OWNER'S MANUAL

Model 2810

Z-80 CPU

California Computer Systems

CCS MODEL 2810

Z-80 CPU MODULE

OWNER'S MANUAL

CALIFORNIA COMPUTER SYSTEMS
250 CARIBBEAN DRIVE
SUNNYVALE, CA 94086

MANUAL NO. 89000-02810

TABLE OF CONTENTS

HOW TO USE THIS MANUAL

No manual can be everything to everybody. But we have tried to design this manual so that it will be a useful reference tool for most of its users. The chapters up to "Theory of Operation" contain the information you need to configure the board to your system and to operate it with the provided firmware. "Theory of Operation" and the appendices are designed for those of you who want more information about the board, whether from curiosity or a desire to further customize it. Programming information on the Z-80 is not included in this manual; the information is simply too extensive. You will need to acquire a Z-80 programming manual.

CHAPTER 1

INTRODUCTION TO THE 2810 Z-80 CPU

California Computer Systems' 2810 Z-80 CPU provides you with a CPU, a master serial I/O port, and monitor firmware. As a result, it is the ideal foundation for an S-100 system; with the addition of RAM memory and a console device, you can have a complete system that allows considerable add-on flexibility. The 2810 Z-80 CPU is also an excellent choice for upgrading a present system. It has been carefully designed to be compatible with the major S-100 systems on the market.

The 2810 CPU and CCS's line of S-100 peripheral boards are designed to work uniquely well with each other. For example, the 2422 Multimode Floppy Disk Controller board contains ROM-resident firmware which can overlay the CPU firmware with its own, changing the monitor firmware from a paper tape-oriented firmware to a floppy-disk oriented firmware. No reprogramming of ROMs is necessary; after a minimum amount of setup, the disk controller board can be plugged in and operated with the 2810 CPU.

1.1 THE CPU

The 2810 Z-80 CPU is an S-100 bus compatible card designed for the Z-80 microprocessor. As such it combines the best of two worlds: the speed and large instruction set of the Z-80 processor with the versatility of the S-100 bus. The Z-80, a third generation processor, represents a real advance over the earlier 8080. Its large instruction set (80 more instructions than the 8080) and internal register configuration simplify the the programmer's task and reduce program size. The Z-80 is also designed to run at 4 MHz as well as 2 MHz. The 2810 CPU interfaces this powerful processor with the popular, 8080-oriented S-100 bus. This bus is used by numerous

manufacturers, allowing the user of an S-100 system a wide choice of products. To ensure compatibility with these products, the 2810 simulates as closely as possible the 8080 signals used on the S-100 bus.

Since this board will be used in a wide variety of systems and for a wide variety of applications, a number of optional features have been incorporated. These include a power-on jump for systems without front panels, address mirroring circuitry for 8080 system compatibility, and an M1 Wait State for slow memory. Moreover, bus signals for which possible bus conflicts exist are made jumper enabled.

Three diagnostic LEDs have been provided on the 2810. One indicates that the ROM is enabled and selected. The second indicates that the CPU is executing a software Halt instruction and is waiting for an interrupt. The third LED indicates that CPU has been programmed to accept interrupts. Since the CPU will remain halted while executing a Halt instruction until the system is reset or the CPU receives an interrupt, the last two LEDs can be used in combination to detect the software problem of the CPU receiving a Halt instruction before it receives an Interrupt Enable instruction.

1.2 THE ASYNCHRONOUS SERIAL I/O PORT

The 2810 Z-80 CPU contains an on-board, asynchronous serial I/O port which allows you to interface to your CPU any serial I/O device which conforms to a major subset of the RS-232-C standards for asynchronous serial communications. You have several options in using this port. If you are using the monitor firmware as is, you are provided with driver routines for the port. These routines intend that the port be used to interface the CPU to some type of console device, preferably a CRT. For flexibility, the baud rate can be set through console control. Or you can, of course, use your own driver software for the port. Appendix B contains information on programming the port's Asynchronous Communications Element. The number of stop bits, the baud rate, the type of parity, and word length are all software-selectable and the handshake lines are under software control. The port's address is jumper-selectable. Finally, you can disable the serial port with an on-board jumper.

CHAPTER 2

SETUP AND INSTALLATION

The first section of this chapter deals with configuring the 2810 to meet your system's requirements. Those of you who do not plan to use the serial port and do not have a front panel can install the board in your system after having configured the board. If you do plan to use the serial port or a front panel, section 4.2 gives additional setup and installation procedures concerning the port, while section 4.3 gives information on installing this board in a front panel system.

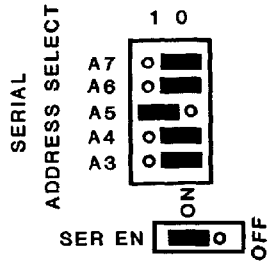
2.1 BOARD SETUP

The 2810 CPU has a number of features which are enabled or configured through on-board plug jumpers. Each of these features is discussed below, roughly in the order of the jumpers on the board, starting with the upper left corner of the board and proceeding clockwise. In addition to the plug jumpers, there is a switch to be set and an optional jumper that can be soldered in. If you are having difficulty locating or identifying any of the jumpers or the switch, the board layout in Appendix C should help.

2.1.1 Serial Port Enable and Address Select Jumpers

The SER EN jumper allows you to enable or disable the on-board serial port. If you enable the port, the SERIAL ADDRESS SELECT jumpers allow you to select the base address for the interface's registers. The address lines A0-A2 are needed to select one register out of the registers used by the serial

interface; the address lines A3-A7 are thus left to form the interface's base address. By setting the SERIAL ADDRESS SELECT jumpers A3-A7, you can select the registers' base address. The registers are addressed as seven I/O ports, either as X0h through X6h or as X8h through XEh, where X0h and X8h are the base addresses. Set A4-A7 to the binary equivalent of X. Set A3 to 0 if you wish the register addresses to begin at X0h; set it to 1 if you wish the addresses to begin at X8h.



If you are using the ROM resident I/O driver for the serial port, set SER EN to ON and jumpers A7-A3 to 00100, as shown in Figure 2-1. The I/O driver addresses the serial interface's registers at 20h through 26h; thus the base address is 20h, or binary 00100000.

FIGURE 2-1

2.1.2 Address Mirror Jumper

When addressing an I/O port, the 8080 processor duplicates or "mirrors" the port address contained in the low-order address byte in the high-order address byte. The Z-80, on the other hand, uses the low-order 8 bits only for port addressing; it puts data on the high-order address lines. Since some 8080 systems need address mirroring, we have provided address mirroring circuitry which allows the 2810 to mimic the 8080's port address method if the ADD MIR jumper is set ON. None of CCS's S-100 peripheral boards require address mirroring.

2.1.3 ROM Enable Jumper

The ROM EN jumper enables or disables the on-board ROM. If you plan to use the monitor firmware, the ROM must be enabled. If you disable the ROM, you free the memory space from F000h to F7FFh for other use.

2.1.4 M1 Wait State Select Jumper

By setting the WAIT jumper to ON, you will force the CPU into one Wait state during every M1 (op code fetch) cycle of an instruction cycle. In a Z-80, the memory access time requirements are strictest during an M1 cycle; the Memory Read and Write cycles allow an additional half a cycle to complete memory access. Thus by enabling the M1 Wait circuitry, you can use memories with access times half a clock cycle slower. In practice, this means that when the CPU is operating at 4 MHz, enabling the M1 Wait state circuitry slows the memory access requirements by approximately 110 nsecs; at 2 MHz it slows the requirements by approximately 220 nsecs. Theoretically, memories with access times slower than 400ns need a Wait state when the CPU is operating at 4 MHz. However, practice is often different than theory; you should experiment with the requirements of your system.

Most of CCS's memory boards do not need Wait states. All have provisions, however, for on-board Wait state generation, allowing Wait states to be inserted on an individual board basis. Thus you can slow down the processor for slow memory and allow it to run at full speed with fast. On-board Wait state generation can also be used for very slow memory: adding a Wait state by this method slows access times by approximately 250 nsecs at 4 MHz and 500 nsecs at 2 MHz. The disadvantage of on-board Wait state generation is that it adds a Wait state to every memory cycle in which the memory board is selected. You will have to experiment to discover which method, or combination of methods, is most efficient for your system. Note that the M1 Wait circuitry will also add a Wait state to Interrupt Acknowledge cycles, since the Z-80's M1 control signal is active at that time. The WAIT jumper set to ON enables the M1 Wait circuitry.

2.1.5 Power-on Jump Enable and Address Select Jumpers

If enabled by the JMP EN jumper, the power-on jump circuitry forces the CPU to jump to the address set by the JMP ADDR SEL jumpers when your system is turned on or reset. If the circuitry is disabled, the processor looks for its first instruction at memory location 0000h on power-on or reset. Should you enable the power-on jump circuitry, set the JMP ADDR SEL jumpers, JA15-JA0, to the binary value of the jump address you wish. Please note that JA15 is the high order bit; you should enter the binary address from the bottom up.

If you plan to use the ROM-resident firmware, you must force

a jump to the beginning address of the on-board ROM, F000h, on power-on or reset. To do so, set JA15-JA12 to 1, JA11-JA0 to 0, and JMP EN to ON.

2.1.6 2/4 MHZ Signal Enable Jumper

In the early 8080 systems, pin 98 of the bus was assigned to the status signal sSTACK, indicating that a stack read or write was in progress. Some manufacturers of S-100 systems, noting that sSTACK is little used, have converted this line to a 2 MHz/4 MHz operation indicator, where a high indicates the processor is operating at 4 MHz. We have done so also. This is a convenient feature for those of you with front panels; the sSTACK LED will tell you at a glance at which frequency the CPU is operating. It also allows peripheral devices which can monitor this line to request Wait states only when the processor is operating at 4 MHz. The newly proposed standards for the S-100 bus, however, suggest using pin 98 for an error signal input, ERROR*. To avoid possible bus conflicts, we have made the 2/4 MHZ line jumper-enabled/disabled.

2.1.7 PHANTOM Enable Jumper

The PHANTOM line is used to overlay memory at a common address. On the the 2810 Z-80 CPU, the PHANTOM line allows an external device generating the PHANTOM signal to overlay the ROM's memory space on a byte-to-byte basis. Such a device might be one of CCS's I/O boards. The ROMs on these boards can generate the PHANTOM signal, allowing portions of the CPU's firmware to be overlaid with the I/O boards' firmware. Thus driver firware for the I/O boards can be patched onto the CPU's firmware, without the CPU's ROM being reprogrammed.

Disable the signal if you do not plan to use it.

2.1.8 NMI Enable Jumper

Unlike the 8080 processor, the Z-80 processor allows two types of interrupts: a maskable interrupt (INT) and a nonmaskable interrupt (NMI). A maskable interrupt request will be accepted by the CPU depending on the state of the processor-internal Interrupt Enable flip-flop, which can be set or reset through

software commands. A nonmaskable interrupt request, on the other hand, forces the CPU to do a restart at address 0066h, regardless of the state of the Interrupt Enable flip-flop. On the 2810 board, the nonmaskable interrupt control input appears on pin 12 of the bus, as required by the proposed S-100 bus standards. However, since the 8080 processor does not provide for nonmaskable interrupts, some systems may use pin 12 for another signal. To avoid bus conflicts, we have made the NMI line jumper-enabled/disabled.

2.1.9 REFRESH Enable Jumper

The Z-80, unlike the 8080, is designed to work with dynamic as well as static RAM. At the end of every M1 (op code fetch) cycle, while the CPU is busy decoding the current instruction, the Z-80's refresh register puts out a refresh address on the address lines and the control signal REFRESH goes active. If you have in your system a dynamic RAM board, such as CCS's 65K dynamic RAM board, that can use the REFRESH signal for refresh control, you should enable this line. Consult your memory manual. Some 8080 systems may have the REFRESH line, pin 66, assigned to another signal. If this is true of yours, disable this line.

2.1.10 2/4 MHz Toggle Switch

This toggle switch, located on the top right half of the board, allows you to select the operating frequency of the Z-80. The switch positions are marked on the board. The position of this switch should be set before you turn on your system or reset it. It should not be changed during system operation.

2.1.11 MREQ jumper

Some memory boards require that the MREQ (Memory Request) control signal from the Z-80 be available on the bus at pin 65. If you have such a memory board, you can run a jumper wire from the hex pad marked 65 near the REFRESH jumper at the bottom of the board to the hex pad marked 65 near the WAIT jumper at the top of the board. Consult your memory board manuals to determine if your boards need this signal.

2.2 SERIAL I/O PORT SETUP

The following instructions apply only if you are planning to use the serial port.

2.2.1 I/O Cable Installation

CCS does not supply the cable assembly that plugs into J2, the serial port's connector. You will have to obtain one. The mating connector for J2 is a standard flat ribbon cable connector; the other end of the cable requires a DB-25S connector. If you assemble the cable yourself, be careful not to twist it; the pin 1 strip on the ribbon cable (usually the colored outside strip) should match pin 1 on both connectors. Plug the cable assembly into J2, matching pin 1s. (Pin 1 for J2 is labeled on the board). Push the cable connector down firmly until you can no longer see the metal pins. The DB-25S connector should be fastened to one of the slots in the back of your mainframe. Plug the DB-25P connector on your peripheral's signal cable into it.

2.2.2 Peripheral Configuration

If you plan to use the I/O driver and initialization firmware provided, your peripheral should be set to expect a serial data format of 8 data bits, no parity bit, a 0 stick bit and one stop bit per word. Set your peripheral for the baud rate at which you wish to operate; the firmware will initialize the port to any standard baud rate. Consult your peripheral manual for setup instructions.

If you are not using the initialization firmware provided, you will have to configure your peripheral to match your software.

2.3 FRONT PANEL SETUP

If you will be using the 2810 in a front panel system, you must connect the data cable from the front panel to the front panel data socket, J3. Specific instructions for the Altair and Imsai microcomputers follow.

2.3.1 ALTAIR 8800

You must replace the molex connector on the front panel cable with a DIP plug that you supply yourself. Be careful when soldering the connections: Unlike the data lines on J3, the data lines on the Altair molex connector are not arranged sequentially.

2.3.2 IMSAI

Plug the data cable connector directly into J3, matching pin 1's. Pin 1 is labeled on the board for J3. Pin 1 on the cable connector is identified by a mark or tick on the underside; it does not necessarily correspond with any numbering on top.

CHAPTER 3

THE MOSS 2.2 MONITOR

CCS's MOSS 2.2 Monitor contains powerful routines for program debugging and for controlling from a console keyboard a system using the 2810 Z-80 CPU. It allows you to display a block of memory in hex and ASCII, to move, change, and verify memory, and to transfer control to another program in memory with breakpoints set. You can also output or input a data byte to or from any I/O port and command the monitor to read, write, and format paper tape.

Note that for the MOSS Monitor to work exactly as described below, the on-board ROM, serial I/O port, and power-on jump circuitry must be enabled, with the serial port's base address set to 20h and the jump address set to F000h.

3.1 THE MONITOR'S MEMORY SPACE

The monitor is resident in the on-board ROM, the starting address of which is F000h. In addition, it needs some RAM space for the system stack and temporary storage area. The monitor scans the available memory until it finds the highest active RAM address and then counts down 56 bytes to store the breakpoints, registers, and register restore routine. It locates the system stack below that: you should reserve at least 88 bytes of high RAM memory for the monitor's use. The monitor also requires some low RAM as well: you should reserve locations 0000h-0003h and, if you use breakpoints, locations 0008h-000Ah.

3.2 SOFTWARE ENTRY POINTS

A cold-start entry at F000h sets up the system stack and work area, initializes the serial port and register storage area, selects the on-board serial port as the console interface, and loads memory locations 0000h-0003h with a jump instruction to the warm-start routine. It also loads the following locations, called by the Z-80 restart commands, with jump vectors to a restart error message: 0008h-000Ah, 0010h-0012h, 0018h-001Ah, 0020h-0022h, 0028h-002Ah, 0030h-0032h, and 0038h-003Ah. These locations can be overwritten with restart routines.

A warm-start entry at F10Fh resets the stack pointer and the warm start jump vector located at 0000-0002h. All other conditions remain unaffected.

The breakpoint entry at F024 saves all register contents; all other conditions remain unaffected.

3.3 THE BASIC I/O ROUTINES AND THE IOBYTE

You can call the monitor's basic I/O subroutines from your own programs. The jump vectors are as follows:

Routine name	Address	Description
CONIN	F003	Console input
CONOUT	F009	Console output
CONST	F012	Console status
READER	F006	Paper tape reader input
PUNCH	F00C	Paper tape punch output
LIST	F00F	List device output

These routines perform the IOBYTE handling to support the IOBYTE function, as developed in the Intel MDS system and as used by CP/M. The IOBYTE function allows you to assign a physical device to one or more of four logical peripheral device categories: Console, Punch, Reader, and List. The current physical to logical device assignment is stored in the IOBYTE in location 0003h. When an I/O routine, such as CONIN, is called, it examines the contents of IOBYTE and jumps to the peripheral driver routine indicated by the physical device assignment. The contents of the IOBYTE, and hence the physical device assignments, can be changed through the Assign command.

The monitor firmware contains driver routines to support

only the teletype physical assignment in all four logical categories. (Please note that the physical assignment names do not have to accurately describe the actual peripheral used. The teletype assignment, for example, could be used to implement console operations with a CRT.) All other physical assignments cause a jump to the I/O Assignment Error message when one of the above routines is called. For more information, see the Assign command, 3.7.1.

With the exception of CONIN, the above basic I/O routines are CP/M compatible when used with the default teletype assignment. They conform to the CP/M calling conventions, passing the data in the C register for any output and in the A register for any input. For a CP/M compatible console input routine, use entry point F68Fh. This routine, CONI, strips the ASCII parity bit as CP/M convention requires.

3.4 BRINGING UP THE MONITOR

To enter the monitor, turn your system on or reset it. This results automatically in a cold-start entry into the monitor. Set your terminal to the baud rate at which you wish to operate. You have a choice of any baud rate between 2 and 56K baud. Hit the carriage return key until the monitor responds with

```
MOSS VERS 2.2
```

The maximum number of carriage returns needed before the monitor responds is three. When the monitor prompt appears, you may start entering commands.

3.5 MONITOR COMMANDS

The MOSS Monitor commands must conform to a specific format. The general form is

```
-CE1 E2 E3
```

where C is the command character and E1-E3 are the address and data entries, if any. The essential parts of a command are as follows:

The Command Character: The monitor is controlled by one-character commands entered from the keyboard in response to the monitor prompt, a dash (-). No space is allowed between the prompt and the command character.

Address and data entries: The general form for an address is a four digit hex number; for data, a two digit hex number. Leading zeros need not be entered; the monitor will supply them. No space is allowed between the command character and the first address or data entry. Subsequent entries must be separated by a delimiter. The monitor looks at only the last four address characters or last two data characters before a delimiter. So if you make a mistake while typing an entry, keep typing until the last two or four characters are correct.

Delimiters: The MOSS Monitor recognizes three delimiters: a carriage return (CR), a space, or a comma. A carriage return indicates to the monitor that the current command is complete and should be executed. Either a space or a comma can mark the end of an address or data entry. In our command examples we will generally use a space as a delimiter, unless a comma makes the command form clearer. Please note, however, that you can use the space and the comma interchangeably. In certain commands a space or a comma can also be interchanged with a carriage return. These are commands for which the Monitor expects a fixed number of entries (and hence delimiters) following the command character.

Sample Command

The following commands to display the block of memory 0FFBh to 100Ah are all equivalent. Although the spacing is not form free, some variety in the command form is allowed. Note that the display command requires two and only two address parameters, so that the last delimiter can be a comma or a space as well as a carriage return.

```
-D0FFB 100A[CR]
-DFFB,100A,
-DFFB,100A[CR]
-DFFB 100A[space]
-D0EF0FFB,100A[space]
```

3.6 ERROR MESSAGES

The MOSS monitor detects three types of error conditions and responds witn a different error message for each. They are as follows:

Command Error: Should you make an invalid entry, the command will be aborted, a warm boot of the system will occur, and the error message

????

will be printed, followed by the monitor prompt.

I/O Assignment Error: As described in section 3.3, the Assign command allows you to assign a physical device to a logical peripheral category. When an I/O routine involving the logical category is called, the CPU will jump to the driver routine indicated by the physical assignment. If there is no driver routine, it will jump instead to the I/O Assignment Error routine. This routine sets the IOBYTE to its default value, outputs the error message

I/O ERR

and does a warm boot of the system. If you are using the monitor's basic I/O routines with CP/M, an I/O assignment error will cause the error message to be printed and control returned to CP/M. See the Assign command for more detail.

Restart Error: During cold-start initialization, jump-vectors to a restart error message are loaded in the memory locations called by the Z-80 restart instructions. This is done to prevent a program jump to a restart address without code. A restart error causes a warm boot of the system and the following message to be printed:

RST ERR

The message is followed by the monitor prompt. If you are running CP/M with the monitor enabled, a restart error will cause the error message to be printed and control returned to CP/M.

3.7 COMMAND DESCRIPTION

3.7.1 Assign (A)

The Assign command allows you to change the physical-to-logical device assignments and thus choose the peripherals you wish to work with while in the monitor. The IOBYTE function as developed by Intel for the MDS systems divides peripherals into four logical categories: Console, typically a teletype or a CRT; Reader, a paper tape reading device; Punch, a paper tape punching device; and List, a hard-copy printing device. Each of the four logical categories may have one of four physical devices assigned to them. The possible physical-to-logical assignments are as follows:

```
(C) Console Logical Device
    (T) Teletype
    (C) CRT
    (B) Batch Mode (input from logical reader device;
                    output to logical list device)
    (1) User Console #1

(R) Reader Logical Device
    (T) Teletype
    (P) Paper tape reader
    (1) User reader #1
    (2) User reader #2

(P) Punch Logical Device
    (T) Teletype
    (P) High speed paper tape punch
    (1) User punch #1
    (2) User punch #2

(L) List Logical Device
    (T) Teletype
    (L) High speed line printer (CRT in CP/M)
    (1) User list #1 (High speed line printer in CP/M)
    (2) User list #2 (User List #1 in CP/M)
```

To assign a peripheral to a logical device category, enter

```
-AX
```

where X equals either C,R,P, or L, the logical device codes. If you enter a character other than these four, the computer will return with ???? and another prompt. If you enter a valid

logical device code, the computer will return immediately with a prompt for the physical device code. Enter

-Y

where Y equals the physical device code. Should you enter a delimiter only or a nonvalid device code, the device assignment will remain unchanged.

EXAMPLE:

Entering

-AR-P

assigns a high speed paper tape reader to the Reader logical device category.

Assigning a physical device to a logical category alters the contents of the IOBYTE, stored in location 0003h. Every time an input or output routine involving a specific logical device is performed, the I/O routine examines the contents of the IOBYTE to determine the physical device assignment and jumps to the driver routine called by the physical assignment. If there is no driver routine, the I/O routine jumps to I/O assignment error routine, resulting in the I/O Assignment Error message being output and physical assignments being set to their default value, the teletype.

For all the basic I/O routines, the teletype assignment forces a jump to the on-board serial port drivers. The serial port is designed to be the console interface; it is best used for a CRT, although any console device can be used. Please note the port drivers cannot drive the paper tape reader or punch of a teletype. If you have not altered the firmware in any way, calling the Reader or Punch I/O routines results in the CPU reading from or writing to the console device when the teletype assignment is used.

None of the other physical device assignments are supported by driver routines. You can patch driver routines for different devices onto the monitor firmware by two techniques. One is to have the routines residing in a ROM device capable of generating the PHANTOM signal (section 2.1.8), so that the jump instruction to the I/O error message for a particular physical device assignment is overlaid with a jump instruction to the driver routine. CCS's S-100 peripheral boards can work in this manner; each generates the PHANTOM signal when its on-board ROM is selected. If you choose to use this method, you have the choice of programming the ROM yourself or using a CCS preprogrammed ROM.

The second technique is to change the jump instruction in the ROM itself. For example, if you wished to connect a line printer to your system, you would change the jump instructions at locations F61D and F676 so that they contained the starting addresses of your driver routines and not the address of the I/O error message. This, of course, means erasing and reprogramming the ROM.

3.7.2 Display (D)

This command allows you to display the contents of a specified block of memory. The general form for the command is

```
-DA1 A2
```

where A1 and A2 are the first and last bytes, respectively, of the memory block.

The resulting display divides the memory into 16 bytes per line. Each line starts with the address of the first byte in the line, followed by the data in hex and their ASCII equivalents. The contents of locations having the same last hex digit in their address are aligned vertically. Periods represent data for which there are no ASCII equivalents. As the output fills the screen, it will automatically scroll up. To freeze the display, type a control-S. To start it again, hit any key on the keyboard. Should you wish to escape from the display mode, hitting any key on the keyboard will abort the command and cause the monitor prompt to appear.

EXAMPLE

Entering

```
DF450 F4BF
```

results in the following display:

```
F453             E1 08 D9 D1 C1 F1 E1 F9 00 21 00 00 C3     a.YQAqay.!..C
F460  00 00 AF 32 03 00 21 6C F4 C3 B5 F6 49 2F 4F 20  ../2..!ltC5vI/O
F470  45 52 D2 CD E8 F6 B0 47 82 57 78 C9 0E 0D CD 7C  ERRMhv0G.WxI..M|
F480  F6 0E 0A C3 7C F6 CD 56 F6 E6 7F C9 3F 3F 3F BF  v..C|vMVvf.I????
F490  4D 4F 53 53 20 56 45 52 53 20 32 2E 32 0D 8A 3E  MOSS VERS 2.2..>
F4A0  0F D3 24 11 40 00 62 6A DB 26 A3 28 FB DB 26 23  .S$.@.bj[&#([[&#
F4B0  A3 A3 C2 AD F4 E5 29 5C 19 19 E5 29 29 DB 20 2B  ##B-te)\..e))[ +
F4C0  7D B4 C2 BD F4 E1 3E 83 D3                       }4B=ta>.S
-
```

3.7.3 End Of File (E)

The E command informs the computer to type punch an Intel format End Of File record at the end of a just-punched paper tape file. The Intel EOF format contains both the entry address for the file and six inches null leader. The E command allows you to specify the entry address and change the length of the leader, if you wish. The general form for the command is

```
-EA L
```

where A is the entry address and L is the length of null leader in tenths of inches expressed in hex. For example, for a four inch leader, enter hex 28 (4"=40 tenths=28h). The default value for the length is six inches; for the address, 0000h. An entry address of 0000h will return control to the monitor after the paper tape has been read.

The Monitor expects two parameters for the E command. A carriage return after the E or first parameter will result in the error message ????. If you wish to set the length and entry address to their default values, simply enter a space or a comma twice.

If you have assigned to the logical punch category a physical punch device for which there is no driver code, using the E command will result in the error message

```
I/O ERR
```

and the return of the monitor prompt. The exception for this is the teletype default assignment. The firmware is designed to output the EOF record to the console device.

3.7.4 Fill (F)

The fill command allows you to fill a block of memory with a specified constant. The general command form is

```
-FA1 A2 C
```

where A1 and A2 are the addresses of the first and last bytes of the memory block and C is the constant in hexidecimal.

EXAMPLE

Entering

```
-F10AA 10BB 1
```

fills the memory block 10AAh to 10BBh with the constant 1.

3.7.5 Goto (G)

The G command allows you to transfer control from the monitor to another program. It allows you to specify the entry address and to set up to two breakpoints for returning control to the monitor. When the monitor encounters a breakpoint, it saves the contents of the Z-80 registers in the system's temporary storage and outputs to the console device an asterisk followed by the next address in the program. It then returns the prompt. You can use the Examine Register command (X) at this time to examine or change the saved registers.

The general form for the G command is

```
-GA B1 B2
```

where A is the entry address, and B1 and B2 are the addresses of the breakpoints. There are many allowed variations on this command, however, which makes it a powerful and convenient command. You have the option of establishing 0, 1, or 2 breakpoints: simply enter a [CR] when you have established the number of breakpoints you wish. If you enter the maximum, two, a delimiter (comma or space) is all that is necessary to begin command execution.

You may also begin execution of the program at the PC address saved in the register storage area. Thus you can return control to the address where the program stopped when it encountered a breakpoint, or to the address you have loaded in the saved PC register through the Examine Register command. Note that since all breakpoints are cleared when any breakpoint is encountered, you must specify any desired breakpoints in the command if you use it this way. The form of the command for transferring program control to the address in the PC register is

```
-G[CR]   (no breakpoints)
          or
-G,B1,B2   (breakpoints set)
```

There are two more points regarding breakpoints that ought

to be mentioned. Because breakpoints are generated by the monitor inserting a RST 8 instruction (CF) into the program at the breakpoint location, breakpoints can be set only in programs residing in RAM. Further, a breakpoint must be inserted at an op code location. If it is inserted in an operand or data field, it will not be executed.

3.7.6 Hex Number Addition (H)

This command provides an easy way to add or subtract hex addresses. Entering

-HA1 A2

where A1 and A2 are the hex addresses results in the output

AS AD

where AS=A1+A2 and AD=A1-A2. Note that if the sum is greater than FFFF, the carried one is lost. If A2 is greater than A1, A2 will be subtracted from A1 + 10000h.

3.7.7 Input (I)

This general purpose input command allows you to read a data byte from any input port. To do so, enter

-IA

where A is the port address in hex. The monitor will respond by printing the data byte in binary.

3.7.8 Leader (L)

The L command allows you to output hex-number nulls for a paper tape leader. As with the E command, you may specify length of the leader in tenths of inches in hex, the default value being six inches. The form for the L command is

-LH

where H is the length in tenths of inches expressed in hex.

If the current physical-to-logical assignment for the Punch category is the teletype, the null leader will be output to the console device unless punch driver routines have been provided for the teletype assignment.

3.7.9 Move (M)

The M command moves a block of data to a specified address. The general form for the command is

-MA1 A2 AD

where A1 and A2 are the addresses of the first and last bytes of the memory block and AD is the destination address.

When using this command, be careful not to locate the destination address within the source block. Since the block is moved byte by byte, starting with the byte with the lowest address, the data being transferred will write over the original contents of the section of the source block that follows the destination address.

3.7.10 Output (O)

This general purpose output command allows you to output a data byte to any output port. Enter

-OA D

where A is the port address and D is the data in hex.

If you have CCS memory boards in your system, you can use this command to select a memory bank by outputting a Bank Select Byte to the Bank Select Port. (See your memory board manual.)

3.7.11 Query (Q)

The Q command displays the current physical-to-logical device assignments. Entering the command

-Q[CR]

results in the current assignments being displayed in the format

C-X R-X P-X L-X

where X equals the physical device code.

3.7.12 Read (R)

The read command allows you to read from an Intel format paper tape in the currently assigned paper tape reader and to add a bias to the starting address in the paper tape header. The general form for the read command is

-RB

where B is the address bias in hex.

The monitor checks for errors while reading the paper tape. If it encounters one, the program is aborted. The read routine also provides error checking of the program loaded in memory; if an error is found, the address of the byte in error is displayed, along with an 8-bit binary representation of the bit error, in which a 1 indicates a bit in error. For example, the display

F038 00010000

would indicate that bit 4 of the byte in memory location F038 is in error.

After the paper tape has been read, control will be returned to the monitor if the entry address in the EOF record is zero. If it is a non-zero number, control is transferred to that address.

If the current physical device assigned to the Reader logical category is the teletype, the monitor will respond to the Read commmand by reading a a program in binary typed by hand from the console unless you provide paper tape reader rountines for the teletype assignment.

3.7.13 Substitute (S)

The substitute command allows you to examine the contents of a specific memory location and alter them if you desire. Begin the S command by entering

```
-SA,
```

where A is the address of the memory location you wish to examine. The computer will immediately respond with the data contents followed by a prompt:

```
-SA,D-
```

If you wish to leave the data unaltered, simply enter a delimiter. If the delimiter is a space or a comma, the computer will respond with the contents of the next consecutive memory location and another prompt. If it is a carriage return, the command is terminated and control is returned to the monitor. Should you wish to alter the data, enter the desired data followed by a delimiter: a carriage return if you want to terminate the command or a space or a comma if you wish to review the next memory location. You can continue examining and altering memory byte by byte in this way as long as you wish. To make it easier for you to keep track of where you are, on every 8-byte boundary (that is, an address ending with either 0 or 8, the monitor will do a line feed and print the address along with the data.

3.7.14 Test (T)

The test command provides a quick way to test RAM memory for hard data bit failures without destroying the contents of the RAM. To test a block of memory for bit failures, enter

```
-TA1 A2
```

where A1 and A2 are the addresses of the first and last bytes in the block, respectively. The monitor will respond by printing the address of any byte in error, followed by an 8-bit representation of the bits in error. (See the Read command for further details). If you wish to freeze the display type a Control-S. To start it again, hit any key. Hitting any key while the command is executing returns you to the monitor.

3.7.15 Verify (V)

You can use the V command to compare two blocks of memory and verify that they are the same. Type

```
-VA1 A2 AD
```

where A1 and A2 are the addresses of the first and last byte in the source block and AD is the starting address of the block to be verified. Should the two blocks match, the monitor will return with the prompt. Should two correspondng bytes differ, the monitor will display the source address and its contents in hex, followed by a dash and the contents of the corresponding address of the block being verified. During the execution of the command, the display can be frozen or control returned to the monitor as described in previous section.

3.7.16 Write (W)

Use the W command to punch a memory block on paper tape. Enter

-WA1 A2 R

where A1 and A2 are the addresses of the first and last byte of the block and R is the record length. The Intel paper tape format specifies a record length of 16 data bytes. You can change that length to any number of bytes from 1 to 255. Enter the length you want in hex. The default value is 16 data bytes. Note the monitor expects three delimiters with this command.

If you want a null leader to begin your file, you must use the L command before the W command. If you want to end your file with an EOF record or null leader, use the E or L command after the file has been punched.

Again, the monitor will output the memory block to the console device if the logical punch category is at its default value and no driver routine has been provided for the teletype punch assignment.

3.7.17 Examine (X)

The X command is a very useful command when used in conjunction with the G command's breakpoint facilities. Entering

-X[CR, space or comma]

causes the Z-80 registers currently stored in the system stack area to be displayed for examination. These registers are the main and alternate accumulator and general purpose registers, the

Interrupt register (I), the Program Counter register (P), the Stack Pointer register (S), the two Index Registers (X and Y) and the Refresh register (R). In addition, the contents of the memory locations addressed by the main and alternate H and L registers are also displayed (M and M'). The registers are displayed in the following four-row format

```
A-xx B-xx C-xx D-xx E-xx F-xx H-xx L-xx
M-xx P-xxxx S-xxxx I-xx
A'-xx B'-xx C'-xx D'-xx E'-xx F'-xx H'-xx L'-xx
M'-xx X-xxxx Y-xxxx R-xx
```

where xx equals a two digit hex byte and xxxx equals a four digit hex address.

To examine or alter the contents of one register, enter

```
-Xr[CR, space or comma]
        or
-X'r[CR, space or comma]
```

where r is a main register and 'r is an alternate register. (Note that if you wish to examine the X, Y, or R registers, you must preface register character with the prime mark.) The monitor will return with the contents of the register and a prompt:

```
-Xr,Dh-
```

As in the substitute memory command, you have the option of altering the memory (entering desired contents followed by a delimiter) or leaving the contents unchanged (entering a delimiter). A carriage return terminates the command; a space or a comma causes the contents of the next register to be displayed. Note that altering the contents of the H and L registers changes the address; if you wish to alter the contents of the memory location, alter the M register. (See section B.1 for a discussion of the Z-80 registers.)

3.7.18 Initialize Baud Rate (Y)

To change the baud rate of your system without a system reset, use the Y command. Enter

```
-Y (no delimiter)
```

and then set the baud rate of your terminal to the desired rate. Hit the carriage return key until the monitor returns with the

prompt. The monitor will accept any baud rate between 2 and 56K baud.

3.7.19 Zleep (Z)

The Z command is used to prevent unauthorized use of your system. Entering

-Z[CR, space or comma]

locks up the system so it will not respond to anything other than the ASCII bell character (control G). Entering two consecutive bell characters will unlock the system, returning control to the monitor without altering anything.

CHAPTER 4

THEORY OF OPERATION

This chapter is divided into two main sections: the CPU and the Serial Port. In both sections, active low signals are indicated by an asterisk (*) following the signal name. Definitions of the signals used by the CPU bus and the serial interface can be found in Appendix A.

4.1 THE CPU

This section describes the 2810's support circuitry for the Z-80. Where it is pertinent, we discuss the Z-80's operation. However, a complete description of the Z-80 is beyond the scope of this manual. Should you wish to know more about it, we suggest you consult a Z-80 technical manual.

Since the S-100 is an 8080-oriented bus, much of the circuitry in the 2810 Z-80 CPU is devoted to interfacing the Z-80 to the S-100 bus. Because of this, and because this board will be used in 8080-based systems, the following discussion of the 2810's operation will often deal with the differences between the 8080 and the Z-80.

4.1.1 The Reset Logic

The gates generating POC*, pRESET*, and EXT CLR* are connected in series, so that when POC* goes low, pRESET is pulled low, which in turn pulls EXT CLR* low. POC* goes low approximately 50 msecs after power-on. The delay is provided by a one-shot which emits a positive-going pulse 50 msecs after

power-on. This pulse is inverted and pulls POC* low. Both pRESET* and EXT CLR* can also be pulled low by external switches.

4.1.2 The External Clock Circuitry

The early 8080 microprocessor required a 2 MHz, two-phase, nonoverlapping clock. Thus, by convention, there are three clocks on the S-100 bus: CLOCK, which is a 2 MHz signal; phase one, Φ1; and phase two, Φ2. The Z-80, on the other hand, can operate at either 2 or 4 MHz and requires only a one-phase clock. Thus the functions of the Φ1, Φ2, and CLOCK signals on the 2810 differ from those on an 8080 CPU. On the 2810, Φ1 and Φ2 can be either 2 MHz or 4 MHz signals. Once inverted, Φ2 is the processor's clock, pCLK, while Φ1 is available on the bus simply for those devices that need it. CLOCK remains a 2 MHz signal, regardless of processor speed, for those devices that need a clock of a constant frequency.

The clocks on the 2810 are derived from the on-board 16 MHz crystal oscillator. The 16 MHz signal is divided by 2, 4, and 8 by a synchronous 4-bit counter, U24. Thus the outputs of this counter are in-phase 8 MHz, 4 MHz, and 2 MHz signals. These signals are multiplexed by U22, a 4-to-2 line multiplexer. The select line for the multiplexer is controlled by the 2/4 MHz toggle switch. When the switch selects 2 MHz, the multiplexer's outputs are the 2 and 4 MHz signals. The 2 MHz signal is the Φ2 clock and is inverted and buffered to become pCLK. The 4 MHz signal is inverted and ANDed with the 2 MHz signal, creating the non-overlapping Φ1 clock (see figure 4-1). When 4 MHz operation is selected, the multiplexer's outputs are the 4 MHz and an 8 MHz signals, which, through the process described above, become the 4 MHz Φ1, Φ2, and pCLK signals.

4.1.3 The Address Bus and Address Mirroring

The Z-80's low-order address lines are buffered by a three state bus driver, the outputs of which are bus address lines A0-A7. They are also multiplexed with the Z-80's high-order address lines by U28 and U29, the outputs of which are the bus address lines A8-A15. The select line to the multiplexers is controlled by the address mirroring circuitry. When it is enabled through the address mirror jumper, it will pull the select line high, allowing the low-order address bits onto the high-order address bus whenever the I/O request signal from the Z-80 (IOREQ*) is active while the M1 signal (M1*) is inactive.

(An Interrupt Acknowledge cycle is distinquished by both signals being active.) In any other case, or if the address mirror circuitry is disabled, the select line to the multiplexer will be low, allowing only the high-order address bits onto the high-order address bus.

The signal ADD DSB*, when active during DMA operations, places the address bus driver and multiplexers in their high impedance state, allowing an external device to control the address bus without interference from the CPU.

4.1.4 The Data Out and Data In Busses

During pSYNC's active period, status bits must be available on the Data Out bus. On the 2810, this is accomplished by multiplexing the Status signals with the data lines from the Z-80. The output of the multiplexers is the Data Out bus, DO0-DO7. The signal pSYNC controls the state of the select lines. When pSYNC is active high, the status bits are multiplexed onto the Data Out bus. When pSYNC is inactive low, the data bits are multiplexed onto the Data Out bus. The Data Out bus can be placed in its high impedance state by DO DSB* for DMA operations.

The Data In bus is buffered by an 8-bit, three-state bus driver. This driver is disabled whenever pDBIN is inactive, except during DMA operations (indicated by the active BUS ACK*). It is also disabled under a number of other conditions. When either the ROM, the serial port, or the power-on jump circuitry is enabled, the driver is disabled, since data will be passed to the CPU on the internal bi-directional data lines. Front panel examination of memory will also disable the Data In bus while the front panel is commanding the CPU through the front panel data lines to fetch the data.

4.1.5 The Control Signals

Because the S-100 is an 8080-oriented bus, the signals on its control bus are generally the functional equivalents of the control signals of the 8080 itself. Thus the 2810 Z-80 CPU must emulate the 8080's control signals if it is to be S-100 compatible. With the control inputs this causes no problem, since the 8080's control inputs have their functional equivalents in the Z-80. The control outputs of the 8080, however, are quite different from those of the Z-80. The 2810 must then generate

8080-like control outputs from the Z-80 outputs. The following section describes how each 8080 control output is emulated by the 2810.

pSYNC In an 8080 system, this signal is generated by the processor during T1 (the first clock cycle) of every machine cycle and indicates to external devices that they can read the current status of the processor on the data bus.

The Z-80 has no equivalent signal; pSYNCH must be generated entirely through external circuitry. On the 2810 CPU, it is generated primarily by two flip-flops, one to generate pSYNC and the other to turn it off. The first flip-flop, U35b, is clocked by the rising edge of either the inverted M1*, MREQ*, or IOREQ*--whichever goes active first in a bus cycle. It is set by the state of the REFRESH* line: only when REFRESH* is inactive high will pSYNC, the Q output of the flip-flop, be high. This prevents pSYNC from being generated during the latter part of an M1 cycle when MREQ* goes low again with the signal REFRESH*. So that it can be turned off, pSYNC is input to the second flip-flop, U35a. When U35a is clocked, its Q* output clears U35b, turning off pSYNC. This flip-flop is clocked by the Φ2 clock during cycles in which M1* or IOREQ* is active and by the inverted Φ2 during bus cycles in which MREQ* only is active, causing pSYNC to last approximately one clock cycle in any bus cycle, as it does when generated by an 8080. Note that during an I/O cycle, pSYNC occurs during T2, instead of T1, since IOREQ* goes active then (see Figure 4-1). Its function remains exactly the same, however; it still marks the beginning of the bus cycle and indicates that valid status bits are on the bus.

pWR* PWR* indicates that valid data is present on the data bus and thus becomes active after pSYNC. The Z-80's write control output, WR*, serves the same function as pWR*; it simply needs to be disqualified during the active pSYNC. Flip-flop U34b serves this purpose. The flip-flop, its D input tied high, is clocked on the falling edge of pSYNC and cleared on the rising edge. Thus its Q* output will be low only when pSYNC is inactive. The Q* output is ORed with WR*. Only if both signals are low will the output of the OR gate, pWR*, be active low. See Figure 4-1.

pDBIN In 8080-based S-100 systems, pDBIN indicates that the data bus is conditioned to accept data from external devices. It goes active with the falling pSYNC signal and occurs during Read and Interrupt Acknowledge cycles. On the

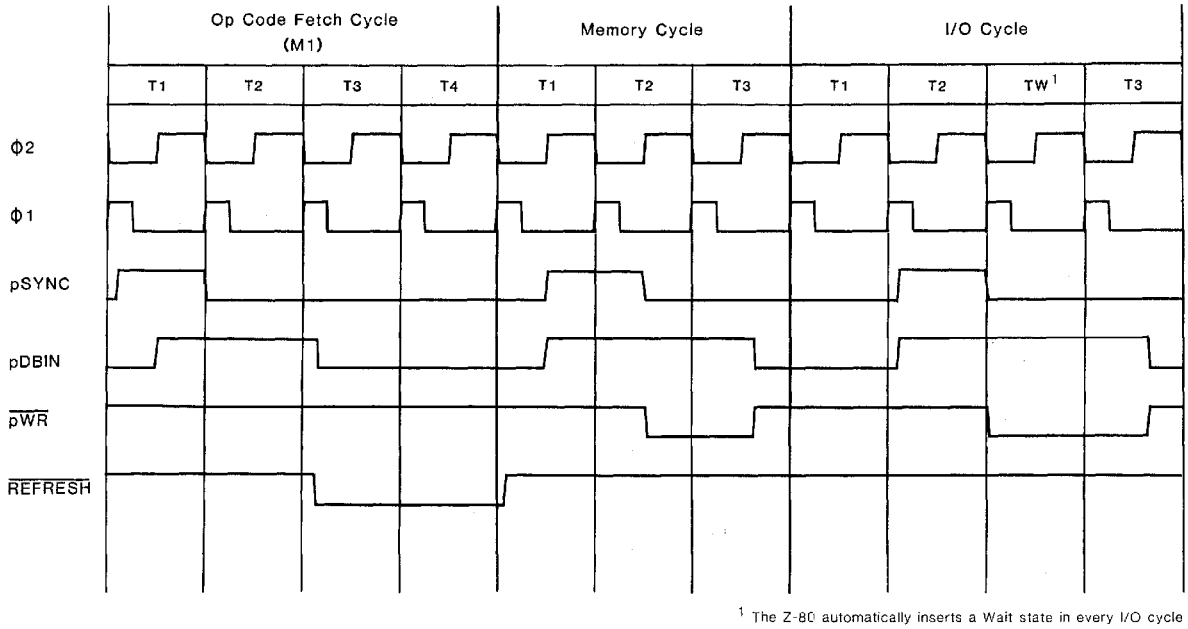


FIGURE 4-1 TIMING WAVEFORMS FOR SELECTED CLOCK AND CONTROL SIGNALS

2810, the Z-80's Read signal, RD*, is inverted and ORed with sINTA, producing pDBIN. Thus pDBIN will be active whenever either RD* or sINTA is active. Note that pDBIN is not disqualified by pSYNC; during a Read cycle it will be active while pSYNC is active (see Figure 4-1). This allows a longer memory access time, yet causes no bus conflict. During the time pSYNC is active, the Data In Bus and the internal data lines are not being used, the status bits having been gated onto the Data Out bus from the status lines themselves.

pINTE The signal pINTE indicates the state of the processor's internal interrupt enable flip-flop. The 8080 generates this signal itself; on the 2810 board it is generated by an external flip-flop, U14a, since the Z-80 has no equivalent signal. The state of the Z-80 internal interrupt flip-flop can be set by the EI (Enable Interrupts) and DI (Disable Interrupts) commands. In binary these commands are 1111 1011 and 1111 0011. Note that these commands are distinguished by the state of bit 3 only. The rest of the bit pattern is the same. U32 monitors the data lines D0-D2 and D4-D7 for the EI/DI bit pattern. When it occurs, U32 enables flip-flop U14b, allowing it to be clocked by M1* going inactive. When U14b is clocked, its Q output in turn clocks U14a. If D3 is high, the output of U14a, pINTE, will be set high and the Interrupt Enable LED lit. If D3 is low, pINTE will be low. U14a is cleared and pINTE made inactive low by either the active pRESET* or sINTA. Thus the state of pINTE can be changed only by an EI or DI op code, a system reset, or an Interrupt Acknowledge. It should therefore accurately reflect the state of the processor internal interrupt flip-flop.

pHLDA pHLDA goes active in an 8080 system in response to a HOLD request, indicated by the active pHOLD*. In the Z-80, there are two equivalent signals, BUSRQ* (Bus Request) and BUSAK* (Bus Acknowledge). Thus on the 2810, BUSAK* is simply inverted to create pHLDA.

pWAIT The signal pWAIT indicates that the processor has entered a Wait state. The Z-80 has no equivalent signal. On the 2810 this signal is generated by the Wait state flip-flop, U34a. This flip-flop is preset every time a device requests a Wait state. This forces its Q output, pWAIT, high. This signal remains high until Preset is released and the flip-flop is clocked by the rising edge of the 8 MHz clock from U24. Please note that on the 2810, pWAIT may be active high even if the processor itself has not entered a Wait state. pWAIT goes high whenever a device requests a Wait state. The CPU, however, samples the

state of its Wait input only on the falling edge of pCLOCK during T2. A device must make its first Wait request then or the CPU does not recognize it.

4.1.6 The Status Bus

The status bus on the S-100 bus communicates to external devices the current state of the processor--i.e, what bus cycle it is in--and qualifies the nature of the address on the address lines. At the beginning of each instruction cycle, the 8080 puts the 8-bit status information from its internal register out on the data bus where it can be sampled by external devices. The active pSYNC indicates its stable presence on the bus. At the same time the status information is latched in the external status latch to generate the status bus signals. The meaning of the status bits are summarized in the table below.

DATA BUS BIT	D7	D6	D5	D4	D3	D2[1]	D1	D0
STATUS BIT	MEMR	INP	M1	OUT	HLTA		$\overline{WO}$	INTA
Instruction Fetch	1	0	1	0	0	x	1	0
Memory Read	1	0	0	0	0	x	1	0
Memory Write	0	0	0	0	0	x	0	0
Input Read	0	1	0	0	0	x	1	0
Output Write	0	0	0	1	0	x	0	0
Interrupt Acknowledge	0	0	1	0	0	x	1	1
Halt Acknowledge	1	0	0	0	1	x	1	0

[1] In 8080 systems D2 is the STACK bit. On the 2810 sSTACK is not generated. See 2.1.6.

TABLE 4-1 STATUS WORD DEFINITIONS

Because the status of the Z-80 can be decoded from the control outputs themselves, the Z-80 has no internal status register. Therefore, the S-100 Status lines must be generated from the control outputs. When pSYNC is active, the status lines, with two exceptions, are gated onto the data bus by the bus multiplexers. Two of the status lines, sWO* and sINTA, will not always be active when pSYNC is active. The WO and INTA status bits must be generated separately.

sINTA This signal indicates that the CPU has accepted an interrupt and is awaiting instruction from the interrupting device. The Z-80 indicates an Interrupt Acknowledge cycle by both M1* and IOREQ* being active in the same bus cycle. IOREQ* in this case goes active almost 2 1/2 clock cycles after M1* and is the Z-80's read

strobe for this cycle. The bus signal sINTA is generated by ANDing the inverted signals M1* and IOREQ*. Thus sINTA will be high only when IOREQ* is active. This is important since the 2810 uses sINTA to generate the bus Data In strobe, pDBIN, during an Interrupt Acknowledge cycle. However, sINTA generated this way does not become active until T3--too late to be gated onto the Data Out bus by pSYNC. Therefore the INTA status bit is generated by the inverted M1* being ANDed with RD*. Only when RD* is inactive high will the INTA bit be high. Since an active M1* occurs without an active RD* only during an Interrupt Acknowledge cycle, the state of the INTA bit accurately reflects the bus cycle.

sWO* When active low, sWO* indicates that the CPU is in a Write cycle. On the 2810 board, sWO* and the status bit WO are generated by two different methods. The status signal is simply the Z-80's WR* signal. However, WR* goes active low during T2 of a Memory Write cycle--too late to be present on the data bus when pSYNC is active. Thus the status bit WO is generated by either MREQ* or IOREQ* being active while RD* is inactive. Only during an I/O or Memory Write cycle would RD* be inactive. The method by which the status bit WO* is generated cannot be used to generate sWO*, since sWO* would then be generated during an Interrupt Acknowledge cycle.

sHLTA sHLTA and the Z-80 HALT* both indicate that the CPU has received a HALT instruction and is awaiting an interrupt. Thus sHLTA on the 2810 board is the inverted HALT*. The active sHLTA lights the Halt Acknowledge LED.

sOUT Indicating that the CPU is outputting data to an I/O device, this signal is generated when both IORQ* and WR* are active.

sM1 This signal is active during the Op Code Fetch cycle of an instruction execution cycle and during an Interrupt Acknowledge cycle in both the 8080 and Z-80. Thus sM1 is generated by the inverted M1* of the Z-80.

sINP Indicating that the CPU is reading data from an I/O device, this signal is active when both IORQ* and RD* are active.

sMEMR Active high when during a Memory Read cycle, sMEMR is active only when both MREQ* and RD* are active.

4.1.7 The Wait Circuitry

The WAIT* input to the Z-80 is low when any of the following four conditions occurs: 1) the XRDY line is pulled low; 2) the pRDY line is pulled low; 3) M1* is active when the M1 Wait states are enabled; 4) the ROM is enabled when the Z-80 is operating at 4 MHz. U21c monitors for these conditions, its output going high whenever one of them is met. This high is inverted and pulls the Preset line to the Wait flip-flop, U34a, low. The resulting low on the flip-flop's Q* output pulls the WAIT* input to the Z-80 low. Q* will remain low as long as U21c continues to pull the Preset input to the flip-flop low. As soon as U21c releases the Preset line, the flip-flop will be reset when it is clocked by the rising edge of the 8 MHz clock from U24.

The 8 MHz clock is used to ensure that one and only one Wait state is generated per cycle in which the M1 or ROM Wait state circuitry is active. A Wait request from either circuit is qualified by pSYNC; only if pSYNC is active will U21c be pulled high. In most memory cycles, qualifying the signal with pSYNC ensures one Wait state per cycle. However, during an M1 cycle, pSYNC goes inactive before T2. Resetting the Wait flip-flop with the 8 MHz clock allows WAIT* to remain active long enough for the CPU to sample it, but not so long as to generate an extra Wait state.

4.1.8 The Rom Enable Circuitry

Address lines A0-A10 from the Z-80 are input directly to the ROM, since eleven address bits are necessary to select one location out of 2K. Address lines A11-A15 are input to the Address decoding ROM, U9, along with MREQ* and PHANTOM*. When U9 receives address bits on the high order address lines in the range of F0-F7 when PHANTOM* is inactive and MREQ* active, the output of U9 is pulled low. If the ROM enable jumper is set ON, this low is jumpered to the enable inputs of the ROM, enabling it and lighting the ROM LED. At the same time, the Data In bus will be disabled. If either PHANTOM* is active or MREQ* is inactive, U9's output will be high, disabling the ROM.

4.1.9 Power-on Jump Circuitry

The power-on jump circuitry works by placing on the data bus the unconditional jump command C3 (11000011) during the first M1 cycle after power-on or a system reset and the low byte and high byte of the jump address during the two memory read cycles that follow a jump instruction. Because the Power-on Jump circuitry, when enabled, disables the Data In bus, there is no conflict with memory.

The correct order and timing of the command and address bytes are achieved through the use of four D-type flip-flops and two 8-line-to-4-line multiplexers. The flip-flops are used as a 4-bit shift register, the Q output of one flip-flop being tied to the D input of the next. The flip-flops are triggered by the inverted RD*. When the CPU is reset or turned on, it executes an M1 cycle, pulling the RD* line low. This triggers the first flip-flop, the output of which simply is tied to the next. In the meantime, the A input lines to the multiplexers are tied in such a way as to generate the data byte 11000011, which is multiplexed onto the internal data bus and read by the CPU. The CPU then executes a memory read cycle as a result of receiving a jump instruction, pulling the RD* line low again. This clocks the second flip-flop, the outputs of which change the state of the A input lines such that they reflect the address settings on the Low Byte Address jumpers. The low address byte thus can be read by the CPU. During the next memory read cycle, the third flip-flop is clocked, its output changing the state of the Select inputs on the the multiplexers, allowing the B inputs to the multiplexers onto the internal data bus. Because the B inputs reflect the settings of the High Byte Address jumpers, the CPU receives the high byte address. After having received the jump address, the CPU executes another M1 cycle to fetch the op code at the jump address. When RD* goes low again for the M1 cycle, the fourth flip-flop is clocked, the output of which disables the multiplexer, effectively disqualifying the power-on jump circuitry, and enables the Data In bus, allowing the CPU to read from the jump address. When the system is reset, pRESET* clears all the flip-flops, allowing the process to begin again.

4.2 THE SERIAL I/O PORT

National's 8250 Asynchronous Communications Element performs almost all the necessary functions to interface the CPU to a serial peripheral device. It takes the parallel data it receives from the CPU and converts it to serial, adds start and stop bits, and transmits it over a single wire one bit at a time. When

receiving serial data from the peripheral, it does the reverse, stripping the start and stop bits from the data and converting the data to parallel for output over the eight internal data lines to the CPU. The 8250 requires a external clock, provided on the 2810 by a 1.8432 crystal oscillator. It also requires some minimal circuitry to interface it to the CPU and the peripheral.

4.2.1 The CPU Interface

The 8250 is selected when its chip select inputs, CS0 and CS1, are high. CS1 is high when IOREQ* is active when M1* is inactive. (The qualifying of IOREQ* with M1* is necessary to distinguish a valid I/O cycle from an Interrupt Acknowledge cycle.) CS0 is high when the address bits on A3-A7 match the settings of the Serial Address Select jumpers. Read/Write control is provided by pDBIN and pWR*, which control the Data Out Strobe and Data In Strobe of the 8250 respectively, allowing the CPU to read and write to the registers selected by A0-A2. When the CPU is reading from the 8250's registers, the 8250's DDIS* line goes active, disabling the CPU's Data In bus, since data will be transferred on the 2810's internal bi-directional data lines.

4.2.2 The Peripheral Interface

The Peripheral side of the interface consists of a set of line drivers and receivers which translate between the TTL signals of the 8250 and the nominal +5 to -5 volt signals required by the RS-232-C interface. The 8250's handshake lines are also used in a way which requires explanation.

The RS-232-C specifications are concerned with the communication link between a MODEM (or data communications equipment, DCE for short) and a computer terminal (or data terminal equipement, DTE for short). Thus equipment conforming to the RS-232-C specifications must take on the role of either a DCE or DTE device. The 2810's serial port is designed to be the DCE side of the interface. The problem here is that the 8250's handshake lines are defined as those of a DTE device. Thus the roles of the 8250 handshake lines must change. For example, the input into the 8250's CTS (Clear To Send) pin comes actually from the DCE-type connector's RTS (Request to Send) line. The 8250's output DTR (Data Terminal Ready) appears on the connector's DSR (Data Set Ready) line. The 8250's auxiliary output, OUT 1, is

tied to the connector's Received Line Signal Detect (RLSD), allowing RLSD to be available to signals that require the signal. The following table summarizes the connections between the 8250 and the DCE-type connector.

8250	CONNECTOR
DSR	DTR
CTS	RTS
RTS	CTS
DTR	DSR
OUT 1	RLSD

TABLE 4-3

If you have reason to consult an 8250 data sheet, please keep these role changes in mind. The serial input from the peripheral is also connected to the 8250's Ring Indicator input to support the auto-baud feature of the 2810's firmware.

APPENDIX A

THE 2810 Z-80 CPU BUSSES

A.1 THE SYSTEM BUS

A.1.1 The S-100 Bus

The S-100 bus came into being with the Altair line of microcomputers using the 8080 microprocessor. Known then as the Altair bus, it was adopted by many other microcomputer manufacturers and became an unoffical industry standard; hence the name "standard-100" bus.

Recently the IEEE has undertaken the development of an official standard for the S-100 bus. The proposed standard differs from the unofficial standard in the definitions of several lines. The changes reflect in part the changes in the microcomputer industry. New processors have come onto the market with new capabilities: 16-bit data transfer, dynamic memory refresh, nonmaskable interrupts, etc. And as system design has become more sophisticated, there has been a move away from front panels. In the proposed standards, for example, several signals previously used for front panel functions have been eliminated and the lines themselves reserved for future use. The differences between the proposed standard and the unofficial standard present a dilemma for the manufacturer of S-100 product: Should he conform to the proposed standard or aim for current product compatability?

The 2810 board represents a compromise; we have conformed to the proposed standards where possible without sacrificing compatiblity with the major S-100 systems currently on the market. In the next section, we define the signals used by the 2810 system bus, and make note of discrepancies between our line use and those of the unofficial or the proposed standards.

A.1.2 The 2810 System Bus

The following are definitions of the signals used by the 2810 system bus. We have followed the convention of indicating active low signals with an asterisk (*) following the signal mnemonics.

For clarity's sake, we have divided the signals on the 2810 bus into 6 categories: 1) the address and data busses, 2) the status bus, 3) processor control signals, 4) front panel control, 5) DMA control, and 6) system utilities.

1. Data and Address Lines

A0-A15 The 16-bit parallel address lines.

DI0-DI7 The 8-bit parallel data input lines.

DO0-DO7 The 8-bit parallel data output lines.

2. The Status Signals

The Status signals indicate the nature of the bus cycle in progress and are the functional equivalents of the outputs of the 8080's status latch. The mnemonics for the status lines begin with a lower case "s."

sINTA The Interrupt Acknowledge signal indicates that the CPU has accepted an interrupt.

sWO* The Write/Output signal indicates that the CPU is in a write or output cycle.

sHLTA The Halt Acknowledge signal indicates that the CPU is executing a HALT instruction.

sOUT The Output signal indicates that the CPU is executing an output instruction.

sM1 The M1 cycle signal indicates that the CPU is in the Op Code fetch portion of an instruction cycle.

sINP The Input signal indicates that the CPU is executing an input instruction.

sMEMR The Memory Read signal indicates that the CPU is reading from memory.

3. The Processor Control Signals

The processor control signals are concerned with synchronizing the movement of data to and from the processor during any machine cycle. With the exception of NMI*, REFRESH*, and MREQ*, they are the functional equivalents of the 8080 control inputs and outputs and are generally prefixed with the letter "p."

Outputs

pSYNC The Sync signal indicates the presence of status bits on the Data Out bus.

pDBIN The Data Bus In signal gates the data on the Data In bus onto the 2810's internal data lines.

pWR* The Write signal indicates the presence of valid data on the Data Out bus.

pHLDA The Hold Acknowlege signal indicates that the CPU has relinquished control of the bus in response to a Hold request.

pWAIT The Wait signal indicates that the CPU has entered a Wait state. In the proposed standard, this signal is eliminated and the line is reserved for future use.

pINTE The Interrupt Enable signal indicates that the CPU will respond to interrupt requests. In the proposed standard, this signal is eliminated and the line is reserved for future use.

REFRESH* (Optional) The Refresh signal is a control signal for dynamic memory refresh. During the time REFRESH* is active, a dynamic memory refresh is totally transparent to the processor. This line is left undefined by the proposed standard.

MREQ* (Optional) The Memory Request signal from the Z-80 indicates that the address bus holds a valid address for a memory read or write. This line is left undefined by the proposed standard.

Inputs

pRDY The Ready signal allows external devices to place the CPU in a Wait state.

pINT* The Interrupt signal allows external devices to request service from the CPU.

pHOLD* The Hold signal allows external devices to request control of the bus.

NMI* (Optional) The Nonmaskable Interrupt signal allows external devices to assert an interrupt request that

cannot be masked off by the CPU.

pRESET* The Reset signal, when active low, resets the CPU. It is generated usually by a front panel switch and is also asserted by POC*.

4. Front Panel Control

XRDY The External Ready signal is a ready line generally used by front panels for single-step or stop operations.

SSW DSB* The Sense Switch Disable signal disables the data input lines DI0-DI7 so that the input from the front panel sense switches can be strobed onto the internal bi-directional data bus. The proposed standard eliminates this signal and reserves the line for future use.

RUN The Run signal indicates the state of the Run/Stop flip-flop on the front panel is set to Run. This proposed standard eliminates this signal and reserves the line for future use.

SS The Single Step signal indicates a single step is being performed. The proposed standard eliminates this signal and reserves the line for future use.

5. DMA Control

STAT DSB* The Status Bus Disable signal allows external devices to place the status bus driver in its high impedance state.

C/C DSB* The Command/Control Disable signal allows external devices to place the control bus driver in its high impedance state.

ADD DSB* The Address Disable signal allows external devices to place the address bus driver in its high impedance state.

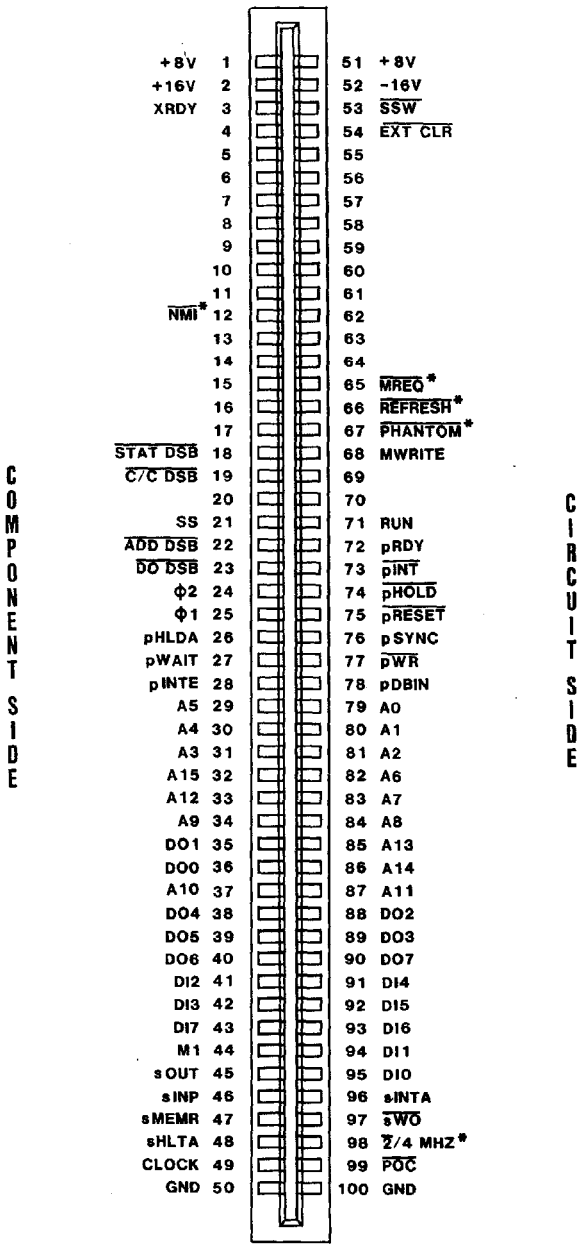
DO DSB* The Data Out Disable signal allows external devices to place the Data Out driver in its high impedance state.

6. System Utilities

POC*	Active only during power-on, the Power-On Clear signal asserts EXT CLR* and RESET*.
EXT CLR*	When active, the External Clear signal resets external devices.
MWRT	The Memory Write signal indicates that the current data on the Data Out bus is to be written into the memory location specified by the address bus. Often generated by front panel devices, it usually is used for front panel memory deposit.
PHANTOM	(Optional) The Phantom signal is used to control memory overlay. On the 2810 board, an external device can use it to overlay the memory space occupied by the on-board ROM.
Φ1	Φ1 is the phase one clock for the 8080.
Φ2	Φ2 is the phase two clock for the 8080.
CLOCK	Clock is a 2 MHz signal, regardless of processor speed.
2*/4 MHZ	(Optional) When high, this signal indicates the processor is operating at 4 MHz. When it is low, it indicates the processor is operating at 2 MHz. The early S-100 bus used this line for the sSTACK signal; the proposed standard suggests this line be used for the signal ERROR*.
+8 VOLTS	This is the unregulated +8 Volts from the power supply.
+16 VOLTS	This is the unregulated +16 Volts from the power supply.
-16 VOLTS	This is the unregulated -16 Volts from the power supply.

A.1.3 The System Bus Pin Assignments

2810 BUS CONNECTOR PINOUT



TOP VIEW

*Jumper-enabled signals

A.2 SERIAL INTERFACE BUS

A.2.1 Signal Definitions

The following are the RS-232-C signals used by the asynchronous serial port.

Inputs

DTR — Data Terminal Ready. When active, this signal indicates that the peripheral is ready to establish a communications link and receive or transmit data to/from the 8250.

RTS — Request to Send. When active, this signal indicates that the peripheral's transmit data buffer is full and is ready to transmit data.

TxD — Transmit Data. This signal is the serial data input from the peripheral to the 8250.

Outputs

RxD — Receive Data. This signal is the serial output from the 8250 to peripheral.

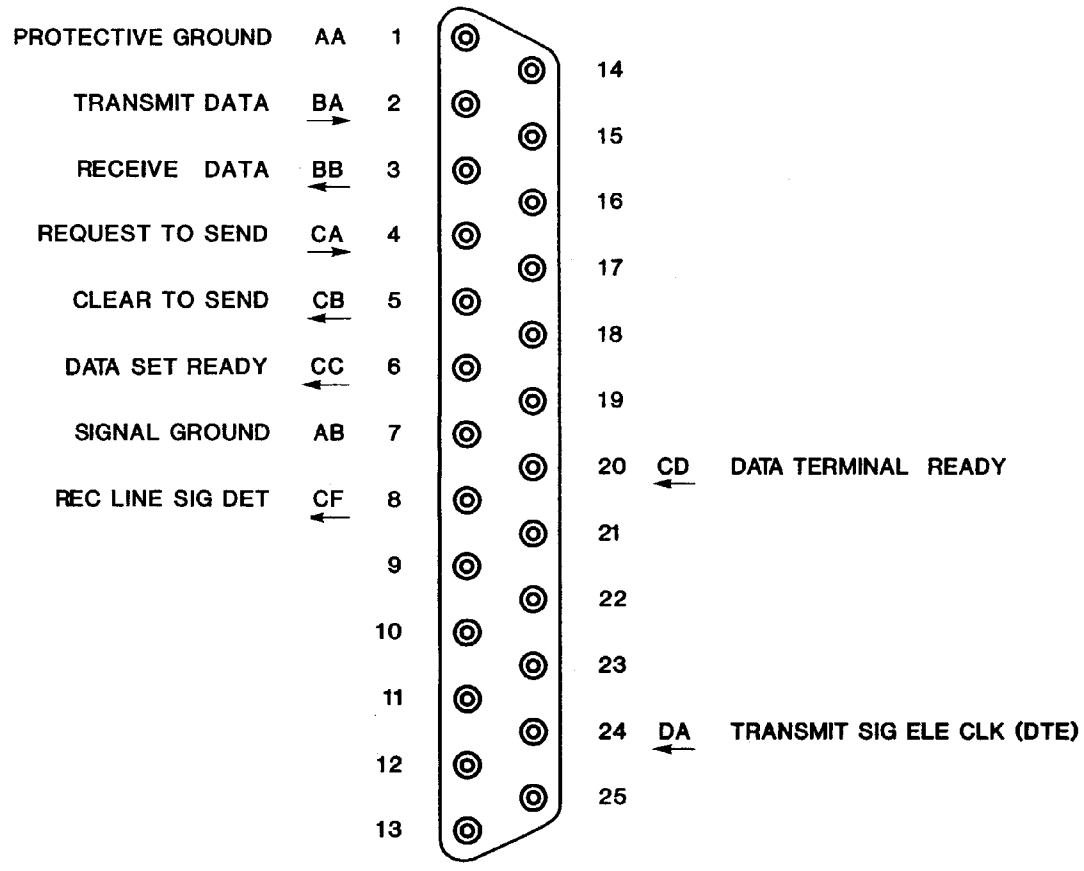
CTS — Clear To Send. The active signal informs the peripheral that the 8250 is ready to send data.

DSR — Data Set Ready. This informs the peripheral that the 8250 is ready to communicate.

RLSD — Received Line Signal Detect. This signal indicates that the 8250 has detected a signal from the peripheral.

A.2.2 RS-232-C Pin Assignments

2810 DCE-TYPE CONNECTOR PIN ASSIGNMENTS
EIA RS-232-C STANDARD



DB-25S (FEMALE)
FRONT VIEW

APPENDIX B

THE 2810 ACCESSIBLE REGISTERS

B.1 THE Z-80 PROGRAM ACCESSIBLE REGISTERS

Twenty-two of the Z-80's internal registers are accessible to the programmer. Figure B-1 shows the configuration of the accessible registers, while sections B.1.1 through B.1.3 give a short description of them.

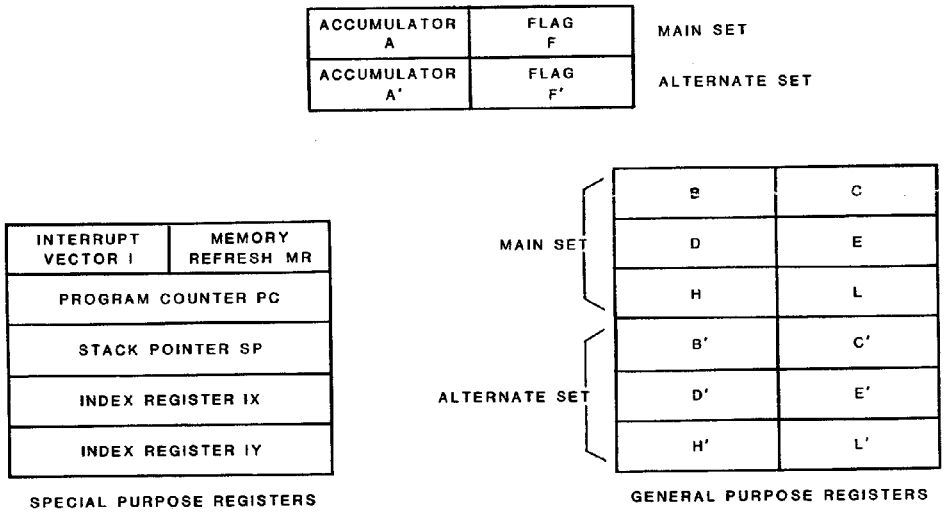


FIGURE B-1 Z-80 REGISTERS

B.1.1 Accumulator and Flag Registers

The two 8-bit accumulators hold the result of arithmetic and logical operations while their associated flag registers indicate the special results of such operations. A single exchange instruction allows the programmer to work with either pair of registers.

B.1.2 Special Purpose Registers

Program Counter (PC)--This 16-bit register holds the memory address of the current instruction. The PC is automatically

incremented after its contents have been transferred to the address lines. A program jump overrides the incrementer and places a new value in the PC.

Stack Pointer (SP)--This 16-bit register holds the address of the current top of a stack located anywhere in external RAM memory. The PUSH and POP instructions push data from specific registers onto the stack or pop the data off the stack into specific registers.

Index Registers (IX and IY)--These two independent 16-bit registers hold a base address that is used in indexed addressing modes. This base address is used in conjunction with a displacement byte (a two's complement integer) in an indexed instruction to specify a location in memory.

Interrupt Page Address Register (I)--This register is used for interrupt response mode involving an indirect call to memory. The register stores the high order 8-bits of the indirect address; the interrupting device provides the lower 8-bits. (See your programming manual for more details.)

Memory Refresh Register (R)--This register is used as counter register for dynamic memory refresh. It contains a refresh address which is placed on the address bus during the last two clock cycles of every M1 cycle. The address is then automatically incremented. You would not normally access this register, although you can load it for testing purposes.

B.1.3 General Purpose Registers

The general purpose registers consist of a main and alternate set of six 8-bit registers. They can be used as individual 8-bit registers or as 16-bit register pairs. The main set pairs are BC, DE, and HL; the alternate set pairs are BC', DE', and HL'. A single exchange command allows the programmer to select either set. See your Z-80 programming manual for more details.

B.2 THE 8250 ADDRESSABLE REGISTERS

There are nine accessible registers of concern in the 8250. These registers are addressed through the low-order three bits of the serial port address. The registers are addressed as follows:

DLAB	A2	A1	A0	REGISTER
0	0	0	0	Receiver Buffer (read), Transmitter Holding Register (write)
0	0	0	1	Interrupt Enable
x	0	1	1	Line Control
x	1	0	0	Peripheral Control
x	1	0	1	Line Status
x	1	1	0	Peripheral Status
1	0	0	0	Divisor Latch (least significant byte)
1	0	0	1	Divisor Latch (most significant byte)

TABLE B-1 8250 REGISTER ADDRESSING

Note that the address lines alone are not always sufficient to select a register; the state of the Divisor Latch Bit (DLAB) of the Line Control Register determines which of the registers sharing the same address will be selected.

The contents and function of each register are summarized in Table B-2 below. In addition, six of the registers are described in more detail in the the following pages. If you consult the 8250's data sheet, you will notice discrepancies between our bit descriptions and the data sheet's descriptions for some of the bits. Such discrepancies are more apparent than real: the data sheet assumes the 8250 will be used as a DTE device and thus has named the bits accordingly; we use it as a DCE device and thus have renamed the bits. Note that since we do not use the 8250's interrupt capabilities, the first four bits of the Interrupt Enable Register should be set to 0.

BIT NUMBER	REGISTER ADDRESS								
	0 DLAB=0	0 DLAB=0	1 DLAB=0	3	4	5	6	0 DLAB=1	1 DLAB=1
	Receiver Buffer Register (Read Only) RBR	Transmitter Holding Register (Write Only) THR	Interrupt Enable Register IER	Line Control Register LCR	Peripheral Control Register PCR	Line Status Register LSR	Peripheral Status Register PSR	Divisor Latch DLL	Divisor Latch MSR
0	Data Bit 0	Data Bit 0	Set to 0	Word Length Select Bit 0	DSR	Data Ready	Delta CTS	Bit 0	Bit 8
1	Data Bit 1	Data Bit 1	Set to 0	Word Length Select Bit 1	CTS	Overrun Error	Delta DSR	Bit 1	Bit 9
2	Data Bit 2	Data Bit 2	Set to 0	Number of Stop Bits	RLSD	Parity Error	0	Bit 2	Bit 10
3	Data Bit 3	Data Bit 3	0	Parity Enable	Set to 1	Framing Error	0	Bit 3	Bit 11
4	Data Bit 4	Data Bit 4	0	Even Parity Select	Loop	Break Interrupt	RTS	Bit 4	Bit 12
5	Data Bit 5	Data Bit 5	0	Stick Parity	0	Transmitter Holding Register Empty	DTR	Bit 5	Bit 13
6	Data Bit 6	Data Bit 6	0	Set Break	0	Transmitter Shift Register Empty	0	Bit 6	Bit 14
7	Data Bit 7	Data Bit 7	0	Divisor Latch Access Bit	0	0	0	Bit 7	Bit 15

TABLE B-2 8250 REGISTER SUMMARY

B.2.1 Peripheral Control Register

This register controls the interface with the peripheral. Bits 0 through 2 control the state of the DSR, CTS, and RLSD outputs. To set one of these signals active high, write a 1 to its bit. Bit 4, when set to 1, enables loopback testing, in which the data in the transmitter register is looped to the receiver register, without having been output. Thus data that is transmitted is immediately received. See Table B-2 for a summary of the register.

B.2.2 Line Control Register

The line control registers allows you to specify the serial data format. For ease of programming, you can examine the contents of the line control register at any time. The bit definitions and functions are summarized in Table B-3.

BIT NO.	BIT NAME	FUNCTION	DEFINITION
0 1	Word Length Select	Bit 0 and B 1 select the number of bits in each serial character.	Bit 0 Bit 2 = Word Length 0 0 5 bits 0 1 6 bits 1 0 7 bits 1 1 8 bits
2	Stop Bits Select	Selects the number of stop bits in each serial character.	0 = 1 Stop bit 1 = 1 1/2 Stop bits (5-bit word) 2 Stop bits (6-, 7-, 8-bit words)
3	Parity Enable	Selects whether or not a parity bit is generated between the last data bit and stop bit(s).	0 = No Parity bit 1 = Parity bit
4	Even Parity Select	Selects whether the parity bit will make an even or odd number of 1s in the data word.	0 = Odd parity 1 = Even parity
5	Stick Parity	Selects whether a 1 or a 0 will be sent in the parity bit position.	Bit 3 Bit 4 Bit 5 = Stick parity bit x x 0 None 1 0 1 1 1 1 1 0
6	Set Break	Selects whether or not sOUT is forced to spacing (logic 0)	0 = Break disabled 1 = Break (spacing enabled)
7	Divisor Latch	Determines which register of those sharing the same address is selected.	0 = Receiver buffer or transmitter holding register 1 = Divisor latches

TABLE B-3 LINE CONTROL REGISTER

B.2.3 Peripheral Status Register

This register indicates the current state of the control lines from the peripheral device. The first two bits are set to a logic 1 whenever the state of the control line has changed since the peripheral status register was last read by the CPU. See Table B-2 for a summary of the register's contents.

B.2.4 Line Status Register

This register provides status information to the CPU concerning the data transfer. The bit definitions and functions are summarized in Table B-4 below. Except where otherwise noted, the bits are reset when the CPU reads the line status register.

BIT NO.	NAME	DEFINITION
0	Data Ready (DR)	Set to 1 if the Receiver Buffer is full. Reset by CPU reading buffer or writing a 0 to it.
1	Overrun Error (OE)	Set to 1 if the CPU did not read the data in the Receiver Buffer before the next character was transferred to it.
2	Parity Error (PE)	Set to logic 1 when a parity error is detected.
3	Framing Error (FE)	Set to 1 if incoming character has no valid stop bit.
4	Break Interrupt (BI)	Set to 1 whenever the received data input is held in the spacing state for longer than a full word transmission time.
5	Transmitter Holding Register Empty (THRE)	Set to 1 when Transmitter Holding Register is empty, having transferred its data to the Transmitter Shift Register. Reset when CPU loads the THR.
6	Transmitter Shift Register Empty (TSRE)	Set to 1 when Transmitter Shift Register is idle. Reset upon data transfer from THR. A read-only bit.
7		Permanently set to 0

TABLE B-4 LINE STATUS REGISTER

B.2.5 Divisor Latch Registers

The divisor latch registers are used to select the baud rate you wish. The programmable baud rate generator can divide the 1.8432 Mhz clock input by any divisor from 1 to (2**16)-1. The output frequency of the baud rate generator is 16X the baud rate (divisor# = frequency input/ (baud rate * 16)). The divisor is stored in the two divisor latches in a 16-bit binary format. Table B-5 shows the divisors for some common baud rates.

BAUD RATE	DIVISOR FOR 16x CLOCK	PERCENT ERROR DIFFERENCE BETWEEN DESIRED AND ACTUAL
50	2304	-
75	1536	-
110	1047	0.026
134.5	857	0.058
150	768	-
300	384	-
600	192	-
1200	96	-
1800	64	-
2000	58	0.69
2400	48	-
3600	32	-
4800	24	-
7200	16	-
9600	12	-
19200	6	-
38400	3	-
56000	2	2.86

TABLE B-5 BAUD RATE DIVISOR

APPENDIX C

FIRMWARE LISTING

```
CP/M MACRO ASSEM 2.0    #001    MOSS 2.2 MONITOR

                ;
                        TITLE   'MOSS 2.2 MONITOR'
                        PAGE    68
                        MACLIB  Z80
                ;
                ; MOSS MONITOR (VERSION 2.2)
                ;
                ; 20 JUNE 1980
                ; ALL RIGHTS RESERVED BY ROBERT B. MASON
                ;
F000            MOSS:   ORG     0F000H
F000 =          ROM:    EQU     0F000H  ;ROM START ADDRESS
0000 =          WSVEC:  EQU     0       ;VECTOR FOR WARM RESTART
0002 =          NBKPTS: EQU     2       ;NUMBER OF BREAKPOINTS
0013 =          CTRLS:  EQU     13H     ;ASCII DC3
000D =          CR:     EQU     0DH     ;ASCII CARRIAGE RETURN
000A =          LF:     EQU     0AH     ;ASCII LINE FEED
000C =          FMFD:   EQU     0CH     ;ASCII FORM FEED
0007 =          BELL:   EQU     7       ;ASCII CNTRL CHAR TO RING THE BELL
0003 =          IOBYTE: EQU     3       ;ADDRESS OF I/O CONTROL BYTE
0020 =          SDATA:  EQU     20H     ;SERIAL DATA PORT BASE ADDRESS
0021 =          SINTEN: EQU     SDATA+1 ;SERIAL INTERRUPT ENABLE REGISTER
0022 =          SIDENT: EQU     SDATA+2 ;SERIAL INTERRUPT IDENTIFICATION REGIS
0023 =          SLCTRL: EQU     SDATA+3 ;SERIAL LINE CONTROL REGISTER
0024 =          SMDMCT: EQU     SDATA+4 ;SERIAL MODEM CONTROL REGISTER
0025 =          SLSTAT: EQU     SDATA+5 ;SERIAL LINE STATUS REGISTER
0026 =          SMDMST: EQU     SDATA+6 ;SERIAL MODEM STATUS REGISTER
                ;
0006 =          SPSV:   EQU     6       ;STACK POINTER SAVE LOCATION
                ;
                ;
                ; REGISTER STORAGE DISPLACEMENTS FROM
                ; NORMAL SYSTEM STACK LOCATION.
                ;
0015 =          ALOC:   EQU     15H
0013 =          BLOC:   EQU     13H
0012 =          CLOC:   EQU     12H
0011 =          DLOC:   EQU     11H
0010 =          ELOC:   EQU     10H
0014 =          FLOC:   EQU     14H
0031 =          HLOC:   EQU     31H
0030 =          LLOC:   EQU     30H
0034 =          PLOC:   EQU     34H
0017 =          SLOC:   EQU     17H
0035 =          TLOC:   EQU     35H
0025 =          TLOCX:  EQU     25H
0020 =          LLOCX:  EQU     20H
                ;
0009 =          APLOC:  EQU     9
000B =          BPLOC:  EQU     11
000A =          CPLOC:  EQU     10
000D =          DPLOC:  EQU     13
000C =          EPLOC:  EQU     12
0008 =          FPLOC:  EQU     8
000F =          HPLOC:  EQU     15
000E =          LPLOC:  EQU     14
0007 =          XLOC:   EQU     7
0005 =          YLOC:   EQU     5
0002 =          RLOC:   EQU     2
0003 =          ILOC:   EQU     3
                ;
                ; JUMP TARGETS FOR BASIC INPUT/OUTPUT
                ;
F000 C35BF0     CBOOT:  JMP     INIT    ;COLD START
F003 C346F6     CONIN:  JMP     CI      ;CONSOLE INPUT
F006 C356F6     READER: JMP     RI      ;READER INPUT
```

```
CP/M MACRO ASSEM 2.0    #002    MOSS 2.2 MONITOR

F009 C300F6    CONOUT: JMP     CO      ;CONSOLE OUTPUT
F00C C37CF6    PUNCH:  JMP     PO      ;PUNCH OUTPUT
F00F C310F6    LIST:   JMP     LO      ;LIST OUTPUT
F012 C323F6    CONST:  JMP     CSTS    ;CONSOLE STATUS
F015 C36AF1            JMP     IOCHK   ;PUT IOBYTE INTO (A)
F018 C365F1            JMP     IOSET   ;(C) HAS A NEW IOBYTE
F01B C38AF0            JMP     MEMCK   ;MEMORY LIMIT CHECK
F01E C394F6            JMP     RTS     ;IODEF- DEFINE USER I/O ENTRY POINTS
F021 C394F6            JMP     RTS     ;SPCL- I/O CONTROL
F024 C3CFF3            JMP     REST    ;BREAKPOINT ENTRY POINT
                       ;
                       ; TBL CONTAINS THE ADDRESSES OF THE ACTION ROUTINES
                       ;   THE EXECUTIVE USES IT TO LOOK UP THE DESIRED ADDRESS.
                       ;
F027 F8F0      TBL:    DW      ASGN
F029 09F1              DW      QPRT
F02B 09F1              DW      QPRT
F02D ACF1              DW      DISP
F02F F6F4              DW      EOF
F031 3CF1              DW      FILL
F033 FDF1              DW      GOTO
F035 D0F5              DW      HEXN
F037 4DF2              DW      INPT
F039 09F1              DW      QPRT
F03B 09F1              DW      QPRT
F03D 0EF5              DW      LEADER
F03F 5DF2              DW      MOVE
F041 09F1              DW      QPRT
F043 55F2              DW      OUPT
F045 09F1              DW      QPRT
F047 21F5              DW      QUERY
F049 4CF5              DW      READ
F04B 67F2              DW      SUBS
F04D 8FF2              DW      MTEST
F04F 09F1              DW      QPRT
F051 91F1              DW      COMP
F053 8DF5              DW      WRITE
F055 ECF2              DW      XMNE
F057 9FF4              DW      I8250
F059 82F1              DW      BYE
                       ;
                       ; THE COLD INITIALIZATION CODE
                       ;
F05B F3        INIT:   DI              ;DISABLE INTERRUPTS
F05C 313F00            LXI     SP,3FH  ;USE STACK TO INITIALIZE RESTARTS
F05F 2100C3            LXI     H,JMP*256       ;  WITH RESTART ERROR VECTORS
F062 11B2F6            LXI     D,RSTER
F065 0610              MVI     B,16    ;16 TIMES (64 BYTES)
F067 D5        INIT1:  PUSH    D
F068 E5                PUSH    H
                       DJNZ    INIT1

F069+10FC
F06B 3195F0            LXI     SP,FAKE-2       ;SET UP TEMPORARY STACK
F06E 3E00              MVI     A,0     ; SKIP THE NEXT INST
F06F                   ORG     $-1     ;SAVE A BYTE HERE
                       ;
                       ; MEMSIZ CALCULATES THE TOP OF CONTIGUOUS RAM.  IT SEARCHES
                       ;       FROM THE BOTTOM UP UNTIL A NON-RAM LOCATION IS
                       ;       FOUND.  IT THEN TAKES OFF FOR MONITOR WORK SPACE
                       ;       NEEDS AND RETURNS THE VALUE IN (H,L).
                       ;
F06F C5        MEMSIZ: PUSH    B       ;MONITOR START LOCATION
F070 0100F0            LXI     B,ROM
F073 21FFFF            LXI     H,-1    ;START OF MEMORY ADDRESS SPACE
F076 24        MEMSZ1: INR     H
F077 7E                MOV     A,M
F078 2F                CMA
F079 77                MOV     M,A
```

```
CP/M MACRO ASSEM 2.0     #003     MOSS 2.2 MONITOR

 F07A BE                 CMP      M
 F07B 2F                 CMA
 F07C 77                 MOV      M,A
                         JRNZ     MEMSZ2

 F07D+2004
 F07F 7C                 MOV      A,H      ;SEE IF ON MONITOR BORDER
 F080 B8                 CMP      B
                         JRNZ     MEMSZ1

 F081+20F3
 F083 25          MEMSZ2: DCR      H        ;TAKE OFF WORKSPACE
 F084 01DEFF             LXI      B,EXIT-ENDX-3*NBKPTS+1
 F087 09                 DAD      B
 F088 C1                 POP      B        ;(B,C) IS UNPREDICTABLE DURING INIT
 F089 C9                 RET

                 ;
                 ; ROUTINE MEMCHK FINDS THE CURRENT TOP OF CONTIGUOUS MEMORY
                 ;       (LESS THE MONITOR WORKSPACE) AND RETURNS THE VALUE.
                 ;
 F08A E5         MEMCK:  PUSH     H        ;SAVE (H,L)
 F08B CD6FF0             CALL     MEMSIZ   ;GET THE RAM SIZE
 F08E 7D                 MOV      A,L
 F08F D63C               SUI      60       ;TAKE OFF WORK SPACE
                         JRNC     MEMCK0

 F091+3001
 F093 25                 DCR      H
 F094 44         MEMCK0: MOV      B,H
 F095 E1                 POP      H
 F096 C9                 RET

                 ;
 F097 99F0       FAKE:   DW       FAKE+2
 F099 F9                 SPHL
 F09A 1145F4             LXI      D,EXIT
 F09D EB                 XCHG
 F09E 011D00             LXI      B,ENDX-EXIT
                         LDIR

 F0A1+EDB0
 F0A3 010600             LXI      B,3*NBKPTS
 F0A6 D5                 PUSH     D
 F0A7 E1                 POP      H
 F0A8 2B                 DCX      H
                         LDIR

 F0A9+EDB0
 F0AB 21E8FF             LXI      H,-24
 F0AE 39                 DAD      SP
 F0AF E5                 PUSH     H
 F0B0 23                 INX      H        ;ADJUST USER STACK LOCATION
 F0B1 23                 INX      H
 F0B2 220600             SHLD     SPSV     ;SAVE THE STACK INITIAL VALUE
 F0B5 160A               MVI      D,10     ;INITIALIZE REGISTER STORAGE AREA
 F0B7 C5         INIT2:  PUSH     B
 F0B8 15                 DCR      D        ;LOOP CONTROL
                         JRNZ     INIT2

 F0B9+20FC
                 ; INSERT I/O INIT CODE HERE
 F0BB CD94F6             CALL     RTS
 F0BE CD9FF4             CALL     I8250    ;INITIALIZE THE 8250
 F0C1 CD94F6             CALL     RTS
 F0C4 2190F4             LXI      H,LOGMSG ;LOG ONTO THE SYSTEM
 F0C7 CD95F6             CALL     PRTWD
                         JMPR     WINIT    ;GO TO MONITOR EXECUTIVE

 F0CA+1843
                 ;
                 ;  ROUTINE EXF READS ONE PARAMETER.  IT EXPECTS THE FIRST
                 ;       CHARACTER OF THE PARAMETER TO BE IN THE A REGISTER
                 ;       ON ENTRY.
                 ;
 F0CC 0601       EXF:    MVI      B,1      ;SET UP FOR ONE PARAMETER
 F0CE 210000             LXI      H,0
```

```
CP/M MACRO ASSEM 2.0    #004    MOSS 2.2 MONITOR

                        JMPR    EX1     ;FIRST CHARACTER IN A ALREADY
 F0D1+180C
                ;
                ;  ROUTINE EXPR READS PARAMETERS FROM THE CONSOLE
                ;       AND DEVELOPS A 16 BIT HEXADECIMAL FOR EACH ONE.
                ;       THE NUMBER OF PARAMETERS WANTED IS IN THE B REG
                ;       ON ENTRY.  A CARRIAGE RETURN WILL TERMINATE THE
                ;       ENTRY SEQUENCE; A BLANK OR A COMMA WILL END THE
                ;       CURRENT PARAMETER ENTRY.  EACH PARAMETER ONLY
                ;       TAKES THE LAST 4 DIGITS TYPED IN; ANY EXCESS IS
                ;       DISCARDED.  A NON-HEX DIGIT WILL TERMINATE THE
                ;       ENTRY SEQUENCE AND CAUSE A WARM BOOT OF THE MON.
                ;
                AS3:    DJNZ    AS2     ;PART OF THE ASSIGN CODE
 F0D3+1079
                EX3:    JRNZ    QPRT    ;NON-ZERO IS ERROR
 F0D5+2032
 F0D7 05        EXPR1:  DCR     B       ;MORE PARAMETERS?
 F0D8 C8                RZ              ;NO, RETURN
 F0D9 210000    EXPR:   LXI     H,0     ;INITIALIZE PARAMETER
 F0DC CD7BF3    EX0:    CALL    ECHO    ;GET NEXT NUMBER
 F0DF 4F        EX1:    MOV     C,A     ;SAVE CHAR FOR LATER USE
 F0E0 CDB0F3            CALL    NIBBLE
                        JRC     EX2     ;NOT A NUMBER, JUMP
 F0E3+3808
 F0E5 29                DAD     H       ;MULTIPLY BY 16
 F0E6 29                DAD     H
 F0E7 29                DAD     H
 F0E8 29                DAD     H
 F0E9 B5                ORA     L       ;ADD ON NEW DIGIT
 F0EA 6F                MOV     L,A
                        JMPR    EX0     ;GO GET NEXT DIGIT
 F0EB+18EF
 F0ED E3        EX2:    XTHL            ;PUT UNDER RETURN ADDRESS ON STACK
 F0EE E5                PUSH    H       ;RESTORE RETURN ADDRESS
 F0EF 79                MOV     A,C     ;REGET THE LAST CHARACTER
 F0F0 CDC3F3            CALL    P2C     ;TEST FOR DELIMITER
                        JRNC    EX3     ;JUMP IF NOT CARRIAGE RETURN
 F0F3+30E0
                        DJNZ    QPRT    ;CARRET WITH MORE PARAM MEANS ERROR
 F0F5+1012
 F0F7 C9                RET
                ;
                ; MAIN ACTION ROUTINES
                ;
                ;
                ; LOGICAL ASSIGNMENT OF PERIPHERALS
                ;
                ;THIS ROUTINE CONTROLS THE ASSIGNMENT OF PHYSICAL
                ;       PERIPHERALS TO THE FOUR LOGICAL DEVICE TYPES. IT
                ;       ALTERS IOBYTE (MEMORY LOCATION 0003) TO MATCH THE
                ;       CURRENT ASSIGNMENT.  THE FOUR LOGICAL DEVICES ARE
                ;       CONSOLE, READER, LIST, AND PUNCH.  IN ALL CASES,
                ;       THE TTY DEVICE IS SET UP AS THE DEFAULT DEVICE.
                ;
 F0F8 CD7BF3    ASGN:   CALL    ECHO    ;GET THE LOGICAL DEVICE DESIRED
 F0FB 216EF1            LXI     H,ALT   ;START OF CONVERSION TABLE
 F0FE 110500            LXI     D,APT-ALT       ;DISTANCE BETWEEN LOGICAL CHOI
 F101 0604              MVI     B,4     ;NUMBER OF LOGICAL CHOICES
 F103 BE        AS0:    CMP     M       ;IS THIS ONE IT?
                        JRZ     AS1     ;YES, JUMP
 F104+2842
 F106 19                DAD     D       ;NO, GO TO NEXT LOGICAL ENTRY
                        DJNZ    AS0
 F107+10FA
 F109 218CF4    QPRT:   LXI     H,QMSG  ;GET ADDRESS OF QUESTION MARK MSG
 F10C CD98F6            CALL    PRTWA   ;PRINT IT
                ;
```

```
CP/M MACRO ASSEM 2.0    #005    MOSS 2.2 MONITOR

                ; THE WARM START CODE
                ;
F10F 2A0600     WINIT:  LHLD    SPSV    ;RESET THE STACK
F112 F9                 SPHL
F113 210FF1     WINITA: LXI     H,WINIT ;RESET RETURN AND WARM START VECTOR
F116 E5                 PUSH    H
F117 220100             SHLD    WSVEC+1
F11A 3EC3               MVI     A,0C3H
F11C 320000             STA     WSVEC
F11F CDA9F6             CALL    CRLF    ;START A NEW LINE
F122 CD78F3             CALL    DECHO   ;GET THE COMMAND
F125 D641               SUI     'A'     ;GET RID OF ASCII ZONE
                        JRC     QPRT    ;BAD COMMAND
F127+38E0
F129 FE1A               CPI     'Z'-'A'+1       ;CHECK UPPER LIMIT
                        JRNC    QPRT    ;BAD COMMAND
F12B+30DC
F12D 87                 ADD     A       ;DOUBLE IT FOR TABLE OFFSET
F12E 5F                 MOV     E,A     ;SET UP FOR DOUBLE ADD
F12F 1600               MVI     D,0
F131 0602               MVI     B,2     ;SET UP FOR TWO PARAMETERS
F133 2127F0             LXI     H,TBL   ;GET ACTION ROUTINE ADDRESS
F136 19                 DAD     D
F137 7E                 MOV     A,M     ;LOAD H,L INDIRECT
F138 23                 INX     H
F139 66                 MOV     H,M
F13A 6F                 MOV     L,A
F13B E9                 PCHL            ;GO TO ACTION ROUTINE
                ;
                ; FILL ACTION ROUTINE
                ;
                ;       THIS ROUTINE FILLS A BLOCK OF MEMORY WITH A USER-
                ;       DETERMINED CONSTANT.  IT EXPECTS THREE PARAMETERS
                ;       TO BE ENTERED IN THE FOLLOWING ORDER:
                ;
                ;       START ADDRESS
                ;       FINISH ADDRESS
                ;       FILL VALUE
                ;
F13C CD86F3     FILL:   CALL    EXPR3   ;GET THREE PARAMETERS
F13F 71         FIO:    MOV     M,C     ;PUT DOWN THE FILL VALUE
F140 CD8FF3             CALL    HILO    ;INCREMENT AND CHECK THE POINTER
                        JRNC    FIO     ;NOT DONE YET, JUMP
F143+30FA
F145 D1                 POP     D       ;RESTORE STACK POINTER IN CASE
                        JMPR    WINIT   ; STACK WAS OVERWRITTEN
F146+18C7
                ;
F148 50         AS1:    MOV     D,B     ;SAVE THE COUNTER RESIDUE
F149 0604               MVI     B,4     ;LOOP CONTROL
F14B CD78F3             CALL    DECHO   ;GET THE NEW ASSIGNMENT
F14E 23         AS2:    INX     H       ;INCREMENT POINTER
F14F BE                 CMP     M       ;SEE IF THIS IS IT
                        JRNZ    AS3
F150+2081
F152 68                 MOV     L,B     ;SAVE THE RESIDUE TO FORM ASGT
F153 2D                 DCR     L       ;ADJUST VALUE
F154 42                 MOV     B,D     ;REGET THE LOGICAL RESIDUE
F155 2603               MVI     H,3     ;SET UP THE IOBYTE MASK
F157 05                 DCR     B       ;ADJUST THIS ONE ALSO
                        JRZ     AS5     ;NO SHIFT NEEDED
F158+2804
F15A 29         AS4:    DAD     H       ;SHIFT THE MASKS INTO POSITION
F15B 29                 DAD     H
                        DJNZ    AS4     ;NOT DONE YET, JUMP
F15C+10FC
F15E 3A0300     AS5:    LDA     IOBYTE
F161 B4                 ORA     H       ;MASK THE DESIRED ASSIGNMENT IN
```

```
CP/M MACRO ASSEM 2.0    #006    MOSS 2.2 MONITOR

F162 AC                 XRA     H       ;LOGICAL ASGT BITS NOW OFF
F163 B5                 ORA     L       ;PUT IN NEW VALUE
F164 4F                 MOV     C,A
F165 79         IOSET:  MOV     A,C
F166 320300             STA     IOBYTE  ;SAVE NEW ASSIGNMENTS
F169 C9                 RET
F16A 3A0300     IOCHK:  LDA     IOBYTE
F16D C9                 RET
                ;
F16E 4C         ALT:    DB      'L'     ;LOGICAL LIST DEVICE TABLE
F16F 32                 DB      '2'     ;USER DEVICE #2
F170 31                 DB      '1'     ;USER DEVICE #1
F171 4C                 DB      'L'     ;LIST TO HIGH SPEED PRINTER
F172 54                 DB      'T'     ;LIST TO TTY
F173 50         APT:    DB      'P'     ;LOGICAL PUNCH DEVICE TABLE
F174 32                 DB      '2'     ;USER DEVICE #2
F175 31                 DB      '1'     ;USER DEVICE #1
F176 50                 DB      'P'     ;PUNCH TO HIGH SPEED PUNCH
F177 54                 DB      'T'     ;PUNCH TO TTY
F178 52         ART:    DB      'R'     ;LOGICAL READER DEVICE TABLE
F179 32                 DB      '2'     ;USER DEVICE #2
F17A 31                 DB      '1'     ;USER DEVICE #1
F17B 50                 DB      'P'     ;READER TO HIGH SPEED READER
F17C 54                 DB      'T'     ;READER TO TTY
F17D 43         ACT:    DB      'C'     ;LOGICAL CONSOLE DEVICE TABLE
F17E 31                 DB      '1'     ;USER DEVICE #1
F17F 42                 DB      'B'     ;CONSOLE TO BATCH (PRINTER OR PTR)
F180 43                 DB      'C'     ;CONSOLE TO CRT
F181 54                 DB      'T'     ;CONSOLE TO TTY
                ;
                ;  THE BYE ROUTINE IS USED TO PREVENT UNAUTHORIZED USAGE
                ;       OF THE SYSTEM.  THE SYSTEM LOCKS UP AND WILL NOT
                ;       RESPOND TO ANYTHING OTHER THAN TWO ASCII BELL
                ;       CHARACTERS.  WHEN IT SEES THEM CONSECUTIVELY,
                ;       CONTROL IS RETURNED TO THE MONITOR WITHOUT ALTERING
                ;       ANYTHING.
                ;
F182 0602       BYE:    MVI     B,2     ;SET UP FOR TWO CHARACTERS
F184 CD8FF6     BYE1:   CALL    CONI    ;GO READ THE CONSOLE
F187 FE07               CPI     BELL    ;SEE IF AN ASCII BELL
                        JRNZ    BYE     ;NO, START OVER AGAIN
F189+20F7
F18B CD7EF3             CALL    ECH1    ;ECHO THE BELL
                        DJNZ    BYE1    ;NOT YET, GET NEXT ONE
F18E+10F4
F190 C9                 RET             ;RETURN TO MONITOR
                ;
                ;  COMPARE ROUTINE
                ;
                ;THIS ROUTINE COMPARES TWO BLOCKS OF MEMORY AGAINST EACH
                ;       OTHER.  IF A DIFFERENCE IN THE RELATIVE ADDRESS
                ;       CONTENTS IS DETECTED, THE ADDRESS OF THE FIRST
                ;       BLOCK IS DISPLAYED, ALONG WITH ITS CONTENTS AND
                ;       THE CONTENTS OF THE OTHER BLOCK'S SAME RELATIVE
                ;       ADDRESS.
                ;
F191 CD86F3     COMP:   CALL    EXPR3   ;GO GET THREE PARAMETERS
F194 0A         CMPA:   LDAX    B       ;GET SOURCE 2 DATA
F195 C5                 PUSH    B       ;SAVE SOURCE 2 POINTER
F196 46                 MOV     B,M     ;READ SOURCE 1 DATA
F197 B8                 CMP     B       ;COMPARE DATA
                        JRZ     CMPB    ;JUMP IF OK
F198+280C
F19A F5                 PUSH    PSW     ;SAVE SOURCE 2 DATA
F19B CDFBF5             CALL    LADRB   ;WRITE THE ADDRESS
F19E 78                 MOV     A,B     ;GET SOURCE 1 DATA
F19F CDF4F5             CALL    DASH1   ;FORMAT
F1A2 F1                 POP     PSW     ;REGET SOURCE 2 DATA
```

```
CP/M MACRO ASSEM 2.0    #007    MOSS 2.2 MONITOR

 F1A3 CDE6F5            CALL    HEX1    ;OUTPUT IT
 F1A6 C1        CMPB:   POP     B
 F1A7 CD9BF3            CALL    HILOXB  ;INCREMENT SOURCE 1 POINTER AND SEE IF
                        JMPR    CMPA    ;JUMP IF NOT DONE YET
 F1AA+18E8
                ;
                ; DISPLAY ACTION ROUTINE
                ;
                ;       THIS ROUTINE DISPLAYS A BLOCK OF MEMORY ON THE
                ;       CURRENT CONSOLE DEVICE (CONSOLE DUMP).  THE USER
                ;       MUST SPECIFY THE START AND FINISH ADDRESSES.
                ;       THE DISPLAY IS ORGANIZED TO DISPLAY UP TO 16 BYTES
                ;       PER DISPLAY LINE, WITH ALL COLUMNS ALIGNED SO
                ;       EACH COLUMN HAS THE SAME LAST HEX DIGIT IN ITS ADDRESS
                ;
 F1AC CDA4F6    DISP:   CALL    EXLF    ;GO GET BLOCK LIMITS
 F1AF CDFBF5    DIS1:   CALL    LADRB   ;DISPLAY THE START ADDRESS
 F1B2 7D                MOV     A,L     ;SEE IF ON 16 BYTE BOUNDARY
 F1B3 CDF0F1            CALL    TRPLSP  ;SKIP OVER TO RIGHT COLUMN
 F1B6 E5                PUSH    H       ;SAVE (H,L)
 F1B7 7E        DIS2:   MOV     A,M     ;GET THE CONTENTS
 F1B8 CDE6F5            CALL    HEX1    ;OUTPUT IT
 F1BB CD8FF3            CALL    HILO    ;INCREMENT, CHECK POINTER
                        JRC     DIS7    ;DONE IF CARRY SET
 F1BE+382A
 F1C0 CDFEF5            CALL    BLK     ;MAKE COLUMNS
 F1C3 7D                MOV     A,L     ;READY FOR NEW LINE?
 F1C4 E60F              ANI     0FH
                        JRNZ    DIS2
 F1C6+20EF
 F1C8 E1        DIS3:   POP     H       ;REGET LINE START ADDRESS
 F1C9 7D                MOV     A,L     ;SKIP OVER TO RIGHT SPACE
 F1CA E60F              ANI     0FH
 F1CC CDF5F1            CALL    TRPL2
 F1CF 7E        DIS4:   MOV     A,M     ;GET MEMORY VALUE
 F1D0 E67F              ANI     7FH     ;STRIP OFF PARITY BIT
 F1D2 4F                MOV     C,A     ;SET UP FOR OUTPUT
 F1D3 FE20              CPI     ' '     ;SEE IF PRINTABLE IN ASCII
                        JRC     DIS5    ;JUMP IF SO
 F1D5+3804
 F1D7 FE7E              CPI     7EH
                        JRC     DIS6
 F1D9+3802
 F1DB 0E2E      DIS5:   MVI     C,'.'   ;ELSE, PRINT A DOT
 F1DD CD09F0    DIS6:   CALL    CONOUT
 F1E0 CD9CF3            CALL    HILOX   ;INCREMENT (H,L) AND SEE IF DONE
 F1E3 7D                MOV     A,L     ;NOT DONE, READY FOR NEW LINE?
 F1E4 E60F              ANI     0FH
                        JRNZ    DIS4    ;JUMP IF NOT
 F1E6+20E7
                        JMPR    DIS1    ;DO THE NEXT LINE
 F1E8+18C5
 F1EA 93        DIS7:   SUB     E       ;SKIP OVER TO START ASCII PRINTOUT
 F1EB CDF0F1            CALL    TRPLSP
                        JMPR    DIS3    ;GO PRINT THE ASCII
 F1EE+18D8
                ;
 F1F0 E60F      TRPLSP: ANI     0FH     ;ISOLATE THE LOW FOUR BITS
 F1F2 47                MOV     B,A     ;PREPARE TO SPACE OVER TO RIGHT COLUMN
 F1F3 87                ADD     A       ;TRIPLE THE COUNT
 F1F4 80                ADD     B
 F1F5 47        TRPL2:  MOV     B,A     ;PUT BACK INTO B
 F1F6 04                INR     B       ;ADJUST COUNTER
 F1F7 CDFEF5    TRPL1:  CALL    BLK     ;DO THE SPACING
                        DJNZ    TRPL1   ;NO, DO ANOTHER COLUMN
 F1FA+10FB
 F1FC C9                RET
                ;
```

```
CP/M MACRO ASSEM 2.0    #008    MOSS 2.2 MONITOR

                ; GO TO ACTION ROUTINE
                ;
                ;
                ; GOTO COMMAND TRANSFERS CONTROL TO A SPECIFIED ADDRESS.
                ;   IT ALLOWS THE SELECTIVE SETTING OF UP TO TWO BREAKPOINTS
                ;   AS WELL AS ALLOWING ANY CONSOLE INPUT TO BREAKPOINT
                ;   THE RUN, AS LONG AS INTERRUPT 1 IS ACTIVE.
                ;
F1FD CDC0F3     GOTO:   CALL    PCHK    ;SEE IF OLD ADDRESS WANTED
                        JRC     GO3     ;  YES, JUMP
F200+3837
                        JRZ     GO0     ;  YES, BUT SET SOME BREAKPOINTS
F202+2810
F204 CDCCF0             CALL    EXF     ;GET NEW GOTO ADDRESS
F207 D1                 POP     D
F208 213400             LXI     H,PLOC  ;PUT ADDRESS IN PC LOCATION
F20B 39                 DAD     SP
F20C 72                 MOV     M,D     ;LOW BYTE
F20D 2B                 DCX     H
F20E 73                 MOV     M,E     ;HIGH BYTE
F20F 79                 MOV     A,C
F210 FE0D               CPI     CR      ;SEE IF A CR WAS LAST ENTERED
                        JRZ     GO3
F212+2825
F214 0602       GO0:    MVI     B,NBKPTS
F216 213500             LXI     H,TLOC  ;POINT TO TRAP STORAGE
F219 39                 DAD     SP
F21A C5         GO1:    PUSH    B       ;SAVE NUMBER OF BREAKPOINTS
F21B E5                 PUSH    H       ;SAVE STORAGE POINTER
F21C 0602               MVI     B,2     ;SET UP TO GET A TRAP ADDRESS
F21E CDD7F0             CALL    EXPR1   ;GET A TRAP ADDRESS
F221 D1                 POP     D       ;GET THE TRAP ADDRESS INTO (D,E)
F222 E1                 POP     H       ;REGET THE STORAGE ADDRESS
F223 7A                 MOV     A,D     ;INSURE THE TRAP ADDRESS ISN'T ZERO
F224 B3                 ORA     E
                        JRZ     GO2     ;JUMP IF SO
F225+280A
F227 73                 MOV     M,E     ;SAVE THE BREAKPOINT ADDRESS
F228 23                 INX     H
F229 72                 MOV     M,D
F22A 23                 INX     H
F22B 1A                 LDAX    D       ;SAVE THE INSTRUCTION FROM THE BP ADDR
F22C 77                 MOV     M,A
F22D 23                 INX     H
F22E 3ECF               MVI     A,RST OR 8      ;INSERT THE BREAKPOINT
F230 12                 STAX    D
F231 79         GO2:    MOV     A,C     ;REGET THE DELIMITER TO SEE
F232 FE0D               CPI     CR      ;  IF WE ARE DONE SETTING BREAKPOINTS
F234 C1                 POP     B       ;  UNLOAD THE STACK FIRST
                        JRZ     GO3     ;YES, JUMP
F235+2802
                        DJNZ    GO1     ;JUMP IF NOT AT BP LIMIT
F237+10E1
F239 CDA9F6     GO3:    CALL    CRLF
F23C E1                 POP     H       ;GET RID OF STACK JUNK
F23D 2143F4             LXI     H,RS9
F240 E5                 PUSH    H
F241 21CFF3             LXI     H,REST
F244 220900             SHLD    9       ;SET BREAKPOINT JUMP VECTOR ADDRESS
F247 211800             LXI     H,24    ;FIND REGISTER SET ROUTINE ADDRESS
F24A 39                 DAD     SP
F24B D1                 POP     D       ;ADJUST THE STACK
F24C E9                 PCHL            ;GO TO THE DESIRED PLACE
                ;
                ; GENERAL PURPOSE INPUT/OUTPUT ROUTINES
                ;
                ;THESE ROUTINES ALLOW BYTE-BY-BYTE INPUT OR OUTPUT FROM
                ;       THE CURRENT CONSOLE DEVICE.  THEY ARE INVOKED BY
```

```
CP/M MACRO ASSEM 2.0    #009    MOSS 2.2 MONITOR

                ;       THE MONITOR "I" OR "O" COMMAND.
                ;
F24D CDD7F0     INPT:   CALL    EXPR1   ;GET INPUT PORT NUMBER
F250 C1                 POP     B       ;GET PORT # INTO C REGISTER
                        INP     E       ;READ VALUE INTO E REGISTER
F251+ED58
                        JMPR    BITS2   ;GO DO A BINARY PRINT OF THE VALUE
F253+1851
                ;
F255 CDD9F0     OUPT:   CALL    EXPR    ;GET THE ADDRESS AND DATA FOR OUTPUT
F258 D1                 POP     D       ;DATA VALUE INTO E
F259 C1                 POP     B       ;PORT INTO C
                        OUTP    E       ;DO THE OUTPUT
F25A+ED59
F25C C9                 RET
                ;
                ;  MOVE ROUTINE
                ;
                ;       THIS ROUTINE EXPECTS THREE PARAMETERS, ENTERED IN THE
                ;       SOURCE FIRST BYTE ADDRESS
                ;       SOURCE LAST BYTE ADDRESS
                ;       DESTINATION FIRST BYTE ADDRESS
                ;
F25D CD86F3     MOVE:   CALL    EXPR3   ;GET THREE PARAMETERS
F260 7E         MOV1:   MOV     A,M     ;GET NEXT BYTE
F261 02                 STAX    B       ;MOVE IT
F262 CD9BF3             CALL    HILOXB  ;GO INCREMENT, CHECK SOURCE POINTER
                        JMPR    MOV1    ;NOT THERE YET, GO DO IT AGAIN
F265+18F9
                ;
                ; SUBSTITUTE ACTION ROUTINE
                ;
                ;THIS ROUTINE ALLOWS THE USER TO INSPECT ANY MEMORY LOCATION
                ;       AND ALTER THE CONTENTS, IF DESIRED AND IF THE ADDRESS
                ;       IS IN RAM.  THE CONTENTS MAY BE LEFT UNALTERED
                ;       BY ENTERING A SPACE, COMMA, OR A CARRIAGE RETURN.  IF
                ;       A CARRIAGE RETURN IS ENTERED, THE ROUTINE IS TERMINATE
                ;       IF A SPACE OR COMMA IS ENTERED, THE ROUTINE
                ;       PROCEEDS TO THE NEXT LOCATION AND PRESENTS THE USER
                ;       WITH AN OPPORTUNITY TO ALTER IT.
                ;
F267 CDD7F0     SUBS:   CALL    EXPR1   ;GO GET ONE PARAMETER
F26A E1                 POP     H       ;GET THE START ADDRESS
F26B 7E         SUB1:   MOV     A,M     ;GET THE CONTENTS OF THE ADDRESS
F26C CDF4F5             CALL    DASH1   ;DISPLAY IT ON CONSOLE AND A DASH
F26F CDC0F3             CALL    PCHK    ;GET, CHECK CHARACTER
F272 D8                 RC              ;DONE IF CARRIAGE RETURN
                        JRZ     SUB2    ;NO CHANGE IF BLANK OR ,
F273+280F
F275 FE0A               CPI     LF      ;SEE IF PREVIOUS BYTE WANTED
                        JRZ     SUB3    ;YES, DO IT
F277+280D
F279 E5                 PUSH    H       ;SAVE MEMORY POINTER
F27A CDCCF0             CALL    EXF     ;GO GET REST OF NEW VALUE
F27D D1                 POP     D       ;NEW VALUE TO E REGISTER
F27E E1                 POP     H       ;RESTORE MEMORY POINTER
F27F 73                 MOV     M,E     ;PUT DOWN NEW VALUE
F280 79                 MOV     A,C     ;GET THE DELIMITER
F281 FE0D               CPI     CR      ;SEE IF DONE (CARRIAGE RETURN)
F283 C8                 RZ              ;YES, RETURN TO MONITOR
F284 23         SUB2:   INX     H       ;NO, INCREMENT MEMORY POINTER
F285 23                 INX     H       ;ALLOW A FALL-THROUGH ON THE NEXT INST
F286 2B         SUB3:   DCX     H       ;ADJUST (H,L) AS APPROPRIATE
F287 7D                 MOV     A,L     ;GET LO ADDRESS BYTE
F288 E607               ANI     7       ;SEE IF ON A BOUNDARY
F28A CCFBF5             CZ      LADRB   ;CALL IF ON THE BOUNDARY
                        JMPR    SUB1    ;GO DO THE NEXT LOCATION
F28D+18DC
```

```
CP/M MACRO ASSEM 2.0    #010    MOSS 2.2 MONITOR

                ;
                ; MTEST ROUTINE TESTS A SPECIFIED BLOCK OF MEMORY TO
                ;   SEE IF ANY HARD DATA BIT FAILURES EXIST.  IT IS
                ;   NOT AN EXHAUSTIVE TEST, BUT JUST A QUICK INDICATION
                ;   OF THE MEMORY'S OPERATIVENESS.
                ;
F28F CDA4F6     MTEST:  CALL    EXLF
F292 7E         MTEST1: MOV     A,M         ;READ A BYTE
F293 F5                 PUSH    PSW         ;SAVE IT
F294 2F                 CMA                 ;COMPLEMENT IT
F295 77                 MOV     M,A         ;WRITE IT
F296 AE                 XRA     M           ;RESULT SHOULD BE ZERO
F297 C4A1F2             CNZ     BITS        ;LOG ERROR IF NOT
F29A F1         MTEST2: POP     PSW         ;RESTORE ORIGINAL BYTE
F29B 77                 MOV     M,A
F29C CD9CF3             CALL    HILOX       ;POINT TO NEXT AND SEE IF DONE
                        JMPR    MTEST1      ;NO, CONTINUE
F29F+18F1
                ;
F2A1 D5         BITS:   PUSH    D           ;SAVE (D,E)
F2A2 5F                 MOV     E,A         ;SAVE ERROR PATTERN IN E
F2A3 CDFBF5             CALL    LADRB       ;FIRST PRINT THE ADDRESS
F2A6 0608       BITS2:  MVI     B,8         ;LOOP CONTROL FOR 8 BITS
F2A8 7B         BITS1:  MOV     A,E         ;GET NEXT BIT
F2A9 07                 RLC                 ;  INTO CARRY
F2AA 5F                 MOV     E,A         ;SAVE REST
F2AB 3E18               MVI     A,'0'/2     ;BUILD ASCII 1 OR 0
F2AD 17                 RAL                 ;  CARRY DETERMINES WHICH
F2AE 4F                 MOV     C,A         ;NOW, OUTPUT IT
F2AF CD09F0             CALL    CONOUT
                        DJNZ    BITS1       ;DO IT AGAIN
F2B2+10F4
F2B4 D1                 POP     D
F2B5 C9                 RET
                ;
                ; EXAMINE REGISTERS COMMAND INSPECTS THE VALUES OF THE
                ;   THE REGISTERS STORED BY THE LAST ENCOUNTERED BREAKPOINT.
                ;   THE VALUES MAY BE MODIFIED IF DESIRED.
                ;
F2B6 23         XAA:    INX     H           ;SKIP OVER TO NEXT ENTRY
F2B7 23                 INX     H
F2B8 34         XA:     INR     M           ;SEE IF AT END OF TABLE
F2B9 C8                 RZ                  ;COULDN'T FIND MATCH, QUIT
F2BA F2C1F2             JP      XAB         ;SORT OUT BIT 7 OF TABLE
F2BD F680               ORI     80H         ;SET IT ON TEST VALUE
                        JMPR    XAC
F2BF+1802
F2C1 E67F       XAB:    ANI     7FH         ;RESET BIT 7
F2C3 35         XAC:    DCR     M           ;TO BE PULLED OUT IN ROM
F2C4 BE                 CMP     M           ;SEE IF THIS IS IT
                        JRNZ    XAA         ;NO, GO TRY AGAIN
F2C5+20EF
F2C7 CDFEF5             CALL    BLK         ;YES, PREPARE TO SHOW CURRENT VALUE
F2CA CD15F3             CALL    PRTVAL      ;GO PRINT THE VALUE
F2CD CDF7F5             CALL    DASH        ;PROMPT A NEW VALUE
F2D0 CDC0F3             CALL    PCHK        ;GET THE INPUT
F2D3 D8                 RC                  ;DONE IF CARRIAGE RETURN
                        JRZ     XF          ;JUMP IF NO CHANGE DESIRED
F2D4+2812
F2D6 E5                 PUSH    H           ;TO BE CHANGED, SAVE POINTER
F2D7 CDCCF0             CALL    EXF         ;GET THE NEW VALUE
F2DA E1                 POP     H           ;  INTO (H,L)
F2DB 7D                 MOV     A,L         ;GET THE NEW LOW BYTE
F2DC 13                 INX     D           ;ADJUST POINTER
F2DD 12                 STAX    D           ;PUT IT DOWN
F2DE E3                 XTHL                ;RECOVER THE TABLE POINTER
F2DF 7E                 MOV     A,M         ;GET THE ATTRIBUTES
F2E0 E3                 XTHL                ;SET THE STACK STRAIGHT
```

```
CP/M MACRO ASSEM 2.0    #011    MOSS 2.2 MONITOR

 F2E1 07                RLC             ;SEE IF 8 BIT REGISTER
                        JRNC    XE      ;JUMP IF SO
 F2E2+3003
 F2E4 13                INX     D       ;REGISTER PAIR, DO OTHER 8 BITS
 F2E5 7C                MOV     A,H
 F2E6 12                STAX    D
 F2E7 E1        XE:     POP     H       ;RESTORE THE TABLE POINTER
 F2E8 79        XF:     MOV     A,C     ;SEE IF IT WAS A CR
 F2E9 FE0D              CPI     CR
 F2EB C8                RZ              ;DONE IF SO
 F2EC 213DF3    XMNE:   LXI     H,ACTBL ;GET ADDRESS OF REGISTER LOOK-UP TABLE
 F2EF CDC0F3    XMNE1:  CALL    PCHK    ;FIND OUT WHAT ACTION IS WANTED
                        JRC     XG      ;SHOW ALL IF CARRIAGE RETURN
 F2F2+380B
                        JRZ     XMNE1   ;IGNORE BLANKS OR COMMAS
 F2F4+28F9
 F2F6 FE27              CPI     ''''    ;SEE IF PRIMES WANTED
                        JRNZ    XA      ;NO, MUST BE SINGLE REGISTER
 F2F8+20BE
 F2FA 2155F3            LXI     H,PRMTB ;YES, SET TABLE ADDRESS
                        JMPR    XMNE1   ;  AND FIND OUT WHICH ONE
 F2FD+18F0
                ;
 F2FF 7E        XG:     MOV     A,M
 F300 4F                MOV     C,A
 F301 3C                INR     A       ;SEE IF AT END OF TABLE
 F302 C8                RZ              ;DONE IF SO
 F303 FCA9F6            CM      CRLF    ;START A NEW LINE IF BIT 7 IS SET
 F306 CD09F0            CALL    CONOUT
 F309 CDF7F5            CALL    DASH    ;PROMPT FOR A NEW VALUE
 F30C CD15F3            CALL    PRTVAL  ;GO PRINT THE VALUE
 F30F CDFEF5            CALL    BLK     ;FORMATTER
 F312 23                INX     H       ;POINT TO NEXT ENTRY
                        JMPR    XG      ;DO THE NEXT VALUE
 F313+18EA
                ;
 F315 23        PRTVAL: INX     H       ;POINT TO NEXT ENTRY
 F316 7E                MOV     A,M     ;GET OFFSET AND ATTRIBUTES BYTE
 F317 E63F              ANI     3FH     ;ISOLATE THE OFFSET
 F319 C602              ADI     2       ;ALLOW FOR RETURN ADDRESS
 F31B EB                XCHG            ;SWAP POINTERS
 F31C 6F                MOV     L,A     ;BUILD THE ADDRESS OF THE REG CONTENTS
 F31D 2600              MVI     H,0
 F31F 39                DAD     SP
 F320 EB                XCHG            ;RE-SWAP THE POINTERS
 F321 7E                MOV     A,M     ;NOW FIND OUT ATTRIBUTES
 F322 0601              MVI     B,1     ;SET UP FOR SINGLE REG VALUE
 F324 07                RLC
                        JRNC    PV1     ;JUMP IF SINGLE REGISTER VALUE WANTED
 F325+300E
 F327 04                INR     B       ;SET UP FOR REGISTER PAIR
 F328 07                RLC
                        JRNC    PV1     ;JUMP IF REGISTER PAIR IS NEXT
 F329+300A
 F32B E5                PUSH    H       ;SPECIAL CASE FOR MEMORY REGISTER
 F32C 1A                LDAX    D       ;BUILD ADDRESS IN (H,L)
 F32D 67                MOV     H,A
 F32E 1B                DCX     D
 F32F 1A                LDAX    D
 F330 6F                MOV     L,A
 F331 7E                MOV     A,M     ;GET THE MEMORY VALUE
 F332 E1                POP     H       ;RESTORE (H,L)
                        DJNZ    PV2     ;ALWAYS JUMP
 F333+1001
 F335 1A        PV1:    LDAX    D       ;GET THE REGISTER CONTENTS
 F336 CDE6F5    PV2:    CALL    HEX1    ;OUTPUT THE VALUE
 F339 1B                DCX     D       ;ADJUST THE MEMORY POINTER
                        DJNZ    PV1
```

```
CP/M MACRO ASSEM 2.0    #012    MOSS 2.2 MONITOR

 F33A+10F9
 F33C C9                RET
                        ;
 F33D C115      ACTBL:  DB      80H+'A',ALOC
 F33F 4213              DB      'B',BLOC
 F341 4312              DB      'C',CLOC
 F343 4411              DB      'D',DLOC
 F345 4510              DB      'E',ELOC
 F347 4614              DB      'F',FLOC
 F349 4831              DB      'H',HLOC
 F34B 4C30              DB      'L',LLOC
 F34D CDF1              DB      80H+'M',HLOC+0C0H
 F34F 50B4              DB      'P',PLOC+80H
 F351 5397              DB      'S',SLOC+80H
 F353 4903              DB      'I',ILOC
                        ;
                        ; REST OF Z-80 REGISTER OFFSETS
                        ;
 F355 C109      PRMTB:  DB      80H+'A',APLOC
 F357 420B              DB      'B',BPLOC
 F359 430A              DB      'C',CPLOC
 F35B 440D              DB      'D',DPLOC
 F35D 450C              DB      'E',EPLOC
 F35F 4608              DB      'F',FPLOC
 F361 480F              DB      'H',HPLOC
 F363 4C0E              DB      'L',LPLOC
 F365 CDCF              DB      80H+'M',HPLOC+0C0H
 F367 5887              DB      'X',XLOC+80H
 F369 5985              DB      'Y',YLOC+80H
 F36B 5202              DB      'R',RLOC
 F36D FF                DB      0FFH
                        ;
                        ; GENERAL PURPOSE ROUTINES
                        ;
                        ;
                        ; ROUTINE CONV CONVERTS THE LOW ORDER NIBBLE OF THE
                        ;       ACCUMULATOR TO ITS ASCII EQUIVALENT.  IT
                        ;       PUTS THE RESULT INTO C FOR LATER OUTPUT.
                        ;
 F36E E60F      CONV:   ANI     0FH     ;STRIP OFF BITS 4-7
 F370 C690              ADI     90H     ;PUT ON THE ASCII ZONE
 F372 27                DAA
 F373 CE40              ACI     40H
 F375 27                DAA
 F376 4F                MOV     C,A     ;PUT IN OUTPUT PASS REGISTER
 F377 C9                RET
                        ;
                        ; ROUTINE ECHO READS A BYTE FROM A HALF-DUPLEX CONSOLE
                        ;       DEVICE, THEN ECHOES THE CHARACTER BACK TO THE
                        ;       CONSOLE.
                        ;
 F378 CDF7F5    DECHO:  CALL    DASH    ;PRINT A DASH
 F37B CD8FF6    ECHO:   CALL    CONI    ;CONSOLE READ, WRITE ROUTINE
 F37E C5        ECH1:   PUSH    B       ;  SAVE (B,C)
 F37F 4F                MOV     C,A     ;  PASS CHARACTER IN C REGISTER
 F380 CD09F0            CALL    CONOUT  ;  OUTPUT IT
 F383 79                MOV     A,C     ;  PUT CHARACTER BACK INTO A
 F384 C1                POP     B       ;  RESTORE (B,C)
 F385 C9                RET
                        ;
                        ; ROUTINE EXPR3 GETS THREE PARAMETERS, DOES A CR, LF AND
                        ;     THEN LOADS (B,C), (D,E), AND (H,L) WITH THE PARAMETERS.
                        ;
 F386 04        EXPR3:  INR     B       ;2 IS ALREADY IN THE B REGISTER
 F387 CDD9F0            CALL    EXPR    ;GET THE PARAMETERS
 F38A C1                POP     B       ;PUT PARAMETERS INTO REGISTERS
 F38B D1                POP     D
 F38C C3AAF6            JMP     CRLFA   ;GO DO THE CARRIAGE RETURN SEQUENCE
```

```
                ;
                ; ROUTINE HILO INCREMENTS (H,L).  IT THEN CHECKS FOR (AND
                ;       DISALLOWS) A WRAP-AROUND SITUATION.  IF IT OCCURS,
                ;       THE CARRY BIT WILL BE SET ON RETURN.  IF NO WRAP-
                ;       AROUND OCCURRED, (H,L) IS COMPARED TO (D,E) AND
                ;       THE FLAG BITS SET ACCORDINGLY.
                ;
F38F 23         HILO:   INX     H       ;INCREMENT (H,L)
F390 7C                 MOV     A,H     ;TEST IF ZERO
F391 B5                 ORA     L       ;  IN (H,L)
F392 37                 STC             ;SET CARRY FOR (H,L)=0
F393 C8                 RZ              ;RETURN IF (H,L) = 0
F394 7B                 MOV     A,E     ;COMPARE (H,L) TO (D,E)
F395 95                 SUB     L
F396 7A                 MOV     A,D
F397 9C                 SBB     H
F398 C9                 RET             ;RETURN WITH FLAGS SET
                ;
                ; ROUTINE HILOX INCREMENTS (H,L), COMPARES IT TO (D,E) AND
                ;   IF EQUAL, RETURNS CONTROL TO THE MONITOR EXECUTIVE.
                ;   OTHERWISE, CONTROL RETURNS TO THE CALLING ROUTINE.
                ;
F399 D1         HILOD:  POP     D       ;GET RID OF RETURN ADDRESS
F39A C9                 RET             ;RETURN TO MONITOR
F39B 03         HILOXB: INX     B       ;INCREMENT (B,C)
F39C CD8FF3     HILOX:  CALL    HILO    ;INC AND CHECK (H,L)

                        JRC     HILOD   ;DONE IF CARRY SET

F39F+38F8
F3A1 CD12F0             CALL    CONST   ;SEE IF CONSOLE BREAK PENDING
F3A4 B7                 ORA     A
F3A5 C8                 RZ              ;NONE, RETURN TO CONTINUE
F3A6 CD8FF6             CALL    CONI    ;SEE IF WAIT OR BREAK
F3A9 FE13               CPI     CTRLS
                        JRNZ    HILOD   ;JUMP IF BREAK

F3AB+20EC
F3AD C38FF6             JMP     CONI    ;GO WAIT FOR NEXT CHARACTER
                ;
                ; ROUTINE NIBBLE CONVERTS THE ASCII CHARACTERS 0-9 AND
                ;       A-F TO THEIR EQUIVALENT HEXADECIMAL VALUE.  IF
                ;       THE CHARACTER IS NOT IN RANGE, THE CARRY BIT IS SET TO
                ;       FLAG THE ERROR.
                ;
F3B0 D630       NIBBLE: SUI     '0'     ;ASCII TO HEX CONVERSION
F3B2 D8                 RC              ;  DONE IF OUT OF RANGE
F3B3 FE17               CPI     'G'-'0' ;CHECK UPPER END
F3B5 3F                 CMC             ;  TOGGLE THE CARRY BIT
F3B6 D8                 RC              ;  DONE IF OUT OF RANGE
F3B7 FE0A               CPI     '9'-'0'+1  ;SEE IF NUMERIC
F3B9 3F                 CMC             ;  TOGGLE THE CARRY BIT
F3BA D0                 RNC             ;  DONE IF SO
F3BB D607               SUI     'A'-'9'-1  ;SUBTRACT THE ALPHA BIAS
F3BD FE0A               CPI     10      ;  SET CARRY FOR INVALID CHAR
F3BF C9                 RET
                ;
                ; ROUTINE PCHK READS A CHARACTER FROM THE CONSOLE, THEN
                ;       CHECKS IT FOR A DELIMITER.  IF IT IS NOT
                ;       A DELIMITER, A NON-ZERO CONDITION IS RETURNED.
                ;       IF IT IS A DELIMITER, A ZERO CONDITION IS RETURNED.
                ;       FURTHER, IF THE DELIMITER IS A CARRIAGE RETURN,
                ;       THE CARRY BIT IS SET.  A BLANK OR A COMMA RESETS
                ;       THE CARRY BIT.
                ;
F3C0 CD7BF3     PCHK:   CALL    ECHO    ;GET, TEST FOR DELIMITER
F3C3 FE20       P2C:    CPI     ' '     ;  BLANK?
F3C5 C8                 RZ              ;  YES, DONE
F3C6 FE2C               CPI     ','     ;  NO, COMMA?
F3C8 C8                 RZ              ;  YES, DONE
```

```
CP/M MACRO ASSEM 2.0     #014     MOSS 2.2 MONITOR

 F3C9 FE0D                CPI     CR       ;  NO, CARRIAGE RETURN?
 F3CB 37                  STC              ;  SHOW IT IN CARRY BIT
 F3CC C8                  RZ               ;  DONE IF CR
 F3CD 3F                  CMC              ;CLEAR CARRY FOR NO DELIMITER
 F3CE C9                  RET

                  ;
                  ; ROUTINE REST TRAPS ALL OF THE REGISTER CONTENTS WHENEVER A
                  ;       RESTART 1 INSTRUCTION IS EXECUTED.  THE TRAPPED CONTEN
                  ;       ARE STORED IN THE SYSTEM STACK AREA FOR LATER ACCESS A
                  ;       USE BY THE GOTO AND THE EXAMINE REGISTERS COMMANDS.
                  ;
                  ; INSERT INTERRUPT DISABLER SOFTWARE AT START OF REST:
 F3CF E5          REST:   PUSH    H        ;SAVE ALL THE REGISTERS
 F3D0 D5                  PUSH    D
 F3D1 C5                  PUSH    B
 F3D2 F5                  PUSH    PSW
 F3D3 CD6FF0              CALL    MEMSIZ   ;GET THE MONITOR'S STACK LOCATION
 F3D6 EB                  XCHG
 F3D7 210A00              LXI     H,10     ;GO UP 10 BYTES IN THE STACK
 F3DA 39                  DAD     SP       ;  TO SKIP OVER TEMP REGISTER SAVE
 F3DB 0604                MVI     B,4      ;PICK OFF THE REGISTER VALUES
 F3DD EB                  XCHG
 F3DE 2B          RS1:    DCX     H
 F3DF 72                  MOV     M,D      ;SAVE IN WORK AREA
 F3E0 2B                  DCX     H
 F3E1 73                  MOV     M,E
 F3E2 D1                  POP     D
                  DJNZ    RS1

 F3E3+10F9
 F3E5 C1                  POP     B        ;GET THE BREAKPOINT LOCATION
 F3E6 0B                  DCX     B
 F3E7 F9                  SPHL             ;SET THE MONITOR STACK
 F3E8 212500              LXI     H,TLOCX  ;SET UP TO RESTORE BREAKPOINTS
 F3EB 39                  DAD     SP
 F3EC D5                  PUSH    D
 F3ED 1602                MVI     D,NBKPTS ;LOOP CONTROL FOR N BREAKPOINTS
 F3EF 7E          RS2:    MOV     A,M
 F3F0 91                  SUB     C        ;SEE IF A SOFTWARE TRAP
 F3F1 23                  INX     H
 F3F2 7E                  MOV     A,M
 F3F3 98                  SBB     B        ;MAYBE, TRY REST OF ADDRESS
                  JRZ     RS5      ;FOUND ONE, JUMP TO RESET IT

 F3F4+2806
 F3F6 23          RS3:    INX     H        ;NOT FOUND, TRY NEXT ONE
 F3F7 23                  INX     H
 F3F8 15                  DCR     D
                  JRNZ    RS2

 F3F9+20F4
 F3FB 03          RS4:    INX     B        ;NONE FOUND
 F3FC 212000      RS5:    LXI     H,LLOCX
 F3FF D1                  POP     D
 F400 39                  DAD     SP
 F401 73                  MOV     M,E      ;STORE USER (H,L)
 F402 23                  INX     H
 F403 72                  MOV     M,D
 F404 C5                  PUSH    B        ;SAVE (B,C)
 F405 0E2A                MVI     C,'*'    ;TYPE THE BREAK INDICATION
 F407 CD09F0              CALL    CONOUT
 F40A D1                  POP     D        ;REGET THE BREAKPOINT LOCATION
 F40B 3EF4                MVI     A,RS9/256
 F40D BA                  CMP     D        ;SEE IF A RET BREAKPOINT
                  JRZ     RS6

 F40E+2809
 F410 23                  INX     H
 F411 23                  INX     H
 F412 73                  MOV     M,E      ;RESTORE USER PROGRAM COUNTER
 F413 23                  INX     H
 F414 72                  MOV     M,D
```

```
CP/M MACRO ASSEM 2.0    #015    MOSS 2.2 MONITOR

F415 EB                 XCHG            ;PRINT THE BREAKPOINT LOCATION
F416 CDE1F5             CALL    LADR
F419 212500     RS6:    LXI     H,TLOCX
F41C 39                 DAD     SP
F41D 010002             LXI     B,NBKPTS*256
F420 5E         RS7:    MOV     E,M     ;RESTORE BREAKPOINTED LOCATIONS
F421 71                 MOV     M,C     ;RESET SYSTEM BP SAVE AREA
F422 23                 INX     H
F423 56                 MOV     D,M
F424 71                 MOV     M,C
F425 23                 INX     H
F426 7B                 MOV     A,E
F427 B2                 ORA     D
                        JRZ     RS8     ;DO NOTHING IF ZERO

F428+2802
F42A 7E                 MOV     A,M
F42B 12                 STAX    D
F42C 23         RS8:    INX     H       ;SAME THING FOR OTHER
                        DJNZ    RS7     ; BREAKPOINT

F42D+10F1
                        EXAF            ;NOW SAVE THE Z-80 UNIQUES

F42F+08
                        EXX

F430+D9
F431 E5                 PUSH    H
F432 D5                 PUSH    D
F433 C5                 PUSH    B
F434 F5                 PUSH    PSW
                        PUSHIX

F435+DDE5
                        PUSHIY

F437+FDE5
                        LDAI

F439+ED57
F43B 47                 MOV     B,A
                        LDAR

F43C+ED5F
F43E 4F                 MOV     C,A
F43F C5                 PUSH    B
F440 C313F1             JMP     WINITA  ;RETURN TO MONITOR
F443 E5         RS9:    PUSH    H       ;RET BREAKPOINT ENCOUNTERED, ADJUST TH
F444 CF                 RST     1       ;DO THE BREAKPOINT
                ;
F445 C1         EXIT:   POP     B
F446 79                 MOV     A,C
                        STAR

F447+ED4F
F449 78                 MOV     A,B
                        STAI

F44A+ED47
                        POPIX

F44C+DDE1
                        POPIY

F44E+FDE1
F450 F1                 POP     PSW
F451 C1                 POP     B
F452 D1                 POP     D
F453 E1                 POP     H
                        EXAF

F454+08
                        EXX

F455+D9
F456 D1                 POP     D
F457 C1                 POP     B
F458 F1                 POP     PSW
F459 E1                 POP     H
F45A F9                 SPHL
F45B 00                 DB      0       ;PLACE FOR EI
```

```
CP/M MACRO ASSEM 2.0    #016    MOSS 2.2 MONITOR

F45C 210000             LXI     H,0
F45F C30000             JMP     0
F462 =          ENDX:   EQU     $
                ;
                ; ERROR HANDLERS
                ;
                ;       THREE TYPES OF ERRORS ARE DETECTED:  A RESTART
                ;       ERROR; AN I/O ASSIGNMENT ERROR; AND CERTAIN PROGRAM
                ;       ERRORS (DETERMINED BY THE PARTICULAR ROUTINE WHERE
                ;       THE ERROR CONDITION WAS ENCOUNTERED.)  EACH CAUSES
                ;       A UNIQUE MESSAGE TO BE PRINTED, THEN DOES A WARM
                ;       INITIALIZATION OF THE MONITOR.  THE I/O ERROR
                ;       CAUSES THE I/O ASSIGNMENTS TO BE RESET TO DEFAULT ASSI
                ;
F462 AF         IOER:   XRA     A       ;SET IOBYTE TO DEFAULT VALUE
F463 320300             STA     IOBYTE
F466 216CF4             LXI     H,IOMSG ;GET ADDRESS OF I/O ERROR MSG
F469 C3B5F6             JMP     COMERR  ;GO PROCESS IT
F46C 492F4F2045IOMSG:   DB      'I/O ER','R'+80H
                ;
                ; BYTE ROUTINE READS TWO ASCII CHARACTERS FROM THE
                ;   CURRENT PAPER TAPE READER AND ASSEMBLES THEM INTO TWO
                ;   HEXADECIMAL BYTES OF DATA.  IT UPDATES A CHECKSUM
                ;   ACCUMULATED IN REGISTER D.
                ;
F473 CDE8F6     BYTE:   CALL    BYT     ;GET NEXT BYTE
F476 B0                 ORA     B       ;COMBINE THEM
F477 47                 MOV     B,A
F478 82                 ADD     D       ;UPDATE CHECKSUM
F479 57                 MOV     D,A
F47A 78                 MOV     A,B     ;RESTORE BYTE
F47B C9                 RET
                ;
F47C 0E0D       PEOL:   MVI     C,CR
F47E CD7CF6             CALL    PO
F481 0E0A               MVI     C,LF
F483 C37CF6             JMP     PO      ;GO PUNCH THE OUTPUT
                ;
                ; RIX ROUTINE READS ONE CHARACTER FROM THE CURRENT
                ;   PAPER TAPE READER AND STRIPS OFF THE PARITY BIT.
                ;
F486 CD56F6     RIX:    CALL    RI
F489 E67F               ANI     7FH
F48B C9                 RET
                ;
F48C 3F3F3FBF   QMSG:   DB      '???','?'+80H
F490 4D4F535320LOGMSG:  DB      'MOSS VERS 2.2'
F49D 0D8A               DB      CR,LF+80H
                ;
                ; INITIALIZATION CODE FOR THE 8250 ASYNCHRONOUS COMMUNICATION
                ;   ELEMENT.  THIS CODE WILL INITIALIZE THE BAUD RATE OF THE
                ;   8250, AS WELL AS THE WORD FORMAT.  8 DATA BITS, 1 STOP BIT
                ;   AND NO PARITY ARE SELECTED.  EITHER 2 OR 3 CARRIAGE RETURN
                ;   MUST BE ENTERED TO ESTABLISH THE CORRECT BAUD RATE.
                ;
F49F 3E0F       I8250:  MVI     A,0FH   ;SET UP THE 8250
F4A1 D324               OUT     SMDMCT
F4A3 114000             LXI     D,40H   ;SET UP TO TIME THE START BIT
F4A6 62                 MOV     H,D
F4A7 6A                 MOV     L,D     ;ZEROES TO (H,L)
F4A8 DB26       I8250A: IN      SMDMST  ;WAIT FOR START BIT
F4AA A3                 ANA     E
                        JRZ     I8250A
F4AB+28FB
F4AD DB26       I8250B: IN      SMDMST  ;NOW, TIME THE START BIT DURATION
F4AF 23                 INX     H
F4B0 A3                 ANA     E
F4B1 A3                 ANA     E
```

```
CP/M MACRO ASSEM 2.0    #017    MOSS 2.2 MONITOR

 F4B2 C2ADF4              JNZ     I8250B
 F4B5 E5                  PUSH    H       ;SAVE COUNT IN CASE OF 4 MHZ
 F4B6 29                  DAD     H       ;PREPARE THE 2 MHZ DIVISOR
 F4B7 5C                  MOV     E,H     ;SET UP THE FUDGE FACTOR
 F4B8 19                  DAD     D       ;APPLY THE FUDGE FACTOR
 F4B9 19                  DAD     D
 F4BA E5                  PUSH    H       ;SAVE FOR LATER USE
 F4BB 29                  DAD     H       ;WAIT FOR 8 BIT TIMES
 F4BC 29                  DAD     H
 F4BD DB20        I8250C: IN      SDATA   ;WASTE SOME TIME
 F4BF 2B                  DCX     H
 F4C0 7D                  MOV     A,L
 F4C1 B4                  ORA     H
 F4C2 C2BDF4              JNZ     I8250C
 F4C5 E1                  POP     H       ;REGET 2 MHZ DIVISOR
 F4C6 3E83        I8250D: MVI     A,83H   ;SET DIVISOR REGISTER ACCESS
 F4C8 D323                OUT     SLCTRL
 F4CA 7D                  MOV     A,L     ;SET THE DIVISOR
 F4CB D320                OUT     SDATA
 F4CD 7C                  MOV     A,H
 F4CE D321                OUT     SINTEN
 F4D0 3E03                MVI     A,3     ;SET DATA REGISTER ACCESS
 F4D2 D323                OUT     SLCTRL
 F4D4 AF                  XRA     A       ;DISABLE INTERRUPTS
 F4D5 D321                OUT     SINTEN
 F4D7 D325                OUT     SLSTAT  ;AND RESET ERROR FLAGS
 F4D9 CDCEF6              CALL    TTYIN   ;GET A CHARACTER
 F4DC E67F                ANI     7FH     ;STRIP OFF ANY PARITY BIT
 F4DE FE0D                CPI     0DH     ;SEE IF IT IS A CARRIAGE RETURN
 F4E0 E1                  POP     H       ;SET THE STACK STRAIGHT
 F4E1 C8                  RZ              ;DONE IF CARRIAGE RETURN RECEIVED
 F4E2 5D                  MOV     E,L     ;ELSE, MUST BE 4 MHZ SYSTEM
 F4E3 54                  MOV     D,H     ; SO, COUNT=COUNT*5/4
 F4E4 CDEEF4              CALL    DIV2
 F4E7 CDEEF4              CALL    DIV2
 F4EA 19                  DAD     D
 F4EB E5                  PUSH    H
                          JMPR    I8250D  ;GO SET THE NEW DIVISOR
 F4EC+18D8
                ;
                ;
 F4EE B7          DIV2:   ORA     A       ;CLEAR THE CARRY BIT
 F4EF 7C                  MOV     A,H     ;DO A 16-BIT RIGHT SHIFT
 F4F0 1F                  RAR
 F4F1 67                  MOV     H,A
 F4F2 7D                  MOV     A,L
 F4F3 1F                  RAR
 F4F4 6F                  MOV     L,A
 F4F5 C9                  RET
                ;
                ; EOF ROUTINE PUNCHES AN END OF FILE RECORD (INTEL HEX
                ;   FORMAT) ONTO THE CURRENTLY ASSIGNED PAPER TAPE PUNCH
                ;   DEVICE.  AN ENTRY POINT ADDRESS FOR THE FILE WILL ALSO
                ;   BE PUNCHED, IF SPECIFIED.
                ;
 F4F6 CDA4F6      EOF:    CALL    EXLF    ;GET JUMP ADDRESS
 F4F9 D5                  PUSH    D       ;SAVE THE # OF TRAILER NULLS
 F4FA CDC8F5      EOFA:   CALL    PSOR    ;PUNCH START OF RECORD
 F4FD AF                  XRA     A       ;ZERO OUT THE CHECKSUM
 F4FE 57                  MOV     D,A
 F4FF CDF6F6              CALL    PBADR   ;OUTPUT THE RECORD LENGTH AND EP
 F502 3E01                MVI     A,1     ;PUNCH RECORD TYPE = 1
 F504 CDFEF6              CALL    PBYTE
 F507 AF                  XRA     A
 F508 92                  SUB     D       ;OUTPUT THE CHECKSUM
 F509 CDFEF6              CALL    PBYTE
                          JMPR    LE0     ;GO DO THE TRAILER
 F50C+1803
```

```
CP/M MACRO ASSEM 2.0    #018    MOSS 2.2 MONITOR

                ;
                ; LEADER ROUTINE "PUNCHES" SIX INCHES (OR AS SPECIFIED)
                ;   OF LEADER ON THE PAPER TAPE PUNCH.  NULLS ARE PUNCHED
                ;   TO FORM THE LEADER (OR TRAILER).
                ;
F50E CDD7F0     LEADER: CALL    EXPR1   ;SEE IF SOME OTHER LENGTH WANTED
F511 C1         LE0:    POP     B       ;GET THE VALUE
F512 78                 MOV     A,B
F513 B1                 ORA     C       ;TEST FOR DEFAULT SELECT
F514 41                 MOV     B,C     ;MOVE NEW VALUE IN JUST IN CASE
F515 0E00               MVI     C,0     ;GET A NULL CHARACTER
                        JRNZ    LE1     ;JUMP IF NEW VALUE WANTED

F517+2002
F519 063C               MVI     B,60    ;DEFAULT, SET 60 NULLS
F51B CD0CF0     LE1:    CALL    PUNCH   ;PUNCH ONE NULL
                        DJNZ    LE1     ;KEEP GOING TIL DONE

F51E+10FB
F520 C9                 RET
                ;
                ; QUERY ROUTINE WILL TELL THE OPERATOR WHAT HIS CURRENT LOGICA
                ;       PHYSICAL PERIPHERAL DEVICE ASSIGNMENTS ARE.  NO PARAME
                ;       (OTHER THAN A CARRIAGE RETURN) ARE REQUIRED ON ENTRY.
                ;
F521 3A0300     QUERY:  LDA     IOBYTE  ;GET THE ASSIGNMENT CONTROL BYTE
F524 0604               MVI     B,4     ;SET UP FOR FOUR LOGICAL DEVICES
F526 217DF1             LXI     H,ACT   ;ADDRESS OF CONVERSION TABLE
F529 11FBFF             LXI     D,ALT-APT  ;NEGATIVE OFFSET FOR LOGICAL TABLE
F52C F5         QUE1:   PUSH    PSW
F52D CDFEF5             CALL    BLK     ;FORMAT THE PRINT-OUT
F530 4E                 MOV     C,M     ;GET THE CURRENT LOGICAL DEVICE CODE
F531 CD09F0             CALL    CONOUT  ;OUTPUT IT
F534 CDF7F5             CALL    DASH    ;OUTPUT A DASH
F537 F1                 POP     PSW     ;REGET THE CONTROL BYTE
F538 F5                 PUSH    PSW     ;RESAVE IT
F539 E5                 PUSH    H       ;SAVE THE TABLE POINTER
F53A 23         QUE2:   INX     H       ;ADJUST POINTER TO CURRENT PHYSICAL DE
F53B 3C                 INR     A
F53C E603               ANI     3       ;BITS 0 AND 1 ARE 0 WHEN ON CURRENT AS
                        JRNZ    QUE2    ;NOT THERE YET, TRY AGAIN

F53E+20FA
F540 4E                 MOV     C,M     ;FOUND IT, NOW PRINT IT
F541 CD09F0             CALL    CONOUT
F544 E1                 POP     H
F545 F1                 POP     PSW     ;GO TO NEXT LOGICAL DEVICE
F546 1F                 RAR             ;ADJUST THE IOBYTE
F547 1F                 RAR
F548 19                 DAD     D       ;ADJUST THE TABLE POINTER
                        DJNZ    QUE1    ;GO DO NEXT LOGICAL DEVICE

F549+10E1
F54B C9                 RET             ;RETURN TO MONITOR
                ;
                ; READ ROUTINE READS AN INTEL HEX FORMAT PAPER TAPE FROM
                ;   THE CURRENT PAPER TAPE READER.  IF A NON-ZERO ADDRESS
                ;   IS SPECIFIED IN THE END OF FILE RECORD, CONTROL WILL
                ;   BE TRANSFERRED TO THAT ADDRESS.  OTHERWISE, CONTROL
                ;   WILL REVERT TO THE EXECUTIVE.
                ;
F54C CDD7F0     READ:   CALL    EXPR1   ;GET OFFSET BIAS
F54F E1         RED0:   POP     H       ;   INTO (H,L)
F550 E5                 PUSH    H       ;SAVE THE BIAS
F551 CD86F4     RED1:   CALL    RIX     ;READ A BYTE
F554 DE3A               SBI     ':'     ;LOOK FOR START OF RECORD
                        JRNZ    RED1    ;JUMP TO KEEP LOOKING

F556+20F9
F558 57                 MOV     D,A     ;INITIALIZE CHECKSUM
F559 CD73F4             CALL    BYTE    ;GET RECORD LENGTH
                        JRZ     RED3    ;JUMP IF EOF RECORD

F55C+2823
```

```
CP/M MACRO ASSEM 2.0     #019     MOSS 2.2 MONITOR

 F55E 5F                         MOV     E,A       ;ELSE, ASSUME DATA RECORD
 F55F CD73F4                     CALL    BYTE      ;GET LOAD ADDRESS HIGH BYTE
 F562 F5                         PUSH    PSW       ;SAVE IT
 F563 CD73F4                     CALL    BYTE      ;GET LOAD ADDRESS LOW BYTE
 F566 C1                         POP     B         ;BUILD ADDRESS IN (B,C)
 F567 4F                         MOV     C,A
 F568 09                         DAD     B         ;ADD ON THE BIAS
 F569 CD73F4                     CALL    BYTE      ;SKIP OVER RECORD TYPE
 F56C CD73F4             RED2:   CALL    BYTE      ;GET A DATA BYTE
 F56F 77                         MOV     M,A       ;PUT IT INTO MEMORY
 F570 2F                         CMA               ;DO A QUICK CHECK
 F571 AE                         XRA     M         ;  RESULT SHOULD BE ZERO
 F572 C4A1F2                     CNZ     BITS      ;IF ERROR, PRINT ADDRESS AND DATA
 F575 23                         INX     H         ;INCREMENT MEMORY POINTER
 F576 1D                         DCR     E         ;RECORD LENGTH FOR LOOP CONTROL
                                 JRNZ    RED2      ;DO REST OF THE RECORD

 F577+20F3
 F579 CD73F4                     CALL    BYTE      ;GET THE CHECKSUM
 F57C C209F1                     JNZ     QPRT      ;ABORT IF ERROR
                                 JMPR    REDO      ;GO DO NEXT RECORD

 F57F+18CE
 F581 CD73F4             RED3:   CALL    BYTE      ;EOF RECORD, GET ENTRY POINT
 F584 67                         MOV     H,A       ;HIGH BYTE TO (H)
 F585 CD73F4                     CALL    BYTE      ;GET THE LOW BYTE
 F588 6F                         MOV     L,A
 F589 B4                         ORA     H         ;SEE IF IT IS ZERO
 F58A D1                         POP     D         ;RESTORE THE STACK
 F58B C8                         RZ                ;RETURN TO MONITOR IF EP=0
 F58C E9                         PCHL              ;ELSE, GO TO THE ENTRY POINT
                         ;
                         ; WRITE ROUTINE IS USED TO PUNCH AN INTEL HEX FORMAT
                         ;   PAPER TAPE ON THE CURRENT ASSIGNED PUNCH UNIT.
                         ;
 F58D CD86F3             WRITE:  CALL    EXPR3     ;GET 3 PARAMETERS, DO CRLF
 F590 AF                         XRA     A         ;SEE IF RECORD LENGTH CHANGE
 F591 47                         MOV     B,A       ;SET HIGH BYTE TO ZERO
 F592 B1                         ORA     C         ;NOW SEE IF CHANGE WANTED
                                 JRNZ    WRI1      ;YES, JUMP AND SET IT UP

 F593+2002
 F595 0E10                       MVI     C,16      ;NO, DEFAULT TO 16 BYTES/RECORD
 F597 E5                 WRI1:   PUSH    H         ;SAVE MEMORY POINTER
 F598 09                         DAD     B         ;ADD THE RECORD LENGTH
 F599 B7                         ORA     A         ;CLEAR THE CARRY BIT
                                 DSBC    D         ;SEE IF FULL RECORD REMAINS

 F59A+ED52
 F59C E1                         POP     H         ;RESTORE (H,L)
                                 JRC     WRI2      ;GO DO A FULL RECORD

 F59D+380A
 F59F D5                         PUSH    D         ;SAVE LAST BYTE ADDRESS
 F5A0 EB                         XCHG              ;SWAP (D,E) AND (H,L)
 F5A1 B7                         ORA     A         ;RESET THE CARRY BIT
                                 DSBC    D         ;FIND # OF BYTE REMAINING

 F5A2+ED52
 F5A4 23                         INX     H         ;ADJUST TO INCLUDE LAST BYTE
 F5A5 E3                         XTHL              ;SWAP TOP OF STACK
 F5A6 EB                         XCHG              ;SET (D,E), (H,L) TO NORMAL
 F5A7 C1                         POP     B         ;NEW RECORD LENGTH TO (B,C)
 F5A8 D8                         RC                ;DONE IF ZERO LENGTH RECORD
 F5A9 C5                 WRI2:   PUSH    B         ;SAVE LOOP COUNT
 F5AA D5                         PUSH    D
 F5AB 50                         MOV     D,B       ;ZERO THE CHECKSUM
 F5AC 41                         MOV     B,C       ;MOVE LOOP CONTROL TO B
 F5AD CDC8F5                     CALL    PSOR      ;PUNCH START OF RECORD
 F5B0 78                         MOV     A,B       ;GET RECORD LENGTH
 F5B1 CDF6F6                     CALL    PBADR     ;PUNCH IT
 F5B4 AF                         XRA     A         ;PUNCH RECORD TYPE '0'
 F5B5 CDFEF6                     CALL    PBYTE
 F5B8 7E                 WRI3:   MOV     A,M       ;GET NEXT DATA BYTE
```

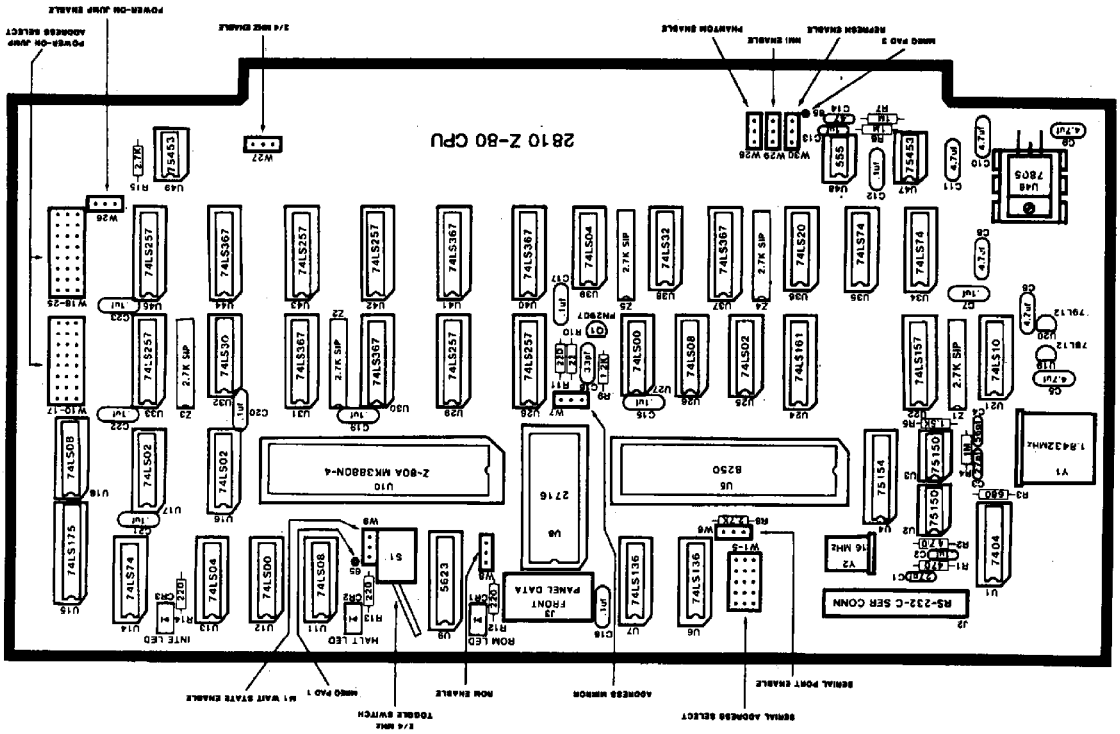
```
CP/M MACRO ASSEM 2.0    #020    MOSS 2.2 MONITOR

 F5B9 23                INX     H       ;BUMP THE POINTER
 F5BA CDFEF6            CALL    PBYTE   ;PUNCH THE DATA
                        DJNZ    WRI3    ;DO REST OF RECORD

 F5BD+10F9
 F5BF AF                XRA     A       ;NOW, DO THE CHECKSUM
 F5C0 92                SUB     D
 F5C1 CDFEF6            CALL    PBYTE   ;PUNCH IT
 F5C4 D1                POP     D       ;RESTORE THE REGISTERS
 F5C5 C1                POP     B
                        JMPR    WRI1    ;GO DO NEXT RECORD

 F5C6+18CF
                ;
 F5C8 CD7CF4    PSOR:   CALL    PEOL
 F5CB 0E3A              MVI     C,':'
 F5CD C37CF6            JMP     PO
                ;
                ; HEXN ROUTINE
                ;
                ;THIS ROUTINE ADDS AND SUBTRACTS TWO HEXADECIMAL 16-BIT
                ;       UNSIGNED NUMBERS AND DISPLAYS THE RESULTS ON THE
                ;       CONSOLE.
                ;
 F5D0 CDA4F6    HEXN:   CALL    EXLF    ;GET THE TWO NUMBERS
 F5D3 E5                PUSH    H       ;SAVE IT FOR THE SUBTRACT
 F5D4 19                DAD     D       ;ADD THEM
 F5D5 CDFBF5            CALL    LADRB   ;OUTPUT THEM
 F5D8 E1                POP     H       ;REGET THE FIRST NUMBER
 F5D9 B7                ORA     A       ;CLEAR THE CARRY BIT
                        DSBC    D       ;DO THE SUBTRACT

 F5DA+ED52
                        JMPR    LADR    ;GO OUTPUT THE RESULT

 F5DC+1803
                ;
                ; ROUTINE LADR PRINTS THE CONTENTS OF (H,L) ON THE
                ;       CURRENT CONSOLE, EITHER AT THE START OF A NEW
                ;       LINE (EP = LADRA) OR AT THE CURRENT LOCATION (EP
                ;       = LADR).
                ;
 F5DE CDA9F6    LADRA:  CALL    CRLF    ;START A NEW LINE
 F5E1 7C        LADR:   MOV     A,H     ;GET HIGH TWO DIGITS
 F5E2 CDE6F5            CALL    HEX1    ;PRINT THEM
 F5E5 7D                MOV     A,L     ;GET LOW TWO DIGITS
 F5E6 F5        HEX1:   PUSH    PSW     ;SAVE THE LOW DIGIT
 F5E7 0F                RRC             ;PUT HIGH NIBBLE INTO BITS 0-3
 F5E8 0F                RRC
 F5E9 0F                RRC
 F5EA 0F                RRC
 F5EB CDEFF5            CALL    HEX2    ;GO PRINT SINGLE DIGIT
 F5EE F1                POP     PSW     ;REGET THE LOW DIGIT
 F5EF CD6EF3    HEX2:   CALL    CONV    ;GO INSERT ASCII ZONE
                        JMPR    CO      ;DO THE CHARACTER OUTPUT

 F5F2+180C
                ;
                ; ROUTINE DASH TYPES A DASH ON THE CURRENT CONSOLE DEVICE.
                ;
 F5F4 CDE6F5    DASH1:  CALL    HEX1    ;FIRST, PRINT ACCUM AS TWO HEX DIGITS
 F5F7 0E2D      DASH:   MVI     C,'-'   ;GET AN ASCII DASH
                        JMPR    CO      ;GO TYPE IT

 F5F9+1805
                ;
                ;  IOBYTE HANDLERS
                ;
 F5FB                   ORG     MOSS+5FBH
                ;
 F5FB CDDEF5    LADRB:  CALL    LADRA   ;OUTPUT (H,L) AS 4 ASCII DIGITS
                ;
 F5FE 0E20      BLK:    MVI     C,' '   ;OUTPUT A BLANK
                ;
```

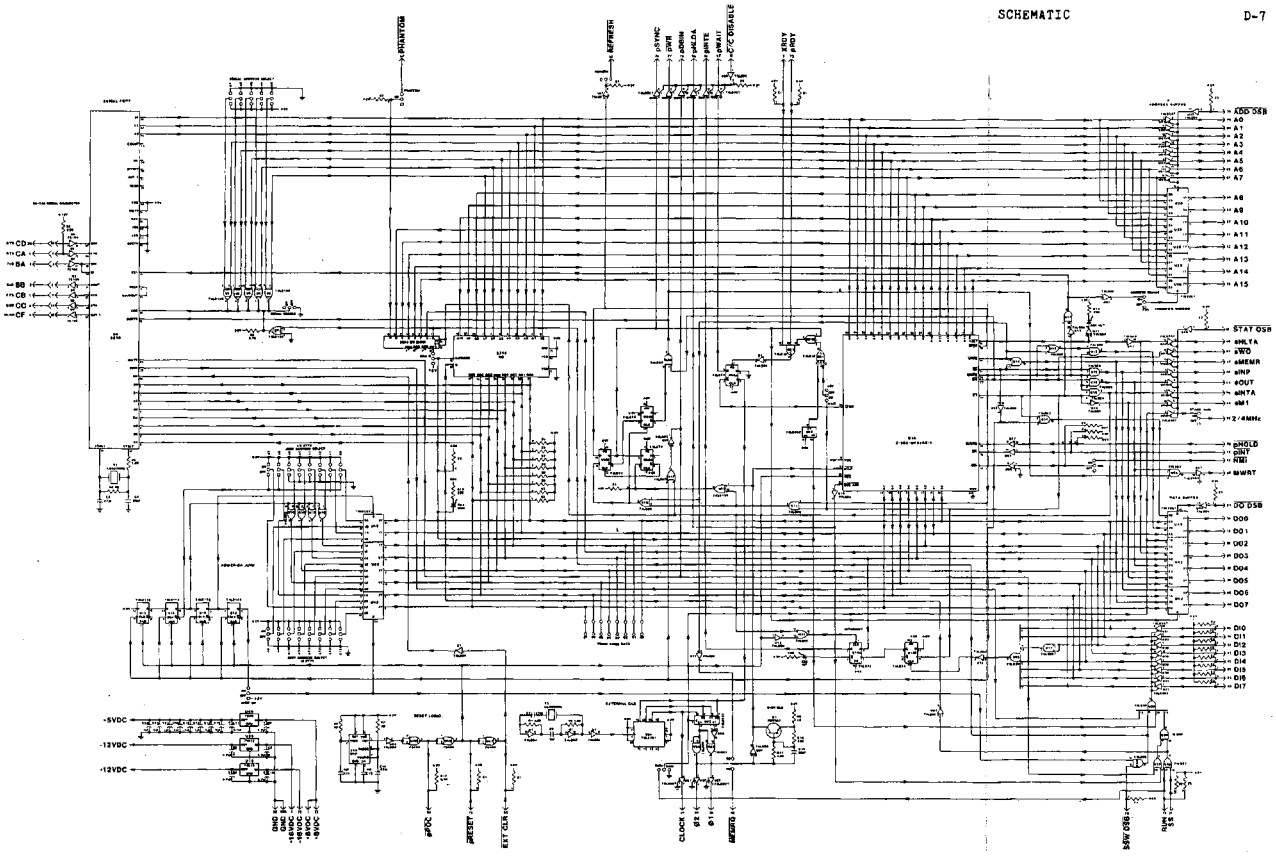
```
CP/M MACRO ASSEM 2.0    #021    MOSS 2.2 MONITOR

F600 3A0300    CO:     LDA     IOBYTE
F603 E603              ANI     3       ;ISOLATE CONSOLE ASGT
F605 CADEF6            JZ      TTYOUT  ;TTY DEVICE ACTIVE
F608 FE02              CPI     2
F60A FA62F4            JM      CRTOUT  ;CRT ACTIVE
F60D C262F4            JNZ     CUSO1   ;USER CONSOLE 1 ACTIVE

F610 3A0300    LO:     LDA     IOBYTE
F613 E6C0              ANI     0C0H    ;ISOLATE LIST ASGT
F615 CADEF6            JZ      TTYOUT  ;TTY DEVICE ACTIVE
F618 FE80              CPI     80H
F61A FA62F4            JM      CRTOUT  ;CRT ACTIVE
F61D CA62F4            JZ      LPRT    ;LINE PRINTER ACTIVE
F620 C362F4            JMP     LUSE1   ;USER PRINTER 1 ACTIVE

F623 3A0300    CSTS:   LDA     IOBYTE
F626 E603              ANI     3       ;ISOLATE CONSOLE ASGT
F628 CAC6F6            JZ      TTST    ;TTY ACTIVE
F62B FE02              CPI     2
F62D FA62F4            JM      CRTST   ;CRT ACTIVE
F630 C262F4            JNZ     CUST1   ;USER CONSOLE 1 ACTIVE

F633 3A0300    BATST:  LDA     IOBYTE
F636 E60C              ANI     0CH     ;ISOLATE BATCH ASGT
F638 CAC6F6            JZ      TTST    ;TTY ACTIVE
F63B FE08              CPI     8
F63D FA62F4            JM      PTRST   ;PAPER TAPE READER ACTIVE
F640 CA62F4            JZ      RUST1   ;USER READER 1 ACTIVE
F643 C362F4            JMP     RUST2   ;USER READER 2 ACTIVE

F646 3A0300    CI:     LDA     IOBYTE
F649 E603              ANI     3       ;ISOLATE CONSOLE ASGT
F64B CACEF6            JZ      TTYIN   ;TTY DEVICE ACTIVE
F64E FE02              CPI     2
F650 FA62F4            JM      CRTIN   ;CRT ACTIVE
F653 C262F4            JNZ     CUSI1   ;USER CONSOLE 1 ACTIVE

F656 3A0300    RI:     LDA     IOBYTE
F659 E60C              ANI     0CH     ;ISOLATE BATCH ASGT
F65B CACEF6            JZ      TTYRDR  ;TTY ACTIVE
F65E FE08              CPI     8
F660 FA62F4            JM      PTRIN   ;PAPER TAPE READER ACTIVE
F663 CA62F4            JZ      RUSI1   ;USER READER 1 ACTIVE
F666 C362F4            JMP     RUSI2   ;USER READER 2 ACTIVE

F669 3A0300    LSTAT:  LDA     IOBYTE
F66C E6C0              ANI     0C0H    ;ISOLATE THE LIST DEVICE ASSIGNMENT
F66E CAD6F6            JZ      TTOST
F671 FE80              CPI     80H
F673 FA62F4            JM      CRTOST
F676 CA62F4            JZ      LPRST
F679 C362F4            JMP     LUST1

F67C 3A0300    PO:     LDA     IOBYTE
F67F E630              ANI     30H     ;ISOLATE PUNCH ASGT
F681 CADEF6            JZ      TTPNCH  ;TTY ACTIVE
F684 FE20              CPI     20H
F686 FA62F4            JM      HSP     ;HIGH SPEED PUNCH ACTIVE
F689 CA62F4            JZ      PUSO1   ;USER PUNCH 1 ACTIVE
F68C C362F4            JMP     PUSO2   ;USER PUNCH 2 ACTIVE

                       ;
                       ; ROUTINE CONI READS THE CONSOLE AND STRIPS OFF THE ASCII
                       ;       PARITY BIT.
                       ;
F68F CD46F6    CONI:   CALL    CI      ;GET THE NEXT CHARACTER
F692 E67F              ANI     7FH     ;STRIP OFF THE PARITY BIT
F694 C9        RTS:    RET
                       ;
```

```
CP/M MACRO ASSEM 2.0    #022    MOSS 2.2 MONITOR

                ; ROUTINE PRTWD PRINTS AN ASCII STRING ONTO THE CONSOLE.
                ;       THE STRING MUST BE TERMINATED BY BIT 7 SET IN THE
                ;       LAST CHARACTER OF THE STRING.  THE STRING WILL START
                ;       A NEW LINE (EP = PRTWD) OR CONTINUE ON THE SAME
                ;       LINE (EP = PRTWA)
                ;
F695 CDA9F6     PRTWD:  CALL    CRLF    ;START A NEW LINE
F698 C5         PRTWA:  PUSH    B       ;SAVE (B,C)
F699 4E         PRTA:   MOV     C,M     ;GET NEXT CHARACTER FROM MEMORY
F69A CD00F6             CALL    CO      ;OUTPUT IT
F69D 23                 INX     H       ;INCREMENT MEMORY POINTER
F69E 79                 MOV     A,C
F69F 07                 RLC             ;TEST FOR BIT 7 DELIMITER
                        JRNC    PRTA    ;NO DELIMITER, GO DO NEXT CHARACTER
F6A0+30F7
F6A2 C1         PRTB:   POP     B       ;RESTORE (B,C)
F6A3 C9                 RET
                ;
                ; ROUTINE EXLF READS TWO PARAMETERS, PUTS THEM INTO THE
                ;       D,E AND H,L REGISTERS, THEN DOES A CARRIAGE RETURN,
                ;       LINE FEED SEQUENCE.
                ;
F6A4 CDD9F0     EXLF:   CALL    EXPR    ;GO GET TWO PARAMETERS
F6A7 D1                 POP     D
F6A8 E1                 POP     H
                ;
                ; ROUTINE CRLF GENERATES A CARRIAGE RETURN, LINE FEED
                ;       SEQUENCE ON THE CURRENT CONSOLE TO START A NEW LINE
                ;       IT INCLUDES TRHEE NULL CHARACTERS FOR TTY TYPE
                ;       DEVICES FOR THE HEAD MOVEMENT TIME.
                ;
F6A9 E5         CRLF:   PUSH    H       ;SAVE THE CONTENTS OF (H,L)
F6AA 21C2F6     CRLFA:  LXI     H,CRMSG ;ADDRESS OF CR,LF MESSAGE
F6AD CD98F6             CALL    PRTWA   ;  OUTPUT IT
F6B0 E1                 POP     H       ;RESTORE (H,L)
F6B1 C9                 RET
                ;
F6B2 21BBF6     RSTER:  LXI     H,RSTMSG ;GET ADDRESS OF RESTART ERROR MSG
F6B5 CD95F6     COMERR: CALL    PRTWD   ;PRINT IT ON NEW LINE
F6B8 C30000             JMP     WSVEC   ;GO TO WARM BOOT
                ;
F6BB 5253542045 RSTMSG: DB      'RST ER','R'+80H
F6C2 0D0A0080   CRMSG:  DB      CR,LF,0,80H
                ;
                ; I/O DRIVERS FOR THE 8250 ASYNC COMM ELEMENT
                ;
F6C6 DB25       TTST:   IN      SLSTAT  ;GET 8250 LINE STATUS
F6C8 E601               ANI     1       ;SEE IF RECEIVE DATA AVAILABLE
F6CA C8                 RZ              ;RETURN IF NOT
F6CB C6FE               ADI     0FEH    ;FLAG THAT DATA IS AVAILABLE
F6CD C9                 RET
                ;
F6CE DB25       TTYIN:  IN      SLSTAT  ;GET 8250 LINE STATUS
F6D0 1F                 RAR             ;MOVE RX DATA READY BIT INTO CARRY
                        JRNC    TTYIN   ;LOOP UNTIL DATA IS IN
F6D1+30FB
F6D3 DB20               IN      SDATA   ;READ THE DATA
F6D5 C9                 RET
                ;
F6D6 DB25       TTOST:  IN      SLSTAT  ;GET 8250 LINE STATUS
F6D8 E620               ANI     20H     ;ISOLATE TX BUFFER EMPTY BIT
F6DA C8                 RZ              ;RETURN IF NOT EMPTY
F6DB C6BF               ADI     0BFH    ;FLAG THE EMPTY STATE
F6DD C9                 RET
                ;
F6DE DB25       TTYOUT: IN      SLSTAT  ;GET 8250 LINE STATUS
F6E0 E620               ANI     20H     ;ISOLATE THRE BIT
                        JRZ     TTYOUT  ;WAIT UNTIL ONE OF THE REGISTERS EMPTI
```

```
CP/M MACRO ASSEM 2.0    #023    MOSS 2.2 MONITOR

 F6E2+28FA
 F6E4 79                MOV     A,C     ;MOVE THE DATA OVER
 F6E5 D320              OUT     SDATA   ;OUTPUT THE DATA
 F6E7 C9                RET

                ;
                ; EQUATES FOR ADDITIONAL CONSOLE DEVICES
                ;
 F462 =         CRTIN:  EQU     IOER
 F462 =         CRTOUT: EQU     IOER
 F462 =         CRTST:  EQU     IOER
 F462 =         CRTOST: EQU     IOER    ;UNASSIGNED CRT OUTPUT STATUS
 F462 =         CUSI1:  EQU     IOER    ;UNASSIGNED USER CONSOLE (INPUT)
 F462 =         CUSO1:  EQU     IOER    ;UNASSIGNED USER CONSOLE (OUTPUT)
 F462 =         CUST1:  EQU     IOER
                ;
                ;  EQUATES FOR ADDITIONAL PAPER TAPE PUNCH DEVICES
                ;
 F6DE =         TTPNCH: EQU     TTYOUT  ;UNASSIGNED TELETYPE PUNCH
 F462 =         HSP:    EQU     IOER    ;UNASSIGNED HIGH SPEED PUNCH
 F462 =         HSPST:  EQU     IOER    ;UNASSIGNED HIGH SPEED PUNCH STATUS
 F462 =         PUSO1:  EQU     IOER    ;UNASSIGNED USER PUNCH 1
 F462 =         PUSO2:  EQU     IOER    ;UNASSIGNED USER PUNCH 2
                ;
                ;  EQUATES FOR ADDITIONAL LIST DEVICES
                ;
 F462 =         LPRT:   EQU     IOER    ;UNASSIGNED LINE PRINTER
 F462 =         LPRST:  EQU     IOER    ;UNASSIGNED PRINTER STATUS
 F462 =         LUSE1:  EQU     IOER    ;LIST DEVICE 1
 F462 =         LUST1:  EQU     IOER    ;LIST DEVICE 1 STATUS
                ;
                ;  EQUATES FOR ADDITIONAL PAPER TAPE READER DEVICES
                ;
 F6CE =         TTYRDR: EQU     TTYIN   ;UNASSIGNED TELETYPE PAPER TAPE READER
 F462 =         PTRIN:  EQU     IOER    ;UNASSIGNED HIGH SPEED PAPER TAPE READ
 F462 =         PTRST:  EQU     IOER    ;UNASSIGNED HS PTR STATUS
 F462 =         RUSI1:  EQU     IOER    ;UNASSIGNED PAPER TAPE READER 1
 F462 =         RUST1:  EQU     IOER    ;UNASSIGNED PAPER TAPE READER 1 (STATU
 F462 =         RUSI2:  EQU     IOER    ;UNASSIGNED PAPER TAPE READER 2
 F462 =         RUST2:  EQU     IOER    ;UNASSIGNED PAPER TAPE READER 2 (STATU
                ;
 F6E8 CDF0F6    BYT:    CALL    RIBBLE  ;READ AND CONVERT ONE CHARACTER
 F6EB 07                RLC             ;SHIFT INTO HIGH NIBBLE
 F6EC 07                RLC
 F6ED 07                RLC
 F6EE 07                RLC
 F6EF 47                MOV     B,A     ;SAVE IN B TEMPORARILY
 F6F0 CD86F4    RIBBLE: CALL    RIX     ;READ A CHARACTER
 F6F3 C3B0F3            JMP     NIBBLE  ;GO CONVERT TO HEX DIGIT
                ;
                ; PADR ROUTINE PUNCHES (H,L) AS FOUR ASCII CHARACTERS.
                ;   IT IS USED TO PUT THE ADDRESS INTO AN INTEL HEX
                ;   FORMAT RECORD.
                ;
 F6F6 CDFEF6    PBADR:  CALL    PBYTE
 F6F9 7C        PADR:   MOV     A,H
 F6FA CDFEF6            CALL    PBYTE
 F6FD 7D                MOV     A,L
                ;
                ; PBYTE ROUTINE PUNCHES (A) AS TWO ASCII CHARACTERS ON
                ;   THE CURRENT PUNCH DEVICE.
                ;
 F6FE F5        PBYTE:  PUSH    PSW     ;SAVE THE BYTE
 F6FF 0F                RRC             ;DO HIGH NIBBLE FIRST
 F700 0F                RRC
 F701 0F                RRC
 F702 0F                RRC
 F703 CD6EF3            CALL    CONV    ;CONVERT TO ASCII
 F706 CD0CF0            CALL    PUNCH   ;PUNCH IT
```

```
CP/M MACRO ASSEM 2.0     #024     MOSS 2.2 MONITOR

 F709 F1                 POP      PSW       ;GET LOW NIBBLE
 F70A F5                 PUSH     PSW       ;RESAVE FOR CHECKSUM
 F70B CD6EF3             CALL     CONV      ;CONVERT TO ASCII
 F70E CD0CF0             CALL     PUNCH     ;PUNCH IT
 F711 F1                 POP      PSW
 F712 82                 ADD      D         ;UPDATE CHECKSUM
 F713 57                 MOV      D,A
 F714 C9                 RET
                  ;
 F715                    END
```


2810 Z-80 CPU
Z-80A MK3880N-4
8250
2716
FRONT PANEL DATA
RS-232-C SER CONN
74LS257
74LS367
74LS04
74LS32
74LS20
74LS74
74LS30
74LS00
74LS08
74LS02
74LS161
74LS157
74LS10
74LS175
74LS136
5623
75154
75150
7404
7805
555
2.7K SIP
16 MHz
1.8432MHz


PHANTOM
REFRESH
ADD OSB
STAT OSB
DO OSB
+5VDC
-12VDC
+12VDC
GND
+5VDC
-5VDC
+12VDC
-12VDC
CLOCK
RESET
MEMREQ
SAM OSB

APPENDIX D

PARTS LIST, BOARD LAYOUT, SCHEMATIC, SPECIFICATIONS

QTY	REF NO.	DESCRIPTION	CCS PART NO.*
Capacitors			
2	C1,3	27pf. Mica	42215-52705
12	C2,7,12,13 15,C17-23	.1uf 50v Monolythic	42034-21046
1	C4	56pf 500v Mica	42215-55605
6	C5,6,8-11	4.7uf 35v Dip Tantalum	42804-54756
1	C14	.47uf 50v Monolythic	42034-24746
1	C16	33pf Mica	42215-53305
Integrated Circuits			
1	U1	7404	30200-07404
2	U13,39	74LS04	30000-00004
2	U2,3	75150	30300-00150
1	U4	75154	30300-00154
1	U5	8250	31200-08250
2	U6,7	74LS136	30000-00136
1	U8	2716, 2048 X 8 EPROM	31900-02716
1	U9	5623, 256 X 4 ROM	30900-05623
1	U10	Z-80	31200-38804
3	U11,18,26	74LS08	30000-00008
2	U12,27	74LS00	30000-00000
3	U14,34,35	74LS74	30000-00074
1	U15	74LS175	30000-00175
3	U16,17,25	74LS02	30000-00002
1	U19	78L12, +12V Regulator	32000-17812
1	U20	79L12, -12V Regulator	32000-17912
1	U21	74LS10	30000-00010
7	U22,28,29,33,42, 43,45	74LS257	30000-00257
1	U24	74LS161	30000-00161
6	U30,31,37,40,41, 44	74LS367	30000-00367
1	U32	74LS30	30000-00030
1	U36	74LS20	30000-00020
1	U38	74LS32	30000-00032
1	U46	7805, +5V Regulator	32000-07805
2	U47,49	75453	30300-00453
1	U48	555	30900-00555
Resistors			
2	R1,2	470 1/4W 5%	40002-04715
1	R3	680 ohm 1/4W 5%	40002-06815
1	R4	1.5K 1/4W 5%	40002-01525

* Use CCS part number when ordering spare parts or replacements.

CONTINUED

QTY	REF NO.	DESCRIPTION	CCS PART NO.
3	R5-7	1M 1/4W 5%	40002-01055
2	R8,R15	2.7K 1/4W 5%	40002-02725
1	R9	1.2K 1/4W 5%	40002-01225
1	R10	22 ohm 1/4W 5%	40002-02205
4	R11-14	220 ohm 1/4W 5%	40002-02215
5	Z1-5	2.7K X 7 SIP Network	40930-72726
IC Sockets			
20	XU1,6,7,11-14,16-18,21,25-27,32,34-36,38,39	14-Pin Low Profile	58102-00140
5	XU2,3,47-49	8-Pin Low Profile	58102-00080
18	XU4,9,15,22,24,28-31,33,37,40-45, J3	16-Pin Low Profile	58102-00160
1	XU8	24-Pin Low Profile	58102-00240
2	XU5,10	40-Pin Low Profile	58102-00400
Miscellaneous			
3	CR1-3	LED, Rectangular Red	37400-00001
1	J2	Header, 2 x 13 Right Angle	56005-02013
1	Q1	Transistor, PN2907	36100-02907
1	S1	Switch, Toggle	27391-12000
30	W1-30	Header, 1 x 3 Straight	56004-01003
30	W1-30	Berg Jumper Plugs	56200-00001
1	Y1	Crystal, 1.8432 MHz	48132-84321
1	Y2	Crystal, 16.000 MHz	48231-60003
1	--	Heatsink	60022-00001
1	--	Nut, Hex Kep 6-32	73006-32001
1	--	Screw, 6-32 x 5/16"	71006-32051
2	--	Tape, Foam Two-sided	60003-00001
1	--	PC Board, 2810 CPU, rev A	02810-00002
2	--	Extractor, PCB nonlocking	60010-00001
2	--	Extractor Roll Pins	60010-00000

2810 Z-80 CPU SPECIFICATIONS

BOARD MEASUREMENTS

Board: 10" L x 5" W
Connector: 6.35" L x .3" W (2.125" from right of board)
0.125" pin spacing
Component Height: less than .5"
Weight: approximately 11 ounces

POWER

Supply: Unregulated +8, +16, -16 volts
Maximum power draw: .650 amps at +8 volts
.030 amps at +16 volts
.025 amps at -16 volts
Power Dissipation: 6.2 watts

ENVIRONMENTAL REQUIREMENTS

Temperature: 0 to 70 degrees Celsius
Humidity: 0 to 90% noncondensing

APPENDIX E

LIMITED WARRANTY

California Computer Systems (CCS) warrants to the original purchaser of its products that its CCS assembled and tested products will be free from materials defects for a period of one (1) year, and be free from defects of workmanship for a period of ninety (90) days.

The responsibility of CCS hereunder, and the sole and exclusive remedy of the original purchaser for a breach of any warranty hereunder, is limited to the correction or replacement by CCS at CCS's option, at CCS's service facility, of any product or part which has been returned to CCS and in which there is a defect covered by this warranty; provided, however, that in the case of CCS assembled and tested products, CCS will correct any defect in materials and workmanship free of charge if the product is returned to CCS within ninety (90) days of original purchase from CCS; and CCS will correct defects in materials in its products and restore the product to an operational status for a labor charge of $25.00, provided that the product is returned to CCS within one (1) year in the case of CCS assembled and tested products. All such returned products shall be shipped prepaid and insured by original purchaser to:

Warranty Service Department
California Computer Systems
250 Caribbean Drive
Sunnyvale, California
94086

CCS shall have the right of final determination as to the existence and cause of a defect, and CCS shall have the sole right to decide whether the product should be repaired or replaced.

This warranty shall not apply to any product or any part

thereof which has been subject to

(1) accident, neglect, negligence, abuse or misuse;

(2) any maintenance, overhaul, installation, storage, operation, or use, which is improper; or

(3) any alteration, modification, or repair by anyone other than CCS or its authorized representative.

THIS WARRANTY IS EXPRESSLY IN LIEU OF ALL OTHER WARRANTIES EXPRESSED OR IMPLIED OR STATUTORY INCLUDING THE WARRANTIES OF DESIGN, MERCHANTABILITY, OR FITNESS OR SUITABILITY FOR USE OR INTENDED PURPOSE AND OF ALL OTHER OBLIGATIONS OR LIABILITIES OF CCS. To any extent that this warranty cannot exclude or disclaim implied warranties, such warranties are limited to the duration of this express warranty or to any shorter time permitted by law.

CCS expressly disclaims any and all liability arising from the use and/or operation of its products sold in any and all applications not specifically recommended, tested, or certified by CCS, in writing. With respect to applications not specifically recommended, tested, or certified by CCS, the original purchaser acknowledges that he has examined the products to which this warranty attaches, and their specifications and descriptions, and is familiar with the operational characteristics thereof. The original purchaser has not relied upon the judgement or any representations of CCS as to the suitability of any CCS product and acknowledges that CCS has no knowledge of the intended use of its products. CCS EXPRESSLY DISCLAIMS ANY LIABILITY ARISING FROM THE USE AND/OR OPERATION OF ITS PRODUCTS, AND SHALL NOT BE LIABLE FOR ANY CONSEQUENTIAL OR INCIDENTAL OR COLLATERAL DAMAGES OR INJURY TO PERSONS OR PROPERTY.

CCS's obligations under this warranty are conditioned on the original purchaser's maintenance of explicit records which will accurately reflect operating conditions and maintenance preformed on CCS's products and establish the nature of any unsatisfactory condition of CCS's products. CCS, at its request, shall be given access to such records for substantiating warranty claims. No action may be brought for breach of any express or implied warranty after one (1) year from the expiration of this express warranty's applicable warranty period. CCS assumes no liability for any events which may arise from the use of technical information on the application of its products supplied by CCS. CCS makes no warranty whatsoever in respect to accessories or parts not supplied by CCS, or to the extent that any defect is attributable to any part not supplied by CCS.

CCS neither assumes nor authorizes any person other than a

duly authorized officer or representative to assume for CCS any other liability or extension or alteration of this warranty in connection with the sale or any shipment of CCS's products. Any such assumption of liability or modification of warranty must be in writing and signed by such duly authorized officer or representative to be enforceable. These warranties apply to the orginal purchaser only, and do not run to successors, assigns, or subsequent purchasers or owners; AS TO ALL PERSONS OR ENTITIES OTHER THAN THE ORIGINAL PURCHASER, CCS MAKES NO WARRANTIES WHATSOEVER, EXPRESS OR IMPLIED OR STATUTORY. The term "original purchaser" as used in this warranty shall be deemed to mean only that person to whom its product is originally sold by CCS.

Unless otherwise agreed, in writing, and except as may be necessary to comply with this warranty, CCS reserves the right to make changes in its products without any obligation to incorporate such changes in any product manufactured theretofore.

This warranty is limited to the terms stated herein. CCS disclaims all liability for incidental or consequential damages. Some states do not allow limitations on how long an implied warranty lasts and some do not allow the exclusion or limitation of incidental or consequential damages so the above limitations and exclusions may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

COMMENT SHEET

2810 Z-80 CPU MANUAL
89000-02810A

Any comments, criticisms, or suggestions you have will be appreciated.

Name: Position:
Company:
Address:

Publications • California Computer Systems
250 Caribbean Dr. • Sunnyvale, CA 94086