# XRoar 0.36

Dragon and Tandy Colour Computer emulator

This manual is for XRoar (version 0.36), a Dragon and Tandy Colour Computer emulator.

# Table of Contents

# Introduction

XRoar is a Dragon emulator that runs on a wide variety of platforms. Due to hardware similarities, XRoar also emulates the Tandy Colour Computer (CoCo) models 1 & 2. Some features are:

- Emulates Dragon 32, Dragon 64, Dragon 200-E, Tandy CoCo 1 & 2, and compatibles.
- Emulates DragonDOS, Delta and RSDOS disk systems.
- Emulates the Orchestra 90-CC stereo sound cartridge.
- Supports both raw and translated keyboard modes.
- Reads and writes virtual cassettes (compact `.cas` files and audio files).
- Reads and writes VDK, JVC and DMK format virtual floppy diskettes.
- Saves and loads machine snapshots.
- Provides a GDB target for remote debugging.
- Games Master Cartridge support, including SN76489 sound chip.
- MOOH RAM expansion + SPI support.
- Glenside IDE support.

XRoar is easily built from source under Linux, and binary packages are provided for Mac OS X and Windows.

XRoar can also be compiled to WebAssembly, and redistributing it in this form may provide a convenient way for users to run your Dragon software. See XRoar Online (`http://www.6809.org.uk/xroar/online/`) for an example.

# 1 Getting started

## 1.1 Installation & running

Pre-built binary packages are available from the XRoar home page (`http://www.6809.org.uk/xroar/`). If one is not available for your architecture, you will need to build from source. XRoar should build and run on any POSIX-like system for which SDL version 2 is available.

You will also need BASIC ROM images—binary dumps of the firmware from an original machine. The originals were part-written by Microsoft, so they are not distributed in the XRoar packages.

### 1.1.1 Mac OS X binary package

Download and unzip the appropriate `.zip` distribution for your system. Drag the application icon to `/Applications/`.

For troubleshooting or testing options, it's often a good idea to run from the command line, but application packages don't make that trivial. A symbolic link to somewhere in your `PATH` is all that's required. e.g.:

```
$ sudo ln -s /Applications/XRoar.app/Contents/MacOS/xroar \
        /usr/local/bin/xroar
```

After this, you can start the emulator by simply typing `xroar` followed by any command line options.

ROM images should be placed in a directory you create under your `HOME` named `~/Library/XRoar/roms/` (not the system directory, `/Library/`). Name any configuration file you create `~/Library/XRoar/xroar.conf`.

The Mac OS X build provides a menu for access to certain features, and often accepts the more familiar *`Command`*+*key* in place of the *`Ctrl`*+*key* shortcuts listed in this manual.

### 1.1.2 Windows binary package

Download and unzip the appropriate `.zip` distribution for your system.

The easiest way forward is to simply put ROM images into the directory created when you unzip the distribution, and then run the `.exe` straight from there. You can also put any configuration file (`xroar.conf`) here.

However, if you want to avoid having to move files around each time you upgrade, you can create `Documents/XRoar` to contain your configuration file, and a subdirectory of that, `Documents/XRoar/roms` for ROM images.

Note when troubleshooting that the logging from the Windows binary is probably only going to be visible if you run it with the `-C` option (must be the first option) to allocate a console.

The Windows build provides menu-based access to certain features.

## 1.2 Building from source

### 1.2.1 Dependencies

If there is no binary package for your system, you will have to build from source. XRoar can use various backend toolkits, and you will need to ensure you have their development files installed. If you're using Debian, this can (at the time of writing) be achieved with the following simple command:

```
$ sudo apt install build-essential libsndfile1-dev libgtk2.0-dev \
        libgtkglext1-dev libasound2-dev
```

Under Mac OS X, first be sure to install Apple's Xcode (`https://developer.apple.com/xcode/`) package. The easiest way to then ensure you have XRoar's dependencies available is to use a system like Homebrew (`https://brew.sh/`) or MacPorts (`http://www.macports.org/`). For Homebrew, the following command will install the required dependencies:

```
$ brew install libsndfile sdl2
```

Otherwise, you'll have to do a bit of platform-specific research to ensure you have all the dependencies for a full build:

GTK+ (`http://www.gtk.org/`), the GIMP toolkit, provides the most full-featured user interface. It is only usable as such if you also have GtkGlExt (`http://projects.gnome.org/gtkglext/`), an OpenGL extension used to provide video output. Otherwise, it can provide a file requester for use by other user interfaces.

SDL (`http://www.libsdl.org/`), Simple Directmedia Layer, provides a slightly more basic user experience. Menus are added using native code under Mac OS X and Windows; any other target using SDL will support only keyboard shortcuts. Unless you are building for Linux, SDL is required to use joysticks. Version 2 required.

POSIX Regular Expressions are used in option parsing, so TRE (`https://laurikari.net/tre/about/`) is required on non-POSIX platforms (e.g. Windows).

Other supported audio APIs: OSS, ALSA, PulseAudio, CoreAudio. Some other options are still in the code base, but have not been tested in a while.

libsndfile (`http://www.mega-nerd.com/libsndfile/`) is recommended to enable support for using audio files as cassette images.

### 1.2.2 Compilation

Once you have the dependencies, building XRoar follows a familiar procedure:

```
$ gzip -dc xroar-0.36.tar.gz | tar xvf -
$ cd xroar-0.36
$ ./configure
$ make
$ sudo make install
```

The `configure` script has a lot of options guiding what it tests for, specifying cross-compilation, changing the install path, etc. List them all with the `--help` option.

By default, `configure` will set up an install *prefix* of `/usr/local`, but this can be changed by using the `--prefix=`*`path`* option.

Once built, run `make install` as root (or use `sudo`, as in the example above) to install the binary and info documentation on your system. The executable is called `xroar`. ROM images should be placed either in your home directory as `~/.xroar/roms/`, or under the installation *prefix* as *`prefix`*`/share/xroar/roms/`. Any configuration file should be created as `~/.xroar/xroar.conf`.

XRoar can be built on one platform to run on another. The Windows binary package is built like this. To specify a cross-compile, use the `--host=`*`host`* argument to `configure`. For example, to build for Windows, you might use '`./configure --host=i686-w64-mingw32`'. Getting everything "just so" for a cross-build can be a tricky procedure, and the details are beyond the scope of this manual.

XRoar can be built to a WebAssembly target using Emscripten (`https://emscripten.org/`). With the SDK installed, run `emconfigure ./configure --enable-wasm` to set up the build environment. Build with `emmake make`. HTML/JavaScript and CSS examples for interfacing to the output are included in the `wasm/` subdirectory.

## 1.3 Running programs

Images of ROM cartridges, cassettes and disks can all be attached by pressing *Ctrl+L*. Cassettes will be positioned at the beginning of the "tape" and disks will be inserted into the first drive.

To attach an image and try to automatically start it, press *Ctrl+Shift+L* instead. XRoar will try and start the program using the normal method for the type of image:

- If a BASIC program is found on a cassette, XRoar types "CLOAD", waits for the program to load, then types "RUN".
- For machine code programs on cassette, XRoar types "CLOADM" if they appear to load in an autorun fashion, otherwise "CLOADM:EXEC". Special built-in rules try and recognise some programs and issue a customised series of commands.
- If a ROM cartridge is inserted, the emulated machine is power cycled.
- If a disk image is autorun, XRoar types "BOOT" (Dragon) or "DOS" (CoCo).

If you have trouble with a program, refer to its loading instructions: it may require a different series of commands to get going. If you still have issues, see Section 2.2 [Cartridges], page 7, Section 3.1 [Cassettes], page 9, or Section 3.2 [Floppy Disks], page 9.

The keyboard layout of modern machines is not identical to that of the Dragon, so XRoar provides two ways of working. The default, "untranslated", is to map key *positions* as closely as possible (e.g. the key to the right of *P* maps to *@*). Pressing *Ctrl+Z* toggles "translated" mode, which tries to map key *symbols* closely instead (e.g. pressing *'* will press *Shift+7* in the emulated machine).

In addition, a couple of keys aren't always available on modern keyboads, so `Escape` maps to the Dragon's `BREAK` key, and `Home` maps to `CLEAR`. See Section 4.4 [Keyboard], page 14.

XRoar will make use of attached joysticks, but can also emulate them with the cursor keys and `Left Alt`. Press *Ctrl+J* to cycle through three emulation modes: No joystick emulation (default), Right joystick, Left joystick. See Section 4.5 [Joysticks], page 15.

## 1.4 Configuration

This manual details the options available for configuring XRoar. Each option can be passed on the command line or placed in a configuration file called `xroar.conf`. `xroar.conf` is read first, and any command line options then take precedence. Directives are listed in `xroar.conf` one per line.

Lines in the configuration file are either of the form:

```
OPTION [VALUE [, VALUE]...]
```

Or, for list assignment (e.g. ROM lists):

```
OPTION KEY = [VALUE [, VALUE]...]
```

Each '`OPTION`', '`KEY`' or '`VALUE`' may be surrounded by whitespace, include quoted sections (text surrounded by single or double quote characters to protect special characters), and include escape sequences (special sequences prefixed with a backslash, see the table below). There is no special distinction between single- and double-quotes, and escape sequences may still occur within either. Currently, an '`=`' symbol may follow an '`OPTION`', but this is entirely optional, and deprecated (may be removed in a future version).

Empty lines are skipped, and any line where the first non-whitespace character is a '`#`' is treated as a comment.

An '`OPTION`' may be preceded by leading dashes as from the command line, but within the configuration file they are not necessary. Most options may be preceded by '`no-`' to invert their meaning or clear an option.

| Sequence | Description |
|---|---|
| '`\0`' | Null (NUL), ASCII 0. Note that this is only permitted when *not* followed by another octal digit, as it may be confused with an octal byte, so it may be preferable to use '`\x00`' instead. |
| '`\a`' | Bell (BEL), ASCII 7, no equivalent on the Dragon keyboard. |
| '`\b`' | Backspace (BS), ASCII 8, *`LEFT`*. |
| '`\e`' | Escape (ESC), ASCII 27, no equivalent on the Dragon keyboard, but mapped to *`BREAK`* by the `-type` option. |
| '`\f`' | Form Feed (FF), ASCII 12, *`CLEAR`*. |
| '`\n`' | Newline (NL), ASCII 10, *`DOWN`*. Not usually used by the Dragon as a line ending, instead try '`\r`'. |
| '`\r`' | Carriage Return (CR), ASCII 13, *`ENTER`*. |
| '`\t`' | Horizontal Tab (HT), ASCII 9, *`RIGHT`*. |
| '`\v`' | Vertical Tab (VT), ASCII 11, no equivalent on the Dragon keyboard. |
| '`\x`*nnn*' | 8-bit byte with value specified as a three-digit octal number, *nnn*. |
| '`\x`*hh*' | 8-bit byte with value specified as a two-digit hexadecimal number, *hh*. |
| '`\u`*hhhh*' | 16-bit Unicode codepoint specified as a four-digit hexadecimal number, *hhhh*. Internally, this will be encoded as UTF-8. |

Any other character following a backslash is included verbatim.

Note that when specifying options on the command line, it is assumed that your shell will be handling any quoting, thus quote characters that make it through to XRoar are included verbatim. For similar reasons, leading and trailing whitespace is also included verbatim. However, the escape sequences documented above may still be included.

Section 1.1 [Installation], page 2, and Section 1.2.2 [Compilation], page 3, list good default locations for `xroar.conf`, but it is actually searched for in a list of directories. You can override this search path with the `XROAR_CONF_PATH` environment variable, which contains a '`:`'-separated list of directories. Here are the defaults:

| Platform | Default `XROAR_CONF_PATH` |
|---|---|
| Unix/Linux | `~`/.xroar:*prefix*/etc:*prefix*/share/xroar |
| Mac OS X | `~`/Library/XRoar:`~`/.xroar:*prefix*/etc:*prefix*/share/xroar |
| Windows | :`~`/Documents/XRoar:`~`/AppData/Local/XRoar:`~`/AppData/Roaming/XRoar |

Note the leading '`:`' in the Windows default indicates an empty entry, meaning it will look in the current working directory first (i.e. you can put `xroar.conf` into the directory from which you run XRoar).

'`~`' indicates the user's home directory: the `HOME` environment variable on Unix systems, or `USERPROFILE` on Windows. *prefix* is the installation prefix, which is usually `/usr/local`.

To bypass the search path and start XRoar using a specific configuration file, pass `-c` *`filename`* as the very first option to XRoar.

To print the current configuration to standard output (suitable for redirection to a config file), run with `-config-print`. This will include all the built-in machine and cartridge definitions. For a complete version including default values, use `-config-print-all`.

# 2 Hardware emulation

## 2.1 Machines

With no other options, XRoar searches the ROM path and determines which supported machine has firmware ROM images are available. It tries in this order: Dragon 64, Dragon 32, Tandy CoCo. This can be overridden with the `-default-machine` *name* option. XRoar has built-in configurations for the following machines:

| **Name** | **Description** |
|---|---|
| '`dragon32`' | Dragon 32 (PAL). |
| '`dragon64`' | Dragon 64 (PAL). |
| '`dragon200e`' | Dragon 200-E (PAL). |
| '`tano`' | Tano Dragon (NTSC). |
| '`coco`' | Tandy Colour Computer (PAL). |
| '`cocous`' | Tandy Color Computer (NTSC). |
| '`mx1600`' | Dynacom MX-1600 (PAL-M). |

You can create new machine configurations or reconfigure existing ones. Select a machine configuration with `-machine` *name*, and use the following options to modify it:

`-machine-desc` *name*
: Description shown in `-machine help`.

`-machine-arch` *arch*
: Base machine architecture. One of '`dragon64`', '`dragon32`' or '`coco`'.

`-machine-cpu` *cpu*
: Fitted CPU. One of '`6809`' or '`6309`'. Hitachi 6309 support is not as well tested.

`-bas` *rom*
: ROM image to use for Colour BASIC (CoCo only).

`-extbas` *rom*
: ROM image to use for Extended BASIC.

`-altbas` *rom*
: 64K-mode Extended BASIC for Dragon 64 & Dragon 200-E.

`-nobas`
`-noextbas`
`-noaltbas`
: Indicate the corresponding ROM is not fitted in this machine.

`-ext-charset` *rom*
: ROM image to use for external character generator.

`-tv-type` *type* `One of 'pal', 'ntsc' or 'pal-m'.`
: PAL-M is treated the same as NTSC, except with magenta-green artefacting instead of blue-red (simulated composite rendering only at the moment).

`-vdg-type` *type*
: Indicate the VDG variant fitted. One of '`6847`' or '`6847t1`'.

`-ram` *kbytes*
: Amount of RAM fitted.

`-machine-cart` *name*
: Default cartridge to attach. See Section 2.2 [Cartridges], page 7.

`-nodos` Indicate that XRoar is not to automatically attempt to attach a DOS cartridge to this machine (the default is to try).

For example, if the following lines were placed in your `xroar.conf`, a new machine could be selected with `-default-machine pippin`:

```
machine pippin
machine-desc Dragon Pippin (prototype)
machine-arch dragon32
ram 16
```

## 2.2 Cartridges

The default cartridge for a machine is selected with the `-machine-cart` option. XRoar has built-in definitions for four cartridges:

'`dragondos`'
DragonDOS, official disk interface cartridge from Dragon Data Ltd. Based on the WD2797 Floppy Disk Controller chip.

'`delta`'
Delta System, Premier Microsystems' disk interface cartridge for the Dragon. Mk 2, based on the WD2791 controller.

'`rsdos`'
RSDOS, Tandy's disk interface cartridge for use with the CoCo. Based on the WD1793 controller.

'`gmc`'
John Linville's Games Master Cartridge provides the ability to bank switch up to 64K of cartridge ROM, along with an on-board SN76489 sound chip. Be aware that as provided, the cartridge is designed to auto-start an on-board ROM, and XRoar reflects this use. If you want to use it as a standalone cartridge, you'll need to use the `-no-cart-autorun` option.

'`orch90`'
The Orchestra 90-CC cartridge provides stereo 8-bit audio output. The default ROM is CoCo-only, but the hardware is compatible with the Dragon.

'`ide`'
Glenside IDE interface for hard disks. See Section 3.3 [Hard Disks], page 10.

'`nx32`'
Tormod Volden's NX32 RAM expansion cartridge.

'`mooh`'
Tormod Volden's MOOH RAM expansion cartridge.

'`mpi`'
Multi-Pak Interface. A CoCo add-on by Tandy that allows up to four cartridges to be connected, selectable by software or hardware switch.

You can create new cartridge configurations or reconfigure existing ones. Select a cartridge configuration with `-cart` *`name`*, and use the following options to modify it:

`-cart-desc` *`text`*
Cartridge description shown in `-cart help`.

`-cart-type` *type*
: Set cartridge base type. One of '`rom`', '`dragondos`', '`delta`', '`rsdos`' or '`orch90`'.

`-cart-rom` *filename*
: The ROM image specified will be mapped from $C000.

`-cart-rom2` *filename*
: The ROM image specified will be mapped from $E000.

`-cart-becker`
: Enable becker port where supported.

`-cart-autorun`
: Auto-start cartridge using FIRQ.

Defining extra cartridges is most usefully done in the configuration file, for example:

```
cart sdose6
cart-desc SuperDOS E6
cart-type dragondos
cart-rom sdose6.rom
cart-rom2 dosdream.rom
```

This will define a cartridge called '`sdose6`' as a DragonDOS cartridge with its ROM replaced with `sdose6.rom`, and an additional ROM called `dosdream.rom`.

XRoar will automatically attempt to find a disk interface relevant to the current machine unless a specific default has been configured for the machine with `-machine-cart`, or automatic selection is disabled with the `-nodos` option.

Selecting a ROM image file with the `-load` or `-run` command line options, or with *Ctrl+L* or *Ctrl+Shift+L*, will attach a ROM cartridge.

Within the emulator, cartridges can be enabled or disabled by pressing *Ctrl+E*. You will almost certainly want to follow this with a hard reset (*Ctrl+Shift+R*).

### 2.2.1 Multi-Pak Interface

If you attach a Multi-Pak Interface (MPI), you'll want to populate one or more of its slots (numbered 0-3). Use `-mpi-load-cart [`*slot*`=]`*name* to attach a named cartridge to the specified (or next) slot. Configure the initially selected slot with `-mpi-slot` *slot*.

It's not recommended to load more than one DOS cartridge into the MPI. As things stand, only the last one (in slot order) will have the emulated drives properly connected.

### 2.2.2 Becker port

XRoar supports an emulator-only feature that enables it to connect to a server using a TCP connection to access remote facilities such as disk images and MIDI devices—the "becker port". This appears as a memory-mapped device, and XRoar supports it as an optional feature of the DOS cartridge.

Enable this port when configuring a cartridge with `-cart-becker`. The `-becker` option tells XRoar to prefer a cartridge with it enabled when automatically selecting one.

The IP and port to connect to can be specified with the `-becker-ip` and `-becker-port` options. These default to '`127.0.0.1`' and '`65504`' respectively, matching the defaults for DriveWire 4, the most popular server application used to provide such facilities.

# 3 Files

In general, files can be attached on the command line with `-load` *`filename`*, or by pressing *`Ctrl+L`*. XRoar judges the type of file based on its filename extension. To attempt to intelligently autorun a file, use `-run` *`filename`* or press *`Ctrl+Shift+L`*. See Section 1.3 [Running programs], page 4, for the methods XRoar will use to autorun a file.

## 3.1 Cassettes

XRoar supports three types of cassette image:

| Extension | Description |
|---|---|
| `.cas` | Cassette file. Simple binary representation of data contained on a tape. CUE support appends metadata describing cycle widths and representing silence. |
| `.wav` | Cassette audio file. XRoar can read sampled audio from original cassettes. |
| `.bas`, `.asc` | ASCII BASIC file. XRoar will convert text on the fly into blocks suitable for loading in ASCII mode. Read-only. |

To create a cassette image for writing (with the `CSAVE` or `CSAVEM` BASIC commands, for example), use the `-tape-write` *`filename`* option, or press *`Ctrl+W`*. If a file already exists, new data will be appended to it. Rewind the output take to overwrite it. Note a slight behaviour difference: CAS files can only be overwritten or appended to, WAV files can overwrite existing data (leaving later data intact).

The currently open tape files used for reading and writing are distinct.

Some cassette tapes are distributed with only one of the stereo channels containing data, and these sometimes cause loading issues. How stereo audio files are read can be adjusted with the `-tape-channel-mode` *`mode`* option, where *mode* is one of: '`mix`' - mix two channels into one (the default), '`left`' - read only the left channel, '`right`' - read only the right channel.

Select the output frame rate when writing to audio files with `-tape-ao-rate` *`hz`*. The default is 9600Hz.

Some debugging actions can be triggered each time the cassette motor switches off (e.g. when finished loading). See Section 4.7 [Debugging], page 16.

The `-tape-fast` option accelerates tape loading by intercepting ROM calls. Disable with `-no-tape-fast`. On by default.

When fast loading is enabled, the `-tape-pad-auto` option will, for `.cas` files, ignore the motor delay when insufficient initial leader bytes are detected. This helps with old tape images that would not load on real hardware if simply converted to an audio file. On by default; switch off with `-no-tape-pad-auto`.

The `-tape-rewrite` option enables rewriting of anything read from the input tape to the output tape. This is useful for creating "well formed" `.cas` files.

Where available, these options can be changed on the fly in the GUI.

## 3.2 Floppy Disks

If a disk interface cartridge is selected, XRoar supports virtual disks. Three virtual disk formats are supported:

| Extension | Description |
|---|---|
| `.dmk` | Disk image file in a format defined by David Keil. These images store a lot of information about the structure of a disk and support both single and double density data. |

`.jvc`, `.os9`, `.dsk` Disk image file in a basic sector-by-sector format with optional header information.

`.vdk` Another disk image file format, used by PC-Dragon.

To insert a disk into a particular drive, press *`Ctrl`+[1-4]*.

When you attach a disk, it is read into memory, and subsequent disk operations are performed on this in-memory copy. Write enable defaults to on (so write operations on the copy will work), but write back defaults to off, so updates will not be written to the disk image file. To toggle write enable, press *`Ctrl`+[5-8]*, where the number to press is the drive number plus 4. To toggle write back, press *`Ctrl`+`Shift`+[5-8]*. Even with write back enabled, image files will not be updated until the disk in a virtual drive is changed, or you quit the emulator.

Where available, these options can also be changed on the fly in the GUI.

Write back can be set to default to on with the `-disk-write-back` command line option.

The JVC format specifies that the disk images without headers are single-sided, but some double-sided disk images have been made available without headers. These cannot normally be distinguished from a single-sided disk that happens to have twice the number of tracks. If an OS-9 filesystem is present, the identification sector is inspected to determine the correct disk structure. This step will always be performed for headerless images with the `.os9` filename extension, but may be disabled for the other valid JVC filename extensions with `-no-disk-auto-os9`.

You can create a new blank disk in a virtual drive by pressing *`Ctrl`+`Shift`+[1-4]*. You will be a prompted for a filename, and the filename extension determines which type of file will be written.

Note that RS-DOS for the Tandy Colour Computer numbers its drives from zero instead of one, so when you perform operations on Drive 1, from the CoCo's point of view, that will be Drive 0.

## 3.3 Hard Disks

Some support is now in place for IDE hard disk images. The '`ide`' cartridge type emulates a Glenside IDE controller mapped to addresses $FF50–$FF58.

A fixed image file named `hd0.img` in the current working directory is assumed. If the image does not exist when you try to use the IDE cartridge, a new 256MB image is created. Images consists of a 512 byte identifier block (ASCII string "1DED15C0" then all zeroes) followed by a 512 byte IDENTIFY header, then the sector data. In examples, the 256 byte sector data of standard CoCo disks is doubled up to fill the 512 byte sectors of the hard disk image.

## 3.4 ROM cartridges

ROM cartridge images have a `.rom` or `.ccc` filename extension. Because XRoar supports other types of cartridge, loading a ROM image actually just creates a cartridge instance of type '`rom`'. See Section 2.2 [Cartridges], page 7.

## 3.5 Snapshots

XRoar can save out a snapshot of the emulated machine state and read the snapshots back in later. To save a snapshot, press *`Ctrl`+`S`*. When using *`Ctrl`+`L`* to load a file, anything ending in `.sna` will be recognised as a snapshot.

What is included in snapshots: Selected machine architecture, complete hardware state, current keyboard map, filenames of attached disk image files.

What is *not* (yet) included: Actual disk image data (only where to find it), attached cassettes or cartridge ROM contents.

## 3.6 Binary files

File types containing raw binary data to be loaded into RAM:

| Extension | Description |
|---|---|
| .bin | Binary file (DragonDOS or CoCo). XRoar can load these directly into memory and optionally autorun them. Read-only |
| .hex | Intel hex record. An ASCII format that encodes binary data and where in memory to load it. Read-only |

## 3.7 Firmware ROM images

Firmware ROM image files are configured as part of a machine or a cartridge. They have a filename extension of `.rom` or `.dgn`, and can be specified as:

- Complete path to a file.
- Base filename of an image, to be discovered within a search path.
- Base filename of an image, omitting the extension. XRoar will search as above, appending the known ROM filename extensions.
- An '`@`' character followed by the name of a ROM list.

A ROM list is a comma-separated list of images, each following the rules above. ROM lists may refer to other ROM lists. Define a ROM list with `-romlist` *`name=image[,image]...`*. View the defined ROM lists with `-romlist-print`.

To make life easier, the default image for each type of machine or cartridge usually refers to a ROM list which contains all the corresponding filenames seen "in the wild":

| **Firmware ROM** | **ROM list** | **Canonical image names** |
|---|---|---|
| Dragon 32 BASIC | '`@dragon32`' | `d32` |
| Dragon 64 32K BASIC | '`@dragon64`' | `d64_1` |
| Dragon 64 64K BASIC | '`@dragon64_alt`' | `d64_2` |
| Dragon 200-E 32K BASIC | '`@dragon200e`' | `d200e_1` |
| Dragon 200-E 64K BASIC | '`@dragon200e_alt`' | `d200e_2` |
| Dragon 200-E Charset | '`@dragon200e_charset`' | `d200e_26` |
| Tandy Colour BASIC | '`@coco`' | `bas13`, `bas12`, `bas11`, `bas10` |
| Tandy Extended BASIC | '`@coco_ext`' | `extbas11`, `extbas10` |
| DragonDOS | '`@dragondos_compat`' | `dplus49b`, `sdose6`, `ddos40`, `cdos20` |
| Delta System | '`@delta`' | `delta` |
| RS-DOS | '`@rsdos`' | `disk11`, `disk10` |
| RS-DOS with becker port | '`@rsdos_becker`' | `hdbdw3bck` |
| Orchestra 90-CC | '`orch90`' | |

The default search path for images specified only as a base filename varies by platform, and is detailed in Chapter 1 [Getting started], page 2. This path can can be overridden with the option `-rompath=`*`path`*, where *path* is a colon-separated list of directories to search. The path is parsed in the same manner as the configuration file search path (see Chapter 1 [Getting started], page 2).

The `XROAR_ROM_PATH` environment variable can also be used to specify the search path, but this behaviour is deprecated and may be removed in a future version.

A CRC32 value is calculated and reported for each ROM image loaded. XRoar uses these CRCs to determine whether certain breakpoints can be used (e.g. for fast tape loading). The lists of CRCs matched can be defined in a similar way to ROM lists using the `-crclist` *`list=crc[,crc]...`* option. Each *crc* is a 8-digit hex number preceeded by '`0x`', or the name of a nested list preceeded by '`@`'. Use this if you have a modified version of a BASIC ROM that maintains compatible entry points with an original. View the current lists with `-crclist-print`.

Sometimes it may be useful to force CRC matching so that breakpoints apply (e.g. you are modifying a ROM image and don't wish to have to add its CRC to the match list each time you modify it). The `-force-crc-match` option forces the CRCs to be as if an original ROM image were loaded.

# 4 User interface

## 4.1 User interface module

The user interface depends on supporting toolkit packages as described in Section 1.2 [Building from source], page 2. Selection of user interface module may affect which other types of module are available: in particular, video output is strongly tied to the user interface.

`-ui` *`module`*
: Select user-interface module. `-ui help` to list compiled-in modules.

### 4.1.1 GTK+ user interface

Select with `-ui gtk2`.

This is the most full-featured user interface. It provides extensive dynamic menus, and control dialogues for cassette and disk files. This is the preferred interface under Linux.

Only one video module is usable with this user interface: '`gtkgl`'.

### 4.1.2 SDL user interface

Select with `-ui sdl`.

More limited than the GTK+ interface, providing keyboard shortcuts and, in the Mac OS X and Windows variants, some basic menus.

Under Mac OS X, many operations are usable by pressing *`Command+`*`key` as well as the usual shortcut of *`Ctrl+`*`key`.

## 4.2 Video output

`-vo` *`module`*
: Video output module to use. Available modules depend on the selected user-interface module. `-vo help` gives a complete list for each possible user-interface.

`-fs`
: Start full-screen. Toggle full-screen with *`Ctrl+F`* or *`F11`*.

`-fskip` *`frames`*
: Specify frameskip. Default is '`0`'. For slower machines.

`-gl-filter` *`filter`*
: Filtering method to use when scaling the screen. One of '`linear`', '`nearest`' or '`auto`' (the default). OpenGL output modules only.

`-invert-text`
: Start up with inverted text mode.

`-ccr` *`renderer`*
: Composite video cross-colour renderer. One of '`simple`' (very fast), '`5bit`' (fast, more accurate) or '`simulated`' (slow, very accurate). Default is '`5bit`'.

Real NTSC machines start in one of two cross-colour states at random. Games often prompt the user to "Press Enter if the screen is red", for example. You can press *`Ctrl+A`*, to cycle through three modes: Off, Blue-red and Red-blue.

Inverted text mode may be toggled by pressing *`Ctrl+Shift+I`*.

## 4.3 Audio output

`-ao` *`module`*
: Select audio output module. `-ao help` for a list.

`-ao-device` *`device`*
: Module-specific device specifier. e.g. `/dev/dsp` for OSS.

`-ao-format` *`format`*
: Specify audio sample format. `-ao-format help` for a list.

`-ao-rate` *`hz`*
: Specify audio frame rate, where supported. The default is taken from the operating system if possible, otherwise it will usually be '`48000`'.

`-ao-channels` *`n`*
: Specify number of channels (1 or 2). Default is usually 2.

`-ao-fragments` *`n`*
: Specify number of audio fragments.

`-ao-fragment-ms` *`ms`*
: Specify audio fragment size in milliseconds.

`-ao-fragment-frames` *`frames`*
: Specify audio buffer size in frames.

`-ao-buffer-ms` *`ms`*
: Specify total audio buffer size in milliseconds.

`-ao-buffer-frames` *`frames`*
: Specify total audio buffer size in frames.

`-ao-gain` *`db`*
: Specify audio gain in dB relative to 0 dBFS. Only negative values really make sense here. The default is '`-3.0`'.

`-volume` *`volume`*
: Older way to specify volume. Simple linear scaling, using values 0–100.

`-fast-sound`
: Slightly faster audio support by ignoring certain uncommon state changes.

Audio latency is a concern for emulators, so XRoar allows the buffering characteristics to be configured with the fragment and buffer options above. Not all audio modules support all options, but setting the total audio buffer size will usually have an effect. Bear in mind that any figures reported by XRoar reflect what it was able to request, and won't include any extra buffering introduced by the underlying sound system.

When the Orchestra 90-CC cartridge is attached, its stereo output needs to be mixed with the Dragon's normal audio. To allow a small amount of headroom for this, the default gain is set to '`-3.0`' (dB relative to full scale), but be aware that it would still be possible for this to clip depending on what's happening on the internal sound bus. A setting of `-ao-gain -9.0` would give plenty of headroom (at the expense of a quieter overall sound).

## 4.4 Keyboard

The default mapping of host keys to emulated keys is based on the original *positions* of the keys, with certain exceptions: cursor keys are mapped directly, `Escape` maps to the Dragon's `BREAK` key, and `Home` maps to `CLEAR`. Other keys may also be mapped to `CLEAR` if there is a choice in your selected keymap that doesn't conflict with a regular character in translated mode.

For position-based mapping, XRoar needs to be informed of the layout of the host's keyboard. If it is not the default (UK), use the `-keymap` *`code`* option.

XRoar can also be put into "translated" keyboard mode, where characters typed on a PC keyboard are translated into the equivalent keystrokes on the Dragon. Use the `-kbd-translate` option to default to this mode. Press *`Ctrl+Z`* at any time to toggle between the two modes.

In translated mode, *`Shift+Return`* is mapped to the Caps Lock combination (*`Shift+0`* usually, *`Shift+ENTER`* on the Dragon 200-E). Similarly, *`Shift+Space`* is mapped to the "pause output" combination (*`Shift+@`* usually, *`Shift+Space`* on the Dragon 200-E).

`-keymap` *`code`*
: Specify host keyboard layout. `-keymap help` for a list. Default is '`uk`'.

`-kbd-translate`
: Start up in "translated" keyboard mode.

`-type` *`string`*
: Intercept ROM calls to type *string* into BASIC on startup.

The keyboards of the Dragon, Dragon 200-E and Tandy CoCo operate in the same way, but the matrix and/or key layouts differ. When you select a machine (see Section 2.1 [Machines], page 6), the appropriate layout is selected for you, but you can cycle between the three by pressing *`Ctrl+K`*.

XRoar will simulate the "ghosting" effects inherent in a simple matrix design, but the accuracy of this simulation will depend very much on your host keyboard, which vary greatly in the amount of simultaneous keypresses they support (for more information, search for "NKRO").

## 4.5 Joysticks

XRoar supports attached joysticks, or can emulate them using the keyboard or mouse ("virtual joysticks"). There are a few built-in configurations, or new ones can be defined. Here are the built-ins:

| **Name** | **Description** |
|---|---|
| '`joy0`' | First two axes and first two buttons of first physical joystick |
| '`joy1`' | First two axes and first two buttons of second physical joystick |
| '`kjoy0`' | Keyboard based virtual joystick using cursor keys and `Left Alt`. |
| '`mjoy0`' | Mouse based virtual joystick mapped to screen position |

By default, '`joy0`' (the first physical joystick) is mapped to the Dragon's right joystick port, and '`joy1`' (the second physical joystick) to the left port. Map different named joysticks with `-joy-right` *`name`* and `-joy-left` *`name`*. Right and left joystick mapping can be easily swapped by pressing *`Ctrl+Shift+J`*.

A configured "virtual joystick" can be used by pressing *`Ctrl+J`*. The first press substitutes it for the right joystick, the second press with the left joystick and a third press disables it again. The virtual joystick defaults to the keyboard-based '`kjoy0`' described above, but can be reconfigured with `-joy-virtual` *`name`*.

A joystick configuration can be created or configured by selecting it by name with `-joy` *`name`*, and then configuring its axes with `-joy-axis` *`index=spec`* and buttons with `-joy-button` *`index=spec`*. In each case, *spec* has the syntax *module*:*args*, with *args* being a comma-separated list, the format of which is specific to *module*:

| **Module** | **Axis args** | **Button args** |
|---|---|---|
| '`physical`' | *joystick-index*,*[-]axis-index* | *joystick-index*,*button-index* |
| '`keyboard`' | *key-name0*,*key-name1* | *key-name* |

'`mouse`' *screen-offset0,screen-offset1* *button-number*

For physical joysticks a '`-`' before the axis index inverts the axis. Key names for the keyboard module depend on the underlying toolkit. The default screen offsets for the mouse module are '`X=2,254`' and '`Y=1.5,190.5`' which gives reasonable behaviour for some games and utilities.

## 4.6 Printing

XRoar supports redirecting the Dragon parallel printer output to a file or pipe with the `-lp-file` or `-lp-pipe` option. Printed data will be sent to the appropriate stream. Pressing *Ctrl+Shift+P* will flush the current stream by closing it (so if the stream is a pipe, the filter will complete). The stream will be re-opened when any new data is sent.

The pipe feature allows you to use useful print filters such as `enscript`, e.g. `-lp-pipe "enscript -B -N r -d printer-name"`. This will send a job to your printer, using carriage returns as line feeds (the Dragon default), each time you press *Ctrl+Shift+P* (or exit the emulator).

`-lp-file` *filename*
: Append printer output to *filename*.

`-lp-pipe` *command*
: Pipe printer output to *command*.

Note that the CoCo uses a serial printer port. As full serial support is yet to be added, a very limited form of print redirection is implemented for the CoCo using a ROM BASIC intercept. This is enough to support BASIC commands like LLIST, but will not cope with programs implementing their own serial routines.

## 4.7 Debugging

XRoar can act as a remote target for GDB using a network socket. When GDB connects, emulation is stopped. GDB can then inspect memory, instruct the target to set breakpoints and watchpoints (read, write and access), single step or continue execution. A version of GDB patched to specifically support 6809 targets can also perform disassembly and inspect registers. For more information on how to use GDB, see the GDB Documentation (`http://www.gnu.org/software/gdb/documentation/`).

Enable the GDB remote target with `-gdb`. The default IP and port for the target are '`127.0.0.1`' and '`65520`'. These can be overridden with the `-gdb-ip` and `-gdb-port` options.

XRoar also supports a simpler "trace mode", where it will dump a disassembly of every instruction it executes to the console. Toggle trace mode on or off with *Ctrl+V*. Trace mode can be enabled from startup with the `-trace` option.

User-interface debugging flag can be enabled with `-debug-ui` *value*, where only one value is currently supported:

0x0001 Keyboard event debugging.

Hex & binary file debugging can be enabled with `-debug-file` *value*, where the value is a bitwise ORing of the following:

0x0001 Print summary information such as load or exec addresses.

0x0002 Hex dump of all data read into memory.

0x0004 Print filename block metadata when autorunning a tape.

Floppy controller debugging can be enabled with `-debug-fdc` *value*, where the value is a bitwise ORing of the following:

0x0001 Show FDC commands.

0x0002 Show all FDC states.

0x0004 Hex dump of read/write sector data.

0x0008 Hex dump of becker port conversation data.

The GDB stub can also emit debug information about its own operation with `-debug-gdb` *`value`*, where value is a bitwise ORing of:

0x0001 Connection open and close.

0x0002 Show packet data.

0x0004 Checksum reporting.

0x0008 Report on general queries.

The special value argument of -1 parses as "all bits set", and so enables all corresponding debug options.

XRoar prints various other informational messages to standard output by default, including when the state of certain toggles is modified. Verbosity can be changed with the `-verbose` *`level`* option. `-quiet` is equivalent to `-verbose 0`. Levels are:

0 Quiet. Only warnings and errors printed.

1 Print startup diagnostics and emulator state changes (default).

2 Report some emulated machine state changes.

3 Miscellaneous internal debugging.

XRoar can be told to exit after a number of (emulated) seconds with the `-timeout` *`seconds`* option.

XRoar can quit a number of seconds after the cassette motor is switched off with the `-timeout-motoroff` *`seconds`* option. This is useful in the case of automatic tape rewriting. A value of 1 is usually sufficient to account for the brief motor click that occurs after header blocks and during gapped loading.

Similarly, a snapshot can be automatically written after loading with the `-snap-motoroff` *`file`* option. The file is overwritten each time the motor transitions to off. This can be used to help analyse the machine state immediately after loading, before any autorun code has taken effect.

To see debug output from the pre-built Windows binary, run it with `-C` as the first option to allocate a console.

## 4.8 Keyboard shortcuts

A summary of commonly available shortcuts.

*`Ctrl+A`* Cycle through cross-colour video modes (hi-res only).

*`Ctrl+D`* Open disk control dialogue (GTK+ only).

*`Ctrl+E`* Toggle DOS emulation on/off - reset to take effect.

*`Ctrl+F`*, *`F11`*
: Toggle full screen mode.

*`Ctrl+Shift+H`*, *`Pause`*
: Toggles pause mode (HALTs the CPU). As seen on the Dynacom MX-1600.

*`Ctrl+Shift+I`*
: Toggle text mode inverse video.

*Ctrl+J* Cycle through joystick emulation modes (None, Right, Left).

*Ctrl+Shift+J*
Swap left and right joysticks.

*Ctrl+K* Cycle through Dragon, Dragon 200-E and CoCo keyboard layouts.

*Ctrl+L* Load a file (see below).

*Ctrl+Shift+L*
Load a file and attempt to autorun it where appropriate.

*Ctrl+M* Cycle through emulated machine types (resets machine).

*Ctrl+Shift+P*
Flush printer output.

*Ctrl+Q* Quit emulator.

*Ctrl+R* Soft reset emulated machine.

*Ctrl+Shift+R*
Hard reset emulated machine.

*Ctrl+S* Save a snapshot.

*Ctrl+T* Open the tape control dialogue (GTK+ only).

*Ctrl+V* Toggle trace mode.

*Ctrl+W* Attach a virtual cassette file for writing.

*Ctrl+Z* Enable keyboard translation mode.

*F12* While held, the emulator will run at the maximum possible speed.

*Shift+F12*
Toggle rate limiting. Emulator will run at maximum speed until pressed again.

# 5 Troubleshooting

Some commonly encountered issues:

- I only see a checkerboard pattern of orange and inverse '`@`' signs.

This probably indicates that XRoar could not locate any BASIC ROM images. Acquire some and put them in the directory appropriate to your platform. XRoar tries to find ROMs for machines in the following order if you do not specify a default machine: Dragon 64, Dragon 32, CoCo.

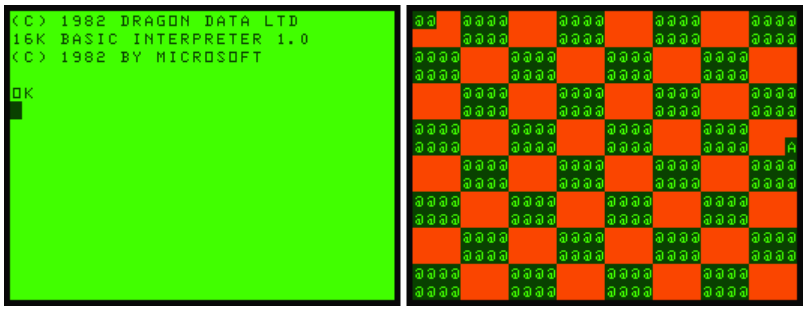


Figure 5.1: Emulator with and without BASIC ROM

- This program is supposed to be in colour, but all I see is black & white.

Try pressing *`Ctrl+A`* one or more times. What this really means is that you're used to running the program on an NTSC machine, and it makes use of cross-colour, but XRoar is emulating a PAL machine. Try starting with `-default-machine tano` or `-default-machine cocous`.


Figure 5.2: *Time Bandit, Dunlevy & Lafnear, 1983* in different artefact modes

# 6 Acknowledgements

I made reference to the MAME 6809 core for clues on how the overflow bit in the condition code register was handled.

Thanks to all the people on the Dragon Archive Forums (`http://archive.worldofdragon.org/phpBB3/`) for helpful feedback and insight.

Darren Atkinson's *Motorola 6809 and Hitachi 6309 Programmers Reference* has been very useful for 6309 support and fleshing out some of the illegal instructions on the 6809.

Alan Cox contributed the IDE code.