US005732268A

United States Patent [19]

Bizzarri

[11] Patent Number: 5,732,268

[45] Date of Patent: Mar. 24, 1998

[54] **EXTENDED BIOS ADAPTED TO ESTABLISH REMOTE COMMUNICATION FOR DIAGNOSTICS AND REPAIR**

[75] Inventor: **Maurice W. Bizzarri**, Palo Alto, Calif.

[73] Assignee: **Award Software International**, Mt. View, Calif.

[21] Appl. No.: **608,028**

[22] Filed: **Feb. 26, 1996**

[51] **Int. Cl.**[6] **G06F 11/34**

[52] **U.S. Cl.** **395/652**; 395/712

[58] **Field of Search** 395/652, 712, 395/704

[56] **References Cited**

U.S. PATENT DOCUMENTS

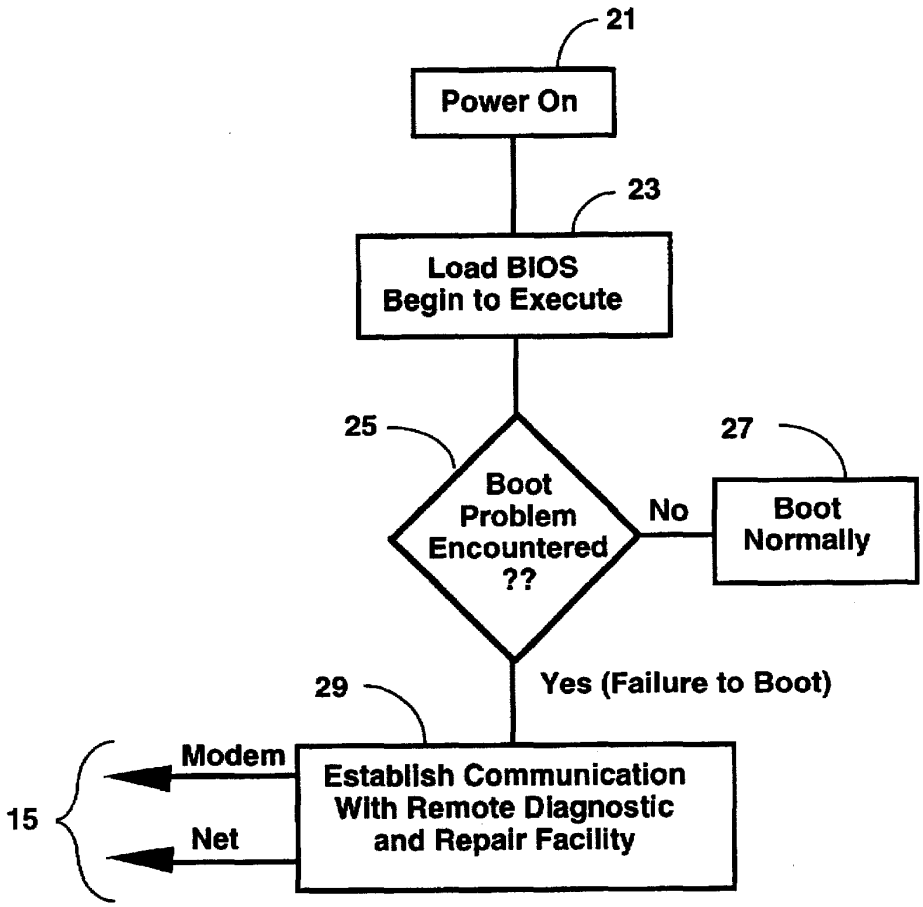
5,280,627	1/1994	Flaherty et al.	395/652
5,452,454	9/1995	Basu	395/652
5,596,711	1/1997	Burckhartt et al.	395/182.21

Primary Examiner—Emanuel T. Voeltz
Assistant Examiner—John Q. Chavis
Attorney, Agent, or Firm—Donald R. Boys

[57] **ABSTRACT**

An extended basic input output system (E-BIOS) has a first portion of code for providing power-on self-test (POST) and boot functions for a first computer, including code for sensing if the first computer does not boot. In the event of failure to boot, a second portion of code in the E-BIOS directs establishing communication link with a remote diagnostics and repair computer. When communication is established, a master code kernel at the diagnostics and repair computer may be executed to download a slave kernel to random access memory of the first computer, blowing an automatic software kernel or an operator at the diagnostics and repair computer to access and modify code and data in memory devices of the first computer, and to reboot the first computer after repair. Communication links may be by telephone modem, either analog or integrated Services Digital Network (ISDN), or by network links. In one embodiment cooperation between the slave kernel and the master kernel is such that an operator may operate the diagnostics and repair computer as though it were the first computer.

20 Claims, 3 Drawing Sheets



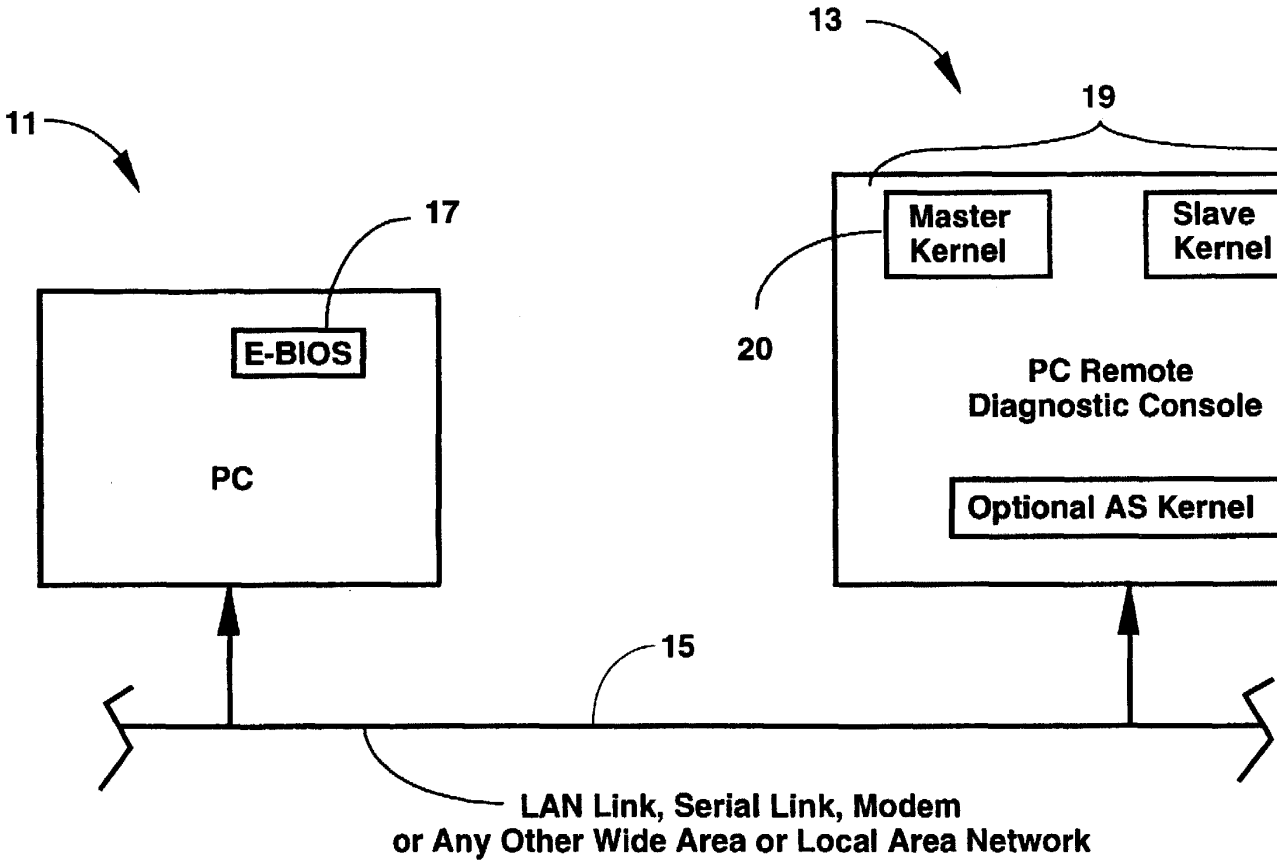


Fig. 1

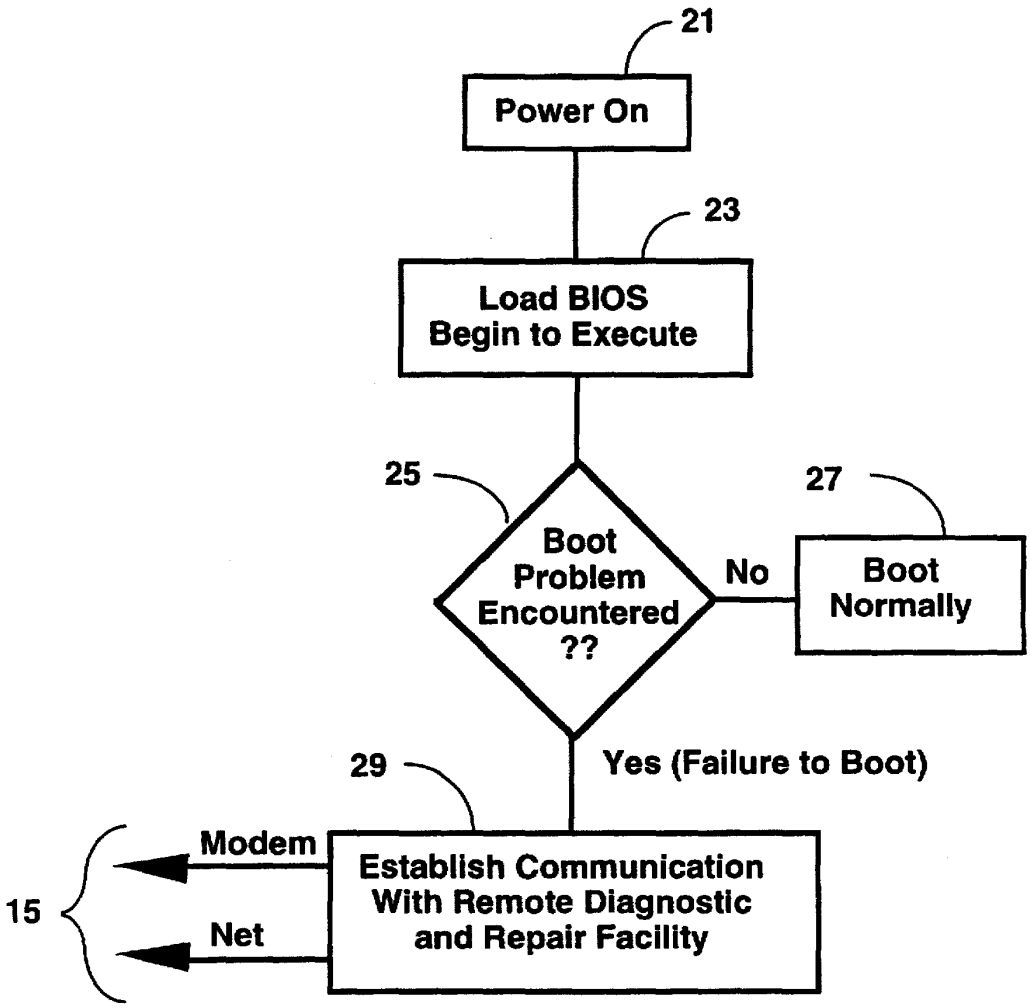


Fig. 2

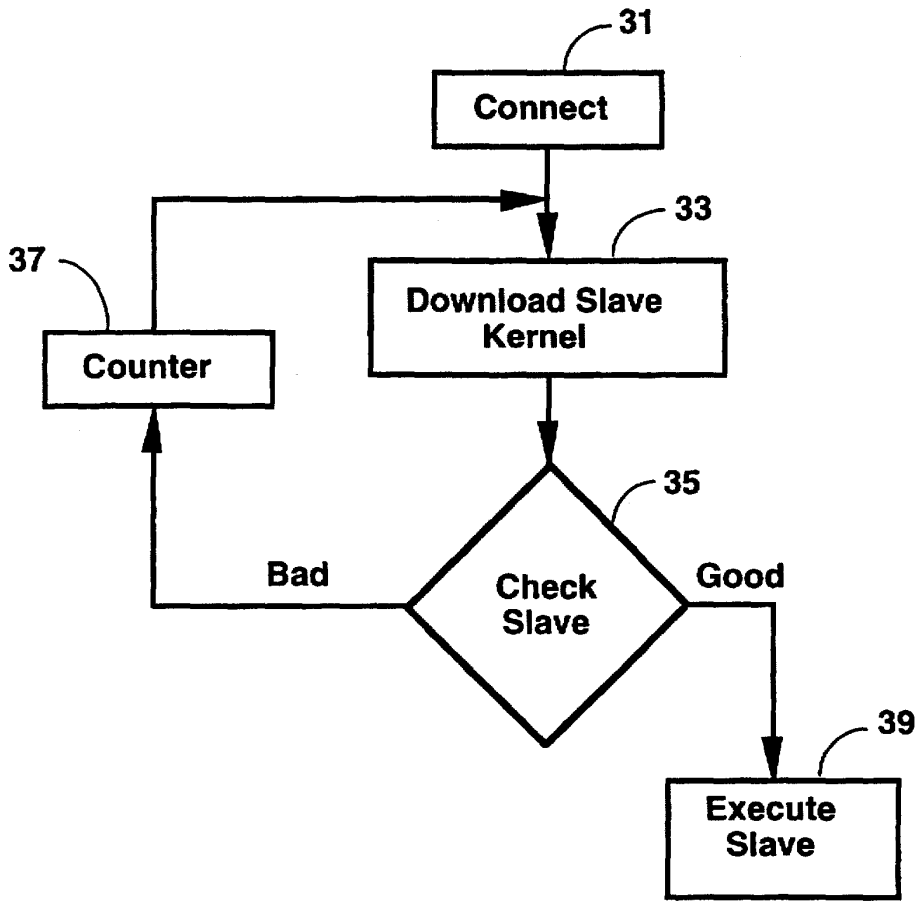


Fig. 3

EXTENDED BIOS ADAPTED TO ESTABLISH REMOTE COMMUNICATION FOR DIAGNOSTICS AND REPAIR

FIELD OF THE INVENTION

The present invention is in the area of apparatus and methods for diagnosing and repairing failed computer systems, and pertains more particularly to computer code for basic input output systems (BIOS) devoted to diagnosis and repair.

BACKGROUND OF THE INVENTION

Because a computer, without operating instructions loaded, is essentially a piece of dumb hardware, there must be some relatively standard set of beginning instructions for the computer to follow, readily accessible and loadable into the computer's operating memory to direct the process of testing the computer's hardware and connectivity, and then loading all of the various operating code that is necessary for the computer to perform useful tasks. Typically this beginning code is called a basic input output system (BIOS), which includes a power-on self-test (POST) procedure, to assure that all circuits are active, properly connected, and functional before attempting to load and initiate all of the code needed to operate.

The BIOS in most instances is a relatively short code set embedded in a non-volatile, read-only memory (ROM), such as an electrically-erasable programmable read-only memory (EEPROM), accessible to the central processing unit (CPU) of a computer at the instant of power on. Partly because code can be accessed and executed from random access memory (RAM) more quickly than from read-only memory (ROM), the BIOS code is typically loaded immediately into RAM from ROM at power on, and executed from RAM.

Although BIOS is fundamental to testing a computer's operating equipment and basic BIOS functions continue to be used during subsequent operation, BIOS is not a sufficient system for continuing successful system operations. There are many functions, among them reading and writing data to mass storage systems such a floppy disk drives (FDD), hard disk drives (HDD) and CD-ROM drives, which have to be loaded as well after executing the initial BIOS to render a computer fully operable. All of the code to make the computer capable of performing such basic functions is referred to in the art as the Operating System (OS).

In very early personal computers, before widespread use of hard disk drives (HDD), an operating system such as the IBM-compatible Disk Operating System (DOS) would typically be recorded on a floppy disk along with an application such as a word processor or a spread-sheet program. A floppy disk with DOS and a boot sector was called a Boot disk, and the name is still used, although operating systems and boot sectors are typically now recorded on hard disk drives. One would place a boot disk in a floppy drive in the computer, then turn the thing on. The BIOS would load from a resident ROM, perform a simple POST routine, then poll the floppy drives (often there was only one) to find DOS. When Dos was located, the BIOS would cause it to be loaded, and then present the DOS prompt for instruction from the user. BIOS code and DOS would remain in RAM for access by the CPU as needed for basic and routine functions. The user could then boot the word processor or other application on the floppy and go to work.

In this early and simple scenario, if DOS code on a boot disk were to become corrupted, the computer would not boot, and one could simply remove the boot disk from the floppy drive, place another in its place, and try again. Once a usable DOS was loaded, one could present different disks to the floppy drive(s) for access to applications and data files.

Things in a sense are not a lot different now. The principle differences are that mass storage devices have evolved and operating systems and applications are much larger and more complicated.

The motivation for evolution is functionality. As developers have envisioned ever more elaborate and intricate computer-related goodies such as high-resolution, color displays, CD-rom drives, high-density hard disk drives, menu-driven user interfaces, windows-type interfaces, modem communications, the Internet and the World-Wide-Web, and much more, it has been necessary to develop faster and more powerful CPUs, Higher density and faster mass storage, and ever-larger operating systems.

Because of the large size of operating systems today, such as Windows™ and UNIX™ systems for example, it is completely impractical to store the operating system on a floppy disk or in ROM, although some minimum versions of some operating systems are available in ROM for specialized purposes. Operating systems are therefore stored in non-volatile mass storage, typically on a hard disk. In such a system, at power on or reset, the BIOS ROM is accessed first, loaded into RAM, and executed. After POST, a BIOS code causes the system to access a local hard disk at an address, typically called the boot sector, where code is stored for directing loading of the operating system from the hard disk and finalizing preparation of the system for continuing operation. The operating system, which may be several megabytes of code, is then loaded, and the system is directed to a starting point for operator instructions, which may be with an application loaded and ready to run, or at a point where an operator may select an application.

For those familiar with DOS operation the point that the user is able to select and manipulate operation is the familiar DOS prompt. For those familiar with Microsoft Windows™ this point is the familiar interactive interface showing ICONS for program groups. For Apple Macintosh™ users this point is the familiar Desktop graphics interface.

As long as all hardware and software systems are operable, such an initiation sequence is routine. A hard disk drive, however, is a mechanical device, and thus vulnerable to a much broader variety of potential difficulties than an IC ROM. Because of this mechanical vulnerability, it often happens that a system cannot boot because the boot disk becomes inoperable. There are many other hardware failures that might occur other than the failure of a hard disk that would prevent a PC from booting and operating as intended.

As well as hardware-only failures, there are many situations that can occur to render software corrupted, which condition may leave a PC unable to boot, such as corruption of the boot sector of the hard disk boot drive, or corruption of a File Allocation Table (FAT).

Early PCs were self-sufficient, stand-alone systems, comprising all of the hardware and software necessary to perform intended functions. PCs at the time of the present invention, however, are typically provided with some form of communication link to other computers. Most PCs at a minimum have a telephone modem connection, and may communicate with another computer over a telephone line in either an analog or a digital protocol.

PCs used in a business setting are often connected together in a local area network (LAN), whereby individual workstations may have relatively fewer hardware compo-

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.