QL DISC INTERFACE MANUAL

CST

(Version 1)

CST QL DISC INTERFACE MANUAL

This manual was written and published by Gallivan Communications Cambridge

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 that take three different types of floppy disc:

3 inch, 3 1/2 and 5 1/4 inch

Each of these can be either single or double sided, and 40 or80 track

Note: If your Disc Interface unit was purchased from Computamate, 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 on a separate sheet (If there is not a separate sheet supplied, then follow the instructions outlined below)

If you have Disc Drives bought for use with the BBC Micro, then you may have to have a suitable connector fitted to plug into the Disc Interface unit. (Consult your supplier for advice)

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.

Given below is a brief introduction to Floppy Discs and general use and care of them.

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 with them

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 QL Peripheral Expansion Module, available from Sinclair Research. 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. YOU MUST POWER-UP IN THE FOLLOWING SEQUENCE:

a) First connect the QL to the mains supply

b) Then connect the Disc Drives to the power source (Without a floppy disc in any of the drives)

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/Monltor 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

Random Access I/O

File Handling

Job Control

File Maintenance

Radix Conversions

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.

4.1 The EXTRAS command

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.

4.2 BGET BPUT GET PUT FPOS

File Handling

Direct 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 #1) 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.

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

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$= 'ab258.z'
....
BPUT #3,37,xyz$(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).

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 either

```
      BPUT #3,15,27,45,1
or    PRINT #3,chr$(15);chr$(27);'-';chr$(1);
```

Which is easier?

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.

4.3 FOPEN FOP_IN FOP_OVER FOP_DIR FLEN FTYP FDAT

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 in the same way as EX.

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
```

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 an FPOS function call.

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.**

4.4 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.

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**

4.5 Conversions

A set of 4 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 return a string of sufficient length to represent the value of the specified number_of_bits of the least significant end of the value. In thecase of HEX$ the number_of_bits is rounded up to the nearest multiple of 4

BIN (string)
HEX (string)

each convert 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.