DOWNLOADING A COMPUTER FILE FROM A TRANSMITTER VIA A RECEIVER/DECODER TO A COMPUTER

This invention relates to:-

- a method of downloading a computer file from a transmitter (such as a satellite television uplink transmitter) via a receiver/decoder (such as a digital satellite television set top box) to a computer (such as a PC or workstation);
- a computer for use in performing part of such a method;
- a recording medium (such as a diskette or CD-ROM) bearing a computer program for programming a processing means of such a computer;
- an MPEG transmission system; and
- a receiver/decoder which is arranged to receive tables of MPEG video and/or audio data and blocks of computer file data.

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 make computer files or software available for downloading to an end user.

Data errors arise not infrequently in such transmission systems. Whilst these can often be glossed over, in the case of errors in video or audio data, using interpolation techniques and the like, an error in a mere single bit of a computer file may render that file useless.

A first aspect of the present invention relates more particularly to a method of downloading a computer file from a transmitter via a receiver/decoder to a computer, comprising the steps of: dividing the file into blocks of data; transmitting the blocks from the transmitter; receiving the transmitted blocks at the receiver/decoder; and supplying the received blocks from the receiver/decoder to the computer. The blocks typically include CRC codes so that they can be tested for validity, and the transmission of the blocks might typically be repeated so that, if a block is not received correctly,

or at all, the first time it is transmitted, it can be received the second or a subsequent time it is transmitted. A problem with such a scheme is the additional time taken to receive the file if one or more blocks are not validly received. In the extreme, if only the last block is not validly received, then the total download time for the file will be approximately doubled. Another problem with such a scheme is that, if the blocks are not downloaded in the order in which they are transmitted, for example because the download rate from the receiver/decoder to the computer is less than the download rate from the transmitter to the receiver/decoder, then the waiting time for the last few blocks may be relatively large. In the extreme, the waiting time for the last block may be almost as large as the time taken to transmit a complete set of the blocks, even though all of the blocks are validly received.

In accordance with a first aspect of the present invention, there is provided a method of downloading a computer file from a transmitter via a receiver/decoder to a computer, comprising the steps of: dividing the file, or a part thereof, into a series of R groups of data, and subdividing each of the groups into a series of C blocks of the data; generating a group of C redundancy blocks, each redundancy block being related to respective ones of the data blocks in the R groups by being a bitwise exclusive OR or NOR of those data blocks; transmitting each of the groups of data blocks and the group of redundancy blocks from the transmitter; receiving at least some of the transmitted blocks at the receiver/decoder; supplying the received blocks from the receiver/decoder to the computer; and determining with the computer for at least one of the received redundancy blocks whether that redundancy block and all except one of its respective data blocks have been received, and if so regenerating said one data block by performing a bitwise exclusive OR or NOR operation on that redundancy block and the respective received data blocks.

For the avoidance of doubt, the reference to a bitwise exclusive OR operation on a plurality of data blocks means that each bit of a first one of the blocks is XORed with a respective bit of a second one of the blocks, and the result is XORed with a respective bit of a third one of the blocks, and so on until the last one of the blocks. For simplicity, it is the first bit of the first block which is XORed with the first bit of

the second block, with the result being XORed with the first bit of the third block, and so on to give the first bit of the result, and similarly for the second bits, third bits and so on. Nevertheless, it is possible to employ different predetermined permutations of the bits of the blocks in the XOR operations. Similar comments apply in the case where exclusive NOR, rather than XOR, operations are employed.

The first aspect of the invention provides the advantage, especially towards the end of a file downloading operation, that if a data block has not yet been received, it may be possible to regenerate the block from the respective redundancy block and the respective data blocks which have been received, without having to wait for the remaining data block to be transmitted and validly received. If there has been a problem in transmission or reception of a particular block, then there is a relatively high probability that there will also be a problem with transmission or reception of the next block in the same series of C blocks. However, by appropriate selection of the value C, the probability that there will also be a problem with transmission or reception of the corresponding block in the next series of C blocks will be relatively small. It will be appreciated that the addition of such redundancy blocks will increase the time required to transmit all of the blocks by 100/R %. However, by appropriate selection of the value of R, the benefits provided by this aspect of the invention can be caused to outweigh this disadvantage. Typically, the values of C and R might be chosen to be 50 and 10, respectively.

The determining and regenerating steps may be performed for only some of the received redundancy blocks, especially those towards the end of a file downloading operation. However, the determining and regenerating steps are preferably performed for each of the received redundancy blocks for which all except one of the respective data blocks have been received. In this latter case, the method may further comprise the step of determining when sufficient of the blocks have been received to enable any remaining data blocks to be regenerated and then ceasing to receive further blocks for the file or said part thereof. The regeneration steps can therefore take place after all the required blocks have been received.

The values of the numbers R and C may be fixed for all file downloading operations. However, to provide flexibility, the method preferably further comprises the steps of transmitting from the transmitter data indicative of the values of the numbers R and C, receiving the R and C data at the receiver, and supplying the R and C data to the computer; and wherein the computer is operable to use the supplied R and C data in determining to which of the data blocks each redundancy block relates.

The blocks are preferably ordered such that an *i*th (for *i*=1 to C) one of the redundancy blocks in the group of C redundancy blocks is so related to the *i*th data block in each of the R groups of C data blocks. However, other relationships are also possible.

In accordance with a second aspect of the present invention, there is provided a computer for use in performing part of the method of the first aspect of the invention, the computer comprising: means for receiving the blocks from the receiver/decoder; means for processing the received blocks, the processing means being programmed to determine for at least one of the received redundancy blocks whether that redundancy block and all except one of its respective data blocks have been received, and if so to regenerate said one data block by performing a bitwise exclusive OR or NOR operation on that redundancy block and the respective received data blocks; and means for storing the received data blocks and any such regenerated data block.

Preferably, the processing means is programmed to perform such a determination and regeneration for each of the received redundancy blocks for which all except one of the respective data blocks have been received. In this case, the processing means is preferably programmed to determine when sufficient of the blocks have been received to enable any remaining data blocks to be regenerated and then to cause the receiving means to cease receiving further blocks of the file or said part thereof.

The receiving means is preferably operable to receive data indicative of the values of the numbers R and C, and the processing means is programmed to use the R and C data in determining to which of the data blocks each redundancy block relates.

Preferably, the processing means is programmed to link together the received data blocks and any such regenerated data blocks to reconstruct the computer file or part thereof, and the storage means is operable to store the reconstructed file or part thereof.

In accordance with a third aspect of the present invention, there is provided a recording medium bearing a computer program for programming a processing means of a computer according to the second aspect of the invention to operate as specified above.

In accordance with a fourth aspect of the present invention, there is provided an MPEG transmission system, comprising: means for dividing a file, or part thereof, to be downloaded to a receiver/decoder into a series of R groups of data and subdividing each of the groups into a series of C blocks of the data; means for generating a group of C redundancy blocks, each being related to respective ones of the data blocks in the R groups by being a bitwise exclusive OR or NOR of those data blocks; and means for transmitting each of the groups of data blocks and the group of redundancy blocks.

A fifth aspect of the present invention also relates more particularly to a method of downloading a computer file from a transmitter via a receiver/decoder to a computer, comprising the steps of: dividing the file into blocks of data; repeatedly transmitting the blocks from the transmitter; receiving the transmitted blocks at the receiver/decoder; and supplying the received blocks from the receiver/decoder to the computer.

The blocks of data are transmitted from the transmitter to the receiver/decoder at a fixed data rate, which might typically be 700 kbits/s. To enable the receiver/decoder to be used with computers having different capabilities, and to exploit those capabilities, the receiver/decoder needs to be able to download the blocks to the computer in various modes having different data rates, such as IEEE 1284 ECP–mode having a typical data rate of 1 Mbit/s, IEEE 1284 parallel byte–mode having a typical data rate of 400 kbits/s, IEEE 1284 parallel nibble–mode having a typical data rate of 150 to 200 kbits/s, and RS–232 serial mode having a maximum data rate of perhaps 115 kbits/s and a typical data rate possibly substantially lower than this.

When the data rate into the receiver/decoder is faster than the data rate from the receiver/decoder, steps need to be taken to ensure that all of the data blocks are received and downloaded to the computer. This could be achieved by providing a buffer in the receiver/decoder which is large enough to store a complete computer file. However, this would be expensive. Alternatively, the minimum data rate from the receiver/decoder to the computer could be estimated, divided into the expected data rate to the receiver/decoder to provide a rounded–up integer X, and then one in every X of the received blocks could be downloaded to the computer so that N/X blocks are received in one phase, and this phase could then be repeated X times so as to download all N blocks. However, for efficiency, this is reliant upon the data rate to the computer being accurately estimated, and this is not possible because that data rate can vary. Also, serious problems would arise with this method if a particular data block were lost. The fifth aspect of the invention is concerned with finding a solution to these problems.

In accordance with a fifth aspect of the present invention, there is provided a method of downloading a computer file from a transmitter via a receiver/decoder to a computer, comprising the steps of: dividing the file into a number (N) of blocks; repeatedly transmitting the blocks of the file from the transmitter, each with an identity of that block; and, at the receiver/decoder, providing a plurality of N flags which are initially reset, one for each block; and cyclically performing the steps of:– receiving a currently transmitted one of the blocks, and determining whether the flag for that block is set, and if not supplying that block from the receiver to the computer and setting the flag for that block; until all of the flags have been set.

It will therefore be appreciated that it does not matter in what order the blocks are received by the receiver/decoder. The receiver/decoder is able to determine whether a block is needed, and if so to download that block to the computer. If the receiver/decoder can download the blocks to the computer as quickly as the blocks can be downloaded to the receiver/decoder, then most, if not all, of the blocks will be downloaded sequentially in order. However, if the receiver/decoder cannot download the blocks to the computer as quickly as the blocks can be downloaded to the receiver/decoder, for example at only half the speed, then generally speaking alternate

blocks will be downloaded during one phase, and then the remaining blocks will be downloaded during a subsequent phase. If the ratio between the download rate to the receiver/decoder to the download rate from the receiver/decoder varies, then the arrangement described above will automatically adjust the interleaving of the blocks.

Preferably, following each supplying step, the computer is arranged to send an acknowledgement of receipt of the supplied block to the receiver/decoder, and wherein the setting step for each block is performed in response to receipt of the acknowledgement for that block. Accordingly, if a block is not properly acknowledged by the computer, then the receiver/decoder downloads the block again at the next available opportunity.

The method may further include the step of repeating the method but without the step of resetting the flags. Thus, if there is an interruption in reception of the blocks, the downloading method can be resumed without repeating the downloading of blocks which have already been received.

In accordance with a sixth aspect of the present invention, there is provided a receiver/decoder which is arranged to receive tables of MPEG video and/or audio data and blocks of computer file data, wherein a computer file is received as a plurality of N such blocks which are cyclically received and each of which includes an identification of that block, the receiver/decoder including: means for storing a plurality of N flags which are initially reset, one for each computer file block; means for temporarily storing a currently received computer file block; means for supplying the block stored in the block storing means to a computer; and control means which is arranged to determine whether the flag for the currently received block is set in the flag storing means, and if not to cause the block supplying means to supply that block from the block storing means to the computer and to set the flag for that block in the flag storing means.

The receiver/decoder preferably further comprises means for receiving an acknowledgement from the computer that such a block has been received from the

block supplying means, and wherein the control means is operable so to set the flag for that block in response to receipt of such an acknowledgement for that block.

Preferably, the control means can be operated to cause resumption of the reception of computer file blocks after a lost connection without resetting the flags.

The block supplying means may be one of a plurality of such block supplying means of different types, for example including at least two of: an ECP–mode IEEE parallel port; a byte–mode IEEE parallel port; a nibble–mode IEEE parallel port; and an RS–232 serial port.

It should be noted that the features of the first and fifth aspects of the invention may be employed together in a method of downloading a computer file from a transmitter via a receiver/decoder to a computer.

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;

Figure 2 shows the architecture of an interactive system of the digital television system of figure 1;

Figure 3 is a schematic diagram of interfaces of a receiver/decoder forming part of the system of figures 1 and 2;

Figure 4 is a schematic diagram of a remote controller used in the digital television system;

Figure 5 illustrates an arrangement of blocks making up a file to be downloaded;

Figure 6 illustrates the sequence of the blocks while the file is being downloaded;

Figure 7 is a functional diagram of the receiver/decoder during the file download operation;

Figures 8–12 are flow diagrams illustrating processes which take place in the receiver/decoder during the file download operation;

Figure 13 is an alternative illustration of an arrangement of blocks making up a file

to be downloaded;

Figure 14 is an illustration in greater detail of a first matrix of the blocks shown in figure 13;

Figure 15 is an illustration of a generalised matrix of the blocks shown in figure 13;

Figure 16 is a flow diagram illustrating a process which takes place in an application and data server in the television system in preparing a file for downloading; and

Figure 17 is a flow diagram illustrating a process which takes place in a computer during a file download operation.

An overview of a digital television system 1000 according to the present invention 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 telecommunications 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 interacting 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 their 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, at 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 of the 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. A "memory volume" of the receiver/decoder is a storage space for modules. Modules may be downloaded into the receiver/decoder 2020 from the MPEG–2 transport stream.

Physical interfaces of the receiver/decoder 2020 are used for downloading data. With reference to Figure 3, the decoder 2020 contains, for example, six downloading devices; MPEG flow tuner 4028, serial interface 4030, parallel interface 4032, modem 4034 and two card readers 4036.

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 2026 (as shown in more detail in figure 4) on a button object seen on the screen of the television set 2022 and presses the validation key, an 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) of the 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 applications 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 the 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 receiver/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.

Particular features of the PC Download application will now be described. A particular technical problem to be solved is that the computer to which it is ultimately desired to download the software will have a different rate of data input from the rate at which it is broadcast. For example, the difference may be from 9 kbits/s to 1000 kbit/s, with the software being transmitted via the MPEG–2 stream at, say, 500 kbits/s.

To deal with this problem, in overview, in the broadcasting centre, the software is split into N blocks (see Figure 5) and then repeatedly inserted into the MPEG private section (see Figure 6). The receiver/decoder 2020 includes a bitmap which describes all of the software blocks. When the downloading is commenced, every bit in the bitmap is set to zero. The steps in the downloading procedure are: read the block number; if bit=0 send block to the receiving computer; and the receiving computer checks the block and sends an acknowledgement as appropriate. The receiving computer requires its own software, usually provided on a floppy disk, to join the various blocks of the transmitted software. Auto–interleaving provides a measure of flow control. Given that the broadcast data rate cannot be altered, the blocks in successive cycles are auto–interleaved. According to the bit rate of the receiving computer, in successive cycles of a carousel every, say, third or fourth block is transmitted to the decoder 2020 until all the blocks have been received.

Problems can arise if a block is lost. In overview, a solution to this problem is to use the concept of redundancy using an XOR function. Additional redundant blocks are added to the data stream from which the values of lost blocks can be derived if necessary.

The PC download application will now be described in more detail. During a PC download, the data is typically downloaded from the transponder 2014 to the

receiver/decoder 2020 via the MPEG flow tuner 4028 at a data rate typically of 700 kbits/s, and from the receiver/decoder 2020 to the computer via the parallel interface 4032 or serial interface 4030. The receiver/decoder supports downloading to the computer via the parallel interface 4032 in IEEE 1284 ECP-mode having a typical data rate of 1 Mbit/s, in byte-mode having a typical data rate of 400 kbits/s and in nibble-mode having a typical data rate of 150 to 200 kbits/s. The receiver/decoder also supports downloading to the computer via the serial interface 4030 in RS-232 mode having a maximum data rate of perhaps 115 kbits/s and a typical data rate possibly substantially lower than this. The receiver/decoder 2020 provides all of these modes of downloading to the computer in order to provide flexibility, and the user will normally choose the fastest available mode of downloading.

When the data rate into the receiver/decoder 2020 is faster than the data rate from the receiver/decoder 2020, steps need to be taken to ensure that all of the data blocks are received and downloaded to the computer. This could be achieved by providing a buffer in the receiver/decoder 2020 which is large enough to score a complete computer file. However, this would be expensive. Alternatively, the minimum data rate from the receiver/decoder 2020 to the computer could be estimated, divided into the expected data rate to the receiver/decoder 2020 to provide a rounded-up integer X, and then one in every X of the received blocks could be downloaded to the computer so that N/X blocks are received in one phase, and this phase could then be repeated X times so as to download all N blocks. However, for efficiency, this is reliant upon the data rate to the computer being accurately estimated, and this is not possible because that data rate can vary. Also, serious problems would arise with this method if a particular data block were lost.

In order to deal with these problems, the receiver/decoder 2020 is functionality arranged during a PC download operation as shown in figure 7. The incoming bitstream from the receiver 2018 undergoes a block detection function 10 and can then be passed by a selector 12 to a buffer 14A or a buffer 14B. The contents of a selected one of the buffers 14A, 14B can then be supplied via a selector 16 and a selected one of the parallel port 4032 and the serial port 4030 to a corresponding port of the PC 14. The

buffers 14A, 14B are provided by portions of RAM of the receiver/decoder 2020, as too is a bitmap 20. The block detection function 10, selectors 12, 16, buffers 14 A, 14 B, parallel and serial ports 4032, 4030 and the bitmap 20 are controlled by a processor 22 running the PC download application.

The operation of the receiver/decoder 2020 during a PC download operation will now be described with reference to the flow diagrams of figures 8 to 12. During an initialisation process, as shown in figure 8, in step 24 a received block is detected by the block detection function 10, and the number N of blocks is determined from a header for each of the blocks which includes the number i of that block and the total number N of blocks. In step 26, the bitmap 20 is set up with a number of bits equal to the number N, and each bit is reset to 0. In steps 28, 30, a buffer A flag and a buffer B flag are both set to "empty". In the example being described, the possible states of these flags are "empty", "filling", "full" and "emptying". In step 32, an output mode to the PC is set. Then, in step 34, the initialisation process hands over to the six processes shown in figures 9, 10A, 10B, 11A, 11B and 12, which are performed substantially in parallel.

Figure 9 shows the routine which controls the start of downloading to the buffers 14A, 14B. In step 36, the process waits until a new block is detected by the block detection function 10. Once a new block is detected, in step 38, the number i of the new block is detected by the block detection function 10. In step 40, a check is made in the bitmap 20 whether the bit b(i) is set to 1, and if so the block is ignored and the process reverts to its beginning. If not, then in step 42, the buffer A flag is tested for "empty", and if so, then in step 44 the buffer A flag is set to "filling", and then in step 46 downloading of the block into the buffer 14A via the selector 12 is started and the process then reverts to its beginning. If, in step 42, the buffer A flag is not set to "empty", then in step 48, the buffer B flag is tested for "empty". If so, then in step 50, the buffer B flag is set to "filling", and then in step 52 downloading of the block into the buffer 14B via the selector 12 is started and the process then reverts to its beginning. If in steps 42 and 48, neither buffer flag is set to "empty", then the process reverts to its beginning.

Figure 10A shows the routine which is carried out at the end of downloading of a block to the buffer 14A. In step 54, the process waits until the end of the block is detected by the block detection function 10. Then, in steps 56, 58, the block is checked using a CRC included in the block. If the block is good, then in step 60, the buffer A flag is set to "full", whereas if the block is bad, then in step 62, the buffer A flag is set to "empty", as a result of which the block will be ignored. A complementary routine is carried out at the end of downloading of a block to be buffer 14B, as is shown in figure 10B.

Figure 11A shows the routine which controls the downloading of a block from the buffer A to the PC 18. In step 64, the process waits until the buffer A flag becomes set to "full". Then, in step 66, a check is made whether the buffer B flag is set to "emptying", and if it is then the process waits until the buffer 10B has completed emptying. Then, in step 68, the buffer A flag is set to "emptying". Then, in step 70, the block i in the buffer 14A starts to be downloaded via the selector 16 to the PC 18 using the set output mode. In step 72, the process waits until the download is complete. Then, in step 74 a check is made for an acknowledgement from the PC 18. If the acknowledgement is received, then in step 76 the bit b(i) for the block i in the bitmap 20 is set to 1, and then in step 78 the buffer B flag is set to "empty". Then, the process reverts to its beginning. If, in step 74, the acknowledgement is not received, then the step 76 is omitted, and therefore the downloading of the block will be disregarded. A complementary routine is carried out to control the downloading of a block from the buffer 14B to the PC 18, as shown in figure 11B.

Figure 12 shows the routine which checks that the downloading of the computer file is complete. In step 80, a check is made whether all of the bits b(1) to b(N) in the bitmap 20 are set to 1, and if so, then the file downloading operation is terminated insofar as the receiver/decoder 2020 is concerned.

From the above, it will be appreciated that it does not matter in what order the blocks are received by the receiver/decoder 2020. The receiver/decoder 2020 is able to determine whether a block is needed, and if so then it places the block in one of the

buffers 14A, 14B and waits until the PC 18 is ready to receive that block. If a block fails the CRC check, or if a block is not properly acknowledged by the PC 18, then the receiver/decoder 2020 downloads the block again at the next available opportunity. Accordingly, if the receiver/decoder 2020 can download the blocks to the PC 18 as quickly as the blocks can be downloaded to the receiver/decoder 2020, then most, if not all, of the blocks will be downloaded sequentially in order. However, if the receiver/decoder 2020 cannot download the blocks to the PC 18 as quickly as the blocks can be downloaded to the receiver/decoder 2020, for example at only half the speed, then generally speaking alternate blocks will be downloaded during one phase, and then the remaining blocks will be downloaded during a subsequent phase. If the ratio between the download rate to the receiver/decoder 2020 to the download rate from the receiver/decoder 2020 varies, then the arrangement described above will automatically adjust the interleaving of the blocks.

The memory requirements of the arrangement described above, apart from for the operating software, are the two buffers 14A, 14B and the bitmap 20. Typically, the size of a block would be 2 kbytes (although up to 4 kbytes may be included in a single MPEG–2 table), and a maximum number of blocks in a file would be 2^{16} blocks. Therefore, the memory requirement would be 4 kbytes for the buffers 14A, 14B and $2^{16}/8 = 8$ kbytes for the bitmap 20, making a total of 12 kbytes, to enable a file as large as 2^{16} x 2 kbytes = 128 Mbytes to be downloaded.

In the arrangement described above, the PC 18 is programmed so that, once each block has been received, that block is written to the hard disk of the PC 18 with a file name which includes the number i of the block. Then, once all of the blocks have been received, the data in the N files are concatenated in order to regenerate the original computer file.

A modification to the arrangement described above with reference to Figures 5 to 12 will now be described, to deal with the case where a connection is lost between the broadcasting centre and the receiver/decoder 2020. In this case, the lost connection is detected by the receiver/decoder 2020, and the user may be prompted as to whether or

not they wish to recommence the download operation. If so, then the file download is requested, in the manner described above, except that step 26 in Figure 8 is omitted so that the bits b(1) to b(N) of the existing bitmap 20 are not reset to zero. It may therefore be appreciated that the downloading operation will resume without requiring that the blocks which have already been downloaded be downloaded again.

The concept of redundancy using an XOR function will now be described in detail. In the arrangement described above with reference to figures 5 and 6, the blocks of data were considered to be a mere sequence of the blocks 1 to N. In the arrangement which will now be described, the blocks can be considered to be arranged as a series of matrices, each containing (R + 1) rows and C columns of the blocks as shown in figures 13 to 15. As shown in figure 14 in particular, in the first matrix (matrix 1), the top row contains the blocks numbered i = 1 to C, the next row contains the blocks numbered i = C+1 to 2C, and so on until the bottom row, which contains the blocks numbered i = RC + 1 to (R+1)C. In the more general case of a matrix numbered j, as shown in figure 15, the top row contains the blocks numbered i = (j−1)(R+1)C + 1 to ((j−1)(R+1)+1)C and the bottom row contains the blocks numbered i = (j(R+1) − 1)C + 1 to j(R+1)C. At general position at row r and column c in matrix j, the number of the block is i = ((j−1)(R+1) + (r−1))C + c. In the matrices, the blocks in all of the rows except the bottom row (r = 1 to R) are data blocks into which the computer file to be downloaded has been divided. However, the blocks in the bottom row (r = R+1) are redundancy blocks which have been calculated and added into the MPEG bitstream by the application and data server 4006. More specifically, as shown in figure 16, in step 100, the file to be downloaded is divided into data blocks each of 2 kbytes. In step 102, values of R and C are set. In step 104, a matrix number counter j is set to 1. In step 106, the first RC data blocks are read, being considered as a matrix of C columns and R rows of the data blocks. In step 108, for each column c = 1 to C, a redundancy block for that column is calculated. More specifically the redundancy block is calculated as a bitwise XOR operation on the data blocks in the respective column. In step 110, the redundancy blocks are considered to be an additional bottom row (r=R+1) of the matrix. In step 112, block numbers i = (j−1)(R+1)C + 1 to (j(R+1) − 1)C are added as headers to the data blocks, block numbers i = (j(R+1) − 1)C + 1 to j(R+1)C are added as

headers to the redundancy blocks, and the blocks are stored. In step 114, a test is made as to whether the end of the file has been reached, and if not in step 116 the matrix counter j is incremented. Steps 102 to 108 are then repeated for the next RC data blocks. This continues until the end of the computer file is reached. An entry for the file is then added to a directory or index MPEG-2 table including the name, size and datestamp for the file and the values of R and C. The processed data blocks and redundancy blocks are then ready for transmission.

When a user wishes to download such a file, using the receiver/decoder 2020 they select the entry for the file from the directory or index, and the receiver/decoder 2020 then supplies the file information, including the values of R and C to the PC 18. Then, during a PC download of that file, the receiver/decoder 2020 need not function any differently to the manner described above with reference to figures 7 to 12. However, the PC 18 is programmed (by a program loaded onto the PC 18 from a diskette or CD-ROM or downloaded to the PC 18 from the receiver/decoder 2020) to deal with the redundancy blocks in the manner described below. In overview, each time that the PC 18 detects that a redundancy block has been downloaded, it checks whether all but one of the data blocks in the same column of the same matrix as the received redundancy block have been received. If so, the PC 18 regenerates the missing data block by performing a bitwise XOR operation on the received data blocks and the received redundancy block and stores the regenerated data block as if it had been received from the receiver/decoder 2020.

More specifically, referring to figure 17, in initial step 120 the PC 18 receives the values R,C, and then in step 122 receives a block. In step 124, the PC 18 extracts from the header of the received block its block number i. In step 126, the PC 18 stores the block as block i. In step 128 sends an acknowledgement to the receiver/decoder 2020. In step 130, the PC 18 checks whether at least all but one (R-1) of the blocks have been received in each of the C columns. This may be done by checking a directory of the hard disk of the PC 18 where the blocks are stored, or preferably, for reasons of speed, by employing a bitmap in the memory of the PC 18, which is marked off as each block is stored. If at least all but one of the blocks have been stored for each column,

then sufficient blocks have been received and the process proceeds to step 132, but if not, the process reverts back to step 122 to receive the next block. Step 132 controls a FOR–NEXT loop. For each column, in step 134, the PC 18 determines whether all of the data blocks for that column (that is R–1 data blocks) have been received. If so, then the FOR–NEXT loop proceeds to the next column. However, if not, then in step 136 the PC 18 regenerates the missing data block by performing a bitwise XOR operation on the data blocks which have been received and the redundancy block. Then, in step 138, the PC 18 stores the regenerated data block, and proceeds to the next column in the FOR–NEXT loop. At the end of the FOR–NEXT loop, the process proceeds to step 140, in which the computer file is regenerated from the received data blocks, and in the step 142 the computer file is stored on the hard disk of the PC 18. In step 134, determining whether a block is a data block or a redundancy block may be done by checking the truth of the relation $0 < i - jRC \leq C$ where the matrix number j is determined from $j = \text{int}((i-1)/(R+1)C) + 1$, where int( ) refers to the integer part of ( ). If true, the block is a redundancy block, and if false it is a data block.

The arrangement described above deals with two main problems. The first problem is concerned with brief interruptions in the bitstream from the transponder 2014 to the earth receiver 2018. It will be appreciated that if a particular block i has been affected by such an interruption, there is a relatively high probability that the next block i+1 in the same row will also be affected. The value of C is chosen such that there is a relatively small probability that the next block i+C in the same column will also be affected. Provided that R of the (R+1) blocks in each column are properly received, then using the redundancy block and the property of the bitwise XOR operation, the remaining block can be regenerated. The other problem is concerned with the effective rate at which blocks are received towards the end of the file download operation. It will be appreciated that if only one block is awaited, the waiting period may be almost as large as the time taken to transmit all of the blocks in the file. With the arrangement described above, there is a high probability that if only a few data blocks would otherwise be awaited, they can be regenerated from the redundancy blocks and the data blocks which have already been received. It should, of course, be noted that the transmission of additional redundancy blocks adds to the total time to transmit a

complete file, and therefore the advantages of dealing with transmission interruptions and reducing the time to wait for the last few blocks need to be balanced against the disadvantage of the increase in the total transmission time, by appropriate selection of the values of C and R. The choice of the values of C and R can be the subject of statistical studies or trial and error. However, typically the values might be C=50 and R=10. Preferably, both C and R are greater than five.

In the example described above, the values of C and R are determined when the file is prepared into blocks ready for transmission, and the values of C and R are included in the directory or index table for downloadable files. Alternatively, the values of C and R may be predetermined without their values being included in the directory or index table, and the PC 18 may be pre-programmed with the values of C and R.

In the example described above, blocks are downloaded until sufficient blocks have been downloaded to enable the file to be regenerated. Then, the missing blocks are regenerated and the file is regenerated. In an alternative example, if the redundancy block for a particular column has been downloaded, and if all but one of the data blocks of that column have been downloaded, the missing data block may be regenerated and marked as received at that stage, without waiting for sufficient blocks to have been downloaded to enable the whole file to be regenerated.

In the apparatus described above, the receiver/decoder 2020 has been illustrated as being separate from the PC 18. It will be appreciated, however, that the receiver/decoder 2020 may form a part of the PC 18, for example in the form of an add-on card.

It will be understood that the present invention has been described above purely by way of example, and other modifications of detail may 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 downloading a computer file from a transmitter via a receiver/decoder to a computer, comprising the steps of:

dividing the file, or a part thereof, into a series of R groups of data, and subdividing each of the groups into a series of C blocks of the data;

generating a group of C redundancy blocks, each redundancy block being related to respective ones of the data blocks in the R groups by being a bitwise exclusive OR or NOR of those data blocks;

transmitting each of the groups of data blocks and the group of redundancy blocks from the transmitter;

receiving at least some of the transmitted blocks at the receiver/decoder;

supplying the received blocks from the receiver/decoder to the computer; and

determining with the computer for at least one of the received redundancy blocks whether that redundancy block and all except one of its respective data blocks have been received, and if so regenerating said one data block by performing a bitwise exclusive OR or NOR operation on that redundancy block and the respective received data blocks.

2. A method as claimed in claim 1, wherein said determining and regenerating steps are for each of the received redundancy blocks for which all except one of the respective data blocks have been received.

3. A method as claimed in claim 2, further comprising the step of determining when sufficient of the blocks have been received to enable any remaining data blocks to be regenerated, and then ceasing to receive further blocks for the file or said part thereof.

4. A method as claimed in any preceding claim, further comprising the steps of transmitting from the transmitter data indicative of the values of the numbers R and C, receiving the R and C data at the receiver, and supplying the R and C data to the computer; and wherein the computer is operable to use the supplied R and C data in

This Sheet Has Been Replaced



determining to which of the data blocks each redundancy block relates.

5. A method as claimed in any preceding claim, wherein the blocks are ordered such that an *i*th (for *i*=1 to C) one of the redundancy blocks in the group of C redundancy blocks is so related to the *i*th data block in each of the R groups of C data blocks.

6. A method of downloading a computer file from a transmitter via a receiver/decoder to a computer, comprising the steps of:

dividing the file into a number (N) of blocks;

repeatedly transmitting the blocks of the file from the transmitter, each with an identity of that block; and

at the receiver/decoder:-

providing a plurality of N flags which are initially reset, one for each block; and

cyclically performing the steps of:-

receiving a currently transmitted one of the blocks; and

determining whether the flag for that block is set, and if not supplying that block from the receiver to the computer and setting the flag for that block;

until all of the flags have been set.

7. A method as claimed in claim 6, wherein, following each supplying step, the computer is arranged to send an acknowledgement of receipt of the supplied block to the receiver/decoder, and wherein the setting step for each block is performed in response to receipt of the acknowledgement for that block.

8. A method as claimed in claim 6 or 7, and further including the step of repeating the method of claim 6 or 7 but without the step of resetting the flags.

9. A method as claimed in any one of claims 1 to 5 and as claimed in any of claims 6 to 8.

This Sheet Has Been Replaced

10. A method of downloading a computer file from a transmitter via a receiver/decoder to a computer, substantially as described with reference to the drawings.

11. A computer for use in performing part of the method of any one of claims 1 to 5, 9 and 10, the computer comprising:
means for receiving the blocks from the receiver/decoder;
means for processing the received blocks, the processing means being programmed to determine for at least one of the received redundancy blocks whether that redundancy block and all except one of its respective data blocks have been received, and if so to regenerate said one data block by performing a bitwise exclusive OR or NOR operation on that redundancy block and the respective received data blocks; and
means for storing the received data blocks and any such regenerated data block.

12. A computer as claimed in claim 11, wherein the processing means is programmed to perform such a determination and regeneration for each of the received redundancy blocks for which all except one of the respective data blocks have been received.

13. A computer as claimed in claim 12, wherein the processing means is programmed to determine when sufficient of the blocks have been received to enable any remaining data block to be regenerated and then to cause the receiving means to cease receiving further blocks for the file or said part thereof.

14. A computer as claimed in any of claims 11 to 13, wherein the receiving means is operable to receive data indicative of the values of the numbers R and C, and the processing means is programmed to use the R and C data in determining to which of the data blocks each redundancy block relates.

15. A computer as claimed in any of claims 11 to 14, wherein the processing means is programmed to link together the received data blocks and any such regenerated data block to reconstruct the computer file or part thereof, and the storage means is operable

This Sheet Has Been Replaced

to store the reconstructed file or part thereof.

16. A computer for use in downloading a computer file from a transmitter via a receiver/decoder to a computer, substantially as described with reference to the drawings.

17. A recording medium bearing a computer program for programming a processing means of a computer as claimed in any of claims 11 to 16 to operate as specified in any of those claims.

18. A recording medium bearing a computer program, substantially as described with reference to the drawings.

19. An MPEG transmission system, comprising:
means for dividing a file, or part thereof, to be downloaded to a receiver/decoder into a series of R groups of data and subdividing each of the groups into a series of C blocks of the data;
means for generating a group of C redundancy blocks, each being related to respective ones of the data blocks in the R groups by being a bitwise exclusive OR or NOR of those data blocks; and
means for transmitting each of the groups of data blocks and the group of redundancy blocks.

20. An MPEG transmission system, substantially as described with reference to the drawings.

21. A receiver/decoder which is arranged to receive tables of MPEG video and/or audio data and blocks of computer file data, wherein a computer file is received as a plurality of N such blocks which are cyclically received and each of which includes an identification of that block, the receiver/decoder including:
means for storing a plurality of N flags which are initially reset, one for each computer file block;

This Sheet Has Been Replacec

means for temporarily storing a currently received computer file block;

means for supplying the block stored in the block storing means to a computer; and

control means which is arranged to determine whether the flag for the currently received block is set in the flag storing means, and if not to cause the block supplying means to supply that block from the block storing means to the computer and to set the flag for that block in the flag storing means.

22. A receiver/decoder as claimed in claim 21, further comprising means for receiving an acknowledgement from the computer that such a block has been received from the block supplying means, and wherein the control means is operable so to set the flag for that block in response to receipt of such an acknowledgement for that block.

23. A receiver/decoder as claimed in claims 21 or 22, wherein the control means is operable to cause resumption of the reception of the computer file blocks after a lost connection without resetting the flags.

24. A receiver/decoder as claimed in any of claims 21 to 23, wherein the block supplying means is one of a plurality of such block supplying means of different types.

25. A receiver/decoder as claimed in claim 24, wherein the plurality of block supplying means include at least two of: an ECP-mode IEEE parallel port; a byte-mode IEEE parallel port; a nibble-mode IEEE parallel port; and an RS-232 serial port.

26. A receiver/decoder substantially as described with reference to the drawings.

This Sheet Has Been Replacec