166166

31 OCT 2003

METHOD FOR RECOVERING FROM A MACHINE CHECK INTERRUPT DURING RUNTIME

BACKGROUND OF THE INVENTION

1. Technical Field:

The present invention relates generally to the field of computer architecture and, more specifically, to methods and systems for managing machine check interrupts during runtime.

2. Description of Related Art:

A logical partitioning option (LPAR) within a data processing system (platform) allows multiple copies of a single operating system (OS) or multiple heterogeneous operating systems to be simultaneously run on a single data processing system platform. A partition, within which an operating system image runs, is assigned a non-overlapping sub-set of the platform's resources. These platform allocable resources include one or more architecturally distinct processors with their interrupt management area, regions of system memory, and input/output (I/O) adapter bus slots. The partition's resources are represented by its own open firmware device tree to the OS image.

Each distinct OS or image of an OS running within the platform is protected from each other such that software errors on one logical partition can not affect the correct operation of any of the other partitions. This is provided by allocating a disjoint set of platform resources to be directly managed by each OS image and by

G00002

31 OCT 2003

providing mechanisms for ensuring that the various images can not control any resources that have not been allocated to it. Furthermore, software errors in the control of an OS's allocated resources are prevented from affecting the resources of any other image. Thus, each image of the OS (or each different OS) directly controls a distinct set of allocable resources within the platform.

Currently, in both LPAR systems and non-partitioned systems, when a machine check occurs due to a bad I/O adapter in the system, data about the condition causing the machine check is presented to the operating system in the form of an error log entry. The operating system then performs a complete shutdown of the system. The user must then replace the bad I/O adapter and then reboot the system. Such a requirement may not be terribly problematic for users with a simple configuration in which a reboot is relatively quick or for users in which having the system available at all times is not critical. However, for other users with complex configurations, such as, for example, multiple racks of serial storage architecture (SSA) or networked systems, a considerable amount of time will be spent rebooting the system just to replace one bad I/O adapter. Such expenditure of time may be very costly for those users. For example, if the system is a web server critical for taking internet sales orders for products, such as, for example, books or compact disks (CDs), each minute of time that the system is shut down to replace a bad I/O adapter may result in many thousands of dollars in lost sales.

Therefore, a method and system for replacing bad I/O

adapters without the need for powering down or rebooting the system would be desirable.

SUMMARY OF THE INVENTION

According to a first aspect of the invention there is provided a method of managing a failed input/output adapter within a data processing system, the method comprising:

coupling a plurality of hardware devices to an I/O bus utilizing a bridge such that the plurality of hardware devices communicate with the I/O bus utilizing the bridge;

receiving an indication that a hardware device within the plurality of hardware devices has failed;

attempting to identify the hardware device as a failed hardware device; and

responsive to successfully identifying the failed hardware device, disabling the failed hardware device without powering down the data processing system, wherein remaining ones of the plurality of hardware devices remain operational and continue to communicate with the I/O bus utilizing the bridge after the failed hardware device has been disabled.

According to a second aspect of the invention there is provided a computer program product in a computer readable medium for use in a data processing system for managing a failed hardware device within the data processing system, the computer program product comprising:

first instructions for receiving an indication that a hardware device within a plurality of hardware devices has failed, the plurality of hardware devices being coupled to an I/O bus utilizing a bridge;

second instructions for attempting to identify the hardware device as a failed hardware device; and

third instructions, responsive to successfully identifying the failed hardware device, for disabling the failed hardware device without powering down the data processing system, wherein remaining ones of the plurality of hardware devices remain operational and continue to communicate with the I/O bus utilizing the bridge after the failed hardware device has been disabled.

According to a further aspect of the invention there is provided a system for managing a failed hardware device within a data processing system in which a plurality of hardware devices communicate with an I/O bus via a bridge, the system comprising:

first means for receiving an indication that a hardware device within the plurality of hardware devices has failed;

second means for attempting to identify the hardware device as a failed hardware device; and

third means, responsive to successfully identifying the failed hardware device, for disabling the failed hardware device without powering down the data processing system, wherein remaining ones of the plurality of hardware devices remain operational and continue to communicate with the I/O bus via the bridge after the failed hardware device has been disabled.

According to a further aspect of the invention there is provided a method of replacing a failed input/output adapter in a data processing system in which a plurality of input/output adapters are coupled to an I/O bus via a bridge, the method comprising:

receiving an indication of a failed input/output adapter;

consulting an error log to determine an identity of the failed input/output adapter;

responsive to a lack of a determination of the identity of the failed input/output adapter, shutting down the data processing system; and

responsive to a determination of the identity of the failed input/output adapter:

refraining from shutting down the data processing system;

disabling the failed input/output adapter;

deallocating any processes bound to the failed input/output adapter; and

indicating to a user which one of a plurality of input/output adapters is the failed input/output adapter;

wherein the failed input/output adapter may be replaced without shutting down the system, remaining ones of the plurality of input/output adapters continuing to communicate with the I/O bus via the bridge after the failed input/output adapter has been disabled.

According to a further aspect of the invention there is provided a data processing system, comprising:

a bus system in which a plurality of hardware devices communicate with an I/O bus via a bridge;

a memory; and

a processor unit; wherein

the processor unit executes instructions to identify a failed hardware device within the plurality of hardware devices; and

responsive to successfully identifying the failed hardware device, disables the failed hardware device without powering down the data processing system, wherein remaining ones of the plurality of hardware devices remain operational and continue to communicate with the I/O bus via the bridge after the failed hardware device has been disabled.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figure 1 depicts a block diagram of a data processing system in accordance with the present invention;

Figure 2 depicts a block diagram of a system for handling machine check interrupts without the necessity of powering down the system is depicted in accordance with the present invention;

Figure 3 depicts an exemplary table depicting possible contents and information contained within an error log in accordance with the present invention;

Figure 4 depicts a flowchart illustrating an exemplary process of replacing a bad I/O adapter without powering down the system in accordance with the present invention;

Figure 5 depicts a flowchart illustrating an exemplary method of removing a bad I/O adapter via hotplug in accordance with the present invention; and

Figure 6 depicts an example menu allowing a user to initiate a hotplug procedure for removing and replacing a bad I/O adapter in accordance with the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

With reference now to the figures, and in particular with reference to **Figure 1**, a block diagram of a data processing system in which the present invention may be implemented is depicted. Data processing system **100** may be a symmetric multiprocessor (SMP) system including a plurality of processors **101**, **102**, **103**, and **104** connected to system bus **106**. For example, data processing system **100** may be an IBM RS/6000, a product of International Business Machines Corporation in Armonk, New York, implemented as a server within a network. Alternatively, a single processor system may be employed. Also connected to system bus **106** is memory controller/cache **108**, which provides an interface to a plurality of local memories **160-163**. I/O bus bridge **110** is connected to system bus **106** and provides an interface to I/O bus **112**. Memory controller/cache **108** and I/O bus bridge **110** may be integrated as depicted.

Data processing system **100** is a logically partitioned data processing system. Thus, data processing system **100** may have multiple heterogeneous operating systems (or multiple instances of a single operating system) running simultaneously. Each of theses multiple operating systems may have any number of software programs executing within in it. Data processing system **100** is logically partitioned such that different I/O adapters **120-121**, **128-129**, **136-137**, and **146-147** may be assigned to different logical partitions.

Thus, for example, suppose data processing system **100** is divided into three logical partitions, P1, P2, and P3. Each of I/O adapters **120-121**, **128-129**, and **136-137**, each of processors **101-104**, and each of local memories **160-164** is assigned to one of the three partitions. For example, processor **101**, memory **160**, and I/O adapters **120**, **128**, and **129** may be assigned to logical partition P1; processors **102-103**, memory **161**, and I/O adapters **121** and **137** may be assigned to partition P2; and processor **104**, memories **162-163**, and I/O adapters **136** and **146-147** may be assigned to logical partition P3.

Each operating system executing within data processing system **100** is assigned to a different logical partition. Thus, each operating system executing within data processing system **100** may access only those I/O units that are within its logical partition. Thus, for example, one instance of the Advanced Interactive Executive (AIX) operating system may be executing within partition P1, a second instance (image) of the AIX operating system may be executing within partition P2, and a Windows 2000™ operating system may be operating within logical partition P1. Windows 2000 is a product and trademark of Microsoft Corporation of Redmond, Washington.

Peripheral component interconnect (PCI) Host bridge **114** connected to I/O bus **112** provides an interface to primary PCI local bus **115**. A number of Input/Output adapters **120-121** may be connected to primary PCI bus **115** via respective secondary PCI buses **118-119** and external

address strobe (EADS) **116**. Typical PCI bus implementations will support between four and eight I/O adapters (i.e. expansion slots for add-in connectors). Each I/O Adapter **120-121** provides an interface between data processing system **100** and input/output devices such as, for example, other network computers, which are clients to data processing system **100**.

EADS **116**, **124**, **132**, and **142** are multifunction PCI-PCI bridges that support hot plugging of I/O adapters on the secondary buses **118-119**, **126-127**, **134-135**, and **144-145**. Hot plugging allows the addition, removal, and replacement of I/O adapters **120-121**, **128-129**, **136-137**, and **148-149** during runtime. Each adapter **120-121**, **128-129**, **136-137**, and **148-149** has its own secondary PCI bus **118-119**, **126-127**, **134-135**, and **144-145**, which makes hot plugging possible. Each EADS **116**, **124**, **132**, and **142** may be configured to function as a bridge from a primary bus **115**, **123**, **131**, and **141** to up to eight secondary hot plug PCI busses **118-119**, **126-127**, **134-135**, and **144-145**. In the depicted embodiment primary PCI buses **115**, **123**, **131**, and **141** are 64-bits wide, 3.3 volt (V) tolerant and run at between 16-66 MHz. Secondary PCI buses **118-119**, **126-127**, **134-135**, and **144-145** are 32-bits wide, wherein each pair of secondary PCI buses **118-119**, **126-127**, **134-135**, and **144-145** may be combined into a 64-bit wide bus.

Additional PCI host bridges **122** and **130** provide interfaces for additional primary PCI buses **123** and **131**. Each of additional primary PCI buses **123** and **131** are connected to a respective one of EADS **124** and **132**. Each

of EADS **124** and **132** is connected to a plurality of secondary PCI buses **126-127** and **134-135** as depicted in **Figure 1**. Secondary PCI buses **126-127** and **134-135** provide a connection between a plurality of PCI I/O adapters **128-129** and **136-137** to EADS **124** and **132**. Thus, additional I/O devices, such as, for example, modems or network adapters may be supported through each of PCI I/O adapters **128-129** and **136-137**. In this manner, data processing system **100** allows connections to multiple network computers.

A memory mapped graphics adapter **148** and hard disk **150** may also be connected to I/O bus **112** via EADS **142** and PCI Host Bridge **140**. EADS **142** is connected to PCI Host Bridge **140** through primary PCI bus **141**. Graphics adapter **148** is connected to EADS **142** through secondary PCI bus **144** and hard disk adapter **149**, through which hard disk **150** is connected to data processing system **100**, is connected to EADS **142** through secondary PCI bus **145**. Hard disk **150** may be logically partitioned between various partitions without the need for additional hard disks. However, additional hard disks may be utilized if desired.

Those of ordinary skill in the art will appreciate that the hardware depicted in **Figure 1** may vary. For example, other peripheral devices, such as optical disk drives and the like, also may be used in addition to or in place of the hardware depicted. Furthermore, the present invention is not limited to implementation on a multiprocessor logically partitioned system, but may also be implemented on other types of data processing systems,

such as, for example, a single processor system running a single image of an operating system, such as a typical personal computer. The depicted example is not meant to imply architectural limitations with respect to the present invention.

With reference now to **Figure 2**, a block diagram of a system for handling machine check interrupts without the necessity of powering down the system is depicted in accordance with the present invention. A machine check interrupt is an interrupt that operating systems, such as, for example, AIX, use when the data processing system detects that an adapter has failed. A catastrophic error that will always have AIX report a machine check is a system error (SERR). A SERR may be caused for various reasons such as, for example, parity errors. Parity errors may include data parity error during a special cycle and address parity errors. A SERR may also be cause by other critical error other than parity, such as, for example, a master abort error. In prior art systems, a machine check interrupt resulted in a sytsem shutdown. However, the present invention, provides a mechanism for handling the machine check error and replacing the bad I/O adapter without resort to shutting down the data processing system.

An operating system (OS) **202** executing within a data processing system, such as, for example, data processing system **100** in **Figure 1**, includes an OS interrupt handler **204** for handling machine check interrupts occurring within the data processing system. OS **202** may be, for example, AIX. A run-time abstraction service (RTAS) **208** provides a mechanism for receiving machine check

interrupts from PCI Host Bridges **210-212**. In the present example, RTAS **208** is implemented within firmware. Firmware is software stored in a memory chip that holds its content without electrical power, such as, for example, read-only memory (ROM), programmable ROM (PROM), erasable programmable ROM (EPROM), electrically erasable programmable ROM (EEPROM), and non-volatile random access memory (non-volatile RAM).

The machine check interrupt indicates that one of I/O adapters **214-220**, connected to PCI Host Bridges **210-212** is bad (i.e. has ceased to perform correctly). OS handler **204** makes calls to RTAS **208** to determine whether a machine check has occurred and RTAS **208** presents the machine check interrupt data to OS handler **204** in the form of an error log entry. An example of an error log entry is:

```
BFE4C025 0607120300 P H sysplanar0 MACHINE_CHECK_CHRP
```

Such an error log entry indicates to OS **202** that machine check interrupt has been received indicating that an I/O adapter **222-228** identified by the ID "BFE4C025" has failed. OS handler **204** than posts data to error log **206** based on the information received from RTAS **208**. Error log **300** depicted in **Figure 3** is an exemplary table, in accordance with the present invention, depicting possible contents and information contained within error log **206**.

The contents of error log **206** indicates which of I/O adapters **214-220** is bad, if RTAS **208** is able to determine such information. Otherwise, the entry in error log **206** merely reflects the fact that a machine check interrupt

has occurred within the data processing system without indicating which of I/O Adapters **214-220** caused the machine check interrupt.

Once OS handler **204** has written to error log **206**, OS **202** analyzes error log **206** to determine the identity of the failing I/O adapter **222-228**. For example, an AIX operating system may analyze error log **206** using the "DIAG" command and produce the following result:

```
A03-030: I/O bus time-out, access, or other error
 n/a        FRU:n/a                  U0.1-P1-I3
```

The data structure of the form "U0.X" indicates by "X" the drawer number of the bad adapter. Thus, in the example of above, "U0.1" (X=1), indicates that the drawer number of the bad adapter is "1." The data structure of the form "IY" indicates the slot number of the bad adapter where "Y" is the slot number. Thus, in the example above, the bad adapter is in slot 3, drawer 1.

If OS interrupt handler **204** is unable to determine the one of I/O adapters **214-220** that failed, then the data processing system is powered off. If, however, OS interrupt handler **204** is able to determine the one of I/O adapters **213-220** that failed, then operating system **202** will call RTAS **208** hot plug to disable the failing I/O adapter **214-220** and deallocate any processes bound to the failing one of I/O adapters **214-220**. RTAS writes directly to the EADS register of the one of EADS **214** and **218** corresponding to the bad one of I/O adapters **222-228** to turn off the power to the slot containing the bad one

of I/O adapters **222-228**.

Once the failing one of I/O adapters **214-220** is disabled, an urgent sysplanar message can be sent to a user notifying the user of the bad adapter, such as, for example, via a page, a message to a video display terminal, and/or blinking lights on the slot containing the bad adapter. An urgent sysplanar is a message that the system will output to an output device, typically at predefined time intervals, so that a user may be alerted the adapter is no longer working.

The user will then be able to replace the bad adapter, re-enable the adapter, using the hotplug feature provided by OS **202**, RTAS **208**, and EADS **214** and **218** working in concert. The user may then reconfigure the adapter once replaced.

Those of ordinary skill in the art will appreciate that the hardware and software depicted in **Figure 2** may vary. For example, more or fewer I/O adapters may be used than those depicted in **Figure 2**. The depicted example is not meant to imply architectural limitations with respect to the present invention.

With reference now to **Figure 4**, a flowchart illustrating an exemplary process of isolating a bad I/O adapter without powering down the system (i.e. a hotplug operation) is depicted in accordance with the present invention. Once the system is running and a machine check occurs indicating a bad I/O adapter, an AIX handler takes over control of the system (step **402**). The AIX handler may be implemented as OS handler **204** in **Figure 2**. The AIX handler retrieves the error log to which the RTAS entered the machine check (step **404**) and analyzes the

error log to determine the failing adapter (step **406**). The AIX handler then determines whether the failing card has been isolated (step **408**).

If the failing card has not been isolated by the RTAS, then the AIX handler causes the system to be shut down (step **410**). If the failing card has been isolated and determined, then disable the failing card by, for example, a call to RTAS-hot plug (step **412**). The AIX also deallocates any processes bound to the failing I/O adapter (step **414**). Once the failing I/O adapter has been successfully disabled, signal the appropriate expansion slot with, for example, blinking lights to indicate which I/O adapter has failed (step **416**). The AIX then posts a message informing a user to replace the bad adapter (step **418**). The message may be posted, for example, to a video display terminal or by page to an appropriate person assigned to maintaining the system.

With reference now to **Figure 5**, a flowchart illustrating an exemplary method of removing a bad I/O adapter via hotplug is depicted in accordance with the present invention. Once an adapter has been identified as bad, the RTAS, such as, for example, RTAS **208** in **Figure 2**, sets the visual indicator of the expansion slot in which the bad I/O adapter is located to alert a user that the expansion slot has been identified as containing a bad I/O adapter (step **502**). The visual indicator may be, for example, blinking lights on the expansion slot. The OS, such as OS **202** in **Figure 2**, unconfigures the devices associated with the bad I/O adapter (step **504**). This process may be performed, for example, by using the AIX command "rmdev".

31 OCT 2003

Once the devices associated with the bad I/O adapter have been unconfigured, all nodes corresponding to devices associated with the bad I/O adapter are removed from the OS copy of the open firmware device tree provided by RTAS (step **506**). RTAS then isolates and turns the power off to the expansion slot containing the bad I/O adapter (step **508**). RTAS then sets the affected expansion slot's visual indicator to a condition that informs a user that the I/O adapter connected to that expansion slot may be removed (step **510**). Once, the bad I/O adapter has been removed, the RTAS turns off the visual indicator to the affected expansion slot (step **512**) indicating that the user has removed the correct I/O adapter. The user may then insert a replacement I/O adapter and reconfigure the system to utilize the new I/O adapter. Thus, a bad I/O adapter may be removed and replaced during runtime (i.e. without shutting down the system).

Once the user has been notified of the bad I/O adapter, the process of removing and replacing the bad I/O adapter may be initiated through a menu, such as, for example, an AIX System Management Interface Tool (SMIT) menu. An example of a SMIT menu is depicted in **Figure 6**. Menu **600** provides a user with a list of hotplug services performed by the data processing system, such as data processing system **100** in **Figure 1**. In menu **600**, a user may select item 3 to place the bad I/O adapter in a condition in which the user can remove and replace the bad I/O adapter within the data processing system.

It is important to note that while the present invention has been described in the context of a fully

functioning data processing system, those of ordinary skill in the art will appreciate that the processes of the present invention are capable of being distributed in the form of a computer readable medium of instructions and a variety of forms and that the present invention applies equally regardless of the particular type of signal bearing media actually used to carry out the distribution. Examples of computer readable media include recordable-type media such a floppy disc, a hard disk drive, a RAM, and CD-ROMs and transmission-type media such as digital and analog communications links.

The description of the present invention has been presented for purposes of illustration and description, but is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. The embodiment was chosen and described in order to best explain the principles of the invention, the practical application, and to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are suited to the particular use contemplated.

CLAIMS:

What is claimed is:

1. A method of managing a failed input/output adapter within a data processing system, the method comprising:

coupling a plurality of hardware devices to an I/O bus utilizing a bridge such that the plurality of hardware devices communicate with the I/O bus utilizing the bridge;

receiving an indication that a hardware device within the plurality of hardware devices has failed;

attempting to identify the hardware device as a failed hardware device; and

responsive to successfully identifying the failed hardware device, disabling the failed hardware device without powering down the data processing system, wherein remaining ones of the plurality of hardware devices remain operational and continue to communicate with the I/O bus utilizing the bridge after the failed hardware device has been disabled.

2. The method as recited in claim 1, further comprising:

sending a message to a user indicating that one of the plurality of hardware devices has failed and the identity of the failed hardware device.

3. The method as recited in claim 2, wherein the step of sending a message comprises activating blinking lights on an expansion slot containing the failed hardware device to indicate to a user which one of the plurality of hardware devices is the failed hardware device.

4. The method as recited in claim 2, wherein the step of sending a message comprises paging a user.

5. The method as recited in claim 2, wherein the step of sending a message comprises displaying a message on a video display indicating that one of the plurality of hardware devices has failed and the identity of the failed hardware device.

6. The method as recited in claim 1, further comprising:

responsive to an indication that the failed hardware device has been replaced with a new hardware device, enabling the new hardware device.

7. The method as recited in claim 1, wherein the failed hardware device is an input/output adapter.

8. The method as recited in claim 1, further comprising:

deallocating any processes bound for the failed hardware device.

9. The method as recited in claim 1, further comprising:

responsive to a failure to identify the failed hardware device, powering down the data processing system.

10. The method as recited in claim 1, wherein the step of attempting to determine the identity comprises consulting an error log table.

11. A computer program product in a computer readable medium for use in a data processing system for managing a failed hardware device within the data processing system, the computer program product comprising:

first instructions for receiving an indication that a hardware device within a plurality of hardware devices has failed, the plurality of hardware devices being coupled to an I/O bus utilizing a bridge;

second instructions for attempting to identify the hardware device as a failed hardware device; and

third instructions, responsive to successfully identifying the failed hardware device, for disabling the failed hardware device without powering down the data processing system, wherein remaining ones of the plurality of hardware devices remain operational and continue to communicate with the I/O bus utilizing the bridge after the failed hardware device has been disabled.

12. The computer program product as recited in claim 11, further comprising:

31 OCT 2003

fourth instructions for sending a message to a user indicating that one of the plurality of hardware devices has failed and the identity of the failed hardware device.

13. The computer program product as recited in claim 12, wherein the step of sending a message comprises activating blinking lights on an expansion slot containing failed hardware device to indicate to a user which one of the plurality of hardware devices is the failed hardware device.

14. The computer program product as recited in claim 12, wherein the step of sending a message comprises paging a user.

15. The computer program product as recited in claim 12, wherein the step of sending a message comprises displaying a message on a video display indicating that one of the plurality of hardware devices has failed and the identity of the failed hardware device.

16. The computer program product as recited in claim 11, further comprising:

fourth instructions, responsive to an indication that the failed hardware device has been replaced with a new hardware device, for enabling the new hardware device.

17. The computer program product as recited in claim 11, wherein the failed hardware device is an input/output adapter.

18. The computer program product as recited in claim 11, further comprising:

fourth instructions for deallocating any processes bound for the failed hardware device.

19. The computer program product as recited in claim 11, further comprising:

fourth instructions, responsive to a failure to identify the failed hardware device, for powering down the data processing system.

20. The computer program product as recited in claim 11, wherein the step of attempting to determine the identity comprises consulting an error log table.

21. A system for managing a failed hardware device within a data processing system in which a plurality of hardware devices communicate with an I/O bus via a bridge, the system comprising:

first means for receiving an indication that a hardware device within the plurality of hardware devices has failed;

second means for attempting to identify the hardware device as a failed hardware device; and

third means, responsive to successfully identifying the failed hardware device, for disabling the failed hardware device without powering down the data processing system, wherein remaining ones of the plurality of hardware devices remain operational and continue to communicate with the I/O bus via the bridge after the failed hardware device has been disabled.

22. The system as recited in claim 21, further comprising:

fourth means for sending a message to a user indicating that one of the plurality of hardware devices has failed and the identity of the failed hardware device.

23. The system as recited in claim 22, wherein the step of sending a message comprises activating blinking lights on an expansion slot containing failed hardware device to indicate to a user which one of the plurality of hardware devices is the failed hardware device.

24. The system as recited in claim 22, wherein the step of sending a message comprises paging a user.

25. The system as recited in claim 22, wherein the step of sending a message comprises displaying a message on a video display indicating that one of the plurality of hardware devices has failed and the identity of the failed hardware device.

26. The system as recited in claim 21, further comprising:

31 OCT 2003

fourth means, responsive to an indication that the failed hardware device has been replaced with a new hardware device, for enabling the new hardware device.

27. The system as recited in claim 21, wherein the failed hardware device is an input/output adapter.

28. The system as recited in claim 21, further comprising:

fourth means for deallocating any processes bound for the failed hardware device.

29. The system as recited in claim 21, further comprising:

fourth means, responsive to a failure to identify the failed hardware device, for powering down the data processing system.

30. The system as recited in claim 21, wherein the step of attempting to determine the identity comprises consulting an error log table.

31. A method of replacing a failed input/output adapter in a data processing system in which a plurality of input/output adapters are coupled to an I/O bus via a bridge, the method comprising:

receiving an indication of a failed input/output adapter;

consulting an error log to determine an identity of the failed input/output adapter;

responsive to a lack of a determination of the identity of the failed input/output adapter, shutting down the data processing system; and

responsive to a determination of the identity of the failed input/output adapter:

refraining from shutting down the data processing system;

disabling the failed input/output adapter;

deallocating any processes bound to the failed input/output adapter; and

indicating to a user which one of a plurality of input/output adapters is the failed input/output adapter;

wherein the failed input/output adapter may be replaced without shutting down the system, remaining ones of the plurality of input/output adapters continuing to communicate with the I/O bus via the bridge after the failed input/output adapter has been disabled.

31 OCT 2003

32. The method as recited in claim 31, further comprising:

responsive to a determination that the failed input/output adapter has been replaced with a new input/output adapter, enabling the new input/output adapter.

33. A data processing system, comprising:

a bus system in which a plurality of hardware devices communicate with an I/O bus via a bridge;

a memory; and

a processor unit; wherein

the processor unit executes instructions to identify a failed hardware device within the plurality of hardware devices; and

responsive to successfully identifying the failed hardware device, disables the failed hardware device without powering down the data processing system, wherein remaining ones of the plurality of hardware devices remain operational and continue to communicate with the I/O bus via the bridge after the failed hardware device has been disabled.

34. The data processing system as recited in claim 33, further comprising:

responsive to a determination that the failed hardware device has been replaced by a replacing hardware device, enabling the new hardware device.

35. The data processing system as recited in claim 33, wherein the hardware device is an input/output adapter.

36. The data processing system as recited in claim 33, wherein the instructions are in the memory.

37. The data processing system as recited in claim 33, wherein the instructions are in a chip.

38. The data processing system as recited in claim 33, wherein the instructions are in an electrically erasable and programmable read only memory.

39. The data processing system as recited in claim 33, wherein the instructions are in read only memory.

40. The data processing system as recited in claim 33, wherein the instructions are in a non-volatile random access memory.