(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 March 2001 (29.03.2001)

PCT

(10) International Publication Number
WO 01/22209 A1

(51) International Patent Classification[7]: G06F 3/06

(21) International Application Number: PCT/US00/24848

(22) International Filing Date:
11 September 2000 (11.09.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/400,570 21 September 1999 (21.09.1999) US

(63) Related by continuation (CON) or continuation-in-part (CIP) to earlier application:
US 09/400,570 (CON)
Filed on 21 September 1999 (21.09.1999)

(71) Applicant *(for all designated States except US)*: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **RHOADS, Edward, R.** [US/US]; 15533 SW Bowment Court, Sherwood, OR 97140 (US). **KETRENOS, James, P.** [US/US]; 6280 McNeil Drive, #1801, Austin, TX 78729 (US).

(74) Agent: **TROP, Timothy, N.**; Trop, Pruner & Hu, P.C., 8554 Katy Freeway, Suite 100, Houston, TX 77024 (US).

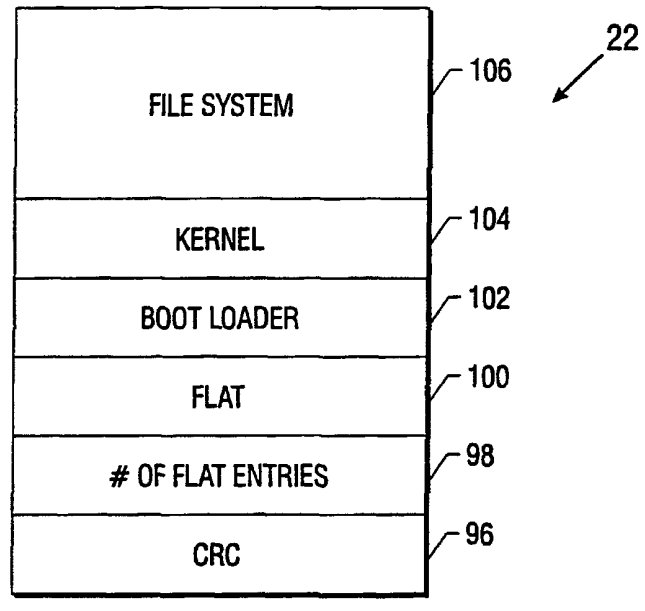
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: ORGANIZING INFORMATION STORED IN NON-VOLATILE RE-PROGRAMMABLE SEMICONDUCTOR MEMORIES



(57) Abstract: A plurality of partitions (96-106) may be formed in a non-volatile re-programmable memory (14) which may act as the primary non-volatile file system for a processor-based system. The memory may store, for example, the basic input/output system for the processor-based system together with its operating system. An address partition (100) may include information about the location of the other partitions, in association with information about the type of information stored in each partition.

WO 01/22209 A1

Organizing Information Stored In Non-Volatile Re-Programmable Semiconductor Memories

Background

This invention relates generally to processor-based systems using semiconductor memory as their primary, non-volatile, re-programmable storage medium.

There is increasing interest in so-called embedded processor-based systems. These systems often operate with reduced functionalities to provide desired performance at relatively low cost. In many cases, these embedded systems may be battery operated. Thus, their capabilities may be limited to improve battery lifetime.

For a variety of reasons including conserving battery life, reducing cost and providing a compact form factor, processor-based systems may be provided which do not use as their non-volatile storage medium, a hard disk drive. In many processor-based systems, a hard disk drive provides a convenient non-volatile storage medium that stores most of the information which the user desires to maintain permanently. This may include among other things, the operating system, application software, files and data, as examples. The information that is stored in the hard disk drive may be transferred for execution to system memory which conventionally is a volatile memory.

In many systems, hard disk drives provide a very high capacity, relatively quick storage medium. However, hard disk drives take more space and use more power than non-volatile semiconductor memories. In many embedded systems, re-programmable, non-volatile semiconductor memories are used as the primary storage system for processor-based systems. These semiconductor memories store the panoply of information normally stored in hard disk drives including operating systems.

In many cases, semiconductor memories utilized as primary non-volatile storage media for processor-based systems use FLASH memories. These FLASH memories may be re-programmed without user intervention using well known on-board capabilities. These memories are generally accessed using row and column addresses. Thus, the memories are generally monolithic in that the location of files and other data in that memory is generally stored outside the memory.

While this system has worked well with relatively simple embedded processor-based systems, as the demands on the processor-based systems increase, this simple storage system may be inadequate to handle some desired functions. Thus, there is a need for an improved

way of using non-volatile re-programmable semiconductor memories as the primary non-volatile file system for processor-based systems.

Summary

In accordance with one aspect, a method of organizing stored information on a non-volatile, re-programmable semiconductor memory includes partitioning the memory into a plurality of partitions, each having a defined address. A defined address for one partition is stored in another partition.

Other aspects are set forth in the accompanying detailed description and claims.

Brief Description of the Drawings

Fig. 1 is a schematic depiction of a client/server system in accordance with one embodiment of the present invention;

Fig. 2 is a depiction of the memory architecture of the storage device shown in Fig. 1;

Fig. 3 is a depiction of a memory architecture of a BIOS and recovery operating system used in the system shown in Fig. 2;

Fig. 4 is a flow chart for implementing software for re-loading operating systems in accordance with one embodiment of the present invention;

Fig. 5 is a depiction of a memory architecture for the primary operating system shown in Fig. 2;

Fig. 6 is a hardware implementation of the client shown in Fig. 1; and

Fig. 7 is a flow chart illustrating the operation of the FLAT shown in Fig. 5.

Detailed Description

A client/server computer system 10, shown in Fig. 1, may include one or more servers 18 that may be coupled over a network 16 to one or more clients 12. Each client 12 may have a storage device 14. The client 12 may be a processor-based system such as a desktop computer system, a handheld computer system, a processor-based television system, a set top box, an appliance, a thin client, a cellular telephone, or the like. The network 16 may be any of a variety of networks including a local area network (LAN), a metropolitan area network (MAN), a wide area network (WAN), a wireless network, a home network or an internetwork such as the Internet.

In the system 10, the client 12 may permanently store its operating system on a re-programmable storage device 14. The storage device 14 may conventionally be a hard disk drive or a FLASH memory. When the operating system is corrupted or needs updating, the client 10 can access the network 16 and the server 18 in order to obtain an uncorrupted or updated operating system and automatically re-load the new operating system onto the storage device 14.

The storage device 14 may be electrically reprogrammed. The storage device 14 may also act as the BIOS memory for the client 12 in one embodiment of the invention. While conventionally the BIOS memory is a read only memory (ROM), by using a re-programmable memory, the operating system as well as the BIOS code may be updated or replaced when corrupted, as will be explained hereinafter. In other embodiments a conventional BIOS ROM may be used in addition to the storage device 14.

A variety of FLASH memories are available for implementing the storage device 14, such as Intel's StrataFlash™ brand memory. One advantageous memory is the 28F64OJ5 eight megabyte FLASH array available from Intel Corporation. This memory includes a plurality of 128 kilobyte blocks. Each block may be data protected so that it may not be erased or overwritten. In other words, data protection may be selectively applied to one or more of a plurality of blocks in the memory.

The BIOS may be stored in one or more data protected blocks in the FLASH memory. Likewise, the recovery operating system may be stored in one or more blocks that are also data protected. In one embodiment, the BIOS may be stored in two 128 kilobyte blocks and the recovery operating system may use two 128 kilobyte blocks. The remainder of the memory may be utilized to store the primary operating system and a file system. Additional information about FLASH memories may be found in the "FLASH Memory" Databooks, January 1998, Order No. 2108830-017 available from Intel Corporation, Santa Clara, California.

Referring now to Fig. 2, the memory architecture of the storage device 14 may include addressable locations for a BIOS and recovery operating system 20 and a primary operating system 22. The primary operating system may be a general purpose operating system such as Microsoft Windows® 98 or CE, Linux, or the BE operating systems, as examples. The primary operating system may also be a real time operating system (RTOS) such as the PalmOS. The BIOS and recovery operating system 20 operates in cases where

the primary operating system 22 is corrupted or needs updating. The recovery operating system 20 may be an operating system of a reduced size which includes basic, essential BIOS functions and the limited software needed to obtain a new primary operating system. Thus, as used herein a "recovery operating system" is an operating system that is responsible for updating and/or obtaining a replacement for a primary operating system.

Referring to Fig. 3, in one embodiment of the invention, the recovery operating system 20 includes a kernel 26, a network interface controller (NIC) drivers 30 and a network stack 28. The kernel 26 is the core of the recovery operating system 20. The stack 28, for example, may include the User Datagram Protocol/Internet Protocol (UDP/IP), Trivial File Transfer Protocol (TFTP), Dynamic Host Control Protocol (DHCP), Address Resolution Protocol (ARP) and the boot strap protocol (BOOTP). (These protocols may be found at www.ietf.org/rfc.html.) The recovery operating system 20 may also include the operating system recovery and update application software 24. A FLASH driver 34 and BIOS services 35 may also be included. The FLASH driver is used to write a new primary operating system to the FLASH memory, where a FLASH memory is used as the storage device 14. The hardware interface 36 interfaces the software layers with a hardware motherboard.

Ideally, the recovery operating system 22 may be stripped down as much as possible to conserve memory. If possible, the kernel 26 may be reduced to only that code which is necessary to implement its recovery and update functions. One kernel which is particularly applicable is the LINUX kernel. The LINUX kernel includes an X-based kernel configuration utility called make xconfig. This utility provides a graphical user interface to facilitate selecting the elements of the kernel and the operating system. That is, the LINUX operating system allows the user to answer a series of questions, posed through a graphical user interface, indicating whether particular functionalities are desired. The code for de-selected functionalities may then be excluded. As a result, a relatively trimmed down operating system may be readily developed, without access to object code.

In the case of some software errors or crashes, the system may reboot, thereby resolving the error. A watchdog timer in the CMOS memory keeps a count of unsuccessful attempted reboots. If that number exceeds a threshold level (e.g. three), the recovery operating system may be invoked. When the system attempts to reboot, it checks the CMOS memory re-boot count and automatically boots the recovery operating system if the re-boot

count threshold is exceeded. The recovery operating system 20 is started so that a new version of the primary operating system image may be fetched.

The recovery operating system 20 may also acquire operating system updates. This may be done in a number of ways. In one embodiment, the user may request an update, thereby setting a separate update bit in the CMOS memory. In another embodiment, an operating system provider may broadcast a message to its users indicating that an update is available. The user systems that receive the message may have their update bit automatically set in CMOS memory. On the next attempted boot, the recovery operating system is booted to automatically acquire the update.

Alternatively, the recovery and update application software 24 may be configured so that the update is automatically acquired at a predicted low usage time. For example, if the system detects that the update bit is set, indicating an update is desired, the system may wait until the middle of the night to automatically download the update.

The recovery operating system in turn may communicate through the network interface controller and the network 16 to fetch a new version of the primary operating system image. This may be done by accessing another device in the same network or in another example, accessing the desired operating system over the Internet.

After the new operating system has been checked in system memory and loaded into the memory 14, the system is rebooted. When the system reboots the primary operating system, the primary operating system resets the update bit in CMOS memory.

In some cases when booting is attempted, an analysis of the stored operating system may determine that the operating system is corrupted. For example, during booting a checksum analysis may be undertaken. If the stored operating system is corrupted, a recovery bit may be set in the CMOS memory and the boot aborted. The next time a boot is attempted, the recovery bit is identified, and the system boots to the recovery operating system.

Referring now to Fig. 4, recovery and update application software 24 begins by checking the storage device 14 as indicated in diamond 40. Upon power up, after going through the power on self test (POST), the start-up code checks the primary operating system image in the memory 14 for checksum errors. If there is an error, the system boots the recovery operating system 20 and launches the recovery application. An error code may arise because the operating system image is corrupted or one of the recovery or update flags are

set. The recovery flag may be set, for example, because of a defect in the operating system. The update flags may be set, for example, because a time period has elapsed for an old primary operating system or because the user has indicated a desire to obtain an upgrade. Thus, after applying the checksum as indicated in block 42, the primary operating system is booted as indicated in block 44 if the checksum indicates a valid operating system. Otherwise, the recovery operating system is booted as indicated in block 46.

During the boot routine, start-up code which is part of the BIOS, sets the recovery bit in the CMOS memory if appropriate. The start-up code may also include the code for counting the number of times a reboot has been attempted and for storing information about the number of attempted reboots.

The application 24 may initiate a request over the network to the server 18 for an operating system download (block 48), in one embodiment of the present invention. Once the new image is downloaded, it is written to the storage device 14. The recovery bit is then cleared, as indicated in block 50, and the system reboots as indicated in block 52. The next time through, the system boots into the primary operating system and performs it usual functions.

The memory architecture of a portion of the storage device 14 storing the primary operating system 22, shown in Fig. 5, has, at the lowest memory address, a checksum or cyclic redundancy check (CRC) field 96. Above the checksum field 96 is a field 98 which indicates the number of entries in a FLASH allocation table (FLAT) 100. The FLASH allocation table partitions the FLASH memory portion 22 and allows multiple code and data images to be stored in the storage device 14. This in turn allows multiple boot loaders to exist within the FLASH memory for booting different operating system images. At boot time, the BIOS selects which boot loader to load and execute based on the status of the recovery bit, as described above.

The boot loader 102 for loading the primary operating system is stored above the flash allocation table 100. Above the boot loader 102 is the kernel 104 or core of the primary operating system. The primary operating system kernel may be the same or different from the kernel utilized by the recovery operating system. For example, while LINUX may be used for the recovery operating system, Windows® CE could be used in one embodiment for the primary operating system.

Above the kernel 104 is a file system 106. The FLASH allocation table 100 includes one entry for each item stored in the FLASH memory portion 22 including the items stored in the file system 106. The file system 106 includes files, directories and information used to locate and access operating system files and directories.

Each item contained in the FLASH allocation table includes information about the software version, the flags, the data offsets, the length of the data and its load address. The version number just keeps track of which version of software was loaded in a particular memory 14. The data offset determines where, in the FLASH memory, an entry is located.

The flag field has information about the nature of the respective entries. The least significant bit of the flag field includes information about the status of the cyclic recovery check (CRC). This in effect tells the BIOS whether a CRC must be calculated. The next most significant bit includes the block type. The block types include "boot" which indicates a boot loader, "kernel" or "file system". If the block type is boot loader, this flag field tells where, in random access memory, to load the boot loader out of the FLASH memory. An additional area in the flag field may be reserved for other information. A boot loader or bootstrap loader loads and passes control to another loader program which loads an operating system.

While the present invention may be used in connection with a variety of processor-based systems, an application which uses a set top computer system is illustrated in Fig. 6. A set top computer systems works with a television receiver. The client 12 may include a processor 65 coupled to an accelerated graphic board (AGP) chipset 66. The Accelerated Graphic Port Specification, Rev. 2.0 is available from Intel Corporation of Santa Clara, California. The chipset 66 may be coupled to system memory 68 in the accelerated graphics port bus 70. The bus 70 in turn may be coupled to a graphic accelerator 72, also coupled to a video or television receiver 73.

A portion 75 of system memory 68, called the CMOS memory, may be implemented by a memory integrated circuit which is adapted to save system data. Conventionally, the CMOS includes the real time clock (RTC), which keeps the time of day. The recovery and update bits are stored in the CMOS memory at predefined locations.

The chipset 66 may also be coupled to a bus 74 that receives a television tuner/capture card 76. The card 76 may be coupled to a television antenna 78 which may also be a satellite antenna or a cable connection as additional examples. An interface to a network 16, such as a

modem interface connection to the Internet or a network interface controller connection to a computer network may also be provided.

A bridge 80 may in turn be coupled to another bus 84 which supports a serial input/output interface 86 and a memory interface 94. The interface 86 may be coupled to a modem 88 or a keyboard 92. The interface 94 may couple the FLASH memory 14 storing the recovery operating system and BIOS 20 and the primary operating system 22. The bridge 80 may be the 82371AB PCI ISA IDE Xcelerator (PIIX4) chipset available from Intel Corporation. Thus, it may include a general purpose input/output pins (GP[I,O]).

With the number of chipsets used to implement computer systems, the chipset may be set up so that it sees only a certain number of lines of BIOS at any one time. In embodiments in which the primary operating system and the recovery operating system are stored in FLASH memory, they may be accessed in the same way as the BIOS memory is accessed. Thus, since the FLASH memory that is accessed is considerably larger than a BIOS memory, it may be desirable to use other techniques to allow accessing all of the memory stored in the FLASH. One technique for doing this in processors from Intel Corporation is to use the GP[I,O] pins, for example on the PIIX4 device. These pins can be coupled to the pins responsible for developing the signals reading the BIOS. By providing appropriate GP[I,O] signals, FLASH memory reading may be bank switched to sequentially read the entire memory.

Turning now to Fig. 7, in accordance with one embodiment, software that uses the FLAT to allow multiple code and data images to be stored in FLASH memory, begins on power up or system reset with the BIOS executing and performing system initialization and Power on Self Test activities (block 110). The contents of the FLASH memory may be validated by checking the CRC stored at field 96 in the FLASH memory, as indicated in block 112. At this point, the BIOS selects the boot loader (block 114) to execute by scanning the FLAT and selecting the entry marked as the boot loader. The boot loader then uses the FLAT to find where in the FLASH memory the primary operating system is located (block 116), loads the operating system at the appropriate address in system memory (block 118) and starts its execution (block 120).

In some embodiments the BIOS may continue to be independent from the operating system. The operating system dependencies can reside in the boot loader. The boot loader allows a conventional computer operating system to reside in FLASH memory.

While the present invention has been illustrated in connection with an embodiment wherein the primary operating system and the recovery operating system are stored in a storage device such as a FLASH memory, other re-programmable storage devices may be utilized as well. In the case of FLASH memory, given current economies, the memory is relatively expensive and mirroring is generally not used. Thus, the use of the recovery operating system in connection with FLASH memories is particularly advantageous.

However, the present invention may be utilized in connection with other configurations. For example, in systems that store the primary operating system in a hard disk drive, the recovery operating system may also be included on the hard disk drive. The BIOS may continue to be stored in a BIOS ROM in such cases, if desired.

Alternatively, the recovery operating system may actually be provided on an external or removable memory, such as a compact disc ROM (CD-ROM). When necessary, the user may simply load the CD-ROM into a CD-player. A processor executes the recovery operating system off of the CD-ROM, and then uses the recovery and update application software to update and replace the primary operating system. This approach offers advantages over providing the full operating system in disk form since the use of a compact recovery operating system facilitates updates. That is, the compact recovery system may be quickly loaded and used to acquire updates. Otherwise, a full operating system would need to be provided in disk form to each user, for each update, so that the user can then acquire the updates.

In addition, while the present invention has been described with respect to a client/server environment, the present invention is available to a variety of other environments. For example, the present invention may be implemented on a server in a client/server environment. In addition, it is applicable to stand-alone computer systems including processor-based systems that are battery powered. For example, in connection with hand-held computer systems, the present invention may provide an update or replacement functionality using available wired or wireless communication links. In a system which may be temporarily hard wire linked to a desktop computer, such as a PalmPilot personal digital assistant, the recovery operating system may communicate with the desktop to obtain a new operating system. Similarly, upgrades may be obtained using a variety of wireless communication links including radio and cellular telephone links. Moreover, in systems

which are linked through cable or satellite broadcast systems, new operating systems may be achieved using these communication links as well.

In connection with custom operating systems, it may be necessary to go to a specific remote location in order to update or replace the operating system. However, in connection with non-custom operating systems, a variety of sites within the user's extended computer system, accessible over the Internet or over a variety of communication links, may be utilized to acquire such replacements. In addition, a plurality of such sites may be preprogrammed into the recovery operating system application software so that if the system is unsuccessful in acquiring the needed replacement at one location, it can query a plurality of other locations.

In some cases, the recovery application software can not be programmed with information about additional locations which contain future updates. However, when an operating system provider broadcasts information about updates, that broadcast may also include information about how to automatically acquire the desired updates. This information may then be used by the recovery application software.

In some embodiments, the system user is oblivious to the operation of the recovery operating system. The recovery operating system works in the background making the primary operating system to appear to the user to be more robust.

While the present invention has been described with respect to a limited number of embodiments, those skilled in the art will appreciate numerous modifications and variations therefrom. It is intended that the appended claims cover all such modifications and variations as fall within the true spirit and scope of this present invention.

What is claimed is:

1. A method of organizing stored information on a non-volatile, re-programmable semiconductor memory comprising:

partitioning said memory into a plurality of partitions, each having a defined address; and

storing the defined address for one partition in another partition.

2. The method of claim 1 further including storing information about the number of partitions.

3. The method of claim 1 further including storing a boot loader in one of said partitions.

4. The method of claim 1 further including storing a file system in one of said partitions.

5. The method of claim 1 further including storing a kernel for an operating system in one of said partitions.

6. The method of claim 1 further including storing information in association with said addresses about whether or not an integrity check needs to be done on the data stored at the associated address.

7. The method of claim 1 further including storing, in association with the address of a partition, information about the type of information stored in the partition.

8. The method of claim 7 further including storing information about whether or not the information stored at a given partition is a boot loader, a kernel or a file system.

9. The method of claim 7 including storing information about the load address for said information in association with said address.

10. A method of initializing a processor-based system comprising:

validating information stored in a non-volatile, re-programmable semiconductor memory; and

using a allocation table stored in said memory to find an operating system stored in said memory;

loading said operating system; and

executing said operating system.

11. The method of claim 10 further including selecting a boot loader to load said operating system.

12. The method of claim 11 including using said allocation table to locate said boot loader.

13. The method of claim 10 including performing initialization and the power on self test before validating information stored in said memory.

14. The method of claim 10 including validating information stored in said memory using a cyclic recovery check software stored in said memory.

15. An article comprising a medium storing instructions that cause a processor-based system to:

validate information stored in a non-volatile, re-programmable semiconductor memory;

use an allocation table to find an operating system stored in said memory;

load said operating system; and

execute said operating system.

16. A processor-based system comprising:

a processor;

a volatile memory coupled to said processor; and

a re-programmable, non-volatile semiconductor memory coupled to said processor, said semiconductor memory including a plurality of partitions, one of said partitions storing an operating system and another of said partitions storing the addresses of the other partitions in association with information about what is stored in each of said partitions.

17. The system of claim 16 wherein said semiconductor memory is a FLASH memory.

18. The system of claim 16 wherein one of said partitions stores a basic input/output system.

19. The system of claim 16 wherein one of said partitions stores a file system.

20. The system of claim 16 wherein one of said partitions stores a boot loader.

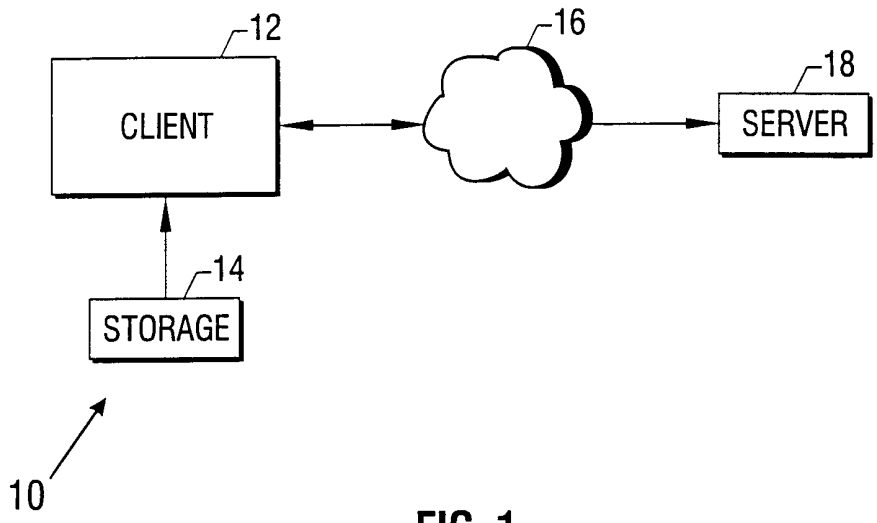


FIG. 1

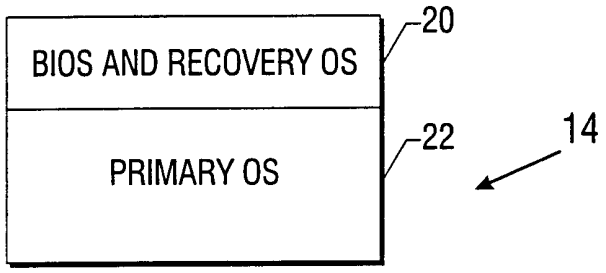


FIG. 2

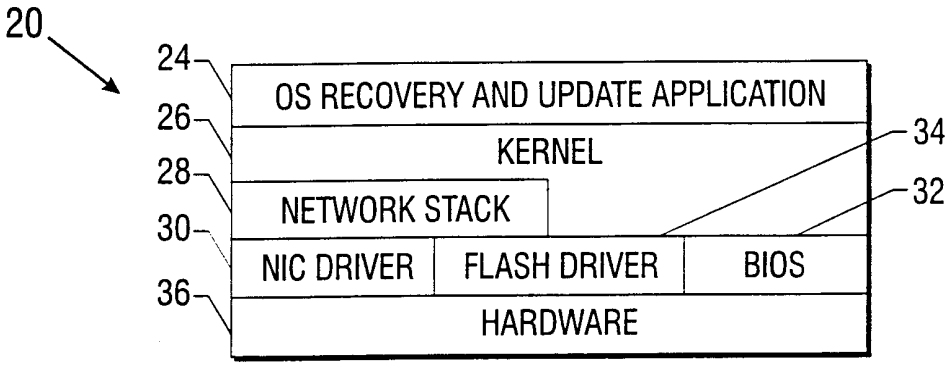


FIG. 3

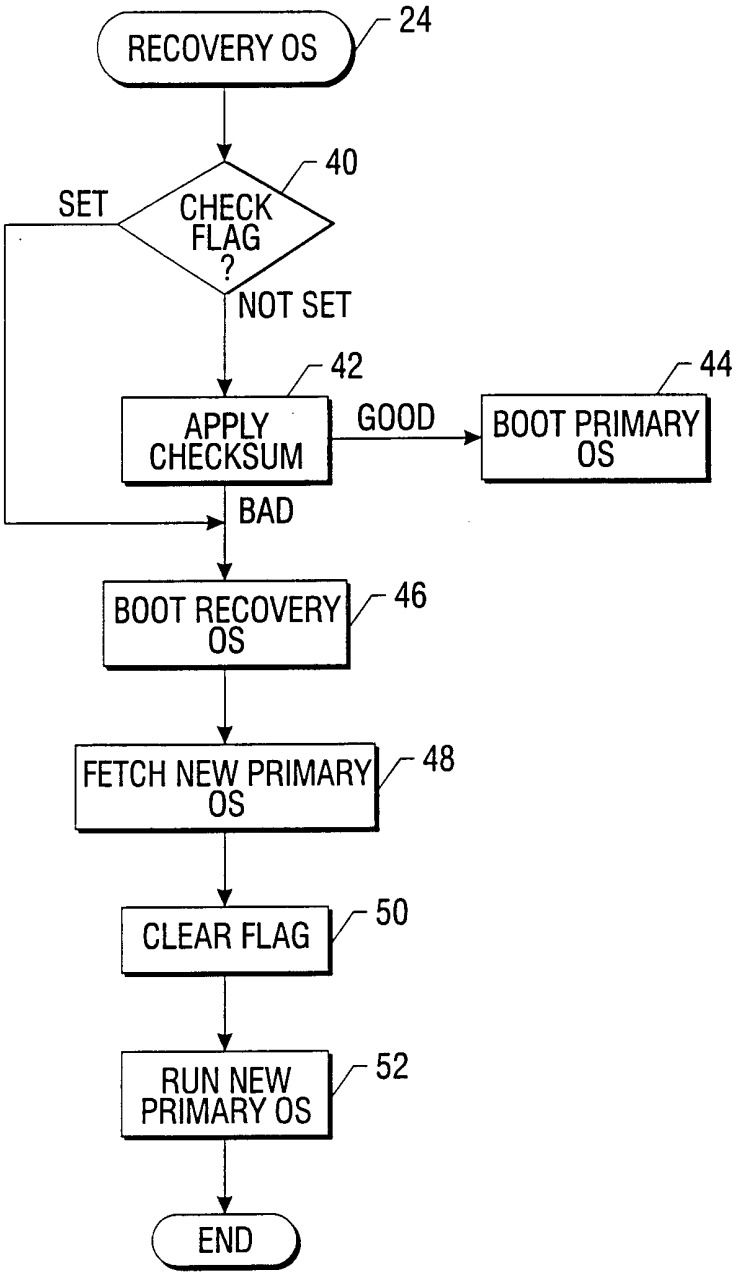

RECOVERY OS
24
40
SET
CHECK FLAG ?
NOT SET
42
APPLY CHECKSUM
GOOD
44
BOOT PRIMARY OS
BAD
BOOT RECOVERY OS
46
FETCH NEW PRIMARY OS
48
CLEAR FLAG
50
RUN NEW PRIMARY OS
52
END

FIG. 4

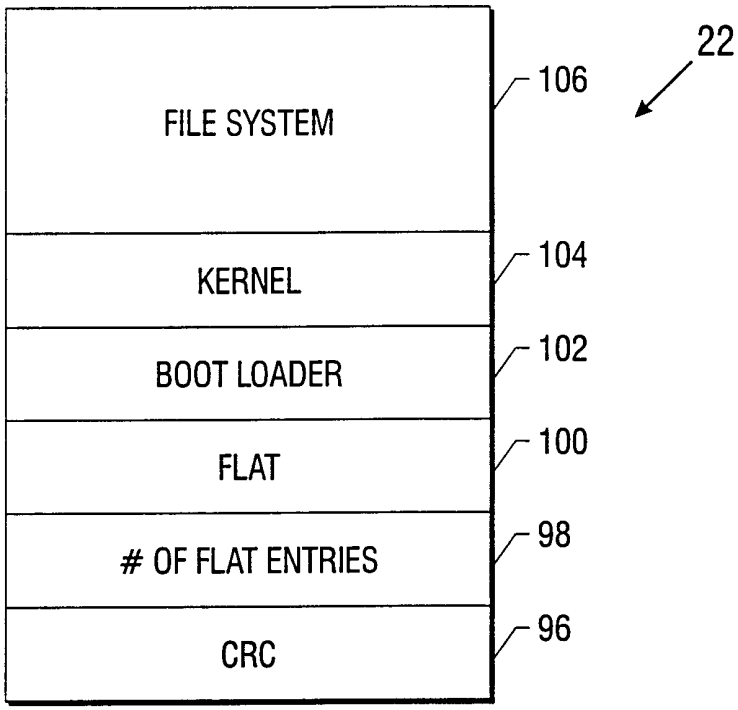

22
FILE SYSTEM
106
KERNEL
104
BOOT LOADER
102
FLAT
100
OF FLAT ENTRIES
98
CRC
96

FIG. 5

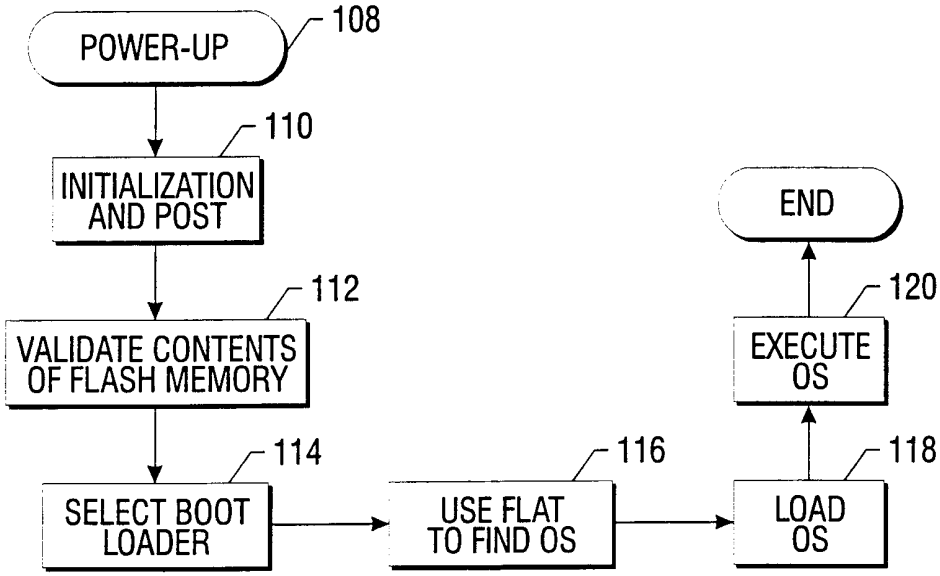

POWER-UP
108
INITIALIZATION AND POST
110
VALIDATE CONTENTS OF FLASH MEMORY
112
SELECT BOOT LOADER
114
USE FLAT TO FIND OS
116
LOAD OS
118
EXECUTE OS
120
END

FIG. 7

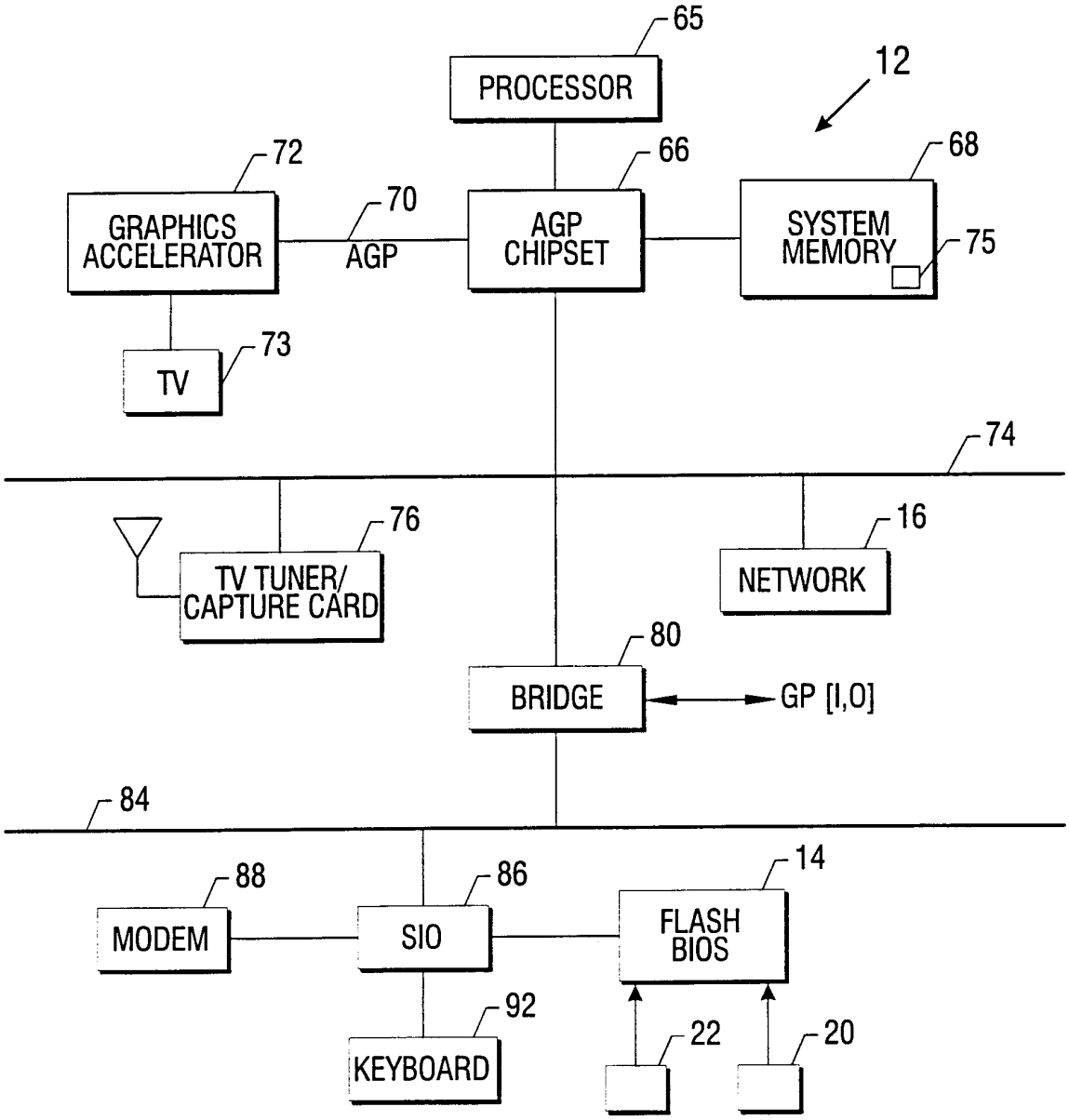

65
PROCESSOR
12
72
66
68
GRAPHICS ACCELERATOR
70
AGP
AGP CHIPSET
SYSTEM MEMORY
75
TV
73
74
76
16
TV TUNER/ CAPTURE CARD
NETWORK
80
BRIDGE
GP [I,O]
84
88
86
14
MODEM
SIO
FLASH BIOS
92
22
20
KEYBOARD

FIG. 6

INTERNATIONAL SEARCH REPORT

Interr ıal Application No
PCT/US 00/24848

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F3/06

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data, PAJ, INSPEC, IBM-TDB

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	AUZAS E: "DIE KUNST DES RICHTIGEN DIMENSIONIERENS. BAUSTEINE FUR DIE ENTWICKLUNG EINES EMBEDDED-PC" ELEKTRONIK,DE,FRANZIS VERLAG GMBH. MUNCHEN, vol. 45, no. 3, 6 February 1996 (1996-02-06), pages 66-75, XP000553705 ISSN: 0013-5658 page 74, right-hand column, line 40 -page 75, left-hand column, line 9 ---	1-20
X	US 5 745 418 A (LEE JU-XU ET AL) 28 April 1998 (1998-04-28) column 3, line 57 -column 5, line 52; figure 1 -----	1-20

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
22 December 2000	10/01/2001

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Lindquist, J

1

INTERNATIONAL SEARCH REPORT

...ormation on patent family members

Interr ıal Application No
PCT/US 00/24848

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5745418 A	28-04-1998	EP 0882264 A	09-12-1998
		US 5933368 A	03-08-1999
		WO 9824029 A	04-06-1998