

**Chapter 4**  
**8080A**  
**Functional**  
**Description**

**MOS**

**8080A**

**MOS**

**8080A**



## CHAPTER 4 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 1-1), 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<sub>0</sub>-D<sub>7</sub>). Memory and peripheral device addresses are transmitted over a separate 16-

bit 3-state Address Bus (A<sub>0</sub>-A<sub>15</sub>). Six timing and control outputs (SYNC, DBIN, WAIT, 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 ( $\phi_1$  and  $\phi_2$ ) are accepted by the 8080.

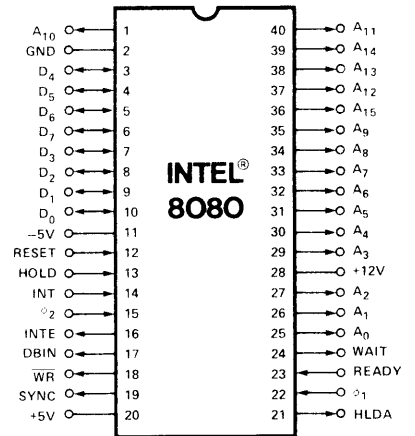
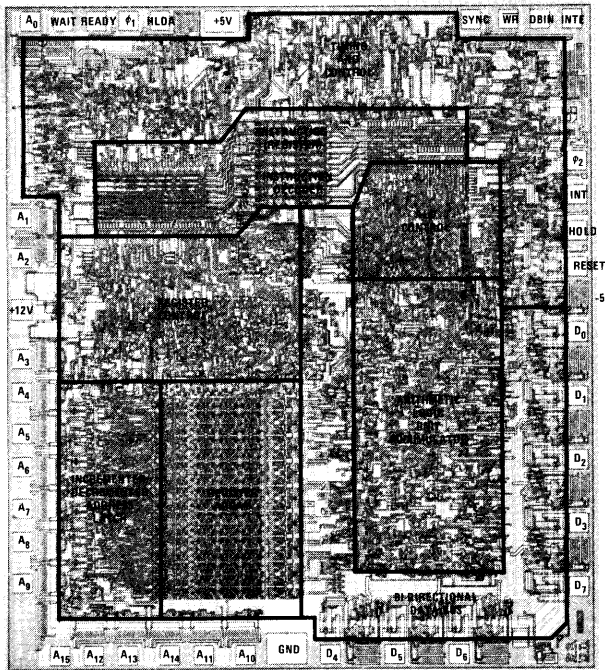


Figure 4-1. 8080 Photomicrograph With Pin Designations

## 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 4-2 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 next program instruction and is incremented auto-

matically 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 four register pairs and drives the 16 address output buffers (A<sub>0</sub>-A<sub>15</sub>), 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.

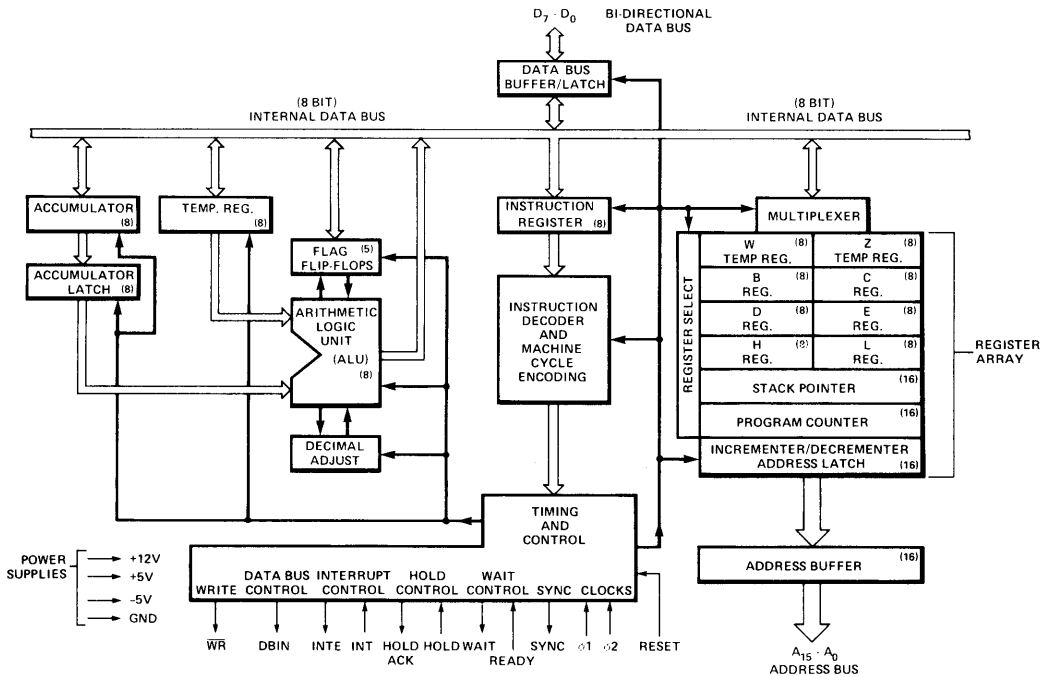


Figure 4-2. 8080 CPU Functional Block Diagram

## 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 Section 5).

## 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<sub>0</sub> through D<sub>7</sub>). 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.

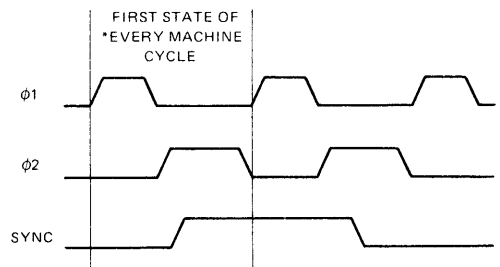
During the input mode, data from the external data bus is transferred to the internal bus. The internal bus is precharged at the beginning of each internal state, except for the transfer state (TW and T<sub>3</sub>—described later in this chapter).

## 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 Section 5).

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  $\phi_1$  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  $\phi_1$  and  $\phi_2$ , are furnished by external circuitry. It is the  $\phi_1$  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  $\phi_2$ , as shown in Figure 4-3.



\*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 4-3.  $\phi_1$ ,  $\phi_2$  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 eight-teen 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 (IN) 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 than 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<sub>0</sub>-D<sub>7</sub>), during the SYNC interval. This data should be saved in latches, and used to develop control signals for external circuitry. Table 4-1 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<sub>1</sub> status bit (D<sub>5</sub>), 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 4-1 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 six active states (referred to as T<sub>1</sub>, T<sub>2</sub>, T<sub>3</sub>, T<sub>4</sub>, T<sub>5</sub> or T<sub>W</sub>). 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 4-4 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, 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  $\phi_1$  clock. Figure 4-5 shows the timing relationships in a typical FETCH machine cycle. Events that occur in each state are referenced to transitions of the  $\phi_1$  and  $\phi_2$  clock pulses.

The SYNC signal identifies the first state (T<sub>1</sub>) in every machine cycle. As shown in Figure 4-5, the SYNC signal is related to the leading edge of the  $\phi_2$  clock. There is a delay (t<sub>DC</sub>) between the low-to-high transition of  $\phi_2$  and the positive-going edge of the SYNC pulse. There also is a corresponding delay (also t<sub>DC</sub>) between the next  $\phi_2$  pulse and the falling edge of the SYNC signal. Status information is displayed on D<sub>0</sub>-D<sub>7</sub> during the same  $\phi_2$  to  $\phi_2$  interval. Switching of the status signals is likewise controlled by  $\phi_2$ .

The rising edge of  $\phi_2$  during T<sub>1</sub> also loads the processor's address lines (A<sub>0</sub>-A<sub>15</sub>). These lines become stable within a brief delay (t<sub>DA</sub>) of the  $\phi_2$  clocking pulse, and they remain stable until the first  $\phi_2$  pulse after state T<sub>3</sub>. 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<sub>RS</sub>) which occurs during the  $\phi_2$  pulse within state T<sub>2</sub> or T<sub>W</sub>. 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 4-5.

The processor responds to a wait request by entering an alternative state (T<sub>W</sub>) at the end of T<sub>2</sub>, rather than proceeding directly to the T<sub>3</sub> state. Entry into the T<sub>W</sub> 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  $\phi_1$  clock and occurs within a brief delay (t<sub>DC</sub>) of the actual entry into the T<sub>W</sub> 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  $\phi_2$  clock by a specified interval (t<sub>RS</sub>), in order to guarantee an exit from the T<sub>W</sub> state. The cycle may then proceed, beginning with the rising edge of the next  $\phi_1$  clock. A WAIT interval will therefore consist of an integral number of T<sub>W</sub> states and will always be a multiple of the clock period.





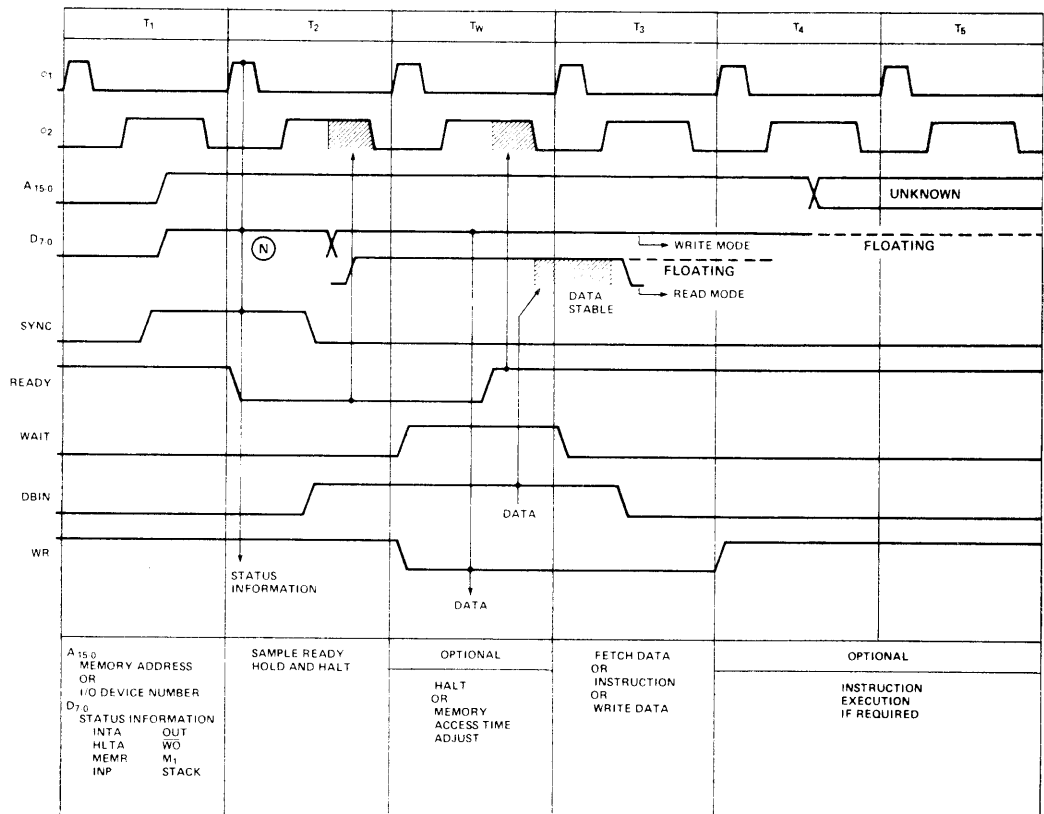
The events that take place during the  $T_3$  state are determined by the kind of machine cycle in progress. In a FETCH machine cycle, the processor interprets the data on its data bus as an instruction. During a MEMORY READ or a STACK READ, data on this bus is interpreted as a data word. The processor 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 4-7 illustrates the timing that is characteristic of a data input operation. As shown, the low-to-high transition of  $\phi_2$  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 " $\phi_1$ -data set-up" interval ( $t_{DS1}$ ), that precedes the falling edge of the  $\phi_1$  pulse defining state  $T_3$ , and the " $\phi_2$ -data set-up" interval ( $t_{DS2}$ ), that precedes the rising edge of  $\phi_2$  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  $\phi_2$  pulse. Data placed on these lines by memory or by other external devices will be sampled during  $T_3$ .

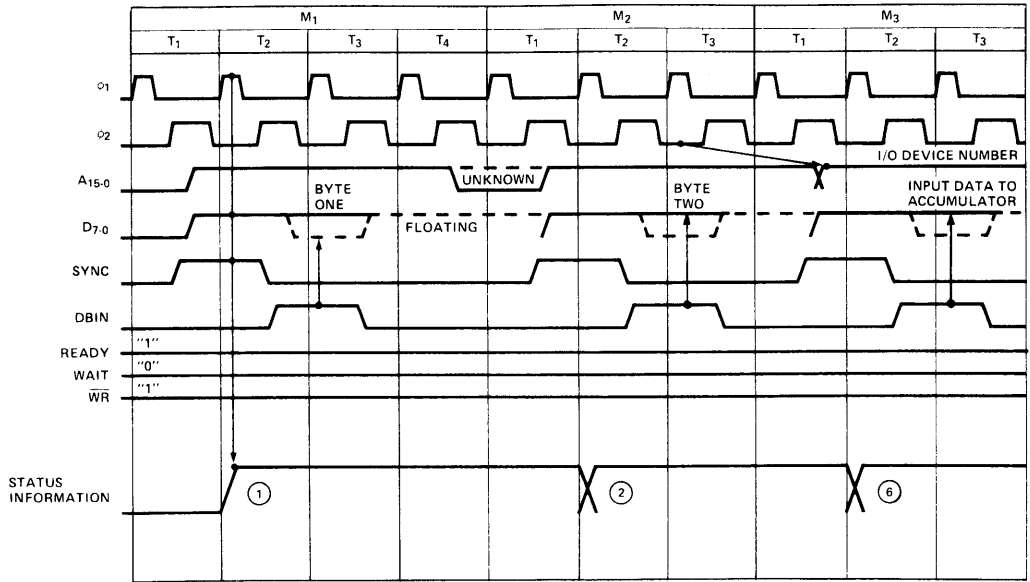
During the input of data to the processor, the 8080 generates a DBIN signal which should be used externally to enable the transfer. Machine cycles in which DBIN is available include: FETCH, MEMORY READ, STACK READ, and INTERRUPT. DBIN is initiated by the rising edge of  $\phi_2$  during state  $T_2$  and terminated by the corresponding edge of  $\phi_2$  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 4-7 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  $\phi_2$  within state  $T_2$  clears status information from the CPU's data lines, and loads in the data which is to be output to external devices. This substitution takes place within the



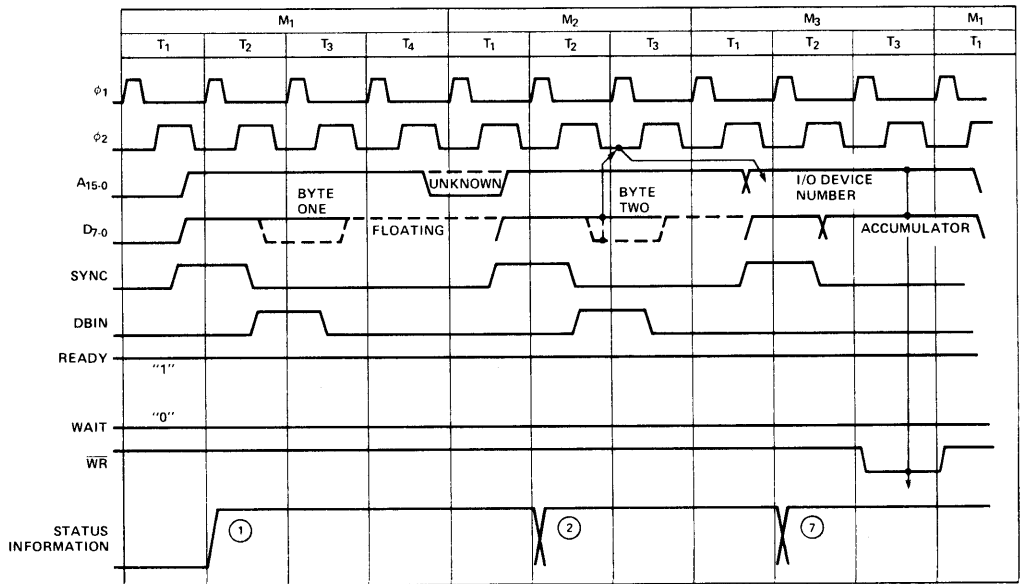
NOTE: (N) Refer to Status Word Chart on Page 4-6.

Figure 4-5. Basic 8080 Instruction Cycle



NOTE: (N) Refer to Status Word Chart on Page 4-6.

Figure 4-6. Input Instruction Cycle



NOTE: (N) Refer to Status Word Chart on Page 4-6.

Figure 4-7. Output Instruction Cycle

“data output delay” interval ( $t_{\text{OD}}$ ) following the  $\phi_2$  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  $\overline{\text{WR}}$  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  $\overline{\text{WR}}$  is referenced to the rising edge of the first  $\phi_1$  clock pulse following  $T_2$ , and occurs within a brief delay ( $t_{\text{DC}}$ ) of that event.  $\overline{\text{WR}}$  remains low until re-triggered by the leading edge of  $\phi_1$  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 neces-

sarily extend  $\overline{\text{WR}}$ , 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 (A15.0); status information is placed on Data Bus (D7.0).  |
| $T_2$                        | The CPU samples the READY and HOLD inputs and checks for halt instruction.  |
| $T_W$<br>(optional)          | Processor enters wait state if READY is low or if HALT instruction has been executed.   |
| $T_3$                        | An instruction byte (FETCH machine cycle), data byte (MEMORY READ, STACK READ, INPUT) 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. |
| $T_4$<br>$T_5$<br>(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 4-2. 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.

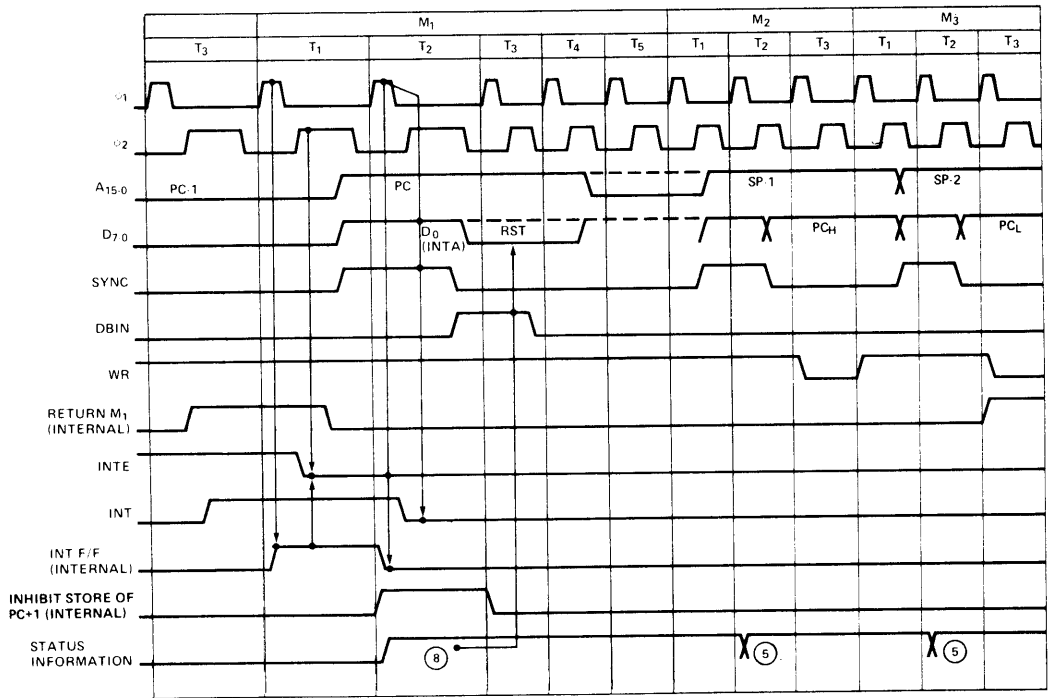
The interrupt (INT) input is asynchronous, and a request may therefore originate at any time during any instruction cycle. Internal logic re-clocks the external request, so that a proper correspondence with the driving clock is established. As Figure 4-8 shows, an interrupt request (INT) arriving during the time that the interrupt enable line (INTE) is high, acts in coincidence with the  $\phi_2$  clock to set the internal interrupt latch. This event takes place during the last state of the instruction cycle in which the request occurs, thus ensuring that any instruction in progress is completed before the interrupt can be processed.

The INTERRUPT machine cycle which follows the arrival of an enabled interrupt request resembles an ordinary FETCH machine cycle in most respects. The  $M_1$  status bit is transmitted as usual during the SYNC interval. It is accompanied, however, by an INTA status bit ( $D_0$ ) which acknowledges the external request. The contents of the program counter are latched onto the CPU's address lines during  $T_1$ , but the counter itself is not incremented during the INTERRUPT machine cycle, as it otherwise would be.

In this way, the pre-interrupt status of the program counter is preserved, so that data in the counter may be restored by the interrupted program after the interrupt request has been processed.

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

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



NOTE: (N) Refer to Status Word Chart on Page 4-6.

Figure 4-8. Interrupt Timing

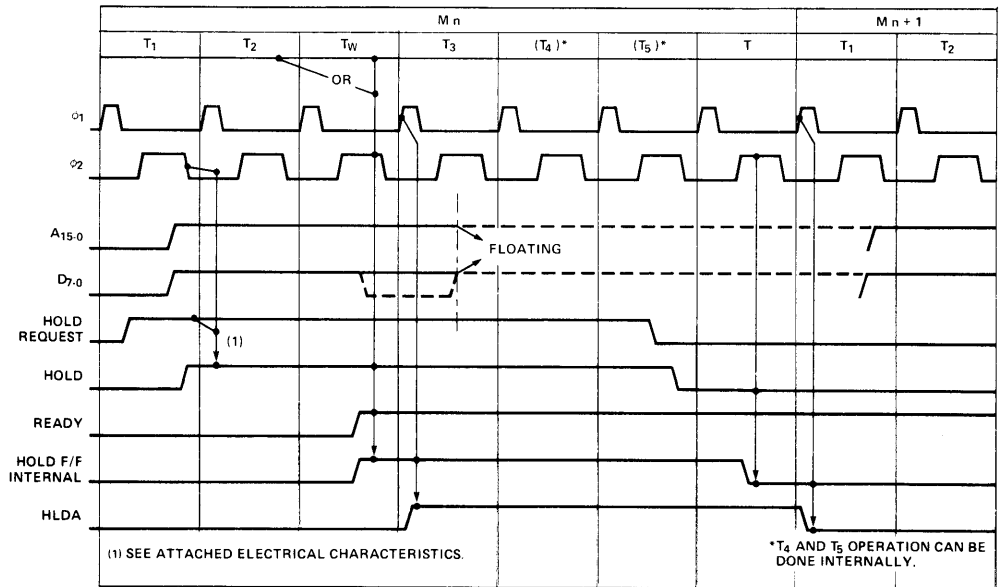


Figure 4-9. HOLD Operation (Read Mode)

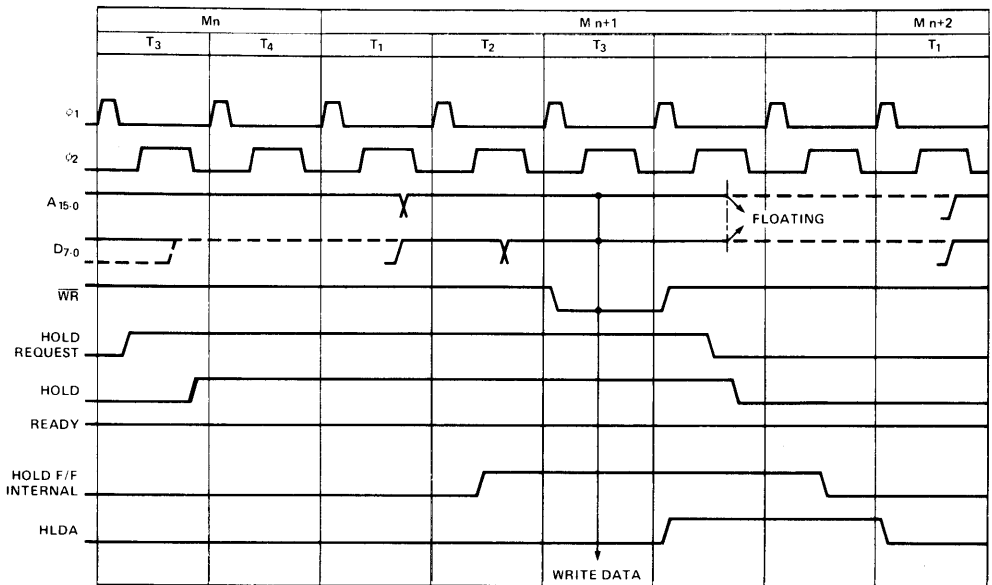


Figure 4-10. HOLD Operation (Write Mode)

## 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 output 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  $\phi_2$ .

Figures 4-9 and 4-10 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  $\phi_2$  clocks sets the internal hold latch. Setting the latch enables the subsequent rising edge of the  $\phi_1$  clock pulse to trigger the HLDA output as described below.

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 4-9). Otherwise, acknowledgement is deferred until the beginning of the state following  $T_3$  (see Figure 4-10). In both cases, however, the HLDA goes high within a specified delay ( $t_{DC}$ ) of the rising edge of the selected  $\phi_1$  clock pulse. Address and data lines are floated within a brief delay after the rising edge of the next  $\phi_2$  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 4-11. There are only three ways in which the 8080 can exit the halt state:

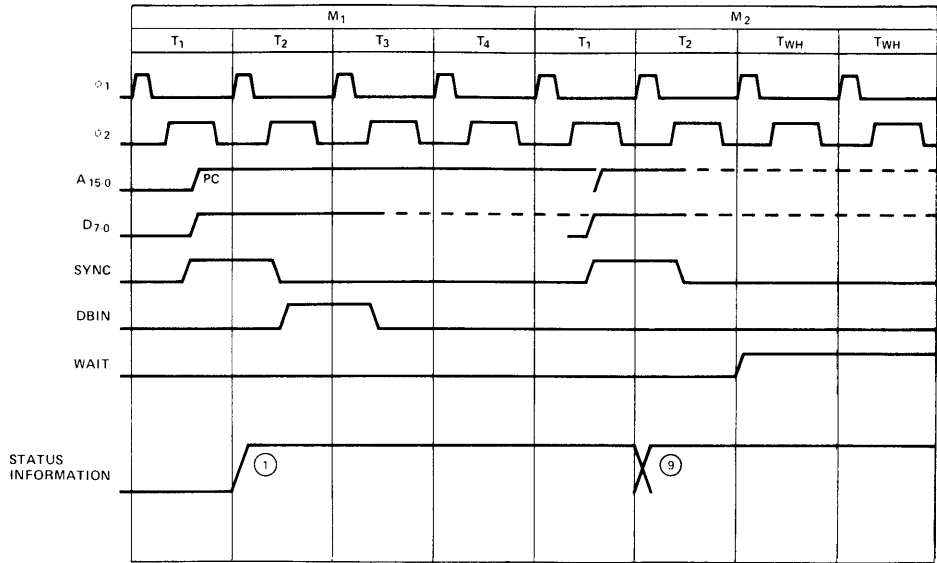
- A high on the RESET line will always reset the 8080 to state  $T_1$ ; RESET also clears the program counter.
- A HOLD input will cause the 8080 to enter the hold state, as previously described. When the HOLD line goes low, the 8080 re-enters the halt state on the rising edge of the next  $\phi_1$  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  $\phi_1$  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 4-12 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 4-6

Figure 4-11. HALT Timing

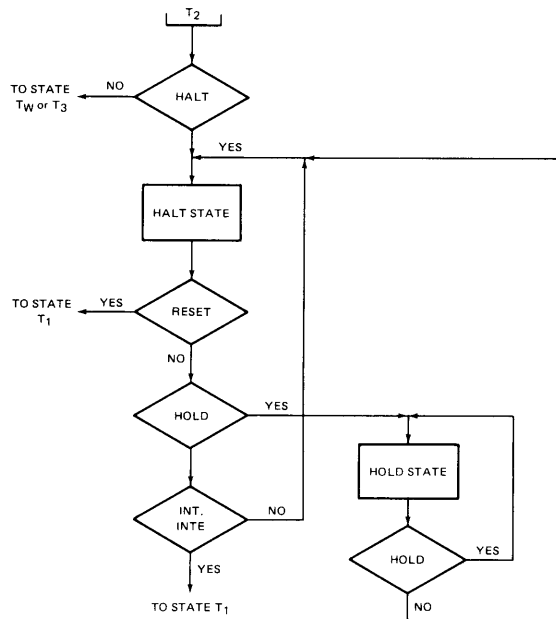
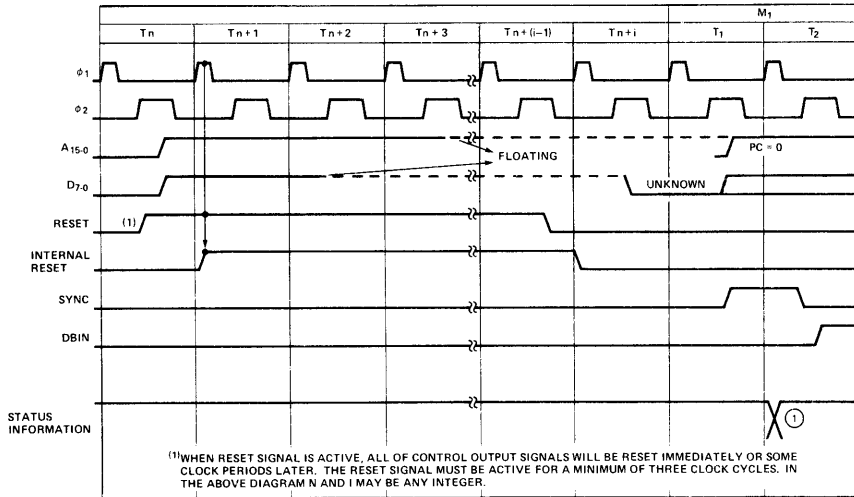
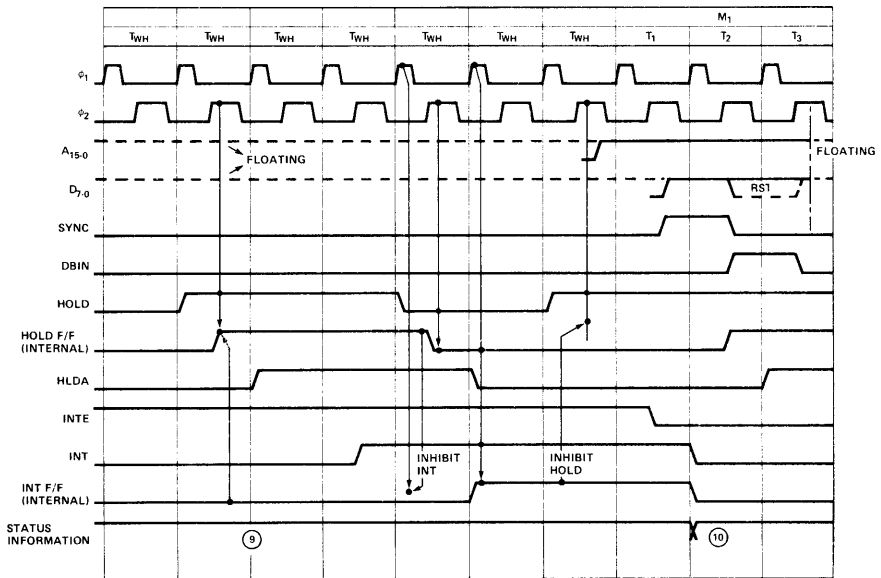


Figure 4-12. HALT Sequence Flow Chart



NOTE: (N) Refer to Status Word Chart on Page 4-6.

Figure 4-13. Reset



NOTE: (N) Refer to Status Word Chart on Page 4-6.

Figure 4-14. Relation between HOLD and INT in the HALT State

| MNEMONIC     | OP CODE   |   | M1[1]         |             |             |                              |           | M2               |                                     |           |
|--------------|---|---|---------------|-------------|-------------|------------------------------|-----------|------------------|-------------------------------------|-----------|
|              | D <sub>7</sub> D <sub>6</sub> D <sub>5</sub> D <sub>4</sub> | D <sub>3</sub> D <sub>2</sub> D <sub>1</sub> D <sub>0</sub> | 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                    |           |
| SPLH         | 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] | PC = PC + 1 B2 → DDD                |           |
| MVI M, data  | 0 0 1 1   | 0 1 1 0   |               |             |             | X                            |           |                  | PC = PC + 1 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   |               |             |             | X                            |           | {11}             | (HL) ↔ (DE)                         |           |
| ADD r        | 1 0 0 0   | 0 S S S   |               |             |             | (SSS)→TMP<br>(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<br>(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<br>(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<br>(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<br>(TMP) + 1 → ALU | ALU→DDD   |                  |                                     |           |
| INR M        | 0 0 1 1   | 0 1 0 0   |               |             |             | X                            |           | HL OUT STATUS[6] | DATA (TMP)+1 → TMP<br>(TMP)+1 → ALU |           |
| DCR r        | 0 0 D D   | D 1 0 1   |               |             |             | (DDD)→TMP<br>(TMP)-1 → ALU   | ALU→DDD   |                  |                                     |           |
| DCR M        | 0 0 1 1   | 0 1 0 1   |               |             |             | X                            |           | HL OUT STATUS[6] | DATA (TMP)-1 → TMP<br>(TMP)-1 → ALU |           |
| INX rp       | 0 0 R P   | 0 0 1 1   |               |             |             | (RP) + 1 → RP                |           |                  |                                     |           |
| DCX rp       | 0 0 R P   | 1 0 1 1   |               |             |             | (RP) - 1 → RP                |           |                  |                                     |           |
| DAD rp[8]    | 0 0 R P   | 1 0 0 1   |               |             |             | X                            |           | (r1)→ACT         | (L)→TMP<br>(ACT)+(TMP)→ALU          | ALU→L, CY |
| DAA          | 0 0 1 0   | 0 1 1 1   |               |             |             | 66→ACT [10]<br>(A)→TMP       |           | {9}              | DAA→A<br>FLAGS [10]                 |           |
| ANA r        | 1 0 1 0   | 0 S S S   |               |             |             | (SSS)→TMP<br>(A)→ACT         |           | {9}              | (ACT)+(TMP)→A                       |           |
| ANA M        | 1 0 1 0   | 0 1 1 0   | PC OUT STATUS | PC = PC + 1 | INST→TMP/IR | (A)→ACT                      |           | HL OUT STATUS[6] | DATA → TMP                          |           |

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

| MNEMONIC        | OP CODE   |   | M1[1]         |             |                 |                                      |    | M2                  |                    |           |
|-----------------|---|---|---------------|-------------|-----------------|--------------------------------------|----|---------------------|--------------------|-----------|
|                 | D <sub>7</sub> D <sub>6</sub> D <sub>5</sub> D <sub>4</sub> | D <sub>3</sub> D <sub>2</sub> D <sub>1</sub> D <sub>0</sub> | 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 (SSI)→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 (SSI)→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 (SSI)→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   |               |             |                 | A→ALU COMPLEMENT                     |    | [9]                 | ALU→A              |           |
| CMC             | 0 0 1 1   | 1 1 1 1   |               |             |                 | CY→ALU COMPLEMENT                    |    | [9]                 | ALU→CY             |           |
| STC             | 0 0 1 1   | 0 1 1 1   |               |             |                 | 1→ALU                                |    | [9]                 | ALU→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   | →PCL      |
| 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   | →PCL      |
| RST n           | 1 1 N N   | N 1 1 1   |               |             | φ-W INST→TMP/IR | SP = SP - 1                          |    | SP OUT STATUS[16]   | SP = SP - 1 (PCH)  | →DATA BUS |
| PCHL            | 1 1 1 0   | 1 0 0 1   |               |             | INST→TMP/IR     | (HL) → PC                            |    |                     |                    |           |
| PUSH rp         | 1 1 R P   | 0 1 0 1   |               |             |                 | SP = SP - 1                          |    | SP OUT STATUS[16]   | SP = SP - 1 (rh)   | →DATA BUS |
| PUSH PSW        | 1 1 1 1   | 0 1 0 1   |               |             |                 | SP = SP - 1                          |    | SP OUT STATUS[16]   | SP = SP - 1 (A)    | →DATA BUS |
| POP rp          | 1 1 R P   | 0 0 0 1   |               |             |                 |                                      | X  | SP OUT STATUS[15]   | SP = SP + 1 DATA   | →rI       |
| 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   |               |             |                 |                                      | X  | SET INTE F/F [11]   |                    |           |
| DI              | 1 1 1 1   | 0 0 1 1   |               |             |                 |                                      | X  | RESET INTE F/F [11] |                    |           |
| 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  |                     |                    |           |



**NOTES:**

1. The first memory cycle (M1) is always an instruction fetch; the first (or only) byte, containing the op code, is fetched during this cycle.
2. If the READY input from memory is not high during T2 of each memory cycle, the processor will enter a wait state (TW) until READY is sampled as high.
3. States T4 and T5 are present, as required, for operations which are completely internal to the CPU. The contents of the internal bus during T4 and T5 are available at the data bus; this is designed for testing purposes only. An "X" denotes that the state is present, but is only used for such internal operations as instruction decoding.
4. Only register pairs rp = B (registers B and C) or rp = D (registers D and E) may be specified.
5. These states are skipped.
6. Memory read sub-cycles; an instruction or data word will be read.
7. Memory write sub-cycle.
8. The READY signal is not required during the second and third sub-cycles (M2 and M3). The HOLD signal is accepted during M2 and M3. The SYNC signal is not generated during M2 and M3. During the execution of DAD, M2 and M3 are required for an internal register-pair add; memory is not referenced.
9. The results of these arithmetic, logical or rotate instructions are not moved into the accumulator (A) until state T2 of the next instruction cycle. That is, A is loaded while the next instruction is being fetched; this overlapping of operations allows for faster processing.
10. If the value of the least significant 4-bits of the accumulator is greater than 9 or if the auxiliary carry bit is set, 6 is added to the accumulator. If the value of the most significant 4-bits of the accumulator is now greater than 9, or if the carry bit is set, 6 is added to the most significant 4-bits of the accumulator.
11. This represents the first sub-cycle (the instruction fetch) of the next instruction cycle.

12. If the condition was met, the contents of the register pair WZ are output on the address lines (A<sub>0-15</sub>) 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<sub>0-7</sub>) and 8-15 (A<sub>8-15</sub>).
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   |    |       |