6 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.

6.1 Screen Modes.

The screen has three modes of operation, numbered 0, 1 and 2. The modes have different resolutions and display different numbers of inks on the screen.

All modes have a vertical resolution of 200 pixels (picture elements or dots on the screen). The horizontal resolution varies from 160 pixels to 640 pixels. As each character is 8 pixels by 8 pixels the number of characters across the screen varies with the mode - from 20 characters to 80 characters. The screen is always 25 characters high.

The number of inks that can be displayed on the screen varies with the screen resolution. When the screen is 640 pixels wide only 2 inks can be displayed, when the screen is 320 pixels wide 4 inks can be displayed and when the screen is 160 pixels wide 16 inks can be displayed.

In summary, the modes are:

Mode	Pixel Size	Character Size	Inks
0	160x200	20x25	16
1	320x200	40x25	4
2	640x200	80x25	2

The default screen mode, set at EMS, is mode 1.

The screen mode is set by calling SCR SET MODE which also has other effects.

Firstly, the screen is cleared to ink 0. If the text and graphics paper inks are not set to ink 0 then this will become apparent on the screen when characters are written or windows are cleared. If the user wishes to alter this screen clearing operation for some reason then it may be intercepted at the SCR MODE CLEAR indirection.

Secondly, the Text and Graphics VDUs are set into standard states. The windows are all set to cover the whole screen. If the pen and paper inks are out of range for the new mode then they are masked (with #01 or #03) to bring them into range. The current text positions are moved to the top left corner of the screen and the text cursors are turned off (see TXT CUR OFF). The current graphics position and the user origin are moved to the bottom left corner of the screen.

6.2 Inks and Colours.

The various screen modes allow pixels (dots on the screen) to be set to different numbers of inks as follows:

Mode 0:	16 inks, 0..15
Mode 1:	4 inks, 0..3
Mode 2:	2 inks, 0..1

How the ink for a pixel is encoded into a byte of screen memory is described in section 6.4. The ink that a pixel is set to determines what colour the pixel is displayed in. However, the colour associated with an ink is not fixed, it can be changed.

There are 27 colours available. Each ink may be set to any of these colours. The border to the screen acts much like an ink and can have its colour specified as well. The display hardware fetches the ink value from the screen memory for each pixel as it is displayed. This ink value is used to access a small area of RAM inside the gate array called the 'palette'. The palette contains the actual colour which is to be displayed by the monitor for that particular ink. Changing entries in the palette thus causes all pixels set to that ink to change colour when they are next displayed (i.e. within 1/50th of a second or so).

In fact the Screen Pack allows two colours to be associated with an ink (or the border). These are loaded into the palette alternately under software control. If the two colours associated with an ink are different then the ink will flash, if the colours are the same then the ink will be steady. The user can change the rate of alternation, from the default of 5 cycles per second, if required (see SCR SET FLASHING).

When specifying colours the Screen Pack uses an ordering that corresponds to a grey scale on a monochrome monitor. This runs from the darkest colour (black), colour 0, to the brightest colour (bright white), colour 26. The colours do not appear to have any particular ordering when viewed on a colour monitor.

The palette uses a different (and apparently nonsensical) numbering scheme for the colours. The Screen Pack automatically translates the grey scale number to the hardware number and vice versa when appropriate. Unless the user is driving the hardware directly the hardware numbers will never be encountered.
The default settings for the colour of each ink and a list of the 27 colours available are given in Appendix V.

6.3 Screen Addresses.

The Screen Pack does not use a coordinate system itself. It uses screen addresses. However, it does work with the physical and base coordinate systems of the Text and Graphics VDUs described in sections 4.1 and 5.1 respectively. In particular, routines are provided to convert positions given in physical or base coordinates to screen addresses.

A screen address is, prosaically enough, the address of a byte within the screen memory. To specify a particular pixel a screen address is often passed to a routine along with a mask that indicates exactly which pixel is required. Routines are provided for stepping a screen address up, down, right and left one byte. (The screen map makes this a non-trivial operation.)

6.4 Screen Memory Map.

The screen is a memory mapped pixel screen. The screen memory fills 16K of RAM in all modes. The default location for the screen, set at EMS, is the 16K of RAM starting at #C000. This lies underneath the upper ROM, when it is enabled, which keeps the screen out of the way of the rest of the system. However, this also means that the upper ROM has to be disabled whenever the screen is read. The firmware jumpblock uses LOW JUMP restarts which turn the upper ROM off to ensure that the screen memory is accessible if required.

It is possible to change the location of the screen memory to any of the 4 16K memory blocks on 16K boundaries (see SCR SET BASE). However, only #C000 and #4000 are useful; #0000 and #8000 both overlap firmware jumpblocks or other system areas. The descriptions below all assume the default screen location at #C000.

In V1.1 firmware it is possible to set the location of the screen that is used by the screen pack routines independently of setting the hardware value. This will then enable text and graphics to be produced in the 'screen' that isn't currently on view - switching to the other possible location (#4000 to #C000) will cause the already prepared graphics etc. to instantly appear - thus avoiding flicker and enabling smooth animation effects.

The screen memory map is not simple. Fortunately it is not necessary to understand it because the Text and Graphics VDUs provide idealised models of the screen. However, to achieve maximum speed for certain applications (such as animated games) it may be necessary to access the screen memory directly.

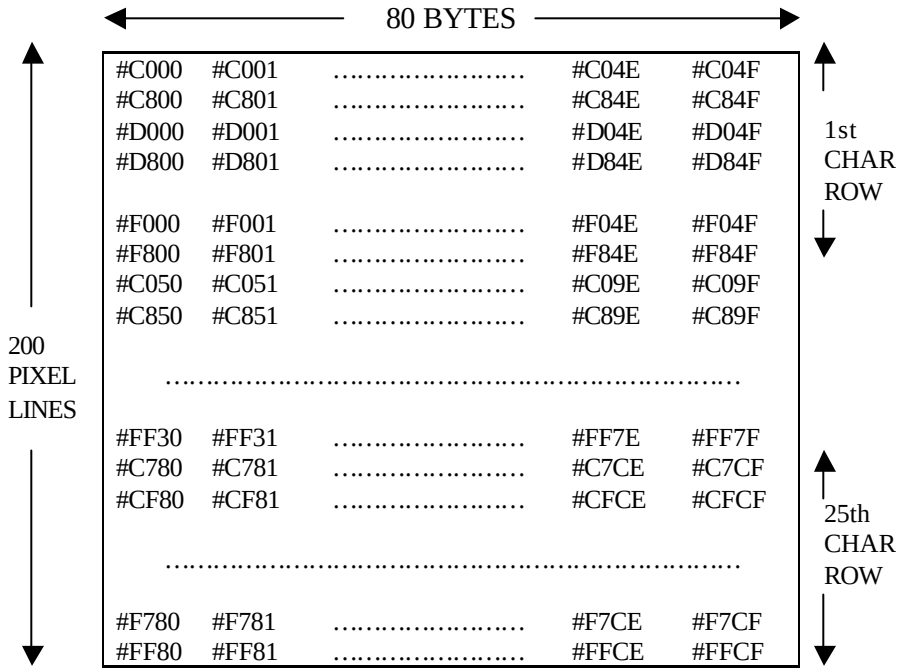
The screen memory is divided into 8 blocks, each 2K bytes long. Block 0 runs from #C000 to #C7FF, block 1 runs from #C800 to #CFFF, and so on. Each line of pixels on the screen uses 80 consecutive bytes from a block. The top line of the screen comes from block 0, the second line from block 1 and so on until the eighth line which comes from block 7. The sequence starts with block 0 again on the ninth line and repeats in this fashion all the way down the screen. The successive lines in a block are stored consecutively so there are 48 unused bytes at the end of each block.

There is a further complication to this screen map. The description above assumes that the first byte displayed from the block is the first byte of the block. In practice the offset in a block of the first byte to be displayed can be set to any even value (see SCR SET OFFSET). The same offset applies to all eight blocks. A block wraps around from its last byte to its first byte, thus #C7FE, #C7FF and #C000 are consecutive bytes in block 0 and could all be on the same line of the screen. Altering the offset by ± 80 MOD 2048 (the length of a line) rolls the screen up or down by one character line (8 pixel lines). This effect is used by the Text VDU when rolling the entire screen.

The meaning of the bytes accessed as described above varies with the screen mode. Each byte stores the inks for 2, 4 or 8 pixels. The bits used to encode each pixel are not arranged in an obvious manner. The following table specifies which bits of screen memory are used to encode which pixel in the various modes. The bit numbers given in the table are the bits of the screen byte. They are given in the order of bits in the pixel - the first bit given is most significant bit of the pixel and the last bit is the least significant bit.

	Mode 0	Mode 1	Mode 2
Leftmost pixel	Bits 1,5,3,7	Bits 3,7	Bit 7
			Bit 6
		Bits 2,6	Bit 5
			Bit 4
	Bits 0,4,2,6	Bits 1,5	Bit 3
			Bit 2
		Bits 0,4	Bit 1
Rightmost pixel			Bit 0

The following diagram illustrates the mapping from pixels on the screen to addresses in screen memory for the simple case of a base address of #C000 and an offset of 0.



#C7D0..#C7FF, #CFD0..#CFFF,..., #FFD0..#FFFF are unused.

On the CPC6128 the base address sets which block will be used for the screen memory. Base addresses of #0000, #4000, #8000 and #C000 correspond to blocks 0, 1, 2, and 3. It is not possible for the screen memory to be located in blocks 4 ... 7. Where the block being used for screen memory actually appears in the memory map depends on the bank switching (see section 2.5).