19 **FEDERAL REPUBLIC OF GERMANY**

12 **Patent Application**
10 **DE 197 08 755 A 1**

Int. Cl.[6]
G 06 F 13/12

[stamp:]

GERMAN
PATENT OFFICE

21 File no.: 197 08 755.8
22 Application date: 03/04/1997
43 Patent app. date: 09/17/1998

71 Applicant:
Tasler, Michael, 63773 Goldbach, Germany

72 Inventor:
Same as applicant

74 Representative

Schoppe, F., Dipl.-Ing.Univ., Pat.-Anw.
(patent lawyer), 81479 Munich, Germany

56 Documents cited:

EP 06 85 799 A
IBM TDB 38, 5, 249;
National Instruments IEEE 488 and VXI bis Control.
Catalog 1994, see 3-188-3-122;

The following information is taken from the documents submitted by the applicant

Request for examination as per Sec. 44 PatG (Patent Law) is submitted

54 Flexible interface

57 An interface device supplies fast data communication between a host device with input/output interfaces and a unit that sends/receives data with which the interface device is furnished with devices for processing, storage, and initial coupling with the interface device and a second coupling device for linking the interface device with the unit for sending/receiving data. The interface device must be configured by the processor and storage device in a way that the when the host device makes a request via the first coupling device, which affects the type of device connected to the same one. The interface device sends a signal via the first coupling device to the host device regardless of the type of data sending/receiving unit, which signals to the host device that it is communicating with the input/output device.

DE 197 08 755 A 1

Description

The present invention concerns the transfer of data, and in particular, to interface devices for communicating between a computer or host device and a data sending/receiving unit from which data shall be gathered and/or with which data shall be communicated.

Previous data collection systems for computers are extremely limited in their range of use. In general, the same systems can be classified into two groups.

With the first group, host devices or computer systems are connected to the device using an interface whose data shall be gathered. The interfaces of this group are normally standard interfaces that can be used with special driver software for different host systems. An advantage of these interface devices is the fact that they are widely independent of the host device. However, the disadvantage is that they require very complex drivers that are prone to malfunction and limit the data transfer rates between the device connected with the interface and the host device and vice versa. Moreover, implementing these interfaces for portable systems is very difficult in some cases and customization options are few which is why these systems have low flexibility.

The devices from which data must be gathered fill the entire bandwidth of the electrical engineering. It can be assumed with a typical scenario that a customer, for example in the medical technology sector who operates an x-ray diagnostic device, reports an error. A service representative of the device manufacturer then goes to the customer and reads the system log files created by the x-ray diagnostic device using, for example, a portable computer or laptop. If the error cannot be localized or if an error only occurs sporadically, it will be necessary for the service representative not only to have to read an error log file but also data from the on-going operation. It is obvious that fast data transfer as well as fast data analysis is necessary.

In another case for using an interface, as an example, an electronic measuring device (e.g., a multimeter) can be connected to the computer system to transfer measured data from the multimeter to the computer. For long-term measurements in particular or when there is a large volume of data, it is necessary for the interface to allow a high rate of data transfer.

From these coincidentally selected examples, it can be seen that the application of an interface can be completely different from each other. Therefore it is desirable that an interface be so flexible that very different electrical or electronic systems can be linked to a host device using an interface. In order to prevent operating errors, it is also desirable that a service representative must not operate different interfaces in different ways for every different application rather the representative should operate them in a way that one universal interface service is created for as large a number of applications as possible.

In order to increase the data transfer rates via an interface, the decision was made for the second group of interface devices to individually adapt the interface to individual host systems or computer systems heavily. The advantage of this solution is that high transfer rates are possible, but the disadvantage is that the driver for the interfaces of the second group are heavily adapted to one individual host system which is why they cannot be used in general or are only very ineffectual for other host systems. Moreover, these types of interfaces have the disadvantage of having to be installed in the computer housing since they access the internal host bus system in order to obtain maximum data transfer rates. They are therefore not generally suitable for portable host systems (laptops) that do not have free inner volume to insert an interface card due to their size being designed as small as possible.

Interface devices manufactured by IOtech (business address: 25971 Cannon Road, Cleveland, Ohio 44146, USA), which is suitable for laptops (e.g., the WaveBook/512 model (registered trademark)), provides a solution for this problem. The interface devices are connected with the PCMCIA interface using a plug-in card – approximately the size of a debit card – which are in the meantime provided on laptops as standard. The plug-in card effects a transformation of the PCMCIA interface for an IEEE 1284 interface that is recognized in technology. The mentioned plug-in card establishes a special printer interface enhanced with regards to the data rate, which delivers a data transfer rate of approximately 2 MB/s compared to a rate of 1 MB/s with known printer interfaces. The recognized interface device usually comprises a driver module, a digital signal processor, a buffer, and a hardware assembly, which joins in a connector to which the device is connected and whose data is gathered. The driver assembly is directly linked to the enhanced printer interface whereby the recognized interface device establishes a connection between a computer and the device whose data shall be gathered.

In order to operate the mentioned interface, a driver specific to the interface must be installed in the host device so that the host device can communicate with the digital signal processor of the interface card. As mentioned before, the driver must be installed on the host device. If the driver is specially designed for the host device, fast data transfer is namely made possible, but the driver cannot be readily installed on another host system. However, if the driver is a general driver, which is as flexible as possible, and can be used for many host devices, compromises must then be accepted in terms of the data transfer rate.

Especially with an application for multi-tasking systems for which several different tasks, e.g., data collection, data representation, or editing must essentially be processed at the same time, the host system usually assigns each task a certain priority. A driver that supports a special task requests in the central processing system of the host device if it can have processor resources to perform the task. Depending on the respective priority allocation process and depending on the implementation of the driver, a special task will receive a certain proportion of the process resources in certain time slots. Conflicts then result if one or more drivers are implemented in a way that they have the highest priority by default; i.e., that they are incompatible as is the case with many applications in practice. It may happen that both drivers being used have the highest priority, which can even lead to a system crash in the worst-case scenario.

The task of the present invention is to create an interface device for communication between a host device and a data sending/receiving unit, which can be used independent of the host device allowing a high rate of data transfer.

This will be achieved by an interface device as per Claim 1 and by the procedure as per Claim 12.

The present invention is based on knowledge that a high rate of data transfer and application independent of the host device can be achieved if it is used on an input/output interface of the host device, which is normally available in the host devices available on the market usually. Input/output interfaces that are practically available in every host device, are for example, hard disk interfaces, graphic interfaces, or printer interfaces. However, since the hard disk interfaces for common host devices, which can, for example, be IBM PCs, IBM-compatible PCs, Commodore PCs, Apple computers or workstations are interfaces with the fastest data transfer rate, the present invention is used on the hard disk interface with the preferred design example of the interface device. This could also be used on other storage interfaces, e.g., disk drives, CD-ROM drives, or tape drives in order to implement the interface device as per the present invention.

The interface device as per the present invention comprises a processor, a storage device, a first coupling device for linking the host device via the interface to the interface device and a second coupling device for linking the interface device via the interface to the data sending/receiving unit. The interface device is configured by the processor and the storage device so that the interface device in the case of a request from the host device via the first coupling device, which affects the type of device that is linked to the host device, sends a signal via the first coupling device to the host device regardless of the type of data sending/receiving unit. This signals to the host device that it is communicating with an input/output device. Therefore, the interface system as per the present invention simulates hardware and the function, in terms of software, of a common input/output device and preferably of a hard disk drive. Since the support of hard disks is implemented in all available host systems as standard, the simulation of a hard disk can, for example, achieve independence from the host system used. Thus, the interface device no longer communicates – according to the invention – with the host device or computer via a specially designed driver rather via a program available in the BIOS (BIOS = Basic Input/Output System), which is usually adjusted precisely on the special computer system on which it is installed. As a result, the interface device as per the present invention combines the advantages of both groups. On the one hand, the data communication between the computer and the interface takes place via a BIOS program specific to the host device, which could be considered the “device-specific driver.” On the other hand, the BIOS program, which operates one of the common input/output interfaces, is even available in each host system which is why the interface device as per the present invention is independent of the host system

The preferred design examples of the present invention are explained in detail in the following with reference to the enclosed drawings. It shows the following:

Fig. 1 A basic block diagram of the interface device as per the present invention

Fig. 2 A detailed block diagram of an interface device as per a preferred design example of the present invention

Fig. 1 shows a basic block diagram of an interface device **10** as per the present invention.

Via a host line **11**, an initial coupling device **12** of the interface device **10** can be linked to a host device (not shown). The first coupling device is connected to a digital signal processor **13** and a storage device **14**. The digital signal processor **13** and the storage device **14** are also coupled with a second coupling device **15** using bi-directional communication lines (for all lines shown by two arrows). By means of an output line **16**, the second coupling device can be coupled with a data sending/receiving unit, which shall receive data from the host device or read by the data; i.e., collected, and shall be transferred to the host device.

Communication with the host system or host device is based on recognized standard access commands as they are supported by all known operating systems (e.g., DOS, Windows, UNIX). Preferably, the interface device as per the present invention simulates a hard disk with a root directory whose entries are “virtual” files that can be created for the widest variety of functions. If the host device system with which the interface device as per the present invention is connected for which a data sending/receiving unit is also linked to the interface device **10**, is booted, normal BIOS routines output a command to each input/output interface available in the host device that is recognized among experts as an “INQUIRY” command. Via the first coupling device, the digital processor **13** receives this request and generates a signal, which in turn is sent via the first coupling device **12** and the host line **11** to the host device (not shown). This signal indicates to the host device that a hard disk drive is connected to the relevant interface to which the INQUIRY command was sent. As an option, the host device can send a “Test Unit Ready” command that is recognized by experts to the interface device, which requires more precise details with regards to the requested device.

Regardless of which data sending/receiving unit is connected with the second coupling device to the output line **16**, the digital signal processor **13** communicates to the host device that the host device is communicating with a hard disk drive. If the host device receives a response that a drive is available, it will now send the request to the interface device **10** to read the boot sequence that is normally located in the first sectors themselves with physical disk drives. The digital signal processor **13** whose operating system is stored in the storage device **14** will respond to this command by it sending a virtual boot sequence for the host device, which, with physical drives, contains it, the initial position, and the length of the FAT (FAT = File Allocation Table), the number of sectors, etc. This is recognized by experts. If the host device received this data, it is assumed that the interface device **10** as per a preferred design example of the present invention is a hard disk drive.

Based on a command by the host device, the directory of the "virtual" hard disk drive, which is simulated by the interface device **10** to the host device, the digital signal processor responds to the host device exactly as a conventional hard disk would respond; namely, the file position table or FAT on a sector specified in the boot sequence is read, which is normally the first writable sector, and transferred to the host device. It is also possible that the FAT is first directly read before data of the "virtual" hard disk is read or stored and not already when initializing.

With a preferred design example of the present invention, the digital signal processor **13**, which can be designed not necessarily as a digital signal processor but also any other microprocessor, comprises one initial and one secondary command interpreter. The first command interpreter carries out precisely stated steps while the second command interpreter performs the read/write allocation for certain functions. If the user now wants to read the data from the data sending/receiving unit via the line **16**, the host device sends a command to the interface device, which could read "Read file xy" for example. As previously mentioned, the interface device appears as a hard disk to the host device. The second interpreting device of the digital signal processor now interprets the read command of the host processor by decrypting whether "xy", for example, designates a "real-time entry" file, "configuration" file, or an executable file. As a data transfer command, for which it starts by itself, data transfers from the data sending/receiving unit via the second coupling device to the first coupling device and via the line **11** to the host device.

Preferably, the volume of data collected by a data sending/receiving unit is specified in the configuration file described below while the user specifies in the configuration file that a measurement shall extend, for example, over five minutes. For the host device, the "real-time entry" file will appear as a file whose length corresponds to the data volume expected in the five minutes. It is recognized by experts that communication between a processor and a hard disk takes place by the processor communicating the hard disk numbers from the blocks or clusters or sectors whose content it wants to read. The processor knows from the FAT which information is in which block. Communication from the host device to the interface device of the present invention therefore, with this scenario in rapid transfer, consists of block numbers and preferably block number ranges because a "virtual" "real-time entry" file will not be fragmented. If the host device now wants to read the "real-time entry" file, it communicates a range of block numbers for the interface device whereupon as a result a process starts where data is received via the second coupling device and sent via the first coupling device to the host device.

Aside from the command storage for the digital signal processor, which can comprise the operating system itself and can be designed as EPROM or EEPROM, the storage device **14** can have an additional buffer, which serves for synchronization purposes between the data transfer from the sending/receiving unit for the interface device **10** and the data transfer from the interface device **10** to the host device.

Preferably, the buffer is designed as a faster direct access storage unit or RAM buffer.

The user can also create a configuration file from the host device to the interface device, which appears as a hard disk to the host device. The entries of the configuration file set and control various functions of the interface device **10**. These can be, for example, settings for strengthening, multiplex, or scanning. By creating and editing a configuration file, which can be easily understood without much previous knowledge, the user of the interface device **10** can perform essentially the same operation for nearly any sending/receiving units, which can be coupled via the line **16** with the second coupling device. By doing so, a source of errors is eliminated, which would make it necessary for the user to know many different command codes for various applications. With the interface device **10** as per the present invention, it is only necessary that the user make a note of the conventions of the configuration file one time. Afterward, the user can use the interface device **10** as an interface between a host device and nearly any sending/receiving unit.

By means of the option to save any files in agreed formats taking into consideration the maximum storage capacity on the interface device **10** in the storage device **14**, any enhancements or even entirely new functions of the interface device **10** must be realized without losing time. Even files that can be executed by the host device, e.g., batch files or executable files (BAT files or EXE files) or even help files can be implemented in the interface device and therefore can be achieved independent of the interface device **10** of any additional software (irrespective of the BIOS routines) of the host device. This prevents licensing and/or log-in problems. On the other hand, installations of certain routines that can often be used, e.g., an FFT routine to be able to consider collected time domain data in the frequency range become invalid. This is because these EXE files are already installed on the interface device **10** and appear in the virtual root directory through which the host device can access any programs stored on the interface device **10**.

With a preferred design example of the present invention with which the interface device **10** simulates a hard disk drive to the host device, the same is already automatically recognized when switching on or booting the host system. This corresponds to the currently expanding "plug-and-play" standard. The user no longer has to deal with the installation of the interface device **10** on the host device by loading special drivers rather the interface device **10** is automatically made available when booting the host system.

However, it is obvious for experts that the interface device **10** is not necessarily registered when switching on the computer rather that a special BIOS routine can be started on the host device also while the computer runs in order to connect or "mount" the interface device **10** as an additional hard disk. This design example is suitable for larger workstation systems which essentially are never switched off because they are performing in a "multi-tasking" environment, e.g., e-mail functions or process monitoring, which are continuously running.

In the interface unit according to this invention, an enormous advantage of separating the actual hardware required for connecting the interface device 10 to the sending/ receiving device, as is apparent from the embodiment described in the following, from the communication unit, which is implemented as the digital signal processor 13, the storage 14 and the first connecting device 12, consists of being able to operate the most different unit types parallel to each other in an identical manner. Therefore, many interface devices 10 can be connected to a host unit, which will then recognize the most different, in a manner of speaking "virtual" hard drives. For another, a possible modification of the special hardware, which is symbolized by the second connecting device 15, can essentially be implemented without changing the operation of the interface unit according to this invention. Furthermore, a user can intervene at any time to any desired extent in the existing second connecting device by using the above mentioned option of creating a configuration file or by adding or saving new program components for the second connecting device.

A significant advantage of the interface device 10 of this invention also consists of it enabling extremely high data transfer rates and this already by using the host unit's own BIOS routines, which the manufacturer of the host unit or BIOS system has optimized for each host unit, for exchanging data. Furthermore, the data are administered and made available due to simulating a virtual bulk memory so they can be transferred directly, in a manner of speaking, to other storage media, e.g., an actual hard drive of the host unit, without processor intervention by the host unit. The only limitation on a long-term data transfer with high speed is, therefore, caused by the speed and storage capacity of the bulk memory of the host system. This is so because the digital signal processor 13 already formats the data read in over the second connecting device 15 of the sending/ receiving device into block sizes suitable for a hard drive of the host unit, as a result of which the data transfer speed is merely limited by the mechanical sluggishness of the hard drive system of the host unit. It will be noted here that a data flow from a host unit must normally be formatted in blocks to be written on a hard drive and subsequently be retrieved, as is known to persons skilled in the art.

This data transfer rate can be increased even more by setting up a direct memory access (DMA; DMA = Direct Memory Access) or RAM drive in the host system. However, as persons skilled in the art know, setting up a RAM drive requires processor resources of the host unit, which is why the advantage due to the data being written on the hard drive of the host unit and essentially requiring no processor resources is lost.

As already noted, a data buffer can be implemented in the memory 14 that enables independence in timing of the sending/ receiving device, which is coupled to the second connecting device, from the host unit, which is coupled to the first connecting device. In this way, a flawless operation of the interface device 10 is ensured even for time-critical applications and even in multitasking host systems.

Fig. 2 shows a detailed block diagram of an interface device 10 according to this invention.

A digital signal processor (DSP) 1300 constitutes, in a manner of speaking, the heart of the interface device 10. The DSP can be any DSP but it is preferred that it has an on-chip direct access memory (RAM) of 20 KB. In the direct access memory, which is already integrated on the DSP, certain instruction sets, for example, can be stored. An 80-MHz timing component 1320 for timing the DSP is connected to the DSP 1300. The DSP implements a rapid Fourier transformation (FFT) in real-time and an optional data compression of data to be transferred from the sending/ receiving device to the host unit to achieve greater efficiency and to be able to cooperate with host units that have smaller memory devices.

The first connecting device 12 in Fig. 1 includes the following components for the preferred embodiment of the interface device 10 shown in Fig. 2: an SCSI interface 1220 and a 50-pin SCSI connector 1240 for connecting with an SCSI interface present in most host units or laptops. The SCSI interface (SCSI = Small Computer System Interface) 1220 transforms the data received via the SCSI connector 1240 into data comprehensible to the DSP 1300, as is known to persons skilled in the art. The first connecting unit 12 also includes an EPP with a data rate of about 1 MB/s (EPP = Enhanced Parallel Port) for a moderate data transfer rate of 1 MB/s, as compared with a data rate of 10 MB/s of the SCSI. The EPP 1260 is connected to a 25-pin sub-D connector 1280 to enable connecting, for example, to a printer interface of a host unit. Optionally, the first connecting device 12 also comprises a 25-pin connector 1282 that enables the connection of 8 digital outputs and 8 digital inputs 1284 to a host unit.

The second connecting device preferably comprises 8 BNC inputs with calibration circuits 1505, a block 1510 with 8 unit amplifiers with a surge protection of ±75 V, where this block is connected in turn with 8 sample/ hold elements 1515 (sample/ hold = S&H). The calibration circuits are circuits that permit controlled switching between a metering voltage and a calibration reference voltage. Each sample/ hold device is connected to an input of an 8-channel multiplexer 1520, that feeds its output signals via a programmable amplifier 1525 in an analog/digital converter (ADC) with 12 bit and 1.25 MHz 1530 to the DSP 1300. The ADC 1530 is controlled by means of a 20-bit timer 1535, as is known to persons skilled in the art. The programmable amplifier 1525 and the 8-channel multiplexer 1520 are controlled via an selectable amplifying-channel component 1540 that in turn is controlled by the DSP 1300.

The entire interface device 10 is supplied by an external AC/DC converter 1800 that provides a digital power supply of +5 V and is connected to a DCDC converter 1810 that can deliver the analog power supply voltages of ±5 V und ±15 V that are required for the interface device 10. The DCDC converter also controls a precision voltage reference 1820 that controls both the 8-BNC inputs 1505 and the ADC 1530 as well as a digital/ analog converter (DAC) 1830, which over an output amplifier block with 4 output amplifiers 1840 and a 9-pin converter 1850 enables analog output directly from the DSP 1300 to an output device that can be connected to the 9-pin connector 1850, such as a printer device or a monitor device, as a result of which data monitoring of the data transferred to the host unit is optionally enabled or, for example, without using processor time of the host unit, an FFT can be considered for achieving fast and comprehensive data analysis.

The memory device 14 in Fig. 1 is implemented in Fig. 2 by an EPROM 1400 that holds in a preferred embodiment of this

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.