**The Pointer Environment**

**Ian Bruntlett explains a significant upgrade to QL usability.**

This article describes, explains and demonstrates the Pointer Environment while assuming only a little QL knowledge.

In the future I will cover Qpac2 and other things there are no room for here. Some   
explanations are in the Appendices at the end of this article. Inexperienced users may wish to refer to the appendices while they are still gaining their “QL-legs”.

If you know next to nothing about the QL I suggest you browse through the *QL User Guide* (don’t worry if you don’t understand it) and gossip with some people who know about computers - you may pick something up. See Appendix “Commands you’ll find useful” for the SuperBasic commands to use when taming your QL in the future.

**Meetings**

A good source of QL people is a local Quanta meeting.Non- members are often welcome, and you will find out more about Quanta and your QL.

The novice QL user is likely to get more from attending Quanta meetings than most people. Most of a novice user’s problems will have been experienced and solved by the people there, so don’t be afraid to ask questions. No-one in computing knows everything - be wary of people who reckon they do.

The QL uses the Qdos operating system and its SuperBasic interpreter and (when new) comes bundled with the four Psion programs - *Quill Archive, Abacus, Easel* and (sometimes) a games microdrwe cartridge. No disk drives, printers or extra software were ever supplied with the standard setup by Sinclair.

This presents some interesting problems. You can do plenty of useful work on an unexpanded QL with just the Psion programs, but this does not do the machine justice.

**Disk drives**

Disk drives are faster than microdrive cartridges, more reliable and, when storing large amounts of information, much cheaper. A floppy disk costing less than 50p holds 720 kilobytes (KB) of information while a microdrive cartridge holds just over 100KB and costs about £2. In real terms, when you fill a single disk with information it is at least £10 cheaper than storing the same amount of information on microdrive cartridges. It is worth having a pair of reliable disk drives - cheap, unreliable disk drives are not worth the trouble they cause.

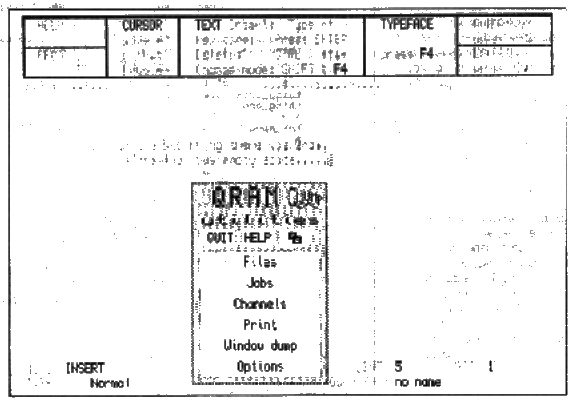
Additional memory allows you to edit longer documents with greater ease and have more than one program in memory. A unique feature of Qdos is that it uses “free/spare memory” to speed up file access, remembering the contents of files it has recently handled.

On my Gold Card QL with plenty of free memory (1252KB at the moment), file handling can almost be too fast. This happened when I was converting some C68 manuals into *DataDesign* files. The manuals were converted from Quill documents to text files and saved on my slow 525-in drive. A small SuperBasic program read the work files and produced a DataDesign database. The text files had to be edited and re-saved occasionally because the SuperBasic program sometimes became confused. So a loop of “write text file”, ‘process text file”, “find a problem”, “fix it” and go back to “write text file” started. This was faster on a Gold Card because when “write text file” was done, Qdos had enough free/spare memory to keep a copy of the text file in memory. When the “process text file” started, Odos checked that the disk had not been changed and just used the memory-based copy of the text file. This process of keeping copies of files in free memory without any involvement of the user is called “slaving” in the *QL Technical Guide.*

This tactic of the QL may seem like common sense but no other operating system does this as well as Qdos - other operating systems insist on having a fixed amount of memory for buffering. I use an improvement of Qdos, Minerva, which has given me no problems with Gold Card - it has been said that the Sinclair versions of Qdos don’t take as much advantage of the Gold Card as Minerva. I have no intention of messing about with a Trump Card and a Sinclair version of Qdos to verify this.

**Hardware**

Back to the Pointer Erwironment This needs a QL with at least 384KB of memory, a monitor, power supply and a copy of a program called ‘ptr\_gen’. What an anti-climax! After the last paragraph you were beginning to think you’d need a second mortgage to afford all the kit.



**Multi-tasking.**

This is where a computer can have more than one program in memory and split its working time between the different programs (jobs). A bit like a secretary answering a phone, writing a memo using a word- processor, drinking coffee and chatting with a friend. The QL’s multi-tasking does not have ‘virtual machine” protection so if a program crashes, it can affect all the programs in your computer - the equivalent of the secretary swearing at a telephone caller instead of the photocopier and dropping the coffee on the word processor. ‘Virtual machine” protection on a micro-computer is very unusual - OS/2 on a 486/386 PC has it but Qdos has yet to take advantage of the 68030 or 68040 where virtual   
machines may be possible. This lack of protection really is as drastic as it seems. When working with a computer you should save data regularly, make sure you have at least one back-up and gently introduce new software into your work.

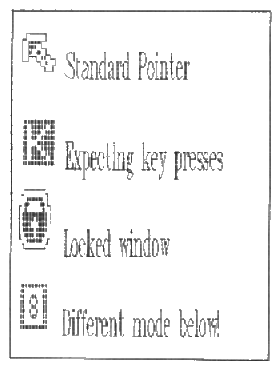
It is no good to take an isolationist point of view with software, buying some software, getting that working and sticking with that permanently. Ignoring the rest of the world put the Chinese Empire at a   
disadvantage, so don’t make the same mistake. Try to be aware of what is on offer and get   
friends to show you new programs they have bought.

**Window-wipe**

Before the Pointer Environment was introduced, the QL had ‘destructive windows”. When multi-tasking, the user would swap from one program to another and would usually press a key (F4 or Shift-F5) to get the progrwn to re-draw its screen. Unfortunately this would demolish the displays of the other programs. The result was a mess, but **it worked** and was economical with memory. The QL would multi-task and the user would just have to wait for the screen to be re-drawn. This mutually assured screen destruction worked as long as each program could re-draw its windows and had an active cursor. Many programs couldn’t re-draw their windows, and many didn’t have an active cursor. Those programs that didn’t have an active cursor could sometimes ‘lock out’ the user when involved in multitasking.

This state of affairs could not go on forever. It is easy enough to re-draw the screen of a word- processor but very inconvenient to re-draw that of a Fractal Generation program just because something else has corrupted its display.

The Pointer Environment improves the QLs handling of the screen so that windows are preserved automatically This was not done when the QL was first brought out because there wasn’t enough memory inside an unexpanded QL and expansion memory was expensive (Roughly £1 per KB - at those prices Trump Card would be £900-plus, instead of £95 plus P&P from Qubbesoft PD).



**The Pointer**

The Pointer Environment was introduced in 1987 with Qjumps Qram (Figure one), a software package intended to take advantage of expanded QLs. The Pointer Environment stopped programs from overwriting each other’s windows and added the use of a pointer device (usually a mouse) at a time when mice were becoming very popular on other machines.

Because most the mouse supply to QLs was still limited. the Pointer Environment was written to be used with either a mouse, or the keyboard, or both. Try using Microsoft Windows on a PC or Open-Windows on a Sun work-station without a mouse and you will appreciates that this “mouse independence’ is no mean feat *(****The man******speaks true, say Mac and PC users.)*** On some systems, if your mouse breaks down the whole system is virtually useless. The Pointer Environment had some other facilities for Qram - it introduced standard routines for handling images. Little symbols could be defined as spntes and put on the screen without the programmer having to think about how the data should be created (sprite editors are available) or how to display the image. The ‘Qjump” logo on the Qram menu is a sprite. This is different from other computers which usually have sprites flying all over the screen, changing colours and blasting each other to pieces. The Pointer Environment sprites (apart from the cursor sprite) don’t move.

The ‘pointer’ in pointer environment refers to the cursor sprite which the user can move about the screen using a mouse or the keyboard. The shape of the pointer sprite/picture actually changes according to the state of your QL (Figures two and three). If I move my QlMl mouse while typing this article into text87, a picture of a “K” appears, telling me that the program underneath the pointer is waiting for keyboard input If the pointer sprite was a red “no-entry’ sign then it would mean that the program under the pointer does not want any keyboard input.

**Compatibility**

If a piece of software is described as “incompatible with Qram”, you can be sure that it is conflicting with ‘ptr\_gen’ from the Pointer Environment The conflict is sometimes blamed on the ***Window Manager*** ‘wman’. This is nonsense.

In the Pointer Environment ‘ptr\_gen’ is the active partner, extending the system The Window manager’ is a set of routines for handling a consistent window/pointer- based interface that uses ‘ptr\_gen’ to handle the mouse pointer, keyboard, preserving windows and drawing sprites. The Window Manager only manages individual windows, telling the programmer what the user is doing - all the nitty-gritty is done by ‘ptr\_gen’.

**Technical**

Qjump used the Window Manager to give a mouse-driven interface for Qram. So that other programmers could use the Pointer Interface and Window Manager routines, Qjump produced *Qptr* As a reference guide to ‘ptr\_gen’ and ‘wman’, Qptr is good. If you know about what you want and where you will find it the Qptr manual will be very easy to use. For someone who is learning something new about Pointer programs, Qptr is a complicated manual that pushes you back on your own resources Qptr provided. some complete programs as examples (a sprite editor, a paint program, and a ‘demo’).

A Utopian Qptr would also have had many small examples with explanations. Each demo could have shown an individual aspect of either the Pointer Interface or the Window Manager. Qmon would have been updated to help when debugging Pointer programs.

For years I laboured under the idea that the Window Manager made life hard for the programmer to make life easier for the user. Now, I am being told that inexperienced users are finding pointer-driven applications hard to get started on. The Window Manager is responsible for drawing the mouse-driven windows of most Pointer Environment applications. It is possible to bypass the Window Manager and call the ‘ptr\_gen’ routines directly with TRAP calls but you will have to wait until another episode for examples.

**Demonstrating**

Assuming that you have a disk drive QL, with at least 384 of memory, *Toolkit 2*and a pointer program, I will give a brief demo of a selection of pointer programs (*Qram, Qpac1,* Quanta library). **Commands to be typed in will be in italics.**

I have tried to keep this as simple as possible so do not be upset if you think I am talking down to you. Great care is being taken to keep this article accessible to comparative novices.

In the QL, most jobs have windows. When the Pointer Environment is installed, it is possible to move from one job’s windows to another by pressing CTRL-C.

When using a QIMI mouse (the QIMI interface and mouse is now available from Quanta), I tis possible to select a job by moving the pointer (via the mouse) from the current window to another job’s windows and pressing the Left mouse button. QIMI is the best interface for this as it can move the pointer even if *Quill* is being used.

There is a serial mouse interface available which supports the use of the third (Middle) button. This usually acts as Quit/Escape. When pressed simultaneously with the Left button, it emulates “Wake”. When pressed simultaneously with Right button, it emulates “Sleep”.

****

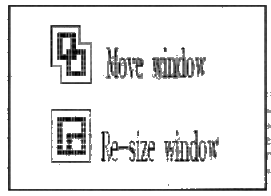
**Key and Mouse**

The QL’s keyboard can emulate its mouse quite well – the cursor keys will move the pointer arrow and the mouse buttons are emulated by the Space bar (Left button) and Enter (Right button). The pointer environment from v1.23 of ptr\_gen onwards, has commands to allow cursor key movement of the pointer to be disabled (CKEYOFF) and re-enabled (CKEYON).

If you are using the Pointer Environment without a mouse, then the pointer can be moved to another window with the cursor keys. To select a job under these conditions, just move the pointer over the desired job’s windows and press Space. If you are in Quill or some other text-related program the cursor keys are used for moving around a document. So you use Ctrl-C to switch from one program to another until you reach the right one.

If you look at the various screen dumps of pointer programs you should notice similarities. They have a title, a scattering of options at the top of the window and (usually) a display specific to the task inhand – a list of files, a list of jobs, etc.

Referring to **Figure one**, the “Quit”, “Help” and “picture of two little squares” options may be selected by moving the pointer over them. When the pointer moves over and option, that options becomes the “current option”. The “current option”, if any, will be outlined in either black or white. Selected options that affect the status of a program (e.g. Sound On in a game) appear in a different colour, according to their status On/Off, etc. If an option is unavailable then this will usually appear in a subdued colour and you will not be able to select it.



**Subtle**

Pointer programs handle their options in a very subtle way. The Qram manual gives good guidance on these points.

**To Do something from a menu:**

A “do” is achieved by hitting the Enter key or Right mouse button. A “do” will select an option, and usually invoke any associated function, such as file copy. A “do” on an option may also be put into action by typing a single character, usually the first letter (such as N to “Save with New name”). QUIT breaks this rule by using ESCape. Some options may not have this feature – filenames and the Format option are examples (From the Qram Files menu).

**To change the status of an option:**

A “hit” on an option is achieved by moving the pointer to the option using the mouse or the cursor keys, and pressing Space or the Left mouse button. A hit will toggle an option’s status between available and selected.

The above definitions are best used when playing around with a pointer program. The difference

between a Hit and a Do is important but don’t try to learn it by rote. Just get used to the Pointer

Environment and you will “learn” it naturally.

The Sleep and Wake/Update options are not supported by all Pointer programs They were introduced to users when Qpac2 was launched. See Figure six for the Sleep and Wake symbols.

*To put a job to sleep*

Press Ctrl-Fl orHA’ the “Zzz” symbol (see Figure six). This hides the job’s windows and it is stopped. A button appears in the button frame with the program’s name. The button

frame is specific to Qpac2 and will be explained, with Qpac2, later in the series.

*To update a window (Wake)*

Press Ctrl-F2 or Hit the “jagged/lightning strike symbol (see Figure six). The windows will be brought up to date. This would be used in a jobs menu, for instance, to update its jobs list if the user had quit some of the programs and returned to the jobs menu.

*To Resize a job‘s windows*

This option has been supported since Qpacl came out (See Figure four, “Re-size window”). Press Crl-F3. Programs handle re-sizing of windows in three different ways. (Most QL programs ignore

requests to change the window size.)

Many Pointer programs will change the pointer arrow to a Bizarre Looking Blob (BLB) when changing the window size. The pointer moves to a new part of the screen and the new window size follows the pointer position (a process widely known as “dragging’). If you move the pointer to a new position above the old position, for instance, the window will be taller. If the pointer is moves to the left of the old position, the window is wider. If you move inwards or down, the window will be smaller.

Other Pointer programs, in particular the Opac 1 CaIendar have a distinct set of window formats. Telling this kind of program to re-size its windows will result in the next window format in the cycle being displayed. The Qpac1 Calendar goes from a medium sized Calendar (displaying Clock and Month) to a large sized calendar (displaying Clock, Month and extra displacement info) to a tiny clock, not showing the days of the month at all.

**To Move a job’s windows**

This option has nearly always been supported (see figure four, ‘Move Window’). Press Ctrl-F4. The pointer will change into a “double block” symbol. Moving the pointer and pressing Hit Do or Escape wilI move the window to the pointer’s current position. It is possible that Escape will one day abort the move window option, so don’t get into the habit of using it to complete a ‘move window’ operation.

**Freezing the display**This is available without the Pointer Environment *Press Ctrl F5****.*** It delays any attempt to print anything on the screen until Ctrl-F5 or another key is pressed.

If the Pointer arrow is visible when Ctrl-F5 is pressed then the pointer will turn into a padlock (See **Figure two,** “locked window”).

**Preparing for the demo**

Switch on your QL.  
Press F1/F2 according to your display type.  
If you pressed F2 type in the command MODE 4 and press Enter.

Put the appropriaie disk in disk drive one.

**DEMO 1: With Qram**

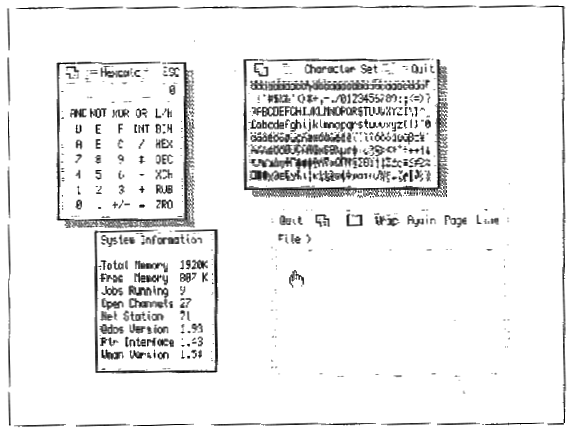
**TK2\_EXT   
DATA\_USE FLP1\_  
PROG\_USE FLP1\_**   
**LRESPR FLP1 \_Boot\_REXT**

This loads PTR\_GEN, WMAN and Qram itself.

*Press the Alt key and / simu/taneously*and you should get the Qram menu, a little rectangle, appearing on the screen (Figure one). There should be a little arrow on screen (the Pointer) which you can move around with a mouse or the cursor keys.

When the pointer moves over an option that can be selected, the option is highlighted with a little white box. Qjump’s manual calls this the “current item” (page 9).

*Move the pointer around the screen using the cursor keys.*When the pointer is out of the Qram window, it changes to a padlock, which means that the window below the pointer is “locked” because it is buried by another window, in our case Qram’s window. A locked window cannot be updated until it is unlocked, usually by ‘unburying” (exhuming?) its windows. To do this, *move the pointer to the locked window and Hit it by pressing Space or the Left mouse button.*The selected job will be brought to the top of the pile of windows and any keypresses will be sent to it



**Psion**

To multi-task Psion programs with Qram, you need to process your copy of Quill, etc., with the program *Grabber* Simply put your Qram disk in drive one, enter the command EX FLP1 \_GRABBER, then put your Quill disk in drive two. Follow the instructions on Page 19 of the Qram manual. They worked for my ***Psion XChange*** a long time ago. Grabber will modify Quill so it will behave itself property. If you save the new Quill to FLP1\_GRQUILL then EXEC FLP1\_GRQUILL will run your new Quill. This process can be applied to Abacus, Archive and Easel. The “Grabbed” copies ofthe Psion programs cannot be configured with the Psion CONFIG program but there is a program by Rich Mellor, “CONFiGURE”, that will- it is on the Quanta libraiy disk PSION\_1. I am unable tovenfy that “CONFIGURE’ works as my copy of PSlON1 seems to have disappeared!

The Qram program HOT\_MAKE will set up a file of programs that can be resident inmemory and called up with a Hotkey. Simply put the Oram disk in FLP1 - and follow the instructions. It cannot put normal SuperBasic programs on ahotkey but it can put most QL jobs (started with an EXEC or EX command) on a hotkey.

Having the Psion programs behaving themselves and running from a hotkey is fine, but why not have them automatically LOAD or have them in memory all the time? It is possible to do all this with the Hotkey System Two (hot\_rext). This was developed as the successor to Qrams HOT\_MAKE and is much more flexible.

Qram is now obsolete - when Qram was replaced by Qpac2, old Qram users were allowed a discount when upgrading to Qpac2. But Qram had a few facilities that Qpac2 doesn’t - it had a screen dump menu and the ability to automatically configure itself. Qpac2 has many things that Qram does not have and there are ways for a Qpac2 user to get round the missing facilities See the Qpac2 demo in a subsequent article for more details.

**DEMO 2: With Qpac1**

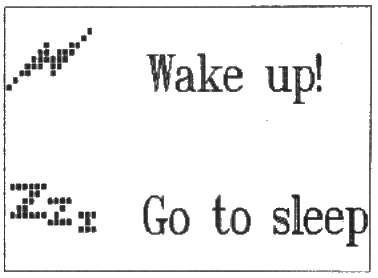
Qpac1 introduced the Hotkey System Two, a new CONFIG program and a set of six utilities (Calendar, Clock, Alarm Clock, Calculator and System Monitor). Because Qpac1 offers more flexibility, it is easy to be overwhelmed by the choices when trying to create a convenient boot disk for the first time

**Using the QPAC1 utilIties**

**TK2\_EXT   
Put a copy of the QPAC1 disk in disk drive one.   
LRUN flp1\_Boot**

After booting from a copy of the Qpac1 disk, *press the**keys Alt and c together****.*** You should get the Qpac1 calculator to “pop-up”. *Press Alt-k* The calendar should pop up. There should be a little arrow on screen (the Pointer), which you can move wh a mouse or the cursor keys. When the pointer moves over an option that can be selected, the option is highlighted with (usually) box this is the “current item”.

Move the pointer over the screen using the cursor keys. When the pointer becomes a padlock, the window below is “locked”. To un-bury it, move the pointer to the locked window and Hit it by pressing Space or the Left mouse button. If the Calendar or the Calculator overlap each others windows, you will need to Hit the overlapped program (either Calendar or Calculator) before you can use it You can shift the windows. Moving the pointer arrow over the “double box” symbol (See *Figure four,*“move window”) and Hitting it changes the pointer arrow to a “double box”. Move the pointer to where you want the window and press Space or Left mouse. Now you will be able to go from the Calculator to the Calendar and back again, without having to “un-bury” windows.



**Be thorough**

The Qpac1 manual covers the use of the Qpac1 utilities well. Do not be fooled by its brevity, go through each section, or you will miss something useful.

**Config**The ***Config*** program is a “universal” configuration program. Most Pointer-driven programs use this program to CONFiGure themselves. The advantage is that.only one program is required to CONFIG many programs. Programmers can incorporate “config” blocks in their programs instead of writing a config program for each program they write, saving time. Users only need to handle one config program, so they have fewer programs to learn and keep on disk.

Type EX fIp1 \_CONFIG to run Config. The prompt “Give the name of the next file to be configured or press ESC to quit the program” will appear. If Config has a flashing cursor next to the prompt then it is waiting for keyboard entry of the filename - type in fIp1\_calculator and press Enter.

The later versions of config use the ‘menus’ extensions if they are present. If the prompt doesn’t have a flashing cursor next to it you should see a large window with the title “Select Filename”. This is the “File Select” menu by Jochen Merz. See the Appendix *Using File Select”* forfull details.

The question “Configure Calculator version etc (Y, N or ESC)’ appears. *Press Y*. Then the question “Prioritised or immediate calculation?” appears. Space will toggle between “Prioritised” and “Immediate”, Enter sets the current choice.

Then we are asked where the configured calculator is going to be saved. The cursor is at the start of the filename (fIp1 \_calculator) and this may be edited to save Calculator with a different name. If you attempt to save the configured calculator over the old one then approval is asked “File exists, OK to overwrite?”. If you change your mind about the choices you made during the configuration then pressing Escape will abandon the configuration changes.

**Rebooting**

Common sense would suggest that once a file has been configured and saved, the changes made with Config will take effect immediately. If a resident program (a toolkit system extension or job that has been HOT\_RESed or HOT\_CHPed) has been configured and saved then nothing will take effect until the QL is re-booted.

The only way to configure a resident program without the need for re-booting is to configure it remove the old version from memory and load the new version into memory. This is not usually practical so most people tend to reset their QL instead.

As usual, if you find a newer version of Config when you get a new Pointer program, use that instead of your old version. It may be a good idea to put the newest Config on your Boot disk The very early versions of Config had bugs, and the later versions have extra features.

Config’s biggest disadvantage for the user is that there is no “update” option - given an old configured file and its new, unconfigured replacement file, Config won’t configure the replacement file automatically with the information present in the old configured file.

The drawback for programmers is that Config is mainly for assembly language. Oliver Fink wrote a program to incorporate Config information in SuperBasic programs compiled by Qliberator (see Quanta disk SPECIALS\_4). It is possible to write “action routines” to perform the configuration if the standard behaviour of Config is not enough. Unfortunately they have to be written in assembler. It would have made life so much easier if they could have been Qliberated SuperBasic as well.

**Integrating Qpacl**

Integrating Qpacl into your existing set-up can be very easy or very hard. If you already use the Pointer Environment and Hotkey Two then integrating Qpacl is merely a matter of deciding how you want the Qpac1 jobs to be invoked. You need to decide whether they should be loaded from disk when needed, or made resident when Booting. Making the programs resident uses more memory but is convenient especially if you don’t have a hard disk

The people who will find integrating Qpacl hard are those who don’t use the Pointer Environment and don’t know much about SuperBasic. Some of their programs will clash with the Pointer Environment and they will be faced with an alarming case of culture-shock. If you use some old QL software with a “swapper type of program, your best course of action is to get an experienced person to help you.   
This section refers to certain “Hotkey” commands (prefixed by a HOT\_) refer to the appendix “Hotkeys” for brief definitions

A simple way of having your Hotkey programs set up is to HOT\_RES them - whenever the Hotkey is pressed, a new job will be created. No additional memory will be used for the program code but memory will be used for the job’s data-space and its screen image. Also, your computer can become cluttered if you have alot of jobs in memory. This wastes a lot of the flexibility of the Hotkey Two.   
A better way is to use the Hotkey Two to start a new job only if it is needed. Consider Qpac1’s calculator - to pop-up a new Calculator job every time I pressed Alt-C, I would set it up with the command:

***70 ERT HOT\_RES(’C’,’flp1\_calculator’)***

Normally I would only ever need one calculator running at a time so I could set it up on a hotkey with:

***70 ERT HOT\_RES1(’C’,’flp1\_calculator’)***

When this statement is executed (usually) in the Boot program it loads the calculator into memory and links it in as a ‘Thing’. It also tells the Hotkey system that when Alt-C is pressed it is to look for a calculator job - if it already exists then it is to display the calculators windows (Pick it) instead of creating a new calculator job.

None of the above methods are good enough for me. I prefer to have my cake and eat it So I define two hotkeys to deal with the same program.

***70 ERT HOT\_RES(’C’,’flp1\_calculator’):REMark make resident & set key***

***75 ERT HOT\_WAKE(’c’,’calculator’):REMark set another key for calculator***

I press Alt-c to run a calculator and Aft-Shift-c to run a new copy of calculator. I manage to remember which key does what by using the Alt-Shift-key combination to perform various things (HOT\_LOAD, HOT\_RES) and the normal Alt-key combination to run the program “normally”. The above sequence of commands will work if I want the program to be resident but what if I wanted the program to be loaded from disk? I would use these commands:

*70 ERT HOT\_LOAD(‘C’,’flp1\_calculator’)*

*75 ERT HOT\_LOAD1(‘c’,’calculator’)*

The calculator would be loaded from disk when needed - each time Alt-c or Alt-C is loaded, a new calculator would be loaded from disk. If it is likely that many calculator jobs will be running at any one time then a HOT\_RESed calculator would be more economical in its use of memory The HOT\_LOAD routine is ideal for making hotkeys for programs that are only run occasionally.

People who are determined to build a Qpac1 boot should refer to the Appendix *‘“Building a QL boot program”* and *“Hotkeys“*

**DEMO 3: Pointer Software**

This demo needs two disks from the Quanta library, SPECIALS\_0’ and ‘UTILS\_GEN4’. Put SPECIALS\_0 in flp1\_

*TK2\_EXT   
LRESPR flp1\_PTR\_GEN   
LRESPR flp1\_WMAN*

This loads an old copy of the Pointer Environment. Some Pointer programs may refuse to run with it saying “Pointer Interface too old”.

The disk UTILS\_GEN4 has a selection of pointer programs. I’ll deal with HEXC, VIEW, CHAR and INFO. **Figure five** is a screen dump of HEXC, VIEW, CHAR and INFO in action. Put UTILS\_GEN4 in fIp1\_. Type:

*EX flp1\_HEXC\_exe:EX flp1\_INFO\_exe*

Youshould get a display givng system information - the amount of free memory, network station number, varlous version numbers (be careful, my copy gets the version numbers of the Pointer Interface and the Window Manager mixed up, its easy to miss - Dick Copland spotted this one).

There should be a little arrow on screen (the Pointer). When the pointer moves over an option that can be selected, the option is highlighted with a white box - this is the “current item”. Move the pointer over the screen using the cursor keys. When it is a padlock the window below it “locked” as before.

The INFO program has buried the windows of HEXC, the hex/decimal/binary calculator (Hexcalc. Press Space and the pointer arrow will turn into a “double box’ symbol to move a window. Move the pointer away to the far side of the screen and press Space or Left mouse button. Thisbehaviour of INFO is not standard. The key-press for move-window is usually Ctrl-F4. You should be able to see the Hexcalc window and the *‘System Information”* window **(Figure five.)**

Havingmoved the windows so they don’t overlap move the Pointer to the Hexcalc window and press CtrI-F4 or Hit the “double box” option for “move window”. Move Hexcalc so its windows are only just overlapping System Information’s windows. Move the Pointer over System Information’s windows - it should change to a padlock. Press Space or Left button and Hexcalc’s windows wil be “unburied” covering System Information’s windows If System Information or the Hex Calculator jobs overlap each other’s windows, you will need to Hit the overlapped program before you can use it.

Press Ctrl-C until you reach SuperBasic and type in the command:

***EX flp1\_VIEW***

So far, in all these demos, the pointer is normally an arrow. The View program provides a custom pointer, a “hand” symbol. This is shown in Figure five. When the Pointer **is over the** View window, it turns into a hand. Press CtrI-C to get another program on screen. Move the pointer over its windows You should find that the pointer is again an arrow in the other program. View is a simple program - simply give it a file name and it will display the file in its windows, waiting for you to ask for more. As View does not read the whole file into memory (it opens a channel to the file and reads information when it is required), you should not remove a disk that has a file you are ViEWing. Any attempts to read a bit more of the file by the View program will fail. Any attempt by other programs to use other disks in the same drive will fail with the error message “Files still open” and “not found”. Wrap is the only obscure option in View - see the Appendix ***“****Viewing files on your QL****’*** forWrap On andWrap Oft

**Play around**

Play around with the pointer programs A Pointer Environment manual isn’t provided with these Quanta disks - experlment and use this article for information. You won’t be able to define hotkeys, as ‘hot\_rext isn’t in the Quanta library, but the other Pointer programs will be available.   
**THE APPENDICES**

***Viewing files on your QL***

If you have Qram then pop-up the Qram menu and select/Hit Files. Then select/Hit any files you want **to** view andDo View (press ‘V’). Pressing “N” for “Next” will move to the next file if you selected more than one file to view.

It you have Qpac2 then popup the Files menu. Then select/Hit any files you want to view, press F4 and Do on the name ofa file you want to view. Pressing ‘<‘ or ‘>’ will move to “previous” and “next” files if you selected more than one file to view

If you have *Toolkit Two* thenthe View command will showfiles in aSuperBasic window but long lines will be not be shown in full- only the part that fits on a single row willbe displayed, the extra characters ignored by View. This is equivalent to having WrapOff when viewing files in Qram or menu\_rext’s VIEW\_FILE This explains why VIEWing a Quill document does not show the full file on screen.

Toolkit Two can View files, with the equivalent of Wrap On, if you usethe SPL command. The SPL command sets up a job toSPooL data from one place to another. The command *SPL filename,SCR* will copy files to the screen. Because of the “non-destructive” windows of the Pointer Interface, the window will disappear when the whole file has been displayed. This is a nuisance if you don’t press Ctrl F5 quickly enough to freezethe screen to stop this from happening. An alternative - *SPL “filename,#1* - will send the file to Superbasic channel 1. You can view many files easily, if you use a combination of the SPLUSE and WCOPY commands.

On TV use:

***SPL\_USE con\_x200   
WCOPY flp1\_***

On a Monitor use:

*SPL\_USE con\_512x256a0x0   
WCOPY flp1\_*

When more than one file is displayed, one after the other, the display can be messy. But the use of WCOPY allows wild- cards to be used and allows simple browsing of a disk or sub-directory.

Note on SPL\_USE: The effects of the SPL\_USE command are permanent until a Reset or a SPL\_USE/DEST\_USE command is executed. Putting a SPL\_USE command in your Boot filecould save you time - you could view files with a simple WCOPY command.

**Copying files on your QL**

This is not thedefinitive guide to copying files but it will “get you started”.

If you have two disk drlves then it is very easy, just put the disk you want to copy from in drlve two and the disk you want to copy to in drive one and type:

*WCOPY* ***flp2\_ TO flp1\_***

This Toolkit Two command will give you a filename and ask ‘..Y/N/A/Q?’. For example:

*flp2\_Boot TO**flp1\_Boot..Y/N/A/Q?*

The action the QL takes depends on your next key-press:

Y or y - to copy the file, ask me about the following files   
A or a - copy this file and ALL the files after it   
N or n - don’t copy the file, ask me about the following files

Q or q - don’t copy the flle, don’t ask me about any more files

If you don’t have a double disk drive, and you have a ramdisk set up, follow these instructions:

Sinclair/QL World February 1993

It you type *DIR RAM1\_* and you get the following two lines:

RAM1   
200/400 sectors

Then you have a ramdisk (ramdisks are standard with Trump Cards and Gold Cards and don’t have to be formatted). The first number refers to the amount of free space left on the ram disk, in units of “half a KB” (the 200 shows that 100KB is free, this is approximately the amount of space on a   
microdrive cartridge). The second number refers to the total amount of space on the ramdisk (simply divide by two to get the capacity in KB - in this case 200KB).

If you don’t get the line ‘RAM1’ followed by a sector count then your ramdisk may need to be FORMATted before it is used. This was necessary with the ramdisk on the Sandy SuperQboard. Enter the command FORMAT RAM1\_200. If you get “not found” or “out of memory” then please contact an experienced friend to help you further.

With a ramdisk ready to go, put the disk you want to copy FROM and type WCOPY flp1\_, ram1\_. Copy the files you want into the ramdisk. Then put the disk you want to copy to into the disk drive and type *WCOPY ram1,flp1\_.* Press A to put a copy of every file present on the ramdisk onto the floppy disk. The files are still on the ramdisk; to get rid of them type *WDEL ram1\_.* It is a pity there is no WMOVE command.

**Creating a Boot disk for your QL**

These instructions will help you set up a Boot disk for the first time and teach simple fault finding.   
If you are using the Pointer Environment it is a good idea to have a single disk to boot from (but keep backup copies of it). This is because the Pointer Environment (ptr\_gen, wman) and related files (hot\_rext, menu\_rext etc) have been improving over time. If you get a new Pointer program that has a newer version of ptr\_gen in it put the newest version on your Boot disk. It is possible that your new Pointer program needs facilities that are only present in the newer files.

Finding out the version of most Pointer Environment files is easy - all you have to do is View them (see ‘VIEWing files” appendix).

The usual way to create a Boot file is to type “lines” into SuperBasic and then SAVE them. A “line” in SuperBasic is different from a “command” because it has a line number in front of it. When a “line” is typed into SuperBasic it stores that line in its memory. Type NEW to clear any old “lines” from SuperBasic’s memory. Enter these lines into SuperBasic:

5 REMark QPAC1 boot file   
10 TK2\_EXT   
20 LRESPR FLP1 \_ptr\_gen   
30 LRESPR FLP1\_wman   
40 LRESPR FLP1\_hot\_rext

Line 5 is a comment line, used to keep notes on the Boot program. Line 10 activates toolkit two. Lines 20 to 40 load the Pointer Environment (ptr\_gen), the Window Manager (wman) and the Hotkey System Two (hot\_rext).

To show all the “lines” that SuperBasic has in memory, type the command LIST. A group of SuperBasic “lines” is called a SuperBasic program and they may be stored on disk by the commands LOAD and SAVE. Try LISTing the lines you have typed in so far. If you have made a mistake then type the command EDIT followed by the number of the line you made the mistake in.

The “lines” entered so far are instructions to load the Pointer Environment. As we have Qpac1, we would like to load those utilities as well. If we were using Qram, we would use the programs Boot\_MAKE and HOT\_MAKE. Being lucky Qpac1 users we get more flexibility and the thrills of typing in our own Boot programs. To complete the boot program, we need to add the commands for using the Opac1 utilities.

50 REMark Load QPAC1 utilities   
60 ERT HOT\_RES(’A’,‘flp1\_alarm)   
65 ERT HOT\_WAKE(’a’,‘aIarm’)  
70 ERT HOT\_RES(‘C’,‘fIp1\_calculator’)   
75 ERT HOT\_WAKE(‘c’,‘calculator’)   
80 ERT HOT\_RES(’K’,‘flp1\_calendar’)   
85 ERT HOT\_WAKE(‘k’,‘flp1\_calendar’)   
90 ERT HOT\_RES(‘D’,‘flp1\_clock’)   
95 ERT HOT\_WAKE(’d’,‘clock’)   
100 ERT HOT\_RES(’M’,‘flp1\_sysmon’)   
105 ERT HOT\_WAKE(’m’,‘sysmon’)   
110 ERT HOT\_RES(‘W’,’flp1\_typer’)   
115 ERT HOT\_WAKE(‘w’,‘typer’)   
120 ERT HOT\_PICK(‘b’,’’)   
130 ERT HOT\_LOAD(‘s’,‘fIp1\_config’)   
140 HOT\_GO   
150 HOT\_DO ‘rn’   
160 PAUSE 50   
170 HOT\_DO ‘b’   
180 HOT\_LIST

When you have typed all these “lines” in, put a blank, formatted disk in drive one and type SAVE FLP1\_Boot. The SAVE and LOAD commands must always be given a file name to tell SuperBasic which file it is to save/load. The Boot file we have saved may be read into SuperBasic again with the command LOAD FLP1\_Boot. When a file is LOADed, the previous SuperBasic program is removed from memory - it is NEWed.

Note - If your QL says “FLP1 \_Boot exists, OK to overwrite.. Y or N?” *press N* and use a disk that really is blank

When your disk drive tight switches off, your QL will have finished saving the Boot file. Now we need to copy the other files onto your disk. Following the instructions “Copying files” in the appendix copy the files ptr\_gen, wman, hot\_rext alarm, calculator, calendar, clock, sysmon, typer and config to the disk you saved our “boot” file to. When the disk drives have stopped whirring, press Reset and F1 /F2 when the prompt appears.

**Simple fault finding in Boot programs**

Some resident programs insist on being loaded before others. The manual with *Lightning* (most editions) insists that it should be loaded before Toolkit Two is activated. By the time my QL gets around to looking for Lightning, *Toolkit Two* is already activated. Nothing happens so I seem to be getting away with it.

When loading the Pointer Ehvironment the Pointer Interface “ptr\_gen” has to be loaded before “wman”. You can load “hot\_rext” whenever you want as far as ptr\_gen and wman are concerned.   
When your QL stops in the middle of Booting and prints:

5 REMark QPAC1 boot file   
10 TK2\_EXT   
20 LRESPR FLP1\_ptr\_gen   
30 LRESPR FLP1\_wman   
40 LRESPR FLP1\_hot\_rext   
50 REMark Load QPAC1 utilities   
60 ERT HOT\_RES(‘A’,‘flp1\_alarm’)   
65 ERT HOT\_WAKE(’a’,‘alarm’)   
70 HIT\_LIST   
At line 70;1

bad name

An error has occurred. The line number will vary and so will the message. (Note - Minerva error messages give a command number as well as a line number, a non-Minerva QL would have given the message At line 70” instead of “At line 70;1. The first thing to do is to see the line that the QL has stopped at The command LIST 70 will list the offending line, in our case:

70 HIT\_LIST

This is a spelling mistake - when writing the line, I was thinking of people who don’t pay their invoices and so quite understandably typed “HIT\_LIST” instead of “HOT\_LIST”. Most “bad name” error messages are caused by bad spelling of command names. Attempting to use toolkit commands before they have been loaded will also give a “bad name” error:

20 LRESPR flp1\_PTR\_GEN   
30 LRESPR flp1\_WMAN   
35 TK2\_EXT   
38 HOT\_LIST   
40 LRESPR flp1\_HOT\_REXT

The above Boot program attempts to use a Toolkit Two command (LRESPR) before Toolkit Two is active (Line 35 should be before line 20). It also tries a Hotkey command before Hotkey Two is loaded (line 38 should be after Line 40).

10 TK2\_EXT   
20 LRESPR flp1\_PTR\_GEN   
30 LRESPR flp1\_WMAN

40 LRESPR flp1\_HOT\_REXT   
50 HOT\_LIST

Another common error is “not found” - this usually occurs when a file to be loaded isn’t on the disk. It could be that you have forgotten to put the file on the disk or you have given your QL the wrong filename.

10 TK2\_EXT  
20 LRESPR fIp1\_POINTER\_GEN   
30 LRESPR flp1\_WINDOW\_MANAGER  
40 LRESPR flp1\_HOTKEY\_TWO

The above program would be understood by a person but a computer likes a bit more   
precision. A QL would just give up with an error message. The QL-friendly version of the above is:

10 TK2\_EXT   
20 LRESPR flp1\_PTR\_GEN   
30 LRESPR flp1\_WMAN   
40 LRESPR flp1\_HOT\_REXT

When you are sorting out a Boot program, when an error occurs, you may want to tell the QL to carry on going, ignoring the current statement

10 TK2\_EXT   
20 LRESPR flp1\_PTR\_GEN   
30 LRESPR flp1\_WMAN   
38 HOT\_LIST   
40 LRESPR flp1\_HOT\_REXT

When the above Boot runs it will give an “At line 38;1 bad name” error message. LIST displays the whole Boot and we see that line 38 is a Hotkey command being used before the Hotkey Two is loaded. To tell the QL to skip this command and carry on running the Boot we type CONTINUE The QL finishes the Boot

To edit the Boot type ED. Move the cursor down to line 38 and delete the 38 by pressing CtrI-Right arrow, press 5 then 0 to replace the 38 with 50 and press Down arrow. The screen should show:

10 TK2\_EXT   
20 LRESPR flp1\_PTR\_GEN   
30 LRESPR fIp1\_WMAN   
38 HOT\_LIST   
40 LRESPR fIp1\_HOT\_REXT   
50 HOT\_LIST

Delete the unwanted line, 38, by moving the cursor over line 38 and pressing Ctrl-Alt-Left

arrow. Press ESCape to leave ED and save the new Boot program with the command SAVE flp1 \_Boot. Your QL should ask “flp1\_boot already exists, OK to overwrite Y or N?”. PressYand wait for the disk drive light to go out before removing the disk or resetting your QL. Full instructions for the ED command are in the Toolkit Two manual (this may be found in the user guide of your expansion card, such as the SuperQboard manual, Trump Card manual, etc).

Sometimes you will want to trya command in a boot program again. Consider the following ‘Boot’ file being run on a disk with the files PTR\_GEN and HOT\_REXT on it.

10 TK2\_EXT   
20 LRESPR flp1\_PTR\_GEN   
30 LRESPR flp1\_WMAN   
40 LRESPR flp1\_HOT\_REXT   
50 HOT\_LIST

The Boot will stop with the error message “At line 30;1 Not found”. I would type DIR flp1\_ to get the list of files present on the disk. Hopefully I would realise that the file WMAN isn’t **on** the disk and put a disk that has WMAN’ on it into drive two, and use the command COPY ’fIp2\_WMAN to flp1\_WMAN. Then, with the problem fixed. I could type RETRY to tell the QL to try the line it stopped at (line 30) and carry **on. As the file ‘**WMAN’ has been copied, the boot will finish with no more errors.

**File lengths and RESPR**

When the QL flrst came out the only way of loadingtoolkits was to allocate some resident procedure space (RESPR). load the toolkit into that space (LBYTES) and initialise it (CALL). This was inconvenient because when a new version of a toolkit was produced, all the boot programs had to be updated toaccommodate thenew size ofthe toolkit.

To load PTR\_GEN 1.54 (file size 14534 bytes) using the oldmethod:

*address=RESPR(14534)**LBYTES fIp1\_PTR\_GEN,address: CALL address*

The Toolkit Two equivalent Is:

LRESPR flp1\_PTR\_GEN

It is only safe to replace the old method of RESPR-LBYTES-CALL with LRESPR if the code actually CALLs “address”. If, in the example above, I had to use CALL address+54, the LRESPR command would not be suitable. Fortunately most programmers make sure their toolkits can be LRESPRed.

**Coping with unexpected “freezes”**

This can be caused by some toolkits clashing or by replacing a CALL-LBYTES-RESPR   
sequence by an inappropriate LRESPR.

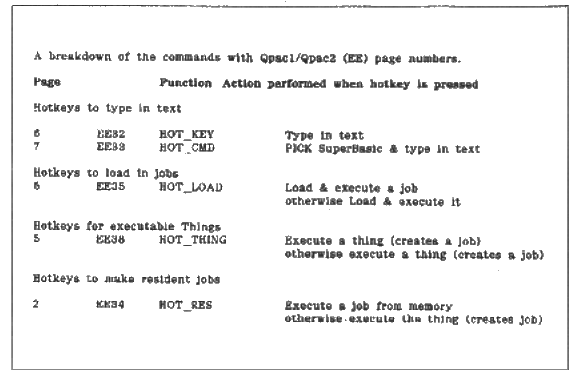
Start tracking down the problem until you know exactly where in the Boot your QL is stopping/crashing. One way to do this is to insert lines into the program which display a message. For example, if WMAN’ had become corrupted (without my knowledge) and this Boot program was crashing my QL for no apparent reason:

10 TK2\_EXT   
20 LRESPR flp1 \_PTR\_GEN   
30 LRESPR flp1\_WMAN   
40 LRESPR flp1\_HOT\_REXT   
50 HOT\_LIST

I would start putting PRINT lines in to indicate the QL’s progress

10 TK2\_EXT   
20 LRESPR flp1\_PTR\_GEN   
25 PRINT ‘ptr\_gen loaded’   
30 LRESPR flp1\_WMAN : PRINT ‘wman loaded’   
40 LRESPR flp1\_HOT\_REXT: PRINT ‘hot\_rext loaded’   
50 HOT\_LIST

The above listing shows line 25 has been added and that lines 30 and 40 have had extra commands added to the end of them The commands are separated by a colon : and the only reason that they are on the same line is to give an example of this. Putting the PRINT commands on a new line or at the end of other lines does not affect the behaviour of this Boot program.



**Hotkeys**

The Hotkey System Two uses the Thing System to keep track of its resident programs. The

Thing system is an extension to Qdos found in ‘hot\_rext’. It acts as a caretaker, keeping track of Things - where they are, what they are, who is using them, etc. QL users don’t need to know much about them. QL programmers should use them more often.

**Hotkeys to make resIdent Jobs**

When the Hotkey is defined, the program referred to will be loaded into memory and linked into the system as an executable thing.

The HOT\_RES, HOT\_RES1 functions will attempt to load the program into the resident procedure area (RESPR space) – if this is not possible (because some jobs are already running), it automatically loads the programs into the common heap. The HOT\_CHP, HOT\_CHP1 functions will load the program into the common heap.

If you weren’t happy with the “Hotkey System 2” manual when you first looked at it go back to look at the example boot programs (page 11) and refer to the table above. It may seem a bit clearer.   
If that fails, have a look at the listing that accompanied my Qpacl article, ***(OL*** *World* March 1991). The listing has some Hotkey definitions and the article gave some explanations.

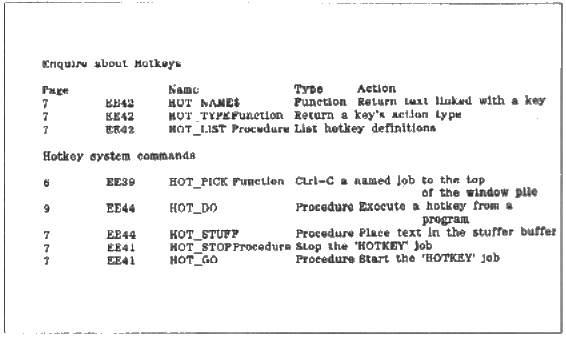
**FILE SELECT - A SIMPLE GUIDE**

The File Select menu, written by Jochen Merz, is part of “menus” extension from the file “menu\_rext’.

**Installing menu\_rext**

When modifying your boot to load menu\_rext, you will need to make sure hot\_rext (the Thing & Hotkey system) is loaded before menu\_rext. This is because menu\_rext links itself into Qdos using the Thing system.

Have a look in your Boot file. If there is a command that loads hot\_rext (usually LRESPR flp1\_hot\_rext) then place the command LRESPR flp1\_menu\_rext after that. If hot\_rext isn’t being loaded by your Boot file, change it so it does - menu\_rext can’t work without it



**Using File\_Select**

Move the pointer over the window – notice how items are highlighted as the pointer move sover them. This highlighting shows the user different options that they can trigger.

.   
The file select menu has three main areas - “file name”, “device name” and “file list”.

The “file name” area, just below the title “Select File Name”, will allow you to enter the file name in full. There are two other options - Current and Previous. This refers to the Hotkey buffer (Alt-Space) which is set by some programs, usually to the name of a file being worked on. This “stuffing” of file names saves the user having to remember and type file names.

The “device/directory name” area, at th ebottom left of the “file area”, allows you to change the drive whose files are listed in the “file list”. The current device can be changed to (MDV, FLP etc) and FILE\_SELECT is intelligent. It searches the system tables for device names so even if you added a laser disk to your QL (LAS1\_? OPT1\_? CD1\_?), FILE\_SELECT would be able to handle it. The device/directory name may be Hit to edit it via the keyboard or it may be Done to pull down a Select Directory menu. Some of the facilities it offers are duplicated by FILE\_SELECT – its unique options are “Data default” and a “common directory list”. The “Data default” options reads the current Toolkit 2 data default so that it can be edited or approved by pressing ‘OK’. The “common directory list” may be tailored to the individual user’s needs by using config on menu\_rext.

The “file list” area, at the bottom right of the “file area”, is a menu of file names. Doing a file name quits the FILE\_SELECT menu and marks that file name as “chosen”. Hitting that filename puts it in the file name buffer where it can be edited or chosen by Hitting/Doing “OK’. The “file list” has some controls to make life easier.

The “EXT” control will allow the user to set a file extension so that only files ending with a particular file extension will be listed. Hitting EXT (or pressing E) will allow keyboard entry of a new file extension. Doing EXT will pull down a menu of common file extensions. This menu of file extensions may be tailored to the individual users needs by using config on menu\_rext.

The ‘View” control, a recent improvement to File Select is normally switched off. When View is switched on (by Hitting it) then the handling of the file menu is subtly changed. Doing a file in the file menu will still chose that particular file. But Hitting a file will now ask your QL to display the file. The View screen has a few simple commands - ESCape (Quit viewing), Wake (go to start) and Wrap. Wrap is usual[y switched off. With Wrap switched off, only the first part of long text lines will be displayed. With WRAP switched on, long lines will wrap around onto the next display line. It is best to view Quill documents, program files, any binary data with Wrap switched on.

The Directory Up and Tree controls deal with “level-2 device dirivers” where “hard subdirectories” are being used - mainly of interest to Atari QL or Gold Card QL users. The Tree control, when switched on, will look for directories below the current directory (sub-directories). Then the names of files found in these sub-directories will be placed in the file menu. Switching Tree off will cause all the files from sub-directories to be removed from the file list. The Directory Up control is only of use in a sub-directory. It tells the file menu to get its file names from the “parent” directory of the current directory.

**Sources of QL Expansion/related items**

*Miracle Systems Ltd   
25 Broughton Way   
Osbaldwick   
York YO1 3BG   
0904 423986*

Gold Card system expansion £225: all Trump Card facilities or better, 1920KB memory, QL 5x faster, plus DEV device, Hard sub-directories.

Dual ED disk drives £175. Can handle 720KB DD, 1.44MB HD and 3200KB ED 3.5in disks. Intended mainly for Gold Card

*Qubbesoft PD   
38 Brunwin Road   
Braintree   
Essex CM7 5BU   
0376 347852*

*Trump Card* £95.00 plus p&p. 768KB memory for a total of 896KB, Toolkit Two, screen   
dump, printer buffer, dynamic ram disk disk interlace.

*Expanderam* 512K £45 plus p&p. Expands QL’s memory to 512+128 giving 640KB. The drawback is that another interface is needed if disk drives *are* to be used.

*Dual 720KB 3.5-in disk dnves* £100 plus p&p.

See their regular adverts in QL World for more details.

*EEC (W.N. Richardson* & *Co.)   
18-21 Misbourne House   
Chiltern Hill   
Chalfont St Peter   
SL9 9UE   
0753 888866*

Various types of *Universal dlsk drives* (suitable for use on many computers). £25-£130.   
*OL serial mouse* (three buttons) £45. In this almost complete package you get some driver software (a file called E\_SERMOUSE), a mouse, a cable, instructions and a mouse mat. The only missing component is a copy of PTR\_GEN. PTR\_GEN is included with all commercial Pointer programs these days.

*Mouse suitable for QIMI £10*

*Quanta   
c/o Bill Newell*

*213 Manor Road*

*Benfleet   
Essex SS7 4JD*

*QIMI mouseinterface* £27   
*Motorola 68000 reference* £6   
*Definitive Guide to SuperBasic* by Jan Jones £10   
*Membership to Quanta* is by annual subscription

Various second hand QL systems - Caveat emptor and vendor!

*Dilwyn Jones Computing   
41 Bro Emrys   
Tal-y-Bont   
Bangor   
GyneddLL57 3YT   
0248 354023*

Main UK dealer for *Qjump software   
Opac1* £19.95, Qpac2 £29.50 Other pointer programs sold include *Disa Disassembler* (intelligent machine code tool) £29.00 and *Data Design* (database) £50

See adverts in QL World.

*Pointer Products   
68 King Down Rd   
Blandford  
Dorset DT11 8BN   
Sells many different Pointer Progiams   
0258 455117*

*Jochen Merz Software, lm Stillen Winkel 12, D-4100 Duisberg 11 Germany   
0203 591706*

A bulletin board ‘QdosMailbox’ service is planned to allow registered users to update

easily. Main dealer tor many German programs - English manuals provided, and English versions of German programs. Can accept UK cheques. Also sells some QL hardware and QL emulators for the Atari ST. See adverts in QL World. Recent releases include:

*EasyPtr3* - a pointer program development tool. The new version provides support for C programmers.

*Qspreadspread* sheet This has been revised and has a new manual

*QVME QL emulator for the Atan STE*