The valid pixels are stored in bit 6 to bit 2 (inclusive) of each character definition byte. The top row of each character is left blank, and therefore should not be stored along with the nine printable rows.

If a character is not valid in the first font, it is written using the second font. If it is not valid in the second font either then the lowest valid character in the second font is used.

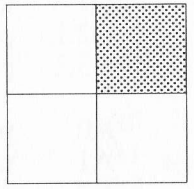
7.3.9 Setting the colours

There are two screen modes available on the QL. Mode 8 is the lower resolution 512x256 mode with a choice of 8 colours plus flashing. Mode 4 is the higher resolution 512x512 mode, which allows 4 colours without flashing. The colour codes are:

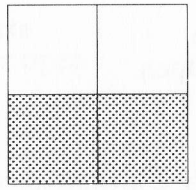
colour	256 mode	512 mode
0	black	black
1	blue	black
2	red	red
3	magenta	red
4	green	green
5	cyan	green
6	yellow	white
7	white	white

In addition to the above colours, stippled areas can be created by putting two colours into a 2x2 pixel square. When placed in a large area, this produces a stippled effect which can be very attractive.

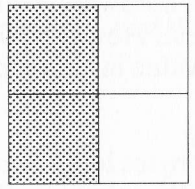
The base colour is always placed in bits 2 to 0 of the colour byte. Bits 5 to 3 contain the XOR of the base colour and the stippling colour in the pattern. If the second colour is zero, displayed colour is then solid. Bits 7 and 6 are used to indicate the pattern as shown below:



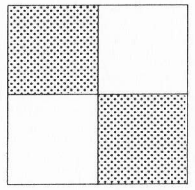
00sssbbb one dot of (sss XOR bbb) per four pixels



01sssbbb horizontal stripes of (sss XOR bbb)



10sssbbb vertical stripes of (sss XOR bbb)



11sssbbb checkerboard pattern

SD.RECOL TRAP#3 D0=26

Recolour a window

Call parameters		Return parameters	
D1		D1	undefined
D2		D2.L	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1	pointer to colour list	A1	undefined
A2		A2	preserved

Error returns:

NC	not complete
NO	channel not open

Description:

The whole of a window can be recoloured without changing the information on it. This consists of changing each of the eight colours to some new colour. Some computers allow screen attributes to be changed in hardware, which allows colours to be changed at will and very fast. However, on the QL this process is much slower because the value in every memory location has to be changed.

The colour change list is 8 bytes long. It contains the new colours required for each of the eight colours in the window. Each of the new colours must be in the range 0 to 7. It is only necessary to fill in bytes 0, 2, 4 and 6 for the 4 colour mode.

SD.SETPA TRAP#3 D0=27

Set the paper colour

Call parameters		Return parameters	
D1.B	colour	D1	preserved
D2		D2	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1		A1	undefined
A2		A2	preserved

Error returns:

NC	not complete
NO	channel not open

Description:

The screen driver uses three colours. This trap is used to set the *paper* colour. The *paper* colour is the background colour of the window, and will appear whenever the window is scrolled, panned or cleared. See also SD.SETST and SD.SETIN for a description of the other colours and their uses.

SD.SETST TRAP#3 D0=28

Set the strip colour

Call parameters		Return parameters	
D1.B	colour	D1	preserved
D2		D2	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1		A1	undefined
A2		A2	preserved

Error returns:

NC	not complete
NO	channel not open

Description:

The screen driver uses three colours. This trap is used to set the *strip* colour. The *strip* colour is used by the character generator to provide a highlighting background for individual characters or words. This colour will therefore appear whenever characters are printed to the screen. See also SD.SETPA and SD.SETIN for a description of the other colours and their uses.

SD.SETIN TRAP#3 D0=29

Set the ink colour

Call parameters		Return parameters	
D1.B	colour	D1	preserved
D2		D2	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1		A1	undefined
A2		A2	preserved

Error returns:

NC	not complete
NO	channel not open

Description:

The screen driver uses three colours. This trap is used to set the *ink* colour. The *ink* colour is used for writing characters and drawing graphics. See also SD.SETST and SD.SETST for a description of the other colours and their uses.

7.3.9 Setting the character attributes

SD.SETFL TRAP#3 D0=2A

Set flash mode

Call parameters		Return parameters	
D1.B	flash attribute	D1	undefined
D2		D2.L	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1		A1	undefined
A2		A2	preserved
A3		A3	

Error returns:

NC	not complete
NO	channel not open

Description:

This trap allows flashing mode to be set for printed characters. If the attribute byte passed in D1 is 0 then printed characters will be in normal non-flashing mode. If D1 is non-zero then subsequently printed characters will flash.

SD.SETUL TRAP#3 D0=2B

Set underline mode

Call parameters		Return parameters	
D1.B	underline attribute	D1	undefined
D2		D2.L	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1		A1	undefined
A2		A2	preserved
A3		A3	

Error returns:

NC	not complete
NO	channel not open

Description:

This trap allows the underline mode to be set for printed characters. If the attribute byte passed in D1 is 0 then printed characters will be in normal non-underlined mode. If D1 is non-zero then subsequently printed characters will be underlined.

SD.SETMD TRAP#3 D0=2C

Sets the character writing or plotting mode

Call parameters		Return parameters	
D1.W	mode	D1	undefined
D2		D2.L	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1		A1	undefined
A2		A2	preserved

Error returns:

NC	not complete
NO	channel not open

Description:

This trap allows the manner in which the character is produced to be defined. The following modes are valid:

D1 = −1	ink is XORed into the background
D1 = 0	character background is strip colour, plotting is in ink colour
D1 = 1	character background is transparent, so it is not overwritten. Plotting is in the ink colour

SD.SETSZ TRAP#3 D0=2D

Sets the character size and spacing

Call parameters		Return parameters	
D1.W	character width/spacing	D1	undefined
D2.W	character height/spacing	D2.L	preserved
		D3.W	timeout
		3.L	reserved
A0	channel ID	A0	preserved
A1		A1	undefined
A2		A2	preserved

Error returns:

NC	not complete
NO	channel not open

Description:

This trap is used to select the character width, height and spacing within the character rectangle. Two standard character widths and heights are supported directly by the character generator. Single width is 5 pixels in a spacing of either 6 or 8 pixels. Double width is 10 pixels in a spacing of either 12 or 16 pixels. Other spacings could be produced by user screen drivers if required, but are not directly supported by this trap. The default single width mode is 5 pixels in a 6 pixel space. The default in the double width mode is 10 pixels in a 12 pixel space. In 8 colour mode, only the double width characters can be used. Calls with D1=0 or 1 in eight colour mode will therefore appear as if a call had been made with D1 equal to 2 or 3. An error will not be generated.

The following are available for character width/spacing:

0	single width (5 pixels) in 6 pixel space
1	single width (5 pixels) in 8 pixel space
2	double width (10 pixels) in 12 pixel space
3	double width (10 pixels) in 16 pixel space

The following are available for character height/spacing:

0	single height (9 pixels) in 10 pixel space
1	double height (18 pixels) in 20 pixels space

7.3.10 Block filling

SD.FILL TRAP#3 D0=2E

Fill rectangular block in window

Call parameters		Return parameters	
D1.B	colour	D1	undefined
D2		D2.L	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1	base of block definition	A1	undefined
A2		A2	preserved

Error returns:

NC	not complete
NO	channel not open
OR	block falls outside window

Description:

This trap modifies a rectangular block inside a window. The definition block consists of 4 words in the following format:

base	block width in pixels
base+2	block height in pixels
base+4	X origin of block (relative to window origin)
base+6	Y origin of block (relative to window origin)

The requested colour is either written directly to the screen (so it covers anything which was there), or is XORed into the screen. The particular manner of filling will be controlled by the current print/plot mode as defined by SD.SETMD (trap#3 with D0=$2C). Stipple colours can also be used. Setting the print/plot mode to −1 (XOR) can be a very fast way of changing all the colours within a particular area of a window.

Useful tip – It is faster to draw *horizontal* and *vertical* lines with this trap than with the ordinary line plotting trap.

7.3.11 Graphics traps

SD.POINT TRAP#3 D0=30

Plot a point

Call parameters		Return parameters	
D1		D1	undefined
D2		D2.L	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1	arithmetic stack pointer	A1	undefined
A2		A2	preserved

Error returns:

NC	not complete
NO	channel not open

Description:

This trap plots a point inside a window. It is the same as the SuperBASIC command POINT in terms of its operation. If the point is outside the window, it is not plotted. The parameters required by the trap should be in floating point format on the arithmetic stack (pointed to by A1). At least 240 bytes should be allocated on this stack before calling the trap. Coordinates are referred to an arbitrary origin (default 0,0) with an arbitrary scale (default is height of window = 100). These can be altered by SD.SCALE.

The parameters required are:

0(A1)	Y coordinate
6(A1)	X coordinate

Note that 6(A1) means 6 bytes away from the point which is addressed by A1.

SD.LINE TRAP#3 D0=31

Plots a line

Call parameters		Return parameters	
D1		D1	undefined
D2		D2.L	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1	arithmetic stack pointer	A1	undefined
A2		A2	preserved

Error returns:

NC	not complete
NO	channel not open

Description:

This trap plots a line inside a window. It is the same as the SuperBASIC command LINE in terms of operation. If any part of the line falls outside the window, it is not plotted. The parameters required by the trap should be in floating point format on the arithmetic stack (pointed to by A1). At least 240 bytes should be allocated on this stack before calling the trap. Coordinates are referred to an arbitrary origin (default 0,0) with an arbitrary scale (default is height of window = 100). These can be altered by SD.SCALE.

The parameters required are:

0(A1)	Y coordinate of end of line
6(A1)	X coordinate of end of line
C(A1)	Y coordinate of start of line
12(A1)	X coordinate of start of line

Note that 6(A1) means 6 bytes away from the point which is addressed by A1.

SD.ARC TRAP#3 D0=32

Plots an arc

Call parameters		Return parameters	
D1		D1	undefined
D2		D2.L	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1	arithmetic stack pointer	A1	undefined
A2		A2	preserved

Error returns:

NC	not complete
NO	channel not open

Description:

This trap plots an arc inside a window. It is the same as the SuperBASIC command ARC in terms of operation. If any part of the arc falls outside of the window, it is not plotted. The parameters required by the trap should be in floating point format on the arithmetic stack (pointed to by A1). At least 240 bytes should be allocated on this stack before calling the trap. Coordinates are referred to an arbitrary origin (default 0,0) with an arbitrary scale (default is height of window = 100). These can be altered by SD.SCALE.

The parameters required are:

0(A1)	angle subtended by arc
6(A1)	Y coordinate of finish of line
C(A1)	X coordinate of finish of line
12(A1)	Y coordinate of start of line
18(A1)	X coordinate of start of line

Note that 6(A1) means 6 bytes away from the point which is addressed by A1.

SD.ELIPS TRAP#3 D0=33

Plot an ellipse

Call parameters		Return parameters	
D1		D1	undefined
D2		D2.L	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1	arithmetic stack pointer	A1	undefined
A2		A2	preserved

Error returns:

NC	not complete
NO	channel not open

Description:

This trap plots an ellipse inside a window. It is the same as the SuperBASIC command CIRCLE in terms of operation. If any part of the ellipse falls outside of the window, it is not plotted. The parameters required by the trap should be in floating point format on the arithmetic stack (pointed to by A1). At least 240 bytes should be allocated on this stack before calling the trap. Coordinates are referred to an arbitrary origin (default 0,0) with an arbitrary scale (default is height of window = 100). These can be altered by SD.SCALE.

The parameters required are:

0(A1)	rotation angle
6(A1)	radius of ellipse
C(A1)	eccentricity of ellipse
12(A1)	Y coordinate of centre
18(A1)	X coordinate of centre

Note that 6(A1) means 6 bytes away from the point which is addressed by A1.

SD.SCALE TRAP#3 D0=34

Sets the scale for plotting

Call parameters		Return parameters	
D1		D1	undefined
D2		D2.L	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1	arithmetic stack pointer	A1	undefined
A2		A2	preserved

Error returns:

NC	not complete
NO	channel not open

Description:

This trap is the same as the SuperBASIC SCALE command. It allows the scale factor used by all of the graphics traps to be altered. Two facets are changed – the graphics origin and the maximum length of the vertical axis. The trap can be used to move the window being used for output around a much larger graphics space. The parameters required by the trap should be in floating point format on the arithmetic stack (pointed to by A1). At least 240 bytes should be allocated on this stack before calling the trap.

The default height is 100 and the default origin is at 0,0.

The parameters required are:

0(A1)	Y coordinate of bottom line of window
6(A1)	X coordinate of lefthand pixel of window
C(A1)	length of Y axis (height of window)

Note that 6(A1) means 6 bytes away from the point which is addressed by A1.

SD.FLOOD TRAP#3 D0=35

Turns area flood on and off

Call parameters		Return parameters	
D1.L	status byte	D1	undefined
D2		D2	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1		A1	undefined
A2		A2	preserved
A3		A3	preserved

Error returns:

NC	not complete
NO	channel not open

Description:

This trap will turn graphics fill on and off in a similar way to the SuperBASIC FILL command. If fill has been turned on, non re-entrant shapes will fill with the ink colour as they are being drawn. Once a shape has been completely encircled, this trap should be called again to start flooding the next enclosed shape. The only special parameter passed to the routine is the status byte. If this is set to 0 then area flood is turned off. If it is set to 1 then area flood is turned on.

SD.GCUR TRAP#3 D0=36

Sets the graphics cursor position

Call parameters		Return parameters	
D1		D1	undefined
D2		D2.L	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1	arithmetic stack pointer	A1	undefined
A2		A2	preserved

Error returns:

NC	not complete
NO	channel not open

Description:

This trap is the same as the SuperBASIC CURSOR command with four parameters. It allows the graphics cursor to be positioned anywhere within the window of the defined channel. The parameters required by the trap should be in floating point format on the arithmetic stack (pointed to by A1). At least 240 bytes should be allocated on this stack before calling the trap. Coordinates are referred to the top lefthand corner of the window as origin. The first two parameters fix the origin of the graphics cursor. The second two parameters position the cursor relative to the first point.

The parameters required are:

0(A1)	graphics X coordinate
6(A1)	graphics Y coordinate
C(A1)	pixel offset to the right
12(A1)	pixel offset downwards

Note that 6(A1) means 6 bytes away from the point which is addressed by A1.

7.4 Filing system traps

The selection of filing system traps included in this section on I/O Utilisation cover general file handling. Opening and closing files was dealt with in chapter 6, whilst reading and writing data from and to files was dealt with in section 7.1. This section deals with particular file structures and utilities.

Under utilities are included operations which are needed to get information about the storage medium, to read and write the file header, and to load or save complete files to and from the memory.

Trap level support for structured files is only provided by the file pointer positioning trap. This pointer can be directed to any byte in a file, from which point a string of bytes of arbitary length can be read (see 7.1). With this facility, high level software can impose any file structure which is required. Database files require some method of ensuring that all data has been written to the desired file, and that it isn't still lurking in memory somewhere. The trap to flush the file buffers can be used to ensure that all data has gone where it was meant to go.

FS.CHECK TRAP#3 D0=40

Check all pending operations on a file

Call parameters		Return parameters	
D1		D1	undefined
D2		D2	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1		A1	undefined
A2		A2	preserved
A3		A3	preserved

Error returns:

NC	not complete
NO	channel not open

Description:

This trap is used to check whether all of the pending operations on a file have been completed. If they are complete, the call returns with no errors; if there are still some pending operations, the call returns not complete. This checking is necessary, because all attempts to read or write to a file or change a file pointer causes implicit buffering. This means that the required physical blocks of the file in question are transferred into the slave block area in the QL. This buffering occurs in the background and continues even if the calls causing the operations return complete.

FS.FLUSH TRAP#3 D0=41

Flush all buffers for this file

Call parameters		Return parameters	
D1		D1	undefined
D2		D2	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1		A1	undefined
A2		A2	preserved
A3		A3	preserved

Error returns:

NC	not complete
NO	channel not open

Description:

This call ensures that any data which may be in the slave blocks area of memory is written to the file concerned. It is very important that this call should be made in cases where it must be known that all of a file has been saved before going on to some other operation. It is not sufficient to rely upon the fact that a write trap returned complete. Such a return does not imply that data has actually been transferred to the relevant file!

FS.POSAB TRAP#3 D0=42

Position file pointer absolute

Call parameters		Return parameters	
D1.L	position in file	D1.L	new file position
D2		D2	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1		A1	undefined
A2		A2	preserved
A3		A3	preserved

Error returns:

NC	not complete
NO	channel not open
EF	end of file

Description:

This trap allows the file pointer to be positioned at a known position in a file. If this trap returns with an off file limits error then the pointer is set to the nearest limit (either 0 or the end of the file).

FS.POSRE TRAP#3 D0=43

Position file pointer relative to previous position

Call parameters		Return parameters	
D1.L	offset to file pointer	D1.L	new file position
D2		D2	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1		A1	undefined
A2		A2	preserved
A3		A3	preserved

Error returns:

NC	not complete
NO	channel not open
EF	end of file

Description:

This trap allows the file pointer to be positioned at a specific location in a file relative to the previous position. If this trap returns with an off file limits error then the pointer is set to the nearest limit (either 0 or the end of the file). By setting the offset to zero, the trap will return with the current file position in D1. Even if the call returns not complete, the pointer will be set, and the background task will continue to load the block containing the next byte into the memory.

FS.MDINF TRAP#3 D0=45

Get information about the medium

Call parameters		Return parameters	
D1		D1.L	empty/good sectors
D2		D2	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1	points to 10 byte buffer	A1	end of medium name
A2		A2	undefined
A3		A3	undefined

Error returns:

NC	not complete
NO	channel not open

Description:

This trap will return the name of the medium, the available space and the capacity. In order to do so, the file or directory must be open. The medium name is 10 bytes long, space filled to the right. A1 should be set to point to 10 bytes of available memory before the call is made. The MSW of D1 contains the number of empty sectors on the medium. The total number of good sectors on the medium is held in the LSW of D1. Sectors are 512 bytes long.

FS.HEADS TRAP#3 D0=46

Set file header

Call parameters		Return parameters	
D1		D1.W	length of header set
D2		D2	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1	base of header block	A1	end of header definition
A2		A2	preserved
A3		A3	preserved

Error returns:

NC	not complete
NO	channel not open

Description:

Every file has a header which contains information about the file. This trap sets the first 14 bytes of the header. The standard header for directory driver files is 64 bytes long and has the following format:

$00	file length (long word)
$04	file access (byte) – set to 0 in QDOS V1.03
$05	file type (byte). For ordinary data files and BASIC programs, the file type is zero. For executable programs the file type is 1.
$06	8 bytes of type dependent information. For executable programs (type 1), the first 4 bytes hold the default size of the data space for the program.
$0E	length of file name
$10	up to 36 characters of file name
$34	reserved for date information (not implemented on early versions of the QL)

Note that when the trap is concerned with a purely serial device, the length of the header in D1 will be spurious (=15). This is because the 14 bytes of the header are preceded by a byte $FF when the header is sent over a serial device.

FS.HEADR TRAP#3 D0=47

Read file header

Call parameters		Return parameters	
D1		D1.W	length of header read
D2.W	buffer length	D2	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1	base of read buffer	A1	top of read buffer
A2		A2	preserved
A3		A3	preserved

Error returns:

NC	not complete
NO	channel not open
BO	buffer overflow

Description:

The read file header call is provided so that a Job can determine the relevant characteristics of a file. Included in these characteristics is the length of the file. This information allows enough space to be allocated (using one of the trap #1s) to load in the file.

A1 is set to point to a read buffer which must be at least 14 bytes long. In the case of a trap to a serial device, the length of the header returned in D1 will be spurious (=15). Upon return from the trap, the buffer will contain the file header information.

The file pointer has been designed so that position zero is the first data byte after the header. Block boundaries on a standard directory driver are therefore at positions 512*n−64.

FS.LOAD TRAP#3 D0=48

Load a file into memory

Call parameters		Return parameters	
D1		D1	undefined
D2.L	length of file	D2	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1	base address for load	A1	top address after load
A2		A2	preserved
A3		A3	preserved

Error returns:

NO	channel not open

Description:

This call allows the whole of a file to be loaded into memory. Before attempting to load a file into memory, enough space must have been reserved for it. If this space is in the transient program area then it must be allocated by a trap#1 before attempting to load. Note that the timeout should be infinite (D3=−1). It is also important to ensure that A1 is an even address because programs can only sit on word boundaries.

FS.SAVE TRAP#3 D0=49

Save file from memory

Call parameters		Return parameters	
D1		D1	undefined
D2.L	length of file	D2	preserved
D3.W	timeout	D3.L	preserved
A0	channel ID	A0	preserved
A1	base address of file	A1	top address of file
A2		A2	preserved
A3		A3	preserved

Error returns:

NO	channel not open
DF	drive full

Description:

This trap allows a file to be saved from memory. D3 should be set to −1 before calling this trap, and A1 must be even so that the program sits on a word boundary when it is reloaded.