superHermes

a replacement for the 8049 and Hermes co-processors

A processor is the controller of any computer system. The QL's 68008 is the Central Processing Unit (CPU) and controls all aspects of the QL. However, there is ANOTHER processor in the QL! This is known as the IPC (Intelligent Peripheral Controller) and is an 8049, a member of Intel's MCS-48 family of single component 8-bit microcomputers.

The IPC handles the detail for sound, serial input and keyboard input. Having a second processor doing these jobs is a good idea, as it frees the 68008 to perform much more interesting tasks. Unfortunately the internal code driving the 8049 was badly written and suffered from a number of problems. HERMES was developed to solve the major problems with the QL co-processor, and even give some bonuses!

OUR NEW BABY

As you all I am sure know, progress in the computer field is staggering year by year. Since the QL was first launched over 10 years ago, progress in co-processors and microcontrollers has continued apace.

We were looking at the GI-Microchip PIC 17C42 to improve on Hermes and add an IBM keyboard interface. However on thinking about the design, we were amazed at the capability of this processor. It has many more available legs, has more memory and is very much faster than the 8049 (and Hermes 8749) with an improved faster instruction set. It has a RISC (Reduced Instruction Set Code) and each instruction is two bytes, meaning most operations take one machine cycle. It also uses a modified Harvard architecture, where program and data are handled from separate memories. We even thought of a hardware method of reducing 16 lines to 4, using multiplexers, to allow extra signal lines (eg DSR3/ KLOCK/TURBO to be added). Night by Night we were finding other things to add. Many of these features were added as a result of comments on the Bulletin Boards, so it is fitting that the first production board was fitted to QBBS (01344-890987), and is still there now.

superHermes will help high speed BBS access and use of high speed fax modems.

The most surprising aspect was that all features are on a board about twice the size of the original 8049, and has no less than 37 interface pins connected excluding those needed to connect to the QL. Individually soldered pins are used to connect to the 8049 socket, meaning replacement of broken pins can be done easily. We will do this for postage costs only at any time.

SERIAL PORT 3

The PIC has hardware support for serial input/output, and we have chosen the main oscillator carefully to give 100% accuracy for all standard baud rates. ***superHermes*** offers an interrupt driven full feature two way serial port at speeds

up to 57600bps, thus supporting V34 modems and V42bis compression speeds. When the replacement 'ultra' Gold Card arrives, using the coldfire processor, speeds of 115200bps will probably be possible.

IBM KEYBOARD INTERFACE

superHermes has an IBM AT keyboard interface. At bootup the normal start keys (F1/F2) will be available, but IPCEXTCC_BIN needs to be loaded (RESPR) in your normal boot file for a fully working IBM keyboard (CC is country code). You will be able to boot from the supplied utilities disk. Many keyboard types are supported (see updates_doc on utilities disk for full information).

SERIAL MOUSE/RTTY

There are three low speed RS232 INPUT ONLY ports - SER4/5/6 (1200bps down to 28.1bps).

Typical uses:

- RTTY (Radio teletype)
- Serial mouse. Both Microsoft and Mouse Systems mice are supported in conjunction with the pointer environment.

KEYLOCK

There is a connector for the IBM cased QL keylock. When two pins are connected (usually via the keylock), code in the pic will lock IBM keyboard, QL keyboard, and ***superHermes*** mouse.

TURBO

A ***superHermes*** command in association with a 'turbo' connector and switch can slow the QL down. See 'fitting instructions for more details of how to connect the LED.

LED

An LED (Light emitting diode) connector is provided for CAPSLOCK/SCROLL LOCK. This will not be needed when used with an IBM keyboard, as ***superHermes*** will control the keyboard LEDs, unless you want an extra LED in ***superHermes*** fitted in a QL case.

SPARE SENSE/CONTROL LINES

A connector provides 3 logic level input/output lines, along with +5v and GND.

DEFAULTS

The EEPROM can be used for defaults (see Appendix 6 for an example).

The above are all brand new features of ***superHermes***.

COMMON FEATURES WITH HERMES

Serial Input (SER1/2)

superHermes is not involved with serial OUTPUT (SER1 and SER2), hence this stays as it was (supporting 19200). In particular, note that the QL will always send two stop bits at all baud rates.

All QL roms will now support baud rates up to 19200 input with one stop bit, and give full data throughput, subject only to speed of the QL main board and storage devices. At high data rates, RAMdisk should be used, and avoid using intensive jobs (disk access, networking etc).

Separate input baud rates for ser1 and ser2, and different from output can be set. In fact with Minerva 1.97 and greater, different output baud rates can also be set.

IPC control

In order to control functions of the IPC directly from the keyboard, it is necessary to come up with some combination of keypresses which doesn't have any prior meaning. The original 8049 only required one such function, reset, for which it used CTRL ALT 7. In fact, CTRL ALT 7, according to the Users Guide, should just send the "FF" code for the ALT key, followed by CHR$(151).

superHermes (and Hermes) uses the combination of CTRL ALT ESC, plus another key, to effect IPC controls. This will usually result in some spurious characters having to be typed, but they can be kept safe in almost any circumstance, as ***superHermes*** stops recognising characters as soon as two are pressed down, except for its special combinations. E.g. you could press both left and right cursor keys down first, then, while they are still held, get ESC and then the required extra key down, and only then release the cursor keys and press CTRL and ALT.

Currently, just two special combinations are in use:

Key Click

This can be turned off/on by pressing CTRL ALT ESC 4/5 respectively. It can also be turned off or on by "IPCEXT 6" and "IPCEXT 7" respectively. (Applies to Hermes also)

Note that it doesn't know about auto-repeat, which is controlled by the software in the 68008, so a held key only clicks once.

Reset/INT7

This can be invoked with CTRL ALT ESC 7.

CTRL ALT 7 used to "freeze" a QL. It resets the code in the 8049 and, if there was no code added to the QL to handle the situation, or indeed, sometimes even when such code was present, the effect was disastrous: fairly similar to hitting the QL with a hammer... repeatedly.

On ***superHermes***, the ESC key must also be pressed, making everything a little cleaner and safer. A few spurious characters will crop up, but at the time of requesting a reset, this hardly matters.

As ***superHermes*** uses an IC to drive the interface to the QL keyboard, and this should improve reliability for some types of added keyboard using long cables connected to the membrane sockets.

USING superHermes SERIAL 3

The new fast serial port on ***superHermes*** (serial 3) is an industry standard port with a 25D plug. This will accept any standard 25D socket. These are are available off the shelf very cheaply from practically everyone! For instance a standard 25D RS 232 extender (male » female) will connect any modem with a 25D connector to ***superHermes*** serial 3 lead.

The simplest way to set up for communications software is to use the ***superHermes*** OPEN from superBasic to redirect ser2 to ser3. If the program makes no provision for our extra parameters for serial 3 (as a single parameter string) then this is the only method available.

Say you want ser3 used at 57600bps instead of ser2, with a 6k input and 2k output buffer, in superBasic the following command will do this:

```
OPEN#3;'ser2\3 b57.6k u6k p u2k':CLOSE#3
```

This will work for any program using ser2 (eg LFAX/QFAX/QEM/QuaLsoft Terminal etc) and give a maximum hard wire link of about 4800cps (QTPI/ supergoldcard).

See next section for the complicated description of the details.

Jonathan Hudson's excellent QTPI comms program (available from all QL based bulletin boards or Qubbesoft) has a field COMDEV in its CONNECTIONS menu. Simply enter the parameter string (data in quotes) in COMDEV. All other comms parameters are then ignored by QTPI.

EXTENSIONS

To make it easy to utilise the new features offered by ***superHermes***, a set of SuperBASIC extensions are supplied. (see "getting started")

These can be loaded into ROM (IPCEXTCC_ROM). If you have a Care ROM cartridge which you want modified, and don't know how to do this, then contact us.

Other than IPCVER$, the extensions will report "not found" if a non ***superHermes*** /Hermes IPC is present or a ***superHermes***-only command is requested on a plain Hermes. The "%" sign in parameters is identifying the value as an integer and does not need to be typed literally.

DEFAULTS

Startup values for IPCSLOW, IPCSTUFF 0 (middle button), IPCENABLE items, and mouse movement downscaling and Numlock can be set in the extensions file using the CONFIG program (supplied).

The supplied extensions default to all IPCENABLE features enabled, NUMLOCK on, Mouse movement factor 2 (equivalent to QIMI), centre hotbutton (chr$(255) of "." - ALT always appended. Use IPCSTUFF direct when ALT not required.

SER3, SER4, SER5 and SER6

These can work in much the same way as SER1 and SER2, except that SER4/5/6 are input only with no handshake. Also the specification of parity, handshaking and protocol is no longer restricted to being in that order, spaces or underscores may be used to increase clarity and the overall functionality is greatly increased. eg:

```
OPEN#3;'ser3_b57.6k 8n1 u6k t 7e2 u128'
```

This opens channel three for i/o to SER3 at 57600 baud. The received data comes via a 6000 byte buffer, but only 128 bytes of data is buffered for output. Eight bit data is received and stored, but transmitted data will have its eight bit replaced by a even parity bit and will further be given an extra stop bit.

```
OPEN#4;'ser55n b45.45'
```

This opens channel four for input from SER5 at 45.56962 baud, which is only a quarter of a percent off the exact figure. Five bit data is expected with no parity.

The parity codes include "N" for "none" as well as the standard "E", "O", "M", "S" for "even", "odd", "mark" and "space". They may be prefixed by a digit, one of "5", "6", "7" or "8", to specify the data word length (i.e. not including the parity bit, if selected). They may be suffixed with a digit, either "1" or "2", to specify the number of stop bits. If no parity code is given, the default is "8N2". If the parity code is given without a word length, it will default to eight for parity code "N" or seven for the rest. The default number of stop bits is always two. The number of stop bits is ignored for all but SER3 output, as all receive ports function with a single stop bit, and further, even then, only some combinations are significant (see below).

The baud rate may be specified by a value preceded with "B", but will default to 1200 baud.

The buffer length to be used may be specified by a value preceeded with "U", but will default to 80 bytes.

The behaviour of SER3 may be selected differently on input versus output by giving the transmit characteristics after a "T", but note that the baud rate cannot differ. Also note that in this case, unspecified transmit characteristics will default to the same as the receive characteristics.

Due to the nature of the hardware being used for SER3, only a limited selection of word length, parity and stop bits is actually available. The hardware can only ever handle 10 or 11 bit data, when the start and stop bit are included. Hence SER3 receive is restricted to a word length of eight when no parity is specified, and may only be seven or eight when parity is present. For transmit, any combination is accepted, but the shorter combinations of word length and parity will result in extra stop bits, and only one stop bit will be sent when eight bit data with parity is selected. I.e. where "P" means a parity code other than "N", the following are precisely supported:

SER3 receive: 7P1, 8N1 and 8P1.
SER3 transmit: 6P2, 7P1, 7P2, 8N1, 8N2 and 8P2.

Both the baud rate and the buffer length values may include a decimal point and may be followed with a "K" to indicate that the value is to be multiplied by factor of 1000.

The baud rate may be set with a range of values, with 1200 being one that happens to be available on all the new serial ports. It may be given with decimal places, e.g. "B45.45". For the low speed devices, the baud rate should be one which is a result of dividing 7200 by a number from six to 256, so the highest rate they can have specified is 1200 baud and the lowest is 28.1 baud. They can actually be told to do 2400, 1800 and 1440, but these will not be reliable. The same sort of baud rates, but multiplied by a factor of 32, are available on SER3. It its case, the highest reliable rate is 57.6K and the lowest is 900 baud. Where an exact figure is not given, the closest possible value will take effect. In summary, we accept:

SER<port><receive>[T<transmit>]

where <receive> and <transmit> may include any of:
[<word length>]<parity>[<stop bits>], <handshaking>,
<protocol>, B<baud rate> or U<use buffer length>
and <port> : 3, 4, 5 or 6
<word length> : 5, 6, 7 or 8
<parity> : "£", "L", "N", "O", "E", "M" or "S"
<handshaking> : "I" or "H"
<protocol> : "R", "Z" or "C"

<baud rate> and <use buffer length> are <value>s and a <value> is decimal digits, with an optional decimal point, and may be followed with a "K" to multiply it up by a factor of a thousand.

Note: In order to get the full throughput on individual parts of the system, memory organisation is extremely flexible. However, for instance, if ser3 output is not taken away then memory buffers available will reduce, and therefore efficiency will be reduced. You should avoid having input or output ports open

and receiving or sending data, where the data is not being cleared from the buffers.

In addition to all of the above, there is a facility to redirect serial ports. In reality, the open name may start with “SER” and a optional digit from one to eight. It may then include all the above, plus one may indicate the specific port to be used by prefixing it with a backslash “\”) or “D”. Also, the receive and transmit specifications may be separated by “P” or “X” as well as “T”.

An additional parity code of “£” or “L” may be used to strip parity from received data without checking it. On transmit it is the same as specifying “M”.

An additional protocol code of “A” may be specified, which will transpose carriage return and line feed characters, as for “C”, but does not impose the “CTRL/Z” handling that that protocol includes.

The digit after “SER” is the “unit” and defaults to one.
The digit after a backslash is the “port”, from zero to six.

The separator “P”, “T”, “X” or none is used to set up defaults or overrides.

The rest of the name consists of zero to six “elements” of receive specification, optionally followed by the “separator” (“P”, “X” or “T”) and zero to six “elements” of the transmit specification.

The elements are the queue length (“U” followed by the value), the baud rate (“b” followed by value), the handshake control (“H” or “I”), the port (“\” followed by “0” to “6”), the protocol (“A”, “R”, “C” or “Z”) and the word definition (optional word length 5..8, parity “£LNEOMS” and optional stop bits “1” or “2”)

If the “port” is specified as zero (“\0”), it is a request that this driver should just pass the call on to some other driver.

Specifying the port as one (default) or two will cause this driver to reconstruct a “proper” name for the standard serial driver to function with. Naturally enough, this will give an error if the “£” or “L” parity option or the “A” protocol is being attempted.

When the port is given as 3..6, that port in ***superHermes*** is selected.

When a “P” is given, any other elements supplied become permanent for this unit and will force themselves into any subsequent open call. The receive and transmit capabilities are also latched according to what is shown.

With a “T”, any supplied elements are made “temporary”, so that this call, and subsequent calls will get them as defaults, but may supply them if they wish.

With an “X”, or just a receive specification and no separator, supplied elements are used, except when the “override” flag for the element has been set. Any permanent elements and elements that are not supplied will come from the current defaults.

The initial defaults are set up to the standard sort of thing, and could be reinstated by the following sequence:

```
OPEN#3;“serupu”:OPEN#3;“seru80b9600h\1r8n2tu80b9600h\1r8n2”
OPEN#3;“ser2upu”:OPEN#3;“ser2u80b9600h\2r8n2tu80b9600h\2r8n2”
OPEN#3;“ser3upu”:OPEN#3;“ser3u80b9600h\3r8n2tu80b9600h\3r8n2”
OPEN#3;“ser4up”:OPEN#3;“ser4u80b1200h\4r8n2t”
OPEN#3;“ser5up”:OPEN#3;“ser5u80b1200h\5r8n2t”
OPEN#3;“ser6up”:OPEN#3;“ser6u80b1200h\6r8n2t”:CLOSE#3
```

At present, not all the possibilities implied by the above are implemented.

Only the port in the receive specification is used, although splitting the receive and transmit sides of port 3 will eventually be allowed, except that the baud rates would have to match, as the hardware restricts it.

All transmit specifications other than port 3 are ignored.

Only the handshake, parity and protocol specification can be passed on to the standard port one or two driver. The baud rate is currently ignored, but it may be implemented later.

Even so, as an example, the functionality is such that one can very easily force a program that thinks it is talking to the standard “ser2” to actually find itself running at 57.6k baud on sH port 3 with the recommended ram buffers :

```
OPEN#3;“ser2\3 b57.6k u6k p u2k”:CLOSE#3
```

IPCMOUSE [port]

Any of ports 4 to 6 may be selected as the mouse port. A negative parameter turns off the mouse and releases the port (if it was on). The driver must link into the pointer environment. If the pointer environment is not present it returns 'not complete'. eg IPCMOUSE with no parameters will open the default SER6 for mouse input (marked MOUSE on the circuit board).

IPCSLOW [factor%]

Syntax: IPCSLOW factor%

A factor of zero to 7 will change the mouse acceleration. IPCSLOW 2 will be roughly equivalent to the QIMI mouse. A smaller factor gives a faster acceleration. This allows the acceleration of the mouse to be changed independantly of the pointer environment value, thus not affecting the cursor key movement. (default in extensions config block).

IPCSTUFF[button,]“string”

The default (in config block) is for the middle mouse button, or holding both buttons on a two button mouse to act as <ALT> |< . >. This may be used to set a one or two chr string to be sent instead, or to ignore it by giving a null string.

The optional “buttons” may be used to configure what the behaviour is for each of the left or right buttons, which default to producing space and enter . Values are -1 (left), 0 (middle) or 1 (right).

IPCVER$

This is a function to tell you what version of IPC you have. It will return the version number, a dot and the revision number as a text string. For non ***superHermes/Hermes*** IPCs, it will always return "0.0", as they can't be asked what their versions are.

RXBAUD%

This is a function to read and set the ser1 and ser2 serial port input rates.

Syntax: RXBAUD%(parameter%)

parameter% gives the port and the value to send.

The result is the old value, including whether the port is open or not.

The value consists of the following:

bit 7	: Send 0=ser1, 1=ser2. Return 0=open or 1=closed. (code 128)
bit 6	: Set if BAUD command is not to affect this input port. (code 64)
bits 3 to 0	: Standard baud rates (bit 4 ignored at present) codes 0-7 = 19200, 9600, 4800, 2400, 1200, 600, 300, 75 codes 8-15 duplicate the above.

All other bits are reserved and must be zero.

E.g. To pick off ser1 receive to be used for a mouse operating at 1200 baud only, one would do "rx0% = RXBAUD%(64+4)" (provided the current serial mouse driver is available from Albin Hessler). (If this is done before anything else, "rx0%" will be set to 129, showing that "ser1_" was not open (128), it was being affected by the normal "BAUD" (no 64) and its current baud rate was 9600 (1).) If other compiled jobs, etc., change the baud rate, or you do a new "BAUD 19200" say, ser1 input will stay running at 1200 baud. To revert to what was there originally, you could do a "PRINT RXBAUD%(rx0% && 127)", which would print 68 if the port was open, or 196 if it was closed.

IPCEXT

This may be used to set the LED, spare output lines and keyclick.

Syntax: IPCEXT command%

command% may be one of the following:

	superHermes	Hermes
0:	turn on LED	set P23 low (bend out pin 24)
1:	turn off LED	set P23 high
2:	set SPARE 1 low (pin 5)	set P20 low (bend out pin 21)
3:	set SPARE 1 high	set P20 high
4:	set SPARE 2 low (pin 4)	set P26 low
5:	set SPARE 2 high	set P26 low
6:	keyclicks off	keyclicks off
7:	keyclicks on	keyclicks on
8:	set SPARE 3 low (pin 3)	enable clock to T0 (note 2)
9:	set SPARE 3 high (note 1)	
10.	Turn on DTR3	-
11.	Turn off DTR3	-

Note 1. The very first call of MDRS/9 is made internally by the extension code, and turns on ***superHermes***. One CAN turn off ***superHermes*** (see below), although that is probably not a very good idea! If that has been done, the first subsequent IPCEXT 9 will turn it back on again.

Note 2. The clock is a 11/3 MHz output on Hermes T0, until the QL is powered down. A similar frequency (14.7456/4 MHz - pin 0SC2) is available on the ***superHermes*** board, if one really needed it - IC6 pin 13)

IPCSIG%

This function allows the spare input lines to be read.

Syntax: IPCSIG%

The value returned with be 0 to 15, where the value is made up of four bits showing the states of various lines/pins:

	superHermes	Hermes
bit 0 (result%&&1)	DCD	state of P20 (pin 21)
bit 1 (result%&&2)	SP1	state of T1 (pin 39)
bit 2 (result%&&4)	SP2	state of T0 (pin 1)
bit 3 (result%&&8)	SP3	state of P23 (pin 24)

In order to be useful, the outputs should be connected to something!

Note that the state of DCD3 - carrier detect (and WP for Hermes) is also reflected in the system variable SV_WP, which is actually inverted. I.e. a zero input will be seen as $FF and a one input shows as zero.

IPCISET

Syntax: IPCISET mask%

The three spare control/sense lines start off as input signals on power up or reset. They are automatically made outputs when they are set or cleared using the IPCEXT calls. This allows them to be selectively set back to being inputs.

The mask% is made up by adding together whichever of the following are required to be inputs: 2 - SP1, 4 - SP2, 8 - SP3. Only even values should be supplied. It doesn't matter if the line is selected as input more than once. A parameter of zero is accepted, and has absolutely no effect.

IPCMXI%

Syntax: IPCMXI%

This may be used to read ***superHermes*** "MultipleXed Input" signals. The byte returned is zero to 255, and is made up of the following bits:-

Bit	Mask	
0	1	DCD3 (high speed ser3)
1	2	DSR3 (high speed ser3)
2	4	KLK (tower case K/LOCK)
3	8	TRB (tower case TURBO)
4	16	RX4 (low speed ser4)
5	32	RX5 (low speed ser5, RTTY)
6	64	RX6 (low speed ser6, mouse)
7	128	CTS3 (high speed ser3)

IPCDISABLE and IPCENABLE

Syntax: IPCDISABLE feature%

IPCENABLE feature%

The feature% is from zero to seven for IPCDISABLE, but only zero to six for IPCENABLE. IPCENABLE/IPCDISABLE -1 enables/disables NUMLOCK respectively. Defaults in extensions CONFIG block.

superHermes has various modes of operation that may be controlled via these extensions. Some of them are rather unlikely to be useful once the extensions are

loaded, as the drivers will get rather confused! However, there are the following that may be controlled:

Value	
0	reserved
1	reserved
2	keyboard lock processing
3	turbo processing
4	reserved
5	external interrupt
6	IPC interrupt
7	Hermes

When the keyboard lock processing is enabled, and the tower case K/LOCK is locked, ***superHermes*** stops scanning the keyboards (which includes the QL joysticks) and responds to KEYROW reads with zeroes. If ser6 is being used as the ***superHermes*** mouse, that is also inhibited. If some other supplier's mouse or keyboard interface is being used, ***superHermes*** can do nothing about that.

Turbo processing is explained further in the IPCDELAY extension below. The external interrupt processing is essential for operation of the additional serial ports and the IBM keyboard. The IPC interrupt is used in order to ensure that 19200 baud ser1/2 data may be handled at full speed. Disabling Hermes is possible, but not recommended! It will most likely result in a crashed machine.

IPCEXTI%

Syntax: IPCEXTI%

This is provided for interest, and shows the ***superHermes*** devices that require attention from drivers:

Bit	
0	Keyboard lock
1	IBM keyboard
2	ser3 transmit
3	ser3 receive
4	ser4
5	ser5
6	ser6
7	reserved (0)

IPCDELAY

Syntax: IPCDELAY factor%

A factor of zero to 255 may be provided by this extension, and will slow the QL down when TURBO is off and turbo processing is enabled. By default, the factor is zero, which means there will be no actual slowing up. A value of one will mean that the QL will seem to run at half speed. The QL may be made to seem to run 256 times slower than usual by giving a factor of 255.

When this slowing up of the QL is in operation, it may impact on other hardware on the QL, if that hardware is sensitive to timeouts. The way the slowing up operates is to refuse to immediately service requests from the QL for a period of approximately factor%/56.25 seconds, during which time the QL will be in code that has turned off all interrupts, and hence will be doing nothing at all! It is very much device dependant what might be the effect of such a situation, but in terms of timeouts, they will certainly count slower! E.g. a "PAUSE 50" with a factor of 60 in operation will actually take a minute!

This slowing up will definitely work even with games that "take over" the QL, and do the equivalent of KEYROW reads only.

Note however, that a job in the QL that is timing out events by using the QLs real time clock (e.g. the DATE function) will not be affected.

IPCRAM%

This function is at least in part a debugging aid, but it will allow all of the RAM in the IPC to be read.

Syntax: IPCRAM%(address%)

The address may be anywhere from zero to 255. The return value is the byte value (0 to 255) found there.

There are only a few addresses that would be of any real interest e.g. location 4 is the ***superHermes*** version and location 5 is the revision.

IPCLED

This operates an LED connected to ***superHermes*** (Light Emitting Diode)

Syntax: IPCLED [parameter]

With missing parameter (ie IPCLED), the supplied led is set to be used as a capslock/scrollock indicator. When scrollock (CTRL F5) is active, it will flash. If capslock set/unset there will be long/short flashes respectively.

IPCLED 0- turn on LED

IPCLED 1 - turn off LED

This extension is partly for compatibility with the QView CAPSLED kit, as one can see that IPCEXT will also allow the LED to be controlled.

IPCWRP%

This writes data to the EEPROM on ***superHermes***. This will stay even after power down.

Syntax: IPCWRP%(page%,string$)

page% is a 16 byte page in the EEPROM and should be in the range 32 to 127. string$ is a 16 character string (pages 0 to 31 are reserved).

A return value of zero indicates a successful write.
Error returns:

1 Write control ACK failed

2 Write address ACK failed

3 Write data ACK failed

Case 1 can conceivably happen if you get back within the 10ms after a prior write, which I think we just managed to do from SuperBASIC, and would just mean the chip isn't talking to us for the moment. The others are disasters.

WARNING: The eprom is designed for 1 million writes, so this memory should not be used like ordinary ram. Use it to store data that changes infrequently (eg only once every bootup).

The first 512 bytes (pages 0-31) in EEPROM are reserved. The last 1.5k (pages 32-127) are for user read/write. Page 31 is used by QUBIDE to store default IDE drive parameters.

The eeprom is not write protected. If a 3 pin header is fitted next to the keyboard connector, and the linking track underneath ***superHermes*** is cut, then a jumper can be used to select protect (RH pair) or unprotected (LH pair).

IPCRDP$

This function reads a 16 byte page.

Syntax: IPCRDP$ (page%)

page% must be in the range 32 to 255 A sixteen byte string is returned after a successful read.

Error returns are indicated as a single byte return string containing:

1 Write control ACK failed

2 Write address ACK failed

3 Read control ACK failed

If you get any errors, other than 1, when trying to access the EEPROM too soon after a write to it, then contact us for possible EEPROM replacement.

SOFTWARE

SOFT RESET The Minerva soft resets (CALL 390,<data> and CTRL ALT SHIFT TAB) (as well as Miracle's RES_128) cause problems with GoldCard and Super Gold Card as Miracle Systems patch out the reset command in their copy of the ROM code, so the PIC on ***superHermes*** is not reset, and it thinks keys are still being held down on the IBM keyboard. We are working on ways around this, which may involve patching the Gold Card/Super Gold Card protected ram area. For the present, CTRL ALT SHIFT TAB will not work. To use, for instance CALL 390,17 from the keyboard, use PAUSE 40:CALL 390,17. This ensures that there are no pending keypresses. CALL 390 from programs should be OK, unless keypresses are pending when it is invoked. RES_128 will work with PAUSE 40:RES_128.

The following is mainly for the assembler programmers.

The current 8049 code left no provision for adding commands to the existing set, so how can ***superHermes*** have 99.9% compatibility with all existing usage, but still have more functions? The answer, which was a long time coming, was to "switch on" a new command set. This is done by an MDRS ("microdrive reduced sensitivity") command (MT.IPCOM command $C) with an unlikely parameter (i. e. 9). The QView CAPSLED already used this command with parameters 0 and 15, and others may have used 0 and 1 as parameters, but with only the bottom bit significant, it is unlikely that anyone has ever passed 9 as a parameter before!

Now, having sent MDRS/9, which causes no grand effect on old 8049s, at worst turning off the QView CAPSLED LED, how can HERMES be recognised? The answer here is to change the effect of an existing command! The TEST command (MT.IPCOM command $F) normally echoes back the next byte sent to it, but once HERMES has been "switched on", TEST will subsequently echo the complement of the byte sent.

Having got this far, ***superHermes*** goes one further, and reads back the version (and revision) codes, and that establishes whether plain old Hermes is there, with its version = 2, or ***superHermes*** with version >= 3. All the new commands are MDRS commands with parameter values 2 to 13, leaving parameter values 0/1 and 14/15 to operate as they would on a normal IPC. The new commands for Hermes were all useable via the MT.IPCOM trap call, as they only ever returned a byte, a nibble or nothing.

Parameter values of 0 to 9 are the ones described for the IPCEXT extension. Note that ***superHermes*** has no equivalent to the moderately useless ENT0 CLK for command 8 and the extension code issues the initial command 9 itself. It was ignored after the first time on Hermes, as once switched on, HERMES can only be

turned off by a reset.

Parameter value 10 returns the nibble as described for IPCSIG%.

Parameter value 11 expects a byte for the address, and returns the contents of RAM at that address, as described for IPCRAM%.

Parameter 12 was reserved for even more goodies, and its effect was totally undefined on Hermes. On ***superHermes***, it is the gateway down to the next level of commands.

Parameter 13 expects a byte and returns a byte, as described for RXBAUD%.

Parameter values 14 and 15 are treated the same as 0 and 1.

The ***superHermes*** commands start with $CC, and the next nibble selects the ***superHermes*** command as follows:

super 0: reserved.

super 1: EXTI returns byte as described for IPCEXTI%.

super 2: set inputs according to nibble, as described for IPCSETI.

super 3: read multiplexed signals byte, as for IPCMXI%.

super 4/5: clear/set DTR3 signal, as for IPCEXT 10/11.

super 6: close/open/read ser3..6. The first nibble select the port in its top two bits (0..3 = ser3..6) and the operation in its low two bits (0:close, 1: read, 2:open with not receive parity, 3: open with receive parity. A close stops there. A read cannot be done with MT.IPCOM, and is similar to the ser1/ser2 operation from this point on. It return a byte with overrun and framing error bits in the top two bits. Unlike ser1/2, the next bit supplies the parity error flag. The bottom five bits are the number of data bytes to follow, and will not exceed 25. The open is followed by a nibble for receive parity (top 2 bits, 0:odd, 1:even, 2: mark, 3:space) and the data word length (ls 2 bits, 0..3 = 8..5). For ser3, there are two extra nibbles next. The first has bit 0 = receiver handshake ignore, bit 1 = transmit two stop bits, bit 2 = transmitter handshake ignore and bit 3 = transmit parity enable. The second extra nibble is the transmiter parity and word length, the same as for the receiver. A final byte specifies the baud rate code.

super 7: write to ser3. Send nibble for required write length, receive one for acceptable length, send bytes.

super 8: read IBM keyboard shift register byte and status. The status byte is quite complex, and indicates the current state of the interface. The top three bits show the current state of the data and clock lines, and if the data line is being read from the keyboard. The low bitsare the past, dapa, save, tmok and active flags. If bits 5 and 0 are both zero, the interface is locked. The tmok flag is set if there was no trouble with timing. The past flag indicates that the lock occured at a parity or stop bit and the dapa flag says it was data or parity. The save flag is an additional qualifier which sorts out some cases, such as framing errors. When locked, the shift register may be a correctly completed byte, and the keyboard is ready to either have a byte written to it or to go back into the unlocked state to read another byte.

super 9: write byte to IBM keyboard (must be locked).

super 10: unlock IBM keyboard for reading (must be locked).

super 11: lock IBM keyboard (only issue one of these).

super 12: reserved for superDUPER.

super 13: disable/enable features. Send nibble top 3 bits are bit, lsb is required value, except that nibble 15 is followed by byte for the turbo factor.

super 14: EEPROM write/read. Send byte with page in 7 msbs, lsb set for read. If write, send 16 bytes, else receive 16 bytes. Then a nibble comes back to show the completion status in its two lsbs, as per the extensions. Note that MT.IPCOM cannot handle the read.

super 15: reserved.

FITTING INSTRUCTIONS

STANDARD QL CASES:

1) Turn off your QL and unplug all the cables from it.

2) Turn it over, and remove the four short screws along its front edge, and the four long screws in similar positions along the back edge. (Don't touch the other two screws, or your microdrives might start wobbling!)

3) CAREFULLY turn your QL right side up, and lift off the top half, from the front. Be very gentle with the “keyboard tails”, or you might need to get yourself a new membrane. It will better to unplug them, if you are not sure of what you are doing. The collection of single strand wires are best left alone, as they are pretty robust, and it's a real pain to plug them back in!

4) Just to the side of “mdv1”, under where the ENTER/SHIFT/ALT keys were (before you lifted them all off!) you should find a nice long chip, in a socket (40 pins). For users with re-housed QLs, this chip is where the QL changes from a wide to narrow circuit board, just next to where mdv1_ was. This is your 8049 or QFLP or Hermes replacement 8749. Using a small screwdriver, gently ease it out of its socket. Take care not to bend any of its legs, as they can snap off. You may find another keyboard interface here. As you have ***superHermes*** you will want to remove this, including any extra spacing socket underneath (if fitted). Some keyboard interfaces are fitted externally, with a connecting ribbon cable to the 8049 socket.

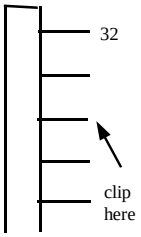
5) Put your old device/interface away in a safe place. (See step 15).

6) If you unplugged the keyboard tails, now's the time to plug them back in. They also should be firmly pushed in, but do so by holding them as close as possible to the socket. If you bend them, they'll probably crack. The tails will not fit over ***superHermes*** so make sure they bend towards the back of the QL. You will need to ease them into an `S' shape, but make sure you do not crease the tails - that will mean they may fail if the plastic is brittle. Use some masking tape to hold them in place.

7) Any cables should be connected to ***superHermes*** before fitting it.

8) Any cables fitted to the connectors underneath ***superHermes*** should be folded so that the come out underneath the ser3/RTTY connectors. Be VERY careful not

to bend the pins - they will probably break. Make sure all pins on the base of the ***superHermes*** board are not bent. If they are, straighten them CAREFULLY, preferably with a long-nosed pliers. Line up the pins with the pins of the now empty QL 8049 socket - note that the pin at the bottom right hand corner (under the PIC 17C42) is not fitted. (The > > > mark on ***superHermes*** is meant to line up with the mdv1_ circuit board). When you are sure everything is lined up, push it home firmly.



9) The LED (capslock/scroll lock) if used, will fit in a 5mm hole. The best place for this is between F3 and CAPS LOCK, but feel free to fit it wherever you like. If you have a ***superHermes*** with the extra serial port (ser3) fitted, then you will also have a hook clip. This must be connected to the expansion connector (where disk drive expansions eg Gold Card are connected). This is located to the left of the QL board. Find the end marked `A B'. Look for a right angled wire with `32' printed next to it on the circuit board. You need to clip the hook clip to wire 30B - ie the third pin down on the outside row. You cannot easily clip to the inner row, so you should not find this difficult. Make sure the cable for this lead does NOT go over the RTTY/SERIAL3 connectors if it is in a QL case - there is not enough clearance when the QL top is replaced. If you fit the ***superHermes*** to Aurora (new QL motherboard) you will not need to use this lead.

10) Any connectors can be taken out through any convenient hole - eg external microdrive slot for keyboard lead. The space between ser1 and uhf is particularly suitable (with a cutout) for fitting a 9pin mouse socket. A 25D ser3 socket can fit to the left of the rom socket. If you don't use the CTRL ports, then cables can be run out over that. Also plug in the KLOCK and TURBO if used.

11) Put the top back on the bottom and turn it all over, making sure the membrane tails are not between ***superHermes*** and the case top.

12) Put back all eight screws.

13) Plug all your leads back in and turn on the power.

14) Play about with it for a while (see getting started)

15) Move the extracted junk from its safe place to the wastepaper bin.

IBM STYLE CASES (EG MINITOWER):

1) Disconnect the power cord and open the case

2) Locate the 8049 socket marked IC24. This will be where the wide QL circuit board ends, and will probably have a large keyboard interface fitted. Remove this and keep in a safe place (see step 11)

3) Any cables should be connected to ***superHermes*** before fitting it. In particular turbo, capslock led and keylock connectors should be fitted such that the embossed writing is visible. A single black wire (usually marked IN) can be connected (see earlier). This is where the control of the Turbo LED is by grounding this 'IN' pin - it may not be called this! If all you have is a turbo LED with separate lead, then do the following: If you are not using the capslock led,

then simply connect the Turbo led to both pins of this connector on ***superHermes*** and connect the yellow lead to the IN pin. Use two short lengths of signal wire (like the flying lead for EXTI), one should join a connecotr to a diode (eg 1N4001 - BAR end). The other end, connected to the diode, can be bared and inserted in the +ve connector of the led (yellow) which is then plugged into the led connector with the yellow wire nearest the edge of ***superHermes*** circuit board. (If you are doubtful about making this lead, send an SAE or IRC to us for a free lead). Polarity of the IBM PC LED varies. The IBM PC I based ***superHermes*** design had letters uppermost when fitted. Some modern minitower cases I have tried recently require the lettering downwards - experiment - you will not damage the LED or ***superHermes***. In all cases black should be ground, and is nearest the centre of the superHermes board.

Of course, you have to connect the turbo switch to the connector on ***superHermes***. The IBM PC I used as a model had a three pin connector. For some modern cases (esp without an LED display) with only a two pin connector, connect this to the two turbo pins nearest the 'IN' pin.

Issue the command IPCLED 0 in SuperBasic to activate the LED. When the LED is off, then the setting of IPCDELAY will determine how slow (if at all) the QL will be.

4) Keep the serial 3 side of the board nearest the J1 expansion connector (furthest from where the microdrives were). Any cables fitted to the connectors underneath should be folded so that they come out underneath the ser3/RTTY connectors. Be VERY carefull not to bend the pins - they will probably break. Make sure all pins on the base of the ***superHermes*** board are not bent. If they are, straighten them CAREFULLY, preferably with a long-nosed pliers. Line up the pins with the pins of the now empty QL 8049 socket - note that the pin at the bottom right hand corner (under the PIC 17C44) is not fitted. When you are everything is lined up, push it home firmly.

5) If you have a ***superHermes*** with the extra serial ports (ser3 etc) fitted, then you will also have a hook clip. This must be connected to the expansion connector (where disk drive expansions eg Gold Card are connected). This is located to the left of the QL board. Find the end marked `A B'. Look for a right angled wire with `32' printed next to it on the circuit board. You need to clip the hook clip to wire 30B - ie the third pin down on the outside row. You cannot easily clip to the inner row, so you should not find this difficult. (See diagram under fitting in QL case section)

6) There are cutouts in the case for `D' connectors. The panel mounting keyboard socket will fit in the circular hole for the IBM keyboard.

7) Replace any expansion cards

8) Put back the case cover

9) Plug all your leads back in

10) Play about with it for a while (see getting started)

11) Move the extracted junk from its safe place to the wastepaper bin

GETTING STARTED

BEFORE STARTING READ UPDATES_DOC ON UTILITIES DISK

All standard QL features (QL keyboard/ser1 and ser2 input and sound will be there at bootup. To get the extra features, the IPCEXTCC_BIN file needs to be loaded. Either use “a=RESPR(<size>): LBYTES'flp1_IPCEXTCC_bin',a: CALL a” or, if you have TK2, just “LRESPR 'flp1_IPCEXTCC_bin'” and put this in a BOOT file. Naturally change the device (flp1_) to where you have saved the extensions file. 'CC' is the country code. Note that IPCMOUSE and IPCLED commands are needed enable the mouse (after pointer environment loaded) and LED (as capslock/scrollock) respectively.

If you do not have a QL keyboard, then you MUST boot your QL off the disk provided. Then modify your BOOT (eg win1_boot) to include the above code. The IBM keyboard (other than F1/F2) and most extra features in ***superHermes*** will be inoperative without the extensions.

APPENDIX 1
PINOUTS

SERIAL 3

superHermes	Colour	Function	Direction	25D	9D
1	brown	DCD	input	8	1
2	yellow	DSR	input	6	6
3	green	RX	input	3	2
4	blue	RTS	output	4	7
5	white	TX	output	2	3
6	red	CTS	input	5	8
7	orange	DTR	output	20	4
8	-	RI	-	-	9
9	black	GND	-	7	5

If a 9 way ribbon cable is connected, then this will fit directly onto an IDC 9D plug with the pin numbers shown above. Use of a 25D plug is recommended (as sold with ***superHermes***) as a mouse also uses a 9D plug.

KEYBOARD

superHermes	Colour	Function	Keyboard
1	green	clock	1
2	white	data	2
3	-	-	-
4	black	GND	4
5	red	+5V	5

EXTRA THREE RS232 INPUTS

superHermes	Function	Direction	capacity	9D MOUSE
1	GND	-	-	5
2	RX	input	-	2
3	+9v	-	25ma *	4,7
4	-10v	-	reference	3

* total capacity for all three ports

SPARE CONTROL/SENSE LINES

superHermes	Function	Direction
1	GND	-
2	+5V	-
3	spare 3	bi-directional
4	spare 2	bi-directional
5	spare 1	bi-directional

APPENDIX 2

Major problems with the old 8049 code

Serial Input

Serial input requires signals from the QL when its character receive buffer is full (ser1 CTS - Clear To Send and ser2 DTR - Data TerminalReady). This could easily not be activated in time by the 8049, due to badly designed code. It results in, at best, the loss of incoming characters. Even worse, erroneous code in the 8049 could result in a "serial overrun". This is a familiar occurence to people using modems, and has the strange effect that the 8049 seems to "hoard" a numberof characters, doling them out only when a new serial character arrives. When this happens, in practical terms, serial input is no longer usable without a full QL power down.

There can also be hardware problems outside the 8049 (see appendix 3)

Keyboard handling

Key rollover (especially when shift keys are involved) doesn't work well. Also, when keys are pressed there is a tendency for unwanted repeat characters (key bounce). This occurs in various ways, especially on the early version of the 8049 when a key was typed, but a neighbouring key was also touched.

It is also a particular problem with added keyboards, such as the Schön and Keyboard Products keyboards, where (it would appear) the mechanism results in a slightly "crackly" key.

There are two versions of the 8049 attributable to Sinclair Research, the later version 2.0 offering a slight improvement.

There is another de-bounce 8049 replacement available, which does achieve full de-bounce, but at the cost of totally wrecking serial input from ser2 and slowing KEYROW input (mainly a problem for games software).

Baud rates

Independent baud rates are not possible.

Input at 9600 requires two stop bits.

Input at 19200 is impossible.

Sound

"Fuzzy" and "random" on sound commands shift the underlying pitch.
Sound duration is dependent on the pitch.

All these problems are cured by *superHermes* (and Hermes)

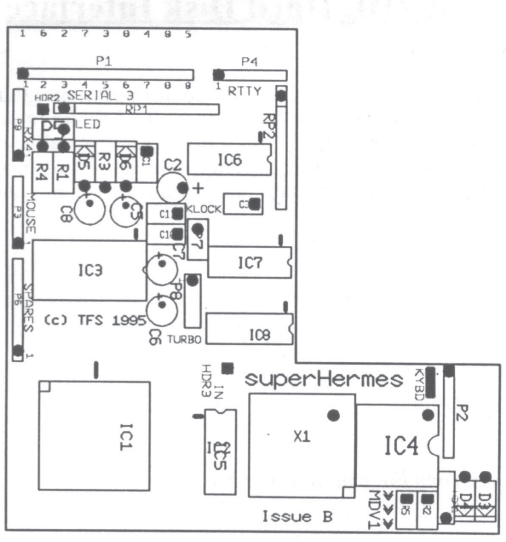
APPENDIX 3
Hardware Considerations

If you find you are getting data input corruption using ser1/2 then check the following:

- Have you opened the QL port with handshaking enabled.
- Does your serial cable support handshaking. QL/CTS to SENDER/CTS and QL/DTR to SENDER/RTS. (Many leads have “loopbacked” handshaking)
- Is the QL IC25 working (There should be output on relevant handshake line of -12v to stop ser1 CTS/pin8 or ser2 DTR/pin11). A 25D RS232 tester, with red/green leds, can help diagnose faults.
- Is the sending device set to accept handshaking. Check the QL serial socket wires are all there and contacting the plug.

(ie superHermes cannot perform miracles)

APPENDIX 4
Circuit board layout (top view)



APPENDIX 5
Falkenburg Hard Disk Interface

The Falkenburg hard disk interface makes WIN1_ a higher priority boot than FLP1_ (or even MDV1_). This means for instance, that if you make a win1_boot error, or the win1_boot file loads but fails later due to a 'bad medium', and it has not yet loaded the ***superHermes*** extensions, then you will not be able to use the IBM keyboard. It is suggested that you put the following as your first lines, and do this BEFORE installing ***superHermes*** and IBM keyboard (and make the necessary changes to the BOOT).

```
10 TK2_EXT
20 FLP_USE 'flp'
30 IF FTEST('flp1_boot')=0:LRUN 'flp1_boot'
```

This will allow you to boot off floppy disk (and load the ***superHermes*** extensions) if you get problems, unless of course 'win1_boot' fails with bad medium. Clearly it is very desirable to use the IPCEXTCC_ROM (see earlier).

APPENDIX 6
Qubide IDE Hard Disk Interface

Version 1.51 or greater of the Qubide interface comes with a program to save the IDE disk parameters in Minerva MKII battery backed RAM or ***superHermes*** EEPROM (EEPROM page 31). If you run the program (storeSH_obj) this will store the parameters in ***superHermes*** and will allow a faster boot, especially when SMSQ is loaded. Thanks to Phil Borman for the initiative in incorporating this.