# *TurboGX and TurboGXplus Graphics Technology*

*A White Paper*

Sun

# Contents

# Introduction 1

The TurboGX/TurboGXplus™ Graphics Accelerators are Sun's low-cost 8-bit graphics accelerators designed for a wide variety of users including Electronic Design Automation (EDA), Architecture, Engineering and Construction (AEC), Electronic Publishing (EPUBS), Computer Aided Software Engineering (CASE), mid- to low-end Mechanical Computer Aided Design (MCAD), Geographic Information Systems (GIS) and Mapping, Pharmaceutical/Chemical, and low-end image processing markets. The TurboGX/TurboGXplus accelerators represents the third-generation of GX™-based accelerators and incorporates a large number of new features, including substantial performance improvement, memory system enhancements, increased resolution support, and a variety of other capabilities.

The original GX accelerator was initially introduced in early 1989. Subsequent enhancements over several generations of more powerful and more highly integrated implementations have led to over 300,000 GX systems installed today — making GX-based graphics platforms the most popular workstation graphics technology in history.

The TurboGX/TurboGXplus accelerators are low-cost, general purpose graphics accelerators designed using highly integrated and sophisticated ASIC technology. Over the last two generations, the GX chip set has migrated to a highly optimized single-chip implementation, with increasingly greater performance and functionality.

The TurboGX/TurboGXplus accelerators speed the rendering of all of Sun's application programming interfaces (API's) including Xlib, the Direct Xlib™ Graphics Library, the SunPHIGS™ Graphics Library, the XIL™ Imaging Library, and the XGL™ Graphics Library. The TurboGX/TurboGXplus accelerators also improve overall windowing performance for all users, regardless of the applications in use. Sun continues to lead the way in providing graphics acceleration that is affordable enough to be on every desktop.

The TurboGX/TurboGXplus accelerators comes with multiple-resolution support, 8 bits of indexed color, and a 256-entry colormap loadable with 24-bit RGB entries for a total choice of 16.67 million colors. The TurboGX/TurboGXplus accelerator offers fast 2-D and 3-D vectors for direct manipulation of complex wireframe models, and fast screen clear, text scrolling, and raster operations for an interactive window environment, as well as fast flat shading for solids rendering.

With multiple-resolution support ranging from 1024 x 768 to 1600 x1280, the TurboGX/TurboGXplus accelerators can be configured with a range of monitors, thus providing a number of alternatives to meet the budgets and display requirements of many types of users.

The TurboGXplus accelerator also incorporates 100 MFLOPS of equivalent floating point for graphics computation, hardware double buffering, a hardware cursor, and many other capabilities. With extremely fast screen clear times, a hardware transformation pipeline, and double buffering, the TurboGXplus accelerator allows for smooth animation of sophisticated 2-D and 3-D wireframe objects.

The TurboGX/TurboGXplus accelerators are a milestone in integrated 8-bit vector-based graphics. Extending beyond the features and performance of its predecessors, they are ideal graphics accelerators for a wide range of graphics application requirements. This white paper discusses the design, architecture, and features of the TurboGX/TurboGXplus ASIC, TurboGX accelerator card, and TurboGXplus accelerator card.

# TurboGX/TurboGXplus Accelerators Features

2

The TurboGX/TurboGXplus accelerators are implemented as single-width (approximately 3 x 5 in.) SBus boards, populated with the TurboGX/TurboGXplus custom ASIC, VRAM, a digital-to-analog controller (DAC), and a programmable clock device for video timing.

The single-chip TurboGX/TurboGXplus ASIC provides a sophisticated synchronous SBus interface, multi-buffering capability, and 100% binary code compatibility with the existing GX software base. In addition, this chip is the heart of the TurboGX accelerator and the multi-buffered, high-resolution TurboGXplus accelerator. Figure 2.1 illustrates the TurboGXplus accelerator.

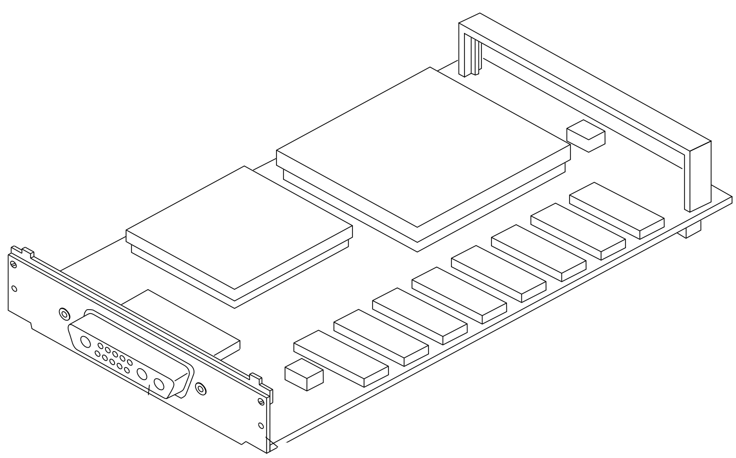

*Figure 2-1* The TurboGXplus Single SBus Accelerator

The original FBC/TEC/SIF (frame buffer controller, transform engine and cursor, and SBus interface chips) and its single-chip successor GX implementation introduced new levels in device and functional integration within the graphic accelerator hardware arena. Once again, with the introduction of the TurboGX/TurboGXplus ASIC, the concept of highly integrated design is redefined. No other geometry graphics product requires the addition of as little as video RAM and a clock device to make a fully functional accelerator in the class of the TurboGX/TurboGXplus accelerators. Figure 2.2 illustrates the graphics system architecture of the TurboGX/TurboGXplus implementation

.

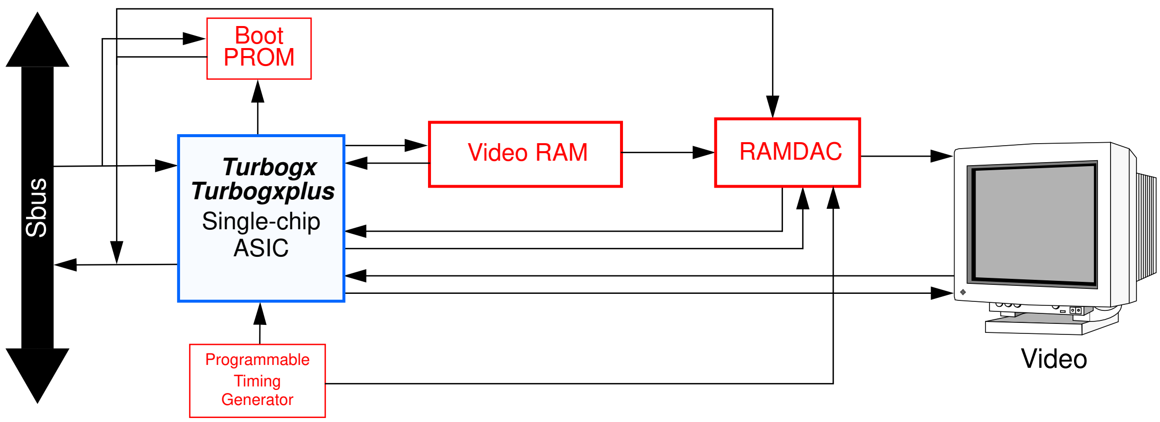


*Figure 2-2* The TurboGX/TurboGXplus Single-Chip Implementation

## *TurboGX and TurboGXplus Accelerator Differences*

The TurboGX and TurboGXplus accelerators are based upon the same, highly integrated, low-cost ASIC. The TurboGX is Sun's baselevel graphics accelerators, offered standard with SPARCstation 10 systems. The TurboGXplus offers higher performance and more flexibility than the TurboGX.

The TurboGXplus offers higher performance, supports double buffering and supports a wider range of monitor resolutions. The differences between TurboGX and TurboGXplus are shown below in Table 2.1

| Feature | TurboGX | TurboGXplus |
|---|---|---|
| **Performance**<br>**- 2D vectors (Xlib)**<br>**- 3D vectors (XGL)** | 762,000<br>405,000 | 1,500,000<br>499,000 |
| **Frame Buffer Memory** | 1 MB | 4 MB |
| **ASIC** | TurboGX/TurboGXplus ASIC | TurboGX/TurboGXplus ASIC |
| **Clock speed** | 35 MHz | 46MHz |
| **Monitor Resolutions Supported** | 1024x768 @ 60 Hz, 77Hz<br>1152x900 @ 66 Hz, 76 Hz | 1024x768 @ 60 Hz, 77Hz<br>1152x900 @ 66 Hz, 76 Hz<br>1280x1024 @ 67Hz, 76 Hz<br>1600x1280 @ 76 Hz |
| **Double Buffering Support** | no | yes |
| **Form Factor** | Single SBus slot | Single SBus slot |

*Table 2-1* Differences between the TurboGX and TurboGXplus

# *TurboGX/TurboGXplus Accelerator Architectural Design Goals*

## *Standard Color*

The first design of the GX accelerator was motivated by the desire for it to become a standard feature on Sun workstations. The pursuit of this goal focussed the GX accelerator designers on supported functionality and implementation costs while they strove for acceleration that would outpace the rapidly increasing performance levels of the base workstation.

By departing from the way graphics had been accelerated in the past, the GX accelerator and successor products achieved these goals. All the ideas and technology behind the GX accelerator were incorporated in the industry's first high-performance, single-chip graphics accelerator. The latest, third-generation implementation resulting in the TurboGX/TurboGXplus accelerators introduces further enhancements and refinements to the earlier GX architectures while maintaining the original design goals.

## The 80/20 Rule

Previous graphics accelerators have typically incorporated either a CPU or bit-slice processor to control specialized graphics hardware and to do much of the rendering. Until now, the support hardware, memory, cache, power supply, and board size prevented graphics acceleration from being placed on entry-level workstations. The TurboGX/TurboGXplus accelerators use the host CPU as the intelligent controller, and hard wires all its graphics functionality within custom ASICs.

From an instruction set point of view, the TurboGX/TurboGXplus chip incorporates principles most commonly associated with RISC computer architectures. RISC architects studying CISC CPU instruction usage uncovered the 80/20 rule, which states that typically 80% of the time is spent doing 20% of the operations. The TurboGX/TurboGXplus architects found that this same 80/20 rule applied to most graphics applications. This led the designers of the TurboGX/TurboGXplus architecture to support only a minimal set of orthogonal operators, as is the case for most RISC CPU architectures. This reduced instruction set yields just three basic functions. These functions were determined to be the key primitive operations, or the 20% used most often, whose acceleration would yield the most benefit to the majority of applications.

The basic operations supported by the TurboGX/TurboGXplus chip are:

- Rendering arbitrary, filled quadrilaterals (DRAWs)
- Displaying precomputed pixel images (FONTs)
- Block image transfers (BLITs)

With these instructions the TurboGX/TurboGXplus chip can directly accelerate at least 80% of the most-used functions in its targeted markets: window systems, 2-D geometry, and 3-D wireframe.

Within its restricted set of graphical operators, the TurboGX/TurboGXplus accelerators extract the highest performance possible by hard wiring, customizing, and optimizing all hardware resources and their control. The highly parallel, highly pipelined, state machine design style accounts for much of the TurboGX/TurboGXplus accelerators' performance advantage over all other comparably priced accelerators.

By utilizing this high degree of parallelism, coupled with an integral high performance math unit for transformations and a hardware cursor, the TurboGX/TurboGXplus accelerators consistently yield a significant graphical performance advantage across a wide range of price and performance workstation platforms.

## TurboGX/TurboGXplus Design Characteristics

The heart of TurboGX/TurboGXplus accelerators is a highly integrated ASIC employing over 128,000 gates and implementing an extended superset of previous GX architectures, including the GX and GXplus boards. This TurboGX/TurboGXplus chip provides an integral SBus interface, VRAM (video random access memory) controller, a high-performance math engine, plus a high performance rendering (or drawing) engine. A complete graphics accelerator can be built utilizing this chip, a clock device, VRAM, and a RAMDAC.

The new TurboGX/TurboGXplus chip delivers an unparalleled level of integration that has allowed for its inclusion on a single-wide SBus card. These products also make multi-headed acceleration a cost-effective reality. In addition to the TurboGX/TurboGXplus chip's enhanced functionality, it offers high-resolution, off-screen memory, and multi-buffering capabilities.

Some key features of the single-chip TurboGX/TurboGXplus ASIC are:

- 100 percent compatibility with previous GX implementations
- Complete graphics accelerator in a single chip — just add VRAM, a RAMDAC and a clock device
- Large-scale integration in. 6 micron custom CMOS
- Over 128,000 gates (~500,000 transistors)
- Integral high-performance rendering engine (up to 1600 Mpixels/s)
- Integral high-performance math engine (up to 100 MFLOPS)
- Integral asynchronous SBus interface (rendering and math performance independent of SBus clock rate)
- 1-Mb and 2-Mb VRAM support
- Integral high-performance VRAM controller
- New block mode support for high-speed solid fill
- Multi-vector processing capability for 2 vectors per command
- 8x16 Font FIFO
- FIFO and destination caches
- Multi-mode, multi-buffering support assuring MIT X11 compliance
- High-speed block copy for raster copies between buffers (eliminates visual anomalies, or tearing during double buffering)

- Off-screen rendering support (improves window system performance)
- Enhanced programmable timing logic with support for flicker-free 76 Hz vertical refresh rates and multi-resolution support including 1024x768, 1152x900, 1280x1024 and 1600x1280..

## *Frame Buffer Memory*

The TurboGX/TurboGXplus graphics accelerators are based on an 8-bit indexed color model. This allows for the simultaneous display of 256 colors out of a possible 16,777,216 color choices.

The TurboGX accelerator is a 1-MB frame buffer using eight 80-ns 1 Mb VRAM devices.

The TurboGXplus accelerator is a 4-MB frame buffer using sixteen 60-ns 2-Mb VRAM devices.

## *Off-Screen Memory*

The TurboGXplus accelerator's 4-MB frame buffer can be flexibly and dynamically mapped into either two 2-MB frame buffers (for double and multi-buffered applications), or a single 4-MB frame buffer. In most cases, a large percentage of the frame buffer beyond that required for the TurboGXplus accelerator's screen display is non-displayed and available. This additional memory is used automatically in the OpenWindows™ environment for off-screen accelerated rendering and to cache and store images/objects for later use. Note that when double buffering is used, less off-screen memory is available for the caching of pixmaps. Table 2.2 shows the available off-screen memory associated with each TurboGXplus resolution. Off-screen memory and its use by Sun's graphics APIs and OpenWindows environment is discussed further in Chapter 4: *TurboGX/TurboGXplus Software*.

| VRAM | Off-screen Memory | Resolution | Buffering | Scan Rate |
|---|---|---|---|---|
| 4 MByte | None | 1600x1280 | Double | 76 Hz |
| | 2 MB | 1600x1280 | Single | 76 Hz |
| | .75 MB | 1280x1024 | Double | 67/76 Hz |
| | 2.75 MB | 1280x1024 | Single | 67/76 Hz |
| | 1 MB | 1152x900 | Double | 66/76 Hz |
| | 3 MB | 1152x900 | Single | 66/76 Hz |
| | 1.25 MB | 1024x768 | Double | 60/77 Hz |
| | 3.25 MB | 1024x768 | Single | 60/77 Hz |

*Table 2-2* TurboGXplus Resolutions, Memory, and Buffering Options

# TurboGX/TurboGXplus Accelerators Functionality

## Accelerated Graphics Primitives

The TurboGX/TurboGXplus boards accelerate common geometric primitives including points, lines, triangles, and rectangles. Each of these primitives is considered a special case of a more general primitive, the quadrilateral.

The TurboGX/TurboGXplus accelerators have a wide data path into its frame buffer that allows 32 adjacent pixels to be modified (or accessed) in each memory access. This high degree of parallelism enables rendering rates of up to 1600 Mpixels per second An object's ability to utilize this bandwidth depends on its width. A 32 pixel-wide object will be able to utilize all of the available data bandwidth while a 1 pixel-wide object (a vertical vector for instance) will only make use of 1/32 of it.

The TurboGX/TurboGXplus also provide, in hardware, a pattern memory for use in blending or stippling a flat-shaded object between two colors, typically a foreground and a background color.

The TurboGX/TurboGXplus accelerators' pattern memory consists of application-defineable memory, which is 16x16x1 pixels deep and is used as source input to a variety of ROP operations that may be performed on a given quadrilateral or triangle. Such operations utilize the foreground and background colors to determine which ROPs are available (generally 4 out of the total of 16). The foreground/background colors are then combined with the ROP operation and pattern memory set by the application to render the resulting stipple pattern onto the primitive.

Pattern memory can be used to provide a greater variety of shaded appearance for polygons, such as in 3-D images, simulation, and 2-D applications including mapping and business graphics.

## Three-color Hardware Cursor — 2 Colors + Transparent

The TurboGX/TurboGXplus accelerators provide support for a three-color 32x32-pixel hardware cursor. This capability can have a noticeably positive effect on window system and graphic application performance. The alternative, a software based cursor, has much higher overhead in terms of CPU cycles and bus traffic. The TurboGX/TurboGXplus hardware cursor provides a low overhead, non-flashing, nondestructive cursor image that maximizes the availability of both the CPU, the SBus, and the accelerator.

## Fast Screen Clear

The TurboGX/TurboGXplus accelerators use its fast flat shading of rectangular objects (another form of a quadrilateral) to clear the full screen in under .7 ms (milliseconds) for an 1152x900 display and under .9 ms for a full-screen 1280x1024 display.

With a full-screen display of a moderately complex geometric object (on the order of 500 10-pixel vectors), the TurboGX/TurboGXplus accelerators update the screen at a rate of approximately 100 frames per second on an 1152x900 display, or up to about 80 frames per second for a 1280x1024 display.

## Fast Scrolling Text and Raster Operations

Providing for maximum possible window system performance is a hallmark of the TurboGX/TurboGXplus. The TurboGX/TurboGXplus accelerators move rectangular images (BITBLT) about the screen at speeds up to 120 Mpixels per second. This hardware facility is utilized to achieve exceptional text scrolling performance.

## Hardware Transformation Pipeline

In addition to the rendering capabilities of the TurboGX/TurboGXplus, integral support is provided for the computational requirements of graphics programs. The TurboGX/TurboGXplus accelerators contain a vector register set and a 32-bit multiply/accumulator, which combine to provide arithmetic calculations at a rate up to 100 MFLOPS. The TurboGX/TurboGXplus accelerators include explicit support for the typical 2-D and 3-D graphics transformation pipeline, as shown in Figure 2.3

.

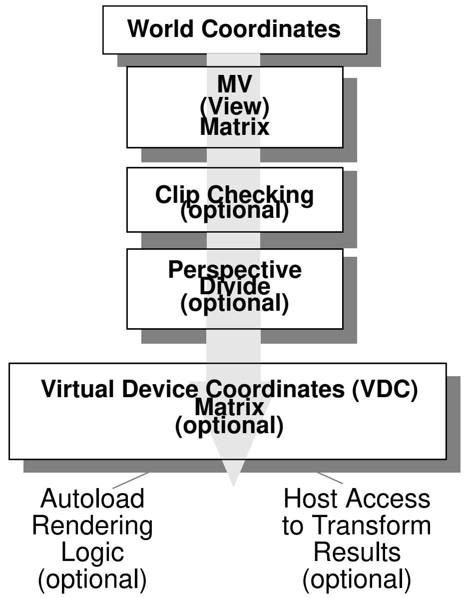


*Figure 2-3* The TurboGX/TurboGXplus Transformation Pipeline

The modeling and viewing matrix (MV) can be anything from a simple 3x2 matrix used by 2-D graphics, up to a 4x4 matrix used by 3-D graphics. The MV matrix is used for rotating, scaling, and translating the object or its viewing position. Optional clip checking is provided to a rectangular region for 2-D graphics, to a viewing pyramid for 3-D graphics, or to a double pyramid for homogeneous 3-D objects such as parametric curves. An optional perspective divide is also available in the hardware.

At the end of the pipeline, there is an optional virtual device coordinate matrix (VDC) that transforms the coordinates into the final screen coordinate space. The transformation results can be read by the host and/or automatically loaded into the drawing logic for rendering. The host also has access to intermediate pipeline results. This flexibility is used by graphic library software running on the TurboGX/TurboGXplus accelerators, to accelerate other math operations beyond coordinate transformations such as lighting and shading calculations, depth cueing, and in recovering from certain math errors.

The three most common graphical data formats are supported. Data can be read or written in either signed integer, fract (16.16), or IEEE single precision floating point formats. Different formats can be intermixed and all conversions are done without any performance penalty. The TurboGX/TurboGXplus accelerators accept both absolute and relative 2-D (x,y), 3-D (x,y,z), and homogeneous (x,y,z,w) coordinate data.

The TurboGX/TurboGXplus math capability extends to general purpose vector operations as well. This allows it to do the arithmetic required for matrix concatenation, dot products, lighting models, and even some common imaging operations, such as convolutions.

## *Hardware Double Buffering Support*

To provide a high-quality display when animating a drawing, it is often desirable to only view completely drawn images or scenes. To accomplish this, some means must be provided for preventing objects being rendered from appearing on the display until they are complete. The TurboGXplus accelerator has implemented a sophisticated, hardware-intensive, high-performance multi-buffering scheme, as well as supporting colormap double buffering (described in the next section).

In the more general case of multiple applications requiring more than one buffer, a second mode is utilized. In this case, all rendering is done to the non-displayed buffer and then copied (via the high speed buffer copy discussed below) into the visible buffer. This mode supports the semantics required by the *MIT X11 Multi-Buffering Extension*. When supporting multi-buffered applications, this “buffer copy” mode is utilized by the XGL and SunPHIGS graphics libraries and the OpenWindows environment.

Although the TurboGXplus accelerator provides new functionality, it retains 100% software compatibility with applications and libraries written for a single buffered GX-based system. This compatibility is implemented in such a way as to achieve the same high performance on either the single-buffered or multi-buffered application.

The TurboGXplus accelerator uses hardware double buffering for multi-buffering requests of two buffers. Other multi-buffering requests may still accelerate drawing if the buffers are cached into off-screen memory.

## *Colormap Multi-Buffering Support*

An additional multi-buffering technique available with the TurboGX/TurboGXplus accelerators is referred to as colormap double buffering. This technique involves manipulating its 256-entry color map table. The colormap table is used to translate the frame buffer's 8-bit color indices into the appropriate 24-bit RGB values intended for color reproduction by the monitor. Given that hardware double buffering is a standard feature on the TurboGX/TurboGXplus accelerators, this multi-buffering approach is generally useful to applications with buffering requirements beyond what is provided (e.g., more than two buffers of greater than 256 colors each), or to maintain compatibility with other GX systems that do not include double-buffering support.

In the most common application of this technique, the 8 bits of color index per pixel are logically divided into 2 groups of 4 bits. Each group is viewed as a buffer. The colormap is alternately loaded with translation values that ignore (i.e., make a NOP) the upper or lower 4 bits of each 8-bit pixel's index. The TurboGX/TurboGXplus rendering hardware then is instructed to render only into one or the other 4-bit group of each pixel. By rendering into the 4-bit group (or buffer) that is currently "ignored" by the colormap translation, the effects of the rendering are not visible on the system's monitor. When the rendering is complete, the colormap can be reloaded to ignore the other buffer and display just the newly completed image or scene. The process then repeats by alternating the rendering between the two buffers/plane groups. This is frequently referred to as colormap double buffering.

The software double buffering scheme is functionally and reasonably inexpensive to implement. However, it has the drawback of limiting the number of colors available for rendering within each buffer. In the case described, only 4 bits of index are available for color selection in each group therefore, only 16 (2**4) colors can be identified (versus the normal 256 selectable colors with the full 8-bit index). This also implies that the double-buffering application assumes it has exclusive use of the entire colormap. However, the colormap is actually a shared resource. Unless all other concurrently running applications are either non-displayed, obscured, or cooperating, some artifacts of the double buffering, such as color flashing, may be observed elsewhere on the screen. Another limitation of software double buffering is the need to completely reload the 256-entry color map between each frame. This reloading is required to switch the ignored and the displayed plane groups. Since the reloading takes a finite amount of time (a minimum of 768 SBus clock cycles) the achievable frame rate is reduced.

## *Monitor Resolution Support*

The TurboGX/TurboGXplus accelerators support the resolutions and refresh frequencies illustrated in Table 2.3 below:

| Frame Buffer | Resolution | Refresh Frequency |
|---|---|---|
| TurboGX and TurboGXplus | 1024 x 768 | 60 Hz |
| TurboGX and TurboGXplus | 1024 x 768 | 77 Hz |
| TurboGX and TurboGXplus | 1152 x 900 | 66 Hz |
| TurboGX and TurboGXplus | 1152 x 900 | 76 Hz |
| TurboGXplus only | 1280 x 1024 | 67 Hz |
| TurboGXplus only | 1280 x 1024 | 76 Hz |
| TurboGXplus only | 1600 x 1280 | 76 Hz |

*Table 2-3* TurboGX/TurboGXplus Accelerators Resolutions and Frequencies

The TurboGX/TurboGXplus accelerators employ a sense code to determine the correct resolution and refresh frequency of the attached monitor. The sense code is defined by 3 pins on the monitor cable. At boot time, the TurboGX/TurboGXplus accelerators detect the sense code and uses its value to obtain the resolution and frequency from a table in the PROM.

In some cases, it may be desirable to override the sense bus. This is possible by using the `setenv` command while in the boot PROM monitor program.

## *TurboGX/TurboGXplus Accelerators Enhancements over GX and GXplus Systems*

In many ways, the TurboGX/TurboGXplus accelerators have leveraged the design philosophy and architectural strengths of previous GX implementations. From a detailed perspective, however, many new refinements were added to the TurboGX/TurboGXplus accelerators that are not readily apparent but are nonetheless very important. The following points illustrate some of these additions and refinements:

- Internal clock rate doubled for improved rendering, fetch, and other operations
- Improvements in memory system, including 2-cycle, 16-pixel writes (versus 3-cycle previously), and a new memory model that allows 2 bank implementations (versus 1 previously)
- New math engine running up to 100 MFLOPS, providing a doubling of transformation performance
- Block mode support added allowing a 4x improvement in solid fills

- New BLIT algorithm that eliminates previous performance penalties for unaligned pixels on copy
- New multi-vector mode providing for rendering of 2 vectors per command (packed vector format)
- Numerous pipeline improvements including an additional stage for the internal rendering setup of short vectors
- Improvements in fast ROP modes, permitting most raster operations to become write-only with 2-to-1 performance gains
- Improved asynchronous SBus interface with increased read rates of 4-5 cycles (versus 6-7 previously)
- Exception handling list reduced, enabling the TurboGX/TurboGXplus logic to manage more exceptions rather that defaulting to software
- Improved video timing generator programmability, allowing for greater flexibility and precision

In addition to the enhancements listed above, the TurboGX/TurboGXplus accelerators incorporates many other design changes. The memory subsystem (VRAM) access time has been reduced, a sense bus is now employed to autoselect the resolution and frequency for the attached monitor, a wider range of monitor support has been added, and substantial performance improvements have resulted from the new design. Table 2.4 illustrates some of the VRAM, resolution, and performance differences between the TurboGX/TurboGXplus and the GX/GXplus boards. The benchmarks were run on a SPARCstation™ 10 Model 40 running the Solaris™ 2.3 Operating System, the XGL 3.0.2 Graphics Library, and the Direct Xlib 3.0 Graphics Library. Adding to the existing 30+ GX patents currently in place, the TurboGX/TurboGXplus accelerators' enhancements have led to 7 more patents, covering new BLIT algorithms, memory system architecture features, raster operation support, VRAM block mode facilities, and a new asynchronous interface design.

| Description | GX | GXplus | TurboGX | TurboGXplus |
|---|---|---|---|---|
| **Resolution** | 1152x900 @66Hz 1152x900 @76Hz | 1280x1024@67Hz 1152x900@66Hz 1152x900@76Hz | 1024x768 @60Hz 1024x768 @77Hz 1152x900 @66Hz 1152x900 @76Hz | 1024x768@60Hz 1024x768@77Hz 1152x900@66Hz 1152x900@76Hz 1280x1024@67Hz 1280x1024@76Hz 1600x1280@76 Hz |
| **Framebuffer Memory** | 1-MB | 4-MB | 1-MB | 4-MB |
| **Buffering** | Single | Single or Double | Single | Single or Double |
| **Off-screen Memory** | 0 MB | 0.75 - 2.75 MBytes | 0 MB | 0.75 - 3.25 MBytes |
| **Performance** | | | | |
| **X11 - Dots** | 3,110K dots/s | 3,110K dots/s | 3,200K dots/s | 3,200K dots/s |
| **X11 - 10x10 Rectangles** | 277K rect/s | 277K rect/s | 429K rect/s | 581K rect/s |
| **X11 - 500x500 Rectangles** | 446K rect/s | 446K rect/s | 1620K rect/s | 4690K rect/s |
| **X11 - 10 Pixel Lines** | 400K lines/s | 409K lines/s | 762K lines/s | 1,500K lines/s |
| **X11 - 500x500 Scroll** | 107 | 107 | 163 | 384 |
| **X11 - 500x500 Copy Window** | 82.5 | 82.5 | 163 | 384 |
| **X11 - PutImage 500x500** | 61.0 | 61.0 | 61.5 | 61.5 |
| **GBench - 2-D Vectors/s** | 464K v/s | 464K v/s | 872K v/s | 975K/s |
| **GBench - 3-D Vector/s** | 357K v/s | 357K v/s | 405K v/s | 499K/s |
| **GBench - 3-D Triangles/s** | 107K tri/s | 107K tri/s | 156K tri/s | 232 K/s |
| **GPC - pc_board (literal)** | 27.7 | 27.7 | 32.1 | 37.0 |
| **GPC - pc_board (optimal)** | 32.1 | 32.1 | 38.0 | 43.9 |
| **GPC - floor_plan (literal)** | 27.6 | 27.6 | 30.5 | 36.2 |
| **GPC - floor_plan (optimall)** | 32.0 | 32.0 | 37.2 | 43.9 |
| **GPC - site_plan (literal)** | 21.7 | 21.7 | 25.8 | 30.7 |
| **GPC - site_plan (optimal)** | 22.3 | 22.3 | 25.0 | 32.0 |
| **GPC - sys_chassis (literal)** | 16.4 | 16.4 | 18.2 | 21.6 |
| **GPC - sys_chassis (optimal)** | 22.4 | 22.4 | 26.0 | 33.4 |
| **GPC - race_car (literal)** | 21.0 | 21.0 | 23.7 | 28.1 |
| **GPC - race_car (optimal)** | 24.7 | 24.7 | 28.3 | 34.9 |
| **PLB2D** | 26.9 | 26.9 | 31.0 | 36.9 |

*Table 2-4* Characteristics and Performance: TurboGX/TurboGXplus versus GX/GXplus Boards

# TurboGX/TurboGXplus Accelerator Architecture 3

## TurboGX/TurboGXplus Accelerator Hardware Design

The original GX accelerator was implemented primarily in two large and complex ASICs, the Frame Buffer Controller (FBC) and the Transformation Engine and Cursor (TEC). The FBC was responsible for rendering, and the TEC for transformations, hardware cursor, and video timing. External to the chips were bus interface, clock generation, and memory control circuitry. On the original SBus-based GX board, a good portion of this external logic was rolled into a third ASIC, the SBus Interface (SIF) chip. Also implemented in discrete logic was the data path for direct (unaccelerated) frame buffer accesses.

The second-generation GX implementation further integrated the GX architecture to combine the separate ASICs into a single chip. Additional resolution support was added, together with double buffering features (SPARCstation GXplus) and offscreen memory.

With the introduction of the TurboGX/TurboGXplus accelerators, a third-generation implementation of the GX architecture has been introduced in the form of a new highly integrated single-chip ASIC.

### Functional Blocks

Across the three major implementations of the GX architecture, the following functional blocks are present. In the original implementation, these represented individual ASICs or blocks of discrete logic. In the single chip GX and TurboGX/TurboGXplus accelerators, however, these blocks have been maintained as functional, identifiable sections within the device itself. What follows is a brief description of each of the major functional blocks of the TurboGX/TurboGXplus ASIC.

## *Frame Buffer Controller (FBC)*

The FBC is responsible for all rendering and accelerated frame buffer manipulation within the TurboGX/TurboGXplus architecture. It implements the three graphical commands (DRAW, BLIT, and FONT), which are utilized as the underpinnings of most other higher-level graphic operations. It supports a wide range of attributes that dictate how any given command will ultimately affect the frame buffer. A visual display of the three commands and the attributes that affect them can be seen in Figure 3.1. All rendering is always confined to a rectangular clipping window.

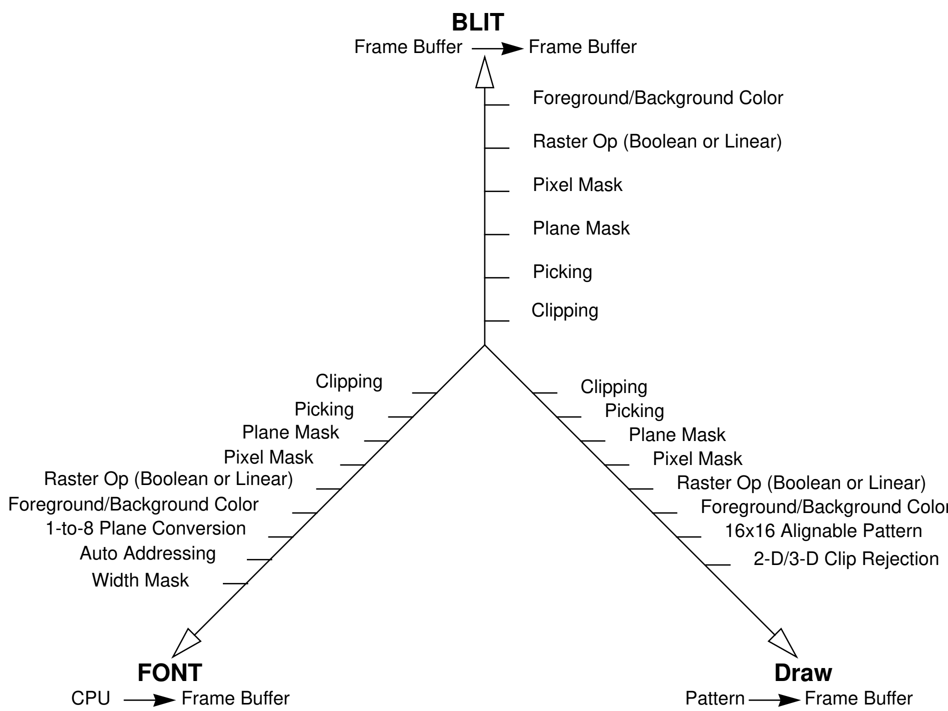


*Figure 3-1* The TurboGX/TurboGXplus ASIC Command Set

## Transformation and Cursor (TEC)

The TEC part of the TurboGX/TurboGXplus ASIC provides for three key parts of the overall architecture — vector math support, the hardware cursor, and a flexible, programmable video timing generator.

The TEC math unit is a 100-MFLOP vector processor. It supports the complete transformation pipeline typical of 2-D/3-D computer graphic needs: modeling and viewing (MV) transforms, perspective divide, 2-D/3-D clip checking, as well as a final virtual device coordinate (VDC) transform. A more general-purpose vector math mode allows for usage in matrix concatenation, dot products, and lighting calculations.

A three-color, 32-pixel, hardware cursor is used to overlay a cursor image non-destructively on top of the 8-bit frame buffer.

The flexible, programmable video timing generator gives the TurboGX/TurboGXplus accelerators the ability to adapt to and accommodate many different monitors, video rates, and resolutions.

## SBus Interface (SIF)

The SBus interface component of the TurboGX/TurboGXplus chip implements an SBus slave interface for interconnecting the accelerator to the workstation's primary I/O bus. The accelerated portions of the TurboGX/TurboGXplus ASIC (the FBC/TEC blocks) support 32-bit accesses. The direct frame buffer memory access path supports 8-, 16-, and 32-bit aligned data accesses. Also, as all SBus peripherals are required to do, this interface works over the frequency range of 16 MHz through 25 MHz.

The SBus Interface logic is integral to the single chip TurboGX/TurboGXplus ASIC but is implemented in the SIF ASIC on older GX implementations. The SIF-based designs run the bulk of the internal TurboGX/TurboGXplus accelerators functionality synchronously with the SBus's clock, while the single-chip TurboGX/TurboGXplus ASIC internal clock rates are asynchronous and independent of the SBus clock rates.

## VRAM Memory Controller

The TurboGX/TurboGXplus VRAM controller handles all aspects of dual-ported VRAM control, including all memory handshakes (e.g., RAS/CAS), transfer cycles (a VRAM-specific memory cycle), and refresh. This controller also orchestrates both accelerated and non-accelerated frame buffer accesses. It supports page-mode accessing (a high-performance VRAM access mode) whenever possible.

Another special VRAM feature utilized is a “write-per-bit” data-write cycle. This is used to implement a graphics capability known as plane masking. This feature permits selective accessing (or masking from modification) of each of the 8 bits that comprise a pixel. The TurboGX/TurboGXplus ASIC stores the actual plane mask value within the VRAM itself. The VRAM controller is responsible for the management of this feature and for ensuring its integrity across context switches and access types.

The VRAM controller is integral to the single-chip TurboGX/TurboGXplus ASIC and was implemented in the SIF ASIC on older GX implementations. The older SIF-based designs were intended for 256 K and 1 Mb VRAM support, while the single-chip TurboGX/TurboGXplus design is intended for 1 Mb and 2 Mb VRAM.

# TurboGX/TurboGXplus Accelerator Software

4

## TurboGX/TurboGXplus Accelerator Software Support

The TurboGX/TurboGXplus accelerators support all Sun graphics APIs, including the SunPHIGS Graphics Library, the XGL Graphics Library, the XIL Imaging Library, the PostScript© Library, the Direct Xlib Graphics Library, and MIT's Xlib, under the Solaris OpenWindows environment. Additionally, TurboGX/TurboGXplus systems support any 3rd-party graphics API that reside above the foundation-level APIs. Examples of such APIs include Ithaca's HOOPS™, Nth GL™, Liant's Figaro PHIGS™, Gallium Software (Prior) GKS™, ATC GRAFPAK GKS™ and many others. Figure 4.1 illustrates these graphics APIs. Please refer to Sun and 3rd-party product literature for specific details.

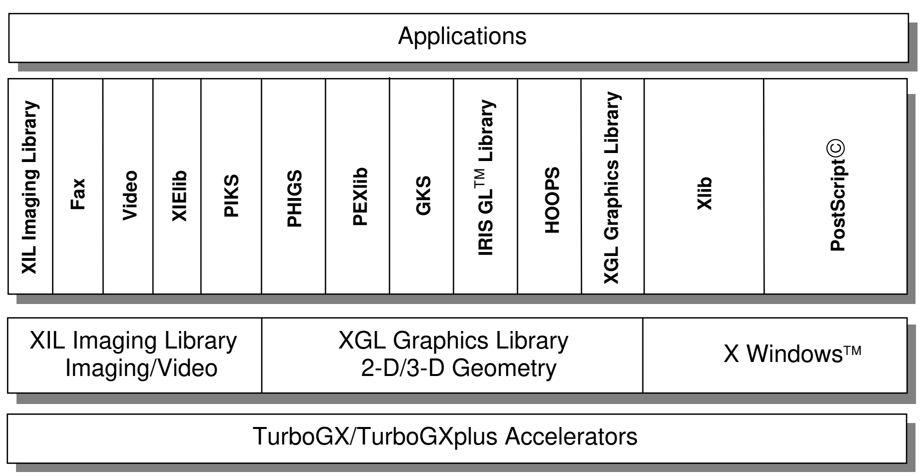


*Figure 4-1* TurboGX/TurboGXplus Accelerators Graphics API Support

# Software-Accessible Hardware Features

Because of the unique architecture of the TurboGX/TurboGXplus accelerators, special capabilities are accessible through Sun’s graphics and window system environments. The following sections describe these features and how to access them.

## TurboGXplus Off-screen Memory

Only a portion of TurboGXplus video RAM is used to generate the visible display. The remaining memory, “off-screen memory,” is used for double buffering and for storage of image data. Depending on resolution and whether double buffering is employed, the available video memory enables the TurboGXplus accelerator to support off-screen memory. Table 2.2 illustrates available off-screen memory for the TurboGXplus configurations.

Window systems often need to render into temporary pixel areas to prepare such things as menus and buttons. Many window systems need to periodically save and restore pixels from the screen, for example, when a menu comes up over one or more windows on the screen. Some also support saving and restoring the pixels from windows that become overlapped by other windows. All of these operations require memory, usually system memory, where the window system can allocate space for pixel areas and can render into those areas.

On the TurboGXplus accelerator, significant off-screen memory is available for allocation and storage and has the added advantage that the graphics accelerator can draw directly into this memory, thus providing high-performance rendering power.

The OpenWindows environment has been designed to take advantage of this extra off-screen memory when it is not being used for double buffering. It will make use of and manage this memory in a manner that is transparent to application programs, thereby allowing any existing or newly designed application program to benefit when it performs rendering operations on off-screen 8-bit pixel areas. The window system allocates this memory on a first-come, first-served basis. It automatically attempts to allocate any 8-bit pixmaps, retained windows, or save under areas first in the off-screen frame buffer memory, and only if this fails will it allocate the memory in system memory.

The OpenWindows environment gives priority for the off-screen memory to multi-buffering. Whenever an application starts and attempts to use the multi-buffering features of the board, the window system automatically moves all pixel areas currently in off-screen frame buffer memory transparently back into system memory. To avoid frequent delays due to pixel copying operations, after pixel areas have been moved out of off-screen pixel memory, they will only return if they get reallocated. Reallocation may occur when a window is resized, for example.

Rendering into any pixel area in off-screen memory provides a performance gain over the CPU's main memory due to the TurboGXplus accelerator's speed. Operations on pixmaps stored in off-screen memory are done at the speed of screen-to-screen operations.

The `realxfishdb` demo program is an example of an application which gains significant performance transparently from this feature in the OpenWindows environment on the TurboGXplus accelerator. `realxfishdb` is included in binary form in the OpenWindows demo directory. This is a standard X11 application program that makes extensive use of 8-bit pixmaps and pixel copy operations.

## *X Performance Enhancements with Off-screen Memory*

Off-screen memory on the TurboGXplus accelerator provides faster performance for X copy operations. Such copy operations amortize the server overhead over many pixels, and in those cases, significant performance can be seen. Of particular note is a 500x500 pixel window to pixmap case. With the TurboGXplus accelerator, the CPU avoids frame buffer reads or writes. Since the windows and pixmaps are all in the TurboGXplus accelerator's memory (visible and off-screen, respectively), the TurboGXplus accelerator's hardware does the copy.

## *Multi-buffering through Sun's Graphics API's*

The TurboGX/TurboGXplus accelerators provide hardware support for the 4-bit double buffering known as colormap or software double buffering that all GX products support. The TurboGX/TurboGXplus accelerators' double buffering can be accessed through the X Multi-Buffering Extension in the OpenWindows environment, through Sun's XGL and SunPHIGS graphics libraries. Each application programming interface (API) allows an application to render into an invisible buffer and causes it to be displayed entirely at once without update artifacts such as visible screen clears or video tearing.

### *X11 Multi-Buffering*

The X Multi-Buffering Extension is a draft standard of the MIT X Consortium and is described in *Extending X for Double-Buffering, Multi-Buffering, and Stereo* by Jeffrey Friedberg, Larry Seiler and Jeff Vroom, 1990. Multi-buffering is a generalization of double buffering to permit more than two buffers, and is possibly useful in animation loops. The OpenWindows environment provides multi-buffering functionality for all frame buffers; however the implementation is accelerated on the TurboGXplus accelerators. The example X multi-buffering code fragment illustrated in Figure 4.2 outlines a simple case for smooth animation using the TurboGXplus' hardware double buffering capability.

```
#include <X11/extensions/multibuf.h>
<Open display, create window, allocate colors>
/* query to see if extension exists */
if (!XmbufQueryExtension (display, &mbevbase, &mberrbase)) {
    printf("Multibuffering extension not supported in
server.\n");
    exit(1);
}
/*  Create multi buffers. Note that while all multi-buffering
update frequency
 *  hints and update actions are supported in OpenWindows
Version 3,
 *  hardware multi-buffering support is only available for
update frequency hint
 *  MultibufferUpdateHintFrequent and update actions
 *  MultibufferUpdateActionUndefined or
 *  MultibufferUpdateActionBackground.
 */
nbufs = XmbufCreateBuffers(display, window, 2,
MultibufferUpdateActionBackground,
    MultibufferUpdateHintFrequent,buffers);

if (nbufs != 2) {
    printf("\nCouldn't create 2 buffers.");
    exit(1);
}
...
/* Animation loop */
<Calculate new scene>
XFillRectangle(display, buffers[draw_buffnum],
gc,x,y,width,height);
XmbufDisplayBuffers(display, 1, &buffers[draw_buffnum], 0, 0);
draw_buffnum = 1 - draw_buffnum;
```

*Figure 4-2* X11 Window System Multi-buffering Code Example

## XGL Double Buffering

Double buffering on the TurboGXplus accelerator through the XGL graphics library is enabled by requesting two buffers for a window and verifying that two were allocated. The XGL library allows an application to explicitly set the buffer displayed and the buffer in which to draw, as well as a convenient implicit technique. With the implicit technique, an application tells the XGL library (through a context attribute) that it is to switch buffers on a “context new frame” command. The application may in addition opt to have the XGL library clear the non-displayed buffer when the command is given. Figure 4.3 illustrates the use of the XGL library for double buffering.

```
<Create your XView frame, canvas>
<Open xgl, create raster, context, colormap>

/* Double buffering setup */
xgl_object_get(ctx, XGL_CTX_DEVICE, &wras);
if (!wras) {
    printf(“no window raster in context\n”);
    return;
}
xgl_object_set(wras, XGL_WIN_RAS_BUFFERS_REQUESTED, 2,
0);
xgl_object_get(wras,XGL_WIN_RAS_BUFFERS_ALLOCATED,&bufs)
;
if (bufs !=2) {
    printf(“no hardware double buffering\n”);
}
xgl_object_set(wras,
    XGL_WIN_RAS_BUF_DISPLAY, 0, /* display default front
buffer */
    XGL_WIN_RAS_BUF_DRAW, 1, /* draw into default back
buffer */
     0);
xgl_object_set(ctx,
    XGL_CTX_NEW_FRAME_ACTION,
    XGL_CTX_NEW_FRAME_SWITCH_BUFFER |
    XGL_CTX_NEW_FRAME_CLEAR,
    0);

...

/* Animation loop (itimer function called from notifier)
*/
<Calculate new scene>
xgl_polygon(ctx, XGL_FACET_NONE,NULL,NULL,1,&pl_2d);
xgl_context_new_frame (ctx);
```

*Figure 4-3* XGL Graphics Library Multi-buffering Code Example

## SunPHIGS Double Buffering

With the SunPHIGS graphics interface, an application creates a double-buffered workstation type and opens a workstation of that type to enable double buffering. The SunPHIGS graphics library switches buffers and clears the buffer removed from display when an `UPDATE WORKSTATION` command is given by the application. Figure 4.4 illustrates an example SunPHIGS application code fragment for double buffering.

```
<Create your XView frame, canvas>
<Open phigs>

/*
 *  Tell SunPHIGS that we want a hardware double buffered
PHIGS
 *  workstation by creating a ws_type with that attribute
 */

ws_type = phigs_ws_type_create(phigs_ws_type_x_drawable,
    PHIGS_X_BUF_MODE, PHIGS_BUF_DOUBLE,
    0);
popen_ws(ws1,(Pconnid)&xconnid,ws_type);

/* Did we get a hw dbl buffered workstation? */
pinq_ws_conn_type (ws1, 0, &err, &connid, &p_wst);
escin.escape_in_u16.ws_type = p_wst;
pcreate_store (&err, &store);
pescape (PUESC_INQ_WS_TYPE_SUPPORT, &escin, store,
&escout);
if
(escout.escape_out_u16.ws_type_sup.out_sup.double_buffer
    != PHIGS_ACCELERATED) {
    printf("no hardware double buffering\n");
}
/* Tell PHIGS to only update when we ask (does ASAP by
default) */
pset_disp_upd_st(ws1,PDEFER_WAIT,PMODE_UWOR);
...
/* Animation loop (itimer function called from notifier)
*/
<Calculate new scene>
<Create, edit and post structures, update transforms>
pupd_ws(1,PUPD_PERFORM);
```

*Figure 4-4* SunPHIGS Graphics Library Multi-buffering Code Example

## The Direct Xlib Graphics Library

The Direct Xlib graphics library is a supplemental product that transparently enhances graphics rendering performance for graphics-intensive Xlib applications in the OpenWindows environment. The Direct Xlib library was developed to provide an optimized Xlib interface for applications that typically run locally (client application and X Windows server on the same machine).

The Direct Xlib library allows most X Windows graphics rendering operations to occur directly in the client application process. This serves to avoid the overhead associated with the copying of client application data to the X Windows request buffer, client/server context switching, and server parsing and processing of the request buffer for these operations. The key to acceleration with the Direct Xlib library is the use of Sun's Direct Graphics Access (DGA) technology, which provides a synchronization mechanism between the Direct Xlib library and the OpenWindows server. Other graphics interfaces supported on the TurboGX/TurboGXplus accelerator also employ DGA, including the XGL graphics library, the XIL imaging library, and the SunPHIGS graphics library (through the XGL interface). As with these other interfaces, any third-party API layered above Direct Xlib will automatically utilize DGA. The use of Direct Xlib requires no alterations to the client application code. The Direct Xlib library automatically manages any synchronization necessary with the OpenWindows server. Figure 4.5 illustrates the Direct Xlib primitives accelerated by the TurboGX/TurboGXplus accelerators.

| Direct Xlib Primitive |
|---|
| XDrawPoint, XDrawPoints |
| XDrawLine, XDrawLines, XDrawSegments |
| XDrawRectangle, XDrawRectangles |
| XFillRectangle, XFillRectangles |
| XFillPolygon |
| XDrawString, XDrawImageString, XDrawString16, XDrawImageString16 |
| XDrawText, XDrawText16 |
| XPutImage, XGetImage, XCopyArea |
| XmbDrawString |
| XmbDrawImageString |
| XwcDrawString |
| XwcDrawImageString |
| XmbDrawText |
| XwcDrawText |
| XFillArc, XFillArcs |
| XDrawArc, XDrawArcs |
| XClearWindow |
| XClearArea |

*Figure 4-5* Direct Xlib Primitives Accelerated by the TurboGX/TurboGXplus Accelerators

The principal benefit in using the Direct Xlib library is rendering performance. In general, most applications experience two times the rendering performance using the Direct Xlib library versus Xlib. Use of the standard Xlib library is appropriate when graphics performance is not a factor, and when the application must be statically linked.

While the Direct Xlib library does not provide any acceleration to Xlib library functions that require interaction with the server (such as input), key graphics rendering operations may be substantially accelerated, depending on the underlying accelerator capabilities and the architecture of the client application. In general, performance improvements occur where the overhead of copying geometry data and/or pixels is great enough to outweigh the time it takes to render these primitives on the particular graphics device in use.

# References


*Solaris XGL Reference Manual,* SunSoft Inc.
*Solaris XGL Programmer's Manual*, SunSoft Inc.

*Solaris XIL Imaging Library Reference Manual,* SunSoft Inc.
*Solaris XIL Imaging Library Programmer's Guide,* SunSoft Inc.

*SunPHIGS Reference Manual*, Sun Microsystems Computer Company
*SunPHIGS Extensions Reference Manual*, Sun Microsystems Computer Company
*SunPHIGS* Porting Guide, Sun Microsystems Computer Company

*Direct Xlib 2.0 Users Manual*, Sun Microsystems Computer Company

*OpenWindows User's Guide,* SunSoft Inc.
*OpenWindows Programmers Guide,* SunSoft Inc.

*X Window System User's Guide*, O'Reilly & Associates. Inc.
*Xlib Reference Manual*, O'Reilly & Associates. Inc.
*Xlib Programming Manua*l, O'Reilly & Associates. Inc.

*X Protocol Reference Manual,* O'Reilly & Associates. Inc
*PEX Protocol Specification Version 5.0P,* MIT

*PEXlib Specification and C Language Binding,* Hewlett-Packard Company, Version 5.1

*PEXlib Programming Manual*, O'Reilly & Associates. Inc., 1992
*X Image Extension Protocol Reference Manual*, MIT X Consortium, Public Review Draft Version 4.1.2

*The Display PostScript System*, Adobe Systems Inc., 1990
*An Introduction to Computer Graphics Concepts*, Addison-Wesley
*The Renderman Companion*, Upstill, Addison-Wesley

Sun Microsystems
Computer Corporation
A Sun Microsystems, Inc. Business

For U.S. Sales Office locations, call: 800 821-4643
In California: 800 821-4642

Australia: (02) 844 5000
Belgium: +32 2 730 38 11
Brazil: 011-55-11-887-9011
Canada: 416 477-6745
Caribbean: +1 415 688-9464
Finland: +358-0-502 27 00

France: (1) 30 67 50 00
Germany: (0) 89-46 00 8-0
Hong Kong: 852 802 4188
Italy: 039 60551
Japan: (03) 3221-7021
Korea: 822-563-8700

Latin America: 415 688-9464
Mexico: 001-525-580-5229
The Netherlands: 033 501234
New Zealand: (04) 499 2344
Nordic Countries: +46 (0) 8 623 90 00
PRC: 861-831-5568

Singapore: 224 3388
Spain: (91) 5551648
Switzerland: (01) 825 71 11
Taiwan: 2-514-0567
UK: 0276 20444
Elsewhere in the world, call
Corporate Headquarters:
415 960-1300
Intercontinental Sales:
415 688-9000

Printed in USA