**AltairZ80 Simulator Usage**

**30-Jul-2023**

**COPYRIGHT NOTICE**

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

Original code for the AltairZ80 part published in 2002-2014, written by Peter Schorn

Copyright (c) 2002-2023, 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, Ernie Price, Mike Douglas and Patrick Linstruth.

[1 Simulator Files 5](#_Toc140521615)

[2 Revision History 7](#_Toc140521616)

[3 Background 9](#_Toc140521617)

[4 Hardware 10](#_Toc140521618)

[4.1 CPU 10](#_Toc140521619)

[4.1.1 Registers for the 8080 and Z80 12](#_Toc140521620)

[4.1.2 Registers for the 8086 13](#_Toc140521621)

[4.1.3 Registers for the MC68000 14](#_Toc140521622)

[4.2 The Serial I/O Card (2SIO) 16](#_Toc140521623)

[4.3 MITS 88-2SIO Serial Adapter (M2SIO) 18](#_Toc140521624)

[4.3.1 Using the M2SIO device with serial ports 19](#_Toc140521625)

[4.3.2 Using the M2SIO device with sockets 19](#_Toc140521626)

[4.3.3 M2SIO limitations 19](#_Toc140521627)

[4.4 The SIMH pseudo device 20](#_Toc140521628)

[4.5 The 88-DISK controller 21](#_Toc140521629)

[4.6 The 88-HDSK controller 22](#_Toc140521630)

[4.7 The simulated hard disk 22](#_Toc140521631)

[4.8 The simulated network 24](#_Toc140521632)

[5 Sample Software 24](#_Toc140521633)

[5.1 CP/M Version 2.2 24](#_Toc140521634)

[5.2 CP/M Version 3 with banked memory 27](#_Toc140521635)

[5.3 MP/M II with banked memory 30](#_Toc140521636)

[5.4 CP/NET 33](#_Toc140521637)

[5.5 CPNOS 34](#_Toc140521638)

[5.6 CP/M application software 34](#_Toc140521639)

[5.7 MITS Disk Extended BASIC Version 4.1 36](#_Toc140521640)

[5.8 Altair DOS Version 1.0 37](#_Toc140521641)

[5.9 Altair Basic 3.2 (4k) 38](#_Toc140521642)

[5.10 Altair Basic 4.0 (4k) 38](#_Toc140521643)

[5.11 Altair 8k Basic 39](#_Toc140521644)

[5.12 Altair Basic 4.0 39](#_Toc140521645)

[5.13 Altair Disk Extended Basic Version 300-5-C 40](#_Toc140521646)

[5.14 Altair Disk Extended Basic Version 5.0 40](#_Toc140521647)

[5.15 Altair Hard Disk Basic 300-5-C 41](#_Toc140521648)

[5.16 Altair programming languages VTL-2 and MINOL 42](#_Toc140521649)

[5.17 UCSD Pascal II.0 42](#_Toc140521650)

[5.18 CP/M-68K 42](#_Toc140521651)

[6 Special simulator features 43](#_Toc140521652)

[6.1 Memory access breakpoints (8080/Z80 only) 43](#_Toc140521653)

[6.2 Instruction breakpoints (8080/Z80/8086) 43](#_Toc140521654)

[6.3 Breakpoints and instruction history (8080/Z80 only) 43](#_Toc140521655)

[7 Brief summary of all major changes to the original Altair simulator 43](#_Toc140521656)

[8 Appendix: Python script for converting MBL files to plain binary files 44](#_Toc140521657)

[9 Appendix: How to bring up UCSD Pascal II.0 on SIMH 46](#_Toc140521658)

[10 Vector Graphic, Inc. Simulation 54](#_Toc140521659)

[10.1 Overview 54](#_Toc140521660)

[10.2 48K Vector MZ 55](#_Toc140521661)

[10.3 56K Vector MZ 56](#_Toc140521662)

[10.4 56K Vector with HD-FD Controller 56](#_Toc140521663)

[10.5 Notes on Simulated Hardware 57](#_Toc140521664)

[10.6 Notes on the Vector Graphic Disk Image (VGI) File Format 57](#_Toc140521665)

[11 IMSAI 8080 Simulation 58](#_Toc140521666)

[11.1 Overview 58](#_Toc140521667)

[11.2 IMSAI 8080 with FIF Disk Controller 58](#_Toc140521668)

[12 NorthStar MDS-A and MDS-AD FDC Simulation 59](#_Toc140521669)

[12.1 Overview 59](#_Toc140521670)

[12.2 MDS-A Single Density Disk Controller 59](#_Toc140521671)

[12.3 MDS-AD Double Density Disk Controller 59](#_Toc140521672)

[13 Compupro 8-16 Simulation 61](#_Toc140521673)

[13.1 Overview 61](#_Toc140521674)

[13.2 DISK1A High Performance Floppy Disk Controller 61](#_Toc140521675)

[13.2.1 DISK1A Controller Parameters 61](#_Toc140521676)

[13.2.2 DISK1A Controller Limitations 62](#_Toc140521677)

[13.3 DISK2 Compupro Hard Disk Controller 62](#_Toc140521678)

[13.3.1 DISK2 Controller Parameters 62](#_Toc140521679)

[13.3.2 DISK2 Controller Configuration Registers 63](#_Toc140521680)

[13.4 SELCHAN Compupro Selector Channel Controller 63](#_Toc140521681)

[13.4.1 DISK2 Controller Parameters 63](#_Toc140521682)

[13.5 DISK3 Viasyn ST-506 Hard Disk Controller 63](#_Toc140521683)

[13.5.1 DISK3 Controller Parameters 63](#_Toc140521684)

[13.5.2 DISK3 Controller Configuration Registers 64](#_Toc140521685)

[13.5.3 DISK3 Controller Limitations 64](#_Toc140521686)

[14 Compupro CPU-68K Simulation 65](#_Toc140521687)

[14.1 Overview 65](#_Toc140521688)

[14.2 M68K Registers Added 65](#_Toc140521689)

[14.3 M68K Additional Parameters 65](#_Toc140521690)

[15 Cromemco 4/16/64FDC and CCS-2422 FDC Simulation 66](#_Toc140521691)

[15.1 Overview 66](#_Toc140521692)

[15.1.1 CROMFDC Controller Parameters 66](#_Toc140521693)

[15.1.2 CROMFDC Controller Configuration Registers 66](#_Toc140521694)

[15.1.3 CROMFDC Controller Limitations 67](#_Toc140521695)

[16 Seattle Computer Products Simulation 68](#_Toc140521696)

[16.1 Overview 68](#_Toc140521697)

[16.2 SCP300F CPU Support Board 68](#_Toc140521698)

[16.2.1 SCP300F Parameters 68](#_Toc140521699)

[16.2.2 SCP300F Configuration Registers 68](#_Toc140521700)

[16.2.3 SCP300F Limitations 69](#_Toc140521701)

[16.3 TDD – Tarbell Double-Density Controller 69](#_Toc140521702)

[16.3.1 TDD Parameters 69](#_Toc140521703)

[16.3.2 TDD Configuration Registers 69](#_Toc140521704)

[16.3.3 TDD Limitations 69](#_Toc140521705)

[16.4 DJHDC – Morrow HDC / DMA Hard Disk Controller 70](#_Toc140521706)

[16.4.1 DJHDC Controller Parameters 70](#_Toc140521707)

[16.4.2 DJHDC Controller Configuration Registers 70](#_Toc140521708)

[16.4.3 DJHDC Controller Limitations 70](#_Toc140521709)

[17 Tarbell MDL-1011/2022 Floppy Disk Interface Simulation 72](#_Toc140521710)

[17.1 Overview 72](#_Toc140521711)

[17.1.1 TARBELL Controller Parameters 72](#_Toc140521712)

[17.1.2 TARBELL Example Usage 73](#_Toc140521713)

[17.1.3 TARBELL Controller Limitations 73](#_Toc140521714)

[18 JADE Double D Floppy Disk Controller Simulation 74](#_Toc140521715)

[18.1 Overview 74](#_Toc140521716)

[18.1.1 JADE Double D Controller Parameters 74](#_Toc140521717)

[18.1.2 JADEDD Example Usage 74](#_Toc140521718)

[18.1.3 JADE Double D Controller Limitations 75](#_Toc140521719)

[19 iCOM FD3712/FD3812 Flexible Disk System Simulation 76](#_Toc140521720)

[19.1 Overview 76](#_Toc140521721)

[19.1.1 iCOM FD3712/FD3812 Flexible Disk System Parameters 76](#_Toc140521722)

[19.1.2 iCOM Example Usage 77](#_Toc140521723)

[20 Morrow DISK JOCKEY 2D Disk Controller Simulation 78](#_Toc140521724)

[20.1 Overview 78](#_Toc140521725)

[20.1.1 DJ2D Parameters 78](#_Toc140521726)

[20.1.2 DJ2D Example Usage 79](#_Toc140521727)

[20.1.3 DJ2D Simulator Limitations 79](#_Toc140521728)

[21 Advanced Digital Corporation Simulation 80](#_Toc140521729)

[21.1 ADC Super-Six Single-Board Computer 80](#_Toc140521730)

[21.1.1 ADCS6 SBC Overview 80](#_Toc140521731)

[21.1.2 ADCS6 SBC Parameters 80](#_Toc140521732)

[21.1.3 ADCS6 SBC Configuration Registers 80](#_Toc140521733)

[21.1.4 ADCS6 SBC Limitations 81](#_Toc140521734)

[21.2 ADCHD – Advanced Digital HDC-1001 Hard Disk Controller 81](#_Toc140521735)

[21.2.1 ADCHD Controller Parameters 81](#_Toc140521736)

[21.2.2 ADCHD Controller Configuration Registers 82](#_Toc140521737)

[21.2.3 ADCHD Controller Limitations 82](#_Toc140521738)

[22 N8VEM Single Board Computer Simulation 83](#_Toc140521739)

[22.1 Overview 83](#_Toc140521740)

[22.1.1 N8VEM SBC Parameters 83](#_Toc140521741)

[22.1.2 N8VEM SBC Configuration Registers 83](#_Toc140521742)

[22.1.3 N8VEM SBC Limitations 84](#_Toc140521743)

[23 Morrow Micro Decision (MD) Computer Simulation 85](#_Toc140521744)

[23.1 Overview 85](#_Toc140521745)

[23.1.1 Morrow MD Parameters 85](#_Toc140521746)

[23.1.2 Morrow MD Configuration Registers 85](#_Toc140521747)

[23.1.3 Morrow MD Limitations 86](#_Toc140521748)

[23.1.4 Morrow MD Example Usage 86](#_Toc140521749)

[24 Processor Technology Sol-20 Terminal Computer Simulation 88](#_Toc140521750)

[24.1 Overview 88](#_Toc140521751)

[24.1.1 Sol-20 Parameters 88](#_Toc140521752)

[24.1.2 Sol-20 Configuration Registers 88](#_Toc140521753)

[24.1.3 Sol-20 Keyboard 89](#_Toc140521754)

[24.1.4 Sol-20 Example Usage 89](#_Toc140521755)

[24.1.5 Sol-20 ENT Files 90](#_Toc140521756)

[25 PMMI MM-103 MODEM and Communications Adapter 91](#_Toc140521757)

[25.1 Overview 91](#_Toc140521758)

[25.1.1 PMMI Device Parameters 91](#_Toc140521759)

[25.1.2 PMMI Example Usage 91](#_Toc140521760)

[25.1.3 PMMI and SIMH Timing 93](#_Toc140521761)

[26 Hayes Micromodem 100 95](#_Toc140521762)

[26.1 Overview 95](#_Toc140521763)

[26.1.1 HAYES Device Parameters 95](#_Toc140521764)

[26.1.2 HAYES Example Usage 95](#_Toc140521765)

[26.1.3 HAYES and SIMH Timing 97](#_Toc140521766)

[27 ImageDisk (IMD) Disk Image Support in SIMH 99](#_Toc140521767)

[27.1 Overview 99](#_Toc140521768)

[27.2 References 99](#_Toc140521769)

[28 CP/M-68K Simulation 99](#_Toc140521770)

This memorandum documents the Altair 8800 Simulator.

# Simulator Files

scp.h

sim\_console.h

sim\_defs.h

sim\_fio.h

sim\_rev.h

sim\_sock.h

sim\_timer.h

sim\_tmxr.h

sim\_serial.h

sim\_tape.c

sim\_disk.c

sim\_ether.c

scp.c

sim\_console.c

sim\_fio.c

sim\_sock.c

sim\_timer.c

sim\_tmxr.c

sim\_serial.c

sim\_tape.c

sim\_disk.c

sim\_ether.c

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)

AltairZ80/altairz80\_defs.h

altairz80\_cpu\_nommu.c

altairz80\_cpu.c

altairz80\_dsk.c

altairz80\_hdsk.c

altairz80\_mhdsk.c (MITS/Pertec 88-HDSK hard disk support by Mike Douglas)

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)

mmd.c (Morrow Micro-Decision support by Patrick Linstruth)

n8vem.c (N8VEM Single-Board Computer I/O module by Howard M. Harte)

sol20.c (Processor Technology Sol-20 Terminal Computer by Patrick Linstruth)

s100\_2sio.c (MITS 88-2SIO Serial Adapter by Patrick Linstruth)

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\_dj2d.c (Morrow Disk Jockey 2D Model B Disk Controller by Patrick Linstruth)

s100\_djhdc.c (Morrow Disk Jockey HDC-DMA Hard Disk Controller by Howard M. Harte)

s100\_fif.c (IMSAI FIF Disk Controller by Ernie Price)

s100\_hayes.c (Hayes 80-103A and Micromodem 100 by Patrick Linstruth)

s100\_hdc1001.c (Advanced Digital Corporation (ADC) HDC-1001 Hard Disk Controller by Howard M. Harte)

s100\_icom.c (iCOM FD3712/FD3812 Flexible Disk System by Patrick Linstruth)

s100\_if3.c (CompuPro System Support 1 by Howard M. Harte)

s100\_jadedd.c (JADE Double D Disk Controller by Patrick Linstruth)

s100\_mdriveh.c (CompuPro M-DRIVE/H Controller by Howard M. Harte)

s100\_mdsa.c (NorthStar MDS-A Single Density Disk Controller by Mike Douglas)

s100\_mdsad.c (NorthStar MDS-AD disk controller by Howard M. Harte)

s100\_pmmi.c (PMMI MM-103 MODEM by Patrick Linstruth)

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)

s100\_tarbell.c (Altair Tarbell controller by Patrick Linstruth)

s100\_tdd.c (Tarbell Double-Density Floppy Controller by Howard M. Harte)

s100\_vdm1.c (Processor Technology VDM-1 Video Display Module by Patrick Linstruth)

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)

insns.h  (8086 Disassembler by Simon Tatham and Julian Hall)

nasm.h (8086 Disassembler by Simon Tatham and Julian Hall)

disasm.c (8086 Disassembler by Simon Tatham and Julian Hall)

insnsd.c (8086 Disassembler by Simon Tatham and Julian Hall)

i86.h (8086 CPU by Jim Hudgens)

i86\_decode.c (8086 CPU by Jim Hudgens)

i86\_ops.c (8086 CPU by Jim Hudgens)

i86\_prim\_ops.c (8086 CPU by Jim Hudgens)

m68k.h (Motorola M68000 CPU by Karl Stenerud)

m68kasm.y(.txt) (Motorola M68000 assembler by Holger Veit, Bison source)

m68kasm.c (Motorola M68000 assembler by Holger Veit, Bison output)

m68kconf.h (Motorola M68000 CPU by Karl Stenerud)

m68ksim.c (CPU driver for CP/M-68K simulation, based on work by David W. Schultz)

m68ksim.h (CPU driver for CP/M-68K simulation, based on work by David W. Schultz)

m68kcpu.c (Motorola MC68000 CPU by Karl Stenerud)

m68kcpu.h (Motorola MC68000 CPU by Karl Stenerud)

m68kdasm.c (Motorola MC68000 CPU by Karl Stenerud, disassembler)

m68kopac.c (Motorola MC68000 CPU by Karl Stenerud)

m68kopdm.c (Motorola MC68000 CPU by Karl Stenerud)

m68kopnz.c (Motorola MC68000 CPU by Karl Stenerud)

m68kops.c (Motorola MC68000 CPU by Karl Stenerud)

m68kops.h (Motorola MC68000 CPU by Karl Stenerud)

# Revision History

* 30-Jul-2023 Patrick Linstruth (added CTS command for M2SIO)
* 17-Jul-2023 Peter Schorn (corrections, formatting)
* 20-Mar-2023 Patrick Linstruth (added support for Processor Technology VDM-1 Video Display Module and Sol-20 Terminal Computer.)
* 16-Jan-2022 Howard Harte (updated ADC, CompuPro documentation, added Seattle Computer Products documentation.)
* 22-Nov-2022 Patrick Linstruth (added MEM and REG commands)
* 19-Nov-2022 Patrick Linstruth (added support for Morrow Micro Decision)
* 18-Jan-2021 Patrick Linstruth (added support for Morrow Disk Jockey 2D original disk controller)
* 10-Jan-2021 Peter Schorn (corrections)
* 01-Jan-2021 Patrick Linstruth (added support for Morrow Disk Jockey 2D Model B disk controller)
* 27-Nov-2020, Peter Schorn (added port command for PTP/PTR)
* 27-Nov-2020, Patrick Linstruth (added support for iCOM FD3712/FD3812 FDS)
* 14-Aug-2020, Patrick Linstruth (added support for Hayes S-100 modems)
* 02-Jul-2020, Patrick Linstruth (added support for PMMI MM-103 MODEM),
* 30-Jun-2020 Patrick Linstruth (added DSDD support to Tarbell disk controller)
* 29-Jun-2020 Patrick Linstruth (added support for MITS 88-2SIO)
* 10-Jun-2020, Mike Douglas (added documentation for North Star MDS-A (single density FDC))
* 05-Jun-2020, Peter Schorn (additional documentation for CPU SET commands and CPU pseudo registers)
* 04-Jun-2020, Patrick Linstruth (added support for the JADE Double D disk controller)
* 24-May-2020, Peter Schorn (added M68000 assembler based on Holger Veit’s Bison code)
* 28-Apr-2020, Patrick Linstruth (added support for CPU instruction history)
* 09-Dec-2019, Peter Schorn (small updates)
* 05-Dec-2019, Patrick Linstruth (added support for the Altair Tarbell SSSD disk controller)
* 27-Dec-2015, Peter Schorn (updated SIO device documentation)
* 24-May-2014, Peter Schorn (added support for the Altair Mini-Disk contributed by Mike Douglas)
* 6-May-2014, Peter Schorn (added Motorola MC68000 CPU, updated HDSK and added a driver for CP/M-68K simulation)
* 21-Apr-2014, Peter Schorn (added debug flags for the MHDSK device and support for the NEXT command)
* 29-Mar-2014, Peter Schorn (added support for the MITS/Pertec 88-HDSK hard disk contributed by Mike Douglas)
* 15-Apr-2013, Peter Schorn (added correct cycle count timing for 8080 CPU, improved .IMD file processing, SIO ‘C’ switch was renamed to ‘N’)
* 24-Aug-2012, Peter Schorn (added capability to HDSK device for IMD disk processing)
* 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)
* 14-Apr-2007, Peter Schorn (added documentation for Howard M. Harte’s hard disk extensions)
* 05-Jan-2007, Peter Schorn (added networking capability, included CP/NET and CPNOS)
* 26-Nov-2006, Peter Schorn (SIO can now be attached to a file, SIO rewritten for better efficiency)
* 15-Oct-2006, Peter Schorn (updated CP/M 2 operating system and application software description)
* 17-Sep-2006, Peter Schorn (added Altair Basic 5.0 to the sample software, corrected TTY/ANSI description)
* 21-Aug-2006, Peter Schorn (added MINOL and VTL-2 software, retyping courtesy of Emmanuel ROCHE, fixed a bug in memory breakpoints and added a create (“C”) switch to the attach command)
* 24-Jan-2006, Peter Schorn (transcribed documentation to Word / PDF format)
* 05-Apr-2005, Peter Schorn (removed bogus t-state stepping support)
* 24-Jul-2004, Peter Schorn (updated CP/M 2 and SPL packages)
* 12-Apr-2004, Peter Schorn (added MAP/NOMAP capability to switch off key mapping)
* 26-Jan-2004, Peter Schorn (added support for t-state stepping)
* 25-Feb-2003, Peter Schorn (added support for real time simulation)
* 9-Oct-2002, Peter Schorn (added support for simulated hard disk)
* 28-Sep-2002, Peter Schorn (number of tracks per disk can be configured)
* 19-Sep-2002, Peter Schorn (added WARNROM feature)
* 31-Aug-2002, Peter Schorn (added extended ROM features suggested by Scott LaBombard)
* 4-May-2002, Peter Schorn (added description of MP/M II sample software)
* 28-Apr-2002, Peter Schorn (added periodic timer interrupts and three additional consoles)
* 15-Apr-2002, Peter Schorn (added memory breakpoint)
* 7-Apr-2002, Peter Schorn (added ALTAIRROM / NOALTAIRROM switch)

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

# 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.

# 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.

## CPU

We have three 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 M68K Simulates Motorola M68000 CPU. This also enables 16 MB of memory by default. After the M68K CPU type has been selected, the specific variant can be set.

SET CPU 68000 Sets the M68K CPU Variant to 68000.

SET CPU 68010 Sets the M68K CPU Variant to 68010.

SET CPU 68020 Sets the M68K CPU Variant to 68020.

SET CPU 68030 Sets the M68K CPU Variant to 68030.

SET CPU 68040 Sets the M68K CPU Variant to 68040.

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

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

SET CPU 4K

SET CPU 8K

SET CPU 12K

SET CPU 16K

… (in 4K steps)

SET CPU 64K All these set various CPU memory configurations.

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

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

SET CPU NONBANKED Disables banked memory support.

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

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

SET CPU NOALTAIRROM Disables standard Altair ROM behavior.

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

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

SET CPU AZ80 Sets the RAM type to AltairZ80 RAM for 8080 / Z80 / 8086.

SET CPU HRAM Sets the RAM type to NorthStar HRAM for 8080 / Z80 / 8086.

SET CPU VRAM Sets the RAM type to Vector RAM for 8080 / Z80 / 8086.

SET CPU CRAM Sets the RAM type to Cromemco RAM for 8080 / Z80 / 8086.

SET CPU SWITCHER Sets the CPU switcher port for 8080 / Z80 / 8086 from the CPU pseudo-register SWITCHERPORT.

SET CPU NOSWITCHER Resets the CPU switcher port for 8080 / Z80 / 8086.

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.

SET CPU HISTORY Clears CPU instruction history buffer (8080 and Z80).

SET CPU HISTORY=0 Disables CPU instruction history (8080 and Z80).

SET CPU HISTORY=<n> Enables CPU instruction history buffer with a size of <n> (8080 and Z80).

SHOW CPU HISTORY Displays CPU instruction history buffer in CP/M DDT format (8080 and Z80).

SHOW CPU HISTORY=<n> Displays last <n> entries of the CPU instruction history buffer in CP/M DDT format (8080 and Z80).

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.

### Registers for the 8080 and Z80

CPU registers include the following for the Z80 / 8080:

**Name Size Comment**

PC 20 The Program Counter for the 8080 and Z80

AF 16 The accumulator (8 bits) and the flag register

F = S Z - AC - P/V N C

S = Sign flag.

Z = Zero Flag.

- = not used (undefined)

AC = Auxiliary Carry flag.

P/V = Parity flag on 8080 (Parity / Overflow flag on Z80)

- = not used (undefined)

N = Internal sign flag

C = Carry flag.

BC 16 The BC register pair.

Register B is the high 8 bits, C is the lower 8 bits

DE 16 The DE register pair.

Register D is the high 8 bits, E is the lower 8 bits.

HL 16 The HL register pair.

Register H is the high 8 bits, L is the lower 8 bits.

AF1 16 The alternate AF register (on Z80 only)

BC1 16 The alternate BC register (on Z80 only)

DE1 16 The alternate DE register (on Z80 only)

HL1 16 The alternate HL register (on Z80 only)

IX 16 The IX index register (on Z80 only)

IY 16 The IY index register (on Z80 only)

IFF 8 Interrupt flag (on Z80 only)

IR 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)

COMMONLOW 1 When set to 1, the common area is in low RAM. When set to 0 (the default,) the common area is in high RAM. The OASIS operating system requires this set to 1.

PCQ 16 x 64 Circular buffer of the last 64 PC jump targets. This can be viewed with “e pcq[0-63]”.

PCQP 16 The head index of the circular buffer

CLOCK 32 The clock speed of the simulated CPU (8080 / Z80) 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.

SLICE 16 The slice length in milliseconds for the clock speed simulation. The default is 10 ms.

TSTATES 32 The number of executed t-states (8080 / Z80) – read only.

CAPACITY 32 Capacity of the RAM – read only. Use the SET commands above to set the memory capacity.

PREVCAP 32 The previous capacity of the RAM – read only.

SWITCHERPORT The 8-bit port number of the CPU switcher port. The default is FD.

### Registers for the 8086

CPU registers include the following for the 8086:

**Name Size Comment**

AX 16 AX general purpose register

AL 8 low 8 bits of AX

AH 8 high 8 bits of AX

BX 16 BX general purpose register

BL 8 low 8 bits of BX

BH 8 high 8 bits of BX

CX 16 CX general purpose register

CL 8 low 8 bits of CX

CH 8 high 8 bits of CX

DX 16 DX general purpose register

DL 8 low 8 bits of DX

DH 8 high 8 bits of DX

BP 16 Base Pointer

SI 16 Source Index

DI 16 Destination Index

SP86 16 Stack Pointer

CS 16 Code Segment

DS 16 Data Segment

ES 16 Extra Segment

SS 16 Stack Segment

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 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

### Registers for the MC68000

CPU registers include the following for the MC68000

**Name Size Comment**

M68K\_D0 32 D0 – general purpose data register

M68K\_D1 32 D1 – general purpose data register

M68K\_D2 32 D2 – general purpose data register

M68K\_D3 32 D3 – general purpose data register

M68K\_D4 32 D4 – general purpose data register

M68K\_D5 32 D5 – general purpose data register

M68K\_D6 32 D6 – general purpose data register

M68K\_D7 32 D7 – general purpose data register

M68K\_A0 32 A0 – general purpose address register

M68K\_A1 32 A1 – general purpose address register

M68K\_A2 32 A2 – general purpose address register

M68K\_A3 32 A3 – general purpose address register

M68K\_A4 32 A4 – general purpose address register

M68K\_A5 32 A5 – general purpose address register

M68K\_A6 32 A6 – general purpose address register

M68K\_A7 32 A7 – general purpose address register

M68K\_PC 32 PC – program counter

M68K\_SR 32 SR – status register

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

T1 T0 S M 0 I2 I1 I0 0 0 0 X N Z V C

T1 = Trace Enable T1

T0 = Trace Enable T0

S = Supervisor / User State

M = Master / Interrupt State

I2 = Interrupt Priority Mask I2

I1 = Interrupt Priority Mask I1

I0 = Interrupt Priority Mask I0

X = Extend

N = Negative

Z = Zero

V = Overflow

C = Carry

M68K\_SP 32 SP – stack pointer (located in A7)

M68K\_USP 32 USP – user stack pointer

M68K\_ISP 32 ISP – interrupt stack pointer

M68K\_MSP 32 MSP – master stack pointer

M68K\_SFC 32 SFC – source function code register

M68K\_DFC 32 DFC – destination function code register

M68K\_VBR 32 VBR – vector base register

M68K\_CACR 32 CACR – cache control register

M68K\_CAAR 32 CAAR – cache address register

M68K\_PREF\_ADDR 32 last prefetch address

M68K\_PREF\_DATA 32 last prefetch data

M68K\_PPC 32 PPC – previous value of program counter

M68K\_IR 32 IR – instruction register

M68K\_CPU\_TYPE 32 CPU type (read only), 1 for MC68000

MMIOBASE 24 Memory Mapped I/O Base Address

MMIOSIZE 17 Memory Mapped I/O Size

M68KVAR 17 M68K CPU Type

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.

## 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 SIOWLEV 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 milliseconds 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. Use “D SLEEP <n>” to change the number of milliseconds – the default value is 1.

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).

The standard setting for the console / keyboard port is equivalent to

SET SIO PORT=10/0/1/0/2/T/3/F

SET SIO PORT=11/0/1/0/2/T/3/T

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/PTR can be configured in SIMH with the following command:

SET PTP PORT=StatusPort/DataPort or SET PTR PORT=StatusPort/DataPort

StatusPort: two digit hex address of the new status port (originally 0x12)

DataPort: two digit hex address of the new data port (originally 0x13)

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

IN All IN operations on the PTP ports (status and data)

OUT All OUT operations on the PTP ports (status and data)

CMD All OUT operations which are interpreted as commands

VERBOSE All warning messages (currently: use of unattached PTP)

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

IN All IN operations on the PTR ports (status and data)

OUT All OUT operations on the PTR ports (status and data)

CMD All OUT operations which are interpreted as commands

VERBOSE All warning messages (currently: use of unattached PTR, attempt to read past end of attached file)

## MITS 88-2SIO Serial Adapter (M2SIO)

This M2SIO device provides an alternative to the SIO device for serial communications. The M2SIO device provides two independent MITS 88-2SIO serial ports, M2SIO0 and M2SIO1, with full support for attaching to sockets and host serial ports.

By default, the M2SIO0 port directs I/O to/from the screen and keyboard. The second port also directs its output to the screen. The default I/O base port for M2SIO0 is 10H and M2SIO1 is 12H.

Unlike the SIO device, the M2SIO device provides raw ports, meaning data is sent and received as is without any translation or other manipulations. Also, there are no PTP or PTR devices.

Before either of these ports can be used, they must be enabled:

SET M2SIO0 ENA Enable the first 88-2SIO port (replaces SIO port).

SET M2SIO1 ENA Enable the second 88-2SIO port (replaces PTP and PTR ports).

Once enabled, the M2SIO ports can be configured in SIMH with the following commands, where ‘x’ is 0 for port 0 and 1 for port 1:

SET M2SIOx IOBASE Sets MITS 2SIO base I/O address.

SET M2SIOx CONSOLE Port checks console for input.

SET M2SIOx NOCONSOLE Port does not check console for input.

SET M2SIOx DTR DTR follows RTS.

SET M2SIOx NODTR DTR does not follow RTS (default).

SET M2SIOx CTS Force CTS active low.

SET M2SIOx NOCTS CTS follows status line (default).

SET M2SIOx DCD Force DCD active low.

SET M2SIOx NODCD DCD follows status line (default).

SET M2SIOx BAUD=val Set baud rate (default=9600).

SET M2SIOx DEBUG Enables debugging for device M2SIOx.

SET M2SIOx NODEBUG Disables debugging for device M2SIOx.

The M2SIO device implements these registers:

**Name Size Comment**

M2STAx 8 2SIO port x status register

M2CTLx 8 2SIO port x control register

M2RXDx 8 2SIO port x rx data buffer

M2TXDx 8 2SIO port x tx data buffer

M2TXPx 8 2SIO port x tx data pending

M2CONx 1 2SIO port x connection status

M2RIEx 1 2SIO port x receive interrupt enable

M2TIEx 1 2SIO port x transmit interrupt enable

M2RTSx 1 2SIO port x RTS status (active low)

M2RDRFx 1 2SIO port x RDRF status

M2TDREx 1 2SIO port x TDRE status

M2DCDx 1 2SIO port x DCD status (active low)

M2CTSx 1 2SIO port x CTS status (active low)

M2OVRNx 1 2SIO port x OVRN status

M2WAITx 32 2SIO port x wait cycles

### Using the M2SIO device with serial ports

It is possible to attach host serial ports to the M2SIO ports using the “Attach” command. The following example shows how to attach the second 88-2SIO port (M2SIO1) to a serial port to on a UNIX-type platform:

sim> set m2sio1 enable

sim> attach m2sio1 connect=/dev/cu.USA19H14411P1.1

The 88-2SIO does not have a DTR modem output. If you need DTR, configure the port to have DTR follow RTS with the “set m2siox dtr” command.

The 88-2SIO will not enable the receiver unless DCD is present. If you need the port to receive without DCD, configure the port to force DCD to an active low state with the “SET M2SIOx DCD” command.

The 88-2SIO will not transmit unless CTS is present. If you need the port to transmit without CTS, configure the port to force CTS to an active low state with the “SET M2SIOx CTS” command.

### Using the M2SIO device with sockets

The M2SIO devices may also be attached to sockets. The following example show how to attach the second 88-2SIO port to a socket listening on port 8800:

sim> set m2sio1 enable ;Enable M2SIO1 device

sim> set m2sio1 dtr ;TMXR sockets require DTR

sim> attach m2sio1 -U :8800 ;Bind to all interfaces on port 8800

Now that the simulator is listening on port 8800, you may connect to the simulator’s 88-2SIO port with “telnet <ip address> 8800”, where <ip address> is the IP address of your computer.

### M2SIO limitations

The CP/M utilities R.COM and W.COM use the PTP and PTR devices to transfer files between the simulator and host file system. If the M2SIO1 device is enabled using the default 12H port address, these utilities will no longer work as the M2SIO1 device replaces the PTP and PTR devices.

The M2SIO device makes it possible to run communications software for CP/M, such as PCGET/PUT, MEX, MODEM 7, and BYE using host serial ports and sockets. Many of these programs use loops for timeout processing. For these timeouts to work properly, it is necessary to use the “Clock” register to simulate your CPU’s speed.

## 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)

IN All IN operations on the SIMH port

OUT All OUT operations on the SIMH port

CMD All illegal commands

VERBOSE All other warning or error messages

There are also some useful simulator-specific commands available:

MEM Provides a HEX and ASCII dump of memory

sim> mem

DF00 C3 5C E2 C3 58 E2 7F 00 43 6F 70 79 72 69 67 68 .\..X...Copyrigh

DF10 74 20 31 39 37 39 20 28 63 29 20 62 79 20 44 69 t 1979 (c) by Di

DF20 67 69 74 61 6C 20 52 65 73 65 61 72 63 68 20 20 gital Research

DF30 20 20 20 20 00 00 00 00 00 00 00 00 00 00 00 00 ............

DF40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

DF50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

DF60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

DF70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

DF80 00 00 00 00 00 00 00 00 08 DF 00 00 5F 0E 02 C3 ............\_...

DF90 05 00 C5 CD 8C DF C1 C9 3E 0D CD 92 DF 3E 0A C3 ........>....>..

DFA0 92 DF 3E 20 C3 92 DF C5 CD 98 DF E1 7E B7 C8 23 ..> ........~..#

DFB0 E5 CD 8C DF E1 C3 AC DF 0E 0D C3 05 00 5F 0E 0E .............\_..

DFC0 C3 05 00 CD 05 00 32 EE E6 3C C9 0E 0F C3 C3 DF ......2..<......

DFD0 AF 32 ED E6 11 CD E6 C3 CB DF 0E 10 C3 C3 DF 0E .2..............

DFE0 11 C3 C3 DF 0E 12 C3 C3 DF 11 CD E6 C3 DF DF 0E ................

DFF0 13 C3 05 00 CD 05 00 B7 C9 0E 14 C3 F4 DF 11 CD ................

REG Displays a DDT-formatted dump of 8080/Z80 registers

sim> reg

C0Z1M0E1I1 A=00 B=007F D=DF06 H=EA0E S=EA37 P=FA6D NOP

sim> set on

sim> on error reg ;Display registers upon STOP, STEP, and BREAKs

sim> s

Step expired, PC: 0FA69 (IN 10h)

C0Z1M0E1I1 A=00 B=007F D=DF06 H=EA0E S=EA37 P=FA69 IN 10h

sim> s

Step expired, PC: 0FA6B (ANI 01h)

C0Z1M0E1I1 A=02 B=007F D=DF06 H=EA0E S=EA37 P=FA6B ANI 01h

sim> s

Step expired, PC: 0FA6D (JZ 0FA69h)

C0Z1M0E1I1 A=00 B=007F D=DF06 H=EA0E S=EA37 P=FA6D JZ 0FA69h

sim>

## 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 Mini-Disk support was added by Mike Douglas in May 2014.

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)

## The 88-HDSK controller

The 88-HDSK from MITS/Pertec consists of a 5mb removable platter and a fixed 5mb platter. Each platter is double sided. Head 0 and 1 are the top and bottom surface of the removable platter and head 2 and 3 are the top and bottom surface of the fixed platter. Hard disk BASIC treats the two platters as two separate drives. Each platter has 406 cylinders with 24 sectors per track and 256 bytes per sector.

The disk image file starts with head 0, track 0, sector 0 (0,0,0) through (0,0,23), followed by head 1, track 0, sector 0 (1,0,0) through (1,0,23). The pattern then repeats starting with (0,1,0). The external hard disk is accessed through eight ports of a 4-PIO card at I/O addresses A0h-A7h. The module was written by Mike Douglas in March 2014 and the disk images were provided by Martin Eberhard.

The MHDSK device can be configured with

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

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

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

READ All read operations of the disk

WRITE All write operations on the disk

VERBOSE All other operations of the disk

## 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)
* CPM68K (simulated hard drive 16’384 kB capacity for CP/M-68K)
* 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. 256 kB capacity, no skew)
* SSSD8S (standard 8" Single Side Single Density floppy disk with 77 tracks of 26 sectors with 128 bytes, i.e. 256 kB capacity, standard skew factor 6)
* SSDD8 (standard 8" Single Side Double Density floppy disk with 77 tracks of 26 sectors with 256 bytes, i.e. 512 kB capacity, no skew)
* SSDD8S (standard 8" Single Side Double Density floppy disk with 77 tracks of 26 sectors with 256 bytes, i.e. 512 kB capacity, standard skew factor 6)
* DSDD8 (standard 8" Double Side Double Density floppy disk with 77 tracks of 26 sectors with 512 bytes, i.e. 1’025 kB capacity, no skew)
* DSDD8S (standard 8" Double Side Double Density floppy disk with 77 tracks of 26 sectors with 512 bytes, i.e. 1’025 kB capacity, standard skew factor 6)
* 512SSDD8 (standard 8" Single Side Double Density floppy disk with 77 tracks of 15 sectors with 512 bytes, i.e. 591 kB capacity, no skew)
* 512DSDD8 (standard 8" Double Side Double Density floppy disk with 77 tracks of 15 sectors with 512 bytes, i.e. 1’183 kB capacity, no skew)
* 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)
* V1050 (410 kB, Visual Technology Visual 1050, 512 byte sectors for CP/M 3)
* Note1: The CP/M 3 implementation that comes with AltairZ80 automatically adapts to the attached hard disk. It supports sector sizes of 128 bytes, 256 bytes and 512 bytes.
* Note2: 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.
* Note3: 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)

## 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.

DROP Show a message when a connection is dropped.

IN Show all data received from the network.

OUT Show all data transmitted to the network.

# 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. The software is available from <https://schorn.ch/altair.html>.

## CP/M Version 2.2

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

To boot CP/M:

sim> attach dsk cpm2.dsk

sim> boot dsk

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

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

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

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

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

## CP/M Version 3 with banked memory

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

Running CP/M 3 with banked memory:

sim> attach dsk cpm3.dsk

sim> reset cpu

sim> set cpu banked

sim> set cpu itrap

sim> boot dsk

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

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

| Name | Ext | Size | Comment |
| --- | --- | --- | --- |
| ASM | .COM | 8K | CP/M assembler |
| ASSIGN | .SYS | 2K |  |
| BDOS3 | .SPR | 10K |  |
| BIOS3 | .MAC | 28K | CP/M 3 BIOS source for Altair SIMH |
| BIOS3 | .SPR | 4K |  |
| BNKBDOS3 | .SPR | 14K |  |
| BNKBIOS3 | .SPR | 4K |  |
| BOOT | .COM | 2K | transfer control to boot ROM |
| BOOTGEN | .COM | 2K | put a program on the boot sectors |
| CCP | .COM | 4K |  |
| COPYSYS | .COM | 2K |  |
| CPM3 | .SYS | 18K |  |
| CPMLDR | .MAC | 38K | CP/M 3 loader assembler source |
| DATE | .COM | 4K | date utility |
| DDT | .COM | 6K | 8080 debugger |
| DDTZ | .COM | 10K | Z80 debugger |
| DEFS | .LIB | 2K | include file for BIOS3.MAC to create banked CP/M 3 |
| DEVICE | .COM | 8K |  |
| DIF | .COM | 4K | determine differences between two files |
| DIR | .COM | 16K | directory utility |
| DO | .COM | 6K | batch processing (SUBMIT.COM) |
| DUMP | .COM | 2K |  |
| ED | .COM | 10K |  |
| ERASE | .COM | 4K |  |
| GENCOM | .COM | 16K |  |
| GENCPM | .COM | 22K |  |
| GENCPM | .DAT | 4K | CP/M generation information for banked version |
| GENCPMNB | .DAT | 4K | CP/M generation information for non-banked version |
| GET | .COM | 8K |  |
| HELP | .COM | 8K | help utility |
| HELP | .HLP | 62K | help files |
| HEXCOM | .CPM | 2K |  |
| HIST | .UTL | 2K |  |
| INITDIR | .COM | 32K |  |
| L80 | .COM | 12K | Microsoft linker |
| LDR | .COM | 4K | CP/M loader with optimized loader BIOS |
| LDRBIOS3 | .MAC | 14K | optimized (for space) loader BIOS |
| LIB | .COM | 8K | Digital Research librarian |
| LINK | .COM | 16K | Digital Research linker |
| LOAD | .COM | 2K |  |
| M80 | .COM | 20K | Microsoft macro assembler |
| MC | .SUB | 2K | assemble and link an assembler program |
| MCC | .SUB | 2K | read, assemble and link an assembler program |
| PATCH | .COM | 4K |  |
| PIP | .COM | 10K | Peripheral Interchange Program |
| PROFILE | .SUB | 2K | commands to be executed at start up |
| PUT | .COM | 8K |  |
| R | .COM | 4K | read files from SIMH environment |
| RENAME | .COM | 4K |  |
| RESBDOS3 | .SPR | 2K |  |
| RMAC | .COM | 14K | Digital Research macro assembler |
| RSETSIMH | .COM | 2K | reset SIMH interface |
| SAVE | .COM | 2K |  |
| SCB | .MAC | 2K |  |
| SET | .COM | 12K |  |
| SETDEF | .COM | 6K |  |
| SHOW | .COM | 10K |  |
| SHOWSEC | .COM | 4K | show sectors on a disk |
| SID | .COM | 8K | 8080 debugger |
| SUBMIT | COM | 6K | batch processing |
| SYSCOPY | .COM | 2K | copy system tracks between disks |
| SYSCPM3 | .SUB | 2K | create banked CP/M 3 system |
| TRACE | .UTL | 2K |  |
| TSHOW | .COM | 2K | show split time |
| TSTART | .COM | 2K | create timer and start it |
| TSTOP | .COM | 2K | show final time and stop timer |
| TYPE | .COM | 4K |  |
| UNERA | .COM | 2K | un-erase a file |
| W | .COM | 4K | write files to SIMH environment |
| XREF | .COM | 16K | cross reference utility |
| ZSID | .COM | 10K | Z80 debugger |

## MP/M II with banked memory

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

sim> attach dsk mpm.dsk

sim> set cpu itrap

sim> set cpu z80

sim> set cpu altairrom

sim> set cpu banked

sim> attach sio 23

sim> d common b000

sim> boot dsk

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

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

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

## 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 (“telnet 127.0.0.1” or “telnet localhost”) to the simulator and type “mpm <return>” at the “A>” command prompt to start the MP/M CP/NET server.

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

sim> attach dsk cpnetclient.dsk

sim> set cpu 64k

sim> set cpu noitrap

sim> set cpu z80

sim> set cpu altairrom

sim> set cpu nonbanked

sim> reset cpu

sim> set sio ansi

sim> set net client

sim> at net 127.0.0.1:4000

sim> boot dsk

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

A>cpnetldr<return> ; loads CP/NET client

A>login<return> ; to login

A>network b:=a: ; to map server drive A: to client drive B:

A>dir b: ; shows the contents of the server drive A:

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

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

## CPNOS

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

sim> set cpu 64k

sim> set cpu noitrap

sim> set cpu z80

sim> set cpu noaltairrom

sim> set cpu nonbanked

sim> reset cpu

sim> set sio ansi

sim> set net client

sim> at net 127.0.0.1:4000

sim> load cpnos.com f000

sim> g f000

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

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

## CP/M application software

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

- SPL a Small Programming Language with a suite of sample programs

- PROLOGZ a Prolog interpreter written in SPL with sources

- PASCFORM a Pascal pretty printer written in Pascal

- Pascal MT+ Pascal language system needed to compile PASCFORM

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

sim> attach dsk1 app.dsk

before booting CP/M.

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

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

## 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]

## Altair DOS Version 1.0

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

sim> d tracks[0-7] 77 ;set to Altair settings

sim> set cpu altairrom

sim> attach dsk altdos.dsk

sim> set sio upper

sim> go ff00

MEMORY SIZE? [return]

INTERRUPTS? N [return]

HIGHEST DISK NUMBER? [0 return] (3 here = 4 drive system)

HOW MANY DISK FILES? [3 return]

HOW MANY RANDOM FILES? [2 return]

056449 BYTES AVAILABLE

DOS MONITOR VER 1.0

COPYRIGHT 1977 BY MITS INC

.[MNT 0]

.[DIR 0]

## 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

## 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

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 4kbas40.bin;load it at 0

sim> g ;and start it

MEMORY SIZE? [return]

TERMINAL WIDTH? [return]

WANT SIN? [Y]

61900 BYTES FREE

4K BASIC 4.0

COPYRIGHT MITS 1976

OK

## Altair 8k Basic

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

sim> set cpu 8080 ;note 8k Basic will not run on a Z80 CPU

sim> set sio upper ;8k Basic does not like lower case letters as input

sim> set sio ansi ;8k Basic produces 8-bit output, strip to seven bits

sim> d sr 8 ;good setting for the Switch Register

sim> load 8kbas.bin 0 ;load it at 0

sim> go 0 ;and start it

MEMORY SIZE? [A]

WRITTEN FOR ROYALTIES BY MICRO-SOFT

MEMORY SIZE? [return]

TERMINAL WIDTH? [return]

WANT SIN-COS-TAN-ATN? [Y]

58756 BYTES FREE

ALTAIR BASIC REV. 4.0

[EIGHT-K VERSION]

COPYRIGHT 1976 BY MITS INC.

OK

## 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

## Altair Disk Extended Basic Version 300-5-C

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

sim> d tracks[0-7] 77 ;set to Altair settings

sim> at dsk extbas5.dsk

sim> g 0

MEMORY SIZE? [return]

LINEPRINTER? [C]

HIGHEST DISK NUMBER? [0]

HOW MANY FILES? [3]

HOW MANY RANDOM FILES? [3]

42082 BYTES FREE

ALTAIR DISK EXTENDED BASIC

VERSION 300-5-C [01NOV78]

COPYRIGHT 1978 BY MITS INC.

OK

## Altair Disk Extended Basic Version 5.0

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

sim> d tracks[0-7] 77 ;set to Altair settings

sim> at dsk disbas50.dsk

sim> d sr 8

sim> load disbas50.bin 0

sim> g 0

MEMORY SIZE? [return]

LINEPRINTER? [C]

HIGHEST DISK NUMBER? [return]

HOW MANY FILES? [3]

HOW MANY RANDOM FILES? [3]

41695 BYTES FREE

ALTAIR BASIC 5.0 [14JUL78]

[DISK EXTENDED VERSION]

COPYRIGHT 1978 BY MITS INC.

OK

## Altair Hard Disk Basic 300-5-C

This version of Basic was provided by Martin Eberhard and uses the MHDSK device contributed by Mike Douglas. This software is part of the archive **althdsw.zip**. To execute use the following commands or type “AltairZ80 hdbasic” in a command shell.

sim> set sio ansi

sim> set sio nobell ;avoid problems with accounting software

sim> attach mhdsk0 hdbasic-300-5-c-acct.dsk

sim> attach mhdsk1 hdbasic-300-5-f.dsk

sim> attach dsk0 fdbasic-300-5-f.dsk ;floppy disk for copying

sim> boot mhdsk

HDBL 1.01

LOADING

MEMORY SIZE? [return]

LINEPRINTER? [C]

HIGHEST DISK NUMBER? [2]

HOW MANY FILES? [4]

CURRENT DAY? [4]

CURRENT MONTH? [4]

CURRENT YEAR? [88]

32762 BYTES FREE

ALTAIR HARD DISK BASIC

VERSION 300-5-C [02NOV78]

COPYRIGHT 1978 BY MITS INC.

OK

## Altair programming languages VTL-2 and MINOL

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

## UCSD Pascal II.0

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

Useful hints:

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

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

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

DSK0:

SYSTEM.MICRO 19 9-Feb-79 10 512 Datafile

SYSTEM.FILER 28 10-Apr-79 29 512 Codefile

SYSTEM.EDITOR 45 10-Feb-79 57 512 Codefile

SYSTEM.LINKER 22 10-Feb-79 102 512 Codefile

SYSTEM.COMPILER 68 8-Feb-79 124 512 Codefile

SYSTEM.SYNTAX 14 2-May-79 192 512 Textfile

SETUP.CODE 25 14-May-79 206 512 Codefile

BINDER.CODE 6 3-May-79 231 512 Codefile

SYSTEM.MISCINFO 1 10-Feb-79 237 192 Datafile

VT100GOTO.TEXT 4 10-Apr- 7 238 512 Textfile

VT100GOTO.CODE 2 10-Apr- 7 242 512 Codefile

SYSTEM.PASCAL 33 10-Apr- 7 244 512 Datafile

SYSTEM.LIBRARY 17 10-Apr- 7 277 512 Datafile

BASIC.COMPILER 30 11-Apr-79 294 512 Codefile

LOOP.TEXT 4 10-Apr- 7 324 512 Textfile

LOOP.CODE 4 10-Apr- 7 328 512 Codefile

Z80.ERRORS 8 28-Mar-79 332 70 Datafile

Z80.OPCODES 3 20-Dec-78 340 68 Datafile

SYSTEM.ASSMBLER 53 13-Apr-79 343 512 Codefile

< UNUSED > 98 396

19/19 files<listed/in-dir>, 396 blocks used, 98 unused, 98 in largest

## CP/M-68K

The software is part of the **cpm68k.zip** archive. To run it, type “altairz80 cpm68k” at your command prompt or alternatively invoke altairz80 and type “do cpm68k” at the “sim>” command prompt.

# Special simulator features

## 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.

## 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>.

## Breakpoints and instruction history (8080/Z80 only)

One can use breakpoints with the CPU instruction history. For example, suppose one wanted to determine what lead up to memory address 05C00 being accessed:

sim> break –m 5c00

sim> set cpu history=32

sim> g ff00

Memory access breakpoint [05c00h], PC: 0FF29 (MOV M,A)

sim> show cpu history=10

CPU: C0Z1M0E1I0 A=16 B=0000 D=0000 H=0000 S=0000 P=FF0D OUT 0FEh

CPU: C0Z1M0E1I0 A=12 B=0000 D=0000 H=0000 S=0000 P=FF0F MVI A,12h

CPU: C0Z1M0E1I0 A=12 B=0000 D=0000 H=0000 S=0000 P=FF11 OUT 0FEh

CPU: C0Z1M0E1I0 A=00 B=0000 D=0000 H=0000 S=0000 P=FF13 IN 0FEh

CPU: C0Z1M0E1I0 A=00 B=0000 D=0000 H=0000 S=0000 P=FF15 ORA A

CPU: C0Z1M0E1I0 A=00 B=0000 D=0000 H=0000 S=0000 P=FF16 JZ 0FF20h

CPU: C0Z1M0E1I0 A=00 B=0000 D=0000 H=5C00 S=0000 P=FF20 LXI H,5C00h

CPU: C0Z1M0E1I0 A=00 B=0000 D=FF33 H=5C00 S=0000 P=FF23 LXI D,0FF33h

CPU: C0Z1M0E1I0 A=00 B=0088 D=FF33 H=5C00 S=0000 P=FF26 MVI C,88h

CPU: C0Z1M0E1I0 A=31 B=0088 D=FF33 H=5C00 S=0000 P=FF28 LDAX D

# 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

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

#! /usr/bin/python

# -\*- coding: UTF-8 -\*-

# formatted for tab-stops 4

#

# By Peter Schorn, peter.schorn@acm.org, September 2006

#

#

# Transform an MBL file to a binary file suitable for loading with SIMH

#

# Structure of MBL files is as follows:

# <byte>+ 0x00 0x00

# (checkSum<byte> 0x3c count<byte> loadLow<byte> loadHigh<byte>

# <byte> \* count)+

# where the lower 8 bit of the load address are determined by loadLow

# and the upper 8 bit of the load address are determined by loadHigh

# For checkSum the following rules hold:

# For the first load record: 0

# For the second load record: (sum of all load bytes of the first

# load record) mod 256

# For the third and higher load records: (sum of all load bytes of

# the preceding load record - 1) mod 256

# A header with count = 0 or second position is unequal to 0x3c denotes

# end of file.

import sys

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

if len(sys.argv) <> 3:

print 'Usage %s inputmbl.bin output.bin\n' % sys.argv[0]

sys.exit(1)

f = file(sys.argv[1], 'rb')

b = f.read()

f.close()

i = b.index(chr(0) \* CHR0 + chr(0) + chr(0x3c)) + CHR0 + 2

result = [chr(0)] \* len(b)

k = 0

count = ord(b[i])

while count and (ord(b[i - 1]) == 0x3c):

l = ord(b[i + 1]) + (ord(b[i + 2]) << 8)

checkSum = 0

for j in range(count):

result[l + j] = b[i + 3 + j]

checkSum += ord(b[i + 3 + j])

expectedCheckSum = ord(b[i-2])

receivedCheckSum = expectedCheckSum

if k == 1:

receivedCheckSum = previousCheckSum & 255

elif k > 1:

receivedCheckSum = (previousCheckSum - 1) & 255

if receivedCheckSum <> expectedCheckSum:

print 'Checksum error in record %i. Got %02X and expected %02X ' % (

k, receivedCheckSum, expectedCheckSum)

i += count + 5

count = ord(b[i])

k += 1

previousCheckSum = checkSum

i = len(result)

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

i -= 1

f = file(sys.argv[2], 'wb+')

for c in result[:i]:

f.write(c)

f.close()

print '%i load records processed and %i bytes written to %s' % (k, i,

sys.argv[2])

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

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

U002A.5\_Z80\_SYS1.raw.gz U012.1\_SYS\_2.raw.gz ucsd ucsd.dsk

**Step 1**: Get U002A.5\_Z80\_SYS1.raw.gz and U012.1\_SYS\_2.raw.gz from the distribution and gunzip “gunzip \*gz”.

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

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

change

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

to

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

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

before

Current Track Current Sector Current Block Current Drive

0005 000C 000B A:

Offset 0 1 2 3 4 5 6 7 8 9 A B C D E F -----ASCII------

000000 09 29 29 EB 01 36 00 2A 94 02 19 09 C9 E1 22 90 |.))k.6.\*....Ia".|

000010 02 E1 22 92 02 D1 EB 22 94 02 EB 2A 90 02 06 08 |.a"..Qk"..k\*....|

000020 1A BE C2 BA 1A 23 13 10 F7 21 00 00 E5 2A 94 02 |.>B:.#..w!..e\*..|

000030 EB 2A 92 02 73 23 72 C3 A4 03 D2 D3 1A 2A 94 02 |k\*..s#rC$.RS.\*..|

000040 11 08 00 19 5E 23 56 7B 3D B2 C2 96 1A 21 01 00 |....^#V{=2B..!..|

000050 C3 AC 1A 2A 94 02 11 0A 00 19 5E 23 56 7B 3D B2 |C,.\*......^#V{=2|

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

000070 EB 73 23 72 D1 2A 1C 03 EB 73 23 72 C3 B0 03 07 |ks#rQ\*..ks#rC0..|

after

Current Track Current Sector Current Block Current Drive

0005 000C 000B A:

Offset 0 1 2 3 4 5 6 7 8 9 A B C D E F -----ASCII------

000000 09 29 29 EB 01 36 00 2A 94 02 19 09 C9 E1 22 90 |.))k.6.\*....Ia".|

000010 02 E1 22 92 02 D1 EB 22 94 02 EB 2A 90 02 06 08 |.a"..Qk"..k\*....|

000020 1A BE C2 BA 1A 23 13 10 F7 21 00 00 E5 2A 94 02 |.>B:.#..w!..e\*..|

000030 EB 2A 92 02 73 23 72 C3 A4 03 D2 D3 1A 2A 94 02 |k\*..s#rC$.RS.\*..|

000040 11 08 00 19 5E 23 56 7B 3D B2 C2 96 1A 21 01 00 |....^#V{=2B..!..|

000050 C3 AC 1A 2A 94 02 11 0A 00 19 5E 23 56 7B 3D B2 |C,.\*......^#V{=2|

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

000070 EB 73 23 72 D1 2A 1C 03 EB 73 23 72 C3 B0 03 07 |ks#rQ\*..ks#rC0..|

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

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

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

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

Execute what file? setup

INITIALIZING............................

...........................

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

C

CHANGE: S(INGLE) P(ROMPTED) R(ADIX)

H(ELP) Q(UIT)

P

FIELD NAME = **BACKSPACE**

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

10 8 8 BS ^H

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **EDITOR ACCEPT KEY**

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

0 0 0 NUL ^@

**Y** WANT TO CHANGE THIS VALUE? (Y,N,!)

**26** NEW VALUE: 26

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

32 26 1A SUB ^Z

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **EDITOR ESCAPE KEY**

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

33 27 1B ESC ^[

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **ERASE LINE**

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

0 0 0 NUL ^@

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **ERASE SCREEN**

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

0 0 0 NUL ^@

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **ERASE TO END OF LINE**

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

0 0 0 NUL ^@

**Y** WANT TO CHANGE THIS VALUE? (Y,N,!)

**75** NEW VALUE: 75

OCTAL DECIMAL HEXADECIMAL ASCII

113 75 4B K

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **ERASE TO END OF SCREEN**

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

0 0 0 NUL ^@

**Y** WANT TO CHANGE THIS VALUE? (Y,N,!)

**74** NEW VALUE: 74

OCTAL DECIMAL HEXADECIMAL ASCII

112 74 4A J

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **HAS 8510A**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **HAS BYTE FLIPPED MACHINE**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **HAS CLOCK**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **HAS LOWER CASE**

CURRENT VALUE IS FALSE

**Y** WANT TO CHANGE THIS VALUE? (Y,N,!)

**T** NEW VALUE: T

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **HAS RANDOM CURSOR ADDRESSING**

CURRENT VALUE IS FALSE

**Y** WANT TO CHANGE THIS VALUE? (Y,N,!)

**T** NEW VALUE: T

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **HAS SLOW TERMINAL**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **HAS WORD ORIENTED MACHINE**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **KEY FOR BREAK**

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

0 0 0 NUL ^@

**Y** WANT TO CHANGE THIS VALUE? (Y,N,!)

**3** NEW VALUE: 3

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

3 3 3 ETX ^C

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **KEY FOR FLUSH**

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

6 6 6 ACK ^F

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **KEY FOR STOP**

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

23 19 13 DC3 ^S

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **KEY TO DELETE CHARACTER**

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

10 8 8 BS ^H

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **KEY TO DELETE LINE**

OCTAL DECIMAL HEXADECIMAL ASCII

177 127 7F DEL

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **KEY TO END FILE**

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

3 3 3 ETX ^C

**Y** WANT TO CHANGE THIS VALUE? (Y,N,!)

**26** NEW VALUE: 26

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

32 26 1A SUB ^Z

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **KEY TO MOVE CURSOR DOWN**

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

12 10 A LF ^J

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **KEY TO MOVE CURSOR LEFT**

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

10 8 8 BS ^H

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **KEY TO MOVE CURSOR RIGHT**

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

34 28 1C FS ^\

**Y** WANT TO CHANGE THIS VALUE? (Y,N,!)

**12** NEW VALUE: 12

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

14 12 C FF ^L

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **KEY TO MOVE CURSOR UP**

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

37 31 1F US ^\_

**Y** WANT TO CHANGE THIS VALUE? (Y,N,!)

**11** NEW VALUE: 11

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

13 11 B VT ^K

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **LEAD IN FROM KEYBOARD**

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

0 0 0 NUL ^@

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **LEAD IN TO SCREEN**

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

0 0 0 NUL ^@

**Y** WANT TO CHANGE THIS VALUE? (Y,N,!)

**27** NEW VALUE: 27

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

33 27 1B ESC ^[

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **MOVE CURSOR HOME**

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

15 13 D CR ^M

**Y** WANT TO CHANGE THIS VALUE? (Y,N,!)

**72** NEW VALUE: 72

OCTAL DECIMAL HEXADECIMAL ASCII

110 72 48 H

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **MOVE CURSOR RIGHT**

OCTAL DECIMAL HEXADECIMAL ASCII

41 33 21 !

**Y** WANT TO CHANGE THIS VALUE? (Y,N,!)

**68** NEW VALUE: 68

OCTAL DECIMAL HEXADECIMAL ASCII

104 68 44 D

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **MOVE CURSOR UP**

OCTAL DECIMAL HEXADECIMAL ASCII CONTROL

0 0 0 NUL ^@

**Y** WANT TO CHANGE THIS VALUE? (Y,N,!)

**65** NEW VALUE: 65

OCTAL DECIMAL HEXADECIMAL ASCII

101 65 41 A

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **NON PRINTING CHARACTER**

OCTAL DECIMAL HEXADECIMAL ASCII

77 63 3F ?

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **PREFIXED[DELETE CHARACTER]**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **PREFIXED[EDITOR ACCEPT KEY]**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **PREFIXED[EDITOR ESCAPE KEY]**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **PREFIXED[ERASE LINE]**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **PREFIXED[ERASE SCREEN]**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **PREFIXED[ERASE TO END OF LINE]**

CURRENT VALUE IS FALSE

**Y** WANT TO CHANGE THIS VALUE? (Y,N,!)

**T** NEW VALUE: T

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **PREFIXED[ERASE TO END OF SCREEN]**

CURRENT VALUE IS FALSE

**Y** WANT TO CHANGE THIS VALUE? (Y,N,!)

**T** NEW VALUE: T

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **PREFIXED[KEY FOR BREAK]**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **PREFIXED[KEY FOR FLUSH]**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **PREFIXED[KEY FOR STOP]**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **PREFIXED[KEY TO DELETE CHARACTER]**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **PREFIXED[KEY TO DELETE LINE]**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **PREFIXED[KEY TO END FILE]**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **PREFIXED[KEY TO MOVE CURSOR DOWN]**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **PREFIXED[KEY TO MOVE CURSOR LEFT]**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **PREFIXED[KEY TO MOVE CURSOR RIGHT]**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **PREFIXED[KEY TO MOVE CURSOR UP]**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **PREFIXED[MOVE CURSOR HOME]**

CURRENT VALUE IS FALSE

**Y** WANT TO CHANGE THIS VALUE? (Y,N,!)

**T** NEW VALUE: T

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **PREFIXED[MOVE CURSOR RIGHT]**

CURRENT VALUE IS FALSE

**Y** WANT TO CHANGE THIS VALUE? (Y,N,!)

**T** NEW VALUE: T

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **PREFIXED[MOVE CURSOR UP]**

CURRENT VALUE IS FALSE

**Y** WANT TO CHANGE THIS VALUE? (Y,N,!)

**T** NEW VALUE: T

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **PREFIXED[NON PRINTING CHARACTER]**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **SCREEN HEIGHT**

OCTAL DECIMAL HEXADECIMAL

30 24 18

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **SCREEN WIDTH**

OCTAL DECIMAL HEXADECIMAL

120 80 50

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **STUDENT**

CURRENT VALUE IS FALSE

N WANT TO CHANGE THIS VALUE? (Y,N,!)

FIELD NAME = **VERTICAL MOVE DELAY**

OCTAL DECIMAL HEXADECIMAL

5 5 5

**Y** WANT TO CHANGE THIS VALUE? (Y,N,!)

**0** NEW VALUE: 0

OCTAL DECIMAL HEXADECIMAL

0 0 0

N WANT TO CHANGE THIS VALUE? (Y,N,!)

CHANGE: S(INGLE) P(ROMPTED) R(ADIX)

Q H(ELP) Q(UIT)

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

D QUIT: D(ISK) OR M(EMORY) UPDATE,

R(ETURN) H(ELP) E(XIT)

M QUIT: D(ISK) OR M(EMORY) UPDATE,

R(ETURN) H(ELP) E(XIT)

E QUIT: D(ISK) OR M(EMORY) UPDATE,

R(ETURN) H(ELP) E(XIT)

**Step 4**: Rename NEW.MISCINFO to SYSTEM.MISCINFO

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

F

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

L

Dir listing of what vol ? \*

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

U002A.5:

SYSTEM.STARTUP 5 28-Feb-79

SYSTEM.MICRO 16 9-Feb-79

Z80T.MICRO 19 9-Feb-79

SYSTEM.FILER 28 10-Apr-79

SYSTEM.PASCAL 33 7-Mar-79

SYSTEM.EDITOR 45 10-Feb-79

SYSTEM.LINKER 22 10-Feb-79

SYSTEM.COMPILER 68 8-Feb-79

SYSTEM.LIBRARY 8 17-Apr-79

SYSTEM.SYNTAX 14 2-May-79

SAMPLEGOTO.TEXT 4 17-Nov-78

SETUP.CODE 25 14-May-79

READ.ME.TEXT 4 17-Apr-79

BINDER.CODE 6 3-May-79

NEW.MISCINFO 1 10-Feb-79

15/15 files<listed/in-dir>, 308 blocks used, 186 unused, 186 in largest

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

C

Change what file ? NEW.MISCINFO

Change to what ? SYSTEM.MISCINFO

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

**Step 6**: Set date with D command in Filer

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

(\*$U-\*)

PROGRAM DUMMY;

(\* Direct cursor addressing for VT100 terminal \*)

(\* '[' after escape is done by BIOS - trick from Udo Munk \*)

PROCEDURE FGOTOXY(X,Y:INTEGER);

BEGIN

IF X<0 THEN X:=0;

IF X>79 THEN X:=79;

IF Y<0 THEN Y:=0;

IF Y>23 THEN Y:=23;

WRITE (CHR(27),Y+1,';',X+1,'H')

END;

BEGIN

END.

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

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

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

# Vector Graphic, Inc. Simulation

Vector Graphic support was added by Howard M. Harte.

## Overview

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

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

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

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

History and Background

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

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

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

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

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

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

Mailing List

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

Documentation

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

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

Documentation / Disk Images

[http://vector-archive.org](http://vector-archive.org/)

[FluxEngine: Read hard-sectored floppy disks](https://github.com/davidgiven/fluxengine/blob/master/doc/disk-micropolis.md)

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:

## 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**.

## 56K Vector MZ

sim> **load MON40C.BIN e000** ; load Vector 4.0C Monitor

sim> **set mdsk enabled** ; enable Micropolis disk controller

sim> **attach mfdc0 VG00.VGI** ; attach disk to MDSK0 drive

When booting the 56K configuration, type:

sim> g e000

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

## 56K Vector with HD-FD Controller

sim> **set vfdhd enabled** ; enable HD-FD controller

sim> **load MON43B.BIN e000** ; load Vector 4.3 Monitor

sim> **att fwii0 f000** ; enable the Flashwriter2 at F000.

sim> **set telnet 23** ; set up telnet port for Flashwriter2

sim> **attach vfdhd1 VGBOOT.VGI** ; attach disk to VFDHD1 drive

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

sim> g e000

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

## 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.

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

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

The 275-byte sector format is as follows:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SYNC | TRACK | SECTOR | UNUSED | USER DATA | CHKSUM | ECC | ECC\_VALID |
| 1 | 1 | 1 | 10 | 256 | 1 | 4 | 1 |

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

# IMSAI 8080 Simulation

IMSAI FIF Disk Controller support was added by Ernie Price.

## Overview

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

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

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

Additional devices include:

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

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

sim> **set hdsk dis** ; disable the AltairZ80 HDSK device.

sim> **set fif ena** ; enable the IMSAI FIF device.

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

**imdos**  IMSAI 8080 with FIF Disk Controller

## 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.

# NorthStar MDS-A and MDS-AD FDC Simulation

NorthStar MDS-A (single density FDC) support was added by Mike Douglas, [deramp5113@yahoo.com](mailto:deramp5113@yahoo.com).

NorthStar MDS-AD (double density FDC) support was added by Howard M. Harte.

## Overview

The single density MDS-A disk controller was the first disk controller made by NorthStar. This controller supports 48 TPI 5.25” disks with 35 tracks and 10 hard sectors per track. Sectors are 256 bytes in length for a total storage capacity of 89,600 bytes.

Later, NorthStar introduced the double sided, double density, MDS-AD disk controller. This controller also uses 48 TPI 5.25” media with 35 tracks and 10 hard-sectors per track, but with 512 bytes per sector, disk capacity increased to 179,200 bytes on a single sided drive, or 358,400 bytes on a double sided drive.

The first computer offered by NorthStar, the Horizon, used the double density controller. However, both the single density and double density controllers were a very popular choice in Altair, IMSAI, Sol-20, and numerous other early microcomputers.

## MDS-A Single Density Disk Controller

The single density controller device is **MDSA**. The NorthStar controller is a memory mapped device in the range 0xE800-0xEBFF. The boot PROM is at 0xE900 on the single density controller. NorthStar documentation refers to drives as 1-3, however, the drive numbers are 0-2 in the simulation.

Following are the commands to mount and boot CP/M for the NorthStar Horizon computer.

sim> **set mdsa enabled**  ; enable MDS-A Controller

sim> **attach mdsa0 CPM22b14-48K-SDC-HORIZON.NSI** ; attach CP/M boot disk

sim> **boot mdsa0** (or **go e900**) ; boot the disk

The referenced disk image can be found in the “cpm” folder at the link below. NorthStar DOS disk images can be found in the “nsdos” folder.

<https://deramp.com/downloads/north_star/horizon/single_density_controller/disk_images/>

Disk images for the NorthStar single density controller are available for a variety of different computers. Drill down the folder tree of the computer of interest at <https://deramp.com/downloads/>.

## MDS-AD Double Density Disk Controller

The double density controller device is **MDSAD**. This controller supports both single- and double-density media. The NorthStar controller is a memory mapped device in the range 0xE800-0xEBFF. The boot PROM is at 0xE800 on the double density controller. NorthStar documentation refers to drives as 1-4, however, the drive numbers are 0-3 in the simulation.

Following are the commands to mount and boot CP/M for the NorthStar Horizon computer.

sim> **set mdsad enable**  ; enable NorthStar MDS-AD Controller

sim> **attach mdsad0 D01B01.NSI** ; attach CP/M boot disk to MDSAD0 drive

sim> **boot mdsad0**(or **go e800**) ; boot the disk

There is a configuration file that configures SIMH to simulate a NorthStar 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 preceding description if there is any discrepancy. This configuration file is:

**nshrz**  NorthStar Horizon with MDS-AD Disk Controller

Additional Horizon disk images can be found in the “cpm” folder at the link below. Additional NorthStar DOS disk images can be found in the “nsdos” folder.

<https://deramp.com/downloads/north_star/horizon/double_density_controller/disk_images/>

Disk images for the NorthStar double density controller are available for a variety of different computers. Drill down the folder tree of the computer of interest at <https://deramp.com/downloads/>.

# Compupro 8-16 Simulation

Compupro Controller support was added by Howard M. Harte. The 8086 simulation was added by Peter Schorn.

## 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)

## 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.

### DISK1A Controller Parameters

The DISK1A controller supports several parameters which can be configured in 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.

### 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.

## 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.

### DISK2 Controller Parameters

The DISK2 controller supports several parameters which can be configured in 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.

### DISK2 Controller Configuration Registers

The DISK2 controller has several configuration registers that can be examined and deposited in 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.

## SELCHAN Compupro Selector Channel Controller

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

### DISK2 Controller Parameters

The DISK2 controller supports several parameters which can be configured in 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.

## 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.

### DISK3 Controller Parameters

The DISK3 controller supports several parameters which can be configured in 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.

### DISK3 Controller Configuration Registers

The DISK3 controller has several configuration registers that can be examined and deposited in 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.

### DISK3 Controller Limitations

The DISK3 controller has been tested with the Compupro DISK3.COM diagnostic utility, but has not been tested with CP/M.

# Compupro CPU-68K Simulation

Compupro CPU-68K support was added by Howard M. Harte

## Overview

The [Compupro CPU-68K](http://www.bitsavers.org/pdf/compupro/CPU-68K/CPU-68K_S-100_198311.pdf) is a Motorola 68000-based CPU board. Adding support for this board involved updating AltairZ80’s 68000 CPU emulation to the latest upstream [Musashi codebase](https://github.com/kstenerud/Musashi), and adding a memory-mapped I/O window to allow S-100 peripherals to be accessible via the CPU.

No additional devices were added to support this simulation.

In November, 2022, [CompuPro CP/M-68K 1.1I System Master disks](https://bitsavers.org/bits/Compupro/CompuPro_68K_System_Masters.zip) were imaged from the original 8” floppies. CP/M-68K 1.3I System Master disks were created from [Digital Research source code](http://cpm.z80.de/download/68kv1_3.zip). The following CP/M-68K packages containing AltairZ80 configuration files are available:

* [CompuPro CP/M-68K 1.1I](https://github.com/hharte/altairz80-packages/tree/main/ccpm68k11)
* [CompuPro CP/M-68K 1.3I](https://github.com/hharte/altairz80-packages/tree/main/ccpm68k13)

## M68K Registers Added

MMIOBASE 24 Memory Mapped I/O Base Address

MMIOSIZE 17 Memory Mapped I/O Size

MMIOBASE defaults to 0xFF0000, with MMIOSIZE default 0x10000, providing an I/O Window of 64K I/O ports at the top of the M68K’s 16MB address space. The Cromemco 68000 CPU card has a 256-byte I/O window from 0xFFFF00-0xFFFFFF which can be set with:

sim> d MMIOBASE FFFF00

sim> d MMIOSIZE 100

## M68K Additional Parameters

The M68K simulation supports several CPU variants which can be set as follows:

set CPU <variant> - Where variant is one of:

* 68000
* 68010
* 68020
* 68EC20
* 68030
* 68EC30
* 68040
* 68EC040
* 68LC040
* SCC68070

# Cromemco 4/16/64FDC and CCS-2422 FDC Simulation

Cromemco 4/16/64FDC (CROMFDC) Floppy Controller support was added by Howard M. Harte.

## 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.

### CROMFDC Controller Parameters

The CROMFDC controller supports several parameters which can be configured in 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.

### CROMFDC Controller Configuration Registers

The CROMFDC controller has several configuration registers that can be examined and deposited in 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.

### 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.

# Seattle Computer Products Simulation

Seattle Computer Products SCP-300F, TDD, and DJHDC support were added by Howard M. Harte.

## Overview

The Seattle Computer Products line of products with the 8086 CPU were used for development of 86-DOS, which was later sold to Microsoft and re-branded as MS-DOS. AltairZ80 provides enough simulated hardware to run 86-DOS as well as MS-DOS 1.x and [MS-DOS 2.11](https://github.com/hharte/altairz80-packages/tree/main/msdos211_test) ([Source code](https://github.com/hharte/MS-DOS/tree/main/v2.0/source).) Seattle Computer Products supported various floppy disk controllers, so there are a few 86-DOS disk images around that can use the Cromemco 16FDC, while most support the Tarbell Double-Density controller. Seattle Computer Products also supported the NorthStar MDS-AD disk controller, although the SCP300F support board does not include the monitor ROM for this disk controller. In 2022, some [NorthStar disk images for SCP](http://bitsavers.org/bits/SeattleComputerProducts/NSI/) were made available, so those would be interesting to get working in simulation.

Additional devices include:

**SCP300F** – Seattle Computer Products CPU Support Board.

**TDD** – Tarbell Double-Density disk controller.

**DJHDC** – Morrow Disk Jockey/HDC Hard Disk Controller

## SCP300F CPU Support Board

The SCP-300F CPU Support Board provides an array of peripherals to run MS-DOS.

### SCP300F Parameters

The SCP300F controller supports several parameters which can be configured in the simulator:

ROM=[TARBELL | CROMEMCO] – Select the SCP Monitor for either the Tarbell (default) or Cromemco floppy controller.

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.
* PIC – 8259A Priority Interrupt Controller messaging
* TIMER – 9513 Real-time clock messaging
* ROM – ROM enable / disable messaging
* PIO – Parallel I/O port messaging
* UART – UART messaging
* IRQ – Interrupt debugging
* SSW – Sense Switch messaging
* VERBOSE – Extra verbosity for debugging

NODEBUG – Turn off one or more debug message levels.

### SCP300F Configuration Registers

The SCP300F controller has several configuration registers that can be examined and deposited in the simulator. These registers are:

* MPIC\_\* – Master 8259A Interrupt Controller registers.
* SPIC\_\* – Slave 8259A Interrupt Controller registers.
* 9513\_\* – 9513 RTC registers.

### SCP300F Limitations

Functionality not required to support 86-DOS, MS-DOS 1.25, 2.00, 2.11 may not be fully implemented and/or exercised.

## TDD – Tarbell Double-Density Controller

The TDD disk controller is a wrapper around AltairZ80’s wd179x simulation that adds Tarbell-specific functionality. The TDD supports up to four drives, with disks in ImageDisk format as well as raw sector format.

### TDD Parameters

The SCP300F controller supports several parameters which can be configured in 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.
* DRIVE – Drive select and control messaging
* IRQ – Interrupt debugging
* VERBOSE – Extra verbosity for debugging

NODEBUG – Turn off one or more debug message levels.

### TDD Configuration Registers

The TDD controller does not currently contain any configuration registers.

### TDD Limitations

Functionality not required to support 86-DOS, MS-DOS 1.25, 2.00, 2.11 may not be fully implemented and/or exercised.

## DJHDC – Morrow HDC / DMA Hard Disk Controller

The [Morrow HDC / DMA Hard Disk Controller](http://www.bitsavers.org/pdf/morrow/boards/HDC_DMA_Technical_Manual_1983.pdf) (DJHDC) hard disk controller supports up to four ST-506 8” or 5.25” hard disk drives. Nearly all communication with this controller is through a memory mapped parameter block read by the controller via DMA. Only a single, write-only I/O port is used to reset the controller or initiate operation of the controller. Architecturally, this controller is very similar to the CompuPro DISK3 controller.

### DJHDC Controller Parameters

The DJHDC controller supports several parameters which can be configured in the simulator. Defaults are set for a 15-megabyte Miniscribe drive commonly used with SCP MS-DOS:

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
* OPCODE – Disk controller operation codes
* READ – Read messaging
* WRITE – Write messaging
* IRQ – Interrupt messaging
* FORMAT – Format-related messaging
* VERBOSE – Extra verbosity for debugging

NODEBUG – Turn off one or more debug message levels.

GEOMETRY=val – Set disk geometry C:nnnn/H:n/S:nnn/N:nnnn, where:

* C:nnnn – Number of Cylinders (1-1024)
* H:n – Number of Heads (1-8)
* S:nnn – Number of Sectors per cylinder (1-256)
* N:nnnn – Sector size (128, 256, 512, 1024, 2048)

### DJHDC Controller Configuration Registers

The DJHDC controller has several configuration registers that can be examined and deposited in the simulator. These registers are:

SEL\_DRIVE – Currently selected disk drive.

LINK\_ADDR – Link address for next IOPB.

DMA\_ADDR – DMA address for the current IOPB.

IOPB[0-15] – Contents of the current IOPB.

The DJHDC supports four drives, labeled DJHDC0 through DJHDC3. 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.

### DJHDC Controller Limitations

The DJHDC controller has only been tested with the MS-DOS 2.11.

# Tarbell MDL-1011/2022 Floppy Disk Interface Simulation

Tarbell Electronics MDL-1011/2022 Floppy Disk Interface support was added by Patrick A. Linstruth, [patrick@deltecent.com](mailto:patrick@deltecent.com).

## Overview

The Tarbell MDL-1011 floppy disk interface is a single-sided, single-density disk controller supporting 8” media with 77 tracks, 26 sectors per track, with 128 byte sectors.

The Tarbell MDL-2022 floppy disk interface is a double-sided, double-density disk controller supporting 8” SSSD, SSDD, DSSD, and DSDD media with 77/154 tracks and 26/51 sectors per track, with 128 byte sectors.

Both controller models include a 32-byte PROM bootstrap loader located at 0000H that is automatically enabled when the simulator is reset and switches itself out after the bootstrap has run.

Using the Tarbell MDL-1011/2022 floppy disk interfaces, it is possible to run CP/M and other operating systems that are designed for these interfaces on the simulator.

### TARBELL Controller Parameters

The TARBELL controller supports several parameters which can be configured in the simulator:

MODEL – Select TARBELL controller model:

* SD – MDL-1011 single density controller (default)
* DD – MDL-2022 double density controller

WRTPROT – Enable write protect (emulates switch 6 on).

WRTENB – Enable writing to disks (emulates switch 6 off).

PROM – Enable boot PROM at 0000-001Fh (emulates switch 7 on).

NOPROM – Disable boot PROM (emulated switch 7 off).

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 TARBELL controller supports four drives, labeled TARBELL0 through TARBELL3. If a drive is attached to a non-existent disk file, the disk file will be created.

### TARBELL Example Usage

Download CPM22-48K-SSSD.DSK via [https://deramp.com/downloads/altair/software/tarbell\_floppy\_controllers/single\_density\_controller/CPM%202.2%20(1982)/CPM22-48K-SSSD.DSK for the Tarbell MDL-1011](https://deramp.com/downloads/altair/software/tarbell_floppy_controllers/single_density_controller/CPM%202.2%20(1982)/CPM22-48K-SSSD.DSK%20for%20the%20Tarbell%20MDL-1011) and then

sim> **set cpu 48k**

sim> **set cpu 8080**

sim> **set cpu noaltairrom**

sim> **set sio ansi**

sim> **set sio sleep**

sim> **set tarbell enable** ; enable Tarbell Controller

sim> **set tarbell model=sd**  ; single density controller

sim> **set tarbell debug=ERROR** ; show debug ERROR messages

sim> **attach tarbell0 CPM22-48K-SSSD.DSK** ; attach CP/M boot disk to TARBELL0 drive

sim> **boot tarbell0** ; boot CPM22.DSK (or “g 0”)it

CP/M disk images supporting the Tarbell MDL-1011 floppy disk interface are available at <https://deramp.com/downloads/altair/software/tarbell_floppy_controllers/single_density_controller/>.

### TARBELL Controller Limitations

The TARBELL controller does not currently support the WD17XX Type II command multibyte record flag (‘m’ bit 4), block length flag (‘b’ bit 3), or data address mark flag (‘a1a0’ bits 1-0), the Read Track Type III command, or the Type IV command interrupt condition flags (‘Ii’ bits 3-0). This functionality will be added at a later date.

DMA is not currently supported by the TARBELL simulator.

# JADE Double D Floppy Disk Controller Simulation

JADE Double D Floppy Disk Controller support was added by Patrick A. Linstruth, [patrick@deltecent.com](mailto:patrick@deltecent.com).

## Overview

The JADE Double D Floppy Disk Controller simulator is capable of supporting single density and double density 8” media with 77 tracks and 128 byte sectors. Single and double density diskettes may be mixed on a drive-by-drive basis. Single density disks have 26 sectors per track. Double density have 50. The simulated controller includes a PROM bootstrap loader located at F000H. The PROM has been patched to utilize the MITS 88-2SIO. Using the simulated Jade Double D floppy disk controller, it is possible to load and run CP/M 2.2 as was provided by JADE Computer Products.

### JADE Double D Controller Parameters

The JADEDD controller supports several parameters which can be configured in the simulator:

IOBASE – Change I/O port address.

MEMBASE – Change memory bank window base address.

WRTPROT – Enable write protect.

WRTENB – Enable writing to disks.

PROM – Enable boot PROM at F000.

NOPROM – Disable boot PROM.

VERBOSE – Enable operational status messages.

QUIET – Disable operational status messages.

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.
* CMD – Disk controller commands
* RDDATA – Read Data messaging
* WRDATA – Write Data messaging
* FORMAT – Format Track messaging
* VERBOSE – Extra verbosity for debugging

NODEBUG – Turn off one or more debug message levels.

The JADEDD controller supports four drive units, labeled JADEDD0 through JADEDD3. If a drive is attached to a non-existent disk file, the disk file will be created.

### JADEDD Example Usage

sim> **set jadedd ena**  ; enable JADE DD Controller

sim> **attach jadedd0 jadedd-sd-sim-56k.dsk** ; attach 56K CP/M disk image

sim> **boot jadedd0**  ; boot jade56k.dsk (or “g f000”)

CP/M 2.2 disk images supporting the JADE Double D Disk Controller and 88-2SIO are available at: <https://github.com/deltecent/jadedd-cpm22>

### JADE Double D Controller Limitations

The JADE Double D contains an on-board Z80A microprocessor with 2K of static memory. The on-board processor runs simultaneously with and transparent to the Altair’s 8080 or Z80 processor and memory. The Double D’s Z80A and on-board Disk Controller Module (DCM) software are only simulated. Although the simulator, like the actual hardware, loads the DCM from disk into memory bank 0 during the boot process (and can be examined/altered by the host CPU), the DCM code does not execute within the simulator. Changes to the DCM in RAM or on disk will not change how the emulator operates. This is also true for the Boot Loader Transient (BLT) module and FORMAT.COM.

The JADE Double D simulator is specifically designed to run CP/M 2.2 as distributed by JADE Computer Products as described in the “CP/M 2.2 – Double D – Release 2” software manual. Other versions of CP/M or operating systems designed to operate with the JADE Double D will likely not work with this simulator.

When changing memory size, MOVCPM updates the BLT at track 0, sector 2, to match the new location of DDBIOS. Since the simulator emulates the BLT, the boot PROM built into the simulator determines this location by looking at the jump vector table at the beginning of DDBIOS (JMP INIT). If DDBIOS is modified such that INIT does not appear within the first 256 bytes of DDBIOS, this mechanism will not work, and CP/M will fail to load.

The JADE Double D Disk Controller emulator does not currently support double-sided drives.

The Serial Interface on the Double D is not supported.

# iCOM FD3712/FD3812 Flexible Disk System Simulation

iCOM FD3712/FD3812 support was added by Patrick A. Linstruth, patrick@deltecent.com.

## Overview

The iCOM FD3712/FD3812 Flexible Disk System simulator is capable of supporting single density and double density 8” media with 77 tracks and 128/256 byte sectors. Single density disks have 26 128 byte sectors per track. Double density disks have 26 256 byte sectors per track. Track 0 of double density disks are always formatted single density. The simulated controller includes 3712 and 3812 PROM bootstrap loaders located at F000H. Using the simulated iCOM FD3712/FD3812 Flexible Disk System, it is possible to load and run CP/M 1.41 as was provided by Lifeboat Associates and other compatible operating systems.

### iCOM FD3712/FD3812 Flexible Disk System Parameters

The iCOM controller supports several parameters which can be configured in the simulator:

TYPE – Select iCOM interface board type:

* 3712 – FD3712 single density interface board
* 3812 – FD3812 double density interface board (default)

IOBASE – Change I/O port address.

MEMBASE – Change interface board 256K memory address.

WRTPROT – Enable write protect.

WRTENB – Enable writing to disks.

PROM – Enable/Disable PROM.

* ENABLE – Enable boot PROM at F000 (default).
* DISABLE – Disable boot PROM.

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:

* VERBOSE – Operational status messages.
* 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.
* CMD – Disk controller commands
* STATUS – Status information messages
* RDDATA – Read Data messages
* RDDETAIL – Read Sector messages
* WRDATA – Write Data messages
* WRDETAIL – Write Sector messages

NODEBUG – Turn off one or more debug message levels.

The iCOM controller supports four drive units, labeled ICOM0 through ICOM3. If a drive is attached to a non-existent disk file, the disk file will be created.

### iCOM Example Usage

sim> **set icom ena**  ; enable iCOM Disk System

sim> **set icom type=3812**  ; simulate iCOM/Pertec 3812

sim> **attach icom0 CPM141-48K-DD.DSK** ; attach iCOM CP/M disk image

sim> **boot icom**  ; boot disk (or “g f000”)

CP/M 1.41 disk images supporting the iCOM FD3712/FD3812 Flexible Disk System and 88-2SIO are available at: <https://github.com/deltecent/icom-fds> and <https://deramp.com/downloads/altair/software/iCOM%20FD3712%20FD3812/>

# Morrow DISK JOCKEY 2D Disk Controller Simulation

Morrow DISK JOCKEY 2D disk controller support was added by Patrick A. Linstruth, patrick@deltecent.com.

## Overview

The Morrow DISK JOCKEY 2D (DJ2D) disk controller simulator is capable of supporting single density and double density soft sector 8” media with 77 tracks and 128/256/512/1024 byte sectors. Single density disks have 26 128-byte sectors per track. Double density disks have 26 256-byte, 15 512-byte, or 8 1024-byte sectors per track. Track 0 of double density disks are always formatted single density. The standard format for CP/M is 1024-byte sectors. The simulated controller include 1K PROMs for addresses E000 and F800. The PROM is followed by 1024 bytes of RAM. The DJ2D provides for 8 memory-mapped I/O addresses starting at location PROMBASE+03F8. Using the simulated DJ2D, it is possible to load and run CP/M 1.4 and 2.2 as was provided by Thinker Toys, DISK/ATE, and other compatible operating systems.

### DJ2D Parameters

The DJ2D supports several parameters which can be configured in the simulator:

MODEL – Select DJ2D controller model:

* A – DJ2D
* B – DJ2D Model B (default)

SIDES – Set drive as single or double sided:

* 1 – Single sided drive (default)
* 2 – Double sided drive

PROMBASE – Change PROM starting address (E000 or F800).

WRTPROT – Enable write protect.

WRTENB – Enable writing to disks.

PROM – Enable/Disable PROM.

* ENABLE – Enable PROM (default).
* DISABLE – Disable PROM.

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:

* VERBOSE – Verbose status messages
* DEBUG – Debugging messages
* 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.
* CMD – Disk controller commands
* STATUS – Status information messages
* RDDATA – Read Data messages
* RDDETAIL – Read Sector messages
* WRDATA – Write Data messages
* WRDETAIL – Write Sector messages

NODEBUG – Turn off one or more debug message levels.

The DJ2D controller supports four drive units, labeled DJ2D0 through DJ2D3 and one serial interface labeled DJ2D4. If a drive is attached to a non-existent disk file, the disk file will be created.

### DJ2D Example Usage

sim> **;Boot 56K CP/M 2.2**

sim> **set dj2d ena**  ; enable DJ2D disk controller

sim> **attach dj2d0 CPM22-56K-E000.DSK** ; attach DJ2D CP/M disk image

sim> **boot dj2d**  ; boot disk (or “g e000”)

sim> **;Boot 56K CP/M 2.2 with serial port attached to socket**

sim> **set dj2d ena**  ; enable DJ2D disk controller

sim> **attach dj2d0 CPM22-56K-E000.DSK** ; attach DJ2D CP/M disk image

sim> **attach -u dj2d4 127.0.0.1:8800** ; attach DJ2D serial port to socket

sim> **boot dj2d**  ; boot disk (or “g e000”)

sim> **;telnet localhost 8800** ; telnet to simulator on port 8800

CP/M 2.2 disk images supporting the DJ2D simulator are available at <https://github.com/deltecent/dj2d-cpm22> (GitHub).

### DJ2D Simulator Limitations

The DJ2D devices detects the format of the DSK image by its size.

Bank select logic is not currently supported.

The Western Digital FD1791 Read Track command is not currently supported.

# Advanced Digital Corporation Simulation

ADC Super-Six SBC and HDC-1001 support were added by Howard M. Harte.

## ADC Super-Six Single-Board Computer

### ADCS6 SBC 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, WD179x-based floppy disk controller, two UARTs, a Counter/Timer, memory banking, parallel ports, and a DMA controller.

Two monitor ROMs are included:

ADC – The Advanced Digital Monitor v3.6

DIGITEX – The DIGITEX Monitor version 1.2.A

Additional devices include:

**ADCS6** – ADC Super-Six SBC.

### ADCS6 SBC Parameters

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

ROM=[ADC | DIGITEX] – Enable Monitor ROM at F000-F7FFh.

* ADC - The default, and can run Advanced Digital Corp [CP/M 2.2](https://github.com/hharte/altairz80-packages/tree/main/adc_cpm2) or [3.0](https://github.com/hharte/altairz80-packages/tree/main/adc_cpm3).
* DIGITEX - Can run DIGITEX [CP/M 2.2](https://github.com/hharte/altairz80-packages/tree/main/dig_cpm2) or [3.0](https://github.com/hharte/altairz80-packages/tree/main/dig_cpm3) ([source code](https://github.com/hharte/digitex/tree/main/ADC/cpm22/src)) as well as [OASIS v5.6](https://github.com/hharte/digitex/tree/main/sim).

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.
* DRIVE – Messaging specific to drive, i.e.: Motor on/off, side selection
* VERBOSE – Extra verbosity for debugging
* CTC – Messages related to the on-board Z80-CTC (Counter/Timer)
* DMA – Z80-DMA register debugging
* PIO – Z80-PIO register debugging
* BANK – Messages related to banked memory on the ADCS6
* UART – Messages related to the on-board UARTs

NODEBUG – Turn off one or more debug message levels.

### ADCS6 SBC Configuration Registers

The ADCS6 controller has several configuration registers that can be examined and deposited in the simulator. These registers are:

EXTADR – S-100 Extended Address bits A23:16

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

MCTRL0 – MCTRL0 Register

MCTRL1 – MCTRL1 Register

CTCxCCW – CTC (x=0-3) Channel Control Word Register

CTCxTC – CTC (x=0-3) Time Constant Register

CTCxCNT – CTC (x=0-3) Count Register

CTCVEC – CTC Interrupt Vector Word Register

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.

### ADCS6 SBC Limitations

The ADCS6 simulation does not support the Z80-DMA or parallel ports, other than by emitting debug messages.

## ADCHD – Advanced Digital HDC-1001 Hard Disk Controller

The Advanced Digital Corporation HDC-1001 Hard Disk controller supports up to four ST-506 8” or 5.25” hard disk drives. The programming interface is very similar to the modern ATA “task file” structure.

### ADCHD Controller Parameters

The ADCHD controller supports several parameters which can be configured in the simulator. Defaults are set for a Quantum Q2020 8” hard disk:

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 commands
* READ – Read messaging
* WRITE – Write messaging
* VERBOSE – Extra verbosity for debugging

NODEBUG – Turn off one or more debug message levels.

GEOMETRY=val – Set disk geometry C:nnnn/H:n/S:nnn/N:nnnn, where:

* C:nnnn – Number of Cylinders (1-1024)
* H:n – Number of Heads (1-8)
* S:nnn – Number of Sectors per cylinder (1-256)
* N:nnnn – Sector size (128, 256, 512)

### ADCHD Controller Configuration Registers

The ADCHD controller exposes the task file through configuration registers that can be examined and deposited in the simulator. These registers are:

TF\_\* – Task file registers.

The ADCHD supports four drives, labeled ADCHD0 through ADCHD3. 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.

### ADCHD Controller Limitations

The DJHDC controller has been tested with the CP/M 2.2, CP/M 3.0, and OASIS 5.6 operating systems. The DJHDC does not support interrupts.

.

# N8VEM Single Board Computer Simulation

N8VEM Single Board Computer support was added by Howard M. Harte.

## Overview

The N8VEM Single Board Computer is a homebrew Z80 system designed by Andrew Lynch. This SBC can has 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.

### N8VEM SBC Parameters

The N8VEM SBC supports several parameters which can be configured in 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.

### N8VEM SBC Configuration Registers

The N8VEM SBC has several configuration registers that can be examined and deposited in 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.

### 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.

# Morrow Micro Decision (MD) Computer Simulation

Morrow Micro Decision Computer support was added by Patrick A. Linstruth, [patrick@deltecent.com](mailto:patrick@deltecent.com).

## Overview

The Morrow Micro Decision was a single-board Z80 CP/M machine designed by George Morrow. The Micro Decision included a Z80 processor with 64K of memory, two serial ports, one parallel port, and up to four 5 ¼ inch double-density floppy disk drives or four 5 ¼ inch double-sided double-density floppy disk drives.

Additional devices include:

**MMD0** – Drive A.

**MMD1** – Drive B.

**MMD2** – Drive C.

**MMD3** – Drive D.

**MMD4** – Console Serial Port.

**MMDM** – Modem Serial Port.

Both MMD4 and MMDM serial ports fully support attaching sockets and real serial ports.

### Morrow MD Parameters

The Morrow MD supports several parameters which can be configured in the simulator:

DIAG – Set ROM Diagnostics enabled/disabled status.

ROM – Set ROM Version [13 | 23 | 24 | 25 | 31].

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.
* FDC – Floppy Disk Controller Command messages
* RBUF – Read Buffer messages
* WBUF – Write Buffer messages
* STATUS – Status messages
* VERBOSE – Extra verbosity for debugging

NODEBUG – Turn off one or more debug message levels.

### Morrow MD Configuration Registers

The Morrow MD has several configuration registers that can be examined and deposited in the simulator.

The MMD device registers are:

DRIVE – Current drive.

STATUS – FDC main status register.

CMD – FDC command register.

SECTOR – FDC sector register.

NUMBER – FDC number register.

HEAD – FDC head register.

ROM – ROM enabled.

DIAG – ROM diagnostics enabled.

BAUD – Console port baud rate.

STAT – Console port status register.

TXP – Console port TX pending register.

TXD – Console port TX register.

RXD – Console port RX register.

INTENA – Interrupt enable.

INTVEC – Mode 2 Interrupt Vector (0-7).

DATABUS – Mode 2 Interrupt Data Bus.

The MMDM device registers are:

BAUD – Console port baud rate.

STAT – Console port status register.

TXP – Console port TX pending register.

TXD – Console port TX register.

RXD – Console port RX register.

TICKS – Modem port timer ticks.

### Morrow MD Limitations

The Morrow MD parallel port is not currently supported.

The MDMM modem port device is disabled by default as it uses the same port address as SIMH. If this device is enabled, CP/M utilities such as R.COM and W.COM will not work.

### Morrow MD Example Usage

sim> **;Boot 64K CP/M 2.2**

sim> **set cpu z80**  ; Z80 required

sim> **set noaltairrom**  ; disable Altair ROM

sim> **set mmd ena**  ; enable MMD system

sim> **attach mmd0 MORR\_01\_CPM22R31\_DS.DSK** ; attach MMD CP/M disk image

sim> **boot mmd**  ; boot disk

CP/M 2.2 disk images supporting the Morrow Micro Decision simulator are available at <https://github.com/deltecent/morrow-md> (GitHub).

# Processor Technology Sol-20 Terminal Computer Simulation

The Processor Technology Sol-20 Terminal Computer and Video Display Module support was added by Patrick A. Linstruth, [patrick@deltecent.com](mailto:patrick@deltecent.com).

## Overview

The Sol-20, advertised as the first complete small computer, was a single-board 8080-based system designed by Processor Technology. The Sol-20 included a keyboard, audio cassette interface, both serial and parallel interfaces, an integrated VDM-1 video display module, and an S-100 expansion bus all housed in a case with walnut sides.

The Sol-20 simulator provides the following devices:

**SOL20** – Personality Module ROMs and General Status Register

**SOL20K** – Keyboard Interface

**SOL20P** – Printer Interface

**SOL20S** – Serial Interface

**SOL20T** – Cassette Tape Interface

**VDM1** – Video Display Module

The SOL20K, SOL20P and SOL20S devices fully support attaching sockets and real serial ports. The SOL20T device supports attaching virtual cassette tape images.

Enabling the SOL20 device in the simulator will automatically enable the other SOL20 and VDM1 devices.

### Sol-20 Parameters

The SOL20 device supports several parameters which can be configured in the simulator:

PORT – Set Sol-20 I/O General Status Register Port Address.

ROM – Set SOLOS ROM Memory Address.

VER – Set SOLOS ROM Version [13 | 13C | 41].

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.
* STATUS – Status messages

NODEBUG – Turn off one or more debug message levels.

### Sol-20 Configuration Registers

The Sol-20 has several configuration registers that can be examined and deposited in the simulator.

The SOL20K device registers are:

BAUD – Keyboard port baud rate

RDR – Keyboard receive data ready register

RXD – Keyboard receive data register

The SOL20P and SOL20S device registers are:

BAUD – Serial port baud rate

TBE – Serial port transmit pending register

TXD – Serial port transmit data register

RDR – Serial port receive data ready register

RXD – Serial port receive data register

### Sol-20 Keyboard

The Sol-20 simulator has special Sol-20 keyboard keys and other functions assigned to the F1 – F10 function keys:

F1 - Rewind tape (seek to start of virtual tape image)

F2 - Eject tape (detach virtual tape image)

F3 - Erase tape (truncate virtual tape image)

F4 - Toggle tape speed (normal and fast)

F5 - LOAD key

F6 - MODE SELECT key

F7 - CLEAR key

F8 - Toggle CAPS lock

F10 - Reboot System

### Sol-20 Example Usage

sim> **; Enter SOLOS**

sim> **set sol20 ena**  ; enable SOL20 device

sim> **set throttle 220k**  ; simulate 2MHz 8080

sim> **boot sol20**  ; enter SOLOS

sim> **; Load Air Traffic Controller from cassette tape**

sim> **set sol20 ena**  ; enable SOL20 device

sim> **set throttle 220k**  ; simulate 2MHz 8080

sim> **attach sol20t atc.cas**  ; insert ATC cassette in tape deck

sim> **set sol20t tape=fast** ; fast tape reads

sim> **boot sol20**  ; enter SOLOS

At the SOLOS “>” prompt, enter “XEQ” to load ATC from tape and execute the program.

sim> **; Boot CP/M 2.2 using NorthStar disk controller**

sim> **set sol20 ena**  ; enable SOL20 device

sim> **set sol20 ver=13c** ; SOLOS 1.3 modified for CP/M

sim> **set mdsa ena**  ; enable MDSA disk controller

sim> **attach mdsa0 CPM22b14-48K-SDC-SOL.NSI** ; insert CP/M disk in drive 0

sim> **boot sol20**  ; enter SOLOS

At the SOLOS “>” prompt, enter “EX E800” or press F5 (LOAD) key to boot CP/M from disk.

Virtual cassette images and CP/M 2.2 disk images supporting NorthStar single and double-density disk controllers are available at <https://github.com/deltecent/sol-20>.

### Sol-20 ENT Files

ENT files are text files similar to Intel HEX files that can be used to load binary files directly into SOLOS. To load an ENT file, boot the Sol-20 simulator into SOLOS then copy and paste the ENT file into the SIMH console. The ENT file should appear in the Sol-20 video window. It is also possible to attach a socket or serial port to the SOL20S device, enter “SET I=1” in SOLOS, then send the ENT file over the serial interface from your computer’s terminal emulation software.

# PMMI MM-103 MODEM and Communications Adapter

PMMI MM-103 support was added by Patrick A. Linstruth, [patrick@deltecent.com](mailto:patrick@deltecent.com).

## Overview

The MM-103 is a MODEM from Potomac Micro-Magic, Inc. (PMMI). The term MODEM is a contraction of MOdulator-DEModulator. The MM-103 combines data transmission and telephone line handling functions in a single device. The MM-103 supports baud rates from 61 to 600 baud using FSK modulation.

This device simulates the MM-103 data communications portion but does not simulate the MODEM functionality. MODEM functionality must be simulated by attaching a host serial port or modem to the PMMI device using the “Attach” command.

### PMMI Device Parameters

The PMMI device supports several parameters which can be configured in 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
  + STATUS – Status change messages
  + VERBOSE – Extra verbosity for debugging
* NODEBUG – Turn off one or more debug message levels.
* IOBASE – Change the I/O port address of the PMMI (default = C0H).
* RTS – RTS will follow DTR.
* NORTS – RTS will not follow DTR.
* BAUD – Set baud rate (default = 300).

### PMMI Example Usage

The MM-103 was supported by various communications software for CP/M. This example will use NightOwl Software’s Modem Executive (MEX) which can be downloaded from GitHub using the link below.

Using a breakout box, connect the following pins on your host’s serial port:

RxD <-> TxD

DTR <-> CTS

The following example will use MEX under CP/M to simulate dialing a phone number and communicating with a remote system by looping back transmitted data to the receiver:

sim> **dep CLOCK 2000** ; simulate 2MHz clock speed

sim> **set SIO nosleep** ; disable console sleeps

sim> **set PMMI ena** ; enable PMMI device

sim> **attach PMMI connect=/dev/ttyUSB0** ; Attach host serial port

sim> **attach dsk0 cpm56k.dsk** ; attach CP/M boot disk

sim> **attach dsk1 MEXPMMI.DSK** ; attach MEX disk

sim> **load dbl.bin ff00** ; load Disk Boot Loader

sim> **go ff00** ; boot the disk

56K CP/M

Version 2.2mits (07/28/80)

Copyright 1980 by Burcon Inc.

A>b:

B>mexpmmi

MEX (Modem Executive) V1.14

Clone Level 1

(for aid, type HELP or "?")

Copyright (C) 1984, 1985

by NightOwl Software, Inc.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* PMMI overlay version - 2.2c \*

\* Base port = C0 hex. \*

\* No carrier present. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[MEX] B0>> call 5551212

Dialing: .disc. .off-h. 5551212: Try #1 .wait.

Connected @300 bps

[MEX] [Term: CTL-J + "?" for help]

Hello, world!

The following example will use MEX under CP/M to connect to a R/CPM system over Telnet:

sim> **dep CLOCK 2000** ; simulate 2MHz clock speed

sim> **set SIO nosleep** ; disable console sleeps

sim> **set PMMI ena** ; enable PMMI device

sim> **attach pmmi connect=67.164.159.109:4667** ; attach to IP:PORT

sim> **attach dsk0 cpm56k.dsk** ; attach CP/M boot disk

sim> **attach dsk1 MEXPMMI.DSK** ; attach MEX disk

sim> **load dbl.bin ff00** ; load Disk Boot Loader

sim> **go ff00** ; boot the disk

56K CP/M

Version 2.2mits (07/28/80)

Copyright 1980 by Burcon Inc.

A>b:

B>mexpmmi

MEX (Modem Executive) V1.14

Clone Level 1

(for aid, type HELP or "?")

Copyright (C) 1984, 1985

by NightOwl Software, Inc.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* PMMI overlay version - 2.2c \*

\* Base port = C0 hex. \*

\* No carrier present. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[MEX] B0>>set online

.on-line.

[MEX] B0>>t

[MEX] [Term: CTL-J + "?" for help]

HOW MANY NULLS (0-9) DO YOU NEED? 0

Welcome to the virtualaltair.com Remote CP/M System!

The disk images used in the above examples may be downloaded from <https://github.com/deltecent/pmmi-cpm22>.

### PMMI and SIMH Timing

The PMMI device and communications software for CP/M depend on predicable timing to function properly. This is accomplished by setting CPU’s “CLOCK” register and using the M2SIO0 device for console input and output. If using the SIO device for console input and output, use “SET SIO NOSLEEP” to disable sleeps during keyboard checks as this interferes with SIMH’s timing.

# Hayes Micromodem 100

Hayes Micromodem 100 support was added by Patrick A. Linstruth, [patrick@deltecent.com](mailto:patrick@deltecent.com).

## Overview

Hayes made two S-100 MODEM adapters, the 80-103A and Micromodem 100. With the exception of a 50ms hardware timer available on the Micromodem 100, both modems are software compatible.

This device simulates the Micromodem 100 data communications portion but does not simulate the MODEM (MOdulator / DEModulator) functionality. MODEM functionality must be simulated by attaching a host serial port or modem to the HAYES device using the “Attach” command.

The Micromodem 100 does not support DTR, RTS, DSR, or CTS modem signals. To accommodate using the HAYES device with serial ports and sockets, the device will always raise RTS and will raise DTR when RI goes high or when the modem goes “off hook” in originate mode. The device will lower DTR when the modem goes “on hook” or DCD goes low.

### HAYES Device Parameters

The HAYES device supports several parameters which can be configured in 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
  + STATUS – Status change messages
  + VERBOSE – Extra verbosity for debugging
  + DEBUG – Debug messages
* NODEBUG – Turn off one or more debug message levels.
* IOBASE – Change the I/O port address of the HAYES (default = 80H).

### HAYES Example Usage

The Hayes Micromodem 100 is supported by various communications software for CP/M. This example will use the MM100 utility that was provided by Hayes. The software used in this example can be downloaded from GitHub using the link below.

Using a breakout box, connect the following pins on your host’s serial port:

RxD <-> TxD

DTR <-> DCD

RTS <-> RI

The following example will use MM100 under CP/M 2.2 to simulate dialing a phone number and communicating with a remote system by looping back transmitted data to the receiver. RTS will be used to drive RI and DTR will drive DCD:

sim> **dep CLOCK 2000** ; simulate 2MHz clock speed

sim> **set SIO nosleep** ; disable console sleeps

sim> **set HAYES ena** ; enable HAYES device

sim> **attach HAYES connect=/dev/ttyUSB0** ; Attach host serial port

sim> **attach dsk0 CPM56K.DSK** ; attach CP/M boot disk

sim> **attach dsk1 MM100.DSK** ; attach MM100 disk

sim> **load DBL.BIN ff00** ; load Disk Boot Loader

sim> **go ff00** ; boot the disk

56K CP/M

Version 2.2mits (07/28/80)

Copyright 1980 by Burcon Inc.

A>b:

B>mm100

Hayes Microcomputer Products, Inc.

Micromodem 100 Terminal program ver. 1.2

8 DATA BITS

1 STOP BITS

NO PARITY

300 BAUD

FULL DUPLEX

CAPTURE BUFFER CLEARED

DEBUG MODE OFF

COMMAND:D 555-1212

DIALING-555-1212

WAITING FOR CARRIER, PRESS ANY KEY TO ABORT.

CONNECTION ESTABLISHED

Hello, world!

[^A]

COMMAND:G

[ HUNG UP ]

[\* LOST CARRIER \*]

[ HUNG UP ]

COMMAND:X

The disk images used in the above examples may be downloaded from:

<https://github.com/deltecent/hayes-mm100>

### HAYES and SIMH Timing

The HAYES device and communications software for CP/M depend on predicable timing to function properly. This is accomplished by setting the CPU’s “CLOCK” register and using the M2SIO0 device for console input and output. If using the SIO device for console input and output, use “SET SIO NOSLEEP” to disable sleeps during keyboard checks that interfere with SIMH’s timing. Software that uses the 50ms hardware timer on the Micromodem 100 should be able run at any CPU clock speed.

# ImageDisk (IMD) Disk Image Support in SIMH

ImageDisk (IMD) disk image file support for SIMH was added by Howard M. Harte.

## 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.

## 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](http://dunfield.classiccmp.org/img/index.htm).

# CP/M-68K Simulation

Based on work by David W. Schultz (<http://home.earthlink.net/~david.schultz>) you can run Digital Research CP/M-68K as follows. Unpack cpm68k.zip and issue the command altairz80 cpm68k or the following commands at the simh prompt:

sim> set cpu m68k

sim> attach hdsk2 diskc.cpm.fs

sim> boot hdsk2

CP/M-68K(tm) Version 1.2 03/20/84

Copyright (c) 1984 Digital Research, Inc.

CP/M-68K BIOS Version 1.0

Simulated system of April 2014

TPA =16251 K

C>AUTOST.SUB

AUTOST.SUB?

C>

Typing “bbye” at the “C>” command prompt brings you back to SIMH.