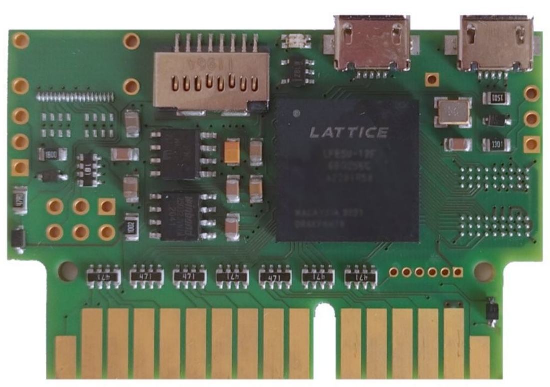
QIMSI

QL INTERFACE

USER MANUAL



Version 01/2024

Chapter 1: Overview

Introduction

The name “QIMSI” stands for “**Q**L **I**ntelligent **M**ouse and **S**torage **I**nterface”. The similarity to “QIMI”, the traditional QL mouse interface, is intended. This interface fits into the standard QL ROM Port and requires no modifications to the QL case or motherboard for its basic functionality. It thus is especially suitable for users who prefer to keep their QL in its original form.

Key Features

QIMSI is based on a specialized chip (FPGA) which includes various QL interfaces, a CPU and some ROM/RAM on a single integrated circuit.

The basic QIMSI functions for the Sinclair QL home computer are PS/2 mouse and mass storage interfaces, a PS/2 keyboard interface and an optional sampled sound interface.

microSD Card Interface

QIMSI offers an SDHC card interface for the QL, supporting one microSD card slot. It works like a hard drive for the QL and can be accessed via „WIN1_” for example. Function-wise this interface is compatible to the internal QL-SD interface originally designed by Peter Graf.

Mouse Interface

QIMSI comes with a PS/2 mouse interface for the QL, compatible with the Pointer Environment. The mouse scrollwheel is supported by the interface, but may lack driver software support.

Keyboard Interface

QIMSI also offers a PS/2 keyboard interface for the QL. Currently software support it is available for Minerva. Also under SMSQ/E where it requires a Gold Card or a Super Gold Card.

Sound Interface

QIMSI includes a sampled sound interface for the QL’s internal speaker. Connecting the speaker requires a small modification of the QL’s mother board.

ROM

QIMSI provides a 16 KB ROM area for the QL in the address range $C000 … $FEC8. The ROM contains the driver for the microSD card interface. The driver is compatible to the internal QL-SD interface.

Other Specifications

Internal CPU (MiniQ68)

QIMSI has an internal microprocessor, which is code compatible to the 68000, and runs at 40 MHz clock rate. It is not directly accessible from the QL side, but can load and run code from SDHC card or a serial port. The CPU is compatible to the one used by the Q68 computer and is surrounded by a similar, but reduced infrastructure. This system may also be referred to as „MiniQ68“ in this document. For the basic QIMSI functions, the MiniQ68 can be ignored. Due to memory limitations, no operating system is provided for the MiniQ68.

Internal ROM/RAM

The QIMSI has two small ROM/RAM areas of 32 KB and 12 KB size for the internal CPU, running without waitstates. They are part of the MiniQ68 and not accessible from the QL side.

QL-MiniQ68 Link

QIMSI provides a bidirectional fast data link between the QL and the MiniQ68. It is buffered with two 512 Bytes hardware FIFOs, one for each direction.

Serial Port

The MiniQ68 has access to a primitive serial port with up to 115200 Baud. This port is not directly accessible from the QL side.

I/O Port

The MiniQ68 has access to a single wire I/O port. It is also not directly accessible from the QL side.

Board Size

The QIMSI board has a small size of 53 x 35 mm.

Power Supply

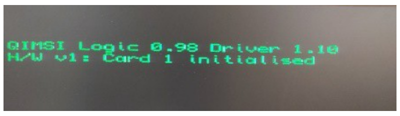
The QIMSI is powered by the QL 5V supply available on the ROM port.

Setup

The QIMSI interface fits into the ROM port of the Sinclair QL. Before plugging it in, please make sure to power-down the QL. Take care to attach all add-ons (Mouse, Keyboard and SD-Card) before power-up.

When attaching any components to the board, please pay attention to the correct orientation.

QIMSI should be recognized on the boot screen and also SD-card initialization should be displayed, if a card is inserted.



```
QIMSI Logic 0.98 Driver 1.10
H/W v1: Card 1 initialised
```

Figure 1: Boot screen with QIMSI properly recognized and SD card initialised

Important notice: Before using QIMSI, please detach all other hardware devices which also use the same address space, namely the ROM area between $C000 to $FFFF.

As a special exception, QL-SD with CPLD logic version 0.92 (jumperable) can coexist with QIMSI, if the QLROMEXT board of QL-SD is jumpered to IO3+IO4. In that case, please do not insert a microSD card into the QIMSI socket.

Chapter 2: Hardware

Board Overview

The connectors of the QIMSI interface are located according to the following figure:

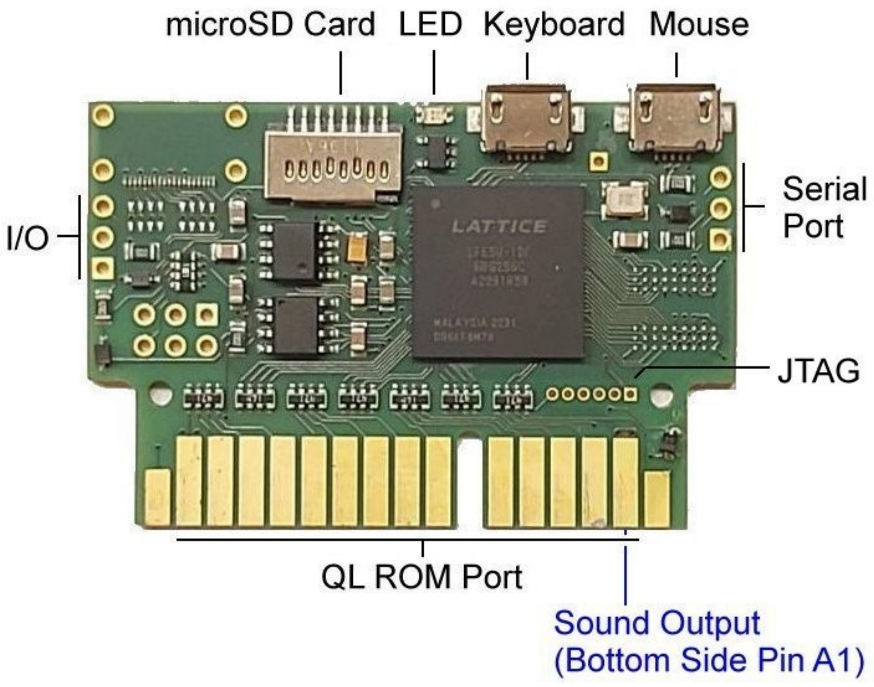


Figure 2: Connector Locations

MicroSD Card

The card slot accepts a standard micro SD card. The card should be an SDHC type card and must be FAT32 formatted. While QIMSI is electrically capable of hot-swapping the card during QL operation, this is not recommended to avoid mechanical fluctuation of the QIMSI board.

Mouse

A PS/2 compatible mouse can be connected to the right hand side micro USB port. Nowadays many of these mice are dual function, supporting PS/2 as well as USB, and are terminated in a USB connector. To connect the USB Type A connector you need an “USB on the go” (OTG) adaptor for micro USB, **not** USB-C. These adapters are cheap and easily available in different variants. One type is shown in the picture below for your reference. (For a traditional PS/2 mouse, a passive PS/2 to USB adaptor can be added.)

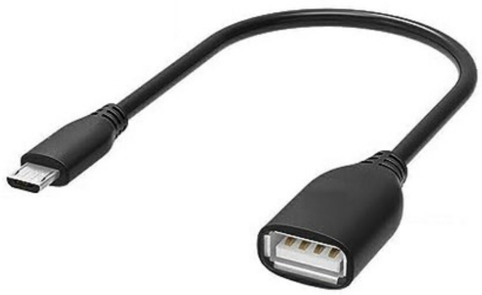


Figure 3: Typical micro USB on the go (OTG) adaptor

Figure 4: Typical passive PS/2 to USB adaptor

Many different mice were tested successfully. Nevertheless, due to the wide range on the market, it may happen that some types of mice are not compatible. You may contact the QL forum for suggested mice known to be compatible.

Please note that, whilst a micro USB connector is fitted, only the PS/2 protocol is currently supported by QIMSI – i.e. QIMSI does not provide actual USB support and only 'pure' PS/2 or PS/2/USB 'combi' mice are compatible.

Keyboard

A PS/2-USB combo keyboard can be connected to the left hand side micro USB port. Like for the mouse, an OTG adaptor is required. Plus a passive PS/2 to USB adaptor, if a traditional PS/2 keyboard is to be used. A software driver is currently abvailable only for SMSQ/E.

LED

A dual color LED indicates QIMSI operation:

Red color signals an ongoing micro SD card access.

Green color single flash during power-up signals detection of a Standard PS/2 mouse.

Green color double flash during power-up signals detection of an extended PS/2 mouse with scrollwheel.

After the first QL-side access to SD card, the green LED lights up continually.

The green LED can be user-programmed afterwards. A byte-wide read access to address 65238 switches the green LED off (e.g. PRINT PEEK (65238)), and to address 65239 switches the green LED on (e.g. PRINT PEEK (65239)).

Sound Output

A mono sampled sound output is also provided and can be connected to the internal QL speaker. As there is no dedicated sound signal connection on the standard QL ROM port, a small hardware modification is necessary to route the sound output signal to the internal QL speaker. This modification will be now be explained in detail.

The QIMSI sound output is provided at the last „large“ right hand **bottom side** pin, which is QL ROM port pin A1 as described in the QL User Guide. This pin is normally unconnected to the QL mother board, but may be linked to +5V on some boards. To connect the QIMSI sound output to the QL speaker, the user needs to first cut the connection from the ROM port socket to the QL mother board. The following picture shows in detail where to cut the pin.

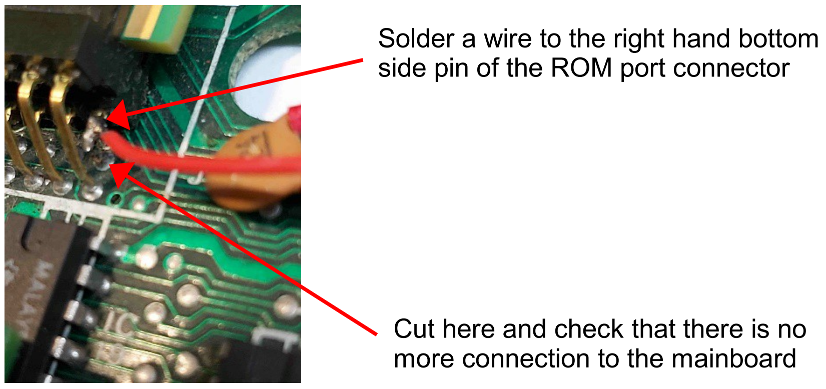

Solder a wire to the right hand bottom side pin of the ROM port connector
Cut here and check that there is no more connection to the mainboard

Figure 5: ROM port modification

Solder a length of insulated wire to the pin side on the ROM port socket thus establishing a connection to the QIMSI PCB. Be careful when doing this modification, soldering skills are necessary.
The other end of the wire must be attached / soldered to the QL speaker pin header. As there are several different QL mainboard configurations, this manual only gives partial advice on how to do this.

On some QL mainboards (Issue 6) the speaker pin header is beneath the big heat sink of the 5V regulator 7805. The heatsink must be carefully removed before the pin header can be accessed. Figure 6 shows a QL mainboard with removed heatsink and the dedicated pin header.

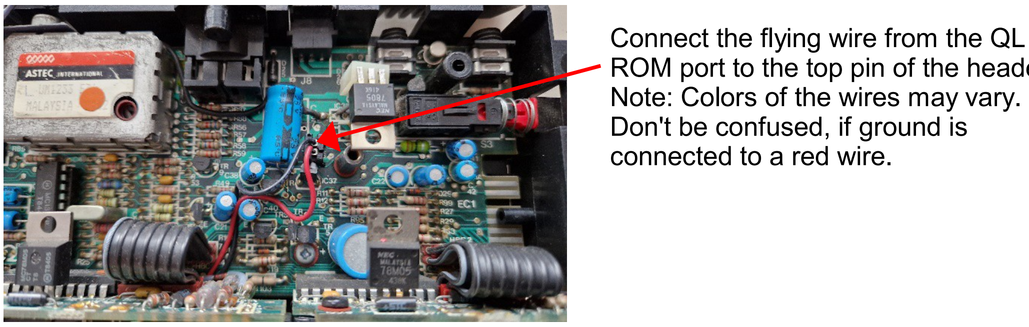

Connect the flying wire from the QL ROM port to the top pin of the header. Note: Colors of the wires may vary. Don't be confused, if ground is connected to a red wire.

Figure 6: Speaker pin header modification (Issue 6)

One pin of the pin header is ground, the other one is speaker output. The flying wire must be connected to the speaker output. Do not connect it to ground! If you are unsure which pin is ground, please use a multimeter to check the resistance of the pins to mainboard ground, which can be easily accessed by the modulator housing.

Figure 7 shows a different QL mainboard (Issue 5) where the speaker pin header is just at the right hand side of the ZX8302 chip, near the lower left corner of the modulator. It is more easily accessible compared to the one above, the heatsink does not have to be removed.

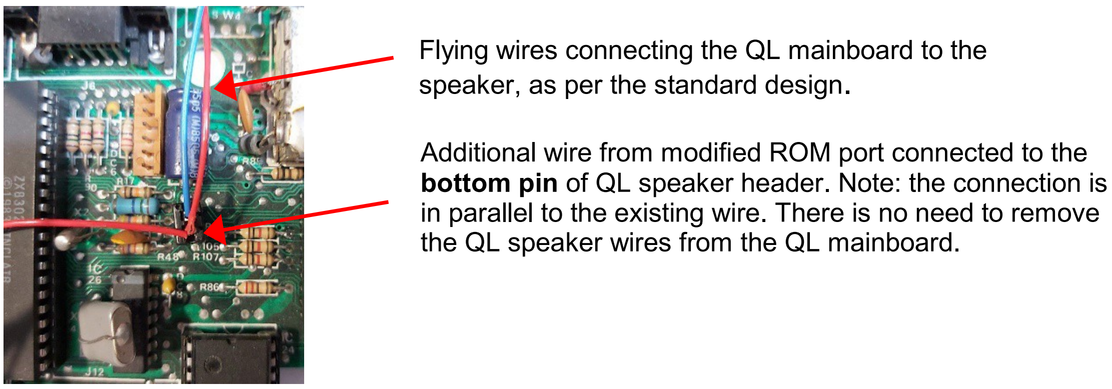

Flying wires connecting the QL mainboard to the speaker, as per the standard design.
Additional wire from modified ROM port connected to the **bottom pin** of QL speaker header. Note: the connection is in parallel to the existing wire. There is no need to remove the QL speaker wires from the QL mainboard.

Figure 7: Speaker pin header modification (Issue 5)

On those mainboards, the flying wire must be connected to the bottom pin of the QL speaker pin header. There might be further mainboard variants, where the location differs from the two examples shown here. Before powering on the QL, please double check all solder joints and modifications so as to not have any shorts, e.g. by solder splash. Test that your modification works as expected.

There are example sound files „catwalk_ub“, „forgotten_ub“ and „western_ub“ that can be played using a simple demo sound player called PLAYSS, which are both provided. Please note that PLAYSS is implemented without a realtime driver, so if other jobs are running, they might lead to disrupted playback. You can list the running jobs with the Toolkit 2 Basic command **JOBS**.

E.g. try **EX playss;catwalk_ub**. Without further commandline parameter, PLAYSS tries to reduce the so-called SLAVE buffering of the operating system, because it slows down read access to mass storage a lot. In order to reduce SLAVEing, PLAYSS temporarily allocates most free memory of the machine. If this is not desired, please use commandline option -s, for example **EX playss;“-s catwalk_ub“**, to keep SLAVEing active. If sound data is short enough, it could be loaded it to a ramdisk before playing with the -s option.

Do not connect the ROM port pin to the speaker <u>without</u> first cutting the ROM port pin as described above! This modification comes at your own risk! No responsibility can be taken if your QL or QIMSI is damaged. If you are unsure, please consider to not do the modification.

Note: There is a protection diode on the QIMSI board. This means: If you do not want to use the sound capabilities there is no need for any modification. The QIMSI cannot be harmed by just putting it into the ROM port.

QL ROM Port

QIMSI is connected to the QL bus on the ROM port with an 8 bit data bus and 16 bit address bus. In the QL memory map, the extension bus is located at $0000 ... $FFFF, of which $C000 ... $FFFF are used by QIMSI. For a decription of the signals, please refer to the QL User Guide.

Deviant from the QL User Guide, QIMSI provides a sound output on the „unused“ QL ROM port pin A1. This output is protected against the 5V voltage found on pin A1 of some QL mainboards.

Serial Port

This 3 pin header provides an RS-232 like port for the internal MiniQ68. While the input is fully RS-232 compliant, the output works with a reduced voltage swing. It is tested to work with QL, Q40/Q60, Q68 and PCs, if the cable length is kept below 2 m.

Pin	**Description**
1	Ground (marked by rectangular pad)
2	Transmit Date (TxD) Output
3	Receive Data (RxD) Input

I/O Port

This 3 pin header allows to connect an I/O signal to the internal MiniQ68 or to access the 5V power supply.

Pin	**Description**
1	Ground (marked by rectangular pad)
2	+5V Power connected to QL ROM Port
3	I/O Signal with 47 Ω series resistor and clamp diode to 3.3V

JTAG

This connector is for manufacturing use only.

Chapter 3: Software

Mass Storage

QIMSI relies on SDHC memory cards for mass storage. For use with QIMSI, they must be partitioned and the first primary partition must be in FAT32 format.

The files used for QIMSI must the be put into that FAT32 partition, which should be possible by copying from any Linux, Windows or MAC machine. However, make sure to understand the special precautions for these files as described in the following chapter.

QIMSI Container Files

The QIMSI is a QL-native hardware device, and files in the FAT32 partition serve only as containers for direct lowlevel access by QL-native drivers.

To access these containers efficiently, QIMSI expects the container files not to be fragmented in the FAT32 file structure. Once QIMSI has found the beginning of a container file, it assumes the rest of that container file lies in continuous sectors on the card. Moreover, these container files must be located within the first 16 root directory entries of the FAT32 partition.

The best way to achieve this, is to make sure that before writing the the container files, the card is freshly formatted. Then write each container file immediately after formatting the card.

The container files follow a specific naming scheme. The file name of a container file must have 1 to 8 characters, followed by a decimal point and a three letter extension.

Preparing an SDHC Card

This cannot be done on the QIMSI itself yet. To prepare a new card, the following procedure is recommended:

- Linux: Use gparted or the command-line version parted.
  Then format the entire disk as FAT32, or at least the first partition if you have more.

- Windows: Download and install the SD Association's Formatting tool from:
  https://www.sdcard.org/downloads/formatter/sd-memory-card-formatter-for-windows-download/
  Then open the installed tool, set "Format Size Adjustment“ to „On“ and click „Format“

- MAC: Download and install the SD Association's Formatting tool from:
  https://www.sdcard.org/downloads/formatter/sd-memory-card-formatter-for-mac-download/
  Then open the installed tool, select "Overwrite Format“ and click „Format“

Now you are ready to write the container files. All QL side files are stored within a container file named QLWA.WIN or further containers. This means that on your PC, you can only see the container file(s). The container file system format is QLWA and can be read and created with most common QL emulators. Please note that the larger the size of the container file, the more QL memory will be used by the driver. On QL setups that are tight on memory, it is better to go for a modest container size of 32 or 64 MB. Or even 3 MB on an unexpanded QL - but it is not recommended to use QIMSI without any memory expansion.

Important reminder: The container file must not be fragmented. You will experience data loss if the file is fragmented. Therefore it is recommended to have only the container file(s) on this card and not to delete files from this card, unless a fresh format is done. Also do not have any other (PC) files on this card. If you have ordered a microSD card along with QIMSI, you will already find a container file on the card and you are ready to go. The default assignment of QL WIN drives and container filenames is depicted below.

WIN1_	QLWA.WIN
WIN2_	QLWA2.WIN
…	…
WIN8_	QLWA8.WIN

Always have a backup of your container file(s)!

Using Directory Structures

A directory is where the system expects to find a file. On small QL storage devices like microdrive or floppy disk, this was often as simple as the name of a device, like FLP1_.
But the QIMSI interface, just like QL-SD, offers much more disk space and it is therefore useful to organise files into subdirectories. QL Toolkit 2 or the SMSQ/E operation system come with a number of commands to use directory structures. A few are briefly covered as follows:

MAKE_DIR

The command MAKE_DIR is included in the Gold Card, Super Gold Card, SMSQ/E and newer versions of Toolkit 2. It takes a string argument with the name of a level 2 subdirectory to be created. That subdirectory allows a group of files to be regarded as one unit when the contents of a medium is listed. When a directory listing is shown with the DIR command, they are marked with „->“. Empty level 2 directories can be deleted with the DELETE command. Please note: A QL path name is restricted to 36 bytes of length.

DUP

The command DUP moves up the directory tree by one level.

DDOWN

The command DDOWN takes a string argument and moves down the directory tree to the specified subdirectory.

PROG_USE

The command PROG_USE takes a string argument and sets the default program directory accordingly. The default directory is then is used to find program files for EX, EXEC, EW, EXEC_W.

DATA_USE

The command DATA_USE takes a string argument and sets the default data directory accordingly. The DATA_USE default is used for most filing system commands in Toolkit 2 or SMSQ/E.

Initializing a card

An SD card is automatically initialized at boot time or before the first access.

Swapping cards

Theoretically, cards can be swapped in and out, even when the system is running - but this is not a recommended practice. If you insist on doing this, you must be aware of the following:

1. The QIMSI board must sit firmly inside the QL ROM port connector. It must not be mechanically moved while the card is swapped.

2. Do not remove a card when there are files still open and certainly not whilst the machine is reading/writing to a card. If you remove a card while there are still files open or files being written, data loss will occur.

BASIC commands for WIN drives

The BASIC commands related to WIN drives are as follows:

WIN_DRIVE *drive, card, filename$*

Assigns a container file on a card to become a drive. Where

d*rive* is the WIN drive number (1... 8) to be assigned,

card is the SDHC card on which the file can be found (always 1 on QIMSI) and

file_name$ is the name of the container file. The file name MUST be in "8.3" format, i.e. a name of one to 8 characters, a decimal point and an extension of up to three letters. The extension, if present, must be separated from the name by a period.

WIN_USE *device*

Allows to assign another three letter device name to the device driver. If no device is specified, the device name is returned to the default „win“.

This command is also useful, if a different device, e.g. floppy contains a BOOT file. In that case, the WIN drive might not be found unless a WIN_USE command is given.

WIN_CHECK *drive*

Checks whether a WIN container file on the card is indeed in continuous sectors on the card, where *drive* is the container file containing the WIN drive in question. If the command does not return an error, the container file corresponding to the drive is okay.

Mouse

During the QL boot process you need to load the required mouse driver. After having activated Toolkit II and loaded the Pointer Environment, you can load the driver by adding “**LRESPR WIN1_mouse_bin**” to your boot file. Please note that the current driver requires the Pointer Environment and can not work otherwise.

If you intend to use the QIMSI mouse driver, please make sure that other mouse interfaces are either detached or disabled in their respective driver software. For example, if you have QIMI installed, please configure SMSQ/E to ignore QIMI.

Keyboard

Currently, only a drivers for SMSQ/E and Minerva are available. SMSQ/E version 3.39 and later has inbuilt driver support. After booting your QL you boot into SMSQ/E, e.g. by **LRESPR WIN1_SMSQE.**

Once SMSQ/E is booted you can activate the PS/2 keyboard by using the command “**KBD_PS2**”. You can change the keyboard language by “**KBD_TABLE_PS2** x”, where x is your country code (e.g. UK = 44, Germany = 49). When the PS2 keuboard is active, the QL keyboard no longer works, and vice-versa. You can switch back to the QL keyboard by “**KBD_QL**”. Only the Minerva ROM supports auto start, so this is required to fully omit the QL keyboard. Also Gold Card or Super Gold Card is mandatory for SMSQ/E.

For the Minerva operating system, a driver is provided at https://github.com/janbredenbeek/QIMSI-KBD, where you can also find more information about using QIMSI as keyboard interface. The Minerva driver also supports the **KBD_PS2**, **KBD_QL** and **KBD_TABLE_PS2** commands.

Register Map

In addition to the SD card interface registers, which are fully compatible to the internal QL-SD device and therefore not described here, QIMSI provides the following registers:

Address(hex)	**Address(decimal)**	**Name**
FED0	65232	**QL_SND_STATUS**
FED1	65233	**QL_SND_WRITE**
FED2	65234	**QL_LINK_TXDATA0**
FED3	65235	**QL_LINK_TXDATA1**
FED4	65236	**QL_LINK_RXDATA**
FED5	65237	**QL_LINK_STATUS**
FED6	65238	**QL_LED_OFF**
FED7	65239	**QL_LED_ON**
FED8	65240	**QL_KEYOUT_DATA0**
FED9	65241	**QL_KEYOUT_DATA1**
FEDA	65242	**QL_KEY_CODE**
FEDB	65243	**QL_KEY_UNLOCK**
FEDC	65244	**QL_KEY_STATUS**
FEDD	65245	**QL_MOUSE_CODE**
FEDE	65246	**QL_MOUSE_UNLOCK**
FEDF	65247	**QL_MOUSE_STATUS**
FF00...FFFF	65280...65535	**QL_SND_DATA***

* This area is shared with the SD card interface and requires special care not to interfere with the SD card operation.

All QIMSI registers can only be accessed with read operations, as the QL ROM port does not support writing. However, some of the QIMSI registers transfer data from the QL to QIMSI by reading a specific address. Specific documentation of register operation is beyond the scope of this document.

Chapter 4: MiniQ68

The MiniQ68 inside QIMSI has a 32 KB ROM area at the start of the address range, and a 12 KB RAM area at $19000. Both areas have a double usage. At power-up, they are first used by the internal hardware, containing a small firmware, mainly initializing the PS/2 mouse. Presence of a PS/2 mouse is checked and the result displayed on the green LED. This firmware can then, as a second step, load a binary from SDHC card into an emulated ROM area at the same location.

The emulated ROM behaves like a physical ROM, and is booted by an actual hardware reset. It could execute non QL-specific 68000 code.

As peripherals, the MiniQ68 can access the serial port, the LED, the I/O port and the QL-MiniQ68 communication link. Initially. it can can also access a microSD card, keyboard and mouse, but only as long as they were not accessed from the QL side before. Any access from QL side will put the microSD card, mouse and keyboard interfaces into an inactive state for the MiniQ68.

Booting from ROM image

If a file named „**Q68_ROM.SYS**“ is present on the SDHC card, it will be loaded at address $0 into the MiniQ68 ROM emulation area. The maximum allowed ROM length is 32 KB.

After the ROM image was loaded, the CPU will automatically be restarted by a 68000 hardware reset. Therefore, like a physical ROM, the image file must contain the 68000 vector table, especially the initial supervisor stackpointer at address 0 and the initial program sounter at address 4.

Booting from serial port

If no ROM image is found, or no valid SDHC card is present, QIMSI will attempt to boot from serial port. A fixed baudrate of 115200, no parity, no hardware handshake is expected.

The remote station must first send a header of 4 bytes to the MiniQ68 which contains the length of the following data stream as a big endian 32 bit unsigned integer. Then it sends the data. If the MiniQ68 receives the data correctly, it will respond with a byte containing the ASCII code 'Q', otherwise the ASCII code 'E' for error. During the transfer, QIMSI’s green LED will blink.

The serial data stream will be loaded at address $0 to QIMSI ROM area and started by a hardware reset like a ROM image from SDHC card.

Using the QL-MiniQ68 Link

On the QL side, the QL-MiniQ68 link is accessed by the follwing registers:

$FED2 65234 QL_LINK_TXDATA0
$FED3 65235 QL_LINK_TXDATA1
$FED4 65236 QL_LINK_RXDATA
$FED5 65237 QL_LINK_STATUS

The status register QL_LINK_STATUS is compatible to the Q68 UART:

Bit 0: Tx Empty (Still room inside send FIFO)
Bit 1: Rx Empty (Receive FIFO is empty)
Bit 2: 0
Bit 3: Rx Overrun (Receive FIFO overrun)
Bit 4..7: 0

The read-only registers TXDATA0 and TXDATA1 are used to write a byte into the send FIFO, in a bit-wise fashion. This mechanism is used, because the QL ROM area does not allow a direct write. Reading TXDATA0 writes a bit=0 and TXDATA1 writes a bit=1. The least significant bit comes first. After 8 read accesses, the byte is complete and will be written into the send FIFO. (As a side effect, reading QL_LINK_STATUS will reset the internal bit counter of this mechanism, so an accidental access to QL_LINK_TXDATA0 or QL_LINK_TXDATA1 can not lead to a permanent, unintended bit shift. Normally, it is not required to read the status register to reset the internal bit counter. It is automatically reset after 8 consecutive read accesses.)

The receive data register QL_LINK_RXDATA is compatibel to the Q68 UART and reads one byte from the receive FIFO.

On the MiniQ68 side, the QL-MiniQ68 link is accessed by the follwing registers:
LINK_TXDATA $1C260
LINK_RXDATA $1C264
LINK_STATUS $1C268

These are also compatible to the Q68 UART. The only difference to the QL side is, that the send FIFO data can be written directly by a byte wide write access to LINK_TXDATA.

MiniQ68 Memory map

Below you can see the QIMSI internal memory map as an overview of the physical locations. These are **not** accessible from the QL side. More detailed documentation for developers is also available on request, but is not included in this User's Manual.

Address(hex)	**QIMSI**
0000 0000 0000 BFFF	**ROM 32K** **(Normal operation: ROM image from SDHC card or SER.** **Powerup: 4K Flash)**
0001 8000 0001 8FFF	**Internal I/O 4K** **(QL style registers)**
0001 9000 0001 BFFF	**RAM 12K** **(Powerup: 12K Flash)**
0001 C000 0001 CFFF	**Internal I/O 4K** **(LED, PS/2, I²C, SER, SDHC, Sound)**
FF00 0000 FFFF FFFF	**Internal I/O 16384K**

Chapter 5: Copyright and disclaimer

The QIMSI hardware design, programmable logic, bootloader firmware and mouse driver software are Copyright Peter Graf. The programmable logic also contains the TG68K.C core, which is Copyright Tobias Gubener under the terms of the GNU Lesser General Public License as published by the Free Software Foundation.

The SD card driver provided by the QIMSI ROM is copyright Tony Tebby, Wolfgang Lenerz and Marcel Kilgus. It is free software under the terms of the SMSQ/E license and the source code is part of the SMSQ/E distribution. The definitive version of the licence is contained in the SMSQ/E source code, provided at the official SMSQ/E website www.wlenerz.com/smsqe

Example music is from Ronald Kah, website ronaldkah.de

QIMSI is a hobby project intended for retro-computing and fun purposes, not to process important data. It is distributed in the hope that it will be useful, but without any warranty of merchantability or fitness for any particular purpose. No responsibility is accepted for the loss of data or consequent damage of any kind resulting from the use of this design, hardware and/or software.

All specifications are subject to change without notice. No responsibility is assumed for inaccuracies or errors.