H8 28C256 Monitor ROM
January 28, 2020
PROPOSED

Background

This ROM upgrade to the v3.x H8-Z80-CPU board expands the ROM space and allows for online updating of ROM contents. This new ROM will replace the PAM37 and MMS monitor ROMs with one that combines the features of both, supporting the H8 Front Panel as well as a terminal attached to the console serial port.

The console support will follow the command (particularly Boot command) syntax introduced by Heath in their H89 monitors, and as extended by Magnolia Microsystems.

The H8 front panel support will expand on the PAM37 monitor operation.

Conventions

In examples in this document, underlined characters indicate what the user types, non-underlined characters indicate what the computer prints. The string “(cr)” is used to indicate pressing the RETURN (Enter) key.

Supported Features

Operating systems (boot): HDOS 2 and 3, Heath CP/M 2, MMS CP/M 2 and 3 (others TBD).

Boot devices: H17, H47, H37, H67, MMS77316, GIDE, WizNet, VDIP1

Console commands: Boot, Go, Memory test, PC, Substitute, Terminal mode, Version

Dropped Support

baud Rate command, auto-baud detect, and any other dependency on the H19 as the console terminal.

Boot support for MMS MagNET, Corvus, XComp, and REMEX devices.

Auto-boot.

New Commands

L – List boot modules available in ROM
H – boot Help – list boot module details
D – Dump memory (hex, ASCII)

DipSwitch definitions (H8-SW1):

7	6	5	4	3	2	1	0
Baud 0 = 9600 1 = 19.2K	Default Boot Device 0 0 0 = MMS77316 5” 0 0 1 = MMS77316 8” 0 1 0 = Device at 7CH/174Q 0 1 1 = Device at 78H/170Q 1 0 0 = VDIP1 1 0 1 = GIDE 1 1 0 = WizNet 1 1 1 = none			Port 78H/170Q 0 0 = H37 0 1 = H47 1 0 = H67 1 1 = unused		Port 7CH/174Q 0 0 = H17 0 1 = H47 1 0 = H67 1 1 = unused	

Console/Keypad operation

Only one source of command input may be active at a time. If command entry has been started on the console, and a key is pressed on the front panel, the console command will be aborted. Likewise, if a command sequence has begun on the front panel keypad, and a command key is pressed on the console, the front panel command will be aborted.

Booting (console)

Note that backspace, or other edit keys, do not function when entering commands. However, the DELETE key (ASCII “DEL”, 7FH) may be used to cancel a command.

Full syntax

H8: Boot LL-D:string(cr)

1. The key ‘B’ starts the boot command. The monitor echos “Boot ” and waits for more characters.
    1. RETURN may be pressed in order to boot the default device, or more characters may be entered to define the desired boot device, unit, etc.
2. The next character must be a letter, which selects the type of device to boot. One exception is the letter ‘A’ which selects whatever is defined as the default device.
    1. When a letter is entered, it will be echoed twice followed by a dash.
    2. RETURN may be pressed in order to boot unit 0, or more characters may be entered.
3. Next a digit is entered to define the unit. The exact definition of this digit (“unit”) varies between different devices.
    1. RETURN may be pressed in order to boot the designated unit, or more characters may be entered.
4. Next the colon character (‘:’) followed by an alpha-numeric string may be entered.
    1. Interpretation of this string depends entirely on the boot target.
    2. Some boot targets completely ignore this string.

Alternate syntax

H8: Boot NN(cr)

Instead of a letter, etc, the entry may be digits to define the physical drive number. Note that the physical drive number specifically defines a boot target, and no additional specifiers may be entered.

Booting (front panel)

Note, booting through the front panel does not permit specification of a boot string. Some boot features will not be available.

Universal boot

1. Press the “0” key to start the Universal Boot sequence.
2. Display shows “dEU” (for “device”). Press a key “0” through “5” to select the device. The device mnemonic will be displayed in the first 3 elements. Fixed-port devices will also display the port in the middle 3 elements.
3. If display shows “Por” (for “port”), select I/O port using “0” through “3”. Port will be displayed in the middle 3 elements.
4. Display shows “Uni” (for “unit”), select the unit number using “0” through “9”.

Boot will commence automatically after selecting the unit.

One-key boot

The primary or secondary boot device may be selected using the “1” and “2” keys, respectively. “Primary” is the device defined as the default boot device in the dipswitches SW1. “Secondary” is the device on the “other” port (174Q/170Q). TODO: define secondary independent of ports 174Q/170Q.

Boot Device Designations

Letter	Phy Drv	Device Descr.	Num Units	H8-FP key	Cmd String
A	(default device selection dictates parameters)				
B	0-2	H17	3	0	n/u
C	46-49	H37	4	3	n/u
D	5-8	H47	4	1	n/u
E	3-4	H67	2	2	n/u
F		Reserved			
G		MMS77314 REMEX(deprecated)			
H		MMS77314 Corvus (deprecated)			
I	29-32	MMS77316 8”	4		n/u
J	33-36	MMS77316 5.25”	4		n/u
K		MMS77317 XCOMP (deprecated)			
L		Reserved			
M	40	MMS77318 ramdisk	1		n/u
N		MMS77422 MagNET (deprecated)			
O		MMS77320 Ctrl 0	4		partition
P		MMS77320 Ctrl 1	4		partition
Q		Reserved (was MMS77320 Ctrl 2)			
R		Reserved (was MMS77320 Ctrl 3)			
S		Reserved (was MMS77320 Ctrl 4)			
T		Reserved (was MMS77320 Ctrl 5)			
U		Reserved (was MMS77320 Ctrl 6)			
V	41	VDIP1 (was MMS77320 Ctrl 7)	1	6	file/prog
W	60	WizNet	1	5	file/prog
X	70	GIDE	9 (partition)	4	segment/LUN
Y		Reserved			
Z		Reserved			

Console Commands Summary

Boot (cr)
Boot AA-(cr)
Boot AA-U(cr)
Boot AA-U:string(cr)
    Boot from the default device, optionally specifying unit and string.
Boot LL-(cr)
Boot LL-U(cr)
Boot LL-U:string(cr)
    Boot from device L, optionally specifying unit and string.
Boot NN(cr)
    Boot from device physical drive NN.

Dump AAAA(cr)
Dump (cr)
    Dump 128 bytes of memory, in hex and ASCII, starting at address AAAA. If AAA is not specified, starts where the last dump ended or substitute started (or last memory address on front panel).

Go AAAA(cr)
Go (cr)
    Go to (start execution at) address AAAA. If AAAA is not specified, use the PC of the saved debug registers.

Help boot
    Print a table showing all boot modules/devices included in the ROM. Sample output:

```
 GIDE   X 4 70-78
 H17    B 0 00-02
*H37    C 3 46-49
 H47    D 1 05-08
 H67    E 2 03-04
 VDIP1  V 6 41
 WizNet W 5 60
```

    The first column is the device name, with an asterisk indicating the default device according to the dipswitch settings.
    Second column is the device letter to be used in the "Boot XX-..." form of the boot command.
    Third column is the Front Panel key for the front panel universal boot command sequence.
    Fourth column is the physical drive number range, used in the "Boot NN" form of the boot command.

List boot modules
    List the names of the boot modules included in the ROM. An asterisk indicates the default device according to the dipswitch settings.

Memory test(cr)
    Run a simple 64K memory test. Use RESET to end the test.

Prog Counter (cr)
Prog Counter AAAA(cr)

Set the debug PC register. Note that the front panel debugger and console monitor use the same set of saved, debug, registers. If AAAA is not specified, then the current PC is displayed and the option given to change it.

Substitute (cr)
Substitute AAAA(cr)

Interactively alter memory. If no AAAA is specified, use the stored address from the last substitute or dump (or last memory address on front panel). For each address, enter hex digits and (cr) to modify, (cr) to skip to next, '-' to skip to previous, and '.' to end command.

Terminal mode(cr)

Enter terminal mode, where the console is effectively connected to the "modem" serial port (0D8H/330Q). Use RESET to return to monitor.

Version

Display the ROM version.

Updating the ROM

The standalone, VDIP1-based, program VFLASH.SYS is used to update the entire ROM image from a file (that is also) on the USB stick. This program may be run using the boot command:

H8: Boot VV-0:vflash(cr)

The program will sign-on and prompt you for a file to flash. This file must be a 32K ROM image, complete with 32-bit checksum stored little-endian in the last four bytes. Once loaded and verified, the program will prompt you to start the flash operation. Note, using Ctrl-C at a prompt will cancel the program and return to the monitor. Once the flash has started, no interruption is possible. When the flash completes, the new ROM is verified and a message printed indicating whether or not it succeeded. (future versions may allow you to start over if the flash failed, or may verify and retry each page). An example session is:

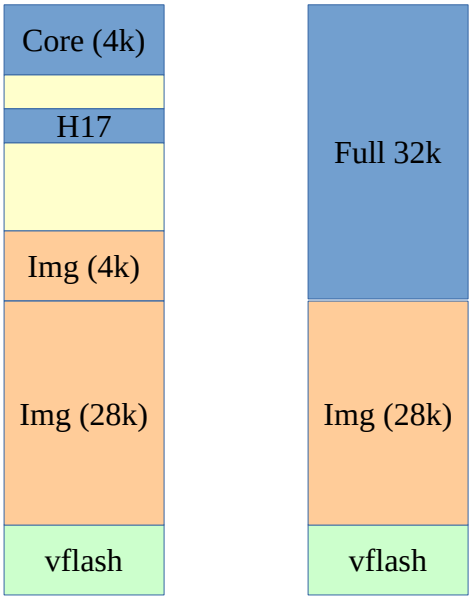
```
H8 Monitor v2.0(alpha1)

H8: Boot VV-0:vflash(cr)

VFLASH v1.0 - Update ROM from VDIP1
Enter ROM image file: H8MON2B.ROM
Press RETURN to start flash: (cr)
ROM update complete
Press RESET
(reset)
H8 Monitor v2.0(beta2)

H8:
```

During image file load and ROM flash, a “spinner” progress indicator is shown. Note that checksum computation, done at both the end of image load and the end of flash, does not advance the spinner.

Note, if the flash fails then the system may not be usable (depending on the state of the core monitor code). The ROM chip will need to be removed and flashed on an external device.

ROM Flash Details (Z80-based systems)

The flashing scheme depends on the legacy map, in order to be able to fully load the ROM image and verify it's integrity before starting the flash. The following figures and text describe the flash algorithm:

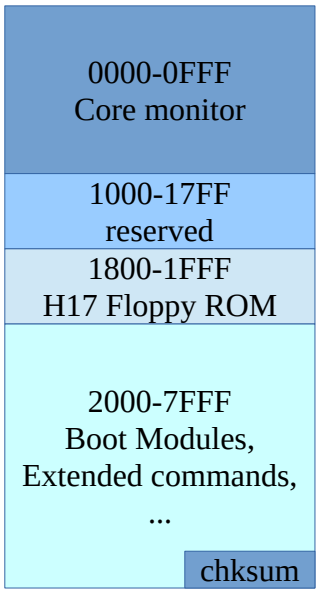


During phase one, legacy mapping is enabled. The ROM image file is loaded into memory at a point 4k below the full-ROM (32k) boundary. Once the image is verified, flashing can begin. The first step in the flash will write the first 4k (the core monitor) into the ROM. Then, mapping is switched to the extended mode, and the remaining 28k of the ROM can be written. After flashing completes, the entire (newly written) ROM is verified to ensure that the flashing was successful. Note that the act of flashing the ROM also writes to the corresponding RAM locations, as a side-effect. This means that the first 4k of the image gets overwritten when the last 4k of the ROM is flashed. This requires that the entire image file be selected and loaded again if a retry is to be done.

Note that flashing these devices requires writing 64 bytes at a time (with less than 150uS between bytes), and then polling the last byte written to wait for the flash write cycle to (begin and) complete.

ROM Layout

The ROM contents is organized as follows:



The core monitor contains the code to enter and execute commands and perform basic system initialization and management. It also contains the code to search the boot modules. There are also some well-defined entry points into the core monitor, which may be used by boot modules or even standalone programs.

The boot modules area is a variable-length region that may extend to the physical end of the ROM, with the exception that the last 4 physical bytes contain a 32-bit checksum, stored little-endian. Each boot module has a header area that allows the core monitor to search through them and to match one to a given boot target. By convention, modules are a multiple of 256 bytes in length. The modules contain basic information, such as strings to represent the device mnemonic and the number of units supported. It also contains the code necessary to boot from the device.

The default, power-on/RESET, mode (ORG0 off) only maps the “Core monitor” and “H17 Floppy” segments, leaving the rest of the 64K address space as RAM. This provides a “legacy” mode that is compatible with existing software and OSes. A program may turn on MEM1 (bit 00001000b in the control port at 362Q/0F2H) and cause the entire 32K of ROM to be mapped, however this requires caution as the software cannot be running in (or using) any of the RAM below the 32K boundary (8000H). The core monitor code enables this bit under strict conditions, in order to access the ROM extensions. Since the core monitor code is never self-modifying, and nothing uses that area as RAM, it is acceptable for the core monitor to alternate between the ROM modes without risk of crashing. The ROM flashing software also uses this control bit to flash, and verify, the whole ROM – however it is running entirely in high memory (and interrupts are disabled) and so conforms to the requirements.

ROM Extension Modules

Modules contain a header that is used to organize them, and also distinguish between boot modules and auxiliary commands or software extensions. All modules are a multiple of 256-byte pages in length. Modules must also start (be “ORGed”) on (256-byte) page boundaries. The basic module header is defined as follows:

```
first:  db      HIGH (last-first)       ; +0: num pages
        db      HIGH first              ; +1: ORG page
        db      255,0   ; +2,+3: phy drv base, num

        jmp     init    ; +4: init entry
        jmp     exec    ; +7: boot/execute entry

        db      '?'     ; +10: command letter
        db      -1      ; +11: front panel key
        db      0       ; +12: port, 0 if variable
        db      11111111b,11111111b,11111111b   ; +13: FP display
        db      'MYMOD',0 ; +16: mnemonic string

init:   xra     a       ; NC – no error
        ret

exec:   ret             ; boot returns on error,
...                     ; commands return on completion.
        rept    (($+0ffh) and 0ff00h)-$
        db      0ffh
        endm
last:   end
```

The physical drive field, +2, also indicates boot modules. A value of 0-199 indicates the base physical drive number for a boot module. A value of 255 indicates an auxiliary software extension.

For boot modules: fields +2, +3, +10, and +11 may be used to locate a module during boot, and so must be complete (although, for example, if a module cannot be activated from the front panel, then no valid front panel key should be given). Fields +12, +13, and +16 must also have correct information. Field +12 contains the base I/O port of the device, or “0” if the port may be configured as 170Q(78H) or 174Q(7CH). Field +13 contains the 3-digit front panel display 7-segment codes to form the mnemonic for the device. Field +16 contains a NUL-terminated ASCII string mnemonic for the device.

For extended commands: Extended commands are only available for the console mode of operation. Field +10 must contain an upper-case ASCII letter, not used by any other command, to represent the implemented command. Fields +2,+3 must contain 255,0.

For auxiliary software: Fields +10 and/or +11 are used to find and load the module. The value(s) depend on how the module is used. For example, the module to provide Cassette Tape load/save uses the field +11 with a value that cannot match a front panel key code (e.g. 88H). Fields +2,+3 must contain 255,0.

The entry point at +4 will be called when the module is loaded. It will be called once the module has been copied into RAM as indicated by the ORG page (field +1), and once the RAM map has been restored. Since the core ROM code is copied into RAM at start up, that code is available to the module

init routine. Note that the init routine might be called only once, even if the module function is executed multiple times.

The entry point at +7 will be called to execute the module function. In the case of boot modules, this is to attempt to boot from the indicated device (unit, etc). The return address for boot is the error/recovery routine. A call to boot should not return under normal circumstances (it should execute the OS from the boot device).

In the case of non-boot modules, the execute entry is used to perform the desired function. Additional information or parameters are either gleaned from the ROM environment or are queried from the user. The variable ‘lstcmd’ contains the command key/letter that was pressed. Front panel key codes are 0-15 with the high bit set (i.e. 80H-8FH). Console commands are the uppercase ASCII letters (‘A’-’Z’). When the operation has completed, it returns. Errors may need to jump directly to an error recovery entry in the core monitor code.