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 syste

- How This Book Is Organized
- Asssistance
- Ordering Information

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.

From:
https://osfree.ru/doku/ - **osFree wiki**

Permanent link:
https://osfree.ru/doku/doku.php?id=en:ibm:gradd:index&rev=1403753921

Last update: **2014/06/26 03:38**