9600369-4
IPOS Copy
THIS SHEET HAS BEEN REPLACED

COMPUTER SYSTEM

The present invention relates to a computer system and in particular to apparatus for installing BIOS 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

This Sheet Has Been Replaced

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.

THIS SHEET HAS BEEN REPLACED

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, it has become necessary to decrease the operating size of BIOS while increasing or at least maintaining the same level of functionality. Since marketability and consumer acceptance of personal computer systems appear to require the ability to maintain compatibility, it should be appreciated that maintaining the same level of functionality of BIOS is a substantial factor in achieving success in accordance with this invention. Thus, there exists a need for developing a method and apparatus which decreases the code space required for BIOS in Family II machines.

An aim of of the present invention is to provide a computer system having apparatus for decreasing the operating size of BIOS by storing a portion of BIOS on a direct access storage device, while maintaining compatibility with previous BIOS type operating systems.

In accordance with the present invention, there is now provided.....

Viewing the present invention from a second aspect, there is now provided, an apparatus for decreasing the operating size of BIOS for a personal computer system , the personal computer system having a system processor electrically coupled to a read only memory , random access memory and direct access storage device, the apparatus comprising; a first portion of BIOS resident in the read only memory, the first portion of BIOS having means for initializing the system and the direct access storage device; a master boot record included in the direct access storage device, the master boot record including an executable code segment; and a remaining portion of BIOS being included on the direct access storage device, the remaining portion of BIOS having means for operating the personal computer system, wherein the first portion of BIOS initializes the system and the direct access storage device to load the master boot record, the first portion of BIOS transferring control to the executable

This Sheet Has Been Replaced



THIS SHEET HAS BEEN REPLACED

code segment of the master boot record in order to effect the loading of the remaining portion of BIOS into the random access memory to begin operating the system.

Viewing the present invention from a third aspect, there is now provided an apparatus for loading BIOS into a personal computer system, the personal computer system having a system processor, a read only memory, a random access memory, and at least one direct access storage device, the apparatus comprising: a first portion of BIOS included in the read only memory, the first portion of BIOS including: means for initializing the system processor and the at least one direct access storage device, a loading means for loading data records from the at least one direct access storage device into random access memory, a validation means for confirming the personal computer system is compatible with BIOS; a remaining portion of BIOS included in the at least one direct access storage device, the remaining portion of BIOS including reusable means for assisting in the operation of the personal computer system, wherein the first portion of BIOS initializes the system and the at least one direct access storage device for effecting the loading of the remaining BIOS portion into random access memory after confirming that the remaining portion of BIOS is compatible with the system.

Viewing the present invention from a fourth aspect, there is now provided a method for storing BIOS in a personal computer system, the system including a system processor electrically coupled to a read only memory, a random access memory, and direct storage access device, the method comprising the steps of:

(a) A storing a first portion of BIOS in the read only memory, the first portion of BIOS including means for initializing the system and the direct access storage device;

(b) storing a remaining portion of BIOS on the direct access storage device, the remaining portion of BIOS including means for operating the personal computer system in a normal manner;

This Sheet Has Been Replaced



This Sheet Has Been Replaced



THIS SHEET HAS BEEN REPLACED

(c) initializing the system and the direct access storage device with the first portion of BIOS;

(d) loading a master boot record into random access memory, the master boot record including an executable code segment;

(e) transferring control to the executable code segment to load the remaining portion of BIOS into the random access memory.

A preferred example of a method of the present invention includes the step (f) of verifying the master boot record is compatible with the system by comparing data stored in the first BIOS portion with corresponding data stored in the master boot record. Another preferred method of the present invention also include the step (g) of verifying the master boot record is compatible with the system processor by comparing data in the read only memory to corresponding data included in the master boot record. A further preferred method of the present invention includes the step (h) of generating an indication that the master boot record is not compatible.

In an example of the present invention, there is provided a computer system having apparatus for permitting a portion of BIOS to be executed and then rendered inaccessible to the system. In preferred examples of the present invention, there is provided a BIOS record format which segments BIOS according to function in order to store those BIOS routines which are executed only once on ROM, while those BIOS routines which are executed more than once are stored on the direct access storage device.

In a preferred example of the present invention to be described later, there is provided a personal computer system comprising a system processor, a random access main memory, a read only memory, and at least one direct access storage device. The read only memory includes a first portion of BIOS while the remaining portion is stored on the direct access storage device. The first BIOS portion includes routines that are used for initialization and set up, while the remaining portion of BIOS

THIS SHEET HAS BEEN REPLACED

includes routines that are reusable by the operating system. In operation, the first portion of BIOS initializes the system processor and the direct access storage device to read in a master boot record into the random access memory from the direct access storage device.

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 supported by the master boot record. The first BIOS portion 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 the system processor, system planar, and planar I/O configuration.

If the master boot record is compatible with the system hardware, the first BIOS portion vectors the system processor to execute the executable code segment of the master boot record. The first BIOS portion is then discarded by the system. The executable code segment confirms that the system configuration has not changed and loads in the remaining BIOS portion from the direct access storage device 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. The remaining portion of BIOS includes only those routines needed for normally operating the system. The first BIOS portion, being no longer addressable and superseded by the remaining BIOS portion, is abandoned.

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

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;

This Sheet Has Been Replaced

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 a BIOS image 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 executable code segment of the master boot record to load a BIOS image.

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 instructions to input, process, and output information.

In use, the personal computer system 10 is designed 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 an operating system, such as IBM's OS/2 Operating System or PC-DOS. This type of system includes a BIOS 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 indepth operating knowledge of a particular device. 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 58 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 a 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, 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 NVRAM 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 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 "hardfile" 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 includes routines used only for initializing the system and for loading a second or remaining BIOS portion. When the first BIOS portion is executed by the system processor 26, it inputs from either the fixed disk 62 or diskette 66 the second or remaining portion of BIOS, hereinafter referred to as a BIOS image. This BIOS image supersedes the first BIOS portion and includes operating routines which are reusable and therefore are an integral part of the system. In practice, these routines remain 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. In general upon power up, the processor is vectored to ROM-BIOS which initializes the system and DASD 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. ROM-BIOS transfers control to the code segment which code uses the data information to validate hardware compatibility and system configuration. After testing for hardware compatibility and proper system configuration, the executable code loads the BIOS image into RAM. The BIOS image supersedes ROM-BIOS and includes operating system routines which are useable to provide for the 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. These code modules are judicially chosen to include only routines that are necessary to initialize the system, test the hardware necessary to load the BIOS image, and provide communications with the user. 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 hardfile 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 functions 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 data values to confirm hardware compatibility and system configuration which are described later with respect to Fig. 5. It is important to note that the ROM-BIOS routines are not necessary to normally operate the system. The ROM-BIOS routines are used only once to initialize, test, and load. ROM-BIOS is then discarded and superseded by the BIOS image. This segregation of BIOS functions effectively reduces the memory requirements for BIOS under normal operating conditions.

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, the system processor is vectored to the entry point of POST Stage I, steps 101 and 100. POST Stage I initializes 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, determines whether the BIOS image is stored on fixed disk or can be loaded from diskette; and second, loads the master boot record from the selected media (either disk or diskette) 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 110. In particular, the BIOS image is loaded into the 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 of RAM thus superseding ROM-BIOS. Control is then transferred to POST Stage II which was included in the newly loaded BIOS image thus effectively abandoning ROM-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.

Referring back to step 110, it is noted that entry to POST Stage II can be put in effect through a reset function, step 107. The reset function is normally actuated by the user whenever a "warm start" is desired, step 108. For example, the reset function can be activated by simultaneously pressing the ctrl-alt-del keys on the keyboard. The effect of a warm start is to bypass the pre-initialization, testing, and initial BIOS load. It is important to note that a warm start does not reboot BIOS, however, the operating system will be rebooted by POST Stage II. Thus if the BIOS in RAM has not been affected, the system should operate in a normally manner after rebooting.

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 verifies 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 map the image to RAM and then transfer control to the POST Stage II routine of the system image.

MBR 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 to 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. If a system does not have a hardfile, 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 one, 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 default mode in which the master boot record can be loaded from diskette. However, the IBL routine does not allow the diskette default mode to be performed if the system contains the IBL media on the system fixed disk and a valid password is present in NVRAM. The user has the option of setting the password in NVRAM. The purpose of preventing the diskette default mode from being effected is to prevent loading an authorized BIOS image from diskette. In other words, the diskette default mode is used only when a system fixed disk is not operational and the user has indicated (by not setting the password) the desire to be able to load from the diskette. If the IBL routine is not able to load the master boot record from either media, an error message is generated and the system is halted.

Referring now to Fig. 6A, under normal circumstances the system will contain a system fixed disk which the IBL routine initializes, step 150. 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 C to determine whether it contains IBL media, step 152. Attention is directed to Fig. 6B which describes in detail this process. 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 it is not planar or processor compatible, then an error is reported, step

158. Referring back to step 152, if no master boot record is found on the last 99 sectors of the fixed disk (primary hardfile), an error is reported, step 154.

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 IBL routine determines if a valid password is included in NVRAM, step 162. This password determines whether the BIOS image can be loaded from diskette. Note that the password will exist only upon being installed by the user running a setup utility. If a password is installed in NVRAM, the BIOS image is prevented from being loaded from diskette, step 164. This permits the user to ensure the integrity of the operation of the system by causing the system to be loaded only with the BIOS image on the fixed disk. 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. The reason for the absence of the validity checks is for loading a noncompatible IBL image from diskette. For example, if a new processor is added to the system, a new BIOS image will be included on a diskette. Since a new processor will cause validity errors when loading from fixed disk, the IBL routine provides the ability to bypass these tests by loading the BIOS image from diskette.

To recapitulate, 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 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 reference diskette.

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 224, 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 is 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 process or 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 configuration 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 Configuration 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, steps 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 configuration has changed, and an error is generated, step 306. The system will halt when the IBL record is incompatible with 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, steps 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, steps 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 decreasing the operating size of BIOS. A first portion of BIOS is stored in ROM and includes system initialization and tests necessary to boot up a remaining portion of BIOS, stored on a DASD into RAM. The first and remaining portions of BIOS are executed upon powering up, while the remaining portion need only be executed upon resetting the system.

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 and equivalent.

This Sheet Has Been Replaced

THIS SHEET HAS BEEN REPLACED

CLAIMS

1. A computer system (10) comprising: a system processor (26) electrically coupled to a read only memory (36), a random access memory (32) and at least one direct access storage device (62,66),

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 having means for initializing the system (10) and the direct access storage device (62,66);

a master boot record included in the direct access storage device (62,66), the master boot record including an executable code segment (120); and

a remaining portion of BIOS included on the direct access storage device (62,66), the remaining portion of BIOS having means for operating the system (10), wherein the first portion of BIOS initializes the system (10) and the direct access storage device (62,66) to load the master boot record, the first portion of BIOS transferring 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) to begin operating the system (10).

2. A computer system (10) as claimed in claim 1, wherein the initializing means of the first portion of BIOS includes a power on self test routine, the power on self test routine initializing and testing only those system functions necessary to load the remaining portion of BIOS.

3. A computer system (10) as claimed in claim 2, wherein the power on self test initializes the system processor (26) functions, memory subsystems, and direct access storage device (62,66) subsystem.

* C L M S *

THIS SHEET HAS BEEN REPLACED

4. A computer system (10) as claimed in claim 3, wherein the initializing means of the first portion of BIOS includes a load routine for loading the master boot record into random access memory (32) and transferring control to the master boot record.

5. A computer system (10) as claimed in claim 4, wherein the master boot record further includes a data segment (122-138), the data segment (122-138) representing a hardware configuration of the system (10) which is compatible with the master boot record, and 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).

6. A computer system (10) as claimed in claim 5, wherein the data segment (122-138) of the master boot record includes a value (134) representing a system planar (24) which is compatible with the master boot record and 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).

7. A computer system (10) as claimed in claim 6, 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

CLMS

This Sheet Has Been Replaced

THIS SHEET HAS BEEN REPLACED

the master boot record is compatible with the system processor (26) and the I/O configuration of the system planar (24).

8. A computer system (10) as claimed in claim 1 further comprising: loading means for loading data records from at least one direct access storage device (62,66) into random access memory (32); and a validation means for confirming the system (10) is compatible with BIOS.

9. A computer system (10) as claimed in claim 8, wherein the validation means includes data representing the type of system processor (26) and configuration of a system planar (24) coupled to the system processor (26).

10. A computer system (10) as claimed in claim 9, wherein the validation means includes a means for indicating errors in loading the remaining portion of BIOS.

11. A computer system (10) as claimed in claim 10, wherein the means for indicating errors comprises a diagnostic panel.

12. A computer system (10) as claimed in claim 10, wherein the means for indicating errors comprises a video I/O controller.

13. A computer system (10) as claimed in claim 8, wherein the direct access storage device (62,66) comprises a fixed disk drive (62) wherein the loading means loads data records from the fixed disk drive (62) into random access memory (32).

14. A computer system (10) as claimed in claim 8, wherein the direct access storage device (62,66) comprises a diskette drive (66) wherein the loading means loads data records from the diskette drive (66) into random access memory (32).

15. A method for storing BIOS in a personal computer system (10). the system (10) including a system processor (26) electrically coupled to a

CLMS

This Sheet Has Been Replaced

THIS SHEET HAS BEEN REPLACED

read only memory (36), a random access memory (32), and direct access storage device (62,66), the method comprising the steps of:

(a) storing a first portion of BIOS in the read only memory (36), the first portion of BIOS including means for initializing the system (10) and the direct access storage device (62,66);

(b) storing a remaining portion of BIOS on the direct access storage device (62,66), the remaining portion of BIOS including means for operating the system (10) in a normal manner;

(c) initializing the system (10) and the direct access storage device (62,66) with the first portion of BIOS;

(d) loading a master boot record into random access memory (32), the master boot record including an executable code segment (120);

(e) transferring control to the executable code segment (120) to load the remaining portion of BIOS into the random access memory (32).

* C L M S *

This Sheet Has Been Replaced