[barcode]
(51) Int. Cl.[6]:
G 06 F 13/12
G 06 F 3/00

DE 197 08 755 A1

(19) **Federal Republic of Germany**
[emblem]
German Patent Office

(12) **Offenlegungsschrift**
[= published patent application]

(10) **DE 197 08 755 A1**

(21) Application number: 197 08 755.8
(22) Filing date: March 4, 1997
(43) Disclosure date: September 17, 1998

(71) Applicant:
Tasler, Michael,
63773 Goldbach, DE

(74) Representative:
Schoppe, F., Grad. Eng. Univ., Patent Attorney, 81479 Munich

(72) Inventor:
Same as the applicant

(56) Documents cited:
EP 06 85 799 A
IBM Technical Disclosure Bulletin, vol. 38 (No. 05), p. 249;
National Instruments IEEE 488 and VXI bis Control. Catalog 1994, p. 3-188-3-122;

The following information is taken from the documents filed by the applicant.
An examination has been requested in accordance with § 44 of the German Patent Act.

(54) Flexible Interface

(57) An interface device delivers fast data communication between a host device with input/output interfaces and a data transmit/receive device, wherein the interface device comprises a processor unit, a memory unit, a first connecting device for interfacing the host device with the interface device, and a second connecting device for interfacing the interface device with the data transmit/receive device. The interface device is configured by means of the processor unit and the memory unit in such a way that, when receiving an inquiry from the host device through the first connecting device as to the type of device that is connected to said host device, irrespective of the type of data transmit/receive device, the interface device sends to the host device by way of the first connecting device a signal, which signals to the host device that it is communicating with the input/output device.

[right margin:] DE 197 08 755 A1

GERMAN FEDERAL GOVERNMENT PRINTING OFFICE BERLIN 07 98 802 038/45/1 23

Number: DE 197 08 755 A1
Int. Cl.[6]: G 06 F 13/12
Disclosure date: September 17, 1998

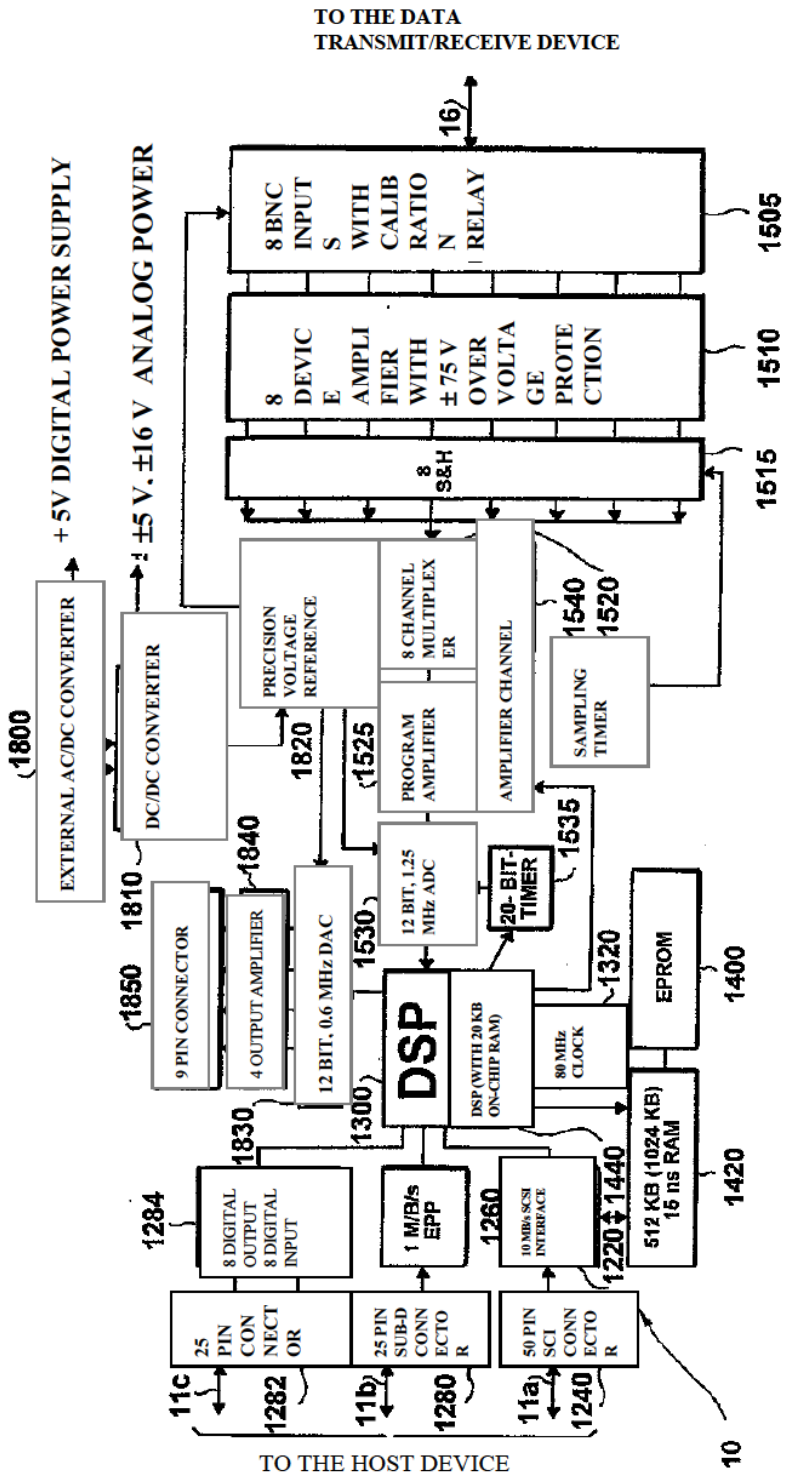

FIG.2

802 038/45

Specification

The present invention relates to the transfer of data and, in particular, to interface devices for communication between a computer or host device and a data transmit/receive device, from which data are to be acquired or with which communication is to take place.

Existing data acquisition systems for computers are very limited in their area of application. Generally such systems can be divided into two groups.

In the first group host devices or computer systems are connected by means of an interface to a device, the data of which are to be acquired. The interfaces of this group are typically standard interfaces, which can be used with a variety of host systems, using special driver software. One advantage of such interfaces is that they are largely independent of the host device. However, there is the disadvantage that they generally require very sophisticated drivers, which are fault-prone and which limit data transfer rates between the device, connected to the interface, and the host device and vice versa. Furthermore, it is often very difficult to implement such interfaces for portable systems; and they offer few possibilities for adaptation, with the result that such systems exhibit little flexibility.

The devices, from which data are to be acquired, cover the entire spectrum of electrical engineering. Hence, in a typical scenario it is assumed that a customer, who runs, for example, a diagnostic radiology system in the medical technology sector, reports a fault. Then a service technician of the system manufacturer will go to the customer and read the system log files, generated by the diagnostic radiology system, by means of, for example, a portable computer or laptop. Then, if the fault cannot be localized or if the fault occurs only intermittently, it will be necessary for the service technician to read not only an error log file, but also data from the current operation. It is apparent that in this case fast data transfer and rapid data analysis are necessary.

Another case for using an interface could be, for example, when an electronic measuring device, e.g. a multimeter, is connected to a computer system, in order to transfer the data, measured by the multimeter, to the computer. Especially in the case of long-term measurements or when there is a large volume of data, it is necessary that the interface allows a high data transfer rate.

From these randomly chosen examples it can be seen that the applications that are possible with an interface may vary widely. Therefore, it is desirable that an interface be so flexible that the interface can be used to connect very different kinds of electrical or electronic systems to a host device. In order to prevent operator error, it is also desirable that a service technician does not have to use different interfaces in different ways for different applications, but rather that, if possible, a universal method for operating the interface be provided for a large number of possible applications.

In order to increase the data transfer rates over an interface, the approach that was chosen for the second group of interface devices was to match individually the interface very closely to individual host systems or computer systems. The advantage of this solution is that high data transfer rates are possible. However, one drawback is that the drivers for the interfaces of the second group are very closely matched to a single host system, for which reason they generally cannot be used or can be used very ineffectively with other host systems. Furthermore, such types of interfaces that have the disadvantage that they must be mounted inside the computer housing, because they access the internal host bus system, in order to achieve maximum data transfer rates. Therefore, they generally do not lend themselves to portable host systems in the form of laptops have no free internal space to plug in an interface card on account of their size that is as small as possible.

A solution to this problem is offered by the interface devices of the company IOtech (business address: 25971 Cannon Road, Cleveland, Ohio 44146, USA). These interface devices are suitable for laptops, such as, for example, the WaveBook/512 model (registered trademark). The interface devices are connected by means of a plug-in card, which is approximately the size of a credit card, to the PCMCIA interface, which are now routinely provided as a standard in laptops. The plug-in card effects a transformation of the PCMCIA interface to an IEEE 1284 interface, which is known in the art. The said plug-in card provides a special printer interface, which is expanded with respect to the data transfer rate and which delivers a data transfer rate of approximately 2 MB/s, as compared to a rate of approximately 1 MB/s for known printer interfaces. The known interface device generally consists of a driver component, a digital signal processor, a buffer and a hardware module, which terminates in a connector, to which the device, the data of which are to be acquired, is connected. The driver component is connected directly to the expanded printer interface, as a result of which the known interface device establishes a connection between a computer and the device, the data of which are to be acquired.

In order to work with the said interface, an interface specific driver has to be installed in the host device, so that the host device can communicate with the digital signal processor of the interface card. As already stated above, the driver has to be installed on the host device. If the driver is a driver that is developed specifically for the host device, it is, indeed, possible to achieve a high data transfer rate, but the driver cannot be easily installed on a different host system. However, if the driver is a general driver, which is as flexible as possible and which can be used on many host devices, then compromises must be accepted with respect to the data transfer rate.

Especially in the case of an application for multi-tasking systems, in which several different tasks, such as, for example, data acquisition, data display and editing, are to be performed more or less simultaneously, each task is usually assigned a certain priority by the host system. A driver, which supports a specific task, asks in the central processing system of the host device, whether it may have processor resources in order to do its task. Depending on the respective priority assignment method and depending on the implementation of the driver, a specific task is allotted a certain percentage of the processor resources in defined time slots. There is a conflict, if one or more drivers are implemented in such a way that they have the highest priority by default. That is, they are incompatible, as is the case in many practical applications. Hence, it may occur that both drivers are set to have the highest priority, an aspect that in the worst case may even result in a system crash.

The object of the present invention to provide an interface device for communication between a host device and a data transmit/receive device in such a way that said interface device can be used independently of the host device and that makes possible a high data transfer rate.

This engineering object is achieved by means of an interface device in accordance with claim 1 as well as by means of a method in accordance with claim 12.

The present invention is based on the finding that both a high data transfer rate and a host device-independent usability can be achieved if an input/output interface of the host device is used. Said input/output interface is usually present in most commercially available host devices. Input/output interfaces, which are found in practically all host devices, are, for example, hard disks interfaces, graphics interfaces or printer interfaces. Since, however, the hard disk interfaces in the common host device that can be, for example, IBM PCs, IBM-compatible PCs, Commodore PCs, Apple computers or even workstations, are the interfaces with the fastest data transfer rate, the hard disk interface is used in the preferred exemplary embodiment of the interface device of the present invention. However, other storage interfaces, such as, for example, floppy disk drives, CD-ROM drives or tape drives, could also be used, in order to implement the interface device in accordance with the present invention.

The interface device, according to the present invention, comprises a processor unit, a memory unit, a first connecting device for interfacing the host device with the interface device, and a second connecting device for interfacing the interface device with the data transmit/receive device. The interface device is configured by means of the processor unit and the memory unit in such a way that the interface device, when receiving an inquiry from the host device through the first connecting device as to the type of a device that is connected to the host device, sends, irrespective of the type of the data transmit/receive device, to the host device through the first connecting device a signal that signals to the host device that it is communicating with an input/output device. Thus, the interface system, according to the present invention, simulates in terms of both the hardware and the software the way, in which a conventional input/output device functions, preferably that of a hard disk drive. Since the support of hard disks is implemented, according to the standards, in all commercially available host systems, the simulation of a hard disk, for example, can achieve independence from the host system that is used. Therefore, the interface device of the present invention no longer communicates with the host device or computer by means of a specially designed driver, but rather by means of a program, which is present in the BIOS system (Basic Input/Output System) and which is usually adapted precisely to the specific computer system, on which it is installed. Consequently the interface device, according to the present invention, combines the advantages of both groups. On the one hand, the data communication between the computer and the interface takes place by means of a host device-specific BIOS program, which could be regarded as a "device-specific driver". On the other hand, the BIOS program, which operates one of the common input/output interfaces in host systems, is simply present in all host systems, so that the interface device, according to the present invention, is host device-independent.

Preferred exemplary embodiments of the present invention will be explained in more detail below with reference to the accompanying drawings. The drawings show in:

Figure 1 a general block diagram of the interface device according to the present invention; and

Figure 2 a detailed block diagram of an interface device according to a preferred exemplary embodiment of the present invention.

Figure 1 shows a general block diagram of an interface device **10**, according to the present invention.

A first connecting device **12** of the interface device **10** can be connected to a host device (not shown) by means of a host line **11**. The first connecting device is connected to both a digital signal processor **13** and to a memory **14**. Furthermore, the digital signal processor **13** and the memory **14** are connected to a second connecting device **15** by means of bi-directional communication lines (shown by means of two directional arrows for all lines). The second connecting device can be coupled by means of an output line **16** to a data transmit/receive device, which is supposed to receive data from the host device or from which data are to be read, i.e. acquired, and transferred to the host device.

Communication with the host system or host device is based on known standard access commands, as supported by all known operating systems (for example, DOS, Windows, Unix). Preferably the interface device, according to the present invention, simulates a hard disk with a root directory, the entries of which are "virtual" files, which can be created for a wide variety of functions. When the host device system, to which the interface device, according to the present invention, is connected, is booted, and a transmit/receive device is also connected to the interface device **10**, typical BIOS routines issue an instruction, which is known by those skilled in the art as the "INQUIRY" instruction, to each input/output interface that is present in the host device. The digital signal processor **13** will receive this inquiry by way of the first connecting device and will generate a signal, which is sent to the host device (not shown) again by way of the first connecting device **12** and the host line **11**. This signal signals to the host device that, for example, a hard disk drive is connected at the respective interface, to which the INQUIRY instruction was sent. Optionally the host device can send an instruction, known by those skilled in the art as "Test Unit Ready", to the interface device that requests more precise details regarding the queried device.

Irrespective of which transmit/receive device at the output line **16** is connected to the second connecting device, the digital signal processor **13** informs the host device that the host device is communicating with a hard disk drive. If the host device receives the response that a drive is present, then it will now send the request to the interface device **10** to read the boot sequence, which in the case of actual hard disks is usually found in the first sectors of the disk. The digital signal processor **13**, the operating system of which is stored in the memory unit **14**, will respond to this instruction by sending to the host device a virtual boot sequence, which in the case of actual drives includes the type, the starting position and the length of the file allocation table (FAT), the number of sectors, etc., as known to those skilled in the art.

When the host device has received these data, it assumes that the interface device **10**, according to a preferred exemplary embodiment of the present invention, is a hard disk drive. In reply to an instruction from the host device to display the directory of the "virtual" hard disk drive, which is simulated by the interface device **10** with respect to the host device, the digital signal processor can respond to the host device in exactly the same way as a conventional hard disk would, namely by reading, upon request, the file allocation table or FAT on a sector, specified in the boot sequence, which is generally the first writable sector, and by transferring it to the host device. Furthermore, it is possible that the FAT is not read until immediately prior to reading or storing the data of the "virtual" hard disk and not already at initialization.

In a preferred exemplary embodiment of the present invention, the digital signal processor **13**, which does not necessarily have to be implemented as a digital signal processor, but rather may be any other kind of microprocessor, comprises a first and a second command interpreter. The first command interpreter carries out the steps, described above, while the second command interpreter carries out the read/write assignment to specific functions. If at this point the user wishes to read data from the transmit/receive device over the line **16**, then the host device sends a command, which, for example could read "read file xy", to the interface device. As already stated above, the interface device appears to the host device as a hard disk. At this point the second command interpreter of the digital signal processor interprets the read command of the host processor as a data transfer command, by decoding whether "xy" denotes, for example, a "real-time input" data file, a "configuration" data file or an executable data file, where in this case the second command interpreter begins to transfer data from the transmit/receive device by way of the second connecting device to the first connecting device and over the line **11** to the host device.

Preferably the volume of data to be acquired by a data transmit/receive device is specified in a configuration file, described in the following, by the user specifying in the said configuration file that a measurement is to last, for example, five minutes. To the host device the "real-time input" data file then appears as a file having a length that corresponds to the volume of data anticipated in the five minutes. Those skilled in the art know that communication between a processor and a hard disk consists of the processor transferring to the hard disk the numbers of the blocks or clusters or sectors whose contents it wishes to read. From the FAT the processor knows which information is contained in which block. Therefore, in this scenario the communication between the host device and the interface device of the present invention consists of the very fast transfer of block numbers and preferably of block number ranges, because a "virtual" "real-time input" file will not be fragmented. If at this point the host device now wants to read the "real-time input" file, it transfers a range of block numbers to the interface device, whereupon the process is started that data are received by way of the second connecting device and are sent to the host device by way of the first connecting device.

In addition to the instruction memory for the digital signal processor, where in this case said memory comprises the operating system of the digital signal processor and can be implemented as an EPROM or EEPROM, the memory unit **14** can have an additional buffer, which is used for purposes of synchronizing the data transfer from the transmit/receive device to the interface device **10** and the data transfer from the interface device **10** to the host device.

Preferably the buffer is implemented as a fast random access memory or RAM buffer.

Furthermore, the user can also create from the host device a configuration file, the entries of which automatically set and control various functions of the interface device **10**, on the interface device **10**, which appears to the host device as a hard disk. These settings can be, for example, gain, multiplex or sampling rate settings. By creating and editing a configuration file, which is usually a text file, which is simple to understand with little prior knowledge, the user of the interface device **10** can perform essentially the same operator actions for almost any transmit/receive device, which can be coupled by means of the line **16** to the second connecting device. As a result, a source of errors that are generated from the fact that a user has to know many different command codes for different applications, is eliminated. In the case of the interface device **10**, according to the present invention, it is necessary that the user note the conventions of the configuration file just once in order to be able to use the interface device **10** as an interface between a host device and almost any transmit/receive device.

The option of storing any file in agreed upon formats in the memory unit **14** of the interface device **10**, taking into consideration the maximum capacity of the memory unit, makes it possible to implement any expansions or even completely new functions of the interface device **10** without any loss of time. Even files that can be executed by the host device, such as, for example, batch files or executable files (BAT or EXE files), and also help files can be implemented in the interface device, thus achieving independence of the interface device **10** from any additional software (with the exception of the BIOS routines) of the host device. On the one hand, this feature avoids licensing and/or registration problems; and, on the other hand, installation of certain routines, which can often be used, such as, for example, an FFT routine, in order to be able to look at acquired time-domain data in the frequency range, is no longer necessary, since the EXE files are already installed on the interface device **10** and appear in the virtual root directory, by means of which the host device can access all programs that are stored on the interface device **10**.

In a preferred exemplary embodiment of the present invention, in which the interface device **10** simulates a hard disk drive to the host device, the interface device is automatically detected and readied for operation as soon as the host system is powered up or booted. This corresponds to the currently increasing and widespread "plug-and-play" standard. The user needs no longer to be concerned about installing the interface device **10** on the host device by means of specific drivers, which have to be loaded, but rather the interface device **10** is automatically readied for operation when the host system is booted.

However, it is obvious to those skilled in the art that the interface device **10** is not necessarily signed on when the computer system is turned on, but that a special BIOS routine can also be started on the host device when the computer is running, in order to sign on or "mount" the interface device **10** as an

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.