TITLE: Writing Device Drivers - Getting the Most out of OS/2
IBM Developer Connection News Volume 4
by Steve Mastrianni

OS/2 provides several base device drivers for retrieving information about the system configuration. The two most popular are TESTCFG and OEMHLP.

OEMHLP was designed to allow programs and drivers information about the system hardware configuration, enabling both drivers and applications to configure themselves based on the hardware. These functions have been recently documented in the Device Driver Source Kit, and are worth noting.

OEMHLP can retrieve information such as the OEM adaptation information, machine information, video display chipset, video fonts, EISA and ROM BIOS information, memory available, and several other tidbits of information.

TESTCFG provides additional functionality, allowing you to retrieve the bus type (EISA, ISA, Micro Channel), get the contents of the Micro Channel Programmable Option Select (POS) registers, the EISA card IDs, perform direct register I/O (without the need for an IOPL segment), and get a copy of non-system memory, that is, the memory between 640K and 1MB.

If you are interested in either of these base drivers, or device drivers in general, check out the DDK! The latest DDK, version 1.2, contains the source of TESTCFG . Refer to the documentation on the DDK and the source code for the latest information. Using these tools can save you a lot of time developing your driver.

See the Directory of this Newsletter for information about the DDK.

Steve Mastrianni is an Industry Consultant specializing in device drivers and real-time applications for OS/2. The author of Writing OS/2 2.1 Device Drivers in C, Steve is regarded as one of the industry's leading experts in OS/2 and OS/2 device drivers. Steve can be reached at CompuServe @ 73354,746 or Internet @ stevemas@vnet.ibm.com.

Device Driver Tips

TIP: Install a small bootable partition to use when developing a device driver on a single machine.

TECHNIQUE: Because many boot cycles are required for testing, install a small bootable partition that contains a line pointing to the device driver under development. No changes to CONFIG.SYS files are needed between booting into the test environment or the development environment. And, there is no risk of making the primary environment unbootable.

TIP: When installing a timer handler in your driver, don't call SetTimer or TickCount until the end of the Init section, just before returning to the kernel. Timer handlers usually reference DS-relative data items, and these items become dereferenced when the driver fails and returns 0 code and data offsets. What happens is that the timer handler is still in the list of timer handlers to be called, and gets called after the driver fails but before the timer handler is removed, causing a GP fault. An alternative is to always call ResetTimer if an error occurs, prior to returning the 0 code and data segment offsets, or make the call to SetTimer/TickCount the last operation in the Init section.

TIP: To set a breakpoint in a particular place in your driver, just insert an INT 3 instruction in assembly language, or write a simple assembly language function that is callable from C to perform the INT 3. Insert the INT 3 or call to INT3() in your driver source, recompile, link and reboot. Make sure you have the kernel debugger installed, or your system will not boot.

TIP: When designing your drivers, put a little extra effort in the design process to try to "layer" your approach. Attempt to separate the software specific parts of your driver from the hardware-specific portion, much like the ADD model. This layering approach will leave you with large portions of reusable code, and gets you thinking about reusability.