QL-SD

Driver Source Code

This document is intended to give a bit of guidance to the QL-SD driver source code.

This packacke is not just the driver for the QL-SD hardware, but instead a generic driver framework for QUBIDE and BDI file systems on nearly any arbitrary hardware that is capable to do LBA (Logical Block Adressing).

The original author, Adrian Ives, has invested a lot of work to pull apart the QUBIDE driver software into a generic piece of filesystem code (the EDDE2 driver core) and the actual physical hardware support code. This makes it easy to adapt the driver to new hardware while re-using the complete file system code. The „only“ thing that needs to be done to support new hardware, is supplying not much more than sector read and write functionality on LBA level.

The driver uses its own slave block mechanism to work around some of the QDOS deficiencies in slave block handling, especially if there is (too (!)) much free memory available.

Note this document is not intended as an introduction to driver programming on QDOS - It is assumed you have a fair bit of knowledge of assembler programming as well as quite some knowledge about driver programming in QDOS in general.

1. What's in the Package?

The sources unpack into three directories:

- *lib* - This contains support routines used by both the physical and the logical driver layer
- *edde2* - This contains the logical driver source - The actual file system is implemented here. The edde2 driver core expects an assortment of *logical blocks* that it creates its QUBIDE file system on
- *sdcard* - This contains the routines that supply the logical blocks to the edde2 core - In the case of QL-SD, these can either be supplied directly from the SD card (the „native“ mode) or from a BDI image file on a VFAT32 file system (the recommended „BDI“ mode).

2. Ingredients

You will need the following to successfully build the QL-SD drivers (You could be using different software, emulators or even build on the “real” QL, but I recommend setting the build tree up on QPC2, as this will allow you to use a comfortable Windows/MacOS/whatever editor and <ALT><TAB> between your build and your edit environment[1]):

1. QPC2 - Any version will do that supports the dos*x*_ device
2. A working installation of C68 in that QPC2, basically needed for *make* and *rm*
3. QMAC/QLINK, the Quanta Assembler and Linker

3. How to build the driver

The environment in the supplied zip file is intended to be unpacked to dos2_ on QPC2 - So please make sure that drive is actually mapped to somewhere reasonable on your QPC2 installation (we will not be going into any depth how to map Windows drives to SMSQ/E drives here)

---

[1] I personally use *BBEdit* on the Mac for editing the source code - I have built a number of syntax highlighting rules for QDOS assembler to make my life a little bit more comfortable. I also use SVN to store versionized copies of the developed code on a server - I have really long periods of not doing anything for the QL, so a versioning system comes in very handy if you come back after a few weeks and don't know what it was that you have done last....

Once unpacked to the QDOS drive dos2_, you should find the 3 directories mentioned above. Each of them has its own makefile that builds all the object files in the directory. You compile the driver by

```
DATA_USE dos2_
DDOWN lib
EX make ;
DUP
DDOWN edde2
EX make ;
DUP
DDOWN sdcard
EX make ;
```

(This assumes your PROG_USE directories are set so that make, rm, MAC, LINK and other programs needed by the makefile are within reach of S*BASIC. I personally use the pth_ device to have the computer find the programs for me....). Depending on your installation, you might have to adjust some path names in the makefile and the ver_in configuration file.

Make sure you run make in the directory order above - Some of the code relies on other code already built. Your compilation might fail if you change the order.

The above sequence of commands will produce a *bin* file in the *sdcard* directory that can either be burnt into an EPROM or be LRESPRed from RAM depending on configuration.

4. How to control what to build

To control the actual target that is being built, you use the file *ver_in* in the *sdcard* directory.

The most important switch in the *ver_in* file is `IS_QLROMEXT` - The driver sources contain switches to support quite some hardware that you most probably won't have - So leave `IS_QLROMEXT` at 1 and the other switches at 0.

The other switch you might want to change is IS_ROM. This (obviously) switches between a driver version to run from EPROM versus one that can be LRESPRed and run from RAM (it basically adds a QL ROM header in front of the code). The other switches control various defaults or debug builds that you might want to leave alone for the moment.

The zip file contains a number of preconfigured *ver_in* files for various targets that you can simply copy to *ver_in.* The *ver_in* file that will produce the code for your QL-SD ROM is called *dos2_sdcard_ver_qrx082_ROM_in.*

Note the compiled binary contains a QDOS CONFIG block that can be used to set a number of defaults for the driver (obviously before it is actually burnt to ROM...)

The makefiles have one additional target except the default one which is *clean* - This simply removes any intermediate files after compilation in order to keep your build tree tidy.

August 2014, T. Fröschle