Z80 Home Brew #3
Monitor ROM
December 15, 2022

Table of Contents

Overview

The machine consists of a Z80-CPU, 2K EPROM (expandable to 4K), 4K of RAM, two Z80-PIO, one Z80CTC, one Z80-SIO/0, and an HDSP-2111 8-character 5x7 LED dot matrix display. All four PIO channels are wired to a 50-pin FCC connector (P3). Both SIO channels are wired to a DIP14 socket/connector (P4). General-purpose I/O expansion is provided on a pair of DIP14 sockets/connectors (P1, P2). The HDSP-2111 display is wired as an I/O expansion.

The original EPROM for this machine was damaged, and all odd pages (256-byte chunks) are missing (A8 shorted to GND). Half the monitor code is missing, but some things can be reconstructed or restored based how the existing code interacts.

General Operation

The monitor may be entered from a running program by pressing the NMI button or by executing a RST 1 instruction (0xCF) in the program. In the case of RST 1, the saved PC will point to the RST 1 instruction (not the next instruction). In both cases, the saved PC is printed after a ‘>’ character before entering the monitor loop.

Pressing NMI while running the monitor is similar to RESET.

The monitor prompt is the asterisk (‘*’) character, however that is reconstructed code and there is no indication what the prompt character was originally.

User Commands

In the following, *addr* and *byte* are entered in hexadecimal. **[ ]** means the parameter is optional. CR means the carriage return key. LF means the line feed key.

Implemented Commands

The following commands either have code that defines the operation, or can be reasonably determined based on space available and logical deduction.

[*addr*]CR

print the byte stored at *addr*, advance *addr* by 1.

[*addr*]**/**

print the word stored at *addr*, advance *addr* by 2. Also used after '**T**' or '**R**' commands to view subsequent values.

[*addr*]**I**

Input bytes starting at *addr*. Prints current address and current contents and waits for input. Commands are:

[*byte*]CR

store *byte* (if entered) in *addr*, increment *addr* by 1. CR alone is used to skip to next location without altering contents.

-

decrement *addr* by 1.

.

return to monitor

[*addr*]**G**

Go (start execution) at *addr* or saved PC

P

start execution at PC+1 (for continuing after RST 1 traps).

T

print word at saved SP (top of stack), increment *addr* by 2. The top of stack will not include the PC that was pushed as part of the NMI or RST 1. The rest of the stack may be viewed by using '**/**' commands.

R*reg***/**

print contents of saved register pair. *reg* is one of: A,B,D,H,A',B',D',H',X,Y,S,P. Increment *addr* by 2, effectively selecting the next register pair. Successive registers may be viewed by using '**/**' commands.

H

Host control mode? Appears to accept commands from SIO channel B, as if connected to a remote computer. Remote commands are echoed to console (channel A), unclear what other interaction there is. Or possibly just an alternate set of commands still using the console (current implementation, for testing).

Unknown Commands

The following commands do not have code.

LF
(unknown)

^
(unknown) “up”?

V
(unknown) “down”?

Host Control Commands

Host control commands may be prefixed with an octal number. Unclear how data is sent back to host (in octal?). Might still be controlled by console, unclear if any interaction with host.

.
(unknown)

,
(unknown)

\
(unknown)

LF
(unknown)

CR
(unknown)

S
(unknown)

R
(unknown)

X
Exit Host mode, return to monitor

M
(unknown interaction with PIO-connected device) possible device memory test

/
(unknown)

^

(unknown)

<

(unknown)

>

(unknown)

G

Get 2K bytes from PIO-connected device. Read pairs of 4-bits from PIO2B. Buffer at 1600H. Missing routines to initialize and perform actions between each 4-bits.

P

Put 2K bytes to PIO-connected device. Writes pairs of 4-bits to PIO2A. Buffer at 1600H. Missing routines to initialize and perform actions between each 4-bits.

TBD

enter Terminal mode, with HEX loading enabled.

[*addr*]TBD

write *addr* to the PIO-connected device as two 6-bit values on pio1A, strobing pio2B bit 4 for low value and pio2B bit 5 for high value. The value *addr* is first decremented by 2 and xor'ed with 1. Any bits beyond the 12 are ignored.

Terminal Mode Operation

The code passes characters both directions between SIO channels A and B. In addition, there is a mode where the stream coming from channel B may contain Intel HEX format data, which is presumably loaded into memory. The console (channel A) user may press Ctrl-] (0x1d, ASCII "GS") to cause a BREAK condition to be sent to channel B, presumably to disconnect from the host or terminate a program. There is some number of successive Ctrl-] presses required to activate the BREAK feature, to avoid accidental disconnect. At least one other key command seems to exist, possibly to exit terminal mode. Considering the nature of "terminal mode", this exit key would have been some control character that is not normally used (similar to the BREAK code).

Hardware Clues in Software

Device on 50-pin Connector

- Data path seems to be 4-bits wide, output on low bits of PIO2-A and input on low bits of PIO2-B.
- There may be a 6-bit path, sometimes used for address hi/lo, on PIO1-A.
- Capacity seems to be 2K bytes (4K nibbles).
- Addressing seems to be 12-bit, with the LSB selecting the nibble.

    - In one case, the user-entered address is modified before sending: addr = (addr – 2) ^ 1, and 6-bit halves are successively sent to PIO1-A while pulsing PIO2-B bits 4 and 5 (for low and high halves, respectively).
- There seems to be either RAM associated with the device (e.g. a buffered EPROM programmer) or else the device allows plugging in of RAM chips (e.g. an IC tester).
- There is a subset of commands that operate on this device, activated by the ‘H’ command.
- These subcommands use octal, in contrast to the main monitor commands that use hexadecimal. Although, parameters that are input for various commands may use some other base (the code is missing).
- In some cases, it appears that the PIO RDY outputs are used by the device to latch data, but in one case (at least) PIO2-B bits 4 and 5 are bit-banged to latch data (from PIO1-A). Note that, in order for RDY to be effective, the /STB inputs must also be used.
- The ‘H’ subcommand mode programs PIO1-A, PIO1-B, and PIO2-A in output mode (RDY signals active), and programs PIO2-B in control mode with bits 0-3 as inputs, 4-7 as outputs.

LED 5x7 Display

- No code has been found that accesses the display. It may have been added for experimentation purposes (vs. being part of the development environment).
- This display is capable of showing eight ASCII characters (full 7-bit decoding).
- The display datecode indicates 1987, while the rest of the parts have datecodes 1977-1981. It is possible that the display is not part of the original equipment. The wire-wrap shows that that device was added last (in that vicinity).
- It is possible that the display was used to duplicate console output or show some other status, but that would have to be in the missing code.

Serial Ports

- SIO-A is the user console. Unclear if that was a hard-copy like ASR-33 or a CRT terminal.
- Under some conditions, SIO-B is connected to a remote system (“terminal mode”).
- The only confirmed code to use SIO-B is the terminal mode loop.
- It is not know if the 4800/9600 switch affects SIO-A or SIO-B. It would depend on what each is connected to, e.g. an ASR-33 can only be 110 baud, so 4800/9600 must be SIO-B. But if SIO-B connects to a modem, then 4800/9600 must be for SIO-A (given the time period and modem speeds of that era). The actual data used to program SIO bauds is missing.
- All four toggle switches feed to the handshake inputs on the SIO. The only code known to access those inputs is the test for 4800/9600 done during initialization.