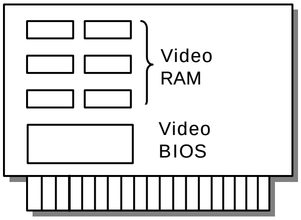
20 Programming Video RAM

"Truth will continue forever."

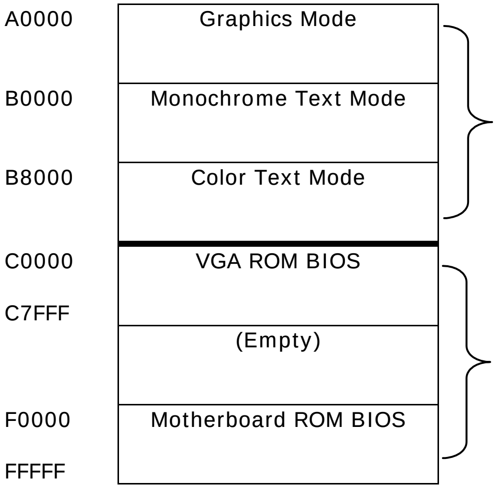
To get a display we have to add a component called video adapter with the motherboard. Hardware engineers sometimes call this video adapter as video card. On the video card we can see a group of video RAM chips. The video card may have upto 8MB in board, but most of them are used by circuits on the card and cannot be directly accessed by processor. In the basic VGA mode (e.g., DOS mode, Windows safe mode), the processor can directly access upto 128KB (i.e., A0000h to BFFFFh) of video RAM . Usually all video cards also have onboard video BIOS normally addressed at C0000h TO C7FFFh.



Video Adapter

20.1 Memory map

Not all the memory is used for display purpose because, we have so many video modes that support different resolutions. The video modes are usually set by the programs that are stored in video ROM BIOS area. Note that it is ROM, which means you cannot write into it! Whereas in video RAM, you can write! But you should know in which display mode, which memory area is used. You can use *far pointers* to write into video RAM. Since VGA and SVGA adapters are used almost everywhere, here I have given the memory map for VGA and SVGA. Other adapters' memory map will be slightly different. If you use other adapters, refer its documentation.



20.2 Programming the video RAM

VGA supports each of the mode supported by its predecessors. VGA is backward compatible. So it is enough to know about programming VGA RAM.

20.2.1 Color Text Mode

This mode uses the video RAMs addressed at B8000 to BFFFFh. In normal color text mode 3h(80x25x16 mode), the address space is divided into 4 video pages of 4KB each (page 0, page 1, page 2 & page 3). At the same time we can see the characters in any one of the pages. The screen's resolution is 80x25 (i.e. 80 columns x 25 rows). It supports 16 colors at a time. To display a single character, two bytes are being used namely character byte and attribute byte. The character byte contains the ASCII value of the character. The attribute byte is organized as:

Bitfields for character's display attribute				
7	654	3	210	Purpose
X				Foreground Blink or (alternate) Background bright
	XXX			Background color
		X		Foreground Bright or (alternate) Alternate character set
			XXX	Foreground color

The following program fills the screen with 'C' with given attributes.

```
#include <dos.h>

#define  _4KB     (4096)      /* size of vdu page */

int main( void )
{
   int i;
   const int attribute = 0x20;
   char far *Vid_RAM;
   FP_SEG( Vid_RAM ) = 0xb800;
   FP_OFF( Vid_RAM ) = 0x0000;
   for ( i=0; i<_4KB ; i +=2 )
     {
      *(Vid_RAM + i) = 'C';
      *(Vid_RAM + i + 1) = attribute;
     }
   return(0);
} /*--main( )-----------*/
```

We can also declare the `Vid_RAM` pointer as

```
      char far *Vid_RAM = (char far*) 0xb8000000;
```

But programmers prefer the declaration, that we used in the above program, because it provides good readability and helps us to clearly identify segment address and offset address.

20.2.1.1 Codes

```
#include <dos.h>

#define  _4KB     (4096)      /* size of vdu page */

char far *Vid_RAM;
```

```
void WriteCh2VidRAM( int vdupage, int x, int y, char ch, int attribute )
{
   FP_SEG( Vid_RAM ) = 0xb800;
   FP_OFF( Vid_RAM ) = 0x0000;

   *(Vid_RAM + _4KB * vdupage + 160 * y + 2 * x) = ch;
   *(Vid_RAM + _4KB * vdupage + 160 * y + 2 * x + 1) = attribute;
} /*--WriteCh2VidRAM( )-----------*/

void WriteStr2VidRAM( int vdupage, int x, int y, char *str, int
attribute )
{
   while(*str)
         WriteCh2VidRAM( vdupage, x++, y, *str++, attribute );
} /*--WriteStr2VidRAM( )------------*/
```

You can use the above functions for normal use. For better programming, you should add condition to check whether the character is on the last row of the screen. In such a case, you have to scroll the screen upward by 1 row.

20.2.1.2 cprintf( )

We have written our functions to directly write into video RAM. But Turbo C also has got inbuilt functions like `cprintf() & cputs()` (defined in `conio.h`) to directly write into video RAM. The global variable `directvideo` determine whether the console output (by cprintf, cputs… functions) go directly to video RAM (`directvideo = 1;`) or go via ROM BIOS calls (`directvideo = 0;`). The default value is `directvideo = 0`. To use `directvideo = 1`, the system's video hardware must be be identical to IBM's display adapter.

The functions of interest in this context are `window(), clrscr(), textcolor(), textbankground(), textattr(), gettextinfo(), highvideo(), normalvideo().`

Following is the example program:

```
#include <conio.h>

int main( void )
{
   clrscr( );
   window( 10,10,40,15 );
   textcolor( WHITE );
   textbackground( RED );
   normvideo( );
   cprintf( "Normal Intensity Text\r\n" );
   textcolor( BLUE );
   textbackground( WHITE );
   lowvideo( );
```

```
      cprintf( "Low Intensity Text\r\n" );
      textcolor( WHITE );
      textbackground( GREEN );
      highvideo( );
      cprintf( "High Intensity Text\r\n" );

      return(0);
   } /*--main( )----------*/
```

20.2.2 Monochrome Text Mode

Monochrome text mode is similar to color text mode. But this mode uses B0000h as a segment address, it displays the character as normal or even reverse video and or underlined for the given attribute colors.

20.2.3 Graphics mode

The segment address of graphics mode is A0000h. mode 13h (320x200x256) and mode 14h (640x480x16) are the modes that are very often used.

Exercises

1. Write a program that finds number of video pages supported by your Video RAM for each mode.
2. Find out the reason, why graphics mode occupies more video memory. (Why graphics mode is slower than text mode?)