SD.CLEAR $20 (32)

Clear entire window

Entry parameters:	D3.W Timeout A0 Channel ID
Return parameters:	none
Affected registers:	D1, A1
Additional errors:	NC (-1) not complete NO (-6) channel not open

Description

This procedure will clear the whole of the specified channel window. Cleared pixel positions will be filled with the 'paper' colour.

SD.CLRTP $21 (33)

Clear top of window

Entry parameters:	D3.W A0	Timeout Channel ID
Return parameters:	none	
Affected registers:	D1, A1	
Additional errors:	NC (-1) not complete NO (-6) channel not open	

Description

This procedure will clear the top part of the specified channel window. Cleared pixel positions will be filled with the 'paper' colour.

The top part of the window is defined as the area of the window above (and not including) the cursor line. The cursor position will not be altered.

SD.CLRBT $22 (34)

Clear bottom of window

Entry parameters:	D3.W	Timeout
	A0	Channel ID
Return parameters:	none	
Affected registers:	D1, A1	
Additional errors:	NC (-1) not complete NO (-6) channel not open	

Description

This procedure will clear the bottom part of the specified channel window. Cleared pixel positions will be filled with the 'paper' colour.

The bottom part of the window is defined as the area of the window below (and not including) the cursor line. The cursor position will not be altered.

SD.CLRLN $23 (35)

Clear cursor line

Entry parameters:	D3.W	Timeout
	A0	Channel ID
Return parameters:	none	
Affected registers:	D1, A1	
Additional errors:	NC (-1) not complete NO (-6) channel not open	

Description

This procedure will clear the whole of the current cursor line in the specified channel window. Cleared pixel positions will be filled with the 'paper' colour.

The height of the cursor line will depend upon the character font size (i.e., either 10 or 20 pixel rows). The cursor position will not be altered.

SD.CLRRT $24 (36)

Clear RHS of cursor line

Entry parameters:	D3.W	Timeout
	A0	Channel ID
Return parameters:	none	
Affected registers:	D1, A1	
Additional errors:	NC (-1)	not complete
	NO (-6)	channel not open

Description

This procedure will clear the whole of the right-hand end of the current cursor line in the specified channel window. Cleared pixel positions will be filled with the 'paper' colour.

The height of the cursor line will depend upon the character font size (i.e., either 10 or 20 pixel rows). The right-hand end includes the character at the current cursor position. The cursor position will not be altered.

SD.FONT $25 (37)

Set/reset character font

Entry parameters:	D3.W	Timeout
	A0	Channel ID
	A1	Base of first font
	A2	Base of second font
Return parameters:	none	
Affected registers:	D1, A1	
Additional errors:	NC (-1) not complete	
	NO (-6) channel not open	

Description

A character font consists of a 5x9 array of pixels within a 6x10 character rectangle. The top row of any character is implicitly blank, as is the right-hand column of the character. Two character fonts already exist within the QL, but other fonts may be selected if desired. The normal font caters for characters in the range $20 to $7F.

If, on entry, the base addresses of the fonts are zero, the default fonts will be used. Switching fonts will not alter the current contents of the screen.

The specified font tables must have the following structure:

Offset	Use
00	**lowest valid character in the set (e.g., $20 <space>)**
01	**number of valid characters - 1**
02 - 0A	**nine bytes specifying first valid character**
0B - 13	**nine bytes specifying next valid character**
14 - 1C	**etc.**

Figure 6.3 illustrates how the nine definition bytes are used to define any one character. Bits 7, 1, and 0 should always be set to zero. Byte 0 should be first in the font definition table.

If a character to be written is found to be illegal (i.e., it is outside the specified range) for the first font, it will be written from the second font. If it is found to be illegal for the second font, the lowest valid character of the second font will be used.

EXAMPLE

If the character shown in Fig.6.3 was the second valid character in the font, the byte definition would be as follows:

Offset	Byte
$0B	$40
$0C	$20
$0D	$10
$0E	$08
$0F	$04
$10	$08
$11	$10
$12	$20
$13	$40

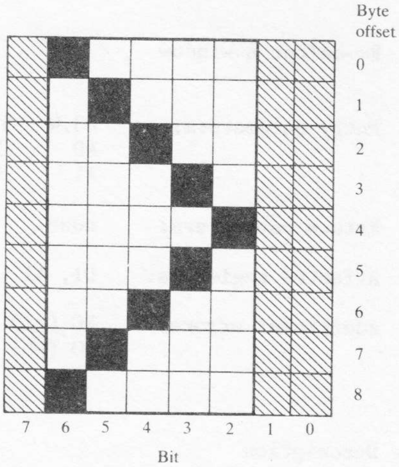

Figure 6.3

Note that you are not redefining a default character set, you are simply setting up an entirely new one. There is no limit, apart from physical memory constraints, to the number of character set fonts that can be defined.

SD.RECOL $26 (38)

Re-colour a window

Entry parameters:	D3.W	Timeout
	A0	Channel ID
	A1	Pointer to colour list
Return parameters:	none	
Affected registers:	D1, A1	
Additional errors:	NC (-1) not complete	
	NO (-6) channel not open	

Description

A window may be re-coloured via this procedure. The display information in the window will not be altered. A maximum of eight colours are permitted for each pixel on any one screen, and on entry to the procedure, register A1 must be pointing to a colour list of eight bytes which defines the new colour (in the range 0..7) for each possible original colour:

Byte offset	Use
0	new colour for black pixel
1	new colour for blue pixel
2	new colour for red pixel
3	new colour for magenta pixel
4	new colour for green pixel
5	new colour for cyan pixel
6	new colour for yellow pixel
7	new colour for white pixel

Two points are worth noting here. First, the above table will only refer to the eight colour mode of screen. The four colour mode screen only requires bytes 0, 2, 4, and 6 to be specified in order to re-colour black, red, green, and white. Second, we are only concerned with re-colouring each individual pixel on the screen. Clearly a pixel cannot have a stipple pattern and, therefore, the range of colours specified must be between 0 and 7. Any stipple patterns on the screen will get re-coloured according to the alteration of pixel colour.

SD.SETPA $27 (39)

Set paper colour

Entry parameters:	D1.B	Colour
	D3.W	Timeout
	A0	Channel ID
Return parameters:	none	
Affected registers:	A1	
Additional errors:	NC (-1) not complete NO (-6) channel not open	

Description

This procedure will set the colour of the paper (i.e., the background colour) for the specified channel window. Any colour may be specified, including stipple colours (see Sec.6.4).

SD.SETST $28 (40)

Set strip colour

Entry parameters:	D1.B	Colour
	D3.W	Timeout
	A0	Channel ID
Return parameters:	none	
Affected registers:	A1	
Additional errors:	NC (-1) not complete	
	NO (-6) channel not open	

Description

This procedure will set the colour of the highlight strip (i.e., the local background when printing characters) for the specified channel window. Any colour may be specified, including stipple colours (see Sec.6.4).

SD.SETIN $29 (41)

Set ink colour

Entry parameters:	D1.B	Colour
	D3.W	Timeout
	A0	Channel ID
Return parameters:	none	
Affected registers:	A1	
Additional errors:	NC (-1) not complete NO (-6) channel not open	

Description

This procedure will set the colour of the ink (i.e., the foreground colour used when printing or plotting) for the specified channel window. Any colour may be specified, including stipple colours (see Sec.6.4).

SD.SETFL $2A (42)

Set/reset flash

Entry parameters:	D1.B	Attribute flag
	D3.W	Timeout
	A0	Channel ID
Return parameters:	none	
Affected registers:	D1, A1	
Additional errors:	NC (-1) not complete	
	NO (-6) channel not open	

Description

This procedure can be used to set (i.e., turn on) or reset (i.e., turn off) the flash mode for the specified window. If the attribute flag on entry is set to zero, flash mode will be suppressed. If the attribute flag is any other value, flash mode will be enabled.

Note that switching flash mode on will only affect subsequent printing (not plotting). Current window contents will not be affected. Likewise when flash mode is switched off, any current flashing items will continue to flash.

SD.SETUL $2B (43)

Set/reset underscore

Entry parameters:	D1.B	Attribute flag
	D3.W	Timeout
	A0	Channel ID
Return parameters:	none	
Affected registers:	D1, A1	
Additional errors:	NC (-1) not complete	
	NO (-6) channel not open	

Description

This procedure can be used to set (i.e., turn on) or reset (i.e., turn off) the underline mode for the specified window. If the attribute flag on entry is set to zero, underline mode will be suppressed. If the attribute flag is any other value, underline mode will be enabled.

Note that switching underline mode on will only affect subsequent printing (not plotting). Current window contents will not be affected. Likewise when underline mode is switched off, any current items underlined will remain as such.

SD.SETMD $2C (44)

Set writing/plotting mode

Entry parameters:	D1.W	Mode byte
	D3.W	Timeout
	A0	Channel ID
Return parameters:	none	
Affected registers:	D1, A1	
Additional errors:	NC (-1) not complete NO (-6) channel not open	

Description

This procedure will affect both subsequent printing and plotting. The mode byte may take one of three values:

Mode	Effect
-1	ink colour is XORed (exclusive-ORed) with the background colour
0	character background becomes strip colour, and plotting is in ink colour
1	character background is transparent (i.e., only the character foreground replaces original screen contents), and plotting is in ink colour.

SD.SETSZ $2D (45)

Set character size

Entry parameters:	D1.W	Character width/space
	D2.W	Character height/space
	D3.W	Timeout
	A0	Channel ID
Return parameters:	none	
Affected registers:	D1, A1	
Additional errors:	NC (-1) not complete	
	NO (-6) channel not open	

Description

The character generator in the QL supports two widths and two heights of character. In eight colour mode only double width, single/double height characters may be used. Additionally, two alternative width spacings are supported directly. In theory the spacing of characters is entirely flexible, but access must be made to the variables SD.XINC and SD.YINC in the window definition block for this flexibility to be fully realized.

The entry parameters for specifying the character width, height, and spacing may be set to any of the following:

D1.W	**Character**	
	Width /	**Spacing**
0	single	6 pixel
1	single	8 pixel
2	double	12 pixel
3	double	16 pixel

D2.W	**Character**	
	Height /	**Spacing**
0	single	10 pixel
1	double	20 pixel

Because of the limitation of only having double width type characters in the eight colour mode, a call to this procedure (while in eight colour mode) with D1 set to 0 or 1 will produce the same effect as if the call had been made with D1 set to 2 or 3 respectively.

Note that if you change the screen mode from four colour to eight colour, or vice versa, the width will automatically switch between double and single as appropriate. Character height will remain unchanged.

SD.FILL $2E (46)

Fill rectangle

Entry parameters:	D1.B	Colour
	D3.W	Timeout
	A0	Channel ID
	A1	Base of definition block
Return parameters:	none	
Affected registers:	D1, A1	
Additional errors:	NC (-1)	not complete
	OR (-4)	range error - block outside window
	NO (-6)	channel not open

Description

This procedure will fill a rectangular block, in the specified channel window, with the specified colour. Any colour may be chosen, including stipple colours (see Sec.6.4).

On entry to the procedure, register A1 must hold a pointer to a four word definition block that defines the size and position of the rectangle:

Offset	Use
0	width of rectangle
2	height of rectangle
4	X position of top left-hand corner
6	Y position of top left-hand corner

The X and Y coordinates are specified relative to the origin of the window.

This procedure can be used to provide a fast horizontal or vertical line drawing operation. Additionally, it may be used as a fast 're-colour block within window' procedure, by setting the character/plotting mode to -1 (XOR mode) before calling it (see SD.SETMD; D0=$2C).

SD.POINT $30 (48)

Plot a point

Entry parameters:	D3.W	Timeout
	A0	Channel ID
	A1	Arithmetic stack pointer
Return parameters:	none	
Affected registers:	D1, A1	
Additional errors:	NC (-1)	not complete
	NO (-6)	channel not open

Description

On entry to this procedure, register A1 must point to a local arithmetic parameter stack of at least 240 bytes. At the top of the stack there should be two floating point parameters (each of six bytes) as follows:

Stack position	Use
0(A1)	Y coordinate of point
6(A1)	X coordinate of point

Remember that stacks grow downwards and therefore the top of the stack is at a physically lower address than the bottom of the stack.

The coordinates referred to are relative to an arbitrary origin (default [0,0]) and an arbitrary scale (default 0..100). Any point which lies outside the specified channel window will not be plotted. No error is returned in such cases.

SD.LINE $31 (49)

Plot a line

Entry parameters:	D3.W	Timeout
	A0	Channel ID
	A1	Arithmetic stack pointer
Return parameters:	none	
Affected registers:	D1, A1	
Additional errors:	NC (-1) not complete	
	NO (-6) channel not open	

Description

On entry to this procedure, register A1 must point to a local arithmetic parameter stack of at least 240 bytes. At the top of the stack there should be four floating point parameters (each of six bytes) as follows:

Stack position	Use
00(A1)	Y coordinate of end of line
06(A1)	X coordinate of end of line
0C(A1)	Y coordinate of start of line
12(A1)	X coordinate of start of line

Remember that stacks grow downwards and therefore the top of the stack is at a physically lower address than the bottom of the stack.

The coordinates referred to are relative to an arbitrary origin (default [0,0]) and an arbitrary scale (default 0..100). Any part of a line which lies outside the specified channel window will not be plotted. No error is returned in such cases.

SD.ARC $32 (50)

Plot an arc

Entry parameters:	D3.W	Timeout
	A0	Channel ID
	A1	Arithmetic stack pointer
Return parameters:	none	
Affected registers:	D1, A1	
Additional errors:	NC (-1) not complete	
	NO (-6) channel not open	

Description

On entry to this procedure, register A1 must point to a local arithmetic parameter stack of at least 240 bytes. At the top of the stack there should be five floating point parameters (each of six bytes) as follows:

Stack position	Use
00(A1)	angle subtended by the arc
06(A1)	Y coordinate of end of arc
0C(A1)	X coordinate of end of arc
12(A1)	Y coordinate of start of arc
18(A1)	X coordinate of start of arc

Remember that stacks grow downwards and therefore the top of the stack is at a physically lower address than the bottom of the stack.

The coordinates referred to are relative to an arbitrary origin (default [0,0]) and an arbitrary scale (default 0..100). Any part of the arc which lies outside the specified channel window will not be plotted. No error is returned in such cases.

SD.ELIPS $33 (51)

Plot an ellipse

Entry parameters:	D3.W	Timeout
	A0	Channel ID
	A1	Arithmetic stack pointer
Return parameters:	none	
Affected registers:	D1, A1	
Additional errors:	NC (-1) not complete	
	NO (-6) channel not open	

Description

On entry to this procedure, register A1 must point to a local arithmetic parameter stack of at least 240 bytes. At the top of the stack there should be five floating point parameters (each of six bytes) as follows:

Stack position	Use
00(A1)	rotation angle
06(A1)	radius of ellipse
0C(A1)	eccentricity of ellipse
12(A1)	Y coordinate of centre
18(A1)	X coordinate of centre

Remember that stacks grow downwards and therefore the top of the stack is at a physically lower address than the bottom of the stack.

The coordinates referred to are relative to an arbitrary origin (default [0,0]) and an arbitrary scale (default 0..100). Any point on the ellipse which lies outside the specified channel window will not be plotted. No error is returned in such cases.

SD.SCALE $34 (52)

Set scale

Entry parameters:	D3.W	Timeout
	A0	Channel ID
	A1	Arithmetic stack pointer
Return parameters:	none	
Affected registers:	D1, A1	
Additional errors:	NC (-1) not complete	
	NO (-6) channel not open	

Description

On entry to this procedure, register A1 must point to a local arithmetic parameter stack of at least 240 bytes. At the top of the stack there should be three floating point parameters (each of six bytes) as follows:

Stack position	Use
00(A1)	Y position of bottom line of window
06(A1)	X position of left-hand pixel of window
0C(A1)	length of Y axis (height of window)

Remember that stacks grow downwards and therefore the top of the stack is at a physically lower address than the bottom of the stack.

This procedure sets the arbitrary scale for use for all subsequent graphic plotting calls.

SD.FLOOD $35 (53)

Set/reset area flood

Entry parameters:	D1.L	Flood flag
	D3.W	Timeout
	A0	Channel ID
Return parameters:	none	
Affected registers:	D1, A1	
Additional errors:	NC (-1) not complete	
	NO (-6) channel not open	

Description

This procedure can be used to set (i.e., turn on), or reset (i.e., turn off) the area flood mode for the specified window. If the attribute flag on entry is set to zero, area flood will be suppressed. If the attribute flag is set to 1, area flood will be enabled.

Note that switching flood mode on will only affect subsequent plotting. Current window contents will not be affected. Likewise when flood mode is switched off, any currently filled areas will remain as such.

SD.GCUR $36 (54)

Set graphic cursor position

Entry parameters:	D3.W	Timeout
	A0	Channel ID
	A1	Arithmetic stack pointer
Return parameters:	none	
Affected registers:	D1, A1	
Additional errors:	NC (-1) not complete NO (-6) channel not open	

Description

On entry to this procedure, register A1 must point to a local arithmetic parameter stack of at least 240 bytes. At the top of the stack there should be four floating point parameters (each of six bytes) as follows:

Stack position	Use
00(A1)	graphics X coordinate
06(A1)	graphics Y coordinate
0C(A1)	pixel offset to right
12(A1)	pixel offset downwards

Remember that stacks grow downwards and therefore the top of the stack is at a physically lower address than the bottom of the stack.

The coordinates referred to are relative to an arbitrary origin (default [0,0]) and an arbitrary scale (default 0..100).

FS.CHECK $40 (64)

Check pending file operations

Entry parameters:	D3.W	Timeout
	A0	Channel ID
Return parameters:	none	
Affected registers:	D1, A1	
Additional errors:	NC (-1) not complete	
	NO (-6) channel not open	

Description

Implicit buffering operations will be carried out by the standard device drivers whenever a file is read from, written to, or had its file pointer adjusted. These buffering operations, which are carried out in the background and which continue even if the calls which invoked them return 'not complete', will cause physical blocks of a file to be loaded into the slave block area (see Chapter 3).

This TRAP procedure can be used to check whether all of the pending operations have been completed.

FS.FLUSH $41 (65)

Flush file buffers

Entry parameters:	D3.W A0	Timeout Channel ID
Return parameters:	none	
Affected registers:	D1, A1	
Additional errors:	NC (-1) not complete NO (-6) channel not open	

Description

This procedure does not flush the specified file buffer in the sense of throwing the contents of the buffer away. When write operations to a file are complete, as far as an applications program is concerned, the data written may still be in the internal slave blocks rather than physically in the backing store file.

This procedure can, therefore, be used to ensure that all data are written out to the physical device file. It could be called during file operations for the purposes of security, but closing a channel will flush the buffers anyway.

FS.POSAB $42 (66)

Set file pointer absolute

Entry parameters:	D1.L	File pointer position
	D3.W	Timeout
	A0	Channel ID
Return parameters:	D1.L	New file pointer position
Affected registers:	D1, A1	
Additional errors:	NC (-1)	not complete
	NO (-6)	channel not open
	EF (-10)	end of file

Description

This procedure enables byte positioning of a file pointer. The specified position is absolute.

The error ERR.EF (-10) will be returned if the pointer is set either to a position before the start, or to a position after the end. In both cases the pointer will be set to the respective extreme limit (i.e., 0 or end of file).

FS.POSRE $43 (67)

Set file pointer relative

Entry parameters:	D1.L	Relative file pointer position
	D3.W	Timeout
	A0	Channel ID
Return parameters:	D1.L	New file pointer position
Affected registers:	D1, A1	
Additional errors:	NC (-1)	not complete
	NO (-6)	channel not open
	EF (-10)	end of file

Description

This procedure enables byte positioning of a file pointer. The specified position is relative to the current position. Note that the procedure could be used to obtain the current absolute position of the file pointer, simply by calling it with D1 set to zero.

The error ERR.EF (-10) will be returned if the pointer is set either to a position before the start, or to a position after the end. In both cases the pointer will be set to the respective extreme limit (i.e., 0 or end of file).

FS.MDINF $45 (69)

Get medium information

Entry parameters:	D3.W	Timeout
	A0	Channel ID
	A1	Pointer to buffer
Return parameters:	D1.L	Sector count
	A1	Pointer to end of buffer
Affected registers:	D1, A1 - A3	
Additional errors:	NC (-1)	not complete
	NO (-6)	channel not open

Description

Provided a file or device is open, information about the medium may be obtained using this procedure.

On entry, register A1 should point to the beginning of a 10-byte buffer that will be used to store the medium name.

On return, register A1 will point to the byte following the end of the specified buffer. Also, the most significant word of D1 will hold a count of the number of empty sectors, and the least significant word of D1 will hold a count of the total number of good sectors.

A sector holds 512 bytes.

FS.HEADS $46 (70)

Set file header

Entry parameters:	D3.W	Timeout
	A0	Channel ID
	A1	Base of header table
Return parameters:	D1.W	Length of header set
	A1	Pointer to end of header table
Affected registers:	D1, A1	
Additional errors:	NC (-1)	not complete
	NO (-6)	channel not open

Description

Each file has a header containing information about the file. For standard files that can be listed by a directory command this header is 64 bytes long, as follows:

Offset	Use
00	file length (long-word)
04	file access byte
05	file type byte
06	eight bytes of type information
0E	length of file name (word)
10	file name (maximum 36 bytes)
34	reserved

The access byte is normally set to zero. For data files and SuperBASIC programs the file type is also zero. Executable programs have a file type of 1, and the first four bytes of the type information field will contain the default size of the data space for the program.

This procedure enables the first 14 ($0E) bytes of the header to be set. The filing system will normally, however, overwrite the file length parameter. The procedure cannot be used to rename the file.

The length of the header set, as returned by the procedure, will be spurious if the channel ID refers to a pure serial device. Also, when the header is sent over such a device, the 14 bytes will be preceded by the byte $FF (255).

FS.HEADR $47 (71)

Read file header

Entry parameters:	D2.W	Buffer length
	D3.W	Timeout
	A0	Channel ID
	A1	Base of buffer
Return parameters:	D1.W	Length of header read
	A1	Top of read buffer
Affected registers:	D1, A1	
Additional errors:	NC (-1)	not complete
	BO (-5)	buffer overflow
	NO (-6)	channel not open

Description

Each file has a header containing information about the file. For standard files that can be listed by a directory command this header is 64 bytes long, as follows:

Offset	Use
00	**file length (long-word)**
04	**file access byte**
05	**file type byte**
06	**eight bytes of type information**
0E	**length of file name (word)**
10	**file name (maximum 36 bytes)**
34	**reserved**

The access byte is normally set to zero. For data files and SuperBASIC programs the file type is also zero. Executable programs have a file type of 1, and the first four bytes of the type information field will contain the default size of the data space for the program.

This procedure enables a specified buffer length of bytes (or the first 14 ($0E) bytes in the case of a serial device) of the header to be read. The information provided is useful for allocating space for a file load operation, and also for determining certain characteristics of files. The buffer length must be at least 14 bytes.

Position zero for a file pointer is at the first byte following the header block. The header block is part of a normal sector block and therefore sector block boundaries will be found at file positions 512*n-64.

The length of the header set, as returned by the procedure, will be spurious if the channel ID refers to a pure serial device.

FS.LOAD $48 (72)

Load a file

Entry parameters:	D2.L	Length of file
	D3.W	Timeout
	A0	Channel ID
	A1	Base address for load
Return parameters:	A1	Top address after load
Affected registers:	D1, A1	
Additional errors:	NO (-6)	channel not open

Description

This procedure will transfer a file into memory in its entirety. If the transient program area is being used for the load, enough space must have been allocated (via TRAP #1, MT.CJOB; D0=1) prior to the call.

On entry, register D3 should be set to -1 (indefinite timeout), and the base address specified by A1 must be even.

When the procedure returns, register A1 will point to the byte following the last byte loaded.

FS.SAVE $49 (73)

Save a file

Entry parameters:	D2.L	Length of file
	D3.W	Timeout
	A0	Channel ID
	A1	Base address of file
Return parameters:	A1	Top address of file
Affected registers:	D1, A1	
Additional errors:	NO (-6)	channel not open

Description

This procedure will transfer a file from memory to backing store in its entirety. On entry, register D3 should be set to -1 (indefinite timeout), and the base address specified by A1 must be even.

When the procedure returns, register A1 will point to the byte following the last byte saved.