altairsim — Changelog

2026-10-24 · 1facdc2

A simulation of the MITS Altair 8800 and the S-100 bus

Changelog

Every release of **altairsim**, newest first. Each entry is the short version — what you can do here that you could not do in the release before it. The User Manual describes the program as it is now; this document is the record of how it got there.

An entry for a fix says what the simulator did wrong, and what it does now. It uses the guest as the example: the program, the message it printed, the board and the setting. It names the register, the bit or the command that was wrong.

1.3.5

1.3.5 adds a math chip to the CompuPro System Support 1 and an overlay to the CADzilla board. Board clocks now keep real time when the processor runs flat out. The macOS packages are signed, and the host bridge has several corrections.

Board clocks keep real time. The 88-VI/RTC, the RTC-100 and the SS-1 interval timer counted their time in processor T-states. At flat out (`clock_hz = 0`, the default), they ticked as fast as the host ran the guest. For example, the MP/M II time of day ran fast, and the MP/M tick from SS-1 counter 0 came many times too often. Now each board keeps real time at every `clock_hz`. The SS-1 timer counts the 2 MHz bus clock of pin 49. The 88-VI/RTC also restarted its divider on each write to port `0xFE`, so each tick came late by the time the service routine took. Now only a write with bit 5 set restarts the divider.

The SS-1 math socket takes a 9511A or a 9512. The new `math` property is `none` (the default, an empty socket), `9511a` or `9512`. With a chip in the socket, ports `58` and `59` are the chip's stack and its command and status. Each command takes the cycle count of the data sheet, and END and SVREQ go to the slave 8259A. For example, the SS-1 manual's 9511A test program prints `OK` under CP/M with `SET ss1 math=9511a`.

An edge-triggered SS-1 interrupt needs its line held high. In edge-triggered mode, the SS-1's 8259A kept a request after its IR line fell. A real 8259A drops the request when the line falls before the acknowledge. Now the request falls with the line. An acknowledge with no request now gives the default IR7, not `RST 7`.

The CADzilla board has its overlay. A second picture, of 1 bit for each pixel, shows overlay color 2 where a bit is 1. A guest draws it through an ACRTC screen with the CHR bit set. The new `OLSEL` bit selects the half of the overlay memory, and `OLEN` now shows the overlay. A fetch now starts at a multiple of 8 words, and the area outside the picture is black, as on the hardware.

Snapshots are now version 5. A snapshot now holds the SS-1 timer phase and the CADzilla overlay memory. **`RESTORE` does not accept a snapshot from 1.3.4 or earlier.**

Host bridge.

- A wildcard in `R` and `HDIR` skips a host file whose name starts with `.`, such as `.DS_Store`. `HDIR .*` lists these files.
- `W *.ASM SRC` copies each match into the host folder `SRC`.
- Each part of a host path ignores case, also on Linux. Before, `R SRC/FOO.ASM` did not find the folder `src`.
- The sources of `R`, `W` and `HDIR` are 40% smaller. The programs are the same, byte for byte.

The macOS packages are signed and notarized. macOS starts the program with no message, also after a browser download, so the `xattr` step is gone. The first start of a new copy needs the internet. The *Troubleshooting* chapter says what to do on a Mac that is offline.

The MCP `run` tool stops at an address. The new `stop_at` argument stops the run before the instruction at that address runs. The stop is for that one call only.

Other corrections. Two machine file examples in the serial chapter now load. Other samples in the *User Manual* and the Monitor document now show what the program prints.

1.3.4

1.3.4 adds two CompuPro floppy controllers, `--set` and `SHOW SERIAL`. It makes a halted processor and `DEPOSIT` act as a front panel does, keeps the crystal on every run, and corrects many faults in the monitor, `--mcp` and the video windows.

The CompuPro Disk 1 and Disk 1A floppy controllers are new boards, `disk1` and `disk1a`. Each has an NEC µPD765A floppy chip, a DMA engine and a boot EPROM. The new machines `disk1` and `disk1a` add the board to a Z80, 64K of RAM and a CompuPro System Support 1. Mount a CompuPro CP/M 2.2 disk in drive 0 and run from `0000`, and the EPROM boots it to `A>`. The boards read 8″ disks, and the `disk1a` also boots a 5.25″ disk from unit 2. CompuPro's `FORMAT.COM` formats a blank disk. The `media` key of a drive sets the size and sides of a blank disk, and `create` makes an empty disk file.

A halted processor stays halted until `RESET`, and `DEPOSIT` uses the program counter. `RUN <addr>`, `EXAMINE` and `DEPOSIT` now do nothing while the processor is halted, as on a real front panel. They print the `PC` and "*RESET ends the halt.*" A `HLT` after `DI` now ends the run. `DEPOSIT <addr> <bytes>` is EXAMINE, DEPOSIT and DEPOSIT NEXT, so the `PC` ends on the last byte written. A value larger than `FF` is an error, and no byte is written.

A crystal is kept on every run. `RUN` ignored `clock_hz` when the program was started from a script or a pipe. It now keeps the crystal each time `clock_hz` is not 0: at the prompt, with `-x`, with `-s`, from a pipe and under `--mcp`. **A script that must be fast must change if its machine sets a crystal.** The `sbc200`, `sbc200v` and `dazzler` machines set 4 MHz, so start such a script with `SET cpu0 clock_hz=0`.

Numbers and case. A hex number that starts with `0B` is read as hex, and binary is written `%1010`. **A machine file that sets `sense = 0b10001110` must change to `sense = %10001110`.** Board types, endpoints, command-line options and settings with a list of values now work in any case. A machine-file word with the wrong case gets an error that names the fix, and `remove` refuses a value other than `true` or `false`.

New commands and options.

- `SHOW SERIAL` lists the serial ports on your computer and the unit that holds each one.
- `--set "cpu0 clock_hz=2000000"` changes settings for one run, before the machine's `startup`.
- `--mcp` refuses `-x`, `-s` and `-i`.
- `SET hb0 hostdir="My Project"` takes a value with a space in it.
- `SET` and `CONNECT` print the value as the board stored it.
- `W` chooses text mode from the extension of the CP/M file when you give no mode.
- `MOUNT … media=NAME` gives the disk its format before the mount.
- The terminal window's `cr` setting is now `crlf = on | off`, as on the console.
- `SET CONSOLE DEBUG=<file>` writes each line when the line is made.
- `tools/lbr-extract.py`, `tools/uncrunch.py` and `tools/unsqueeze.py` open CP/M library files.

Under `--mcp`, the console line is more reliable. `SHOW CONSOLE` names the unit that holds the console on every board. `CONNECT … console` takes the console from the unit that holds it, and the `ss1` board takes the console settings. The error for a lost line gives advice that works, and the `connect` tool lists the endpoints that `CONNECT` takes.

Video and terminal windows. At each stop, the console gets the keyboard back on macOS, Windows and Linux with X11. A terminal window sends its keys when `[display] keyboard = none`. On a Mac with a Retina screen, a video window is drawn at full resolution, and its text no longer changes between normal and bold. The help and the manual use one name for each window.

Other corrections. The `X` command of `ssm-8080mon` shows each register, not one value. The `vi` editor is on the CP/M example disk, in place of WordMaster and `OTHELLO.COM`. The Quick Start gives the macOS step for a downloaded copy. `HELP MOUNT` names `tnfs://`. The manual shows how to run the SSM PB1 routines from the SSM 8080 monitor. `DRIVING-WITH-AI.md` answers five more questions, and the manual puts the skills where Claude Code loads them.

1.3.3

1.3.3 adds the Cromemco Dazzler II, a mirror on a real serial port and a start-up file in your home folder. It also makes the floppy controllers check the density, and corrects three floppy bugs that stopped a guest from booting or formatting.

`~/altairsim.ini` runs at every start. Put monitor commands in a file called `altairsim.ini` in your home folder, and `altairsim` runs them before the machine's own `startup`. Use it for settings that are yours and not a machine's, such as where the debug output goes. `--mcp`, `-x` and `-s` do not run it, so a script or a test gives the same result on every computer.

A mirror can go to a real serial port. Add `|serial:DEVICE` to any endpoint, or start `--mcp` with `--mirror serial:DEVICE`, and a hardware terminal on that port watches the line and types on it. The port is 8N1 at 9600 baud with no flow control, and `?baud=N` selects another rate. It works on every platform. The program cannot tell when a terminal is on the port, so it sends the output at all times and the guest never waits for it.

New board: `dazzler2`, the Cromemco Dazzler II. It works like the `dazzler`, but it keeps the picture in 4 KB of RAM on the board and copies the writes it sees. Use it to test a program for a real Dazzler II. It shows random data when a program draws before it sets the base. It shows a picture at an address with no RAM, and it can show the second 2 KB half (the `page` property). `IN 0E` gives `0` in bits 0 to 5. The `dazzler` is not changed.

The Cromemco 16FDC and 64FDC boards can boot from drive B, C or D. The new `boot_drive` setting sets the boot-drive switches, which RDOS reads for an automatic boot. Before, the switches were fixed at drive A, and the only way to boot a different drive was to stop the boot with ESC and type the RDOS command, for example `BB`.

An 8-inch CDOS disk boots on the Cromemco 64FDC board. Before, the disk printed the CDOS sign-on on the `64fdc` board and then stopped with `Read error` and `Status=10`. The board gave the wrong value for bit D6 of port 04: it read `0`, as the 16FDC does, and the 64FDC always reads `1`. CDOS reads that bit for each 8-inch drive. A `0` marks a PerSci drive, which CDOS homes with the `¬RESTORE` line of port 04 (D3), and the 64FDC has no such line. The head never reached track 0. Now the bit reads `1` and CDOS homes the drive with the disk chip.

The floppy controllers check the density. The density bit must now agree with the density of the track, as on the hardware, and a read or a write at the wrong density gives Record Not Found. This applies to the `tarbelldd` (`OUT FC`), the `versafloppy` (port 63) and the Cromemco `16fdc` and `64fdc` (D6 of port 34). Before, the chip set its data rate from the bit and did not compare the rate with the track, so a read at the wrong density worked. The chip now has the density as its own input, because the Cromemco boards run 5.25-inch double density at the same 250 kbit/s as single density, and the rate could not tell the two apart. A 5.25-inch double-density track that a program formats on these boards is now recorded as double density.

A disk of your own for the `tarbelldd` may need a new boot sector. The two double-density CP/M disks in the Tarbell example have a new boot loader, built from Tarbell's own source with the double-density option on; the old loader read track 1 with the bit clear. A disk that has the old loader no longer boots: write the boot sector again with the double-density option on.

A single-density disk boots on the `versafloppy` board. The SD Systems DDBIOS PROM finds the type of a disk by a Read Address at double density first, then at single density. The board did not check the density, so the first try always worked, and the PROM took a single-density disk for a double-density disk and stopped with `COLD BOOT ERROR`. A read at the wrong density now ends in Record Not Found, and a seek with verify ends in Seek Error. Double-density disks boot as before.

A Write Track command ends at the index pulse with the default `timing`. With `timing = full` the simulated disk chip ended a Write Track only after the guest had sent one full revolution of bytes. The real chip ends the command at the index pulse, also when the guest has stopped sending. `FORMAT.COM` for CP/M Plus on the SD Systems board sends 9,672 bytes of a 10,416-byte track and then reads the status until the command ends. The chip waited for the rest of the bytes for ever, so `FORMAT` stopped after the first track. Now the command ends one revolution after the last request for a byte: the rest of the track is zeros, Lost Data is set and the track is written, as on the hardware. This applies to `versafloppy`, `tarbelldd` and the Cromemco FDCs.

A drive select written directly after a Restore or a Seek is obeyed with the default `timing`. The Tarbell CBIOS writes a Restore command and then selects the new drive, with no wait between them. With `timing = full` the old drive moved and the new drive did not. Now the chip steps at the next access of the guest to the board, so the drive that is selected when the head moves is the drive that moves, on the Tarbell, VersaFloppy and Cromemco boards.

`SHOW BOARD <type>` shows the default of each property, and `CONFIG SAVE` marks the settings that have it. Under the legal values of each setting, `SHOW BOARD` gives a `default:` line. A saved file ends a setting that has the default value with a comment, for example `baud = 9600  # default X.Y.Z`, where `X.Y.Z` is the release that the default belongs to. A setting without the comment was set on purpose. The comment does nothing when the file loads.

The errors for a wrong machine file or command line say more. An unknown key in `[machine]` ends with `Valid keys: base name startup`, and an unknown property of a board, a unit, `[console]`, `[display]` or `[terminal]` ends with `Valid properties:` and the names in alphabetical order. An error for a key in a `[[board.drive]]` or `[[board.region]]` table gives the line of the key, and the table it is in; before, it gave the line where the table started. Two machine names on the command line now say that only one machine file or built-in name may be given, and how to start from a built-in and change it: put `base = "name"` under `[machine]` in a file.

The AI skill's two long files open with a list of contents. `driving-with-ai.md` and `cheatsheet.md` each start with a `Contents` list, so an assistant that reads only the top of a file sees every section.

1.3.2

1.3.2 puts a serial line or the mirror of the console on a pseudo-terminal, adds a text log of the console, and corrects a tape bug.

A serial line can go to a pseudo-terminal: `CONNECT sio0:b pty`, on macOS and Linux. The program prints a name, `/tmp/altairsim0` for the first line. Open that name with a terminal program as you open a serial port, for example `screen /tmp/altairsim0`. No cable and no port number are necessary, and a file transfer program can use the line. `pty:PATH` selects the name, so that a machine file gets the same name each time. The carrier of the line is on while a program has the name open. When no program has it open, the line discards what the guest sends, and the guest does not wait. Windows has no pseudo-terminal, and refuses `pty`.

`--mirror pty` puts the mirror of the console on a pseudo-terminal, on macOS and Linux. With `--mcp --mirror pty`, the program prints a name, `/tmp/altairsim0` for the first mirror. Open that name with a terminal program to watch the session that the AI assistant controls and to type on it. `--mirror pty:PATH` selects the name. In the monitor and in a machine file, the same mirror is `<endpoint>|pty`, and a machine file that has it prints the name at startup. Windows has no pseudo-terminal, so `--mirror socket:PORT` is the mirror there. A mirror now also sends the end of a long listing to the watcher immediately, and does not wait for the next command from the assistant.

`--log FILE` keeps a text log of the console while an AI assistant controls the machine. With `--mcp --log session.log`, the program writes each character that the guest prints to the file, at the time that the guest prints it. Read it, search it or follow it in a second window, for example with `tail -f`, or with `Get-Content -Wait` in PowerShell. In the monitor and in a machine file, the same log is `<endpoint>|FILE?fmt=text`. The file tap also has a new `append` option, and a tap can now go on a mirror: `<endpoint>|socket:PORT|FILE`.

The package has two scripts that watch a mirror. `tools/mirror-watch.sh` (macOS and Linux) and `tools/mirror-watch.ps1` (Windows) show the console of the machine in a second window while an AI assistant controls it through `--mcp --mirror socket:2323`, and connect again after the simulator starts again. Windows had no viewer before, because it installs neither `nc` nor `telnet`. The scripts are by trgeuy. The docs now name `nc` first and `telnet` second: a new Mac has `nc` but no `telnet`.

A machine file with the wrong brackets on a table is refused. `[board]` with single brackets loaded as if it was `[[board]]`, and `[[console]]` loaded as if it was `[console]`. Now each is an error that gives the line and the correct form. A file that has this mistake loaded before and does not load now; change the brackets as the error says.

A tape that you mount after another tape starts at its first byte. After a program loaded from a cassette, the next `MOUNT` on that recorder gave the guest one byte of the old tape before the new tape. A

bootstrap loader stored that byte, and the program did not start. This is corrected for the 88-ACR, the 88-UIO and the Sol-20. `UNMOUNT` also removes that byte.

The documents are corrected and extended. The manual describes `SHOW BUS`, and its configuring chapter says how a machine file is loaded and what `CONFIG SAVE` saves. The AI driving guide says to send one line in each `run`, to give the full prompt as `until`, and that a server registered as `./altairsim` can be a different release from the one on `PATH`.

1.3.1

1.3.1 is a small release that starts the SBC-200 in its monitor, and makes the monitor show the memory that the processor reads. The `basic1` example also tells you how to start BASIC.

The SBC-200 starts in its monitor after a reset. The `sbc` board has the auto-start circuit of the real board. Set `start` to a 4K boundary, and after a reset the Z80 reads the PROM there until the PROM reads port `7F`. The `sbc200` and `sbc200v` machines have `start = E000`: they start with `RUN`, and `RESET` then `RUN` brings the monitor back. The board also has the memory jumpers of the real board. `rom_size` and `bank` set where the four PROM sockets are, and `ram` is the 1K onboard RAM, which is new. A `[[board.socket]]` address that is not a socket of the board is refused: before, any address from `E000` to `FFFF` was accepted. A snapshot from an earlier version does not load.

`SHOW BUS MAP` shows the memory that answers now. Before, it listed a range that a board had switched out, such as the boot PROM of the `turnkey` machine after the loader ran, and `WHO` gave a different answer. Now the two agree. `SHOW MACHINE` shows how each board is built, as before.

`DISASM` and `DUMP` show the boot PROM of the Turnkey board. Before, they showed `FF` there and at the Auto-Start jump, while the processor read the correct bytes.

The `basic1` example tells you how to start BASIC. After the tape loads, the machine is quiet, because the loader of BASIC 1.0 does not start BASIC. The notes on the screen now tell you to press Ctrl-E and type `RUN 0`. They also tell you that `_` deletes a character: the Backspace key does not work in BASIC 1.0.

1.3.0

1.3.0 is the release that adds new boards and sound, and lets a floppy take the time of a real disk. The North Star MDS-A and MDS-A-D controllers, the CADzilla graphics board and the Newtech music board are new. The Cromemco D+7A plays the JS-1 speaker. You can paste a file into the guest, and an AI assistant can let a guest run between calls.

New boards

The North Star floppy controllers. `mdsa` is the MDS-A single-density controller, and `mdsad` is the MDS-A-D double-density controller. Each one has its own boot PROM, and each one boots CP/M and North Star DOS from a 5¼″ North Star disk image (`.NSI`). The `mdsad` reads and writes single-density, double-density and two-sided disks. These boards have no ports. A board decodes a 1 K block of memory at `E800`, and a guest gives a command when it reads an address in the block. The machines `northstar` and `northstardd` have the boards. Their drives start empty. Mount your image in `fd0:drive0`. To boot, type `RUN E900` on `northstar` or `RUN E800` on `northstardd`.

The CADzilla graphics board. `cadzilla` is a new design built from two chips of the mid 1980s. The Hitachi **HD63484 ACRTC** is a CRT controller with a drawing processor and 2 MB of its own frame memory. The Brooktree **Bt453** is a RAMDAC, a color table of 256 entries. The board uses one block of 8 I/O ports and no memory. The guest writes drawing commands into the ACRTC's FIFO. The board does every command of the chip: lines, rectangles, polygons, circles, ellipses, arcs, area paint, patterns and block copies. The picture has 8 bits for each pixel. It shows on a fixed-frequency VESA monitor that you select with `mode`: `640x480`, `800x600` or `1024x768` (the default).

The `draw_rate` strap sets the drawing speed. With `full`, the default, each command completes immediately. With `real`, each command takes the time that the HD63484 data sheet gives it. Use `real` for a game or any guest that is sensitive to time. The `interrupt` strap connects the ACRTC's IRQ* to `int` or `vi0`..`vi7`. It is `none` by default. `SHOW <id>` has a `wiring` line. It tells you when the guest set the ACRTC in a way that the board is not wired for.

For developers, a test can now check the whole picture of a video board as a text grid. When the check fails, it writes what the board drew as a `.ppm` file that you can open. The Developer Guide chapter *Writing a video board* shows how.

The Newtech Model 6 Music Board. `music6` has one output port, a 6-bit D/A converter and a speaker. A guest writes the port in a timed loop, and you hear the result on the sound output of your computer. The default port is `24`, and the board also answers at `25`, `26` and `27`, as the real board does. The `port` setting takes `04`, `14`, `24` and so on to `F4`.

Sound

The Cromemco D+7A (`d7a`) now plays the speaker of each JS-1 joystick. A guest that writes a waveform to the speaker port is heard on the sound output of your computer. Two new settings, `speaker1` and `speaker2`, give the analog channel of each speaker: `none`, or `1` to `7`. The defaults are channel `1` (port `19`) and channel `3` (port `1B`).

The `d7a` also makes the processor wait 5.5 µs on every `IN` or `OUT` to an analog port, as the real board does. An analog `OUT` now takes 21 T-states at 2 MHz, not 10. Before, a tone played too high, and a game

that reads the joysticks ran slightly too fast.

Sound needs a crystal. At full speed (`clock_hz = 0`, the default), a sound board plays nothing. Set the speed, for example `SET cpu0 clock_hz=2000000` for the `music6` (Newtech's programs are timed for 2 MHz) or `4000000` for the `d7a`. `SHOW` on the board tells you if the speaker plays and, if it is silent, the reason.

Boards that do what the real board did

Floppy controllers can take the time of a real disk. A new board setting, `timing`, is on the `tarbell`, `tarbelldd`, `versafloppy`, `16fdc`, `64fdc`, `mdsa` and `mdsad`. These boards make the processor wait until the disk has the next byte. With `full` (the default), the wait takes no time, as before. With `real`, it takes as long as on the real machine. A seek takes its step time, and each byte comes at the speed of the disk. Set it with `SET fdc0 timing=real`, or `timing = "real"` in the machine file. On the `tarbelldd`, bit 7 of port `FD` now shows that a disk command is still running. Before, it always showed that the command was done.

Cromemco's GOTCHA runs. The status port of the Dazzler (`dazzler`, `IN 0E`) read 0 on its six unused bits. They now read 1, as on the board. The odd/even line bit (D7) also stays 0 during the vertical blank, and the port reads `3F` for the full 4 ms between frames. GOTCHA times each move on that value. Before, it drew its picture and then stopped. A joystick on the `d7a` pushed all the way gave half of the range of the A/D converter (`3F` and `C0`). It now gives the full range (`7F` and `81`), so GOTCHA can be steered right and up. Dazzle Doodle draws only when the stick is in the middle half of its range. If you push the stick more than half, Doodle stops the line until the stick comes back.

A guest at a prompt runs at the speed of its crystal. With a crystal set, a guest that waited at a prompt after `RUN` ran at about 1.3 times its crystal. A guest that counted a timeout while it waited for a key counted it too fast. At full speed, nothing changes. `idle` still lets the processor rest at a prompt.

Driving a machine from the monitor and over `--mcp`

- **`PASTE <file>`** sends a host file to the keyboard of the guest, as if you pasted it from the clipboard. The file can be of any size and no character is lost. Use it in a `startup` list to enter a program that the guest reads from its keyboard. Examples are a SOLOS `ENTER` script and an Intel HEX file for `PIP`. `NOPASTE` stops a paste that is not finished. The Sol-20 keyboard now gives the guest the next key when the guest looks for it. Text pasted into a Sol-20 no longer arrives very slowly.
- **`TYPE` sends a quote and a control key.** `TYPE "PRINT \"HI\"\r"` typed only `PRINT \`. `TYPE` now sends the full line. It also has two new escapes: `\^X` is the key Ctrl-X (`\^C`, `\^Z`, `\^[` for ESC), and `\xHH` is the byte with the hex value `HH`. If your text had `\x` and two hex digits, or `\^` and a letter, write the backslash as `\\`.
- **The guest can run between MCP calls.** Two new MCP tools, `start` and `stop`. `start` starts the guest and returns at once. The guest then runs between tool calls until `stop`, a `HLT` or a

breakpoint. All the other tools still work while it runs, and `status` tells if it still runs. Use it for a server on the guest that must answer its clients in time, and for two machines that talk to each other. It also lets a person take over the console through `--mirror`.

- **A trace started by an assistant is written to its file.** `TRACE ON <file>`, sent through the MCP `monitor` tool, now writes the trace to the file. Before, the file stayed empty or the simulator stopped with an error on the next `run`. Through MCP, `TRACE ON` with no file is refused. Give a file, or use the `bus_trace` tool.
- **`R` no longer puts a space in a CP/M name.** `R *.TXT` copied a host file such as `my notes.txt` to a CP/M file with a space in its name, which you cannot type at `A>`. `R` now removes a space, as it removes the other characters that CP/M cannot have in a name. The disks in `examples` have the new `R`, which shows `(1.2)` in its help.
- **Skills for the programs that run in the machine.** The package has four new Agent Skills in `skills/`, for an AI assistant: `altairsim-mbasic`, `altairsim-cpm-build`, `altairsim-cpm-text` and `altairsim-hostbridge`. The assistant loads a skill only when the task needs it. `DRIVING-WITH-AI.md` names the skill files in place of the steps to build a CP/M program.

Machine files

- **Errors give the line.** Each error in a machine file now gives the line number: `mine.toml: line 7: ...`.
- **A table written two times is an error.** A file that has `[machine]`, `[console]`, `[display]` or `[terminal]` two times does not load. The same is true for a `[board.unit.<name>]` written two times below one `[[board]]`, and for a key written two times in one table. The error gives both lines. A key above the first table, text after a table header, and a `startup` list with no closing `]` are also errors now. **A machine file that loaded before can be refused now.** The error tells you which line to remove.
- **Two files that name each other as `base`** now give one short line: the file that you loaded, the line of its `base`, and the cause. Before, the file and line were there once for each of the 8 levels.

The documents

- The serial and disks chapters of the User Manual now show the machine-file form of `CONNECT` and `MOUNT`. The form is below the command, under the words "In a machine file:". The serial chapter has a new section, "The same in a machine file". The disks chapter has a table that gives the key for each part of a `MOUNT` command.
- The configuration chapter has a new example of a machine file that uses a second machine file as its `base`. It also shows the message for two files that name each other. The samples that change a board of the base now leave the `type` out. A `[[board]]` with a `type` replaces the board, and it loses the settings of the base.
- The chapter *Moving files in and out* has the correct steps for `R` and the list of the characters that `R` removes.

Fewer examples in the package, and more at altairsim.com

The package now has five examples: `cpm`, `basic4k`, `basic1`, `debugger` and `ai-mcp`. `basic4k` and `basic1` replace `basic`. `basic4k` has Altair 4K BASIC 3.1 as a `.tap` and as audio. `basic1` has Altair BASIC 1.0, the first one, as a `.tap` and as audio. More examples, with their media and more documentation, are at https://altairsim.com.

The `cpm` example no longer has the two FDC+ machine files, `cpm22-fdcplus.toml` and `cpm22-fdcplus-hdf.toml`, and the 1.5 MB disk `CPM22-48K-HDF.dsk`. The boards chapter still tells you how to set up the FDC+ in the `default` machine. The `recipes/` folder is removed from the package, and the worked-examples chapter no longer has the BASIC 1.0, 88-HDSK and Disk BASIC walkthroughs.

Smaller things

- `SHOW BUS`, `SHOW ROMS` and `SHOW JOYSTICKS` now line up their columns.
- In `HELP`, an example remark starts with `;` and the remarks line up.

1.2.0

1.2.0 is the release that puts a disk at the other end of a serial line, and makes the documents easier to read. A new board gets its disks from an FDC+ drive server, as a real FDC+ does. Several boards now do what the real board did and not less. And the User Manual, *The Monitor* and *The Debugger* are rewritten in short, simple sentences.

The FDC+ serial drive

The new `fdcplus` board is the FarmTek FDC+. In its serial drive mode (drive types 6 and 7), a drive server on another computer keeps the disk images, and the board gets them a track at a time over a serial line. So the simulator and a real FDC+ Altair can use the same images through one server, and you can test a drive server without a real FDC+. Type 7 is an 8″ drive, including the 8 MB disk, and type 6 is a minidisk. `examples/cpm/cpm22-fdcplus.toml` boots CP/M from a server: set your serial port in the file and run it. `SET fdc0 DEBUG=error` reports the line's faults: a server that does not answer, a bad track, a write that the server did not take.

The board also does drive type 5, the FDC+'s 1.5 MB floppy, from disk images that you `MOUNT`. The stock DBL boot PROM boots it, as on the real board. `examples/cpm/cpm22-fdcplus-hdf.toml` boots Mike Douglas's 1.5 MB CP/M 2.2, and the disk is in the package.

The SciTronics RTC-100 clock board

A new board, `rtc100`: the SciTronics RTC-100, a battery-backed S-100 calendar clock from 1980. It reads your computer's date and time. A guest can set it, and the setting stays after a RESET. It can interrupt once a second with an `RST` that you select.

Boards that do what the real board did

- **The VDM-1 shows its whole character set.** Codes 00 to 1F now show the graphics characters of the board's MCM6576 ROM, as the board did when it left the factory. Before, they were always blank. The new `blanking` property sets the SW5/SW6 switches. The screen RAM now holds random bytes at power-on, as real RAM does; set `fill = zero` or a `seed` to get the same screen each time.
- **8″ disks need a 2 MHz processor.** The `dcdd` and `mds` data port now holds one byte, as the real board does, and not a queue. The 8″ software times its second byte for 2 MHz, so a faster `clock_hz` cannot read the disk — the same as on a real Altair. Full speed and 2 MHz work as before, and minidisk software works at 4 MHz too.
- **The Generic SIO's `sior0` is really Rev 0.** The profile that was called `sior0` was wired as a Rev 1 88-SIO. It is now named `sior1`, and it is still the default, so a machine that names no profile is not changed. `sior0` is now a real Rev 0 board. A machine file that says `profile = "sior0"` and wants the old wiring must now say `sior1`.
- **An empty ROM socket is empty.** A `rom` region with no `mount` now reads `FF`, even when it has a `size`.

Documents that are easier to read

The User Manual, *The Monitor* and *The Debugger* are rewritten in shorter, simpler sentences, with one name for each thing. Many facts in them were checked against the program and corrected on the way.

A new kind of document is in the package: **`recipes/`**, short walkthroughs that you type along with. Each one starts with an empty chassis (`altairsim -n`), adds the boards one line at a time, saves the machine with `CONFIG SAVE`, and loads it back. One builds a CP/M Altair, one builds a Cromemco Dazzler machine with a Z80, and one changes a machine that already works. The **Dazzler example is now in the package** too: `examples/dazzler/` boots Li-Chen Wang's Kaleidoscope.

At the prompt and over `--mcp`

`SHOW BOARDS` now lists each board type on one line; `SHOW BOARD <type>` gives the full description. `SET MACHINE name=<name>` names the running machine, so a machine that you build at the prompt is not saved as `name = "none"`. A path in an `altairsim -s` script is now relative to the script's folder, as in a `DO` script, so each `.ini` in the examples runs from any folder.

The MCP server now checks the type of each tool argument and gives an error that names it. Before, `run {"from": "0xFF00"}` ran from address 0. `run` now obeys `SET BUS UNCLAIMED` and stops on a `BREAK TAPE STOP`, and the `--mcp` console follows the console's transforms when you change them.

Smaller things

`MOUNT` and `UNMOUNT` messages now fit the unit, and a suggested `MOUNT … CREATE` keeps the quotes of your path. `DEBUG=` reports line up while a guest runs. The CP/M disks in the examples carry the current `R.COM`, `W.COM` and `HDIR.COM`, and so does the hard-disk image; `W` no longer puts CP/M attribute bits in the name of the file that it writes. `CONFIG SAVE` now writes a value that contains a `"` so that it loads back.

1.1.0

1.1.0 is the release that makes a running machine reachable. Where 1.0.0 filled out the processors and the boards, 1.1 is about getting at the machine once it is running: a person telnets into a line and lands in a session that behaves, an AI drives one over `--mcp` and can stop a `run` it started, and a boot sequence is composed at the prompt instead of hand-edited into a TOML. The Altair 680b, meanwhile, leaves for a simulator of its own.

A `telnet:` endpoint, so a human telnets in without the double echo

`CONNECT` ing a unit to a plain `socket:PORT` gives you a raw pipe — which is right for wiring one machine to another, but when a person points `telnet` (or `nc`) at it, nothing negotiates the terminal: the client echoes every keystroke locally *and* the guest echoes it back, and Enter arrives as a whole line with the CR mangled to LF. The new **`telnet:PORT`** endpoint is `socket:`'s twin that speaks the Telnet protocol: on connect it offers `WILL ECHO` / `SUPPRESS GO AHEAD`, so a stock client drops its local echo and sends one key at a time, and it strips the inbound protocol bytes the guest should never see. `telnet:HOST:PORT` dials out as the client. Use it wherever a human telnets into a BBS or a monitor; `socket:` stays a raw pipe for machine-to-machine links and the live mirror.

A `telnet:` line also **greets each caller** with one line the guest never sees — `Connected to AltairSim X.Y.Z (sio0:b) on port 2323` — so you can tell at once whether you reached the right machine and the right line. `?banner=off` turns it off; a raw `socket:` stays silent unless you ask for `?banner`, because another machine is usually the one calling it.

The PMMI modem answers a real BBS

A PMMI configured to answer (`answer=PORT`) now keeps its phone line **plugged in for the life of the machine**, instead of only while the guest is holding DTR high. Answer-mode software that waits for a ring

with the modem on-hook — CBBS is the canonical example — can finally hear the call: it sits in its ring-wait loop with DTR low, and an inbound connection rings it, exactly as a real auto-answer modem behaves. Dropping DTR hangs up the current call but leaves the line listening, and a caller who hangs up before being answered is cleaned up promptly, so the next call still gets through.

The PMMI's `dial=` / `answer=` line speaks Telnet too, and **by default** — its far end is almost always a person's telnet client — so a BBS the PMMI answers behaves the moment someone telnets in, with no configuration. Set `telnet=off` on the board for a raw modem link to another simulator.

Driving a machine over `--mcp`: a `run` you can stop, and a `status` that always answers

Three things that made `--mcp` hard to drive are fixed together.

A `run` can be stopped early. Cancel the request with the standard MCP `notifications/cancelled` message — the server reads its input while a `run` is going, so it sees the cancel at once — or send the `altairsim` process a ^C (or `kill -INT`). Either way the `run` stops, returning `stopped: "interrupted"` with what the guest had printed so far and the machine as it was. ^C on a hand-started server is no longer instantly fatal; a second ^C, before the first is reported, ends it as before.

`status` answers immediately, always — board id, whether the server is busy on any call, and the step count and PC as of the last `run`'s completed slice. It is a snapshot, not a live read: `pc` and `steps` are only as fresh as the last `run`, and `generation` is the one field guaranteed to climb. Poll it standalone rather than sequencing it with the rest of a script.

`timeout_ms` is a real ceiling. Bytes arriving on a line used to renew the deadline, so a peer that said anything at all kept the call going; ask for six seconds and you could wait two minutes. It now bounds the call in wall-clock time no matter what arrives. Give a long transfer the time it needs up front (up to 600000 ms) and let `until` end it early — the budget is a ceiling, not a wait. A call that hits it returns `stopped: "timeout"` with what it read, which is a normal result to resume another `run` on.

`STARTUP` — build a machine's boot list at the prompt

A machine file's `startup = [...]` is the operator's keystrokes written down — `MOUNT` the disk, `LOAD` the loader, `RUN`. Until now the only way to compose that list was to hand-edit the TOML, with the quote-escaping a path with a space needs, because nothing in the monitor wrote to it. The new `STARTUP` command edits the list in place: `STARTUP ADD <command>` appends a line exactly as typed, `STARTUP REMOVE <n>` drops one, `STARTUP CLEAR` empties it, and a bare `STARTUP` shows it numbered. `ADD` stores the rest of the line verbatim, quotes and all, so a boot sequence you assemble at the prompt is what `CONFIG SAVE` writes and `CONFIG LOAD` reads back.

Smaller things

`SHOW CLOCK` reports emulated time in the guest's own seconds, so you can see how far the machine thinks it has run against the wall clock. **A ROM region can relocate an image**: a HEX or S-record file whose addresses do not match where you want it loaded no longer has to be rewritten first. And the run path is faster — the bus peeks through its cached page decode instead of scanning the backplane, and the CPU `HISTORY` snapshot no longer builds register definitions it will not use.

Fixes: the first byte to arrive on an idle 6850 line takes a character time to shift in, as the real chip does, instead of appearing instantly; the Host Bridge's `R.COM` upper-cases the CP/M name it creates, so the file it writes is the one CP/M can open; a board that outlives its clock is no longer left holding a dead pointer; and under `--mcp`, a machine file's `#>` notes go to stderr instead of landing on stdout ahead of the first MCP reply, where a strict client could not parse them.

The Altair 680b moves to its own simulator

The **MITS Altair 680b** — the Motorola 6800 machine 1.0.0 added — leaves altairsim for a simulator of its own, **swtpcsim** (https://github.com/deltecent/swtpcsim), where it joins the rest of the 6800/6809 world it belongs to. The 6800 CPU core, its disassembler and assembler, the `altair680` machine, the `680io` / `680uio` / `680kcacr` boards, and the MON680 / KCACR PROMs go with it. altairsim is once again purely an 8080/Z80/8085 S-100 simulator, which is what its bus, its boards and its machines all are. The **Motorola S-record** support that arrived alongside the 680b stays — `LOAD` still reads `.S19` and takes `FORMAT=SREC`, because it is a general file format and not a 6800-only one.

1.0.0

1.0.0 is the version that says the simulator is what it set out to be. Where 0.4.0 filled the backplane, 1.0 fills out the processors behind it and the world around it: two more CPU cores — including the first that is not an Intel — more S-100 boards and the machines they boot, a disk that arrives over the network, and an AI assistant that can now drive a machine in real time against real hardware. It is the release the earlier ones were building toward.

Two more CPU cores — and the first that isn't an Intel

The **Motorola 6800** joins the 8080 and Z80, and with it the **MITS Altair 680b** — a different computer from the 8800, a different bus, a different instruction set. `altairsim altair680` brings it up under **MON680**, the 680b's own monitor ROM, and `DISASM` and `EDIT` speak 6800 the way they speak 8080 where that core is active, down to a 6800 assembler for entering code a byte at a time. Its serial I/O board (`680io`) carries the console, the Universal I/O board (`680uio`) adds a second port, and the **KCACR cassette** board loads and saves off tape — enough of the machine that MITS **680 BASIC** loads from

cassette and runs. Because Motorola tools speak Motorola formats, `LOAD` now reads a **Motorola S-record** (`.S19`) file as well as Intel HEX, and takes an explicit `FORMAT=SREC`.

> **Since moved.** The 680b and its 6800 core left altairsim after 1.0.0 for their own simulator, **swtpcsim** (https://github.com/deltecent/swtpcsim) — see 1.1.0. The S-record `LOAD` support described here stays.

The **Intel 8085** joins them too — the 8080's binary superset, with `RIM`/`SIM` and the on-chip interrupt system (the non-maskable `TRAP` and the maskable `RST 5.5/6.5/7.5`, each with its mask and pending latch, in hardware priority order) that the 8080 never had. It is faithful where the two chips actually differ, down to the two undocumented condition bits — **V** (signed overflow) and **K** — and all ten undocumented opcodes, which execute rather than trap. Like the other cores it earns its place at the gate rather than by inspection: Ian Bartholomew's 8085 exerciser, whose expected CRCs were read off real silicon, runs its 2.9 billion instructions against it on every CI push, on all three platforms. `altairsim 8085` is the direct analog of the `z80` machine — an 8085, 64K, and a 2SIO console — and the disassembler and assembler speak 8085, undocumented opcodes flagged the way `DDT` flags a byte outside the published set (`??= 08  *DSUB`).

More boards, and the machines they boot

Five more controllers and their operating systems boot alongside the rest:

- The **iCOM FD3712/FD3812** 8″ floppy controller — a programmed-I/O command engine with its driver in a high-memory boot PROM — boots CP/M 2.2 in single and double density and both revisions of iCOM's own **FDOS** disk operating system (`altairsim icom`).
- **S100Computers**' modern reproduction boards boot **CP/M 3**: the **Dual SD** controller runs two microSD cards as raw drives, and the **IDE-AB** board runs a **CompactFlash** card as drives A:/B:; because both lay a card out identically, one system image is interchangeable between them, and the combination board spans all four drives (`altairsim dualsd`, `altairsim dualide`, `altairsim dualidesd`).
- The **CompuPro System Support 1** is a real-time clock, a serial channel, an interval timer and a pair of cascaded interrupt controllers on one card — its OKI MSM5832 reads your host's own date and time and survives a RESET, and everything but the empty math-chip socket is implemented (`altairsim compupro`).
- The **SSM PB1** is a **PROM burner**: you *run* a period EPROM-programmer routine against a board that presents the real socket-and-arm interface, burn a 2708 or 2716, and `SAVE` the result as Intel HEX. `examples/pb1` carries SSM's own driver routines from the PB1 manual.

The **Tarbell #2022** now boots a disk whose CBIOS moves sectors by **DMA**, driving the card's on-board Intel 8257 to steal S-100 bus cycles and drop each sector into memory itself — the first board in the simulator to master the bus (`altairsim tarbelldd-dma.toml`). **Bank-switched RAM** is now its own

board, `bankmem`, modeling four real decoders (Vector, Cromemco, North Star, and the SD Systems **ExpandoRAM II**), and on the ExpandoRAM II an SD Systems machine boots **banked CP/M 3** with a full 48K TPA under the operating system's own bank. And the **SSM 8080 System Monitor V1.0** is now a built-in ROM the way the Eberhard monitors are — boots by name, nothing to fetch — with a cheatsheet to match.

Two new serial boards land the SSM cards properly. **Generic SIO** (`gsio`) is a describe-it-by-strap serial card: you say where the status/control and data ports sit, which status bits carry data-available and transmit-empty, and whether the shared inverter gate is engaged, and that is the port — with two independent channels and built-in profiles that imitate the MITS SIO Rev 0, the Cromemco TU-ART, the IMSAI SIO-2 and the CompuPro channels. And the fully emulated **SSM IO-4** (`io4`) models the actual 2P+2S card — its two 1602-family UARTs with real programmable word length, parity and stop bits; a strappable status word matching the W1/W2 header; four 8212 latched parallel ports; and header-W4 interrupts, where each serial receive and transmit and each parallel input can be strapped onto any vectored-interrupt line. It boots the SSM 8080 monitor as its console.

A disk that arrives over the network

`MOUNT` now takes a **`tnfs://host/path`** URL and fetches the image off a **TNFS server** — the network file system the FujiNet project speaks — instead of reading a file on your host. Once mounted it behaves like any other disk: the guest reads and writes it, `WP` protects it, and your changes are written back to the server. It works for every disk and tape controller, since it is the image that arrives over the network, not anything the board can tell apart from a local file. And because a network can vanish mid-session, altairsim now says so out loud if the server stops accepting writes and keeps retrying, rather than letting your changes pile up unsaved in silence.

Driving a machine over MCP got real teeth

An AI assistant driving a machine over `--mcp` no longer has to screen-scrape the text monitor to inspect it. The interface gained **twelve first-class, typed tools** for the work it used to reach through the monitor prompt by hand — `step` and `breakpoints`, `snapshot` / `restore`, the always-on `bus_trace` flight recorder, `mem_fill` / `mem_search` / `mem_save`, a stateless `disasm` that decodes a ROM with no CPU running, typed `mount` / `connect` wiring, and a `bus_irq` view of who is pulling the interrupt lines — bringing the MCP surface to the parity the design lays out, structured data in and out where there used to be a prompt to parse.

And two things it could not do at all, it can now. It can **share a session**: `--mirror` opens a live bidirectional socket onto the same console the assistant is driving, so a person can watch what it is doing and take over the keyboard, then hand it back. And it can work with **real hardware in real time** — wire a serial line to an actual device and set a real clock speed, and the `run` tool paces the guest to that clock instead of running flat out, so a reply that takes the device a fraction of a second arrives while the guest is still waiting for it; `run` no longer cuts a transfer off when its time budget runs out, so a boot loader

pulling its whole system image in over a serial disk finishes in a single call. The console's own text transforms now ride along under `--mcp` and `--mirror`, so what the assistant reads — and what a watcher sees — matches what a person at the real terminal would, parity bit and carriage returns and all.

A terminal in its own window

A serial line can now open its own **built-in terminal window** the simulator draws itself — `CONNECT sio0:a terminal`, or `connect = "terminal"` in a machine file — so the machine's console and the monitor's command prompt live in separate windows with no telnet client and nothing to install. It speaks four dialects the way period software expects them, `?emulation=` picking one: **VT100/ANSI** (the default), the CP/M **ADM-3A**, the **VT52**, and the Heath/Zenith **H19** — the last three being terminals no modern emulator provides, which was the whole point. It draws in the real **DEC VT220** character set decoded from the terminal's own character ROM, on a softened green phosphor (or `?phosphor=amber`), and carries the same fold-the-bytes settings the console has in a `[terminal]` section — which earn their keep on a period even-parity monitor that computes parity into bit 7. In a headless build the endpoint refuses cleanly rather than opening a line nobody can see.

The video window

`SET DISPLAY crt=on` (or `[display] crt = true`) paints any video window like the period tube — the 4:3 aspect of a real monitor, the raster softened into a phosphor glow rather than a grid of hard pixels — and under that look a window opened at a chosen `width` fills exactly that many pixels. Each video board now opens its **own** host window, so a machine with two video cards shows two pictures at once rather than sharing one.

Coming from another simulator

For anyone arriving from AltairZ80 (SIMH) or z80pack, 1.0 adds both a map and an on-ramp. A **migration guide** lays out what carries across untouched, an objective side-by-side, the command mappings — including the deliberate SIMH `D`/`E` swap — disk-image compatibility, and worked machine-file conversions (`docs/migrating.md`). And the monitor now runs **command scripts**: `DO <file>` reads a file of monitor commands and executes them in order, a machine file can name one to run at startup, and the `.ini` form is deliberately close to what a SIMH user already writes. A new `MACHINE` command builds or clears a machine from the prompt, so a script can stand one up from nothing the way a SIMH `.ini` does — and every shipped example now carries a `.ini` twin of its `.toml` to read from.

The debugger, and small conveniences everywhere

Cycle breakpoints now take a condition: `BREAK MEM W 100 IF B==0` stops only on the access whose registers hold, and `BREAK IO R 10 LOADS A>7F` tests the byte an `IN` actually read rather than the state it read it with. `EXAMINE <addr>` now shows the register line and the next instruction along with the byte.

The byte-at-a-time `EDIT` assembler learned the **Z80**, so between the new cores above and this, `EDIT` and `DISASM` now speak whichever of the four processors is running. And a scatter of the prompt's rough edges are smoother: a leading `~` in a typed path expands to your home directory, monitor subcommands abbreviate by unique prefix, `LOAD` reports the page count of a CP/M `SAVE`, the host serial layer accepts non-standard baud rates like 76800, and the Sol-20's `MODE SELECT`, `CLEAR` and `LOAD` keys are reachable from the keyboard. A relative path now resolves against **one** place — the directory the machine was loaded from, whether a machine file mounts the file or you type its name at the prompt — so a name that boots from a machine file works the same when you type it, and `SHOW PATHS` prints the one base directory it uses (the hostbridge sandbox is the only directory kept separate). And the console property that carries the `^E` stop key is now spelled `stop`, matching the switch it presses, with `attn` kept as an alias.

The package, and the documents in it

The monitor prompt and the debugger are now their **own shipped PDFs** — `altairsim-monitor.pdf` and `altairsim-debugger.pdf` — pulled out of the manual so each reads as the reference it is. Every PDF in the package is now paginated with a cover, a page-numbered table of contents and running page numbers. And, quietly, MSVC's `/W4` warnings now fail the Windows build the way the other two toolchains already did, closing the last corner where a warning could slip through unseen.

0.4.0

0.4.0 is the boards release. Seventeen new S-100 cards join the backplane — enough that three whole new machine families (Cromemco, SD Systems, Tarbell) boot alongside the MITS originals for the first time — disks and cassettes learn to be built and formatted by the guest instead of only read, and the debugger, the monitor prompt and the video window all got noticeably sharper along the way.

Three new machine families join the MITS originals

Cromemco arrives as a full boot chain: the **16FDC/64FDC** floppy controller (Western Digital FD1793, a TMS5501 console UART, and the RDOS boot PROM in one card) boots **CDOS**, and the **Dazzler** paints S-100's first color graphics out of a framebuffer in main RAM, with the **D+7A** analog/parallel card and a **JS-1 joystick** (real gamepad or keyboard) to drive games on it. **SD Systems** shows up as a matching trio — the **SBC-100/200** single-board Z80 (an 8251 console that auto-bauds to your terminal, and later a full interrupt-driven CP/M boot with the onboard PROM switching itself out), the **VersaFloppy** WD177x controller booting SDOS, and the **VDB-8024** 80×24 video terminal card. The **Tarbell #1011** and **#2022** floppy controllers boot CP/M entirely on their own, no monitor involved, straight off their own boot PROM. On the MITS side, the **88-HDSK** boots CP/M off a hard disk, **88-PIO/88-4PIO** add parallel I/O, **88-LPC** drives a real line printer, **88-UIO** combines a serial port and a cassette deck on one card, and the **8800b Turnkey Module** brings up a front-panel-less Altair with its boot PROM jammed onto the bus at RESET. Two boards round it out: the **PMMI MM-103**, the first S-100 modem (dial and answer a real

phone line over TCP), and **usio**, a serial card you describe by strap instead of one we picked, with built-in profiles for the Cromemco TU-ART, IMSAI SIO-2 and CompuPro serial channels. A new built-in ROM, **ROM BASIC**, boots Altair BASIC 4.1 straight out of PROM with the full 48K free underneath it.

Disks and tapes the guest can build, not just read

A hard-sector disk or a cassette tape can now be created **blank** and brought to life by the guest's own software: `MOUNT … CREATE` writes an empty image, and it grows one sector — or one byte of tape — at a time as the guest's FORMAT program defines it, exactly as unformatted media behaves on real hardware. Soft-sector disks caught up too: the Tarbell and VersaFloppy controllers now honor the WD177x Write Track command, so CP/M's `FORMAT` and SDOS's disk formatter lay down a bootable disk from nothing, across every geometry those cards support. `MOUNT` also now reads **ImageDisk (`.IMD`)** files directly, converting the interleaved container to the raw sector-linear image the disk boards expect. On the cassette side, both cassette decks show where the head is (`mm:ss / total (percent)`), a new `WIND` command moves it to a time so a multi-program tape is finally navigable, a `stop` mark parks playback at a boundary, and `EXTRACT` splits a multi-program recording into one file per program. Tapes the simulator writes now load on a real Sol-20, because the cassette modem's output is modeled the way the real hardware's clock-divider and filter actually shape a signal, not as a clean oscillator a real deck could never produce.

A debugger with sharper eyes

Cycle breakpoints (`BREAK MEM` / `BREAK IO`) now stop **before** the triggering instruction runs instead of after, so a breakpoint on a port read shows you the registers exactly as they were the instant the instruction fired — no port read, no byte written, nothing to unwind. `BREAK TAPE STOP` adds a third kind of breakpoint, alongside PC and bus-cycle stops, that halts the instant a cassette reaches its own auto-stop. `HISTORY` now defaults to a flight recorder of CPU instructions rather than raw bus cycles, and its bus view (`HISTORY BUS`) names which board drove and which answered every cycle it logs. `EDIT`, the byte-at-a-time memory editor, now assembles a full instruction where a byte would go. Loaded symbols make disassembly and single-stepping read by name instead of address, `SAVE` can write a disassembly or octal listing to a file, and the monitor can display and parse addresses, ports and bytes in authentic split **octal**, MITS's own notation. A new diagnostic-channel facility (`SET <id> DEBUG=`, `SHOW DEBUG`) lets individual boards and shared chips narrate what they're doing — a disk stepping heads, a UART taking a byte — each line stamped with the PC that caused it, aimed at the console, a file, or a `log` that tees your whole session transcript to disk. `SET BUS UNCLAIMED` catches a guest reaching for a board that was never fitted, and a bare `!` drops you to your host shell without losing the machine's state underneath it.

A prompt that helps you type

`Tab` now completes commands, board ids, unit names, property names and their legal values at the monitor prompt, reading the candidates straight off the running machine so a board you plug in is

completable immediately. The line editor grew word motion, `Home` / `End`, and kill-to-end-of-line, and command history now survives a restart, saved per project directory. Asking what you can build is one family of commands now — `SHOW BOARDS`, `SHOW BOARD <type>`, `SHOW MACHINES`, `SHOW MACHINE <name>` — with legal values listed under every property. A machine file can leave you a note: a `#>` comment line prints itself to the operator when the machine loads, the one or two sentences a config's author needs you to see before you start typing. And `HELP` now answers at whatever level you actually asked for, instead of always dropping you at the top.

The video window and the joysticks behind it

A video window now opens locked to its picture's aspect ratio and resizes proportionally from the first drag, with an even bezel on all four sides and its title bar reading "simulator stopped" whenever the guest is halted. How big it opens is a per-board `width` in pixels, so each card sizes itself off its own resolution; the Dazzler's tiny frame gets its own auto-scaling so it lands near the same size as a VDM-1's instead of a sixth of it. The D+7A now reports what's actually behind each joystick console — a named gamepad, the keyboard, or nothing — and a new `SHOW JOYSTICKS` lists every controller the host can see; two consoles default to two different gamepads automatically, so a pair of controllers just works with no configuration at all.

Save a machine, and pick it back up later

`SNAPSHOT <file>` writes the whole machine's state — every board's registers and RAM, the clock, and the CPU down to the microstate a register dump never shows — to a small, checksummed file, and `RESTORE <file>` loads it back into a machine of the same shape, refusing a corrupt or mismatched file before it touches a single byte of the machine you're running.

Building, driving and reading altairsim got easier

A clean clone now goes from nothing to a running binary with one command, `build.sh` or `build.bat`, with SDL3 staying entirely optional. Every warning the compiler can find now fails continuous integration outright, so nothing that used to be quiet creeps back in unnoticed, and a local sanitizer build is one flag away when a bug needs one. The AI-driving guide gained the step it was missing — how to actually register altairsim as an MCP server — plus a worked example where an assistant assembles a small buggy program, single-steps it to find the fault, and fixes it, entirely over the wire. And a font fix means copying a line out of any built PDF — the manual, an example's README, this changelog — now pastes back as the words it actually says, not text with spaces jammed into the middle of them.

0.3.0

0.3.0 adds no machines and no boards. It changes one thing, and it is the thing the manual has described all along: the copy you download now opens the video window.

The window the manual documents is finally in the package

Every release through 0.2.0 shipped a **headless** binary. SDL3 was not compiled into it, so the VDM-1 and Sol-20 windows the boards and configuring chapters describe at length could not open from anything you were handed — the machines ran, and drew nothing. `SHOW VERSION` said so in a row nobody had reason to read:

```
altairsim> SHOW VERSION
  altairsim  0.3.0
  video      SDL3 -- windowed        <- 0.2.0 and before, the download read: none -- headless
  commit     v0.3.0
  tree       clean
```

0.3.0 is the first release whose downloaded package reads `SDL3 -- windowed`. Run `altairsim sol20`, and a window opens. That is the headline, and most of the rest of this entry is how it was made true on every platform at once.

Nothing to install, on any of the four

The packages are built on the hardware they target now, rather than assembled by CI, and SDL3 is **linked statically into the binary**. So there is no `SDL3.dll` to sit beside the `.exe`, no `.framework`, no `libSDL3.so.0`, and nothing to install before the program runs — the claim altairsim has always made for itself is now true of its video too. On Windows the C runtime is static as well, so a clean machine that has never had a compiler on it runs the `.exe` with no Microsoft redistributable to chase.

The one visible cost is a single extra file in the archive — `LICENSE-SDL3`, SDL3's zlib licence, because SDL3's code now travels inside the binary and its licence travels with it.

macOS ships as two builds, each tested on its own hardware

0.2.0 shipped one *universal* macOS archive. 0.3.0 ships two — `altairsim-0.3.0-macos-arm64` and `altairsim-0.3.0-macos-x86_64` — and the reason is honesty, not size. The universal binary's Intel half had been built and tested by nobody, because the machine that produced it was Apple Silicon; the previous two releases said so in their own notes. Each of the two archives is now built **and** run on the architecture it targets, so the Intel download is exercised on an Intel Mac before it ships. Take the one that matches your Mac; `altairsim --version` and the download filename both name the architecture.

Windows, proven end to end

The Windows package is built with MSVC, links SDL3 and the C runtime statically, passes the full test suite on the machine that builds it, and opens a real VDM-1 window that a person has sat in front of. `dumpbin /dependents` on the shipped `.exe` shows system DLLs only — no `SDL3.dll`, no `VCRUNTIME140`. It is held to the same bar as the other three, and it is no longer an asterisk on the release.

The four downloads

Platform	File
macOS Apple Silicon	`altairsim-0.3.0-macos-arm64.tar.gz`
macOS Intel	`altairsim-0.3.0-macos-x86_64.tar.gz`
Linux x86_64	`altairsim-0.3.0-linux-x86_64.tar.gz`
Windows x86_64	`altairsim-0.3.0-windows-x86_64.zip`

Each holds the program, this changelog, the **User Manual**, `DRIVING-WITH-AI.md`, both licences, and `examples/` — four machines that boot, media included. Unzip it and run it; nothing needs fetching first.

What did not change

The simulator itself is byte-for-byte the machine 0.2.0 was — the same machines, the same boards, the same two CPU cores. The holes named in the manual's introduction are still holes: no snapshot, no replay, the six reserved monitor verbs still reserved, still no audio. Everything that moved is in the box the program arrives in.

0.2.0

The second release. It added machines and made the video window behave like a window — but note that the packages were still **headless** (see 0.3.0): everything below was true of a build made *with* SDL3, which is not what the archives carried until 0.3.0.

Three more monitors that boot from a bare command line

`amon`, `acuter` and `cdbl` became machines, so Martin Eberhard's Altair ROMs boot by name — nothing to fetch, nothing to mount:

```
altairsim amon          AMON 3.1 in a 4K EPROM at F000 -- a full-featured Altair monitor
altairsim acuter        ACUTER at F000 -- CUTER on a plain Altair, driving a terminal
altairsim cdbl          the default machine, with the Combo Disk Boot Loader in the socket
```

The ROM images shipped in 0.1.0 already; what was new is that each got a machine built around it — the whole distance between shipping an image and being able to run it. `hdbl` was deliberately left out: it boots an 88-HDSK hard disk, and there is no 88-HDSK board here.

The video window behaves like a window

- **It does not steal the keyboard when it opens.** The terminal keeps the keys while you type at the monitor prompt; `SET DISPLAY focus=on` hands them to the guest, stopping it hands them back.
- **It is named after the machine**, so `sol20` and `vdm1` are two windows you can tell apart.
- **It is sized to fit the screen it opened on**, and **arrows and HOME reach the guest**.
- **Closing it stops the guest**, instead of leaving a machine running with nothing to draw on.

Two of those were bugs worth naming: typed input could lag a whole frame, and the VDM-1 could repaint hundreds of times per emulated millisecond. Both gone. The VDM-1's cursor now blinks on the board's own oscillator — wall-clock time, as the hardware did.

Disk BASIC, and smaller things

- `examples/diskbasic` boots **Altair Disk BASIC 4.1** off a floppy, media included — a fourth worked example alongside CP/M, cassette BASIC and the Sol-20.
- A binary now names the commit it was built from: `SHOW VERSION` and `--version` carry it, so a report against a nightly or a CI artifact can be traced to the code that produced it.
- `writeprotect` is accepted wherever `readonly` is, in machine files and at `SET`.
- The manual and the program say **board**, not card.

0.1.0

The first release — a simulator for the MITS Altair 8800 and the S-100 bus, in C++20, with **nothing to fetch**: the TOML parser, the JSON encoder and the line editor are all in-tree, so a fresh clone builds with a C++20 compiler and CMake and no network.

Two validated CPU cores

Both cleared their gate before a single board was built on them, and for the release the exercisers were re-run on all three platforms — roughly 15 billion instructions each:

- **8080** — TST8080, 8080PRE, CPUTEST, 8080EXM
- **Z80** — ZEXDOC, ZEXALL

Boards and machines

88-2SIO · 88-SIO · 88-ACR · 88-DCDD · 88-MDS · 88-VI/RTC · 88-C700 · front panel · memory · 8080 CPU · Z80 CPU · VDM-1 · Processor Technology Sol-PC · Host Bridge (our own design).

CP/M 2.2 cold-boots from both 8-inch and 5.25-inch disks. MITS 4K and 8K BASIC and Programming System II load from cassette — including **real `.WAV` audio, played and recorded**, at measured CUTS/ACR parameters.

Debugging, and an assistant that can drive it

Breakpoints, watchpoints, tracepoints, conditional breaks (`BREAK … IF`), execution history, and symbolic reference loaded from `.PRN` and `.SYM` files — DDT and SID run under it, self-modifying RST 7 breakpoints and all. An **MCP server** lets an AI assistant drive a running guest.