

altairsim — The tools

2026-10-18 · cea34cb

This folder has two kinds of scripts:

- **The watch scripts** show the console of a machine in a second window.
- **The LBR tools** copy the files out of a CP/M library file, and decode crunched and squeezed files.

The watch scripts

Two scripts that show the console of the machine in a second window, while an AI assistant controls the machine. Start the simulator with a mirror:

```
altairsim examples/cpm/cpm22-buffered.toml --mcp --mirror socket:2323
```

Then run one script in a second window:

Script	Runs on	Needs
mirror-watch.sh	macOS, Linux	bash only
mirror-watch.ps1	Windows	PowerShell only

Each script connects to the mirror itself. You do not need `nc` or `telnet`.

A script waits if the mirror is not there yet. It shows the output while it is connected. When the simulator stops, the script waits, and it connects again when the simulator starts again. You can leave the window open for a full session.

Press Ctrl-C to stop a script.

On macOS and Linux

```
tools/mirror-watch.sh
```

On Windows

Windows does not run a PowerShell script file by default. There are two ways to run this one.

One time. This command changes no setting:

```
powershell -ExecutionPolicy Bypass -File tools\mirror-watch.ps1
```

Each time, for your account. Run this command once:

```
Set-ExecutionPolicy -Scope CurrentUser RemoteSigned
```

Windows marks a file that came from a web browser, and the files of a zip that came from a web browser. Remove the mark once:

```
Unblock-File tools\mirror-watch.ps1
```

After that, `tools\mirror-watch.ps1` runs directly.

Arguments

The two scripts use the same arguments:

Arguments	Watches
(none)	localhost:2323
altairsim	localhost:2323
port	localhost:port
host port	host:port , for a mirror on a different computer

Port 2323 is the port that the manual uses in its examples. If you start the simulator with a different port, give the same port to the script.

Limits

- **The scripts only watch.** They do not send the keys that you type. To type on a mirror, use `nc localhost 2323`.
- **One watcher at a time.** The mirror accepts one connection. Open only one watch window for each mirror.
- **The mirror accepts a watcher only while the machine runs.** A window can show `Connected` and nothing more until the assistant runs the machine. If the window stays empty while the machine runs, look for a second watcher that has the mirror.

Credit

The scripts were written by trgeuy for serial-mcp-server (<https://github.com/trgeuy/serial-mcp-server>). They have the MIT license. The license text is in `LICENSE-mirror-watch.txt`.

The LBR tools

Much CP/M software is kept in library files, with the extension `.LBR`. A library holds a set of files, as a zip file does. The files in a library are often compressed, and the middle letter of the extension shows how:

- A **crunched** file has a `Z`. For example, `SUPSUB13.ASM` is stored as `SUPSUB13.AZM`.
- A **squeezed** file has a `Q`. For example, `EX15.ASM` is stored as `EX15.AQM`.

A short extension gets the letter at its end. For example, `READ.ME` is stored as `READ.MQ`.

Script	Does
<code>lbr-extract.py</code>	Lists the files in a library, or copies them out.
<code>uncrunch.py</code>	Decodes a crunched file, and writes it with its original name.
<code>unsqueeze.py</code>	Decodes a squeezed file, and writes it with its original name.

The scripts need Python 3. They run on macOS, Linux and Windows.

Copy the files out of a library

1. List the files in the library:

```
tools/lbr-extract.py -l SUPSUB13.LBR
```

```
SUPSUB13.AZM  10880  CRC ok
SUPSUB13.CZM   1920  CRC ok
SUPSUB13.DZC   3072  CRC ok
```

2. Copy the files into a folder. `-d` gives the folder. Without `-d`, the files go into the current folder:

```
tools/lbr-extract.py -d supsub SUPSUB13.LBR
```

3. Decode the crunched files:

```
tools/uncrunch.py supsub/SUPSUB13.AZM supsub/SUPSUB13.CZM supsub/SUPSUB13.DZC
```

```
supsub/SUPSUB13.AZM -> supsub/SUPSUB13.ASM  
supsub/SUPSUB13.CZM -> supsub/SUPSUB13.COM  
supsub/SUPSUB13.DZC -> supsub/SUPSUB13.DOC
```

Decode a squeezed file in the same way, with `unsqueeze.py` :

```
tools/unsqueeze.py ex15/EX15.AQM ex15/READ.MQ
```

```
ex15/EX15.AQM -> ex15/EX15.ASM  
ex15/READ.MQ -> ex15/READ.ME
```

If you give a file to the wrong script, the script tells you which script decodes it.

To copy only some files, put their names after the library name: `tools/lbr-extract.py SUPSUB13.LBR SUPSUB13.DZC` . `uncrunch.py` and `unsqueeze.py` write each file into the folder of the compressed file. `-d` gives a different folder.

On Windows, start each script with `py` :

```
py tools\lbr-extract.py -l SUPSUB13.LBR
```

A script replaces a file that has the same name.

Checks

Each member of a library has a CRC, and each crunched or squeezed file has a checksum. The scripts check them. If a check fails, the script does not write that file, and shows an error. The exit status is 1 when a check fails, and 0 when all files are correct.

Limits

- `uncrunch.py` decodes the files from CRUNCH 2.x. It does not decode files from CRUNCH 1.x.
- The scripts do not decode LZH-crunched files (a `Y` in the extension, such as `.AYM`). They show the type of the file and write nothing.
- A text file from CP/M keeps its CP/M form: CR/LF at the end of each line, and `Ctrl-Z` bytes after the end of the text.