XRoar 0.30.2

Dragon and Tandy Colour Computer emulator

This manual is for XRoar (version 0.30.2), a Dragon and Tandy Colour Computer emulator.

Table of Contents

Introduction

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

- Emulates Dragon 32, Dragon 64, Tandy CoCo 1 & 2, and compatibles.
- Emulates DragonDOS, Delta and RSDOS disk systems.
- Emulates the Orchestra 90-CC stereo sound cartridge.
- Supports both raw and translated keyboard modes.
- Reads and writes virtual cassettes (compact `.cas` files and audio files).
- Reads and writes VDK, JVC and DMK format virtual floppy diskettes.
- Saves and loads machine snapshots.
- Provides a GDB target for remote debugging.

XRoar was originally written to run on Solaris, Linux and the GP32 handheld. Later, it was ported to the Nintendo DS. Support has now been dropped for the GP32 and NDS as emulation accuracy, and thus the CPU requirement, has increased.

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

1 Getting started

1.1 Installation & running

Pre-built binary packages are available from the XRoar home page[1]. If one is not available for your architecture, you will need to build from source. XRoar should build and run on any POSIX-like system for which SDL is available.

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

1.1.1 Mac OS X binary package

Download the appropriate `.dmg` image for your system. Open the image and drag the application icon to `/Applications/`.

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

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

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

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

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

1.1.2 Windows binary package

Download and unzip the appropriate `.zip` distribution for your system. The easiest way forward is to simply run it from this directory.

Copy ROM images to the same directory, and if you create a configuration file, simply name if `xroar.conf` and put it in the same directory.

Note when troubleshooting that the Windows packages are built using MinGW[2] (Minimalist GNU for Windows), and instead of printing to the console, useful information will end up in files called `stdout.txt` or `stderr.txt`.

1.2 Building from source

1.2.1 Dependencies

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

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

[1] http://www.6809.org.uk/dragon/xroar.shtml
[2] http://www.mingw.org/

Under Mac OS X, first be sure to install Apple's Xcode[3] package. The easiest way to then ensure you have XRoar's dependencies available is to use a system like Homebrew[4] or MacPorts[5]. For Homebrew, the following command will install the required dependencies:

```
$ brew install libsndfile sdl
```

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

GTK+[6], the GIMP toolkit, provides the most full-featured user interface. It is only usable as such if you also have GtkGlExt[7], an OpenGL extension used to provide video output. Otherwise, it can provide a file requester for use by other user interfaces.

SDL[8], Simple Directmedia Layer, provides a more basic user experience, with most functionality only available through the keyboard shortcuts. Under Mac OS X, a special case of the SDL UI is built that also provides some menus. Unless you are building for Linux, SDL is required to use joysticks.

Other supported audio APIs: OSS, ALSA, PulseAudio, CoreAudio, JACK, Sun audio (not tested in quite a while).

libsndfile[9] is recommended to enable support for using audio files as cassette images.

1.2.2 Compilation

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

```
$ gzip -dc xroar-0.30.2.tar.gz | tar xvf -
$ cd xroar-0.30.2
$ ./configure
$ make
$ sudo make install
```

The `configure` script has a lot of options guiding what it tests for, specifying cross-compilation, changing the install path, etc. List them all with the `--help` option. Be aware that this is a custom script, it is not generated by the GNU build system.

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

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

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

---

[3] https://developer.apple.com/xcode/
[4] http://mxcl.github.com/homebrew/
[5] http://www.macports.org/
[6] http://www.gtk.org/
[7] http://projects.gnome.org/gtkglext/
[8] http://www.libsdl.org/
[9] http://www.mega-nerd.com/libsndfile/

1.3 Running programs

Images of ROM cartridges, cassettes and disks can all be attached by pressing *Control+L*. Cassettes will be positioned at the beginning of the "tape" and disks will be inserted into Drive 1.

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

- If a BASIC program is found on a cassette, XRoar types "CLOAD", waits for the program to load, then types "RUN".
- For machine code programs on cassette, XRoar types "CLOADM:EXEC".
- If a ROM cartridge is inserted, the emulated machine is power cycled.
- If a disk image is autorun, XRoar types "BOOT" (Dragon) or "DOS" (CoCo).

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

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

In addition, a couple of keys aren't always available on modern keyboads, so `Escape` maps to the Dragon's `BREAK` key, and `Grave` (`) maps to `CLEAR`. See Section 4.5 [Keyboard], page 14.

XRoar will make use of attached joysticks, but can also emulate them with the cursor keys and `Left Alt`. Press *Control+J* to cycle through three emulation modes: No joystick emulation (default), Right joystick, Left joystick. See Section 4.6 [Joysticks], page 15.

1.4 Configuration file

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

The installation and compilation sections above list good default locations for `xroar.conf`, but it is actually searched for in a list of directories. You can override this search path with the `XROAR_CONF_PATH` environment variable, which contains a '`:`'-separated list of directories. Here are the defaults:

Platform	**Default `XROAR_CONF_PATH`**
Unix/Linux	`~`/.xroar:*prefix*/etc:*prefix*/share/xroar
Mac OS X	`~`/Library/XRoar:`~`/.xroar:*prefix*/etc:*prefix*/share/xroar
Windows	:`~`/Local Settings/Application Data/XRoar:`~`/Application Data/XRoar

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

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

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

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

2 Hardware emulation

2.1 Machines

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

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

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

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

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

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

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

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

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

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

`-tv-type` *type*
: One of 'pal', 'ntsc' or 'pal-m'. PAL-M support is not complete, and will instead be treated the same as NTSC.

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

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

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

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

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

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

2.2 Cartridges

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

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

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

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

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

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

`-cart-desc` *text*
: Cartridge description (showed in `-cart help`).

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

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

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

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

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

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

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

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

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

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

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

2.2.1 Becker port

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

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

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

3 Files

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

3.1 Cassettes

XRoar supports three types of cassette image:

Extension	Description
`.cas`	Cassette file. Simple binary representation of data contained on a tape. Cannot represent silence, or some custom encodings.
`.wav`	Cassette audio file. XRoar can read sampled audio from original cassettes.
`.bas`, `.asc`	ASCII BASIC file. XRoar will convert text on the fly into blocks suitable for loading in ASCII mode. Read-only.

To create a cassette image for writing (with the `CSAVE` or `CSAVEM` BASIC commands, for example), use the `-tape-write` *`filename`* option, or press *`Control+W`*. Created files will be truncated to zero length, so be careful not to overwrite any existing files with this command.

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

Four options affect how tapes are read:

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

The `-tape-pad` option tries to make loading more reliable by intercepting ROM calls and inserting extra leader bytes where appropriate. Disable with `-no-tape-pad`.

The `-tape-pad-auto` option will, for `.cas` files, automatically switch on leader padding when insufficient initial leader bytes are found at the beginning of the file, otherwise it is left alone. Disable with `-no-tape-pad-auto`. On by default.

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

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

3.2 Disks

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

Extension	Description
`.dmk`	Disk image file in a format defined by David Keil. They store a lot of information about the structure of a disk and support both single and double density data. All disk images are manipulated internally in (near enough) this format.
`.jvc`, `.dsk`	Disk image file in a basic sector-by-sector format with optional header information.
`.vdk`	Another disk image file format, used by PC-Dragon.

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

When you attach a disk, it is read into memory, and subsequent disk operations are performed on this in-memory copy. Write enable defaults to on (so write operations on the copy will work), but write back defaults to off, so updates will not be written to the disk image file. To toggle

write enable, press *Control+[5-8]*, where the number to press is the drive number plus 4. To toggle write back, press *Control+Shift+[5-8]*. Even with write back enabled, image files will not be updated until the disk in a virtual drive is changed, or you quit the emulator.

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

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

The JVC format specifies that the default number of sides in a headerless image is 1, but some disk images are distributed without headers that are 40 track, double-sided. As these cannot be distinguished from 80 track single-sided disks, an option is provided to force the issue. If you're having such problems, run with the `-disk-jvc-hack` option.

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

3.3 ROM cartridges

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

3.4 Snapshots

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

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

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

3.5 Binary files

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

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

3.6 Firmware ROM images

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

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

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

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

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

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

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

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

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

4 User interface

4.1 User interface module

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

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

4.1.1 GTK+ user interface

Select with `-ui gtk2`.

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

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

4.1.2 SDL user interface

Select with `-ui sdl`.

A basic interface that provides no menus or control dialogues. All actions are performed through keyboard shortcuts. Provides three video output modules: '`sdlgl`' (OpenGL, preferred), '`sdlyuv`' (YUV, often accelerated under X11) and '`sdl`'. `-vo sdl` is the least capable, but most compatible; try this if you are having problems with OpenGL.

This is the only available user interface in the Windows binary distribution.

4.1.3 Mac OS X user interface

Select with `-ui macosx`.

This is actually a special case of the SDL user interface. A basic set of menus are created, and many operations are usable by pressing *`Command`*`+`*key* instead of *`Control`*`+`*key*.

This is the only available user interface in the Mac OS X binary distribution.

4.2 Keyboard shortcuts

A summary of commonly available shortcuts.

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

`Control+D`
: Open disk control dialogue (GTK+ only).

`Control+E`
: Toggle DOS emulation on/off - reset to take effect.

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

`Control+J`
: Cycle through joystick emulation modes (None, Right, Left).

`Control+Shift+J`
: Swap left and right joysticks.

`Control+K`
: Toggle between Dragon and CoCo keyboard layout.

Control+L
: Load a file (see below).

Control+Shift+L
: Load a file and attempt to autorun it where appropriate.

Control+M
: Cycle through emulated machine types (resets machine).

Control+Shift+P
: Flush printer output.

Control+Q
: Quit emulator.

Control+R
: Soft reset emulated machine.

Control+Shift+R
: Hard reset emulated machine.

Control+S
: Save a snapshot.

Control+T
: Open the tape control dialogue (GTK+ only).

Control+W
: Attach a virtual cassette file for writing.

Control+Z
: Enable keyboard translation mode.

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

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

Pause Toggles pause mode (HALTs the CPU). As seen on the Dynacom MX-1600.

XRoar still supports the use of some old keyboard commands that were used to attach specific types of file. *Control+B* and *Control+H* are synonymous with *Control+L*.

4.3 Video output

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

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

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

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

`-ccr` *renderer*
: Cross-colour renderer. Either '`simple`' (very fast) or '`5bit`' (fast, more accurate). Default is '`5bit`'.

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

4.4 Audio output

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

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

`-ao-rate` *hz*
: Specify sample 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-buffer-ms` *ms*
: Specify audio buffer size in milliseconds, where supported. Bigger buffers mean greater latency (the delay between what you see and what you hear), but may help alleviate glitchy audio on slow machines.

`-ao-buffer-samples` *samples*
: Specify audio buffer size in samples.

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

`-volume` *volume*
: Specify audio volume (0 - 100).

When the Orchestra 90-CC cartridge is attached, audio levels will be reduced due to the need to mix in a stereo output.

4.5 Keyboard

The default mapping of host keys to emulated keys is based on the original *positions* of the keys, with certain exceptions: cursor keys are mapped directly, `Escape` maps to the Dragon's `BREAK` key, and ` (grave or back-tick) maps to `CLEAR`.

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

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

`-keymap` *code*
: Specify host keyboard layout. One of '`uk`' (British), '`us`' (American), '`fr`' (French AZERTY), '`fr_CA`' (Canadian French QWERTY) or '`de`' (German QWERTZ). Default is '`uk`'.

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

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

The keyboards of the Dragon and Tandy CoCo operate in the same way, but the matrix is connected slightly differently. When you select a machine (see Section 2.1 [Machines], page 6), the appropriate matrix layout is selected for you, but you can toggle between the two configurations by pressing *Control+K*.

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

4.6 Joysticks

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

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

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

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

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

Interface	Axis args	Button args
'physical'	*joystick-index*,*[-]axis-index*	*joystick-index*,*button-index*
'keyboard'	*key-name0*,*key-name1*	*key-name*
'mouse'	*screen-offset0*,*screen-offset1*	*button-number*

For physical joysticks a '-' before the axis index inverts the axis. Key names for the keyboard interface depend on the underlying toolkit. The default screen offsets for the mouse interface are 'X=2,254' and 'Y=1.5,190.5' which gives a nice spread.

4.7 Printing

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

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

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

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

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

4.8 Debugging

XRoar can act as a remote target for GDB using a network socket. When GDB connects, emulation is stopped. GDB can then inspect memory, instruct the target to set breakpoints and watchpoints (read, write and access), single step or continue execution. A version of GDB patched to specifically support 6809 targets can also perform disassembly and inspect registers. For more information on how to use GDB, see the GDB Documentation[1].

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

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

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

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

0x0002 Hex dump of all data read into memory.

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

0x0001 Show FDC commands.

0x0002 Show all FDC states.

0x0004 Hex dump of read/write sector data.

0x0008 Hex dump of becker port conversation data.

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

0x0001 Connection open and close.

0x0002 Show packet data.

0x0004 Checksum reporting.

0x0008 Report on general queries.

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

[1] http://www.gnu.org/software/gdb/documentation/

5 Troubleshooting

Some commonly encountered issues:

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

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

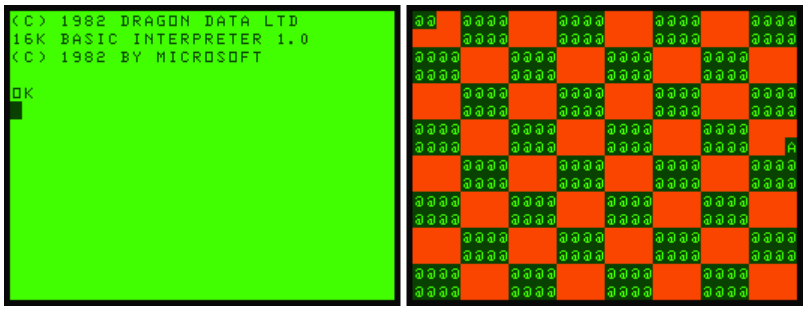

Figure 5.1: Emulator with and without BASIC ROM

- A black window appears. Commands typed blind appear to work.

This has cropped up a few times for people using Windows.

By default, XRoar tries to use OpenGL for its video output. Windows versions even as late as Windows 7 have been reported to come with drivers that do not even support basic texture rendering (OpenGL Extension Viewer reports "Max texture image units: 0"). If this is the case, try running with the `-vo sdl` option, which should select an unaccelerated, non-scalable output.

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

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

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

6 Acknowledgements

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

Thanks to all the people on the Dragon Archive Forums[1] for helpful feedback and insight.

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

[1] http://archive.worldofdragon.org/phpBB3/