3Dlabs®

GLINT® & PERMEDIA®

Software Release Note for Windows NT 4.0 Display & OpenGL Drivers

Issue 10

Email: info@3dlabs.com
Web: http://www.3dlabs.com

3D*labs Inc.*
181 Metro Drive, Suite 520
San Jose, CA 95110
United States

Tel: (408) 436 3455
Fax: (408) 436 3458

3D*labs Ltd.*
Meadlake Place
Thorpe Lea Road, Egham
Surrey, TW20 8HE
United Kingdom

Tel: +44 (0) 1784 470555
Fax: +44 (0) 1784 470699

Change History

Document	Issue	Date	Change
Nt29r20.doc	9	10-Dec-96	New Release
	10	01-Apr-97	Release 2.9

CONTENTS

1 Introduction

This note describes the Windows NT 4.0 Display Driver and OpenGL Installable Client Driver for the GLINT 300SX/500TX reference board (Montserrat), the 300SX/500TX/MX + DELTA board (Racer) and the PERMEDIA reference boards with and without Delta. It also explains how to install these drivers.

Note: *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.*

1.1 Release Identity

Once the driver has been installed and the machine rebooted, the display driver release number can be determined by starting the Display Applet Control Panel in the Main Window Group.

Press the "Display Type ..." button. In the "Display Type" window the Version Numbers "4.00, 4.0.n" will be seen, where n is a 2 digit field giving the display driver version.

The OpenGL release number can be determined by running the X29 (tumbling plane) demonstration program, and selecting "Who's Rendering ?" from the "Help" pull down menu.

1.2 Prerequisites

- Windows NT 4.0 (Build No 1381)
- Intel 486 processor or later or DEC Alpha.
- 3Dlabs Montserrat, Racer board or Racer Pro board, PERMEDIA or PERMEDIA2 reference board.

2 Installation

Before installing the software, power down the machine and install the Montserrat, Racer, Racer Pro, PERMEDIA or PERMEDIA2 board as per 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:

- 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 release floppy for your machine architecture into the drive and press OK. The "Change Display" window will appear with two sub-windows. The left hand sub-window contains a list specifying chip types (e.g. PERMEDIA or GLINT); the left hand sub-window will contain a selection within that group. 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: There are no options to select a given resolution at install time. When the machine reboots Windows 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 GLINT 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 GLINT or PERMEDIA display board. If the standard Windows NT screen does not appear and the display remains at the blue boot screen, ensure that the VGA pass-through cable has been correctly connected.

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

15bpp 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. The display will change dynamically.

The above procedure installs the NT display driver, GLINT 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.

3 Resolutions and Color Depths

A full list of all modes is available via the display applet once the GLINT driver has been installed and the system rebooted. Choose the "List all Modes" option to get this list.

From release 1.8 onwards 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.

4 3D Graphics & Double Buffering

The display driver contains an extension to allow 3D applications, and the OpenGL installable client driver, to drive the GLINT 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 ≥ 2 (see below). This buffer is used in 256, 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. The Montserrat board features 4MB of VRAM, the Racer 8MB and the Racer Pro 16MB. Note that to provide double buffered accelerated capability there must be enough memory to contain two frame buffers. This may not be the case at high resolutions. Also note that on a PERMEDIA Board the memory is unified and the frame buffer memory, depth buffer memory and texture memory all resides in the same place. On TX/SX/MX boards the video memory is separate from the depth buffer/ texture memory.

4.1 Full Screen Double Buffering (All Boards)

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 Windows 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.

4.2 Full Flip Double Buffering (TX/MX Only)

Flip double buffering is available in True-Color mode running at screen resolutions greater than 1024 by 768 on a TX/MX Racer/Racer Pro board. The flip double buffering is available for apps that run in a single window and is facilitated by all of the desktop rendering being written to both frame buffers. Note that when this is available, only the first double buffered window will use this approach, when there are multiple double buffered windows, blit double buffering is reverted to.

4.3 Blit Buffering (All Boards)

Blit double buffering is the simplest form and simply involves copying the contents of the back buffer into the displayed buffer.

Note that for the given configuration, the ICD will always try and use the fastest double buffering method that it can. Also note that if the selected screen resolution/Color depth is too high, then double buffered applications may start to run slowly as they revert to using the Microsoft Generic OpenGL renderer.

For example: When running at a resolution greater than 800 by 600 in True Color Mode (16 Million Colors) on a board with 4Mbyte of VRAM.

5 Control Panel Applet

Ntote: This has been superseded by extensions to the Display 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 (grayed out). The control panel is split into a number of groups as listed below.

Note: It is currently necessary to have administrator privileges to change any settings in the control panel applet. If you do not have administrator privileges, a message box will inform you of the fact and display the control panel with the Apply and OK buttons disabled.

5.1 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

Note: This will take effect with release 1.8 of the GLINT display driver.

5.2 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. This option should be selected by users wishing to run 2D applications at the highest available resolutions.

Note: This takes effect with release 1.8 and later of the GLINT display driver.

5.3 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 Racer SX and TX boards. Changing this option requires a re-boot of the system.

5.4 SoftImage Version 3.51/3.7 Application support

This box must to checked to correctly run Softimage version 3.51/3.7. 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.

5.5 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.

5.6 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.

5.7 Dynamic Buffer Option - Number of Sub-Buffers

Each DMA buffer is sub-divided 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.

5.8 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. This option is not relevant to PERMEDIA.

5.9 Disable Delta

Checking this tick box will prevent OpenGL using the GLINT DELTA triangle setup chip on Racer based boards. This is useful for performance comparisons.

5.10 Draw Line Endpoints

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

5.11 Force Nearest Neighbor Texturing

Setting this registry value will ensure that OpenGL only performs nearest neighbor texturing operations. In some applications this can give better 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.

5.12 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 palleted texture maps. This setting applies to all hardware configurations.

5.13 Use High Quality Texture

This variable is only relevant to the GLINT 300SX . If this box is checked, the texturing code in OpenGL will use a floating point mechanism as opposed to an integer pipe. The floating point mechanism is slower (by 2.5-3 times on an Intel), but has a far greater dynamic range. The integer pipe uses a fixed point format which adjusts dynamically but has a limited range. Once this limit is reached, artifacts can appear. Often the range is more than sufficient to satisfy application needs, however, in cases where the integer pipe cannot cope, this box should be checked.

5.14 Force Hardware Texturing

This variable is only relevant to the GLINT 500TX . Checking this tick box will prevent OpenGL from allowing level of detail (including 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 (to be more precise, the minification filter which can be take on various mipmap quality settings will be forced to use the magnification setting). These two filtering options are rendered much faster than the mip-map filtering option on the GLINT 500TX.

5.15 Perspective Correction

This variable is only relevant to PERMEDIA. The accuracy of the perspective-correction divide performed during textured rendering can be varied on PERMEDIA. The default option, Force Nicest, enforces that the most accurate divide always operates, resulting in the best image quality. The Force Fastest option switches to a reduced accuracy divide when Delta hardware is present and no perspective-correction at all if Delta is not present. This results in best performance at the cost of lower image quality. The third option gives control of the divide accuracy to OpenGL applications through the API function glint. A smart application can vary the divide accuracy used on a per primitive basis, since primitives that are heavily perspected require higher accuracy for good image quality than those that are not.

5.16 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.

5.17 Chip Clock Speed

For information only. This reports the frequency to the nearest MHz of the Glint graphics chip (not to be confused with the RAMDAC frequency etc).

5.18 OpenGL Ver

For information only. Reports 1.1.xx where the last two digits identify the ICD internal release number (increments with every release).

5.19 ICD Build

For information only. This reports 4.0x.00.xxxx-xxxx corresponding to the File and Product Version information required for Window Hardware Quality Labs compliance under Windows95 (the same ICD binary runs under both WindowsNT and 95). The vendor is free to use the last four digits for their own use - in this case the daily build number that uniquely identifies the Installable Client Driver (ICD) binary.

5.20 Texture Memory

For information only. This reports the total amount of graphics card memory left over for storing texture maps - after allowing for full screen front, back and depth buffers. Smaller desktop colour depths and/or screen resolutions will free up more graphics card memory for texture use increasing the amount available for non-swappable textures (refer to the notes below on the texture memory manager).

5.21 Depth/Stencil Buffer

For information only. This reports the total amount of memory used by auxillary buffers for use in hardware hidden surface removal (z-buffering) and masking/clipping (stencil and gid planes).

5.22 Sync DblBuffer with Vblank

Smooth animation of 3d applications can be achieved by rendering to an off-screen window/desktop sized colour buffer and copying or swapping the contents to the displayable front buffer at the completion of each frame. Enabling this option prevents tearing of the display by synchronizing the swap of the back and front buffers to the vertical blank retrace interval of the monitor display.

Leave this option unticked if the highest rendering frame rates of a double buffered application are desired (i.e. not locked to sub-multiples of the current display refresh rate).

5.23 Disable PCI Disconnect

Higher 2D graphics performance can be achieved by using PCI Disconnect, however this feature can, sometimes, adversely affect the performance of other devices, such as modems and sound cards.

Tick this option if you are experiencing problems with the performance of other devices.

6 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`

6.1 Timing Register Configuration

See the GLINT Hardware Reference Manual for details of the registers described in this section. The timing related variables are not applicable to PERMEDIA based boards.

`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.

A reboot is necessary for changes in these variables 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.

A reboot is necessary for changes in these variables 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.

A reboot is necessary for changes in these variables to take effect.

```
GlintTiming.Use2ClockMemoryCtl:
```

If defined and non-zero 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 non-zero. 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.

A reboot is necessary for a change in this variable to take effect.

```
GlintClockSpeed:
PERMEDIAClockSpeed:
```

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. PERMEDIAClockSpeed is identical to GlintClockSpeed except that it applies specifically to PERMEDIA and PERMEDIA2 which are driven by oscillators running at higher frequencies than for GLINT chips.

6.2 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.

The machine must be rebooted for a change in this variable to take effect.

```
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).

The machine must be rebooted for a change 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 release 1.4 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.

It is necessary to reboot the machine in order for a change in this variable to take effect.

Note: *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.

A system reboot is necessary 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).

6.3 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 ≥ 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.

A system reboot is necessary 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 create 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. The not created at installation by default which is same as setting it to 0, meaning that the lock is in force.

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

6.4 OpenGL Registry Variables

`OpenGL.SupportSoftimage:`

This variable is used to optimize 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 optimize the drivers for version 3.51/3.7 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 optimizations will be enabled.

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

`OpenGL.MaxTextureSize:`

This variable specifies the size of the texture cache equivalent to a maximum width and height (minus any border) of a texture as a power of 2 (a restriction of the OpenGL API) for use by the texture memory manager. Applicable to the TX/MX and PERMEDIA only, this variable requests a portion of local buffer memory be set aside for use as a texture cache for swapping textures in/out of host memory as needed. This is only a *request*, which the driver will attempt to satisfy in the available memory left over (after allocation of any stencil/z buffers etc) down to a minimum size of 64 which all OpenGL implementations are required to guarantee.

Setting this variable to any value less than 6 (i.e. 2^6 or 64) will clamp to 6. The only exception is a setting of zero which disables the texture memory manager, does not set aside any texture cache so that texture downloads will return the GL_OUT_OF_MEMORY error when texture memory becomes exhausted. (See below under OpenGL Texturing Issues for more information on the texture memory manager).

A system reboot is only necessary under NT v3.51 for changes in this variable to take effect. A display resolution change made on the fly under NT v4 will reinitialize the size of the texture memory cache.

`OpenGL.DisableDisplayListTextureSwitch:`

This variable disables the non-standard switching between textures placed in display lists. The recommended method for new applications written for OpenGL 1.1 is to use texture objects instead. The default is not to disable this behaviour. (For more information refer to the discussion below on the efficient use of multiple textures).

6.5 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's 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.

`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 on-board 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.

7 OpenGL Texturing & Extensions

7.1 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 and much preferred option is to use the OpenGL texture object functionality. Texture objects are fully editable and may have their images and parameters altered at any time (unlike the use of textures in display lists). Details on texture object functionality is available in the OpenGL 1.1 specification and also in Appendix E in the 3Dlabs OpenGL Extensions document for further details. The performance gain using this approach will benefit performance for both the GLINT 300SX ,GLINT 500TX/MX and PERMEDIA.

The second option is to define each texture (or array of mip map resolutions) within a display list - *with the limitation that only one texture is allowed per 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. In effect the display list identifier acts as a texture handle. 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.

From release 1.1.20 onwards this non-standard behaviour can be disabled with the registry variable OpenGL.DisableDisplayListTextureSwitch - this will prevent conflicts with texture object routines in display lists (e.g. glBindTexture etc) and also prevent glTexImage calls from downloading texel data in compile only display lists (hence overwriting any existing immediate mode texture).

7.2 Considerations specific to GLINT 500TX/MX and PERMEDIA Texture Memory Cache Management

For the GLINT 500TX/MX and PERMEDIA, 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.

On 3Dlabs OpenGL releases prior to version 1.0.14 (or 1.1.14 under NT4), 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. To improve on this behavior a scheme for swapping textures to/from system host memory is required. By setting aside a portion of texture memory on the graphics card for use as a texture cache and tracking when a texture switch takes place, textures can be reloaded to the cache as needed from a copy kept in host memory when the texture was first downloaded. If the requested texture is already present in the cache, then no reload is performed.

Ideally for the greatest flexibility and most efficient use of available texture memory, all textures should be cacheable. However for a software texture cache manager there is a small performance overhead to be paid for this tracking plus any delay in reloading a swapped out texture (as a texture could be swapped out at any time by another OpenGL process). By allowing the user to specify the size of the texture cache through the use of the registry variable OpenGL.MaxTextureSize (as described above), an approximate balance between non-swappable and swappable textures can be made (and hence performance). Thus an application should load any real-time critical textures first as the texture manager will only place textures in cache and/or host memory if a space of sufficient size is not available in the non-swappable area of texture memory.

In order to guarantee all texture requests no matter how large, any texture whose size in texels is greater than the cache size will be silently filtered down to fit in the cache (while preserving aspect ratio). On the GLINT TX/MX, the size of the texture cache allows extra room for all lower mipmap level textures including border texels (texture borders and mipmapping are not supported on Permedia).

As noted under the description for MaxTextureSize, the texture cache manager can be disabled by setting this registry variable to zero so that all textures are non-swappable as in previous releases.

7.3 Texture Filter Modes

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 neighbor filtering and the fastest possible performance.

Only linear and nearest neighbor texture filtering modes are supported on PERMEDIA (mip-map filtering is not supported).

7.4 BGRA Extension

This extension provides an additional pixel color format for compatibility with the blue, green, red component ordering of Microsoft Windows DIB's (device independent bitmaps). Refer to Appendix D in the 3Dlabs OpenGL Extensions document for further details.

7.5 Palette Texture Extension

The GLINT 500TX/MX and PERMEDIA both provide direct support for palette textures, where each texel represents an index into an on-chip RGBA (8-bits per component) lookup-table. An OpenGL palette texture extension has been defined by Microsoft which is supported by 3Dlabs OpenGL ICD from release 1.0.11. The supported texel depths depend on the chip. They are as follows:

GLINT MX : 1,2,4 and 8 bit texel depths.

GLINT TX : 1,2,4 bit texel depths.

PERMEDIA only supports 4 bit texel depths.

PERMEDIA2 only supports 4 bit and 8 bit texel depths.

Besides improving texture performance and reducing the memory requirements for storing textures in the local buffer, by repeatedly updating the texture LUT, animation effects such as real-time color cycling are also possible. Refer to Appendices B and C for further details.

If many textures share the same look-up table (LUT), performance gains can be obtained with paletted texture objects by forcing the textures to share the same palette (particularly for 8-bit palette textures on the MX and P2). The default behaviour when texture switching through calls to glBindTexture is to send down the LUT on every switch. This can be disabled by the 3Dlabs Driver LOCK_GLOBAL_TEXTURE_PALETTE extension described in Appendix G of the OpenGL Extensions document.

7.6 3Dlabs Driver extension

In addition to the extensions mentioned above, the 3Dlabs_DriverState extension has been added and is detailed at Appendix G in the OpenGL Extensions document. This extension is simply a mechanism for adding extra state to the Client Driver and add extra control to the currently selected context.

8 GLINT 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 % GLINT 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.

9 OpenGL Overlay Planes Support

9.1 Introduction

Overlay planes provide a method by which OpenGL applications can render over screen areas in a non-destructive 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.

9.2 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's own layer plane descriptor structure. This structure is similar to the standard pixel format descriptor and allows each layer to define it's 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.

3Dlabs drivers support OpenGL overlays on Racer based boards only. On such boards 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 GLINT 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 optimize the OpenGL drivers for use with the package.

Readers are referred to Appendix A in the 3Dlabs OpenGL Extensions document for further information about the use of OpenGL overlay planes.

10 Known Anomalies and Restrictions

10.1 PCI BIOS

Some PCI BIOS's may not assign correct physical addresses to PCI regions. Experience shows that this sometimes happens with the PCI region for the GLINT 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:

- run regedt32
- open the key `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\glint\Device0`
- From the Edit menu choose the "Add New Value" option.
- Set the Value Name to be "`PhysicalAddress.Region2`". Be careful to spell the name exactly as specified – it is case sensitive.
- Set the data type to be "REG_DWORD" and press OK.
- In the DWORD editor window set the physical address value (see below for suggestions) and press OK.
- 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.

10.2 Display Driver

10.2.1 OpenGL

- The depth clear conformance test will fail on a GLINT 300SX unless the environment variable `GLINT_DONT_USE_FCP` is set to TRUE, or the corresponding box in the GLINT 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 3Dlabs 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 GLINT 500 TX if the Force Hardware Texturing check box in the GLINT control panel applet is NOT 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 GLINT to cater for the Alpha value as well as R, G, and B.
- In some cases, there is confusion over the meaning of the PFD_SUPPORT_GDI bit in the dwFlags field of the PIXELFORMAT descriptor. 3Dlabs 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 Windows NT, 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 backbuffer. 3Dlabs 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 GLINT control panel applet. This issue arises when more than one context is being used to render to the same window (e.g. OpenGL pipes screen-saver with multiple option selected). If this variable is set then this disables the use of 3Dlabs 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 standard maze screen saver does not get hardware accelerated. This is due to a bug in the Microsoft screen saver library. A customized accelerated 3Dlabs version that also supports linear filtering in the settings option has been provided with this release.
- The following OpenGL version 1.1 API calls are not currently implemented for this release:
  glIndexub
  glIndexubv
  glPolygonOffset
  glCopyTexImage1D
  glCopyTexImage2D
  glCopyTexSubImage1D
  glCopyTexSubImage2D
  glTexSubImage1D
  glTexSubImage2D
- Support for texture memory caching is automatically disabled for a dual-head TX or PERMEDIA setup.

10.2.2 PCI Disconnect

As mentioned earlier, on occasions, the use of PCI Disconnect can cause other devices to suffer from performance problems, tick the 'Disable PCI Disconnect' option in the Control Panel Applet to disable this feature.