# altairsim — MP/M II 2.1 on an Altair

2026-10-23 · 1cc5be2

MP/M II is the multi-user operating system of Digital Research. This machine runs it on an Altair with **88-DCDD** 8″ floppies, an **88-2SIO** with two consoles, and an **88-VI/RTC** for the clock tick. The memory is not bank-switched. A second machine file, `mpm2-ss1.toml`, has a CompuPro **System Support 1** in place of the 88-VI/RTC. The System Support 1 gives the tick, and its clock sets the date and time at boot.

DRI shipped MP/M II with no hardware code. Each computer maker wrote its own XIOS. The XIOS in this folder is for the Altair.

| File | What it is |
|---|---|
| `mpm2.toml` | The machine. |
| `MPM2.DSK` | The MP/M II system disk, on A:. The DBL PROM boots it into MP/M II. |
| `mpm2-ss1.toml` | The machine with a System Support 1 in place of the 88-VI/RTC. |
| `MPM2SS1.DSK` | The MP/M II system disk of `mpm2-ss1.toml`. |
| `MPMII_1.DSK` | The files of DRI disk 1, on B:. |
| `MPMII_2.DSK` | The files of DRI disk 2, on C:. |
| `build.toml` | The CP/M 2.2 machine with two drives that builds the system disk. |
| `CPM22.DSK` | A CP/M 2.2 disk that boots, with the tools for the build. |
| `BUILD.DSK` | The sources and the DRI files that GENSYS needs. |
| `RESXIOS.ASM` | The XIOS, for RMAC. |
| `SS1XIOS.ASM` | The XIOS of `mpm2-ss1.toml`, for RMAC. |
| `SS1TEST.ASM`, `SS1TEST.HEX` | A test of a real System Support 1 from the front panel, for ASM. It runs at 0100h with no operating system. |
| `LDRBIOS.ASM` | The loader BIOS, for ASM. |
| `BOOT.ASM` | The cold start loader on track 0, for ASM. |
| `AFORMAT.ASM` | The disk format program, for ASM. It runs under CP/M and MP/M II. |
| `make-mpm2-disk.py` | The script that does the steps of "Build the system disk" in the simulator. It built `MPM2.DSK` and `MPM2SS1.DSK`. |
| `MPM_II_System_Implementors_Guide_Aug82.md` | Notes on the *MP/M II System Implementor's Guide* of Digital Research. |
| `MPMII_1/`, `MPMII_2/` | The two DRI MP/M II 2.1 distribution disks, as ImageDisk files (`MPMII_1.IMD`, `MPMII_2.IMD`), and the files on each disk. |

The CP/NET server of the `cpnet` example is this MP/M II with a network line.

## Run it

1. Start the machine from this folder:

   ```
   cd examples/mpm2
   altairsim mpm2.toml
   ```

2. Wait for `0A>` on console 0. The loader prints the memory map first. The boot takes less than a second, or about 7 seconds at 2 MHz.

3. For console 1, type `telnet localhost 8820` in a second window. Console 1 prints `1A>`.

4. For drive D:, give the monitor a copy of a CP/M 2.2 disk for the 88-DCDD:

   ```
   MOUNT dsk0:drive3 data.dsk
   ```

The prompt shows the user number and the drive. MP/M II starts console 1 in user 1, so `1A>` is user 1 on drive A:. A file that console 0 makes is in user 0. Type `USER 0` on console 1 to see it.

Type a command after the prompt comes back. While a program runs, MP/M II reads the keys that come in and discards them.

This boot and these commands ran through `altairsim --mcp` on 2026-10-21:

```
MP/M II V2.1 Loader
Copyright (C) 1981, Digital Research
Nmb of consoles     =  2
Breakpoint RST #    =  6
Memory Segment Table:
SYSTEM  DAT  FE00H  0100H
...
MP/M II Sys  7600H  8A00H
Memseg  Usr  0000H  3B00H
Memseg  Usr  3B00H  3B00H

MP/M II V2.1
Copyright (C) 1982, Digital Research

0A>TOD
Mon 02/01/82 00:00:02
0A>TOD
Mon 02/01/82 00:00:03
```

## Format a disk

`AFORMAT` on `MPM2.DSK` formats a disk in drive A:, B:, C: or D:.

1. Type `AFORMAT D:` for drive D:.
2. Type `Y` at `Ready (Y/N)?`.
3. Wait for `Format another (Y/N)?`, then type `N`.

While `AFORMAT` formats a disk, the other console waits for each disk command. To format A:, put the new disk in A: before you type `Y`. At the end, `AFORMAT` prints `Put MP/M II disk in A:`. Put `MPM2.DSK` back in A:, then type `Y`. The other console waits for its disk commands until then. At the monitor, `MOUNT dsk0:drive0` changes the disk in A:.

## The machine

`mpm2.toml` is the default Altair with these changes:

| Board | Change |
|---|---|
| `cpu0` | No change: the processor runs flat out. The RTC keeps real time at every processor speed. For the 2 MHz clock of a real Altair, remove the `#` from the three `cpu0` lines in `mpm2.toml`, or type `SET cpu0 clock_hz=2000000` at the monitor. The 88-DCDD code counts cycles, so the disks work at 2 MHz or flat out only. |
| `mem0` | RAM to FEFFh. The DBL boot PROM stays at FF00h. |
| `dsk0` | `MPM2.DSK` in drive 0 (A:), `MPMII_1.DSK` in drive 1 (B:) and `MPMII_2.DSK` in drive 2 (C:). Drive 3 (D:) is empty. |
| `sio0` | Port B connected to `telnet:8820`, for console 1. |
| `vi0` | An 88-VI/RTC. The RTC counts the 60 Hz line, divides by 1, and interrupts on VI5. |

GENSYS gives the user memory two segments of about 15K each, 0000h–3AFFh and 3B00h–75FFh. Each console can run PIP, ED or ASM while the other console runs a command. A `.COM` file runs only in the segment at 0000h. SDIR needs 19K, so SDIR does not load. Use DIR.

The `.PRL` commands have the system attribute, so every user number can run them.

## The XIOS

| File | What it is |
|---|---|
| `RESXIOS.ASM` | Console 0 is 2SIO port A (10h/11h), and console 1 is port B (12h/13h). The 88-VI/RTC gives 60 ticks a second on RST 5. The disk code reads and writes the CP/M 2.2 disk format of the 88-DCDD. |
| `LDRBIOS.ASM` | The loader BIOS, at 1700h. It reads drive A: one track at a time, into a track buffer at 2000h. DDT puts it into `MPMLDR.COM`. |
| `BOOT.ASM` | The cold start loader. It reads `MPMLDR.COM` from the system tracks to 0100h and starts it. |

`RESXIOS.ASM` and `LDRBIOS.ASM` start from DRI's files of the same name on disk 2. `BOOT.ASM` starts from the cold boot loader of the Altair CP/M. Their 88-DCDD code comes from the CP/M 2.2b v2.3 BIOS and boot loader of Mike Douglas, for the Altair 8″ floppy. The comment at the top of each file lists what is different for MP/M. For example, each write of the XIOS goes to the disk before `WRITE` returns.

`AFORMAT.ASM` is AFORMAT 1.11 of Martin Eberhard, from https://deramp.com/downloads/mfe_archive/010-S100%20Computers%20and%20Boards/00-MITS/41-Martins%20Altair%20Software/AFORMAT/. Version 1.12 also runs under MP/M II. The comment at the top of the file lists the changes. `AFORMAT` reads and writes the 88-DCDD directly, without the XIOS. Because of that, the XIOS selects the drive again before each write, and a seek that reaches track 0 too soon starts again from track 0.

## The System Support 1 machine

`mpm2-ss1.toml` is `mpm2.toml` with two changes:

| Board | Change |
|---|---|
| `dsk0` | `MPM2SS1.DSK` in drive 0 (A:). |
| `vi0` | Removed. |
| `ss0` | A CompuPro System Support 1 at 50h. |

Start it from this folder:

```
cd examples/mpm2
altairsim mpm2-ss1.toml
```

Do not add an 88-VI/RTC to this machine. The 8259A interrupt controllers of the System Support 1 also read the VI lines, and the two boards answer the same interrupt acknowledge.

On a real Altair, set the System Support 1 as shown in the tables below. The settings come from the System Support 1 manual (Rev G). ON matches a 0 bit on the address line, and OFF matches a 1 bit.

Switch 1 is at the right edge of the board.

| Position | Label | Setting | Function |
|---|---|---|---|
| 1 | W8 | OFF | No wait states. |
| 2 | W4 | OFF | |
| 3 | W2 | OFF | |
| 4 | W1 | OFF | Set ON for a processor of 4 MHz or more. |
| 5 | RDI | ON | Disables the memory on the board. This machine does not use it. |
| 6 | XA | ON | No extended addressing. |
| 7 | PHD | ON | PHANTOM* disables the memory on the board. |
| 8 | PHE | OFF | |

Switch 2 is between U32 and U33. It sets the extended address of the memory on the board. Set positions 1–8 OFF.

Switch 3 is between U35 and U36.

| Position | Setting | Function |
|---|---|---|
| 1–4 | OFF | The address of the memory on the board. This machine does not use it. |
| 5 | ON | I/O address bit 7 = 0 |
| 6 | OFF | I/O address bit 6 = 1 |
| 7 | ON | I/O address bit 5 = 0 |
| 8 | OFF | I/O address bit 4 = 1. The I/O ports are 50h–5Fh. |

| Jumper | Setting | Function |
|---|---|---|
| J13 (lower right corner) | Shorting plug on Z–C, right of center | An 8080 or Z80 interrupt acknowledge, with PHANTOM*. |
| J8 (lower left, between U30 and U31) | A DIP shunt, wired as in the table below | The standard interrupts. |
| J7 | Open | |
| J4 | No cut traces | Each timer counts the 2 MHz clock of bus pin 49. |
| J5 | No change, and no crystal at X1 | The 2 MHz clock comes from bus pin 49. |
| J6 | Open | The board has no math chip, or the math chip does not interrupt. |
| J3 | The battery cable, red wire to the left | The MSM5832 keeps the date and time with the power off. |
| J2 | A DIP shunt | The serial channel as a slave. This machine does not use the serial channel. |

A DIP shunt connects each pin of J8 to the pin across from it: 1 to 16, 2 to 15, and so on to 8 to 9. Pins 1–8 are the sources and pins 9–16 are the inputs of the slave 8259A:

| J8 pins | Source | Slave 8259A input |
|---|---|---|
| 8–9 | S-100 VI7* | IR0 |
| 7–10 | Timer 0 output | IR1. This is the tick of MP/M II. |
| 6–11 | Timer 1 output | IR2 |
| 5–12 | Timer 2 output | IR3 |
| 4–13 | Math chip SVRQ | IR4 |
| 3–14 | Math chip END | IR5 |
| 2–15 | 2651 TxRDY | IR6 |
| 1–16 | 2651 RxRDY | IR7 |

The only connection that MP/M II needs is pins 7–10. The XIOS masks the other inputs. The board connects the rest without a jumper: VI0*–VI6* go to IR0–IR6 of the master 8259A, and the slave goes to IR7 of the master. If you use a DIP header in place of the shunt, connect pins 7 and 10.

Set the MSM5832 to the date and time with a different program. The XIOS reads the clock but does not set it.

An 8080 reads bytes 2 and 3 of the `CALL` as memory reads. The System Support 1 asserts PHANTOM* (bus pin 67) during those reads and puts the address on the bus. Set every memory board, and the board of the DBL PROM, to turn off when PHANTOM* is low. If a memory board answers, the processor reads the address from memory, and the tick goes to a wrong address. Make sure that the processor board puts the 2 MHz clock on bus pin 49. The count of timer 0 is correct only for 2 MHz.

The handler's correction is for a 2 MHz 8080. When the simulator runs flat out, the 108 counts take no time, and the time of day gains about 0.3%. At 2 MHz, 60 seconds of `TOD` took 119,977,571 T-states in the simulator, 0.02% from 2 MHz. That difference is less than one tick and one second of `TOD`.

`SS1XIOS.ASM` is `RESXIOS.ASM` with a different clock:

- Timer 0 of the 8253 counts the 2 MHz bus clock in mode 0. It interrupts after 33333 counts, for 60 ticks a second.
- Timer 0 interrupts on IR1 of the slave 8259A. The slave interrupts through IR7 of the master 8259A. The 8259A sends a `CALL 0024H`, not an RST. `SYSTEMINIT` puts a jump to the interrupt handler at 0024h.
- The 8259As are level-triggered, as in the sample program of the System Support 1 manual. In mode 0, the timer output stays high until the handler loads the next count. Because of that, a tick that comes while interrupts are disabled waits until the next `EI`.
- After the timer interrupts, it continues to count down. The handler reads how late it is and loads that much less. The handler also subtracts the 108 counts of its own code on a 2 MHz 8080. Because of that, the time of day does not drift on a 2 MHz Altair.
- The interrupt handler sends an end of interrupt to the slave and to the master.
- `SYSTEMINIT` reads the MSM5832 clock and sets the date and time of MP/M II. In the simulator, the clock reads the time of the host.

`SS1XIOS.ASM` is one page longer than `RESXIOS.ASM`. The operating system starts at 7500h, and each console has a segment of 3Ah pages, 0000h–39FFh and 3A00h–73FFh.

This boot and these commands ran through `altairsim --mcp` on 2026-10-22:

```
MP/M II Sys  7500H  8B00H
Memseg  Usr  0000H  3A00H
Memseg  Usr  3A00H  3A00H

MP/M II V2.1
Copyright (C) 1982, Digital Research

0A>TOD
Thu 10/22/<6 15:53:07
```

The date is correct. `TOD` shows the year 2026 as `<6`. DRI's `TOD` prints two digits for the year, and counts the years from 1978.

### Test a real System Support 1

`SS1TEST` tests the parts of a System Support 1 that `SS1XIOS.ASM` uses. It needs no console and no operating system. It writes a jump at 0024h and uses memory from 0100h to 028Dh.

1. Set the System Support 1 as shown in the tables above.
2. Load `SS1TEST.HEX` into memory, or deposit its bytes from the front panel.
3. Run from 0100h.
4. When the HLTA light comes on, stop the machine.
5. Examine 0103h. The data lights show the result.

| Result at 0103h | Meaning |
|---|---|
| 00 | All the tests passed. The program stops at the HLT at 023Ch. |
| 01 | The clock digits are not valid. Check the I/O address of the board (switch 3). |
| 02 | Timer 0 does not count. Check for a clock on bus pin 49. |
| 03 | Timer 0 counts at the wrong rate. Check for 2 MHz on bus pin 49. |
| 04 | Slave IR1 is high while the output of timer 0 is low. Check J8, pins 7–10. |
| 05 | Slave IR1 does not go high at the end of the count. Check J8, pins 7–10. |
| 06 | The master 8259A does not see the interrupt of the slave. |
| 07 | Slave IR1 stays high after the program loads timer 0 again. |
| 08 | No interrupt arrives. Check bus pin 73 and the processor board. |
| 09 | An 8259A stays in service after the end of interrupt. |
| FF | The program did not start. |

A failed test stops at the HLT at 0240h. If the machine does not stop, look at 0103h too. It holds the number of the test that was running. A result of 08 when the machine does not stop means the `CALL` went to a wrong address. Check J13 (Z–C), and check that every memory board turns off when PHANTOM* is low.

The program also leaves these values. The low byte comes first in each 16-bit value.

| Address | Value |
|---|---|
| 0104h–0105h | The counts of timer 0 in a delay of test 3. A 2 MHz Altair gives about 5E1Fh. |
| 0106h–0107h | The loops until slave IR1 went high in test 5. A 2 MHz Altair gives about 028Dh. |
| 0108h–0114h | The 13 digits of the MSM5832, ones digit first: seconds, minutes, hours, day of the week, day, month, year. |

The tests ran in the simulator on 2026-10-22. On a 2 MHz machine, the result was 00. With no System Support 1, the result was 01. With the processor flat out, the result was 03.

To assemble the program, put `SS1TEST.ASM` on a CP/M disk and type `ASM SS1TEST`.

## The boot

`MPM2.DSK` boots as section 1.5 of the *MP/M II System Implementor's Guide* describes. No CP/M runs. The system tracks hold the cold start loader and `MPMLDR.COM`:

| Track | Sectors | Memory | What it is |
|---|---|---|---|
| 0 | 0 and 2 | 0000h–00FFh | `BOOT`. The DBL PROM loads it and jumps to 0000h. |
| 0 | 3–31 | 0100h–0F7Fh | `MPMLDR.COM`: the loader, the loader BDOS and the loader BIOS |
| 1 | 0–22 | 0F80h–1AFFh | `MPMLDR.COM`, continued |

1. The DBL PROM loads `BOOT` from track 0.
2. `BOOT` reads `MPMLDR.COM` to 0100h and jumps to it.
3. `MPMLDR` reads `MPM.SYS` from drive A: and prints the memory map. It then starts MP/M II.

`BOOT` leaves the blank FCB at 005Ch that the CCP of CP/M gives a command with no parameters. `MPMLDR` then loads `MPM.SYS`. If `MPMLDR` cannot read `MPM.SYS`, it prints an error and jumps to 0000h. `BOOT` puts `DI` and `HLT` there, so the processor stops.

The loader BIOS reads a whole track into a track buffer at 2000h–311Fh, as the XIOS does. On a data track, the skew puts each record 8 sectors after the record before it. A read of one sector at a time gets only 4 records in one turn of the disk, and needs 8 turns for a track. A read of the whole track takes about one turn. At 2 MHz in the simulator, `MPMLDR` loads `MPM.SYS` in 3.8 seconds with the track buffer, and took 14.4 seconds with one sector at a time. `BOOT` uses the same memory for its own track buffer. `MPMLDR` does not use the memory from 1B00h to the bottom of MP/M II.

## The disks and the file system

MP/M II uses the file system of CP/M 2.2. The XIOS uses the disk layout of the Altair CP/M disks: 77 tracks of 32 sectors, two system tracks, 2K blocks and 64 directory entries. Only the system tracks of `MPM2.DSK` are different from a CP/M disk. Because of that, drive D: of `mpm2.toml` takes a CP/M 2.2 data disk, and CP/M 2.2 can read the files of the three disks. `MPM2.DSK` does not boot CP/M.

MP/M II can add three things to a drive with `SET`. CP/M 2.2 has none of them:

| Feature | Command | Kept in |
|---|---|---|
| A directory label | `SET [NAME=label]` | A label entry in the directory |
| Passwords | `SET [PASSWORD=pw]`, `SET file [PASSWORD=pw]` | The label entry, and an XFCB for each file |
| Time stamps | `SET [CREATE=ON,UPDATE=ON]` or `SET [ACCESS=ON,UPDATE=ON]` | The label entry, and an XFCB for each file |

An XFCB is an extra directory entry for one file. CP/M 2.2 does not use the label entry or the XFCBs, and it does not check the passwords. `MPM2.DSK` has no label.

### Time stamps

Time stamps work on this machine. Each stamp is the date and time of `TOD` when MP/M II makes or changes the file. On 2026-10-21, these commands put create and update stamps into an XFCB of `COPY.COM` on a copy of `MPM2.DSK`:

```
0A>TOD 10/21/86 08:15:00
Strike key to set time

Tue 10/21/86 08:15:00
0A>SET [CREATE=ON,UPDATE=ON]
...
0A>PIP COPY.COM=DDT.COM
0A>SHOW [LABEL]
...
Label   .       off     on      on      on     10/21/86 08:15  10/21/86 08:15
```

Know these limits:

- Set the date with `TOD` after each boot. The 88-VI/RTC gives the tick, and it does not keep the date. MP/M II starts at `Mon 02/01/82 00:00:00`. On `mpm2-ss1.toml`, the clock of the System Support 1 sets the date at boot.

- Give a year from 1978 to 1999. `TOD` takes two digits for the year, and refuses `26` for 2026. `86` works.
- Use an altairsim later than 1.3.4. In 1.3.4 and earlier, the RTC runs fast when the processor runs flat out, and the clock of MP/M II gains about 100 seconds each second. With those versions, set the processor to 2 MHz (see "The machine").
- `SDIR` shows the stamps of the files, and `SDIR` does not fit in a user segment. `SHOW` shows the stamps of the label only. On the host, the stamps are bytes 24–31 of each XFCB.

## The DRI disks

`MPM2.DSK` is built from the DRI MP/M II 2.1 end-user disks. They are two 8″ single-sided single-density disks in IBM 3740 format, archived as ImageDisk files. Each file is in a folder of its own, with the files that are on the disk:

| | **Disk 1** | **Disk 2** |
|---|---|---|
| Label | END USER DISKETTE, Program MP/M II, Version 2.1 | the same |
| Serial | MP2-000-01384 | MP2-000-01384 |
| File | `MPMII_1/MPMII_1.IMD`, 170,262 bytes | `MPMII_2/MPMII_2.IMD`, 172,162 bytes |
| Files on the disk | 44, in `MPMII_1/` | 28, in `MPMII_2/` |
| Altair disk | `MPMII_1.DSK` | `MPMII_2.DSK` |
| SHA-256 | `1432d5ea…d951a736` | `1adcc657…32c6650c` |

Both files come in `MPMIIOEM.ZIP` (SHA-256 `fdcd2294…77f64a67`). Patrick Linstruth supplied it on 2026-10-18. The same two images, byte for byte, are in https://github.com/avwohl/mpm2/tree/main/mpm2_external.

Disk 1 has the operating system and the commands. `MPM2.DSK` holds every file of disk 1. Disk 2 has the tools (`RMAC`, `LINK`) and DRI's sample sources for a new XIOS.

`MPMII_1.DSK` and `MPMII_2.DSK` hold the same files as `MPMII_1.IMD` and `MPMII_2.IMD`, on 88-DCDD disks in the format of the Altair CP/M. `mpm2.toml` mounts them on B: and C:, so MP/M II can read the tools and the sources. The two disks do not boot.

### Make `.dsk` images of the DRI disks

altairsim converts an `.imd` file to a raw `.dsk` image when it mounts it. Each `.dsk` is 256,256 bytes: 77 tracks of 26 sectors of 128 bytes. A Tarbell controller reads it. The 88-DCDD of `mpm2.toml` uses a different format, and cannot read it.

1. Start the Tarbell machine from this folder, with the two `MOUNT` commands:

```
cd examples/mpm2
altairsim tarbell -x "MOUNT fdc0:drive1 MPMII_1/MPMII_1.IMD" \
    -x "MOUNT fdc0:drive2 MPMII_2/MPMII_2.IMD"
```

2. Read the output. Each `MOUNT` prints the comment line of its disk, for example `DRI MP/M II  Disk 1 of 2`. `MPMII_1/MPMII_1.dsk` and `MPMII_2/MPMII_2.dsk` are now beside the `.IMD` files.

3. Type `QUIT` at the `altairsim>` prompt.

The `.imd` file stays as it was. A `.dsk` image does not keep the comment line of the `.imd`.

## Build the system disk

`MPM2.DSK` is built under CP/M 2.2, as sections 1.2 and 1.5 of the *MP/M II System Implementor's Guide* describe. The build needs an Altair with two 88-DCDD drives and 56K of RAM, and these three disks:

| Disk | What is on it |
|---|---|
| `CPM22.DSK` | The 56K CP/M 2.2b v2.3 of Mike Douglas for the 88-DCDD, from altairsim 1.3.4. It has PIP, ED, ASM, DDT, LOAD, STAT, SYSGEN and the other CP/M tools. It also has `AFORMAT.COM` 1.12, and `RMAC.COM` and `LINK.COM` from DRI disk 2. The games, `LS.COM`, and the host-bridge programs `R`, `W` and `HDIR` are removed. |
| `BUILD.DSK` | `RESXIOS.ASM`, `SS1XIOS.ASM`, `LDRBIOS.ASM`, `BOOT.ASM` and `AFORMAT.ASM`. From DRI disk 1: `GENSYS.COM`, `MPMLDR.COM`, and the `.SPR`, `.RSP` and `.BRS` files. |
| A copy of `MPMII_1.DSK` | The new system disk. It has every file of DRI disk 1, and its system tracks are empty. |

The build writes on all three disks. Use copies of them. On a real Altair, write each image to a disk. In the simulator, copy this folder and start the build machine:

```
cp -R examples/mpm2 ~/mpm2-build
cd ~/mpm2-build
cp MPMII_1.DSK NEW.DSK
altairsim build.toml
```

CP/M boots from `CPM22.DSK` in A:, with `BUILD.DSK` in B:.

### Assemble the XIOS and the loaders

The tools are on A:, and the files that they make go to B:. `$PZ SZ` for RMAC and `.BBZ` for ASM make no listing, so that every file fits on `BUILD.DSK`.

1. Type `B:`.
2. Type `A:RMAC RESXIOS $PZ SZ`. RMAC makes `RESXIOS.REL`.
3. Type `A:LINK RESXIOS[OS]`. LINK makes `RESXIOS.SPR`.
4. Type `A:ASM LDRBIOS.BBZ`. ASM makes `LDRBIOS.HEX`.
5. Type `A:DDT MPMLDR.COM`.
6. At the `-` prompt of DDT, type `ILDRBIOS.HEX`, then `R`. DDT puts the loader BIOS at 1700h, on top of the loader BIOS of DRI.
7. Type `G0`. DDT stops, and the `B>` prompt comes back.
8. Type `SAVE 26 MPMLDR.COM`. CP/M saves 0100h–1AFFh.
9. Type `A:ASM BOOT.BBZ`. ASM makes `BOOT.HEX`.

Steps 5 to 8 are section 1.2 of the guide.

### Make MPM.SYS

1. Type `GENSYS`.
2. Give the answers of the table below. A `#` makes a number decimal. A number without it is hex.
3. Wait for `** GENSYS DONE **`. GENSYS writes `MPM.SYS` on B:.

| GENSYS asks | Type |
|---|---|
| `Top page of operating system (FF) ?` | `FE` |
| `Number of TMPs (system consoles) (#4) ?` | `#2` |
| `Number of Printers (#1) ?` | `#1` |
| `Breakpoint RST (06) ?` | `06` |
| `Enable Compatibility Attributes (N) ?` | `N` |
| `Add system call user stacks (Y) ?` | `Y` |
| `Z80 CPU (Y) ?` | `N` |
| `Number of ticks/second (#60) ?` | `#60` |
| `System Drive (A:) ?` | `A:` |
| `Temporary file drive (A:) ?` | `A:` |
| `Maximum locked records/process (#16) ?` | `#16` |
| `Total locked records/system (#32) ?` | `#32` |
| `Maximum open files/process (#16) ?` | `#16` |
| `Total open files/system (#32) ?` | `#32` |
| `Bank switched memory (Y) ?` | `N` |
| `Number of user memory segments (#3) ?` | `#2` |
| `Dayfile logging at console (Y) ?` | `N` |
| `Accept new system data page entries (Y) ?` | `Y` |
| `ABORT   RSP (N) ?` | `Y` |
| `MPMSTAT RSP (N) ?` | `Y` |
| `SCHED   RSP (N) ?` | `N` |
| `SPOOL   RSP (N) ?` | `N` |
| `Base,size,attrib (76,8A,80) ?` | Press `RETURN` |
| `Base,size,attrib (00,C0,00) ?`, the first time | `00,3B,00` |
| `Base,size,attrib (00,C0,00) ?`, the second time | `3B,3B,00` |
| `Accept new memory segment table entries (Y) ?` | `Y` |

The top page is FE, because the DBL PROM is at FF00h. MP/M II takes RST 6 for its debugger, because RST 5 is the tick of the 88-VI/RTC. The machine has no printer, so SPOOL is not added. "The machine" says why the user memory has two segments of 3Bh pages.

### Make AFORMAT.COM

1. Type `A:ASM AFORMAT.BBZ`. ASM makes `AFORMAT.HEX`.
2. Type `A:LOAD AFORMAT`. LOAD makes `AFORMAT.COM`.

### Copy the new files to A:

B: gets a different disk next, so the new files go to A: first.

1. Type `A:PIP A:=MPM.SYS[V]`.
2. Type `A:PIP A:=MPMLDR.COM[V]`.
3. Type `A:PIP A:=BOOT.HEX[V]`.
4. Type `A:PIP A:=AFORMAT.COM[V]`.
5. Type `A:`.

### Make the new system disk

1. Put the copy of `MPMII_1.DSK` in B:. In the simulator, press `Ctrl-E`, type `MOUNT dsk0:drive1 NEW.DSK`, and type `RUN`.
2. Press `Ctrl-C`. CP/M reads the directory of the new disk. Without `Ctrl-C`, CP/M does not write on the new disk.
3. Type `PIP B:=MPM.SYS[V]`.
4. Type `PIP B:=MPMLDR.COM[V]`. The new `MPMLDR.COM` replaces the one of DRI.
5. Type `PIP B:=AFORMAT.COM[V]`.
6. Type `STAT B:*.PRL $SYS`. Each `.PRL` file gets the system attribute, so every user number can run it.

### Write the system tracks

DDT puts `BOOT` and `MPMLDR.COM` into the memory image of SYSGEN, and SYSGEN writes them to the system tracks of B:. This is section 1.5.1 of the guide, for the SYSGEN of the Altair CP/M. Its image starts at 0900h, one 128-byte sector after another from track 0 sector 0. The DBL PROM reads `BOOT` from sectors 0 and 2, and `MPMLDR` starts at sector 3 (0A80h).

1. Type `DDT MPMLDR.COM`.

2. At the `-` prompt, type these commands, one at a time:

```
M100,1AFF,3000
M3000,49FF,A80
IBOOT.HEX
R900
M980,9FF,A00
G0
```

   The first two `M` commands move `MPMLDR` to 0A80h, through 3000h, because the two places overlap. `IBOOT.HEX` and `R900` read `BOOT` to 0900h. The last `M` puts the second half of `BOOT` in sector 2.

3. Type `SYSGEN`.

4. At `Source drive name (or RETURN to skip)`, press `RETURN`.

5. At `Destination drive name (or RETURN to reboot)`, type `B`.

6. At `Destination on B, then type RETURN`, press `RETURN`. SYSGEN prints `Function complete`.

7. At `Destination drive name (or RETURN to reboot)`, press `RETURN`. CP/M starts again from A:.

### Build the System Support 1 disk

The steps are the same, with these changes:

1. Before you assemble, type `ERA B:RESXIOS.ASM`. `BUILD.DSK` has no space for the files of the build and both XIOS sources.
2. Type `A:RMAC SS1XIOS $PZ SZ`, not `A:RMAC RESXIOS $PZ SZ`.
3. Type `A:LINK RESXIOS=SS1XIOS[OS]`, not `A:LINK RESXIOS[OS]`. GENSYS reads the XIOS from `RESXIOS.SPR`.
4. In GENSYS, give `00,3A,00` and `3A,3A,00` for the two user segments.

### Boot the new disk

The new disk has the same files as `MPM2.DSK`, and the same `BOOT` and `MPMLDR` on its system tracks.

1. On a real Altair, put the new disk in A: and start the DBL PROM at FF00h. The machine needs RAM to FEFFh and an 88-VI/RTC (see "The machine"). The System Support 1 disk needs a System Support 1 at 50h in place of the 88-VI/RTC.

2. In the simulator, type `QUIT` at the monitor. At the host prompt, type these commands:

```
cp NEW.DSK MPM2.DSK
altairsim mpm2.toml
```

## Rebuild the disk with the script

`make-mpm2-disk.py` types the commands of "Build the system disk" through `altairsim --mcp`, on copies of the disks of this folder. It checks what CP/M prints after each command. At the end, it checks the system tracks and `MPM.SYS` of the new disk, and boots the disk into MP/M II.

The script needs Python 3, the `altairsim` on your PATH, and cpmtools with the `altair` disk format. It reads the new disk with `cpmcp`. The build of 2026-10-21 used altairsim 1.3.4.

Run the script from this folder, with the name of the new disk:

```
./make-mpm2-disk.py MPM2.DSK
```

For the System Support 1 disk, add `--xios SS1XIOS`:

```
./make-mpm2-disk.py --xios SS1XIOS MPM2SS1.DSK
```

The script builds from the sources on `BUILD.DSK`. The `.ASM` files of this folder are copies of them. After you change a source in this folder, put the new file on `BUILD.DSK`. For example, for `RESXIOS.ASM`:

```
cpmrm -f altair BUILD.DSK 0:resxios.asm
cpmcp -f altair BUILD.DSK RESXIOS.ASM 0:
```

After you change a source on `BUILD.DSK` with ED, copy it back with `cpmcp -f altair BUILD.DSK 0:resxios.asm RESXIOS.ASM`.

The script prints each step and the memory map that GENSYS makes. A build from the same disks gives the same bytes. The SHA-256 of `MPM2.DSK` is `f31f5872…fca96172`, and the SHA-256 of `MPM2SS1.DSK` is `9a4853fc…67b55159`.

## Read more

- The *MP/M II System Implementor's Guide* of Digital Research (August 1982), for how to write an XIOS and build a system.
- The README of the `cpnet` example, for CP/NET 1.2.