altair 8800b
DOCUMENTATION

mits

2450 Alamo S.E. / Albuquerque, New Mexico 87106

8800b Documentation

Addenda, April, 1977

1. Page 3-67, Figure 3-15, sheet 2 of 3.

   There are several notations at the top of the page reading, "FROM INTERFACE", "TO INTERFACE", etc.

   CHANGE TO:

   "TO DISPLAY/CONTROL BOARD"

2. Page 3-71, Figure 3-16, sheet 1 of 3.

   Resistor R41 (zone A3) is labelled incorrectly.

   CHANGE TO:

   R23

ALTAIR 8800b
TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS - Continued

LIST OF ILLUSTRATIONS - Continued

ALTAIR 8800b
SECTION I
INTRODUCTION

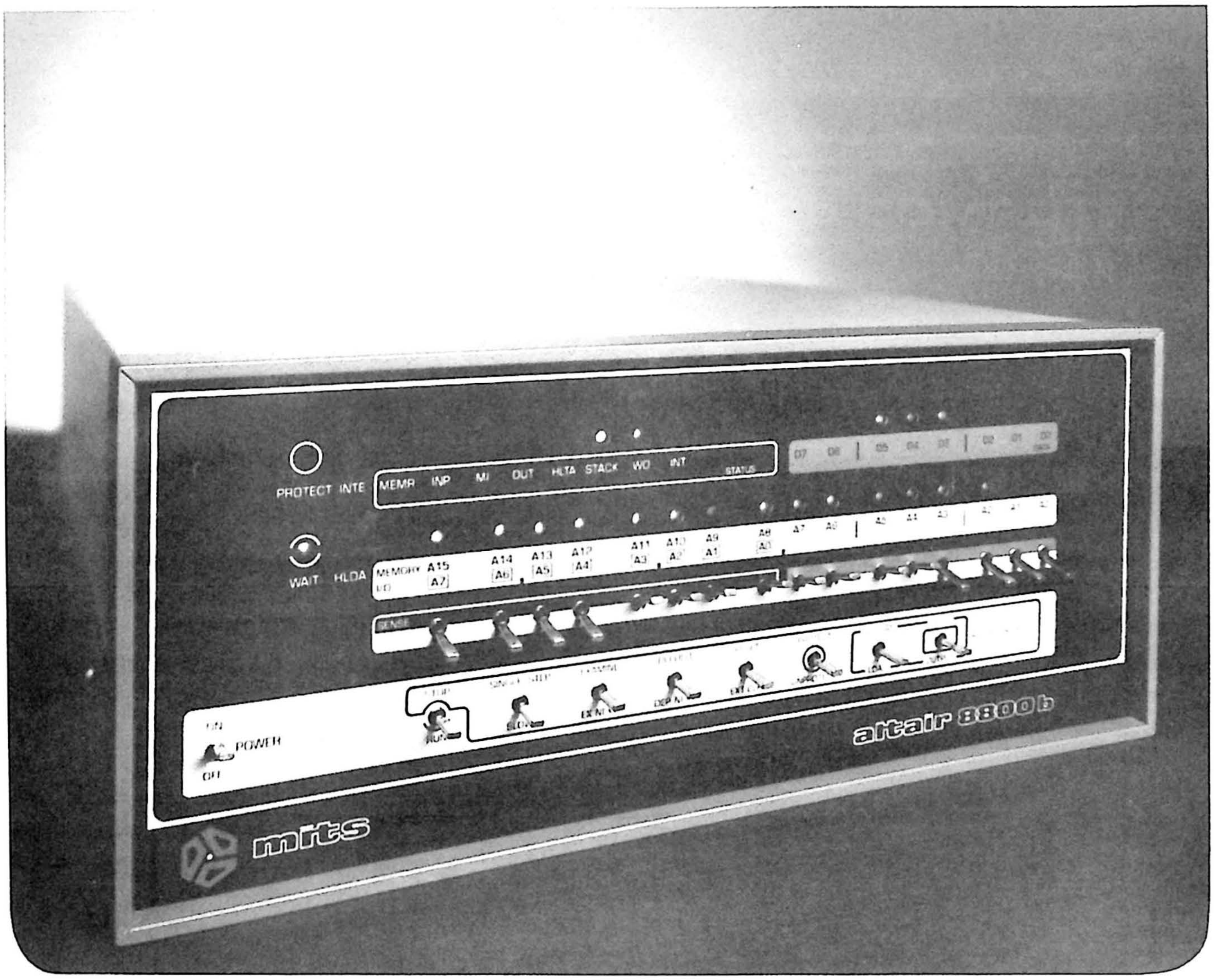


Figure 1-1 **ALTAIR 8800b COMPUTER**

SECTION I

INTRODUCTION

1-1. SCOPE

This ALTAIR™ 8800b Documentation provides a general description of the various printed circuit cards contained in the ALTAIR 8800b and detailed theory of their operation. Included in the documentation is an operator's guide which familiarizes the operator with the various switches and indicators on the ALTAIR 8800b front panel. Detailed assembly instructions are also provided.

1-2. ARRANGEMENT

This manual contains five sections as follows:

1. Section I contains a general description of the ALTAIR 8800b computer and associated printed circuit cards.
2. Section II contains information on the controls and indicators which are located on the ALTAIR 8800b front panel.
3. Section III contains a detailed theory explanation of the ALTAIR 8800b circuit operation.
4. Section IV contains troubleshooting information for the ALTAIR 8800b.
5. Section V contains the detailed assembly instructions for the ALTAIR 8800b.

1-3. DESCRIPTION

The ALTAIR 8800b computer (Figure 1-1) is a general purpose, byte-oriented machine (8-bit word). It uses a common 100-pin bus structure that allows for expansion of either standard or custom plug-in modules. It supports up to 64K of directly addressable memory and can address 256 separate input and output devices. The ALTAIR 8800b computer has 78 basic machine language instructions and consists of a power supply board, an interface board, a central processing unit (CPU) board, and a display/control board.

1-4. <u>POWER SUPPLY BOARD (Figure 1-2)</u>

The Power Supply Board provides two of the three output voltages to the ALTAIR 8800b computer bus, a positive and negative 18 volts. It includes a bridge rectifier circuit and associated filter capacitors, a 10-pin terminal block connector, and the regulating transistors for the positive and negative 18 volt supplies.

1-5. INTERFACE BOARD (Figure 1-3)

The Interface Board buffers all signals between the display/control board and the ALTAIR 8800b bus. It also contains eight parallel data lines which transfer data to the CPU from the Display/Control board.

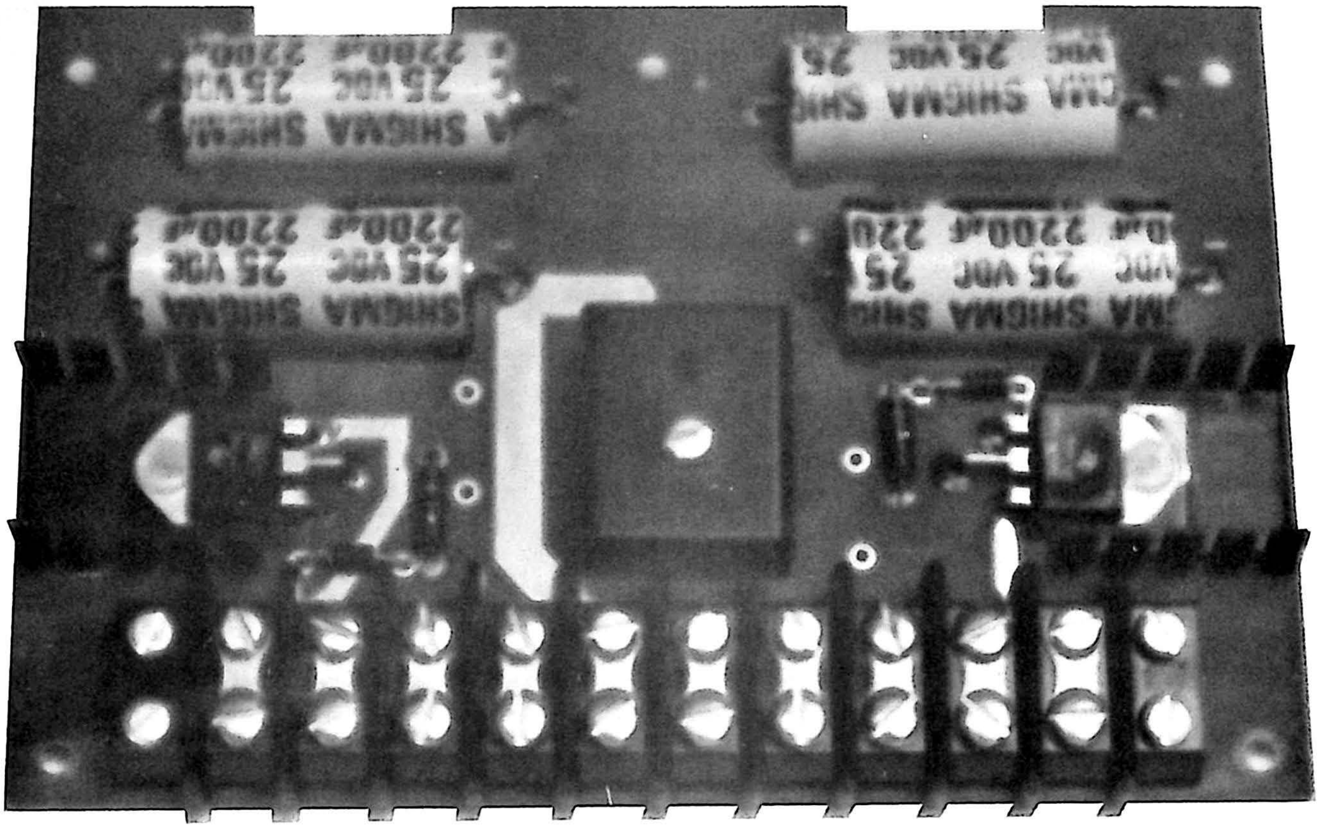


Figure 1-2. Power Supply Board

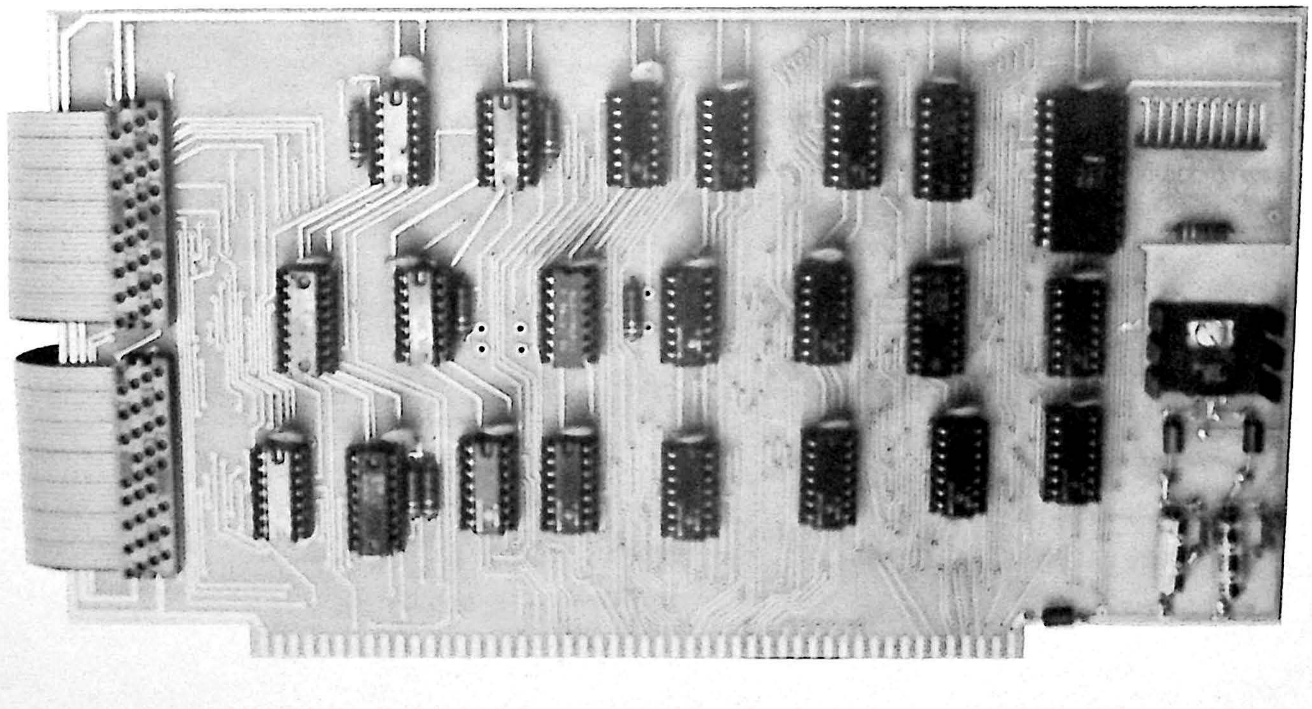


Figure 1-3. Interface Board

1-6. CPU BOARD (Figure 1-4)

The CPU board controls and processes all instructions and data within the ALTAIR 8800b computer. It contains the Intel Corporation model 8080A microprocessor circuit, the master timing circuit, eight input and eight output data lines to the ALTAIR bus control circuits.

1-7. DISPLAY/CONTROL BOARD (Figure 1-5)

The Display/Control Board conditions all ALTAIR 8800b front panel switches and receives information to be displayed on the front panel. It contains a programmable read only memory (PROM), switch and display control circuits, and control circuits to condition the CPU.

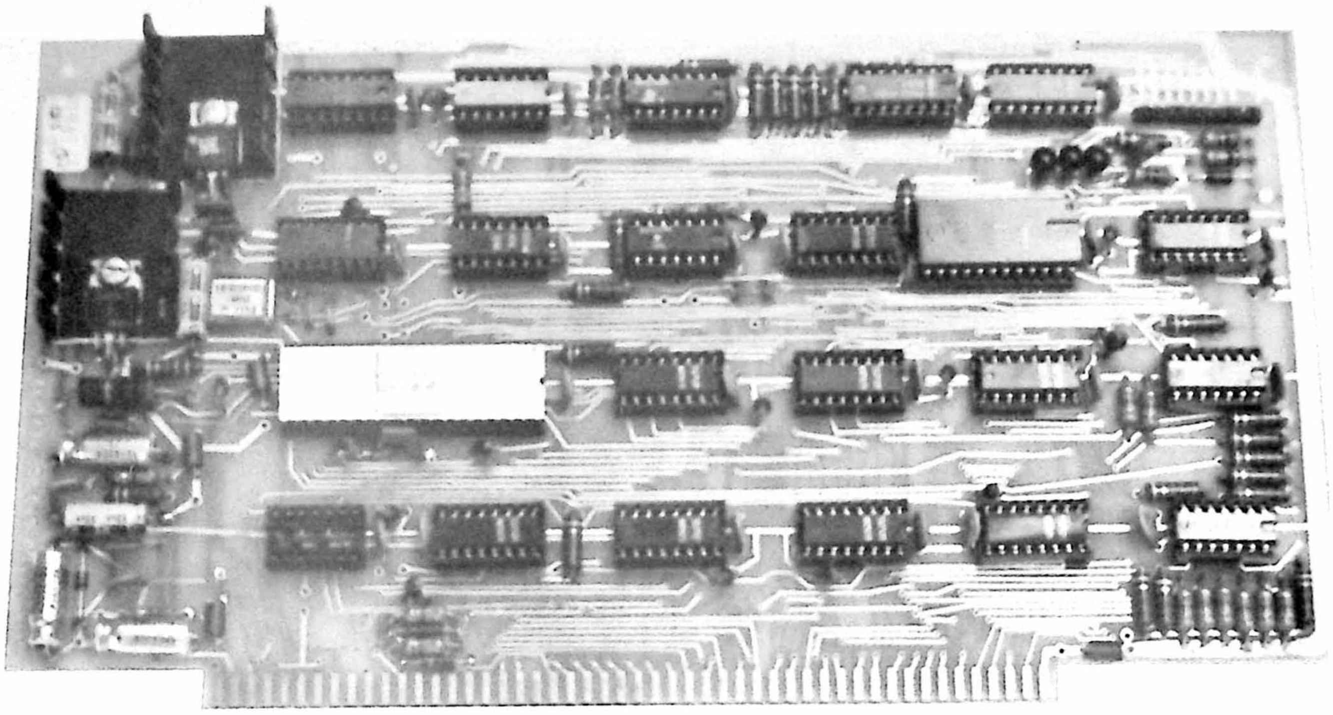


Figure 1-4. CPU Board

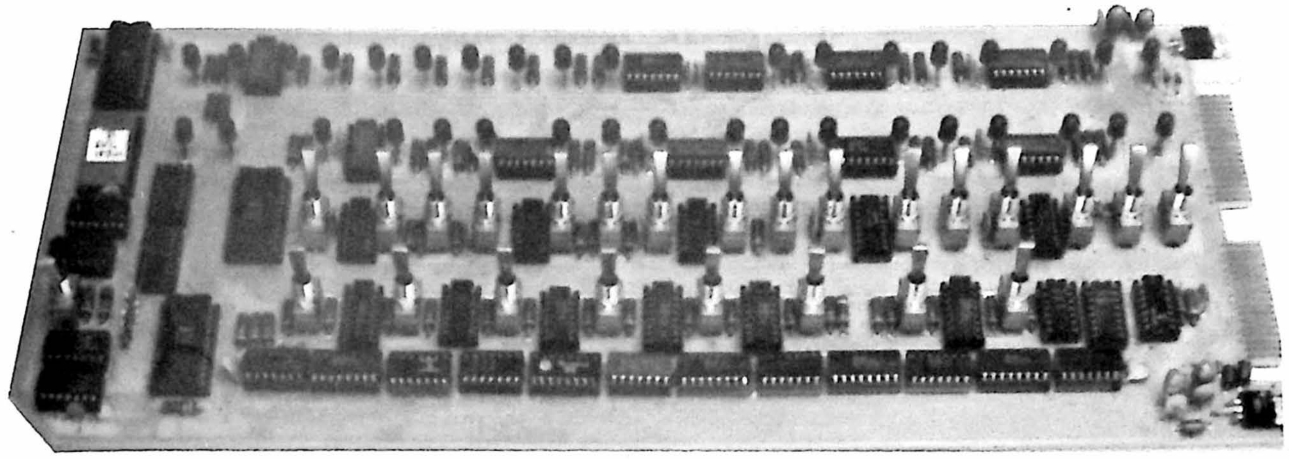


Figure 1-5. Display/Control Board

ALTAIR 8800b SECTION II OPERATOR'S GUIDE

2-1. GENERAL

The Operators Guide contains information on the ALTAIR 8800b computer (8800b) front panel controls and indicators. It includes general switch operation exercises and a sample program which is intended to familiarize the operator with the various front panel operations. Provided in this section are portions of the Intel 8080 Microcomputer Systems Users Manual which contain Central Processor Unit, Interface and Software information. Additional programs available to the user are described in the ALTAIR Software Library. Update information is contained with your unit.

2-2. FRONT PANEL SWITCHES AND INDICATORS

The Front Panel switches permit the operator to perform various ALTAIR 8800b operations, and the indicators display address information, data information, and primary status control line information. Refer to Figure 2-1 for the location of the switches and indicators and Table 2-1 for an explanation of each.

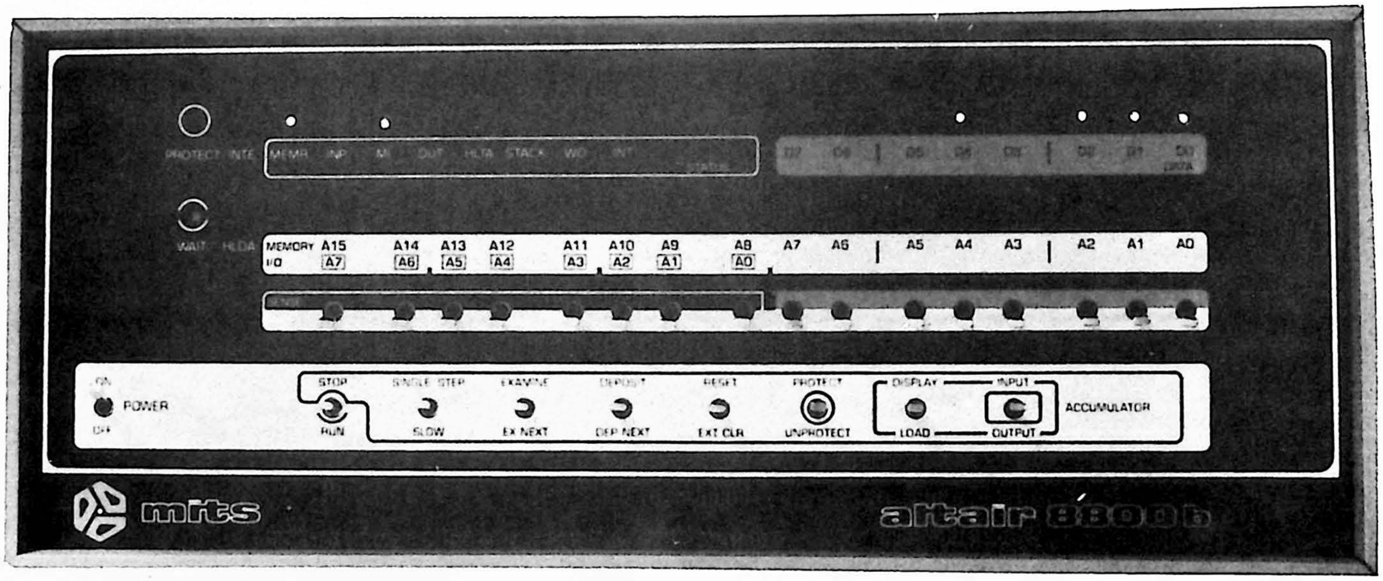


Figure 2-1. Altair 8800b Front Panel

Table 2-1. ALTAIR 8800b Switches and Indicators

Switch	Function or Indication
POWER ON/OFF	Applies power to the ALTAIR 8800b
STOP/RUN	The RUN position allows the CPU to process data and disables all functions on the front panel except reset. The STOP position conditions the CPU to a wait state and enables all functions on the front panel.
SINGLE STEP/ SLOW	The SINGLE STEP position allows execution of one machine cycle or one instruction cycle (depending upon the option selected). SLOW position allows execution of machine or instruction cycles at a rate of approximately 2 cycles per second. (Normal speed is approximately 500,000 machine cycles per second.) The CPU will execute the cycles as long as the SLOW position is maintained.
EXAMINE/ EX NEXT	The EXAMINE position allows the operator to examine the memory address selected on the A0-A15 MEMORY switches. The contents at that address are displayed on the DATA D0-D7 indicators. The EX NEXT position allows the operator to examine the next sequential memory address. Each time EX NEXT is actuated, the contents of the next sequential memory address are displayed.

Table 2-1. ALTAIR 8800b Switches and Indicators - Continued

Switch	Function or Indication
DEPOSIT/ DEP NEXT	The DEPOSIT position stores the contents of the lower address switches (A0-A7) into the memory address that is displayed on the MEMORY address A0-A15 indicators. The DEP NEXT position stores the contents of the lower address switches (A0-A7) into the next successive memory address.
RESET/ EXT CLR	The RESET position resets the program counter to zero and the interrupt enable flag in the CPU. The EXT CLR position produces an external clear signal on the system bus which generally clears an input/output.
PROTECT/ UNPROTECT*	The PROTECT position conditions the write protect circuits on the currently addressed memory board, preventing data in that block of memory from being changed. The front panel or the CPU cannot affect the memory when protected. UNPROTECT position allows the contents of memory to be changed.
ACCUMULATOR DISPLAY/LOAD	The DISPLAY position allows the contents of the CPU accumulator register to be displayed on the DATA D0-D7 indicators. The LOAD position allows the lower eight address switch (A0-A7) information to be stored in the CPU accumulator register.

*Protect switch only applies to memory boards with a protect circuit.

Table 2-1. ALTAIR 8800b Switches and Indicators - Continued

Switch or Indicator	Function or Indication
INPUT/ OUTPUT	The INPUT position allows an external device, selected on the I/O A0-A7 switches (upper eight address switches), to input data into the CPU accumulator. The OUTPUT position allows an external device, selected on the I/O A0-A7 switches, to receive data from the CPU accumulator register.
Address Switches A0-A15	These switches are used to select an address in memory or to enter data. The up position denotes a one bit and the down position denotes a zero bit.
SENSE switches A8-A15	The upper eight address switches (A8-A15) also function as SENSE switches. The data present on these switches is stored in the accumulator if an input from channel 377_8 (front panel) is executed.
MEMORY A0-A15	Display the memory address being examined or loaded with data.
PROTECT	Memory is protected.
INTE	Interrupts are enabled.
MEMR	The CPU is reading data from memory.
INP	An external device is inputting data to the CPU.
M1	The CPU is in machine cycle one of an instruction cycle.
OUT	The CPU is outputting data to an external device.

Table 2-1. ALTAIR 8800b Switches and Indicators - Continued

Indicator	Function or Indication
HLTA	The CPU is in a halt condition.
STACK	The address bus contains the address of the stack pointer.
WO	The CPU is writing out data to an external device or memory.
INT	The CPU has acknowledged an interrupt request.
DATA DO-D7	Data from memory, an external device, or the CPU
WAIT	The CPU is in a wait condition.
HLDA	The CPU has acknowledged a hold signal.

2-3. FRONT PANEL SWITCH APPLICATIONS

The following switch applications are intended to familiarize the operator with the ALTAIR 8800b front panel switches and indicators. Perform the operations in a sequential manner as shown in the following tables.

2-4. POWER ON SEQUENCE (Table 2-2)

The power on sequence resets the CPU program counter to the first memory address and places the CPU in a wait condition at the beginning of an instruction cycle.

Table 2-2. Power On Sequence

Step	Function	Indication
1	Position the POWER ON/OFF switch to ON.	MEMR, M1, and WAIT indicators are on. Some DATA D0-D7 indicators may also be on. All other indicators are off.

2-5. RUN OPERATION (Table 2-3)

The run operation releases the CPU from a wait condition, and allows it to execute a program. When the run operation is enabled, all other front panel switches are inactive except the RESET switch.

Table 2-3. Run Operation

Step	Function	Indication
1	Momentarily position the STOP/RUN switch to RUN.	WAIT indicator is off (or may be dimly lit). The machine can now execute a program.

2-6. STOP OPERATION (Table 2-4)

The stop operation places the CPU in a wait condition and allows the operator to use the switches on the 8800b front panel.

Table 2-4. Stop Operation

Step	Function	Indication
1	Position the STOP/RUN switch to STOP.	WAIT, MEMR, and M1 indicators are on. The operator now has control of the front panel.

2-7. EXAMINE MEMORY OPERATION (Table 2-5)

This procedure allows the operator to select a memory address and examine its contents.

Table 2-5. Examine Memory Operation

Step	Function	Indication
1	Position the address switches A0-A15 down.	
2	Position the EXAMINE/ EX NEXT switch to EXAMINE.	A0 through A15 indicators are off, indicating memory address location 000_8 is being examined. DATA D0 through D7 indicators are displaying the contents of location 000_8.
3	Position address switches A1 and A2 up.	
4	Position the EXAMINE/ EX NEXT switch to EXAMINE.	A1 and A2 indicators are on, indicating memory address 006_8 is being examined. DATA D0 through D7 indicators are displaying the contents of location 006_8.

2-8. ALTERING MEMORY CONTENTS (Table 2-6)

This procedure allows the operator to select a memory address and change its contents.

Table 2-6. Altering Memory Contents

Step	Function	Indication
1	Position address switch A5 up and the remaining switches down.	
2	Position the EXAMINE/ EX NEXT switch to EXAMINE	A5 indicator is on, indicating memory address 040_8. DATA D0 through D7 indicators are displaying the contents of location 040_8.
3	Position the A0 through A7 address switches up.	
4	Position the DEPOSIT/DEP NEXT to DEPOSIT	DATA D0 through D7 indicators are on, indicating the new data that has been placed in address location 040_8.

2-9. EXAMINE NEXT MEMORY LOCATION (Table 2-7)

This procedure allows the operator to examine the next sequential memory location, as determined by the address switches.

Table 2-7. Examine Next Memory Location

Step	Function	Indication
1	Position address switches A0 and A5 up, and the remaining switches down.	
2	Position the EXAMINE/EX NEXT switch to EXAMINE	A0 and A5 indicators are on, indicating memory address 041_8.

Table 2-7. Examine Next Memory Location - Continued

Step	Function	Indication
3	Position address switches A1, A4, and A6 up, and the remaining switches down.	
4	Position the DEPOSIT/ DEP NEXT switch to DEPOSIT	DATA D1, D4, and D6 indicators are on.
5	Position address switch A5 up, and the remaining switches down.	
6	Position the EXAMINE/EX NEXT switch to EXAMINE	A5 indicator is on, indicating memory address 040_8. DATA D0 through D7 indicators are on.
7	Position the EXAMINE/EX NEXT switch to EX NEXT	A5 and A0 indicators are on, indicating address 041_8. DATA D1, D4, and D6 indicators are on.

2-10. ALTER NEXT MEMORY LOCATION CONTENTS (Table 2-8)

This procedure allows the operator to select a memory address and change the contents of the address that immediately follows.

Table 2-8. Altering Next Memory Contents

Step	Function	Indication
1	Position address switches A0 and A5 up, and the remaining switches down.	
2	Position the EXAMINE/EX NEXT switch to EXAMINE	A0 and A5 indicators are on.
3	Position address switches A0 through A7 up	

Table 2-8. Altering Next Memory Contents - Continued

Step	Function	Indication
4	Position the DEPOSIT/ DEP NEXT switch to DEP NEXT	A1 and A5 indicators are on, indicating 042_8. DATA D0 through D7 are on, displaying the new contents of location 042_8.
5	To verify, position address switches A5 and A1 up, and the remaining switches down.	
6	Position the EXAMINE/ EX NEXT switch to EXAMINE	A1 and A5 indicators are on, and DATA D0 through D7 are on.

2-11. LOADING AND DISPLAYING ACCUMULATOR DATA (Table 2-9)

This procedure allows the operator to load new data into the accumulator or check the contents of the accumulator.

Table 2-9. Loading and Displaying Accumulator Data

Step	Function	Indication
1	Position address switches A0, A1, and A2 up, and the remaining switches down.	
2	Position the ACCUMULATOR DISPLAY/LOAD switch to LOAD	
3	Position the ACCUMULATOR DISPLAY/LOAD switch to DISPLAY	DATA D0, D1, and D2 indicators are on while "DISPLAY" is activated.

2-12. LOADING A SAMPLE PROGRAM

The sample program is designed to retrieve two numbers from memory, add them together, and store the result in memory. The exact program in mnemonic form can be written as follows:

0. LDA
1. MOV B,A
2. LDA
3. ADD B
4. STA
5. JMP

The mnemonics for all 78 8800b instructions are explained in detail in the excerpt from the Intel 8080 Microcomputer System User's Manual contained in this section. However, the instructions used in this program are explained as follows:

0. LDA--Load the accumulator with the contents of a specified memory address.
1. MOV B,A--Move the contents of the accumulator into register B.
2. LDA--Same as 0.
3. ADD B--Add the contents of register B to the contents of the accumulator and store the result in the accumulator.
4. STA--Store the contents of the accumulator in a specified memory address.
5. JMP--Jump to the first step in the program.

Step 5, the JMP instruction (followed by the memory address of the first instruction), causes the CPU to "jump" back to the beginning of the sample program and execute the program repeatedly until the CPU is halted. Without a JMP instruction the CPU would continue to run randomly through memory.

2-13. LOADING THE PROGRAM

To load the program into the 8800b, first determine the memory addresses for the two numbers to be added and where the result is to be stored. Store the program instructions in successive memory addresses, beginning at the first memory address, 000_8. In this example the first number to be added will be located at memory address 200_8 (10 000 000), the second at memory address 201_8 (10 000 001), and the sum will be stored in memory address 202_8 (10 000 010). Now that the memory addresses have been specified, the program can be converted into its machine bit patterns (Table 2-10).

Table 2-10. Machine Language Bit Patterns

MNEMONIC	BIT PATTERN	EXPLANATION
LDA 200	00 111 010	Load Accumulator in the CPU with con-
	10 000 000	tents of Memory address 200_8 (2 bytes
	00 000 000	required for memory addresses)
MOV B,A	01 000 111	Move Accumulator data to Register B
LDA 201	00 111 010	Load Accumulator with the contents
	10 000 001	of Memory address 201_8
	00 000 000	
ADD B	10 000 000	Add Register B to Accumulator
STA 202	00 110 010	Store the Accumulator contents
	10 000 010	in Memory address 202_8
	00 000 000	
JMP 000	11 000 011	Jump to Memory location 0.
	00 000 000	
	00 000 000	

The octal equivalent of each bit pattern is also frequently included in the program listing. It is easy to load octal numbers on the front panel switches, since it is only necessary to know the binary equivalents for the numbers 0-7. The resulting program, including octal equivalents, may be written as shown in Table 2-11:

Table 2-11. Addition Program

MEMORY ADDRESS	MNEMONIC	BIT PATTERN	OCTAL EQUIVALENT
000	LDA 200	00 111 010	0 7 2
001	(address)	10 000 000	2 0 0
002	(address)	00 000 000	0 0 0
003	MOV B,A	01 000 111	1 0 7
004	LDA 201	00 111 010	0 7 2
005	(address)	10 000 001	2 0 1
006	(address)	00 000 000	0 0 0
007	ADD B	10 000 000	2 0 0
010	STA 202	00 011 010	0 6 2
011	(address)	10 000 010	2 0 2
012	(address)	00 000 000	0 0 0
013	JMP 000	11 000 011	3 0 3
014	(address)	00 000 000	0 0 0
015	(address)	00 000 000	0 0 0

Using the front panel switches, the program may now be entered into the computer. To begin loading the program at the first memory address 000, position the RESET/CLR switch to RESET. The data to be stored in address 000 is entered on address switches A0 through A7. After the address switches are set, position the DEPOSIT/DEP NEXT switch to DEPOSIT to enter the A0-A7 bit pattern into memory address 000. Enter the second byte of data on the address switches and position the DEPOSIT/DEP NEXT switch to DEP NEXT. The bit pattern will be loaded automatically into the next sequential memory address (001). Continue loading the data into memory for the remainder of the program. The complete program loading procedure is shown in Table 2-12:

Table 2-12. Addition Program Loading

MEMORY ADDRESS	ADDRESS SWITCHES DATA 0-7	CONTROL SWITCH
		RESET
000	00 111 010	DEPOSIT
001	10 000 000	DEPOSIT NEXT
002	00 000 000	DEPOSIT NEXT
003	01 000 111	DEPOSIT NEXT
004	00 111 010	DEPOSIT NEXT
005	10 000 001	DEPOSIT NEXT
006	00 000 000	DEPOSIT NEXT
007	10 000 000	DEPOSIT NEXT
010	00 110 010	DEPOSIT NEXT
011	10 000 010	DEPOSIT NEXT
012	00 000 000	DEPOSIT NEXT
013	11 000 011	DEPOSIT NEXT
014	00 000 000	DEPOSIT NEXT
015	00 000 000	DEPOSIT NEXT

The program is now ready to be run, but first it is necessary to store data at each of the two memory addresses (200_8 and 201_8) to be added together. To load the first address, set address switches A0-A7 to $10\ 000\ 000_2$ and position the EXAMINE/EX NEXT switch to EXAMINE. Now load any desired number into this address by using address switches A0-A7. When the number has been loaded onto the switches, position the DEPOSIT/DEP NEXT to DEPOSIT to load the data into memory. To load the next address, enter a second number on the address switches A0-A7 and position the DEPOSIT/DEP NEXT switch to DEP NEXT. Since sequential memory addresses were selected, the number will be loaded automatically into the proper address ($10\ 000\ 001_2$). Once the program has been loaded and the two numbers have been stored in memory locations 200_8 and 201_8, the program can be run. Return to address 000 by positioning all A0-A7 address switches down and positioning the EXAMINE/EX NEXT switch to EXAMINE. Then position the STOP/RUN switch to RUN. Wait a moment and position the STOP/RUN switch to STOP. Check the answer of your addition program by selecting memory location 202_8 on the address switches and positioning the EXAMINE/EX NEXT switch to EXAMINE. The result is displayed on the DATA D0-D7 indicators.

2-14. INTEL 8080 MICROCOMPUTER SYSTEMS USER'S INFORMATION

Pages 2-16 through 2-65 are excerpts from the Intel 8080 Microcomputer Systems User's Manual, reprinted by permission of Intel Corporation, Copyright 1975. Included is detailed Central Processor Unit, Interface and Software information pertaining to the 8080 Microcomputer System.

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 by 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.

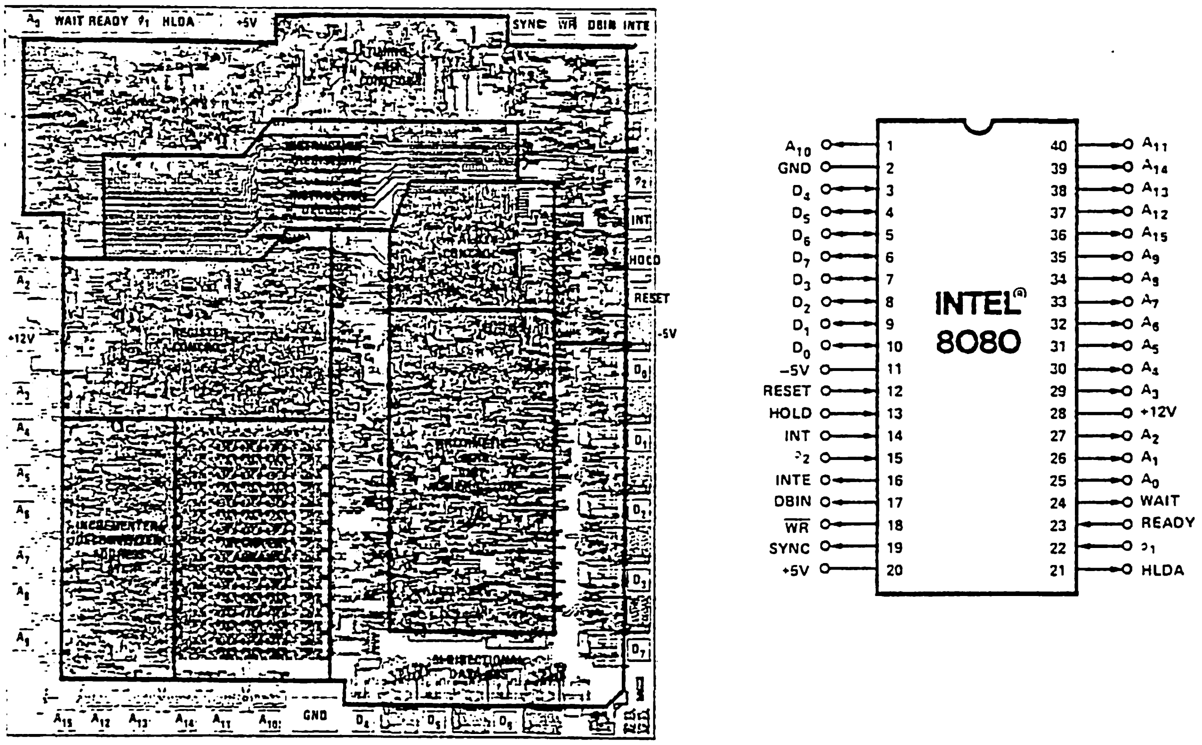


Figure 2-1. 8080 Photomicrograph With Pin Designations

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.

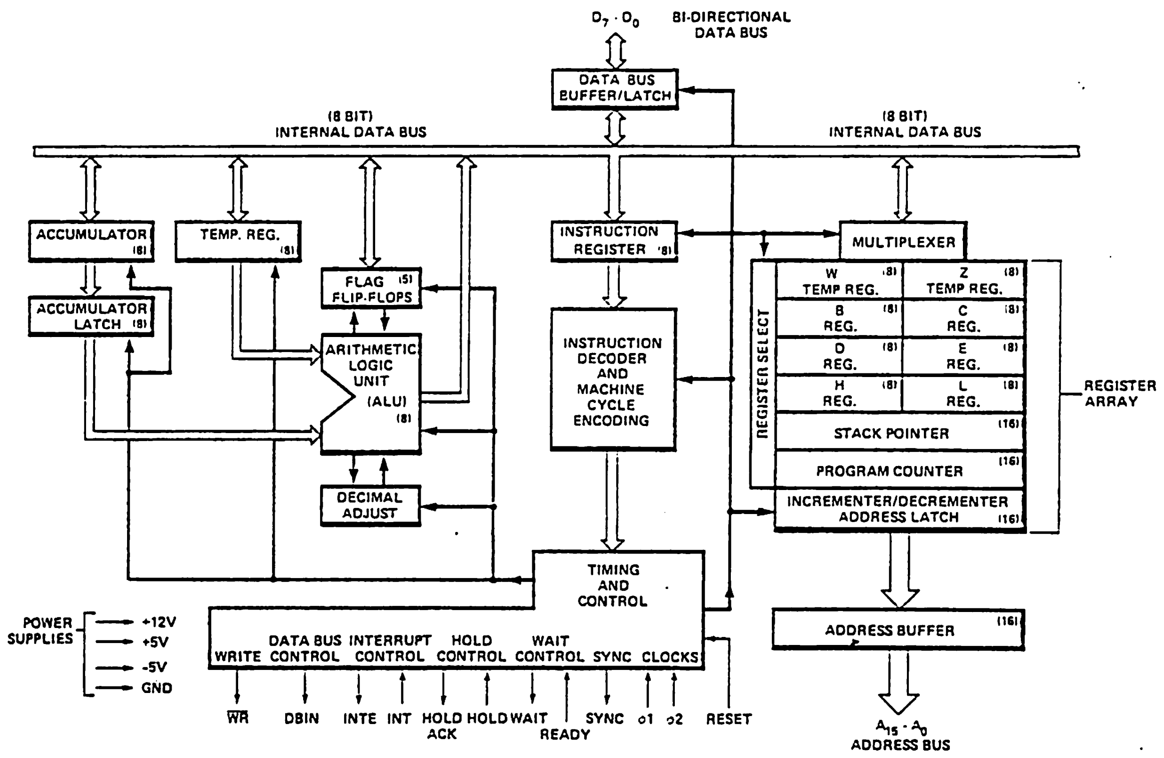


Figure 2-2. 8080 CPU Functional Block Diagram

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.

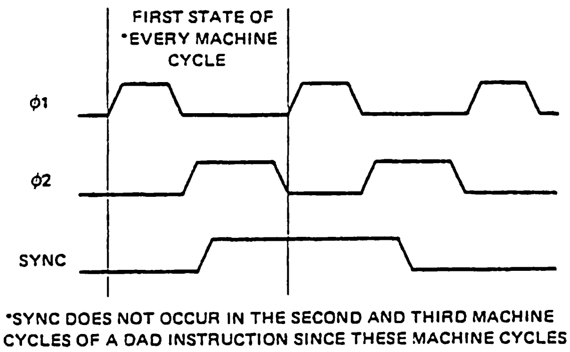
During the input mode, data from the external data bus is transferred to the internal bus. The internal bus is precharged at the beginning of each internal state, except for the transfer state (T_3—described later in this chapter).

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.

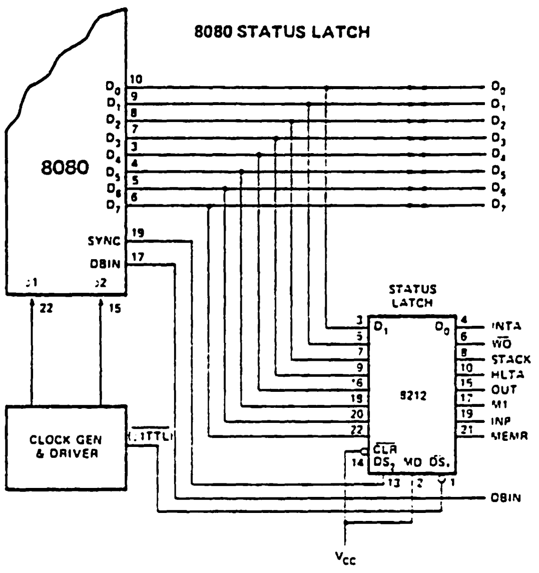
Instructions for the 8080 require from one to five machine cycles for complete execution. The 8080 sends out 8 bit of status information on the data bus at the beginning of each machine cycle (during SYNC time). The following table defines the status information.

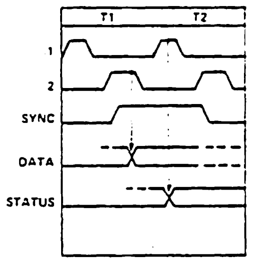
STATUS INFORMATION DEFINITION

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.
$\overline{\text{WO}}$	D_1	Indicates that the operation in the current machine cycle will be a WRITE memory or OUTPUT function ($\overline{\text{WO}}$ = 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 $\overline{\text{WR}}$ 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.

8080 STATUS LATCH





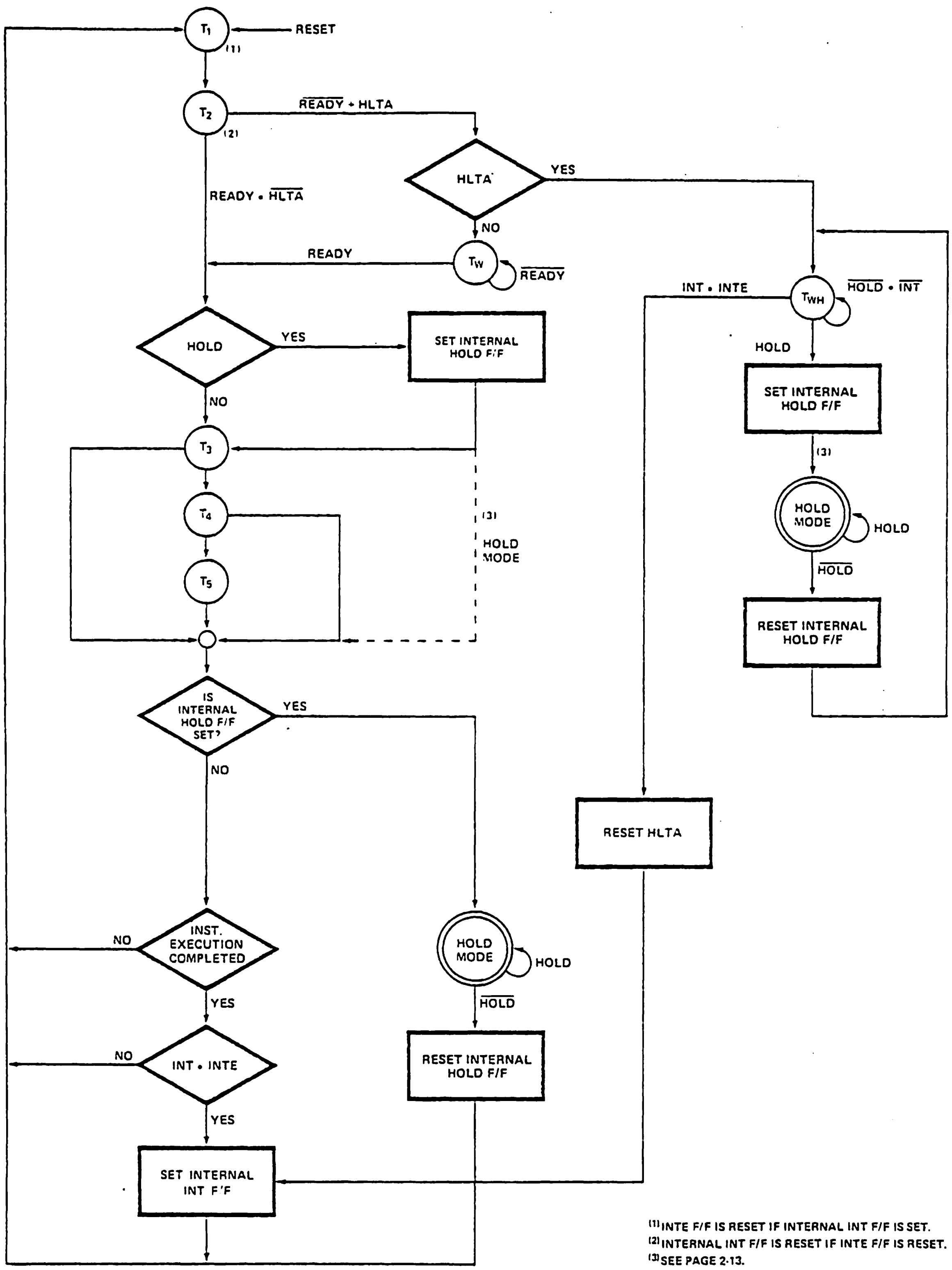
STATUS WORD CHART

TYPE OF MACHINE CYCLE

Ⓝ STATUS WORD

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	$\overline{\text{WO}}$	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

Table 2-1. 8080 Status Bit Definitions



(1) INTE F/F IS RESET IF INTERNAL INT F/F IS SET.
(2) INTERNAL INT F/F IS RESET IF INTE F/F IS RESET.
(3) SEE PAGE 2-13.

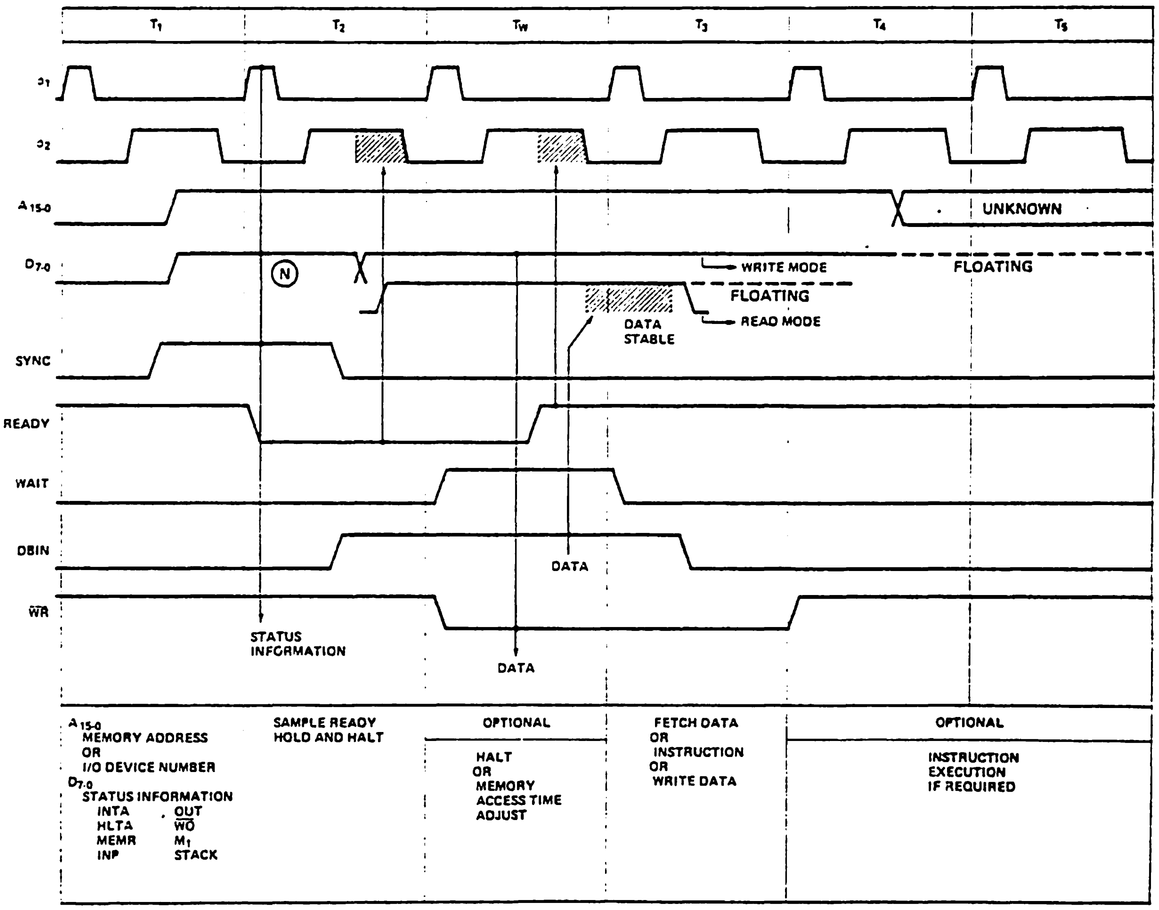
Figure 2-4. CPU State Transition Diagram

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.

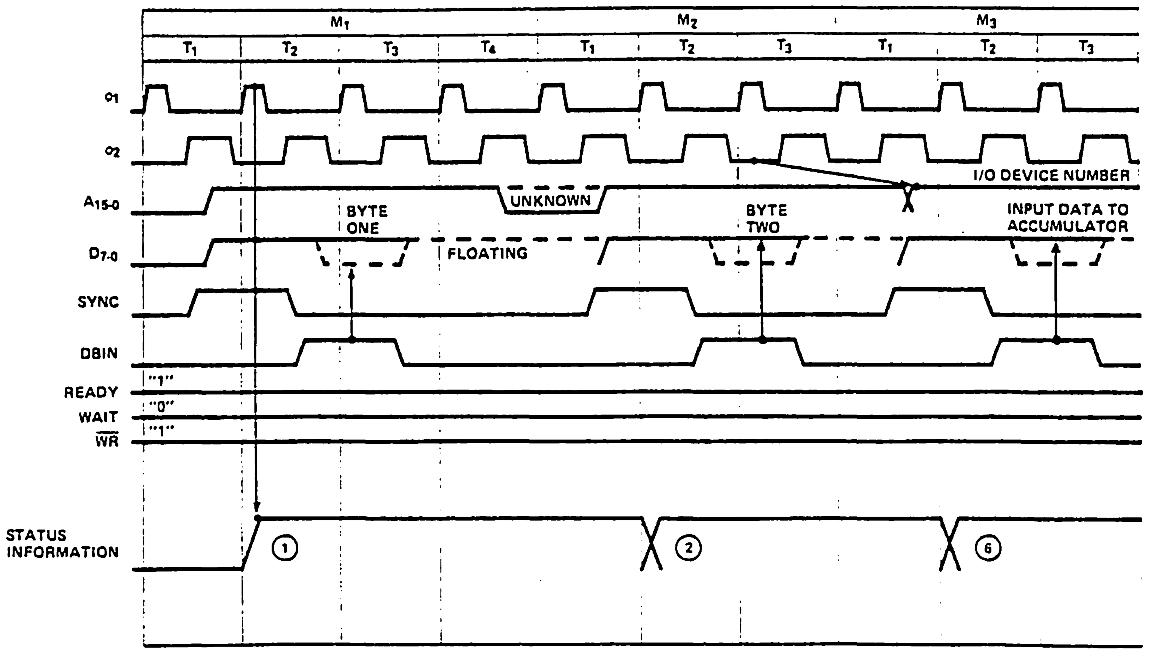
During the input of data to the processor, the 8080 generates a DBIN signal which should be used externally to enable the transfer. Machine cycles in which DBIN is available include: FETCH, MEMORY READ, STACK READ, and INTERRUPT. DBIN is initiated by the rising edge of ϕ_2 during state T2 and terminated 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



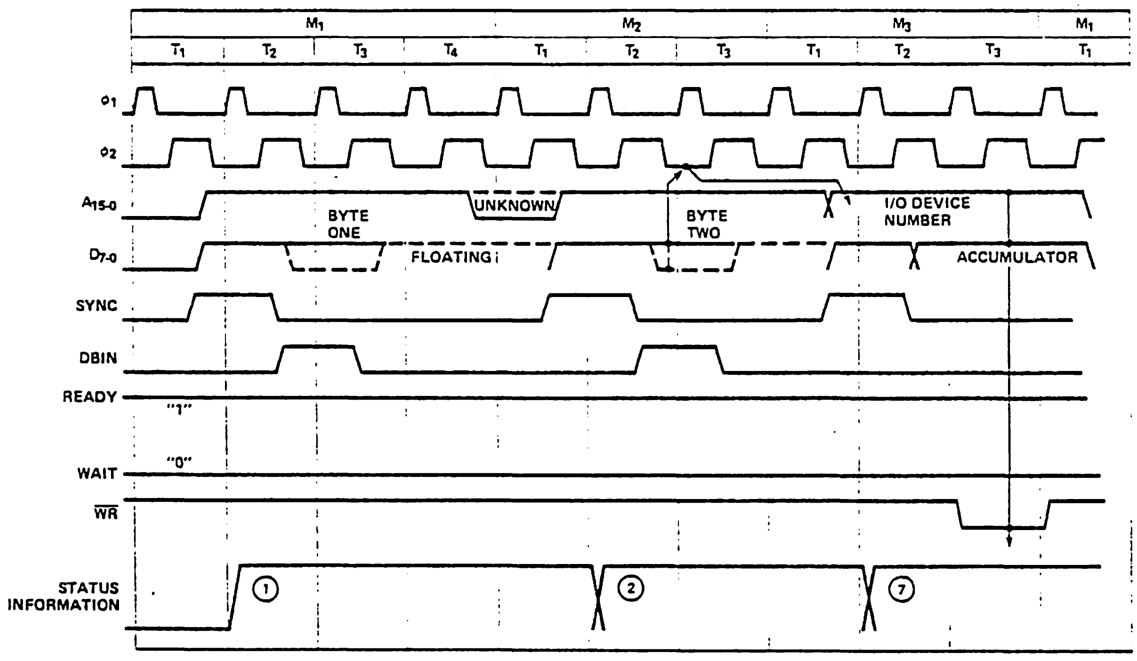
NOTE: Ⓝ Refer to Status Word Chart on Page 2-6.

Figure 2-5. Basic 8080 Instruction Cycle



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

Figure 2-6. Input Instruction Cycle



NOTE: Ⓝ 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.

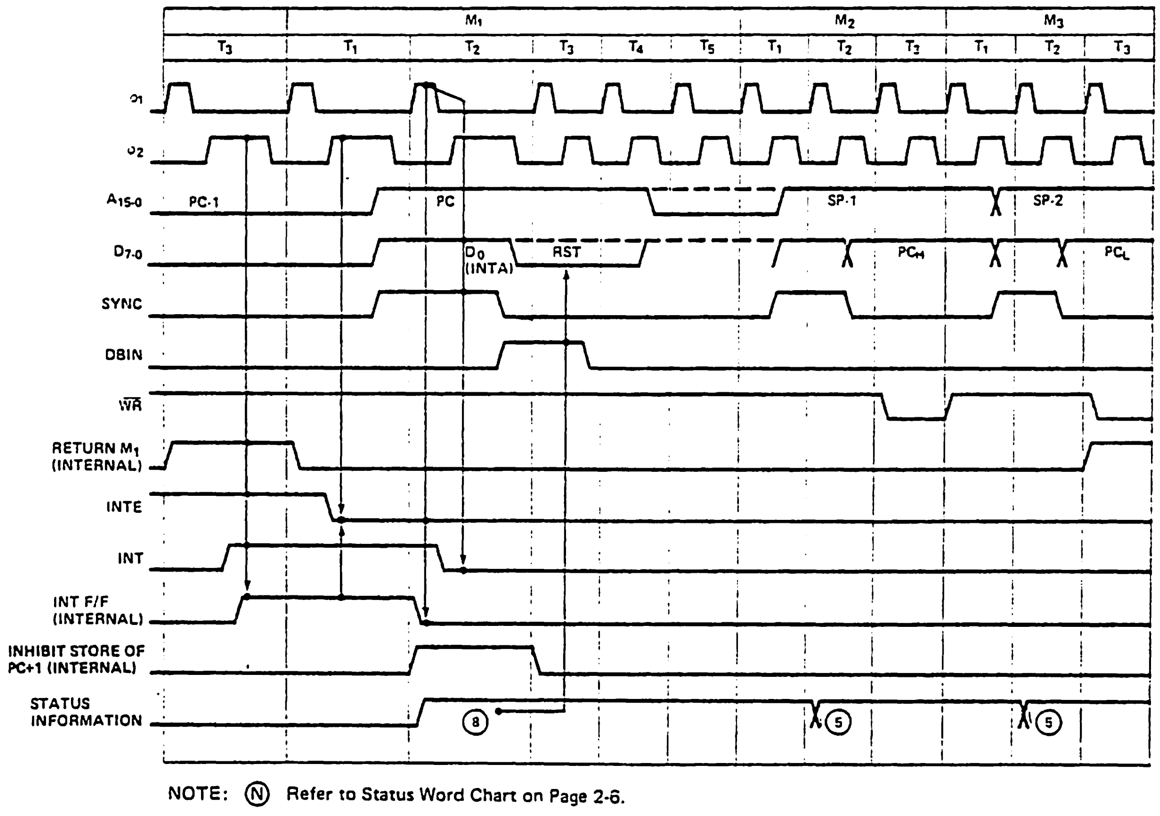


Figure 2-8. Interrupt Timing

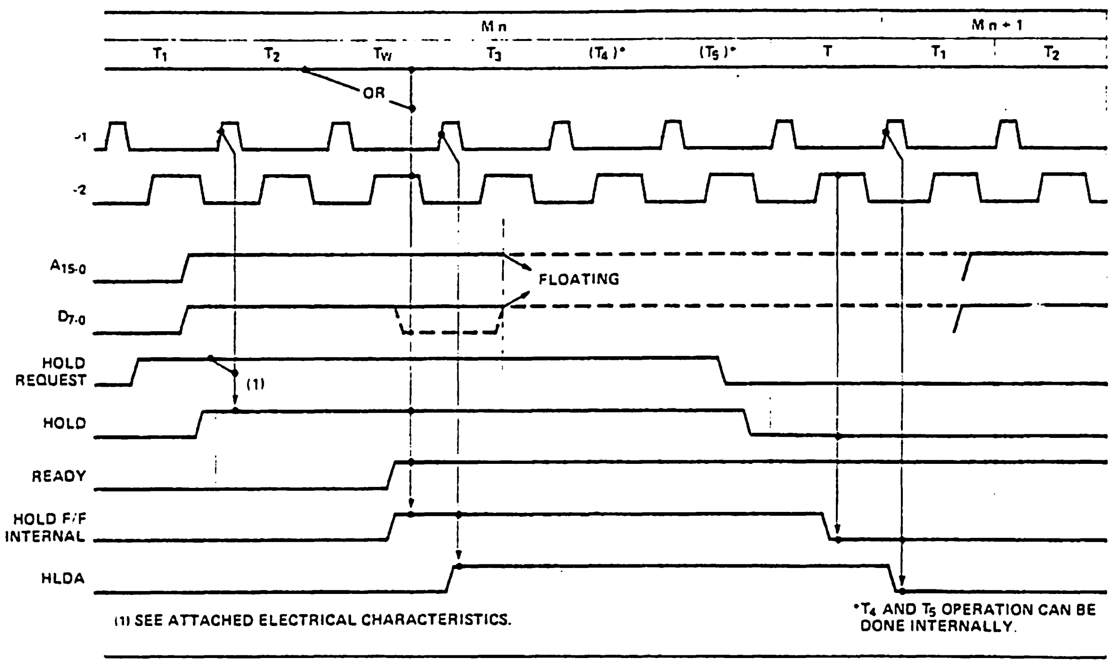


Figure 2-9. HOLD Operation (Read Mode)

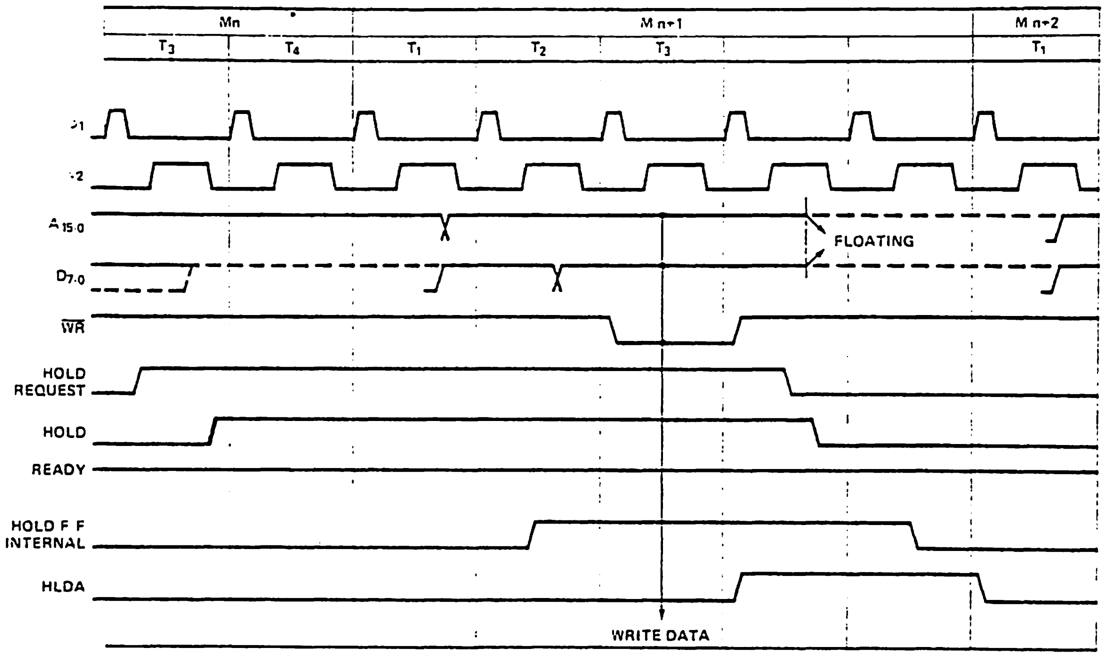


Figure 2-10. HOLD Operation (Write Mode)

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:

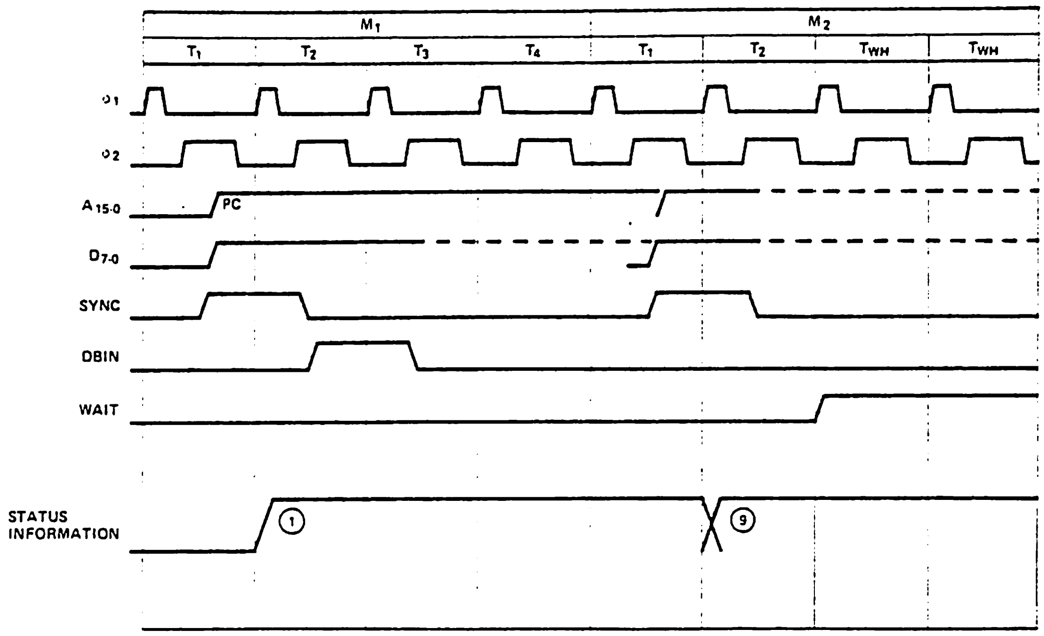
- A high on the RESET line will always reset the 8080 to state T_1; RESET also clears the program counter.
- A HOLD input will cause the 8080 to enter the hold state, as previously described. When the HOLD line goes low, the 8080 re-enters the halt state on the rising edge of the next ϕ_1 clock pulse.
- 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.



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

Figure 2-11. HALT Timing

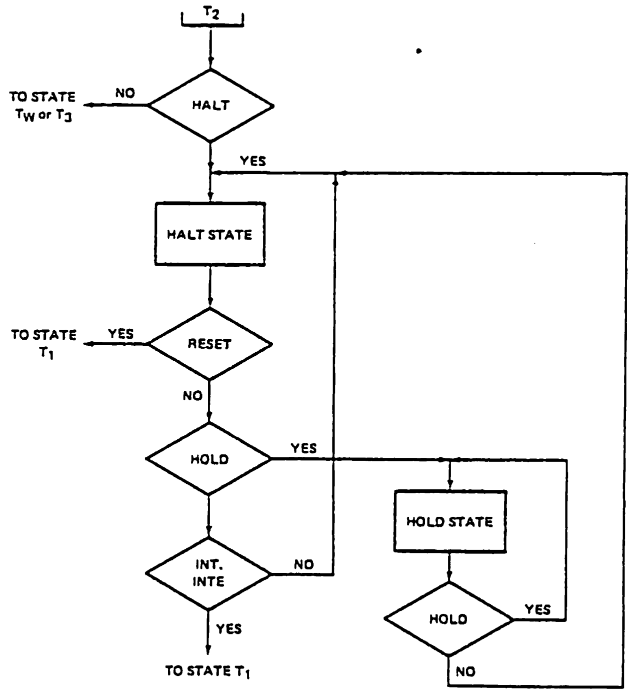


Figure 2-12. HALT Sequence Flow Chart.

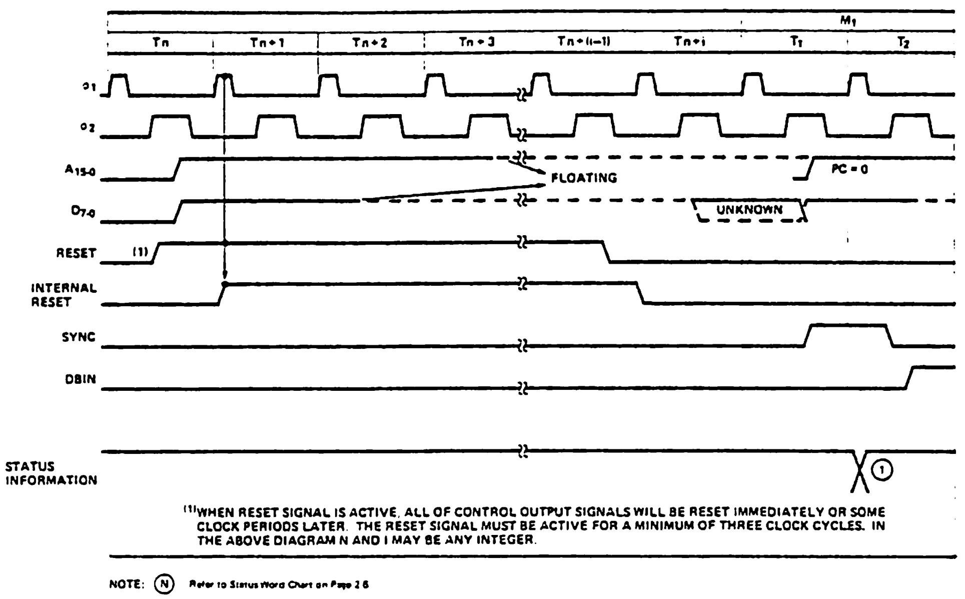

M1
Tn
Tn+1
Tn+2
Tn+3
Tn+(i−1)
Tn+i
T1
T2
Φ1
Φ2
A15-0
FLOATING
PC = 0
D7-0
UNKNOWN
RESET
(1)
INTERNAL RESET
SYNC
DBIN
STATUS INFORMATION
1
(1)WHEN RESET SIGNAL IS ACTIVE, ALL OF CONTROL OUTPUT SIGNALS WILL BE RESET IMMEDIATELY OR SOME CLOCK PERIODS LATER. THE RESET SIGNAL MUST BE ACTIVE FOR A MINIMUM 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.

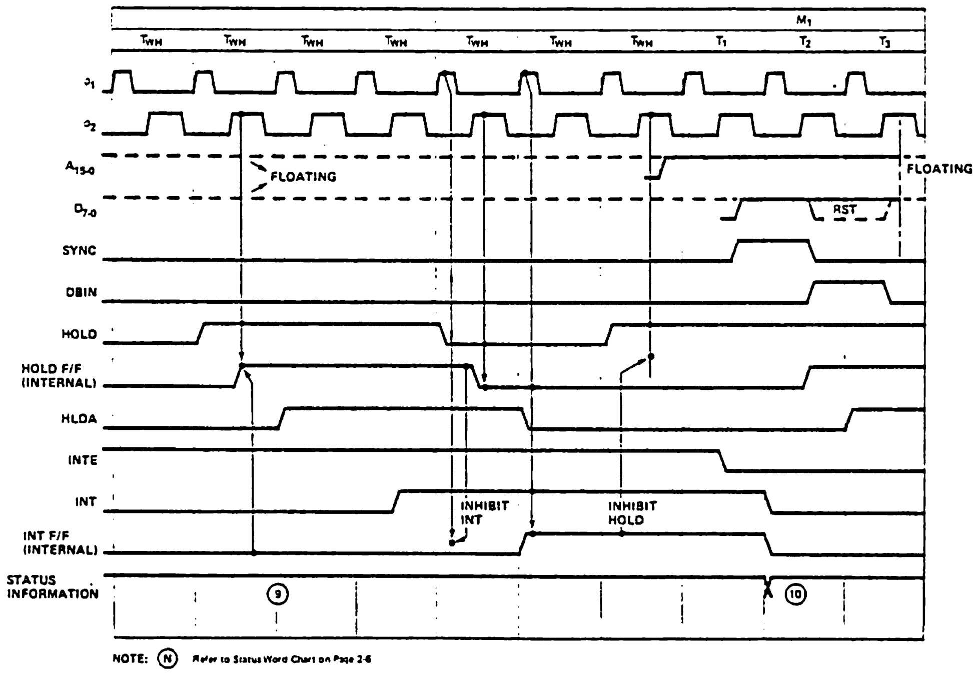


Figure 2-14. Relation between HOLD and INT in the HALT State.

MNEMONIC	OP CODE		M1[1]					M2		
	D7 D6 D5 D4	D3 D2 D1 D0	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	→DDD
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)+1	→TMP →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)-1	→TMP →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		(ri)→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 83 →	rh								
↑	PC = PC + 1 83 →	W	WZ OUT STATUS[6]	DATA →	A					
		PC = PC + 1 B3 →	W	WZ OUT STATUS[7]	(A) →	DATA BUS					
↓	PC = PC + 1 83 →	W	WZ OUT STATUS[6]	DATA → WZ = WZ + 1	L	WZ OUT STATUS[6]	DATA →	H		
PC OUT STATUS[6]	PC = PC + 1 83 →	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		
	$D_7D_6D_5D_4$	$D_3D_2D_1D_0$	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			0→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) → SP = SP - 1	DATA BUS	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) → SP = SP - 1	DATA BUS	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) → (PCL) →	Z DATA BUS									WZ OUT STATUS[11]	(WZ) + 1 → PC
SP OUT STATUS[16]	(rl) →	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 <u>or</u> 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, <u>or</u> 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 ($A_{0\text{-}15}$) 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 ($A_{0\text{-}7}$) and 8-15 ($A_{8\text{-}15}$).

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

*"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.

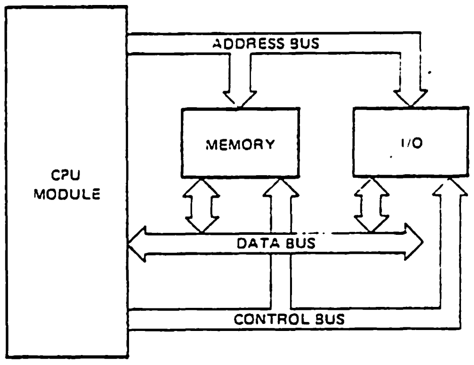


Figure 3-1. Typical Computer System Block Diagram

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.

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,

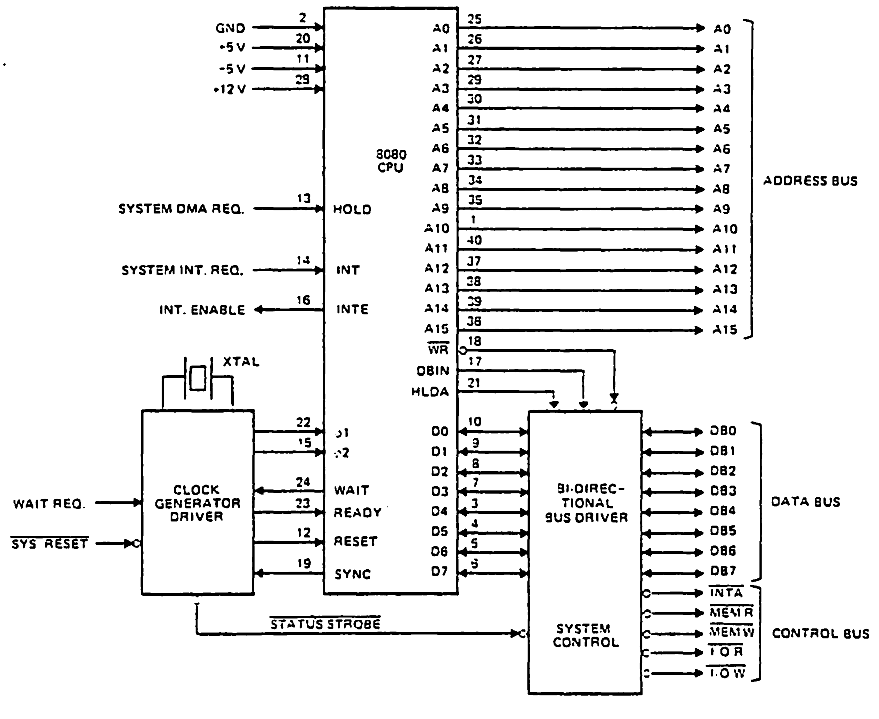


Figure 3-2. 8080 CPU Interface

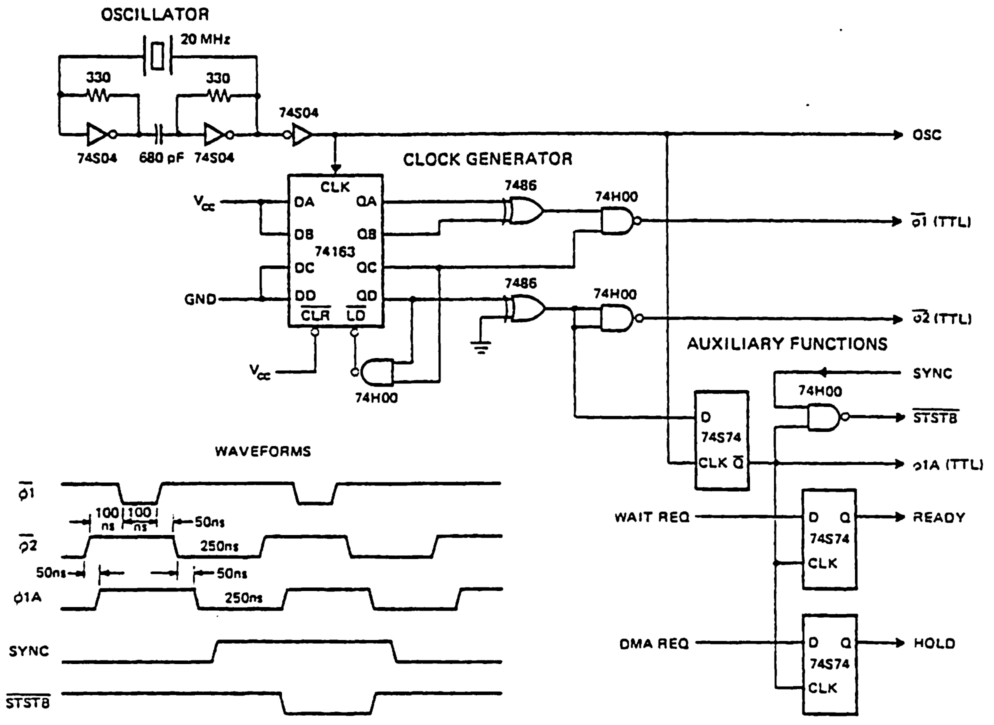


Figure 3-3. 8080 Clock Generator

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.

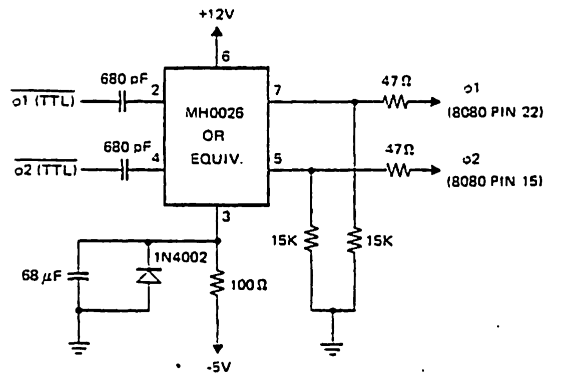


Figure 3-4. High Level Driver

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 ($\overline{\text{STSTB}}$) 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.

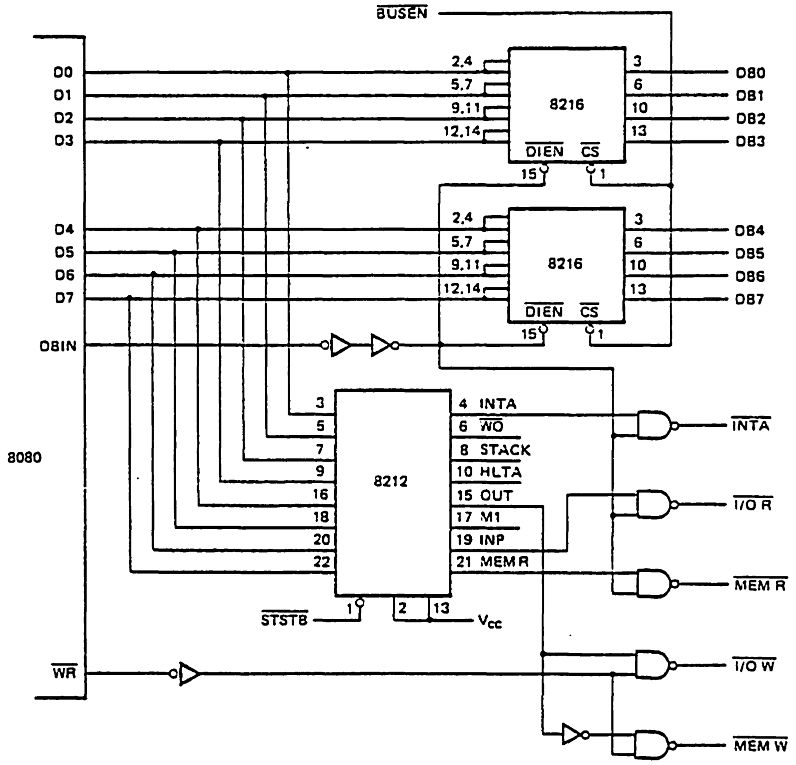


Figure 3-5. 8080 System Control

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 ($\overline{\text{DIEN}}$) so the correct flow of data on the bus is maintained. The chip select ($\overline{\text{CS}}$) of the 8216 is connected to BUS ENABLE ($\overline{\text{BUSEN}}$) 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 ($\overline{\text{STSTB}}$) 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 $\overline{\text{WR}}$ signals from the 8080 generate the (4) four Control signals that make up the basic Control Bus.

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

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

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

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

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

A fifth signal, Interrupt Acknowledge ($\overline{\text{INTA}}$) 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 $\overline{\text{WO}}$, 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 ($\overline{\text{BUSEN}}$) 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.

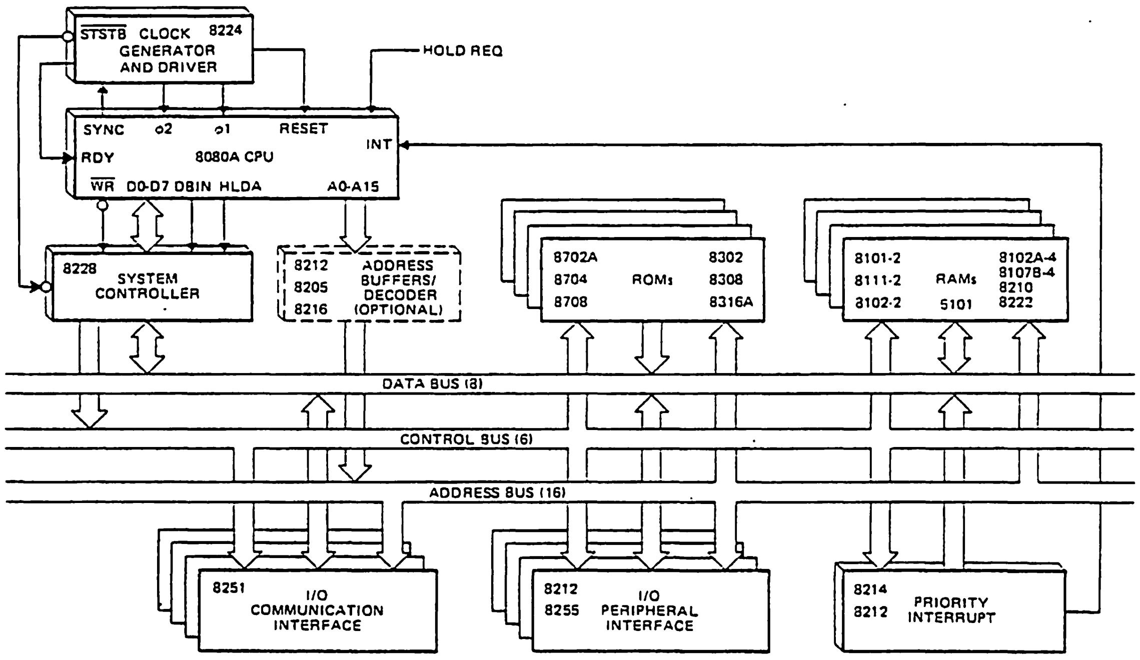
ROM INTERFACE

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 $\overline{\text{MEMR}}$ 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 ($\overline{\text{MEMR}}$) 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



STSTB CLOCK 8224 GENERATOR AND DRIVER
HOLD REQ
SYNC ø2 ø1 RESET
INT
RDY 8080A CPU
WR D0-D7 DBIN HLDA A0-A15
8228 SYSTEM CONTROLLER
8212 8205 8216 ADDRESS BUFFERS/ DECODER (OPTIONAL)
8702A 8704 8708 ROMs 8302 8308 8316A
8101-2 8111-2 8102-2 RAMs 5101 8102A-4 8107B-4 8210 8222
DATA BUS (8)
CONTROL BUS (6)
ADDRESS BUS (16)
8251 I/O COMMUNICATION INTERFACE
8212 8255 I/O PERIPHERAL INTERFACE
8214 8212 PRIORITY INTERRUPT

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 $\overline{\text{MEMW}}$ command is generated. The $\overline{\text{MEMW}}$ 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.

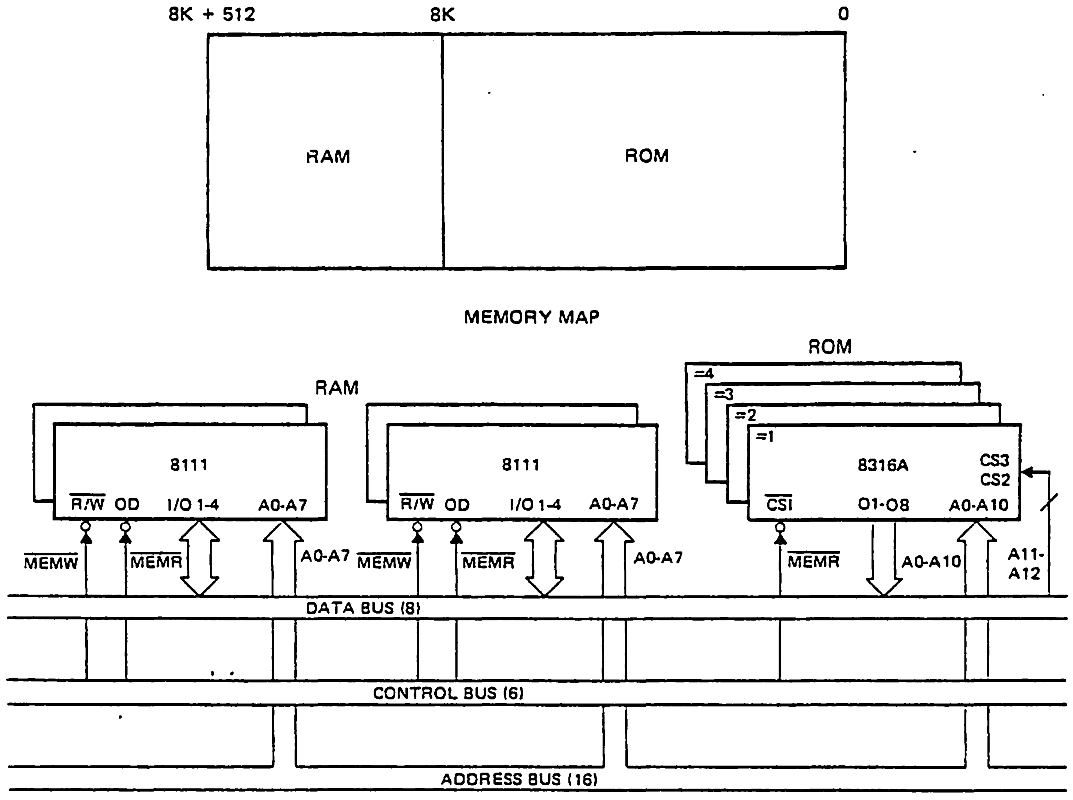


Figure 3-7. Typical Memory Interface

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.

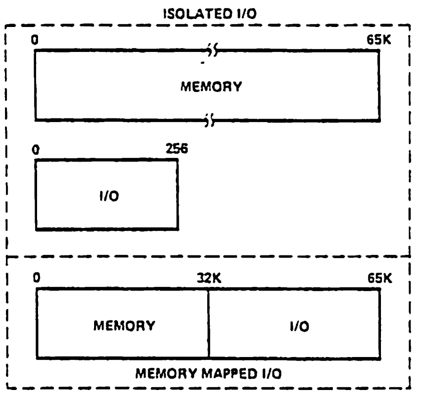


Figure 3-8. Memory/I/O Mapping.

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.

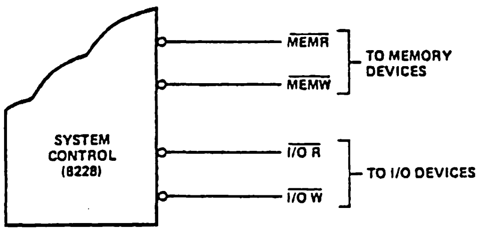


Figure 3-9. Isolated I/O.

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 $\overline{\text{MEMR}}$ and $\overline{\text{MEMW}}$ 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.

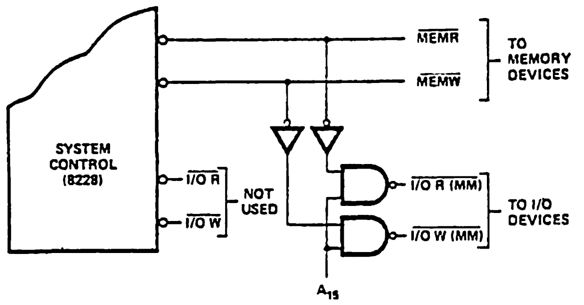


Figure 3-10. Memory Mapped I/O.

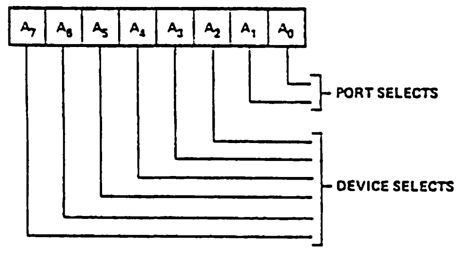
I/O Addressing

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



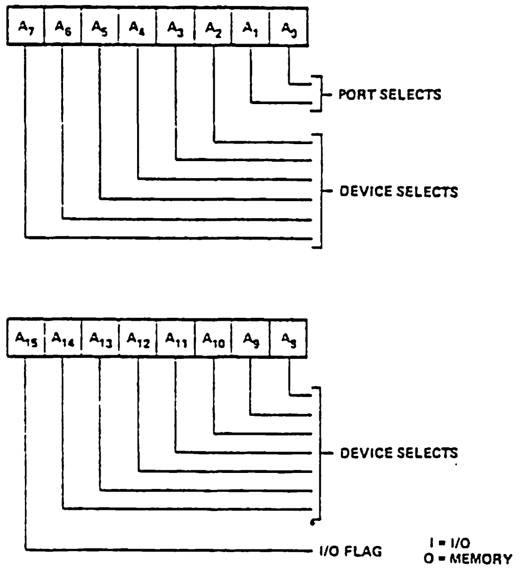
ADDRESSES – 6 – 8255s
(18 PORTS – 144 BITS)

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



ADDRESSES – 13 – 8255s
(39 PORTS – 312 BITS)

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.

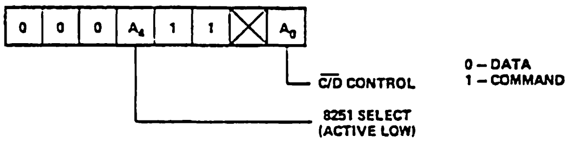


Figure 3-13. 8251 Format.

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.

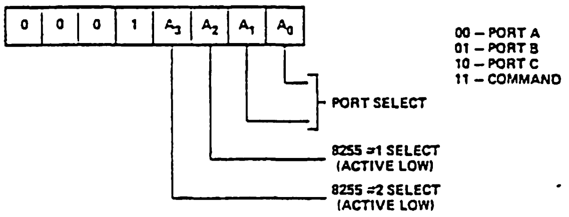
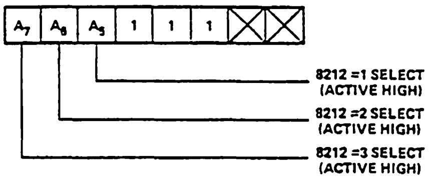


Figure 3-14. 8255 Format.

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



A7
A6
A5
1
1
1
8212 #1 SELECT (ACTIVE HIGH)
8212 #2 SELECT (ACTIVE HIGH)
8212 #3 SELECT (ACTIVE HIGH)

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.

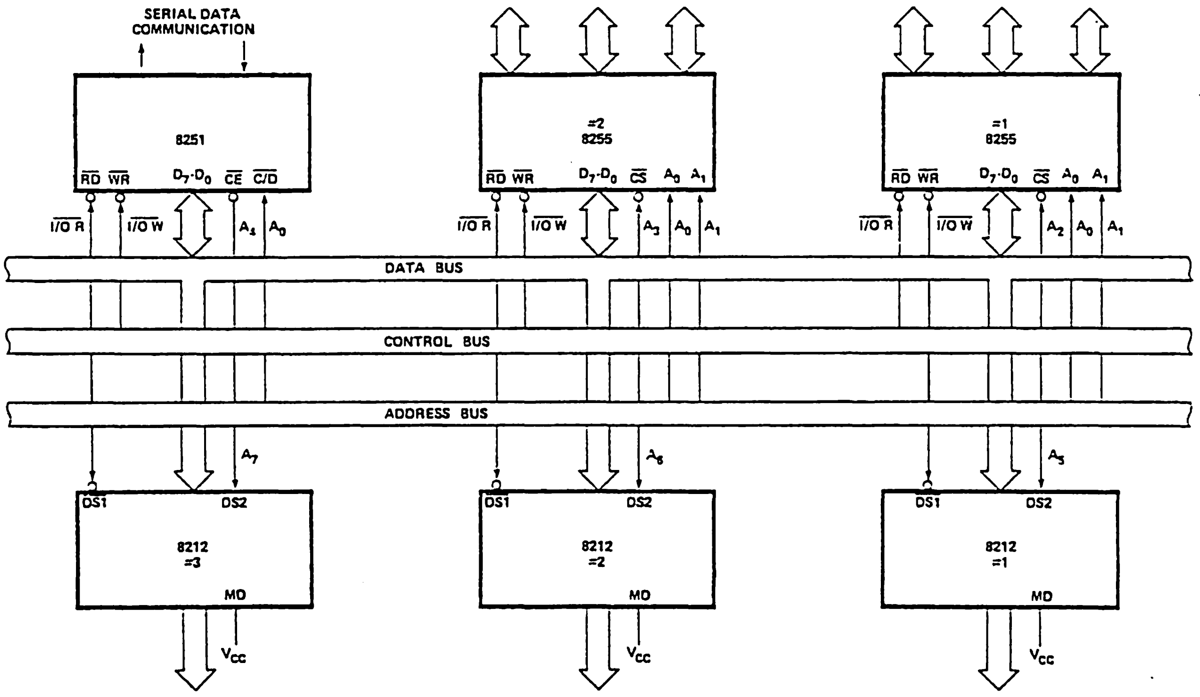


Figure 3-16. Typical I/O Interface.

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:

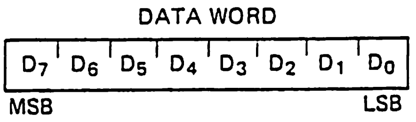
- **Data Transfer Group**—move data between registers or between memory and registers
- **Arithmetic Group** — add, subtract, increment or decrement data in registers or in memory
- **Logical Group** — AND, OR, EXCLUSIVE-OR, compare, rotate or complement data in registers or in memory
- **Branch Group** — 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.

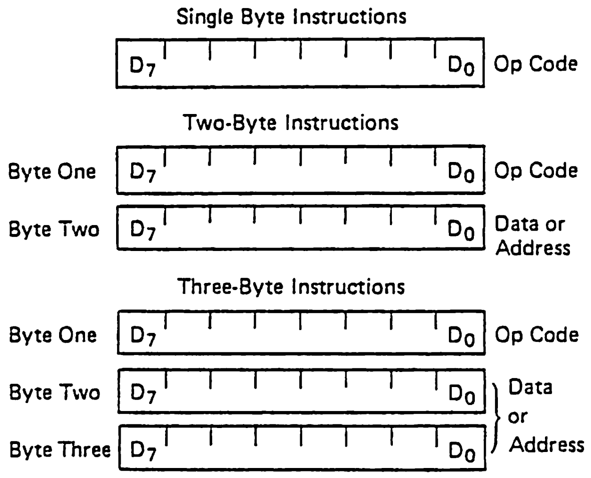
The 8080 can directly address up to 65,536 bytes of memory, which may consist of both read-only memory (ROM) elements and random-access memory (RAM) elements (read/write memory).

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



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.
$\leftarrow$	"Is transferred to"
$\wedge$	Logical AND
$\veebar$	Exclusive OR
$\vee$	Inclusive OR
+	Addition
−	Two's complement subtraction
*	Multiplication
$\leftrightarrow$	"Is exchanged with"
$\overline{}$	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)

$(r1) \leftarrow (r2)$

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)

$(r) \leftarrow ((H)\ (L))$

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)

$((H)\ (L)) \leftarrow (r)$

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)

$(r) \leftarrow (\text{byte } 2)$

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)

$((H)\ (L)) \leftarrow (\text{byte } 2)$

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)

$(rh) \leftarrow (\text{byte } 3),$
$(rl) \leftarrow (\text{byte } 2)$

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) $\leftarrow$ ((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)) $\leftarrow$ (A)

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

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) $\leftarrow$ ((byte 3)(byte 2))

(H) $\leftarrow$ ((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)) $\leftarrow$ (L)

((byte 3)(byte 2) + 1) $\leftarrow$ (H)

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

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) $\leftarrow$ ((rp))

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

0	0	R	P	1	0	1	0

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

STAX rp (Store accumulator indirect)

((rp)) $\leftarrow$ (A)

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

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) $\leftrightarrow$ (D)

(L) $\leftrightarrow$ (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.

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

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

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

Mnemonic	Description	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
				Instruction Code[1]						
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	C	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.

ALTAIR 8800b SECTION III THEORY OF OPERATION

3-1. GENERAL

This section contains information needed to understand the operation of the MITS Altair 8800b computer (8800b). It contains a basic description of the logic symbols used in the 8800b schematics and detailed theory of the 8800b Central Processing Unit, Interface and Front Panel circuits.

3-2. LOGIC CIRCUITS

The logic circuits used in the 8800b drawings are presented as a tabular listing in Table 3-1. The table is constructed to present the functional name, symbolic representation, and a brief description of each logic circuit. Where applicable, a truth table is provided to aid in understanding circuit operation. Although Table 3-1 does not include every logic circuit used in the drawings, all unmentioned circuits (and their symbolic representations) are variations of the circuits presented with their functional descriptions basically the same. The active state of the inputs and outputs of the logic circuits is graphically displayed by small circles. A small circle, at an input to a logic circuit, indicates that the input is an active LOW; that is, a LOW signal will enable the input. A small circle, at the output of a logic circuit, indicates that the output is an active LOW; that is, the output is low in the actuated state. Conversely, the absence of a small circle indicates that the input or output is active HIGH.

Table 3-1. Symbol Definitions

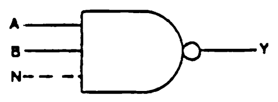
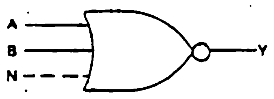
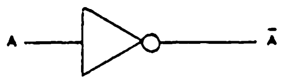
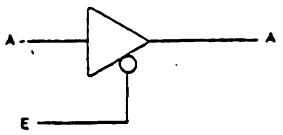
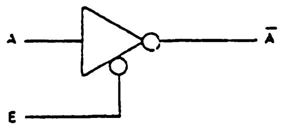
NAME	LOGIC SYMBOL	DESCRIPTION
NAND gate	 $Y = \overline{AB \ldots N}$	The NAND gate performs one of the fundamental logic functions. All of the inputs have to be enabled (HIGH) to produce the desired (LOW) output. The output is HIGH if any of the inputs are LOW.
NOR gate	 $Y = \overline{A + B \ldots +N}$	The NOR gate performs one of the fundamental logic functions. Any of the inputs need to be enabled (HIGH) to produce the desired (LOW) output. The output is HIGH if all of the inputs are LOW.
Inverter		The inverter is a device whose output is the opposite state of the input.
Non-Inverting Bus Driver		The non-inverting bus driver is a device whose output is the same state as the input. Data is enabled through the device by applying a (LOW) signal to the E input.
Inverting Bus Driver		The inverting bus driver is a device whose output is the opposite state of the input. Data is enabled through the driver by applying a (LOW) signal to the E input.

Table 3-1. Symbol Definitions - Continued

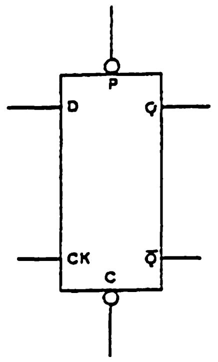
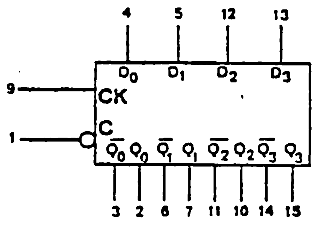
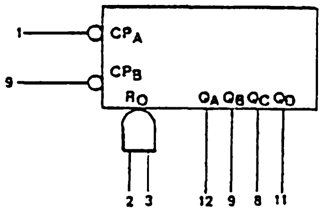
NAME	LOGIC SYMBOL	DESCRIPTION
Edge triggered D type flip-flop	 **Truth Table** T_n — T_{n+1} D, Q, $\overline{Q}$ L, L, H H, H, L	Applying a LOW signal to the preset input (P) sets the flip-flop with output Q HIGH and output $\overline{Q}$ LOW. Applying a LOW signal to the clear input (C) resets the flip-flop with Q LOW and $\overline{Q}$ HIGH. This method of setting and resetting the flip-flop is independent of the clock (asynchronous). If a signal is applied to the D input, the flip-flop Q output is directly affected on the positive edge of the clock (truth table).
QUAD D flip-flop	 4 5 12 13 D_0 D_1 D_2 D_3 9 CK 1 C $\overline{Q}_0$ Q_0 $\overline{Q}_1$ Q_1 $\overline{Q}_2$ Q_2 $\overline{Q}_3$ Q_3 3 2 6 7 11 10 14 15	The information on the D inputs is stored during the positive edge of the clock (CK). The clear (C) input, when LOW, resets all flip-flops independent of the clock or D inputs.
4-Bit Binary Ripple Counter	 1 CP_A 9 CP_B RO Q_A Q_B Q_C Q_D 2 3 12 9 8 11	The 4-bit binary ripple counter operation requires that the QA output be externally connected to input CP_B. The input count pulses (negative edge) are applied to input CP_A enabling a divide by 2, 4, 8, and 16 at the QA, QB, QC, and QD outputs. The reset (RO) input resets the counter regardless of the clock input (CP_A) when both inputs are HIGH.

Table 3-1. Symbol Definitions - Continued

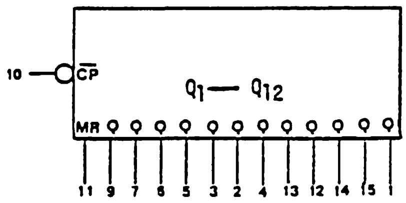
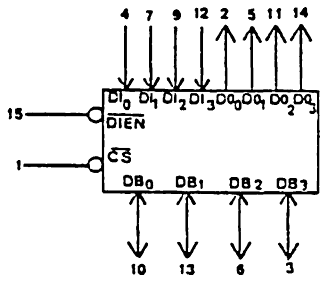
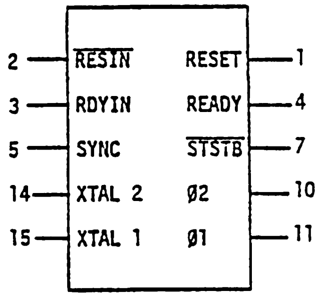
NAME	LOGIC SYMBOL	DESCRIPTION
12-Bit Binary Counter	 10 CP Q1 — Q12 MR 11 9 7 6 5 3 2 4 13 12 14 15 1	The 12-bit counter is triggered on the negative edge of the clock input (CP). A HIGH on the master reset input (MR) clears all counter stages and forces all outputs (Q0-Q11) LOW which is independent of the clock input.
Bi-Directional Device	 4 7 9 12 2 5 11 14 DI_0 DI_1 DI_2 DI_3 DO_0 DO_1 DO_2 DO_3 15 $\overline{DIEN}$ 1 $\overline{CS}$ DB_0 DB_1 DB_2 DB_3 10 13 6 3	Output data from a device is present on the DI_0-DI_3 lines and is enabled when $\overline{DIEN}$ and $\overline{CS}$ are LOW. Lines DB_0-DB_3 transfer the data to the receiving unit. Input data to the device is present on the DB_0-DB_3 lines and is enabled when $\overline{DIEN}$ is HIGH and $\overline{CS}$ is LOW. Input data is transferred to the device on the DO_0-DO_3 lines.
Clock Generator	 2 $\overline{RESIN}$ RESET 1 3 RDYIN READY 4 5 SYNC $\overline{STSTB}$ 7 14 XTAL 2 Ø2 10 15 XTAL 1 Ø1 11	The XTAL 1 and 2 inputs allow for an external crystal connection which produces a Ø1 and Ø2 master clock for the 8800b. The SYNC input from the 8080 (CPU) and internal timing generate a LOW status strobe ($\overline{STSTB}$) signal. The reset in ($\overline{RESIN}$) input generates a RESET output to condition the 8080 (CPU). A HIGH ready in (RDYIN) input generates a READY output to enable the CPU.

Table 3-1. Symbol Definitions - Continued

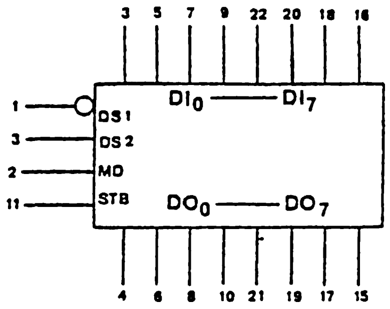
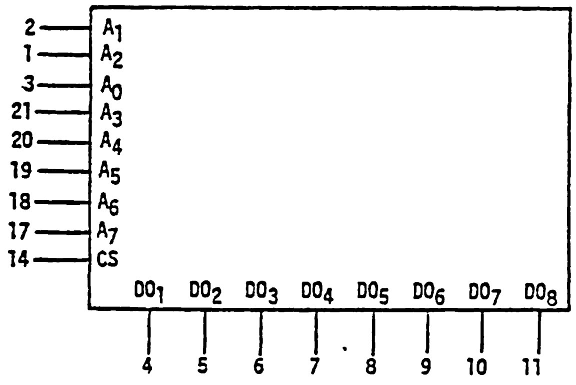
NAME	LOGIC SYMBOL	DESCRIPTION
Data Latch	 3 5 7 9 22 20 18 16 1 3 2 11 DS1 DS2 MD STB DI_0 — DI_7 DO_0 — DO_7 4 6 8 10 21 19 17 15	The data latch is used to store or transfer data on the DO_0-DO_7 outputs by affecting the data latch control inputs. There are several different ways used to store data or transfer it to the data latch. When data is presented to the DI_0-DI_7 inputs and the device selection 2 (DS2), mode MD, and strobe (STB) are HIGH, a LOW device selection 1 ($\overline{DS1}$) allows the input data to be present on the DO_0-DO_7 outputs. When data is presented to the DI_0-DI_7 inputs and MD and STB are HIGH, a HIGH DS2 and LOW $\overline{DS1}$ allow the input data to be present on the DO_0-DO_7 outputs. When data is presented to the DI_0-DI_7 inputs and $\overline{DS1}$ and MD are LOW, a HIGH DS2 and STB allow the input data to be present on the DO_0-DO_7 outputs. When data is presented to the DI_0-DI_7 inputs, and MD and DS2 are HIGH with $\overline{DS1}$ LOW, the input data is directly transferred to the DO_0-DO_7 outputs as long as these states are present.

Table 3-1. Symbol Definitions - Continued

NAME	LOGIC SYMBOL	DESCRIPTION
PROM (programmable read only memory)	 2 1 3 21 20 19 18 17 14 A1 A2 A0 A3 A4 A5 A6 A7 CS DO1 DO2 DO3 DO4 DO5 DO6 DO7 DO8 4 5 6 7 8 9 10 11	When the chip select input (CS) is LOW, the binary address at input A_0 through A_7 is decoded to select one of 256 address locations. The data is present on the DO_1 through DO_8 outputs.

3-3. INTEL 8080 MICROCOMPUTER SYSTEMS USER'S INFORMATION

Pages 3-9 through 3-38 are excerpts from the Intel 8080 Microcomputer Systems User's Manual, reprinted by permission of Intel Corporation, Copyright, 1975. Included is information on the 8080A Microprocessor, the 8212 Input/Output Port, the 8216 Bi-Directional Bus Driver, and the 8224 Clock Generator and Driver. It is recommended that a good understanding of these integrated circuit operations be developed before continuing this section.

intel® Silicon Gate MOS 8080A

SINGLE CHIP 8-BIT N-CHANNEL MICROPROCESSOR

The 8080A is functionally and electrically compatible with the Intel® 8080.

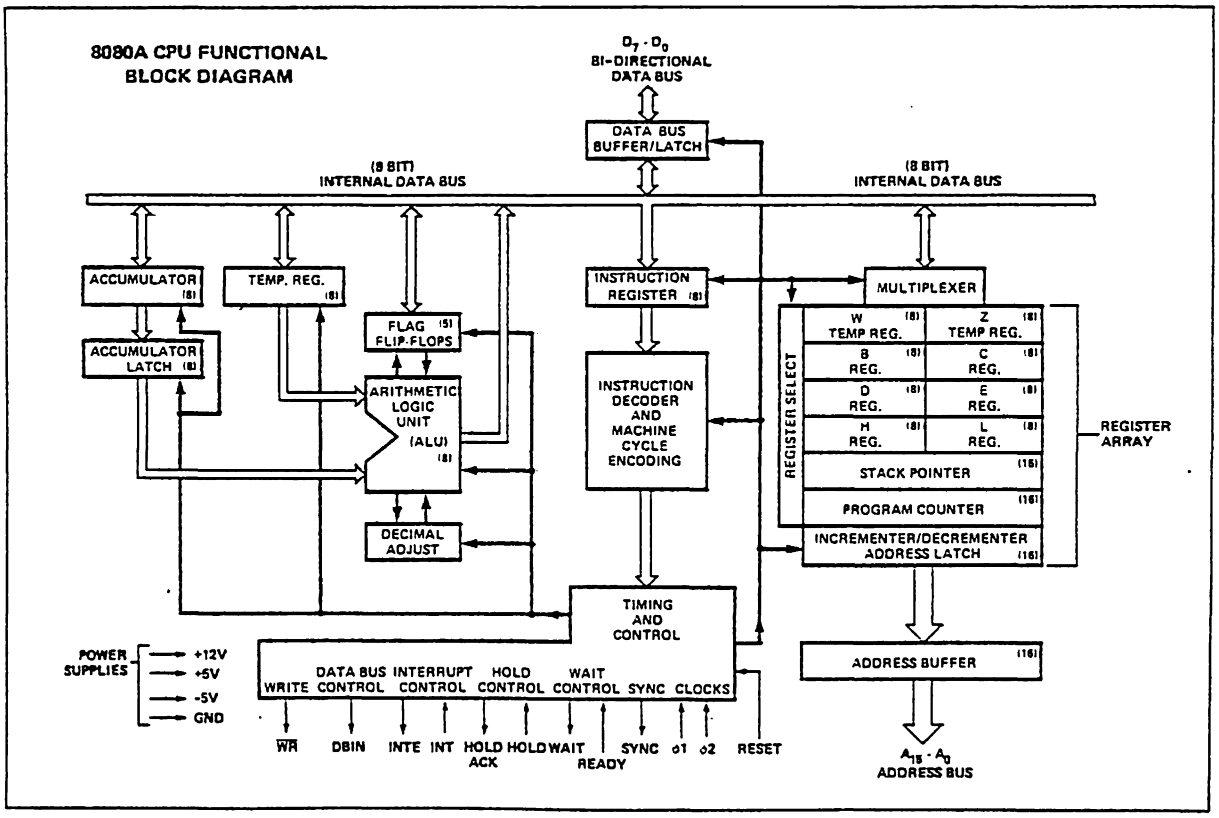
- **TTL Drive Capability**
- **2 μs Instruction Cycle**
- **Powerful Problem Solving Instruction Set**
- **Six General Purpose Registers and an Accumulator**
- **Sixteen Bit Program Counter for Directly Addressing up to 64K Bytes of Memory**
- **Sixteen Bit Stack Pointer and Stack Manipulation Instructions 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

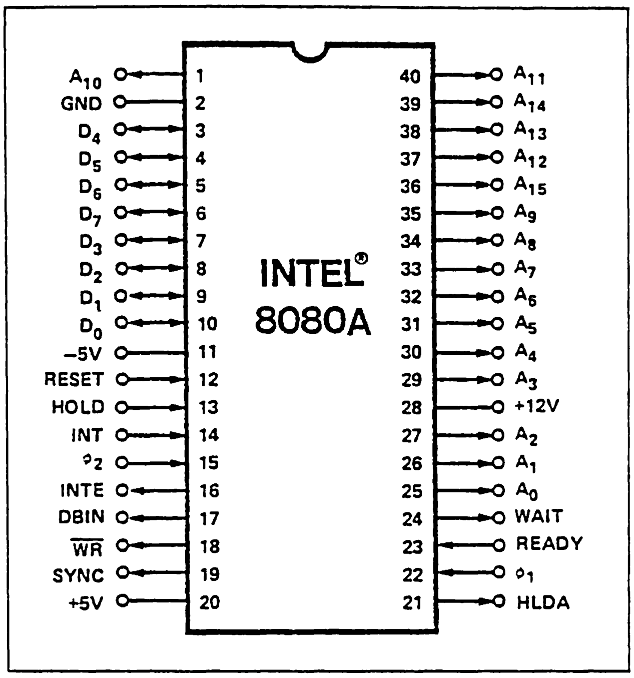
The following describes the function of all of the 8080A I/O pins. Several of the descriptions refer to internal timing periods.

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.



Pin Configuration

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.

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,
V_{OH}	Output High Voltage	3.7			V	I_{OH} = -150μA.
$I_{DD\,(AV)}$	Avg. Power Supply Current (V_{DD})		40	70	mA	Operation
$I_{CC\,(AV)}$	Avg. Power Supply Current (V_{CC})		60	80	mA	T_{CY} = .48 μsec
$I_{BB\,(AV)}$	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

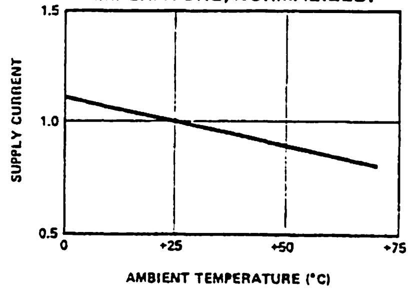
T_A = 25°C $V_{CC} = V_{DD} = V_{SS}$ = 0V, V_{BB} = -5V

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}

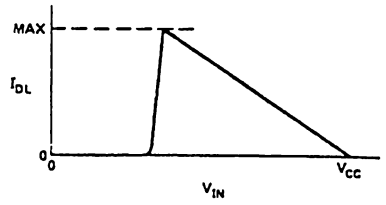
NOTES:

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



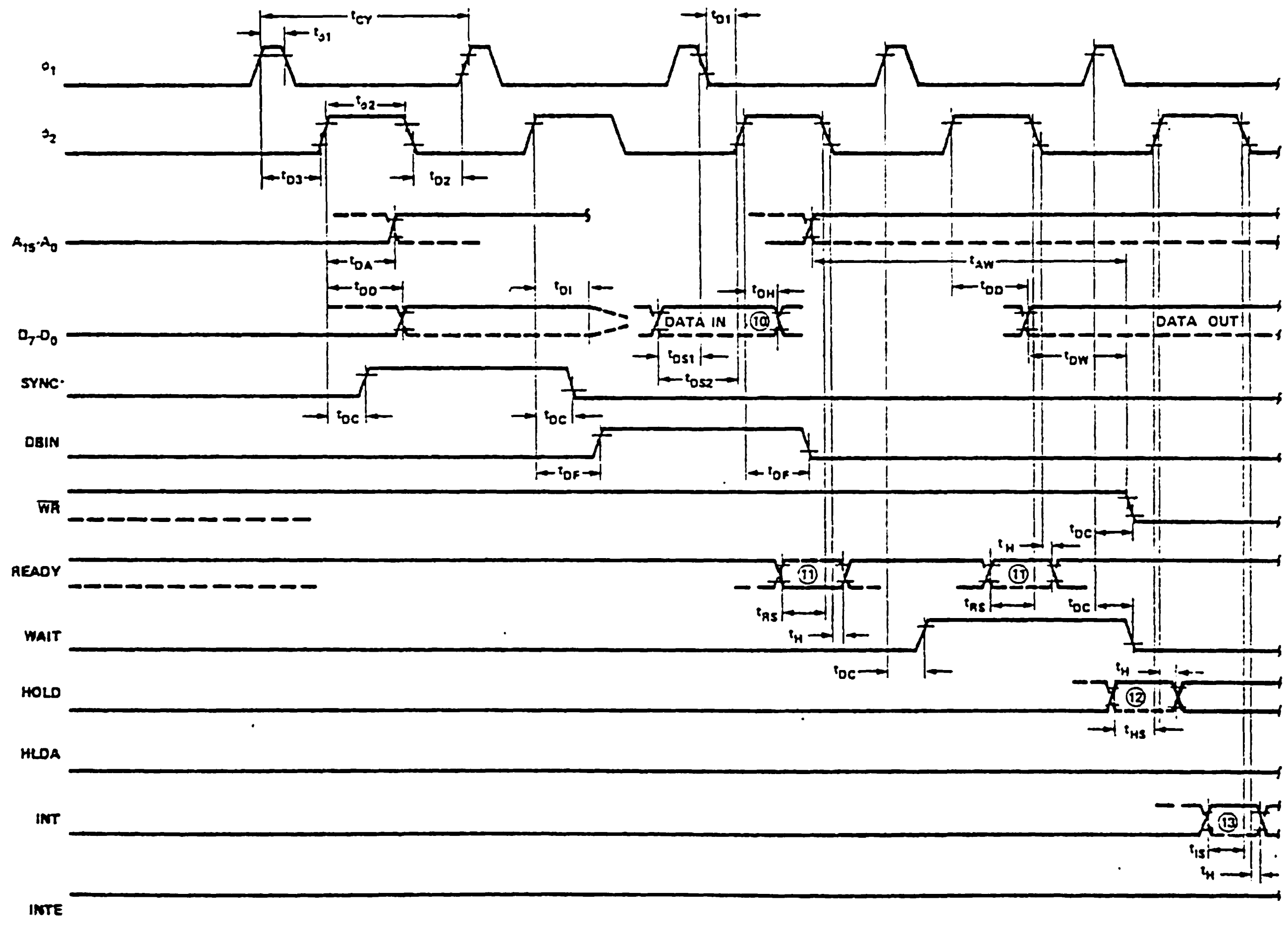
A.C. CHARACTERISTICS

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]

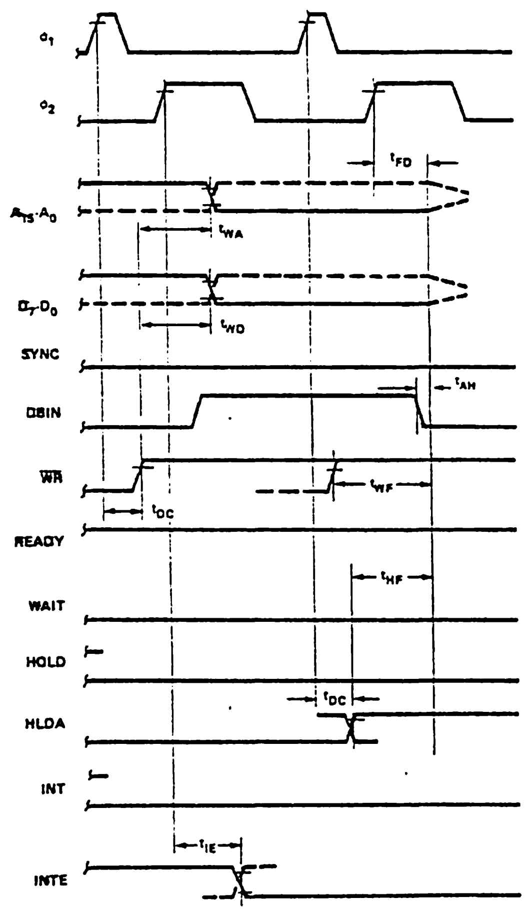
(Note: Timing measurements are made at the following reference voltages: CLOCK "1" = 8.0V "0" = 1.0V; INPUTS "1" = 3.3V, "0" = 0.8V; OUTPUTS "1" = 2.0V, "0" = 0.8V.)



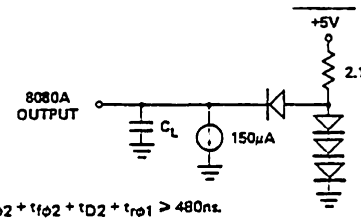
A.C. CHARACTERISTICS (Continued)

$T_A = 0°C$ 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	

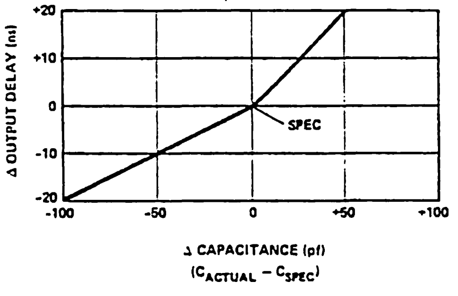


NOTES:
1. Data input should be enabled with DBIN status. No bus conflict can then occur and data hold time is assured. t_{DH} = 50 ns or t_{DF}, whichever is less.
2. Load Circuit.



3. $t_{CY} = t_{D3} + t_{r\phi2} + t_{\phi2} + t_{f\phi2} + t_{D2} + t_{r\phi1} \geq 480ns$.

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 dalay 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. $t_{AW} = 2\ t_{CY} - t_{D3} - t_{r\phi2}$ -140nsec.
6. $t_{DW} = t_{CY} - t_{D3} - t_{r\phi2}$ -170nsec.
7. If not HLDA, $t_{WD} = t_{WA} = t_{D3} + t_{r\phi2}$ +10ns. If HLDA, $t_{WD} = t_{WA} = t_{WF}$.
8. $t_{HF} = t_{D3} + t_{r\phi2}$ -50ns.
9. $t_{WF} = t_{D3} + t_{r\phi2}$ -10ns
10. Data in must be stable for this period during DBIN $\cdot T_3$. 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.

$\boxed{D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0}$

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.

One Byte Instructions		TYPICAL 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	Instruction Code[1] D7	D6	D5	D4	D3	D2	D1	D0	Clock[2] Cycles
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

Mnemonic	Description	Instruction Code[1] D7	D6	D5	D4	D3	D2	D1	D0	Clock[2] Cycles
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	C	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

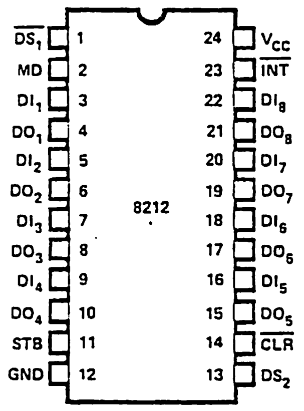
EIGHT-BIT INPUT/OUTPUT PORT

- **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**

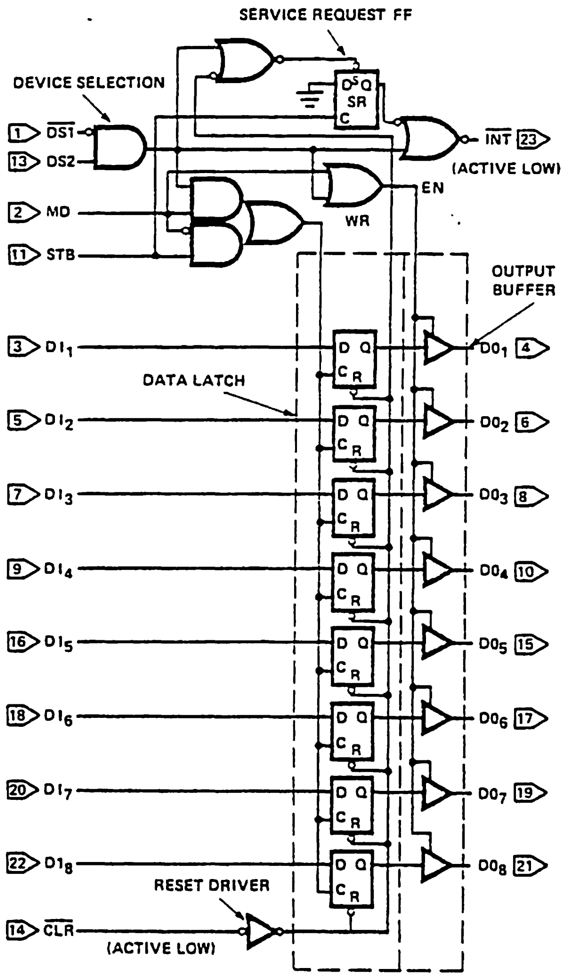
The 8212 input/output port consists of an 8-bit latch with 3-state output buffers along with control and device selection logic. Also included is a service request flip-flop for the generation and control of interrupts to the microprocessor.

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



LOGIC DIAGRAM



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)

April, 1977
8800b

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 ($\overline{\text{CLR}}$). (Note: Clock (C) Overides Reset ($\overline{\text{CLR}}$).)

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 $\overline{\text{DS1}}$, DS2, MD and STB. These inputs are used to control device selection, data latching, output buffer state and service request flip-flop.

$\overline{\text{DS1}}$, DS2 (Device Select)

These 2 inputs are used for device selection. When $\overline{\text{DS1}}$ is low and DS2 is high ($\overline{\text{DS1}}$ · 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 ($\overline{\text{DS1}}$ · DS2).

When MD is low (input mode) the output buffer state is determined by the device selection logic ($\overline{\text{DS1}}$ · DS2) and the source of clock (C) to the data latch is the STB (Strobe) input.

STB (Strobe)

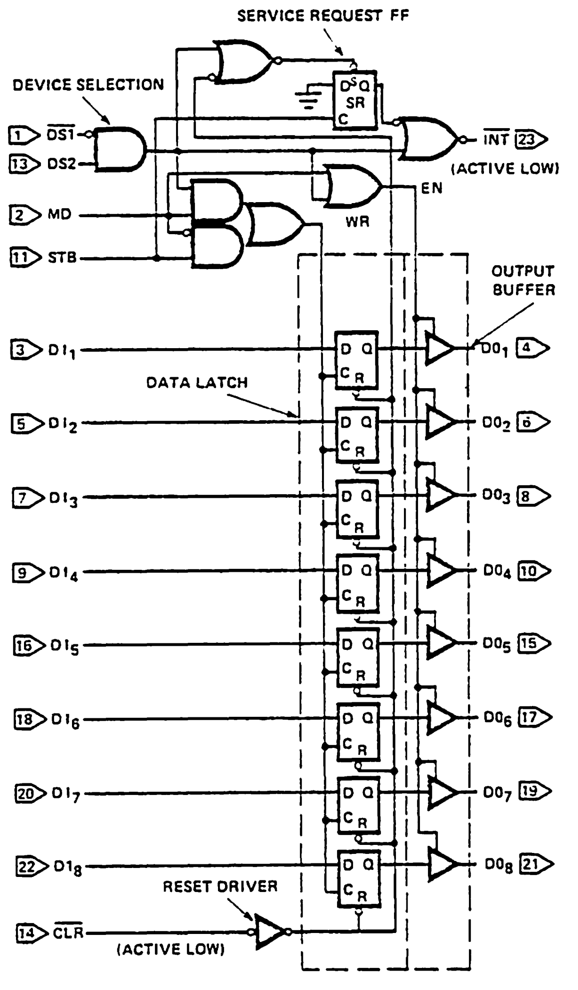
This input is used as the clock (C) to the data latch for the input mode MD = 0) and to synchronously reset the service request flip-flop (SR).

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 $\overline{\text{CLR}}$ 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 ($\overline{\text{DS1}}$ · DS2). The output of the "NOR" gate ($\overline{\text{INT}}$) 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

$\overline{\text{CLR}}$ – 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

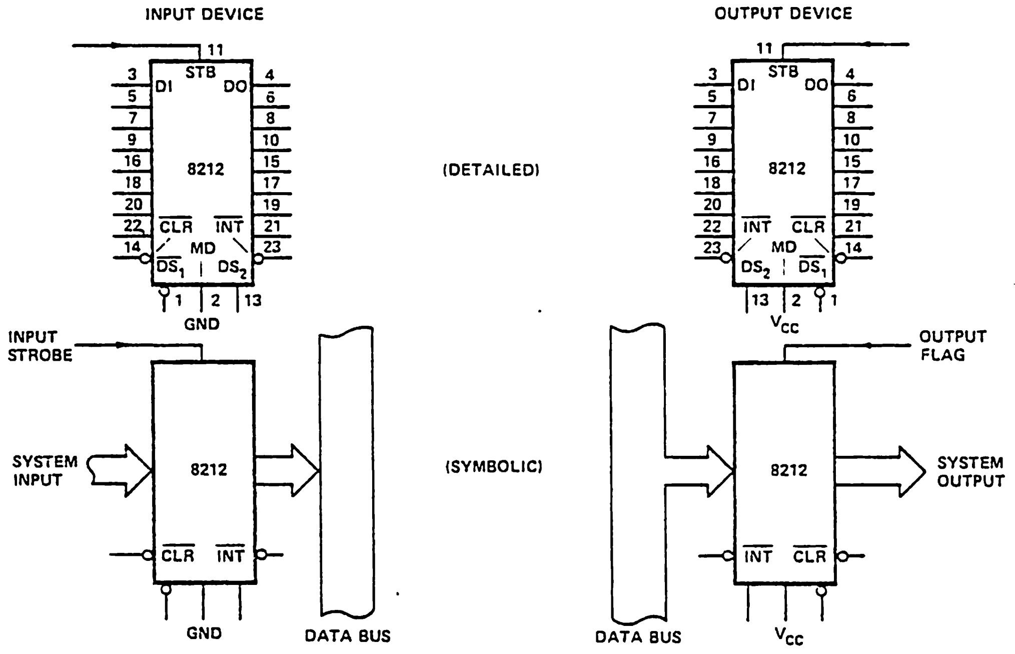
*INTERNAL SR FLIP-FLOP

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



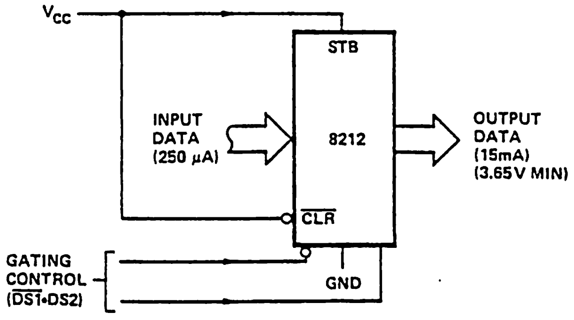
II. Gated Buffer (3 - STATE)

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 $\overline{\text{DS1}}$ 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.

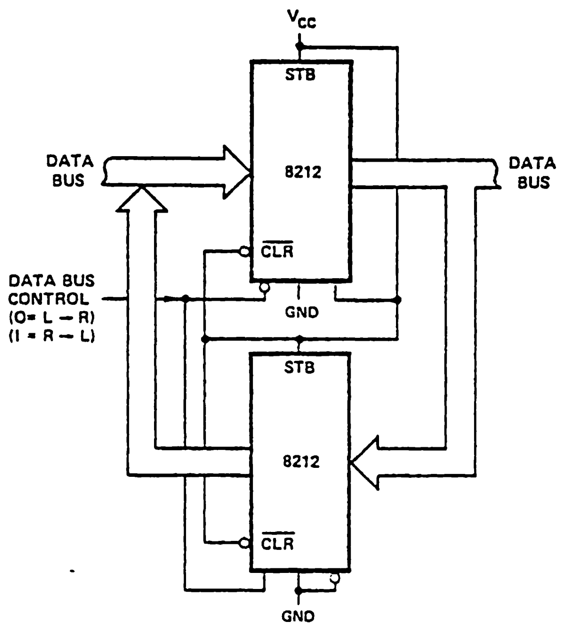
GATED BUFFER
3-STATE



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.

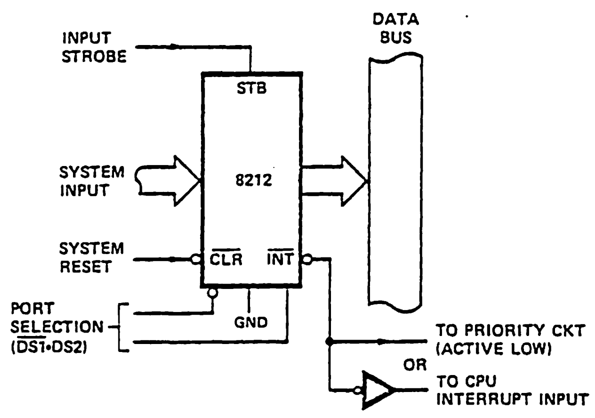
BI-DIRECTIONAL BUS DRIVER



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.

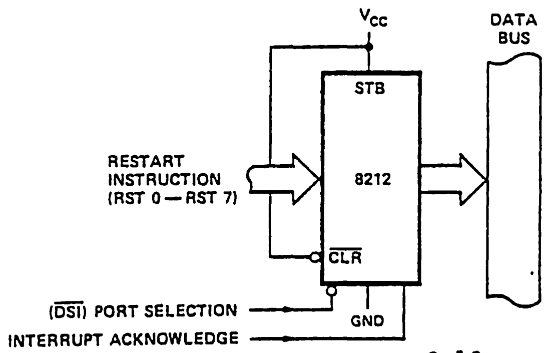
INTERRUPTING INPUT PORT



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

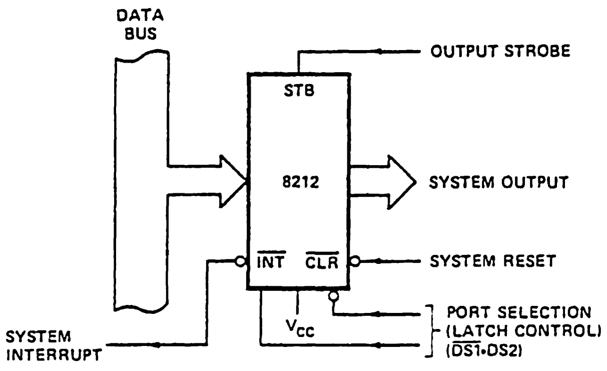


V_{CC}
DATA BUS
STB
RESTART INSTRUCTION (RST 0 — RST 7)
8212
$\overline{CLR}$
($\overline{DS1}$) PORT SELECTION
GND
INTERRUPT ACKNOWLEDGE

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)



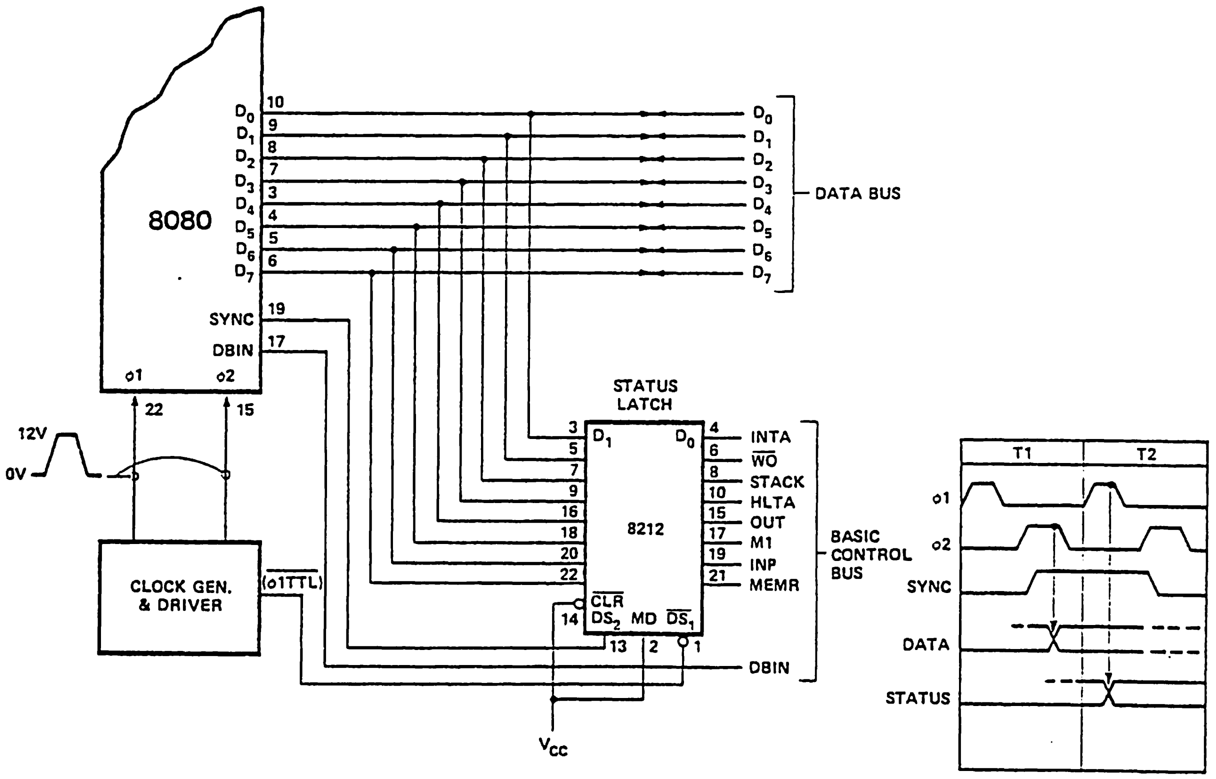
VII. 8080 Status Latch

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.

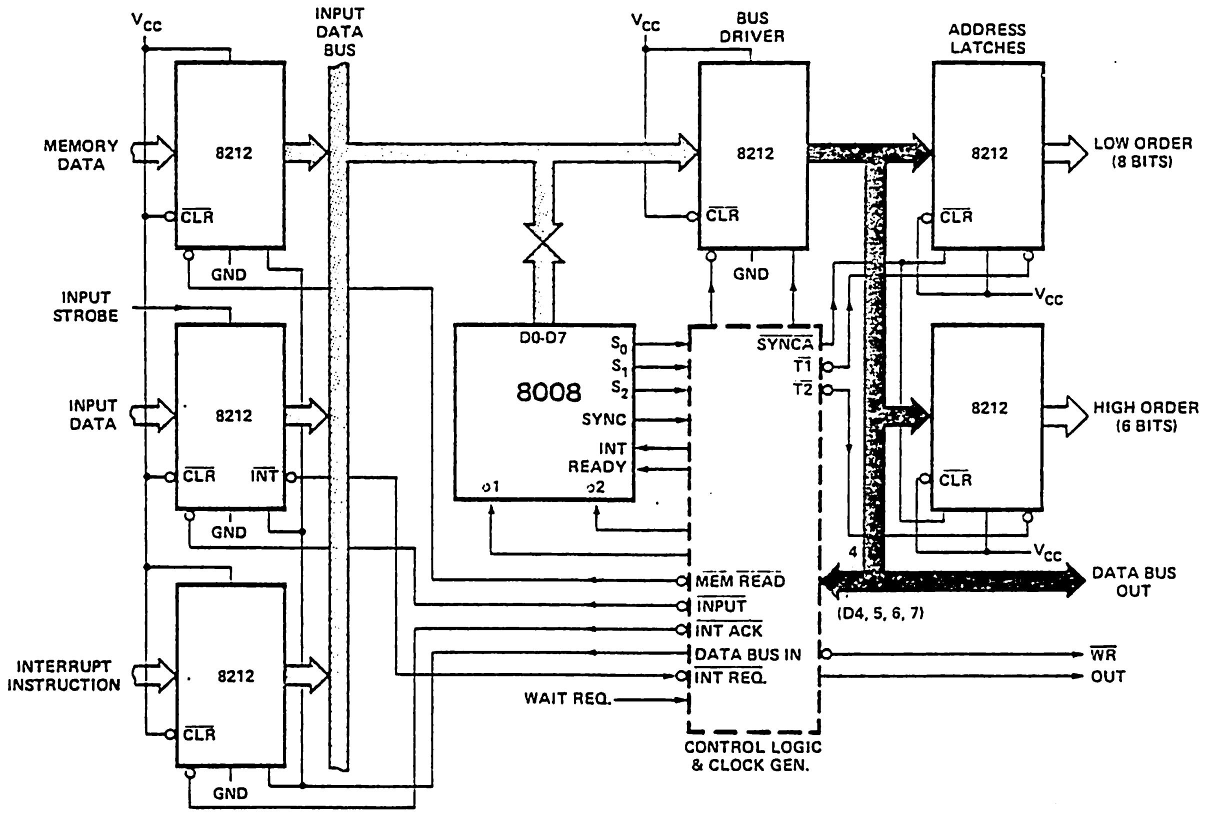
8080 STATUS LATCH



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

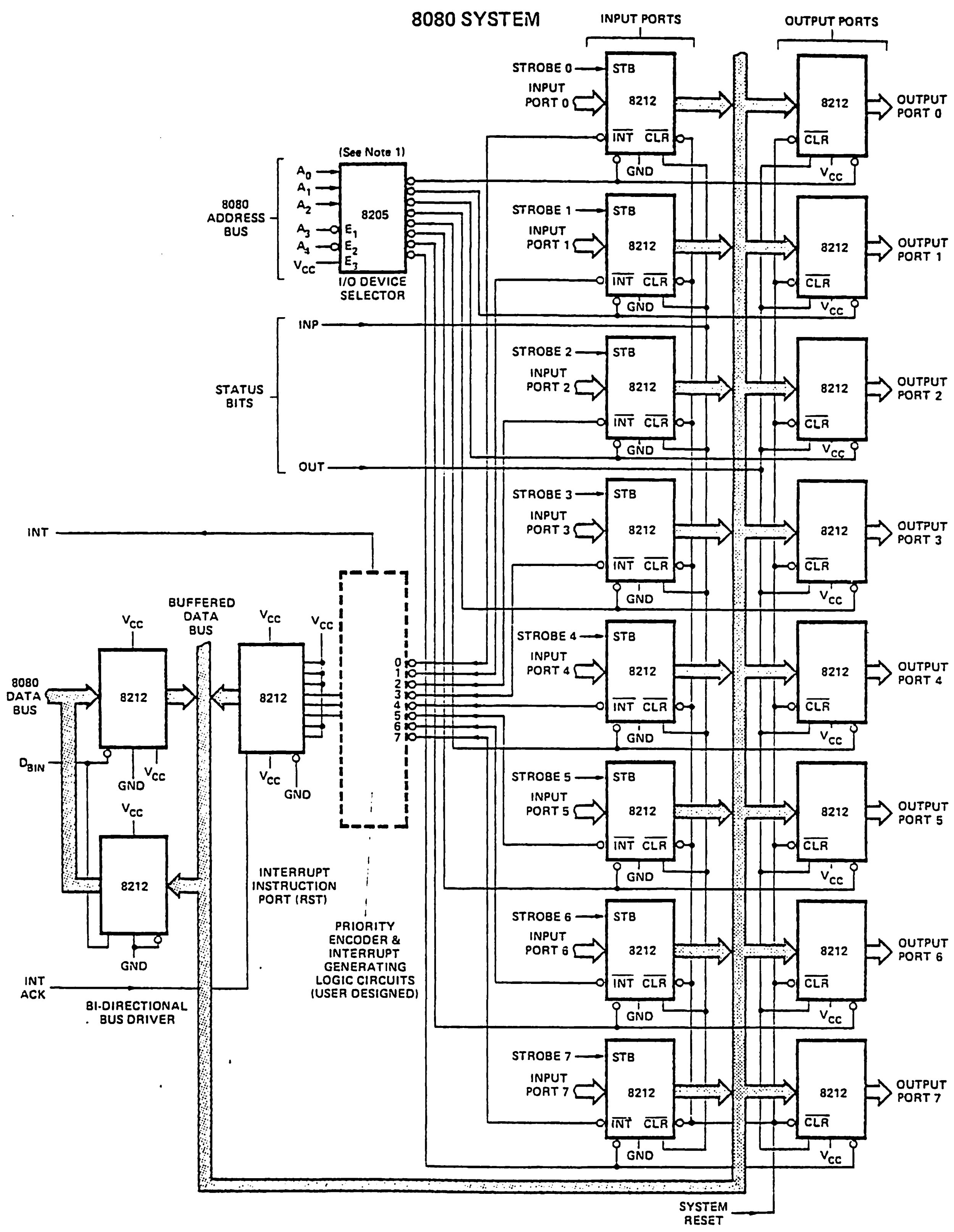
This drawing shows the 8212 used in the I/O section of an 8080 microcomputer system. The system consists of 8 input ports, 8 output ports, 8 level priority systems, and a bidirectional bus driver. (The data bus within the system is darkened for emphasis).

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 $\overline{\text{INT}}$ 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.*

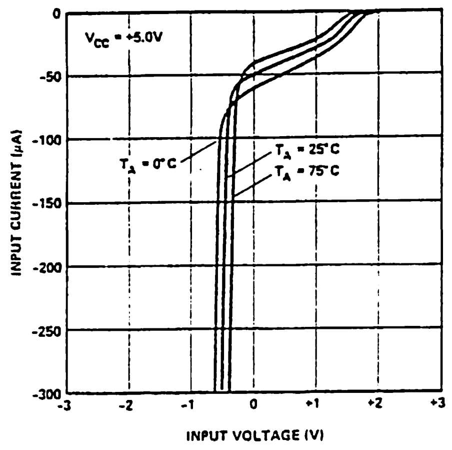
D.C. Characteristics

T_A = 0°C to +75°C V_{CC} = +5V ±5%

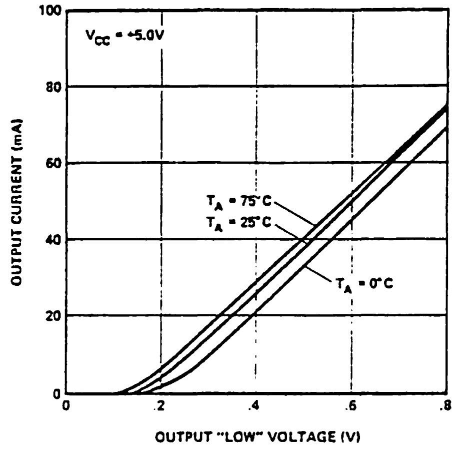
Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
I_F	Input Load Current ACK, DS_2, CR, DI_1-DI_8 Inputs			−.25	mA	V_F = .45V
I_F	Input Load Current MD Input			−.75	mA	V_F = .45V
I_F	Input Load Current DS_1 Input			−1.0	mA	V_F = .45V
I_R	Input Leakage Current ACK, DS, CR, DI_1-DI_8 Inputs			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

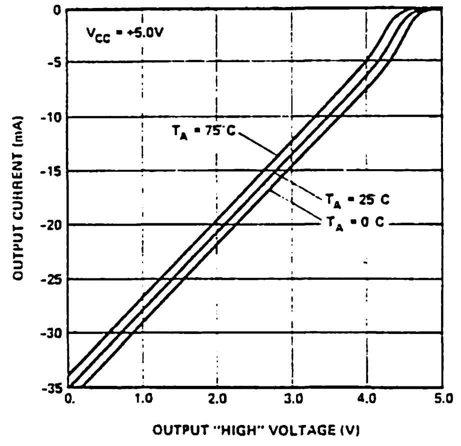
INPUT CURRENT VS. INPUT VOLTAGE



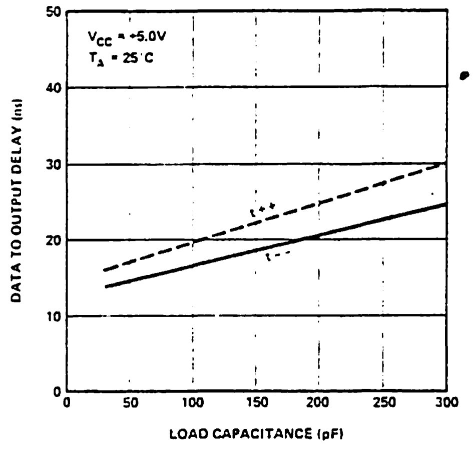
OUTPUT CURRENT VS. OUTPUT "LOW" VOLTAGE



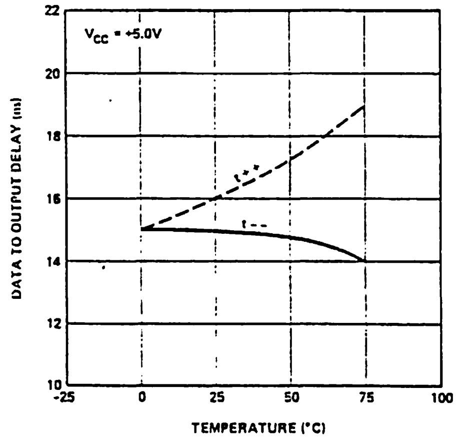
OUTPUT CURRENT VS.
OUTPUT "HIGH" VOLTAGE



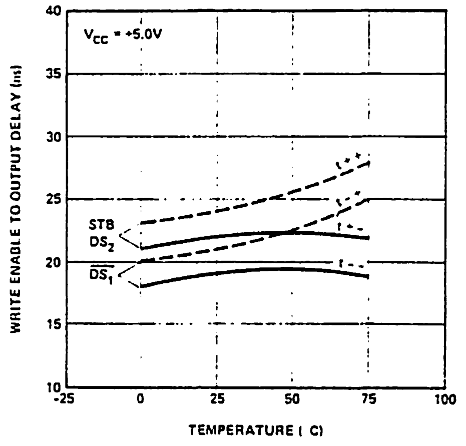
DATA TO OUTPUT DELAY
VS. LOAD CAPACITANCE



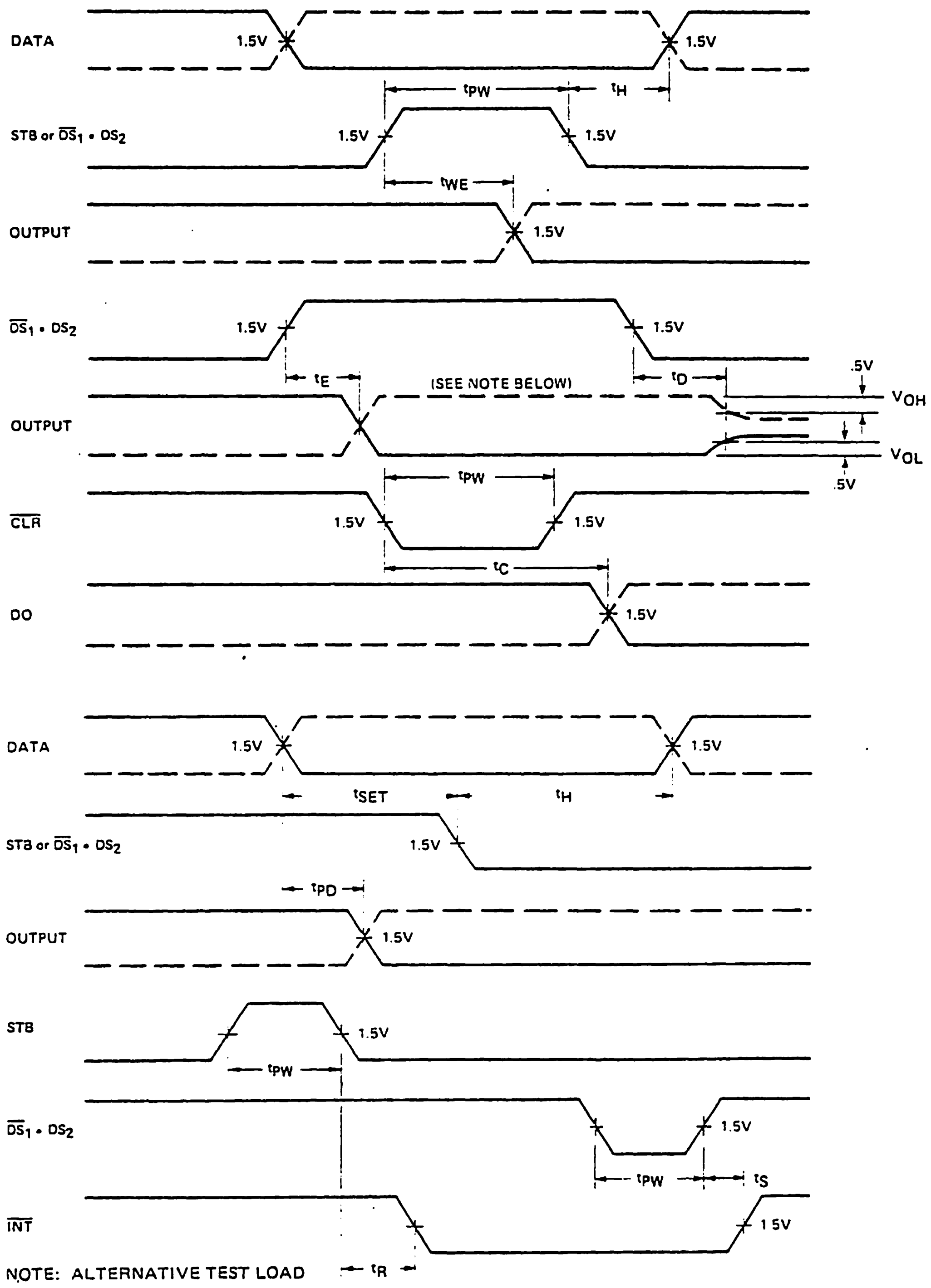
DATA TO OUTPUT DELAY
VS. TEMPERATURE



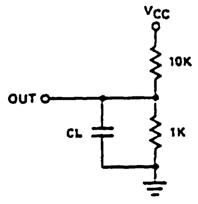
WRITE ENABLE TO OUTPUT DELAY
VS. TEMPERATURE



Timing Diagram



NOTE: ALTERNATIVE TEST LOAD



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.

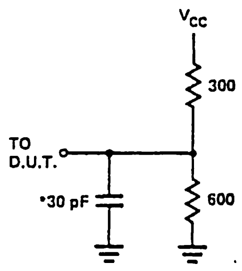
Switching Characteristics

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

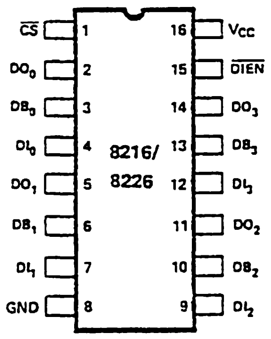
- **Data Bus Buffer Driver for 8080 CPU**
- **Low Input Load Current — .25 mA Maximum**
- **High Output Drive Capability for Driving System Data Bus**
- **3.65V Output High Voltage for Direct Interface to 8080 CPU**
- **Three State Outputs**
- **Reduces System Package Count**

The 8216/8226 is a 4-bit bi-directional bus driver/receiver.

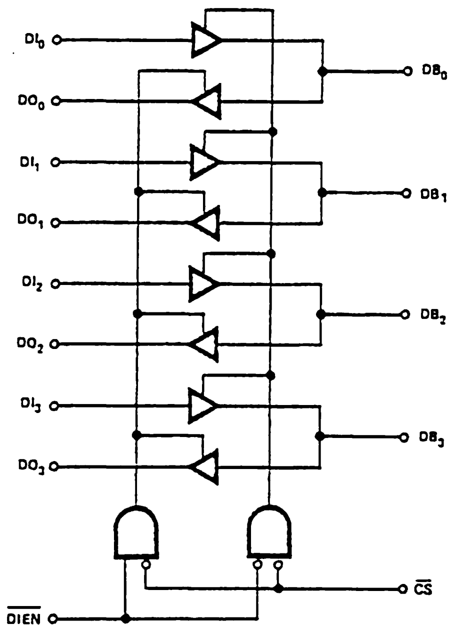
All inputs are low power TTL compatible. For driving MOS, the DO outputs provide a high 3.65V V_{OH}, and for high capacitance terminated bus structures, the DB outputs provide a high 50mA I_{OL} capability.

A non-inverting (8216) and an inverting (8226) are available to meet a wide variety of applications for buffering in microcomputer systems.

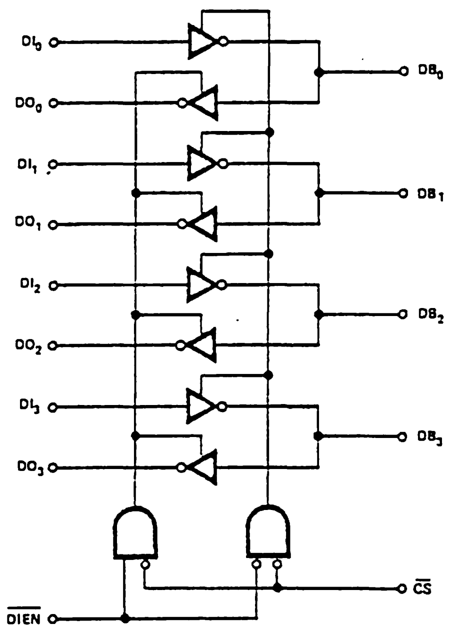
PIN CONFIGURATION



LOGIC DIAGRAM 8216



LOGIC DIAGRAM 8226



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

April, 1977
8800b

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

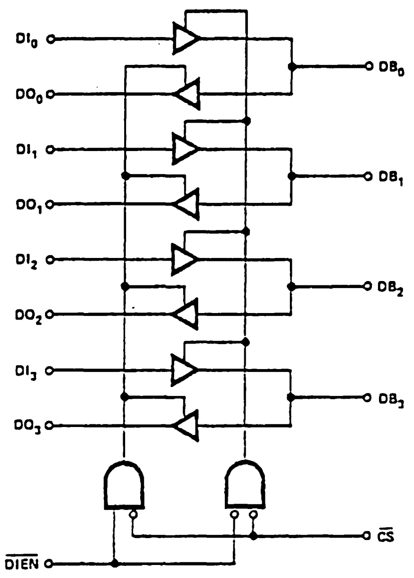
Each buffered line of the four bit driver consists of two separate buffers that are tri-state in nature to achieve direct bus interface and bi-directional capability. On one side of the driver the output of one buffer and the input of another are tied together (DB), this side is used to interface to 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 $\overline{\text{DIEN}}$, $\overline{\text{CS}}$

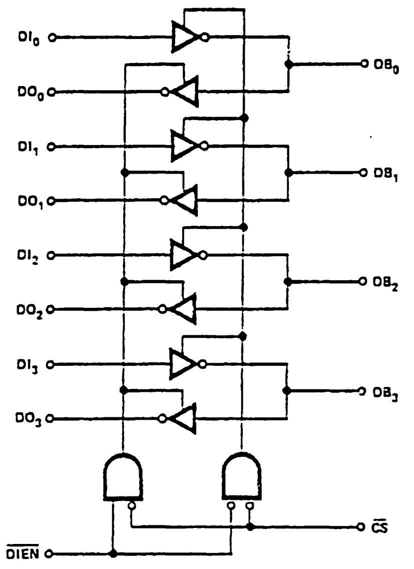
The $\overline{\text{CS}}$ input is actually a device select. When it is "high" the output drivers are all forced to their high-impedance state. When it is at "zero" the device is selected (enabled) and the direction of the data flow is determined by the $\overline{\text{DIEN}}$ input.

The $\overline{\text{DIEN}}$ 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

$\overline{\text{DIEN}}$	$\overline{\text{CS}}$	
0	0	DI → DB
1	0	DB → DO
0	1	HIGH IMPEDANCE
1	1	HIGH IMPEDANCE

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 3.65 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 $\overline{\text{DIEN}}$ inputs to 8216/8226 is connected directly to the 8080. $\overline{\text{DIEN}}$ is tied to DBIN so that proper bus flow is maintained, and $\overline{\text{CS}}$ is tied to $\overline{\text{BUSEN}}$ 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 $\overline{\text{MEMR}}$ signal, which is connected to the $\overline{\text{DIEN}}$ 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 $\overline{\text{I/O R}}$ signal is connected directly to the $\overline{\text{DIEN}}$ 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.

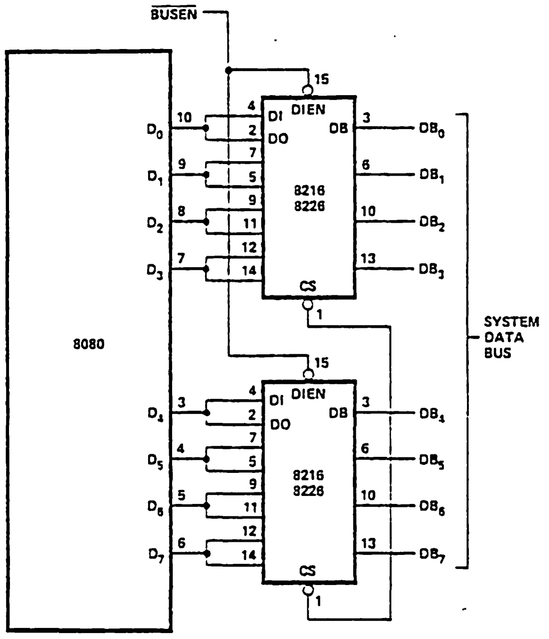


Figure 2. 8080 Data Bus Buffer.

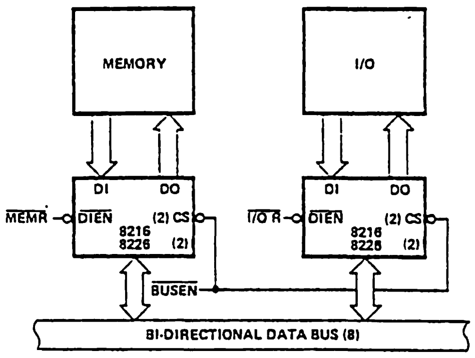


Figure 3. Memory and I/O Interface to a Bi-Directional Bus.

D.C. AND OPERATING CHARACTERISTICS

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias	0°C to 70°C
Storage Temperature	-65°C to +150°C
All Output and Supply Voltages	-0.5V to +7V
All Input Voltages	-1.0V to +5.5V
Output Currents	125 mA

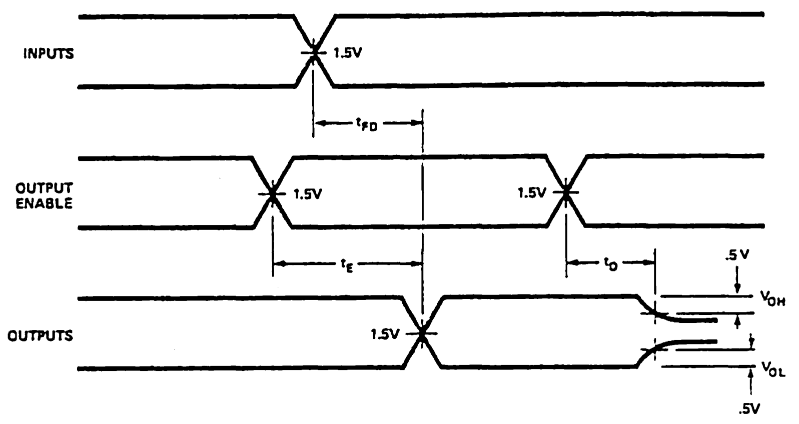
*COMMENT: Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operational sections of this specification is not implied.

T_A = 0°C to +70°C, V_{CC}=+5V ±5%

Symbol	Parameter		Limits		Unit	Conditions
		Min.	Typ.	Max.		
I_{F1}	Input Load Current $\overline{DIEN}$, $\overline{CS}$		-0.15	-.5	mA	V_F = 0.45
I_{F2}	Input Load Current All Other Inputs		-0.08	-.25	mA	V_F = 0.45
I_{R1}	Input Leakage Current $\overline{DIEN}$, $\overline{CS}$			20	µA	V_R = 5.25V
I_{R2}	Input Leakage Current DI Inputs			10	µA	V_R = 5.25V
V_C	Input Forward Voltage Clamp			-1	V	I_C = -5mA
V_{IL}	Input "Low" Voltage			.95	V	
V_{IH}	Input "High" Voltage	2.0			V	
$\|I_O\|$	Output Leakage Current (3-State) DO / DB			20 / 100	µA	V_O = 0.45V/5.25V
I_{CC}	Power Supply Current 8216		95	130	mA	
	Power Supply Current 8226		85	120	mA	
V_{OL1}	Output "Low" Voltage		0.3	.45	V	DO Outputs I_{OL}=15mA DB Outputs I_{OL}=25mA
V_{OL2}	Output "Low" Voltage 8216		0.5	.6	V	DB Outputs I_{OL}=55mA
	Output "Low" Voltage 8226		0.5	.6	V	DB Outputs I_{OL}=50mA
V_{OH1}	Output "High" Voltage	3.65	4.0		V	DO Outputs I_{OH} = -1mA
V_{OH2}	Output "High" Voltage	2.4	3.0		V	DB Outputs I_{OH} = -10mA
I_{OS}	Output Short Circuit Current	-15 -30	-35 -75	-65 -120	mA mA	DO Outputs $V_O \cong 0V$, DB Outputs V_{CC}=5.0V

NOTE: Typical values are for T_A = 25°C, V_{CC} = 5.0V.

WAVEFORMS



A.C. CHARACTERISTICS

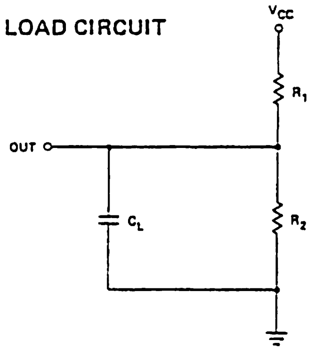
$T_A = 0°C$ to $+70°C$, $V_{CC} = +5V \pm 5\%$

Symbol	Parameter	Limits Min.	Limits Typ.[1]	Limits Max.	Unit	Conditions
T_{PD1}	Input to Output Delay DO Outputs		15	25	ns	C_L=30pF, R_1=300Ω R_2=600Ω
T_{PD2}	Input to Output Delay DB Outputs 8216		20	30	ns	C_L=300pF, R_1=90Ω
	8226		16	25	ns	R_2 = 180Ω
T_E	Output Enable Time 8216		45	65	ns	(Note 2)
	8226		35	54	ns	(Note 3)
T_D	Output Disable Time		20	35	ns	(Note 4)

TEST CONDITIONS:

Input pulse amplitude of 2.5V.
Input rise and fall times of 5 ns between 1 and 2 volts.
Output loading is 5 mA and 10 pF.
Speed measurements are made at 1.5 volt levels.

TEST LOAD CIRCUIT



Capacitance[5]

Symbol	Parameter	Limits Min.	Limits Typ.[1]	Limits Max.	Unit
C_{IN}	Input Capacitance		4	8	pF
C_{OUT1}	Output Capacitance		6	10	pF
C_{OUT2}	Output Capacitance		13	18	pF

TEST CONDITIONS: V_{BIAS} = 2.5V, V_{CC} = 5.0V, T_A = 25°C, f = 1 MHz.

NOTES:
1. Typical values are for T_A = 25°C, V_{CC} = 5.0V.
2. DO Outputs, C_L = 30pF, R_1 = 300/10 KΩ, R_2 = 180/1KΩ; DB Outputs, C_L = 300pF, R_1 = 90/10 KΩ, R_2 = 180/1 KΩ.
3. DO Outputs, C_L = 30pF, R_1 = 300/10 KΩ, R_2 = 600/1K; DB Outputs, C_L = 300pF, R_1 = 90/10 KΩ, R_2 = 180/1 KΩ.
4. DO Outputs, C_L = 5pF, R_1 = 300/10 KΩ, R_2 = 600/1 KΩ; DB Outputs, C_L = 5pF, R_1 = 90/10 KΩ, R_2 = 180/1 KΩ.
5. This parameter is periodically sampled and not 100% tested.

intel®

Schottky Bipolar 8224

CLOCK GENERATOR AND DRIVER FOR 8080A CPU

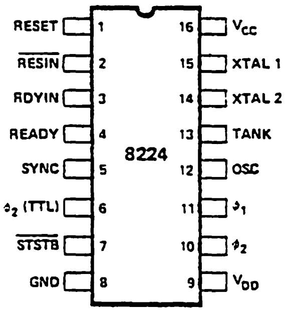
- **Single Chip Clock Generator/Driver for 8080A CPU**
- **Power-Up Reset for CPU**
- **Ready Synchronizing Flip-Flop**
- **Advanced Status Strobe**
- **Oscillator Output for External System Timing**
- **Crystal Controlled for Stable System Operation**
- **Reduces System Package Count**

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.

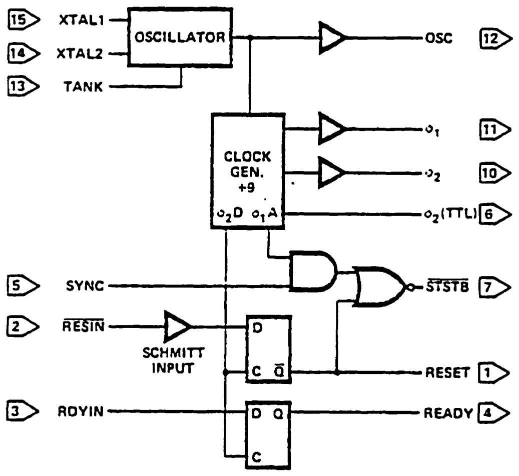
Also included are circuits to provide power-up reset, advance status strobe and synchronization of ready.

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

$\overline{RESIN}$	RESET INPUT
RESET	RESET OUTPUT
RDYIN	READY INPUT
READY	READY OUTPUT
SYNC	SYNC INPUT
$\overline{STSTB}$	STATUS STB (ACTIVE LOW)
ϕ_1	8080 CLOCKS
ϕ_2	8080 CLOCKS

XTAL 1	CONNECTIONS FOR CRYSTAL
XTAL 2	CONNECTIONS FOR CRYSTAL
TANK	USED WITH OVERTONE XTAL
OSC	OSCILLATOR OUTPUT
ϕ_2 (TTL)	ϕ_2 CLK (TTL LEVEL)
V_{CC}	+5V
V_{DD}	-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.

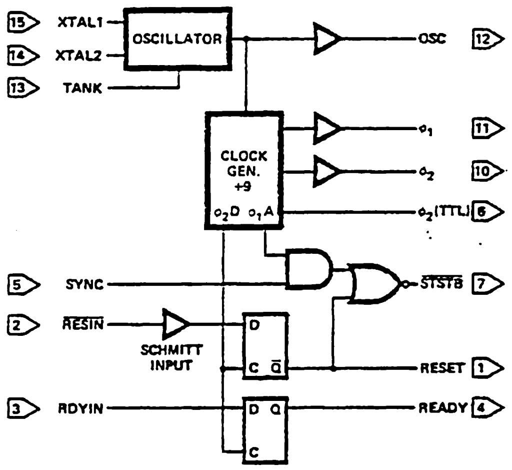
Clock Generator

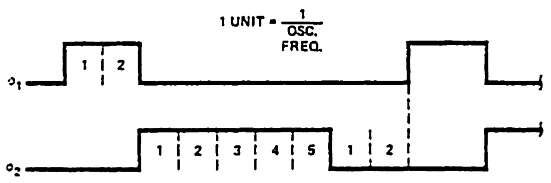
The Clock Generator consists of a synchronous "divide by nine" counter and the associated decode gating to create the waveforms of the two 8080A clocks and auxiliary timing signals.

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 ($\overline{\text{STSTB}}$) 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)

STSTB (Status Strobe)

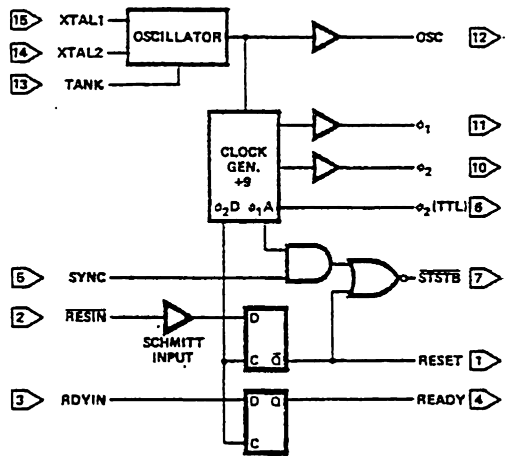
At the beginning of each machine cycle the 8080A CPU issues status information on its data bus. This information tells what type of action will take place during that machine cycle. By bringing in the SYNC signal from the CPU, and gating it with an internal timing signal (ϕ1A), an active low strobe can 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 STSTB signal connects directly to the 8228 System Controller.

The power-on Reset also generates 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

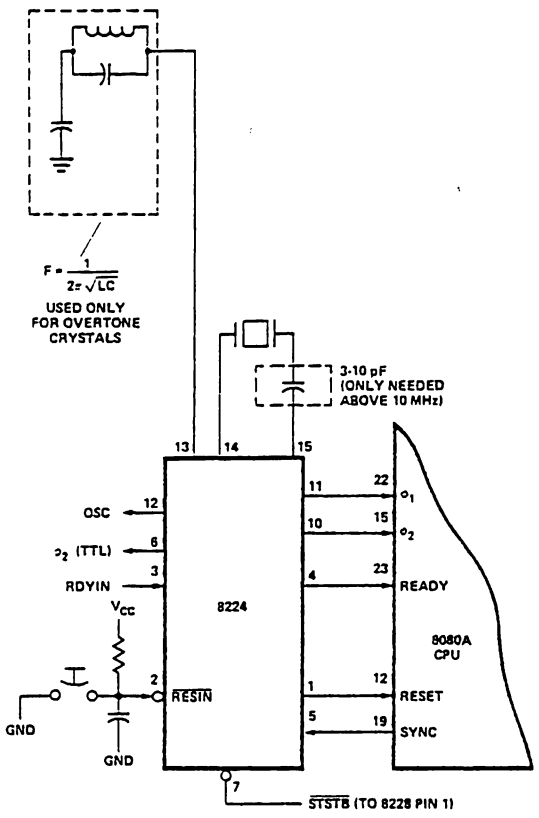
A common function in 8080A Microcomputer systems is the generation of an automatic system reset and start-up upon initial power-on. The 8224 has a built in feature to accomplish this feature.

An external RC network is connected to the 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 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

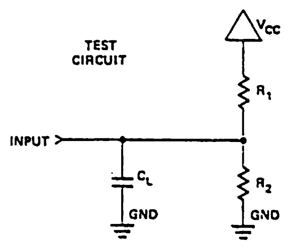
Tolerance: .005% at 0°C -70°C
Resonance: Series (Fundamental)*
Load Capacitance: 20-35pF
Equivalent Resistance: 75-20 ohms
Power Dissipation (Min): 4mW

*With tank circuit use 3rd overtone mode.

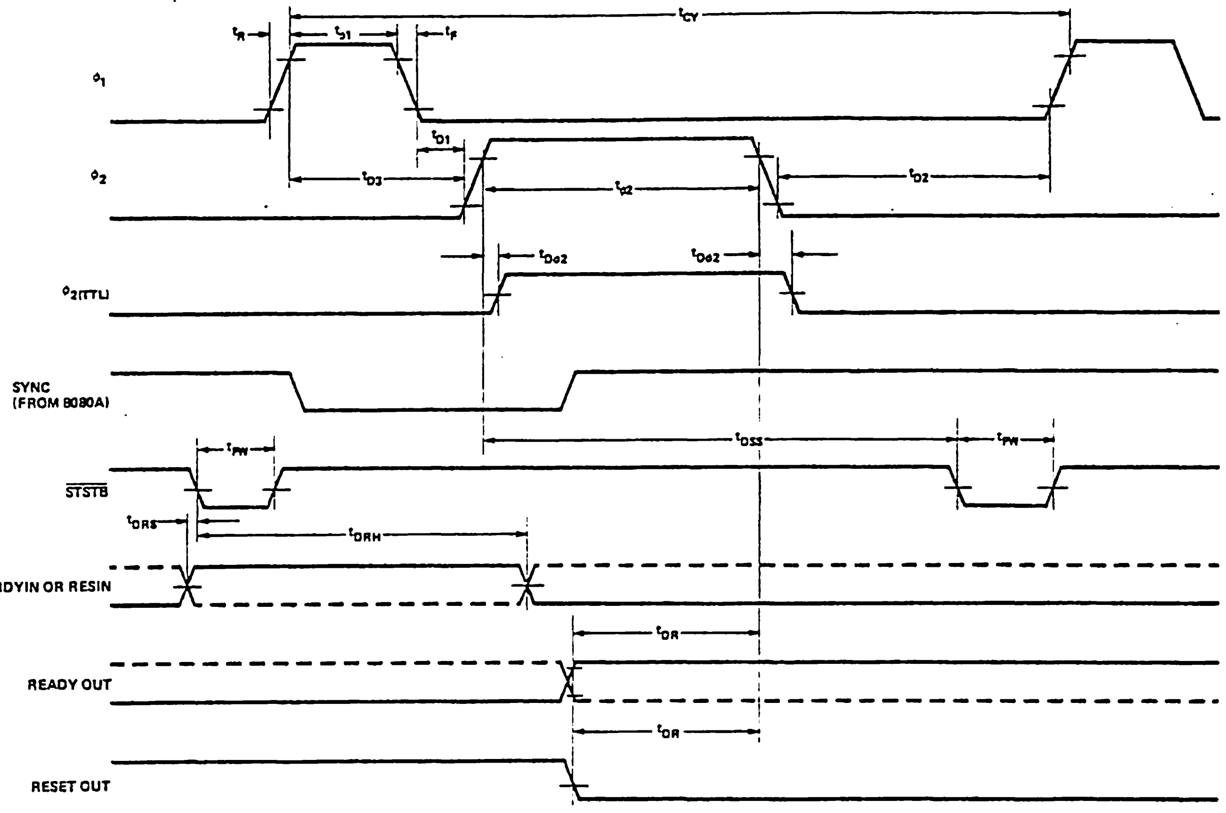
A.C. Characteristics

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		$\frac{6tcy}{9}$		
t_{PW}	$\overline{STSTB}$ Pulse Width	$\frac{tcy}{9}$ - 15ns				$\overline{STSTB}$,CL=15pF R_1 = 2K R_2 = 4K
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			Units	Test Conditions
		Min.	Typ.	Max.		
$t_{\phi 1}$	ϕ_1 Pulse Width	89			ns	t_{CY}=488.28ns
$t_{\phi 2}$	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	
$t_{D\phi 2}$	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	

3-4. SCHEMATIC REFERENCING

The detailed schematics of the Interface circuit, CPU circuit, and Display/Control panel are provided to aid in determining signal direction and tracing.. A solid arrow (⟶) on the signal line indicates direction, and the tracing of the signal through the schematics is referenced as it leaves the page. The reference is shown as a number - letter number (e.g. 2-A3), indicating sheet 2 and schematic zone A3. The reference may be shown alone or in a bracket. If the reference is bracketed, the signal is going to another schematic which is referenced outside the bracket. If the reference is shown alone, the signal is going to another page of the multisheet schematic.

3-5. 8800b BLOCK DIAGRAM DESCRIPTION (Figure 3-1)

The 8800b computer contains four basic circuits; the Central Processing Unit (CPU), Memory, an Input/Output (I/O) section, and the Front Panel. The CPU controls the interpretation and execution of software instructions, and Memory stores the software information to be used by the CPU. The I/O section provides a communication link between the CPU and external devices. The Front Panel allows the operator to manually perform various operations with the 8800b. The 8800b basic block diagram and accompanying text (paragraphs 3-6 and 3-7) explain the CPU's communication with the memory (and I/O) circuits and with the front panel. The system clock, power-on operation and run operation are explained in paragraphs 3-8 through 3-10.

3-6. CPU TO MEMORY OR I/O OPERATION

The Memory or I/O section operation repuires several signals that allow transfer of data to and from the CPU. The ADDRESS bus (AØ-A15) consists of sixteen individual lines from the CPU to Memory and I/O devices. The signals on this bus represent a particular

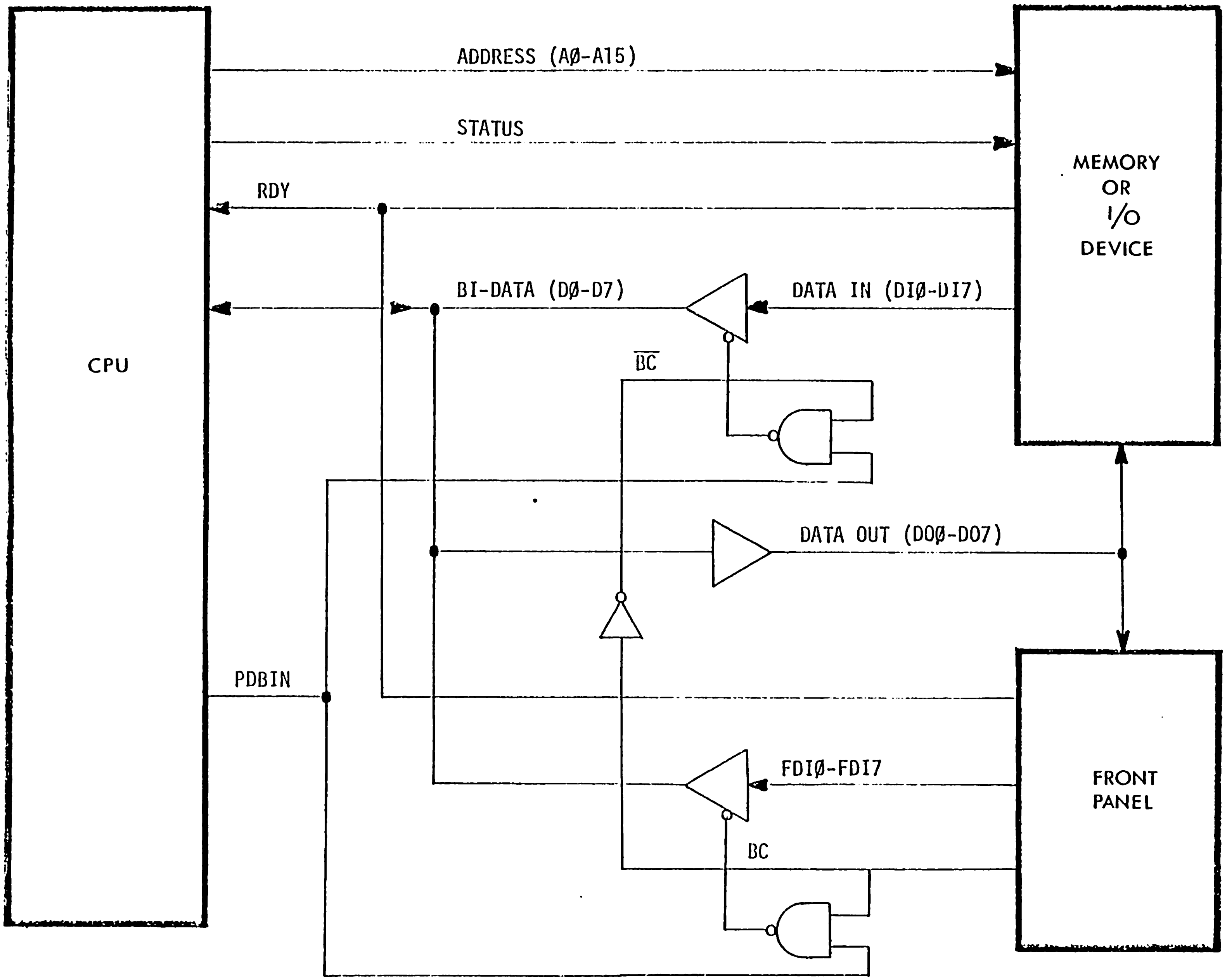


Figure 3-1. 8800b Basic Block.

memory address location or external device number that is needed to establish communications with Memory or I/O devices. Once the address data (AØ-A15) is presented to Memory or I/O devices, the CPU generates various STATUS signals. The STATUS signals enable decoding of a memory address or conditions the I/O device card to send or receive data from the CPU

Data from Memory or I/O devices is presented on the DATA IN lines (DIØ-DI7) and applied to eight non-inverting bus drivers. The drivers are enabled by a PDBIN signal from the CPU and a $\overline{\text{BC}}$ (bus control) signal. The BC signal is LOW when the Front Panel is not in operation. The eight non-inverting bus drivers, when enabled, present the input data to BI-DATA lines (DØ-D7) which input the data to the CPU.

Data outputted to Memory or I/O devices is presented to the DATA OUT lines (DOØ-DO7) from the CPU. The RDY (ready) line either forces the CPU to a wait state while data is being transferred or allows the CPU to process data.

3-7. FRONT PANEL OPERATION

The Front Panel Operation is very similar to Memory or I/O section operation. The Front Panel gains control of the CPU by producing a HIGH BC signal. The BC signal disables the DATA IN (DIØ-DI7) lines from a Memory or I/O Device and enables the FDIØ-FDI7 lines. The FDIØ-FDI7 lines contain Front Panel data which is transferred to the CPU upon the occurence of the PDBIN signal. All data from the CPU to the Front Panel is applied to the DATA OUT (DOØ-DO7) lines and displayed on the Front Panel.

3-8. SYSTEM CLOCK

The system clock (F) for the 8800b is located on the CPU circuit card (Figure 3-14, zone B7). The system clock generates phase 1 and phase 2 outputs derived from the external crystal (XTAL 1). The Ø1 and Ø2 outputs operate at a frequency of 2 MHz, which determines the speed at which the 8080 (M) will operate. The Ø1 and Ø2 clock signals are presented to the bus (zone A7) through inverter A and inverter bus driver J, respectively. The Ø1 clock is used by memory and external I/O cards, and the Ø2 clock is applied to the 24-bit counter on the Display/Control card (Figure

3-16, sheet 1, zone D2) through the Interface card (Figure 3-15, sheet 2, zone B3).

3-9. POWER ON CLEAR OPERATION

Positioning the ON/OFF switch to ON causes a power on clear (POC) operation to be performed, resetting the 8800b circuitry. The POC signal is generated on the CPU card (Figure 3-14, zone A3) when VCC is applied. With VCC present, capacitor C4 will charge to the VCC potential in 100 milliseconds because of the RC time constant of C4 and resistor R17. The 100 millisecond delay disables (turns off) transistor Q3, producing a LOW $\overline{\text{POC}}$ signal to the bus (pin 99) through inverters S and J (zone A2). The $\overline{\text{POC}}$ signal is inverted by U on the Interface card (Figure 3-15, sheet 2, zone B2) and presented to the Display/Control card as a HIGH POC signal (Figure 3-16, sheet 2, zone D6). The POC input is inverted LOW by T1 (zone C6) and applied to three circuits on the Display/Control Card. It clears the M1 flip-flops (zone C7) through NOR gate T1 and inverter J1 (zone C6), insuring that single step operation is disabled. It presets the M1 flip-flop (zone C9) and disables NAND gate P1 (zone B8) to insure that the 8800b is not running. The $\overline{\text{POC}}$ signal (zone D9) is also present at NOR gate R1 which inverts it HIGH to reset the PROM counter. The $\overline{\text{POC}}$ signal is present to the external input/output (I/O) cards and memory for similar initialization operations. During the POC operation, two other functions are being performed.

On the Display/Control card (Figure 3-16, sheet 1, zone D2), a 24-bit counter is being clocked by Ø2 which will condition circuits on the Display/Control card. The $\overline{\text{C13}}$ output (zone D1) from the counter is applied to the clock (CK) input of quad latches C1, F1, H1, G1, N1, U1, Y1, and W1 (zones B9-B1) through non-inverting bus driver K1 (zones A1 and D1) and inverter J1 (zone C1). The $\overline{\text{C13}}$ signal clears the quad latches in the following manner to insure all latches are conditioned after POC. The inputs to quad latches C1, F1, H1, and G1 are HIGH because no switches are activated. After the first $\overline{\text{CT3}}$ clock, all the $\overline{\text{Q}}$ outputs are LOW and applied to the inputs of quad latches N1, U1, Y1, and W1 (zones B9-B1).

The occurrence of the next $\overline{\text{C13}}$ clock latches the Q outputs LOW and the $\overline{\text{Q}}$ outputs HIGH during the POC operation.

When VCC is present in the CPU circuits, another RC time constant affects the clock generator F (Figure 3-14, zone B7). Capacitor C2 will charge to the VCC potential in 33 microseconds which is the time constant of C2 and resistor R10. The 33 microsecond delay allows the RESET output from F (zone B7) to clear the 8080 M internal circuits. The 8080 remains in this state because the READY output (zone B7) is LOW from F. The READY output from F will be affected during the run operation.

3-10. RUN OPERATION

The Run Operation allows the 8080 on the CPU Board to start processing data to and from memory and external devices. The Run Operation is activated when the RUN/STOP switch on the 8800b front panel is momentarily depressed to RUN.

The RUN/STOP circuits are located on the Display/Control card (Figure 3-16, sheet 2, zone A9). When the RUN/STOP switch is momentarily depressed, a LOW is applied to quad latch C1, input D2. The occurrence of the next C13 clock (zone A1) causes the $\overline{\text{Q}}$ output at pin 6 of C1 (zone B9) to go HIGH. This HIGH is applied to quad latch N1, input D2. The next C13 clock causes the Q output at pin 2 of N1 (zone B9) to go HIGH and allows NAND gate P1 to clear M1 (zone C9). The Q output of M1 generates a LOW $\overline{\text{RUN}}$ signal and LOW $\overline{\text{FRDY}}$ signal through NOR gate P1 and inverter R1 (zone D9).

The $\overline{\text{RUN}}$ signal is applied to the Interface Card (Figure 3-15, sheet 2, zone D2) to condition the MD input of data latch G (sheet 3, zone A6). With MD enabled, output data from the CPU can be displayed on the 8800b front panel if a STB input is present to G (discussed in Paragraph 3-40).

The $\overline{\text{FRDY}}$ signal is applied to the Interface Card (Figure 3-15, sheet 2) to allow the 8080 to start processing data. The FRDY output is applied to pin 58 of the bus through inverter R and non-inverting bus driver H as a HIGH (zone A1). The HIGH on pin 58 of the bus enables NAND gate C, pin 8, LOW on the CPU (Figure 3-14, zone A7) which is inverted HIGH by B (zone B7) and applied to the clock generator F RYDIN input. The RYDIN signal enables the READY output at F HIGH (zone B7) which allows the 8080 M (zone A8) to start processing data.

3-11. 8800b DATA PROCESSING OPERATION

The 8800b data processing begins when the 8080 IC is enabled (Paragraph 3-10). With the 8080 IC enabled, the program (P) counter in the 8080 starts to increment or begins at a predetermined count established by the operator. The count in the P counter represents a location in memory which is examined by the CPU before the P counter increments to the next location. To examine each memory location, the CPU initiates an instruction cycle operation. Every instruction cycle consists of one, two, three, four, or five machine cycles. In order to perform a data processing operation, basic machine cycles are required.

The Instruction Fetch Machine cycle is a basic machine cycle needed to allow the CPU to fetch an instruction from memory. A memory read machine cycle is also a basic machine cycle that enables the CPU to communicate with a memory or external device for data transfer operations.

The following paragraphs discuss data transfers from an external device to the CPU, from the CPU to memory, from memory to the CPU, and from the CPU to an external device. However, the instruction fetch and memory read machine cycles used in the data transfers are discussed first because their operation is identical in all of the data transfers. It is important to note that there are many variations of data transfer which are dependent on the programmer.

3-12. INSTRUCTION FETCH CYCLE

The Instruction Fetch Cycle is the first machine cycle (M1) to be performed by the CPU in any data transfer operation. The memory location specified by the P counter contains data that the CPU interprets as an instruction. The first cycle must be a fetch cycle because, during the fetch cycle, the CPU is informed as to what operation will be performed next.

3-13. INSTRUCTION FETCH CYCLE OPERATION (Figure 3-2)

The Instruction Fetch Cycle is initiated whenever the P counter is incremented to a new memory address location (e.g. 000 100_8) where an instruction (e.g. 072_8) is stored. In order to fetch the 072_8 data from memory during machine cycle one, several signals are generated by the CPU.

A PSYNC output from the CPU is applied to memory to condition for address decoding. Next the ADDRESS (000 100_8), consisting of sixteen parallel outputs (AØ-A15) from the CPU, is presented to the Display/Control Card and memory. The AØ through A15 signals drive the appropriate address buffers, illuminating the light emitting diodes (LEDs) on the Display/Control Card. The ADDRESS and PSYNC signals present at the memory from the CPU initiate decoding of the memory address (000 100_8).

The CPU then generates three signals, SM1, SMEMR, and Ø1 CLOCK to complete the Instruction Fetch Cycle. The SM1 output is applied to the Display/Control Card through the Interface Card to light the M1 (machine cycle 1) LED on the 8800b front panel. The SMEMR and Ø1 CLOCK outputs are applied to memory to allow decoding of the memory address (000 100_8). With the memory address decoded, the 072_8 data present in that location is transferred to the CPU on the eight DATA IN (DIØ-DI7) lines. The DIG 1 input to the CPU from the Interface Card is enabled when the 8800b is in the run mode (see paragraph 3-10). This permits the memory data to be transferred to the CPU. The SMEMR output is applied to the Display/Control Card through the Interface Card to light the MEMR (memory read) LED on the 8800b front panel. This operation is performed when the P counter is incremented, indicating a new memory address.

3-14. INSTRUCTION FETCH CYCLE DETAILED OPERATION

The following paragraphs describe the Instruction Fetch Cycle operation in detail. Refer to Figure 3-3, Instruction Fetch Cycle Timing, during the explanation. The Instruction Fetch Cycle operation (M1) requires four Ø1 and Ø2 clock pulses. Each clock period performs a particular operation as described in the following paragraphs.

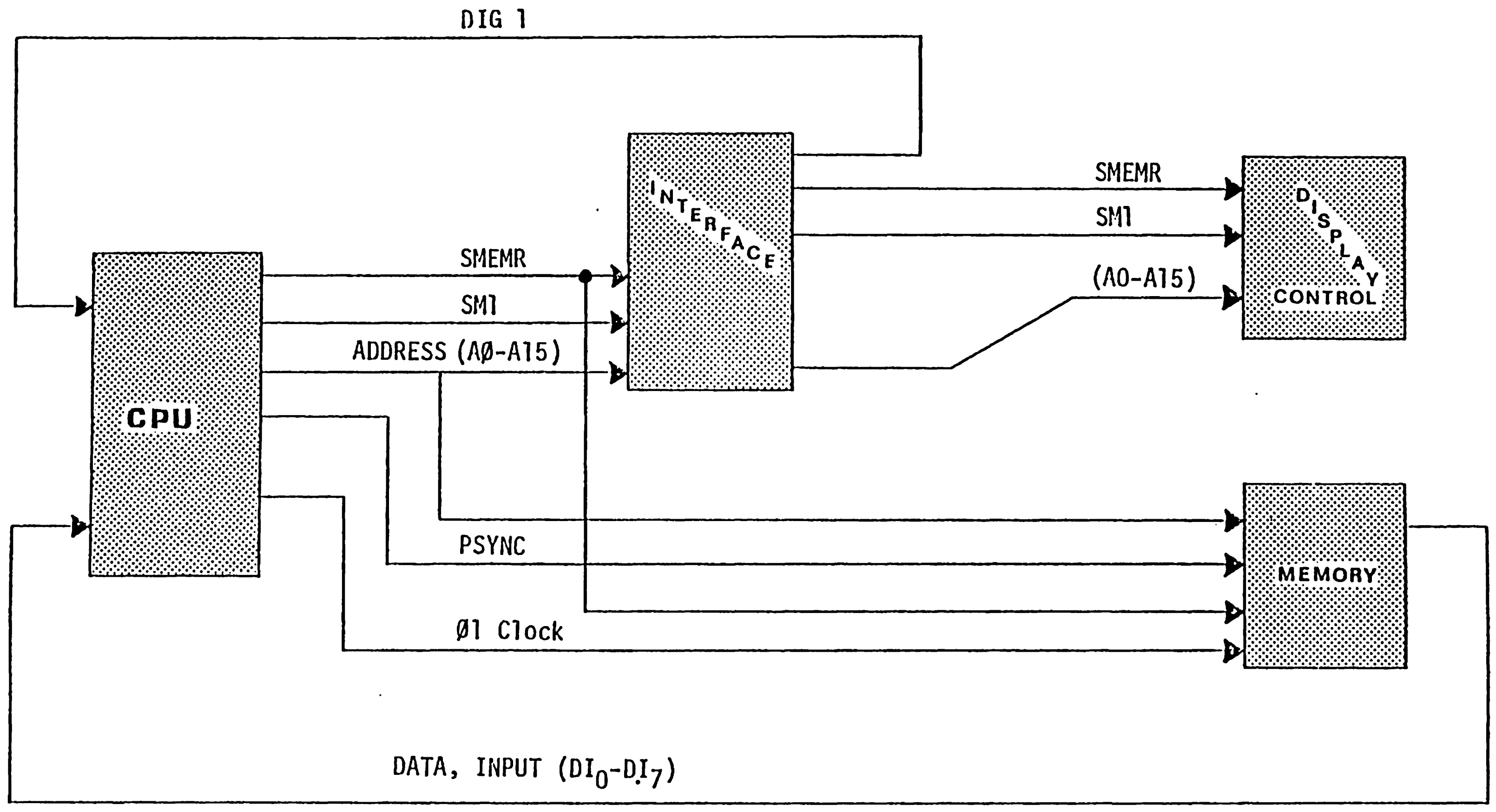


Figure 3-2. Instruction Fetch Cycle Block Digaram

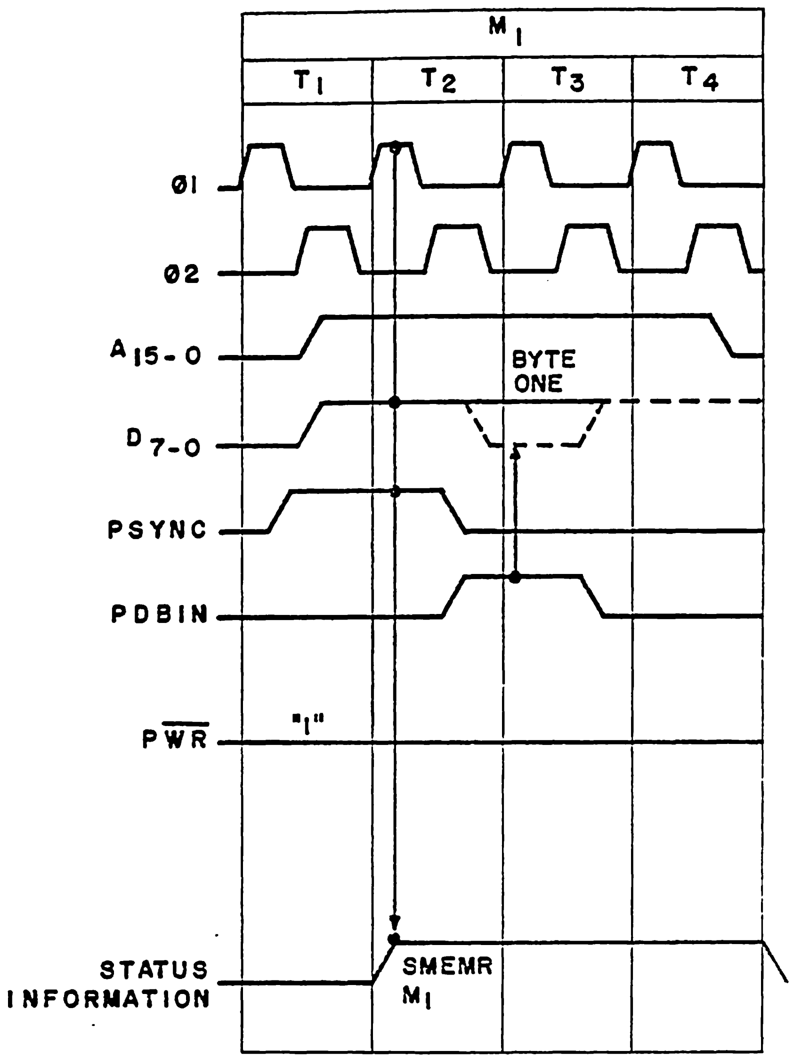


Figure 3-3. Instruction Fetch Cycle Timing.

During the latter portion of T1, several outputs are generated by the CPU (M) (Figure 3-14): address data AØ through A15 (zone B8), status data DØ through D7, and a SYNC signal (zone C8). The AØ through A15 data is applied to memory via the bus through non-inverting bus drivers, U, P, and N (zone B9) on the CPU. The address data (AØ-A15) is also applied through inverters P, N, and X on the Interface card (Figure 3-15, sheet 1, zone B5) and presented to the Display/Control card. The AØ through A15 signals present on the Display/Control card light the appropriate AØ through A15 LEDs, indicating the memory address.
The DØ through D7 data is applied to K (zone B5) on the CPU through the bi-directional circuits D and E. The status data is enabled through D and E at this time because $\overline{CS}$ and $\overline{DIEN}$ are LOW. The SYNC output is applied to the clock generator F (zone B7) and memory as PSYNC via pin 76 (zone D1) on the bus through the non-inverting bus driver V (zone D8). The PSYNC signal conditions memory to decode the address data. The SYNC input at F will enable a signal during T2.

During the beginning of T2, a low $\overline{STSTB}$ (zone B7) is generated from F as a result of the HIGH SYNC input and internal timing of F. The $\overline{STSTB}$ is applied to the data latch K (zone B5), allowing the status data DØ through D7 to be stored in K. The status data present at the output of K conditions the memory to fetch the instruction (072_8) from its addressed memory location (e.g. $000\ 100_8$) by enabling the following signals.

A SM1 and SMEMR HIGH output from K is presented on pins 44 and 47 of the bus (zone A5) through non-inverting bus drivers X and R. The SM1 and SMEMR signals are applied through inverter V on the Interface card (Figure 3-15, sheet 2, zone B5) and presented to the Display/Control card as $\overline{SM1}$ and $\overline{SMEMR}$. The $\overline{SM1}$ and $\overline{SMEMR}$ signals present on the Display/Control card light the M1 and MEMR LEDs (Figure 3-16, sheet 3, zone C3) on the front panel of the 8800b, indicating machine cycle one is performing a memory read operation. The SMEMR output from the CPU (Figure 3-14, zone A5) is applied to memory, initiating a data transfer to the CPU during T3.

At the beginning of T3, the instruction (072_8) data is transferred from memory to M on the CPU. The memory data (DIØ through DI7) is supplied to the CPU card (Figure 3-14, zone B1) from the bus. The data is presented to M through bi-directional gates D and E (zone C7), inverter bus drivers L and J (zone B4), and inverters Y and S (zone B3) by the DBIN signal.

At the latter portion of T2 and the beginning of T3, a high DBIN output (zone C8) is generated by M. The DBIN output is applied to the $\overline{\text{DIEN}}$ inputs (zone C7) of D and E and pin 4 of NAND gate C (zone B4) as PDBIN. This signal enables pin 6 of NAND gate C LOW (DIG1 is high when the front panel is not used). This allows data input from memory (DIØ-DI7) to be enabled through inverting bus drivers L and J (zone B4) and applied through bi-directional gates D and E to M (zone C7).

Clock period T4 of machine cycle one allows for 8080 processing of the received instruction data from memory. If the instruction data present in the CPU requires a data transfer to or from an external device, a memory read cycle (M2) is initiated. However, if the instruction data present in the CPU requires a data transfer to or from memory, two memory read cycles (M2 and M3) are initiated.

3-15. MEMORY READ CYCLE

The Memory Read Cycle (M2) follows the Instruction Fetch Cycle (M1). During a Memory Read Cycle, an address is transferred to the CPU from memory. This address is either an external device number or a memory location (depending upon the instructions received during M1).

3-16. MEMORY READ CYCLE OPERATION (Figure 3-4)

The CPU performs one or two Memory Read Cycle operations. If the CPU is to communicate with an external device, one Memory Read Cycle is required because the external device number consists of 8 data

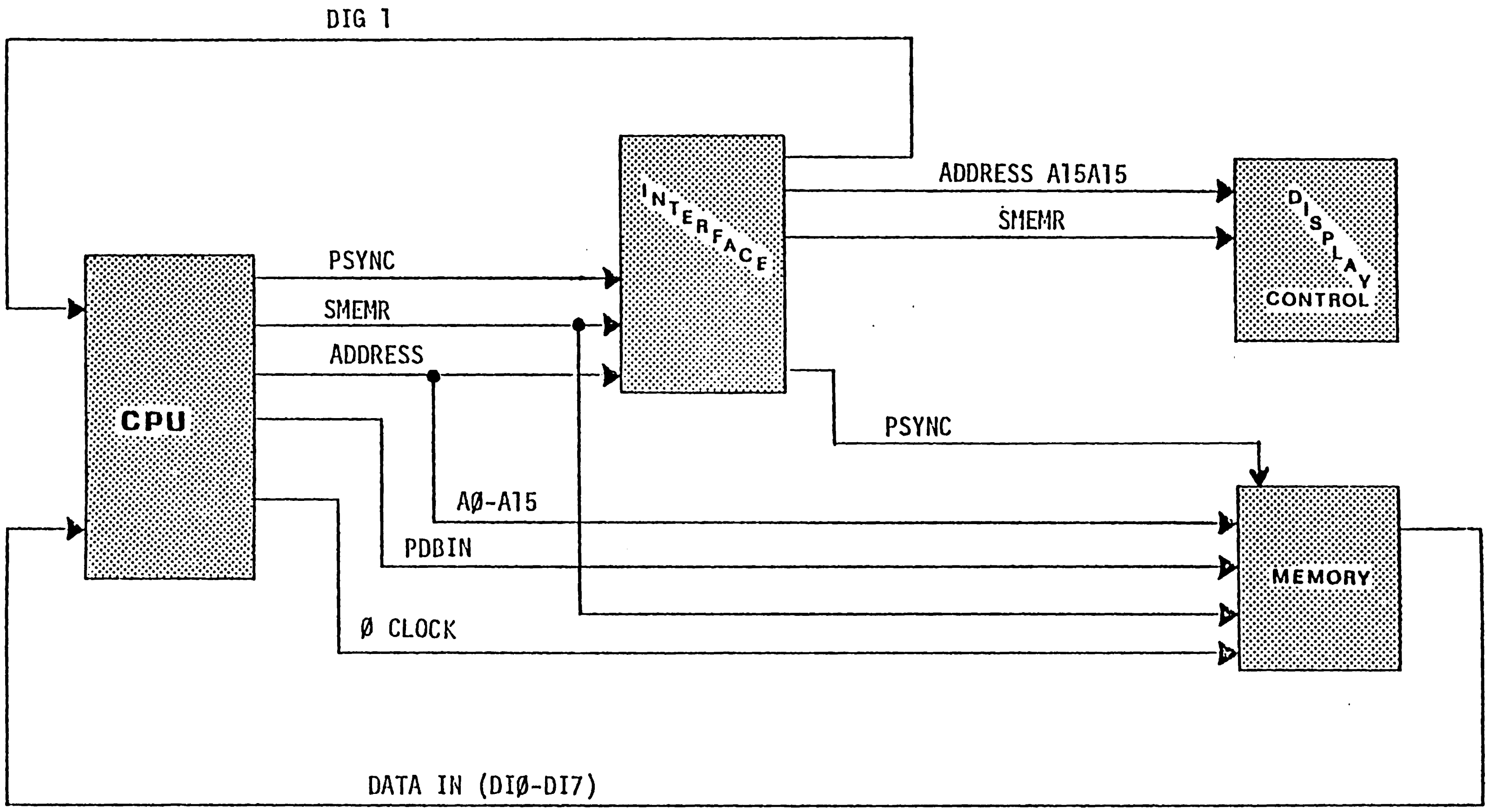


Figure 3-4. Memory Read Cycle Block Diagram

bits (1 byte). However, if the CPU is instructed to communicate with memory, two Memory Read Cycles are required because the memory address consists of 16 data bits (2 bytes).

The two Memory Read Cycles obtain the memory address (e.g. 000 200_8) that is required by the CPU to complete the instruction. Since one byte (8 bits) of the two byte address is transferred during one Memory Read Cycle, two cycles are required. The first Memory Read Cycle obtains the least significant bits (LSBs) of the address (200_8) from memory and stores them in the CPU. The second cycle obtains the most significant bits (MSBs) of the address (000_8) from memory and stores them in the CPU.

The Memory Read Cycles are very similar to the Instruction Fetch Cycle. They require a memory address location (e.g. 000 101_8 and 000 102_8) that indicates where the LSBs and MSBs of the address (000 200_8) are stored. After completion of the Instruction Fetch Cycle, the program counter in the CPU is incremented to 000 101_8 and the first Memory Read Cycle is initiated. Several signals are generated by the CPU in order to read the LSBs of the address (200_8) from memory.

A PSYNC output from the CPU is applied to memory through the Interface Card to condition the memory for address decoding. Next the ADDRESS (000 101_8), consisting of sixteen parallel outputs (AØ-A15) from the CPU, is presented to the Display/Control Card and memory. The AØ through A15 signals light the appropriate address light emitting diodes (LEDs) on the Display/Control Card. The ADDRESS and PSYNC signals present at the memory from the CPU initiate decoding of the address (000 101_8).

The CPU then generates three signals, SMEMR, PDBIN, and Ø1 to complete the Memory Read Cycle. The SMEMR, PDBIN, and Ø1 outputs are presented to memory to enable decoding of the address (000 101_8). With the address decoded, the 200_8 data present in that location is transferred to the CPU on the eight DATA IN (DIØ-DI7) lines. The DIGl input to the CPU from the Interface Card is enabled when the 8800b is in the run mode, permitting memory data to be transferred to the CPU.

The SMEMR output is presented to the Display/Control Card through the Interface Card to light the MEMR (memory read) LED on the 8800b front panel. The second Memory Read Cycle operation is identical to the first. It transfers the MSBs of the address (000_8) to the CPU.

3-17. MEMORY READ CYCLE DETAILED OPERATION

The following paragraphs describe the Memory Read Cycle operation in detail. Refer to Figure 3-5, Memory Read Cycle Timing, during the explanation.

The two Memory Read Cycle operations (M2 and M3) obtain the memory address (e.g. 000 200_8) required by the CPU to complete an instruction. As stated previously, the LSBs of the address (200_8) are transferred to the CPU during M2, and the MSBs of the address (000_8) are transferred to the CPU during M3. There are three clock periods (T1-T3) required for each Memory Read Cycle operation.

During the latter portion of T1, several outputs are generated by the CPU (Figure 3-14); Address data AØ through A15 (zone B8), status data DØ through D7, and a SYNC signal (zone C8). The AØ through A15 data is presented to memory and the 8800b front panel via the bus through non-inverting bus drivers U, P, and N (zone B9) on the CPU. The DØ through D7 data is applied to K (zone B5) on the CPU through the bi-directional circuits D and E. The status data is enabled through D and E at this time because $\overline{\text{CS}}$ and $\overline{\text{DIEN}}$ are LOW. The SYNC output is applied to the clock generator F (zone B7) and memory as PSYNC via pin 76 (zone D1) on the bus through non-inverting bus driver V (zone D8). The PSYNC signal conditions memory to decode the address data.

During the beginning of T2, a $\overline{\text{STSTB}}$ (zone B7) is generated (LOW) from F as a result of the HIGH SYNC input and internal timing of F. The $\overline{\text{STSTB}}$ is applied to the data latch K (zone B5), allowing the status data DØ through D7 to be stored in K. The status data present at the output of K allows the CPU to read the LSBs of the memory address

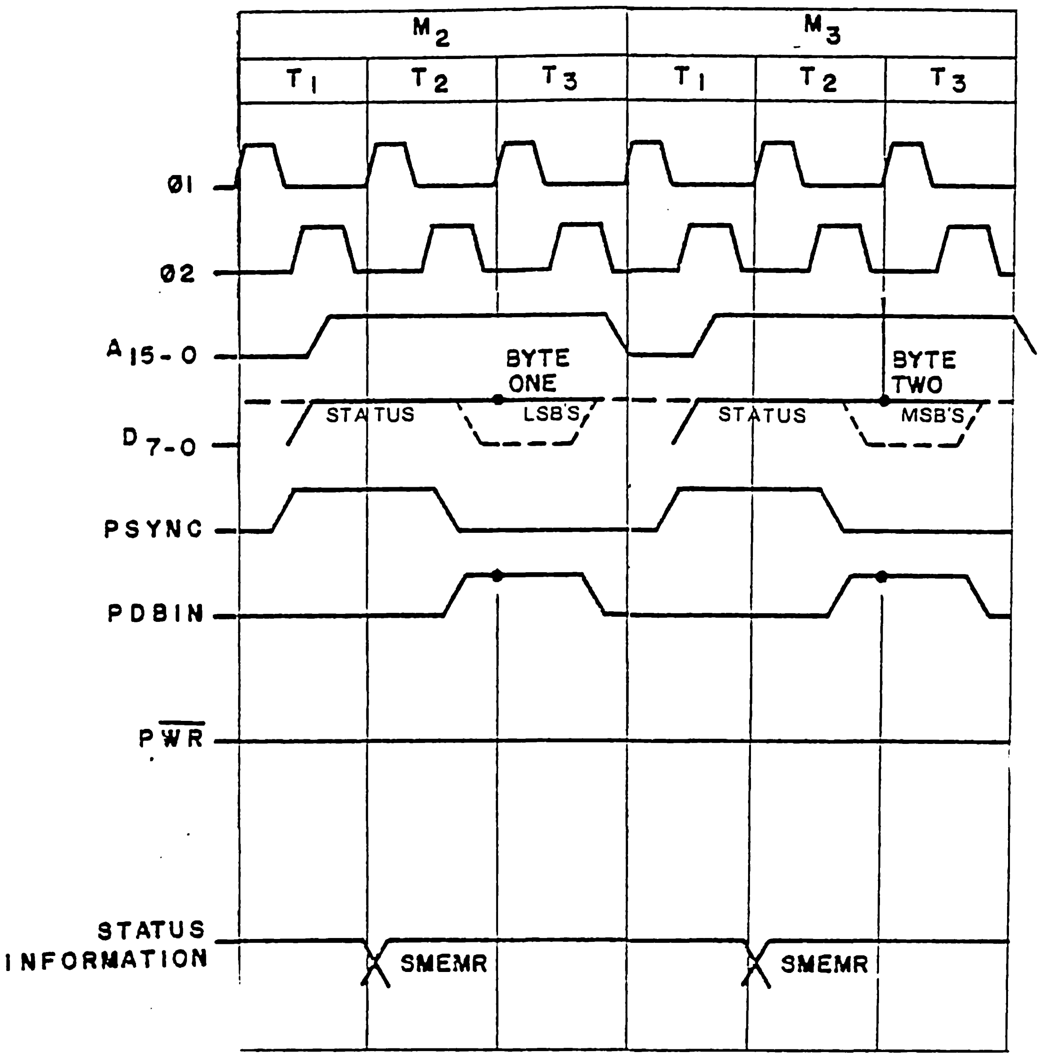


Figure 3-5. Memory Read Cycle Timing.

location (ex. 000 101_8) by enabling the SMEMR signal.

A SMEMR output (HIGH) from K is presented on pin 47 of the bus (zone A4) through non-inverting bus drivers X and R. The SMEMR signal is applied through inverter V on the Interface Card (Figure 3-15, sheet 2, zone B4) and presented to the Display/Control card as $\overline{\text{SMEMR}}$. The $\overline{\text{SMEMR}}$ signal present on the Display/Control card lights the MEMR LED (Figure 3-16, zone C3) on the front panel of the 8800b, indicating a memory read operation is occurring. The SMEMR output from the CPU (Figure 3-14, zone A5) is applied to memory in order to initiate a data transfer to the CPU during T3.

At the beginning of T3, the LSBs of the memory storage location (200_8) are transferred from memory to the 8080 (M) on the CPU. The memory data in (DIØ through DI7) is applied to the CPU card (Figure 3-14, zone B1) from the bus. The data is presented to M through bi-directional gates D and E (zone C7), inverter bus drivers L and J (zone B4), and inverters Y and S (zone B3) by the PDBIN signal.

At the latter portion of T2 and the beginning of T3, a DBIN output (zone C8) HIGH is generated by M. The DBIN output is applied to the $\overline{\text{DIEN}}$ inputs (zone C7) of D and E and pin 4 of NAND gate C (zone B4) as PDBIN. This signal enables pin 6 of NAND gate C LOW (DIG 1 is high when front panel is not used). This allows the data in from memory (DIØ - DI7) to be enabled through inverting bus drivers L and J (zone B4) and applied through bi-directional gates D and E to M (zone C7). The second Memory Read Cycle operation (M3) transfers the contents of memory address (000 102_8) which contain the MSBs of the memory address number to the CPU. It is important to note that only one Memory Read Cycle operation is required if the CPU is to communicate with an external device.

3-18. EXTERNAL DEVICE TO CPU DATA TRANSFER

An External Device to CPU data transfer is accomplished when an input instruction (333_8) is fetched from a memory location during M1, and the external device number (XXX_8) is read from a memory location during M2 by the CPU. The data from the external device is transferred to the CPU by an Input Read Cycle operation (M3).

3-19. INPUT READ CYCLE OPERATION (Figure 3-6)

The Input Read Cycle operation will allow the CPU to obtain data from an external device. After the completion of the Memory Read Cycle (M2), the program counter is not incremented until the completion of the Input Read Cycle. Several signals are generated by the CPU in order to obtain data from the external device.

The SINP output and external device ADDRESS (XXX_8) number, consisting of the first eight individual outputs (AØ-A7) from the CPU, is presented to the external device input/output channel, thereby enabling the I/O card. With the I/O enabled, a PDBIN signal from the CPU allows the I/O to transfer the external device data to the CPU on the eight DATA IN (DIØ-DI7) lines for storage. The DIG 1 input to the CPU from the Interface is enabled during the 8800b run mode and allows the external device data to be stored in the CPU. The SINP and AØ through A15 outputs are supplied to the Display/Control Card through the Interface Card to illuminate the INP (input) and ADDRESS LEDs on the 8800b front panel.

3-20. INPUT READ CYCLE DETAILED OPERATION

The following paragraphs describe the Input Read Cycle operation in detail. Refer to Figure 3-7, Input Read Cycle Timing, during the explanation. The Input Read Cycle operation (M3) requires three Ø1 and Ø2 clock pulses. During each clock period, a specific operation is performed as described in the following paragraphs.

During the latter portion of T1, several outputs are generated by the CPU (Figure 3-14); address data AØ through A15 (zone B8), status data DØ through D7, and a SYNC signal (zone C8). The AØ through A15 data contains the external device number (AØ-A7 and A8-A15 contain identical data) and is applied to the I/O card via the bus through non-inverting bus drivers U, P, and N (zone B9) on the CPU in order to enable the I/O card. The address data (AØ-A15) is also applied through inverters P, W, and X on the Interface Card (Figure 3-15, sheet 1, zone B5) and

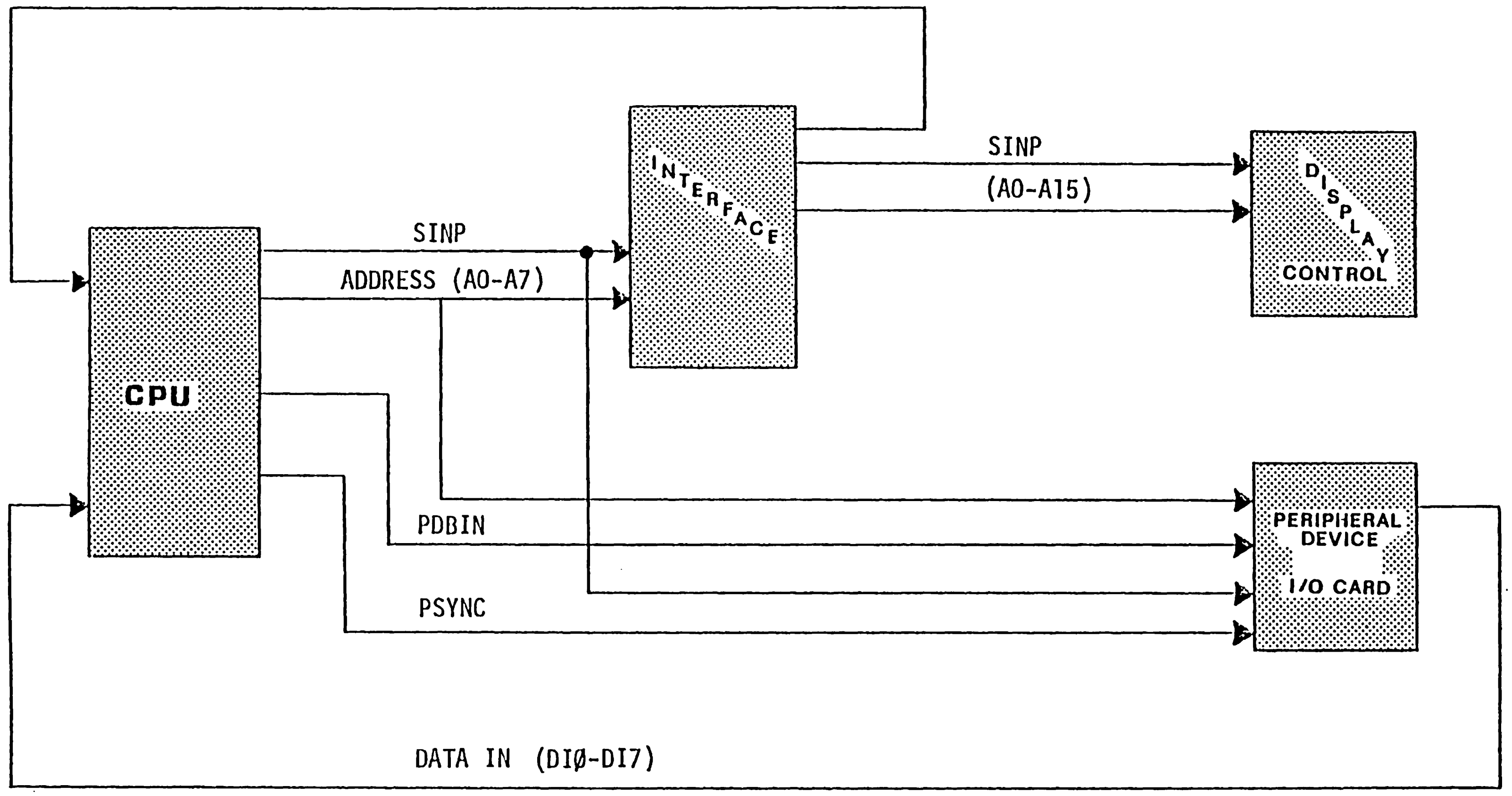


Figure 3-6. Input Read Cycle Block Diagram

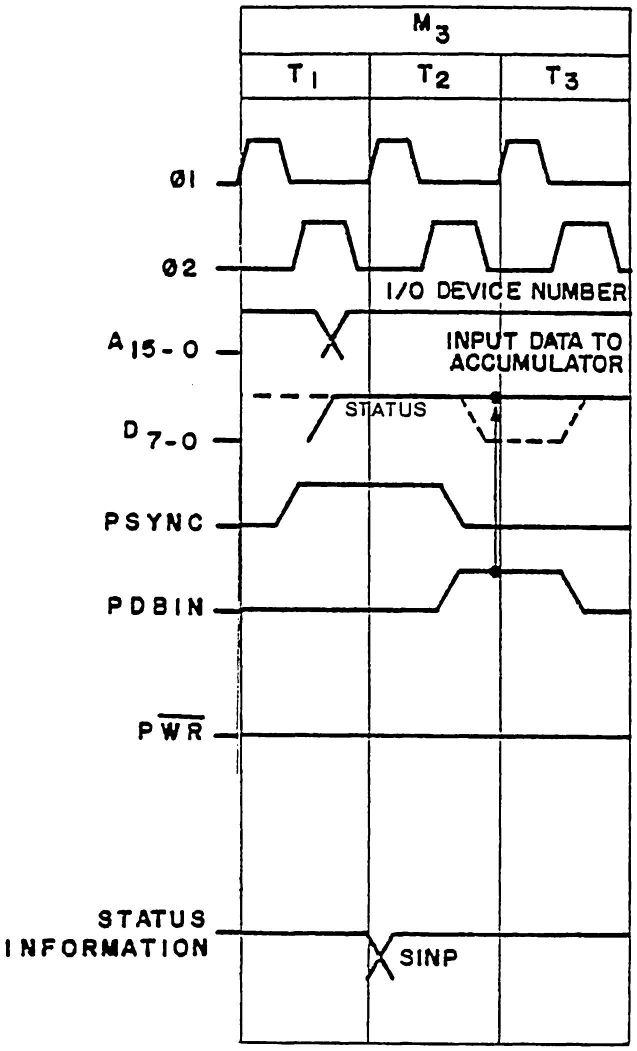


Figure 3-7. Input Read Cycle Timing.

presented to the Display/Control card. The AØ through A15 signals present on the Display/Control card (Figure 3-16, sheet 3, zone A9-A4) light the appropriate AØ through A15 LEDs, indicating the address of the external device. (Recall that when addressing an I/O device, the address is repeated on the upper eight and lower eight address LEDs.) The DØ through D7 data is applied to K (Figure 3-14, zone B5) on the CPU through the bi-directional circuits D and E. The status data is enabled through D and E at this time because $\overline{\text{CS}}$ and $\overline{\text{DIEN}}$ are LOW. The SYNC output is applied to the clock generator F (zone B7), conditioning F to generate a signal during T2.

At the beginning of T2, a $\overline{\text{STSTB}}$ (zone B7) is generated LOW from F as a result of the HIGH SYNC input and internal timing of F. The $\overline{\text{STSTB}}$ is applied to the data latch K (zone B5), allowing the status data DØ through D7 to be stored into K. The status data present at the output of K conditions the I/O card to send data to the CPU by enabling the SINP signal.

A SINP output from K is presented HIGH on pin 46 of the bus (zone A4) through non-inverting bus driver R. The SINP signal is applied through inverter V on the Interface Card (Figure 3-15, sheet 2, zone B5) and presented to the Display/Control card as $\overline{\text{SINP}}$. The $\overline{\text{SINP}}$ signal present on the Display/Control card lights the INP LED (Figure 3-16, sheet 3, zone C3) on the front panel of the 8800b, indicating data is being received from an external device. The SINP output from the CPU is applied to the external device I/O card in order to initiate a data transfer to the CPU during T3.

At the beginning of T3, the external device data is transferred to M on the CPU via the bus. The external device data in (DIØ through DI7) is applied to the CPU card (Figure 3-14, zone B1) from the bus. The data is presented to the 8080 (M) through bi-directional gates D and E (zone C7), inverter bus drivers L and J (zone B4), and inverters Y and S (zone B3) by the PDBIN signal.

At the latter portion of T2 and the beginning of T3, a DBIN output (zone C8) HIGH is generated by M. The DBIN output is applied to the $\overline{\text{DIEN}}$ inputs (zone C7) of D and E, pin 4 of NAND gate C (zone B4) and the bus pin 78 (zone D1) as PDBIN. This signal enables pin 6 of NAND gate C LOW (DIG 1 is HIGH when the front panel is not used), allowing the data input from the I/O card (DIØ-DI7) to be enabled through inverting bus drivers L and J (zone B4) and applied through bi-directional gates D and E to M (zone C7). The data at the external device is presented on the bus by the occurrence of PDBIN. After the external device data is stored in the CPU, the P counter is incremented, thus ending the Input Read Cycle operation.

3-21. CPU TO MEMORY DATA TRANSFER

A CPU to Memory data transfer is accomplished whenever an instruction is encountered to perform this operation. For example, a store accumulator STA (062_8) instruction requires the accumulator in the CPU to transfer its contents to memory. The STA instruction is fetched during M1 and its storage location determined in memory read cycles M2 and M3. The accumulator data is transferred to memory by a Memory Write Cycle operation (M4).

3-22. <u>MEMORY WRITE CYCLE BASIC OPERATION (Figure 3-8)</u>

The Memory Write Cycle operation will allow the CPU to transfer data to the memory. Several signals are generated by the CPU in order to transfer data to the memory.

The $\overline{\text{SWO}}$ output from the CPU is applied to the Display/Control through the Interface to light the WO (write out) LED on the 8800b front panel. The ADDRESS (XXX XXX_8), consisting of fifteen individual outputs (AØ-A15) from the CPU, is presented to the Display/Control and memory. The AØ through A15 signals light the appropriate address LEDs on the Display/Control. The ADDRESS and PSYNC signals present at the memory from the CPU can also initiate decoding of the memory address. With the memory conditioned, eight DATA OUT lines (DOØ-DO7) transfer the CPU data to the memory for storage. The $\overline{\text{PWR}}$ and $\overline{\text{SOUT}}$ outputs from the CPU are applied to the Interface to produce a MWRITE signal which allows the memory to store the data.

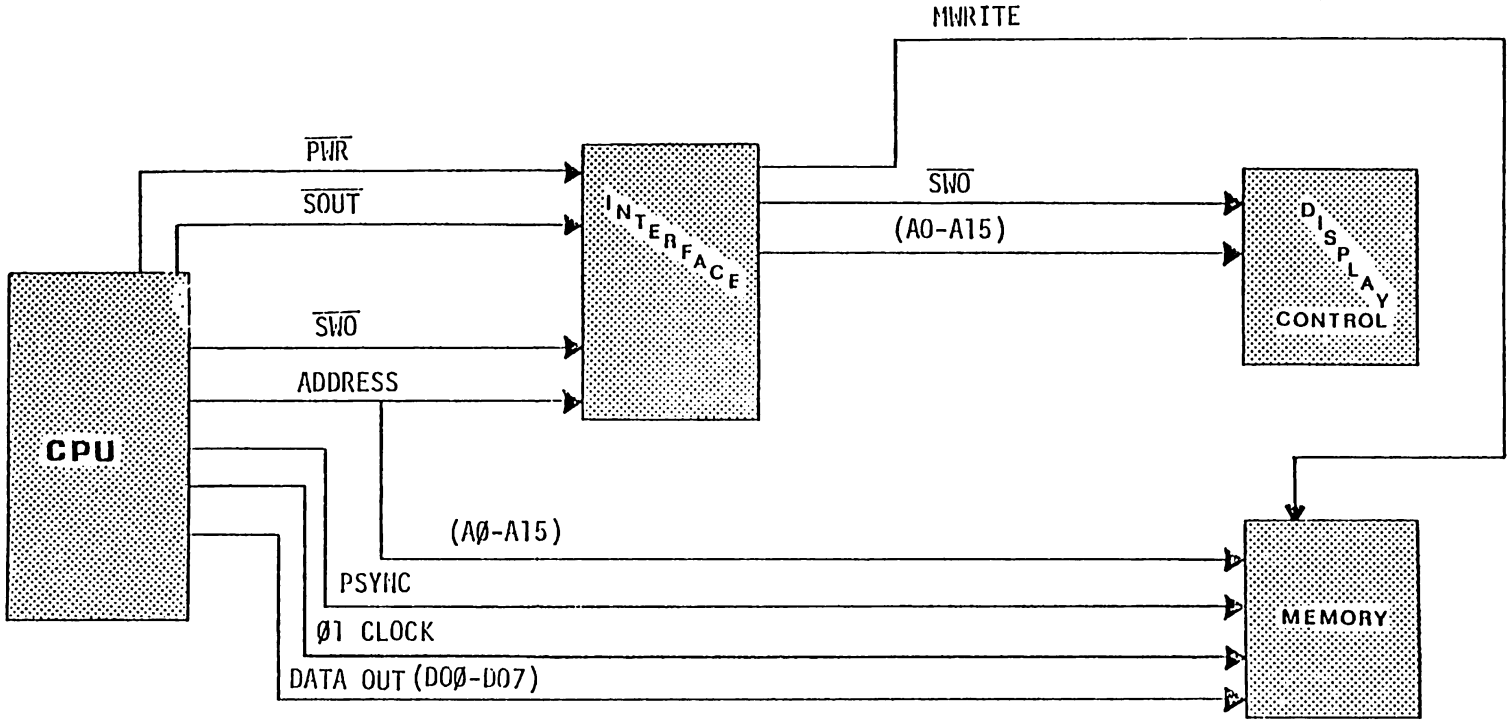


Figure 3-8. Memory Write Cycle Block Diagram

3-23. MEMORY WRITE CYCLE DETAILED OPERATION

The following paragraphs describe the Memory Write Cycle operation in detail. Refer to Figure 3-9, Memory Write Cycle Timing, during the explanation. The Memory Write Cycle operation (M4) requires three Ø1 and Ø2 clock pulses. Each period performs a certain operation as described in the following paragraphs.

During the latter portion of T1, several outputs are generated by the CPU 8080 IC (Figure 3-14); Address data AØ through A15 (zone B8), status data DØ through D7, and a SYNC signal (zone C8). The AØ through A15 data contains the memory storage location address (ex. 000 200_8) which is applied to the memory card via the bus through non-inverting bus drivers U, P, and N (zone B9) on the CPU in order to enable the memory. The address data (AØ-A15) is also applied through inverters P, W, and X on the Interface Card (Figure 3-15, sheet 1, zone B5) and presented to the Display/Control card. The AØ through A15 signals present on the Display/Control card (Figure 3-16, sheet 3, zones A9-A5) light the appropriate AØ through A15 LEDs, indicating the memory location address. The DØ through D7 data is applied to K on the CPU (Figure 3-14, zone B5) through the bi-directional circuits D and E. The status data is enabled through D and E at this time because $\overline{\text{CS}}$ and $\overline{\text{DIEN}}$ are LOW. The SYNC output is applied to the clock generator F (zone B7), conditioning F to generate a signal during T2.

During the beginning of T2, a LOW $\overline{\text{STSTB}}$ (zone B7) is generated from F as a result of the HIGH SYNC input and internal timing of F. The $\overline{\text{STSTB}}$ is applied to the data latch K (zone B5) allowing the status data DØ through D7 to be stored into K. The status data present at the output of K indicates a write output operation is being performed. However, the distinction of whether the data from the CPU is being transferred to a memory or an external device is determined by the status of the SOUT signal (zone A5). During a Memory Write Cycle, the SOUT signal is LOW and applied to the Interface Card (Figure 3-15, sheet 2). The SOUT signal is inverted HIGH by V and applied to pin 2 of NAND gate A (zone C3).

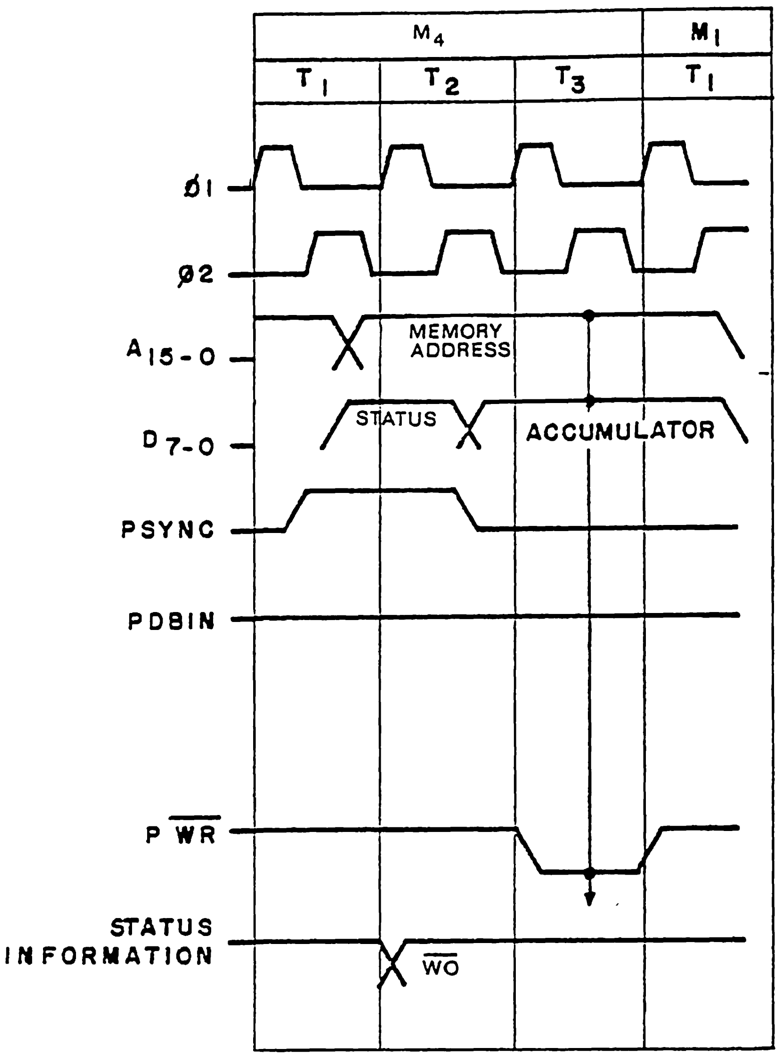


Figure 3-9. Memory Write Cycle Timing.

The $\overline{\text{SWO}}$ output from K is presented on pin 97 of the bus (zone A4) through non-inverting bus driver X as a LOW. The $\overline{\text{SWO}}$ signal is applied through inverter M on the Interface Card (Figure 3-15, sheet 2, zone B6) and presented to the Display/Control card as SWO. The SWO signal present on the Display/Control card lights the WO LED (Figure 3-16, zone C3) on the front panel of the 8800b, indicating data is being transferred to memory from the CPU.

At the beginning of T3, the CPU data is transferred to the memory via the bus. The CPU data out (DOØ through DO7) is applied to the bus (zone C1) through bi-directional gates D and E ($\overline{\text{CS}}$ and $\overline{\text{DIEN}}$ are LOW) and non-inverting bus drivers M and W (zones C7 and C3). The bus data is presented to memory and written in by the MWRITE signal.

After the CPU data is settled on the bus and presented to memory, a $\overline{\text{WR}}$ signal (zone C8) is generated LOW by M. The $\overline{\text{WR}}$ signal is applied to pin 77 (zone D1) of the bus through non-inverting bus driver V (zone D8) as $\overline{\text{PWR}}$. The $\overline{\text{PWR}}$ signal is inverted HIGH by U on the Interface Card (Figure 3-15, sheet 2, zone B3) and applied to pin 1 of NAND gate A (zone C3), enabling pin 6 LOW ($\overline{\text{SOUT}}$ is HIGH on pin 2). The LOW at pin 6 forces the output of NOR gate A (zone C2) HIGH which is applied to pin 68 of the bus through non-inverting bus driver H (zone B2) as MWRITE. The MWRITE signal allows the memory to store the CPU data in the addressed memory location, thus completing the CPU to memory data transfer.

3-24. MEMORY TO CPU DATA TRANSFER

A Memory to CPU data transfer is accomplished whenever an instruction is encountered to perform this operation. For example, a load accumulator LDA (072_8) instruction requires the specified addressed memory location to transfer its contents to the accumulator in the CPU. The LDA instruction was fetched during M1 and the specified memory location determined during the memory read cycles, M2 and M3. The memory data is transferred to the CPU by an additional Memory Read Cycle operation (M4). The M4 operation requires the CPU to output the specified addressed memory location to memory, allowing the data in the specified addressed memory location to be transferred to the CPU in an identical manner as M2.

For a detailed operation description of the M2 cycle, refer to Paragraph 3-17. Note as you read the description that the specified memory address location is presented to memory on the fifteen individual address lines, allowing that location to transfer its data to the CPU.

3-25. CPU TO EXTERNAL DEVICE DATA TRANSFER

A CPU to External Device data transfer is accomplished when an output instruction (323_8) is fetched from a memory location during M1, and the external device number (XXX_8) is read from a memory location during M2 by the CPU. The data from the CPU is transferred to the external device by an Output Write Cycle operation (M3).

3-26. <u>OUTPUT WRITE CYCLE BASIC OPERATION (Figure 3-10)</u>

The Output Write Cycle operation will allow the CPU to output data to an external device. After completion of the Memory Read Cycle (M2), the program counter is not incremented until the completion of the Output Write Cycle. Several signals are generated by the CPU in order to transfer the data to the external device.

The SOUT and PSYNC external device ADDRESS (XXX_8) number, consisting of sixteen individual outputs (AØ-A7) from the CPU, is presented to the external device (I/O) to condition the I/O card. With the I/O conditioned, a $\overline{\text{PWR}}$ signal from the CPU allows the I/O to transfer the CPU data via the DATA OUT (DOØ-DO7) lines to the external device. The $\overline{\text{SWO}}$ output from the CPU is presented to the Display/Control through the Interface to light the WO (write output) LED on the 8800b front panel. The SOUT and AØ through A15 outputs are applied to the Display/Control through the Interface to light the OUT output and ADDRESS LEDs on the 8800b front panel.

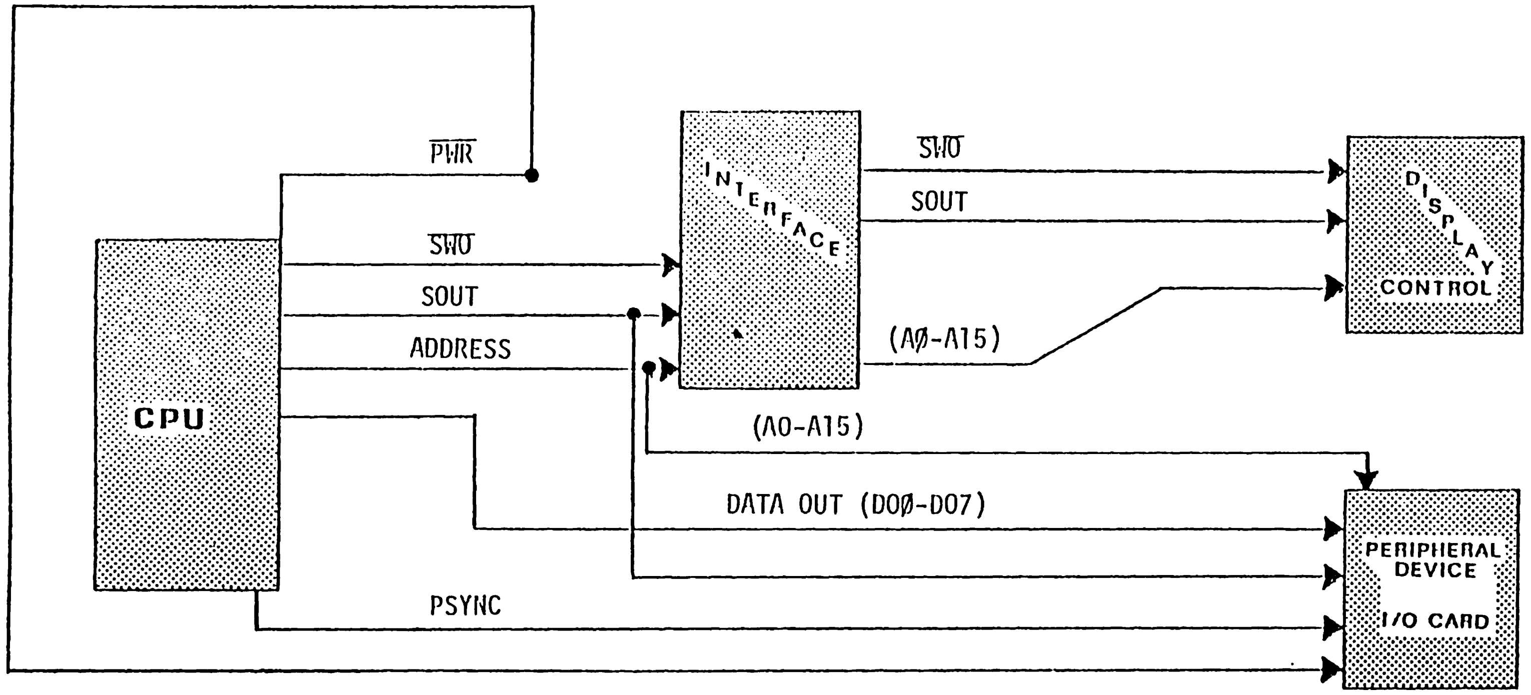


Figure 3-10. Output Write Cycle Block Diagram

3-27. OUTPUT WRITE CYCLE DETAILED OPERATION

The following paragraphs describe the Output Write Cycle operation in detail. Refer to Figure 3-11, Output Write Cycle Timing, during the explanation. The Output Write Cycle operation (M3) requires three Ø1 and Ø2 clock pulses. Each clock period performs a certain operation as described in the following paragraphs.

During the latter portion of T1, several outputs are generated by the CPU 8080 IC (Figure 3-14); Address data AØ through A15 (zone B6), status data DØ through D7, and a SYNC signal (zone C8). The AØ through A15 data contains the external device number and is applied to the I/O card via the bus through non-inverting bus drivers U, P, and N (zone B9) on the CPU in order to enable the I/O card. The address data (AØ-A15) is also applied through inverters P, W, and X on the Interface Card (Figure 3-15, sheet 1, zone B5) and presented to the Display/Control card. The AØ through A15 signals present on the Display/Control card light the appropriate AØ through A15 LEDs, indicating the address of the external device. The DØ through D7 data is applied to K (zone B5) on the CPU through bi-directional circuits D and E. The status data is enabled through D and E at this time because $\overline{\text{CS}}$ and $\overline{\text{DIEN}}$ are LOW. The SYNC output is applied to the clock generator F (zone B7) which conditions F to generate a signal during T2.

At the beginning of T2, a $\overline{\text{STSTB}}$ (zone B7) is generated LOW from F as a result of the HIGH SYNC input and internal timing of F. The $\overline{\text{STSTB}}$ is applied to the data latch K (zone B5), allowing the status data DØ through D7 to be stored into K. The status data present at the output of K conditions the I/O card to receive data from the CPU by enabling the SOUT and $\overline{\text{SWO}}$ signals.

A SOUT output from K is presented HIGH on pin 45 of the bus (zone A4) through non-inverting bus driver X. The SOUT signal is applied through inverter V on the Interface Card (Figure 3-15, zone B5) and presented to NAND gate A (zone C3) and the Display/Control card as SOUT. The $\overline{\text{SOUT}}$ signal disables NAND gate A to insure that a MWRITE output is not produced when writing data to an external

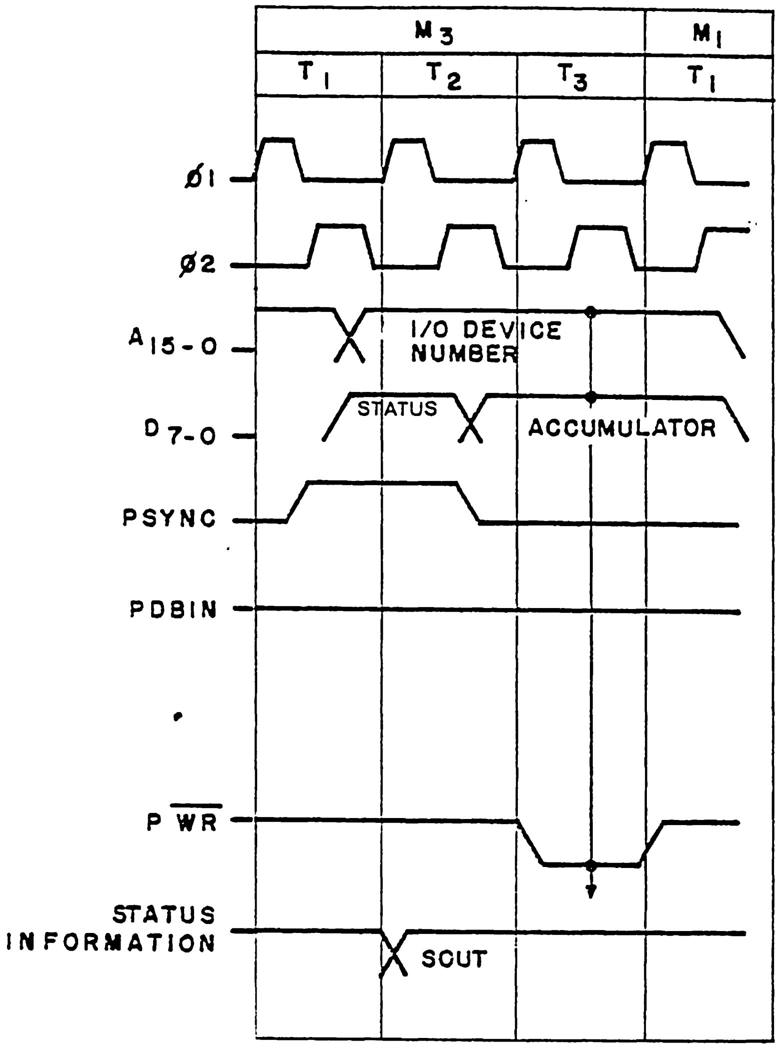


Figure 3-11. Output Write Cycle Timing.

device. It is applied to the Display/Control to light the "OUT" LED (Figure 3-16, sheet 3, zone B3), indicating data is being transferred from the CPU to an external device. The SOUT output from the CPU (Figure 3-14, zone A5) is applied to the external device I/O card in order to initiate a data transfer from the CPU during T3.

At the beginning of T3, the CPU data is transferred to the external device via the bus. The CPU data out (DOØ through DO7) is applied to the bus (zone C1) through bi-directional gates D and E ($\overline{CS}$ and $\overline{DIEN}$ are LOW) and non-inverting bus drivers M and W (zones C7 and C3). The bus data is presented to the external device and written in by the $\overline{PWR}$ signal.

After the CPU data is settled on the bus, a $\overline{WR}$ signal (zone C8) is generated LOW by M. The $\overline{WR}$ signal is applied to pin 77 (zone D1) of the bus through non-inverting bus driver V (zone D8) as $\overline{PWR}$. The $\overline{PWR}$ signal allows the external device to store the CPU data, thus completing the CPU to external device data transfer.

3-28. FRONT PANEL OPERATION

A variety of functions may be performed through the operation of the front panel: e.g. selecting a starting location for a program, examining memory locations, single stepping through a program, depositing and displaying CPU accumulator data, and depositing data into a specified memory location. Each of the functions performed on the 8800b front panel are discussed in the following paragraphs. The run operation was discussed in Paragraph 3-10.

3-29. FRONT PANEL BLOCK DIAGRAM (Figure 3-12)

The front panel switches allow the operator to assume control of the CPU. The CPU is controlled by a FRDY signal which is generated from the front panel display control circuits. The FRDY signal places the CPU in either a wait condition or a run operation.

The CPU is placed in a wait condition when the Switches and Decoding circuits sense that the RUN/STOP switch on the front panel is positioned to STOP. A STOP signal is applied to the Stop/Run

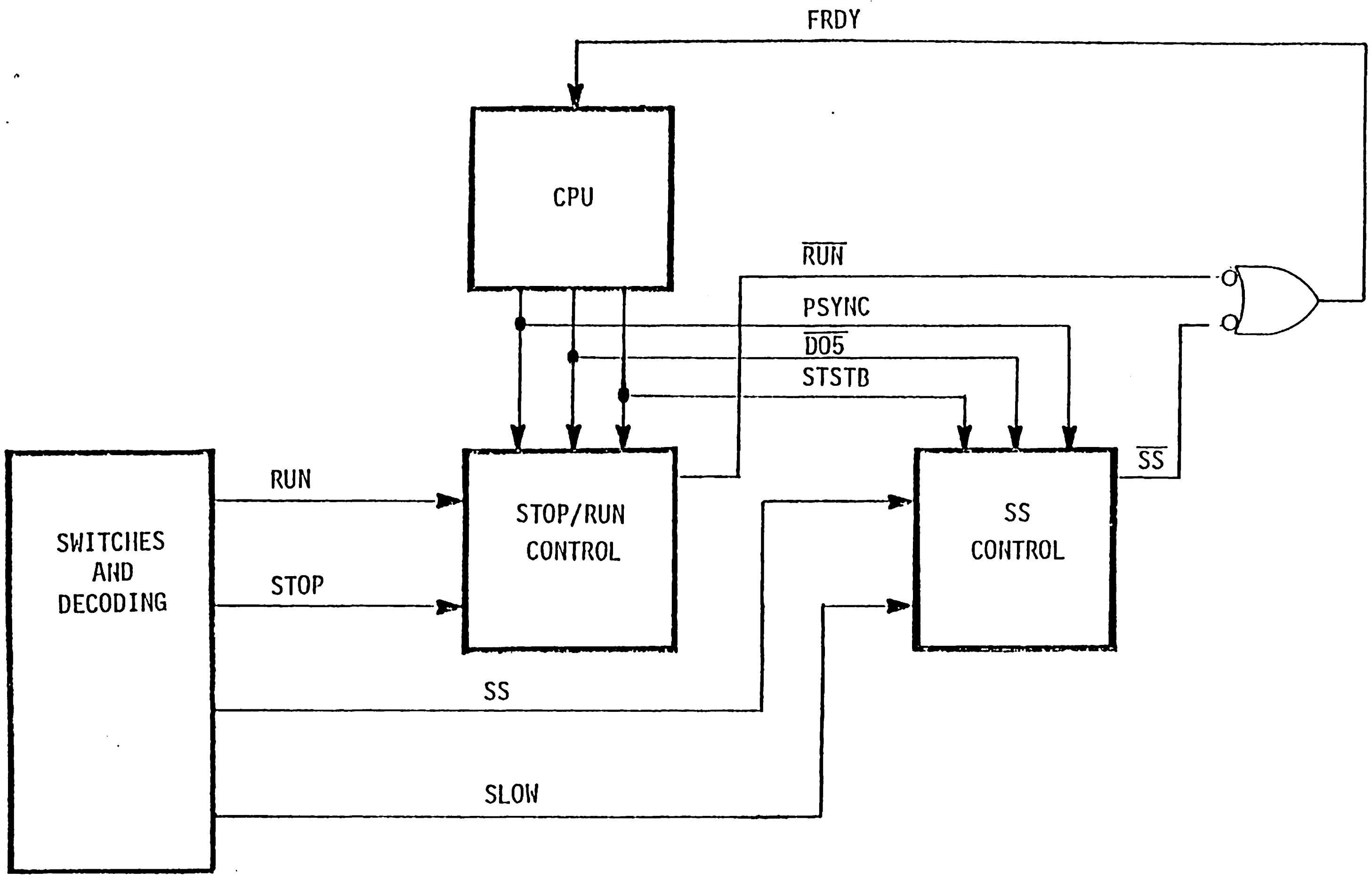


Figure 3-12. Front Panel Block Diagram

Control circuits to disable (HIGH) the $\overline{\text{RUN}}$ signal. The $\overline{\text{RUN}}$ signal forces the CPU to a wait condition by disabling (LOW) the FRDY line. The CPU will not enter a wait condition until the PSYNC, $\overline{\text{DO5}}$, and STSTB signals are presented to the Stop/Run Control circuits. The presence of these signals insures that the CPU will stop during the first machine cycle of an instruction cycle.

The CPU is placed in a single step (SS) or slow run operation by the generation of an SS or SLOW signal from the Switches and Decoding circuits. The SS or SLOW run operation allows the CPU to perform one instruction cycle. The SS signal is applied to the SS Control circuit, enabling (LOW) the $\overline{\text{SS}}$ signal. The $\overline{\text{SS}}$ signal allows the CPU to execute one instruction cycle by enabling the FRDY signal. Upon the completion of the instruction cycle, the CPU attempts to perform another instruction cycle, but the PSYNC, $\overline{\text{DO5}}$, and STSTB signals reset the SS Control circuits forcing the CPU to a wait condition.

3-30. STOP OPERATION

The stop operation allows the operator to use the switches on the 8800b front panel. The stop operation is activated when the RUN/STOP switch on the 8800b front panel is momentarily depressed to STOP.

The RUN/STOP circuits are located on the Display/Control card (Figure 3-16, sheet 2, zone A9). With the RUN/STOP switch momentarily depressed, a LOW is applied to quad latch C1, input D1. The occurrence of the next C13 clock (zone A1) causes the $\overline{\text{Q}}$ output at pin 3 of C1 (zone B9) to go HIGH which is applied to quad latch N1, input D1. The next C13 clock causes the Q output at pin 7 of N1 (zone B9) to go HIGH which is applied to the D input of M1. A HIGH present at D produces a clock pulse to set M1, stopping the CPU.

The clock pulse that sets M1 is derived from three signals: $\overline{\text{DO5}}$, $\overline{\text{PSYNC}}$, and $\overline{\text{STSTB}}$ (zone D8). The signals are enabled during machine cycle 1 (paragraph 3-14) of an 8800b instruction cycle, and their presence generates a clock to M1 (zone C9). This insures that the 8800b stops during the first machine cycle of an instruction cycle. The DO5 signal is generated by the CPU (Figure 3-14, zone C1) and presented to pin 39 of the bus as a HIGH through the bi-directional gate E (zone C7) and non-inverting bus driver W (zone C3) and applied to the Interface Card (Figure 3-15, sheet 2, zone C2). The DO5 signal is inverted by Y (zone B2) and inverted again by R1 on the Display/Control Card (Figure 3-16, sheet 2, zone D8) and applied HIGH to pin 3 of NAND gate D1 (zone C8). The PSYNC is generated by the CPU (Figure 3-14, zone D1) on pin 76 of the bus as a HIGH through non-inverting bus driver V (zone D8) and applied to the Interface Card (Figure 3-15, sheet 2, zone A3). PSYNC is inverted by U (zone B3) and R1 on the Display/Control Card (sheet 5, zone B3) and applied HIGH to pin 4 of NAND gate D1 (zone C8).

The $\overline{\text{STSTB}}$ is generated by the CPU (Figure 3-14, zone A4) to pin 56 of the bus as a LOW through non-inverting bus driver R and applied to the Interface Card (Figure 3-15, sheet 2, zone A4). The $\overline{\text{STSTB}}$ is inverted and then inverted again by the Interface Card (sheet 2, zone A4) and applied to pin 5 of NAND gate D1 on the Display/Control Card (Figure 3-16, sheet 2, zone C8) as a HIGH. These signals allow NAND gate D1 to produce a HIGH at gate P1, pin 6 (zone C8), which sets M1. The $\overline{\text{Q}}$ output of M1 goes LOW and is applied through K1 (zone A8) to enable all the front panel switches. The $\overline{\text{Q}}$ output is also presented to gate P1 which keeps a high on the CK input of M1 (zone C9), insuring that M1 remains set after the stop switch is released.

Because M1 is set, the Q output of M1 (zone C9) is HIGH, disabling the $\overline{\text{RUN}}$ and $\overline{\text{FRDY}}$ signals. The $\overline{\text{FRDY}}$ signal is applied to NAND gate C on the CPU (Figure 3-14, zone A8) through the Interface (Figure 3-15, sheet 2, zone A1) as a LOW. This inhibits the RDYIN signal at F (Figure 3-14, zone B7) which disables the READY signal to M (zone A8), thereby halting the CPU.

3-31. <u>SINGLE STEP OPERATION</u>

The single step operation allows the operator to increment one instruction cycle at a time. The single step operation is activated when the SINGLE STEP/SLOW switch is momentarily positioned to SINGLE STEP.

The SINGLE STEP circuits are located on the Display/Control card (Figure 3-16, sheet 1, zone A8). With the SINGLE STEP/SLOW switch momentarily positioned to SINGLE STEP, a LOW is presented to pin 1 of gate P1 (zone C8). The LOW input at D1 generates a clock pulse which sets M1 (zone A7), producing a LOW at the $\overline{Q}$ output of M1. The LOW output is applied to pin 13 of gate P1 (zone C9), enabling the $\overline{FRDY}$ signal (zone D9). The CPU performs one instruction cycle with $\overline{FRDY}$ enabled. At the completion of the instruction cycle, the $\overline{D05}$, $\overline{PSYNC}$, and $\overline{STSTB}$ input (zone D8) enable NAND gate T1, pin 12 (zone C6), LOW which produces a LOW at the output of inverter J1 (zone C6). The LOW clears the M1 flip-flop, thereby ending the first single step operation. Additional single step operations are enabled by momentarily depressing the SINGLE STEP/SLOW switch to SINGLE STEP.

The D05 input is applied to pin 1 of NAND gate T1 through jumpers JE and JF (zone D7). If this jumper is removed, pin 1 of NAND gate is always HIGH. Under this condition, the PSYNC and $\overline{STSTB}$ signals would reset M1 after each machine cycle.

3-32. SLOW OPERATION

The slow operation is very similar to the single step operation except the slow operation allows the 8800b to execute instruction cycles at a very slow rate (786 milliseconds vs. 3 milliseconds normal operation).

The slow circuits are located on the Display/Control card (Figure 3-16, sheet 2, zone A8). When the SINGLE STEP/SLOW switch is positioned to SLOW, a HIGH is presented to pin 9 of NAND gate P1 (zone B7). The HIGH at pin 9 enables the C18 clock (zone D7) from a 24-bit counter (sheet 1, zone D1) through NAND gate P1 (sheet 2, zone B7). This clock enables pin 12 of gate D1 (zone C8) HIGH, providing a clock pulse to set M1 (zone A7), producing a LOW at the $\overline{Q}$ output, M1. The LOW output is applied to pin 13 of gate P1 (zone C9), enabling the $\overline{FRDY}$ signal (zone D9). With $\overline{FRDY}$ enabled, the CPU performs one instruction cycle. At the completion of the instruction cycle, the $\overline{D05}$, $\overline{PSYNC}$, and $\overline{STSTB}$ input (zone D8) enable NAND gate T1, pin 12 (zone C6), LOW which produces a LOW at the output of inverter J1 (zone C6). This LOW clears the M1 flip-flop, ending the first single step operation. If the SINGLE STEP/

SLOW switch is still positioned to SLOW, another instruction cycle operation is performed. Otherwise, the machine halts. If jumpers JE and JF (zone C7) are removed, the machine may not stop at the beginning of an instruction cycle.

3-33. RESET OPERATION

The reset operation allows the operator to reset the CPU at anytime during machine operation. The reset is activated when the RESET/EXT CLR switch on the front panel is positioned to RESET.

The reset circuits are located on the Display/Control card (Figure 3-16, sheet 2, zone A2). With the RESET/EXT CLR switch momentarily positioned to RESET, a PRESET signal (zone D3) is applied to the Interface (Figure 3-15, sheet 2, zone D1) as a HIGH. The HIGH is inverted by R and applied to pin 75 (zone A1) of the bus through non-inverting bus driver N (zone B1). The CPU receives the $\overline{\text{PRESET}}$ signal and inverts it twice through G and B (Figure 3-14, zone B6). The output of B is applied to the clock generator F $\overline{\text{RESIN}}$ (reset in) input (zone B7), producing a RESET output to the 8080 (M).

3-34. PROTECT AND UNPROTECT OPERATION

The protect/unprotect operation either prevents any new data from being written into a particular region of memory (protect) or allows new data to be written into a particular region of memory (unprotect). The protect/unprotect operation is controlled by the positioning of the PROTECT/UNPROTECT switch on the front panel.

The protect/unprotect circuits are located on the Display/Control card (Figure 3-16, sheet 2, zone A1). With the PROTECT/UNPROTECT switch positioned to either PROTECT or UNPROTECT, a $\overline{\text{PROTECT}}$ or $\overline{\text{UNPROTECT}}$ signal (zone D3) is applied to the Interface as a LOW. The LOW is inverted by R (Figure 3-15, sheet 2, zone B6) and applied to pin 70 and 20 on the bus to condition the memory. These signals are used to set or reset the protect/unprotect circuits on the addressed memory board.

3-35. PROGRAMMABLE READ ONLY MEMORY (PROM) CIRCUIT

The PROM circuit on the Display/Control Card is used when one of the following operations is performed: Examine, Examine Next, Deposit, Deposit Next, Accumulator Display, Accumulator Load, Accumulator Input and Accumulator Output. Each of the functions requires a program operation that is stored in the PROM. Access to these programs is determined by the type of function to be performed. The PROM operation is similar for each function, therefore two functions are discussed in detail.

3-36. PROM BLOCK DIAGRAM (Figure 3-13)

The PROM circuit contains eight individual programs which are used in conjunction with the following switches: EXAMINE/EX NEXT, DEPOSIT/DEP NEXT, ACCUMULATOR DISPLAY/LOAD, and ACCUMULATOR INPUT/OUTPUT. Activating any of these switches produces a specific binary number on the RA4, RA5, RA6, and RA7 lines (MSBs) from the Switches and Decoding circuit. At the same time the RA4 through RA7 data is generated, a RESET signal is applied to the 4-Bit Counter, conditioning the RAØ, RA1, RA2, and RA3 outputs (LSBs) to zero. The RAØ-RA7 signals are applied to the PROM, and they represent an 8-bit starting address location. There are eight different starting address locations which correspond to the eight different front panel switch settings (refer to Table 3-2). Any of the eight different starting address locations are always even because of the resetting of the 4-Bit Counter.

The PROM circuit outputs a DATA OUT (RDØ-RD7) signal, consisting of eight individual lines, to either the Control Latch or the non-inverting bus driver F. The DATA OUT is transferred to one of these two circuits by the status of the RAØ signal from the 4-Bit Counter. When the RAØ signal is LOW, representing a PROM even address, the Control Latch receives the data. The even addresses of the PROM contain data that is used to enable the Control Latch output lines (S1-S8). After the Control Latch receives the PROM data, a CLOCK signal increments the 4-Bit Counter to an odd PROM address location. During an odd PROM address cycle, the CPU will execute one machine cycle (assuming the S8 bit has been set in the Control Latch). If the cycle is a memory read cycle, an instruction

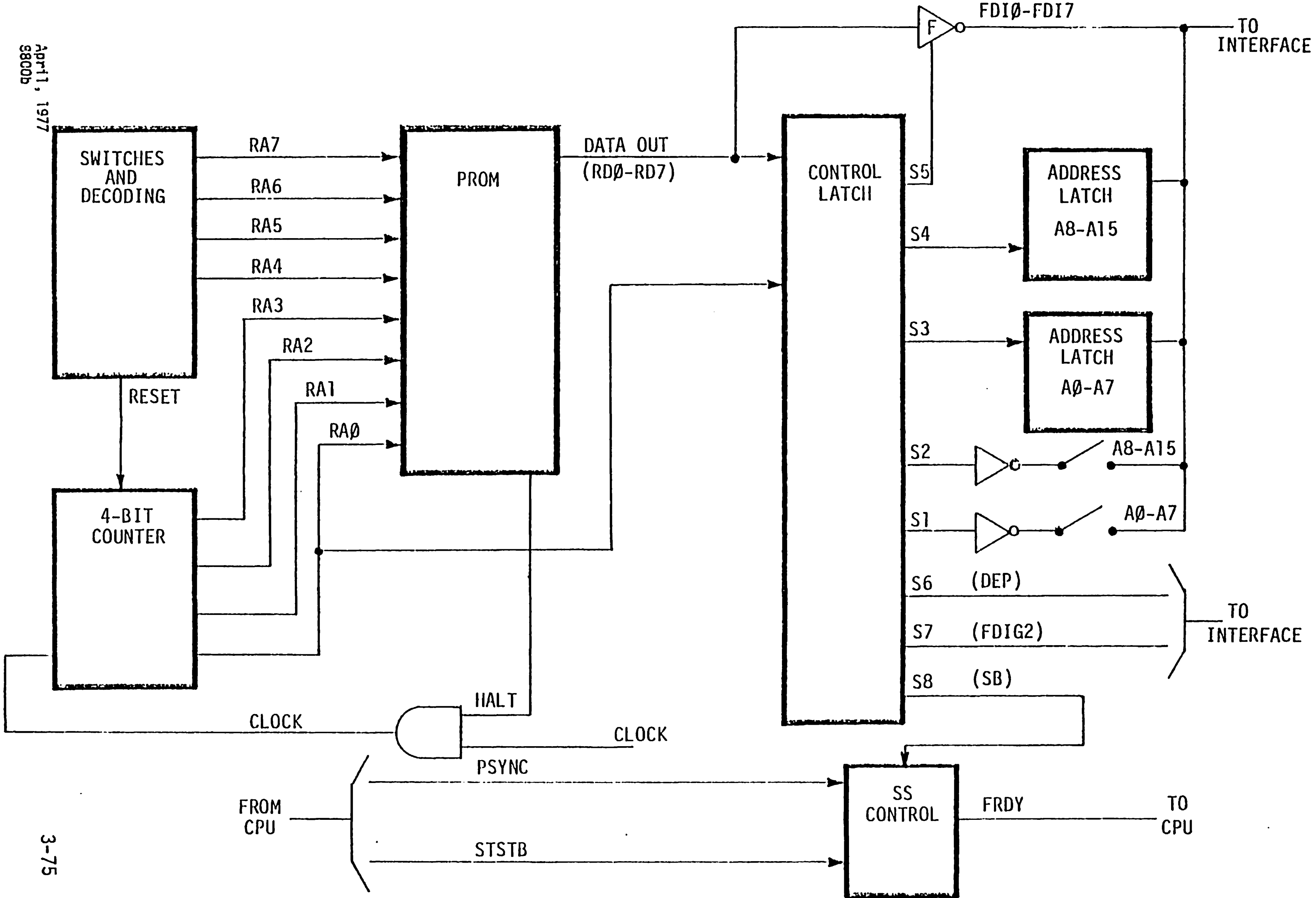


Figure 3-13. PROM Block Diagram

TABLE 3-2. PROM Programs

Front Panel Operations	PROM Address	PROM DATA	Function
Examine	160*	013*	Set S5, S7, S8
	161	303	Jam Jump Instruction to CPU
	162	203	Set S1, S7, S8
	163	000	Jam AØ-A7 switch data to CPU
	164	103	Set S2, S7, S8
	165	000	Jam A8-A15 switch data to CPU
	166	000	Clear control latch
	167	177	Stop
Examine Next	260	013	Set S5, S7, S8
	261	000	Jam NOP instruction to CPU
	262	000	Clear control latch
	263	177	Stop
Deposit	320	206	Set S1, S6, S7
	321	000	Put AØ-A7 switch data and MWRITE pulse on bus
	322	000	Clear control latch
	323	177	Stop
Deposit Next	340	013	Set S5, S7, S8
	341	000	Jam NOP instruction to CPU
	342	206	Set S1, S6, S7
	343	000	Put AØ-A7 switch data and MWRITE pulse on bus
	344	000	Clear control latch
	345	177	Stop
Display Accumulator	060	013	Set S5, S7, S8
	061	323	Output Instruction
	062	013	Set S5, S7, S8

*All PROM address and data information is octal.

TABLE 3-2. PROM Programs - Continued

Front Panel Operations	PROM Address	PROM DATA	Function
	063*	377*	Jam front panel address to CPU
	064	001	Set S8
	065	000	Data in accumulator is transferred to the DØ-D7 LEDs
	066	013	Set S5, S7, S8
	067	303	Jam jump instruction to CPU
	070	043	Set S3, S7, S8
	071	000	Jam AØ-A7 latch data to CPU
	072	023	Set S4, S7, S8
	073	000	Jam A8-A15 latch data to CPU
	074	000	Clear control latch
	075	177	Stop
Accumulator Deposit	220	013	Set S5, S7, S8
	221	333	Jam input instruction to CPU
	222	013	Set S5, S7, S8
	223	376	Jam front panel address to CPU
	224	203	Set S1, S7, S8
	225	000	Data in accumulator is transferred to CPU
	226	013	Set S5, S7, S8
	227	303	Jam jump instruction to CPU
	230	043	Set S3, S7, S8
	231	000	Jam AØ-A7 latch data to CPU
	232	023	Set S4, S7, S8
	233	000	Jam A8-A15 latch data to CPU
	234	000	Clear control latch
	235	177	Stop
Input from external device selected by ADDRESS switches A8-A15	300	013	Set S5, S7, S8
	301	333	Jam input instruction to CPU
	302	103	Set S2, S7, S8
	303	000	Jam A8-A15 switch data to CPU

*All PROM address and data information is octal.

TABLE 3-2. PROM Programs - Continued

Front Panel Operations	PROM Address	PROM DATA	Function
	304*	001*	Set S8
	305	000	Data in accumulator is transferred to specific I/O device
	306	013	Set S5, S7, S8
	307	303	Jam jump instruction to CPU
	310	043	Set S3, S7, S8
	311	000	Jam AØ-A7 latch data to CPU
	312	023	Set S4, S7, S8
	313	000	Jam A8-A15 latch data to CPU
	314	000	Clear control latch
	315	177	Stop
Output from external device selected by ADDRESS switches A8-A15	240	013	Set S5, S7, S8
	241	323	Jam output instruction to CPU
	242	103	Set S2, S7, S8
	243	000	Jam A8-A15 switch data to CPU
	244	001	Set S8
	245	000	Data is transferred from specific I/O device to accumulator
	246	013	Set S5, S7, S8
	247	303	Jam jump instruction to CPU
	250	043	Set S3, S7, S8
	251	000	Jam AØ-A7 latch data to CPU
	252	023	Set S4, S7, S8
	253	000	Jam A8-A15 latch data to CPU
	254	000	Clear control latch
	255	177	Stop

*All PROM address and data information is octal.

byte is supplied to the CPU on the FDIØ-FDI7 lines.

The instruction data at the odd PROM address is transferred to the CPU through the Interface from five different sources. The source is determined by the output control lines S1 through S5 from the Control Latch.

The S1 and S2 control lines enable the front panel switch data, AØ through A15, to the Interface. The S3 and S4 control lines enable the Address Latch data, AØ through A15, to the Interface. The S5 control line enables the DATA OUT (RDØ-RD7) from the PROM to the Interface.

The data present at the Interface is applied to the CPU by output control lines S7 and S8 from the Control Latch. The S7 control line allows the Interface to apply the instruction data to the CPU, and the S8 control line enables the FRDY signal. The FRDY signal allows the CPU to receive the instruction data and execute one machine cycle. After the completion of the machine cycle, the PSYNC and STSTB signals from the CPU reset the SS Control circuit. The S6 control line is enabled from the Control Latch to allow data to be deposited into memory. Upon the completion of a PROM program, a HALT signal is generated by the PROM, disabling the CLOCK signal to the 4-Bit Counter.

3-37. <u>EXAMINE OPERATION</u>

The examine operation allows the operator to examine a memory location by using the ADDRESS switches on the front panel. Refer to Table 3-2 during the explanation. The examine operation is activated when the EXAMINE/EXAMINE NEXT switch is momintarily positioned to EXAMINE.

The EXAMINE circuit is located on the Display/Control card (Figure 3-16, sheet 2, zone B7). With the EXAMINE/EXAMINE NEXT switch momentarily positioned to EXAMINE, a LOW is generated at pin 6 of inverter V1 (zone B7) and a HIGH at the output of the remaining V1 and Z1 inverters (zones B6 through B3). The LOW output is applied to pin 6 of gate X1 which generates a HIGH to set L1 (zone D4). The $\overline{\text{RC-CLR}}$ (LOW) and AL-STB (HIGH) outputs from L1 reset a 4-bit binary counter to zero (sheet 2, zone A9) and strobes the current address data into data latches B1 and T1 (zone B6). The L1 latch is cleared by the $\overline{C6}$ signal from the 24-bit binary counter (sheet 1, zone D3). The LOWs and HIGHs from the inverters are presented as RA7 through RA4 inputs to the PROM (sheet 1, zone B9).

The RAØ through RA7 inputs to the PROM (zone B9) represent an address location (160_8). This location is the beginning of the examine program stored in the PROM. The data in address location 160_8 is presented on the RDOØ through RDO7 outputs (013_8) and applied to data latch A (zone D8). After the 4-bit binary counter (zone B9) is LOW during the even addresses (RAØ=0), and a control strobe (CS) to DS2 (zone C8) is generated, the data present at latch A is stored by the A output.

The CS strobe is produced by the 24-bit counter outputs C6, C7, and $\overline{C8}$ (zone D3). When the C6, C7, and $\overline{C8}$ counter outputs are HIGH, NAND gate V (zone D5) is enabled LOW, and CS (zone D6) is applied HIGH to the DS2 input of data latch A (zone C8). With DS2 and $\overline{DS1}$ enabled, the RDØ through RD7 data (013_8) is latched into A. The 013_8 data enables outputs S5, S7, and S8 (zone D7) HIGH. Output S5 is inverted LOW by A1 (zone A6), enabling inverting bus drivers R and S. Outputs S7 and S8 are applied to pins 3 and 13 of NAND gates J (zone D6). With the PROM data stored in latch A and the associated circuits conditioned, NAND gate Z (zone A7) produces a clock pulse to INP A of the 4-bit counter (zone A8). When C8 goes HIGH from the 24-bit counter (zone D3), the 4-bit counter, A output, goes HIGH which addresses PROM location 161_8.

The data in address location 161_8 is present on the RDØ through RD7 outputs 303_8. The 303_8 data is transferred to the Interface on the FDIØ-FDI7 (zone C2) outputs through enabled inverting bus drivers R and S (zone A6). The data is not stored in Latch A because the A output (zone B9) of the 4-bit counter is HIGH (odd address RAØ=1), disabling the $\overline{DS1}$ input (zone C7). The A output is applied to pins 1 and 5 of NAND gates J (zone D6).

The $\overline{\text{FDIØ}}$ through $\overline{\text{FDI7}}$ data presented to the Interface Card (Figure 3-15, sheet 2, zone D8) represents a jump instruction to be stored in the CPU. The CPU cannot receive this instruction and execute it until the $\overline{\text{FDIG2}}$ (zone D7) signal is LOW, and the CPU is released from the wait condition generated when the CPU was stopped. The following operation allows the CPU to receive the jump instruction.

When the C6, C7, and $\overline{\text{C8}}$ outputs of the 24-bit counter on the Display/Control (Figure 3-16, sheet 1, zone D3) are HIGH, another CS signal (zone D6) is generated. The CS signal allows NAND gate J, pins 6 and 12, to produce $\overline{\text{SB}}$ (zone D4) and $\overline{\text{FDIG2}}$ (zone C2) signals.

The $\overline{\text{SB}}$ signal is applied to pin 13 of gate D1 (sheet 2, zone C8) as a LOW, producing a HIGH clock pulse to set M1 (zone C7). The $\overline{\text{Q}}$ output of M1 is applied to pin 13 of NOR gate P1 and inverter R1 (zone D9), allowing the $\overline{\text{FRDY}}$ signal to release the CPU from its wait condition.

The $\overline{\text{FDIG2}}$ signal is applied to pin 12 of NOR gate B (Figure 3-15, sheet 2, zone C7) on the Interface as a LOW which enables NAND gate B, pin 6, LOW (PDBIN is HIGH because the CPU is in a wait condition). The LOW enables the non-inverting drivers F (zone B7), allowing the PROM data (303$_8$) to be applied to M on the CPU through bi-directional gates D and E on the CPU (Figure 3-14, zone C7). Because the READY line to M (zone A8) is HIGH, the CPU inputs the 303$_8$ data which is interpreted by the CPU as a jump instruction. After the completion of the machine cycle, the $\overline{\text{PSYNC}}$ and $\overline{\text{DO5}}$ signals (sheet 2, zone D8) are inverted by R1 and applied to pins 11 and 10 of NAND gate T1 (zone D6). These signals and $\overline{\text{SB}}$ (zone D8) enable T1 which generates a clear to M1 (zone C7), halting the CPU.

The CPU contains a jump instruction but no information as to where to jump. The remaining part of the examine operation allows the ADDRESS switch data to be read into the CPU from the front panel in order for the CPU to jump to that address. NAND gate Z (sheet 1, zone A7) produces another clock pulse to INP A of the 4-bit counter. When C8 goes HIGH and returns LOW (zone D3), the 4-bit counter increments to an even PROM address 162$_8$.

The data in address location 162_8 is present on the RDØ through RD7 outputs (203_8) and applied to data latch A (zone D8). The data present at latch A is stored by the LOW A output (zone B9) during even addresses (RAØ=0) and the generation of the CS strobe (C6, C7, and $\overline{C8}$ HIGH). The 203_8 data enables outputs S1, S7, and S8 (zone D7) HIGH. Output S1 is applied through inverters Y and W (zone C4) to the AØ through A7 switches (open switch HIGH, closed switch LOW), and the switch information is presented to the Interface as $\overline{\text{FDIØ}}$ through $\overline{\text{FDI7}}$. Outputs S7 and S8 are applied to pins 3 and 13 of NAND gate J (zone D6) and are used to generate the $\overline{\text{FDIG2}}$ and $\overline{\text{SB}}$ signals as described in the jump instruction transfer. With the data presented to the Interface Card and the associated circuits conditioned, NAND gate (zone A7) is enabled (C8 HIGH), producing a clock pulse to INP A, incrementing the 4-bit counter (zone A8) to address 163_8.

The data in address 163_8 is not stored in latch A because it is an odd address. However, the A output (zone B9) is applied to pins 1 and 5 of NAND gates J (zone D6) as a HIGH, allowing the CS signal to produce the $\overline{\text{SB}}$ and $\overline{\text{FDIG2}}$ outputs. The $\overline{\text{SB}}$ and $\overline{\text{FDIG2}}$ signals allow the transfer of the first eight address data bits (address switches AØ-A7) to the CPU, and the operation is identical to the jump instruction.

After the CPU receives the eight address bits, the 4-bit binary counter is incremented to address 164_8. The data in 164_8 (01000110 - 103_8) is stored in latch A (zone D7) because it is an even address. The 103_8 data enables S2, S7, and S8 (zone D7) HIGH. Output S2 is applied to inverter A1 (zone C6), gate Z (zone C5), and inverters W and U (zone C4) to the A8 through A15 address switches. The switch information is presented to the Interface as $\overline{\text{FDIØ}}$ through $\overline{\text{FDI7}}$. Outputs S7 and S8 condition NAND gates J (zone D6) and are used to generate $\overline{\text{SB}}$ and $\overline{\text{FDIG2}}$ during the next address. With the data present to the Interface and the associated circuits conditioned, NAND gate Z (zone A7) is enabled (C8 HIGH), producing a clock pulse to INP A, incrementing the 4-bit counter (zone A8) to address 165_8.

Address 165_8 operation is the same as address 163_8, allowing the A8 through A15 address data to be stored in the CPU. After the CPU receives the second byte of the address, it executes a jump to that address. Address 166_8 clears the data latch A (zone D7) and allows the CPU to address memory (Figure 3-14, zone B9). The memory presents the addressed memory location data to the CPU via data input lines DIØ through DI7 (zone B1). The data is enabled through inverters Y, S, L, and J (zone B4) and non-inverters P, W (zone C3) to the Interface (Figure 3-15, sheet 1, zone B1). The data is enabled through the G data latch (sheet 3, zone B4) to the Display/Control (Figure 3-16, sheet 3, zone D1) and displayed on the LEDs. The G latch (sheet 3, zone B4) is enabled because the $\overline{\text{RUN}}$ signal (zone A6) is HIGH, producing a HIGH at input MD of the data latch.

While the memory data was being displayed, the 4-bit binary counter (Figure 3-16, sheet 1, zone A9) is incremented to address 167_8. The data in 167_8 (01111111 - 177_8) is applied to NAND gate N (zone B7), producing a HIGH at gate Z (zone B8). The HIGH at gate Z disables NAND gate Z (zone A8), inhibiting any following clock pulses to the 4-bit binary counter, thus ending the examine operation.

3-38. ACCUMULATOR DISPLAY OPERATION

The accumulator (ACC) display operation allows the operator to monitor the contents of the CPU accumulator. Refer to Table 3-1, PROM Programs, during the explanation. The ACC display operation is activated when the ACC DISPLAY/ACC DEPOSIT switch is momentarily positioned to ACC DISPLAY.

The ACC DISPLAY circuit is located on the Display/Control card (Figure 3-16, sheet 2, zone A5). With the ACC DISPLAY/ACC DEPOSIT switch momentarily positioned to ACC DISPLAY, a LOW is generated at pins 8 and 10 of inverter V1 (zone B5), and a HIGH is generated at the output of the remaining V1 and Z1 inverters (zones B7 through B3). The LOW outputs are applied to pins 6 and 5 of gate X1 which generates a HIGH to set L1 (zone D4). The $\overline{\text{RC-CLR}}$ (LOW) and AL-STB (HIGH) outputs from L1 reset a 4-bit binary counter to all zeros (sheet 1, zone A9) and strobe the address in the P counter into data latches B1 and T (zone B6). The P counter address data is stored because the P counter increments during the accumulator display operation. The original P count is saved and restored in the CPU after the ACC display operation is complete. The L1 latch is cleared by the $\overline{C6}$ signal from the 24-bit binary counter (sheet 1, zone D3). The LOW and HIGHs from the inverters are presented as RA7 through RA4 inputs to the PROM (sheet 1, zone B9). An $\overline{ACC\ DSP}$ signal (zone D3) is also applied LOW to the Interface (Figure 3-15, sheet 3, zone A1), producing a LOW to the MD input of data latch G (zone A4).

The RAØ through RA7 inputs to the PROM (zone B9) represent an address location (060_8). This location is the beginning of the ACC display program stored in the PROM. The data in address location 060_8 is presented on the RDØ through RD7 outputs (013_8) and applied to data latch A (zone D8). The data present at latch A is stored by the LOW A output (zone B9) during the even addresses (RAØ=0) and the generation of a control strobe (CS) to DS2 (zone C8).

The CS strobe is produced by the 24-bit counter outputs C6, C7, and $\overline{C8}$ (zone D3). When the C6, C7, and $\overline{C8}$ counter outputs are HIGH, NAND gate V (zone D5) is enabled LOW, the CS (zone D6) is applied HIGH to the DS2 input (zone C8). The RDØ through RD7 data (013_8) is latched into A with DS2 and $\overline{DS1}$ enabled. The 013_8 data enables outputs S5, S7, and S8 (zone D7) HIGH. Output S5 is inverted LOW by A1 (zone A6), enabling inverting bus drivers R and S. Outputs S7 and S8 are applied to pins 3 and 13 of NAND gates J (zone D6). With the PROM data stored in latch A and the associated circuits conditioned, NAND gate Z (zone A7) is enabled, producing a clock pulse to INP A of the 4-bit counter (zone A8). When C8 goes HIGH from the 24-bit counter (zone D3), the 4-bit counter A output goes HIGH which addresses PROM location 061_8.

The data in address location 061_8 is present on the RDØ through RD7 outputs (323_8). The 323_8 data is transferred to the Interface on the $\overline{FDIØ}$-$\overline{FDI7}$ (zone C2) outputs through enabled inverting bus drivers R and S (zone A6). The data is not stored in latch A because the A output (zone B9) of the 4-bit counter is HIGH (odd address), disabling the $\overline{DS1}$ input (zone C7). The A output is applied to pins 1 and 5 of NAND gates J (zone D6).

The FDIØ through $\overline{FDI7}$ data presented to the Interface (Figure 3-15, sheet 2, zone D8) represents an output instruction to be stored in the CPU. The CPU cannot receive this instruction and execute it until the FDIG2 (zone D7) signal is LOW, and the CPU is released from the wait condition generated when the CPU was stopped. The following operation allows the CPU to receive the output instruction.

When the C6, C7, and $\overline{\text{C8}}$ outputs of the 24-bit counter on the Display/Control (Figure 3-16, sheet 1, zone D3) are HIGH, another CS signal (zone D6) is generated. The CS signal allows NAND gate J, pins 6 and 12, to produce a $\overline{\text{FDIG2}}$ (zone C2) and $\overline{\text{SB}}$ (zone D4) signal.

The $\overline{\text{SB}}$ signal is applied to pin 13 of gate D1 (sheet 2, zone C8) as a LOW which produces a HIGH clock pulse to set M1 (zone C7). The $\overline{\text{Q}}$ output of M1 is applied to gate P1 and inverter R1 (zone D9), allowing the $\overline{\text{FRDY}}$ signal to release the CPU from its wait condition.

The $\overline{\text{FDIG2}}$ signal is applied to pin 12 of gate 13 (Figure 3-15, sheet 2, zone C7) as a LOW which enables NAND gate B, pin 6, LOW. The LOW allows the PROM data (323_8) to be applied to M on the CPU through bi-directional gates D and E on the CPU (Figure 3-14, zone C7). Because the READY line to M (zone A8) is HIGH, the CPU inputs the 323_8 data which is interpreted as an output instruction. After the completion of the machine cycle, the $\overline{\text{PSYNC}}$ and $\overline{\text{DO5}}$ signals (Figure 3-14, sheet 2, zone D8) are inverted by R1 and applied to pins 11 and 10 of NAND gate T1 (zone D6). These signals and $\overline{\text{SB}}$ (zone D8) enable T1 which generates a clear to M1 (zone C7), halting the CPU.

The CPU contains an output instruction but no information as to where to output data. The next part of the ACC display operation allows the CPU to output data to the front panel data LEDs (DØ through D7). NAND gate Z (sheet 1, zone A7) is enabled (C8 HIGH), producing a clock pulse to INP A, incrementing 4-bit counter (zone A8) to address 062_8.

The data in address location 062_8 is present on the RDØ through RD7 outputs (013_8) and stored in data latch A (zone D8) in the same manner as address 060_8. This insures that the S5, S7, and S8 outputs (zone D7) are enabled as in address 060_8. After the completion of this operation, NAND gate Z (zone A7) is enabled, producing a clock pulse to INP A, incrementing the 4-bit counter (zone A8) to address 063_8.

The data in address location 063_8 is present on the RDØ through RD7 outputs (377_8) which is the I/O channel number for the front panel. The 377_8 data is transferred to the CPU in the same manner as the output instruction at address 061_8. The 377_8 data allows the CPU to address the front panel and output the accumulator data to the DØ through D7 LEDs on the front panel. With the output instruction and front panel address number stored in the CPU, NAND gate Z (zone B7) is enabled, producing a clock pulse to INP A, incrementing the 4-bit binary counter (zone A8) to address 064_8.

The data in address location 064_8 is present to the RDØ through RD7 outputs (001_8) and stored in data latch A (zone D8). The 001_8 data enables output S8 (zone D7) HIGH which is used during address 065_8. After the data in address location 064_8 is stored in data latch A, NAND gate Z (zone B7) is enabled, producing a clock pulse to INP A, incrementing the 4-bit binary counter (zone A8) to address 065_8.

Address 065_8 enables the $\overline{\text{SB}}$ signal (zone D4) as described in address 061_8. The CPU performs one machine cycle with $\overline{\text{SB}}$ enabled. During the one machine cycle, the CPU outputs address 377_8 on the AØ - A7 and A8 - A15 address lines to the bus (Figure 3-14, zone B9). The CPU also outputs accumulator data through bi-directional gates D and E (zone C7) and non-inverting bus drivers P and W (zone C3) to the data out (DOØ-DO7) bus. The address data (377_8) enables NAND gates L on the Interface board (Figure 3-15, sheet 3, zone C6) LOW. The LOWs enable gate D (zone C4) HIGH which is applied through jumper JE/JF to pin 9 of NAND gate K (zone B4). During an output instruction, the $\overline{\text{SOUT}}$ and PWR signals (zone B6) are generated by the CPU which enables NAND gate K (zone B4) output LOW. The LOW is applied through jumper JD/JC and inverted HIGH by gate J (zone C3) and presented to the STB input (zone B4) of latch G.

The data from the CPU is presented to the Interface (sheet 1, zone C1) and stored in data latch G (sheet 3, zone B4) during the output instruction because the STB and MD inputs are enabled.

The outputs of data latch G light the appropriate data LED (DØ-D7) on the Display/Control Panel (Figure 3-16, sheet 3, zone D2). After the machine cycle is complete, NAND gate Z (sheet 1, zone B7) is enabled, producing a clock pulse to INP A, incrementing the 4-bit binary counter to address 066_8.

The data (013_8) in address location 066_8 is stored in data latch A (zone D8) and enables the S5, S7, and S8 outputs (zone D7) HIGH. After the completion of this operation, NAND gate Z is enabled, and the 4-bit binary counter is incremented to address 067_8. Address 067_8 contains a jump instruction (303_8) which is stored in the CPU in the same manner as the previous instructions. The jump instruction will force the CPU back to the original P counter address which was stored in data latches B1 and T (zone B5) at the beginning of the ACC display operation. The remainder of the ACC display operation will transfer the address stored (AØ-A7) in B1 and (A8-A15) in T to the CPU. After the jump instruction is stored in the CPU, the 4-bit binary counter is incremented to address 070_8.

The data in address location 070_8 is present on the RDØ through RD7 outputs (043_8) and applied to data latch A (zone D8). The data present at latch A is stored by the A output of the 4-bit binary counter (zone B9) being LOW during even addresses and the generation of the CS strobe (C6, C7, and $\overline{\text{C8}}$ HIGH). The 043_8 data enables outputs S3, S7, and S8 (zone D7) HIGH. Output S3 is applied to the DS2 input of data latch B1 (zone C5), presenting the output data (AØ-A7) to the Interface as $\overline{\text{FDIØ}}$ through $\overline{\text{FDI7}}$. Outputs S7 and S8 are applied to pins 3 and 13 of NAND gate J (zone D6) and are used to generate the $\overline{\text{SB}}$ and $\overline{\text{FDIG2}}$ signals as described in the previous instruction transfers. With the data present to the Interface and the associated circuits conditioned, NAND gate Z (zone A7) is enabled, producing a clock pulse to INP A, incrementing the 4-bit counter (zone A8) to address 071_8.

The data in address 071_8 is not stored in latch A because it is an odd address. However, the A output (zone B9) is applied to pins 1 and 5 of NAND gates J (zone D6) as a HIGH, enabling the CS signal to produce the $\overline{\text{SB}}$ and $\overline{\text{FDIG2}}$ outputs. The $\overline{\text{SB}}$ and $\overline{\text{FDIG2}}$ signals allow the transfer of the first eight address data latch bits to the CPU, and the operation is identical to the previous instructions.

After the CPU receives the eight address bits, the 4-bit binary counter increments to address 072_8. The data in 072_8 (023_8) is stored in latch A (zone D7) because it is an even address. The 023_8 data enables S4, S7, and S8 (zone D7) HIGH. Output S4 is applied to the DS2 input of data latch T (zone A6), presenting the output data (A8-A15) to the Interface as $\overline{\text{FDI}\emptyset}$ through $\overline{\text{FDI7}}$. Outputs S7 and S8 condition NAND gates J (zone D6) and are used to generate $\overline{\text{SB}}$ and $\overline{\text{FDIG2}}$ during the next address (073_8). With the data present to the Interface and the associated circuits conditioned, NAND gate Z (zone A7) is enabled (C8 HIGH), producing a clock pulse to INP A, incrementing the 4-bit counter (zone A8) to address 073_8.

Address 073_8 operation is the same as address 071_8, allowing the A8 through A15 address data to be stored in the CPU. Address 074_8 clears the data latch A (zone D7) and allows the CPU to jump to the original P counter address, conditioning the CPU for normal operation.

After conditioning the CPU, the 4-bit binary counter (zone A9) is incremented to address 075_8. The data in 075_8 (177_8) is applied to NAND gate N (zone B7), producing a HIGH at gate Z (zone B8). The HIGH at gate Z disables NAND gate Z (zone A8), inhibiting any following clock pulses to the 4-bit binary counter, thus ending the ACC display operation.

3-39. 8800b OPTIONS

The 8800b has several options which may be selected by the operator. Two options may be used on the Display/Control card, and three options may be used on the Interface card.

3-40. DISPLAY/CONTROL CARD OPTIONS

The Display/Control card options contain a choice of front panel slow operation clock frequencies and a choice of completing one instruction cycle or machine cycle in single step or slow operation. The normal slow operation clock frequency requires a connection between jumpers JA and JD (Figure 3-16, sheet 1, zone D2). For slower operation, jumpers JB to JD or JC to JD may be connected. The normal single/step or slow operation requires a connection between jumpers JE and JF (sheet 2, zone D7) which allows the 8800b CPU to complete one instruction cycle before resuming a wait condition. However, if the operator wishes to execute one machine cycle after each single/step or slow operation, remove jumpers JE and JF which disables the $\overline{\text{DO5}}$ signal (zone D8).

3-41. INTERFACE CARD OPTIONS

One Interface Card option allows the operator to monitor any data from an external device on the DØ through D7 front panel LEDs. Data may be monitored from an external device if jumpers JA and JB are connected (Figure 3-15, sheet 3, zone C3). NAND gate K is enabled LOW when the Ø1, $\overline{\text{PDBIN}}$, and $\overline{\text{SINP}}$ signals (zone C6) are present during an external device to CPU data transfer. The LOW is presented through JB and JA (zone C3) to gate J which produces a HIGH to the STB input of data latch G (zone B4). The HIGH on STB allows the data present on the DOØ-DO7 line (zone B6) to be displayed on the DØ-D7 LEDs on the front panel.

The remaining Interface card options pertain to jumpers JE and JF (zone C4) and jumpers JD and JC (zone C3). If jumpers JE and JF and JC and JD are connected, only data addressed to the front panel (377_8) is displayed. If jumpers JE and JF are removed, all output data from the CPU is displayed on the front panel.

3-42. 8800b POWER SUPPLIES

The 8800b requires a positive 8 volt, 18 ampere supply, a positive 18 volt, 2 ampere supply, and a -18 volt, 2 ampere supply (Figure 3-17). When the ON/OFF switch on the front panel is positioned to ON, a 110 AC voltage is applied to transformer T1. Two bridge rectifiers on the secondary of T1 produce the positive 8, 18, and negative 18 voltage supplies which are applied to the 8800b circuits. The positive and negative 18 volt supplies are pre-regulated by the Q1 and Q2 transistor circuits on the power supply board.

The 8800b printed circuit cards receive the supply voltages on the bus. Each printed circuit card contains its own voltage regulator circuits which produce the operating voltage for the particular printed circuit card.

The CPU card (Figure 3-18) requires a regulated positive and negative 5 volt source and a regulated positive 12 volt source. These voltages are produced by VR1, VR2, and D2 circuits.

The Interface card (Figure 3-19) requires a regulated positive 5 volt source which is produced by the VR1 circuit.

The Display/Control card (Figure 3-20) requires an unregulated positive 8 volt source, a regulated positive 5 volt source, and a regulated negative 9 volt source. The regulated voltages are produced by the VR1 and VR2 circuits.

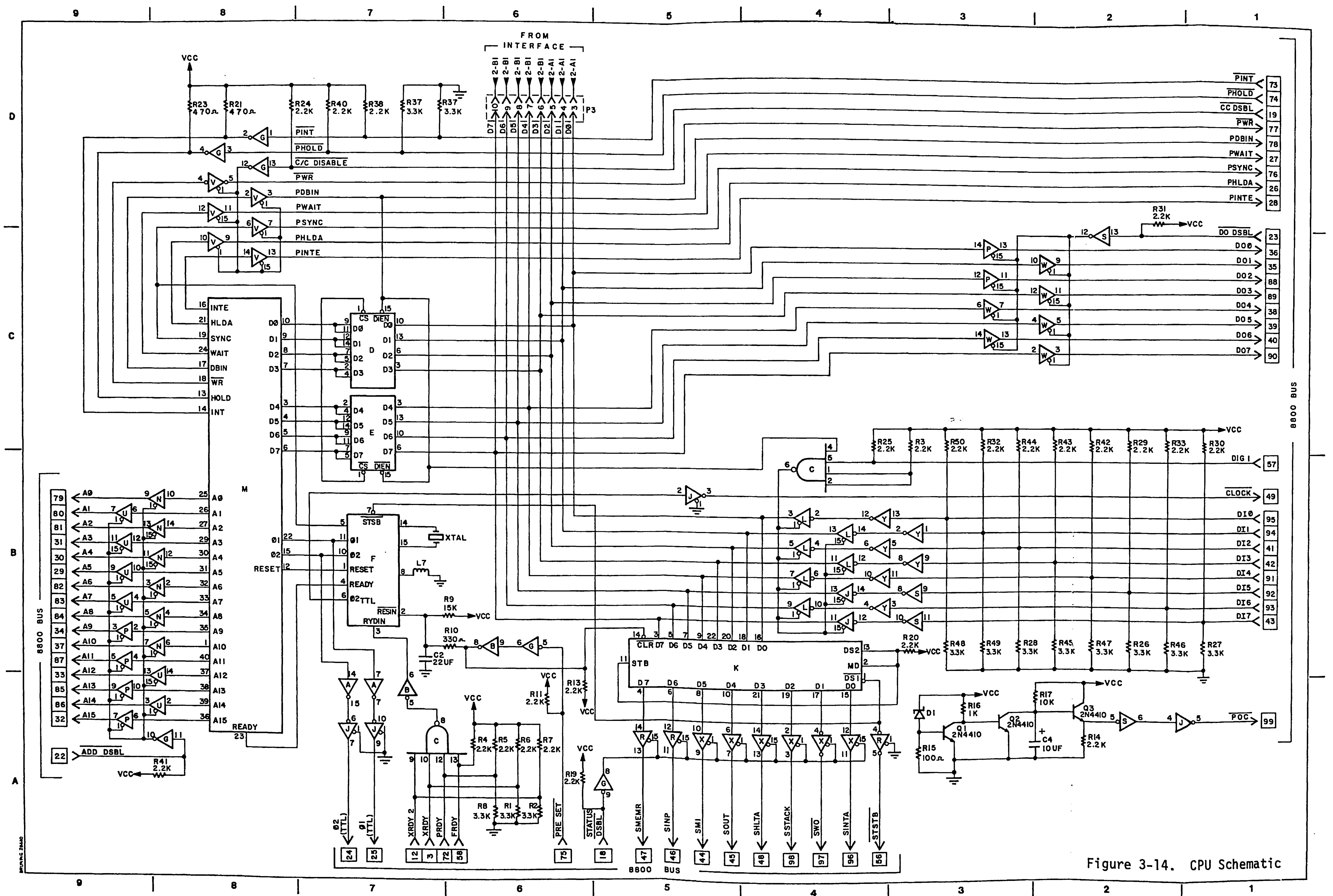


Figure 3-14. CPU Schematic

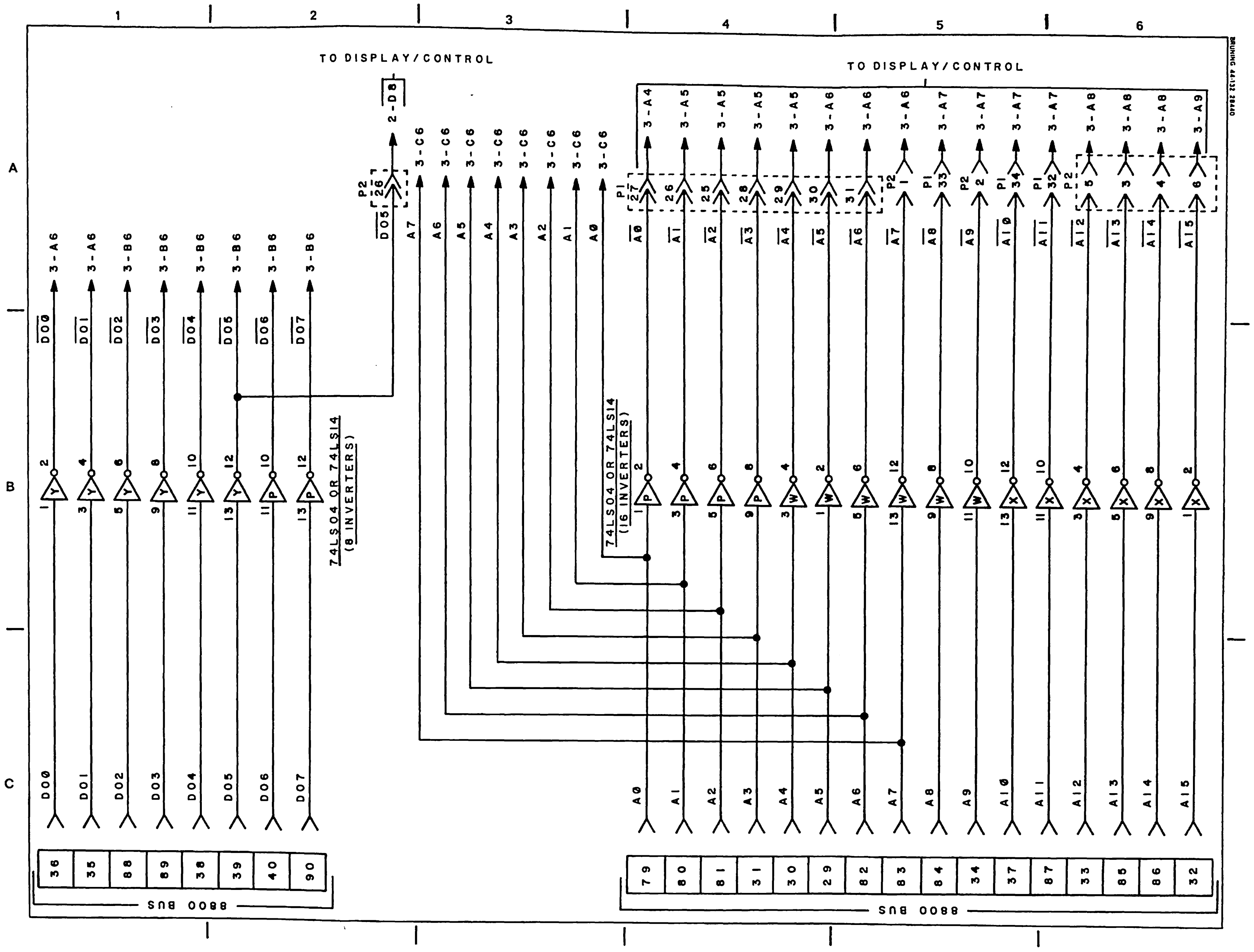

TO DISPLAY/CONTROL
8800 BUS
74LS04 OR 74LS14 (8 INVERTERS)
74LS04 OR 74LS14 (16 INVERTERS)
D00 D01 D02 D03 D04 D05 D06 D07
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15
P1
P2
2-D8
3-A4 3-A5 3-A6 3-A7 3-A8 3-A9 3-B6 3-C6

Figure 3-15. Interface Schematic (sheet 1 of 3)

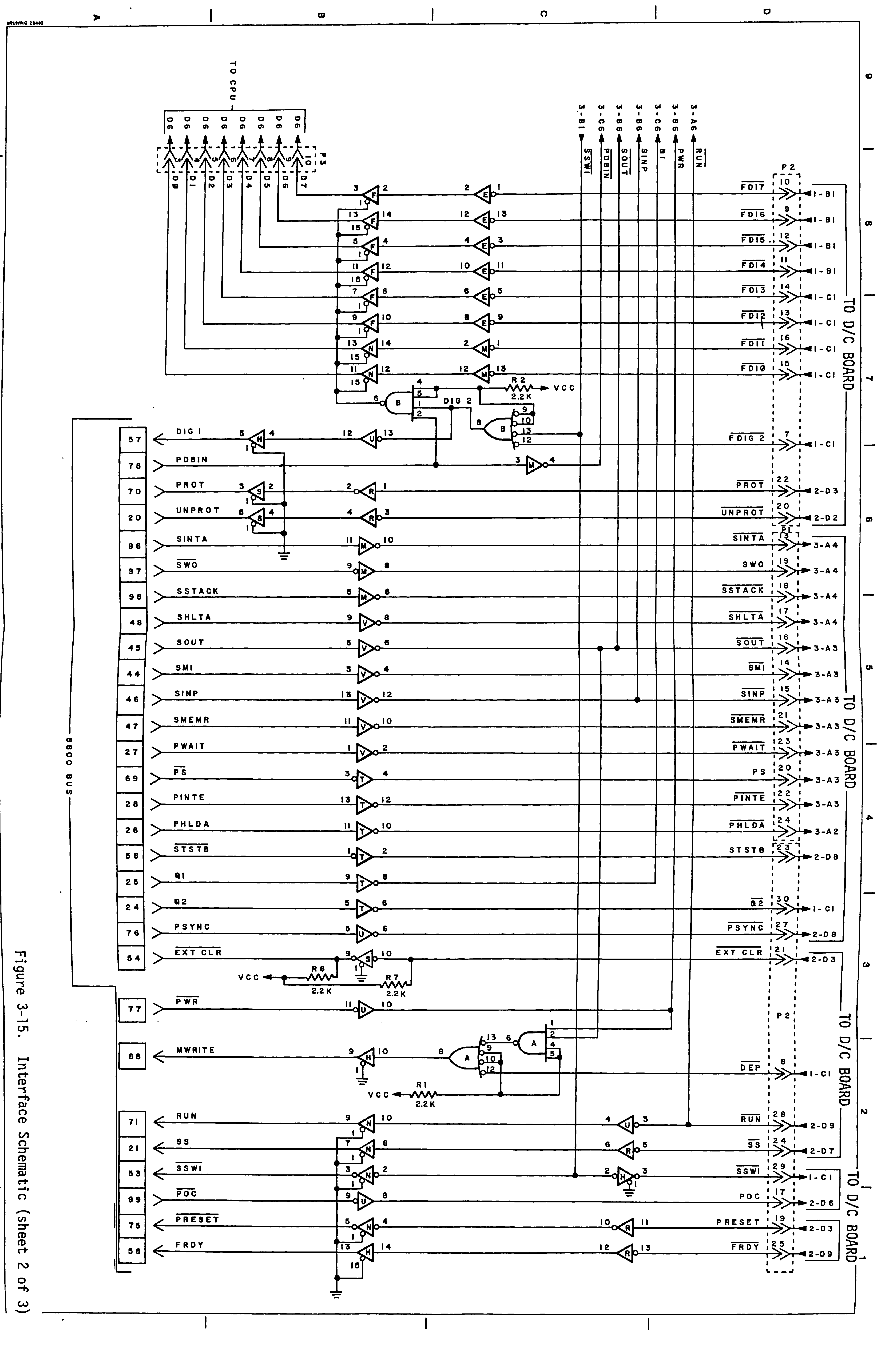
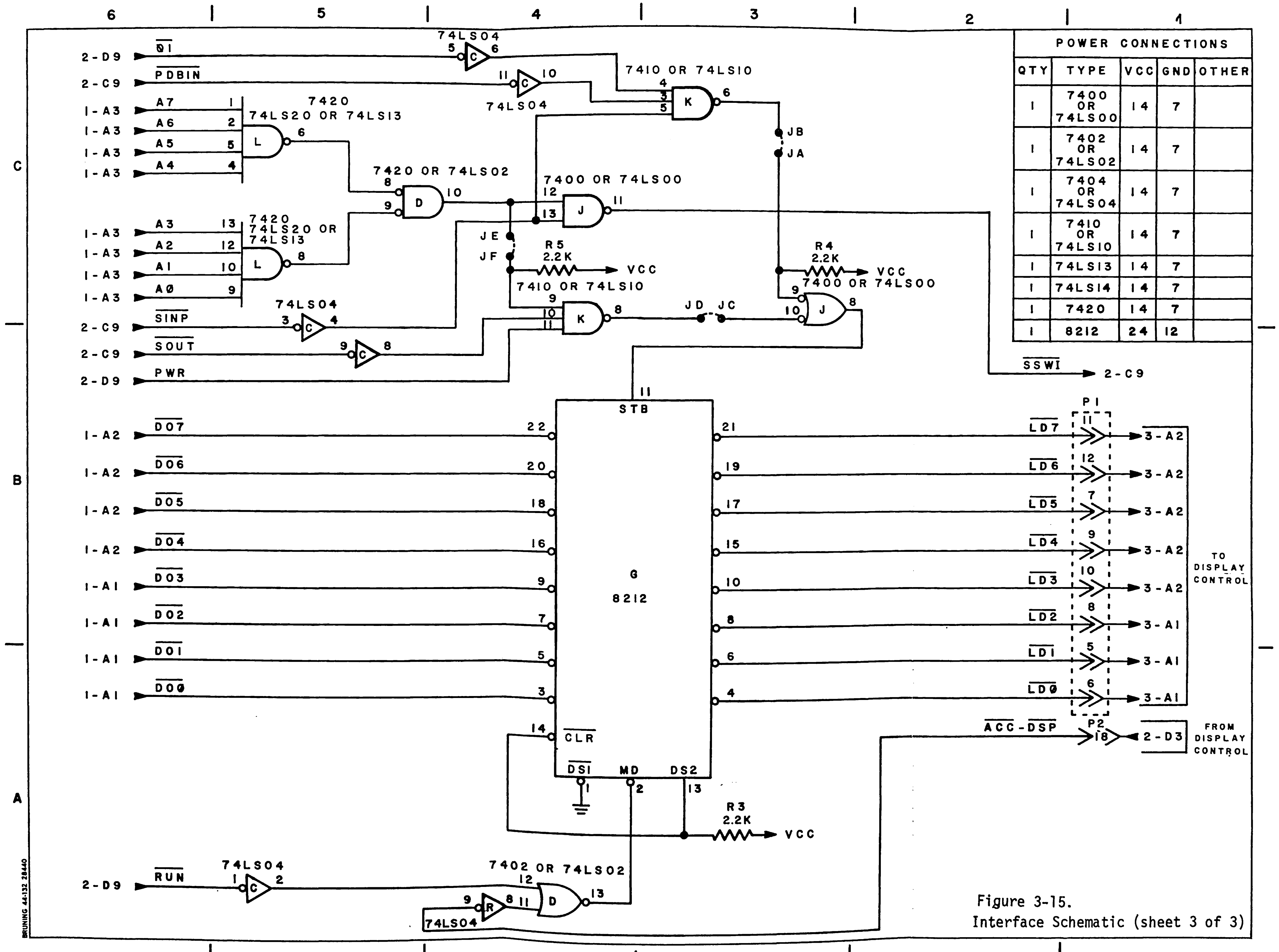
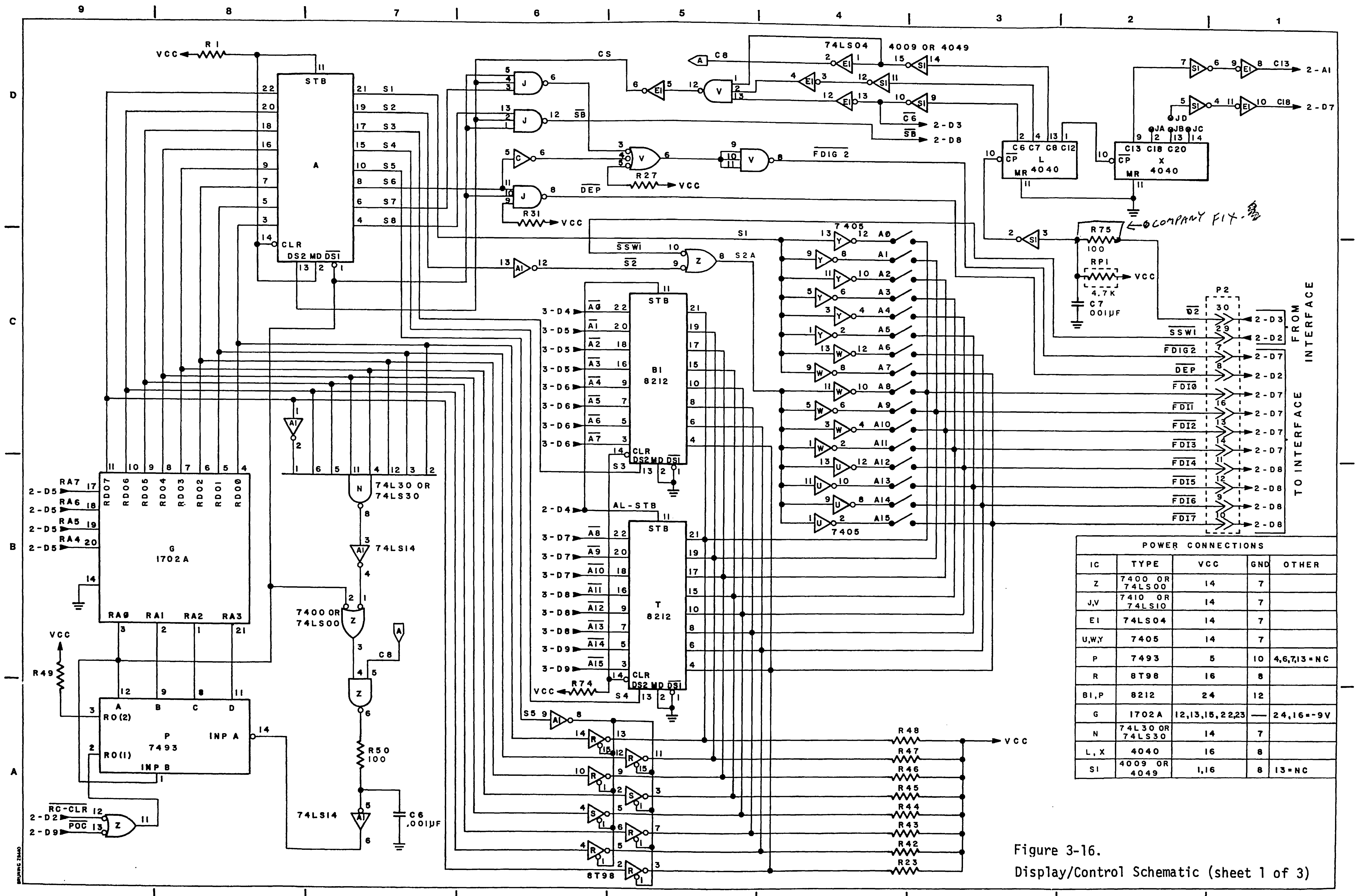


Figure 3-15. Interface Schematic (sheet 2 of 3)



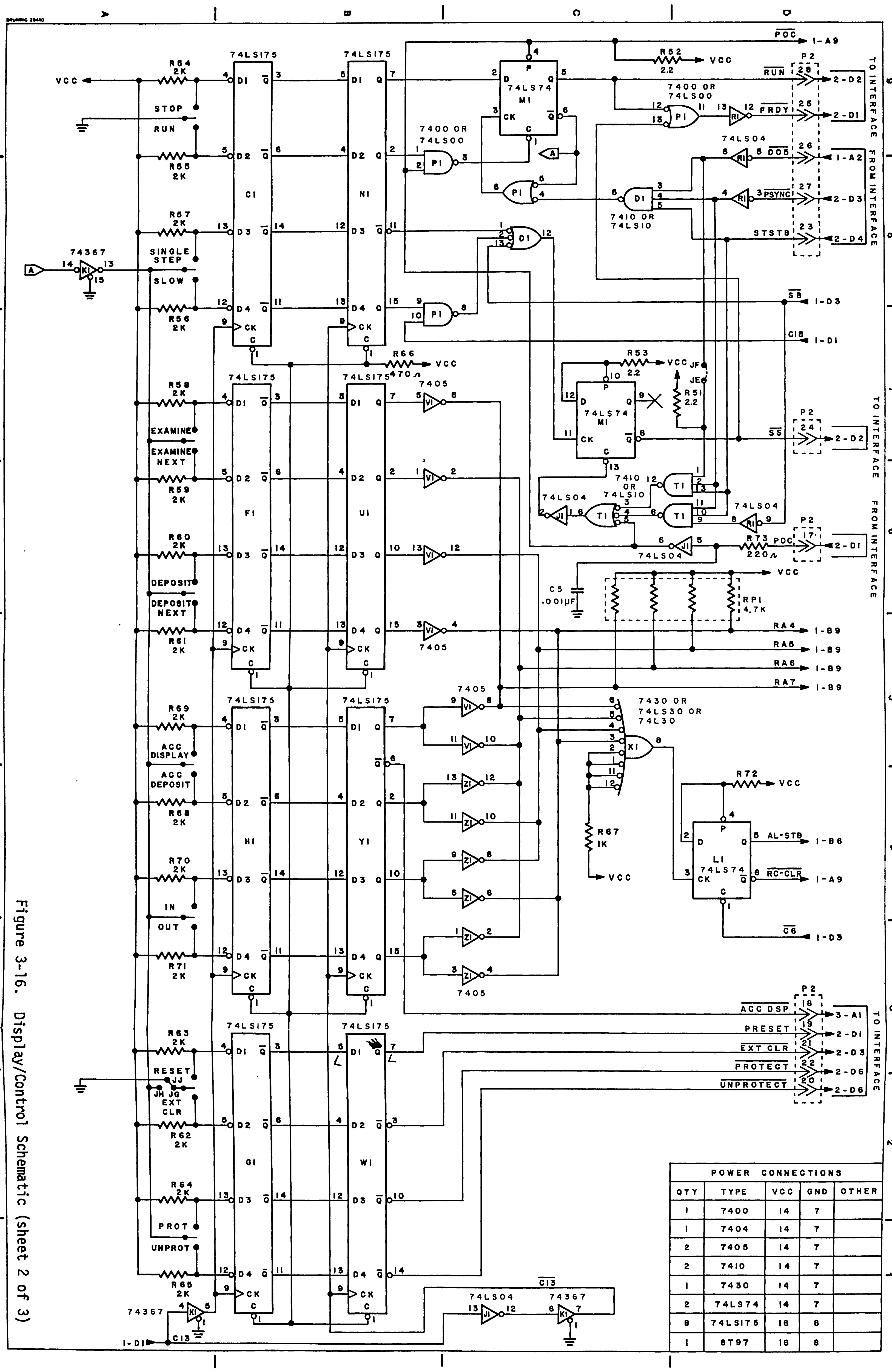
POWER CONNECTIONS				
QTY	TYPE	VCC	GND	OTHER
1	7400 OR 74LS00	14	7	
1	7402 OR 74LS02	14	7	
1	7404 OR 74LS04	14	7	
1	7410 OR 74LS10	14	7	
1	74LS13	14	7	
1	74LS14	14	7	
1	7420	14	7	
1	8212	24	12	

Figure 3-15.
Interface Schematic (sheet 3 of 3)



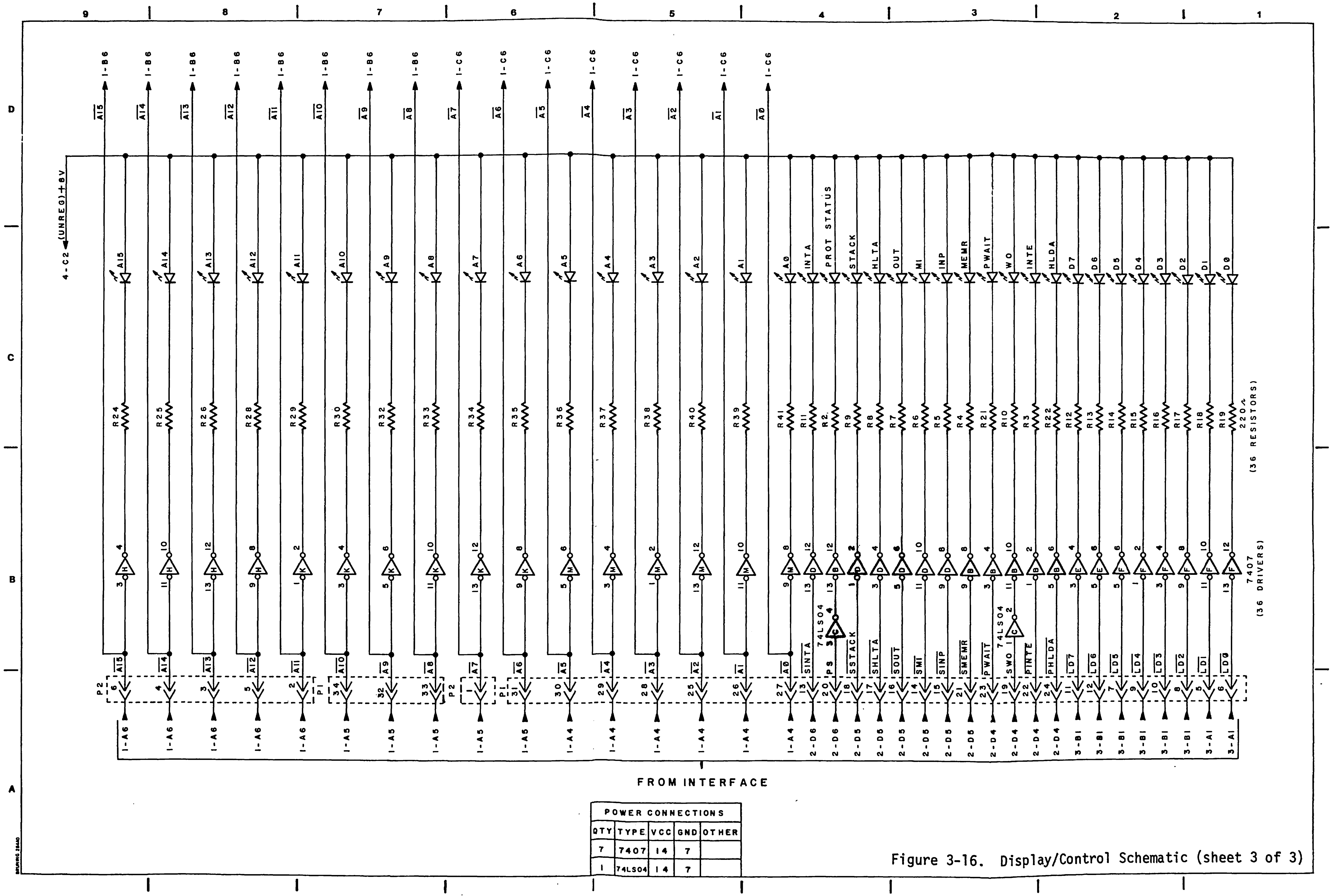
POWER CONNECTIONS				
IC	TYPE	VCC	GND	OTHER
Z	7400 OR 74LS00	14	7	
J,V	7410 OR 74LS10	14	7	
E1	74LS04	14	7	
U,W,Y	7405	14	7	
P	7493	5	10	4,6,7,13 = NC
R	8T98	16	8	
B1,P	8212	24	12	
G	1702A	12,13,15,22,23	—	24,16 = -9V
N	74L30 OR 74LS30	14	7	
L, X	4040	16	8	
S1	4009 OR 4049	1,16	8	13 = NC

Figure 3-16.
Display/Control Schematic (sheet 1 of 3)



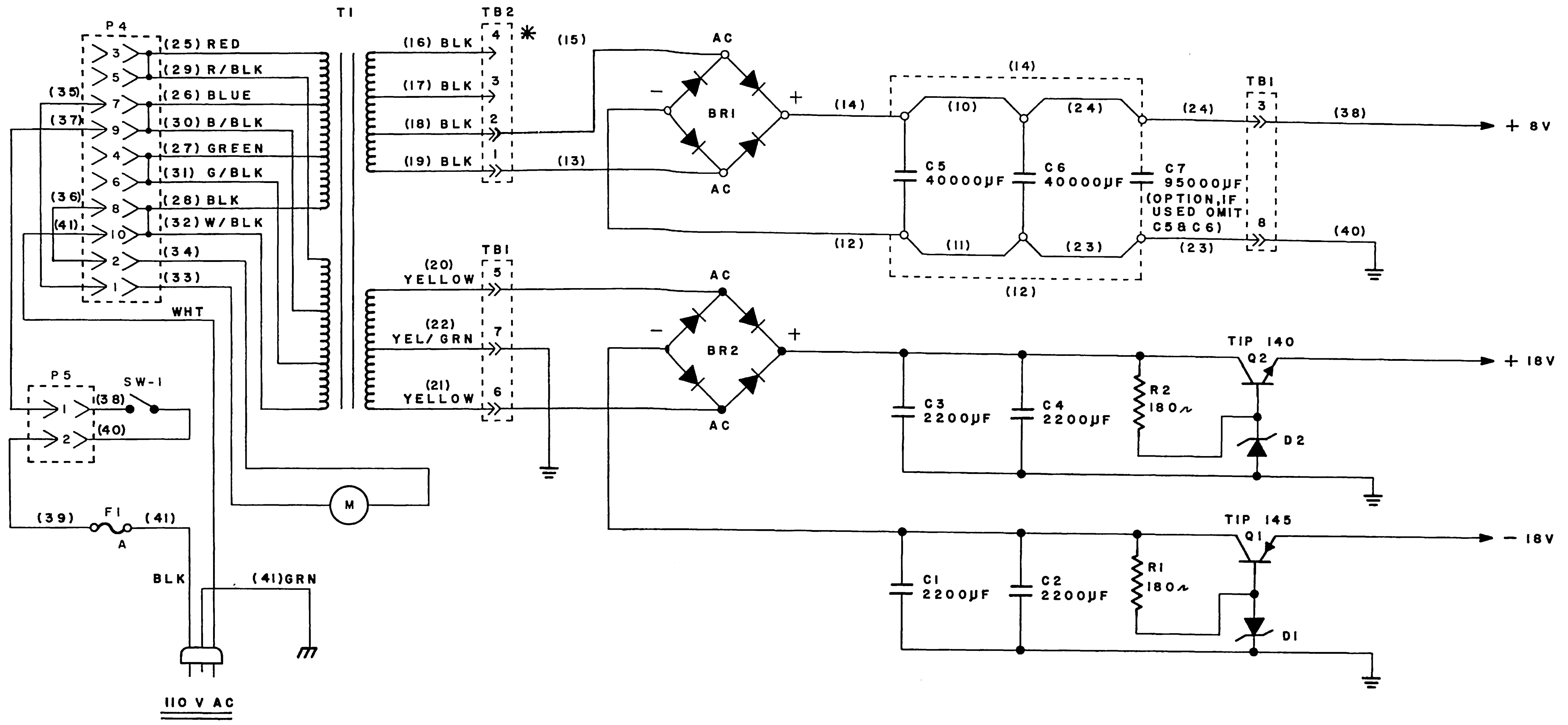
POWER CONNECTIONS				
QTY	TYPE	VCC	GND	OTHER
1	7400	14	7	
1	7404	14	7	
2	7405	14	7	
2	7410	14	7	
1	7430	14	7	
2	74LS74	14	7	
8	74LS175	16	8	
1	8T97	16	8	

Figure 3-16. Display/Control Schematic (sheet 2 of 3)



POWER CONNECTIONS				
QTY	TYPE	VCC	GND	OTHER
7	7407	14	7	
1	74LS04	14	7	

Figure 3-16. Display/Control Schematic (sheet 3 of 3)



90 V
// GREEN & GRN./BLK. TO PIN 1
// BLACK & WHT./BLK. TO PIN 2

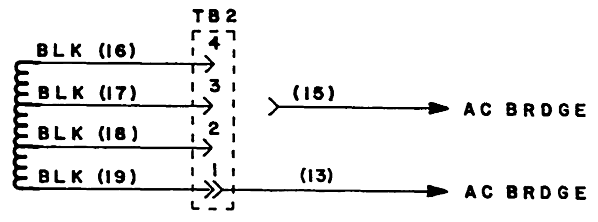
110 V
// BLUE & BLUE/BLK. TO PIN 1
// BLACK & WHT./BLK. TO PIN 2

130 V
// RED & RED/BLK. TO PIN 1
// BLACK & WHT./BLK. TO PIN 2

180 V
IN 1 WHT./BLK.
IN 2 GREEN
JUMP
BLACK & GREEN/BLK.

220 V
IN 1 WHT./BLK.
IN 2 BLUE
JUMP
BLACK & BLUE/BLK.

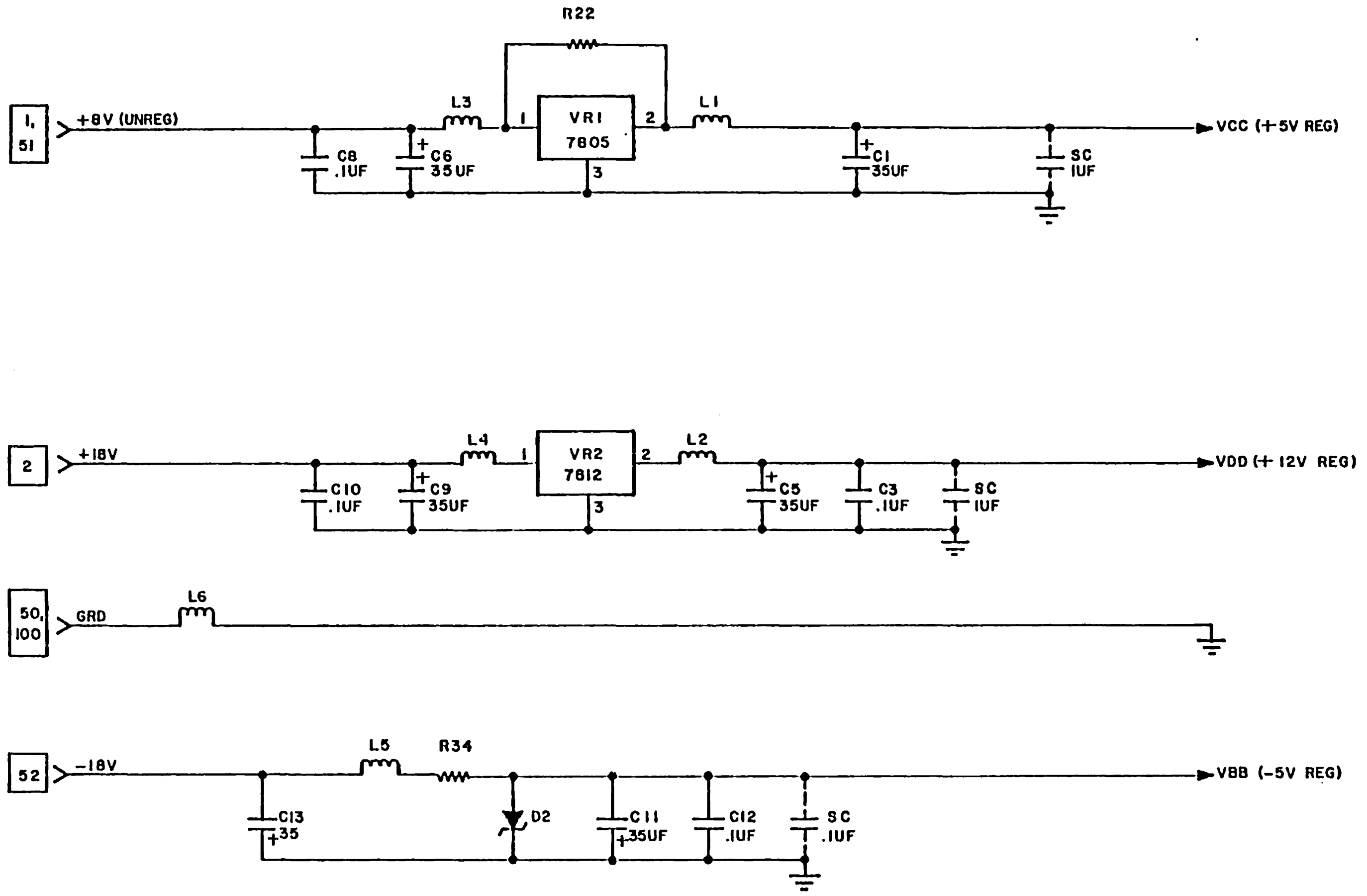
260 V
IN 1 WHT./BLK.
IN 2 RED
JUMP
BLACK & RED/BLACK



*

AMPS	TB2
0 - 4	pin 2
4 - 9	pin 3
9 - 18	pin 4

Figure 3-17. Power Supply Board Schematic



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	74368 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 3-18. CPU Voltage Regulator Schematic

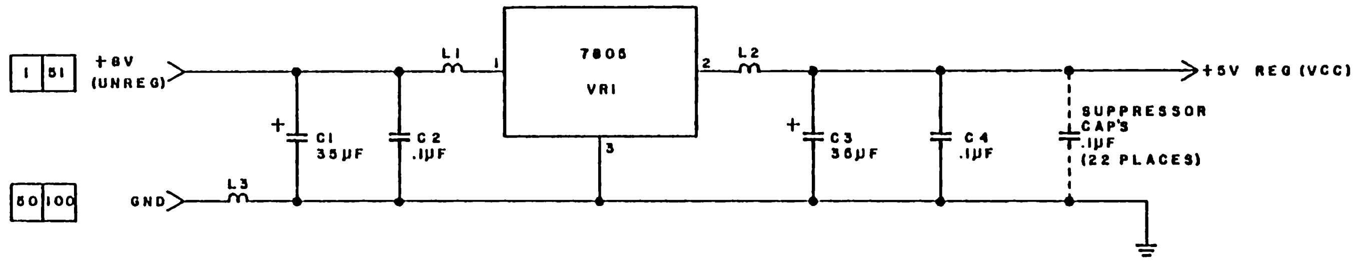


Figure 3-19. Interface Voltage Regulator Schematic

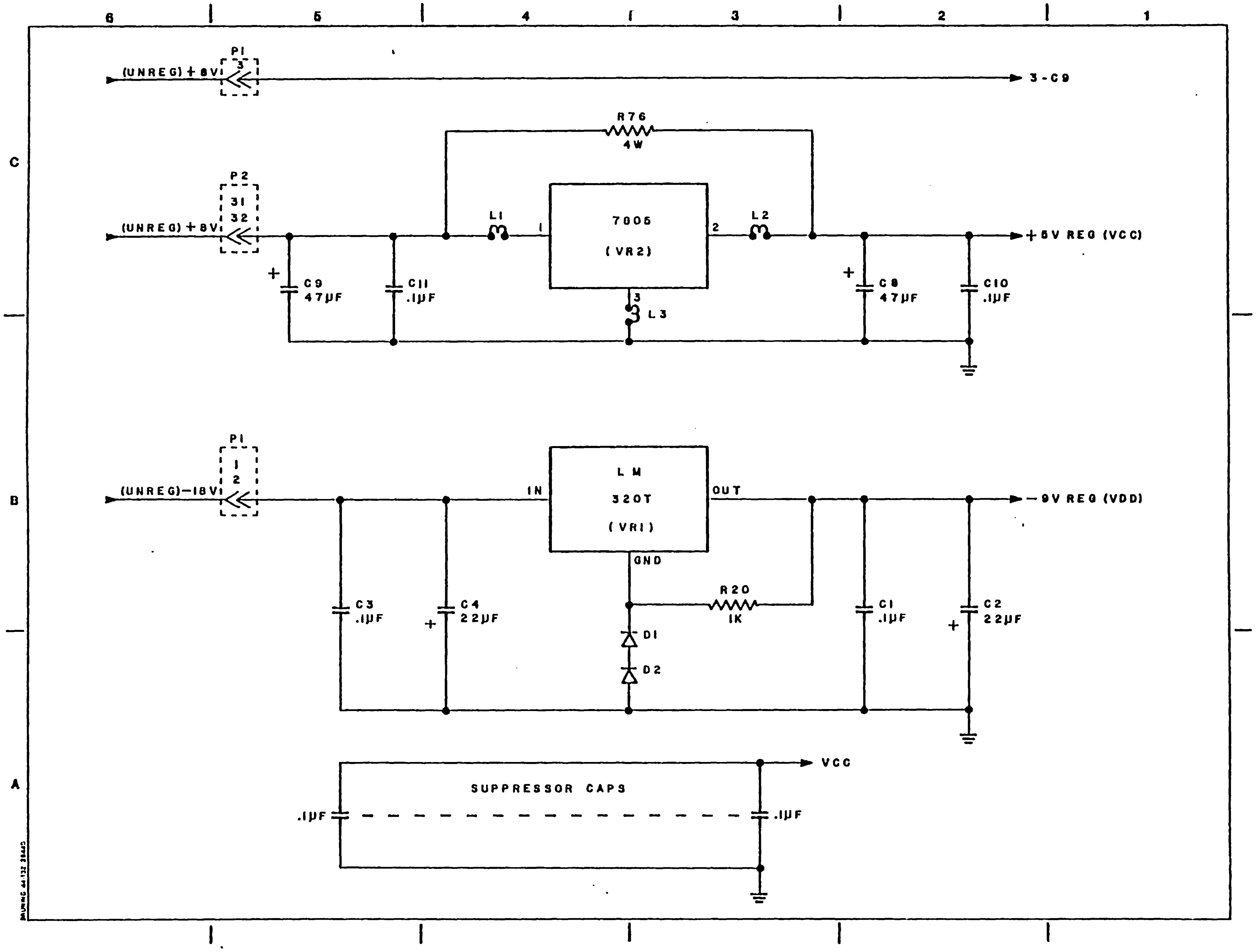


Figure 3-20. Display/Control Voltage Regulator Schematic

altair 8800b

SECTION IV
TROUBLESHOOTING

PREFACE

Section IV is designed to aid the user in pinpointing trouble areas and correcting problems that may be encountered with the Altair 8800b computer. The text that follows contains detailed instructions that should help in locating and correcting most problems. However, if the malfunction(s) cannot be rectified, send the unit to the MITS Repair Department or your local Altair dealer.

Section IV is divided into five major sections :

4-1) Introduction to Troubleshooting which contains general procedures that should always be followed, and IC static level charts showing the proper indications for the most common trouble areas;

4-2) Visual Inspection which contains procedures for locating problems caused by improper assembly;

4-3) Preliminary Check which contains tests for voltages and waveforms;

4-4) Non-PROM Related Switch Problems which concerns the RUN/STOP, SINGLE STEP/SLOW, RESET/EXTERNAL CLEAR and PROTECT/UNPROTECT switches;

4-5) PROM Related Switch Problems which deals with the EXAMINE/ EXAMINE NEXT, DEPOSIT/DEPOSIT NEXT, ACCUMULATOR DISPLAY/ ACCUMULATOR LOAD and IN/OUT switches.

Sections 4-3, 4-4 and 4-5 are presented in chart form, indicating the testing instructions, the correct indication, the incorrect indication and the procedures for remedying the problem.

Before beginning the actual troubleshooting procedures, the Theory of Operation and Section 4-1 should be reviewed. Refer to these portions of the manual when necessary.

An oscilloscope and an inexpensive multimeter will be needed to perform these troubleshooting procedures. The oscilloscope should be used to detect and measure pulses; the multimeter should be used to check voltage levels and continuity.

4-1. INTRODUCTION TO TROUBLESHOOTING

A. Basic Troubleshooting Procedures

Paragraphs 1, 2, 3 and 4 contain general instructions for testing ICs, diodes, transistors and bridge rectifiers, respectively. These procedures should be followed each time the instructions (in the tables that follow) specify that one of the above mentioned components be checked.

1. ICs
   a. With a voltmeter (or oscilloscope), check the IC pin for the proper voltage level or pulse. Make sure that the voltmeter is touching only one pin at a time; if the voltmeter should come in contact with more than one pin, erroneous readings and shorts may occur. (Note: Because the entire system is based upon the 8080 microprocessor chip, IC M on the CPU board, be especially careful when checking this component.) If the correct voltage is not present at the IC:
      1. Use the schematic to trace the signal back to its original source, checking for proper logic operation at each gate.
      2. Visually inspect the area surrounding the IC for solder bridges or opens.
   b. Never assume that when a signal leaves its source it will always reach its destination. Check for continuity with an ohmmeter (set at X1K ohms or higher to protect the ICs from the ohmmeter's current). If opens in the lands are found, solder over them.
   c. Check for power (Vcc) and Ground at the IC. Several of the schematics (in the Theory of Operation section) contain charts indicating the Vcc and Ground pins for each IC. If Vcc and Ground are present, test the IC according to the steps below.
      1) For ICs with sockets:
         a) Turn power off and remove the IC from its socket.
         b) Bend the suspected output pin up and reinstall the IC into its socket.
         c) Turn power on and check for proper logic operation.

NOTE

Removing an IC pin from its socket or from the board may change the IC's input level. When checking for proper logic, refer to the truth tables (pages 3-5 through 3-8) associated with that type of gate.

d) If the IC does not operate properly, replace it. If it does operate properly, bend the pin back and reinsert it into the socket. Look for a solder short or bridge and repair as necessary.

2) <u>For ICs without sockets</u>:

a) Turn power off and cut the suspected IC pin where it meets the component side of the board.

b) Bend the pin up and turn power on.

c) Check for proper logic operation (as shown in the appropriate truth table).

d) If the IC does not operate properly, replace it. If it does operate properly, resolder the pin to the board and look for a solder short or bridge. Repair as necessary.

NOTE

If an IC without a supplied socket needs to be replaced, you may wish to install a good-quality socket with it. Because sockets don't have to be removed from the board in order to test the IC, installation of sockets will aid in future troubleshooting and will prevent wear and tear on the board.

2. Diodes

Diodes can be easily tested with an ohmmeter set at X100 ohms. Turn power off and unsolder one lead from the board. To forward bias the diode, place the ohmmeter's positive lead on the diode's anode lead and the ohmmeter's negative lead on the diode's cathode lead. (The cathode lead is on the side marked with a bar.) The ohmmeter should show a LOW reading (15-300 ohms). To reverse bias the diode, transpose the ohmmeter's leads and check for a HIGH resistance reading (above 1K ohms). If the diode's readings do not correspond with the readings shown here, the diode should be replaced.

3. Transistors

Transistors can be tested with an ohmmeter set at X100 ohms. The following chart shows the correct readings for transistors with at least two leads removed from the board. Refer to the chart on page 5-9 in the Assembly section of the manual for lead identification, and compare the transistor's resistance to the resistance indicated in the chart below. Q1, Q2 and Q3 on the CPU board and Q2 on the Power Supply board are NPN transistors. Q1 on the Power Supply board is a PNP transistor.

Ohmmeter Lead Placement	Transistor Resistance NPN Transistors	PNP Transistors
Positive lead to emitter Negative lead to base	HIGH resistance	LOW resistance
Positive lead to base Negative lead to emitter	LOW resistance	HIGH resistance
Positive lead to base Negative lead to collector	LOW resistance	HIGH resistance
Positive lead to collector Negative lead to base	HIGH resistance	LOW resistance

HIGH = 2K ohms or higher
LOW = 1K ohms or lower		

4. Bridge Rectifiers

Unplug the chassis, remove the AC wires to TB1 and refer to the Diode testing instructions on page 4-6 to test the bridge rectifiers.

B. <u>Normal Output Voltage Levels</u>

1. TTL Gates (7400 Series ICs) and MOS ICs:

Condition	Voltage
Valid LOW	.8v or less
Valid HIGH	2v - 4v

An output in the range of .8v - 2v indicates a problem. (Note: Voltages can vary ±10%.)

2. Open Collector Gates

Open collector outputs, such as those of ICs Y, W, U, F, B and K on the Display/Control board, must be connected to +5v or +8v to operate properly. The outputs of ICs Y, W and U are tied to Vcc through resistors R41-R48 when the corresponding address switch is in the "up" position. When the switch is in the down position, the output will be disconnected from Vcc and will not allow signals to go through.

3. Tri-State Buffers (when enabled)

Condition	Voltage
Valid LOW	.8v or lower
Valid HIGH	2v or higher

An output in the range of .8v - 2v indicates a problem. (Note: es Voltages can vary $\pm 10\%$.)

When disabled, tri-state buffers will have various voltages at their outputs.

C. Static Levels

1. IC Levels

Table 4-1, starting on page 4-9, shows the proper static levels of the most common problem areas, assuming the computer is in a "stopped" state (M1, MEMR and WAIT).

Table 4-1. Static Levels of the Most Common Problem Areas

Board	Schematic	IC	Pin #	Static Level
Display/Control	3-16, sheet 1 of 3	G	17, 18, 19, 20	HIGH
		P	2, 8, 9, 11, 14	LOW
		P	12	HIGH
		N	8	LOW
		A	4, 6, 8, 10, 15, 17, 19, 21	LOW
		R	15, 1	HIGH
		S	1	HIGH
		Y	1, 3, 5, 11, 9, 13	LOW
		W	13, 9, 11, 5, 3, 1	LOW
		U	1, 9, 11, 13	LOW
		J	8, 12	HIGH
		Z	5	C8 (see waveform #5, page 4-30)
		V	8	HIGH
		J	4, 2	CS
		E1	6	CS
		A	13	CS
	3-16, sheet 2 of 3	C1, F1, H1, G1, N1, U1, Y1, W1	9	C13 (see waveform #4, page 4-29)
		V1	6, 2, 4, 12, 8, 10	HIGH
		Z1	2, 4, 6, 8, 10, 12	HIGH
		K1	13	LOW
		M1	11	LOW
		M1	8, 13	HIGH
		L1	3, 5	LOW

Table 4-1 (continued)

Board	Schematic	IC	Pin #	Static Level
		L1	1	$\overline{C6}$ (see waveform #6, page 4-30)
		R1	12	HIGH
		M1	2	LOW
		M1	1, 3, 5	HIGH
Interface	3-15, sheet 3 of 3	G	2, 13, 14	HIGH
		G	1, 11	LOW
		K	6	HIGH
		D	10	LOW
		J	11	HIGH
		K	8	HIGH
	3-15, sheet 2 of 3	B	6, 2, 12, 13	HIGH
		E	2, 4, 6, 8, 10, 12	LOW
		M	2, 12	LOW
		A	12, 13, 6, 2	HIGH
		A	1, 8	LOW
		N	6, 10	LOW
		N	4, 2	HIGH
		H	14	LOW
CPU	3-14	C	6, 13	LOW
		C	8, 4	HIGH
		M	23, 12	LOW
		D, E	15	HIGH
		F	7, 2	HIGH

2. Mother Board Static Levels

Table 4-2 shows the proper static levels of the mother board, assuming the computer is in a "stopped" state (M1, MEMR and WAIT). Note that the levels on the pins of the 8080a (IC M on the CPU board) are reflected on the mother board as well as the front panel LEDs. For example:

A HIGH level on pin 24 (WAIT) of IC M on the CPU board causes bus pin 27 to go HIGH, which in turn causes the WAIT light on the front panel to light.

HIGH pulses on pin 27 (address line A2) of IC M on the CPU board produce pulses on bus pin 81, which cause A2 on the front panel to light (dimly).

Table 4-2. Mother Board Static Levels

Bus #	Symbol	Name	Static Level
1	+8v	+8 volts	
2	+18v	+18 volts	
3	XRDY	EXTERNAL READY	HIGH
4	VI0	VECTORED INTERRUPT LINE #0	LOW
5	VI1	VECTORED INTERRUPT LINE #1	LOW
6	VI2	VECTORED INTERRUPT LINE #2	LOW
7	VI3	VECTORED INTERRUPT LINE #3	LOW
8	VI4	VECTORED INTERRUPT LINE #4	LOW
9	VI5	VECTORED INTERRUPT LINE #5	LOW
10	VI6	VECTORED INTERRUPT LINE #6	LOW
11	VI7	VECTORED INTERRUPT LINE #7	LOW
12*	XRDY2	Extra READY Line	HIGH
13-17	Not Used		
18	$\overline{\text{STA DSB}}$	$\overline{\text{STATUS DISABLE}}$	HIGH
19	$\overline{\text{C/C DSB}}$	$\overline{\text{COMMAND/CONTROL DISABLE}}$	HIGH
20**	UNPROT	UNPROTECT	LOW
21**	SS	SINGLE STEP	LOW
22	$\overline{\text{ADD DSB}}$	$\overline{\text{ADDRESS DISABLE}}$	HIGH
23	$\overline{\text{DO DSB}}$	$\overline{\text{DATA OUT DISABLE}}$	HIGH
24	Ø2	PHASE 2 CLOCK	See waveforms 2 and 3, page 4-26
25	Ø1	PHASE 1 CLOCK	See waveforms 2 and 3, page 4-26

Bus #	Symbol	Name	Static Level
26	PHLDA	HOLD ACKNOWLEDGE	LOW
27	PWAIT	WAIT	HIGH
28	PINTE	INTERRUPT ENABLE	LOW
29	A5	ADDRESS LINE #5	
30	A4	ADDRESS LINE #4	
31	A3	ADDRESS LINE #3	
32	A15	ADDRESS LINE #15	
33	A12	ADDRESS LINE #12	
34	A9	ADDRESS LINE #9	
35	DO1	DATA OUT LINE #1	
36	DO0	DATA OUT LINE #0	
37	A10	ADDRESS LINE #10	
38	DO4	DATA OUT LINE #4	
39	DO5	DATA OUT LINE #5	
40	DO6	DATA OUT LINE #6	
41	DI2	DATA IN LINE #2	
42	DI3	DATA IN LINE #3	
43	DI7	DATA IN LINE #7	
44	SM1	M1 (Instruction Fetch Cycle)	HIGH
45	SOUT	OUT (Output Write)	LOW
46	SINP	INP (Input Read)	LOW
47	SMEMR	MEMR (Memory Read)	HIGH
48	SHLTA	HLTA (Halt Acknowledge)	LOW
49	$\overline{\text{CLOCK}}$	$\overline{\text{CLOCK}}$	See Waveforms 2 and 3, page 4-28
50	GND	GROUND	
51	+8v	+8 volts	
52	-18v	-18 volts	
53**	$\overline{\text{SSW DSB}}$	$\overline{\text{SENSE SWITCH DISABLE}}$	HIGH
54	$\overline{\text{EXT CLR}}$	$\overline{\text{EXTERNAL CLEAR}}$	HIGH
55	RTC	REAL TIME CLOCK	
56*	$\overline{\text{STSTB}}$	$\overline{\text{STATUS STROBE}}$	HIGH
57**	DIG1	DIGITAL #1	HIGH
58**	FRDY	Front Panel READY	LOW
59-67	Not Used		
68	MWRT	MEMORY WRITE	LOW
69	$\overline{\text{PS}}$	$\overline{\text{PROTECT STATUS}}$	HIGH

Bus #	Symbol	Name	Static Level
70**	PROT	PROTECT	LOW
71**	RUN	RUN	LOW
72	PRDY	READY	HIGH
73	$\overline{\text{PINT}}$	$\overline{\text{INTERRUPT REQUEST}}$	HIGH
74	$\overline{\text{PHOLD}}$	$\overline{\text{HOLD}}$	HIGH
75	$\overline{\text{PRESET}}$	$\overline{\text{RESET}}$	HIGH
76	PSYNC	SYNC	LOW
77	$\overline{\text{PWR}}$	$\overline{\text{WRITE}}$	HIGH
78	PDBIN	DATA BUS IN	HIGH
79	A0	ADDRESS LINE #0	
80	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	DO2	DATA OUT LINE #2	
89	DO3	DATA OUT LINE #3	
90	DO7	DATA OUT LINE #7	
91	DI4	DATA IN LINE #4	
92	DI5	DATA IN LINE #5	
93	DI6	DATA IN LINE #6	
94	DI1	DATA IN LINE #1	
95	DI0	DATA IN LINE #0	
96	SINTA	INTA (Interrupt Request Acknowledge)	LOW
97	$\overline{\text{SWO}}$	$\overline{\text{WO}}$ (Write Operation)	HIGH
98	SSTACK	STACK	LOW
99	$\overline{\text{POC}}$	POWER ON CLEAR	HIGH
100	GND	GROUND	

* = Not used in 8800a system.

** = Not used in 8800b Turnkey system.

Note: If a static level is not indicated, the signal can be either HIGH or LOW.

4-2. VISUAL INSPECTION

A. Component Inspection

The first step in troubleshooting is to carefully examine each board for solder bridges, open lands, misplaced components, etc. A thorough inspection of this kind will eliminate one possibility for errors and will allow troubleshooting efforts to be concentrated elsewhere. Carefully check each board using the list below:

1. Look for solder bridges.
2. Look for leads that have not been soldered.
3. Look for cold solder connections (cold solder connections do not have a "shiny" appearance).
4. Examine the board's lands for "hairline opens" or bridges."
5. Check the ICs for proper pin placement and good socket connections.
6. Examine the electrolytic and tantalum capacitors for proper polarity.
7. Examine the diodes for proper polarity.
8. Examine the LEDs for proper polarity.
9. Check the color codes on all resistors.

B. Wiring Inspection

CAUTION The computer should be unplugged for this check.

1. Referring to Figure 5-50 on page 5-58 in the Assembly section of the manual, check for incorrect wiring on the mother board.
2. With an ohmmeter, check the power supply wiring on the terminal block (TB1). Check for resistance (about 100 ohms) between pins 2 and 7, 10 and 7, 1 and 7, 2 and 10, 2 and 1 and 1 and 10. If a reading of less than 10 ohms appears, recheck the wiring. Also check continuity from mother board bus pins 1, 2, 52 and 50 to corresponding terminal block pins 2, 10, 1 and 7. If a reading of more than 100 ohms appears, inspect the wiring from the mother board to TB1.

4-3. PRELIMINARY CHECK

The procedures outlined in Section 4-3 are general tests that should be made before going on to the specific problems presented in Sections 4-4 and 4-5. Follow the instructions in the order in which they are given, and always complete each step before going on to the next.

1. Before installing the boards and applying power to the computer, use an ohmmeter to check the resistance of the edge connectors on the mother board. Test the consecutively numbered pins down each row (1, 2, 3 . . . etc.), then cross check the pins (1-51, 2-52 . . . etc.). A LOW resistance reading should appear at pins 1, 50, 51 and 100. If a LOW reading appears at any other location, examine the back of the board for solder bridges or etching errors.
2. Turn the computer on and check for the following voltages on the Power Supply board's terminal block (TB1). See page 5-58 in the Assembly section of the manual for pin locations.

Pin # only	Voltage
2, ③ 4	+8v to +10v (unregulated)
10	+16v to +18v (pre-regulated)
1	-16v to -18v (pre-regulated)
7, 8	Ground

WARNING

When testing components on the Power Supply board, be extremely careful not to touch the AC wiring. Always unplug the chassis when testing continuity or replacing components.

a. If the +8 voltage is absent from pins 2, 3 or 4 of TB1, check for AC at pins 1 and 2 of TB2. If absent, unplug the chassis and check continuity and wiring at connector P4. Also check the fuse and the wiring to the AC cord. Plug in the chassis. If AC is present at pins 1 and 2 of TB2, check the wiring from TB2 to BR1 and from BR1 to TB1. If AC is present at BR1, but no output voltage appears across the "+" and "-" pins of BR1, BR1 is probably defective and should be replaced.