Graphics Adapter Device Drivers

About this Book

The Graphics Adapter Device Driver (GRADD) Reference supports OS/2 Warp on the Intel hardware platform. The information in this book describes the GRADD driver model, how the related components work together, and why the GRADD model enhances OS/2 Warp device-driver support.

Detailed descriptions of control structures, data structures, and I/O formats have been included to help you understand and use the interfaces.

Developers using this book should be familiar with the C or assembler programming language and the OS/2 operating system

Introduction to Graphics Adapter Device Drivers

This chapter briefly describes the design and intent of the Graphics Adapter Device Driver (GRADD) model. Details on specific components of the GRADD model are located in GRADD Model Components.

The GRADD model is divided into several components that work together on the same desktop to support a variety of operating system services for OS/2 Warp. GRADD components include the following:

- Video Manager and the Video Manager Interface (VMI) protocol
- Translation layers, one for each graphics engine in OS/2 Warp
- SOFTDRAW for default software simulation of graphics functions
- GRADDs and the Graphics Hardware Interface (GHI) protocol

GRADDs support all the graphics subsystems designed to run on OS/2 Warp. A GRADD contains only the hardware-dependent code required for graphic functions that are common among different graphics subsystems. These common functions are designed to act as a small set of building blocks for the larger, more complex operations that are typically required by a graphics engine. The translation layers that exist between the graphics subsystems and the components of the GRADD model provide access to the GRADD building blocks. The transition layer converts the complex function calls issued by a graphics subsystem into the protocol required by the GRADD model.

By reducing the set of mandatory functions required within a GRADD, this model makes video display driver development easier and faster than it was for OS/2 Releases 2.1 and earlier. The only mandatory GHI functions in the GRADD architecture are initialization, return capabilities, return mode information, mode setting, and palette setting (if using 256 colors).

Graphics adapter support for direct access to video memory renders all other GHI functions optional. The GRADD model uses a software library called SOFTDRAW to simulate drawing functions (such as drawing bit maps and lines, and handling pointer support). Software simulation allows developers to write a driver in incremental stages. Once the mandatory functions are written, a developer can use

SOFTDRAW for optional functions that have not been written to use the accelerated features of the hardware. When an optional function is handled by a GRADD, the results can be compared with the results of the software simulation. This comparison gives developers a way to ensure that their GRADDs are producing correct output.

Most developers who write device drivers based on the GRADD model will be required to create only the hardware-specific code confined exclusively to the GRADD module. In the GRADD model, components that do not need direct access to the hardware are not located in the GRADD. This modular design makes it possible for developers to write new device drivers easily and quickly.

GRADD Model Components

Video Manager (VMAN)

Video manager Interface (VMI)

The VMAN component relies on a special protocol, called the Video Manager Interface (VMI), to receive requests from the translation layers. VMI consists of a small set of operations, each identified by a unique function number. Separate function numbers are required for each operation because VMAN exports only one entry point for the translation layers to communicate with VMAN. This exported function is called VMIEntry. VMIEntry expects four parameters from each function call it receives from a translation layer.

Because the VMIEntry function receives many different types of requests from the translation layers, the **gid** provides an ID number that identifies the GRADD to receive the operation and the **ulFunction** provides a function number that identifies the requested operation. The last two parameters **(pIn, pOut)** pointer to input and output data structures that are unique to each VMI function.

Most of the requests VMAN receives from a translation layer are passed directly to the appropriate GRADD. Each GRADD has its own exported function, called HWEntry, which is the same function type as VMIEntry (see Graphics Adapter Device Drivers for more information about GHI protocol).

Pointer Management in the Video Manager

VMAN is responsible for pointer management. When a pointer movement occurs, VMAN is notified by the VMI_CMD_MOVEPTR function. VMAN calls down the GRADD chain for the pointer update. The GRADD can either update the pointer or return to VMAN for simulation. If RC_SIMULATE is returned, VMAN uses the regular bitblit command to simulate the pointer movement.

VMAN tracks information about the current state of the pointer. All drawing VMI commands that affect the display surface must include the areas of the screen being updated by the primitive. VMAN uses this information to determine whether or not the pointer should be hidden before it passes a drawing request down the GRADD chain. On return from the drawing command, VMAN will restore the pointer if it was previously hidden.

Video Helper Functions

VMAN exports a number of helper services that GRADDs may use for common required functions. By using video helpers, a GRADD can avoid operating system specific calls. These helper functions are described below:

The VHAllocMem helper returns a 32-bit pointer to a piece of memory. The caller of this function supplies the byte count required.

The VHFreeMem helper frees memory that has been allocated via the VHAllocMem helper.

The VHLockMem helper makes code or data segments available for ring 0 interrupt-time processing.

The VHPhystoVirt helper converts physical address ranges to linear virtual address ranges for processes using VMI.

The VHMap helper aliases process memory to a global ring 0 context.

The VHMapVRAM helper converts the physical address of VRAM to linear virtual aperture for both process and global ring 0 context.

Translation Layers

Translation layers use the CHAININFO structure to gain access to the hardware capabilities and available modes. The VMI_CMD_QUERYCHAININFO function returns the CHAININFO structure.

When the VMIEntry function receives an operation from a translation layer, VMAN checks the function number and either handles the operation or forwards it to the appropriate GRADD. VMAN will handle the operation if the GRADD returns an RC_SIMULATE. The GRADD Model diagram shows how the VMAN component handles communication among the various components and the paths that commands can take during processing.

Translation Layer between OS/2 Graphics Engine and VMAN

The translation layer for the OS/2 Graphics Engine (GRE) is called GRE2VMAN. For a system that uses OS/2 as the dominant operating system service, GRE2VMAN is the first translation layer and the first component of the GRADD model to be loaded. When GRE2VMAN is loaded, it calls VMAN's VMIEntry function with a VMI_CMD_INIT. When VMAN receives a VMI_CMD_INIT for the first time, it loads the other GRADD model components.

Virtual VMI device driver (VVMI)

Video Manager (VMAN) creates a thread which is used to process all VMAN requests from the VDM. This thread is blocked by VVMI until a request is made, at which time the thread is unblocked and the request is serviced by VMAN.

Future Translation Layers

In the future, other graphics subsystems can be adapted to work in the IBM Operating System/2 operating system. To accomplish this, a translation layer must be provided (shown as 'n2VMAN' in the GRADD Model diagram). This translation layer must map the graphics primitives of the graphics subsystem to the appropriate VMI_CMD_ functions.

Translation Layer For Extensions

The GRADD Model can be extended using the VMI extension protocol. Using this protocol, a translation layer directs extension functions to a GRADD through the VMI_CMD_EXTENSION function. In order to accomplish this, a translation layer must be provided (shown as 'EXT2VMAN' in the GRADD Model diagram).

See Adding Extensions for more information.

Graphics Adapter Device Driver (GRADD)

This section describes how the Graphics Adapter Device Driver components interface with VMAN.

Graphics Hardware Interface (GHI)

The entry point and functions supported by a GRADD are referred to as the Graphics Hardware Interface (GHI). The differences between the VMAN protocol (VMI) and the protocol for the GRADDs GHI include the following:

The GHI is a subset of the VMI. The **ulFunction** parameter value changes to the appropriate GHI_CMD_ function name.

Each GRADD has an exported function, called HWEntry, which is the same function type as VMIEntry in VMAN.

HWEntry receives all of the operations from VMAN.

In the GRADD model, the GRADDs are the only components that have direct access to the video hardware. GRADD code uses the accelerated features of a graphics adapter. By returning RC_SIMULATE for a graphics operation, the GRADD gives the SOFTDRAW component permission to draw directly to the video memory of the hardware. The serialization of video memory is handled by the Video Manager through the the GHI_CMD_REQUESTHW function.

Each GRADD must process the following GHI_CMD_ functions:

- GHI_CMD_INIT
- GHI_CMD_QUERYCAPS
- GHI_CMD_QUERYMODES
- GHI_CMD_SETMODE
- GHI_CMD_PALETTE

* This function is mandatory only when a 256-color mode has been chosen.

The following GHI functions can return RC_SIMULATE and let VMAN handle the operations:

- GHI_CMD_BANK
- GHI_CMD_BITBLT
- GHI_CMD_LINE
- GHI_CMD_MOVEPTR
- GHI_CMD_SETPTR
- GHI_CMD_SHOWPTR
- GHI_CMD_TEXT

The GHI function numbers are assigned in such a way that they can be used as a zero-based index into a function jump table.

System Management (RAS)

Because performance is critical in a video graphics subsystem, all systems management and RAS hooks should be placed in a filter GRADD. This protects the majority of users, who do not need this support, from performance degradations caused by the addition of trace points and hooks.