registers we can now use the new screen as normal. Note that if you wish to try changing the screen's position in BASIC, you will have to write and run a short program to do it.

We can use the Commodore 64's ability to move its screen to produce several interesting utilities. In particular, we can change the display quickly and with ease. The problem is that as we move the screen we also need to move the corresponding colour RAM, because the display will not make any sense unless the colour RAM contains the data appropriate to the screen being displayed. Although we can set up several different screen areas within memory and quickly change between them, there is only one unmoveable area of colour RAM, so to allow us to hold a number of separate screen displays we must set aside areas of memory to hold the 1,000 bytes of colour data for each screen. When we wish to display a screen we must copy this information into the colour RAM, and we must save this data to one of the areas of memory we have designated to hold colour RAM, before changing to a different screen.

ORGANISING MEMORY

The principle task of this utility is to organise memory for alternate screens (along with their corresponding colour data areas), and carry out the transfer of blocks of memory. As the VIC chip can 'see' 16 Kbytes of memory, we can design a system that allows us to have up to eight different screens and a sizeable BASIC program. The diagram shows the arrangement of memory used by the utility.

To ensure that none of the screen or colour areas is overwritten by a BASIC program, we must lower the top of BASIC memory. The following instruction in our BASIC calling program will perform this:

```
POKE 55,0:POKE 56,32:CLR
```

The base address of any screen can be calculated from its number by this formula:

```
Screen Base = $1C00 + ($0400*Screen Number)
```

The base address of the corresponding colour area can be calculated simply by adding an offset to the screen base address. The formula is:

```
Colour Base = $2400 + Screen Base
```

The colour areas could be located anywhere in RAM, but it is most convenient to position them just above the last screen that can be seen by the VIC chip. Notice that a colour area for screen 0 — the normal screen — is included. For this particular colour area, the offset will be different and our program will have to take account of this.

The VIC control register and operating system register can be set for any screen by:

```
VIC register = $70 + ($10*Screen Number)
OS register = HI-byte of Screen Base Address
```

In addition to handling the setting of registers, and making the appropriate transfers of colour data to and from the colour RAM area, the program also stores the colour of the screen background and the border. These two features are controlled by a pair of registers in the VIC chip: 53280 ($D020) controls the border colour and 53281 ($D021) controls the screen background, or paper colour. The utility sets up a table within its own program area to store these two attributes for each screen.

MODES OF OPERATION

The utility has two modes of operation: it can either edit or display a selected screen. In each case, the screen number to be used must be POKEd into 49152 ($C000). So that the same SYS call can be made, we use a special flag to indicate which mode has been selected. This flag is set by POKEing to location 49153 ($C001).

0 – indicates display mode
1 – indicates edit mode

The edit mode works in an unusual way, in order to allow all the facilities of the screen editor (such as changing text colours, setting reverse mode and clearing the screen) to be used. The utility must first be called and the edit flag set to one. It will then save the normal screen colour area and the normal screen paper and border colours, set the VIC and operating system registers appropriately and then return to BASIC. At this stage, the BASIC calling program takes over, homing the cursor and then using the BASIC INPUT instruction. This instruction will wait for a return character (ASCII 13) before moving on. In the meantime, all the screen editor functions can be used in the normal way to edit the selected screen, pressing Return when each task is finished.

The BASIC program must then call the utility a second time, but on this occasion the edit flag is set to 2. This will save the colour data and paper/border colours of the screen you have worked on before restoring the normal screen. If you want to change the border or paper colour then the colours must be POKEd into locations 49154 ($C002) and 49155 ($C003) respectively. Although the routine is designed for this particular task, it incorporates general routines to set the control registers and copy data to or from the colour RAM. It would not, therefore, be a difficult task to produce a utility to your own specifications from these general routines.

The BASIC calling program is designed to display a menu giving the option to edit or display. The display routine calls up each of the eight different screens in sequence, in response to a keypress. The screens will continue to cycle round until the Return key is pressed. The program then restores the normal screen and returns to the menu. If the edit option is selected, the user can set the paper and border colours for the selected screen, and may then use the screen editor in the normal way to produce a picture. When you are finished, pressing Return will cause the picture to be stored and the normal screen to be restored.