The Plug and Play Framework: Advancing the PC Architecture Backgrounder

September 1993

Microsoft Corporation

Table of Contents

Introduction

Configuring PC hardware and operating systems has become a significant problem in the PC industry, resulting in customer dissatisfaction and increased support costs, and possibly affecting PC market growth. A broad-based group of companies within the industry is tackling this problem with the development of the Plug and Play framework architecture. This paper describes the problem, the requirements for Plug and Play, the work that has been done so far, and gives a technical overview of a complete Plug and Play system architecture.

The Current PC Installation/Configuration Problem

For anyone but a trained technician, installing or configuring a device on a PC today is a daunting exercise that may be difficult, time-consuming and frustrating. To perform even common configuration tasks the computer user must understand arcane terms that describe the technical details of the inner workings of the PC, such as IRQ lines, DMA channels, base memory addresses, I/O ports and more. Cryptic records of the PC anatomy such as autoexec.bat, config.sys and *.ini files must be examined and modified to make most changes. The process is so intimidating that it discourages nearly all computer users from upgrading components of their PCs or adding some new capability to their systems without expert help. Those who try frequently fail and tell others about it, perpetuating the PC’s reputation as a complex, intimidating, and difficult-to-use product.

The root of the problem is that when the PC was designed, no architecture was defined for installing, identifying and configuring hardware devices, nor for integrating hardware and operating system software. One of the great strengths of the PC is the wide variety of add-in devices; yet the hardware and software configuration problem is compounded by the fact that the typical PC contains devices made by a multitude of vendors. The hardware, operating system and applications can’t tell what’s in the system, and the hardware can’t tell when conflicts exist between different devices trying to share the same system resource. Until now the PC industry has left customers to struggle with complicated and inadequate technical documentation, rather than designing intelligence into the PC itself to handle installation and configuration tasks without user intervention.

Today’s PCs based on the ISA bus architecture — the most popular expansion bus in the PC industry — lack any mechanisms for configuration management. The ISA bus architecture requires the allocation of resources such as memory and I/O address spaces, DMA channels and interrupts among multiple ISA cards. However, the ISA specification has no standard hardware or software interface for allocating these resources. As a result, configuration of ISA cards is typically done with “jumpers,” manual changes to the card that change the decode maps for memory and I/O address space, and steer the DMA and interrupt signals to different pins on the bus. In addition to making these manual changes to the hardware, the user is also usually required to edit system software configuration files. Even for software-configurable cards, the user generally needs to determine the correct settings and enter them into an install program to set the card.

With the increasing popularity of add-in boards such as fax cards, audio cards and CD-ROM drives, users are putting more complicated options into their systems and running into conflicts among the settings for system resources more frequently.

Alternative bus architectures (e.g., Micro Channel® and EISA) have hardware and software mechanisms to identify the resources requested by a card and resolve conflicts. But these mechanisms are not compatible with the installed base of PCs with ISA card slots. Moreover, these bus architectures are considerably more expensive than ISA designs and lack the wide range of add-on and peripheral device choices available for ISA systems. The configuration mechanisms are also inherently hard to use and functionally limited because they are not integrated with the operating system. When adding or removing devices, end users must first configure the hardware using a character-mode configuration utility that requires a set of floppies for the configuration program and for each card. These floppies are frequently lost or separated from the PC. Subsequently, each operating system driver needs to be set up separately. The setup mechanisms require users to restart their systems to make any sort of configuration change — a major limitation for notebook computer users who want the convenience of changing devices and configurations without interrupting their work.

Some vendors include elaborate installation programs with their add-in cards. These programs attempt to identify the installed hardware and configuration. However, they only work reliably with cards made by that vendor, and with simple systems that do not contain many option cards. Also, many PCs use cards from multiple vendors, and the installation programs often amount to little more than a data entry screen.

Systems using the ISA bus architecture have only a limited number of system resources, particularly IRQ lines and DMA channels. Furthermore, many ISA cards are designed to be able to use only a limited number of the possible choices. Thus someone has to determine a workable combination — one that meets all the requirements and does not cause conflicts. Unfortunately, that task often falls to the user.

The customer's problem has been helped somewhat by the trend toward providing more full-featured products with preconfigured hardware and preinstalled software. But this approach is inflexible and limited because it does not accommodate customers who want to upgrade their systems after purchase, nor does it address the technical requirements of new form factors and system designs. Mobile systems in particular demand greater hardware and software integration, in order to deliver dynamic insertion and removal of devices, hot docking and instant-on. The configuration solution must be very general and flexible to accommodate the mix of device architectures prevalent on today's system designs, from local bus to ISA to PCMCIA.

The current situation is costing the industry millions of dollars in lost sales and increased costs. Frequently, when customers try to add a new device, the result is an expensive product support call and a negative perception of the product manufacturer. The high rate of problems encountered by users generates a tremendous support burden — for example, at Microsoft nearly *half* of the support calls for the Microsoft® Windows™ operating system version 3.1 are related to installation, configuration, and hardware or software integration. Customer frustration with the configuration process depresses demand for add-on and upgrade products. For businesses, the high cost of supporting PCs inhibits the penetration of PCs in the workplace and diverts MIS personnel from focusing on using computer technology to solve business problems. If the industry can solve this problem, it will result in more satisfied customers, more capable PCs as users add new functions, and new sales as the market for new PCs and add-on devices is expanded.

The Solution: A Plug and Play Framework Architecture for PCs

The solution to the PC installation and configuration problem is the concept called "Plug and Play." In practice, Plug and Play is both a design philosophy and a set of PC architecture specifications. The ultimate goal of Plug and Play is to enable changes to a PC configuration with no intervention by the user. A Plug and Play system has a number of characteristics. First, any installation is a simple, fail-safe operation. For devices, the installation is automatic — plug the device in, turn on the system, and it works. With a Plug and Play system, the user can insert and remove certain devices (such as PCMCIA cards), or connect to or disconnect from a docking station or network, without restarting the system or fiddling with configuration parameters. The system figures out the optimal configuration, and applications automatically adjust to take full advantage of the new configuration. Take your notebook with infrared communications features into a room with an infrared printer, and your applications are ready to print.

Design Requirements for a Plug and Play System

To deliver the Plug and Play functionality described above, the PC industry has been working to develop an architecture that addresses several critical design requirements.

The goal of a Plug and Play architecture is to provide **easy installation and configuration of new devices**. Plug and Play devices must be capable of identifying themselves and declaring their services and resource requirements. This information enables the operating system to determine and establish a working configuration for all devices on the system and load the appropriate device drivers without any user intervention. For customers, installing a new device will then be as easy as plugging it in, turning on the system, and if necessary inserting a floppy disk upon request by the system. In the event that a driver is already present on the system, even the latter step would not be required.

The system must be able to accommodate **seamless dynamic configuration changes**. This capability is critical for mobile systems. The Plug and Play architecture must allow hot docking and insertion of devices. When a device is inserted, the operating system must recognize the new device, its services and requirements, as well as load the necessary device drivers. Applications must be notified about dynamic events, so they can take advantage of the new functionality or stop attempting to use unavailable devices. The user should not be required to turn off and reboot the system, and should not be asked to intervene in the configuration process unless the required resources are not available to the new device.

The Plug and Play architecture must provide **compatibility with the installed base of existing (non Plug and Play) systems and peripherals** while still meeting the goal of easy installation and configuration of new devices. To accomplish this goal, components in the Plug and Play architecture must be able to accommodate the lack of device-reporting mechanisms for non Plug and Play devices. Information about such devices must be stored in the system, and devices that cannot be software-configured must receive first priority in resource allocation. When unresolvable conflicts occur, the system should guide the user through device-configuration options.

The Plug and Play architecture must **support existing classes of devices and be extensible to new classes of devices** that may arise in the future. Examples of existing classes of devices include those on the ISA bus, EISA and Micro Channel devices, and SCSI devices. Examples of new classes of devices include local buses such as PCI and VL bus, and new types of devices like PCMCIA cards and Access bus. It is important for Plug and Play to encompass all devices currently in wide use, and be able to support devices that may be important in the future.

To make the solution comprehensive, the Plug and Play architecture must be **open and independent of specific operating systems and hardware implementations**. The components of the Plug and Play architecture must be based on published interfaces and abstracted to a level that enables the architecture to accommodate different bus and device architectures existing today, as well as future designs.

To make the solution economically feasible for the industry, the Plug and Play architecture must **reduce hardware complexity, and increase hardware flexibility.** The availability of clear design guidelines and standardized interfaces between system components will provide an economically attractive solution for system vendors.

Plug and Play must be **inexpensive to implement**. In particular, it must not significantly increase the per-unit cost of PCs or devices.

Plug and Play: Progress to Date

Plug and Play development is being performed by a broad-based group of companies representing all major segments of the computer industry. The common goal of this group is to develop an architecture for system configuration that meets all the design requirements defined above. A comprehensive solution addressing the problems of today while accommodating the innovations of tomorrow is being developed by involving all components of the PC — the base system, the BIOS, I/O architectures, devices and drivers, and the operating system.

The Plug and Play effort was formally introduced in March 1993 at the *Windows* Hardware Engineering Conference to more than 1,300 attendees. Microsoft and Intel Corporation presented the general Plug and Play framework as well as the first application of that framework in the form of the Plug and Play ISA Specification (version 0.8). More than 3,000 copies of the Plug and Play ISA specification have been distributed, and feedback has been received through more than 500 pieces of electronic mail on the “plugplay” Internet alias.

Microsoft and Intel revised the specification based on feedback they received, and released version 0.9 of the Plug and Play ISA Specification in April 1993. Those who provided feedback were invited to a full-day design review, which resulted in the completion of version 1.0 of the specification. The design preview was attended by representatives of more than 60 companies. The first implementation workshop to discuss and teach implementation of the specification, held in June, was attended by more than 150 engineers from more than 80 hardware vendors. Microsoft also released the MS-DOS® and Windows 3.1 Plug and Play Driver Interface Specification for system vendors who would like to provide a partial implementation of Plug and Play functionality in systems based on Windows 3.1. At the same event, Intel announced its plans to release a configuration utility for Windows 3.1 based on that interface. Two Plug and Play ISA cards are available today: a SCSI card from Future Domain and an audio card from Intel. Many more Plug and Play ISA cards are expected before the end of the year.

The most recent Plug and Play events have been the *Windows* Strategy Design Review held in July 1993 at Microsoft headquarters and the Intel Plug and Play Software Briefing held in August 1993 at Intel headquarters. Nearly 150 technical executives and system architects from 100 key hardware vendors attended the two-day Microsoft Design Review event, which covered the overall Plug and Play framework in detail. This event focused on the broad Plug and Play architecture in a future Plug and Play operating system. Technical feasibility of the Plug and Play ISA design was shown when four Plug and Play cards from various vendors were automatically installed and simultaneously activated in a live demonstration that used an early testing release of a future version of Microsoft Windows. Participants reacted very positively to the overall
Plug and Play framework and indicated strong support for moving forward to implementing the framework into their products. Nearly 100 technical executives attended the Intel event, which focused on delivering Plug and Play functionality with today’s operating systems by providing driver development kits and configuration utilities. Intel released a beta version of its Plug and Play development kit at the briefing.

A broad coalition of system and component manufacturers is currently working to provide Plug and Play functionality across a wide range of devices. Specifications are being written for PCMCIA, SCSI, PCI, ECP, EISA, Micro Channel, VL and IDE architectures. All of these specifications will be subjected to an open review process similar to the one used to review the ISA specification.

A CompuServe® forum (GO PLUGPLAY) has been established to support the industry-wide Plug and Play development work. Copies of all the final specifications are posted there as well as certain specifications in progress. The CompuServe forum is the main industry-wide support source for Plug and Play development.

The Plug and Play Solution

A General Framework

To design a Plug and Play solution that meets the requirements stated above, an architectural framework is needed that provides a complete definition of the functions each system component performs in the configuration process — BIOS, operating system, devices and device drivers — and how these components will interact with each other.

The basic functions that must be performed are readily apparent. Devices must be uniquely and positively identified, their resource requirements must be read, and their drivers loaded. Resources must be allocated and re-allocated when devices request identical resources. If any change is made to the system configuration, the process must be repeated. There must be centralized management of this entire configuration process to ensure complete coordination.

To create a Plug and Play framework, it is helpful to start by defining a model that describes the general characteristics of a system and the functions which must be performed for the system to be “plug and play.” This model is presented in this discussion from the perspective of the operating system, due to the central role it must play in configuration management.

Today’s PCs consist of multiple bus and device architectures coexisting in a given system. These systems can be viewed using a “tree” model to describe the relationships between these buses and devices. In the system tree, buses and devices may appear at different levels of the system hierarchy in multiple “parent-child” relationships; may require system resources; and may provide resources that are understood, or not understood, by their parents. The relationships are complicated by the fact that some of the buses (SCSI, PCMCIA) are actually hidden from their parents until they are initialized.

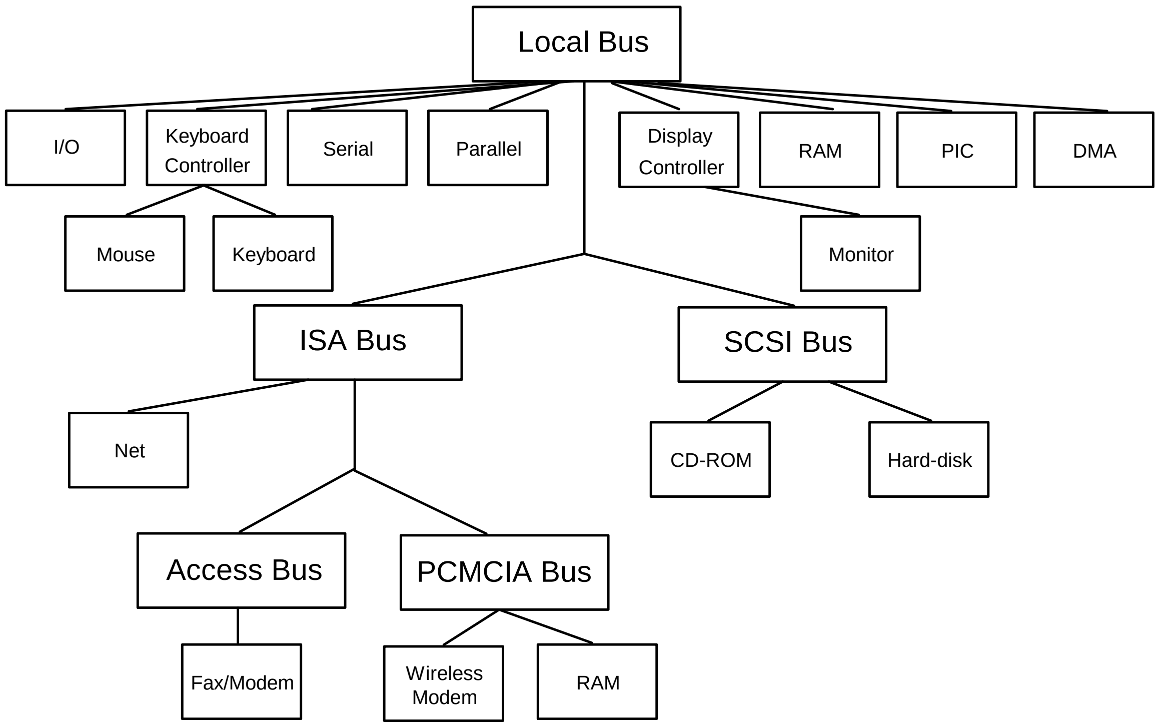

Figure 1: Tree model of an arbitrary PC system

Each branch in the tree defines an object that must be addressed by the Plug and Play framework. This object is called a “device node.” Configuring the device node requires the following information:

- A unique identification code
- The device node resource requirements
- The resources allocated to the device node
- Whether the device node is a bus (a bus has “child” device nodes)

The identification code is simply a string that uniquely describes the device. Each resource requirement must identify both the **resource type** (such as IRQs and memory ranges) and **constraints** associated with that specific resource. For example, certain devices may require specific IRQs. Also, constraints may have some interdependencies — for example, a COM port may require either IRQ3 and I/O port 02F8, or IRQ4 and I/O port 03F8.

If the device is a bus, the system must identify the additional device nodes associated with that bus and keep track of the device node's resources. This is important even if these are non-shareable resources (such as SCSI ID) because it enables the system to maintain a central database with all configuration information, so device drivers can access the database to learn about their assigned resources (e.g., SCSI ID).

Managing the Configuration of the PC

The tree model defines the **device node** as the fundamental data structure that can be manipulated by components of the system to manage installation and configuration of devices. The next step toward building a framework for a Plug and Play architecture is to define the configuration process that will manipulate the device nodes to establish a working system configuration. This process will in turn define the system components required to deliver Plug and Play functionality.

A certain amount of configuration must be performed by the system BIOS during the power-up phase. In order for the system to boot, the BIOS must at a minimum configure a display device, input device and initial program load device, passing the information about each of these device nodes to the operating system for additional configuration of the system.

The operating system must continue the configuration process by identifying every device node on the system and its respective resource requirements. Each non-boot device should be inactive upon power-up so that the operating system can identify any conflicts between the resource requirements of different devices before configuring them. Information about each device node must be stored by the operating system in a central database. The operating system then must load the device drivers for each device node.

In the event that different devices require the same resources, the devices must be able to provide information to the operating system about alternative resource requirements, which the operating system will in turn use to identify a working system configuration. Once the resource conflict has been resolved, the operating system must store the new configuration information in the central database and notify the device drivers of the new resource assignments.

If a change occurs to the system configuration during operation (e.g., the insertion of a PCMCIA card or a docking event) the relevant bus (PCMCIA or BIOS) must be able to notify the operating system of the event so that the operating system can configure the new device. Applications must be able to respond to configuration changes to take advantage of new devices and to cease calling devices that have been removed.

Overview of the Architecture and Components

In order to implement a fully Plug and Play system, it is apparent that changes are required to all components of the PC system, including the BIOS, operating system, devices and device drivers, and applications. The following discussion outlines the architecture that has been developed to date to deliver the benefits of Plug and Play, and the functions of each of the components defined by that architecture.

The general structure of the Plug and Play architecture is shown below:

Plug and Play Architecture

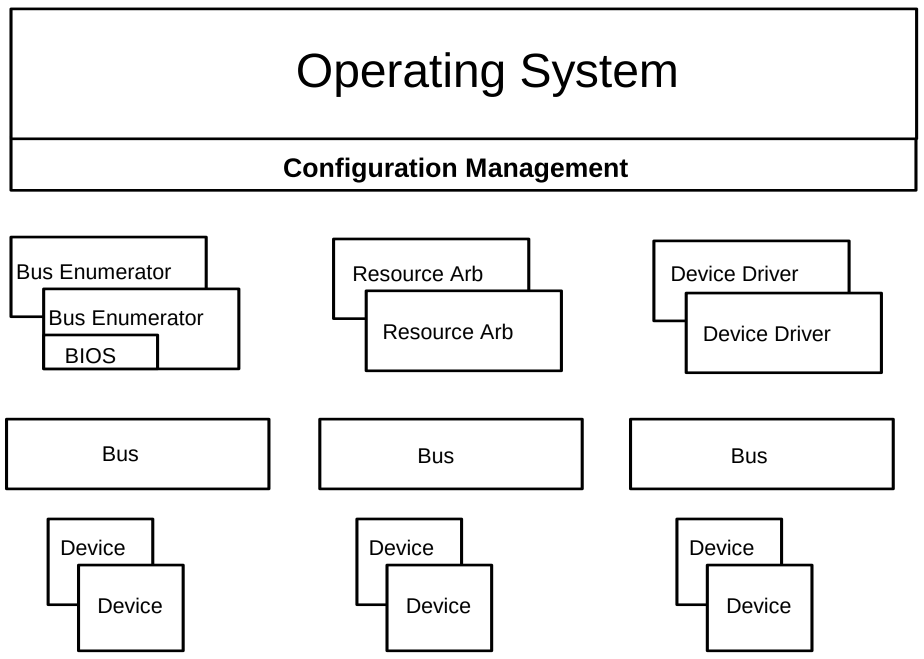

Figure 2: Diagram of Plug and Play Components

Plug and Play BIOS

To meet the requirements of Plug and Play, the system BIOS must be enhanced to provide boot device configuration and dynamic event notification services. In addition, these capabilities must be tightly integrated with the operating system.

A Plug and Play BIOS must be able to configure the system board devices (at a minimum) before handing control of the configuration process to the operating system. This process involves isolating and initializing the system board devices (Programmable Interrupt Controller, DMA Controller, System Video Controller, Floppy Controller, etc.). In the Plug and Play framework, each of these devices is associated with a unique identification code that is recognized by the operating system. The BIOS also maintains a list of system board device configuration information and communicates that information to the operating system after the POST process is complete.

To provide complete Plug and Play functionality, the BIOS must be able to notify the operating system of dynamic configuration events, such as the insertion of a notebook system into a docking station. A Plug and Play BIOS provides a mechanism for a Plug and Play operating system to reconfigure system board devices in response to a dynamic event. This enables the operating system to reconfigure the system without requiring the user to turn the system off, and also to notify applications and drivers of the new system configuration. In addition, for software-controlled devices (such as VCR-style docking systems), the BIOS can provide early warning to the operating system and prevent errors and data loss due to removal of the device.

Plug and Play Operating System

The operating system requires several new components to provide complete Plug and Play functionality, including:

- Software to control the configuration process and communicate with all components involved in that process (the **Configuration Manager**)
- A database of information that is used to configure the system (the **Hardware Tree**)
- Drivers to identify all the devices on a particular bus and their resource requirements (**Bus enumerators**)
- Software to allocate resources among all devices (**Resource Arbitrators**)

In addition, a number of components must be modified, including the operating system **setup** program and **user interface**.

Configuration Manager

The Configuration Manager is the central figure during all phases of the configuration process, orchestrating the entire flow of operations performed by all the components involved in configuration.

The Configuration Manager takes control of the configuration process when it receives the system board device configuration list from the BIOS, and also when the BIOS sends notification of a dynamic configuration event. The Configuration Manager then coordinates communication between the bus enumerators, hardware tree, device drivers and resource arbitrators to establish a working configuration for the system. In addition, the Configuration Manager notifies device drivers and applications of any pending or present changes in the system layout (i.e., new or removed devices).

To perform this role the Configuration Manager calls on the bus enumerators to identify all the devices on their specific buses and their respective resource requirements, then stores this information in the hardware tree as a hierarchical arrangement of device nodes. For each device a driver is loaded, and instructed by the Configuration Manager to await assignment of specific resources. The Configuration Manager calls on the resource arbitrators to allocate resources for each device, and in the event of a conflict performs an interactive process of reconfiguration until a working configuration is determined. Device drivers are then informed of the device-configuration information to complete the configuration process. The above process is reinitiated when the BIOS or one of the other bus enumerators informs the Configuration Manager of an event that requires a change to the system configuration, such as the removal or insertion of a PCMCIA card.

Hardware Tree

The hardware tree is a record in RAM of the current system configuration. The tree information is drawn from a central database of configuration information for all devices, whether they are currently installed or not. This record is created every time the system boots or a run-time change occurs to the system configuration. The format for the hardware tree defines a standard scheme for identifying each device, its resource requirements and resource constraints, if any. There may also be interdependencies between specific resources, such as COM ports and IRQs, for example. The central database is accessible to applications and drivers to provide them with information about alternative configurations, software required to operate these devices, and user-defined settings. The existence of this database will eliminate the need for many of the device and application-specific *.ini files used today.

Bus Enumerators

Bus enumerators are responsible for building (“enumerating”) the hardware tree on a Plug and Play system. The bus enumerators are a new type of driver required for each specific bus type.

The driver is designed to understand the implementation details of a specific bus architecture so it can identify the devices on that bus, read their resource requirements, and configure them as instructed by the Configuration Manager.

These drivers may leverage existing drivers or BIOS services to access hardware. For example, the SCSI bus enumeration is performed by calling the SCSI driver, and the PCMCIA enumeration is performed by the Card Services and Socket Services drivers. They may also be implemented at the BIOS level for specific buses such as the system board.

The critical function of the bus enumerator is to assign a unique identification code to each device on its bus. In addition, bus enumerators are expected to retrieve the device configuration information either directly from the device (for example, PCMCIA card configuration tuples) or from the central database (for existing ISA cards). The only requirement for the identification code is that it be unique and consistent so that each time the system boots the ID for a particular device is the same. The Plug and Play framework uses existing identification codes for most buses. For example, Plug and Play ISA devices are identified using the EISA device ID scheme, while PCMCIA cards are identified via the Manufacturer ID tuple on the card.

Resource Arbitrator

The Resource Arbitrator allocates specific types of resource to devices, and resolves conflicts between devices that request identical resource assignments. To accomplish this function the Resource Arbitrator contains all the information about how a resource is structured, and algorithms for determining a feasible resource configuration given a set of device requirements and constraints. This functional separation of the Resource Arbitrator and the Configuration Manager provides for future extensibility of the operating system to address new types of resources.

The Resource Arbitrator interacts extensively with the Configuration Manager to perform the iterative process of assigning resources both at power-up and in response to dynamic configuration events. In the case of a run-time reconfiguration of the system, the Configuration Manager may require the resource arbitrator to release resources and reassign them to satisfy the needs of a new device.

Changes to Other Operating System Components

The central configuration database is created during the operation of the system setup program. During setup, the operating system calls various detection and enumeration modules, to perform an inventory of all devices on the system and record information about those devices in the configuration database.

Although under normal circumstances the system will not require intervention by the end user to perform configuration operations, there are some exceptions. For example, if the user is installing a non Plug and Play device and the system fails to detect that device, the user can force an installation by pushing a button in the control panel or dragging an installation icon into the system folder from a floppy disk. At times the system may be unable to generate a non-conflicting configuration for a device. In this case user-interface components are needed to explain to the user what is happening, and ask what the user wants to do about the problem (e.g., turn off a device to make room for the new device, disable the new device, reconfigure a non Plug and Play card to make room for the new device). And since the Plug and Play system provides a way to change the system configuration, the system needs to provide a specialized tool to access and edit this information, primarily for use by the advanced user.

Device Drivers

The Plug and Play architecture builds on existing device driver models to provide additional APIs required for Plug and Play device configuration. Device drivers must become dynamically loadable and unloadable, both to enable reconfiguration and make most efficient use of system memory. Device drivers must communicate with other components of the system in a variety of ways: they must register with Configuration Manager when they are first loaded, remain inactive until they are given their resource assignments, and be able to communicate with applications to respond to dynamic configuration events. For example, a word processing application may want to save files before removal of a SCSI disk, or block the removal altogether. A PCMCIA socket enumerator that recognizes the insertion of a new card will want to notify the Configuration Manager of the insertion of the newly inserted card so that the appropriate drivers can be found and loaded, and resources assigned to the card.

Plug and Play Devices

ISA

A specification for design of Plug and Play ISA adapters has been published, reviewed and released to the industry. This specification defines a hardware and software mechanism for incorporation in the next generation of ISA cards, referred to as Plug and Play ISA cards. The software mechanism enables resolution of conflicts between Plug and Play ISA cards. In other words, the Plug and Play software optimally allocates system resources between the Plug and

Play ISA cards and other devices in the system without requiring user intervention. The Plug and Play ISA specification does not require any changes to the ISA buses, so Plug and Play ISA devices can run on existing PCs.

In a system that uses only Plug and Play ISA cards, it will be possible to achieve fully automatic configuration. However, during a transition period the current generation of standard ISA cards will coexist with Plug and Play ISA cards in the same system. In such systems, the ISA configuration solution needs to be supplemented with additions to the BIOS and/or operating system to manage and arbitrate ISA bus resources. User intervention may still be necessary in some cases.

The technical requirements of Plug and Play functionality on ISA systems include:

- A mechanism to uniquely address individual cards even when two or more cards request the same system resources
- A protocol for reading a configuration template on each card, to identify current resource usage and options for each system resource requested
- A mechanism to set or modify the configuration of each card

The Plug and Play ISA specification defines the mechanisms that each Plug and Play ISA card must implement to support identification, resource usage determination, conflict detection and conflict resolution. In addition, the specification presents a process for Plug and Play software to automatically configure new Plug and Play cards without user intervention. The major steps of the auto-configuration process are as follows:

- Put all Plug and Play ISA cards in configuration mode
- Isolate one Plug and Play ISA card at a time and assign a “handle” to the card
- Read the card’s resource data structure to determine the card’s resource requirements and capabilities
- Allocate conflict-free resources to each card
- Activate all Plug and Play ISA cards and remove them from configuration mode

The Plug and Play software identifies and configures devices using a set of commands that are executed using three 8–bit I/O ports. A sequence of data writes to one of the ports (the initiation key) is used to enable the Plug and Play logic on the card.

Once the cards are put in configuration mode, the Plug and Play software needs an isolation mechanism to address one particular card at a time, because all the cards respond to the same I/O port addresses. The isolation protocol uses a unique number on each card (the EISA device ID and a serial number) to isolate one Plug and Play card at a time. After isolation, the Plug and Play software assigns each card a handle, which is used to select that specific Plug and Play card. The handle eliminates the need for the more elaborate and time-consuming isolation protocol to select a specific card.

Each Plug and Play ISA card supports a read-only resource data structure that describes the resources supported and those requested by the functions on that card. The structure supports the concept of multiple functions per ISA card. Each function is defined as a logical device.
Plug and Play resource information is provided for each logical device and each logical device is independently configured through the Plug and Play standard registers. Following isolation, the Plug and Play software reads the resource data structure on each card. When all resource capabilities and demands are known, a process of resource arbitration is invoked to determine resource allocation to each ISA card.

The configuration of ISA cards is performed using the command registers specified for each resource type. Some ISA functions may not be reconfigurable. In this case, the resources requested will be equivalent to the resources assigned. However, the resource data structure informs the configuration software that it cannot assign these resources to other Plug and Play cards in the system. After the assignment of resources, an I/O conflict-detection mechanism may be invoked. This mechanism provides a means to ensure that I/O resources assigned are not in conflict with standard ISA cards. The command set also supports activating or deactivating the functions on the card.

Once configuration is complete, Plug and Play cards are removed from configuration mode. To re-enable configuration mode, the initiation key needs to be re-issued. This prevents accidental erasure of the configuration information.

PCMCIA

In order for PCMCIA devices to be supported under the Plug and Play framework, changes must be made to device drivers and PCMCIA system software. In addition, PCMCIA cards must include the manufacturer identification and configuration tuples for proper identification and configuration.

The approach to supporting PCMCIA under the Plug and Play framework calls for a PCMCIA adapter to be controlled by a PCMCIA bus enumerator that incorporates the PCMCIA Card Services functions as well as the enumeration and configuration functions described in the Plug and Play specification. Using this approach the PCMCIA device can be treated like any other Plug and Play device during the configuration process, so resources can be allocated and reclaimed dynamically by the operating system, allowing for easy addition of static devices and support for hot or warm docking.

To support PCMCIA cards as standard Plug and Play devices, the PCMCIA Card must be able to supply a unique identifier for itself and declare its resource requirements. This information can then be kept in the operating system hardware tree, where it can be used to configure the device dynamically.

PC Card device drivers can be implemented in several ways. If the device driver does not require the use of the Card Services API, the card vendor can implement a standard protect-mode Plug and Play driver. Resource allocation for this driver will then be performed by the configuration management software in the operating system, and configuration of the card itself will be handled by the PCMCIA bus enumerator after the system has allocated resources. A benefit of this approach is that it enables card vendors to write a single driver for the ISA/EISA and PCMCIA implementations of a specific I/O card. If the device is supported natively in the operating system, then the generic driver in the operating system for that device can be used. Specialized functions of the device can be implemented by directly accessing the card using I/O commands.

If the device driver requires Card Services memory functions, then the driver can be implemented as a “mixed” Plug and Play and Card Services driver. The structure of the driver would be the same, but it would register as a Card Services client at initialization time, and then call Card Services memory functions as needed.

In a Plug and Play system the PCMCIA system software is integrated with the operating system configuration management software, so that the Configuration Manager provides the resource allocation for the card rather than the Card Services module. This can be done by adding a PCMCIA bus enumerator and driver to Card Services, and changing Card Services to accept configurations from the configuration manager instead of tracking the resources itself. This implementation will provide support for existing Card Services clients as well as new Plug and Play drivers.

SCSI

SCSI devices require both design changes and general ease-of-use enhancements to deliver Plug and Play functionality. The configuration of a SCSI system can be separated into two distinct processes — configuring the SCSI bus itself, such as terminating both ends of the SCSI bus and setting device IDs, and configuring the SCSI host adapter, such as assigning an IRQ channel, DMA channel, etc.

Configuring the SCSI bus is difficult for a consumer. The concepts are mystifying to anyone who is unfamiliar with SCSI. In a Plug and Play world, the user would be able to just plug in the new device and the bus would adjust. But in today’s world the list of issues the user must confront to configure a SCSI bus is long, including SCSI device ID assignment, termination, SCSI parity, command sets, disk geometry and software. The SCSI-2 specification does not define an automated ID assignment mechanism, so the end user is responsible for making sure that no two SCSI devices on the same SCSI bus share the same SCSI ID. It is not easy for the non-technical user to tell whether or not a SCSI device is terminated, because termination can be done by jumpers, switches or by adding/removing resistor packs. Today, users can replace a SCSI host adapter with one from a different company and find it doesn’t work due to differences in disk geometries or the way devices are mapped to INT 13h parameters. Each of these issues needs to be addressed to provide Plug and Play functionality for SCSI buses.

ISA SCSI host adapters wishing to provide Plug and Play functionality must support the Plug and Play ISA Specification. The specification enables hardware to be built to obtain resources such as IRQ lines and DMA Channels automatically, so that users need not worry about assigning system resources and ensuring the assignment is coordinated with the rest of their system.

PCI

The current PCI bus architecture meets most of the requirements for providing Plug and Play functionality. PCI devices use a standard identification scheme and a mechanism for declaring their resource requirements. The BIOS on PCI platforms contains the configuration logic required to configure all PCI devices during POST, and to enable only boot devices and devices with option ROMs. In addition, the BIOS already provides a mechanism for device drivers to access PCI configuration data.

To become fully Plug and Play capable, a bus enumerator for PCI and some PCI BIOS extensions to the Plug and Play BIOS specification are required. The PCI bus enumerator will build the hardware tree in the operating system for both PCI devices and PCI bridges. The BIOS extensions are required to enable the operating system to correctly configure systems that contain both PCI and ISA devices. Since PCI devices can share interrupts but ISA devices may not, the BIOS must be able to prevent the operating system from mapping PCI devices to interrupts being used by non-shareable ISA devices.

EISA and Micro Channel

EISA and Micro Channel devices already provide a standard identification mechanism and a mechanism for configuring their resources through software. To integrate with the Plug and Play architecture, a bus enumerator for each bus standard is required to make configuration information about EISA and Micro Channel devices accessible to the operating system. The bus enumerators need to incorporate the functions previously provided by the configuration utilities for those buses. In addition, it is desirable to integrate the currently required configuration utilities with the operating system.

Other Device Types (Serial, Parallel, VL, IDE, Infrared)

Additional types of devices can be incorporated into the Plug and Play architecture as long as they provide the necessary mechanisms for identifying and configuring the device. VL bus devices can leverage much of the work that has been done with the ISA specification. IDE controllers already provide a way to support multiple disk drives, so with the addition of an identification scheme and mechanism for declaring resource requirements, the IDE standard could provide a low-cost solution for adding Plug and Play CD-ROM drives. The ECP specification can be extended to support Plug and Play devices that utilize the PC’s parallel port, and with the development of a standard for serial devices, the serial port could be used to add Plug and Play devices to the system as well. In the case of evolving technologies such as infrared communications, the

Plug and Play architecture provides a way to increase the usability and attractiveness of the technology.

Levels of Plug and Play Implementation and Corresponding Benefits

To provide complete Plug and Play functionality, a system must incorporate all three Plug and Play system components — BIOS (motherboard), devices (buses) and operating system. However, partial Plug and Play functionality can be provided on systems that include one or two of those components. So systems vendors can begin to provide customers some of the benefits of Plug and Play prior to the release of operating systems that include Plug and Play capability; and Plug and Play operating systems can provide benefits to users of existing systems.

There are some benefits in purchasing Plug and Play devices and installing them on current systems that do not have either a Plug and Play BIOS or Plug and Play operating system. If the customer installs a configuration utility that is separate from the operating system, and then installs a Plug and Play device with a new driver, the system will be able to automatically configure the new device. However, if the customer later installs a non Plug and Play device on that system, there may be configuration problems.

If the customer has a system with a Plug and Play BIOS and devices, but lacks a Plug and Play operating system, the system will automatically configure boot devices and system board devices so the customer can be certain that the system will boot. A configuration utility can handle new Plug and Play devices that are added. However, configuration problems may occur if the customer adds a non Plug and Play device to that system, because the BIOS will not be able to configure that device.

When a system includes a Plug and Play operating system as well as Plug and Play BIOS and devices, the customer receives the full benefits of Plug and Play. Both Plug and Play and non Plug and Play devices can be configured automatically, because the operating system will be able to read the non Plug and Play card’s resource requirements more effectively using an improved configuration information file format, and assign resources first to the non Plug and Play cards before allocating the remaining resources to the more flexible Plug and Play cards. In the event that a resource conflict arises between multiple non Plug and Play cards, the operating system can provide an improved interface to the user for making manual modifications to the system configuration.

The major incremental benefit of having a system with complete Plug and Play components is the ability of that system to respond to dynamic configuration events. Excellent docking system solutions will finally be feasible, because the system will be able to automatically load and unload device drivers to reflect the different devices attached to the system when it is docked or undocked. Also, applications will be able to automatically adjust their configurations to reflect the insertion or removal of devices, such as a network card and a fax-modem.

Conclusion

The Plug and Play framework advances the PC architecture in ways that will make it possible to both improve the experience that customers have working with existing PCs, and enable them to work with PCs in new ways. The results will not only be more satisfied customers, but also a healthier industry with lower support costs and higher demand for PC products. In recognition of the benefits of Plug and Play for the entire PC industry, a variety of companies representing all major aspects of the industry are working together closely to define Plug and Play architecture and to implement that architecture in their products. In the near future, products that deliver some of the benefits of Plug and Play will begin to appear in the market, with fully Plug and Play products appearing as all of the components are completed.

Sources for More Information

A number of specifications addressing the Plug and Play framework are in various stages of completion, and more are being proposed. Copies of the following specifications and others, as they are completed, can be requested by sending e-mail to plugplay@microsoft.com.

- Plug and Play ISA Specification (Intel Corporation and Microsoft Corporation)
- Plug and Play Device Driver Interface Specification for Microsoft Windows 3.1 and MS-DOS (Microsoft Corporation)
- Plug and Play BIOS Specification (Compaq Computer Corporation and Phoenix Technologies, Ltd.)
- Plug and Play PCMCIA Specification (Intel Corporation and Microsoft Corporation)
- Plug and Play PCI Specification (Intel Corporation)

Founded in 1975, Microsoft (NASDAQ “MSFT”) is the worldwide leader in software for personal computers. The company offers a wide range of products and services for business and personal use, each designed with the mission of making it easier and more enjoyable for people to take advantage of the full power of personal computing every day.

#########