



Mnemonic	Symbolic Operation	Flags						Op-Code			No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	V	S	N	H	76	543	210				
CALL nn	(SP-1) $\leftarrow$ PC <sub>H</sub> (SP-2) $\leftarrow$ PC <sub>L</sub> PC $\leftarrow$ nn	•	•	•	•	•	•	11	001	101	3	5	17	
CALL cc, nn	If condition cc is false continue, otherwise same as CALL nn	•	•	•	•	•	•	11	cc	100	3	3	10	If cc is false
		•	•	•	•	•	•	←	n	→	3	5	17	If cc is true
RET	PC <sub>L</sub> $\leftarrow$ (SP) PC <sub>H</sub> $\leftarrow$ (SP+1)	•	•	•	•	•	•	11	001	001	1	3	10	
RET cc	If condition cc is false continue, otherwise same as RET	•	•	•	•	•	•	11	cc	000	1	1	5	If cc is false
		•	•	•	•	•	•	←	n	→	1	3	11	If cc is true
RETI	Return from interrupt	•	•	•	•	•	•	11	101	101	2	4	14	
RETN	Return from non maskable interrupt	•	•	•	•	•	•	11	101	101	2	4	14	
		•	•	•	•	•	•	01	000	101				
RST p	(SP-1) $\leftarrow$ PC <sub>H</sub> (SP-2) $\leftarrow$ PC <sub>L</sub> PC <sub>H</sub> $\leftarrow$ 0 PC <sub>L</sub> $\leftarrow$ P	•	•	•	•	•	•	11	t	111	1	3	11	

cc	Condition
000	NZ non zero
001	Z zero
010	NC non carry
011	C carry
100	PO parity odd
101	PE parity even
110	P sign positive
111	M sign negative

t	P
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown  
 † = flag is affected according to the result of the operation.

CALL AND RETURN GROUP  
 TABLE 7.0-10



## 8.0 INTERRUPT RESPONSE

The purpose of an interrupt is to allow peripheral devices to suspend CPU operation in an orderly manner and force the CPU to start a peripheral service routine. Usually this service routine is involved with the exchange of data, or status and control information, between the CPU and the peripheral. Once the service routine is completed, the CPU returns to the operation from which it was interrupted.

### INTERRUPT ENABLE – DISABLE

The Z80 CPU has two interrupt inputs, a software maskable interrupt and a non maskable interrupt. The non maskable interrupt (NMI) can *not* be disabled by the programmer and it will be accepted whenever a peripheral device requests it. This interrupt is generally reserved for very important functions that must be serviced whenever they occur, such as an impending power failure. The maskable interrupt (INT) can be selectively enabled or disabled by the programmer. This allows the programmer to disable the interrupt during periods where his program has timing constraints that do not allow it to be interrupted. In the Z80 CPU there is an enable flip flop (called IFF) that is set or reset by the programmer using the Enable Interrupt (EI) and Disable Interrupt (DI) instructions. When the IFF is reset, an interrupt can not be accepted by the CPU.

Actually, for purposes that will be subsequently explained, there are two enable flip flops, called IFF<sub>1</sub> and IFF<sub>2</sub>.



The state of IFF<sub>1</sub> is used to actually inhibit interrupts while IFF<sub>2</sub> is used as a temporary storage location for IFF<sub>1</sub>. The purpose of storing the IFF<sub>1</sub> will be subsequently explained.

A reset to the CPU will force both IFF<sub>1</sub> and IFF<sub>2</sub> to the reset state so that interrupts are disabled. They can then be enabled by an EI instruction at any time by the programmer. When an EI instruction is executed, any pending interrupt request will not be accepted until after the instruction following EI has been executed. This single instruction delay is necessary for cases when the following instruction is a return instruction and interrupts must not be allowed until the return has been completed. The EI instruction sets both IFF<sub>1</sub> and IFF<sub>2</sub> to the enable state. When an interrupt is accepted by the CPU, both IFF<sub>1</sub> and IFF<sub>2</sub> are automatically reset, inhibiting further interrupts until the programmer wishes to issue a new EI instruction. Note that for all of the previous cases, IFF<sub>1</sub> and IFF<sub>2</sub> are always equal.

The purpose of IFF<sub>2</sub> is to save the status of IFF<sub>1</sub> when a non maskable interrupt occurs. When a non maskable interrupt is accepted, IFF<sub>1</sub> is reset to prevent further interrupts until reenabled by the programmer. Thus, after a non maskable interrupt has been accepted, maskable interrupts are disabled but the previous state of IFF<sub>1</sub> has been saved so that the complete state of the CPU just prior to the non maskable interrupt can be restored at any time. When a Load Register A with Register I (LD A, I) instruction or a Load Register A with Register R (LD A, R) instruction is executed, the state of IFF<sub>2</sub> is copied into the parity flag where it can be tested or stored.

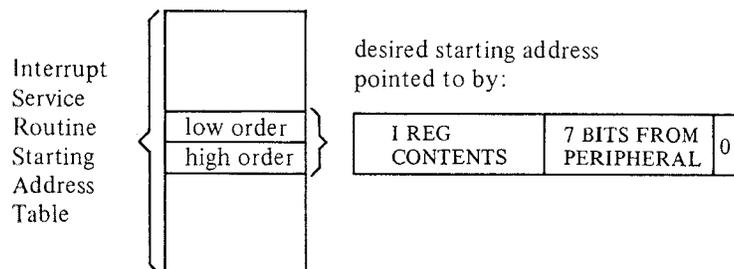
A second method of restoring the status of IFF<sub>1</sub> is thru the execution of a Return From Non Maskable Interrupt (RETN) instruction. Since this instruction indicates that the non maskable interrupt service routine is complete, the contents of IFF<sub>2</sub> are now copied back into IFF<sub>1</sub>, so that the status of IFF<sub>1</sub> just prior to the acceptance of the non maskable interrupt will be restored automatically.



## Mode 2

This mode is the most powerful interrupt response mode. With a single 8 bit byte from the user an indirect call can be made to any memory location.

With this mode the programmer maintains a table of 16 bit starting addresses for every interrupt service routine. This table may be located anywhere in memory. When an interrupt is accepted, a 16 bit pointer must be formed to obtain the desired interrupt service routine starting address from the table. The upper 8 bits of this pointer is formed from the contents of the I register. The I register must have been previously loaded with the desired value by the programmer, i.e. LD I, A. Note that a CPU reset clears the I register so that it is initialized to zero. The lower eight bits of the pointer must be supplied by the interrupting device. Actually, only 7 bits are required from the interrupting device as the least significant bit must be a zero. This is required since the pointer is used to get two adjacent bytes to form a complete 16 bit service routine starting address and the addresses must always start in even locations.



The first byte in the table is the least significant (low order) portion of the address. The programmer must obviously fill this table in with the desired addresses before any interrupts are to be accepted.

Note that this table can be changed at any time by the programmer (if it is stored in Read/Write Memory) to allow different peripherals to be serviced by different service routines.

Once the interrupting device supplies the lower portion of the pointer, the CPU automatically pushes the program counter onto the stack, obtains the starting address from the table and does a jump to this address. This mode of response requires 19 clock periods to complete (7 to fetch the lower 8 bits from the interrupting device, 6 to save the program counter, and 6 to obtain the jump address.)

Note that the Z80 peripheral devices all include a daisy chain priority interrupt structure that automatically supplies the programmed vector to the CPU during interrupt acknowledge. Refer to the Z80-PIO, Z80-SIO and Z80-CTC manuals for details.



Figure 8.0-1 is a summary of the effect of different instructions on the two enable flip flops.

Action	IFF <sub>1</sub>	IFF <sub>2</sub>	
CPU Reset	0	0	
DI	0	0	
EI	1	1	
LD A, I	•	•	IFF <sub>2</sub> → Parity flag
LD A, R	•	•	IFF <sub>2</sub> → Parity flag
Accept NMI	0	•	
RETN	IFF <sub>2</sub>	•	IFF <sub>2</sub> → IFF <sub>1</sub>

“•” indicates no change

**FIGURE 8.0-1**  
**INTERRUPT ENABLE/DISABLE FLIP FLOPS**

## CPU RESPONSE

### Non Maskable

A nonmaskable interrupt will be accepted at all times by the CPU. When this occurs, the CPU ignores the next instruction that it fetches and instead does a restart to location 0066H. Thus, it behaves exactly as if it had received a restart instruction but, it is to a location that is not one of the 8 software restart locations. A restart is merely a call to a specific address in page 0 of memory.

### Maskable

The CPU can be programmed to respond to the maskable interrupt in any one of three possible modes.

#### Mode 0

This mode is identical to the 8080A interrupt response mode. With this mode, the interrupting device can place any instruction on the data bus and the CPU will execute it. Thus, the interrupting device provides the next instruction to be executed instead of the memory. Often this will be a restart instruction since the interrupting device only need supply a single byte instruction. Alternatively, any other instruction such as a 3 byte call to any location in memory could be executed.

The number of clock cycles necessary to execute this instruction is 2 more than the normal number for the instruction. This occurs since the CPU automatically adds 2 wait states to an interrupt response cycle to allow sufficient time to implement an external daisy chain for priority control. Section 5.0 illustrates the detailed timing for an interrupt response. After the application of RESET the CPU will automatically enter interrupt Mode 0.

#### Mode 1

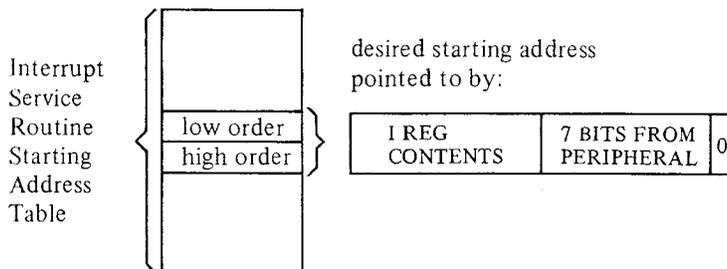
When this mode has been selected by the programmer, the CPU will respond to an interrupt by executing a restart to location 0038H. Thus the response is identical to that for a non maskable interrupt except that the call location is 0038H instead of 0066H. Another difference is that the number of cycles required to complete the restart instruction is 2 more than normal due to the two added wait states.



## Mode 2

This mode is the most powerful interrupt response mode. With a single 8 bit byte from the user an indirect call can be made to any memory location.

With this mode the programmer maintains a table of 16 bit starting addresses for every interrupt service routine. This table may be located anywhere in memory. When an interrupt is accepted, a 16 bit pointer must be formed to obtain the desired interrupt service routine starting address from the table. The upper 8 bits of this pointer is formed from the contents of the I register. The I register must have been previously loaded with the desired value by the programmer, i.e. LD I, A. Note that a CPU reset clears the I register so that it is initialized to zero. The lower eight bits of the pointer must be supplied by the interrupting device. Actually, only 7 bits are required from the interrupting device as the least significant bit must be a zero. This is required since the pointer is used to get two adjacent bytes to form a complete 16 bit service routine starting address and the addresses must always start in even locations.



The first byte in the table is the least significant (low order) portion of the address. The programmer must obviously fill this table in with the desired addresses before any interrupts are to be accepted.

Note that this table can be changed at any time by the programmer (if it is stored in Read/Write Memory) to allow different peripherals to be serviced by different service routines.

Once the interrupting device supplies the lower portion of the pointer, the CPU automatically pushes the program counter onto the stack, obtains the starting address from the table and does a jump to this address. This mode of response requires 19 clock periods to complete (7 to fetch the lower 8 bits from the interrupting device, 6 to save the program counter, and 6 to obtain the jump address.)

Note that the Z80 peripheral devices all include a daisy chain priority interrupt structure that automatically supplies the programmed vector to the CPU during interrupt acknowledge. Refer to the Z80-PIO, Z80-SIO and Z80-CTC manuals for details.





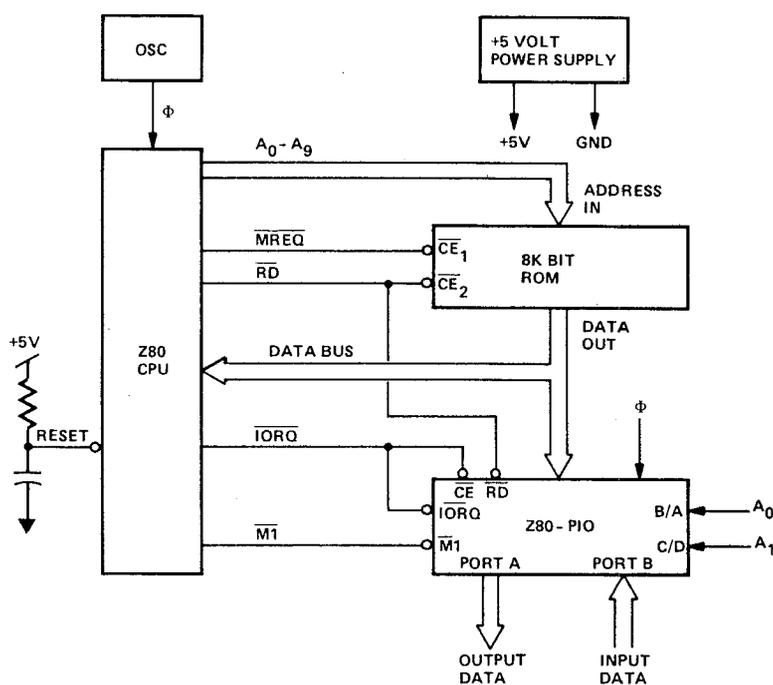
## 9.0 HARDWARE IMPLEMENTATION EXAMPLES

This chapter is intended to serve as a basic introduction to implementing systems with the Z80-CPU.

### MINIMUM SYSTEM

Figure 9.0-1 is a diagram of a very simple Z-80 system. Any Z-80 system must include the following five elements:

- 1) Five volt power supply
- 2) Oscillator
- 3) Memory devices
- 4) I/O circuits
- 5) CPU



**FIGURE 9.0-1**  
**MINIMUM Z80 COMPUTER SYSTEM**

Since the Z80-CPU only requires a single 5 volt supply, most small systems can be implemented using only this single supply.

The oscillator can be very simple since the only requirement is that it be a 5 volt square wave. For systems not running at full speed, a simple RC oscillator can be used. When the CPU is operated near the highest possible frequency, a crystal oscillator is generally required because the system timing will not tolerate the drift or jitter that an RC network will generate. A crystal oscillator can be made from inverters and a few discrete components or monolithic circuits are widely available.

The external memory can be any mixture of standard RAM, ROM, or PROM. In this simple example we have shown a single 8K bit ROM (1K bytes) being utilized as the entire memory system. For this example we have assumed that the Z-80 internal register configuration contains sufficient Read/Write storage so that external RAM memory is not required.

Every computer system requires I/O circuits to allow it to interface to the "real world." In this simple example it is assumed that the output is an 8 bit control vector and the input is an 8 bit status word. The input data could be gated onto the data bus using any standard tri-state driver while the output data could be latched with any type of standard TTL latch. For this example we have used a Z80-PIO for the I/O circuit. This single circuit attaches to the data bus as shown and provides the required 16 bits of TTL compatible I/O. (Refer to the Z80-PIO manual for details on the operation of this circuit.) Notice in this example that with only three LSI circuits, a simple oscillator and a single 5 volt power supply, a powerful computer has been implemented.

### ADDING RAM

Most computer systems require some amount of external Read/Write memory for data storage and to implement a "stack." Figure 9.0-2 illustrates how 256 bytes of static memory can be added to the previous example. In this example the memory space is assumed to be organized as follows:

1K bytes ROM	Address 0000H
256 bytes RAM	03FFH
	0400H
	04FFH

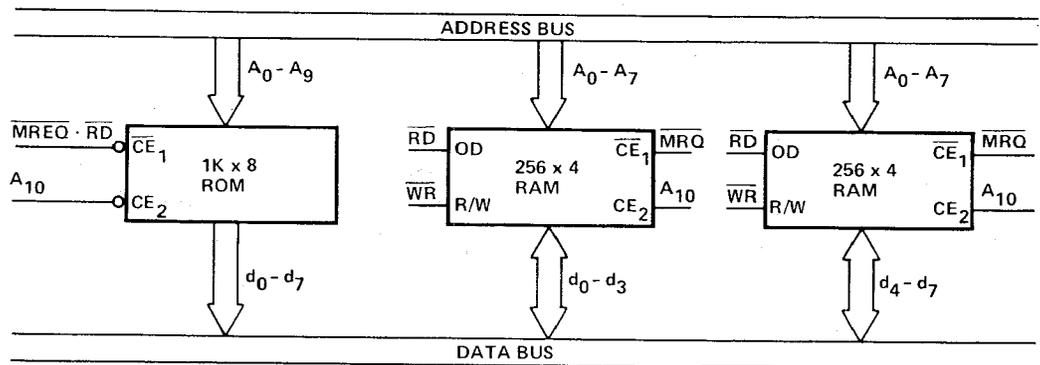


FIGURE 9.0-2  
ROM & RAM IMPLEMENTATION EXAMPLE

In this diagram the address space is described in hexadecimal notation. For this example, address bit A<sub>10</sub> separates the ROM space from the RAM space so that it can be used for the chip select function. For larger amounts of external ROM or RAM, a simple TTL decoder will be required to form the chip selects.

### MEMORY SPEED CONTROL

For many applications, it may be desirable to use slow memories to reduce costs. The WAIT line on the CPU allows the Z-80 to operate with any speed memory. By referring back to section 4 you will notice that the memory access time requirements are most severe during the M1 cycle instruction fetch. All other memory accesses have an additional one half of a clock cycle to be completed. For this reason it may be desirable in some applications to add one wait state to the M1 cycle so that slower memories can be used. Figure 9.0-3 is an example of a simple circuit that will accomplish this task. This circuit can be changed to add a single wait state to any memory access as shown in Figure 9.0-4.

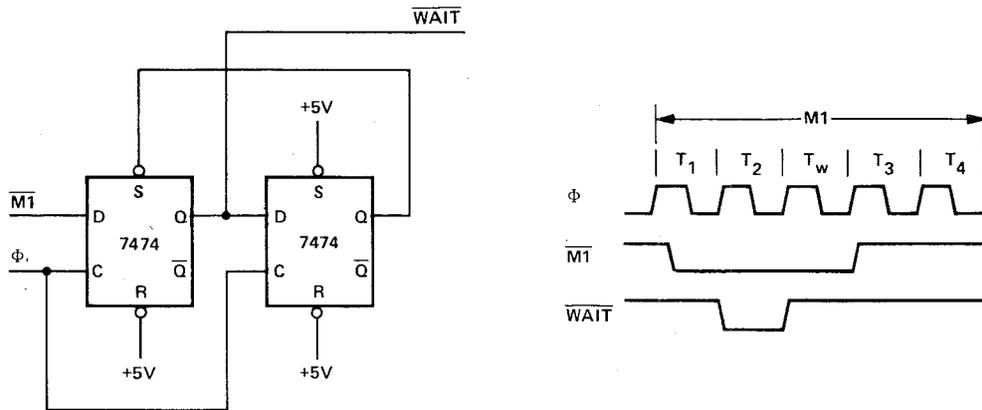
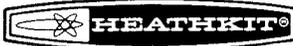


FIGURE 9.0-3  
ADDING ONE WAIT STATE TO AN M1 CYCLE

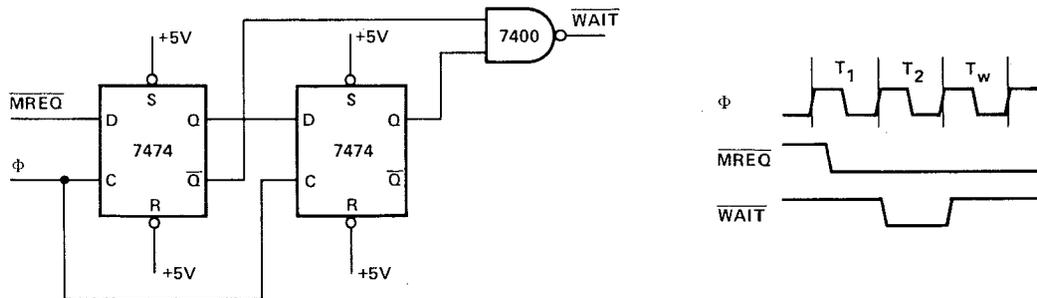
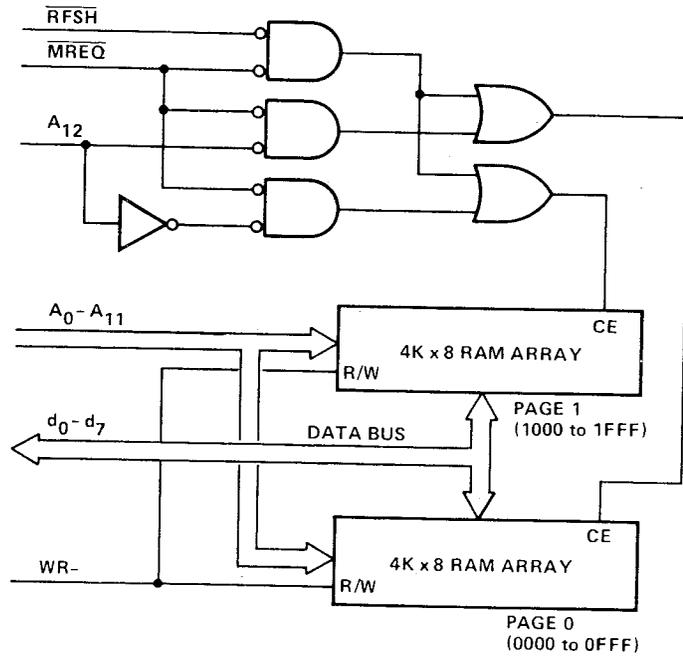


FIGURE 9.0-4  
ADDING ONE WAIT STATE TO ANY MEMORY CYCLE

## INTERFACING DYNAMIC MEMORIES

This section is intended only to serve as a brief introduction to interfacing dynamic memories. Each individual dynamic RAM has varying specifications that will require minor modifications to the description given here and no attempt will be made in this document to give details for any particular RAM. Separate application notes showing how the Z80-CPU can be interfaced to most popular dynamic RAM's are available from Zilog.

Figure 9.0-5 illustrates the logic necessary to interface 8K bytes of dynamic RAM using 18 pin 4K dynamic memories. This figure assumes that the RAM's are the only memory in the system so that  $A_{12}$  is used to select between the two pages of memory. During refresh time, all memories in the system must be read. The CPU provides the proper refresh address on lines  $A_0$  through  $A_6$ . To add additional memory to the system it is necessary to only replace the two gates that operate on  $A_{12}$  with a decoder that operates on all required address bits. For larger systems, buffering for the address and data bus is also generally required.



**FIGURE 9.0-5**  
**INTERFACING DYNAMIC RAMS**



## 10.0 SOFTWARE IMPLEMENTATION EXAMPLES

### 10.1 METHODS OF SOFTWARE IMPLEMENTATION

Several different approaches are possible in developing software for the Z-80 (Figure 10.1). First of all, Assembly Language or PL/Z may be used as the source language. These languages may then be translated into machine language on a commercial time sharing facility using a cross-assembler or cross-compiler or, in the case of assembly language, the translation can be accomplished on a Z-80 Development System using a resident assembler. Finally, the resulting machine code can be debugged either on a time-sharing facility using a Z-80 simulator or on a Z-80 Development System which uses a Z80-CPU directly.

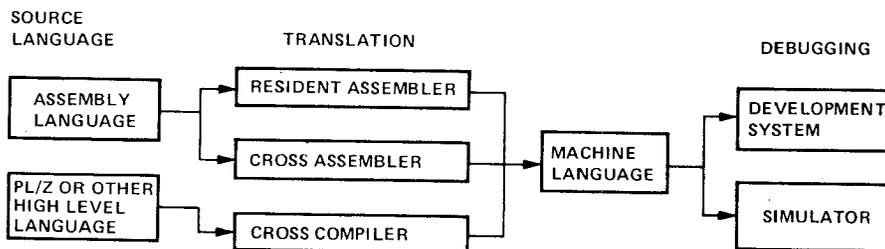


FIGURE 10.1

In selecting a source language, the primary factors to be considered are clarity and ease of programming vs. code efficiency. A high level language such as PL/Z with its machine independent constructs is typically better for formulating and maintaining algorithms, but the resulting machine code is usually somewhat less efficient than what can be written directly in assembly language. These tradeoffs can often be balanced by combining PL/Z and assembly language routines, identifying those portions of a task which must be optimized and writing them as assembly language subroutines.

Deciding whether to use a resident or cross assembler is a matter of availability and short-term vs. long-term expense. While the initial expenditure for a development system is higher than that for a time-sharing terminal, the cost of an individual assembly using a resident assembler is negligible while the same operation on a time-sharing system is relatively expensive and in a short time this cost can equal the total cost of a development system.

Debugging on a development system vs. a simulator is also a matter of availability and expense combined with operational fidelity and flexibility. As with the assembly process, debugging is less expensive on a development system than on a simulator available through time-sharing. In addition, the fidelity of the operating environment is preserved through real-time execution on a Z80-CPU and by connecting the I/O and memory components which will actually be used in the production system. The only advantage to the use of a simulator is the range of criteria which may be selected for such debugging procedures as tracing and setting breakpoints. This flexibility exists because a software simulation can achieve any degree of complexity in its interpretation of machine instructions while development system procedures have hardware limitations such as the capacity of the real-time storage module, the number of breakpoint registers and the pin configuration of the CPU. Despite such hardware limitations, debugging on a development system is typically more productive than on a simulator because of the direct interaction that is possible between the programmer and the authentic execution of his program.



## 10.2 SOFTWARE FEATURES OFFERED BY THE Z80-CPU

The Z-80 instruction set provides the user with a large and flexible repertoire of operations with which to formulate control of the Z80-CPU.

The primary, auxiliary and index registers can be used to hold the arguments of arithmetic and logical operations, or to form memory addresses, or as fast-access storage for frequently used data.

Information can be moved directly from register to register; from memory to memory; from memory to registers; or from registers to memory. In addition, register contents and register/memory contents can be exchanged without using temporary storage. In particular, the contents of primary and auxiliary registers can be completely exchanged by executing only two instructions, EX and EXX. This register exchange procedure can be used to separate the set of working registers between different logical procedures or to expand the set of available registers in a single procedure.

Storage and retrieval of data between pairs of registers and memory can be controlled on a last-in first-out basis through PUSH and POP instructions which utilize a special stack pointer register, SP. This stack register is available both to manipulate data and to automatically store and retrieve addresses for subroutine linkage. When a subroutine is called, for example, the address following the CALL instruction is placed on the top of the push-down stack pointed to by SP. When a subroutine returns to the calling routine, the address on the top of the stack is used to set the program counter for the address of the next instruction. The stack pointer is adjusted automatically to reflect the current "top" stack position during PUSH, POP, CALL and RET instructions. This stack mechanism allows pushdown data stacks and subroutine calls to be nested to any practical depth because the stack area can potentially be as large as memory space.

The sequence of instruction execution can be controlled by six different flags (carry, zero, sign, parity/overflow, add-subtract, half-carry) which reflect the results of arithmetic, logical, shift and compare instructions. After the execution of an instruction which sets a flag, that flag can be used to control a conditional jump or return instruction. These instructions provide logical control following the manipulation of single bit, eight-bit byte (or) sixteen-bit data quantities.

A full set of logical operations, including AND, OR, XOR (exclusive - OR), CPL (NOR) and NEG (two's complement) are available for Boolean operations between the accumulator and 1) all other eight-bit registers, 2) memory locations or 3) immediate operands.

In addition, a full set of arithmetic and logical shifts in both directions are available which operate on the contents of all eight-bit primary registers or directly on any memory location. The carry flag can be included or simply set by these shift instructions to provide both the testing of shift results and to link register/register or register/memory shift operations.

## 10.3 EXAMPLES OF USE OF SPECIAL Z80 INSTRUCTIONS

- A. Let us assume that a string of data in memory starting at location "DATA" is to be moved into another area of memory starting at location "BUFFER" and that the string length is 737 bytes. This operation can be accomplished as follows:

```
LD      HL , DATA      : START ADDRESS OF DATA STRING
LD      DE , BUFFER     : START ADDRESS OF TARGET BUFFER
LD      BC , 737        : LENGTH OF DATA STRING
LDIR                                         : MOVE STRING -- TRANSFER MEMORY POINTED TO
                                         : BY HL INTO MEMORY LOCATION POINTED TO BY DE
                                         : INCREMENT HL AND DE. DECREMENT BC
                                         : PROCESS UNTIL BC = 0.
```

11 bytes are required for this operation and each byte of data is moved in 21 clock cycles.



- B. Let's assume that a string in memory starting at location "DATA" is to be moved into another area of memory starting at location "BUFFER" until an ASCII \$ character (used as string delimiter) is found. Let's also assume that the maximum string length is 132 characters. The operation can be performed as follows:

```

LD      HL , DATA      ; STARTING ADDRESS OF DATA STRING
LD      DE , BUFFER     ; STARTING ADDRESS OF TARGET BUFFER
LD      BC , 132        ; MAXIMUM STRING LENGTH
LD      A , '$'         ; STRING DELIMITER CODE
LOOP:CP (HL)            ; COMPARE MEMORY CONTENTS WITH DELIMITER
JR      Z , END - $     ; GO TO END IF CHARACTERS EQUAL
LDI     ; MOVE CHARACTER (HL) to (DE)
INC     HL AND DE, DECM BC ; INCREMENT HL AND DE, DECREMENT BC
JP      PE , LOOP       ; GO TO "LOOP" IF MORE CHARACTERS
END:    ; OTHERWISE, FALL THROUGH
        ; NOTE: P/V FLAG IS USED
        ; TO INDICATE THAT REGISTER BC WAS
        ; DECREMENTED TO ZERO.

```

19 bytes are required for this operation.

- C. Let us assume that a 16-digit decimal number represented in packed BCD format (two BCD digits/byte) has to be shifted as shown in the Figure 10.2 in order to mechanize BCD multiplication or division. The operation can be accomplished as follows:

```

LD      HL , DATA      ; ADDRESS OF FIRST BYTE
LD      B , COUNT       ; SHIFT COUNT
XOR     A               ; CLEAR ACCUMULATOR
ROTAT:RLD              ; ROTATE LEFT LOW ORDER DIGIT IN ACC
                        ; WITH DIGITS IN (HL)
INC     HL              ; ADVANCE MEMORY POINTER
DJNZ   ROTAT - $       ; DECREMENT B AND GO TO ROTAT IF
                        ; B IS NOT ZERO, OTHERWISE FALL THROUGH

```

11 bytes are required for this operation.

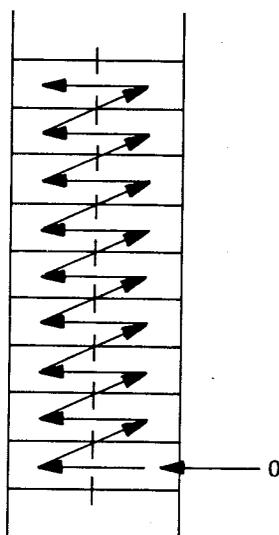


FIGURE 10.2



- D. Let us assume that one number is to be subtracted from another and a) that they are both in packed BCD format, b) that they are of equal but varying length, and c) that the result is to be stored in the location of the minuend. The operation can be accomplished as follows:

```

LD      HL , ARG1      ; ADDRESS OF MINUEND
LD      DE , ARG2      ; ADDRESS OF SUBTRAHEND
LD      B , LENGTH     ; LENGTH OF TWO ARGUMENTS
AND     A              ; CLEAR CARRY FLAG
SUBDEC: LD  A , (DE)    ; SUBTRAHEND TO ACC
SBC     A , (HL)       ; SUBTRACT (HL) FROM ACC
DAA     ; ADJUST RESULT TO DECIMAL CODED VALUE
LD      (HL) , A       ; STORE RESULT
INC     HL             ; ADVANCE MEMORY POINTERS
INC     DE
DJNZ   SUBDEC -- $     ; DECREMENT B AND GO TO "SUBDEC" IF B
                        ; NOT ZERO, OTHERWISE FALL THROUGH
  
```

17 bytes are required for this operation.

#### 10.4 EXAMPLES OF PROGRAMMING TASKS

- A. The following program sorts an array of numbers each in the range (0,255) into ascending order using a standard exchange sorting algorithm.



01/22/76 11:14:37

## BUBBLE LISTING

PAGE 1

```

LOC  OBJ CODE  STMT  SOURCE STATEMENT
      1 ; *** STANDARD EXCHANGE (BUBBLE) SORT ROUTINE ***
      2 ;
      3 ; AT ENTRY: HL CONTAINS ADDRESS OF DATA
      4 ;           C CONTAINS NUMBER OF ELEMENTS TO BE SORTED
      5 ;           (1<C<256)
      6 ;
      7 ; AT EXIT: DATA SORTED IN ASCENDING ORDER
      8 ;
      9 ; USE OF REGISTERS
     10 ;
     11 ; REGISTER  CONTENTS
     12 ;
     13 ; A          TEMPORARY STORAGE FOR CALCULATIONS
     14 ; B          COUNTER FOR DATA ARRAY
     15 ; C          LENGTH OF DATA ARRAY
     16 ; D          FIRST ELEMENT IN COMPARISON
     17 ; E          SECOND ELEMENT IN COMPARISON
     18 ; H          FLAG TO INDICATE EXCHANGE
     19 ; L          UNUSED
     20 ; IX         POINTER INTO DATA ARRAY
     21 ; IY         UNUSED
     22 ;
0000  222600   23  SORT:  LD   (DATA), HL   ; SAVE DATA ADDRESS
0003  CB84    24  LOOP:  RES  FLAG, H     ; INITIALIZE EXCHANGE FLAG
0005  41      25          LD   B, C       ; INITIALIZE LENGTH COUNTER
0006  05      26          DEC  B         ; ADJUST FOR TESTING
0007  DD2A2600 27          LD   IX, (DATA) ; INITIALIZE ARRAY POINTER
000B  DD7E00   28  NEXT:  LD   A, (IX) `   ; FIRST ELEMENT IN COMPARISON
000E  57      29          LD   D, A       ; TEMPORARY STORAGE FOR ELEMENT
000F  DD5E01   30          LD   E, (IX+1) ; SECOND ELEMENT IN COMPARISON
0012  93      31          SUB  E         ; COMPARISON FIRST TO SECOND
0013  3008    32          JR   NC, NOEX-$ ; IF FIRST > SECOND, NO JUMP
0015  DD7300   33          LD   (IX), E     ; EXCHANGE ARRAY ELEMENTS
0018  DD7201   34          LD   (IX+1), D
001B  CBC4    35          SET  FLAG, H     ; RECORD EXCHANGE OCCURRED
001D  DD23    36  NOEX:  INC  IX         ; POINT TO NEXT DATA ELEMENT
001F  10EA    37          DJNZ NEXT-$   ; COUNT NUMBER OF COMPARISONS
      38          ; REPEAT IF MORE DATA PAIRS
0021  CB44    39          BIT  FLAG, H     ; DETERMINE IF EXCHANGE OCCURRED
0023  20DE    40          JR   NZ, LOOP-$ ; CONTINUE IF DATA UNSORTED
0025  C9      41          RET          ; OTHERWISE, EXIT
      42 ;
0026          43  FLAG:  EQU  0         ; DESIGNATION OF FLAG BIT
0026          44  DATA:  DEFS  2       ; STORAGE FOR DATA ADDRESS
      45          END

```



- B. The following program multiplies two unsigned 16 bit integers and leaves the result in the HL register pair.

LOC	OBJ CODE	STMT	SOURCE STATEMENT
01/22/76	11:32:36		MULTIPLY LISTING
			PAGE 1
0000		1	MULT;; UNSIGNED SIXTEEN BIT INTEGER MULTIPLY.
		2	; ON ENTRANCE: MULTIPLIER IN DE.
		3	; MULTIPLICAND IN HL.
		4	;
		5	; ON EXIT: RESULT IN HL.
		6	;
		7	; REGISTER USES:
		8	;
		9	;
		10	; H HIGH ORDER PARTIAL RESULT
		11	; L LOW ORDER PARTIAL RESULT
		12	; D HIGH ORDER MULTIPLICAND
		13	; E LOW ORDER MULTIPLICAND
		14	; B COUNTER FOR NUMBER OF SHIFTS
		15	; C HIGH ORDER BITS OF MULTIPLIER
		16	; A LOW ORDER BITS OF MULTIPLIER
		17	;
0000	0610	18	LD B, 16; NUMBER OF BITS- INITIALIZE
0002	4A	19	LD C, D; MOVE MULTIPLIER
0003	7B	20	LD A, E;
0004	EB	21	EX DE, HL; MOVE MULTIPLICAND
0005	210000	22	LD HL, 0; CLEAR PARTIAL RESULT
0008	CB39	23	MLOOP: SRL C; SHIFT MULTIPLIER RIGHT
000A	1F	24	RRA LEAST SIGNIFICANT BIT IS
		25	; IN CARRY.
000B	3001	26	JR NC, NOADD-; IF NO CARRY, SKIP THE ADD.
000D	19	27	ADD HL, DE; ELSE ADD MULTIPLICAND TO
		28	; PARTIAL RESULT.
000E	EB	29	NOADD: EX DE, HL; SHIFT MULTIPLICAND LEFT
000F	29	30	ADD HL, HL; BY MULTIPLYING IT BY TWO.
0010	EB	31	EX DE, HL;
0011	10F5	32	DJNZ MLOOP-; REPEAT UNTIL NO MORE BITS.
0013	C9	33	RET;
		34	END;

## Absolute Maximum Ratings

Temperature Under Bias	Specified operating range.	*Comment
Storage Temperature	-65°C to +150°C	
Voltage On Any Pin with Respect to Ground	-0.3V to +7V	
Power Dissipation	1.5W	

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 any other condition 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.

Note: For Z80-CPU all AC and DC characteristics remain the same for the military grade parts except  $I_{CC}$ .

$$I_{CC} = 200 \text{ mA}$$

## Z80-CPU D.C. Characteristics

$T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ .  $V_{CC} = 5V \pm 5\%$  unless otherwise specified

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
$V_{ILC}$	Clock Input Low Voltage	-0.3		0.45	V	
$V_{IHC}$	Clock Input High Voltage	$V_{CC} - 0.6$		$V_{CC} + 0.3$	V	
$V_{IL}$	Input Low Voltage	-0.3		0.8	V	
$V_{IH}$	Input High Voltage	2.0		$V_{CC}$	V	
$V_{OL}$	Output Low Voltage			0.4	V	$I_{OL} = 1.8 \text{ mA}$
$V_{OH}$	Output High Voltage	2.4			V	$I_{OH} = -250 \mu\text{A}$
$I_{CC}$	Power Supply Current			150	mA	
$I_{LI}$	Input Leakage Current			10	$\mu\text{A}$	$V_{IN} = 0$ to $V_{CC}$
$I_{LOH}$	Tri-State Output Leakage Current in Float			10	$\mu\text{A}$	$V_{OUT} = 2.4$ to $V_{CC}$
$I_{LOL}$	Tri-State Output Leakage Current in Float			-10	$\mu\text{A}$	$V_{OUT} = 0.4 \text{ V}$
$I_{LD}$	Data Bus Leakage Current in Input Mode			$\pm 10$	$\mu\text{A}$	$0 \leq V_{IN} \leq V_{CC}$

## Capacitance

$T_A = 25^\circ\text{C}$ ,  $f = 1 \text{ MHz}$ ,  
unmeasured pins returned to ground

Symbol	Parameter	Max.	Unit
$C_\Phi$	Clock Capacitance	35	pF
$C_{IN}$	Input Capacitance	5	pF
$C_{OUT}$	Output Capacitance	10	pF

## Z80-CPU

### Ordering Information

C - Ceramic  
P - Plastic  
S - Standard  $5V \pm 5\%$   $0^\circ$  to  $70^\circ\text{C}$   
E - Extended  $5V \pm 5\%$   $-40^\circ$  to  $85^\circ\text{C}$   
M - Military  $5V \pm 10\%$   $-55^\circ$  to  $125^\circ\text{C}$

## Z80A-CPU D.C. Characteristics

$T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ .  $V_{CC} = 5V \pm 5\%$  unless otherwise specified

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
$V_{ILC}$	Clock Input Low Voltage	-0.3		0.45	V	
$V_{IHC}$	Clock Input High Voltage	$V_{CC} - 0.6$		$V_{CC} + 0.3$	V	
$V_{IL}$	Input Low Voltage	-0.3		0.8	V	
$V_{IH}$	Input High Voltage	2.0		$V_{CC}$	V	
$V_{OL}$	Output Low Voltage			0.4	V	$I_{OL} = 1.8 \text{ mA}$
$V_{OH}$	Output High Voltage	2.4			V	$I_{OH} = -250 \mu\text{A}$
$I_{CC}$	Power Supply Current		90	200	mA	
$I_{LI}$	Input Leakage Current			10	$\mu\text{A}$	$V_{IN} = 0$ to $V_{CC}$
$I_{LOH}$	Tri-State Output Leakage Current in Float			10	$\mu\text{A}$	$V_{OUT} = 2.4$ to $V_{CC}$
$I_{LOL}$	Tri-State Output Leakage Current in Float			-10	$\mu\text{A}$	$V_{OUT} = 0.4 \text{ V}$
$I_{LD}$	Data Bus Leakage Current in Input Mode			$\pm 10$	$\mu\text{A}$	$0 \leq V_{IN} \leq V_{CC}$

## Capacitance

$T_A = 25^\circ\text{C}$ ,  $f = 1 \text{ MHz}$ ,  
unmeasured pins returned to ground

Symbol	Parameter	Max.	Unit
$C_\Phi$	Clock Capacitance	35	pF
$C_{IN}$	Input Capacitance	5	pF
$C_{OUT}$	Output Capacitance	10	pF

## Z80A-CPU

### Ordering Information

C - Ceramic  
P - Plastic  
S - Standard  $5V \pm 5\%$   $0^\circ$  to  $70^\circ\text{C}$

# A.C. Characteristics

# Z80-CPU

$T_A = 0^\circ C$  to  $70^\circ C$ ,  $V_{CC} = +5V \pm 5\%$ , Unless Otherwise Noted.

Signal	Symbol	Parameter	Min	Max	Unit	Test Condition
$\phi$	$t_c$	Clock Period	4	1121	$\mu sec$	
	$t_w(\phi H)$	Clock Pulse Width, Clock High	180	[E]	nsec	
	$t_w(\phi L)$	Clock Pulse Width, Clock Low	180	2000	nsec	
	$t_{r,f}$	Clock Rise and Fall Time		30	nsec	
$A_{0-15}$	$t_D(AD)$	Address Output Delay		145	nsec	$C_L = 50pF$
	$t_F(AD)$	Delay to Float		110	nsec	
	$t_{acm}$	Address Stable Prior to $\overline{MREQ}$ (Memory Cycle)	[11]		nsec	
	$t_{aci}$	Address Stable Prior to $\overline{IORQ}$ , $\overline{RD}$ or $\overline{WR}$ (I/O Cycle)	[2]		nsec	
	$t_{ca}$	Address Stable from $\overline{RD}$ , $\overline{WR}$ , $\overline{IORQ}$ or $\overline{MREQ}$	[3]		nsec	
$D_{0-7}$	$t_D(D)$	Data Output Delay		230	nsec	$C_L = 50pF$
	$t_F(D)$	Delay to Float During Write Cycle		90	nsec	
	$t_S(D)$	Data Setup Time to Rising Edge of Clock During M1 Cycle	50		nsec	
	$t_{SF}(D)$	Data Setup Time to Falling Edge of Clock During M2 to M5	60		nsec	
	$t_{dem}$	Data Stable Prior to $\overline{WR}$ (Memory Cycle)	[5]		nsec	
	$t_{dei}$	Data Stable Prior to $\overline{WR}$ (I/O Cycle)	[6]		nsec	
	$t_{cdf}$	Data Stable From $\overline{WR}$	[7]		nsec	
		$t_H$	Any Hold Time for Setup Time	0		
$\overline{MREQ}$	$t_{DL\phi}(MR)$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ Low		100	nsec	$C_L = 50pF$
	$t_{DH\phi}(MR)$	$\overline{MREQ}$ Delay From Rising Edge of Clock, $\overline{MREQ}$ High		100	nsec	
	$t_{w}(MRL)$	Pulse Width, $\overline{MREQ}$ Low	[8]		nsec	
	$t_{w}(MRH)$	Pulse Width, $\overline{MREQ}$ High	[9]		nsec	
$\overline{IORQ}$	$t_{DL\phi}(IR)$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ Low		90	nsec	$C_L = 50pF$
	$t_{DL\phi}(IR)$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ Low		110	nsec	
	$t_{DH\phi}(IR)$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ High		100	nsec	
	$t_{DH\phi}(IR)$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ High		110	nsec	
$\overline{RD}$	$t_{DL\phi}(RD)$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ Low		100	nsec	$C_L = 50pF$
	$t_{DL\phi}(RD)$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ Low		130	nsec	
	$t_{DH\phi}(RD)$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ High		100	nsec	
	$t_{DH\phi}(RD)$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ High		110	nsec	
$\overline{WR}$	$t_{DL\phi}(WR)$	$\overline{WR}$ Delay From Rising Edge of Clock, $\overline{WR}$ Low		80	nsec	$C_L = 50pF$
	$t_{DL\phi}(WR)$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ Low		90	nsec	
	$t_{DH\phi}(WR)$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ High		100	nsec	
	$t_w(\overline{WRL})$	Pulse Width, $\overline{WR}$ Low	[10]		nsec	
$\overline{M1}$	$t_{DL}(M1)$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ Low		130	nsec	$C_L = 50pF$
	$t_{DH}(M1)$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ High		130	nsec	
$\overline{RFSH}$	$t_{DL}(RF)$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ Low		180	nsec	$C_L = 50pF$
	$t_{DH}(RF)$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ High		150	nsec	
$\overline{WAIT}$	$t_s(WT)$	$\overline{WAIT}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{HALT}$	$t_D(HT)$	$\overline{HALT}$ Delay Time From Falling Edge of Clock		300	nsec	$C_L = 50pF$
$\overline{INT}$	$t_s(IT)$	$\overline{INT}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{NM1}$	$t_w(\overline{NML})$	Pulse Width, $\overline{NM1}$ Low	80		nsec	
$\overline{BUSRQ}$	$t_s(BQ)$	$\overline{BUSRQ}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{BUSA\overline{K}}$	$t_{DL}(BA)$	$\overline{BUSA\overline{K}}$ Delay From Rising Edge of Clock, $\overline{BUSA\overline{K}}$ Low		120	nsec	$C_L = 50pF$
	$t_{DH}(BA)$	$\overline{BUSA\overline{K}}$ Delay From Falling Edge of Clock, $\overline{BUSA\overline{K}}$ High		110	nsec	
$\overline{RESET}$	$t_s(RS)$	$\overline{RESET}$ Setup Time to Rising Edge of Clock	90		nsec	
	$t_F(C)$	Delay to Float ( $\overline{MREQ}$ , $\overline{IORQ}$ , $\overline{RD}$ and $\overline{WR}$ )		100	nsec	
	$t_{mr}$	$\overline{M1}$ Stable Prior to $\overline{IORQ}$ (Interrupt Ack.)	[11]		nsec	

[12]  $t_c = t_w(\phi H) + t_w(\phi L) + t_r + t_f$

[11]  $t_{acm} = t_w(\phi H) + t_r - 75$

[2]  $t_{aci} = t_c - 80$

[3]  $t_{ca} = t_w(\phi L) + t_r - 40$

[4]  $t_{cdf} = t_w(\phi L) + t_r - 60$

[5]  $t_{dem} = t_c - 210$

[6]  $t_{dei} = t_w(\phi L) + t_r - 210$

[7]  $t_{cdf} = t_w(\phi L) + t_r - 80$

[8]  $t_w(MRL) = t_c - 40$

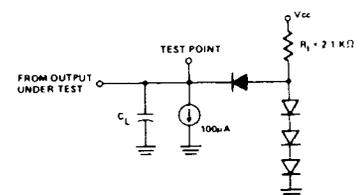
[9]  $t_w(MRH) = t_w(\phi H) + t_r - 30$

[10]  $t_w(\overline{WRL}) = t_c - 40$

[11]  $t_{mr} = 2t_c + t_w(\phi H) + t_r - 80$

NOTES:

- A. Data should be enabled onto the CPU data bus when  $\overline{RD}$  is active. During interrupt acknowledge data should be enabled when  $\overline{M1}$  and  $\overline{IORQ}$  are both active.
- B. All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.
- C. The  $\overline{RESET}$  signal must be active for a minimum of 3 clock cycles.
- D. Output Delay vs. Loaded Capacitance  
 $T_A = 70^\circ C$      $V_{CC} = +5V \pm 5\%$   
 Add 10nsec delay for each 50pf increase in load up to a maximum of 200pf for the data bus & 100pf for address & control lines
- E. Although static by design, testing guarantees  $t_w(\phi H)$  of 200  $\mu sec$  maximum



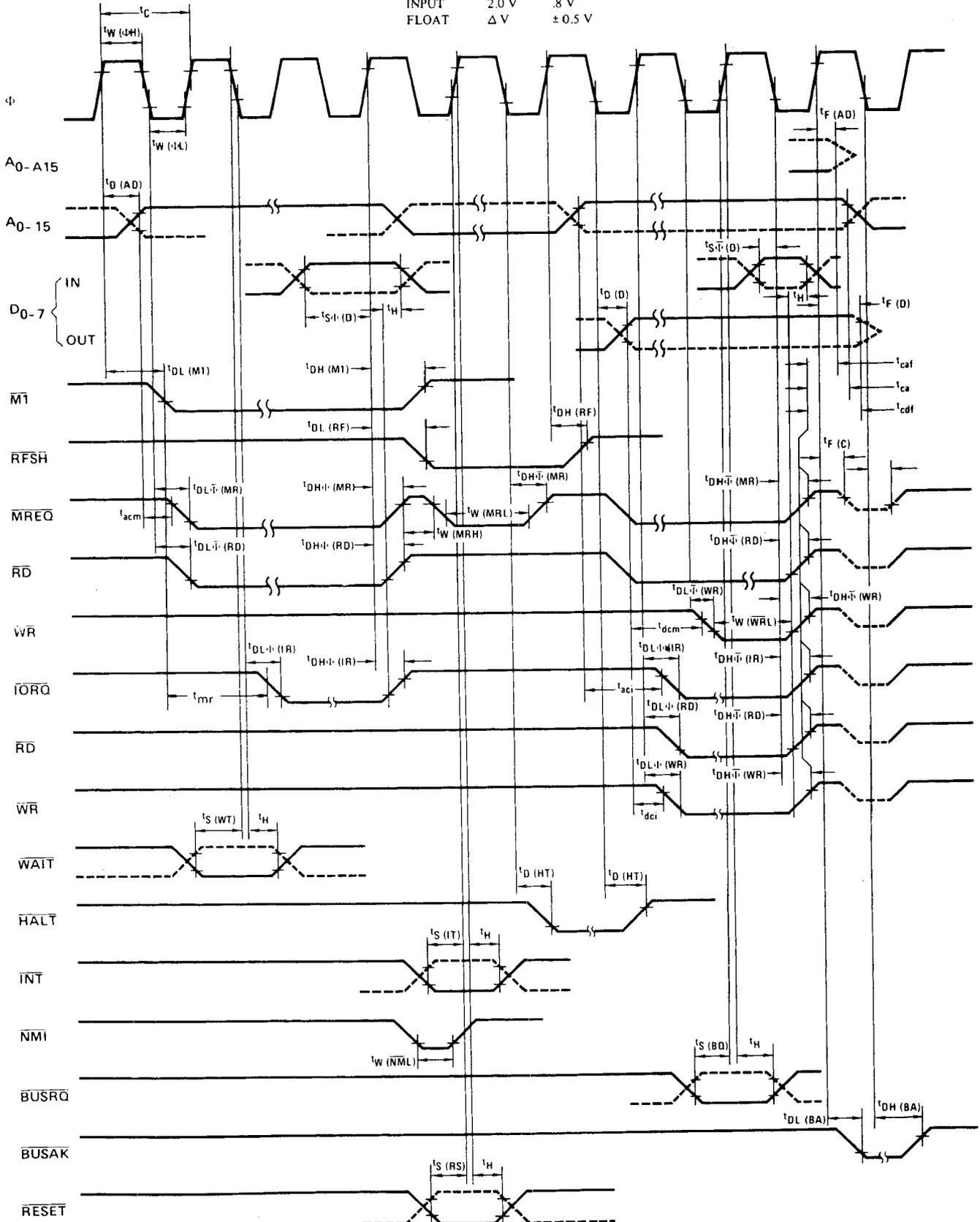
Load circuit for Output



# A.C. Timing Diagram

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	V <sub>cc</sub> - .6V	.45V
OUTPUT	2.0 V	.8 V
INPUT	2.0 V	.8 V
FLOAT	Δ V	± 0.5 V



# A.C. Characteristics

# Z80A-CPU

T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = +5V ± 5%, Unless Otherwise Noted.

Signal	Symbol	Parameter	Min	Max	Unit	Test Condition
φ	t <sub>c</sub>	Clock Period	.25	[12]	μsec	
	t <sub>w</sub> (ΦH)	Clock Pulse Width, Clock High	110	[E]	nsec	
	t <sub>w</sub> (ΦL)	Clock Pulse Width, Clock Low	110	2000	nsec	
	t <sub>r, f</sub>	Clock Rise and Fall Time		30	nsec	
A <sub>0-15</sub>	t <sub>D</sub> (AD)	Address Output Delay		110	nsec	C <sub>L</sub> = 50pF
	t <sub>F</sub> (AD)	Delay to Float		90	nsec	
	t <sub>acm</sub>	Address Stable Prior to $\overline{MREQ}$ (Memory Cycle)	[1]		nsec	
	t <sub>aci</sub>	Address Stable Prior to $\overline{IORQ}$ , $\overline{RD}$ or $\overline{WR}$ (I/O Cycle)	[2]		nsec	
	t <sub>ca</sub>	Address Stable from $\overline{RD}$ , $\overline{WR}$ , $\overline{IORQ}$ or $\overline{MREQ}$	[3]		nsec	
D <sub>0-7</sub>	t <sub>D</sub> (D)	Data Output Delay		150	nsec	C <sub>L</sub> = 50pF
	t <sub>F</sub> (D)	Delay to Float During Write Cycle		90	nsec	
	t <sub>SΦ</sub> (D)	Data Setup Time to Rising Edge of Clock During M1 Cycle	35		nsec	
	t <sub>SΦ</sub> (D)	Data Setup Time to Falling Edge of Clock During M2 to M5	50		nsec	
	t <sub>dcm</sub>	Data Stable Prior to $\overline{WR}$ (Memory Cycle)	[5]		nsec	
	t <sub>dci</sub>	Data Stable Prior to $\overline{WR}$ (I/O Cycle)	[6]		nsec	
	t <sub>cdf</sub>	Data Stable From $\overline{WR}$	[7]		nsec	
	t <sub>H</sub>	Any Hold Time for Setup Time		0	nsec	
$\overline{MREQ}$	t <sub>DLΦ</sub> (MR)	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ Low		85	nsec	C <sub>L</sub> = 50pF
	t <sub>DHΦ</sub> (MR)	$\overline{MREQ}$ Delay From Rising Edge of Clock, $\overline{MREQ}$ High		85	nsec	
	t <sub>DHΦ</sub> (MR)	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ High		85	nsec	
	t <sub>w</sub> (MRL)	Pulse Width, $\overline{MREQ}$ Low	[8]		nsec	
	t <sub>w</sub> (MRH)	Pulse Width, $\overline{MREQ}$ High	[9]		nsec	
$\overline{IORQ}$	t <sub>DLΦ</sub> (IR)	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ Low		75	nsec	C <sub>L</sub> = 50pF
	t <sub>DLΦ</sub> (IR)	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ Low		85	nsec	
	t <sub>DHΦ</sub> (IR)	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ High		85	nsec	
	t <sub>DHΦ</sub> (IR)	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ High		85	nsec	
$\overline{RD}$	t <sub>DLΦ</sub> (RD)	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ Low		85	nsec	C <sub>L</sub> = 50pF
	t <sub>DLΦ</sub> (RD)	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ Low		95	nsec	
	t <sub>DHΦ</sub> (RD)	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ High		85	nsec	
	t <sub>DHΦ</sub> (RD)	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ High		85	nsec	
$\overline{WR}$	t <sub>DLΦ</sub> (WR)	$\overline{WR}$ Delay From Rising Edge of Clock, $\overline{WR}$ Low		65	nsec	C <sub>L</sub> = 50pF
	t <sub>DLΦ</sub> (WR)	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ Low		80	nsec	
	t <sub>DHΦ</sub> (WR)	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ High		80	nsec	
	t <sub>w</sub> (WRL)	Pulse Width, $\overline{WR}$ Low	[10]		nsec	
M1	t <sub>DL</sub> (M1)	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ Low		100	nsec	C <sub>L</sub> = 50pF
	t <sub>DH</sub> (M1)	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ High		100	nsec	
RFSH	t <sub>DL</sub> (RF)	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ Low		130	nsec	C <sub>L</sub> = 50pF
	t <sub>DH</sub> (RF)	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ High		120	nsec	
WAIT	t <sub>S</sub> (WT)	$\overline{WAIT}$ Setup Time to Falling Edge of Clock	70		nsec	
HALT	t <sub>D</sub> (HT)	$\overline{HALT}$ Delay Time From Falling Edge of Clock		300	nsec	C <sub>L</sub> = 50pF
INT	t <sub>S</sub> (IT)	$\overline{INT}$ Setup Time to Rising Edge of Clock	80		nsec	
NMI	t <sub>w</sub> (NML)	Pulse Width, $\overline{NMI}$ Low	80		nsec	
BUSRQ	t <sub>S</sub> (BQ)	$\overline{BUSRQ}$ Setup Time to Rising Edge of Clock	50		nsec	
$\overline{BUSA}$	t <sub>DL</sub> (BA)	$\overline{BUSA}$ Delay From Rising Edge of Clock, $\overline{BUSA}$ Low		100	nsec	C <sub>L</sub> = 50pF
	t <sub>DH</sub> (BA)	$\overline{BUSA}$ Delay From Falling Edge of Clock, $\overline{BUSA}$ High		100	nsec	
RESET	t <sub>S</sub> (RS)	$\overline{RESET}$ Setup Time to Rising Edge of Clock	60		nsec	
	t <sub>F</sub> (C)	Delay to Float ( $\overline{MREQ}$ , $\overline{IORQ}$ , $\overline{RD}$ and $\overline{WR}$ )		80	nsec	
	t <sub>mr</sub>	M1 Stable Prior to $\overline{IORQ}$ (Interrupt Ack.)	[11]		nsec	

[12] t<sub>c</sub> = t<sub>w</sub>(ΦH) + t<sub>w</sub>(ΦL) + t<sub>r</sub> + t<sub>f</sub>

[1] t<sub>acm</sub> = t<sub>w</sub>(ΦH) + t<sub>r</sub> - 65

[2] t<sub>aci</sub> = t<sub>c</sub> - 70

[3] t<sub>ca</sub> = t<sub>w</sub>(ΦL) + t<sub>r</sub> - 50

[4] t<sub>caf</sub> = t<sub>w</sub>(ΦL) + t<sub>r</sub> - 45

[5] t<sub>dcm</sub> = t<sub>c</sub> - 170

[6] t<sub>dci</sub> = t<sub>w</sub>(ΦL) + t<sub>r</sub> - 170

[7] t<sub>cdf</sub> = t<sub>w</sub>(ΦL) + t<sub>r</sub> - 70

[8] t<sub>w</sub>(MRL) = t<sub>c</sub> - 30

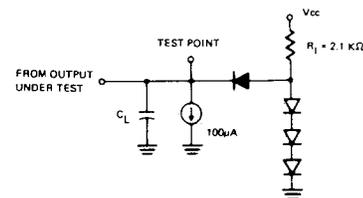
[9] t<sub>w</sub>(MRH) = t<sub>w</sub>(ΦH) + t<sub>r</sub> - 20

[10] t<sub>w</sub>(WRL) = t<sub>c</sub> - 30

[11] t<sub>mr</sub> = 2t<sub>c</sub> + t<sub>w</sub>(ΦH) + t<sub>r</sub> - 65

NOTES:

- A. Data should be enabled onto the CPU data bus when  $\overline{RD}$  is active. During interrupt acknowledge data should be enabled when  $\overline{M1}$  and  $\overline{IORQ}$  are both active.
- B. All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.
- C. The  $\overline{RESET}$  signal must be active for a minimum of 3 clock cycles.
- D. Output Delay vs. Loaded Capacitance  
 T<sub>A</sub> = 70°C V<sub>CC</sub> = +5V ± 5%  
 Add 10nsec delay for each 50pf increase in load up to maximum of 200pf for data bus and 100pf for address & control lines.
- E. Although static by design, testing guarantees t<sub>w</sub>(ΦH) of 200 μsec maximum



Load circuit for Output



12.0

**Z80-CPU  
INSTRUCTION SET**

ADC HL, ss	Add with Carry Reg. pair ss to HL	DEC IY	Decrement IY
ADC A, s	Add with carry operand s to Acc.	DEC ss	Decrement Reg. pair ss
ADD A, n	Add value n to Acc.	DI	Disable interrupts
ADD A, r	Add Reg. r to Acc.	DJNZ e	Decrement B and Jump relative if B $\neq$ 0
ADD A, (HL)	Add location (HL) to Acc.	EI	Enable interrupts
ADD A, (IX+d)	Add location (IX+d) to Acc.	EX (SP), HL	Exchange the location (SP) and HL
ADD A, (IY+d)	Add location (IY+d) to Acc.	EX (SP), IX	Exchange the location (SP) and IX
ADD HL, ss	Add Reg. pair ss to HL	EX (SP), IY	Exchange the location (SP) and IY
ADD IX, pp	Add Reg. pair pp to IX	EX AF, AF'	Exchange the contents of AF and AF'
ADD IY, rr	Add Reg. pair rr to IY	EX DE, HL	Exchange the contents of DE and HL
AND s	Logical 'AND' of operand s and Acc.	EXX	Exchange the contents of BC, DE, HL with contents of BC', DE', HL' respectively
BIT b, (HL)	Test BIT b of location (HL)	HALT	HALT (wait for interrupt or reset)
BIT b, (IX+d)	Test BIT b of location (IX+d)	IM 0	Set interrupt mode 0
BIT b, (IY+d)	Test BIT b of location (IY+d)	IM 1	Set interrupt mode 1
BIT b, r	Test BIT b of Reg. r	IM 2	Set interrupt mode 2
CALL cc, nn	Call subroutine at location nn if condition cc if true	IN A, (n)	Load the Acc. with input from device n
CALL nn	Unconditional call subroutine at location nn	IN r, (C)	Load the Reg. r with input from device (C)
CCF	Complement carry flag	INC (HL)	Increment location (HL)
CP s	Compare operand s with Acc.	INC IX	Increment IX
CPD	Compare location (HL) and Acc. decrement HL and BC	INC (IX+d)	Increment location (IX+d)
CPDR	Compare location (HL) and Acc. decrement HL and BC, repeat until BC=0	INC IY	Increment IY
CPI	Compare location (HL) and Acc. increment HL and decrement BC	INC (IY+d)	Increment location (IY+d)
CPIR	Compare location (HL) and Acc. increment HL, decrement BC repeat until BC=0	INC r	Increment Reg. r
CPL	Complement Acc. (1's comp)	INC ss	Increment Reg. pair ss
DAA	Decimal adjust Acc.	IND	Load location (HL) with input from port (C), decrement HL and B
DEC m	Decrement operand m	INDR	Load location (HL) with input from port (C), decrement HL and decrement B, repeat until B=0
DEC IX	Decrement IX	INI	Load location (HL) with input from port (C); and increment HL and decrement B



INIR	Load location (HL) with input from port (C), increment HL and decrement B, repeat until B=0	LD (nn), A	Load location (nn) with Acc.
JP (HL)	Unconditional Jump to (HL)	LD (nn), dd	Load location (nn) with Reg. pair dd
JP (IX)	Unconditional Jump to (IX)	LD (nn), HL	Load location (nn) with HL
JP (IY)	Unconditional Jump to (IY)	LD (nn), IX	Load location (nn) with IX
JP cc, nn	Jump to location nn if condition cc is true	LD (nn), IY	Load location (nn) with IY
JP nn	Unconditional jump to location nn	LD R, A	Load R with Acc.
JP C, e	Jump relative to PC+e if carry=1	LD r, (HL)	Load Reg. r with location (HL)
JR e	Unconditional Jump relative to PC+e	LD r, (IX+d)	Load Reg. r with location (IX+d)
JP NC, e	Jump relative to PC+e if carry=0	LD r, (IY+d)	Load Reg. r with location (IY+d)
JR NZ, e	Jump relative to PC+e if non zero (Z=0)	LD r, n	Load Reg. r with value n
JR Z, e	Jump relative to PC+e if zero (Z=1)	LD r, r'	Load Reg. r with Reg. r'
LD A, (BC)	Load Acc. with location (BC)	LD SP, HL	Load SP with HL
LD A, (DE)	Load Acc. with location (DE)	LD SP, IX	Load SP with IX
LD A, I	Load Acc. with I	LD SP, IY	Load SP with IY
LD A, (nn)	Load Acc. with location nn	LDD	Load location (DE) with location (HL), decrement DE, HL and BC
LD A, R	Load Acc. with Reg. R	LDDR	Load location (DE) with location (HL), decrement DE, HL and BC; repeat until BC=0
LD (BC), A	Load location (BC) with Acc.	LDI	Load location (DE) with location (HL), increment DE, HL, decrement BC
LD (DE), A	Load location (DE) with Acc.	LDIR	Load location (DE) with location (HL), increment DE, HL, decrement BC and repeat until BC=0
LD (HL), n	Load location (HL) with value n	NEG	Negate Acc. (2's complement)
LD dd, nn	Load Reg. pair dd with value nn	NOP	No operation
LD HL, (nn)	Load HL with location (nn)	OR s	Logical 'OR' or operand s and Acc.
LD (HL), r	Load location (HL) with Reg. r	OTDR	Load output port (C) with location (HL) decrement HL and B, repeat until B=0
LD I, A	Load I with Acc.	OTIR	Load output port (C) with location (HL), increment HL, decrement B, repeat until B=0
LF IX, nn	Load IX with value nn	OUT (C), r	Load output port (C) with Reg. r
LD IX, (nn)	Load IX with location (nn)	OUT (n), A	Load output port (n) with Acc.
LD (IX+d), n	Load location (IX+d) with value n	OUTD	Load output port (C) with location (HL), decrement HL and B
LD (IX+d), r	Load location (IX+d) with Reg. r	OUTI	Load output port (C) with location (HL), increment HL and decrement B
LD IY, nn	Load IY with value nn		
LD IY, (nn)	Load IY with location (nn)		
LD (IY+d), n	Load location (IY+d) with value n		
LD (IY+d), r	Load location (IY+d) with Reg. r		



POP IX	Load IX with top of stack	RR m	Rotate right through carry operand m
POP IY	Load IY with top of stack	RRA	Rotate right Acc. through carry
POP qq	Load Reg. pair qq with top of stack	RRC m	Rotate operand m right circular
PUSH IX	Load IX onto stack	RRCA	Rotate right circular Acc.
PUSH IY	Load IY onto stack	RRD	Rotate digit right and left between Acc. and location (HL)
PUSH qq	Load Reg. pair qq onto stack	RST p	Restart to location p
RES b, m	Reset Bit b of operand m	SBC A, s	Subtract operand s from Acc. with carry
RET	Return from subroutine	SBC HL, ss	Subtract Reg. pair ss from HL with carry
RET cc	Return from subroutine if condition cc is true	SCF	Set carry flag (C=1)
RETI	Return from interrupt	SET b, (HL)	Set Bit b of location (HL)
RETN	Return from non maskable interrupt	SET b, (IX+d)	Set Bit b of location (IX+d)
RL m	Rotate left through carry operand m	SET b, (IY+d)	Set Bit b of location (IY+d)
RLA	Rotate left Acc. through carry	SET b, r	Set Bit b of Reg. r
RLC (HL)	Rotate location (HL) left circular	SLA m	Shift operand m left arithmetic
RLC (IX+d)	Rotate location (IX+d) left circular	SRA m	Shift operand m right arithmetic
RLC (IY+d)	Rotate location (IY+d) left circular	SRL m	Shift operand m right logical
RLC r	Rotate Reg. r left circular	SUB s	Subtract operand s from Acc.
RLCA	Rotate left circular Acc.	XOR s	Exclusive 'OR' operand s and Acc.
RLD	Rotate digit left and right between Acc. and location (HL)		



# INS8250 ASYNCHRONOUS COMMUNICATIONS ELEMENT\*

## INS8250 Functional Pin Description

The function of all INS8250 input/output pins are described in the following paragraphs. (See the INS8250 Block Diagram, Illustration Booklet, Page 16). Some of these descriptions reference internal circuits. A low in these descriptions represents a logic 0 (0 volt nominal) and a high represents a logic 1 (+2.4 volts nominal).

### INPUT SIGNALS

**Chip Select (CS $\emptyset$ , CS1,  $\overline{\text{CS2}}$ ), Pins 12 - 14:** When CS $\emptyset$  and CS1 are high and  $\overline{\text{CS2}}$  is low, the chip is selected. Chip selection is complete when the decoded chip select signal is latched with an active (low) Address Strobe ( $\overline{\text{ADS}}$ ) input. This enables communication between the INS8250 and the CPU.

**Data Input Strobe (DISTR,  $\overline{\text{DISTR}}$ ), Pins 22 and 21:** When DISTR is high or  $\overline{\text{DISTR}}$  is low while the chip is selected, this allows the CPU to read status information or data from a selected register of the INS8250.

NOTE: Only an active DISTR or  $\overline{\text{DISTR}}$  input is required to transfer data from the INS8250 during a read operation. Therefore, tie either the DISTR input permanently low or the  $\overline{\text{DISTR}}$  input permanently high, if not used.

**Data Output Strobe (DOSTR,  $\overline{\text{DOSTR}}$ ), Pins 19 and 18:** When DOSTR is high or  $\overline{\text{DOSTR}}$  is low while the chip is selected, this allows the CPU to write data or control words into a selected register of the INS8250.

NOTE: Only an active DOSTR or  $\overline{\text{DOSTR}}$  input is required to transfer data to the INS8250 during a write operation. Therefore, tie either the DOSTR input permanently low or the  $\overline{\text{DOSTR}}$  input permanently high, if not used.

**Address Strobe ( $\overline{\text{ADS}}$ ), Pin 25:** When low, it provides latching for the Register Select (A0, A1, A2) and Chip Select (CS $\emptyset$ , CS1,  $\overline{\text{CS2}}$ ) signals.

NOTE: An active  $\overline{\text{ADS}}$  input is required when the Register Select (A0, A1, A2) signals are not stable for the duration of a read or write operation. If not required, tie the  $\overline{\text{ADS}}$  input permanently low.

**Register Select (A0, A1, A2), Pins 26 - 28:** These three inputs are used during a read or write operation to select an INS8250 register to read from or write into as indicated in the table below. Note that the state of the Divisor Latch Access Bit (DLAB), which is the most significant bit of the line control register, affects the selection of certain INS8250 registers. The DLAB is reset low when the Master Reset (MR) input is active (low); the DLAB must be set high by the system software to access the baud generator divisor latches.

DLAB	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>	Register
0	0	0	0	Receiver Buffer (read), Transmitter Holding Register (write)
0	0	0	1	Interrupt Enable
X	0	1	0	Interrupt Identification (read only)
X	0	1	1	Line Control
X	1	0	0	MODEM Control
X	1	0	1	Line Status
X	1	1	0	MODEM Status
X	1	1	1	None
1	0	0	0	Divisor Latch (least significant byte)
1	0	0	1	Divisor Latch (most significant byte)

**Master Reset (MR), Pin 35:** When high, it clears all the registers (except the receiver buffer, transmitter holding, and divisor latches), and the control logic of the INS8250. Also, the state of various output signals (SOUT, INTRPT,  $\overline{\text{OUT 1}}$ ,  $\overline{\text{OUT 2}}$ ,  $\overline{\text{RTS}}$ ,  $\overline{\text{DTR}}$ ) are affected by an active MR input. (Refer to Table 1 on Page 14-3.)

^

**Receiver Clock (RCLK), Pin 9:** This input is the 16x baud rate clock for the receiver section of the chip.

**Serial Input (SIN), Pin 10:** Serial data input from the communications link (peripheral device, MODEM, or data set).

\* Portions of this section are reprinted with the permission of National Semiconductor.



**Clear to Send ( $\overline{\text{CTS}}$ ), Pin 36:** The  $\overline{\text{CTS}}$  signal is a MODEM control function input whose condition can be tested by the CPU by reading bit 4 (CTS) of the MODEM status register. Bit 0 (DCTS) of the MODEM status register indicates whether the  $\overline{\text{CTS}}$  input has changed state since the previous reading of the MODEM status register.

NOTE: Whenever the CTS bit of the MODEM status register changes state, an interrupt is generated if enabled.

**Data Set Ready ( $\overline{\text{DSR}}$ ), Pin 37:** When low, it indicates that the MODEM or data set is ready to establish the communications link and transfer data with the INS8250. The  $\overline{\text{DSR}}$  signal is a MODEM-control function input whose condition can be tested by the CPU by reading bit 5 (DSR) of the MODEM status register. Bit 1 (DDSR) of the MODEM status register indicates whether the  $\overline{\text{DSR}}$  input has changed state since the previous reading of the MODEM status register.

NOTE: Whenever the DSR bit of the MODEM status register changes state, an interrupt is generated if enabled.

**Received Line Signal Detect ( $\overline{\text{RLSD}}$ ), Pin 38:** When low, it indicates that the data carrier has been detected by the MODEM or data set. The  $\overline{\text{RLSD}}$  signal is a MODEM-control function input whose condition can be tested by the CPU by reading bit 7 (RLSD) of the MODEM status register. Bit 3 (DRLSD) of the MODEM status register indicates whether the  $\overline{\text{RLSD}}$  input has changed state since the previous reading of the MODEM status register.

NOTE: Whenever the RLSD bit of the MODEM status register changes state, an interrupt is generated if enabled.

**Ring Indicator ( $\overline{\text{RI}}$ ), Pin 39:** When low, it indicates that a telephone ringing signal has been received by the MODEM or data set. The  $\overline{\text{RI}}$  signal is a MODEM control function input whose condition can be tested by the CPU by reading bit 6 (RI) of the MODEM status register. Bit 2 (TERI) of the MODEM status register indicates whether the  $\overline{\text{RI}}$  input has changed from a low to a high state since the previous reading of the MODEM status register.

NOTE: Whenever the RI bit of the MODEM status register changes from a high to a low state, an interrupt is generated if enabled.

$V_{CC}$ , Pin 40: +5-volt supply.

$V_{SS}$ , Pin 20: Ground (0-volt) reference.

## OUTPUT SIGNALS

**Data Terminal Ready ( $\overline{\text{DTR}}$ ), Pin 33:** When low, it informs the MODEM or data set that the INS8250 is ready to communicate. The DTR output signal can be set to an active low by programming bit 0 (DTR) of the MODEM control register to a high level. The  $\overline{\text{DTR}}$  signal is set high upon a Master Reset operation.

**Request to Send ( $\overline{\text{RTS}}$ ), Pin 32:** When low, it informs the MODEM or data set that the INS8250 is ready to transmit data. The  $\overline{\text{RTS}}$  output signal can be set to an active low by programming bit 1 (RTS) of the MODEM control register. The  $\overline{\text{RTS}}$  signal is set high upon a Master Reset operation.

**Output 1 ( $\overline{\text{OUT 1}}$ ), Pin 34:** A user-designated output that can be set to an active low by programming bit 2 ( $\overline{\text{OUT 1}}$ ) of the MODEM control register to a high level. The  $\overline{\text{OUT 1}}$  signal is set high upon a Master Reset operation.

**Output 2 ( $\overline{\text{OUT 2}}$ ), Pin 31:** A user-designated output that can be set to an active low by programming bit 3 (OUT 2) of the MODEM control register to a high level. The  $\overline{\text{OUT 2}}$  signal is set high upon a Master Reset operation.

**Chip Select Out (CSOUT), Pin 24:** When high, it indicates that the chip has been selected by active  $\overline{\text{CS0}}$ , CS1, and  $\overline{\text{CS2}}$  inputs. No data transfer can be initiated until the CSOUT signal is a logic 1.

**Driver Disable (DDIS), Pin 23:** Goes low whenever the CPU is reading data from the INS8250. A high-level DDIS output can be used to disable an external transceiver (if used between the CPU and INS8250 on the  $D_7 - D_0$  Data Bus) at all times, except when the CPU is reading data.

**Baud Out ( $\overline{\text{BAUDOUT}}$ ), Pin 15:** 16x clock signal for the transmitter section of the INS8250. The clock rate is equal to the main reference oscillator frequency divided by the specified divisor in the baud generator divisor latches. The  $\overline{\text{BAUDOUT}}$  may also be used for the receiver section by typing this output to the RCLK input of the chip.

**Interrupt (INTRPT), Pin 30:** Goes high whenever any one of the following interrupt sources has an active high condition: Receiver Error Flag; Received Data Available; Transmitter Holding Register Empty; and MODEM Status. The INTRPT signal is reset low upon a Master Reset operation.

**Serial Output (SOUT), Pin 11:** Composite serial data output to the communications link (peripheral, MODEM or data set). The SOUT signal is set to the Marking (logic 1) state upon a Master Reset operation.

## INPUT/OUTPUT SIGNALS

**Data ( $D_7 - D_0$ ) Bus, Pins 1 - 8:** This bus comprises eight TRI-STATE input/output lines. The bus provides bidirectional communications between the

INS8250 and the CPU. Data, control words, and status information are transferred via the  $D_7 - D_0$  data bus.

**External Clock Input/Output (XTAL 1, XTAL 2), Pins 16 and 17:** These two pins connect the main timing reference (crystal or signal clock) to the INS8250.

Register/Signal	Reset Control	Reset State
Receiver Buffer Register	First Word Received	Data
Transmitter Holding Register	Writing into the Transmitter Holding Register	Data
Interrupt Enable Register	Master Reset	All bits Low (0 - 3 forced and 4 - 7 permanent)
Interrupt Identification Register	Master Reset	Bit 0 is High and Bits 1 - 7 Are Permanently Low
Line Control Register	Master Reset	All Bits Low
MODEM Control Register	Master Reset	All Bits Low
Line Status Register	Master Reset	All Bits Low, Except Bits 5 & 6 Are High
MODEM Status Register	Master Reset	Bits 0 - 3 Low
	MODEM Signal Inputs	Bits 4 - 7 — Input Signal
Divisor Latch (low order bits)	Writing into the Latch	Data
Divisor Latch (high order bits)	Writing into the Latch	Data
SOUT	Master Reset	High
BAUDOUT	Writing into Either Divisor Latch	Low
CSOUT	$\overline{ADS}$ Strobe Signal and State of Chip Select Lines	High/Low
DDIS	$DDIS = \overline{CSOUT} \cdot RCLK \cdot \overline{DISTR}$ (AT Master Reset, the CPU sets RCLK and DISTR low.)	High
INTRPT	Master Reset	Low
$\overline{OUT 2}$	Master Reset	High
$\overline{RTS}$	Master Reset	High
$\overline{DTR}$	Master Reset	High
$\overline{OUT 1}$	Master Reset	High
$D_7 - D_0$ Data Bus Lines	In TRI-STATE Mode, Unless $CSOUT \cdot \overline{DISTR} = \text{High}$ or $CSOUT \cdot \overline{DOSTR} = \text{High}$	TRI-STATE DATA (ACE to CPU) DATA (CPU to ACE)

Table 1

Reset Control of Registers and Pinout Signals.



## Programming

When you use Heath software, you will not be concerned with programming the 8250 ACE (asynchronous communications element) in the H88-3. However, this section will be indispensable if you intend to assemble your own program code.

In order to easily program the 8250, you should:

1. Disable all UART interrupts by clearing the interrupt enable register.
2. Set the ACE in its loop-back mode.
3. Program the ACE as you want it.
4. Read a character.
5. Wait two character times.
6. Read a second character.
7. Take the ACE out of the loop-back mode.

### INS8250 ACCESSIBLE REGISTERS

You (the system programmer) may access or control any of the INS8250 registers summarized in Table 1 via the CPU. These registers are used to control INS8250 operations and to transmit and receive data.

#### INS8250 Line Control Register

Specify the format of the asynchronous data communications exchange via the Line Control Register. In addition to controlling the format, you may retrieve the contents of the Line Control Register for inspection. This feature simplifies system programming and eliminates the need for separate storage in system memory of the line characteristics. The contents of the Line Control Register are indicated in Table 2 and are described below.

**Bits 0 and 1:** These two bits specify the number of bits in each transmitted or received serial character. The encoding of bits 0 and 1 is as follows:

Bit 1	Bit 0	Word Length
0	0	5 Bits
0	1	6 Bits
1	0	7 Bits
1	1	8 Bits

**Bit 2:** This bit specifies the number of Stop bits in each transmitted or received serial character. If bit 2 is a logic 0, 1 Stop bit is generated or checked in the transmit or receive data, respectively. If bit 2 is a logic 1 when a 5-bit word length is selected via bits 0 and 1, 1-1/2 Stop bits are generated or checked. If bit 2 is a logic 1 when either a 6-, 7-, or 8-bit word length is selected, 2 Stop bits are generated or checked.

**Bit 3:** This is the Parity Enable bit. When Bit 3 is a logic 1, a Parity bit is generated (transmit data) or checked (receive data) between the last data word bit and Stop bit of the serial data. (The Parity bit is used to produce an even or odd number of 1s when the data word bits and the Parity bit are summed.)

**Bit 4:** This is the Even Parity Select bit. When bit 3 is a logic 1 and bit 4 is a logic 0, an odd number of logic 1s is transmitted or checked in the data word bits and Parity bit. When bit 3 is a logic 1 and bit 4 is a logic 1, an even number of bits is transmitted or checked.

**Bit 5:** This is the Stick Parity bit. When bit 3 is a logic 1 and bit 5 is a logic 1, the Parity bit is transmitted and then detected by the receiver in the opposite state indicated by bit 4.

**Bit 6:** This is the Set Break Control bit. When bit 6 is a logic 1, the serial output (SOUT) is forced to the spacing (logic 0) state and remains there (until reset by a low-level bit 6) regardless of other transmitter activity. This feature enables the CPU to alert a terminal in a computer communications system.

**Bit 7:** This is the Divisor Latch Access bit (DLAB). It must be set high (logic 1) to access the Divisor Latches of the baud rate generator during a Read or Write operation. It must be set low (logic 0) to access the receiver buffer, the transmitter holding register, or the interrupt enable register.

Bit No.	Register Address									
	0 DLAB = 0	0 DLAB = 0	1 DLAB = 0	2	3	4	5	6	0 DLAB = 1	1 DLAB = 1
	Receiver Buffer Register (Read Only)	Transmitter Holding Register (Write Only)	Interrupt Enable Register	Interrupt Identification Register	Line Control Register	MODEM Control Register	Line Status Register	MODEM Status Register	Divisor Latch (LS)	Divisor Latch (MS)
0	Data Bit 0*	Data Bit 0	Enable Received Data Available Interrupt (ERBFI)	"0" if Interrupt Pending	Word Length Select Bit 0 (WLS0)	Data Terminal Ready (DTR)	Data Ready (DR)	Delta Clear to Send (DCTS)	Bit 0	Bit 8
1	Data Bit 1	Data Bit 1	Enable Transmitter Holding Register Empty Interrupt (ETBEI)	Interrupt ID Bit (0)	Word Length Select Bit 1 (WLS1)	Request to Send (RTS)	Overrun Error (OR)	Delta Data Set Ready (DDSR)	Bit 1	Bit 9
2	Data Bit 2	Data Bit 2	Enable Receiver Line Status Interrupt (ELSI)	Interrupt ID Bit (1)	Number of Stop Bits (STB)	Out 1	Parity Error (PE)	Trailing Edge Ring Indicator (TERI)	Bit 2	Bit 10
3	Data Bit 3	Data Bit 3	Enable MODEM Status Interrupt (EMBSI)	0	Parity Enable (PEN)	Out 2	Framing Error (FE)	Delta Receive Line Signal Detect (DRLSD)	Bit 3	Bit 11
4	Data Bit 4	Data Bit 4	0	0	Even Parity Select (EPS)	Loop	Break Interrupt (BI)	Clear to Send (CTS)	Bit 4	Bit 12
5	Data Bit 5	Data Bit 5	0	0	Stick Parity	0	Transmitter Holding Register Empty (THRE)	Data Set Ready (DSR)	Bit 5	Bit 13
6	Data Bit 6	Data Bit 6	0	0	Set Break	0	Transmitter Shift Register Empty (TSRE)	Ring Indicator (RI)	Bit 6	Bit 14
7	Data Bit 7	Data Bit 7	0	0	Divisor Latch Access Bit (DLAB)	0	0	Received Line Signal Detect (RLSD)	Bit 7	Bit 15

\* Bit 0 is the least significant bit. It is the first bit serially transmitted or received.

Table 2  
Summary of INS8250 Accessible Registers.



## 8250 PROGRAMMABLE BAUD RATE GENERATOR

The 8250 contains a programmable baud rate generator that takes the 1.8432 MHz clock and divides it by any divisor from 1 to  $2^{16} - 1$ . The output frequency of the baud generator is  $16 \times$  the baud rate. Two 8-bit latches store the divisor in a 16-bit binary format. These Divisor Latches must be loaded during initialization in order to insure desired operation of the baud rate generator. Upon loading either of the divisor latches, a 16-bit baud counter is immediately loaded. This prevents long counts on initial load.

Table 3 illustrates the standard baud rates and the contents of the LS (least significant) and MS (most significant) latches expressed in byte octal.

BAUD RATE	DIVISOR LATCH	
	(LS)	(MS)
75	000	006
110	027	004
134.5	131	003
150	000	003
300	200	001
600	300	000
1200	140	000
2400	060	000
4800	030	000
9600	014	000
19200	006	000

Table 3  
Baud Rates.

## LINE STATUS REGISTER

This 8-bit register provides status information to the CPU concerning the data transfer. The contents of the line status register are indicated in Table 2 and are described below.

**Bit 0:** This bit is the receiver Data Ready (DR) indicator. Bit 0 is set to a logic 1 whenever a complete incoming character has been received and transferred into the receiver buffer register. Bit 0 may be reset to a logic 0 either by the CPU reading the data in the receiver buffer register or by writing a logic 0 into it from the CPU.

**Bit 1:** This bit is the Overrun Error (OE) indicator. Bit 1 indicates that data in the Receiver Buffer Register was not read by the CPU before the next character was transferred into the receiver buffer register, thereby destroying the previous character. The OE indicator is reset whenever the CPU reads the contents of the line status register.

**Bit 2:** This bit is the Parity Error (PE) indicator. Bit 2 indicates that the received data character does not have the correct even or odd parity, as selected by the even-parity-select bit. The PE bit is set to a logic 1 upon detection of a parity error and is reset to a logic 0 whenever the CPU reads the contents of the line status register.

**Bit 3:** This bit is the Framing Error (FE) indicator. Bit 3 indicates that the received character did not have a valid Stop bit. Bit 3 is set to a logic 1 whenever the Stop bit following the last data bit or parity bit is detected as a zero bit (Spacing level).

**Bit 4:** This bit is the Break Interrupt (BI) indicator. Bit 4 is set to a logic 1 whenever the received data input is held in the Spacing (logic 0) state for longer than a full word transmission time (that is, the total time of Start bit + data bits + Parity + Stop bits).

NOTE: Bits 1 through 4 are the error conditions that produce a Receiver Line Status interrupt whenever any of the corresponding conditions are detected.

**Bit 5:** This bit is the Transmitter Holding Register Empty (THRE) indicator. Bit 5 indicates that the INS8250 is ready to accept a new character for transmission. In addition, this bit causes the INS8250 to issue an interrupt to the CPU when the Transmit Holding Register Empty Interrupt enable is set high. The THRE bit is set to a logic 1 when a character is transferred from the transmitter holding register into the transmitter shift register. The bit is reset to logic 0 concurrently with the loading of the transmitter holding register by the CPU.

**Bit 6:** This bit is the Transmitter Shift Register Empty (TSRE) indicator. Bit 6 is set to a logic 1 whenever the transmitter shift register is idle. It is reset to logic 0 upon a data transfer from the transmitter holding register to the transmitter shift register. Bit 6 is a read-only bit.

**Bit 7:** This bit is permanently set to logic 0.

## INTERRUPT IDENTIFICATION REGISTER

The INS8250 has an on-chip interrupt capability that allows for complete flexibility in interfacing to all the popular microprocessors presently available. In order to provide minimum software overhead during data character transfers, the INS8250 prioritizes interrupts into four levels. The four levels of interrupt conditions are as follows: Receiver Line Status (priority 1); Received Data Ready (priority 2); Transmitter Holding Register Empty (priority 3); and MODEM Status (priority 4).

Information indicating that a prioritized interrupt is pending and the source of that interrupt are stored in the interrupt identification register (refer to Table 4). The interrupt identification register (IIR, when addressed during chip-select time, freezes the highest priority interrupt pending and no other interrupts are acknowledged until the particular interrupt is serviced by the CPU. The contents of the IIR are indicated in Table 2 and are described below.

**Bit 0:** This bit can be used in either a hardwired prioritized or polled environment to indicate whether an interrupt is pending. When bit 0 is a logic 0, an interrupt is pending and the IIR contents may be used as a pointer to the appropriate interrupt service routine. When bit 0 is a logic 1, no interrupt is pending and polling (if used) continues.

**Bits 1 and 2:** These two bits of the IIR are used to identify the highest priority interrupt pending as indicated in table 3.

**Bits 3 through 7:** These five bits of the IIR are always logic 0.

## INTERRUPT ENABLE REGISTER

This 8-bit register enables the four interrupt sources of the INS8250 to separately activate the chip Interrupt (INTRPT) output signal. It is possible to totally disable the interrupt system by resetting bits 0 through 3 of the interrupt enable register. Similarly, by setting the appropriate bits of this register to a logic 1, selected interrupts can be enabled. Disabling the interrupt system inhibits the interrupt identification register and the active (high) INTRPT output from the chip. All other system functions operate in their normal manner, including the setting of the line status and MODEM status registers. The contents of the interrupt enable register are indicated in Table 1 and are described below.

**Bit 0:** This bit enables the Received Data Available Interrupt when set to logic 1. Bit 0 is reset to logic 0 upon completion of the associated interrupt service routine.

**Bit 1:** This bit enables the Transmitter Holding Register Empty Interrupt when set to logic 1. Bit 1 is reset to logic 0 immediately upon reading the Interrupt Identification Register.

**Bit 2:** This bit enables the Receiver Line Status Interrupt when set to logic 1. Bit 2 is reset to logic 0 upon completion of the associated interrupt service routine.

**Bit 3:** This bit enables the MODEM Status Interrupt when set to logic 1. Bit 3 is reset to logic 0 upon completion of the associated interrupt service routine.

**Bits 4 through 7:** These four bits are always logic 0.

## MODEM CONTROL REGISTER

This 8-bit register controls the interface with the MODEM or data set (or a peripheral device emulating a MODEM). The contents of a MODEM control register are indicated in Table 2 and are described below.

**Bit 0:** This bit controls the Data Terminal Ready (DTR) output. When bit 0 is set to a logic 1, the  $\overline{\text{DTR}}$  output is forced to a logic 0. When bit 0 is reset to a logic 0, the  $\overline{\text{DTR}}$  output is forced to a logic 1.

NOTE: The  $\overline{\text{DTR}}$  output of the INS8250 may be applied to an EIA inverting line driver (such as the DS1488) to obtain the proper polarity input at the succeeding MODEM or data set.

**Bit 1:** This bit controls the Request to Send ( $\overline{\text{RTS}}$ ) output. Bit 1 affects the  $\overline{\text{RTS}}$  output in a manner identical to that described above for bit 0.

**Bit 2:** This bit controls the Output 1 ( $\overline{\text{OUT 1}}$ ) signal, which is an auxiliary user-designated output. Bit 2 affects the  $\overline{\text{OUT 1}}$  output in a manner identical to that described above for bit 0.

**Bit 3:** This bit controls the Output 2 ( $\overline{\text{OUT 2}}$ ) signal, which is an auxiliary user-designated output. Bit 3 affects the  $\overline{\text{OUT 2}}$  output in a manner identical to that described above for bit 0.



**Bit 4:** This bit provides a loopback feature for diagnostic testing of the INS8250. When bit 4 is set to logic 1, the following occur: the transmitter Serial Output (SOUT) is set to the Marking (logic 1) state; the receiver Serial Input (SIN) is disconnected; the output of the transmitter shift register is "looped back" into the receiver shift register input; the four MODEM control inputs ( $\overline{\text{CTS}}$ ,  $\overline{\text{DSR}}$ ,  $\overline{\text{RLSD}}$ , and  $\overline{\text{RI}}$ ) are disconnected; and the four MODEM control outputs ( $\overline{\text{DTR}}$ ,  $\overline{\text{RTS}}$ ,  $\overline{\text{OUT 1}}$ , and  $\overline{\text{OUT 2}}$ ) are internally connected to the four MODEM Control inputs. In the diagnostic mode, data that is transmitted is immediately received. This feature allows the processor to verify the transmit- and receive-data paths of the INS8250.

In the diagnostic mode, the receiver and transmitter interrupts are fully operational. The MODEM control Interrupts are also operational but the interrupt sources are now the lower four bits of the MODEM control register instead of the four MODEM control inputs. The interrupts are still controlled by the interrupt enable register.

The INS8250 interrupt system can be tested by writing into the lower six bits of the line status register and the lower four bits of the MODEM status register. Setting any of these bits to a logic 1 generates the appropriate interrupt (if enabled). The resetting of these interrupts is the same as in normal INS8250 operation. To return to this operation, the registers must be reprogrammed for normal operation and then bit 4 must be reset to logic 0.

**Bits 5 through 7:** These bits are permanently set to logic 0.

### MODEM STATUS REGISTER

This 8-bit register provides the current state of the control lines from the MODEM (or peripheral device) to the CPU. In addition to this current-state information, four bits of the MODEM status register provide

change information. These bits are set to a logic 1 whenever a control input from the MODEM changes state. They are reset to logic 0 whenever the CPU reads the MODEM status register.

The contents of the MODEM status register are indicated in Table 2 and are described below.

**Bit 0:** This bit is the Delta Clear to Send (DCTS) indicator. Bit 0 indicates that the  $\overline{\text{CTS}}$  input to the chip has changed state since the last time it was read by the CPU.

**Bit 1:** This bit is the Delta Data Set Ready (DDSR) indicator. Bit 1 indicates that the  $\overline{\text{DSR}}$  input to the chip has changed state since the last time it was read by the CPU.

**Bit 2:** This bit is the Trailing Edge of Ring Indicator (TERI) detector. Bit 2 indicates that the  $\overline{\text{RI}}$  input to the chip has changed from an On (logic 1) to an Off (logic 0) condition.

**Bit 3:** This bit is the Delta Received Line Signal Detector (DRLSD) indicator. Bit 3 indicates that the  $\overline{\text{RLSD}}$  input to the chip has changed state.

NOTE: Whenever bit 0, 1, 2, or 3 is set to logic 1, a MODEM Status interrupt is generated.

**Bit 4:** This bit is the complement of the Clear to Send ( $\overline{\text{CTS}}$ ) input.

**Bit 5:** This bit is the complement of the Data Set Ready ( $\overline{\text{DSR}}$ ) input.

**Bit 6:** This bit is the complement of the Ring Indicator ( $\overline{\text{RI}}$ ) input.

**Bit 7:** This bit is the complement of the Received Line Signal Detect ( $\overline{\text{RLSD}}$ ) input.

Interrupt Identification Register			Interrupt Set and Reset Functions			
Bit 2	Bit 1	Bit 0	Priority Level	Interrupt Flag	Interrupt Source	Interrupt Reset Control
0	0	1	—	None	None	—
1	1	0	Highest	Receiver Line Status	Overrun Error or Parity Error or Framing Error or Break Interrupt	Reading the Line Status Register
1	0	0	Second	Received Data Available	Receiver Data Available	Reading the Receiver Buffer Register
0	1	0	Third	Transmitter Holding Register Empty	Transmitter Holding Register Empty	Reading the IIR Register (if source of interrupt) or Writing into the Transmitter Holding Register
0	0	0	Fourth	MODEM Status	Clear to Send or Data Set Ready or Ring Indicator or Received Line Signal Detect	Reading the MODEM Status Register

Table 4  
Interrupt Control Functions.



# CUSTOMER SERVICE

## REPLACEMENT PARTS

Please provide complete information when you request replacements from either the factory or Heath Electronic Centers. Be certain to include the **HEATH** part number exactly as it appears in the parts list.

## ORDERING FROM THE FACTORY

Print all of the information requested on the parts order form furnished with this product and mail it to Heath. For telephone orders (parts only) dial 616 982-3571. If you are unable to locate an order form, write us a letter or card including:

- Heath part number.
- Model number.
- Date of purchase.
- Location purchased or invoice number.
- Nature of the defect.
- Your payment or authorization for COD shipment of parts not covered by warranty.

Mail letters to: Heath Company  
Benton Harbor  
MI 49022  
Attn: Parts Replacement

**Retain original parts until you receive replacements. Parts that should be returned to the factory will be listed on your packing slip.**

## OBTAINING REPLACEMENTS FROM HEATH ELECTRONIC CENTERS

For your convenience, "over the counter" replacement parts are available from the Heath Electronic Centers listed in your catalog. Be sure to bring in the original part and purchase invoice when you request a warranty replacement from a Heath Electronic Center.

## TECHNICAL CONSULTATION

Need help with your kit? — Self-Service? — Construction? — Operation? — Call or write for assistance. you'll find our Technical Consultants eager to help with just about any technical problem except "customizing" for unique applications.

The effectiveness of our consultation service depends on the information you furnish. Be sure to tell us:

- The Model number and Series number from the blue and white label.
- The date of purchase.
- An exact description of the difficulty.
- Everything you have done in attempting to correct the problem.

Also include switch positions, connections to other units, operating procedures, voltage readings, and any other information you think might be helpful.

**Please do not send parts for testing**, unless this is specifically requested by our Consultants.

Hints: Telephone traffic is lightest at midweek — please be sure your Manual and notes are on hand when you call.

Heathkit Electronic Center facilities are also available for telephone or "walk-in" personal assistance.

## REPAIR SERVICE

Service facilities are available, if they are needed, to repair your completed kit. (Kits that have been modified, soldered with paste flux or acid core solder, cannot be accepted for repair.)

**If it is convenient, personally deliver your kit to a Heathkit Electronic Center. For warranty parts replacement, supply a copy of the invoice or sales slip.**

If you prefer to ship your kit to the factory, attach a letter containing the following information directly to the unit:

- Your name and address.
- Date of purchase and invoice number.
- Copies of all correspondence relevant to the service of the kit.
- A brief description of the difficulty.
- Authorization to return your kit COD for the service and shipping charges. (This will reduce the possibility of delay.)

Check the equipment to see that all screws and parts are secured. (Do not include any wooden cabinets or color television picture tubes, as these are easily damaged in shipment. Do not include the kit Manual.) Place the equipment in a strong carton with at least **THREE INCHES** of *resilient* packing material (shredded paper, excelsior, etc.) on all sides. Use additional packing material where there are protrusions (control sticks, large knobs, etc.). If the unit weighs over 15 lbs., place this carton in another one with 3/4" of packing material between the two.

Seal the carton with reinforced gummed tape, tie it with a strong cord, and mark it "Fragile" on at least two sides. Remember, the carrier will not accept liability for shipping damage if the unit is insufficiently packed. Ship by prepaid express, United Parcel Service, or insured Parcel Post to:

Heath Company  
Service Department  
Benton Harbor, Michigan 49022

HEATH

**Schlumberger**

**HEATH COMPANY • BENTON HARBOR, MICHIGAN**  
***THE WORLD'S FINEST ELECTRONIC EQUIPMENT IN KIT FORM***

LITHO IN U.S.A.