DIY Constructors Manual

*The **QLAN to USB Bridge** (QLUB) **Adapter** is an open-source solution to allow the connection and communication between real Sinclair QL hardware and a QL-Emulator running on a host PC using Sinclair's QLAN/QL-Net protocol and cabling, with support for the TK2 QLAN extensions, including FSERVE.*

Several options exist today to allow transfer of programs and data between emulated and QL hardware, including SERNET and removeable SD-card media, however these require either additional hardware or software at the QL end and thus limit their accessibility for 'unexpanded QL' users. QL-Net on the other hand is built-in to all QLs and is thus compatible with even a minimal QL configuration.

The QLUB Adapter is composed of both hardware and software components – the hardware based on a microcontroller development board (Teensy ++2.0) with custom firmware and some additional components, as well as software written to run within a QL emulator on the host PC, interfacing between QDOS/SMSQE client applications or device-drivers and the microcontroller which is connected to the host PC via an available USB 2.0 port.

The adapter requires a few additional components, readily available and simple enough for most enthusiasts to assemble on a breadboard with minimal soldering required - instructions and a schematic appear later.

A SBASIC program has also been devised to test the basic sending and receiving of files between the QL emulator and a peer QL (or compatible) and is already effective for transferring files between these platforms.

This 'proof of concept' software is being made available to the QL Community at this stage (March 2021), along with the latest microcontroller firmware and these instructions so as to allow user testing and feedback whilst work continues on the software release-candidate that will ultimately be fully integrated in to QDOS/SMSQE running in the emulator.

The QL emulator QPC2 has been used throughout the development cycle, but the final solution should be compatible with any QL emulator that exposes the host PC's COM/serial ports and a platform for which a suitable USB Virtual COM port driver exists for the Atmel microcontroller.

Members of the QL Community have since tested the solution using the QEmuLator and uQLx emulators, connecting to both unexpanded QLs as well as those fitted with Gold and SuperGold Cards running various versions of QDOS and Minerva – with or without TK2.

As the ZX Spectrum with the Interface-1 also provides compatible network hardware and software, the QLUB will work equally well connected to a Spectrum (or indeed, a Spectrum Next with Int-1). The same applies to any QL-like solution that includes the QLAN NET port hardware such as the Q68, QXL and Aurora, among others.

As of March 2021, the project continues a-pace, with the microcontroller firmware working well, most of the software API interface and data-structure design complete and the realisation of the 'message-queue' server task that interfaces the adapter to QDOS now well advanced.

MARTYN HILL
March 2021

DIY Constructors Manual

Goals and Motivation

The original goal of this project was to ease the process of developing QL software on a PC within a QL emulator - to then be transferred to a target QL without recourse to swapping removeable SD media back and forth which, at the time of inception at least, was particularly cumbersome.

As well as removable media, SERNET was also tried but proved unsuccessful on the author's hardware and was not pursued further; in any case, I was not enamored with the bulky serial-cables needed to connect hosts, versus the simplicity of the original QL-Net cabling.

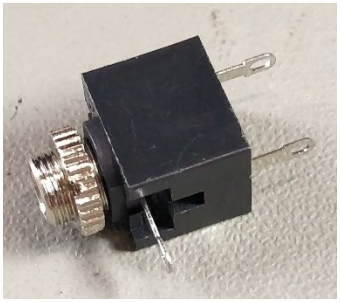
Recent, exciting developments adding Ethernet and more modern IP-based comms to the QL and emulators offer a future-proofed approach to the problem, but the 'Sinclair Network Standard' and the general availability of the QL Network remain a useful and readily accessible solution to linking QLs – and now QL emulators.

It has always been the intention to make this solution accessible and available to all QL users under an open-source arrangement. I would be equally happy to work with any of the hardware developers in the community to bring a ready-made solution to market - as long as the original source code and designs remain open to all.

Building the QLUB Adapter

To build and program the adapter you will need:

1. The **Teensy ++2.0 development board**, ideally with header-pins already fitted (c £20, local distributors are listed on the PJRC website: https://www.pjrc.com/store/teensypp.html)
2. A **mini-USB to USB-A cable** to connect the adapter to the host PC (during both programming **and** 'in action.')
3. The **Arduino IDE** (free to download: https://www.arduino.cc/en/software)
4. The **Teensy add-on libraries** and **Virtual Com Port driver** for your PC platform (from https://www.pjrc.com/teensy/td_download.html)
5. The latest **microcontroller firmware** - currently, **QLAN-USB_v22a.ino** (or the raw Intel HEX format equivalent – both available from the author via the Sinclair QL Forum: http://www.qlforum.co.uk/)
6. A **breadboard** with at least 30 rows of contacts (generally available)
7. Various **electronics components** and some wiring/soldering equipment (all generally available):
    - 2x 330ohm, 1x 47ohm, 1x 1Kohm, 1x 3K9ohm, 1x 10Kohm resistors
    - 1x PNP bi-polar high-speed switching transistor, ZTX-510 or equivalent
    - 2x mono-switched chassis Jack sockets (e.g. RS Stock# 106-874)

Switched chassis Jack socket example *(note: **3** connections)*	
	

NB – Stereo Jack sockets also have 3 connections, ***but are not equivalent, nor suitable*** for this project. Make sure you purchase the **mono-switched-type** explicitly.

DIY Constructors Manual

8. Ideally, a suitable **project-box** (c 110mm x 60mm, 20-25mm deep)

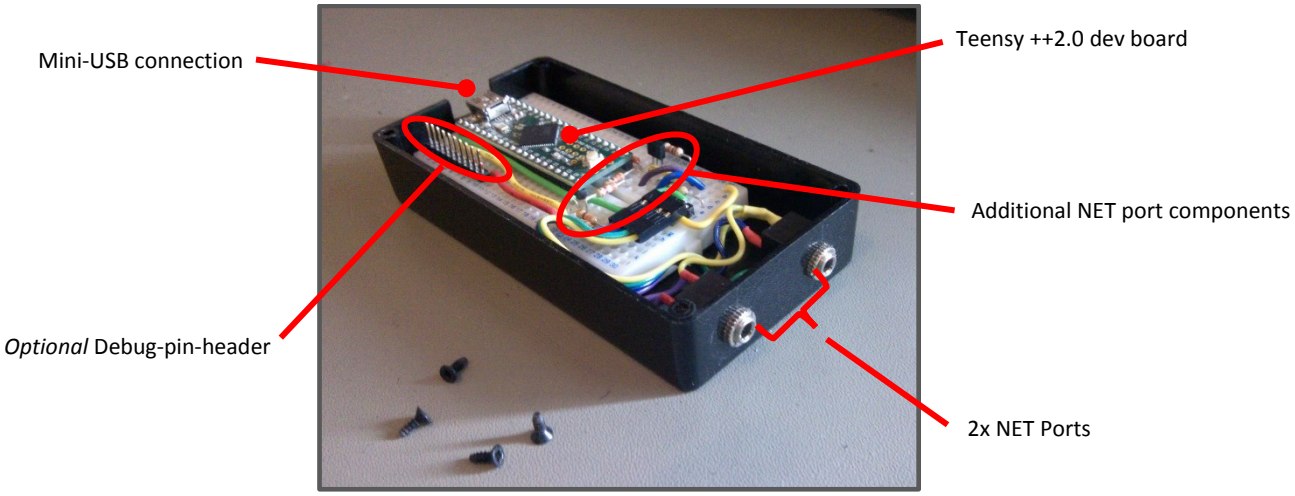


Figure 1: QLUB Adapter installed in project box

9. Don't forget one or more **mono (or stereo) cables**, terminated with 3.5mm TS/TRS Jack plugs at each end to connect the QLUB Adapter to your QL/Spectrum.

NET Port Connection Plugs	
	
3.5mm 'TS' Jack-plug (mono)	**3.5mm 'TRS' Jack-plug (stereo)**

The schematic for the additional hardware components will be familiar to anyone who has studied the QL or Spectrum Interface-1 Technical Service Manuals, as it is a like-for-like match, thus:

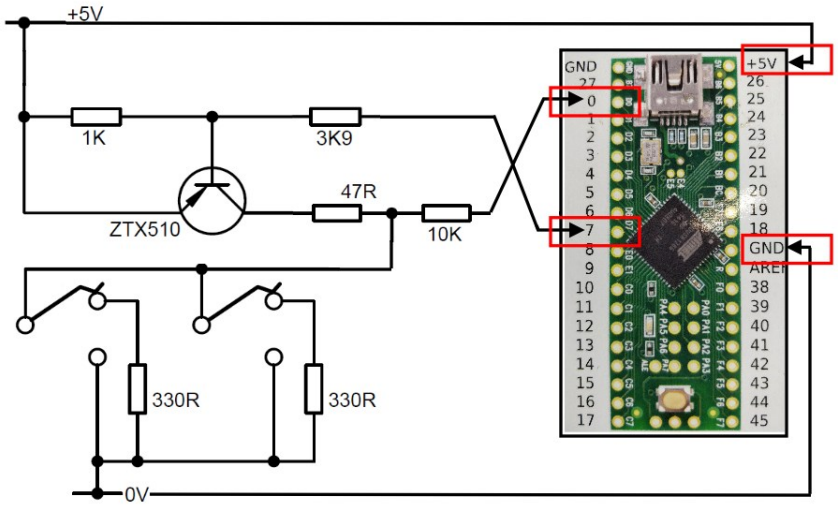


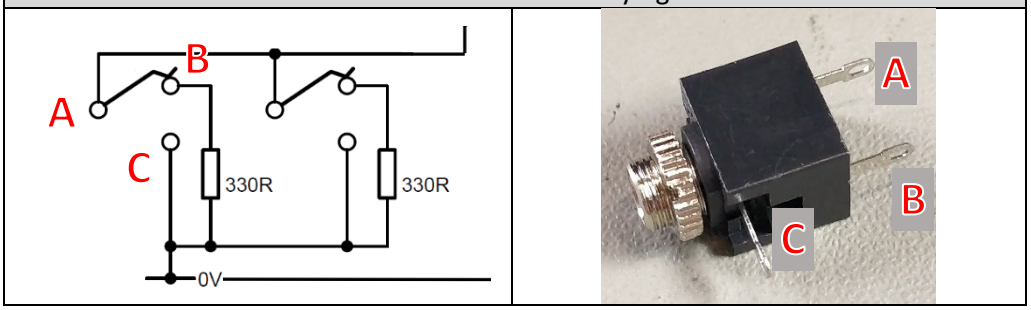
Figure 2: QLUB Adapter Wiring Schematic
*(copied in part from QL Schematic by **D Westbury** – see http://www.dilwyn.me.uk/docs/hardware/QLissue5.pdf)*

DIY Constructors Manual

More on the Jack sockets…

Knowing how to connect the three solder-tags of the switched-Jack socket can prove perplexing – the following diagram should help, but bear in mind that there are slightly different designs of switched sockets on the market, so take care to validate your work against the model of socket you purchase…

Switched Chassis Jack Socket – Identifying the connections



Connection points **A and C** are common to both sockets in a two-socket design – whereas connection **B** on each socket attaches to its own pull-down/termination resistor.

As alternatives to the fully terminated design, there are a couple of alternative, slightly simpler options which I outline here – the preferred option, however remains as above:

***Option B**: It is perfectly possible to build **only one socket** in your design as long as the QLUB Adapter will always act as the terminating device **at either end** of your network. In this case you can use a single **non-switched** type socket (only 2 connections) and will need to connect the pull-down/termination resistor **in parallel across** the two connection points. The termination is still required, but at only half the resistance value (i.e. 165ohm.) Simply solder 2x 330ohm resistors in parallel across the socket connections to achieve the desired termination.*

***Option C:** Even more simply, **if** the QLUB Adapter is going to sit **between** two other stations on the network, then you can use 2x **non-switched** type Jack sockets and **dispense** with the termination resistors altogether – the two other end stations then take-on the role of terminating the network.*

Optional 'Debug' signal wiring…

The firmware is coded to signal what it's doing throughout the transmission of each NET packet plus other diagnostic information sufficient to troubleshoot timing problems and of great value if you don't seem to be getting reliable – or any - communications between the QLUB and a QL or Spectrum machine.

Check-out the section below **Diagnosing Issues** for further details on wiring-up and use of the debug pins – you'd be well advised to attach these pins when first assembling the adapter…

Programming the Teensy Dev Board

There is ample description and instruction on use of the Arduino IDE for programming Atmel microcontroller boards like the Teensy, plus additional information on the PJRC website specific to the Teensy range of development boards to get you going.

Please refer to these resources before posting questions.

Once programmed - and after every subsequent restart/reconnection to the host PC - the QLUB adapter will automatically start listening for suitably formatted command-messages from the host

PC via the USB port. Thus, the USB connection remains an ***active part*** of the QLUB design and **not** just used during programming.

When new firmware is released, it is a simple matter of downloading and opening the latest **.INO** source-code file in the Arduino IDE, recompiling and re-flashing the device.

NB. *Early versions of the QLUB firmware (up to v1.7) were coded for and reliant upon a now deprecated version of the TeensyDuino libraries (v1.35) – itself compatible only with an older Arduino IDE (v1.18.0).*

QLUB firmware v2.2a has instead been built using the latest versions of the IDE **v1.18.13** and TeensyDuino **v1.53** – the versions currently available on the respective websites. It is these versions you should download and use - or else upgrade to – when using the latest QLUB firmware.

Two entirely equivalent versions of the firmware itself are available – in the source **.INO** or the pre-compiled **.HEX** formats. Feel free to peruse the source-file to understand the code and compile/flash using this, or alternatively, just flash the pre-compiled file directly - which places no dependency on the version of IDE or TeensyDuino libraries installed. They both produce exactly the same result.

To flash successfully, **no other application** can be holding the virtual COM Port open, so you must close your QL Emulator to release the SERial port when ready to re-flash the microcontroller.

File-Transfer Software – 'SendFileMQ'

The QDOS software currently available (**SendFileMQv22_bas**) is written in SBASIC and allows for simple file-transfer back/forth between emulator and the QL or Spectrum. Like the microcontroller firmware, you can request this from the author via the Sinclair QL Forum.

The final software package will support the full TK2 NET extensions, including FSERVE and will be fully integrated into QDOS/SMSQE as a revised NET driver and MQ Server task.

Once you have validated which SERial port the USB Virtual COM port appears-as within the emulator when plugged-in, simply edit the variable `virtualCOMPort%` near the top of the listing to match and then RUN the program.

Please read the initial `REMarks` in the listing for ideas on what to enter at the QL/Spectrum end of the link and other configuration options you might need to adjust to run successfully in your QL emulator. Those remarks form an integral part of these instructions…

QDOS and ZX File headers are both managed correctly and automatically by the provided transfer routines. Any ZX files available to the QL Emulator are presumed to include their own 9-byte file-header embedded at the beginning of each file.

Furthermore, the file-transfer program and QLUB firmware have both been extended in v2.2a to allow real-time reconfiguration of the all-important NET Timing Constants that underpin successful comms between compatible machines – without the need to recompile the source as was required in earlier versions. It has been observed that the timing values needed for reliable QL comms need to be very slightly adjusted to better support the ZX Spectrum/Interface-1 and the two distinct sets of timing values have been included in `DATA` statements near the bottom of the SendFileMQ listing, selected automatically depending upon your choice of 'Peer type' when running the program.

For a full explanation of the use and workings of the QL LAN in general, please refer to the **Networking the QL** article, also by this author and available via the Sinclair QL Forum.

DIY Constructors Manual

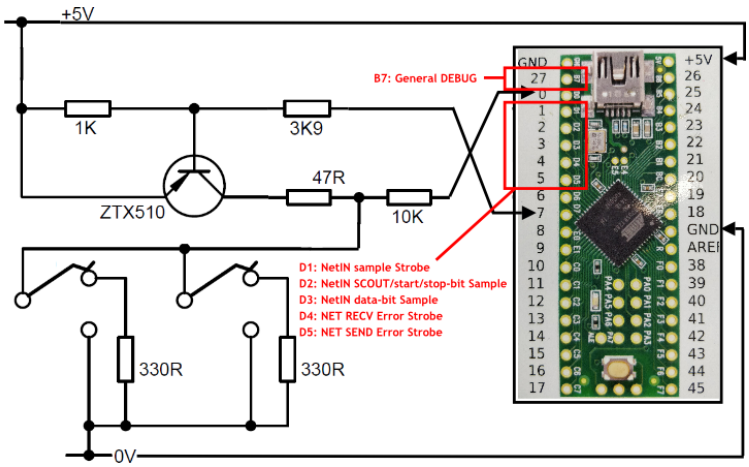
Diagnosing Connection Issues

If you can't seem to get the QLUB to speak reliably with your QL or Spectrum even after reading-up the available resources (including the section below **Things to consider and further reading**), you may be hitting some NET timing issue – typically quite difficult to diagnose and a common reason for many users giving-up altogether with networking their QLs – until the QLUB, that is!

The debug signals provided by the QLUB can provide real-time information when read with a digital logic analyser such as the cheap but effective 'USB 8CH 24MHz 8 Channel Logic Analyser' based on the Sigrok chipset and supported by the open-source 'PulseView' application.

The debug signals (all 5V logic) are made available on the PORT pins `D0` (NET-In), `D1-D5`, `D7` (NET-Out) and `B7`. If you look-back at the photo in Figure-1 above, you'll notice a line of 10 pin-headers installed that allow a digital Logic Analyser to be connected to monitor the debug signals and help understand the problem.

The development of the QLUB and much of my research in to the QL Network would have proven almost impossible without the use of the QLUB debug pins!



Debug Pin Functions

Name	Function	Port	Cable Colour
GND		GND	Black
DEBUG_NET6	General DEBUG	PORTB-7	White
NET_IN	NetIN	PORTD-0	Grey
DEBUG_NET1	NetIN sample Strobe	PORTD-1	Purple
DEBUG_NET2	NetIN SCOUT/start/stop-bit Sample	PORTD-2	Blue
DEBUG_NET3	NetIN data-bit Sample	PORTD-3	Green
DEBUG_NET4	NET RECV Error Strobe	PORTD-4	Yellow
	5us: RECV_NO_GAP		
	15us: RECV_NO_SCOUT		
	25us: RECV_HDR_FRAME_ERROR		
	35us: RECV_HDR_CHKSUM_ERROR		
	45us: RECV_HDR_NOT_FOR_ME		
	55us: RECV_HDR_PHANTOM_BLOCK		
	65us: RECV_HDR_BLK_MISMATCH		
	75us: RECV_DATA_FRAME_ERROR		
	85us: RECV_DATA_CHKSUM_ERROR		
	95us: RECV_DATA_ACK_NO_GAP		
	105us: RECV_OTHER_ERROR		
DEBUG_NET5	NET SEND Error Strobe	PORTD-5	Orange
	5us: SEND_NO_GAP		
	15us: SEND_CONTENTION		
	25us: SEND_NO_HACK		
	35us: SEND_NO_DACK		
(Unused)		PORTD-6	Red
NET_OUT	NetOUT	PORTD-7	Brown

Things to consider and further reading…

- The initial installation of the microcontroller development board itself can prove problematic and sometime takes a few attempts to install the Virtual COM Port driver (in Windows) before you can successfully program the device – just persevere.
- The dev board can also appear to hang from time to time when running the provided QLUB firmware and may occasionally need to be unplugged/re-connected to get correctly re-enumerated by the host OS USB stack before the QL emulator can access the SERial/COM port successfully.
- Use of the QL network has seen mixed success in practice, due to a number of software-timing and hardware issues over time. The QLUB Adapter can't fix those directly, but otherwise follows the Sinclair defined timing even more closely than native QL hardware. Modified timing-constants in the TK2 code have been seen to address most issues and details can be provided on request.
- Likewise, interconnecting QLs and Spectrums is often problematic and the Shadow ROM code in the Interface-1 has a couple of previously undiscovered bugs that can make use of the network unreliable with unexpanded QLs. The author can make modified ROM software available for the Interface-1 that overcomes these issues.
- Otherwise, most issues you are likely to face using the QLUB Adapter are common to native QL/Spectrum networking and lots of advice is provided in the **Networking the QL** article previously mentioned. *Have you read it yet?*
- You'll find further helpful information on getting the QLUB up and running shared by other QL Community members in the QL Forum thread here: https://qlforum.co.uk/viewtopic.php?f=2&t=3590

I'd like to extend my thanks to those QL users who have already provided valuable feedback and testing of the early-release QLUB code, especially *maskenlos, m68008, ġbejniet* and *mk79.*

Enjoy - and please do share your findings and questions via the QL Forum!