QL DISC INTERFACE MANUAL

(Version 3)

CST QL DISC INTERFACE MANUAL

IMPORTANT NOTE

DO NOT ATTEMPT TO FIT THE DISC INTERFACE UNIT OR THE DISC DRIVES TO THE QL UNTIL YOU HAVE READ THE INSTRUCTIONS BELOW THOROUGHLY.

UNDER NO CIRCUMSTANCES SHOULD YOU ATTEMPT TO CONNECT THE DISC INTERFACE UNIT OR THE DISC DRIVES TO THE QL WITH THE POWER TO ANY OF THEM CONNECTED - WHILST FITTING TO THE QL THE POWER MUST BE DISCONNECTED. SERIOUS DAMAGE MAY RESULT TO THE QL AND THE DISC INTERFACE UNIT IF THIS ADVICE IS NOT FOLLOWED.

Section One: Getting Started

1.0 Introduction

The CST Disc Interface unit allows you to use Disc Drives on your QL computer to store and load your programs. There are many advantages to using Disc Drives compared with Microdrives, the more obvious ones being the fact that Disc Drives are faster and more convenient to use A useful feature of the CST QL Disc Interface unit is that it allows you to copy files (data and programs) from Microdrive cartridges to Floppy Discs and vice versa. This means that all the existing software that you have been using on your QL can be transferred to floppy disc (Including the PSION suite of packages).

The CST Disc Interface unit can be used with three types of disc drive which take 3 inch, 3 1/2 and 5 1/4 inch floppy discs.

Each of these can be either single or double sided, and 40 or 80 track

Note: If your Disc Interface unit was purchased from CST, you will have a disc drive complete with supplied cabling. All you have to do is follow the instructions below and plug the ribbon cable from the Disc Drives into the socket on the Disc Interface unit. If you have Disc Drives purchased from another outlet, then refer to the fitting instructions supplied with them.

1.1 Care And Use Of Floppy Discs

Floppy discs are constructed from a thin piece of plastic sheet, treated with a magnetic compound which allows the storage and retrieval of data to and from the disc. When

tne disc is placed in the Disc Drives it is spun at high speed, in order to read or write information on to the disc the disc drive has a read/write head. This head moves in and out along the large round cut out on the disc jacket, (in the centre of the disc.) The head actually rests on the surface of the disc as it spins inside the jacket and reads or writes data to or from the disc. Obviously a Disc Drive is a sophisticated piece of equipment and should be treated with a certain amount of respect.

Great care should be taken when handling discs:

- Always insert the discs into the drives carefully and the correct way round
- Do not bend the discs
- Do not touch any of the exposed areas of the disc
- Do not allow smoke or other contaminents to come into contact with the surface of the disc
- Never switch the power to the disc drive or the QL on or off whilst a disc is actually in a drive. (This would damage the disc.)
- Always store the discs in the protective envelopes supplied.

To prevent losing important programs and data by either loss or damage to a disc or by accidental erasure of data from a disc, you are strongly advised to make "backup" copies of all important discs. A backup of a disc is simply an exact copy where all the data on one disc has been copied onto another, then two copies of the disc exist. You can make backups of discs by using the COPY command, this is explained below in section 2.4

1.2 Fitting the Interface

If you wish to use the Disc Interface with any others, you will need to use the Q+4 Peripheral Expansion Module or RAM Plus, available from CST. Please refer to its manual for details on how to install the card.

Otherwise, you will be installing the Disc Interface into the expansion slot on the QL as follows.

Firstly DISCONNECT THE QL FROM THE MAINS - if you do not you may damage the Disc Interface, the QL or both!

On the left hand end of the QL, you will find a rectangular plastic cover with a small clip which covers the QL expansion bus slot. Remove this cover by pulling on the clip away from the computer. The cover is often a very tight fit, and may require some effort to remove.

As you will see, the Disc Interface has a plastic cover over approximately half its length the other half being a bare circuit board with a plastic connector at its end. Slot this end of the Interface card into the end of the QL. The components on the board should be face up and it should slide in under retaining slots on the QL.

Now push the Disc Interface firmly home - this may take a little effort. You should be able to feel when the card is firmly in place and the plastic cover will be flush with the case of the QL.

1.3 Fitting the Disc Drives to the Interface

Ensure the QL and the Disc Drives are NOT connected to the mains/power supply. The connecting lead supplied wIth the Disc Drives wi1l already be correctly wired and tested and simply pushes into the socket protruding from the Disc Interface. The interface should already be fitted to the QL (if you have not done this refer to section 1.2 above). Check that the connector from your DISC Drive is correctly seated in the Disc Interface.

Note: If you have not got Disc Drives supplied by CST then you may have to fit a suitable connector to them that match the connector on the Disc Interface. Refer to the documentation supplied wIth your Disc Drives for wiring information.

1.4 Testing the Disc Interface

Once you are sure that you have followed all of the steps outlined in sections 1.2 and 1.3 above you can then start using your Disc Drives on your QL.

IMPORTANT: POWER-UP AS FOLLOWS:

EITHER:

(a) First connect the QL to the mains supply and then connect the Disc Drives to the power source (Without a floppy disc in any of the drives)

OR:

(b) Apply power to QL and Disc Drives simultaneously by means of a multi-way mains power block.

If you do not follow this sequence for connecting the power then damage may be caused to the QL and the Disc units

Do not put a floppy disc in the drive(s) at this stage. After the memory test screen you should see the usual TV/Monitor selection screen with the message

CST Q Disc Interface V n.nn (C) 1984

If this message is not displayed or the TV/Monitor message does not appear check that your operating system is version AH or later. If it is not, then see your supplier, who can arrange an update for you. If your operating system is up to date, then check that you have correctly installed the Disc Interface and the Disc Drives as outlined above. If neither of these cures the fault contact your supplier for advice.

If the message appears and TV/Monitor selection works normally then your QL and Disc Drives are ready for use.

1.5 Removing the Disc Interface

If you need to remove the Disc Drive(s) and Interface for any reason, first DISCONNECT THE QL FROM THE MAINS and the Disc Drive(s) from their power source, if you fail to do this you may damage your QL and Disc Interface unit. Then reverse the above procedure for installation. Unplug the cable from the Disc Drive(s) to the Disc Interface and gently pull the Disc Interface unit fom the expansion slot on the QL. (Put it in its original packing for safe keeping). Lastly, replace the cover on the expansion slot of the QL - this may again need some pressure.

Section Two: Simple Use of the Disc Interface

In all of the examples in this section and (following ones) you are invited to type a command followed by the appropriate syntax, for example:

DIR flp1_

After typing the text you must press the enter key for the command to work (An explanation of the DIR command is given below in section 2.1).

Note: In general you will find that floppy disc use is very similar to using microdrives because of the QL's device independent I/O. As a general rule, anywhere mdv (Microdrive) is used as part of the syntax of a command you may use flp to denote Floppy Disc Drive(s)

2.0 FORMATTING a Disc

Before a disc can be used to store data, it first has to be formatted by the Disc system. This process divides the disc into tracks and sectors so that any files (data or programs) you store on the disc are automatically stored in an orderly fashion and can easily be retrieved by the computer at a later date.

To format a disc, put a disc in drive 1 (the top drive), type the following and press the enter key

FORMAT flp1_xxx

In this example flp1 is Disc Drive 1 and xxx is the disc name which can be up to ten characters long. You may wish to give your discs names such as "Documents" or "Letters" so that you can easily identify the kind of information held on a disc without having to look at the files.

After you have typed in the above line and pressed the enter key the disc drive will make a "whirring" noise as it formats the disc. The following will be displayed on the screen when the formatting operation has been completed:

1440/1440

The two sets of digits displayed on the screen refer to the number of useful sectors and total sectors on the disc respectively. These digits will vary depending on the storage capacity of the disc.

Important Note: If you FORMAT a disc with data already on it, all the data on the disc will be deleted. Use this command with care!

2.1 The DIR command

The **DIR** (DIRECTORY) command will display a list of all the files held on a disc. (Except on a newly Formatted disc where no files already exist). For example:

DIR flp1_

will display a list of the files on the disc in disc drive 1.

The DIR command will display the information on the screen in the following way.

disc name	i.e. The name you have allocated to the disc.
free sectors	i.e. The number of free sectors on the Disc.
available sectors	i.e. The number of sectors on the Disc.
file name(s)	i.e. A list of the files on the Disc.

2.2 The LOAD command

You use the LOAD command to take a Basic file from the disc and load it into the memory in the computer

LOAD flp1_xxx

In this example flp1 is Disc Drive 1 and xxx is the name of the file that you wish to load into the computer. Filenames can be up to nine characters long and can be any character on the keyboard. As with Disc names (as described in section 2.0) it is useful to give names to your files that describe the type of data that is held in them.

2.3 The SAVE command

The SAVE command is used to save Basic programs to the specific disc drive:

SAVE flp1_xxx

will save the file xxx to the disc in Drive 1

You can SAVE all of a file or just specific parts of it if you wish, for example:

SAVE flp1_xxx;20 TO 70

will save lines 20 to 70 in the file called xxx to the disc in Disc drive 1. You can also SAVE from a specified line number to the end of the program as follows:

SAVE_flp1_xxx;20 TO

will SAVE from line 20 to the end of the file.

2.4 The COPY command

The COPY command is used to COPY files from one Disc Drive to another or copy files from a Microdrive to a Disc Drive. (Use this command for making backup copies of discs). For Example:

COPY mdv1_xxx TO flp1_xxx

will copy the file xxx from Microdrive 1 to Disc Drive 1

Or

COPY flp1_xxx TO flp2_xxx

will COPY the file xxx from the disc in drive 1 to the disc in drive 2.

2.5 Using the Disc Interface from PSION programs

The Disc Interface works with the PSION supplied packages. The later versions of the packages have an option in the install program to change the default channel from MDV1. This is described in the manual supplied with the packages.

Further commands and utilities are described in the following section (Section 3)

Section Three: Summary of Commands Applicable to Disc Systems

3.0 Naming the Disc Device Driver

The on-board device driver in the Disc Interface is treated in the same way as other device drivers on the QL. The way this driver is named determines how data for output to the Disc is handled. The syntax of a <disc device> is as follows

flp drive_xxx

where

Drive is either drive 1 or 2.

XXX is the filename. (Refer to section 2.2 for a description on using filenames.)

3.1 Numbering Channels

The syntax of <channel> in Super Basic is #n, where n is an integer. Channels 0, 1 and 2 are used by the screen, so should not be used for the Disc Interface. To save memory channel numbers should be kept as low as possible, since the QL allocates memory for all channels up to the highest number one used i.e.

open #5000,flp#_xxx

attempts to reserve a total of 5000 channels from memory.

3.2 DELETE

The DELETE command will erase a file from the directory of the Disc in the Drive specified, for example:

DELETE flp1_xxx

will delete the file xxx from the disc in Drive 1

3.3 LBYTES

This command is used to load a data file from disc into the memory in the computer at a specified start address (i.e. memory location) for example:

LBYTES flp1_xxx,65536

will take the file xxx on the disc in Drive 1 and load it at memory location 65536

3.4 MERGE

MERGE will load a file from the specified Disc Drive and interpret it as a SuperBASIC program. If the new file contains a line number which doesn't appear in the program then the line will be added. If the new file contains a replacement line for one that already exists then the line will be replaced. All other old program lines are unaffected.

For example:

MERGE flp1_xxx

will take the file xxx on the Disc in Disc Drive 1 and load it into the computer as a SuperBASIC program.

Note: If a line input during a MERGE does not contain the correct SuperBasic syntax, the word MISTAKE is inserted between the line number and the body of the line. A line with a mistake in it will generate an error.

3.5 OPEN

With the OPEN command you can link a logical channel to a physical QL device such as a Disc Drive.

If the channel is linked to a Disc Drive then the disc file can either be an existing file or a new file OPEN_IN will open an already existing Disc file for input and OPEN_NEW will create a new Disc file for output.

OPEN #9,"flp1_xxx"

will open channel 9 to Drive 1 with filename xxx

3.6 SBYTES

SBYTES allows specific areas of the QL memory to be saved to Disc. The start address and the length (in bytes) must be specified. For example:

SBYTES flp1_xxx, 131072, 32768

will save 32768 bytes of memory from start address 131072 to the disc in Drive 1 as the file xxx. (In this example the contents of the screen would be saved to disc)

Section Four. Additional Commands

The commands described in this section fall into a number of categories and are additional to the standard QL commands

1) Device Naming and Defaulting
2) Random Access I/O
3) File Handling
4) File Maintenance
5) Executing Programs
6) Job Control
7) Screen Handling and Character Fount Setting
8) Memory Allocation
9) Conversions
10) Resident Clock
11) Disc Control

Given below is the name of each command, a brief explanation of what it does and an example of the syntax employed when using the command.

4.0.1 EXTRAS

A very useful command that you may wish to use is the EXTRAS command. This command will list on the screen all other additional commands provided by the CST Disc Interface Unit, together with any commands loaded into RAM at power up (boot).

To use the EXTRAS command simply type:

EXTRAS

and press the ENTER key. All other additional commands will be displayed on the screen

4.1 Device Naming And Defaulting

4.1.1 FLP_USE

Renaming the Floppy Driver.

It is possible to change the name of the floppy disc device to any sequence of three characters using the command FLP_USE. This command may be used to cause the discs to emulate the microdrives if the characters 'mdv' are used. The syntax is:

FLP_USE string

where string is any 3 characters, e.g.

FLP_USE mdv

will start microdrive emulation. Note changing the name of th efloppy disks does not change the strings set by PROG_USE and DATA_USE.

4.1.2 PROG_USE, DATA_USE

The Q-disc ROM provides additional commands to allow you to use default devices in file names. The ROM includes new definitions of the **EXEC** and **EXEC_W** SuperBasic commands (**EX** and **EW**) which allow the program name specified to miss off the initial drive specification. The default drive specification is set by the PROG USE command. All of the other additional commands provided by the ROM add the default set by the DATA USE command. The syntax of these commands is:

```
PROG USE name
DATA_USE name
```

where name is any sequence of characters which will be appended to the file name, e.g.

```
PROG_USE flp2_
DATA_USE flp1_data_
```

If the directory name supplied does not end with '_', '_' will be appended to the_directory name. The directory name can be more detailed than just a device name.

For example:

```
DATA_USE flp1_project5_library
.....
WDIR
ferr = FOP_NEW(#3, fred)
```

will produce a directory listing Of all files with names starting with 'flp1_project5_library' and then open a new file called 'flp1_project5_library_fred' . The default set by this command is optional, and is only used if the name supplied to a command is not a valid file or device name. Thus:

```
ferr = FOP_NEW(#3, flp2_fred)
```

will open the file 'flp2_fred' rather than the file 'flp1_project5_library_flp2_fred' !

The initial values of PROG_USE and DATA_USE are flp1_ and flp2_ respectively.

N.B. These commands set defaults only for commands provided by the Q-disc rom - they do not affect the normal commands such as LOAD, SAVE etc.

4.2 Random Access I/O

In QDOS, files appear as a continuous stream of bytes. On directory devices (Microdrives, floppy disks etc.) the file pointer can be set to any position in a file. This provides 'direct access' to any data stored in the file. Access implies both read access and, if the file is not open for read only (OPEN_IN from SuperBASIC, IO.SHARE in QDOS), write access. Parts of a file as small as a byte may be read from, or written to any position within a file. QDOS does not impose any fixed record structures upon files: applications may provide these if they wish.

Procedures are provided for accessing single bytes, integers, floating point numbers and strings. There is also a function for finding the current file position.

The general form of the direct I/O commands is:

command #n [\ pointer] {,item}
or **command item {,item}**

It is usual (although not essential - the default is #3) to give a channel number for the direct I/O commands. If the pointer is given, the file position is set before processing the list of I/O items: if the pointer is a floating point variable rather than an expression, then, when all items have been read from or written to the file, the pointer is updated to the current file position.

4.2.1 BPUT BGET - Byte I/O

BPUT #n [\ pointer] {,item}
BPUT #n [\ pointer] {,item}

BGET gets 0 or more bytes from the channel. BPUT puts 0 or more bytes into the channel. For BGET, each item must be floating point or integer variable: for each variable, a byte is fetched from the channel For BPUT, for each item a each item must evaluate to an integer between 0 and 255, byte is sent to the output channel.

For example the statements

```
abcd=2.6
zz%=243
BPUT #3,abcd+1,'12',zz%
```

will put the byte values 4, 12 and 243 after the current file position

Provided no attempt is made to set a file position, the direct 1/O routines can be used to send unformatted data to devices which are not part of the file system. If, for example, a channel is opened to an Epson compatible printer (channel #3) then the printer may put into condensed underline mode by

BPUT #3,15,27,45,1

instead of

PRINT #3,chr$(15);chr$(27);'-';chr$(1);

4.2.2 GET PUT - Unformatted I/O

It is possible to put or get values in their internal form. The PRINT and INPUT commands of SuperBASIC handle formatted IO, whereas the direct I/O routines GET and PUT handle unformatted I/O. For example, if the value 1.5 is PRINTed the byte values 49 ('1'), 46 ('.') and 53 ('5') are sent to the output channel. Internally, however the number 1.5 is represented by 6 bytes (as are all other floating point numbers). These six bytes have the value 08 01 60 00 00 00 (in hexadecimal). If the value is PUT, these 6 bytes are sent to the output channel.

The internal form of an integer is 2 bytes (most significant bytes first). The internal form of a floating point number is a 2 byte exponent to base 2 (offset by hex 81F), followed by a 4 byte maintissa, normalised so that the most significant bits (bits 31 and 30) are different. The internal form of a string is a 2 byte positive integer, holding the number of characters in the string, followed by the characters.

GET #n [\ pointer) {,item}
PUT #n [\ pointer) {,item}

GET gets data in internal format from the channel. PUT puts data in internal format into the channel. For GET, each item must be an integer floating point, or string variable. Each item should match the type of the next data item from the channel. For PUT the type of data, put into the channel, is the type of the item in the parameter list. The commands

```
fpoint=54
...
wally% =42: salary=78000: name$='Smith'
PUT #3, fpoint, wally%, salary, name$
```

will position the file, open on #3, to the 54th byte, and put 2 bytes (integer 42), 6 bytes (floating point 78000), 2 bytes (integer 5) and the 5 characters 'Smith'. Fpoint will be set to 69 (54+2+6+2+5)

For variables or array elements the type is self evident, while for expressions there are some tricks which can be used to force the type:

```
....+0          will force floating point type;
....&"          will force string type;
....||0         will force integer type
xyz$= 'ab258.z'
....PUT #3,37,xyz$(3 to 5)||0
```

will position the file opened on channel #3 to the 37th byte and then will put the integer 258 on the file in the form of 2 bytes (value 1 and 2, i e. 1*256+2).

4.2.3 FPOS - File Position Enquiry

There is one function to assist in direct access 1/O: FPOS returns the current file position for a channel. The syntax is:

FPOS (#n)

For example:

```
PUT #4\102,valuel,value2
ptr = FPOS (#4)
```

will set 'ptr' to 114 (=102+6+6).

The file pointer can be set by using any of GET, BGET , PUT or BPUT with no items to be got or put. If an attempt is made to put the file pointer beyond the end of file, the file pointer will be set to the end of file and no error will be returned. Note that setting the file pointer does not mean that the required part of the file is actually in a buffer but that the required part of the file is being fetched. In this way it is possible for an application to control prefetch of parts of a file.

4.2.4 FLEN FTYP FDAT - File Enquiry Functions

There are three functions to extract information from the header of a file. Note that in current versions of the microdrive handler, the header is only updated on a FS.HEADS call or on closing the file. This means that the file length read from the header is the file length as it was when the file was opened.

If a file is being extended, the file length can be found by using the FPOS function to find the current file position. (If necessary the file pointer can be set to the end of file by the command GET #n\9999

FLEN(#n)	**returns the file length,**
FTYP(#n)	**returns the file type (O=normal l=EXEC),**
FDAT(#n)	**returns the data space for EXEC files.**
OPEN #3,flp1_fred	**PRINTs the length of file fred**
PRINT FLEN(#3)	**on flp1_**

4.3 File Handling

4.3.1 FOPEN FOP_IN FOP_NEW FOP_OVER FOP_DIR - File Open Functions

There is a set of functions for opening files. These functions differ from the OPEN procedures in ROM in two ways: firstly if a file system error occurs (e.g. 'not found' or 'already exists') these functions return the error code and continue: secondly the functions use the DATA_USE directory default.

FOPEN (#3,name)	**open for read/write**
FOP_IN (#3,name)	**open for read only**
FOP_NEW (#3,name)	**open a new file**
FOP_OVER (#3,name)	**open a new file, or overwrite old file**
FOP_DIR (#3,name)	**open a directory**

Directory entries may be read using GET to get information. Each entry is 64 bytes long, the length of the file is at the start of the entry, there is a standard string starting at the 14th byte of the entry giving the filename and there is the update date as a long integer starting at the 56th byte.

Example of File Open

A file may be opened for read onlv with an optional extension using the following code

```
ferr= FOP_IN (#3,name$&'_ASM') : REMark try to open _ASM file
IF ferr=-7: ferr= FOP_IN (#3,name$) : REMark ERR.NF, try no _ASM
```

4.4 File Maintenance

4.4.1 RENAME - Changing A File's Name

```
RENAME old,new
```

renames a file: the DATA_USE default directory is used for both filnames.

4.4.2 TRUNCATE - Shortening A FIle

```
TRUNCATE #n
```

truncates the file open on #n to the current file position.

4.4.3 VIEW - Examining a File

VIEW is a procedure intended to allow a file to be examined in a window on the QL display.

VIEW name	**view a file (in #1): lines are truncated to fit in the window, and when the window is full,~ CTRL F5 is generated.**
VIEW #window,name	**view a file in a given window; the DATA_USE directory default is used.**

4.4.4 STAT - Examining A Medium

```
STAT [#n,][name]
```

prints medium name, number of free sectors, total number of sectors.

4.4.5 WDIR WSTAT - Examining a Directory

WDIR[#n,][wild_name]

lists directory, generates CTRL F5 when the window is full.

WSTAT [#n,][wild_name]

list file name, length and last update date, generates CTRL F5 when the window is full.

4.4.6 WDEL WDEL_F - Deleting Multiple Files

WDEL [#n,][wild_name]

deletes files (requests confirmation).

WDEL_F [wild_name]

deletes files (forced)

When using WDEL each filename is written to the chosen channel and the user is requested to press one of the keys.

Y	(yes)	delete this file;
N	(no)	do not delete this file;
Q	(quit)	do not delete this or any of the next files
A	(all)	delete this and all the next matching files

4.4.7 Wild File Names

The wild_name in these procedures may refer to more than one file. To do this file names are divided into sections (e.g. mdv2_fred_bin has three sections) and a wild name may have missing sections (e g mdv2_old_ _list has one missing section). All those files whose names have sections matching the sections in the wild name are referenced hy the commands. In the following examples flp2_ is assumed to be the default data directory.

Wild name	Typical matching files
fred	flp2_fred flp2_freda_llst
_fred	flp2_fred flp2_freda_llst flp2_old_fred flp2_old_freda_list
flp1_old_ _list	flp1_old_jo_list flp1_old_freda_list

4.4.8 WCOPY - Wild Card Copying

The WCOPY command has several optional forms:

WCOPY source wild name TO destination wild name
WCOPY source wild name, destination wild name
WCOPY #channel,source wild name TO destination wild name
WCOPY #channel,source wild name, destination wild name

If no channel is given, the dialogue will be in channel #0.

When using WCOPY, each source and destination filename is written to the chosen channel, and the user is requested to press one of:

Y	(yes)	copy this file
N	(no)	do not copy this file
Q	(quit)	do not copy this or any more files
A	(all)	copy this and all the next matching files.

If the destination file already exists, the user is requested to press one of.

Y	(yes)	copy this file, overwriting the old file
N	(no)	do not copy this file
Q	(quit)	do not copy this or any more files
A	(all)	overwrite the old file, and any other files requested to be copied

WCOPY may be used to copy entire directories. The destination name is made up from the actual source file name and the destination wild name. If a missing section of the source wild name is matched by a missing section of the destination wild name then that part of the actual source file name will be used as the corresponding part of the actual destination name. Otherwise the actual destination file name is taken from the destination wild name. If there are more sections in the destination wild name than in the source wild name, these extra sections will be inserted after the drive name, and vice versa.

For example, if the default data directory is flp2_, then

WCOPY flp1_,flp2_	would copy all files on flp1 to flp2
WCOPY fred, mog	would copy
flp2_fred	to flp2_mog
flp2_freda_list	to flp2_moga_list
WCOPY _fred,_mog	would copy
flp2_fred	to flp2_mog
flp2_freda_list	to flp2_moga_list

flp2_old_fred	to flp2_oldmog
flp2_old_freda_list	to flp2_old_moga_list
WCOPY list,old_list	would copy
flp2_jo_list	to flp2_old_jo_list
flp2_freda_list	to flp2_old_freda_list
WCOPY old_list,flp1_list	would copy
flp2_old_jo_list	to flp1_jo_list
flp2_old_freda_list	to flp1_freda_list

4.4.9 SPL SPL_USE - File Spooler

The SPL procedure sets up a job to copy a file. Only the source need be given: the destination may be defaulted. The source file has its default set up by the DATA USE command. The default destination is SER. The SuperBASIC interpreter will continue after the job has been set up, the file being copied in the background. SPL differs from COPY not only in that it operates as a job in the background, but also in its handling of file headers. The COPY procedure copies both the file and its header: to copy a file to a device like a printer, the variant, COPY N is used to copy without a header. SPL will will however, not copy the header from an ordinary data file, but it will copy the header of a file which is one of the special types (e.g. executable program file). Furthermore, when using SPL to copy from file to file, if the destination file already exists, then it will be overwritten.

The command syntax is

SPL source_file or
SPL source_file TO destination

The source and destination files may be given as names, or as a SuperBASIC channel number (e.g. #3).

The default set by the DATA_USE command is used to find the source file, and there is aspecial command, SPL USE, to set the default destination. The default destination device or directory may be up to 32 characters long.

SPL_USE device_name
or **SPL_USE directory_name**

A device name does not end in '_' ; a directory_name must end in '_'

If the **SPL** command is given with only one parameter (the source filename) the output file (or device) will be derived from the current default set by SPL USE as follows:

1) directory_name & source_filename or
2) device_name

If the **SPL** command is given with two parameters, the output file (or device) will be derived as follows:

1) destination_filename or
2) directory_name & destination_filename

SPL will often be used to copy files in the background, but it can be used as a true spooler when used with the default output device. In this case, if the output device is in use, the **SPL** job will suspend itself until the device is available.

SPL Examples

SPL myfile	using the supplied defaults this will spool **FLP2_MYFILE** to **SER**.
SPL flp1_demo_myfile TO ser2	the file **FLP1_DEMO_MYFILE** will be spooled to **SER2**.
DATA_USE flp2_demo **SPL_USE ser2** ..... **SPL myfile**	this will also spool the file **FLP2_DEMO_MYFILE** to **SER2**
SPL mdv2_myfile, mdv1_myfile	does the obvious
SPL_USE mdv1_ ..... **SPL tax**	using the supplied **DATA_USE** default, this will also_spool **FLP2_TAX** to **MDV1_TAX**.
SPL yourfile to #3	will spool yourfile to the file or device already opened as **#3**.

4.5 Executing Programs

4.5.1 EX EW

These commands are enhanced versions of the standard SuperBASIC EXEC and EXEC_W respectively. For simple use, the commands are interchangeable. The syntax is:

EX program_filename
ET program_filename, filename, filename ...
EX program_filename; option string
ET program_filename, filename.../;option_string

EX also provides default directories for the program and data files. At power on programs are taken from **FLP1**_ and data files are assumed to be on **FLP2**_. For example:

EX cpy, myfile, myfile_sav

will, by default, use the program **FLP1_CPY** to copy **FLP2_MYFILE** to **FLP2_MYFILE_SAV**.

In place of the data filename, a SuperBASIG channel number may be used; the channel must be open and have the access (read or write) required by the filter. The following command will copy myfile to window **#2**:

EX cpy, myfile, #2

4.5.2 EX and multitasking

As we have already said it is possible to have several jobs running in the QL at any one time. Furthermore, it is possible to have a chain of cooperating jobs engaged in processing the same data in a 'production line'. When using a production line of this type each job performs a well-defined part of a total process. The first job takes the original data and does its part of the process, the partially processed data is then passed on to the next job which carries out its own part of the process. The data is passed from one job to the next using a 'pipe'; the data itself is called a 'stream' (which flows in one direction), and the jobs processing the data are called 'filters'.

The I/O subsystem within Qdos ensures that a job can handle a pipe just like any other simple I/O device, and so a job does not need to know that it is being used as a filter.

EX can initiate chains of programs (filters) connected by pipes. It is also possible to use **EX** to connect the SuperBASIC interpreter to a chain of jobs.

The complete form of the **EX** command is:

EX [#n TO] prog_spec {TO prog_spec} [to #m]

Each **TO** separator creates a pipe.

If the parameters of **EX** start with **#n**, then the SuperBASIC channel **#n** will be closed (if it was already open) and a new channel **#n** will be opened for output only. Sending any output to this channel will send the output down the pipe to the chain of jobs. When the channel is CLOSED, the chain of jobs will be removed from the QL.

If the parameters of **EX** end with **#n**, then the SuperBASIC channel **#n** will be closed (if it was already open) and a new channel **#n** will be opened for input only. Any data passing down the chain of jobs will appear in this input channel. When all the data has been passed, the jobs will remove themselves, and any further attempt to take input

from this channel will get an 'end of file' error, which may be tested using the EOF function.

The program specification, prog_spec in the example above, is defined as:-

program filename {,data_filename} [;option string]

All filenames and the option string may be names, strings or string expressions. In addition the data filename may be a SuperBASIC channel number.

The significance of the filenames is to a certain extent program dependent, but there are two general rules which should be used by all filters:

the primary input of a filter is the pipe from the previous lob in the chain (if it exists), or else the first data file;

the primary output of a filter is the pipe to the next job in the chain (if it exists), or else the last data_file.

Many filters will have only two I/O channels: the primary input and the primary output.

4.5.3 EW Variant

The **EW** variant of **EX** will start a chain of lobs and suspend the SuperBASIC interpreter until the last lob in the chain has completed. Clearly the constructions '#n TO' and 'TO #n' cannot be used as, if it is suspended, the interpreter cannot send any output down a pipe, or take input from a pipe.

4.6 Job Control

As QDOS is a multitasking operating system, it is possible to have, at one time, in the QL a number of competing or co-operating jobs. Jobs compete for resources in line with their priority, and they may co-operate using pipes or shared memory to communicate. The basic attributes of a job are its priority and its position within the tree of jobs (ownership). A job is identified by two numbers: one is the job number which is an index into the table of jobs, and the other is a tag which is used to identify a particular job so that it cannot be confused with a previous job occupying the same position in the job table. Within QDOS the two numbers are combined into the job ID which is job number + tag*65536.

4.6.1 JOBS - Listing Running Jobs

There is a procedure which will list all the jobs running in the QL at the same time. If there are more jobs in the machine than can be listed in the output window, the procedure will freeze the screen (CTRL F5) when it is full. The procedure may fail if jobs are removed from the QL while the procedure is listing them. The following information is given for each job:

the job number
the job tag
the job's owner job number
a flag 'S' if the job is suspended
the job priority
the job (or program) name.

The syntax of the command is:

JOBS [#n] where #n is the channel for the listing

4.6.2 RJOB SPJOB - Controlling A Job

RJOB allows jobs to be removed from the machine without resetting the whole machine. The syntax is:

RJOB job_number, tag, error_number

where job_number and job_tag are as given by the JOBS command and error_code is the code that will be returned to any job that is waiting for the killed job to complete. e.g.

RJOB 1,5,0

will kill job number 1, tag 5 and report an error code of zero (success) to any waiting job.

SPJOB allows the priority of a job to be changed.

SPJOB job_id, priority sets a job's priority

4.7 Screen Handling and Character Fount Setting

4.7.1 CURSEN CURDIS WMON WTV

The function INKEY$ is designed so that data can be taken from the keyboard without enabling the cursor. Sometimes, however, it may prove useful to enable, the cursor in a particular window. When it is enabled, the cursor will appear solid. When an INKEY$ is called to get data from that window, the keyboard queue will be switched to the window (unless the window with the keyboard has an active cursor) and the cursor will start to flash. Note that INKEY$ defaults to input from #0, whereas CURSEN and CURDIS, like most other screen I/O commands, default to channel #1.

CURSEN	**enables the cursor in #1**
CURSEN #Channel	**enables the channel's cursor**
CURDIS	**disables the cursor in #1**
CURDIS #Channel	**disables the channel's cursor**

For example, the following code will enable the cursor in window #2, wait for ten seconds for a character to be typed in and then disable the cursor. If nothing is typed in within the ten seconds, in$ will be a null string.

```
CURSEN #2
in$ = INKEY$ (#2, 500)
CURDIS #2
```

There are two commands to resed the windows to the turn-on state:

WMON mode	**resets windows do monitor default.**
WTV mode	**resets windows to TV default.**

The mode should be 0, 4 or 512 for 4 colour (512 pixel) mode, or 8 or 256 for 8 colour (256 pixel) mode. Only the window sizes, positions and borders are set by these commands; the paper, strip and ink colours are unchanged.

4.7.2 CHAR_USE CHAR_INC

The QL has two character founts built in. The first provides the patterns for the character values from 32 (space) to 127 (copyright). The other provides the characters from 128-191 (foreign and special characters). The character generator will use a pattern from the first fount if one exists for that character value, or from the second fount, or if none exists, the lowest entry in the second fount.

CHAR_USE [#channel], address 1/0, address 2/0

sets the two founts for one window (default #1). If an address is zero, the fount is reset to the default.

The format of a fount in the QL is:

byte	lowest valid character value
byte	number of valid characters-l
9 bytes	pixels for first character.
9 bytes	pixels for next character; etc .

The pixels are stored with the top line in the lowest addressed byte. For each pixel of ink colour, a bit is set in the byte. The leftmost pixel is in bit 6 of the byte; the rightmost in bit 2.

CHAR_INC [#channel], width, height

sets the horizontal and vertical character spacing in pixel units. Extreme care should be taken if the increments are set to less than the size of the character size, in case characters at the right hand, or bottom edges of a window are drawn partly outside the window; if this is at the edge of, the screen, random corruption can occur. To avoid this, use borders to reduce the effective size of windows.

4.8 Memory Allocation

4.8.1 FREE_MEM ALCHP RECHP CLCHP

QDOS is a multitasking operating system; therefore, there may be several jobs running in a QL, and the amount of free memory may vary unpredicably. No job may assume that the amount of free memory is fixed. To find the amount of free memory (this is defined as the space used for filing system slave blocks, less the space required for one slave block), the function FREE_MEM (which has no parameters) is used. The function ALCHP (allocate from common heap) is used to obtain memory. If there is not enough free memory in one piece, the function returns 0, otherwise it returns the address of the base of the area allocated. The area may be returned to QD0S by invoking RECHP (release to common heap). If the base address of an area in the heap has been forgotten (CLEAR or NEW), then all area may be CLCHP (clear common heap).

FREE MEM	returns current free memory
ALCHP (no of bytes)	allocate a memory area
RECHP base_address	release a memory area
CLCHP	release all areas allocated

It is inadvisable to take all the memory; at least 512 bytes should be left to avoid problems with microdrive handling.

4.9 Conversions

4.9.1 BIN$ HEX$ BIN HEX - Radix Conversions

A set of numeric conversion routines is provided: these convert values to hexadecimal or binary strings and vice versa, as well as values to fixed format decimal strings.

BIN$ (value, number_of_bits)
HEX$ (value, number_of_bits)

Each returns a string of sufficient length to represent the value of the specified **number_of_bits** of the least significant end of the value. In the case of **HEX$** the **number_of_bits** is rounded up to the nearest multiple of 4.

BIN (string)
HEX (string)

Each converts the string supplied to a value. For **BIN**, any character in the string, whose ASCII value is even, is treated as 0; any character, whose ASCII value is odd, is treated as 1. E.g. **BIN ('.#.#')** returns the value 5. For **HEX** the 'digits' '0' to '9' 'A' to 'F' and 'a' to 'f' have their conventional meanings. **HEX** will return an error if it encounters a non-recognised character.

4.9.2 FDEC$ IDEC$ CDEC$

The functions convert values to decimal strings.

FDEC$ (value, number_of_chars, number_of_places)
IDEC$ (value, number_of_chars, number_of_places)
CDEC$ (value, number_of_chars, number_of_places)

FDEC$** converts the value as it is, whereas **IDEC$ assumes that the value given is an integral representation in units of the least significant digit displayed. **CDEC$** is for currency conversion, which is similar to **IDEC$** except that there are commas every three digits.

FDEC$(1234.56, 9, 2)	**returns ' 1234.56'**
IDEC$(123456,9,2)	**returns ' 1234.56'**
CDEC$(123456,9,2)	**returns ' 1,234.56'**

4.9.3 PARTYP PARUSE - Type Checking

The dummy parameters off a SuperBasic procedure or function have the same type and dimensions as the actual (calling) parameter. Two functions are provided to determine the type and usage of a parameter.

PARTYP(name) returns type:
0 = null
1 = string
2 = floating point
3 = integer

PARUSE(name) returns usage:
0 = unset
2 = variable
3 = array

4.10 Resident Clock

4.10.1 CLOCK

There are a number of optional forms of the CLOCK command:

CLOCK	default clock, two rows of ten characters in default position
CLOCK #channel	default clock in open channel
CLOCK string	user defined clock in defined position
CLOCK #channel,string	user defined clock in defined channel

CLOCK is a procedure to set up a resident digital clock. If no window is specified, a default window is set up in the top right of the monitor channel 0. This window is 60x20

pixels and is only suitable for four colour mode. The clock may be invoked to execute within a window set up by BASIC. In this case the clock job will be removed when the window is closed.

The string is used to define the characters written to the clock window: any character may be written except '$' or '%'. If a dollar sign is found in the string the next character is checked and:

$d or $D	will insert the first three characters of the name of the day of the week.
$m or $M	will insert the first three characters of the name of the month.

If a percentage sign is found then:

%y or %Y	will insert the two digit year.
%d or %D	will insert the two digit day of month
%h or %H	will insert the two digit hour
%m or %M	will insert the two digit minute
%s or %S	will insert the two digit second

The default string is '**$d %d $m %h/%m/%s** '; a new line should be forced by padding out a line with spaces until the right hand margin of the window is reached.

Example:

```
MODE 8
OPEN #6,'scr_156x10a32x16'
INK #6,0 : PAPER #6,4
CLOCK #6,'QL time %h:%m'
```

4.11 Disc Control

4.11.1 FLP_SEC FLP_START FLP_TRACK

There are three parameters of the floppy disc system which are available as user options.

The security level is selectable to allow a user to choose higher speed of access at the cost of reduced immunity to erroneous disk swapping. There are three security levels, the lowest level still being at least as secure as common disc based operating systems (e.g. MSDOS and CP/M).

FLP_SEC security level

Security Level 0

At this lowest level of security, confusion or loss of data can be expected if a disc is changed while there are still files open or the motor is running.

Security Level 1

At this level of security, discs should only be changed while the motor is stopped (all select lights off). If a disc is changed while there are files open, then read operations will be confused, but any write operations will be aborted. This should maintain the integrity of the data on the disc.

Security Level 2

This is the default security level and data should be quite secure unless a disc is changed while the motors are running.

A user may specify the time taken for the disc drive motor to get the disk speed to within the specification.

FLP_START start_up_time

As a default this is set to .6 second, which is more than enough for most modern drives. The start up time parameter is in 20 millisecond units, so the default value is 30. A value of 13 (260 milliseconds) is adequate for the most recent direct drive 3.5 inch drives, while some older drives may require a value of about 60 (1.2 seconds).

A user may specify the number of tracks to be formatted on a disc.

FLP_TRACK nr_of_tracks

The QL format for discs allows the number of tracks on a disc to be read from the disc itself. However, the number of tracks must be determined when a disc is formatted. Normally the disc system will do this itself by checking if there are at least 55 tracks on a disc. If there are, there are assumed to be 80 tracks, otherwise it is assumed that there are 40 tracks. This internal check may be overriden, allowing 37 track and 75 track drives to be formatted.

SECTION FIVE : RAM Drive

The Q-Disc contains software to utilise spare QL ram as a very high-speed (but limited capacity) directory device. Eight logical drives are supported, named 'ram1_' through 'ram8_'. If required, the drives may be renamed by the **RAM_USE** command:

RAM_USE name

where name is any combination of three letters, e.g. **mdv**, **xyz** or **ram** (to restore to default).

In general, the ram drives can be used in the same way as any other directory device. All of the standard QDOS calls and the Q-Disc extensions are supported. The drives may be used immediately and do not require formatting.

While this is convenient for occasional use with moderate sized files, programs (such as the Psion packages and Metacomco Lisp) which reserve a large amount of memory will rapidly run out of drive space. To reserve space for a ram drive, it is necessary t o **FORMAT** it.

There are two versions of **FORMAT** for a ram drive:

FORMAT RAMdrive_no_

e.g. **FORMAT ram1_**

This version deletes all files on the drive, and releases the memory to Qdos.

FORMAT RAMdrive_no_number_of_blocks

e.g. **FORMAT ram1_42**

This version does not delete files on the drive (if this is required use the first version first), but changes the number of blocks allocated to the drive to that specified in the "drive name". If the amount of space used on the drive exceeds that allocated, it will "overdraw" from QDOS if possible; when space is freed, such as when files are deleted, it is returned to QDOS until the drive is in credit. This is how a drive can be used without being formatted; an unformatted drive is equivalent to one that has been formatted as 'ram1_0'.

It is inadvisable to allocate all of free memory to the RAM drives. The function **FREE_MEM** (q.v.) may be used to obtain the current amount of memory free in bytes. Each block allocated to the ram drive requires 536 bytes. When using the Psion packages, it is useful to insert a line such as:

1 FORMAT ram1_100

into the appropriate boot file. Obviously, the amount of space required depends upon the particular application and the amount of memory fitted to the QL.

Produced by

BRADLEY CONSULTANTS LTD
Technical and Commercia1 Publications

Bradley House, Norton Green Road, Stevenage, Herts. SG1 2BA, England
Telephone: (0438) 315735