Monitor 2.3

User's Manual

For use with the
Proton Z80 Modular Computer

Written by Ricardo Kaltchuk
kaltchuk@gmail.com
Version 1 - October, 2021

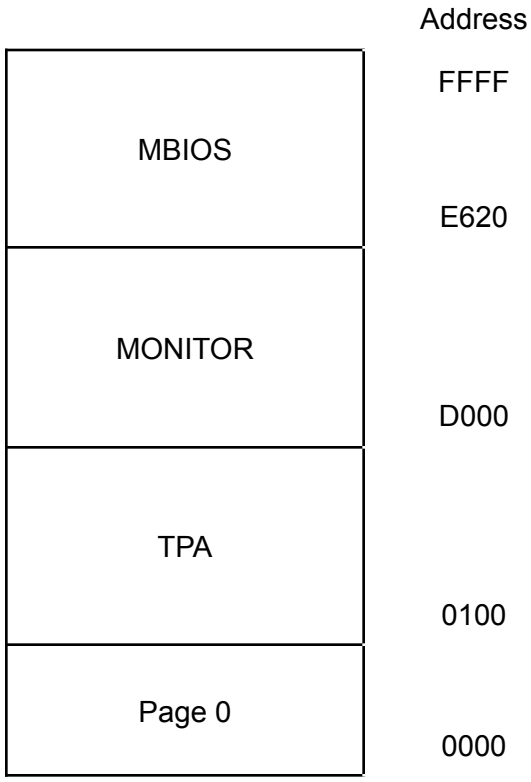
Index

Introduction

Monitor is a very basic operating system designed for the Proton Z80 Modular Computer. It was intended to serve as a first way to interact with the hardware during the development of the Proton. It was crucial to test and implement the BIOS for CP/M.

Version 1.0 was monolithic and ran on ROM (bottom 16K) with 48KB of RAM and had a few basic memory access functions. Differently, version 2.0 was built with the same concept of CP/M, i.e. the Monitor *per se* and a BIOS, which is almost the same BIOS used by CP/M.

When Proton resets (cold boot), the bottom 16KB are allocated to ROM and the upper 48KB are allocated to RAM. Jumper J5 on MEM Card specifies which of the four ROM blocks will be used. Normally CP/M resides on block 0 and Monitor on block 1. The RAM bank is selected by jumper J4 on MEM Card.The bootloader copies the BIOS to RAM address E620h and the Monitor to RAM address D000h. The following diagram shows the memory allocation after boot is completed:



Page 0 is the first 256-byte block of memory and is used by the BIOS and the Monitor for information exchange and function calls. Please check the documentation on CP/M for details about page 0 and its structure. The Transient Program Area - TPA, is a block of almost 52KB, between page 0 and the Monitor, which can be used to store data and user-written programs.

After the bootloader copies the BIOS and the Monitor to RAM, it swaps to full RAM configuration, i.e, all 64KB become RAM. From this point on, Monitor is in control, exhibits a prompt (‘>’) and waits for a command.

```
Z80 Modular Computer by Kaltchuk 2020.
MBIOS 2.1.
Monitor by Kaltchuk 2020

>
```

If the user types “? <ENTER>”, the Monitor will show a list of all available commands.

```
>?

 MONITOR 2.3 - Dec/2021.
 Options:   READ aaaa              read from memory.
            WRITE aaaa,c1 c2 cN    write to memory.
            COPY aaaa-bbbb,cccc    copy memory block.
            FILL aaaa-bbbb,cc      fill memory block.
            COMPARE aaaa,bbbb      compare memory areas.
            FLASH                  initialize Flash Card.
            DREAD d,ttt,ss         read from disk.
            DOWN d,ttt,ss          download one sector from disk.
            UP d,ttt,ss            upload one sector to disk.
            VERIFY d               verify disk.
            FORMAT d               format disk.
            XMODEM r aaaa          receive file using xmodem protocol.
            XMODEM s aaaa-bbbb     send memory block using xmodem protocol.
            HEX2COM aaaa           convert intel hex to executable.
            RUN aaaa               run program.
            BOOT                   warm boot.
```

The following pages detail each command.

Memory Commands

READ aaaa

Read and display one memory block (256 bytes), starting from address “aaaa” (four characters in hexadecimal). After that, the user can type <ENTER> to read the next block or <ESC> to abort the command. Example:

```
>READ 0000
>ADDR: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
>----- -----------------------------------------------  ----------------
>0000: C3 23 E6 00 00 C3 FF FF FF FF 7F FF FF FF FF FF  .#..............
>0010: FF 6D FF FE F4 EB 9E FB FD 3B 7E DD 67 BE FB FE  .m.......;~.g...
>0020: D7 DF F3 F9 DF FF FE FE FF 7B B7 FF FF FE FF BB  .........{......
>0030: FF EF FF 58 EA FB FD DF DF FE 6A E7 A1 FF DD C7  ...X......j.....
>0040: 3E FF FE FF EF EF FF FF FF 7F FF FF FF FF FF 7F  >...............
>0050: 7D BF FE EA F6 DE 3F BD F3 7F DF 6B 9E AC FB F9  }.....?....k....
>0060: FF FF F7 DD 57 FF EA 7F FF 7F F9 C8 7E DD FE F8  ....W.......~...
>0070: EF FA B5 FA DF EF 23 BD 6E CB B7 DB FB E8 77 13  ......#.n.....w.
>0080: 52 45 41 44 20 30 30 30 30 00 30 2C 30 30 00 30  READ 0000.0,00.0
>0090: 30 00 7E BE FF E6 7F 5F FF DE F7 FE AE 8E BB F9  0.~...._........
>00A0: 7F FF FF BF DF ED FD FD 75 FA 5E 9F B7 FD FF E7  ........u.^.....
>00B0: DE FF 7C 7D 3E DF B6 3F DE AB 33 4F BD FE D7 D7  ..|}>..?..3O....
>00C0: F7 FF B9 FE FD FF EB FF FF E7 F7 FD FD FE FF FF  ................
>00D0: FF 3B DF 9B FD FF F7 F7 DD FF FF 92 EF BF 3F F7  .;............?.
>00E0: FF FD 7F AE EF BF FF FF FF 9D FF F1 BF BC 67 FB  ..............g.
>00F0: 7F EB FE CB FE FF DD 1F 7A 2C 38 99 B1 B7 22 7B  ........z,8..."{
>#================= <ENTER> = next page, <ESC> = quit =================#
```

WRITE aaaa,c1 c2 cN

Write a list of bytes to memory, starting at address “aaaa” (four characters in hexadecimal). The bytes are all two characters hexadecimal separated by space. Example of use:

```
>WRITE 0200,21 7F A0 35
```

COPY aaaa-bbbb,cccc

Copy a memory block, from address “aaaa” to “bbbb” (included), to address “cccc”. All addresses are four characters in hexadecimal. Example of use:

```
>COPY 0100-02FF,A000
```

FILL aaaa-bbbb,cc

Fill a memory area, from address “aaaa” to “bbbb” (included), with byte “cc” (two characters hexadecimal). All addresses are four characters in hexadecimal. Example of use:

```
>FILL 1000-10FF,AA
```

COMPARE aaaa,bbbb

Compare memory areas “aaaa” and “bbbb”, 16 bytes at a time and display the results. All addresses are four characters in hexadecimal. On the first line it prints the content of the first address and on the second line, the content of the second address. If the contents are equal it will print “=”, indicating that we have a match. After that, the user can type <ENTER> to compare the next line or <ESC> to abort the command. Example of use:

```
>COMPARE 0100 0200

>0100: F7 F7 FF FF FF FE FF FF FF FF FF DF BF FF FF FF
>0200: F3 FF 6D == == FF == == EF == == FF FF == == ==
>      <ENTER> = next line, <ESC> = quit
```

HEX2COM aaaa

Convert a program in Intel hex format, starting at memory address “aaaa”, to an executable program written to the address specified in the original hex file. Let's say that the user wrote an assembly program in Windows, with a text editor and compiled it with TASM, which generates a .OBJ file, in Intel hex format. This file can be downloaded to the Z80MC with XMODEM and converted to an executable with HEX2COM. Example of use:

```
>HEX2COM 1000
```

RUN aaaa

Run (execute) a user written program, starting at address “aaaa” (four characters in hexadecimal). The user’s program should end with a jump to address 0000 (JP 0000), so a warm boot is performed and Monitor can resume command. Example of use:

```
>RUN 0200
```

Disk Drive Commands

DREAD d,ttt,ss

Read one sector (512 bytes) from disk “d”, starting at track “ttt” and sector “ss”. “d” must be one of the 16 drives, a letter from ‘a’ to ‘p’. “ttt” is a three character hexadecimal track, from 000 to 1FF; and “ss” is a two character hexadecimal sector, from 00 to 1F. Actually, this command copies the sector to RAM address E400h and shows the memory content like in the “read” command, but 32 lines at a time. Example of use:

```
>DREAD C,000,00
> DTS: C-0000-00 ,LBA: E0008000 (DISKPAD = E400)
>ADDR: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
>----- -----------------------------------------------  ----------------
>E000: 00 53 54 41 52 53 20 20 20 43 20 20 00 00 00 0E  .STARS   C  ....
>E010: 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
>E020: 00 43 43 32 20 20 20 20 20 43 4F 4D 01 00 00 08  .CC2     COM....
>E030: 08 00 09 00 0A 00 0B 00 0C 00 00 00 00 00 00 00  ................
>E040: 00 43 4C 49 4E 4B 20 20 20 43 4F 4D 00 00 00 2C  .CLINK   COM...,
>E050: 0D 00 0E 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
>E060: 00 43 20 20 20 20 20 20 20 43 43 43 00 00 00 0C  .C       CCC....
>E070: 0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
>E080: 00 44 45 46 46 20 20 20 20 43 52 4C 00 00 00 60  .DEFF    CRL...`
>E090: 10 00 11 00 12 00 00 00 00 00 00 00 00 00 00 00  ................
>E0A0: 00 44 45 46 46 32 20 20 20 43 52 4C 00 00 00 2A  .DEFF2   CRL...*
>E0B0: 13 00 14 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
>E0C0: 00 53 54 44 49 4F 20 20 20 48 20 20 00 00 00 0E  .STDIO   H  ....
>E0D0: 15 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
>E0E0: 00 43 43 4F 4E 46 49 47 20 43 4F 4D 01 00 00 1C  .CCONFIG COM....
>E0F0: 16 00 17 00 18 00 19 00 1A 00 00 00 00 00 00 00  ................
>E100: 00 48 45 4C 4C 4F 20 20 20 43 20 20 00 00 00 02  .HELLO   C  ....
>E110: 1B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
>E120: 00 48 45 4C 4C 4F 20 20 20 43 52 4C 00 00 00 06  .HELLO   CRL....
>E130: 1C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
>E140: 00 48 45 4C 4C 4F 20 20 20 43 4F 4D 00 00 00 1C  .HELLO   COM....
>E150: 1D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
>E160: 00 53 54 41 52 53 20 20 20 43 52 4C 00 00 00 0F  .STARS   CRL....
>E170: 1E 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
>E180: 00 53 54 41 52 53 20 20 20 43 4F 4D 00 00 00 27  .STARS   COM...'
>E190: 1F 00 20 00 00 00 00 00 00 00 00 00 00 00 00 00  .. ............
>E1A0: 00 43 4C 49 42 20 20 20 20 43 4F 4D 00 00 00 2A  .CLIB    COM...*
>E1B0: 21 00 22 00 00 00 00 00 00 00 00 00 00 00 00 00  !."............
>E1C0: 00 44 20 20 20 20 20 20 20 43 4F 4D 00 00 00 0E  .D       COM....
>E1D0: 2E 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
>E1E0: 00 54 41 52 47 45 54 20 20 43 20 20 00 00 00 1E  .TARGET  C  ....
>E1F0: 29 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  )...............
>#================= <ENTER> = next page, <ESC> = quit ==================#
```

DOWN d,ttt,ss

Download one sector (512 bytes) from disk “d”, track “ttt”, sector “ss”. “d” must be one of the 16 drives, a letter from ‘a’ to ‘p’. “ttt” is a three character hexadecimal track, from 000 to 1FF; and “ss” is a two character hexadecimal sector, from 00 to 1F. The content is downloaded to RAM address E400h. Actually, this command does exactly the same as “dread”, but does not display the content, which can be read with a “read e400” command. Example of use:

```
>DOWN P,0A5,1F
```

UP d,ttt,ss

Upload one sector (512 bytes) from RAM E400h to disk “d”, track “ttt”, sector “ss”. “d” must be one of the 16 drives, a letter from ‘a’ to ‘p’. “ttt” is a three character hexadecimal track, from 000 to 1FF; and “ss” is a two character hexadecimal sector, from 00 to 1F. Example of use:

```
>UP C,002,0F
```

VERIFY d

Verify the integrity of disk “d”. “d” must be one of the 16 drives, a letter from ‘a’ to ‘p’. This command is executed in four steps. First, the original content of the sector is copied to memory; second, the sector is written with a predefined sequence of characters; third, the sector is read again and compared with the written sequence; forth, the original content is uploaded back to the disk. In fact, steps 2 and 3 are executed four times, each time with a different pattern (00, FFh, 55h and AAh).

This command takes several minutes (23 minutes @ 8MHz) to verify all 16384 sectors of the selected disk. The operation can be aborted at any time by typing <ESC>. Example of use:

```
>VERIFY M

Track 0000................................
Track 0001................................
Track 0002................................
Track 0003................................
...

Track 01FC................................
Track 01FD................................
Track 01FE................................
Track 01FF................................
>
```

FORMAT d

Format disk “d” using CP/M standard. “d” must be one of the 16 drives, a letter from ‘a’ to ‘p’. If drive ‘a’ is selected, the first track will be reserved and formatting will be performed on track 001. This is done so the user can write CP/M and the BIOS on track 000 for disk boot. This command does not erase the disk. It only initializes the file allocation table and makes the disk look “empty” if accessed by CP/M. Example of use:

```
>FORMAT M
Format drive M (y/n)? y
Format complete.
>
```

FLASH

Initialize the Flash Card so it can be accessed by the CPU. The Monitor doesn't need a Flash Card to operate. If the BIOS tries to initialize the Flash Card during the boot and the card isn't present, it will block. Due to this fact, the BIOS does not initialize the Flash Card during the boot. A Flash Card initialization must be performed prior to any drive operation. All the commands that deal with drive operations (DREAD, UP, DOWN, VERIFY or FORMAT) check if the Flash Card has been initialized and automatically execute a FLASH command if necessary. This command only needs to be executed if a user-written program will be executed and no other disk drive command has been executed after the last reset. If an operation on the compact flash is performed before initializing the Flash Card, the results are unpredictable and may eventually corrupt the data on the compact flash. Example of use:

```
>FLASH
FLASH Card initialized.
>
```

Other Commands

XMODEM r aaaa

or

XMODEM s aaaa-bbbb

In the first form, a file is received using xmodem protocol and saved at RAM address “aaaa”. It is the user’s responsibility to assure that the data received will be written only inside the TPA.

In the second form, a block of memory, from “aaaa” to “bbbb”, is sent using xmodem protocol. All addresses are four characters in hexadecimal. Xmodem protocol works with blocks (records) of 128 bytes. So even if you want to send 1 byte to the host, actually 128 bytes will be sent.

Remember that Monitor is not a development platform, so it doesn't have a text editor nor a compiler; so any program that the user wants to run must be written on another computer, compiled and transferred to the Z80MC via the XMODEM command. (Unless you want to do it the hard way). Examples of use:

```
>XMODEM R 1000
File received.
>
>XMODEM S 1000-1100
File transmitted.
>
```

BOOT

Performs a warm boot.