VESA® VXE 1.0 Standard

Video Electronics Standards Association

2150 North First Street, Suite 440
San Jose, CA 95131-2020

Phone: (408) 435-0333
FAX: (408) 435-8225

VESA XGA* Extensions Standard

Standard # VXE 1.0
May 8, 1992

PURPOSE

To standardize a software interface to XGA-compatible video subsystems, independent of the system bus type, in order to provide uniform driver and application access to XGA-compatible products.

SUMMARY

This standard defines a set of register and bit functions which allow software developers to write a single version of an application or driver for XGA-compatible video subsystems in EISA, Micro Channel Architecture® and ISA Personal Computer Systems. It also defines a set of functions accessible via INT 10h as an extension to the BIOS video services. The functions defined provide information about capabilities and characteristics of a specific hardware implementation and provide control of video mode setting and video buffer access. The defined functions explicitly do not address any drawing engine functionality.

Specific areas covered include bus master support for all three industry standard PC buses, video subsystem setup requirements, and issues related to multiple video subsystems. Special attention is given to ISA resource limitations with respect to XGA-compatible configurations.

Trademarks

The trademarks mentioned in this document are credited to the respective corporations listed below:

Trademark	Corporation
IBM®	IBM Corporation
VGA	IBM Corporation
XGA®	IBM Corporation
Micro Channel Architecture®	IBM Corporation
Personal System/2®	IBM Corporation
EISA	Compaq
Microsoft	Microsoft Corporation
MS-DOS	Microsoft Corporation
INMOS®	INMOS Limited
iAPX	Intel Corporation
VESA®	Video Electronics Standards Association

Revision History

1.0 Initial release: May 8, 1992

Table of Contents

1. Introduction

This document contains a specification for a standardized interface to XGA video modes and functions. The specification consists of mechanisms for supporting standard and extended video modes and functions that have been approved by the VESA General Membership, in a uniform manner that application software can utilize without having to understand the intricate details of the particular XGA hardware. In particular, the interface is intended to allow a single application or driver to run on the three standard PC platforms: ISA, EISA and Micro Channel Architecture.

The primary topics of this specification are the definitions of the interface to XGA video hardware and the definitions necessary for application software to understand the characteristics of the video modes and manipulate the memory associated with the video modes.

Readers of this document should already be familiar with programming XGAs at the hardware level and Intel iAPX real mode assembly language. Readers who are unfamiliar with programming the XGA should first read the *Update for the PS/2 Hardware Interface Technical Reference: Common Interfaces, Video Subsystems* published by IBM Corporation.

1.1. Recommended Reference Documents

EISA reference: *EISA Specification*, BCPR Services, 1201 New York Ave, N.W., Suite 1225, Washington, D.C. 20005; (202) 962-8349.

Micro Channel reference: *Update for the PS/2 Hardware Interface Technical Reference: Architectures*, September 1991 (IBM document number S04G-3282-00, may be obtained from (800) IBM-PCTB).

ISA reference: *IBM AT Technical Reference* (IBM document number 1502494).

Feature Connector references: *VESA 8514/A Standard* (VS890802) and *VGA Pass Through Connector Standard* (VS890803).

BIOS references: *Super VGA BIOS Extension* (VS911022) and *Super VGA Protected Mode Interface* (VS11020).

XGA and VGA references: *Update for the PS/2 Hardware Interface Technical Reference: Common Interfaces, Video Subsystems*, September 1991 (IBM document number S04G-3281-00, may be obtained from (800) IBM-PCTB).

1.2. Goals and Objectives

The IBM XGA has become the new high end graphics standard on the PS/2 platform. Several XGA-style graphics adapters will soon be coming to the marketplace, each one providing compatibility with the IBM XGA. These XGA-compatible products will implement various super sets of the XGA

standard. These extensions range from higher resolutions and more colors to improved performance and support of other platforms besides the PS/2.

However, several problems face a software developer who intends to take advantage of these XGA graphics adapters. Because of the various hardware platforms, the developer is faced with disparate XGA implementations. Lacking a common software interface, designing applications for these environments would be costly and technically difficult. Except for applications supported by OEM-specific display drivers, very few software packages could take advantage of the power and capabilities of XGA products.

With the above issues in mind, the VESA XGA Technical Committee adopted the following set of objectives:

- Develop a standard for XGA-compatible subsystems on EISA, Micro Channel Architecture and ISA platforms.
- Remain IBM XGA-compatible.
- Provide equivalent functionality to IBM's Micro Channel based XGA.
- Provide a robust standard.
- Require only the minimum of environment specific software.
- Ensure that software which was written solely to the IBM XGA will operate, if possible.
- Allow expansion and product differentiation within the VESA XGA Standard.
- Define a standard access and/or ID function for any unique extensions or special requirements for any manufacturer.

2. Overview of Document

This document is broken into two major areas, including 1) the VESA XGA Hardware Specification and 2) the VESA XGA Real Mode BIOS Specification. The Hardware Specification is broken into XGA Architecture, Setup, EISA Systems, Micro Channel Systems, and ISA Systems sections. The Software Specification is broken into Software Goals and Objectives, OEM-Defined Mode Tags, and VESA XGA BIOS sections. This document also includes a Programming Considerations Appendix, an Adapter Identification Appendix and a Multiple Video Subsystem Appendix.

All addresses, register values and bit fields are in hexadecimal, unless otherwise noted. Byte order is low byte/high byte at increasing addresses. Bit definitions are assumed to start from zero (0). A logic high is defined as a one (1) and a logic low as zero (0). Reserved bits shown by a pound sign (#) **must** be read and have their values preserved during a write. Reserved bits shown by a dash (-) are reserved and should be written with a logic low, unless otherwise noted. Bits shown by a one (1) are reserved and must be written as a logic high.

I. VESA XGA Hardware Specification

3. XGA Architecture

3.1. Bus Architecture ID

Due to the differences among the three industry standard PC bus architectures, it may be necessary for software to identify the host bus at run time. The BIOS function calls defined in the XGA BIOS provide the necessary bus information. In addition, bits are provided in the XGA subsystem to identify the host bus type and width.

3.2. Hierarchical Compliance

VESA compliance requires compatibility with the IBM XGA as defined in IBM document number S04G-3281-00, with the exception of modifications defined in this specification, and one of the following two classes:

A. Products which are 100% IBM XGA-compatible,**with no extensions**, are not required to include the VESA items in section B below. These products must use the appropriate POS ID of the IBM XGA with which they are compatible.

OR

B. Compliance with all of the following:

1. The POS Enable mechanism defined for each bus type is required for that bus type. Only that mechanism required for a specific bus type may be used on that bus type.

2. The Micro Channel Architecture requires a unique POS ID number for each model of video subsystem. The POS ID number is assigned by IBM and may be obtained through VESA. A manufacturer's EISA and/or ISA products should use the same POS ID number as their corresponding Micro Channel Architecture products. Manufacturers without a Micro Channel Architecture POS ID should use the POS ID number of the IBM XGA product with which they are compatible.

3. External Memory Enable, POS 3 bit 1, is required.

4. VESA Manufacturer ID at POS Data 4, Index 1 is required and the value is assigned by VESA.

5. If the bytes at POS Data 4, Index 2 and XGA Index 75h donot contain ID information, these bytes must read back as zero (0).

6. The bits defined in the Auto Configuration register (XGA Index 4) are required. Not all bus types and sizes need to be supported.

7. DMA Channel Readback (XGA Index 74h) is required in ISA systems only. Not all DMA channels need to be supported.

8. XGA Index 76h and 77h are only used for manufacturer-specific extensions.

9. All XGA video subsystems supplied with a video BIOS must comply with the requirements in Section II, VESA XGA Real Mode BIOS Specification.

10. All XGA video subsystems supplied with a video BIOS must include four bytes of storage for the BIOS chain vector.

3.3. Manufacturer ID Mechanism

Each XGA video subsystem has up to seven bytes of manufacturer ID in hardware. The first two bytes are the POS ID at POS 0 and 1. IBM has allocated 1024 (1K) POS IDs for VESA-compliant Micro Channel XGA products.

VESA XGA POS IDs
0240h-027Fh
0830h-0A7Fh
0A90h-0BFFh

These IDs are available for Micro Channel systems only. Manufacturers developing Micro Channel and ISA/EISA subsystems must use the same ID for both systems. Assignment of POS IDs is controlled by IBM. IDs may be requested through VESA. Manufacturers designing only ISA/EISA subsystems should use the POS ID of the IBM XGA product with which they are compatible (8FD8h-8FDBh or 8FD0h-8FD3h).

Additional ID bytes are contained in POS 4, Index 1 and 2. Encoding for the ID byte at POS 4, Index 1 is assigned by VESA. The second ID byte is manufacturer-specific and optional. It is suggested that the silicon manufacturer use this byte to identify product type and revision. If this byte is not used, it must read back as zero (00h).

One or more bytes of subsystem vendor ID information is provided in the standard XGA index space (21xAh) at Index 75h. This byte is used for subsystem vendor identification and is assigned by the silicon supplier. If this byte is not used, it must read back as zero (00h). Two additional bytes at Indices 76h-77h are reserved for manufacturer-specific functions. These bytes are manufacturer-specific and optional.

4. Setup

The subsystem location mechanism is based on allowing the IBM-defined Micro Channel Architecture POS register definitions to be used in all three bus configurations. This allows a minimum of additional chip hardware and application software by encoding subsystem configuration information independently of the bus.

The extension registers in this document fall into several categories:

- Registers required to allow access to the POS information.
- Registers which contain configuration information required to support non-Micro Channel Architecture systems.
- Registers to support the identification of silicon manufacturer and subsystem vendor ID.

4.1. POS Mechanism

The POS Mechanism is used to locate the XGA subsystem in the system memory and I/O space. Registers are provided to control memory location and bus interface controls. These registers are enabled during the "setup" phase of system configuration and when software needs to determine the location of subsystem resources. The setup mechanism for each bus environment is as follows:

- Micro Channel Architecture: Use the IBM-prescribed method using the INT 15h BIOS call and register 94h to enable the POS registers.
- ISA: Use 108h-10Fh to enable the POS registers.
- EISA: Read the POS data from slot-specific addresses zC88h-zC8Fh or use the EISA Setup register to enable the POS registers.

4.2. Extension Register Descriptions

The following register descriptions list the extension registers for a VESA-compatible XGA subsystem. There has been no attempt to document registers which exist within an IBM XGA; therefore, all registers present in the IBM XGA documentation which are **not** described below are unchanged from the IBM XGA implementation. Also, these descriptions cover all system configurations (ISA, EISA, and Micro Channel Architecture) and a specific subsystem could omit registers or fields based on a desired configuration.

Extension registers have been added at POS Data 4, Indices 1-3 and to the XGA indexed registers at 21xAh, Indices 74h-77h. Indices 76h and 77h have been allocated for manufacturer expansion. Indices 72h and 73h are reserved for future expansion of the VESA XGA specification and may **not** be used for manufacturer expansion.

While they are not explicitly listed below, the POS registers implemented within the XGA are re-mapped to slot-specific addresses in EISA systems, starting at zC88h. This is shown in the register descriptions for POS 3, POS 6, POS 7, POS Data 3 and POS Data 4, discussed below.

4.3. POS Register Extensions

The following extensions have been added to the IBM defined POS register specification. Features include mechanisms to support Manufacturer ID and a VGA-style BIOS. These registers are present only during setup and are addressed at the POS locations of 100h-107h.

4.3.1. POS 3

Register Name: **POS 3** Register Address: 103h[1] or 0zC8B - Index 0							
bit 7	*bit 6*	*bit 5*	*bit 4*	*bit 3*	*bit 2*	*bit 1*	*bit 0*
#	#	#	#	#	#	External Memory Enable	#

1 Setup Only

Bit 1*:* External Memory Enable

This bit is used to enable the 8KB XGA External Memory space. When it's value is 1, access to the external memory is enabled. The memory address is determined by the value in the POS 2 register. Note that XGA subsystems do not necessarily contain an 8KB ROM. This bit has no effect on the accessibility of the XGA memory registers.

Bits 7-2 & 0*:* Manufacturer Reserved

The function of these bits are bus-type and manufacturer-specific. If this register is written, the value contained in these bits **must** be preserved.

4.3.2. POS 3, 4, 6 & 7

POS registers 3, 4, 6 and 7 are used to access one of 128K possible 8-bit registers. The index to the registers is placed in POS 6 (low byte) and POS 7 (high byte). The registers are then visible through the data ports at POS 3 and4.

The implementation of POS 3, Index 0 and POS 4, Index 0 is fully compatible with the function of POS 3 and POS 4 in the IBM XGA product.

At this time only Index 0 is defined for POS Data 3. For POS Data 4, only Indices 0 through 3 are defined. In order to maintain compatibility with future specifications, the full 16-bit index value **must** be used.

4.3.3. POS 6

Register Name: **POS 6** Register Address: 106h[1], zC8Eh[2]							
bit 7	*bit 6*	*bit 5*	*bit 4*	*bit 3*	*bit 2*	*bit 1*	*bit 0*
POS Index Low							

1 Setup Mode only

2 EISA systems only

4.3.4. POS 7

Register Name: **POS 7** Register Address: 107h[1], zC8Fh[2]							
bit 7	*bit 6*	*bit 5*	*bit 4*	*bit 3*	*bit 2*	*bit 1*	*bit 0*
POS Index High							

1 Setup Mode only

2 EISA systems only

POS 7 & 6*:* Index High and Low

These two fields define a 16-bit register index value. Values 0004h through 000Fh are RESERVED and should not be used, all greater values are undefined. The entire 16-bit value **must** be used to access the data registers at POS 3 and POS 4. POS 7 is the high order 8 bits and POS 6 is the low order 8 bits. The reset value of POS 6 and POS 7 must be 0.

4.3.5. POS Data 3

Register Name: **POS Data 3** Register Address: 103h[1], zC8Bh[2]							
bit 7	*bit 6*	*bit 5*	*bit 4*	*bit 3*	*bit 2*	*bit 1*	*bit 0*
POS Data 3							

1 Setup Mode only

2 EISA systems only

4.3.6. POS Data 4

Register Name: **POS Data 4** Register Address: 104h[1], zC8Ch[2]							
bit 7	*bit 6*	*bit 5*	*bit 4*	*bit 3*	*bit 2*	*bit 1*	*bit 0*
POS Data 4							

1 Setup Mode only

2 EISA systems only

POS 4 & 3*:* Data Registers

POS Data 3 and POS Data 4 are the data registers used to access the POS extensions. POS Data 3 is undefined for non-zero index values. POS Data 4 is currently defined for indices 0-3 only.

4.3.7. Manufacturer ID

Register Name: **Manufacturer ID** Register Address: POS Data 4 - Index 1-2[1]							
bit 15	*bit 14*	*bit 13*	*bit 12*	*bit 11*	*bit 10*	*bit 9*	*bit 8*
Manufacturer ID 15-8[2]							
bit 7	*bit 6*	*bit 5*	*bit 4*	*bit 3*	*bit 2*	*bit 1*	*bit 0*
Manufacturer ID 7-0[3]							

1 Setup Mode only

2 Optional

3 Required, VESA assigned

This 2-byte field contains the Manufacturer ID. Only the first byte (Index 1, bits 7-0) of the ID is required and the value read from this field is assigned by the VESA organization. The second byte (Index 2, bits 7-0) is manufacturer-specific and will read back the value 0 if it does not contain ID information. Since this field may be implemented as read/write or read only, software should take care to assure that this field is not written.

4.3.8. VGA BIOS Configuration

Register Name: **VGA BIOS Configuration** Register Address: POS Data 4 - Index 3							
bit 7	*bit 6*	*bit 5*	*bit 4*	*bit 3*	*bit 2*	*bit 1*	*bit 0*
-	-	-	-	VGA BIOS ROM Decode Location			VGA BIOS ROM Decode Enable

Bit 0*:* VGA BIOS ROM Decode Enable

This bit enables decoding of the VGA BIOS ROM within the XGA subsystem. This bit has no effect on the XGA-compatible relocatable 8KB ROM Aperture. When this bit is 1, the VGA BIOS is decoded at the location determined by the VGA BIOS ROM Decode Location field (see below). When this bit is 0, the XGA subsystem does not respond to VGA BIOS accesses. This bit does not affect accessibility to the XGA memory registers.

Bits 3-1*:* VGA BIOS ROM Decode Location

This field determines the VGA BIOS addresses to which the XGA subsystem responds. If the VGA BIOS and the XGA external memory are decoded at the same location, the XGA external memory has priority. The starting address of the 32KB VGA BIOS is encoded as follows:

Value	Starting Address
000	C000:0
001	C800:0
010	D000:0
011	D800:0
100	E000:0
101	E800:0
110	Reserved for future VESA use
111	Reserved for future VESA use

Note that if the 8KB External Memory space is programmed to overlap with the 32KB VGA BIOS space, accesses to the overlapping locations will access the 8KB External Memory.

4.4. XGA Index Register Extensions

The following extensions have been added to the IBM defined XGA register set. Features include Bus Identification, Subsystem Vendor ID, Manufacturer Expansion and VESA-reserved registers. These registers are accessed through the standard XGA index and data space at 21xAh and 21xBh through 21xFh, respectively.

4.4.1. Auto-Configuration

Register Name: **Auto-Configuration** Register Address: 21xAh, Index 4							
bit 7	*bit 6*	*bit 5*	*bit 4*	*bit 3*	*bit 2*	*bit 1*	*bit 0*
-	-	Subsystem Interface Configuration		Bus Size 1	-	-	Bus Size 0

The bits in this register may be configured at power up by strapping options. This register must not be written.

Bits 3 & 0*:* Bus Size 1 & 0

These bits indicate the bus size as defined below:

Bits	Bus Size
00	16-bit
01	32-bit
10	8-bit
11	RESERVED

Bits 5-4*:* Subsystem Interface Configuration

This 2-bit field is used to read the configuration of the XGA subsystem's host interface. The system bus selection bits are encoded below.

Value	System Bus Type
00	Micro Channel
01	ISA
10	RESERVED
11	EISA

4.4.2. Index 72h-73h

XGA Index registers 72h and 73h are reserved by VESA for future expansion of the VESA XGA standard and may not be used.

4.4.3. DMA Channel Readback (ISA Only)

Register Name: **DMA Channel Readback**
Register Address: 21xAh, Index 74h[1]

bit 7	*bit 6*	*bit 5*	*bit 4*	*bit 3*	*bit 2*	*bit 1*	*bit 0*
-	-	-	-	DMA Channel Select			DMA Channel Enable

1 ISA Systems only

This register is read only. Application software must not attempt to write this register.

Bit 0*:* DMA Channel Enable

This bit is used to indicate whether the DMA Channel is enabled for bus mastering.

Bits 3-1*:* DMA Channel Select

This field encodes which DMA channel is selected for bus master arbitration in an ISA system bus. This field is encoded as follows:

Value	ISA DMA Channel Number
000	Channel 0
001	Channel 1
010	Channel 2
011	Channel 3
100	RESERVED
101	Channel 5
110	Channel 6
111	Channel 7

4.4.4. Subsystem Vendor ID

Register Name: **Subsystem Vendor ID**
Register Address: 21xAh, Index 75h

bit 7	*bit 6*	*bit 5*	*bit 4*	*bit 3*	*bit 2*	*bit 1*	*bit 0*
Subsystem Vendor ID							

Bits 7-0*:* Subsystem Vendor ID

This byte is used to identify the subsystem (board or mother board) vendor. A write to the XGA Index Register (21xAh) initializes this register. After initializing the register, the first read will return one of three sets of values:

Value	Description
00h	The Subsystem Vendor ID mechanism has not been implemented.
01-FEh	Chip manufacturer-assigned subsystem vendor ID. All further subsequent reads are undefined, and may be used by either the chip manufacturer or the subsystem vendor.
FFh	The software must read this register twice more to get a 16-bit, VESA-allocated subsystem vendor ID. The first subsequent read is the low-order byte of the 16-bit VESA assigned ID, while the second subsequent read is the high-order byte. All further subsequent reads are undefined, and may be used by either the chip manufacturer or the subsystem vendor. (Note that this definition is optional, but if not implemented, then the FFh value is reserved and should not be assigned by the chip manufacturer).

This register may be implemented as RAM or ROM and software other than initialization software must not write to this register.

4.4.5. Manufacturer Expansion

XGA Index registers 76h and 77h are allocated for manufacturer expansion. These two registers are manufacturer-specific and cannot be relied upon to be consistent across different products. Note that these registers should not be accessed without knowledge of the manufacturer's implementation.

4.4.6. BIOS Chain Function

For those XGA subsystems supplied with a BIOS, four 8-bit storage locations are required for storage of a double word value. These locations are used to store the double word value for the startup video subsystem's INT 10h vector. The location and implementation of this BIOS Chain function is manufacturer-specific and cannot be relied upon to be consistent across product lines.

5. EISA Systems

This section discusses issues related to implementation of an IBM XGA-compatible video subsystem in the EISA bus structure, including:

- Bus Hardware Interface
- EISA Registers
- EISA Implementation of POS Registers for XGA

5.1. Bus Hardware Interface

5.1.1. Address Range and Data Range

Both EISA and Micro Channel Architecture have 32-bits of address and 32-bits of data. Therefore, EISA XGA implementation have the same address range capabilities as the Micro Channel Architecture XGA.

5.1.2. I/O Address and Data Range

Because of the slot-specific implementation of the EISA bus, there are some limitations of addressing the full range of the XGA I/O address. In particular, Instance 0 and 7 are not recommended to be implemented in the EISA system. The Instance 0, if implemented in the EISA system, causes potential conflict of I/O registers with other products. Instance 7 has potential conflict with the CPU. Although the XGA hardware specification indicates Instance 0 and 7 may be used, the IBM XGA is not configured to use Instances 0 and 7. Therefore, the limitation of the EISA system has no effect on the current software that supports the IBM XGA.

The EISA bus requires a set of I/O registers (four bytes) for manufacturers to specify their product identification. The I/O register address is 0zC80h to 0zC83h, where the "z" is the slot number.

5.1.3. Bus Mastership

EISA allows bus mastership by the expansion board. The EISA bus master arbitrator uses a rotational arbitration, whereas the Micro Channel Architecture has priority level based arbitration. This is a hardware implementation which is not seen by software; therefore, EISA versions of XGA can perform bus master without any change.

5.1.4. POS Registers

The POS registers are available through the EISA slot-specific addresses at all times. They are accessible through I/O addresses 100h-107h when enabled by the EISA Setup register described below.

5.1.5. EISA Setup

Register Name: **EISA Setup** Register Address: zC85h[1]							
bit 7	*bit 6*	*bit 5*	*bit 4*	*bit 3*	*bit 2*	*bit 1*	*bit 0*
-	-	-	-	-	-	-	Setup Mode Enable

1 EISA systems only

This register is only accessible in systems using the EISA bus, as identified by the Subsystem Interface Configuration. In these systems, this register is located at a slot-specific address. Thus, this register may be located at any one of 16 addresses in the EISA I/O address space, depending on the system bus slot to which the XGA subsystem is configured to respond.

Bit 0*:* Setup Mode Enable

This bit is used to enable access to the POS registers. When Setup Mode Enable is 1 and EISA Access Enable Arm is 1, the CPU may access the POS registers through addresses 100h-107h.

5.1.6. EISA Implementation of POS Registers for XGA

The EISA version of XGA has POS registers in order to tell the host of its configuration. The POS registers reside after the EISA ID and EISA Control registers, following the same order as in the IBM XGA. The following table shows the address correspondence:

EISA XGA I/O Address	IBM XGA I/O Address
0zC80h (EISA ID)	
0zC81h (EISA ID)	
0zC82h (EISA ID)	
0zC83h (EISA ID)	
0zC84h (Expansion Board Control Bits)	
0zC85h (Setup Control)	
0zC88h	100h (POS 0)
0zC89h	101h (POS 1)
0zC8Ah	102h (POS 2)
0zC8Bh	103h (POS 3)
0zC8Ch	104h (POS 4)
0zC8Dh	105h (POS 5)
0zC8Eh	106h (POS 6)
0zC8Fh	107h (POS 7)

The location for the POS registers is used because 0zCxx in the EISA bus is a slot-specific address range designed not to overlap with the ISA I/O alias or equivalent locations on other EISA adapters.

6. Micro Channel Systems

Products which are 100% IBM XGA compatible, with no extensions, may use the product ID of 8FD8h-8FDBh or 8FD0h-8FD3h, based on the ID of the IBM XGA with which they are compatible. All other subsystems use one of the 1K reserved XGA IDs. The reserved XGA values are 0240h-027Fh, 0830h-0A7Fh and 0A90h-0BFFh. A valid POS ID must be requested from IBM or through VESA.

7. ISA Systems

This section describes the bus master and memory mapping functions of XGA on the 16-bit ISA bus. The ISA XGA implementation outlined in this section provides full programming interface compatibility with the Micro Channel Architecture and EISA XGA platforms. This compatibility is achieved by implementing a POS mechanism for ISA adapters and by the POS register settings themselves.

This standard defines a single instance of XGA in an ISA system. The current Instance number is limited to a value of one (1). There is nothing in this standard which precludes multiple instances.

7.1. ISA POS Enable

The mechanism to enable the POS registers on the ISA bus requires that the Instance number be used as both data and address. The address is computed as (108h + Instance). The data written to the lower nibble must also match the Instance + 8 to enable setup. If the Instance number fields in the address and data do not match, the XGA subsystem will not recognize the access.

Register Name: **ISA POS Enable** Register Address: 108h + Instance[1]							
bit 7	*bit 6*	*bit 5*	*bit 4*	*bit 3*	*bit 2*	*bit 1*	*bit 0*
-	-	-	-	Setup Mode Enable	Instance Number		

1 ISA systems only

Only Instance 1 is currently defined; therefore,**this register must be accessed at location 109h** .

Bits 2-0*:* Instance Number

These bits must match the Instance number of the subsystem (001b).

Bit 3*:* Setup Mode Enable

This bit is used to enable access to the POS registers. When Setup Mode Enable is 1, the CPU may access the POS registers. Setup mode should be disabled by writing this register with the value 01h (the Instance number and a 0 for Setup Mode Enable).

7.2. ISA Bus Mastership for XGA

7.2.1. ISA Bus Requirements

VESA-compliant subsystems support bus mastership. In order to run on the ISA platform this requires a 16-bit ISA architecture, which includes:

- 24-bit addressing
- 16-bit data and I/O
- 7 DMA channels
- Alternate bus master (cascade DMA) operation

7.2.2. DMA Channel Assignment

This standard allows any of the seven DMA channels called for in the 16-bit ISA bus to be made available to an ISA XGA. This standard does not require an ISA XGA to support the use of all DMA channels. The DMA Channel Readback register defines which DMA channel is used by a given XGA. It is read at XGA Index 74h.

II. VESA XGA Real Mode BIOS Specification

8. Software Goals and Objectives

By providing a common software interface to XGA graphics products, the primary objective is to enable application and system software to adapt to and exploit the wide range of features available on current and future XGAs.

Specifically, the VESA XGA BIOS attempts to address the following two main issues a) return information about the video environment to the application and b) assist the application in initializing and programming the hardware.

8.1. Video Environment Information

The VESA XGA BIOS provides several functions to return information about the video environment. These functions return system level information as well as video mode specific details. Function 00h returns general system level information, including the number of XGAs installed in the system and a system OEM identification string. Function 01h returns board dependent information for each XGA installed in the system. The function also returns a pointer to a list of the video modes supported by the particular board. Function 02h may be used by the application to obtain information about each supported video mode. Function 04h returns the current video mode for a particular XGA.

8.2. Programming Support

Due to the fact that different XGA products have different hardware implementations, application software could have great difficulty in adapting to each environment. However, since each is based on the XGA hardware architecture, differences are most common in platform specific implementation issues and video mode initialization. The rest of the architecture is kept intact.

The VESA XGA BIOS provides several functions to interface to the different XGA hardware implementations. The most important of these is Function 03h, Set XGA Video Mode. This function isolates the application from the tedious and complicated task of setting up a video mode.

8.2.1. Compatibility

A primary design objective of the VESA XGA BIOS is to preserve maximum compatibility to the standard XGA and VGA environments. In no way should the BIOS compromise compatibility or performance. Another related concern is to minimize the changes necessary to an existing VGA BIOS. RAM- as well as ROM-based implementations of the VESA XGA BIOS should be possible.

8.2.2. Scope of VESA XGA BIOS Standard

The purpose of the VESA XGA BIOS is to provide support for XGA environments. Thus, the underlying hardware architecture is assumed to be an XGA. Graphics software that drives a VESA-compatible XGA board, will perform its graphics output in generally the same way it drives a standard XGA, i.e., writing directly to an XGA style frame buffer, manipulating graphic primitives registers, directly programming the palette, etc.

8.3. Support of Existing BIOS Standards

A primary design goal with the VESA XGA BIOS is to minimize the effects on the standard VGA BIOS and the VESA VGA BIOS Extensions. Standard VGA BIOS functions should not be affected in any way. Also, the VESA XGA BIOS should peacefully coexist with the VESA VGA BIOS Extensions. This is important since it is possible that all three standards will be implemented in a single BIOS ROM or in a RAM TSR.

8.3.1. INT 10h Vector Preservation

To facilitate compatibility, all VESA compatible-XGA video subsystems will contain 4 bytes of BIOS Chain storage that will be reserved for the BIOS to store the address of the INT 10h service handler that was present before the VESA XGA BIOS hooked the interrupt. The VESA XGA BIOS will pass any INT 10h service requests that do not apply to an XGA to the routine at this address.

8.3.2. XGA Board Resident VESA XGA BIOS Address Standardization

In a system environment that contains multiple XGAs, it may be desirable to request service from each of the different XGA's BIOS. This is accomplished by selectively enabling the target board's BIOS decode and then calling the BIOS at a predetermined address. In order to provide for such a feature, the ROM-based VESA XGA BIOS uses a standard entry point address for the INT 10h VESA XGA BIOS service routines. This address is at **offset 40h** within the target board's XGA ROM aperture.

8.3.3. XGA Board Resident VGA BIOS Address Standardization

In a system environment that contains multiple XGAs (and therefore VGAs), it may be desirable to request service from each of the different VGA's BIOS. This is accomplished by selectively enabling the target board's VGA BIOS decode and then calling the VGA BIOS at a predetermined address. In order to provide for such a feature, the XGA Board-Resident VGA BIOS uses a standard entry point address for the INT 10h VGA BIOS service routines. This address is at **offset 40h** within the target board's VGA ROM space.

8.4. OEM-Defined Mode Tags

The VESA XGA BIOS does not assign specific numbers to set resolutions. It provides an interface by which the application video driver may query the XGA display adapter and determine what resolutions, pixel depths and pixel formats are supported. The action required to set a video mode

using the VESA XGA BIOS involves querying the BIOS about what mode tags the currently installed XGA's OEM has defined, then querying the BIOS to find out the resolution and pixel depth particulars for each mode tag returned, and finally, calling Function 03h (Set XGA Video Mode) with the mode tag representing the desired resolution and pixel format.

Due to the limitations of 7-bit mode numbers, VESA video mode tags are 15-bits wide. To initialize an XGA mode, its tag is passed in the BX register to VESA XGA BIOSFunction 03h (Set XGA Video Mode).

The format of VESA XGA mode tags is as follows:

D15= Reserved for Clear Memory flag

D14-10= Reserved by VESA for future expansion (= 0)

D9-0= Mode tag

If D9 == 0, this is not an XGA mode

If D9 == 1, this is an XGA mode

D8 = 0, Reserved by VESA for future expansion

Thus, VESA XGA mode tags begin at 200h.

8.5. VESA XGA BIOS

Several new BIOS calls have been defined to support XGA modes. For maximum compatibility with the standard VGA BIOS, these calls are grouped under one function number. This number is passed in the AH register to the INT 10h handler.

The designated VESA XGA extended function number is**4Eh**. This function number is presently unused in most, if not all, VGA BIOS implementations. A standard VGA BIOS performs no action when function call 4Eh is made. This standard defines subfunctions 00h through 06h. Subfunction numbers 07h through 0FFh are reserved for future use.

8.5.1. Status Information

Every function returns status information in the AX register. The format of the status word is as follows:

AL ==	4Eh:	Function is supported
AL !=	4Eh:	Function is not supported
AH ==	00h:	Function call successful
AH ==	01h:	Function call failed

Software should treat a non-zero value in the AH register as a general failure condition. In later versions of this standard, new error codes might be defined.

8.5.2. XGA Handles

It is possible to have more than one XGA installed in a system at a given time. Therefore the VESA XGA BIOS assigns a logical "handle" to each XGA currently installed. The first 16-bit handle assigned is 0000h and additional handles are assigned incrementally. Therefore, valid handles are 0 through (Number of XGAs Installed - 1). VESA XGA BIOS Functions 01h through 06h require that a valid handle be passed in the DX register.

8.5.3. Function 00h - Return XGA Environment Information

The purpose of this function is to provide information to the calling program about the general capabilities of the XGA environment. The function fills an information block structure at the address specified by the caller. The information block size is 256 bytes.

Input:

```
AH= 4Eh      ; XGA support
AL= 00h      ; Return XGA environment information
ES:DI= Pointer to buffer
```

Output:

```
AX= Status   ; If AX equals 014Eh upon return, then there is no XGA currently
             installed in the system.
```

All other registers are preserved.

The environment information block has the following structure:

```
EnvInfoBlock       struc
VESASignature      db    'VESA'       ; 4 signature bytes
VESAVersion        dw    100h         ; VESA version number
OEMStringPtr       dd    ?            ; pointer to system OEM string
EnvironmentFlag    dd    ?            ; capabilities of the system environment
NumXGAs            dw    ?            ; number of XGAs installed
Reserved           db    240 dup (?)  ; remainder of EnvInfoBlock
EnvInfoBlock       ends
```

The `VESASignature` field contains the ASCII characters 'VESA' if this is a valid block.

The `VESAVersion` is a binary field which specifies to what level of the VESA standard the VESA XGA BIOS conforms. The higher byte specifies the major version number. The lower byte specifies the minor version number. The current VESA XGA version number is 1.0. Applications written to use the features of a specific version of the VESA XGA BIOS, are **guaranteed** to work in later versions. The VESA XGA BIOS will be fully upwards compatible.

The `OEMStringPtr` is a 32-bit pointer (segment:offset) to a null terminated OEM-defined string. The string may be used to identify the system environment by system-specific display drivers. There are no restrictions on the format of the string.

The `EnvironmentFlag` field describes what general features are supported in the system environment. The bits are defined as follows:

D31-3= Reserved by VESA for future expansion

D2= Bus Mastering Capability

0 = Bus Mastering not available in this system

1 = Bus Mastering is available in this system

D1-0= System Bus Architecture ID[1]

00 = Micro Channel Architecture Bus Platform

01 = ISA Bus Platform

10 = RESERVED

11 = EISA Bus Platform

The `NumXGAs` field indicates the total number of XGAs installed in the system. The VESA XGA BIOS recognizes handle values of 0 through (`NumXGAs` minus one).

Note that the VESA XGA BIOS will set to zero all unused fields in the environment information block, always returning exactly 256 bytes. This facilitates upward compatibility with future versions of the standard, as any newly-added fields will be designed such that values of zero will indicate nominal defaults or non-implementation of optional features. (For example, a field containing a bit-mask of extended capabilities would reflect the absence of all such capabilities.)

8.5.4. Function 01h - Return XGA Subsystem Information

The purpose of this function is to provide information to the calling program about the capabilities of a particular XGA in the system. The function fills an information block structure at the address specified by the caller. The information block size is 256 bytes. To obtain information for all of the XGAs present, the application should load the DX register with 0, and then call this function `NumXGAs` times, incrementing DX after each call.

Input:

```
AH= 4Eh      ; XGA support
AL= 01h      ; Return XGA subsystem information
DX= XGA Handle
ES:DI= Pointer to buffer
```

[1] The VESA XGA BIOS will determine the system's bus mastering capabilities during the POST, with diagnostic test routines.

Output:

```
AX= Status   ; If AX equals 014Eh upon return, then there is no XGA currently
             installed in the system.
```

All other registers are preserved.

The adapter information block has the following structure:

```
XgaInfoBlock       struc
OEMStringPtr       dd    ?     ; pointer to board OEM string
CapabilitiesFlag   dd    ?     ; capabilities of this XGA board
XGARomAddr         dd    ?     ; pointer to XGA's 8KB ROM Aperture
XGAMemRegAddr      dd    ?     ; pointer to XGA's memory mapped registers
XGAIORegAddr       dw    ?     ; base address of XGA's I/O regist  ers
VidMemBaseAddr     dd    ?     ; base address of XGA's physical video memory
XGA4MBAddr         dd    ?     ; pointer to XGA's 4MB Aperture
XGA1MBAddr         dd    ?     ; pointer to XGA's 1MB Aperture
XGA64KBAddr        dd    ?     ; pointer to XGA's 64KB Aperture
OemAptrAddr        dd    ?     ; pointer to XGA's OEM defined Aperture
OemAptrSize        dw    ?     ; size of OEM defined Aperture, in 64KB units
VideoModePtr       dd    ?     ; pointer to supported XGA modes
TotalMemory        dw    ?     ; number of 64KB memory blocks on board
XGAManId           dd    ?     ; XGA Manufacturer's ID
Reserved           db    194 dup (? ) ; remainder of XgaInfoBlock
XgaInfoBlock       ends
```

The `OEMStringPtr` is a 32-bit pointer (segment:offset) to a null terminated OEM-defined string. The string may be used to identify the XGA board to hardware-specific display drivers. There are no restrictions on the format of the string.

The `CapabilitiesFlag` field describes what features are supported on the XGA**board**. The bits are defined as follows:

D31-8= Reserved by VESA for future expansion

D7= DMA Channel Status[2]

0 = Disabled

1 = Enabled

D6-4= DMA Channel assigned for acquiring bus mastership[2]

D3-2= Reserved by VESA for future expansion

[2] The information in this field is valid for ISA bus board architectures only.

D1-0= Board Bus Architecture ID

00 = Micro Channel Architecture Bus Interface

01 = ISA Bus Interface

10 = RESERVED

11 = EISA Bus Interface

The `XGARomAddr` is a 32-bit pointer (segment:offset) that specifies the address at which this XGA's 8KB ROM resides. If this value is zero, then the 8KB ROM is not enabled.

The `XGAMemRegAddr` is a segment:offset pair that specifies the address at which this XGA's memory mapped registers reside.

The `XGAIORegAddr` is a word that specifies the base address of this XGA's I/O registers.

The `VidMemBaseAddr` is a 32-bit **absolute address** that specifies the physical address of this XGA's video memory.

The `XGA4MBAddr` is a 32-bit **absolute address** that specifies where this XGA's 4MB Aperture resides. If this value is zero, then the 4MB Aperture is not enabled.

The `XGA1MBAddr` is a 32-bit **absolute address** that specifies where this XGA's 1MB Aperture resides. If this value is zero, then the 1MB Aperture is not enabled.

The `XGA64KBAddr` is a segment:offset that specifies the address at which this XGA's 64KB Aperture resides. If this value is zero, then the 64KB Aperture is not enabled.

The `OemAptrAddr` is a 32-bit **absolute address** that specifies where this XGA's OEM defined Aperture resides. If this value is zero, then an OEM Aperture is not available. This field is provided to allow for the definition of apertures larger than 4MB.

The `OemAptrSize` is a word that specifies the size of the OEM defined aperture, in 64KB units.

The `VideoModePtr` points to a list of mode tags representing all the XGA modes supported by the current VESA XGA BIOS. Each mode tag occupies one word (16 bits). The list of mode tags is terminated by a -1 (0FFFFh). Please refer to the OEM-Defined Mode Tags section for a complete description of VESA XGA mode tags. The pointer could point into either ROM or RAM, depending on the specific implementation. Either the list would be a static string stored in ROM or the list would be generated at run-time in the information block (see above) in RAM. It is the application's responsibility to verify the current availability of any mode returned by this function through the Return XGA Mode Information (Function 02h) call. Some of the returned modes may not be available due to the video board's current memory and monitor configuration.

The `TotalMemory` field indicates the amount of memory installed on the XGA board. Its value represents the number of 64KB blocks of memory currently installed.

The `XGAManId` field indicates the content of POS Data 4, Index 1 (byte 0) and Index 2 (byte 1) and XGA Index 75h (byte 2). Byte 3 is undefined and reserved by VESA for future expansion.

Note that the VESA XGA BIOS will set to zero all unused fields in the adapter information block, always returning exactly 256 bytes. This facilitates upward compatibility with future versions of the standard, as any newly-added fields will be designed such that values of zero will indicate nominal defaults or non-implementation of optional features. (For example, a field containing a bit-mask of extended capabilities would reflect the absence of all such capabilities.)

8.5.5. Function 02h - Return XGA Mode Information

This function returns information about a specific XGA video mode, on a specific XGA, that was returned by Function 01h. The function fills a mode information block structure at the address specified by the caller. The mode information block size is 256 bytes.

Input:

```
AH= 4Eh      ; XGA support
AL= 02h      ; Return XGA mode information
CX= OEM video mode tag
DX= XGA Handle
ES:DI= Pointer to 256 byte buffer
```

Output:

```
AX= Status
```

All other registers are preserved.

The mode information block has the following structure:

```
ModeInfoBlock           struc
ModeAttributes          dw    ?     ; mode attributes
BytesPerScanLine        dw    ?     ; bytes per scan line
XResolution             dw    ?     ; horizontal resolution
YResolution             dw    ?     ; vertical resolution
XCharSize               db    ?     ; character cell width
YCharSize               db    ?     ; character cell height
NumberOfPlanes          db    ?     ; number of memory planes
BitsPerPixel            db    ?     ; bits per pixel
MemoryModel             db    ?     ; memory model type
NumberOfImagePages      db    ?     ; number of images
RedMaskSize             db    ?     ; number of red bits in a single pixel's
                                      data
RedFieldPosition        db    ?     ; position of red bits in a single pixel's
                                      data
GreenMaskSize           db    ?     ; number of green bits in a single pixel's
                                      data
```

```
GreenFieldPosition      db    ?      ; position of green bits in a single
                                     pixel's data
BlueMaskSize            db    ?      ; number of blue bits in a single pixel's
                                     data
BlueFieldPosition       db    ?      ; position of blue bits in a single
                                     pixel's data
ReservedMaskSize        db    ?      ; number of reserved bits in a single
                                     pixel's data
ReservedFieldPosition   db    ?      ; position of reserved bits in a single
                                     pixel's data
Reserved                db    234 dup (?)  ; remainder of ModeInfoBlock
ModeInfoBlock           ends
```

The `ModeAttributes` field describes certain important characteristics of the video mode. Bit D0 specifies whether this mode can be initialized in the present video configuration. This bit can be used to block access to a video mode if it requires a certain monitor type and that this monitor is presently not connected. Bit D2 indicates whether the BIOS has support for output functions like TTY output, scroll, pixel output, etc., in this mode.

The field is defined as follows:

D15-5= Reserved by VESA for future expansion

D4= Mode type

0 = Text mode (VGA registers active, XGA registers inactive)

1 = Graphics mode (VGA registers inactive, XGA registers active)

D3= Reserved by VESA for future expansion

D2= Output functions supported by BIOS

0 = Output functions not supported by BIOS

1 = Output functions supported by BIOS

D1= Reserved by VESA for future expansion

D0= Mode supported in current hardware configuration

0 = Mode not supported in present configuration

1 = Mode supported in present configuration

The `BytesPerScanline` field specifies how many bytes are contained within each logical scanline. The logical scanline could be equal to or larger than the displayed scanline.

The `XResolution` and `YResolution` specify the width and height of the video mode. In graphics modes, this resolution is in units of pixels. In text modes this resolution is in units of characters. Note that text mode resolutions, in units of pixels, can be obtained by multiplying `XResolution` and `YResolution` by the cell width and height.

The `XCharSize` and `YCharSize` specify the size of the character cell in pixels.

The `NumberOfPlanes` field specifies the number of bit planes available to software in that mode. For XGA packed pixel modes, the field is set to one.

The `BitsPerPixel` field specifies the total number of bits that define the color of one pixel. The number of bits per pixel per plane can normally be derived by dividing the `BitsPerPixel` field by the `NumberOfPlanes` field.

The `MemoryModel` field specifies the general type of memory organization used in this mode. The following models have been defined:

00h=	Text mode
01h=	CGA graphics
02h=	Hercules graphics
03h=	4-plane planar
04h=	Packed pixel
05h=	Non-chain 4, 256 color
06h=	Direct Color
07h=	YUV-24
08h-0FFh=	Reserved by VESA for future expansion

The `NumberOfImagePages` field specifies the number of complete display images (minus 1) that will fit into the XGA's memory, at one time, in this mode. The application may load more than one image into the XGA's memory if this field is non-zero and flip the display between.

The remaining fields in the `ModeInfoBlock` contain valid data only when the `MemoryModel` equals 06h (Direct Color) or 07h (YUV-24).

The `RedMaskSize` specifies the number of bits in a single pixel's data field that contain the Red component information, when in a Direct Color mode. When in a YUV mode, this field contains the number of bits used by the V component. (e.g.., in the standard 640 x 480 x 65536 XGA mode, this field would contain a five.)

The `RedFieldPosition` specifies the location of the least significant bit of the Red component within a single pixel's data field. When in a YUV mode, this field contains the bit location of the first bit of the V component. (e.g.., in the standard 640 x 480 x 65536 XGA mode, this field would contain a 0Bh.)

The `GreenMaskSize` specifies the number of bits in a single pixel's data field that contain the Green component information, when in a Direct Color mode. When in a YUV mode, this field contains the number of bits used by the Y component. (e.g.., in the standard 640 x 480 x 65536 XGA mode, this field would contain a six.)

The `GreenFieldPosition` specifies the location of the least significant bit of the Green component within a single pixel's data field. When in a YUV mode, this field contains the bit location of the first bit of the Y component. (e.g.., in the standard 640 x 480 x 65536 XGA mode, this field would contain a 05h.)

The `BlueMaskSize` specifies the number of bits in a single pixel's data field that contain the Blue component information, when in a Direct Color mode. When in a YUV mode, this field contains the number of bits used by the U component. (e.g.., in the standard 640 x 480 x 65536 XGA mode, this field would contain a five.)

The `BlueFieldPosition` specifies the location of the least significant bit of the Blue component within a single pixel's data field. When in a YUV mode, this field contains the bit location of the first bit of the U component. (e.g.., in the standard 640 x 480 x 65536 XGA mode, this field would contain a 00h.)

The `ReservedMaskSize` specifies the number of bits in a single pixel's data field that do not contain color component information. (e.g.., in a 32-bit per pixel, 16.7 million color mode, this field might contain an 08h.)

The `ReservedFieldPosition` specifies the location of the least significant bit of the field that does not contain color component information, within a single pixel's data field. (e.g.., in a 32-bit per pixel, 16.7 million color mode, this field might contain an 18h.)

Note that the VESA XGA BIOS will set to zero all unused fields in the mode information block, always returning exactly 256 bytes. This facilitates upward compatibility with future versions of the standard, as any newly-added fields will be designed such that values of zero will indicate nominal defaults or non-implementation of optional features. (For example, a field containing a bit-mask of extended capabilities would reflect the absence of all such capabilities.)

8.5.6. Function 03h - Set XGA Video Mode

This function initializes a video mode. The BX register contains the OEM tag for the mode to set. The format of VESA XGA mode tags is described in the OEM-Defined Mode Tags section. The DX register contains the virtual handle of the XGA to which the mode set request applies. If the mode cannot be set, the BIOS will leave the video environment unchanged and return a failure error code. This function should be used to set all graphics modes on the desired XGA adapter, including VGA modes. It **must** be used to switch between XGA and VGA modes on the given adapter.

Input:

```
AH= 4Eh      ; XGA support
AL= 03h      ; Set XGA video mode
BX= OEM Video mode tag
      D15= Clear memory flag
                  0 = Clear video memory
                  1 = Don't clear video memory
      D14-0= Video mode
CX= Other command flags
      D15-1= Reserved (0)
      D0= Feature connector flag
```

```
                0 = Set Feature connector to default state [3]
                1 = Do not change state of feature connector
DX= XGA Handle
```

Output:

```
AX= Status
```

All other registers are preserved.

8.5.7. Function 04h - Return Current Video Mode

This function returns the OEM tag for the current video mode in BX. The format of VESA XGA video mode tags is described in the OEM-Defined Mode Tags section.

Input:

```
AH= 4Eh      ; XGA support
AL= 04h      ; Return current video mode
DX= XGA Handle
```

Output:

```
AX= Status
BX= OEM mode tag or VGA mode number for current video mode
```

All other registers are preserved.

8.5.8. Function 05h - Set Feature Connector State

This function enables or disables the transmission of video data through the XGA board's feature connector and sets the direction in which the data is transferred.

Input:

```
AH= 4Eh      ; XGA support
AL= 05h      ; Set feature connector state
BX= Feature Connector state
      D15-2= Reserved (0)
      D1= Feature connector Input/Output control
            0 = Set feature connector to input mode
```

---

[3] In single monitor systems utilizing a VGA that is not on the XGA board, the VGA's output will usually be fed to the XGA through the feature connector, for display on the XGA's monitor. In default mode in this configuration, the VESA XGA BIOS will disable the feature connector when switching to an XGA mode and enable it when switching to a VGA mode. This action can be overridden by setting bit D0 of the CX register when making the Set XGA Video Mode BIOS call (see Function 05h - Enable Feature Connector).

```
            1 = Set feature connector to output mode
      D0= Feature connector enable
            0 = Disable feature connector
            1 = Enable feature connector
DX= XGA Handle
```

Output:

```
AX= Status
```

All other registers are preserved.

8.5.9. Function 06h - Get Feature Connector State

This function returns the current state of the feature connector.

Input:

```
AH= 4Eh      ; XGA support
AL= 06h      ; Get feature connector state
DX= XGA Handle
```

Output:

```
AX= Status
BX=Feature Connector state
      D15-2= Reserved (0)
      D1= Feature connector Input/Output state
            0 = Feature connector is in input mode
            1 = Feature connector is in output mode
      D0= Feature connector enable
            0 = Feature connector is disabled
            1 = Feature connector is enabled
```

All other registers are preserved.

III. Appendices

A. Programming Considerations Appendix

The following sequence illustrates how an application interfaces to the VESA XGA BIOS. The hypothetical application is VESA XGA-aware and calls the VESA XGA BIOS functions.

The application first allocates a 256-byte buffer. This buffer is used by the VESA XGA BIOS to return information about the video environment. Some applications will statically allocate this buffer, while others will use system calls to temporarily obtain buffer space.

The application then calls VESA XGA BIOS Function 00h (Return XGA Environment Information). If the AX register does not contain 004Eh on return from the function call, the application can determine that either the VESA XGA BIOS or an XGA is not present and handle such a situation.

If no error code is returned in AX, the function call was successful. The buffer has been zeroed and then filled by the VESA XGA BIOS with various information, including the number of XGAs installed in the system. The application can verify that it is a valid VESA block by identifying the characters 'VESA' in the beginning of the block. The application can inspect the `VESAVersion` field to determine whether the VESA XGA BIOS has sufficient functionality. The application may use the `OEMStringPtr` to determine system OEM-specific information.

The application then creates a new 256-byte buffer for each XGA in the system. Then the application calls VESA XGA BIOS Function 01h (Return XGA Subsystem Information) for each XGA present. If the AX register does not contain 004Eh on return from the function call, the application can determine that either the VESA XGA BIOS or an XGA is not present and handle such a situation.

If no error code is returned in AX, the function call was successful. The buffer has been filled by the VESA XGA BIOS with information about the adapter indicated by the virtual handle passed in the DX register, including its access address. The application may use the `OEMStringPtr` to determine board OEM-specific information.

Finally, the application can obtain a list of the supported XGA modes, by using the `VideoModePtr` . This field points to a list of OEM defined tags, representing the video modes supported by the specified XGA adapter.

The application then creates and clears new buffers and calls VESA XGA BIOS Function 02h (Return XGA Mode Information) to obtain information about the supported video modes on each XGA. Stepping through the tags pointed to by the `VideoModePtr` , obtained in the previous step, the application calls this function with a new tag until a suitable video mode is found. If no appropriate video mode is found, it is up to the application to handle the situation.

The Return XGA Mode Information function fills a buffer specified by the application with information describing the features of the video mode. The data block contains all the information an application needs to take advantage of the video mode.

To verify that the mode is supported, the application inspects bit D0 of the `ModeAttributes` field. If D0 is cleared, then the mode is not supported by the hardware. This might happen if a specific mode requires a certain type of monitor, but that monitor is not present.

After the application has selected a video mode, the next step is to initialize the mode. However, the application should save the present video mode. When the application exits, this mode should be restored. To obtain the present video mode, the VESA XGA BIOS Function 04h (Get XGA Video Mode) is used. If a non-VESA (standard VGA) mode is in effect, only the lower byte in the mode number is filled. The upper byte is cleared.

To initialize the video mode, the application would use VESA XGA BIOS Function 03h (Set XGA Video Mode). From this point on the application has full access to the XGA hardware and video memory.

When the application terminates, it should restore the prior video mode. The prior video mode, obtained above, could be either a standard VGA mode or an OEM-specific mode. It reinitializes the video mode by calling VESA XGA BIOSFunction 03h (Set XGA Video Mode). The application would then exit.

B. Adapter Identification Appendix

In order to differentiate between the original IBM XGA and VESA compatible systems, the following mechanism is suggested:

1. Check the POS ID to determine if the POS ID is within the 1K set of POS ID numbers. If it is, then the product is a VESA compliant XGA. If not, then step (2).

2. Check for POS ID=8FD8h through 8FDBh or 8FD0h through 8FD3h. If not found, then the product is not XGA. If found, then step (3).

3. Determine if indexing of POS registers is supported. This could be determined by writing multiple values to the POS Index registers (POS 6 and POS 7), and verifying that the values read back correctly. Note that values larger than 3 are not guaranteed to read back correctly. If indexing is not supported, the product is an original IBM XGA or 100% compatible. If indexing is supported, then step (4).

4. Read Manufacturer ID and save.

5. Check for revision number at POS 4, Index 2. If zero (0), no revision numbering is supported, otherwise save.

6. Check for vendor subsystem ID at XGA Index 75h. If zero (0), no vendor ID is supported, otherwise save.

C. Multiple Video Subsystems Appendix

XGA/XGA Combination

In the case of two coexisting XGA subsystems, the standard IBM mechanism using the POS registers and subsystem-specific Instance numbers will keep the I/O locations from conflicting. However, VESA XGA subsystems will usually contain a VGA BIOS within them. This means that some mechanism will be required to allow coexistence of the VGA BIOS ROMs. The register bits in the VGA BIOS Configuration register allow coexistence of multiple VESA compatible BIOSes. The only requirement is that a subsystem disable its own VGA BIOS prior to enabling the next subsystem. This can be achieved by locating the executable code which performs the switch in the 8KB relocatable ROM area.

Multiple XGAs from multiple manufacturers will create special problems. This issue will not be addressed in this standard.

XGA/VGA Combination

There are three cases of coexistence which must be handled by an XGA subsystem, as follows:

- XGA Adapter in a system with a VGA using an E000h BIOS
- XGA Adapter in a system with a VGA using a C000h BIOS
- XGA subsystem coexisting with another XVGA (an XGA subsystem in a VGA compatibility mode) subsystem

XGA/E000 VGA Combination

If the VGA subsystem is a properly constructed mother board subsystem (using 3C3h for sleep and an E000h segment VGA BIOS), then an XGA adapter (using 46E8h for sleep control and a C000h segment VGA BIOS) will coexist as a by-product of VGA compatibility.

If the VGA subsystem uses 46E8h for sleep, or does not respond to sleep, then the XGA subsystem's VGA compatibility may be disabled, and the pass-through mechanism used from the existing VGA.

XGA/C000 VGA Combination

If the VGA subsystem is logically an adapter design (46E8h sleep, C000h VGA BIOS), regardless of whether it is physically on the mother board, there are two solutions:

- If the system in which the XGA Adapter is installed allows a device on the system bus to decode the E000h segment, then the XGA subsystem can be configured as a mother board subsystem (3C3h sleep and E000h VGA BIOS). This will allow the two subsystems to coexist. However, since most system BIOSes will not recognize an E000h VGA BIOS as a video BIOS, the C000h VGA BIOS will be POSTed first (rather than second) and will be in control following POST.

- If this is not possible, then a pass-through mechanism from the VGA could be used. This would allow the VGA to be used for VGA modes, with the XGA used solely for Extended Graphics mode. This would be compatible for applications that do not take advantage of 132 Column Text mode and which do not make assumptions about the overlap between the VGA and XGA video buffer.

XGA/EGA Combination

The addressing issues are similar to those associated with XGA/VGA, except that there is no POS mechanism or equivalent available, and most EGAs cannot be software disabled. This issue will not be addressed in this standard.

XGA/CGA or MDA Combination

The issues are the same as a VGA with CGA/Mono and will not be addressed in this standard.

D. Obtaining VESA Identification Numbers Appendix

To obtain a VESA identification number, contact the Video Electronics Standards Association at 2150 North First Street, Suite 440, San Jose, CA 95131-2020 or phone (408) 435-0333.