Z80MC Operating Manual

with ZMC v1.5 ROM

by Crash Kernigan *

Congratulations on your purchase and building of the Z80 Membership Card system! Four cards are presently available:

1. Main CPU card: Has a 4MHz Z80, RAM, ROM, I/O, and a voltage regulator. Can run stand-alone as a single-board computer. Has a bit-banged serial I/O port with TTL levels, 8-bit parallel input port, and 8-bit parallel output port. ROM contains monitor program and Micro BASIC by Dave Dunfield.

2. Front Panel card: Adds a 7-digit 7-segment LED display, 16-key keypad, piezo beeper, and 1 mSec timer interrupt. Optionally converts CPU card serial port TTL levels to RS-232 levels.

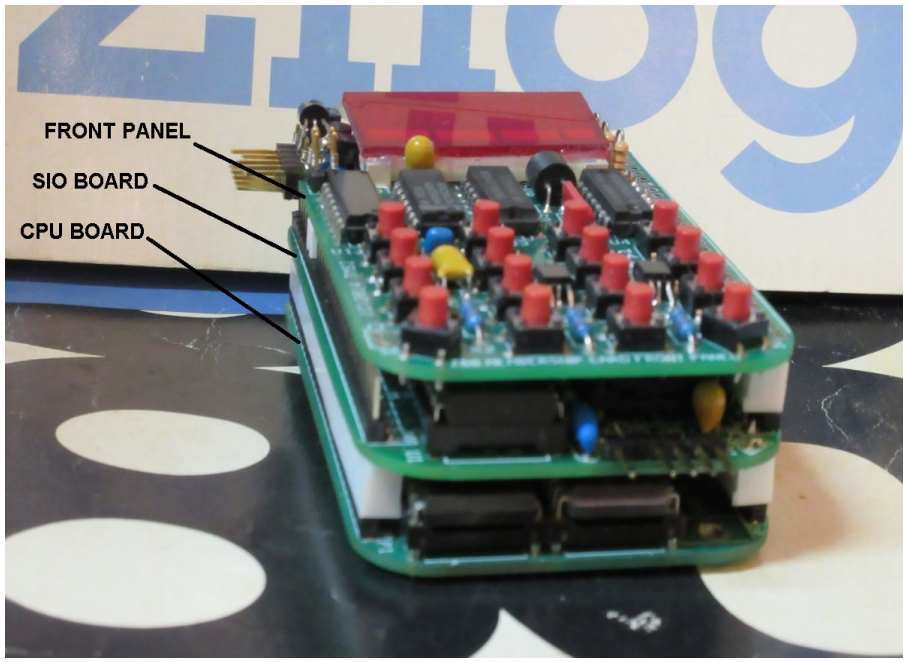
3. SIO card: Adds bank-switchable RAM (up to 512K). Allows the ROM on the Main CPU to be replaced with RAM to have an all-RAM memory map. Adds another serial port with a real UART. Adds a Micro SD-Card to simulate disk drives. Runs the CP/M-80 operating system.

4. Prototyping board: Has matching connectors, and an array of holes on 0.1" centers. Add whatever your imagination can conjure.

The boards stack on top of each another (no "motherboard" is needed). There is no special order, but the Front Panel (FP) card works best when it is on top. The CPU card is best at the bottom, because it lacks connectors on the bottom of the board. Other two cards are placed in between, like patties in a burger. If 8 bits make a byte, then 8 bytes should equal a burger!

Because of the multiple configurations, the ROM firmware tests to see which boards are installed. This information is stored as a byte in the System RAM area at location 0xFF95 (assembler label HW_LIST).

00 = No boards installed
01 = FP Board installed
02 = SIO Board installed
03 = SIO and FP Board



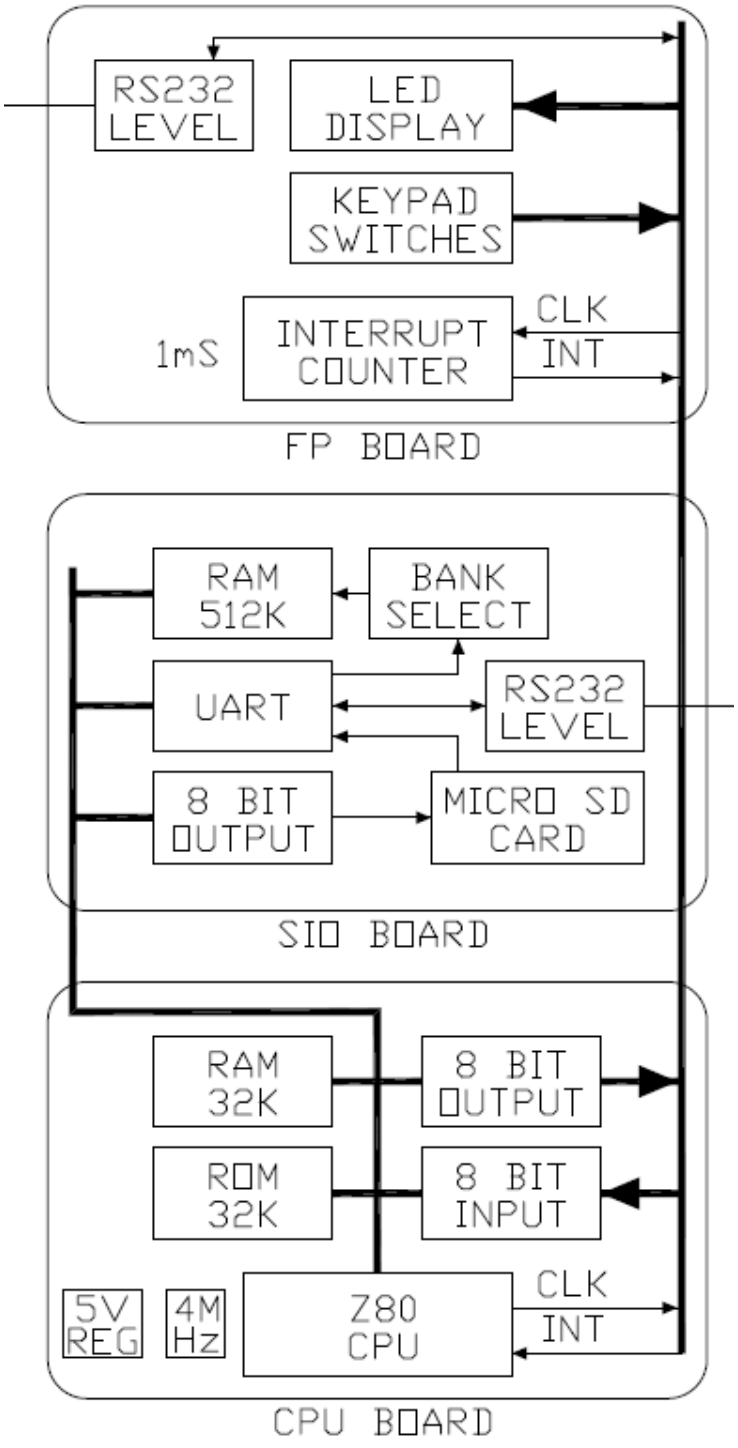
Power

The CPU Board has a 5V regulator. You can power the boards from an external 6-8v source, or a regulated 5V supply.

JP2 pin 10 is Common.
JP2 pin 19 is for +5V.
JP2 pin 20 is unregulated +5V to +8V power.

** OK, truth time. It's really written by Josh Bensadon and Lee Hart. Last update Feb 27, 2019*

Z80 System Block Diagram



Block Diagram of all the cards in the burger. The thick line on the left represents the Data, Address and Control busses. The thick line on the right represents the I/O and other signals intended for the FP (Front Panel). These signals pass through the middle board. The RS232 Level shifters are optional; either board can be configured through hardware for RS232 or TTL levels. The CPU card can only be TTL level.

Memory & I/O Map

Memory Map of Z80 MC system

	CPU BOARD	SIO BOARD	FP BOARD
FFFF 8000	32K RAM	NONE	NONE
7FFF 0000	32K ROM	512K RAM	NONE

I/O Map of Z80 MC system

	CPU BOARD	SIO BOARD	FP BOARD
FF D0	-	-	-
CF C8	-	ACE	-
C7 C0	-	BIT LATCH	-
BF 60	-	-	-
5F 40	8 IN 8 OUT	-	-
3F 00	-	-	-

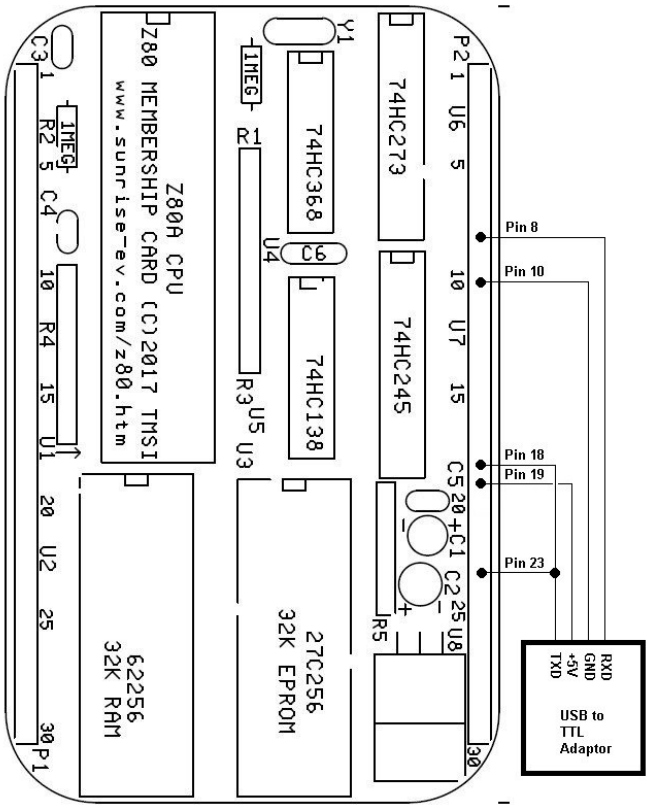
The LED and Keypad are on the FP card (Front Panel), but the I/O ports that control them are on the CPU card. The Front Panel also produces timer interrupts. Both FP and SIO cards provide a com port and a power connector.

The com port can be configured as TTL or RS-232. The SIO card may have up to 512K of RAM, it accepts three sizes; a 32K, 128K or 512K RAM chip. Only 32K at a time can be accessed in a bank switching configuration. The lower 32K of memory space can either be CPU ROM or SIO RAM (bank of 32K). Upon Reset, the CPU ROM / firmware is selected to operate the system.

CPU Card Only

A good place to start is at the bottom. Alone, the CPU board has everything an embedded computer needs. It can hook up directly to a USB-Serial adapter for communications and power. Configure the terminal program to 9600, N81 and the Z80MC will prompt you with a menu. That is to say, a working CPU board will unconditionally send a "Hello world" message through output port bit 8 on power up or reset. If this message is not coming, then something is wrong. A USB-Serial adaptor with TTL levels makes it easy to power & connect. See Appendix A for help in using USB-Serial adapters.

Connect:
Pin10 to Ground (0V)
Pin19 to +5V
Pin8 to Rx Data
Pin18 and Pin23 to Tx Data.



Running the CPU card alone is a great place to start troubleshooting your system. At power-up or reset, Pin OUT7 should send a burst of serial data. The "usual suspects" are poor solder joints (shorts or opens), part in backwards or in the wrong places, etc. Then check for power, clock, reset, activity on the Z80 control signals, and data and address buses. If that doesn't do it, look for other trouble shooting help from Lee Hart (leeahart@earthlink.net).

Please see the next section for information on the Monitor. When running the Z80MC CPU board alone, the I/O for the Front Panel LED and Keypad will not be scanned (with the exception of the hardware test on boot up). The input port at 0x40 can be freely read. The output port at 0x40 can be written to, but it will be overwritten by any Serial I/O activity with the byte value stored at memory location 0xFFF5 (assembler label SCAN_LED). To use both serial data out and the I/O port, it is advised to write to both the port and this memory address. Set bit 7 to avoid corrupting the serial output. I.e.

```
OR   0x80
OUT (0x40), A
LD (0xFFF5), A
```

Without the front panel board, there are no timer interrupts. Features like the Tic Counter will not run, and tests for <Ctrl>-C or HALT instructions will not happen. Any program that runs will have full control of the system. A return to the monitor can only happen if the program jumps there or Reset is applied.

Micro-BASIC by Dave Dunfield is included in the ROM (with his permission of course). The manual for it is at <http://www.classiccmp.org/dunfield/altair/d/basic.txt>. On power-up or Reset, if no other boards are detected, the firmware will provide an option to run the Monitor. If the Monitor is not selected after a few seconds, it will then enter BASIC and run any program in memory. The first time you start BASIC, there will not be a BASIC program in RAM and the RUN command will generate an error message. This is normal and can just be ignored.

BASIC has been altered a little to include 2 new instructions and the ability to autorun an existing BASIC program in RAM. By using a NVRAM (i.e. ST M48Z35, digikey.com #497-2885-5-ND), a program can be entered into memory and will remain there even without power, so it will automatically run on reset or power-up. Dave's manual on Micro-BASIC covers it all.

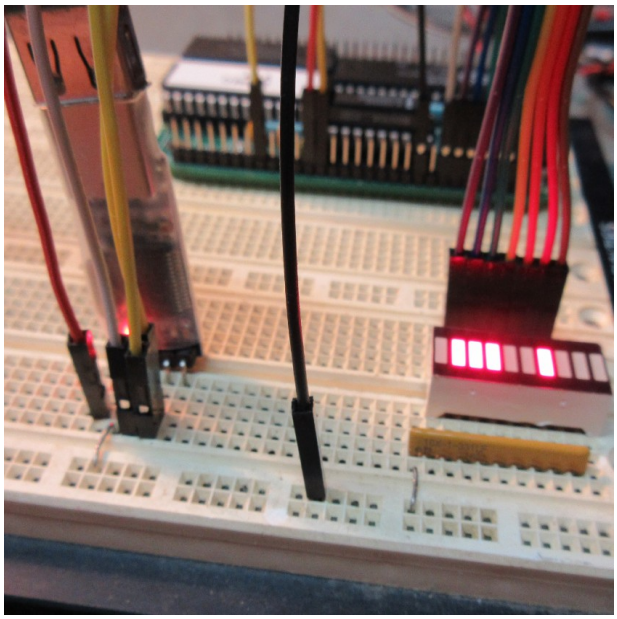
The only exceptions to his manual that don't apply here are the Memory allocation(s) and the UART. And the manual does not cover the 2 new instructions which are:

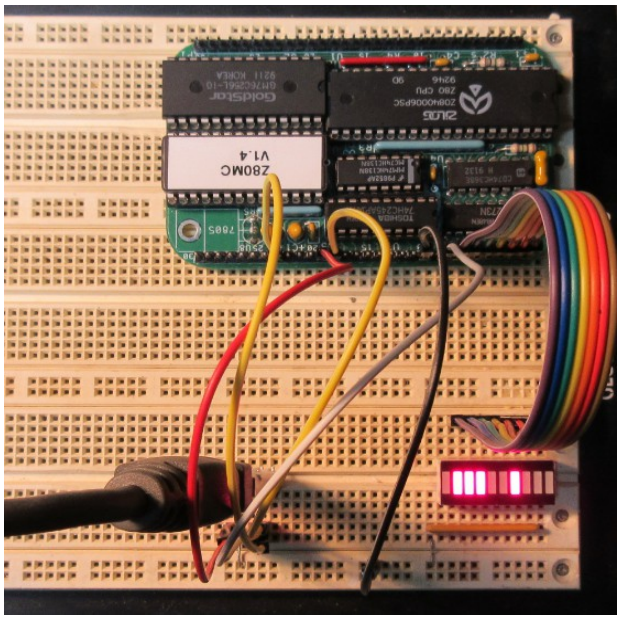
```
OUT p, v
IN(p)
```

Where *p* is a port number and *v* is a value to be written to that port. OUT is a statement and works on its own. IN is a function and must be used within a statement, i.e.

```
PRINT   IN(64)
LET   A = IN(I + 5)
OUT  64, IN(64)
```

The following pictures show the CPU board connected alone. A single line program was entered and run to produce output on the LED bar graph.



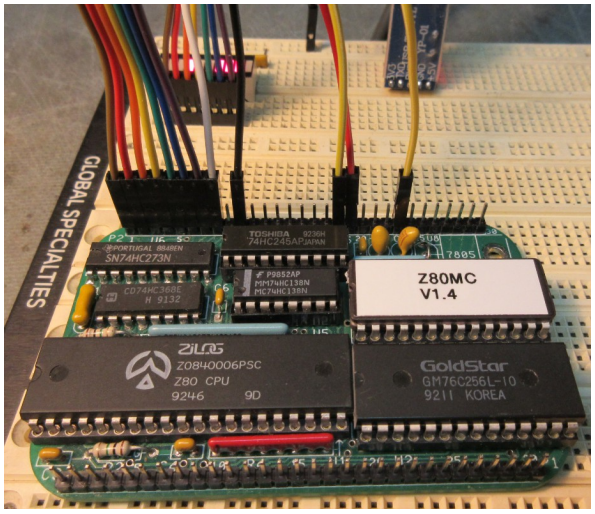


The device in the lower left of the above photos is the USB-Serial Adaptor. The wiring to these devices is not standardized; every one is different.

```
 MICRO-BASIC COPYRIGHT 1983 BY DAVE DUNFIELD
READY
?NO PROGRAM ERROR
READY
10 OUT 40#,F2#
RUN
READY
```

Serial Port output captured on terminal. The OUT command sends F2 (hex) to output port 40 (hex), which is the output port on the CPU card

F4 (hex) = 1111 0110 binary. Note that since the most significant bit is used for the bit-bangerd serial output, it is not displayed.



You could also connect a DIP switch or other TTL-level input devices to the CPU card's input port, and read them with the BASIC IN(40#) command. Here too, the most significant bit of this port is being used for the serial input.

CPU + FRONT PANEL Cards

The Front Panel gives a nice view and control of the system. It is completely software driven and uses the I/O ports on the CPU board. The LED display consists of 7 digits (d1 to d7) and 7 annunciator LED's (x1 to x7). Upon power on, the display shows "DELAY" for a moment.

Next it will enter Monitor mode (or try to boot from an SD card, if the SIO card is present). The system can be in two modes; *monitor mode* and *run mode*. Booting to CP/M from the SD card is considered *run mode*.

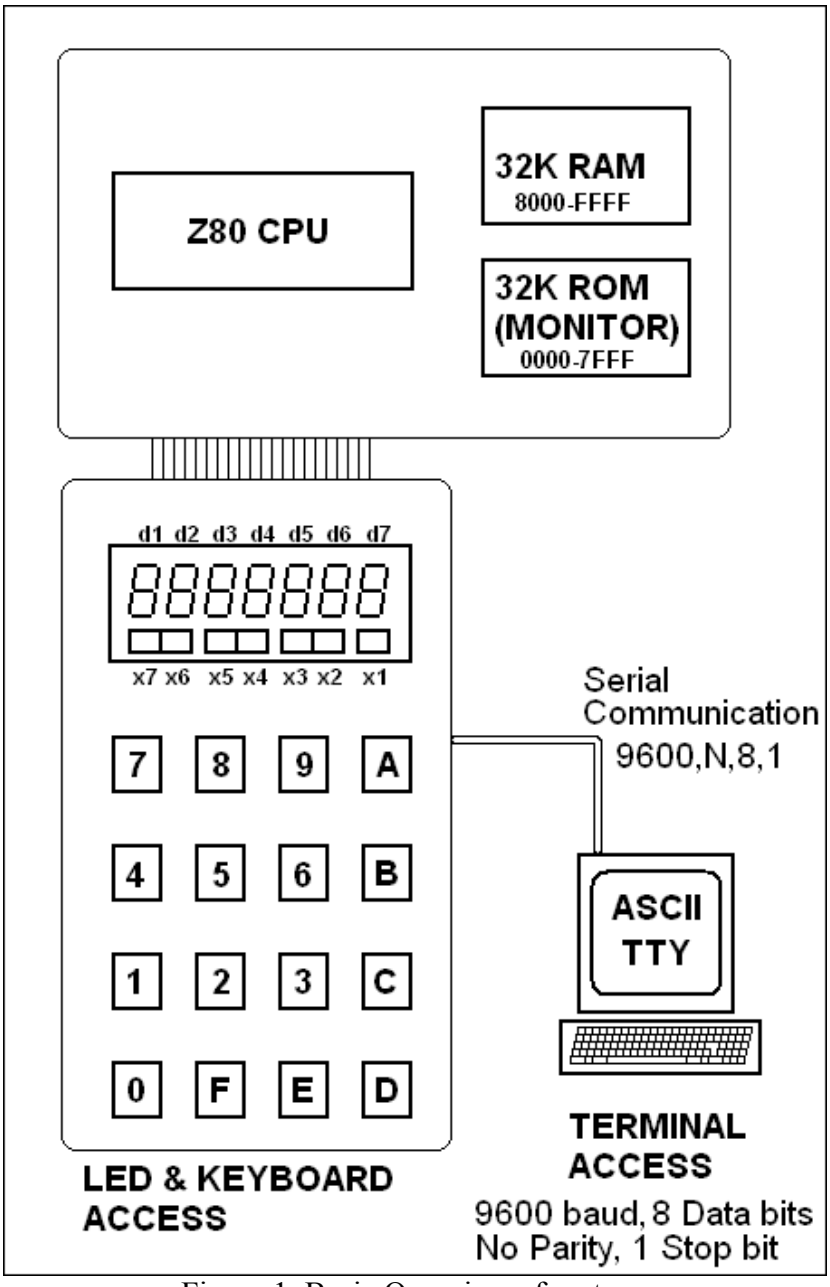


Figure 1. Basic Overview of system.

MONITOR mode vs RUN mode

On power-up or Reset of the Z80, the system will execute the Monitor program in ROM. The monitor allows you to control the system. You can examine and/or modify (alter) the RAM, Z80 registers, or I/O ports.

After you have entered or uploaded a Z80 program, you can switch to *run mode* and have the system execute the Z80 program in RAM. During *monitor mode*, there are two ways to access the system: Keyboard Access (through the front panel LED & Keyboard) and Terminal Access (through a CRT & Keyboard Terminal connected via RS-232 serial communication).

Keyboard Access was written into the Interrupt Service Routine (ISR), and so can still operate even when the user has switched to *run mode*. The Keyboard Access Monitor can be turned off, if desired, by the executing program.

Terminal Access was written as a foreground (main) program, so it stops working in *run mode*. There are several ways to switch back and forth between *monitor mode* and *run mode*, which will be explained later through each access method.

Keyboard & LED Access

In the ISR (which happens every 1millisecond), the multiplexed LED display is being refreshed and the keyboard is read and debounced. The keys can invoke commands that will alter the LED display and/or the Z80 system. There are nine (9) keyboard commands. Pressing a key invokes that command. Most commands expect parameter(s), to be entered after starting the command. You need to enter all keys with less than 3 seconds between each key, or else the command will time out.

There are also two (2) keyboard functions to reset the Z80 system. A nice feature of keyboard access is the ability to operate even while the Z80 executes your user code. It is calculated to take about 12% of the CPU time. Details of this calculation are in the firmware source code as comments in the ISR chapter, prior to any actual ISR code.

Any key pressed for longer than 2 seconds will be repeated (auto repeat). The F key operates a little differently than the rest of the keys. If you press and hold it, it acts as a “Shift” key for any other key pressed. If you release it without pressing any other key, it simply acts as an “F” key.

By default, the display and keyboard will operate the Keyboard Access monitor. The display and keyboard can be utilized by the user's own program; but the user's program would need to disable the LED Display updates and Keyboard command table.

The display will show any of the following:

- The startup mode (e.g. “COLd 00”)
- A memory address, and contents updated in real time (eg. “FFF1 xx”)
- A register name, and contents updated in real time (eg. “HL xxxx”)
- An input port, and contents updated in real time (eg. “inPP xx”)
- An output port, and its commanded value (eg “ouPP xx”)
- The ROM or RAM bank currently selected. (eg “ROM”)

Annunciator LEDs

(right) x1 – Beeper, flashes on for a short interval when any key pressed.
x2 – Run Mode
x3 – Monitor Mode
x4 – Sending data to Output Port
x5 – Modifying Memory or Register
x6 – Viewing Memory or Input
(left) x7 – Viewing Register

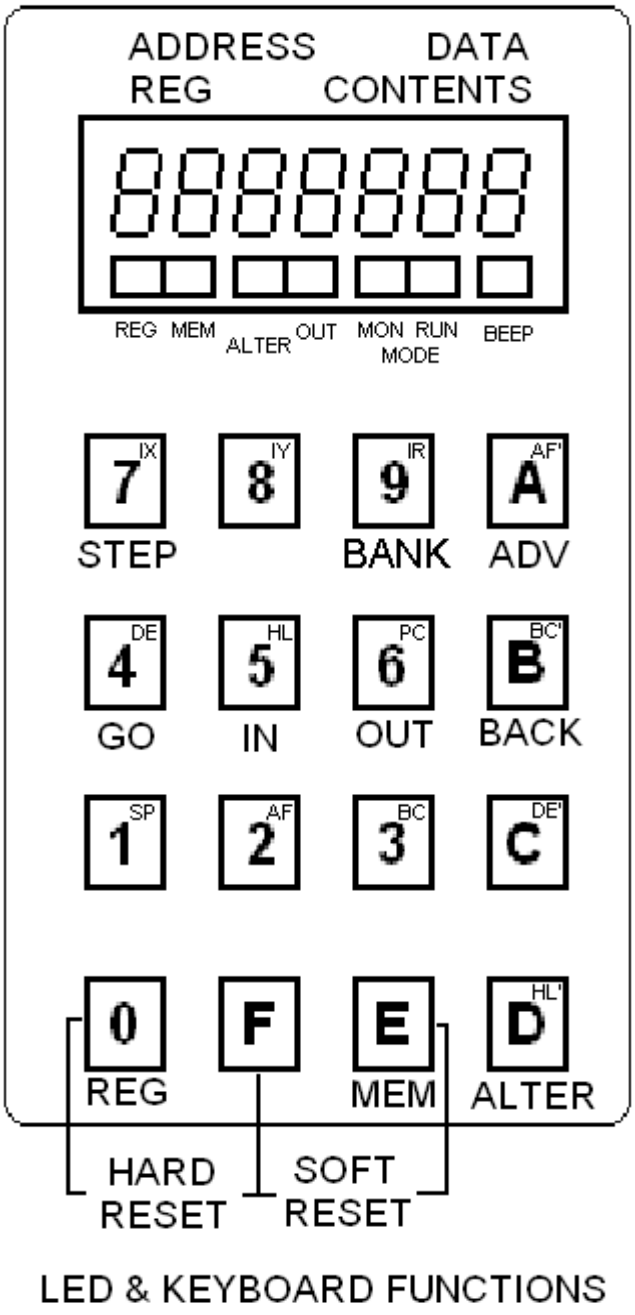


Figure 2. A mapping of keys and display.

Examine RAM

- Press ***E***, LED x6 turns on to indicate Examine Command waiting for Address
- Enter the address by pressing four hex digits, try FFF1 (Tic Counter)
- Address is shown on d1-d4, contents are shown on d6,d7. LED x6 turns off.

Next/Previous RAM

- Press ***A*** to examine the next RAM location
- Press ***B*** to examine the previous RAM location

Think of them as Advance and Backup.

Modify RAM

- First Examine the RAM location
- Press ***D***, both LED x6 and x5 will turn on to indicate waiting for Data
- Enter data by pressing two hex digits.
- The Address will automatically advance to the next location so you may enter more data for all consecutive addresses.
- To end the data entry, wait 3 seconds for the command to time out. LED's x6 and x5 will turn off.

RAM is located at hex address 8000 to FFFF. However, the firmware uses address FE00 to FFFF for the monitor system. Avoid writing to these locations.

Examine Registers

- Press ***0***, LED x1 turns on to indicate Examine Command waiting for Register
- Press any key ***1*** to ***D*** to select register, x6 will turn off
  - 1 = SP
  - 2 = AF
  - 3 = BC
  - 4 = DE
  - 5 = HL
  - 6 = PC
  - 7 = IX
  - 8 = IY
  - 9 = IR
  - A = AF'
  - B = BC'
  - C = DE'
  - D = HL'

Next/Previous Register

- Press ***A*** to examine the next Register
- Press ***B*** to examine the previous Register

Commands ***A*** or ***B*** will do the next/previous RAM, Register or I/O, depending on the last examine (or I/O port) command given.

Modify the Registers

- First Examine the Register
- Press ***D***, both LED x6 and x5 will turn on to indicate waiting for Data
- Enter data by pressing four hex digits. LED's x6 and x5 will turn off.

There are differences in accessing the registers if the system is in *monitor mode* vs *run mode*. In *monitor mode*, the values you view & change are really in RAM. These values get transferred to the registers when you switch to *run mode*. While in run mode, the actual value of the registers will be fetched and/or changed. Only the SP cannot be changed. However, the PC can be changed and it will force the program to continue executing at the new PC location. When *monitor mode* is re-entered, the registers are saved back to RAM to be examined and changed as desired. See figure 3.

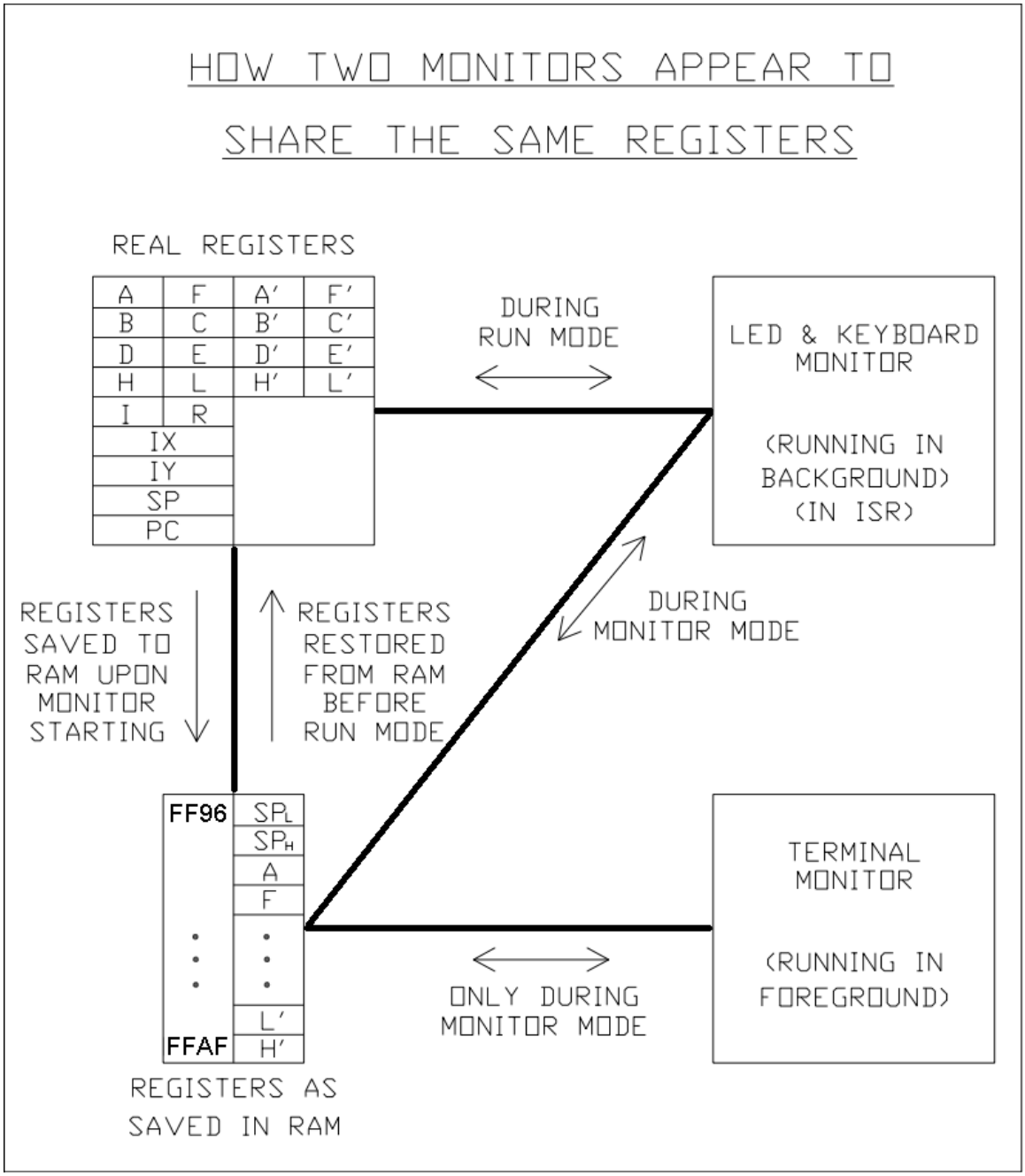


Figure 3. This is how the two monitor programs view/modify the registers.

RUN mode

This will switch the system from monitor mode to run mode and start executing your Z80 Code at the location specified in the PC register.

- Set PC Register to desired starting location.
- Press ***4.*** LED x3 turns off, LED x2 turns ON.

Return to Monitor mode (or Reset)

These are the manual or software ways that will switch from run mode to monitor mode and to stop executing user code and restart Terminal Access monitor.

- Pressing ***F*** and ***E*** keys – Soft Break detected by the keyboard routine in the ISR
- Pressing ***F*** and ***0*** keys – Hard Reset, electrically activates the Z80 Reset Input.
- Executing a HALT instruction – Detected by the ISR
- Executing a RST0 instruction – Starts the monitor by running it at $0000, since the address will be on the stack, the PC will be saved.
- Any form of JMP to $0000 – Similar to RST0, but PC cannot be saved, that is to say it will be whatever number is on the top of the stack, which is not meaningful.

Single Step

- Press ***7.*** The program will execute 1 instruction and return to monitor mode, displaying "StEP ##".

The single step operation is worth explaining. It uses a very clever idea thought of by Lee Hart. Instead of temporarily replacing the 2nd instruction with a "Return to monitor mode", it was devised that the ISR should wait X number of cycles before executing the next instruction. X was calculated such that there would be only time for 1 instruction to execute before the assertion of the next timer interrupt cycle (which occurs every 1024uSec or 4096 machine cycles). In this manner, no code modification is required and single stepping can work with code in ROM as well as in RAM.

Reading an Input Port

- Press ***5***, display reads "inPP xx", x6 turns on while waiting for port number.
- Enter the port (PP) by pressing two hex digits.
- Input is continuously read and displayed on digits d6 and d7

Writing to an Output Port

- Press ***6.*** Display reads "ouPP xx". x6 turns on while waiting for port number.
- Enter the port (PP) by pressing two hex digits. x4 turns on while waiting for output data.
- Enter output data by pressing two more hex digits, they will be shown on digits d6 and d7
- Different output data can be entered if done before the time out. x6 & x4 turn off.

Monitor Start Code

Every time the monitor starts, it saves the registers and tries to determine the previous condition of the system or the cause of the monitor (re)start, then displays this along with the count of times it occurred. Table 1 shows a list of possible conditions.

Cause	Test	LED Msg	Registers Saved
Powering on, or a runaway program	RAM signature not valid.	COLd 00	SP and PC are defaulted. System RAM is initialized (all counts set to 00).
Press F & E Keys	Flag = FE	SoFt ##	All Registers Saved
Single Step	Flag = D1	StEP ##	All Registers Saved
Control – C	Flag = CC	^C ##	All Registers Saved
Halt Instruction	Flag = 76	HALt ##	All Registers Saved
Press F & 0 Keys	F or 0 Key down	F-0 ##	PC will not be correct.
RST 0 Instruction	M(PC) = C7	RSt0 ##	All Registers Saved
RESET Line or JMP 0000	Default Case	Hard ##	PC will not be correct.

Table 1. List of Monitor Start Codes. The "##" is the count of times for this message.

This message is intended to help you use this system by giving you the status of the monitor re-entry. The PC cannot just be copied from the PC register because its value will be what ever Monitor Instruction is currently running. The RST0 instruction was designed to save the PC on the stack, so that is from where the PC register is taken from. The other software initiated resets are from within the ISR and the PC is on the stack already. Any hard reset (like ***F*** & ***0***) or a JMP to 0000 do not place the PC on the stack so the value fetched from the top of the stack will NOT be correct.

When everything is working and you turn it on for the first time, the display will show "DELAY" for a few seconds, then "COLd 00". This message is meant to tell you that the system has entered the monitor code from a "cold boot". This is determined by checking the RAM for a written pattern or signature. This signature will not be there if the system was just powered on.

It is also possible that a user program may have gone out of control and overwritten the signature, but you'll recognize this condition by the pulling of your hair out. During run mode, the Terminal Monitor is no longer running, however, there is a "<Ctrl>-C" check that happens in the ISR which will allow the system to switch back to monitor mode by pressing <Ctrl>-C a few times from the Terminal. :-)

The initial delay was added to prevent the Z80MC monitor from outputting serial data for a short time. Without this delay, it was found that Microsoft Windows may confuse the Z80MC menu as a Microsoft Mouse! This happened to my system when plugging in the USB Serial adaptor.

Terminal Access

The Terminal Monitor will improve your productivity with the Z80 Membership Card by giving you a friendly interface on any terminal that communicates at 9600 baud, No Parity, 8 data bits, 1 stop bit. You are free to use any terminal or terminal program you like. The electrical cable connections for this interface are shown in the "Kit Assembly Manual". The screen captures used here are from Hyper Terminal version 5.1 which is included with Windows XP. Please search the internet for instructions on how to use this terminal software. Tera Term and PuTTY are other good terminal programs that can be used. The system offers a full duplex bit-banged com port. The receiver has a 256 byte buffer to allow entire packets or lines of data to be received for processing without the need for character pacing (delay).

The opening screen when powering on the Z80 MC will display:

```
Cold Start

Z80 MEMBERSHIP CARD MICRO-SD, v1.5 beta July 23, 2017

Main Menu >
```

Displaying The Help Screen

Press either ***<ENTER>*** or ***?***

```
HELP
D XXXX YYYY    Dump memory from XXXX to YYYY
C XXXX YYYY    Continuous Dump (no pause)
E XXXX         Edit memory starting at XXXX
M XXXX YY..YY  Enter many bytes into memory at XXXX
G [XXXX]       GO (PC Optional)
W              Single Step
I XX           Input from I/O
O XX YY        Output to I/O
R rr [=xx]     Register
L              Loop back test
T XX YY        RAM TEST from pages XX to YY
V              Version
H              UPLOAD Intel HEX file
X U XXXX       XMODEM Upload to memory at XXXX
X D XXXX CCCC  XMODEM Download from XXXX for CCCC #of 128 byte blocks
P              Port Speed (ACE Only)
Y              Identify Port
U              Use ROM or RAM BANK
S              SD CARD BOOT
B              BASIC By DAVE DUNFIELD

Main Menu>
```

Dump Memory

The start/end memory locations can be entered as any number of hex digits. Only the last 4 hex digits are interpreted. The memory dump will start at whatever byte is selected, but will align the display in even multiples of 16 bytes. They say a picture is worth a thousand words, but I believe the following picture is only worth a few bytes. The same data is shown in ASCII format on the right side; printable ASCII characters are displayed, and unprintable characters are replaced with dots. The screen will pause before the start of the next page. In the second example "d 2200230 23f", only the last 4 digits of the starting parameter was interpreted ie. "0230".

```
Main Menu >d a  21
M000A                                 FF FF FF FF FF FF ;           ......
M0010  C9 FF FF FF FF FF FF FF C9 FF FF FF FF FF FF FF ; ................
M0020  C9 FF                                           ; ..

Main Menu >d 2200230  23f
M0230  0D 0A 43 6F 6C 64 20 53 74 61 72 74 0D 0A 00 0D ; ..Cold Start....
Main Menu >
```

Continuous Dump

This is same as Dump Memory, but does not pause for each page. This allows you to do an ASCII download or capture of memory. A captured text file can be transferred back into memory by having your terminal program "type" or ASCII upload the file. The "M" that starts each text line starts the "Enter Many bytes" command. You may need to experiment with line or character pacing; see what options your terminal program has. See Appendix B regarding USB Serial Ports and line pacing.

Enter Memory

The display will show the previous memory contents, waits for you to enter a new value and then writes this value and reads it back for confirmation. Press ***<Esc>*** when done.

```
Main Menu >e 8000
8000 : FF c3 C3
8001 : FF 00 00
8002 : FF 80 80
8003 : FF
```

Enter Memory Continuous

The M command is written to match the ASCII output from the "**Continuous Dump"** command. It expects the address after the M, accepts any number of white space, ignores the semicolon and all characters following it. In anticipation of consecutive "M" command lines, the monitor will not respond to each line with the "Main Menu >" prompt.

Go (Execute)

Start the program at the PC or memory location if provided. This will enter Run Mode, and as stated earlier, the Terminal Monitor will stop running (with the exception of <Ctrl>-C checking). Example shows <Ctrl>-C response to the "JMP 8000" program.

```
Main Menu >d 8000  800F
M8000  C3 00 80 FF FF FF FF FF FF FF FF FF FF FF FF ; ................
Main Menu >g 8000 PC=8000
<Ctrl>-C 02
AF=0045  BC=0000  DE=D800  HL=2000  AF'=BFBD  BC'=BDFF  DE'=FFFF  HL'=FDBF
IX=AFFF  IY=FFFF  IR=0076  PC=8000  SP=FF5A
Main Menu >
```

Single Step

Allows execution of a single instruction at the current PC.

```
Main Menu >W
Step 01
AF=0045  BC=0000  DE=D800  HL=2000  AF'=BFBD  BC'=BDFF  DE'=FFFF  HL'=FDBF
IX=AFFF  IY=FFFF  IR=0076  PC=8000  SP=FF5A
Main Menu >
```

Input/Output data to I/O ports

The port input command expects an I/O address from 0 through FF, and responds with the value seen by the Z80. Unused input ports will show 00. Avoid input ports 40 to 5F, which all map to the port that is used to read the Front Panel.

The port output command expects an I/O address followed by a byte data value. There's no response. Avoid the I/O ports 40 to 5F, which is used to write to the Front Panel.

```
Main Menu >I FF 00
Main Menu >O FF 11
Main Menu >I FF 00
Main Menu >
```

Display/Change Register

Entering ***R*** alone will dump all registers. Specifying a register will dump only that register, use "=" to set its value. Register names are case sensitive (must be upper case).

```
Main Menu >r
AF=FFFF  BC=BFBD  DE=FFFF  HL=FFFF  AF'=BFBD  BC'=BDFF  DE'=FFFF  HL'=FDBF
IX=AFFF  IY=FFFF  IR=001B  PC=8000  SP=FF5E
Main Menu >r b?
Main Menu >R BC'=1234=1234
Main Menu >R
AF=FFFF  BC=BFBD  DE=FFFF  HL=FFFF  AF'=BFBD  BC'=1234  DE'=FFFF  HL'=FDBF
IX=AFFF  IY
```

Loop back test

Simply echoes the characters back to the Terminal and to the LED Display.
<Enter> will restart a new line, use <Esc> to quit

```
Main Menu >L
ELF1802
```

RAM Test

This tests RAM for correct addressing and data across multiple 256-byte pages. Details of the RAM test are in the source code. It is suggested that you run this test as part of your system testing after building the unit. Then you can test again as needed. Note: You cannot test pages FA-FF, because they are being used by the system.

```
Main Menu >T 80 F9
TESTING RAM
RAM PAGE MARCH PASSED
RAM BYTE MARCH 1 PASSED
RAM BYTE MARCH 2 PASSED
RAM BIT MARCH PASSED
RAM SEQUENCE TEST PASSED
```

Version

This command will display the same version information shown upon system boot up.

XMODEM Upload and Download

Here's the fastest way to save and restore your work. This XMODEM auto-detects and works with Checksum or CRC error checking. The download command transfers data from the Z80MC to a file on your computer. Uploading sends data from your computer to the Z80MC. In other words, "Downloading" saves data from, and "Uploading" loads data to the Z80MC.

Download command is "x d" followed by the start address and count of 128 byte blocks. Upload command is "x u" followed by only the start address. It uploads all the blocks sent in the file. To cancel a command, <Ctrl>-X a few times or wait 2 minutes for the command to time out.

The example shown "x d 0 100" saves a binary image of the EPROM from 0000 to 7FFF. You can save RAM from 8000 to 7DFF with "x d 8000 FE". Do not download or upload pages FExx and FFxx because they are used by the system. Uploading to them will crash the stack!

```
Main Menu >x d 0  100                          TRANSFER COMPLETE
```

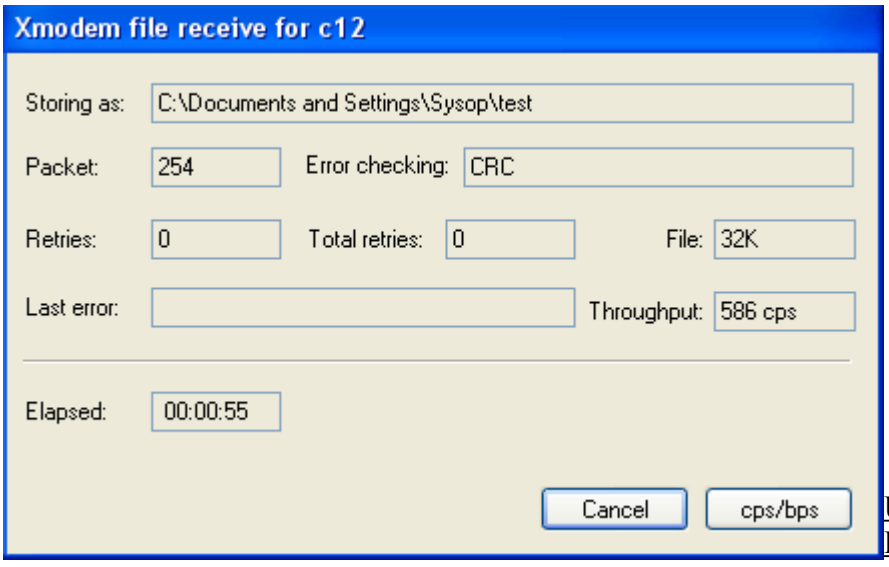


Figure 3. Snapshot of a 256-packet transfer with 2 more packets to go at 55 seconds elapsed time. This is a screen capture is from Windows Hyper Terminal. First enter the "x d 0 100" command to the Z80 MC. Within 2 minutes (before the command times out), click *Transfer*, *Receive File*. Select *Xmodem* from drop list, click Receive. Do not use *1K Xmodem*.

Uploading Intel HEX Format Files

This is the easiest way to upload your program. Just do an ASCII Upload of an Intel HEX file. Enter H. Then in Hyper Terminal, click "Transfer", "Send Text File". HEX files usually end with ":00000001FF", but if this line is missing or the transfer is canceled, the receive routine will time out after 10 seconds or upon reception of an <Esc> character. The monitor will report # of lines read without errors. The transfer checks for valid hex digits, correct pairing of digits and checksum errors. Any errors will cancel the upload and report the error.

```
Main Menu >H
WAITING FOR HEX TRANSFER
HEX TRANSFER COMPLETE, LINES=004A

Main Menu >
```

CPU + FRONT PANEL + SIO Cards

The SIO board adds more RAM, a real com port, and a Micro SD Card interface. These features make it possible to run the classic CP/M-80 operating system.

Additional communications serial port

An 8250 or 16C450 UART (aka Asynchronous Communications Element or ACE) provides a hardware com port. It can use the Z80 CPU interrupt, but the current software does not use this feature. The system's bit-banged com port uses interrupts, so it was necessary to minimize the sources of interrupts to allow time for it to work.

The firmware uses both comports equally upon start-up. The Menu output to both ports, and data received on either port will drive the menu interface. The ports are named "BIT" for the bit banger, and "ACE" for the UART. There are 5 new Terminal Monitor commands.

1 – BIT port only Serial data to/from ACE will be ignored by terminal program.
2 – ACE port only Serial data to/from FP port BIT will be ignored.
3 – BOTH ports The default upon cold boot.

BIT Port	ACE Port

`Main Menu>1`
`BIT`
 	
`Main Menu>`
	`Main Menu>1`

`Main Menu>2`	`Main Menu>2`
	`ACE`
	
	`Main Menu>`

`Main Menu>3`
`Both`
 	
`Main Menu>`
	`Main Menu>3`
	`Both`
	
	`Main Menu>`

Y – Identify Port Displays the port that is connected.

`Main Menu>Y`
`BIT`
 	
`Main Menu>`
	`Main Menu>Y`
	`ACE`
	
	`Main Menu>`

P – Port Speed (ACE Only)

The current value is displayed, then you may enter the new value. Enter the baud rate divisor desired. You may enter a new hex number (nn) from 00 to 0xFF. Some useful values are:

0C – 9600 baud
06 – 19,200 baud
03 – 38,400 baud

```
Main Menu >P
(0C=9600) BAUD:0C nn
```

Additional RAM

The Z80-SIO card's RAM chip can be 32K, 128K or 512K bytes. This RAM is controlled by a new Terminal Monitor command, and a new Keypad Monitor key.

When enabled, this RAM is mapped into the lower 32K of Z80 memory (replacing the CPU board's EPROM). One I/O bit selects RAM or EPROM at 0-32k (power-up and reset default to the EPROM). Another four I/O bits select which 32K bank of the SIO RAM to use. Bank switching a 32K chip has no effect. A 128K chip will have 4 banks (bit 2 of bank must be low for chip select), and a 512K will have 16 banks. Writes always go directly to the SIO RAM Bank, even when the EPROM is enabled. The system switches to Bank 0 RAM to run CP/M. (The other banks are not used for now).

U – Use ROM or RAM BANK

The current memory and bank is displayed, then you may enter a new value. You may enter any hex number from 0 to F or R for ROM.

```
Main Menu >U
Currently using: ROM
Enter 0-F,R
```

Bank selection can also be done from the Keypad Monitor. Press key 9 to cycle through the 16 banks and the ROM.

This command changes what reads and writes do in the lower 32K address space. Selecting ROM shows the ROM contents, but RAM will still be written to by Memory Write functions! This allows initializing the RAM even with ROM selected.

When using 128K RAM chips, the Bank number must be 0-3 so that bit 2=0. (The hardware inverts the Bank bits, so setting bit 2=0 correctly sets CE2=1 high for a 128k RAM chip.)

Bank 0 is selected upon cold boot to ensure RAM can be accessed.

	32K	128K	512K
BANK $\overline{b_0}$	n.c.	A15	A15
BANK $\overline{b_1}$	n.c.	A16	A16
BANK $\overline{b_2}$	n.c.	CE2	A17
BANK $\overline{b_3}$	n.c.	n.c.	A18

Micro SD Card

Through I/O, data can be read or written to the Micro SD Card. This is using the public SD-SPI interface. SPI is bit-banged by the Z80. Since SPI is synchronous, there is no critical timing and the Z80 is free to receive data and timer interrupts during any SPI interfacing. Presently the terminal functions only read from the SD Card. The card must be formatted with FAT-16 and then the files in the root directory can be listed and any Intel Hex formatted file can be opened and loaded to memory. Sub directories/folders are not supported. Creating or appending to a FAT-16 file is not supported. Only writing to an existing fixed length file is possible. The LED/Keypad Monitor will remain functional after loading an SD Card File.

S – SD Card Boot

The SD Card is initialized, and tested for the correct type before further access is made. This is to prevent any possible corruption of an SD Card.

```
Main Menu>S

Init SD Type#2 ACMD41+MBR S Type06 PBR S VOL=NO NAME     SYS=FAT16

L              LIST FILES
W              WITNESS LOAD
? >
```

W – WITNESS LOAD

This command just sets a flag that will display the hex file as it's being loaded. At the end of the load, a prompt will ask to execute the loaded code. You may choose to not execute and simply return to the menu system.

L – LIST FILES

This will display all the files in the root directory. You are then prompted for the file name of a hex file you wish to load into memory (and execute by default). The files must not have more than 8 letters for the name and 3 letters for the extension.

```
? >w -ON
L              LIST FILES
W              WITNESS LOAD
? >l
DIRECTORY:
DISK-A.BIN      DISK-B.BIN      DISK-C.BIN      CPMDISKS.TXT    Z80MC_GO.HEX
DISK-D.BIN
ENTER 8.3 FILE NAME> z80mc_go.hex
Z80MC_GO.HEX -EXISTS FILE SIZE=0x00000D2C
:20F40000CD0EF60D0A5A38304D432043502F4D204C4F414445522E202041707220322C207D
:20F42000323031370D0A00C354F40043504D4449534B535458540000FFFFFFFFFFFFFFFF8A
…
:00000001FF
Execute?y
Execute at:F400
```

Files can be loaded in the lower 32K. Memory writes will always go to RAM, even when ROM is selected as the view/read source. Execute will switch to RAM before jumping to the first address loaded in the HEX file.

Changing the start address of a HEX file loaded.

The start address for execution can be changed by either turning on the Witness Load, then decline the "Execute?", and issue your own execute command G as previously described. Or, an Intel HEX file is simply an ASCII file, so you can add your own first line to change the starting address. For example:

:0112340000B9

The data "00" will be later overwritten by data in following lines in the Intel HEX file. Address 0x1234 will become the start address. The checksum value B9 was calculated in hex as follows:

01+12+34+00+00 = 47. Then subtract from 100. 100 – 47 = B9.

Auto Boot the SD Card program

On a Cold or Hard Reset, the system will look for, and if found automatically load and execute a file called "**Z80MC_GO.HEX**" from the SD Card. The option to get to the regular monitor menu will be provided for a short time while the system slowly prints dots.

CP/M

The SD Card files come with 4 files called Disk-A.BIN to Disk-D.BIN, a file called CPMDISKS.TXT and the Z80MC_GO.HEX file needed to load CP/M. The Z80MC_GO program has two significant parts.

Part 1 (located at 0xF400) CP/M Loader

- Reads the CPMDISKS.txt file for the default files to be used for the 4 virtual disk drives. These are the first four lines of the that file.
- Checks for the presence of each disk file, checks its size and finds the matching size in the CPMDISKS.txt file. The CPM DPB in the BIOS section is then updated with the correct parameters for that size of virtual disk.
- Optionally allows selection of a different file for each virtual disk drive. Press 1 to 4.
- Boot CP/M when ready. This will cause a jump to the cold boot entry in the BIOS section (part 2).

Part 2 (located at 0xEE00) CP/M BIOS

- This is the location of the CP/M Jump Table. The cold and warm boot routines will then read 44 sectors from the DISK-A.BIN file. These will be sectors 2 to 45, which skips the first "boot" sector and the last "bios" sectors.
- Drive Parameter Block (DPB). Holds the attributes that govern how CP/M will read each drive.
- BIOS routines as pointed to by the Jump Table.

If this file gets loaded by the Auto Boot of the firmware, this loader program will in turn automatically run CP/M (cold boot loader).

CP/M Loader file - Z80MC_GO.HEX

```
Z80MC CP/M LOADER.  Apr 2, 2017
CPMDISKS.TXT -EXISTS FILE SIZE=0x000006F5
Menu

C   BOOT CP/M
L   LIST FILES
1 - Disk A = DISK-A.BIN -EXISTS FILE SIZE=0x0003E900 = 8" SSSD 250K
2 - Disk B = DISK-B.BIN -EXISTS FILE SIZE=0x0003E900 = 8" SSSD 250K
3 - Disk C = DISK-C.BIN -EXISTS FILE SIZE=0x0003E900 = 8" SSSD 250K
4 - Disk D = DISK-D.BIN -EXISTS FILE SIZE=0x0003E900 = 8" SSSD 250K

>.........
Z80MC 61K CP/M 2.2

WBOOT
a>dir
A: MON-32K  COM : ASM      COM : BIOS     ASM : STAT     COM
A: NSWP     COM : XMODEM   CFG : WM       COM : DDT      COM
A: DEBLOCK  ASM : WM       HLP : NSWP     TXT : DUMPCPD  COM
A: DISKDEF  LIB : DUMP     ASM : DUMP     COM : XMODEM   COM
A: GENMOD   COM : MOVCPM   COM : PIP      COM : RLOCBIOS COM
A: STDBIOS  ASM : SUBMIT   COM : SYSGEN   COM : XSUB     COM
A: LOAD     COM : ED       COM : LS       COM : MBASIC   COM
A: SURVEY   COM : VIEW     COM
a>
```

Cold boot prints “Z80MC 61K CP/M 2.2”.
Warm boot prints the “WBOOT”.
CP/M CCP prints the “a>”

At the time of writing this manual, there is no way to return to the monitor, other than through a reset.

SIO Card LEDs

There are 3 LEDs on the SIO Board.

- SD Card Access (directly connected to SD Card Select).
- SD Card Write (turned on by software during CP/M Disk Write).
- Spare (not used at this time – we'll think of something)!

CPU + SIO Cards

The system will work fine with just the CPU and SIO board. In this configuration it will also autoboot to CP/M.

Note about the XMODEM program included in Disk-A.BIN. This program is by Martin Eberhard and is configured to work with the ACE comport. It cannot use the BIT comport since BIT does not use UART style I/O hardware.

Appendix A - USB Adaptors

These are relatively new (compared to the Z80). Most of them use one of these 4 common chips:

- FT232R by FTDI <http://www.ftdichip.com/FTDrivers.htm>
- PL2303 by Prolific
- CH340 by Jiangsu Heng Qin Ltd. <http://www.wch.cn>
- CP2102 by Silabs

This is not a plug-and-play environment: It is a Wild-West situation, much like the early days of microcomputing! There are no standards. You probably won't get a manual. The adapter might provide normal, or inverted serial data. It may have 3.3v, 5v TTL, or RS-232 voltage levels. It might, or might not implement the handshake lines (The Z80MC uses RTS to control Reset).

They all require drivers. Some install the drivers automatically, on some operating systems. Some drivers don't work with some operating systems; for example, in my experience the CP2102 drivers fail with Windows XP. Expect to have to hunt down drivers, and install them yourself.

The FP and SIO board connectors match the Sparkfun.com #9718 USB-serial cable. It uses the FT232R chip, and plugs right in. They have drivers that work for just about any version of Windows, Mac OS, and Linux. It costs about $18, but is the quickest and easiest to get working.

Or, these adaptors can be cheaply obtained on Ebay for a couple of bucks. If you are in a country where Chinese food is just called food, then shipping should be quick. If you plug it in before loading the drivers and the drivers don't automatically install, you will need to unplug it, install the drivers and look for the USB device in your device manager then either update the driver or just delete it and let the system find the driver when you plug it back in. You can, from the device manager, click the properties of the USB port and select any com port you wish.

I don't know a lot about the USB interface and drivers, but I have found out through experimenting that the data from the terminal emulator program is sent to the USB device in some form of packet protocol. Your knowledge, experience or drivers may be different, so please take the following information provided with a grain of silicon.

In normal typing, the keys are sent one by one to the USB adapter, and come out as TTL serial data. But when sending a text file, strange things happen as you introduce character and line delays. When selecting 1mSec or more character delay, the characters all come out with about 14mSec delay. I'm guessing this is the time it takes to process 1 packet, and the 1mSec delay is long enough to have the driver start the real "send" process. But when sending a hex file line of about 70 characters, line delays of less than 70mSec do not provide **any** line delays. When the delay is about 100mSec, then you get a 30mSec delay. At 9600 baud, 70 characters take about 70mSec to send. So, it's my guess that the terminal program sends all 70 characters to the USB within a few microseconds, and the delay matches the time the USB chip sends out those 70 characters.

Bottom line, when sending hex or other text files, set character pacing to 0mSec and line pacing to 150mSec. You can choose more or less and experiment if you need more speed. Otherwise this value worked well with all tests performed. Please let us know what you discover; We're learning this right along with you!