AltairZ80 Simulator Usage

25-Mar-2008

COPYRIGHT NOTICE

The following copyright notice applies to the SIMH source, binary, and documentation:

Based on work by Charles E. Owen (c) 1997. Additional device support by Howard M. Harte and Ernie Price.

This memorandum documents the Altair 8800 Simulator with Z80 support.

1 Simulator Files

scp.h

sim_console.h

sim_defs.h

sim_fio.h

sim_rev.h

sim_sock.h

sim_timer.h

sim_tmxr.h

scp.c

sim_console.c

sim_fio.c

sim_sock.c

sim_timer.c

sim_tmxr.c

AltairZ80/altairz80_defs.h

altairz80_cpu_nommu.c	
altairz80_cpu.c	
altairz80_dsk.c	
altairz80_hdsk.c	
altairz80_net.c	
altairz80_sio.c	
altairz80_sys.c	
flashwriter2.c	(Vector Graphic, Inc. FlashWriter II support by Howard M. Harte)
i8272.c	(Generic Intel 8272 Disk Controller by Howard M. Harte)
i8272.h	(Generic Intel 8272 Disk Controller by Howard M. Harte)
mfdc.c	(Micropolis FDC support by Howard M. Harte)
mfdc.h	(Micropolis FDC support by Howard M. Harte)
s100_disk1a.c	(CompuPro DISK1A Floppy Controller by Howard M. Harte)
s100_disk1a.c	(CompuPro DISK1A Floppy Controller by Howard M. Harte)
s100_disk2.c	(CompuPro DISK2 Hard Disk Controller by Howard M. Harte)
s100_fif.c	(IMSAI FIF Disk Controller by Ernie Price)
s100_mdriveh.c	(CompuPro M-DRIVE/H Controller by Howard M. Harte)
s100_mdsad.c	(North Star MDS-AD disk controller by Howard M. Harte)

s100_selchan.c	(CompuPro Selector Channel module by Howard M. Harte)
s100_ss1.c	(CompuPro System Support 1 module by Howard M. Harte)
sim_imd.c	(ImageDisk Disk Image File access module by Howard M. Harte)
sim_imd.h	(ImageDisk Disk Image File access module by Howard M. Harte)
vfdhd.c	(Micropolis FDC support by Howard M. Harte)
vfdhd.h	(Micropolis FDC support by Howard M. Harte)
insns.h	(8086 Disassembler by Simon Tatham and Julian Hall)
nasm.h	(8086 Disassembler by Simon Tatham and Julian Hall)
disasm.c	(8086 Disassembler by Simon Tatham and Julian Hall)
insnsa.c	(8086 Disassembler by Simon Tatham and Julian Hall)
insnsd.c	(8086 Disassembler by Simon Tatham and Julian Hall)
i86.h	(8086 CPU by Jim Hudgens)
i86_decode.c	(8086 CPU by Jim Hudgens)
i86_ops.c	(8086 CPU by Jim Hudgens)
i86_prim_ops.c	(8086 CPU by Jim Hudgens)

2 Revision History

- 29-Feb-2008, Howard M. Harte / Peter Schorn (added support for additional S100 and CompuPro hardware modules, added 8086 CPU)
- 29-Dec-2007, Howard M. Harte / Peter Schorn (added support for Vector Graphic Flashwriter II, Micropolis FDC, ImageDisk disk image File, IMSAI FIF disk controller, North Star MDS-AD disk controller)
- 21-Apr-2007, Peter Schorn (added documentation for UCSD Pascal II.0)
- 14-Apr-2007, Peter Schorn (added documentation for Howard M. Harte's hard disk extensions)
- 05-Jan-2007, Peter Schorn (added networking capability, included CP/NET and CPNOS)
- 26-Nov-2006, Peter Schorn (SIO can now be attached to a file, SIO rewritten for better efficiency)
- 15-Oct-2006, Peter Schorn (updated CP/M 2 operating system and application software description)
- 17-Sep-2006, Peter Schorn (added Altair Basic 5.0 to the sample software, corrected TTY/ANSI description)
- 21-Aug-2006, Peter Schorn (added MINOL and VTL-2 software, retyping courtesy of Emmanuel ROCHE, fixed a bug in memory breakpoints and added a create ("C") switch to the attach command)
- 24-Jan-2006, Peter Schorn (transcribed documentation to Word / PDF format)
- 05-Apr-2005, Peter Schorn (removed bogus t-state stepping support)
- 24-Jul-2004, Peter Schorn (updated CP/M 2 and SPL packages)
- 12-Apr-2004, Peter Schorn (added MAP/NOMAP capability to switch off key mapping)
- 26-Jan-2004, Peter Schorn (added support for t-state stepping)
- 25-Feb-2003, Peter Schorn (added support for real time simulation)
- 9-Oct-2002, Peter Schorn (added support for simulated hard disk)

- 28-Sep-2002, Peter Schorn (number of tracks per disk can be configured)
- 19-Sep-2002, Peter Schorn (added WARNROM feature)
- 31-Aug-2002, Peter Schorn (added extended ROM features suggested by Scott LaBombard)
- 4-May-2002, Peter Schorn (added description of MP/M II sample software)
- 28-Apr-2002, Peter Schorn (added periodic timer interrupts and three additional consoles)
- 15-Apr-2002, Peter Schorn (added memory breakpoint)
- 7-Apr-2002, Peter Schorn (added ALTAIRROM / NOALTAIRROM switch)

The first version of this document was written by Charles E. Owen

3 Background

The MITS (Micro Instrumentation and Telemetry Systems) Altair 8800 was announced on the January 1975 cover of Popular Electronics, which boasted you could buy and build this powerful computer kit for only $397. The kit consisted at that time of only the parts to build a case, power supply, card cage (18 slots), CPU card, and memory card with 256 *bytes* of memory. Still, thousands were ordered within the first few months after the announcement, starting the personal computer revolution as we know it today.

Many laugh at the small size of that first kit, noting there were no peripherals and the 256 byte memory size. But the computer was an open system, and by 1977 MITS and many other small startups had added many expansion cards to make the Altair quite a respectable little computer. The "Altair Bus" that made this possible was soon called the S-100 Bus, later adopted as an industry standard, and eventually became the IEE-696 Bus.

4 Hardware

We are simulating a fairly "loaded" Altair 8800 from about 1977, with the following configuration:

CPU Altair 8800 with Intel 8080 CPU board 62KB of RAM, 2K of EPROM with start boot ROM.

SIO MITS 88-2SIO Dual Serial Interface Board. Port 1 is assumed to be connected to a serial "glass TTY" that is your terminal running the Simulator.

PTR Paper Tape Reader attached to port 2 of the 2SIO board.

PTP Paper Tape Punch attached to port 2 of the 2SIO board. This also doubles as a printer port.

DSK MITS 88-DISK Floppy Disk controller with up to eight drives.

4.1 CPU

We have 2 CPU options that were not present on the original machine but are useful in the simulator. We also allow you to select memory sizes, but be aware that some sample software requires the full 64K (i.e. CP/M) and the MITS Disk Basic and Altair DOS require about a minimum of 24K.

SET CPU 8080	Simulates the 8080 CPU (default)
SET CPU Z80	Simulates the Z80 CPU. Note that some software (e.g. most original Altair software such as 4K Basic) requires an 8080 CPU and will not or not properly run on a Z80. This is mainly due to the use of the parity flag on the 8080 which has not always the same semantics on the Z80.
SET CPU 8086	Simulates 8086 CPU. This also enables 1’024 KB of memory by default.

SET CPU ITRAP
Causes the simulator to halt if an invalid opcode is detected (depending on the chosen CPU).

SET CPU NOITRAP
Does not stop on an invalid opcode. This is how the real 8080 works. Note that some software such as 4K Basic apparently tries to execute nonexistent 8080 instructions. Therefore it is advisable in this case to SET CPU NOITRAP.

SET CPU 4K
SET CPU 8K
SET CPU 12K
SET CPU 16K
… (in 4K steps)
SET CPU 64K
All these set various CPU memory configurations.

SET CPU MEMORY=<nnn>K
Sets the memory to <nnn> kilo bytes.

SET CPU BANKED
Enables the banked memory support. The simulated memory has eight banks with address range 0..'COMMON' (see registers below) and a common area from 'COMMON' to 0FFFF which is common to all banks. The currently active bank is determined by register 'BANK' (see below). You can only switch to banked memory if the memory is set to 64K. The banked memory is used by CP/M 3.

SET CPU NONBANKED
Disables banked memory support.

SET CPU CLEARMEMORY
Resets all internal memory to 0 and also resets the Memory Management Unit (MMU) such that all memory pages are RAM. Note that resetting the CPU does only clear the CPU registers but not the memory nor the MMU.

SET CPU ALTAIRROM
Enables the slightly modified but downwards compatible Altair boot ROM at addresses 0FF00 to 0FFFF. This is the default.

SET CPU NOALTAIRROM
Disables standard Altair ROM behavior.

SET CPU MMU
Enables the Memory Management Unit (MMU) and clock frequency support.

SET CPU NOMMU
Disables the Memory Management Unit (MMU) and clock frequency support. The simulator will run with maximum speed which can be more than twice the speed as with MMU enabled. This feature is only available for the Z80 and 8080 CPU using 64 KB.

SET CPU VERBOSE
Enables warning messages to be printed when the CPU attempts to write into ROM or into non-existing memory. Also prints a warning message if the CPU attempts to read from non-existing memory. Also shows the status of the MMU.

SET CPU QUIET
Suppresses all warning messages.

SET CPU STOPONHALT
Z80 or 8080 CPU stops when HALT instruction is encountered.

SET CPU LOOPONHALT
Z80 or 8080 CPU does not stop when a HALT instruction is encountered but waits for an interrupt to occur.

The BOOT EPROM card starts at address 0FF00 if it has been enabled by 'SET CPU ALTAIRROM'. Jumping to this address will boot drive 0 of the floppy controller (CPU must be set to ROM or equivalent code must be present). If no valid bootable software is present there the machine crashes. This is historically accurate behavior.

CPU registers include the following for the Z80 / 8080:

Name	**Size**	**Comment**	
PC	20	The Program Counter for all processors (8080, Z80, 8086)	

AF	16	The accumulator (8 bits) and the flag register
		F = S Z - AC - P/V N C
		S = Sign flag.
		Z = Zero Flag.
		- = not used (undefined)
		AC = Auxiliary Carry flag.
		P/V = Parity flag on 8080 (Parity / Overflow flag on Z80)
		- = not used (undefined)
		N = Internal sign flag
		C = Carry flag.	

BC	16	The BC register pair. Register B is the high 8 bits, C is the lower 8 bits	
DE	16	The DE register pair. Register D is the high 8 bits, E is the lower 8 bits.	
HL	16	The HL register pair. Register H is the high 8 bits, L is the lower 8 bits.	
AF1	16	The alternate AF register	(on Z80 only)
BC1	16	The alternate BC register	(on Z80 only)
DE1	16	The alternate DE register	(on Z80 only)
HL1	16	The alternate HL register	(on Z80 only)
IX	16	The IX index register	(on Z80 only)
IY	16	The IY index register	(on Z80 only)
IFF	8	Interrupt flag	(on Z80 only)
INT	8	Interrupt register	(on Z80 only)
SR	16	The front panel switches (use D SR 8 for 4k Basic).	
WRU	8	The interrupt character. This starts as 5 (Control-E) but some Altair software uses this keystroke so best to change this to something exotic such as 035 (which is Control-]).	
BANK	3	The currently active memory bank (if banked memory is activated - see memory options above)	
COMMON	16	The starting address of common memory. Originally set to 0C000 (note this setting must agree with the value supplied to GENCPM for CP/M 3 system generation)	

CLOCK	32	The clock speed of the simulated CPU in kHz or 0 to run at maximum speed. To set the clock speed for a typical 4 MHz Z80 CPU, use D CLOCK 4000. The CP/M utility SPEED measures the clock speed of the simulated CPU.

CPU registers include the following for the 8086:

Name	Size	Comment
AX	16	AX general purpose register
AL	8	low 8 bits of AX
AH	8	high 8 bits of AX
BX	16	BX general purpose register
BL	8	low 8 bits of BX
BH	8	high 8 bits of BX
CX	16	CX general purpose register
CL	8	low 8 bits of CX
CH	8	high 8 bits of CX
DX	16	DX general purpose register
DL	8	low 8 bits of DX
DH	8	high 8 bits of DX
BP	16	Base Pointer
SI	16	Source Index
DI	16	Destination Index
SP86	16	Stack Pointer
CS	16	Code Segment
DS	16	Data Segment
ES	16	Extra Segment
SS	16	Stack Segment
IP	16	Instruction Pointer, read-only, to set use PC (20 bit addresses are allowed)
FLAGS	16	Flags

```
15 14 13 12 11 10 09 08 07 06 05   04 03   02 01 00
 1  1  1  1 OF DF IF TF SF ZF Res. AF Res. PF  1 CF
```

OF = Overflow Flag

DF = Direction Flag

IF = Interrupt Flag

TF = Trace Flag

SF = Sign Flag

ZF = Zero Flag

AF = Auxiliary Carry Flag

PF = Parity Flag

CF = Carry Flag

4.2 The Serial I/O Card (2SIO)

This simple programmed I/O device provides 2 serial ports to the outside world, which could be hardware jumpered to support RS-232 plugs or a TTY current loop interface. The standard I/O addresses assigned by MITS was 10-11 (hex) for the first port, and 12-13 (hex) for the second. We follow this standard in the simulator.

The simulator directs I/O to/from the first port to the screen. The second port reads from an attachable "tape reader" file on input, and writes to an attachable "punch file" on output. These files are considered a simple stream of 8-bit bytes.

The SIO can be configured in SIMH with the following commands:

SET SIO ANSI	Bit 8 is set to zero on console output
SET SIO TTY	Bit 8 is not touched on console output
SET SIO ALL	Console input remain unchanged
SET SIO UPPER	Console input is transformed to upper case characters only (This feature is useful for most Altair software). SET SIO MAP must also have been executed for this option to take effect - otherwise no mapping occurs.
SET SIO BS	Map the delete character to backspace SET SIO MAP must also have been executed for this option to take effect - otherwise no mapping occurs.
SET SIO DEL	Map the backspace character to delete SET SIO MAP must also have been executed for this option to take effect - otherwise no mapping occurs.
SET SIO QUIET	Do not print warning messages
SET SIO VERBOSE	Print warning messages (useful for debugging) The register SIOWL determines how often the same warning is displayed. The default is 3.
SET SIO MAP	Enable mapping of characters (see also SET SIO ALL/UPPER/BS/DEL)
SET SIO NOMAP	Disable mapping of characters (see also SET SIO ALL/UPPER/BS/DEL)
SET SIO BELL	Displaying ^G (Control-G) sounds the bell
SET SIO NOBELL	Do not display ^G (Control-G, bell character. This feature is useful when a simulated program makes excessive use of the bell character.
SET SIO INTERRUPT	Status port 0 creates an interrupt when a character becomes available. The handler is at SIO register KEYBDH.
SET SIO NOINTERRUPT	Status port 0 does not create interrupts.
SET SIO SLEEP	Sleeps for SLEEP microseconds after a keyboard status check where no character was available. This is useful in many operating systems to avoid high real CPU usage in busy wait loops.
SET SIO NOSLEEP	Do not sleep after unsuccessful keyboard status checks.
SET SIO PORT=Port/Terminal/Read/NotRead/Write/Reset/Reset/Data	
	Port: two digit hex address of the new port
	Terminal: one digit decimal number of terminal line
	Read: two digit hex mask indicating the bit(s) set when a character is available
	NotRead: two digit hex mask indicating the bit(s) to set in case no character is available
	Write: two digit hex mask indicating the bits set when a character can be written
	Reset: T (port has reset command) or F (port has no reset command)
	Reset: two digit hex value of the reset command
	Data: T (port accepts OUT, i.e. is a data port) or F (port only has IN, i.e. is a status port).

You can also attach the SIO to a port or a file:

ATTACH SIO 23	Console IO goes via a Telnet connection on port 23 (often requires root privileges, you can also use another port and use telnet with this port)
ATTACH SIO <filename>	Console input is taken from the file with name <filename> and output goes to the SIMH console. Note that sometimes this does not work as expected since some application programs or operating system commands periodically check for input.
DETACH SIO	Console IO goes via the regular SIMH console

4.3 The SIMH pseudo device

The SIMH pseudo device facilitates the communication between the simulated ALTAIR and the simulator environment. This device defines a number of (most R/O) registers (see source code) which are primarily useful for debugging purposes.

The SIMH pseudo device can be configured with

SET SIMH QUIET	Do not print warning messages
SET SIMH VERBOSE	Print warning messages (useful for debugging)
SET SIMH TIMERON	Start periodic timer interrupts
SET SIMH TIMEROFF	Stop the periodic timer interrupts

The following variables determine the behavior of the timer:

TIMD	This is the delay between consecutive interrupts in milliseconds. Use D TIMD 20 for a 50 Hz clock.
TIMH	This is the address of the interrupt handler to call for a timer interrupt.

4.4 The 88-DISK controller

The MITS 88-DISK is a simple programmed I/O interface to the MITS 8-inch floppy drive, which was basically a Pertec FD-400 with a power supply and buffer board built-in. The controller supports neither interrupts nor DMA, so floppy access required the sustained attention of the CPU. The standard I/O addresses were 8, 9, and 0A (hex), and we follow the standard. Details on controlling this hardware are in the altairz80_dsk.c source file.

The only difference is that the simulated disks may be larger than the original ones: The original disk had 77 tracks while the simulated disks support up to 254 tracks (only relevant for CP/M). You can change the number of tracks per disk by setting the appropriate value in TRACKS[..]. For example "D TRACKS[0] 77" sets the number of tracks for disk 0 to the original number of 77. The command "D TRACKS[0-7] 77" changes the highest track number for all disks to 77.

For debugging purposes you can set the trace level of some disk I/O functions. To do so the following bits in TRACELEVEL (a register of the disk) have been defined with the following meaning:

1	Trace all IN and OUT instructions on the disk ports 8 and 9
2	Trace all read and writes to full sectors on the disk
4	Print a message whenever an unnecessary step-in or step out of the disk head occurs (often an indication of an infinite loop)
8	Print a message whenever the disk head appears to be waiting for a sector which does not show up (often an indication of an infinite loop)

For example the command "D DSK TRACELEVEL A" will trace options 2+8 from above.

The DSK device can be configured with

SET DSK<n> QUIET Do not print warning messages for disk <n>.

SET DSK<n> VERBOSE Print warning messages for disk <n> (useful for debugging). The register DSKWL determines how often the same warning is displayed. The default is 3.

SET DSK<n> WRTENB Allow write operations for disk <n>.

SET DSK<n> WRTLCK Disk <n> is locked, i.e. no write operations will be allowed.

4.5 The simulated hard disk

In order to increase the available storage capacity, the simulator features 8 simulated hard disks with a capacity of 8MB (HDSK0 to HDSK7). Currently only CP/M supports two hard disks as devices I: and J:.

For debugging purposes one can set the trace flag by executing the command "D HDSK TRACELEVEL 1". The default value is 0 (no trace).

The HDSK device can be configured with

SET HDSK<n> QUIET Do not print warning messages for hard disk <n>.

SET HDSK<n> VERBOSE Print warning messages for hard disk <n> (useful for debugging).

SET HDSK<n> WRTENB Allow write operations for hard disk <n>.

SET HDSK<n> WRTLCK Hard disk <n> is locked, i.e. no write operations will be allowed.

SET HDSK<n> FORMAT=<value> Set the hard disk to <value>. Possible values are HDSK (standard simulated AltairZ80 hard disk with 8’192 kB capacity), EZ80FL (128 kB flash), P112 (1’440 kB P112), SU720 (720 kB Super I/O) and SSSD8 (standard 8" SS SD floppy disk with 77 tracks of 26 sectors with 128 bytes, i.e. 250.25 kB capacity).

SET HDSK<n> GEOM=<t>/<s>/<l> Set the hard disk geometry to <t> tracks with <s> sectors with sector length <l>. Alternatively you can also use GEOM=T:<t>/N:<s>/S:<s>.

Note that the “Attach” command will choose the correct format based on the size of the attached file. In case the file does not yet exist it is created and the HDSK format will be used with the currently set capacity.

4.6 The simulated network

The simulator supports networking via sockets (TCP/IP) for simulating operating systems such as CP/NET (see section 5.4) and CPNOS (see section 5.5) which require at least two machines connected by a network.

The NET device can be configured with

SET NET CLIENT Puts this machine into client mode.

SET NET SERVER Puts this machine into server mode.

ATTACH NET <IP-addr>:<port> Attaches the machine to the given IP address and listening on the specified port. The IP address is given in a.b.c.d format (0 ≤ a, b, c, d ≤ 255). A typical example is “ATTACH NET 127.0.0.1:4000” which

	attaches to the local host at port 4000. Note that certain “small” port numbers might require special permissions.
DETACH NET	Detaches the machine from the network.

5 Sample Software

Running an Altair in 1977 you would be running either MITS Disk Extended BASIC, or the brand new and sexy CP/M Operating System from Digital Research. Or possibly, you ordered Altair DOS back when it was promised in 1975, and are still waiting for it to be delivered in early 1977.

We have samples of all three for you to check out. We can't go into the details of how they work, but we'll give you a few hints.

5.1 CP/M Version 2.2

This version is my own port of the standard CP/M to the Altair. There were some "official" versions but I don't have them. None were endorsed or sold by MITS to my knowledge, however.

To boot CP/M:

```
sim> attach dsk cpm2.dsk
sim> boot dsk
```

CP/M feels like DOS, sort of. DIR will work. I have included all the standard CP/M utilities, plus a few common public-domain ones. I also include the sources to the customized BIOS and some other small programs. TYPE will print an ASCII file. DUMP will dump a binary one. LS is a better DIR than DIR. ASM will assemble .ASM files to hex, LOAD will "load" them to binary format (.COM). ED is a simple editor, #A command will bring the source file to the buffer, T command will "type" lines, L will move lines, E exits the editor. 20L20T will move down 20 lines, and type 20. Very DECish. DDT is the debugger, DO is a batch-type command processor. A sample batch file that will assemble and write out the bootable CP/M image (on drive A) is "SYSCPM2.SUB". To run it, type "DO SYSCPM2".

In order to efficiently transfer files into the CP/M environment use the included program R <filename.ext>. If you have a file named foo.ext in the current directory (i.e. the directory where SIMH is), executing R FOO.EXT under CP/M will transfer the file onto the CP/M disk. Transferring a file from the CP/M environment to the SIMH environment is accomplished by W <filename.ext> for text files or by W <filename.ext> B for binary files. The simplest way for transferring multiple files is to create a ".SUB" batch file which contains the necessary R resp. W commands.

If you need more storage space you can use a simulated hard disk on drives I: and J:. To use do "attach HDSK0 hdi.dsk" and issue the "XFORMAT I:" resp. "XFORMAT J:" command from CP/M do initialize the disk to an empty state.

The disk "cpm2.dsk" contains the following files:

Name	Ext	Size	Comment
ASM	.COM	8K	CP/M assembler
BDOS	.MAC	66K	Basic Disk Operating System assembler source code
BOOT	.COM	2K	transfer control to boot ROM
BOOT	.MAC	2K	source for BOOT.COM
BOOTGEN	.COM	2K	put a program on the boot sectors
CBIOSX	.MAC	48K	CP/M 2 BIOS source for Altair

Name	Ext	Size	Comment
CCP	.MAC	26K	Console Command Processor assembler source code, original Digital Research
CCPZ	.MAC	50K	Console Command Processor assembler source code, Z80 replacement with some extra features
CCPZ	.TXT	40K	documentation for CCPZ
CFGCCP	.LIB	2K	configuration file for system generation, original CCP
CFGCCPZ	.LIB	2K	configuration file for system generation, with CCPZ
COPY	.COM	2K	copy disks
CPU	.COM	2K	get and set the CPU type (8080 or Z80)
CPU	.MAC	2K	source for CPU.COM
CREF80	.COM	4K	cross reference utility
DDT	.COM	6K	8080 debugger
DDTZ	.COM	10K	Z80 debugger
DIF	.COM	4K	determine differences between two files
DO	.COM	4K	batch processing with SuperSub (SUBMIT.COM replacement)
DSKBOOT	.MAC	8K	source for boot ROM
DUMP	.COM	2K	hex dump a file
ED	.COM	8K	line editor
ELIZA	.BAS	10K	Eliza game in Basic
EX	.MAC	48K	source for EX8080.COM, EXZ80DOC.COM, EXZ80ALL.COM
EX	.SUB	2K	benchmark execution of EX8080.COM,EXZ80DOC.COM,EXZ80ALL.COM
EX8080	.COM	12K	exercise 8080 instruction set
EXZ80ALL	.COM	12K	exercise Z80 instruction set, undefined status bits taken into account
EXZ80DOC	.COM	12K	exercise Z80 instruction set, no undefined status bits taken into account
FORMAT	.COM	2K	format disks
GO	.COM	0K	start the currently loaded program at 100H
HALT	.COM	2K	execute the HALT operation for returning to the sim> command prompt – useful as the last command in a script
HDSKBOOT	.MAC	6K	boot code for hard disk
L80	.COM	12K	Microsoft linker
LADDER	.COM	40K	game
LADDER	.DAT	2K	high score file for LADDER.COM
LIB80	.COM	6K	library utility
LOAD	.COM	2K	load hex files
LS	.COM	4K	directory utility

Name	Ext	Size	Comment
LU	.COM	20K	library utility
M80	.COM	20K	Microsoft macro assembler
MBASIC	.COM	24K	Microsoft Basic interpreter
MC	.SUB	2K	assemble and link an assembler program
MCC	.SUB	2K	read, assemble and link an assembler program
MCCL	.SUB	2K	assemble, link and produce listing
MOVER	.MAC	2K	moves operating system in place
OTHELLO	.COM	12K	Othello (Reversi) game
PIP	.COM	8K	Peripheral Interchange Program
PRELIM	.COM	2K	preliminary CPU tests
PRELIM	.MAC	6K	source code for PRELIM.COM
R	.COM	4K	read files from SIMH environment. Supports wild card expansion on UNIX and Windows for reading multiple files.
RSETSIMH	.COM	2K	reset SIMH interface
RSETSIMH	.MAC	2K	assembler source for RSETSIMH.COM
SHOWSEC	.COM	2K	show sectors on a disk
SID	.COM	8K	debugger for 8080
SPEED	.COM	2K	utility to measure the clock speed of the simulated CPU
STAT	.COM	6K	provide information about currently logged disks
SUBMIT	.COM	2K	batch processing
SURVEY	.COM	2K	system survey
SURVEY	.MAC	16K	assembler source for SURVEY.COM
SYSCOPY	.COM	2K	copy system tracks between disks
SYSCPM2	.SUB	2K	create CP/M 2 on drive A:, Digital Research CCP and BDOS
SYSCPM2Z	.SUB	2K	Create CP/M 2 on drive A:, CCPZ and Digital Research BDOS
TIMER	.COM	2K	perform various timer operations
TIMER	.MAC	2K	source code for TIMER.COM
UNCR	.COM	8K	un-crunch utility
UNERA	.COM	2K	un-erase a file
UNERA	.MAC	16K	source for UNERA.COM
USQ	.COM	2K	un-squeeze utility
W	.COM	2K	write files to SIMH environment. Supports CP/M wild card expansion for writing multiple files.
WM	.COM	12K	word master screen editor
WM	.HLP	4K	help file for WM.COM

Name	Ext	Size	Comment
WORM	.COM	4K	worm game for VT100 terminal
XFORMAT	.COM	2K	initialize a drive (floppy or hard disk)
XSUB	.COM	2K	support for DO.COM
ZAP	.COM	10K	SuperZap 5.2 disk editor configured for VT100
ZSID	.COM	10K	debugger for Z80
ZTRAN4	.COM	4K	translate 8080 mnemonics into Z80 equivalents

5.2 CP/M Version 3 with banked memory

CP/M 3 is the successor to CP/M 2.2. A customized BIOS (BIOS3.MAC) is included to facilitate modification if so desired. The defaults supplied in GENCPM.DAT for system generation can be used. BOOTGEN.COM is used to place the CP/M loader (LDR.COM) on the boot tracks of a disk.

Running CP/M 3 with banked memory:

```
sim> attach dsk cpm3.dsk
sim> reset cpu
sim> set cpu banked
sim> set cpu itrap
sim> boot dsk
```

Executing "DO SYSCPM3" will re-generate the banked version of CP/M 3. You can boot CP/M 3 with or without a Z80 CPU. The Z80 CPU is needed for both sysgens due to the use of BOOTGEN.COM which requires it.

The disk "cpm3.dsk" contains the following files:

Name	Ext	Size	Comment
ASM	.COM	8K	CP/M assembler
ASSIGN	.SYS	2K	
BDOS3	.SPR	10K	
BIOS3	.MAC	28K	CP/M 3 BIOS source for Altair SIMH
BIOS3	.SPR	4K	
BNKBDOS3	.SPR	14K	
BNKBIOS3	.SPR	4K	
BOOT	.COM	2K	transfer control to boot ROM
BOOTGEN	.COM	2K	put a program on the boot sectors
CCP	.COM	4K	
COPYSYS	.COM	2K	
CPM3	.SYS	18K	
CPMLDR	.MAC	38K	CP/M 3 loader assembler source
DATE	.COM	4K	date utility

Name	Ext	Size	Comment
DDT	.COM	6K	8080 debugger
DDTZ	.COM	10K	Z80 debugger
DEFS	.LIB	2K	include file for BIOS3.MAC to create banked CP/M 3
DEVICE	.COM	8K	
DIF	.COM	4K	determine differences between two files
DIR	.COM	16K	directory utility
DO	.COM	6K	batch processing (SUBMIT.COM)
DUMP	.COM	2K	
ED	.COM	10K	
ERASE	.COM	4K	
GENCOM	.COM	16K	
GENCPM	.COM	22K	
GENCPM	.DAT	4K	CP/M generation information for banked version
GENCPMNB	.DAT	4K	CP/M generation information for non-banked version
GET	.COM	8K	
HELP	.COM	8K	help utility
HELP	.HLP	62K	help files
HEXCOM	.CPM	2K	
HIST	.UTL	2K	
INITDIR	.COM	32K	
L80	.COM	12K	Microsoft linker
LDR	.COM	4K	CP/M loader with optimized loader BIOS
LDRBIOS3	.MAC	14K	optimized (for space) loader BIOS
LIB	.COM	8K	Digital Research librarian
LINK	.COM	16K	Digital Research linker
LOAD	.COM	2K	
M80	.COM	20K	Microsoft macro assembler
MC	.SUB	2K	assemble and link an assembler program
MCC	.SUB	2K	read, assemble and link an assembler program
PATCH	.COM	4K	
PIP	.COM	10K	Peripheral Interchange Program
PROFILE	.SUB	2K	commands to be executed at start up
PUT	.COM	8K	
R	.COM	4K	read files from SIMH environment

Name	Ext	Size	Comment
RENAME	.COM	4K	
RESBDOS3	.SPR	2K	
RMAC	.COM	14K	Digital Research macro assembler
RSETSIMH	.COM	2K	reset SIMH interface
SAVE	.COM	2K	
SCB	.MAC	2K	
SET	.COM	12K	
SETDEF	.COM	6K	
SHOW	.COM	10K	
SHOWSEC	.COM	4K	show sectors on a disk
SID	.COM	8K	8080 debugger
SYSCOPY	.COM	2K	copy system tracks between disks
SYSCPM3	.SUB	2K	create banked CP/M 3 system
TRACE	.UTL	2K	
TSHOW	.COM	2K	show split time
TSTART	.COM	2K	create timer and start it
TSTOP	.COM	2K	show final time and stop timer
TYPE	.COM	4K	
UNERA	.COM	2K	un-erase a file
W	.COM	4K	write files to SIMH environment
XREF	.COM	16K	cross reference utility
ZSID	.COM	10K	Z80 debugger

5.3 MP/M II with banked memory

MP/M II is an acronym for MultiProgramming Monitor Control Program for Microprocessors. It is a multi-user operating system for an eight bit microcomputer. MP/M II supports multiprogramming at each terminal. This version supports four terminals available via Telnet. To boot:

```
sim> attach dsk mpm.dsk

sim> set cpu itrap

sim> set cpu z80

sim> set cpu altairrom

sim> set cpu banked

sim> attach sio 23

sim> d common b000

sim> boot dsk
```

Now connect a Telnet session to the simulator and type "MPM" at the "A>" prompt. Now you can connect up to three additional terminals via Telnet to the Altair running MP/M II. To re-generate the system perform "DO SYSMPM" in the CP/M environment (not possible under MP/M since XSUB is needed).

The disk "mpm.dsk" contains the following files:

Name	Ext	Size	Comment
ABORT	.PRL	2K	abort a process
ABORT	.RSP	2K	
ASM	.PRL	10K	MP/M assembler
BNKBDOS	.SPR	12K	banked BDOS
BNKXDOS	.SPR	2K	banked XDOS
BNKXIOS	.SPR	4K	banked XIOS
BOOTGEN	.COM	2K	copy an executable to the boot section
CONSOLE	.PRL	2K	print console number
CPM	.COM	2K	return to CP/M
CPM	.MAC	2K	source for CPM.COM
DDT	.COM	6K	MP/M DDT
DDT2	.COM	6K	CP/M DDT
DDTZ	.COM	10K	CP/M DDT with Z80 support
DIF	.COM	4K	difference between two files
DIR	.PRL	2K	directory command
DO	.COM	2K	batch processing (SUBMIT.COM)
DSKRESET	.PRL	2K	disk reset command
DUMP	.MAC	6K	source for DUMP.PRL
DUMP	.PRL	2K	dump command
ED	.PRL	10K	MP/M line editor
ERA	.PRL	2K	erase command
ERAQ	.PRL	4K	erase command (verbose)
GENHEX	.COM	2K	
GENMOD	.COM	2K	
GENSYS	.COM	10K	
L80	.COM	12K	Microsoft linker
LDRBIOS	.MAC	14K	loader BIOS
LIB	.COM	8K	library utility
LINK	.COM	16K	linker
LOAD	.COM	2K	loader
M80	.COM	20K	Microsoft macro assembler

Name	Ext	Size	Comment
MC	.SUB	2K	assemble and link an assembler program
MCC	.SUB	2K	read, assemble and link an assembler program
MPM	.COM	8K	start MP/M II
MPM	.SYS	26K	MP/M system file
MPMD	.LIB	2K	define a banked system
MPMLDR	.COM	6K	MP/M loader without LDRBIOS
MPMSTAT	.BRS	6K	status of MP/M system
MPMSTAT	.PRL	6K	
MPMSTAT	.RSP	2K	
MPMXIOS	.MAC	26K	XIOS for MP/M
PIP	.PRL	10K	MP/M peripheral interchange program
PIP2	.COM	8K	CP/M peripheral interchange program
PRINTER	.PRL	2K	
PRLCOM	.PRL	4K	
R	.COM	4K	read a file from the SIMH environment
RDT	.PRL	8K	debugger for page relocatable programs
REN	.PRL	4K	rename a file
RESBDOS	.SPR	4K	non-banked BDOS
RMAC	.COM	14K	Digital Research macro assembler
RSETSIMH	.COM	2K	reset SIMH interface
SCHED	.BRS	2K	schedule a job
SCHED	.PRL	4K	
SCHED	.RSP	2K	
SDIR	.PRL	18K	fancy directory command
SET	.PRL	8K	set parameters
SHOW	.PRL	8K	show status of disks
SPOOL	.BRS	4K	spool utility
SPOOL	.PRL	4K	
SPOOL	.RSP	2K	
STAT	.COM	6K	CP/M stat command
STAT	.PRL	10K	MP/M stat command
STOPSPLR	.PRL	2K	stop spooler
SUBMIT	.PRL	6K	MP/M submit
SYSCOPY	.COM	2K	copy system tracks

Name	Ext	Size	Comment
SYSMPM	.SUB	2K	do a system generation
SYSTEM	.DAT	2K	default values for system generation
TMP	.SPR	2K	
TOD	.PRL	4K	time of day
TSHOW	.COM	2K	show split time
TSTART	.COM	2K	create timer and start it
TSTOP	.COM	2K	show final time and stop timer
TYPE	.PRL	2K	type a file on the screen
USER	.PRL	2K	set user area
W	.COM	4K	write a file to SIMH environment
XDOS	.SPR	10K	XDOS
XREF	.COM	16K	cross reference utility
XSUB	.COM	2K	for CP/M DO

5.4 CP/NET

This software is included as part of the archive **cpnet.zip**. To bring up the server component:

```
sim> attach dsk cpnetserver.dsk
sim> d common ab00
sim> set cpu   64k
sim> set cpu itrap
sim> set cpu z80
sim> set cpu altairrom
sim> set cpu banked
sim> set simh timeroff
sim> attach sio 23
sim> set net server
sim> at net 127.0.0.1:4000
sim> boot dsk
```

You can also execute “AltairZ80 cpnetserver” for the same effect or type “do cpnetserver<return>” at the “sim>” command prompt. Then connect via Telnet to the simulator and type “mpm <return>” at the “0A>” command prompt to start the MP/M CP/NET server.

To bring up a client, start another instance of AltairZ80 and type the following at the command prompt:

```
sim> attach dsk cpnetclient.dsk
sim> set cpu 64k
sim> set cpu noitrap
sim> set cpu z80
sim> set cpu altairrom
sim> set cpu nonbanked
sim> reset cpu
sim> set sio ansi
sim> set net client
sim> at net 127.0.0.1:4000
sim> boot dsk
```

You can also execute “AltairZ80 cpnetclient” for the same effect or type “do cpnetclient<return>” at the “sim>” command prompt. Then

```
A>cpnetldr<return>    ; loads CP/NET client
A>login<return>       ; to login
A>network b:=a:       ; to map server drive A: to client drive B:
A>dir b:              ; shows the contents of the server drive A:
```

The MP/M server is configured to accept one or two network clients. So you can repeat the previous procedure for a second client if you wish.

Note that all system specific sources (SNIOS.MAC, NETWRKIF.MAC, MPMXIOS.MAC) are included on cpnetclient.dsk respectively cpnetserver.dsk. When executing “GENSYS” for re-creating MP/M, keep in mind to include SERVER.RSP and NETWRKIF.RSP as this is not automatically suggested by GENSYS.

5.5 CPNOS

CPNOS is a thin client front-end for the CP/NET server. This software is also included as part of the archive **cpnet.zip**. In order to execute, first bring up a CP/NET server as described in section 5.4. Then for the client, start another instance of AltairZ80:

```
sim> set cpu 64k
sim> set cpu noitrap
sim> set cpu z80
sim> set cpu noaltairrom
sim> set cpu nonbanked
sim> reset cpu
sim> set sio ansi
sim> set net client
sim> at net 127.0.0.1:4000
sim> load cpnos.com f000
sim> g f000
```

For the same effect you can also execute “AltairZ80 cpnos” or type “do cpnos<return>” at the “sim>” command prompt. At the “LOGIN=” prompt, just type return and you will see the familiar “A>” prompt but the drive is the A: drive of the MP/M CP/NET server (you can also attach other disks to the server and they will become available to the CPNOS client). You can also connect a second CPNOS client to the same CP/NET server – further connection attempts will block after logging in until another CPNOS client is disconnected (e.g. by typing ^E to stop the simulator and then typing “bye<return>” at the simh command prompt). It is also possible to have both a CP/NET client and a CPNOS thin client connect to the same CP/NET server.

Note that all system specific sources (CPBIOS.MAC and CPNIOS.MAC) are included on cpnetclient.dsk.

5.6 CP/M application software

There is also a small collection of sample application software containing the following items:

- SPL	a Small Programming Language with a suite of sample programs
- PROLOGZ	a Prolog interpreter written in SPL with sources
- PASCFORM	a Pascal pretty printer written in Pascal
- Pascal MT+	Pascal language system needed to compile PASCFORM

The sample software comes on "app.dsk" and to use it do

```
   sim> attach dsk1 app.dsk
```

before booting CP/M.

The disk "app.dsk" contains the following files:

Name	Ext	Size	Comment
ACKER	.COM	2K	compute the Ackermann function
ACKER	.SPL	4K	compute the Ackermann function, SPL source
BOOTGEN	.COM	2K	copy the operating system to the rights sectors and tracks
BOOTGEN	.SPL	6K	SPL source for BOOTGEN.COM
C	.SUB	2K	batch file for compiling an SPL source file
CALC	.PRO	4K	Prolog demo program: Calculator
DIF	.COM	4K	
DIF	.SPL	10K	SPL source for DIF.COM
FAC	.COM	2K	compute the factorial
FAC	.SPL	4K	compute the factorial, SPL source
FAMILY	.PRO	4K	Prolog demo program: Family relations
FORMEL	.COM	4K	calculator
FORMEL	.SPL	6K	calculator, SPL source
INTEGER	.PRO	2K	Prolog demo program: Integer arithmetic
KNAKE	.PRO	2K	Prolog demo program: Logic puzzle
LINKMT	.COM	12K	Pascal MT+ 5.5 linker
MTERRS	.TXT	6K	Pascal MT+ error messages
MTPLUS	.000	14K	Pascal MT+ 5.5 compiler file
MTPLUS	.001	12K	Pascal MT+ 5.5 compiler file
MTPLUS	.002	8K	Pascal MT+ 5.5 compiler file
MTPLUS	.003	8K	Pascal MT+ 5.5 compiler file
MTPLUS	.004	18K	Pascal MT+ 5.5 compiler file

Name	Ext	Size	Comment
MTPLUS	.005	8K	Pascal MT+ 5.5 compiler file
MTPLUS	.006	6K	Pascal MT+ 5.5 compiler file
MTPLUS	.COM	36K	Pascal MT+ 5.5 compiler
PASCFORM	.COM	36K	Pascal formatter
PASCFORM	.PAS	54K	Pascal formatter source code
PASCFORM	.SUB	2K	create Pascal formatter
PASLIB	.ERL	24K	Pascal MT+ 5.5 run time library
PINST	.COM	4K	terminal installation program for PROLOGZ
PINST	.SPL	16K	terminal installation program for PROLOGZ, SPL source
PRIM	.COM	2K	compute prime numbers
PRIM	.SPL	2K	compute prime numbers, SPL source
PROLOGZ	.COM	16K	PROLOGZ interpreter and screen editor
PROLOGZ	.SPL	54K	SPL source for PROLOGZ
PROLOGZ	.TXT	40K	PROLOGZ documentation in German
PROLOGZU	.MAC	2K	helper functions for PROLOGZ in assembler
QUEEN	.PRO	2K	Prolog demo program: N-queens problem
READ	.COM	4K	transfer a file from the file system to the CP/M disk, see also WRITE.COM
READ	.SPL	10K	SPL source for READ.COM
RELDUMP	.COM	4K	dump a .REL file to the console
RELDUMP	.SPL	10K	dump a .REL file to the console, SPL source
SHOWSEC	.COM	2K	show a disk sector
SHOWSEC	.SPL	6K	SPL source for SHOWSEC.COM
SIEVE	.COM	2K	compute prime numbers with a sieve
SIEVE	.SPL	6K	compute prime numbers with a sieve, SPL source
SPEED	.COM	2K	utility to measure the clock speed of the simulated CPU
SPEED	.SPL	4K	SPL source for SPEED.COM
SPL	.COM	28K	the SPL compiler itself
SPL	.TXT	50K	SPL language and compiler documentation
SPLERROR	.DAT	8K	error messages of the compiler
SPLRTLB	.REL	2K	SPL runtime library
SYSCOPY	.COM	2K	copy the system tracks between disks
SYSCOPY	.SPL	6K	SPL source for SYSCOPY.COM
WC	.COM	6K	word count and query facility
WC	.SPL	14K	word count and query facility, SPL source

Name	Ext	Size	Comment
WRITE	.COM	2K	write a CP/M file to the file system, see also READ.COM
WRITE	.SPL	8K	SPL source for W.COM
XFORMAT	.COM	2K	format a regular disk or a hard disk
XFORMAT	.SPL	6K	SPL source for XFORMAT.COM

5.7 MITS Disk Extended BASIC Version 4.1

This was the commonly used software for serious users of the Altair computer. It is a powerful (but slow) BASIC with some extended commands to allow it to access and manage the disk. There was no operating system it ran under. This software is part of the archive **altsw.zip**. To boot:

```
sim> set cpu 8080                 ;Z80 will not work
sim> attach dsk mbasic.dsk
sim> set sio upper
sim> go ff00
MEMORY SIZE? [return]
LINEPRINTER? [C return]
HIGHEST DISK NUMBER? [0 return] (0 here = 1 drive system)
NUMBER OF FILES? [3 return]
NUMBER OF RANDOM FILES? [2 return]
44041 BYTES FREE
ALTAIR BASIC REV. 4.1
[DISK EXTENDED VERSION]
COPYRIGHT 1977 BY MITS INC.
OK
[MOUNT 0]
OK
[FILES]
```

5.8 Altair DOS Version 1.0

This was long promised but not delivered until it was almost irrelevant. A short attempted tour will reveal it to be a dog, far inferior to CP/M. This software is part of the archive **altsw.zip**. To boot:

```
sim> d tracks[0-7] 77             ;set to Altair settings
sim> set cpu altairrom
sim> attach dsk altdos.dsk
sim> set sio upper
sim> go ff00
MEMORY SIZE? [return]
INTERRUPTS? N [return]
```

```
HIGHEST DISK NUMBER? [0 return] (3 here = 4 drive system)
HOW MANY DISK FILES? [3 return]
HOW MANY RANDOM FILES? [2 return]

056449 BYTES AVAILABLE
DOS MONITOR VER 1.0
COPYRIGHT 1977 BY MITS INC
.[MNT 0]

.[DIR 0]
```

5.9 Altair Basic 3.2 (4k)

In order to run the famous 4k Basic, use the following commands (the trick is to get the Switch Register right). This software is part of the archive **altsw.zip**.

```
sim> set cpu 8080    ;note 4k Basic will not run on a Z80 CPU
sim> set sio upper   ;4k Basic does not like lower case letters as input
sim> set cpu noitrap ;4k Basic likes to execute non 8080 instructions-ignore
sim> set sio ansi    ;4k Basic produces 8-bit output, strip to seven bits
sim> d sr 8          ;good setting for the Switch Register
sim> load 4kbas.bin 0      ;load it at 0
sim> go 0            ;and start it
MEMORY SIZE? [return]
TERMINAL WIDTH? [return]
WANT SIN? [Y]

61911 BYTES FREE

BASIC VERSION 3.2
[4K VERSION]

OK
```

5.10 Altair 8k Basic

Running 8k Basic follows the procedure for 4k Basic. This software is part of the archive **altsw.zip**.

```
sim> set cpu 8080    ;note 8k Basic will not run on a Z80 CPU
sim> set sio upper   ;8k Basic does not like lower case letters as input
sim> set sio ansi    ;8k Basic produces 8-bit output, strip to seven bits
sim> d sr 8          ;good setting for the Switch Register
sim> load 8kbas.bin 0      ;load it at 0
sim> go 0            ;and start it
MEMORY SIZE? [A]

WRITTEN FOR ROYALTIES BY MICRO-SOFT

MEMORY SIZE? [return]
TERMINAL WIDTH? [return]
WANT SIN-COS-TAN-ATN? [Y]

58756 BYTES FREE
ALTAIR BASIC REV. 4.0
[EIGHT-K VERSION]
COPYRIGHT 1976 BY MITS INC.
OK
```

5.11 Altair Basic 4.0

This software is part of the archive **altsw.zip**. Execute the following commands to run Altair Extended Basic:

```
sim> set sio upper   ;Extended Basic requires upper case input
sim> set sio ansi    ;Extended Basic produces 8-bit output, strip to 7 bits
sim> d sr 8          ;good setting for the Switch Register
sim> load exbas.bin 0      ;load it at 0
sim> go 0            ;and start it
16384 Bytes loaded at 0.

MEMORY SIZE? [return]
WANT SIN-COS-TAN-ATN? [Y]

50606 BYTES FREE
ALTAIR BASIC REV. 4.0
[EXTENDED VERSION]
COPYRIGHT 1977 BY MITS INC.
OK
```

5.12 Altair Disk Extended Basic Version 300-5-C

This version of Basic was provided by Scott LaBombard. This software is part of the archive **altsw.zip**. To execute use the following commands:

```
sim> d tracks[0-7] 77      ;set to Altair settings
sim> at dsk extbas5.dsk
sim> g 0

MEMORY SIZE? [return]
LINEPRINTER? [C]
HIGHEST DISK NUMBER? [0]
HOW MANY FILES? [3]
HOW MANY RANDOM FILES? [3]

42082 BYTES FREE

ALTAIR DISK EXTENDED BASIC
VERSION 300-5-C [01NOV78]
COPYRIGHT 1978 BY MITS INC.

OK
```

5.13 Altair Disk Extended Basic Version 5.0

This version of Basic can be found on Andrew Kessel’s http://www.altairage.com/ site. Note that the MBL files on this site need to be converted to plain binary files using the Python script in the appendix. This software is part of the archive **altsw.zip**. To execute use the following commands:

```
sim> d tracks[0-7] 77       ;set to Altair settings
sim> at dsk disbas50.dsk
sim> d sr 8
sim> load disbas50.bin 0
sim> g 0

MEMORY SIZE? [return]
LINEPRINTER? [C]
HIGHEST DISK NUMBER? [return]
HOW MANY FILES? [3]
HOW MANY RANDOM FILES? [3]

41695 BYTES FREE
ALTAIR BASIC 5.0 [14JUL78]
[DISK EXTENDED VERSION]
COPYRIGHT 1978 BY MITS INC.
OK
```

5.14 Altair programming languages VTL-2 and MINOL

Emmanuel ROCHE retyped the manuals and assembler code for these two tiny languages. You need the archive **minolvtl.zip** which contains full documentation, sources and command files to execute the simulator.

5.15 UCSD Pascal II.0

The software is part of the **ucsd.zip** archive. To run it, type altairz80 ucsd at your command prompt or alternatively invoke altairz80 and type "do ucsd" at the "sim>" command prompt.

Useful hints:

- ? shows additional commands.
- V shows online volumes in the Filer.
- “ :” denotes the prefixed volume.
- Compiling the compiler and similar tools: Attach the correct disk and set the prefix to the name of the mounted volume. Then the include files will be found.
- To invoke the Basic compiler rename SYSTEM.COMPILER to PASCAL.COMPILER and then rename BASIC.COMPILER to SYSTEM.COMPILER.
- If you get "Please re-boot" after crunching a disk: type ^E, "g 0" and "pascal" to restart the system.

DSK0 contains a fairly complete development system with Pascal, Assembler and Basic.

```
Filer: G(et, S(ave, W(hat, N(ew, L(dir, R(em, C(hng, T(rans, D(ate, Q(uit [B]
    DSK0:
    SYSTEM.MICRO      19  9-Feb-79    10   512  Datafile
    SYSTEM.FILER      28 10-Apr-79    29   512  Codefile
    SYSTEM.EDITOR     45 10-Feb-79    57   512  Codefile
    SYSTEM.LINKER     22 10-Feb-79   102   512  Codefile
    SYSTEM.COMPILER   68  8-Feb-79   124   512  Codefile
    SYSTEM.SYNTAX     14  2-May-79   192   512  Textfile
    SETUP.CODE        25 14-May-79   206   512  Codefile
    BINDER.CODE        6  3-May-79   231   512  Codefile
    SYSTEM.MISCINFO    1 10-Feb-79   237   192  Datafile
    VT100GOTO.TEXT     4 10-Apr- 7   238   512  Textfile
    VT100GOTO.CODE     2 10-Apr- 7   242   512  Codefile
    SYSTEM.PASCAL     33 10-Apr- 7   244   512  Datafile
    SYSTEM.LIBRARY    17 10-Apr- 7   277   512  Datafile
    BASIC.COMPILER    30 11-Apr-79   294   512  Codefile
    LOOP.TEXT          4 10-Apr- 7   324   512  Textfile
    LOOP.CODE          4 10-Apr- 7   328   512  Codefile
    Z80.ERRORS         8 28-Mar-79   332    70  Datafile
    Z80.OPCODES        3 20-Dec-78   340    68  Datafile
    SYSTEM.ASSMBLER   53 13-Apr-79   343   512  Codefile
    < UNUSED >        98             396
    19/19 files<listed/in-dir>, 396 blocks used, 98 unused, 98 in largest
```

6 Special simulator features

6.1 Memory access breakpoints (8080/Z80 only)

In addition to the regular SIMH features such as PC queue, breakpoints etc., this simulator supports memory access breakpoints. A memory access breakpoint is triggered when a pre-defined memory location is accessed (read, write or update). To set a memory location breakpoint enter

```
sim> break -m <location>
```

Execution will stop whenever an operation accesses <location>. Note that a memory access breakpoint is not triggered by fetching code from memory (this is the job of regular breakpoints). This feature has been implemented by using the typing facility of the SIMH breakpoints.

6.2 Instruction breakpoints (8080/Z80/8086)

One can also set a breakpoint once a certain instruction is executed. To set an instruction breakpoint enter

```
sim> break –I <first byte of instruction>
```

Execution will stop whenever an instruction is executed which starts with <first byte of instruction>.

7 Brief summary of all major changes to the original Altair simulator

- Full support for Z80. CP/M software requiring a Z80 CPU now runs properly. DDTZ and PROLOGZ are included for demonstration purposes.
- Added banked memory support.
- PC queue implemented.
- Full assembler and dis-assembler support for Z80 and 8080 mnemonics. Depending on the current setting of the CPU, the appropriate mnemonics are used.
- The BOOT ROM was changed to fully load the software from disk. The original code basically loaded a copy of itself from the disk and executed it.
- ROM and memory size settings are now fully honored. This means that you cannot write into the ROM or outside the defined RAM (e.g. when the RAM size was truncated with the SET CPU commands). This feature allows programs which check for the size of available RAM to run properly (e.g. 4k Basic). In addition one can enable and disable the ROM which is useful in special cases (e.g. when testing a new version of the ROM).
- The console can also be used via Telnet. This is useful when a terminal is needed which supports cursor control such as a VT100. PROLOGZ for example has a built-in screen editor which works under Telnet.
- Simplified file exchange for CP/M. Using the READ program under CP/M one can easily import files into CP/M from the regular file system. Note that PIP does not work properly on non-text files on PTR.
- The WRITE program can be used to transfer files from the CP/M environment to the regular environment (binary or ASCII transfer).
- The last character read from PTR is always Control-Z (the EOF character for CP/M). This makes sure that PIP (Peripheral Interchange Program on CP/M) will terminate properly.
- Fixed a bug in the BIOS warm boot routine which caused CP/M to crash.
- Modified the BIOS for CP/M to support 8 disks.

– Added CP/M 3 banked version as sample software
– Changed from octal to hex
– Made the DSK and SIO device more robust (previously malicious code could crash the simulator)
– Added memory access break points
– Added periodic timer interrupts (useful for MP/M)
– Added additional consoles (useful for MP/M)
– Added MP/M II banked version as sample software
– Added networking support for CP/NET and CPNOS

8 Appendix: Python script for converting MBL files to plain binary files

```
#! /usr/bin/python
# -*- coding: UTF-8 -*-
# formatted for tab-stops 4
#
#     By Peter Schorn, peter.schorn@acm.org, September 2006
#
#
# Transform an MBL file to a binary file suitable for loading with SIMH
#
# Structure of MBL files is as follows:
# <byte>+ 0x00 0x00
# (checkSum<byte> 0x3c count<byte> loadLow<byte> loadHigh<byte>
#  <byte> * count)+
# where the lower 8 bit of the load address are determined by loadLow
# and the upper 8 bit of the load address are determined by loadHigh
# For checkSum the following rules hold:
#     For the first load record: 0
#     For the second load record: (sum of all load bytes of the first
#            load record) mod 256
#     For the third and higher load records: (sum of all load bytes of
#            the preceding load record - 1) mod 256
# A header with count = 0 or second position is unequal to 0x3c denotes
# end of file.

import sys

CHR0 = 2    # i.e. first header is prefixed by 2 chr(0)

if    len(sys.argv) <> 3:
      print 'Usage %s inputmbl.bin output.bin\n' % sys.argv[0]
      sys.exit(1)

f = file(sys.argv[1], 'rb')
b = f.read()
f.close()
i = b.index(chr(0) * CHR0 + chr(0) + chr(0x3c)) + CHR0 + 2
result = [chr(0)] * len(b)

k = 0
```

```
count = ord(b[i])
while count and (ord(b[i - 1]) == 0x3c):
      l = ord(b[i + 1]) + (ord(b[i + 2]) << 8)
      checkSum = 0
      for j in range(count):
            result[l + j] = b[i + 3 + j]
            checkSum += ord(b[i + 3 + j])
      expectedCheckSum = ord(b[i-2])
      receivedCheckSum = expectedCheckSum
      if    k == 1:
            receivedCheckSum = previousCheckSum & 255
      elif k > 1:
            receivedCheckSum = (previousCheckSum - 1) & 255
      if    receivedCheckSum <> expectedCheckSum:
            print 'Checksum error in record %i. Got %02X and expected %02X ' % (
                        k, receivedCheckSum, expectedCheckSum)
      i += count + 5
      count = ord(b[i])
      k += 1
      previousCheckSum = checkSum

i = len(result)
while result[i - 1] == chr(0):
      i -= 1

f = file(sys.argv[2], 'wb+')
for c in result[:i]:
      f.write(c)
f.close()
print '%i load records processed and %i bytes written to %s' % (k, i,
      sys.argv[2])
```

9 Appendix: How to bring up UCSD Pascal II.0 on SIMH

Precondition: Your current working directory contains the files mentioned below and altairz80 is available. The files *.raw.gz are here: http://bitsavers.org/bits/UCSD_Pascal/ucsd_II.0/

```
   U002A.5_Z80_SYS1.raw.gz    U012.1_SYS_2.raw.gz    ucsd    ucsd.dsk
```

Step 1: Get U002A.5_Z80_SYS1.raw.gz and U012.1_SYS_2.raw.gz from the distribution and gunzip “gunzip *gz”.

Step 2: Patch H command with ZAP (H command will otherwise indefinitely loop as patched command is a jump to itself). Execute altairz80 with "altairz80 ucsd", type ^E and "G 0" at the "sim>" command prompt. This brings you back to CP/M. At the "E>" type "ZAP" to invoke the disk editor for fixing on drive A: sector 13 on track 5 as shown below.

- Change drive to A (D command)
- Select track/Sector (S command)
- Select Track (T command) - type 5 <return>
- Select Sector (S command) - type C <return>
- Edit sector (E command)

change

```
   000060   C2 96 1A 21   FF FF C3 AC   1A C3 E9 1A   D1 2A 1A 03   |B..!C,.Ci.Q*..|
```

to

```
000060   C2 96 1A 21  FF FF C3 AC  1A C3 00 00  D1 2A 1A 03   |B..!C,.Ci.....|
```

- Commit change with ^W command
- Exit ZAP with Z command
- Exit simulator (^E and bye)

before

```
      Current Track    Current Sector     Current Block      Current Drive
       0005             000C               000B                A:
Offset    0  1  2  3   4  5  6  7   8  9  A  B   C  D  E  F    -----ASCII------
000000   09 29 29 EB  01 36 00 2A  94 02 19 09  C9 E1 22 90   |.))k.6.*....Ia".|
000010   02 E1 22 92  02 D1 EB 22  94 02 EB 2A  90 02 06 08   |.a"..Qk"..k*....|
000020   1A BE C2 BA  1A 23 13 10  F7 21 00 00  E5 2A 94 02   |.>B:.#..w!..e*..|
000030   EB 2A 92 02  73 23 72 C3  A4 03 D2 D3  1A 2A 94 02   |k*..s#rC$.RS.*..|
000040   11 08 00 19  5E 23 56 7B  3D B2 C2 96  1A 21 01 00   |....^#V{=2B..!..|
000050   C3 AC 1A 2A  94 02 11 0A  00 19 5E 23  56 7B 3D B2   |C,.*......^#V{=2|
000060   C2 96 1A 21  FF FF C3 AC  1A C3 E9 1A  D1 2A 1A 03   |B..!C,.Ci.Q*..|
000070   EB 73 23 72  D1 2A 1C 03  EB 73 23 72  C3 B0 03 07   |ks#rQ*..ks#rC0..|
```

after

```
      Current Track    Current Sector     Current Block      Current Drive
       0005             000C               000B                A:
Offset    0  1  2  3   4  5  6  7   8  9  A  B   C  D  E  F    -----ASCII------
000000   09 29 29 EB  01 36 00 2A  94 02 19 09  C9 E1 22 90   |.))k.6.*....Ia".|
000010   02 E1 22 92  02 D1 EB 22  94 02 EB 2A  90 02 06 08   |.a"..Qk"..k*....|
000020   1A BE C2 BA  1A 23 13 10  F7 21 00 00  E5 2A 94 02   |.>B:.#..w!..e*..|
000030   EB 2A 92 02  73 23 72 C3  A4 03 D2 D3  1A 2A 94 02   |k*..s#rC$.RS.*..|
000040   11 08 00 19  5E 23 56 7B  3D B2 C2 96  1A 21 01 00   |....^#V{=2B..!..|
000050   C3 AC 1A 2A  94 02 11 0A  00 19 5E 23  56 7B 3D B2   |C,.*......^#V{=2|
000060   C2 96 1A 21  FF FF C3 AC  1A C3 00 00  D1 2A 1A 03   |B..!C,.Ci.....|
000070   EB 73 23 72  D1 2A 1C 03  EB 73 23 72  C3 B0 03 07   |ks#rQ*..ks#rC0..|
```

Step 3: Proceed to UCSD Pascal by typing "pascal" <return> at the "E>" command prompt. Type <return> until you see the menu bar:

```
Command: E(dit, R(un, F(ile, C(omp, L(ink, X(ecute, A(ssem, D(ebug,? [II.0]
```

X(ecute setup and choose Prompted mode to update parameters as follows:

```
Command: E(dit, R(un, F(ile, C(omp, L(ink, X(ecute, A(ssem, D(ebug,? [II.0]x
    Execute what file? setup
    INITIALIZING...........................
    ...........................
    SETUP: C(HANGE T(EACH H(ELP Q(UIT [D1]
C
    CHANGE: S(INGLE) P(ROMPTED) R(ADIX)
        H(ELP) Q(UIT)
P
    FIELD NAME = BACKSPACE
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
        10      8     8        BS     ^H
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = EDITOR ACCEPT KEY
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
         0      0     0        NUL    ^@
Y   WANT TO CHANGE THIS VALUE? (Y,N,!)
```

```
26  NEW VALUE: 26
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
        32     26    1A        SUB    ^Z
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = EDITOR ESCAPE KEY
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
        33     27    1B        ESC    ^[
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = ERASE LINE
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
         0      0     0        NUL    ^@
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = ERASE SCREEN
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
         0      0     0        NUL    ^@
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = ERASE TO END OF LINE
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
         0      0     0        NUL    ^@
Y   WANT TO CHANGE THIS VALUE? (Y,N,!)
75  NEW VALUE: 75
     OCTAL DECIMAL HEXADECIMAL ASCII
       113     75    4B        K
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = ERASE TO END OF SCREEN
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
         0      0     0        NUL    ^@
Y   WANT TO CHANGE THIS VALUE? (Y,N,!)
74  NEW VALUE: 74
     OCTAL DECIMAL HEXADECIMAL ASCII
       112     74    4A        J
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = HAS 8510A
    CURRENT VALUE IS FALSE
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = HAS BYTE FLIPPED MACHINE
    CURRENT VALUE IS FALSE
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = HAS CLOCK
    CURRENT VALUE IS FALSE
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = HAS LOWER CASE
    CURRENT VALUE IS FALSE
Y   WANT TO CHANGE THIS VALUE? (Y,N,!)
T   NEW VALUE: T
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = HAS RANDOM CURSOR ADDRESSING
    CURRENT VALUE IS FALSE
Y   WANT TO CHANGE THIS VALUE? (Y,N,!)
T   NEW VALUE: T
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = HAS SLOW TERMINAL
    CURRENT VALUE IS FALSE
```

```
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = HAS WORD ORIENTED MACHINE
    CURRENT VALUE IS FALSE
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = KEY FOR BREAK
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
         0      0     0        NUL    ^@
Y   WANT TO CHANGE THIS VALUE? (Y,N,!)
3   NEW VALUE: 3
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
         3      3     3        ETX    ^C
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = KEY FOR FLUSH
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
         6      6     6        ACK    ^F
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = KEY FOR STOP
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
        23     19    13        DC3    ^S
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = KEY TO DELETE CHARACTER
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
        10      8     8        BS     ^H
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = KEY TO DELETE LINE
     OCTAL DECIMAL HEXADECIMAL ASCII
       177    127    7F        DEL
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = KEY TO END FILE
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
         3      3     3        ETX    ^C
Y   WANT TO CHANGE THIS VALUE? (Y,N,!)
26  NEW VALUE: 26
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
        32     26    1A        SUB    ^Z
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = KEY TO MOVE CURSOR DOWN
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
        12     10     A        LF     ^J
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = KEY TO MOVE CURSOR LEFT
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
        10      8     8        BS     ^H
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = KEY TO MOVE CURSOR RIGHT
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
        34     28    1C        FS     ^\
Y   WANT TO CHANGE THIS VALUE? (Y,N,!)
12  NEW VALUE: 12
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
        14     12     C        FF     ^L
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = KEY TO MOVE CURSOR UP
```

```
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
        37     31    1F        US     ^_
Y    WANT TO CHANGE THIS VALUE? (Y,N,!)
11   NEW VALUE: 11
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
        13     11     B        VT     ^K
N    WANT TO CHANGE THIS VALUE? (Y,N,!)
     FIELD NAME = LEAD IN FROM KEYBOARD
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
         0      0     0        NUL    ^@
N    WANT TO CHANGE THIS VALUE? (Y,N,!)
     FIELD NAME = LEAD IN TO SCREEN
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
         0      0     0        NUL    ^@
Y    WANT TO CHANGE THIS VALUE? (Y,N,!)
27   NEW VALUE: 27
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
        33     27    1B        ESC    ^[
N    WANT TO CHANGE THIS VALUE? (Y,N,!)
     FIELD NAME = MOVE CURSOR HOME
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
        15     13     D        CR     ^M
Y    WANT TO CHANGE THIS VALUE? (Y,N,!)
72   NEW VALUE: 72
     OCTAL DECIMAL HEXADECIMAL ASCII
       110     72    48        H
N    WANT TO CHANGE THIS VALUE? (Y,N,!)
     FIELD NAME = MOVE CURSOR RIGHT
     OCTAL DECIMAL HEXADECIMAL ASCII
        41     33    21        !
Y    WANT TO CHANGE THIS VALUE? (Y,N,!)
68   NEW VALUE: 68
     OCTAL DECIMAL HEXADECIMAL ASCII
       104     68    44        D
N    WANT TO CHANGE THIS VALUE? (Y,N,!)
     FIELD NAME = MOVE CURSOR UP
     OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
         0      0     0        NUL    ^@
Y    WANT TO CHANGE THIS VALUE? (Y,N,!)
65   NEW VALUE: 65
     OCTAL DECIMAL HEXADECIMAL ASCII
       101     65    41        A
N    WANT TO CHANGE THIS VALUE? (Y,N,!)
     FIELD NAME = NON PRINTING CHARACTER
     OCTAL DECIMAL HEXADECIMAL ASCII
        77     63    3F        ?
N    WANT TO CHANGE THIS VALUE? (Y,N,!)
     FIELD NAME = PREFIXED[DELETE CHARACTER]
     CURRENT VALUE IS FALSE
N    WANT TO CHANGE THIS VALUE? (Y,N,!)
     FIELD NAME = PREFIXED[EDITOR ACCEPT KEY]
     CURRENT VALUE IS FALSE
N    WANT TO CHANGE THIS VALUE? (Y,N,!)
```

```
    FIELD NAME = PREFIXED[EDITOR ESCAPE KEY]
    CURRENT VALUE IS FALSE
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = PREFIXED[ERASE LINE]
    CURRENT VALUE IS FALSE
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = PREFIXED[ERASE SCREEN]
    CURRENT VALUE IS FALSE
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = PREFIXED[ERASE TO END OF LINE]
    CURRENT VALUE IS FALSE
Y   WANT TO CHANGE THIS VALUE? (Y,N,!)
T   NEW VALUE: T
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = PREFIXED[ERASE TO END OF SCREEN]
    CURRENT VALUE IS FALSE
Y   WANT TO CHANGE THIS VALUE? (Y,N,!)
T   NEW VALUE: T
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = PREFIXED[KEY FOR BREAK]
    CURRENT VALUE IS FALSE
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = PREFIXED[KEY FOR FLUSH]
    CURRENT VALUE IS FALSE
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = PREFIXED[KEY FOR STOP]
    CURRENT VALUE IS FALSE
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = PREFIXED[KEY TO DELETE CHARACTER]
    CURRENT VALUE IS FALSE
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = PREFIXED[KEY TO DELETE LINE]
    CURRENT VALUE IS FALSE
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = PREFIXED[KEY TO END FILE]
    CURRENT VALUE IS FALSE
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = PREFIXED[KEY TO MOVE CURSOR DOWN]
    CURRENT VALUE IS FALSE
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = PREFIXED[KEY TO MOVE CURSOR LEFT]
    CURRENT VALUE IS FALSE
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = PREFIXED[KEY TO MOVE CURSOR RIGHT]
    CURRENT VALUE IS FALSE
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = PREFIXED[KEY TO MOVE CURSOR UP]
    CURRENT VALUE IS FALSE
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = PREFIXED[MOVE CURSOR HOME]
    CURRENT VALUE IS FALSE
Y   WANT TO CHANGE THIS VALUE? (Y,N,!)
T   NEW VALUE: T
```

```
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = PREFIXED[MOVE CURSOR RIGHT]
    CURRENT VALUE IS FALSE
Y   WANT TO CHANGE THIS VALUE? (Y,N,!)
T   NEW VALUE: T
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = PREFIXED[MOVE CURSOR UP]
    CURRENT VALUE IS FALSE
Y   WANT TO CHANGE THIS VALUE? (Y,N,!)
T   NEW VALUE: T
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = PREFIXED[NON PRINTING CHARACTER]
    CURRENT VALUE IS FALSE
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = SCREEN HEIGHT
     OCTAL DECIMAL HEXADECIMAL
        30     24    18
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = SCREEN WIDTH
     OCTAL DECIMAL HEXADECIMAL
       120     80    50
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = STUDENT
    CURRENT VALUE IS FALSE
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    FIELD NAME = VERTICAL MOVE DELAY
     OCTAL DECIMAL HEXADECIMAL
         5      5     5
Y   WANT TO CHANGE THIS VALUE? (Y,N,!)
0   NEW VALUE: 0
     OCTAL DECIMAL HEXADECIMAL
         0      0     0
N   WANT TO CHANGE THIS VALUE? (Y,N,!)
    CHANGE: S(INGLE) P(ROMPTED) R(ADIX)
Q       H(ELP) Q(UIT)
Q   SETUP: C(HANGE T(EACH H(ELP Q(UIT [D1]
D   QUIT: D(ISK) OR M(EMORY) UPDATE,
          R(ETURN) H(ELP) E(XIT)
M   QUIT: D(ISK) OR M(EMORY) UPDATE,
          R(ETURN) H(ELP) E(XIT)
E   QUIT: D(ISK) OR M(EMORY) UPDATE,
          R(ETURN) H(ELP) E(XIT)
```

Step 4: Rename NEW.MISCINFO to SYSTEM.MISCINFO

```
Command: E(dit, R(un, F(ile, C(omp, L(ink, X(ecute, A(ssem, D(ebug,? [II.0]
F
    Filer: G(et, S(ave, W(hat, N(ew, L(dir, R(em, C(hng, T(rans, D(ate, Q(uit [B]
L
    Dir listing of what vol ? *
    Filer: G(et, S(ave, W(hat, N(ew, L(dir, R(em, C(hng, T(rans, D(ate, Q(uit [B]L
    U002A.5:
    SYSTEM.STARTUP     5 28-Feb-79
    SYSTEM.MICRO      16  9-Feb-79
    Z80T.MICRO        19  9-Feb-79
    SYSTEM.FILER      28 10-Apr-79
    SYSTEM.PASCAL     33  7-Mar-79
    SYSTEM.EDITOR     45 10-Feb-79
    SYSTEM.LINKER     22 10-Feb-79
    SYSTEM.COMPILER   68  8-Feb-79
    SYSTEM.LIBRARY     8 17-Apr-79
    SYSTEM.SYNTAX     14  2-May-79
    SAMPLEGOTO.TEXT    4 17-Nov-78
    SETUP.CODE        25 14-May-79
    READ.ME.TEXT       4 17-Apr-79
    BINDER.CODE        6  3-May-79
    NEW.MISCINFO       1 10-Feb-79
    15/15 files<listed/in-dir>, 308 blocks used, 186 unused, 186 in largest
    Filer: G(et, S(ave, W(hat, N(ew, L(dir, R(em, C(hng, T(rans, D(ate, Q(uit [B]
C
    Change what file ?  NEW.MISCINFO
    Change to what ? SYSTEM.MISCINFO
```

Step 5: Delete SYSTEM.STARTUP (R command in Filer)

Step 6: Set date with D command in Filer

Step 7: Create new goto file for VT100 (VT100GOTO.TEXT)

```
(*$U-*)
PROGRAM DUMMY;
(* Direct cursor addressing for VT100 terminal *)
(* '[' after escape is done by BIOS - trick from Udo Munk *)
PROCEDURE FGOTOXY(X,Y:INTEGER);
BEGIN
  IF X<0 THEN X:=0;
  IF X>79 THEN X:=79;
  IF Y<0 THEN Y:=0;
  IF Y>23 THEN Y:=23;
  WRITE (CHR(27),Y+1,';',X+1,'H')
END;
BEGIN
END.
```

Take SAMPLEGOTO.TEXT as basis and modify using the editor. You can delete a complete line by moving the cursor to the line (^J for down, ^K for up) and then do D and <return> and ^Z.

Step 8: Compile result and save codefile (using Filer Save command).

Step 9: Update SYSTEM.PASCAL by X)cuting BINDER. When prompted for the file with the procedure type in VT100GOTO. The change takes effect after restart: Type H at top level and "pascal" at E> prompt.

10 Vector Graphic, Inc. Simulation

Howard M. Harte, hharte@hartetec.com

10.1 Overview

Vector Graphic is a early microcomputer from the mid 1970's, based on the S-100 bus using the Z80 microprocessor. There were several Vector Graphic models produced. Although primarily used with the CP/M operating system, it ran several others including OASIS, Micropolis Disk Operating System (MDOS), and Micropolis Z80 Operating System (MZOS).

Early Vector Graphic models used the Micropolis floppy disk controller and Micropolis floppy disk drives. Later models were designed with the integrated Floppy Drive/Hard Drive controller and used Tandon floppy drives. Almost all used unusual 100 track per inch 5.25" floppy drives and 16 sector 5.25" hard sector media. Some models included 8" floppy drives and hard disk drives.

Vector Graphic computers had many innovations such as the Flashwriter integrated video and keyboard controller. Vector Graphic is commonly known for their MEMORITE word processing application. When combined with the Flashwriter, the Vector Graphic MEMORITE software gave low cost word processing capability which had previously only been available with dedicated word processors.

Vector Graphic has a small but active user community. The following are links to resources and information about the Vector Graphic computer systems:

History and Background

http://en.wikipedia.org/wiki/Vector_Graphic

http://old-computers.com/museum/company.asp?st=1&l=V

http://www.classiccmp.org/dunfield/s100/index.htm#v1p

http://www.vintage-computer.com/vector1plus.shtml

http://retrotechnology.com/herbs_stuff/d_vector.html

http://www.vectorgraphics.org.uk/

Mailing List

http://h-net.msu.edu/cgi-bin/wa?A0=VECTOR-GRAPHIC

Documentation

http://www.hartetechnologies.com/manuals/Vector%20Graphics/

http://maben.homeip.net/static/S100/vector/index.html

Documentation / Disk Images

http://vector-archive.org

The Vector Graphic simulation was realized by making several architectural modifications to support additional disk controllers and the Flashwriter2 video card. The architectural modifications include the ability to install and uninstall devices in the simulator’s memory and I/O map at runtime, and pave the way for further extension of SIMH/AltairZ80 to support other hardware with a minimum of integration effort.

These additional devices specific to the Vector Graphic systems include:

MDSK – Micropolis FD Controller Board, memory mapped to 0xF800-0xFBFF

VFDHD – Vector HD-FD Controller Board, I/O Mapped to 0xC0-0xC3

FWII – Flashwriter 2 Video Card, memory mapped to 0xF000-0xF800

These devices can be enabled/disabled (installed/uninstalled) from the memory map with:

```
sim> set <device> ena – to enable the device.
sim> set <device> dis – to disable the device.
```

If there is an I/O or memory map conflict when enabling a device, the conflicting device must first be disabled.

In addition to the new devices added to SIMH/AltairZ80, additional ROM images are provided for the Vector 4.0C Monitor and the Vector 4.3 Monitor. The 4.0C Monitor uses the simulated serial port for I/O, and the 4.3 Monitor uses the Flashwriter2 video card for output and a simulated parallel keyboard for input. One of these monitors should be loaded at address 0xE000, depending on the simulated system configuration.

Generally, when using the HD-FD disk controller, you will need to use Monitor 4.3, since it supports booting from this controller. When using the Micropolis FD Controller board, you should use the 4.0C Monitor.

The simulator can be configured for a 48K Vector MZ or a 56K Vector MZ. Some boot disk images require a 48K configuration, and some require a 56K configuration. In the 48K configuration on a real Vector MZ system, an older version of the monitor ROM was at address 0xC000. Since the image for this ROM has not been obtained, a small "helper" ROM is loaded at address 0xC000, in addition to the 4.0C Monitor at 0xE000. The "helper" ROM redirects calls to perform terminal I/O to the corresponding entry points in the 4.0C monitor.

There are several configuration files that configure SIMH to simulate various Vector Graphic systems. These configuration files are the definitive reference for proper simulator configuration, and should be preferred over the following descriptions if there is any discrepancy. These configuration files are:

vgmz48k	Vector 48K MZ with Micropolis FD Controller
vgmz56k	Vector 56K MZ with Micropolis FD Controller
vgfdhd	Vector 56K System with HD-FD Disk Controller

Here are some sample configurations for 48K, 56K, and HD-FD Systems:

10.2 48K Vector MZ

```
sim> load MON40C.BIN e000   - load Vector 4.0C Monitor
sim> load MONC000.BIN c000  - load "Helper" ROM at 0xC000
sim> set mdsk membase=D800  - set Micropolis disk controller base address
sim> set mdsk enabled       - enable Micropolis disk controller
sim> attach mfdc0 VG02.VGI - attach disk to MDSK0 drive
```

When booting the 48K configuration, type:

```
sim> g e000
```

and at the `Mon>` prompt, you can boot from the disk controller by doing **`G D800`**.

10.3 56K Vector MZ

```
   sim> load MON40C.BIN e000         - load Vector 4.0C Monitor
   sim> set mdsk enabled             - enable Micropolis disk controller
   sim> attach mfdc0 VG00.VGI        - attach disk to MDSK0 drive
```

When booting the 56K configuration, type:

```
   sim> g e000
```

and at the `Mon>` prompt, you can boot from the disk controller by using the **B** (boot) command.

10.4 56K Vector with HD-FD Controller

```
   sim> set vfdhd enabled            - enable HD-FD controller
   sim> load MON43B.BIN e000         - load Vector 4.3 Monitor
   sim> att fwii0 f000               - enable the Flashwriter2 at F000.
   sim> set telnet 23                - set up telnet port for Flashwriter2
   sim> attach vfdhd1 VGBOOT.VGI     - attach disk to VFDHD1 drive
```

When booting the 56K HD-FD configuration, type:

```
   sim> g e000
```

You will then need to start a Telnet session to the simulator to use the simulated Flashwriter2. From a console window, do **`telnet localhost 23`**, or use your favorite telnet client, such as “Putty” under Windows. In the Telnet window, the 4.3 Monitor should sign on and at the `Mon>` prompt, you can boot from the disk controller by using the **B** (boot) command.

10.5 Notes on Simulated Hardware

The Vector HD-FD Controller supports four drives, one of which may be a Winchester (hard disk) drive. For the included VGBOOT.VGI disk image, CP/M is configured such that the VFDHD0 is drive "B" and VFDHD1 is drive "A." VFDHD2 is drive "C" and VFDHD3 is drive "D." The simulation assumes that whatever image is attached to VFDHD0 is a "Hard disk" image, so drive "B" using the VGBOOT.VGI disk image is not supported.

10.6 Notes on the Vector Graphic Disk Image (VGI) File Format

The Vector Graphic Disk Image (VGI) File Format uses a 275-byte sector format. This sector includes 256 bytes of User Data, and various other fields (metadata) used by controller hardware and the operating system running on the simulator.

The 275-byte sector format is as follows:

SYNC	TRACK	SECTOR	UNUSED	USER DATA	CHKSUM	ECC	ECC_VALID
1	1	1	10	256	1	4	1

SYNC	One byte, always 0xFF.
TRACK	Track number that this sector belongs to.
SECTOR	Sector number
UNUSED	Used by the operating system when running Micropolis DOS (MDOS) to store the load address and record length for the sector. This field is not used by CP/M.
USER DATA	256-bytes of user data
CHECKSUM	An operating system dependent checksum.
ECC	Four bytes of ECC code, generated and checked by the HD-FD Controller, but not used by the Micropolis FD Controller
ECC_VALID	One byte that contains 0xAA if the ECC field is valid. Disks written by the HD-FD controller typically have this field set to 0xAA to indicate that the ECC field should contain valid data. For disk images created by the Micropolis FD controller, this field is 0x00, since ECC is not supported. For disk images that were generated using the CPT program, this field will be 0x00 because the ECC bytes were not recoverable from the original disk. For disk images originally written with the HD-FD Controller, and imaged with Catweasel/Vector Graphic (CWVG) this field will be set to whatever it was set to on the original disk. This should be 0xAA.

11 IMSAI 8080 Simulation

IMSAI FIF Disk Controller support was added by Ernie Price.

11.1 Overview

The IMSAI FIF Disk Controller consists of an IFM (Interface Master Board) and a FIB (Floppy Disk Interface board) which interface the disk to the computer. The combination of FIB and IFM boards create an intelligent controller including DMA transfer, which permits the computer to perform other tasks during disk operations.

The FIF simulation can control up to eight disk drives. Commands include Read Clock and Data Bits, Write Sector, Read Sector, Verify Sector, Format Track, Write Deleted Data Sector Mark, Write Protect, Write Enable and Restore Drive. Logical and physical track addresses may be different. Cyclic redundancy checks are performed automatically. When an error is detected in reading or writing, the logic automatically retries up to 10 times.

Using the IMSAI FIF Controller, it is possible to run IMDOS 2.05 on the simulator.

Additional devices include:

FIF – IMSAI FIF Disk Controller, I/O Mapped to 0xFD

Since the IMSAI FIF and AltairZ80 HDSK devices both use I/O port 0xFD, the HDSK must be disabled before enabling the FIF:

```
sim> set hdsk dis – disable the AltairZ80 HDSK device.
sim> set fif ena – enable the IMSAI FIF device.
```

There is a configuration file that configures SIMH to simulate an IMSAI 8080 with FIF Disk Controller. This configuration file is the definitive reference for proper simulator configuration, and should be preferred over the following description if there is any discrepancy. This configuration file is:

imdos	IMSAI 8080 with FIF Disk Controller

11.2 IMSAI 8080 with FIF Disk Controller

```
sim> set hdsk dis                - disable AltairZ80 HDSK Controller
sim> set fif ena                 - enable IMSAI FIF Controller
sim> load IMSAI.BIN d800         - load IMSAI Monitor at 0xD800
sim> attach fif0 IMDOS_A.DSK     - attach disk to FIF0 drive
```

When booting the IMSAI 8080 with FIF Disk Controller, type:

```
sim> g d800
```

This will start the IMSAI Monitor, which will automatically boot from FIF0 if a valid boot disk image is attached.

12 North Star Horizon Simulation

North Star Horizon MDS-AD Disk Controller support was added by Howard M. Harte, hharte@hartetec.com.

12.1 Overview

The North Star MDS-AD disk controller is a double-sided, double-density disk controller supporting 48-TPI 5.25" Media, with 35 tracks, and 10 hard-sectors per track.

Using the North Star MDS-AD disk controller, it is possible to run CP/M 2.2 and North Star DOS on the simulator.

* * * NOTE: The MDS-AD Controller only supports Double-Density disks at this time * * *

Additional devices include:

MDSAD – North Star MDS-AD Disk Controller, Memory Mapped to 0xE800-0xEBFF.

There is a configuration file that configures SIMH to simulate a North Star Horizon System with an MDS-AD Disk Controller. This configuration file is the definitive reference for proper simulator configuration, and should be preferred over the following description if there is any discrepancy. This configuration file is:

`nshrz`	North Star Horizon with MDS-AD Disk Controller

12.2 North Star Horizon with FIF Disk Controller

```
   sim> set mdsad ena                - enable North Star MDS-AD Controller
   sim> attach mdsad0 D01B01.NSI     - attach CP/M boot disk to MDSAD0 drive
```

When booting the North Star Horizon, type:

```
   sim> boot mdsad0
```

This will start the CPU at 0xE800, which is the boot ROM for the MDS-AD disk controller.