LES VECTEURS SYSTÈMES

AMSTRAD CPC 6128

Liste complète des vecteurs systèmes

INDEX

THE KEY MANAGER

The **Key Manager** is the pack associated with the keyboard. All the attributes of the keyboard are generated and controlled by the Key Manager. These attributes include repeat speed, shift and control keys, function keys and key translation. The joysticks are also scanned by the Key Manager.

&BB00 - KM INITIALISE

Initialises the Key Manager and sets up everything as it is when the computer is first switched on:

- the key buffer is emptied
- Shift and Caps lock are turned off
- All the expansion and translation tables are reset to normal

See also the routine **KM RESET** below.

EXIT

- **AF**, **BC**, **DE** and **HL** corrupt.
- All other registers are preserved.

&BB03 - KM RESET

Resets the Key Manager. The key buffer is emptied and all current keys/characters are ignored.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt.
- All other registers are preserved.
-

! See also KM INITIALISE above: on the 664 or 6128, the key buffer can also be cleared separately by calling the KM FLUSH routine.

&BB06 - KM WAIT CHAR

Waits for the next character from the keyboard buffer.

EXIT

- Carry is true
- **A** holds the character value
- The other flags are corrupt
- All other registers are preserved.

&BB09 - KM READ CHAR

Tests to see if a character is available from the keyboard buffer, but doesn't wait for one to become available.

EXIT

- If a character was available, then Carry is true, and **A** contains the character
- otherwise Carry is false, and **A** is corrupt.

In both cases, the other registers are preserved.

&BB0C - KM CHAR RETURN

Saves a character for the next use of **KM WAIT CHAR** or **KM READ CHAR**.

ENTRY

- **A** contains the ASCII code of the character to be put back.

EXIT

- All registers are preserved.

&BB0F - KM SET EXPAND

Assigns a string to a key code.

ENTRY

- **B** holds the key code.
- **C** holds the length of the string.
- **HL** contains the address of the string (must be in RAM).

EXIT

- If it is OK, then Carry is true, otherwise Carry is false.

In either case:

- **A, BC, DE** and **HL** are corrupt.
- All other registers are preserved.

&BB12 - KM GET EXPAND

Reads a character from an expanded string of characters.

ENTRY

- **A** holds an expansion token (ie a key code).
- **L** holds the character position number (starts from 0).

EXIT

- If it is OK, then Carry is true, and **A** holds the character
- otherwise Carry is false, and **A** is corrupt.

In either case:

- **DE** and flags are corrupt.
- The other registers are preserved.

&BB15 - KM EXP BUFFER

Sets aside a buffer area for character expansion strings.

ENTRY

- **DE** holds the address of the buffer.
- **HL** holds the length of the buffer.

EXIT

- If it is OK, then Carry is true; otherwise Carry is false.*

In either case, **A, BC, DE** and **HL** are corrupt.

! The buffer must be in the central 32K of RAM and must be at least 49 bytes long.

&BB18 - KM WAIT KEY

Waits for a key to be pressed. This routine does not expand any expansion tokens.

EXIT

- Carry is true.
- **A** holds the character or expansion token
- All other registers are preserved.

&BB1B - KM READ KEY

Tests whether a key is available from the keyboard.

EXIT

- If a key is available, then Carry is true, and **A** contains the character.
- Otherwise Carry is false, and **A** is corrupt.

In either case, the other registers are preserved.

! Any expansion tokens are not expanded.

&BB1E - KM TEST KEY

Tests if a particular key [or joystick direction or button] is pressed.

ENTRY

- **A** contains the key/joystick number.

EXIT

- If the requested key is pressed, then Zero is false, otherwise Zero is true

In either case:

- Carry is false.
- **A** and **HL** are corrupt.
- **C** holds the Shift and Control status.
- All other registers are preserved.

! After calling this, C will hold the state of shift and control:
if bit 7 is set then Control was pressed.
if bit 5 is set then Shift was pressed.

&BB21 - KM GET STATE

Gets the state of the Shift and Caps locks.

EXIT

- If **L** holds **&FF** then the shift lock is **on**.
- If **L** holds **&00** then the Shift lock is **off**.
- If **H** holds **&FF** then the caps lock is **on**.
- If **H** holds **&00** then the Caps lock is **off**.

Whatever the outcome, all the other registers are preserved.

&BB24 - KM GET JOYSTICK

Reads the present state of any joysticks attached.

EXIT

- **H** and **A** contains the state of joystick 0.
- **L** holds that state of joystick 1.
- All other registers are preserved.

Notes

The joystick states are bit significant and are as follows:

- Bit 0 - Up
- Bit 1 - Down
- Bit 2 - Left
- Bit 3 - Right
- Bit 4 - Fire2
- Bit 5 - Fire1
- Bit 6 - Spare
- Bit 7 - Always zero

! The bits are set when the corresponding buttons or directions are operated.

&BB27 - KM SET TRANSLATE

Sets the token or character that is assigned to a key when **neither Shift nor Control** are pressed.

ENTRY

- **A** contains the key number.
- **B** contains the new token or character.

EXIT

- **AF** and **HL** are corrupt.
- All other registers are preserved.

Notes

Special values for **B** are as follows:

- **&80** to **&9F**: these values correspond to the expansion tokens.
- **&FD**: this causes the caps lock to toggle on and off.
- **&FE**: this causes the shift lock to toggle on and off.
- **&FF**: causes this key to be ignored.

&BB2A - KM GET TRANSLATE

Finds out what token or character will be assigned to a key when **neither Shift nor Control** are pressed.

ENTRY

- **A** contains the key number.

EXIT

- **A** contains the token / character that is assigned.
- **HL** and flags are corrupt.
- All other registers are preserved.

See **KM SET TRANSLATE** for special values that can be returned.

&BB2D - KM SET SHIFT

Sets the token or character that will be assigned to a key when **Shift is pressed** as well.

ENTRY

- **A** contains the key number.
- **B** contains the new token or character.

EXIT

- **AF** and **HL** are corrupt
- All other registers are preserved.

Notes

See KM SET TRANSLATE for special values that can be set.

&BB30 - KM GET SHIFT

Finds out what token/character will be assigned to a key when **Shift is pressed** as well.

ENTRY

- **A** contains the key number.

EXIT

- **A** contains the token/character that is assigned
- **HL** and flags are corrupt
- All other registers are preserved.

Notes

See **KM SET TRANSLATE** for special values that can be returned.

&BB33 - KM SET CONTROL

Sets the token or character that will be assigned to a key when **Control is pressed** as well.

ENTRY

- **A** contains the key number and **B** contains the new token/character.

EXIT

- **AF** and **HL** are corrupt.
- All other registers are preserved.

Notes

See **KM SET TRANSLATE** for special values that can be set.

&BB36 - KM GET CONTROL

Finds out what token or character will be assigned to a key when **Control is pressed** as well.

ENTRY

- **A** contains the key number.

EXIT

- **A** contains the token/character that is assigned.
- **HL** and flags are corrupt.
- All other registers are preserved.

Notes

See **KM SET TRANSLATE** for special values that can be set.

&BB39 - KM SET REPEAT

Sets whether a key may repeat or not.

ENTRY

- **A** contains the key number.
- **B** contains &00 if there is no repeat and &FF if it is to repeat.

EXIT

- **AF**, **BC** and **HL** are corrupt
- All other registers are preserved.

&BB3C - KM GET REPEAT

Finds out whether a key is set to repeat or not.

ENTRY

- **A** contains a key number.

EXIT

- If the key repeats, then Zero is false.
- If the key does not repeat, then Zero is true.

In either case:

- **A**, **HL** and flags are corrupt.
- Carry is false.
- All other registers are preserved.

&BB3F - KM SET DELAY

Sets the time that elapses before the first repeat, and also set the repeat speed.

ENTRY

- **H** contains the time before the first repeat.
- **L** holds the time between repeats (repeat speed).

EXIT

- **AF** is corrupt.
- All other registers are preserved.

! The values for the times are given in 1/50th seconds, and a value of 0 counts as 256.

&BB42 - KM GET DELAY

Finds out the time that elapses before the first repeat and also the repeat speed.

EXIT

- **H** contains the time before the first repeat
- **L** holds the time between repeats.
- All others are preserved.

&BB45 - KM ARM BREAK

Arms the Break mechanism.

ENTRY

- **DE** holds the address of the Break handling routine.
- **C** holds the ROM select address for this routine.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt.
- All the other registers are preserved.

&BB48 - KM DISARM BREAK

Disables the Break mechanism.

EXIT

- **AF** and **HL** are corrupt.
- All the other registers are preserved.

&BB4B - KM BREAK EVENT

Generates a Break interrupt if a Break routine has been specified by **KM ARM BREAK**.

EXIT

- **AF** and **HL** are corrupt.
- All other registers are preserved.

THE TEXT VDU

The **Text VDU** is a character based screen driver. It controls 8 different streams each of which can have an area of screen allocated to it (a window). The **Text VDU** allows characters to be written to the screen and read from the screen. It also treats certain 'characters' as 'control codes' which can have various effects, from moving the cursor to setting the colour of an ink.

&BB4E - TXT INITIALISE

Initialise the text VDU to its settings when the computer is switched on:

- resetting all the text VDU indirections.
- selecting Stream 0.
- resetting the text paper to pen 0 and the text pen to pen 1.
- moving the cursor to the top left corner of the screen and setting the writing mode to be opaque.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BB51 - TXT RESET

Resets the text VDU indirections and the control code table.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BB54 - TXT VDU ENABLE

Allows characters to be printed on the screen in the current stream.

EXIT

- **AF** is corrupt, and all other registers are preserved.

&BB57 - TXT VDU DISABLE

Prevents characters from being printed to the current stream.

EXIT

- **AF** is corrupt, and all the other registers are preserved.

&BB5A - TXT OUTPUT

Output a character or control code (&00 to &1F) to the screen.

ENTRY

- **A** contains the character to output.

EXIT

- All registers are preserved.

Notes

Any control codes are obeyed and nothing is printed if the VDU is disabled; characters are printed using the **TXT OUT ACTION** indirection. If using graphics printing mode, then control codes are printed and not obeyed.

&BB5D - TXT WR CHAR

Print a character at the current cursor position - control codes are printed and not obeyed.

ENTRY

- **A** contains the character to be printed.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all others are preserved.

This routine uses the **TXT WRITE CHAR** indirection to put the character on the screen.

&BB60 - TXT RD CHAR

Read a character from the screen at the current cursor position.

EXIT

- If it was successful then **A** contains the character that was read from the screen and Carry is true;
- Otherwise Carry is false, and **A** holds **0**.

In either case, the other flags are corrupt, and all registers are preserved.

Notes

This routine uses the **TXT UNWRITE** indirection.

&BB63 - TXT SET GRAPHIC

Enables or disables graphics print character mode.

ENTRY

- To switch graphics printing mode on, **A** must be non-zero.
- To turn it off, **A** must contain zero.

EXIT

- **AF** corrupt, and all other registers are preserved.
-

Notes

When turned on, control codes are printed and not obeyed; characters are printed by **GRA WR CHAR**.

&BB66 - TXT WIN ENABLE

Sets the boundaries of the current text window - uses physical coordinates.

ENTRY

- **H** holds the column number of one edge.
- **D** holds the column number of the other edge.
- **L** holds the line number of one edge.
- **E** holds the line number of the other edge.

EXIT

- **AF, BC, DE** and **HL** are corrupt.
-

Notes

The window is not cleared but the cursor is moved to the top left corner of the window.

&BB69 - TXT GET WINDOW

Returns the size of the current window - returns physical coordinates.

EXIT

- **H** holds the column number of the left edge.
- **D** holds the column number of the right edge.
- **L** holds the line number of the top edge.
- **E** holds the line number of the bottom edge.
- **A** is corrupt.
- Carry is false if the window covers the entire screen, and the other registers are always preserved.

&BB6C - TXT CLEAR WINDOW

Clears the window (of the current stream) and moves the cursor to the top left corner of the window.

EXIT

- **AF, BC, DE** and **HL** are corrupt, and all others are preserved.

&BB6F - TXT SET COLUMN

Sets the cursor's horizontal position.

ENTRY

- **A** contains the logical column number to move the cursor to.

EXIT

- **AF** and **HL** are corrupt, and all the other registers are preserved.

Notes

See also **TXT SET CURSOR**.

&BB72 - TXT SET ROW

Sets the cursor's vertical position.

ENTRY

- **A** contains the logical line number to move the cursor to.

EXIT

- **AF** and **HL** are corrupt, and all other registers are preserved.

Notes

See also **TXT SET CURSOR**.

&BB75 - TXT SET CURSOR

Sets the cursor's vertical and horizontal position.

ENTRY

- **H** contains the logical column number.
- **L** contains the logical line number.

EXIT

- **AF** and **HL** are corrupt, and all other registers are preserved.

Notes

See also **TXT SET COLUMN** and **TXT SET ROW**.

&BB78 - TXT GET CURSOR

Gets the cursor's current position.

EXIT

- **H** holds the logical column number.
- **L** holds the logical line number.
- **A** contains the roll count.
- The flags are corrupt, and all the other registers are preserved.

Notes

The roll count is increased when the screen is scrolled down, and is decreased when it is scrolled up.

&BB7B - TXT CUR ENABLE

Allows the text cursor to be displayed (if it is allowed by **TXT CUR ON**) - intended for use by the user.

EXIT

- **AF** is corrupt, and all other registers are preserved.

&BB7E - TXT CUR DISABLE

Prevents the text cursor from being displayed -intended for use by the user.

EXIT

- **AF** is corrupt, and all others are preserved.

&BB81 - TXT CUR ON

Allows the text cursor to be displayed - intended for use by the operating system.

EXIT

- All registers and flags are preserved.

&BB84 - TXT CUR OFF

Prevents the text cursor from being displayed -intended for use by the operating system.

EXIT

- All registers and flags are preserved.

&BB87 - TXT VALIDATE

Checks whether a cursor position is within the current window.

ENTRY

- **H** contains the logical column number to check.
- **L** holds the logical line number.

EXIT

- **H** holds the logical column number where the next character will be printed.
- **L** holds the logical line number.
- If printing at this position would make the window scroll up, then Carry is false and **B** holds **&FF**.
- If printing at this position would make the window scroll down, then Carry is false and **B** contains **&00**.
- If printing at the specified cursor position would not scroll the window, then Carry is true and **B** is corrupt.
- **A** and the other flags are corrupt, and all other registers are preserved.

&BB8A - TXT PLACE CURSOR

Puts a '<u>cursor blob</u>' on the screen at the current cursor position.

EXIT

- **AF** is corrupt, and all other registers are preserved.

<u>Notes</u>

It is possible to have more than one cursor in a window [see also **TXT DRAW CURSOR**]; do not use this routine twice without using **TXT REMOVE CURSOR** between.

&BB8D - TXT REMOVE CURSOR

Removes a 'cursor blob' from the current cursor position.

EXIT

- **AF** is corrupt, and all the others are preserved.

Notes

This should be used only to remove cursors created by **TXT PLACE CURSOR**, but see also **TXT UNDRAW CURSOR**.

&BB90 - TXT SET PEN

Sets the foreground PEN for the current stream.

ENTRY

- **A** contams the PEN number to use.

EXIT

- **AF** and **HL** are corrupt, and all other registers are preserved.

&BB93 - TXT GET PEN

Gets the foreground PEN for the current stream.

EXIT

- **A** contains the PEN number.
- The flags are corrupt, and all other registers are preserved.

&BB96 - TXT SET PAPER

Sets the background PAPER for the current stream.

ENTRY

- **A** contains the PEN number to use.

EXIT

- **AF** and **HL** are corrupt, and all other registers are preserved.

&BB99 - TXT GET PAPER

Gets the background PAPER for the current stream.

EXIT

- **A** contains the PEN number
- The flags are corrupt, and all other registers are preserved.

&BB9C - TXT INVERSE

Swaps the current PEN and PAPER colours over for the current stream.

EXIT

- **AF** and **HL** are corrupt, and all others are preserved.

&BB9F - TXT SET BACK

Sets the character write mode to either opaque or transparent.

ENTRY

- For transparent mode, **A** must be non-zero.
- For opaque mode, **A** has to hold zero.

EXIT

- **AF** and **HL** are corrupt, and all other registers are preserved.

Notes

Setting the character write mode has no effects on the graphics VDU.

&BBA2 - TXT GET BACK

Gets the character write mode for the current stream.

EXIT

- If in transparent mode, **A** is non-zero.
- If in opaque mode, **A** is zero.
- In either case **DE**, **HL** and flags are corrupt, and the other registers are preserved.

&BBA5 - TXT GET MATRIX

Gets the address of a character matrix.

ENTRY

- **A** contains the character whose matrix is to be found.

EXIT

- If it is a user-defined matrix, then Carry is true.
- If it is in the lower ROM then Carry is false.

In either case:

- **HL** contains the address of the matrix.
- **A** and other flags are corrupt, and others are preserved.
-

Notes

The character matrix is stored in 8 bytes; the first byte is for the top row of the character, and the last byte refers to the bottom row of the character; bit 7 of a byte refers to the leftmost pixel of a line, and bit 0 refers to the rightmost pixel in Mode 2..

&BBA8 - TXT SET MATRIX

Installs a matrix for a user-defined character.

ENTRY

- **A** contains the character which is being defined.
- **HL** contains the address of the matrix to be used.

EXIT

- If the character is user-definable then Carry is true; otherwise Carry is false, and no action is taken.
- In both cases **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BBAB - TXT SET M TABLE

Sets the address of a user-defined matrix table.

ENTRY

- **DE** is the first character in the table.
- **HL** is the table's address (in the central 32K of RAM).

EXIT

- If there are no existing tables then Carry is false, and **A** and **HL** are both corrupt
- otherwise Carry is true, **A** is the first character and **HL** is the table's address.
- In both cases **BC**, **DE** and the other flags are corrupt.

&BBAE - TXT GET M TABLE

Gets the address of a user-defined matrix table.

EXIT

See **TXT SET M TABLE** above for details of the values that can be returned.

&BBB1 - TXT GET CONTROLS

Gets the address of the control code table.

EXIT

- **HL** contains the address of the table.
- All other registers are preserved.

Notes

The table has 32 entries, and each entry has three bytes: byte 1 is the number of parameters needed by the control code, bytes 2 and 3 are the address of the routine, in the Lower ROM, to execute the control code.

&BBB4 - TXT STR SELECT

Selects a new **VDU** text stream.

ENTRY

- **A** contains the value of the stream to change to.

EXIT

- **A** contains the previously selected stream.
- **HL** and the flags are corrupt, and all other registers are preserved.

&BBB7 - TXT SWAP STREAMS

Swaps the states of two stream attribute tables.

ENTRY

- **B** contains a stream number.
- **C** contains the other stream number.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

Notes

The foreground pen and paper, the window size, the cursor position, the character write mode and graphic character mode are all exchanged between the two streams.

THE GRAPHICS VDU

The **Graphics VDU** allows individual pixels (dots) on the screen to be set or tested and lines to be drawn. The plotting takes place on an ideal screen that is always 640 points wide and 400 points high. This means that more than one point on the ideal screen will map onto a particular pixel on the real screen. The width of the ideal screen (640 points) is chosen to be the horizontal number of pixels on the screen in the highest resolution mode (mode 2). The height of the ideal screen (400 points) is chosen to be twice the vertical number of pixels on the screen in all modes. This ensures that the aspect ratio of the screen is approximately unity, i.e. a circle looks circular and not elliptical.

&BBBA - GRA INITIALISE

Initialises the graphics **VDU** to its default set-up (ie its set-up when the computer is switched on).

EXIT

- **AF, BC, DE** and **HL** are corrupt, and all other registers are preserved.

Notes

Sets the graphics indirections to their defaults, sets the graphic paper to text **pen 0** and the graphic pen to text **pen 1**, reset the graphics origin and move the graphics cursor to the bottom left of the screen, reset the graphics window and write mode to their defaults.

&BBBD - GRA RESET

Resets the graphics **VDU**.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

Notes

Resets the graphics indirections and the graphics write mode to their defaults.

&BBC0 - GRA MOVE ABSOLUTE

Moves the graphics cursor to an absolute screen position.

ENTRY

- **DE** contains the user X-coordinate.
- **HL** holds the user Y-coordinate.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BBC3 - GRA MOVE RELATIVE

Moves the graphics cursor to a point relative to its present screen position.

ENTRY

- **DE** contains the X - distance to move.
- **HL** holds the Y - distance.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are

preserved.

&BBC6 - GRA ASK CURSOR

Gets the graphics cursor's current position.

EXIT

- **DE** holds the user X-coordirlate.
- **HL** holds the user Y- coordinate.
- **AF** is corrupt, and all other registers are preserved.

&BBC9 - GRA SET ORIGIN

Sets the graphics user origin's screen position.

ENTRY

- **DE** contains the standard X-coordinate.
- **HL** holds the standard Y-coordinate.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BBCC - GRA GET ORIGIN

Gets the graphics user origin's screen position.

EXIT

- **DE** contains the standard X-coordinate.
- **HL** holds the standard Y-coordinate.
- All other registers are preserved.

&BBCF - GRA WIN WIDTH

Sets the left and right edges of the graphics window.

ENTRY

- **DE** contains the standard X-coordinate of one edge.
- **HL** holds the standard X-coordinate of the other side.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all the other registers are preserved.

Notes

The default window covers the entire screen and is restored to its default when the mode is changed; used in conjunction with GRA WIN HEIGHT.

&BBD2 - GRA WIN HEIGHT

Sets the top and bottom edges of the graphics window.

ENTRY

- **DE** contains the standard Y-coordinate of one side.
- **HL** holds the standard Y-coordinate of the other side.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

Notes

See **GRA WIN WIDTH** for further details.

&BBD5 - GRA GET W WIDTH

Gets the left and right edges of the graphics window.

EXIT

- **DE** contains the standard X-coordinate of the left edge.
- **HL** contains the standard Y-coordinate of the right edge.
- **AF** is corrupt, and all other registers are preserved.

&BBD8 - GRA GET W HEIGHT

Gets the top and bottom edges of the graphics window.

EXIT

- **DE** contains the standard Y-coordinate of the top edge.
- **HL** contains the standard Y-coordinate of the bottom edge
- **AF** is corrupt, and all other registers are preserved.

&BBDB - GRA CLEAR WINDOW

Clears the graphics window to the graphics paper colour and moves the cursor back to the user origin.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BBDE - GRA SET PEN

Sets the graphics PEN.

ENTRY

- **A** contains the required text PEN number.

EXIT

- **AF** is corrupt, and all other registers are preserved.

&BBE1 - GRA GET PEN

Gets the graphics PEN.

EXIT

- **A** contains the text PEN number.
- The flags are corrupt, and all other registers are preserved.

&BBE4 - GRA SET PAPER

Sets the graphics PAPER.

ENTRY

- **A** contains the required text PEN number.

EXIT

- **AF** corrupt, and all others are preserved.

&BBE7 - GRA GET PAPER

Gets the graphics PAPER.

EXIT

- **A** contains the text PEN number.
- The flags are corrupt, and all other registers are preserved.

&BBEA - GRA PLOT ABSOLUTE

Plots a point at an absolute user coordinate, using the GRA PLOT indirection.

ENTRY

- **DE** contains the user X-coordinate.
- **HL** holds the user Y-coordinate.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BBED - GRA PLOT RELATIVE

Plots a point at a position relative to the current graphics cursor, using the **GRA PLOT** indirection.

ENTRY

- **DE** contains the relative X - coordinate.
- **HL** contains the relative Y - coordinate.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BBF0 - GRA TEST ABSOLUTE

Moves to an absolute position, and tests the point there using the **GRA TEST** indirection.

ENTRY

- **DE** contains the user X-coordinate.
- **HL** holds the user Y-coordinate for the point you wish to test.

EXIT

- **A** contains the pen at the point.
- **BC, DE, HL** and flags are corrupt, and all other registers are preserved.

&BBF3 - GRA TEST RELATIVE

Moves to a position relative to the current position, and tests the point there using the **GRA TEST** indirection.

ENTRY

- **DE** contains the relative X-coordinate.
- **HL** contains the relative Y-coordinate.

EXIT

- **A** contains the pen at the point.
- **BC, DE, HL** and flags are corrupt, and all other registers are preserved.

&BBF6 - GRA LINE ABSOLUTE

Draws a line from the current graphics position to an absolute position, using **GRA LINE**.

ENTRY

- **DE** contains the user X-coordinate.
- **HL** holds the user Y-coordinate of the end point.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

Notes

The line will be plotted in the current graphics pen colour (may be masked to produce a dotted line on a 6128).

&BBF9 - GRA LINE RELATIVE

Draws a line from the current graphics position to a relative screen position, using **GRA LINE**.

ENTRY

- **DE** contains the relative X-coordinate.
- **HL** contains the relative Y-coordinate.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

Notes

See **GRA LINE ABSOLUTE** above for details of how the line is plotted.

&BBFC - GRA WR CHAR

Writes a character onto the screen at the current graphics position.

ENTRY

- **A** contains the character to be put onto the screen.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all the other registers are preserved.

Notes

As in BASIC, all characters including control codes are printed; the character is printed with its top left comer at the current graphics position; the graphics position is moved one character width to the right so that it is ready for another character to be printed.

THE SCREEN PACK

The **Screen Pack** is used by the **Text** and **Graphics VDUs** to access the hardware of the screen. It also controls the features of the screen that affect both the **Text VDU** and **Graphics VDU**, such as what mode the screen is in.

&BBFF - SCR INITIALISE

Initialises the Screen Pack to the default values used when the computer is first switched on.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

Notes

All screen indirections are restored to their default settings, as are inks and flashing speeds ; the mode is switched to **MODE 1** and the screen is cleared with **PEN 0** ; the screen address is moved to **&C000** and the screen offset is set to zero.

&BC02 - SCR RESET

Resets the Screen Pack's indirections, flashing speeds and inks to their default values.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BC05 - SCR SET OFFSET

Sets the screen offset to the specified values - this can cause the screen to scroll.

ENTRY

- **HL** contains the required offset, which should be even.

EXIT

- **AF** and **HL** are corrupt, and all other registers are preserved.

Notes

The screen offset is **reset to 0** whenever its mode is set, or it is cleared by **SCR CLEAR** [but not BASIC's CLS].

&BC08 - SCR SET BASE

Sets the location in memory of the screen - effectively can only be **&C000** or **&4000**.

ENTRY

- **A** contains the most significant byte of the screen address required.

EXIT

- **AF** and **HL** are corrupt, and all other registers are preserved.

Notes

The screen memory can only be set at 16K intervals [ie **&0000**, **&4000**, **&8000**, **&C000**] and when the computer is first switched on the 16K of screen memory is located at **&C000**].

&BC0B - SCR GET LOCATION

Gets the location of the screen memory and also the screen offset.

EXIT

- **A** holds the most significant byte of the screen address.
- **HL** holds the current offset.
- All other registers are preserved.

&BC0E - SCR SET MODE

Sets the screen mode.

ENTRY

- **A** contains the mode number - it has the same value and characteristics as in BASIC.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

Notes

The windows are set to cover the whole screen and the graphics origin is set to the bottom left corner of the screen; in addition, the current stream is set to zero, and the screen offset is zeroed.

&BC11 - SCR GET MODE

Gets the current screen mode.

EXIT

- If the mode is 0, then:
  - Carry is true.
  - Zero is false.
  - **A** contains 0.
- If the mode is 1, then:
  - Carry is false.
  - Zero is true.
  - **A** contains 1.
- If the mode is 2, then:
  - Carry is false.
  - Zero is false.
  - **A** contains 2.
- In all cases the other flags are corrupt and all the other registers are preserved.

&BC14 - SCR CLEAR

Clears the whole of the screen.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BC17 - SCR CHAR LIMITS

Gets the size of the whole screen in terms of the numbers of characters that can be displayed.

EXIT

- **B** contains the number of characters across the screen.
- **C** contains the number of characters down the screen.
- **AF** is corrupt, and all other registers are preserved.

&BC1A - SCR CHAR POSITION

Gets the memory address of the top left corner of a specified character position.

ENTRY

- **H** contains the character physical column.
- **L** contains the character physical row.

EXIT

- **HL** contains the memory address of the top left comer of the character.
- **B** holds the width in bytes of a character in the present mode.
- **AF** is corrupt, and all other registers are preserved.

&BC1D - SCR DOT POSITION

Gets the memory address of a pixel at a specified screen position.

ENTRY

- **DE** contains the base X - coordinate of the pixel.
- **HL** contains the base Y - coordinate.

EXIT

- **HL** contains the memory address of the pixel.
- **C** contains the bit mask for this pixel.
- **B** contains the number of pixels stored in a byte minus 1.
- **AF** and **DE** are corrupt, and all other registers are preserved.

&BC20 - SCR NEXT BYTE

Calculates the screen address of the byte to the right of the specified screen address (may be on the next line).

ENTRY

- **HL** contains the screen address.

EXIT

- **HL** holds the screen address of the byte to the right of the original screen address.
- **AF** is corrupt, all others are preserved.

&BC23 - SCR PREV BYTE

Calculates the screen address of the byte to the left of the specified screen address (this address may actually be on the previous line).

ENTRY

- **HL** contains the screen address.

EXIT

- **HL** holds the screen address of the byte to the left of the original address.
- **AF** is corrupt, all others are preserved.

&BC26 - SCR NEXT LINE

Calculates the screen address of the byte below the specified screen address.

ENTRY

- **HL** contains the screen address.

EXIT

- **HL** contains the screen address of the byte below the original screen address.
- **AF** is corrupt, and all the other registers are preserved.

&BC29 - SCR PREV LINE

Calculates the screen address of the byte above the specified screen address.

ENTRY

- **HL** contains the screen address.

EXIT

- **HL** holds the screen address of the byte above the original address.
- **AF** is corrupt, and all other registers are preserved.

&BC2C - SCR INK ENCODE

Converts a PEN to provide a mask which, if applied to a screen byte, will convert all of the pixels in the byte to the appropriate PEN.

ENTRY

- **A** contains a PEN number.

EXIT

- **A** contains the encoded value of the PEN.
- Flags are corrupt, and all other registers are preserved.

Notes

The mask returned is different in each of the screen modes.

&BC2F - SCR INK DECODE

Converts a PEN mask into the PEN number (see **SCR INK ENCODE** for the reverse process).

ENTRY

- **A** contains the encoded value of the PEN.

EXIT

- **A** contains the PEN number.
- Flags are corrupt, and all other registers are preserved.

&BC32 - SCR SET INK

Sets the colours of a PEN - if the two values supplied are different then the colours will alternate (flash).

ENTRY

- **A** contains the PEN number.
- **B** contains the first colour.
- **C** holds the second colour.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BC35 - SCR GET INK

Gets the colours of a PEN.

ENTRY

- **A** contains the PEN number.

EXIT

- **B** contains the first colour.
- **C** holds the second colour.
- **AF**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BC38 - SCR SET BORDER

Sets the colours of the border - again if two different values are supplied, the border will flash.

ENTRY

- **B** contains the first colour.
- **C** contains the second colour.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BC3B - SCR GET BORDER

Gets the colours of the border.

EXIT

- **B** contains the first colour.
- **C** holds the second colour.
- **AF**, **DE** and **HL** are corrupt, and all other registers are

preserved.

&BC3E - SCR SET FLASHING

Sets the speed with which the border's and PENs colours flash.

ENTRY

- **H** holds the time that the first colour is displayed.
- **L** holds the time the second colour is displayed for.

EXIT

- **AF** and **HL** are corrupt, and all other registers are preserved.

Notes

The length of time that each colour is shown is measured in 1/50ths of a second, and a value of 0 is taken to mean 256 * 1/50 seconds - the default value is 10 * 1/50 seconds.

&BC41 - SCR GET FLASHING

Gets the periods with which the colours of the border and PENs flash.

EXIT

- **H** holds the duration of the first colour.
- **L** holds the duration of the second colour.
- **AF** is corrupt, and all other registers are preserved.

Notes

See **SCR SET FLASHING** for the units of time used.

&BC44 - SCR FILL BOX

Fills an area of the screen with an ink - this only works for 'character-sized' blocks of screen.

ENTRY

- **A** contains the mask for the ink that is to be used.
- **H** contains the left hand colurnn of the area to fill.
- **D** contains the right hand column.
- **L** holds the top line.
- **E** holds the bottom line of the area (using physical coordinates).

EXIT

- **AF, BC, DE** and **HL** are corrupt, and all other registers are preserved.

&BC17 - SCR FLOOD BOX

Fills an area of the screen with an ink - this only works for 'byte-sized' blocks of screen.

ENTRY

- **C** contains the encoded PEN that is to be used.
- **HL** contains the screen address of the top left hand corner of the area to fill.
- **D** contains the width of the area to be filled in bytes.
- **E** contains the height of the area to be filled in screen lines.

EXIT

- **AF, BC, DE** and **HL** are corrupt, and all other registers are preserved.

Notes

Notes

The whole of the area to be filled must lie on the screen otherwise unpredictable results may occur.

&BC4A - SCR CHAR INVERT

Inverts a character's colours; all pixels in one PEN's colour are printed in another PEN's colour, and vice versa.

ENTRY

- **B** contains one encoded PEN.
- **C** contains the other encoded PEN.
- **H** contains the physical column number.
- **L** contains the physical line number of the character that is to be inverted.

EXIT

- **AF, BC, DE** and **HL** are corrupt, and all the other registers are preserved.

&BC4D - SCR HW ROLL

Scrolls the entire screen up or down by eight pixel rows [ie one character line].

ENTRY

- **B** holds the direction that the screen will roll.
- **A** holds the encoded PAPER which the new line will appear in.

EXIT

- **AF, BC, DE** and **HL** are corrupt, and all other registers are preserved.

Notes

This alters the screen offset; to roll down, B must hold zero, and

to roll upwards B must be non-zero.

&BC50 - SCR SW ROLL

Scrolls part of the screen up or down by eight pixel lines - only for 'character-sized' blocks of the screen.

ENTRY

- **B** holds the direction to roll the screen.
- **A** holds the encoded PAPER which the new line will appear in.
- **H** holds the left column of the area to scroll.
- **D** holds the right column.
- **L** holds the top line.
- **E** holds the bottom line.

EXIT

- **AF, BC, DE** and **HL** are corrupt, and all other registers are preserved.

Notes

The area of the screen is moved by copying it; to roll down, B must hold zero, and to roll upwards B must be non-zero; this routine uses physical roordinates.

&BC53 - SCR UNPACK

Changes a character matrix from its eight byte standard form into a set of pixel masks which are suitable for the current mode - four *8 bytes are needed in mode 0, two *8 bytes in mode 1, and 8 bytes in mode 2.

ENTRY

- **HL** contains the address of the matrix.
- **DE** contains the address where the masks are to be stored.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BC56 - SCR REPACK

Changes a set of pixel masks (for the current mode) into a standard eight byte character matrix.

ENTRY

- **A** contains the encoded foreground PEN to be matched against (ie the PEN that is to be regarded as being set in the character).
- **H** holds the physical column of the character to be 'repacked'.
- **L** holds the physical line of the character.
- **DE** contains the address of the area where the character matrix will be built.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all the others are preserved.

&BC59 - SCR ACCESS

Sets the screen write mode for graphics.

ENTRY

- **A** contains the write mode (**0=Fill**, **1=XOR**, **2=AND**, **3=OR**).

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, amd all other registers are preserved.

Notes

The fill mode means that the ink that plotting was requested in is

the ink that appears on the screen; in **XOR** mode, the specified ink is XORed with ink that is at that point on the screen already before plotting; a similar situation occurs with the **AND** and **OR** modes.

&BC5C - SCR PIXELS

Puts a pixel or pixels on the screen regardless of the write mode specified by **SCR ACCESS** above.

ENTRY

- **B** contains the mask of the PEN to be drawn with.
- **C** contains the pixel mask.
- **HL** holds the screen address of the pixel.

EXIT

- **AF** is corrupt, amd all others are preserved.

&BC5F - SCR HORIZONTAL

Draws a honzontal line on the screen using the current graphics write mode.

ENTRY

- **A** contains the encoded PEN to be drawn with.
- **DE** contains the base X - coordinate of the start of the line.
- **BC** contains the end base X - coordinate.
- **HL** contains the base Y - coordinate.

EXIT

- **AF**, **BC**, **DE** and **HL** are conupt, and all other registers are preserved.

Notes

The start X - coordinate must be less than the end X - coordinate.

&BC62 - SCR VERTICAL

Draws a vertical line on the screen using the current graphics write mode.

ENTRY

- **A** contains the encoded PEN to be drawn with.
- **DE** contains the base X - coordinate of the line.
- **HL** holds the start base Y - coordinate.
- **BC** contains the end base Y - coordinate. The start coordinate must be less than the end coordinate.

EXIT

- **AF, BC, DE** and **HL** are corrupt, and all the other registers are preserved.

THE CASSETTE/AMSDOS MANAGER

Some of these routines are only applicable to the cassette manager. Where a disc version exists it is indicated by an asterisk [*] next to the command name. These disc version jumpblocks are automatically installed by the Disc Operating System on switch on.

&BC65 - CAS INITIALISE

Initialises the cassette manager.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all the other registers are preserved.

Notes

Both read and write streams are closed. Tape messages are switched on and the default speed is reselected

&BC68 - CAS SET SPEED

Sets the speed at which the cassette manager saves programs.

ENTRY

- **HL** holds the length of 'half a zero' bit.
- **A** contains the amount of precompensation.

EXIT

- **AF** and **HL** are corrupt.

Notes

The value in **HL** is the length of time that half a zero bit is written as; a “one” bit is twice the length of a “zero” bit; the

default values (ie SPEED WRITE 0) are 333 microseconds (HL) and 25 microseconds (A) for SPEED WRITE 1, the values are given as 107 microseconds and 50 microseconds respectively.

&BC6B - CAS NOISY

Enables or disables the display of cassette handling messages.

ENTRY

- To enable the messages then **A** must be 0, otherwise the messages are disabled.

EXIT

- **AF** is corrupt, and all other registers are preserved.

&BC6E - CAS START MOTOR

Switches on the tape motor.

EXIT

- If the motor operates properly then Carry is true.
- If ESC was pressed then Carry is false.
- In either case:
    - **A** contains the motor's previous state.
    - Flags are corrupt, and all other registers are preserved.

&BC71 - CAS STOP MOTOR

Switches off the tape motor.

EXIT

- If the motor turns off then Carry is true.
- If ESC was pressed then Carry is false.
- In both cases:
    - **A** holds tbe motor's previous state.
    - Flags are corrupt, all others are preserved.

&BC74 - CAS RESTORE MOTOR

Resets the tape motor to its previous state.

ENTRY

- **A** contains the previous state of the motor [eg from **CAS START MOTOR** or **CAS STOP MOTOR**]

EXIT

- If the motor operates properly then Carry is true.
- If **ESC** was pressed then Carry is false.
- In all cases, **A** and the other flags are corrupt and all others are preserved.

&BC77 - *CAS IN OPEN

Opens an input buffer and reads the first block of the file.

ENTRY

- **B** contains the length of the filename.
- **HL** contains the filename's address.
- **DE** contains the address of the 2K buffer to use for reading the file.

EXIT

- If the file was opened successfully, then:
  - Carry is true.
  - Zero is false.
  - **HL** holds the address of a buffer containing the file header data.
  - **DE** holds the address of the destination for the file.
  - **BC** holds the file length.
  - **A** holds the file type.
- If the read stream is already open then:
  - Carry and Zero are false.
  - **A** contains an error number **[664/6128 only]**.
  - **BC**, **DE** and **HL** are corrupt.
- If **ESC** was pressed by the user, then:
  - Carry is false.
  - Zero is true.
  - **A** holds an error number **[664/6128 only]**.
  - **BC**, **DE** and **HL** are corrupt.
- In all cases, **IX** and the other flags are corrupt, and the others are preserved.

Notes

A filename of zero length means “read the next file on the tape”. The stream remains open until it is closed by either **CAS IN CLOSE** or **CAS IN ABANDON**.

DISC

Similar to tape except that if there is no header on the file, then a fake header is put into memory by this routine.

&BC7A - *CAS IN CLOSE

Closes an input file.

EXIT

- If the file was closed successfully, then Carry is true and **A** is corrupt.
- If the read stream was not open, then Carry is false, and A holds an error code **(664/6128 only)**.
- In both cases, **BC**, **DE**, **HL** and the other flags are all corrupt.

DISC

All the above applies, but also if the file failed to close for any other reason, then Carry is false, Zero is true and **A** contains an error number. In all cases the drive motor is turned off immediately.

&BC7D - *CAS IN ABANDON

Abandons an input file.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

DISC

All the above applies for the disc routine.

&BC80 - *CAS IN CHAR

Reads in a single byte from a file.

EXIT

- If a byte was read, then Carry is true, Zero is false, and **A** contains the byte read from the file.
- If the end of file was reached, then Carry and Zero are false, **A** contains an error number **[664/6128 only]** or is corrupt [for the 464].
- If ESC was pressed, then Carry is false, Zero is true, and **A** holds an error number **[664/6128 only]** or is corrupt [for the 464].
- In all cases, **IX** and the other flags are corrupt, and all other registers are preserved.

DISC

All the above applies for the disc routine.

&BC83 - *CAS IN DIRECT

Reads an entire file directly into memory.

ENTRY

- **HL** contains the address where the file is to be placed in RAM.

EXIT

- If the operation was successful, then:
    - Carry is true.
    - Zero is false.
    - **HL** contains the entry address.
    - **A** is corrupt.
- If it was not open, then:
    - Carry and Zero are both false.
    - **HL** is corrupt.

    - **A** holds an error code **(664/6128)** or is corrupt (464).
- If ESC was pressed:
    - Carry is false.
    - Zero is true.
    - **HL** is corrupt.
    - **A** holds an error code **(664/6128 only)**.
- In all cases, **BC**, **DE** and **IX** and the other flags are corrupt, and the others are preserved.

Notes

This routine cannot be used once **CAS IN CHAR** has been used.

DISC

All the above applies to the disc routine.

&BC86 - *CAS RETURN

Puts the last byte read back into the input buffer so that it can be read again at a later time.

EXIT

- All registers are preserved.

Notes

The routine can only return the last byte read and at least one byte must have been read.

Disc: All the above applies to the disc routine.

&BC89 - *CAS TEST EOF

Tests whether the end of file has been encountered.

EXIT

- If the end of file has been reached, then Carry and Zero are false, and **A** is corrupt.
- If the end of file has not been encountered, then Carry is true, Zero is false, and **A** is corrupt.
- If ESC was pressed then Carry is false, Zero is true and **A** contains an error number **(664/6128 only)**.
- In all cases, **IX** and the other flags are corrupt, and all other registers are preserved.

DISC

All the above applies to the disc routine.

&BC8C - *CAS OUT OPEN

Opens an output file

ENTRY

- **B** contains the length of the filename.
- **HL** contains the address of the filename.
- **DE** holds the address of the 2K buffer to be used.

EXIT

- If the file was opened correctly, then:
  - Carry is true.
  - Zero is false.
  - **HL** holds the address of the buffer containing the file header data that will be written to each block.
  - **A** is corrupt.
- If the write stream is already open, then:
  - Carry and Zero are false.

    - **A** holds an error number **[664/6128]**.
    - **HL** is corrupt.
- If ESC was pressed then:
    - Carry is false.
    - Zero is true.
    - **A** holds an error number **664/6128]**.
    - **HL** is corrupt.
- In all cases, **BC**, **DE**, **IX** and the other flags are corrupt, and the others are preserved.

Notes

The buffer is used to store the contents of a file block before it is actually written to tape.

DISC

The same as for tape except that the filename must be present in its usual AMSDOS format.

&BC8F - *CAS OUT CLOSE

Closes an output file.

EXIT

- If the file was closed successfully, then Carry is true, Zero is false, and **A** is corrupt.
- If the write stream was not open, then Carry and Zero are false and **A** holds an error code **[664/6128 only]**.
- If ESC was pressed then Carry is false, Zero is true, and **A** contains an error code **[664/6128 only]**.
- In all cases, **BC**, **DE**, **HL**, **IX** and the other flags are all corrupt.
-

Notes

The last block of a file is written only when this routine is called; if writing the file is to be abandoned, then

CAS OUT ABANDON should be used instead.

DISC

All the above applies to the disc routine.

&BC92 - *CAS OUT ABANDON

Abandons an output file.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all others are preserved.

Notes

When using this routine, the current last block of the file is not written to the tape.

DISC

Similar to the tape routine; if more than **16K** of a file has been written to the disc, then the first 16K of the file will exist on the disc with a file extension of .**$$$** because each **16K** section of the file requires a separate directory entry.

&BC95 - *CAS OUT CHAR

Writes a single byte to a file.

ENTRY

- **A** contains the byte to be written to the file output buffer.

EXIT

- If a byte was written to the buffer, then Carry is true, Zero is false, and **A** is corrupt.
- If the file was not open, then Carry and Zero are false, and **A** contains an error number **[664/6128 only]** or is corrupt [on the 464].

- If ESC was pressed, then Carry is false, Zero is true, and **A** contains an error number **(664/6128 only)** or it is corrupt (on the 464).
- In all cases, **IX** and the other flags are corrupt, and all other registers are preserved.

Notes

If the **2K** buffer is full of data then it is written to the tape before the new character is placed in the buffer; it is important to call **CAS OUT CLOSE** when all the data has been sent to the file so that the last block is written to the tape.

Disc: All the above applies to the disc routine.

&BC98 - *CAS OUT DIRECT

Writes an entire file directly to tape.

ENTRY

- **HL** contains the address of the data which is to be written to tape.
- **DE** contains the length of this data.
- **BC** contains the execution address.
- **A** contains the file type.

EXIT

- If the operation was successful, then Carry is true, Zero is false, and **A** is corrupt.
- If the file was not open, Carry and Zero are false, **A** holds an error number **(664/6128)** or is corrupt (464).
- If ESC was pressed, then Carry is false, Zero is true, and **A** holds an error code **(664/6128 only)**.
- In all cases **BC, DE, HL, IX** and the other flags are corrupt, and the others are preserved.

Notes

This routine cannot be used once **CAS OUT CHAR** has been used.

DISC

All the above applies to the disc routine.

&BC9B - *CAS CATALOG

Creates a catalogue of all the files on the tape.

ENTRY

- **DE** contains the address of the 2K buffer to be used to store the information.

EXIT

- If the operation was successful, then Carry is true, Zero is false, and **A** is corrupt.
- If the read stream is already being used, then Carry and Zero are false, and **A** holds an error code **[664/6128]** or is corrupt [for the 464].
- In all cases, **BC, DE, HL, IX** and the other flags are corrupt and all other registers are preserved.

Notes

This routine is only left when the ESC key is pressed [cassette only] and is identical to BASIC's CAT command.

DISC

All tbe above applies, except that a sorted list of files is displayed; system files are not listed by this routine.

&BC9E - CAS WRITE

Writes data to the tape in one long file (ie not in 2K blocks).

ENTRY

- **HL** contains the address of the data to be written to tape.
- **DE** contains the length of the data to be written.
- **A** contains the sync character.

EXIT

- If the operation was successful, then Carry is true and **A** is corrupt.
- If an error occurred then Carry is false and **A** contains an error code.
- In both cases, **BC**, **DE**, **HL** and **IX** are corrupt, and all other registers are preserved.

Notes

For header records the sync character is **&2C**, and for data it is **&16**; this routine starts and stops the cassette motor and also turns off interrupts whilst writing data.

&BCA1 - CAS READ

Reads data from the tape in one long file (ie as originally written by **CAS WRITE** only).

ENTRY

- **HL** holds the address to place the file.
- **DE** holds the length of the data.
- **A** holds the expected sync character.

EXIT

- If the operation was successful, then Carry is true and **A** is corrupt.

- If an error occurred then Carry is false and **A** contains an error code.
- In both cases, **BC**, **DE**, **HL** and **IX** are corrupt, and all other registers are preserved.

Notes

For header records the sync character is **&2C**, and for data it is **&16**; this routine starts and stops the cassette motor and turns off interrupts whilst reading data.

&BCA4 - CAS CHECK

Compares the contents of memory with a file record [ie header or data] on tape.

ENTRY

- **HL** contains the address of the data to check.
- **DE** contains the length of the data.
- **A** holds the sync character that was used when the file was originally written to the tape.

EXIT

- If the two are identical, then Carry is true and **A** is corrupt.
- If an error occurred then Carry is false and **A** holds an error code.
- In all cases, **BC**, **DE**, **HL**, **IX** and other flags are corrupt, and all other registers are preserved.

Notes

For header records the sync character is **&2C**, and for data it is **&16**; this routine starts and stops the cassette motor and turns off interrupts whilst reading data; does not have to read the whole of a record, but must start at the beginning.

THE SOUND MANAGER

&BCA7 - SOUND RESET

Resets the sound manager by clearing the sound queues and abandoning any current sounds.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all others are preserved.

&BCAA - SOUND

Adds a sound to the sound queue of a channel.

ENTRY

- **HL** contains the address of a series of bytes which define the sound and are stored in the central **32K** of **RAM**.

EXIT

- If the sound was successfully added to the queue, then Carry is true and **HL** is corrupt.
- If one of the sound queues was full, then Carry is false and **HL** is preserved.
- In either case, **A**, **BC**, **DE**, **IX** and the other flags are corrupt, and all others are preserved.

Notes

The bytes required to define the sound are as follows:

- **byte 0** - channel status byte
- **byte 1** - volume envelope to use
- **byte 2** - tone envelope to use
- **bytes 3&4** - tone period
- **byte 5** - noise period
- **byte 6** - start volume
- **bytes 7&8** - duration of the sound, or envelope repeat count

&BCAD - SOUND CHECK

Gets the status of a sound channel.

ENTRY

- **A** contains the channel to test - for channel A, bit 0 set; for channel B, bit 1 set; for channel **C**, bit 2 set.

EXIT

- **A** contains the channel status.
- **BC, DE, HL** and flags are corrupt, and all other registers are preserved.

Notes

The channel status returned is bit significant, as follows:

- **bits 0 to 2** - the number of free spaces in the sound queue
- **bit 3** - trying to rendezvous with channel A
- **bit 4** - trying to rendezvous with channel B
- **bit 5** - trying to rendezvous with channel C
- **bit 6** - holding the channel
- **bit 7** - producing a sound

&BCB0 - SOUND ARM EVENT

Sets up an event which will be activated when a space occurs in a sound queue.

ENTRY

- **A** contains the channel to set the event up for [see **SOUND CHECK** for the bit values this can take].
- **HL** holds the address of the event block.

EXIT

- **AF, BC, DE** and **HL** are corrupt, and all other registers are preserved.

Notes

The event block must be initialised by **KL INIT EVENT** and is disarmed when the event itself is run.

&BCB3 - SOUND RELEASE

Allows the playing of sounds on specific channels that had been stopped by **SOUND HOLD**.

ENTRY

- **A** contains the sound channels to be released (see **SOUND CHECK** for the bit values this can take).

EXIT

- **AF**, **BC**, **DE**, **HL** and **IX** are corrupt, and all other registers are preserved.

&BCB6 - SOUND HOLD

Immediately stops all sound output (on all channels).

EXIT

- If a sound was being made, then Carry is true.
- If no sound was being made, then Carry is false.
- In all cases, **A**, **BC**, **HL** and other flags are corrupt, and all other registers are preserved.

Notes

When the sounds are restarted, they will begin from exactly the same place that they were stopped.

&BCB9 - SOUND CONTINUE

Restarts all sound output (on all channels).

EXIT

- **AF**, **BC**, **DE** and **IX** are corrupt, and all other registers are preserved.

&BCBC - SOUND AMPL ENVELOPE

Sets up a volume envelope.

ENTRY

- **A** holds an envelope number (from 1 to 15).
- **HL** holds the address of a block of data for the envelope.

EXIT

- If it was set up properly:
  - Carry is true.
  - **HL** holds the data block address + 16.
  - **A** and **BC** are corrupt.
- If the envelope number is invalid, then:
  - Carry is false
  - **A**, **B** and **HL** are preserved.
- In either case, **DE** and the other flags are corrupt, and all other registers are preserved.

Notes

All the rules of enevelopes in BASIC also apply; the block of the data for the envelope is set up as follows:

- **byte 0** - number of sections in the envelope
- **bytes 1 to 3** - first section of the envelope
- **bytes 4 to 6** - second section of the envelope
- **bytes 7 to 9** - third section of the envelope
- **bytes 10 to 12** - fourth section of the envelope
- **bytes 13 to 15** - fifth section of the envelope

Each section of the envelope has three bytes set out as follows:

- **byte 0** - step count (with bit 7 set)
- **byte 1** - step size
- **byte 2** - pause time or if it is a hardware envelope, then each section takes the following form:
    - **byte 0** - envelope shape (with bit 7 not set)
    - **bytes 1 and 2** - envelope period

See also **SOUND TONE ENVELOPE** below.

&BCBF - SOUND TONE ENVELOPE

Sets up a tone envelope.

ENTRY

- **A** holds an envelope number (from 1 to 15).
- **HL** holds the address of a block of data for the envelope.

EXIT

- If it was set up properly:
    - Carry is true.
    - **HL** holds the data block address + 16.
    - **A** and **BC** are corrupt.
- If the envelope number is invalid, then Carry is false, and A, B and **HL** are preserved.
- In either case, **DE** and the other flags are corrupt, and all other registers are preserved.

Notes

All the rules of envelopes in BASIC also apply; the block of the data for the envelope is set up as follows:

- **byte 0** - number of sections in the envelope
- **bytes 1 to 3** - first section of the envelope
- **bytes 4 to 6** - second section of the envelope
- **bytes 7 to 9** - third section of the envelope
- **bytes 10 to 12** - fourth section of the envelope
- **bytes 13 to 15** - fifth section of the envelope

Each section of the envelope has three bytes set out as follows:

- **byte 0** - step count
- **byte 1** - step size
- **byte 2** - pause time

See also **SOUND AMPL ENVELOPE** above.

&BCC2 - SOUND A ADDRESS

Gets the address of the data block associated with a volume envelope.

ENTRY

- **A** contains an envelope number [from 1 to 15].

EXIT

- If it was found, then:
  - Carry is true
  - **HL** holds the data block's address.
  - **BC** holds its length.
- If the envelope number is invalid, then Carry is false, **HL** is corrupt and **BC** is preserved.
- In both cases, **A** and the other flags are corrupt, and all other registers are preserved.

&BCC5 - SOUND T ADDRESS

Gets the address of the data block associated with a tone envelope.

ENTRY

- **A** contains an envelope number [from 1 to 15].

EXIT

- If it was found, then:
  - Carry is true.
  - **HL** holds the data block's address.
  - **BC** holds its length.
- If the envelope number is invalid, then Carry is false, **HL** is corrupt and **BC** is preserved.
- In both cases, **A** and the other flags are corrupt, and all others are preserved.

THE KERNEL

&BCC8 - KL CHOKE OFF

Clears all event queues and timer lists, with the exception of keyboard scanning and sound routines.

EXIT

- **B** contains the foreground ROM select address (if any)
- **DE** contains the ROM entry address.
- **C** holds the ROM select address for a RAM foreground program.
- **AF** and **HL** are corrupt, and all other registers are preserved.

&BCCB - KL ROM WALK

Finds and initialises all background ROMs.

ENTRY

- **DE** holds the address of the first usable byte of memory.
- **HL** holds the address of the last usable byte.

EXIT

- **DE** holds the address of the new first usable byte of memory.
- **HL** holds the address of the new last usable byte.
- **AF** and **BC** are corrupt, and all other registers are preserved.

Notes

This routine looks at the ROM select addresses from 0 to 15 (1 to 7 for the 464) and calls the initialisation routine of any ROMs present; these routines may reserve memory by adjusting **DE** and **HL** before returning control to **KL ROM WALK**, and the **ROM** is then added to the list of command handling routines.

&BCCE - KL INIT BACK

Finds and initialises a specific background ROM.

ENTRY

- **C** contains the ROM select address of the ROM.
- **DE** holds the address of the first usable byte of memory.
- **HL** holds the address of the last usable byte of memory.

EXIT

- **DE** holds the address of the new first usable byte of memory.
- **HL** holds the address of the new last usable byte.
- **AF** and **B** are corrupt, and all other registers are preserved.

Notes

The ROM select address must be in the range of 0 to 15 (or 1 to 7 for the 464) although address 7 is tor the AMSDOS/CPM ROM if present. The ROM's initialisation routine is then called and some memory may be reserved for the ROM by adjusting the values of **DE** and **HL** before returning control to KL IN1T BACK.

&BCD1 - KL LOG EXT

Logs on a new RSX to the firmware.

ENTRY

- **BC** contains the address of the RSX's command table.
- **HL** contains the address of four bytes exclusively for use by the firmware.

EXIT

- **DE** is corrupt, and all other registers are preserved.

&BCD4 - KL FIND COMMAND

Searches an RSX, background ROM or foreground ROM, to find a command in its table.

ENTRY

- **HL** contains the address of the command name (in RAM only) which is being searched for.

EXIT

- If the name was found in a RSX or background ROM then:
  - Carry is true
  - **C** contains the ROM select address.
  - **HL** contains the address of the routine
- If the command was not found, then:
  - Carry is false.
  - **C** and **HL** are corrupt.
- In either case, **A**, **B** and **DE** are corrupt, and all other registers are preserved.

Notes

The command names should be in upper case and the last character should have &80 added to it; the sequence of searching is RSXs, then ROMs with lower numbers before ROMs with higher numbers.

&BCD7 - KL NEW FRAME FLY

Sets up a frame flyback event block which will be acted on whenever a frame flyback occurs.

ENTRY

- **HL** contains the address of the event block in the central 32K of RAM.
- **B** contains the event class.
- **C** contains the ROM select address (if any).
- **DE** contains the address if the event routine.

EXIT

- **AF**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BCDA - KL ADD FRAME FLY

Adds an existing but deleted frame flyback event block to the list of routines run when a frame flyback occurs.

ENTRY

- **HL** contains the address of the event block (in the central 32K of RAM).

EXIT

- **AF**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BCDD - KL DEL FRAME FLY

Removes a frame flyback event block from the list of routines which are mn when a frame flyback occurs.

ENTRY

- **HL** contains the address of the event block.

EXIT

- **AF**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BCE0 - KL NEW FAST TICKER

Sets up a fast ticker event block which will be run whenever the 1/300th second ticker interrupt occurs.

ENTRY

- **HL** contains the address of the event block [in the central 32K of RAM].
- **B** contains the event class.
- **C** contains the ROM select address [if any].
- **DE** contains the address of the event routine.

EXIT

- **AF**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BCE3 - KL ADD FAST TICKER

Adds an existing but deleted fast ticker event block to the list of routines which are run when the 1/300th sec ticker interrupt occurs.

ENTRY

- **HL** contains the address of the event block.

EXIT

- **AF**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BCE6 - KL DEL FAST TICKER

Removes a fast ticker event block from the list of routines run when the 1/300th sec ticker interrupt occurs.

ENTRY

- **HL** contains the address of the event block.

EXIT

- **AF**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BCE9 - KL ADD TICKER

Sets up a ticker event block which will be run whenever a 1/50th second ticker interrupt occurs.

ENTRY

- **HL** contains the address of the event block (in the central 32K of RAM).
- **DE** contains the initial value for the counter.
- **BC** holds the value that the counter will be given whenever it reaches zero.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all the other registers are preserved.

Notes

Every 1/50th of a second all the tick blocks are looked at and their counter is decreased by 1; when the counter reaches zero, the event is 'kicked' and the counter is loaded with the value in BC; any tick block with a counter of 0 is ignored, and therefore if the value in **BC** is 0, the event will be kicked only once and ignored after that.

&BCEC - KL DEL TICKER

Removes a ticker event block from the list of routines that are run when a 1/50th sec ticker interrupt occurs.

ENTRY

- **HL** contains the address of the event block.

EXIT

- If the event block was found, then Carry is true, and **DE** holds the value remaining of the counter.
- If the event block was not found, then Carry is false, and DE is corrupt.
- In both cases, **A, HL** and the other flags are corrupt, and all other registers are preserved.

&BCEF - KL INIT EVENT

Initialises an event block.

ENTRY

- **HL** contains the address of the event block [in the central 32K of RAM].
- **B** contains the class of event.
- **C** contains the ROM select address.
- **DE** holds the address of the event routine.

EXIT

- **HL** holds the address of the event block+7, and all other registers are preserved.

Notes

The event class is derived as follows:

- **bit 0** - indicates a near address
- **bits 1 to 4** - hold the synchronous event priority
- **bit 5** - always zero
- **bit 6** - if it is set, then it is an express event
- **bit 7** - if it is set, then it is an asynchronous event.
- Asynchronous events do not have priorities.
- If it is an express asynchronous event, then its event routine is called from the interrupt path.
- If it is a normal asynchronous event, then its event routine is called just before returning from the interrupt.
- If it is an express synchronous event, then it has a higher priority than normal synchronous events, and it may not be disabled through use of **KL EVENT DISABLE**.
- If the near address bit is set, then the routine is located in the central 32K of RAM and is called directly, so saving time; no event may have a priority of zero.

&BCF2 - KL EVENT

Kicks an event block.

ENTRY

- **HL** contains the address of the event block.

EXIT

- **AF, BC, DE** and **HL** are corrupt, and all other registers are preserved.

&BCF5 - KL SYNC RESET

Clears the synchronous event queue.

EXIT

- **AF** and **HL** are corrupt, and all other registers are preserved.

Notes

When using this routine, all events that are waiting to be dealt with are simply discarded.

&BCF8 - KL DEL SYNCHRONOUS

Removes a synchronous event from the event queue.

ENTRY

- **HL** contains the address of the event block.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BCFB - KL NEXT SYNC

Finds out if there is a synchronous event with a higher priority.

EXIT

- If there is an event to be processed, then:
  - Carry is true.
  - **HL** contains the address of the event block.
  - **A** contains the priority of the previous event.
- If there is no event to be processed, then:
  - Carry is false.
  - **A** and **HL** are corrupt.
- In either case, **DE** is corrupt, and all other registers are preserved.

&BCFE - KL DO SYNC

Runs a synchronous event routine.

ENTRY

- **HL** contains the address of the event block.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

Notes

See KL DONE SYNC below.

&BD01 - KL DONE SYNC

Finishes running a synchronous event routine.

ENTRY

- **A** contains the priority of the previous event.
- **HL** contains the address of the event block.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

Notes

When an event that is waiting to be processed has been found by **KL NEXT SYNC**, the event routine should be run by **KL DO SYNC** ; after this **KL DONE SYNC** should be called so that the event counter can be decreased - if the counter is greater than zero then the event is placed back on the synchronous event queue.

&BD04 - KL EVENT DISABLE

Disables normal synchronous events.

EXIT

- **HL** is corrupt, and all other registers are preserved.

&BD07 - KL EVENT ENABLE

Enables normal synchronous events.

EXIT

- **HL** is corrupt, and all other registers are preserved.

&BD0A - KL DISARM EVENT

Disarms a specific event and stops it from occurring.

ENTRY

- **HL** contains the address of the event block.

EXIT

- **AF** is corrupt, and all other registers are preserved.

Notes

This routine should be used to disarm only asynchronous events ; see also **KL DEL SYNCHRONOUS**.

&BD0D - KL TIME PLEASE

Returns the time that has elapsed since the computer was switched on or reset (in 1/300ths of a second).

EXIT

- **DEHL** contains the four byte count of the time elapsed.
- All other registers are preserved.

Notes

D holds the most signifilcant byte of the time elapsed, and **L** holds the least significant; the four byte count overflows after approximately 166 days have elapsed.

&BD10 - KL TIME SET

Sets the elapsed time (in 1/300ths of a second).

ENTRY

- **DEHL** contains the four byte count of the time to set.

EXIT

- **AF** is corrupt, and all other registers are preserved.

THE MACHINE PACK

&BD13 - MC BOOT PROGRAM

Loads a program into **RAM** and then executes it.

ENTRY

- **HL** contains the address of the routine which is used to load the program.

EXIT

Control is handed over to the program and so the routine is not returned from.

Notes

- All events, sounds and interrupts are turned off.
- The firmware indirections are returned to their default settings.
- The stack is reset.

The routine to run the program should be in the central block of memory, and should obey the following exit conditions:

- If the program was loaded successfully:
  - Carry is true
  - **HL** contains the prograrn entry point.
- If the program failed to load:
  - Carry is false
  - **HL** is corrupt.

In either case, **A, BC, DE, IX, IY** and the other flags are all corrupt. Should the program fail to load, control is returned to the previous foreground program.

&BD16 - MC START PROGRAM

Runs a foreground program.

ENTRY

- **HL** contains the entry point for the program.
- **C** contains the ROM selection number.

EXIT

Control is handed over to the program and so the routine is not returned from.

&BD19 - MC WAIT FLYBACK

Waits until a frame flyback occurs.

EXIT

- All registers are preserved.

Notes

When the frame flyback occurs the screen is not being written to and so the screen can be manipulated during this period without any flickering or ghosting on the screen.

&BD1C - MC SET MODE

Sets the screen mode.

ENTRY

- **A CONTAINS THE REQUIRED MODE.**

EXIT

- **AF** is corrupt, and all other registers are preserved.

Notes

Although this routine changes the screen mode it does not inform the routines which write to the screen that the mode has been changed; therefore these routines will write to the screen as if the mode had not been changed; however as the hardware is now interpreting these signals differently, unusual effects may occur.

&BD1F - MC SCREEN OFFSET

Sets the screen offset.

ENTRY

- **A** contains the screen base.
- **HL** contains the screen offset.

EXIT

- **AF** is corrupt, and all other registers are preserved.

Notes

As with **MC SET MODE**, this routine changes the hardware setting without telling the routines that write to the screen ; therefore these routines may cause unpredictable effects if called ; the default screen base is **&C0**.

&BD22 - MC CLEAR INKS

Sets all the PENs and the border to one colour, so making it seem as if the screen has been cleared.

ENTRY

- **DE** contains the address of the ink vector.

EXIT

- **AF** is corrupt, and all other registers are preserved.

Notes

The ink vector takes the following form:

- **byte 0** - holds the colour for the border
- **byte 1** - holds the colour for all of the PENs.

The values for the colours are all given as hardware values.

&BD25 - MC SET INKS

Sets the colours of all the PENs and the border.

ENTRY

- **DE** contains the address of the ink vector.

EXIT

- **AF** is corrupt, and all other registers are preserved.

Notes

The ink vector takes the following form:

- **byte 0** - holds the colour for the border
- **byte 1** - holds the colour for PEN 0
- …
- **byte 16** - holds the colour for PEN 15

The values for the colours are all given as hardware values; the routine sets all sixteen PEN's.

&BD28 - MC RESET PRINTER

Sets the MC WAIT PRINTER indirection to its original routine.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BD2B - MC PRINT CHAR

Sends a character to the printer and detects if it is busy for too long (more than 0.4 seconds).

ENTRY

- **A** contains the character to be printed - only characters upto ASCII 127 can be printed.

EXIT

- If the character was sent properly, then Carry is true.
- If the printer was busy, then Carry is false.
- In either case, **A** and the other flags are corrupt, and all other registers are preserved.

Notes

This routine uses the **MC WAIT PRINTER** indirection.

&BD2E - MC BUSY PRINTER

Tests to see if the printer is busy.

EXIT

- If the printer is busy, then Carry is true.
- If the printer is not busy, then Carry is false.
- In both cases, the other flags are corrupt, and all other registers are preserved.

&BD31 - MC SEND PRINTER

Sends a character to the printer, which must not be busy.

ENTRY

- **A** contains the character to be printed - only characters up to ASCII 127 can be printed.

EXIT

- Carry is true.
- **A** and the other flags are corrupt, and all other registers are preserved.

&BD34 - MC SOUND REGISTER

Sends data to a sound chip register

ENTRY

- **A** contains the register number.
- **C** contains the data to be sent.

EXIT

AF and **BC** are corrupt, and all other registers are preserved.

&BD37 - JUMP RESTORE

Restores the jumpblock to its default state.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

Notes

This routine does not affect the indirections jumpblock, but restores all entries in the main jumpblock.

&BD3A - KM SET LOCKS

Turns the shift and caps locks on and off.

This vector is only available on firmware V2 and later [CPC664, CPC6128, 464Plus, 6128Plus and GX-4000].

ENTRY

- **H** contains the caps lock state.
- **L** contains the shift lock state.

EXIT

- **AF** is corrupt, and all others are preserved.

Notes

In this routine, **&00** means turned off, and **&FF** means turned on.

&BD3D - KM FLUSH

Empties the key buffer.

This vector is only available on firmware V2 and later [CPC664, CPC6128, 464Plus, 6128Plus and GX-4000].

EXIT

- **AF** is corrupt, and all other registers are preserved.

Notes

This routine also discards any current expansion string.

&BD40 - TXT ASK STATE

Gets the VDU and cursor state.

This vector is only available on firmware V2 and later (CPC664, CPC6128, 464Plus, 6128Plus and GX-4000).

EXIT

- **A** contains the VDU and cursor state.
- Flags are corrupt, and all other registers are preserved.

Notes

The value in the **A** register is bit significant, as follows:

- if **bit 0** is set, then the cursor is **disabled**, otherwise it is enabled.
- if **bit 1** is set, then the cursor is turned **off**, otherwise it is on.
- if **bit 7** is set, then the VDU is **enabled**, otherwise it is disabled.

&BD43 - GRA DEFAULT

Sets the graphics VDU to its default mode.

This vector is only available on firmware V2 and later (CPC664, CPC6128, 464Plus, 6128Plus and GX-4000).

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

Notes

Sets the background to opaque, the first point of line is plotted, lines aren't dotted, and the write mode is force.

&BD46 - GRA SET BACK

Sets the graphics background mode to either opaque or transparent.

This vector is only available on firmware V2 and later (CPC664, CPC6128, 464Plus, 6128Plus and GX-4000).

ENTRY

- **A** holds zero if opaque mode is wanted, or holds &FF to select transparent mode.

EXIT

- All registers are preserved.

&BD49 - GRA SET FIRST

Sets whether the first point of a line is plotted or not.

This vector is only available on firmware V2 and later (CPC664, CPC6128, 464Plus, 6128Plus and GX-4000).

ENTRY

- **A** holds zero if the first point is not to be plotted, or holds **&FF** if it is to be plotted.

EXIT

- All registers are preserved.

&BD4C - GRA SET LINE MASK

Sets how the points in a line are plotted - ie defines whether a line is dotted or not.

This vector is only available on firmware V2 and later (CPC664, CPC6128, 464Plus, 6128Plus and GX-4000).

ENTRY

- **A** contains the line mask that will be used when drawing lines.

EXIT

- All registers are preserved.

Notes

The first point in the line corresponds to bit 7 of the line mask and after bit 0 the mask repeats ; if a bit is set then that point will be plotted; the mask is always applied from left to right, or from bottom to top.

&BD4F - GRA FROM USER

Converts user coordinates into base coordinates.

This vector is only available on firmware V2 and later (CPC664, CPC6128, 464Plus, 6128Plus and GX-4000).

ENTRY

- **DE** contains the user X coordinate.
- **HL** contains the user Y coordinate.

EXIT

- **DE** holds the base X coordinate.
- **HL** holds the base Y coordinate.
- **AF** is corrupt, and all other registers are preserved.

&BD52 - GRA FILL

Fills an area of the screen starting from the current graphics position and extending until it reaches either the edge of the window or a pixel set to the PEN.

This vector is only available on firmware V2 and later (CPC664, CPC6128, 464Plus, 6128Plus and GX-4000).

ENTRY

- **A** holds a PEN to fill with.
- **HL** holds the address of the buffer.
- **DE** holds the length of the buffer.

EXIT

- If the area was filled properly, then Carry is true.
- If the area was not filled, then Carry is false.
- In either case, **A**, **BC**, **DE**, **HL** and the other flags are corrupt, and all other registers are preserved.

Notes

The buffer is used to store complex areas to fill, which are remembered and filled when the basic shape has been done ; each entry in the buffer uses seven bytes and so the more complex the shape the larger the buffer ; if it runs out of space to store these complex areas, it will fill what it can and then return with Carry false.

&BD55 - SCR SET POSITION

Sets the screen base and offset without telling the hardware.

This vector is only available on firmware V2 and later (CPC664, CPC6128, 464Plus, 6128Plus and GX-4000).

ENTRY

- A contains the screen base.
- HL contains the screen offset.

EXIT

- A contains the masked screen base.
- HL contains the masked screen offset, the flags are corrupt, and all other registers are preserved.

&BD58 - MC PRINT TRANSLATION

Sets how ASCII characters will be translated before being sent to the printer.

This vector is only available on firmware V2 and later (CPC664, CPC6128, 464Plus, 6128Plus and GX-4000).

ENTRY

- HL contains the address of the table.

EXIT

- If the table is too long, then Carry is false (ie more than 20 entries).
- If the table is correctly set out, then Carry is true.
- In either case, A, BC, DE, HL and the other flags are corrupt, and all others are preserved.

Notes

The first byte in the table is the number of entries. Each entry requires two bytes, as follows:

- **byte 0** - the character to be translated.
- **byte 1** - the character that is to be sent to the printer If the character to be sent to the printer is **&FF**, then the character is ignored and nothing is sent.

&BD5B - KL BANK SWITCH

Sets which RAM banks are being accessed by the Z80.

This vector is only available on firmware V2 and later (CPC664, CPC6128, 464Plus, 6128Plus and GX-4000).

ENTRY

- **A** contains the organisation that is to be used.

EXIT

- **A** contains the previous organisation
- Flags are corrupt, and all other registers are preserved.

THE FIRMWARE INDIRECTIONS

&BDCD - TXT DRAW CURSOR

Places the cursor on the screen, if the cursor is enabled.

EXIT

- **AF** is corrupt, and all other registers are preserved.

Notes

The cursor is an inverse blob which appears at the current text position.

&BDD0 - TXT UNDRAW CURSOR

Removes the cursor from the screen, if the cursor is enabled.

EXIT

- **AF** is corrupt, and all the other registers are preserved.

&BDD3 - TXT WRITE CHAR

Writes a character onto the screen.

ENTRY

- **A** holds the character to be written.
- **H** holds the physical column number.
- **L** holds the physical line number.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

&BDD6 - TXT UNWRITE

Reads a character from the screen.

ENTRY

- **H** contains the physical column number.
- **L** contains the physical line number to read from.

EXIT

- If a character was found, then Carry is true, and **A** contains the character.
- If no character was found, then Carry is false, and **A** contains zero.
- In either case, **BC**, **DE**, **HL** and the other flags are corrupt, and all other registers are preserved.

Notes

This routine works by comparing the image on the screen with the character matrices ; therefore if the character matrices have been altered the routine may not find a readable a character.

&BDD9 - TXT OUT ACTION

Writes a character to the screen or obeys a control code [**&00** to **&1F**].

ENTRY

- **A** contains the character or code.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

Notes

Control codes may take a maximum of nine parameters. When a control code is found, the required number of parameters is read into the

control code buffer, and then the control code is acted upon ; if the graphics character write mode is enabled, then characters and codes are printed using the graphics VDU ; when using the graphics VDU control codes are printed and not obeyed.

&BDDC - GRA PLOT

Plots a point in the current graphics PEN.

ENTRY

- **DE** contains the user X coordinate.
- **HL** contains the user Y coordinate of the point.

EXIT

- **AF, BC, DE** and **HL** are corrupt, and all other registers are preserved.

Notes

This routine uses the **SCR WRITE** indirection to write the point to the screen.

&BDDF - GRA TEST

Tests a point and finds out what PEN it is set to.

ENTRY

- **DE** contains the user X coordinate.
- **HL** contains the user Y coordinate of the point.

EXIT

- **A** contains the PEN that the point is written in
- **BC, DE** and **HL** are corrupt, and all other registers are preserved.

Notes

This routine uses the **SCR READ** indirection to test a point on the screen.

&BDE2 - GRA LINE

Draws a line in the current graphics PEN, from the current graphics position to the specified point.

ENTRY

- **DE** contains the user X coordinate.
- **HL** contains the user Y coordinate for the endpoint.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

Notes

This routine uses the **SCR WRITE** indirection to write the points of the line on the screen.

&BDE5 - SCR READ

Reads a pixel from the screen and returns its decoded PEN.

ENTRY

- **HL** contains the screen address of the pixel.
- **C** contains the mask for the pixel.

EXIT

- **A** contains the decoded PEN of the pixel.
- Flags are corrupt, and all others are preserved.

Notes

The mask should be for a single pixel, and is dependent on the screen mode.

&BDE8 - SCR WRITE

Writes one or more pixels to the screen.

ENTRY

- **HL** contains the screen address of the pixel.
- **C** contains the mask.
- **B** contains the encoded PEN.

EXIT

- **AF** is corrupt, and all other registers are preserved.

Notes

The mask should determine which pixels in the screen byte are to be plotted.

&BDEB - SCR MODE CLEAR

Fills the entire screen memory with &00, which clears the screen to PEN 0.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all the other registers are preserved.

&BDEE - KM TEST BREAK

Tests if the ESC key has been pressed, and acts accordingly.

ENTRY

- Interrupts must be disabled.
- **C** contains the Shift and Control key states.

EXIT

- **AF** and **HL** are corrupt, and all other registers are preserved.

Notes

- If **bit 7 of C** is set, then the Control key is pressed.
- If **bit 5 of C** is set, then the Shift key is pressed.
- If **ESC, Shift and Control** are pressed at the same time, then it initiates a system reset, otherwise it reports a break event.

&BDF1 - MC WAIT PRINTER

Sends a character to the printer if it is not busy

ENTRY

- **A** contains the character to be sent to the printer.

EXIT

- If the character was printed successfully, then Carry is true.
- If the printer was busy for too long (more than 0.4 seconds), then Carry is false.
- In either case, **A** and **BC** are corrupt, and all other registers are preserved.

&BDF4 - KM SCAN KEYS

Scans the keyboard every 1/50th of a second, and updates the status of all keys.

ENTRY

- All interrupts must be disabled.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt, and all other registers are preserved.

THE MATHS FIRMWARE

&BD61 - MOVE REAL

(&BD3D for the 464) Copies the five bytes that are pointed to by **DE** to the location held in **HL**.

ENTRY

- **DE** points to the source real value.
- **HL** points to the destination.

EXIT

- **HL** points to the real value in the destination.
- Carry is true if the move went properly.
- **F** is corrupt, and all other registers are preserved.

Notes

For the 464 only, **A** holds the exponent byte of the real value when the routine is exited.

&BD64 - INTEGER TO REAL

(&BD40 for the 464) Converts an integer value into a real value.

ENTRY

- **HL** holds the integer value.
- **DE** points to the destination for the real value.
- **bit 7 of A** holds the sign of the integer value (it is taken to be negative if bit 7 is set).

EXIT

- **HL** points to the real value in the destination.
- **AF** and **DE** are corrupt, and all others are preserved.

&BD67 - BINARY TO REAL

(&BD43 for the 464) Converts a four byte binary value into a real value at the same location.

ENTRY

- **HL** points to the binary value.
- **bit 7 of A** holds the sign of the binary value (negative if it is set).

EXIT

- **HL** points to the real value in lieu of the four byte binary value.
- **AF** is corrupt, and all others are preserved.

Notes

A four byte binary value is an unsigned integer up to **&FFFFFFFF** and is stored with the least significant byte first, and with the most significant byte last.

&BD6A - REAL TO INTEGER

(&BD46 for the 464) Converts a real value, rounding it into an unsigned integer value held in **HL**.

ENTRY

- **HL** points to the real value.

EXIT

- **HL** holds the integer value.
- Carry is true if the conversion worked successfully.
- the Sign flag holds the sign of the integer (negative if it is set).
- **A, IX** and the other flags are corrupt, and all other registers are preserved.

Notes

This rounds the decimal part down if it is less than 0.5, but rounds up if it is greater than, or equal to 0.5

&BD6D - REAL TO BINARY

[&BD49 for the 464] Converts a real value, rounding it into a four byte binary value at the same location.

ENTRY

- **HL** points to the real value.

EXIT

- **HL** points to the binary value in lieu of the real value.
- **bit 7 of B** holds the sign for the binary value [it is negative if bit 7 is set].
- **AF**, **B** and **IX** are corrupt, and all other registers are preserved.

Notes

See **REAL TO INTEGER** for details of how the values are rounded up or down.

&BD70 - REAL FIX

[&BD4C for the 464] Performs an equivalent of BASIC's **FIX** function on a real value, leaving the result as a four byte binary value at the same location.

ENTRY

- **HL** points to the real value.

EXIT

- **HL** points to the binary value in lieu of the real value.
- **bit 7 of B** has the sign of the binary value [it is negative if bit 7 is set].
- **AF**, **B** and **IX** are corrupt, and all other registers are preserved.

Notes

FIX removes any decimal part of the value, rounding down whether positive or negative - see the BASIC handbook for more details on the **FIX** command.

&BD73 - REAL INT

(&BD4F for the 464) Performs an equivalent of BASIC's **INT** function on a real value, leaving the result as a four byte binary value at the same location

ENTRY

- **HL** points to the real value.

EXIT

- **HL** points to the binary value in lieu of the real value.
- **bit 7 of B** has the sign of the binary value (it is negative if bit 7 is set).
- **AF**, **B** and **IX** are corrupt, and all other registers are preserved.

Notes

INT removes any decimal part of the value, rounding down if the number is positive, but rounding up if it is negative.

&BD76 - INTERNAL SUBROUTINE

not useful (&BD52 for the 464)

&BD79 - REAL *10^A

(&BD55 for the 464) Multiplies a real value by 10 to the power of the value in the **A** register, leaving the result at the same location.

ENTRY

- **HL** points to the real value.
- **A** holds the power of 10.

EXIT

- **HL** points to the result.
- **AF**, **BC**, DE, IX and IY are corrupt.

&BD7C - REAL ADDITION

(&BD58 for the 464) Adds two real values, and leaves the result in lieu of the first real number.

ENTRY

- **HL** points to the first real value.
- **DE** points to the second real value.

EXIT

- **HL** points to the result.
- **AF**, **BC**, **DE**, **IX** and **IY** are corrupt.

&BD82 - REAL REVERSE SUBTRACTION

[&BD5E for the 464] Subtracts the first real value from the second real value, and leaves the result in lieu of the first number.

ENTRY

- **HL** points to the first real value.
- **DE** points to the second real value.

EXIT

- **HL** points to the result in place of the first real value.
- **AF**, **BC**, **DE**, **IX** and **IY** are corrupt.

&BD85 - REAL MULTIPLICATION

[&BD61 for the 464] Multiplies two real values together, and leaves the result in lieu of the first number.

ENTRY

- **HL** points to the first real value.
- **DE** points to the second real value.

EXIT

- **HL** points to the result in place of the first real value.
- **AF**, **BC**, **DE**, **IX** and **IY** are corrupt.

&BD88 - REAL DIVISION

[&BD64 for the 464] Divides the first real value by the second real value, and leaves the result in lieu of the first number.

ENTRY

- **HL** points to the first real value.
- **DE** points to the second real value.

EXIT

- **HL** points to the result in place of the first real value.
- **AF, BC, DE, IX** and **IY** are corrupt.

&BD8E - REAL COMPARISON

[&BD6A for the 464] Compares two real values.

ENTRY

- **HL** points to the first real value.
- **DE** points to the second real value.

EXIT

- **A** holds the result of the comparison process.
- **IX, IY**, and the other flags are corrupt, and all others are preserved.

Notes

After this routine has been called, the value in **A** depends on the result of the comparison as follows:

- if the first real number is greater than the second real number, then **A** holds **&01**.
- if the first real number is the same as the second real number, then **A** holds **&00**.
- if the second real number is greater than the first real number, then **A** holds **&FF**.

&BD91 - REAL UNARY MINUS

[&BD6D for the 464] Reverses the sign of a real value.

ENTRY

- **HL** points to the real value.

EXIT

- **HL** points to the new value of the real number [which is stored in place of the original number].
- **bit 7 of A** holds the sign of the result [it is negative if bit 7 is set].
- **AF** and **IX** are corrupt, and all other registers are preserved.

&BD94 - REAL SIGNUM/SGN

[&BD70 for the 464] Tests a real value, and compares it with zero.

ENTRY

- **HL** points to the real value.

EXIT

- **A** holds the result of this comparison process.
- **IX** and the other flags are corrupt, and all others are preserved.

Notes

After this routine has been called, the value in **A** depends on the result of the comparison as follows:

- if the real number is greater than 0, then **A** holds **&01**,Carry is false, and Zero is false.
- if the real number is the same as 0, then **A** holds **&00**, Carry is false, and Zero is true.
- if the real number is smaller than 0, then **A** holds **&FF**, Carry is true, and Zero is false.

&BD97 - SET ANGLE MODE

(&BD73 for the 464) Sets the angular calculation mode to either degrees (DEG) or radians (RAD).

ENTRY

- **A** holds the mode setting - 0 for RAD, and any other value for DEG.

EXIT

- All registers are preserved.

&BD9A - REAL PI

(&BD76 for the 464) Places the real value of pi at a given memory location.

ENTRY

- **HL** holds the address at which the value of pi is to be placed.

EXIT

- **AF** and **DE** are corrupt, and all other registers are preserved.

&BD9D - REAL SQR

(&BD79 for the 464) Calculates the square root of a real value, leaving the result in lieu of the real value.

ENTRY

- **HL** points to the real value.

EXIT

- **HL** points to the result of the calculation.
- **AF**, **BC**, **DE**, **IX** and **IY** are corrupt.

&BDA0 - REAL POWER

[&BD7C for the 464] Raises the first real value to the power of the second real value, leaving the result in lieu of the first real value.

ENTRY

- **HL** points to the first real value.
- **DE** points to the second real value.

EXIT

- **HL** points to the result of the calculation.
- **AF**, **BC**, **DE**, **IX** and **IY** are corrupt.

&BDA3 - REAL LOG

[&BD7F for the 464] Returns the naperian logarithm (to base e) of a real value, leaving the result in lieu of the real value.

ENTRY

- **HL** points to the real value.

EXIT

- **HL** points to the logarithm that has been calculated.
- **AF**, **BC**, **DE**, **IX** and **IY** are corrupt.

&BDA6 - REAL LOG 10

[&BD82 for the 464] Returns the logarithm [to base 10] of a real value, leaving the result in lieu of the real value.

ENTRY

- **HL** points to the real value.

EXIT

- **HL** points to the logarithm that has been calculated.
- **AF**, **BC**, **DE**, **IX** and **IY** are corrupt.

&BDA9 - REAL EXP

[&BD85 for the 464] Returns the antilogarithm [base e] of a real value, leaving the result in lieu of the real value.

ENTRY

- **HL** points to the real value.

EXIT

- **HL** points to the antilogarithm that has been calculated.
- **AF**, **BC**, **DE**, **IX** and **IY** are corrupt.

Notes

See the BASIC handbook for details of **EXP**.

&BDAC - REAL SINE

[&BD88 for the 464] Returns the sine of a real value, leaving the result in lieu of the real value.

ENTRY

- **HL** points to the real value [ie all angle].

EXIT

- **HL** points to the sine value that has been calculated.
- **AF**, **BC**, **DE**, **IX** and **IY** are corrupt.

&BDAF - REAL COSINE

[&BD8B for the 464] Returns the cosine of a real value, leaving a the result in lieu of the real value.

ENTRY

- **HL** points to the real value [ie an angle].

EXIT

- **HL** points to the cosine value that has been calculated,
- **AF**, **BC**, **DE**, **IX** and **IY** are corrupt.

&BDB2 - REAL TANGENT

[&BD8E for the 464] Returns the tangent of a real value, leaving the result in lieu of the real value.

ENTRY

- **HL** points to the real value [ie an angle].

EXIT

- **HL** points to the tangent value that has been calculated.
- **AF**, **BC**, **DE**, **IX** and **IY** are corrupt.

&BDB5 - REAL ARCTANGENT

(&BD91 for the 464) Returns the arctangent of a real value, leaving the result in lieu of the real value.

ENTRY

- **HL** points to the real value (ie an angle).

EXIT

- **HL** points to the arctangent value that has been calculated.
- **AF, BC, DE, IX** and **IY** are corrupt.

All of the above routines to calculate sine, cosine, tangent and arctangent are slightly inaccuarate.

&BDB8 - INTERNAL SUBROUTINE

not useful [**&BD94** for the 464]

&BDBB - INTERNAL SUBROUTINE

not useful [**&BD97** for the **464**]

&BDBE - INTERNAL SUBROUTINE

not useful [**&BD9A** for the **464**]

SUBROUTINES FOR THE 664 AND 6128 ONLY

&BD5E - TEXT INPUT

Allows upto 255 characters to be input from the keyboard into a buffer.

ENTRY

- **HL** points to the start of the buffer.

A **NUL** character must be placed after any characters already present, or at the start of the buffer if there is no text

EXIT

- **A** has the last key pressed.
- **HL** points to the start of the buffer.
- the flags are corrupt, and all others are preserved.

Notes

This routine prints any existing contents of the buffer (upto the NUL character) and then echoes any keys used; it allows full line editing with the cursor keys and DEL, etc. It is exited only by use of ENTER or ESC.

&BD7F - REAL RND

Creates a new RND real value at a location pointed to by **HL**.

ENTRY

- **HL** points to the destination for the result.

EXIT

- **HL** points to the RND value.
- **AF, BC, DE** and **IX** registers are corrupt.
- All other registers are preserved.

&BD8B - REAL RND(0)

Returns the last RND value created, and puts it in a location pointed to by HL.

ENTRY

- **HL** points to the place where the value is to be returned to.

EXIT

- **HL** points to the value created.
- **AF**, **DE** and **IX** are corrupt.
- All other registers are preserved.

Notes

See the BASIC handbook for more details on RND[0].

AMSDOS AND BIOS FIRMWARE

&C033 - BIOS SET MESSAGE

Enables or disables disc error messages

ENTRY

- To enable messages, **A** holds &00.
- To disable messages, **A** holds &FF.

EXIT

- **A** holds the previous state,
- **HL** and the flags are corrupt
- All other registers are preserved.

Notes

Enabling and disabling the messages can also be achieved by poking **&BE78** with **&00** or **&FF**.

&C036 - BIOS SETUP DISC

Sets the parameters which effect the disc speed.

ENTRY

- **HL** holds the address of the nine bytes which make up the parameter block.

EXIT

- **AF**, **BC**, **DE** and **HL** are corrupt.
- All other registers are preserved.

Notes

The parameter block is arranged as follows:

- **bytes 0&1** - the motor on time in 20ms units ; the default is &0032; the fastest is **&0023**
- **bytes 2&3** - the motor off time in 20ms units ; the default is &00FA; the fastest is **&00C8**
- **byte 4** - the write off time in 10ms units ; the default is &AF; should not be changed
- **byte 5** - the head settle time in 1ms units ; the default is &0F; should not be changed
- **byte 6** - the step rate time in 1ms units ; the default is &0C; the fastest is **&0A**
- **byte 7** - the head unload delay; the default is **&01** ; should not be changed
- **byte 8** - a byte of **&03** and this should be left unaltered

&C039 - BIOS SELECT FORMAT

Sets a format for a disc.

ENTRY

- **A** holds the type of format that is to be selected.

EXIT

- **AF, BC, DE** and **HL** are corrupt.
- All the other registers are preserved.

Notes

To select one of the normal disc formats, the following values should be put into the **A** register:

- **&C1**: Data format.
- **&41**: System format (Used by CP/M).
- **&01**: IBM format (compatible with CP/M-86).

This routine sets the extended disc parameter block (**XDPB**) at **&A890** to **&A8A8**. To set other formats, the **XDPB** must be altered directly.

&C03C - BIOS READ SECTOR

Reads a sector from a disc into memory.

ENTRY

- **HL** holds the address in memory where the sector will be read to.
- **E** holds the drive number [&00 for drive A and &01 for drive B].
- **D** holds the track number.
- **C** holds the sector number.

EXIT

- If the sector was read properly, then Carry is true, **A** holds 0, and **HL** is preserved.
- If the read failed, then Carry is false, **A** holds an error number, and **HL** is corrupt.

In either case, the other flags are corrupt, and all other registers are preserved.

&C03F - BIOS WRITE SECTOR

Writes a sector from memory onto disc.

ENTRY

- **HL** holds the address of memory which will be written to the disc.
- **E** holds the drive number [&00 for drive A, and &01 for drive B].
- **D** holds the track number.
- **C** holds the sector number.

EXIT

- If the sector was written properly, then Carry is true, **A**

holds 0, and **HL** is preserved.
- If the write failed, then Carry is false, **A** holds an error number, and **HL** is corrupt.

In either case, the other flags are corrupt, and all other registers are preserved.

&C042 - BIOS FORMAT TRACK

Formats a complete track, inserts sectors, and fills the track with bytes of **&E5**.

ENTRY

- **HL** contains the address of the header information buffer which holds the header information blocks.
- **E** contains the drive number [&00 for drive A, and &01 for drive B].
- **D** holds the track number.

EXIT

- if the formatting process was successful, then Carry is true, **A** holds 0, and **HL** is preserved.
- if the formatting process failed, then Carry is false, **A** holds an error number, and **HL** is corrupt.

In either case, the other flags are corrupt, and all the other registers are preserved.

Notes

The header information block is laid out as follows:
- **byte 0** - holds the track number
- **byte 1** - holds the head number [set to zero]
- **byte 2** - holds the sector number
- **byte 3** - holds log2[sector size]-7 [usually either **&02=512** bytes, or **&03=1024** bytes].

Header information blocks must be set up contiguously for every sector on the track, and in the same sequence that they are to be laid down [eg **&C1, &C6, &C2, &C7, &C3, &C8, &C4, &C9, &C5**].

&C045 - BIOS MOVE TRACK

Moves the disc drive head to the specified track.

ENTRY

- **E** holds the drive number [&00 for drive A, and &01 for drive B].
- **D** holds the track number.

EXIT

- If the head was moved successfully, then Carry is true, **A** holds **0**, and **HL** is preserved.
- If the move failed, then Carry is false, **A** holds an error number, and **HL** is corrupt.

In both cases, the other flags are corrupt, and all other registers are preserved.

Notes

There is normally no need to call this routine as **READ SECTOR**, **WRITE SECTOR** and **FORMAT TRACK** automatically move the head to the correct position

&C048 - BIOS GET STATUS

Returns the status of the specified drive.

ENTRY

- **A** holds the drive number [&00 for drive A, and &01 for drive B].

EXIT

- If Carry is true, then **A** holds the status byte and **HL** is preserved.
- If Carry is false, then **A** is corrupt, and **HL** holds the address of the byte before the status byte.

In either case, the other flags are preserved, and all other registers are preserved.

Notes

The status byte indicates the drive's status as follows:

- if **bit 6** is set, then either the write protect is set or the disc is missing
- if **bit 5** is set, then the drive is ready and the disc is fitted [whether the disc is formatted or not]
- if **bit 4** is set, then the head is at track **0**

&C04B - BIOS SET RETRY COUNT

Sets the number of times the operation is retried in the event of disc error.

ENTRY

- **A** holds the number of retries required.

EXIT

- **A** holds the previous number of retries.
- **HL** and the flags are corrupt.
- All other registers are preserved.

Notes

The default setting is &10, and the minimum setting is &01; the number of retries can also be altered by poking &BE66 with the required value.

&C56C - GET SECTOR DATA

Gets the data of a sector on the current track.

ENTRY

- **E** holds the drive number.

EXIT

- If a formatted disc is present, then Carry is true, and **HL** is preserved.
- If an unforrnatted disc is present or the disc is missing, then Carry is false, and **HL** holds the address of the byte before the status byte.

In either case, **A** and the other flags are corrupt, and all other registers are preserved.

Notes

The track number is held at &BE4F, the head number is held at &BE50, the sector number is held at &BE51, and the `log2(sector size)-7` is held at &BE52; disc parameters do not need to be set to the format of the disc; this routine is best used with the disc error messages turned off.