OC69030

VGA BIOS OEM
Reference Guide

Revision 1.0

July 1999

PRELIMINARY

CHIPS

Copyright Notice

Restricted Rights Legend

Trademark Acknowledgment

Disclaimer

This document provides general information for the customer. Intel Corporation reserves the right to modify the information contained herein, as necessary and the customer should ensure that it has the most recent revision of the document. Intel makes no warranty for the use of its products and bears no responsibility for any errors, which may appear in this document. The customer should be on notice that many different parties hold patents on products, components, and processes within the personal computer industry. Customers should ensure that their use of the products does not infringe upon any patents. Intel respects the patent rights of third parties and shall not participate in direct or indirect patent infringement.

Revision History

Revision	Date	By	Comment
1.0	7/2/99	JC/dak	First draft - initial release

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

This document covers the standard video BIOS interfaces, function definitions, and supported modes. This document also covers the external interfaces to the video BIOS, the interrupt 10h-function descriptions, and additional interface components. Legacy VGA and standard external VESA interface chapters have been added for reference. Refer to the original documentation for official interface definitions. Deviations from the standard specifications will be noted in these chapters.

1.2 Scope

BIOS initializes the hardware by setting registers in required sequences to put the hardware in a known state. BIOS allows other software to set hardware states by offering an interface to BIOS functions that load registers. BIOS provides work-arounds for hardware bugs.

The video BIOS is traditionally located at physical memory location C000: 0000h and uses the interrupt 10h vector. On mobile systems however the location of the video BIOS ROM is anywhere from C000: 0000h to 0EFFF:FFFFh.

1.3 About this Manual

This is the reference document for the video BIOS architecture, functionality, and interfaces. An outline of this document follows:

Chapter 1 – Introduction: Introduces the Video BIOS documents and details BIOS support.

Chapter 2 – Model: Describes how the video BIOS fits in with other software products.

Chapter 3 – BIOS Features: Describes many of the features of the video BIOS including compatibility, flat panel, PCI and others.

Chapter 4 – Video Modes: Lists the many different video modes supported by the video BIOS. Along with the mode is an array of data for each supported mode.

Chapter 5 – Legacy VGA Interface Functions: Describes many of the legacy standard VGA interfaces put in place by IBM VGA BIOS.

Chapter 6 – Extended Interface Functions: Presents a detailed description of the extended video BIOS interface functions.

Chapter 7 – VESA Interface Functions: Documents the supported VESA functions for reference including VBE, VBE / DDC and VBE / PM.

Chapter 8 – Hooks for System BIOS: Lists the interrupt 15h hooks that are called by the video BIOS that allow the system BIOS temporary control.

Chapter 9 – OEM Utility Programs: Introduces and describes how to use some of the OEM utility program that are used with the video BIOS.

Appendix A – Building the video BIOS: This appendix describes how to build the video BIOS if source code was purchased.

Appendix B – Suspend / Resume Procedures: Presents procedures on suspending and resuming the video controller.

1.3.1 Definitions, Acronyms, and Abbreviations

Acronym	Description
ACPI	Advanced Configuration and Power Interface – Normally handled by the operating system and system BIOS.
APM	Advanced Power Management – Normally handled by the system BIOS through a SMI interrupt.
BIOS	Basic Input Output System
DDC	Display Data Channel – VESA standard used to retrieve EDID data from a monitor.
DPMS	Display Power Management Signaling – Hardware mechanism used to save power used by CRT monitor.
DSP	Display Property Sheet (Control Panel)
EDID	Extended Display Identification Data – Monitor data that describes the monitor characteristics.
MDS	Multiple Display Support
VESA PM	VESA Power Management – VESA software standard to control a DPMS
POST	Power On Self-Test – Chipset initialization code.
VBE	VESA BIOS Extensions
VESA	Video Electronics Standards Association
VGA	Video Graphics Adapter

1.3.2 References

This section provides a complete list of all documents referenced elsewhere in the document. The internal document reference should be listed.

1. IBM VGA Reference Manual
2. Advanced Programmer’s Guide to the EGA/VGA, George Sutty
3. VESA VBE Specification – Version 2.0, November 18, 1994
4. VESA/DDC Specification – Version 1.0, August 12, 1994
5. VESA EDID Specification – Version 3.0, November 13, 1997
6. VESA/PM Specification – Version 1.0, February 4, 1994
7. PCI Local Bus, Revision 2.1, June 1, 1995

1.4 Introducing 69030 Mobile Video BIOS

The mobile video BIOS is an enhanced, high performance BIOS that is used with video Flat Panel / CRT Controllers to provide a competitive integrated solution. The BIOS is optimized for Intel's flat panel / CRT Controllers and provides:

- Compatibility with the IBM VGA BIOS.
- Supports VESA BIOS Extensions, including VBE 2.0, VBE / DDC 1.0, and VBE / PM 1.0.
- Allows object code BIOS modification through a simple to use BIOS Modification Program (BMP).
- Supports high-resolution modes 640x480 @ 100Hz, 800x600 @ 100Hz, 1024x768 @ 100Hz, 1280 x1024 @ 75Hz and 1600x1200 @ 60Hz. Includes 8bpp, 15bpp, 16bpp and 24bpp modes.
- Supports low-resolution modes 320x200 @ 70Hz, 320x240 @ 60Hz, 400x300 @ 60Hz, 512x384 @ 60Hz, and 640x400 @ 70Hz. All low-resolution modes have the 8bpp, 16bpp and 24bpp versions.
- Allows a generic mode resolution by updating tables in the BIOS with the BMP utility.
- Support for monochrome LCD, 640x480 STN, or TFT, 800x600 STN or TFT, 1024x768 STN or TFT, and 1280x1024 STN or TFT flat panel displays. Optional support is available for other displays.
- 32K BIOS supports eight panels.
- Large BIOS supports 16 panels.
- Support for simultaneous display modes.
- Allows different memory clock settings for flat panel only display device to save power.
- Support Brooktree digital TV chipset.
- Supports popup icon.
- Supports PCI bus and I^{2}C bus.
- Allows the run time BIOS size to be adjusted to the nearest granularity unit. The new size can be smaller (removing POST and white space) or larger (allowing for shadow granularities).
- BIOS segment can be relocated on 32K boundaries between C000h to F800h using the BMP utility.
- Interrupt 15h hooks throughout the video BIOS to give system BIOS control at essential video BIOS locations.
- Extended BIOS functions that offer easy access to controller features and capabilities.

1.5 BIOS Kits

The 69030 BIOS is available in three kit formats. These kits and their contents are as follows:

SE69030 VGA BIOS Evaluation Kit:

- Evaluation diskette (Evaluation copy of BIOS and utility programs)
- *OC69030 VGA BIOS OEM Reference Guide*
- Release notes
- Software Incident Report (SIR) forms

SK69030 VGA BIOS Binary Kit:

- Binary diskette (Master copy of BIOS and utility programs)
- *OC69030 VGA BIOS OEM Reference Guide*
- Release notes
- Software Incident Report (SIR) forms

SC69030 VGA BIOS Source Code Kit:

- Source code diskette
- SK69030 VGA BIOS binary kit

1.6 Customer Support

Software products are supported by field application engineers located in each sales office. If you encounter a problem, or have any questions regarding a software product, please complete a copy of the Software Incident Report (SIR) form included with your product. Forward the completed form to the following address:

Intel Corporation
Mail Stop: CHP3-202
Attn.: Software Product Support
350 East Plumeria Drive
San Jose, CA 95134

FAX SIR forms to: (408) 545-9817

2 Model

2.1 Overview

The Video BIOS may be thought of as an operating system with an independent hardware abstraction layer. It does not prevent or monitor direct access to hardware by operating system, applications, or device drivers. Though not recommended, some DOS applications do change basic hardware settings without going through video BIOS at all. Most modern day applications and operating systems avoid going directly to hardware. They could use the legacy VGA entry point for basic functionality, which is the lowest common denominator, or they could use the VESA entry point for accessing the extended functions without the support of device specific drivers. They might take advantage of the functionality of device specific drivers to access all features of the hardware. In turn, the device specific driver may access the hardware directly or may access any of the entry points the video BIOS makes available.

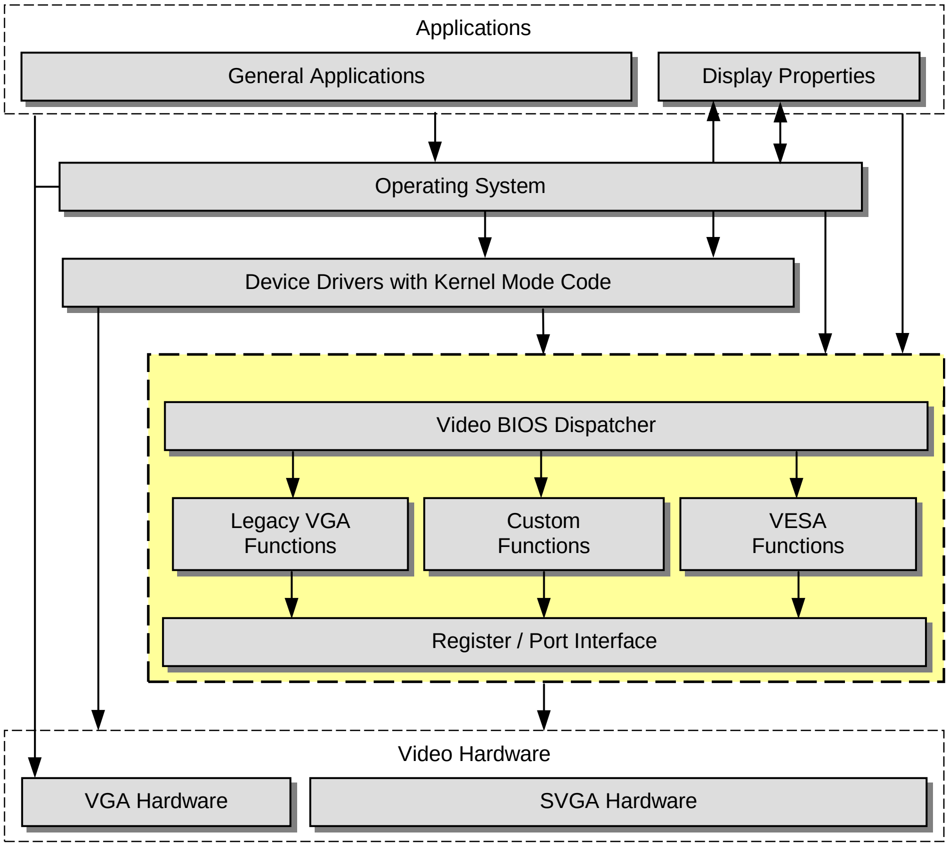


Figure 2-1: Video BIOS Model

This page intentionally left blank

3 BIOS Features

3.1 Compatibility

This chapter discusses the compatibility issues of modern video BIOS systems. The importance of compatibility is immeasurable to manufacturers introducing new products to the marketplace.

3.1.1 VGA Compatibility

The video BIOS is fully compatible with the documented standard IBM VGA BIOS, including standard VGA functions, register setting, mode resolutions, and RAM data area values. IBM VGA compatibility assures that the video BIOS will support legacy software as well as today's software that still use the VGA interface.

3.1.2 Industry Compatibility

Industry compatibility is just as important, if not more so, than IBM VGA compatibility. This compatibility is achieved by testing the video BIOS on current and legacy industry platforms using current software packages. As the industry pushes forward with today's most demanding platforms and software packages, our compatibility labs develop new innovative ways to assure a robust video BIOS product.

3.1.3 VESA Compatibility

VESA is a well-known standards committee that is composed of representatives from many fast paced companies. The standards committee develops helpful standards that are used when developing products throughout the computer industry. VESA first gained its recognition with its VESA VBE standard that allowed software to set non-IBM standard mode resolutions. The video BIOS currently supports VESA VBE 2.0, VESA VBE / DDC, and VESA VBE / PM standards (see VESA chapters for details page 7-1).

3.2 Flat Panel Support

The video BIOS provides support for features used in conjunction with a flat panel display.

3.2.1 Supported Flat Panel Types

The video BIOS supports 16 flat panel classes. These panel classes are listed in table groups in the BIOS referenced by a number. They are identified as panel classes because they are made up of register tables that will set the most popular panel in a given panel class (resolution and type). These registers tables may have to be updated with the BIOS modification utility for some flat panel manufactures. By adjusting all the registers in a panel class, the panel number can be made to support any panel resolution and type supported by the video controller.

The 32 K BIOS only supports the first eight panel classes. The following is a list of the default panel numbers and the associated panel class.

Table 3.1 Default Supported Panel Classes

Panel #	Panel Class
1	1024x768 Dual Scan STN Color Panel
2	128x1024 TFT Color Panel
3	640x480 Dual Scan STN Color Panel
4	800x600 Dual Scan STN Color Panel
5	640x480 Sharp TFT Color Panel
6	640x480 18-bit TFT Color Panel
7	1024x768 TFT Color Panel
8	800x600 TFT Color Panel
9	800x600 TFT Color Panel (Large BIOS only)
10	800x600 TFT Color Panel (Large BIOS only)
11	800x600 Dual Scan STN Color Panel (Large BIOS only)
12	800x600 Dual Scan STN Color Panel (Large BIOS only)
13	1024x768 TFT Color Panel (Large BIOS only)
14	1280x1024 Dual Scan STN Color Panel (Large BIOS only)
15	1024x600 Dual Scan STN Color Panel (Large BIOS only)
16	1024x600 TFT Color Panel (Large BIOS only)

3.2.2 Vertical Compensation

The video BIOS supports the following vertical compensation modes for flat panel operation:

None	Image is top justified.
Automatic Centering	Image is automatically centered vertically.
Tall Font/Text Compensation	Text is compensated by stretching the font in the hardware.
Line Replication/Graphic Compensation	Line replication stretches graphics image to fill the display.

The vertical compensation can be set by using function 5F5Eh (Set Vertical Compensation). Vertical Compensation status can be read by using function 5F50h (See Return Panel Information) or 5F61h (Set Horizontal & Vertical Compensation).

3.2.3 Horizontal Compensation

The video BIOS supports the following horizontal compensation modes for flat panel operation:

None	Image is left justified.
Automatic Centering	Image is automatically centered horizontally.
Text Compression	720 dot wide applications can be compressed to fit on 640 horizontal resolution panels by either adding the eighth and ninth pixels or deleting the ninth pixel.
Automatic Horizontal Expansion	640/800 dot wide images can be automatically expanded to fill 800/1024 dot wide flat panels.

The horizontal compensation can be set with function 5F5Bh or 5F61h.

3.3 Extended Save and Restore

The video BIOS provides functions to save and restore the video Chipset State. This includes all standard and extended registers, the memory latches, and the attribute Flip/flop State. The functions provided are 5FA0h (Extended BIOS Save/Restore State), 5FA1h (Save Video State), and 5FA2h (Restore Video State).

3.4 SMI and Hot Key Support

An alternate INT 10h entry point (word pointer) is located at 8Bh in the BIOS that will bypass the STI instruction at the beginning of the usual INT 10h handler. STI instructions are not allowed during processing of a System Management Interrupt (SMI).

The video BIOS and Flat Panel Windows drivers are designed to support display switching with hot keys. The following paragraphs describe how to use the video BIOS to implement hot key display switching. The system BIOS hot key function handler should call the video BIOS switch display function (INT 10h, function 5F51h) when the switch display hot key is pressed.

3.5 Hardware Pop-Up Window Interface

The video BIOS has the capability of overlaying a 32x32 / 64x64 / 128x128 area of screen with the off-screen data stored in different formats. The off-screen data could be an AND / XOR format cursor (Windows or OS/2), or a monochrome 2 bit per pixel format bit map. The 69030 BIOS provides the pop-up support under SMI through the 5F14H function. The BIOS supports up to eight pop-ups.

3.6 PCI Support

3.6.1 Video BIOS

The video BIOS is developed for use with PCI Local Bus board configurations as defined in the PCI Local Bus Specification. The BIOS has word pointers to the PCI Data Structure at offset C000:18h / E000:18h. The PCI Data Structure is defined as follows:

Table 3.1 PCI Data Structure

Offset	Length	Data	Description
0	4	PCIR	Signature
4	2	102Ch	Vendor ID
6	2	0C30h	Device ID
8	2	0000h	Pointer to vital product data
A	2	0018h	PCI data structure length
C	1	00h	PCI structure rev
D	3	00h, 00h, 03h	Class code
10	2	0040h / 058h	Image Length 32K / 44K
12	2	0000h	Rev level of code/data
14	1	00h	Code Type
15	1	08h	Indicator
16	2	0000h	Reserved

3.6.2 System BIOS

The Flat Panel / CRT controller supports expansion ROM Base Address at offset 30h in the configuration space. The video BIOS is usually merged with the system BIOS and is located at address E000:0 in the system. To find the video device during power up, the system BIOS reads class code at offset 09h (00h, 00h, 03h) in the configuration space. The system BIOS then looks for PCIR signatures in the C000 / E000 segment (word pointer to the PCIR string is at C000:18h / E000:18h) followed by the video device class code (00h, 00h, 03h) at offset 0Dh in the PCI data structure of the video BIOS. When the system BIOS finds the video device, it should map the video BIOS at a very high address, then copy the video BIOS at Address C000:0 / E000:0.

3.7 Major BIOS Component Changes

The 69030 dual pipe architecture will require many BIOS component changes from today's 69000 BIOS including video POST, set mode, switch display device, and others. The following sections will address affected portions of these components.

3.7.1 Video POST

The initial part of POST (Power On Self-Test) loads the majority of the registers in table form to put the chipset in a know state. This known state will include a display device of CRT on pipe A with pipe B off.

After the chipset is in a know state, POST will determine the boot display configuration, size video memory, and so on. The main change in POST will be to support the new boot display devices. These devices will be more complicated as shown with a list of the current and the new possible boot display devices.

The current 69000 boot display devices are listed below:

- CRT
- Flat Panel
- Simultaneous (CRT and Flat Panel)
- Flat Panel or CRT if a CRT is detected
- Flat Panel or Simultaneous if a CRT is detected

The new 69030 dual pipe boot display devices are as follows:

• Pipe A: CRT,	Pipe B: Off
• Pipe A: Flat Panel (FP),	Pipe B: Off
• Pipe A: TV Digital (TVd),	Pipe B: Off
• Pipe A: Simultaneous (SM = CRT and FP),	Pipe B: Off
• Pipe A: Simultaneous (SM = CRT and TVd),	Pipe B: Off
• Pipe A: FP or CRT if a CRT is detected,	Pipe B: Off
• Pipe A: FP or SM if a CRT is detected,	Pipe B: Off
• Pipe A: Flat Panel,	Pipe B: CRT
• Pipe A: TV Digital,	Pipe B: CRT
• Pipe A: Off,	Pipe B: Tva
• Pipe A: Flat Panel,	Pipe B: Tva

Currently, booting into mosaic mode is not supported.

3.7.2 Set Mode

Externally, the set mode function (interrupt 10h, AH = 00h) will work fine as long as the correct pipe I/O port access is selected before set mode is called. A function to set the pipe I/O port access will be discussed in the New Software Components section of this document. Internally, set mode will have to adjust many registers depending on the display device.

Only one of the pipelines may display a text mode at any one time. This limitation is the result of only having one BIOS data area (address 40:XX) as discussed in the next paragraph. It is, however, possible to run the same text mode on both pipelines.

There will only be a single copy of the BIOS data area. The data will reflect the pipe that is in text mode or last in text mode. Set mode must not write to or read from the BIOS data area if the pipe without I/O port access is in control of the BIOS data area.

Output functions are the IBM standard functions that write text or pixel data to the screen. The output functions may need to be blocked if the current pipe with I/O port access is not the pipe that controls the BIOS data area variables.

3.7.3 Software Flag (Scratch) Registers

The BIOS uses software flags to hold current states, to allow or disallow certain functionality, and to interface with the device drivers. Software flags are simply chipset registers with no hardware function except to hold a value (a flag) that can be retrieved by reading back the register. The following table defines the use of these flags, but these flags change from time to time as BIOS development continues.

Note: The values and location of the data in the software flag registers may change at any time. They are listed here for information only.

3.7.3.1 69030 BIOS / Driver Interface Scratch Bits

The 69030 BIOS and Device Drivers interface scratch registers will use shadowed versions for each pipe. The following tables list the scratch bits used in the BIOS / Driver interface for both pipeline A and B.

Table 3.1 Pipe A – BIOS / Driver Interface Software Flags

Flag Register	Flag State	Flag Description
XRE0		
[7:0]	xxxxxxxx	Reserved for future use.
XRE1		
[7]	x	Reserved for future use.
[6]	x	The BIOS sets this flag when PAL mode is requested. PAL mode is activated when this flag is set and the current state is a valid PAL state (i.e. CRT display, modes 03h, 12h, 13h, or extended mode). The drivers use this bit to determine if CRT panning may be required.
[5]	x	The BIOS sets this flag when NTSC mode is requested. NTSC mode is activated when this flag is set and the current state is a valid NTSC state (i.e. CRT display, modes 03h, 12h, 13h, or extended mode). The drivers use this bit to determine if CRT panning may be required.
[4]	x	The BIOS sets this bit to inform the drivers to use software cursor. Note: The BIOS will use IOSS to determine whether the pipe A or pipe B version of this bit will be updated.
[3:2]		Reserved for future use.
[1:0]	 00 01 10 11	The BIOS sets these bits to inform the drivers that a state change may have taken place. This state change may include a display device switch, a mode set, a refresh rate change, a stretching state change, or other state changes that may need the drivers to update its environment. After the drivers process these bits, the driver will set them to 00. No state change since bits last processed State changed – Current display type is pipe A, CRT State changed – Current display type is pipe A, flat panel or TV Digital State changed – Current display type is pipe A, CRT + FB or CRT & TVD Note: The BIOS will use IOSS to determine whether the pipe A or pipe B version of this bit will be updated.
XRE2		
[7:0]	XXh	The BIOS sets the current video mode number for pipe A in this register. Both the BIOS and drivers use this as the actual current mode number. This is required since some systems (Japanese windows) change the mode number in 40:49.

Table 3.2 Pipe B – BIOS / Driver Interface Software Flags

Flag Register	Flag State	Flag Description
XRE0		
[7:0]	xxxxxxxx	Reserved for future use.
XRE1		
[7]	x	Reserved for future use.
[6]	x	The BIOS sets this flag when PAL mode is requested. PAL mode is activated when this flag is set and the current state is a valid PAL state (i.e. CRT display, modes 03h, 12h, 13h, or extended mode). The drivers use this bit to determine if CRT panning may be required.
[5]	x	The BIOS sets this flag when NTSC mode is requested. NTSC mode is activated when this flag is set and the current state is a valid NTSC state (i.e. CRT display, modes 03h, 12h, 13h, or extended mode). The drivers use this bit to determine if CRT panning may be required.
[4]	x	The BIOS sets this bit to inform the drivers to use software cursor. Note: The BIOS will use IOSS to determine whether the pipe A or pipe B version of this bit will be updated.
[3:2]		Reserved for future use.
[1:0]		The BIOS sets these bits to inform the drivers that a state change may have taken place. This state change may include a display device switch, a mode set, a refresh rate change, a stretching state change, or other state changes that may need the drivers to update its environment. After the drivers process these bits, the driver will set them to 00.
	00	No state change since bits last processed
	01	State changed – Pipe B, CRT or TV Analog
	10	State changed – Reserved
	11	State changed – Reserved
		Note: The BIOS will use IOSS to determine whether the pipe A or pipe B version of this bit will be updated.
XRE2		
[7:0]	XXh	The BIOS sets the current video mode number for pipe B in this register. Both the BIOS and drivers use this as the actual current mode number. This is required since some systems (Japanese windows) change the mode number in 40:49.

This page intentionally left blank.

4 Video Modes

The BIOS set mode function supports all standard VGA mode resolutions as well as a wide selection of extended mode resolutions. The following tables list the modes and vertical refresh rates for this BIOS.

The actual availability of any particular mode, however, depends on the capabilities of the display device, the amount of memory installed, the MCLK setting, and other system parameters.

Table 4.1 Standard Video Display Modes

Video Mode	VESA VBE Mode	Pixel Resolution	Color Depth	Mode Type	Display Adapter	Font Size	Character Resolution	Dot Clock (MHz)	Horiz. Freq. (KHz)	Vert. Freq. (Hz)	Video Memory (KB)
00h	—	320x200 320x350 360x400	16 (gray) 16 (gray) 16	Text	CGA EGA VGA	8x8 8x14 9x16	40x25 40x25 40x25	25 25 28	31.5 31.5 31.5	70 70 70	256 256 256
01h	—	320x200 320x350 360x400	16 16 16	Text	CGA EGA VGA	8x8 8x14 9x16	40x25 40x25 40x25	25 25 28	31.5 31.5 31.5	70 70 70	256 256 256
02h	—	640x200 640x350 720x400	16 (gray) 16 (gray) 16	Text	CGA EGA VGA	8x8 8x14 9x16	80x25 80x25 80x25	25 25 28	31.5 31.5 31.5	70 70 70	256 256 256
03h	—	640x200 640x350 720x400	16 16 16	Text	CGA EGA VGA	8x8 8x14 9x16	80 x 25 80 x 25 80 x 25	25 25 28	31.5 31.5 31.5	70 70 70	256 256 256
04h	—	320x200	4	Graph	All	8x8	40x25	25	31.5	70	256
05h	—	320x200 320x200 320x200	4 (gray) 4 (gray) 4	Graph	CGA EGA VGA	8x8 8x8 8x8	40x25 40x25 40x25	25 25 25	31.5 31.5 31.5	70 70 70	256 256 256
06h	—	640x200	2	Graph	All	8x8	80x25	25	31.5	70	256
07h	—	720x350 720x350 720x400	Mono Mono Mono	Text	MDA EGA VGA	9x14 9x14 9x16	80x25 80x25 80x25	28 28 28	31.5 31.5 31.5	70 70 70	256 256 256
08h – 0Ch	—	Reserved			-		-				
0Dh	—	320x200	16	Graph	E/VGA	8x8	40x25	25	31.5	70	256
0Eh	—	640x200	16	Graph	E/VGA	8x8	80x25	25	31.5	70	256
0Fh	—	640x350	Mono	Graph	E/VGA	8x14	80x25	25	31.5	70	256
10h	—	640 x 350	16	Graph	E/VGA	8x14	80x25	25	31.5	70	256
11h	—	640x480	2	Graph	VGA	8x16	80x30	25	31.5	60	256
12h	—	640x480	16	Graph	VGA	8x16	80x30	25	31.5	60	256
13h	—	320x200	256	Graph	VGA	8x8	40x25	25	31.5	70	256

Table 4.2 Extended Low Resolution Video Modes

Video Mode	VESA VBE Mode	Pixel Resolution	Colors	Mode Type	Memory Orgn	Font Size	Character Resolution	Dot Clock (MHz)	Horiz. Freq. (KHz)	Vert. Freq. (Hz)	Video Memory (KB)
14h	—	320x200	256	Graph (L)	Pack Pix	8x16	40x12	12.587	31.5	70	256
15h	—	320x200	64K	Graph (L)	Pack Pix	8x16	40x12	12.587	31.5	70	256
16h	—	320x200	16M	Graph (L)	Pack Pix	8x16	40x12	12.587	31.5	70	256
17h	—	320x240	256	Graph (L)	Pack Pix	8x16	40x15	12.587	31.5	60	256
18h	—	320x240	64K	Graph (L)	Pack Pix	8x16	40x15	12.587	31.5	60	256
19h	—	320x240	16M	Graph (L)	Pack Pix	8x16	40x15	12.587	31.5	60	256
1Ah	—	400x300	256	Graph (L)	Pack Pix	8x16	50x18	20	37.5	60	256
1Bh	—	400x300	64K	Graph (L)	Pack Pix	8x16	50x18	20	37.5	60	256
1Ch	—	400x300	16M	Graph (L)	Pack Pix	8x16	50x18	20	37.5	60	352
1Dh	—	512x384	256	Graph (L)	Pack Pix	8x16	64x24	32.5	48.4	60	256
1Eh	—	512x384	64K	Graph (L)	Pack Pix	8x16	64x24	32.5	48.4	60	384
1Fh	—	512x384	16M	Graph (L)	Pack Pix	8x16	64x24	32.5	48.4	60	576
31h	100h	640x400	256	Graph (L)	Pack Pix	8x16	80x25	25.175	31.5	70	256
61h	—	640x400	64K	Graph (L)	Pack Pix	8x16	80x25	25.175	31.5	70	500
62h	—	640x400	16M	Graph (L)	Pack Pix	8x16	80x25	25.175	31.5	70	750

Notes: I = Interlaced; L = Linear

Table 4.3 Extended Video Modes

Video Mode	VESA VBE Mode	Pixel Resolution	Colors	Mode Type	Memory Org	Font Size	Character Resolution	Dot Clock (MHz)	Horiz. Freq. (KHz)	Vert. Freq. (Hz)	Video Memory (KB)
30h	101h	640x480	256	Graph (L)	Pack Pix	8x16	80x30	25.175 31.5 36 46	31.5 37.5 43.3 53.2	60 75 85 100	300 300 300 300
31h	100h	640x400	256	Graph (L)	Pack Pix	8x16	80x25	25.175	31.5	70	256
32h	103h	800x600	256	Graph (L)	Pack Pix	8x16	100x37	40 49.5 56.25 74	37.9 46.9 53.7 66.1	60 75 85 100	469 469 469 469
34h	105h	1024x768	256	Graph (L)	Pack Pix	8x16	128x48	44.9 65 78.75 94.5 121	35.5 48.4 60 68.7 84	43 (I) 60 75 85 100	768 768 768 768 768
36h	—	Generic	256	Graph (L)	Pack Pix	8 x 16	—	— — — —	— — — —	— — — —	— — — —
38h	107h	1280x1024	256	Graph (L)	Pack Pix	8x16	160x64	78.75 108 135	47 64 79.98	43 (I) 60 75	1280 1280 1280
3Ah	—	1600x1200	256	Graph (L)	Pack Pix	8x16	200x75	162	75	60	1875
40h	110h	640x480	32K	Graph (L)	Pack Pix	8x16	80x30	25.175 31.5 36 46	31.5 37.5 43.3 53.2	60 75 85 100	600 600 600 600

Video Mode	VESA VBE Mode	Pixel Resolution	Colors	Mode Type	Memory Org	Font Size	Character Resolution	Dot Clock (MHz)	Horiz. Freq. (KHz)	Vert. Freq. (Hz)	Video Memory (KB)
41h	111h	640x480	64K	Graph (L)	Pack Pix	8x16	80x30	25.175 31.5 36 46	31.5 37.5 43.3 53.2	60 75 85 100	600 600 600 600
42h	113h	800x600	32K	Graph (L)	Pack Pix	8x16	100x37	40 49.5 56.25 74	37.9 46.9 53.7 66.1	60 75 85 100	938 938 938 938
43h	114h	800x600	64K	Graph (L)	Pack Pix	8x16	100x37	40 49.5 56.25 74	37.9 46.9 53.7 66.1	60 75 85 100	938 938 938 938
44h	116h	1024x768	32K	Graph (L)	Pack Pix	8x16	128x48	44.9 65 78.75 94.5 121	35.5 48.4 60 68.7 84	43 (I) 60 75 85 100	1536 1536 1536 1536 1536
45h	117h	1024x768	64K	Graph (L)	Pack Pix	8x16	128x48	44.9 65 78.75 94.5 121	35.5 48.4 60 68.7 84	43 (I) 60 75 85 100	1536 1536 1536 1536 1536
46h	—	Generic	32K	Graph (L)	Pack Pix	8x16	—	— — — —	— — — —	— — — —	— — — —
47h	—	Generic	64K	Graph (L)	Pack Pix	8x16	—	— — — —	— — — —	— — — —	— — — —
48h	119h	1280x1024	32K	Graph (L)	Pack Pix	8x16	160x64	78.75 108 135	47 64 79.98	43 (I) 60 75	2560 2560 2560
49h	11Ah	1280x1024	64K	Graph (L)	Pack Pix	8x16	160x64	78.75 108 135	47 64 79.98	43 (I) 60 75	2560 2560 2560
50h	112h	640x480	16M	Graph(L)	Pack Pix	8x16	80x30	25.175 31.5 36 46	31.5 37.5 43.3 53.2	60 75 85 100	900 900 900 900
52h	115h	800x600	16M	Graph(L)	Pack Pix	8x16	100x37	40 49.5 56.25 74	37.9 46.9 53.7 66.1	60 75 85 100	1407 1407 1407 1407
54h	118h	1024x768	16M	Graph(L)	Pack Pix	8x16	128x48	44.9 65 78.75 94.5 121	35.5 48.4 60 68.7 84	43 (I) 60 75 85 100	2304 2304 2304 2304 2304
56h	—	Generic	16M	Graph (L)	Pack Pix	8x16	—	— — — —	— — — —	— — — —	— — — —
58h	11Bh	1280x1024	16M	Graph(L)	Pack Pix	8x16	160x64	78.75 108 135	47 64 79.98	43 (I) 60 75	3840 3840 3840

Video Mode	VESA VBE Mode	Pixel Resolution	Colors	Mode Type	Memory Org	Font Size	Character Resolution	Dot Clock (MHz)	Horiz. Freq. (KHz)	Vert. Freq. (Hz)	Video Memory (KB)
6Ah	102h	800x600	16	Graph	Planar	8x16	100x37	40 49.5 56.25 74	37.9 46.9 53.7 66.1	60 75 85 100	256 256 256 256
64h	104h	1024x768	16	Graph	Planar	8x16	128x48	44.9 65 78.75 94.5 121	35.5 48.4 60 68.7 84	43 (I) 60 75 85 100	384 384 384 384 384
68h	106h	1280x1024	16	Graph	Planar	8x16	160x64	78.75 108 135	47 64 79.98	43 (I) 60 75	640 640 640
70h	101h	640x480	256	Graph	Pack Pix	8x16	80x30	25.175 31.5 36 46	31.5 37.5 43.3 53.2	60 75 85 100	300 300 300 300
71h	100h	640x400	256	Graph	Pack Pix	8x16	80x25	25.175	31.5	70	256
72h	103h	800x600	256	Graph	Pack Pix	8x16	100x37	40 49.5 56.25 74	37.9 46.9 53.7 66.1	60 75 85 100	469 469 469 469
74h	105h	1024x768	256	Graph	Pack Pix	8x16	128x48	44.9 65 78.75 94.5 121	35.5 48.4 60 68.7 84	43 (I) 60 75 85 100	768 768 768 768 768
78h	107h	1280x1024	256	Graph	Pack Pix	8x16	160x64	78.75 108 135	47 64 79.98	43 (I) 60 75	1280 1280 1280

Notes: (I) = Interlaced (L) = Linear.

5 Legacy VGA Interface Functions

The video BIOS supports the original documented IBM VGA display adapter video BIOS functions. These functions have been included in this chapter for convenience as a video BIOS reference manual. Please consult an IBM reference manual for the exact definitions. The interface of some functions has been updated to include extended hardware features (i.e. Save and Restore functions).

Important Note: IOSS (see function 5F1Ch) must be set to the appropriate values before calling any BIOS function.

Table 5.1 List of Legacy VGA Functions

Function	Function Name	Function Description (Short)
00h	Set Mode	Load display adapter registers that produce a desired display resolution know as a mode.
01h	Set Cursor Type	Sets the shape of the cursor by setting starting and ending scan lines.
02h	Set Cursor Position	Position the text cursor at a specified location on the display.
03h	Read Cursor Position	Returns the cursor position and type for a specified display page.
04h	Read Light Pen Position	Always returns light pen switch not activated.
05h	Select Active Display Page	Select the page that is displayed to the display device.
06h	Scroll Active Page Up	Scrolls up the number of requested lines in a defined window on the active display page.
07h	Scroll Active Page Down	Scrolls down the number of requested lines in a defined window on the active display page.
08h	Read Attribute / Character at Cursor	Read a character and attribute at the current cursor position.
09h	Write Attribute / Character at Cursor	Writes a character and attribute at the current cursor position.
0Ah	Write Character at Cursor Position	Writes a character to the current cursor position.
0Bh	Set Color Palette	Emulate one of the two standard CGA graphics color palettes.
0Ch	Write Pixel	Write pixels in a graphics mode.
0Dh	Read Pixel	Read pixels in graphics modes.
0Eh	Write Teletype Character	Writes a character at the cursor and then advances the cursor.
0Fh	Read Current Video State	Read screen width, video mode, and active display page.
1000h	Set Individual Palette Register	Sets an Attribute Controller color register with the given value.
1001h	Set Overscan Color	Sets the overscan (border) color register with the given value.
1002h	Set All Palette Registers and Overscan	Loads all Attribute Controller colors and overscan registers.
1003h	Toggle Blink / Intensity Bit	Sets the blink / intensity field of the Mode Control Register (AR10).
1007h	Read Individual Palette Register	Read contents of a requested Attribute Controller palette register.
1008h	Read Overscan Register	Read the contents of the overscan (border) register.
1009h	Read Palette Registers and Overscan	Read all Attribute Controller color and overscan (border) registers.
1010h	Set Individual Color Register	Sets a requested DAC color to a given red green and blue value.
1012h	Set Block of Color Registers	Loads a block of DAC color registers given a starting index, the number to load and a pointer to a table of red / green / blue values.
1013h	Select Color Page	Divides DAC colors into blocks and then makes a block active.
1015h	Read Individual Color Register	Reads a requested DAC color register.
1017h	Read Block of Color Registers	Reads a block of DAC color registers.
101Ah	Read Color Page Status	Read the DAC block mode and active block.

Function	Function Name	Function Description (Short)
101Bh	Sum Color Values to Gray Shades	Converts a block of DAC color registers from color values to monochrome gray scale values.
1100h	Load User Font	Loads a user-defined font into one of eight font areas in memory.
1101h	Load ROM 8 x 14 Font	Load 8x14 font without recalculating mode parameters.
1102h	Load ROM 8 x 8 Font	Load 8x8 font without recalculating mode parameters.
1103h	Set Font Block Specifier	Select which of eight character sets in memory is active.
1104h	Load ROM 8 x 16 Font	Load 8x16 font without recalculating mode parameters.
1110h	Load User Font	Loads a user defined font and recalculate mode parameters.
1111h	Load ROM 8 x 14 Font	Loads 8x14 font and recalculate mode parameters.
1112h	Load ROM 8 x 8 Font	Loads 8x8 font and recalculate mode parameters.
1114h	Load ROM 8 x 16 Font	Loads 8x16 font and recalculate mode parameters.
1120h	Set Interrupt 1Fh Font Pointer	Sets interrupt 1Fh vector to the second half of an 8x8 user defined character set.
1121h	Set Interrupt 43h for User's Font	Sets interrupt 43h vector to point to a user's font table and updates the video ROM BIOS data area.
1122h	Set Interrupt 43h for 8 x 14 Font	Sets interrupt 43h vector to point to the 8x14 font table and updates the video ROM BIOS data area.
1123h	Set Interrupt 43h for 8 x 8 Font	Sets interrupt 43h vector to point to the 8x8 font table and updates the video ROM BIOS data area.
1124h	Set Interrupt 43h for 8 x 16 Font	Sets interrupt 43h vector to point to the 8x16 font table and updates the video ROM BIOS data area.
1130h	Get Font Information	Returns a pointer to a font character definition table.
1210h	Return VGA Information	Returns information on the current VGA configuration.
1220h	Select Alternate Print Screen Routine	Selects an alternate print-screen routine that works properly when the number of character lines is not the normal 25 lines.
1230h	Set Text Mode Scan Lines	Sets the number of scan lines for text modes.
1232h	Enable / Disable Video Adapter	Enables or disables the video adapter from responding to any I/O or memory reads and writes.
1233h	Enable / Disable Gray Scale Summing	Enable or disable the gray scale-summing feature.
1234h	Enable / Disable Cursor Emulation	Enable or disable the cursor emulation feature.
1235h	Switch Active Display	Allow selection between one of two video adapters in the system.
1236h	Video Screen On / Off	Turn on or off video refresh on the display devices.
13h	Write String	Allows a string to be written to display memory by the video BIOS.
1A00h	Read Display Combination Code	Reads the Display Combination Code.
1A01h	Write Display Combination Code	Writes the Display Combination Code.
1Bh	Return Functionality / State Info	Returns functionality and state information.
1C00h	Return Save / Restore State Buffer Size	Returns the minimum buffer size required to hold the data loaded by the save and restore functions.
1C01h	Save State	Loads the given buffer with the requested state data.
1C02h	Restore State	Restores a previously saved state from the given buffer.

00h – Set Mode

This function will load display adapter registers that produce a desired display pixel resolution and timing knows as a mode. Many predefined mode resolutions have been built into the BIOS with an assigned mode number. These modes include standard IBM VGA adapter and newer resolutions as listed in the mode tables included in this document. Along with the x and y components of a resolution, modes have the concepts of color depth, refresh rate and memory organization.

Use the Get Current Video State (interrupt 10h, AH = 0Fh) function to determine if a mode number has set successfully. In most cases, this mode number can be read from the system RAM data area offset 40:49.

Calling Registers:

AH = 00h, Set Mode function
AL = Mode Number:
Bit 7 = Clear Video Memory Bit:
= 0, Clear video memory used to display resolution
= 1, Do not clear video memory
Bits 6-0 = Display mode number (0 to 7Fh)

Return Registers:

None

01h – Set Cursor Type

This function sets the shape of the cursor by setting the starting and ending scan lines of the cursor within the 32 scan line character space. Data from this function is store in the RAM data area 40:60h. Setting the end scan line less than the start scan line will result in no cursor being displayed.

Calling Registers:

AH = 01h, Set Cursor Type function
CH = Start scan line (0-31)
CL = End scan line (0-31)

Return Registers:

None

02h – Set Cursor Position

This function will position the text cursor at a specified location on the display screen. For modes with multiple display pages, a separate cursor, up to eight, is maintained for each display page. The cursor positions for each page are stored in the RAM data area 40:50h to 40:5Fh.

Calling Registers:

AH = 02h, Set Cursor Position function
BH = Display page number
DH:DL = Row : Column where 0:0 is upper left corner

Return Registers:

None

03h – Read Cursor Position

This function returns the cursor position and type for a specified display page.

Calling Registers:

AH = 03h, Read Cursor Position function
BH = Display page number

Return Registers:

CH:CL = Start : End scan line of cursor type
DH:DL = Row : Column where 0:0 is upper left corner

04h – Read Light Pen Position

This function always returns light pen switch not activated.

Calling Registers:

AH = 04h, Read Light Pen Position function

Return Registers:

AH = 00h, Light pen switch not activated

05h – Select Active Display Page

This function selects the page that is displayed to the display device. The active display page is stored in the RAM data area 40:62h.

Calling Registers:

AH = 05h, Select Active Display Page function
AL = Display page number to be made active

Return Registers:

None

06h – Scroll Active Page Up

This function will scroll up the number of requested lines in a defined window on the active display page. Empty lines created at the bottom of the window are cleared using the given attribute. If the given number of lines to scroll is zero the entire window is cleared using the given attribute.

Calling Registers:

AH = 06h, Scroll Active Page Up function
AL = Number of lines to scroll (AL = 0 blanks entire window)
BH = Attribute used when clearing lines
CH:CL = Row : Column of upper-left corner of window
DH:DL = Row : Column of lower-right corner of window

Return Registers:

None

07h – Scroll Active Page Down

This function will scroll down the number of requested lines in a defined window on the active display page. Empty lines created at the top of the window are cleared using the given attribute. If the given number of lines to scroll is zero the entire window is cleared using the given attribute.

Calling Registers:

AH = 07h, Scroll Active Page Down function
AL = Number of lines to scroll (AL = 0 blanks entire window)

BH = Attribute used when clearing lines
CH:CL = Row : Column of upper-left corner of window
DH:DL = Row : Column of lower-right corner of window

Return Registers:

None

08h – Read Attribute / Character at Current Cursor Position

This function returns the character ASCII code and its attribute at the current cursor position of the given display page. No attribute is returned for graphics modes.

Calling Registers:

AH = 08h, Read Attribute / Character at Current Cursor Position function
BH = Display page number

Return Registers:

AL = Character ASCII code
AH = Character attribute

09h – Write Attribute / Character at Current Cursor Position

This function writes a character ASCII code and its attribute at the current cursor position. The character is written the requested number of times with some limitations. The cursor location is not changed.

Calling Registers:

AH = 09h, Write Attribute / Character at Current Cursor Position function
AL = Character ASCII code
BH = Display page number
BL = Character attribute (text modes) or color value (graphics modes)
CX = Repetition count

Return Registers:

None

0Ah – Write Character at Current Cursor Position

This function writes a character ASCII code to the current cursor position. The character attribute is unaffected. The character is written the requested number of times with some limitations. The cursor location is not changed. If the current mode is a graphics mode, and bit D7 of register BL equals 1, the character being written will be exclusive Ored with the previous data in display memory.

Calling Registers:

AH = 0Ah, Write Character at Current Cursor Position function
AL = Character ASCII code
BH = Display page number
BL = Color value (graphics modes)
CX = Repetition count

Return Registers:

None

0Bh – Set Color Palette

This function is provided for CGA compatibility. This will configure the card to emulate one of the two standard CGA graphics color palettes.

Calling Registers:

AH = 0Bh, Set Color Palette function
BH = Color ID:
= 00h
BL = Graphics background color or text border color (0-15)
= 01h
BL = Palette number (0 or 1)

Return Registers:

None

0Ch – Write Pixel

This function provides for a device independent method for manipulating pixels in a graphics mode. Pixel value range is dependent on the color depth of the mode; for example, mode 4 is a 4 bpp mode so the range of values is 0 through 3, whereas mode D is a 16 bpp mode so the range is 0 through 15.

Calling Registers:

AH = 0Ch, Write Pixel function
AL = Pixel value (valid range dependent on color depth)
BH = Display page number
DX:CX = Row : Column of the pixel to write

Return Registers:

None

0Dh – Read Pixel

This function provides a device independent method for reading pixels in graphics modes.

Calling Registers:

AH = 0Dh, Read Pixel function
BH = Display page number
DX:CX = Row : Column of the pixel to write

Return Registers:

AL = Pixel value

0Eh – Write Teletype Character

This function writes a character ASCII code at the current cursor position and then automatically advances the cursor to the next character position. At the end of the line, the text automatically wraps around to the next line. At the end of the page, the screen is automatically scrolled up. The ASCII codes for BELL, BACKSPACE, CARRIAGE RETURN and LINE FEED are all recognized and handled properly. This function is used by the MS-DOS console driver extensively when printing characters to the display screen.

Calling Registers:

AH = 0Eh, Write Teletype Character function
AL = Character ASCII code
BH = Display page number
BL = Character color (graphics modes only)

Return Registers:

None

0Fh – Read Current Video State

This function returns the screen width in character columns, video display mode, and active display page. It is a good way to test if the set mode function was successful.

Calling Registers:

AH = 0Fh, Read Current Video State function

Return Registers:

AH = Number of character columns
AL = Display mode number
BH = Active Display Page

5.1 10h – Set / Get Palette Registers

This function has a group of sub-functions that help in reading or writing of the video adapter palettes. The adapter palettes include the 16 colors for the Attribute Controller; the border color and the 256 RGB colors of the RAMDAC look up table. These functions allow the palette colors to be updated individually or as a block of colors.

1000h – Set Individual Palette Register

This sub-function loads a given individual Attribute Controller color register with the given value.

Calling Registers:

AX = 1000h, Set Individual Palette Register sub-function
BH = Color value
BL = Palette register number (00h – 0Fh)

Return Registers:

None

1001h – Set Overscan Color

This sub-function loads the overscan (border) color register with the given value.

Calling Registers:

AX = 1001h, Set Overscan Color sub-function
BH = Color value

Return Registers:

None

1002h – Set All Palette Registers and Overscan

This sub-function provides a fast method of programming all palette registers. A pointer to a 17 byte table must be provided; bytes 0 through 15 contain data for Attribute Controller palette registers 00h through 0Fh and byte 16 holds the overscan (border) color.

Calling Registers:

AX = 1002h, Set All Palette Registers and Overscan sub-function
ES:DX = Pointer to 17 byte table with Attribute Controller palette information

Return Registers:

None

1003h – Toggle Blink / Intensity Bit

This sub-function loads the blink / intensity field of the Mode Control Register (AR10). This field defines whether bit 7 of the character attribute byte controls blinking or background intensity.

Calling Registers:

AX = 1003h, Toggle Blink / Intensity Bit sub-function
BL = Value for blink / intensity bit in AR10:
= 0, Enable background intensities
= 1, Enable blinking

Return Registers:

None

1007h – Read Individual Palette Register

This sub-function returns the current contents of a requested Attribute Controller palette register.

Calling Registers:

AX = 1007h, Read Individual Palette Register sub-function
BL = Palette register index (00h – 0Fh)

Return Registers:

BH = Palette register value

1008h – Read Overscan Register

This sub-function returns the current contents of the overscan (border) register.

Calling Registers:

AX = 1008h, Read Overscan Register sub-function

Return Registers:

BH = Overscan color register value

1009h – Read All Palette Registers and Overscan

This sub-function read all Attribute Controller palette registers plus the overscan (border) register and places them into a given 17 byte buffer.

Calling Registers:

AX = 1009h, Read All Palette Registers and Overscan sub-function
ES:DX = Pointer to 17 byte buffer

Return Registers:

None

1010h – Set Individual Color Register

This sub-function sets a requested RAMDAC look up table color to the given red, green and blue values.

Calling Registers:

AX = 1010h, Set Individual Color Register sub-function
BX = Color register index (0-255)
DH:CH:CL = Red : Green : Blue values to set in color register

Return Registers:

None

1012h – Set Block of Color Registers

This sub-function sets a block of RAMDAC look up table color registers given a starting color register index, the number of color registers to set and a pointer to a table of red / green / blue values.

Calling Registers:

AX = 1012h, Set Block of Color Registers sub-function
BX = Starting color register index (0-255)
CX = Number of color registers to set (1-256)
ES:DX = Pointer to table of color register values

Return Registers:

None

1013h – Select Color Page

There are 256 color registers, but most standard VGA modes do not use all of them. Depending on the mode color resolution, the 256 color registers can be split up into 16 blocks with 16 color registers, or four blocks with 64 color registers.

This sub-function allows an application to select how the 256 color registers are to be divided, as explained above, and then allow the application to select which of these blocks is to be made active.

Calling Registers:

AX = 1013h, Select Color Page sub-function
BL = Select function of this call:
= 0, Select paging mode

BH = Paging mode:
= 0, Select 4 pages of 64 color registers
= 1, Select 16 pages of 16 color registers
= 1, Select Active Color Page Mode
BH = Select Active color register page:
= 0-3, if in 4 page mode
= 0-15, if in 16 page mode

Return Registers:

None

1015h – Read Individual Color Register

This sub-function returns the contents of a requested individual RAMDAC look up table color register.

Calling Registers:

AX = 1015h, Read Individual Color Register sub-function
BX = Color register index (0-255)

Return Registers:

DH:CH:CL = Red : Green : Blue values to set in color register

1017h – Read Block of Color Registers

This sub-function returns a block of RAMDAC look up table color registers in the provided memory buffer given a starting color register index and the number of color registers to read.

Calling Registers:

AX = 1017h, Read Block of Color Registers sub-function
BX = Starting color register index (0-255)
CX = Number of color registers to read (1-256)
ES:DX = Pointer to memory buffer large enough to contain requested data

Return Registers:

None

101Ah – Read Color Page Status

This sub-function returns the current 256 color register paging mode and active page as set in function Select Color Page (interrupt 10h, AX = 1013h).

Calling Registers:

AX = 101Ah, Read Color Page Status sub-function

Return Registers:

BH = Current active page
BL = Current page mode

101Bh – Sum Color Values to Gray Shades

This sub-function converts a block of RAMDAC look up table color registers from color values to monochrome gray scale values. For each register, the color data is read, a weighted sum computed (30% red, 59% green, 11% blue), and the result written back to the register. The original color data is lost.

Calling Registers:

AX = 101Bh, Sum Color Values to Gray Shades sub-function
BX = Starting color register index (0-255)
CX = Number of color registers to sum (1-256)

Return Registers:

None

5.2 11h – Character Generator

This function has a group of sub-functions that help deal with internal video BIOS and user font sets. Two sets of calls are provided for use with text modes. The first simply loads the requested font in plane 2 of the frame buffer memory. The second loads the font in plane 2 and then adjusts certain display parameters.

For text modes, individual characters of a font can be up to 32 bytes in length. The font area reserved in display memory can hold up to eight 256 character font sets. The font characters of any length are loaded on 32 byte boundaries for easy access. Therefore, a 256 character font will take up to 8K (8192) bytes. The active character set in the font area is selected through VGA sequencer register SR03.

1100h – Load User Font

This sub-function loads a user-defined font or a portion of it into one of eight font areas reserved in text mode memory.

Note: This function does not recalculate the CRT controller registers.

Calling Registers:

AX = 1100h, Load User Font sub-function
BH = Number of bytes per character (1-32)
BL = Number of font buffer to load the user's font (0-7)
CX = Number of characters to load (1-256)
DX = Character offset into character generator table (0-255)
ES:BP = Pointer to character table in system memory

Return Registers:

None

1101h – Load ROM 8x14 Font

This sub-function will load the video BIOS 8x14 font into the specified font area in display memory. This font is mainly a monochrome / EGA font. To save space on a video BIOS's, this font is sometimes removed and replaced with a conversion from the 8x16 font.

Note: This function does not recalculate the CRT controller registers.

Calling Registers:

AX = 1101h, Load ROM 8x14 Font sub-function
BL = Number of font buffer to load the BIOS 8x14 font (0-7)

Return Registers:

None

1102h – Load ROM 8x8 Font

This sub-function will load the video BIOS 8x8 font into the specified font area in display memory.

Note: This function does not recalculate the CRT controller registers.

Calling Registers:

AX = 1102h, Load ROM 8x8 Font sub-function
BL = Number of font buffer to load the BIOS 8 x 8 font (0-7)

Return Registers:

None

1103h – Set Font Block Specifier

This sub-function allows the application to select which of the eight internal character generator tables (font buffers) are active. Before a table can be made active, it must be loaded with character data.

Calling Registers:

AX = 1103h, Set Font Block Specifier sub-function
BL [0, 1, 4] = Character selected by attribute bytes with bit 3 = 0
BL [2, 3, 5] = Character selected by attribute bytes with bit 3 = 1

Return Registers:

None

1104h – Load ROM 8x16 Font

This sub-function will load the video BIOS 8x16 font into the specified font area in display memory.

Note: This function does not recalculate the CRT controller registers.

Calling Registers:

AX = 1104h, Load ROM 8x16 Font sub-function
BL = Number of font buffer to load the BIOS 8x16 font (0-7)

Return Registers:

None

1110h – Load User Font

This sub-function loads a user-defined font or a portion of it into one of eight font areas reserved in text mode memory.

Note: Certain CRT controller settings, such as Max Scan Line, Cursor Start and End, and Underline Location are recalculated based on the character information provided.

Calling Registers:

AX = 1110h, Load User Font sub-function
BH = Number of bytes per character (1-32)
BL = Number of font buffer to load the user's font (0-7)
CX = Number of characters to load (1-256)
DX = Character offset into character generator table (0-255)
ES:BP = Pointer to character table in system memory

Return Registers:

None

1111h – Load ROM 8x14 Font

This sub-function will load the video BIOS 8x14 font into the specified font area in display memory. This font is mainly a monochrome / EGA font. To save space on a video BIOS's, this font is sometimes removed and replaced with a conversion from the 8x16 font.

Note: Certain CRT controller settings, such as Max Scan Line, Cursor Start and End, and Underline Location are recalculated based on the character information provided.

Calling Registers:

AX = 1111h, Load ROM 8x14 Font sub-function
BL = Number of font buffer to load the BIOS 8x14 font (0-7)

Return Registers:

None

1112h – Load ROM 8x8 Font

This sub-function will load the video BIOS 8x8 font into the specified font area in display memory.

Note: Certain CRT controller settings, such as Max Scan Line, Cursor Start and End, and Underline Location are recalculated based on the character information provided.

Calling Registers:

AX = 1112h, Load ROM 8x8 Font sub-function
BL = Number of font buffer to load the BIOS 8x8 font (0-7)

Return Registers:

None

1114h – Load ROM 8x16 Font

This sub-function will load the video BIOS 8x16 font into the specified font area in display memory.

Note: Certain CRT controller settings, such as Max Scan Line, Cursor Start and End, and Underline Location are recalculated based on the character information provided.

Calling Registers:

AX = 1114h, Load ROM 8x16 Font sub-function
BL = Number of font buffer to load the BIOS 8x16 font (0-7)

Return Registers:

None

1120h – Set Interrupt 1Fh Font Pointer

This sub-function sets a pointer to the second half of an 8x8 user defined character set in the interrupt 1Fh vector. This second half refers to characters with ASCII codes from 128 to 255. Only modes 4, 5, and 6 utilize this pointer.

Calling Registers:

AX = 1120h, Set Interrupt 1Fh Font Pointer sub-function
ES:BP = Pointer to upper half of 8x8 character set

Return Registers:

None

1121h – Set Interrupt 43h for User's Font

This sub-function sets the vector for interrupt 43h to point to the user's font table and updates the video ROM BIOS data area.

Note: The video controller is not reprogrammed.

Calling Registers:

AX = 1121h, Set Interrupt 43h for User's Font sub-function
BL = Number of character rows to be displayed
= 00h, User specified in DL
= 01h, 14 character rows
= 02h, 25 character rows
= 03h, 43 character rows
CX = Bytes per character
DL = Character rows per screen (if BL = 00h)
ES:BP = Pointer to user defined font

Return Registers:

None

1122h – Set Interrupt 43h for 8x14 Font

This sub-function sets the vector for interrupt 43h to point to the 8x14 font table and updates the video ROM BIOS data area.

Note: The video controller is not reprogrammed.

Calling Registers:

AX = 1122h, Set Interrupt 43h for 8x14 Font sub-function
BL = Number of character rows to be displayed
= 00h, User specified in DL
= 01h, 14 character rows
= 02h, 25 character rows
= 03h, 43 character rows
DL = Character rows per screen (if BL = 00h)

Return Registers:

None

1123h – Set Interrupt 43h for 8x8 Font

This sub-function sets the vector for interrupt 43h to point to the 8x8 font table and updates the video ROM BIOS data area.

Note: The video controller is not reprogrammed.

Calling Registers:

AX = 1123h, Set Interrupt 43h for 8x8 Font sub-function
BL = Number of character rows to be displayed
= 00h, User specified in DL
= 01h, 14 character rows
= 02h, 25 character rows
= 03h, 43 character rows
DL = Character rows per screen (if BL = 00h)

Return Registers:

None

1124h – Set Interrupt 43h for 8x16 Font

This sub-function sets the vector for interrupt 43h to point to the 8x16 font table and updates the video ROM BIOS data area.

Note: The video controller is not reprogrammed.

Calling Registers:

AX = 1124h, Set Interrupt 43h for 8x16 Font sub-function
BL = Number of character rows to be displayed
= 00h, User specified in DL
= 01h, 14 character rows
= 02h, 25 character rows
= 03h, 43 character rows
DL = Character rows per screen (if BL = 00h)

Return Registers:

None

1130h – Get Font Information

This sub-function returns a pointer to the character definition table for a font. It also returns the bytes per character and character rows for that font.

Calling Registers:

AX = 1130h, Get Font Information sub-function
BH = Font code:
= 00h, Return current interrupt 1Fh pointer
= 01h, Return current interrupt 44h pointer
= 02h, Return BIOS 8x14 font pointer
= 03h, Return BIOS 8x8 font pointer (characters 0 – 127)
= 04h, Return BIOS 8x8 font pointer (characters 128 – 255)
= 05h, Return BIOS 9x14 font pointer
= 06h, Return BIOS 8x16 font pointer
= 07h, Return BIOS 9x16 font pointer

Return Registers:

CX = Number of bytes per character
DL = Character rows on screen – 1
ES:BP = Pointer to requested font

5.3 12h – Alternate Select

This function performs a variety of seemingly unrelated tasks. These tasks range from returning video information to selecting the number of scan lines in a mode resolution.

12h, 10h – Return VGA Information

This function returns information on the current VGA configuration. Although most of this information is no longer applicable, information is returned for compatibility.

Calling Registers:

AH = 12h, Alternate Select function
BL = 10h, Return VGA Information function

Return Registers:

BH = Mode currently in effect:
= 00h, Color mode (3Dx address range)
= 01h, Monochrome mode (3Bx address range)
BL = 03h, 256K bytes of video memory for VGA modes
CH = Feature bits
CL = Switch setting

12h, 20h – Select Alternate Print Screen Routine

This function selects an alternate print-screen routine that works properly if the number of character lines is not the normal 25 lines. The ROM BIOS default print-screen routine always prints 25 lines.

Calling Registers:

AH = 12h, Alternate Select function
BL = 20h, Select Alternate Print Screen Routine function

Return Registers:

None

12h, 30h – Set Text Mode Scan Lines

This function selects the number of scan lines for text modes. The selected value takes effect the next time Set Mode (Interrupt 10h, AH = 00h) is called.

Calling Registers:

AH = 12h, Alternate Select function
BL = 30h, Set Text Mode Scan Lines function
AL = Scan line code:
= 00h, 200 Scan lines
= 01h, 350 Scan lines
= 02h, 400 Scan lines

Return Registers:

AH = 12h, If function was successful

12h, 31h – Enable / Disable Default Palette Loading

This function enables or disables loading of a default palette when Set Mode (Interrupt 10h, AH = 00h) is called.

Calling Registers:

AH = 12h, Alternate Select function
BL = 31h, Enable / Disable Default Palette Loading function
AL = Default palette loading:
= 00h, Enable
= 01h, Disable

Return Registers:

AH = 12h, If function was successful

12h, 32h – Enable / Disable Video Adapter

This function enables or disables the video adapter from responding to any I/O or memory reads or writes. The display is not affected.

Calling Registers:

AH = 12h, Alternate Select function
BL = 32h, Enable / Disable Video Adapter function
AL = Video Adapter:
= 00h, Enable
= 01h, Disable

Return Registers:

AH = 12h, If function was successful

12h, 33h – Enable / Disable Gray Scale Summing

This function will enable or disable the gray scale-summing feature. If enabled, gray scale summing will be performed to all 256 RAMDAC colors whenever the color registers are loaded through video BIOS functions.

Calling Registers:

AH = 12h, Alternate Select function
BL = 33h, Enable / Disable Gray Scale Summing function
AL = Gray Scale Summing:
= 00h, Enable
= 01h, Disable

Return Registers:

AH = 12h, If function was successful

12h, 34h – Enable / Disable Cursor Emulation

This function addresses the cursor compatibility problem when CGA software that sets cursor shape is used with EGA or VGA text modes. Because of the larger character cell, the cursor will appear in the wrong part of the cell unless CGA cursor emulation is enabled to translate the cursor parameters to different values.

Calling Registers:

AH = 12h, Alternate Select function
BL = 34h, Enable / Disable Cursor Emulation function
AL = Cursor Emulation:
= 00h, Enable
= 01h, Disable

Return Registers:

AH = 12h, If function was successful

12h, 35h – Switch Active Display

This function will allow selection between one of two video adapters in the system, when memory usage or port addresses conflict between the two adapters.

Calling Registers:

AH = 12h, Alternate Select function
BL = 35h, Switch Active Display function
AL = Switching function:
= 00h, Disable initial video adapter
= 01h, Enable system board video adapter
= 02h, Disable active video adapter
= 03h, Enable inactive video adapter
ES:DX = Pointer to 128 byte buffer (if AL = 00h, 02h or 03h)

Return Registers:

AH = 12h, If function was successful

12h, 36h – Video Screen On / Off

This function will turn on or off video refresh on the display devices.

Calling Registers:

AH = 12h, Alternate Select function
BL = 36h,Video Screen On / Off function
AL = Screen Refresh:
= 00h, Enable
= 01h, Disable

Return Registers:

AH = 12h, If function was successful

13h – Write String

This function allows an application to pass an entire text string to the BIOS for display given a pointer to the string. The string may or may not include embedded attribute data. ASCII characters for BELL, BACKSPACE, CARRIAGE RETURNS, and LINEFEED are all recognized and handled accordingly.

Calling Registers:

AH = 13h, Write String function
AL = Write Mode:
= 00h, BL contains attribute for all characters; cursor not advanced
= 01h, BL contains attribute for all characters; cursor advanced
= 02h, String contains ASCII characters with attributes; cursor not advanced
= 03h, String contains ASCII characters with attributes; cursor advanced
BH = Display page number
BL = Attribute for all characters (if AL = 0 or 1)
CX = Character count (length of string)
DH = Row number on display for start of string
DL = Column number on display for start of string
ES:BP = Pointer of source text string in system memory

Return Registers:

None

5.4 1Ah – Read / Write Display Combination Code

This function has separate read and write sub-functions that deal with information on the current installed display adapters. The video BIOS supports the following display combination codes.

Table 5.1 Function AH = 1Ah Display Combination Codes

Code	Adapter	Monitor
00h	Any Display Adapter	No monitor
01h	Monochrome Display Adapter (MDS)	Monochrome monitor
07h	Video Graphics Adapter (VGA)	Monochrome monitor
08h	Video Graphics Adapter (VGA)	Color monitor

1A00h – Read Display Combination Code

This sub-function is used to read the Display Combination Code, which stores the current installed display adapters.

Calling Registers:

AX = 1A00h, Read Display Combination Code sub-function

Return Registers:

AL = 1Ah, Function Supported
BH = Inactive display code
BL = Active display code

1A01h – Write Display Combination Code

This sub-function is used to write the Display Combination Code, which stores the current installed display adapters.

Calling Registers:

AX = 1A01h, Write Display Combination Code sub-function
BH = Inactive display code
BL = Active display code

Return Registers:

AL = 1Ah, Function Supported

1Bh – Return Functionality / State Information

This function returns functionality and state information in a provided memory buffer for the current adapter and mode. See the following details of the returned information.

Calling Registers:

AH = 1Bh, Return Functionality / State Information function
BX = 00h, Implementation type
ES:DI = Pointer to 64 byte buffer

Return Registers:

AL = 1Bh, Function Supported

Table 5.2 Function AH = 1Bh State Information

Offset	Size	Contents
00h	Dword	Pointer (Offset/ Segment) to Static Functionality Table (See following table)
04h	Byte	Current video mode number
05h	Word	Number of character columns
07h	Word	Length of each display page in bytes (regeneration buffer)
09h	Word	Offset address of current page in display memory
0Bh	8 Words	Cursor positions, two words per page, for up to 8 pages
1Bh	Byte	Cursor starting line
1Ch	Byte	Cursor ending line
1Dh	Byte	Active display page
1Eh	Word	CRT controller address (3B4h or 3D4h)
20h	Byte	CGA / MDA mode register value (value of 3B8h or 3D8h)
21h	Byte	CGA / MDA color register value (value of 3B9h or 3D9h)
22h	Byte	Number of character rows
23h	Byte	Character height (in scan lines)
25h	Byte	Display Combination Code (active display)
26h	Byte	Display Combination Code (inactive display)
27h	Word	Number of colors in current mode (0 for mono modes)
29h	Byte	Number of display pages in current mode

Offset	Size	Contents
2Ah	Byte	Number of scan lines in current mode 00h = 200 scan lines 01h = 350 scan lines 02h = 400 scan lines 03h = 480 scan lines
2Bh	Byte	Primary character generator block (0-7)
2Ch	Byte	Secondary character generator block (0-7)
2Dh	Byte	Miscellaneous state information Bit 0 = 1, All modes supported on all monitors Bit 1 = 1, Gray scale conversion enabled Bit 2 = 1, Monochrome display attached Bit 3 = 1, Default palette initialization disabled Bit 4 = 1, CGA cursor emulation enabled Bit 5 = 1, Blinking enabled = 0, Background intensify enabled
2Eh – 30h	Byte	Reserved
31h	Byte	Video memory available 00h = 64 K bytes 01h = 128 K bytes 02h = 192 K bytes 03h = 256 K bytes
32h	Byte	Save Pointer State information Bit 0 = 1, 512 Character set active Bit 1 = 1, Dynamic save area active Bit 2 = 1, Alpha font override active Bit 3 = 1, Graphics font override active Bit 4 = 1, Palette override active Bit 5 = 1, DCC extension is active (DCC override)
33h – 3Fh	Byte	Reserved
00h	Byte	Supported video modes (0 = Not supported, 1 = Supported) Bit 0 = 1, Mode 00h Bit 1 = 1, Mode 01h Bit 2 = 1, Mode 02h Bit 3 = 1, Mode 03h Bit 4 = 1, Mode 04h Bit 5 = 1, Mode 05h Bit 6 = 1, Mode 06h Bit 7 = 1, Mode 07h
01h	Byte	Supported video modes (0 = Not supported, 1 = Supported) Bit 0 = 0, Mode 08h Bit 1 = 0, Mode 09h Bit 2 = 0, Mode 0Ah Bit 3 = 0, Mode 0Bh Bit 4 = 0, Mode 0Ch Bit 5 = 1, Mode 0Dh Bit 6 = 1, Mode 0Eh Bit 7 = 1, Mode 0Fh
02h	Byte	Supported video modes (0 = Not supported, 1 = Supported) Bit 0 = 1, Mode 10h Bit 1 = 1, Mode 11h Bit 2 = 1, Mode 12h Bit 3 = 1, Mode 13h

Offset	Size	Contents
03h – 07h	Bytes	Reserved
07h	Byte	Scan lines available in text modes (0 = Not supported, 1 = Supported) Bit 0 = 1, 200 scan lines Bit 1 = 1, 350 scan lines Bit 2 = 1, 400 scan lines
08h	Byte	02h Character blocks available in text modes
09h	Byte	08h Maximum number of active character blocks in text modes
0Ah	Byte	Miscellaneous functions (0 = Not supported, 1 = Supported) Bit 0 = 1, All modes on all displays Bit 1 = 1, Gray scale summing available Bit 2 = 1, Character font loading available Bit 3 = 1, Set mode default palette loading available Bit 4 = 1, Cursor emulation available Bit 5 = 1, EGA (64 color) palette available Bit 6 = 1, Color register loading available Bit 7 = 1, Color register paging mode select available
0Bh	Byte	Miscellaneous functions (0 = Not supported, 1 = Supported) Bit 0 = 0, Light pen available Bit 1 = 1, Save / Restore video state available Bit 2 = 1, Background intensity / blinking control available Bit 3 = 1, Get / Set display combination code available
0Ch–0Dh	Bytes	Reserved
0Eh	Byte	Miscellaneous functions (0 = Not supported, 1 = Supported) Bit 0 = 1, Supports 512 character set Bit 1 = 1, Dynamic save area available Bit 2 = 1, Alpha font override available Bit 3 = 1, Graphics font override available Bit 4 = 1, Palette override available Bit 5 = 1, Display combination code extension available
0Fh	Bytes	Reserved

5.5 1Ch – Save / Restore Video State

This function with its sub-functions save and restore specified video environment parameters (BIOS data area, color palette, and video adapter registers). The video BIOS has extended this function (CX bits 13 and 15) to include the extended state.

1C00h – Return Save / Restore State Buffer Size

This sub-functions returns the minimum buffer size (in 64 byte blocks) required to hold the data loaded by the save and restore functions. The size of this buffer is updated according to the save / restore options selected in CX.

Calling Registers:

AX = 1C00h, Return Save / Restore State Buffer Size sub-function
CX = Save / Restore options:
Bit 15 = Extended I/O registers
Bit 13 = Extended memory mapped registers
Bit 2 = Video DAC state and color registers

Bit 1 = Video BIOS data area
Bit 0 = Video hardware state (VGA registers)

Return Registers:

AL = 1Ch, If function was successful
BX = Required buffer size in 64 byte blocks

1C01h – Save State

This sub-function loads the given buffer with the requested state data.

Note: This function does not check the size of the given buffer. Use the Return Save / Restore State Buffer Size function (Interrupt 10h, AX = 1C00h) to get minimum buffer size.

Calling Registers:

AX = 1C01h, Save State sub-function
CX = Save state options:
Bit 15 = Extended I/O registers
Bit 13 = Extended memory mapped registers
Bit 2 = Video DAC state and color registers
Bit 1 = Video BIOS data area
Bit 0 = Video hardware state (VGA registers)
ES:BX = Pointer to state buffer

Return Registers:

AL = 1Ch, If function was successful

1C02h – Restore State

This sub-function restores a previously saved state from the given buffer.

Calling Registers:

AX = 1C02h, Restore State sub-function
CX = Restore state options:
Bit 15 = Extended I/O registers
Bit 13 = Extended memory mapped registers
Bit 2 = Video DAC state and color registers
Bit 1 = Video BIOS data area
Bit 0 = Video hardware state (VGA registers)
ES:BX = Pointer to state buffer

Return Registers:

AL = 1Ch, If function was successful

5.6 BIOS Data Area

The BIOS data area is a section of low memory used by video BIOS to store working variables. These variables may be read from by any application to determine the current state of the video BIOS. If an application makes changes directly to the hardware without calling the video BIOS, the application should also update the corresponding BIOS data areas to avoid confusing the BIOS.

Table 5.1 BIOS Data Area

Address	Size	Contents
0:0040h	Dword	Video BIOS main vector (10h * 4)
0:007Ch	Dword	Font extension vector (1Fh * 4)
0:0108h	Dword	MDA / CGA video BIOS vector (42h * 4)
0:010Ch	Dword	Font vector (43h * 4)
0:01B4h	Dword	Video BIOS secondary entry point vector (6Dh * 4)
40:10h	Byte	Installed hardware: Bit 5,4 = Video mode type: = 00, Reserved = 01, 40 x 25 color = 10, 80 x 25 color = 11, 80 x 25 monochrome
40:49h	Byte	Current video mode number
40:4Ah	Word	Number of character columns
40:4Ch	Word	Length of each display page in bytes (regeneration buffer)
40:4Eh	Word	Offset address of current page in display memory
40:50h	Word	Cursor positions page 1 (row, column)
40:52h	Word	Cursor positions page 2 (row, column)
40:54h	Word	Cursor positions page 3 (row, column)
40:56h	Word	Cursor positions page 4 (row, column)
40:58h	Word	Cursor positions page 5 (row, column)
40:5Ah	Word	Cursor positions page 6 (row, column)
40:5Ch	Word	Cursor positions page 7 (row, column)
40:5Eh	Word	Cursor positions page 8 (row, column)
40:60h	Word	Cursor type (start, end)
40:62h	Byte	Current active display page number
40:63h	Word	CRT controller base I/O port address (3B4h or 3D4h)
40:65h	Byte	CGA Mode register setting
40:66h	Byte	CGA Color register setting
40:72h	Word	Reset flag: 1234h = Bypass memory test 4321h = Preserve Memory (PS/2)
40:84h	Byte	Number of character rows – 1
40:85h	Word	Bytes per character of the active font

Address	Size	Contents
40:87h	Byte	VGA miscellaneous state information: Bit 7 = Mode number clear display memory bit from last mode set Bits 6-5 = Display memory size: = 00, 64 K bytes = 01, 128 K bytes = 10, 192 K bytes = 11, 256 K bytes Bit 4 = Reserved Bit 3 = VGA is primary display: = 0, Yes = 1, No Bit 2 = CPU write to display buffer status bit: = 0, Write anytime = 1, Write only when display is inactive Bit 1 = Display monitor: = 0, Color = 1, Monochrome Bit 0 = Cursor emulation: = 0, Direct cursor setting = 1, Emulate 8 x 8 cursor
40:88h	Byte	VGA switch data: Bits 7-4 Feature bits 3-0 Bits 3-0 Dip switches 3-0
40:89h	Byte	Miscellaneous flags: Bits 7, 4= Scan line count = 00, 350 scan lines = 01, 400 scan lines = 10, 200 scan lines = 11, reserved Bit 6 = Display switching = 0, Disabled = 1, Enabled Bit 5 = Reserved Bit 3 = Default palette loading = 0, Enabled = 1, Disabled Bit 2 = Color / Monochrome monitor status bit = 0, Color = 1, Monochrome Bit 1 = Gray scale summing = 0, Disabled = 1, Enabled Bit 0 = MDA / CGA adapter status bit = 0, VGA in not only adapter = 1, VGA is only adapter
40:8Ah	Byte	Index into display combination code table for current display combination
40:A8h	Dword	Pointer to VGA environment structure

6 BIOS Extended Interface Functions

The BIOS provides a set of proprietary function calls to control operation of the extended features. These function calls all use AH = 5Fh in there designed interface for easy identification as a proprietary function.

Note: IOSS (see function 5F1Ch) must be set to the appropriate values before calling any BIOS function.

Note: The extended video BIOS functions only document registers that supports the current primary feature of the function. All registers especially registers not documented are subject to change without notice. Registers of importance must be saved before and restored after calling a video BIOS function including any that is not currently documented by a function definition. Segment registers CS, DS, DS, SS, and registers SP and BP are saved and restored by the video BIOS unless otherwise documented. The flags register are saved and restored by the assembly INT and IRET instructions.

Table 6.1 List of BIOS Extended Functions

Function	Function Name	Function Description (Short)
5F00h	Get Controller Information	Returns video controller and BIOS information.
5F02h	Set Dot and Memory Clocks	Programs the internal Dot or Memory clock synthesizer with a given frequency.
5F04h	Get Refresh Rate	Returns vertical refresh rate information for a given mode.
5F05h	Set Refresh Rate	Sets a new vertical refresh rate for a given mode.
5F10h	Get Linear Display Memory Info	Returns information regarding the linear memory starting address, size and width.
5F11h	Get Memory Map I/O Information	Returns information regarding memory mapped I/O.
5F12h	Get Video Memory Information	Returns the amount of video memory available for the current video mode and all video modes.
5F14h, 00h	Set Pop-Up Memory Mode	Sets the pop-up memory mode.
5F14h, 01h	Reset Pop-Up Memory Mode	Resets the pop-up memory mode.
5F14h, 02h	Enable Pop-Up	Enables a selected type of pop-up.
5F14h, 03h	Disable Pop-Up	Disables a pop-up.
5F14h, 04h	Get Pop-Up Memory Offset	Returns pop-up memory offset.
5F14h, 05h	Set X, Y Pop-Up Position	Specifies the pop-up position on the screen.
5F19h, 00h	Get NTSC / PAL Support	Gets the state of the NTSC / PAL flags and the current NTSC / PAL display mode.
5F19h, 01h	Set NTSC / PAL Support	Sets flags to indicate if NTSC, PAL or neither mode should be activated when an appropriate system state is entered.
5F19h, 04h	Get TV Flicker Reduction State	Gets the current TV flicker reduction state.
5F19h, 05h	Set TV Flicker Reduction State	Sets the TV flicker reduction state.
5F19h, 08h	Get TV Scaling State	Gets the current TV scaling state.
5F19h, 09h	Set TV Scaling State	Sets the current TV scaling state.

Function	Function Name	Function Description (Short)
5F00h	Get Controller Information	Returns video controller and BIOS information.
5F02h	Set Dot and Memory Clocks	Programs the internal Dot or Memory clock synthesizer with a given frequency.
5F04h	Get Refresh Rate	Returns vertical refresh rate information for a given mode.
5F05h	Set Refresh Rate	Sets a new vertical refresh rate for a given mode.
5F10h	Get Linear Display Memory Info	Returns information regarding the linear memory starting address, size and width.
5F11h	Get Memory Map I/O Information	Returns information regarding memory mapped I/O.
5F12h	Get Video Memory Information	Returns the amount of video memory available for the current video mode and all video modes.
5F14h, 00h	Set Pop-Up Memory Mode	Sets the pop-up memory mode.
5F14h, 01h	Reset Pop-Up Memory Mode	Resets the pop-up memory mode.
5F14h, 02h	Enable Pop-Up	Enables a selected type of pop-up.
5F14h, 03h	Disable Pop-Up	Disables a pop-up.
5F14h, 04h	Get Pop-Up Memory Offset	Returns pop-up memory offset.
5F14h, 05h	Set X, Y Pop-Up Position	Specifies the pop-up position on the screen.
5F19h, 00h	Get NTSC / PAL Support	Gets the state of the NTSC / PAL flags and the current NTSC / PAL display mode.
5F1Ah	GPIO Pins Used by I^2C Busses	Returns GPIO pins that are used by I^2C bus hardware applications.
5F1Ch, 00h	Set Pipe Register Access	Set IOSS and MSS as necessary to set the requested pipe register access.
5F1Ch, 01h	Get Pipe Register Access	Get the current pipe register access.
5F1Dh	Hardware Adjustments	Executes code to prevent or fix a hardware problem.
5F1Eh, 00h	Before Low Power Suspend	Allows the video BIOS control when entering a suspend state.
5F1Eh, 01h	After Low Power Resume from Suspend	Allows the video BIOS control when resuming from a suspended state.
5F22h	Get Mode Support Information	Returns display types that are supported by the given or current mode.
5F24h	Limited Set Mode	Used by Windows 3.1 drivers to fix a bug in Windows 3.1.
5F26h, 00h	Initialize Before I^2C Bus Functions	Initialize the system state before any I^2C bus functions are executed.
5F26h, 01h	Reset After I^2C Bus Functions	Reset the system state after executing I^2C bus functions.
5F26h, 02h	I^2C Bus Start Cycle	Performs an I^2C bus start cycle.
5F26h, 03h	I^2C Bus Stop Cycle	Performs an I^2C bus stop cycle.
5F26h, 04h	Read I^2C Bus Byte	Reads a byte from an I^2C bus.
5F26h, 05h	Write I^2C Bus Byte	Writes a byte to an I^2C bus.
5F26h, 06h	Set I^2C Bus Clock High	Sets the I^2C bus clock (SCL) pin high.
5F26h, 07h	Set I^2C Bus Clock Low	Sets the I^2C bus clock (SCL) pin low.
5F26h, 08h	Read I^2C Bus Pin	Reads the requested I^2C bus pin.
5F26h, 09h	Write I^2C Bus Pin	Writes the requested I^2C bus pin to given values.

Function	Function Name	Function Description (Short)
5F00h	Get Controller Information	Returns video controller and BIOS information.
5F02h	Set Dot and Memory Clocks	Programs the internal Dot or Memory clock synthesizer with a given frequency.
5F04h	Get Refresh Rate	Returns vertical refresh rate information for a given mode.
5F05h	Set Refresh Rate	Sets a new vertical refresh rate for a given mode.
5F10h	Get Linear Display Memory Info	Returns information regarding the linear memory starting address, size and width.
5F11h	Get Memory Map I/O Information	Returns information regarding memory mapped I/O.
5F12h	Get Video Memory Information	Returns the amount of video memory available for the current video mode and all video modes.
5F14h, 00h	Set Pop-Up Memory Mode	Sets the pop-up memory mode.
5F14h, 01h	Reset Pop-Up Memory Mode	Resets the pop-up memory mode.
5F14h, 02h	Enable Pop-Up	Enables a selected type of pop-up.
5F14h, 03h	Disable Pop-Up	Disables a pop-up.
5F14h, 04h	Get Pop-Up Memory Offset	Returns pop-up memory offset.
5F14h, 05h	Set X, Y Pop-Up Position	Specifies the pop-up position on the screen.
5F19h, 00h	Get NTSC / PAL Support	Gets the state of the NTSC / PAL flags and the current NTSC / PAL display mode.
5F28h	Get Mode Support	Determines if the given state (i.e. pipe modes and display device) can be supported by the hardware and BIOS.
5F29h	Get Mode Information	Returns the requested mode’s resolutions, color depth, and maximum required bandwidth using its current refresh rate.
5F50h	Get Display Information	Returns display device information.
5F51h	Switch Display Device	Switches between CRT, flat panel, and simultaneous displays.
5F54h	Set Panel ON / OFF	Sets the panel ON or OFF (standby state).
5F55h	Monitor Detect	Detects if a monitor is currently attached to the adapter, and if one is found, determines whether it is color or monochrome.
5F56h	Get Panel Type	Return panel type information.
5F5Ah	Set Flat Panel Video Polarity	Sets the polarity of the video output to the flat-panel.
5F5Bh	Set Horizontal Compensation	Enables or disables Horizontal Graphics compensation.
5F5Eh	Set Vertical Compensation	Enables or disables Vertical Graphics compensation.
5F60h, 00h	Set FP Low Power State	Sets the flat panel low power on state.
5F60h, 01h	Get FP Low Power State	Gets the current flat panel low power on state.
5F61h, 00h	Set Horizontal & Vertical Comp	Sets horizontal and vertical compensation components.
5F61h, 01h	Get Horizontal & Vertical Comp	Gets horizontal and vertical compensation components.
5F63h	Adapter Power State	Sets up the video controller for the power state switches and notifies the system BIOS of the requested power state switch.

Function	Function Name	Function Description (Short)
5F00h	Get Controller Information	Returns video controller and BIOS information.
5F02h	Set Dot and Memory Clocks	Programs the internal Dot or Memory clock synthesizer with a given frequency.
5F04h	Get Refresh Rate	Returns vertical refresh rate information for a given mode.
5F05h	Set Refresh Rate	Sets a new vertical refresh rate for a given mode.
5F10h	Get Linear Display Memory Info	Returns information regarding the linear memory starting address, size and width.
5F11h	Get Memory Map I/O Information	Returns information regarding memory mapped I/O.
5F12h	Get Video Memory Information	Returns the amount of video memory available for the current video mode and all video modes.
5F14h, 00h	Set Pop-Up Memory Mode	Sets the pop-up memory mode.
5F14h, 01h	Reset Pop-Up Memory Mode	Resets the pop-up memory mode.
5F14h, 02h	Enable Pop-Up	Enables a selected type of pop-up.
5F14h, 03h	Disable Pop-Up	Disables a pop-up.
5F14h, 04h	Get Pop-Up Memory Offset	Returns pop-up memory offset.
5F14h, 05h	Set X, Y Pop-Up Position	Specifies the pop-up position on the screen.
5F19h, 00h	Get NTSC / PAL Support	Gets the state of the NTSC / PAL flags and the current NTSC / PAL display mode.
5F64h, 00h	Set Display Device	Sets the given display device combination.
5F64h, 01h	Get Display Device	Gets the current display device combination.
5F64h, 02h	Set Mosaic Mode	Informs the BIOS that mosaic mode has been enabled or disabled.
5F64h, 03h	Get Mosaic Mode	Gets the current state of mosaic mode.

5F00h – Get Controller Information

This function returns video controller and BIOS information.

Calling Registers:

AX = 5F00h, Get Controller Information function

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
BH = Video memory available:
= 07h, 4.0 megabytes
BL = Chipset type:
= 2Ch (CS69030)
CX = Device ID:
= 0C30h (CS69030)
EDX = BIOS version number:
Bits 31-16 = Frozen BIOS OEM code and version number
Bits 15-0 = Generic BIOS version number (decimal, e.g. 205 = 2.05)
SI = Product and Chipset code

5F02h – Set Dot and Memory Clocks

This function is used to program the given internal clock synthesizer with a given frequency. If BH is set to 0FFh, dot clock 2 and memory clock are set to BIOS default values ignoring the value in BL.

Calling Registers:

AX = 5F02h, Set Dot and Memory Clocks function
BH = Clock to set:
= 00h, Dot clock 0
= 01h, Dot clock 1
= 02h, Dot clock 2
= 03h, Memory clock
= FFh, Program BIOS default dot clock 2 and memory clock
BL = Clock value:
= 6 to 150 MHz (decimal)

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F04h – Get Refresh Rate

This function returns current vertical refresh rate and available refresh rates information for a given mode.

Note: This function returns a status of supported but failed (AX = 005Fh) if executed with a standard VGA mode.

Calling Registers:

AX = 5F04h, Get Refresh Rate function
BL = Mode number

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
BL = Available refresh rates (indicated by one or more bits set):
Bit 0 = Interlaced
Bit 1 = 56Hz
Bit 2 = 60Hz
Bit 3 = 70Hz
Bit 4 = 72Hz
Bit 5 = 75Hz
Bit 6 = 85Hz
Bit 7 = 100Hz
BH = Current refresh rate (see BL for bit definitions)

5F05h – Set Refresh Rate

This function sets a new vertical refresh rate for a given mode. First not depending on the mode being currently active, an internal flag is set to remember the refresh rate for the given mode. If the mode is currently active, the CRT controller and other, registers will be automatically programmed setting the requested refresh rate.

Note: The refresh rates may be set for other modes that have been grouped by the BIOS with the given mode. For example, normally the BIOS groups modes with the same pixel resolutions. However, the correct mode must be currently active for the refresh rate registers to be automatically programmed.

Note: The refresh rates are effective in CRT display modes only and may not be automatically programmed for other displays.

Note: This function returns a status of supported but failed (AX = 005Fh) if executed with a standard VGA mode.

Calling Register:

AX = 5F05h, Set Refresh Rate function
BL = Mode Number
BH = Set refresh rate (indicated by setting one bit):
Bit 0 = Interlaced
Bit 1 = 56Hz
Bit 2 = 60Hz
Bit 3 = 70Hz
Bit 4 = 72Hz
Bit 5 = 75Hz
Bit 6 = 85Hz
Bit 7 = 90Hz

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F10h – Get Linear Display Memory Information

This function returns information regarding the linear memory starting address, size and width.

Note: The information that is returned is for the pipe with register I/O read access.

Calling Register:

AX = 5F10h, Get Linear Display Memory Information function

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
BX:CX = Display memory base address (High:Low)
SI:DI = Display memory size (High:Low)
DX = Display width in bytes

5F11h – Get Memory Map I/O Information

This function returns information regarding memory mapped I/O on a PCI configuration.

Note: The information returned is for the pipe with register I/O read access.

Calling Registers:

AX = 5F11h, Get Memory Mapped I/O Information function

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
BX:CX = Memory Mapped I/O Base Address (High:Low)

5F12h – Get Video Memory Information

This function returns the amount of video memory available for the current video mode and all video modes. Available video memory is often less than the total video memory due to memory used by dual STN panels, popup icons and others.

Calling Registers:

AX = 5F12h, Get Video Memory Information function

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
EBX = Available memory for any mode in bytes
ECX = Available memory for current state in bytes

6.1 5F14h – Hardware Pop-Up Support

This function is made up of several sub-functions to set, get information about and configure the hardware pop-up (icon).

Note: The number of pop-ups (1 to 8) and the size of a pop-up (1K bytes or 2K bytes) are set through the BIOS modification program.

5F14h, 00h – Set Pop-Up Memory Mode

This sub-function sets the pop-up memory mode. The video BIOS saves the necessary registers in the 32-byte buffer passed by the SMI, and then sets up registers for dumping the pop-up bit map into the off-screen video memory.

Note: If the BLTer is in System to Screen BLT mode, the video BIOS may return with an error to indicate that video memory cannot be accessed at this time, and that the SMI handler should exit SMI mode in this situation.

Note: If the function is successful, then the SMI must call the Video BIOS with AX = 5F14h, BL = 01h to restore video controller registers after the SMI loads the pop-up screen into the video memory.

Calling Registers:

AX = 5F14h, Hardware Pop-Up Support function
BL = 00h, Set Pop-Up Memory Mode sub-function
ES:DX = Pointer to 32 byte buffer for Video BIOS in SMI

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
BH = Total number of pop-up screens supported minus one (00h – 07h)

BL = 00h, Function special error (system to screen BLT mode)
= 01h, Function successful
CX = Size of each pop-up screen in bytes
ES:DI = Pointer to the first pop-up screen memory @ A000h

5F14h, 01h – Reset Pop-Up Memory Mode

This sub-function resets the pop-up memory mode. The Video BIOS restores all the registers changed by the *Set Pop-Up Memory Function* from the buffer passed by the SMI.

Calling Registers:

AX = 5F14h, Hardware Pop-Up Support function
BL = 01h, Reset Pop-Up Memory Mode sub-function
ES:DX = Pointer to 32 byte buffer for Video BIOS in SMI

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F14h, 02h – Enable Pop-Up

This sub-function enables a selected type of pop-up.

Calling Registers:

AX = 5F14h, Hardware Pop-Up Support function
BL = 02h, Enable Pop-Up sub-function
CX = Pop-up configuration:
Bits 14-15 = Reserved
Bits 13-12 = Cursor Position:
= 00, Upper Left corner
= 01, Bottom Left corner
= 11, Bottom Right corner
Bits 11-9 = Pop-Up Type:
= 000, Pop-Up Disabled (default)
= 001, 32 x 32 x 2 bpp (AND / XOR)
= 010, 128 x 128 x 1 bpp (2-color)
= 011, 128 x 128 x 1 bpp (1-color + transparency)
= 100, 64 x 64 x 2 bpp (3-color + transparency)
= 101, 64 x 64 x 2 bpp (AND / XOR)
= 110, 64 x 64 x 2bpp (4-color)
Bits 8-3 = Reserved
Bits 2-0 = Enable Pop-Up Number minus one (00h – 07h)
ES:DX = RGB Pointer to 32 byte buffer for Video BIOS in SMI Bytes:
Bits 31-12 = Video BIOS data area in SMI
Bits 11-9 = Cursor Color 3
Bits 8-6 = Cursor Color 2
Bits 5-3 = Cursor Color 1
Bits 2-0 = Cursor Color 0

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed

= 015Fh, Function supported and successful

5F14h, 03h – Disable Pop-Up

This sub-function disables a pop-up.

Calling Registers:

AX = 5F14h, Hardware Pop-Up Support function
BL = 03h, Disable Pop-Up sub-function
ES:DX = Pointer to 32 byte buffer for Video BIOS in SMI

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F14h, 04h – Get Pop-Up Memory Offset

This sub-function returns pop-up memory offset. This offset should be added to the video memory start address to get the absolute pop-up memory address.

Calling Registers:

AX = 5F14h, Hardware Pop-Up Support function
BL = 04h, Get Pop-up Memory Offset sub-function

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
BX:DI = Pop-up memory address offset (High)
CX = Size of each pop-up screen in bytes
DL = Number of pop-ups supported minus one (00h – 07h)

5F14h, 05h – Set Pop-up X and Y Position

This sub-function specifies the pop-up position on the screen.

Calling Registers:

AX = 5F14h, Hardware Pop-Up Support function
BL = 05h, Set Pop-up X and Y Position sub-function
CX = X Position
DX = Y Position

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

6.2 5F19h – Analog TV Support

This function will set or get NTSC (National Television System Committee) or PAL (Phase Alternation Line) supported features. NTSC and PAL are analog television standards that will allow the video chipset output to be displayed on a television or other display that supports these standards.

Note: For NTSC or PAL modes to be active, the system must be set to CRT display and set in a NTSC / PAL supported mode (03h, 12h, 13h, and all extended modes).

5F19h, 00h – Get NTSC / PAL Support

This sub-function will get the state of the NTSC / PAL flags and the current NTSC / PAL display mode. The NTSC / PAL flags (BH) show the mode that will be set when in a supported NTSC / PAL adapter state. These flags allow the video BIOS to appropriately activate or deactivate a NTSC or PAL mode without external intervention. The current NTSC / PAL mode (BL) shows which, if any, NTSC or PAL mode is active.

Calling Registers:

```
AX = 5F19h, Analog TV Support function
BL = 00h,    Get NTSC / PAL Support sub-function
```

Return Registers:

```
AX = Return Status (function not supported if AL != 5Fh):
   = 005Fh, Function supported but failed
   = 015Fh, Function supported and successful
BH = NTSC / PAL Flag State:
   = 00h, Flags set for no NTSC or PAL (normal display)
   = 01h, Flags set for NTSC
   = 02h, Flags set for PAL
BL = Current NTSC / PAL Mode:
   = 00h, NTSC and PAL inactive
   = 01h, NTSC active
   = 02h, PAL active
```

5F19h, 01h – Set NTSC / PAL Support

This sub-function will set flags to indicate if NTSC, PAL or neither mode should be activated when an appropriate system state is entered. For this reason, this function will return a successful status even if a NTSC or PAL mode is not activated. If the System State at the time of calling this sub-function is an acceptable NTSC or PAL State, the set function will activate NTSC or PAL according to the flags. If the flags are set to disable NTSC and PAL modes, NTSC and PAL modes will be deactivated before the return of this sub-function.

Calling Registers:

```
AX = 5F19h, Analog TV Support function
BL = 01h,    Set NTSC / PAL Support sub-function
BH = NTSC / PAL Flags and Mode:
   = 00h, Disable NTSC and PAL (normal display)
   = 01h, Enable NTSC
   = 02h, Enable PAL
```

Return Registers:

```
AX = Return Status (function not supported if AL != 5Fh):
```

= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F19h, 04h – Get TV Flicker Reduction State

This sub-function gets the current TV flicker Reduction State.

Calling Registers:

AX = 5F19h, Analog TV Support function
BL = 04h, Get TV Flicker Reduction State sub-function

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
BH = Current TV flicker reduction state:
= 00h, Disabled
= 01h, Enabled

5F19h, 05h – Set TV Flicker Reduction State

This sub-function sets the TV flicker Reduction State.

Calling Registers:

AX = 5F19h, Analog TV Support function
BL = 05h, Set TV Flicker Reduction State sub-function
BH = TV flicker reduction state:
= 00h, Disabled
= 01h, Enabled

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F19h, 08h – Get TV Scaling State

This sub-function gets the current TV scaling state.

Calling Registers:

AX = 5F19h, Analog TV Support function
BL = 08h, Get TV Scaling State sub-function

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
BH [7] = TV Scaling (for 480 line modes):
= 0, Disabled
= 1, Enabled
BH[4:0] = Vertical scaling lines skip interval

5F19h, 09h – Set TV Scaling State

This sub-function sets the current TV scaling state.

Note: One line is dropped off for every BH [4:0] lines.

Calling Registers:

AX = 5F19h, Analog TV Support function
BL = 09h, Set TV Scaling State sub-function
BH [7] = TV Scaling (for 480 line modes):
= 0, Disabled
= 1, Enabled
BH[4:0] = Vertical scaling lines skip interval
= 00000, Default
= 00001, Reserved
= 00010 – 11111, Vertical scaling lines skip interval

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F1Ah – GPIO Pins Used by I^{2}C Busses

This function returns the GPIO pins that are used by I^{2}C bus hardware applications.

Calling Registers:

AX = 5F1Ah, GPIO Pins Used by I^{2}C Busses function

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
BH = I^{2}C bus SDA GPIO number
= FFh, No I^{2}C bus SDA GPIO number
= Feh – 00h, Valid I^{2}C bus SDA GPIO number
BL = I^{2}C bus SCL GPIO number
= FFh, No I^{2}C bus SCL GPIO number
= Feh – 00h, Valid I^{2}C bus SCL GPIO number
CH = DDC SDA GPIO number
= FFh, No BIOS DDC1 or DDC2 support
= Feh – 00h, Valid DDC SDA GPIO number
CL = DDC SCL GPIO number
= FFh, No BIOS DDC2 support
= Feh – 00h, Valid DDC SCL GPIO number

6.3 5F1Ch – Pipe Register Access

This function will set or get the pipe register access through IOSS and MSS.

5F1Ch, 00h – Set Pipe Register Access

This sub-function will set IOSS and MSS as needed to set the requested pipe register access.

Note: Setting both bits on CL will allow writing a register value to both pipes at the same time. Reading a register value can only happen from one pipe at a time for obvious reasons.

Calling Registers:

AX = 5F1Ch, Pipe Register Access function
BH = 00h, Set Pipe Register Access sub-function
CH = Pipe with Read access:
= 00h, Pipe A
= 01h, Pipe B
CL = Pipe with Write access:
Bits 7-2 = Reserved
Bit 1 = Pipe B
Bit 0 = Pipe A

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F1Ch, 01h – Get Pipe Register Access

This sub-function will get the current pipe register access.

Calling Registers:

AX = 5F1Ch, Pipe Register Access function
BH = 01h, Get Pipe Register Access sub-function

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
CH = Pipe with Read access:
= 00h, Pipe A
= 01h, Pipe B
CL = Pipe with Write access:
Bits 7-2 = Reserved
Bit 1 = Pipe B
Bit 0 = Pipe A

5F1Dh – Hardware Adjustments

This function executes code to prevent or fix a hardware problem.

Calling Registers:

AX = 5F1Dh, Hardware Adjustments function
CX = Hardware adjustment number
= 0000h, Standard hardware adjustments
= 0001h – FFFFh, Special hardware adjustments

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

6.4 5F1Eh – Suspend and Resume Video Chipset

This function is called by the system BIOS to allow the video BIOS control before the #STDBY pin is set on a suspend state change and after the #STDBY pin is reset on a resume from suspend state change. This control can be used, if needed, to fix hardware problems with the suspend and resume procedure.

5F1Eh, 00h – Before Low Power Suspend

This sub-function allows the video BIOS control when entering a suspend state.

Note: This sub-function should be called by the system BIOS just before setting the #STDBY pin to low power suspend state.

Calling Registers:

AX = 5F1Eh, Suspend and Resume Video Chipset function
BH = 00h, Before Low Power Suspend sub-function

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F1Eh, 01h – After Low Power Resume from Suspend

This sub-function allows the video BIOS when resuming from a suspend state.

Note: This sub-function should be called by the system BIOS just after resetting the #STDBY pin to the on state when resuming from a suspended state.

Calling Registers:

AX = 5F1Eh, Suspend and Resume Video Chipset function
BH = 01h, After Low Power Resume from Suspend sub-function

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F22h – Get Mode Support Information

This function returns display types that are supported by the given or current mode. It also returns the current display type. These two pieces of information can be used to determine if a mode is supported on the current display type.

Note: The functionality of this function has been replaced by the more powerful function (AX = 5F28h). This function will continue to be supported for legacy software.

Note: This function assumes that Pipe B is set as off.

Calling Registers:

AX = 5F22h, Get Mode Support Information function
BH = Mode to use for pipe A:
= 00h, Current Mode
= 01h, Mode specified in BL
BL = Mode Number

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
BH = Current Display Mode:
Bit 0 = CRT display type
Bit 1 = Flat panel display type
Bit 2 = Simultaneous display type
BL = Mode Support Information:
Bit 0 = Mode supported in CRT display type
Bit 1 = Mode supported in flat panel display type
Bit 2 = Mode supported in simultaneous display type
CX = Mode Horizontal (x) Resolution in Pixels
DX = Mode Vertical (y) Resolution in Pixels

5F24h – Limited Set Mode

This function is used by Windows 3.1 drivers to fix a bug in Windows 3.1.

Calling Registers:

AX = 5F24h, Limited Set Mode function
BL = Mode Number

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

6.5 5F26h – I^{2}C Bus Support

This function will help an application read and write over an I^{2}C bus. Extra I^{2}C bus functionality has been developed to allow for many different I^{2}C bus solutions.

5F26h, 00h – Initialize Before I^{2}C Bus Functions

This sub-function will initialize the System State, if necessary, before any I^{2}C bus functions are executed. This does not to be run before every I^{2}C function, only once before the first function is executed.

Calling Registers:

AX = 5F26h, I^{2}C Bus Support function
BH = 00h, Initialize Before I^{2}C Bus Functions sub-function
CH = SCL GPIO pin number
CL = SDA GPIO pin number

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F26h, 01h – Reset After I^2C Bus Functions

This sub-function will reset the System State, if necessary, after executing I^2C bus functions. This does not to be run after every I^2C function, only once after the last function is executed.

Calling Registers:

AX = 5F26h, I^2C Bus Support function
BH = 01h, Reset After I^2C Bus Functions sub-function
CH = SCL GPIO pin number
CL = SDA GPIO pin number

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F26h, 02h – I^2C Bus Start Cycle

This sub-function performs an I^2C bus start cycle.

Calling Registers:

AX = 5F26h, I^2C Bus Support function
BH = 02h, I^2C Bus Start Cycle sub-function
CH = SCL GPIO pin number
CL = SDA GPIO pin number

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F26h, 03h – I^2C Bus Stop Cycle

This sub-function performs an I^2C bus stop cycle.

Calling Registers:

AX = 5F26h, I^2C Bus Support function
BH = 03h, I^2C Bus Stop Cycle sub-function
CH = SCL GPIO pin number
CL = SDA GPIO pin number

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F26h, 04h – Read I^{2}C Bus Byte

This sub-function reads a byte from an I^{2}C bus.

Calling Registers:

AX = 5F26h, I^{2}C Bus Support function
BH = 04h, Read I^{2}C Bus Byte sub-function
BL != 01h, Send ACK bit after reading byte
== 01h, Send NAK bit after reading byte
CH = SCL GPIO pin number
CL = SDA GPIO pin number

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
BL = Byte read

5F26h, 05h – Write I^{2}C Bus Byte

This sub-function writes a byte to an I^{2}C bus.

Calling Registers:

AX = 5F26h, I^{2}C Bus Support function
BH = 05h, Write I^{2}C Bus Byte sub-function
BL = Byte to write
CH = SCL GPIO pin number
CL = SDA GPIO pin number

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F26h, 06h – Set I^{2}C Bus Clock High

This sub-function sets the I^{2}C bus clock (SCL) pin high.

Calling Registers:

AX = 5F26h, I^{2}C Bus Support function
BH = 06h, Set I^{2}C Bus Clock High sub-function
CH = SCL GPIO pin number

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F26h, 07h – Set I^{2}C Bus Clock Low

This sub-function sets the I^{2}C bus clock (SCL) pin low.

Calling Registers:

AX = 5F26h, I^{2}C Bus Support function
BH = 07h, Set I^{2}C Bus Clock Low sub-function
CH = SCL GPIO pin number

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F26h, 08h – Read I^{2}C Bus Pin

This sub-function reads the requested I^{2}C bus pin.

Calling Registers:

AX = 5F26h, I^{2}C Bus Support function
BH = 08h, Read I^{2}C Bus Pin sub-function
BL [4] = I^{2}C bus pin to read:
= 0, SDA
= 1, SCL
CH = SCL GPIO pin number
CL = SDA GPIO pin number

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
BL = I^{2}C bus pin level
= 00h, Low
= 01h, High

5F26h, 09h – Write I^{2}C Bus Pin

This sub-function writes the requested I^{2}C bus pin to a given value.

Calling Registers:

AX = 5F26h, I^{2}C Bus Support function
BH = 09h, Write I^{2}C Bus Pin sub-function
BL [0] = I^{2}C bus pin level
= 0, Low
= 1, High
BL [4] = I^{2}C bus pin to write:
= 0, SDA
= 1, SCL
CH = SCL GPIO pin number
CL = SDA GPIO pin number

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F28h – Check Mode Support

This function determines if the given state (i.e. pipe modes and display device) can be supported by the hardware and BIOS.

Note: Maximum adjusted memory bandwidth is the video memory bandwidth minus a percentage that is required for user to set video memory through the CPU.

Note: This function does not take into account overlay or any other functionality that may draw video memory bandwidth from the system. Maximum pipe A and pipe B bandwidths are supplied as outputs to allow an user to determine if there is enough bandwidth remaining to support other features that draw bandwidth resources.

Calling Registers:

```
AX  = 5F28h, Get Mode Support function
BH  = Pipe B Mode To Use:
    = 80h, Current Mode
    = 00-7Fh, Given Mode Number
BL  = Pipe A Mode to Use:
    = 80h, Current Mode
    = 00-7Fh, Given Mode Number
CX  = Display Device Combination:
    Bit 15      = Display device to use:
                = 0, Given display in bits 14-0
                = 1, Current display
    Bits 14-11  = Pipe B – Reserved
    Bit 10      = Pipe B – TV Analogue (If set bits 0 and 8 can not be set)
    Bit 9       = Pipe B – Reserved
    Bit 8       = Pipe B – CRT (If set bits 0 and 10 can not be set)
    Bits 7-4    = Pipe A – Reserved
    Bit 3       = Pipe A – TV Digital (If set bit 1 can not be set)
    Bit 2       = Pipe A – Reserved
    Bit 1       = Pipe A – Flat Panel (If set bit 3 can not be set)
    Bit 0       = Pipe A – CRT (If set bits 8 and 10 can not be set)
```

Return Registers:

```
AX  = Return Status (function not supported if AL != 5Fh):
    = 005Fh, Function supported but failed
    = 015Fh, Function supported and successful
BH  = Requested State:
    = 00h, is not supported
    = 01h, is supported
CX  = Maximum adjusted Memory Bandwidth in Megabytes per second
EDX = Pipe Bandwidth in Megabytes per second:
    Bits 31-16 = Pipe B
    Bits 15-0  = Pipe A
```

5F29h – Get Mode Information

This function returns the requested mode's resolutions, color depth, and maximum required bandwidth using its current refresh rate.

Calling Registers:

```
AX  = 5F29h, Get Mode Information function
```

BH = Mode To Use:
= 80h, Current Mode
= 00-7Fh, Given Mode Number

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
EBX bits 31-16= Mode horizontal (X) resolution in pixels

EBX bits 15-0 = Mode vertical (Y) resolution in pixels

ECX bits 31-16= Maximum bandwidth in megabytes per second
ECX bits 15-0 = Color depth in bits per pixel

5F50h – Get Display Information

This function returns current display device information.

Calling Registers:

AX = 5F50h, Get Display Information function

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
BX = Flat Panel Horizontal size in pixels
CX = Flat Panel Vertical size in pixels
DX = F69000 status:
Bits 15-12 = Reserved
Bit 11 = Support NTSC / PAL in Applications:
= 0, Disabled
= 1, Enabled
Bits 10-5 = Reserved
Bit 4 = Drivers check for CRT Panning:
= 0, Disabled
= 1, Enabled
Bit 3 = Simultaneous (CRT and Flat Panel) support:
= 0, Disabled
= 1, Enabled
Bit 2 = Video polarity state:
= 0, Normal
= 1, Inverted
Bit 1 = Current display device:
= 0, CRT
= 1, Flat Panel
Bit 0 = Panel Type:
= 0, TFT
= 1, STN

5F51h – Switch Display Device

This function switches between pipe A CRT, flat panel, and simultaneous displays. The current display device can be retrieved from the Get Display Device function (5F64h, 01h).

Note: System using TV Out must assure that the NTSC / PAL is disabled (5F19) when entering CRT only state. Otherwise, the system may enter the TV Out State.

Note: The functionality of this function has been replaced by the more powerful switch display function (AX = 5F64h). This function will continue to be supported for legacy software.

Note: This function will disable any display on pipe B and set the requested pipe A display.

Calling Registers:

AX = 5F51h, Switch Display Device function
BL = Pipe A Display Device Options:
= 00, Switch to CRT
= 01, Switch to Flat Panel
= 02, Switch to Simultaneous
= 03, If CRT attached toggle to next display state: LCD → CRT → Simultaneous → LCD

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F54h – Set Panel ON / OFF

This function sets the panel ON or OFF. Power consumption is reduced in Panel OFF Mode.

Calling Registers:

AX = 5F54h, Set Panel ON / OFF function
BL = Power Down Mode:
= 00h, Panel ON
= 01h, Panel OFF

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F55h – Monitor Detect

This function detects if a monitor (CRT) is currently attached to the adapter, and if one is found, determines whether it is color or monochrome.

Note: Due to Windows 95 trapping status registers 00h and 01h, this function may not work in a Windowed DOS box. A full screen DOS box and Windows 95 should work fine.

Calling Registers:

AX = 5F55h, Monitor Detect function

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
BL = Monitor Type:
= 00h, Color CRT detected
= 01h, Monochrome CRT detected

= 02h, No CRT detected

5F56h – Get Panel Type

This function is used to return panel type information.

Calling Registers:

AX = 5F56h, Get Panel Type function

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
BL = Internal panel number minus one (0 – 15)

5F5Ah – Set Flat Panel Video Polarity

This function sets the polarity of the video output to the flat-panel.

Calling Registers:

AX = 5F5Ah, Set Flat Panel Video Polarity function
BL = Flat Panel Video Polarity:
= 00h, Normal polarity
= 01h, Inverted polarity
= 02h, Toggle polarity

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F5Bh – Set Horizontal Compensation

This function enables or disables Horizontal Graphics Expansion, Text Font Expansion and Centering.

Calling Registers:

AX = 5F5Bh, Set Horizontal Compensation function
BL = Horizontal compensation state:
= 00h, Enable Horizontal Text Compensation
= 01h, Disable Horizontal Text Compensation
= 02h, Enable Horizontal Centering
= 03h, Disable Horizontal Centering
= 04h, Enable Horizontal Graphics Compensation
= 05h, Disable Horizontal Graphics Compensation
= 06h, Set 8-dot to 8-dot and 9-dot to 8-dot
= 07h, Set 8-dot to 9-dot and 9-dot to 9-dot
= 08h, Set 8-dot to 10-dot and 9-dot to 10-dot
= 09h, Enable Horizontal Graphics Compensation in 640 Column Mode
= 0Ah, Disable Horizontal Graphics Compensation in 640 Column Mode
= 0Bh, Enable Horizontal Graphics Compensation in 800 Column Mode
= 0Ch, Disable Horizontal Graphics Compensation in 800 Column Mode
= 0Dh, Enable Horizontal Graphics Compensation in 1024 Column Mode
= 0Eh, Disable Horizontal Graphics Compensation in 1024 Column Mode

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F5Eh – Set Vertical Compensation

This function enables or disables vertical text stretching and centering, and line replication. Use function 5F50 to get compensation status.

Calling Registers:

AX = 5F5Eh, Set Vertical Compensation function
BL = Vertical compensation state:
= 00h, Enable Vertical Text Compensation
= 01h, Disable Vertical l Text Compensation
= 02h, Enable Vertical Centering
= 03h, Disable Vertical Centering
= 04h, Enable Vertical Graphics Compensation
= 05h, Disable Vertical Graphics Compensation
= 06h, Enable Vertical Text Compensation in 350 Line
= 07h, Disable Vertical Text Compensation in 350 Line
= 08h, Enable Vertical Text Compensation in 400 Line
= 09h, Disable Vertical Text Compensation 400 Line
= 0Ah, Enable Vertical Graphics Compensation 350 Line
= 0Bh, Disable Vertical Graphics Compensation 350 Line
= 0Ch, Enable Vertical Graphics Compensation 400 Line
= 0Dh, Disable Vertical Graphics Compensation 400 Line
= 0Eh, Enable Vertical Graphics Compensation 480 Line
= 0Fh, Disable Vertical Graphics Compensation 480 Line
= 10h, Enable Vertical Graphics Compensation 600 Line
= 11h, Disable Vertical Graphics Compensation 600 Line

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

6.6 5F60h – FP Low Power On State

This function sets or gets the flat panel low power on state. The flat panel low power state is a BIOS state where special low power (slower) memory clocks are used instead of the normal high performance (faster) memory clocks are used on a flat panel only display device. A software flag is use to remember the flat panel power state which allows this function to be called successfully from a display device.

Note: The values of this set of flat panel low power memory clocks can be adjusted through the BIOS modification utility.

5F60h, 00h – Set FP Low Power State

The sub-function sets the flat panel low power on state.

Calling Registers:

AX = 5F60h, FP Low Power State

BH = 00h, Set FP Low Power State
BL = Select high performance verses low power:
= 00h, High performance
= 01h, Low power

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F60h, 01h – Get FP Low Power State

This sub-function gets the current flat panel low power on state.

Calling Registers:

AX = 5F60h, FP Low Power State
BH = 01h, Get FP Low Power State

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
BL = High performance or low power state:
= 00h, High performance
= 01h, Low power

6.7 5F61h – Horizontal and Vertical Compensation

This function has a set and get sub-function to control the Horizontal and vertical compensation features.

Note: For future compatibility, call the Get function before calling the Set function, and then change the desired components.

5F61h, 00h – Set Horizontal and Vertical Compensation

This function sets the horizontal and vertical compensation components.

Calling Registers:

AX = 5F61h, Horizontal and Vertical Compensation function
BH = 00h, Set Horizontal and Vertical Compensation sub-function
CX = Horizontal compensation (1 = Enable, 0 = Disable)
Bit 0 = Centering
Bit 1 = Text Stretching
Bit 2 = Graphics Stretching
DX = Vertical compensation (1 = Enable, 0 = Disable)
Bit 0 = Centering
Bit 1 = Text Stretching
Bit 2 = Graphics Stretching

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F61h, 01h – Get Horizontal and Vertical Compensation

This function gets the horizontal and vertical compensation components.

Calling Registers:

```
AX  = 5F61h, Horizontal and Vertical Compensation function
BH  = 01h,     Get Horizontal and Vertical Compensation sub-function
```

Return Registers:

```
AX  = Return Status (function not supported if AL != 5Fh):
    = 005Fh, Function supported but failed
    = 015Fh, Function supported and successful
CX  = Horizontal compensation (1 = Enable, 0 = Disable)
    Bit 0 = Centering
    Bit 1 = Text Stretching
    Bit 2 = Graphics Stretching
DX  = Vertical compensation (1 = Enable, 0 = Disable)
    Bit 0 = Centering
    Bit 1 = Text Stretching
    Bit 2 = Graphics Stretching
```

5F63h – Adapter Power State

This function sets up the video controller for the power state switch if needed and notifies the system BIOS of the requested power state switch. A table to convert the ACPI power state names to the APM power state names is as follows.

D0 = On
D1 = Standby
D2 = Suspend
D3 = Off

Calling Registers:

```
AX  = 5F63h, Adapter Power State
BL  = Power state switch requested:
    = 01h, D0 to D1
    = 02h, D0 to D2
    = 03h, D0 to D3
    = 04h, D1 to D0
    = 05h, D2 to D0
    = 06h, D3 to D0
```

Return Registers:

```
AX  = Return Status (function not supported if AL != 5Fh):
    = 005Fh, Function supported but failed
    = 015Fh, Function supported and successful
```

6.8 5F64h – Switch Display Device

This function has set and get sub-functions that allow control of the Display devices including displays on either pipe A or pipe B.

5F64h, 00h – Set Display Device

This sub-function sets the given display device combination. The current display device combination can be retrieved from the Get Display Device function (5F64h, 01h).

Note: The settable display device combination may be restricted when the system is in a Mosaic mode (see 5F64h, 02h).

Note: This function may or may not execute if the given input display device is the same as the current active display device. This function can be forced to execute by setting BL bit 0.

Calling Registers:

AX = 5F64h, Switch Display Device function
BH = 00h, Set Display Device sub-function
BL = Set Display Device Options:
Bits 7-3 = Reserved
Bit 0 = Force complete execution even if same display device
CX = Display Device Combination to Set (1 = Enable display, 0 = Disable display):
Bits 15-11 = Pipe B – Reserved
Bit 10 = Pipe B – TV Analogue (If set bits 0 and 8 can not be set)
Bit 9 = Pipe B – Reserved
Bit 8 = Pipe B – CRT (If set bits 0 and 10 can not be set)
Bits 7-4 = Pipe A – Reserved
Bit 3 = Pipe A – TV Digital (If set bit 1 can not be set)
Bit 2 = Pipe A – Reserved
Bit 1 = Pipe A – Flat Panel (If set bit 3 can not be set)
Bit 0 = Pipe A – CRT (If set bits 8 and 10 can not be set)

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F64h, 01h – Get Display Device

This sub-function gets the current display device combination.

Calling Registers:

AX = 5F64h, Switch Display Device function
BH = 01h, Get Display Device sub-function

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
CX = Current Display Device Combination (1 = Active display):
Bits 15-11 = Pipe B – Reserved
Bit 10 = Pipe B – TV Analogue
Bit 9 = Pipe B – Reserved
Bit 8 = Pipe B – CRT
Bits 7-4 = Pipe A – Reserved
Bit 3 = Pipe A – TV Digital
Bit 2 = Pipe A – Reserved
Bit 1 = Pipe A – Flat Panel
Bit 0 = Pipe A – CRT

5F64h, 02h – Set Mosaic Mode

This sub-function informs the BIOS that mosaic mode has been enabled or disabled.

Calling Registers:

AX = 5F64h, Switch Display Device function
BH = 02h, Set Mosaic Mode sub-function
BL = Mosaic mode state to set:
= 00h, Disabled
= 01h, Enabled

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F64h, 03h – Get Mosaic Mode

This sub-function gets the current state of mosaic mode.

Calling Registers:

AX = 5F64h, Switch Display Device function
BH = 03h, Get Mosaic Mode sub-function

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
BL = Mosaic mode state:
= 00h, Disabled
= 01h, Enabled

5F64h, 04h – Copy Pipe

This sub-function will copy the register values from one pipe to another.

Calling Registers:

AX = 5F64h, Switch Display Device function
BH = 04h, Copy Pipe sub-function
BL = Copy register values:
= 00h, From Pipe B to Pipe A
= 01h, From Pipe A to Pipe B

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

7 VESA BIOS Extension Interface Functions

The video BIOS supports the VESA (Video Electronics Standards Association) specifications VESA VBE (VESA BIOS Extensions) version 2.0, VESA VBE / PM (Power Management) version 1.0 and VESA VBE / DDC (Display Data Channel) version 1.0. The VBE interface functions provide a standardized method for accessing resolutions, color depths, and frame buffer organizations which do not exist under the VGA standard. The VBE / PM interface functions provide a standardized method for display power management. Finally, the VBE / DDC interface functions provide a standardized method retrieving the EDID block of data from a display. This is a component of Plug and Play monitor.

Important Note: IOSS (see function 5F1Ch) must be set to the appropriate values before calling any BIOS function.

Note: If there is any discrepancy between the description of a feature or function in this document, and the description of the same feature or function in a VESA counterpart specification, the VESA specification will take precedence, and this document should be fixed to reflect that. These functions have been included in this chapter for convenience as a video BIOS reference manual.

Table 7.1 VESA Extended VGA BIOS Functions

Function	Function Name	Function Description (Short)
4F00h	Return VBE Controller Information	Returns VBE revision and capability information.
4F01h	Return VBE Mode Information	Returns information about a given VBE mode.
4F02h	Set VBE Mode	Sets a given VBE mode.
4F03h	Get Current VBE Mode	Returns the current VBE mode number.
4F04h, 00h	Return Save / Restore State Buffer Size	Returns buffer size required to hold the save and restore state data.
4F04h, 01h	Save State	Loads the given buffer with the requested state data.
4F04h, 02h	Restore State	Restores a previously saved state from the given buffer.
4F05h, 00h	Set Memory Window	Sets the position of the specified display window.
4F05h, 01h	Get Memory Window	Gets the position of the specified display window.
4F06h, 00h	Set Scan Line Length in Pixels	Sets the logical scan line length in pixels.
4F06h, 01h	Get Scan Line Length	Gets the logical scan line length.
4F06h, 02h	Set Scan Line Length in Bytes	Sets the logical scan line length in bytes.
4F06h, 03h	Get Maximum Scan Line Length	Gets the maximum logical scan line length possible.
4F07h, 00h	Set Display Start	Sets the pixel to be displayed in the upper left corner of the display.
4F07h, 01h	Get Display Start	Gets the pixel displayed in the upper left corner of the display.
4F07h, 80h	Set Display Start During Vertical Retrace	Sets the pixel to be displayed in the upper left corner of the display during a vertical retrace period to prevent line tearing.
4F08h, 00h	Set DAC Palette Format	Sets the operating mode or format of the DAC palette.
4F08h, 01h	Get DAC Palette Format	Gets the operating mode or format of the DAC palette.
4F09h, 00h	Set Palette Data	Sets color registers in the RAMDAC.

VESA

Function	Function Name	Function Description (Short)
4F09h, 01h	Get Palette Data	Gets color registers in the RAMDAC.
4F0Ah	VBE 2.0 Protected Mode Interface	Returns code for a 32-bit protected mode interface.
4F10h, 00h	Report VBE / PM Capabilities	Returns the power management capabilities of the video controller.
4F10h, 01h	Set Display Power State	Sets the display power state.
4F10h, 02h	Get Display Power State	Gets the display power state.
4F15h, 00h	Report DDC Capabilities	Returns the DDC capabilities of both video controller and display.
4F15h, 01h	Read EDID	Reads an EDID block from an attached display.

7.1 VBE Return Status

The AX register is universally used by all VESA VBE functions to indicate the return status of the requested function. This definition is as follows:

VESA VBE Return Status:

AX = Return Status (function not supported if AL != 4Fh):
= 004Fh Function call successful
= 014Fh Function call failed
= 024Fh Function is not supported in current hardware configuration
= 034Fh Function call invalid in current video mode

7.2 VESA VBE Core Interface Functions

The VBE specification is for standard software access to graphics display controllers which support resolutions, color depths, and frame buffer organizations beyond the VGA hardware standard. VBE functions are known as core interface functions as it encompasses other VESA specifications. OEM information is also provided by these VBE core functions.

4F00h – Return VBE Controller Information

This function returns VBE revision and capability information. The purpose of this function is to provide information to the calling program about the general capabilities of the installed VBE software and hardware. This function fills an information block structure at the address specified by the caller. The VbeInfoBlock size is 256 bytes for VBE 1.x, and 512 bytes for VBE 2.0.

Note: The VBE signature should be set to “VBE2” by the calling application to indicate VBE 2.0 information is desired and the information block is 512 bytes in size.

Calling Registers:

AX = 4F00h, Return VBE Controller Information function
ES:DI = Pointer to buffer for the VbeInfoBlock

Return Registers:

AX = VESA VBE Return Status (see page 7-2)

Table 7.1 VbeInfoBlock Structure for VESA VBE Function 00h

Name	Size	Description
VbeSignature	4 Bytes	‘VESA’ VBE signature
VbeVersion	Word	0200h VBE version number (2.0)

Name	Size	Description
OemStringPtr	Dword	Pointer to OEM string
Capabilities	4 Bytes	Capabilities of graphics controller: Bit 2 = RAMDAC operation: = 0, Normal = 1, Use blank bit when programming large blocks Bit 1 = Controller is VGA compatible: = 0, Yes = 1, No Bit 0 = DAC width: = Fixed with 6 bits per primary color = Switchable to 8 bits per primary color
VideoModePtr	Dword	Pointer to video mode list (see video mode appendix for supported VBE modes page A-1) **Note:** The listed mode numbers represent potentially supported modes. An application must verify these modes through VBE function 01h.
TotalMemory	Word	Number of 64KB memory blocks (all physically installed memory)
OemSoftwareRev	Word	VBE implementation software revision
OemVendorNamePtr	Dword	Pointer to vendor name string
OemProductNamePtr	Dword	Pointer to product name string
OemProductRevPtr	Dword	Pointer to product revision string
Reserved	222 Bytes	Reserved for VBE implementation scratch area
OemData	256 Bytes	Data for OEM strings

4F01h – Return VBE Mode Information

This function returns information about a given VBE mode in the supplied memory buffer.

Calling Registers:

AX = 4F01h, Return VBE Mode Information function
CX = VBE mode number
ES:DI = Pointer to a 256 byte buffer for the ModeInfoBlock

Return Registers:

AX = VESA VBE Return Status (see page 7-2)

Table 7.2 ModeInfoBlock Structure for VESA VBE Function 01h

Name	Size	Description

Name	Size	Description
ModeAttributes	Word	Mode attributes: Bit 7 = Linear frame buffer mode is available: = 0, No = 1, Yes Bit 6 = VGA compatible windowed memory mode is available: = 0, Yes = 1, No Bit 5 = VGA compatible mode: = 0, Yes = 1, No Bit 4 = Mode type: = 0, Text mode = 1, Graphics Mode Bit 3 = Monochrome / color mode = 0, Monochrome mode = 1, Color mode Bit 2 = TTY output functions supported by BIOS: = 0, No = 1, Yes Bit 1 = 1, Reserved Bit 0 = Mode supported by hardware configuration = 0, No = 1, Yes
WinAAttributes	Byte	Window A attributes: Bit 2 = Window is writeable: = 0, No = 1, Yes Bit 1 = Window is readable: = 0, No = 1, Yes Bit 0 = Relocatable window(s) supported: = 0, No = 1, Yes
WinBAttributes	Byte	Window B attributes (see Window A attributes)
WinGranularity	Word	Window granularity in K bytes
WinSize	Word	Window size in K bytes
WinASegment	Word	Window A start segment
WinBSegment	Word	Window B start segment
WinFuncPtr	DWord	Pointer to window function (segment:offset) **Note:** This provides a fast direct call to the paging registers.
BytesPerScanLine	Word	Bytes per scan line
XResolution	Word	Horizontal resolution
YResolution	Word	Vertical resolution
XCharSize	Byte	Character cell width
YCharSize	Byte	Character cell height
NumberOfPlanes	Byte	Number of memory planes
BitsPerPixel	Byte	Bits per pixel
NumberOfBanks	Byte	Number of banks

Name	Size	Description
MemoryModel	Byte	Memory model type: 00h = Text mode 01h = CGA graphics 02h = Hercules graphics 03h = Planar 04h = Packed pixel 05h = Non-chain 4, 256 color 06h = Direct color 07h = YUV
BankSize	Byte	Bank size in K bytes
NumberOfImagePages	Byte	Number of images minus one
Reserved	Byte	1 Reserved for page function
RedMaskSize	Byte	Size if direct color red mask in bits
RedFieldPosition	Byte	Bit position of lsb of red mask
GreenMaskSize	Byte	Size of direct color green mask in bits
GreenFieldPosition	Byte	Bit position of lsb of green mask
BlueMaskSize	Byte	Size of direct color blue mask in bits
BlueFieldPosition	Byte	Bit position of lsb of blue mask
RsvdMaskSize	Byte	Size of direct color reserved mask in bits
RsvdFieldPosition	Byte	Bit position of lsb of reserved mask
DirectColorModeInfo	Byte	Direct color mode attributes: Bit 1 = Bits in reserved field are: = 0, Reserved = 1, Usable by an application Bit 0 = Color ramp is: = 0, Fixed = 1, Programmable
PhysBasePtr	DWord	Physical address for linear frame buffer
OffScreenMemOffset	DWord	Pointer to start of offscreen memory
OffScreenMemSize	Word	Amount of offscreen memory in 1K units
Reserved	206 Bytes	Remainder of ModeInfoBlock

4F02h – Set VBE Mode

This function sets a given VBE mode. It will also set a seven bit mode number as set with set mode (interrupt 10h, AH = 00h) including standard VGA mode numbers.

Note: Mode 81FFh is a special mode designed to help save video memory before going into a state that could lose its contents. The mode is a paged packed pixel mode.

Calling Registers:

AX = 4F02h, Set VBE Mode function
BX = VBE Mode to set:
Bit 15 = Clear display memory bit:
= 0, Clear memory
= 1, Don't clear memory
Bit 14 = Frame buffer model
= 0, Windowed (paged)
= 1, Linear (flat)
Bits 13-9 = Reserved, (must = 0)

Bits 8-0 = VBE mode number

Return Registers:

AX = VESA VBE Return Status (see page 7-2)

4F03h – Return Current VBE Mode

This function returns the current VBE mode number.

Note: The mode number returned is the same mode number that was last set whether it was set through VBE function 02h or set mode (interrupt 10h, AH = 00h).

Calling Registers:

AX = 4F03h, Return Current Video Mode function

Return Registers:

AX = VESA VBE Return Status (see page 7-2)
BX = VBE Mode to set:
Bit 15 = Clear display memory bit:
= 0, Last set mode cleared memory
= 1, Last set mode did not clear memory
Bit 14 = Frame buffer model
= 0, Windowed (paged)
= 1, Linear (flat)
Bits 13-0= VBE mode number

7.2.1 4F04h – Save / Restore State

This function has three sub-functions to provide a mechanism for saving and restoring selected parts of the Video State. These sub-functions offer a compatible method to save and restore the extended SVGA State. Bit 3 will replace the nonstandard bits 13 and 15 of the save and restore VGA function 1Ch.

4F04h, 00h – Return Save / Restore State Buffer Size

This sub-functions returns the minimum buffer size (in 64 byte blocks) required to hold the data loaded by the save and restore functions. The size of this buffer is updated according to the save / restore options selected in CX.

Calling Registers:

AX = 4F04h, Save / Restore State function
DL = 00h, Return Save / Restore State Buffer Size sub-function
CX = Requested states:
Bit 3 = Extended register state
Bit 2 = DAC state and color registers
Bit 1 = BIOS data area
Bit 0 = Hardware state (VGA registers)

Return Registers:

AX = VESA VBE Return Status (see page 7-2)
BX = Minimum Number of 64 byte blocks to hold the state buffer

4F04h, 01h – Save State

This sub-functions loads the given buffer with the requested state data.

Note: This function does not check the size of the given buffer. Use the Return Save / Restore State Buffer Size function above to get minimum buffer size.

Calling Registers:

AX = 4F04h, Save / Restore State function
DL = 01h, Save State sub-function
CX = Requested states:
Bit 3 = Extended register state
Bit 2 = DAC state and color registers
Bit 1 = BIOS data area
Bit 0 = Hardware state (VGA registers)
ES:BX = Pointer to state buffer

Return Registers:

AX = VESA VBE Return Status (see page 7-2)

4F04h, 02h – Restore State

This sub-functions restores a previously saved state from the given buffer.

Calling Registers:

AX = 4F04h, Save / Restore State function
DL = 02h, Restore State sub-function
CX = Requested states:
Bit 3 = Extended register state
Bit 2 = DAC state and color registers
Bit 1 = BIOS data area
Bit 0 = Hardware state (VGA registers)
ES:BX = Pointer to state buffer

Return Registers:

AX = VESA VBE Return Status (see page 7-2)

7.2.2 4F05h – Display Window Control

This function sets or gets the position of the specified display window or page in the frame buffer memory by adjusting the paging registers.

Note: For performance reasons, it may be more efficient to call this function directly, without incurring the interrupt 10h overhead. VBE function 01h returns a pointer of this windowing function that may be called directly.

4F05h, 00h – Set Memory Window

This sub-function sets the position of the specified display window or page in the frame buffer memory by adjusting the paging registers.

Note: If this function is called in a linear memory model, it will fail with AH = 03h.

Calling Registers:

AX = 4F05h, Display Window Control function
BH = 00h, Set Memory Window sub-function
BL = Window number:

= 00h, Window A
= 01h, Window B (Not supported)
DX = Window position in video memory (in 64 K byte units)

Return Registers:

AX = VESA VBE Return Status (see page 7-2)

4F05h, 01h – Get Memory Window

This sub-function gets the position of the specified display window or page in the frame buffer memory.

Calling Registers:

AX = 4F05h, Display Window Control function
BH = 01h, Get Memory Window sub-function
BL = Window number:
= 00h, Window A
= 01h, Window B (Not supported)

Return Registers:

AX = VESA VBE Return Status (see page 7-2)
DX = Window position in video memory (in 64 K byte units)

7.2.3 4F06h – Logical Scan Line Length

This functions sets or gets the logical scan line length. This allows an application to set up a logical display memory buffer that is wider than the display area. VBE function 07h can then allows the application to set the starting position that is to be displayed.

4F06h, 00h – Set Scan Line Length in Pixels

This sub-functions sets the logical scan line length in pixels.

Note: The desired width may not be achievable due to hardware considerations. The next larger value will be selected.

Calling Registers:

AX = 4F06h, Logical Scan Line Length function
BL = 00h, Set Scan Line Length in Pixels sub-function
CX = Desired scan line length in pixels

Return Registers:

AX = VESA VBE Return Status (see page 7-2)
BX = Bytes per scan line
CX = Actual pixels per scan line (truncated to the nearest complete pixel)
DX = Maximum number of scan lines

4F06h, 01h – Get Scan Line Length

This sub-functions gets the logical scan line length.

Calling Registers:

AX = 4F06h, Logical Scan Line Length function
BL = 01h, Get Scan Line Length sub-function

Return Registers:

AX = VESA VBE Return Status (see page 7-2)
BX = Bytes per scan line
CX = Actual pixels per scan line (truncated to the nearest complete pixel)
DX = Maximum number of scan lines

4F06h, 02h – Set Scan Line Length in Bytes

This sub-functions sets the logical scan line length in bytes.

Note: The desired width may not be achievable due to hardware considerations. The next larger value will be selected.

Calling Registers:

AX = 4F06h, Logical Scan Line Length function
BL = 02h, Set Scan Line Length in Bytes sub-function
CX = Desired scan line length in bytes

Return Registers:

AX = VESA VBE Return Status (see page 7-2)
BX = Bytes per scan line
CX = Actual pixels per scan line (truncated to the nearest complete pixel)
DX = Maximum number of scan lines

4F06h, 03h – Get Maximum Scan Line Length

This sub-functions gets the maximum logical scan line length possible.

Calling Registers:

AX = 4F06h, Logical Scan Line Length function
BL = 01h, Get Maximum Scan Line Length sub-function

Return Registers:

AX = VESA VBE Return Status (see page 7-2)
BX = Bytes per scan line
CX = Actual pixels per scan line (truncated to the nearest complete pixel)
DX = Maximum number of scan lines

7.2.4 4F07h – Display Start

This functions sets or gets the pixel to be displayed in the upper left corner of the display on the logical page. This functionality will allow a user to pan and scan around a logical image that is larger than the viewable display screen. It can also be use to rapidly switch between two different displayed images for double buffering animation effects.

4F07h, 00h – Set Display Start

This sub-functions sets the pixel to be displayed in the upper left corner of the display.

Calling Registers:

AX = 4F07h, Display Start functions
BX = 0000h, Set Display Start sub-function
CX = First displayed pixel in the scan line

DX = First displayed scan line

Return Registers:

AX = VESA VBE Return Status (see page 7-2)

4F07h, 01h – Get Display Start

This sub-functions gets the pixel displayed in the upper left corner of the display.

Calling Registers:

AX = 4F07h, Display Start functions
BX = 0001h, Get Display Start sub-function

Return Registers:

AX = VESA VBE Return Status (see page 7-2)
BH = 00h, Reserved and will be 00h
CX = First displayed pixel in scan line
DX = First displayed scan line

4F07h, 80h – Set Display Start During Vertical Retrace

This sub-functions sets the pixel to be displayed in the upper left corner of the display during a vertical retrace period to prevent line tearing.

Calling Registers:

AX = 4F07h, Display Start functions
BX = 0000h,Set Display Start During Vertical Retrace sub-function
CX = First displayed pixel in the scan line
DX = First displayed scan line

Return Registers:

AX = VESA VBE Return Status (see page 7-2)

7.2.5 4F08h – DAC Palette Format

This function manipulates the operating mode or format of the DAC palette. Some DACs are configurable to provide 6 bits, 8 bits, or more of color definition for the 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.

Note: An application can determine if DAC switching is available by querying bit 0 of the capabilities field of the VbeInfoBlock structure returned by VBE function 00h.

Note: This function will return failure code (Ah = 03h) if called in a direct color or YUV mode.

4F08h, 00h – Set DAC Palette Format

This sub-function sets the operating mode or format of the DAC palette.

Calling Registers:

AX = 4F08h, DAC Palette Format function
BL = 00h, Set DAC Palette Format
BH = Desired bits of color per primary

Return Registers:

AX = VESA VBE Return Status (see page 7-2)
BH = Current number of bits of color per primary

4F08h, 01h – Get DAC Palette Format

This sub-function gets the operating mode or format of the DAC palette.

Calling Registers:

AX = 4F08h, DAC Palette Format function
BL = 01h, Get DAC Palette Format

Return Registers:

AX = VESA VBE Return Status (see page 7-2)
BH = Current number of bits of color per primary

7.2.6 4F09h – Palette Data

This function sets or gets the color registers in the RAMDAC. It is more important for RAMDAC's which are larger than a standard VGA RAMDAC. The standard interrupt 10h video BIOS color register interface functions assume standard VGA ports and VGA palette widths. This function offers a palette interface that is independent of the VGA assumptions.

Note: The following data structure is the structure used for a palette entry in the given palette table.

```
Blue        db   ?   ; Blue channel value (6 or 8 bits)
Green       db   ?   ; Green channel value (6 or 8 bits)
Red         db   ?   ; Red channel value (6 or 8 bits)
Alignment   db   ?   ; DWord alignment byte (unused)
```

4F09h, 00h – Set Palette Data

This sub-function sets color registers in the RAMDAC.

Calling Registers:

AX = 4F09h, Palette Data function
BL = 00h or 80h, Set Palette Data sub-function
CX = Number of palette registers to update (max 255)
DX = First palette register index to update
ES:DI = Table of palette values (see above note)

Return Registers:

AX = VESA VBE Return Status (see page 7-2)

4F09h, 01h – Get Palette Data

This sub-function gets color registers in the RAMDAC.

Calling Registers:

AX = 4F09h, Palette Data function
BL = 01h, Get Palette Data sub-function
CX = Number of palette registers to update (max 255)
DX = First palette register index to update
ES:DI = Table of palette values (see above note)

Return Registers:

AX = VESA VBE Return Status (see page 7-2)

4F0Ah VBE 2.0 Protected Mode Interface

This function returns a pointer to a table that contains code for a 32-bit protected mode interface that can be either copied into local 32-bit memory space or can be executed from ROM providing the calling application sets all required selectors and I/O access correctly.

Table format:

ES:DI	Word offset in table of protected mode code for the Set Window portion of Function 05h.
ES:DI + 2	Word offset in table of protected mode code for the Set Display Start portion of Function 07h.
ES:DI + 4	Word offset in table of protected mode code for the Set Palette Data portion of Function 09h.
ES:DI + 6	Word offset in table of a list of ports and memory locations that the calling application may need I/O privileges.
ES:DI + ?	Variable length remainder of table, including code.

Calling Registers:

AX = 4F0Ah, VBE 2.0 Protected Mode Interface function
BL = 00h, Return VBE protected mode information

Return Registers:

AX = VESA VBE Return Status (see page 7-2)
ES = Real mode segment of table
DI = Offset of table
CX = Length of table including protected mode code, in bytes

7.3 VESA VBE / PM (Display Power Management) Extensions

The VBE / PM standards defines a set of function that application software can use to control the power saving features of VESA DPMS complaint displays or a flat panel controller. These functions supplement the VESA video BIOS extensions supplied in this BIOS and are accessed through interrupt 10h. They will allow an application to set display power states without specific hardware knowledge or direct access.

The hardware mechanism for controlling the power states of display devices is defined by the VESA Display Power Management Signaling (DPMS) standard.

4F10h, 00h – Report VBE/Power Management Capabilities

This sub-function returns the power management capabilities of the video controller.

Note: The REDUCED ON feature is not supported by the DPMS standard. It is intended for use by the flat panel displays.

Calling Registers:

AX = 4F10h, VBE Power Management Services function
BL = 00h, Report VBE / Power Management Capabilities sub-function
ES:DI = Null pointer, must be 0000:0000 in version 1.0, reserved for future use

Return Registers:

AX = VESA VBE Return Status (see page 7-2)

BH = Power saving state signals supported by the controller (1 = Supported, 0 = Not supported)
Bit 3 = REDUCED ON
Bit 2 = OFF
Bit 1 = SUSPEND
Bit 0 = STANDBY
BL = VBE/PM version number:
Bits 7-4 = Major version number
Bits 3-0 = Minor version number

4F10h, 01h – Set Display Power State

This sub-function sets the display power state.

Calling Registers:

AX = 4F10h, VBE Power Management Services function
BL = 01h, Set Display Power State sub-function
BH = Requested power state:
= 00h, ON
= 01h, STANDBY
= 02h, SUSPEND
= 04h, OFF
= 08h, REDUCED ON

Return Registers:

AX = VESA VBE Return Status (see page 7-2)

4F10h, 02h – Get Display Power State

This sub-function gets the display power state.

Calling Registers:

AX = 4F10h, VBE Power Management Services function
BL = 02h, Get Display Power State sub-function

Return Registers:

AX = VESA VBE Return Status (see page 7-2)
BH = Power state currently requested by the controller:
= 00h, ON
= 01h, STANDBY
= 02h, SUSPEND
= 04h, OFF
= 08h, REDUCED ON

7.4 VESA VBE / DDC (Display Data Channel) Extensions

The following functions support the VESA VBE / DDC (Display Data Channel) standard. The DDC standard defines a set of functions to retrieve the EDID (Extended Display Identification Data) structure from the display over the Display Data Channel.

The hardware mechanism and the identity information content that can be retrieved from the display devices is defined by the VESA Display Data Channel (DDC) standard.

4F15h, 00h – Report DDC Capabilities

This sub-function returns the DDC capabilities of both the video controller and the display.

Calling Registers:

AX = 4F15h, VBE / DDC Services function
BL = 00h, Report DDC Capabilities sub-function
CX = 00h, Controller unit number (00 = primary controller)
ES:DI = Null pointer, must be 0:0 in version 1.0. Reserved for future use.

Return Registers:

AX = VESA VBE Return Status (see page 7-2)
BH = Approximate time in seconds, rounded up to transfer one EDID block (128 bytes).
BL = DDC level supported by both the display and the controller:
Bit 2 = 0, Screen not blanked during data transfer.
= 1, Screen blanked during data transfer.
Bit 1 = 0, DDC 2 not supported
= 1, DDC 2 supported
Bit 0 = 0, DDC 1 not supported
= 1, DDC 1 supported

4F15h, 01h – Read EDID

This sub-function reads an EDID block from an attached display.

Calling Registers:

AX = 4F15h, VBE / DDC Services function
BL = 01h, Read EDID sub-function
CX = 00h, Controller unit number (00 = primary controller)
ES:DI = Pointer to area in which the EDID block (128 bytes) shall be returned.

Return Registers:

AX = VESA VBE Return Status (see page 7-2)

8 Hooks for the System BIOS

The video BIOS performs several system BIOS interrupt function calls (interrupt 15h / 42h hooks). Each function provides the system BIOS with the opportunity to gain control at specific times to perform any custom processing that may be required. After each interrupt hook, the system BIOS must return control to the video BIOS. These hooks can be BMPed to disabled, use interrupt 15h, or use interrupt 42h. Interrupt 42h is given as an option because of the growing unreliability on interrupt 15h to return register data. These functions are implemented at the discretion of the system BIOS designer.

Table 8.1 INT 15h / INT 42h Hooks for the System BIOS

Hook	Hook Name	Hook Description (Short)
5F31h	POST Completion Notification Hook	Signals the completion of video POST.
5F33h	Hook After Mode Set	Allows system BIOS control at the end of a mode set.
5F35h	Video Display Hook	Allows system BIOS to perform monitor sensing and to override the video display setting in the BMP.
5F36h	Set NTSC / PAL Hook	Notifies system BIOS of NTSC or PAL state that is about to be set.
5F38h	Hook Before Mode Set	Allows system BIOS control before setting the mode.
5F40h	Set Panel Type Hook	Allows system BIOS to select a flat panel type upon power up.
5F45h	Hook VESA VBE / DDC Functions	Allows system BIOS to initialize data before or take over the VESA VBE / DDC functions.
5F46h	Hook VESA VBE / PM Functions	Allows system BIOS to initialize data before, or take over the Video BIOS VESA VBE / PM functions.
5F47h	Notify Display Switch Hook	Notifies system BIOS that a possible display switch has taken place.
5F48h, 00h	Read I^2C Data Line Sub-hook	Calls system BIOS to read the current level of the I^2C data line.
5F48h, 01h	Write I^2C Data Line Sub-hook	Calls system BIOS to write a given level to the I^2C data line.
5F48h, 02h	Read I^2C Clock Line Sub-hook	Calls system BIOS to read the current level of the I^2C clock line.
5F48h, 03h	Write I^2C Clock Line Sub-hook	Calls system BIOS to write a given level to the I^2C clock line.
5F48h, 04h	Initiate Before I^2C Functions Sub-hook	Calls system BIOS to initialize the system state before the I^2C functions are executed.
5F48h, 05h	Reset After I^2C Functions Sub-hook	Calls system BIOS to reset the system state after the I^2C functions have executed.
5F63h	Notify Power State Switch Hook	Notifies the system BIOS of the requested power state switch.

5F31h – POST Completion Notification Hook

This hook signals the completion of video POST (Power On Self-Test). The hook executes after the sign-on message is displayed and PCI BIOS resizing.

Calling Registers:

AX = 5F31h, POST Completion Notification Hook

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F33h – Hook After Mode Set

This hook allows the system BIOS to intercept the video BIOS at the end of a mode set.

Calling Registers:

AX = 5F33h, Hook After Mode Set
BH = Number of character columns
BL = Current mode number
CH = Active display page

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F35h – Video Display Hook

This hook allows the system BIOS to perform display device sensing and to override the video display default setting. The video BIOS will set the returned video display upon exiting video POST.

Calling Registers:

AX = 5F35h, Video Display Hook

Return Registers:

AX = Return Status (function not supported if AL != 5Fh);
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
CX = Display Device Combination to Boot (1 = Enable display, 0 = Disable display):
Bits 15-11 = Pipe B – Reserved
Bit 10 = Pipe B – TV Analogue (If set bits 0 and 8 can not be set)
Bit 9 = Pipe B – Reserved
Bit 8 = Pipe B – CRT (If set bits 0 and 10 can not be set)
Bits 7-4 = Pipe A – Reserved
Bit 3 = Pipe A – TV Digital (If set bit 1 can not be set)
Bit 2 = Pipe A – Reserved
Bit 1 = Pipe A – Flat Panel (If set bit 3 can not be set)
Bit 0 = Pipe A – CRT (If set bits 8 and 10 can not be set)

5F36h – Set NTSC / PAL Hook

This hook notifies the system BIOS of the NTSC or PAL state that is about to be set.

Calling Registers:

AX = 5F36h, Set NTSC / PAL Hook
BH = NTSC / PAL state being set:
= 00h, Disable NTSC and PAL
= 01h, Enable NTSC

= 02h, Enable PAL

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F38h – Hook Before Mode Set

This hook allows the system BIOS to intercept the video BIOS before setting the mode.

Calling Registers:

AX = 5F38h, Hook Before Mode Set
CL = New video mode to be set

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

5F40h – Set Panel Type Hook

This hook allows the system BIOS to select one of the flat panel types upon power up (see supported panel class table). 32K BIOS's only support the first 8 flat panel types.

Calling Registers:

AX = 5F40h, Set Panel Type Hook

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
CL = 0 – 15, Panel type minus one

5F45h – Hook for VESA VBE / DDC Functions

This hook allows the system BIOS to initialize data before or take over the VESA VBE / DDC (Display Data Channel) functions. The video BIOS DDC functions are executed if a 5Fh is not returned in AL or if AH is greater than 01h. The system BIOS must return the correct DDC function status in AH when 5Fh is returned in AL. This status will be converted into VESA VBE / DDC function status.

Calling Registers:

AX = 5F45h, Hook VESA VBE / DDC Functions

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
= 025Fh-FF5Fh, Function was successful, but run video BIOS DDC functions

5F46h – Hook for VESA VBE / PM Functions

This hook allows the system BIOS to initialize data before, or take over the Video BIOS VESA VBE / PM (Monitor Power Management) functions. The video BIOS PM functions are executed if a 5Fh is not returned in AL or if AH is greater than 01h. The system BIOS must return the correct PM function status in AH when 5Fh is returned in AL. This status will be converted into VESA VBE / PM function status.

Calling Registers:

AX = 5F46h, Hook for VESA VBE / PM Functions
BH = Power Saving State (BL = 01h only)
= 00h, On
= 01h, Standby
= 02h, Suspend
= 04h, Off
= 08h, Reduced On
BL = VESA PM Sub Function Number
= 00h, Report VBE / PM services
= 01h, Set display power state
= 02h, Get display power state

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful
= 025Fh-FF5Fh, Function was successful, but run video BIOS DDC functions

5F47h – Notify Display Switch Hook

This hook will notify the system BIOS that a possible display switch has taken place. It will offer the current new display device combination as an input. The purpose of this hook is to inform the system BIOS to activate or deactivate hardware used by certain displays (i.e. backlight or digitizer for flat panels).

Calling Registers:

AX = 5F47h, Notify Display Switch Hook
CX = New Display Device Combination:

Bits 15-11	= Pipe B – Reserved
Bit 10	= Pipe B – TV Analogue
Bit 9	= Pipe B – Reserved
Bit 8	= Pipe B – CRT
Bits 7-4	= Pipe A – Reserved
Bit 3	= Pipe A – TV Digital
Bit 2	= Pipe A – Reserved
Bit 1	= Pipe A – Flat Panel
Bit 0	= Pipe A – CRT

Return Registers:

AX = Return Status (function not supported if AL != 5Fh):
= 005Fh, Function supported but failed
= 015Fh, Function supported and successful

8.1 5F48h – I^{2}C Read and Write Functions Hook

This hook allows a system with a unique hardware I^{2}C design to use the bulk of the video BIOS's I^{2}C code. It does this by calling the system BIOS with sub-functions (sub-hooks) that will read and write the I^{2}C data lines, read and write the I^{2}C clock lines, and initialize and restore the I^{2}C state.

Note: The BMP utility must be used to active this hook. If this hook is implemented totally, it is not important which GPIO pin is selected for I^{2}C, DDC1 or DDC2. The initialize and reset sub-hooks should return AL = 5Fh (System BIOS Support) even if no initialization is needed in order to stop unwanted initialization due to the GPIO pin selected.

5F48h, 00h – Read I^{2}C Data Line Sub-hook

This sub-hook calls the system BIOS to read and return the current level of the I^{2}C data line. The video BIOS read I^{2}C data line function is executed if a 5Fh is not returned in AL.

Calling Registers:

AX = 5F48h, I^{2}C Read and Write Functions Hook
BH = 00h, Read I^{2}C Data Line Sub-hook

Return Registers:

AL = 5Fh, System BIOS Support
BL = I^{2}C Data Line Level:
= 00h, Data line is low
= 01h, Data line is high

5F48h, 01h – Write I^{2}C Data Line Sub-hook

This sub-hook calls the system BIOS to write a given level to the I^{2}C data line. The video BIOS write I^{2}C data line function is executed if a 5Fh is not returned in AL.

Calling Registers:

AX = 5F48h, I^{2}C Read and Write Functions Hook
BH = 01h, Write I^{2}C Data Line Sub-hook
BL = I^{2}C Data Line Level:
= 00h, Set data line low
= 01h, Set data line high

Return Registers:

AL = 5Fh, System BIOS Support

5F48h, 02h – Read I^{2}C Clock Line Sub-hook

This sub-hook calls the system BIOS to read and return the current level of the I^{2}C clock line. The video BIOS read I^{2}C clock line function is executed if a 5Fh is not returned in AL.

Note: This sub-hook must not change the clock line direction from an output to an input. This action may cause clock line noise and bad I^{2}C data.

Calling Registers:

AX = 5F48h, I^{2}C Read and Write Functions Hook
BH = 02h, Read I^{2}C Clock Line Sub-hook

Return Registers:

AL = 5Fh, System BIOS Support
BL = I^{2}C Clock Line Level:
= 00h, Clock line is low
= 01h, Clock line is high

5F48h, 03h – Write I^{2}C Clock Line Sub-hook

This sub-hook calls the system BIOS to write a given level to the I^{2}C clock line. The video BIOS write I^{2}C clock line function is executed if a 5Fh is not returned in AL.

Calling Registers:

AX = 5F48h, I^{2}C Read and Write Functions Hook
BH = 03h, Write I^{2}C Clock Line Sub-hook
BL = I^{2}C Clock Line Level:
= 00h, Set clock line low
= 01h, Set clock line high

Return Registers:

AL = 5Fh, System BIOS Support

5F48h, 04h – Initialize Before I^{2}C Functions Sub-hook

This sub-hook calls the system BIOS to initialize the system state if necessary before the I^{2}C functions are executed. The video BIOS initialize state before I^{2}C function is executed if a 5Fh is not returned in AL.

Note: This sub-hook is also call in video POST.

Calling Registers:

AX = 5F48h, I^{2}C Read and Write Functions Hook
BH = 04h, Initialize Before I^{2}C Functions Sub-hook

Return Registers:

AL = 5Fh, System BIOS Support

5F48h, 05h – Reset After I^{2}C Functions Sub-hook

This sub-hook calls the system BIOS to reset the system state if necessary after the I^{2}C functions have executed. The video BIOS reset state after I^{2}C function is executed if a 5Fh is not returned in AL.

Calling Registers:

AX = 5F48h, I^{2}C Read and Write Functions Hook
BH = 05h, Reset After I^{2}C Functions Sub-hook

Return Registers:

AL = 5Fh, System BIOS Support

5F63h – Notify Adapter Power State Switch Hook

This hook notifies the system BIOS of the requested power state switch. A table to convert the ACPI power state names to the APM power state names is as follows.

D0 = On
D1 = Standby
D2 = Suspend
D3 = Off

Calling Registers:

```
AX  = 5F63h, Notify Adapter Power State Switch Hook
BL  = Power state switch requested:
    = 01h, D0 to D1
    = 02h, D0 to D2
    = 03h, D0 to D3
    = 04h, D1 to D0
    = 05h, D2 to D0
    = 06h, D3 to D0
```

Return Registers:

```
AX  = Return Status (function not supported if AL != 5Fh):
    = 005Fh, Function supported but failed
    = 015Fh, Function supported and successful
```

This page intentionally left blank.

9 OEM Utility Programs

The OEM utility programs allow the OEM to prepare the BIOS for use. The BMP utility program enables the OEM to prepare a custom version of the BIOS.

Note: The OEMs may not reproduce nor distribute these programs.

9.1 BIOS Modification Program

The BIOS Modification Program (BMP) enables OEMs to customize the video BIOS for their own specific requirements. BMP allows the OEM to modify certain parameters of a binary version of the BIOS to be modified. The parameters that the BMP can modify include:

- Sign-on message
- General and flat panel BIOS Features
- Display type determination
- Set FP Dot Clock
- Set FP Memory Clock
- Extended display modes
- Register tables

BMP provide the capability to save the state of the BMPed data to a text file (.BMS Files). This file can be edited and used to initialize BMP data on other BIOS's that use the save BMP data variables.

9.1.1 Usage

BMPxxx [File]

Where:

xxx = a number assigned by the video chipset (normally the last 3 digits of the chipset number)

[File] = Optional filename of the BIOS file input to the BMP. A default extension of .DAT is assumed if no extension is specified. A default filename of VGAxxx.DAT is assumed if no filename is specified.

Examples:

ROM Binary:

BMPxxx [VGA030.DAT]

Executes BMPxxx with the default file VGAxxx.DAT as the input file.

RAM Executable:

BMPxxx RAMxxx.EXE

Executes BMPxxx with the RAMxxx.EXE or utility program as the input file.

Utils

9.1.2 Commands

BMP organizes the modifiable parameters of the BIOS into several windows. Some values are entered as text or as hexadecimal values within these windows. The following keys are used to change fields or edit values:

Table 9.1 BMP Commands.

Command	Function
<Tab>	Go to next window.
<Shift Tab>	Go to previous window.
<PgUp>	Move up one page within a window.
<PgDn>	Move down one page within a window.
<↑>,<↓>	Move up or down one line or field.
<←>,<→>	Move left or right one character or field.
<+>,<->	Enable/disable parameter. Increment/decrement a value in the field.
<F1>	Help.
<F5>	Save BMS file.
<F6>	Load BMS file.
<F10>	Save changes to the BIOS file.
<Esc>	Exit program.

9.1.3 Error Messages

If BMP encounters an error during operation, a red window will appear which will contain the error condition. Table 9-1 lists these errors, the possible cause, and recommended solution.

Table 9.1 BMP Error Messages

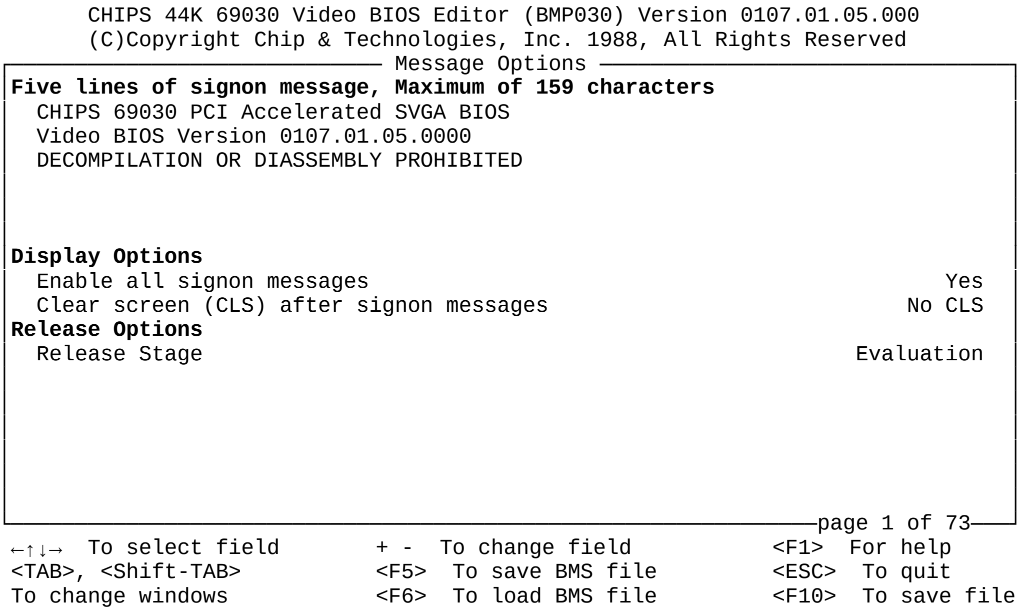
Error Message	Problem Description, Recommended Action
Use original BIOS file	The file has already been modified and saved. Use the binary file that was supplied on the original disk. If this does not work, contact local CHIPS sales representative.
Editable Structure not found	The file can not be modified. This is the wrong file. The binary file that was supplied on the original disk should be used.
This program is unable to edit the BMP structure in that file	There is an incompatible version of BMP and binary file. Use the binary file and BMP program supplied on the original disk.
Bad BMP structure, Old version was Num, header version was Num	This is an incompatible version of BMP or binary file. Use the binary file and BMP program supplied on original disk.
Unable to allocate memory	There is not enough system memory. Remove all unnecessary resident programs and reboot the system. BMP requires approximately 300K of memory.
Binary file File not found	BMP could not find the specified file. Verify that the specified file exists.
Unable to read binary file File	BMP could not read the specified file. Specified file may be corrupted, use backup copy.
Unable to write to File	There was an error during write to specified file. The file may be marked read-only. Try making changes to a file that has read and write access.
Unable to reopen File for saving	Unable to re-open binary file. The file may be a read-only file. Try making changes to a file with read and write access.
Unable to open my own .EXE file File	Unable to open BMP030.exe for reading. This may be due to insufficient memory, or because the BMP030.exe filename has been changed. Use the BMP and binary files from the original disk.
Unable to open BMS file BMSfile	Unable to find or read BMS file. Try specifying a file that does exist.
Unable to create file BMSfile	Unable to write a BMS file. There may be insufficient disk space, or an existing file has read-only access.

Note:

File	Binary filename used.
Num	Version number of BMP structure in BMP and/or binary file.
BMSfile	BMS filename used.

9.1.4 BMP Screens

The figure below shows a sample BMP frame .



```
      CHIPS 44K 69030 Video BIOS Editor (BMP030) Version 0107.01.05.000
      (C)Copyright Chip & Technologies, Inc. 1988, All Rights Reserved
┌───────────────────────────── Message Options ──────────────────────────────┐
│Five lines of signon message, Maximum of 159 characters                     │
│  CHIPS 69030 PCI Accelerated SVGA BIOS                                     │
│  Video BIOS Version 0107.01.05.0000                                        │
│  DECOMPILATION OR DIASSEMBLY PROHIBITED                                    │
│                                                                            │
│                                                                            │
│                                                                            │
│Display Options                                                             │
│  Enable all signon messages                                             Yes│
│  Clear screen (CLS) after signon messages                            No CLS│
│Release Options                                                             │
│  Release Stage                                                   Evaluation│
│                                                                            │
│                                                                            │
│                                                                            │
│                                                                            │
│                                                                            │
└─────────────────────────────────────────────────────────────page 1 of 73───┘
←↑↓→  To select field        + -  To change field          <F1>  For help
<TAB>, <Shift-TAB>           <F5>  To save BMS file        <ESC>  To quit
To change windows            <F6>  To load BMS file        <F10>  To save file
```

Figure 9-2: Sample BMP Frame

The BMP screens are a set of windows that allows the OEM to adjust the BIOS data and activate selected features. These screens are presented in this document with the following series of tables. Each table corresponds to one screen of the BMP editor. Each table lists the title of each field, provides a description of the field, and provides the possible values for each field. The default values appear in **bold** text.

Table 9.1 BMP Page 1, Message Options

Description or Help	Value(s)
Five lines of sign-on message, maximum of 159 characters	
This is the sign-on message that will be displayed when the machine is booted. You may enter a maximum of five lines of text, with no more than 70 characters on each line, and no more than 159 total characters.	**CHIPS 69030 VGA PCI Accelerated SVGA BIOS Version 0106.01.14.000** **DECOMILATION OR DISASSEMBLY PROHIBITED**
Display Options	
Enable all signon messages Setting this field to yes will enable the eval message, the signon message, the copyright message and the message wait.	**Yes**, No

Description or Help	Value(s)
Clear screen (CLS) after singon message Setting this field to anything other than 'No CLS' will cause the video BIOS to pause for the specified number of seconds while displaying the eval, singon and / or copyright message(s). The default is 'NO CLS' which causes the BIOS to display the message(s), does not pause and does not clear the screen.	**No CLS**, 0.5 Second Delay + CLS, 1.0 Second Delay + CLS, 1.5 Second Delay + CLS, 2.0 Second Delay + CLS, 2.5 Second Delay + CLS, 3.0 Second Delay + CLS, 3.5 Second Delay + CLS
Release Options	
Release Stage Set the BIOS release stage. The BIOS POST evaluation message is not displayed when the 'Production' option is selected.	**Evaluation**, Production

Table 9.2 BMP Page 2, BIOS Features

Description or Help	Value(s)
General Features	
DDC2 Clock Line GPIO pin: DDC1 implementations should set this entry to disabled. If a DDC line will use GPIO 1 or GPIO 0, FR0C should set that GPIO pin as an input. (See the FP & SM Boot Parameters page.)	GPIO 0, GPIO 1, GPIO 2, GPIO 3, **BIOS DDC2 Disabled**
DDC1 or DDC2 Data Line GPIO pin: If this entry is set to BIOS DDC1 and DDC2 Disabled, both DDC1 and DDC2 will be disabled. However, DDC dispatching and the system BIOS interrupt hook will continue to take place. If a DDC line will use GPIO 1 or GPIO 0, FR0C should set that GPIO pin as an input.(See the FP & SM Boot Parameters page.)	GPIO 0, GPIO 1, GPIO 2 , GPIO 3 , **BIOS DDC1 and DDC2 Disabled**
FP display power state after POST: This entry will inform the BIOS to use the low power memory clocks on the flat panel type screens instead of the normal high performance memory clocks.	**High Performance**, Low Power
Use panel off in DPMS functions on SM display. Using panel off in simultaneous display will force the backlight off in standby and suspend states. However, this will make the VESA PM function not conform to spec.	Yes, **No**
Popup Support The popup is a hardware icon.	Enabled, **Disabled**
Popup Position is adjustable?	**Yes**, No
Popups (Total Number Of Popups)	**1**, 2, 3, 4, 5, 6, 7, 8
Popup Memory Size	1 KB, **2 KB**

Resize PCI BIOS This will allow the runtime BIOS to be as small as possible or to make shadowed memory granularity.	Disabled, **Resize to 0.5K boundary,** Resize to 1K boundary, Resize to 2K boundary, Resize to 4K boundary, Resize to 8K boundary, Resize to 16K boundary
ROM Segment The BIOS will run at this code segment. The segment of an AT adapter board BIOS is usually C000h. The segment of a motherboard BIOS is not specifically defined, and varies from board to board. C000h and E000h are typical.	**C000**, C800, D000, D800, E000, E800, F000, F800
CRT Display Memory Clock Frequencies:	
Standard VGA Modes	**100 MHz**, 4 - 135 MHz
640x480 4/8 Bpp modes	**100 MHz**, 4 - 135 MHz
640x480 15/16 Bpp modes	**100 MHz**, 4 - 135 MHz
640x480 24 Bpp modes	**100 MHz**, 4 - 135 MHz
800x600 4/8 Bpp modes	**100 MHz**, 4 - 135 MHz
800x600 15/16 Bpp modes	**100 MHz**, 4 - 135 MHz
800x600 24 Bpp modes	**100 MHz**, 4 - 135 MHz
1024x768 4/8 Bpp modes	**100 MHz**, 4 - 135 MHz
1024x768 15/16 Bpp modes	**100 MHz**, 4 - 135 MHz
1024x768 24 Bpp modes	**100 MHz**, 4 - 135 MHz
1280x1024 4/8 Bpp modes	**100 MHz**, 4 - 135 MHz
1280x1024 15/16 Bpp modes	**100 MHz**, 4 - 135 MHz
1280x1024 24 Bpp modes	**100 MHz**, 4 - 135 MHz
Dot and Memory Clock Overrides: Note: For Clock Type – 0 = Dot, 1 = Memory, 2 = Dot and Memory The following will allow any dot or memory clock to be overridden with new clock values. This is useful for EMI and other similar problems.	
Override Clock (MHz) , Clock Type	**1** (1 - 255 MHz), **2** (0 - 2)
M (XRC8 + 2), N (XRC9 + 2) and Divisor (XRCB)	**00** (00h - FFh), **00** (00h - FFh), **00** (00h - FFh)
Override Clock (MHz) , Clock Type	**2** (1 - 255 MHz), **2** (0 - 2)
M (XRC8 + 2), N (XRC9 + 2) and Divisor (XRCB)	**00** (00h - FFh), **00** (00h - FFh), **00** (00h - FFh)
Override Clock (MHz) , Clock Type	**3** (1 - 255 MHz), **2** (0 - 2)
M (XRC8 + 2), N (XRC9 + 2) and Divisor (XRCB)	**00** (00h - FFh), **00** (00h - FFh), **00** (00h - FFh)
Override Clock (MHz) , Clock Type	**4** (1 - 255 MHz), **2** (0 - 2)
M (XRC8 + 2), N (XRC9 + 2) and Divisor (XRCB)	**00** (00h - FFh), **00** (00h - FFh), **00** (00h - FFh)
Override Clock (MHz) , Clock Type	**5** (1 - 255 MHz), **2** (0 - 2)
M (XRC8 + 2), N (XRC9 + 2) and Divisor (XRCB)	**00** (00h - FFh), **00** (00h - FFh), **00** (00h - FFh)
Override Clock (MHz) , Clock Type	**6** (1 - 255 MHz), **2** (0 - 2)
M (XRC8 + 2), N (XRC9 + 2) and Divisor (XRCB)	**00** (00h - FFh), **00** (00h - FFh), **00** (00h - FFh)

Table 9.3 BMP Page 3, BIOS Features

Description or Help	Value(s)
Display Determination	
Boot Up Display Type This selects a boot up display type. The selections 'Boot on FP or CRT if CRT found' and 'Boot on FP or SM if CRT found' will boot to the second display if the BIOS finds a monitor attached. The other boot displays will not check for a monitor.	Pipe A: CRT, Pipe B: Off Pipe A: Flat Panel (FP), Pipe B: Off Pipe A: TV Digital (TVD), Pipe B: Off Pipe A: CRT / FP, Pipe B: Off **Pipe A: CRT / TVD, Pipe B: Off** Pipe A: Flat Panel (FP), Pipe B: CRT Pipe A: TV Digital (TVD), Pipe B: CRT Pipe A: Flat Panel (FP), Pipe B: TVA Pipe A: FP (CRT if CRT), Pipe B: Off Pipe A: FP (TVD if TVD), Pipe B: Off Pipe A: (FP if no CRT), Pipe B: (CRT if CRT)
Select Panel If 'Read Hardware Pins' is selected the BIOS will determine the panel type via hardware pins. If a 'Panel #?' is selected the BIOS will boot to that panel. Both of these mechanism can be overridden by a system BIOS hook. The default panel types are as follows. PANEL #01: 1024 x 768 Dual Scan STN Color PANEL #02: 1280 x 1024 TFT Color PANEL #03: 640 x 480 Dual Scan STN Color PANEL #04: 800 x 600 Dual Scan STN Color PANEL #05: 640 x 480 Sharp TFT Color PANEL #06: 640 x 480 18-Bit TFT Color PANEL #07: 1024 x 768 TFT Color PANEL #08: 800 x 600 TFT Color PANEL #09: User Defined PANEL #10: User Defined PANEL #11: User Defined PANEL #12: User Defined PANEL #13: User Defined PANEL #14: 1280 x 1024 Dual Scan STN Color PANEL #15: 1024 x 600 Dual Scan STN Color PANEL #16: 1024 x 600 TFT Color	Read HW Pins, PANEL #01, PANEL #02, PANEL #03, PANEL #04, PANEL #05, **PANEL #06**, PANEL #07, PANEL #08, PANEL #09, PANEL #10, PANEL #11, PANEL #12, PANEL #13, PANEL #14, PANEL #15, PANEL #16
System BIOS Hooks The video BIOS will call interrupt 15h, interrupt 42h, or no interrupt as requested. Interrupt 42h is made available because interrupt 15h is becoming less reliable.	
5F31h – POST Completion Hook	**Disabled**, Use INT 15h, Use INT 42h
5F33h – Hook After Mode Set	**Disabled**, Use INT 15h, Use INT 42h
5F35h – Boot Up Display Type Hook (CRT, FP, SM)	**Disabled**, Use INT 15h, Use INT 42h
5F36h – Set NTSC and PAL Hook	**Disabled**, Use INT 15h, Use INT 42h
5F38h – Hook Before Mode Set	**Disabled**, Use INT 15h, Use INT 42h
5F40h – Set Panel Type Hook	**Disabled**, Use INT 15h, Use INT 42h
5F45h – Hook Before VESAVBE / DDC	**Disabled**, Use INT 15h, Use INT 42h
5F46h – Hook Before VESAVBE / PM	**Disabled**, Use INT 15h, Use INT 42h
5F47h – Notify Display Switch Hook (CRT, FP, SM)	**Disabled**, Use INT 15h, Use INT 42h
5F48h – VESAVBE / DDC Read and Write Functions Hook	**Disabled**, Use INT 15h, Use INT 42h
5F49h – Hook After VESA VBE / PM	**Disabled**, Use INT 15h, Use INT 42h
5F4Ah – Notify Interlace State Hook	**Disabled**, Use INT 15h, Use INT 42h
5F4Bh – Switch Video Chipset Clock for PAL Hook	**Disabled**, Use INT 15h, Use INT 42h
5F63h – Set Adapter Power State Hook	**Disabled**, Use INT 15h, Use INT 42h

Table 9.4 BMP Page 4, Enable / Disable Modes

Description or Help	Value(s)
Enable / Disable Modes for CRT Display Refresh Rates Modes: 58, 56, 54, 52, 50, 48/49, 47/46, 45/44, 43/42, 41/40, 3A, 38, 36, 34, 32, 30 Note: 3x modes = 7x, 6x, 3x (note is also for panels)	
43 Hz (Interlaced) Refresh Rates	**1010 0101 0001 0100**
56 Hz Refresh Rates	**0000 0000 0000 0000**
60 Hz Refresh Rates	**1011 1101 1111 0111**
75 Hz Refresh Rates	**1011 1101 1101 0111**
85 Hz Refresh Rates	**0011 1001 1100 0111**
100 Hz Refresh Rates	**0011 1001 1100 0111**
Enable / Disable Modes for Panel #1 Displays Modes: 58, 56, 54, 52, 50, 48/49, 47/46, 45/44, 43/42, 41/40, 3A, 38, 36, 34, 32, 30	
Pipe A: CRT + FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Pipe A: FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Enable / Disable Modes for Panel #2 Displays Modes: 58, 56, 54, 52, 50, 48/49, 47/46, 45/44, 43/42, 41/40, 3A, 38, 36, 34, 32, 30	
Pipe A: CRT + FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Pipe A: FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Enable / Disable Modes for Panel #3 Displays Modes: 58, 56, 54, 52, 50, 48/49, 47/46, 45/44, 43/42, 41/40, 3A, 38, 36, 34, 32, 30	
Pipe A: CRT + FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Pipe A: FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Enable / Disable Modes for Panel #4 Displays Modes: 58, 56, 54, 52, 50, 48/49, 47/46, 45/44, 43/42, 41/40, 3A, 38, 36, 34, 32, 30	
Pipe A: CRT + FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Pipe A: FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Enable / Disable Modes for Panel #5 Displays Modes: 58, 56, 54, 52, 50, 48/49, 47/46, 45/44, 43/42, 41/40, 3A, 38, 36, 34, 32, 30	
Pipe A: CRT + FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Pipe A: FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Enable / Disable Modes for Panel #6 Displays Modes: 58, 56, 54, 52, 50, 48/49, 47/46, 45/44, 43/42, 41/40, 3A, 38, 36, 34, 32, 30	
Pipe A: CRT + FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Pipe A: FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Enable / Disable Modes for Panel #7 Displays Modes: 58, 56, 54, 52, 50, 48/49, 47/46, 45/44, 43/42, 41/40, 3A, 38, 36, 34, 32, 30	
Pipe A: CRT + FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Pipe A: FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Enable / Disable Modes for Panel #8 Displays Modes: 58, 56, 54, 52, 50, 48/49, 47/46, 45/44, 43/42, 41/40, 3A, 38, 36, 34, 32, 30	
Pipe A: CRT + FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Pipe A: FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Enable / Disable Modes for Panel #9 Displays Modes: 58, 56, 54, 52, 50, 48/49, 47/46, 45/44, 43/42, 41/40, 3A, 38, 36, 34, 32, 30	
Pipe A: CRT + FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Pipe A: FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Enable / Disable Modes for Panel #10 Displays Modes: 58, 56, 54, 52, 50, 48/49, 47/46, 45/44, 43/42, 41/40, 3A, 38, 36, 34, 32, 30	
Pipe A: CRT + FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Pipe A: FP, Pipe B: Off Displays	**1011 1101 1111 0111**

Description or Help	Value(s)
Enable / Disable Modes for Panel #11 Displays	
Modes: 58, 56, 54, 52, 50, 48/49, 47/46, 45/44, 43/42, 41/40, 3A, 38, 36, 34, 32, 30	
Pipe A: CRT + FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Pipe A: FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Enable / Disable Modes for Panel #12 Displays	
Modes: 58, 56, 54, 52, 50, 48/49, 47/46, 45/44, 43/42, 41/40, 3A, 38, 36, 34, 32, 30	
Pipe A: CRT + FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Pipe A: FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Enable / Disable Modes for Panel #13 Displays	
Modes: 58, 56, 54, 52, 50, 48/49, 47/46, 45/44, 43/42, 41/40, 3A, 38, 36, 34, 32, 30	
Pipe A: CRT + FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Pipe A: FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Enable / Disable Modes for Panel #14 Displays	
Modes: 58, 56, 54, 52, 50, 48/49, 47/46, 45/44, 43/42, 41/40, 3A, 38, 36, 34, 32, 30	
Pipe A: CRT + FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Pipe A: FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Enable / Disable Modes for Panel #15 Displays	
Modes: 58, 56, 54, 52, 50, 48/49, 47/46, 45/44, 43/42, 41/40, 3A, 38, 36, 34, 32, 30	
Pipe A: CRT + FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Pipe A: FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Enable / Disable Modes for Panel #16 Displays	
Modes: 58, 56, 54, 52, 50, 48/49, 47/46, 45/44, 43/42, 41/40, 3A, 38, 36, 34, 32, 30	
Pipe A: CRT + FP, Pipe B: Off Displays	**1011 1101 1111 0111**
Pipe A: FP, Pipe B: Off Displays	**1011 1101 1111 0111**

Table 9.5 BMP Page 5, Generic Extended Mode 36h Parameters

Description or Help	Value(s)
General Parameters:	
Number of Text Columns	**80**
Number of Text Rows - 1	**24**
Font Height	**10h**
Registers:	
Miscellaneous Output Register	**2Bh**
Sequencer Registers SR01 – SR04	**01h, 0Fh, 00h, 0Eh**
Offset Register (CR13)	**80h**
CRT Registers for 43 Hz (Interlaced):	
Dot Clock	**45 MHz**
CR00 – CR07	**99h, 7Fh, 80h, 9Ch, 83h, 19h, 96h, 5Dh**
CR09, CR10 – CR12, CR15 – CR16	**40h, 54h, 86h, 57h, 2Ch, 96h**
CRT Registers for 60 Hz:	
Dot Clock	**65 MHz**
CR00 – CR07	**A3h, 7Fh, 80h, 86h, 85h, 96h, 24h, F1h**
CR09, CR10 – CR12, CR15 – CR16	**60h, BAh, 80h, 57h, 58h, 24h**
CRT Registers for 75 Hz:	
Dot Clock	**79 MHz**
CR00 – CR07	**9Fh, 7Fh, 80h, 82h, 84h, 90h, 1Eh, F1h**
CR09, CR10 – CR12, CR15 – CR16	**60h, B5h, 88h, 57h, 58h, 1Eh**
CRT Registers for 85 Hz:	
Dot Clock	**94 MHz**
CR00 – CR07	**A7h, 7Fh, 80h, 8Ah, 88h, 94h, 26h, F1h**

Description or Help	Value(s)
CR09, CR10 – CR12, CR15 – CR16	60h, B5h, 88h, 57h, 58h, 26h
Extended CR Registers:	
Address, Data, Mask	30h, 00h, 0Ch
Address, Data, Mask	31h, 00h, 0Ch
Address, Data, Mask	32h, 00h, 0Ch
Address, Data, Mask	33h, 00h, 0Ch
Address, Data, Mask	70h, 4Fh, 7Fh
Extended Registers:	
Address, Data, Mask	00h, 00h, 00h
Address, Data, Mask	00h, 00h, 00h
Address, Data, Mask	00h, 00h, 00h
Reset Extended Registers:	
Address, Data, Mask	00h, 00h, 00h
Address, Data, Mask	00h, 00h, 00h
Address, Data, Mask	00h, 00h, 00h

Table 9.6 BMP Page 6, NTSC / PAL Mode Support

Description or Help	Value(s)
Options:	
Applications show NTSC / PAL support? This field will only inform application (i.e. Windows control panel) to show or hide any NTSC and PAL support. NTSC is enabled or disabled elsewhere.	Enabled, **Disabled**
Output composite sync in NTSC / PAL State? If enabled composite sync will be outputted instead of normal Hsync and Vsync while in NTSC or PAL state.	**Enabled**, Disabled
Enable / Disable TV though GPIO0? If enabled, TV / CRT output switching is controlled though GPIO0.	Enabled, **Disabled**
Mode 03+ NTSC:	
Clock M, N	**2Bh, 14h**
CR00 – CR07	**68h, 4Fh, 4Fh, 8Ch, 58h, 9Eh, 06h, 01h**
CR09, CR10 – CR12, CR15 – CR16	**47h, F5h, 0Eh, C7h, Dah, 07h**
Mode 03+ PAL:	
Clock M, N	**18h, 0Bh**
CR00 – CR07	**6Ah, 4Fh, 4Fh, 8Eh, 57h, 9Eh, 36h, 01h**
CR09, CR10 – CR12, CR15 – CR16	**47h, F5h, 0Eh, C7h, Efh, 37h**
Mode 13 NTSC:	
Clock M, N	**15h, 0Bh**
CR00 – CR07	**68h, 4Fh, 55h, 8Ch, 58h, 86h, 04h, 13h**
CR09, CR10 – CR12, CR15 – CR16	**DFh, Feh, 03h, 8Fh, DDh, 05h**
Mode 13 PAL:	
Clock M, N	**40h, 21h**
CR00 – CR07	**6Ah, 4Fh, 4Fh, 8Eh, 57h, 9Fh, 36h, 17h**
CR09, CR10 – CR12, CR15 – CR16	**DFh, 20h, 04h, 8Fh, F7h, 37h**
Mode 640 x 480 NTSC:	
Clock M, N	**15h, 0Bh**

Description or Help	Value(s)
Options:	
CR00 – CR07	**67h, 4Fh, 55h, 8Bh, 58h, 86h, 04h, 13h**
CR09, CR10 – CR12, CR15 – CR16	**40h, Feh, 03h, 8Fh, DDh, 05h**
Mode 640 x 480 PAL:	
Clock M, N	**29h, 15h**
CR00 – CR07	**6Bh, 4Fh, 54h, 8Fh, 57h, 87h, 3Eh, 1Fh**
CR09, CR10 – CR12, CR15 – CR16	**40h, 20h, 04h, DFh, 0Ch, 3Fh**
Mode 800 x 600 PAL:	
Clock M, N	**34h, 15h**
CR00 – CR07	**89h, 63h, 6Ch, 8Dh, 6Fh, 15h, 3Eh, 1Fh**
CR09, CR10 – CR12, CR15 – CR16	**40h, 2Fh, 0Fh, DFh, 0Fh, 3Fh**
Composite Sync Table:	
Composite Sync Table Indexes	**70h, 71h, 72h, 73h, 74h, 75h, 76h, 77h, 79h**
NTSC 8 bpp Composite Sync Values	**23h, 00h, 1Ah, 51h, 06h, 07h, 0Ah, 0Fh, 04h**
NTSC 15 / 16 bpp Composite Sync Values	**23h, 00h, 1Ah, 51h, 06h, 07h, 14h, 0Fh, 04h**
NTSC 24 bpp Composite Sync Values	**23h, 00h, 1Ah, 51h, 06h, 07h, 1Eh, 0Fh, 04h**
PAL 8 bpp Composite Sync Values	**26h, 80h, 1Fh, 50h, 07h, 04h, 13h, 0Fh, 02h**
PAL 15 / 16 bpp Composite Sync Values	**26h, 80h, 1Fh, 50h, 07h, 07h, 19h, 0Fh, 02h**
PAL 24 bpp Composite Sync Values	**26h, 80h, 1Fh, 50h, 07h, 05h, 32h, 0Fh, 02h**

Table 9.7 BMP Page 7 - 8, Extended Register (XR) Boot Parameters

Description or Help	Value(s)
Extended Register (XR) Boot Parameters	
Extended Register (Address, Data)	Extended register boot table index and data
Flat Panel Register (FR) Boot Parameters	
Extended Register (Address, Data)	Flat panel register boot table index and data
Multimedia Register (MR) Boot Parameters	
Extended Register (Address, Data)	Multimedia register boot table index and data
CRT Display Switch Register Parameters	
Extended Register (Address, Data)	CRT display switch register table index and data

Table 9.8 BMP Page 9, Text Mode Vertical Stretching Parameters

Description or Help	Value(s)
640 x 480 Resolution Parameters [FR49 – FR4C]	
200 Scan Line Mode	00h, 00h, 00h, C1h
350 Scan Line Mode	20h, 01h, 20h, 00h
400 Scan Line Mode	00h, 01h, 80h, 00h
800 x 600 Resolution Parameters [FR49 – FR4C]	
200 Scan Line Mode	00h, 00h, 00h, FFh
350 Scan Line Mode	00h, 00h, 35h, ABh
400 Scan Line Mode	00h, 00h, 86h, 5Bh
1024 x 768 Resolution Parameters [FR49 – FR4C]	
200 Scan Line Mode	00h, EFh, 00h, 80h
350 Scan Line Mode	20h, 01h, 1Fh, FEh
400 Scan Line Mode	00h, 00h, 7Fh, FEh

The standard BIOS supports 16 panel classes. The following table holds 4 BMP screen that are repeated for each of the different panel classes. The values of this table are from panel number 6 which is the default panel.

Table 9.9 BMP Page 10 – 74, Panel Screens

Description or Help	Value(s)
Panel Configuration	
Panel Types	**640 x 480 18-Bit TFT Color**
Flat Panel Display Mode Clock Frequencies:	
Std / 4 / 8 Bpp Dot Clock	**20 MHz** (6 - 135 MHz)
15 / 16 Bpp Dot Clock	**20 MHz** (6 - 135 MHz)
24 Bpp Dot Clock	**20 MHz** (6 - 135 MHz)
Std / 4 / 8 Bpp Memory Clock	**100 MHz** (6 - 135 MHz)
15 / 16 Bpp Memory Clock	**100 MHz** (6 - 135 MHz)
24 Bpp Memory Clock	**100 MHz** (6 - 135 MHz)
Flat Panel Display Low Power Mode Clock Frequencies:	
Text Mode Memory Clock	**50 MHz** (6 - 135 MHz)
Std / 4 / 8 Bpp Memory Clock	**50 MHz** (6 - 135 MHz)
15 / 16 Bpp Memory Clock	**50 MHz** (6 - 135 MHz)
24 Bpp Memory Clock	**50 MHz** (6 - 135 MHz)
Simultaneous Display Mode Clock Frequencies:	
Dot Clock	**25 MHz** (6 - 135 MHz)
Std / 4 / 8 Bpp Memory Clock	**100 MHz** (6 - 135 MHz)
15 / 16 Bpp Memory Clock	**100 MHz** (6 - 135 MHz)
24 Bpp Memory Clock	**100 MHz** (6 - 135 MHz)
Panel #6 Boot Parameters (FR)	
Extended Register (Address, Data)	Boot table index and data values
Panel #6 Flat Panel Parameters (FR)	
Extended Register (Address, Data)	Flat panel table index and data values
Panel #6 Simultaneous Parameters (FR)	

Description or Help	Value(s)
Panel Configuration	
Panel Types	**640 x 480 18-Bit TFT Color**
Flat Panel Display Mode Clock Frequencies:	
Extended Register (Address, Data)	Simultaneous table index and data values

9.2 LogBIOS Video BIOS Utility

9.2.1 Introduction

Welcome to the LOGBIOS Video BIOS Function Logging utility. LOGBIOS is a debugging tool that is designed to assist engineers debugging Video BIOS and Graphic Controller related problems with minimal user intervention. LOGBIOS works by trapping INT 10H interrupt vector (Video ROM BIOS Interrupt) while the operating system boots up and logs the information in a file on the hard disk which can be viewed at a later point of time. LOGBIOS works across a variety of Operating System platforms that includes MS-DOS, Windows 3.x and Win’ 95. LOGBIOS is also easily manageable at run-time through a configuration utility.

9.2.1.1 Method of Operation

LOGBIOS consists of 2 modules:

1. a back-end module, LOGBIOS.SYS, which is a MS-DOS device driver
2. a front-end module, LOGBIOS.EXE, a MS-DOS executable that configures the back-end at run-time.

LOGBIOS.SYS is loaded through CONFIG.SYS and takes no parameters. LOGBIOS.EXE can take several parameters on the MS-DOS command line to control the operation of LOGBIOS.SYS at run-time. Logging is enabled as soon as the back-end is loaded.

Features provided by LOGBIOS.SYS are:

A re-entrant INT 10H handler with Nested INT 10H function support.
An INT 2FH MS-DOS Multiplex handler for communication with the front-end.
Low-level support for all the features provided in the front-end module.
Low-level support to handle file/disk I/O operations.

Features provided by LOGBIOS.EXE are:

Global Enable or Disable of LOGBIOS.SYS.
Selective Enable or Disable of INT 10H functions.
Binary Log Output File (typically, C:\LOGBIOS.OUT) conversion to an ASCII file.
Re-Initializing LOGBIOS.SYS device driver (typically required after a TSR BIOS is executed).
Change Log Output File (from default to user-defined file).
Specifying log output content level (whether descriptive or short).

9.2.1.2 Method of Operation of LOGBIOS.SYS

LOGBIOS.SYS is internally divided into 4 functional blocks: An MS-DOS device driver interface, an INT 10H handler, an INT 2FH handler, and a CPU register/VGA Register/BIOS Data Area dump module.

The device driver interface conforms to the MS-DOS character device driver specification and includes a device header, an initialization routine, an interrupt service routine, and a strategy routine. The device header is uniquely identified by its signature '$LOGBIOS' and provides the entry points to the device interrupt service and strategy routines. The initialization routine is invoked once during CONFIG.SYS processing and at run-time if LOGBIOS needs to be re-initialized.

The strategy and interrupt service routines exist only to support the initialization function and nothing else. The initialization routine aside from initializing the internal data structures traps Interrupt 10H and Interrupt 2FH vectors. It also captures the MS-DOS Busy and Critical Error Flags for use with MS-DOS INT 21H functions. For this purpose, it is assumed those MS-DOS versions 5.0 and above will be used because the above mentioned flag locations and their definitions vary from one MSDOS version to another before version 5.0.

A sign-on message is displayed after the device driver initialization is successful. At this point, logging is enabled as evident from a noticeable slowdown in video screen refresh that arises from the overhead of capturing log information and writing it out to a disk file.

The INT 10H handler is a re-entrant module designed to handle nested INT 10H function calls (that frequently occur in Windows like operating systems in notebook environments). The INT 10H handler has four major tasks:

- Detect if global logging is enabled and if the current INT10H function logging is enabled
- Log CPU, VGA registers and VGA BIOS Data Area Variables before and after performing the actual INT10H function
- Call the next INT 10H handler in chain to perform the actual INT 10H function
- Perform file I/O (if required) to write the log information

It also performs the additional task of handling nested INT 10H calls by stacking the log data internally before writing it to disk. There are 2 levels of logging provided in this utility: A global enable/disable control that instructs LOGBIOS.SYS whether logging should be performed at all or not and a function specific enable/disable that lets the user pick a function or a range of functions that need to be enabled or disabled. A range may be specified by the use of wild cards ('x' denotes wild card) in the command line option. The user through LOGBIOS.EXE command line options can instruct LOGBIOS.SYS if the specified range is for function logging exclusion or inclusion. Function range inclusions always supersede exclusions. One could, for example, specify the same function ranges for exclusion and inclusion and LOGBIOS would always include those functions for logging. LOGBIOS.SYS maintains all of this information internally and uses them on every INT 10H function call to determine if the function needs to be logged or not.

Three categories of data are logged for every INT 10H function: CPU registers upon entry and exit to the function, CHIPS VGA register set (including standard VGA registers) and VGA BIOS Data Area Variables (between 40:40h and 40:90h). It is not recommended to use LOGBIOS with non-CHIPS VGA controller.

File I/O is performed through the MS-DOS standard INT 21H interface. The default log file is C:\LOGBIOS.OUT and the log is written in binary and append mode. One has to keep in mind that since this file is opened in append mode, the file size could be enormous after "several" sessions of logging and hence the need for binary mode, logging enable/disable etc. The default log file can also be changed to an user-defined file (filenames have to be MS-DOS compliant (no long file names). There are occasions where one may not see any logging activity although logging is enabled which could be attributed to the DOS Busy flag being set.

9.2.1.3 Method of Operation of LOGBIOS.EXE

LOGBIOS.EXE is an MS-DOS executable that can also be run from a full-screen or windowed DOS box under Win 3.x and Win'95 operating systems. This front-end controls LOGBIOS.SYS and provides a few command-line options for finer control. LOGBIOS.EXE communicates with LOGBIOS.SYS through MS-DOS 2FH Multiplex Interrupt Interface. Executing LOGBIOS.EXE without installing LOGBIOS.SYS would result in an error. LOGBIOS.EXE without any command-line options reports the status of LOGBIOS.SYS. Space is used as the delimiter between options and options are denoted by a / sign.

9.2.2 LOGBIOS.EXE Command Line Switches

/Y Enable Global Logging

/X Enables LOGBIOS.SYS to log output for all INT 10H functions unless specified with option.

/N Disable Global Logging

Disables LOGBIOS.SYS from logging output for all INT 10H functions regardless of /I option.

/R Re-Initialize LOGBIOS.SYS

Re-Initializes LOGBIOS.SYS data structures and re-traps INT 10H vector. This is required any time INT 10H vector is reloaded (not chained) by some other program. For example, RAM BIOS and Device Driver BIOS reload INT 10H vector. In such cases, this option would chain LOGBIOS ahead of such programs.

WARNING! DO NOT USE THIS OPTION IN A WINDOWED OR FULL SCREEN DOS BOX AS IT MAY DESTABILIZE THE SYSTEM UPON EXIT FROM THE DOS BOX.

/Fname Change Log Output File Name

Change the output binary file name to an user-specified file. File names have to be specified with an absolute path including drive letter. There should be no space between the option and the file name. The file name is not optional. For example, D:\TEMP\MYDIR\MYLOG.OUT is correct. The default file name when LOGBIOS.SYS is loaded for the first time is C:\LOGBIOS.OUT.

/Xnnnn,nnnn,…. Exclude certain INT 10H functions from Logging

Excludes an INT 10H function or a range of functions or a list of function ranges from logging. Functions are four hexadecimal characters long with the exception of 'x' that denotes a wild card entry. Functions are delimited by the comma ',' character with no space anywhere within the option. Function codes are the same as in AX register when an INT 10H function is invoked. The following examples illustrate some of the possible scenarios:

/X0exx,4f02,xf19 - Exclude standard VGA functions 0e00 through 0eff, VESA function 4f02, and all functions whose uppermost nibble is between 0 and f and the lower 3 nibbles are f19.

/Xxxxx - Exclude all functions (standard, VESA, and Chips) from logging.

/Innnn,nnnn,…. Include certain INT 10H functions for Logging

Includes an INT 10H function or a range of functions or a list of function ranges for logging. Functions are 4 hexadecimal characters long with the exception of ‘x’ that denotes a wild card entry. Functions are delimited by the comma ‘,’ character with no space anywhere within the option. Function codes are the same as in AX register when an INT 10H function is invoked. The following examples illustrate some of the possible scenarios:

/I0exx,4f02,xf19 - Include standard VGA functions 0e00 through 0eff, VESA function 4f02, and all functions whose uppermost nibble is between 0 and f and the lower 3 nibbles are f19.

/Ixxxx - Exclude all functions (standard, VESA, and Chips) from logging.

A combination of /X and /I options in any order presented with the same function ranges will result in /X being superseded by /I. For example,

/Xxxxx /I4f02 would result in all functions excluded except VESA 4F02 function.

/Ixxxx /Xxxxx would result in all functions included.

/C[123]binfile,txtfile Convert a binary file to a descriptive or concise readable text file

Since the log output file is a binary file, this option would facilitate converting the log data into a presentable text format in the file specified in txtfile. There should no space anywhere within this option and the binfile should exist. If the txtfile already exists it will be truncated and recreated.

The ‘/C’ option instructs the converter to provide a detailed txtfile that will include CPU registers, VGA registers and BIOS data area variables related to Video. Here is a sample dump.

```
0021      AX    BX    CX    DX    SI    DI    DS    ES
[ In ]    5f51  0000  00ff  096c  0100  fffe  096c  096c  Switch Display Device
[ Out ]   015f  0000  00ff  096c  0100  fffe  096c  096c
[ Time ]  < 55 Msecs

MSR: 67
SR[ 00:08 ]  Register Dump:
00 ->     03    00    03    00    02    00    00    00

GR[ 00:09 ]  Register Dump:
00 ->     00    00    00    00    00    10    0e    00
08 ->     ff

AR[ 00:16 ]  Register Dump:
00 ->     00    00    00    00    00    00    00    00
08 ->     00    00    00    00    00    00    00    00
10 ->     04    00    0f    08    00    00



CR[ 00:80 ]  Register Dump:
00 ->     5f    4f    50    82    55    81    bf    1f
08 ->     00    4f    0d    0e    00    00    07    80
10 ->     9c    8e    8f    28    1f    96    b9    a3
```

```
18 ->   ff   00   00   00   00   00   00   00
20 ->   00   00   20   00   80   00   00   00
28 ->   00   00   00   00   00   00   00   00
30 ->   00   00   00   00   00   00   00   00
38 ->   00   00   00   00   00   00   00   00
40 ->   00   00   00   00   00   00   00   00
48 ->   00   00   00   00   00   00   00   00
50 ->   00   00   00   00   00   00   00   00
58 ->   00   00   00   00   00   00   00   00
60 ->   00   00   00   00   00   00   00   00
68 ->   00   00   00   00   00   00   00   00
70 ->   4f   00   00   00   00   00   00   00
78 ->   00   00   00   00   00   00   00   00

XR[ 00:FF ]  Register Dump:
00 ->   2c   10   e5   00   c3   00   fe   00
08 ->   03   00   00   01   00   00   00   00
10 ->   00   00   00   00   00   00   00   00
18 ->   00   00   00   00   00   00   00   00
20 ->   00   00   00   00   00   00   00   00
28 ->   00   00   00   00   00   00   00   00

30 ->   00   00   00   00   00   00   00   00
38 ->   00   00   00   00   00   00   00   00
40 ->   00   01   01   18   00   02   22   22
48 ->   00   00   00   00   00   00   00   09
50 ->   00   00   00   00   00   00   00   00
58 ->   00   00   00   00   00   00   00   00
60 ->   00   00   00   0c   00   00   00   00
68 ->   00   00   00   00   00   00   00   00
70 ->   de   fa   00   00   00   00   00   00
78 ->   00   00   00   00   00   00   00   00
80 ->   00   00   00   00   00   00   00   00
88 ->   00   00   00   00   00   00   00   00
90 ->   00   00   00   00   00   00   00   00
98 ->   00   00   00   00   00   00   00   00
a0 ->   00   00   00   00   00   00   00   00
a8 ->   00   00   00   00   00   00   00   00
b0 ->   00   00   00   00   00   00   00   00
b8 ->   00   00   00   00   00   00   00   00
c0 ->   0c   02   00   31   45   10   00   31
c8 ->   0c   03   00   21   13   04   a1   06
d0 ->   0f   00   00   00   00   00   00   00
d8 ->   00   00   00   00   00   00   00   00
e0 ->   07   01   03   07   00   40   55   41
e8 ->   14   52   00   00   00   00   00   00
f0 ->   00   00   00   00   00   00   00   00
f8 ->   00   00   00   00   00   00   00   00

FR[ 00:50H ]  Register Dump:
00 ->   05   01   00   08   81   01   03   00
08 ->   00   00   0c   01   00   00   00   02
10 ->   0c   e0   50   00   00   00   00   bd
18 ->   00   88   00   00   00   00   80   00
20 ->   63   68   18   7f   67   00   00   0f
28 ->   00   00   00   00   00   00   00   00
```

```
30 ->   57   58   0c   72   00   22   02   80
38 ->   00   00   00   00   00   00   00   00
40 ->   1f   03   00   00   00   00   00   00
48 ->   13   00   00   86   5b   50   3f   00
```

VGA BIOS Data Area Variables Dump:

```
40:40 ->  00  00  c0  00  00  00  00  00    00  03  50  00  00  10  00  00
40:50 ->  00  18  00  00  00  00  00  00    00  00  00  00  00  00  00  00
40:60 ->  0e  0d  00  d4  03  09  30  00    00  00  00  ff   10  e7  12  00
40:70 ->  00  00  00  12  00  02  08  00    14  14  14  34  01  01  01  01
40:80 ->  1e  00  3e  00  18  10  00  60    09  11  0b  00  58  01  00  04
```

Registers Different From Previous INT10 Call

```
XR[ 63 ]:  0e -> 0c
XR[ e0 ]:  27 -> 07
XR[ e1 ]:  03 -> 01
FR[ 01 ]:  02 -> 01
FR[ 05 ]:  21 -> 01
FR[ 16 ]:  03 -> 00
```

The '/C1' option instructs the converter to provide a concise txtfile that will include only CPU register log information. Here is a sample dump.

```
0026     AX     BX     CX     DX     SI     DI     DS     ES
[ In  ]  1130   0000   0000   0018   2e02   2e02   11e5   0000   Get Font Info
[ Out ]  1130   0000   0010   0018   2e02   2e02   11e5   c000
[ Time ]   < 55 Msecs
```

The '/C2' option instructs the converter to provide a txtfile that will include CPU register and VGA register log information.

```
0002     AX     BX     CX     DX     SI     DI     DS     ES
[ In  ]  5f54   0000   0000   0000   0000   0000   0f19   0f19   Set Panel ON / OFF
[ Out ]  015f   0000   0000   0000   0000   0000   0f19   0f19
[ Time ]  0001 Ticks

 MSR: 67

SR[ 00:08 ]  Register Dump:
00 ->   03   00   03   00   02   00   00   00

GR[ 00:09 ]  Register Dump:
00 ->   00   00   00   00   00   10   0e   00
08 ->   ff

AR[ 00:16 ]  Register Dump:
00 ->   00   00   00   00   00   00   00   00
08 ->   00   00   00   00   00   00   00   00
10 ->   04   00   0f   08   00   00

CR[ 00:80 ]  Register Dump:
00 ->   5f   4f   50   82   55   81   bf   1f
08 ->   00   4f   0d   0e   00   00   07   80
```

```
10 ->   9c   8e   8f   28   1f   96   b9   a3
18 ->   ff   00   00   00   00   00   00   00
20 ->   00   00   20   00   80   00   00   00
28 ->   00   00   00   00   00   00   00   00
30 ->   00   00   00   00   00   00   00   00
38 ->   00   00   00   00   00   00   00   00
40 ->   00   00   00   00   00   00   00   00
48 ->   00   00   00   00   00   00   00   00
50 ->   00   00   00   00   00   00   00   00
58 ->   00   00   00   00   00   00   00   00
60 ->   00   00   00   00   00   00   00   00
68 ->   00   00   00   00   00   00   00   00
70 ->   4f   00   00   00   00   00   00   00
78 ->   00   00   00   00   00   00   00   00

XR[ 00:FF ]  Register Dump:
00 ->   2c   10   e5   00   c3   00   fe   00
08 ->   03   00   00   01   00   00   00   00
10 ->   00   00   00   00   00   00   00   00
18 ->   00   00   00   00   00   00   00   00
20 ->   00   00   00   00   00   00   00   00
28 ->   00   00   00   00   00   00   00   00
30 ->   00   00   00   00   00   00   00   00
38 ->   00   00   00   00   00   00   00   00
40 ->   00   01   01   18   00   02   22   22
48 ->   00   00   00   00   00   00   00   09
50 ->   00   00   00   00   00   00   00   00
58 ->   00   00   00   00   00   00   00   00
60 ->   00   00   00   0e   00   00   00   00
68 ->   00   00   00   00   00   00   00   00
70 ->   de   fa   00   00   00   00   00   00
78 ->   00   00   00   00   00   00   00   00
80 ->   00   00   00   00   00   00   00   00
88 ->   00   00   00   00   00   00   00   00
90 ->   00   00   00   00   00   00   00   00
98 ->   00   00   00   00   00   00   00   00
a0 ->   00   00   00   00   00   00   00   00
a8 ->   00   00   00   00   00   00   00   00
b0 ->   00   00   00   00   00   00   00   00
b8 ->   00   00   00   00   00   00   00   00
c0 ->   0c   02   00   31   45   10   00   31
c8 ->   0c   03   00   21   13   04   a1   06

d0 ->   0f   00   00   00   00   00   00   00
d8 ->   00   00   00   00   00   00   00   00
e0 ->   07   01   03   07   08   40   55   41
e8 ->   14   52   00   00   00   00   00   00
f0 ->   00   00   00   00   00   00   00   00
f8 ->   00   00   00   00   00   00   00   00

FR[ 00:50H ]  Register Dump:
00 ->   05   01   00   08   81   01   03   00
08 ->   00   00   0c   01   00   00   00   02
10 ->   0c   e0   50   00   00   00   00   bd
18 ->   00   88   00   00   00   00   80   00
20 ->   63   68   18   7f   67   00   00   0f
```

```
28 ->   00   00   00   00   00   00   00   00
30 ->   57   58   0c   72   00   22   02   80
38 ->   00   00   00   00   00   00   00   00
40 ->   1f   03   00   00   00   00   00   00
48 ->   13   00   00   86   5b   50   3f   00

Registers Different From Previous INT10 Call
XR[ 61 ]:  0a -> 00
FR[ 05 ]:  09 -> 01
```

The '/C3' option instructs the converter to provide a txtfile that will include CPU register and VGA BIOS Data Area variable log information.

```
0023     AX     BX     CX     DX     SI     DI     DS     ES
[ In  ]  4f00   0000   7f50   0000   2e02   410e   11e5   11e5    VESA: Get Ctrl'er Info
[ Out ]  004f   0000   7f50   0000   2e02   410e   11e5   11e5
[ Time ]  < 55 Msecs

VGA BIOS Data Area Variables Dump:

40:40 ->    00  00  c0  00  00  00  00  00    00  03  50  00  00  10  00  00
40:50 ->    00  18  00  00  00  00  00  00    00  00  00  00  00  00  00  00
40:60 ->    0e  0d  00  d4  03  09  30  00    00  00  00  ff  14  e7  12  00
40:70 ->    00  00  00  12  00  02  08  00    14  14  14  34  01  01  01  01
40:80 ->    1e  00  3e  00  18  10  00  60    09  11  0b  00  50  01  00  04
```

/? Or /H Display a help screen with all these options and a brief explanation

9.2.3 Installation Instructions

The only files that are required for installation and proper run-time operation are LOGBIOS.SYS and LOGBIOS.EXE. Prior to installation, make sure at lease 20 MB of free space exists to accommodate a growing log output file. The installation needs are few and simple. Install LOGBIOS.SYS as a device driver in the bootable disk's CONFIG.SYS. This is done by adding the following line in CONFIG.SYS assuming LOGBIOS.SYS exists in the root directory.

```
DEVICE=C:\LOGBIOS.SYS
```

Save CONFIG.SYS and reboot the system. During the processing of CONFIG.SYS, if a copyright message (similar to the one below) appears with LOGBIOS mentioned then installation is complete. From then on, LOGBIOS.EXE can be executed to configure LOGBIOS.SYS.

9.2.4 Messages - Status, Warning and Error

There are 3 types of messages displayed by LOGBIOS: Status of LOGBIOS.SYS when operating normally, Warning messages and Error messages. A brief explanation of these messages is given below:

Normal Messages appear when LOGBIOS.SYS is installed and is functioning properly and when the user's request is carried out successfully. For instance, typing LOGBIOS on the command-line without any switches reports the status of LOGBIOS.SYS in a variety of messages, which may span several lines. Examples of these are:

Logging is Disabled:

This means LOGBIOS.SYS is disabled from logging any output.

Logging is Enabled:

LOGBIOS.SYS is logging output to a file.

LOG Binary File : c:\logbios.out

The current output file used for logging is c:\logbios.out, which is also the default output file.

INT 10H Functions NOT Logged:

0exxh
5fxxh

The INT 10H functions that are selectively not being logged are standard VGA function 0Eh (Write TTY Character) and all CHIPS Extended functions. The xx are wildcard entries.

INT 10H Functions Logged:

All other functions

All functions other than those specified under "INT 10H Functions NOT Logged" are logged. Using the above example, all standard VGA functions except 0Eh and all VESA functions are logged.

INT 10H Vector Recaptured:

LOGBIOS.SYS has rechained itself to the interrupt 10H vector .

Conversion Error: Unable to open input file [filename]

An invalid file was specified as the input file for conversion. Check file name, path, drive letter in the input file name specified.

Conversion Error: Unable to create output file [filename]

An output file name was specified that probably already refers to en existing file that is hidden/Read only or the disk is full/write-protected/does not exist.

Error LOGBIOS.SYS file is corrupted!:

LOGBIOS was not properly installed or bad original installation diskettes or files are infected by virus.

LOGBIOS Mismatch Error: LOGBIOS.SYS [ver] LOGBIOS.EXE [ver]

LOGBIOS.SYS version differs from that of LOGBIOS.EXE. ReInstall LOGBIOS from original diskettes.

Error: LOGBIOS.SYS is not installed in CONFIG.SYS

Install LOGBIOS.SYS in CONFIG.SYS before using LOGBIOS.EXE. Add "DEVICE=\path\LOGBIOS.SYS" in CONFIG.SYS and reboot the system.

Converted Binary File [filename] -> Text File [filename]:

A successful message indicating the specified conversion was performed.

Appendix A Building the Video BIOS

This appendix describes the process for creating binary BIOS from source code. Building, modifying, or updating the BIOS Source Code requires the following software utilities:

- A text editor capable of editing ASCII files
- Microsoft Macro Assembler (MASM) version 5.1.0
- Microsoft Linker (LINK) version 5.31
- Microsoft MAKE utility

The INSTALL.BAT file on the source code diskette will install all the files necessary to create a binary version of the BIOS. Use the following command line to install:

```
A:INSTALL C:\VGA030
```

INSTALL.BAT will create the directory \VGA030 on drive C:. The following subdirectories will also be created:

```
\OBJ030
\LST
```

The BIOS source files will be placed in the \VGA030 or directory, along with batch files for assembling and linking the BIOS. INSTALL.BAT will then copy the binary files into the \OBJ030 subdirectory.

To create the binary copy of the BIOS, run the Microsoft Make utility with the following command line:

```
MAKE MKF030.MAK or use MKB030.BAT
```

Note: Several “warning” messages may appear while Make is assembling certain source modules. These warning messages should be ignored. However, there should be no “error” messages from the assembler.

This page intentionally left blank.

Appendix B Suspend / Resume Procedure

B 1.0 Introduction

The following section describes the Suspend / Resume procedure required for the Chips and Technologies, Inc.

Following this procedure will allow the chipset to perform optimally during Suspend / Resume operations. Ignoring this procedure will result in rare intermittent failures during Suspend / Resume operations. Chips and Technologies, Inc. cannot be responsible for the operation of the chipset during Suspend / Resume if this procedure is not properly followed.

This section contains brief description of the procedure followed by an example code. The actual method that implements this Suspend/Resume procedure depends on the system logic chip set as well as the power management software. Please remember that it is only an example code. If you have questions regarding this procedure, please contact your local Chips and Technologies, Inc. sales office.

B 2.0 Operation

In Standby mode, the chipset suspends all CPU, memory, and display activities. It places the DRAM(s) in slow- or self-refresh mode (FR05[6]), and may shut off the 14.31818MHz reference clock and/or the 32KHz depending on the configuration of the chip during Standby.

In slow-refresh mode (FR05[6] = 1), using the internal RCLK (XRCF[3] = 0) for slow-refresh timing, the 14.31818MHz clock cannot be turned off. The 14.31818MHz clock generates the 37.5KHz RCLK used in the Standby slow-refresh timing. If the chip is using the external 32KHz on pin 154 (AA9) as the slow-refresh timing reference clock, then the 14.31818MHz clock can be shut off.

In self-refresh mode (FR05[6] = 0), the 14.31818MHz clock can be shut off. If the external 32KHz are also used, it can also be shut off during Standby.

The external 32KHz or internal RCLK is used for slow-refresh and panel power sequencing timing (XRCF[3]).

If the clock(s) may be shut off, they must be shut off after waiting twice the time programmed in FR04[3-0] (Panel Power Sequencing Delay Register - Power Down Delay) after the STNDBY# pin is asserted. This will allow the chip to completely finish all housekeeping activities after the STNDBY# pin is asserted.

When exiting Standby mode (Resume), the clocks must be applied (if turned off) to the chip and be stable before the STNDBY# pin may be de-asserted. After the STNDBY# pin has been de-asserted, the chip can be accessed after waiting twice the value time programmed in FR04[7-4] (Power Up Delay). This will allow the chip to fully come out of Standby.

The VGA subsystem dissipates a minimum amount of power during Standby. Since the chipset is a fully static device, the contents of the controller’s registers and on-chip palette are maintained during Standby. Therefore, Standby mode provides fast Suspend/Resume operations. Standby mode may be activated by asserting the STNDBY# pin low or programming FR05[4] = 1. The only way to come out of Standby is by de-asserting the STNDBY# pin.

The chipset has been designed to minimize power consumption during Standby in either Panel-only or Simultaneous modes. During these modes, it is assumed that AC power is not available and the system is running on batteries. During CRT-only mode, it is assumed that AC power is available and therefore power consumption does not need to be minimized.

To minimize power consumption during CRT-only mode, then it is recommended that the chip switch to Panel only mode before entering Standby.

The chipset has also been designed to enter Standby mode only from Normal operation mode, therefore the chipset cannot enter Standby mode when in the Panel-Off mode (FR05[3] = 1). To enter Standby mode from Panel-Off mode, it is recommended that it first come out of Panel-Off mode (Panel-On mode - FR05[3] = 0) then enter Standby mode.

It must be remembered that after setting FR05[3] = 0 (Panel-On mode), the chip cannot enter Standby mode until waiting twice the value time programmed into FR04[7-4]. This will allow the chip to fully come out of Panel-Off mode.

The following example procedure and code assumes the chip is in Panel-only or Simultaneous modes and Normal operation before entering Standby.

B 3.0 Procedure

In order to provide optimal Suspend/Resume operation (Standby mode) with the chipset, the following software procedure must be implemented in the system BIOS, or the power management software.

1. Before Entering Suspend Mode

   Software must execute the following procedure before asserting the STNDBY# pin of the chipset:

   a) SAVE the contents of register 3C6h (Color Palette Pixel Mask Register).

   b) PROGRAM register 3C6h to 00 - Disabling access to palette contents.

   c) SAVE all DAC registers (Video DAC State and Color Registers) using the Video BIOS function call 5FA1h.

   d) Enter Standby mode by asserting the STNDBY# pin of the chipset.

   Wait a minimum time delay of twice the value programmed into register FR04[3-0] (Panel Power Sequencing Register - Power Down) in Msec. before turning off the external 14.31818MHz oscillator (if applicable). This allows the chipset to completely finish all activities ('housekeeping') after the STNDBY# pin is asserted .

2. After Exiting Suspend Mode (Resume)

   The 14.31818MHz external oscillator must be applied to the chipset and be stable before de-asserting the STNDBY# pin (if applicable). After de-asserting the STNDBY# pin of the chipset, the software must execute the following procedure:

   a) Wait a minimum time delay of twice the value programmed into register FR04[7-4] (Power Up) in Msec. This allows the chipset to completely come out of Standby after the STNDBY# pin is de-asserted.

   b) RESTORE all DAC registers (Video DAC State and Color Registers) using the Video BIOS function call 5FA2h.

   c) RESTORE the saved contents of register 3C6h (Color Palette Pixel Mask Register).

The following pages show an example of the code.

Example Code

```
;---------------------------------------------------------------------------
; Module Name          : STANDBY.asm
; Program Name         : STANDBY.com
; Description          : Standby (69K)
; Date                 : Sept 6, 1994
; Version              : 1.0
; Programmer           : Chips and Technologies, Inc.
; (C) 1995 Chips and Technologies, Inc.
;----------------------------------------------------------------------------
; Code Segment Starts
        code    segment
        assume  cs:code, ds:code, ss:code, es:code
        org     100h; for making program .COM type
begin:

STANDBY proc  near

;*********************************************************************************************
; This delay routine is in case the system is already in Standby.
;*********************************************************************************************
        mov     dx,3d0h                         ; Set to FR Index
        in      al,dx                           ; Read FR Index
        push    ax                              ; Save FR Index
        mov     al,04h                          ; Set Index to FR04
        out     dx,al
        in      ax,dx                           ; Read contents of FR04 (Panel Power Sequencing Delay
Register)
        mov     CS:FR04,ax                      ; Save FR04
        pop     ax
        out     dx,al                           ; Restore FR Index
        call    Wdelay                          ; delay
```

```
;PREPARE FOR STANDBY
        mov     dx,3c6h
        in      al,dx                   ; Read Color Palette Pixel Mask Register
        mov     CS:D_3c6,al             ; Save Color Palette Pixel Mask Register
        mov     al,0
        out     dx,al                   ; Disable access to Palette contents
        mov     ax,5fa1h                ; Video BIOS function call to Save Video State
        mov     cx,04h                  ; Video DAC state
        push    cs
        pop     es
        mov     bx,offset Buff_DAC      ; Set Correct buffer
        int     10h                     ; Save

; [ A ] STANDBY
;*****************************************************************************************
; Code to enter Standby should be placed here.  This code depends on the Standby
; implementation in the System
;*****************************************************************************************
        call    Sdelay                  ; delay
;*****************************************************************************************

; [ B] WAKE UP
;*****************************************************************************************
;
; Code to exit Standby should be placed here.  This code depends on the Standby
; implementation in the System.
;
;*****************************************************************************************
        call    Wdelay                  ;delay
; RESTORE STATE AFTER WAKEUP
        mov     ax,5fa2h                ; Video BIOS function call to Restore Video State
        mov     cx,04h                  ; Video DAC state and Color Registers
        push    cs
        pop     es
        mov     bx,offset Buff_DAC      ; From buffer
        int     10h                     ; Restore
        mov     dx,3c6h                 ; Set to Color Palette Pixel Mask Register
        mov     al,CS:D_3c6
        out     dx,al                   ; Restore Color Palette Pixel Mask Register
        ret                             ; Terminate

STANDBY         endp

; Wake-up delay routine
Wdelay  Proc    near
```

```
          mov     ax,CS:FR04
          and     ah,0f0h                         ; Select Power Up Delay (bits 4 - 7)
          .386
          shr     ah,1                            ; 8 times
          .286
          xor     cx,cx
          mov     cl,ah                           ; CX = delay count in msec
          call    delay                           ; Call User System Specific Delay Routine
          ret

Wdelay    endp

; Standby delay routine
Sdelay    Proc    near
          mov     ax,CS:FR04
          and     ah,0fh                          ; Select Power Down Delay (bits 3 - 0)
          mov     al,ah
          mov     ah,2*29                         ; msec
          mul     ah
          mov     cx,ax                           ; CX = delay count in msec
          call    delay                           ; Call User Specific Delay Routine
          ret

Sdelay    endp

; [C] User Specific Delay Routine
;*******************************************************************************************
; Code to implement User Specific Delay Routine goes here.
; This is to allow User to implement the delay routine based on the system requirements.
;
; Entry: CX = delay count in msec
;*******************************************************************************************
delay     Proc    near                            ; Delay

          ret
delay     endp
;*******************************************************************************************

;-------------------------
; Data Declaration
;-------------------------
D_3c6             db      ?
FR Index          db      ?
FR04              dw      ?
Buff_DAC          db      1000h dup(0)
```

```
code    ends
        end     begin
```

Note : [A], [B], and [C] should be implemented based on system requirements.

Appendix C Tabs

The tabs on this page are designed to help use this document as a video BIOS reference document. Simply cut them out and tape them to the appropriate chapters. On the first page of these chapters is a block that indicates where the tabs go.

Note: Due to the copying process these marks may be a little out of place. Check them before taping the tabs down.

	MODES	MODES	
	VGA	VGA	
	Ext.	Ext.	
	VESA	VESA	
	Hooks	Hooks	
	Utils	Utils	

Intel Corporation
350 East Plumeria Drive
CHP3-102
GCD San Jose
San Jose, California 95134
Ph#   408-765-8080
FAX# 408-545-9812
www.intel.com

Title: VGA BIOS OEM Reference Guide
Publication No.: UG180
Stock No.: 050-180-000
Revision No.: 1.0
Date: 07/2/99