VAXstation 4000 Model 60 3D Graphics Options

Technical Summary

EK-SCP8P-TS

Digital Equipment Corporation
Maynard, Massachusetts

First Printing, March 1992

Contents

Tables

1 Introduction

The VAXstation 4000 Model 60 workstation offers all the advantages of an integrated computing environment based on Digital's VAX architecture. The VAXstation 4000 Model 60 is compact desktop system that offers the following features:

- Desktop computing.
- Industry-standard personal productivity tools.
- Transparent access to local and distributed applications and resources.
- Timesharing.
- Up to 104 Megabytes of memory.
- Standard Ethernet and ThinWire Ethernet ports.
- A variety of options for adding storage capacity, communications, memory, and enhanced graphics.

Graphics options include:

- Low-resolution, low-cost 8-plane 2D graphics.
- High-resolution, low-cost 8-plane 2D graphics.
- High-resolution, dual screen 2D graphics.
- High-resolution, 8-plane 3D graphics (**SPXg**).
- High-resolution, 24-plane 3D graphics (**SPXgt**).

The SPXg and SPXgt are the new generation of high-resolution (1280×1024) 3D graphics accelerators for the VAX/VMS environment. These graphics options are available on the VAXstation 4000 Model 60 either as factory-integrated systems or as upgrades to existing systems.

1.1 SPXg and SPXgt Overview

The SPXg and SPXgt accelerators support 2D and 3D acceleration for graphics primitives such as vectors and polygons, animation, depth cueing, and multiple color maps. They contain hardware acceleration to support major industry standard graphics interfaces.

The 8-plane SPXg (**PV61G-BA** option) is suited for design automation, molecular modeling, architectural design, and pseudo-color imaging applications. It supports full double buffering and 16-bit Z buffering.

The higher performance 24-plane SPXgt (**PV61G-AA** option) is suited for 3D modeling, true-color imaging, and visualization applications. It also supports full double buffering along with 24-bit Z buffering.

Typical applications supported by the SPXg and SPXgt include:

- Mechanical 3D CAD/CAE/CAM applications such as solid modeling, 3D design, finite element modeling, and numerical control (N/C) machining
- Geographic information systems (GIS) and mapping applications such as seismic modeling and interpretation and digital terrain modeling.
- True-color imaging applications such as medical diagnostics, scanning, and remote sensing (SPXgt).

Advanced Graphics Terms

Depth Cueing is the gradual variation of color across a primitive to indicate depth in the 3D scene.

Double Buffering is the process of rendering a picture into an off-screen portion of the frame buffer, then displaying the complete picture all at once. The completed picture is displayed either by rapidly copying the complete raster image to the on-screen portion of the frame buffer or by making the off-screen portion the on-screen portion through switching. Double buffering is relevant when the time to draw a picture is noticeable, as is frequently the case with complex pictures. Some applications, such as animation, are more effective if the viewer does not see the drawing in progress, but only completed frames.

Geometry Processing maps the 3D lines and polygons to the plane of the 2D surface. This processing involves a projection from 3D space to 2D space, as well as the transformation of coordinates from the device-independent 3D coordinates of the application to the device-specific 2D coordinates that locate the image elements on the display surface. This processing consists mostly of matrix multiplication and requires floating-point arithmetic. Geometry processing may also include *clipping*, which is the elimination of primitives and parts of primitives that lie outside the defined region of interest.

Scan conversion is the process of determining which pixels must be set and what values they must be given in order to draw a given 2D primitive (for example, lines and polygons).

Smooth Shading is the gradual variation of color across primitives to make pictures more realistic by hiding the facets when polygonal meshes are used to approximate smooth surfaces. Such color variation is also used for data mapping in data visualization applications.

Z Buffering attaches a depth value to every pixel as a means of determining which parts of the objects in a scene must be discarded from processing because they would be hidden from view by other opaque objects (hidden surface removal).

2 Software and Architecture

2.1 Graphical User Interfaces and DECwindows

Virtually all workstation and personal computer users are familiar with window systems for running several applications at the same time. Each window acts as a virtual graphics terminal for the application's output.

Because the window system must be common to all the concurrent applications, it is part of the system software, placed between the operating system and the applications. The *window manager* is central to the window system. The window manager resolves contention for the display area among the applications and gives the user some control over the window layout. It is also concerned with associating keyboard and pointing device input with a particular application and window.

Window systems are also platforms for *graphical user interfaces* (GUIs), in which interactive users manipulate screen objects — menus, dialogue boxes, scroll bars, buttons — by pointing and clicking with a mouse and cursor. Using the same GUI (that is, a common set of interactive screen objects with a common means of manipulation) across different applications makes each application easier to learn and use.

2.1.1 DECwindows

DECwindows is Digital's enhanced implementation of the *X window system.*

X Window System

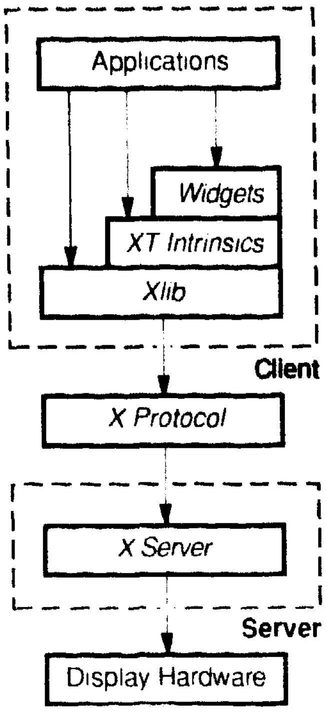
The *X window system* (also called *X* or *X11*) is an industry-standard networked window system developed at MIT with financial and technical support from an industry consortium of which Digital is a leading member. It is a networked window system of the *client/server* type: The server being the software that controls the interactive graphics workstation — the display and associated interactive input devices; and the client being the application that carries on interactive input/output through requests to the server. The client and server can run on the same processor, or the server can reside on a remote host and communicate with the client over a network.

The X client and server (Figure 1) communicate through a precise protocol, the *X protocol*, which is designed to communicate simple graphics and window control information. In case the client and server are on different hosts, the X protocol is implemented over the underlying network protocol. The X protocol is independent of the architecture and operating system and supports interoperability for transparent network graphics among diverse systems.

Xlib

In the X software client/server architecture, the server is graphical device dependent; the client is device independent. The basic application programming interface on the client side is *Xlib*, a library of almost 400 routines for 2D graphical primitives, window management functions, and input event management. Xlib is the application programmer's lowest level of access to the X window system.

Figure 1 X Window System Architecture



WMO_SPXGGT_029

Xtk

In the X window system, the *Xtk* tool kit is a higher-level programming interface that gives the programmer the tools to make and manage interactive screen objects, such as buttons and sliders. Such objects are called *widgets* in X. The Xtk includes a set of widgets, called the *intrinsics*, but does not define "look-and-feel" standards. It can be used to implement GUIs with a particular look and feel, such as *OSF/Motif* (see Section 2.1.2).

DECwindows enhancements to the standard X window system include:

- Faster algorithms.
- International keyboard support.
- Better security features.
- Additional language bindings.

On VAXstation 4000 Model 60 workstations, DECwindows is an exceptionally fast and robust implementation of X. The version of DECwindows shipped with VMS Version 5.5 embodies the X11R4 version of the X windows system.

2.1.2 OSF/Motif

OSF/Motif is an industry-standard GUI adopted by the Open Software Foundation and based largely on *XUI*, the DECwindows GUI developed by Digital. OSF/Motif includes a tool kit for constructing interactive widgets, a *user interface language* (UIL) for building applications with Motif user interfaces, and a style guide. By adhering to the style guide, an application programmer can create applications having a "standard" look and feel, making them easy to learn and use by anyone familiar with other Motif-based applications.

The VMS Version 5.5 operating system for VAXstations supports both OSF/Motif and XUI.

2.1.3 X Protocol Extensions: PEX, Display PostScript, and Xv

The standard X protocol does not include semantics for addressing all the conceivable functionality that a graphics server may provide. When new graphical functionality is added to the server, it becomes necessary to extend the protocol and server code to use the new functionality in the X window system context. The X window system provides a standard way for making such extensions.

PEX

For example, the original X protocol graphical content is limited to 2D graphics. However, some graphics subsystems, such as the SPXg and SPXgt, provide hardware and firmware for 3D graphics processing. The graphics subsystem is clearly on the server side of the X protocol. To use such 3D hardware for making pictures in X windows, the X consortium has defined *PEX* as a standard extension to the X protocol. PEX is an acronym for "PHIGS extension to X" and reflects the fact that PEX is based largely on *PHIGS* (see Section 2.2.2). However, PEX is only a protocol for communicating graphics information between client applications and 3D graphics servers; it can also be used with other graphics *application programming interfaces* (APIs).

Implementing a PEX server on 3D graphics hardware makes the hardware available for network-transparent, inter-vendor, distributed applications. Digital is a leading member of the PEX development and standardization effort, and produced the industry's first PEX-based workstation in 1989. The Digital PEX server is bundled with the SPXg and SPXgt.

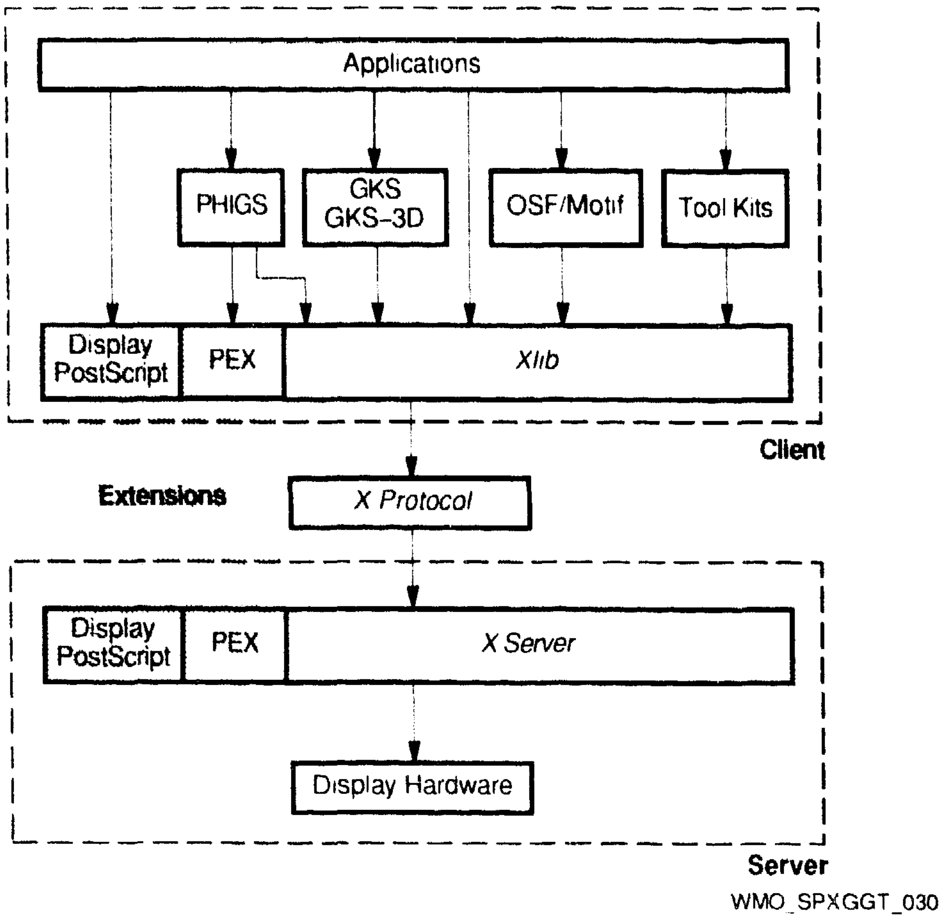
Digital has developed a second extension to the X protocol to support *Display PostScript* graphics (see Section 2.2.1). The X consortium has adopted this extension as a standard.

Digital has worked with MIT to develop another X protocol extension, called *Xv*, to support DECvideo multimedia options. Digital is actively working to have the X consortium adopt Xv as standard.

2.2 Graphics Programming Interfaces

The Xlib API (Section 2.1.1) provides drawing primitives only for simple 2D figures such as lines, arcs, polygons, and text. To support 2D and 3D application development, programmers need much richer and higher-level APIs for defining, manipulating, editing, displaying, and storing complex graphical objects. Digital offers several higher-level graphics APIs for 3D graphics as well as for 2D applications that need more graphics primitives and richer graphical data structuring capabilities. For display input/output, all of these higher-level APIs work through the X window system and its extensions. Figure 2 shows the architecture of the DECwindows graphics programming environment.

Figure 2 DECwindows Graphics Programming Architecture



2.2.1 Display PostScript

PostScript, the page description language developed by Adobe Systems, Inc., has become the industry-standard means to access the advanced graphical and typographical capabilities of laser printers. Display PostScript is a language adapted from PostScript for programming interactive raster displays. Digital's implementation of the Display PostScript interpreter is licensed directly from Adobe and is complete and fully compatible.

Digital has implemented the Display PostScript interpreter as part of the X server and extended the X protocol to access it (Section 2.1.3). Server-side implementation of the interpreter has certain advantages concerning performance and compatibility. As PostScript is frequently implemented by a PostScript engine within a laser printer, it is logical to anticipate the advent of Display PostScript accelerators in workstations, which will then *require* server-side implementation and protocol extension. By taking the lead in developing server-side implementation and standardization of the protocol extension, Digital ensures future compatibility of Display PostScript applications written today.

2.2.2 DEC PHIGS and PEXlib

PHIGS is the preferred ANSI/ISO standard API for 3D device-independent graphics. *PHIGS PLUS* is a proposed extension to the standard. It adds advanced surface primitives and lighting-and-shading rendering capability to the original PHIGS. When the extension is adopted, the term "PHIGS" will be understood to include PHIGS PLUS.

Standard PHIGS is a *structure-oriented* (as opposed to an *immediate-mode*) API. Through calls to the API library functions, the application builds data structures that represent the picture to be drawn. These structures are retained and stored by the PHIGS run-time system and can be redisplayed with subsequent calls. PHIGS structures permit hierarchical organization and structure editing, two important features that help to distinguish PHIGS from *GKS* and *GKS–3D* (Section 2.2.3).

DEC PHIGS

DEC PHIGS is a complete PHIGS implementation, including PHIGS PLUS and several other extensions. The PHIGS PLUS features of DEC PHIGS include lighting, shading, depth cueing, and non-rational B-spline (NURBS) curves and surfaces. The additional extensions include circle and arc primitives, an immediate-mode feature, and certain other extensions that, while not part of the standard, are advocated and informally encouraged by a group of CAD/CAM software companies. DEC PHIGS offers C and FORTRAN language bindings, and also supports PHIGS programming in several other languages.

DEC PHIGS is well integrated with DECwindows and provides the programmer with several choices for managing the interaction between PHIGS and X and for mixing PHIGS and Xlib calls. DEC PHIGS can also generate PEX protocol requests to make use of hardware graphics acceleration. Alternatively, DEC PHIGS can perform 3D graphics processing in software to generate drawing requests through the ordinary X protocol. A few of the more compute-intensive capabilities of PHIGS PLUS are provided only when supported by 3D hardware acceleration.

The PHIGS run-time support is bundled with the SPXg and SPXgt.

PEXlib

PEXlib is a 3D graphics API that allows programmers to access the PEX protocol at a slightly lower level than PHIGS. PEXlib has no status as a standard, but it is of interest to programmers who want to avoid some of the overhead of PHIGS and have a little more direct control of the PEX server. PEXlib shipped for the first time with DEC PHIGS 2.3.

2.2.3 GKS and GKS–3D

GKS is an ISO standard for a 2D graphics API. It was originally developed in Europe and predates PHIGS. GKS–3D is an extension of the standard to include 3D geometry; however, it does not include the advanced primitives and lighting-based features of PHIGS PLUS.

While GKS and GKS–3D have a form of retained structures (called *segments*), they lack the hierarchical structuring and structure-editing features of PHIGS, as well as the lighting and shading capabilities of PHIGS PLUS. Strictly 2D graphics applications sometimes perform better when implemented in GKS rather than PHIGS, because PHIGS is intrinsically 3D and carries some 3D overhead, even when used for 2D applications. Similarly, 2D graphics performance may be somewhat better on a 2D system than it is on a system with an SPXg or SPXgt 3D graphics accelerator installed.

Digital's implementation of DEC GKS–3D supports full level-2c functionality of the standard and is upward compatible with the DEC GKS implementation.

2.3 XMAP Video Architecture

Digital designed the *XMAP* video architecture to meet the display requirements of multi-window workstations. A primary XMAP function is to support the simultaneous display of multiple independent applications on workstations and windowing terminals.

The XMAP architecture provides virtual mapping of the available physical display resources, such as the frame buffer, color maps, and other separate requirements. The XMAP architecture provides each application with the capability for an individually tailored video display environment, independent of other simultaneous applications, on a common set of hardware resources shared by all applications. The XMAP architecture alleviates hardware resource limitations that prevent display process independence.

The XMAP architecture contains many components that support the simultaneous display of independent applications, including individual window control mechanisms for:

- Window priority and display.
- Frame buffer plane position selection.
- Number of frame buffer planes.
- Double buffering and animation.
- Cursor generation, control, and clipping.
- Multiple overlay and underlay modes
- Multiple variable-sized color maps.
- True color, pseudo color, gray, and bitonal visual types.
- Digital and analog self-test modes.

Any specific XMAP architecture implementation can support an arbitrary number of independent or overlapped virtual color tables within a single physical color map. The XMAP architecture provides simultaneous support for multiple visual types by interpreting frame buffer data on a per-window basis. It provides more efficient frame buffer use; for example, pseudo-color applications need to load only one set of the RGB planes.

The XMAP architecture provides per-window video display control that can be used to integrate synthetic graphics, live video, imaging, and other multimedia data sources within a single workstation, PC, or X terminal display.

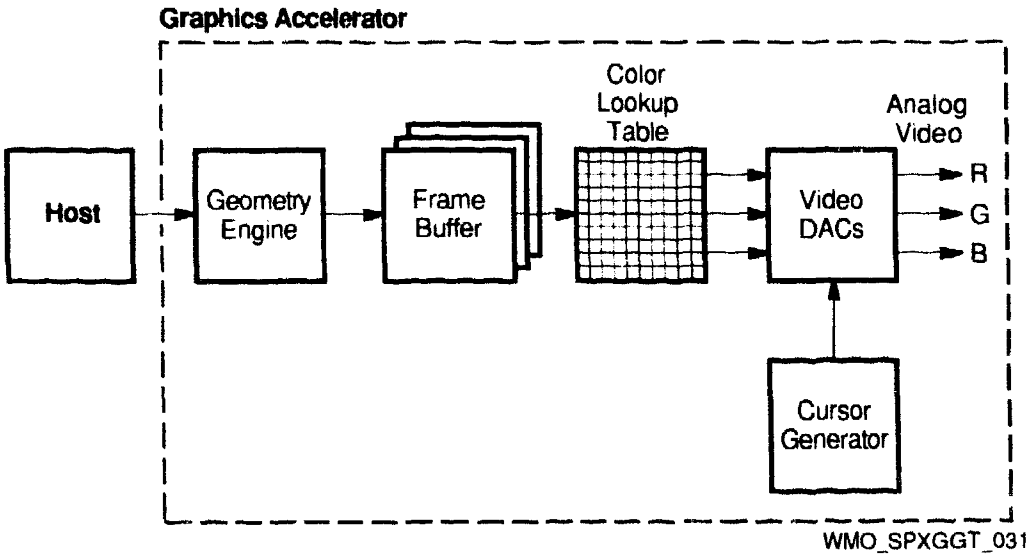
Both the SPXg and SPXgt support components of the XMAP architecture.

3 Graphics Options

3.1 Overview

Raster graphics systems are built around a *frame buffer*, which is a memory containing a pixel-level description of the displayed image and is used to refresh the raster display. Refreshing the display involves clocking-out the frame buffer data in synchronism with the scan cycle of a monitor, using the pixel values to produce the analog signals that drive the monitor's color guns. The frame buffer is frequently implemented in a special dual-ported RAM, called a video RAM (VRAM), that allows the graphics hardware to draw the picture by writing to the frame buffer without interfering with the display refresh cycle. Figure 3 shows a generic raster graphics system.

Figure 3 Generic Raster Graphics System



The number of bits-per-pixel in the frame buffer is called its *depth*, and is usually expressed as the number of *bit planes* or *planes*. The maximum number of different colors or intensities that can be simultaneously displayed is: 2^d, where d = *depth*. More specifically: 2 colors when *depth* = 1 plane and 256 colors when *depth* = 8 planes. When *depth* = 24 planes, the maximum number of possible simultaneous colors is about 16.7 million — the number of physically available pixels is much less.

On most 8-plane and many 24-plane systems, the pixel values are indices into a *color lookup table*. The table entries determine the red, green, and blue (RGB) components of the displayed color. In a typical lookup table, each entry contains 24 bits, eight each for the red, green, and blue components. The values of the color lookup table entries are under program control. Therefore, when *depth* = 8 planes, a lookup table has 256 24-bit entries, allowing the simultaneous display of 256 different colors selected from a 16.7-million color palette.

A 24-plane system that has no color lookup table is called a *true-color* system. For each pixel, the 24 planes are partitioned into three 8-bit fields that directly determine the pixel RGB values. In 24-plane systems that use color lookup tables, there are three 256×8 lookup tables that allow each color component to be mapped independently.

Systems that "map" color values to a lookup table are called *pseudo-color* systems.

Many frame buffers have additional planes that are used for purposes other than color information; for example, double buffering, Z buffering, and multiple lookup tables for different windows.†

The SPXg frame buffer uses color lookup tables with 512 entries. Normally, access to all 512 colors would require nine frame buffer planes (*depth* = 9). To access 512 colors with 8 planes, the interface, control, and timing chip (JChip—see Section 4.2.1) includes a priority-ordered rectangle generator. The output of this generator, a binary value generated on a per-pixel basis, provides the ninth-plane access to the color lookup table.

The SPXgt frame buffer is a *true-color* frame buffer. The 24 planes can be partitioned into three 8-bit fields to directly determine the RGB values for the pixel. The SPXgt also supports 9-bit pseudo color and multiple lookup tables for different windows.

Video digital-to-analog converters (VDACs) convert the digital RGB values from the lookup table to analog signals that drive the monitor color guns. One parameter related to VDAC performance is the display refresh frequency — the number of times per second that the monitor screen is scanned. A display refresh frequency that is too low (less than 60 Hz) will produce an annoying flicker. The refresh frequency that produces no detectable flicker varies with individual users and ambient conditions. A frequency of 66 Hz is suitable for most applications, and a refresh frequency of 72 Hz is the current optimum. The major factor limiting refresh frequency is the bandwidth of the VDACs. The SPXg and SPXgt support 66 Hz monitors and are capable of supporting 72 Hz monitors.

The color lookup table (RAM) and VDAC are frequently implemented on the same chip. Brooktree calls their version of such chips *RAMDACs*. A cursor generator may be included on the same chip or a separate chip. Mixing the cursor signal with the image signal in the analog domain allows the cursor to move about the display without disturbing the underlying image values in the frame buffer. The SPXg uses a RAMDAC with an internal cursor generator. The SPXgt uses a different RAMDAC and two separate cursor generators.

Graphics acceleration implemented in hardware or in dedicated high-performance processors running firmware can speed up applications in at least three ways:

1. The graphics acceleration hardware is specialized to perform specific graphics processing tasks and can do so faster than a general purpose CPU working with system memory.

† See Advanced Graphics Terms at the end of Section 1.1.

2. Graphics acceleration hardware usually reduces the amount of data that must be communicated over the I/O channel between the CPU and the graphics subsystem. For example, if the accelerator does line drawing, the CPU need send only the end points to the accelerator. But, if the CPU is doing line drawing, it must send every pixel in the line to the graphics subsystem.
3. The graphics accelerator frees the CPU from routine graphics processing to do other work simultaneously with the off-loaded graphics processing.

3.2 SPXg and SPXgt

The SPXg and SPXgt are different implementations of a common graphics acceleration architecture, shown in Figure 4. The graphics acceleration is applied in two compute-intensive stages of the *3D rendering pipeline*: the *geometry engine* and the *scan conversion processor*.

3.2.1 The Rendering Pipeline

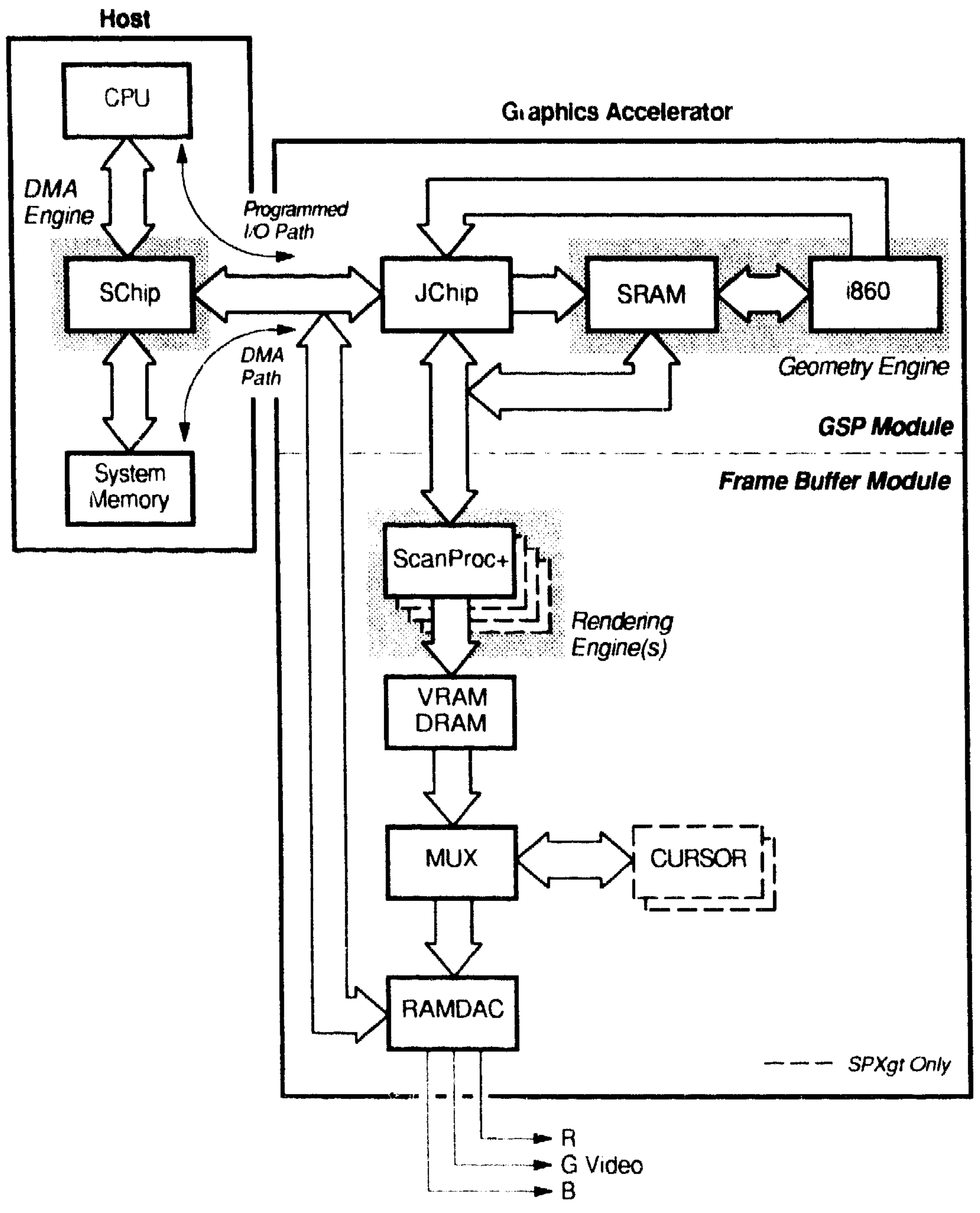
The rendering pipeline is a sequence of processing steps which transform the application-level representation of a 3D scene into the pixel-level description written into the frame buffer. It is called a pipeline because it consists of a sequence of steps to be applied identically to a stream of graphical elements. In the SPXg and SPXgt it is implemented as a hardware pipeline; that is, a sequence of processing stages, each performing one of the rendering functions, operating simultaneously in assembly-line fashion on the incoming stream of elements.

Applications represent the 3D scene by data structures in which the elements are the primitives and attributes defined by the application programming interface (API). The primitives are 3D geometric objects, typically lines and polygons in an abstract 3D model space, and more complex objects made up of such lines and polygons. Typical attributes are line style, surface color, light-source color, and surface reflectance properties. For smooth shading and other advanced features, the application must supply additional data with each primitive, such as vertex colors or vertex normal vectors.

The rendering pipeline begins with traversal of the application data structures defining the scene. The traversal is performed by the API run-time software (in a structure-oriented API) or by the application itself (in an immediate-mode API). In either case, the API run-time software running in the client CPU generates PEX requests and sends them to the PEX server running in the CPU to which the graphics accelerator is attached. The PEX requests are in the terms of geometric primitives, similar to those of the API, so that PEX request generation amounts essentially to reformatting the data structures. The PEX protocol also permits the API run-time software to build graphical data structures which reside in the PEX server process. In this mode, structure traversal is performed by the PEX server in response to a single request from the client — greatly reducing client/server communication overhead.

The PEX server converts the high-level 3D PEX requests to 3D request packets acceptable to the 3D graphics acceleration hardware. This conversion consists essentially of breaking complex primitives into simpler components, such as line segments, convex polygons, and triangle strips, still in the 3D model space of the application.

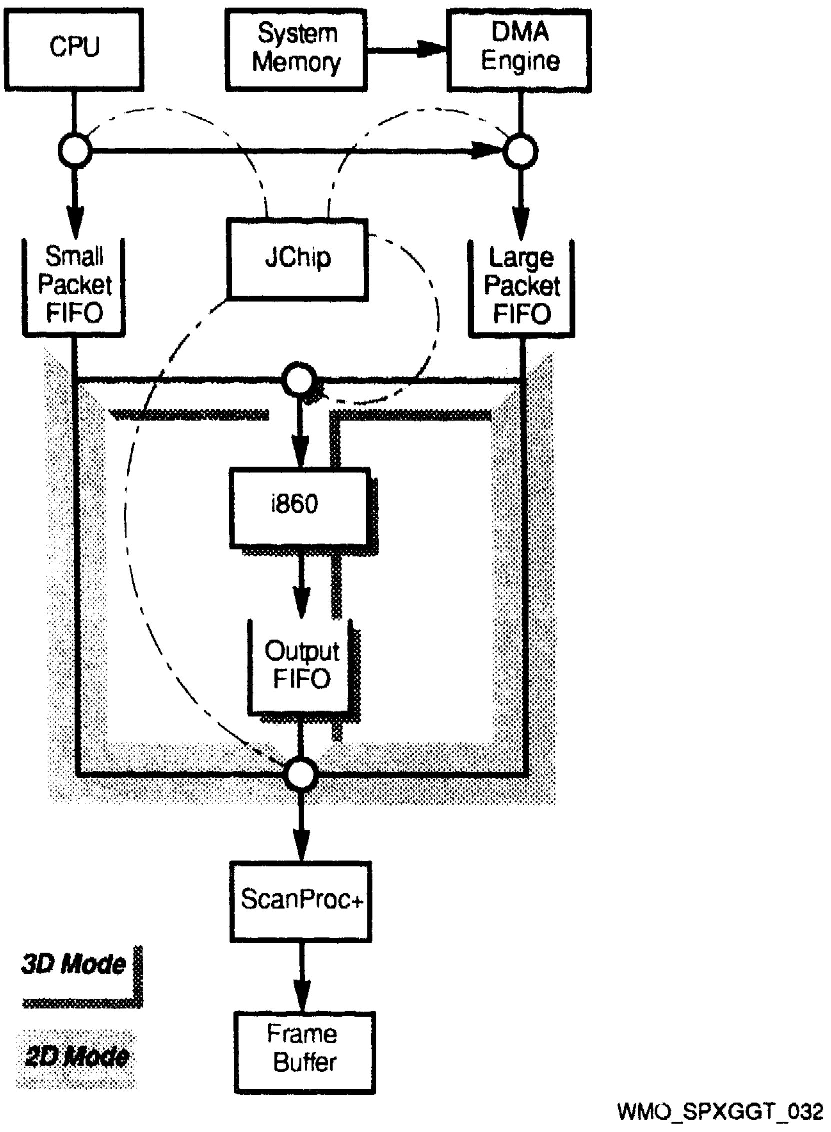
Figure 4 3D Graphics Accelerator Architecture



WMO_SPXGGT 021

The PEX server then transfers these simpler packets to FIFOs located in SRAM on the SPXg and SPXgt and managed by the JChip (Section 4.2.1). Packets can be transferred by the server either directly using programmed I/O, through the *DMA engine* incorporated in the SChip of the base system, or by a combination of both mechanisms (Figure 5).

Figure 5 2D and 3D Packet Flow



The first computational stage in the graphics accelerator is the *geometry engine* (Figure 4). This is an Intel i860 (Section 4.2.1) high-performance, floating-point processor, running software stored in the fast SRAM. The i860 reads 3D requests packets from one set of (input) FIFOs, performs the required geometry and lighting processing, and stores the results in another (output) FIFO. The results include the description of 2D lines and triangles in device coordinates. They may also include per-vertex or per-polygon data, such as color or depth values, that carry information about the original 3D arrangement of the scene. This residual 3D data is needed for advanced rendering functions such as shading, depth cueing, or Z buffering†, which are computed using interpolation at the scan conversion stage.

In the SPXg and SPXgt, scan conversion is performed by the *ScanProc+* (Section 4.2.2) rendering engine(s), which computes the pixel addresses and pixel values for drawing lines and triangles to the frame buffer. Although ScanProc+ computations are in a 2D image space, they include the interpolation in 2D space of certain quantities carrying information about the original 3D scene, such as color or depth values. These interpolations are needed for some rendering features, such as smooth shading or Z-buffer hidden-surface removal.

3.3 Performance

The SPXg and SPXgt offer graphics performance comparable to the DECstation PXG graphics option. The SPXg and SPXgt rendering engine (ScanProc+) is an improved version of the ScanProc (used in the VXT 2000 Decwindows color terminal and in the original SPX graphics option available on the VAXstation 3100 Models 76, 48, 40, 38, and 30). Table 1 compares 2D and 3D performance of the SPX, PXG, SPXg, and SPXgt graphics options in their respective systems.

3.3.1 3D Graphics Benchmarks

Digital's 3D graphics benchmarks measure 3D line and 3D polygon performance using a PEX server benchmark (Table 1). For Digital systems, 3D vectors are 10-pixel, 10 lines/polyline, structure mode. 3D vector results are reported in kilo-vectors per second (Kvectors/s) or 1000 lines per second. For Digital systems, 3D polygons are 100-pixel triangles, 10 triangles per triangle strip, Z buffered, default plus directional lighting. 3D polygon results are shown in kilo-polygons per second (Kpolygons/s) or 1000 polygons per second.

Results and Conclusions

When evaluating graphics performance, it is important to understand the benchmark quoted. One primitive performance number should not be compared to another without knowing how the entities are defined and measured. In addition, primitive-level benchmarks are good for measuring drawing rates for particular entities, but do not take into account other operations that an application may perform (such as *picking* or structure editing), or characteristics of the entire system (such as typical background load or disk I/O). The best way to evaluate a system is to run the actual application.

Characterizing graphics performance is a complex task. How the application is written, the characteristics of the system the application is running on, and the nature of the graphics data itself are all factors that affect graphics performance. In addition, it is possible to measure graphics performance at several different levels. For example, you could measure how long it takes to draw an individual primitive at peak hardware rates, or you could measure how long it takes to set up and draw an entire picture using a high-level API.

Because of the complexity of the problem, there are widely varying approaches within the industry for generating graphics metrics. While *X11perf* offers some standardization in the realm of 2D benchmarks, examination of commonly quoted 3D numbers show that there is little uniformity regarding what is being measured, and at what level it is measured.

3.3.2 Picture-Level Benchmarks (PLB)

Picture-level benchmark (PLB) is software that allows comparisons to be made of graphics display performance for different hardware platforms (Table 1). It is the first product from the Graphics Performance Characterization (GPC) committee, a volunteer group of vendors, users, and consultants that provide and support standardized benchmarks for measuring graphics performance as related to specific applications. The National Computer Graphics Association (NCGA) is administrator for the committee. PLB is designed to measure the performance of CRT-based display systems such as engineering workstations, personal computers, and special-purpose attached display systems. Two requirements exist for the PLB to work. The geometry must be presented to the system in a specified format and the PLB code must have been ported to the device under test. The five major components of the PLB are:

1. *Benchmark Interchange Format* (BIF), the file format for specifying the geometry.
2. *Benchmark Timing Methodology* (BTM) which provides a standardized performance measurement.
3. *Benchmark Reporting Format* (BRF), for standardized reporting of test results.
4. *Picture-Level Benchmark* (PLB) program which implements BIF file processing and runs the test.
5. A suite of standard tests and a report summary sheet.

In order to run BIF files, the PLB code must be customized for each hardware configuration. To date, five application files have been approved by the GPC committee for use. They are:

pc_board	A typical 2D electrical CAD application.
sys_chassis	A 3D wireframe model of a computer chassis.
cyl_head	A 3D solid model of an automobile engine cylinder head.
head	A 3D human head modeled using laser-generated data.
shuttle	An example of low-end 3D simulation.

Note that although the PLB allows buyers to compare performance, it does not address the issue of display quality. It is the user's responsibility to look at the image on the screen and determine superiority.

Results and Conclusions

PLB performance results are reported using a measure called the "GPCmark." The GPCmark is a ratio determined by dividing a normalizing constant by the elapsed time in seconds required to perform the test. The higher the number, the better the performance. Each benchmark generates two numbers; the *PLBlit* (PLB Literal) and the *PLBopt* (PLB Optimized). The PLBlit results of the GPC are most useful for users who know how their applications draw pictures. They select the benchmarks which most closely approximates the software they use, or they develop BIF files for benchmarks. They want to know what the performance of the workstation will be if the picture is drawn as is. PLBopt results are for the users who may make whatever changes necessary to their applications to get the best possible performance for the workstation. The picture will not be drawn as is. Instead, the drawing may be re-ordered, or it might use different primitives, or additional information such as surface normals may be provided. Each GPCmark is reported in the format: PLBlit:PLBopt

Table 1 compares System Performance Evaluation Cooperative (SPEC), X11perf, 3D graphics, and PLBlit benchmark results for the following accelerators and systems:

SPX	VAXstation 3100 Model 76
PXG	DECstation 5000 Model 125
SPXg	VAXstation 4000 Model 60
SPXgt	VAXstation 4000 Model 60

Table 1 Graphics Performance Summary

Benchmark		SPX	PXG	SPXg	SPXgt
SPECmark	SPEC	6.8	19.3	12.0	12.0
SPECint	SPEC	7.1	16.1	11.1	11.1
SPECfp	SPEC	6.6	21.7	12.6	12.6
2D Fill Area Mpixels/s[1]	X11perf	14.2	13.9	24.8	10.4
2D Kvectors/s [2]	X11perf	183.0	259.0	365.0	371.0
3D Kvectors/s [2]	Digital 3D	57.0	288.0	295.0	300.0
3D Kpolygons/s [3]	Digital 3D	5.9	51.0	26.0	33.0
pc_board	PLB	NR[4]	9.8	11.9	12.3
sys_chassis	PLB	NR	10.5	11.0	11.1
cyl_head	PLB	NR	14.4	NR	8.4
head	PLB	NR	18.8	NR	8.5
shuttle	PLB	NR	17.6	NR	12.5

[1]Copy 500 × 500 pixels from pixmap to windows

[2]10 pixels/vector

[3]100 pixels/triangle, in strips

[4]Not reported

4

Option Hardware Overview

4.1 Physical Overview

Figure 4 (above) is a simplified block diagram showing the major functional areas and partitioning of either 3D graphics accelerator. Both accelerators are multi-board options.

4.1.1 SPXg

The SPXg is contained on four modules: Graphics subsystem processor (GSP), frame buffer, and two single in-line memory modules (SIMMs).

GSP Module

This module is plug-compatible with the 2D graphics options. It occupies the same space and has the same physical interface. Its dimensions are approximately 16.0×20.3 cm (6.3×8.0 in). It includes the i860 processor, junction chip (JChip), SRAM, diagnostic ROM, 150-pin system connector, and two 96-pin ScanProc bus connectors. The two 96-pin connectors mate with corresponding connectors on the frame buffer module. The module uses single-sided mixed surface-mount and through-hole technology. The through-hole devices are primarily i860 and JChip pin-grid arrays (PGAs) and connectors.

8-Plane Frame Buffer Module

This module mounts horizontally on top of the GSP module. Its dimensions approximately 16.0×20.3 cm (6.3×8.0 in). It includes the scanline processor (ScanProc+), VRAM, Bt460 RAMDAC, monitor select switch, two SIMM connectors, two 96-pin ScanProc bus connectors, and the RGB connector for high resolution monitors. The two 96-pin connectors mate with corresponding connectors on the GSP module. The module uses single-sided mixed surface-mount technology with some PGA and ZIP through-hole components.

SIMMs

The two SIMMs mount on top of the frame buffer module, at angle of about 45° from the vertical. Each is approximately 11.9×2.7 cm (4.7×1.05 in), and contains 16 DRAMs specially configured as off-screen frame buffer memory. The modules use double-sided surface-mount technology consisting of DRAMS and capacitors. Each provides 2 Mbytes of off-screen memory.

4.1.2 SPXgt

The SPXgt is contained on two modules: GSP and frame buffer.

GSP Module

This is the same module used in the SPXg described above.

24-Plane Frame Buffer Module

This module mounts horizontally on top of the GSP module. It is roughly L-shaped and its overall dimensions are approximately 25.9×20.3 cm (10.2×8.0 in). It includes four ScanProc+, VRAM, DRAM, Bt463 RAMDAC, two Bt431 cursor generators, two 96-pin ScanProc bus connectors, and the RGB connector for high resolution monitors The two 96-pin connectors mate with corresponding connectors on the GSP module. The module uses double-sided surface-mount active components as well as capacitors under RAMs. All components are surface-mount except the RAMDAC and connectors. VRAM and DRAM are wholly contained on this module.

4.1.3 Component Summary

The following table compares the major components of the SPXg and SPXgt.

SPXg	SPXgt
1 DC7208 JChip	1 DC7208 JChip
1 DC242 ScanProc+	4 DC242 ScanProc+
2 Mbyte VRAM	8 Mbyte VRAM
4 Mbyte DRAM	8 Mbyte DRAM
1 Bt460 RAMDAC	1 Bt463 RAMDAC
	2 Bt431 Cursor Generator
1 i860 Processor	1 i860 Processor
512 Kbyte SRAM	512 Kbyte SRAM
128 Kbyte Diagnostic ROM	128 Kbyte Diagnostic ROM

4.2 Functional Overview

Both accelerators include: the new Digital junction chip (JChip) for interface, control, timing functions; the improved Digital scanline processor (ScanProc+) rendering engine; an Intel i860 geometry processor; and either the Brooktree Bt460 (SPXg) or Bt463 (SPXgt) RAMDAC.

4.2.1 GSP Module

The Graphics Subsystem Processor module includes the JChip, i860, and diagnostic ROM.

JChip

The JChip (DC7208) is a 53-Kgate application-specific integrated circuit (ASIC) in a 299-pin PGA through hole package. It provides the following major functions:

- Interface between the i860, ScanProc+, SRAM, and the host's SChip.
- Shared memory access that allows the host and the i860 to share the ScanProc+ registers, JChip registers, and SRAM.
- FIFO control logic that allows a selected portion of SRAM to appear as FIFOs.
- Buffers drawing commands from the host to the graphics processor through the FIFO command buffer. This frees the host to do other work without waiting for the graphics processor.

- Data conversion facilities that allow the host, when writing to a FIFO, to optionally instruct the JChip to convert the write command to a ScanProc+ command or IEEE single-precision floating-point data.
- Monitor and VRAM refresh timing.

i860

The Intel i860 is a 1-Mtransistor, 64-bit general-purpose CPU. It contains:

- Integer control, bus, and paging units.
- Floating-point control, adder/subtractor, multiplier, and 3D graphics units.
- Instruction and data caches.

Diagnostic ROM

The diagnostic ROM is a 128-Kbyte ROM. (Setting a bit in the JChip diagnostic register provides 128 Kbytes of additional addressing, allowing a 256-Kbyte ROM to be used.) The ROM contents must be copied into system main memory before execution.

4.2.2 Frame Buffers

Both frame buffers include ScanProc+, RAMDACs, on-screen and off-screen memory, and configuration register.

The 8-plane frame buffer includes one ScanProc+, 2 Mbytes of VRAM, 4 Mbytes of DRAM, and a Bt460 RAMDAC.

The 24-plane frame buffer includes four ScanProc+, 8 Mbytes of VRAM, 8 Mbytes of DRAM, a Bt463 RAMDAC, and two Bt431 cursor generators. In addition to the 24 color planes, this frame buffer has four window tag planes and four overlay planes. (Note: The X server does not currently support the overlay planes. The Bt463 RAMDAC description, below, describes how all 32 planes are used.)

When enabled by the JChip, RAMDAC or cursor data can be loaded directly from a program image in the off-screen portion of VRAM. Bits in the JChip diagnostic register control whether the Bt463 or Bt431s are loaded during an SPXgt frame buffer load image operation. The load image begins at the end of the visible portion of the frame buffer (that is, line 1025 in a 1280×1024 display).

ScanProc+

The ScanProc+ (DC242) is a 192-Ktransistor, 132-pin surface-mounted, fully custom graphics processor. It runs at 15.6M graphics instructions per second. The ScanProc+ is dual-ported: to the host through a processor interface and the JChip, and to the VRAMs and DRAMs through a frame buffer interface. The ScanProc+ features include:

- A controller running from on-chip RAM (128×48) and ROM (640×48).
- An ALU with specialized functions for fast rendering of graphics objects.
- An address generator to perform next-pixel-address calculations for common objects in parallel with the graphics ALU.
- A scratch memory that takes advantage of fast page-mode access to the VRAMs.

The ScanProc+ is coded to implement the Xlib instruction set and perform Xlib functions with a minimum of CPU intervention. The graphics hardware accepts Xlib parameters and provides the exact pixel coverage specified by X11. This provides applications with faster graphics as well as more efficient CPU usage, since the CPU is not required to accommodate X requests to limited or mismatched hardware capabilities.

Configuration Register

The configuration register bits indicate whether an SPXg or SPXgt frame buffer is installed, which SIMMs are present in the SPXg frame buffer, and whether the SPXg frame buffer is set up for a 66 Hz or 72 Hz monitor.

Bt460 RAMDAC

The Bt460 provides up to 512 entries in the color lookup table. Normally, fully general access to the 512 colors would require nine frame buffer planes. To avoid the addition of a ninth frame buffer plane, the JChip provides a four-level priority-ordered rectangle generator. The rectangle generator output is a binary value generated on a per-pixel basis. It specifies the ninth-plane input to the Bt460.

The Bt460 has approximately 2.5 Kbytes of state consisting of 1536 bytes of color lookup table and 1024 bytes of two-plane cursor pixel maps. In addition, the Bt460 includes control registers that must be programmed at least once to configure the Bt460 or move the cursor position.

The Bt460 is a 132-pin PGA device. It provides video multiplexers for eight video planes and five overlay planes, a 512×24 primary color map memory, and three 8-bit video DACs. Each video DAC can drive 1V ground-referenced RS343A-compatible video into double terminated 75 Ohm cable.

For more information refer to the vendor's product literature.

Bt463 RAMDAC

The Bt463 RAMDAC is the result of a cooperative effort by Digital and Brooktree. It implements the XMAP architecture to provide extensive pixel mapping support.

The Bt463 supports 24-bit pixel values, four overlay planes, and four window tag planes. The Bt463 treats the 24 color planes (0 through 23) and four overlay planes (24 through 27) as a general 28-bit pixel value. XMAP control registers in the Bt463 determine the assignment of input planes to color table entries based on the value of the four plane window tag field.

The red channel planes (0 through 7) are wired only to the Bt463. The green channel planes (8 through 15) are wired to the Bt463 and Bt431 cursor plane 0. The blue channel planes (16 through 23) are wired to the Bt463 and Bt431 cursor plane 1.

Planes 28 through 31 (window tag planes), the two cursor planes (CUR0 and CUR1), and the JChip priority-encoded rectangle generator (LMAP) are combined as the window tag (0 through 3) input to the Bt463, according to the values of CUR0 and CUR1. This scheme allows device drivers to use the LMAP rectangle generator or the four window tag planes to control the Bt463 XMAP functions.

The Bt463 has approximately 1.5 Kbytes of state consisting of 1536 bytes of color lookup table and 48 bytes of cursor color. In addition, the Bt463 includes control registers that must be programmed at least once to configure the Bt463 or redefine the internal XMAP registers.

The Bt463 is a 169-pin PGA device. It provides three 528×8 look-up tables. It supports 24-bit true color, 9-bit pseudo color, and color graphics that address multiple look-up tables for different windows. Each of the three 8-bit video DACs can drive 1V ground-referenced RS343A-compatible video into double terminated 75 Ohm cable.

For more information refer to the vendor's product literature.

Bt431 Cursor Generator

The SPXgt uses two Bt431s to provide a two-plane cursor, where the planes define the cursor as foreground, background, or transparent (the fourth combination is not used).

The green channel planes (8 through 15) are wired to the Bt463 and Bt431 cursor plane 0. The blue channel planes (16 through 23) are wired to the Bt463 and Bt431 cursor plane 1.

Each Bt431 has 512 bytes of cursor pixel map as well as control registers that must be programmed at least once to configure the Bt431.

The Bt431 provides a user-definable cursor and a cross-hair cursor. Both cursors can be displayed simultaneously or individually enabled or disabled. The cursors can be positioned with pixel resolution and can be moved off the display without wraparound.

For more information refer to the vendor's product literature.

4.3 Direct Programmed I/O

The SPXg and SPXgt provide direct programmed I/O access to the following peripheral devices.

SPXg	SPXgt
Configuration Register	Configuration Register
Diagnostic ROM	Diagnostic ROM
Bt460 RAMDAC	Bt463 RAMDAC
	Bt431 Cursor Plane 0
	Bt431 Cursor Plane 1