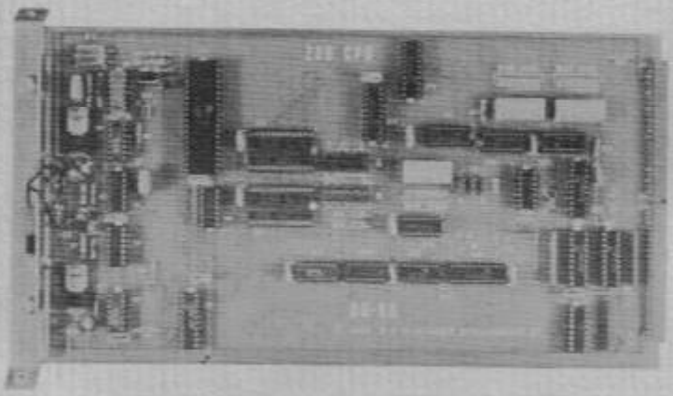
Instruction
Manual

DG-80
Z80 CPU



Post Office Box 1124 • 1827 South Armstrong
Denison, Texas 75020 • (214) 465-7805

R47-6977-1

5/80

TABLE OF CONTENTS

WARRANTY

The DG-80 CPU is warranted for a period of ninety (90) days from the date of purchase to be free from defects in material and workmanship. Should this product fail to perform satisfactorily, arrangements should be made with D-G Electronic Developments Co. for warranty service as follows.

Return of the DG-80 is subject to the issuance of a RETURN MERCHANDISE AUTHORIZATION NUMBER by D-G Electronic Developments Co.. This RMA number must be clearly visible on the outside of the returned package which must be returned freight pre-paid. FAILURE TO CONFORM WITH THIS PROCEDURE WILL RESULT IN REFUSAL TO ACCEPT SAID PACKAGE UPON ARRIVAL.

D-G Electronic Developments will, at our option, repair or replace defective units received during the warranty period. This warranty is invalid if the product has been misused or modified. Warranty is limited to replacement of defective parts and no responsibility is assumed for damage to other equipment.

THIS WARRANTY IS MADE IN LIEU OF ALL OTHER WARRANTIES EXPRESSED OR IMPLIED.

INTRODUCTION

The D-G Electronic Developments Company Model DG-80 is an advanced Central Processing Unit for the Heath H8® microcomputer. The DG-80 utilizes the powerful Z80 microprocessor and provides the following features and options:

- Compatible with Heath H8® hardware and software*
- Enhanced Instruction Set — 158 instructions including all 78 instructions of the 8080A
- Provisions for on-board memory up to 8K ROM/EPROM or 4K RAM
- Jump-On-Reset to any 1K boundary
- Operation up to 4 MHz (2.048 MHz standard)
- DIP-switch selectable wait states for any or all 8K blocks of system memory
- All Z80 interrupt response modes available
- Interrupt Acknowledge and Dynamic Memory Refresh signals available on the bus
- Frequently selected options by DIP-switch or solderless jumpers
- Machined contact gold sockets for on-board memory provide long life and enhanced reliability
- Many advanced features for future expansion

* Due to signal differences between the 8080A and the Z80 microprocessors, the DG-80 is not compatible with the single-step feature offered by the Heath® PAM-8 Panel Monitor and the Heath® Console Debugger. Note however, that other features of the DG-80 (i.e. the jump-on-reset feature and the flexible on-board EPROM capacity) will allow the sophisticated user to employ advanced software from vendors other than Heath®. Many vendors supply software which include software debugging utilities with (DG-80 compatible) single step in addition to assemblers, disassemblers, and monitors. For the slightly less sophisticated user, D-G Electronic Developments Co. will soon make available products to allow the average DG-80 user to utilize much of this available software.

SPECIFICATIONS

Microprocessor	Z80A (158 instructions including all 8080A instructions)
Computer Interface	Heath H50 Bus (modified). All inputs 1 TTL load or less. Output Buffers-74LS240.
Clock Standard	2.048 MHz
Optional	Up to 4.000 MHz
Interrupts Standard	Seven, priority vectored
Optional	Z80 Mode 1 or Mode 2
On-Board Memory	2 on-board sockets allow up to 8K ROM/EPROM or up to 4K RAM
Sockets	All IC's socketed
Switches	All frequently changed options determined by solderless jumpers or DIP switches
Power Requirements (Note 1)	+7◆+12 VDC at 540 mA +14◆+20 VDC at 50 mA −14◆−20 VDC at 30 mA
Manuals and Documentation	Includes extensive instruction manual and Mostek Z80 Programming Manual

Note 1: Currents given are typical with Heath PAM-8 ROM (2308) on board and may vary according to RAM or ROM used. The DG-80 does not include the PAM-8 ROM. Maximum currents (U10 and U11 combined) for the ROM sockets are:

+5 VDC @ 200mA
+12 VDC @ 100mA

INSTALLATION INSTRUCTIONS

Make sure that the H8® power switch is in the **OFF** position and the computer line cord is unplugged.

1) Remove the top cover and tie bracket from the H8® computer. Remove the 6-32 × ¼" screw that holds the bottom of the CPU board.

2) Disconnect the 5 pin connector (S201) from the Heath® CPU and remove the Heath® CPU board from the computer.

NOTE: In the following step, you will remove the Heath PAM-8 ROM (IC204) from your Heath CPU board. Before removing the ROM, **identify pin 1** so that the ROM will be installed properly on the DG-80. Remember that care should be exercised when MOS devices are handled to avoid damage to them.

3) Remove IC204 from your Heath CPU board. Install this ROM at U10 on the DG-80 CPU board. Be sure all of the pins are straight and pin 1 is aligned properly before pushing the ROM into the socket.

4) Make sure the following switches and jumpers are positioned:

WAIT ADDR	All switches OFF	
MEM ADDR	ØK ON, all others OFF	
MEM SPACE	ØK ON, all others OFF	
Jumper J9	No jumpers installed	
Jumper "A"	PIN 18	GND
	PIN 19	+12V
	PIN 21	−5V
Jumper "B"	No jumpers installed	

5) Double-check the above jumpers and switches.

6) Install the DG-80 into the H8® computer. Be sure to align Pin 1 of P201 properly and plug S201 (the 5 pin connector) onto P201.

7) Re-install the tie bracket and computer top cover. Install the 6-32 × ¼" screw in the bottom of the CPU board.

8) Make sure the H8 power switch is in the "OFF" position and then plug in the computer.

9) Push the H8 power switch to the "ON" position. The PWR, RUN, MON, and ION LED's should light and the nine LED displays should indicate random numbers. If this is not the case, turn the power switch to "OFF", unplug the computer and recheck this installation procedure.

10) If step 9 is successful, proceed to the Memory Test Routine on page 9 of your H8® operation manual or the H8® Memory Test given in the H17 or WH 17 Disk System operating manual. The memory test will allow you to check the performance of your computer system before putting it in use.

NOTE: IF DIFFICULTY IS ENCOUNTERED AT STEP 9, REFER TO APPENDIX A: OPERATING NOTES. This section discusses operational differences between the Heath 8080 CPU and the DG-80. If this section does not explain your symptoms refer to the Warranty section of this manual.

DG-80 BUS IDENTIFICATION

PIN # 0	Gnd	PIN # 25	$\overline{BUSAK}$
PIN # 1	Gnd	PIN # 26	I/O R
PIN # 2	−18 Volts	PIN # 27	$\overline{BUSRQ}$
PIN # 3	$\overline{INT_3}$	PIN # 28	MEMR
PIN # 4	$\overline{INT_4}$	PIN # 29	$\overline{RESET}$
PIN # 5	$\overline{INT_5}$	PIN # 30	$\overline{A_0}$
PIN # 6	$\overline{INT_6}$	PIN # 31	$\overline{A_1}$
PIN # 7	$\overline{INT_7}$	PIN # 32	$\overline{A_2}$
PIN # 8	$\overline{INT_1}$	PIN # 33	$\overline{A_3}$
PIN # 9	$\overline{INT_2}$	PIN # 34	$\overline{A_4}$
PIN # 10	$\overline{D_0}$	PIN # 35	$\overline{A_5}$
PIN # 11	$\overline{D_1}$	PIN # 36	$\overline{A_6}$
PIN # 12	$\overline{D_2}$	PIN # 37	$\overline{A_7}$
PIN # 13	$\overline{D_3}$	PIN # 38	$\overline{A_8}$
PIN # 14	$\overline{D_4}$	PIN # 39	$\overline{A_9}$
PIN # 15	$\overline{D_5}$	PIN # 40	$\overline{A_{10}}$
PIN # 16	$\overline{D_6}$	PIN # 41	$\overline{A_{11}}$
PIN # 17	$\overline{D_7}$	PIN # 42	$\overline{A_{12}}$
PIN # 18	$\overline{INTA}$	PIN # 43	$\overline{A_{13}}$
PIN # 19	M1	PIN # 44	$\overline{A_{14}}$
PIN # 20	$\overline{WAIT}$	PIN # 45	$\overline{A_{15}}$
PIN # 21	I/O W	PIN # 46	$\overline{ROM DISABLE}$
PIN # 22	Φ	PIN # 47	+18 Volts
PIN # 23	MEMW	PIN # 48	+8 Volts
PIN # 24	RFSH	PIN # 49	+8 Volts

$\overline{A_0}$ - $\overline{A_{15}}$ **ADDRESS BUS** — Tri-state output, active low. Provides address to memory (up to 64K words) and Input/Output Devices (up to 256 devices addressed by A_0-A_7) A_0 is the least significant bit. During refresh time, A_0-A_6 contain a valid refresh address.

$\overline{D_0}$ - $\overline{D_7}$ **DATA BUS** — bidirectional - Input and Output active low. This 8 bit data bus is used for data exchanges with memory and I/O devices.

$\overline{INT_1}$ - $\overline{INT_7}$ **INTERRUPT REQUEST** — Vectored priority interrupt request inputs, active low. The various interrupt modes available are discussed in the Interrupt Response section of this manual.

$\overline{INTA}$ **INTERRUPT ACKNOWLEDGE** — Output active low. Signals that the CPU has accepted an interrupt request.

Φ **CPU CLOCK OUTPUT** — This signal replaces the Φ_2 clock found on the Heath H50 Bus.

M1 **INSTRUCTION FETCH CYCLE** — Output active high. Indicates that the current machine cycle is the Op-code fetch cycle of instruction execution.

$\overline{WAIT}$ **WAIT STATE REQUEST** — Input active low. Indicates that memory or I/O devices are not ready for data transfer thus placing the CPU in a WAIT state until transfer can be completed. Replaces the READY INPUT signal on the Heath H50 Bus.

I/O W **INPUT/OUTPUT WRITE** — Active high. Indicates that the current cycle is an I/O Write cycle.

MEMW **MEMORY WRITE** — Active high. Indicates that the current cycle is a Memory Write Cycle.

RFSH **REFRESH** — Output active high. Indicates that A_0-A_6 contain a valid refresh address for use with dynamic memory refresh operations.

$\overline{BUSRQ}$ **BUS REQUEST** — Input active low. Used by peripheral devices to request control of system bus for use in DMA operations. Replaces the HOLD signal on the Heath H50 Bus.

$\overline{\text{BUSAK}}$	**BUS ACKNOWLEDGE** — Output active low. Used by the CPU to indicate to peripheral devices that they may take control of the system bus. The following bus lines will be tri-stated at this time: $\overline{A}_0$-$\overline{A}_{15}$, $\overline{D}_0$-$\overline{D}_7$, MEMR, MEMW, I/O R, I/O W, $\overline{\text{INTA}}$, AND M1. Replaces the HOLD ACKNOWLEDGE signal on the Heath H50 Bus.
I/O R	**INPUT/OUTPUT READ** — Output active high. Indicates that the current cycle is an I/O Read cycle.
MEMR	**MEMORY READ** — Output active high. Indicates that the current cycle is a Memory Read Cycle.
$\overline{\text{RESET}}$	**RESET** — Output active low. System Reset output to peripherals.
$\overline{\text{ROM DISABLE}}$	**ROM DISABLE** — Input active low. Allows peripheral devices to disable memory on the DG-80 CPU board.

See the Heath H8® operation manual for information on power supply specifications.

ON-BOARD MEMORY OPTIONS

The DG-80 CPU allows for up to 8K bytes of on-board ROM/EPROM, up to 4K bytes of on-board RAM or combinations of ROM and RAM up to 4K ROM and 2K RAM. This memory is placed in sockets U10 (Memory A) and U11 (Memory B). The options associated with this memory are controlled by jumpers "A", "B", and J9 as well as the "MEM ADDR" switch and the "MEM SPACE" switch.

The DG-80 will accept the following memory devices: 2308 (MK30000)-1K × 8 ROM, 2708 (2758)-1K × 8 EPROM, MK 4118-1K × 8 RAM, 2716 (TMS2516)-2K × 8 EPROM, TMS4016-2K × 8 RAM or many other devices which adhere to the pin-out and function of these devices. All of the above devices have the same pinout with the exception of pins 18, 19, and 21; therefore the function of these pins is controlled by jumper "A" for U10 (Memory A) and jumper "B" for U11 (Memory B).

The following table summarizes the jumper positions for the above devices.

NOTE: DO NOT INSTALL MORE THAN ONE JUMPER FOR EACH PIN (18, 19 and 21) OF THE DEVICE BEING USED AT U10 AND ONE JUMPER FOR EACH PIN OF THE DEVICE BEING USED AT U11.

JUMPER POSITIONS FOR COMMON MEMORY DEVICES

DEVICE	FUNCTION	18**	19	21**
2308	1K × 8 ROM	GND	+12V	−5V
MK30000	1K × 8 ROM	GND	NC	NC
2316*	2K × 8 ROM	GND	A10	+5V
2708	1K × 8 EPROM	GND	+12V	−5V
TMS2508	1K × 8 EPROM	CS	NC	+5V
2758†	1K × 8 EPROM	CS	+5V	+5V
2716 (INTEL)	2K × 8 EPROM	CS	A10	+5V
TMS2516	2K × 8 EPROM	CS	A10	+5V
2732	4K × 8 EPROM	CS	A10	A11
MK4118	1K × 8 RAM	CS	+5V	MW
TMS4016	2K × 8 RAM	CS	A10	MW

NC — No Connection

For other memory devices, see the appropriate manufacturer's data sheets.

* The Intel 2316 ROM has several mask programmed (i.e. factory programmed) CS options. Some of these options may not be compatible with the DG-80. Consult with the ROM vendor for the options chosen on your ROM.

† There are two versions of the Intel 2758. Only the 2758-S1865 is compatible with the DG-80.

**The signals CS and MW labelled on the board are active LOW signals but were not labelled as such to enhance readability.

The memory address of the on-board memory is determined by the MEM ADDR and MEM SPACE switches and jumper J9. The starting address of the 8K block occupied by the memory is determined by the MEM ADDR switch. This switch is labelled ØK, 8K, 16K, 24K, 32K, 40K, 48K and 56K. If, for example, the Heath® PAM-8 ROM is being used, the MEM ADDR switch would be set at ØK so that the ROM would occupy the lowest available memory locations.

The MEM SPACE switch is set to determine where in the 8K block chosen by MEM ADDR memory is located. The ØK block is always assigned to Memory A (U10) and the 4K, 5K, 6K and 7K blocks are always assigned to Memory B (U11). The 1K, 2K and 3K blocks may occupy Memory A or Memory B depending on the memory devices used. This choice is made by use of jumper J9. A few examples will clarify switch and jumper selections:

The Heath PAM-8 ROM is a 1K byte ROM. If this device were being used, the MEM ADDR switch would be set to ØK to place on-board memory in the lowest 8K block of the H8 memory. The MEM SPACE switch would be set to ØK to show that the 1K ROM occupies the lowest 1K block of the 8K on-board memory space. No jumpers would be installed in J9 since the ØK space is always assigned to Memory A and the ROM would be plugged in at Memory A (U10).

As another example, suppose the user wished to use a 2716 2K EPROM for program storage (his own custom panel monitor) and wished to use a MK4118 RAM addressed immediately above the EPROM. If the EPROM is to be located at the bottom of memory, then the MEM ADDR switch would again be set to ØK. Since the user has 3K bytes of on-board memory, the ØK, 1K and 2K MEM SPACE switches would be set to the **ON** position. Since the EPROM occupies 2K bytes, jumper J9 would have the 1K position set to **A** (recall that the ØK position is always assigned to Memory A) and the 2716 would be plugged into Memory A (U10). The 2K position of jumper J9 would be set to **B** and the MK4118 plugged into Memory B (U11).

DG-80 JUMPER OPTIONS

The DG-80 CPU has many jumper selectable operating modes. A detailed discussion of all modes is beyond the scope of this manual, but a brief description of each of the jumpers and their function will be given. It is strongly recommended that the advanced user of the DG-80 acquire a copy of the Mostek MICROCOMPUTER DATA BOOK if he requires additional information on Z80 operation.

Note: Many of the DG-80 options are not compatible with the Heath® PAM-8 Panel Monitor and certain other Heath peripherals. This fact will be noted with each option.

J1 This jumper connects the DG-80 RFS signal to the system bus, pin 24. Pin 24 of the Heath Bus was not assigned and therefore chosen for this function. If the user already employs pin 24 of the bus for some function and does not desire this Dynamic Memory Refresh signal, then jumper J1 should be removed.

J2 This jumper connects the DG-80 $\overline{\text{INTA}}$ signal to the system bus, pin 18. This pin was not assigned on the Heath Bus and therefore chosen for this function. If the user already employs pin 18 of the bus and does not desire the $\overline{\text{INTA}}$ function, then this jumper should be removed.

J3 Connects $\overline{\text{INT 2}}$ to the system bus.

J4 Connects $\overline{\text{INT 1}}$ to the system bus.

J5 When pin 1 is connected to pin 2 and J4 is connected, INT 1 is tied to the Heath Bus pin 8. If J4 is connected and pin 1 of J5 is connected to pin 3, then pin 8 of the Heath Bus becomes the $\overline{\text{NMI}}$ (non-maskable Interrupt) input. **Warning: The NMI function is not compatible with the PAM-8 Panel Monitor.**

J6 Determines whether the BUSAK signal is active HIGH or active LOW

1•2	Active Low ($\overline{\text{BUSAK}}$)
1•3	Active High (BUSAK)

J7 Used in selection of DG-80 CPU Interrupt response. When pin 1 is connected to pin 2, the CPU operates in Z80 Interrupt Response Mode 0 (this is the normal Heath-compatible mode). When pin 1 is tied to pin 3, the CPU will operate in Z80 Interrupt Response Mode 1 or 2. See the Z80 CPU Interrupt Response section in this manual for information on these interrupt response modes. **Warning: The Z80 Interrupt Response Modes 1**

J8	Used in conjuction with J7 to determine DG-80 CPU interrupt response. When pin 1 is connected to pin 3, the CPU will operate in either Z80 Interrupt Response Mode 0 or 1. When pin 1 is connected to pin 2, an $\overline{\text{INTA}}$ signal from the CPU will allow data on the System Data Bus to be placed on the CPU Data Bus. This allows the CPU to operate in Z80 CPU Interrupt Response Mode 2. See the Z80 CPU Interrupt Response section in this manual for information on these interrupt response modes.
J9	Used in conjunction with the MEM SPACE switch to determine addressing of on-board memory. See the "ON BOARD MEMORY SELECTION" section of this manual for information.
CLK	This jumper is used to select between 2 and 4 MHz operation of the DG-80 CPU. The oscillator made up of U2 and its associated components oscillates at a frequency determined by Y1. As distributed, Y1 is a 4.096 MHz crystal. This frequency is divided by 2 using flip-flop U3 to provide the 2.048 MHz clock frequency to the CPU. Note that this is the standard clock frequency used by the Heath H8-8080A CPU. If operation at 4 MHz is desired, Y1 should be changed to a 4.000 MHz crystal and jumper CLK set to the 4 MHz position. **Warning: D-G Electronic Developments Co. DOES NOT recommend operation of the CPU at frequencies above 4.000 MHz.** NOTE: The Heath H-17 Disk System was not designed to operate at a system clock frequency of 4 MHz.

WAIT STATE SELECTION

When the DG-80 is operated at clock frequencies greater than 2.048 MHz, problems may be encountered with existing slow memory in the system. To avoid this problem, the DG-80 allows one wait state (equivalent to one t-cycle) to be inserted when accessing memory. Wait states may be selected for any or all 8K blocks of memory in the computer memory space by means of the DIP switch labelled WAIT ADDR on the upper right-hand corner of the CPU board.

To illustrate the use of this feature, assume the user has one 16K memory board which will operate at 4 MHz and one 16K board that will only operate at 2.048 MHz but wishes to operate the system at 4 MHz. He might address the fast 16K board to occupy the 8K and 16K blocks of memory and the "slow" board to occupy the 24K and 32K blocks. Wait states would then be selected by use of the DG-80 WAIT ADDR switch only for the 24K and 32K blocks. Note that in this example the "fast" memory is located in the lower memory locations where it will probably be accessed more often and thus maximize overall system speed.

INSTRUCTION SET

The Z80 CPU executes 158 different instruction types including the 78 instructions of the 8080A used in the Heath H8 computer. The 78 common instructions may not be immediately apparent upon inspection of the Z80 PROGRAMMING MANUAL due to the fact that the makers of the Z80 chose to use a new set of instruction mnemonics. To aid the user in becomming acquainted with the Z80 instruction set, we have included a table of the 8080A instruction set along with the Z80 mnemonics for each instruction (see accompanying table). By studying these common instructions in the Z80 PROGRAMMING MANUAL, the user will begin to familiarize himself with the format of the manual and the conventions used in the Z80 instruction set.

The opcodes for the instructions found in the Z80 PROGRAMMING MANUAL are given in binary and hexadecimal. Since many users of the DG-80 are accustomed to the octal notation suggested by Heath, a brief review of binary to octal conversion will be given.

An eight bit binary number may be grouped and converted to its octal equivalent in the following manner:

$$\underbrace{11}_{3}\quad\underbrace{111}_{7}\quad\underbrace{111}_{7}\qquad\qquad \begin{array}{l}\text{Binary}\\ \\ \text{Octal Equivalent}\end{array}$$

For example, on page 2-82 of the Z80 PROGRAMMING MANUAL may be found the HALT instruction. The binary opcode for this instruction is converted to octal in the following way:

$$\underbrace{01}_{1}\quad\underbrace{110}_{6}\quad\underbrace{110}_{6}\qquad\qquad \begin{array}{l}\text{Binary}\\ \\ \text{Octal Equivalent}\end{array}$$

For further information concerning the instruction set of the Z80 microprocessor we refer the user to the Z80 PROGRAMMING MANUAL.

8080A INSTRUCTIONS WITH EQUIVALENT Z80 MNEMONICS

8080A INSTRUCTION	BYTES	8080A CLOCK PERIODS	Z80 INSTRUCTION	Z80 CLOCK PERIODS
ACI Data	2	7	ADC Data	7
ADC Reg	1	4	ADC Reg	4
ADC M	1	7	ADC (HL)	7
ADD Reg	1	4	ADD Reg	4
ADD M	1	7	ADD (HL)	7
ADI Data	2	7	ADD Data	7
ANA Reg	1	4	AND Reg	4
ANA M	1	7	AND (HL)	7
ANI Data	2	7	AND Data	7
CALL Label	3	17	CALL Label	17
CC Label	3	11 or 17	CALL C, Label	10 or 17
CM Label	3	11 or 17	CALL M, Label	10 or 17
CMA	1	4	CPL	4
CMC	1	4	CCF	4
CMP Reg	1	4	CP Reg	4
CMP M	1	7	CP (HL)	7
CNC Label	3	11 or 17	CALL NC, Label	10 or 17
CNZ Label	3	11 or 17	CALL NZ, Label	10 or 17
CP Label	3	11 or 17	CALL P, Label	10 or 17
CPE Label	3	11 or 17	CALL PE, Label	10 or 17
CPI Data	2	7	CP Data	7
CPO Label	3	11 or 17	CALL PO, Label	10 or 17
CZ Label	3	11 or 17	CALL Z, Label	10 or 17
DAA	1	4	DAA	4
DAD RP	1	10	ADD HL, RP	11
DCR Reg	1	5	DEC Reg	4
DCR M	1	10	DEC (HL)	11
DCX RP	1	5	DEC RP	5
DI	1	4	DI	4
EI	1	4	EI	4
HLT	1	7	HALT	4
IN Port	2	10	IN A, Port	10
INR Reg	1	5	INC Reg	4
INR M	1	10	INC (HL)	11
INX RP	1	5	INC RP	6
JC Label	3	10	JP C, Label	10
JM Label	3	10	JP M, Label	10
JMP Label	3	10	JP Label	10
JNC Label	3	10	JP NC, Label	10
JNZ Label	3	10	JP NZ, Label	10
JP Label	3	10	JP P, Label	10
JPE Label	3	10	JP PE, Label	10
JPO Label	3	10	JP PO, Label	10
JZ Label	3	10	JP Z, Label	10
LDA Addr	3	13	LD A, (Addr)	13
LDAX RP	1	7	LD A, (RP)	7
LHLD Addr	3	16	LD HL, (Addr)	16
LXI RP, Data 16	3	10	LD RP, Data 16	10
MOV Reg, Reg	1	5	LD Reg, Reg	4
MOV M, Reg	1	7	LD (HL), Reg	7
MOV Reg, M	1	7	LD Reg, (HL)	7
MVI Reg, Data	2	7	LD Reg, Data	7
MVI M, Data	2	10	LD (HL), Data	10

8080A INSTRUCTION	BYTES	8080A CLOCK PERIODS	Z80 INSTRUCTION	Z80 CLOCK PERIODS
ORA Reg	1	4	OR Reg	4
ORA M	1	7	OR (HL)	7
ORI Data	2	7	OR Data	7
OUT Port	2	10	OUT PORT, A	11
PCHL	1	5	JP (HL)	4
POP RP	1	10	POP PR	10
PUSH RP	1	11	PUSH PR	11
RAL	1	4	RLA	4
RAR	1	4	RRA	4
RC	1	5 or 11	RET C	5 or 11
RET	1	10	RET	10
RLC	1	4	RLCA	4
RM	1	5 or 11	RET M	5 or 11
RNC	1	5 or 11	RET NC	5 or 11
RNZ	1	5 or 11	RET Z	5 or 11
RP	1	5 or 11	RET P	5 or 11
RPE	1	5 or 11	RET PE	5 or 11
RPO	1	5 or 11	RET PO	5 or 11
RRC	1	4	RRCA	4
RST N	1	11	RST N	11
RZ	1	5 or 11	RET Z	5 or 11
SBB Reg	1	4	SBC Reg	4
SBB M	1	7	SBC (HL)	7
SBI Data	2	7	SBC Data	7
SHLD Addr	3	16	LD (Addr), HL	16
SPHL	1	5	LD SP, HL	6
STA Addr	3	13	LD (Addr), A	13
STAX RP	1	7	LD (RP), A	7
STC	1	4	SCF	4
SUB Reg	1	4	SUB Reg	4
SUB M	1	7	SUB (HL)	7
SUI Data	2	7	SUB Data	7
XCHG	1	4	EX DE, HL	4
XRA Reg	1	4	XOR Reg	4
XRA M	1	7	XOR (HL)	7
XRI Data	2	7	XOR Data	7
XTHL	1	18	EX (SP), HL	19

Refer to the appropriate programming manual for an explanation of the symbols and abbreviations used in this table.

NOTE: If you are writing assembly language programs and intend to use the Heath assembler, you must only use 8080A instructions. The Heath assembler (and many other assemblers available) will not handle the added Z80 instructions and are not compatible with the Z80 mnemonics given in the Z80 PROGRAMMING MANUAL.

2.0 Z80-CPU ARCHITECHURE

A block diagram of the internal architecture of the Z80-CPU is shown in Figure 2.0-1 The diagram shows all of the major elements in the CPU and it should be referred to throughout the following description.

Z80-CPU BLOCK DIAGRAM

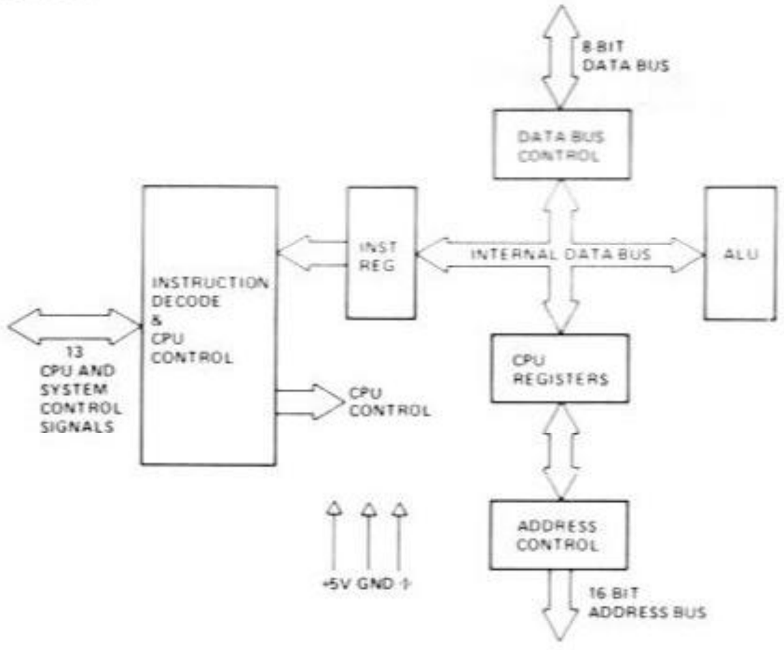


FIGURE 2.0-1

2.1 CPU REGISTERS

The Z80-CPU contains 208 bits of R/W memory that are accessible to the programmer. Figure 2.0-2 illustrates how this memory is configured into eighteen 8-bit registers and four 16-bit registers. All Z80 registers are implemented using static RAM. The registers include two sets of six general purpose registers that may be used individually as 8-bit registers or in pairs as 16-bit registers. There are also two sets of accumulator and flag registers.

Special Purpose Registers

1. **Program Counter (PC).** The program counter holds the 16-bit address of the current instruction being fetched from memory. The PC is automatically incremented after its contents have been transferred to the address lines. When a program jump occurs the new value is automatically placed in the PC, overriding the incrementer.

2. **Stack Pointer (SP).** The stack pointer holds the 16-bit address of the current top of a stack located anywhere in external system RAM memory. The external stack memory is organized as a last-in first-out (LIFO) file. Data can be pushed onto the stack from specific CPU registers or popped off of the stack into specific CPU registers through the execution of PUSH and POP instructions. The data popped from the stack is always the last data pushed onto it. The stack allows simple implementation of multiple level interrupts, unlimited subroutine nesting and simplification of many types of data manipulation.

Z80-CPU REGISTER CONFIGURATION

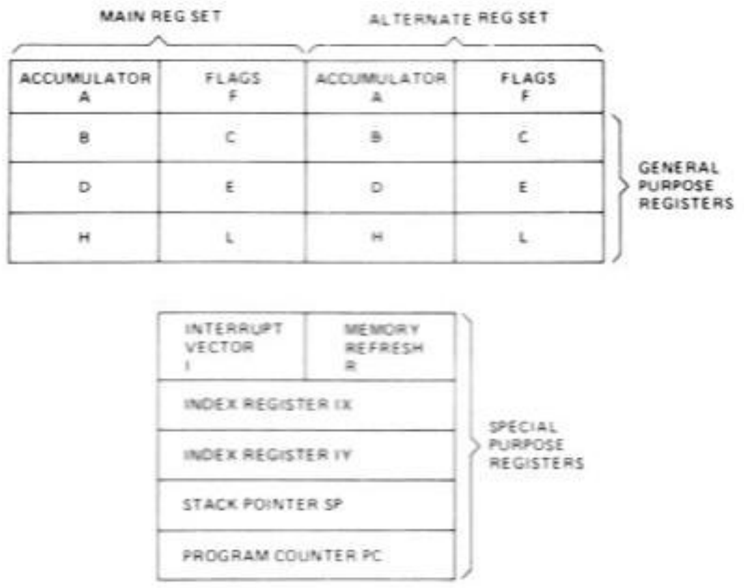


FIGURE 2.0-2

3. **Two Index Registers (IX & IY).** The two independent index registers hold a 16-bit base address that is used in indexed addressing modes. In this mode, an index register is used as a base to point to a region in memory from which data is to be stored or retrieved. An additional byte is included in indexed instructions to specify a displacement from this base. This displacement is specified as a two's complement signed integer. This mode of addressing greatly simplifies many types of programs, especially where tables of data are used.

4. **Interrupt Page Address Register (I).** The Z80-CPU can be operated in a mode where an indirect call to any memory location can be achieved in response to an interrupt. The I Register is used for this purpose to store the high order 8-bits of the indirect address while the interrupting device provides the lower 8-bits of the address. This feature allows interrupt routines to be dynamically located anywhere in memory with absolute minimal access time to the routine.

5. **Memory Refresh Register (R).** The Z80-CPU contains a memory refresh counter to enable dynamic memories to be used with the same ease as static memories. This 7-bit register is automatically incremented after each instruction fetch. The data in the refresh counter is sent out on the lower portion of the address bus along with a refresh control signal while the CPU is decoding and executing the fetched instruction. This mode of refresh is totally transparent to the programmer and does not slow down the CPU operation. The programmer can load the R register for testing purposes, but this register is normally not used by the programmer.

Accumulator and Flag Registers

The CPU includes two independent 8-bit accumulators and associated 8-bit flag registers. The accumulator holds the results of 8-bit arithmetic or logical operations while the flag register indicates specific conditions for 8 or 16-bit operations, such as indicating whether or not the result of an operation is equal to zero. The programmer selects the accumulator and flag pair that he wishes to work with with a single exchange instruction so that he may easily work with either pair.

General Purpose Registers

There are two matched sets of general purpose registers, each set containing six 8-bit registers that may be used individually as 8-bit registers or as 16-bit register pairs by the programmer. One set is called BC, DE, and HL while the complementary set is called BD', DE' and HL'. At any one time the programmer can select either set of registers to work with through a single exchange command for the entire set. In systems where fast interrupt response is required, one set of general purpose registers and an accumulator/flag register may be reserved for handling this very fast routine. Only a simple exchange command need be executed to go between the routines. This greatly reduces interrupt service time by eliminating the requirement for saving and retrieving register contents in the external stack during interrupt or subroutine processing. These general purpose registers are used for a wide range of applications by the programmer. They also simplify programming, especially in ROM based systems where little external read/write memory is available.

2.2 ARITHMETIC & LOGIC UNIT (ALU)

The 8-bit arithmetic and logical instructions of the CPU are executed in the ALU. Internally the ALU communicates with the registers and the external data bus on the internal data bus. The type of functions performed by the ALU include:

Add	Left or right shifts or rotates (arithmetic and logical)
Subtract	Increment
Logical AND	Decrement
Logical OR	Set bit
Logical Exclusive OR	Reset bit
Compare	Test bit

2.3 INSTRUCTION REGISTER AND CPU CONTROL

As each instruction is fetched from memory, it is placed in the instruction register and decoded. The control section performs this function and then generates and supplies all of the control signals necessary to read or write data from or to the registers, controls the ALU and provides all required external control signals.

4.0 CPU TIMING

The Z80-CPU executes instructions by stepping through a very precise set of a few basic operations. These include:

Memory read or write

I/O device read or write

Interrupt acknowledge

All instructions are merely a series of these basic operations. Each of these basic operations can take from three to six clock periods to complete or they can be lengthened to synchronize the CPU to the speed of external devices. The basic clock periods are referred to as T states and the basic operations are referred to as M (for machine) cycles. Figure 4.0-0 illustrates how a typical instruction will be merely a series of specific M and T cycles. Notice that this instruction consists of three machine cycles (M1, M2 and M3). The first machine cycle of any instruction is a fetch cycle which is four, five or six T states long (unless lengthened by the wait signal which will be fully described in the next section). The fetch cycle (M1) is used to fetch the OP code of the next instruction to be executed. Subsequent machine cycles move data between the CPU and memory or I/O devices and they may have anywhere from three to five T cycles (again they may be lengthened by wait states to synchronize the external devices to the CPU). The following paragraphs describe the timing which occurs within any of the basic machine cycles. In section 7, the exact timing for each instruction is specified.

BASIC CPU TIMING EXAMPLE

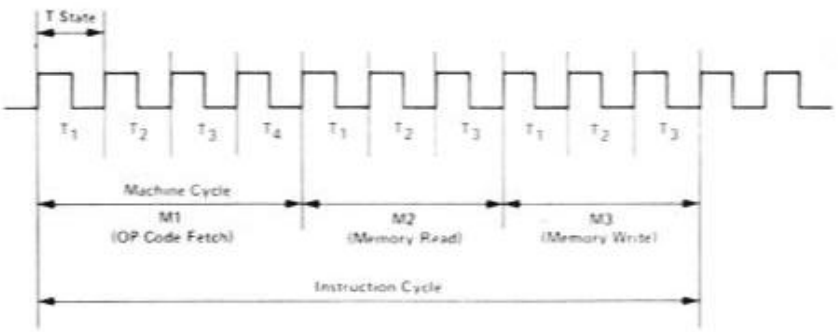


FIGURE 4.0-0

All CPU timing can be broken down into a few very simple timing diagrams as shown in Figure 4.0-1 through 4.0-7. These diagrams show the following basic operations with and without wait states (wait states are added to synchronize the CPU to slow memory or I/O devices).

4.0-1. Instruction OP code fetch (M1 cycle)

4.0-2. Memory data read or write cycles

4.0-3. I/O read or write cycles

4.0-4. Bus Request/Acknowledge Cycle

4.0-5. Interrupt Request/Acknowledge Cycle

4.0-6. Non maskable Interrupt Request/Acknowledge Cycle

4.0-7. Exit from a HALT instruction

INSTRUCTION FETCH

Figure 4.0-1 shows the timing during an M1 cycle (OP code fetch). Notice that the PC is placed on the address bus at the beginning of the M1 cycle. One half clock time later the $\overline{\text{MREQ}}$ signal goes active. At this time the address to the memory has had time to stabilize so that the falling edge of $\overline{\text{MREQ}}$ can be used directly as a chip enable clock to dynamic memories. The $\overline{\text{RD}}$ line also goes active to indicate that the memory read data should be enabled onto the CPU data bus. The CPU samples the data from the memory on the data bus with the rising edge of the clock of state T3 and this same edge is used by the CPU to turn off the $\overline{\text{RD}}$ and $\overline{\text{MREQ}}$ signals. Thus the data has already been sampled by the CPU before the $\overline{\text{RD}}$ signal becomes inactive. Clock state T3 and T4 of a fetch cycle are used to refresh dynamic memories. (The CPU uses this time to decode and execute the fetched instruction so that no other operation could be performed at this time). During T3 and T4 the lower 7 bits of the address bus contain a memory refresh address and the $\overline{\text{RFSH}}$ signal becomes active to indicate that a refresh read of all dynamic memories should be accomplished. Notice that a $\overline{\text{RD}}$ signal is not generated during refresh time to prevent data from different memory segments from being gated onto the data bus. The $\overline{\text{MREQ}}$ signal during refresh time should be used to perform a refresh read of all memory elements. The refresh signal can not be used by itself since the refresh address is only guaranteed to be stable during $\overline{\text{MREQ}}$ time.

INSTRUCTION OP CODE FETCH

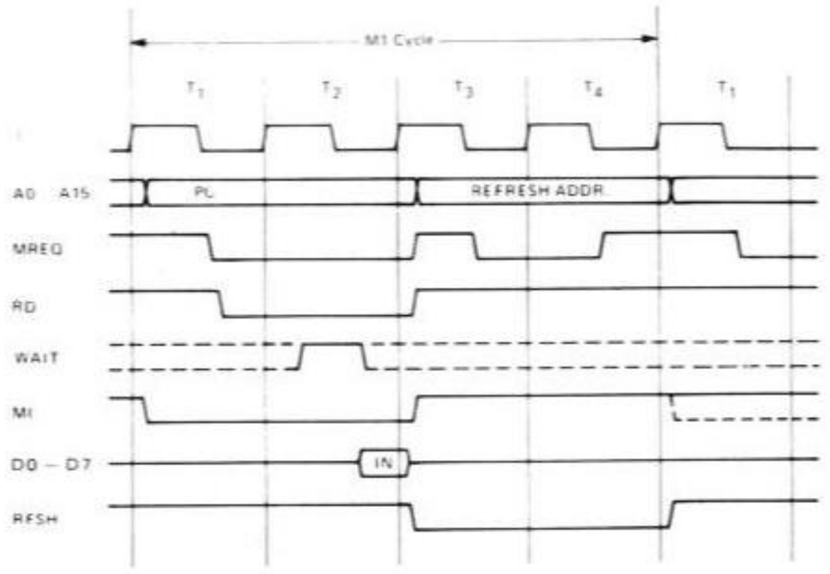


FIGURE 4.0-1

Figure 4.0-1A illustrates how the fetch cycle is delayed if the memory activates the $\overline{\text{WAIT}}$ line. During T2 and every subsequent Tw, the CPU samples the $\overline{\text{WAIT}}$ line with the falling edge of Φ. If the $\overline{\text{WAIT}}$ line is active at this time, another wait state will be entered during

INSTRUCTION OP CODE FETCH WITH WAIT STATES

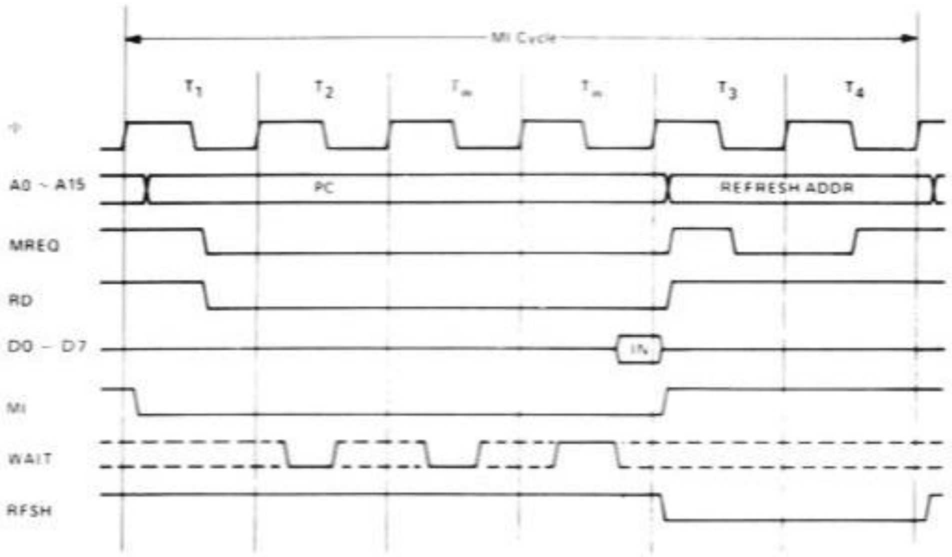


FIGURE 4.0-1A

MEMORY READ OR WRITE

Figure 4.0-2 illustrates the timing of memory read or write cycles other than an OP code fetch (M1 cycle). These cycles are generally three clock periods long unless wait states are requested by the memory via the $\overline{\text{WAIT}}$ signal. The $\overline{\text{MREQ}}$ signal and the $\overline{\text{RD}}$ signal are used the same as in the fetch cycle. In the case of a memory write cycle, the $\overline{\text{MREQ}}$ also becomes active when the address bus is stable so that it can be used directly as a chip enable for dynamic memories. The $\overline{\text{WR}}$ line is active when data on the data bus is stable so that it can be used directly as a R/W pulse to virtually any type of semiconductor memory. Furthermore the $\overline{\text{WR}}$ signal goes inactive one half T state before the address and data bus contents are changed so that the overlap requirements for virtually any type of semiconductor memory type will be met.

MEMORY READ OR WRITE CYCLES

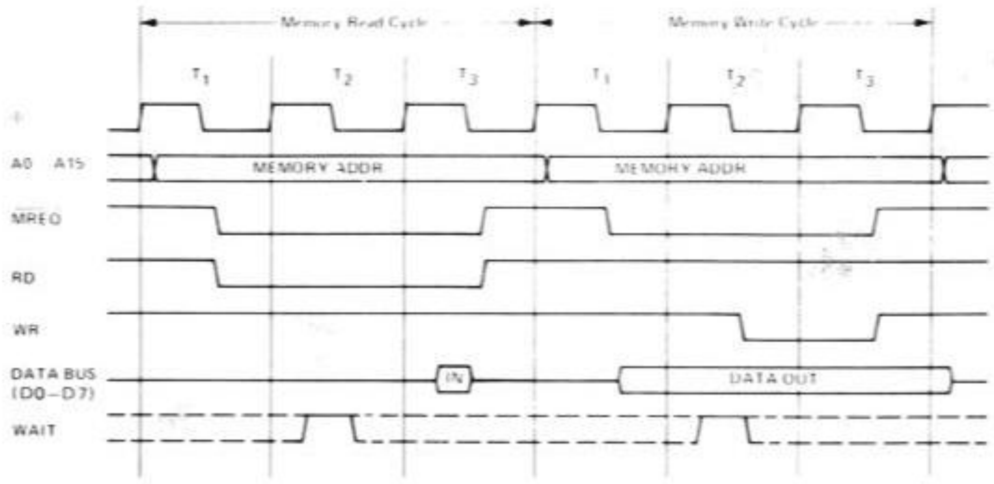


FIGURE 4.0-2

Figure 4.0-2A illustrates how a $\overline{\text{WAIT}}$ request signal will lengthen any memory read or write operation. This operation is identical to that previously described for a fetch cycle. Notice in this figure that a separate read and a separate write cycle are shown in the same figure although read and write cycles can never occur simultaneously.

MEMORY READ OR WRITE CYCLES WITH WAIT STATES

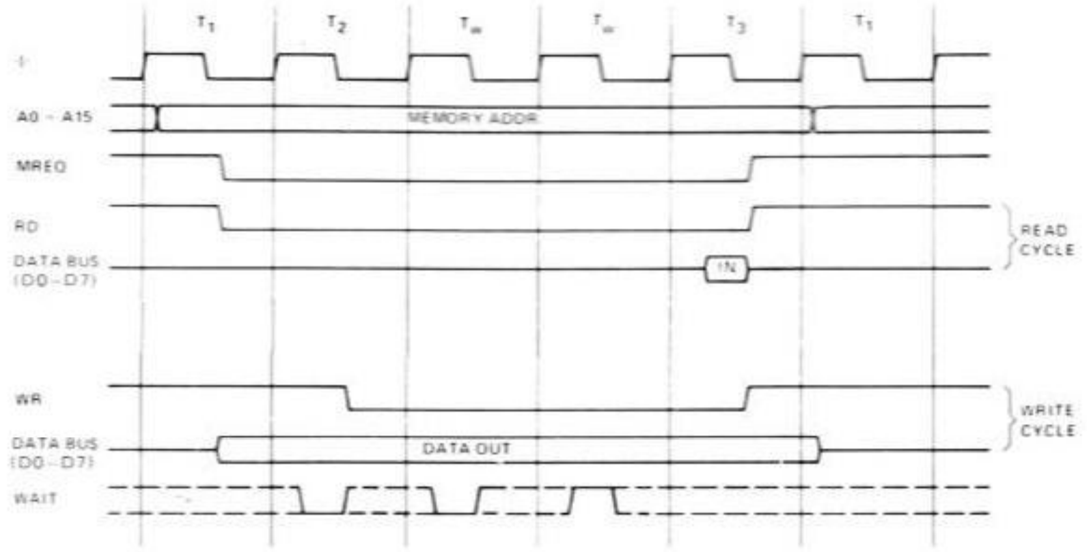


FIGURE 4.0-2A

INPUT OR OUTPUT CYCLES

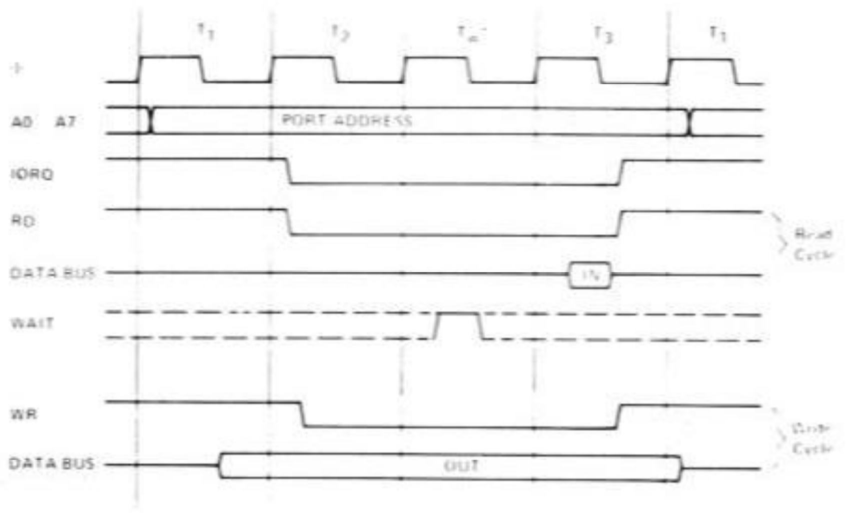
Figure 4.0-3 illustrates an I/O read or I/O write operation. Notice that during I/O operations a single wait state is automatically inserted. The reason for this is that during I/O operations, the time from when the $\overline{\text{IORQ}}$ signal goes active until the CPU must sample the $\overline{\text{WAIT}}$ line is very short and without this extra state sufficient time does not exist for an I/O port to decode its address and activate the $\overline{\text{WAIT}}$ line if a wait is required. Also, without this wait state it is difficult to design MOS I/O devices that can operate at full CPU speed. During this wait state time the $\overline{\text{WAIT}}$ request signal is sampled. During a read I/O operation, the $\overline{\text{RD}}$ line is used to enable the addressed port onto the data bus just as in the case of a memory read. For I/O write operations, the $\overline{\text{WR}}$ line is used as a clock to the I/O port, again with sufficient overlap timing automatically provided so that the rising edge may be used as a data clock.

Figure 4.0-3A illustrates how additional wait states may be added with the $\overline{\text{WAIT}}$ line. The operation is identical to that previously described.

BUS REQUEST/ACKNOWLEDGE CYCLE

Figure 4.0-4 illustrates the timing for a Bus Request/Acknowledge cycle. The $\overline{\text{BUSRQ}}$ signal is sampled by the CPU with the rising edge of the last clock period of any machine cycle. If the $\overline{\text{BUSRQ}}$ signal is active, the CPU will set its address, data and tri-state control signals to the high impedance state with the rising edge of the next clock pulse. At that time any external device can control the buses to transfer data between memory and I/O devices. (This is generally known as Direct Memory Access [DMA] using cycle stealing). The maximum time for the CPU to respond to a bus request is the length of a machine cycle and the external controller can maintain control of the bus for as many clock cycles as is desired. Note, however, that if very long DMA cycles are used, and dynamic memories are being used, the external controller must also perform the refresh function. This situation

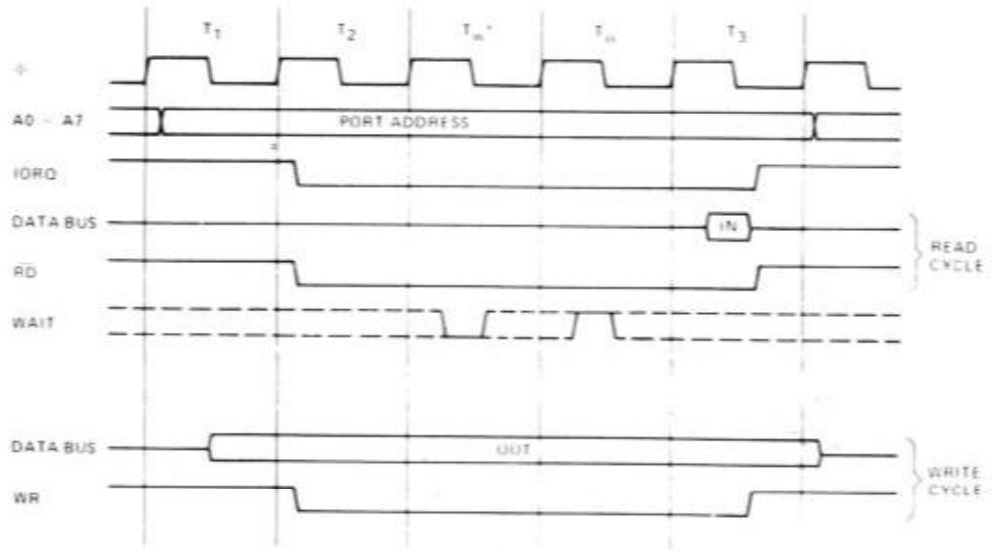
INPUT OR OUTPUT CYCLES



*Inserted by Z80 CPU

FIGURE 4.0-3

INPUT OR OUTPUT CYCLES WITH WAIT STATES



*Inserted by Z80 CPU

FIGURE 4.0-3A

BUS REQUEST/ACKNOWLEDGE CYCLE

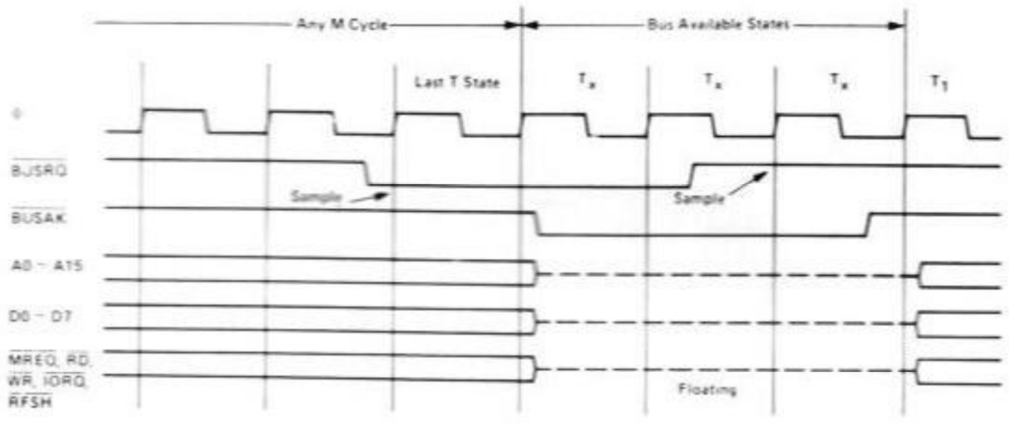


FIGURE 4.0-4

INTERRUPT REQUEST/ ACKNOWLEDGE CYCLE

Figure 4.0-5 illustrates the timing associated with an interrupt cycle. The interrupt signal ($\overline{INT}$) is sampled by the CPU with the rising edge of the last clock at the end of any instruction. The signal will not be accepted if the internal CPU software controlled interrupt enable flip-flop is not set or if the $\overline{BUSRQ}$ signal is active. When the signal is accepted a special M1 cycle is generated. During this special M1 cycle the $\overline{IORQ}$ signal becomes active (instead of the normal $\overline{MREQ}$) to indicate that the interrupting device can place an 8-bit vector on the data bus. Notice that two wait states are automatically added to this cycle. These states are added so that a ripple priority interrupt scheme can be easily implemented. The two wait states allow sufficient time for the ripple signals to stablilize and identify which I/O device must insert the response vector. Refer to section 8.0 for details on how the interrupt response vector is utilized by the CPU.

INTERRUPT REQUEST/ACKNOWLEDGE CYCLE

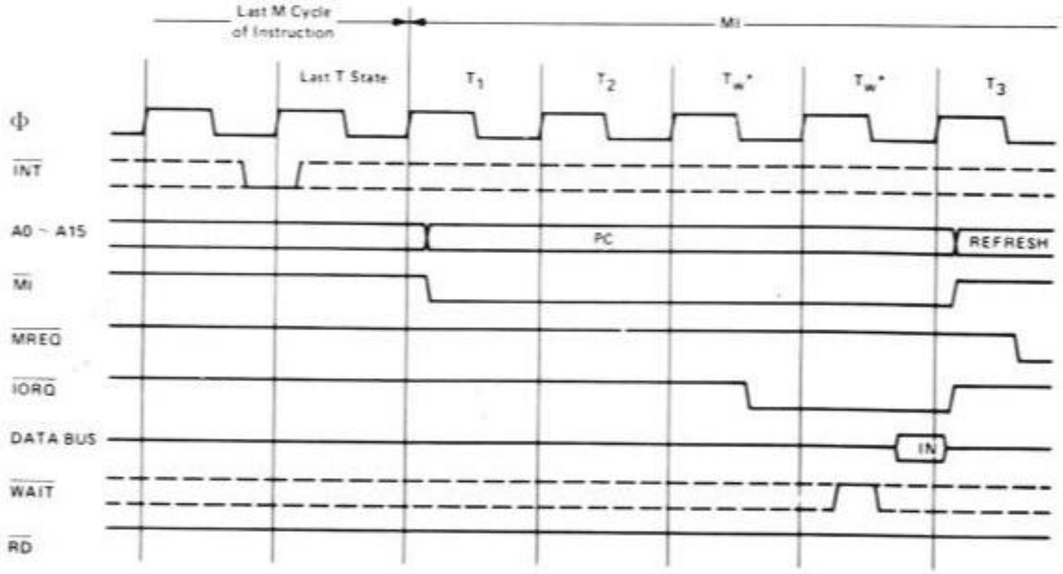


FIGURE 4.0-5

Mode 0 shown

Figure 4.0-5A illustrates how additional wait states can be added to the interrupt response cycle. Again the operation is identical to that previously described.

INTERRUPT REQUEST/ACKNOWLEDGE WITH WAIT STATES

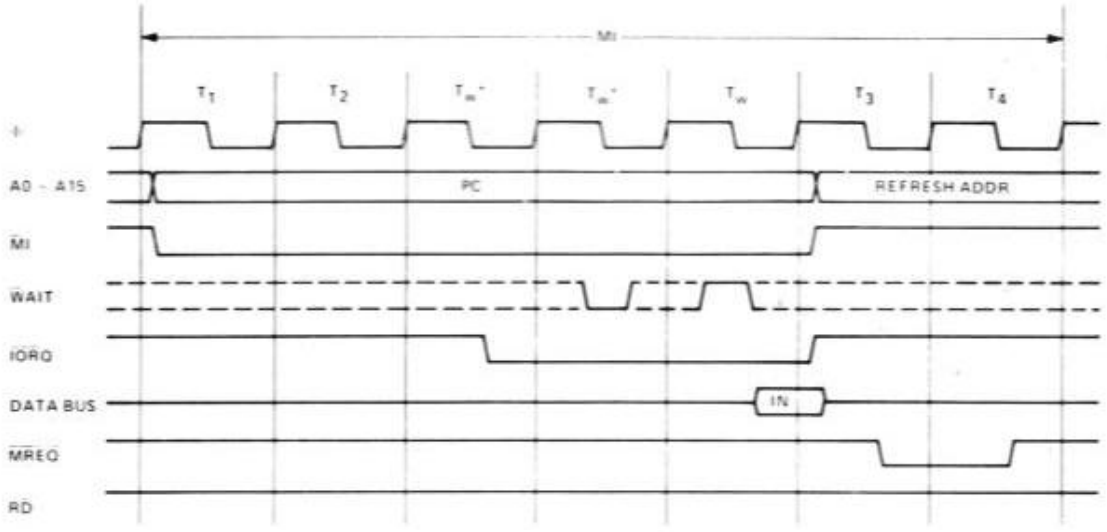


FIGURE 4.0-5A

Mode 0 shown

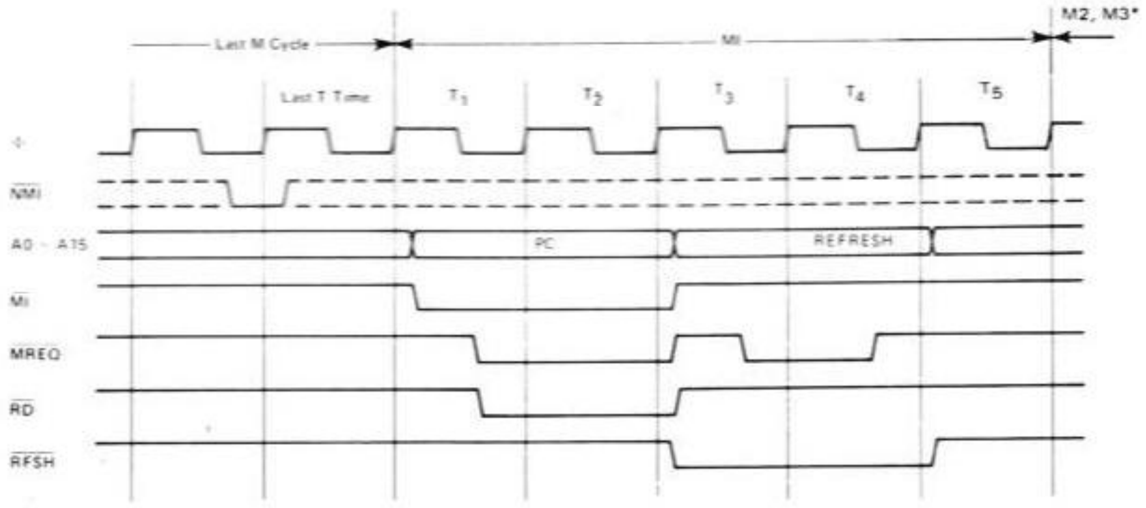
NON MASKABLE INTERRUPT RESPONSE

Figure 4.0-6 illustrates the request/acknowledge cycle for the non-maskable interrupt. A pulse on the $\overline{\text{NMI}}$ input sets an internal NMI latch which is tested by the CPU at the end of every instruction. This NMI latch is sampled at the same time as the interrupt line, but this line has priority over the normal interrupt and it can not be disabled under software control. Its usual function is to provide immediate response to important signals such as an impending power failure. The CPU response to a non maskable interrupt is similar to a normal memory read operation. The only difference being that the content of the data bus is ignored while the processor automatically stores the PC in the external stack and jumps to location 0066_H. The service routine for the non maskable interrupt must begin at this location if this interrupt is used.

HALT EXIT

Whenever a software halt instruction is executed the CPU begins executing NOP's until an interrupt is received (either a non-maskable or a maskable interrupt while the interrupt flip flop is enabled). The two interrupt lines are sampled with the rising clock edge during each T4 state as shown in Figure 4.0-7. If a non-maskable interrupt has been received or a maskable interrupt has been received and the interrupt enable flip-flop is set, then the halt state will be exited on the next rising clock edge. The following cycle will then be an interrupt acknowledge cycle corresponding to the type of interrupt that was received. If both are received at this time, then the non maskable one will be acknowledged since it was highest priority. The purpose of executing NOP instructions while in the halt state is to keep the memory refresh signals active. Each cycle in the halt state is a normal M1 (fetch) cycle except that the data received from the memory is ignored and a NOP instruction is forced internally to the CPU. The halt acknowledge signal is active during this time to indicate that the processor is in the halt state.

NON MASKABLE INTERRUPT REQUEST OPERATION



*M2 and M3 are stack write operations

FIGURE 4.0-6

HALT EXIT

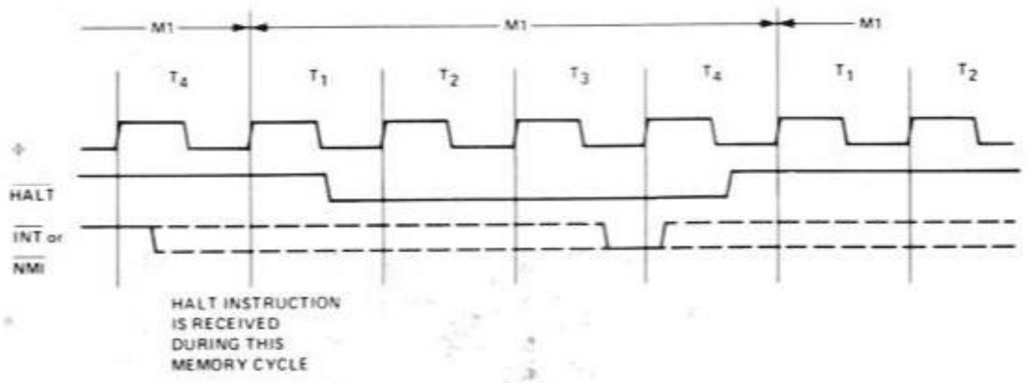


FIGURE 4.0-7

8.0 INTERRUPT RESPONSE

The prupose 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 ($\overline{NMI}$) 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 ($\overline{INT}$) 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_1 and IFF_2.

$\boxed{IFF_1}$	$\boxed{IFF_2}$
Actually disables interrupts from being accepted.	Temporary storage location for IFF_1.

The state of IFF_1 is used to actually inhibit interrupts while IFF_2 is used as a temporary storage location for IFF_1. The purpose of storing the IFF_1 will be subsequently explained.

A reset to the CPU will force both IFF_1 and IFF_2 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 instructions sets both IFF_1 and IFF_2 to the enable state. When an interrupt is accepted by the CPU, both IFF_1 and IFF_2 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_1 and IFF_2 are always equal.

The purpose of IFF_2 is to save the status of IFF_1 when a non-maskable interrupt occurs. When a non–maskable interrupt is accepted, IFF_1 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_1 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_2 is copied into the parity flag where it can be tested or stored.

A second method of restoring the status of IFF_1 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_2 are now copied back into IFF_1, so that the status of IFF_1 just prior to the acceptance of the non-maskable interrupt will be restored automatically.

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

INTERRUPT ENABLE/DISABLE FLIP FLOPS

Action	IFF_1	IFF_2	
CPU Reset	0	0	
DI	0	0	
EI	1	1	
LD A, I	•	•	$IFF_2 \rightarrow$ Parity flag
LD A, R	•	•	$IFF_2 \rightarrow$ Parity flag
Accept NMI	0	•	
RETN	IFF_2	•	$IFF_2 \rightarrow IFF_1$
Accept INT	0	0	
RETI	•	•	

"•" indicates no change

FIGURE 8.0-1

CPU RESPONSE

Non-Maskable

A non-maskable 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 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 4.0 illustrates the detailed timing for an interrupt response. After the application of RESET the CPU will automatically enter interrupt Mode 0.

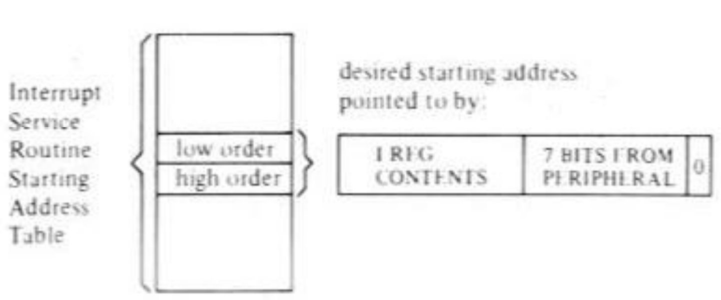
Mode 1

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

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 bit must be a zero. This is required since the pointer is used to get two adjacent bytes to from a complete 16 bit service routine starting address and the addresses must always start in even locations.


Interrupt Service Routine Starting Address Table
low order
high order
desired starting address pointed to by:
I REG CONTENTS
7 BITS FROM PERIPHERAL
0

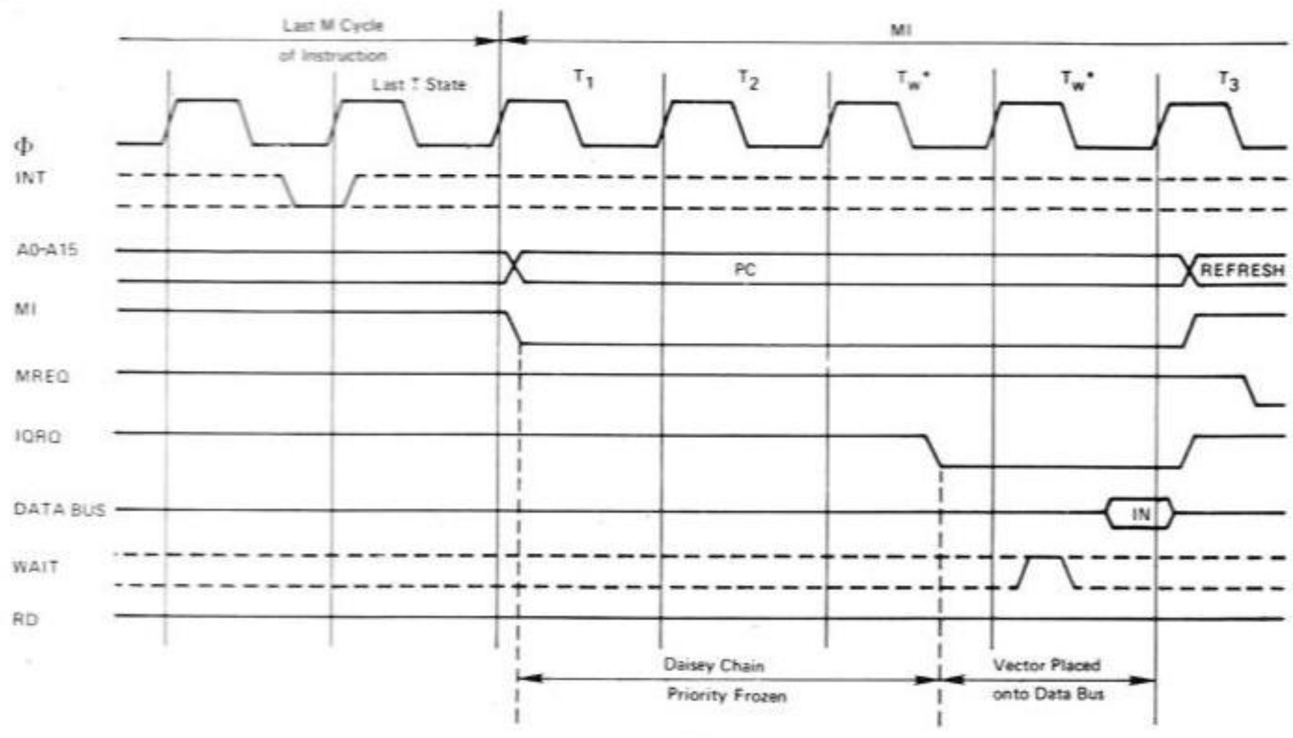
The first byte in the table is the least significant (low order) portion of the address. The programmer must obviously fill this table in with the desired addresses before any interrupts are to be accepted.

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 automat-cally 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.

INTERRUPT REQUEST/ACKNOWLEDGE CYCLE



Last M Cycle of Instruction
Last T State
MI
T1
T2
Tw*
Tw*
T3
Φ
INT
A0-A15
PC
REFRESH
MI
MREQ
IORQ
DATA BUS
IN
WAIT
RD
Daisey Chain Priority Frozen
Vector Placed onto Data Bus

Z80 INTERRUPT ACKNOWLEDGE SUMMARY

1) PERIPHERAL DEVICE REQUESTS INTERRUPT. Any device requesting and interrupt can pull the wired-or line $\overline{\text{INT}}$ low.

2) CPU ACKNOWLEDGES INTERRUPT. Priority status is frozen when $\overline{\text{M1}}$ goes low during the Interrupt Acknowledge sequence. Propagation delays down the IEI/IEO daisy chain must be settled out when $\overline{\text{IORQ}}$ goes low. If IEI is HIGH, an active Peripheral Device will place its Interrupt Vector on the Data Bus when $\overline{\text{IORQ}}$ goes low. That Peripheral then releases its hold on $\overline{\text{INT}}$ allowing interrupts from a higher priority device. Lower priority devices are inhibited from placing their Vector on the Data Bus or Interrupting because IEO is low on the active device.

3) INTERRUPT IS CLEARED. An active Peripheral device (IEI=1, IEO=0) monitors OP Code fetches for an RETI (ED 4D) instruction which tells the peripheral that its Interrupt Service Routine is over. The peripheral device then re-activates its internal Interrupt structure as well as raising its IEO line to enable lower priority devices.

INTERRELATIONSHIP OF $\overline{\text{INT}}$, $\overline{\text{NMI}}$, AND $\overline{\text{BUSRQ}}$

The following flow chart details the relationship of three control inputs to the Z80-CPU. Note the following from the flow chart.

1. $\overline{\text{INT}}$ and $\overline{\text{NMI}}$ are always acted on at the end of an instruction.
2. $\overline{\text{BUSRQ}}$ is acted on at the end of a machine cycle.
3. While the CPU is in the DMA MODE, it will not respond to active inputs on $\overline{\text{INT}}$ or $\overline{\text{NMI}}$.
4. These three inputs are acted on in the following order of priority: a) $\overline{\text{BUSRQ}}$ b) $\overline{\text{NMI}}$ c) $\overline{\text{INT}}$

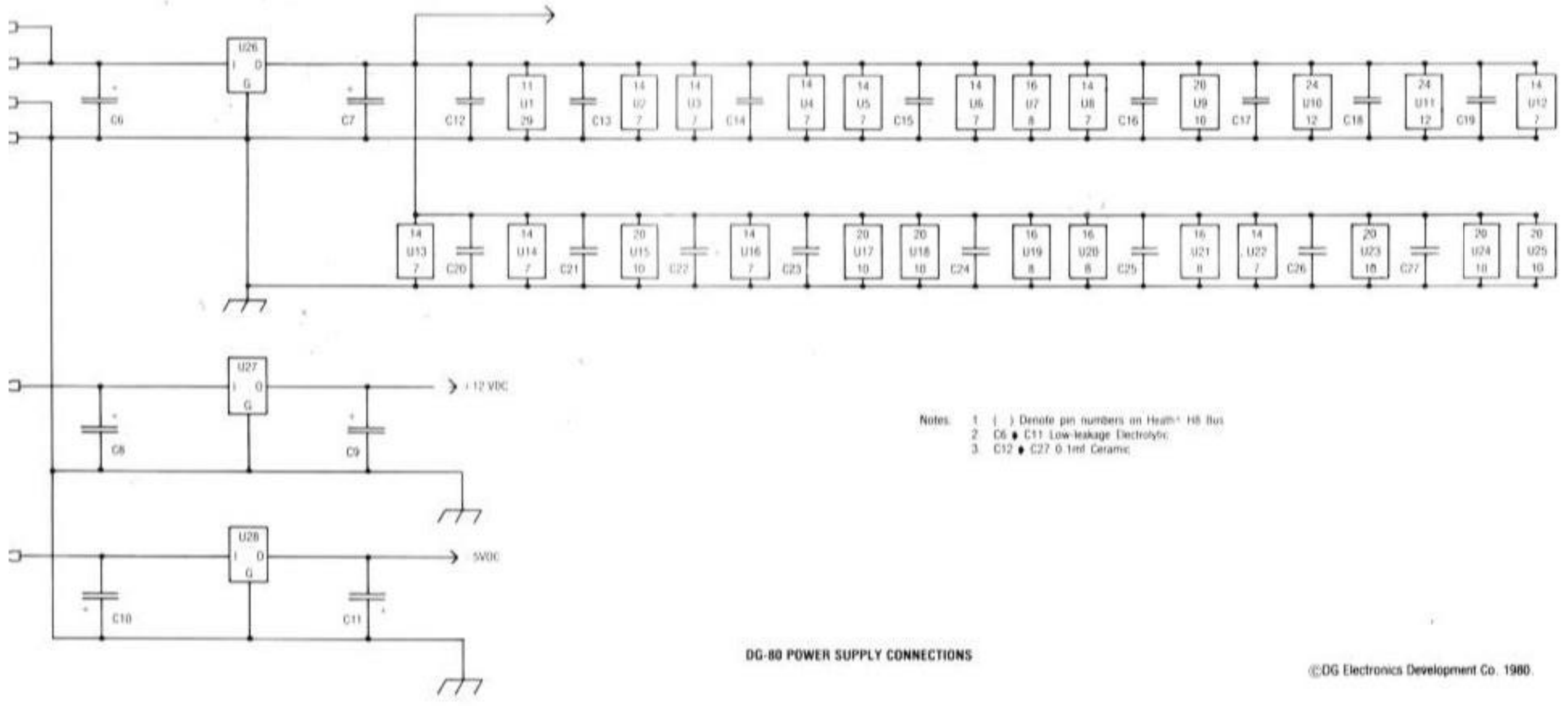
PARTS LIST AND SCHEMATIC

The Parts List and Schematic included with this manual are for the use of persons wishing to repair or modify their board after the warranty period has lapsed. **Remember, any attempt to modify or repair the board during the warranty period will void the warranty!!** The advanced user will also find the schematic useful when he attempts to implement the many advanced features of the DG-80.

PARTS LIST

U1	Z80A
U2	74S04
U3	7474
U4	74121
U5	74132
U6	7474
U7	74148
U8	74LS32
U9	74LS240
U10	User ROM
U11	User ROM
U12	7474
U13	7410
U14	7408
U15	74LS240
U16	74LS32
U17	74LS240
U18	74LS240
U19	74156
U20	74156
U21	74156
U22	74LS04
U23	74LS240
U24	74LS240
U25	74LS240
U26	7805
U27	LM341P12
U28	LM320MP5

RP-1, RP-2	1K ohm ¼W SIP
RP-3	4.7K ohm ¼W SIP
R1, R2, R5, R6, 119 R10-R19	1K ohm ¼W 5%
R3	22 ohm ¼W 5%
R4	220 ohm ¼W 5%
R7	15K ohm ¼W 5%
R8, R9	47K ohm ¼W 5%
C1, C3	0.0015mf Polyester
C2	33pf Silver Mica
C4, C5	10mf 16V Tantulum
C6, C7	10mf 35V Low-Leakage Electrolytic
C8,C9,C10,C11	2.2mf 35V Low-Leakage Electrolytic
C12 — C27	0.1mf Ceramic
D1, D2	1N4148
Y1	4.096 MHz
Q1	2N3906



Notes: 1 ( ) Denote pin numbers on Heath* H8 Bus
2 C6 ◆ C11 Low-leakage Electrolytic
3 C12 ◆ C27 0.1mf Ceramic

DG-80 POWER SUPPLY CONNECTIONS

APPENDIX A: OPERATING NOTES

Operation of the Heath H8 computer using the DG-80 CPU differs from that using the Heath 8080 CPU in the following areas:

1) **ION LED** — The H8® front panel provides an LED labelled **ION** which normally indicates whether hardware interrupts have been enabled or disabled. The designers of the Z80 microprocessor determined that in most applications this information was irrelavent and therefore did not provide a hardware signal equivalent to the 8080 "interrupts enabled" signal. For this reason, the **ION LED** on the Heath H8® front panel will remain on regardless of whether the interrupts are enabled or disabled when the DG-80 CPU is being used.

2) **MASTER RESET** — The DG-80 is designed to provide a short reset signal to the CPU in order to protect the contents of any dynamic RAM in the system. For this reason, if the Ø and RST/Ø keys are held down for a long period of time, the system will reset and then interpret these keys improperly. Therefore, when a **"Master Reset"** is desired, the Ø and RST/Ø keys should be pressed simultaneously and held only briefly. (A few tries will give you the "feel" for this operation!) If the reset is held too long, the H8® will go into the cassette tape load mode in which the front panel keyboard will appear non-functional.

3) **PAM-8 SINGLE STEP MODE** — As mentioned previously, the Z80 microprocessor does not provide a hardware "INTERRUPTS ENABLED" signal as found on the 8080A. Since the Heath PAM-8 single step function (as well as the Heath Console Debugger single step function) require this signal, the H8® computer cannot be used in this mode of operation with the DG-80 installed. When trouble shooting relatively simple programs, we suggest using the PAM-8 breakpointing feature discussed in the Heath H8® Operations Manual. The advanced user of the DG-80 will find that features of this CPU coupled with his own ingenuity will allow him to use software from sources other than Heath. This will provide him with many options in the area of debugging routines.

APPENDIX B: USING THE DG-32D DYNAMIC RAM BOARD WITH THE DG-80 CPU

The DG-32D memory board provides several jumper selectable modes of operation. Among these are two modes designed to be compatible with the DG-80 CPU.

MODE 1: This mode adjusts the DG-32D **ON-BOARD** refresh timing for use with the DG-80. For operation in Refresh Mode 1, the following jumpers should be set on the DG-32D:

J1	1♦3
J2	1♦2
J3	1♦2
J4	1♦3
J5	NO CONNECTION OR 1♦3

MODE 2: This mode allows the DG-32D to be refreshed from the Z80 Refresh Signals provided by the DG-80 and is the **RECOMMENDED** mode of operation. Since the Z80 microprocessor provides refresh signals upon execution of the HALT instruction, memory contents may be maintained indefinitely in the HALT state. Jumpers for this mode of operation should be set as follows:

J1	1♦3
J2	DOES NOT MATTER
J3	1♦4
J4	1♦2
J5	1♦3