MT.LDD TRAP#1 D0=22

Links a directory device driver into QDOS

Call parameters		Return parameters	
D1		D1	preserved
D2		D2	preserved
D3		D3	preserved
A0	address of link	A0	preserved
A1		A1	undefined
A2		A2	preserved
A3		A3	preserved

Error returns:

none

Description:

see section 5.4.5 for a description of linked lists

MT.RDD TRAP#1 D0=23

Removes a directory device driver from QDOS

Call parameters		Return parameters	
D1		D1	preserved
D2		D2	preserved
D3		D3	preserved
A0	address of link	A0	preserved
A1		A1	undefined
A2		A2	preserved
A3		A3	preserved

Error returns:

none

Description:

see section 5.4.5 for a description of linked lists

6 Input/Output Allocation

6.1 Introduction to I/O

This chapter deals with the allocation of input and output resources on the QL. The input/output resources cover every aspect of communicating with the computer; keyboard, serial ports, screen, microdrives etc. However, it is not necessary to learn a different set of rules for dealing with all of these different devices. The QL has been designed so that communication with all of these devices is carried out in a well structured manner.

Input and output on the QL can be thought of as writing data to and reading data from a *logical file*. This *logical file* is in a standard device independent form. Data which is being sent to a serial port for printing, or to a microdrive for storage is all treated in a similar way. The specific details of interfacing to any particular device are left up to the *device drivers*. Device drivers have been specially written for each of the devices on the QL. The system will automatically find any device drivers which are in the system. These are all in a special device driver format (see section 9.2). Details of input and output are explained in greater detail in chapter 7.

I/O allocation can be divided into one of two general categories. That of opening a file so that data can be input from it and output to it, and that of closing a file. This section explains how to open and close files. It is necessary to know how to specify a channel and all of the information about that channel before it can be opened. Section 6.2 therefore explains the way in which device names should be specified.

The following IO Allocation traps are available:

D0	Name	Description
01	IO.OPEN	Opens a channel
02	IO.CLOSE	Closes a channel
03	IO.FORMT	Formats a sectored medium
04	IO.DELET	Deletes a file

6.2 Specification of device names

On the QL, all input and output is sent to or fetched from a logical device or file. Logical devices and physical devices are not linked, except that a logical device will normally connect to a physical device through a device driver. In order to activate a device, a channel which is linked to that device must first of all be opened. In order to open a channel properly, it is often necessary to provide some extra basic information for the device driver (eg. parity on a serial port or window size for a console). This extra information is appended to the device name. The combination of a device name and appended information is called a filename.

This section covers the way in which device names should be specified.

Logical devices are named using the same conventions as true filenames, so there are certain reserved filenames which cannot be used (like mdv1 or ser1). Within filenames, there is no distinction between upper and lower case letters. The following are standard devices available on the QL.

Standard QL devices

CON_WxHaXxY_K this is the console I/O with window area 'W' by 'H' pixels at the top left hand corner position 'X', 'Y'. 'K' is the number of characters in the keyboard type-ahead buffer. The size and position are defined in terms of pixels on a 512 x 256 display map (position 256 x 128 is the screen centre in both display modes). The width and X position are both specified in multiples of 2 pixels. The default is CON_448x180a32x16_128

SCR_WxHaXxY this is the screen output. The size and position are defined in terms of pixels on a 512 x 256 display map (position 256 x 128 is the screen centre in both display modes). The width and X position are both specified in multiples of 16 pixels. The default is SCR_448x180a32x16

SERnphz serial I/O, with port 'n'. 'p' indicates parity: E, O, M, or S for even, odd, mark or space respectively. 'h' is the handshaking parameter, I for ignore and H for handshake. 'z' indicates protocol: R, Z or C for raw data no EOF, control Z as EOF or control Z as EOF with <CR> as data separator instead of <LF>. The default is SER1R (8 bits no parity)

NETI_nn serial network link from node 'nn'

NETO_nn serial network link to node 'nn'

PIPE_n if 'n' is given, it is an output pipe of length n bytes. Otherwise it is an input pipe connected to the channel ID passed in D3.

MDVn_name microdrive file access, 'n' is microdrive number, *name* is file name

6.3 Filing system names specification

The name specifications outlined in this section are **NOT** implemented on current QLs. There is a possibility that this naming convention will be used at some time in the future, and it is therefore included here for completeness.

6.3.1 Requirements of filing system names

1. File names should be compatible with QL SuperBasic name conventions.

This allows the use of constructions like:

```
OPEN #1,mytext
```

where 'mytext' is the name of the file, as well as

```
OPEN #chan,fname$
```

where the filename is in the string variable 'fname$'.

2. File names should allow the use of directory structures, whether these are actual or simulated.

3. File names should allow the medium or drive to be

explicit (included in the name)
defaulted (system or previously specified default) or
undefined (all drives are searched).

4. File names should allow the automatic creation of related filenames.

6.3.2 Approach to filing system names

1. File names should only consist of letters, digits and underscores.

2. The name should be a series of groups of alphanumeric characters separated by underscores. Each group is regarded as a directory.

3. A Job can provide a default string to the operating system. This default string will then be appended to the start of any filename given in a call to open a file. The filing system will recognise certain strings (such as MDV1) as being physical device names. Such names will not be appended to the default string.

The option to search for a file on all drives (by specifying the medium name for example) could be very expensive in terms of processor overhead. This has not therefore been considered for initial software on the QL.

4. Automatic creation of a filename related to a given filename is usually implemented by adding an *extension* to the end of the name. In QDOS, it isn't possible to distinguish between a file_extension name (such as JIM_BAS) and a directory_file name (such as THISFILE_JIM). Extensions will therefore appear to be files within a directory.

Both the filename and the extension should be supplied by an OPEN call. The system will then deal with this in the best possible way.

6.3.3 Components of filenames

Filenames have the general format of groups of alphanumeric strings connected by underscores.

In the following definitions, '**aaa**' is used to denote an alphanumeric string which starts with a letter.

drive.name is	**aaan**	name as defined by device driver followed by one digit, eg. mdv1
medium.name is	**aaa**	name as defined when medium is formatted.
default is	**drive.name{_aaa}**	
or	**medium.name{_aaa}**	
file.name is	**aaa{_aaa}**	
full.name is	**drive.name_file.name**	
extension is	**aaa**	

Note that in QDOS V1.03, full.name is the only permitted form of filename. Defaults will not be assumed. It is not possible to distinguish between **file.name** and **medium.name_file.name**.

6.4 Opening files

6.4.1 General

There are several methods available for opening files. It is possible to open a file explicitly with an OPEN operation. Alternatively, files will be opened implicitly by either of the `COPY` or `DELETE` operations.

Each open operation is characterised by the state of the file store before the file is opened and the access rights after the file is opened. The state before opening is that of whether the file exists or not. After opening, access right s to the file can be exclusive or shared.

The following table shows the states pertinent to file operations:

Operation	state		access		type
	new	exists	exclusive	shared	
OPEN new	X		X		
OPEN overwrite	X	X	X		
OPEN exclusive		X	X		
OPEN share		X		X	
COPY source		X		X	share
COPY destination	X		X		new
DELETE		X	X		exclusive

Shared files are ones which can be accessed by several Jobs simultaneously. An example of this might be some master database to which several Jobs are referring. *Exclusive* access files can only be accessed by one Job at a time.

6.4.2 Order of search for files

This sub-section covers the order of search for files. It should be noted by readers that this has **NOT** been implemented on current versions of the operating system. Since there is a possibility of it being implemented at some later date, it has been included here for completeness.

There are three possible parts to an open file's name. Each Job defines a default name. As an additional default, an extension may also be supplied (per open call). Finally, either a full name or file name is provided.

When a new file is opened, the name used by the open call is fully defined. In the case of an existing file being opened, the file system attempts to find a file name which corresponds to one out of a number of combinations of the supplied name and the defaults.

The order of search is:

full.name supplied	**file.name** supplied
for new and overwrite:	
1st full.name	1st default_file.name
for exclusive and share	
1st full.name_extension	1st default_file.name_extension
2nd full.name	2nd default_file.name
	3rd file.name_extension
	4th file.name

Examples of opening existing files

Default	filename	extension	Order of search
mdv2	jim	bas	mdv2_jim_bas mdv2_jim *and if jim is a medium:* jim_bas jim
mdv2	fred_data	data	mdv2_fred_data_data mdv2_fred_data *and if fred is a medium:* fred_data_data fred_data
mdv2	mdv1_temp	data	mdv1_temp_data mdv1_temp
mdv2	mdv1_temp	null	mdv1_temp

ser1_utils	backup	exec	ser1_utils_backup_exec ser1_utils_backup *if backup is a medium:* backup_exec backup
ser1_utils	mdv2	null	mdv2 (directory read only)

Note that full.name is the only permitted type of filename in QDOS V1.03

6.5 IO Allocation Traps Reference section

IO.OPEN TRAP#2 D0=1

Open a channel

Call parameters		**Return parameters**	
D1.L	Job ID	D1	Job ID
D2		D2	preserved
D3.L	code where bit:	D3	preserved

0 = old (exclusive) file or device
1 = old (shared) file
2 = new (exclusive) file
3 = new (overwrite) file (not supported for microdrives in QDOS V1.03)
4 = open directory

A0	address of channel name	A0	channel ID
A1		A1	preserved
A2		A2	preserved
A3		A3	preserved

Error returns:

NO not opened because too many channels are open
NJ Job does not exist
OM out of memory
NF file or device not found
EX file already exists
IU file or device in use
BN bad file or device name

Note that any non-zero error code means that a channel hasn't been opened.

Description:

This trap call opens a channel. The file or device name is used to determine the type of device required. Each Job keeps its own list of channels, so the Job number must be provided. If the Job ID is a negative word then the channel will be associated with the current Job.

The file or device name should be stored as a string of ASCII characters. The string is preceded by a word which contains the number of characters in the string. This word is then followed by the characters. The pointer passed in A0 should point to this word (on a word boundary).

A **BN** error return means that the name of the device was recognised but that the parameters were incorrect in some way. For example CON_20y30.

Under normal circumstances, when accessing any non-shared device, the code information is ignored. In practice, this means that the code will only be relevant if a file store is being accessed.

IO.CLOSE TRAP#2 D0=2

Closes a channel

Call parameters		Return parameters	
D1		D1	preserved
D2		D2	preserved
D3		D3	preserved
A0	channel ID	A0	undefined
A1		A1	preserved
A2		A2	preserved
A3		A3	preserved

Error returns:

NO channel is not open

Description:

This trap causes a previously opened file to be closed.

IO.FORMT TRAP#2 D0=3

Format a sectored medium

Call parameters		Return parameters	
D1		D1.W	number of good sectors
D2		D2.W	total number of sectors
D3		D3	preserved
A0	pointer to medium name	A0	undefined
A1		A1	preserved
A2		A2	preserved
A3		A3	preserved

Error returns:

OM out of memory
FF format failed
NF not found
IU in use

Description:

IO.FORMT causes a sectored medium such as a microdrive cartridge to be formatted. The medium name is stored as a character count word followed by ASCII characters for the drive name, the drive number, underscore and then up to ten characters for the medium name. E.g. mdv2_Utilities.

The *total number of sectors* and the *number of good sectors* are reurned from this trap. The total number of sectors is defined by the maximum number which could possibly be fitted onto the medium. Since some parts of the medium could be faulty, the actual number of useable sectors may be less than this total.

IO.DELET TRAP#2 D0=4

Delete a file

Call parameters		Return parameters	
D1.L	Job ID (as file open!)	D1	undefined
D2		D2	preserved
D3		D3	undefined
A0	address of channel name	A0	undefined
A1		A1	undefined
A2		A2	undefined
A3		A3	preserved

Error returns:

NO not opened because too many channels are open
OM out of memory
NF file or device not found
BN bad file or device name

Description:

This trap causes the specified file to be deleted. A0 points to a word containing the number of ASCII characters to follow. These ASCII characters define the name.