AltairZ80 Simulator Usage
01-Aug-2011

COPYRIGHT NOTICE

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

Original code for the AltairZ80 part published in 2002-2011, written by Peter Schorn

Copyright (c) 2002-2011, Peter Schorn

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL PETER SCHORN BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of Peter Schorn shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from Peter Schorn.

Based on work by Charles E. Owen (c) 1997. Additional device support by Howard M. Harte and Ernie Price.
1 Simulator Files ........................................................................................................... 4
2 Revision History .......................................................................................................... 5
3 Background .................................................................................................................. 6
4 Hardware ...................................................................................................................... 7
  4.1 CPU .......................................................................................................................... 7
  4.2 The Serial I/O Card (2SIO) ...................................................................................... 10
  4.3 The SIMH pseudo device ......................................................................................... 12
  4.4 The 88-DISK controller ......................................................................................... 12
  4.5 The simulated hard disk ......................................................................................... 13
  4.6 The simulated network ......................................................................................... 14
5 Sample Software .......................................................................................................... 15
  5.1 CP/M Version 2.2 .................................................................................................... 15
  5.2 CP/M Version 3 with banked memory .................................................................... 18
  5.3 MP/M II with banked memory ................................................................................ 20
  5.4 CP/NET .................................................................................................................. 23
  5.5 CPNOS .................................................................................................................... 24
  5.6 CP/M application software ..................................................................................... 25
  5.7 MITS Disk Extended BASIC Version 4.1 ............................................................... 27
  5.8 Altair DOS Version 1.0 .......................................................................................... 27
  5.9 Altair Basic 3.2 (4k) .............................................................................................. 28
  5.10 Altair Basic 4.0 (4k) ............................................................................................ 28
  5.11 Altair 8k Basic ....................................................................................................... 29
  5.12 Altair Basic 4.0 ..................................................................................................... 30
  5.13 Altair Disk Extended Basic Version 300-5-C ......................................................... 30
  5.14 Altair Disk Extended Basic Version 5.0 ............................................................... 31
  5.15 Altair programming languages VTL-2 and MINOL .................................................. 31
  5.16 UCSD Pascal II.0 ................................................................................................. 31
6 Special simulator features ............................................................................................ 32
  6.1 Memory access breakpoints (8080/Z80 only) ......................................................... 32
  6.2 Instruction breakpoints (8080/Z80/8086) ................................................................. 32
7 Brief summary of all major changes to the original Altair simulator ................................ 33
8 Appendix: Python script for converting MBL files to plain binary files ....................... 33
9 Appendix: How to bring up UCSD Pascal II.0 on SIMH ............................................. 35
10 Vector Graphic, Inc. Simulation ................................................................................ 43
  10.1 Overview ............................................................................................................... 43
  10.2 48K Vector MZ ...................................................................................................... 44
  10.3 56K Vector MZ ...................................................................................................... 45
  10.4 56K Vector with HD-FD Controller ...................................................................... 45
  10.5 Notes on Simulated Hardware ............................................................................. 46
  10.6 Notes on the Vector Graphic Disk Image (VGI) File Format ................................ 46
11 IMSAI 8080 Simulation .............................................................................................. 47
  11.1 Overview .............................................................................................................. 47
  11.2 IMSAI 8080 with FIF Disk Controller .................................................................. 47
12 North Star Horizon Simulation .................................................................................. 48
  12.1 Overview .............................................................................................................. 48

SIMH AltairZ80 2 of 59
12.2 North Star Horizon with MDS-AD Disk Controller .......................................................... 48
13 Compupro 8-16 Simulation .................................................................................................. 49
  13.1 Overview ..................................................................................................................... 49
  13.2 DISK1A High Performance Floppy Disk Controller ................................................... 49
     13.2.1 DISK1A Controller Parameters ........................................................................... 49
     13.2.2 DISK1A Controller Limitations ........................................................................... 50
  13.3 DISK2 Compupro Hard Disk Controller ..................................................................... 50
     13.3.1 DISK2 Controller Parameters ............................................................................. 50
     13.3.2 DISK2 Controller Configuration Registers ......................................................... 51
  13.4 SELCHAN Compupro Selector Channel Controller .................................................. 51
     13.4.1 DISK2 Controller Parameters ............................................................................. 51
  13.5 DISK3 Viasyn ST-506 Hard Disk Controller ............................................................... 51
     13.5.1 DISK3 Controller Parameters ............................................................................. 51
     13.5.2 DISK3 Controller Configuration Registers ......................................................... 52
     13.5.3 DISK3 Controller Limitations ............................................................................. 52
14 Cromemco 4/16/64 FDC and CCS-2422 FDC Simulation .................................................. 53
  14.1 Overview ..................................................................................................................... 53
     14.1.1 CROMFDC Controller Parameters .................................................................... 53
     14.1.2 CROMFDC Controller Configuration Registers .............................................. 53
     14.1.3 CROMFDC Controller Limitations .................................................................... 54
15 Advanced Digital Corporation Super-Six Simulation .......................................................... 55
  15.1 Overview ..................................................................................................................... 55
     15.1.1 ADCS6 SBC Parameters .................................................................................... 55
     15.1.2 ADCS6 SBC Configuration Registers ............................................................... 55
     15.1.3 ADCS6 SBC Limitations .................................................................................... 56
16 N8VEM Single Board Computer Simulation ................................................................... 57
  16.1 Overview ..................................................................................................................... 57
     16.1.1 N8VEM SBC Parameters .................................................................................... 57
     16.1.2 N8VEM SBC Configuration Registers ............................................................... 57
     16.1.3 N8VEM SBC Limitations .................................................................................... 58
17 ImageDisk (IMD) Disk Image Support in SIMH ................................................................. 59
  17.1 Overview ..................................................................................................................... 59
  17.2 References ................................................................................................................... 59
This memorandum documents the Altair 8800 Simulator.

1 Simulator Files

scp.h
sim_console.h
simdefs.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)
n8vem.c (N8VEM Single-Board Computer I/O module by Howard M. Harte)
s100_64fdc.c (Cromemco 4FDC/16FDC/64FDC Floppy Controller by Howard M. Harte)
s100_adcs6.c (Advanced Digital Corporation (ADC) Super-Six CPU Board 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_disk3.c (CompuPro DISK3 Hard Disk Controller by Howard M. Harte)
s100_fif.c  (IMSAI FIF Disk Controller by Ernie Price)
s100_hdc1001.c  (Advanced Digital Corporation (ADC) HDC-1001 Hard Disk Controller by Howard M. Harte)
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_scp300f.c  (Seattle Computer Products SCP300F Support Board module 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)
wd179x.h  (WD179X support by Howard M. Harte)
wd179x.c  (WD179X support by Howard M. Harte)
isns.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)
isnisd.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

- 01-Aug-2011,  Peter Schorn (added some explanation to Altair Basic)
- 29-Sep-2009,  Peter Schorn (added debug flags to SIO, PTR and PTP)
- 18-Apr-2009,  Peter Schorn (fixed some errata in the manual found by Kim Sparre and added additional disk layouts to HDSK)
- 17-Aug-2008,  Peter Schorn (moved VERBOSE/QUIET for DSK and HDSK to debug flags)
- 03-Jul-2008,  Howard M. Harte (added support for hardware modules from Cromemco, Advanced Digital Corporation, Seattle Computer Products and N8VEM)
- 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)
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 0FFF. 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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td>20</td>
<td>The Program Counter for the 8080 and Z80</td>
</tr>
<tr>
<td>AF</td>
<td>16</td>
<td>The accumulator (8 bits) and the flag register $F = S \cdot Z - AC - P/V \cdot N \cdot C$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$S = \text{Sign flag.}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$Z = \text{Zero Flag.}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$- = \text{not used (undefined)}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$AC = \text{Auxiliary Carry flag.}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$P/V = \text{Parity flag on 8080 (Parity / Overflow flag on Z80)}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$- = \text{not used (undefined)}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$N = \text{Internal sign flag}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$C = \text{Carry flag.}$</td>
</tr>
<tr>
<td>BC</td>
<td>16</td>
<td>The BC register pair. Register B is the high 8 bits, C is the lower 8 bits</td>
</tr>
<tr>
<td>DE</td>
<td>16</td>
<td>The DE register pair. Register D is the high 8 bits, E is the lower 8 bits</td>
</tr>
</tbody>
</table>
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 1D (which is Control-]). But make sure you can actually create this character via the keyboard.

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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>AX</td>
<td>16</td>
<td>AX general purpose register</td>
</tr>
<tr>
<td>AL</td>
<td>8</td>
<td>low 8 bits of AX</td>
</tr>
<tr>
<td>AH</td>
<td>8</td>
<td>high 8 bits of AX</td>
</tr>
<tr>
<td>BX</td>
<td>16</td>
<td>BX general purpose register</td>
</tr>
<tr>
<td>BL</td>
<td>8</td>
<td>low 8 bits of BX</td>
</tr>
<tr>
<td>BH</td>
<td>8</td>
<td>high 8 bits of BX</td>
</tr>
<tr>
<td>CX</td>
<td>16</td>
<td>CX general purpose register</td>
</tr>
<tr>
<td>CL</td>
<td>8</td>
<td>low 8 bits of CX</td>
</tr>
<tr>
<td>CH</td>
<td>8</td>
<td>high 8 bits of CX</td>
</tr>
<tr>
<td>DX</td>
<td>16</td>
<td>DX general purpose register</td>
</tr>
<tr>
<td>DL</td>
<td>8</td>
<td>low 8 bits of DX</td>
</tr>
<tr>
<td>DH</td>
<td>8</td>
<td>high 8 bits of DX</td>
</tr>
<tr>
<td>BP</td>
<td>16</td>
<td>Base Pointer</td>
</tr>
<tr>
<td>SI</td>
<td>16</td>
<td>Source Index</td>
</tr>
<tr>
<td>DI</td>
<td>16</td>
<td>Destination Index</td>
</tr>
<tr>
<td>SP86</td>
<td>16</td>
<td>Stack Pointer</td>
</tr>
<tr>
<td>CS</td>
<td>16</td>
<td>Code Segment</td>
</tr>
</tbody>
</table>
DS 16  Data Segment
ES 16  Extra Segment
SS 16  Stack Segment
PCX 20  virtual 20-bit program counter
SPX 16  Stack Pointer
IP 16  Instruction Pointer, read-only, to set use PCX which allows 20 bit addresses
FLAGS 16  Flags

15 14 13 12 11 10  09  08  07  06  05   04  03   02  01  00
1 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

The CPU device supports the following debug flags (set with “SET CPU DEBUG=f1{;f}” or “SET CPU DEBUG” to enable all of them)

LOG_IN  Log all IN operations to the file specified with “SET DEBUG <file>”
LOG_OUT Log all OUT operations to the file specified with “SET DEBUG <file>”. Use “SET NODEBUG” to close the file. Also note that there is no logging if no file has been specified.

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. Furthermore, the SHOW command prints more information.

**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

The SIO device supports the following debug flags (set with "SET SIO DEBUG=f1{;f}" or "SET SIO DEBUG" to enable all of them):

- **IN**
  All IN operations on the SIO ports (status and data)

- **OUT**
  All OUT operations on the SIO ports (status and data)

- **CMD**
  All OUT operations which are interpreted as commands

- **VERBOSE**
  All warning messages (currently: none)
The PTP device supports the following debug flags (set with “SET PTP DEBUG=f1{:f}” or “SET PTP DEBUG” to enable all of them)

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>All IN operations on the PTP ports (status and data)</td>
</tr>
<tr>
<td>OUT</td>
<td>All OUT operations on the PTP ports (status and data)</td>
</tr>
<tr>
<td>CMD</td>
<td>All OUT operations which are interpreted as commands</td>
</tr>
<tr>
<td>VERBOSE</td>
<td>All warning messages (currently: use of unattached PTP)</td>
</tr>
</tbody>
</table>

The PTR device supports the following debug flags (set with “SET PTR DEBUG=f1{:f}” or “SET PTR DEBUG” to enable all of them)

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>All IN operations on the PTR ports (status and data)</td>
</tr>
<tr>
<td>OUT</td>
<td>All OUT operations on the PTR ports (status and data)</td>
</tr>
<tr>
<td>CMD</td>
<td>All OUT operations which are interpreted as commands</td>
</tr>
<tr>
<td>VERBOSE</td>
<td>All warning messages (currently: use of unattached PTR, attempt to read past end of attached file)</td>
</tr>
</tbody>
</table>

**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 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.

The SiMH device supports the following debug flags (set with “SET SIMH DEBUG=f1{:f}” or “SET SIMH DEBUG” to enable all of them)

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>All IN operations on the SIMH port</td>
</tr>
<tr>
<td>OUT</td>
<td>All OUT operations on the SIMH port</td>
</tr>
<tr>
<td>CMD</td>
<td>All illegal commands</td>
</tr>
<tr>
<td>VERBOSE</td>
<td>All other warning or error messages</td>
</tr>
</tbody>
</table>

**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.
The DSK device can be configured with

```
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.
```

The DSK device supports the following debug flags (set with "SET DSK DEBUG=f1{;f}" or "SET DSK DEBUG" to enable all of them)

- **IN** All IN operations on the controller port
- **OUT** All OUT operations on the controller port
- **READ** All read operations of the disk
- **WRITE** All write operations on the disk
- **SECTOR_STUCK** Warn when the controller appears to be stuck searching for a sector
- **TRACK_STUCK** Warn when the controller appears to be stuck searching for a track
- **VERBOSE** All other warning and error messages (e.g. disk is write locked, disk is not attached)

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:.

The HDSK device can be configured with

```
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)
  - OSB1 (100 kB Osborne 1 5.25" Single Side Single Density)
  - OSB2 (200 kB Osborne 1 5.25" Single Side Dual Density)
  - NSSS1 (175 kB Northstar Single Side Dual Density Format 1)
  - NSSS2 (175 kB Northstar Single Side Dual Density Format 2)
  - NSDS2 (350 kB Northstar Dual Side Dual Density Format 2)
  - VGSS (308 kB Vector Single Side Single Density)
  - VGDS (616 kB Vector Dual Side Single Density)
  - DISK1A (616 kB CompuPro Disk1A Single Side Single Density)
  - SSSD8 (standard 8" Single Side Single Density floppy disk with 77 tracks of 26 sectors with 128 bytes, i.e. 250.25 kB capacity, no skew).
  - SSSD8S (standard 8" Single Side Single Density floppy disk with 77 tracks of 26 sectors with 128 bytes, i.e. 250.25 kB capacity, standard skew factor 6).
```
– APPLE-DO (140 kB, Apple II, DOS 3.3)
– APPLE-PO (140 kB, Apple II, PRODOS)
– APPLE-D2 (140 kB, Apple II, DOS 3.3, 128 byte sectors for CP/M 2)
– APPLE-P2 (140 kB, Apple II, PRODOS, 128 byte sectors for CP/M 2)
– MITS (308 kB Altair standard disk with skew)
– MITS2 (1'016 kB Altair extended disk with skew)
– Note 1: The CP/M 3 implementation that comes with AltairZ80 automatically adapts to the attached hard disk.
– Note 2: The CP/M 2 implementation that comes with AltairZ80 can also adapt to all hard disk formats with 128 byte sectors. You need to set the correct format with this command after attaching a file.
– Note 3: When attaching a file to a hard disk, the format is guessed based on the size of the file. In case there is more than one possibility you may need to change the format after attaching.

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.

The HDSK device supports the following debug flags (set with “SET HDSK DEBUG=f1{;f}” or “SET HDSK DEBUG” to enable all of them)

- READ       All read operations of the disk
- WRITE      All write operations on the disk
- VERBOSE    All other warning and error messages (e.g. disk is write locked, disk is not attached)

### 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.

The NET device supports the following debug flags (set with “SET NET DEBUG=f1{;f}” or “SET NET DEBUG” to enable all of them)

- ACCEPT       Show a message when a connection is accepted
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Ext</th>
<th>Size</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASM</td>
<td>.COM</td>
<td>8K</td>
<td>CP/M assembler</td>
</tr>
<tr>
<td>BDOS</td>
<td>.MAC</td>
<td>66K</td>
<td>Basic Disk Operating System assembler source code</td>
</tr>
<tr>
<td>BOOT</td>
<td>.COM</td>
<td>2K</td>
<td>transfer control to boot ROM</td>
</tr>
<tr>
<td>BOOT</td>
<td>.MAC</td>
<td>2K</td>
<td>source for BOOT.COM</td>
</tr>
<tr>
<td>BOOTGEN</td>
<td>.COM</td>
<td>2K</td>
<td>put a program on the boot sectors</td>
</tr>
<tr>
<td>CBIOSX</td>
<td>.MAC</td>
<td>48K</td>
<td>CP/M 2 BIOS source for Altair</td>
</tr>
<tr>
<td>Name</td>
<td>Ext</td>
<td>Size</td>
<td>Comment</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td>------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CCP</td>
<td>.MAC</td>
<td>26K</td>
<td>Console Command Processor assembler source code, original Digital Research</td>
</tr>
<tr>
<td>CCPZ</td>
<td>.MAC</td>
<td>50K</td>
<td>Console Command Processor assembler source code, Z80 replacement with some extra features</td>
</tr>
<tr>
<td>CCPZ</td>
<td>.TXT</td>
<td>40K</td>
<td>documentation for CCPZ</td>
</tr>
<tr>
<td>CFGCCP</td>
<td>.LIB</td>
<td>2K</td>
<td>configuration file for system generation, original CCP</td>
</tr>
<tr>
<td>CFGCCPZ</td>
<td>.LIB</td>
<td>2K</td>
<td>configuration file for system generation, with CCPZ</td>
</tr>
<tr>
<td>COPY</td>
<td>.COM</td>
<td>2K</td>
<td>copy disks</td>
</tr>
<tr>
<td>CPU</td>
<td>.COM</td>
<td>2K</td>
<td>get and set the CPU type (8080 or Z80)</td>
</tr>
<tr>
<td>CPU</td>
<td>.MAC</td>
<td>2K</td>
<td>source for CPU.COM</td>
</tr>
<tr>
<td>CREF80</td>
<td>.COM</td>
<td>4K</td>
<td>cross reference utility</td>
</tr>
<tr>
<td>DDT</td>
<td>.COM</td>
<td>6K</td>
<td>8080 debugger</td>
</tr>
<tr>
<td>DDTZ</td>
<td>.COM</td>
<td>10K</td>
<td>Z80 debugger</td>
</tr>
<tr>
<td>DIF</td>
<td>.COM</td>
<td>4K</td>
<td>determine differences between two files</td>
</tr>
<tr>
<td>DO</td>
<td>.COM</td>
<td>4K</td>
<td>batch processing with SuperSub (SUBMIT.COM replacement)</td>
</tr>
<tr>
<td>DSKBOOT</td>
<td>.MAC</td>
<td>8K</td>
<td>source for boot ROM</td>
</tr>
<tr>
<td>DUMP</td>
<td>.COM</td>
<td>2K</td>
<td>hex dump a file</td>
</tr>
<tr>
<td>ED</td>
<td>.COM</td>
<td>8K</td>
<td>line editor</td>
</tr>
<tr>
<td>ELIZA</td>
<td>.BAS</td>
<td>10K</td>
<td>Eliza game in Basic</td>
</tr>
<tr>
<td>EX</td>
<td>.MAC</td>
<td>48K</td>
<td>source for EX8080.COM, EXZ80DOC.COM, EXZ80ALL.COM</td>
</tr>
<tr>
<td>EX</td>
<td>.SUB</td>
<td>2K</td>
<td>benchmark execution of EX8080.COM, EXZ80DOC.COM, EXZ80ALL.COM</td>
</tr>
<tr>
<td>EX8080</td>
<td>.COM</td>
<td>12K</td>
<td>exercise 8080 instruction set</td>
</tr>
<tr>
<td>EXZ80ALL</td>
<td>.COM</td>
<td>12K</td>
<td>exercise Z80 instruction set, undefined status bits taken into account</td>
</tr>
<tr>
<td>EXZ80DOC</td>
<td>.COM</td>
<td>12K</td>
<td>exercise Z80 instruction set, no undefined status bits taken into account</td>
</tr>
<tr>
<td>FORMAT</td>
<td>.COM</td>
<td>2K</td>
<td>format disks</td>
</tr>
<tr>
<td>GO</td>
<td>.COM</td>
<td>0K</td>
<td>start the currently loaded program at 100H</td>
</tr>
<tr>
<td>HALT</td>
<td>.COM</td>
<td>2K</td>
<td>execute the HALT operation for returning to the sim&gt; command prompt – useful as the last command in a script</td>
</tr>
<tr>
<td>HDSKBOOT</td>
<td>.MAC</td>
<td>6K</td>
<td>boot code for hard disk</td>
</tr>
<tr>
<td>L80</td>
<td>.COM</td>
<td>12K</td>
<td>Microsoft linker</td>
</tr>
<tr>
<td>LADDER</td>
<td>.COM</td>
<td>40K</td>
<td>game</td>
</tr>
<tr>
<td>LADDER</td>
<td>.DAT</td>
<td>2K</td>
<td>high score file for LADDER.COM</td>
</tr>
<tr>
<td>LIB80</td>
<td>.COM</td>
<td>6K</td>
<td>library utility</td>
</tr>
<tr>
<td>LOAD</td>
<td>.COM</td>
<td>2K</td>
<td>load hex files</td>
</tr>
<tr>
<td>LS</td>
<td>.COM</td>
<td>4K</td>
<td>directory utility</td>
</tr>
<tr>
<td>Name</td>
<td>Ext</td>
<td>Size</td>
<td>Comment</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>LU</td>
<td>.COM</td>
<td>20K</td>
<td>library utility</td>
</tr>
<tr>
<td>M80</td>
<td>.COM</td>
<td>20K</td>
<td>Microsoft macro assembler</td>
</tr>
<tr>
<td>MBASIC</td>
<td>.COM</td>
<td>24K</td>
<td>Microsoft Basic interpreter</td>
</tr>
<tr>
<td>MC</td>
<td>.SUB</td>
<td>2K</td>
<td>assemble and link an assembler program</td>
</tr>
<tr>
<td>MCC</td>
<td>.SUB</td>
<td>2K</td>
<td>read, assemble and link an assembler program</td>
</tr>
<tr>
<td>MCCL</td>
<td>.SUB</td>
<td>2K</td>
<td>assemble, link and produce listing</td>
</tr>
<tr>
<td>MOVER</td>
<td>.MAC</td>
<td>2K</td>
<td>moves operating system in place</td>
</tr>
<tr>
<td>OTHELLO</td>
<td>.COM</td>
<td>12K</td>
<td>Othello (Reversi) game</td>
</tr>
<tr>
<td>PIP</td>
<td>.COM</td>
<td>8K</td>
<td>Peripheral Interchange Program</td>
</tr>
<tr>
<td>PRELIM</td>
<td>.COM</td>
<td>2K</td>
<td>preliminary CPU tests</td>
</tr>
<tr>
<td>PRELIM</td>
<td>.MAC</td>
<td>6K</td>
<td>source code for PRELIM.COM</td>
</tr>
<tr>
<td>R</td>
<td>.COM</td>
<td>4K</td>
<td>read files from SIMH environment. Supports wild card expansion on UNIX and Windows for reading multiple files.</td>
</tr>
<tr>
<td>RSETSIMH</td>
<td>.COM</td>
<td>2K</td>
<td>reset SIMH interface</td>
</tr>
<tr>
<td>RSETSIMH</td>
<td>.MAC</td>
<td>2K</td>
<td>assembler source for RSETSIMH.COM</td>
</tr>
<tr>
<td>SHOWSEC</td>
<td>.COM</td>
<td>2K</td>
<td>show sectors on a disk</td>
</tr>
<tr>
<td>SID</td>
<td>.COM</td>
<td>8K</td>
<td>debugger for 8080</td>
</tr>
<tr>
<td>SPEED</td>
<td>.COM</td>
<td>2K</td>
<td>utility to measure the clock speed of the simulated CPU</td>
</tr>
<tr>
<td>STAT</td>
<td>.COM</td>
<td>6K</td>
<td>provide information about currently logged disks</td>
</tr>
<tr>
<td>SUBMIT</td>
<td>.COM</td>
<td>2K</td>
<td>batch processing</td>
</tr>
<tr>
<td>SURVEY</td>
<td>.COM</td>
<td>2K</td>
<td>system survey</td>
</tr>
<tr>
<td>SURVEY</td>
<td>.MAC</td>
<td>16K</td>
<td>assembler source for SURVEY.COM</td>
</tr>
<tr>
<td>SYSCOPY</td>
<td>.COM</td>
<td>2K</td>
<td>copy system tracks between disks</td>
</tr>
<tr>
<td>SYSCPM2</td>
<td>.SUB</td>
<td>2K</td>
<td>create CP/M 2 on drive A:, Digital Research CCP and BDOS</td>
</tr>
<tr>
<td>SYSCPM2Z</td>
<td>.SUB</td>
<td>2K</td>
<td>Create CP/M 2 on drive A:, CCPZ and Digital Research BDOS</td>
</tr>
<tr>
<td>TIMER</td>
<td>.COM</td>
<td>2K</td>
<td>perform various timer operations</td>
</tr>
<tr>
<td>TIMER</td>
<td>.MAC</td>
<td>2K</td>
<td>source code for TIMER.COM</td>
</tr>
<tr>
<td>UNCR</td>
<td>.COM</td>
<td>8K</td>
<td>un-crunch utility</td>
</tr>
<tr>
<td>UNERA</td>
<td>.COM</td>
<td>2K</td>
<td>un-erase a file</td>
</tr>
<tr>
<td>UNERA</td>
<td>.MAC</td>
<td>16K</td>
<td>source for UNERA.COM</td>
</tr>
<tr>
<td>USQ</td>
<td>.COM</td>
<td>2K</td>
<td>un-squeeze utility</td>
</tr>
<tr>
<td>W</td>
<td>.COM</td>
<td>2K</td>
<td>write files to SIMH environment. Supports CP/M wild card expansion for writing multiple files.</td>
</tr>
<tr>
<td>WM</td>
<td>.COM</td>
<td>12K</td>
<td>word master screen editor</td>
</tr>
<tr>
<td>WM</td>
<td>.HLP</td>
<td>4K</td>
<td>help file for WM.COM</td>
</tr>
<tr>
<td>Name</td>
<td>Ext</td>
<td>Size</td>
<td>Comment</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td>------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>WORM</td>
<td>.COM</td>
<td>4K</td>
<td>worm game for VT100 terminal</td>
</tr>
<tr>
<td>XFORMAT</td>
<td>.COM</td>
<td>2K</td>
<td>initialize a drive (floppy or hard disk)</td>
</tr>
<tr>
<td>XSUB</td>
<td>.COM</td>
<td>2K</td>
<td>support for DO.COM</td>
</tr>
<tr>
<td>ZAP</td>
<td>.COM</td>
<td>10K</td>
<td>SuperZap 5.2 disk editor configured for VT100</td>
</tr>
<tr>
<td>ZSID</td>
<td>.COM</td>
<td>10K</td>
<td>debugger for Z80</td>
</tr>
<tr>
<td>ZTRAN4</td>
<td>.COM</td>
<td>4K</td>
<td>translate 8080 mnemonics into Z80 equivalents</td>
</tr>
</tbody>
</table>

### 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:

```bash
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Ext</th>
<th>Size</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASM</td>
<td>.COM</td>
<td>8K</td>
<td>CP/M assembler</td>
</tr>
<tr>
<td>ASSIGN</td>
<td>.SYS</td>
<td>2K</td>
<td></td>
</tr>
<tr>
<td>BDOS3</td>
<td>.SPR</td>
<td>10K</td>
<td></td>
</tr>
<tr>
<td>BIOS3</td>
<td>.MAC</td>
<td>28K</td>
<td>CP/M 3 BIOS source for Altair SIMH</td>
</tr>
<tr>
<td>BIOS3</td>
<td>.SPR</td>
<td>4K</td>
<td></td>
</tr>
<tr>
<td>BNKBDOS3</td>
<td>.SPR</td>
<td>14K</td>
<td></td>
</tr>
<tr>
<td>BNKBIO3</td>
<td>.SPR</td>
<td>4K</td>
<td></td>
</tr>
<tr>
<td>BOOT</td>
<td>.COM</td>
<td>2K</td>
<td>transfer control to boot ROM</td>
</tr>
<tr>
<td>BOOTGEN</td>
<td>.COM</td>
<td>2K</td>
<td>put a program on the boot sectors</td>
</tr>
<tr>
<td>CCP</td>
<td>.COM</td>
<td>4K</td>
<td></td>
</tr>
<tr>
<td>COPYSYS</td>
<td>.COM</td>
<td>2K</td>
<td></td>
</tr>
<tr>
<td>CPM3</td>
<td>.SYS</td>
<td>18K</td>
<td></td>
</tr>
<tr>
<td>CPMLDR</td>
<td>.MAC</td>
<td>38K</td>
<td>CP/M 3 loader assembler source</td>
</tr>
<tr>
<td>DATE</td>
<td>.COM</td>
<td>4K</td>
<td>date utility</td>
</tr>
<tr>
<td>Name</td>
<td>Ext</td>
<td>Size</td>
<td>Comment</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td>------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>DDT</td>
<td>.COM</td>
<td>6K</td>
<td>8080 debugger</td>
</tr>
<tr>
<td>DDTZ</td>
<td>.COM</td>
<td>10K</td>
<td>Z80 debugger</td>
</tr>
<tr>
<td>DEFS</td>
<td>LIB</td>
<td>2K</td>
<td>include file for BIOS3.MAC to create banked CP/M 3</td>
</tr>
<tr>
<td>DEVICE</td>
<td>.COM</td>
<td>8K</td>
<td></td>
</tr>
<tr>
<td>DIF</td>
<td>.COM</td>
<td>4K</td>
<td>determine differences between two files</td>
</tr>
<tr>
<td>DIR</td>
<td>.COM</td>
<td>16K</td>
<td>directory utility</td>
</tr>
<tr>
<td>DO</td>
<td>.COM</td>
<td>6K</td>
<td>batch processing (SUBMIT.COM)</td>
</tr>
<tr>
<td>DUMP</td>
<td>.COM</td>
<td>2K</td>
<td></td>
</tr>
<tr>
<td>ED</td>
<td>.COM</td>
<td>10K</td>
<td></td>
</tr>
<tr>
<td>ERASE</td>
<td>.COM</td>
<td>4K</td>
<td></td>
</tr>
<tr>
<td>GENCOM</td>
<td>.COM</td>
<td>16K</td>
<td></td>
</tr>
<tr>
<td>GENCPM</td>
<td>.COM</td>
<td>22K</td>
<td></td>
</tr>
<tr>
<td>GENCPM</td>
<td>.DAT</td>
<td>4K</td>
<td>CP/M generation information for banked version</td>
</tr>
<tr>
<td>GENCPMNB</td>
<td>.DAT</td>
<td>4K</td>
<td>CP/M generation information for non-banked version</td>
</tr>
<tr>
<td>GET</td>
<td>.COM</td>
<td>8K</td>
<td>help utility</td>
</tr>
<tr>
<td>HELP</td>
<td>.COM</td>
<td>8K</td>
<td>help utility</td>
</tr>
<tr>
<td>HELP</td>
<td>.HLP</td>
<td>62K</td>
<td>help files</td>
</tr>
<tr>
<td>HEXCOM</td>
<td>.CPM</td>
<td>2K</td>
<td></td>
</tr>
<tr>
<td>HIST</td>
<td>.UTL</td>
<td>2K</td>
<td></td>
</tr>
<tr>
<td>INITDIR</td>
<td>.COM</td>
<td>32K</td>
<td></td>
</tr>
<tr>
<td>L80</td>
<td>.COM</td>
<td>12K</td>
<td>Microsoft linker</td>
</tr>
<tr>
<td>LDR</td>
<td>.COM</td>
<td>4K</td>
<td>CP/M loader with optimized loader BIOS</td>
</tr>
<tr>
<td>LDRBIOS3</td>
<td>.MAC</td>
<td>14K</td>
<td>optimized (for space) loader BIOS</td>
</tr>
<tr>
<td>LIB</td>
<td>.COM</td>
<td>8K</td>
<td>Digital Research librarian</td>
</tr>
<tr>
<td>LINK</td>
<td>.COM</td>
<td>16K</td>
<td>Digital Research linker</td>
</tr>
<tr>
<td>LOAD</td>
<td>.COM</td>
<td>2K</td>
<td></td>
</tr>
<tr>
<td>M80</td>
<td>.COM</td>
<td>20K</td>
<td>Microsoft macro assembler</td>
</tr>
<tr>
<td>MC</td>
<td>.SUB</td>
<td>2K</td>
<td>assemble and link an assembler program</td>
</tr>
<tr>
<td>MCC</td>
<td>.SUB</td>
<td>2K</td>
<td>read, assemble and link an assembler program</td>
</tr>
<tr>
<td>PATCH</td>
<td>.COM</td>
<td>4K</td>
<td></td>
</tr>
<tr>
<td>PIP</td>
<td>.COM</td>
<td>10K</td>
<td>Peripheral Interchange Program</td>
</tr>
<tr>
<td>PROFILE</td>
<td>.SUB</td>
<td>2K</td>
<td>commands to be executed at start up</td>
</tr>
<tr>
<td>PUT</td>
<td>.COM</td>
<td>8K</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>.COM</td>
<td>4K</td>
<td>read files from SIMH environment</td>
</tr>
<tr>
<td>Name</td>
<td>Ext</td>
<td>Size</td>
<td>Comment</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>RENAME</td>
<td>.COM</td>
<td>4K</td>
<td></td>
</tr>
<tr>
<td>RESBDOS3</td>
<td>.SPR</td>
<td>2K</td>
<td></td>
</tr>
<tr>
<td>RMAC</td>
<td>.COM</td>
<td>14K</td>
<td>Digital Research macro assembler</td>
</tr>
<tr>
<td>RSETSIMH</td>
<td>.COM</td>
<td>2K</td>
<td>reset SIMH interface</td>
</tr>
<tr>
<td>SAVE</td>
<td>.COM</td>
<td>2K</td>
<td></td>
</tr>
<tr>
<td>SCB</td>
<td>.MAC</td>
<td>2K</td>
<td></td>
</tr>
<tr>
<td>SET</td>
<td>.COM</td>
<td>12K</td>
<td></td>
</tr>
<tr>
<td>SETDEF</td>
<td>.COM</td>
<td>6K</td>
<td></td>
</tr>
<tr>
<td>SHOW</td>
<td>.COM</td>
<td>10K</td>
<td></td>
</tr>
<tr>
<td>SHOWSEC</td>
<td>.COM</td>
<td>4K</td>
<td>show sectors on a disk</td>
</tr>
<tr>
<td>SID</td>
<td>.COM</td>
<td>8K</td>
<td>8080 debugger</td>
</tr>
<tr>
<td>SUBMIT</td>
<td>.COM</td>
<td>6K</td>
<td>batch processing</td>
</tr>
<tr>
<td>SYSCOPY</td>
<td>.COM</td>
<td>2K</td>
<td>copy system tracks between disks</td>
</tr>
<tr>
<td>SYSCPM3</td>
<td>.SUB</td>
<td>2K</td>
<td>create banked CP/M 3 system</td>
</tr>
<tr>
<td>TRACE</td>
<td>.UTL</td>
<td>2K</td>
<td></td>
</tr>
<tr>
<td>TSHOW</td>
<td>.COM</td>
<td>2K</td>
<td>show split time</td>
</tr>
<tr>
<td>TSTART</td>
<td>.COM</td>
<td>2K</td>
<td>create timer and start it</td>
</tr>
<tr>
<td>TSTOP</td>
<td>.COM</td>
<td>2K</td>
<td>show final time and stop timer</td>
</tr>
<tr>
<td>TYPE</td>
<td>.COM</td>
<td>4K</td>
<td></td>
</tr>
<tr>
<td>UNERA</td>
<td>.COM</td>
<td>2K</td>
<td>un-erase a file</td>
</tr>
<tr>
<td>W</td>
<td>.COM</td>
<td>4K</td>
<td>write files to SIMH environment</td>
</tr>
<tr>
<td>XREF</td>
<td>.COM</td>
<td>16K</td>
<td>cross reference utility</td>
</tr>
<tr>
<td>ZSID</td>
<td>.COM</td>
<td>10K</td>
<td>Z80 debugger</td>
</tr>
</tbody>
</table>

### 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:

```plaintext
sim> attach dsk mpm.dsk
sim> set cpu intrap
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Ext</th>
<th>Size</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABORT</td>
<td>.PRL</td>
<td>2K</td>
<td>abort a process</td>
</tr>
<tr>
<td>ABORT</td>
<td>.RSP</td>
<td>2K</td>
<td></td>
</tr>
<tr>
<td>ASM</td>
<td>.PRL</td>
<td>10K</td>
<td>MP/M assembler</td>
</tr>
<tr>
<td>BNKBDO S</td>
<td>.SPR</td>
<td>12K</td>
<td>banked BDOS</td>
</tr>
<tr>
<td>BNKXDOS</td>
<td>.SPR</td>
<td>2K</td>
<td>banked XDOS</td>
</tr>
<tr>
<td>BNKXIOS</td>
<td>.SPR</td>
<td>4K</td>
<td>banked XIOS</td>
</tr>
<tr>
<td>BOOTGEN</td>
<td>.COM</td>
<td>2K</td>
<td>copy an executable to the boot section</td>
</tr>
<tr>
<td>CONSO LE</td>
<td>.PRL</td>
<td>2K</td>
<td>print console number</td>
</tr>
<tr>
<td>CPM</td>
<td>.COM</td>
<td>2K</td>
<td>return to CP/M</td>
</tr>
<tr>
<td>CPM</td>
<td>.MAC</td>
<td>2K</td>
<td>source for CPM.COM</td>
</tr>
<tr>
<td>DDT</td>
<td>.COM</td>
<td>6K</td>
<td>MP/M DDT</td>
</tr>
<tr>
<td>DDT2</td>
<td>.COM</td>
<td>6K</td>
<td>CP/M DDT</td>
</tr>
<tr>
<td>DDTZ</td>
<td>.COM</td>
<td>10K</td>
<td>CP/M DDT with Z80 support</td>
</tr>
<tr>
<td>DIF</td>
<td>.COM</td>
<td>4K</td>
<td>difference between two files</td>
</tr>
<tr>
<td>DIR</td>
<td>.PRL</td>
<td>2K</td>
<td>directory command</td>
</tr>
<tr>
<td>DO</td>
<td>.COM</td>
<td>2K</td>
<td>batch processing (SUBMIT.COM)</td>
</tr>
<tr>
<td>DSKRESE T</td>
<td>.PRL</td>
<td>2K</td>
<td>disk reset command</td>
</tr>
<tr>
<td>DUMP</td>
<td>.MAC</td>
<td>6K</td>
<td>source for DUMP.PRL</td>
</tr>
<tr>
<td>DUMP</td>
<td>.PRL</td>
<td>2K</td>
<td>dump command</td>
</tr>
<tr>
<td>ED</td>
<td>.PRL</td>
<td>10K</td>
<td>MP/M line editor</td>
</tr>
<tr>
<td>ERA</td>
<td>.PRL</td>
<td>2K</td>
<td>erase command</td>
</tr>
<tr>
<td>ERAQ</td>
<td>.PRL</td>
<td>4K</td>
<td>erase command (verbose)</td>
</tr>
<tr>
<td>GENHEX</td>
<td>.COM</td>
<td>2K</td>
<td></td>
</tr>
<tr>
<td>GENMOD</td>
<td>.COM</td>
<td>2K</td>
<td></td>
</tr>
<tr>
<td>GENSYS</td>
<td>.COM</td>
<td>10K</td>
<td></td>
</tr>
<tr>
<td>L80</td>
<td>.COM</td>
<td>12K</td>
<td>Microsoft linker</td>
</tr>
<tr>
<td>LDRBIOS</td>
<td>.MAC</td>
<td>14K</td>
<td>loader BIOS</td>
</tr>
<tr>
<td>LIB</td>
<td>.COM</td>
<td>8K</td>
<td>library utility</td>
</tr>
<tr>
<td>LINK</td>
<td>.COM</td>
<td>16K</td>
<td>linker</td>
</tr>
<tr>
<td>LOAD</td>
<td>.COM</td>
<td>2K</td>
<td>loader</td>
</tr>
<tr>
<td>M80</td>
<td>.COM</td>
<td>20K</td>
<td>Microsoft macro assembler</td>
</tr>
<tr>
<td>Name</td>
<td>Ext</td>
<td>Size</td>
<td>Comment</td>
</tr>
<tr>
<td>---------</td>
<td>-----</td>
<td>------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>MC</td>
<td>.SUB</td>
<td>2K</td>
<td>assemble and link an assembler program</td>
</tr>
<tr>
<td>MCC</td>
<td>.SUB</td>
<td>2K</td>
<td>read, assemble and link an assembler program</td>
</tr>
<tr>
<td>MPM</td>
<td>.COM</td>
<td>8K</td>
<td>start MP/M II</td>
</tr>
<tr>
<td>MPM</td>
<td>.SYS</td>
<td>26K</td>
<td>MP/M system file</td>
</tr>
<tr>
<td>MPMD</td>
<td>.LIB</td>
<td>2K</td>
<td>define a banked system</td>
</tr>
<tr>
<td>MPMLDR</td>
<td>.COM</td>
<td>6K</td>
<td>MP/M loader without LDRBIOS</td>
</tr>
<tr>
<td>MPMSTAT</td>
<td>.BRS</td>
<td>6K</td>
<td>status of MP/M system</td>
</tr>
<tr>
<td>MPMSTAT</td>
<td>.PRL</td>
<td>6K</td>
<td></td>
</tr>
<tr>
<td>MPMSTAT</td>
<td>.RSP</td>
<td>2K</td>
<td></td>
</tr>
<tr>
<td>MPMXIOS</td>
<td>.MAC</td>
<td>26K</td>
<td>XIOS for MP/M</td>
</tr>
<tr>
<td>PIP</td>
<td>.PRL</td>
<td>10K</td>
<td>MP/M peripheral interchange program</td>
</tr>
<tr>
<td>PIP2</td>
<td>.COM</td>
<td>8K</td>
<td>CP/M peripheral interchange program</td>
</tr>
<tr>
<td>PRINTER</td>
<td>.PRL</td>
<td>2K</td>
<td></td>
</tr>
<tr>
<td>PRLCOM</td>
<td>.PRL</td>
<td>4K</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>.COM</td>
<td>4K</td>
<td>read a file from the SIMH environment</td>
</tr>
<tr>
<td>RDT</td>
<td>.PRL</td>
<td>8K</td>
<td>debugger for page relocatable programs</td>
</tr>
<tr>
<td>REN</td>
<td>.PRL</td>
<td>4K</td>
<td>rename a file</td>
</tr>
<tr>
<td>RESBDOS</td>
<td>.SPR</td>
<td>4K</td>
<td>non-banked BDOS</td>
</tr>
<tr>
<td>RMAC</td>
<td>.COM</td>
<td>14K</td>
<td>Digital Research macro assembler</td>
</tr>
<tr>
<td>RSETSIMH</td>
<td>.COM</td>
<td>2K</td>
<td>reset SIMH interface</td>
</tr>
<tr>
<td>SCHED</td>
<td>.BRS</td>
<td>2K</td>
<td>schedule a job</td>
</tr>
<tr>
<td>SCHED</td>
<td>.PRL</td>
<td>4K</td>
<td></td>
</tr>
<tr>
<td>SCHED</td>
<td>.RSP</td>
<td>2K</td>
<td></td>
</tr>
<tr>
<td>SDIR</td>
<td>.PRL</td>
<td>18K</td>
<td>fancy directory command</td>
</tr>
<tr>
<td>SET</td>
<td>.PRL</td>
<td>8K</td>
<td>set parameters</td>
</tr>
<tr>
<td>SHOW</td>
<td>.PRL</td>
<td>8K</td>
<td>show status of disks</td>
</tr>
<tr>
<td>SPOOL</td>
<td>.BRS</td>
<td>4K</td>
<td>spool utility</td>
</tr>
<tr>
<td>SPOOL</td>
<td>.PRL</td>
<td>4K</td>
<td></td>
</tr>
<tr>
<td>SPOOL</td>
<td>.RSP</td>
<td>2K</td>
<td></td>
</tr>
<tr>
<td>STAT</td>
<td>.COM</td>
<td>6K</td>
<td>CP/M stat command</td>
</tr>
<tr>
<td>STAT</td>
<td>.PRL</td>
<td>10K</td>
<td>MP/M stat command</td>
</tr>
<tr>
<td>STOPSPLR</td>
<td>.PRL</td>
<td>2K</td>
<td>stop spooler</td>
</tr>
<tr>
<td>SUBMIT</td>
<td>.PRL</td>
<td>6K</td>
<td>MP/M submit</td>
</tr>
<tr>
<td>SYSCOPY</td>
<td>.COM</td>
<td>2K</td>
<td>copy system tracks</td>
</tr>
<tr>
<td>Name</td>
<td>Ext</td>
<td>Size</td>
<td>Comment</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>SYSMPM</td>
<td>.SUB</td>
<td>2K</td>
<td>do a system generation</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>.DAT</td>
<td>2K</td>
<td>default values for system generation</td>
</tr>
<tr>
<td>TMP</td>
<td>.SPR</td>
<td>2K</td>
<td></td>
</tr>
<tr>
<td>TOD</td>
<td>.PRL</td>
<td>4K</td>
<td>time of day</td>
</tr>
<tr>
<td>TSHOW</td>
<td>.COM</td>
<td>2K</td>
<td>show split time</td>
</tr>
<tr>
<td>TSTART</td>
<td>.COM</td>
<td>2K</td>
<td>create timer and start it</td>
</tr>
<tr>
<td>TSTOP</td>
<td>.COM</td>
<td>2K</td>
<td>show final time and stop timer</td>
</tr>
<tr>
<td>TYPE</td>
<td>.PRL</td>
<td>2K</td>
<td>type a file on the screen</td>
</tr>
<tr>
<td>USER</td>
<td>.PRL</td>
<td>2K</td>
<td>set user area</td>
</tr>
<tr>
<td>W</td>
<td>.COM</td>
<td>4K</td>
<td>write a file to SIMH environment</td>
</tr>
<tr>
<td>XDOS</td>
<td>.SPR</td>
<td>10K</td>
<td>XDOS</td>
</tr>
<tr>
<td>XREF</td>
<td>.COM</td>
<td>16K</td>
<td>cross reference utility</td>
</tr>
<tr>
<td>XSUB</td>
<td>.COM</td>
<td>2K</td>
<td>for CP/M DO</td>
</tr>
</tbody>
</table>

### 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 noitraps
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.
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Ext</th>
<th>Size</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKER</td>
<td>.COM</td>
<td>2K</td>
<td>compute the Ackermann function</td>
</tr>
<tr>
<td>ACKER</td>
<td>.SPL</td>
<td>4K</td>
<td>compute the Ackermann function, SPL source</td>
</tr>
<tr>
<td>BOOTGEN</td>
<td>.COM</td>
<td>2K</td>
<td>copy the operating system to the rights sectors and tracks</td>
</tr>
<tr>
<td>BOOTGEN</td>
<td>.SPL</td>
<td>6K</td>
<td>SPL source for BOOTGEN.COM</td>
</tr>
<tr>
<td>C</td>
<td>.SUB</td>
<td>2K</td>
<td>batch file for compiling an SPL source file</td>
</tr>
<tr>
<td>CALC</td>
<td>.PRO</td>
<td>4K</td>
<td>Prolog demo program: Calculator</td>
</tr>
<tr>
<td>DIF</td>
<td>.COM</td>
<td>4K</td>
<td></td>
</tr>
<tr>
<td>DIF</td>
<td>.SPL</td>
<td>10K</td>
<td>SPL source for DIF.COM</td>
</tr>
<tr>
<td>FAC</td>
<td>.COM</td>
<td>2K</td>
<td>compute the factorial</td>
</tr>
<tr>
<td>FAC</td>
<td>.SPL</td>
<td>4K</td>
<td>compute the factorial, SPL source</td>
</tr>
<tr>
<td>FAMILY</td>
<td>.PRO</td>
<td>4K</td>
<td>Prolog demo program: Family relations</td>
</tr>
<tr>
<td>FORMEL</td>
<td>.COM</td>
<td>4K</td>
<td>calculator</td>
</tr>
<tr>
<td>FORMEL</td>
<td>.SPL</td>
<td>6K</td>
<td>calculator, SPL source</td>
</tr>
<tr>
<td>INTEGER</td>
<td>.PRO</td>
<td>2K</td>
<td>Prolog demo program: Integer arithmetic</td>
</tr>
<tr>
<td>KNAKE</td>
<td>.PRO</td>
<td>2K</td>
<td>Prolog demo program: Logic puzzle</td>
</tr>
<tr>
<td>LINKMT</td>
<td>.COM</td>
<td>12K</td>
<td>Pascal MT+ 5.5 linker</td>
</tr>
<tr>
<td>MTERRS</td>
<td>.TXT</td>
<td>6K</td>
<td>Pascal MT+ error messages</td>
</tr>
<tr>
<td>MTPLUS</td>
<td>.000</td>
<td>14K</td>
<td>Pascal MT+ 5.5 compiler file</td>
</tr>
<tr>
<td>MTPLUS</td>
<td>.001</td>
<td>12K</td>
<td>Pascal MT+ 5.5 compiler file</td>
</tr>
<tr>
<td>MTPLUS</td>
<td>.002</td>
<td>8K</td>
<td>Pascal MT+ 5.5 compiler file</td>
</tr>
<tr>
<td>MTPLUS</td>
<td>.003</td>
<td>8K</td>
<td>Pascal MT+ 5.5 compiler file</td>
</tr>
<tr>
<td>MTPLUS</td>
<td>.004</td>
<td>18K</td>
<td>Pascal MT+ 5.5 compiler file</td>
</tr>
<tr>
<td>Name</td>
<td>Ext</td>
<td>Size</td>
<td>Comment</td>
</tr>
<tr>
<td>-------------</td>
<td>-----</td>
<td>------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>MTPLUS</td>
<td>.005</td>
<td>8K</td>
<td>Pascal MT+ 5.5 compiler file</td>
</tr>
<tr>
<td>MTPLUS</td>
<td>.006</td>
<td>6K</td>
<td>Pascal MT+ 5.5 compiler file</td>
</tr>
<tr>
<td>MTPLUS</td>
<td>.COM</td>
<td>36K</td>
<td>Pascal MT+ 5.5 compiler file</td>
</tr>
<tr>
<td>PASCFORM</td>
<td>.COM</td>
<td>36K</td>
<td>Pascal formatter</td>
</tr>
<tr>
<td>PASCFORM</td>
<td>.PAS</td>
<td>54K</td>
<td>Pascal formatter source code</td>
</tr>
<tr>
<td>PASCFORM</td>
<td>.SUB</td>
<td>2K</td>
<td>create Pascal formatter</td>
</tr>
<tr>
<td>PASLIB</td>
<td>.ERL</td>
<td>24K</td>
<td>Pascal MT+ 5.5 run time library</td>
</tr>
<tr>
<td>PINST</td>
<td>.COM</td>
<td>4K</td>
<td>terminal installation program for PROLOGZ</td>
</tr>
<tr>
<td>PINST</td>
<td>.SPL</td>
<td>16K</td>
<td>terminal installation program for PROLOGZ, SPL source</td>
</tr>
<tr>
<td>PRIM</td>
<td>.COM</td>
<td>2K</td>
<td>compute prime numbers</td>
</tr>
<tr>
<td>PRIM</td>
<td>.SPL</td>
<td>2K</td>
<td>compute prime numbers, SPL source</td>
</tr>
<tr>
<td>PROLOGZ</td>
<td>.COM</td>
<td>16K</td>
<td>PROLOGZ interpreter and screen editor</td>
</tr>
<tr>
<td>PROLOGZ</td>
<td>.SPL</td>
<td>54K</td>
<td>SPL source for PROLOGZ</td>
</tr>
<tr>
<td>PROLOGZ</td>
<td>.TXT</td>
<td>40K</td>
<td>PROLOGZ documentation in German</td>
</tr>
<tr>
<td>PROLOGZU</td>
<td>.MAC</td>
<td>2K</td>
<td>helper functions for PROLOGZ in assembler</td>
</tr>
<tr>
<td>QUEEN</td>
<td>.PRO</td>
<td>2K</td>
<td>transfer a file from the file system to the CP/M disk, see also WRITE.COM. Often the name of this program is abbreviated to R.COM.</td>
</tr>
<tr>
<td>READ</td>
<td>.COM</td>
<td>4K</td>
<td>SPL source for READ.COM</td>
</tr>
<tr>
<td>READ</td>
<td>.SPL</td>
<td>10K</td>
<td>SPL source for READ.COM</td>
</tr>
<tr>
<td>RELDUMP</td>
<td>.COM</td>
<td>4K</td>
<td>dump a .REL file to the console</td>
</tr>
<tr>
<td>RELDUMP</td>
<td>.SPL</td>
<td>10K</td>
<td>dump a .REL file to the console, SPL source</td>
</tr>
<tr>
<td>SHOWSEC</td>
<td>.COM</td>
<td>2K</td>
<td>show a disk sector</td>
</tr>
<tr>
<td>SHOWSEC</td>
<td>.SPL</td>
<td>6K</td>
<td>SPL source for SHOWSEC.COM</td>
</tr>
<tr>
<td>SIEVE</td>
<td>.COM</td>
<td>2K</td>
<td>compute prime numbers with a sieve</td>
</tr>
<tr>
<td>SIEVE</td>
<td>.SPL</td>
<td>6K</td>
<td>compute prime numbers with a sieve, SPL source</td>
</tr>
<tr>
<td>SPEED</td>
<td>.COM</td>
<td>2K</td>
<td>utility to measure the clock speed of the simulated CPU</td>
</tr>
<tr>
<td>SPEED</td>
<td>.SPL</td>
<td>4K</td>
<td>SPL source for SPEED.COM</td>
</tr>
<tr>
<td>SPL</td>
<td>.COM</td>
<td>28K</td>
<td>the SPL compiler itself</td>
</tr>
<tr>
<td>SPL</td>
<td>.TXT</td>
<td>50K</td>
<td>SPL language and compiler documentation</td>
</tr>
<tr>
<td>SPLERROR</td>
<td>.DAT</td>
<td>8K</td>
<td>error messages of the compiler</td>
</tr>
<tr>
<td>SPLRTLB</td>
<td>.REL</td>
<td>2K</td>
<td>SPL runtime library</td>
</tr>
<tr>
<td>SYSCOPY</td>
<td>.COM</td>
<td>2K</td>
<td>copy the system tracks between disks</td>
</tr>
<tr>
<td>SYSCOPY</td>
<td>.SPL</td>
<td>6K</td>
<td>SPL source for SYSCOPY.COM</td>
</tr>
<tr>
<td>WC</td>
<td>.COM</td>
<td>6K</td>
<td>word count and query facility</td>
</tr>
<tr>
<td>WC</td>
<td>.SPL</td>
<td>14K</td>
<td>word count and query facility, SPL source</td>
</tr>
<tr>
<td>Name</td>
<td>Ext</td>
<td>Size</td>
<td>Comment</td>
</tr>
<tr>
<td>--------------</td>
<td>-----</td>
<td>------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>WRITE</td>
<td>.COM</td>
<td>2K</td>
<td>write a CP/M file to the file system, see also READ.COM. Often the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>name of this program is abbreviated to W.COM.</td>
</tr>
<tr>
<td>WRITE</td>
<td>.SPL</td>
<td>8K</td>
<td>SPL source for WRITE.COM</td>
</tr>
<tr>
<td>XFORMAT</td>
<td>.COM</td>
<td>2K</td>
<td>format a regular disk or a hard disk</td>
</tr>
<tr>
<td>XFORMAT</td>
<td>.SPL</td>
<td>6K</td>
<td>SPL source for XFORMAT.COM</td>
</tr>
</tbody>
</table>

### 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]
```
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. You can also use “altairz80 bas432” to run this version of Basic. Note that the underscore character (“_”) can be used to cancel the last character entered, i.e. “PRINT 199_8” will print 198.

```
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 4kbas32.bin;load it at 0
sim> g ;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 Basic 4.0 (4k)

An improved 4K Basic is also as part of the archive altsw.zip. You can also use “altairz80 bas440” to run this version of Basic.

```
sim> set cpu 8080 ;note 4k Basic will not run on a Z80 CPU
```
set sio upper ;4k Basic does not like lower case letters as input
set cpu noitrap ;4k Basic likes to execute non 8080 instructions-ignore
set sio ansi ;4k Basic produces 8-bit output, strip to seven bits
d sr 8 ;good setting for the Switch Register
load 4kbas40.bin;load it at 0
g ;and start it

5.11 Altair 8k Basic
Running 8k Basic follows the procedure for 4k Basic. This software is part of the archive altsw.zip.
set cpu 8080 ;note 8k Basic will not run on a Z80 CPU
set sio upper ;8k Basic does not like lower case letters as input
set sio ansi ;8k Basic produces 8-bit output, strip to seven bits
d sr 8 ;good setting for the Switch Register
load 8kbas.bin 0 ;load it at 0
go 0 ;and start it

WRITTEN FOR ROYALTIES BY MICRO-SOFT

MEMORY SIZE? [A]

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
5.12 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

5.13 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.
5.14 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 dispas50.dsk
sim> d sr 8
sim> load dispas50.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

5.15 Altair programming languages VTL-2 and MINOL

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

5.16 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.COMPILE to PASCAL.COMPILE and then rename BASIC.COMPILE to SYSTEM.COMPILE.
– 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.

File: G(et, S(ave, W(hat, N ew, L(dir, R(em, C(hng, T(rans, D(ate, Q uit [B)

DSK0:

<table>
<thead>
<tr>
<th>File</th>
<th>Date</th>
<th>Time</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTEM.MICRO</td>
<td>19-Feb-79</td>
<td>10</td>
<td>512</td>
</tr>
<tr>
<td>SYSTEM.FILER</td>
<td>28-Apr-79</td>
<td>29</td>
<td>512</td>
</tr>
<tr>
<td>SYSTEM.EDITOR</td>
<td>45-Feb-79</td>
<td>57</td>
<td>512</td>
</tr>
<tr>
<td>SYSTEM.LINKER</td>
<td>22-Feb-79</td>
<td>102</td>
<td>512</td>
</tr>
<tr>
<td>SYSTEM.COMPILED</td>
<td>68-Feb-79</td>
<td>124</td>
<td>512</td>
</tr>
<tr>
<td>SYSTEM.SYNTAX</td>
<td>14-May-79</td>
<td>192</td>
<td>512</td>
</tr>
<tr>
<td>SETUP.CODE</td>
<td>25-May-79</td>
<td>206</td>
<td>512</td>
</tr>
<tr>
<td>BINDER.CODE</td>
<td>6-May-79</td>
<td>231</td>
<td>512</td>
</tr>
<tr>
<td>SYSTEM.MISCINFO</td>
<td>1-Feb-79</td>
<td>237</td>
<td>192</td>
</tr>
<tr>
<td>VT100GOTO.TEXT</td>
<td>4-Apr-79</td>
<td>238</td>
<td>512</td>
</tr>
<tr>
<td>VT100GOTO.CODE</td>
<td>2-Apr-79</td>
<td>242</td>
<td>512</td>
</tr>
<tr>
<td>SYSTEM.PASCAL</td>
<td>33-Apr-79</td>
<td>244</td>
<td>512</td>
</tr>
<tr>
<td>SYSTEM.LIBRARY</td>
<td>17-Apr-79</td>
<td>277</td>
<td>512</td>
</tr>
<tr>
<td>BASIC.COMPILER</td>
<td>30-Apr-79</td>
<td>294</td>
<td>512</td>
</tr>
<tr>
<td>LOOP.TEXT</td>
<td>4-Apr-79</td>
<td>324</td>
<td>512</td>
</tr>
<tr>
<td>LOOP.CODE</td>
<td>4-Apr-79</td>
<td>328</td>
<td>512</td>
</tr>
<tr>
<td>Z80.ERRORS</td>
<td>8-Mar-79</td>
<td>332</td>
<td>70</td>
</tr>
<tr>
<td>Z80.OPCODES</td>
<td>3-Dec-78</td>
<td>340</td>
<td>68</td>
</tr>
<tr>
<td>SYSTEM.ASSEMBLER</td>
<td>53-Apr-79</td>
<td>343</td>
<td>512</td>
</tr>
</tbody>
</table>

< 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 resp. R 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 resp. W 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

```python
#!/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]
sysexit(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)

c Change

before

<table>
<thead>
<tr>
<th>Offset</th>
<th>Current Track</th>
<th>Current Sector</th>
<th>Current Block</th>
<th>Current Drive</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000</td>
<td>09 29 29 EB</td>
<td>01 36 00 2A</td>
<td>94 02 19 09</td>
<td>A1 E1 22 90</td>
</tr>
<tr>
<td>000000</td>
<td>02 E1 22 92</td>
<td>02 D1 EB 22</td>
<td>94 02 EB 2A</td>
<td>90 02 06 08</td>
</tr>
<tr>
<td>000000</td>
<td>1A BE C2 BA</td>
<td>1A 23 13 10</td>
<td>F7 21 00 00</td>
<td>E5 2A 94 02</td>
</tr>
<tr>
<td>000000</td>
<td>EB 2A 92 02</td>
<td>73 23 72 C3</td>
<td>A4 03 D2 D3</td>
<td>1A 2A 94 02</td>
</tr>
<tr>
<td>000000</td>
<td>11 08 00 19</td>
<td>5E 23 56 7B</td>
<td>3D B2 C2 96</td>
<td>1A 21 01 00</td>
</tr>
<tr>
<td>000000</td>
<td>11 08 00 19</td>
<td>5E 23 56 7B</td>
<td>3D B2 C2 96</td>
<td>1A 21 01 00</td>
</tr>
<tr>
<td>000000</td>
<td>11 08 00 19</td>
<td>5E 23 56 7B</td>
<td>3D B2 C2 96</td>
<td>1A 21 01 00</td>
</tr>
<tr>
<td>000000</td>
<td>11 08 00 19</td>
<td>5E 23 56 7B</td>
<td>3D B2 C2 96</td>
<td>1A 21 01 00</td>
</tr>
<tr>
<td>000000</td>
<td>11 08 00 19</td>
<td>5E 23 56 7B</td>
<td>3D B2 C2 96</td>
<td>1A 21 01 00</td>
</tr>
</tbody>
</table>

After

<table>
<thead>
<tr>
<th>Offset</th>
<th>Current Track</th>
<th>Current Sector</th>
<th>Current Block</th>
<th>Current Drive</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000</td>
<td>02 E1 22 92</td>
<td>02 D1 EB 22</td>
<td>94 02 EB 2A</td>
<td>90 02 06 08</td>
</tr>
<tr>
<td>000000</td>
<td>1A BE C2 BA</td>
<td>1A 23 13 10</td>
<td>F7 21 00 00</td>
<td>E5 2A 94 02</td>
</tr>
<tr>
<td>000000</td>
<td>EB 2A 92 02</td>
<td>73 23 72 C3</td>
<td>A4 03 D2 D3</td>
<td>1A 2A 94 02</td>
</tr>
<tr>
<td>000000</td>
<td>11 08 00 19</td>
<td>5E 23 56 7B</td>
<td>3D B2 C2 96</td>
<td>1A 21 01 00</td>
</tr>
<tr>
<td>000000</td>
<td>11 08 00 19</td>
<td>5E 23 56 7B</td>
<td>3D B2 C2 96</td>
<td>1A 21 01 00</td>
</tr>
<tr>
<td>000000</td>
<td>11 08 00 19</td>
<td>5E 23 56 7B</td>
<td>3D B2 C2 96</td>
<td>1A 21 01 00</td>
</tr>
<tr>
<td>000000</td>
<td>11 08 00 19</td>
<td>5E 23 56 7B</td>
<td>3D B2 C2 96</td>
<td>1A 21 01 00</td>
</tr>
</tbody>
</table>

---

SIMH AltairZ80 35 of 59
after

<table>
<thead>
<tr>
<th>Current Track</th>
<th>Current Sector</th>
<th>Current Block</th>
<th>Current Drive</th>
</tr>
</thead>
<tbody>
<tr>
<td>0005</td>
<td>000C</td>
<td>000B</td>
<td>A:</td>
</tr>
</tbody>
</table>

Offset 0 1 2 3 4 5 6 7 8 9 A B C D E F
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,.C i.....|
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]

Execute what file? setup

INITIALIZE.........................

SETUP: C(HANGE T(EACH H(ELP Q(UIT [D1]

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,!)  

NEW VALUE: 26

FIELD NAME = ERASE LINE

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
32 26 1A SUB ^Z

N WANT TO CHANGE THIS VALUE? (Y,N,!)  
FIELD NAME = ERASE SCREEN

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL
33 27 1B ESC ^[  

N WANT TO CHANGE THIS VALUE? (Y,N,!)  
FIELD NAME = ERASE TO END OF LINE

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 ^@  

N WANT TO CHANGE THIS VALUE? (Y,N,!)
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Current Value</th>
<th>Change Value?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Erase to end of screen</td>
<td>N</td>
<td>Y / N</td>
</tr>
<tr>
<td>8510A</td>
<td>false</td>
<td>Y / N</td>
</tr>
<tr>
<td>Byte flipped machine</td>
<td>false</td>
<td>Y / N</td>
</tr>
<tr>
<td>Clock</td>
<td>false</td>
<td>Y / N</td>
</tr>
<tr>
<td>Lower case</td>
<td>false</td>
<td>Y / N</td>
</tr>
<tr>
<td>Random cursor addressing</td>
<td>false</td>
<td>Y / N</td>
</tr>
<tr>
<td>Slow terminal</td>
<td>false</td>
<td>Y / N</td>
</tr>
<tr>
<td>Word oriented machine</td>
<td>false</td>
<td>Y / N</td>
</tr>
<tr>
<td>Key for break</td>
<td>N</td>
<td>Y / N</td>
</tr>
<tr>
<td>Key for flush</td>
<td>N</td>
<td>Y / N</td>
</tr>
<tr>
<td>Key for stop</td>
<td>N</td>
<td>Y / N</td>
</tr>
</tbody>
</table>
**FIELD NAME** = **KEY TO DELETE CHARACTER**  
OCTAL DECIMAL HEXADECIMAL ASCII CONTROL  
10 8 8 BS "^H"

**FIELD NAME** = **KEY TO DELETE LINE**  
OCTAL DECIMAL HEXADECIMAL ASCII  
177 127 7F DEL

**FIELD NAME** = **KEY TO END FILE**  
OCTAL DECIMAL HEXADECIMAL ASCII CONTROL  
3 3 3 ETX "^C"

**FIELD NAME** = **KEY TO MOVE CURSOR DOWN**  
OCTAL DECIMAL HEXADECIMAL ASCII CONTROL  
12 10 A LF "^J"

**FIELD NAME** = **KEY TO MOVE CURSOR LEFT**  
OCTAL DECIMAL HEXADECIMAL ASCII CONTROL  
10 8 8 BS "^H"

**FIELD NAME** = **KEY TO MOVE CURSOR RIGHT**  
OCTAL DECIMAL HEXADECIMAL ASCII CONTROL  
34 28 1C FS "\"

**FIELD NAME** = **KEY TO MOVE CURSOR UP**  
OCTAL DECIMAL HEXADECIMAL ASCII CONTROL  
37 31 1F US "_"

**FIELD NAME** = **LEAD IN FROM KEYBOARD**  
OCTAL DECIMAL HEXADECIMAL ASCII CONTROL  
13 11 B VT "^K"

**FIELD NAME** = **LEAD IN TO SCREEN**  
OCTAL DECIMAL HEXADECIMAL ASCII CONTROL  
0 0 0 NUL "@"
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
Step 4: Rename NEW.MISCINFO to SYSTEM.MISCINFO

Command: Edit, Run, File, Compile, Link, Execute, Assemble, Debug, ? [II.0]

Filer: Get, Save, What, New, List, Rename, Change, Trans, Date, Quit [B]

Dir listing of what vol ? 

Filer: Get, Save, What, New, List, Rename, Change, Trans, Date, Quit [B]

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: Get, Save, What, New, List, Rename, Change, Trans, Date, Quit [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.

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 an 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://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-GRA

Documentation
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

```plaintext
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

```plaintext
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:

<table>
<thead>
<tr>
<th>SYNC</th>
<th>TRACK</th>
<th>SECTOR</th>
<th>UNUSED</th>
<th>USER DATA</th>
<th>CHKSUM</th>
<th>ECC</th>
<th>ECC_VALID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>256</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

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 AltairZ80 HDSK device.
sim> set fif ena - enable 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 MDS-AD 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.
13 Compupro 8-16 Simulation

Compupro Controller support was added by Howard M. Harte, hharte@hartetec.com. The 8086 simulation was added by Peter Schorn.

13.1 Overview

The Compupro 8-16 was a fairly advanced IEEE-696 bus based system that included a dual CPU card containing Intel 8085 and 8088 processors. This processor card was capable of switching between CPUs at runtime, and this allowed the user to run CP/M-80 as well as CP/M-86. In the latest version of CP/M-80 released by Viasyn (who had acquired Compupro by that time) uses the 8085 CPU for running CP/M, but offloads some of the memory operations to the 8088 CPU because of its ability to operate faster, and more easily address memory above 64K.

Additional devices include:

- **DISK1A** – Compupro DISK1A High Performance Floppy Disk Controller.
- **DISK2** – Compupro DISK2 Hard Disk Controller.
- **DISK3** – Viasyn DISK3 Hard Disk Controller for ST-506 Drives.
- **SELCHAN** – Compupro Selector Channel DMA Controller.
- **MDRIVEH** – Compupro MDRIVE/H RAM Disk (up to 4MB.)
- **SS1** – Compupro System Support 1 (experimental and incomplete simulation.)

There are configuration files that configure SIMH to simulate a Compupro 8-16, with various attached controllers to run CP/M-80 and CP/M-86. These configuration files are:

- **ccpm22** Compupro 8-16 CP/M-80 2.2
- **ccpm86** Compupro 8-16 CP/M-86
- **ccpm22q** Compupro 8-16 CP/M-80 2.2Q (latest Viasyn version)

13.2 DISK1A High Performance Floppy Disk Controller

The DISK1A High Performance Floppy Disk Controller is based on the Intel i8272 Floppy Disk Controller chip, which is the same as an NEC765 floppy controller. The implementation of the DISK1A uses a generic i8272 controller core with a DISK1A-specific wrapper to implement the Compupro DISK1A-specific features.

The i8272 controller core simulation utilizes the ImageDisk (IMD) file format for accessing simulated floppy disks.

The DISK1A simulation also includes the Compupro bootstrap ROM, which contains bootloaders for 8085, 8088, and 68000 CPUs.

Tony Nicholson provided several enhancements to the DISK1A controller simulation, including variable sector size support, as well as extended addressing support for DMA data transfer.

13.2.1 DISK1A Controller Parameters

The DISK1A controller supports several parameters which can be configured by the simulator:

- **ROM** – Enable bootstrap ROM at 0000-01FFh.
NOROM – Disable bootstrap ROM.

DEBUG – enable debug tracing, useful for debugging software. One or more debug levels may be selected at any given time. Several debug tracing levels are provided:

- **ERROR** – Error messages, these are bugs in the simulation or in the way a program running on the simulator accesses the controller. This message level is on by default.
- **SEEK** – Seek messages, related to head positioning.
- **CMD** – Disk controller commands.
- **RDDATA** – Read Data messaging
- **WRDATA** – Write Data messaging
- **STATUS** – Status register reading
- **VERBOSE** – Extra verbosity for debugging.

NODEBUG – turn off one or more debug message levels.

The DISK1A supports four drives, labeled DISK1A0 through DISK1A3. If a drive is attached to a non-existent image file, the image file will be created, and the user will be asked for a comment description of the disk. SIMH adds its own IMD header to the comment field, along with information about the version of the controller core (in this case i8272) as well as the SIM.IMD module, to help facilitate debugging. The SIM.IMD module will automatically format the new disk image in IBM 3740 Single-Sided, Single-Density format. If the user wishes to use the disk in another format, then it should be reformatted using the CompuPro format program on either CP/M-80 or CP/M-86.

### 13.2.2 DISK1A Controller Limitations

The DISK1A controller and the underlying i8272 controller core only support DMA-mode operation at present. There is no support for polled-mode I/O access for reading/writing data. While the DISK1A simulation is believed to be very accurate in normal operation, the error handling and bad data reporting from the SIM.IMD module are not well implemented/tested. For example, if a disk image contains a CRC error on one of the sectors, this may not be propagated up to the DISK1A status registers. This should not be an issue for disks created with SIMH, because there is no such thing as a CRC error in this case. But, for images that were read from a real floppy using IMD, a sector containing bad data might be reported as good by the simulation.

### 13.3 DISK2 Compupro Hard Disk Controller

The DISK2 Hard Disk Controller provides 20MB of fixed-disk storage, and supports four hard disk drives. Just like a real DISK2 controller, it needs to be used in conjunction with the Compupro Selector Channel DMA controller.

#### 13.3.1 DISK2 Controller Parameters

The DISK2 controller supports several parameters which can be configured by the simulator:

- **DEBUG** – enable debug tracing, useful for debugging software. One or more debug levels may be selected at any given time. Several debug tracing levels are provided:
  - **ERROR** – Error messages, these are bugs in the simulation or in the way a program running on the simulator accesses the controller. This message level is on by default.
  - **SEEK** – Seek messages, related to head positioning.
  - **CMD** – Disk controller commands.
  - **RDDATA** – Read Data messaging
• WRDATA – Write Data messaging
• STATUS – Status register reading
• VERBOSE – Extra verbosity for debugging.

NODEBUG – turn off one or more debug message levels.

13.3.2 DISK2 Controller Configuration Registers
The DISK2 controller has several configuration registers that can be examined and deposited by the simulator. The defaults are configured for a standard 20MB hard disk. These registers are:

NTRACKS – Number of tracks on the simulated hard disks.
NHEADS – Number of heads on the simulated hard disks.
NSECTORS – Number of sectors on the simulated hard disks.
SECTSIZE – Sector size on the simulated hard disks.

The DISK2 supports four drives, labeled DISK20 through DISK23. If a drive is attached to a non-existent image file, the image file will be created. A newly created disk image should be formatted with the CompuPro DISK2.COM command.

13.4 SELCHAN Compupro Selector Channel Controller
The Compupro Selector Channel DMA controller provides DMA support for the Compupro DISK2 Hard Disk Controller.

13.4.1 DISK2 Controller Parameters
The DISK2 controller supports several parameters which can be configured by the simulator:
DEBUG – enable debug tracing, useful for debugging software. One or more debug levels may be selected at any given time. Several debug tracing levels are provided:
• ERROR – Error messages, these are bugs in the simulation or in the way a program running on the simulator accesses the controller. This message level is on by default.
• DMA – DMA Transfer Messages.
• VERBOSE – Extra verbosity for debugging.

NODEBUG – turn off one or more debug message levels.

Tony Nicholson added extended addressing DMA support to the SELCHAN module.

13.5 DISK3 Viasyn ST-506 Hard Disk Controller
The DISK3 Hard Disk Controller is an advanced DMA-based hard disk controller that uses linked-list descriptors to send commands and transfer data between the host CPU and the disk controller.

13.5.1 DISK3 Controller Parameters
The DISK3 controller supports several parameters which can be configured by the simulator:
DEBUG – enable debug tracing, useful for debugging software. One or more debug levels may be selected at any given time. Several debug tracing levels are provided:
• ERROR – Error messages, these are bugs in the simulation or in the way a program running on the simulator accesses the controller. This message level is on by default.
• SEEK – Seek messages, related to head positioning.
• CMD – Disk controller commands.
• RDDATA – Read Data messaging
• WRDATA – Write Data messaging
• STATUS – Status register reading
• VERBOSE – Extra verbosity for debugging.

NODEBUG – turn off one or more debug message levels.

13.5.2 DISK3 Controller Configuration Registers

The DISK3 controller has several configuration registers that can be examined and deposited by the simulator. The defaults are configured for a standard 20MB hard disk. These registers are:

NTRACKS – Number of tracks on the simulated hard disks.
NHEADS – Number of heads on the simulated hard disks.
NSECTORS – Number of sectors on the simulated hard disks.
SECTSIZE – Sector size on the simulated hard disks.

The DISK3 supports four drives, labeled DISK30 through DISK33. If a drive is attached to a non-existent image file, the image file will be created. A newly created disk image should be formatted with the CompuPro DISK3.COM command.

13.5.3 DISK3 Controller Limitations

The DISK3 controller has been tested with the CompuPro DISK3.COM diagnostic utility, but has not been tested with CP/M.
14 Cromemco 4/16/64FDC and CCS-2422 FDC Simulation

Cromemco 4/16/64FDC (CROMFDC) Floppy Controller support was added by Howard M. Harte, hharte@hartetec.com.

14.1 Overview

The Cromemco 4/16/64FDC disk controllers are a family of floppy disk controllers for Cromemco systems. The controller is based on the Western Digital WD179x series of floppy controller chips. The implementation of the DISK1A uses a generic WD179x controller core with a Cromemco-specific wrapper to implement the Cromemco-specific features.

The WD179x controller core simulation utilizes the ImageDisk (IMD) file format for accessing simulated floppy disks.

The Cromemco simulation also includes the Cromemco RDOS 2.52 and RDOS 3.12 boot ROMs. Additional devices include:

CROMFDC – Cromemco 4/16/64FDC Floppy Disk Controller.

14.1.1 CROMFDC Controller Parameters

The CROMFDC controller supports several parameters which can be configured by the simulator:

ROM – Enable RDOS ROM at C000-DFFFh.

NOROM – Disable RDOS ROM.

DEBUG – enable debug tracing, useful for debugging software. One or more debug levels may be selected at any given time. Several debug tracing levels are provided:

- ERROR – Error messages, these are bugs in the simulation or in the way a program running on the simulator accesses the controller. This message level is on by default.
- SEEK – Seek messages, related to head positioning.
- CMD – Disk controller commands.
- RDDATA – Read Data messaging
- WRDATA – Write Data messaging
- STATUS – Status register reading
- VERBOSE – Extra verbosity for debugging.
- DRIVE – Messaging specific to drive, i.e.: Motor on/off, side selection.
- IRQ – Interrupt debugging

NODEBUG – turn off one or more debug message levels.

14.1.2 CROMFDC Controller Configuration Registers

The CROMFDC controller has several configuration registers that can be examined and deposited by the simulator. The defaults are configured for a standard 20MB hard disk. These registers are:

DIPSW – 5-position DIP switch on 64FDC card.
BOOTSTRAP – 0 for RDOS 2.52, 1 for RDOS 3.12.

FDCTYPE – CROMFDC Type: Set to 4, 16, 64 for Cromemco FDC, or 50 for CCS-2422 FDC.

BOOT – BOOT jumper setting, default is 1 (auto-boot.)

INHINIT – Inhibit Init (Format) switch, default is 0 (not inhibited.)

The CROMFDC supports four drives, labeled CROMFDC0 through CROMFDC3. If a drive is attached to a non-existent image file, the image file will be created, and the user will be asked for a comment description of the disk. SIMH adds its own IMD header to the comment field, along with information about the version of the controller core (in this case WD179x) as well as the SIM_IMD module, to help facilitate debugging. The SIM_IMD module will automatically format the new disk image in IBM 3740 Single-Sided, Single-Density format. If the user wishes to use the disk in another format, then it should be reformatted using the Cromemco INIT.COM program under Cromemco DOS (CDOS), or using the INITLARG.COM program under 86-DOS.

14.1.3 CROMFDC Controller Limitations

The CROMFDC controller and the underlying WD179x controller core only support polled I/O mode operation at present. There is no support for DMA access for reading/writing data. The CROMFDC interrupt support is not fully implemented/tested; however, none of the operating systems that use the CROMFDC controller (CDOS, CP/M 2.2, 86-DOS) seem to use this mode. Z80 Cromix does not boot in simulation, but probably not due to the lack of disk controller interrupts.
15 Advanced Digital Corporation Super-Six Simulation

ADC Super-Six Single-Board Computer support was added by Howard M. Harte, hharte@hartetec.com.

15.1 Overview

The Advanced Digital Corporation Super-Six SBC is a Z80-based single board computer on an S-100 bus card. It has 128K of RAM, and a WD179x-based floppy disk controller.

Additional devices include:

ADCS6 – ADC Super-Six SBC.

15.1.1 ADCS6 SBC Parameters

The ADCS6 SBC supports several parameters which can be configured by the simulator:

ROM – Enable RDOS ROM at C000-DFFFh.

NOROM – Disable RDOS ROM.

DEBUG – enable debug tracing, useful for debugging software. One or more debug levels may be selected at any given time. Several debug tracing levels are provided:

- ERROR – Error messages, these are bugs in the simulation or in the way a program running on the simulator accesses the controller. This message level is on by default.
- SEEK – Seek messages, related to head positioning.
- CMD – Disk controller commands.
- RDDATA – Read Data messaging
- WRDATA – Write Data messaging
- STATUS – Status register reading
- VERBOSE – Extra verbosity for debugging.
- DRIVE – Messaging specific to drive, i.e.: Motor on/off, side selection.
- IRQ – Interrupt debugging
- DMA – Z80-DMA register debugging

NODEBUG – turn off one or more debug message levels.

15.1.2 ADCS6 SBC Configuration Registers

The CROMFDC controller has several configuration registers that can be examined and deposited by the simulator. The defaults are configured for a standard 20MB hard disk. These registers are:

J7 – Jumper block J7 setting, see ADC Manual for details.

The ADCS6 supports four drives, labeled ADCS60 through ADCS63. If a drive is attached to a non-existent image file, the image file will be created, and the user will be asked for a comment description of the disk.

SIMH adds its own IMD header to the comment field, along with information about the version of the controller core (in this case WD179x) as well as the SIM_IMD module, to help facilitate debugging. The SIM_IMD module will automatically format the new disk image in IBM 3740 Single-Sided, Single-Density
format. If the user wishes to use the disk in another format, then it should be reformatted using the FMT8.COM program under CP/M 2.2.

15.1.3 ADCS6 SBC Limitations

The ADCS6 simulation does not support the Z80-DMA, CTC, parallel ports, or interrupt-driven operation. The ADCS6 bank switching and ROM are not implemented fully.
16 N8VEM Single Board Computer Simulation

N8VEM Single Board Computer support was added by Howard M. Harte, hharte@hartetec.com.

16.1 Overview

The N8VEM Single Board Computer is a homebrew Z80 system designed by Andrew Lynch. This SBC can have 1MB of EPROM, 512KB of RAM, and can run CP/M 2.2. More details about the N8VEM are on the following newsgroup:

http://groups.google.com/group/n8vem

Additional devices include:

N8VEM – N8VEM Single-Board Computer.

16.1.1 N8VEM SBC Parameters

The N8VEM SBC supports several parameters which can be configured by the simulator:

DEBUG – enable debug tracing, useful for debugging software. One or more debug levels may be selected at any given time. Several debug tracing levels are provided:

- ERROR – Error messages. This message level is on by default.
- PIO – 8255 Parallel I/O port messages.
- UART – Serial port messages.
- RTC – Real-Time Clock messages.
- ROM – ROM messages.
- VERBOSE – Extra verbosity for debugging.

NODEBUG – turn off one or more debug message levels.

16.1.2 N8VEM SBC Configuration Registers

The N8VEM SBC has several configuration registers that can be examined and deposited by the simulator. These registers are:

SAVEROM – When set to 1, the ROM data will be saved to an attached file.

SAVERAM – When set to 1, the RAM data will be saved to an attached file.

PIO1A – 8255 PIO1A port.

PIO1B – 8255 PIO1A port

PIO1C – 8255 PIO1A port

PIOCTRL – 8255 PIO Control register

The N8VEM supports two storage devices: N8VEM0 and N8VEM1.

N8VEM0 saves data from the ROM to a raw binary file when SAVEROM is set to 1, and the unit is detached. Since the N8VEM has 1MB of ROM, the file created will be 1048576 bytes in length. This file can then be burned into an EPROM for use with the actual N8VEM hardware.
N8VEM1 saves data from the RAM to a raw binary file when SAVERAM is set to 1, and the unit is detached. Since the N8VEM has 512KB of RAM, the file created will be 524288 bytes in length. This file is useful as a "core dump" to examine the state of a running system. It can also serve to model persistent storage for the N8VEM RAM drive, in case an 512KB NVRAM is used in place of the 512KB SRAM on the N8VEM.

16.1.3 N8VEM SBC Limitations

The DS1302 Real-Time Clock on the N8VEM is not supported by the simulation. The 8255 PIO ports serve no real purpose, other than that the values written to them can be read by the simulator.
17.1 Overview

The ImageDisk (IMD) file format is a portable format which includes all metadata required to accurately describe a soft-sectored floppy disk. The IMD file format was developed by Dave Dunfield, along with a set of ImageDisk utilities for creating ImageDisk disks from raw binary files and from real floppy disks. In addition, IMD disk images can be written back to real floppies for use on actual hardware. SIMH support for ImageDisk is provided by the SIM_IMD module written by Howard M. Harte. The SIM_IMD module provides functions for creating, opening, reading, writing, formatting, and closing IMD files. The i8272 and WD179x floppy controller core simulations leverage the SIM_IMD module for low-level disk access.

17.2 References

More information and support for ImageDisk, including a detailed description of the IMD file format, and utilities for creating and manipulating IMD files can be found on Dave Dunfield’s website: