uQLx

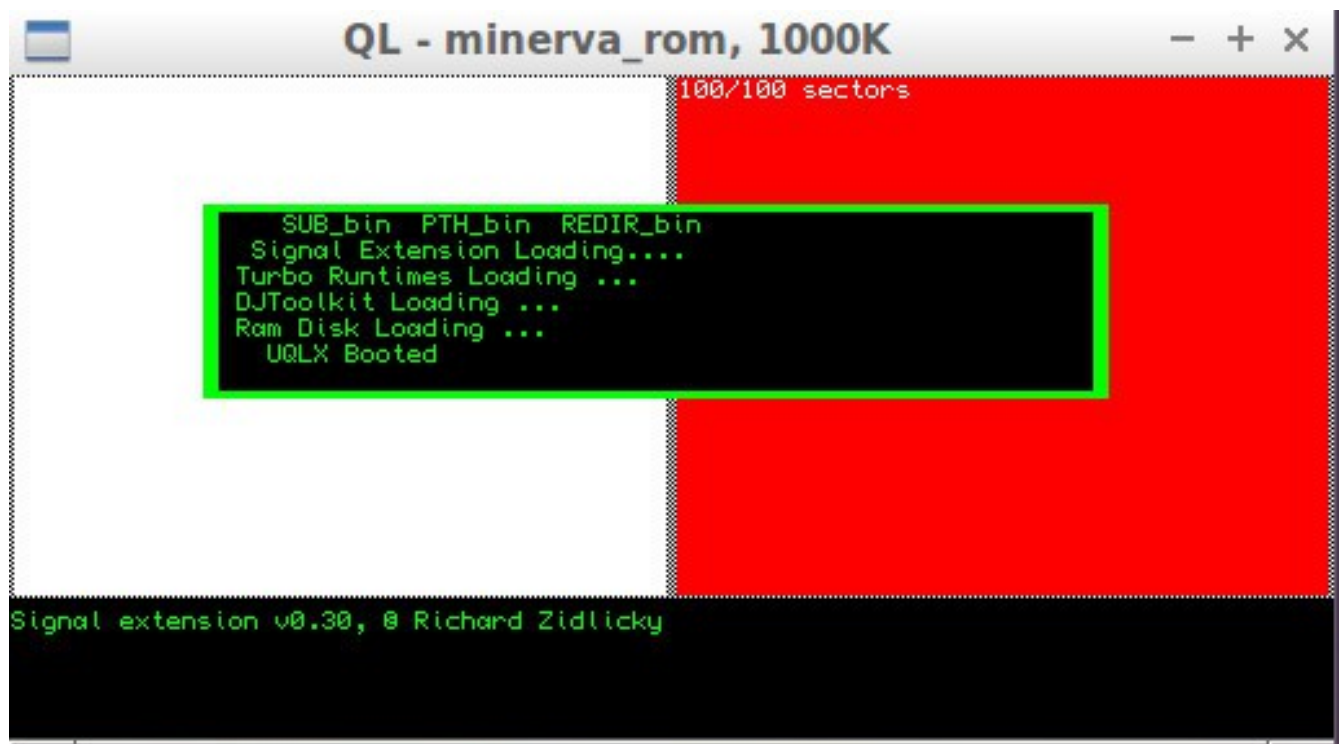## The Sinclair QL Emulator for Linux

Copyright

The JSROM image is copyrighted by others, distribution in North America with kind permission of Frank Davis and Paul Holmgren who own the copyright in this region. In Europe the JSROM copyright seems to be owned by Amstrad.

The Copyright of uQLx is contained in the file called "Copyright" that is part of the distribution.

Distribution

This is the original statement on distribution from the original version of uQLx:

Copying for personal use is not restricted, maximum commercial copying fee must not exceed DM 10 or 5 GBP. Posting or uploading the product on the Internet and other forms of online distribution is allowed. Inclusion on CD ROMS is subject to separate agreement.

Acknowledgments

This version of uQLx has been ported by Graeme Gregory who added the changes to get UQLX to compile for 64-bit. Graeme also created the x86-64-bit and ARM binaries. Timothy Swenson created the x86-23-bit binary. ARM testing by Robert Heaton.

This documentation is derived from the UQLX documentation by Richard Zidlicky with additional material by Timothy Swenson

Introduction

uQLx is an emulator for the Sinclair QL for a variety of Linux and Unix operating systems. It has been updated so that it will compile on recent versions of Linux and supports both 32- and 64-bit systems. uQLx is still a work in progress and may not run all QL software. Even when it was first released it still had issues. uQLx should work with most QL sofware and it should run SuperBasic with no issues.

This manual only details the installation and operation of uQLx and not of the Sinclair QL itself. To learn more about how to use the Sinclair QL, please refer to the following documents:

Sinclair QL User Guide - http://www.dilwyn.me.uk/docs/ebooks/olqlug/index.htm
        This was the manual that was provided with every Sinclair QL sold.

The Pointer Environment - http://www.dilwyn.me.uk/docs/ptr/index.html
        This is the add-on to the QL that allows for Windows and Mice.

The Minerva Manual - http://www.dilwyn.me.uk/docs/manuals/minerva_manual_11.pdf
        This is the other ROM that comes with uQLx.

The ToolKit II Manual - http://www.dilwyn.me.uk/docs/manuals/tk2.zip
        This is a very popular add-on that provides additional command and also comes with uQLx.

Getting Started with the Binary Distribution

There is a binary distribution of uQLx that allows the user to install and go on most 386-based Linux systems. No need to compile from the source code. This distribution comes with an example QXL.win file with a number of QL utilities and a default configuration.

Extracting the Files

Create a uqlx directory on your system. Copy the uqlx distribution (.zip) file into that directory and extract (unzip) the files. The main files are:

qm_x86-32	- Executable (for Intel x86 32-bit)
qm_x86-64	- Executable (for Intel x86 64-bit)
qm_armv6	- Executable (for ARMv6)
qm_armv7	- Executable (for ARMv7)
qm_armv8	- Executable (for ARMv8)
.uqlxrc	- uQLx configuration file
uqlxrc-distro	- uQLx configuration file for this distribution
romdir	- Directory for the ROMs
- js.rom	- JS ROM
- minevra.rom	- Minerva ROM
- tkii220.rom	- ToolKit 2 ROM

uqlx.win   - Bootable QXL.win file
uqlx.sh   - Shell script for use with Desktop shortcut
UQLX.pdf   - User Guide

Renaming the Right Binary

uQLx comes in a 32-bit (qm_x86-32), 64-bit (qm_x86-64) for Intel and 3 versions for ARM.
Determine if the Linux distribution you are using is 32- or 64-bit by running the "uname -i" command.
If the result is i686, then the Linux is 32-bit.

Copy either a version to qm:

```
% cp qm_x86-32 qm
```

Starting uQLx with the Distribution Configuration

The uqlx.sh shell script starts uQLx with the uqlxrc-distro configuration file, which uses the uqlx.qxl
file that comes with this distribution. You can start it up by running the shell script:

```
% ./uqlx.sh
```

Copy and edit the configuration file

Copy the .uqlxrc file into your own home directory. If you are the user "ted", then your home directory
is probably going to be "/home/ted". Edit the .uqlxrc file so that "<user>" is the user name that you are
using (like "ted"). This is the normal location of the .uqlxrc file. Using the -f option when starting
uQLx, the configuration file can be located anywhere and have any name.

Staring uQLx

From the command line, cd to the uqlx dirctory and type the command:

```
%  ./qm
```

Now run the BOOT program;

```
lrun win1_boot
```

Creating Desktop Shortcut

To make it easier to use, you can create a desktop shortcut that will allow you to start uQLx from the
desktop instead of the command line. The desktop shortcut can point the the uqlx.sh shell script file
that comes with the distribution. Edit .uqlx.sh to reflect the location of the qm binary.

The uqlx.win File

The binary distribution comes with a QXL.win file called uqlx.win. It has a boot file that loads a number of toolkits and sets up the configuration. uQLx will not automatically load the win1_boot file, so you must either do it yourself or add it to the command line option (see the file uqlx.sh).

The uqlx.win file contains a number of programs and utilities. The boot program loads the Pointer Environment, Toolkit II, some smaller utilities and a ram disk. It also contains a number of programs:

1 - Xchange
```
exec win1_xc_xchange
```

2. - unzip
```
exec win1_zp_unzip;"file"
```

3. - MineField
```
exec win1_gm_minefield_exe
```

4. - QED editor
```
exec win1_edt_qed
```

Note: If you are new to the QL, a QXL.win file is a single file that acts like a hard drive to the QL. It is a single file that contains a QL file system.

Quick Guide to Using uQLx

Accessing QL Floppies

uQLx is able to use USB floppy drives to read QL floppies. There might need to be some system changes to allow uQLx to access the USB drive.

The first step is to find out which device the USB floppy is installed on. With a Linux system with only one hard drive, then the floppy is probably going to be /dev/sdb. Use the lsblk command to confirm. Put a disk in the USB floppy and insert the USB connector into the system. Run the lsblk command and see which device shows up as size 720K. Edit the .uqlxrc configuration file to reflect the location of the floppy disk.

The next step is to allow the user access to the floppy. Run the 'su' command in a shell to become root, then edit the /etc/groups file. Add the user to the groups disks and floppy. Using the example user "ted" above, there is what the edit would look like:

```
disk:x:6:ted
floppy:x:25:ted
```

Exit from the shell, logout as the user and then log back in for the change in the /etc/group file to take effect.

Accessing Local File System

It is fairly easy to move files from the local file system to uQLx to be copied to a QXL.win file. The default .uqlxrc file sets up WIN2_ as a link to the ../uqlx/win directory. Any files put in the ../uqlx/win directory will be viewable when doing "wdir win2_". To copy a file from WIN2_ to WIN1_ just run the command:

```
wcopy "win2_file.txt" to win1_file_ext
```

Note that putting quotes around the first file name will allow uQLx to handle the ".txt", which is not normally allowed on the QL file system.

Creating a QXL.win file

uQLx does not have a way to create a QXL.win file. Jonathan Hudson has written qlxtool to create, read, and write QLX.win files. A binary version of qxltool has been added to the distribution to allow the user to create larger QXL.win files than the one that comes with the distribution. The way to create a new QXL.win file is to run the following:

```
%  qxltool -W newqxl.win 8 Label
```

Where newqxl.win is the same of the file to create, 8 is the size in MB that the QLX.win file should be, and "Label" is the QDOS label name for the QXL.win file.

Installing from Source Code

uQLx comes with source code that you can compile for your local environment. This may seem a daunting task, but it is fairly easy and should work most of the time, esp. if using a popular Linux distribution. The source code is the best way to get updates.

Prerequisites

To get and compile uQLx, there are two prerequisites:

GIT

GIT is an online repository for source code. uQLx is stored on GIT for others to access, either to compile and run or to contribute by editing the source code. The easiest way to access GIT is to load the client application. It is used to download source code. To install GIT on Ubuntu Linux, here is the command:

```
% sudo apt-get install git
```

This will download and install the client GIT application. Once that install finished, GIT is ready for use.

GCC

GCC is the Gnu C Compiler. It is used for compiling uQLx from the source code. To install GCC on Ubuntu Linux, there is the command:

```
% sudo apt-get install gcc-X.X
```

Where X.X is the version number. Check online for the latest version of gcc. The version that uQLx was tested with is 5.4.0.

Obtaining the Source Code

The uQLx source code can be downloaded to your system with the following command:

```
% git clone https://github.com/SinclairQL/uqlx-fork.git
```

This will create the directory uqlx-fork in the current directory.

Compiling the Source Code

To compile the code, do the following:

```
% cd uqlx-fork
% git checkout XorA/brutal-cleanup
% make
```

Once the compilation process has completed, there is a qm executable file that is uQLx. There might be some errors generated after the compilation of the QM executable. These can be safely ignored.

Updating the Source Code

If there have been any fixed or updates made to uQLx, the changes can be downloaded and the program can be re-compiled to get the latest version. Use the following commands:

```
% cd uqlx-fork
% git pull
% make
```

Configuration

The primary way to configure uQLx is a configuration file that is stored in the users home directory, .uqlxrc. The secondary way is the command line options.

.uqlxrc file

The .uqlxrc file is stored in the users home directory and is read by uQLx when it is started. With this file the user can define the devices that uQLx uses, determine memory, and various other settings.

The uqlx file uses a KEY = VALUE format. The '#' character can be used to start comments, rest of the line is not evaluated.

The keys available are:

SYSROM
> The name of the QDOS ROM to boot.

> SYSROM = js_rom

ROMDIR
> The directory where 'SYSROM' (and other ROMS) may be found.

> ROMDIR = /ql/ROMS/

RAMTOP
> The upper limit of memory; usual QDOS rules apply. The value is in kB. Be warned that large values for this will cause long startup delays unless FAST_START is enabled. I have largely tested UQLX with 4MB, but at least Minerva should handle 16MB. Currently UQLX won't allow more than 16MB, but this could be easily changed if you need more. If a larger screen size is used it has to fit into this value.

> RAMTOP = 4096

COLOUR
> The usage of a colour or mono display. Values are 0 for mono, 1 for colour. This may be used to simulate grayscale on a color monitor, not the opposite unfortunately. This option may get overridden to mono if a specific visual or visual class is requested as specified below.

> COLOUR = 1

XVID
> specify X visual ID to be used. Overrides XDEPTH and XVIS_CLASS. See `xdpyinfo` for list of available visuals.

> XVID = 0x24

XVIS_CLASS
> Specify preferred XVisualClass. This will affect whether color or mono is used and color cell allocation policy. Try experimenting if you have color palette flashing.

> XVIS_CLASS = StaticColor

XDEPTH
>Specify preferred display depth to be used. Should be 8 where possible

REAL_WHITE
>Set to 0 if you prefer greyish screens like me - useful with bad VUD's with insufficient refresh rates.. Redefines QL white to Gray95

SIZE_1
SIZE_2
SIZE_3
>Screen size definitions to be used for fast selection via program name argument('x','xx','xx'). See Program Name.

>>SIZE_1 = 640x400
>>SIZE_2 = 1024x768
>>SIZE_3 = 4096x4096

SER1
>The Unix device used for QDOS ser1.

>>SER1 = /dev/ttyS0

SER2
>The Unix device used for QDOS ser2.

>>SER1 = /dev/ttyS1

PRINT
>The Unix command used to queue print jobs, it used to output data sent to the PRT device. popen() is used to send the data, so you may specify options, flags etc.

>>PRINT = lpr -Pmy_printer

CPU_HOG
>Define it 0 to make UQLX try to behave multitasking friendly, it will go sleeping when it believes that QDOS is idle. The detection whether QDOS is idle usually works pretty well, but in some cases it may get fooled by very frequent IO, eg an high speed serial connection - in this case define it to 1 to get all time UNIX will give us. Alternatively the -h option can be used to enforce CPU_HOG mode. Largely obsolete now as it may be toggled through GUI.

>>CPU_HOG = 1

FAST_START
>Set to 1 if you want to skip the usual RAM test(default), or set it to 0 if you want to enjoy the Ram test pattern.

>>FAST_START=1

ROMIMG

The ROMIMG option defines additional ROMS to be loaded at specific addresses. These should include TK2 if required.

ROMIMG = tk2_rom,0xc000

It is assumed that the ROM image can be found in the ROMDIR directory. The address should be specified in 'C' numeric format.

XKEY_ON

A value of 1 can be used to indicate that qm should start with the alternative input method. This involves using ungrabbed keyboard (if configured) and preferring the X11 input method over QDOS translation of key events See section [Keyboard](). You might prefer this when you have a non-english keyboard and don't use many special QL key combinations. The downside is that typical QL hotkeys are very often interpreted by window managers - these won't be available for QDOS programs and may additionally screw up your desktop or even kill applications. Default is 0.

XKEY_SWITCH

Defines Keysym to be used to switch keyboard input method See section [Keyboard](). The Keysym name should be in the form returned by 'xev', ie without the leading 'XK_'. It should be accessible without modifiers. Default is to use the "F11" key.

XKEY_SWITCH = F16

DO_GRAB

Whether to do keyboard grabbing. This is used to avoid confusion when window manager would try to interpret QDOS key like ALT-F1. Proper fix is to disable it and get a ICCM compliant wm (eg. windowmaker) Enabling it will interfere with the broken Xkb extension See section [Keyboard]()

DO_GRAB = 0

XKEY_ALT

Defines Keysym to be used as (additional) alternative to the Alt keys to simulate QDOS ALT. Reason for this is that many window managers catch away the Alt keys to use them as their hotkeys. Should be accessible without modifiers. Default "F12"

XKEY_ALT = Mode_switch ## frequently this is Alt_R

STRICT_LOCK

Controls whether strict locking applies for disk image files, the alternative being advisory locking. True by default, disable if you hate the ugly warnings. BTW never rely on locking in UNIX anyway.

STRICT_LOCK=1

DEVICE

All directory devices may be defined in the options file. The format is

DEVICE = QDOS_name, UNIX_pathname[, flags]

QDOS_name is the name of the QDOS volume to be defined, eg FLP1, WIN6, QXL1. Currently RAMx is the only name that receives special attention. UQLX does not enforce any further naming conventions, however most QDOS software requires a 3 chars name length. UNIX_pathname refers to a file, directory or device used to simulate the QDOS device. The optional flags field supports this options.

clean

clean together with a "%x" in the unix pathname can be used to simulate RAMdisk. The "%x" is replaced with the process number at runtime so that multitasking QMs don't disturb each other and after killing QM the directory is deleted.

qdos-fs

native

Both flags are synonyms. The qdos-fs option indicates that UNIX_pathname is the name of a file or device in the QDOS floppy disk or QXL.WIN formats; otherwise a Unix directory is assumed.

qdos-like

applicable only to non-qdos-fs. Filenames are not case sensitive and (sub)directory creation mimics SMSQ behaviour.

Devices may be removed from the device list by not supplying a unit (volume) number. This is useful if some devices that are defined by default, eg 'mdv','flp' are unused.

DEVICE = CD

Would remove the above default CDROM specification. Some device mapping and other options are supplied as default; in addition, the following defaults are also set.

SYSROM = jsrom
ROMDIR = /ql/
RAMTOP = 4096
COLOUR = 1

PRINT = /usr/bin/lpr
CPU_HOG = 1

Note that no additional ROM (tk2) is defined by default.

and here is the example of an actual .uqlxrc file. You will find more recent versions of it with every UQLX distribution.

```
SYSROM = js_rom                  # default ROM to use
ROMIM = tk2_rom,0xc000            # extra ROM
ROMDIR = ~/qm/romdir/            # ...search them here
RAMTOP = 16384
DEVICE = MDV1,~/qm/qldata/           # this directory will be accessible as 'mdv1_'
```

```
DEVICE = MDV2,~/qm/qlsoft/
DEVICE = FLP1,~/qm/DiskImage2,qdos-fs   # 'flp1_' is the image of a real QL floppy..
DEVICE = FLP2,~/qm/DiskImage,qdos-fs
DEVICE = FLP3,~/qm/DiskImage3,qdos-fs
DEVICE = WIN1,~/
DEVICE = WIN2,/
DEVICE = WIN3,~/PiQ/
DEVICE = RAM1,/tmp/.ram1-%x/,clean      # temporay dirs, cleared after exit
DEVICE = RAM2,/tmp/.ram2-%x/,clean
DEVICE = RAM3,/tmp/.ram3-%x/,clean
DEVICE = RAM4,/tmp/.ram4-%x/,clean
DEVICE = RAM5,/tmp/.ram5-%x/,clean
DEVICE = RAM6,/tmp/.ram6-%x/,clean
DEVICE = RAM7,/tmp/.ram7-%x/,clean
DEVICE = RAM8,/tmp/.ram8-%x/,clean
DEVICE = CD1                    # devices we don't want
DEVICE = MS1
COLOUR = 0                      # simulate MONO, 1=COLOR monitor


PRINT = lpr            # printer spooler prog used by PRT port
CPU_HOG = 0               # don't burn CPU power in QDOS scheduler loop
FAST_START = 1            # skip ramtest
```

Command Line

uQLx supports the following command line options; these override settings in `~/.uqlxrc.' Note that options in turn can be overridden by program name as described above.

```
qm [-r RAMTOP] [-i] [[-c][-m]] [-f file] [-h] [-o romname] [-z x]
  [-s [string]] [-b [string]]
```

where:

-r RAMTOP

Defines the RAMTOP value in kB. Any enlarged screen also has to fit into this value.

-c

Forces colour mode.

-m

Forces mono mode (even on a colour X display).

-g nXm

Start with screen size nXm, effective only with Minerva roms. See big screen

-f file

Defines an alternative options file.

-h

Force CPU_HOG mode, take all available CPU time for the emulator.

-o romname

Use romname instead of ROM defined in `.uqlxrc' file

-s boot_cmd

No attempt is made to make a connection to the Xserver, See section Scripting. boot_cmd must be present, it defines a QDOS 'BOOT' device to be used, see -b option.

-b boot_cmd

Define QDOS 'BOOT' device that will return the boot_cmd string on read. The boot_cmd should be a string of the form "10 lrun mdv1_progxx" or similar; quoting newlines is tricky and therefore only 1 line expressions are recommended. A QDOS newline char is automatically appended to the string.

-i

Start with uQLx window iconised - if supported by window manage

-z X

Start uQLx window to be X times as large as normal. This increases the size of each pixel so the QL screen appears larger, including larger letters. The default behavior is "-z 2". Use the option "-z 1" to run with the smaller screen.