QL-SD v2

User Manual

1. Contents

IMPORTANT NOTE

By the time of writing this document, the following incompatibilities between QL-SD and standard QL hardware were known:

1. QL-SD is principally incompatible with any software using the ROM slot area (The Eidersoft ICE and mICE suite of programs and some Metacomco compilers are known examples).

2. QL-SD is also incompatible with any hardware already using the ROM port, like the newer TrumpCard clones that include a QUBIDE interface

3. Unexpanded QLs with only the original 128k memory have severe limitations on possible filesystem sizes due to shortage of memory for the driver. It is thoroughly recommended to use QL-SD with a memory extension.

2. Introduction

This is the User Manual for the QL-SD device for Sinclair QLs and other computers running various QL-alike operating systems. QL-SD is intended for retro-computing and fun purposes, not to process important data.

It has been developed with the aim in mind to give the QL a state-of-the-art mass storage device given that Microdrives now nearing their 30th year tend to break a lot and replacements are very hard to find. QL-SD has been put together for you by a number of QL enthusiasts, some of them are listed in the acknowledgements section.

2.1. About This Manual

The QL users' scene is manifold - It ranges from the relatively novice inexperienced retro-interested user who just obtained a QL from a flea market or a garage sale to the long-time experienced QL-Expert who writes an S*Basic extension during lunch break. This manual tries the impossible - it is written for both types of users (and all the rest that are located somewhere in-between). The first (about half of it) part covers all the basics you need to know to operate this new type of mass-storage device with your QL

The second part of the manual handles the more obscure - and potentially dangerous (to your data, and your peace of mind, probably) aspects of the hardware and software.

Should you be willing to explore the commands and procedures described in the latter part of the manual, it is assumed that you know what you are doing and are not trying to risk important data but instead work on backups and test cards. For the more novice users: There is really nothing in there you could probably need or really benefit from - Most of it is diagnostic stuff or low-level commands. You won't be missing any of this.

If you notoriously tend to ignore manuals:

In case you belong to the 99% of people who want to see their newly acquired gadget working in no time and thus typically turn a blind eye to most sections of your precious manual:

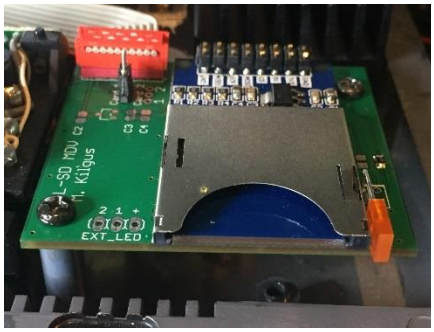
Please make sure you at least read the chapter on hardware installation, followed by the quick guide for preparing an SDHC card. Once you have seen QL-SD working, please do come back and read the rest - it is worth it!

3. QL-SD Package Contents

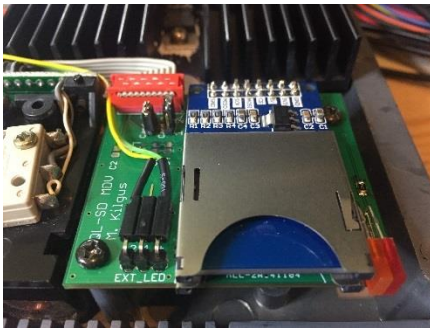
QL-SD consists of the following components:

1) A small PCB board that makes up the actual QL-SD. This is intended to go into the ROM socket of one of the QL system ROMs (which one does not matter). It contains the main QL-SD circuitry and has a socket on top to either hold one of the original ROMs or a new 64 KB (E)EPROM with the operating system plus the QL-SD driver software.

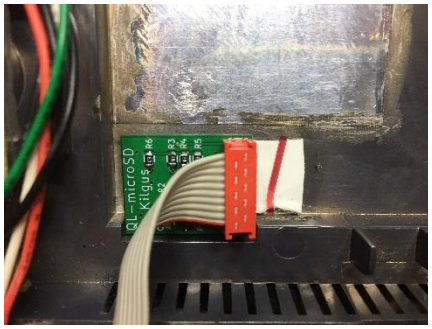
2) A second PCB with one or more SD card sockets. For QL-SD v2 there are currently three variants, the single-board variant being like the boards supplied with older QL-SDs:



A board with a single SD slot installed in place of a Microdrive



A board with two SD slots installed in place of a Microdrive



A small microSD board that is installed in one of the ventilation openings in the QL case

3) A short piece of ribbon cable connecting the two PCB boards.

4) A 64KB EEPROM with driver software and replacement operating system.

4. Hardware Installation

As QL-SD is intended to go inside the QL's case leaving the original stylish outside intact as much as possible, you will need to open up your QL case and replace some chips. No soldering is required, just a bit of caution and common sense. If you have never handled electronic devices before, please be assured installation is actually quite easy - just take your time and don't force it.

Important note regarding keyboard membranes

In order to install QL-SD, you will need to open up the QL case and handle its inner parts.

QL keyboard membranes usually become brittle with age and tend to break easily when moved (and you need to do that when installing QL-SD), resulting in a non-working keyboard.

Sometimes, it is possible to 'repair' a broken membrane by just shortening the ends that go into the QL main board connectors a bit, but that is not recommended and will not last long anyway.

If your QL still has its original membrane, probably now is the time to look for a replacement.

RWAP services in the U.K are selling high-quality replacement membranes.

4.1. Opening the QL Case

Before opening the case, make sure you remove all Microdrive cartridges, expansion and ROM boards from your QL. Also make sure power is disconnected.

The QL case can be opened by removing the screws holding the upper and lower parts of the body together. The screws are located on the underside of the QL, 4 towards the front side (on a recessed ledge) and 4 towards the back side, equally spaced along the length of the QL (Note one of the rear screws might hide underneath a sticker). On the right side of the QL there are two additional screws directly underneath the Microdrives in about the middle of the bottom case – Leave those alone when opening the case, they are holding other QL parts on the inside.

Once you have removed the screws, flip the QL around (making sure that it doesn't fall apart yet) and carefully lift the keyboard and top cover. There are still some cables for the power LED and the keyboard connection that hold the upper and lower part together. Carefully unplug those from the QL main board and remove the upper case. QLs that weren't made in Korea by Samsung (that is,

basically all U.K. and pre-Issue 6-Models) are said to have some tricky connectors for these cables – be careful, because if you bend the wires they seem to be very hard to slide back in.

4.2. ROM / EPROM Considerations

You should now decide whether you want to use QL-SD with its own dedicated ROM that comes with Minerva 1.98 and the QL-SD driver. This has the advantage over any other solution that you will not need to load the driver from disk or Microdrive before you can start using QL-SD. Minerva on the QL-SD's EPROM is probably the most advanced and bug-free versions of all QL ROMs. TK2 is a must for every advanced QL user, so should be the first extension loaded from the SDHC card, if not included from ROM.[1]

The disadvantages are small: There are still a small number of pieces of software around that are said not to be compatible with Minerva, and you will need to load any non-English keyboard layout from disk (or rather, QL-SD, though a special German build of Minerva exists and is maintained by Marcel Kilgus). It is thus thoroughly recommended to use the QL-SD EPROM unless you have very good reason to do otherwise.

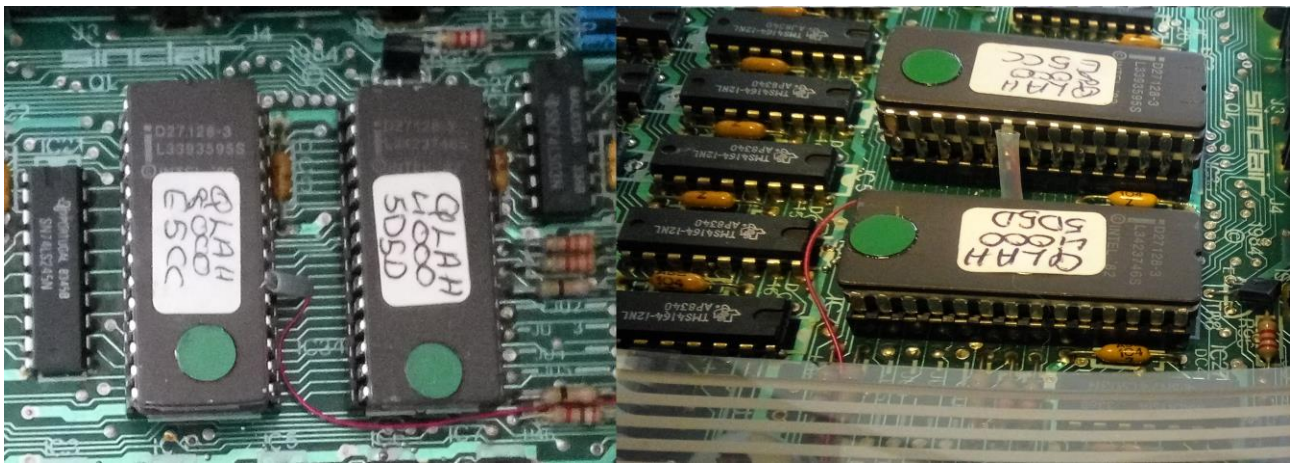
Unexpanded QLs will only work with the driver in ROM due to memory restrictions.

Should you go with the QL-SD EPROM, you need to remove *both* original QL ROMs from their sockets, if you want to stay with your original ROMs, remove only one of them, once QL-SD is installed, it is going to be re-fitted on top of QL-SD. If you want to go with QL-SDs EPROM, remove both original ROMs (IC33, IC34). Actually, it does not matter which of the ROM sockets is used – QL-SD works in both – we recommend the right one.

Please note that the QL-SD driver and hardware registers operate in the memory area originally intended for the ROM slot ($C000 - $FFFF). With QL-SD installed, the QL ROM slot may be no longer usable for other purposes.

4.3. AH conversion

If you appear to have a piggy-backed set of ROMs with flying leads you have an EPROM'd QL which will need converting with the following set of instructions before QL-SD will work correctly.



[1] If you have a disk version of TK2, load TKII like that:

```
base=RESPR(16384):LBYTES WIN1_TK2_EXT,base:CALL base
```

If that does not work (i.e, crashes your QL), it is because you might have a ROM binary which has a header that needs to be skipped. Try this:

```
base=RESPR(16384):LBYTES WIN1_TK2_EXT,base:CALL base+PEEK_W(base+6)
```

WARNING – we cannot be held responsible for any damage you might do to your QL in the AH conversion. If you are doubtful then seek expert advice.

1. Remove and discard any flying wires from A14, IC34 and JU points.

2. Remove all wire links in JU points 1 to 6 (to the right of IC34). Note that JU 1 and JU6 may appear to be fitted with resistors with only a single black band. These are in fact zero resistance and were inserted because the automatic component insertion equipment used to populate the boards cannot handle bare wire links.

3. Remove IC17 (74LS00) - on some PCBs this chip is socketed and on others it is soldered into the PCB.

4. Remove IC33 and 34 (EPROMS)

5. Fit wire links to JU2, JU3 and JU4

4.4. Installing the QL-SD Main PCB

As QL-SD goes where the QL ROMs are normally located, these need to be removed first. Either use an IC removal tool in case you have one (usually supplied now along with QL-SD), or use a small screwdriver, carefully sliding between socket and ROM repeatedly from either side, prying the ROM from its socked slowly but steadily. Make sure you do not bend the legs, you will need the ROM intact afterwards should you decide not to use the QL-SD EPROM.

If you intend to use the QL-SD with its own dedicated ROM you need to remove **both** original QL ROMs (IC33, IC34), in case you want to retain your original ROM version, only remove one of them.

Once the ROM is removed, the QL-SD goes into one of the now empty ROM sockets of the QL. Make sure you align the pins properly and carefully insert the QL-SD into the ROM socket (the connector to the daughterboard facing to the front of the QL), pressing firmly down once you have made sure everything goes where it should.

If you have removed only one ROM, that one goes now into the socket on top of the QL-SD. Make sure the small IC notch faces to the back of the QL. Put the ROM into the inner rows of connectors on the QL-SD and push it back in, cautiously watching you do not bend the pins and all of them are properly inserted in the socket.

Inserting the QL-SD EPROM is a bit more tricky, but is already done for you by your supplier: As the EPROM has much more capacity than the original ROMs, it needs to be inserted a bit differently: Some pins of the EPROM need to be bent outward a bit and do not go into the inner row of connectors, but instead into the outer row on the QL-SD socket. If your QL-SD did not come with the EPROM already inserted, use a pair of fine pliers to slightly bend the respective pins outward, with a short break inwards at the bottom, so that the pin can be inserted vertically in the outer row of socket connectors.

Make sure the EPROM is inserted with its notch facing to the back of the QL and all pins are settled in their respective connectors properly.

4.5. Installing the QL-SD SDHC Daughter Board (Single/Dual)

The second board holds a socket into which the SDHC card is inserted. This board can be mounted instead of a Microdrive unit, allowing to insert and remove the card just like a Microdrive cartridge.

For this, you need to remove one Microdrive, we recommend mdv2_, the rightmost one.

4.5.1. Microdrive Removal

In order to remove the Microdrive (which one doesn't really matter – We recommend to install QL-SD into the right MD slot, but whichever Microdrive you remove, the remaining one will from then on be mdv1_).

It makes life easier if you first remove the large heat sink on the right of the QL, just behind the Microdrives. It is just a single screw in the centre of the heat sink and some washers and, once removed, makes access to the Microdrive connectors on the QL main board much easier. Temporarily remove that heat sink, unplug the Microdrive cable from the main board, then re-install the heat sink.

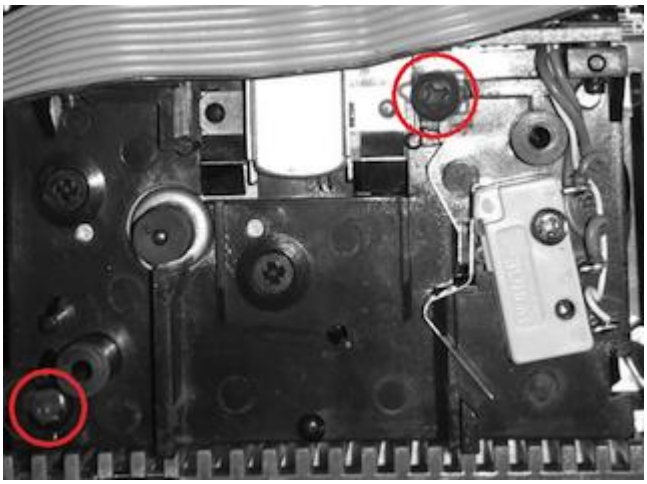
The picture marks the Microdrive screws that need to be removed. Note that there might be a third screw holding the drive on the underside of the QL. Leave the other screws in place! You can now lift the Microdrive from the case and store it in a safe place. You hopefully won't ever need it again...

4.5.2. Daughterboard Installation

The single/dual QL-SD daughterboard goes into the same place as the Microdrive and is mounted with the same screws. Once installed, check with the top case on whether the daughterboard fits correctly. You want the top side of the daughterboard exactly flush or a little bit above with the casing slot. Depending on the PCB material used, the daughterboard might sit slightly deeper than the Microdrive slot in the casing. Should that be the case, you should shim the daughterboard with thin (plastic!) washers in order to lift it a small amount. If the daughterboard sits too deep in the case you might have problems inserting and removing SDHC cards later or you might even damage the SDHC card slot.

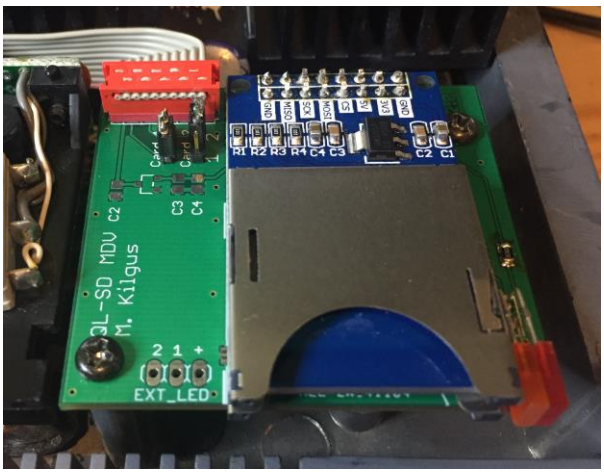
Now you should connect the two boards with the supplied ribbon cable. When you put the cable in place, simply make sure it doesn't interfere with Microdrive #1 it would be passing by in the recommended set-up. The cable should pass mdv1_ on its rear and use the same path mdv1's cable takes.

Remark: It is not mandatory to fit the QL-SD daughterboard into a Microdrive slot of the QL – We just think it is the most convenient solution, given the declining usability of Microdrives these days. If you want to retain the second Microdrive at all cost – Feel free to find another space in the QL casing that can take the daughterboard without interfering with other components or use the microSD daughter board instead. Some space near the ROM slot might as well suffice, simply make sure the daughterboard is fixed firmly in place and does not have any electrical contact with any other QL component.



This is how the installed QL-SD should look like[2]:

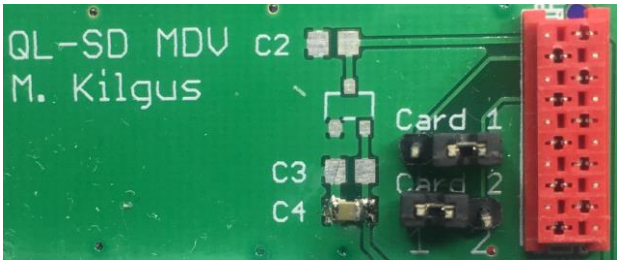
[2] For the hardware cracks that are not afraid of a soldering iron a dedicated Pin header to connect an external LED is provided on the boards. This can be used to connect to the original Microdrive LED, for example. But note that due to brightness differences of the LED colours the series resistors for the two LEDs differ, so the two pins drive different currents.



4.5.3. Card selection

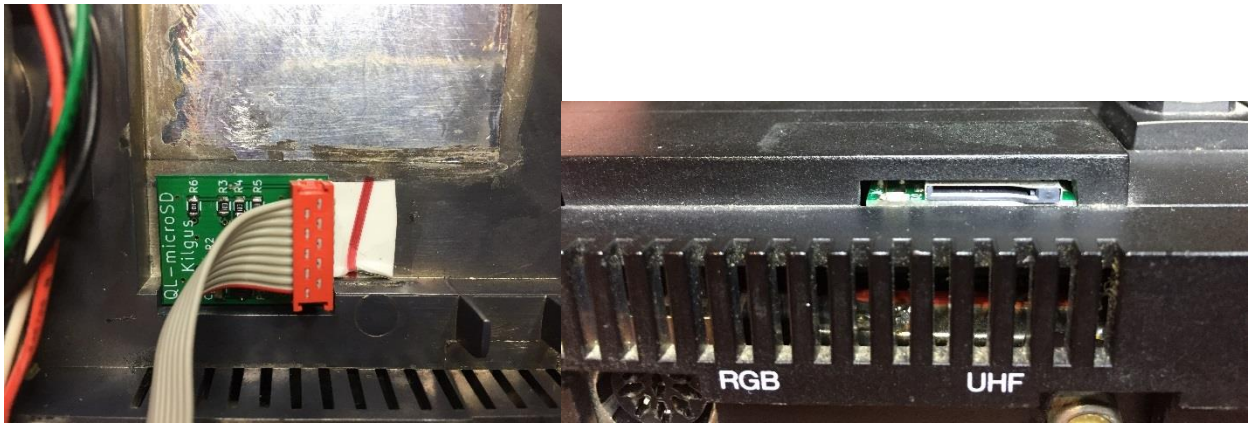
The new dual-slot QL-SD v2 daughter board features jumpers to select if the upper slot is card 1 and the lower slot card 2 or vice versa. As the lower slot is difficult to change while there is a card in the upper slot it is recommended that it acts as the "hard drive" WIN1 and the upper slot as the removable drive WIN2.

"Card1" is the jumper for the blue daughter board, "Card2" is for the slot directly soldered to the green PCB (in retrospect, the labelling was not the best). If the jumper is to the left (over the "1" below) then the slot will act as the first card (usually WIN1), otherwise as the second (usually WIN2). In the example below the lower slot is configured to be the first card and the upper slot is the second card, which is the recommended configuration.



4.6. Installing the QL-SD SDHC Daughter Board (microSD)

Installing the microSD board is straight forward. Remove the protective sheet from one side of the supplied PowerStrip® and press it onto the microSD slot so that the remainder of the strip is to the side of the red connector. Then remove the other protective sheet and press the board into the upper casing of the QL where the ventilation slot is located:



4.7. Reassembly

Bad manuals would now state „Assembly follows the exact steps as disassembly but in the opposite order“- And we do exactly the same.

Re-Insert the top case LED connections, the keyboard connector strips, and place the top cover on the bottom cover. Make sure you didn't forget any screws or washers inside the case - They might do severe damage when you switch on the QL. Also, thoroughly check that you have fitted all circuits the correct way around before applying power to the QL – QL ICs (at least the socketed ones) all have their notches facing towards the *back* of the QL.

Leave the casing screws for the moment, you first want to verify everything works as intended.

Power on your QL (do not insert the SD card yet) and make sure it boots up properly – Also check that all the keys are working. Once you have verified this, you can put the screws back in place.

If you use the QL-SD driver in the EPROM, you should see the driver initialisation message on your trusty F1/F2 selection screen:

```
QLSD WIN driver 1.07 WL + MK 2018
H/W v2 (Sw): Card 1 initialised
```

Congratulations! You have just successfully installed QL-SD in your computer.

5. First steps with QL-SD

In order to operate QL-SD properly, you (obviously) need an SD card. QL-SD will only work with SDHC card media (typically cards with a capacity of 4 GB or higher, although only a fraction of this size is actually used with the QL), cards made according to older standards like „MMC“ or „SD“ only, will *not* work.

QL-SD normally uses the SD card in a way that most QL emulators do: The QL file system resides in a large file on a DOS-formatted disk.

When starting up, the driver consults the card for a file named `QXL.WIN` and assumes it holds the QL file system. This is the same container format used by a lot of other QL platforms, like QXL, QPC, SMSQmulator, QemuLator, etc. At the moment you can use any of these to create the container file or download an empty container file from the QL-SD support page.

5.1. Selecting the File System Image to Use

Memory Considerations for QL-SD Image Files

Any block device on the QL needs memory for various purposes, and so does the QL-SD. Memory is used for slave blocks (temporary buffers that hold data while it is being shoved in and out to the SD card) or drive maps (maps that hold information where on the SD-card which piece of the respective file can be found). In a nutshell, the actual amount of memory needed mainly depends on the chosen file system image size - The larger it is, the more memory is needed for buffers and the map.

If you are tight on memory (maybe you only have 256kb memory extension), do not be tempted to choose a File system too large - You may end up with an unusable system because all your precious memory might be used up by the QL-SD driver for copies of the drive map and buffers. A recommended sensible setup for a QL with at least 256kB expanded memory would be a file system size of 64MB, if you have 640kB memory, you could go for 128MB image size.

Miracle x-Card owners should be able to easily use even larger File system sizes, although this has not been extensively tested by the time of writing.

Be aware the larger the buffer area needs to be, the longer the QL will need to search through it.

Very large file systems can severely slow down your QL.

The most extreme case is with an unexpanded QL: Originally, we thought it wouldn't even work with QL-SD - But a file system size of 3 MB seems to be the optimum working setup leaving just about enough free memory to be able to do at least some sensible work with it - Maybe it's about time you consider expanding your QL's memory?

If you are in doubt, choose a 64MB image (if you have an expanded QL) to start with. See how this works out with your particular QL for a while and decide then.

5.1.1. Unexpanded QL

For an unexpanded QL, the choice of file systems is rather limited: There is exactly one that you can sensibly use. This file system gives you 3MB of space to work with, on a typical unexpanded QL with no other tools loaded, you are left with about 60kB of free memory after mounting that image. This might be a severe limitation depending on the type of usage you take your QL to. Generally, it is not recommended to use QL-SD on unexpanded QLs.

5.1.2. QL with 256kB Memory Expansion

For QLs with a memory expansion up to 256 kB, a sensible file system size would be either 32MB or 64MB. This leaves you with enough elbow room to work with on the QL, while on the other hand you have plenty of space on the SD card (Remember: The Miracle Hard disk came with 40 MB...)

5.1.3. Fully Expanded QLs, Gold Card and SuperGold Card Users

Memory should not be so much an issue here - But remember, the larger you make your file system, the slower disk handling is going to be. Depending on your storage space requirements, 128MB to 256MB seem reasonable.

In any case, it is thoroughly recommended to start with the 64MB file system image and explore your actual requirements before building your system. Also you can just use several files, up to 8 can be used at the same time.

5.2. Installing the File System

This is probably the easiest step. Insert the SD card into your PC's card slot and rename the selected file system to „QXL.WIN“ and you're done. If you are using a new card, make sure no other files are on it before. If the card saw other uses before, consider to format it. QL-SD assumes the file system to be a continuous file of data blocks - If the card was in use before or there are other files on the card, the PC might scatter the QXL.WIN file all around the card - This will not work in the QL! **You can check if the file is continuous using the WIN_CHECK command explained later in this manual.**

5.3. Starting Up the First Time

Now comes the moment of truth: Insert the prepared SD card into QL-SD's SD slot and power up (or reset) your computer. You should be seeing the QL-SD driver's sign-on message followed by a either "Card 1 not found" or "Card 1 initialised".

"Card 1 initialised" mainly means that a SD card was found and could be communicated with, so your hardware works! It does not yet mean that a valid "QXL.WIN" file was found. It also means that the driver temporarily renames itself to MDV so when the operating system looks for MDV1_BOOT it will look for it on the SD card. After the first access the driver renames itself again to WIN.

QL-SD should now be ready - The device name used by QL-SD is normally `win1_`. Simply try a

```
dir win1_
```

This should give you a directory of the QL-SD file system (note you will not see the same files as when you do a directory of the card in your PC - QL-SD does not operate directly on the SD card, but instead in a file system image *within* the QXL.WIN file.)

Any file system command will work as usual, you simply use the win1_ device name instead of the flp1_ or mdv1_ you have been using up to now.

5.4. Changing SD Cards

Unlike the old QL-SD drivers the QL-SD v2 drivers are designed for "hot swapping" the card, so after waiting a second you can remove any SD card and replace it with another or even the same card after it has been altered in another computer. The driver will automatically detect if anything has changed and initialise the card again.

Note that any open channels on the old card will be forcibly closed!

5.5. Switching Off

As with hot-swapping the SD card the driver waits about one second to write the map on the SD card after your last write access to the card. (The driver waits whether there are more write accesses to come because it does not want to constantly update the map on the card and thus wear the card's Flash memory more than necessary). You might have noticed, that after about one second after the last write access the drive LED is switched on for a short period of time. **Do not switch off the QL before this has completed!** Failing to do so will most probably result in loss of the last writes or worse.

6. Data Exchange with your PC

Nowadays, when most of the software for your QL comes from the Internet, you might probably be wondering how you can transfer files from your Internet-connected PC to your QL. For a „normal“ PC without additional software, it is not possible to access the contents of a QXL.WIN file and thus read and write files in the QL file system contained therein.

6.1. Transfer Using an Emulator

It is recommended to use an emulator like QPC or SMSQmulator to copy data to an SD card. In QPC is recommended to mark the SD-card as removable, then you can even swap the card between the PC and the QL at will and both systems will notice the changes automatically. For example:

```
WIN_DRIVE 2,'D:\QXL.WIN'      if D is your SD card slot in the PC
WIN_REMV 2,1                  mark WIN2 as removable
```

You can then copy or unzip data from the PC hard disk using the DOS device to the SD card.

6.2. Transfer Using Disk Drives

You can also use floppy disks, if you have. QL software is typically that small that a lot of software can be pushed onto a single (even DD) floppy disk. You can also transfer zip files in compressed form, and you get even more software onto the disk. If you have owned that disk drive for a while, you might already know how to best transfer from a DOS- or Windows-based computer to a QL disk (There are some programs out there that facilitate FAT- to QL disk transfers. Search your

favourite QL software archive)

7. QL-SD Usage Recommendations

For a lot of users that are used to be using only floppy disks (or even Microdrives), the huge capacity of storage that you can have with QL-SD might be some sort of „Culture Clash“. This chapter gives some general recommendations on how to adapt your organisation of data and ways of working with this huge amount of storage. While we are trying to give some hints, this is obviously not the place to explain all of the possible QL mass storage extensions.

7.1. No more Reboots

From now on, the days of "Insert Disk and re-boot" will probably be gone for you. You *can* go on working like that, but you will be losing a lot of potential of your new set-up. Instead, invest a bit of time to develop a proper BOOT program that sets you up for rather longer sessions. There are a number of good examples for such BOOT files around - Check your favourite QL software repository or discuss the issue in one of the QL fora.

7.2. Make Directories your Friends

Large disk space will become largely unmanageable if you don’t organise your files into directories (It is not the slightest bit of fun to "CTRL-F5" yourself through screens and screens of file names because you decided to put everything into the root directory). As QL-SD comes with a fully-fledged Level 2 device driver, you can use "real" directories provided you are using at least TK2.

7.2.1. Organise your Disk Space

We would recommend a basic structure that employs a "sys_" directory that is going to hold everything that you want loaded at BOOT time using LRESPR. This makes maintenance of file locations, versions and updates much easier. Another one would be a "progs_" directory that would hold all frequently used executable programs that *do not expect a specific subdirectory structure* and *do not store a lot of support files in their program directory (*we just got rid of the mess in the root directory - you do not want to have it here instead).

Executables that do not fall into this category (Psion Exchange or C68 might be examples) should go into a directory of their own. Obviously, you would then have to remember where you have put that program before you would be able to start it - This is where the pth_, dev_ and sub device drivers (see next paragraph) come into play.

Note how the example picked pretty short names for the directory names - Make sure you don’t choose too long path names for the first level directories - A QL path name is restricted to 36 bytes of length (not including the „win1_“ device name), no reason to waste a single character.

7.2.2. Create a WORK_ directory

With a large disk like QL-SD, it is very easy to end up with a completely cluttered root directory after some time working with it - Make sure that when you start a new project (programming, word processing, whatever...), you create a new directory under win1_work_ and put everything that belongs to that project into that directory. You want your root directory clean and tidy without having to search a long time for a file you created last week.

7.2.3. The MAKE_DIR Command

MAKE_DIR is a command that comes with QL Toolkit 2. MAKE_DIR creates a "real" Level 2 directory on QL-SD (or other Level 2 storage devices). Level 2 directories are special files in the file system that hold the names of all further files contained in this directory. You can easily distinguish Level 2 directories from normal files - They are marked with a "->" following the

directory name when you issue a "DIR" command on the drive. MAKE_DIR takes one string argument - the full name of the directory to be created. MAKE_DIR takes account of the TK2 data defaults.

Level 2 directories should be created *before* the actual files that go into them are copied to the directory - So, if you have a number of files like

```
win1_work_file_asm
win1_work_file_rel
win1_work_another_file
```

You might want to *first* create the directory win1_work (By issuing the command "MAKE_DIR win1_work") *before* you actually copy the files there. Doing it the other way round will work just as well - and move already existing files with matching names into the new directory, but might waste disk space.

Level 2 directories can be deleted just like normal files using the DELETE command, but only if they no longer contain any files - You must first make sure a directory is empty before you can actually delete it.

7.2.4. Use DUP, DDOWN, DATA_USE and PROG_USE

When you only have been working with floppy disks or microdrives up to now, you probably didn't have much use for the above TKII commands - Now you have. Make yourself familiar with those commands and do use them.

In case you decide to use the pth_-Extension, you probably want to leave the Program Default Directory (set with PROG_USE) left untouched and constantly set to pth1_. Just add every new directory that is going to hold a specific program you regularly use to that path. Make sure pth_ items are sorted by usage frequency - The earlier a directory comes in there, the faster it will be found.

7.2.5. Use a File Manager

When trying to manage a lot of files in complex directory structures, file management programs can really help a lot. Q-Trans would be worth a recommendation.

7.3. Important Software for Large Disk Users

Make yourself familiar with at least the pth_ and sub_ device driver extensions (if you find one that actually works, the dev_ pseudo device would also be thoroughly recommended). Instead of searching the lost file yourself, the computer can do it for you. pth_ is useful for having the computer find the files you want to execute spread across a number of directories, dev_ and sub_ are particularly useful for programs that assume they can put or find all their files in the root directory or just cannot work in subdirectories (like the Psion Four).

The following is just a list of files you should at least have a look at, most are worth installing them:

- Toolkit 2 - absolutely essential
- pth_ device driver
- sub_ device driver
- dev_ device driver (Note: the DEV device included in the GoldCard and SuperGoldCard ROMs up to and including version 2.49 is NOT compatible with QL-SD. Please download and LRESPR the DEV device v2.05 from the QL-SD support page)
- CueShell (needs Pointer Environment)

- QPAC2 (needs Pointer Environment)
- Launchpad (Q-Trans)
- Pointer Environment
- FiFi (Wolfgang Lenerz) - A fast file finder that helps you locate files by content or name.

This is about all you need to know to be able to work with your QL-SD as a Hard Disk replacement. The rest of this manual can actually be skipped in case you are content with mass storage you have now ;) Most of the S*Basic commands that come with the driver will only needed for troubleshooting

8. Preparing the SDHC Card

The card needs to be freshly formatted FAT32 (A card that has been in use for a while might possibly only be able to store the image file fragmented). In order to prepare your card for QL-SD operation, you should have access to a PC with an SDHC-capable card adaptor (most of these nowadays are).

QL-SD has two ways to store your data: The "native" way of storing data on the card (that is the one we do not recommend and you should only use this method if you

1. know what you are doing
2. know why you are doing it that way.

We can pretty much imagine (1) might be the case, but not (2) - There are actually only very few very obscure reasons why you would want to use the cards in native mode. Data exchange with a PC or other "modern-day hardware" will not be possible in this mode.

The actually recommended method is to use the card in a mode that most QL emulators do: The QL file system resides in a large file on a DOS-formatted disk. The driver that came with the EPROM on your QL-SD is also configured to use this mode as default.

In case you don’t know how to format an SD card on your computer, refer to the text box below.

NOTE: Formatting a card will wipe all data from it!

Formatting an SD Card (Original borrowed from eLinux.org)

Windows

- *Download and install the SD Association's Formatting tool from https://www.sdcard.org/downloads/formatter_4/eula_windows/*
- *Open the Application you have just installed*
- *Set "FORMAT SIZE ADJUSTMENT" to ON in the Options menu.*
- *Make sure you have selected the Drive your SD Card is inserted in*
- *Click "Format"*

Mac

- *Download and install the SD Association's Formatting tools from https://www.sdcard.org/downloads/formatter_4/eula_mac/*
- *Select "Overwrite format"*
- *Make sure you have selected your SD Card, and not something else*
- *Click "Format"*

Linux

- *Use gparted (or the command-line version parted if you prefer), if you don't have it, install it as you usually would.*
- *Format the entire disk as FAT32 (FAT16 will not work! Make sure you select the correct disk!)*

8.1. Preparing the QL File System Image

The step above has prepared the SD card for use with a PC - You want it to be usable with the QL, however. So, the next step is to create the DOS-File that will hold the QL file system. As mentioned above, you can download a number of pre-created file system images of various sizes - If you can't find one that suits your specific needs, you can create your own images: A QL image file can easily be created on a PC by using any of the emulators.

IMPORTANT NOTE:

To keep the driver lean (to fit into the available ROM space) and fast, there is no full implementation of a VFAT32 file system in there. For this reason, the driver will only work if the QXL.WIN image file is located in ***continuous*** *sectors on the VFAT32 SD card partition.*

This is most easily achieved by copying the image on a freshly formatted SD card, or at least on a card that has not had files deleted from it since the last format. Deletion of files will create small areas of free space that a 'normal' VFAT32 driver would collate into larger space, re-use and allocate to files - That files are called 'fragmented' afterwards, because the sectors belonging to them are not in continuous space, but instead inter-mixed with other files' sectors.

The QL-SD driver is ***not able to handle fragmented image files.*** *It is not even detecting this situation automatically, but will instead happily overwrite areas of the SD card that do not belong to the image file - And thus might destroy other (PC-related) data on the card.*

You can use the new WIN_CHECK command to check if your image is continuous on disk.

The image file must reside in the SD-Card's root directory. It must not be resized, renamed or otherwise tampered with other than from the QL or a QL Emulator. The FAT32 system may contain further files, but caution must be taken so the native QL file system image is not harmed.

Memory Considerations when creating an Image File

Any block device on the QL needs memory for various purposes, and so does the QL-SD. Memory is used for slave blocks (temporary buffers that hold data while it is being shoved out to the SD card) or drive maps (maps that hold information where on the SD-card which piece of the respective file can be found).

The actual amount of memory needed depends on two parameters that need to be given when the WIN file is created: File system Size *and* Group Size*. File system size sounds easy - It simply specifies the amount of storage space you want in your image file - But it has some hidden implications: The larger you specify the File system size, the larger your disk map is going to grow - The more QL memory will be used by the driver for buffering the map. On an expanded QL, this should not be a problem unless you specify a ridiculously large file system - On an unexpanded box, it is - A large one.*

Group size is the size (in 512-byte-sized sectors) of an allocation block on disk. The smaller you make the group size for the same size of disk, the larger your map grows, as there are more groups to keep track of. The larger you make it, the larger your slave blocks grow (Groups are slaved together).

If you are tight on memory (maybe you only have 256kb memory extension), do not be tempted to make your File system too large - You may end up with an unusable system because all your precious memory might be used up by the QL-SD driver. A

recommended sensible setup for a QL with expanded memory would be a file system size of 64MB with a group size of 8. Miracle x-Card owners can easily go for larger File system sizes.

The most extreme case is with an unexpanded QL: Originally, we thought it wouldn't even work with QL-SD - But a Group Size of 8 and a file system size of 3 MB seems to be the optimum working setup - Maybe it's about time you consider expanding your QL's memory?

9. QL-SD Driver S*BASIC Procedures and Functions

The new QL-SD driver installs the following new procedures and functions into S*BASIC:

- **CARD_INIT**
- **WIN_CHECK**
- **WIN_DRIVE**
- **WIN_USE**

9.1. CARD_INIT card

This tries to initialised the specified SD card. `card` defaults to 1, which is fine for any single-slot QL-SD but can also be 2 if you have two SD slots. It doesn't show anything if successful or returns with an error if not.

9.2. WIN_CHECK drive

Check if the container file connected to WIN`x` is continuous. If it is not the command will return with an error.

9.3. WIN_DRIVE drive, card, name

Connect WIN`x` to a specific card and container file. By default, the QL-SD driver maps the drives like this:

Device	Card	Name
WIN1	1	QXL.WIN
WIN2	2	QXL.WIN
WIN3	1	QXL3.WIN
WIN4	1	QXL4.WIN
WIN5	1	QXL5.WIN
WIN6	1	QXL6.WIN
WIN7	1	QXL7.WIN
WIN8	1	QXL8.WIN

You can use several container files on the same SD card at the same time or on multiple SD cards if you have more than one slot. Also, the name "QXL.WIN" is just the default and should be used for the container to boot from, apart from that any name can be used for other container files.

Example:

```
WIN_DRIVE 1,2,"GAMES.WIN"
```

Mount the container file GAMES.WIN on card 2 to WIN1.

9.4. WIN_USE name

Change the device name of the driver. It is possible to rename the QL-SD driver to something else.

This can, for example, be useful for compatibility with older programs that cannot recognise devices other than those installed on the basic QL. You can use any three characters. These must be upper case, and the fourth character must always be a zero.

Example:

```
WIN_USE "MDV"
```

The QL-SD now pretends to be a Microdrive.

WIN_USE without a parameter resets the driver name to WIN.

10. Direct Sector Access

For compatibility with existing software that does direct sector I/O on hard disks, the QL-SD driver supports this facility on native QDOS volumes.

To do this, open the special file "*D2D", as in:

```
OPEN #3,"WIN1_*D2D"
```

The driver also allows the shortened form:

```
OPEN #3,"WIN1_*D"
```

The channel so opened has the following limitations:

- Data must be read or written in 512 byte blocks.
- The file pointer is interpreted as a sector (LBA) number, relative to the start of the partition.
- Only absolute file pointer movements are permitted.
- It is not possible to read a sector beyond the end of the partition. Any attempt to do so will return "End of File".
- Note that no files can be open on the logical drive with which you wish to perform direct I/O.

11. QL-SD Driver

The QL-SD driver implements a Level 2 Directory Device Driver with subdirectories. It is based on the SMSQ/E Level 3 WIN hard drive driver and adapted by Wolfgang Lenerz and Marcel Kilgus for QL-SD.

Important: When working with the QL-SD driver a minimum of 256K expansion memory is thoroughly recommended. You will be able to use QL-SD on an unexpanded QL, but possible File system sizes are very limited.

11.1. Device Name

The name of the QL-SD device is normally "WIN". If this causes problems on your system, or you wish to change it for some other reason, you may alter the name of the device with the **WIN_USE** command.

11.2. Starting the Driver

The software comes in RAM or ROM versions. The ROM version starts when booting the machine.

The RAM version is started by loading the file **driver_bin**. It can be loaded either with LBYTES followed by CALL or with the Toolkit II LRESPR command.

The driver prints a message identifying itself to channel 0, for example:

```
QLSD WIN driver 1.07 WL + MK 2018
H/W v2 (Sw): Card 1 initialised
```

11.3. ROM auto-boot

For a QL equipped with QDOS to automatically boot from the SD-card the ROM contains code that, if a card is found on boot ("Card 1 initialised"), the device is renamed briefly to "MDV" as QDOS is looking for a file called "mdv1_boot" to boot up. After QDOS tried to open "mdv1_boot" the device name automatically reverts to "WIN".

This also means that if you insert the SD card after the driver tried to initialise it auto-boot will not work.

This mechanism is disabled on SMSQ/E as that can be configured to look for the boot file on WIN1_ anyway.

11.4. Low level API

The driver incorporates a low-level sector API so 3rd party drivers or applications can access SD card sectors without having to write their own low-level code. To use the API the driver linkage block must be found first:

```
; System variables
sys_fsdl        equ     $0048           ; long    Filing System Driver List

; QLSD device driver linkage block entries
ddl_ddlk        equ     $0018           ; long    io driver linkage
ddl_dname       equ     $0042           ; string  driver name
ddl.dname       equ     $57494E30       ; WIN0
qlsd.magic      equ     $514C5344       ; Driver ID for 3rd party software
qlsd_magic      equ     $0470
qlsd_version    equ     $0474           ; Driver version
qlsd_rscard     equ     $0478           ; Read card sector
qlsd_wscard     equ     $047e           ; Write card sector
qlsd_inicard    equ     $0484

;+++
; Search QLSD WIN driver linkage block
;
;       d0  r err_fdnf or 0
;       d1  s
;       a3  r driver linkage block
;---
get_linkage:
        moveq   #0,d0
        trap    #1
        move.l  #ddl.dname,d1                   ; 'WIN0'
chk_dev:lea     sys_fsdl(a0),a3                 ; device list
srch_lp:move.l  (a3),d0                         ; is there another device?
        beq     err_nf                          ; no, leave with error ->
        move.l  d0,a3                           ; this is the device
```

```
        cmp.l   ddl_dname+2-ddl_ddlk(a3),d1     ; it is WIN?
        bne     srch_lp                         ; ... no
        cmp.l   #qlsd.magic,qlsd_magic-ddl_ddlk(a3) ; Check if this is
                                                    ; the driver we expect
        bne     srch_lp                         ; ... no
        sub.w   #ddl_ddlk,a3
        moveq   #0,d0
        rts
err_nf: moveq   #err.fdnd,d0
        rts
```

There are three functions for initialising the card, reading and writing raw sectors (raw meaning here that they don't need to be part of a QXL.WIN file, the addressing is absolute).

Vector $0484 Initialise SDHC card

qlsd_inicard

Call parameters

D5	Card number (1..3)
A3	Driver linkage block

Return parameters

D5	Preserved
A3	Preserved

Error returns:

ERR.FDIU: QL-SD busy

ERR.MCHK: Error accessing card

Vector $0478 Read sector(s) from SDHC card

qlsd_rscard

Call parameters

D0	Absolute sector number
D2	Number of sectors to read
D7	Card number (1..3)
A1	Address to read into
A3	Driver linkage block

Return parameters

D0	Error code or 0
D2	Preserved
D7	Preserved
A1	Preserved
A3	Preserved

Error returns:

ERR.NC: QL-SD busy

ERR.MCHK: Error accessing card

Vector $047e Write sector(s) to SDHC card **qlsd_wscard**

Call parameters		**Return parameters**	
D0	Absolute sector number	D0	Error code or 0
D2	Number of sectors to read	D2	Preserved
D7	Card number (1..3)	D7	Preserved
A1	Address to read into	A1	Preserved
A3	Driver linkage block	A3	Preserved

Error returns:

ERR.NC: QL-SD busy

ERR.MCHK: Error accessing card

Some C68 compatible example code including the library written in assembler is available on the QL-SD support page.

12. Some Technical Details

This chapter covers some of the inner workings of QL-SD for the technically interested. You don't actually need to read or even understand this chapter in order to operate QL-SD.

12.1. The Address Map

QL-SD uses quite a bit of address space in the QL:

$FEE0 – $FFFF	QL-SD Hardware registers
$C000 – $FEE0	QL-SD on-board ROM (driver)
$0000 – $C000	Minerva 1.98

It is obvious from the above address map that QL-SD is not compatible with any other QL extension that uses the 16k ROM slot addresses ($C000-$FFFF)

12.2. The Hardware

QL-SD's hardware consists mainly of the Lattice Mach 4000 CPLD, a pretty modern (and fast - it runs at 50 MHz) reprogrammable logic chip. The CPLD does the address decoding for the various QL-SD hardware registers that it provides and for the QL-SD EPROM.

The CPLD provides background SPI (Serial Peripheral Interface - The standard used to access SD cards) transfer to and from the SD card. In its normal mode, the CPLD acts as a serial-to-parallel converter clocking the 8 bits of one byte directly in and out of the SD card without the need of further intervention from the QL. (There is, however, SPI mode 0 which allows the QL to directly access the data and clock lines of the SPI bus - Which is obviously much slower and only used during driver startup)

As QL-SD lives in ROM space, there is actually no direct way for the QL 68008 CPU to write

access the QL-SD registers. A write of one single byte to QL-SD is thus initiated by a read access to a specific QL-SD register bank offset by the value of the byte to be written to QL-SD - The CPLD translates this into a write into its data register.

12.3. Register description

Registers are read to be activated. Only two registers actually return usable data.

Name	Address	Purpose
IF_ENABLE	$FEE0	Enable the QL-SD interface
IF_DISABLE	$FEE1	Disable the QL-SD interface. This is located after IF_ENABLE so if some software just scans the ROM area it will not leave the interface enabled
IF_RESET	$FEE2	Reset the interface into a known state
IF_VERSION	$FEE3	QL-SD v2 only: a bitfield that returns the version of the QL-SD interface and in some cases, additional data. See below for the description
SPI_READ	$FEE4	The last byte read using background I/O is provided here. In bit-banged I/O it contains the state of MISO in the lowest bit
SPI_XFER_FAST	$FEE5	Switch the SPI mode to fast background I/O. This is the default used during normal driver operation. It clocks the data with 25Mhz to the SPI bus, which also means that it's fast enough to output any data before the next 68k memory cycle starts (no waiting needed)
SPI_XFER_SLOW	$FEE8	Switch the SPI mode to slow background I/O. The data is clocked with 25Mhz/64 = 390.625kHz. The current driver never uses this mode
SPI_XFER_OFF	$FEEA	Disable background I/O and enable bit-banged I/O
SPI_DESELECT	$FEF0	Disable all chip-select lines
SPI_SELECT1	$FEF1	Enable chip select line 1, disable all other lines
SPI_SELECT2	$FEF2	Enable chip select line 2, disable all other lines
SPI_SELECT3	$FEF3	Only available on models with a 3rd SPI port on the upper connector. Enables the corresponding chip select line
SPI_CLR_MOSI	$FEF4	Clear MOSI line in bit-banged I/O mode
SPI_SET_MOSI	$FEF5	Set MOSI line in bit-banged I/O mode
SPI_CLR_SCLK	$FEF6	Clear SCLK line in bit-banged I/O mode
SPI_SET_SCLK	$FEF7	Set SCLK line in bit-banged I/O mode
SPI_XFER	$FF00… $FFFF	Output the lower 8-bits of the address as data on the SPI bus in background I/O mode

12.3.1. IF_VERSION

There are two variants of the QL-SD hardware that differ in the use of the upper port that: one variant uses the pins to disable various bits of the QL-SD interface ("Switch" version), one variant has a full-fledged 3rd SPI port there ("SPI" version). The type of the current hardware is shown in the boot message.

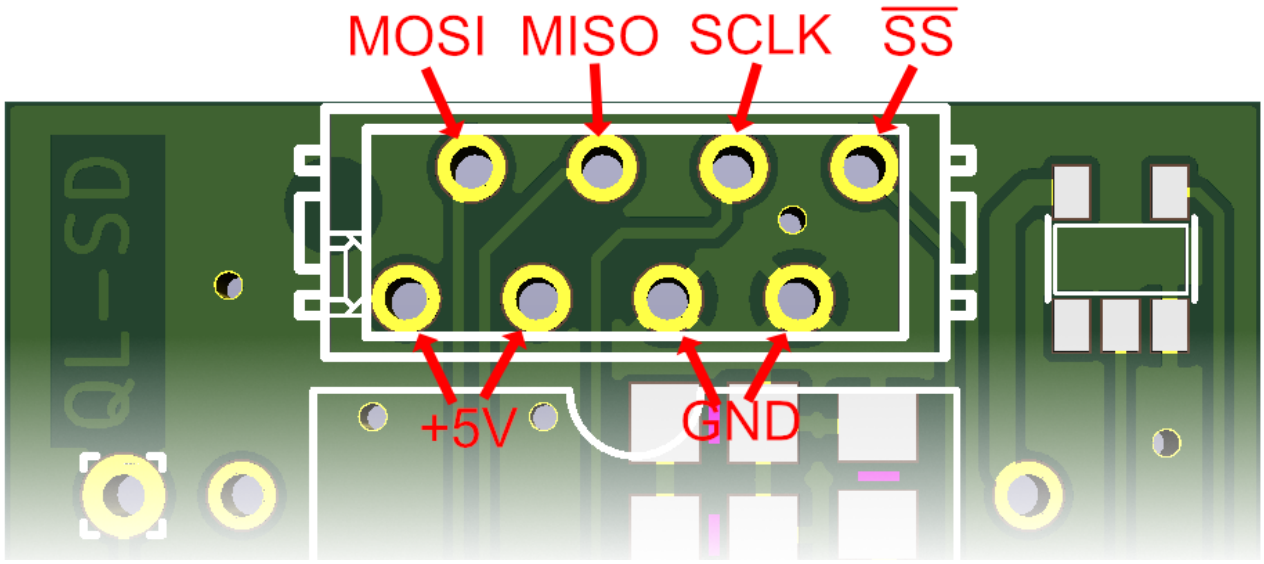
By default, the "SPI" version is shipped to provide a ready-to-use interface for future hardware extensions. If you want to disable QL-SD or at least want to disable the OS part of the ROM so it could be used with another ROM extension like "Minerva MK II" you need to explicitly order the "Switch" version.

SPI version				
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3..0
0	0	0	0 (= "SPI" version)	0001 = QL-SD v2
Switch version				
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3..0
IO1	IO2	IO4 (QDOS ROM enabled)	1 (= "Switch" version)	0001 = QL-SD v2

In the switch version, the IO1 and IO2 lines can be used and read as general-purpose input lines. The lines have internal pull-up resistors, so they are "1" by default and turn to "0" when connected to GND.

QL-SD hardware before v2 does not have this register and just returns the contents of the ROM there (usually $00 or $FF).

12.4. "SPI" version pinout



The pins form the standard SPI protocol and are not explained further here. There is a new system variable `sys_qlsd` ($00c1) which has its high bit set if the QL-SD hardware is currently in use. Do not attempt to use the 3rd SPI port while the bit is set. Acquire the use of the QL-SD interface

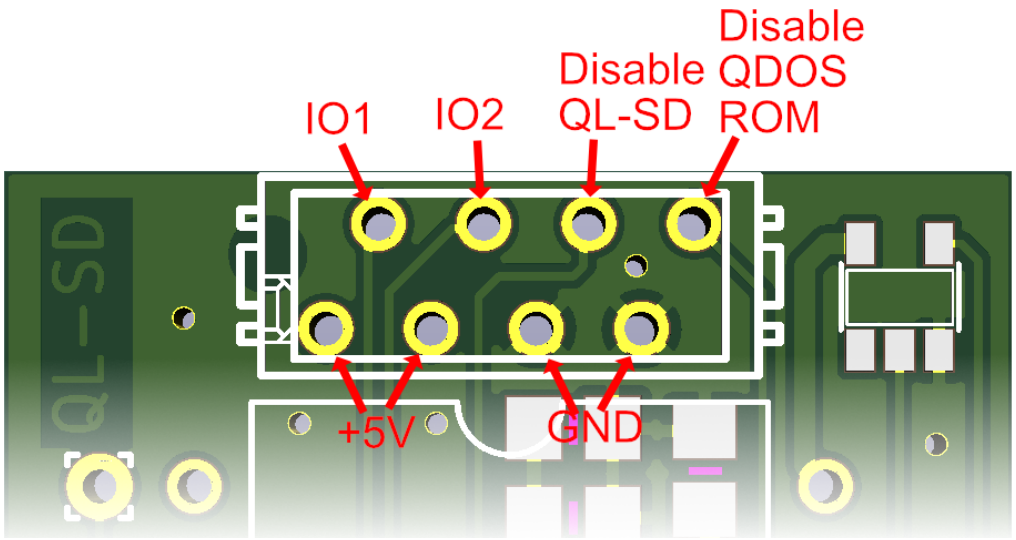
```
; A6 points to system variables
        bset    #7,sys_qlsd(a6)     ; try to acquire interface
        bne.s   qlsd_in_use         ; ... currently in use, wait
```

```
; Can use QL-SD interface now
```

And release it using (of course only if you have successfully acquired it!)

```
; The interface is usually used in the "fast background I/O" mode,
; if you need to change this then revert to the fast mode
; before releasing the interface!
;
; A2 points to if_base ($fee0)
        tst.b   spi_xfer_fast(a2)   ; fast I/O mode again
        bclr    #7,sys_qlsd(a6)     ; release interface
```

12.5. "Switch" version pinout



All inputs have internal pull-up resistors, so they are activated by connecting them to GND (by a switch or transistor usually).

IO1/IO2 are general purpose input pins and can be read by software using the IF_VERSION register.

The "Disable QL-SD" pin, when pulled to GND, disables the IF_ENABLE register function of the QL-SD interface and disables the upper 16KB of the integrated EPROM chip. Probably best not to do this while the QL is running (or at least hold the reset button while doing so).

The "Disable QDOS ROM" pin, when pulled to GND, disables the lower 48KB of the integrated EPROM chip. Probably best not to do this while the QL is running (or at least hold the reset button while doing so). The status of this pin can be read using IF_VERSION register.

13. Known Issues

13.1. Hardware

- QL-SD will most probably not work with other hardware that uses I/O addresses in the range of $C000 to $FFFF (the ROM slot area) or copy-protected software that wants to see its ROM in this area (The Eidersoft ICE and mICE suite of programs are known examples)

13.2. Driver

- There are several programs out there (especially older games) that were written assuming the memory map of an unexpanded QL. Those programs load code onto fixed addresses without using the proper QDOS mechanisms to claim that memory. They (“Cuthbert in Space” is a known example) will happily overwrite crucial memory areas used (and properly claimed) by the QL-SD driver and thus crash the QL.

14. Acknowledgements

Peter Graf has developed the initial hardware and firmware.

Marcel Kilgus developed QL-SD v2 which was finally completely (Super-)GoldCard compatible. The hardware is a new design but compatible to Peter Graf's version, so the older boards can be updated.

The QL-SD driver employed by QL-SD v2 was written by Wolfgang Lenerz and Marcel Kilgus, based on the SMSQ/E source code from Tony Tebby. It, too, is compatible to the older QL-SD boards.

The manual has been written by Tobias Fröschle and Peter Graf and heavily updated for QL-SD v2 by Marcel Kilgus.

15. Copyright Notice & License

16. Disclaimer

QL-SD 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.

No responsibility is accepted for the loss of data or consequent damage of any kind resulting from the use of this device.

And a last word....

Please make sure you install and operate QL-SD with the best possible care and forethought you can. Keep to the operating procedures described in this manual. If something is unclear - Do come back to us, but please don't expect professional level support. The team that brought you QL-SD is a team of hobbyists that neither make any money from it nor have the time to support you with fixing broken parts, damaged partitions and exploded file systems.

Before you care to bother us, thoroughly try yourself, or try and find someone around you with a bit of QL experience who might be able to support you. QL-SD is intended as a community project where people help each other to fix problems. The QL-Forum (http://www.qlforum.co.uk/) or the QL Mailing lists might be a good place to start looking for help.

*This does not mean you're left alone and on your own with any problems you might have - Just make sure **you** support us as much as **we** support you. This might probably give us a bit more hobby time to supply you with more exciting hardware and software developments around the QL.*