

# ZEUS MONITOR May 2023

## 8-BIT HOBBY COMPUTER BOOTSTRAP CODE

### Bootup sequence

At the bootup the Monitor..

1. ..jumps PRIMER (\$7F00); copies memory block \$0000-\$2FFF from bank F (EEPROM) to \$8000 (Monitor code); copies memory block \$7F20-\$7FFF from bank F (EEPROM) to \$FC00 (Call table); updates reboot jump pointer (\$8002); jumps \$8000.
2. .. jumps SETUP (\$A000); sets SP and INT table; generates table of variables and pointers at \$FE00.
3. ..initiates IOs; tests IO ports then prints and sends to UART POST letters "S", "P", "G" for SIO, PIO, GPIO modules respectively if detected in the system.
4. ..initiates lowmem banks 0-E (if present), i.e. copies memory block \$8000-\$8070 to \$0000 of each bank in order to keep RST and NMI primers functional whatever bank is ON; counts number of available memory banks in the system, enumerates them (writes bank ID to \$0007) and prints (in hex) the calculated number of banks.
5. ..switches lowmem to bank E; updates and prints system register state (\$FE81).
6. ..sets INT mode 2; sends "OK" to UART; enters HALT state and awaits for interrupt.

### The List of implemented commands

>clr

Fills most of TFT screen (everything except command line) with background color.

>dump (source)

Dumps on TFT screen 0x70 bytes starting from specified source memory address.

>send (source) (bytes)

Sends to UART specified number of bytes starting from specified source memory address.

>load (dest) (byte1) (byte2)...

Writes specified byte set to specified destination memory address.

>move (dest) (source) (bytes)

Copies data block of specified number of bytes from source memory address to destination memory address.

>jump (dest)

Sets Program Counter register (jumps) to specified memory address.

>exec (byte1) (byte2)...

Executes specified byte set. By default, saves the byte set augmented with 0xC9 (RET) to \$0100.

>sysl (nibble)

Sets system register low nibble value. Doesn't affect system register high nibble value. The command is intended for memory bank switching with no effect on high nibble bits.

>sysh (nibble)

Sets system register high nibble value. Doesn't affect system register low nibble value. The command is intended for sys reg bits control with no effect on memory bank switching.

>out (port) (value)

Outputs specified byte value to specified IO port.

>inp (port)

Reads register byte from specified port.

>crc7 (byte1) (byte2) ...

Calculates crc7 sum augmented with bit0 = 1 (check SD phys. layer specs) for specified byte set. By default saves the byte set (the message) ended with calculated crc7 byte to \$0100.

**Table of IO ports**

port	description
Fxh	SYSR register (write only)
00h	UART channel data (SIO)
01h	PS/2 channel data (SIO)
02h	UART channel instructions (SIO)
03h	PS/2 channel instructions (SIO)
10h	TFT + SD controls data (PIO)
11h	TFT data bus data (PIO)
12h	TFT + SD controls instructions (PIO)
13h	TFT data bus instructions (PIO)
20h/ <b>24h</b> /28h/ <b>2Ch</b>	channel A data (GPIO)
21h/ <b>25h</b> /29h/ <b>2Dh</b>	channel B data (GPIO)
22h/ <b>26h</b> /2Ah/ <b>2Eh</b>	channel A instructions (GPIO)
23h/ <b>27h</b> /2Bh/ <b>2Fh</b>	channel B instructions (GPIO)

**Table of variables and pointers (\$FE00h)**

address	description	state on reset
FE00h	font table pointer	8200h
FE02h	keyboard scan codes table pointer	9200h
FE04h	keyb buffer pointer	FD00h
FE06h	UART buffer pointer	0100h
FE08h	font color (R=0080h, Y=0084h, W=1084h)	0080h
FE0Ah	background color	0000h
FE0Ch	TFT char position row/col	0000h
FE80h	prev keyb char	00h
FE81h	system register state	FFh

**Table of bootload memory blocks (highmem bank)**

address	memory block description	Call table range
8000h	page 0	
8100h	interrupt vectors table	
8200h	font table (8x16)	
9200h	keyboard scan codes table	
9300h	bios subroutines	FC00h – FC2Ch
9500h	command line parser	FC30h – FC4Ch
9700h	command routines	FC50h – FC9Ch
9B00h	SD command routines	FCA0h – FCDCh
A000h	setup	

**Example 1** Dump to display system variables block and change background color.

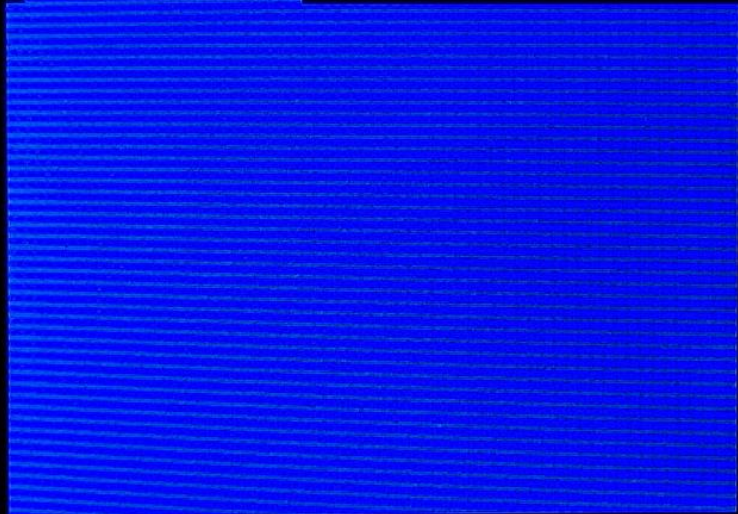
```
>dump fe08
FE08: 84 00 00 00 00 0A E8 D8
FE10: CF A0 CA AE 9C 2C 4E 80
FE18: 3B A8 6E 8E CA FB 83 E2
FE20: A2 B2 BB 2E 8E 0A A8 FE
FE28: AE 8A BB A2 88 AE 08 FA
FE30: A2 A0 20 83 DA 8F 37 BA
FE38: AA EB EF AE 2A 6E AE 0A
FE40: EA BA 8B 9A 3D 28 92 D2
FE48: 08 AA AA EA 2A F3 F2 F0
FE50: 03 AA AA 2B C6 EA FB 68
FE58: A9 B0 C8 B8 22 F2 BA 0F
FE60: 0A B9 EA E8 6F EE 83 BA
FE68: 02 2A A2 0A FF 02 34 EC
FE70: 28 EC BA FE 3C E0 1A EA
```

PS  
FE/10

```
>load fe0a 00 10
FE08: 84 00 00 00 00 0A E8 D8
FE10: CF A0 CA AE 9C 2C 4E 80
FE18: 3B A8 6E 8E CA FB 83 E2
FE20: A2 B2 BB 2E 8E 0A A8 FE
FE28: AE 8A BB A2 88 AE 08 FA
FE30: A2 A0 20 83 DA 8F 37 BA
FE38: AA EB EF AE 2A 6E AE 0A
FE40: EA BA 8B 9A 3D 28 92 D2
FE48: 08 AA AA EA 2A F3 F2 F0
FE50: 03 AA AA 2B C6 EA FB 68
FE58: A9 B0 C8 B8 22 F2 BA 0F
FE60: 0A B9 EA E8 6F EE 83 BA
FE68: 02 2A A2 0A FF 02 34 EC
FE70: 28 EC BA FE 3C E0 1A EA
```

PS  
FE/10

```
>ctrl
```



**Example 2** Load a data block to free memory address and output it to UART, then jump to bootup address.

```
>load 7e00 30 31 32 33
7E00: 30 31 32 33 6C A6 6A 3E
7E08: 88 9E 2A 8A 89 BE 3F A0
7E10: AA A2 22 BA 3E AE 00 1E
7E18: 1A A6 63 B3 23 EA 88 84
7E20: C3 BA AA AF CB A0 02 D5
7E28: BB B8 C8 A8 83 68 BE 28
7E30: 8E 28 2C AB 22 C4 F2 2B
7E38: A8 22 BE CE BC 9A 30 8C
7E40: 8F 00 AA 8A 22 BD DE F2
7E48: B8 82 2F 8E AF AE BB DC
7E50: BA 22 C0 8B D3 5F AA 3B
7E58: 2A 82 8A B8 2A F9 83 FB
7E60: A2 2A 82 AE CE 23 EA B0
7E68: BC BB 30 E8 D9 E3 89 9A
```

PS  
FE/10

```
>send 7e00 0008
7E00: 30 31 32 33 6C A6 6A 3E
7E08: 88 9E 2A 8A 89 BE 3F A0
7E10: AA A2 22 BA 3E AE 00 1E
7E18: 1A A6 63 B3 23 EA 88 84
7E20: C3 BA AA AF CB A0 02 D5
7E28: BB B8 C8 A8 83 68 BE 28
7E30: 8E 28 2C AB 22 C4 F2 2B
7E38: A8 22 BE CE BC 9A 30 8C
7E40: 8F 00 AA 8A 22 BD DE F2
7E48: B8 82 2F 8E AF AE BB DC
7E50: BA 22 C0 8B D3 5F AA 3B
7E58: 2A 82 8A B8 2A F9 83 FB
7E60: A2 2A 82 AE CE 23 EA B0
7E68: BC BB 30 E8 D9 E3 89 9A
```

PS  
FE/10

```
>jump a000
7E00: 30 31 32 33 6C A6 6A 3E
7E08: 88 9E 2A 8A 89 BE 3F A0
7E10: AA A2 22 BA 3E AE 00 1E
7E18: 1A A6 63 B3 23 EA 88 84
7E20: C3 BA AA AF CB A0 02 D5
7E28: BB B8 C8 A8 83 68 BE 28
7E30: 8E 28 2C AB 22 C4 F2 2B
7E38: A8 22 BE CE BC 9A 30 8C
7E40: 8F 00 AA 8A 22 BD DE F2
7E48: B8 82 2F 8E AF AE BB DC
7E50: BA 22 C0 8B D3 5F AA 3B
7E58: 2A 82 8A B8 2A F9 83 FB
7E60: A2 2A 82 AE CE 23 EA B0
7E68: BC BB 30 E8 D9 E3 89 9A
```

PS  
FE/10



**Example 3** Calculate crc7 sum of a message and dump it to display.

```
>crc7 00 01 02 03
45
```

PS  
FE/10

```
>dump 0100
0100: 00 01 02 03 45 E3 CE EE
0108: 0A 8A AB E8 3A F8 B8 3B
0110: EB A8 82 AA 1F 82 AB 8B
0118: EF AA EA AA DF FA 09 8C
0120: B8 AF F3 B8 CF FF BA 8A
0128: 6B BA AA AF B2 F2 AB B2
0130: EC 2A 2E FF 30 67 B6 83
0138: B2 2B 80 AB A3 AA B9 2B
0140: A8 3A E2 82 88 1B F8 E8
0148: A8 BF B0 A9 E3 EF BF F8
0150: B0 A0 C2 AA 8C D3 0C 0A
0158: AA E3 AA 2B 8A C0 CA E6
0160: AC BA BB 28 B8 00 2E AA
0168: A7 1A 8B 98 88 3A EA 3B
```

PS  
FE/10