XRoar 1.13

Dragon and Tandy 8-bit computer emulator

This manual is for XRoar (version 1.13), a Dragon and Tandy 8-bit computer emulator.

Copyright © 2026 Ciaran Anscomb.

Table of Contents

About this manual 1

1 Introduction 2

1.1 Recent changes 2

2 Getting started 4

2.1 Prerequisites 4
2.2 Getting started under Linux/Unix 4
2.3 Getting started under Windows 5
2.4 Getting started under Mac OS X+ 5
2.5 Building from source 6
2.6 The command line 6
2.7 Troubleshooting 7
2.7.1 No BASIC ROM 7
2.7.2 Program lacks colour 7
2.7.3 Can't access HD/SD image 8
2.7.4 Debug messages 8

3 User interface 9

3.1 Selecting a machine 9
3.2 Selecting a cartridge 10
3.3 Running programs 10
3.4 Cassette tape control 11
3.5 Drive control 12
3.6 Video options 13
3.6.1 TV input 13
3.6.2 Composite rendering 13
3.6.3 TV controls 14
3.7 Keyboard layout 15
3.8 Printer control 16

4 Emulated hardware 17

4.1 Machine architectures 17
4.1.1 Dragon 32 17
4.1.2 Dragon 64 17
4.1.3 Dragon Professional 17
4.1.4 Tandy Colour Computer 1/2 17
4.1.5 Tandy MC-10 18
4.1.6 Matra & Hachette Alice 18
4.1.7 Tandy Deluxe Colour Computer 18
4.1.8 Tandy Colour Computer 3 18
4.2 Cartridge types 18
4.2.1 DragonDOS 18
4.2.2 Delta 19
4.2.3 RS-DOS 19
4.2.4 Glenside IDE controller 19

4.2.5 NX32 and MOOH cartridges ... 19
4.2.6 Games Master Cartridge ... 19
4.2.7 Orchestra 90-CC sound cartridge ... 20
4.2.8 Multi-Pak Interface ... 20
4.2.9 MC-10 MCX128 memory expansion ... 20
4.3 Internal add-ons ... 20
4.3.1 SAMx8 512K memory expansion ... 20
4.3.2 iMMUnity memory expansion ... 20
4.4 Keyboard ... 21
4.5 Joysticks ... 21
4.6 Printers ... 22

5 Storage media ... 23
5.1 Cassettes ... 23
5.1.1 Tape image file formats ... 23
5.1.2 Input and Output tapes ... 23
5.1.3 Remote motor control ... 23
5.2 Floppy disks ... 23
5.3 Hard disks ... 24

6 Configuring XRoar ... 25
6.1 Startup options ... 26
6.2 Machines ... 26
6.3 Cartridges ... 27
6.4 Becker port ... 28
6.5 Cassettes ... 28
6.6 Floppy disks ... 29
6.7 Hard disks ... 29
6.8 Keyboard ... 29
6.9 Joysticks ... 30
6.9.1 Gamepad mapping ... 31
6.10 Printers ... 31
6.11 Files ... 32
6.12 Firmware ROM images ... 32
6.13 User interface ... 32
6.14 Audio ... 33
6.15 Debugging ... 34
6.16 Other options ... 36

7 Files ... 37
7.1 Snapshots ... 37
7.2 Screenshots ... 37
7.3 Binary files ... 37
7.4 Firmware ROM images ... 37
7.4.1 BSDOS ... 38

Appendix A Acknowledgements ... 39

Appendix B Keyboard shortcuts ... 40

Appendix C File formats ... 41

About this manual

This manual is for XRoar (version 1.13), a Dragon and Tandy 8-bit computer emulator.

XRoar is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

XRoar is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <https://www.gnu.org/licenses/>.

1 Introduction

XRoar emulates the Dragon 32/64; Tandy Colour Computers 1, 2 and 3; the Tandy MC-10; and some other similar machines or clones. It runs on a wide variety of platforms. Emulated hardware includes:

- Dragon 32, 64, and 200-E; Tandy CoCo 1, 2, & 3; Tandy MC-10; Matra & Hachette Alice 4K.
- Dragon Professional and Tandy Deluxe Colour Computer prototypes, both including the AY-3-891x sound chip.
- DragonDOS, Delta and RS-DOS floppy disk controller cartridges.
- Orchestra 90-CC stereo sound cartridge.
- Games Master Cartridge, including the SN76489 sound chip.
- Glenside IDE cartridge.
- RAM expansions: NX32, MOOH, SAMx8 and iMMUnity.

Other features include:

- Raw and translated keyboard modes.
- Read and write cassette tape, floppy disk and block device images.
- Becker port for communication with remote servers.
- Save and load machine snapshots.
- GDB target for remote debugging.

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

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 (`https://www.6809.org.uk/xroar/online/`) for an example.

1.1 Recent changes

Changes in version 1.13 include:

- RAM dumps now include memory in expansions (e.g. SAMx8 or iMMUnity). This does not yet include cartridge-based expansions.
- The upper bits of MMU registers in a CoCo 3 with more than 512K of RAM are no longer readable by default, but can be configured so with `-machine-opt readable-dat`.
- New IDE cartridge profile '`ide-dragon`' configured to use a built-in bootstrap ROM. Ideal for BOOTing into Fuzix. See Section 7.4.1 [BSDOS], page 38.

Previous changes in 1.x include:

Important: Floppy disk write-back is now *enabled* by default. Writes to images held in memory will overwrite the on-disk file when ejected (or quitting XRoar). You can get the old file-preserving default behaviour back with `-no-disk-write-back`.

The SAMx8 (`https://www.6809.org.uk/dragon/samx8/`) is supported by specifying a Dragon 64 with `-machine-opt sam=samx8`, or by using the built-in '`dragonx8`' machine profile.

Traps can trigger actions when certain user-specified conditions are met, including printing one or more lines of trace from the instruction that matched.

When used with GDB, XRoar will report CPU architecture and a list of registers as a *Target Description*. The `-gdb-pseudo-regs` option will include extra state for some of the support ICs too. See Section 6.15 [Debugging options], page 34.

Gamepad mapping files can provide more consistent button layouts. See Section 6.9.1 [Gamepad mapping], page 31.

MCX128 cartridge support for MC-10.

The `-ccr simulated` renderer is more accurate, and there are many new video options. These can be changed on the fly in a new TV Controls dialog.

Screenshots in PNG format can be saved if XRoar is built with libpng.

More machines are emulated than in 0.x. A new snapshot format that preserves more state was required to support these. Old snapshots should still load for now, though this will likely be removed in time.

HD/SD images are now specified with `-load-hd0` and `-load-hd1`. IDE images with a header should be distinguished from headerless files by giving them a `.ide` extension.

MPI slot configuration is now per-cart rather than global.

2 Getting started

2.1 Prerequisites

To run XRoar, you will need to make sure you have the firmware ROM images available for the system you wish to emulate. These images can be transferred from your original machine (with some effort, outside the scope of this document) or more likely found online on one of the archive websites. Where XRoar looks to find these images depends on your host OS; the rest of this chapter will go into detail.

Firmware ROM image files should have a `.rom` extension, and be headerless (so their file size will be an exact power of two bytes). For most use cases, you'll need the BASIC ROM image(s) and a disk controller ROM image. Here are the expected filenames and sizes (in bytes) for some of the most commonly-required images:

Firmware ROM	**Filename**	**File size**
Dragon 32 BASIC	`d32.rom`	16384
Dragon 64 32K BASIC	`d64_1.rom`	16384
Dragon 64 64K BASIC	`d64_2.rom`	16384
DragonDOS	`ddos10.rom`	8192
Tandy Colour BASIC	`bas13.rom`	8192
Tandy Extended BASIC	`extbas11.rom`	8192
Tandy Super ECB (CoCo 3)	`coco3.rom`	32768
Tandy Super ECB (PAL CoCo 3)	`coco3p.rom`	32768
Tandy RS-DOS	`disk11.rom`	8192
Tandy Microcolour BASIC (MC-10)	`mc10.rom`	8192

Other machines (e.g. the less common Dragon 200-E) will need a different set of ROM images, and supported peripherals may also need their own firmware.

2.2 Getting started under Linux/Unix

If you configure a suitable Apt repository under Debian or Ubuntu, you should simply be able to `apt install xroar` (as root, or using `sudo`). See the XRoar homepage (`https://www.6809.org.uk/xroar/`) for links to an Apt repository for Debian, or to Launchpad for Ubuntu.

Otherwise, if you are comfortable building from source, see Section 2.5 [Building from source], page 6.

In your home directory, create directories `~/.xroar/` and `~/.xroar/roms/`:

```
$ mkdir -p ~/.xroar/roms
```

Copy your firmware ROM images (Section 2.1 [Prerequisites], page 4) into `~/.xroar/roms/`. For example, covering the most common machines, you might end up with a directory looking like this:

```
$ ls -l ~/.xroar/roms/
[...]
-rw-r--r-- 1 user group  8192 Jan  1  1982 bas13.rom
-rw-r--r-- 1 user group 32768 Jul 30  1986 coco3.rom
-rw-r--r-- 1 user group 32768 Jul 30  1986 coco3p.rom
-rw-r--r-- 1 user group 16384 Aug  1  1982 d32.rom
-rw-r--r-- 1 user group 16384 Aug  1  1983 d64_1.rom
-rw-r--r-- 1 user group 16384 Aug  1  1983 d64_2.rom
-rw-r--r-- 1 user group  8192 Jun  1  1983 ddos10.rom
-rw-r--r-- 1 user group  8192 Jan  1  1982 disk11.rom
```

```
-rw-r--r-- 1 user group  8192 Jan  1  1982 extbas11.rom
-rw-r--r-- 1 user group  8192 Oct  1  1983 mc10.rom
```

Start the emulator by typing `xroar` at the command line, or by selecting it from Applications → Games if your environment provides an applications menu.

Running `xroar --help` will display the supported command line options. Each of the command line options can also appear in a configuration file, which should be called `~/.xroar/xroar.conf`. You can configure many defaults and even extra machines and cartridges in this file. See Chapter 6 [Configuring XRoar], page 25, for more details.

2.3 Getting started under Windows

The simplest way to get going under Windows is to unpack the `.zip` file and copy all your ROM images into the created subdirectory, alongside the executable. You can also create a configuration file here called `xroar.conf`. Double click `xroar.exe` to run, and XRoar will look in the same directory that you start it from, and everything should work.

However, if you want a more organised installation where you don't have to re-copy files around every time you upgrade, read on.

In your user profile, there should exist a LocalAppData directory. This is something Windows calls a "Known Folder". You should be able to browse to it by entering `%LOCALAPPDATA%` as a path in an explorer window.[1]

Under `%LOCALAPPDATA%`, create a subdirectory called `XRoar`. Then within *that*, create a further subdirectory named `roms`. You can then copy your ROM images into `%LOCALAPPDATA%\XRoar\roms\`.

Start the emulator by double clicking `xroar.exe` or, if you installed the `.msi`, by selecting XRoar from the start menu.

You can also run XRoar from the command line, and it supports the same options as under Linux/Unix. Run `xroar.exe --help` to display a list of the supported command line options. Each of the command line options can also appear in a configuration file, which should be called `%LOCALAPPDATA%\XRoar\xroar.conf`. You can configure many defaults and even extra machines and cartridges in this file. See Chapter 6 [Configuring XRoar], page 25, for more details.

2.4 Getting started under Mac OS X+

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

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

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. It does not provide control dialog boxes; often, options in these dialogs will instead be found in the menu hierarchy.

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 \
```

[1] The reason for using the *local* version version of the AppData directory under Windows is that recent versions of Windows may offload files in other places to the cloud—I'm told this can happen without it ever informing the user—and we want to keep files local to the machine, as cloud access may require specific application support.

```
/usr/local/bin/xroar
```

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

For example, run `xroar --help` to display a list of the supported command line options. Each of the command line options can also appear in a configuration file, which should be called `~/Library/XRoar/xroar.conf` (under your `HOME` directory). You can configure many defaults and even extra machines and cartridges in this file. See Chapter 6 [Configuring XRoar], page 25, for more details.

2.5 Building from source

It is straightforward to build XRoar from source on any Unix-like OS so long as you have the normal build tools installed, and satisfy a few dependencies.

The binary packages for Windows are cross-compiled under Linux using MinGW; it may be possible to build natively using something like MSYS2 or Cygwin, but this is untested.

XRoar depends on external libraries for most aspects of its user interface:

- GTK+ 3 (`https://www.gtk.org/`) is recommended, and provides video, menus, and dialogs. It may be possible to use GTK+ under Mac OS X+, but this is untested. GTK+ 2 is deprecated, but still usable if you also have GtkGLExt installed.
- SDL 2 (`https://libsdl.org/`) provides a simpler interface, but extra code for Mac OS X+ adds some basic menus and file requester dialogs. For non-Linux systems, it may also be the easiest way to get support for joysticks and audio.
- PulseAudio or ALSA can also be used for audio support. Older code still exists for OSS and Jack, but these have not been tested for a while.
- libpng (`http://www.libpng.org/pub/png/libpng.html`) is recommended, and allows the saving of screenshots in PNG format.

Under Debian, these dependencies can be satisfied with this simple invocation of Apt:

```
$ sudo apt install build-essential libgtk-3-dev \
                   libpulse-dev libpng-dev
```

XRoar uses the GNU Build System (Autotools), so the compilation process should be very familiar. The following process compiles XRoar and installs it into `/usr/local`, like most other software built this way:

```
$ gzip -dc xroar-1.13.tar.gz | tar xvf -
$ cd xroar-1.13
$ ./configure
$ make
$ sudo make install
```

If you have cloned the git repository, you will also need GNU Build System packages installed (‘`autoconf`’, etc.) Running `./autogen.sh` should then generate the configure script, which you can then run as normal.

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.

2.6 The command line

Any option that can go in the configuration file can also be specified on the command line. Just be sure to prefix the option's name with a dash (‘`-`’). See Chapter 6 [Configuring XRoar], page 25.

On the command line, it is assumed that your shell will handle argument quoting, so any quote characters will be included verbatim. Escape sequences are still parsed, except when an option expects a filename, as shells often use their own escaping mechanisms when autocompleting filename arguments.

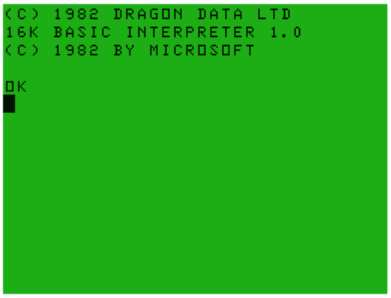
Windows has a bit of an odd relationship with standard I/O. By default, XRoar will try to attach to the so-called "parent console" (e.g. the command shell window), but you can force the allocation of a console (which may create a new window) by specifying `-C` (capital 'c') as the very first option. The `-no-C` alternative inhibits all attempts to use a console.

As a special case, the last machine selected on the command line is used as the default machine, and the last cartridge specified is attached to that machine. This means you can just use `-m` *name* instead of `-default-machine` *name*, and `-cart` *name* instead of `-machine-cart` *name*. Similarly `-no-cart` is equivalent to specifying `-no-machine-cart` for the default machine.

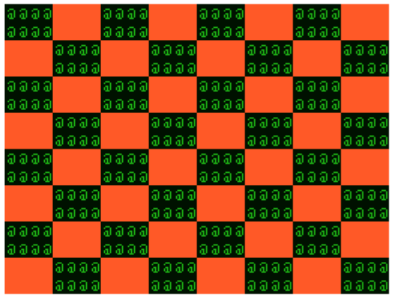
2.7 Troubleshooting

2.7.1 No BASIC ROM

The most common issue when first using XRoar. You start the emulator and only see a checkerboard pattern of orange and inverse '@' signs (or on the CoCo 3, some other pattern that's not the usual copyright messages). This probably indicates that XRoar could not locate any BASIC ROM images. Acquire some and put them in the directory appropriate to your platform.







2.7.2 Program lacks colour

You remember a program being in colour, but all you see is black and white.







American software is often written to exploit cross-colour artefacts, where alternating patterns of black and white will "trick" the TV into displaying colour. XRoar supports this, and should enable it by default when you choose an NTSC machine. See Section 3.6.2 [Composite rendering], page 13.

2.7.3 Can't access HD/SD image

If you've been using previous versions of XRoar with the IDE, MOOH, or NX32 cartridges, you now need to specify the image filename with `-load-hd0`. (HD image for IDE, SD image for NX32, MOOH).

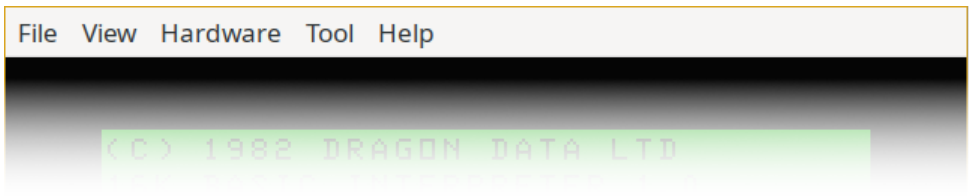
2.7.4 Debug messages

XRoar prints diagnostic messages to standard output and standard error, and these may help narrow down a problem. You can increase their verbosity with various command line options. See Section 6.15 [Debugging options], page 34, for more information.

If you do not see these messages under Windows, you may need to force the allocation of a console by specifying `-C` as the very first option.

3 User interface

This chapter walks through most of the functionality available through the graphical user interface. Where useful, short configuration examples or command line options will be demonstrated. For more detailed information on configuring XRoar through the command line or configuration file, see Chapter 6 [Configuring XRoar], page 25.



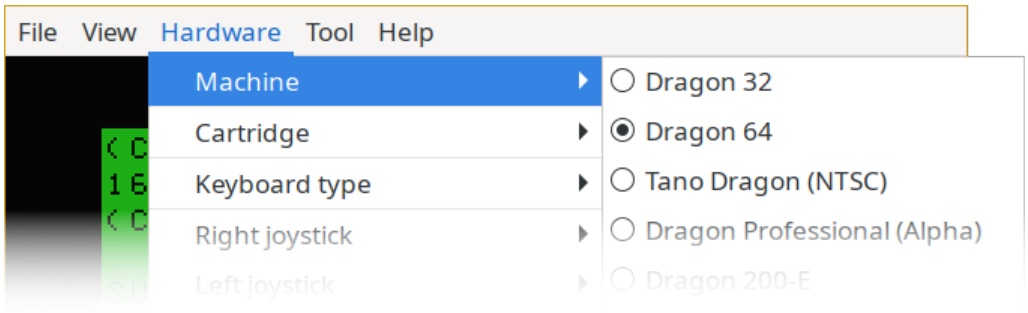
When you run XRoar, it will start emulating the default machine in a window with a menu bar above it.

From this menu bar, File lists general file options, including saving snapshots; View lists display-related functions, including opening the TV Controls dialog; Hardware lists options to modify the currently emulated hardware configuration; and Tool lists various options and control dialogs. Help contains a simple About dialog.

If you check File → Autosave configuration, some options will be automatically written back to the config file (in an OS-specific location) on exit. This is work in progress, however, and not all options are configurable from the UI. You have the most control by modifying this file. See Chapter 6 [Configuring XRoar], page 25.

3.1 Selecting a machine

The Hardware → Machine submenu allows you to select a different machine to emulate. XRoar will hard reset the new machine.



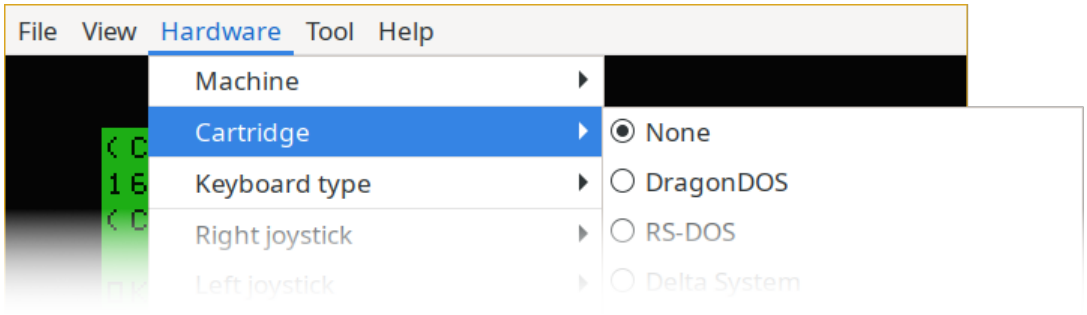
You can specify a default machine in `xroar.conf`:

```
default-machine dragon32
```

Finally, you can select a machine from the command line, e.g. `xroar -m dragon32`. Specify `-m help` for a list of profiles.

3.2 Selecting a cartridge

The Hardware → Cartridge submenu lets you select a new cartridge to be attached to the currently-running machine. XRoar will not hard reset the machine, but it is usually advisable to do so, either by selecting Hardware → Hard Reset or by pressing *CTRL+SHIFT+R*.



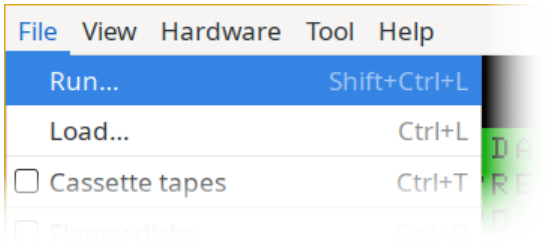
You can specify a default cartridge for a machine in `xroar.conf`:

```
machine dragon64
  machine-cart mydos
```

And you can specify a cartridge on the command line, e.g. `xroar -m dragon64 -cart mydos`. Specifying `-no-machine-cart` will prevent the usual attempt to automatically find a disk controller cartridge for a machine. Specify `-cart help` for a list of profiles.

3.3 Running programs

XRoar tries to make quickly running simple programs easy. If you want to plug in a game cartridge, attach and load from a cassette image, or even run an unadorned DragonDOS or RS-DOS binary, just select File → Run or press *CTRL+SHIFT+L*. To attach media without trying to autorun a program, select File → Load or press *CTRL+L*. Alternatively, open one of the media control dialogs; see Section 3.4 [Cassette tape control], page 11, or Section 3.5 [Drive control], page 12.



From the command line, you can use `-load` *file* or `-run` *file* to achieve the same results. You can even simply specify a filename as the last option to try and run it:

```
xroar -m cocous daggorath.rom
```

XRoar will decide how to treat the file you select based on its extension:

.cas, .c10, .wav, .k7
: Cassette image. XRoar will attach the image as the input cassette and try and look for the first file on the tape. Depending on its type, it will automatically type '`CLOADM`' (machine code) or '`CLOAD`' followed by '`RUN`' (BASIC). Some programs with special load instructions are recognised automatically.

.bas, .asc
: ASCII BASIC program. The Dragon (and Tandy Colour Computer) ROM has the ability to save and load BASIC in an untokenised form. It still requires saving as a series of short blocks though, so if you load one of these, XRoar will automatically simulate this format to load with '`CLOAD`'.

.rom, .ccc Cartridge image. XRoar will create and insert a ROM cartridge with this file as its ROM data. Some cartridges aren't simple ROM images; XRoar can automatically recognise some of these and decide whether a Games Master Cartridge should be used, for example.

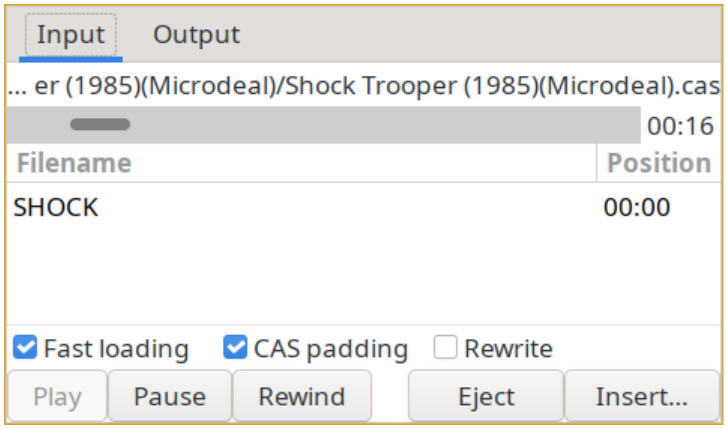
.vdk, .dsk, .jvc, .os9, .dmk
Floppy disk image. XRoar will insert the floppy and type '`BOOT`' (or '`DOS`' under RS-DOS). If the result is '`?BT ERROR`' (DragonDOS) or a clear screen (RS-DOS) and nothing else happening, that just means there was no boot track. You'll have to follow the load instructions for the software in question, which vary too much for XRoar to have built-in rules for.

.bin A DragonDOS or RS-DOS binary file. XRoar will determine which type it is from the header and load it into RAM, then tell the CPU to jump to its start (EXEC) address.

In the case of floppy disk images in particular, there are many different formats, and sometimes you see files in one format with the file extension of another (e.g. just a generic `.dsk`, which XRoar will assume is a simple sector dump). If you have issues, do check your file extensions.

This is not an exhaustive list of the types of file XRoar can make use of, just the ones it knows how to automatically load programs from.

3.4 Cassette tape control



Select File → Cassette tapes or press *CTRL+T* to open the cassette tape control dialog.

In this dialog you can insert a tape, eject the current tape, and control the cassette motor with play and pause controls. Note that for machines without motor control (MC-10, Alice), the tape will default to paused, and you will have to manually press play here after typing the load command.

In addition, you can rewind the tape or specifically set its position by dragging the scroll bar beneath the image filename. XRoar will also scan a tape file for Dragon programs, allowing you to double click a program filename to seek to it directly.

The input and output tapes can be managed separately - click on their tab names to move between them. While this isn't a configuration that you'd often have in real life, it does make some uses of the emulator more convenient.

In the input tab, you can also see some tick boxes for setting emulator options:

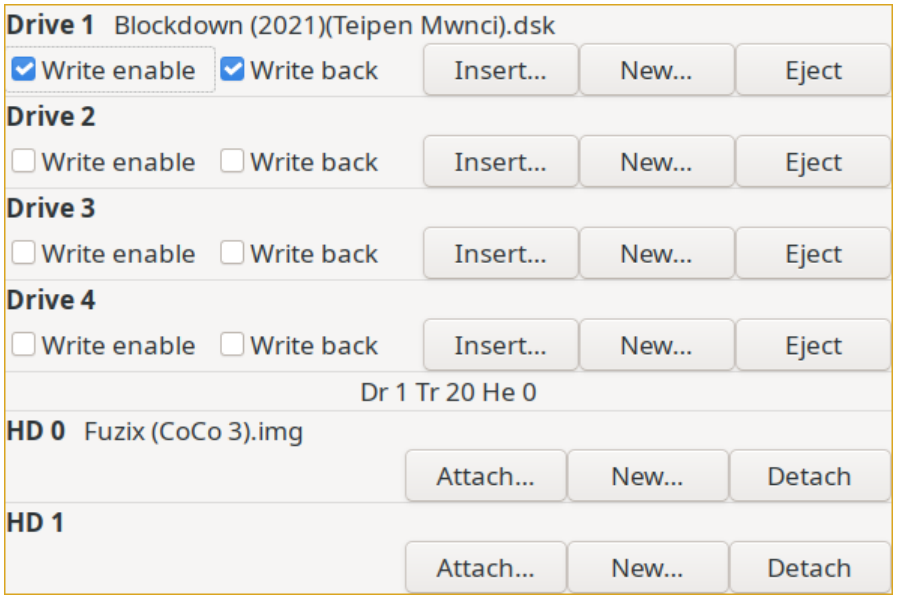
Fast loading
XRoar intercepts ROM calls to speed loading.

CAS padding
: XRoar automatically inserts extra leader if a CAS file doesn't seem to have enough. This is common with early tape conversions.

Rewrite
: XRoar intercepts ROM calls to rewrite any data read from the input tape to the output tape in a sanitised format, with consistent leaders, gaps, and byte-aligned data. Useful for creating CAS files from WAV input, or even to regenerate a cleaner WAV file.

Under Mac OS X+, most functionality is found in the File → Cassette menu.

3.5 Drive control



Select File → Floppy/hard disks or press *CTRL+D* to open the drive control dialog.

In this dialog you can insert existing, create new, or eject floppy disk images from each of four emulated floppy drives. Note that images are loaded into RAM; writes go to the in-RAM copy. The image is written back to the file when ejected (or you quit the emulator). The currently selected drive, track, and head are also displayed.

There are tick boxes per inserted disk to control emulator behaviour:

Write enable
: Basically the "write protect label" (or tab) on a floppy disk. Unticking this prevents the emulated system from writing to the disk.

Write back
: Unticking this box means that whatever changes are made to the in-RAM copy will not be written back to the floppy image file.

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.

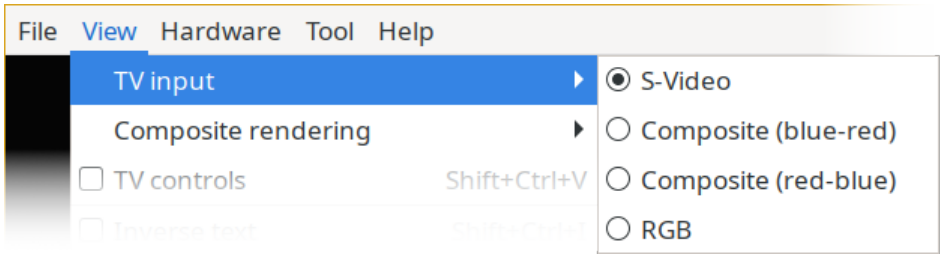
You can also attach, create new or detach up to two hard disk images. How these operate will depend on what hardware is using the image.

Under Mac OS X+, floppy and hard disk options can be found under per-drive submenus of the File menu.

3.6 Video options

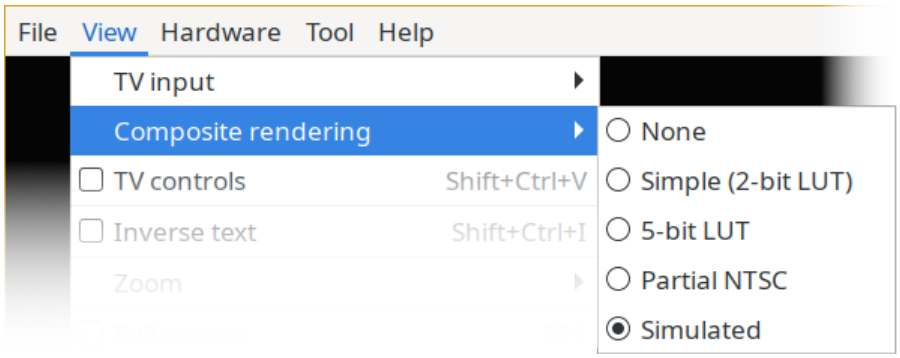
The View menu contains various video-related options. Zoom and Full screen are fairly self-explanatory. Inverse text simulates a common Dragon hardware modification to invert the colours only in text mode. The rest need a bit more explanation.

3.6.1 TV input



The TV input menu lets you pick between various video signals from the emulated machine. "S-Video" is basically a palette-based output, reflecting the way an S-Video cable avoids cross-talk between the components of a video signal. "RGB" is only currently useful on the CoCo 3, which emits a completely different set of colours on its RGB port. The two "Composite" options employ a composite video renderer (see below), with two pixel-to-colour phase relationships; VDG-based systems tended to come up in one or the other at random, so this lets you pick the "other" one if a title expects it.

3.6.2 Composite rendering



XRoar has a selection of composite renderers built in that trade off CPU time for accuracy in different ways to reproduce the artefacts of composite video.

None Disable all composite video effects. Output will be simple colours, the same as picking the "S-Video" TV input.

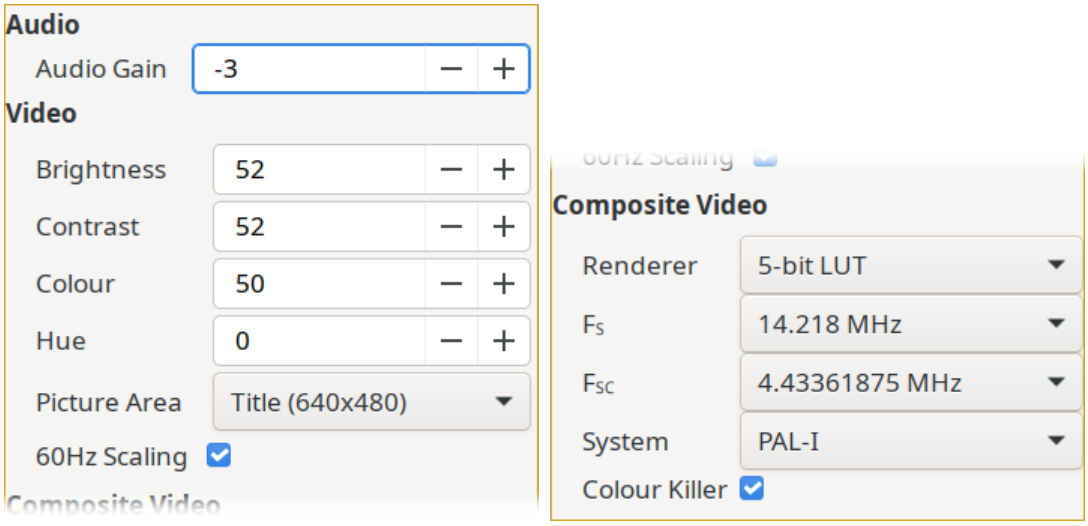
Simple (2-bit LUT)
Gives a very coarse 4-colour rendition of NTSC artefact colours between black & white pixels only.

5-bit LUT A better set of NTSC artefact colours using a 5-bit LUT, but still only between black & white pixels.

Partial NTSC
Performs a composite video encode/decode chain with filtering, taking lots of short-cuts. Pretty decent NTSC output, but doesn't handle PAL.

Simulated Performs a more complete composite video encode/decode chain with filtering. This is the only renderer that will accurately reproduce PAL colour effects. It is also the most CPU-hungry.

3.6.3 TV controls



Select View → TV controls or press *CTRL+SHIFT+V* to open the TV controls dialog. The dialog is divided into three sections. The first, Audio, only contains a gain control. The Video section has the following controls:

Brightness
: Or black level. 0–100, where 0 pulls everything to black, and 100 drives everything up to white, effectively adjusting the *contrast* of the picture.

Contrast
: Or gain. 0–100, with 50 meaning no gain. Scales the RGB output, affecting the overall *brightness*. See Charles Poynton on Brightness & Contrast (`https://poynton.ca/notes/brightness_and_contrast/`) for more on the labelling issue.

Colour
: Adjusts the saturation of colour without affecting the luminance. 0–100, where 0 yields a greyscale image.

Hue
: Adjusts the phase of a composite colour signal. -179–180, with the centre value of 0 being normal. A common control on NTSC displays. Not strictly necessary in an emulator, but it's fun to spin it and watch the pretty colours.

Picture Area
: The picture area is the portion of the output signal that is rendered into the window, and this control lets you show more or less border around the active area (picking from four different crop regions).

60Hz Scaling
: When this is enabled, 60Hz output (NTSC, PAL-M) will be scaled vertically, to better represent the picture shape that would be seen on 60Hz CRTs. This scaling occurs because in a 60Hz TV system, fewer scanlines fill the same vertical space, yet the scanline duration is very similar.

Finally, the Composite Video section of the dialog includes more advanced control over the rendering when a composite signal is being simulated.

Renderer
: Another place to pick the composite renderer, described above.

F_S
: The video sample rate when using the "Simulated" renderer. The options here reflect the different crystals used as SAM oscillators in Dragons and CoCos.

F_{SC}
: The frequency of the colour subcarrier, either 4.43MHz (PAL-I) or 3.58MHz (NTSC, PAL-M). Together with F_S, this affects how the "Simulated" renderer tracks colour modulation and demodulation.

System The TV colour system in use, one of '`PAL-I`' (e.g. UK), '`PAL-M`' (e.g. Brazil) or '`NTSC`' (e.g. USA or Canada). Setting only used by the "Simulated" renderer.

Colour Killer
Television sets vary in how they react to a composite video signal that lacks a colourburst. The recommended approach is to disable colour processing and show a greyscale picture (to be backwards-compatible with black & white broadcasts), but many don't bother and end up decoding luminance information as colour. This option lets you choose a behaviour.

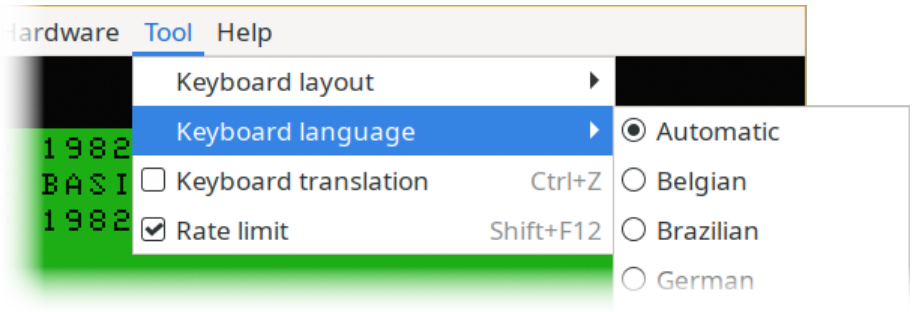
3.7 Keyboard layout

XRoar can operate in one of two keyboard modes. By default, it tries to map the key positions on the host keyboard to what you would expect using the emulated system. This is useful for games (where the relative physical positions of keys will be important), or to just satisfy muscle memory (even using an emulated Dragon, my hands expect to type in a certain way).

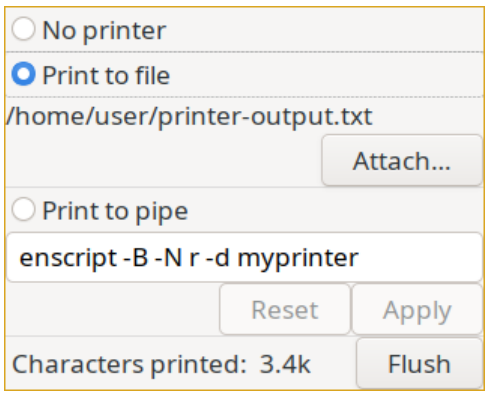
You can select a particular layout with Tool → Keyboard layout, though at time of writing, this doesn't affect very much.

In translated mode (Tool → Keyboard translation, or *`CTRL+Z`* to toggle), XRoar instead tries to map the symbols on your host keyboard to the keystrokes required to produce that symbol on the emulated machine. For example, a UK PC keyboard typically has an unshifted `apostrophe` key, which XRoar can map to pressing *`SHIFT+7`* on an emulated Dragon.

XRoar queries your operating system for information about which symbol is on which key, but in case it gets it wrong, you can pick from several built-in languages using Tool → Keyboard language.



3.8 Printer control



Select File → Printer control or press *CTRL+P* to open the printer control dialog.

You can select between no printer, printing to file, or (only in the GTK+ interface under Linux/Unix), printing through a pipe to an external filter command.

When modifying the pipe configuration, your changes will not take effect until you press Apply.

The amount of characters printed since the last time the buffer was flushed is shown. Pressing the Flush button (or pressing *CTRL+SHIFT+P*) will flush output to file, or close and reopen the pipe. The act of closing the pipe will cause the running filter to complete, e.g. sending a job to your network printer.

4 Emulated hardware

This chapter gives a brief description of the hardware emulated by XRoar, with configuration examples where useful. For a more complete view of the user interface, see Chapter 3 [User interface], page 9. For more detailed information on configuring XRoar through the command line or configuration file, see Chapter 6 [Configuring XRoar], page 25.

4.1 Machine architectures

XRoar supports several underlying machine architectures, and has one or more built-in machine profile configurations based on each one. See Section 6.2 [Machine options], page 26, for more detailed information on modifying or creating profiles. The rest of this section describes the available architectures.

4.1.1 Dragon 32

Released in 1982, the Dragon 32 closely follows Motorola's reference design for the MC6809 CPU, MC6883 Synchronous Address Multiplexer and the MC6847 Video Display Generator. Dragon Data also chose to make it electrically compatible with some of Tandy's peripherals for their Colour Computer; notably the joystick and cartridge ports. In addition, it has a parallel port, making it compatible with the majority of printers on the market at the time.

Architecture '`dragon32`'. Built-in machine profile '`dragon32`'.

4.1.2 Dragon 64

The Dragon 64 was released the next year, in 1983. In upped the on-board RAM to 64K and provided a second version of Microsoft BASIC assembled to make use of it. It also added a serial port, though that is not yet emulated by XRoar.

There are a few more changes to the motherboard than just extra RAM, so XRoar treats this as a separate architecture.

If you specify 512K RAM, the SAMx8 memory expansion is emulated.

Architecture '`dragon64`'. Built-in machine profiles: '`dragon64`', '`tano`' (American NTSC version of Dragon 64 by Tano), '`dragon200e`' (localised Spanish Dragon 64 from Eurohard), '`dragonx8`' (Dragon 64 with SAMx8 fitted).

4.1.3 Dragon Professional

A prototype machine by Dragon Data that was never actually released. A few versions of the system in cases at various stages of development exist.

Essentially a souped-up Dragon 64, it adds an AY sound chip and built-in twin 3.5" drives. An early mention of it appears in the June 1984 issue of Dragon User, and there was a picture in the next issue.

Support is largely based on information from Phill Harvey-Smith.

Architecture '`dragonpro`'. Built-in machine profile: '`dragonpro`'.

4.1.4 Tandy Colour Computer 1/2

An earlier (1980) Tandy machine made using Motorola's reference design, primarily marketed in the USA. Sold at many price points, with 4K (originally), 16K, 32K or 64K of RAM and either with or without Extended Colour BASIC. Later versions come with a new version of the VDG, the MC6847T1, which includes true lowercase characters.

Architecture '`coco`'. Built-in machine profiles: '`coco`', '`cocous`' (NTSC), '`coco2b`' (T1), '`coco2bus`' (NTSC, T1), '`mx1600`' (Mexican clone by Dynacom).

4.1.5 Tandy MC-10

Released in 1983, a little too late to compete with the Sinclair ZX-81, it was discontinued a year later. A cut-down machine based on the Motorola MC6803, but still using the MC6847 VDG and containing a version of Microsoft BASIC. Comes with 4K of RAM, but much of the small amount of software available for it assumes an additional 16K RAM pack.

Architecture '`mc10`'. Built-in machine profile '`mc10`'.

4.1.6 Matra & Hachette Alice

Basically the same machine as an MC-10, but with a French keyboard, 50Hz display, and a nice bright red case. Unlike the MC-10, the Alice line actually continued, with the Alice 32 and Alice 90, though these are not supported by XRoar, as their architectures differ significantly.

Architecture '`mc10`'. Built-in machine profile '`alice`'.

4.1.7 Tandy Deluxe Colour Computer

In development around 1984, this was never actually released. However, a prototype has been discovered with a copy of Advanced BASIC and the planned extended hardware.

Broadly similar to a Colour Computer 2, it adds finer-grained RAM banking, an AY sound chip, and an ACIA, amongst other things.

Support is a work in progress based on what we know so far.

Architecture '`deluxecoco`'. Built-in machine profile: '`deluxecoco`'.

4.1.8 Tandy Colour Computer 3

In 1986, Tandy released the Colour Computer 3. They had developed a custom chip, the TCC1014 (*GIME*), with VLSI to replace the SAM and VDG, and it supported extended graphics modes, more memory (up to 512K directly) and a timer function, along with somewhat better interrupt handling and the ability to run at twice the clock speed. A major development, it maintained a high degree of compatibility with its predecessors, losing some lesser-used (in the USA) graphics modes.

The NTSC version of the CoCo 3 generates different colours depending on whether you use the composite video or RGB outputs. The PAL version always uses the RGB output from the GIME.

If you specify 1024K or 2048K RAM, this enables an optional DAT board function that extends the range of the MMU registers by two bits. For compatibility with early 2M boards, these two bits are write-only by default; specify `-machine-opt readable-dat` to make them readable.

Architecture '`coco3`'. Built-in machine profiles: '`coco3`' (NTSC), '`coco3p`' (PAL).

4.2 Cartridge types

XRoar supports several types of cartridge, and has at least one built-in cartridge profile configurations for each one. See Section 6.3 [Cartridge options], page 27, for more information on modifying or creating profiles. The rest of this section describes the available types.

4.2.1 DragonDOS

The official Dragon Data disk system for the Dragon. Supports 80 track, double sided, double-density floppy disks.

Emulation supports the Becker port mapped to $FF49/$FF4A, if enabled.

Type '`dragondos`'. Built-in cartridge profile '`dragondos`'.

4.2.2 Delta

Premier Microsystems' alternative Dragon disk system. Apparently two versions of this may have existed; XRoar emulates the double-density version.

Type '`delta`'. Built-in cartridge profile '`delta`'.

4.2.3 RS-DOS

Tandy's disk interface for the CoCo. Typically supports only 35-track single-sided double-density disks, though more is accessible using OS-9.

Emulation supports the Becker port.

Type '`rsdos`'. Built-in cartridge profile '`rsdos`', '`becker`' (with Becker port enabled, expecting `hdbdw3bck.rom`).

4.2.4 Glenside IDE controller

Interfaces the Tandy CoCo to up to two IDE hard disks. Its IO is generally memory mapped to addresses $FF50–$FF58. Also optionally supports the Becker port.

To set the base address to some other value (the original cartridge can jumper IO to be from $FF70–, but this is incompatible with the MPI), use the `-cart-opt ide-addr=`*addr*.

The controller supports up to two drives, and you can specify the image to use in each with `-load-hd0` *file* or `-load-hd1` *file*. If *file* does not exist, a 256MB empty image is created when the controller first tries to access it.

Sectors are 512 bytes, and while some software may use all 512, others only access 256 bytes per sector, padding the other 256 bytes (or simply doubling them up).

Type '`ide`'. Built-in cartridge profiles: '`ide`', '`ide-dragon`'.

The '`ide`' profile is for CoCos and will default to trying to use YADOS or HDBLBA.

The '`ide-dragon`' profile specifies a simple built-in ROM compatible with Dragon BASIC that allows you to BOOT from the IDE disk. See Section 7.4.1 [BSDOS], page 38.

4.2.5 NX32 and MOOH cartridges

Two memory expansion cartridges created by Tormod Volden for the Dragon. Both accept an SD card image.

The earlier NX32 provides simple bank switching, while the MOOH provides MMU-like functionality very like that in the Tandy CoCo 3.

Types '`nx32`', '`mooh`'. Built-in cartridge profiles: '`nx32`', '`mooh`'. Both require fleshing out with ROM information, and an SD card image specified, e.g.:

```
cart mooh
  cart-rom sdbdos-eprom8-all-v1.rom

load-hd0 "~/sdcard.img"
```

4.2.6 Games Master Cartridge

The Games Master Cartridge (GMC), created by John Linville, provides the ability to bank switch up to 64K of cartridge ROM, along with an on-board SN76489 sound chip.

This cartridge type is selected automatically (and configured to autostart) if you autorun a ROM image larger than 16K.

Type '`gmc`'. Built-in cartridge profile '`gmc`' is configured with no ROM installed, and to not auto-start.

4.2.7 Orchestra 90-CC sound cartridge

A simple expansion that provides two 8-bit DACs for stereo sound (but still driven by the CPU). An on-board ROM for the CoCo includes an application for composition and playback, but if autorun is disabled, the hardware itself works fine on the Dragon.

Type '`orch90`'. Built-in cartridge profile '`orch90`'.

4.2.8 Multi-Pak Interface

The Multi-Pak Interface (MPI) is a CoCo add-on by Tandy that allows up to four cartridges to be connected, selectable by software or hardware switch.

The RACE Computer Expansion Cage is a Dragon add-on by RACE similar to the MPI. Addressing and behaviour differs.

If you attach an 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.

Types '`mpi`', '`mpi-race`'. Built-in cartridge profiles: '`mpi`', '`mpi-race`' (RACE variant).

```
machine coco
  machine-cart mpi

cart mpi
  mpi-load-cart 0=orch90
  mpi-load-cart 3=rsdos
  mpi-slot 3
```

4.2.9 MC-10 MCX128 memory expansion

Designed by Darren Atkinson, provides 128K of bank-switched RAM for the Tandy MC-10. Can also optionally switch in its own enhanced BASIC ROM.

Type '`mcx128`'. Built-in cartridge profile '`mcx128`'.

4.3 Internal add-ons

XRoar also supports some of the more invasive internal modifications that add functionality to a base machine.

4.3.1 SAMx8 512K memory expansion

The SAMx8 is a 512K SRAM expansion and MMU for the Dragon 64. It enables arbitrary mapping of 16K pages in two tasks, optionally making either or both of the top two 4K regions common to all memory maps.

By replacing the SAM and interposing the video RAM data latch, it can completely replace on-board memory. Combined with high speed SRAM, this also allows the system to run in double speed mode while retaining video display.

The SAMx8 can be configured for the following machine architectures: '`dragon32`', '`dragon64`', '`dragonpro`', '`coco`', '`deluxecoco`'.

Specify explicity with `-machine-opt sam=samx8`, or implicitly with `-ram 512`.

4.3.2 iMMUnity memory expansion

The iMMUnity expansion provides 2MB of SRAM for the Dragon and CoCo 1/2. It includes CoCo 3 GIME-like arbitrary mapping of 8K pages in two tasks, and allows access to the memory on the motherboard, which is still used for video.

During development, the iMMUnity was known as the CoCoMEM Jr.

The iMMUnity can be configured for the following machine architectures: '`dragon32`', '`dragon64`', '`dragonpro`', '`coco`', '`deluxecoco`'.

Specify explicity with `-machine-opt immunity`, or implicitly with `-ram 2048`.

4.4 Keyboard

The Dragon keyboard (and those of all the machines XRoar emulates) is a typical crosspoint matrix with rows and columns connected to an internal interface. BASIC or other programs detect keypresses by strobing values to one port and seeing whether those values can be seen at another.

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 the terms "key rollover" or "NKRO").

By default, XRoar maps host keys to emulated keys based on their position. By enabling keyboard translation (Tool → Keyboard translation or *CTRL+Z* to toggle), it will instead press the emulated keys required to generate the appropriate *symbol*. Be aware that this only works in BASIC; OS-9 uses different chords for some characters. Use the `-kbd-translate` option to default to this mode.

Where a PC keyboard doesn't typically have a good equivalent of an emulated key, some substitutions are made: `Escape` maps to the Dragon's `BREAK` key, and `Home` maps to `CLEAR`. Cursor keys are mapped directly, although they are in different places on a PC keyboard.

Sometimes software written for the Dragon or Tandy CoCo is run on the other machine without fully adapting it to the different keyboard matrix layouts. In this case, you can toggle to the "other" machine's keyboard layout by pressing *CTRL+K* or selecting one from Hardware → Keyboard type.

See Section 6.8 [Keyboard options], page 29, for more details on configuring keyboard handling, including some options that may help if XRoar was unable to properly identify your host keyboard.

4.5 Joysticks

Analogue joysticks are very common peripherals for the Dragon and Tandy CoCo. Many games require them, and some productivity applications even use them as a mouse-like input device. Joysticks are electrically compatible between the Dragon and CoCo 1/2/3, though some CoCo joysticks use a 6-pin DIN connector instead of 5-pin DIN, and these will not plug into the Dragon. On the CoCo 3, this extra pin can carry the signal for an extra firebutton.

XRoar can use physical gamepads and joysticks, or simulate them using mouse or keyboard. Here are the built-in joystick profiles, including several different keyboard layouts for convenience:

Name	**Description**
'`mjoy0`'	Mouse based virtual joystick mapped to screen position
'`kjoy0`'	Keyboard based virtual joystick using cursor keys, with `Left Alt` and `Left Super` as firebuttons.
'`wasd`'	Keyboard based virtual joystick using `W`, `A`, `S`, `D`, with `O` and `P` as firebuttons.
'`ijkl`'	Keyboard based virtual joystick using `I`, `J`, `K`, `L`, with `X` and `Z` as firebuttons.

'`qaop`' Keyboard based virtual joystick using `Q`, `A`, `O`, `P`, with `Space` and `Left Bracket` as firebuttons.

In addition, XRoar should automatically detect attached gamepads or joysticks and add them as named profiles called '`joy0`', '`joy1`', etc. If the device is able to provide its own description, that will be included in the menu text. If a gamepad has two sticks, two more automatic profiles will be created, e.g. '`joy0/l`' and '`joy0/r`' so you can choose between them.

By default, XRoar will map the first physical joystick found ('`joy0`') to the right hand port, and the second ('`joy1`') to the left hand port. If only one device is present, but has two sticks, the *left* stick ('`joy0/l`') is mapped to the *right* hand port and vice-versa. Slightly confusing, but this is because people usually assume the left stick to be active, while the right hand port on a Dragon is the first in hardware, thus usually the "primary" control assumed by software.

A preselected virtual joystick can be quickly cycled through the ports by pressing *CTRL+J*. The first press will map it to the right joystick, the second to the left joystick, and pressing a third time unmaps it. You can change which virtual joystick is cycled in this way with the `-joy-virtual` *name* option, defaulting to '`kjoy0`' from the table above.

The MC-10 has no built-in joystick ports, but an expansion (that can not be used at the same time as the 16K RAM expansion!) allows the connection of digital joysticks. These are not yet supported by XRoar.

See Section 6.9 [Joystick options], page 30, for information on configuring your own joystick profiles.

4.6 Printers

The Dragon machines have parallel printer ports, and XRoar supports these, sending output either to a file, or through a command pipe. The pipe approach allows you to apply a filter to the output, and/or send it to a real attached printer using normal Unix commands.

The CoCo and MC-10 machines have serial printer ports. XRoar doesn't support these directly yet, but a limited form of print redirection is implemented 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.

Use the `-lp-file` *file* option to send printer output to a file, or `-lp-pipe` *command* to send it through a pipe. Pressing *CTRL+SHIFT+P* will flush the current stream by closing it, so if you are using a pipe, the filter will complete. The stream will be re-opened when any new data is sent.

Under Unix, the `enscript` utility is good for processing output and sending it to a configured printer, 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).

5 Storage media

This chapter documents the types of media image files that XRoar supports, and some of the options for manipulating them. For a more complete view of the user interface, see Chapter 3 [User interface], page 9. For more detailed information on configuring XRoar through the command line or configuration file, see Chapter 6 [Configuring XRoar], page 25.

5.1 Cassettes

Cassette tape was the primary method of loading software until floppy disk drives became available, but has remained popular for games distribution even since, as it serves the largest market. Data is encoded onto cassette tape as audio, all currently-emulated machines using the same format, where a single cycle represents one bit of data, and its wavelength determines the bit's value.

5.1.1 Tape image file formats

XRoar supports tapes as raw sampled audio in WAV format (`.wav`), or in the more compact CAS format (`.cas`) which represents bits of data directly (files for the MC-10 are typically still CAS format, but with a `.c10` extension; these will also work).

An extension to the CAS format called CUE is also supported. This comprises extra data at the end of the file that *marks up* the CAS file to indicate portions of silence, or the wavelength used for each bit. This enables it to better represent the structure of the original tape, support certain fast loaders, yet for data within the file to remain readable with a hex editor if it is correctly aligned.

Some MC-10/Alice software has been seen in K7 format (`.k7`). XRoar has read-only support for these files.

XRoar can also attach BASIC ASCII text files (with `.bas` or `.asc` file extensions) and interpret them as cassettes, providing a useful way to edit these in your favourite text editor before loading into the emulator. Note: this feature is not supported by the MC-10.

5.1.2 Input and Output tapes

The tape used for writing is considered separate to the read tape. This is an emulator-friendly approach to prevent overwriting your programs, though it would of course be possible in real life with two cassette decks.

Tape rewriting, enabled with `-tape-rewrite` is a special mode where the ROM is intercepted, and anything read from the input tape is *rewritten* to the output tape. Custom loaders may defeat it, but otherwise this is a good way of creating a well-formed CAS file, with bytes aligned and consistent leader lengths.

5.1.3 Remote motor control

The Dragon and Tandy Colour Computers have a built-in cassette relay that can control the cassette motor remotely. For these platforms, cassette emulation will default to "play" being pressed, letting the remote control start and stop the tape.

The MC-10 and Alice have no remote connection, and so for these platforms cassette emulation defaults to stopped, and you will have to manually start and stop the tape after entering load commands.

5.2 Floppy disks

Floppy disk drives provide much faster access to data than cassette tape. Initially costly, prices did fall somewhat, so these are a fairly common expansion.

To use floppy disk images, an emulated disk controller will need to be configured. XRoar will usually try to do this automatically if it finds the appropriate ROMs for a disk controller suited to the machine you've chosen.

These floppy disk image 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.

5.3 Hard disks

The Glenside IDE controller interfaces hard disks to the Tandy CoCo, and the MOOH and NX32 memory expansions can each provide access to an SD card.

`-load-hd`*`X`* *`file`* Use *file* as the hard disk image for drive *X* (0 or 1).

XRoar supports these types of hard disk image:

IDE images with header information should have a `.ide` extension. This is necessary to distinguish them from images with no header. These images contain metadata describing an IDE drive, and are the only ones usable in CHS mode. Starting with 512 bytes of "magic" and 512 bytes of IDENTIFY DEVICE information, sector data follows in LSN order, 512 bytes per sector.

Raw images with 512 byte sectors should have a `.img` extension. Previous versions of XRoar would create IDE images with this file extension, and you should rename them to `.ide`. Unfortunately this is necessary to support raw images without a header: if you happened to write the "magic" identifying information to the start of it, any attempt to be clever about file contents would fail.

Finally, files with the `.vhd` extension are assumed to be 256 bytes per sector with no header information.

When no IDE metadata is present, XRoar will fake some up so that raw images can still be used with the IDE controller emulation. This means VHD images containing RSDOS filesystems are usable with YA-DOS or HDBDOS.

6 Configuring XRoar

XRoar can read a configuration file. The file is ASCII text, with one directive per line. It covers anything from setting individual options to defining specific named profiles for hardware.

To recap, the expected location for this file varies by platform:

Platform	Default path to configuration file
Unix/Linux	`~/.xroar/xroar.conf`
Windows	`%LOCALAPPDATA%\XRoar\xroar.conf`
Mac OS X+	`~/Library/XRoar/xroar.conf`

To print the current configuration to standard output (suitable for redirection to a config file), run XRoar with the `-config-print` option. This will include all the built-in machine and cartridge definitions. For a complete version including default values, use `-config-print-all`. To enable limited configuration auto-save (to the location listed in the table above), use `-config-auto-save`.

The Windows build will check the current working directory for `xroar.conf` first. This is somewhat at odds with configuration auto-save, but as far as I can tell, the majority of Windows users prefer to unpack a `.zip` file and run it from its own directory. Just be aware that under Windows, if you turn on auto-save, the version in the unpacked directory will still take precedence.

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

Directives are listed in `xroar.conf` one per line. They contain an option, possibly followed by whitespace and a value. Trailing whitespace is ignored. Empty lines are skipped, and any line where the first non-whitespace character is a hash (‘`#`’) is treated as a comment. The leading dash (‘`-`’) is not required in the configuration file, though it is accepted.

If a value contains special characters, or if you want trailing whitespace to be included in the value, you must *escape* those characters. Sections contained within pairs of single or double quotes are escaped, except the backslash (‘\’) which introduces an escape sequence:

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 either mapped to *`BREAK`* or used to introduce limited ANSI escape sequences in the `-type` command, effective for the MC-10.
‘`\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.
‘`\`*`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—including another backslash—is included verbatim. For example, this will be necessary in the configuration file under Windows when file paths include the backslash as a directory separator.

6.1 Startup options

These may only be used on the command line, and must be specified before any other option, as they affect how the rest are treated.

`-C`	Force creation of a console window if attach to parent fails (Windows-only).
`-no-C`	Do not attempt to use any console (Windows-only).
`-c` *file*	Specify a different configuration file.
`-no-c`	Don't read the configuration file.
`-no-builtin`	Disable built-in configuration. Unless you also define a machine yourself, XRoar will abort.

6.2 Machines

XRoar creates a list of machine profiles from built-in and user-supplied configuration. One of these profiles is selected at startup, using either the `-default-machine` *name* option, or by XRoar testing each profile in turn to see if its configured ROM image files are available.

Each machine profile has a base architecture (specified with the `-machine-arch` option). See Section 4.1 [Machine architectures], page 17, for descriptions of the supported architectures, and which machine profiles are built-in.

`-default-machine` *name*	Default machine profile to select on startup.
`-m` *name*, `-machine` *name*	Create or modify named machine profile. The remaining options configure the profile. `-machine help` lists currently defined profiles.
`-machine-desc` *text*	Description shown in `-machine help` and menu options.
`-machine-arch` *arch*	Base machine architecture. See Section 4.1 [Machine architectures], page 17, for information. '`dragon32`', '`dragon64`', '`dragonpro`', '`coco`', '`deluxecoco`', '`coco3`' or '`mc10`'.
`-machine-keyboard` *type*	Override the type of keyboard attached to machine. '`dragon`', '`dragon200e`', '`coco`', '`coco3`', '`mc10`' or '`alice`'.
`-machine-cpu` *cpu*	Fitted CPU. One of '`6809`' or '`6309`'. Not applicable to the MC-10.
`-bas` *rom*	ROM image for Colour BASIC (CoCo) or Microcolour BASIC (MC-10, Alice).
`-extbas` *rom*	ROM image for Extended BASIC (Super Extended BASIC on the CoCo 3).
`-altbas` *rom*	ROM image for 64K-mode Extended BASIC (Dragon 64, Dragon 200-E).
`-no-bas`, `-no-extbas`, `-no-altbas`	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`'.
`-tv-input` *input*	One of '`cmp`' (composite video, no cross-colour), '`cmp-br`' (composite video, blue-red cross-colour), '`cmp-rb`' (composite video, red-blue cross-colour) or '`rgb`' (RGB video, CoCo 3 only).

`-vdg-type` *type*	Indicate the VDG variant fitted. One of '`6847`' or '`6847t1`'. For CoCo 3, '`gime1986`' or '`gime1987`'.
`-ram` *kbytes*	Amount of RAM fitted in kilobytes. Nearest value valid for selected machine will actually be used.
`-ram-org` *org*	Override RAM type, possibly affecting banking behaviour. One of '`4kx1`', '`16kx1`', '`32kx1`' or '`64kx1`'.
`-ram-init` *pattern*	Initial RAM state. One of '`clear`', '`set`', '`pattern`' or '`random`'.
`-machine-cart` *name*	Default cartridge to attach.
`-no-machine-cart`	Indicate that XRoar is not to automatically attempt to attach a DOS cartridge to this machine (the default is to try).
`-machine-opt` *string*	Set machine arch-specific option.

Here is a configuration example, approximating the prototype that led to the Dragon 32:

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

'`pippin`' is the short name used to refer to the profile. The argument to `machine-desc` is the longer descriptive name that would appear in menus or help text. The rest of the section configures the new machine to have a base architecture the same as a Dragon 32, but with only 16K of RAM.

6.3 Cartridges

Similarly, XRoar contains a list of cartridge profiles, each with an underlying type.

`-cart` *name*	Create or modify named cartridge profile. `-cart help` lists currently defined profiles. The remaining options configure the profile.
`-cart-desc` *text*	Cartridge description shown in `-cart help` and menu options.
`-cart-type` *arch*	Cartridge architecture. See Section 4.2 [Cartridge types], page 18, for a list.
`-cart-rom` *file*	The ROM image specified will be mapped from $C000.
`-cart-rom2` *file*	The ROM image specified will be mapped from $E000.
`-cart-becker`	Enable Becker port where supported.
`-cart-autorun`	Auto-start cartridge using FIRQ.
`-cart-opt` *string*	Set cartridge type-specific option.
`-mpi-slot` *slot*	(MPI) Initially select slot (0–3).
`-mpi-load-cart` [*slot*]=*name*	(MPI) Insert cartridge into next or numbered slot.

Built-in cartridge profiles exist with sensible defaults for each of the cartridge types except '`rom`' (for which a profile is simply created when you try to autorun a ROM image), each with the same name as the type.

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 `-no-machine-cart` 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*).

Here is an example profile, replicating the modified DragonDOS cartridge I used to use:

```
cart mydos
  cart-desc "My SuperDOS E6 Cart"
  cart-type dragondos
  cart-rom sdose6.rom
  cart-rom2 dosdream.rom
```

'`mydos`' is the short name used to refer to the profile. The argument to `cart-desc` is the longer descriptive name that would appear in menus or help text. The rest of the section defines a cartridge with DragonDOS hardware but with the DOS ROM replaced by `sdose6.rom` (SuperDOS E6, a common upgrade). The extra 8K of cartridge address space is used for `dosdream.rom` (DOS-Dream, an editor/assembler/debugger package designed to coexist with DragonDOS).

6.4 Becker port

`-becker`	Prefer becker-enabled DOS cartridge when picked automatically.
`-becker-ip` *address*	Address or hostname of DriveWire server. Default: '`127.0.0.1`'
`-becker-port` *port*	Port of DriveWire server. Default: '`65504`'

Not a cartridge in and of itself, XRoar supports an emulator-only feature that enables it to connect to a server using a TCP connection and 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 many cartridge types.

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 py-DriveWire and DriveWire 4.

6.5 Cassettes

`-load-tape` *file*	Attach *file* as tape image for reading.
`-tape-write` *file*	Open *file* for tape writing.
`-tape-pan` *position*	Pan stereo input. Floating point number from '`0.0`' (full left) to '`1.0`' (full right). The default of '`0.5`' mixes the two channels equally.
`-tape-hysteresis` *pc*	Read hysteresis as percentage of full scale (default is 1%).
`-no-tape-fast`	Disable fast tape loading. The default is enabled, which uses ROM intercepts to speed up loading.
`-no-tape-pad-auto`	Disable automatic padding of short leaders in CAS files (see below).
`-tape-ao-rate` *hz*	Set tape writing frame rate to *hz* (affects audio file output, e.g. WAV). Default: '`9600`'Hz.
`-tape-rewrite`	Enable tape rewriting (see below).
`-tape-rewrite-gap-ms` *ms*	Gap length in milliseconds to write in rewrite mode (1-5000ms, default 500ms).
`-tape-rewrite-leader` *n*	Length of leaders in bytes to write in rewrite mode (1-2048 bytes, default 256).
`-snap-motoroff` *file*	Write a snapshot to *file* each time the cassette motor is switched off.

Dragon cassettes are typically recorded in mono. If you are having trouble loading from an audio file recorded in stereo, it may be useful to pan it hard to the left or right with `-tape-pan 0.0` or `-tape-pan 1.0`.

A small amount of hysteresis in the zero crossing detector simulates the same effect in hardware, and helps greatly with loading from some audio files. If you're having difficulties with a recording, adjusting this valie with `-tape-hysteresis` may help.

Tape padding defaults to on: A lot of old CAS tape images were created with their leaders truncated, and this option tries to account for that automatically. It may be useful to try turning this option off (from the UI, or with `-no-tape-pad-auto`) if you are having trouble loading something.

The `-snap-motoroff` *file* option is useful for getting a dump of the machine state at the moment a program has finished loading, but before it has started executing. If you specify *file* with a `.ram` extension, you can get a simple RAM dump, viewable in a hex editor.

6.6 Floppy disks

`-load-fd`*X* *file*	Load disk image file *file* into drive *X* (0–3).
`-no-disk-write-back`	Don't default to enabling write-back for disk images.
`-no-disk-auto-os9`	Don't try to detect headerless OS-9 JVC disk images.
`-no-disk-auto-sd`	Don't assume single density for 10 sector-per-track disks.

Warning: The default of *write back* being enabled can lead to accidental modification of your disk images. You can use the option `-no-disk-write-back` in your configuration file to protect them by default, though be aware that this also means anything "saved" to disk will be lost if you forget to re-enable it when required.

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`.

6.7 Hard disks

`-load-hd`*X* *file*	Use *file* as the hard disk image for drive *X* (0 or 1).

6.8 Keyboard

`-kbd-layout` *layout*	Specify host keyboard layout. `-kbd-layout help` for a list. Default: '`auto`'
`-kbd-lang` *lang*	Specify host keyboard language. `-kbd-lang help` for a list. Default: '`auto`'
`-kbd-bind` *hkey*`=[pre:]`*ekey*	Bind host key *hkey* to emulated key *ekey*.
`-kbd-translate`	Start up in translated keyboard mode.
`-type` *string*	Intercept ROM calls to type *string* into BASIC on startup.
`-load-text` *file*	Type the contents of *file* into BASIC on startup.

Specifying a keyboard layout (`-kbd-layout`) doesn't achieve much yet. In future, it may map certain extra keys from the Unix or JIS layouts to similarly positioned emulated keys.

The `-kbd-translate` option can be used to default to translated mode, where XRoar translates keypresses to reproduce the correct symbol in the emulated machine (only under BASIC; OS-9 uses different chords for some characters).

Specifying a keyboard language (`-kbd-lang`) overrides any key symbol mapping gleaned from the OS with a built-in table. If translated mode isn't working well for you, this option may help. `-kbd-lang help` for a list.

When binding keys with `-kbd-bind`, if the emulated key *ekey* is prefixed with '`preempt:`' or '`pre:`', this binding preempts translation; useful for modifier keys. To get *hkey* names, run with `-debug-ui 1` to enable keypress debugging and see what it reports as you type.

Special values for *ekey* are: '`colon`', '`semicolon`', '`comma`', '`minus`', '`fullstop`', '`period`', '`dot`', '`slash`', '`at`', '`up`', '`down`', '`left`', '`right`', '`space`', '`enter`', '`clear`', '`break`', '`escape`', '`shift`', '`alt`', '`ctrl`', '`control`', '`f1`', '`f2`'.

6.9 Joysticks

`-joy-db-file` *file*	Load gamepad mappings from file.
`-joy` *name*	Create or modify named joystick profile. `-joy help` lists currently defined profiles.
`-joy-desc` *text*	Joysticks description shown in `-joy help`.
`-joy-axis` *axis=input*`:[`*args*`]`	Configure joystick axis. `-joy-axis help` to list physical joysticks.
`-joy-button` *btn=input*`:[`*args*`]`	Configure joystick button. `-joy-button help` to list physical joysticks.
`-joy-right` *name*	Map right joystick.
`-joy-left` *name*	Map left joystick.
`-joy-virtual` *name*	Specify the *virtual* joystick to cycle. Default: '`kjoy0`'

The axis and button mapping options used while configuring a profile need some explaining.

Configure axes with `-joy-axis` *axis=input*`:[`*args*`]`. The *axis* is either '`X`' or '`Y`' (or numbered 0–1).

Configure buttons with `-joy-button` *button=input*`:[`*args*`]`. The *button* is either 0 (first button), or 1 (second button—only useful on the CoCo 3).

In both cases, the *input* selects a source for the input from the list below, and the *args* specify which one to use.

Input	**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*

The '`-`' before the axis index when configuring a physical joystick will invert that axis. 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. Those screen offsets are relative to a 256x192 active area (the dimensions were chosen when that was the only output), but will be scaled to the currently displayed active area.

Joystick configuration is complex, but flexible. For example, you can combine input sources by specifying different modules for each axis. This configuration example creates a profile called '`mixed`' that uses the mouse for the X-axis and firebutton, but the keys `A` and `Z` on the keyboard for the Y-axis. It then ensures this profile is the one used when you press *CTRL+J*.

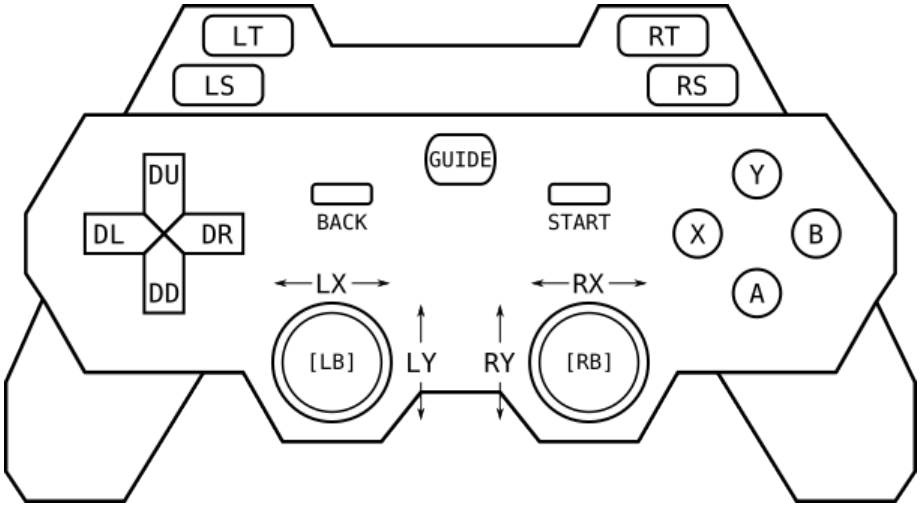
```
joy mixed
  joy-axis X=mouse:
  joy-axis Y=keyboard:a,z
  joy-button mouse:

joy-virtual mixed
```

6.9.1 Gamepad mapping

Although there is an approximation of consensus on what sort of controls go where on modern gamepads, they still tend to report their inputs in different orders. SDL supports user-provided mappings of reported axes, buttons and hat inputs to a set of named controls, and XRoar can process a file containing these mappings one per line, even when not built using SDL (e.g. when using evdev under Linux). Point XRoar to such a file with the `-joy-db-file` option. At time of writing, SDL_GameControllerDB (`https://github.com/mdqinc/SDL_GameControllerDB`) contains a comprehensive list of mappings.

This diagram shows the controls that can be mapped:



The short codes in this diagram correspond to these control names, used in SDL mapping database lines:

Input	Short	SDL	Input	Short	SDL
Axis 0	LX	leftx	Button 0	A	a
Axis 1	LY	lefty	Button 1	B	b
Axis 2	RX	rightx	Button 2	X	x
Axis 3	RY	righty	Button 3	Y	y
Axis 4	LT	lefttrigger	Button 4	BACK	back
Axis 5	RT	righttrigger	Button 5	GUIDE	guide
			Button 6	START	start
D-pad Left	DL	dpleft	Button 7	LB	leftstick
D-pad Right	DR	dpright	Button 8	RB	rightstick
D-pad Up	DU	dpup	Button 9	LS	leftshoulder
D-pad Down	DD	dpdown	Button 10	RS	rightshoulder

Using the evdev support under Linux, for gamepads, whatever is mapped to the D-pad acts as an alternative to axes 0 & 1. For joysticks, it provides an extra pair of axis controls.

6.10 Printers

`-lp-file` *`file`* Append printer output to *file*.
`-lp-pipe` *`command`* Pipe printer output to *command*.

6.11 Files

Many of these are mentioned in their appropriate section, but are collected here for reference.

`-load` *file*	Load or attach *file*. XRoar will try to do the right thing based on the file type (usually determined by file extension).
`-run` *file*	As `-load`, but try to autorun the file after attaching.
`-load-tape` *file*	Attach *file* as tape image for reading. See Section 5.1 [Cassettes], page 23.
`-tape-write` *file*	Open *file* for tape writing. See Section 5.1 [Cassettes], page 23.
`-load-text` *file*	Type the contents of *file* into BASIC on startup.
`-load-fd`*X* *file*	Load disk image file *file* into drive *X* (0–3). See Section 5.2 [Floppy disks], page 23.
`-load-hd`*X* *file*	Use *file* as the hard disk image for drive *X* (0 or 1). See Section 5.3 [Hard disks], page 24.
`-lp-file` *file*	Append printer output to *file*. See Section 4.6 [Printers], page 22.

6.12 Firmware ROM images

`-rompath` *path*	Set ROM search path. A colon-separated list of directories.
`-romlist` *name=list*	Define a ROM list.
`-romlist-print`	Print defined ROM lists and exit.
`-force-crc-match`	Force per-architecture CRC matching.

6.13 User interface

`-ui` *module*	Select user-interface module. `-ui help` to list compiled-in modules.
`-fs`	Start full-screen. Toggle full-screen with *CTRL+F* or `F11`.
`-fskip` *frames*	Specify frameskip. Default is '`0`'. May be helpful on slower machines.
`-vo-pixel-fmt` *format*	Pixel format to use. `-vo-pixel-fmt help` for a list.
`-gl-filter` *filter*	Filtering method to use when scaling the screen. One of '`linear`', '`nearest`' or '`auto`' (the default). OpenGL output modules only.
`-vo-vsync`	Start with vsync enabled.
`-vo-picture` *picture*	Initial picture area. `-vo-picture help` for a list.
`-no-vo-scale-60hz`	Disable vertical scaling for 60Hz video (enabled by default).
`-invert-text`	Start up with inverted text mode.
`-ccr` *renderer*	Composite video cross-colour renderer. One of '`none`', '`simple`', '`5bit`', '`partial`' or '`simulated`'. Default is '`5bit`'.
`-vo-brightness` *value*	Set initial brightness (0-100). Default is 50.
`-vo-contrast` *value*	Set initial contrast (0-100). Default is 50.
`-vo-colour` *value*	Set initial colour saturation (0-100). Default is 50.
`-vo-hue` *value*	Set initial hue (-179 to +180). Default is 0.
`-vo-monochrome`	Use a black-and-white television display (disabled by default).
`-vo-colour-killer`	Enable colour killer (disabled by default).

The pixel format, specified with `-vo-pixel-fmt`, defaults to RGBA with 8 bits per channel, but you may find other pixel layouts or lower bit depths render faster on your machine.

The default picture area is 640x480 (emulated) pixels, equivalent to `-vo-picture title`, which is enough to show normal VDG output with a reasonable border. You can change this to one of a set of defined areas: `-vo-picture action` and `-vo-picture underscan` show more of the picture, and may be more suitable when emulating a CoCo 3 which has some larger video

modes. `-vo-picture zoomed` crops to 512x384; enough to show standard VDG output with no borders at all. *CTRL+comma* and *CTRL+fullstop* zoom the picture area out and in where supported.

By default, 60Hz video is scaled vertically to give a 1:1.2 pixel aspect ratio. This more closely approximates the look of a real CRT (fewer scanlines mean each scanline appears taller within the same aspect ratio overall display). This can be disabled with `-no-vo-scale-60hz` to return to more square pixels. 50Hz displays tend to yield near-enough-square pixels anyway.

Various levels of composite video rendering precision can be selected with `-ccr`, trading off CPU with accuracy. `-ccr simulated` is the only option that tackles PAL video. `-ccr partial` does pretty well for NTSC. `-ccr 5bit` and `-ccr simple` both use LUTs to convert sequences of black & white into NTSC cross-colour.

When the VDG is configured to generate black & white (resolution) graphics, it stops emitting a colourburst signal. Colour displays may (but not always) recognise the lack of burst and stop trying to decode colour, giving a crisper display. You can enable this behaviour with `-vo-colour-killer`. NTSC machines add circuitry to reintroduce a (modified) burst to enable cross-colour in high resolution black & white, so enabling the colour killer would not prevent colour in these modes.

Use `-vo-monochrome` to model a black-and-white receiver for every TV input and video mode. RGB input is converted to luma using the active colour profile; composite input retains its transmitted waveform. Chroma is not decoded, and luminance is filtered more gently, revealing "dot crawl".

A quirk of the VDG is that it can operate in-phase or 180 out of phase with its clock signal, and how it starts up is essentially random. This clock signal is also used in NTSC machines to generate the colour subcarrier, which leads to machines generating the blue and red artefact colours randomly (but consistently, once running) swapped. 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. On the CoCo 3, a fourth mode is included that switches to the RGB output. In PAL machines, `"Blue-red"` and `"Red-blue"` also select the alternate line phase switch, allowing for correct colour in games such as *Tetris* by Ola Eldy or *Donut Dilemma* by Nick Marentes.

Inverted text mode may be toggled by pressing *CTRL+SHIFT+I*.

In the GTK+ and Windows interfaces, View → TV Controls opens a control window allowing you to dynamically modify various display options. Pressing *CTRL+SHIFT+V* will also open this window.

By default, XRoar does not synchronise rendering with the host machine's vertical sync, as this appears to cause audio glitches on some platforms. You can turn this on (where supported) with `-vo-vsync`.

6.14 Audio

`-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` *n*	Specify audio buffer size in frames.

`-ao-buffer-ms` *ms*	Specify total audio buffer size in milliseconds.
`-ao-buffer-frames` *n*	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. Default: '`-3.0`'
`-ao-volume` *volume*	Older way to specify volume. Simple linear scaling, using values 0–100.

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).

6.15 Debugging

`-gdb`	Enable GDB target.
`-gdb-ip` *address*	Address of interface for GDB target. Default: '`127.0.0.1`'
`-gdb-port` *port*	Port for GDB target to listen on. Default: '`65520`'
`-gdb-pseudo-regs`	Report an extended set of pseudo-registers to GDB, allowing inspection of SAM, PIA or GIME state.
`-gdb-flags-as-struct`	Describe CPU flags as a struct to GDB.
`-trace`	Start with trace mode on. *CTRL+V* toggles.
`-debug-fdc` *flags* `-debug-file` *flags* `-debug-gdb` *flags* `-debug-ui` *flags*	Various per-subsystem debugging flags. The special value '`-1`' enables all flags for the subsystem.
`-v` *level* `-verbose` *level*	General debug verbosity (0–3). Default: '`1`'
`-q` `-quiet`	Equivalent to `-verbose 0`.
`-trap` *condition*	Configure a trap to fire on *condition*.
`-trap-range` *m*`[-`*n*`]`	Perform trap's actions only on *m*th-*n*th trigger.
`-trap-snap` *file*	Write snapshot at trap.
`-trap-timeout` *n*	Quit emulator *n* seconds after trap.
`-trap-trace`	Start trace mode at trap.
`-trap-trace-n` *n*	Start trace mode at trap for *n* instructions.
`-trap-no-trace`	Stop trace mode at trap.
`-trap-ratelimit`	Enable rate limiting at trap (run normal speed).
`-trap-no-ratelimit`	Disable rate limiting at trap (run fast).
`-timeout` *n*	Exit emulator after running for *n* seconds.
`-timeout-motoroff` *n*	Exit emulator *n* seconds after cassette motor switches off, or end of tape reached.
`-snap-motoroff` *file*	Write a snapshot to *file* each time the cassette motor switches off, or end of tape reached.

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.
0x0010	General FDC event debugging.

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.

Traps are a somewhat flexible way of triggering emulator actions on particular events. Conditions can be:

Condition	**Description**
immediate (or '-')	Trigger immediately.
pc=*addr*	Trigger at CPU Program Counter value.
read=*a*[-*b*]	Trigger on read from address range.
write=*a*[-*b*]	Trigger on write to address range.
access=*a*[-*b*]	Trigger on either read from or write to address range.
tape-motor-off	Trigger when tape motor switches off.

A trap will usually trigger each time the condition is met. To constrain to a range of matches, specify `-trap-range` *`m`*`[-`*`n`*`]`.

Some older options are now implemented in terms of traps:

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

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. Equivalent to `-trap tape-motor-off`, `-trap-timeout` *`seconds`*.

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 (specifying a `.ram` snapshot may be particularly useful here for analysis). Equivalent to `-trap tape-motor-off`, `-trap-range 1-`, `-trap-snap` *`file`*.

6.16 Other options

Help options

`-config-print`	Print configuration to standard out.
`-config-print-all`	Print configuration to standard out, including defaults.
`-config-auto-save`	Write configuration to known location on exit.
`-h, --help`	Print help text and exit.
`-V, --version`	Print version information and exit.

In addition, various other options accept '`help`' as an argument to print a list of values they accept.

7 Files

If you "Load" a file, XRoar will determine its type using the filename extension and try to attach it in a way that makes sense for that file type. Load a file using File → Load, `-load` *`file`* on the command line, or by pressing *`CTRL+L`*.

If you "Run" a file, the file will be attached in the same way, but XRoar will also attempt to intelligently autorun any program. Run a file using File → Run, `-run` *`file`* on the command line, or by pressing *`CTRL+SHIFT+L`*.

Cassettes, Floppy disks, and Hard disks are each discussed in Chapter 5 [Storage media], page 23. The other kinds of file recognised by XRoar are discussed here.

7.1 Snapshots

XRoar can save a snapshot of the emulated machine state and read it 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.

Most internal state should be dumped to the snapshot. External data like ROM images or disk image files will be referenced by name, so when you read the snapshot back in, they need to exist in the same place they were before.

State that is explicitly *not* included in snapshots includes Becker port DriveWire connections and GDB listen parameters. These will use your local settings, which default to interacting with the local host only.

7.2 Screenshots

XRoar can save a screenshot in PNG format. Press *`CTRL+SHIFT+S`* or select File → Screenshot to PNG.

7.3 Binary files

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

Extension	**Description**
.bin, .dgn, .cco	Binary file (DragonDOS or CoCo). XRoar can load these directly into memory and optionally autorun them. Read-only
.s19	Motorola hex record. An ASCII format that encodes binary data and where in memory to load it. Read-only.
.hex	Intel hex record. Another similar ASCII format. Read-only.

7.4 Firmware ROM images

Firmware ROM image files are configured as part of a machine or a cartridge. They have a filename extension of `.rom`, 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 append `.rom`.
- 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, the primary examples being:

Firmware ROM	**ROM list**	**Canonical image names**
Dragon 32 BASIC	'`@dragon32`'	`d32.rom`
Dragon 64 32K BASIC	'`@dragon64`'	`d64_1.rom`
Dragon 64 64K BASIC	'`@dragon64_alt`'	`d64_2.rom`
Dragon 200-E 32K BASIC	'`@dragon200e`'	`d200e_1.rom`
Dragon 200-E 64K BASIC	'`@dragon200e_alt`'	`d200e_2.rom`
Dragon 200-E Charset	'`@dragon200e_charset`'	`d200e_26.rom`
Dragon Professional Boot	'`@dragonpro_boot`'	`alpha-boot-v1.0.rom`
Dragon Professional BASIC	'`@dragonpro_basic`'	`alpha-basic.rom`
Tandy Colour BASIC	'`@coco`'	`bas13.rom, bas12.rom, bas11.rom, bas10.rom`
Tandy Extended BASIC	'`@coco_ext`'	`extbas11.rom, extbas10.rom`
Tandy Super ECB (CoCo 3)	'`@coco3`'	`coco3.rom`
Tandy Super ECB (PAL CoCo 3)	'`@coco3p`'	`coco3p.rom`
Tandy Microcolour BASIC	'`@mc10`'	`mc10.rom`
Alice Microcolour BASIC	'`@alice`'	`alice.rom`
Tandy Advanced Colour BASIC	'`@deluxecoco`'	`deluxe.rom`
DragonDOS	'`@dragondos_compat`'	`dplus49b.rom, sdose6.rom, ddos10.rom`
Delta System	'`@delta`'	`delta2.rom, delta.rom`
RS-DOS	'`@rsdos`'	`disk11.rom, disk10.rom`
RS-DOS with Becker port	'`@rsdos_becker`'	`hdbdw3bck.rom`
Orchestra 90-CC	'`orch90.rom`'	

The default search path for images specified only as a base filename varies by platform, and is detailed in Chapter 2 [Getting started], page 4. This path can can be overridden with the option `-rompath` *`path`*, where *path* is a colon-separated list of directories to search.

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).

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.

7.4.1 BSDOS

Bootstrap DOS for the Dragon. Currently the only built-in ROM, for use with an IDE cartridge connected to a Dragon. It is configured in the '`ide-dragon`' cartridge profile, but may be referred to manually as '`bsdos`'.

When it starts, it will print the names of the primary and secondary IDE drives, if found. '`BOOT`' will boot off the primary, '`BOOT 1`' will boot off the secondary.

The boot track is expected to start at LSN 1, just after the MBR. This is the same offset in bytes as it would appear in a DragonDOS image. No other DragonDOS functionality is provided, so the boot track must contain all the code necessary to continue loading an OS.

BSDOS may be expanded in the future to support other devices, but for now it it really only provided as a way to start Fuzix from an IDE drive.

Appendix A Acknowledgements

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.

David Banks has published a lot of information on undocumented 6809 and 6309 behaviour learned as a result of hardware fuzzing.

Jim Brain contributed the iMMUnity code.

Alan Cox contributed the IDE code.

Greg Dionne and Ron Klein have been very helpful with information and testing of MC-10 related behaviour.

Erik Gavriluk contributed code to select a monochrome display.

Phill Harvey-Smith is the primary source of information about the Dragon Professional, and as well as his comments about it for MAME, has traced various other connections on his prototype board.

John Kowalski's GIME register reference was invaluable in getting early CoCo 3 support. The ability to see his composite video demos using XRoar's simulation code was also a big nudge towards even starting to add that support.

Tim Lindner has made many of his test cases public, helping with CoCo 3, 6309, and font accuracy.

Stewart Orchard has offered up much sage wisdom over the years. In particular, he figured out what was likely going on with SAM VDG address glitching.

Tormod Volden contributed support for his NX32 and MOOH devices (including general SPI and SD image support).

Various other people have also provided feedback or test cases that have helped nail down bugs; read the ChangeLog for details.

And thanks to all the people on the Dragon Archive Forums (`https://archive.worldofdragon.org/phpBB3/`), IRC and CoCo Discord that have provided helpful feedback and insight.

Various BBC R&D White Papers (`https://www.bbc.co.uk/rd/publications`) and *Video Demystified* by Keith Jack were good references while working on composite video simulation.

And finally, a nod to young me, who did some research into the illegal behaviours of TFR and EXG, and into how the SAM and VDG interact. Old me has spent more time with an oscilloscope, but also keeps introducing bugs into the code.

Appendix B Keyboard shortcuts

A summary of commonly available keyboard shortcuts.

CTRL+[1-4]	Insert disk into drive 1–4.
CTRL+SHIFT+[1-4]	Create new disk in drive 1–4.
CTRL+[5-8]	Toggle write enable on disk in drive 1–4.
CTRL+SHIFT+[5-8]	Toggle write back on disk in drive 1–4.
CTRL+A	Cycle through cross-colour modes (and RGB on CoCo 3).
CTRL+D	Open disk control tool (GTK+ & Windows only).
CTRL+SHIFT+D	Flush disk images.
CTRL+E	Toggle cartridge on/off - reset to take effect.
CTRL+F or F11	Toggle full screen mode.
CTRL+SHIFT+H or PAUSE	Halt the CPU (not on the MC-10).
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	Toggle Dragon/CoCo keyboard layout (not on the MC-10).
CTRL+L	Load a file.
CTRL+SHIFT+L	Load and attempt to autorun a file.
CTRL+M	Toggle menubar.
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+SHIFT+S	Write screenshot as PNG.
CTRL+T	Open the tape control tool (GTK+ & Windows only).
CTRL+V	Toggle trace mode.
CTRL+SHIFT+V	Open TV controls window (GTK+ & Windows only).
CTRL+W	Attach a virtual cassette file for writing.
CTRL+Z	Enable keyboard translation mode.
CTRL+-	Zoom out (smaller emulated display).
CTRL++	Zoom in (larger emulated display).
CTRL+<	Zoom out picture (see more border).
CTRL+>	Zoom in picture area (see less border).
F12	Run at maximum speed while held.
SHIFT+F12	Maximum speed toggle.

Appendix C File formats

XRoar recognises most file types by their file extension.

Extension	Description
`.cas, .c10`	Compact cassette image. CUE data can optionally mark up silence and the wavelength to use for each bit.
`.wav`	Standard audio data file can be used as a cassette image.
`.k7`	Another less popular compact cassette image format. Read-only.
`.bas, .asc`	ASCII BASIC files. XRoar will wrap the ASCII text in the appropriate file structure to present to the emulated machine as saved ASCII BASIC. On the MC-10, these will be "quick-typed" instead, as these machines do not support ASCII BASIC files on tape. Read-only.
`.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, .dsk`	Disk image file in a basic sector-by-sector format with optional header information.
`.os9`	Like `.dsk`, but XRoar knows it can inspect the OS-9 filesystem for geometry information.
`.vdk`	Another disk image file format, used by PC-Dragon.
`.sna`	XRoar-specific snapshots preserve machine state. Old v1 snapshots can still be read, but writing a snapshot uses the new v2 format.
`.ram`	When a `.ram` extension is given while writing a snapshot, a simple RAM dump is generated instead. Write-only.
`.bin, .dgn, .cco`	Binary file in DragonDOS or RS-DOS format (autodetected). Read-only.
`.s19`	Motorola hex record. An ASCII format that encodes binary data and where in memory to load it. Read-only.
`.hex`	Intel hex record. Another similar ASCII format. Read-only.
`.rom, .ccc`	ROM image file. Simple binary dump of a ROM IC. Machine firmware images and ROM cartridge images are in this format. Read-only.
`.ide`	HD image file assumed to be 512 bytes per sector with IDE "magic" and IDENTIFY DEVICE metadata in the first 1024 bytes.
`.img`	HD image file assumed to be 512 bytes per sector with no header.
`.vhd`	HD image file assumed to be 256 bytes per sector with no header.