RAM based Utilities

For the Sinclair QL

You are provided with two Microdrive cartridges containing various files for instant use and to configure the system as you prefer. To get a feel for the QRAM system, put the "Qram Master" cartridge in MDV1_ and restart the QL, as instructed in the manual's introduction.

The first thing to try here is to use the FILES menu to back up both cartridges! First press ALT-/ to bring up the QRAM main menu (see p9). Put a newly formatted cartridge in MDV1_, and the one you want to back up in MDV2_, and select the FILES option by pressing "F" or moving the arrow to the FILES option (it will acquire a white border) and pressing SPACE or ENTER. The Microdrives will spin, and presently the FILES menu (see pp10-12) will appear with a list of the files on the cartridge in MDV2_. Now select ALL the files by pressing "A", and the ALL item and all the files will change to white on red, showing that they are selected. Finally, press "C" to copy all the selected files to the new cartridge. If any files fail to copy, they will be left selected (white on red) - otherwise they are re-drawn in white on black. To get rid of the FILES menu, press ESC: to copy the other cartridge, put it in MDV2_ and another empty cartridge in MDV1_, press "R" to read the directory, and press "C" again. As you left ALL selected, all the files will already be selected and you don't need to press "A" again. To leave the FILES menu, press ESC, and it will disappear.

This all looks complex when typed out, but all you've typed is:

ALT-/	pop-up QRAM
F	select FILES option
A	select all files
C	copy all selected files
R	read new medium
C	and copy it
ESC	exit the FILES menu

The second cartridge "Qram Util" contains the BOOT_MAKE and HOT_MAKE utilities (see pp21 and 17 of the manual) and their associated files, plus the three application processing utilities GRABBER, UNLOCK and GUARDIAN (see pp19/20 of the manual). There are two copies of each of these files, the second one being a SPARE_ in case the first is (or becomes) unreadable.

NOTE that the PTR_KBD file mentioned in the manual is NOT supplied: this is because the Internal Mouse Interface software can spot that the hardware isn't there and take appropriate action, and so can be used with keyboard-only QLs as well as those with the Internal Mouse Interface. Nor is the Hotkey file provided as this is incorporated into the BOOT_REXT and can be made using the HOT_MAKE utility.

Filename	Copies on: Master	Util
BOOT	1	
BOOT_REXT	1	
PTR_IMI	1	1
WMAN	1	1
RAMPRT	1	1
QRAM	1	1
QRAM_HELP	1	1
BOOT_MAKE		2
HOT_MAKE		2
GRABBER		2
GUARDIAN		2
UNLOCK		2

You are only given one copy of BOOT and BOOT_REXT, since you can reconstruct these files, using BOOT_MAKE, if necessary.

This is Version 1 of The QRAM Utilities

It is now 1987 and just over 6 months late we have finally persuaded the boys in the back room to release the first real version of the QRAM utilities package. This now has all the facilities originally planned, together with a few extra bits that we have found necessary. These few extra bits have transformed the package and we now believe that this is a package that all users of expanded QLs should have.

The QRAM utilities package is in active development. It is the policy of QJUMP Ltd to provide free upgrades to purchasers who find that the product is not usable due to errors. Futhermore we offer low cost upgrades to those users who wish to make use of more advanced facilities as these become available.

A few words of warning: the QRAM utilities package operates in a completely new type of QL environment. It is probable that there will be a number of commercially available programs which may not be able to operate within this environment. This is a problem we will not be able to cure in all cases.

Although great care has been taken in the preparation of this product, QJUMP Ltd will not be liable for any direct, indirect or consequential damage or loss which may arise from any error, defect or failure of this product.

QJUMP Ltd, 24 King Street, Rampton, Cambs. CB4 4QD

CONTENTS

Introduction 1
Hints 1
The QL Pointer Interface 2
The QL Pointer Interface and Standard Programs 3
The Window Manager 5

QRAM Files 6
Boot File 6
Resident Extensions File 6
QL Pointer Interface File 6
Window Manager File 6
Hotkey File 6
RAM Disk and Printer Buffer File 7
Making New Boot Files 7

QRAM and how to use it 9
Main Menu 9
Files Menu 10
Format Sub-Menu 12
Sort Sub-Menu 12
Jobs Menu 13
Channels Menu 13
Print Menu 14
Window Dump Menu 15
Options Menu 16

Additional Utility Programs 17
Hotkey Creation: HOT_MAKE 17
Memory Control: GRABBER 18
Unlockable Windows: UNLOCK 19
Window control: GUARDIAN 20
Boot File Creation: BOOT_MAKE 20

RAM Disk Driver and Printer Buffer 22
RAM Disks for Beginners 22
RAM Disk Creation 23
Heap Fragmentation 23
Microdrive Emulation 23
Microdrive Imaging 24
Print Buffering 24

Appendix 1 - Screen Dump Formats 25

Introduction

If, like us, you find manuals difficult to read and understand, just reset your QL and restart it with the QRAM disk or cartridge in drive 1. When you want to try QRAM, press ALT and / together (hold ALT down and press /). You can move the pointer around with the cursor keys (or mouse), select items with the SPACE bar (or left button) and DO things with the ENTER key (or right button). Remember F1 for HELP and ESC to give up. As a last resort, read this manual.

The QRAM utilities package is in many parts. There is the QRAM maintenance program itself, plus one of the fastest and most flexible QL RAM Disks available, print buffering and spooling facilities, hotkey program execution and, above all, the new QL Pointer Interface from QJUMP.

This manual is also in many parts. There is this introduction to the package, to the QL Pointer Interface and to the Window Manager. The second part gives a description of all the files supplied with the package and some hints on writing your own BOOT files. The third part describes the QRAM Utility itself. Each menu within QRAM is described in full. The fourth part introduces the RAM Disk and printer buffer. The fifth part describes each of the utility programs supplied with the package.

All of the facilities in the QRAM utilities package are intended to make the best use of the RAM expansion units available for the QL. The minimum memory configuration that is usable with the package is 256k bytes, but at least 384k bytes (QL+256k) is required for effective use. We work with 640k byte QL configurations and, with the QRAM package, this is well worth the extra cost.

Hints

Any job which covers the whole screen makes it difficult to pick a window beneath the job. This is the normal state for SuperBASIC. At the start of our own BOOT files we have the following commands to leave the top of the screen empty:

```
WINDOW #1;256,182,256,26:BORDER #1;1,255
WINDOW #2;256,182,0,26:BORDER #2;1,255
MODE 512
```

If you are writing programs to work with the QL Pointer Interface, then you should ensure that you have a window which encloses the whole area used by the program. When we use compiled SuperBASIC, we usually redefine window #2 to enclose the area occupied by window #0 (at the bottom of #2) and window #1 (above #0). If you wish to reduce the area occupied by your program, then this should be done by closing the windows no longer required, and opening new ones. This will ensure that the screen stays tidy.

As programs will normally vanish without trace (c.f. Unlock utility) when a job terminates. It is a good idea to pause and wait for keyboard input at the end of a program, or else BEEP and pause. This gives the program user (yourself) a chance to see what has happened.

The QL Pointer Interface

The QRAM package is underpinned by the QL Pointer Interface. It is this interface which handles "pointer directed" (e.g. mouse or cursor key) input and maintains the windows for a job. This interface is fundamental to the operation of the QRAM program and for other "hotkey" programs which use pull down windows. For a proper understanding of the potential of the QRAM utilities package, and to understand why some programs may not work within the pointer environment, it is necessary to understand the workings of the QL Pointer Interface. To those who just wish to use the package and are fortunate enough to use only those programs which work within the pointer environment, we apologise for the following discussion.

The QL Pointer Interface contains the code which turns the QL screen driver inside out. There are two parts to this code: the first handles the pointer, the second handles the windows.

The pointer appears on the screen as required, it may be moved around the screen using either the mouse or the cursor keys. Its appearance depends on circumstances. Normally it will be an arrow or an application defined sprite, but when it is over a window which is not expecting pointer input, it takes on an appropriate appearance:

normal

the job is expecting keyboard input

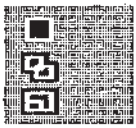
the window is "locked"

the window (invisible) is in mode 4 or 8

there is no window beneath the pointer

the job is not able to take input (NO ENTRY)

In addition there are three special sprites which will remain constant as the pointer is moved over the display:



identify window request

move window request

change window size

If the pointer is not visible, it may be woken up by moving the mouse or pressing CTRL-C.

The code in the pointer interface handles overlapping windows by "locking" a window which is below another window and so preventing the lower window from being modified. The windows form a pile (somewhere between a heap and a stack). Any window which has any part of the window visible, may be brought to the top by moving the pointer over the window and, if it is locked, "hitting" it (pressing a button on a mouse or the space bar or enter key). Alternatively, the bottom window in the pile may be brought to the top by pressing CTRL-C. If the top window is waiting for keyboard input then it will grab the keyboard queue and kill the pointer.

A locked window has its contents saved so that it may be restored when it is brought to the top. If the operation to save the window fails because there is insufficient memory, the QL will make a rude noise. This is a warning to the user to do something to free some memory. We strongly recommend that you do not operate the QL Pointer Interface with inadequate memory.

The code in the pointer interface can even cope with jobs executing in different display modes. However, only one mode is visible at a time. When a window in a different mode is brought to the top, the display mode is changed and the entire screen is regenerated from the bottommost window in the pile upwards.

Unlike most overlapping window systems the QL Pointer Interface does not save the area under a job's windows. This is because any job may be lifted out of the pile and brought to the top, thus changing what is underneath another job's windows. When a job is removed, the area underneath the job's windows is recreated from the saved contents of all the bits of all the other jobs' windows which overlap the area. The operation is done in such a way that even windows which are partly obscured may be safely removed.

The operation of removing a job's windows is usually very clean as the part of the display affected is recreated in free memory before being copied back into the display memory. If, however, there is no free memory, then the screen will be rebuilt in place. When there are 20 or 30 jobs overlapping the area of the display being recreated, the effect can be quite dramatic.

The QL's display memory occupies 32k bytes of memory. For each job which owns and uses windows within the pointer environment, there may be a window save area whose size depends on the size of the window. If there four jobs (including SuperBASIC) each occupying the whole display, then just over 128k bytes of memory will be used for save areas.

The QL Pointer Interface and Standard QL Programs.

The QRAM utilities program uses special facilities provided in the QL Pointer Interface to maintain resizable and moveable non-destructive windows. The really hard bit was to ensure that most programs already written for the QL also operated within non-destructive windows. As the QL provides destructive windows that may be redefined at will, without restoring the screen under windows that have moved, there is a certain amount of conflict between the old and the new. This conflict is resolved by the following compromise.

The QL Pointer Interface resolves the contention between different jobs accessing the display by using the jobs' "outlines" to determine the extent of the windows' overlap. For programs which do not use the extra window maintenance calls in the QL Pointer Interface, the outline is the smallest rectangular area which will just enclose all the windows. As windows are opened or closed, or their sizes are reset, the outline is modified. There is a difference between closing a window and resetting the size. When a window is closed, reducing the job's outline, the QL Pointer Interface restores the area of the display which has been freed. If, however, the job's outline is reduced by resetting the size of a window, the freed area is not restored.

There are five main reasons for an "old" QL program misbehaving when used with the new environment:

repeated console open and close operations,
repeated window size redefinition,
reliance on the QL's destructive windows,
abuse of operating system facilities and
modification of the console or keyboard drivers.

Repeated console open and close operations are a waste of time on a standard QL, and, at best, a greater waste of time using the pointer interface. At worst, the display could start to flicker in a very annoying manner.

Repeated window size redefinition is sometimes used on a standard QL to access different parts of the screen by moving the window around. Within the pointer environment, the uncontrolled changes of outline will result in the program behaving in a very similar way to the execution on a standard QL. There will only be partial window saves and restores. The problem can be cured by providing a "guardian" window which covers the whole of the display that the job will use. This will fix the outline to be the same as the guardian. A utility (GUARDIAN) is provided to attach a guardian window to an executable job.

Reliance on the QL's destructive windows is built into some programs (e.g. various clocks). A facility is included in the pointer interface to provide permanently unlocked windows. This facility may be added to existing programs with the UNLOCK utility.

Abuse of the operating system facilities is a common problem. There are three abuses which can cause problems. The first is the hostile way in which some programs grab all the available memory for themselves. This means that there is no memory left for any other jobs, or even for a window save area. A utility (GRABBER) is provided which can outwit most of these programs. The second abuse is the use of unnecessary MODE calls. The best protection against this disease is the use of a guardian window covering the whole screen. The third abuse is accessing machine resources directly rather than through the operating system. This is usually limited to writing directly to the screen memory. There is very little that can be done about this except to complain to the software supplier.

Modification of the console or keyboard drivers is likely to conflict with the QL Pointer Interface's own modifications. There should be no problem with legitimate interception of the keyboard queue, or the use of console "EXTOP"s. There is no conflict with the SuperToolkit II ALT key definition.

The Window Manager

The QL Pointer Interface handles the on-screen pointer and maintains the windows. The QRAM utilities program uses a standardised window management system for its static, scrollable and pull down menus. This is the Window Manager.

The Window Manager routines perform all the menu handling tasks for programs such as QRAM. As yet incomplete, details of the interface to these routines are available from QJUMP. These routines handle selection of items by keystroke as well as pointer. Items can be text or arbitrary objects and menus can be dynamically sized.

The Window Manager handles multiple window menus. Menu selections may be made either with a mouse, or by moving the pointer with the cursor keys, or by typing a single letter on the keyboard.

The Window Manager distinguishes between "loose menu items" which float around the job's main window, and "sub-window menus" which are neatly enclosed groups of menu items. Loose menu items can be selected by single keystroke from anywhere within the job's main window, while items within a sub-window menu may only be selected when the pointer is in that sub-window. The keystrokes for selecting standard loose menu items are:

ESC	quits from the current menu, without doing anything,
F1	brings up the appropriate help text and
CTRL-F4	brings up the MOVE WINDOW sprite to allow you to move the menu around.

Keystrokes for selecting both loose menu items and sub-window menu items are:

SPACE	(HIT-left button) toggles the current item,
ENTER	(DO-right button) selects the current item, and does any selected function and
other	toggles, selects or does the function beginning with the character (e.g. J for JOBS).

The Window Manager does not distinguish between upper case (e.g. "J") and lower case (e.g. "j") letters.

The "current item" is the menu item which encloses the current pointer position. The current item is outlined by the Window Manager and as the pointer is moved, the outline will follow.

Where a list of items will not fit in a sub-window all at once, scroll bars will appear at top and bottom: hitting these will scroll the menu. Alternatively, ALT-UP or ALT-DOWN may be used to scroll a sub-window.

QRAM Files

The QRAM package comes on a floppy disk or Microdrive cartridge. The basic part of the package is in just two files: BOOT and BOOT_REXT. The first of these is a simple boot file to load the extensions in the file BOOT_REXT. The BOOT_REXT file supplied with QRAM comprises one of the QL Pointer Interface files, the Window Manager file, the Hotkey file and the RAM disk file. These files have been put together by the utility BOOT_MAKE.

Boot File

The boot file has the code required to set up the QL Pointer Interface, the Window Manager and the Hotkey. This file assumes that you do not have SuperToolkit II available. If you don't, you should have. QRAM and SuperToolkit II are complementary packages and with the two together you have a QL system of a totally different kind.

Resident Extensions File

The resident extensions file BOOT_REXT has all the extension files required concatenated into one file. This improves the loading speed, particularily for Microdrives.

QL Pointer Interface File

Unless you have a Sandy SuperQBoard Mouse, QRAM will have been delivered with a file (PTR_KBD) which contains the keyboard version of the QL Pointer Interface and a file (PTR_IMI) which contains the Internal Mouse Interface version of the QL Pointer Interface. One of these files should be loaded into the resident procedure area *before* any other pointer related files, but *after* SuperToolkit II.

If you have a Sandy SuperQBoard Mouse, then the QL Pointer Interface is built into the SuperQBoard. In this case the QL Pointer Interface is installed by the command "POINTER".

Window Manager File

The Window Manager file (WMAN) should be loaded into the resident procedure area after the QL Pointer Interface has been installed. At present the Window Manager is only required for QRAM.

Hotkey File

A Hotkey file is used by loading the file into the resident procedure area of the QL (usually when the QL is reset), and then typing the command "HOTKEY". This will start up a job called "Hotkey" which in turn will start up other jobs when specified ALT key presses are detected. (The ALT key should be held down and then the specified key pressed.) The normal ALT key for QRAM is "/".

ALT space is reserved for copying a special Hotkey buffer into the keyboard queue. The Hotkey buffer may be filled by any program. In particular, the QRAM program fills the buffer whenever a file is selected (or viewed). This means that QRAM can be used to find a file and then, after you have quit from QRAM, the filename can be copied into the current keyboard queue.

The Hotkey program is started up by a command so that there are no jobs executing while you are loading other extensions into the resident procedure area. The Hotkey program may be removed by a SuperToolkit II command (RJOB 'hotkey') or by using the JOBS menu of QRAM. It may be restarted by giving another HOTKEY command.

The QRAM package includes a file "HOTKEY" with a Hotkey version of QRAM, and a utility program HOT_MAKE for extending this file or making a new one.

RAM Disk and Printer Buffer File

The RAM Disk and the dynamic printer buffer are both in one file. The RAM disk can be used unformatted (with dynamic allocation of memory) or formatted (with static allocation of memory). There is a fast Microdrive load option which creates a complete image of a Microdrive cartridge in about 9 seconds. The printer provides a dynamic buffer between a job and any normal output device (e.g. SER or PAR).

The file RAMPRT contains both RAM Disk and Printer Buffer and adds the following commands to SuperBASIC:

RAM_USE *name*	sets the RAM Disk name
PRT_USE *usage,device*	sets printer buffer usage name and printer device name
PRT_ABT	aborts current printer output

The PRT_USE and PRT_ABT operations are also available from QRAM using the "PRINT" menu.

Making New Boot Files

The QL's system software was designed to be extended. The extensions are usually added to the system software by a "BOOT" file which is loaded and run when the QL is turned on or reset.

Simple boot files may be created using the utility BOOT_MAKE. This program enables you to link together many extension files, so that they or all loaded at the same time. If this is what you wish to do, then refer to the part of the manual dealing with the utility programs.

If you wish to keep all your extension files separate, it is still convenient to assemble all the commands to load the extensions you require into just one boot file. It is easiest to do this if you have SuperToolkit II available.

If you have SuperToolkit II on a Sandy SuperQBoard, start your boot file with the command TK2_EXT, if you have a configurable SuperToolkit II start the file with the commands from the SuperToolkit II boot file created by the SuperToolkit II configuration program. If you have a ROM cartridge SuperToolkit II, just carry on as follows.

Next put an LRESPR command for the QL Pointer Interface or the pointer command POINTER. Then put LRESPR commands for the Window Manager and Hotkey files. If you wish, put an LRESPR for the RAM Disk and Printer Buffer file. Follow this with any further LRESPR commands for any other extensions. Next put the HOTKEY command. Finally, put any code you wish to start programs, set the clock, define the printer buffer etc.

```
LRESPR flp1_ptr_kbd     :REMark - or POINTER
LRESPR flp1_wman
LRESPR flp1_hotkey
LRESPR flp1_ramprt
......                  :REMark - more LRESPRs
HOTKEY                  :REMark - start up Hotkey
PRT_USE ser,ser         :REMark - buffer all SER output
......                  :REMark - and so on
```

Note that, on JM and AH versions of SuperBASIC, you may not normally include commands in the same file that loads the extension which creates them. This is not true of SuperToolkit II, Hotkey and the RAMPRT extensions which have special precautions to overcome this problem.

If you do not have SuperToolkit II, then, in place of the LRESPRs, you will need to use the normal resident procedure load sequence:

base=RESPR(*size*): LBYTES *name*, base: CALL base

The size of most of the files can be found from their normal boot files. If, however, you have extended the Hotkey file, you will need to use the FILES menu of QRAM to find the size.

QRAM and how to use it

Throughout this description, disk includes microdrive, the singular includes the plural, masculine includes the feminine etc. etc..

QRAM is a utility program for job and file manipulation: it may be executed directly from disk or from a hotkey. Hotkey execution is preferred, since while it always uses memory, you can pop QRAM up wherever you are, even in the middle of Quill. You can pop up as many copies of QRAM as you wish. We find it particularly useful to leave a copy of the "FILES" menu lying around on the screen to "VIEW" files as required. If it gets in the way, we move it or bury it.

The "current item", in any menu, will be outlined in white. An item that has been selected appears in white on red, and items that are available in white on black. If an item becomes unavailable for some reason (it may be a file you've deleted), then this will appear in red on black.

A "hit" on an item is achieved by moving the pointer to the item using the mouse or the cursor keys, and pressing the space bar or the left mouse button. A "do" is achieved by hitting the enter key or right mouse button. A hit will toggle an item's status between available and selected. A "do" will select it, and usually invoke any associated function, such as file copy.

A "do" on an item may also be effected by typing a single character, usually the the first letter (e.g. "J" for JOBS): QUIT breaks this rule, it uses ESCape. Some items may not have this feature – filenames and the FORMAT item are examples.

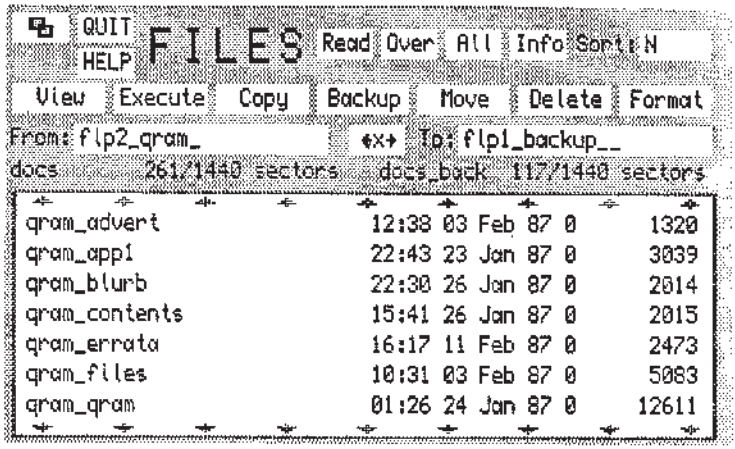
If there are too many items to fit within a sub-window (e.g. too many files in a directory), then arrows will appear at the top and bottom of the sub-window. Hitting the arrows will scroll the menu an item at a time. Alternatively, ALT-UP and ALT-DOWN may be used to scroll the menu.

Main Menu

Simple this: to access any of the QRAM menus point to the appropriate one and hit it. If you are lost, hit the HELP item. If you started QRAM by mistake, hit QUIT. Quit will usually abort the current menu without altering anything, wherever you are. You could of course type "F","J","C","P","W" or "O" to select a menu, "F1" for HELP, "ESC" to QUIT or "CTRL-F4" to move, but this requires typing skills which are currently out of fashion. As there is nothing to select in this menu except actions or other menus, hitting an item will activate it immediately.

FILES Menu

The FILES menu is provided to allow all the normal file maintenance operations required. It is based on directories rather than devices and supports two directories: the FROM (or source) directory and the TO (or destination) directory. A directory is a combination of a device and one or more names which will select just a small number of files on a device rather than all of them. In this example the FROM directory is "qram_" on device FLP1_. The TO directory is "backup_qram_" on device FLP2_.

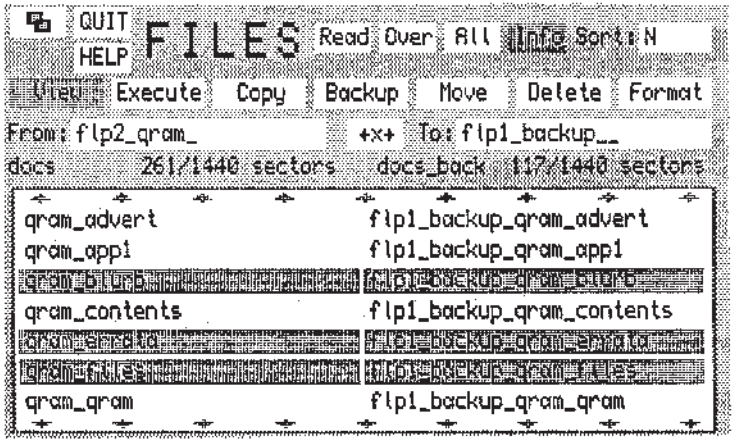


There are four groups of operations outside the main display window. On the left, there are the basic MOVE, HELP and QUIT actions. On the top at the right there are the status items READ, OVER, ALL, INFO and SORT. Below all of these are the actions VIEW, EXECUTE, COPY, BACKUP, MOVE, DELETE and FORMAT: as you go to the right, they become more dangerous. Below this there is the directory control FROM, Xchange and TO. The more dangerous actions and options (Delete, Format and Over) cancel themselves after use.

The big window contains details of the files on the source device, with either their date, type and length

You can change the source or destination pattern by hitting the appropriate item and editing the string - these patterns are used to select which files will be shown and where they will copy to. This allows the display of "directories" rather than whole disks or Microdrives. The wild card algorithm used is the same as SuperToolkit II (FLP1_FRED__DOC = all files starting with FLP1_FRED, followed by anything, followed by _DOC), try experimenting!

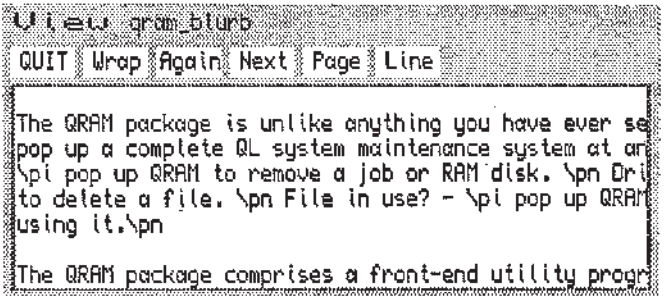
If you hit the INFO item you will change the display to show the file on the destination device where the corresponding source file would be sent by COPY, BACKUP or MOVE. In this example the files are to be copied down a directory level and three files have been selected to be viewed.



When copying files the destination file name is derived from the actual source filename, the "FROM" name and the "TO" name. Each file name is divided into sections separated by underscores. If there are fewer sections in the "TO" name than in the "FROM" name, then the destination file name will be shorter than the source file name and vice versa. This enables files to be copied up a directory level, or down a directory level. For normal copying there should be the same number of sections in the "FROM" and "TO" names.

You can re-read the directories by hitting READ, and select/deselect all files by hitting ALL. The funny item between the TO and the FROM (X) will exchange them and re-read.

VIEW enables you to view all the files so far selected (one at a time!) a line or page at a time.



WRAP selects whether long lines will be truncated or wrapped. AGAIN goes back to the beginning of the file. NEXT goes to the next selected file. PAGE displays another page of the file, LINE another line. Every time a file is VIEWed, the file name is put into the Hotkey buffer and becomes accessible with ALT-SPACE.

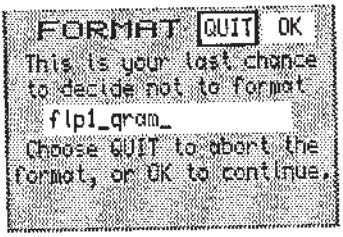
EXECUTE and DELETE should be fairly obvious - they operate only on the source disk. COPY and BACKUP are very similar, but where COPY just copies all selected files, BACKUP only copies those which are more recent on the source medium than on the destination. BACKUP also renames the destination file to <filename>_OLD before doing the copy, so if something goes horribly wrong then you shouldn't lose any files. Once BACKUP has successfully done the copy, it deletes the _OLD file. If it can't rename to _OLD for some reason, it will just overwrite the file IF the OVER item is selected. Similarly, COPY will overwrite only if OVER is selected.

Files which should have been copied, but weren't for some reason, are left selected - the rest are made available.

MOVE either does a rename, if source and destination are the same medium, or copies the file, using the rules above, and if successful deletes it from the source. MOVEd and DELETEd files are made unavailable, since they no longer exist under the name in the menu.

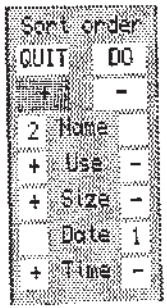
The QL's own microdrive code does not support the file "overwrite" and "rename" operations, nor are the files themselves date stamped. Toolkit II from Care/Qjump adds all of these operations. If, however, you do not have Toolkit II, then QRAM will OVERwrite files by deleting the old file and then copying, and it will MOVE a file by copying and deleting the old file.

FORMAT Sub-Menu



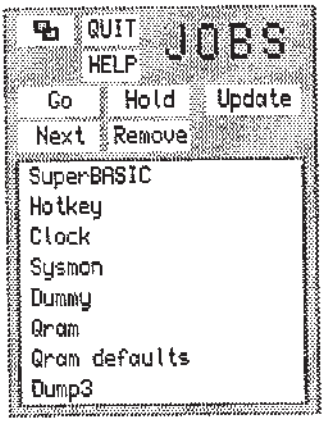
FORMAT pulls down a little sub-menu, and allows you to format the destination - you can edit the given name, as with the source and destination. A "do" on OK or in an empty space will format the medium, and a message will appear telling you whether it succeeded or failed. QUIT from the format menu just goes back to the FILES menu. For security, the pointer will always appear in the QUIT item.

SORT Sub-Menu



SORT also pulls down a sub-menu, and allows you to change the order in which the source files appear. Either hit the + or - beside the key you want to sort on, or select + or - and type the appropriate character. The selected item will become unavailable, and show a number from 1-5 showing which key field is the most important. You can sort first by use, and then by name, for instance, to group all executable files (use code 1) together. The + symbol selects a sort in ascending order, - in descending. A summary of the current sort order appears in the FILES menu, with upper case letters denoting ascending order.

JOBS Menu



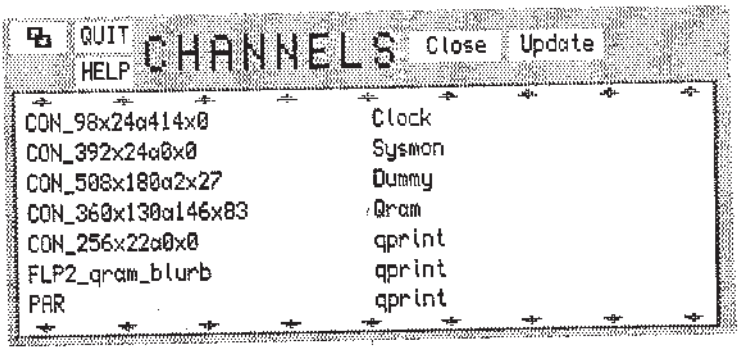
Almost as simple as the main menu. Hitting a job will select or deselect it, but if it has gone away since the Jobs menu appeared then it will become unavailable. A "do" on a job selects the job and invokes REMOVE, HOLD, GO or NEXT if one of these is selected.

Hitting one of REMOVE, HOLD, GO or NEXT will toggle it, and possibly deselect any which is already selected - you can only do one at a time. A "do" on one of them will select it and do the operation to all selected jobs. Pressing "R", "N", "G" or "N" will have much the same effect.

If the operation invoked by a "do" is successful, QRAM will remove itself, and you will have to re-execute it to perform another operation on jobs.

The operation may be unsuccessful if you tried to do something with no jobs selected or only to QRAM itself. Removing SuperBASIC also fails, as does HOLDing or GOing jobs which are already in this state. However, if the operation succeeds on just one of the selected jobs, then the operation is assumed to be successful.

CHANNELS Menu

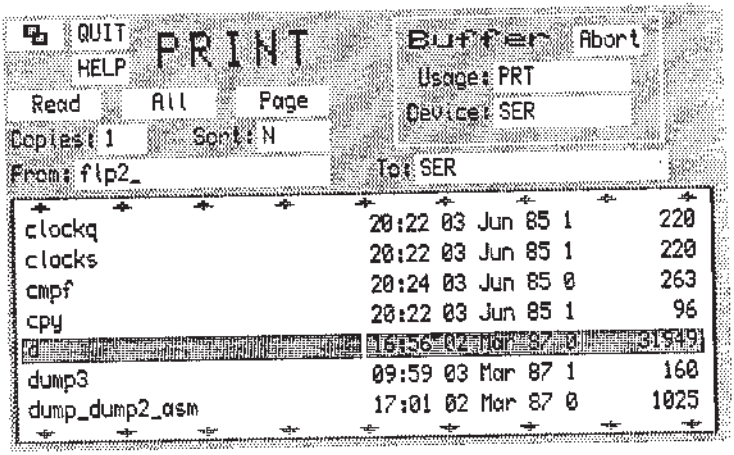


The CHANNELS menu gives you a list of the channels currently open in the the machine, sorted according to which job owns them. You can thus find out which job is hogging the serial port, or a file. If you must, you can close the channel directly, although this isn't usually a good idea unless the job is designed to cope with this. The UPDATE function re-reads the channel table, in case channels have been opened or closed since the menu was invoked.

If you wish to cancel a file which has been "spooled" from the PRINTER menu, then you should use the CHANNELS menu to close the file.

PRINT Menu

The PRINT menu gives access to two sets of facilities: most of the menu is used to access a SPOOLer to copy files to a printer using background jobs. This is fairly cheap on memory, but, as long as the file is being printed, the disk or cartridge may not be removed.



The top right corner of the PRINT menu controls the use of the printer buffer. This is a facility which speeds printing by inserting a dynamically allocated buffer into an output channel.

You can specify the name that will be USED as a buffered output file, and the DEVICE that will be used for printing. The simple case of specifying the name "SER" for both the USAGE and the DEVICE, will insert a dynamic printer buffer into the output channel of any job sending output to a SER device. This can be used to speed up the operation of the PSION programs as well as any other printing programs as the program will not need to wait until printing has finished before continuing. For more details, see the RAM Disk and printer buffer section of the manual.

ABORT is used to terminate the file currently printing from the printer buffer. The ABORTed file is terminated with the message

"******* ABORTED *******".

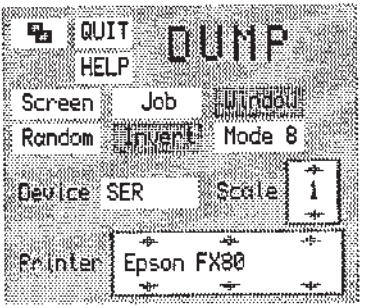
The rest of the PRINT menu is used to select files for spooling to a printer. The number of COPIES may be selected and a new PAGE (form feed) put at the end of each copy. The printer TO device may be a buffered printer but, as this will tend to fill up the memory, it is not recommended unless you wish to free the disk or cartridge.

If you wish to terminate a file which has been "spooled" from the SPOOL menu, then you should CLOSE the source file in the CHANNELS menu rather than REMOVing the SPOOL job from the JOBS menu. A file ABORTED by closing the source file will be terminated with the message

"******* SPOOL ABORTED *******".

Background printing can be achieved from programs such as Quill in one of two ways. If Quill would normally print to a SER device (the default for Quill is SER1HR) then define both the USAGE and the DEVICE to be SER. Printing from Quill will now be buffered in the QL's memory. The second way uses less memory, but is less convenient. Print from Quill to a file, and then use the PRINT menu of QRAM (on hotkey) to SPOOL as many copies of the "_LIS" file to the printer as you wish.

Window DUMP Menu



From the DUMP menu, you can dump the contents of the entire SCREEN, the area owned by an individual JOB, or the current contents of a single WINDOW. Various printers are catered for, and others may be added to the list from time to time.

First you should select the appropriate printer type, scaling (hit with "X"), dump method and mode, and then "do" the SCREEN, a JOB or a WINDOW.

In this case a window is dumped to an Epson FX80 compatible at minimum scaling with inverted shading.

If you are trying to dump a JOB, then you just need to "do" on part of one of the job's windows. If you are trying to dump a WINDOW, then if you hit a window it will be highlighted. There may be more than one window at any particular point of the screen, so hitting the window again may highlight a different window. ENTER or "do" will dump the window occupying the currently highlighted area. ESC will rescue you if you have made a mistake

Note that you may dump a job or a window which is partially buried. Provided you can see enough to hit it, you can dump it. A snapshot is made of the dump, so that the QL is ready for use immediately. Moreover, it is possible to have more than one dump waiting to be printed.

For each printer there are 2 or 3 print resolutions. At the lowest resolution, the screen is dumped at one dot per pixel, at the higher resolutions there are several dots per pixel to give grey scale dumps. These higher resolutions are also an attempt to make circles on the screen appear as circles on the printer. The attempt often fails. The screen may be dumped as is, or INVERTED to print black as white and vice versa. The RANDOM option allows grey scale printing even when the resolution is one dot per pixel.

For EPSON printers and compatibles there are two types of printer definition: the MX80 option uses the normal and double density graphics capability of all EPSON compatible printers, the FX80 option uses the additional graphics modes on more recent EPSON compatible printers. These include the Canon PW1080, the Centronics GLP, and even Sinclair's QL Printer. If in doubt, use the MX80 option.

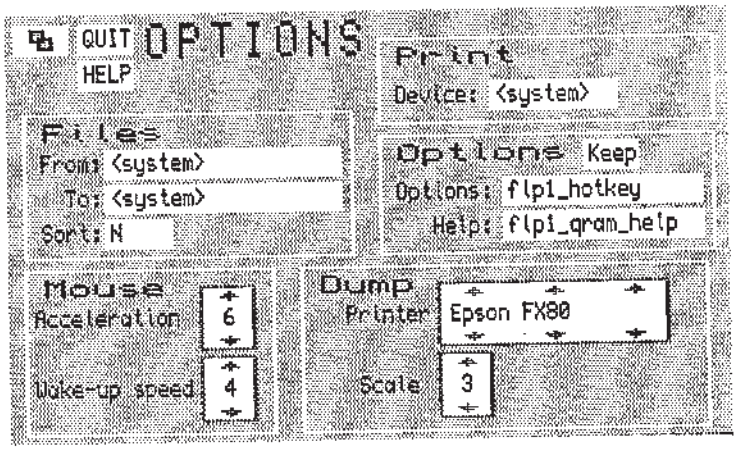
The EPSON LQ2500 is supported in both colour and monochrome 8 pin modes. This can produce dumps up to 13 inches wide. 24 pin graphics modes are not supported.

Other monochrome printers supported are the Seikosha GP-100 and GP-250, the brother HR4 and the Olivetti JP101.

Colour printers supported are the Seikosha GP-700, the Canon PJ1080 and the Epson JX80.

Full details of the print resolutions and the amount by which circles have failed to be circles are in Appendix I.

OPTIONS Menu



You can set static options for source and destination files (FROM and TO), file sort order, mouse/keyboard sensitivity, and printer options from here. Since this is likely to be a matter of personal preference, an option is provided to save your favourite combination back into the program file - you have to know where the program came from to do this! The file will usually be called FLP1_HOTKEY or MDV1_HOTKEY, but you may prefer to modify the executable file FLP1_QRAM or MDV1_QRAM.

The Mouse Acceleration is used to control both mouse and cursor key speeds. Note that "X" is the keystroke required to select the DUMP scaling.

Additional Utility Programs

All of these utilities can be executed using the EXEC (or EX) command or using the EXECUTE menu item in the FILES menu of QRAM.

HOT_MAKE is used for creating files of programs that you wish to be invoked by Hotkey. The other three utilities are used to tame otherwise ill behaved programs to make them suitable for use in the pointer environment. When programs have been processed by these utilities they should still be suitable for use in a standard QL environment. We recommend, however, that you keep the orginal versions of the programs in their untreated form.

Note that Hotkey programs must be pure (otherwise known as ROMable) code executable programs. BCPL and Supercharged and Liberated SuperBASIC programs are not pure (at the time of writing). This may not be a problem as far as the other utilities are concerned since SuperBASIC programs are not usually subject to the types of ailments that these utilities are intended to treat. While Liberated SuperBASIC programs appear to be susceptable to treatment, BCPL programs resist all our attempts to tame them.

Hotkey Creation: HOT_MAKE

The QRAM package is delivered with the QRAM program incorporated into the file HOTKEY. With HOTKEY loaded into the QL, QRAM is available to pop-up on a single keystroke. This "Hotkey" facility can be used with any "pure" executable program to produce pop-up applications. The utility Hot_Make is used to add other programs to HOTKEY or to create new Hotkey files. A Hotkey file can have any name. To make it easier to create new Hotkey files, the QRAM program itself is included on the QRAM disk or cartridge. Hot_Make is invoked by EXEC FLP1_HOT_MAKE or EXEC MDV1_HOT_MAKE.

Note that only one Hotkey file should be loaded into the QL. It is possible, however, to have many programs in one Hotkey file; each one is activated by a different keystroke.

HOT_MAKE: Example 1

This example adds "grquill" from FLP1_ to the file HOTKEY on FLP1_. The "grquill" file is produced by treating Quill with the Grabber utility.

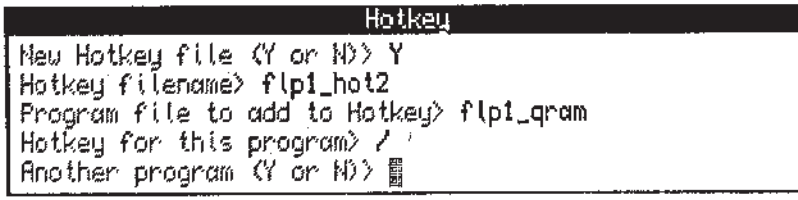
```
                    Hotkey
New Hotkey file (Y or N)> N
Hotkey filename> flp1_hotkey
Program file to add to Hotkey> flp1_grquill
Hotkey for this program> q
Another program (Y or N)>
```

First of all Hot_Make asks whether a new Hotkey file is to be created. In this case we wish to add a program to an existing Hotkey file. Next Hot_Make requests the name of the Hotkey file (FLP1_HOTKEY) and then the name of the file with the new program (FLP1_GRQUILL). Finally it requests the Hotkey, in this case "Q". When it has finished adding our Grabber processed version of Quill to the Hotkey file, it asks whether there is another program to be added. Pressing "N" will complete the operation.

Note that any changes made to the Hotkey file will not become effective until the Hotkey file is loaded into the resident procedure area when the QL is reset.

HOT_MAKE: Example 2

This example creates a new Hotkey file with a different name. Note that only one Hotkey file should be loaded into the QL at any time.



The Hotkey file "hot2" created by this example contains just QRAM. This should be the same as the Hotkey file supplied with QRAM.

Each program is copied into the Hotkey file as it is found. The *source* medium may be changed for each program; the *destination* medium must remain in the drive until Hot_Make has finished.

Memory Control: GRABBER

The Grabber utility (invoked by EXEC FLP1_GRABBER or EXEC MDV1_GRABBER) is used to add a small header to the front of executable programs to prevent them grabbing all the QL's memory (e.g. for Quill and the other Psion programs). It does this by grabbing most of the memory for itself, starting the aberrant program, and then releasing the memory. Note that since this means that the QL will be short of memory while a modified program is starting up, you should wait until the program is waiting for input before you try starting up any other jobs. Some programs (e.g. Quill) are very badly behaved if executed without adequate memory.

Before using any of the Psion programs within the pointer environment, they should be processed by Grabber. Grabber also adds a whole screen guardian window (see below) to the program.

It is only necessary to execute the Grabber program once for each program to create a better behaved copy of the original. The copy can then be used as required in place of the original. When the copy is created, it is necessary to specify a default memory allocation. About 32k bytes seems to be adequate for the Psion programs.

Toolkit users can set the memory required when the program is executed by giving the space required (in kilobytes) as a string parameter on the EX command:

EX grabacus;'64' where grabacus is a grabbed abacus, this will allow abacus to get 64 kilobytes of memory.

GRABBER: Example

This example creates a better behaved version of Quill as "grquill" on FLP1_. An unprocessed copy of Quill should be in drive 2.

```
Grabber
Program file needing treatment> flp2_quill
Destination filename> flp1_grquill
New program name> Quill32
Default memory available (kbytes)> 32
```

Grabber first requests the name of the file containing the unmodified program (FLP2_QUILL). It then requests the name of the file where the modified program will be saved (FLP1_GRQUILL, we usually put "gr" in front of the normal file name). If all is well, it requests a name for the program (Quill32). This name will be put into the program header and need have no connection with the file names used. Finally it requests the default memory you wish to make available to the program.

Unlockable Windows: UNLOCK

The Unlock utility (which is invoked by EXEC FLP1_UNLOCK or EXEC MDV1_UNLOCK) is supplied to treat any program which relies on the QL's destructive windows. These programs are usually small clock programs. A program treated by Unlock will have unlockable windows and so the program will continue to modify the screen, even if its windows have been buried.

UNLOCK: Example

This example of the use of Unlock is the treatment of one of the ubiquitous clock programs. It is very simple in use and just requires two file names: the original program file and the file for the treated program. You may also give the program a name which will identify it in the JOBS menu of QRAM.

```
Unlock
Program file needing treatment> flp1_clock
Destination filename> flp1_uclock
New program name> Clock
```

Window Control: GUARDIAN

The Guardian utility (invoked by EXEC FLP1_GUARDIAN or EXEC MDV1_GUARDIAN) is for treating programs whose windows are dynamically changed during execution. It has two modes of operation. In the simplest case you may specify a guardian window occupying the whole screen. Alternatively, you may let Guardian find the area of the screen that the program will require. In this case Guardian will cover the screen with the reset pattern, and then start your program. You should play with your program until you are satisfied that it has modified all the screen that it is ever likely to use, and then exit from the program. Do not remove your program using QRAM SuperToolkit II or any other utility. Guardian will tidy up and save the treated version of your program. It can take up to a second for guardian to notice that the area occupied by your program has changed, so do not hurry.

GUARDIAN: Example

As an example of Guardian, the GST Macro Assembler is treated to remove the slight untidiness caused when the single window is moved to preserve the Assembler banner.

```
                    Guardian
Program file needing treatment> flp1_asm
Destination filename> flp1_guasm
New program name? Asm
Whole screen Guardian..Y or N? N
```

Guardian requires to be given the name of the file with the original program (FLP1_ASM) and the name of the file for the treated program (FLP1_GUASM). There is no need to use any particular file name for the treated file, nor is the program name (Asm) significant. In this case Guardian is requested to find a suitable size guardian window, and the assembler is started. It is very well behaved, so we can exit without needing to assemble any files. Guardian then finishes off.

Boot File Creation: BOOT_MAKE

The boot file utility is used to assemble a number of QL extension files into a single file for more efficient loading. It also creates a SuperBASIC command file which can have SuperBASIC commands before and after the command which loads the extensions file.

This utility was used to make the BOOT_REXT file on your copy of QRAM. It may be used to create much larger sets of extensions than just the Pointer Interface and the Hotkey.

BOOT_MAKE: Example 1

This first example creates the standard QRAM boot file. First you must give the name of the boot file itself (usually FLP1_BOOT or MDV1_BOOT). The name of the extensions file will be the same except that it will have "_REXT" after the name.

```
Boot_Make
Boot filename> flp1_boot
Command>
Extension File> flp1_ptr_kbd
Extension File> flp1_wman
Extension File> flp1_hotkey
Extension File> flp1_ramprt
Command> hotkey
Command>
```

Next you can give any SuperBASIC commands which are required *before* the extensions are loaded. When you have given them all, press ESC. Then you give the extension file names in the order you wish to have them linked-in. Then press ESC again. Finally, give any commands that are to come *after* the extensions are linked-in (in this case "HOTKEY" to activate the Hotkey program). Pressing ESC will complete the process.

Each program is copied into the extensions file as it is found. The *source* medium may be changed for each program; the *destination* medium must remain in the drive until Boot_Make has finished.

BOOT_MAKE: Example 2

This second example creates a boot file for use with the Sandy SuperQBoard with mouse. This has the QL Pointer Interface and SuperToolkit II in ROM. The first command initialises the SuperToolkit II and the QL Pointer Interface.

```
Boot_Make
Boot filename> flp1_boot
Command> tk2_ext: pointer
Command>
Extension File> flp1_wman
Extension File> flp1_hotkey
Extension File> flp1_ramprt
Command> hotkey
Command>
```

RAM Disk Driver and Printer Buffer

This package has one the fastest and most comprehensive RAM Disk drivers available for the QL. It includes both dynamic and fixed memory allocation as well as fast Microdrive imaging. A bonus is that the memory can also be used a printer buffer, storing invisible files on their way to a printer.

RAM Disks for Beginners

The QL computer is delivered with two "mass storage" devices: the Microdrives. These devices have the same function as the floppy disks on more expensive personal computers, being designed for the permanent storage of programs and data. Other devices which behave in the same way as Microdrives (such as floppy or hard disks) may be added to the QL "transparently". This means that QDOS will ensure that a program does not need to "know" where its data is stored. A Microdrive looks, to a program, exactly the same as a floppy disk. This "device independence" is a built in characteristic of the QDOS operating system.

The term "RAM disk" is a misnomer. It is used to denote a "virtual" device (one that one only exists in the fertile imagination of the QL) that looks and behaves like a very fast disk device. It is so fast because being virtual, there is virtually nothing to move to get information in and out. It is, in fact, no more than a reserved area of the QL's main memory (its RAM - Random Access Memory). This means, of course, that any space taken by a RAM disk is not available to programs executing in the QL. Furthermore, any data stored in a RAM disk will be lost when the QL is turned off or reset!

RAM disks in the QL may be of any size, subject to there being enough memory. The normal usage of a RAM disk would be to copy all working files from Microdrive (or floppy disk) into a RAM disk; rename the RAM device to be MDV (to pretend that the data is really on the Microdrives); execute the programs (e.g. Quill, Archive etc.); and, at the end of the session, rename the RAM device to be RAM before copying the data files back to Microdrive.

On the other hand, it is just as easy to use a RAM disk without changing the name. All the filing system commands described in the "Microdrives" section of the QL Concept Reference Guide will work with RAM disks, provided the filenames start with "RAM" instead of "MDV".

`FORMAT ram2_200`	creates a new RAM disk 2, see below
`DIR ram1_`	directory listing of RAM disk 1
`SAVE ram1_myprog`	save the current SuperBASIC program as "myprog" in RAM disk 1
`OPEN_NEW #3,ram2_data`	creates and opens a new file "data" in RAM disk 2
`COPY mdv1_x TO ram1_x`	copies file x from MDV 1 to RAM Disk 1

RAM Disk Creation

A dynamic RAM Disk is created just by accessing it with any normal operation (e.g. DIR). This type of RAM Disk takes memory as required, and releases any memory as files are deleted or truncated.

A fixed RAM disk is created by formatting it: the size, in sectors, is given in place of the usual medium name. This pre-allocates all the space that will be available in the RAM disk.

```
FORMAT ram2_80
```

removes the old RAM disk number 2, and sets up a new RAM disk of 80 sectors. A RAM disk may be removed by giving either a null name or zero sectors

```
FORMAT ram1_ or FORMAT ram1_0
```

The RAM disk number should be between 1 and eight, inclusive, while the number of sectors (512 bytes) is only limited by the memory available. A RAM disk can be formatted from the FILES menu of QRAM.

Heap Fragmentation

The primary storage mechanism in the QL for permanent or semi-permanent memory allocations is a "heap". Allocating space in a heap, and then re-allocating this space as a different size, inevitably causes holes to be left within the heap. This reduces the amount of memory available to either SuperBASIC or executable programs.

This RAM disk driver has precautions to reduce the possibility of heap fragmentation, but it is preferable to consider any fixed RAM disk to be a permanent feature until the QL is reset.

Using a fixed RAM Disk not only reduces the danger of heap fragmentation, but as provides higher access speeds during file creation. Since it always occupies the maximum space you ever wish to use, it is much less flexible. QJUMP gives you the choice.

Microdrive Emulation

The standard driver also includes a SuperBASIC procedure RAM_USE to change the name of the RAM disk driver.

```
RAM_USE mdv or RAM_USE 'mdv'
```

resets the name of the RAM disk driver to "mdv", so that all subsequent open calls for Microdrives will use the RAM disks instead. Any three letters may be used as a new device name, in particular

```
RAM_USE ram
```

will reset the driver to its normal state.

Microdrive Imaging

Microdrive imaging is a very fast method of loading files from a Microdrive cartridge. To produce a Microdrive image, a RAM Disk is formatted with the name of the Microdrive required:

```
FORMAT ram1_mdv2        loads an image of mdv2 into RAM Disk 1
```

The RAM Disk can even load a Microdrive with a damaged directory. It cannot, however, load a Microdrive with a damaged map.

The RAM Disk will try up to 3 times to read a faulty sector. If it fails, the number of good sectors returned from the format will be fewer than the total number. Any file with bad sectors will be marked with an "*" in the RAM Disk directory.

Printer Buffer

The Printer Buffer has two names: the "usage" and the "device". The default usage name is PRT and the default device is SER. The printer buffer works by intercepting any OPEN call to a device whose name starts with its usage name. It substitutes the device name for the usage, and tries to open the device. If it succeeds, then all the output is buffered within the QL's main memory. If the device is in use, then the output is also buffered until the device is available. There is no limit to the number of buffered output files open at one time. If an error occurs during output, the buffer contents are thrown away. The current printed output may also be thrown away by the command

```
PRT_ABT
```

This will ABORT the file with the message ******* ABORTED *******

Using the default usage and device all references to a device called PRT will use the serial driver SER. Any parameters appended to the PRT name will by transferred to the SER name:

```
OPEN #3, prt            will open SER with a buffer
COPY flp1_fred, prt2c   will copy to SER2C with a buffer
```

The usage and device can be changed in the PRINT menu of QRAM or by using the command

```
PRT_USE usage, device
```

Two cases are particularly useful. In the first, the usage and device names are the same. This has the effect of introducing a buffer transparently into a device. In the second, the device name is of zero length. This means that the usage name may be followed by any device name.

```
PRT_USE ser,ser         buffer all output to SER1 and SER2
PRT_USE b_,''           b_ser1 is buffered SER1, b_par is
                          buffered PAR etc.
```

Appendix I - Screen Dump Formats

These are the various screen dump formats for each printer, with the number of dots on the printer for each pixel on the screen, the maximum picture width (in pixels) and the aspect ratio of the dump (less than one is too fat).

Printer	scale	dots /in	lines /in	dots 512	max width	ratio	dots 256	max width	ratio
Epson type	1	120	72	1x1	512	1.23			
MX80	1	60	72				1x1	256	1.23
	2	60	72	1x2	480	1.23	2x2	240	1.23
	3	120	72	2x2	480	1.23	4x2	240	1.23
Epson FX80	1	90	72	1x1	512	0.92			
additional	1	60	72				1x1	256	1.23
formats	2	90	72	1x1	512	0.92	2x1	256	0.92
	3	90	72	2x2	360	0.92	4x2	180	0.92
Epson	1	90	72	1x1	512	0.92			
JX80	1	60	72				1x1	256	1.23
	2	90	72	1x1	512	0.92	2x1	256	0.92
	3	90	72	2x2	360	0.92	4x2	180	0.92
Epson	1	80	60	1x1	512	0.99			
LQ2500	1	60	60				1x1	256	1.48
8 pin	2	120	60	1x1	512	0.74			
	2	80	60				2x1	256	0.99
	3	80	60	2x2	360	0.99	4x2	256	0.99
Epson	1	80	60	1x1	512	0.99			
LQ2500	1	60	60				1x1	256	1.48
8 pin	2	120	60	1x1	512	0.74			
colour	2	80	60				2x1	256	0.99
	3	80	60	2x2	360	0.99	4x2	180	0.99
Brother	1	120	72	1x1	512	1.23			
HR4	1	60	72				1x1	256	1.23
	2	60	72	1x2	480	1.23	2x2	240	1.23
	3	120	72	2x2	480	1.23	4x2	240	1.23
Olivetti	1	110	72	1x1	512	1.13			
JP101	1	110	108				1x1	256	0.75
	2	110	108	1x2	512	0.75	3x2	256	1.13
	3	110	72	2x2	440	1.13	4x2	220	1.13
Seikosha	1	60	63	1x1	480	0.78	1x1	256	1.41
GP-100A	2,3	60	63	1x2	480	1.49	2x2	240	1.41
Seikosha	1	60	72	1x1	480	0.61	1x1	256	1.23
GP-250X	2,3	60	72	1x2	480	1.23	2x2	240	1.23
Seikosha	1	80	80	1x1	512	0.74	1x1	256	1.48
GP-700A	2	80	80	1x2	512	1.48	2x2	256	1.48
	3	80	80	1x2	512	1.48	3x2	212	0.98
Canon	1	80	80	1x1	512	0.74	1x1	256	1.48
PJ1080A	2	80	80	1x2	512	1.48	2x2	256	1.48
	3	80	80	1x2	512	1.48	3x2	212	0.98