# 35 VESA Programming

"Have courage, and be strong."

VESA (Video Electronics Standards Association) is a non-profit organization established to standardize a common software interface to Super VGA video adapters. When IBM ruled the PC world, it came up with its own standard SVGA and BIOS extensions. Few other vendors followed IBM's standard and others introduced their own standards. So it necessitates the need for standardizing the interface or BIOS to Super VGA video adapters. VESA suggests all vendors to use their standard for VGA BIOS extensions. It believes that soon its standard will be set as a standard for all vendors.

## 35.1 Secrets

VESA programming is also sometimes referred as SVGA programming. According to the documentations all windows based systems might have SVGA cards to provide better resolution and more color. Even though VESA standard is introduced to reduce the burden of programming complexity, programmers still face problem with VESA programming. One of the major problems with VESA programming is compatibility. Few people say mode 98h is the standard VESA mode and other say mode 101h & mode 103h are the standard modes! Another problem is we must use interrupts to detect the modes supported by that particular SVGA card. So we cannot have a single procedure, we must have different procedures for each mode! VESA people are standardizing the existing VESA standards and come out with different versions. At present we have VESA3.0. Thus VESA standard is not much standardized and people still go for mode 13h!

## 35.2 Program

The following program shows how to program for VESA. This is a pretty good example.

```
#include <dos.h>

typedef int WORD;
typedef char BYTE;

typedef struct tagVGAINFOBLOCK
  {
      BYTE VESASignature[4];  //'VESA'  signature bytes
      WORD VESAVersion;       // VESA version number
      char far* OEMStringPtr; // Pointer to OEM string
      BYTE Capabilities[4];   // capabilities of the video environment
      char far* VideoModePtr; // pointer to supported Super VGA modes
```

```
      WORD TotalMemory;       // Number of 64kb memory blocks on board
      BYTE Reserved[236];     // Remainder of VgaInfoBlock
  } VGAINFOBLOCK;

typedef struct tagMODEINFOBLOCK
   {
      // mandatory information
      WORD ModeAttributes;   // mode attributes
      BYTE WinAAttributes;   // window A attributes
      BYTE WinBAttributes;   // window B attributes
      WORD WinGranularity;   // window granularity
      WORD WinSize;          // window size
      WORD WinASegment;      // window A start segment
      WORD WinBSegment;      // window B start segment
      char far* WinFuncPtr;  // pointer to windor function
      WORD BytesPerScanLine; // bytes per scan line
      // formerly optional information (now mandatory)
      WORD XResolution;     // horizontal resolution
      WORD YResolution;     // vertical resolution
      BYTE XCharSize;       // character cell width
      BYTE YCharSize;       // character cell height
      BYTE NumberOfPlanes;  // number of memory planes
      BYTE BitsPerPixel;    // bits per pixel
      BYTE NumberOfBanks;   // number of banks
      BYTE MemoryModel;     // memory model type
      BYTE BankSize;        // bank size in kb
      BYTE NumberOfImagePages;      // number of images
      BYTE Reserved1;       // reserved for page function
      // new Direct Color fields
      BYTE RedMaskSize;  // size of direct color red mask in bits
      BYTE RedFieldPosition;  // bit position of LSB of red mask
      BYTE GreenMaskSize;     // size of direct color green mask in bits
      BYTE GreenFieldPosition;      // bit position of LSB of green mask
      BYTE BlueMaskSize;      // size of direct color blue mask in bits
      BYTE BlueFieldPosition; // bit position of LSB of blue mask
      BYTE RsvdMaskSize;   // size of direct color reserved mask in bits
      BYTE DirectColorModeInfo;     // Direct Color mode attributes
      BYTE Reserved2[216];    // remainder of ModeInfoBlock
  } MODEINFOBLOCK;

VGAINFOBLOCK vgainfoblk, *ptr=&vgainfoblk;

void PutPixel( int x, int y, int color )
{
    char far *scr = (char far*)0xA0000000;
    long temp = 0L+ 640*y + x;
    *(scr + temp) = color;
```

```
} /*--PutPixel( )-------*/

int GetVGAInfo( VGAINFOBLOCK *vptr )
{
    unsigned temp;
    asm{
       MOV AH, 4fh;
       MOV AL, 00h;
       }
   temp = FP_SEG( vptr );
   asm      MOV ES, temp;
   temp = FP_OFF( vptr );
   asm{
      MOV DI, temp;
      INT 10h;
      }
   return( _AX );
} /*--GetVGAInfo( )-----------*/

int GetModeInfo( int mode, MODEINFOBLOCK *mptr )
{
    unsigned temp;
    asm{
       MOV AH, 4fh;
       MOV AL, 01h;
       }
   temp = FP_SEG( mptr );
   asm      MOV ES, temp;
   temp = FP_OFF( mptr );
   asm{
      MOV DI, temp;
      MOV CX, mode;
      INT 10h;
      }
   return( _AX );
} /*--GetModeInfo( )---------*/

int GetCurrentMode( void )
{
   asm{
      MOV AX, 4F03h;
      INT 10h;
      }
   return(_BX);
} /*--GetCurrentMode( )--------*/

int SetSVGAMode( int mode )
```

```
{
   asm{
      MOV AX, 4F02h;
      MOV BX, mode;
      INT 10h;
      }
   return( _AX );
} /*--SetSVGAMode( )----------*/

void DemoDraw( void )
{
   int i, j;
   /* Draw some image on the screen */
   for ( j=0 ; j<100; ++j )
     for (i=0;i<256; ++i )
      PutPixel( i,j, i );
} /*--DemoDraw( )-----*/

int main( void )
{
   VGAINFOBLOCK vgainfoblk, *vptr=&vgainfoblk;
   MODEINFOBLOCK modeinfoblk, *mptr=&modeinfoblk;
   int status, oldmode;
   const int mode = 0x0101; // choose your VESA mode

   oldmode = GetCurrentMode( );
   printf( "Current Mode = %Xh \n", oldmode );

   /* if VESA status = 004f, success & supported */
   printf( "VESA status = %X \n", GetVGAInfo( vptr ) );

   /* Print the information about our VESA */
   printf( "VESASignature = %s \n", vptr->VESASignature );
   printf( "VESAVersion = %X \n", vptr->VESAVersion );
   printf( "OEMStringPtr = %s \n", vptr->OEMStringPtr );
   printf( "Capabilities:" );
   if ( vptr->Capabilities[3] & 0x1 )
      printf( " DAC width is switchable \n" );
     else
      printf( " DAC is fixed width, with 6-bits per primary color \n" );
   printf( "TotalMemory = %d X 64kb \n", vptr->TotalMemory );
   getch( );

   status =  GetModeInfo( mode, mptr );
   /* Print the information about the requested mode */
   printf( "mode = %xh\n", mode );
   printf( "~~~~~~~~~~~\n" );
```

```
if ( status==0x004f ) /* success & function supported */
 {
   printf( "ModeAttributes = %d\n", mptr->ModeAttributes );
   printf( "WinAAttributes = %d\n", mptr->WinAAttributes );
   printf( "WinBAttributes = %d\n", mptr->WinBAttributes );
   printf( "WinGranularity = %d\n", mptr->WinGranularity );
   printf( "WinSize = %d\n", mptr->WinSize );
   printf( "WinASegment = %d\n", mptr->WinASegment );
   printf( "WinBSegment = %d\n", mptr->WinBSegment );
   printf( "WinFuncPtr = %s\n", mptr->WinFuncPtr );
   printf( "BytesPerScanLine = %d\n", mptr->BytesPerScanLine );
   printf( "XResolution = %d\n", mptr->XResolution );
   printf( "YResolution = %d\n", mptr->YResolution );
   printf( "XCharSize = %d\n", mptr->XCharSize );
   printf( "YCharSize = %d\n", mptr->YCharSize );
   printf( "NumberOfPlanes = %d\n", mptr->NumberOfPlanes );
   printf( "BitsPerPixel = %d\n", mptr->BitsPerPixel );
   printf( "NumberOfBanks = %d\n", mptr->NumberOfBanks );
   printf( "MemoryModel = %d\n", mptr->MemoryModel );
   printf( "BankSize = %d\n", mptr->BankSize );
   printf( "NumberOfImagePages = %d\n", mptr->NumberOfImagePages );
   printf( "Reserved1 = %d\n", mptr->Reserved1 );
   printf( "RedMaskSize = %d\n", mptr->RedMaskSize );
   printf( "    Continued...\n" );
   getch( );
   printf( "RedFieldPosition = %d\n", mptr->RedFieldPosition );
   printf( "GreenMaskSize = %d\n", mptr->GreenMaskSize );
   printf( "GreenFieldPosition = %d\n", mptr->GreenFieldPosition );
   printf( "BlueMaskSize = %d\n", mptr->BlueMaskSize );
   printf( "BlueFieldPosition = %d\n", mptr->BlueFieldPosition );
   printf( "RsvdMaskSize = %d\n", mptr->RsvdMaskSize );
   printf( "DirectColorModeInfo = %d\n", mptr->DirectColorModeInfo );
   printf( "-------end----\n" );
   printf( "switch to mode %Xh....\n", mode );
   getch( );

   /* Now set to requested mode */

   status = SetSVGAMode( mode );
   if ( status!=0x004F )
         printf( "Error code = %xh \n", status );

else
        {
         DemoDraw( );

         getch( );
```

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