BIOS Specification

6. BIOS SPECIFICATION

This chapter introduces the programmer to the CL-GD546X BIOS requirements, Scratchpad register usage, and a detailed description of the VGA BIOS, VESA® BIOS extensions, and the Cirrus Logic BIOS extensions. The CL-GD546X BIOS is uniquely adapted to meet and exceed the needs of the CL-GD546X VGA modes, VBE/PM v1.0, DDC Level 2B, and VESA VBE v2.0. The CL-GD546X BIOS also supports single or multiple display adapter configurations.

6.1 Requirements

The following sections describe the CL-GD546X BIOS requirements in relation to environments supported, modes, and functions.

6.1.1 Software and Hardware Environments

The CL-GD546X BIOS operates in the standard IBM PC/AT (X-86) software environment. It operates in conjunction with the Microsoft® MS-DOS or equivalent operating systems. The CL-GD546X BIOS is designed to support the Microsoft Windows® 3.1 and Windows 95® operating systems, implementing standard VGA, SVGA, and VBE modes.

The CL-GD546X BIOS operates in two bus environments: VESA VL-Bus and PCI bus. The BIOS, on the CL-GD546X, is required by the hardware to fit into a 32768-byte ROM. The CL-GD546X BIOS is also architected so that features can be removed and inserted to support generating a smaller size BIOS.

6.1.2 Support Capabilities

The CL-GD546X supports multiple display adapters, bus environments, and video modes.

6.1.2.1 Multiple Display Adapter Support

Although the CL-GD546X is used primarily in systems without other display adapters, the CL-GD546X BIOS supports multiple adapter configurations (by the multi-adapter TSR BIOS).

Table 6-1. CL-GD546X Configurations for Multiple Adapters

Configuration	One CL-GD546X	Second CL-GD546X	VGA	Mono	Hercules	PCI	VESA® VL-Bus™
1	●					✓	✓
2	●	●				✓	–
3	●		●			✓	–
4	●		●	●		✓	–
5	●		●		●	✓	–
6	●			●		✓	✓
7	●				●	✓	✓
8	●	●	●	●		✓	–

6.1.2.2 System Bus Support

The CL-GD546X BIOS supports two different bus environments: PCI and VESA VL-Bus. The PCI header is implemented in the ROM image. The PCI system BIOS reads and writes PCI Configuration registers.

The main difference between the VESA VL-Bus BIOS and PCI BIOS is in configuration setup. It is necessary for the CL-GD546X BIOS to perform the Configuration register setup by configuring the Memory-Mapped registers. Instead of calling the system BIOS and requesting the location of the Base_Address_Register_0, the BAR_0 register decodes at address 2B0h. The same memory mapping that is performed under PCI is also done under VESA VL-Bus with the addition of setting the lower bit of the BAR_0 register to ‘1’. This enables the configuration registers. An OEMSI parameter determines the wakeup port. In the case of the VESA VL-Bus motherboard BIOS, the wakeup port is 3C3h. In the case of the VESA VL-Bus adapter, the wakeup port is 46E8h.

6.1.2.3 BIOS Modes Supported

The following SVGA, VESA, and Cirrus Logic modes are supported by the CL-GD546X BIOS. The BIOS provides mode switching for all of the video modes listed in Table 6-2.

Table 6-2. CL-GD546X Supported BIOS Modes

Mode	VESA® Mode No.	Cirrus Logic Mode No.	dX	dY	Text × bpp[a]	Colors	Refresh Rates	Mode Type[b]
VGA	0		40	25	40 × 25	64, 16 gray	70	V
VGA	1	0, 1	40	25	40 × 25	64, 16/8 color	70	V
VGA	2		80	25	80 × 25	64, 16 gray	70	V
VGA	3	2, 3	80	25	80 × 25	64, 16/8 color	70	V
VGA	4	4, 5	320	200		4 (256)	70	V
VGA	5		320	200		4, gray	70	V
VGA	6	6	640	200		2, gray	70	V
VGA	7	7	80	25		2, monochrome	70, 85[c]	V
VGA	D	D	320	200		16	70	V
VGA	E	E	640	200		16 planar	70	V
VGA	F	F	640	350		Monochrome	70, 85[c]	V
VGA	10	10	640	350		16, 64	70, 85[c]	V
VGA	11	11	640	480	80 × 25	2	60, 75	V
VGA	12	12	640	480	80 × 25	16 planar	60, 75	V
VGA	13	13	320	200	40 × 25	256 linear	60, 75	V
Cirrus Logic	11C[d]	7A	640	400	16	65K	70	V
VESA	100	5E	640	400	8	256	70	V, X

Table 6-2. CL-GD546X Supported BIOS Modes *(cont.)*

Mode	VESA® Mode No.	Cirrus Logic Mode No.	dX	dY	Text × bpp[a]	Colors	Refresh Rates	Mode Type[b]
VESA	101	5F	640	480	8	256	60, 72, 75, 85	V, X
VESA	111	64	640	480	16	65K	60, 72, 75, 85	X
VESA	112	71	640	480	24	16M	60, 72, 75, 85	X
Cirrus Logic	11D[d]	76	640	480	32	16M	60, 72, 75, 85	X
VESA	102	58, 6A	800	600	4	16	56, 60, 72, 75, 85	V
VESA	103	5C	800	600	8	256	56, 60, 72, 75, 85	V, X
VESA	114	65	800	600	16	65K	56, 60, 72, 75, 85	X
VESA	115	78	800	600	24	16M	56, 60, 72, 75, 85	X
Cirrus Logic	11E[d]	72	800	600	32	16M	56, 60, 72, 75, 85	X
VESA	104	5D	1024	768	4	16	43i[e], 60, 70, 75, 85	V
VESA	105	60	1024	768	8	256	43i, 60, 70, 75, 85	V, X
VESA	117	74	1024	768	16	65K	43i, 60, 70, 75, 85	X
VESA	118	79	1024	768	24	16M	43i, 60, 70, 75, 85	X
Cirrus Logic	11F[d]	73	1024	768	32	16M	43i, 60, 70, 75, 85	X
VESA	106	6C	1280	1024	4	16	43i, 60, 70, 71.2, 75	V
VESA	107	6D	1280	1024	8	256	43i, 60, 70, 71.2, 75	X
VESA	11A	75	1280	1024	16	65K	43i, 60, 70, 71.2, 75	X
Cirrus Logic	120[d]	7B	1600	1200	8	256	48i, 60	X

[a] 16 bpp is 5:6:5 (RGB); 24 bpp is 24-bpp packed pixel; 32 bpp is 24 bpp packed into a 32-bit dword.

[b] ‘V’ indicates VGA compatible. ‘X’ indicates that the frame buffer must be accessed by Linear Addressing mode.

[c] Generic fix up TSR / VxD must be run to support higher refresh rates.

[d] VBE v2.0 reported mode number.

[e] ‘i’ indicates interlaced.

6.1.3 VGA BIOS Functions

The CL-GD546X BIOS implements the standard VGA functions listed in Table 6-3.

Table 6-3. CL-GD546X BIOS Functions

BIOS	Function
VGA	INT10h (00, set video mode)
VGA	INT10h (01, set cursor type)
VGA	INT10h (02, set cursor position)
VGA	INT10h (03, get cursor position)
VGA	INT10h (04, get light pen position)
VGA	INT10h (05, set active display page)
VGA	INT10h (06, windows scroll up)
VGA	INT10h (07, windows scroll down)
VGA	INT10h (08, read character/attributes at cursor)
VGA	INT10h (09, write character/attributes at cursor)
VGA	INT10h (0A, write character at cursor)
VGA	INT10h (0B,0, set background, border color)
VGA	INT10h (0B,1, select palette set)
VGA	INT10h (0C, write dot – pixel)
VGA	INT10h (0D, read dot – pixel)
VGA	INT10h (0E, write TTY character to active page)
VGA	INT10h (0F, get video mode)
VGA	INT10h (10, 00, set Palette register)
VGA	INT10h (10, 01, set Overscan (border) register)
VGA	INT10h (10, 02, set all palette registers and Overscan register)
VGA	INT10h (10, 03, intensity/blinking)
VGA	INT10h (10, 04...6, reserved)
VGA	INT10h (10, 07, read individual palette register)
VGA	INT10h (10, 08, read Overscan (border) register)
VGA	INT10h (10, 09, read all palette registers and Overscan register)
VGA	INT10h (10, 0A...F, reserved)
VGA	INT10h (10, 10, set individual color register)
VGA	INT10h (10, 11, reserved)

Table 6-3. CL-GD546X BIOS Functions *(cont.)*

BIOS	**Function**
VGA	INT10h (10, 12, set block of color registers)
VGA	INT10h (10, 13, select color page)
VGA	INT10h (10, 14, reserved)
VGA	INT10h (10, 15, read individual color register)
VGA	INT10h (10, 16, reserved)
VGA	INT10h (10, 17, read block of color registers)
VGA	INT10h (10, 18...19, reserved)
VGA	INT10h (10, 1A, read current state of color page)
VGA	INT10h (10, 1B, sum color values to gray shades)
VGA	INT10h (11, 0, load user text font)
VGA	INT10h (11, 01, load 8 × 14 ROM text font)
VGA	INT10h (11, 02, load 8 × 8 ROM text font)
VGA	INT10h (11, 03, select block specifier)
VGA	INT10h (11, 04, load 8 × 16 ROM text font)
VGA	INT10h (11, 10, load user text font and reprogram controller)
VGA	INT10h (11, 11, load 8 × 14 ROM text font and reprogram controller)
VGA	INT10h (11, 12, load 8 × 8 ROM text font and reprogram controller)
VGA	INT10h (11, 14, load 8 × 16 ROM text font and reprogram controller)
VGA	INT10h (11, 20, set pointer of user graphics font table to INT-1F)
VGA	INT10h (11, 21, set pointer of user graphics font table to INT-43)
VGA	INT10h (11, 22, set pointer of 8 × 14 ROM graphics font table to INT-43)
VGA	INT10h (11, 23, set pointer of 8 × 8 ROM graphics font table to INT-43)
VGA	INT10h (11, 30, get font information)
VGA	INT10h (12, 10, get configuration information)
VGA	INT10h (12, 13, write string in TTY)
VGA	INT10h (12, 1A, get/set display combination code)
VGA	INT10h (12, 20, Select alternate print-screen routine)
VGA	INT10h (12, 30, select scanlines – AN mode)
VGA	INT10h (12, 31, enable/disable default palette loading)
VGA	INT10h (12, 32, enable/disable video)
VGA	INT10h (12, 33, enable/disable greyscale summing)

Table 6-3. CL-GD546X BIOS Functions *(cont.)*

BIOS	Function
VGA	INT10h (12, 34, enable/disable cursor emulation)
VGA	INT10h (12, 35, switch active display)
VGA	INT10h (12, 36, enable/disable screen refresh)
VGA	INT10h (1B, get functionality/state information)
VGA	INT10h (1C, save/restore video state)
VGA	INT10h (1D...FF, reserved)

6.1.4 VESA® BIOS Extended Functions

The VESA SVGA VBE (Video BIOS Extensions, v2.0), PM (Power Management, v1.0), and DDC2B (v1.0) functions listed in Table 6-4 are implemented in the CL-GD546X BIOS.

Table 6-4. CL-GD546X VBE and PM BIOS Functions

BIOS	Function
VBE	INT 10 (4F, 00, return Super VGA information)
VBE	INT 10 (4F, 01, return Super VGA mode information)
VBE	INT 10 (4F, 02, set Super VGA mode)
VBE	INT 10 (4F, 03, return current video mode)
VBE	INT 10 (4F, 04, save/restore Super VGA video state)
VBE	INT 10 (4F, 05, CPU video memory window control)
VBE	INT 10 (4F, 06, set/get logical scanline length)
VBE	INT 10 (4F, 07, set/get display start)
VBE	INT 10 (4F, 08, set/get DAC palette control)
VBE	INT 10 (4F, 09, set/get palette data – VBE v2.0)
VBE	INT 10 (4F, 0A, return VBE protected mode interface – VBE v2.0)
PM	INT 10 (4F, 10, 00, report VBE/PM capabilities)
PM	INT 10 (4F, 10, 01, set display power state)
PM	INT 10 (4F, 10, 02, get display power state)
DDC	INT 10 (4F, 15, 00, report VBE/DDC capabilities)
DDC	INT 10 (4F, 15, 01, read EDID)
DDC	INT 10 (4F, 15, 02, read VDIF) **NOTE:** This function is not supported.

6.1.5 Cirrus Logic Specific BIOS Functions

The Cirrus Logic INT10h functions listed in Table 6-5 are implemented in the CL-GD546X BIOS as required. INT 10 functions 12, A6, and B3, are new Cirrus Logic functions defined for the CL-GD546X.

Table 6-5. Cirrus Logic BIOS Functions

BIOS	Functions
CIRRUS	INT10h (12, 80, inquire VGA type)
CIRRUS	INT10h (12, 81, inquire BIOS version number)
CIRRUS	INT10h (12, 82, inquire design revision code)
CIRRUS	INT10h (12, 85, return installed memory)
CIRRUS	INT10h (12, 9A, inquire user options)
CIRRUS	INT10h (12, A0, query video mode availability)
CIRRUS	INT10h (12, A1, read monitor type and ID)
CIRRUS	INT10h (12, A2, set monitor type – horizontal)
CIRRUS	INT10h (12, A4, set monitor type – vertical)
CIRRUS	INT10h (12, A5, generic fix up)
CIRRUS	INT10h (12, A6, set/get physical address)
CIRRUS	INT10h (12, A7, set/get Memory-mapped register)
CIRRUS	INT10h (12, B3, Enable_Tiled_Mode)
CIRRUS	INT10h (12, B5, get FIFO/format)

6.1.6 Mode Switching

The CL-GD546X BIOS provides the primary mechanism for switching video modes on the CL-GD546X. VGA, VESA/VBE, and Cirrus Mode numbers are supported using INT-10h (00, Set Video Mode) and INT-10 (4F, 02, Set Super VGA Mode).

Switching Video modes configures the RDRAM memory in Linear mode. An additional Cirrus Logic function call puts the memory into tiled format. The reason for this two-step mode switch is that DOS/VBE applications that use the A000 aperture expect to see memory in a linear format and will produce wrong results when the memory is tiled. Standard VESA/VBE and SVGA applications typically use the memory in the default non-tiled format. High-performance drivers require that memory be switched into a tiled format: INT10h (12, B3, Enable_Tiled_Mode).

6.1.7 Register Mapping

The PCI BIOS can map the CL-GD546X registers above the first Mbyte, precluding the BIOS from accessing them during a mode change. The BIOS presumes that the registers are mapped outside the first Mbyte and moves them into the first Mbyte during mode changes. After mode change, the BIOS restores the registers to their previous location. The registers are moved to one of three base locations – A000h, B000h, or B800h – using the algorithm in the following sample code.

```
{
Save original register address;

if (Mode 0-7 switch requested) then
                                      move registers to A0000h;
else if (CGA adapter is present) then
                                      move registers to B0000h;
else
                                      move registers to B8000h;
endif;
Complete Mode change;
Restore registers to original address space;
}
```

6.2 SCRATCHPAD Register Usage

This section documents the data structures contained in the four 8-bit SCRATCHPAD registers listed in Table 6-6 and is subject to change without notice.

Table 6-6. SCRATCHPAD Registers

SR09	7	6	5	4	3	2	1	0
	1024 × 768 refresh		800 × 600 refresh			Resolution (scanlines)		
	000 = 43i Hz 001 = 60 Hz 010 = 70 Hz 100 = 75 Hz 101 = 85 Hz		000 = 56 Hz 001 = 60 Hz 010 = 72 Hz 011 = 75 Hz 100 = 85 Hz			000 = 480 001 = 600 010 = 768 011 = 1024 100 - 1200		

SR0A	7	6	5	4	3	2	1	0
	9BE	1600 × 1200 refresh			1280 × 1024 refresh			
	0 = not installed 1 = installed	000 = 48i Hz 001 = 60 Hz 010 = 65 Hz 011 = 70 Hz 100 = 75 Hz 101 = 80 Hz 110 = 85 Hz			000 = 43i Hz 001 = 60 Hz 010 = 71.2 Hz 011 = 75 Hz 100 = 85 Hz			1024 × 768 refresh

SR14	7	6	5	4	3	2	1	0
	640 × 480 refresh		Pixel depth			Memory size		
	00 = 60 Hz 01 = 72 Hz 10 = 75 Hz 11 = 85 Hz		000 = 4 bpp 001 = 8 bpp 010 = 16 bpp 011 = 24, 32 bpp			000 = 1 Mbyte 001 = 2 Mbyte 010 = 3 Mbyte 011 = 4 Mbyte 100 = 5 Mbyte 101 = 6 Mbyte 110 = 7 Mbyte 111 = 8 Mbyte		

SR15	7	6	5	4	3	2	1	0
		Reserved						Linear mode
	Windows 3.1 and Window 95							0 = VGA 1 = Linear

6.2.1 DDC Implementation

The CL-GD546X BIOS supports the DDC1/2B display type, as specified in paragraph 2.4.2 of the *VESA DDC Standard*, v1.0, revision 0. Also, the CL-GD546X BIOS implements a host system type DDC2B, as specified in paragraph 2.5.4 of the *VESA DDC Standard*, v1.0, revision 0. During DDC implementation, the following functions are integrated into the CL-GD546X BIOS:

- The data is transmitted/received on the DDC2 bidirectional data channel based on the I^2C protocol.
- The BIOS does *not* implement the Access bus protocol.
- The host requests EDID information and returns it to the application.
- The BIOS supports configurations 2 and 5 as specified in Appendix A of the *VESA DDC Standard*.

6.3 VGA BIOS

The CL-GD546X VGA BIOS is a high-performance firmware product optimized to take full advantage of the CL-GD546X VGA controllers. The CL-GD546X BIOS is based on proven BIOS technology, and is fully compatible with the IBM VGA BIOS INT10h interface and the VESA/VBE v2.0 interface. The BIOS is designed to provide a well-defined interface between MS-DOS, application software, and special OEM utility programs. In addition, it provides an extended set of functions to support the CL-GD546X VGA controllers.

6.3.1 Overview

The CL-GD546X VGA BIOS provides high-resolution extended video mode support, direct-color operation, high-performance adapter or system board implementation, and system BIOS integration and customization.

Extended Video Mode Support

The CL-GD546X VGA BIOS provides full support for all extended high-resolution video modes by INT10h function calls. In addition, the CL-GD546X VGA BIOS supports a variety of extended functions, such as VGA display configuration and extended VGA inquiry.

Direct-Color Operation

The CL-GD546X BIOS supports Direct-Color and True-Color video modes. These modes allow the CL-GD546X to display 32,768 colors, 65,636 colors, or 16.8 million colors at resolutions of up to 1024 × 768.

High Performance

The CL-GD546X BIOS is optimized to provide maximum performance in adapter or motherboard implementations. The CL-GD546X local bus, display memory interface, memory clock, and dot clock configurations are configurable using the VGA BIOS.

In addition, time-critical routines, such as TTY output and scroll, are designed to provide maximum throughput in both text and graphics modes.

System Integration

The CL-GD546X VGA BIOS is easily integrated for an adapter or motherboard design. The 32-Kbyte BIOS is provided for both the C000 and E000 address segments. To save space on the system board, the CL-GD546X VGA BIOS is incorporated into the system BIOS ROM at either the C000 or E000 address.

The BIOS does not require DIP switches or external hardware for configuration. A well defined interface to the CL-GD546X BIOS configuration is available for system BIOS or OEM setup routines.

Customization

The default CL-GD546X BIOS is designed for use – without modification – in almost all environments. However, the CL-GD546X BIOS can be easily customized for a specific system environment. Modifications are accomplished with the Cirrus Logic OEMSI utility program; such modifications do not require the CL-GD546X VGA BIOS source code. Many of the BIOS parameters and features can be modified, including:

- Sign-on message
- Display type configuration
- Video mode parameter tables
- Font tables

Compatibility

The CL-GD546X BIOS is fully compatible with the IBM VGA BIOS and supports BIOS-level compatibility for an adapter card, or integrated VGA on the system board. In addition, the CL-GD546X BIOS fully complies with the video modes and specifications issued by VESA.

6.3.2 BIOS Configurations

The CL-GD546X VGA BIOS is provided in three formats as shown in Table 6-7.

Table 6-7. BIOS Formats

BIOS	Wakeup Port	Location	Size (Kbytes)
PCI	–	C000	32
		E000	64
VESA VL-Bus motherboard	3c3h	C000	32
		E000	64
VESA VL-Bus adapter	46e8h	C000	32
		E000	64

The following steps are performed by the VGA BIOS at power-up initialization for the segment C000, adapter-based VGA BIOS. The VGA BIOS:

1) Checks if VGA BIOS vector INT10h is already installed. If this vector is installed, the VGA BIOS calls the INT10h functions to disable the existing VGA card by putting it to sleep.

2) Determines if it is a VESA VL-Bus BIOS, then:

  a) writes I/O port 280h with A0001h (enables Configuration register in memory I/O space)

  b) writes A000:317 with F6 to place frame buffer in memory

  c) writes A000:304 with 3 (command = 3) to enable I/O and MMIO

  d) writes A000:3FC with 1003401h to set up VS_Control

  e) disables the VGA adapter by a write of value of 0x16 to I/O port 46E8h

  f) programs I/O port 102h with data ‘01’ to enable video subsystem

  g) writes a value of 0Eh to 46E8h to enable I/O and memory addressing

  h) writes I/O port 4AE8h to disable 8514/A

3) Disables VGA video by programming SR1[5] to ‘1’.

4) Initializes Video vectors INT10h and INT42h.

5) Initializes the CL-GD546X Extension registers.

6) Checks for coresident MDA video adapter; if MDA is present:

  a) initializes coresident bits

  b) sets up the MDA adapter

7) Checks for CGA; if present:

  a) initializes coresident bits

  b) sets VGA to monochrome

  c) enables CGA

8) Initializes Rambus memory devices.

9) Tests video memory.

10) Initializes text mode 3.

11) Prints error messages if any POST error flags are set.

The following steps below are performed by the VGA BIOS at power-up initialization for segment E000, adapter-based VGA BIOS. The VGA BIOS:

1) Checks if VGA BIOS vector INT10h is already installed. If this vector is installed, it calls the INT10h functions to disable the existing VGA card by putting it to sleep.

2) Determines if it is a VESA VL-Bus BIOS, then:

  a) writes I/O port 280h with A0001h to enable the Configuration register in memory I/O space

  b) writes A000:317 with F6 to place the frame buffer in memory

  c) writes A000:304 with 3 (command = 3) to enable I/O and MMIO

  d) writes A000:3FC with 1003401h to set up VS_Control

  e) disables the VGA adapter by a write of value 0x16 to I/O port 46E8h

  f) programs I/O port 102h with data ‘01’ to enable video subsystem

   - **g)** writes a value of 0Eh to 46E8h to enable I/O and memory addressing
   - **h)** writes I/O port 4AE8h to disable 8514/A
3. Disables VGA video by programming SR1[5] (Sequencer Clocking Mode register) to ‘1’.
4. Initializes video vectors INT10h and INT 42H.
5. Initializes the CL-GD546X Extension registers.
6. Checks for coresident MDA video adapter; if MDA is present:
   - **a)** initializes coresident bits
   - **b)** sets up the MDA adapter
7. Checks for CGA; if present:
   - **a)** initializes coresident bits
   - **b)** sets VGA to monochrome
   - **c)** enables CGA
8. Initializes Rambus memory devices.
9. Tests video memory.
10. Initializes text mode 3.
11. Prints error messages, if any POST error flags are set.
12. Checks to see if a VGA adapter is also present in the system; if so, disables the motherboard VGA controller.

6.3.3 Video BIOS Interrupt Vectors

The interrupt vectors that must be initialized by DOS (including the planar and video BIOS) are listed in Table 6-15 on page 6-58. Of these, the vectors (at locations 0:0040, 0:0074, 0000:007C, 0000:0108, 0000:010C, and 0000:01B4) corresponding to vectors 10, 1D, 1F, 42, 43, 6D are managed by the video BIOS.

Vector 10h, 6Dh — Video Services (Vector Locations, 0000:0040H and 0000:01B4)

The CL-GD546X BIOS functions are accessed using INT10h. Application programs place a function code in AH, and if required, places calling parameters in other registers, then executes an INT10h instruction. When the BIOS gains control, the appropriate code is executed to perform the function. Parameter values may be left in processor registers to be returned to the calling program upon exit from the interrupt routine.

The functions that are supported by the CL-GD546X BIOS, allow the calling program to set the current mode, manipulate the cursor, place characters and individual pixels on the display, scroll the screen, and load character fonts and color palette values. These functions are described in following sections.

Vector 1Dh — 6845 Initialization Data (Vector Location 0000:0074H)

This vector points to the parameter table 6845.

Vector 1Fh — CGA Character Set (Vector Location 0000:007CH)

This pointer is used for the table of the upper 128 characters in CGA modes 4, 5, and 6. The INT43h vector is used for the lower 128 characters for these modes.

Vector 42h — Old Video Services Pointer (Vector Location 0000:0108H)

This location used to be the INT10h vector for planar BIOS video services. When the EGA/VGA is installed, BIOS routines reload this address with a pointer to the planar INT10h video service routine entry point.

Vector 43h — Graphics Character Table (Vector Location 0000:010CH)

BIOS routines use this vector to point to a table of bitmaps that are used when graphics characters are displayable. This table is used for the lower 128 characters in Video modes 4, 5, and 6. This table is used for 256 characters in all additional graphics modes (both IBM standard and Cirrus Logic extensions).

The INT10h calls constitute the bulk of the services provided by the video BIOS and are described in detail later. They are listed along with the function and subfunction that define the particular service required. Note that some INT10h services are introduced with the VGA and are not available on the earlier EGA. The services are divided up into functional groupings.

6.3.4 INT10h: BIOS Video Function Contents

Function	Subfunction	Description	Adapter
00h		Set video mode	EGA, VGA
01h		Set cursor type	EGA, VGA
02h		Set cursor position	EGA, VGA
03h		Get cursor position	EGA, VGA
04h		Get light pen position	EGA, VGA
05h		Select active display page	EGA, VGA
06h		Window scroll-up	EGA, VGA
07h		Window scroll-down	EGA, VGA
08h		Read character/attribute at cursor position	EGA, VGA
09h		Write character/attribute at cursor position	EGA, VGA
0Ah		Write character at cursor position	EGA, VGA
0Bh	00h	Set background/border color	EGA, VGA
	01h	Select the palette set	EGA, VGA
0Ch		Write dot (pixel)	EGA, VGA
0Dh		Read dot (pixel)	EGA, VGA
0Eh		Write teletype character to active page	EGA, VGA
0Fh		Get video mode	EGA, VGA
10h		Palette manipulations	EGA, VGA
	00h	Set individual palette register (internal palette register)	
	01h	Set OverScan (border) register	
	02h	Set all palette registers and OverScan register	
	03h	Intensity/blinking	
	04h–06h	Reserved	
	07h	Read individual palette register (internal palette register)	
	08h	Read OverScan (border) register	
	09h	Read all palette registers and OverScan register	
	0Ah–0Fh	Reserved	
	10h	Set individual color register (RAMDAC/external palette register)	
	11h	Reserved	

6.3.4 INT10h: BIOS Video Function Contents *(cont.)*

Function	Subfunction	Description	Adapter
	12h	Set block of color registers	
	13h	Select color page (not valid in mode 13h)	
	14h	Reserved	
	15h	Read individual color register (RAMDAC/external palette register)	
	16h	Reserved	
	17h	Read block of color registers	
	18h–19h	Reserved	
	1Ah	Read current state of color page	
	1Bh	Sum color values to gray shades	
11h		Character generator	EGA, VGA
	00h	Load user text font	
	01h	Load 8 × 14 ROM text font	
	02h	Load 8 × 8 ROM text font	
	03h	Select block specifier	
	04h	Load 8 × 16 ROM text font	VGA
	10h	Load user text font and reprogram controller	
	11h	Load 8 × 14 ROM text font and reprogram controller	
	12h	Load 8 × 8 ROM text font and reprogram controller	
	14h	Load 8 × 16 ROM text font and reprogram controller	VGA
	20h	Set pointer of user’s graphics font table to INT1Fh	
	21h	Set pointer of user’s graphics font table to INT43h	
	22h	Set pointer of 8 × 14 ROM graphics font table to INT43h	
	23h	Set pointer of 8 × 8 ROM graphics font table to INT43h	
	24h	Set pointer of 8 × 16 ROM graphics font table to INT43h	VGA
	30h	Get font information	

6.3.4 INT10h: BIOS Video Function Contents *(cont.)*

Function	Subfunction	Description	Adapter
12h		Alternate select	EGA, VGA
	10h	Get configuration information	
	20h	Select alternate PrintScreen routine	
	30h	Select scanlines (AN mode)	
	31h	Enable/disable default palette loading	
	32h	Enable/disable video	
	33h	Enable/disable grayscale summing	
	34h	Enable/disable cursor emulation	
	35h	Switch active display	
	36h	Enable/disable screen refresh	
	13h	Write string in teletype	EGA, VGA
	1Ah	Get/set display combination code	VGA
1Bh		Get functionality/state information	VGA
1Ch		Save/restore video state	VGA
	1Dh–FFh	Reserved	

6.3.5 Description of Functions

Function: 00h — Set Video Mode

Input:	AH = 00h
	AL = video mode (see notes below)

Output:	None

NOTES:

1) Table 6-8 specifies standard VGA video modes.

Table 6-8. Standard VGA Video Modes

Mode	Resolution	Type	Colors	Pages
00H/01H	40 × 25 (360 × 400)	Text	16	8
02H/03H	80 × 25 (640 × 400)	Text	16	8
04H/05H	320 × 200 (40 × 25)	Graphics	4	1
06H	640 × 200 (80 × 25)	Graphics	2	1
07H	80 × 25 (720 × 400)	Text	Monochrome	8
08H–0CH	Reserved	–	–	–
0DH	320 × 200 (40 × 25)	Graphics	16	8
0EH	640 × 200 (80 × 25)	Graphics	16	4
0FH	640 × 350 (80 × 25)	Graphics	Monochrome	2
10H	640 × 350 (80 × 25)	Graphics	16	2
11H	640 × 480 (80 × 25)	Graphics	2	1
12H	640 × 480 (80 × 25)	Graphics	16	1
13H	320 × 200 (40 × 25)	Graphics	256	1

2) If bit 7 of AL is set, the display buffer is not cleared. Otherwise, the display buffer is cleared during mode setting (EGA, VGA only) (clear screen).

3) No hardware cursor in graphics modes.

4) Default mode during POST: mode 3H = color monitor; mode 07H = monochrome monitor.

5) There is no difference between modes 00H and 01H, 02H and 03H, or 05H and 06H on EGA/VGA. They are only different on CGA, which supports composite video displays.

6) The default settings of each Video mode can be overridden by several supplantations in Function 12h or by the supply user's video service table; the supply user's video service table address is stored in BIOS data area 0040:A8h.

7) See Table 6-2 on page 6-3 for a list of Cirrus Logic mode numbers.

Function: 01h — Set Cursor Type

Input:	AH =	01h	
		CH =	Start scanline of cursor (0 base)
		CL =	End scanline of cursor (0 base)
Output:	None		

NOTES:

1) This function is only available in text modes. The values of cursor type are stored at [40:60].
2) Table 6-9 shows the definition of value in register CH.

Table 6-9. CH Register Value Definition

Bit	**Definition**
7:6	Reserved = 0
5	1 = No cursor display 0 = Normal blinking cursor
4:0	Start scanline (0 base)

3) Table 6-10 shows the definition of value in register CL.

Table 6-10. CL Register Value Definition

Bit	**Definition**
7	Reserved = 0
6:5	Number of character skew
4:0	End scanline (0 base)

4) Table 6-11 shows the default settings.

Table 6-11. Default Settings

Font Size	**Start**	**End**
8 × 8	6	7
8 × 14	11	12
8 × 16	13	14

5) To allow cursor displaying as the values set in the function call, turn off cursor emulation. The Cursor Emulation Flag is located in bit 0 of [40:87]. Turn it on/off with Subfunction 34h of Function 12h call.

Function: 02h — Set Cursor Position

Input:	AH =	02h	
		BH =	Display page (0 base)
		DH =	Row number of cursor location start (0 base)
		DL =	Column number of cursor location end (0 base)
Output:	None		

NOTES:

1) This function is available for both text and graphics modes.
2) If register DL is specified greater than the width of the screen in the display area, this causes the cursor to wrap to the next row. If register DH is specified greater than the height of the screen in the display area, this causes the cursor to disappear.
3) Default setting for each mode: cursor location at 0000H.
4) BIOS maintains one cursor location for each page and supports up to eight pages. These values are recorded at [40:50] and occupy eight words (one word for each location).

Function: 03h — Get Cursor Position

Input:	AH =	03h	
		BH =	Display page (0 base)
Output:	CH =	Start scanline of cursor (0 base)	
	CL =	End scanline of cursor (0 base)	
	DH =	Row number of cursor start location (0 base)	
	DL =	Column number ofcursor end location (0 base)	

NOTE: Cursor type is same for all pages. The cursor location of each page is maintained separately.

Function: 04h — Get Light Pen Position[1]

Input:	AH =	04h	
Output:	AH =	00h	Light pen inactive
	or		
	AH =	01h	Light pen active and returns following values
	BX =	Pixel column	(X coordinate in graphics modes (0 base))
	CX =	Pixel row	(Y coordinate in graphics modes above mode 06h (0 base))
	CH =	Pixel row	(Y coordinate in graphics modes 04h–06h (0 base))
	DH =	Character row	(Y coordinate in text modes (0 base))
	DL =	Character column	(X coordinate in text modes (0 base))
	AH =	01h	Light pen active and returns following values

NOTES:

1) The color of background and foreground affects the sensitivity of light pen.
2) High-resolution device affects the accuracy of light pen.

Function: 05h — Select Active Display Page

Input:	AH =	05h	
		AL =	Display page (0 base)
Output:	None		

NOTES:

1) The contents of each page are not altered by changes to other pages.
2) Please refer to the video mode table of function 00h.

---

[1] This function is no longer supported.

Function: 06h — Window Scroll Up

Input:	AH =	06h
	AL =	Number of rows to be scrolled up (0 = scroll up and clear entire window)
	BH =	Attribute to be used when inserting blank lines
	CH =	Y coordinate of top left corner of window (0 base)
	CL =	X coordinate of top left corner of window (0 base)
	DH =	Y coordinate of bottom right corner of window (0 base)
	DL =	X coordinate of bottom right corner of window (0 base)
Output:	None	

NOTES:

1) When it encounters the number of rows of window equal to the value in register AL or when AL = 0, this function clears the entire window.
2) The image outside the window is not changed and the cursor is not updated.
3) Whenever an old line at the top of window is scrolled out of window, a new blank line (with the attribute value specified in BH) is inserted from the bottom of window.
4) This function is available for both text and graphics modes.

Function: 07h – Window Scroll Down

Input:	AH =	07h
	AL =	Number of rows to be scrolled down (0 = scroll down and clear entire window)
	BH =	Attribute to be used in inserting blank lines
	CH =	Y coordinate of top left corner of window (0 base)
	CL =	X coordinate of top left corner of window (0 base)
	DH =	Y coordinate of bottom right corner of window (0 base)
	DL =	X coordinate of bottom right corner of window (0 base)
Output:	None	

NOTES:

1) This function clears entire window when it encounters the number of rows of window equal to the value in register AL or when AL = 0.
2) The image outside the window is not changed and the cursor is not updated.
3) Whenever an old line at the bottom of window is scrolled out of window, a new blank line (with the attribute value specified in BH) is inserted from the top of window.
4) This function is available for both text and graphics modes.

Function: 08h — Read Character/Attribute at Cursor Position

Input:	AH =	08h
	BH =	Display page (0 base)
Output:	AH =	Attribute (valid in text modes)
	AL =	ASCII character code

NOTES:

1) This function can read data from other valid inactive pages in multiple page modes at any time.
2) After reading a character from the screen, the cursor is not updated, and must be moved manually.
3) No control characters such as, LF, CR, BACKSPACE, and BELL are recognized.
4) In Graphics modes 04H–06H of CGA adapter, the first half of character font (Code 00H–7FH) is only maintained in system ROM. To support the second half of character font (Code 80H–FFH), the interrupt vector 1FH at 0000:007CH, must be initialized to point to the second half of character font.
5) Graphics modes only return the character code. Three characters, 00H/20H/FFH, cannot be distinguished and the function always reads them back as character code 00H.
6) When they are written with the same color in the background color in graphics modes, the character codes are read back as character code 00H.

Function: 09h — Write Character/Attribute at Cursor Position

Input:	AH =	09h
	AL =	ASCII character code
	BH =	Display page (0 base)
	or	
	BL =	Attribute (text modes)
		Display color (graphics modes)
	CX =	Repeat character count
Output:	None	

NOTES:

1) This function can write data to other valid inactive pages in multiple page modes at any time.
2) After reading a character from the screen, the cursor is not updated, and must be moved manually.
3) No control characters such as, LF, CR, BACKSPACE, and BELL are recognized.
4) Graphics modes 04H–06H of CGA adapter, the first half of character font (code 00H–7FH) is maintained in system ROM does not. To support the second half of character font (code 80H–FFH), the interrupt vector 1FH at 0000:007CH, must be initialized to point to the second half of character font.
5) In graphics modes, the color (attribute) is treated as pixel color to generate an ASCII character pattern. The color value is masked according to the number of colors in the video modes.
6) When they are written with the same color in the background color in graphics modes, the character codes are displayed as blank.

7) The characters written to the screen (specified in CX) should not extend to the next row in graphics modes or invalid results are generated.
8) If bit 7 of register BL is set, the function takes the color value XOR with the value in display memory (valid in all graphics modes except mode 13h). Use this feature in fast character/dot erasing.

Function: 0Ah — Write Character at Cursor Position

Input:	AH =	0AH
	AL =	ASCII character code
	BH =	Display page (0 base)
	or	
	BL =	Foreground color (graphics modes does not)
	CX =	Repeat character count
Output:	None	

NOTES:

1) This function can write data to other valid inactive pages in multiple page modes at any time.
2) After reading a character from the screen, the cursor is not updated, and must be moved manually.
3) No control characters such as, LF, CR, BACKSPACE, and BELL are recognized.
4) In Graphics modes 04H–06H of CGA adapter, the first half of character font (code 00H–7FH) is only maintained in system ROM. To support the second half of charNo control characters such as, LF, CR, BACKSPACE, and BELL are recognized.
5) Graphics modes 04h–06h of CGA adapter, the first half of character font (code 00H–7FH) is only maintained in system ROM. To support the second half of character font (code 80H–FFH), the interrupt vector 1FH at 0000:007CH must be initialized to point to the second font.
6) In graphics modes, the color (attribute) is treated as pixel color to generate an ASCII character pattern. The color value is masked according to the number of colors in the video modes.
7) When they are written with the color same as background color in graphics modes, the character codes are displayed as blank.
8) The characters written to screen (specified in CX) should not extend to next row in graphics modes or invalid results are generated.
9) If bit 7 of register BL is set, the function takes the color value XOR with the value in display memory (valid in all graphics modes except mode 13h). This feature is applicable in fast character/dot erasing.

6.3.6 Function: 0Bh

Subfunction: 00h — Set Background/Border Color

Input:	AH =	0Bh
	BH =	00h
	BL =	Color value (0–31: low-intensity colors = 0–15, high-intensity colors = 16–31)
		Border color for text modes (modes 00h–03h)
		Color for 640 × 200 graphics mode (mode 06h)
Output:	None	

NOTES:

1) There are several functions in function 10h that allow extensive display-color control for both text and graphics modes.

Subfunction: 01h — Select Palette Set

Input:	AH =	0Bh
	BH =	01h (valid on modes 04h and 05h, 320 × 200 only)
	BL =	0 – palette set: Background, Green, Red, Brown
	BL =	1 – palette set: Background, Cyan, Magenta, White
Output:	None	

NOTES:

1) For the CGA adapter, the palette set is defined as shown in Table 6-12.

Table 6-12. CGA Palette Set

Mode	BL	Palette Set
04h	00h	Background, Green, Red, Yellow
	01h	Background, Cyan, Violet, White
05h	00/01h	Background, Cyan, Red, White

Function: 0Ch — Write Dot (Pixel)

Input:	AH =	0CH
	AL =	Color value for pixel (bit 7 is XOR flag)
	BH =	Display page (0 base)
	CX =	X coordinate, column number (0 base)
	DX =	Y coordinate, row number (0 base)
Output:	None	

NOTES:

1) For coordinate range, refer to the resolution field of video mode in Table 6-8 on page 6-19 in function 00h.
2) If bit 7 of AL register is set, it causes the requesting color value XOR'ed with memory color value.

Function: 0Dh — Read Dot (Pixel)

Input:	AH =	0DH
	BH =	Display page (0 base)
	CX =	X coordinate, column number (0 base)
	DX =	Y coordinate, row number (0 base)
Output:	AL =	Dot (Pixel) color

NOTES:

1) For coordinate range, please refer to the resolution field of video mode in Table 6-8 on page 6-19 in Function 00h.

Function: 0Eh — Write Character to Active RAM in Teletype Mode

Input:	AH =	0EH
	AL =	ASCII character
	BL =	Foreground color in graphics modes
Output:	None	

NOTES:

1) Control characters such as LF, CR, Backspace, and BELL are recognized.
   The ASCII codes are: LF = 0AH, CR = 0DH, Backspace = 08H, and BELL = 07H.
2) Line wrapping and screen scrolling are supported.
3) After writing a character to the screen, the cursor is moved to next position.
4) If PC BIOS Version is 10/19/81 or earlier, set register BH to '0'.
5) If bit 7 of the register is set in graphics modes, the color value in register BL is XOR'ed with the content of display memory.

6) In text modes, the attribute of a character written to a new line is taken from the attribute of the last character in previous line. To control the attribute for a character, use function 09h with blank character/attribute before the function issued.

Function: 0Fh — Get Video State

Input:	AH =	0Fh
Output:	AH =	Number of displayable columns (1 base)
	AL =	Current video mode
	BH =	Current active page (0 base)

6.3.7 Function: 10h

Subfunction: 00h — Set Individual Palette Register (Internal Palette Register)

Input:	AH =	10h
	AL =	00h (Subfunction)
	BH =	Color value
	BL =	Palette register (0–0Fh)
Output:	None	

NOTES:

1) Color value in the Internal Palette register serves as a pointer to one of the External registers (RAMDAC).
2) In mode 13h, the color is not changed by this function.

Subfunction: 01h — Set (Border) Register

Input:	AH =	10h
	AL =	01h (Subfunction)
	BH =	Color value (00h–FFh)
Output:	None	

NOTES:

1) The border color is driven by one of the 256 External registers.

Subfunction: 02h — Set All Palette Registers and OverScan Register

Input:	AH =	10h
	AL =	02h (Subfunction)
	ES: DX =	Point to a 17-byte buffer
Output:	None	

NOTES:

1) The first 16 bytes in the buffer store the values for 16 Internal Palette registers. The last byte is the value for Overscan register.
2) The display color is not affected, except for the Overscan register on mode 13h.

Subfunction: 03h — Toggle Intensify/Blinking Bit

Input:	AH =	10h
	AL =	03h (Subfunction)
	BL =	00h – intensify
		01h – blinking
Output:	None	

NOTES:

1) Bit 7 of the attribute byte is interpreted according to the setting state by this function. This function can provide 16 background colors (in intensify state) of 16-color text modes.
2) This function also supports Monochrome modes (07h, 0Fh).

Subfunction: 4–6h — Reserved

Subfunction: 07h — Read Individual Palette Register (Internal Palette Register)

Input:	AH =	10h
	AL =	07h (Subfunction)
	BL =	Palette register (0–0Fh)
Output:	BH =	Color value

NOTES:

1) Color value in the Internal Palette register is served as a pointer to one of the External registers (RAM-DAC).

Subfunction: 08h — Read Overscan (Border) Register

Input:	AH =	10h
	AL =	08h (Subfunction)
Output:	BH =	Color value

NOTES:

1) The border color is from 00h–FFh.

Subfunction: 09h — Read All Palette Registers and OverScan Register

Input:	AH =	10h
	AL =	09h (Subfunction)
	ES: DX =	Point to a 17-byte buffer (The first 16 bytes for returning values from 16 palette registers, respectively, and the last byte for OverScan register).
Output:	ES: DX =	Point to the same buffer that is provided from the entry of function call.

Subfunction: 0A-0Fh — Reserved

Subfunction: 10h — Set Individual Color Register (RAMDAC/External Palette Registers)

Input:	AH =	10h
	AL =	10h (Subfunction)
	BX =	Color register (00h–FFh)
	DH =	Red color
	CH =	Green color
	CL =	Blue color
Output:	None	

NOTES:

1) Each color has only six significant bits. Three colors – RED, GREEN, and BLUE – are formed into a 18-bit datum stored in the Color register.
2) The maximum number of displayable colors is 256 out of 256-Kbyte colors (two exponential 18).
3) In standard VGA, mode 13h uses all 256-Color registers to display colors.
4) Whenever function 00h (Set Video mode) is called, the BIOS loads default values into the Color registers. This is only true when the disable flag, of default palette loading, is not set (refer to subfunction 31h of function 12h).
5) With the gray-summing flag set, the value returned for all three colors is the grayshade value.

Subfunction: 11h — Reserved

Subfunction: 12h — Set Block of Color Registers

Input:	AH =	10h
	AL =	12h (Subfunction)
	BX =	Start Color register (00h–FFh)
	CX =	Number of color registers to set
	ES: DX =	Point to table of color values (each table entry is in RGB format)
Output:	None	

NOTES:

1) Each color only has six significant bits. Three colors – RED, GREEN, and BLUE – are formed into a 18-bit datum stored in the Color register.
2) The number of maximum displayable colors is 256 out of 256-Kbyte colors (2 exponential 18).
3) In standard VGA, mode 13h uses all 256-Color registers to display colors.
4) Whenever function 00h (Set Video mode) is called, BIOS loads default values into Color registers. This is only true when the disable flag of default palette loading is not set (refer to subfunction 31h of Function 12h).
5) With the gray-summing flag set, the value returned for all three colors is the grayshade value.

Subfunction: 13h — Select Color Page (Not valid on Mode 13h)

Input:	AH =	10h
	AL =	13h (Subfunction)
	BL =	00h (select paging mode)
	BL =	01h (select color page)
	When BL = 00h:	
	BH =	00h (select 4 pages of 64-color register page)
		01h (select 16 pages of 16-color register page)
	When BL = 01h:	
	BH =	Color page number (0 base)
Output:	None	

NOTES:

1) Except for 256-color modes, all video modes are supported by the function.
2) This function treats 256-color registers as sets of 16- or 64-color registers. It can quickly display different colors by switching among color sets (pages).
3) After the video mode is set, the default setting is page 0 of 64-color Page mode. Normally, function 00h (Set Video mode) loads the default colors of the first 64-color registers (page 00h). These are loaded for all standard VGA modes, except mode 13h (248 registers loading).

Subfunction: 14h — Reserved

Subfunction: 15h — Read Individual Color Register (RAMDAC/External Palette Registers)

Input:	AH =	10h
	AL =	15h (Subfunction)
	BX =	Color register (00h–FFh)
Output:	DH =	Red color
	CH =	Green color
	CL =	Blue color

NOTES:

1) The maximum number of displayable colors is 256 out of 256-Kbyte colors (2 exponential 18).
2) In standard VGA, mode 13h uses all 256-Color registers to display colors.
3) With the gray-summing flag set, the value returned for all three color elements of the color register is the grayshade value.

Subfunction: 16h — Reserved

Subfunction: 17h — Read Block of Color Registers

Input:	AH =	10h
	AL =	17h (Subfunction)
	BX =	Start Color register (00h–FFh)
	CX =	Number of color registers to read
	ES: DX =	Point to user provided buffer for returned color values
Output:	ES: DX =	Point to same buffer from function call entry (buffer is treated as a color table and each entry of the table consists of three bytes in RGB format).

NOTES:

1) Each color is a 6-bit value. All three colors form a 18-bit datum.
2) The maximum colors displayable are 256 out of 256 Kbytes colors (two exponential 18).
3) In standard VGA, mode 13h uses all 256-color registers to display colors.
4) Whenever Function 00h (set Video mode) is called, BIOS loads default values into Color registers. This is only true when the disable flag of default palette loading is not set (please refer to Subfunction 31h of Function 12h).
5) With the graysumming flag set, the value returned for all three colors is the grayshade value.

Subfunction: 18-19h — Reserved

Subfunction: 1Ah — Read Current State of Color Page (Not valid on Mode 13h)

Input:	AH =	10h
	AL =	1Ah (Subfunction)
Output:	BH =	Current page (value depends on Page mode; 00h is default)
	BL =	Current Page mode
		00h = 4 pages of 64-Color registers (default);
		01h = 16 pages of 16-color registers)

NOTES:

1) All video modes except 256-color modes are supported by the function.
2) This function treats 256-Color registers as sets of 16- or 64-Color registers. It can quickly display different colors by switching among color sets (pages).
3) After the Video mode is set, default setting is page 00h of 6-color page mode.

Subfunction: 1Bh — Sum Color Values to Grayshades

Input:	AH =	10h
	AL =	1Bh (Subfunction)
	BX =	Start Color register (00h–FFh)
	CX =	Number of color registers to sum
Output:	None	

NOTES:

1) This function combines the required Color registers into grayshade values, regardless of the graysumming flag.

6.3.8 Function: 11h

Subfunction: 00h — Load User Text Font

Input:	AH =	11h
	AL =	00h (Subfunction)
	BH =	Number of bytes per character
	BL =	Block to load (00h–07h)
	CX =	Number of characters to store
	DX =	ASCII character ID of the first character in the font table (ES: BP)
	ES: BP =	Point to user-provided font table
Output:	None	

NOTES:

1) This function is only available for text modes. The value in register BH represents the height of each character. The value can be specified a maximum of 32-bytes-per-character in standard VGA specification.
2) In VGA, the character font is loaded into RAM Map 2 (0 base), which can contain up to eight fonts at any time, and is only available for the character generator. Consequently, the block value specified in register BL ranges from 00h (default) to 07h.
3) Two out of eight character fonts can be used at any time. This provides 512 simultaneously displayable characters instead of 256 characters. To display two different fonts at one time:
   - **a)** Load fonts into desired blocks.
   - **b)** Out of eight font blocks, subfunction 03h (Select Block Specifier) of function 11h is called to select two different font blocks – one for primary font and the other for secondary font.
   - **c)** Bit 3 of the attribute byte serves as the font block selector and foreground intensity for the character.
     Bit 3 = 0 – primary font selected and normal display (eight foreground colors) if subfunction 00h of function 10h is called with BX = 0712h.
     Bit 3 = 1 – secondary font selected and normal display (eight foreground colors)
     else,
     Bit 3 = 1 – secondary font selected and intensity display (16 foreground colors)
4) Default setting by the BIOS loads a font into block 0, which is used for both primary font and secondary font (256 displayable characters).
5) Since the controller is not reprogrammed and abnormal character display can occur, the font loading by subfunction 00h requires caution. For example, if a font table is loaded to override the font in a block, a double image may result – especially when the loaded font size is smaller than the one previously displayed.

Subfunction: 01h — Load 8 × 14 ROM Font

Input:	AH =	11h
	AL =	01h (Subfunction)
	BL =	Block to load (00h–07h)
Output:	None	

NOTES:

1) This subfunction actually loads an 8 × 16 font.
2) This function is only available for text modes.
3) The height of character is 14 bytes, but the height of display cell is same as default setting.
4) In VGA, the character font is loaded into RAM map 2 (0 base), which can contain up to eight fonts at any time and is only available for the character generator. Consequently, the block value specified in register BL ranges from 00h (default) to 07h.
5) Two out of eight character fonts can be used at any time. This provides 512 simultaneously displayable characters instead of 256 characters. To display two different fonts at one time:
   a) Load fonts into desired blocks.
   b) Out of eight font blocks, subfunction 03h (Select Block Specifier) of function 11h is called to select two different font blocks – one for primary font and the other for secondary font.
   c) Bit 3 of the attribute byte serves as the font block selector and foreground intensity for the character.
   Bit 3 = 0 – primary font selected and normal display (eight foreground colors) if subfunction 00h of function 10h is called with BX = 0712h.
   Bit 3 = 1 – secondary font selected and normal display (eight foreground colors)
   else,
   Bit 3 = 1 – secondary font selected and intensity display (16 foreground colors)
6) Default setting by the BIOS loads a font into block 0, which is used for both primary font and secondary font (256 displayable character).
7) Since the controller is not reprogrammed and abnormal character display can occur, the font loading by subfunction 00h requires caution. For example: If a font table is loaded to override the font in a block, a double image can result – especially if the loaded font size is smaller than the one previously displayed.

Subfunction: 02h — Load 8 × 8 ROM Font

Input:	AH =	11h
	AL =	08h (Subfunction)
	BL =	Block to load (00h–07h)
Output:	None	

NOTES:

1) This function is only available for text modes.
2) The height of the character is 8 bytes, but the height of the display cell is the same as the default setting.
3) In VGA, the character font is loaded into RAM map 2 (0 base), which can contain up to eight fonts at any time and is only available for the character generator. Consequently, the block value specified in register BL ranges from 00h (default) to 07h.

4) Two out of eight character fonts can be used at any time. This provides 512 simultaneously displayable characters instead of 256 characters. To display two different fonts at one time:

   a) Load fonts into desired blocks.

   b) Out of eight font blocks, subfunction 03h (Select Block Specifier) of function 11h is called to select two different font blocks – one for primary font and the other for secondary font.

   c) Bit 3 of the attribute byte serves as the font block selector and foreground intensity for the character.
   Bit 3 = 0 – primary font selected and normal display (eight foreground colors) if subfunction 00h of function 10h is called with BX = 0712h.
   Bit 3 = 1 – secondary font selected and normal display (eight foreground colors)
   else,
   Bit 3 = 1 – secondary font selected and intensity display (16 foreground colors)

5) Default setting by BIOS loads a font into Block 0, which is used for both primary font and secondary font (256 displayable characters).

6) Since the controller is not reprogrammed and abnormal character display can occur, the font loading by subfunction 00h requires caution. For example, if a font table is loaded to override the font in a block, a double image can result (especially if the loaded font size is smaller than the one previously displayed).

Subfunction: 03h — Select Block Specifier

Input:	AH =	11h
	AL =	03h (Subfunction)
	BL =	Selection of character generator blocks
Output:	None	

NOTES:

1) The definition of the value in register BL is shown in Table 6-13.

Table 6-13. Register BL Value

Bits	Font	Blocks
4, 1, 0	Primary font	00h–07h
5, 3, 2	Secondary font	00h-07h

2) For EGA-compatible operation, bits 1:0 are for the primary font, and bits 3:2 for the secondary font.

3) To retain eight consistent colors during 512-character display, the subfunction 00h of function 10h must be called first, in the following setting:
AX = 1000h, BX = 0712h

Subfunction: 04h — Load 8 × 16 ROM Font

Input:	AH =	11h
	AL =	04h (Subfunction)
	BL =	Block to load (00h–07h)
Output:	None	

NOTES:

1) This function is only available for text modes.
2) The height of character is 16 bytes, but the height of the display cell is the same as the default setting.
3) In VGA, the character font is loaded into RAM Map 2 (0 base), which can contain up to eight fonts at any time and is only available for the character generator. Consequently, the block value specified in register BL ranges from 00h (default) to 07h.
4) Two out of eight character fonts can be used at any time. This provides 512 simultaneously displayable characters instead of 256 characters. To display two different fonts at one time:
   - **a)** Load fonts into desired blocks.
   - **b)** Out of eight font blocks, subfunction 03h (Select Block Specifier) of function 11h is called to select two different font blocks – one for primary font and the other for secondary font.
   - **c)** Bit 3 of the attribute byte serves as the font block selector and foreground intensity for the character.
     Bit 3 = 0 – primary font selected and normal display (eight foreground colors) if subfunction 00h of function 10h is called with BX = 0712h.
     Bit 3 = 1 – secondary font selected and normal display (eight foreground colors)
     else,
     Bit 3 = 1 – secondary font selected and intensity display (16 foreground colors)
5) Default setting by BIOS loads a font into block 0, which is used for both the primary and secondary fonts (256 displayable characters).
6) Since the controller is not reprogrammed and abnormal character display can occur, the font loading by Subfunction 00h requires caution. For example: If a font table is loaded to override the font in a block, a double image can result (especially if the loaded font size is smaller than the one previously displayed).

Subfunction: 10h — Load User Text Font and Reprogram Controller

Input:	AH =	11h
	AL =	10h (Subfunction)
	BH =	Number of bytes per character
	BL =	Block to load (00h–07h)
	CX =	Number of characters to store
	DX =	ASCII character ID of the first character in the font table (ES: BP)
	ES: BP =	Point to user-provided font table
Output:	None	

NOTES:

1) This function is only available for text modes.

2) The value in register BH represents the height of each character. It can be specified a maximum of 32 bytes per character in standard VGA specification.

3) In VGA, the character font is loaded into RAM map 2 (0 base), which can contain up to eight fonts at any time, and is only available for the character generator. Consequently, the block value specified in register BL ranges from 00h (default) to 07h.

4) Two out of eight character fonts can be used at any time. This provides 512 simultaneously displayable characters instead of 256 characters. To display two different fonts at one time:

   a) Load fonts into desired blocks.

   b) Out of eight font blocks, subfunction 03h (Select Block Specifier) of function 11h is called to select two different font blocks – one for primary font and the other for secondary font.

   c) Bit 3 of the attribute byte serves as the font block selector and foreground intensity for the character.
   Bit 3 = 0 – primary font selected and normal display (eight foreground colors) if subfunction 00h of function 10h is called with BX = 0712h.
   Bit 3 = 1 – secondary font selected and normal display (eight foreground colors)
   else,
   Bit 3 = 1 – secondary font selected and intensity display (16 foreground colors)

5) Default setting by BIOS loads a font into block 0, which is used for both primary font and secondary font (256 displayable characters).

6) Subfunction 10h is almost identical to subfunction 00h, except for the following differences:

   a) Page 00h must be active.

   b) The character height (bytes-per-character) is recalculated.

   c) The number of rows (0 base) is recalculated as:
   (scanlines per screen ÷ character height) – 1.

   d) The length of the display buffer is recalculated as:
   (total number of rows × total number of columns) × 2 (1 base).

   e) The CRTC registers are reprogrammed as:

Index	Register Name	Change
09h	Maximum Scanlines	Character height – 1
0Ah	Cursor Start	Character height – 2
0Bh	Cursor End	Character height – 1
12h	Vertical Display Enable End	
	For 350 or 400 scanline modes:	(Rows per screen × Character height) – 1
	For 200 scanline modes:	[(Rows per screen × Character height) × 2] – 1
14h	Underline Location (mode 07h only)	Character height – 1

   f) It must be called immediately after a function 00h call (Set Video mode), otherwise the result is unpredictable.

Subfunction: 11h — Load 8 × 14 ROM Font and Reprogram Controller

Input:	AH =	11h
	AL =	11h (Subfunction)
	BL =	Block to load (00h–08h)
Output:	None	

NOTES:

1) This subfunction actually loads an 8 × 16 font.
2) This function is only available for text modes.
3) The character and display cells are both 14 bytes high (scanlines).
4) In VGA, the character font is loaded into RAM map 2 (0 base), which can contain up to eight fonts at any time and is only available for the character generator. Consequently, the block value specified in register BL ranges from 00h (default) to 07h.
5) Two out of eight character fonts can be used at any time. This provides 512 simultaneously displayable characters instead of 256 characters. To display two different fonts at one time:
   a) Load fonts into desired blocks.
   b) Out of eight font blocks, subfunction 03h (Select Block Specifier) of function 11h is called to select two different font blocks – one for primary font and the other for secondary font.
   c) Bit 3 of the attribute byte serves as the font block selector and foreground intensity for the character.
   Bit 3 = 0 – primary font selected and normal display (eight foreground colors) if subfunction 00h of function 10h is called with BX = 0712h.
   Bit 3 = 1 – secondary font selected and normal display (eight foreground colors)
   else,
   Bit 3 = 1 – secondary font selected and intensity display (16 foreground colors)
6) Default setting by the BIOS loads a font into block 0, which is used for both primary font and secondary font (256 displayable characters).
7) Subfunction 11h is almost identical to subfunction 01h, except for the following differences:
   a) Page 00h must be active.
   b) The character height is 14.
   c) The number of rows (0 base) is recalculated as:
   (scanlines per screen ÷ character height) − 1
   d) The length of the display buffer is recalculated as:
   (total number of rows × total number of columns) × 2 (1 base)
   e) The CRTC registers are reprogrammed as:

Index	**Register Name**	**Change**
09h	Maximum Scanlines	13 (0Dh)
0Ah	Cursor Start	13 (0Dh)
0Bh	Cursor End	13 (0Dh)
12h	Vertical Display Enable End	(Rows per screen × 14) – 1
14h	Underline Location (mode 07h only)	13 (0Dh)

   f) It must be called immediately after a function 00h call (Set Video mode) or the result is unpredictable.

Subfunction: 12h — Load 8 × 8 ROM Font and Reprogram Controller

Input:	AH =	11h
	AL =	12h (Subfunction)
	BL =	Block to load (00h–07h)
Output:	None	

NOTES:

1) This function is only available for text modes.
2) The height of character and display cell are all eight bytes (scanlines).
3) In VGA, the character font is loaded into RAM map 2 (0 base), which can contain up to eight fonts at any time and is only available for the character generator. Consequently, the block value specified in register BL ranges from 00h (default) to 07h.
4) Two out of eight character fonts can be used at any time. This provides 512 simultaneously displayable characters instead of 256 characters. To display two different fonts at one time:
   - **a)** Load fonts into desired blocks.
   - **b)** Out of eight font blocks, subfunction 03h (Select Block Specifier) of function 11h is called to select two different font blocks – one for primary font and the other for secondary font.
   - **c)** Bit 3 of the attribute byte serves as the font block selector and foreground intensity for the character.
     Bit 3 = 0 – primary font selected and normal display (eight foreground colors) if subfunction 00h of function 10h is called with BX = 0712h.
     Bit 3 = 1 – secondary font selected and normal display (eight foreground colors)
     else,
     Bit 3 = 1 – secondary font selected and intensity display (16 foreground colors)
5) Default setting by the BIOS loads a font into block 0, which is used for both primary font and secondary font (256 displayable characters).
6) Subfunction12h is almost identical to subfunction 02h, except for the following differences:
   - **a)** Page 00h must be active.
   - **b)** The character height is 8.
   - **c)** The number of rows (0 base) are recalculated as:
     (scanlines per screen ÷ character height) − 1
   - **d)** The length of the display buffer is recalculated as:
     (Total number of rows × Total number of columns) × 2 (1 base).
   - **e)** The CRTC registers are reprogrammed as:

Index	Register Name	Change
09h	Maximum Scanlines	7 (07h)
0Ah	Cursor Start	6 (06h)
0Bh	Cursor End	7 (07h)
12h	Vertical Display Enable End	(Rows per screen × 8) − 1
14h	Underline Location (mode 07h only)	7 (07h)

   - **f)** It must be called immediately after a function 00h call (Set Video mode) or the result is unpredictable.

Subfunction: 14h — Load 8 × 16 ROM Font and Reprogram Controller

Input:	AH =	11h
	AL =	14h (subfunction)
	BL =	Block to load (00h–07h)
Output:	None	

NOTES:

1) This function is only available for text modes.
2) The height of character and display cell are all 16 bytes (scanlines).
3) In VGA, the character font is loaded into RAM map 2 (0 base), which can contain up to eight fonts at any time and is only available for the character generator. Consequently, the block value specified in register BL ranges from 00h (default) to 07h.
4) Two out of eight character fonts can be used at any time. This provides 512 simultaneously displayable characters instead of 256 characters. To display two different fonts at one time:
   - **a)** Load fonts into desired blocks.
   - **b)** Out of eight font blocks, subfunction 03h (Select Block Specifier) of function 11h is called to select two different font blocks – one for primary font and the other for secondary font.
   - **c)** Bit 3 of the attribute byte serves as the font block selector and foreground intensity for the character.
     Bit 3 = 0 – primary font selected and normal display (eight foreground colors) if subfunction 00h of function 10h is called with BX = 0712h.
     Bit 3 = 1 – secondary font selected and normal display (eight foreground colors)
     else,
     Bit 3 = 1 – secondary font selected and intensity display (16 foreground colors)
5) Default setting by the BIOS loads a font into block 0, which is used for both primary font and secondary font (256 displayable characters).
6) Subfunction 14h is almost identical to subfunction 04h except for the following differences:
   - **a)** Page 00h must be active
   - **b)** The character height is 16
   - **c)** The number of rows (0 base) are recalculated as:
     (scanlines per screen ÷ character height) – 1
   - **d)** The length of the display buffer is recalculated as:
     (Total number of rows × Total number of columns) × 2 (1 base).
   - **e)** The CRTC registers are reprogrammed as follows:

Index	Register Name	Change
09h	Maximum Scanlines	15 (0Fh)
0Ah	Cursor Start	14 (0Eh)
0Bh	Cursor End	15 (0Fh)
12h	Vertical Display Enable End	(Rows per screen × 16) – 1
14h	Underline Location (mode 07h only)	15 (0Fh)

   - **f)** It must be called immediately after a function 00h call (Set Video mode) or the result is unpredictable.

Subfunction: 20h — Set Pointer of User's Graphics Font Table to INT 1Fh

Input:	AH =	11h
	AL =	20h (subfunction)
	ES: BP =	Point to user's graphics font table
Output:	None	

NOTES:

1) The value in this interrupt vector serves as a pointer to the graphics font table table for character codes (80h–FFh) in modes 04h, 05h, and 06h.
2) In the CGA adapter, the planar BIOS only provides 128 character codes (00h–7Fh). The user can supply the other half of character codes (80h–FFh), or use the GRAFTABL utility in DOS to load these character codes.
3) This function must be called immediately after setting Video mode.

Subfunction: 21h — Set Pointer of User's Graphics Font Table to INT 43h

Input:	AH =	11h
	AL =	21h (subfunction)
	BL =	Character rows specifier
		00h = Value in register DL (the number of displayable rows specified by user)
		01h = 14 (0Eh) character rows
		02h = 25 (19h) character rows
		03h = 43 (2Bh) character rows
	CX =	Bytes per character
	DL =	Number of character rows (if register BL = 00h)
	ES: BP =	Point to user's graphics font table
Output:	None	

NOTES:

1) The value in this interrupt vector serves as a pointer to the graphics font table table for character codes (00h–7fh) in modes 04h, 05h, and 06h. In all other graphics modes, the vector also handles the graphics font for character codes (00h–FFh).
2) This function should only be called immediately after setting Video mode.
3) The portion of character rows above displayable rows are not displayed unless it is scrolled up.
4) The overlapping screen can occur in video modes that use all display memory addresses, such as mode 13h.

Subfunction: 22h — Set Pointer of ROM 8 × 14 Graphics Font Table to INT 43h

Input:	AH =	11h
	AL =	22h (subfunction)
	BL =	The specifier of character rows on screen
		00h = value in register DL (the number of displayable rows specified by user)
		01h = 14 (0Eh) character rows
		02h = 25 (19h) character rows
		03h = 43 (2Bh) character rows
	DL =	Number of character rows to display (if register BL = 00h)
Output:	None	

NOTES:

1) This actually sets the pointer of an 8 × 16 font table.
2) The value in this interrupt vector serves as a pointer to the graphics font table for character codes (00h–7fh) in modes 04h, 05h, and 06h. In all other graphics modes, the vector also handles the graphics font for character codes (00h–FFh).
3) This function should only be called immediately after setting Video mode.
4) The portion of character rows above displayable rows are not displayed unless it is scrolled up.
5) The overlapping screen may occur on video modes that use all display memory addresses, such as mode 13h.

Subfunction: 23h — Set Pointer of ROM 8 × 8 Graphics Font Table to INT 43h

Input:	AH =	11h
	AL =	23h (subfunction)
	BL =	Specifier of character rows on screen
		00h = Value in register DL (the number of displayable rows specified by user)
		01h = 14 (0Eh) character rows
		02h = 25 (19h) character rows
		03h = 43 (2Bh) character rows
	DL =	Number of character rows to display (if register BL = 00h)
Output:	None	

NOTES:

1) The value in this interrupt vector serves as a pointer to the graphics font table for character codes (00h–7Fh) in modes 04h, 05h, and 06h. In all other graphics modes, the vector also handles the graphics font for character codes (00h–FFh).
2) This function should only be called immediately after setting Video mode.
3) The portion of character rows above displayable rows are not displayed unless it is scrolled up.

4) The overlapping screen can occur on video modes that use all display memory addresses, such as mode 13h.

Subfunction: 24h — Set Pointer of ROM 8 × 16 Graphics Font Table to INT 43h

Input:	AH =	11h
	AL =	24h (subfunction)
	BL =	Specifier of character rows on screen
		00h = value in register DL (the number of displayable rows specified by user)
		01h = 14 (0Eh) character rows
		02h = 25 (19h) character rows
		03h = 43 (2Bh) character rows
	DL =	Number of character rows to display (if register BL = 00h)
Output:	None	

NOTES:

1) The value in this interrupt vector serves as a pointer to the graphics font table for character codes (00h–7fh) in modes 04h, 05h, and 06h. In all other graphics modes, the vector also handles the graphics font for character codes (00h–FFh).
2) This function should only be called immediately after setting Video mode.
3) The portion of character rows above displayable rows are not displayed unless it is scrolled up.
4) The overlapping screen can occur on video modes that use all display memory addresses, such as mode 13h.

Subfunction: 30h — Get Pointer Information of Fonts

Input:	AH =	11h
	AL =	30h (subfunction)
	BH =	Pointer information of fonts
		00h = current font pointer stored in interrupt vector 1Fh
		01h = current font pointer stored in interrupt vector 43h
		02h = font pointer of ROM 8 × 16 font table
		03h = font pointer of ROM 8 × 8 font table (character codes 00h–7fh)
		04h = font pointer of ROM 8 × 8 font table (character codes 80h–FFh)
		05h = font pointer of ROM 8 × 16 alternate font table
		06h = font pointer of ROM 8 × 16 font table
		07h = font pointer of ROM 9 × 16 alternate font table

Output:	CX =	Current character height (bytes per character)
	DL =	Number of rows of current video mode (0 base)
	ES: BP =	Pointer information of desired font

6.3.9 Function: 12h

Subfunction: 10h — Get Current Video Configuration

Input:	AH =	12h
	BL =	10h (Subfunction)
Output:	BH =	00h = Color mode (3Dx)
		01h = Monochrome mode (3Bx)
	BL =	Video memory size
		00h = 64 Kbytes
		01h = 128 Kbytes
		02h = 192 Kbytes
		03h = 256 Kbytes

CH = Feature Bits	Feature Control Output Bit Setting	Input Status Register 0–(3C2h)
7:4	Reserved	
3	1	bit 6
2	1	bit 5
1	0	bit 6
0	0	bit 5

CL = Switch Settings	Description
7:4	Reserved
3	Configuration switch-4
2	Configuration switch-3
1	Configuration switch-2
0	Configuration switch-1

Subfunction: 20h — Alternate PrintScreen Handler

Input:	AH =	12h
	BL =	20h (Subfunction)
Output:	None	

NOTE: This function call replaces original PrintScreen interrupt handler (INT 05h) to support the modes with displayable rows on screen over 25.

Subfunction: 30h — Select Scanlines for Text Modes

Input:	AH =	12h
	BL =	30h (Subfunction)
	AL =	Specifier of scanlines
		00h = 200 scanlines
		01h = 350 scanlines
		02h = 400 scanlines
Output:	AL =	12h (function supported)

NOTES:

1) The selected scanlines take effect on next mode setting.
2) Mode 07h is only supported by 350/400 scanlines; modes 00h–03h are supported by three scanlines.
3) The modes – 200 scanlines – are double-scanned.

Subfunction: 31h — Enable/Disable Default Palette Loading

Input:	AH =	12h
	BL =	31h (subfunction)
	AL =	00h = enable default palette loading
		01h = disable default palette loading
Output:	AL =	12h (function supported)

NOTES:

1) This function takes effect on next mode setting.
2) All Internal/External Palette registers is affected.

Subfunction: 32h — Enable/Disable Video

Input:	AH =	12h
	BL =	32h (subfunction)
	AL =	00h = enable video
		01h = disable video
Output:	AL =	12h (function supported)

NOTE: The video subsystem does not respond to any I/O or video memory addressing.

Subfunction: 33h — Enable/Disable Summing-to-Grayshades

Input:	AH =	12h
	BL =	33h (subfunction)
	AL =	00h = enable summing-to-grayshades
		01h = disable summing-to-grayshades
Output:	AL =	12h (function supported)

NOTE: This function takes effect on a subsequent mode setting or internal/external palettes setting.

Subfunction: 34h — Enable/Disable Cursor Emulation

Input:	AH =	12h
	BL =	34h (subfunction)
	AL =	00h = enable cursor emulation
		01h = disable cursor emulation
Output:	AL =	12h (function supported)

NOTES:

1) This function takes effect on a subsequent mode setting or function 01h call (Set Cursor Type).
2) Bit 0 of address [40:87] emulation flag is affected.

Subfunction: 35h — Switch Video Display

Input:	AH =	12h
	BL =	35h (subfunction)
	AL =	00h = initial video adapter turned off (ES:DX must point to a 128-byte buffer for switching state save area)
		01h = system board video turned on
		02h = active video turned off (ES:DX must point to a buffer for switching
		state save area)
		03h = inactive video turned on (ES:DX must point to a buffer which saves
		switching state previously)
	ES:DX =	Buffer for switching state (valid when AL = 00h, 02h, or 03h)
Output:	AL =	12h (function supported)

NOTES:

1) There are several requirements that must be met before using this function:
   a) Two video subsystems coexisting: system board video and video adapter.

- **b)** Video resource conflict between two video systems.
- **c)** Video adapter is primary video; system board is secondary video.
- **d)** This function must be supported by system board video and video adapter.

2) If the first time switching from video adapter to system board video:
Call the function with register AL = 00h
Call the function with register AL = 01h
else
Call the function with register AL = 02h
Call the function with register AL = 03h

Subfunction: 36h — Enable/Disable Screen Display

Input:	AH =	12h
	BL =	36h (subfunction)
	AL =	00h = enable screen display
		01h = disable screen display
Output:	AL =	12h (function supported)

NOTE: This function can be used for fast video memory updating without losing synchronization.

6.3.10 Function: 13h — Write Teletype String

Input:	AH =	13h
	AL =	Write function specifier
		00h = write character string without updating cursor (BL = attribute)
		01h = write character string with updating cursor (BL = Attribute)
		02h = write character/attribute string without updating cursor
		03h = write character/attribute string with updating cursor
	BH =	Display page (0 base)
	BL =	Attribute (valid when AL = 00h or 01h)
	CX =	String length
	Dh =	Start Y coordinate of string displayed on screen
	DL =	Start X coordinate of string displayed on screen
	ES: BP =	Start address of string (in segment: Offset Format)
Output:	None	

NOTES:

1) The control characters LF, CR, Backspace, and BELL are recognized. The ASCII codes are: LF = 0Ah, CR = 0Dh, Backspace = 08h, BELL = 07h.

2) The string can be written to any pages regardless of active state.

3) Line wrapping and screen scrolling are supported. Screen scrolling is only supported on active page.
4) The color value in register BL XOR'ed with the contents of display memory, if bit 7 of the register is set in graphics modes.

6.3.11 Function: 1Ah

Subfunction: 00h — Get Display Combination Code (DCC)

Input:	AH =	1Ah
	AL =	00h (subfunction)
Output:	If function supported:	
	AL =	1Ah
	BH =	Alternate display code
	BL =	Active display code

NOTES:

1) The index of the current DCC entry in the DCC table is stored in address [40:8A].
2) The display combination code definitions are:

Code	Definition
00h	No display
01h	Monochrome Display Adapter (MDA)
02h	Color Display Adapter (CGA)
03h	Reserved
04h	EGA with Color Monitor (EGA)
05h	EGA with Monochrome Monitor (MEGA)
06h	Professional Graphics Adapter with Color Display (PGA)
07h	Video Graphics Array with Analog Monochrome Monitor (MVGA)
08h	Video Graphics Array with Analog Color Monitor (VGA)

Subfunction: 01h — Set Display Combination Code (DCC)

Input:	AH =	1Ah
	AL =	01h (Subfunction)
	BH =	Alternate display code
	BL =	Active display code
Output:	If function supported:	
	AL =	1Ah

NOTES:

1) The display combination code definitions are:

Code	Definition
00h	No Display
01h	Monochrome Display Adapter (MDA)
02h	Color Display Adapter (CGA)
03h	Reserved

Code	Definition
04h	EGA with Color Monitor (EGA)
05h	EGA with Monochrome Monitor (MEGA)
06h	Professional Graphics Adapter with Color Display (PGA)
07h	Video Graphics Array with Analog Monochrome Monitor (MVGA)
08h	Video Graphics Array with Analog Color Monitor (VGA)

2) The user is responsible for providing the correct DCC. There is no physical checking device.

6.3.12 Function: 1Bh — Collection of Video Information

Input:	AH =	1Bh
	BX =	00h
	ES: DI =	Pointer to 128-byte buffer
Output:	If function supported:	
	AL =	1Bh

NOTES:

1) Video information in 128-byte buffer:

Offset	Size	Definition
00h	2 words	Pointer to collection of static functionality information
04h	Byte	Current video mode
05h	Word	Number of columns (1 base)
07h	Word	Refresh buffer length (unit: byte)
09h	Word	The starting address of the refresh buffer (Offset value relates to start of video memory; default = 0000h)
0Bh	8 words	Cursor position for each page (maximum eight pages supported)
1Bh	Word	Current cursor type (High byte = start scanline; low byte = end scanline)
1Dh	Byte	Active video page
1Eh	Word	Base port address of CRTC (CRT controller) (Monochrome = 3Bxh, color = 3Dxh)
20h	Byte	Current setting of 3B8h or 3D8h (Mode Control register)
21h	Byte	Current setting of 3B9h or 3D9h
22h	Byte	Number of rows (1 base)
23h	Word	Character height (1 base; unit: scanline)
25h	Byte	Active display code
26h	Byte	Alternate display code
27h	Word	Number of displayable colors (1 base; monochrome = 0000h)
29h	Byte	Number of pages (1 base)
2Ah	Byte	Specifier of vertical resolution 00h = 200 scanlines 01h = 350 scanlines 02h = 400 scanlines 03h = 480 scanlines 04h–FFh = reserved
2Bh	Byte	Primary font block (00h–07h)
2Ch	Byte	Secondary font block (00h–07h)

Offset	Size	Definition
2Dh	Byte	Flags of video state

Bit	Definition
7:6	Reserved
5	0 = background intensity 1 = blinking (default)
4	0 = cursor emulation disable 1 = cursor emulation enable
3	0 = default palette loading enable 1 = default palette loading disable
2	0 = color monitor attached 1 = monochrome monitor attached
1	0 = summing-to-grayshades disable 1 = summing-to-grayshades enable
0	1 = all modes are active on all displays

Offset	Size	Definition
2E–30h		Reserved
31h	Byte	Specifier of total video RAM: 00h = 64 Kbytes 01h = 128 Kbytes 02h = 192 Kbytes 03h = 256 Kbytes 04h–FFh = Reserved
32h	Byte	Save pointer state information:

Bit	Definition
7:6	Reserved
5	1 = extension of display combination code active
4	1 = palette override active
3	1 = graphics font override active
2	1 = alpha font override active
1	1 = dynamic save area active
0	1 = 512-character set active

Offset	Size	Definition
33–3Fh		Reserved

2) Collection of static functionality information:

Offset	Size	Definition
00h	Byte	Available video modes if bit set:

Bit	Video Mode
7	07h
6	06h
5	05h
4	04h
3	03h
2	02h
1	01h
0	00h

Offset	Size	Definition
01h	Byte	Available video modes if bit set:

Bit	Video Mode
7	0Fh
6	0Eh
5	0Dh
4	0Ch
3	0Bh
2	0Ah
1	09h
0	08h

Offset	Size	Definition
00h	Byte	Available video modes if bit set:

Bit	Video Mode
7:4	Reserved
3	13h
2	12h
1	11h
0	10h

Offset	Size	Definition
03–06h		Reserved
07h	Byte	Number of scanlines available in text modes:
08h	Byte	Number of active character blocks available in text modes

Offset	Size	Definition
09h	Byte	Maximum number of character blocks available in text modes: (subfunction 30h, function 12h)

Bit	Scanlines (if bit = 1)
7:3	Reserved
2	400
1	350
0	200

Offset	Size	Definition
0Ah	Byte	Supported functions (#1):

Bit	Function (if bit = 1)
7	Color paging
6	Color palettes (external palettes/RAMDAC)
5	EGA palettes (internal palettes)
4	Cursor emulation
3	Default palette loading
2	Character font loading
1	Summing to grayshades
0	All modes on all displays

Offset	Size	Definition
0Bh	Byte	Supported functions (#2):

Bit	Function (if bit = 1)
7:4	Reserved
3	Set display combination code
2	Background intensity/blinking control
1	Save/restore video state
0	Reserved

Offset	Size	Definition
0C–0Dh		Reserved

Offset	Size	Definition
0Eh	Byte	Save pointer functions:

Bit	Function (if bit = 1)
7:6	Reserved
5	Extension of display combination code
4	Palette override
3	Graphics font override
2	Alpha font override
1	Dynamic save area
0	512-character set

Offset	Size	Definition
0Fh		Reserved

6.3.13 Function: 1Ch

Subfunction: 00h — Get Buffer Size for Video State

Input:

AH = 1Ch

AL = 00h (Subfunction)

CX = Requested Video State:

Bit	Video State
15:3	Reserved
2	Color registers (external palettes/RAMDAC)
1	BIOS data area
0	Hardware state

Output:

If function supported:

AL = 1Ch

BX = Blocks/buffer (unit: 64 byte/block)

NOTE: This function reports the sufficient size of buffer to save video state. To guarantee Subfunction 01h and 02h are performed successfully, call this subfunction first.

Subfunction: 01h — Saving Video State

<table>
<tr><td><b>Input:</b></td><td>AH = 1Ch
AL = 01h (subfunction)
CX = Requested video states:
<table>
<tr><th>Bit</th><th>Video State</th></tr>
<tr><td>15:3</td><td>Reserved</td></tr>
<tr><td>2</td><td>Color registers (external palettes/RAMDAC)</td></tr>
<tr><td>1</td><td>BIOS data area</td></tr>
<tr><td>0</td><td>Hardware state</td></tr>
</table>
ES: BX = Pointer points to buffer (segment: Offset format)</td></tr>
<tr><td><b>Output:</b></td><td>If function supported:
AL = 1Ch
ES: BX = State information saved in user-supplied buffer</td></tr>
</table>

Subfunction: 02h — Restore Video State

<table>
<tr><td><b>Input:</b></td><td>AH = 1Ch
AL = 02h (subfunction)
CX = Requested video states:
<table>
<tr><th>Bit</th><th>Video State</th></tr>
<tr><td>15:3</td><td>Reserved</td></tr>
<tr><td>2</td><td>Color registers (external palettes/RAMDAC)</td></tr>
<tr><td>1</td><td>BIOS data area</td></tr>
<tr><td>0</td><td>Hardware state</td></tr>
</table>
ES: BX = Pointer points to previous saved buffer (segment: Offset format)</td></tr>
<tr><td><b>Output:</b></td><td>If function supported:
AL = 1Ch</td></tr>
</table>

6.4 VGA Sleep Mode and Display Switching

The IBM VGA standard supports a Sleep mode feature to enable/disable CPU addressing of the VGA subsystem video memory and I/O ports. For integrated VGA subsystems on the motherboard, the video subsystem is enabled or disabled by programming a Video Subsystem Enable register at I/O port 3C3h. On VGA adapter cards, a control register at I/O port 46E8h is used. These two separate schemes of enabling/disabling addressing allows two VGAs (driving separate display monitors) to coexist in a system and switch active video from one display to another. The IBM standard VGA BIOS supports a set of function calls to select Sleep mode and display switching features.

Depending on the application, the CL-GD546X VGA controller can be programmed to respond at either 3C3h or 46E8h I/O port for enabling/disabling CPU addressing. This allows for full IBM VGA compatibility, whether the design is an integrated motherboard VGA or an adapter card solution.

6.4.1 Memory Map

The following tables provide background information regarding the usage of system memory, port address space, and interrupt vectors by DOS and its I/O routines (Planar and Peripheral BIOS). The areas of interest to video subsystem users and designers are highlighted in bold text.

Table 6-14. Microsoft® DOS Memory Map

FE0000–FFFFFF	128 Kbytes to ‘shadow’ system ROM BIOS
100000–FDFFFF	15 Mbytes of extended memory in protected mode only
FFFF:000F (I Mbyte)	**Planar BIOS**
F000:0000	**Expansion BIOS (motherboard video BIOS)**
E000:0000	Voice Communication BIOS/LIM EMS page map area
D000:8000	Network BIOS/LIM EMS page map area
D000:0000	LIM EMS page map area
C000:C000	Hard disk BIOS
C000:8000	EGA/VGA adapter BIOS
C000:0000	EGA display RAM
B000:C000	CGA display RAM (or HGC mode graphics RAM)
B000:8000	HGC display RAM
B000:4000	MDA/HGC display RAM

Table 6-14. Microsoft® DOS Memory Map *(cont.)*

Address	Contents
B000:0000	**EGA/VGA display RAM**
A000:0000	**Top of system RAM** COMMAND.COM (transient portion), free RAM, COMMAND.COM (resident portion), installable device drivers, file control blocks, disk buffers, DOS tables, DOS kernel (MSDOS.SYS), resident DOS device drivers (IO.SYS)
0000:0600	ROM BIOS data area
0000:0400	Interrupt vectors
0000:0000	

Table 6-15. BIOS Data Area Assignments

0040:0000	WORD		COM1 port base address
0040:0002	WORD		COM2 port base address
0040:0004	WORD		COM3 port base address
0040:0006	WORD		COM4 port base address
0040:0008	WORD		Printer 1 port base address
0040:000A	WORD		Printer 2 port base address
0040:000C	WORD		Printer 3 port base address
0040:000E	WORD		Printer 4 port base address
0040:0010	WORD	EQUIPMENT_FLAG	

Bit	**Definition**
D15, D14	No. of printer adapters
D13,D12	Reserved
D11,D10,D9	No. of RS232-C
D8	Reserved
D7,D6	No. of diskette drives
D5,D4	Identify current primary display device:

D5	**D4**	**Adapter**
0	0	EGA (or none)
0	1	CGA 40 × 25
1	0	CGA 80 × 25
1	1	MDA

Bit	**Definition**
D3,D2	Reserved
D1	Math coprocessor
D0	IPL diskette

0040:0012	BYTE		Reserved
0040:0013	WORD	USABLE_RAM	Usable memory size in Kbytes
0040:0015	BYTE		Reserved
0040:0016	BYTE		Reserved
0040:0017	WORD	KBD_CNTRL	Stores status of special keys
0040:0019	BYTE	ALT_KBD	Alternate keypad entry
0040:001A	WORD	KBD_BUF_HD	Points to head of keyboard buffer
0040:001C	WORD	KBD_BUF_TL	Points to tail of keyboard buffer
0040:001E	16 WORDS	KBD_BUFFER	Circular keyboard buffer

Table 6-15. BIOS Data Area Assignments *(cont.)*

0040:003E	BYTE		Diskette drive recalibrate status
0040:003F	BYTE		Diskette drive motor status
0040:0040	BYTE		Diskette drive motor off counter
0040:0041	BYTE		Last diskette driver operation status
0040:0042	7 BYTES		Diskette driver controller status
0040:0049	BYTE	VIDEO_MODE	Current BIOS Video mode
0040:004A	WORD	COLUMNS	Number of text columns
0040:004C	WORD	PAGE_LENGTH	Length of each page in bytes
0040:004E	WORD	START_ADDR	Start Address register value for page
0040:0050	8 WORDS	CURSOR_POS	Cursor positions for all eight pages The high byte of each word contains the character row, the low byte, the column
0040:0060	WORD	CURSOR_TYPE	Start and ending lines for text cursor. High byte has start line.
0040:0062	BYTE	ACTIVE_PAGE	Currently displayed page number
0040:0063	WORD	ADDR_CRTC	I/O port address of 6845/CRTC address register (3B4 monochrome; 3D4 color)
0040:0065	BYTE	CRT_MODE_SET	Current value for Mode Control register (3B8 MDA; 3D8 CGA). The EGA and VGA values emulate the MDA/CGA values
0040:0066	BYTE	CRT_PALETTE	Current value for the CGA color select register (3D9); emulated by EGA/VGA
0040:0067	DWORD		Pointer to MCA PS/2 reset code
0040:006B	BYTE		Reserved
0040:006C	DWORD		Timer counter
0040:0070	BYTE		Timer overflow
0040:0071	BYTE		Break key state
0040:0072	WORD		RESET flag
0040:0074	BYTE		Last hard disk drive operation status
0040:0075	BYTE		No. of hard disk drives attached
0040:0076	BYTE		PC XT hard disk drive control
0040:0077	BYTE		PC XT hard disk drive controller port
0040:0078	BYTE		Printer 1 Time-out value
0040:0079	BYTE		Printer 2 Time-out value
0040:007A	BYTE		Printer 3 Time-out value
0040:007B	BYTE		Printer 4 Time-out value
0040:007C	BYTE		COM1 Time-out value
0040:007D	BYTE		COM2 Time-out value
0040:007E	BYTE		COM3 Time-out value
0040:007F	BYTE		COM4 Time-out value
0040:0080	WORD		Keyboard Buffer Start Offset pointer
0040:0082	WORD		Keyboard Buffer End Offset pointer
0040:0084	**BYTE**	**ROWS**	**Number of text rows minus 1**
0040:0085	**WORD**	**CHAR_HEIGHT**	**Bytes-per-character**

Table 6-15. BIOS Data Area Assignments *(cont.)*

0040:0087	**BYTE**	**INFO_1**			
		Bit	**Description**		
		D7	Equals bit D7 from AL register on most recent mode select. (A one indicates display memory was not cleared by mode select.)		
		D6, D5	Display memory size (00=64K, 01=128K, 10=192K, 11=256K).		
		D4	Reserved		
		D3	A zero indicates EGA is the primary display.		
		D2	A one will force the BIOS to wait for Vertical Retrace before memory write.		
		D1	A one indicates that EGA is in Monochrome mode.		
		D0	A zero means that CGA cursor emulation is enabled. The cursor shape is modified if enhanced text is used.		
0040:0088	**BYTE**	**INFO_3**			
		D4–D7	Feature Control bits (from Feature Control register)		
		DO-D3	EGA Configuration Switch settings		
0040:0089	**BYTE**	**FLAGS**	**Miscellaneous flags**		
		D7	Alphanumeric Scanlines (with bit 4):		
			Bit 7	**Bit 4**	
			0	0	350-line mode
			0	1	400-line mode
			1	0	200-line mode
			1	1	(reserved)
		D6	1 – display switching is enabled 0 – display switching is disabled		
		D5	Reserved		
		D4	(see bit 7)		
		D3	1 – default palette loading is disabled 0 – default palette loading is enabled		
		D2	1 – using monochrome monitor 0 – using color monitor		
		D1	1 – grayscale summing is enabled 0 – grayscale summing is disabled		
		D0	1 – VGA active 0 – VGA not active		
0040:008A	BYTE		Reserved		
0040:008B	BYTE		Media control		
0040:008C	BYTE		Hard disk drive controller status		
0040:008D	BYTE		Hard disk drive error status		
0040:008E	BYTE		Hard disk drive interrupt control		
0040:008F	BYTE		Reserved		
0040:0090	BYTE		Drive 0 Media state		
0040:0091	BYTE		Drive 1 Media state		
0040:0092	BYTE		Reserved		
0040:0093	BYTE		Reserved		
0040:0094	BYTE		Drive 0 Current cylinder		
0040:0095	BYTE		Drive 1 Current cylinder		
0040:0096	BYTE		Keyboard mode State and Type flags		
0040:0097	BYTE		Keyboard LED Flags		
0040:0098	WORD		Address offset to User Wait Complete flag		
0040:009A	WORD		Segment address to User Wait Complete flag		
0040:009C	WORD		User wait count – Low word (mseconds)		

Table 6-15. BIOS Data Area Assignments *(cont.)*

0040:009E	WORD		User wait count – High word (mseconds)
0040:00A0	BYTE		Wait active flag
0040:00A1	BYTE		Reserved
0040:00A2	BYTE		Reserved
0040:00A3	BYTE		Reserved
0040:00A4	BYTE		Reserved
0040:00A5	BYTE		Reserved
0040:00A6	BYTE		Reserved
0040:00A7	BYTE		Reserved
0040:00A8	**DWORD**	**SAVE_PTR**	**Pointer to BIOS Save Pointer Table**

NOTE: The next 84 bytes, 0040:00A1 to 0040:00FF, are reserved.

Table 6-16. I/O Port Assignment for PC XT and AT Computers

Port Usage for PC XT	I/O Address	Port Usage for AT
DMA controller	000–0lF	DMA Controller
Interrupt controller	020–03F	Interrupt Controller
Timer	040–04F	Coprocessor Access, Timer
	050–05F	Timer
PPI (system configuration)	060–063	
	060–06F	Keyboard
Reserved	070–07F	Realtime Clock
DMA Page register	080–09F	DMA Page Register
NMI Mask register	0A0–0AF	
	0A0–0BF	Interrupt Controller
Reserved	0B0–0FF	
	0C0–0DF	DMA Controller
	0F0–0FF	Math Coprocessor
Unusable	100–13F	Reserved
Unusable	140–14F	Token Ring Adapter
Unusable	150–15F	Advanced Color Graphics Display
Unusable	160–16F	Advanced Monochrome Graphics Display
Unusable	170–177	Fixed-disk Adapter
Unusable	1C0–1CF	Token Ring Adapter
Unusable	1E8–1EF	Streaming Tape Drive Adapter
Unusable	lF0–lF7	Fixed-disk Adapter
Unusable	1F8–1FF	Reserved
Game I/O	200–20F	Game I/O
Expansion unit	210–217	
Multifunction card, Note 1	218–21F	Multifunction Card
Reserved	220–24F	
	278–27F	Parallel port 2
	2B0	GD546X Host Control Register for VESA VL-Bus
Clock Calendar, Note 1	2C0–2CF	Clock Calendar
	2D0–2DF	3278/79 Emulation Adapter, Clock/Calender
Serial port 4, Note 1	2E0–2E7	
Serial port 3 or 4, Note 1	2E8–2EF	
Reserved	2F0–2F7	Interrupt Sharing
Serial port 2	2F8–2FF	Serial Port 2
Prototype Card	300–31F	Prototype Card
Fixed Disk	320–32F	
	360–36F	PC Network
Parallel port 1	378–37F	Parallel Port 1
SDLC	380–38F	SDLC, Bisync 2
Bisync	3A0–3AF	Bisync 1

Table 6-16. I/O Port Assignment for PC XT and AT Computers *(cont.)*

Port Usage for PC XT	I/O Address	Port Usage for AT
MDA and printer adapter	**3B0–3BF**	**MDA, EGA/VGA and Printer Adapter**
EGA/VGA Adapter	3C0–3CF	EGA/VGA
CGA	3D0–3DF	CGA, EGA/VGA
Reserved	3E0–3E7	
Serial port 3, Note 1	3E8–3EF	
Diskette Controller	3F0–3F7	Diskette Controller
Serial port 1	3F8–3FF	Serial Port 1
	400–43F	Reserved
	440–44F	Coprocessor Access
	450–50F	Reserved
	510–52F	Multi-protocol Adapter
	550–557	Coprocessor to Main CPU Communication
	6F0–6F7	Interrupt Sharing
	910–92F	Multi-protocol Adapter
	D10–D2F	Extended Monochrome Graphics Display
	E90–E9F	PSLA
	1230-124F	1st Address Range: Multi-port Async
	2230-224F	2nd Address Range: Multi-port Async
	3230-324F	3rd Address Range: Multi-port Async
	4230-424F	4th Address Range: Multi-port Async
	46E8	**VGA Add-in Adapter Sleep Enable**

NOTE: Use of port for this function is common, but not standard.

Table 6-17. Interrupt Vector Assignments

Vector Table Entry	INT Number	Name	
0000:0000	0	Divide by zero	
0000:0004	1	Single step	
0000:0008	2	Non-maskable	
0000:000C	3	Break-point	
0000:0010	4	Overflow	
0000:0014	5	Print Screen	
0000:0018	6	Reserved	
0000:001D	7	Reserved	
0000:0020	8	Time	H/W IRQ0
0000:0024	9	Keyboard	H/W IRQ1
0000:0028	A	Network	H/W IRQ2
0000:002C	B	Communications Port 2	H/W IRQ3
0000:0030	C	Communications Port 1	H/W IRQ4
0000:0034	D	Hard Disk	H/W IRQ5
0000:0038	E	Diskette	H/W IRQ6
0000:003C	F	Printer	H/W IRQ7
0000:0040	**10**	**EGA/VGA BIOS Video Services**	
0000:0044	11	Equipment Check	
0000:0048	12	Determine Memory Size	
0000:004C	13	Diskette/Disk	
0000:0050	14	Communications	
0000:0054	15	Cassette (see Notes)	
0000:0058	16	Keyboard	
0000:005C	17	Printer	
0000:0060	18	Resident BASIC	
0000:0064	19	Bootstrap	
0000:0068	1A	Time of Day	
0000:006C	1B	Keyboard Break	

Table 6-17. Interrupt Vector Assignments *(cont.)*

0000:0070	1C	Timer Tick
0000:0074	**1D**	**Video Initialization**
0000:0078	1E	Diskette Parameters
0000:007C	**1F**	**Optional Pointer to Upper 128 CGA 8 × 8 Characters**
0000:0080	20	Program Terminate
0000:0084	21	Function Request
0000:0088	22	Terminate Process Exit Address
0000:008C	23	Control-C Handler Address
0000:0090	24	Critical Error Handler Address
0000:0094	25	Absolute Disk Read
0000:0098	26	Absolute Disk Write
0000:009C	27	Terminate But Stay Resident
0000:00AA-00B8	28-2E	Reserved
0000:00BC	2F	Print Spool Control
0000:00C0-00FC	30-3F	Reserved
0000:0108	**42**	**Old BIOS Video Services**
0000:010C	**43**	**Pointer to CGA 8 × 8 Character Set**

NOTES:

1) The INT 15 interrupt service handler has additional responsibility in systems with an E000 segment video BIOS; besides cassette service, it also manages video subsystem services.
2) The complete list of interrupt numbers ascends to FFH; each vector is a doubleword so the pointer for INT xh is stored at absolute location 4xh.

6.5 VESA® BIOS Extensions

This section discusses the CL-GD546X VESA BIOS extensions that are VESA/VBE 2.0 compliant. For more information about the VESA standard, contact VESA at:

Video Electronics Standards Association
2150 N. First Street, Suite 440
San Jose, CA 95131-2020
TEL: (408) 435-0333 FAX:(408) 435-8225

6.5.1 VBE Mode Numbers

Standard VGA mode numbers are 7-bits wide and range from 00h–13h. OEMs have defined extended display modes ranging from 14h–7Fh. Values from 80h–FFh cannot be used, since VGA BIOS function 00h (Set Video mode) interprets bit 7 as a flag to clear or preserve display memory.

Due to the limitations of 7-bit mode numbers, the optional VBE mode numbers are 14-bits wide. To initialize a VBE mode, the mode number is passed in the BX register to VBE function 02h (Set VBE mode).

The format of VBE mode numbers is:

D0-D8 = Mode number
If D8 == 0; not a VESA-defined mode
If D8 == 1; this is a VESA-defined mode
D9-D13 = Reserved by VESA for future expansion (= 0)
D14 = Linear/flat frame buffer select
If D14 == 0; use VGA frame buffer
If D14 == 1; use linear/flat frame buffer
D15 = Preserve display memory select
If D15 == 0; clear display memory
If D14 == 1; preserve display memory

Thus, VBE mode numbers begin at 100h. This mode-numbering scheme implements standard 7-bit mode numbers for OEM-defined modes. Standard VGA modes may be initialized through VBE function 02h (Set VBE mode) simply by placing the mode number in BL and clearing the upper byte (BH). Seven-bit OEM-defined display modes can be initialized in the same way.

NOTE: Modes can only be set if the mode exists in the Video Mode List returned in function 0.

To date, VESA has defined one special 7-bit mode number, 6Ah, for the 800 × 600, 16-color, 4-plane Graphics mode. The corresponding 15-bit mode number for this mode is 102h. The following VBE mode numbers have been defined:

Table 6-18. VESA®/VBE Mode Numbers

GRAPHICS				TEXT			
15-bit Mode No.	**7-bit Mode No.**	**Resolution**	**Number of Colors**	**15-bit Mode No.**	**7-bit Mode No.**	**Columns**	**Rows**
mode	mode			mode	mode		
number	number		number	number			
100h	–	640 × 400	256	108h	–	80	60
101h	–	640 × 480	256	109h	–	132	25
102h	6Ah	800 × 600	16	10Ah	–	132	43
103h	–	800 × 600	256	10Bh	–	132	50
104h	–	1024 × 768	16	10Ch	–	132	60
105h	–	1024 × 768	256				
106h	–	1280 × 1024	16				
107h	–	1280 × 1024	256				
10Dh	–	320 × 200	32K (1:5:5:5)				
10Eh	–	320 × 200	64K (5:6:5)				
10Fh	–	320 × 200	16.8M (8:8:8)				

Table 6-18. VESA®/VBE Mode Numbers *(cont.)*

GRAPHICS				TEXT			
15-bit Mode No.	**7-bit Mode No.**	**Resolution**	**Number of Colors**	**15-bit Mode No.**	**7-bit Mode No.**	**Columns**	**Rows**
110h	–	640 × 480	32K (1:5:5:5)				
111h	–	640 × 480	64K (5:6:5)				
112h	–	640 × 480	16.8M (8:8:8)				
113h	–	800 × 600	32K (1:5:5:5)				
114h	–	800 × 600	64K (5:6:5)				
115h	–	800 × 600	16.8M (8:8:8)				
116h	–	1024 × 768	32K (1:5:5:5)				
117h	–	1024 × 768	64K (5:6:5)				
118h	–	1024 × 768	16.8M (8:8:8)				
119h	–	1280 × 1024	32K (1:5:5:5)				
11Ah	–	1280 × 1024	64K (5:6:5)				
11Bh	–	1280 × 1024	16.8M (8:8:8)				
81FFh		Special mode, see Notes					

NOTES:

1) Starting with VBE v2.0, VESA no longer defines new VESA mode numbers and it is no longer mandatory to support these old mode numbers. However, it is highly recommended that BIOS implementations continue to support these mode numbers for compatibility with older software.

2) Mode 81FFh is a special mode designed to preserve the current memory contents and give access to the entire video memory. This mode is especially useful for saving the entire video memory contents before going into a state that could lose the contents. For example, set this mode to gain access to all video memory to save it before going into a volatile power down state. This mode is required because the entire video memory contents are not always accessible in every mode. It is recommended that this mode be packed pixel in format, and a ModeInfoBlock must be defined for it. Look in the ModeInfoBlock to determine if paging is required and how paging is supported (if supported). Also note that there are no implied resolutions or timings associated with this mode.

3) Future display resolutions is defined by VESA display vendors. The color depths are not specified and new mode numbers are not assigned for these resolutions. For example, if the VESA display vendors define 1600 × 1200 as a VESA resolution, application developers should target their display resolution for 1600 × 1200 rather than choosing an arbitrary resolution like 1550 × 1190. The VBE implementation should be queried to get the available resolutions and color depths, and the application should be flexible enough to work with this list.

6.5.2 VBE Functions

This section describes each of the functions defined by the VBE standard. VBE functions are called using the INT10h interrupt vector, passing arguments in the 80x86 registers. The INT10h interrupt handler first determines if a VBE function was requested, and if so, processes that request. Otherwise, control is passed to the standard VGA BIOS for completion.

All VBE functions are called with the AH register set to 4Fh to distinguish them from the standard VGA BIOS functions. The AL register is used to indicate which VBE function to perform. For supplemental or extended functionality, the BL register is used when appropriate to indicate a specific subfunction.

Functions 00h–0Fh have been reserved for Standard VBE function numbers; Functions 10h–FFh are reserved for VBE Supplemental Specifications.

In addition to the INT10h interface, a Protected Mode interface is available and is described in the following sections.

6.5.3 VBE Return Status

The AX register indicates the completion upon return from VBE functions. If VBE support for the specified function is available, the 4Fh value passed in the AH register entry is returned in the AL register. If the VBE function completes successfully, 00h is returned in the AH register. Otherwise, the AH register is set to indicate the nature of the failure.

VBE Return Status

AL ==	4Fh:	Function supported.
AL !=	4Fh:	Function not supported.
AH ==	00h:	Function call successful.
AH ==	01h:	Function call failed.
AH ==	02h:	Function not supported in the current hardware configuration.
AH ==	03h:	Function call invalid in current video mode.

NOTE: Applications should consider any non-zero value in the AH register a general failure condition as later versions of the VBE may define additional error codes.

6.5.4 Protected Mode Considerations

VBE services can only be called directly from 32-bit Protected mode.

For 32-bit Protected mode, two selector/segment descriptors for 32-bit code and the data segment are required; these are allocated and initialized by the caller. The segment limit fields are set to 64K. These selectors can be in either the GDT or LDT, but must be valid whenever the VBE is called in Protected mode. The caller must supply a stack large enough for use by VBE, and by potential interrupt handlers. Since the VBE does not switch stacks when interrupts are enabled – including NMI interrupts – the caller's stack is active if or when interrupts are enabled in the VBE routine. The 32-bit VBE interface requires a 32-bit stack.

When the VBE services are called, the current I/O permission bitmap must allow access to the I/O ports required by the VBE. This can be found in the Sub-Table (Ports and Memory) returned by VBE function 0Ah.

NOTE: It is the responsibility of the calling application to ensure there are appropriate I/O and memory privileges, appropriate stack size and selectors are allocated, and, if necessary, preserve registers.

Applications must use the same registers for the function 05h and function 09h Protected Mode interface as for a real mode call. This includes the AX register.

Function 07h protected mode calls have a different format.

AX	=	4F07h	
BL	=	00h	Set display CRTC start
	=	80h	Set display CRTC start during vertical retrace
CX	=		Bits 15:0 of display start address
DX	=		Bits 31:16 of display start address

The Protected Mode application must keep track of the color depth and scanline length to calculate the new start address. A value programmed out-of-range yields unpredictable results.

6.6 Description of VESA®/VBE Functions

The following sections list and describe the VESA/VBE functions.

Function: 00h — Return VBE Controller Information

This required function returns the capabilities of the display controller, the revision level of the VBE implementation, and vendor-specific information to support all display controllers in the field.

The purpose of this function is to provide information to the calling program about the general capabilities of the installed the VBE software and hardware. This function fills an information block structure at the address specified by the caller. The VbeInfoBlock information block size is 256 bytes for VBE v1.x, and 512 bytes for VBE v2.0.

Input:	AX =	4F00h	Return VBE controller information.
	ES:DI =		Pointer to buffer to place VbeInfoBlock structure. (Set VbeSignature to ‘VBE2’ when function is called to indicate VBE v2.0 information is desired; the information block size is 512 bytes.)
Output:	AX =		VBE return status.

NOTES:

1) All other registers are preserved.

The information block has the following structure:

```
VbeInfoBlock struc
VbeSignature          db       ‘VESA’   ; VBE signature
VbeVersion            dw        0200h   ; VBE version
OemStringPtr          dd          ?     ; Pointer to OEM string
Capabilities          db      4 dup (?) ; Capabilities of graphics controller
VideoModePtr          dd          ?     ; Pointer to Video mode list
TotalMemory           dw          ?     ; Number of 64-Kbyte memory blocks
                                        ; added for VBE 2.0
OemSoftwareRev        dw          ?     ; VBE implementation software revision
OemVendorNamePtr      dd          ?     ; Pointer to Vendor Name String
OemProductNamePtr     dd          ?     ; Pointer to Product Name String
OemProductRevPtr      dd          ?     ; Pointer to Product Revision String
Reserved              db     222 dup (?); Reserved for VBE implementation
                                        ; scratch area
OemData               db     256 dup (?); Data Area for OEM Strings
VbeInfoBlock ends
```

NOTE: All data in this structure is subject to change by the VBE implementation when VBE function 00h is called. Therefore, it should not be used by the application to store any data.

6.6.1 Description of the VbeInfoBlock Structure Fields

The **VbeSignature** field is filled with the ASCII characters ‘VESA’ by the VBE implementation. VBE v2.0 applications should preset this field with the ASCII characters ‘VBE2’ to indicate to the VBE implementation that the VBE v2.0 extended information is desired, and the VbeInfoBlock is 512 bytes in size. Upon return from VBE function 00h, this field should always be set to ‘VESA’ by the VBE implementation.

The **VbeVersion** is a BCD value specifiying what level of the VBE standard is implemented in the software. The higher byte specifies the major version number. The lower byte specifies the minor version number.

NOTE: The BCD value for VBE v2.0 is 0200h, and the BCD value for VBE v1.2 is 0102h. In the past there have been some applications misinterpreting these BCD values. For example, BCD 0102h was interpreted as 1.02, which is incorrect.

The **OemStringPtr** is a Real mode far pointer to a null-terminated, OEM-defined string. This string can be used to identify the graphics controller device or OEM product family for hardware-specific display drivers. There are no restrictions on the format of the string. This pointer may point into either ROM or RAM, depending on the specific implementation. VBE v2.0 BIOS implementations must place this string in the OemData area within the VbeInfoBlock if ‘VBE2’ is preset in the VbeSignature field on entry to function 00h. This makes it possible to convert the Real mode address to an offset within the VbeInfoBlock for Protected mode applications.

NOTE: The length of the OEMString is not defined; for space considerations it is recommend that a string length of less than 256 bytes is used.

The **Capabilities** field indicates the support of specific features in the graphics environment. The bits are defined as:

D0 =	0	DAC fixed width with 6 bits per primary color.
	1	DAC width switchable to 8 bits per primary color.
D1 =	0	Controller VGA compatible.
	1	Controller not VGA compatible.
D2 =	0	Normal RAMDAC operation.
	1	When programming large blocks of information to the RAMDAC, use the blank bit in function 09h.
D3–31 =		Reserved.

NOTES:

1) For BIOS implementation, the DAC must always be restored to 6 bits per primary color as default upon a mode set. If the DAC switched to 8 bits per primary, the mode set must restore the DAC to 6 bits per primary color to ensure that the application programmer does not have to reset it.
2) Application programmer:
   If a DAC is switchable, the programmer can assume that the DAC is restored to 6 bits per primary upon a mode set. For DAC use within an application, the application program is responsible for setting the DAC to 8 bits per primary mode using function 08h.

VGA compatibility is defined as supporting all standard IBM VGA modes, fonts and I/O ports. However, VGA compatibility does not guarantee that all modes that can be set are VGA compatible, or that the 8 × 14 font is available.

The need for D2 = 1 – “program the RAMDAC using the blank bit in function 09h” – is for older RAMDACs, where programming the RAM values during display time causes a ‘snow-like’ effect on the screen. Newer RAMDACs do not have this limitation and can be easily programmed at any time. Older RAMDACs require that they be blanked so as not to display the snow while values change during display time. This bit informs the software to make the function call with BL = 80h, rather than BL = 00h, to ensure the minimization of this effect.

The **VideoModePtr** points to a list of mode numbers for all display modes supported by the VBE implementation. Each mode number occupies one word. The list of mode numbers is terminated by a –1 (0FFFFh). The mode numbers in this list represent all potentially supported modes by the display controller. Refer to Table 6-18 for a list of the VESA VBE mode numbers. VBE v2.0 BIOS implementations must place this mode list in the Reserved area of the VbeInfoBlock or have it statically stored within the VBE implementation – if ‘VBE2’ is preset in the VbeSignature field on entry to function 00h.

NOTES:

1) It is the responsibility of the application to verify the actual availability of any mode returned by this function through the Return VBE Mode Information (VBE function 01h) call. Some of the returned modes may not be available due to the actual amount of memory physically installed on the display board or due to the capabilities of the attached monitor.
2) If the VideoModeList is found to contain no entries (starts with 0FFFFh), assume that the VBE implementation is a ‘stub’ implementation where only function 00h is supported for diagnostic or ‘plug and play’ functions. These stub implementations are not VBE v2.0 compliant and should be implemented only in cases where no space is available to implement the entire VBE.

The **TotalMemory** field indicates the maximum amount of memory physically installed and available to the frame buffer in 64-Kbyte units (for example, 256 Kbytes = 4, 512 Kbytes = 8). Not all video modes can address all this memory. See **ModeInfoBlock** for detailed information about the addressable memory for a given mode.

The **OemSoftwareRev** field is a BCD value that specifies the OEM revision level of the VBE software. The high byte specifies the major version number; the low byte specifies the minor version number. This field can identify the OEM VBE software release. This field is only filled in when 'VBE2' is preset in the VbeSignature field on entry to function 00h.

The **OemVendorNamePtr** is a pointer to a null-terminated string containing the name of the vendor that produced the display controller board product. (This string may be contained in the VbeInfoBlock or the VBE implementation.) This field is only filled in when 'VBE2' is preset in the VbeSignature field on entry to function 00h. (Note that the length of the strings OemProductRev, OemProductName, and OemVendorName (including terminators) summed must fit within a 256-byte buffer. This allows a return in the OemData field – if required.)

The **OemProductNamePtr** is a pointer to a null-terminated string containing the product name of the display controller board. (This string may be contained in the VbeInfoBlock or the VBE implementation.) This field is only filled in when 'VBE2' is preset in the VbeSignature field on entry to function 00h. (Note that the length of the strings OemProductRev, OemProductName, and OemVendorName (including terminators) summed must fit within a 256-byte buffer. This allows a return in the OemData field – if required.)

The **OemProductRevPtr** is a pointer to a null-terminated string containing the revision or manufacturing level of the display controller board product. (This string may be contained in the VbeInfoBlock or the VBE implementation.) This field can determine which production revision of the display controller board is installed. This field is only filled in when 'VBE2' is preset in the VbeSignature field on entry to function 00h. (Note that the length of the strings OemProductRev, OemProductName, and OemVendorName (including terminators) summed must fit within a 256-byte buffer; this is to allow for return in the OemData field – if required.)

The **Reserved** field is a space reserved for dynamically building the VideoModeList, if required, if the VideoModeList is not statically stored within the VBE implementation. Do not use this field for anything else. This field may be reassigned in future versions. Application software should not assume that the information in this field is valid.

The **OemData** field is a 256-byte data area that is used to return OEM information returned by VBE function 00h when 'VBE2' is preset in the VbeSignature field. The OemVendorName, OemProductName, and OemProductRev strings are copied into this area by the VBE implementation. This area is only used by VBE v2.0 and above implementations when 'VBE2' is preset in the VbeSignature field.

Function: 01h — Return VBE Mode Information

This required function returns extended information about a specific VBE display mode from the mode list returned by VBE function 00h. This function fills the mode information block, ModeInfoBlock, structure with technical details on the requested mode. The ModeInfoBlock structure is provided by the application with a fixed size of 256 bytes.

Information can be obtained for all listed modes in the Video Mode List returned in function 00h. If the requested mode cannot be used or is unavailable, a bit is set in the ModeAttributes field to indicate that the mode is not supported in the current configuration.

Input:	AX =	4F01h	Return VBE mode information
	CX =		Mode number
	ES:DI =		Pointer to ModeInfoBlock structure
Output:	AX =		VBE return status

NOTE: All other registers are preserved.

The mode information block has the following structure:

```
ModeInfoBlock struc
; Mandatory information for all VBE revisions
ModeAttributes           dw        ?     ; Mode attributes
WinAAttributes           db        ?     ; Window A attributes
WinBAttributes           db        ?     ; Window B attributes
WinGranularity           dw        ?     ; Window granularity
WinSize                  dw        ?     ; Window size
WinASegment              dw        ?     ; Window A start segment
WinBSegment              dw        ?     ; Window B start segment
WinFuncPtr               dd        ?     ; Pointer to window function
BytesPerScanLine         dw        ?     ; Bytes per scanline

; Mandatory information for VBE 1.2 and above
XResolution              dw        ?     ; Horizontal resolution in pixels or
                                         ; characters (pixels in graphics modes,
                                         ; characters in text modes.)
YResolution              dw        ?     ; Vertical resolution in pixels or
                                         ; characters
XCharSize                db        ?     ; Character cell width in pixels
YCharSize                db        ?     ; Character cell height in pixels
NumberOfPlanes           db        ?     ; Number of memory planes
BitsPerPixel             db        ?     ; Bits-per-pixel
NumberOfBanks            db        ?     ; Number of banks
MemoryModel              db        ?     ; Memory model type
BankSize                 db        ?     ; Bank size in Kbytes
NumberOfImagePages       db        ?     ; Number of images
Reserved                 db        1     ; Reserved for page function

; Direct Color fields (required for direct/6 and YUV/7 memory models)
RedMaskSize              db        ?     ; Size of direct color red mask in bits
RedFieldPosition         db        ?     ; Bit position of lsb of red mask
GreenMaskSize            db        ?     ; Size of direct color green mask in bits
GreenFieldPosition       db        ?     ; Bit position of lsb of green mask
BlueMaskSize             db        ?     ; Size of direct color blue mask in bits
BlueFieldPosition        db        ?     ; Bit position of lsb of blue mask
RsvdMaskSize             db        ?     ; Size of direct color reserved mask in
                                         ; bits
RsvdFieldPosition        db        ?     ; Bit position of lsb of reserved mask
DirectColorModeInfo      db        ?     ; Direct color mode attributes

; Mandatory information for VBE 2.0 and above
PhysBasePtr              dd        ?     ; Physical address for flat memory frame
                                         ; buffer
```

```
OffScreenMemOffset        dd        ?     ; Pointer to start off screen memory
OffScreenMemSize          dw        ?     ; Amount of off screen memory in 1k
                                          ; units
Reserved                  db      206 dup ; Remainder of ModeInfoBlock
                                    (?)
ModeInfoBlock ends
```

The **ModeAttributes** field describes certain important characteristics of the graphics mode.

The ModeAttributes field is defined as:

D0 =	Mode supported by hardware configuration	
	0	Mode not supported in hardware.
	1	Mode supported in hardware.
D1 =	1	Reserved.
D2 =	TTY Output functions supported by BIOS	
	0	TTY Output functions not supported by BIOS.
	1	TTY Output functions supported by BIOS.
D3 =	Monochrome/color mode (see note below)	
	0	Monochrome mode.
	1	Color mode.
D4 =	Mode type	
	0	Text mode.
	1	Graphics mode.
D5 =	VGA compatible mode	
	0	Yes
	1	No
D6 =	VGA compatible windowed memory mode available?	
	0	Yes
	1	No
D7 =	Linear frame buffer mode available?	
	0	No
	1	Yes
D8–D15 =		Reserved.

Bit D0 is set to indicate that this mode can be initialized in the present hardware configuration. This bit is reset to indicate the unavailability of a graphics mode if it requires a certain monitor type, more memory than is physically installed, and so on.

Bit D1 was used by VBE v1.0 and v1.1 to indicate that the optional information following the BytesPerScanLine field were present in the data structure. This information became mandatory with VBE version 1.2 and above, so D1 is no longer used and should be set to ‘1’. The Direct Color fields are only valid if the MemoryModel field is set to a 6 (Direct Color) or 7 (YUV).

Bit D2 indicates whether the video BIOS has support for output functions like TTY output, scroll, and so on, in this mode. TTY support is recommended, but not required, for all extended text and graphic modes. If bit D2 is set to ‘1’, then the INT10h BIOS must support all of the standard output functions listed below.

All of the following TTY functions must be supported when this bit is set:

01	Set Cursor Size
02	Set Cursor Position
06	Scroll TTY window up or Blank Window
07	Scroll TTY window down or Blank Window
09	Write character and attribute at cursor position
0A	Write character only at cursor position
0E	Write character and advance cursor

Bit D3 is set to indicate color modes, and cleared for monochrome modes.

Bit D4 is set to indicate graphics modes, and cleared for text modes.

NOTE: Monochrome modes map their CRTC address at 3B4h. Color modes map their CRTC address at 3D4h. Monochrome modes have attributes where only bit 3 (video) and bit 4 (intensity) of the attribute controller output are significant. Therefore, monochrome text modes have attributes of off, video, high intensity, blink, and so on. Monochrome graphics modes are two-plane graphics modes and have attributes of off, video, high intensity, and blink. Extended two-color modes (with the CRTC address at 3D4h) are color modes with one bit-per-pixel and one plane. The standard VGA modes, 06h and 11h, are classified as color modes, while the standard VGA modes, 07h and 0Fh, are classified as monochrome modes.

Bit D5 indicates if the mode is compatible with the VGA hardware registers and I/O ports. If this bit is set, then the mode is *not* VGA compatible and no assumptions should be made about the availability of any VGA registers. If this bit is clear, then the standard VGA I/O ports and frame buffer addres (defined in WinASegment and/or WinBSegment) can be assumed.

Bit D6 indicates if the mode provides windowing or banking of the frame buffer into the frame buffer memory region specified by WinASegment and WinBSegment. If set, then windowing of the frame buffer is *not* possible. If this bit is clear, then the device is capable of mapping the frame buffer into the segment specified in WinASegment and/or WinBSegment. (This bit is used in conjunction with bit D7; see Table 6-19 for usage.)

Bit D7 indicates the presence of a linear frame buffer memory model. If this bit is set, the display controller can be put into a flat memory model by setting the mode (VBE function 02h) with the Flat Memory Model bit set. (This bit is used in conjunction with bit D6; see Table 6-19 for usage.)

Table 6-19. Function 01h – Bits 7 and 6 Usage

D7	D6	Usage
0	0	Windowed frame buffer only
0	1	n/a
	0	Both Windowed and Linear[a]
1	1	Linear frame buffer only

[a] Use D14 of the mode number to select the linear buffer on mode set (function 02h).

The **BytesPerScanLine** field specifies how many full bytes are in each logical scanline. The logical scanline could be equal to or larger than the displayed scanline.

WinAAttributes and **WinBAttributes** describe the characteristics of the CPU windowing scheme, such as if the windows exist and are read/writeable.

D0 =	Relocatable window(s) supported		
		0	Single non-relocatable window only.
		1	Relocatable window(s) supported.
D1 =	Window readable		
		0	Window not readable.
		1	Window readable.
D2 =	Window writeable		
		0	Window not writeable.
		1	Window writeable.
D3–D7 =			Reserved .

Even if windowing is not supported (bit D0 = 0 for both window A and window B), an application can assume that the display memory buffer resides at the location specified by WinASegment and/or WinBSegment.

WinGranularity specifies the smallest boundary (in Kbytes) to place the window in the frame buffer memory. The value of this field is undefined if bit D0 of the appropriate WinAttributes field is not set.

WinSize specifies the size of the window in Kbytes.

WinASegment and WinBSegment address specify the segment addresses where the windows are located in the CPU address space.

WinFuncPtr specifies the segment:offset of the VBE memory windowing function. The windowing function can be invoked either through VBE function 05h, or by calling the function directly. A direct call provides faster access to the hardware paging registers than using VBE function 05h, and is intended for use by high-performance applications. If this field is NULL, then VBE function 05h must set the memory window when paging is supported. This direct call method uses the same parameters as VBE function 05h, including AX; VBE v2.0 implementations return the correct return status. VBE v1.2 implementations and earlier do not require the Return Status information to be returned. For more information on the direct call method, see the notes in VBE function 05h on page 6-79.

The **XResolution** and **YResolution** specify the width and height in pixel elements or characters for this display mode. In graphics modes, these fields indicate the number of horizontal and vertical pixels that can be displayed. In text modes, these fields indicate the number of horizontal and vertical character positions. The number of pixel positions for text modes can be calculated by multiplying the returned XResolution and YResolution values by the character cell width and height indicated in the XCharSize and YCharSize fields described below.

The **XCharSize** and **YCharSize** specify the size of the character cell in pixels. This value is not zero based. For example, XCharSize for mode 3 using the 9 point font has a value of 9.

The **NumberOfPlanes** field specifies the number of memory planes available to software in that mode. For standard 16-color VGA graphics, the field is set to 4. For standard packed pixel modes, the field would be set to 1. For 256-color non-Chain-4 modes where the programmer needs to do banking to address all pixels, this value should be set to the number of banks required to get to all the pixels (typically, 4 or 8).

The **BitsPerPixel** field specifies the total number of bits allocated to one pixel. For example, a standard VGA 4 Plane 16-color graphics mode would have a 4 in this field and a packed pixel 256-color graphics mode would specify 8 in this field. The number of bits-per-pixel per plane can normally be derived by dividing the BitsPerPixel field by the NumberOfPlanes field.

The **MemoryModel** field specifies the general type of memory organization used in this mode. The following models are defined:

Field	Mode
00h =	Text
01h =	CGA graphics
02h =	Hercules graphics
03h =	Planar
04h =	Packed pixel
05h =	Non-Chain 4, 256 color
06h =	Direct Color
07h =	YUV
08h-0Fh =	Reserved; to be defined by VESA.
10h-FFh =	To be defined by OEM.

In VBE v1.1 and earlier, the defined Direct Color graphics modes with pixel formats 1:5:5:5, 8:8:8, and 8:8:8:8 are a Packed Pixel model with 16, 24, and 32 bpp, respectively. In VBE v1.2 and later, the Direct Color modes use the Direct Color memory model and use the MaskSize and FieldPosition fields of ModeInfoBlock to describe the pixel format. BitsPerPixel is always defined in bits to be the total memory size of the pixel.

NumberOfBanks. This field contains the number of banks where the scanlines are grouped. When the scanline number is divided by the number of banks the quotient is the bank that contains the scanline, and the remainder is the scanline number within the bank. For example, CGA graphics modes have two banks; Hercules graphics mode has four banks. For modes that do not have scanline banks (such as VGA modes 0Dh–13h), this field should be set to ‘1’.

The **BankSize** field specifies the size of a bank (group of scanlines) in 1-Kbyte units. For CGA and Hercules graphics modes the size is 8, as each bank is 8192 bytes in length. For modes that do not have scanline banks (such as VGA modes 0Dh–13h), set this field to ‘0’.

The **NumberOfImagePages** field specifies the ‘total number minus one (–1)’ of complete display images that fit into the frame buffer memory. If this field is non-zero, the application can load more than one image into the frame buffer memory and move the display window within each of those pages. This is only to determine the additional display pages available to the application. To determine the available off-screen memory, use the OffScreenMemOffset and OffScreenMemSize information.

NOTE: If the ModeInfoBlock is for an IBM Standard VGA mode and the NumberOfImagePages field contains more pages than would be found in a 256-Kbyte implementation, the TTY support described in the ModeAttributes must be accurate. That is, if the TTY functions are claimed to be supported, they must be supported in all pages, not just the pages normally found in the 256-Kbyte implementation.

The **Reserved** field is defined to support a future VBE feature and is always set to ‘1’ in this version.

The **RedMaskSize**, **GreenMaskSize**, **BlueMaskSize**, and **RsvdMaskSize** fields define the size, in bits, of the RED, GREEN, and BLUE components of a Direct Color pixel. A bit mask can be constructed from the MaskSize fields using simple shift arithmetic. For example, the MaskSize values for a Direct Color 5:6:5 mode would be 5, 6, 5, and 0 for the RED, GREEN, BLUE, and reserved fields, respectively. Note that in the YUV MemoryModel field, the RED field is used for V, the GREEN field is used for Y, and the BLUE field is used for U. Set the MaskSize fields to '0' in modes using a memory model that does not have pixels with component fields.

The **RedFieldPosition**, **GreenFieldPosition**, **BlueFieldPosition**, and **RsvdFieldPosition** fields define the bit position within the direct color pixel or YUV pixel of the least-significant bit of the respective color component. A color value can be aligned with its pixel field by shifting the value left by the FieldPosition. For example, the FieldPosition values for a Direct Color 5:6:5 mode would be 11, 5, 0, and 0, for the RED, GREEN, BLUE, and reserved fields, respectively. Note that in the YUV MemoryModel, the RED field is used for V, the GREEN field is used for Y, and the BLUE field is used for U. Set the FieldPosition fields to '0' in modes using a memory model that does not have pixels with component fields.

The **DirectColorModeInfo** field describes important characteristics of Direct Color modes. Bit D0 specifies if the color ramp of the DAC is fixed or programmable. If the color ramp is fixed, then it cannot be changed. If the color ramp is programmable, it is assumed that the RED, GREEN, and BLUE lookup tables can be loaded by using VBE function 09h. Bit D1 specifies if the bits in the Rsvd field of the Direct Color pixel can be used by the application or are reserved, and thus unusable.

D0 = Color ramp is fixed/programmable
- 0 Color ramp fixed.
- 1 Color ramp programmable.

D1 = Bits in Rsvd field are usable/reserved
- 0 Bits in Rsvd field are reserved.
- 1 Bits in Rsvd field are usable by the application.

The **PhysBasePtr** is a 32-bit physical address of the start of the frame buffer memory when the controller is in Flat-Frame Buffer Memory mode. If this mode is not available, then this field is '0'.

The **OffScreenMemOffset** is a 32-bit offset from the start of the frame buffer memory. Extra off-screen memory required by the controller may be located either before or after this off-screen memory. The programmer must check OffScreenMemSize to determine the amount of off-screen memory available to the application.

The **OffScreenMemSize** contains the amount of available, contiguous off-screen memory in 1-Kbyte units that can be used by the application.

NOTE: VBE v1.1 and later zero-out all unused fields in the ModeInformationBlock, always returning exactly 256 bytes. This facilitates upward compatibility with future versions of the standard, because any newly added fields are designed so that values of zero indicate nominal defaults or non-implementation of optional features. (For example, a field containing a bitmask of extended capabilities would reflect the absence of all such capabilities.) Applications requiring backward compatiblity to VBE v1.0 should preinitialize the 256-byte buffer before calling the Return VBE Mode Information function.

Function: 02h — Set VBE Mode

This required function initializes the controller and sets a VBE mode. The format of VESA VBE mode numbers is described earlier in this chapter. If the mode cannot be set, the BIOS should leave the graphics environment unchanged and return a failure error code.

Input:	AX =	4F02h	Set VBE mode
	BX =		Desired mode to set
	D0–D8 =		Mode number
	D9–D13 =		Reserved (must be ‘0’)
	D14 =	0	Use windowed frame buffer model
		1	Use linear/flat frame buffer model
	D15 =	0	Clear display memory
		1	Do not clear display memory
Output:	AX =	=	VBE return status

NOTE: All other registers are preserved.

If the requested mode number is not available, then the call fails and returns AH = 01h to indicate the failure to the application.

If bit D14 is set, the mode is initialized for use with a flat-frame buffer model. The base address of the frame buffer can be determined from the extended mode information returned by VBE function 01h. If D14 is set and a linear-frame buffer model is not available, then the call fails and returns AH = 01h to indicate the failure to the application.

If bit D15 is not set, all reported image pages, based on function 00h returned information NumberOfImagePages, are cleared to ‘00h’ in Graphics mode and ‘20 07’ in Text mode. Memory over and above the reported image pages does not change. If bit D15 is set, then the contents of the frame buffer after the mode change is undefined. Note that the 1-byte mode numbers used in function 00h of an IBM VGA-compatible BIOS use D7 to signify this function. If D7 is set for an IBM compatible mode set using function 02h, this mode set fails. VBE-aware applications must use the memory clear bit in D15.

NOTES:

1) VBE BIOS v2.0 implementations should also update the BIOS Data Area 40:87 Memory Clear bit so that VBE function 03h can return this flag. VBE BIOS v1.2 and earlier implementations ignore the Memory Clear bit.

2) This call should not set modes that are not listed in the list of supported modes. In addition, all modes (including IBM standard VGA modes), if listed as supported, must have ModeInfoBlock structures associated with them.

Function: 03h — Return Current VBE Mode

This required function returns the current VBE mode. The format of VBE mode numbers is described earlier in this chapter.

Input:	AX =		Return current VBE mode.
Output:	AX =		VBE return status.
	BX =		Current VBE mode.
	D0–D13 =		Mode number.
	D14 =	0	Windowed frame buffer model.
		1	Linear/flat frame buffer model.
	D15 =	0	Memory cleared at last mode set.
		1	Memory not cleared at last mode set.

NOTES:

1) All other registers are preserved.
2) Version 1.x:
   In a standard VGA BIOS, function 0Fh (Read Current Video State) returns the current graphics mode in the AL register. In D7 of AL, it also returns the status of the memory clear bit (D7 of 40:87). This bit is set if the mode was set without clearing memory. In this VBE function, the memory clear bit is not returned in BX because this function only returns the video mode. If it is necessary that an application obtain the memory clear bit, it should call the standard VGA BIOS function 0Fh.
3) Version 2.x:
   Unlike v1.x VBE implementations, the memory clear flag is returned. The application should *not* call the standard VGA BIOS function 0Fh if the mode was set with VBE function 02h.
4) The mode number returned must be the same mode number used in the VBE function 02h mode set.
5) This function is not guaranteed to return an accurate mode value if the mode set was not done with VBE function 02h.

Function: 04h — Save/Restore State

This required function provides a complete mechanism to save and restore the display controller hardware state. The functions are a superset of the three subfunctions under the standard VGA BIOS function 1Ch (Save/Restore State) that does not guarantee that the extended registers of the video device are saved or restored. The complete hardware state (except frame buffer memory) should be saveable/restorable by setting the requested states mask (in the CX register) to ‘000Fh’.

Input:	AX =	4F04h	Save/Restore state.
	DL =	00h	Return Save/Restore state buffer size.
		01h	Save state.
		02h	Restore state.
	CX =		Requested states.
	D0=		Save/Restore controller hardware state.
	D1=		Save/Restore BIOS data state.
	D2=		Save/Restore DAC state.
	D3=		Save/Restore Register state.
	ES:BX =		Pointer to buffer (if DL <> 00h).
Output:	AX =		VBE return status.
	BX =		Number of 64-byte blocks to hold the state buffer (if DL = 00h).

NOTE: All other registers are preserved.

Function: 05h — Display Window Control

This required function sets or gets the position of the specified display window or page in the frame buffer memory by adjusting the necessary hardware paging registers. Proper use of this function requires that the software first use VBE function 01h (Return VBE Mode Information) to determine the size, location, and granularity of the windows.

For performance reasons, it may be more efficient to call this function directly, without incurring the INT 10h overhead. VBE function 01h returns the segment:offset of this windowing function that may be called directly for this reason. Note that a different entry point may be returned based upon the selected mode. Therefore, it is necessary to retrieve this segment: offset specifically for each desired mode.

Input:	AX =	4F05h	VBE Display Window Control.
	BH =	00h	Set memory window.
		01h	Get memory window.
	BL =		Window number.
		00h	Window A.
		01h	Window B.
	DX =		Window number in video memory in window granularity units (Set Memory Window only).
Output:	AX =		VBE Return Status.
	DX =		Window number in window granularity units (Get Memory Window only).

NOTES:

1) In VBE v1.2 implementations, the direct far call version returns no Return Status information to the application. Also, in the far call version, the AX and DX registers is destroyed. Therefore, if AX and/or DX must be preserved, the application must do so prior to making the far call. The application must still load the input arguments in BH, BL, and DX (for Set Window). In VBE v2.0 implementations, the BIOS returns the correct Return Status and therefore, the application must assume that AX and DX are destroyed.
2) Application programmers:
   This function is not intended for use in Linear Frame Buffer mode. If this function is requested, the function call will fail with the VBE Completion code AH = 03h.
3) VBE BIOS implementation:
   If this function is called while in a linear frame buffer memory model, this function must fail with completion code AH = 03h.

Function: 06h — Set/Get Logical Scanline Length

This required function sets or gets the length of a logical scanline. This allows an application to set up a logical display memory buffer that is wider than the displayed area. VBE function 07h (Set/Get Display Start) then allows the application to set the starting position that is to be displayed.

Input:	AX =	4F06h	VBE Set/Get logical scanline length.
	BL =	00h	Set scanline length in pixels.
		01h	Get scanline length.
		02h	Set scanline length in bytes.
		03h	Get maximum scanline length.
	CX =		If BL = 00h, desired width in pixels.
			If BL = 02h, desired width in bytes (ignore for Get functions).
Output:	AX =		VBE return status.
	BX =		Bytes per scanline.
	CX =		Actual pixels per scanline (truncated to nearest complete pixel).
	DX =		Maximum number of scanlines

NOTES:

1) The desired width in pixels or bytes may not be achievable because of hardware considerations. The next larger value is selected that accommodates the desired number of pixels or bytes, and the actual number of pixels is returned in CX. BX returns a value that, when added to a pointer into display memory, points to the next scanline. For example, in VGA mode 13h this is 320, but in mode 12h this is 80. DX returns the number of logical scanlines based upon the new scanline length and the total memory installed and usable in this display mode.
2) This function is also valid in VBE supported text modes. In VBE-supported text modes the application should convert the character line length to pixel line length by getting the current character cell width through the XCharSize field returned in ModeInfoBlock, multiplying that times the desired number of characters per line and passing that value in the CX register. In addition, this function only works if the line length is specified in character granularity. That is, only in 8-dot modes do multiples of 8 work. Any value that is not in character granularity results in a function call failure.

3) On a failure to set scanline length by setting a CX value too large, the function fails with error code 02h.
4) The value returned when BL = 03h is the lesser of either the maximum line length that the hardware can support, or the longest scanline length that would support the number of lines in the current video mode.

Function: 07h — Set/Get Display Start

This required function selects the pixel to be displayed in the upper-left corner of the display. This function can be used to pan and scroll around logical screens that are larger than the displayed screen. This function can also be used to rapidly switch between two different displayed screens for double buffered animation effects.

Input:	AX =	4F07h	VBE Set/Get Display Start control.
	BH =	00h	Reserved; must be 00h.
	BL =	00h	Set Display Start.
		01h	Get Display Start.
		80h	Set Display Start during vertical retrace.
	CX =		First displayed pixel in scanline (Set Display Start only).
	DX =		First displayed scanline (Set Display Start only).
Output:	AX =		VBE return status.
	BH =	00h	Reserved and is ‘0’ (Get Display Start only).
	CX =		First displayed pixel in scanline (Get Display Start only).
	DX =		First displayed scanline (Get Display Start only).

NOTE: This function is also valid in text modes. To use this function in text mode, the application should convert the character coordinates to pixel coordinates by using XCharSize and YCharSize returned in the ModeInfoBlock. If the requested Display Start coordinates do not allow for a full page of video memory or the hardware does not support memory wrapping, the function call should fail and no changes should be made. As a general case, if a requested Display Start is not available, fail the function call and make no changes.

Function: 08h — Set/Get DAC Palette Format

This required function manipulates the operating mode or format of the DAC palette. Some DACs are configurable to provide 6 or 8 bits or more of color definition per red, green, and blue primary colors. The DAC palette width is assumed to be reset to the standard VGA value of 6-bits per primary color during any mode set.

Input:	AX =	4F08h	VBE Set/Get palette format.
	BL =	00h	Set DAC palette format.
		01h	Get DAC palette format.
	BH =		Desired bits of color per primary (Set DAC palette format only).
Output:	AX =		VBE return status.
	BH =		Current number of bits of color per primary.

An application can determine if DAC switching is available by querying bit D0 of the Capabilities field of the VbeInfoBlock structure returned by VBE function 00h (Return Controller Information). The application can then attempt to set the DAC palette width to the desired value. If the display controller hardware is not capable of selecting the requested palette width, then the next lower value available to the display controller hardware is selected. The resulting palette width is returned. This function returns failure code AH = 03h if called in Direct Color or YUV mode.

Function: 09h — Set/Get Palette Data

This required function is important for RAMDACs larger than a standard VGA RAMDAC. The standard INT 10h BIOS Palette function calls assume standard VGA ports and VGA palette widths. This function offers a palette interface that is independent of the VGA assumptions.

Input:	AX =	4F09h	VBE Load/Unload palette data.
	BL =	00h	Set palette data.
		01h	Get palette data.
		02h	Set secondary palette data.
		03h	Get secondary palette data.
		80h	Set palette data during vertical retrace with Blank bit on.
	CX =		Number of palette registers to update (to a maximum of 256).
	DX =		First of the palette registers to update (start).
	ES:DI =		Table of palette values (see below for format).
Output:	AX =		VBE Return Status.
Format of palette values: Alignment byte, Red byte, Green byte, Blue byte.			

NOTES:

1) The need for BL = 80h is for older RAMDACs where programming the RAM values during display time causes a ‘snow-like’ effect on the screen. Newer RAMDACs do not have this limitation and can be easily programmed at any time. Older RAMDACs require programming only during a non-display time to stop the snow effect seen when changing the DAC values. When this is requested, the VBE implementation programs the DAC with blanking on. Check D2 of the Capabilities field returned by VBE function 00h to determine if 80h should be used instead of 00h.
2) The need for the secondary palette is for anticipated future palette extensions. If a secondary palette does not exist in an implementation and these calls are made, the VBE implementation returns error code 02h.
3) In 6-bit mode, these 6 bits are LSB. This is due to speed as the application can typically shift the data faster than the BIOS.
4) All applications should assume the DAC default is 6-bit mode. The application is responsible for switching the DAC to higher color modes using function 08h.
5) Query VBE function 08h to determine the RAMDAC width before loading a new palette.

Function: 0Ah — Return VBE Protected Mode Interface

This required function call returns a pointer to a table that contains code for a 32-bit protected mode interface. This pointer can either be copied into local 32-bit memory space or executed from ROM, providing the calling application sets all required selectors and I/O accesses correctly. This function returns a pointer (in real mode space) with offsets to the code fragments, and additionally returns an offset to a table containing non-VGA port and memory locations that an application may require I/O access to.

Input:	AX =	4F0Ah	VBE v2.0 Protected mode interface.
	BL =	00h	Return protected mode table.
Output:	AX =		Status
	ES =		Real mode segment of table.
	DI =		Offset of table.
	CX =		Length of table including protected mode code in bytes (for copying purposes).

The format of the table is:

ES:DI + 00h	Word offset in table of Protected mode code for function 5 for Set Window call.
ES:DI + 02h	Word offset in table of Protected mode code for function 7 for set Display Start.
ES:DI + 04h	Word Offset in table of Protected mode code for function 9 for set Primary Palette Data.
ES:DI + 06h	Word Offset in table of ports and memory locations where the application requires I/O privileges. (Optional: If unsupported, this must be 0000h – see the sub-table for format.)
ES:DI + ?	Variable remainder of table including code.

The format of the sub-table (ports and memory locations) is:

Port, Port, ... , Port, Terminate Port List with FF FF, Memory locations (4 bytes), Length (2 bytes), Terminate Memory List with FF FF.

Example 1. For Port/Index combination 3DE/Fh and Memory locations DE800-DEA00h (length = 200h) the table would look like:

DE 03 DF 03 FF FF 00 E8 0D 00 00 02 FF FF

Example 2. For only the ports, it would look like:

DE 03 DF 03 FF FF FF FF

Example 3. For only the memory locations, it would look like:

FF FF 00 E8 0D 00 00 02 FF FF

NOTES:

1) All protected mode functions should end with a near RET – as opposed to FAR RET – to allow the application software to CALL the code from within the ROM.
2) The port and memory location sub-table does not include the frame buffer memory location. The frame buffer memory location is contained within the ModeInfoBlock returned by VBE function 01h.
3) The protected mode code is assembled for a 32-bit code segment. When copying it, the application must copy the code to a 32-bit code segment.
4) It is the responsibility of the application to ensure that the selectors and segments are set up correctly.

5) Currently undefined registers may be destroyed, with the exception of ESI, EBP, DS, and SS.
6) Applications must use the same registers for function 05h and function 09h Protected Mode interface as used in a real mode call. This includes the AX register.
7) Function 07h protected mode calls have a different format.

AX =	4F07h	
BL =	00h	Set Display CRTC Start.
	80h	Set Display CRTC Start during vertical retrace.
CX =		Bits 15:0 of display start address.
DX =		Bits 31:16 of display start address.

The protected mode application must keep track of the color depth and scanline length to calculate the new start address. If a value an out-of-range value is programmed, unpredictable results occur.

6.6.2 VBE Display PM (Power Management) Functions

VESA VBE subfunction 10h implements the VBE/PM services. The VBE/PM services are defined below and are not included in the VBE standard documentation.

Function: 00h — Report VBE/PM Capabilities

Input:	AH =	4Fh	VESA extension.
	AL =	10h	VBE/PM services.
	BL =	00h	Report VBE/PM capabilities.
	CX =	00h	Controller unit number (00 = primary controller).
	ES:DI		Null pointer, must be 0000:0000h in v1.0.
			Reserved for future use.
Output:	AX =		Status
	BH =		Power saving state signals supported by the controller.[a] 1 = supported, 0 = not supported.
		bits 7:4	Reserved for future power control of the display controller or other related circuits.
		bit 3	REDUCED ON[b]
		bit 2	OFF
		bit 1	SUSPEND
		bit 0	STAND BY
	BL =		VBE/PM version number (0001 0000b for this version).
		bits 0-3	Minor version number.
		bits 4-7	Major version number.
	CX=		Unchanged.
	ES:DI =		Unchanged.

[a] The attached display may not support all the power states that the controller can signal. It is the responsibility of the power management program to determine which power saving states are offered by the controller. If the controller can determine which power saving state implemented in the attached display device, this function reports the power saving states supported by both the controller and the display.

[b] REDUCED ON is not supported by DPMS v1.0 displays and is intended for use by flat panel displays.

NOTE: All other registers may be destroyed.

Function: 01h — Set Display Power State

Input:	AH =	4Fh	VESA extension.
	AL =	10h	VBE/PM services.
	BL =	01h	Set Display Power state.
	BH =		Requested Power state.
		00h	ON
		01h	STAND BY
		02h	SUSPEND
		04h	OFF
		08h	REDUCED ON[a]
	All other BH values are currently undefined and are reserved for future power control of the display controller.		
	CX =	00h	Controller unit number (00 = primary controller).
Output:	AX =		Status[b]
	BH =		Unchanged
	CX =		Unchanged

[a] REDUCED ON is not supported by DPMS v1.0 displays and is intended for use by flat panel displays.

[b] If the requested state is not available, this function returns AX = 014Fh to indicate that the function is supported but the call failed. In this case, the BH register and Display Power State remain unchanged.

NOTE: All other registers may be destroyed.

Function: 02h — Get Display Power State

Input:	AH =	4Fh	VESA extension.
	AL =	10h	VBE/PM services.
	BL =	02h	Get Display Power state.
	CX =	00h	Controller unit number (00 = primary controller).
Output:	AX =		Status[a]
	BH =		Power state currently requested by the controller.
		00h	ON
		01h	STAND BY
		02h	SUSPEND
		04h	OFF
		08h	REDUCED ON[b]
		All other BH values are reserved and can be used to signal other power saving states in future revisions of VBE/PM. To ensure future compatibility, applications written for VBE/PM v1.0 should ignore the value of bits 7:4.	
	CX =		Unchanged

[a] If this function is not supported by the controller hardware, AL-01 should be returned in the status register. This function should always read the controller hardware to determine the current power state. This function should not rely on any form of information stored in memory about the power state that was last initiated through a VBE/PM call. Power state signaling can be initiated through many different hardware and software interfaces, making any stored information potentially invalid.

[b] REDUCED ON is not supported by DPMS v1.0 displays and is intended for use by Flat Panel Displays.

NOTE: All other registers may be destroyed.

6.6.3 VBE Display Identification (DDC) Functions

VESA VBE subfunction 15h implements the VBE/DDC services. The VBE/DDC services are defined below and are not included in the VBE standard documentation.

Function: 00h — Report VBE/DDC Capabilities

Input:	AH =	4Fh	VESA extension.
	AL =	15h	VBE/DDC services.
	BL =	00h	Report DDC capabilities.
	CX =	00h	Controller unit number (00 = primary controller).
	ES:DI =		NULL pointer; must be ‘00’ in v1.0. Reserved for future use.
Output:	AX =		Status
	BH =		Approximate time in seconds – rounded up – to transfer one EDID block (128 bytes).
	BL =		DDC level supported.[a]
	bit 0	= 0	DDC1 not supported.
		= 1	DDC1 supported.
	bit 1	= 0	DDC2 not supported.
		= 1	DDC2 supported.
	bit 2	= 0	Screen not blanked during data transfer.[b]
		= 1	Screen blanked during data transfer.
	CX =		Unchanged
	ES:DI =		Unchanged

[a] DDC level supported by both the display and the controller.

[b] This refers to the behavior of the controller and the VBE/DDC software.

NOTE: All other registers may be destroyed.

Function: 01h — Read EDID

Input:	AH =	4Fh	VESA extension.
	AL =	15h	VBE/DDC services.
	BL =	01h	Read EDID.
	CX =	00h	Controller unit number (00 = primary controller).
	DX =	00h	EDID block number. Zero is only valid value in v1.0.
	ES:DI =		Pointer to area where the EDID block (128 bytes) is returned.
Output:	AX =		Status
	BH =		Unchanged
	CX =		Unchanged
	ES:DI =		Pointer to area where the EDID block is returned.

NOTE: All other registers may be destroyed.

Function: 02h — Read VDIF Block

Input:	AH =	4Fh	VESA extension.
	AL =	15h	VBE/DDC services.
	BL =	02h	Read VDIF block.
	CX =	00h	Controller unit number (00 = primary controller).
	DX =		VDIF block number (64 byte block).
	ES:DI =		Pointer to area where the VDIF block is returned.
Output:	AX =		Status
	BX =		Unchanged
	CX =		Unchanged
	ES:DI =		Pointer to area containing the VDIF block (64 byte).

NOTE: All other registers may be destroyed.

6.7 Cirrus Logic BIOS Extensions

This section covers the Cirrus Logic extensions to the VGA BIOS. The Cirrus Logic BIOS supports all standard VGA BIOS INT10h video service functions. In addition, the BIOS provides extensive support for various features of the Cirrus Logic VGA controller. These functions are available as extended functions under INT10h. The standard VGA BIOS INT10h video service functions are described in Section 6.1.3. All extended function calls preserve the CPU registers, except those that pass information from the BIOS.

The following sections list and describe the Cirrus Logic functions.

6.7.1 Inquire VGA Type

This function provides a mechanism for software to determine the type of Cirrus Logic VGA controller, silicon revision number, and the corresponding hardware capabilities. BIOS versions that do not support these functionalities preserve the input value in AL register. BL is invalid when runnning in Windows 3.1 or Windows 95.

Input:	AH =	12h	
	BL =	80h	
Output:	AX =		Controller type.
		0 =	No extended alternate select support.
		1 =	Reserved
		60h =	CL-GD5462
		61h =	CL-GD5464
	BL =		Configuration space revision ID register.

6.7.2 Inquire BIOS Version Number

This function provides a mechanism for software to determine the BIOS version number.

Input:	AH =	12h	
	BL =	81h	
Output:	AH =		Major BIOS version number.
	AL =		Minor BIOS version number.
	For example, if BIOS v1.02, then AH is 01 and AL is 02.		

6.7.3 Inquire Cirrus Logic Design Revision Code

This function allows software to determine the Cirrus Logic silicon revision. This function does not work under Windows 3.1 or Windows 95.

Input:	AH =	12h	
	BL =	82h	
Output:	AL =		Configuration space Revision ID register.

6.7.4 Return Installed Memory

This function returns the amount of video memory present in 64-Kbyte units.

Input:	AH =	12h	
	BL=	85h	
Output:	AL =		Amount of video memory present in 64-Kbyte units.

6.7.5 Inquire User Options

This function returns the current status of user options. The values of the vertical frequencies and maximum vertical resolution correspond to the values defines as input for functions A4h, Set Monitor Type (Vertical).

Input:	AH =	12h		
	BL =	9Ah		
Output:	AX =	Contains the following options:		
		Bit		Description
		13:0		Reserved.
		14	= 0	640 × 480 refresh 60 Hz.
			=1	640 × 480 refresh greater than 60 Hz.
		15		Reserved
	BX =	15:0		Reserved
	CX =	Contains the following options:		
		Bit		Description
		0		Reserved
		3:1		1280 × 1024 vertical frequency.
		7:4		Maximum vertical resolution.
		11:8		800 × 600 vertical frequency.
		15:12		1024 × 768 vertical frequency.
	DX=	DL[3:0]		
		DL[7:4]		

6.7.6 Query Video Mode Availability

Input:	AH =	12h	
	AL =		Video mode number (0–7fh).
	BL =	A0h	
Output:	AH =	Bit 0	
		= 0	Video mode not supported.
		= 1	Video mode supported.
	DS:SI =		Pointer to standard video parameters or FFFF:FFFF if standard parameters are undefined for this mode.
	ES:DI =		Pointer to supplemental video parameters or FFFF:FFFF if supplemental parameters are undefined for this mode.
	BX =		Offset to BIOS sub-routine to ‘fixup’ the parameters pointed to by DS:SI. This routine requires that ES:DI points to the proper supplemental video parameters.

6.7.7 Read Monitor ID/Type

This function uses the analog sense circuitry to detect the type of monitor. The digital monitor ID pins are not used to read the monitor type. The monitor ID returned in register BH is determined by the monitor type sensed (color, monochrome, or none) and may not correspond to the actual digital monitor ID of the current monitor. This function is typically used as a diagnostic function to test the monitor sense occurring during POST. The capabilities (refresh rates supported) of the monitor are determined by the parameters passed to function A4h, Set Monitor Type (Vertical).

Input:	AH =	12h	
	BL =	A1h	Read monitor ID and type from 15-pin connector
Output:	BH =		Monitor ID
	09h =		IBM 8604/8507 or equivalent
	0Ah =		IBM 8514 or equivalent
	0Bh =		IBM 8515 or equivalent
	0Dh =		IBM 8503 or equivalent
	0Eh =		IBM 8512/8513 or equivalent
	0Fh =		No monitor
	00..08h, 0Ch =		Reserved
	BL =		Monitor gender
	00 =		Color display
	01 =		Grayscale display
	02 =		No display

6.7.8 Set Monitor Type (Vertical)

This function sets the monitor type in terms of vertical timings. The monitor type information is used by the BIOS to determine the frequency to use when selecting an extended mode. It is also used – in conjunction with the amount of display memory available – to determine the available extended modes. The monitor type can be read back using Function 9A.

To maintain compatibility with previous Cirrus Logic BIOS releases, obsolete frequencies are not removed from this function. The appearance of any frequency in the description of this BIOS call is no guarantee that any given BIOS actually supports that frequency. In general, the trend is toward supporting higher frequencies and deleting support for lower frequencies.

Input:	AH =	012h	
	BL =	0A4h	
	AL[3:0] =		Maximum vertical resolution
		000h	480 scanlines
		001h	600 scanlines
		002h	768 scanlines
		003h	1024 scanlines
		004h	1200h scanlines
		005h–00Fh	Reserved
	AL[7:4] =		640 × 480 frequency
		000h	60 Hz
		001h	72 Hz
		002h	75 Hz
		003h	85 Hz
		004h–00Fh	Reserved
	BH[3:0] =		800 × 600 frequency
		000h	56 Hz
		001h	60 Hz
		002h	72 Hz
		003h	75 Hz
		004h	85 Hz
		005h–00Fh	Reserved
	BH[7:4] =		1024 × 768 frequency
		000h	43i Hz

Input: *(cont.)*		001h =	60 Hz
		002h =	70 Hz
		004h =	75 Hz
		005h =	85 Hz
		003h, 006h–00Fh =	Reserved
	CH[3:0] =		1600 × 1200 frequency
		1 =	60 Hz
		0 =	48i Hz
		002h–00Fh	Reserved
	CH[7:4] =		1280 × 1024 frequency
		4 =	85 Hz
		3 =	75 Hz
		2 =	71.2 Hz
		1 =	60 Hz
		0 =	87i Hz
		005h–00Fh	Reserved
	CL =		Reserved
	DX =		Reserved

6.7.9 Set Monitor Type (Horizontal)

This function sets the monitor type in terms of horizontal timings. The monitor type information is used by the BIOS to select the optimal display timings for extended modes. The current monitor type can be read back using function 9A.

Input:	AL =	12h	
	BL =	A2h	Set monitor type.
	AL =		Monitor type to set.
		7 =	Extended super multi-frequency (31.5–64.0 kHz).
		6 =	Super multi-frequency (31.5–56.0 kHz).
		5 =	Extended Multi-frequency (31.5–38.0 kHz).
		4 =	Multi-frequency (31.5–37.8 kHz).
		3 =	Extended Super VGA (31.5–35.5 kHz).
		2 =	Super VGA (31.5–35.1 kHz).
		1 =	8514-compatible (31.5 kHz and 35.5 kHz – interlaced).
		0 =	VGA (31.5 kHz).
Output:			No action.

6.7.10 Generic Fixup

This function can be used to determine if 'fixups' are supported and to process fixups. Fixups are used to override inappropriate register values or add new registers to be processed on mode changes.

Input:	AH =	12h	
	BL =	A5h	
	AL =	0	Query support.
Output:	AH =	1	Supported.
	AH <>	1	Not supported.

Input:	AH =	12h	
	BL =	A5h	
	AL =	1	Process fixup.
Output:	AX =		Undefined.

6.7.11 Frame Buffer Set/Get Physical Address

This function sets or gets, respectively, the physical address of the frame buffer. In a VESA VL-Bus system, the Get/Put function does not work under Windows 3.1 or Windows 95. Use A7 to get the register address, then read the memory address from the registers (BASE1_ADDRESS 0x314).

Input:	AH=	12h	
	BL=	A6h	
	BH =	1h	Set.
	DX:CX =		Physical address of frame buffer.
Output:	None		

Input:	AH=	12h	
	BL=	A6h	
	BH =	0h	Get
Output:	DX:CX =		Physical address of frame buffer.

6.7.12 Set/Get Memory-Mapped Registers

This function sets or gets, respectively, the memory-mapped register physical address.

Input:	AH =	12h	
	BL =	A7h	
	BH =	1h	Set
	DX:CX =		Physical address of memory-mapped registers.
Output:	None		

Input:	AH =	12h	
	BL =	A7h	
	BH =	0h	Get
Output:	DX:CX =		Physical address of memory-mapped registers.

6.7.13 Enable_Tiled_Mode

This function puts the RDRAM into tiled mode. As the driver or application requires, this function is called after a mode change call to switch the memory from linear to tiled mode. High-performance drivers use tiled mode, whereas DOS VGA and VESA/VBE applications do not. This does not setup 2D Engine registers.

Input:	AH =	12h	
	BL =	B3h	
	Global variable 0:449 is consulted for Video mode.		
Output:	AL =		Tile size
		1 =	128
		2 =	256
	AH =		Mode or FF if error.
	BL =		TPL
	BH =		Interleave
		0 =	1-way
		1 =	2-way
		2 =	4-way

6.7.14 Get FIFO/Format

This function retrieves the FIFO/graphics part of Graphics/Video Format register (offset C0h).

Input:	AH =	12h	
	BL =	B5h	
Output:	AH =	FIFO	FIFO
	AL =		Graphics half of format

6.8 Extended Modes in RAM

6.8.1 Extensions to the Save Area Table

Cirrus Logic BIOS (standard version) support VGA compatible modes along with a set of extended modes. OEMs may add new modes to the system, or redefine existing modes that are in extended modes. OEMs may add new modes to the system, or redefine existing modes that are in the VGA ROM by manipulating the BIOS save area table pointed to by 0040:00a8. This table is located in ROM after the system is booted. Any changes must be made in a RAM copy. Cirrus Logic has extendeed the definition of this table with 'negative' offsets that point to Cirrus Logic defined parameters. Table 6-20 presents the compatible table and the defined extensions.

Table 6-20. Extensions to the Save Area Table

Offset	Type	Description
–14h	dword	Pointer to next negative offset table in linked list
–10h	word	Set to 04h if offset –14h is valid pointer, set to 00h if this link is the last in RAM. To block all ROM based modes, set this field to 04h, and offset –14 to 0:0
–0Eh	word	Size of supplemental table
–0Ch	dword	Pointer to extended mode supplemental parameters
–08h	dword	Pointer to extended mode standard parameters
–04h	word	Number of extended video modes
–02h	word	'RV' identifier
00h	dword	Pointer to standard mode standard parameters
04h	dword	Dynamic saver area pointer (palette save area)
08h	dword	Alpha mode auxiliary character generator pointer
0Ch	dword	Graphics mode auxiliary character generator pointer
10h	dword	Secondary save pointer
14h	dword	Reserved and set to '0'
18h	dword	Reserved and set to '0'

6.9 BIOS Processing

The Cirrus Logic BIOS determines the mode to select by processing a linked list of extended mode supplemental parameter tables, at the same time evaluating several factors such as, memory size, monitor type, memory clock and the associated chip set. Travelling from the top down, the BIOS services a mode set request once all factors are satisfied. A mode that has multiple horizontal frequencies must be sequentially ordered from the highest frequency at the top, to the lowest at the bottom. This ensures that BIOS always sets the correct mode for the given monitor type.

Modes can be added to the BIOS by manipulating the structure described above. It always looks for RAM-defined links first to satisfy a mode set request. If it cannot find a mode based on the current configuration of the video subsystem, the ROM tables are then scanned.

If new modes are to be *added* to the BIOS by defining them in RAM, a TSR need only modify the negative offsets described above (that is, higher refresh rates of the previously defined mode or entirely new mode numbers). If modes are to be *redefined*, special care must be taken. If a TSR needs to modify a particular frequency of a mode that has higher frequencies already defined in ROM, all frequencies must be redefined in RAM.

6.10 Extended Mode Supplemental Parameters

Table 6-21 describes what the BIOS expects in the supplemental structure discussed above. The extended mode supplemental parameters table is divided into two physical tables to conserve space. The format of the tables are shown below. The tables are subject to change without notice.

Table 6-21. Extended Mode Supplemental Parameters Table 1

Offset	Size	Description
00	byte	Video mode number
01	word	VESA video mode number
03	word	Horizontal resolution
05	word	Vertical resolution
07	byte	Bits per color
08	byte	Character width
09	byte	Character height
0A	byte	VESA memory model (Define in VESA function 1)
0B	byte	VESA mode attributes (Defined in VESA function 1)
0C	byte	Standard Parameter List Index
0D	byte	Memory required, in 64-Kbyte blocks specified for tiled mode
0E	byte	TPL
0F	byte	Tiled FIFO
10	byte	Linear FIFO
11	word	Offset into Table 2

Table 6-21. Extended Mode Supplemental Parameters Table 1 *(cont.)*

Offset	Size	Description
13	byte	Number of Entries
14	byte	SR07 8-bit Packed-Pixel{0}/VS Control register (Linear)
15	byte	CR1D Screen Start Extension register
16	byte	CR1B Extended Display Controls register (Linear)
17	byte	CR13 Offset register (Linear)
18	byte	CR1B Extended Display Controls register (Tiled)
19	byte	CR13 Offset register (Tiled)

Table 6-22. Extended Mode Supplemental Parameters Table 2

Offset	Size	Description
00	byte	Refresh
01	byte	Refresh Index
02	byte	Miscellaneous Output Value
03	byte	Format register high byte
04	byte	Monitor List
05	byte	SR0E VCLK3 Numerator
06	byte	SR1E VLCK3 Denominator and Post Scalar
07	byte	CR19 Interlace End
08	byte	CR1A Miscellaneous Control register
0A	byte	CR1E Timing Overflow register