



Implementation of Hot Swap Solution for Windows 2000

White Paper

Introduction

The Plug-and-Play technology becomes more and more popular in the modern computer systems. Adding and removing system peripherals without shutdown, which is also known as ‘hot plug’ or ‘hot swap’, is the advanced version of plug-and-play functionality. For the proper implementation of hot plug technology in computer systems, it should be supported on both the level of I/O interface (hardware device level) and on the OS I/O level (software level).

Plug and play fits perfectly well PCI bus technology, one of the modern widespread I/O interfaces, because of its many architectural advantages, such as availability of configuration space. However, neither specification of the PCI bus nor its basic realizations support ‘hot’ adding/removal of peripherals.

The CompactPCI bus is a version of the PCI bus for industrial systems, developed by consortium of manufacturers of the industrial computers (PCI Industrial Computers Manufacturers’ Group—PICMG). PCI and CompactPCI are electrically compatible, but different in mechanical parameters: type of the socket, the form factor of the card (Eurocard in CompactPCI), and presence of the latches on the card. Moreover, ‘hot’ insertion and extraction of peripherals has been standardized (and named ‘Hot Swap’) in the specification [1], approved by PICMG.

The specification includes hardware and software requirements for Hot Swap support, both on the board level and on the bus level. In addition, it describes the procedure, which the operator should use for the insertion and extraction of peripheral boards.

The hardware requirements include the following:

- The different length of the contact pins on the backplane socket, which provides power supply to the device (“early power”) before any other signals.
- Presence of the blue LED, which signals that the device is ready for extraction.
- The ENUM# signal line on the bus, which is activated according to the specific protocol when the bottom latch on the peripheral boards is opened or closed.

Software requirements include the special Hot Swap control and status register (HS_CSR), situated in the configuration space of the peripheral device in the Extended Capability List. This register consists of the following 1-bit flags:

- INS—is set by hardware when the bottom latch on the board is closed after the board has been inserted into the system; can be reset programmatically;
- EXT—is set by hardware when the bottom latch on the board is opened before extracting the board from the system; can be reset programmatically;

- LOO—is set programmatically and controls the state of the blue LED on the device;
- EIM—is set and reset programmatically; when set it blocks the activation of the ENUM# signal from the device.

To insert a device in hot mode, the operator should plug the board into the CompactPCI slot, close the bottom latch and wait for the blue LED to be turned off. The LED is lit initially after plugging the device into the slot and is turned off after hardware initialization of the device completes normally. If the LED stays permanently on after the insertion, then it means that a hardware error occurred during the initialization of the device; this device should be extracted from the system.

To extract a device in hot mode, the operator should open the bottom latch and wait for the blue LED to be turned on before physically extracting the board. When the device becomes ready for extraction from the OS perspective (for example, after the device driver is unloaded) the blue LED is turned on programmatically. The operation system may not allow to extract the device; in this case the LED will not be turned on and the operator can cancel the extraction request by closing the latch.

Currently Hot Swap specification is intensively evolving, especially in its software-related aspects. It is expected that several new fields in the HS_CSR register will be defined in the new version of the specification. These fields will contain additional information about the current status of the device.

Moreover, the specification PICMG 2.12 [2], which describes the software interfaces of the Hot Swap infrastructure, has been adopted recently. This specification deals with handling of the ENUM# signal, alternate implementations of the HS_CSR register and with the software interfaces for the slot control functionality (which is specified for the High Reliability level of Hot Swap support).

Functionality of Hot Swap Software Solutions

At the moment an adequate implementation of Hot Swap for Compact PCI is the package Hot Swap Kit (HSK) of Pigeon Point Systems, Inc. The Hot Swap Kit (HSK) has been jointly developed by Pigeon Point Systems (USA) and Auriga, Inc. (USA). It performs software support of Hot Swap for CompactPCI in the Windows 2000 environment [3]. The package is embedded into the kernel of the OS [3], advancing its functionality in the area of Plug-And-Play with respect to the CompactPCI Hot Swap.

The basic functions of the package are as follows:

- Verifying presence of devices on the bus and their current Hot Swap status;
- Handling the alternate implementations of the HS_CSR register;
- Notifying the system about changes in the device population on the CompactPCI bus;

- Processing the device extraction requests;
- Filtering the Plug-and-Play input-output request packets (IRPs) for CompactPCI devices;
- Assigning resources to the PCI-PCI bridges on the CompactPCI bus;
- Supplying an application program interface (API)_ for obtaining information about the CompactPCI bus status and for controlling the Hot Swap operation.

One of the main features of Hot Swap implementation in Windows 2000 done by HSK is the compatibility on the level of the functional device drivers. Because of the input/output model in Windows 2000, based on Plug-And-Play (which supports dynamic insertion/extraction of devices), any correctly developed functional device driver automatically supports Hot Swap; HSK provides calls to appropriate functions and sending appropriate IRPs to the functional drivers when Hot Swap events occur.

Let us review the package functions in more details:

Verifying the presence of devices on the bus and their current status

In order to trace the current population of devices on the CompactPCI bus and their status from the Hot Swap perspective the HSK software cyclically polls the configuration space of all devices on the bus and reads their HS_CSR register. The arrival and departure of the proper configuration header at the specific bus location indicates ‘hot’ device insertion or extraction at this location. The EXT flag being set indicates an extraction request for the device, initiated by the operator. The INS flag being set means ‘hot’ device insertion or cancellation of the previous extraction request (closing of the bottom latch), also initiated by the operator. When discovering one of these flags, HSK immediately clears it in order to stop generation of the ENUM# signal from that device and allow recognition of the subsequent device status changes (according to the standard, INS and EXT flags cannot be set simultaneously).

Handling the ENUM# Signal

The ENUM# signal is activated by a CompactPCI device when its Hot Swap state changes (device has been inserted, or device extraction has been requested, or the extraction request has been cancelled). The signal does not indicate which device has its status changed, because outputs from all devices are joined together on the bus in the form of the common signal. The implementation of the signal on the platform level is left to the platform manufacturer. As a rule, the ENUM# signal maps on the certain interruptvector or on a bit flag, which can be polled, or on both of these mechanisms. In addition, the interrupt can be generated by front (Latched) or by level of the signal (levelSensitive).

When the ENUM# signal is detected, the HSK software starts the cycle of polling the CompactPCI bus to find the source of the signal and to perform actions relevant to the status change. In addition, if the platform implements the ENUM# signal as a level-sensitive interrupt, this interrupt is immediately disabled to prevent the system from hanging indefinitely. The interrupt is enabled after the completion of the bus polling cycle.

One example of the ENUM# signal implementation can be found in the design of Motorola Pentium-based single-board computers CPV5300, CPV5350, CPV5370. On these platforms, both interrupt-based and flag-based mechanisms are supported. The ENUM# interrupt is latched and the interrupt vector can be chosen among several values. Besides, the current state of the ENUM# signal is represented as a bit in one of the FPGA registers. The interrupt vector is programmed via another FPGA register. Access to FPGA registers on this platform is done via a block of I/O ports; the base address of this block (usually 0x50) is configurable via the PCI-ISA bridge device in the system chipset (so called SouthBridge).

HSK includes the platform driver for Motorola platforms. It supports both interrupt-driven and polled methods of ENUM# detection. Also, the driver configures the ENUM# interrupt vector during initialization, choosing the first available one from the list of supported vectors and programming it into the FPGA register.

HSK includes the special platform-dependent component (platform driver) for handling the ENUM# signal; this component is responsible for detecting the signal and immediate disabling of the level-sensitive interrupt (if any). Subsequent handling is performed by this component and the platform-independent part of the HSK together. They interact according to the protocol, standardized in [2].

However, the package is able to function on a hardware platform, even if the platform doesn't support the ENUM# signal or no appropriate platform driver has been installed. In this case HSK performs periodical polling of the CompactPCI bus; the time interval between subsequent polls is configurable.

Handling alternate HS_CSR implementations

The Hot Swap specification allows but does not encourage non-standard (alternate) implementations of the HS-CSR register for CompactPCI devices. Alternate implementations can use different mechanisms to access the register bits but should preserve their semantics. Implementations that use the standard mechanism to access the bits but deviate from the standard semantically, are also considered as alternate implementations.

In that case, the special software component should be developed for that device. This component is called an alternate HS_CSR driver and is responsible for emulation of the standard HS-CSR register for this device. The alternate HS_CSR driver interacts with other parts of the HSK package according to the protocol standardized in [2]. Currently, there exist a limited number of devices with alternate HS-CSR implementations; it is expected that all new devices will implement the HS_CSR register according to the standard and the alternate implementations will gradually disappear.

An example of alternate HS_CSR implementation is the Motorola CPV8540 PMC carrier board, based on the transparent PCI-PCI bridge Intel 21154. The board designers mapped the four defined HS_CSR bits onto the four general purpose I/O (GPIO) lines of Intel 21154. Access to the GPIO lines is performed via three different registers in the device-private part of the bridge configuration space (with offsets 0x65, 0x66, 0x67). The first register is used to read the GPIO bits; the second one is used to set the GPIO bits; the third register is used to clear them. HSK includes the alternate HS_CSR driver for CPV8540; this driver provides transparent emulation of the standard HS_CSR for the Hot Swap Driver on the CPV8540 boards.

Notifying the system about the change of device population on the CompactPCI bus

If the change in the device population on the CompactPCI bus (or on several buses simultaneously) occurs (a new device appears or an existing device disappears), HSK software notifies the operating system about it by calling the system function `IoInvalidateDeviceRelations`, which requests the system to re-enumerate the CompactPCI bus, detect the changes from the previous state and handle them appropriately.

Actually, in accordance with the Window 2000 architecture the HSK does not directly inform the system which devices have appeared and which devices have disappeared, but only initiates the re-enumeration of the bus. The PCI bus driver re-enumerates the bus and provides the final information. However, in order to minimize the amount of work to the system, HSK requests the re-enumeration for the PCI-PCI bridge nearest to the location where the changes have occurred.

Processing the device extraction request

The request for device extraction occurs when the operator opens the bottom latch on the board. The HSK software processes the request for device extraction, calling the system function `IoRequestDeviceEject`. This function sends the request for device extraction to the appropriate device stack. The driver can agree for extraction or refuse; other drivers or applications can also reject the device extraction. If nobody rejects the extraction request, the system sends the `IRP_MN_REMOVE_DEVICE` IRP to the device stack, which means that the device has been logically removed, frees the device resources and disables the device on the bus. At this moment HSK sets the LOO flag in the HS_CSR register of the device, illuminating the blue LED and signaling to the operator that the device can be safely removed.

The request for device extraction can be executed programmatically via the HSK API or by calling the system function `CM_Request_Device_Eject`.

Filtering IRPs for devices on the CompactPCI bus

For CompactPCI devices, HSK creates device objects and inserts them as filter device objects between physical device objects (managed by the PCI bus driver) and functional device objects. Thus, HSK performs filtering of input/output request packets (IRPs) for CompactPCI devices. Filtering provides the following possibilities:

1. **Delaying of the IRP_MN_DEVICE_EJECT IRP.** OS sends this request for physical extraction of the device and assumes that a device should be absent (physically removed) on the completion of the request. However, an indefinitely long time can pass between the logical device removal (illumination of the blue LED) and the physical device extraction. Moreover, the operator can cancel the pending extraction, closing the bottom latch. The filter driver delays completion of this request (returning STATUS_PENDING) on its way back upwards from the PCI bus driver. While the request is in the pending status, the device is considered as waiting for extraction and is not served by the system. The request is completed by the filter driver either when the device disappears from the bus (successful completion) or when the extraction request is cancelled by the operator (unsuccessful completion).
2. **Propagation of the filter driver across all the CompactPCI devices.** During the installation, the filter driver associates itself with the root PCI bus. During the process of discovering PCI devices by the system the filter driver attaches itself to all newly discovered devices, including PCI-PCI bridges, thus propagating itself down the PCI tree. This is accomplished by filtering the IRP_MN_QUERY_DEVICE_RELATIONS IRP.
3. **Tracing the device Plug-and-Play status.** The filter driver traces all Plug-and-Play related IRPs for the device and, thus, is aware of status changes (such as logical device removal, which is indicated by the arrival of the IRP_MN_REMOVE_DEVICE IRP).
4. **Declaring CompactPCI-devices as dynamically removable and Ejectable.** In order for Windows 2000 to be able to process an extraction request for a device, the device should be explicitly described as Removable and Ejectable in its capabilities list. This is achieved by the filter drive by filtering the IRP_MN_QUERY_CAPABILITIES IRP.

Assigning Resources to the PCI-PCI bridges

As a rule, the CompactPCI bus connects to the parent PCI bus through a parent PCI-PCI bridge. The resources, assigned to CompactPCI devices (I/O ports, memory, bus numbers) should be inside the range of the corresponding resources, assigned to the parent bridge.

In the process of the system initialization and before OS loading BIOS performs initial assignment of resources to PCI devices in x86-compatible systems. Usually BIOS assigns to the bridges the minimal necessary resource windows so that the existing devices behind the bridge are able to work.. This approach creates problems for 'hot' insertion of devices on the CompactPCI bus behind the bridge, because the bridge resources may be insufficient to accommodate the new device. Dynamic reallocation of resources may also be impossible without touching other devices on the bus (and, possibly on the other PCI buses). This problem is even worse if the inserted device is a bridge, since in that case we also need a free PCI bus number in addition to I/O port and memory resources.

To solve this problem, the HSK allows the system operator to explicitly specify the resources to be assigned to certain PCI-PCI bridges in the system. This functionality is mainly intended for the parent CompactPCI bridges, but could be applied to any other PCI-PCI bridges. The resource assignment instructions are bound to the bridge address on the bus and are stored in the system registry as a character string. The instructions may specify only the size of the resource window or may explicitly the starting address of the resource window.

When Windows 2000 is loaded, HSK checks the presence of this information for the next bridge device and, if it exists, overrides the resources assignments performed by the BIOS, and orders OS to perform new assignment of resources in accordance with saved requirements.

Application programs interface (API)

The HSK software provides interface for the application programs. This interface allows an application to obtain information about the current status of the CompactPCI slots and devices in them, programmatically initiate and cancel requests for device extraction, and also receive notifications about Hot Swap related events.

The interface for application programs is exported by the dynamic link library HSAPI.DLL and is represented as a collection of functions and data structures. To interact with the kernel-level components, the library uses the device I/O control requests (IOCTLs) and standard system Plug-And-Play device notifications.

Architecture of the HSK package

The HSK package operates mostly in Windows 2000 kernel mode and consists of the following set of components, all represented as device drivers.

1. **The filter driver for the CompactPCI bus.** This driver is installed as a filter on the top of the standard PCI bus driver (which serves the CompactPCI bus also). It provides filtering of the input-output request packages (IRPs) for the devices on the bus, informs the system about advanced features of these devices (support for dynamic removal and ejection), traces the current status of devices from the perspective of the system and assigns resources specified by the system integrator, to the PCI-PCI bridges.
2. **The Hot Swap System Driver.** This driver performs tracing the population of the CompactPCI bus and status of devices on it, by polling the bus periodically or in response to the ENUM# signal. In every cycle of polling the driver scans configuration space of devices on the bus and determines arrival or departure of devices, and requests for device extraction as well. This driver also provides the interface to application programs, based on the device I/O control (IOCTL) functions and on the standard system mechanism of Plug-And-Play notifications.

3. **The platform driver (ENUM# driver).** This driver is responsible for recognition of the ENUM# signal, the implementation of which is left to the platform manufacturer according to specification. A driver is created for every supported platform, it interacts with the Hot Swap system driver according to the protocol, which is standardized in the PICMG 2.12 specification. The package can work without platform driver, in this case the bus polling is done periodically by the timer.
4. **The alternate HS-CSR drivers.** The alternate HS-CSR driver is responsible for the emulation of the standard HS_CSR register on a specific peripheral device with an alternate (non-standard) implementation of the HS-CSR. The alternate HS-CSR driver interacts with Hot Swap system driver according to the protocol, which is standardized in the PICMG 2.12 specification.

In addition, the package includes configuration program, which is intended for the visual editing of the HSK parameters, resources for assigning to the PCI-PCI bridges and describing non-standard platforms. This program runs in the user regime; it renews the contents of the system register and signals to HSK software about changes of the parameters by means of API interface.

Conclusion

Currently the package is licensed by several companies, which manufacture devices and systems based on CompactPCI (Motorola Computer Group, Ziatech Corporation, Dialog Inc.) for distribution with systems in OEM-versions of Windows 2000. Moreover, the package is licensed by Microsoft, Inc. for inclusion into the next version of Windows NT Embedded, based on the Windows 2000 kernel.

References

1. PICMG Hot Swap Specification. PICMG 2.1 R1.0.
5. PICMG Hot Swap Infrastructure Interface Specification. PICMG 2.12 R 1.0.