ACCESS CONTROL SYSTEM

The present invention relates to an access control system, in particular to a method of providing access by a program to at least one component of a computer system, more particularly of a computer system used in a receiver/decoder of broadcast digital signals.

The advent of digital transmission systems intended primarily for broadcasting television signals, in particular but not exclusively satellite television systems, has opened up the possibility of using such systems for other purposes. One of these is to provide interactivity with the end user.

One way of doing this is to run an application on the receiver/decoder through which the television signal is received. The code for the application could be permanently stored in the receiver/decoder. However, this would be rather limiting. Preferably, the receiver/decoder should be able to download the code for a required application. In this way, more variety may be provided, and applications can be updated as required without any action on the part of the user.

Applications for execution by a receiver/decoder are generated by various broadcast suppliers. Typically, the receiver/decoder has a number of interfaces, such as a serial interface and parallel interface, for connection to external units. Device drivers for these interfaces are supplied by the manufacturer of the receiver/decoder. With multiple sources of applications and multiple manufacturing sources of receiver/decoder, it is important that one application behaves in the same way on every receiver/decoder, and each receiver/decoder should execute every application in the same, correct manner.

Accordingly, in a first aspect the present invention provides a method of providing access by a program to at least one component of a computer system, said method comprising the steps of:

storing a logical device associated with the or each respective component in

said system, the or each logical device having a respective device identifier;

assigning a program identifier to the program; and

outputting from the program a signal comprising the program identifier and the device identifier of the or each logical device to a common interface between the program and the or each logical device to create a communication channel between said program and the or each logical device via said interface.

By means of the above, a common interface is provided between the various programs in the computer system and the "hardware" of the receiver/decoder, irrespective of the sources of the programs and the hardware.

The signal may further comprise a command enabling receipt by the program of a message output from a respective logical device indicating a change of state of the component associated with the device.

Thus, a program may be alerted to an "unexpected event" occurring at one of the components, for example, the receipt of a message by a serial interface of the computer system or the insertion of a smartcard in a smartcard reader of the computer system.

This message may be stored temporarily in queue means of said computer system for subsequent transfer to said program. In other words, the program may inform the logical device that it need not be supplied immediately with the message.

Preferably, the queue means comprises a plurality of queues, each queue having a respective priority level indicative of the order in which messages are to be transferred from the queue means to the program, and said signal further comprises the priority level of the queue in which said message is to be stored temporarily.

Thus, the program may specify the "urgency" with which the message is to be dealt with by the queue means by indicating a priority level with which the queue means is to deal with the message.

In a second aspect, the present invention provides a method of providing access by a plurality of programs to at least one component of a computer system using the method described above, each program accessing the or each component via said interface.

Thus, the present invention provides a single common interface for communication between a program and one or more logical devices, and for communication between a logical device and one or more programs.

A third aspect of the present invention provides a method of transmitting data between a program and a component of a computer system; said method comprising the steps of

providing access by the program to the component using the method described above; and

subsequently outputting from the program a signal comprising the program identifier, the device identifier of the logical device to a common interface, a command instructing operation of a component by the logical device associated therewith, the address of data to be input to said component by said device and the address for data output by said component to said logical device to said common interface.

In a fourth aspect, the present invention provides a method of transmitting data between a program and a component of a computer system; said method comprising the steps of

providing access by the program to the component using the method described above; and

subsequently outputting from the program a signal comprising the program identifier, the device identifier of the logical device to a common interface, a command instructing operation of a component by the logical device associated therewith, the priority level of the queue in which a message output from the respective logical device indicating a change of state of the component associated with the device is to be stored temporarily prior to transfer to the program, the address of

data to be input to said component by said device and the address for data output by said component to said logical device to said common interface.

Therefore, once access between the program and the component has been provided, the thus established communication route may be used to provide a route for transmitting data from the program to a component and vice versa.

The program and/or the or each logical device may be input to said computer system via a component. This provides for convenient downloading and "up-dating" of the program and the logical device in the computer system.

Preferably, the or each component comprises at least one of an MPEG flow tuner, a serial interface, a parallel interface, a modem and a smartcard reader.

In a fifth aspect, the present invention provides apparatus for providing access by a program to at least one component of a computer system, a logical device associated with the or each respective component being stored in said system, the or each logical device having a respective device identifier; said apparatus comprising:

means for assigning a program identifier to the program; and

means for creating a communication channel between said program and the or each logical device in accordance with a signal output from the program and comprising the program identifier and the device identifier of the or each logical device

The apparatus may further comprise means for enabling receipt by the program of a message output from a respective logical device indicating a change of state of the component associated with the device.

Preferably, the computer system further comprises queue means for storing temporarily a message output by said device for subsequent transfer to said program. The queue means may comprise a plurality of queues, each queue having a respective priority level indicative of the order in which messages are to be transferred from the queue

means to the program.

The computer system may further comprise means for storing data input to said component by said device and data output by said component to said logical device to said common interface.

Preferably, the or each logical component is connected to the respective associated component via a device driver.

In a sixth aspect, the present invention provides a receiver/decoder for receiving broadcast signals, said receiver/decoder comprising apparatus as described above.

Preferably, the receiver/decoder further comprises means for receiving a compressed MPEG-type signal, means for decoding the received signal to provide a television signal and means for supplying the television signal to a television.

Preferred features of the present invention will now be described, purely by way of example, with reference to the accompanying drawings, in which:-

Figure 1 shows the overall architecture of a digital television system according to the preferred embodiment of the present invention;

Figure 2 shows the architecture of an interactive system of the digital television system;

Figure 3 shows the arrangement of files within a module downloaded into the memory of an interactive receiver/decoder;

Figure 4 shows the arrangement of memory volumes of the memory of the interactive receiver/decoder;

Figure 5 is a schematic diagram of interfaces of the receiver/decoder;

Figure 6 shows the architecture of the software in the receiver/decoder; and

Figure 7 shows an example of connections between a device manager, a plurality of clients and a plurality of devices.

An overview of a digital television system 1000 is shown in Figure 1. The invention includes a mostly conventional digital television system 2000 which uses the known MPEG–2 compression system to transmit compressed digital signals. In more detail, MPEG–2 compressor 2002 in a broadcast centre receives a digital signal stream (typically a stream of video signals). The compressor 2002 is connected to a multiplexer and scrambler 2004 by linkage 2006. The multiplexer 2004 receives a plurality of further input signals, assembles one or more transport streams and transmits compressed digital signals to a transmitter 2008 of the broadcast centre via linkage 2010, which can of course take a wide variety of forms including telecom links. The transmitter 2008 transmits electromagnetic signals via uplink 2012 towards a satellite transponder 2014, where they are electronically processed and broadcast via notional downlink 2016 to earth receiver 2018, conventionally in the form of a dish owned or rented by the end user. The signals received by receiver 2018 are transmitted to an integrated receiver/decoder 2020 owned or rented by the end user and connected to the end user's television set 2022. The receiver/decoder 2020 decodes the compressed MPEG–2 signal into a television signal for the television set 2022.

A conditional access system 3000 is connected to the multiplexer 2004 and the receiver/decoder 2020, and is located partly in the broadcast centre and partly in the decoder. It enables the end user to access digital television broadcasts from one or more broadcast suppliers. A smartcard, capable of deciphering messages relating to commercial offers (that is, one or several television programmes sold by the broadcast supplier), can be inserted into the receiver/decoder 2020. Using the decoder 2020 and smartcard, the end user may purchase commercial offers in either a subscription mode or a pay–per–view mode.

An interactive system 4000, also connected to the multiplexer 2004 and the receiver/decoder 2020 and again located partly in the broadcast centre and partly in the decoder, enables the end user to interact with various applications via a modemmed back channel 4002.

Figure 2 shows the general architecture of the interactive television system 4000 of the digital television system 1000 of the present invention.

For example, the interactive system 4000 allows an end user to buy items from on-screen catalogues, consult local news and weather maps on demand and play games through his television set.

The interactive system 4000 comprises in overview four main elements:

an authoring tool 4004 at the broadcast centre (or elsewhere) for enabling a broadcast supplier to create, develop, debug and test applications;

an application and data server 4006 the broadcast centre, connected to the authoring tool 4004 for enabling a broadcast supplier to prepare, authenticate and format applications and data for delivery to the multiplexer and scrambler 2004 for insertion into the MPEG-2 transport stream (typically the private section thereof) to be broadcast to the end user;

a virtual machine including a run time engine (RTE) 4008, which is an executable code installed in the receiver/decoder 2020 owned or rented by the end user for enabling an end user to receive, authenticate, decompress, and load applications into the working memory 2024 of the receiver/decoder 2020 for execution. The engine 4008 also runs resident, general-purpose applications. The engine 4008 is independent of the hardware and operating system; and

a modemmed back channel 4002 between the receiver/decoder 2020 and the application and data server 4006 to enable signals instructing the server 4006 to insert data and applications into the MPEG-2 transport stream at the request of the end user.

The interactive television system operates using "applications" which control the functions of the receiver/decoder and various devices contained therein. Applications

are represented in the engine 4008 as "resource files". A "module" is a set of resource files and data. Several modules may be required to make up an application. A "memory volume" of the receiver/decoder is a storage space for modules. An "interface" is used to download modules. Modules may be downloaded into the receiver/decode 2020 from the MPEG-2 transport stream.

The elements mentioned in the previous paragraph are now described in more detail.

For the purposes of this specification, an application is a piece of computer code for controlling high level functions of preferably the receiver/decoder 2020. For example, when the end user positions the focus of a remote controller on a button object seen on the screen of the television set 2022 and presses a validation key, the instruction sequence associated with the button is run.

An interactive application proposes menus and executes commands at the request of the end user and provides data related to the purpose of the application. Applications may be either resident applications, that is, stored in the ROM (or FLASH or other non-volatile memory) of the receiver/decoder 2020, or broadcast and downloaded into the RAM or FLASH memory of the receiver/decoder 2020.

Examples of applications are:-

- An Initiating Application. The receiver/decoder 2020 is equipped with a resident initiating application which is an adaptable collection of modules (this term being defined in more detail hereunder) enabling the receiver/decoder 2020 to be immediately operative in the MPEG-2 environment. The application provides core features which can be modified by the broadcast supplier if required. It also provides an interface between the resident application and downloaded applications.
- A Startup Application. The startup application allows any application, either downloaded or resident, to run on the receiver/decoder 2020. This application acts as a bootstrap executed on arrival of a service in order to start the application. Startup is downloaded into RAM and therefore can be updated

easily. It can be configured so that the interactive applications available on each channel can be selected and run, either immediately after downloading or after preloading. In the case of preloading, the application is loaded into the memory 2024 and is activated by the startup when required.

- A Program Guide. The Program Guide is an interactive application which gives full information about programming. For example, it may give information about, say, one week's television programmes provided on each channel of a digital television bouquet. By depressing a key on the remote controller 2026, the end user accesses an add-on screen, overlaid on the event shown on the screen of the television set 2022. This add-on screen is a browser giving information on the current and next events of each channel of the digital TV bouquet. By depressing another key on the remote controller 2026, the end user accesses an application which displays a list of information on events over one week. The end user can also search and sort events with simple and customised criteria. The end user can also access directly a selected channel.
- A Pay Per View application. The Pay Per View Application is an interactive service available on each PPV channel of the digital TV bouquet in conjunction with the conditional access system 3000. The end user can access the application using a TV guide or channel browser. Additionally, the application starts automatically as soon as a PPV event is detected on the PPV channel. The end user is then able to buy the current event either through his daughter smartcard 3020 or via the communication server 3022 (using a modem, a telephone and DTMF codes, MINITEL or the like). The application may be either resident in the ROM of the receiver/decoder 2020 or downloadable into the RAM of the decoder 2020.
- A PC Download application. On request, an end user can download computer software using the PC download application.
- A Magazine Browser application. The magazine browser application comprises a cyclic video broadcast of images with end user navigation via on-screen buttons.
- A Quiz application. The quiz application is preferably synchronised with a

broadcast quiz programme. As an example, multiple choice questions are displayed on the screen of the television 2022, and the user can select an answer using the remote controller 2026. The quiz application can inform the user whether the answer is correct or not, and can keep count of the user's score.

- A Teleshopping application. In one example of the teleshopping application, offers of goods for sale are transmitted to the receiver/decoder 2020 and displayed on the television 2022. Using the remote controller, the user can select a particular item to buy. The order for the item is sent via the modemmed back channel 4002 to the application and data server 4006 or to a separate sales system the telephone number of which has been downloaded to the receiver/decoder, possibly with an order to debit the account for a credit card which has been inserted into one of the card readers 4036 of the receiver/decoder 2020.
- A Telebanking application. In one example of the telebanking application, the user inserts a bank card into one of the card readers 4036 of the receiver/decoder 2020. The receiver/decoder 2020 dials up the user's bank, using a telephone number stored in the bank card or stored in the receiver/decoder, and then the application provides a number of facilities which can be selected using the remote controller 2026, for example for downloading via the telephone line a statement of account, transferring funds between accounts, requesting a cheque book, etc.
- An Internet Browser application. In one example of the Internet browser application, instructions from the user, such as a request to view a web page having a particular URL, are entered using the remote controller 2026, and these are sent by the modemmed back channel 4002 to the application and data server 4006. The appropriate web page is then included in the transmissions from the broadcast centre, received by the receiver/decoder 2020 via the uplink 2012, transponder 2014 and downlink 2016, and displayed on the television 2022.

Applications are stored in memory locations in the receiver/decoder 2020 and

represented as resource files. The resource files comprise graphic object description unit files, variables block unit files, instruction sequence files, application files and data files.

The graphic object description unit files describe the screens, the man-machine interface of the application. The variables block unit files describe the data structures handled by the application. The instruction sequence files describe the processing operations of the applications. The application files provide the entry points for the applications.

The applications constituted in this way can use data files, such as icon library files, image files, character font files, colour table files and ASCII text files. An interactive application can also obtain on-line data by effecting inputs and/or outputs.

The engine 4008 only loads into its memory those resource files it needs at a given time. These resource files are read from the graphic object description unit files, instruction sequence files and application files; variables block unit files are stored in memory following a call to procedure for loading modules and remain locked there until a specific call to an procedure for unloading modules is made.

With reference to Figure 3, a module 4010, such as a tele-shopping module, is a set of resource files and data comprising the following:

a single application file 4012;

an undetermined number of graphic object description unit files 4014;

an undetermined number of variables block unit files 4016;

an undetermined number of instruction sequence files 4018; and

where appropriate, data files 4020 such as icon library files, image files, character font files, colour table files and ASCII text files.

In the MPEG data stream, each module comprises a group of MPEG tables. Each MPEG table may be formatted as a number of sections. In the MPEG data stream, each section has a "size" of up to 4 kbytes. For data transfer via the serial and

parallel port, for example, modules similarly are split into tables and sections, the size of the section varying with the transport medium.

Modules are transported in the MPEG data stream in the form of data packets of typically 188 bytes within respective types of data stream, for example, video data streams, audio data streams and teletext data streams. Each packet is preceded by a Packet Identifier (PID) of 13 bits, one PID for every packet transported in the MPEG data stream. A programme map table (PMT table) contains a list of the different data streams and defines the contents of each data stream according to the respective PID. A PID may alert a device to the presence of applications in the data stream, the PID being identified using the PMT table.

The concept of modules 4010 together with the concept of downloading small pieces of code allows the easy evolution of applications. They can be downloaded into permanent FLASH memory of the receiver/decoder 2020 as resident software or broadcast in order to be downloaded into the RAM of the decoder 2020 only when needed by the end user.

A memory volume is a storage space for modules 4010. Such storage spaces are located in the memory 2024 of the receiver/decoder 2020. With reference to Figure 4, the memory 2024 is divided into typically a RAM volume 4022, FLASH volume 4024, and ROM volume 4026. The memory may further be divided into memory volumes associated with the various interfaces through which modules are downloaded into the receiver/decoder 2020, for example an MPEG volume for storing modules downloaded from the MPEG bitstream and a serial volume for storing modules received via a serial interface.

The RAM volume 4022 in turn is divided into a zone dedicated to firmware, a working space for the engine 4008 and the buffers. The FLASH and other non-volatile memory can be accessed either by an application or the engine itself through

a device manager.

Each volume contains a list of modules 4010, each module 4010 containing a list of files 4012, 4014, 4016, 4018, 4020. It is possible to have two files bearing the same name and which may be located in distinct modules. For example, a version of the application is typically stored in the ROM volume 4026, with later versions being downloadable into the FLASH volume 4024 to substitute the version stored in the ROM volume with that volume stored in the FLASH volume 4024. The contents of files may be compressed in LZW format, however as decompression of files takes a certain period of time they may be received in decompressed format.

Physical interfaces of the receiver/decoder 2020 are used for downloading data. With reference to Figure 5, the receiver/decoder 2020 contains, for example, six downloading media; MPEG flow tuner 4028, serial interface 4030, parallel interface 4032, modem 4034 and two card readers 4036.

With multiple sources of applications and multiple manufacturing sources of receiver/decoder 2020, it is important that one application behaves in the same way on every receiver/decoder, and each receiver/decoder should execute every application in the same, correct manner. With reference to Figure 6, the receiver/decoder 2020 comprises a run time engine 4008 running under the control of a microprocessor and a common application programming interface 4054. They are installed in every receiver/decoder 2020 so that all receiver/decoders 2020 are identical from the application point of view.

Figure 6 shows the architecture of the receiver/decoder 2020 for running applications 4056. The virtual machine 4007 executes applications 4056, which may comprise applications 4056' coupled directly to the virtual machine or applications 4056" downloaded to the receiver/decoder 2020 from, for example, the MPEG data stream. The run time engine 4008 also displays graphics and text, calls devices for services, receives "events" and uses functions of a library 4058 for specific computation.

With reference to Figure 6, with respect to an application a function of the decoder 2020 is "seen" as a device 4060. There may, therefore, be functions of the receiver/decoder 2020 which may not be seen by any application.

A presentation function communicating with the virtual machine 4007 administers the presentation of text and graphics to the end user, and the presentation of end user actions to the virtual machine 4007. The text and graphics are overlaid on the display on the television set 2022, and the user may interact with the application 4056 by means of a keyboard. The term "keyboard" includes the remote controller 2026.

The run time engine 4008 is an executable code installed in the virtual machine 4007 of the receiver/decoder 2020 and includes a virtual machine for interpreting and running applications. The engine 4008 is adaptable to any operating system, including a single task operating system (such as MS-DOS).

The engine 4008 comprises a process sequencer unit (which take various events such as a key press, to perform various actions) and contains its own scheduler to manage event queues from the different hardware interfaces. It also handles the display of graphics and text.

The engine 4008 comprises a code loader to load and download applications 4056" into the decoder memory 2024. Only the necessary code is loaded into the RAM volume 4022 or FLASH memory volume 4024 in order to ensure optimal use. The downloaded data is verified by an authentication mechanism to prevent any modification of an application 4056 or the execution of any unknown application.

The engine 4008 further comprises a decompressor. As the application code (a form of intermediate code) is compressed for space saving and fast downloading from the MPEG-2 transport stream or via a built-in receiver/decoder mode, the code must be decompressed before loading it into the RAM.

The virtual machine 4007 also comprises an intermediate code interpreter to interpret

the application code and which uses a process sequencer unit to update various variable values and determine status changes, and an error checker.

With reference to Figure 6, with respect to an application a function of the decoder 2020 is "seen" as a device 4060. There may, therefore, be functions of the receiver/decoder 2020 which may not be seen by any application.

A device 4060 comprises a logical device unit which may correspond to a component 4062 or physical interface 4064 of the hardware 4066. Such devices are referred to as "low level devices" 4068. The output of such a device 4068 may be connected to at least one device driver 4070 for converting the logical signals output by the device 4068 into signals required to drive, for example, a hardware interface 4064. Alternatively, the device 4068 may itself drive a component or interface of the receiver/decoder 2020, that is, the output of the device may be connected directly to the hardware 4066.

Examples of low level devices 4068 are described below.

An LCARD device enables a program to communicate with the smartcard contained in one smartcard reader 4036, and an RCARD device enables a program to communicate with the smartcard contained in the other smartcard reader 4036. For example, these devices enable a program to read the state of the card, read the card history and send an input message to the card. The devices also inform a program of the insertion of a card in to the reader, removal of a card from a reader and card reset if not requested by the program. The LCARD and RCARD devices are specific to the protocol used for running the card. Typically an ISO7816 protocol is used.

An SCTV device enables a program to verify and configure of a scart outlet to the television set 2022. For example, this device enables a program to request information about the sound characteristic of the scart outlet, perform a "MUTE" on the sound and dynamically program RGB levels.

A TUNER device enables a program to use the tuner 4028. For example, the device enables a program to perform a scan from either a minimum frequency or a current frequency of the tuner, read the tuner parameters and program the tuner.

A SERIAL device enables a program to communicate with equipment via a serial link and a PARALLEL device enables a program to communicate with equipment via a parallel link. For example, these devices enable a program to send a message via the respective link and inform a program of the reception of a message via that link.

A MODEM device allows the receiver decoder to communicate with a data service via an internal half–duplex modem supporting V23. The MODEM device requests the dialling of a number, the sending of a message to the data server and disconnection of the modem, and signals reception of a message, the detection of errors and the loss or detection of a carrier.

Remote devices, executed in a remote location, can be any of the local devices, except that a port and protocol must be defined.

In addition to "low level devices" the receiver/decoder 2020 may also include "high level devices" 4072 which control operation of the receiver/decoder 2020.

Examples of high level devices 4072 are described below.

An MLOAD device allows an application to load an MPEG section, a complete MPEG table or a group of MPEG sections from an MPEG bitstream corresponding to hardware and software filtering criteria. For example, the device enables a program to download only those sections of a group that are required at any one time by an application

An FDLOAD device manages automatically file downloading via the serial and/or parallel ports. The device may signal both the start and the end of the downloading of files via the serial and/or parallel ports as requested by a program. The device may

issue a request to the program for an index table which includes the PID of the file to be downloaded and the deciphering PID ECM of the file. The program loads the index table and sends it to the device, which subsequently requests the downloading of the file to the program. The program extracts the PID and PID ECM for the file from the index table and requests demultiplexing to enable the file to be downloaded in its entirety. At any time, a call "fdload_offline" may be issued by the program to instruct the FDLOAD device to stop managing the serial and/or parallel ports.

Devices 4060 are identified with a unique identifier "device_id", for example, "LCARD_DEVICE_ID" identifies the LCARD device and "RCARD_DEVICE_ID" identifies the RCARD device.

When a new device 4060 is created, it can be installed in existing receiver/decoders 2020 by downloading the relevant application 4056" from the broadcast centre.

This downloading is performed in the receiver/decoder 2020 by an application 4056 which checks the hardware and software versions and, if correct, loads the software module representing the new device 4060 and asks a procedure of the library 4058 to install the new device code within the firmware (in FLASH memory). This can provide a flexible and secure installation of new functions within the receiver/decoder 2020 without affecting the rest of the software.

Device manager 4074 is a common interface between the application 4056 and the devices 4060 associated with specific functions of the receiver/decoder 2020. By adopting such a common interface, procedures, stored in a "library" of the run time engine 4008, permitting access to the devices 4060, and "events" associated with the devices may be standardised using a limited number of procedures. The device manager 4074 controls access to devices 4060, declares receipt of an unexpected event, and manages shared memory.

There are 18 procedures that give access to the device manager 4074, to a device 4060 itself or to memory management.

Device_Open_Channel ()	Opens a channel to the manager
Device_Close_Channel ()	Closes a channel to the manager
Device_Open_Device ()	Opens a channel to the device
Device_Close_Device ()	Closes a channel to the device
Device_Event ()	Unexpected event management
Device_Call ()	Synchronous access to a device
Device_Io ()	Asynchronous access to a device
Device_Info ()	Information about a device
Set_Buffer_outline ()	Defines a memory partitioning for buffers
Get_Buffer_outline ()	Reads the present buffer partitioning area
Pool_Info ()	Provides information about the present buffer partitioning area
Device_Alloc_Buffer ()	Allocates memory
Device_Free_Buffer ()	Frees memory
Device_Lock_Buffer ()	Locks memory
Device_Info_Free_Buffer ()	Gives the number of free buffers
Device_Info_Alloc_Buffer ()	Lists the allocated buffers
Get_Buffer_Size ()	Gives the size of the allocated buffer

With reference to Figure 7, before using the services of any device 4060, a program (such as an application script) has to be declared as a "client" 4076, that is, a logical access-way to the device 4060 or the device manager 4074. The manager gives the client 4076 a client number which is referred to in all accesses to a device.

A device 4060 can have several clients 4076, the number of clients 4076 for each device 4060 being specified depending on the type of device 4060.

A client 4076 is introduced to the device manager 4068 with the procedure Device_Open_Channel() 4078. With this procedure the device manager gives the client 4076 the client number. The device manager 4074 outputs a "client_id", comprising the address of the variable containing the allocated client number (there should never be two clients with the same number), and an "error_code", comprising

either error_code "0" indicating that the procedure has been completed successfully, or error_code "e_client_max" indicating that the maximum number of clients that can be handled by the device manager, typically 256, has been reached.

The first client number allocated is 0, increasing by 1 for each call using the procedure "Device_Open_Channel" up to the value of 255. This occurs independently from the removal of any client from the list of clients. When the client number 255 has been allocated and a subsequent "Device_Open_Channel" call occurs, either the error code "e_client_max" is output to indicate that the list of clients is "full" or the client is allocated the smallest client number available, that is, a client number of a client that previously had been removed from the list of clients.

To declare a client 4076 to a device 4060, a client uses the procedure Device_Open_Device() 4080, transmitting therewith its client_id and the device_id. The device manager outputs an error message "0" if the procedure has been successfully completed, "e_client_inconnu" if the client does not exist, "e_periph_inconnu" if the device does not exist, e_client_max, or "w_deja_vu" if the device has already been declared to the device.

A client 4076 can be removed from a list for a device 4060 with the procedure Device_Close_Device(). The client_id and device_id are input to the device manager, which in turn outputs either error_code "0" if the procedure has been completed successfully, error_code "w_client_inconnu" if the client is unknown to the device manager or error code "w_periph_inconnu" if the device is unknown to the device manager.

A client 4076 can be removed from the client list of the device manager 4074 with the procedure Device_Close_Channel (). The "client_id" is input to the device manager, which in turn outputs either error_code "0" if the procedure has been completed successfully or error_code "w_client_inconnu" if the client is unknown to the device manager. This procedure deletes the client from the list of the device manager 4074 and from the list of each device 4060 for which he is a client, ie. if not

already done so by the application, the Device_Close_Channel () procedure calls the Device_Call_Device () procedure for each device for which that client is a client. The client number thus freed using this procedure may then be allocated, using the Device_Open_Channel () procedure, to a new client.

Procedure "Device_Info ()" provides information concerning a device 4060 to a client 4076. The client_id and device_id are input to the device manager 4074, which returns information concerning the device, typically the version of the device, the maximum number of clients for the device and the actual number of clients using the device, to the client, together with error_codes indicating that the procedure has been completed or that either the client or the device is unknown to the device manager 4074.

Procedure "Device_Event" is a means of managing "unexpected events", that is, that a specific circumstance, or something unexpected, has occurred. The "Device_Event" procedure enables a client to declare itself the receiver of an unexpected event from a device.

Messages for applications, called "events", are put into one of, say, five queues of the engine 4008. Each of these queues corresponds to a priority level 0 to 4 (4 = maximum priority, 0 = minimum priority).

When extracting messages from the queue for transfer to an application, the engine 4008 searches for the queue having the highest priority containing an event. The event is removed from the queue and used to activate the process sequencer unit for which it is intended.

All "external" events, whether input to the presentation function by the keyboard or received through an interface pass through an event interface before being processed by the engine 4008. However, any internal events, typically generated by internal devices, do not pass through the event interface but pass directly to the engine 4008.

For each procedure, the last calling parameter is the address of a variable to be set by the device manager indicating how the command was handled by the device manager.

The reason for the issue of an unexpected event to a client depends on the device 4060. For example, with respect to the device LCARD DEVICE, unexpected events include:

removal of a card from the card reader;

insertion of a card in the card reader; or

reset of the card by a client.

Each of the above is identified by a respective unique event code ("ev_lcard_extract", "ev_lcard_insert" and "ev_card_reset" respectively).

To receive unexpected events, the client must declare itself as a receiver of each event that it may wish to receive using the procedure "device_event". The following parameters are input to the device manager:

client_id;

device_id;

a command called "get_event" which enables the sending on an unexpected event;

the event code; and

the event priority, between 0 and 4.

Several devices may declare reception of the same unexpected event, with the same or a different priority. In this case, the event is sent to each client in a specific order (by priority and client number).

For example, client 2 declares himself the receiver of the event ev_card_extract from the LCARD device with priority 2. Client 3 declares himself the receiver of the same event with priority 2, client 1 with priority 4 and finally client 4 with priority 3. If the card is removed from the reader, the following sequences of events occurs;

the event ev_card_extract addressed to client 1 is placed in the queue of the run time engine 4008 corresponding to priority 4;

the same event addressed to client 4 is placed in the queue of the run time engine 4008 corresponding to priority 3;

the same event addressed to client 2 is placed in the queue of the run time engine 4008 corresponding to priority 2; and

the same event addressed to client 3 is placed in the queue of the run time engine 4008 corresponding to priority 2.

For some devices a buffer may be involved in an unexpected event. In such a case, the device allocates this buffer via the device manager 4068.

In the application programming language, a procedure enables access to parameters that may be associated with events;

event_code;

client_id:

evt_param1 (typically having a size of 4 bytes); and

evt_param2 (typically having a length of 2 bytes).

The exact meaning of the evt_param parameters depend on the device and the reason for the posting of the event. The association of a buffer with an unexpected event depends on the device. If the association exist, evt_param1 corresponds to the data zone address of the associated buffer. If there is no buffer, the event will still be issued but with the parameter evt_param1 set at "0".

If the device has a buffer for association with an unexpected event, the removal of a client from a list of a device or a list of the device manager results in deallocation of the buffer, except where the buffer has been "locked" by the application using the procedure "device_lock_buffer".

Evt_param2 indicates that the procedure has been successfully completed and that a result has been obtained by the device.

A client may stop the reception of any unexpected event using the command

"ret_event" instead of the command "get_event".

The device manager 4074 provides access by clients 4076 to devices 4060 in two different modes: "synchronous access" or "asynchronous access".

Synchronous access, using procedure "Device_Call", is a means of accessing a function specific to a device and which does not involve the placing of the response from the device in a queue of the run time engine 4008; the response is available immediately. Typically, this procedure is used in configuration of the hardware interfaces of the receiver/decoder 2020.

Using this procedure, the client inputs the following parameters to the device via the device manager:

client_id;

device_id;

"call_cmde", which comprises the type of operation to be performed by the device. For example, with respect to the SERIAL device allowing a program to communicate with equipment via the serial link, the command "serial_setup" sets up the serial link configuration;

"em_adr", which comprises the address of the memory location of the input data; and

"rec_adr", which comprises the address of the memory location to which the device must write the output data.

Upon completion of the operation requested by the client, the device outputs a "call-report", which comprises a report of the run. The call_report typically comprises one of a predetermined number of reports that may be issued to the client. In turn, the device manager 4074 issues an error_code to the client 4076, the error_code typically comprising one of:

e_client_inconnu;

e_periph_inconnu;

"e_cmde_inconnu", which notifies the client that the command was unknown;

"e_report_periph", which is an error specific to the device; and

"0", which indicates that the procedure was completed successfully.

The device only writes the output data to the address of the memory location identified by the parameter "rec_adr" when the error_code "0" is received.

Device_call blocks all other applications until the run has been completed by the device.

Asynchronous access, using procedure "Device_Io", is a means of accessing a function specific to a device and which involves waiting for a response. For example, with respect to the SERIAL device, the command "serial_send" sends a message via the serial link. Following successful transmission of all of the data (before any pre-set "time-out" parameter is reached) the device sends a report to the program. When this report is available an event is put into a queue of the engine 4008 to signal its arrival.

Using this procedure, the client inputs the following parameters to the device via the device manager:

client_id;

device_id;

"io_cmde", which comprises the type of operation to be performed by the device;

the event code associated with the end of the operation;

the event priority associated with the event;

em_adr; and

rec_adr.

Upon completion of the operation requested by the client, the device outputs an "io-report", which comprises a report of the run.

The io_report typically comprises one of a predetermined number of reports that may be issued to the client and one of two reports ("e_not_done", indicating that the

requested information is not yet available to the client, and "0", indicating that the requested information is available) associated with the event.

Upon receipt of the report "0", the device manager issues an error_code to the client 4076, the error_code comprising one of:

e_client_inconnu;

e_periph_inconnu;

e_cmde_inconnu;

"e_cmde_inconnue" which indicates that the event code is unknown;

"e_priorite_inconnue" which indicates that the priority is unknown;

e_report_periph;

0.

It will be understood that the present invention has been described above purely by way of example, and modifications of detail can be made within the scope of the invention.

Each feature disclosed in the description, and (where appropriate) the claims and drawings may be provided independently or in any appropriate combination.

In the aforementioned preferred embodiments, certain features of the present invention have been implemented using computer software. However, it will of course be clear to the skilled man that any of these features may be implemented using hardware. Furthermore, it will be readily understood that the functions performed by the hardware, the computer software, and such like are performed on or using electrical and like signals.

Cross reference is made to our co-pending applications, all bearing the same filing date, and entitled Signal Generation and Broadcasting (Attorney Reference no. PC/ASB/19707), Smartcard for use with a Receiver of Encrypted Broadcast Signals, and Receiver (Attorney Reference No. PC/ASB/19708), Broadcast and Reception System and Conditional Access System therefor (Attorney Reference No.

PC/ASB/19710), Downloading a Computer File from a Transmitter via a Receiver/Decoder to a Computer (Attorney Reference No. PC/ASB/19711), Transmission and Reception of Television Programmes and Other Data (Attorney Reference No. PC/ASB/19712), Downloading Data (Attorney Reference No. PC/ASB/19713), Computer Memory Organisation (Attorney Reference No. PC/ASB/19714), Television or Radio Control System Development (Attorney Reference No. PC/ASB/19715), Extracting Data Sections from a Transmitted Data Stream (Attorney Reference No. PC/ASB/19716), Access Control System (Attorney Reference No. PC/ASB/19717), Data Processing System (Attorney Reference No. PC/ASB/19718), and Broadcast and Reception System, and Receiver/Decoder and Remote Controller therefor (Attorney Reference No. PC/ASB/19720). The disclosures of these documents are incorporated herein by reference. The list of applications includes the present application.

CLAIMS

1. A method of providing access by a program to at least one component of a computer system, said method comprising the steps of:

storing a logical device associated with the or each respective component in said system, the or each logical device having a respective device identifier;

assigning a program identifier to the program; and

outputting from the program a signal comprising the program identifier and the device identifier of the or each logical device to a common interface between the program and the or each logical device to create a communication channel between said program and the or each logical device via said interface.

2. A method according to Claim 1, wherein said signal further comprises a command enabling receipt by the program of a message output from a respective logical device indicating a change of state of the component associated with the device.

3. A method according to Claim 2, wherein said message is to be stored temporarily in queue means of said computer system for subsequent transfer to said program.

4. A method according to Claim 3, wherein said queue means comprises a plurality of queues, each queue having a respective priority level indicative of the order in which messages are to be transferred from the queue means to the program, and said signal further comprises the priority level of the queue in which said message is to be stored temporarily.

5. A method of providing access by a plurality of programs to at least one component of a computer system using the method according to any preceding claim, each program accessing the or each component via said interface.

This Sheet Has Been Replaced

THIS SHEET HAS BEEN REPLACED

6. A method of transmitting data between a program and a component of a computer system; said method comprising the steps of

providing access by the program to the component using the method according to Claim 1; and

subsequently outputting from the program a signal comprising the program identifier, the device identifier of the logical device to a common interface, a command instructing operation of a component by the logical device associated therewith, the address of data to be input to said component by said device and the address for data output by said component to said logical device to said common interface.

7. A method of transmitting data between a program and a component of a computer system; said method comprising the steps of

providing access by the program to the component using the method according to Claim 4; and

subsequently outputting from the program a signal comprising the program identifier, the device identifier of the logical device to a common interface, a command instructing operation of a component by the logical device associated therewith, the priority level of the queue in which a message output from the respective logical device indicating a change of state of the component associated with the device is to be stored temporarily prior to transfer to the program, the address of data to be input to said component by said device and the address for data output by said component to said logical device to said common interface.

8. A method according to any preceding claim, wherein the program and/or the or each logical device is input to said computer system via a component.

9. A method according to any preceding claim, wherein the or each component comprises at least one of an MPEG flow tuner, a serial interface, a parallel interface, a modem and a smartcard reader.

10. A method of providing access by a program to at least one component of a

THIS SHEET HAS BEEN REPLACED

computer system or a method of transmitting data between a program and a component of a computer system substantially as herein described.

11. Apparatus for providing access by a program to at least one component of a computer system, a logical device associated with the or each respective component being stored in said system, the or each logical device having a respective device identifier; said apparatus comprising:

means for assigning a program identifier to the program; and

means for creating a communication channel between said program and the or each logical device in accordance with a signal output from the program and comprising the program identifier and the device identifier of the or each logical device

12. Apparatus according to Claim 11, further comprising:

means for enabling receipt by the program of a message output from a respective logical device indicating a change of state of the component associated with the device.

13. Apparatus according to Claim 12, wherein the computer system further comprises queue means for storing temporarily a message output by said device for subsequent transfer to said program.

14. Apparatus according to Claim 13, wherein said queue means comprises a plurality of queues, each queue having a respective priority level indicative of the order in which messages are to be transferred from the queue means to the program.

15. Apparatus according to any of Claims 11 to 14, wherein the computer system comprises means for storing data input to said component by said device and data output by said component to said logical device to said common interface.

16. Apparatus according to any one Claims 11 to 15, wherein the or each logical component is connected to the respective associated component via a device driver.

This Sheet Has Been Replaced

THIS SHEET HAS BEEN REPLACED

17. A receiver/decoder for receiving broadcast signals, said receiver/decoder comprising apparatus according to any of Claims 11 to 16.

18. A receiver/decoder according to Claim 17, further comprising means for receiving a compressed MPEG-type signal, means for decoding the received signal to provide a television signal and means for supplying the television signal to a television.

19. Apparatus or a receiver/decoder substantially as herein described with reference to the accompanying drawings.

This Sheet Has Been Replaced

THIS SHEET HAS BEEN REPLACED