ASUS® 3DP-V500TX

User's Manual

USER'S NOTICE

The product name and revision number are both printed on the board itself. Manual revisions are released for each design represented by the digit before and after the period and for manual updates represented by the third digit in the manual revision number. For updated BIOS, drivers, or product release information you may visit ASUSTeK's home page at: http://www.asus.com.tw/

Product Name:	**ASUS 3DP-V500TX**
Manual Revison:	**1.00**
Manual Release:	**October 1996**

ASUS CONTACT INFORMATION

ASUSTeK COMPUTER INC.

Marketing Info:

Address: 150 Li-Te Road, Peitou, Taipei, Taiwan, ROC
Telephone: 886-2-894-3447
Fax: 886-2-894-3449
Email: info@asus.com.tw

Technical Support:

Fax: 886-2-895-9254
BBS: 886-2-896-4667
Email: tsd@asus.com.tw
WWW: http://www.asus.com.tw/
Gopher: gopher.asus.com.tw
FTP: ftp.asus.com.tw/pub/ASUS

ASUS COMPUTER INTERNATIONAL

Marketing Info:

Address: 721 Charcot Avenue, San Jose, CA 95131, USA
Telephone: 1-408-474-0567
Fax: 1-408-474-0568
Email: info-usa@asus.com.tw

Technical Support:

BBS: 1-408-474-0555
Email: tsd-usa@asus.com.tw

ASUS COMPUTER GmbH

Marketing Info:

Address: Harkort Str. 25, 40880 Ratingen, BRD, Germany
Telephone: 49-2102-445011
Fax: 49-2102-442066
Email: info-ger@asus.com.tw

Technical Support:

BBS: 49-2102-448690
Email: tsd-ger@asus.com.tw

CONTENTS

CONTENTS

FCC & DOC COMPLIANCE

Federal Communications Commission Statement

This device complies with FCC Rules Part 15. Operation is subject to the following two conditions:

- This device may not cause harmful interference, and
- This device must accept any interference received, including interference that may cause undesired operation.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with manufacturer's instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Re-orient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment to an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

WARNING: The use of shielded cables for connection of the monitor to the graphics card is required to assure compliance with FCC regulations. Changes or modifications to this unit not expressly approved by the party responsible for compliance could void the user's authority to operate this equipment.

Canadian Department of Communications Statement

This digital apparatus does not exceed the Class B limits for radio noise emissions from digital apparatus set out in the Radio Interference Regulations of the Canadian Department of Communications.

I. INTRODUCTION

Features

The ASUS 3DP-V500TX boards have been designed for high performance professional 3D graphics acceleration. This device combines workstation class 3D graphics acceleration and state of the art 2D performance. All 3D rendering operations are accelerated by 3Dlabs's Glint500TX chip including Gouraud shading, texture mapping, depth buffering, antialiasing and alpha blending. The addition of Delta chip enhances performance by off-loading the host of both the rendering and setup calculations, whilst also reducing the PCI bandwidth requirements.

Based on the powerful ASUS 3DP-V500TX plus Delta combination, this boards provides the ultimate 3D system performance for the power user. The ASUS 3DP-V500TX provides greatly enhanced hardware support for texture mapping and significant performance improvements.

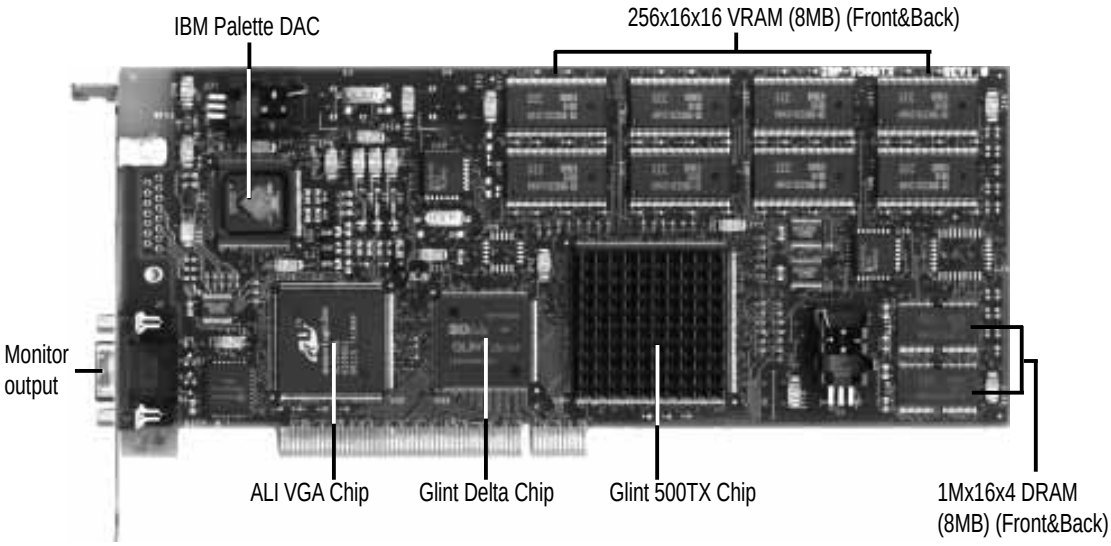
System Capabilities

- Onboard SVGA
- 8Mbytes VRAM and 8Mbytes DRAM

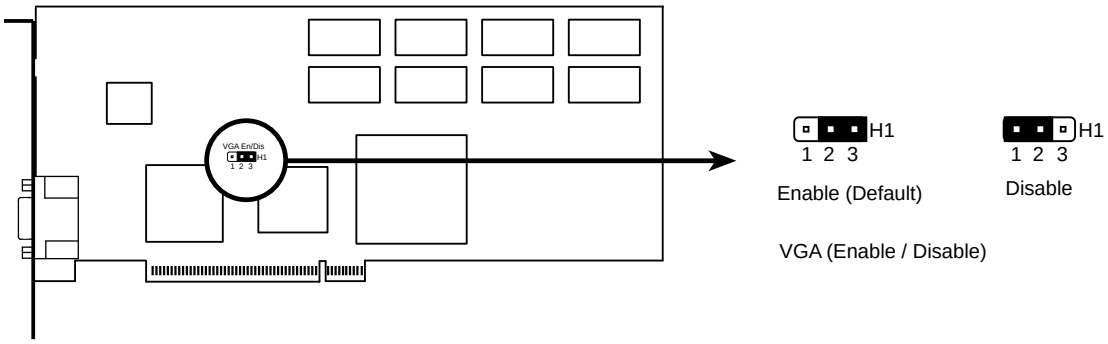
System Requirements

- Suitable for use in computers supporting the PCI bus e.g. IBM AT compatible computers with 80486, Pentium or Pentium Pro CPUs
- One available PCI slot
- VGA, XGA, SuperVGA or Extended VGA compatible monitor

Parts of the ASUS Card



Jumper Settings



VGA (Enabled)
The onboard SVGA chip can be Enabled or Disabled.

II. HARDWARE INSTALLATION

Introduction

The board will have been configured for the options installed at the factory. Before installing the board, please check that the jumper settings are correct and that you have taken note of the static electricity precautions.

Static Electricity Precautions

1. ASUS 3DP-V500TX boards are supplied with disposable grounding wrist straps. Follow the supplied instructions.
2. Alternatively, before handling any components or touching anything inside the system unit, discharge your body static electric charge by holding onto a grounding surface. If the system unit is connected to a grounded outlet, you can touch a suitable part of the system metal chassis.
3. Do not remove boards from their antistatic bags until you are ready to install them.
4. When handling boards, hold them by their edges and their metal mounting brackets. Avoid touching components on the board and the edge connectors that plug into the expansion slots.
5. Avoid plastic, vinyl and styrofoam in your work area.

II. HARDWARE INSTALLATION

Installing the ASUS 3DP-V500TX Board

Installing the ASUS 3DP-V500TX board in your system is simple, so please follow the instructions carefully. To install the board:

1. Switch off your system and all peripheral devices.
2. Follow the static precautions described above.
3. Remove the cover from your system.
4. Remove any existing video boards installed in the system. To do this, detach the monitor cable from an existing video board, remove the screw that holds the board in place and then pull it out from its expansion slot. Store the old video board in an antistatic bag.
5. Find a free PCI expansion slot in your system.
6. Remove the metal cover plate from the slot you have chosen and put the screw to one side.
7. Ensure the onboard jumpers are correctly set on your ASUS 3DP-V500TX board (see earlier chapter).
8. Align the board PCI slot connector with the expansion slot in our PC and gently lower and push the board into the free slot.
9. Secure the board to the expansion slot with the screw you removed from the metal plate.
10. Replace the cover on your system and plug in the power cord.
11. Reattach the video cable to the devices video output connector.
12. Turn on the monitor and power up the system.

Installing a Second ASUS 3DP-V500TX Board

A second Board can be installed in a similar manner to the first. The jumper settings on the second board should normally be set to be the same* as those on the first. Only one board will act as the VGA even if the jumpers on both boards are set to enable VGA. To obtain a picture during boot up, it may therefore be necessary to connect the video cable to the second 3DP-V500TX rather than the first.

*(In some systems, the BIOS can be confused by having VGA enabled on both boards. When this occurs, the VGA jumper should be fitted on one Board to disable VGA. However, it is then necessary that the Board with enabled VGA is positioned in a PCI slot so that it is recognized before the VGA disabled Board. Determining the correct slot is a nontrivial task so this is best resolved by trial and error.)

III. SOFTWARE INSTALLATION

Windows NT Driver Installation

This chapter describes the Windows NT Display Driver and OpenGL Installable Client Driver and Heidi for the ASUS 3DP-V500TX Board.

This document should be read in conjunction with the README.TXT file on the installation floppy, which contains details of any enhancements and/or bug fixes that have been made subsequent to these release notes being written.

Prerequisites

- Windows NT 3.5 (Build 807 or later), Windows NT 3.51 (Build 1057), Windows NT 4.0 (Build 1381)
- Intel 486 processor or later, MIPS or DEC Alpha.
- ASUS 3DP-V500TX Board.

Before installing the software, power down the machine and install the ASUS 3DP-V500TX as the hardware installation instructions. Boot the machine using the non-VGA boot option (new display drivers cannot be installed when the machine has been booted with the VGA boot option). Once booted and you have logged in as an Administrator, perform the following steps:

NT 3.51 (Intel based PC workstation)

1. Open the Control Panel in the Main Program Group and start the Display Applet.
2. Press the "**Change Display Type**..." button. A new window titled "Display Type" will appear.
3. Press the "**Change**..." button in this window. A window titled "Select Drive" will appear.
4. Press the "**Other**..." button in this window. A window titled "Install from Disk" will appear.
5. Specify the path "A:\" Insert the release NT3.51 driver floppy into the drive and click **"Ok"** button.
6. A number of different options for different resolutions, depths and monitor frequencies are supplied. If you know the option you want and you are sure that the monitor supports this option then choose it. If not or you are unsure about the capability of your monitor choose a 640x480 option at 60Hz with a pixel depth of 12. When the machine has rebooted you will be able to select a new option and test the monitor capabilities at this resolution and frequency.
7. Part of the description of each option indicates the 3D double buffering capabilities of the card at that depth and resolution. "Double Buffered" means that multiple double buffered 3D applications can be supported. "1 Double Buffered Window" means that only one window can be double buffered at a given time. Attempting to run a second double buffered application will fail until the first one has exited.
8. A window may appear asking you to confirm that you have been changing. Click "**Yes**".
9. Press "**Continue**" in the "Windows NT Setup" window.
10. Two more information windows will appear. Press "**Ok**" in both.
11. If you are upgrading your drivers you may be asked whether you want to use the "Current" or "New" drivers. Select the "**New**" option.
12. A window titled "Display Settings Change" will appear. Remove the driver floppy from the drive and press the "**Restart Now**" option.

III. SOFTWARE INSTALLATION

NT 4.0 (Intel based workstation):

- Open the Control Panel in the Main Program Group and start the Display Applet.
- Press the "**Change Display** ..." button. A new window titled "Display Type" will appear.
- Press the "**Change**..." button in this window. A window titled "Change Display" will appear.
- Press the "**Have Disk** ..." button in this window. A window titled "Install from Disk" will appear.
- Specify the path A:\. Insert the driver floppy for your machine architecture into the drive and press "**Ok**". The "Change Display" window will appear with two sb-windows. Select the chip type in the left hand sub-window and the nearest board type in the right hand sub-window and press "**Ok**".
- Then follow the instructions and quit the control panel applet. When asked if you want to restart the machine press "**Yes**".

Note that for NT 4.0 there are no options to select a given resolution at install time. When the machine reboots NT 4.0 allows the video mode to be dynamically changed without the need for a reboot.

The machine will now shutdown. On restart again choose the non-VGA boot option. It will restart using the ASUS 3DP-V500TX Board as the display device. This can be checked by opening the "Display" applet again and pressing the "Change Display Type..." button. The "Display Type" window should report that it is running on a ASUS 3DP-V500TX.

If the desired resolution, depth and frequency have not been chosen at install time then open the Display Applet to define the required resolution, color depth and monitor frequency. This selected mode can be tested to ensure that it can be handled by the monitor. For running the default 3D demos a resolution of 1024x768 with 4096 colors is recommended. Selecting 75Hz is desirable if the monitor can support this frequency. On some double buffered applications the higher refresh rate allows higher frame rates to be achieved. For NT 4.0 the display will change dynamically; for NT 3.51 a reboot is necessary.

The above procedure installs the NT display driver, control panel applet and the OpenGL installable client driver. Once the display resolution and pixel depth have been appropriately re-configured the machine is ready to run both Windows NT and OpenGL applications and demos.

Heidi

To install the latest version of your GLINT Heidi driver, locate your 3DStudio Max directory on your hard drive and then enter the Drivers sub-directory. Rename the file in there "**wglint.hdi**" to "**wglint.bak**". Then copy the new "**wglint.hdi**" from the floppy diskette to the Drivers directory. You can now run 3DStudio Max, and the new driver will be used.

3D Graphics & Double Buffering

The display driver contains an extension to allow 3D applications, and the OpenGL installable client driver, to drive the hardware. To provide a double buffering capability for these 3D applications the display driver provides the following features.

A screen-sized off-screen buffer is configured if the "DoubleBuffer.NumberOfBuffers" registry variable is greater or equal to 2 (see below). This buffer is used in 256, 4096, 32768 and True Color modes to provide BitBlt double buffering. The off-screen buffer is also used to provide full screen hardware double buffering if an application window covers the whole screen. For double buffering to be available, the chosen resolution and pixel depth must fit into 2MB. This restriction applies to ASUS 3DP-V500TX boards. The resolution tables below define which pixel depth and resolution combinations provide a double buffering capability.

Resolution and Color Depth Table

# of Colors	Screen Resolution	# of Buffers	Color Space 2x Buffering	Blitted 2x Buffering	# of H/W 2x Buffering Win
4,096	800x600	2	Yes	Yes	Many
4,096	1024x768	2	Yes	Yes	Many
4,096	1152x870	2	Yes	Yes	Many
32,768	1024x768	2	No	Yes	Many
32,768	1152X870	2	No	Yes	Many
32,768	1280x1024	2	No	Yes	Many
32,768	1600x1200	2	No	Yes	Many
True	800x600	2	No	Yes	Many
True	1024x768	2	No	Yes	Many
True	1152X870	2	No	Yes	Many

1600x1200 resolutions are available at 60Hz. The rest of the above modes are available at both 60Hz and 75Hz monitor refresh rates. In addition, the 640x480 resolution is available at 72Hz. A full list of all modes is available via the display applet once the driver has been installed and the system rebooted. Choose the list all Modes" option to get this list.

Currently only double buffered resolutions are selectable using the Display Applet, unless a registry variable is set to override this (see below). This is to avoid confusion where double buffered applications run through software rendering only because of insufficient VRAM available to support accelerated double buffering.

Full Screen Double Buffering

If an application window covers the whole screen, the display driver will automatically switch to use a hardware double buffer mechanism, which can have a significant performance benefit. This mechanism will not be available to an application that has more than a small window border at the top of the screen. It will also be unavailable if, for example, a floating task bar (common on NT 4.0) is at any edge other than the top of the screen, since the display driver will check and find that the application window does not cover the whole screen.

Color Space Double Buffering

If 4096 colors mode is chosen then hardware double buffering is implemented using interleaved nibbles. Each pixel is 32 bits deep with the front buffer occupying bits 0x0F0F0F0F and the back buffer occupying bits 0xF0F0F0F0. Switching between these two buffers is done using the Palette DAC readmask. This is known as Color Space Double Buffering.

In this mode, standard Windows applications see the display as providing 12 bits per pixel true color mode with 4 bits per color component. GDI rendering operations are replicated into both the high and low nibbles. To select this mode choose one of the 4096 color modes from the “List Of Modes” menu in the Display Applet.

Not all 3D Graphics drivers for taking advantage of this mode, though the OpenGL Installable Client Driver does.

Color Space Double Buffering and OpenGL

The OpenGL Installable Client Driver differentiates between the two buffers by setting the writemask appropriately. Swapping the buffers over in this instance does not incur the cost of a blitting operation - the OpenGL buffer swaps over and all the GDI rendering remains intact. This mode therefore provides the optimal double buffering performance since hardware double buffering is available for all window sizes.

Because the swap operates on the whole screen, there can only be one double buffered OpenGL window that uses this technique. Thus the first 3D window created can be color space double buffered, while all subsequent 3D windows must use BitBlt double buffering (assuming there is sufficient VRAM to support another complete screen sized buffer). If the first window closes, the next 3D window created will use color space double buffering.

Dual-Headed Displays

This Release supports dual-headed displays using two ASUS 3DP-V500TX boards. The configurations for each board must be identical. In particular they must have the same amount of VRAM to run dual-headed simply plug in a second board. The driver will automatically detect the presence of the second board and run dual-headed. To switch back to single headed operation, remove the second board.

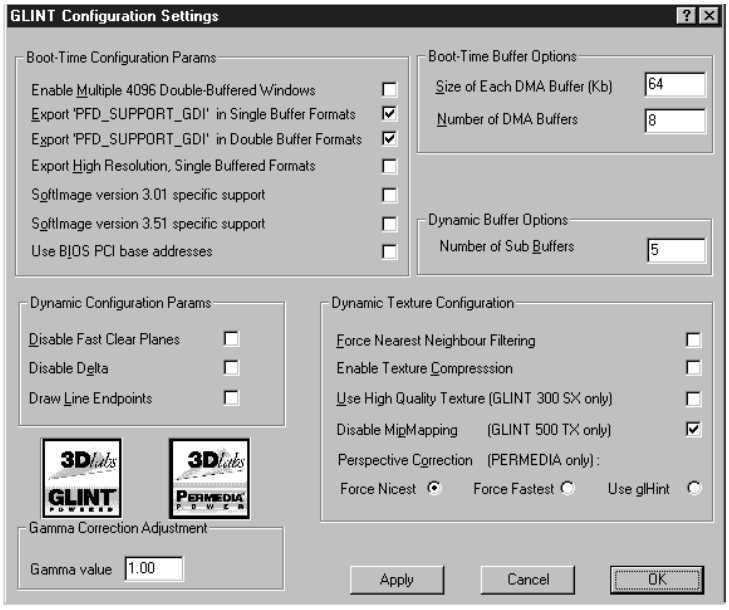
The ordering of the boards depends on the PCI slots they occupy. On startup the Windows logo will be displayed halfway between both boards. If the left and right displays are swapped, simply swap the monitor cables connected to both boards.

Note: In this release dual-headed support on NT 4 is not included, only available for NT3.51

Control Panel Applet

Some of the registry variables, detailed in the next section can be directly changed by the GLINT control panel applet, accessible through the control panel. This applet allows both boot-time and run-time control over the configuration of OpenGL and other applications using the GLINT display driver. Options that are not applicable to the currently installed graphics board will be disabled (greyed out). The control panel is split into a number of groups as listed below.

Note, currently it is necessary to have Administrator privileges to run the control panel applet. This is because it allows updates to the registry which are only allowable to a user with administrator privileges.



Registry Variables

Enable Multiple 4096 Double-Buffered Windows
When in 4096 colour mode, there can only be one double buffered OpenGL window. This check-box allows multiple double buffered windows to coexist in this mode simultaneously, which can be useful for particular applications. Note though that any buffer swap operation will cause a buffer swap to happen for all other double buffered windows.

Export PFD_SUPPORT_GDI modes
These two check-boxes control whether the PFD_SUPPORT_GDI flags are exported for single buffered and double buffered pixel formats. The check boxes alter the registry variables:

```
3DExtensions.SupportSingle
3DExtensions.SupportDouble
```

Export High Resolution, Single Buffered Formats
When this box is checked, it will enable the driver to boot at resolutions where only single buffered pixel formats are supported by GLINT acceleration (because at higher resolutions, there is not enough VRAM to support double buffered formats). By default, this is not enabled and it prevents users from booting into a mode that will result in unaccelerated OpenGL applications that use double buffered modes.

SoftImage Version 3.01 Application support
Version 3.01 of SoftImage requires this to be set to ensure the correct operation of SoftImage on the 3DP-V500 TX board. This option requires a reboot to take effect.

SoftImage Version 3.51 Application support
This box need to be checked to correctly run Softimage version 3.51. The option is mutually exclusive with support for version 3.01 which will be disabled when this is selected. This option requires a reboot to take effect.

Use BIOS PCI base addresses
Normally, the NT HAL allocated PCI base addresses for the GLINT card. These override those which are normally supplied by the PCI BIOS. On some machine configurations these addresses are not valid for the given hardware. In these instances, the base addresses originally configured by the BIOS are often valid and allow the machine to boot. Setting this variable to 1 causes the BIOS base addresses to be used rather than those configured by NT.

In particular, on machines which use the Intel Multi Processor Specification 1.4 (MPS 1.4), there is a bug in the NT HAL for many machines which causes invalid base addresses to be configured. Setting this variable may fix this problem.

Boot-Time Buffer size Options
These boxes can be used to specify the size (in Kb) of a DMA buffer along with the number of DMA buffers allocated at boot time. Ideally, you would like enough DMA buffers to cater for all of the OpenGL contexts, however each of these buffers will use up system memory. For more information, see the notes on the GlintDMA.NumberOfBuffers and GlintDMA.SizeOfBuffer registry variables.

Dynamic Buffer Option - Number of Sub-Buffers
Each DMA buffer is subdivided into sub-buffers which are used in conjunction with an Interrupt DMA mechanism to reduce latency in the system. The number of sub-buffers can be set here, setting it to 2 will disable the interrupt mechanism. For more information, see the notes on the GlintDMA.NumberOfSubBuffers registry variable.

Disable Fast Clear Planes
Checking this box will disable the use of the fast Depth Clear planes and is equivalent to setting the environment variable GLINT_DONT_USE_FCP to TRUE. This option should be used in cases where the Depth buffer needs to be read back into the application..

Disable Delta
Checking this tick box will prevent OpenGL using the GLINT DELTA triangle setup chip on 3DP-V500TX based boards. This is useful for performance comparisons.

Draw Line Endpoints
This option when set can improve the legibility of text rendered by some applications using stroke fonts, such as ProEngineer.

Force Nearest Neighbour Texturing
Setting this registry value will ensure that OpenGL only performs nearest neighbour texturing operations. In some applications this can give a performance, though in some cases using a lower quality texture filter. Note that textures will still be rendered with perspective correction. Tick this box if you are happy with the performance/texturing quality that is achieved with your application.

Enable Texture Compression
Setting this registry value will force OpenGL to shrink 2D texture maps as they are loaded to reduce the memory needed to store them. Texture maps are halved in both x and y dimensions so that they require a quarter of the original memory. The setting has no effect on 1D or paletted texture maps. This setting applies to all hardware configurations.

Disable MipMapping
This variable is only relevant to the GLINT ASUS 3DP-V500TX . Checking this tick box will prevent OpenGL from allowing mip-map texture filtering to be enabled. This option will override mip-map filtering settings of an OpenGL application, forcing the allowed filtering to be either GL_LINEAR or GL_NEAREST. These two filtering options are rendered much faster than the mip-map filtering option on the GLINT ASUS 3DP-V500TX.

Gamma Correction Adjustment
The gamma correction adjustment affects the entire screen display. The default gamma value is 1.0 and the allowable range of floating point values is 0.3 to 4.0. Any new user-defined value will take effect when the Apply or OK button are selected.

OpenGL Environment Variables

The control panel applet is the preferred method for controlling OpenGL behavior. However, two environmental variables are described below which provide an alternative means of changing two of the options.

The environment variable `GLINT_DONT_USE_FCP` can be used to disable the use of the Fast Clear Planes by the OpenGL Installable Client Driver. This is done by setting the variable to TRUE within the system applet of the control panels.

OpenGL Texturing & Extensions

Efficient use of multiple textures

OpenGL applications that wish to render primitives with multiple texture maps will achieve much higher performance by avoiding the invoking of the different textures in immediate mode. There are two alternative options for efficient switching between multiple textures.

The first option is to define each texture (or array of mip map resolutions) within a display list. Switching between different textures is then achieved by referencing the appropriate display list. Since display lists are not editable in OpenGL, the OpenGL implementation is able to cache texture data defined within a display list. This caching cannot be performed when a texture is invoked in immediate mode since the application in this case is at liberty to have changed the texture data since any previous reference. The performance gain using this approach will benefit performance for ASUS 3DP-V500TX.

The second option is to use the OpenGL texture object extension. This functionality has become a standard part of the OpenGL version 1.1 functionality. Unlike the use of textures in display lists, texture objects are fully editable and may have their images and parameters altered at any time (when bound). Refer to Appendix E for further details.

Considerations specific to ASUS 3DP-V500TX Board

For the ASUS 3DP-V500TX, texture data is stored in the local buffer memory on the graphics card. The memory available for textures is therefore constrained by the local buffer memory available. It is also constrained by the amount of local buffer memory already consumed for the depth buffer, stencil buffer, etc., which varies according to the current display resolution in use. I.e. there is more memory available for textures when the display resolution (and therefore the size of the depth buffer, stencil buffer) is lowered.

If the condition is reached where there is insufficient local buffer memory to load a new texture then the OpenGL texture download will not succeed and will set the error code `GL_OUT_OF_MEMORY`. Textured primitives that expected to use this texture will not be rendered correctly.

The default texture minification filtering for OpenGL involves mip-map filtering. This gives good textured rendering quality but at the cost of low performance. Much higher performance can be obtained by changing the default texture filtering such that the minification and magnification filtering modes are the SAME. Setting them to GL_LINEAR gives good quality bilinear filtering and improved performance. Setting both modes to GL_NEAREST will give nearest neighbour filtering and the fastest possible performance.

BGRA Extension
This extension provides an additional pixel colour format for compatibility with the blue, green, red component ordering of Microsoft Windows DIB (device independent bitmaps).

Palette Texture Extension
The ASUS 3DP-V500TX provides direct support for palette textures, where each texel represents an index into an on-chip 16 entry RGBA (8-bits per component) lookup-table. An OpenGL palette texture extension has been defined by Microsoft which is supported by OpenGL ICD from release 1.0.11. The ASUS 3DP-V500TX supports 1, 2 & 4bit texel depths.

ASUS 3D Labs Driver extension
In addition to the extensions mentioned above, the 3Dlabs_DriverState extension has been added. This extension is simply a mechanism for adding extra state to the Client Driver and add extra control to the currently selected context. For more details, please contact ASUSTeK COMPUTER INC.

Performance Monitoring

The current release supports simple performance monitoring. This will be extended in future releases. Currently, the main use is to allow a user to determine how much time is being spent by the GLINT chip and how much time is being spent in the 3D application (including time spent in OpenGL for instance). The performance monitoring is not specific to OpenGL and can be used by any application or DLL which uses the ASUS 3Dlabs extension.

For Pentium and Dec Alpha platforms, this release of the NT 3.51 display driver provides performance counters which allow the standard NT Perfmon utility to be used to measure performance characteristics for 3D, DMA driven applications. This feature is not currently supported under NT 4.0.

The performance monitoring is only available when using interrupt driven DMA. Other configurations will allow graphs to be generated but they may not be meaningful. In addition the BitBlt for swapbuffers is not presently taken into account in the GLINT busy time, so the results are only strictly accurate for single buffered or color space double buffered applications.

Examinable performance counters

% GLINT busy: This variable indicates the percentage of time that the GLINT chip spends rendering.

% Host busy: This variable indicates the percentage of time that the host CPU spends in the application and in the driver but not waiting for GLINT DMA to complete.

% Wait DMA: This variable indicates the percentage of time that the host is waiting for space in the DMA buffer queue. When this value becomes nonzero it indicates that the queue of DMA buffers is full and that the driver must wait for GLINT to complete the existing DMA buffer and start the next one to free a space in the queue.

DMA Buffer loads/sec: This is the number of DMA buffers that are being started every second by the DMA interrupt handler.

DMA dwords/sec: This is the number of dwords of information that are being DMAd to GLINT every second. This number divided by the number of buffer loads per second gives the average size used in each DMA buffer.

% Wait VBlank: This variable can be used in non-interrupt driven DMA mode to see the percentage of time that the host spends waiting for the VBLANK interrupt. In interrupt driven DMA mode this value will always be zero.

Configuring GLINT Performance Monitoring

On a Pentium machine the performance monitoring is installed as follows:

- Open a Command Prompt DOS window
- Insert the Windows NT 3.51 driver floppy into the drive "**A:**"
- Change to drive "**A:**"
- Type the command: **lodctr glntctrs.ini**
- The performance monitoring is now set for use.

If at any time it is necessary to rerun the **lodctr** utility then the following command should first be run to uninstall GLINT performance monitoring capability: **unlodctr glint**

Using Perfmon for Performance Monitoring

The Perfmon utility is found in the "Administrative Tools" program group. It can be run either locally or across the network. The best option is to run the utility on a remote machine across a network. This allows the full screen of the local machine to be used to display the 3D application. Also, since displaying the graphs for the perfmon output is done by the display driver, running perfmon locally will impact the performance of the 3D application being measured. When running remotely, the only overhead is some network traffic every probe interval (by default once a second).

Running perfmon locally

- Start the Perfmon utility from the Administrative Tools program group.
- From the Edit menu choose "**Add to Chart** ..."
- The Object field provides a sorted scrollable list. Choose the GLINT object.
- In the Counter field the GLINT counters described above will be listed.
- Choose "**% GLINT busy**" and press the "**Add**" button. Then press the "**Done**" button (when "**Add**" has been pressed the "**Cancel**" button turns into the "**Done**" button).
- A graph will be added which indicates how busy the GLINT chip is. Initially, this will be zero since there are no 3D interrupt driven DMA applications running.
- Start a 3D OpenGL application such as one of the standard demos provided with the release. The GLINT busy graph will start to register. Sampling is once every second by default.
- To add new counters repeat the steps by choosing "**Add to Chart** ..." from the Edit menu.

Running perfmon remotely

Perfmon can be run on a remote machine connected to a machine with a GLINT card over a network. The remote machine does not need a GLINT card but must be running Windows NT. The remote machine is used solely for displaying the graphical output of the Perfmon utility. Permon takes care of sampling the GLINT counters via a network connection.

Running Perfmon remotely

- On the remote machine, start the Perfmon utility from the Administrative Tools program group.
- In the Computer field, press the button labelled "..." or if you know the name of the machine on which the GLINT installed type \HOSTNAME" where the hostname is the network name of the GLINT machine. After typing the name press Return.
- If the "..." button is used then a standard network dialog will be displayed. Choose the remote machine in the usual way.
- You are now connected to the GLINT machine. Continue in the same way as for the local procedure described above.

Event Logging

This release has been extended to register a number of event log errors and warnings when problems are encountered. The events that can be logged include:

- no DMA support has been configured
- no interrupt driven DMA has been configured
- a non-cache coherent PCI bus has been detected which results in uncached DMA buffers.
- fewer than the required number of DMA buffers have been allocated.

After booting the driver it is advisable to check the system event log to determine the characteristics of your machine. For example, if an event log indicates that interrupt driven DMA has not been configured, this may be because the BIOS has not been configured for PCI interrupts. This would also be an indication that performance monitoring will not provide meaningful results since the % busy counter depends on DMA interrupts working.

To view the system event log, run the Event Viewer from the Administrative Tools program group. From the Log menu ensure that the System Log has been selected. Look for events with the Source type glint. Double click on these events to read the event message.

If no glint events are logged then everything is working perfectly. In this case interrupts are working, all DMA buffers have been allocated and the PCI bus is cache coherent.

V. RESTRICTIONS

Restrictions & Trouble Shooting

PCI BIOS

Some PCI BIOS may not assign correct physical addresses to PCI regions. Experience shows that this sometimes happens with the PCI region for the framebuffer. If this problem does arise, the NT driver will boot but black areas will be seen on the screen. If this happens then a new physical address can be configured for the framebuffer by setting a registry variable. If this variable exists its value will override any address set up by the PCI BIOS.

If having booted the NT driver, black areas are seen on the screen, try setting this override variable as follows:

1. run regedt32.exe
2. open the key

   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\glint\Device0
3. From the Edit menu choose the “Add New Value” option.
4. Set the Value Name to be “PhysicalAddress.Region2”. Be careful to spell the name exactly as specified – it is case sensitive.
5. Set the data type to be “REG_DWORD” and press OK.
6. In the DWORD editor window set the physical address value (see below for suggestions) and press OK.
7. Check that the entry has been created correctly and reboot the machine.

Selecting physical addresses in this way is an empirical task. An address must be chosen which does not conflict with any other in the system (the PCI address space is 4 GigaBytes in size so there is plenty to choose from). This task should be performed by the PCI BIOS but if it fails to do this the user must choose instead. A useful address to start with for the framebuffer is 0xA0000000. If this fails increment the address in units of 32MB. Another good starting address is 0x40000000. The final address chosen **must** be on a 32MB boundary.

On MIPS machines physical addresses should have the top 4 bits set to zero. To ensure this for the whole region the address chosen should be less than or equal to 0x08000000.

Having created the Region2 registry variable and assigned it a value the machine should be rebooted. Continue modifying the address until the black areas of the screen do not appear. Normally, this will work after the first one or two addresses.

Note, this procedure is required very rarely. Generally, the user will never be required to perform these steps.

V. RESTRICTIONS

Display Driver

The following errors have been noted during testing of this software release:

- In 4096 color mode, the standard Paintbrush application from the Accessories Program Group does not work correctly. This application does work correctly at all other pixel depths.
- The HCT GDI and GDI w/clip fail at 4096 color mode. This is believed to be due to a limitation in the conformance tests.

OpenGL

- The depth clear conformance test will fail unless the environment variable **GLINT_DONT_USE_FCP** is set to TRUE, or the corresponding box in the control panel applet is checked. The problem is with the readback of the Depth Buffer, not its clearing. If this variable is set then this disables the use of ASUS proprietary fast clear mechanism that allows the depth(Z) buffer to be cleared up to 16 times more quickly than normal. Typically this becomes significant for animation rates of 10Hz or higher in large windows.
- The conformance tests that involve mip-map texture filtering (miplin.c, mipsel.c and texbc.c) will fail on 500 TX if the DisableMipMap check box in the control panel applet is set. The default state is to have this set because this provides the best trade-off between image quality and performance for the majority of applications.
- When using the glaux library supplied by Microsoft, specifying that you require alpha planes in the visual is not satisfied by requesting a visual type of AUX_RGBA as opposed to AUX_RGB when calling auxInitDisplayMode(type). In these instances the hardware accelerated visual that will be returned in some modes may not have alpha planes. This is because the display driver exports visuals without alpha planes before those that do. This problem can be resolved in two ways: Firstly, if you have the source code, then when specifying the visual type you can OR in AUX_ALPHA, along with AUX_RGB (for Example auxInitDisplayMode(AUX_RGB | AUX_ALPHA)). Secondly, if source is not available, the following registry variable can be set to 1, which enables the visuals with alpha planes to be selected first.

  3DExtensions.ExportAlpha

  in:

  HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\glint\Device0

  Setting this variable will result in a decrease in the performance of some applications as the driver must perform additional setup calculations for ASUS 3DP-V500TX to cater for the Alpha value as well as R, G, and B.

V. RESTRICTIONS

- In some cases, there is confusion over the meaning of the PFD_SUPPORT_GDI bit in the dwFlags field of the PIXELFORMAT descriptor. ASUS have seen applications (for instance Open Inventor) which incorrectly assume that if this flag is set, rendering to bitmaps is supported by the visual. The Installable Client Driver does not support bitmap rendering so these applications fail. To enable these applications to work the exporting of PFD_SUPPORT_GDI can be disabled by setting the following registry variables FALSE. The applications will then choose a Generic pixel format so using unaccelerated software rendering to draw to bitmaps.

  3DExtensions.SupportSingle
  3DExtensions.SupportDouble

  By default **PFD_SUPPORT_GDI** is set to TRUE for single buffer formats and FALSE for double buffered formats.
  Note under WindowsNT, Generic pixel formats that support double buffering and rendering via GDI are mutually exclusive. This is because GDI does not have the ability to render to the back buffer. ASUS have therefore chosen to set the default for double buffering, so as to be in line with the Microsoft implementation. However with care GDI rendering and double buffering may be mixed, so the latter registry variable will cause PFD_SUPPORT_GDI to be exported by double buffer formats, should an application benefit from this added functionality.

- When running multi-threaded applications it may be necessary to disable the use of the fast clear planes by setting the environment variable **GLINT_DONT_USE_FCP** to **TRUE**, or by checking the corresponding box in the control panel applet. This issue arises when more than one context is being used to render to the same window (e.g. Windows NT 4.0 OpenGL pipes screen-saver with multiple option selected). If this variable is set then this disables the use of ASUS proprietary fast clear mechanism that allows the depth (Z) buffer to be cleared up to 16 times more quickly than normal. Typically this becomes significant for animation rates of 10Hz or higher in large windows.

- On Windows NT 4.0, the standard OpenGL screen savers do not get hardware acceleration at the 32668 color depth. The problem is due to the fact that the Microsoft binaries test for a pixelformat with the cColorBits field set to 16 at this color depth, whereas the pixel formats exported for all our hardware is the correct value of 15 at this color depth. All other color depths do not suffer from this bug.

V. RESTRICTIONS

Problems and Solutions

1. **Under 4096 colors depth, Paint Brush cannot fill with colors**
   This is a known problem with Microsoft GDI with Windows NT 3.51. The problem does not exist in Windows NT 4.0.

2. **The window of 3D Paint (included in Microsoft NT3.51 Resource Kit) interferes with rendering progress dialogs. There are wrong results in the dialog block even when it is in background.**
   3DLabs is currently looking into this problem.

3. **About Heidi driver in 3DMAX:**
   a. In 256 color mode, 3DMAX displays a wrong background color under Windows NT 4.0 and Windows NT 3.51 platforms.
   Our Heidi driver does not support 256 color mode. We recommend that you run 3DSMAX at 15 bits per pixel or higher.

   b. When the 3DMAX program window is moved off the screen and back, the window does not refresh itself therefore displaying the wrong content.
   Heidi does not handle window expose events, this is not a limitation of our driver but instead the application's. A user can manually force a redraw of all windows by pressing '1'

A. REGISTRY & DMA VARIABLES

Registry Variables

In addition to the standard registry variables set up by the driver and the display applet, the following configuration variables are used. They are located in: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\glint\Device

Timing Register Configuration
See the GLINT Hardware Reference Manual for details of the registers described in this section.

GlintTiming.VTGSerialClk: If defined this register is used to program the GLINT VTGSerialClk register. If not defined then a suitable default value will be used.

This register controls some fundamental VRAM size values. It should be modified in the oemsetup.inf file supplied with the driver for a given board.

GlintTiming.LBMemoryCtl:
GlintTiming.LBMemoryCtlMask: These two variables are used as a pair. The Ctl variable specifies a value to be loaded into the GLINT LBMemoryCtl register. The CtlMask variable specifies which bits of that register are to be modified. If ulValue is the value read from the LBMemoryCtl register then the new value loaded is: (ulValue & ~CtlMask) | (Ctl & CtlMask).

If the Ctl variable is not defined or is zero then the value defined by resistors on the board is used. If the CtlMask variable does not exist then a value of -1 is used. You must reboot for changes in this variable to take effect.

GlintTiming.FBMemoryCtl:
GlintTiming.FBMemoryCtlMask: These variables are used in the same way as the LBMemoryCtl variables but they control the value to be loaded into the GLINT FBMemoryCtl register.

If the Ctl variable is not defined or is zero then the value defined by resistors on the board is used. If the CtlMask variable does not exist then a value of -1 is used. You must reboot for changes in this variable to take effect.

GlintTiming.FBModeSel:
GlintTiming.FBModeSelMask: These variables are used in the same way as the FBMemoryCtl variables but they control the value to be loaded into the GLINT FBModeSel register.

If the Ctl variable is not defined or is zero then the value defined by resistors on the board is used. If the CtlMask variable does not exist then a value of -1 is used. You must reboot for changes in this variable to take effect.

GlintTiming.Use2ClockMemoryCtl: If defined and nonzero this variable causes default 2 clock memory cycle timing values to be loaded into the LBMemoryCtl and FBMemoryCtl registers. In this case the registry variables GlintTiming.LBMemoryCtl and GlintTiming.FBMemoryCtl are ignored. By default this variable is defined and nonzero. This provides backward compatibility with driver releases before version 1.6 where the clock timing was forced to 2 clocks for a page access and 5 clocks for a non-page access. You must reboot to change.

GlintClockSpeed: This variable is used to indicate the oscillator frequency of the reference board (this value cannot be read from the board). By default this value is zero indicating that the clock frequency used should be the default for the chip revision. This variable should only be changed if the clock speed of the board is not the default for the chip type.

This variable is read every time the Test button is used in the Display Applet (and also at boot time). Thus to change the clock speed, start the Display Applet and press the Test button. On return from the test screen the new clock speed will have been loaded into GLINT. When asked if the test screen could be seen answer No and press Cancel in the Applet. This will leave all system settings unmodified except the GLINT clock speed will have been updated.

DMA Control Variables

GlintDMA.NumberOfBuffers: This defines the number of DMA buffers configured for use by the 3D extension. As each 3D context is created a DMA buffer will be allocated to it. Typically, each 3D application (whether OpenGL or another accelerated 3D API) will use a single context. If all DMA buffers have been allocated then shared memory buffers are used instead.

If this variable does not exist or its value is set to zero then use of DMA is disabled. DMA will not be available if the installed GLINT board does not support DMA transfers. In this case an event is logged in the system log file. This can be viewed using the Event Viewer. The installation default is 4. You must reboot to change.

GlintDMA.SizeOfBuffer: This variable defines the size in bytes of each DMA buffer. Its value will be rounded up to the next system page boundary. The size will be limited to 256KB since this is the maximum size that can be specified to GLINT for a DMA transfer. The installation default is 0x10000 (64KB). You must reboot for changes in this variable to take effect.

GlintDMA.NumberOfSubBuffers: This variable describes the number of sections into which a single DMA buffer is divided for use by a 3D application. This is used by interrupt driven DMA to construct a queue of buffers which are to be loaded into GLINT by the DMA interrupt handler. The maximum size of the queue is always 2 less than the number of sub-buffers specified. This is to allow the 3D application to fill one buffer and GLINT to be performing DMA on another buffer. The remainder can be queued.

Setting this variable to 2 disables interrupt driven DMA since not enough buffers are available to create a DMA queue. DMA still works but the DMA buffer is split into two parts so that GLINT can load one while the 3D application prepares the other. Setting this variable to zero disables DMA and forces FIFOs to be used. This latter feature is generally used only for comparing performance of DMA and non-DMA operation (without needing a reboot).

This variable is read every time a 3D context is created. Thus it is not necessary to reboot the machine for a change to take effect. Changing this variable has no effect on any already running 3D applications. The installation default is 5.

This variable can have considerable 3D performance implications. 2 is usually ideal for single-buffered applications and 5 seems well suited to double-buffered applications.

GlintDMA.AllocateCached: The driver automatically determines whether the DMA buffers can be allocated as cached or uncached. Setting this variable to zero forces buffers to be uncached; setting it to 1 forces buffers to be cached. It is not recommended that this variable be created or modified. The default installation does not create this variable. You must reboot for changes in this variable to take effect.

This variable should not be defined on Alpha platforms as using uncached DMA buffers has undefined results on these machines.

GlintDMA.LatencyTimer: This variable sets the PCI latency timer for the GLINT chip. This determines the maximum number of PCI cycles that GLINT can hold onto the PCI bus while performing DMA once the bus grant has been removed. It is not recommended that this variable be modified. You must reboot for changes in this variable to take effect.

3DInterfaceBuffer.SizeLongs: This defines the length of the shared memory buffer used by the OpenGL DLL to communicate with the driver. The length is specified in DWORDS. The installation default is 0x2000 (32KB).

3D Double Buffering Control

DoubleBuffer.NumberOfBuffers: This specifies the total number of screen-sized buffers to be allocated from VRAM by the driver. One buffer is always allocated for the main, displayed screen. If this variable exists and is greater or equal to 2 then a second off-screen buffer is allocated for use by the 3D extension (values > 2 are reserved for future use). Any VRAM remaining after allocation of the screen-sized buffers is available for use by the display driver for pattern cache and off-screen bitmaps. The installation default is 2. You must reboot for changes in this variable to take effect.

DoubleBuffer.MultiColorSpace: Setting this variable to 1, specifies that the system lock on the double buffering token should be ignored. This will allow applications such as the ProEngineer CAD package which creates multiple double buffered rendering contexts, but only ever display one double buffered window to use color space double buffering. The caveat is that if a second double buffered application is started then it will ignore the system lock resulting in flickering or incorrect pictures being displayed, as there will be no arbitration between the applications, of when the swapbuffers command (which affects the whole screen) is executed. This is not created at installation by default which is the same as setting it to 0, meaning that the lock is in force. You must reboot for changes in this variable to take effect.

OpenGL Overlay Planes Support

Introduction

Overlay planes provide a method by which OpenGL applications can render over screen areas in a nondestructive way. A typical use of such functionality would be to draw dialog boxes above a rendered 3d scene. There would be no need to save and restore from a bitmap or re-render the scene when the dialog is cleared as the standard, main-plane, rendering remains intact. Overlay planes are used by many high end OpenGL applications as they are a common feature on Silicon Graphics hardware.

Overlay planes are not provided as a standard part of OpenGL in the way that stencil or depth buffers are. On Windows NT Microsoft define a standard overlay plane specification as part of the WGL interface. This interface is used by Windows NT OpenGL applications which require overlay functionality, notably Softimage.

Implementation

The Microsoft overlays standard provides for up to 15 levels of underlay and 15 levels of overlay in addition to the main rendering plane. Each level of underlay or overlay, referred to as a layer plane, has it own layer plane descriptor structure.

OpenGL Registry Variables

OpenGL.SupportSoftimage: This variable is used to optimise the OpenGL drivers for use with version 3.01 of the SoftImage rendering package. We do not recommend use of this variable unless you will be using the SoftImage application. If this variable is defined and set to 1 then the optimizations will be enabled.

A system reboot is necessary for changes in this variable to take effect.

OpenGL.SupportSoftimage351: This variable will optimise the drivers for version 3.51 of the SoftImage rendering package. Once again we recommend that you only use this variable when using Softimage. If this variable is defined and set to 1 then the optimization will be enabled.

A system reboot is necessary for changes in this variable to take effect.

Miscellaneous Registry Variables

UseSoftwareCursor: If defined and set to 1 a software cursor is used instead of the normal hardware cursor. This variable is not created by default.

A system reboot is necessary for changes in this variable to take effect.

ExportSingleBufferedModes: If this variable is defined and set to 1 then single buffered modes are made available via the Display Applet. Note, choosing too high a resolution may result in insufficient VRAM being available to support accelerated double buffered rendering. In this case software only rendering will be used instead.

A system reboot is necessary for changes in this variable to take effect.

The following variables may not be supported in future releases, and are not created by the default installation. Some PCI BIOS allocate invalid physical addresses for the GLINT memory regions. Setting the following variables override the BIOS specification and force the physical addresses of the different GLINT PCI address regions. If these variables are used care must be taken to ensure that they are on the correct boundary for the given region and that different regions do not overlap. Typical addresses to try are high memory addresses such as 0xA0000000. See the section on PCI BIOS under known Anomalies and Restrictions" below, for a description of how to override the physical address for the framebuffer.

AllowAddressOverride: This variable must exist and be set to nonzero in order for the following registry variables to take effect.

PhysicalAddress.Region0: Specifies the physical address for GLINT control registers. This value should be on a 128K boundary.

PhysicalAddress.Region1: Specifies the physical address for the localbuffer bypass. Currently, this is not mapped in by the driver but it must have a valid physical address (not zero) which does not conflict with other addresses in the system since GLINT will respond to accesses in this range. This region should be on an 8MB boundary and is 8MB in length.

PhysicalAddress.Region2: Specifies the physical address for the framebuffer bypass. This address must be on a 32MB boundary and is 4MB in length.

PhysicalAddress.ROMBase: Specifies the physical address for the onboard ROM. Currently, this is not mapped in by the driver but it must have a valid physical address (not zero) which does not conflict with other addresses in the system since GLINT will respond to accesses in this range. This address must be on a 64KB boundary and is 64KB in length.

UseBiosAddresses: Normally, the base addresses for PCI devices are allocated by the NT operating system. Some older X86 machines do not operate correctly if these addresses are used. In particular, this may apply to older 486 machines. If this variable is defined and set to 1 then the base addresses assigned by the PCI BIOS to the GLINT chips are used instead of those assigned by the operating system.

This structure is similar to the standard pixel format descriptor and allows each layer to define it own device capabilities independent of the main plane or other layer planes. This allows a double buffered, rgb main plane to coexist with single buffered, color index overlay plane etc. Each layer defines a color which is defined to be transparent and allows rendering in lower planes to show through. Functions are provided for creating rendering contexts in layer planes, manipulating the palettes of color index layers and swapping layer planes independently of the main plane if the graphics device supports this capability.

ASUS drivers support OpenGL overlays are provided in 32bpp truecolor display modes. In standard truecolor modes each 32bit pixel is split into 8 bits of red, green, blue and alpha. Overlays are provided by replacing the alpha component with 8 bits of overlay. Thus a single 8 bit color index overlay is provided in addition to a 24 bit rgb main plane.

The overlay plane is single buffered if the main plane is single buffered and double buffered if the main plane is double buffered. Independent swaps of the main and overlay planes are supported when blit double buffered. When full-screen double buffered, such as when an OpenGL application takes up the entire display, independent swaps are not supported.

To provide access to OpenGL overlays in 32bpp modes extra pixel formats are exported, in addition to the standard pixel formats. These extra pixel formats are defined to have an 8 bit overlay plane but no alpha. As soon as any OpenGL application chooses a pixel format of one type or the other all the pixel formats that are exported are changed to only support that mode. For instance, if the first pixel format chosen includes overlays all the remaining pixel formats exported will be changed to export overlays but not alpha and vice versa. When all OpenGL windows are closed the system reverts to exporting a choice of pixel formats. The driver must work in this way as setting overlay mode is a global change and alpha and overlay planes cannot be supported simultaneously.

When overlay planes are used an additional registry variable must be set, otherwise icon corruption may be seen, DisableOffScreenBitmaps

This variable should be created as a DWORD value in the same place as the other variables (described in the section, Registry Variables) and set to a value of one. A reboot is necessary for this change to take affect

If the Softimage rendering package (which requires overlay plane functionality) is used we recommend that one of the SupportSoftimage registry variables are set. Full details on these variables are given earlier in the section describing all of the registry variables. This will optimise the OpenGL drivers for use with the package.