

Writing a Device Driver Made Easy

WinDriver™ - The "Java" of Device Drivers

Writing kernel mode device drivers to access hardware requires special skills, and requires that a separate device driver be written for each different operating system to be supported. Jungo introduces tools that automate this process, and create cross-platform device drivers.

WHAT IS A DEVICE DRIVER?

Protected operating systems (such as Windows™, Linux, and Unix™) isolate applications running in the application space ("User mode") by limiting their access to system resources. In these operating systems (unlike DOS), applications are not able to access hardware directly. Hardware access is only available to code running at kernel level (ring-zero, or "Kernel mode"). In order to access a hardware device from an application, a programmer must first write a "device driver".

Traditionally, this process includes the following steps:

1. Learn the internals of the operating system to be supported (Windows/Linux/VxWorks...)
2. Learn how to write a device driver on this operating system (DDK, DDI/DKI, etc...)
3. Master new tools for development/debugging in the Kernel mode
4. Write the Kernel mode device driver which does the basic hardware input/output
5. Write the application in the User mode, which accesses the hardware through the device driver written in the Kernel mode
6. Repeat steps 1-4 for each new operating system on



which the code should run

Looking at the process above, it can be a very tedious process and the maintenance of the code for each operating system can be an overwhelming task on its own. Some operating systems do not have support to develop drivers in a high level language such as C/C++; you have to live with assembly. This

leaves little room for improvement. Many a time writing a driver takes weeks and it only works for one operating system.

A typical scenario: you have written an existing DOS driver, forget trying to port it easily using the above (traditional) technique. It is just too much work!

What happens if you are fluent in Visual Basic or Delphi and do not have the time to learn C/C++ or assembly, how do you get over this hurdle?

WinDriver offers a solution to all of the above problems and more...

WHAT IS WINDRIVER?

WinDriver is the leading driver development toolkit, designed to enable you to create high performance PCI/Compact PCI/USB/ISA/ISA PnP/EISA based device drivers. Supported operating systems currently

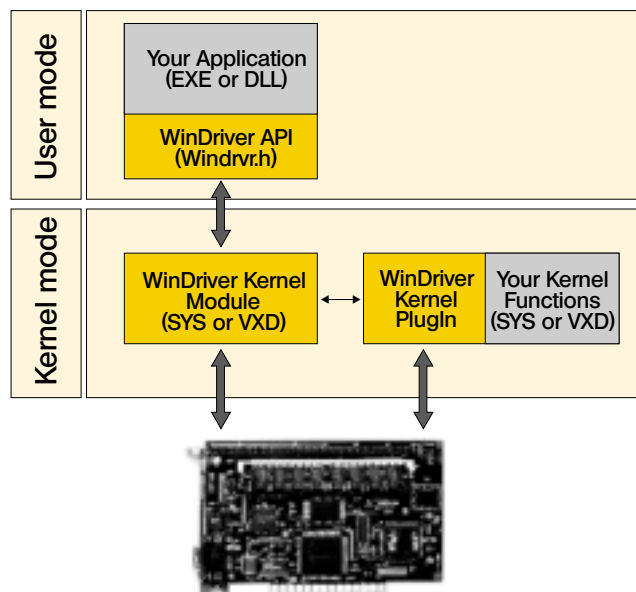


Figure 1. WinDriver's Architecture.

include Windows 9x, NT, NT Embedded, 2000, CE, Me, Linux, Solaris, OS/2 and VxWorks. There is no requirement for kernel or DDK knowledge needed.

WinDriver enables developers to create applications that access new or existing hardware, without the need to write a kernel mode device driver. This new technology saves months of work, and enables your driver to be source code compatible between different operating systems.

There is full support for interrupt and i/o handling, memory mapped devices, DMA transfers, Plug and Play, multiple board, multiple bus and multiple CPU handling.

WINDRIVER'S ARCHITECTURE

Your application accesses hardware through an API in the User mode. Functions such as `inport()` and `outport()`, full DMA transfers and a host of low level functions are available.

For high performance accesses, such as high-speed data streaming, or I/O mapped accesses your code can be moved from the User mode to the lower level (Kernel) mode, using the Kernel PlugIn feature. This feature then provides a very high-speed direct path to the hardware from your code.

WinDriver provides kernel modules for most of the common operating systems - Windows 9x, NT, 2000, Me, CE, Linux, Solaris, OS/2 and VxWorks. These kernel modules enable you to run your WinDriver based driver on any of the supported operating systems - just recompile and run!

This architecture enables you to develop and debug your entire driver code in the User mode. You can use any one of the above operating systems. You can step through the code using the standard debugging tools that are available in most development packages such as MS Visual Studio™, Borland Delphi™, Borland C++ Builder™ environments. A complete debug logger will track all driver calls to the hardware.

Porting your driver just became insignificant! Maintenance is now a "breeze"! Optimal development is now possible!

THE DRIVERWIZARD

WinDriver consists of a complete toolkit that includes a DriverWizard™ (which is a hardware interrogator, hardware detector, driver debugger and a code generator), sample code and complete manuals.

The DriverWizard is able to scan all busses for hardware that is present (ISA bus designs cannot be listed automatically, however ISA PnP will be scanned with no problems). You will then choose a device from the list presented. Based on the choice, you can then read and write to ports, registers, memory and listen to interrupts. A detailed debug logger shows and saves a text file with all calls and results of the interrogator.

Once a developer has interrogated the hardware and it is functioning as expected, DriverWizard can automatically create a cross platform hardware specific API, and a diagnostics application which can then be used as a skeleton driver.

Using the C/C++ output generator, DriverWizard produces the following elements in a subdirectory of your choice:

- XXX_files.txt - A readme files describing the files generated
- XXX_diag.c - A fully working skeleton application showing the use of the library functions created by DriverWizard
- XXX_lib.c - A general library of utility functions for device access used by XXX_diag.c
- XXX_lib.h - A header file for the utility functions
- Other project files for the development platform chosen

Where XXX is the name of the project.

To get a driver built for multiple platforms will simply require a recompile and build in the specific platform.

WINDRIVER APIS

WinDriver APIs for PLX, ALTERA, V3, AMCC, GALILEO and QuickLogic.

WinDriver includes ready-made libraries developed for the PLX, ALTERA, V3, AMCC and GALILEO PCI chipsets, which enable WinDriver users to create device drivers for any of the above PCI chipsets in an extremely short time. WinDriver's ready-made libraries enable both professionals and novice driver writers to dramatically increase productivity when developing hardware using one of the above PCI bridges.

SUMMARY

- **Cost** - Cut up to 90% of your development time on all your driver projects
- **Stability** - WinDriver has been field tested on thousands of hardware and operating systems configurations
- **Performance** - Achieve optimal kernel mode performance using the KernelPlugIn feature
- **Portability** - Your driver will run on any supported operating system. Just recompile!
- **Setting standards** - Same driver API and development environment in any operating system, platform and programming language
- **Vendor support** - Enhanced built-in support for PLX, V3, Galileo, Altera, QuickLogic, PLDA and AMCC

Download a free full-featured 30 day evaluation version from Jungo's web site at www.jungo.com ■

The article was written by Limor Shmerling, WinDriver / KernelDriver product manager.