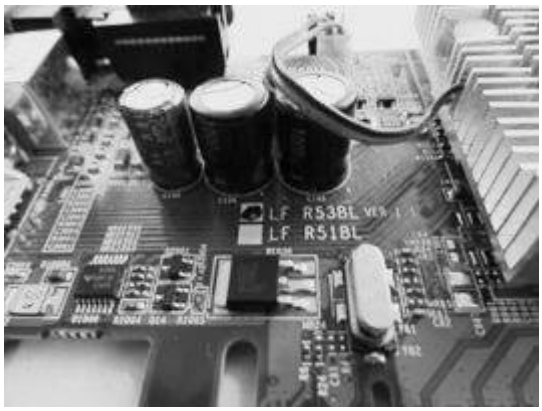
Driver

Device driver is a computer program that operates or controls a particular type of device that is attached to a computer. A driver provides a software interface to hardware devices, enabling operating systems and other computer programs to access hardware functions without needing to know precise details about the hardware being used.



A driver communicates with the device through the computer bus or communications subsystem to which the hardware connects. When a calling program invokes a routine in the driver, the driver issues commands to the device. Once the device sends data back to the driver, the driver may invoke routines in the original calling program. Drivers are hardware dependent and operating-system-specific. They usually provide the interrupt handling required for any necessary asynchronous time-dependent hardware interface.

The main purpose of device drivers is to provide abstraction by acting as a translator between a hardware device and the applications or operating systems that use it.

Writing a device driver requires an in-depth understanding of how the hardware and the software works for a given platform function. Because drivers require low-level access to hardware functions in order to operate, drivers typically operate in a highly privileged environment and can cause system operational issues if something goes wrong. In contrast, most user-level software on modern operating systems can be stopped without greatly affecting the rest of the system. Even drivers executing in user mode can crash a system if the device is erroneously programmed. These factors make it more difficult and dangerous to diagnose problems.

The task of writing drivers thus usually falls to software engineers or computer engineers who work for hardware-development companies. This is because they have better information than most outsiders about the design of their hardware. Moreover, it was traditionally considered in the hardware manufacturer's interest to guarantee that their clients can use their hardware in an optimum way. Typically, the Logical Device Driver (LDD) is written by the operating system vendor, while the Physical Device Driver (PDD) is implemented by the device vendor. But in recent years, non-vendors have written numerous proprietary device drivers, mainly for use with free and open source operating systems. In such cases, it is important that the hardware manufacturer provides information on how the device communicates. Although this information can instead be learned by reverse engineering, this is much more difficult with hardware than it is with software.

Microsoft has attempted to reduce system instability due to poorly written device drivers by creating a new framework for driver development, called Windows Driver Foundation (WDF). This includes User-Mode Driver Framework (UMDF) that encourages development of certain types of drivers—primarily those that implement a message-based protocol for communicating with their devices—as user-mode drivers. If such drivers malfunction, they do not cause system instability. The Kernel-Mode Driver Framework (KMDF) model continues to allow development of kernel-mode device drivers, but attempts to provide standard implementations of functions that are known to cause problems, including cancellation of I/O operations, power management, and plug and play device support.

https://drivernew.com