

# Connecting Computer Peripherals with a Focus on Plug-and-Play

Author: Aaron Schlicht  
Date: 2/3/2003  
Submitted to: Dr. Chris Taylor

# Table of Contents

1. Introduction .....	1
2. What is Plug-and-Play? .....	2
3. Integrating Busses with Plug-and-Play .....	3
3.1. ISA Bus .....	3
3.2. PCI Bus .....	5
3.2.2 PCI Configuration Registers .....	5
3.2.3 PCI Base Address Registers .....	6
3.2.3 PCI Interrupt Lines .....	7
3.2.4 PCI Boot Sequence .....	8
4. Plug-and-Play Qualities .....	8
4.1. Uniquely Identifiable .....	8
4.2. State Provided Services and Required Resources .....	9
4.3. Configurable by Software .....	9
5. Plug-and-Play Drivers .....	10
5.1. Bus Drivers .....	10
5.2. Filter Drivers .....	11
5.3. Function Drivers .....	11
6. Plug-and-Play Driver Stack .....	12
7. Plug-and-Play Manager .....	13
8. Interrupt Sharing .....	14
9. Human Usability .....	14
10. Conclusion .....	15
Works Cited .....	16

# 1. Introduction

The modern age of technology has brought a personal computer into nearly every home and workplace in the United States. A decent computer can be purchased at a very reasonable price. Computers are in the residences and jobs of the technologically savvy and the novice alike. With such a wide range of users, there is an even wider range of uses and devices connected to computers. Users have printers, scanners, digital cameras, external disk drives, mice, keyboards, network cards, video cards, monitors, and much, much more. Before computers were so common, people had to manually configure their system and install drivers to make each device operational. The savvy could handle this issue because they understood terminology, how computers work, and where to go to make appropriate configurations. The novices would have been lost and too dependent on others if the situation had not changed. Operating systems and hardware were redesigned to allow easier connection of devices. Through the designing, a specification called Plug-and-Play (PnP) was created.

The ease of device connection is highly dependent on the operating system installed on the computer. The modern Windows and Macintosh operating systems are extremely user friendly and have advanced Plug-and-Play capabilities. Linux is a little behind, but it is not designed to be a desktop operating system for the novice user and therefore the kernel was not built around the concepts of Plug-and-Play. Plug-and-Play may appear to be working on Linux, but actually there are some behind the scenes tricks to fake the functionality. However, these tricks do not offer full support.

## 2. What is Plug-and-Play?

What exactly is Plug-and-Play? Plug-and-Play allows a device to be connected to a computer and be automatically recognized, configured, and enabled for use with very few demands on the user. It has the responsibility of automatically matching hardware devices with device driver software on the system and establishing communication channels between the physical devices and their drivers. The channels or resources exist as part of the bus to which the device is connected. There are four main resources allocated by Plug-and-Play to complete the connection and configuration: Input/Output (I/O) addresses, Direct Memory Access (DMA), Interrupt Requests (IRQs), and memory regions. I/O addresses are “memory locations that provide an interface between the operating system and the I/O device (peripheral)” (Blanchard). The operating system and device communicate using the I/O address and each device must have its own address. Direct Memory Access allows a device to directly read and write to memory without bothering the CPU. This allows the CPU to be used for other more important operations. IRQs are hardware driven interrupts that are wired to the CPU or an interrupt router and then the CPU. IRQs allow a device to signal the operating system that it is ready for something and needs attention. IRQs avoid CPU expensive polling techniques by only communicating with devices when required. The fourth resource, memory regions, can be areas of memory on removable devices, or defined areas in system memory.

### 3. Integrating Busses with Plug-and-Play

There are a few main busses used to connect device cards to motherboards. There is the older and now phased out ISA bus, the PCI bus, and USB, or Universal Serial Bus. USB is mainly used to connect peripherals with run time connection support, while ISA and PCI are used to connect cards to the motherboard. The ISA and PCI buses will be focused on.

#### 3.1. *ISA Bus*

Since ISA was not designed with Plug-and-Play in mind, some tricks had to be created to communicate with and configure devices. One of the tricks used to configure devices is to temporarily assign a handle or Card Select Number (CSN). The process isolates each device on the shared bus. Every device needs to have its own address in order for messages to be passed from the Plug-and-Play BIOS or manager to the correct device for configuration to properly occur. The address is only used for configuring devices, hence why it is a temporary handle.

The process of assigning handles is relatively simple, but requires only one device to be listening for the handle at a time. Each non-legacy, ISA device contains a unique serial number stored in its ROM by the manufacturer. The Plug-and-Play manager starts the process by requesting every device on the bus to simultaneously send its serial number. A stream of ones and zeros is sent by each device to represent its number. The stream starts with the highest order bit being sent first. When a device sends a 1, it knows it is sending. When a device does not send anything, representing a binary 0, it is

listening. If a device sends a 1 while it while another device is listening, the listening device stops sending its serial number. By the time the lowest priority bit from the serial number is sent, only one device remains. This device is now the only device listening on the bus and also has the greatest valued serial number. A handle can safely be assigned to it without another device receiving and using the same handle. Handle values are assigned starting with one and can range up to 255. The device with the greatest valued serial number receives the lowest valued handle. If a device cannot be identified and isolated, it is assigned a value of zero. Legacy devices with no Plug-and-Play support would normally be assigned the zero handle.

The isolation sequence resumes with the remaining devices. Devices with low serial numbers are eliminated each round leaving the devices with greater valued serial number to get assigned a handle. The entire process continues until all devices on the ISA bus are assigned handle. When a computer is booting, it will go through this process and configure all the Plug-and-Play devices. Once all the devices are configured, the handles are forgotten. Every ISA device must be isolated and assigned a handle again to inspect the configuration or reconfigure the devices while the system is underway.

All of the temporary handle assignments must take place because the ISA bus has no configuration address space like the PCI bus. Three I/O addresses, each one byte in size, are assigned. One is for reading, the second is for writing, and the third is for addresses. A device's handle is sent on the configuration address telling a single device to listen. All other devices ignore the following data transfers because they did not receive their handle and a notification to listen. The address of a register to be configured by Plug-and-Play is sent to the temporary configuration address port. The device is

configured as needed by asking for data on the read port or by writing data to the write port address. Data transfers usually involve manufacturer information about the device, the system IRQs, and which driver supports the device.

The phasing out of ISA devices can now be understood. They must go through a complicated process to be configured by a Plug-and-Play manager by being assigned a handle and then read and write data through temporarily assigned memory addresses. On top of the painful configuration experience, the bus is also much slower than the PCI bus. For these reasons, motherboard manufacturers, such as Intel, have stopped installing ISA slots and busses (Bullough).

### **3.2. PCI Bus**

The PCI specification supplies all of the necessary resources for a simpler Plug-and-Play environment. The PCI Plug-and-Play specification requires three main components. The first is the configuration registers on the physical PCI device. The second is the computer systems BIOS configured for PCI Plug-and-Play. The third is the system software (Girard). Each component is important for configuring and using PCI devices under a Plug-and-Play system.

#### **3.2.2 PCI Configuration Registers**

Every PCI device conforming to the specifications contains 64 bytes of configuration registers. Of the 64 bytes of registers, there are four distinct types important

for Plug-and-Play. The four registers are as follows: the vendor ID, the device ID, the base address register, and the interrupt line register (Girard).

The combination of the vendor ID and the device ID are used to uniquely identify a PCI Plug-and-Play device. Each vendor is assigned an ID by the PCI SIG organization to ensure each vendor has a unique ID. This also allows parts to be tracked back to the manufacturer. Device IDs are in turn assigned by the vendors. Since every device is assigned a unique vendor number and a unique device number, it can be assured that every PCI device has a completely unique set of IDs. Uniqueness is very important from the Plug-and-Play standpoint.

### **3.2.3 PCI Base Address Registers**

Base address registers identify the system resources required by the device upon booting. “Each device may use up to six [base address registers] to identify up to six individual address spaces of various sizes to be used by the device” (Girard). The base address registers contain two different pieces of information depending on the state of the device. Upon startup, the addresses contain the type of resources required by the device. The least significant bit in the registers informs whether the resource is input/output or memory space. The other bits in the address identify how much space is required for this resource. Once Plug-and-Play is configured for the device, the base address registers fit their name appropriately. They contain the base addresses of the resources assigned to that register.

### 3.2.3 PCI Interrupt Lines

The interrupt line register is used to contain the system hardware interrupt line used by the device. The interrupt lines are actually hard wired through an interrupt router on the motherboard. Each PCI device has four interrupts available to trigger. They are called INTA#, INTB#, INTC#, and INTD# (Lawyer). Each one is a specific pin on a PCI device. The interrupt router handles four different interrupt lines that do not have to specifically wired to a specific pin. These interrupts lines are shared with the other PCI devices, as noted in the PCI bus specifications. If interrupt sharing was not possible on a PCI bus, then a system would need four separate interrupt lines for each successive device. Interrupt sharing will be discussed later.

There are ways to prevent too many interrupts being from called on the same interrupt line. Let us call the four interrupts lines handled by an interrupt router 1, 2, 3, and 4. The interrupts could be wired so that all INTA# pins are connected to interrupt line number 1. This is very inefficient because the PCI specifications say that if a device only needs one interrupt, then it should use the INTA# interrupt line. If it needs two interrupts, then it should use INTA# and INT#B. The same sequence applies for three and four interrupts. So, the INTA# interrupt will be used most often causing too many interrupts on line number 1. The operating system could handle excessive interrupts on one line, but it does not make sense to do this if a more efficient way is available. The interrupts are usually wired in a more efficient manner. Interrupt line number 1 will be hooked to INTA# pin of device one, INTB# pin of device two, INTC# pin of device three, and so on. This allows more efficient use of the interrupts lines (Lawyer). The BIOS needs to be configured so it knows how the motherboard is wired.

### **3.2.4 PCI Boot Sequence**

The PCI BIOS controls the initial configuration and registration of PCI devices during the boot sequence. The BIOS starts by polling every PCI slot for a valid PCI device. If a device returns its vendor ID, then it exists, otherwise the value 0xFFFF is returned. When a valid vendor ID is returned by the device, other information is polled. The base address registers are queried and initialized, and the interrupt register is read to see which interrupts are needed (Girard). The BIOS makes the system aware of the resources utilized by the PCI devices. Since each device is assigned unique I/O addresses and memory regions, the system can use the devices without conflicts. A conflict would occur if two or more devices were assigned the same resources. Multiple devices would write to the same memory regions and overwrite the existing data from a different device. Devices would be unresponsive or the system could crash.

## **4. Plug-and-Play Qualities**

### ***4.1. Uniquely Identifiable***

There are three qualities a Plug-and-Play device must have (Plug and Play Technology). The first is that it must be uniquely identifiable. A device must have a unique address in one of its registers to allow the rest of the computer to communicate with it. The address is defined in a two step process. First, the register in a Plug-and-Play device is assigned the device's I/O address. Setting this on the ISA process is done by the previously defined isolation process. The second part of the process involves setting the

address in an operating system's device drivers. Problems occur when users manually set the address of Plug-and-Play devices because only the driver address is set, unless additional care is taken to set the devices address as well. The address defined in the drivers and on the device must match for any communication to occur. Without the address, it would be like trying to find the correct house on a street, but no houses have house numbers.

#### ***4.2. State Provided Services and Required Resources***

The second feature a Plug-and-Play device must have is the ability to state the services it provides and state the resources it requires. The services are just telling the operating system that it is a modem or sound card or some chip on the motherboard. Each device contains codes defining what it is and what version of Plug-and-Play it supports. Older legacy ISA cards that were made before Plug-and-Play may have to be manually configured by setting a jumper before the computer boots.

The resources a device requires are defined in the base address registers in a PCI device and on similar registers in an ISA device. Resources include I/O addresses, memory, and interrupts.

#### ***4.3. Configurable by Software***

The third thing a Plug-and-Play device must do is allow itself to be configured by software. After a device notifies what resources are required, the operating system's Plug-and-Play manager assigns resources specific for that device. If a device specifies

that it requires a certain amount of memory, then it is assigned a memory range and this range is updated on the device. The memory can be system memory or I/O memory on the device itself. Even if the memory is located on the device, it still needs to be assigned an address range so as not to interfere with defined system memory addresses. Usually the I/O memory on a device is given a very high value to signify its difference. Both the device and its driver must know the memory range to keep communication open and for proper functionality.

## **5. Plug-and-Play Drivers**

There are a multitude of driver types for working with Plug-and-Play devices. Each one performs a different way of communicating with the same device. The different types are as follows: bus drivers, bus filter drivers, and function drivers. Each one handles interrupt request packets in different ways.

### **5.1. Bus Drivers**

As might be expected by the name, bus drivers are the lowest level drivers. They perform such tasks as powering up devices, powering down devices, and completing interrupt request packets. The bus driver services a bus controller, an adapter, or any device that has child devices. There is usually one bus driver for each type bus on a system (Plug and Play Technology).

## **5.2. Filter Drivers**

A filter driver sorts I/O requests for a bus, a device, or a class of devices. Filters can exist in any number and are optional. They can be placed above or below function drivers on the driver stack. Lower level filter drivers are usually used to modify the behavior of device hardware. An example is adding a low level filter on mouse movement requests. A mouse can be set up to accelerate when moved quickly. The low level filter would realize the mouse movement speed and modify the output to the function driver appropriately. Higher level bus filters, filters above the function drivers, usually add additional functionality to a device. For example, a filter could add additional security checks (Plug and Play Technology).

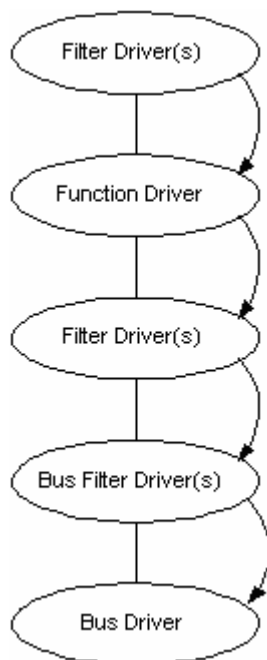
## **5.3. Function Drivers**

Function drivers are the main operational drivers for a device. They provide all of the interfaces for working with devices. Function drivers are normally required, but a device could be controlled directly by a bus driver. Usually function drivers are written in driver/minidriver pairs. In Microsoft's case, the main driver is written by them and the minidriver is written by the device manufacturer to provide specific functionality for their device (Plug and Play Technology). For example, Microsoft writes a driver to handle keyboards. The manufacturer of the keyboard writes a driver to support their exact hardware. Their hardware may have added features such as extra buttons for automatically launching an email client or skipping to the next track on a music CD.

## 6. Plug-and-Play Driver Stack

The most significant specification for Plug-and-Play drivers is that a single driver cannot assume it is the only driver handling the device. It must treat any interrupt request packet it receives and pass the signal on to the next driver. Drivers are setup in a layered architecture as a stack. The bus drivers are at the lowest level. Next up the stack are the bus filter drivers. The next order up the stack is a layer of function filters followed by the function drivers. Above that are more function filters. The layers of the stack are shown in the following diagram.

Figure 1: Model of the Plug-and-Play Driver Stack (Plug and Play Technology)



Every driver in the stack has the opportunity to respond to a Plug-and-Play interrupt request packet. If only one driver responded to the request and sent out a response complete message, then none of the other drivers would have a chance to add their functionality to the device. Sending out the request complete message becomes the responsibility of the lowest driver on the stack, the bus driver.

## 7. Plug-and-Play Manager

The Plug-and-Play manager is the operating system's controller for Plug-and-Play devices. It is responsible for sending interrupt request packets and configuring the devices. The Plug-and-Play manager keeps track of devices in a hierarchical tree fashion. Devices connected to a certain type of bus are counted as children of the bus controller. The tree is updated as devices are added or removed from the system. To discover devices on a particular bus, an enumeration request is sent to the bus driver for that bus. The enumeration process queries for devices on a PCI bus and goes through the isolation process for an ISA bus. The enumeration process is much like the process the BIOS does when booting a system, only now the devices are configured by the operating system and not just initialized by the BIOS. For the system to gain access to a Plug-and-Play device, it usually must go through the Plug-and-Play manager. When the manager gets a request from the system, it sends an interrupt request packet to the appropriate bus on the tree. Since multiple devices receive the interrupt request (IRQ), something must be done to handle the sharing correctly.

## 8. Interrupt Sharing

Interrupt lines can be shared among multiple devices on most bus types. ISA is limited in this way because interrupt sharing is not part of the specification. Interrupt sharing is done by having every device driver check if the interrupt was meant for its device. The driver checks a register on its device to see if the device was the one that triggered the interrupt. ISA devices do not include a register informing they triggered an interrupt. If a device did not trigger the interrupt, then the driver simply does nothing. If its device did in fact trigger the interrupt, then it checks why the interrupt occurred by accessing another register. The device could have filled a buffer which needs emptying or needs more data sent to it. This process works the same way whether receiving an interrupt from the hardware or an interrupt request packet straight from the Plug-and-Play manager.

## 9. Human Usability

So what do all of these Plug-and-Play specifications mean for the average computer user? Plug-and-Play makes life easier. It allows a user to connect a device to a computer, even at run time, and have the computer automatically configure the device. If the device is an ISA or PCI card, the computer should be off during installation. When the computer boots it recognizes there is a new device connected because the old configuration files are saved. The system is examined for matching drivers. Operating systems like WindowsXP usually have a general driver for any device type that could be

connected. Vendor specific minidrivers may need to also be installed for full functionality.

The beauty of Plug-and-Play is allowing devices to be connected at run time. The newly connected device receives power from the bus and sends an interrupt signaling its arrival. The Plug-and-Play manager identifies, configures, and activates the device without any user interaction (Shanley 182). An example is inserting a keychain storage device into the USB port. The computer recognizes the arrival of a new device, identifies that it is an external storage device on the Universal Serial Bus, and activates the device. The only thing the user had to do was physically insert the storage device.

## **10. Conclusion**

Plug-and-Play is an important part of the modern operating system. It allows users of any background to easily connect devices to computers. Compliant devices are required to have unique identification numbers, be configurable by software, and state the resources they require. From the information on the device and software drivers, the Plug-and-Play BIOS and manager can configure and activate devices for use by the system. Since software is managing the resources, device conflicts are avoided and hardware jumpers do not have to be configured. Hopefully as technology advances the concepts of Plug-and-Play continue to be applied for user friendly computers.

## Works Cited

Bullough, Stacie. "Desktop Motherboards." Email to the author from Intel. 22 Jan. 2003.

Girard, Tony. "Plug and Play Operation of the PCI Local Bus." 1 Feb. 2003.  
<<http://www.signatec.com/products/appnotes/an003.asp>>.

Lawyer, David S. "Plug-and-Play-HOWTO." 2 Jan. 2003.  
<<http://en.tldp.org/HOWTO/Plug-and-Play-HOWTO.html>>.

"PCI Local Bus Technical Summary." *TechFest*. 6 Jan 2003.  
<<http://www.techfest.com/hardware/bus/pci.htm>>.

"Plug and Play Technology." 1 Jan 2003.  
<<http://www.microsoft.com/hwdev/tech/pnp/default.asp>>.

Shanley, Tom. *Plug and Play System Architecture*. Addison Wesley. 1998.

Webb, Thomas. "Linux Plug and Play." *WordWonder.com*. 6 Jan 2003.  
<<http://wordwonder.com/lxnpnp.shtml>>.