(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 419 004 B1

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention of the grant of the patent:
29.05.1996 Bulletin 1996/22

(51) Int Cl.[6]: **G06F 9/445**

(21) Application number: **90307295.7**

(22) Date of filing: **04.07.1990**

(54) Computer system with program loading apparatus and loading method

Rechnersystem mit Programmladegerät und Ladeverfahren

Système d'ordinateur avec dispositif pour charger des programmes et méthode de chargement

(84) Designated Contracting States:
AT BE CH DE ES FR GB IT LI NL SE

(30) Priority: **25.08.1989 US 399631**

(43) Date of publication of application:
27.03.1991 Bulletin 1991/13

(73) Proprietor: **International Business Machines Corporation**
Armonk, N.Y. 10504 (US)

(72) Inventors:
- **Bealkowski, Richard**
  Delray Beach, Florida 33444-1033 (US)
- **Blackledge, John Wiley, Jr.**
  Boca Raton, Florida 33487 (US)
- **Cronk, Doyle Stanfill**
  Boca Raton, Florida 33433 (US)
- **Dayan, Richard Alan**
  Boca Raton, Florida 33487 (US)
- **Kinnear, Scott Gerald**
  Boca Raton, Florida 33496 (US)
- **Kovach, George D.**
  Boca Raton, Florida 33434 (US)
- **Palka, Matthew Stephen, Jr.**
  Raleigh, NC 27614 (US)

(74) Representative: **Burt, Roger James, Dr.**
IBM United Kingdom Limited
Intellectual Property Department
Hursley Park
Winchester Hampshire SO21 2JN (GB)

(56) References cited:
US-A- 3 427 593 **US-A- 4 748 561**

- **PATENT ABSTRACTS OF JAPAN vol. 13, no. 391 (P-926)(3739) 30 August 1989 & JP-A-1 140 230**
- **PATENT ABSTRACTS OF JAPAN vol. 13, no. 422 (P-933)(3770) 20 September 1989 & JP-A-1 154 226**
- **PATENT ABSTRACTS OF JAPAN vol. 9, no. 269 (P-400)(1992) 26 October 1985 & JP-A-60 116 054**
- **PATENT ABSTRACTS OF JAPAN vol. 11, no. 129 (P-570) 23 April 1987 & JP-A-61 272 852**
- **IBM TECHNICAL DISCLOSURE BULLETIN. vol. 31, no. 1, June 1988, NEW YORK US pages 352 - 354; 'Configurable Boot Device and Drive Remapping for Personal Computers'**
- **IBM TECHNICAL DISCLOSURE BULLETIN. vol. 27, no. 1B, June 1984, NEW YORK US pages 481 - 482; D. J. BRADLEY ET AL.: 'Feature Extensions for IBM Personal Computer'**

EP 0 419 004 B1

Description

The present invention relates to a computer system and in particular to apparatus for loading BIOS from a diskette into a personal computer system.

Personal computer systems in general have attained widespread use for providing computer power to many segments of today's modern society. Personal computer systems can usually be defined as a desk top, floor standing, or portable microcomputer that consists of a system unit having a single system processor, a display monitor, a keyboard, one or more diskette drives, a fixed disk storage, and an optional printer. One of the distinguishing characteristics of these systems is the use of a motherboard or system planar to electrically connect these components together. These systems are designed primarily to give independent computing power to a single user and are inexpensively priced for purchase by individuals or small businesses. Examples of such personal computer systems include the IBM PERSONAL COMPUTER AT and the IBM PERSONAL SYSTEM/2 Models 30, 50, 60, 70 and 80.

These systems can be classified into two general families. The first family, usually referred to as Family I Models, use a bus architecture exemplified by the IBM PERSONAL COMPUTER AT. The second family, referred to as Family II Models, use MICROCHANNEL bus architecture exemplified by the IBM PERSONAL SYSTEM/2 Models 50 through 80.

In early examples of family I personal computer systems such as the IBM Personal Computer, it was recognised that software compatibility would be of utmost importance. In order to achieve this goal, an insulation layer of system resident code, also referred to as "microcode", was established between the hardware and software. This code provided an operational interface between a user's application program/operating system and the device to relieve the user of the concern about the characteristics of hardware devices. Eventually, the code developed into a Basic Input/Output system (BIOS), for allowing new devices to be added to the system, while insulating the application program from the peculiarities of the hardware. The importance of BIOS was immediately evident because it freed a device driver from depending on specific device hardware characteristics while providing the device driver with an intermediate interface to the device. Since BIOS was an integral part of the system and controlled the movement of data in and out of the system processor, it was resident on the system planar and was shipped to the user in a read only memory (ROM). For example, BIOS in the IBM Personal Computer occupied 8K of ROM resident on the planar board.

As new models of the personal computer family were introduced, BIOS had to be updated and expanded to include new hardware and I/O devices. As could be expected, BIOS started to increase in memory size. For example, BIOS in the IBM PERSONAL COMPUTER AT, occupied 32K bytes of ROM.

Today, with the development of new technology, personal computer systems of the Family II models are growing even more sophisticated and are being made available to consumers more frequently. Since the technology is rapidly changing and new I/O devices are being added to the personal computer systems, modification to the BIOS has become a significant problem in the development cycle of the personal computer system.

For instance, with the introduction of the IBM Personal System/2 with MICROCHANNEL architecture, a significantly new BIOS, known as advanced BIOS, or ABIOS, was developed. However, to maintain software compatibility, BIOS from the Family I models had to be included in the Family II models. The Family I BIOS became known as Compatibility BIOS or CBIOS. However, as previously explained with respect to the IBM PERSONAL COMPUTER AT, only 32K bytes of ROM were resident on the planar board. Fortunately, the system could be expanded to 96K bytes of ROM. Unfortunately, because of system constraints, this turned out to be the maximum capacity available for BIOS. Luckily, even with the addition of ABIOS, ABIOS and CBIOS could still squeeze into 96K of ROM. However, only a small percentage of the 96K ROM area remained available for expansion. With the addition of future I/O devices, CBIOS and ABIOS will eventually run out of ROM space. Thus, new I/O technology will not be able to be easily integrated within CBIOS and ABIOS.

Due to these problems, plus the desire to make modification in Family II BIOS as late as possible in the development cycle, it became necessary to off load portions of BIOS from the ROM. To accomplish this, portions of BIOS were stored and loaded from a fixed disk. However, it quickly became evident that loading only from a fixed disk had some limitations. Mainly, if the disk became Incapacitated, the system was unusable. Also, updates to the system would lead to compatibility problems between BIOS and the new system configuration. Therefore, a need exists for loading BIOS from a direct access storage device other than the fixed disk. Also it is highly desireable to provide a priority and recovery mode for the diskette loaded BIOS. In the priority mode, BIOS is loaded immediately from diskette. In the recovery mode, BIOS is loaded from diskette only after failing to load from disk.

An aim of the present invention is to provide a computer system having apparatus for storing and loading a portion of BIOS from a diskette drive.

In accordance with the present invention, there is now provided a computer system as claimed in claim 1 and a method for loading BIOS from a diskette drive as claimed in claim 20.

The signal producing means preferably generates a priority signal, the priority signal being representative of a mode wherein the master boot record and remaining portion of BIOS are loaded into random access memory before attempting to access the fixed disk drive.

Also, the signal producing means preferably generates a recovery signal, the recovery signal being representative of a mode wherein the master boot record and remaining portion of BIOS are loaded into random access memory after attempting to load BIOS from the fixed disk drive.

Preferably, the apparatus further includes a nonvolatile memory electrically coupled to the system processor, wherein in the absence of the priority signal and the recovery signal the master boot record and the remaining portion of BIOS are loaded into random access memory after attempting to load BIOS from the fixed disk drive and detecting the absence of a password normally included in the nonvolatile memory.

In preferred examples of the present invention the signal producing means comprises a switch electrically connected to a system planar, the system planar being electrically coupled to the system processor.

The switch preferably comprises a hardwired jumper resident on the system planar.

The master boot record preferably includes a data segment, the data segment representing a hardware configuration of the personal computer system which is compatible with the master boot record, and further wherein the read only memory includes data representing a hardware configuration of the system processor, wherein before the remaining portion of BIOS is loaded into random access memory, the executable code segment compares the hardware configuration data from the master boot record with the hardware configuration data from the read only memory to verify the master boot record is compatible with the system processor.

The data segment of the master boot record preferably includes a value representing the system planar which is compatible with the master boot record and further wherein the system planar further includes a means for uniquely identifying the system planar in order to verify that the master boot record is compatible to the system planar.

The hardware configuration data on the master boot record preferably includes a model value and a submodel value, wherein the model value identifies a system processor which is compatible with the master boot record and the submodel value represent an I/O configuration of the system planar which is compatible with the master boot record, and further wherein the read only memory includes a corresponding model value identifying the system processor and submodel value representing the I/O configuration of the system planar, wherein the model value and submodel value of the master boot record are compared to the corresponding model and submodel values of the read only memory respectively, in order to verify that the master boot record is compatible with the system processor and the I/O configuration of the system planar.

In another preferred example of the present invention, the master boot record includes a predetermined character code in order to distinguish the master boot record from other records included on the diskette drive.

In a further example of the present invention the signal producing means generates a recovery signal, the recovery signal being representative of a mode wherein the master boot record and the remaining portion of BIOS are loaded into random access memory after attempting to load BIOS from the disk drive.

The executable code segment preferably generates a first error to indicate the master boot record is not compatible with the system hardware.

The nonvolatile random access memory preferably includes data representing the system configuration, the data being updated when the configuration of the system is changed, wherein the executable code segment compares the data in the nonvolatile random access memory to corresponding data in the read only memory to determine if the configuration of the system has changed.

The executable code segment preferably generates a second error to indicate that the system configuration has changed.

The master boot record preferably includes an identifying means to identify the record in order to distinguish the master boot record from other records included on the diskette drive.

The identifying means preferably comprises a predetermined character code.

The predetermined code is preferably prefatory to the code segment of the master boot record.

The master boot record preferably includes a checksum value to verify the validity of the master boot record when loaded into the random access memory.

The remaining portion of BIOS preferably includes a checksum value to verify the validity of the remaining portion of BIOS when loaded into the random access memory.

The master boot record preferably includes a predetermined pattern, and further wherein the read only memory includes a corresponding predetermined pattern in order to verify that the first portion of BIOS is included within a predef ined read only memory.

The executable code segment preferably generates a third error to indicate that the read only memory is not compatible with the master boot record.

Viewing the present invention from a fourth aspect, there is now provided, a personal computer system comprising: a system processor; a random access memory being the main memory and electrically coupled to the system processor; a diskette drive being electrically coupled to the system processor, the diskette drive capable of storing a plurality of data records; a master boot record included in the diskette drive, the master boot record having a data segment and an executable code segment, the data segment representing a hardware configuration of the personal computer system which is compatible with the master boot record; a read only memory being electrically coupled to the system processor, the read only memory having data rep-

resenting a hardware configuration of the system; a first portion of BIOS being included in the read only memory, the first portion of BIOS initializing the system and the diskette drive to load in the master boot record and transferring control to the executable code segment, the executable code segment further comparing the hardware configuration data from the master boot record to the hardware configuration data of the read only memory to verify the compatibility of the master boot record with the system processor; a remaining portion of BIOS being included in the diskette driver wherein, after verifying the compatibility of the master boot record with the system processor, executable code segment loads the remaining portion of BIOS into the random access memory.

The computer system preferably includes a system planar board being electrically coupled to the system processor, wherein the data segment of the master boot record includes a value representing a system planar being compatible with the master boot record and further wherein the system planar further includes a means for uniquely identifying the system planar in order to verify the compatibility of the master boot record to the system planar.

In preferred exam-ples of the present invention, the computer system further includes a system planar board being electrically coupled to the system processor, wherein the hardware configuration data on the master boot record includes a model value and a submodel value, wherein the model value identifies the system processor and the submodel value represent the I/O configuration of the system planar, the model value and submodel value being compared to corresponding values in the read only memory to verify the compatibility of the master boot record to the hardware configuration.

Preferred examples of the present invention also include a nonvolatile random access memory, the nonvolatile random access memory storing values representing the system configuration, the values being updated when the configuration of the system is changed, wherein the first portion of BIOS compares the values in the nonvolatile random access memory to corresponding values in the read only memory to determine if the configuration of the system has changed.

Viewing the present invention from a fifth aspect, there is now provided a method for loading BIOS from a diskette drive in a personal computer system normally connected to a fixed disk drive, the personal computer system having a system processor electrically coupled to a system planar, the planar further being electrically coupled to a read only memory random access memory and nonvolatile memory, the method comprising the steps of:

(a) initializing the system with a first portion of BIOS resident in the read only memory;

(b) determining if the fixed disk drive is present;

(c) in the absence of the fixed disk drive, searching the nonvolatile memory for a password;

(d) upon detecting the absence of the password, initializing with the first portion of BIOS the diskette drive having a master boot record and the remaining BIOS;

(e) loading with the first portion of BIOS the master boot record into random access memory, the master boot record including a data segment and an executable code segment, the data segment having data representing the hardware configuration of the system for which the remaining BIOS is compatible;

(f) verifying the compatibility of the master boot record with the system hardware by comparing from the data segment the data representing the hardware configuration record to corresponding data stored in read only memory;

(g) executing the code segment of the master boot record to load the remaining BIOS code into main memory; and

(h) passing control to the remaining BIOS code once it is loaded into main memory.

The step (d) of verifying preferably includes the steps of:

(i) verifying that the boot record is compatible with the planar by comparing a planar ID accessible by the system processor with a planar ID value stored in the data segment of the boot record; and

(j) verifying that the boot record is compatible with the system processor by comparing respectively model and submodel values stored in read only memory with model and submodel values stored in the data segment of the boot record.

The method preferably includes the steps of:

(k) comparing the data in nonvolatile random access memory to the data in read only memory to determine whether the system configuration has changed; and

(l) generating an indication that the system configuration as changed before loading the BIOS from the diskette.

In an example of the present invention, there is provided, a personal computer system comprising a system processor, a random access memory, a read only memory, a diskette drive, a switching means and normally a disk drive. The read only memory includes a first portion

of BIOS which in operation, initializes the system and detects the state of the switching means electrically coupled to the system processor. If the switching means is in a priority position, a master boot record is immediately loaded from the diskette drive (priority mode) without testing for the fixed disk drive. If the switching means is in a recovery position the first portion of BIOS attempts to load the master boot record from the fixed disk. If the fixed disk is not available, non-operational, or the master boot record on the disk is invalid, the first portion of BIOS reads in the master boot record from the diskette drive (recovery mode).

Whether loaded from the fixed disk or diskette, the master boot record includes a data segment and an executable code segment. The data segment includes data representing system hardware and a system configuration which is compatible with the master boot record. The first BIOS portion transfers control to the executable code segment which confirms the master boot record is compatible with the system hardware by verifying that the data from the data segment of the master boot record agrees with data stored in the read only memory representing the system processor, system planar, and planar I/O configuration.

If the master boot record is compatible with the system hardware, the executable code segment confirms that the system configuration has not changed and loads in the remaining BIOS portion from either the disk drive or the diskette drive into random access memory. The executable code segment then verifies the authenticity of the remaining BIOS portion and vectors the system processor to begin executing the remaining BIOS now in random access memory. BIOS executing in random access memory then boots up the operating system to begin operation of the personal computer system. The first portion of BIOS, being no longer addressable and superseded by the remaining portion of BIOS, is abandoned.

In a preferred embodiment of the present invention to be described later, there is provided a computer system having apparatus for loading BIOS from a diskette drive wherein the computer system normally loads BIOS from a fixed disk drive. Furthermore, the apparatus can confirm the compatibility between BIOS stored on the diskette drive and the hardware configuration of the personal computer system. The apparatus preferably comprises a signal producing means, the first BIOS portion, the master boot record, signal responding means, and the remaining portion of BIOS. The signal producing means, such as the switching means, produces a signal representative of the mode for diskette loading. The signal responding means included within the first BIOS portion is responsive to the signal to determine if BIOS is loaded from diskette. The first BIOS portion initializes the system and then, if necessary, initializes the diskette to effect the loading of the master boot record into random access memory. The master boot record includes executable code which is activated by the first BIOS portion to effect the loading of the remaining portion of BIOS into random access memory.

An embodiment of the present invention will now be described with reference to the accompanying drawings in which:

Fig. 1 illustrates a cut away view of a personal computer system showing a system planar board connected to a plurality of direct access storage devices;

Fig. 2 shows a system block diagram for the personal computer system of Fig. 1;

Fig. 3 is a memory map for the ROM BIOS included on the planar board;

Fig. 4 is a flowchart describing the overall process for loading a BIOS image from a direct access storage device;

Fig. 5 illustrates the record format for the master boot record;

Fig. 6A is a flowchart describing the operation of the IBL routine;

Fig. 6B is a flowchart showing the steps for loading the master boot record from a fixed disk;

Fig. 6C is a flowchart showing the steps for loading the BIOS image from a diskette;

Fig. 6D is a flowchart showing greater detail in checking the compatibility between the master boot record and the planar/processor; and

Fig. 7 is a detailed flowchart showing the execution of the master boot record to load the BIOS image from a direct access storage device.

The following detailed description is of the best presently contemplated mode for carrying out the invention. This description is not to be taken in a limiting sense but is made merely for the purpose of illustrating the general principles of the invention since the scope of the invention is best defined by the appended claims.

Referring now to the drawings, and in particular to Fig. 1, there is shown a cutaway version of a personal computer system 10, having a plurality of DASD (Direct Access Storage Devices) 12 - 16 connected to a system or planar board 24 through a plurality of I/O slots 18. A power supply 22 provides electrical power to the system 10 in a manner well known. The planar board 24 includes a system processor which operates under the control of an operating system to input, process, and output information.

In use, the personal computer system 10 is de-

signed primarily to give independent computing power to a small group of users or a single user and is inexpensively priced for purchase by individuals or small businesses. In operation, the system processor operates under the operating system, such as IBM's OS/2 Operating System or PC-DOS. This type of operating system includes a BIOS (previously discussed and defined) interface between the DASD 12 - 16 and the Operating System. A portion of BIOS divided into modules by function is stored in ROM on the planar 24 and hereinafter will be referred to as ROM-BIOS. BIOS provides an interface between the hardware and the operating system software to enable a programmer or user to program their machines without an in depth operating knowledge of a particular DASD. For example, a BIOS diskette module permits a programmer to program the diskette drive without an indepth knowledge of the diskette drive hardware. Thus, a number of diskette drives designed and manufactured by different companies can be used in the system. This not only lowers the cost of the system 10, but permits a user to choose from a number of diskette drives.

Prior to relating the above structure to the present invention, a summary of the operation in general of the personal computer system 10 may merit review. Referring to Fig. 2, there is shown a block diagram of the personal computer system 10. Fig. 2 illustrates components of the planar 24 and the connection of the planar 24 to the I/O slots 18 and other hardware of the personal computer system. Located on the planar 24 is the system processor 26 comprised of a microprocessor which is connected by a local bus 28 to a memory controller 30 which is further connected to a random access memory (RAM) 32. While any appropriate microprocessor can be used, one suitable microprocessor is the 80386 which is sold by Intel.

While the present invention is described hereinafter with particular reference to the system block diagram of Fig. 2, it is to be understood at the outset of the description which follows, it is contemplated that the apparatus and methods in accordance with the present invention may be used with other hardware configurations of the planar board. For example, the system processor could be an Intel 80286 or 80486 microprocessor.

Accessible by the processor is a planar identification number (planar ID). The planar ID is unique to the planar and identifies the type of planar being used. For example, the planar ID can be hardwired to be read through an I/O port of the system/processor 26 by using switches.

The local bus 28 is further connected through a bus controller 34 to a read only memory (ROM) 36 on the planar 24.

An additional nonvolatile memory (NVRAM) 58 is connected to the microprocessor 26 through a serial/parallel port interface 40 which is further connected to bus controller 34. The nonvolatile memory can be CMOS with battery backup to retain information whenever power is removed from the system. Since the ROM is normally resident on the planar, model and submodel values stored in ROM are used to identify the system processor and the system planar I/O configuration respectively. Thus these values will physically identify the processor and planar I/O configuration. The NVRAM is used to store system configuration data. That is, the NVRAM will contain values which describe the present configuration of the system. For example, NVRAM contains information describing the capacity of a fixed disk or diskette, the type of display, the amount of memory, time, date, etc. Additionally, the model and submodel values stored in ROM are copied to NVRAM whenever special configuration program, such as SET configuration", is executed. The purpose of the SET configuration program is to store values characterizing the configuration of the system in NVRAM. Thus for a system that is configured properly, the model and submodel values in NVRAM will be equal respectively to the model and submodel values stored in ROM. If these values are not equal, this indicates that the configuration of the system has been modified. Reference is made to Fig. 6D, where this feature in combination with loading BIOS is explained in greater detail.

Continuing, our discussion with reference to Fig. 2, the bus controller 34 is further coupled to I/O slots 18, a signal producing means such as a switch, the serial/parallel interface 40 and peripheral controller 42 by an I/O planar bus 43. The peripheral controller 42 is further connected to a keyboard 44, mouse 46, diagnostic panel 47, and diskette controller 64. Beside the NVRAfl4 58, the serial/parallel interface 40 is further connected to a serial port 48 and parallel port 50 to input/output information to a printer, hard copy device, etc. As is well known in the art, the local bus 28 can also be connected to a cache controller 52, a cache memory 68, a co-processor 54, and a DMA controller 56.

The signal producing means provides signals to the processor 26 depending upon the position of the switch. For instance, the switch is positioned to generate a signal on line 3 for effecting a diskette recovery mode in the system. Similarly, switch 29 is positioned to generate a signal on line 35 for a diskette priority mode. It is also understood that the switch can be replaced with hardware jumpers for effecting said signals. Additionally, as will be explained later, lines 33 and 35 can be combined into a single line for effecting a special function of the priority and recovery modes. It is also noted that switch 29 can be positioned so as not to produce either signal on line 33 or line 35.

The system processor 26 controls its internal operation as well as interfacing with other elements of the personal computer system 10. For example, system processor 26 is shown connected to a small computer system interface (SCSI) I/O card 60 which is further connected to a DASD, such as a fixed disk drive 62. It is to be understood that other than a SCSI disk drive/adapter can be used as a fixed disk in accordance with the

present invention. In addition to the fixed disk 62, the system processor 26 can be interfaced to the diskette controller 64 which controls a diskette drive 66. With respect to terminology, it is also to be understood that the term "fixed disk drive" describes fixed disk drive 62 while the term "floppy" also describes diskette drive 66.

Previous to the present invention, ROM 36 could include all of the BIOS code which interfaced the operating system to the hardware peripherals. According to one aspect of the present invention, however, ROM 36 is adapted to store only a portion of BIOS. This portion, when executed by the system processor 26, inputs from either the fixed disk 62 or diskette 66 a second or remaining portion of BIOS, hereinafter also referred to as a BIOS image. This BIOS image supersedes the first BIOS portion and being an integral part of the system, must be resident in main memory such as RAM 32. The first portion of BIOS (ROM-BIOS) as stored in ROM 36 will be explained generally with respect to Figs. 3-4 and in detail with respect to Figs. 6A-D. The second portion of BIOS (BIOS image) will be explained with respect to Fig. 5, and the loading of the BIOS image with respect to Fig. 7. Another benefit from loading a BIOS image from a DASD is the ability to load BIOS directly into the system processor's RAM 32. Since accessing RAM is much faster than accessing ROM, a significant improvement in the processing speed of the computer system is achieved.

The explanation will now proceed to the operation of the BIOS in ROM 36 and to the operation of loading the BIOS image from either the fixed disk or diskette to succeed the first portion of BIOS. In general, ROM-BIOS prechecks the system and loads a BIOS master boot record into RAM. The master boot record includes a data segment having validation information and a code segment having executable code. The executable code uses the data information to validate hardware compatibility and system conf iguration. After testing for hardware compatibility and proper system configuration, the executable code loads the BIOS image into RAM. The BIOS image succeeds ROM BIOS and loads the operating system to begin operation of the machine. For purposes of clarity, the executable code segment of the master boot record will be referred to as MBR code while the data segment will be referred to as MBR data.

Referring to Fig. 3 there is a memory map showing the different code modules which comprise ROM-BIOS. ROM-BIOS includes a power on self test (POST) stage I module 70, an Initial BIOS Load (IBL) Routine module 72, a Diskette module 74, a fixed disk drive module 76, a video module 78, a diagnostic- panel module 80, and hardware compatibility data 82. Briefly, POST Stage I 70 performs system pre-initialization and tests. The IBL routine 72 determines whether the BIOS image is to be loaded from disk or diskette, checks compatibility and loads the master boot record. Diskette module 74 provides input/output functions for a diskette drive. Hardfile module 76 controls I/O to a fixed disk or the like. Video module 78 controls output functions to a video I/O controller which is further connected to a video display. Diagnostic panel module 80 provides control to a diagnostic display device for the system. The hardware compatibility data 82 includes such values as a system model and submodel values. These values are described later with respect to Fig. 5.

Referring now to Fig. 4, there is shown a process overview for loading a BIOS image into the system from either the fixed disk or the diskette. When the system is powered up or reset, the system processor is vectored to the entry point of POST Stage I, step 100. POST Stage I initializes the system and tests only those system functions needed to load BIOS image from the selected DASD, step 102. In particular, POST Stage I initializes the processor/planar functions, diagnostic panel, memory subsystem, interrupt controllers, timers, DMA subsystem, fixed disk BIOS routine (Hardfile module 76), and diskette BIOS routine (Diskette module 74), if necessary.

After POST Stage I pre-initializes the system, POST Stage I vectors the system processor to the Initial BIOS Load (IBL) routine included in the Initial BIOS Load module 72. The IBL routine first, selects the media (disk or diskette) for loading the BIOS image; and second, loads the master boot record from the selected media into RAM, step 104. The master boot record includes the MBR data and the MBR code. The MBR data is used for verification purposes and the MBR code is executed to load in the BIOS image. A detailed description of the operation of the IBL routine is presented with respect to Figs. 6A-D.

With continuing reference to Fig. 4, after the IBL routine loads the master boot record into RAM, the system processor is vectored to the starting address of the MBR code to begin execution, step 106. The MBR code performs a series of validity tests to determine the authenticity of the BIOS image and to verify the configuration of the system. For a better understanding of the operation of the MBR code, attention is directed to Fig. 7 of the drawings wherein the MBR code is described in greater detail.

On the basis of these validity tests, the MBR code loads the BIOS image into RAM and transfers control to the newly loaded BIOS image in main memory, step 108. In particular, the BIOS image is loaded into the RAM address space previously occupied by ROM-BIOS. That is if ROM BIOS is addressed from E0000H through FFFFFH, then the BIOS image is loaded into this RAM address space, thus superceding ROM-BIOS. Control is then transferred to POST Stage II which is included in the newly loaded BIOS image thus abandoning RON-BIOS. POST Stage II, now in RAM, initializes and tests the remaining system in order to load the operating system boot, step 110. After the system is initialized and tested, Stage II POST transfers control to the operating system boot to load the operating system, steps 112-114.

For clarity, it is appropriate at this point to illustrate a representation for the format of the master boot record. Referring to Fig. 5, there is shown the master boot record. The boot record includes the executable code segment 120 and data segments 122-138. The MBR code 120 includes DASD dependent code responsible for verifying the identity of the ROM-BIOS, checking that the IBL boot record is compatible with the system, verifying the system configuration, and loading the BIOS image from the selected DASD (disk or diskette). The data segments 122-138 include information used to define the media, identify and verify the master boot record, locate the BIOS image, and load the BIOS image.

The master boot record is identified by a boot record signature 122. The boot record signature 122 can be a unique bit pattern, such as a character string "ABC" in the first three bytes of the record. The integrity of the master boot record is tested by a checksum value 132 which is compared to a computed checksum value when the boot record is loaded. The data segments further include at least one compatible planar ID value 134, compatible model and submodel values 136. The master boot record's planar ID value defines which planar that the master boot record is valid for. Similarly, the master boot record's model and submodel values define the processor and planar I/O configuration respectively that the master boot record is valid for. It is noted that the boot record's signature and checksum identify a valid master boot record, while the boot record's planar ID, boot record's model and boot record's submodel comparisons are used to identify a boot record compatible with the system and to determine if the system configuration is valid. Another value, boot record pattern 124 is used to determine the validity of the ROM-BIOS. The boot record pattern 124 is compared to a corresponding pattern value stored in ROM. If the values match this indicates that a valid ROM-BIOS has initiated the load of a BIOS image from the selected media.

The following description further describes in greater detail each of the values in the master boot record and their functions: MBR Identifier (122): The first three bytes of the IBL boot record can consist of characters, such as "ABC". This signature is used to identify a boot record. MBR Code Segment (120): This code verif ies the compatibility of the boot record with the planar and processor by comparing corresponding Planar id and model/submodel values. If these values match, it will load the BIOS image from the chosen media to system RAM. If the system image (BIOS image loaded into memory) checksum is valid and no media load errors occur, the MBR code will transfer control to the POST Stage II routine of the system image NBR Pattern (124): The first field of the IBL boot record data segment contains a pattern, such as a character string "ROM-BIOS 1989". This string is used to validate the ROM-BIOS by comparing the Boot Pattern value to the corresponding value stored in ROM (ROM-Pattern). MBR Version Date (126): The master boot record includes a version date for use by an update utility. System Partition Pointer (128): The data segment contains a media Pointer fro the beginning of the media system partition area for use by Stage II POST. On an IBL diskette, the pointer is in track-head-sector format; on disk the pointer is in Relative Block Address (RBA) format. System Partition Type (130): The system partition type indicates the structure of the media system partition. There are three types of system partition structures - full, minimal and not present. The full system partition contains the setup utility and diagnostics in addition to the BIOS image and master boot record. The minimal system partition contains just the BIOS image and master boot record. It may occur where a system does not have access to a fixed disk drive having an IBL image, in this circumstance the system partition type indicates "not present". In this instance, IBL will occur from the diskette. These three system partition types allow flexibility in how much space the system partition takes up on the media. Checksum value (132): The checksum value of the data segment is initialized to generate a valid checksum for the record length value (1.5k bytes) of the master boot record code. MBR Planar ID Value (134): The data segment includes a value, such as a string of words defining compatible planar IDs. Each word is made up of a 16 bit planar ID and the string is terminated by word value of zero. If a system's planar ID matches the planar ID value in the master boot record, such as one of the words in the string, the IBL media image is compatible with the system planar. If the system's planar ID does not match any word in the string, the IBL media image is not compatible with the system planar. MBR model and submodel values (136): The data segment includes values, such as a string of words defining compatible processors. Each word is made up of a model and submodel value and the string is terminated by a word value of zero. If a system's model and submodel value (stored in ROM) match one of the words in the string, the IBL media image is compatible with the system processor. If the ROM model and ROM submodel values do not match any word in the string, the IBL media image is not compatible with the system processor. MBR Map length (138): The IBL map length is initialized to the number of media image blocks. In other words, if the BIOS image is broken into four blocks, the map length will be four indicating four block pointer/length fields. Usually this length is set to bne, since the media image is one contiguous 128k block. MBR Media Sector Size (138): This word value is initialized to the media sector size in bytes per sector. Media image block pointer (138): The media image block pointer locates a system image block on the media. Normally, there is only one pointer since the media image is stored as one contiguous block. On an IBL diskette, the pointers are in track-head-sector format; on disk the pointers are relative block address format. Media image block length (138): The media image block length indicates the size (in sectors) of the block located

at the corresponding image block pointer. In the case of a 128k contiguous media image, which includes space for BASIC, this field is set to 256, indicating that the BIOS image block takes up 256 sectors (512 bytes/sector) starting at the media image block pointer location.

Referring now to Figs. 6A-D, there is shown a detailed flow chart of the operation of the IBL routine. Under normal circumstances, the IBL routine loads the master boot record from the system fixed disk into RAM at a specific address and then vectors the system processor to begin executing the code segment of the master boot record. The IBL routine also contains provisions for a diskette priority mode and recovery mode in which the master boot record is loaded from diskette. In the priority mode, the master boot record is loaded directly from diskette before attempting to load from the fixed disk. The purpose of the priority mode is to bypass the error checking procedure of the disk BIOS load process. The diskette BIOS load process does not include the validity checks that are used in the fixed disk BIOS load process. This permits system updates to be loaded from diskette into the system. For example, if a new processor is added to the system, a new BIOS image is required. Since a different processor will cause a validity error when loading from fixed disk, the IBL routine provides the ability to bypass these tests by loading the BIOS image from diskette. Thus the new BIOS image, included on diskette, can be given to the user to update the BIOS image on the fixed disk.

The recovery mode permits the system to bypass testing a password stored in NVRAM. The purpose of the password is to prevent unauthorized loading from diskette, however, a recovery mode is included within the system to allow a customer engineer or the like to load from diskette for diagnostic testing. It is noted that the priority mode and recovery mode can be activated by a single switch to accomplish the same results. In this configuration, if the diskette media is not available during the priority mode, the user is given extra time (to the recovery mode check) to insert the diskette to accomplish a BIOS load from diskette. This appears to the user as a priority mode load, but in actuality is a recovery mode operation. If the IBL routine is not able to load the master boot record from either fixed disk or diskette, an error message is generated and the system is halted.

Referring now to Fig. 6A, the switching means is tested to detect activation of the priority mode, step 151. If the priority mode is activated, the diskette subsystem is initialized, step 153. Assume for purposes of illustration that the fixed disk is configured for Drive C of the personal computer system. Similarly, assume Drive A is designated as the diskette drive. The IBL routine then examines Drive A to determine whether it contains IBL media or not, step 155. Attention is directed to Fig. 6C which describes in detail this process. If Drive A does not contain IBL media, the system attempts to load IBL media from the fixed disk, step 150. Referring back to step 155, if Drive A does include IBL media, the master boot record is loaded into RAM, step 160.

Referring back to step 151, if the priority mode is not activated, the fixed disk subsystem is initialized, and the IBL routine then examines Drive C to determine whether it contains IBL media, step 152. Attention is directed to Fig. 6B which describes in detail this process. If Drive C does not contain IBL media, an error is reported step 154.

Referring back to step 152, if Drive C does contain IBL media, the IBL routine starts reading from the fixed disk at the last three sectors and continues reading, decrementing the media pointer, for 99 sectors or until a valid master boot record is found. If a master boot record is found, it is checked for system planar and processor compatibility, step 156. If no master boot record is found on the last 99 sectors of the fixed disk (primary fixed disk drive) or if it is not planar or processor compatible, then an error is reported, step 158.

Referring back to step 156, if a master boot record is found, a series of validity checks are performed to determine if the master boot record is compatible with the computer system. Additionally, the configuration of the system is checked. Attention is directed to Fig. 6D which discloses this process in greater detail. If the boot record is compatible with the planar ID, model and submodel, and if furthermore the system configuration has not changed the master boot record is loaded and the code segment of the master boot record is executed, step 160.

Referring back to steps 154 and 158, if an error occurs in loading the master boot record from the fixed disk or if a fixed disk is not available, the system determines whether the recovery mode is activated, step 157. If the recovery mode is activated a test for valid password in NVRAM is bypassed and the diskette subsystem is initialized, step 166. If the recovery mode is not activated, the system determines if a valid password is included in NVRAM, step 162. This password determines whether the BIOS image can be loaded from diskette by an unauthorized user. Note that the password will only exist in NVRAM when a user has purposely done so. If a password is installed in NVRAM, all users are prevented from loading the BIOS image from diskette, (other than a customer engineer) step 164. This ensures the integrity of the operation of the system by causing the system to be loaded only with the correct configuration of the BIOS image on the fixed disk. If a password exists, the diskette subsystem can not be accessed and the system halts, step 172. This password can be loaded into NVRAM during system configuration, such as when the SETUP program is executed. The password can take the form of a string of characters stored in NVRAM.

Referring back to step 162, if a valid password in NVRAM is not present, thus allowing BIOS image to be loaded from diskette, the IBL routine initializes the diskette subsystem, step 166. The IBL routine then determines if Drive A includes the IBL media on a diskette, step 168. If Drive A does not include IBL media, an error

is generated to notify the user that an invalid diskette has been inserted in the drive, step 170. The system then halts, step 172. Attention is directed to Fig. 6C for a more detailed discussion of step 168.

Referring back to step 168, after Drive A is checked for IBL media, the master boot record is loaded into RAM and the code segment included in the master boot record is executed, step 160. It is important to note that for diskette the IBL routine does not include the validity checks that are used with the fixed disk system. As stated before, the reason for the absence of the validity checks is to allow future modifications in loading a new IBL image from diskette.

To recapitulate, the priority mode is tested for first. In the priority mode, if the diskette drive does not contain IBL media the system tests the fixed disk. If the fixed disk includes IBL media, the master boot record is checked for compatibility with the system through matching the system planar ID and processor model/submodel values to the boot record values. For disk, this check is done first in the IBL routine 72 and then done again in the IBL boot record. The first check (in the IBL routine) is done to make sure the boot record is compatible with the system; the second check (in the boot record) is done to ensure a compatible ROM passed control to the boot record. Notice that the check done in the disk boot record will never fail for a compatible ROM since the IBL routine will have already checked the compatibility. In contrast, the compatibility check is not done for diskette. The planar/processor compatibility is checked only during diskette boot record execution. This method allows future modifications in loading a new BIOS image from a diskette. If the disk does not contain IBL media, the recovery mode and/or password are tested. If the recovery mode is activated or the password is absent, IBL media is loaded from diskette. If the recovery mode is not activated and the password is present, an error is generated and the system halts.

In view of the description of the IBL routine of Fig. 6A, the explanation will now proceed to a comprehensive and full understanding of the validity tests discussed above. Referring to Fig. 6B, there is shown a detailed flowchart of step 152 of Fig. 6A, to determine if a valid master boot record is on drive C. The process begins by obtaining the drive parameters to enable the IBL routine to access drive C, step 200. An IBL load location is set to the last three sectors from the disk (the last three sectors normally contain the master boot record), step 202. A load count indicating the number of attempts to read a master boot record from disk is set to 1, step 204. Three sectors are read from disk at the IBL load location, step 206. Any disk drive errors are detected and if a disk drive read error occurs it is reported, steps 208-210. The process then returns with an error indication, steps 212-214.

Referring back to step 208, if no drive error occurs, the disk record is scanned for the master boot record signature, step 216. The boot record signature, such as the characters "ABC", are compared to the first three bytes of the disk record. If the disk record does have a valid boot record signature (characters "ABC") and the checksum computed from the disk record loaded into memory equals the boot record checksum, the disk record is indicated as being a valid boot record with no errors, step 218. The process then returns, step 214.

Referring back to step 216, if the boot record signature or checksum is invalid, the load count is incremented by 1, step 220. The load count is then compared to a predetermined constant such as 99, step 222. If 99 attempts to read a boot record have resulted in failure, an error is indicated and the process returns, steps 294, 212 and 214. If less than 99 attempts to read a boot record have occurred, the IBL load location is decremented by one and three new sectors are read from the new load location, steps 226 and 206. Thus if a valid IBL boot record cannot be loaded from the last 99 sectors (equivalent to 33 copies) then an error condition is set and control returns to the IBL routine.

Referring now to Fig. 6C, there is shown a detailed flow diagram for loading the master boot record from diskette on drive A. First, the diskette drive parameters to access drive A are retrieved, step 230. The IBL load location is set to the last 3 sectors on diskette (cylinder, head and sector format), step 232. The last 3 sectors are read, step 234. If a diskette drive error is detected, an error indicated, steps 236-238. An error condition is set and control is returned to the IBL routine, steps 240-242.

Referring back to step 236, if no drive error is detected, the diskette record is checked for boot record signature and the checksum is calculated, step 244. If the boot record signature is missing or checksum is invalid, an error is indicated and control returned to the IBL routine, steps 244, 246, 240 and 242. If a valid boot record signature and valid checksum are detected an indication is set and control is returned to the IBL routine, steps 248 and 242. It is noted that in a diskette load, the IBL routine does not search through the media as in the fixed disk load. Therefore, in a diskette load, the IBL media must be stored in a specific location of the diskette.

Finally, Fig. 6D shows how the IBL routines tests for system planar and processor compatibility and for a proper system configuration. The master boot record is checked for compatibility with the system planar by comparing the boot record planar ID value to the system planar ID read by the system processor, step 260. If the system planar ID does not match the boot record planar ID value, this indicates this master boot record is not compatible with this planar. An error is indicated and control return to the IBL routine, steps 262, 264, and 266.

If the master boot record is compatible with the planar, the master boot record is checked for compatibility with the processor, step 268. The boot record model value and submodel value are compared to the model value and submodel value stored in ROM respectively. A

mismatch indicates a new processor has probably been inserted and this boot record is not compatible with the new processor. An error is indicated and control returned to the IBL routine, steps 270, 264 and 266. If the master boot record is compatible with the planar and processor, the processor checks to determine if NVRAM contains reliable information, step 272. If NVRAM is unreliable, an error is indicated and control returned to the IBL routine, steps 274 and 266. If NVRAM is reliable, the system conf iguration is checked, step 276. A change in system configuration is indicated if the model and submodel values stored in NVRAM do not match the model and submodel values stored in ROM. Note that this last comparison will only indicate a configuration error. If a configuration error is indicated, an error is generated for the user. This error notifies the user that the configuration of the system has changed since the last time SET conf iguration was run. The user is notified of the changed configuration and control passed back to the IBL routine steps 278, 264, and 266. This error is not fatal itself, but notifies the user that SET configuration (configuration program) must be executed. Referring back to step 276, if the system model/submodel values match, an indication of compatibility is set and the routine returns, step 276, 274 and 266. Thus, the compatibility between the master boot record and the system are tested along with determining if the system configuration has been modified.

After the IBL routine loads the master boot record into RAM, it transfers control to the MBR code starting address. Referring to Fig. 7, the executable code segment of the master boot record first verifies the boot record pattern to the ROM pattern, step 300. If the pattern in the master boot record does not match the pattern in ROM, an error is generated and the system halts, steps 302 and 305. The check for equality between ROM and boot record patterns ensures that the master boot record loaded from either the disk or diskette is compatible with the ROM on the planar board. Referring back to step 300, if the pattern in ROM matches the pattern in the boot record, the MBR code compares the system planar ID value, model and submodel value against the corresponding master boot record values, step 304. This process was discussed in greater detail with respect to Fig. 6D. If the values don't match, the master boot record is not compatible with the system planar and processor, or the system conf iguration has changed, and an error is generated, step 306. The system will halt when the IBL record is incompatible with the planar, model, or submodel value, step 305.

Referring back to step 304, if the system planar ID value, model and submodel values match the corresponding master boot record values, the MBR code loads the BIOS image from the selected media into the system RAM, step 308. If a media load error occurs in reading the data, step 310, an error is generated and the system halts, step 312 and 305. Referring back to step 310, if no media load error occurs, a checksum is calculated for the BIOS image in memory, step 314. If the checksum is invalid an error is generated and the system halts, step 318 and 305. Referring back to step 316, if the checksum is valid, the system partition pointers are saved, step 320, and the system processor is vectored to POST Stage II to begin loading the system, step 322.

Thus, there has been shown a method and apparatus for loading BIOS from a diskette drive in a personal computer system normally having a fixed disk drive. BIOS is loaded depending upon the position of a switch. In a priority position, BIOS is loaded from diskette before testing the fixed disk drive. In a recovery mode, BIOS is loaded from diskette after testing the fixed disk drive. In a default mode, BIOS is loaded after testing the fixed disk and for the presence of a purposely set password.

While the invention has been illustrated in connection with a preferred embodiment, it should be understood that many variations will occur to those of ordinary skill in the art, and that the scope of the invention is defined only by the claims appended hereto.

Claims

1. A computer system (10) comprising:
a system processor (26), a read only memory (36), a random access memory (32), a diskette drive (66), and a fixed disk drive (62);

characterised in that the system (10) further comprises:

a first portion of BIOS resident in the read only memory (36), the first portion of BIOS initializing the system (10);

a master boot record included in the diskette media, the master boot record including an executable code segment (120);

means for producing a signal, the signal being representative of loading BIOS from the diskette drive (66);

means being responsive to the signal for determining if BIOS is to be loaded from the diskette drive (66) or fixed disk drive (62), wherein if BIOS is to be loaded from the diskette media, the first portion of BIOS initializes the diskette drive (66) to effect the loading of the master boot record into the random access memory (32);

a remaining portion of BIOS included in the diskette media, wherein the first portion of BIOS transfers control to the executable code segment (120) of the master boot record in order

to effect the loading of the remaining portion of BIOS into the random access memory (32).

2. A computer system (10) as claimed in claim 1, wherein the signal producing means generates a priority signal representative of a mode wherein the master boot record and remaining portion of BIOS are loaded into random access memory (32) before attempting to access the fixed disk drive (62).

3. A computer system (10) as claimed in claim 2, wherein the signal producing means generates a recovery signal representative of a mode wherein the master boot record and remaining portion of BIOS dire loaded into random access memory (32) after attempting to load BIOS from the fixed disk drive (62).

4. A computer system (10) as claimed in claim 1, further including a nonvolatile memory (50) electrically coupled to the system processor (26), wherein in the absence of the priority signal and the recovery signal the master boot record and the remaining portion of BIOS are loaded into random access memory (32) after attempting to load BIOS from the fixed disk drive (62) and detecting the absence of a password normally included in the nonvolatile memory (58).

5. A computer system (10) as claimed in claim 1, wherein the signal producing means comprises a switch (29) electrically connected to a system planar (24), the system planar (24) being electrically coupled to the system processor (26).

6. A computer system (10) as claimed in claim 5, wherein the switch comprises a hardwired jumper resident on the system planar (24).

7. A computer system (10) as claimed in claim 6, wherein the master boot record further includes a data segment (122-138) representative of a hardware configuration of the computer system (10) which is compatible with the master boot record, and further wherein the read only memory (36) includes data representing a hardware configuration of the system processor (26), wherein before the remaining portion of BIOS is loaded into random access memory (32), the executable code segment (120) compares the hardware configuration data from the master boot record with the hardware configuration data from the read only memory (36) to verify the master boot record is compatible with the system processor (26).

8. A computer system (10) as claimed in claim 7, wherein the data segment (122-138) of the master boot record includes a value representing the system planar (24) which is compatible with the master boot record and further wherein the system planar (24) further includes a means for uniquely identifying the system planar (24) in order to verify that the master boot record is compatible to the system planar (24).

9. A computer system (10) as claimed in claim 7, wherein the hardware configuration data on the master boot record includes a model and submodel values (136), wherein the model value identifies a system processor (26) which is compatible with the master boot record and the submodel value represent an I/O configuration of the system planar (24) which is compatible with the master boot record, and further wherein the read only memory (36) includes a corresponding model value identifying the system processor (26) and submodel value representing the I/O configuration of the system planar (24), wherein the model value and submodel value of the master boot record are compared to the corresponding model and submodel values of the read only memory (36) respectively, in order to verify that the master boot record is compatible with the system processor (26) and the I/O configuration of the system planar (24).

10. A computer system (10) as claimed in claim 1, wherein the master boot record includes a predetermined character code (122) in order to distinguish the master boot record from other records included on the diskette drive (66).

11. A computer system (10) as claimed in claim 1 wherein the signal producing means generates a recovery signal, the recovery signal being representative of a mode wherein the master boot record and the remaining portion of BIOS are loaded into random access memory (32) after attempting to load BIOS from the disk drive.

12. A computer system (10) as claimed in claim 1, wherein the executable code segment (120) generates a first error to indicate the master boot record is not compatible with the system hardware configuration.

13. A computer system (10) as claimed in claim 4 wherein the nonvolatile random access memory (58) includes data representing the system configuration, the data being updated when the configuration of the system (10) is changed, wherein the executable code segment (120) compares the data in the nonvolatile random access memory (58) to corresponding data in the read only memory (36) to determine if the configuration of the system (10) has changed.

14. A computer system (10) as claimed in claim 13, wherein the executable code segment (120) generates a second error to indicate that the system configuration has changed.

15. A computer system (10) as claimed in claim 10, wherein the predetermined code is prefatory to the code segment of the master boot record.

16. A computer system (10) as claimed in claim 10, wherein the master boot record includes a checksum value (132) to verify the validity of the master boot record when loaded into the random access memory (32).

17. A computer system (10) as claimed in claim 1, wherein the remaining portion of BIOS includes a checksum value (132) to verify the validity of the remaining portion of BIOS when loaded into the random access memory (32).

18. A computer system (10) as claimed in claim 1, wherein the master boot record includes a predetermined pattern, and further wherein the read only memory (36) includes a corresponding predetermined pattern in order to verify that the first portion of BIOS is included within a predefined read only memory (36).

19. A computer system (10) as claimed in claim 14, wherein the executable code segment (120) generates a third error to indicate that the read only memory (36) is not compatible with the master boot record.

20. A method for loading BIOS from a diskette drive (66) in a personal computer system (10) normally connected to a fixed disk drive (62), the system (10) having a system processor (26) electrically coupled to a system planar (24), the planar (24) further being electrically coupled to a read only memory (36) random access memory (32) and nonvolatile memory (58), the method comprising the steps of:

(a) initializing the system (10) with a first portion of BIOS resident in the read only memory (36);

(b) determining if the fixed disk drive (62) is present;

(c) in the absence of the fixed disk drive (62), searching the nonvolatile memory (58) for a password;

(d) upon detecting the absence of the password, initializing with the first portion of BIOS the diskette drive (66) having a master boot record and the remaining BIOS;

(e) loading with the first portion of BIOS the master boot record into random access memory (32), the master boot record including a data segment (122-138) and an executable code segment (120), the data segment (122-138) having data representing the hardware configuration of the system (10) for which the remaining BIOS is compatible;

(f) verifying the compatibility of the master boot record with the system hardware by comparing from the data segment (122-138) the data representing the hardware configuration record to corresponding data stored in read only memory (36);

(g) executing the code segment of the master boot record to load the remaining BIOS code into main memory; and

(h) passing control to the remaining BIOS code once it is loaded into main memory.

Patentansprüche

1. Rechnersystem (10), umfassend:

einen Systemprozessor (26), einen Nur-Lese-Speicher (36), einen Schreib-Lese-Speicher (32), ein Diskettenlaufweik (66) und ein Festplattenlaufwerk (62);

dadurch gekennzeichnet, daß das System (10) des weiteren umfaßt:

einen ersten Teil des BIOS, der im Nur-Lese-Speicher (36) resident ist, wobei der erste Teil des BIOS das System (10) initialisiert;

einen Hauptladesatz, der in dem Diskettenmedium enthalten ist, wobei der Hauptladesatz ein ausführbares Codesegment (120) umfaßt;

Mittel zur Erzeugung eines Signals, wobei das Signal für das Laden des BIOS vom Diskettenlaufwerk (66) steht;

Mittel, die auf das Signal ansprechen und bestimmen, ob das BIOS von dem Diskettenlaufwerk (66) oder dem Festplattenlaufwerk (62) geladen werden soll, wobei, wenn das BIOS von dem Diskettenmedium geladen werden soll, der erste Teil des BIOS das Diskettenlaufwerk (66) initialisiert, um das Laden des Hauptladesatzes in den Schreib-Lese-Speicher (32) zu veranlassen;

einen übrigen Teil des BIOS, der in dem Diskettenmedium enthalten ist, wobei der erste Teil des BIOS die Steuerung des Hauptladesatzes an das ausführbare Codesegment (120) überträgt, um das Laden des übrigen Teils des BIOS in den Schreib-Lese-Speicher (32) zu veranlassen.

2. Rechnersystem (10) nach Anspruch 1, wobei das Signalerzeugungsmittel ein Prioritätssignal erzeugt, das für einen Modus steht, in dem der Hauptladesatz und der übrige Teil des BIOS in den Schreib-Lese-Speicher (32) geladen werden, bevor versucht wird, auf das Festplattenlaufwerk (62) zuzugreifen.

3. Rechnersystem (10) nach Anspruch 2, wobei das Signalerzeugungsmittel ein Wiederherstellungssignal erzeugt, das für einen Modus steht, wobei der Hauptladesatz und der übrige Teil des BIOS in den Schreib-Lese-Speicher (32) geladen werden, nachdem versucht wurde, das BIOS von dem Festplattenlaufwerk (62) zu laden.

4. Rechnersystem (10) nach Anspruch 1, das ferner einen permanenten Speicher (58) umfaßt, der mit dem Systemprozessor (26) elektrisch verbunden ist, wobei beim Fehlen des Prioriätssignals und des Wiederherstellungssignals der Hauptladesatz und der übrige Teil des BIOS in den Schreib-Lese-Speicher (32) geladen werden, nachdem versucht wurde, das BIOS von dem Festplattenlaufwerk (62) zu laden, und nachdem das Fehlen eines Kennworts erkannt wurde, das normalerweise in dem permanenten Speicher (58) enthalten ist.

5. Rechnersystem (10) nach Anspruch 1, wobei das Signalerzeugungsmittel einen Schalter (29) umfaßt, der mit einer Systemplatine (24) elektrisch verbunden ist, wobei die Systemplatine (24) mit dem Systemprozessor (26) elektrisch verbunden ist.

6. Rechnersystem (10) nach Anspruch 5, wobei der Schalter eine festverdrahtete Brücke umfaßt, die sich auf der Systemplatine (24) befindet.

7. Rechnersystem (10) nach Anspruch 6, wobei der Hauptladesatz ferner ein Datensegment (122 bis 138) umfaßt, das für eine Hardwarekonfiguration des Rechnersystems (10) steht, die mit dem Hauptladesatz kompatibel ist, sowie ferner, wobei der Nur-Lese-Speicher (36) Daten enthält, die für eine Hardwarekonfiguration des Systemprozessors (26) stehen, wobei vor dem Laden des übrigen Teils des BIOS in den Schreib-Lese-Speicher (32) das ausführbare Codesegment (120) die Hardwarekonfigurationsdaten aus dem Hauptladesatz mit den Hardwarekonfigurationsdaten aus dem Nur-Lese-Speicher (36) vergleicht, um zu prüfen, ob der Hauptladesatz mit dem Systemprozessor (26) kompatibel ist.

8. Rechnersystem (10) nach Anspruch 7, wobei das Datensegment (122-138) des Hauptladesatzes einen Wert umfaßt, der für die Systemplatine (24) steht und mit dem Hauptladesatz kompatibel ist, sowie ferner, wobei die Systemplatine (24) des weiteren Mittel für die eindeutige Identifizierung der Systemplatine (24) umfaßt, um zu prüfen, ob der Hauptladesatz mit der Systemplatine (24) kompatibel ist.

9. Rechnersystem (10) nach Anspruch 7, wobei die Hardwarekonfigurationsdaten des Hauptladesatzes Modell- und Teilmodellwerte (136) umfassen, wobei der Modellwert einen Systemprozessor (26) angibt, der mit dem Hauptladesatz kompatibel ist, und wobei der Teilmodellwert eine E/A-Konfiguration der Systemplatine (24) angibt, die mit dem Hauptladesatz kompatibel ist, und wobei ferner der Nur-Lese-Speicher (36) einen entsprechenden Modellwert für den Systemprozessor (26) und einen Teilmodellwert für die E/A-Konfiguration der Systemplatine (24) enthält, wobei der Modellwert und der Teilmodellwert des Hauptladesatzes mit dem entsprechenden Modell- bzw. Teilmodellwert des Nur-Lese-Speichers (36) verglichen werden, um zu prüfen, ob der Hauptladesatz mit dem Systemprozessor (26) und der E/A-Konfiguration der Systemplatine (24) kompatibel ist.

10. Rechnersystem (10) nach Anspruch 1, wobei der Hauptladesatz einen vorbestimmten Zeichencode (122) umfaßt, um den Hauptladesatz von anderen Sätzen des Diskettenlaufwerks (66) zu unterscheiden.

11. Rechnersystem (10) nach Anspruch 1, wobei das Signalerzeugungsmittel ein Wiederherstellungssignal erzeugt, wobei das Wiederherstellungssignal für einen Modus steht, wobei der Hauptladesatz und der übrige Teil des BIOS in den Schreib-Lese-Speicher (32) geladen werden, nachdem versucht wurde, das BIOS von dem Diskettenlaufwerk zu laden.

12. Rechnersystem (10) nach Anspruch 1, wobei das ausführbare Codesegment (120) einen ersten Fehler erzeugt, um anzuzeigen, daß der Hauptladesatz nicht mit der System-Hardwarekonfiguration kompatibel ist.

13. Rechnersystem (10) nach Anspruch 4, wobei der permanente Schreib-Lese-Speicher (58) Daten für die Systemkonfiguration umfaßt, wobei die Daten

aktualisiert werden, wenn die Konfiguration des Systems (10) geändert wird, wobei das ausführbare Codesegment (120) die Daten in dem permanenten Schreib-Lese-Speicher (58) mit den entsprechenden Daten in dem Nur-Lese-Speicher (36) vergleicht, um zu ermitteln, ob die Konfiguration des Systems (10) geändert wurde.

14. Rechnersystem (10) nach Anspruch 13, wobei das ausführbare Codesegment (120) einen zweiten Fehler erzeugt, um anzuzeigen, daß die Systemkonfiguration geändert wurde.

15. Rechnersystem (10) nach Anspruch 10, wobei der vorbestimmte Code als Einleitung für das Codesegment des Hauptladesatzes dient.

16. Rechnersystem (10) nach Anspruch 10, wobei der Hauptladesatz einen Kontrollsummenwert (132) umfaßt, um die Gültigkeit des Hauptladesatzes beim Laden in den Schreib-Lese-Speicher (32) zu überprüfen.

17. Rechnersystem (10) nach Anspruch 1, wobei der übrige Teil des BIOS einen Kontrollsummenwert (132) umfaßt, um die Gültigkeit des übrigen Teils des BIOS beim Laden in den Schreib-Lese-Speicher (32) zu überprüfen.

18. Rechnersystem (10) nach Anspruch 1, wobei der Hauptladesatz eine vorbestimmte Struktur umfaßt, und wobei der Nur-Lese-Speicher (36) ferner eine entsprechende vorbestimmte Struktur enthält, um zu prüfen, ob der erste Teil des BIOS in einem vorbestimmten Nur-Lese-Speicher (36) enthalten ist.

19. Rechnersystem (10) nach Anspruch 14, wobei das ausführbare Codesegment (120) einen dritten Fehler erzeugt, um anzuzeigen, daß der Nur-Lese-Speicher (36) nicht mit dem Hauptladesatz kompatibel ist.

20. Verfahren zum Laden des BIOS von einem Diskettenlaufwerk (66) in einem Personalcomputer-System (10), das normalerweise mit einem Festplattenlaufwerk (62) verbunden ist, wobei das System (10) einen Systemprozessor (26) umfaßt, der mit einer Systemplatine (24) elektrisch verbunden ist, und wobei die Platine (24) des weiteren mit einem Nur-Lese-Speicher (36), einem Schreib-Lese-Speicher (32) und einem permanenten Speicher (58) elektrisch verbunden ist, wobei das Verfahren die folgenden Schritte umfaßt:

(a) Initialisieren des Systems (10) mit einem ersten Teil des BIOS, der im Nur-Lese-Speicher (36) resident ist;

(b) Ermitteln, ob das Festplattenlaufwerk (62) vorhanden ist;

(c) beim Fehlen des Festplattenlaufwerks (62) Durchsuchen des permanenten Speichers (58) nach einem Kennwort;

(d) bei Erkennen des Fehlens des Kennworts Initialisieren des Diskettenlaufwerks (66), das über einen Hauptladesatz und das übrige BIOS verfügt, mit dem ersten Teil des BIOS;

(e) Laden des Hauptladesatzes in den Schreib-Lese-Speicher (32) mit dem ersten Teil des BIOS, wobei der Hauptladesatz ein Datensegment (122-138) und ein ausführbares Codesegment (120) umfaßt, wobei das Datensegment (122-138) über Daten verfügt, die für die Hardwarekonfiguration des Systems (10) stehen, mit dem das übrige BIOS kompatibel ist;

(f) Prüfen der Kompatibilität des Hauptladesatzes mit der Systemhardware, indem die Daten des Datensegments (122-138), die für den Hardwarekonfigurationssatz stehen, mit den entsprechenden Daten in dem Nur-Lese-Speicher (36) verglichen werden;

(g) Ausführen des Codesegments des Hauptladesatzes, um den übrigen BIOS-Code in den Hauptspeicher zu laden; und

(h) Übertragen der Steuerung an den übrigen BIOS-Code, nachdem er in den Hauptspeicher geladen wurde.

Revendications

1. Système d'ordinateur (10) comprenant :
un processeur de système (26), une mémoire morte (36), une mémoire vive (32), une unité de disquette (66), et une unité de disque fixe (62) ;

caractérisé en ce que le dit système (10) comprend en outre :

une première partie d'un Système Basique d'Entrée/Sortie (SBES) résidant dans la dite mémoire morte (36), la dite première partie du SBES initialisant le dit système (10) ;

un article de chargement du fichier permanent inclus dans le support de disquette, le dit article de chargement du fichier permanent comprenant un segment de code directement exploitable (120);

un dispositif pour produire un signal, le dit signal étant représentatif du chargement du dit SBES réalisé à partir de la dite unité de disquette (66) ;

un dispositif, sensible au dit signal, qui détermine si le dit SBES doit être chargé à partir de la dite unité de disquette (66) ou à partir de la dite unité de disque fixe (62) où, si le dit SBES doit être chargé à partir du dit support de disquette, la dite première partie du dit SBES initialise la dite première unité de disquette (66) pour effectuer le chargement du dit article de chargement du fichier permanent dans la dite mémoire vive (32) ;

une partie restante du SBES comprise dans le dit support de disquette, où la dite première partie du SBES transfère le contrôle au dit segment de code directement exploitable (120) du dit article de chargement du fichier permanent afin d'obtenir le chargement de la dite partie restante du dit SBES dans la dite mémoire vive (32).

2. Système d'ordinateur (10) tel que revendiqué dans la revendication 1, dans lequel le dit dispositif générateur de signaux produit un signal de priorité représentatif d'un mode dans lequel le dit article de chargement du fichier permanent et la dite partie restant du dit SBES sont chargés dans la dite mémoire vive (32) avant toute tentative pour accéder à la dite unité de disque fixe (62).

3. Système d'ordinateur (10) tel que revendiqué dans la revendication 2, dans lequel le dit dispositif générateur de signaux produit un signal de récupération représentatif d'un mode dans lequel le dit article de chargement du fichier permanent et la dite partie restante du dit SBES sont chargés dans la dite mémoire vive (32) après une tentative pour charger le dit SBES à partir de la dite unité de disque fixe (62).

4. Système d'ordinateur (10) tel que revendiqué dans la revendication 1, comprenant en outre une mémoire rémanente (58) reliée électriquement au dit processeur du système (26), où, en l'absence des dits signaux de priorité et de récupération, le dit article de chargement du fichier permanent et la dite partie restante du dit SBES sont chargés dans la dite mémoire vive (32) après une tentative pour charger le dit SBES à partir de la dite unité de disque fixe (62) et la détection de l'absence du mot de passe normalement inclus dans la dite mémoire rémanente (58).

5. Système d'ordinateur (10) tel que revendiqué dans la revendication 1, où le dit dispositif générateur de signaux comprend un aiguillage (29) connecté électriquement à un circuit planaire du système (24), le dit circuit planaire (24) étant relié électriquement au dit processeur du système (26).

6. Système d'ordinateur (10) tel que revendiqué dans la revendication 5, dans lequel le dit aiguillage comprend un câble de liaison situé sur le dit circuit planaire (24).

7. Système d'ordinateur (10) tel que revendiqué dans la revendication 6, dans lequel le dit article de chargement du fichier permanent comprend en outre un segment de données (122-138) représentatif d'une configuration du matériel pour le dit système d'ordinateur (10) qui est compatible avec le dit article de chargement du fichier permanent, et où, de plus, la dite mémoire morte (36) inclut des données représentant une configuration du matériel du dit processeur de système (26), où, avant que la dite partie restante du dit SBES soit chargée dans la dite mémoire vive (32), le dit segment de code directement exploitable (120) compare les données de la configuration du matériel provenant du dit article de chargement du fichier permanent avec les données de la configuration du matériel provenant de la dite mémoire morte (36) afin de vérifier que le dit article de chargement du fichier permanent est compatible avec le dit processeur du système (26).

8. Système d'ordinateur (10) tel que revendiqué dans la revendication 7, dans lequel le dit segment des données (122-138) du dit article de chargement du fichier permanent comprend une valeur représentant le circuit planaire (24) compatible avec le dit article de chargement du fichier permanent et où, de plus, le dit circuit planaire (24) comprend en outre un moyen pour identifier de manière unique le dit circuit planaire (24) afin de vérifier que le dit article de chargement du fichier permanent est compatible avec le dit circuit planaire (24).

9. Système d'ordinateur (10) tel que revendiqué dans la revendication 7, dans lequel les dites données de configuration du matériel présentes sur le dit article de chargement du fichier permanent comprennent des valeurs de modèle et de sous-modèle (136), où la dite valeur de modèle identifie un processeur de système (26) qui est compatible avec le dit article de chargement du fichier permanent et la dite valeur de sous-modèle représente une configuration d'entrée et de sortie du dit circuit planaire (24) compatible avec le dit article de chargement du fichier permanent, et où, de plus, la dite mémoire morte (36) comprend une valeur de modèle correspondant qui identifie le dit processeur du système (26) et une valeur de sous-modèle qui représente la con-

figuration d'entrée et de sortie du dit circuit planaire (24), où la dite valeur de modèle et la dite valeur de sous-modèle du dit article de chargement du fichier permanent sont comparées respectivement aux dites valeurs des dits modèle correspondant et sous-modèle provenant de la dite mémoire morte (36), afin de vérifier que le dit article de chargement du fichier permanent est compatible avec le dit processeur de système (26) et la dite configuration d'entrée et de sortie du circuit planaire (24) du système.

10. Système d'ordinateur (10) tel que revendiqué dans la revendication 1, dans lequel le dit article de chargement du fichier permanent comprend un code de caractères (122) déterminé au préalable afin de distinguer le dit article de chargement du fichier permanent des autres enregistrements présents sur la dite unité de disquette (66).

11. Système d'ordinateur (10) tel que revendiqué dans la revendication 1, dans lequel le dit dispositif générateur de signaux produit un signal de récupération, le dit signal de récupération étant représentatif d'un mode dans lequel le dit article de chargement du fichier permanent et la dite partie restante du dit SBES sont chargés dans la dite mémoire vive (32) après une tentative pour charger le dit SBES à partir de la dite unité de disque.

12. Système d'ordinateur (10) tel que revendiqué dans la revendication 1, dans lequel le dit segment de code exploitable directement (120) produit une première erreur pour indiquer que le dit article de chargement du fichier permanent n'est pas compatible avec la configuration du matériel du système.

13. Système d'ordinateur (10) tel que revendiqué dans la revendication 4, dans lequel la dite mémoire vive rémanente (58) inclut des données représentant la configuration du système, les données étant remises à jour quand la configuration du dit système (10) est modifiée, où le dit segment de code (120) exploitable directement compare les données comprises dans la dite mémoire vive rémanente (58) aux données correspondantes de la dite mémoire morte (36) pour déterminer si la configuration du dit système (10) a été modifiée.

14. Système d'ordinateur (10) tel que revendiqué dans la revendication 13, dans lequel le dit segment de code exploitable directement (120) produit une seconde erreur pour indiquer que la configuration du dit système a changé.

15. Système d'ordinateur (10) tel que revendiqué dans la revendication 10, dans lequel le dit code déterminé au préalable est préliminaire au dit segment de code du dit article de chargement du fichier permanent.

16. Système d'ordinateur (10) tel que revendiqué dans la revendication 10, dans lequel le dit article de chargement du fichier permanent comprend une valeur qui est un total de contrôle (132) servant à vérifier la validité du dit article de chargement du fichier permanent quand on le charge dans la dite mémoire vive (32).

17. Système d'ordinateur (10) tel que revendiqué dans la revendication 1, dans lequel la dite partie restante du dit SBES comprend un total de contrôle (132) qui sert à vérifier la validité de la dite partie restante du dit SBES quand on le charge dans la dite mémoire vive (32).

18. Système d'ordinateur (10) tel que revendiqué dans la revendication 1, dans lequel le dit enregistrement article de chargement du fichier permanent comprend un schéma déterminé au préalable, et où, en outre, la dite mémoire morte (36) inclut un schéma correspondant déterminé au préalable qui sert à vérifier que la dite première partie du dit SBES est comprise dans une mémoire morte (36) définie au préalable.

19. Système d'ordinateur (10) tel que revendiqué dans la revendication 14, dans lequel le dit segment de code exploitable directement produit une troisième erreur pour indiquer que la dite mémoire morte (36) n'est pas compatible avec le dit article de chargement du fichier permanent.

20. Procédé pour charger, à partir d'une unité de disquette (66), un Système Basique d'Entrée/Sortie (SBES) dans un système d'ordinateur personnel (10) normalement relié à une unité de disque fixe (62), le dit système (10) ayant un processeur de système (26) couplé électriquement à un circuit planaire (24), le dit circuit planaire (24) étant de plus couplé électriquement à une mémoire morte (36), à une mémoire vive (32) et à une mémoire rémanente (58), le procédé comprenant les étapes suivantes :

(a) initialiser le dit système (10) avec une première partie du dit SBES résidant dans la dite mémoire morte (36) ;

(b) déterminer si la dite unité de disque fixe (62) est présente ;

(c) en l'absence de la dite unité de disque fixe (62), rechercher un mot de passe dans la dite mémoire rémanente (58) ;

(d) si l'absence d'un mot de passe est détectée, initialiser, avec la dite première partie du dit SBES, la dite unité de disquette (66) ayant un article de chargement du fichier permanent et la partie restante du dit SBES ;

(e) charger le dit article de chargement du fichier permanent avec la dite première partie du dit SBES dans la dite mémoire vive (32), le dit article de chargement du fichier permanent incluant un segment de données (122-138) et un segment de code exploitable directement (120), le dit segment de données (122-138) contenant des données représentant la configuration du matériel du dit système (10) avec lequel la partie restante du dit SBES est compatible ;

(f) vérifier la compatibilité du dit article de chargement du fichier permanent avec le matériel du dit système en comparant les données, tirées du dit segment de données (122-138), représentant l'enregistrement de la configuration du matériel, avec les données correspondantes enregistrées dans la dite mémoire morte (36) ;

(g) exécuter le segment de code du dit article de chargement du fichier permanent pour charger la partie restante du dit SBES dans la mémoire centrale ; et

(h) passer le contrôle au code de la partie restante du SBES, une fois qu'il a été chargé dans la dite mémoire centrale.

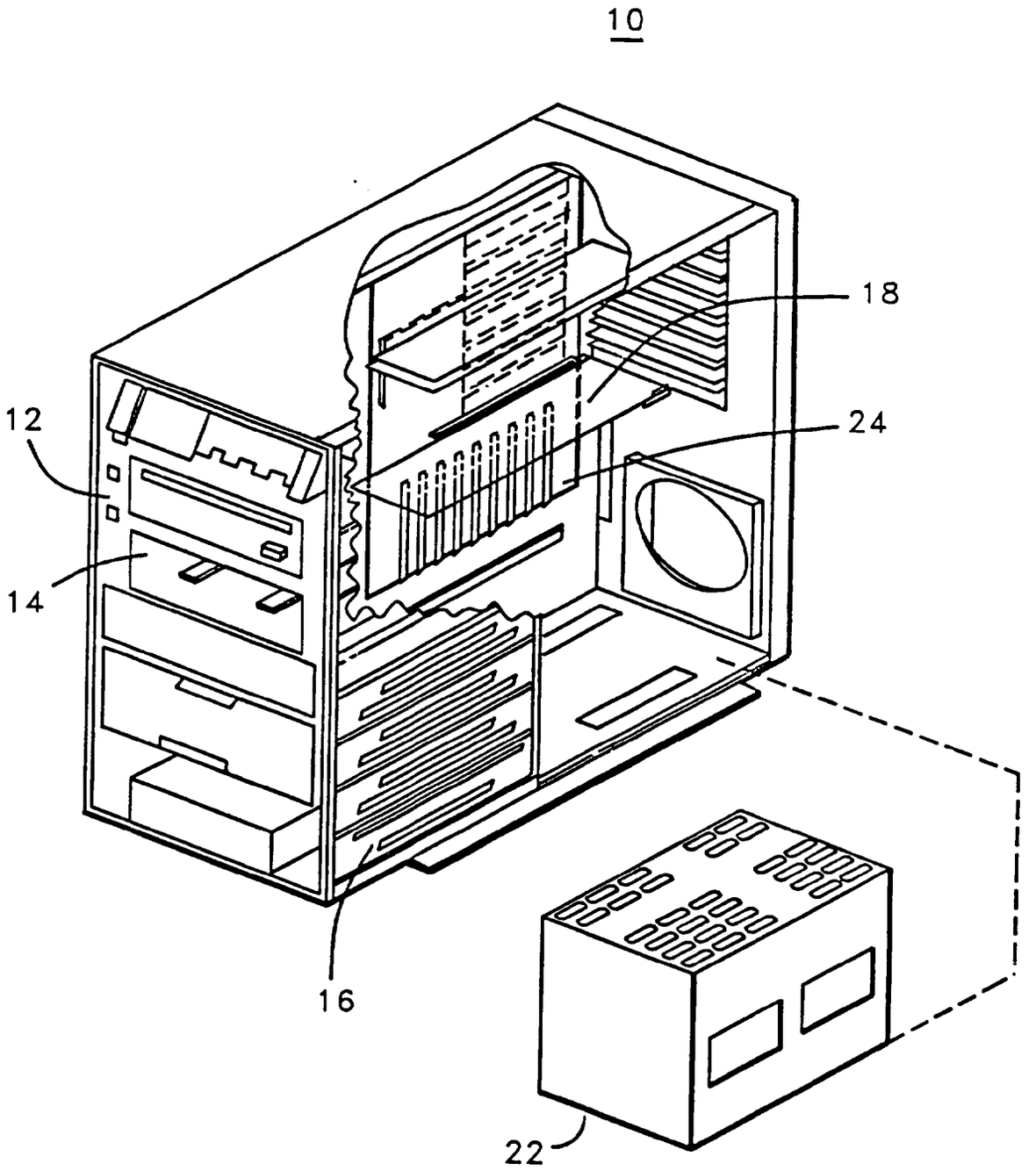

FIG. 1

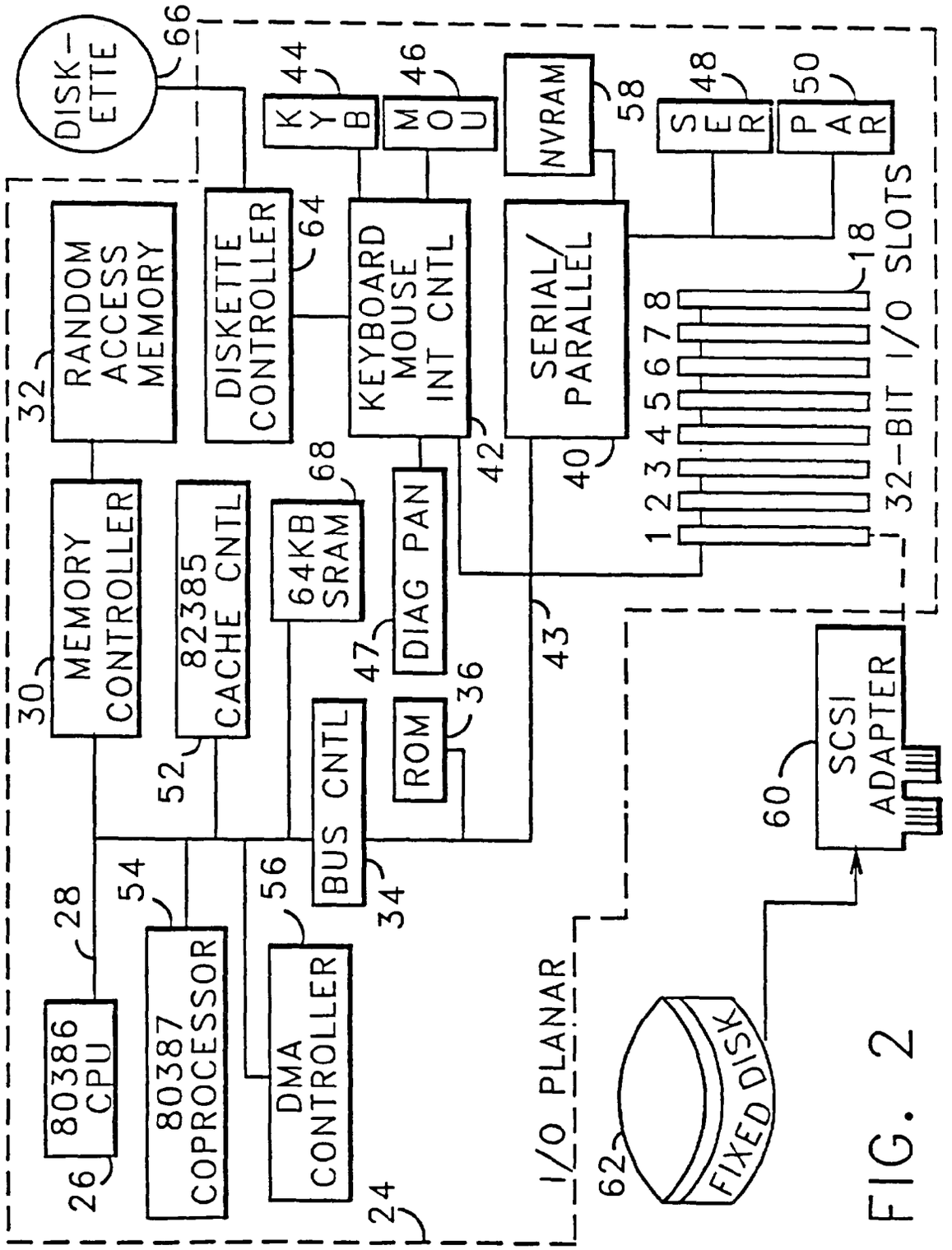
DISK-ETTE
66
K Y B
44
M O U
46
NVRAM
58
S E R
48
P A R
50
RANDOM ACCESS MEMORY
32
DISKETTE CONTROLLER
64
KEYBOARD MOUSE INT CNTL
SERIAL/ PARALLEL
40
42
18
1 2 3 4 5 6 7 8
32-BIT I/O SLOTS
MEMORY CONTROLLER
30
82385 CACHE CNTL
52
64KB SRAM
68
DIAG PAN
47
43
ROM
36
BUS CNTL
34
54
56
28
80386 CPU
26
80387 COPROCESSOR
DMA CONTROLLER
24
I/O PLANAR
SCSI ADAPTER
60
FIXED DISK
62

FIG. 2

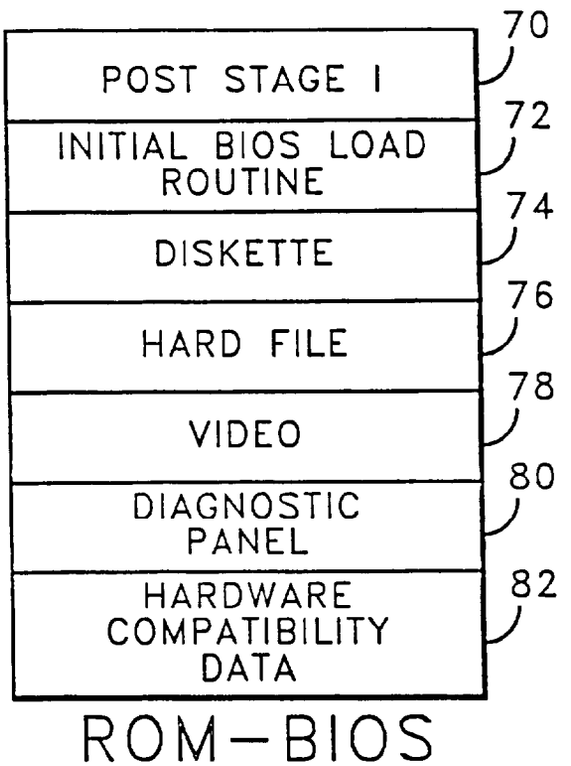
POST STAGE I
70
INITIAL BIOS LOAD ROUTINE
72
DISKETTE
74
HARD FILE
76
VIDEO
78
DIAGNOSTIC PANEL
80
HARDWARE COMPATIBILITY DATA
82
ROM-BIOS

FIG. 3

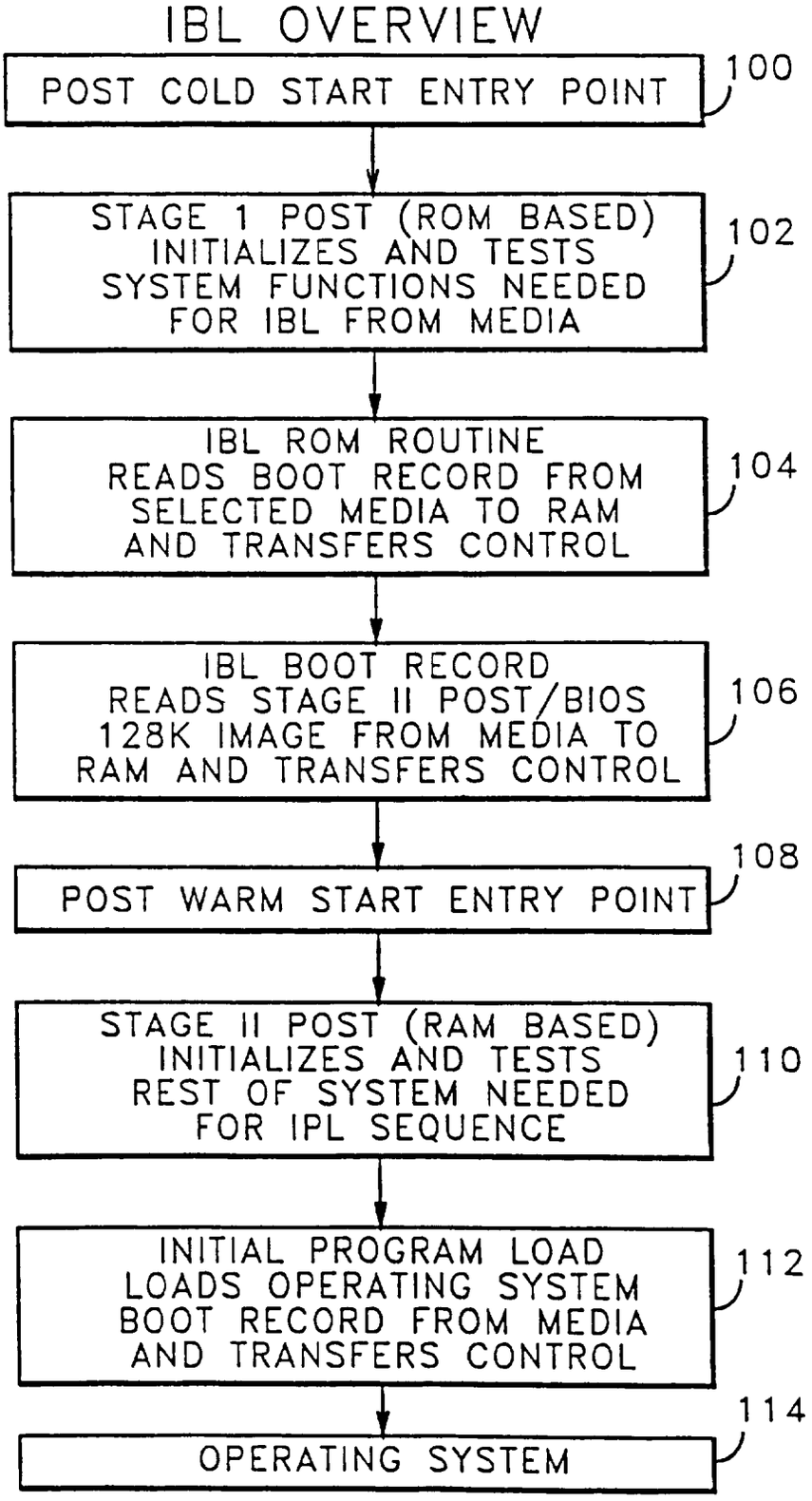
IBL OVERVIEW
POST COLD START ENTRY POINT
100
STAGE 1 POST (ROM BASED) INITIALIZES AND TESTS SYSTEM FUNCTIONS NEEDED FOR IBL FROM MEDIA
102
IBL ROM ROUTINE READS BOOT RECORD FROM SELECTED MEDIA TO RAM AND TRANSFERS CONTROL
104
IBL BOOT RECORD READS STAGE II POST/BIOS 128K IMAGE FROM MEDIA TO RAM AND TRANSFERS CONTROL
106
POST WARM START ENTRY POINT
108
STAGE II POST (RAM BASED) INITIALIZES AND TESTS REST OF SYSTEM NEEDED FOR IPL SEQUENCE
110
INITIAL PROGRAM LOAD LOADS OPERATING SYSTEM BOOT RECORD FROM MEDIA AND TRANSFERS CONTROL
112
OPERATING SYSTEM
114

FIG. 4

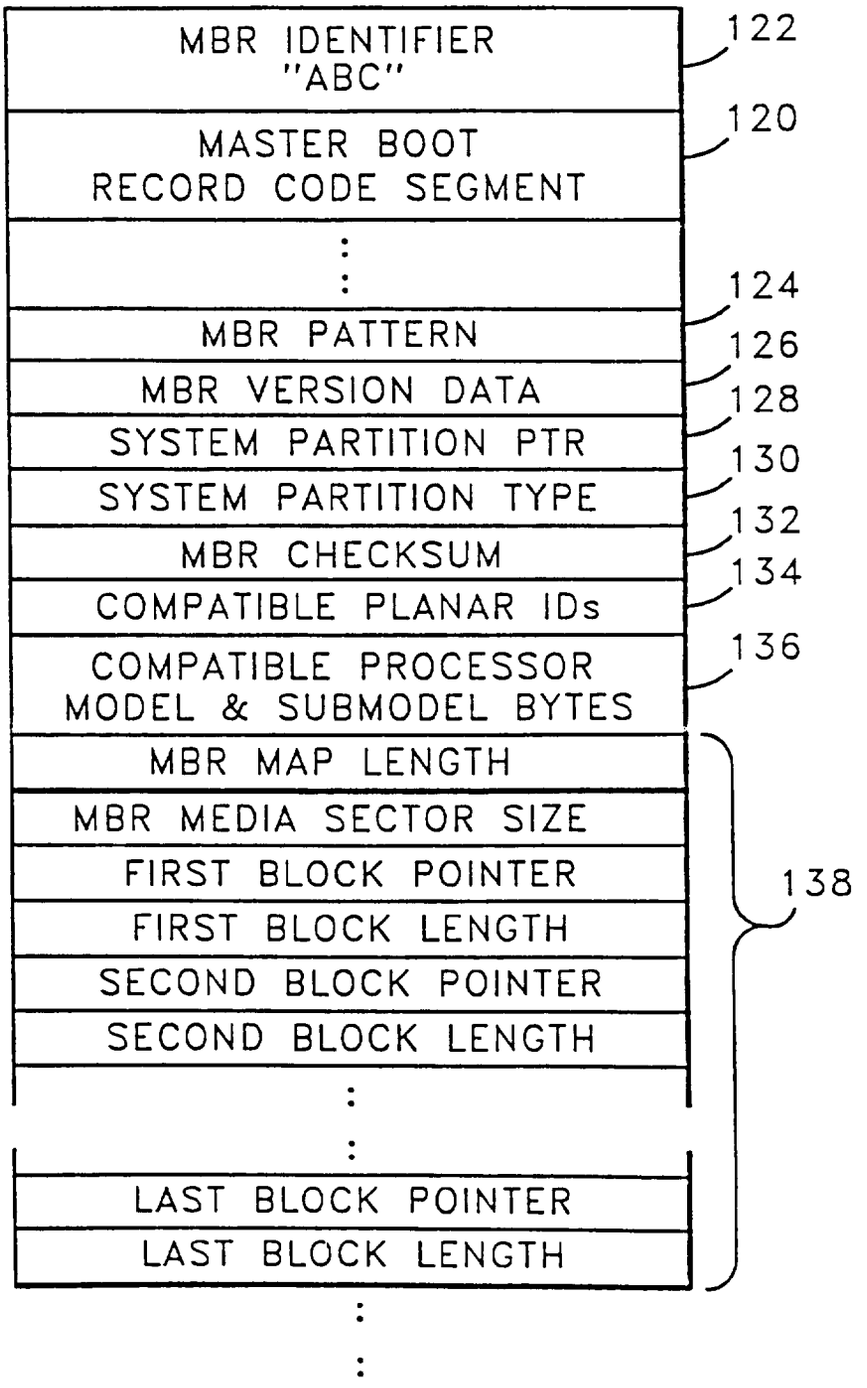
MBR IDENTIFIER "ABC"
122
MASTER BOOT RECORD CODE SEGMENT
120
124
MBR PATTERN
126
MBR VERSION DATA
128
SYSTEM PARTITION PTR
130
SYSTEM PARTITION TYPE
132
MBR CHECKSUM
134
COMPATIBLE PLANAR IDs
136
COMPATIBLE PROCESSOR MODEL & SUBMODEL BYTES
MBR MAP LENGTH
MBR MEDIA SECTOR SIZE
FIRST BLOCK POINTER
FIRST BLOCK LENGTH
SECOND BLOCK POINTER
SECOND BLOCK LENGTH
LAST BLOCK POINTER
LAST BLOCK LENGTH
138

FIG. 5

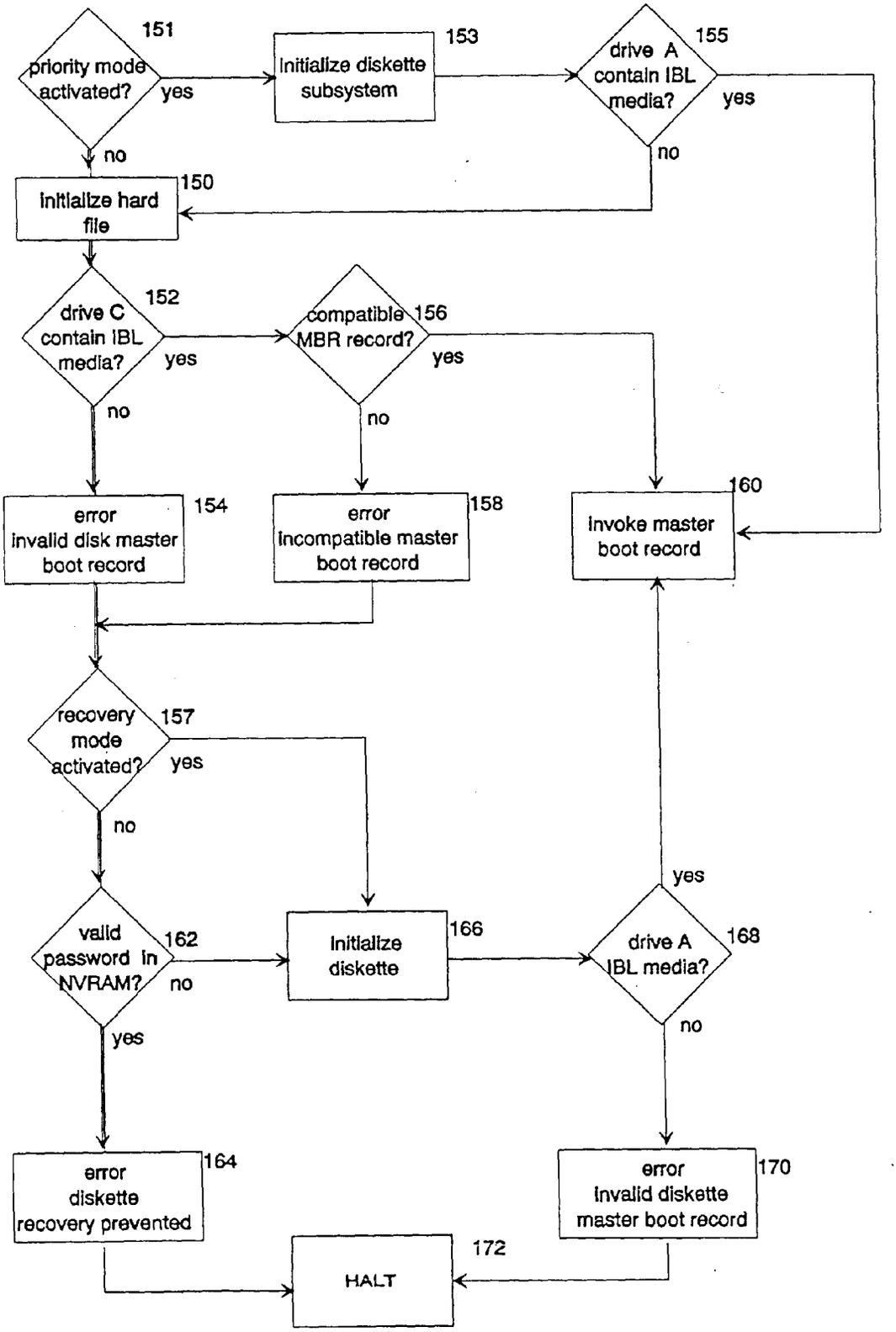
151
priority mode activated?
yes
no
153
Initialize diskette subsystem
155
drive A contain IBL media?
yes
no
150
Initialize hard file
152
drive C contain IBL media?
yes
no
156
compatible MBR record?
yes
no
154
error invalid disk master boot record
158
error incompatible master boot record
160
invoke master boot record
157
recovery mode activated?
yes
no
162
valid password in NVRAM?
no
yes
166
Initialize diskette
168
drive A IBL media?
yes
no
164
error diskette recovery prevented
170
error invalid diskette master boot record
172
HALT

FIG. 6A

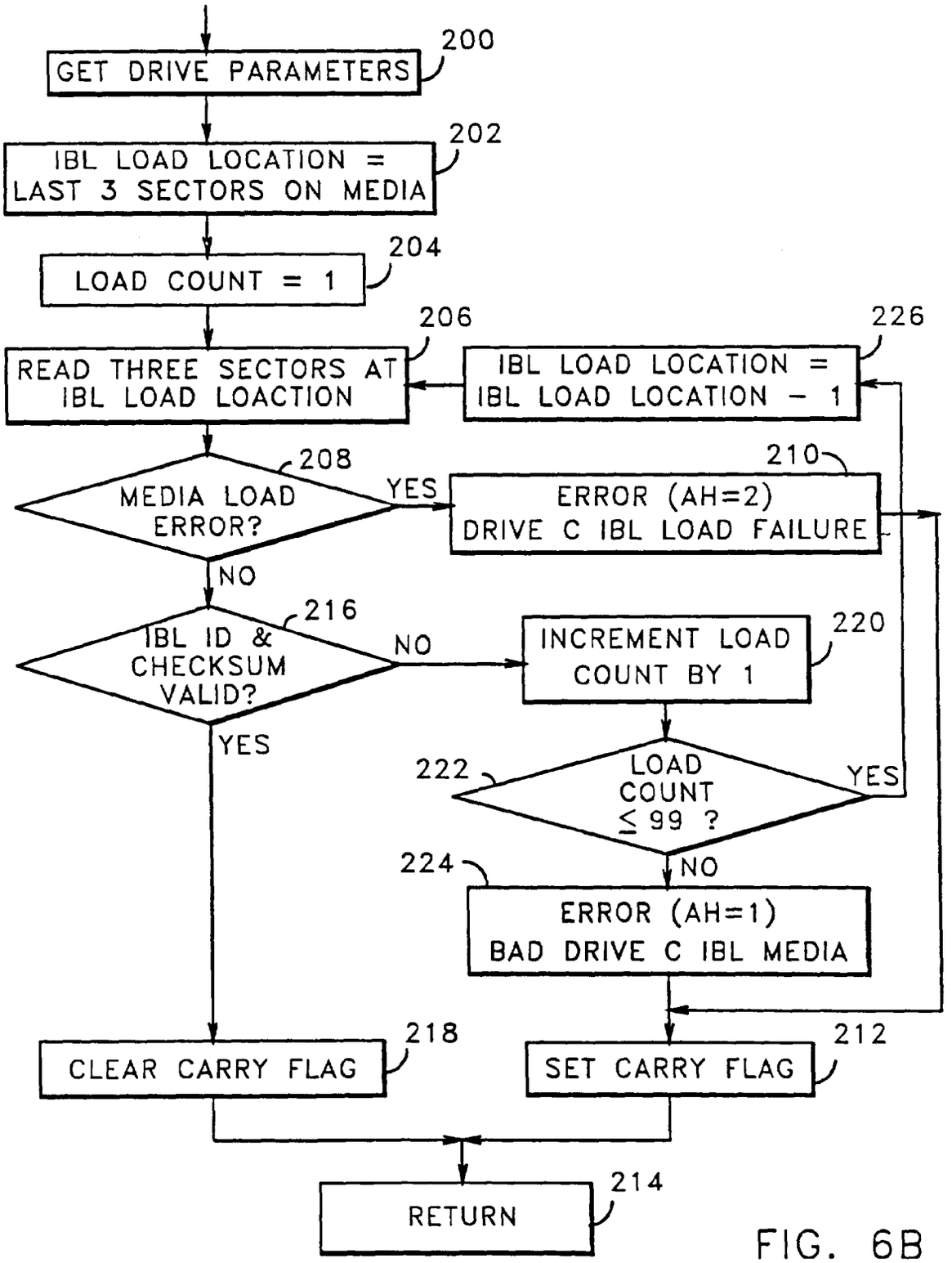
GET DRIVE PARAMETERS
200
IBL LOAD LOCATION = LAST 3 SECTORS ON MEDIA
202
LOAD COUNT = 1
204
READ THREE SECTORS AT IBL LOAD LOACTION
206
IBL LOAD LOCATION = IBL LOAD LOCATION - 1
226
MEDIA LOAD ERROR?
208
YES
ERROR (AH=2) DRIVE C IBL LOAD FAILURE
210
NO
IBL ID & CHECKSUM VALID?
216
NO
INCREMENT LOAD COUNT BY 1
220
YES
LOAD COUNT ≤ 99 ?
222
YES
NO
ERROR (AH=1) BAD DRIVE C IBL MEDIA
224
CLEAR CARRY FLAG
218
SET CARRY FLAG
212
RETURN
214

FIG. 6B

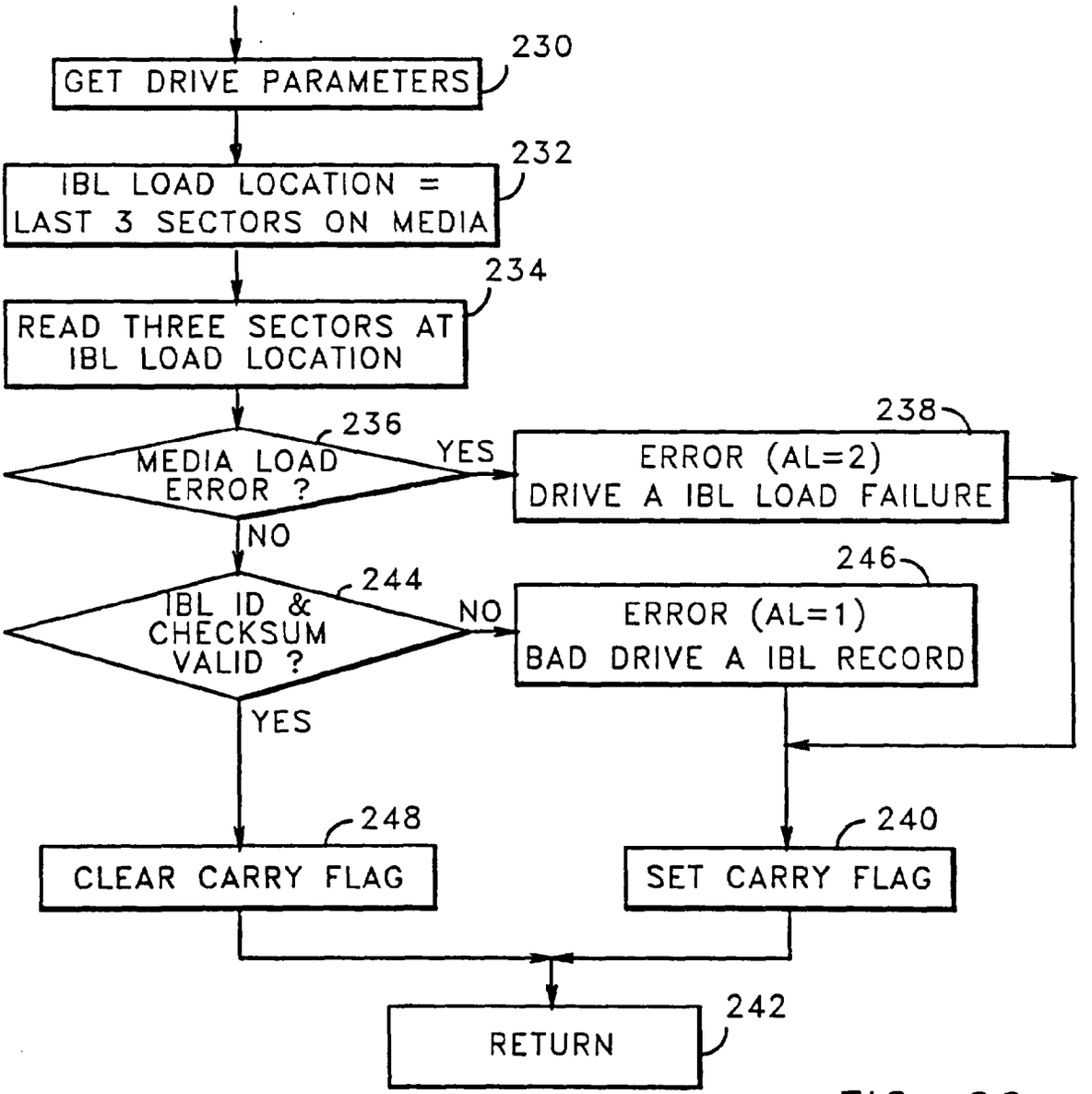
GET DRIVE PARAMETERS
230
IBL LOAD LOCATION = LAST 3 SECTORS ON MEDIA
232
READ THREE SECTORS AT IBL LOAD LOCATION
234
MEDIA LOAD ERROR ?
236
YES
ERROR (AL=2) DRIVE A IBL LOAD FAILURE
238
NO
IBL ID & CHECKSUM VALID ?
244
NO
ERROR (AL=1) BAD DRIVE A IBL RECORD
246
YES
CLEAR CARRY FLAG
248
SET CARRY FLAG
240
RETURN
242

FIG. 6C

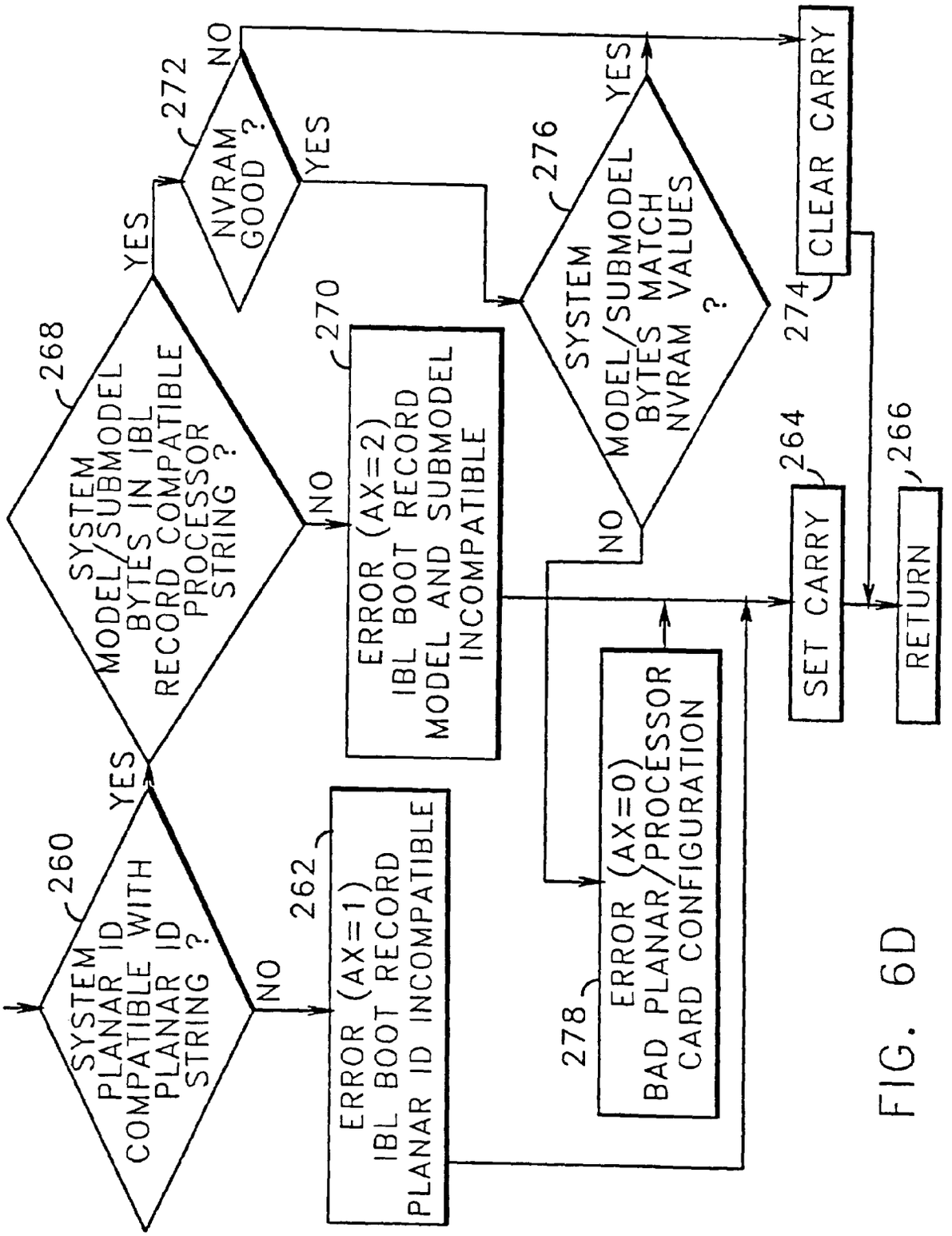
260
SYSTEM PLANAR ID COMPATIBLE WITH PLANAR ID STRING ?
YES
NO
262
ERROR (AX=1) IBL BOOT RECORD PLANAR ID INCOMPATIBLE
268
SYSTEM MODEL/SUBMODEL BYTES IN IBL RECORD COMPATIBLE PROCESSOR STRING ?
YES
NO
270
ERROR (AX=2) IBL BOOT RECORD MODEL AND SUBMODEL INCOMPATIBLE
272
NVRAM GOOD ?
YES
NO
276
SYSTEM MODEL/SUBMODEL BYTES MATCH NVRAM VALUES ?
YES
NO
278
ERROR (AX=0) BAD PLANAR/PROCESSOR CARD CONFIGURATION
274
CLEAR CARRY
264
SET CARRY
266
RETURN

FIG. 6D

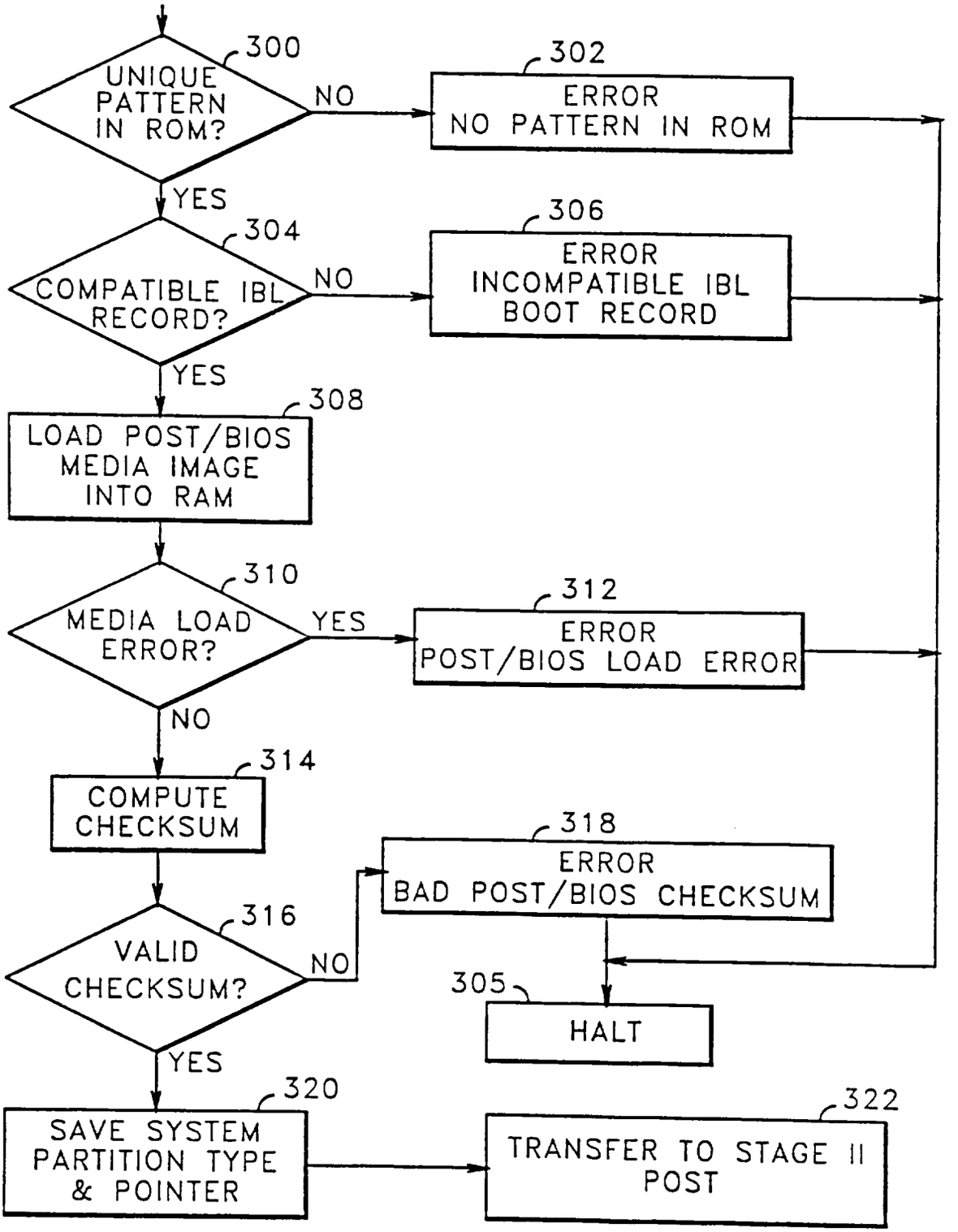
300
UNIQUE PATTERN IN ROM?
NO
302
ERROR NO PATTERN IN ROM
YES
304
COMPATIBLE IBL RECORD?
NO
306
ERROR INCOMPATIBLE IBL BOOT RECORD
YES
308
LOAD POST/BIOS MEDIA IMAGE INTO RAM
310
MEDIA LOAD ERROR?
YES
312
ERROR POST/BIOS LOAD ERROR
NO
314
COMPUTE CHECKSUM
318
ERROR BAD POST/BIOS CHECKSUM
316
VALID CHECKSUM?
NO
305
HALT
YES
320
SAVE SYSTEM PARTITION TYPE & POINTER
322
TRANSFER TO STAGE II POST

FIG. 7