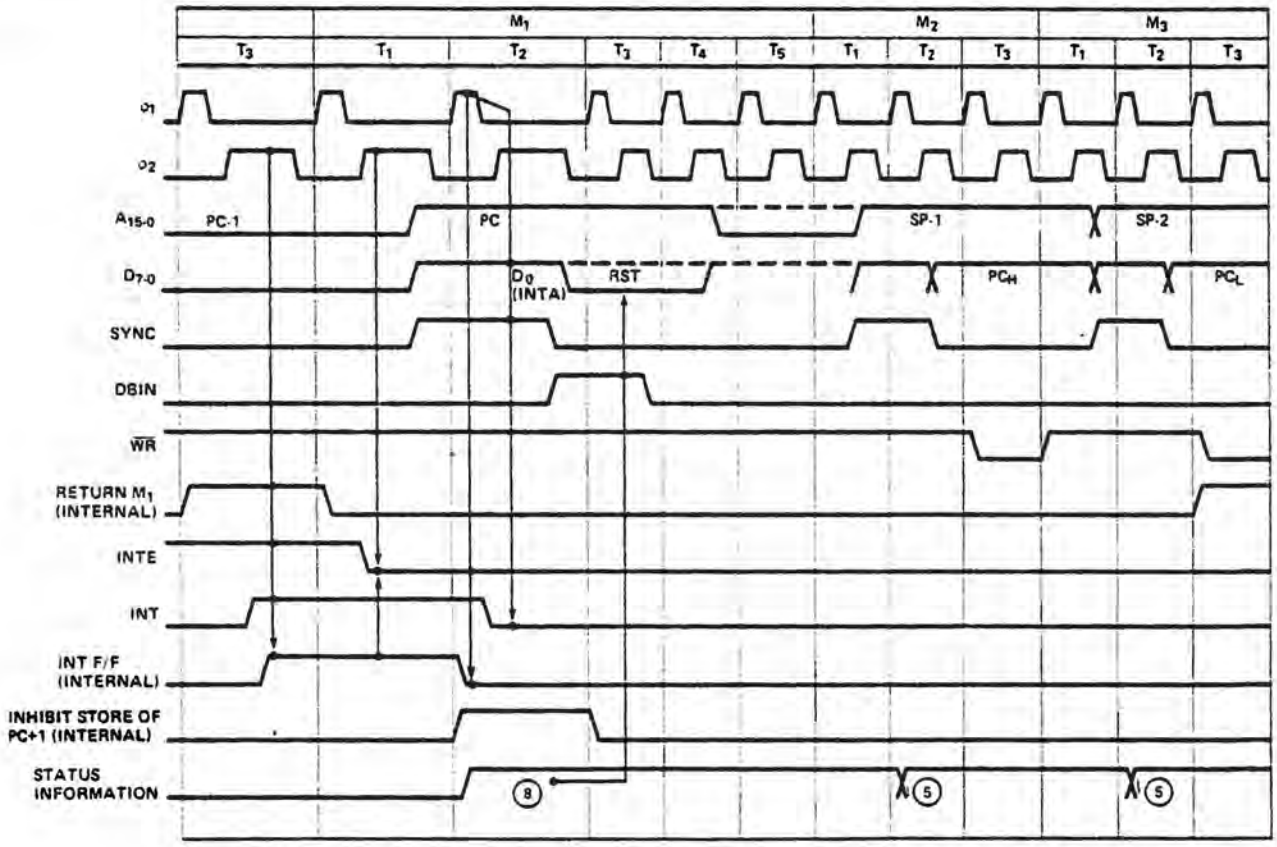
The 8080 has the built-in capacity to handle external interrupt requests. A peripheral device can initiate an interrupt simply by driving the processor's interrupt (INT) line high.

The interrupt (INT) input is asynchronous, and a request may therefore originate at any time during any instruction cycle. Internal logic re-clocks the external request, so that a proper correspondence with the driving clock is established. As Figure 2-8 shows, an interrupt request (INT) arriving during the time that the interrupt enable line (INTE) is high, acts in coincidence with the ϕ_2 clock to set the internal interrupt latch. This event takes place during the last state of the instruction cycle in which the request occurs, thus ensuring that any instruction in progress is completed before the interrupt can be processed.

The INTERRUPT machine cycle which follows the arrival of an enabled interrupt request resembles an ordinary FETCH machine cycle in most respects. The M_1 status bit is transmitted as usual during the SYNC interval. It is accompanied, however, by an INTA status bit (D_0) which acknowledges the external request. The contents of the program counter are latched onto the CPU's address lines during T_1, but the counter itself is not incremented during the INTERRUPT machine cycle, as it otherwise would be. In this way, the pre-interrupt status of the program counter is preserved, so that data in the counter may be restored by the interrupted program after the interrupt request has been processed.

The interrupt cycle is otherwise indistinguishable from an ordinary FETCH machine cycle. The processor itself takes no further special action. It is the responsibility of the peripheral logic to see that an eight-bit interrupt instruction is "jammed" onto the processor's data bus during state T_3. In a typical system, this means that the data-in bus from memory must be temporarily disconnected from the processor's main data bus, so that the interrupting device can command the main bus without interference.

The 8080's instruction set provides a special one-byte call which facilitates the processing of interrupts (the ordinary program Call takes three bytes). This is the RESTART instruction (RST). A variable three-bit field embedded in the eight-bit field of the RST enables the interrupting device to direct a Call to one of eight fixed memory locations. The decimal addresses of these dedicated locations are: 0, 8, 16, 24, 32, 40, 48, and 56. Any of these addresses may be used to store the first instruction(s) of a routine designed to service the requirements of an interrupting device. Since the (RST) is a call, completion of the instruction also stores the old program counter contents on the STACK.



NOTE: (N) Refer to Status Word Chart on Page 2-6.

Figure 2-8. Interrupt Timing

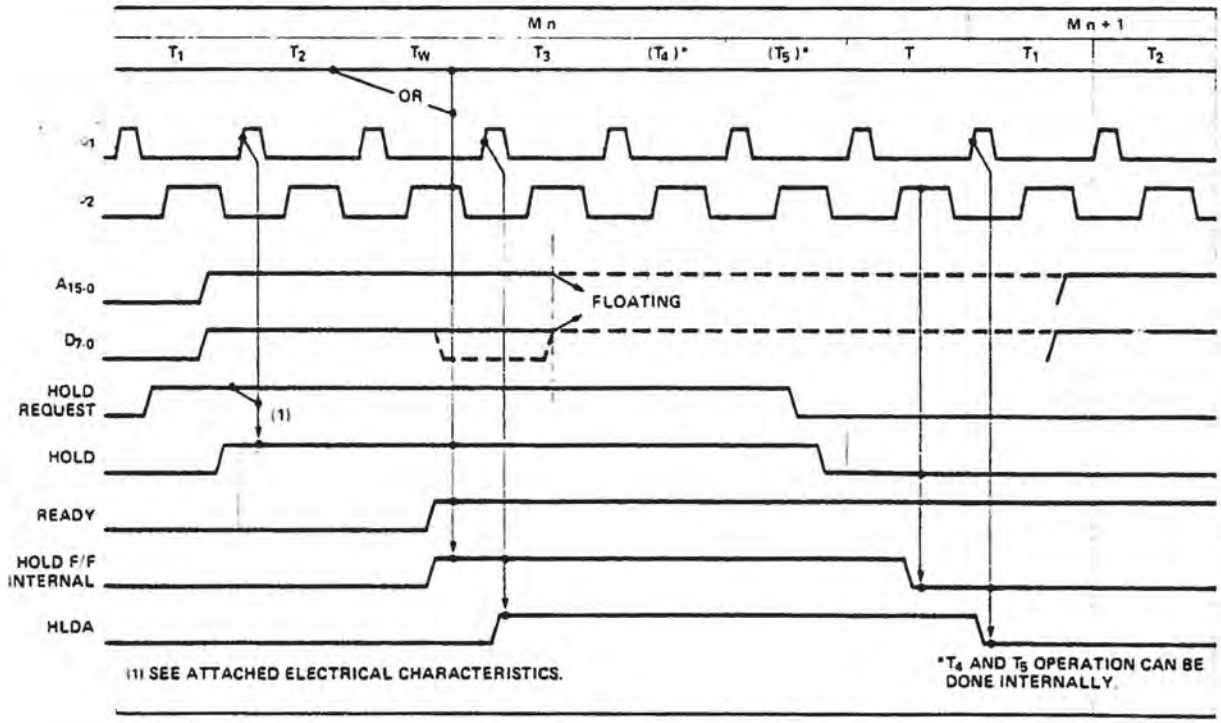


Figure 2-9. HOLD Operation (Read Mode)

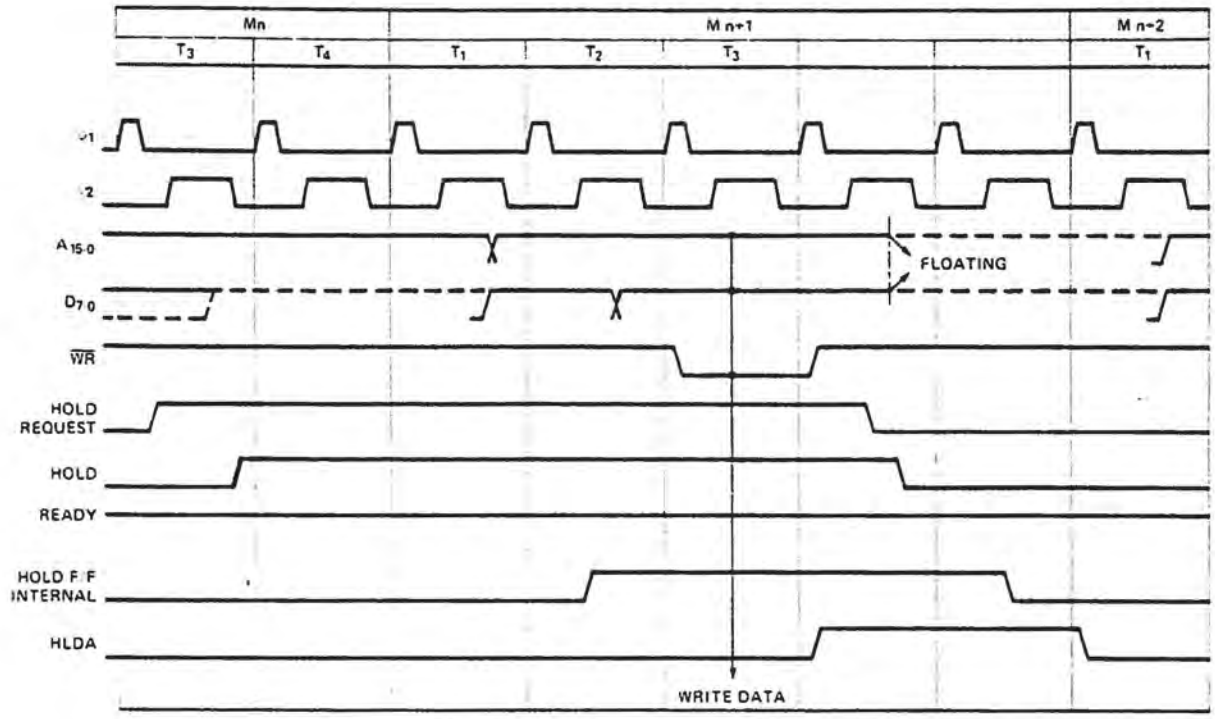


Figure 2-10. HOLD Operation (Write Mode)

HOLD SEQUENCES

The 8080A CPU contains provisions for Direct Memory Access (DMA) operations. By applying a HOLD to the appropriate control pin on the processor, an external device can cause the CPU to suspend its normal operations and relinquish control of the address and data busses. The processor responds to a request of this kind by floating its address to other devices sharing the busses. At the same time, the processor acknowledges the HOLD by placing a high on its HLDA outpin pin. During an acknowledged HOLD, the address and data busses are under control of the peripheral which originated the request, enabling it to conduct memory transfers without processor intervention.

Like the interrupt, the HOLD input is synchronized internally. A HOLD signal must be stable prior to the "Hold set-up" interval (t_{HS}), that precedes the rising edge of ϕ_2.

Figures 2-9 and 2-10 illustrate the timing involved in HOLD operations. Note the delay between the asynchronous HOLD REQUEST and the re-clocked HOLD. As shown in the diagram, a coincidence of the READY, the HOLD, and the ϕ_2 clocks sets the internal hold latch. Setting the latch enables the subsequent rising edge of the ϕ_1 clock pulse to trigger the HLDA output.

Acknowledgement of the HOLD REQUEST precedes slightly the actual floating of the processor's address and data lines. The processor acknowledges a HOLD at the beginning of T_3, if a read or an input machine cycle is in progress (see Figure 2-9). Otherwise, acknowledgement is deferred until the beginning of the state following T_3 (see Figure 2-10). In both cases, however, the HLDA goes high within a specified delay (t_{DC}) of the rising edge of the selected ϕ_1 clock pulse. Address and data lines are floated within a brief delay after the rising edge of the next ϕ_2 clock pulse. This relationship is also shown in the diagrams.

To all outward appearances, the processor has suspended its operations once the address and data busses are floated. Internally, however, certain functions may continue. If a HOLD REQUEST is acknowledged at T_3, and if the processor is in the middle of a machine cycle which requires four or more states to complete, the CPU proceeds through T_4 and T_5 before coming to a rest. Not until the end of the machine cycle is reached will processing activities cease. Internal processing is thus permitted to overlap the external DMA transfer, improving both the efficiency and the speed of the entire system.

The processor exits the holding state through a sequence similar to that by which it entered. A HOLD REQUEST is terminated asynchronously when the external device has completed its data transfer. The HLDA output returns to a low level following the leading edge of the next $\phi 1$ clock pulse. Normal processing resumes with the machine cycle following the last cycle that was executed.

HALT SEQUENCES

When a halt instruction (HLT) is executed, the CPU enters the halt state (T_{WH}) after state T_2 of the next machine cycle, as shown in Figure 2-11. There are only three ways in which the 8080 can exit the halt state:

- A high on the RESET line will always reset the 8080 to state T_1; RESET also clears the program counter.
- A HOLD input will cause the 8080 to enter the hold state, as previously described. When the HOLD line goes low, the 8080 re-enters the halt state on the rising edge of the next ϕ_1 clock pulse.
- An interrupt (i.e., INT goes high while INTE is enabled) will cause the 8080 to exit the Halt state and enter state T_1 on the rising edge of the next ϕ_1 clock pulse. NOTE: The interrupt enable (INTE) flag **must** be set when the halt state is entered; otherwise, the 8080 will only be able to exit via a RESET signal.

Figure 2-12 illustrates halt sequencing in flow chart form.

START-UP OF THE 8080 CPU

When power is applied initially to the 8080, the processor begins operating immediately. The contents of its program counter, stack pointer, and the other working registers are naturally subject to random factors and cannot be specified. For this reason, it will be necessary to begin the power-up sequence with RESET.

An external RESET signal of three clock period duration (minimum) restores the processor's internal program counter to zero. Program execution thus begins with memory location zero, following a RESET. Systems which require the processor to wait for an explicit start-up signal will store a halt instruction (EI, HLT) in the first two locations. A manual or an automatic INTERRUPT will be used for starting. In other systems, the processor may begin executing its stored program immediately. Note, however, that the RESET has no effect on status flags, or on any of the processor's working registers (accumulator, registers, or stack pointer). The contents of these registers remain indeterminate, until initialized explicitly by the program.

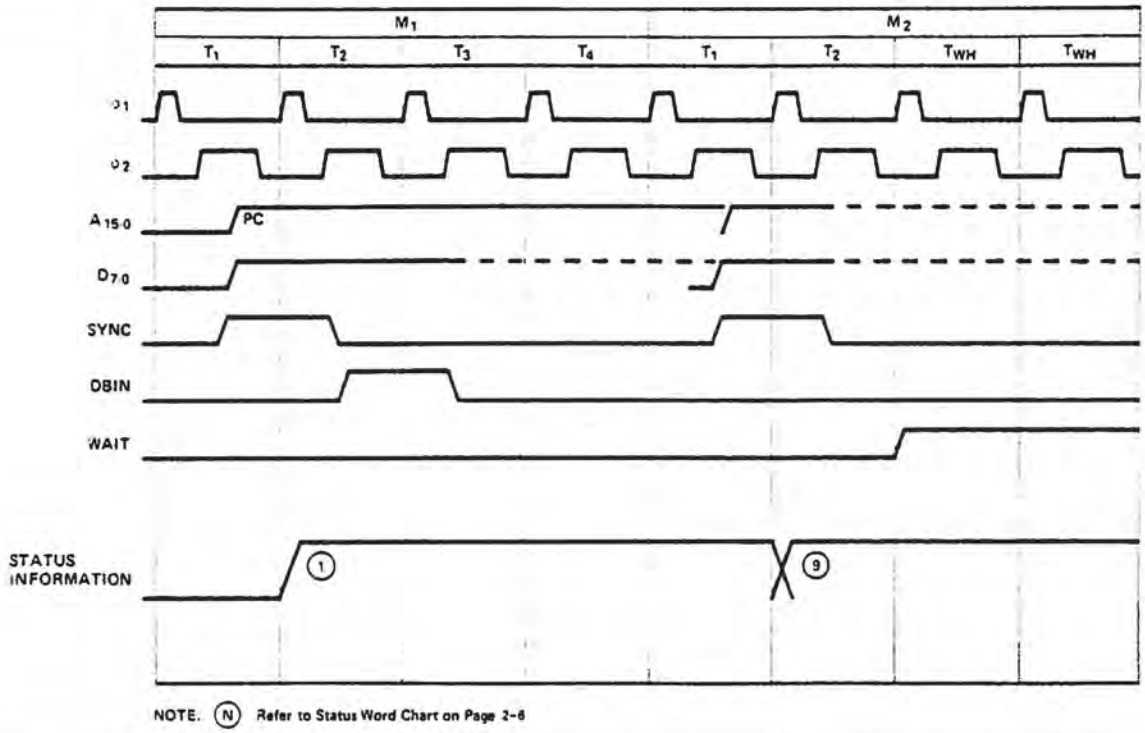


Figure 2-11. HALT Timing

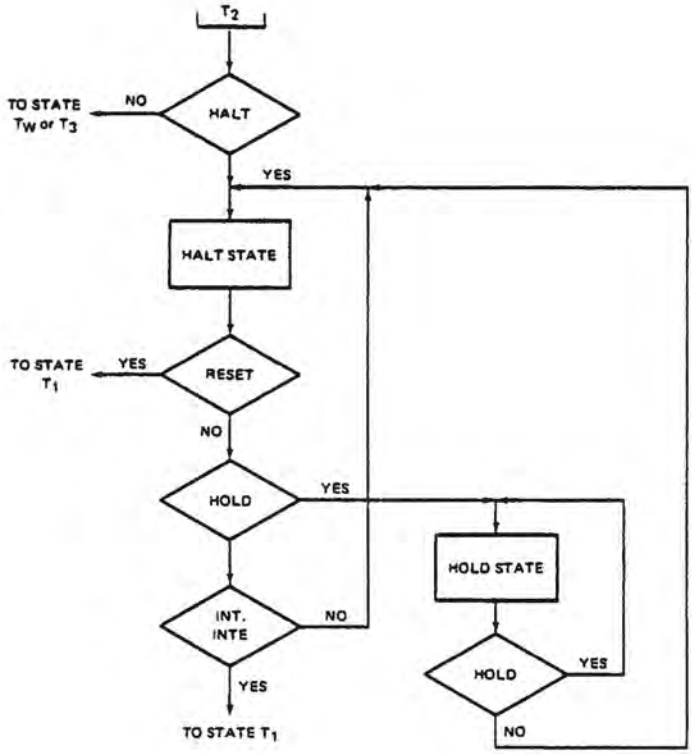
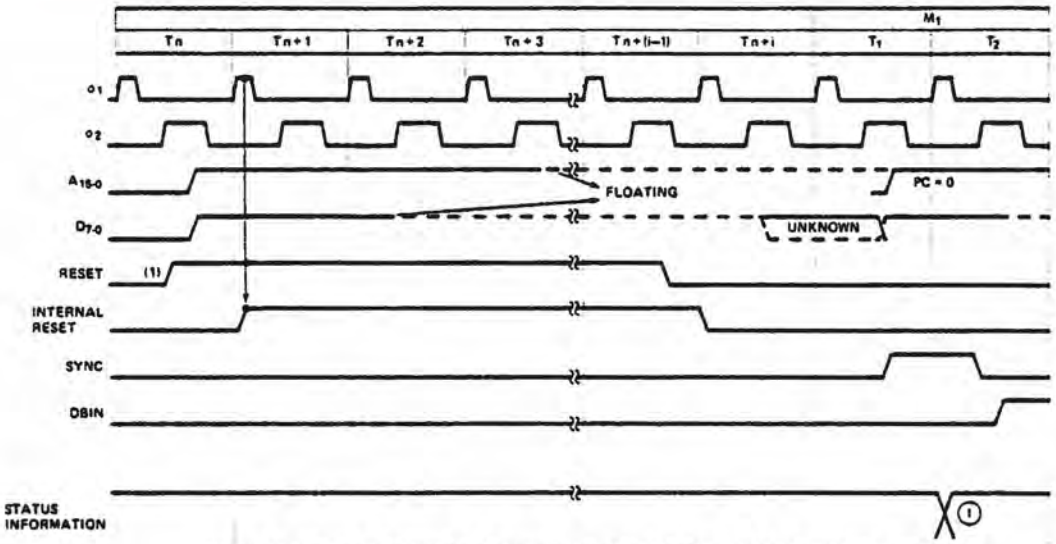


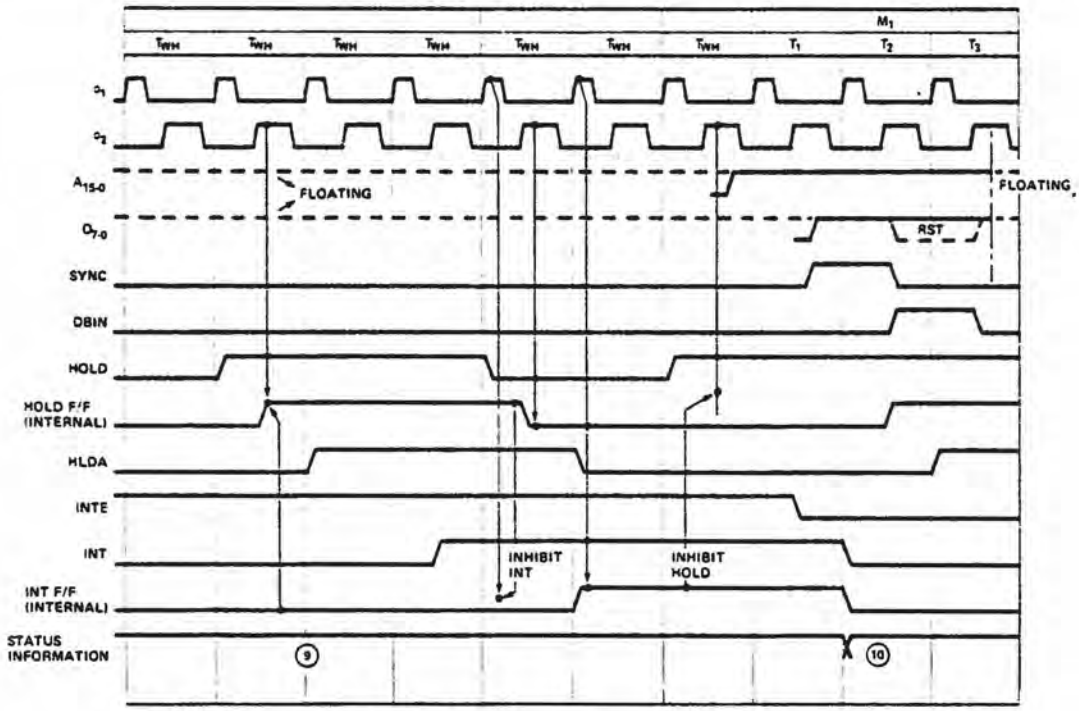
Figure 2-12. HALT Sequence Flow Chart.



(1) WHEN RESET SIGNAL IS ACTIVE, ALL OF CONTROL OUTPUT SIGNALS WILL BE RESET IMMEDIATELY OR SOME CLOCK PERIODS LATER. THE RESET SIGNAL MUST BE ACTIVE FOR A MINIMUM OF THREE CLOCK CYCLES. IN THE ABOVE DIAGRAM N AND I MAY BE ANY INTEGER.

NOTE: (N) Refer to Status Word Chart on Page 2-6.

Figure 2-13. Reset.



NOTE: (N) Refer to Status Word Chart on Page 2-6.

Figure 2-14. Relation between HOLD and INT in the HALT State.

MNEMONIC	OP CODE		M1[1]					M2		
	$D_7 D_6 D_5 D_4$	$D_3 D_2 D_1 D_0$	T1	T2[2]	T3	T4	T5	T1	T2[2]	T3
MOV r1, r2	0 1 D D	D S S S	PC OUT STATUS	PC = PC +1	INST→TMP/IR	(SSS)→TMP	(TMP)→DDD			
MOV r, M	0 1 D D	D 1 1 0				X[3]		HL OUT STATUS[6]	DATA→DDD	
MOV M, r	0 1 1 1	0 S S S				(SSS)→TMP		HL OUT STATUS[7]	(TMP)→DATA BUS	
SPHL	1 1 1 1	1 0 0 1				(HL) →	SP			
MVI r, data	0 0 D D	D 1 1 0				X		PC OUT STATUS[6]	B2→DDDD	
MVI M, data	0 0 1 1	0 1 1 0				X			B2→TMP	
LXI rp, data	0 0 R P	0 0 0 1				X			PC = PC + 1 B2→r1	
LDA addr	0 0 1 1	1 0 1 0				X			PC = PC + 1 B2→Z	
STA addr	0 0 1 1	0 0 1 0				X			PC = PC + 1 B2→Z	
LHLD addr	0 0 1 0	1 0 1 0				X			PC = PC + 1 B2→Z	
SHLD addr	0 0 1 0	0 0 1 0				X		PC OUT STATUS[6]	PC = PC + 1 B2→Z	
LDAX rp[4]	0 0 R P	1 0 1 0				X		rp OUT STATUS[6]	DATA→A	
STAX rp[4]	0 0 R P	0 0 1 0				X		rp OUT STATUS[7]	(A)→DATA BUS	
XCHG	1 1 1 0	1 0 1 1				(HL)⟷(DE)				
ADD r	1 0 0 0	0 S S S				(SSS)→TMP (A)→ACT		[9]	(ACT)+(TMP)→A	
ADD M	1 0 0 0	0 1 1 0				(A)→ACT		HL OUT STATUS[6]	DATA→TMP	
ADI data	1 1 0 0	0 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2→TMP	
ADC r	1 0 0 0	1 S S S				(SSS)→TMP (A)→ACT		[9]	(ACT)+(TMP)+CY→A	
ADC M	1 0 0 0	1 1 1 0				(A)→ACT		HL OUT STATUS[6]	DATA→TMP	
ACI data	1 1 0 0	1 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2→TMP	
SUB r	1 0 0 1	0 S S S				(SSS)→TMP (A)→ACT		[9]	(ACT)-(TMP)→A	
SUB M	1 0 0 1	0 1 1 0				(A)→ACT		HL OUT STATUS[6]	DATA→TMP	
SUI data	1 1 0 1	0 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2→TMP	
SBB r	1 0 0 1	1 S S S				(SSS)→TMP (A)→ACT		[9]	(ACT)-(TMP)-CY→A	
SBB M	1 0 0 1	1 1 1 0				(A)→ACT		HL OUT STATUS[6]	DATA→TMP	
SBI data	1 1 0 1	1 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2→TMP	
INR r	0 0 D D	D 1 0 0				(DDD)→TMP (TMP) + 1→ALU	ALU→DDD			
INR M	0 0 1 1	0 1 0 0				X		HL OUT STATUS[6]	DATA→TMP (TMP)+1→ALU	
DCR r	0 0 D D	D 1 0 1				(DDD)→TMP (TMP)+1→ALU	ALU→DDD			
DCR M	0 0 1 1	0 1 0 1				X		HL OUT STATUS[6]	DATA→TMP (TMP)-1→ALU	
INX rp	0 0 R P	0 0 1 1				(RP) + 1 →	RP			
DCX rp	0 0 R P	1 0 1 1				(RP) - 1 →	RP			
DAD rp[8]	0 0 R P	1 0 0 1				X		(rl)→ACT	(L)→TMP. (ACT)+(TMP)→ALU	ALU→L, CY
DAA	0 0 1 0	0 1 1 1				DAA→A, FLAGS[10]				
ANA r	1 0 1 0	0 S S S				(SSS)→TMP (A)→ACT		[9]	(ACT)+(TMP)→A	
ANA M	1 0 1 0	0 1 1 0	PC OUT STATUS	PC = PC + 1	INST→TMP/IR	(A)→ACT		HL OUT STATUS[6]	DATA→TMP	

M3			M4			M5				
T1	T2[2]	T3	T1	T2[2]	T3	T1	T2[2]	T3	T4	T5
HL OUT STATUS[7]	(TMP) →	DATA BUS								
PC OUT STATUS[6]	PC = PC + 1 B3 →	rh								
	PC = PC + 1 B3 →	W	WZ OUT STATUS[6]	DATA →	A					
	PC = PC + 1 B3 →	W	WZ OUT STATUS[7]	(A) →	DATA BUS					
	PC = PC + 1 B3 →	W	WZ OUT STATUS[6]	DATA → WZ = WZ + 1	L	WZ OUT STATUS[6]	DATA →	H		
PC OUT STATUS[6]	PC = PC + 1 B3 →	W	WZ OUT STATUS[7]	(L) → WZ = WZ + 1	DATA BUS	WZ OUT STATUS[7]	(H) →	DATA BUS		
[9]	(ACT)+(TMP)→A									
[9]	(ACT)+(TMP)→A									
[9]	(ACT)+(TMP)+CY→A									
[9]	(ACT)+(TMP)+CY→A									
[9]	(ACT)-(TMP)→A									
[9]	(ACT)-(TMP)→A									
[9]	(ACT)-(TMP)-CY→A									
[9]	(ACT)-(TMP)-CY→A									
HL OUT STATUS[7]	ALU →	DATA BUS								
HL OUT STATUS[7]	ALU →	DATA BUS								
(rh)→ACT	(H)→TMP (ACT)+(TMP)+CY→ALU	ALU→H, CY								
[9]	(ACT)+(TMP)→A									

MNEMONIC	OP CODE		M1[1]					M2		
	$D_7 D_6 D_5 D_4$	$D_3 D_2 D_1 D_0$	T1	T2[2]	T3	T4	T5	T1	T2[2]	T3
ANI data	1 1 1 0	0 1 1 0	PC OUT STATUS	PC = PC + 1	INST→TMP/IR	(A)→ACT		PC OUT STATUS[6]	PC = PC + 1	B2 → TMP
XRA r	1 0 1 0	1 S S S				(A)→ACT (SSS)→TMP		[9]	(ACT)+(TPM)→A	
XRA M	1 0 1 0	1 1 1 0				(A)→ACT		HL OUT STATUS[6]		DATA → TMP
XRI data	1 1 1 0	1 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1	B2 → TMP
ORA r	1 0 1 1	0 S S S				(A)→ACT (SSS)→TMP		[9]	(ACT)+(TMP)→A	
ORA M	1 0 1 1	0 1 1 0				(A)→ACT		HL OUT STATUS[6]		DATA → TMP
ORI data	1 1 1 1	0 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1	B2 → TMP
CMP r	1 0 1 1	1 S S S				(A)→ACT (SSS)→TMP		[9]	(ACT)-(TMP), FLAGS	
CMP M	1 0 1 1	1 1 1 0				(A)→ACT		HL OUT STATUS[6]		DATA → TMP
CPI data	1 1 1 1	1 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1	B2 → TMP
RLC	0 0 0 0	0 1 1 1				(A)→ALU ROTATE		[9]	ALU→A, CY	
RRC	0 0 0 0	1 1 1 1				(A)→ALU ROTATE		[9]	ALU→A, CY	
RAL	0 0 0 1	0 1 1 1				(A), CY→ALU ROTATE		[9]	ALU→A, CY	
RAR	0 0 0 1	1 1 1 1				(A), CY→ALU ROTATE		[9]	ALU→A, CY	
CMA	0 0 1 0	1 1 1 1				$(\bar{A})$→A				
CMC	0 0 1 1	1 1 1 1				$\overline{CY}$→CY				
STC	0 0 1 1	0 1 1 1				1→CY				
JMP addr	1 1 0 0	0 0 1 1				X		PC OUT STATUS[6]	PC = PC + 1	B2 → Z
J cond addr[17]	1 1 C C	C 0 1 0				JUDGE CONDITION		PC OUT STATUS[6]	PC = PC + 1	B2 → Z
CALL addr	1 1 0 0	1 1 0 1				SP = SP - 1		PC OUT STATUS[6]	PC = PC + 1	B2 → Z
C cond addr[17]	1 1 C C	C 1 0 0				JUDGE CONDITION IF TRUE, SP = SP - 1		PC OUT STATUS[6]	PC = PC + 1	B2 → Z
RET	1 1 0 0	1 0 0 1				X		SP OUT STATUS[15]	SP = SP + 1	DATA → Z
R cond addr[17]	1 1 C C	C 0 0 0			INST→TMP/IR	JUDGE CONDITION[14]		SP OUT STATUS[15]	SP = SP + 1	DATA → Z
RST n	1 1 N N	N 1 1 1			ϕ→W INST→TMP/IR	SP = SP - 1		SP OUT STATUS[16]	SP = SP - 1	(PCH) → DATA BUS
PCHL	1 1 1 0	1 0 0 1			INST→TMP/IR	(HL) → PC				
PUSH rp	1 1 R P	0 1 0 1				SP = SP - 1		SP OUT STATUS[16]	SP = SP - 1	(rh) → DATA BUS
PUSH PSW	1 1 1 1	0 1 0 1				SP = SP - 1		SP OUT STATUS[16]	SP = SP - 1	(A) → DATA BUS
POP rp	1 1 R P	0 0 0 1				X		SP OUT STATUS[15]	SP = SP + 1	DATA → r1
POP PSW	1 1 1 1	0 0 0 1				X		SP OUT STATUS[15]	SP = SP + 1	DATA → FLAGS
XTHL	1 1 1 0	0 0 1 1				X		SP OUT STATUS[15]	SP = SP + 1	DATA → Z
IN port	1 1 0 1	1 0 1 1				X		PC OUT STATUS[6]	PC = PC + 1	B2 → Z, W
OUT port	1 1 0 1	0 0 1 1				X		PC OUT STATUS[6]	PC = PC + 1	B2 → Z, W
EI	1 1 1 1	1 0 1 1				SET INTE F/F				
DI	1 1 1 1	0 0 1 1				RESET INTE F/F				
HLT	0 1 1 1	0 1 1 0				X		PC OUT STATUS	HALT MODE[20]	
NOP	0 0 0 0	0 0 0 0	PC OUT STATUS	PC = PC + 1	INST→TMP/IR	X				

M3			M4			M5						
T1	T2[2]	T3	T1	T2[2]	T3	T1	T2[2]	T3	T4	T5		
[9]	(ACT)+(TMP)→A											
[9]	(ACT)+(TMP)→A											
[9]	(ACT)+(TMP)→A											
[9]	(ACT)+(TMP)→A											
[9]	(ACT)+(TMP)→A											
[9]	(ACT)-(TMP): FLAGS											
[9]	(ACT)-(TMP): FLAGS											
PC OUT STATUS[6]	PC = PC + 1	B3 → W									WZ OUT STATUS[11]	(WZ) + 1 → PC
PC OUT STATUS[6]	PC = PC + 1	B3 → W									WZ OUT STATUS[11,12]	(WZ) + 1 → PC
PC OUT STATUS[6]	PC = PC + 1	B3 → W	SP OUT STATUS[16]	(PCH) → DATA BUS SP = SP - 1		SP OUT STATUS[16]	(PCL) → DATA BUS				WZ OUT STATUS[11]	(WZ) + 1 → PC
PC OUT STATUS[6]	PC = PC + 1	B3 → W[13]	SP OUT STATUS[16]	(PCH) → DATA BUS SP = SP - 1		SP OUT STATUS[16]	(PCL) → DATA BUS				WZ OUT STATUS[11,12]	(WZ) + 1 → PC
SP OUT STATUS[15]	SP = SP + 1	DATA → W									WZ OUT STATUS[11]	(WZ) + 1 → PC
SP OUT STATUS[15]	SP = SP + 1	DATA → W									WZ OUT STATUS[11,12]	(WZ) + 1 → PC
SP OUT STATUS[16]	(TMP = 00NNN000) → Z (PCL) → DATA BUS										WZ OUT STATUS[11]	(WZ) + 1 → PC
SP OUT STATUS[16]	(r) → DATA BUS											
SP OUT STATUS[16]	FLAGS → DATA BUS											
SP OUT STATUS[15]	SP = SP + 1	DATA → rh										
SP OUT STATUS[15]	SP = SP + 1	DATA → A										
SP OUT STATUS[15]	DATA → W		SP OUT STATUS[16]	(H) → DATA BUS		SP OUT STATUS[16]	(L) → DATA BUS		(WZ) → HL			
WZ OUT STATUS[18]	DATA → A											
WZ OUT STATUS[18]	(A) → DATA BUS											

NOTES:

1. The first memory cycle (M1) is always an instruction fetch; the first (or only) byte, containing the op code, is fetched during this cycle.

2. If the READY input from memory is not high during T2 of each memory cycle, the processor will enter a wait state (TW) until READY is sampled as high.

3. States T4 and T5 are present, as required, for operations which are completely internal to the CPU. The contents of the internal bus during T4 and T5 are available at the data bus; this is designed for testing purposes only. An "X" denotes that the state is present, but is only used for such internal operations as instruction decoding.

4. Only register pairs rp = B (registers B and C) or rp = D (registers D and E) may be specified.

5. These states are skipped.

6. Memory read sub-cycles; an instruction or data word will be read.

7. Memory write sub-cycle.

8. The READY signal is not required during the second and third sub-cycles (M2 and M3). The HOLD signal is accepted during M2 and M3. The SYNC signal is not generated during M2 and M3. During the execution of DAD, M2 and M3 are required for an internal register-pair add; memory is not referenced.

9. The results of these arithmetic, logical or rotate instructions are not moved into the accumulator (A) until state T2 of the next instruction cycle. That is, A is loaded while the next instruction is being fetched; this overlapping of operations allows for faster processing.

10. If the value of the least significant 4-bits of the accumulator is greater than 9 or if the auxiliary carry bit is set, 6 is added to the accumulator. If the value of the most significant 4-bits of the accumulator is now greater than 9, or if the carry bit is set, 6 is added to the most significant 4-bits of the accumulator.

11. This represents the first sub-cycle (the instruction fetch) of the next instruction cycle.

12. If the condition was met, the contents of the register pair WZ are output on the address lines (A_{0-15}) instead of the contents of the program counter (PC).

13. If the condition was not met, sub-cycles M4 and M5 are skipped; the processor instead proceeds immediately to the instruction fetch (M1) of the next instruction cycle.

14. If the condition was not met, sub-cycles M2 and M3 are skipped; the processor instead proceeds immediately to the instruction fetch (M1) of the next instruction cycle.

15. Stack read sub-cycle.

16. Stack write sub-cycle.

17.

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

18. I/O sub-cycle: the I/O port's 8-bit select code is duplicated on address lines 0-7 (A_{0-7}) and 8-15 (A_{8-15}).

19. Output sub-cycle.

20. The processor will remain idle in the halt state until an interrupt, a reset or a hold is accepted. When a hold request is accepted, the CPU enters the hold mode; after the hold mode is terminated, the processor returns to the halt state. After a reset is accepted, the processor begins execution at memory location zero. After an interrupt is accepted, the processor executes the instruction forced onto the data bus (usually a restart instruction).

SSS or DDD	Value	rp	Value
A	111	B	00
B	000	D	01
C	001	H	10
D	010	SP	11
E	011		
H	100		
L	101		

CHAPTER 3 INTERFACING THE 8080

This chapter will illustrate, in detail, how to interface the 8080 CPU with Memory and I/O. It will also show the benefits and tradeoffs encountered when using a variety of system architectures to achieve higher throughput, decreased component count or minimization of memory size.

8080 Microcomputer system design lends itself to a simple, modular approach. Such an approach will yield the designer a reliable, high performance system that contains a minimum component count and is easy to manufacture and maintain.

The overall system can be thought of as a simple block diagram. The three (3) blocks in the diagram represent the functions common to any computer system.

CPU Module* Contains the Central Processing Unit, system timing and interface circuitry to Memory and I/O devices.

Memory Contains Read Only Memory (ROM) and Read/Write Memory (RAM) for program and data storage.

I/O Contains circuitry that allows the computer system to communicate with devices or structures existing outside of the CPU or Memory array.

for example: Keyboards, Floppy Disks, Paper Tape, etc.

There are three busses that interconnect these blocks:

Data Bus† A bi-directional path on which data can flow between the CPU and Memory or I/O.

Address Bus A uni-directional group of lines that identify a particular Memory location or I/O device.

Control Bus A uni-directional set of signals that indicate the type of activity in current process.

Type of activities:
1. Memory Read
2. Memory Write
3. I/O Read
4. I/O Write
5. Interrupt Acknowledge

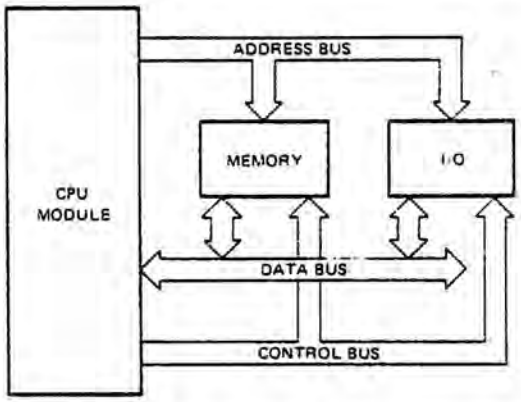


Figure 3-1. Typical Computer System Block Diagram

Basic System Operation

1. The CPU Module issues an activity command on the Control Bus.
2. The CPU Module issues a binary code on the Address Bus to identify which particular Memory location or I/O device will be involved in the current process activity.
3. The CPU Module receives or transmits data with the selected Memory location or I/O device.
4. The CPU Module returns to (1) and issues the next activity command.

It is easy to see at this point that the CPU module is the central element in any computer system.

*"Module" refers to a functional block, it does not reference a printed circuit board manufactured by INTEL.

†"Bus" refers to a set of signals grouped together because of the similarity of their functions.

The following pages will cover the detailed design of the CPU Module with the 8080. The three Busses (Data, Address and Control) will be developed and the interconnection to Memory and I/O will be shown.

Design philosophies and system architectures presented in this manual are consistent with product development programs underway at INTEL for the MCS-80™. Thus, the designer who uses this manual as a guide for his total system engineering is assured that all new developments in components and software for MCS-80 from INTEL will be compatible with his design approach.

CPU Module Design

The CPU Module contains three major areas:

1. The 8080 Central Processing Unit
2. A Clock Generator and High Level Driver
3. A bi-directional Data Bus Driver and System Control Logic

The following will discuss the design of the three major areas contained in the CPU Module. This design is presented as an alternative to the Intel® 8224 Clock Generator and Intel 8228 System Controller. By studying the alternative approach, the designer can more clearly see the considerations involved in the specification and engineering of the 8224 and 8228. Standard TTL components and Intel general purpose peripheral devices are used to implement the design and to achieve operational characteristics that are as close as possible to those of the 8224 and 8228. Many auxiliary timing functions and features of the 8224 and 8228 are too complex to practically implement in standard components, so only the basic functions of the 8224 and 8228 are generated. Since significant benefits in system timing and component count reduction can be realized by using the 8224 and 8228, this is the preferred method of implementation.

1. 8080 CPU

The operation of the 8080 CPU was covered in previous chapters of this manual, so little reference will be made to it in the design of the Module.

2. Clock Generator and High Level Driver

The 8080 is a dynamic device, meaning that its internal storage elements and logic circuitry require a timing reference (Clock), supplied by external circuitry, to refresh and provide timing control signals.

The 8080 requires two (2) such Clocks. Their waveforms must be non-overlapping, and comply with the timing and levels specified in the 8080 A.C. and D.C. Characteristics, page 5-15.

Clock Generator Design

The Clock Generator consists of a crystal controlled,

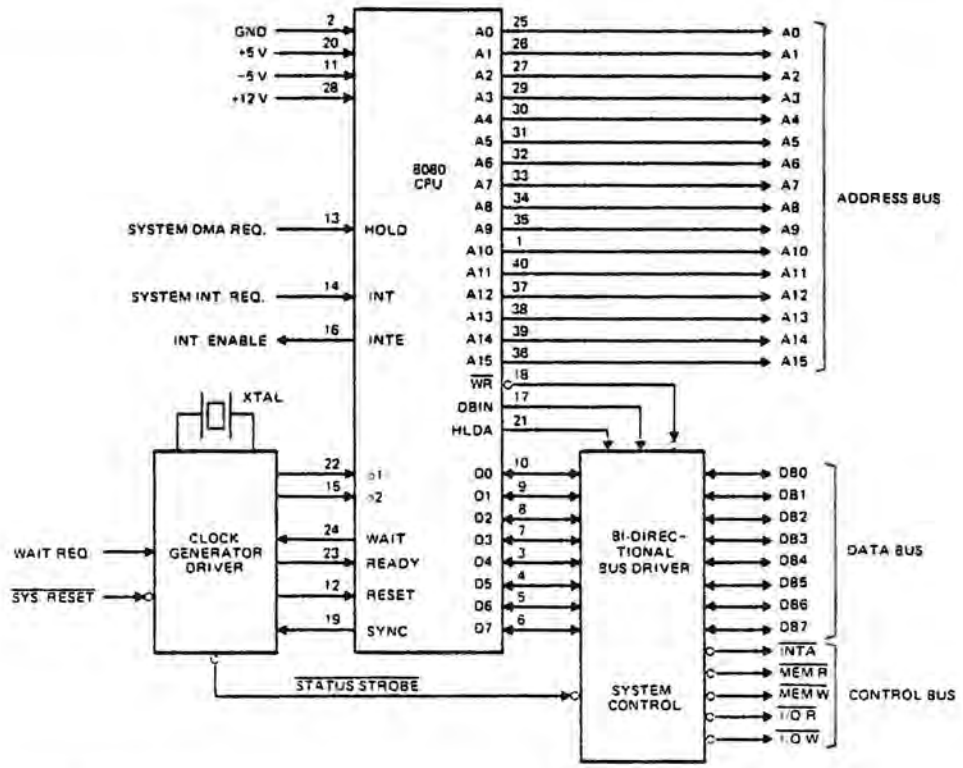


Figure 3-2. 8080 CPU Interface

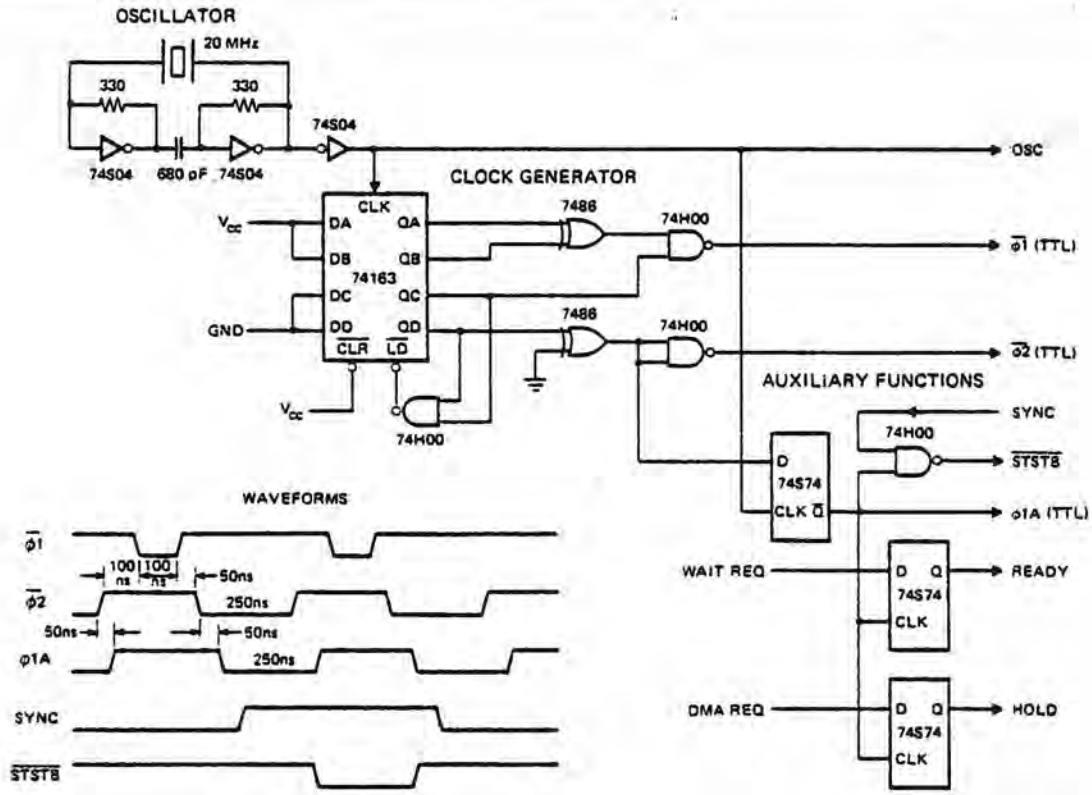


Figure 3-3. 8080 Clock Generator

20 MHZ oscillator, a four bit counter, and gating circuits.

The oscillator provides a 20 MHZ signal to the input of a four (4) bit, presettable, synchronous, binary counter. By presetting the counter as shown in figure 3-3 and clocking it with the 20 MHZ signal, a simple decoding of the counters outputs using standard TTL gates, provides proper timing for the two (2) 8080 clock inputs.

Note that the timing must actually be measured at the output of the High Level Driver to take into account the added delays and waveform distortions within such a device.

High Level Driver Design

The voltage level of the clocks for the 8080 is not TTL compatible like the other signals that input to the 8080. The voltage swing is from .6 volts (V_{ILC}) to 11 volts (V_{IHC}) with risetimes and falltimes under 50 ns. The Capacitive Drive is 20 pf (max.). Thus, a High Level Driver is required to interface the outputs of the Clock Generator (TTL) to the 8080.

The two (2) outputs of the Clock Generator are capacitivity coupled to a dual- High Level clock driver. The driver must be capable of complying with the 8080 clock input specifications, page 5-15. A driver of this type usually has little problem supplying the positive transition when biased from the 8080 V_{DD} supply (12V) but to achieve the low voltage specification (V_{ILC}) .8 volts Max. the driver is biased to the 8080 V_{BB} supply (-5V). This allows the driver to swing from GND to V_{DD} with the aid of a simple resistor divider.

A low resistance series network is added between the driver and the 8080 to eliminate any overshoot of the pulsed waveforms. Now a circuit is apparent that can easily comply with the 8080 specifications. In fact rise and falltimes of this design are typically less than 10 ns.

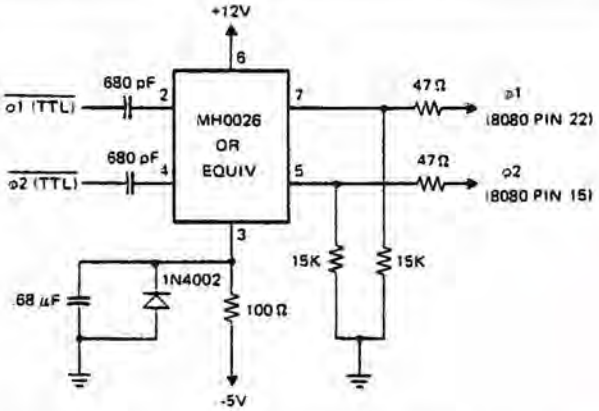


Figure 3-4. High Level Driver

Auxiliary Timing Signals and Functions

The Clock Generator can also be used to provide other signals that the designer can use to simplify large system timing or the interface to dynamic memories.

Functions such as power-on reset, synchronization of external requests (HOLD, READY, etc.) and single step, could easily be added to the Clock Generator to further enhance its capabilities.

For instance, the 20 MHZ signal from the oscillator can be buffered so that it could provide the basis for communication baud rate generation.

The Clock Generator diagram also shows how to generate an advanced timing signal (ϕ1A) that is handy to use in clocking "D" type flipflops to synchronize external requests. It can also be used to generate a strobe ($\overline{STSTB}$) that is the latching signal for the status information which is available on the Data Bus at the beginning of each machine cycle. A simple gating of the SYNC signal from the 8080 and the advanced (ϕ1A) will do the job. See Figure 3-3.

3. Bi-Directional Bus Driver and System Control Logic

The system Memory and I/O devices communicate with the CPU over the bi-directional Data Bus. The system Control Bus is used to gate data on and off the Data Bus within the proper timing sequences as dictated by the operation of the 8080 CPU. The data lines of the 8080 CPU, Memory and I/O devices are 3-state in nature, that is, their output drivers have the ability to be forced into a high-impedance mode and are, effectively, removed from the circuit. This 3-state bus technique allows the designer to construct a system around a single, eight (8) bit parallel, bi-directional Data Bus and simply gate the information on or off this bus by selecting or deselecting (3-stating) Memory and I/O devices with signals from the Control Bus.

Bi-Directional Data Bus Driver Design

The 8080 Data Bus (D7-D0) has two (2) major areas of concern for the designer:

1. Input Voltage level (V_{IH}) 3.3 volts minimum.
2. Output Drive Capability (I_{OL}) 1.7 mA maximum.

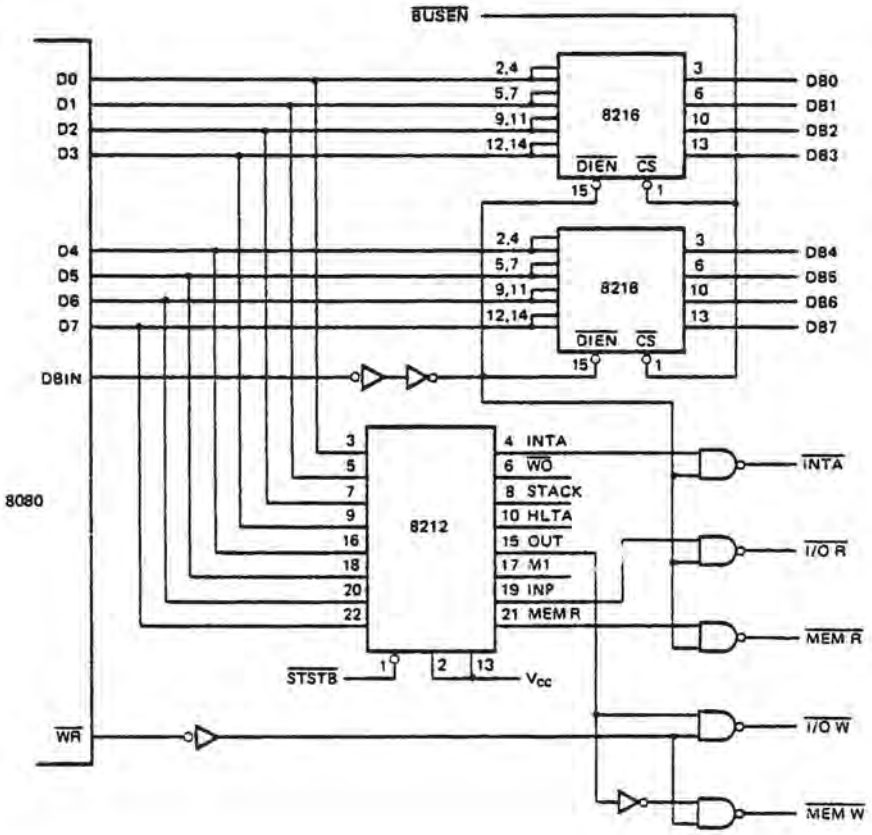


Figure 3-5. 8080 System Control

The input level specification implies that any semiconductor memory or I/O device connected to the 8080 Data Bus must be able to provide a minimum of 3.3 volts in its high state. Most semiconductor memories and standard TTL I/O devices have an output capability of between 2.0 and 2.8 volts, obviously a direct connection onto the 8080 Data Bus would require pullup resistors, whose value should not affect the bus speed or stress the drive capability of the memory or I/O components.

The 8080A output drive capability (I_{OL}) 1.9mA max. is sufficient for small systems where Memory size and I/O requirements are minimal and the entire system is contained on a single printed circuit board. Most systems however, take advantage of the high-performance computing power of the 8080 CPU and thus a more typical system would require some form of buffering on the 8080 Data Bus to support a larger array of Memory and I/O devices which are likely to be on separate boards.

A device specifically designed to do this buffering function is the INTEL® 8216, a (4) four bit bi-directional bus driver whose input voltage level is compatible with standard TTL devices and semiconductor memory components, and has output drive capability of 50 mA. At the 8080 side, the 8216 has a "high" output of 3.65 volts that not only meets the 8080 input spec but provides the designer with a worse case 350 mV noise margin.

A pair of 8216's are connected directly to the 8080 Data Bus (D7-D0) as shown in figure 3-5. Note that the DBIN signal from the 8080 is connected to the direction control input ($\overline{DIEN}$) so the correct flow of data on the bus is maintained. The chip select ($\overline{CS}$) of the 8216 is connected to BUS ENABLE ($\overline{BUSEN}$) to allow for DMA activities by deselecting the Data Bus Buffer and forcing the outputs of the 8216's into their high impedance (3-state) mode. This allows other devices to gain access to the data bus (DMA).

System Control Logic Design

The Control Bus maintains discipline of the bi-directional Data Bus, that is, it determines what type of device will have access to the bus (Memory or I/O) and generates signals to assure that these devices transfer Data with the 8080 CPU within the proper timing "windows" as dictated by the CPU operational characteristics.

As described previously, the 8080 issues Status information at the beginning of each Machine Cycle on its Data Bus to indicate what operation will take place during that cycle. A simple (8) bit latch, like an INTEL® 8212, connected directly to the 8080 Data Bus (D7-D0) as shown in figure 3-5 will store the Status information. The signal that loads the data into the Status Latch comes from the Clock Generator, it is Status Strobe ($\overline{STSTB}$) and occurs at the start of each Machine Cycle.

Note that the Status Latch is connected onto the 8080 Data Bus (D7-D0) before the Bus Buffer. This is to maintain the integrity of the Data Bus and simplify Control Bus timing in DMA dependent environments.

As shown in the diagram, a simple gating of the outputs of the Status Latch with the DBIN and $\overline{WR}$ signals from the 8080 generate the (4) four Control signals that make up the basic Control Bus.

These four signals: 1. Memory Read ($\overline{MEM\ R}$)

2. Memory Write ($\overline{MEM\ W}$)

3. I/O Read ($\overline{I/O\ R}$)

4. I/O Write ($\overline{I/O\ W}$)

connect directly to the MCS™-80 component "family" of ROMs, RAMs and I/O devices.

A fifth signal, Interrupt Acknowledge ($\overline{INTA}$) is added to the Control Bus by gating data off the Status Latch with the DBIN signal from the 8080 CPU. This signal is used to enable the Interrupt Instruction Port which holds the RST instruction onto the Data Bus.

Other signals that are part of the Control Bus such as $\overline{WO}$, Stack and M1 are present to aid in the testing of the System and also to simplify interfacing the CPU to dynamic memories or very large systems that require several levels of bus buffering.

Address Buffer Design

The Address Bus (A15-A0) of the 8080, like the Data Bus, is sufficient to support a small system that has a moderate size Memory and I/O structure, confined to a single card. To expand the size of the system that the Address Bus can support a simple buffer can be added, as shown in figure 3-6. The INTEL® 8212 or 8216 is an excellent device for this function. They provide low input loading (.25 mA), high output drive and insert a minimal delay in the System Timing.

Note that BUS ENABLE ($\overline{BUSEN}$) is connected to the buffers so that they are forced into their high-impedance (3-state) mode during DMA activities so that other devices can gain access to the Address Bus.

INTERFACING THE 8080 CPU TO MEMORY AND I/O DEVICES

The 8080 interfaces with standard semiconductor Memory components and I/O devices. In the previous text the proper control signals and buffering were developed which will produce a simple bus system similar to the basic system example shown at the beginning of this chapter.

In Figure 3-6 a simple, but exact 8080 typical system is shown that can be used as a guide for any 8080 system, regardless of size or complexity. It is a "three bus" architecture, using the signals developed in the CPU module.

Note that Memory and I/O devices interface in the same manner and that their isolation is only a function of the definition of the Read-Write signals on the Control Bus. This allows the 8080 system to be configured so that Memory and I/O are treated as a single array (memory mapped I/O) for small systems that require high thruput and have less than 32K memory size. This approach will be brought out later in the chapter.

ROM INTERFACE

A ROM is a device that stores data in the form of Program or other information such as "look-up tables" and is only read from, thus the term Read Only Memory. This type of memory is generally non-volatile, meaning that when the power is removed the information is retained. This feature eliminates the need for extra equipment like tape readers and disks to load programs initially, an important aspect in small system design.

Interfacing standard ROMs, such as the devices shown in the diagram is simple and direct. The output Data lines are connected to the bi-directional Data Bus, the Address inputs tie to the Address bus with possible decoding of the most significant bits as "chip selects" and the $\overline{MEMR}$ signal from the Control Bus connected to a "chip select" or data buffer. Basically, the CPU issues an address during the first portion of an instruction or data fetch (T1 & T2). This value on the Address Bus selects a specific location within the ROM, then depending on the ROM's delay (access time) the data stored at the addressed location is present at the Data output lines. At this time (T3) the CPU Data Bus is in the "input Mode" and the control logic issues a Memory Read command ($\overline{MEMR}$) that gates the addressed data on to the Data Bus.

RAM INTERFACE

A RAM is a device that stores data. This data can be program, active "look-up tables," temporary values or external stacks. The difference between RAM and ROM is that data can be written into such devices and are in essence, Read/Write storage elements. RAMs do not hold their data when power is removed so in the case where Program or "look-up tables" data is stored a method to load

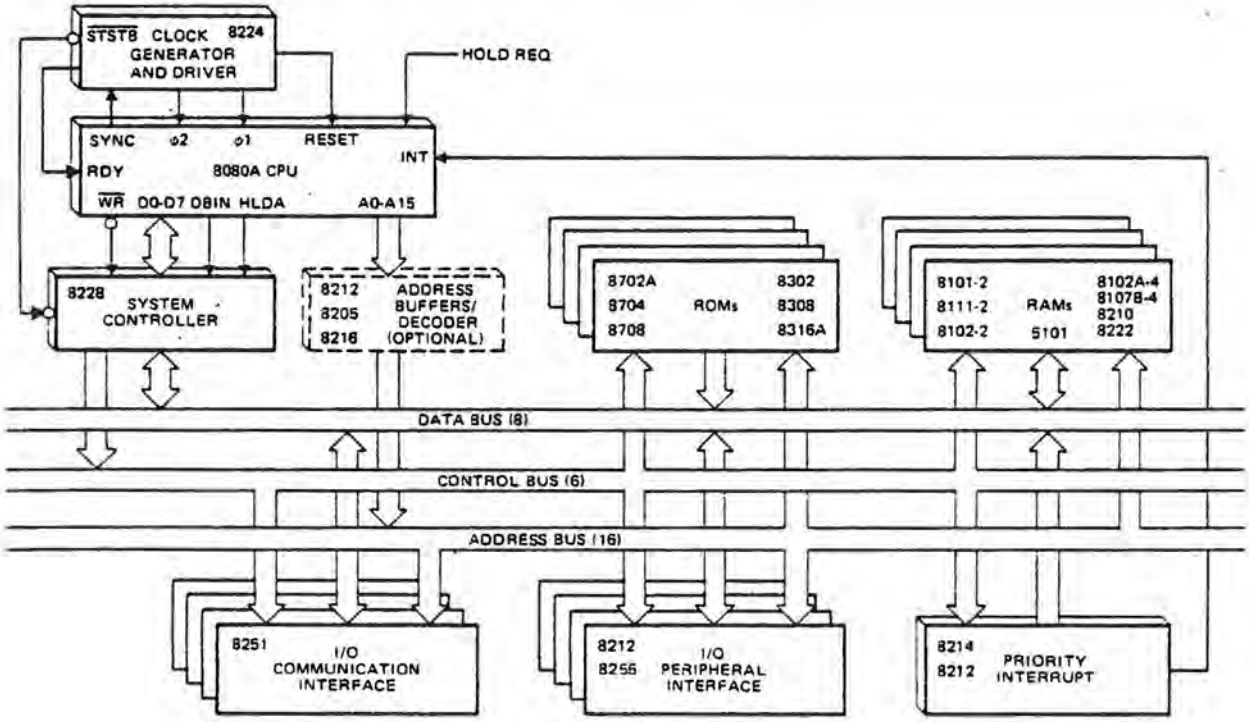


Figure 3-6. Microcomputer System

RAM memory must be provided, such as: Floppy Disk, Paper Tape, etc.

The CPU treats RAM in exactly the same manner as ROM for addressing data to be read. Writing data is very similar; the RAM is issued an address during the first portion of the Memory Write cycle (T1 & T2) in T3 when the data that is to be written is output by the CPU and is stable on the bus an $\overline{MEMW}$ command is generated. The $\overline{MEMW}$ signal is connected to the R/W input of the RAM and strobes the data into the addressed location.

In Figure 3-7 a typical Memory system is illustrated to show how standard semiconductor components interface to the 8080 bus. The memory array shown has 8K bytes (8 bits/byte) of ROM storage, using four Intel® 8216As and 512 bytes of RAM storage, using Intel 8111 static RAMs. The basic interface to the bus structure detailed here is common to almost any size memory. The only addition that might have to be made for larger systems is more buffers (8216/8212) and decoders (8205) for generating "chip selects."

The memories chosen for this example have an access time of 850 nS (max) to illustrate that slower, economical devices can be easily interfaced to the 8080 with little effect on performance. When the 8080 is operated from a clock generator with a tCY of 500 nS the required memory access time is Approx. 450-550 nS. See detailed timing specification Pg. 5-16. Using memory devices of this speed such as Intel® 8308, 8102A, 8107A, etc. the READY input to the 8080 CPU can remain "high" because no "wait" states are required. Note that the bus interface to memory shown in Figure 3-7 remains the same. However, if slower memories are to be used, such as the devices illustrated (8316A, 8111) that have access times slower than the minimum requirement a simple logic control of the READY input to the 8080 CPU will insert an extra "wait state" that is equal to one or more clock periods as an access time "adjustment" delay to compensate. The effect of the extra "wait" state is naturally a slower execution time for the instruction. A single "wait" changes the basic instruction cycle to 2.5 microSeconds.

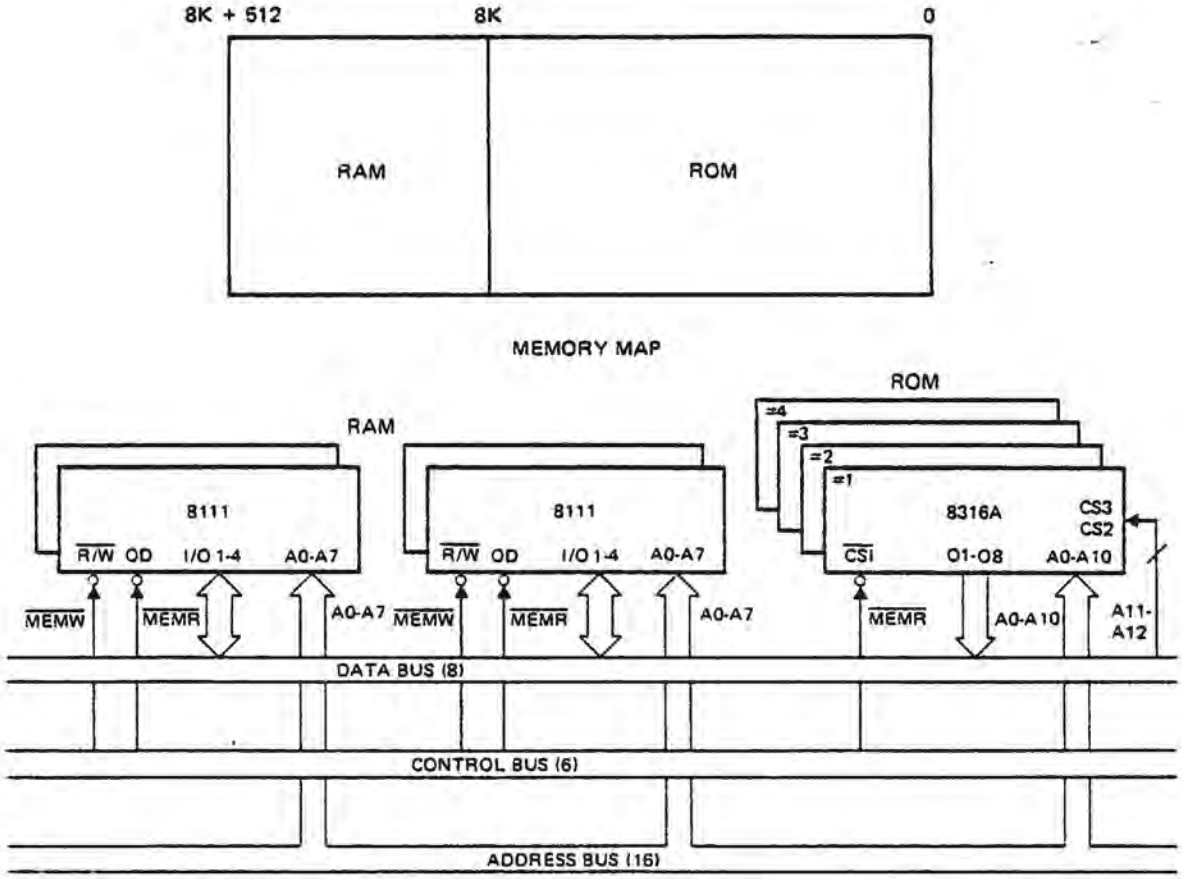


Figure 3-7. Typical Memory Interface

I/O INTERFACE

General Theory

As in any computer based system, the 8080 CPU must be able to communicate with devices or structures that exist outside its normal memory array. Devices like keyboards, paper tape, floppy disks, printers, displays and other control structures are used to input information into the 8080 CPU and display or store the results of the computational activity.

Probably the most important and strongest feature of the 8080 Microcomputer System is the flexibility and power of its I/O structure and the components that support it. There are many ways to structure the I/O array so that it will "fit" the total system environment to maximize efficiency and minimize component count.

The basic operation of the I/O structure can best be viewed as an array of single byte memory locations that can be Read from or Written into. The 8080 CPU has special instructions devoted to managing such transfers (IN, OUT). These instructions generally isolate memory and I/O arrays so that memory address space is not effected by the I/O structure and the general concept is that of a simple transfer to or from the Accumulator with an addressed "PORT". Another method of I/O architecture is to treat the I/O structure as part of the Memory array. This is generally referred to as "Memory Mapped I/O" and provides the designer with a powerful new "instruction set" devoted to I/O manipulation.

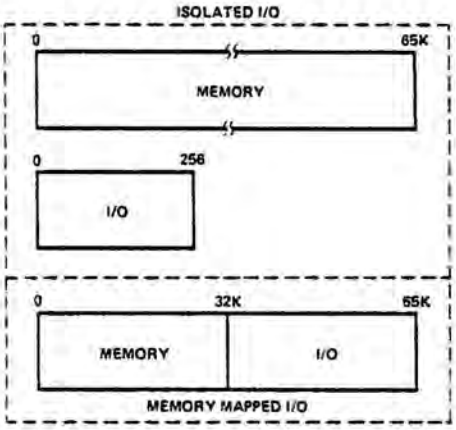


Figure 3-8. Memory/I/O Mapping.

Isolated I/O

In Figure 3-9 the system control signals, previously detailed in this chapter, are shown. This type of I/O architecture separates the memory address space from the I/O address space and uses a conceptually simple transfer to or from Accumulator technique. Such an architecture is easy to understand because I/O communicates only with the Accumulator using the IN or OUT instructions. Also because of the isolation of memory and I/O, the full address space (65K) is uneffected by I/O addressing.

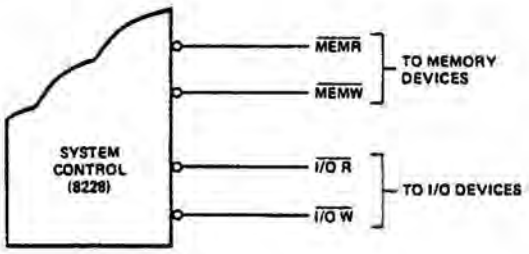


Figure 3-9. Isolated I/O.

Memory Mapped I/O

By assigning an area of memory address space as I/O a powerful architecture can be developed that can manipulate I/O using the same instructions that are used to manipulate memory locations. Thus, a "new" instruction set is created that is devoted to I/O handling.

As shown in Figure 3-10, new control signals are generated by gating the $\overline{MEMR}$ and $\overline{MEMW}$ signals with A_{15}, the most significant address bit. The new I/O control signals connect in exactly the same manner as Isolated I/O, thus the system bus characteristics are unchanged.

By assigning A_{15} as the I/O "flag", a simple method of I/O discipline is maintained:

If A_{15} is a "zero" then Memory is active.
If A_{15} is a "one" then I/O is active.

Other address bits can also be used for this function. A_{15} was chosen because it is the most significant address bit so it is easier to control with software and because it still allows memory addressing of 32K.

I/O devices are still considered addressed "ports" but instead of the Accumulator as the only transfer medium any of the internal registers can be used. All instructions that could be used to operate on memory locations can be used in I/O.

Examples:

MOVr, M	(Input Port to any Register)
MOV M, r	(Output any Register to Port)
MVI M	(Output immediate data to Port)
LDA	(Input to ACC)
STA	(Output from ACC to Port)
LHLD	(16 Bit Input)
SHLD	(16 Bit Output)
ADD M	(Add Port to ACC)
ANA M	("AND" Port with ACC)

It is easy to see that from the list of possible "new" instructions that this type of I/O architecture could have a drastic effect on increased system throughput. It is conceptually more difficult to understand than Isolated I/O and it does limit memory address space, but Memory Mapped I/O can mean a significant increase in overall speed and at the same time reducing required program memory area.

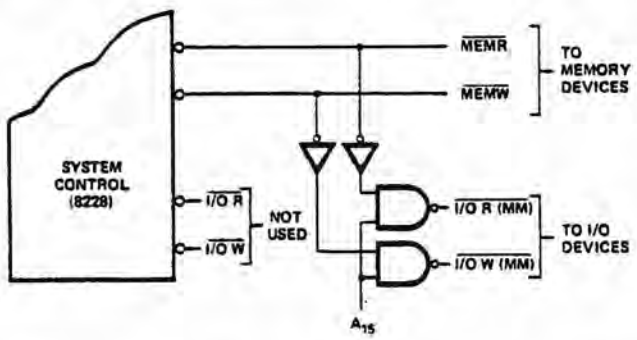


Figure 3-10. Memory Mapped I/O.

I/O Addressing

With both systems of I/O structure the addressing of each device can be configured to optimize efficiency and reduce component count. One method, the most common, is to decode the address bus into exclusive "chip selects" that enable the addressed I/O device, similar to generating chip-selects in memory arrays.

Another method is called "linear select". In this method, instead of decoding the Address Bus, a singular bit from the bus is assigned as the exclusive enable for a specific I/O device. This method, of course, limits the number of I/O devices that can be addressed but eliminates the need for extra decoders, an important consideration in small system design.

A simple example illustrates the power of such a flexible I/O structure. The first example illustrates the format of the second byte of the IN or OUT instruction using the Isolated I/O technique. The devices used are Intel®8255 Programmable Peripheral Interface units and are linear selected. Each device has three ports and from the format it can be seen that six devices can be addressed without additional decoders.

EXAMPLE #1

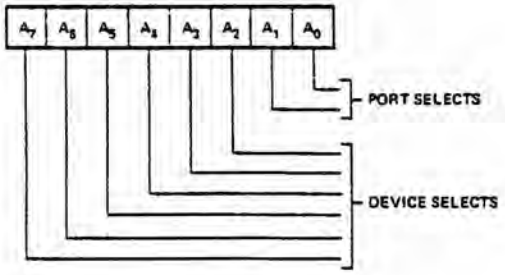


Figure 3-11. Isolated I/O – (Linear Select) (8255)

The second example uses Memory Mapped I/O and linear select to show how thirteen devices (8255) can be addressed without the use of extra decoders. The format shown could be the second and third bytes of the LDA or STA instructions or any other instructions used to manipulate I/O using the Memory Mapped technique.

It is easy to see that such a flexible I/O structure, that can be "tailored" to the overall system environment, provides the designer with a powerful tool to optimize efficiency and minimize component count.

EXAMPLE #2

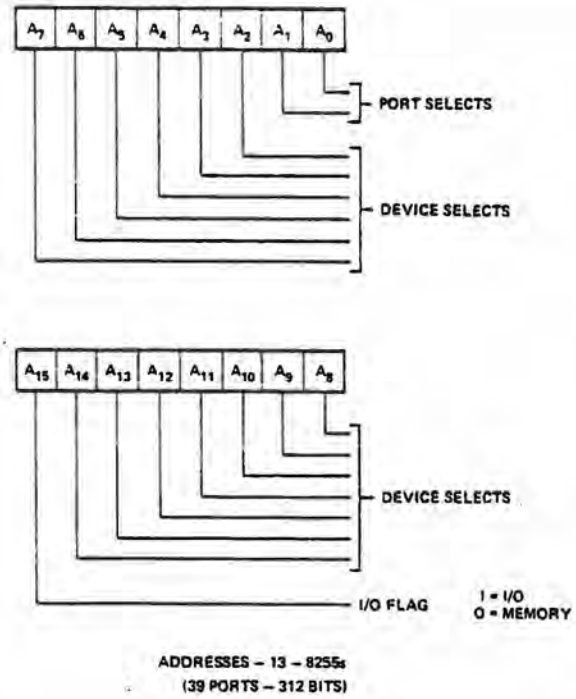


Figure 3-12. Memory Mapped I/O – (Linear Select (8255)

I/O Interface Example

In Figure 3-16 a typical I/O system is shown that uses a variety of devices (8212, 8251 and 8255). It could be used to interface the peripherals around an intelligent CRT terminals; keyboards, display, and communication interface. Another application could be in a process controller to interface sensors, relays, and motor controls. The limitation of the application area for such a circuit is solely that of the designers imagination.

The I/O structure shown interfaces to the 8080 CPU using the bus architecture developed previously in this chapter. Either Isolated or Memory Mapped techniques can be used, depending on the system I/O environment.

The 8251 provides a serial data communication interface so that the system can transmit and receive data over communication links such as telephone lines.

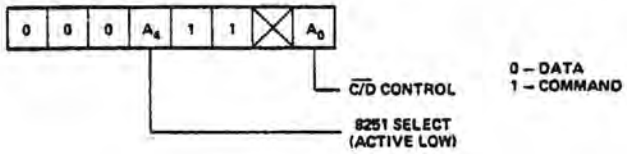


Figure 3-13. 8251 Format.

The two (2) 8255s provide twenty four bits each of programmable I/O data and control so that keyboards, sensors, paper tape, etc., can be interfaced to the system.

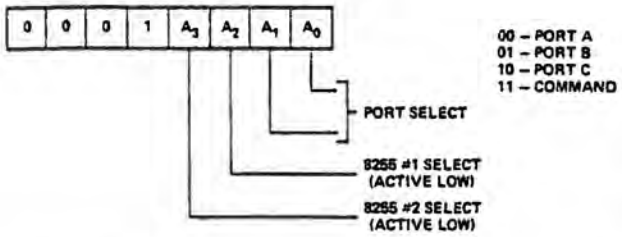


Figure 3-14. 8255 Format.

The three 8212s can be used to drive long lines or LED indicators due to their high drive capability. (15mA)

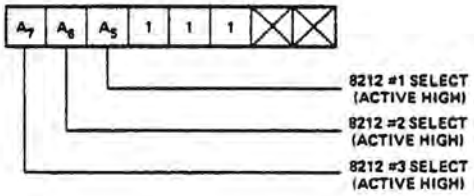


Figure 3-15. 8212 Format.

Addressing the structure is described in the formats illustrated in Figures 3-13, 3-14, 3-15. Linear Select is used so that no decoders are required thus, each device has an exclusive "enable bit".

The example shows how a powerful yet flexible I/O structure can be created using a minimum component count with devices that are all members of the 8080 Microcomputer System.

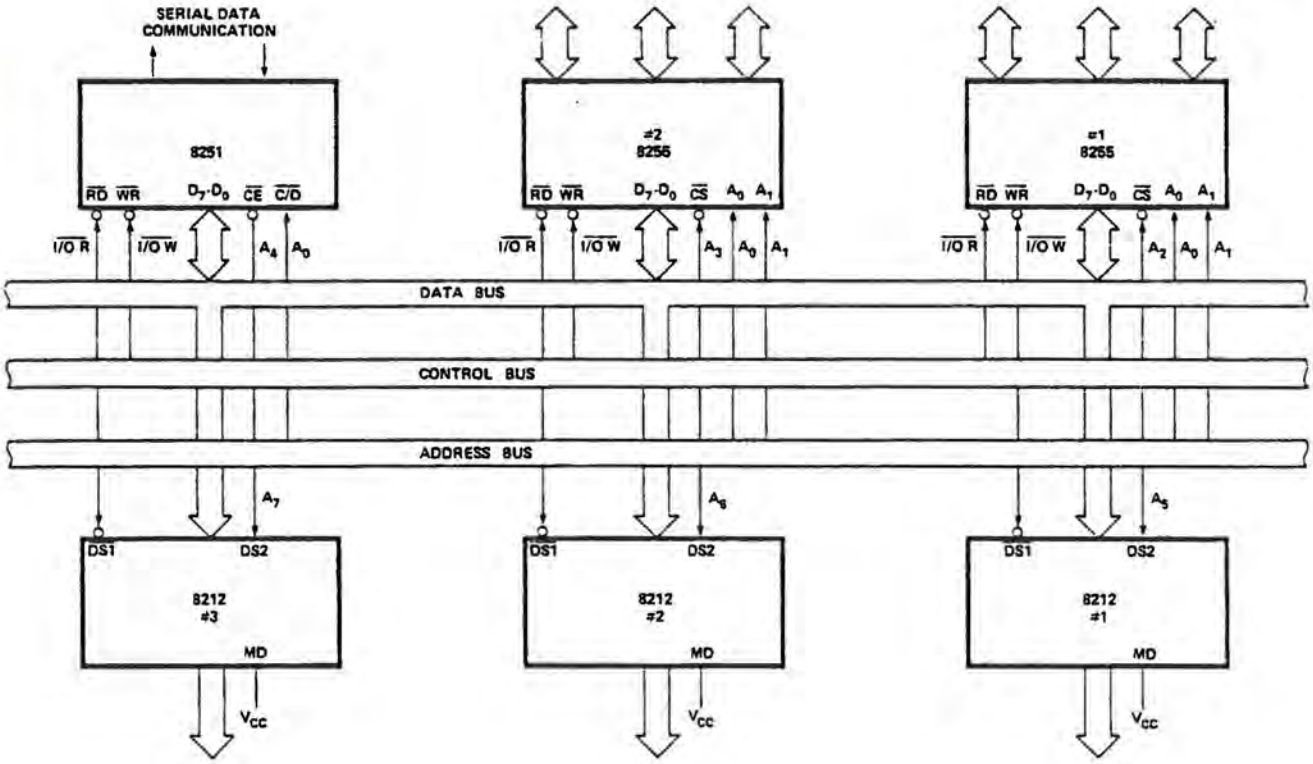


Figure 3-16. Typical I/O Interface.

CHAPTER 4
INSTRUCTION SET

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

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

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

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

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

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

THE 8080 INSTRUCTION SET

The 8080 instruction set includes five different types of instructions:

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

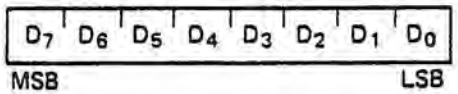
Instruction and Data Formats:

Memory for the 8080 is organized into 8-bit quantities, called Bytes. Each byte has a unique 16-bit binary address corresponding to its sequential position in memory.

The 8080 can directly address up to 65,536 bytes of memory, which may consist of both read-only memory (ROM) elements and random-access memory (RAM) elements (read/write memory).

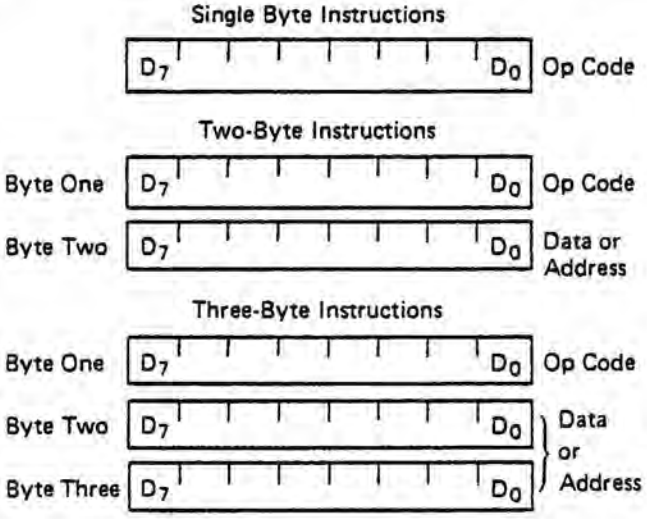
Data in the 8080 is stored in the form of 8-bit binary integers:

DATA WORD



When a register or data word contains a binary number, it is necessary to establish the order in which the bits of the number are written. In the Intel 8080, BIT 0 is referred to as the **Least Significant Bit (LSB)**, and BIT 7 (of an 8 bit number) is referred to as the **Most Significant Bit (MSB)**.

The 8080 program instructions may be one, two or three bytes in length. Multiple byte instructions must be stored in successive memory locations; the address of the first byte is always used as the address of the instructions. The exact instruction format will depend on the particular operation to be executed.



Addressing Modes:

Often the data that is to be operated on is stored in memory. When multi-byte numeric data is used, the data, like instructions, is stored in successive memory locations, with the least significant byte first, followed by increasingly significant bytes. The 8080 has four different modes for addressing data stored in memory or in registers:

- Direct – Bytes 2 and 3 of the instruction contain the exact memory address of the data item (the low-order bits of the address are in byte 2, the high-order bits in byte 3).
- Register – The instruction specifies the register or register-pair in which the data is located.
- Register Indirect – The instruction specifies a register-pair which contains the memory address where the data is located (the high-order bits of the address are in the first register of the pair, the low-order bits in the second).
- Immediate – The instruction contains the data itself. This is either an 8-bit quantity or a 16-bit quantity (least significant byte first, most significant byte second).

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

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

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

Condition Flags:

There are five condition flags associated with the execution of instructions on the 8080. They are Zero, Sign, Parity, Carry, and Auxiliary Carry, and are each represented by a 1-bit register in the CPU. A flag is "set" by forcing the bit to 1; "reset" by forcing the bit to 0.

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

Zero: If the result of an instruction has the value 0, this flag is set; otherwise it is reset.

Sign: If the most significant bit of the result of the operation has the value 1, this flag is set; otherwise it is reset.

Parity: If the modulo 2 sum of the bits of the result of the operation is 0, (i.e., if the result has even parity), this flag is set; otherwise it is reset (i.e., if the result has odd parity).

Carry: If the instruction resulted in a carry (from addition), or a borrow (from subtraction or a comparison) out of the high-order bit, this flag is set; otherwise it is reset.

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

Symbols and Abbreviations:

The following symbols and abbreviations are used in the subsequent description of the 8080 instructions:

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

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

rp — One of the register pairs:

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

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

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

SP represents the 16-bit stack pointer register.

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

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

rh — The first (high-order) register of a designated register pair.

rl — The second (low-order) register of a designated register pair.

PC — 16-bit program counter register (PCH and PCL are used to refer to the high-order and low-order 8 bits respectively).

SP — 16-bit stack pointer register (SPH and SPL are used to refer to the high-order and low-order 8 bits respectively).

r_m — Bit m of the register r (bits are number 7 through 0 from left to right).

Z,S,P,CY,AC — The condition flags:
Zero,
Sign,
Parity,
Carry,
and Auxiliary Carry, respectively.

( ) — The contents of the memory location or registers enclosed in the parentheses.

← — "Is transferred to"

∧ — Logical AND

∀ — Exclusive OR

∨ — Inclusive OR

+ — Addition

− — Two's complement subtraction

* — Multiplication

↔ — "Is exchanged with"

— — The one's complement (e.g., $(\overline{A})$)

n — The restart number 0 through 7

NNN — The binary representation 000 through 111 for restart number 0 through 7 respectively.

Description Format:

The following pages provide a detailed description of the instruction set of the 8080. Each instruction is described in the following manner:

1. The MAC 80 assembler format, consisting of the instruction mnemonic and operand fields, is printed in **BOLDFACE** on the left side of the first line.
2. The name of the instruction is enclosed in parenthesis on the right side of the first line.
3. The next line(s) contain a symbolic description of the operation of the instruction.
4. This is followed by a narative description of the operation of the instruction.
5. The following line(s) contain the binary fields and patterns that comprise the machine instruction.

6. The last four lines contain incidental information about the execution of the instruction. The number of machine cycles and states required to execute the instruction are listed first. If the instruction has two possible execution times, as in a Conditional Jump, both times will be listed, separated by a slash. Next, any significant data addressing modes (see Page 4-2) are listed. The last line lists any of the five Flags that are affected by the execution of the instruction.

Data Transfer Group:

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

MOV r1, r2 (Move Register)

(r1) ← (r2)

The content of register r2 is moved to register r1.

0	1	D	D	D	S	S	S

Cycles: 1
States: 5
Addressing: register
Flags: none

MOV r, M (Move from memory)

(r) ← ((H) (L))

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

0	1	D	D	D	1	1	0

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

MOV M, r (Move to memory)

((H) (L)) ← (r)

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

0	1	1	1	0	S	S	S

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

MVI r, data (Move Immediate)

(r) ← (byte 2)

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

0	0	D	D	D	1	1	0
data							

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

MVI M, data (Move to memory immediate)

((H) (L)) ← (byte 2)

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

0	0	1	1	0	1	1	0
data							

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

LXI rp, data 16 (Load register pair immediate)

(rh) ← (byte 3),

(rl) ← (byte 2)

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

0	0	R	P	0	0	0	1
low-order data							
high-order data							

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

LDA addr (Load Accumulator direct)

(A) ← ((byte 3)(byte 2))

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

0	0	1	1	1	0	1	0
low-order addr							
high-order addr							

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

STA addr (Store Accumulator direct)

((byte 3)(byte 2)) ← (A)

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

0	0	1	1	0	0	1	0
low-order addr							
high-order addr							

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

LHLD addr (Load H and L direct)

(L) ← ((byte 3)(byte 2))

(H) ← ((byte 3)(byte 2) + 1)

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

0	0	1	0	1	0	1	0
low-order addr							
high-order addr							

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

SHLD addr (Store H and L direct)

((byte 3)(byte 2)) ← (L)

((byte 3)(byte 2) + 1) ← (H)

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

0	0	1	0	0	0	1	0
low-order addr							
high-order addr							

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

LDAX rp (Load accumulator indirect)

(A) ← ((rp))

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

0	0	R	P	1	0	1	0

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

STAX rp (Store accumulator indirect)

((rp)) ← (A)

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

0	0	R	P	0	0	1	0

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

XCHG (Exchange H and L with D and E)

(H) ↔ (D)

(L) ↔ (E)

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

1	1	1	0	1	0	1	1

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

Arithmetic Group:

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

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

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

ADD r (Add Register)

(A) ← (A) + (r)

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

1	0	0	0	0	S	S	S

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

ADD M (Add memory)

(A) ← (A) + ((H) (L))

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

1	0	0	0	0	1	1	0

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

ADI data (Add immediate)

(A) ← (A) + (byte 2)

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

1	1	0	0	0	1	1	0
data							

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

ADC r (Add Register with carry)

(A) ← (A) + (r) + (CY)

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

1	0	0	0	1	S	S	S

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

ADC M (Add memory with carry)

(A) ← (A) + ((H) (L)) + (CY)

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

1	0	0	0	1	1	1	0

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

ACI data (Add immediate with carry)

(A) ← (A) + (byte 2) + (CY)

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

1	1	0	0	1	1	1	0
data							

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

SUB r (Subtract Register)

(A) ← (A) − (r)

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

1	0	0	1	0	S	S	S

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

SUB M (Subtract memory)

(A) ← (A) – ((H) (L))

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

1	0	0	1	0	1	1	0

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

SUI data (Subtract immediate)

(A) ← (A) – (byte 2)

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

1	1	0	1	0	1	1	0
data							

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

SBB r (Subtract Register with borrow)

(A) ← (A) – (r) – (CY)

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

1	0	0	1	1	S	S	S

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

SBB M (Subtract memory with borrow)

(A) ← (A) – ((H) (L)) – (CY)

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

1	0	0	1	1	1	1	0

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

SBI data (Subtract immediate with borrow)

(A) ← (A) – (byte 2) – (CY)

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

1	1	0	1	1	1	1	0
data							

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

INR r (Increment Register)

(r) ← (r) + 1

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

0	0	D	D	D	1	0	0

Cycles: 1
States: 5
Addressing: register
Flags: Z,S,P,AC

INR M (Increment memory)

((H) (L)) ← ((H) (L)) + 1

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

0	0	1	1	0	1	0	0

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

DCR r (Decrement Register)

(r) ← (r) – 1

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

0	0	D	D	D	1	0	1

Cycles: 1
States: 5
Addressing: register
Flags: Z,S,P,AC

DCR M (Decrement memory)

((H) (L)) ← ((H) (L)) − 1

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

0	0	1	1	0	1	0	1

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

INX rp (Increment register pair)

(rh) (rl) ← (rh) (rl) + 1

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

0	0	R	P	0	0	1	1

Cycles: 1
States: 5
Addressing: register
Flags: none

DCX rp (Decrement register pair)

(rh) (rl) ← (rh) (rl) − 1

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

0	0	R	P	1	0	1	1

Cycles: 1
States: 5
Addressing: register
Flags: none

DAD rp (Add register pair to H and L)

(H) (L) ← (H) (L) + (rh) (rl)

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

0	0	R	P	1	0	0	1

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

DAA (Decimal Adjust Accumulator)

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

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

NOTE: All flags are affected.

0	0	1	0	0	1	1	1

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

Logical Group:

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

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

ANA r (AND Register)

(A) ← (A) ∧ (r)

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

1	0	1	0	0	S	S	S

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

ANA M (AND memory)

(A) ← (A) ∧ ((H) (L))

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

1	0	1	0	0	1	1	0

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

ANI data (AND immediate)

(A) ← (A) ∧ (byte 2)

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

1	1	1	0	0	1	1	0
data							

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

XRA r (Exclusive OR Register)

(A) ← (A) ∀ (r)

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

1	0	1	0	1	S	S	S

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

XRA M (Exclusive OR Memory)

(A) ← (A) ∀ ((H) (L))

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

1	0	1	0	1	1	1	0

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

XRI data (Exclusive OR immediate)

(A) ← (A) ∀ (byte 2)

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

1	1	1	0	1	1	1	0
data							

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

ORA r (OR Register)

(A) ← (A) V (r)

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

1	0	1	1	0	S	S	S

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

ORA M (OR memory)

(A) ← (A) V ((H) (L))

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

1	0	1	1	0	1	1	0

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

ORI data (OR Immediate)

(A) ← (A) V (byte 2)

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

1	1	1	1	0	1	1	0
data							

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

CMP r (Compare Register)

(A) − (r)

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

1	0	1	1	1	S	S	S

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

CMP M (Compare memory)

(A) − ((H) (L))

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

1	0	1	1	1	1	1	0

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

CPI data (Compare immediate)

(A) − (byte 2)

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

1	1	1	1	1	1	1	0
data							

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

RLC (Rotate left)

$(A_{n+1}) \leftarrow (A_n)$; $(A_0) \leftarrow (A_7)$

$(CY) \leftarrow (A_7)$

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

0	0	0	0	0	1	1	1

Cycles: 1
States: 4
Flags: CY

RRC (Rotate right)

$(A_n) \leftarrow (A_{n-1})$; $(A_7) \leftarrow (A_0)$

$(CY) \leftarrow (A_0)$

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

0	0	0	0	1	1	1	1

Cycles: 1
States: 4
Flags: CY

RAL (Rotate left through carry)

$(A_{n+1}) \leftarrow (A_n)$; $(CY) \leftarrow (A_7)$

$(A_0) \leftarrow (CY)$

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

0	0	0	1	0	1	1	1

Cycles: 1
States: 4
Flags: CY

RAR (Rotate right through carry)

$(A_n) \leftarrow (A_{n+1})$; $(CY) \leftarrow (A_0)$

$(A_7) \leftarrow (CY)$

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

0	0	0	1	1	1	1	1

Cycles: 1
States: 4
Flags: CY

CMA (Complement accumulator)

$(A) \leftarrow (\overline{A})$

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

0	0	1	0	1	1	1	1

Cycles: 1
States: 4
Flags: none

CMC (Complement carry)

(CY) ← ($\overline{CY}$)

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

0	0	1	1	1	1	1	1

Cycles: 1
States: 4
Flags: CY

STC (Set carry)

(CY) ← 1

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

0	0	1	1	0	1	1	1

Cycles: 1
States: 4
Flags: CY

Branch Group:

This group of instructions alter normal sequential program flow.

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

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

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

JMP addr (Jump)

(PC) ← (byte 3) (byte 2)

Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.

1	1	0	0	0	0	1	1
low-order addr							
high-order addr							

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

Jcondition addr (Conditional jump)

If (CCC),

(PC) ← (byte 3) (byte 2)

If the specified condition is true, control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction; otherwise, control continues sequentially.

1	1	C	C	C	0	1	0
low-order addr							
high-order addr							

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

CALL addr (Call)

((SP) – 1) ← (PCH)
((SP) – 2) ← (PCL)
(SP) ← (SP) – 2
(PC) ← (byte 3) (byte 2)

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.

1	1	0	0	1	1	0	1
low-order addr							
high-order addr							

Cycles: 5
States: 17
Addressing: immediate/reg. indirect
Flags: none

Ccondition addr (Condition call)

If (CCC),

((SP) − 1) ← (PCH)

((SP) − 2) ← (PCL)

(SP) ← (SP) − 2

(PC) ← (byte 3) (byte 2)

If the specified condition is true, the actions specified in the CALL instruction (see above) are performed; otherwise, control continues sequentially.

1	1	C	C	C	1	0	0
low-order addr							
high-order addr							

Cycles: 3/5
States: 11/17
Addressing: immediate/reg. indirect
Flags: none

RET (Return)

(PCL) ← ((SP));

(PCH) ← ((SP) + 1);

(SP) ← (SP) + 2;

The content of the memory location whose address is specified in register SP is moved to the low-order eight bits of register PC. The content of the memory location whose address is one more than the content of register SP is moved to the high-order eight bits of register PC. The content of register SP is incremented by 2.

1	1	0	0	1	0	0	1

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

Rcondition (Conditional return)

If (CCC),

(PCL) ← ((SP))

(PCH) ← ((SP) + 1)

(SP) ← (SP) + 2

If the specified condition is true, the actions specified in the RET instruction (see above) are performed; otherwise, control continues sequentially.

1	1	C	C	C	0	0	0

Cycles: 1/3
States: 5/11
Addressing: reg. indirect
Flags: none

RST n (Restart)

((SP) − 1) ← (PCH)

((SP) − 2) ← (PCL)

(SP) ← (SP) − 2

(PC) ← 8 • (NNN)

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two. Control is transferred to the instruction whose address is eight times the content of NNN.

1	1	N	N	N	1	1	1

Cycles: 3
States: 11
Addressing: reg. indirect
Flags: none

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	N	N	N	0	0	0

Program Counter After Restart

PCHL (Jump H and L indirect – move H and L to PC)

(PCH) ← (H)

(PCL) ← (L)

The content of register H is moved to the high-order eight bits of register PC. The content of register L is moved to the low-order eight bits of register PC.

1	1	1	0	1	0	0	1

Cycles: 1
States: 5
Addressing: register
Flags: none

Stack, I/O, and Machine Control Group:

This group of instructions performs I/O, manipulates the Stack, and alters internal control flags.

Unless otherwise specified, **condition flags are not affected by any instructions in this group.**

PUSH rp (Push)

((SP) − 1) ← (rh)
((SP) − 2) ← (rl)
(SP) ← (SP) − 2

The content of the high-order register of register pair rp is moved to the memory location whose address is one less than the content of register SP. The content of the low-order register of register pair rp is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. **Note: Register pair rp = SP may not be specified.**

1	1	R	P	0	1	0	1

Cycles: 3
States: 11
Addressing: reg. indirect
Flags: none

PUSH PSW (Push processor status word)

((SP) − 1) ← (A)
$((SP) - 2)_0$ ← (CY), $((SP) - 2)_1$ ← 1
$((SP) - 2)_2$ ← (P), $((SP) - 2)_3$ ← 0
$((SP) - 2)_4$ ← (AC), $((SP) - 2)_5$ ← 0
$((SP) - 2)_6$ ← (Z), $((SP) - 2)_7$ ← (S)
(SP) ← (SP) − 2

The content of register A is moved to the memory location whose address is one less than register SP. The contents of the condition flags are assembled into a processor status word and the word is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two.

1	1	1	1	0	1	0	1

Cycles: 3
States: 11
Addressing: reg. indirect
Flags: none

FLAG WORD

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	0	AC	0	P	1	CY

POP rp (Pop)

(rl) ← ((SP))
(rh) ← ((SP) + 1)
(SP) ← (SP) + 2

The content of the memory location, whose address is specified by the content of register SP, is moved to the low-order register of register pair rp. The content of the memory location, whose address is one more than the content of register SP, is moved to the high-order register of register pair rp. The content of register SP is incremented by 2. **Note: Register pair rp = SP may not be specified.**

1	1	R	P	0	0	0	1

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

POP PSW (Pop processor status word)

(CY) ← $((SP))_0$
(P) ← $((SP))_2$
(AC) ← $((SP))_4$
(Z) ← $((SP))_6$
(S) ← $((SP))_7$
(A) ← ((SP) + 1)
(SP) ← (SP) + 2

The content of the memory location whose address is specified by the content of register SP is used to restore the condition flags. The content of the memory location whose address is one more than the content of register SP is moved to register A. The content of register SP is incremented by 2.

1	1	1	1	0	0	0	1

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

XTHL (Exchange stack top with H and L)

(L) ⟷ ((SP))
(H) ⟷ ((SP) + 1)

The content of the L register is exchanged with the content of the memory location whose address is specified by the content of register SP. The content of the H register is exchanged with the content of the memory location whose address is one more than the content of register SP.

1	1	1	0	0	0	1	1

Cycles: 5
States: 18
Addressing: reg. indirect
Flags: none

SPHL (Move HL to SP)

(SP) ← (H) (L)

The contents of registers H and L (16 bits) are moved to register SP.

1	1	1	1	1	0	0	1

Cycles: 1
States: 5
Addressing: register
Flags: none

IN port (Input)

(A) ← (data)

The data placed on the eight bit bi-directional data bus by the specified port is moved to register A.

1	1	0	1	1	0	1	1
port							

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

OUT port (Output)

(data) ← (A)

The content of register A is placed on the eight bit bi-directional data bus for transmission to the specified port.

1	1	0	1	0	0	1	1
port							

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

EI (Enable interrupts)

The interrupt system is enabled **following the execution of the next instruction.**

1	1	1	1	1	0	1	1

Cycles: 1
States: 4
Flags: none

DI (Disable interrupts)

The interrupt system is disabled **immediately following the execution of the DI instruction.**

1	1	1	1	0	0	1	1

Cycles: 1
States: 4
Flags: none

HLT (Halt)

The processor is stopped. The registers and flags are unaffected.

0	1	1	1	0	1	1	0

Cycles: 1
States: 7
Flags: none

NOP (No op)

No operation is performed. The registers and flags are unaffected.

0	0	0	0	0	0	0	0

Cycles: 1
States: 4
Flags: none

INSTRUCTION SET

Summary of Processor Instructions

Mnemonic	Description	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
		Instruction Code[1]								
$MOV_{r1,r2}$	Move register to register	0	1	D	D	D	S	S	S	5
MOV M, r	Move register to memory	0	1	1	1	0	S	S	S	7
MOV r, M	Move memory to register	0	1	D	D	D	1	1	0	7
HLT	Halt	0	1	1	1	0	1	1	0	7
MVI r	Move immediate register	0	0	D	D	D	1	1	0	7
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	10
INR r	Increment register	0	0	D	D	D	1	0	0	5
DCR r	Decrement register	0	0	D	D	D	1	0	1	5
INR M	Increment memory	0	0	1	1	0	1	0	0	10
DCR M	Decrement memory	0	0	1	1	0	1	0	1	10
ADD r	Add register to A	1	0	0	0	0	S	S	S	4
ADC r	Add register to A with carry	1	0	0	0	1	S	S	S	4
SUB r	Subtract register from A	1	0	0	1	0	S	S	S	4
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4
ANA r	And register with A	1	0	1	0	0	S	S	S	4
XRA r	Exclusive Or register with A	1	0	1	0	1	S	S	S	4
ORA r	Or register with A	1	0	1	1	0	S	S	S	4
CMP r	Compare register with A	1	0	1	1	1	S	S	S	4
ADD M	Add memory to A	1	0	0	0	0	1	1	0	7
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	7
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	7
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
ANA M	And memory with A	1	0	1	0	0	1	1	0	7
XRA M	Exclusive Or memory with A	1	0	1	0	1	1	1	0	7
ORA M	Or memory with A	1	0	1	1	0	1	1	0	7
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	7
ADI	Add immediate to A	1	1	0	0	0	1	1	0	7
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
XRI	Exclusive Or immediate with A	1	1	1	0	1	1	1	0	7
ORI	Or immediate with A	1	1	1	1	0	1	1	0	7
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7
RLC	Rotate A left	0	0	0	0	0	1	1	1	4
RRC	Rotate A right	0	0	0	0	1	1	1	1	4
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JC	Jump on carry	1	1	0	1	1	0	1	0	10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	10
JZ	Jump on zero	1	1	0	0	1	0	1	0	10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	10
JP	Jump on positive	1	1	1	1	0	0	1	0	10
JM	Jump on minus	1	1	1	1	1	0	1	0	10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	10
CALL	Call unconditional	1	1	0	0	1	1	0	1	17
CC	Call on carry	1	1	0	1	1	1	0	0	11/17
CNC	Call on no carry	1	1	0	1	0	1	0	0	11/17
CZ	Call on zero	1	1	0	0	1	1	0	0	11/17
CNZ	Call on no zero	1	1	0	0	0	1	0	0	11/17
CP	Call on positive	1	1	1	1	0	1	0	0	11/17
CM	Call on minus	1	1	1	1	1	1	0	0	11/17
CPE	Call on parity even	1	1	1	0	1	1	0	0	11/17
CPO	Call on parity odd	1	1	1	0	0	1	0	0	11/17
RET	Return	1	1	0	0	1	0	0	1	10
RC	Return on carry	1	1	0	1	1	0	0	0	5/11
RNC	Return on no carry	1	1	0	1	0	0	0	0	5/11
RZ	Return on zero	1	1	0	0	1	0	0	0	5/11
RNZ	Return on no zero	1	1	0	0	0	0	0	0	5/11
RP	Return on positive	1	1	1	1	0	0	0	0	5/11
RM	Return on minus	1	1	1	1	1	0	0	0	5/11
RPE	Return on parity even	1	1	1	0	1	0	0	0	5/11
RPO	Return on parity odd	1	1	1	0	0	0	0	0	5/11
RST	Restart	1	1	A	A	A	1	1	1	11
IN	Input	1	1	0	1	1	0	1	1	10
OUT	Output	1	1	0	1	0	0	1	1	10
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	10
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	10
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	10
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	11
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	11
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	11
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	11
POP B	Pop register pair B & C off stack	1	1	0	0	0	0	0	1	10
POP D	Pop register pair D & E off stack	1	1	0	1	0	0	0	1	10
POP H	Pop register pair H & L off stack	1	1	1	0	0	0	0	1	10
POP PSW	Pop A and Flags off stack	1	1	1	1	0	0	0	1	10
STA	Store A direct	0	0	1	1	0	0	1	0	13
LDA	Load A direct	0	0	1	1	1	0	1	0	13
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	4
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	18
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	5
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	5
DAD B	Add B & C to H & L	0	0	0	0	1	0	0	1	10
DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1	10
DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1	10
DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	10
STAX B	Store A indirect	0	0	0	0	0	0	1	0	7
STAX D	Store A indirect	0	0	0	1	0	0	1	0	7
LDAX B	Load A indirect	0	0	0	0	1	0	1	0	7
LDAX D	Load A indirect	0	0	0	1	1	0	1	0	7
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	5
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	5
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	5
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	5
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	5
DCX D	Decrement D & E	0	0	0	1	1	0	1	1	5
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	5
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	5
CMA	Complement A	0	0	1	0	1	1	1	1	4
STC	Set carry	0	0	1	1	0	1	1	1	4
CMC	Complement carry	0	0	1	1	1	1	1	1	4
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	16
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	16
EI	Enable Interrupts	1	1	1	1	1	0	1	1	4
DI	Disable Interrupt	1	1	1	1	0	0	1	1	4
NOP	No-operation	0	0	0	0	0	0	0	0	4

NOTES: 1. DDD or SSS – 000 B – 001 C – 010 D – 011 E – 100 H – 101 L – 110 Memory – 111 A.
2. Two possible cycle times, (5/11) indicate instruction cycles dependent on condition flags.

Schottky Bipolar 8224

CLOCK GENERATOR AND DRIVER FOR 8080A CPU

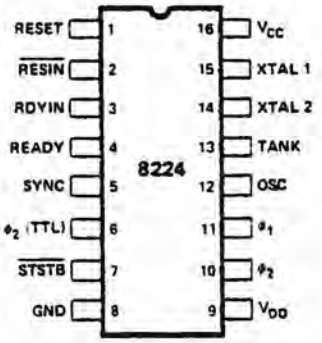
- **Single Chip Clock Generator/Driver for 8080A CPU**
- **Power-Up Reset for CPU**
- **Ready Synchronizing Flip-Flop**
- **Advanced Status Strobe**
- **Oscillator Output for External System Timing**
- **Crystal Controlled for Stable System Operation**
- **Reduces System Package Count**

The 8224 is a single chip clock generator/driver for the 8080A CPU. It is controlled by a crystal, selected by the designer, to meet a variety of system speed requirements.

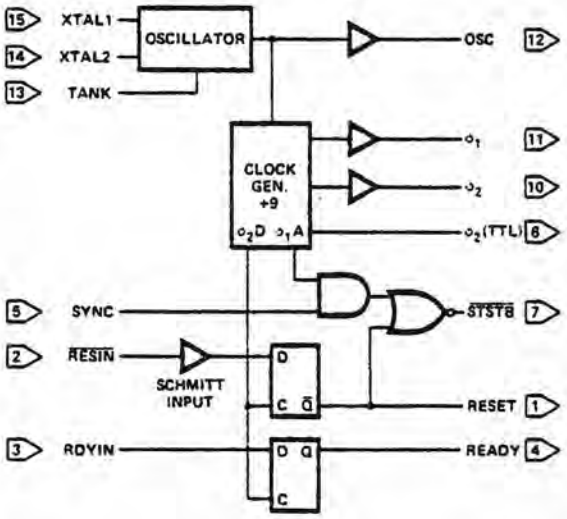
Also included are circuits to provide power-up reset, advance status strobe and synchronization of ready.

The 8224 provides the designer with a significant reduction of packages used to generate clocks and timing for 8080A.

PIN CONFIGURATION



BLOCK DIAGRAM



PIN NAMES

RESIN	RESET INPUT
RESET	RESET OUTPUT
RDYIN	READY INPUT
READY	READY OUTPUT
SYNC	SYNC INPUT
STSTB	STATUS STB (ACTIVE LOW)
φ1	8080
φ2	CLOCKS

XTAL 1	CONNECTIONS
XTAL 2	FOR CRYSTAL
TANK	USED WITH OVERTONE XTAL
OSC	OSCILLATOR OUTPUT
φ2 (TTL)	φ2 CLK (TTL LEVEL)
VCC	+5V
VDD	+12V
GND	0V

FUNCTIONAL DESCRIPTION

General

The 8224 is a single chip Clock Generator/Driver for the 8080A CPU. It contains a crystal-controlled oscillator, a "divide by nine" counter, two high-level drivers and several auxiliary logic functions.

Oscillator

The oscillator circuit derives its basic operating frequency from an external, series resonant, fundamental mode crystal. Two inputs are provided for the crystal connections (XTAL1, XTAL2).

The selection of the external crystal frequency depends mainly on the speed at which the 8080A is to be run at. Basically, the oscillator operates at 9 times the desired processor speed.

A simple formula to guide the crystal selection is:

$$\text{Crystal Frequency} = \frac{1}{t_{CY}} \text{ times } 9$$

Example 1: (500ns t_{CY})
2mHz times 9 = 18mHz*

Example 2: (800ns t_{CY})
1.25mHz times 9 = 11.25mHz

Another input to the oscillator is TANK. This input allows the use overtone mode crystals. This type of crystal generally has much lower "gain" than the fundamental type so an external LC network is necessary to provide the additional "gain" for proper oscillator operation. The external LC network is connected to the TANK input and is AC coupled to ground. See Figure 4.

The formula for the LC network is:

$$F = \frac{1}{2\pi\sqrt{LC}}$$

The output of the oscillator is buffered and brought out on OSC (pin 12) so that other system timing signals can be derived from this stable, crystal-controlled source.

*When using crystals above 10mHz a small amount of frequency "trimming" may be necessary to produce the exact desired frequency. The addition of a small selected capacitance (3pF - 10pF) in series with the crystal will accomplish this function.

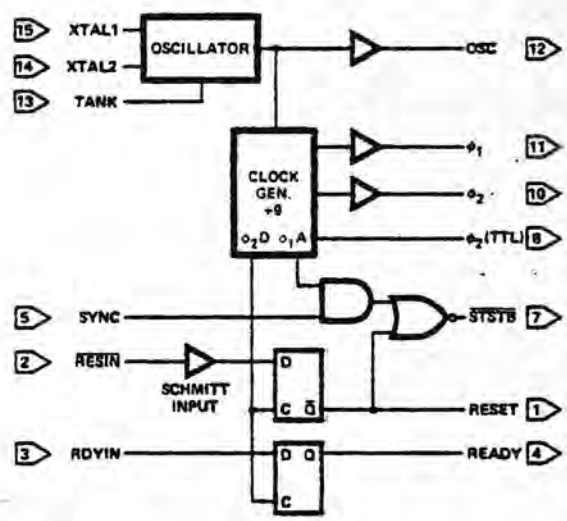
Clock Generator

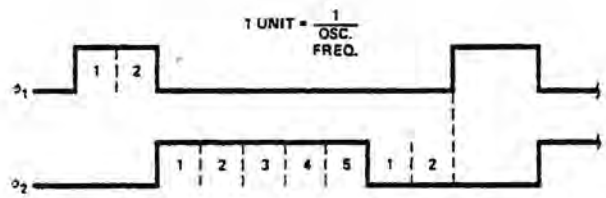
The Clock Generator consists of a synchronous "divide by nine" counter and the associated decode gating to create the waveforms of the two 8080A clocks and auxiliary timing signals.

The waveforms generated by the decode gating follow a simple 2-5-2 digital pattern. See Figure 2. The clocks generated; phase 1 and phase 2, can best be thought of as consisting of "units" based on the oscillator frequency. Assume that one "unit" equals the period of the oscillator frequency. By multiplying the number of "units" that are contained in a pulse width or delay, times the period of the oscillator frequency, the approximate time in nanoseconds can be derived.

The outputs of the clock generator are connected to two high level drivers for direct interface to the 8080A CPU. A TTL level phase 2 is also brought out ϕ_2 (TTL) for external timing purposes. It is especially useful in DMA dependant activities. This signal is used to gate the requesting device onto the bus once the 8080A CPU issues the Hold Acknowledgement (HLDA).

Several other signals are also generated internally so that optimum timing of the auxiliary flip-flops and status strobe ($\overline{STSTB}$) is achieved.





EXAMPLE: (8080 t_{CY} = 500ns)
OSC = 18mHz/55ns
ϕ_1 = 110ns (2 x 55ns)
ϕ_2 = 275ns (5 x 55ns)
ϕ_2-ϕ_1 = 110ns (2 x 55ns)

$\overline{\text{STSTB}}$ (Status Strobe)

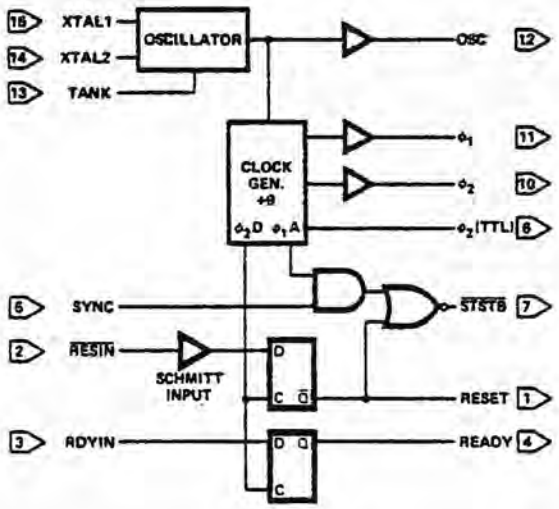
At the beginning of each machine cycle the 8080A CPU issues status information on its data bus. This information tells what type of action will take place during that machine cycle. By bringing in the SYNC signal from the CPU, and gating it with an internal timing signal (ϕ1A), an active low strobe can be derived that occurs at the start of each machine cycle at the earliest possible moment that status data is stable on the bus. The $\overline{\text{STSTB}}$ signal connects directly to the 8228 System Controller.

The power-on Reset also generates $\overline{\text{STSTB}}$, but of course, for a longer period of time. This feature allows the 8228 to be automatically reset without additional pins devoted for this function.

Power-On Reset and Ready Flip-Flops

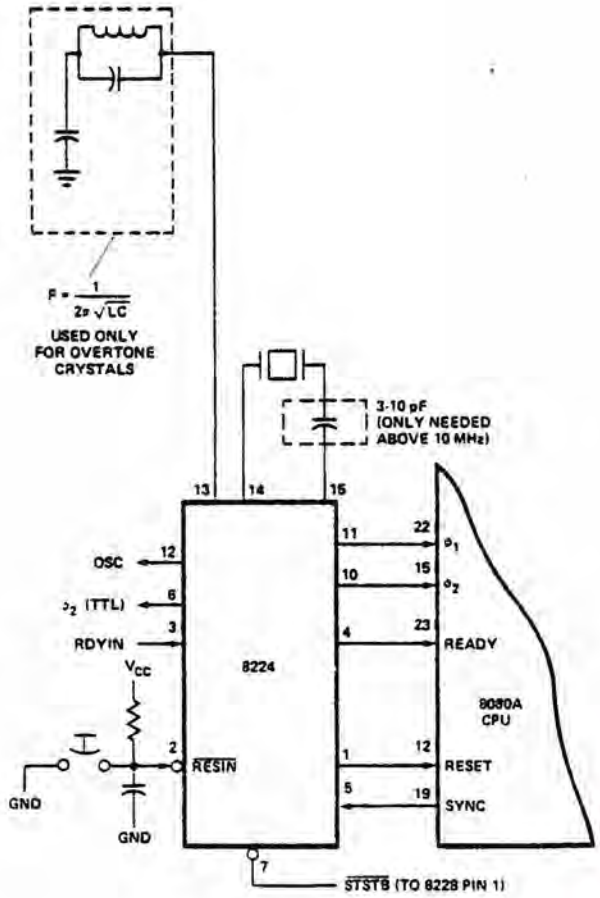
A common function in 8080A Microcomputer systems is the generation of an automatic system reset and start-up upon initial power-on. The 8224 has a built in feature to accomplish this feature.

An external RC network is connected to the $\overline{\text{RESIN}}$ input. The slow transition of the power supply rise is sensed by an internal Schmitt Trigger. This circuit converts the slow transition into a clean, fast edge when its input level reaches a predetermined value. The output of the Schmitt Trigger is connected to a "D" type flip-flop that is clocked with ϕ2D (an internal timing signal). The flip-flop is synchronously reset and an active high level that complies with the 8080A input spec is generated. For manual switch type system Reset circuits, an active low switch closing can be connected to the $\overline{\text{RESIN}}$ input in addition to the power-on RC net-network.



The READY input to the 8080A CPU has certain timing specifications such as "set-up and hold" thus, an external synchronizing flip-flop is required. The 8224 has this feature built-in. The RDYIN input presents the asynchronous "wait request" to the "D" type flip-flop. By clocking the flip-flop with ϕ2D, a synchronized READY signal at the correct input level, can be connected directly to the 8080A.

The reason for requiring an external flip-flop to synchronize the "wait request" rather than internally in the 8080 CPU is that due to the relatively long delays of MOS logic such an implementation would "rob" the designer of about 200ns during the time his logic is determining if a "wait" is necessary. An external bipolar circuit built into the clock generator eliminates most of this delay and has no effect on component count.



D.C. Characteristics

T_A = 0°C to 70°C; V_{CC} = +5.0V ±5%; V_{DD} = +12V ±5%.

Symbol	Parameter	Limits Min.	Limits Typ.	Limits Max.	Units	Test Conditions
I_F	Input Current Loading			-.25	mA	V_F = .45V
I_R	Input Leakage Current			10	μA	V_R = 5.25V
V_C	Input Forward Clamp Voltage			1.0	V	I_C = -5mA
V_{IL}	Input "Low" Voltage			.8	V	V_{CC} = 5.0V
V_{IH}	Input "High" Voltage	2.6 2.0			V	Reset Input All Other Inputs
V_{IH}-V_{IL}	REDIN Input Hysteresis	.25			mV	V_{CC} = 5.0V
V_{OL}	Output "Low" Voltage			.45 .45	V V	(ϕ_1,ϕ_2), Ready, Reset, $\overline{STSTB}$ I_{OL} =2.5mA All Other Outputs I_{OL} = 15mA
V_{OH}	Output "High" Voltage ϕ_1 , ϕ_2 READY, RESET All Other Outputs	 9.4 3.6 2.4			 V V V	 I_{OH} = -100μA I_{OH} = -100μA I_{OH} = -1mA
I_{SC}[1]	Output Short Circuit Current (All Low Voltage Outputs Only)	-10		-60	mA	V_O = 0V V_{CC} = 5.0V
I_{CC}	Power Supply Current			115	mA	
I_{DD}	Power Supply Current			12	mA	

Note: 1. Caution, ϕ_1 and ϕ_2 output drivers do not have short circuit protection

CRYSTAL REQUIREMENTS

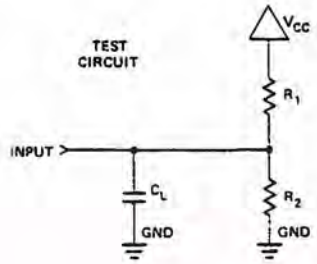
Tolerance: .005% at 0°C -70°C
Resonance: Series (Fundamental)*
Load Capacitance: 20-35pF
Equivalent Resistance: 75-20 ohms
Power Dissipation (Min): 4mW

*With tank circuit use 3rd overtone mode.

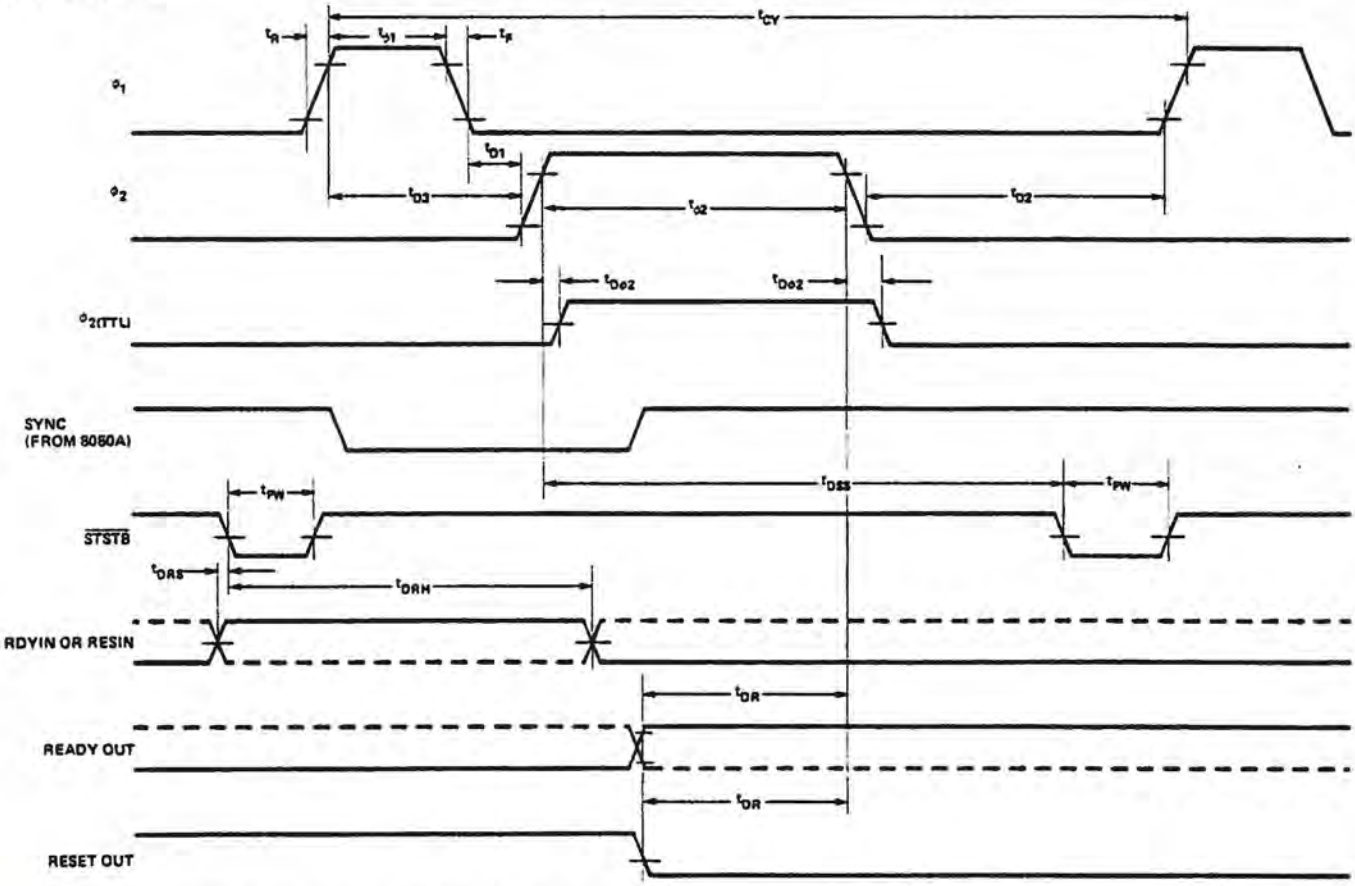
A.C. Characteristics

V_{CC} = +5.0V ± 5%; V_{DD} = +12.0V ± 5%; T_A = 0°C to 70°C

Symbol	Parameter	Limits Min.	Limits Typ.	Limits Max.	Units	Test Conditions
$t_{\phi 1}$	ϕ_1 Pulse Width	$\frac{2tcy}{9}$ - 20ns			ns	C_L = 20pF to 50pF
$t_{\phi 2}$	ϕ_2 Pulse Width	$\frac{5tcy}{9}$ - 35ns				
t_{D1}	ϕ_1 to ϕ_2 Delay	0				
t_{D2}	ϕ_2 to ϕ_1 Delay	$\frac{2tcy}{9}$ - 14ns				
t_{D3}	ϕ_1 to ϕ_2 Delay	$\frac{2tcy}{9}$		$\frac{2tcy}{9}$ + 20ns		
t_R	ϕ_1 and ϕ_2 Rise Time			20		
t_F	ϕ_1 and ϕ_2 Fall Time			20		
$t_{D\phi 2}$	ϕ_2 to ϕ_2 (TTL) Delay	-5		+15	ns	ϕ_2TTL, CL=30 R_1=300Ω R_2=600Ω
t_{DSS}	ϕ_2 to $\overline{STSTB}$ Delay	$\frac{6tcy}{9}$ - 30ns		$\frac{6tcy}{9}$		$\overline{STSTB}$, CL=15pF R_1 = 2K R_2 = 4K
t_{PW}	$\overline{STSTB}$ Pulse Width	$\frac{tcy}{9}$ - 15ns				
t_{DRS}	RDYIN Setup Time to Status Strobe	50ns - $\frac{4tcy}{9}$				
t_{DRH}	RDYIN Hold Time After $\overline{STSTB}$	$\frac{4tcy}{9}$				
t_{DR}	RDYIN or RESIN to ϕ_2 Delay	$\frac{4tcy}{9}$ - 25ns				Ready & Reset CL=10pF R_1=2K R_2=4K
t_{CLK}	CLK Period		$\frac{tcy}{9}$			
f_{max}	Maximum Oscillating Frequency	27			MHz	
C_{in}	Input Capacitance			8	pF	V_{CC}=+5.0V V_{DD}=+12V V_{BIAS}=2.5V f=1MHz



WAVEFORMS



VOLTAGE MEASUREMENT POINTS: ϕ_1, ϕ_2 Logic "0" = 1.0V, Logic "1" = 8.0V. All other signals measured at 1.5V.

EXAMPLE:

A.C. Characteristics (For t_{CY} = 488.28 ns)

T_A = 0°C to 70°C; V_{DD} = +5V ±5%; V_{DD} = +12V ±5%.

Symbol	Parameter	Limits Min.	Limits Typ.	Limits Max.	Units	Test Conditions
$t_{\phi1}$	ϕ_1 Pulse Width	89			ns	t_{CY}=488.28ns
$t_{\phi2}$	ϕ_2 Pulse Width	236			ns	
t_{D1}	Delay ϕ_1 to ϕ_2	0			ns	
t_{D2}	Delay ϕ_2 to ϕ_1	95			ns	ϕ_1 & ϕ_2 Loaded to C_L = 20 to 50pF
t_{D3}	Delay ϕ_1 to ϕ_2 Leading Edges	109		129	ns	
t_r	Output Rise Time			20	ns	
t_f	Output Fall Time			20	ns	
t_{DSS}	ϕ_2 to $\overline{STSTB}$ Delay	296		326	ns	
$t_{D\phi2}$	ϕ_2 to ϕ_2 (TTL) Delay	-5		+15	ns	
t_{PW}	Status Strobe Pulse Width	40			ns	Ready & Reset Loaded to 2mA/10pF
t_{DRS}	RDYIN Setup Time to $\overline{STSTB}$	-167			ns	
t_{DRH}	RDYIN Hold Time after $\overline{STSTB}$	217			ns	All measurements referenced to 1.5V unless specified otherwise.
t_{DR}	READY or RESET to ϕ_2 Delay	192			ns	
f_{MAX}	Oscillator Frequency			18.432	MHz	

Silicon Gate MOS **8080A**

SINGLE CHIP 8-BIT N-CHANNEL MICROPROCESSOR

The 8080A is functionally and electrically compatible with the Intel® 8080.

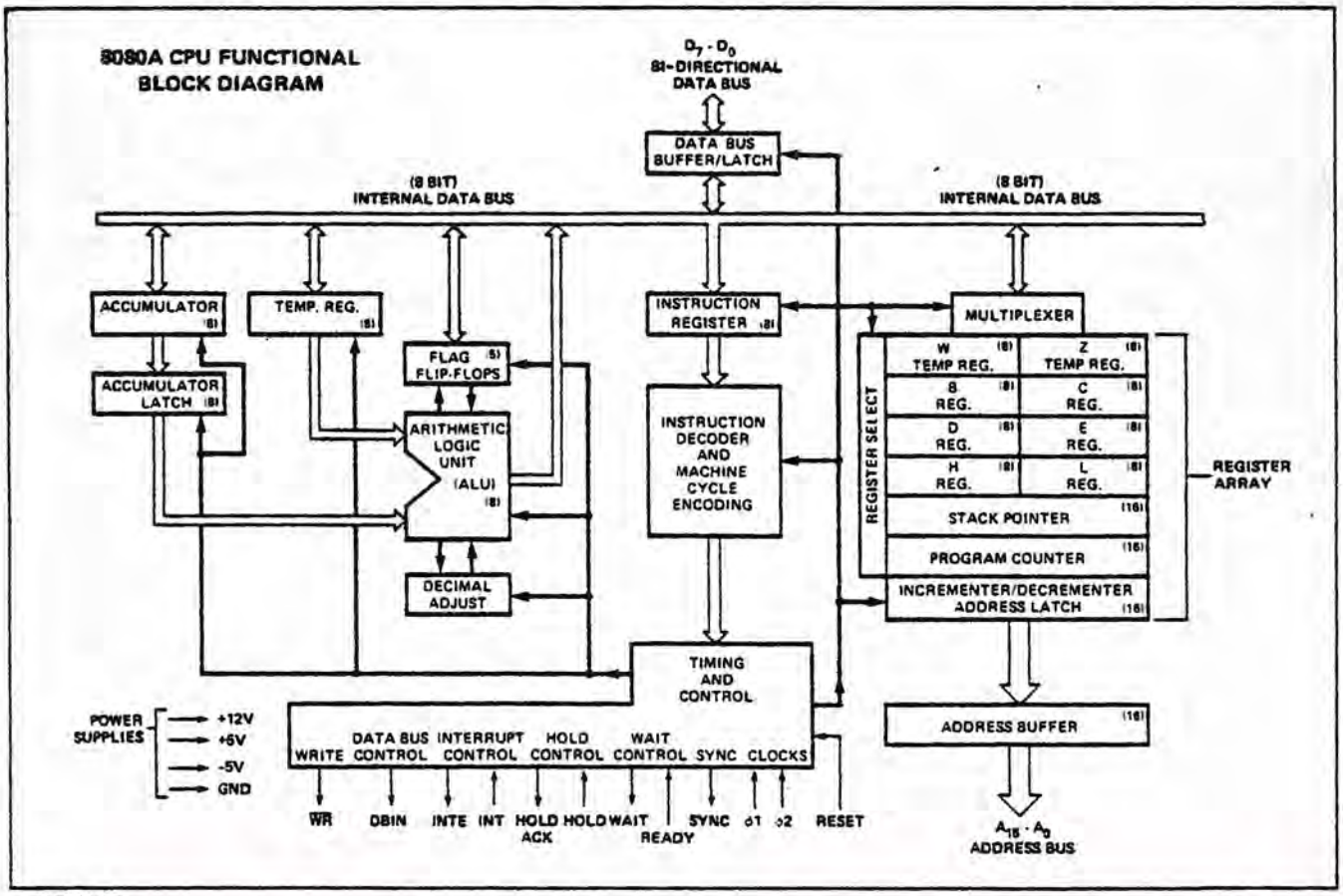
- **TTL Drive Capability**
- **2 μs Instruction Cycle**
- **Powerful Problem Solving Instruction Set**
- **Six General Purpose Registers and an Accumulator**
- **Sixteen Bit Program Counter for Directly Addressing up to 64K Bytes of Memory**
- **Sixteen Bit Stack Pointer and Stack Manipulation Instructions for Rapid Switching of the Program Environment**
- **Decimal, Binary and Double Precision Arithmetic**
- **Ability to Provide Priority Vectored Interrupts**
- **512 Directly Addressed I/O Ports**

The Intel® 8080A is a complete 8-bit parallel central processing unit (CPU). It is fabricated on a single LSI chip using Intel's n-channel silicon gate MOS process. This offers the user a high performance solution to control and processing applications.

The 8080A contains six 8-bit general purpose working registers and an accumulator. The six general purpose registers may be addressed individually or in pairs providing both single and double precision operators. Arithmetic and logical instructions set or reset four testable flags. A fifth flag provides decimal arithmetic operation.

The 8080A has an external stack feature wherein any portion of memory may be used as a last in/first out stack to store/retrieve the contents of the accumulator, flags, program counter and all of the six general purpose registers. The sixteen bit stack pointer controls the addressing of this external stack. This stack gives the 8080A the ability to easily handle multiple level priority interrupts by rapidly storing and restoring processor status. It also provides almost unlimited subroutine nesting.

This microprocessor has been designed to simplify systems design. Separate 16-line address and 8-line bi-directional data busses are used to facilitate easy interface to memory and I/O. Signals to control the interface to memory and I/O are provided directly by the 8080A. Ultimate control of the address and data busses resides with the HOLD signal. It provides the ability to suspend processor operation and force the address and data busses into a high impedance state. This permits OR-tying these busses with other controlling devices for (DMA) direct memory access or multi-processor operation.



8080A FUNCTIONAL PIN DEFINITION

The following describes the function of all of the 8080A I/O pins. Several of the descriptions refer to internal timing periods.

A_{15}-A_0 (output three-state)
ADDRESS BUS; the address bus provides the address to memory (up to 64K 8-bit words) or denotes the I/O device number for up to 256 input and 256 output devices. A_0 is the least significant address bit.

D_7-D_0 (input/output three-state)
DATA BUS; the data bus provides bi-directional communication between the CPU, memory, and I/O devices for instructions and data transfers. Also, during the first clock cycle of each machine cycle, the 8080A outputs a status word on the data bus that describes the current machine cycle. D_0 is the least significant bit.

SYNC (output)
SYNCHRONIZING SIGNAL; the SYNC pin provides a signal to indicate the beginning of each machine cycle.

DBIN (output)
DATA BUS IN; the DBIN signal indicates to external circuits that the data bus is in the input mode. This signal should be used to enable the gating of data onto the 8080A data bus from memory or I/O.

READY (input)
READY; the READY signal indicates to the 8080A that valid memory or input data is available on the 8080A data bus. This signal is used to synchronize the CPU with slower memory or I/O devices. If after sending an address out the 8080A does not receive a READY input, the 8080A will enter a WAIT state for as long as the READY line is low. READY can also be used to single step the CPU.

WAIT (output)
WAIT; the WAIT signal acknowledges that the CPU is in a WAIT state.

$\overline{WR}$ (output)
WRITE; the $\overline{WR}$ signal is used for memory WRITE or I/O output control. The data on the data bus is stable while the $\overline{WR}$ signal is active low ($\overline{WR}$ = 0).

HOLD (input)
HOLD; the HOLD signal requests the CPU to enter the HOLD state. The HOLD state allows an external device to gain control of the 8080A address and data bus as soon as the 8080A has completed its use of these buses for the current machine cycle. It is recognized under the following conditions:

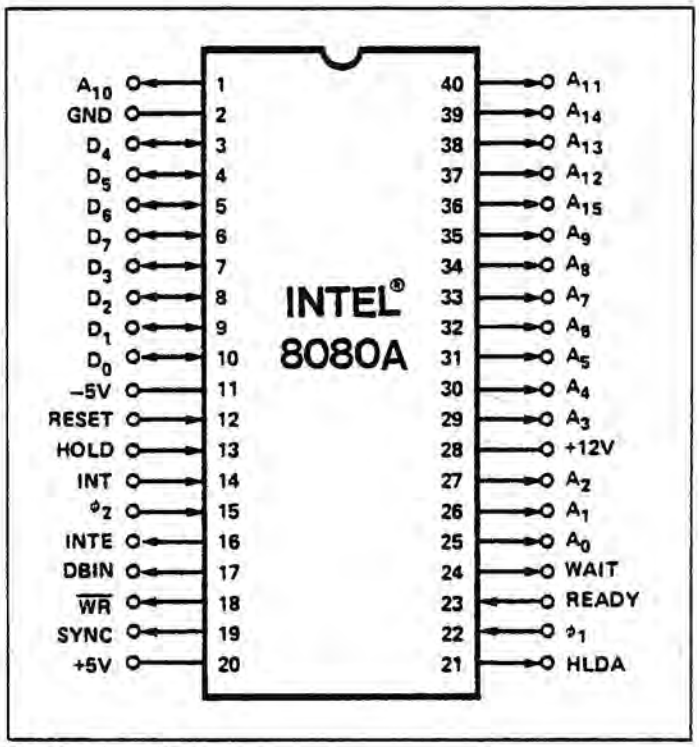
- the CPU is in the HALT state.
- the CPU is in the T2 or TW state and the READY signal is active.

As a result of entering the HOLD state the CPU ADDRESS BUS (A_{15}-A_0) and DATA BUS (D_7-D_0) will be in their high impedance state. The CPU acknowledges its state with the HOLD ACKNOWLEDGE (HLDA) pin.

HLDA (output)
HOLD ACKNOWLEDGE; the HLDA signal appears in response to the HOLD signal and indicates that the data and address bus will go to the high impedance state. The HLDA signal begins at:

- T3 for READ memory or input.
- The Clock Period following T3 for WRITE memory or OUTPUT operation.

In either case, the HLDA signal appears after the rising edge of ϕ_1 and high impedance occurs after the rising edge of ϕ_2.



Pin Configuration

INTE (output)
INTERRUPT ENABLE; indicates the content of the internal interrupt enable flip/flop. This flip/flop may be set or reset by the Enable and Disable Interrupt instructions and inhibits interrupts from being accepted by the CPU when it is reset. It is automatically reset (disabling further interrupts) at time T1 of the instruction fetch cycle (M1) when an interrupt is accepted and is also reset by the RESET signal.

INT (input)
INTERRUPT REQUEST; the CPU recognizes an interrupt request on this line at the end of the current instruction or while halted. If the CPU is in the HOLD state or if the Interrupt Enable flip/flop is reset it will not honor the request.

RESET (input) [1]
RESET; while the RESET signal is activated, the content of the program counter is cleared. After RESET, the program will start at location 0 in memory. The INTE and HLDA flip/flops are also reset. Note that the flags, accumulator, stack pointer, and registers are not cleared.

V_{SS}	Ground Reference.
V_{DD}	+12 ± 5% Volts.
V_{CC}	+5 ± 5% Volts.
V_{BB}	-5 ±5% Volts (substrate bias).
ϕ_1, ϕ_2	2 externally supplied clock phases. (non TTL compatible)

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias . . . . . 0°C to +70° C
Storage Temperature . . . . . -65°C to +150°C
All Input or Output Voltages
With Respect to V_{BB} . . . . . -0.3V to +20V
V_{CC}, V_{DD} and V_{SS} With Respect to V_{BB} -0.3V to +20V
Power Dissipation 1.5W

**COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{DD} = +12V ± 5%, V_{CC} = +5V ± 5%, V_{BB} = -5V ± 5%, V_{SS} = 0V, Unless Otherwise Noted.

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	$V_{SS}-1$		$V_{SS}+0.8$	V	
V_{IHC}	Clock Input High Voltage	9.0		$V_{DD}+1$	V	
V_{IL}	Input Low Voltage	$V_{SS}-1$		$V_{SS}+0.8$	V	
V_{IH}	Input High Voltage	3.3		$V_{CC}+1$	V	
V_{OL}	Output Low Voltage			0.45	V	I_{OL} = 1.9mA on all outputs, I_{OH} = -150μA.
V_{OH}	Output High Voltage	3.7			V	
$I_{DD\,(AV)}$	Avg. Power Supply Current (V_{DD})		40	70	mA	Operation T_{CY} = .48 μsec
$I_{CC\,(AV)}$	Avg. Power Supply Current (V_{CC})		60	80	mA	
$I_{BB\,(AV)}$	Avg. Power Supply Current (V_{BB})		.01	1	mA	
I_{IL}	Input Leakage			±10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{CL}	Clock Leakage			±10	μA	$V_{SS} \leq V_{CLOCK} \leq V_{DD}$
I_{DL} [2]	Data Bus Leakage in Input Mode			-100 -2.0	μA mA	$V_{SS} \leq V_{IN} \leq V_{SS}+0.8V$ $V_{SS}+0.8V \leq V_{IN} \leq V_{CC}$
I_{FL}	Address and Data Bus Leakage During HOLD			+10 -100	μA	$V_{ADDR/DATA} = V_{CC}$ $V_{ADDR/DATA} = V_{SS} + 0.45V$

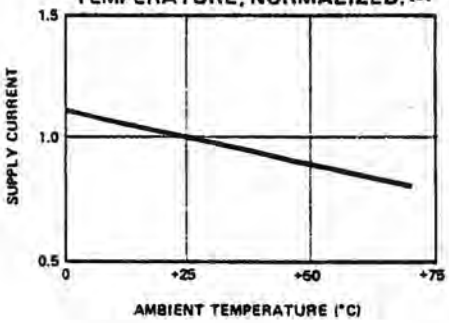
CAPACITANCE

T_A = 25°C $V_{CC} = V_{DD} = V_{SS}$ = 0V, V_{BB} = -5V

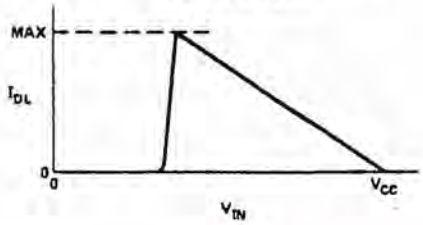
Symbol	Parameter	Typ.	Max.	Unit	Test Condition
C_ϕ	Clock Capacitance	17	25	pf	f_c = 1 MHz
C_{IN}	Input Capacitance	6	10	pf	Unmeasured Pins
C_{OUT}	Output Capacitance	10	20	pf	Returned to V_{SS}

NOTES:
1. The RESET signal must be active for a minimum of 3 clock cycles.
2. When DBIN is high and $V_{IN} > V_{IH}$ an internal active pull up will be switched onto the Data Bus.
3. ΔI supply / ΔT_A = -0.45%/°C.

TYPICAL SUPPLY CURRENT VS. TEMPERATURE, NORMALIZED. [3]



DATA BUS CHARACTERISTIC DURING DBIN



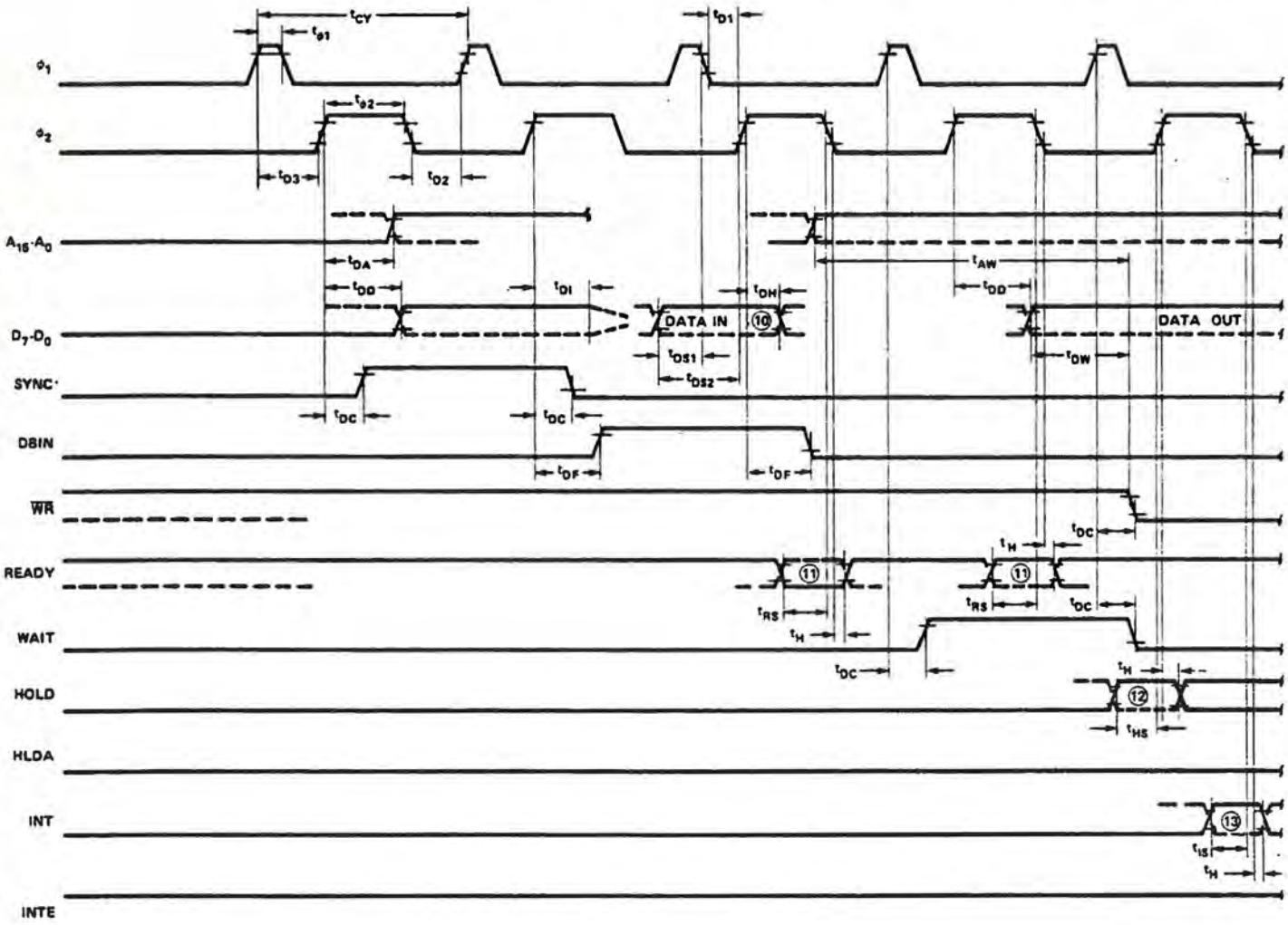
A.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{DD} = +12V ± 5%, V_{CC} = +5V ± 5%, V_{BB} = -5V ± 5%, V_{SS} = 0V, Unless Otherwise Noted

Symbol	Parameter	Min.	Max.	Unit	Test Condition
t_{CY} [3]	Clock Period	0.48	2.0	μsec	
t_r, t_f	Clock Rise and Fall Time	0	50	nsec	
$t_{\phi 1}$	ϕ_1 Pulse Width	60		nsec	
$t_{\phi 2}$	ϕ_2 Pulse Width	220		nsec	
t_{D1}	Delay ϕ_1 to ϕ_2	0		nsec	
t_{D2}	Delay ϕ_2 to ϕ_1	70		nsec	
t_{D3}	Delay ϕ_1 to ϕ_2 Leading Edges	80		nsec	
t_{DA} [2]	Address Output Delay From ϕ_2		200	nsec	C_L = 100pf
t_{DD} [2]	Data Output Delay From ϕ_2		220	nsec	C_L = 100pf
t_{DC} [2]	Signal Output Delay From ϕ_1, or ϕ_2 (SYNC, $\overline{WR}$, WAIT, HLDA)		120	nsec	C_L = 50pf
t_{DF} [2]	DBIN Delay From ϕ_2	25	140	nsec	C_L = 50pf
t_{DI} [1]	Delay for Input Bus to Enter Input Mode		t_{DF}	nsec	
t_{DS1}	Data Setup Time During ϕ_1 and DBIN	30		nsec	

TIMING WAVEFORMS [14]

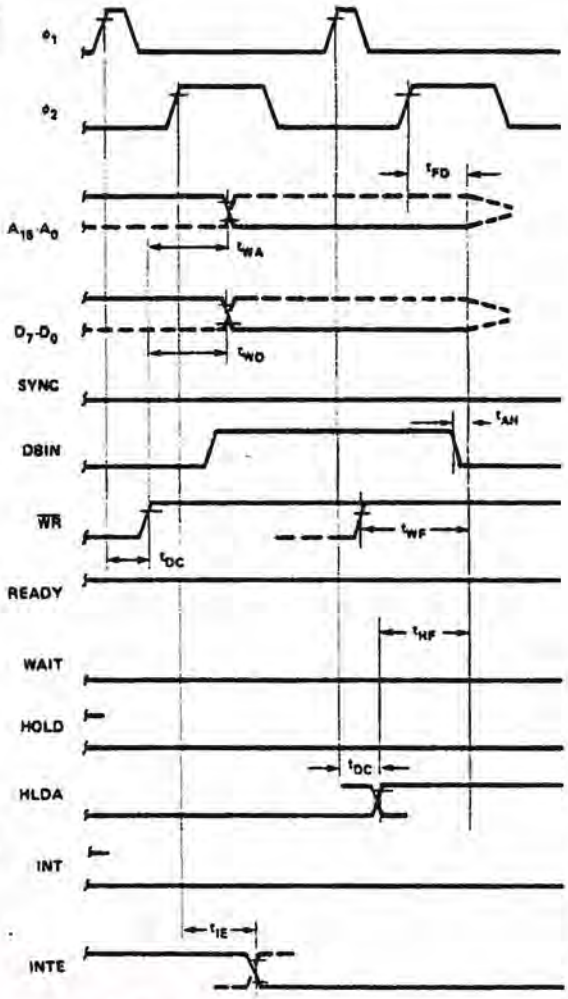
(Note: Timing measurements are made at the following reference voltages: CLOCK "1" = 8.0V "0" = 1.0V; INPUTS "1" = 3.3V, "0" = 0.8V; OUTPUTS "1" = 2.0V, "0" = 0.8V.)



A.C. CHARACTERISTICS (Continued)

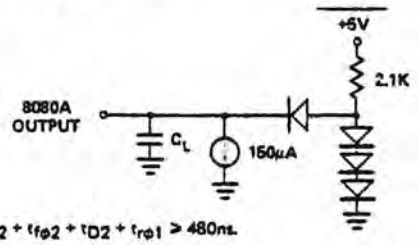
$T_A = 0°C$ to $70°C$, $V_{DD} = +12V \pm 5\%$, $V_{CC} = +5V \pm 5\%$, $V_{BB} = -5V \pm 5\%$, $V_{SS} = 0V$, Unless Otherwise Noted

Symbol	Parameter	Min.	Max.	Unit	Test Condition
t_{DS2}	Data Setup Time to ϕ_2 During DBIN	150		nsec	$C_L = 50pf$
t_{DH} [1]	Data Hold Time From ϕ_2 During DBIN	[1]		nsec	
t_{IE} [2]	INTE Output Delay From ϕ_2		200	nsec	
t_{RS}	READY Setup Time During ϕ_2	120		nsec	
t_{HS}	HOLD Setup Time to ϕ_2	140		nsec	
t_{IS}	INT Setup Time During ϕ_2 (During ϕ_1 in Halt Mode)	120		nsec	
t_H	Hold Time From ϕ_2 (READY, INT, HOLD)	0		nsec	
t_{FD}	Delay to Float During Hold (Address and Data Bus)		120	nsec	
t_{AW} [2]	Address Stable Prior to $\overline{WR}$	[5]		nsec	$C_L = 100pf$: Address, Data $C_L = 50pf$: $\overline{WR}$, HLDA, DBIN
t_{DW} [2]	Output Data Stable Prior to $\overline{WR}$	[6]		nsec	
t_{WD} [2]	Output Data Stable From $\overline{WR}$	[7]		nsec	
t_{WA} [2]	Address Stable From $\overline{WR}$	[7]		nsec	
t_{HF} [2]	HLDA to Float Delay	[8]		nsec	
t_{WF} [2]	$\overline{WR}$ to Float Delay	[9]		nsec	
t_{AH} [2]	Address Hold Time After DBIN During HLDA	-20		nsec	



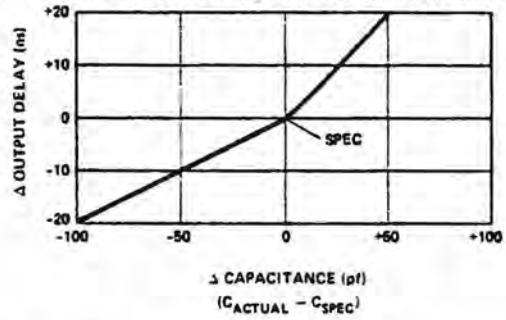
NOTES:

1. Data input should be enabled with DBIN status. No bus conflict can then occur and data hold time is assured. $t_{DH} = 50$ ns or t_{DF}, whichever is less.
2. Load Circuit.



3. $t_{CY} = t_{D3} + t_{r\phi2} + t_{\phi2} + t_{f\phi2} + t_{D2} + t_{r\phi1} > 480$ns.

TYPICAL Δ OUTPUT DELAY VS. Δ CAPACITANCE



4. The following are relevant when interfacing the 8080A to devices having $V_{IH} = 3.3V$:
   a) Maximum output rise time from .8V to 3.3V = 100ns @ C_L = SPEC.
   b) Output delay when measured to 3.0V = SPEC +60ns @ C_L = SPEC.
   c) If $C_L \neq$ SPEC, add .6ns/pF if $C_L > C_{SPEC}$, subtract .3ns/pF (from modified delay) if $C_L < C_{SPEC}$.
5. $t_{AW} = 2\,t_{CY} - t_{D3} - t_{r\phi2} - 140$nsec.
6. $t_{DW} = t_{CY} - t_{D3} - t_{r\phi2} - 170$nsec.
7. If not HLDA, $t_{WD} = t_{WA} = t_{D3} + t_{r\phi2} + 10$ns. If HLDA, $t_{WD} = t_{WA} = t_{WF}$.
8. $t_{HF} = t_{D3} + t_{r\phi2} - 50$ns.
9. $t_{WF} = t_{D3} + t_{r\phi2} - 10$ns
10. Data in must be stable for this period during DBIN ·T_3. Both t_{DS1} and t_{DS2} must be satisfied.
11. Ready signal must be stable for this period during T_2 or T_W. (Must be externally synchronized.)
12. Hold signal must be stable for this period during T_2 or T_W when entering hold mode, and during T_3, T_4, T_5 and T_{WH} when in hold mode. (External synchronization is not required.)
13. Interrupt signal must be stable during this period of the last clock cycle of any instruction in order to be recognized on the following instruction. (External synchronization is not required.)
14. This timing diagram shows timing relationships only; it does not represent any specific machine cycle.

INSTRUCTION SET

The accumulator group instructions include arithmetic and logical operators with direct, indirect, and immediate addressing modes.

Move, load, and store instruction groups provide the ability to move either 8 or 16 bits of data between memory, the six working registers and the accumulator using direct, indirect, and immediate addressing modes.

The ability to branch to different portions of the program is provided with jump, jump conditional, and computed jumps. Also the ability to call to and return from subroutines is provided both conditionally and unconditionally. The RESTART (or single byte call instruction) is useful for interrupt vector operation.

Double precision operators such as stack manipulation and double add instructions extend both the arithmetic and interrupt handling capability of the 8080A. The ability to increment and decrement memory, the six general registers and the accumulator is provided as well as extended increment and decrement instructions to operate on the register pairs and stack pointer. Further capability is provided by the ability to rotate the accumulator left or right through or around the carry bit.

Input and output may be accomplished using memory addresses as I/O ports or the directly addressed I/O provided for in the 8080A instruction set.

The following special instruction group completes the 8080A instruction set: the NOP instruction, HALT to stop processor execution and the DAA instructions provide decimal arithmetic capability. STC allows the carry flag to be directly set, and the CMC instruction allows it to be complemented. CMA complements the contents of the accumulator and XCHG exchanges the contents of two 16-bit register pairs directly.

Data and Instruction Formats

Data in the 8080A is stored in the form of 8-bit binary integers. All data transfers to the system data bus will be in the same format.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

DATA WORD

The program instructions may be one, two, or three bytes in length. Multiple byte instructions must be stored in successive words in program memory. The instruction formats then depend on the particular operation executed.

		TYPICAL INSTRUCTIONS
One Byte Instructions		
D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	OP CODE	Register to register, memory reference, arithmetic or logical, rotate, return, push, pop, enable or disable Interrupt instructions
Two Byte Instructions		
D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	OP CODE	
D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	OPERAND	Immediate mode or I/O instructions
Three Byte Instructions		
D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	OP CODE	Jump, call or direct load and store instructions
D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	LOW ADDRESS OR OPERAND 1	
D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	HIGH ADDRESS OR OPERAND 2	

For the 8080A a logic "1" is defined as a high level and a logic "0" is defined as a low level.

INSTRUCTION SET

Summary of Processor Instructions

Mnemonic	Description	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
		Instruction Code[1]								
$MOV_{r1,r2}$	Move register to register	0	1	D	D	D	S	S	S	5
MOV M, r	Move register to memory	0	1	1	1	0	S	S	S	7
MOV r, M	Move memory to register	0	1	D	D	D	1	1	0	7
HLT	Halt	0	1	1	1	0	1	1	0	7
MVI r	Move immediate register	0	0	D	D	D	1	1	0	7
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	10
INR r	Increment register	0	0	D	D	D	1	0	0	5
DCR r	Decrement register	0	0	D	D	D	1	0	1	5
INR M	Increment memory	0	0	1	1	0	1	0	0	10
DCR M	Decrement memory	0	0	1	1	0	1	0	1	10
ADD r	Add register to A	1	0	0	0	0	S	S	S	4
ADC r	Add register to A with carry	1	0	0	0	1	S	S	S	4
SUB r	Subtract register from A	1	0	0	1	0	S	S	S	4
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4
ANA r	And register with A	1	0	1	0	0	S	S	S	4
XRA r	Exclusive Or register with A	1	0	1	0	1	S	S	S	4
ORA r	Or register with A	1	0	1	1	0	S	S	S	4
CMP r	Compare register with A	1	0	1	1	1	S	S	S	4
ADD M	Add memory to A	1	0	0	0	0	1	1	0	7
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	7
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	7
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
ANA M	And memory with A	1	0	1	0	0	1	1	0	7
XRA M	Exclusive Or memory with A	1	0	1	0	1	1	1	0	7
ORA M	Or memory with A	1	0	1	1	0	1	1	0	7
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	7
ADI	Add immediate to A	1	1	0	0	0	1	1	0	7
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
XRI	Exclusive Or immediate with A	1	1	1	0	1	1	1	0	7
ORI	Or immediate with A	1	1	1	1	0	1	1	0	7
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7
RLC	Rotate A left	0	0	0	0	0	1	1	1	4
RRC	Rotate A right	0	0	0	0	1	1	1	1	4
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JC	Jump on carry	1	1	0	1	1	0	1	0	10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	10
JZ	Jump on zero	1	1	0	0	1	0	1	0	10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	10
JP	Jump on positive	1	1	1	1	0	0	1	0	10
JM	Jump on minus	1	1	1	1	1	0	1	0	10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	10
CALL	Call unconditional	1	1	0	0	1	1	0	1	17
CC	Call on carry	1	1	0	1	1	1	0	0	11/17
CNC	Call on no carry	1	1	0	1	0	1	0	0	11/17
CZ	Call on zero	1	1	0	0	1	1	0	0	11/17
CNZ	Call on no zero	1	1	0	0	0	1	0	0	11/17
CP	Call on positive	1	1	1	1	0	1	0	0	11/17
CM	Call on minus	1	1	1	1	1	1	0	0	11/17
CPE	Call on parity even	1	1	1	0	1	1	0	0	11/17
CPO	Call on parity odd	1	1	1	0	0	1	0	0	11/17
RET	Return	1	1	0	0	1	0	0	1	10
RC	Return on carry	1	1	0	1	1	0	0	0	5/11
RNC	Return on no carry	1	1	0	1	0	0	0	0	5/11
RZ	Return on zero	1	1	0	0	1	0	0	0	5/11
RNZ	Return on no zero	1	1	0	0	0	0	0	0	5/11
RP	Return on positive	1	1	1	1	0	0	0	0	5/11
RM	Return on minus	1	1	1	1	1	0	0	0	5/11
RPE	Return on parity even	1	1	1	0	1	0	0	0	5/11
RPO	Return on parity odd	1	1	1	0	0	0	0	0	5/11
RST	Restart	1	1	A	A	A	1	1	1	11
IN	Input	1	1	0	1	1	0	1	1	10
OUT	Output	1	1	0	1	0	0	1	1	10
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	10
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	10
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	10
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	11
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	11
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	11
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	11
POP B	Pop register pair B & C off stack	1	1	0	0	0	0	0	1	10
POP D	Pop register pair D & E off stack	1	1	0	1	0	0	0	1	10
POP H	Pop register pair H & L off stack	1	1	1	0	0	0	0	1	10
POP PSW	Pop A and Flags off stack	1	1	1	1	0	0	0	1	10
STA	Store A direct	0	0	1	1	0	0	1	0	13
LDA	Load A direct	0	0	1	1	1	0	1	0	13
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	4
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	18
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	5
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	5
DAD B	Add B & C to H & L	0	0	0	0	1	0	0	1	10
DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1	10
DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1	10
DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	10
STAX B	Store A indirect	0	0	0	0	0	0	1	0	7
STAX D	Store A indirect	0	0	0	1	0	0	1	0	7
LDAX B	Load A indirect	0	0	0	0	1	0	1	0	7
LDAX D	Load A indirect	0	0	0	1	1	0	1	0	7
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	5
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	5
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	5
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	5
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	5
DCX D	Decrement D & E	0	0	0	1	1	0	1	1	5
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	5
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	5
CMA	Complement A	0	0	1	0	1	1	1	1	4
STC	Set carry	0	0	1	1	0	1	1	1	4
CMC	Complement carry	0	0	1	1	1	1	1	1	4
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	16
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	16
EI	Enable interrupts	1	1	1	1	1	0	1	1	4
DI	Disable interrupt	1	1	1	1	0	0	1	1	4
NOP	No-operation	0	0	0	0	0	0	0	0	4

NOTES: 1. DDD or SSS – 000 B – 001 C – 010 D – 011 E – 100 H – 101 L – 110 Memory – 111 A.
2. Two possible cycle times, (5/11) indicate instruction cycles dependent on condition flags.

Schottky Bipolar 8212

EIGHT-BIT INPUT/OUTPUT PORT

- **Fully Parallel 8-Bit Data Register and Buffer**
- **Service Request Flip-Flop for Interrupt Generation**
- **Low Input Load Current — .25 mA Max.**
- **Three State Outputs**
- **Outputs Sink 15 mA**
- **3.65V Output High Voltage for Direct Interface to 8080 CPU or 8008 CPU**
- **Asynchronous Register Clear**
- **Replaces Buffers, Latches and Multiplexers in Microcomputer Systems**
- **Reduces System Package Count**

The 8212 input/output port consists of an 8-bit latch with 3-state output buffers along with control and device selection logic. Also included is a service request flip-flop for the generation and control of interrupts to the microprocessor.

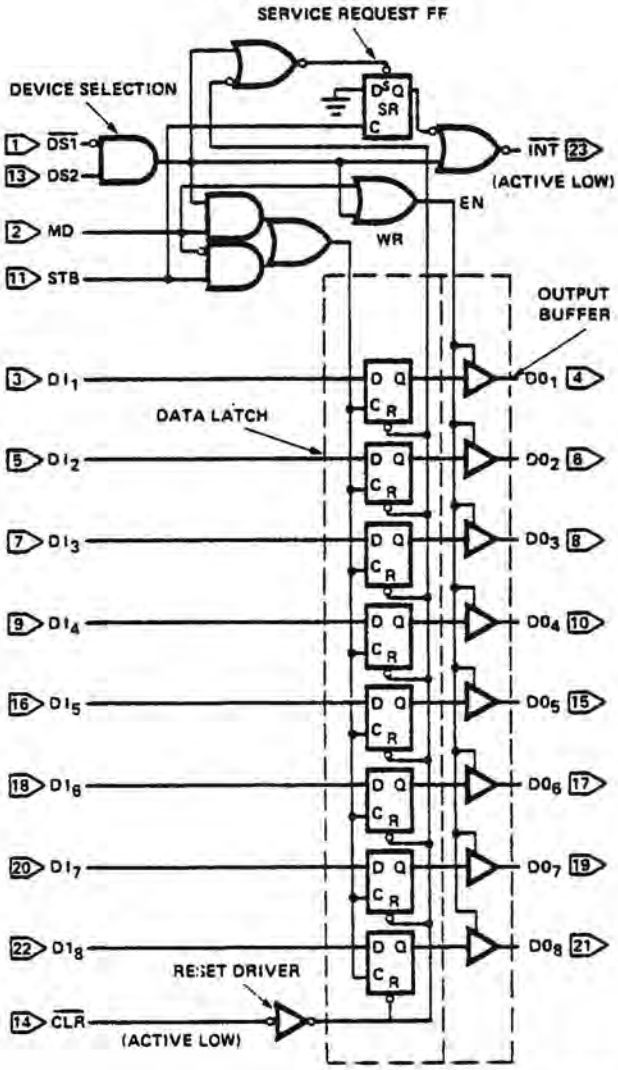
The device is multimode in nature. It can be used to implement latches, gated buffers or multiplexers. Thus, all of the principal peripheral and input/output functions of a microcomputer system can be implemented with this device.

PIN CONFIGURATION

PIN NAMES

DI_1-DI_8	DATA IN
DO_1-DO_8	DATA OUT
$\overline{DS_1}$-DS_2	DEVICE SELECT
MD	MODE
STB	STROBE
$\overline{INT}$	INTERRUPT (ACTIVE LOW)
$\overline{CLR}$	CLEAR (ACTIVE LOW)

LOGIC DIAGRAM



Functional Description

Data Latch

The 8 flip-flops that make up the data latch are of a "D" type design. The output (Q) of the flip-flop will follow the data input (D) while the clock input (C) is high. Latching will occur when the clock (C) returns low.

The data latch is cleared by an asynchronous reset input ($\overline{CLR}$). (Note: Clock (C) Overides Reset ($\overline{CLR}$).)

Output Buffer

The outputs of the data latch (Q) are connected to 3-state, non-inverting output buffers. These buffers have a common control line (EN); this control line either enables the buffer to transmit the data from the outputs of the data latch (Q) or disables the buffer, forcing the output into a high impedance state. (3-state)

This high-impedance state allows the designer to connect the 8212 directly onto the microprocessor bi-directional data bus.

Control Logic

The 8212 has control inputs $\overline{DS1}$, DS2, MD and STB. These inputs are used to control device selection, data latching, output buffer state and service request flip-flop.

$\overline{DS1}$, DS2 (Device Select)

These 2 inputs are used for device selection. When $\overline{DS1}$ is low and DS2 is high ($\overline{DS1} \cdot DS2$) the device is selected. In the selected state the output buffer is enabled and the service request flip-flop (SR) is asynchronously set.

MD (Mode)

This input is used to control the state of the output buffer and to determine the source of the clock input (C) to the data latch.

When MD is high (output mode) the output buffers are enabled and the source of clock (C) to the data latch is from the device selection logic ($\overline{DS1} \cdot DS2$).

When MD is low (input mode) the output buffer state is determined by the device selection logic ($\overline{DS1} \cdot DS2$) and the source of clock (C) to the data latch is the STB (Strobe) input.

STB (Strobe)

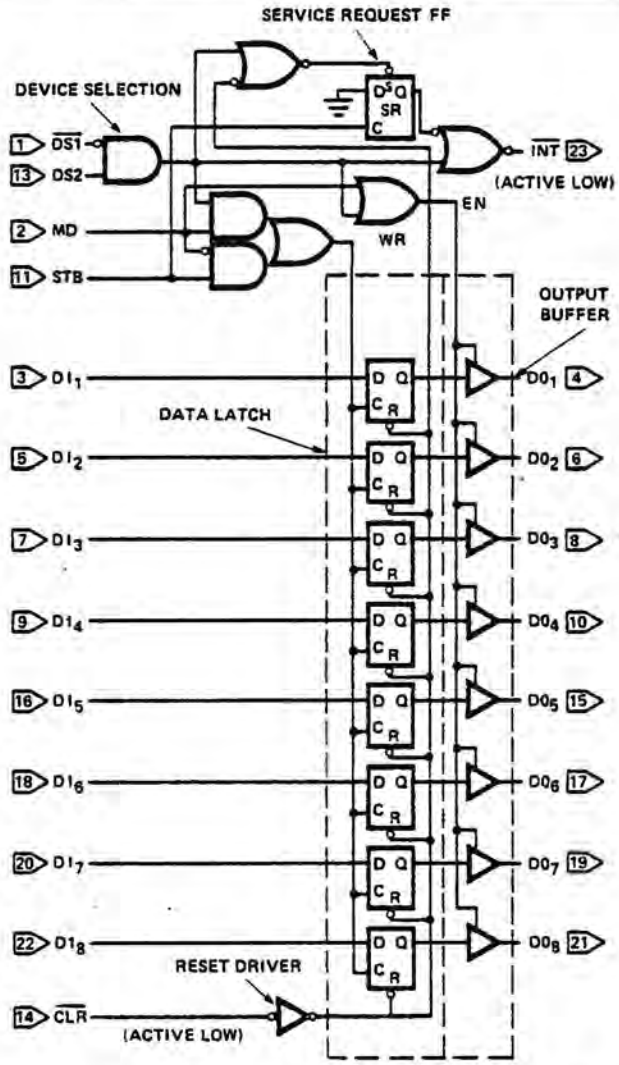
This input is used as the clock (C) to the data latch for the input mode MD = 0) and to synchronously reset the service request flip-flop (SR).

Note that the SR flip-flop is negative edge triggered.

Service Request Flip-Flop

The (SR) flip-flop is used to generate and control interrupts in microcomputer systems. It is asynchronously set by the $\overline{CLR}$ input (active low). When the (SR) flip-flop is set it is in the non-interrupting state.

The output of the (SR) flip-flop (Q) is connected to an inverting input of a "NOR" gate. The other input to the "NOR" gate is non-inverting and is connected to the device selection logic ($\overline{DS1} \cdot DS2$). The output of the "NOR" gate ($\overline{INT}$) is active low (interrupting state) for connection to active low input priority generating circuits.



STB	MD	($\overline{DS_1}$·DS_2)	DATA OUT EQUALS
0	0	0	3-STATE
1	0	0	3-STATE
0	1	0	DATA LATCH
1	1	0	DATA LATCH
0	0	1	DATA LATCH
1	0	1	DATA IN
0	1	1	DATA IN
1	1	1	DATA IN

$\overline{CLR}$ – RESETS DATA LATCH
SETS SR FLIP-FLOP
(NO EFFECT ON OUTPUT BUFFER)

CLR	($\overline{DS_1}$·DS_2)	STB	*SR	INT
0	0	0	1	1
0	1	0	1	0
1	1	↑	0	0
1	1	0	1	0
1	0	0	1	1
1	1	↑	1	0

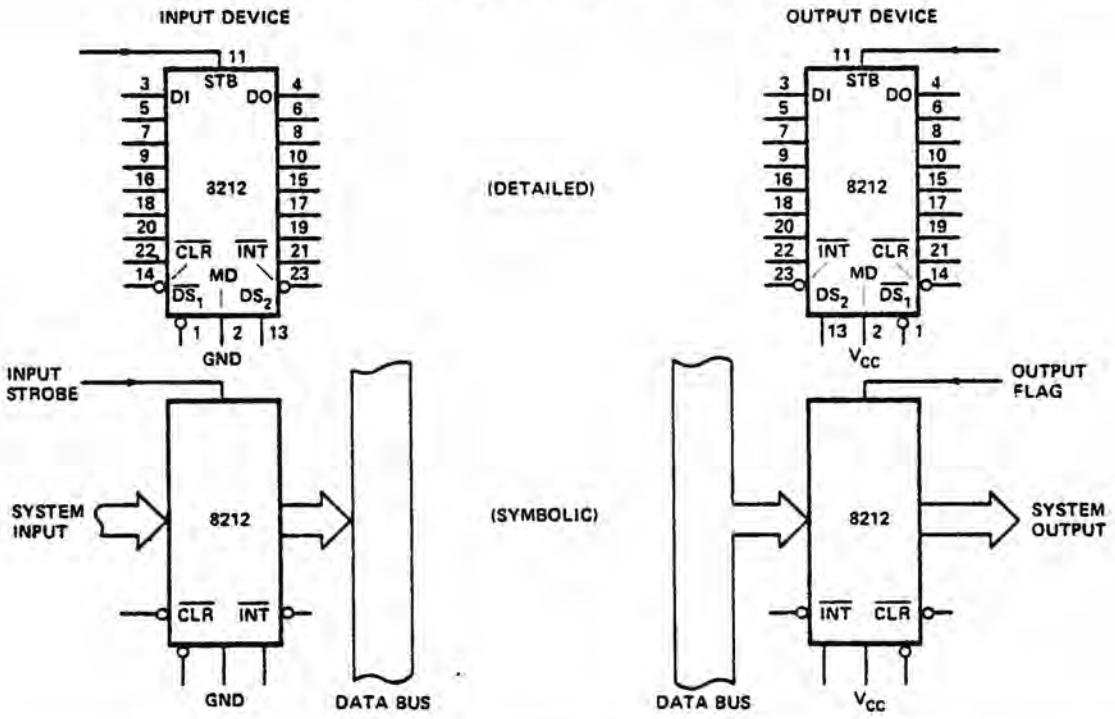
*INTERNAL SR FLIP-FLOP

Applications Of The 8212 -- For Microcomputer Systems

I. Basic Schematic Symbols

Two examples of ways to draw the 8212 on system schematics—(1) the top being the detailed view showing pin numbers, and (2) the bottom being the symbolic view showing the system input or output as a system bus (bus containing 8 parallel lines). The output to the data bus is symbolic in referencing 8 parallel lines.

BASIC SCHEMATIC SYMBOLS



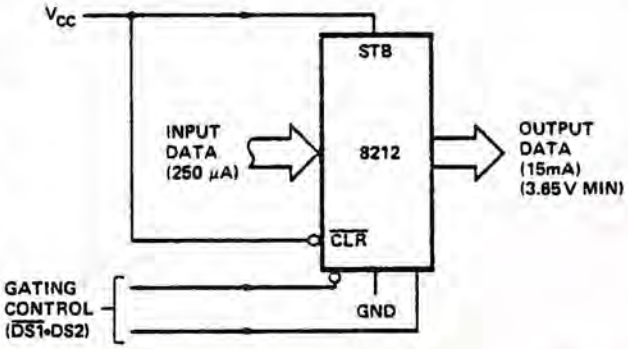
II. Gated Buffer (3 - STATE)

The simplest use of the 8212 is that of a gated buffer. By tying the mode signal low and the strobe input high, the data latch is acting as a straight through gate. The output buffers are then enabled from the device selection logic $\overline{DS1}$ and DS2.

When the device selection logic is false, the outputs are 3-state.

When the device selection logic is true, the input data from the system is directly transferred to the output. The input data load is 250 micro amps. The output data can sink 15 milli amps. The minimum high output is 3.65 volts.

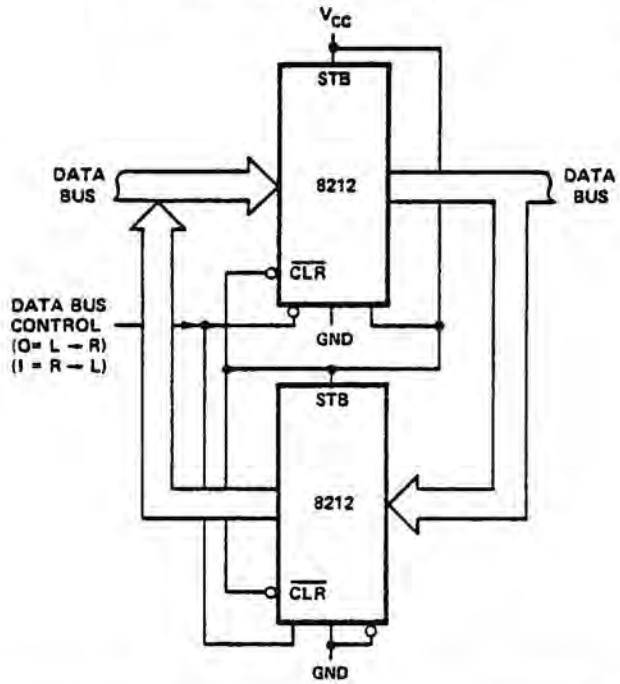
GATED BUFFER
3-STATE



III. Bi-Directional Bus Driver

A pair of 8212's wired (back-to-back) can be used as a symmetrical drive, bi-directional bus driver. The devices are controlled by the data bus input control which is connected to $\overline{DS1}$ on the first 8212 and to DS2 on the second. One device is active, and acting as a straight through buffer the other is in 3-state mode. This is a very useful circuit in small system design.

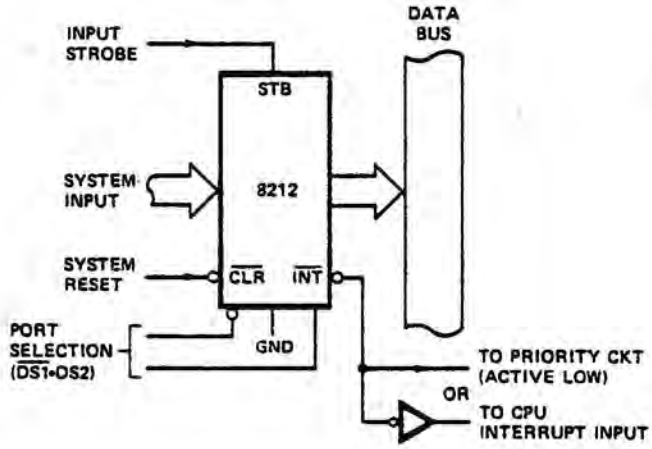
BI-DIRECTIONAL BUS DRIVER



IV. Interrupting Input Port

This use of an 8212 is that of a system input port that accepts a strobe from the system input source, which in turn clears the service request flip-flop and interrupts the processor. The processor then goes through a service routine, identifies the port, and causes the device selection logic to go true — enabling the system input data onto the data bus.

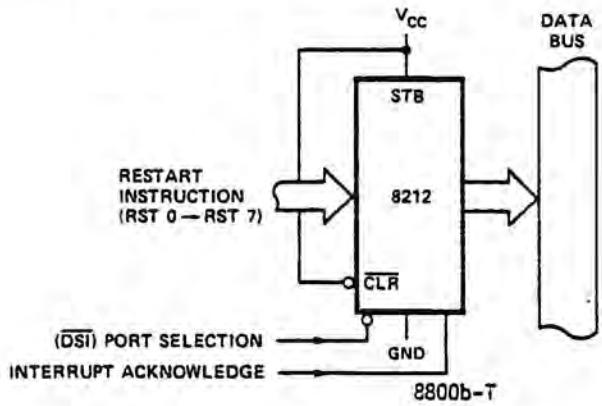
INTERRUPTING INPUT PORT



V. Interrupt Instruction Port

The 8212 can be used to gate the interrupt instruction, normally RESTART instructions, onto the data bus. The device is enabled from the interrupt acknowledge signal from the microprocessor and from a port selection signal. This signal is normally tied to ground. ($\overline{DS1}$ could be used to multiplex a variety of interrupt instruction ports onto a common bus).

INTERRUPT INSTRUCTION PORT

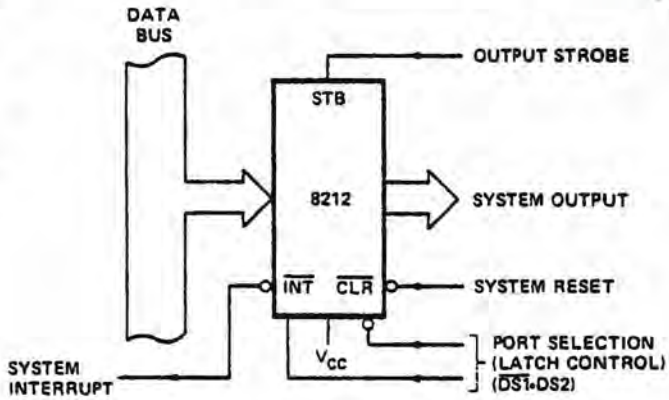


8800b-T
August, 1977

VI. Output Port (With Hand-Shaking)

The 8212 can be used to transmit data from the data bus to a system output. The output strobe could be a hand-shaking signal such as "reception of data" from the device that the system is outputting to. It in turn, can interrupt the system signifying the reception of data. The selection of the port comes from the device selection logic. ($\overline{DS1} \cdot DS2$)

OUTPUT PORT (WITH HAND-SHAKING)



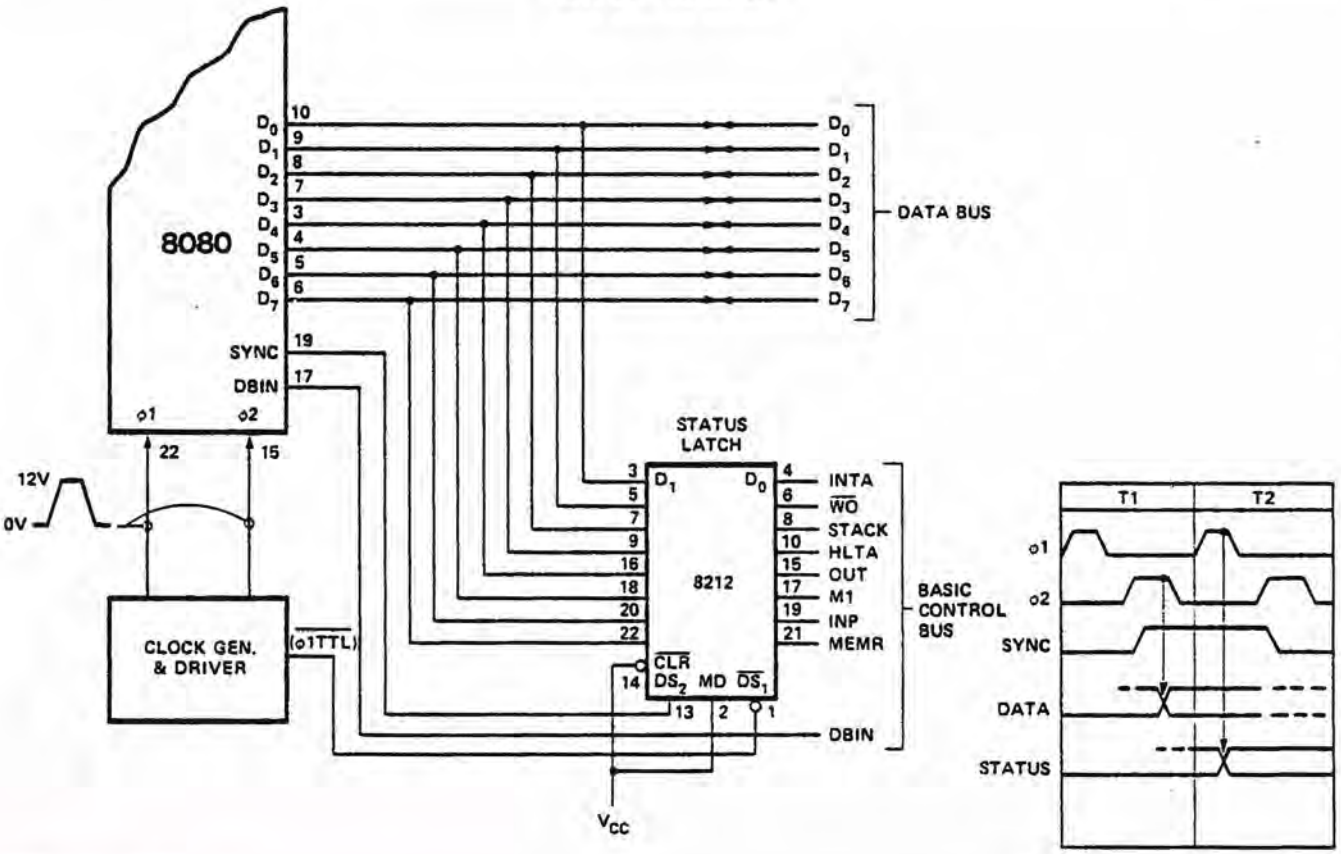
VII. 8080 Status Latch

Here the 8212 is used as the status latch for an 8080 microcomputer system. The input to the 8212 latch is directly from the 8080 data bus. Timing shows that when the SYNC signal is true, which is connected to the DS2 input and the phase 1 signal is true, which is a TTL level coming from the clock generator; then, the status data will be latched into the 8212.

Note: The mode signal is tied high so that the output on the latch is active and enabled all the time.

It is shown that the two areas of concern are the bidirectional data bus of the microprocessor and the control bus.

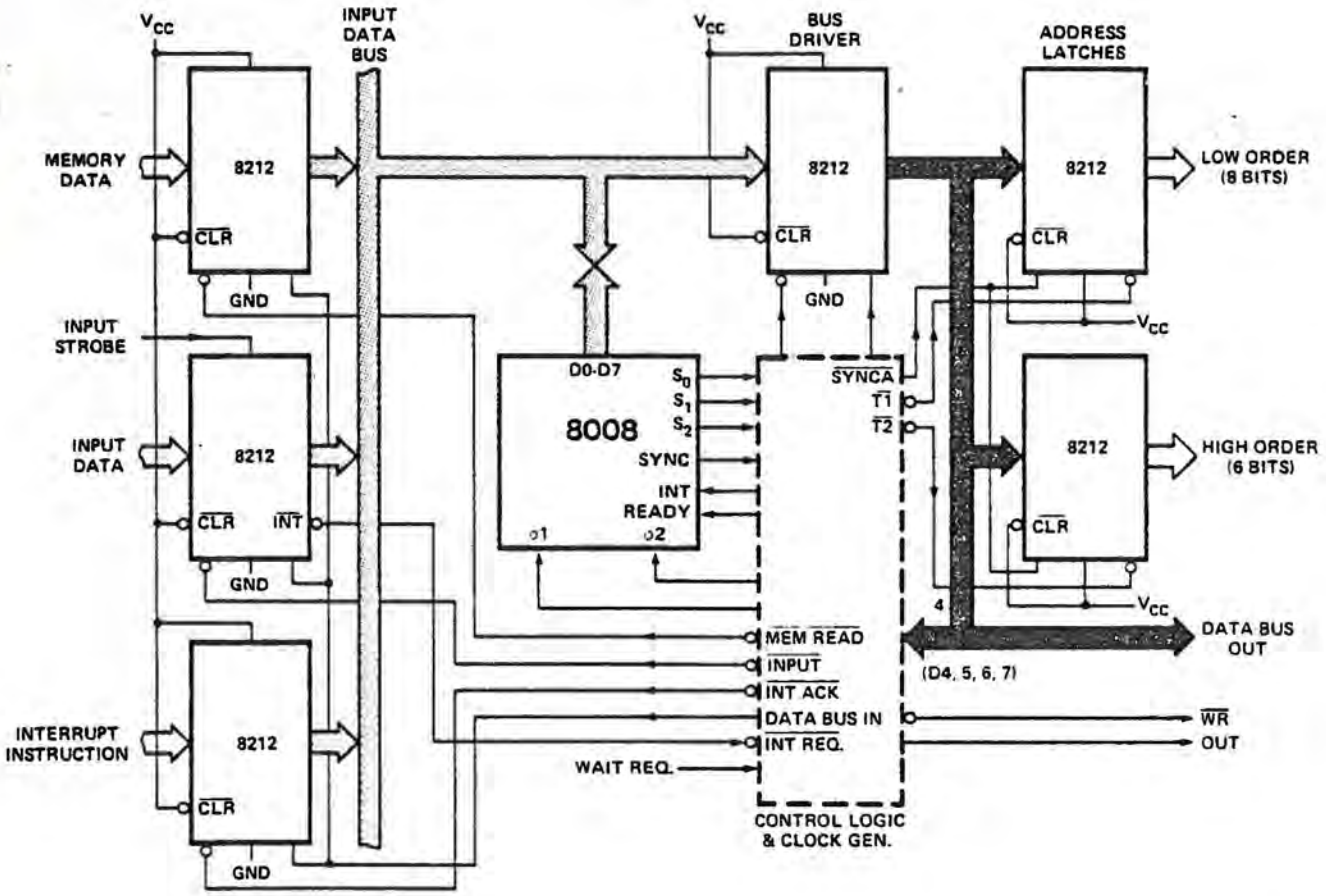
8080 STATUS LATCH



VIII. 8008 System

This shows the 8212 used in an 8008 microcomputer system. They are used to multiplex the data from three different sources onto the 8008 input data bus. The three sources of data are: memory data, input data, and the interrupt instruction. The 8212 is also used as the uni-directional bus driver to provide a proper drive to the address latches (both low order and high order are also 8212's) and to provide adequate drive to the output data bus. The control of these six 8212's in the 8008 system is provided by the control logic and clock generator circuits. These circuits consist of flip-flops, decoders, and gates to generate the control functions necessary for 8008 microcomputer systems. Also note that the input data port has a strobe input. This allows the processor to be interrupted from the input port directly. The control of the input bus consists of the data bus input signal, control logic, and the appropriate status signal for bus discipline whether memory read, input, or interrupt acknowledge. The combination of these four signals determines which one of these three devices will have access to the input data bus. The bus driver, which is implemented in an 8212, is also controlled by the control logic and clock generator so it can be 3-stated when necessary and also as a control transmission device to the address latches. Note: The address latches can be 3-stated for DMA purposes and they provide 15 milli amps drive, sufficient for large bus systems.

8008 SYSTEM



IX. 8080 System

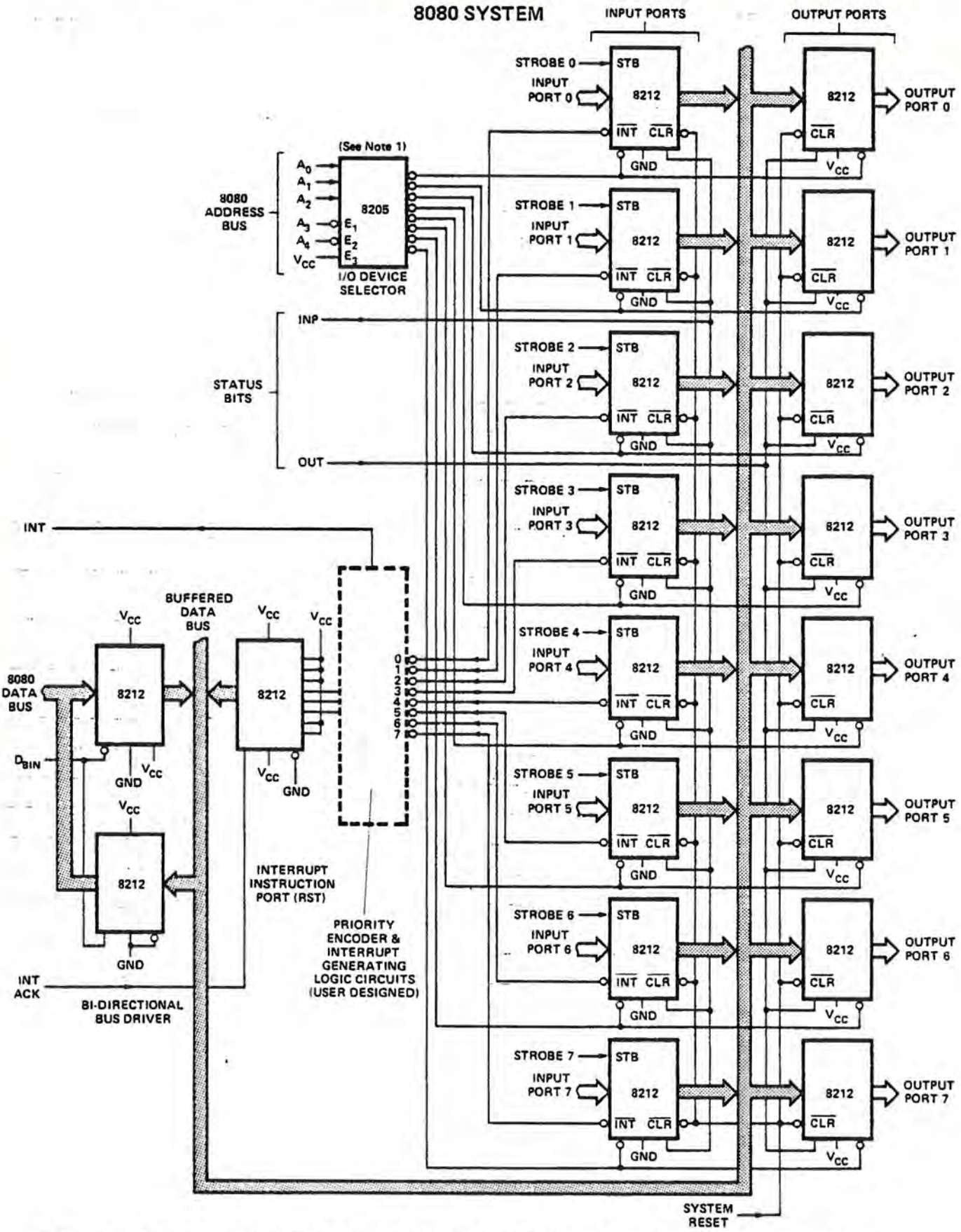
This drawing shows the 8212 used in the I/O section of an 8080 microcomputer system. The system consists of 8 input ports, 8 output ports, 8 level priority systems, and a bidirectional bus driver. (The data bus within the system is darkened for emphasis).

Basically, the operation would be as follows: The 8 ports, for example, could be connected to 8 keyboards, each keyboard having its own priority level. The keyboard could provide a strobe input of its own which would clear the service request flip-flop. The $\overline{INT}$ signals are connected to an 8 level priority encoding circuit. This circuit provides a positive true level to the central processor (INT) along with a three-bit code to the interrupt instruction port for the generation of RESTART instructions. Once the processor has been interrupted and it acknowledges the reception of the interrupt, the Interrupt Acknowledge signal is generated. This signal transfers data in the form of a RESTART instruction onto the buffered data bus. When the DBIN signal is true this RESTART instruction is gated into the microcomputer, in this case, the 8080 CPU. The 8080 then performs a software controlled interrupt service routine, saving the status of its current operation in the push-down stack and performing an INPUT instruction. The INPUT instruction thus sets the INP status bit, which is common to all input ports.

Also present is the address of the device on the 8080 address bus which in this system is connected to an 8205, one out of eight decoder with active low outputs. These active low outputs will enable one of the input ports, the one that interrupted the processor, to put its data onto the buffered data bus to be transmitted to the CPU when the data bus input signal is true. The processor can also output data from the 8080 data bus to the buffered data bus when the data bus input signal is false. Using the same address selection technique from the 8205 decoder and the output status bit, we can select with this system one of eight output ports to transmit the data to the system's output device structure.

Note: This basic I/O configuration for the 8080 can be expanded to 256 input devices and 256 output devices all using 8212 and, of course, the appropriate decoding.

Note that the 8080 is a 3.3-volt minimum high input requirement and that the 8212 has a 3.65-volt minimum high output providing the designer with a 350 milli volt noise margin worst case for 8080 systems when using the 8212.



Note 1. This basic I/O configuration for the 8080 can be expanded to 256 input devices and 256 output devices all using 8212 and the appropriate decoding.

Absolute Maximum Ratings*

Temperature Under Bias Plastic . . −65°C to +75°C

Storage Temperature . . . . . −65°C to +160°C

All Output or Supply Voltages . . . . −0.5 to +7 Volts

All Input Voltages . . . . . −1.0 to 5.5 Volts

Output Currents . . . . . 125 mA

*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operational sections of this specification is not implied.

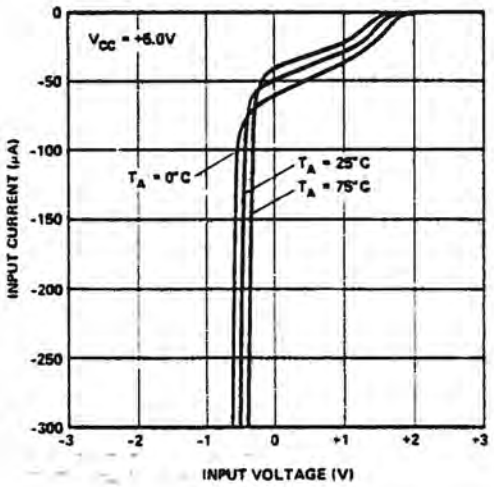
D.C. Characteristics

T_A = 0°C to +75°C V_{CC} = +5V ±5%

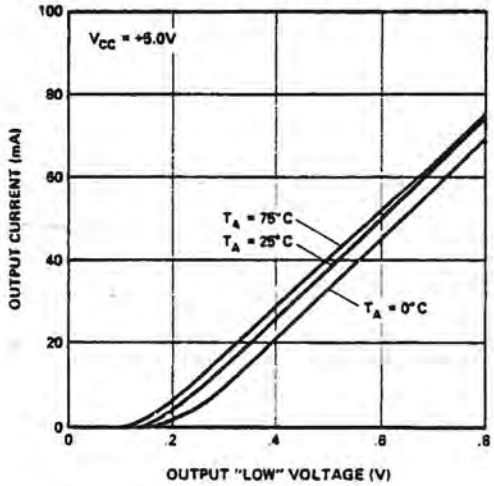
Symbol	Parameter	Limits Min.	Limits Typ.	Limits Max.	Unit	Test Conditions
I_F	Input Load Current ACK, DS_2, CR, DI_1-DI_8 Inputs			−.25	mA	V_F = .45V
I_F	Input Load Current MD Input			−.75	mA	V_F = .45V
I_F	Input Load Current DS_1 Input			−1.0	mA	V_F = .45V
I_R	Input Leakage Current ACK, DS, CR, DI_1-DI_8 Inputs			10	μA	V_R = 5.25V
I_R	Input Leakage Current MO Input			30	μA	V_R = 5.25V
I_R	Input Leakage Current DS_1 Input			40	μA	V_R = 5.25V
V_C	Input Forward Voltage Clamp			−1	V	I_C = −5 mA
V_{IL}	Input "Low" Voltage			.85	V	
V_{IH}	Input "High" Voltage	2.0			V	
V_{OL}	Output "Low" Voltage			.45	V	I_{OL} = 15 mA
V_{OH}	Output "High" Voltage	3.65	4.0		V	I_{OH} = −1 mA
I_{SC}	Short Circuit Output Current	-15		-75	mA	V_O = 0 V
$\|I_O\|$	Output Leakage Current High Impedance State			20	μA	V_O = .45V/5.25V
I_{CC}	Power Supply Current		90	130	mA	

Typical Characteristics

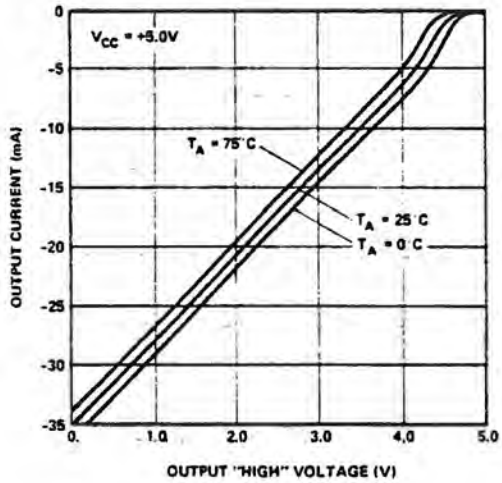
INPUT CURRENT VS. INPUT VOLTAGE



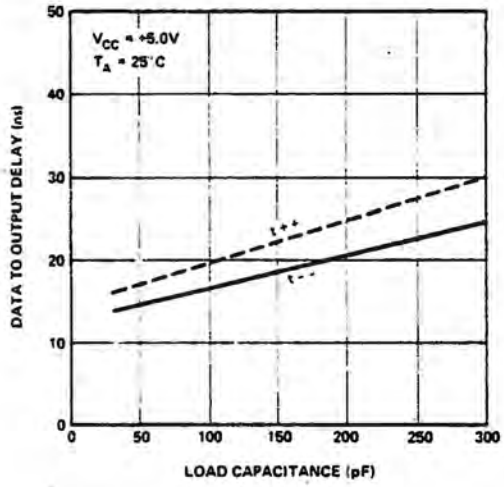
OUTPUT CURRENT VS. OUTPUT "LOW" VOLTAGE



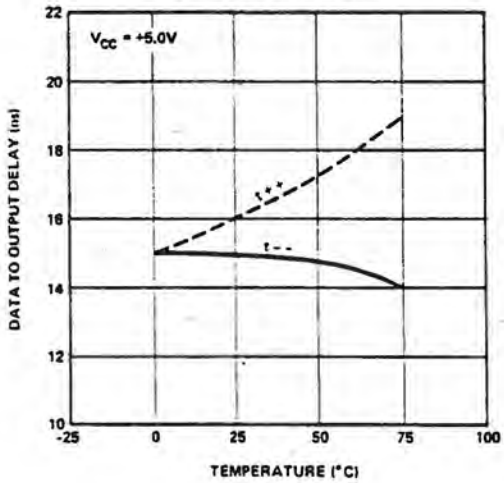
OUTPUT CURRENT VS. OUTPUT "HIGH" VOLTAGE



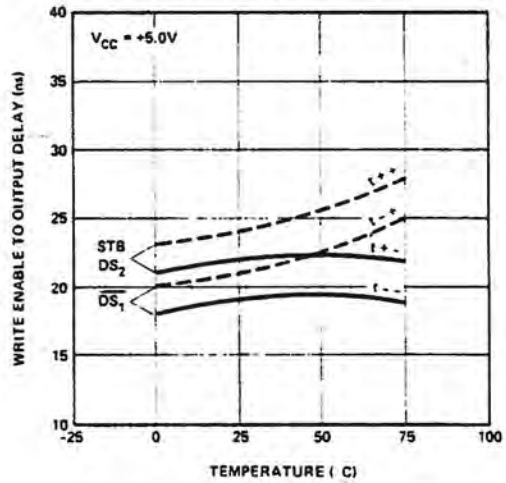
DATA TO OUTPUT DELAY VS. LOAD CAPACITANCE



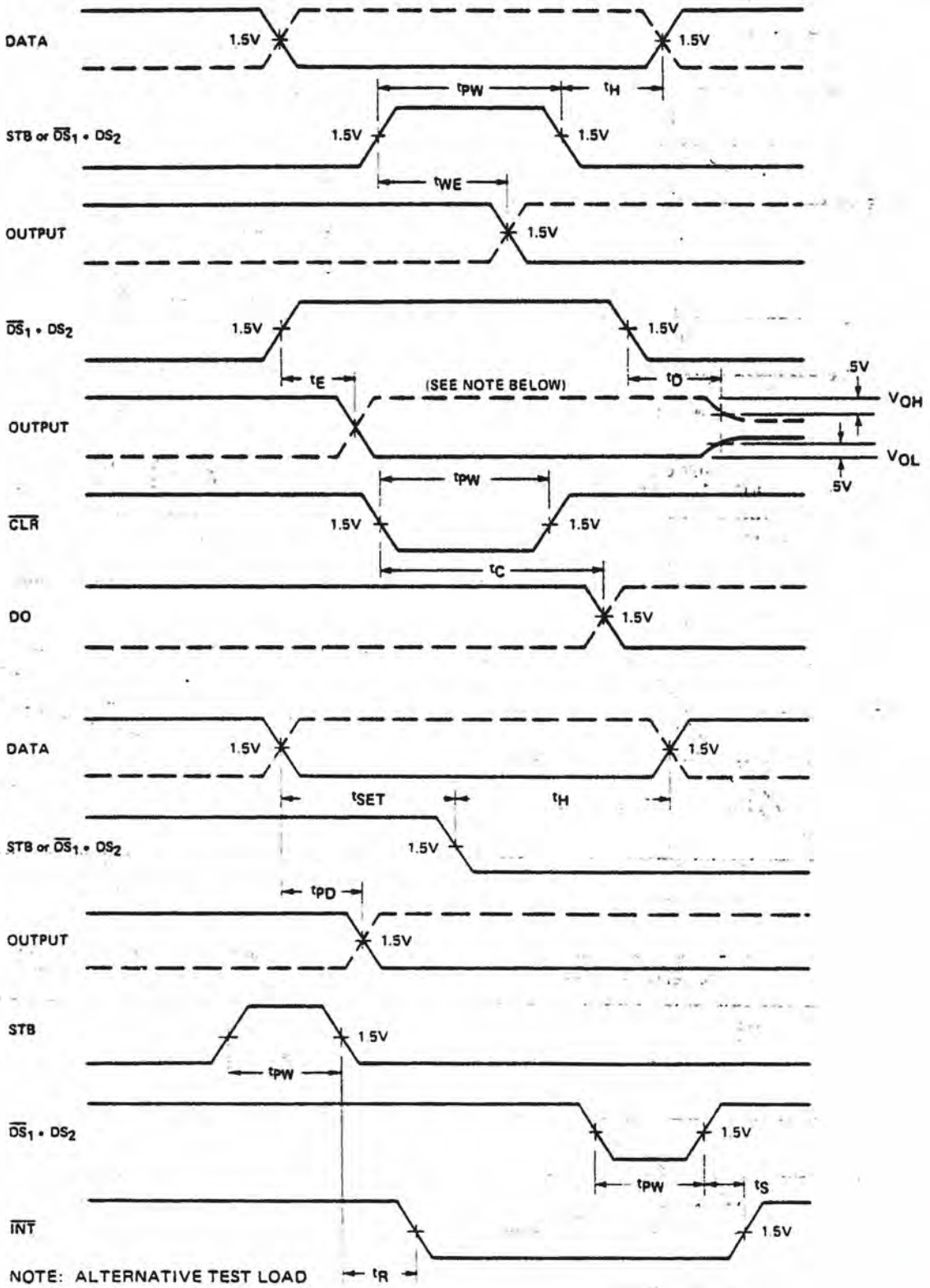
DATA TO OUTPUT DELAY VS. TEMPERATURE



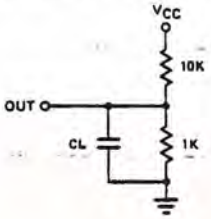
WRITE ENABLE TO OUTPUT DELAY VS. TEMPERATURE



Timing Diagram



NOTE: ALTERNATIVE TEST LOAD



8800b-T

A.C. Characteristics

T_A = 0°C to +75°C V_{CC} = +5V ± 5%

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
t_{pw}	Pulse Width	30			ns	
t_{pd}	Data To Output Delay			30	ns	
t_{we}	Write Enable To Output Delay			40	ns	
t_{set}	Data Setup Time	15			ns	
t_h	Data Hold Time	20			ns	
t_r	Reset To Output Delay			40	ns	
t_s	Set To Output Delay			30	ns	
t_e	Output Enable/Disable Time			45	ns	
t_c	Clear To Output Delay			55	ns	

CAPACITANCE* F = 1 MHz V_{BIAS} = 2.5V V_{CC} = +5V T_A = 25°C

Symbol	Test	LIMITS	
		Typ.	Max.
C_{IN}	DS_1, MD Input Capacitance	9 pF	12 pF
C_{IN}	DS_2, CK, ACK, DI_1-DI_8 Input Capacitance	5 pF	9 pF
C_{OUT}	DO_1-DO_8 Output Capacitance	8 pF	12 pF

*This parameter is sampled and not 100% tested.

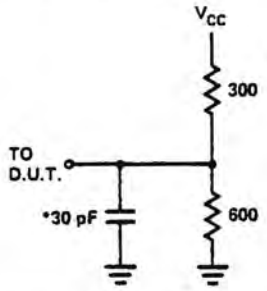
Switching Characteristics

CONDITIONS OF TEST

Input Pulse Amplitude = 2.5 V
Input Rise and Fall Times 5 ns
Between 1V and 2V Measurements made at 1.5V
with 15 mA & 30 pF Test Load

TEST LOAD

15mA & 30pF



* INCLUDING JIG & PROBE CAPACITANCE

Schottky Bipolar 8216/8226

4 BIT PARALLEL BIDIRECTIONAL BUS DRIVER

- **Data Bus Buffer Driver for 8080 CPU**
- **Low Input Load Current — .25 mA Maximum**
- **High Output Drive Capability for Driving System Data Bus**
- **3.65V Output High Voltage for Direct Interface to 8080 CPU**
- **Three State Outputs**
- **Reduces System Package Count**

The 8216/8226 is a 4-bit bi-directional bus driver/receiver.

All inputs are low power TTL compatible. For driving MOS, the DO outputs provide a high 3.65V V_{OH}, and for high capacitance terminated bus structures, the DB outputs provide a high 50mA I_{OL} capability.

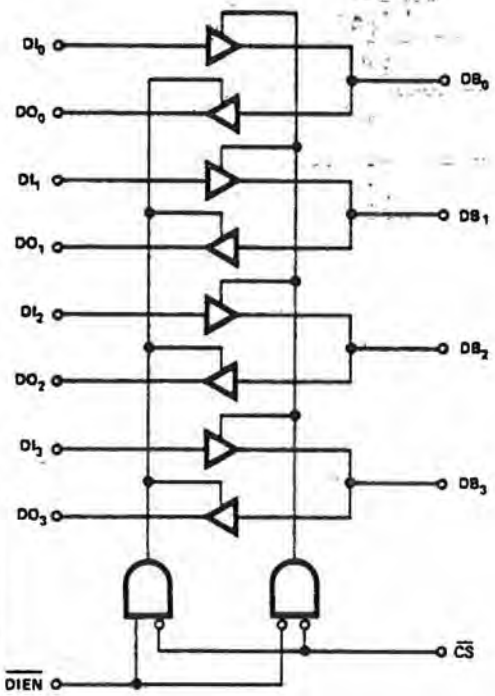
A non-inverting (8216) and an inverting (8226) are available to meet a wide variety of applications for buffering in microcomputer systems.

PIN CONFIGURATION

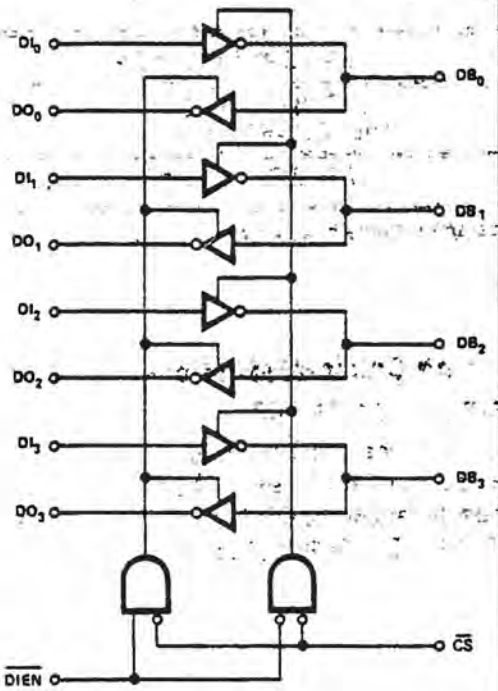
PIN NAMES

DB_0-DB_3	DATA BUS BI-DIRECTIONAL
DI_0-DI_3	DATA INPUT
DO_0-DO_3	DATA OUTPUT
$\overline{DIEN}$	DATA IN ENABLE DIRECTION CONTROL
$\overline{CS}$	CHIP SELECT

LOGIC DIAGRAM 8216



LOGIC DIAGRAM 8226



FUNCTIONAL DESCRIPTION

Microprocessors like the 8080 are MOS devices and are generally capable of driving a single TTL load. The same is true for MOS memory devices. While this type of drive is sufficient in small systems with few components, quite often it is necessary to buffer the microprocessor and memories when adding components or expanding to a multi-board system.

The 8216/8226 is a four bit bi-directional bus driver specifically designed to buffer microcomputer system components.

Bi-Directional Driver

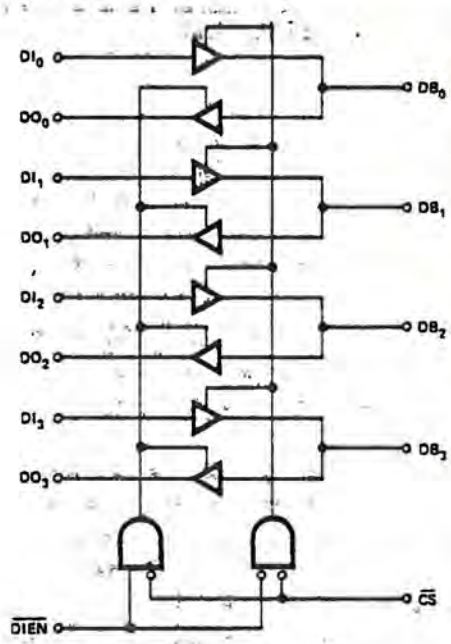
Each buffered line of the four bit driver consists of two separate buffers that are tri-state in nature to achieve direct bus interface and bi-directional capability. On one side of the driver the output of one buffer and the input of another are tied together (DB), this side is used to interface to the system side components such as memories, I/O, etc., because its interface is direct TTL compatible and it has high drive (50mA). On the other side of the driver the inputs and outputs are separated to provide maximum flexibility. Of course, they can be tied together so that the driver can be used to buffer a true bi-directional bus such as the 8080 Data Bus. The DO outputs on this side of the driver have a special high voltage output drive capability (3.65V) so that direct interface to the 8080 and 8008 CPUs is achieved with an adequate amount of noise immunity (350mV worst case).

Control Gating $\overline{DIEN}$, $\overline{CS}$

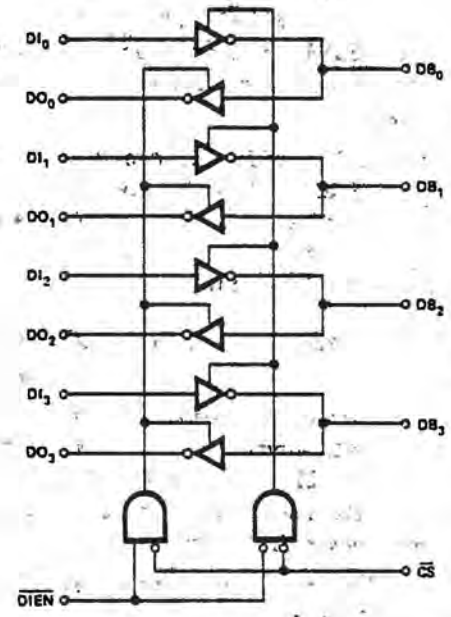
The $\overline{CS}$ input is actually a device select. When it is "high" the output drivers are all forced to their high-impedance state. When it is at "zero" the device is selected (enabled) and the direction of the data flow is determined by the $\overline{DIEN}$ input.

The $\overline{DIEN}$ input controls the direction of data flow (see Figure 1) for complete truth table. This direction control is accomplished by forcing one of the pair of buffers into its high impedance state and allowing the other to transmit its data. A simple two gate circuit is used for this function.

The 8216/8226 is a device that will reduce component count in microcomputer systems and at the same time enhance noise immunity to assure reliable, high performance operation.



(a) 8216



(b) 8226

$\overline{DIEN}$	$\overline{CS}$	
0	0	DI → DB
1	0	DB → DO
0	1	HIGH IMPEDANCE
1	1	

Figure 1. 8216/8226 Logic Diagrams

APPLICATIONS OF 8216/8226

8080 Data Bus Buffer

The 8080 CPU Data Bus is capable of driving a single TTL load and is more than adequate for small, single board systems. When expanding such a system to more than one board to increase I/O or Memory size, it is necessary to provide a buffer. The 8216/8226 is a device that is exactly fitted to this application.

Shown in Figure 2 are a pair of 8216/8226 connected directly to the 8080 Data Bus and associated control signals. The buffer is bi-directional in nature and serves to isolate the CPU data bus.

On the system side, the DB lines interface with standard semiconductor I/O and Memory components and are completely TTL compatible. The DB lines also provide a high drive capability (50mA) so that an extremely large system can be dirven along with possible bus termination networks.

On the 8080 side the DI and DO lines are tied together and are directly connected to the 8080 Data Bus for bi-directional operation. The DO outputs of the 8216/8226 have a high voltage output capability of 3.85 volts which allows direct connection to the 8080 whose minimum input voltage is 3.3 volts. It also gives a very adequate noise margin of 350mV (worst case).

The $\overline{DIEN}$ inputs to 8216/8226 is connected directly to the 8080. $\overline{DIEN}$ is tied to DBIN so that proper bus flow is maintained, and $\overline{CS}$ is tied to $\overline{BUSEN}$ so that the system side Data Bus will be 3-stated when a Hold request has been acknowledged during a DMA activity.

Memory and I/O Interface to a Bi-directional Bus

In large microcomputer systems it is often necessary to provide Memory and I/O with their own buffers and at the same time maintain a direct, common interface to a bi-directional Data Bus. The 8216/8226 has separated data in and data out lines on one side and a common bi-directional set on the other to accomodate such a function.

Shown in Figure 3 is an example of how the 8216/8226 is used in this type of application.

The interface to Memory is simple and direct. The memories used are typically Intel® 8102, 8102A, 8101 or 8107B-4 and have separate data inputs and outputs. The DI and DO lines of the 8216/8226 tie to them directly and under control of the $\overline{MEMR}$ signal, which is connected to the $\overline{DIEN}$ input, an interface to the bi-directional Data Bus is maintained.

The interface to I/O is similar to Memory. The I/O devices used are typically Intel® 8255s, and can be used for both input and output ports. The $\overline{I/O\ R}$ signal is connected directly to the $\overline{DIEN}$ input so that proper data flow from the I/O device to the Data Bus is maintained.

The 8216/8226 can be used in a wide variety of other buffering functions in microcomputer systems such as Address Bus Drivers, Drivers to peripheral devices such as printers, and as Drivers for long length cables to other peripherals or systems.

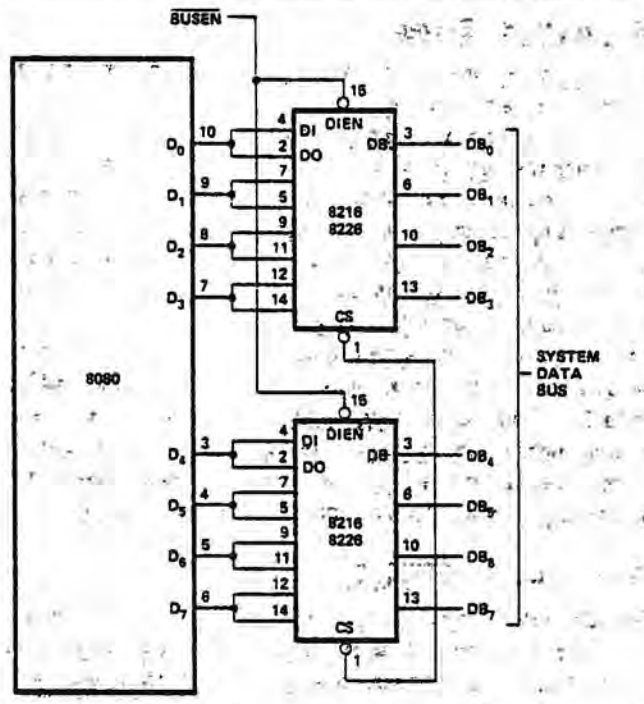


Figure 2. 8080 Data Bus Buffer.

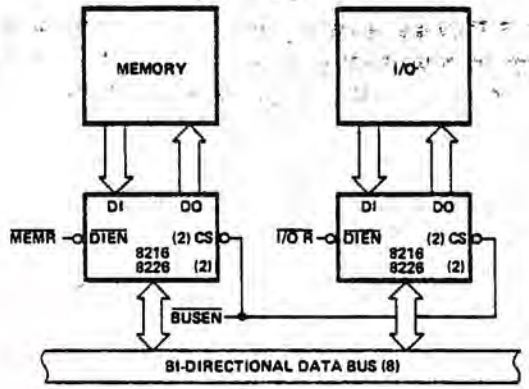


Figure 3. Memory and I/O Interface to a Bi-Directional Bus.