166166

G00002

0 3 APR 2002

DOWNLOADING DATA

This invention relates to:-

- a method of downloading data to a receiver/decoder;
- such a receiver/decoder per se; and
- a transmission system.

The term "receiver/decoder" used herein may connote a receiver for receiving either encoded or non-encoded signals, for example, television and/or radio signals, which may be broadcast or transmitted by some other means. The term may also connote a decoder for decoding received signals. Embodiments of such receiver/decoders may include a decoder integral with the receiver for decoding the received signals, for example, in a "set-top box", such a decoder functioning in combination with a physically separate receiver, or such a decoder including additional functions, such as a web browser, a video recorder, or a television.

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. As used herein, the term "digital transmission system" includes any transmission system for transmitting or broadcasting for example primarily audiovisual or multimedia digital data. Whilst the present invention is particularly applicable to a broadcast digital television system, the invention may also be applicable to a fixed telecommunications network for multimedia internet applications, to a closed circuit television, and so on.

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.

If a portion of data stored in the receiver/decoder becomes corrupted, the Bootstrap loader can be used to download a complete uncorrupted version of this data. If only a very small portion of the data has become corrupted, this can result in a significant amount of time being spent downloading the portions of the data which have not become corrupted.

The present invention seeks to solve these and other problems.

In a first aspect, the present invention provides a method of downloading data to a receiver/decoder, comprising the steps, at the receiver/decoder, of:

receiving a bitstream including the data;

downloading a loader for loading the data from the bitstream into the receiver/decoder; and

downloading said data from the bitstream using said downloaded data loader.

In one embodiment, the downloaded data loader comprises a data loading program. At least part of the data loader, preferably most of, or even all of the data loader, may be in the form of native code. As used herein, the term "native code" includes hardware specific code, code which is specific to a particular hardware platform of the receiver/decoder, code which is non-interpretive, and/or code which is directly executable by a microprocessor of the receiver/decoder. Thus, the structure taken by a piece of native code to be downloaded by a receiver/decoder will depend on the particular apparatus used in the hardware platform of that receiver/decoder. This is in contrast to "interpretive code", such as that known as "p-code" to the applicants, which requires interpretation by software stored in the receiver/decoder so that it may be executed by the microprocessor, and which therefore is functional over a wide range of hardware platforms. The data downloaded by the data loader may be in the form of native code, p-code, or any other suitable form, such as data tables.

By means of the above, a loader for loading the data from the bitstream is downloaded from the bitstream and stored in, preferably temporarily in RAM of, the receiver/decoder. Following downloading using the data loader of the required data

downloading of uncorrupted portions of data is not required. In an alternative embodiment, a portion of data stored in the receiver/decoder is replaced by a corresponding portion of, for example, a section of data transmitted with the downloaded data loader.

The bitstream may comprise at least one data loader, and, accordingly, the method may further comprise the steps, at a transmitting system, of:

for the or each data loader, dividing the data loader into a plurality of modules; and

for the or each data loader, dividing the data into a respective plurality of modules, each plurality of data modules being associated with a respective plurality of data loader modules.

Thus, the bitstream may include a plurality of data loaders and associated data. This can enable receiver/decoders having different hardware platforms to download the appropriate versions of the data loader and associated data. Data, such as an application, may conveniently be made up of a number of modules, which can be downloaded, and if appropriate run, as required.

The method may further comprise the steps, at the transmitting system, of:

for the or each data loader, formatting each of the modules as a respective table, the tables having the same respective table identification ("TID") and respective different table identification extensions ("TID-extensions"); and

for the or each plurality of modules of data, formatting each of the modules of data as a respective table, the tables having the same respective TID as the tables of the data loader modules associated therewith and respective different TID-extensions.

It is preferred that an MPEG protocol is used and, if so, the downloading steps may comprise downloading module MPEG tables.

The tables may have respective different TID-extensions other than a predetermined TID-extension; and the method may further comprise the step, at the transmitting

It is expected that receiver/decoders may be designed and manufactured by various different manufacturers. Each manufacturer may produce a number of different versions of the receiver/decoder. Receiver/decoders may therefore have various different hardware designs, though they will of course all conform to the same functional specification. It is therefore important that data, such as an application, behaves in the same way on every receiver/decoder, and that a receiver/decoder should execute all applications in the same, correct manner.

In order to ensure that the data is compatible with a particular version of a receiver/decoder, the bitstream may include a data loader and data for each version identification of the receiver/decoder, and the directory table having the predetermined TID can enable the TID of the modules of the data loader and data for each version identification of the receiver/decoder to be easily identified.

Preferably, the method further comprises the steps, at the transmitting system, of:

including in each transmitted directory table a directory version identification therefor; and

at the receiver/decoder:-

determining whether the directory version identification of a currently transmitted directory table is more recent than the directory version identification of a previously downloaded directory table having the same TID as the currently transmitted directory table, and if not, aborting the downloading of data.

The Bootstrap loader may be instructed, for example, by an application, periodically to download the directory table to determine whether the directory version identification of the previously downloaded directory table has changed. This can ensure that the receiver/decoder downloads promptly any updated data from the bitstream.

In order to avoid overwriting resident data stored in the receiver/decoder with identical received data, an application requesting updating of the resident data can choose to abort the downloading of data if a table directory is the same as that used in a previous updating of resident data.

Preferably, the method comprises the steps, at a transmitting system, of;

transmitting a second data loader included in said bitstream; and

at the receiver/decoder, of:

downloading the second data loader; and

downloading one of the first-mentioned data loader and the data; said transmitted downloading means performing the downloading of one of the first-mentioned data loader and the data.

In one embodiment, the second data loader is provided by another data loading program, at least part of the second loader preferably being in the form of native code.

This can enable downloading of a data loader to be avoided by using a different data loader previously downloaded from the bitstream. Thus, it is not necessary to download a data loader from the bitstream every time fresh or revised data is to be downloaded if a previously downloaded data loader is able to download the data as efficiently as the data loader. This can reduce significantly the time taken to download fresh or revised data from the bitstream. The second data loader may provide improved functionality over the first-mentioned data loader, for example, the second data loader may be able to download computer programs.

In a second aspect, the present invention provides a receiver/decoder comprising:

a receiver for receiving a bitstream including data;

storage means, such as a memory; and

downloading means for downloading from the bitstream into the storage means a loader for loading the data from the bitstream into the receiver/decoder.

In one preferred embodiment, the downloading means is provided by a boot program stored in the receiver/decoder.

The receiver/decoder may further comprise means for deleting the downloaded data loader from the storage means after the data has been downloaded from the bitstream. The deleting means may be provided by a central processor and associated software

means, such as a transmitter, for transmitting a bitstream including native code comprising at least one data loader for loading data into a receiver/decoder and data associated with the or each data loader; and

means, such as a data server, for dividing the or each data loader into a plurality of modules and dividing the data associated with the or each data loader into a respective plurality of modules for transmittal by said transmitting means.

Preferably, the transmission system further comprises:

means for formatting each of the modules of the or each data loader as a respective table, the tables of the or each data loader having the same respective table identification ("TID") and respective different table identification extensions ("TID-extensions"); and

means for formatting each of the modules of the data associated with the or each data loader as a respective table, the tables of the modules of data having the same respective TID as the tables of the data loader modules associated therewith and respective different TID-extensions.

The formatting means may be conveniently provided by the data server.

The tables may have respective different TID-extensions other than a predetermined TID-extension, and the system may further comprise means for generating a respective directory tables for the or each plurality of modules having the same TID, each directory table having that TID and said predetermined TID-extension, the directory containing for each of the modules a name of that module and the respective TID-extension.

The transmission system may further comprise:

means for generating a directory table having a predetermined table identification ("TID") and containing, for each of a plurality of version identifications of a receiver/decoder, a respective TID associated with that version identification.

The transmission system may further comprise means for including in each transmitted

Figure 8 illustrates the authentication of an MPEG table;

Figure 9 illustrates various areas of memory in a receiver/decoder of the television system;

Figure 10 illustrates a parameters field;

Figure 11 illustrates a hardware directory table;

Figure 12 illustrates a loader directory table; and

Figure 13 illustrates the procedure for downloading data.

An overview of a digital television system 1000 is shown in figure 1. The digital television system 1000 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

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. The receiver/decoder 2020 may also include a display 4038.

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, 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) 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

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 a procedure for loading modules and remain locked there until a specific call to a procedure for unloading modules is made.

With reference to Figure 5, 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,

sections in that table and a two-byte TID extension 34. Each section also includes a data part 36 and a CRC 38. For a particular module/table 24, all of the sections 18 making up that table 24 have the same TID 28 and the same TID extension 34. For a particular application 22, all of the tables 24 making up that application 22 have the same TID 28, but different respective TID extensions.

The authentication of an MPEG table will now be described with reference to Figure 8. The table 40 includes data 42 (typically comprising header 26, TID 28, TID extension 34 and data part 36), a key identification 44 and a cipher area 46. The key identification 44 comprises a 1-byte identification of a particular private key to be used to encrypt the block. The cipher area 46 comprises a block of 96 bytes of data. The first byte 48 is zero. A 16 byte signature 50 begins at an offset of typically between 0 and 31 bytes after the first byte. The signature 50 is produced using the known MD5 signature generating process on the data 42. Dummy data 52 is inserted between the first byte and the signature 50 and the block is encrypted using a known encryption process and the private key to which the key identification 44 corresponds.

If a plurality of MPEG tables are to be authenticated, then a directory listing the names of the tables and the signatures of those tables is included in the carrier signal. In the case of MPEG flow, this directory is transported in one single MPEG table, typically having a TID extension 34 of zero. The directory table is authenticated with the mechanism described above. Once the directory has been downloaded from the carrier signal it is possible for the application to download one or more of the MPEG tables listed in the directory.

The operation of the receiver/decoder 2020 in dealing with signatures and decryption during downloading of an application will now be described. Referring to figure 9, the receiver/decoder 2020 includes EEPROM 68, FLASH 69, ROM 70 and RAM 72. The EEPROM 68 includes a protected region 74 which is used by the virtual machine, and where only the virtual machine (and not a normal application) can write. The protected region 74 includes a key validation bitmap 76 of 16 or 256 bits, and an offset bitmap 80 of 32 bits. The ROM 70 includes, in one embodiment, sixteen public keys 82, in

03 APR 2002

All of the modules of the application can thus be downloaded in the manner specified above, and the application can be run by the receiver/decoder 2020.

The downloading of data into the receiver/decoder 2020 will now be described in more detail with reference to Figures 9 to 13.

The receiver/decoder 2020 includes loader 100, referred to as a "Bootstrap" loader 100, which is used primarily to download a loader for downloading software, such as manufacturer firmware, the run time engine 4008 and applications, present in the MPEG datastream for storage in the FLASH memory 69 of the receiver/decoder 2020. The Bootstrap loader 100 is stored in the FLASH memory 69 of the receiver/decoder 2020 and typically is not erasable therefrom. The Bootstrap loader 100 functions under the control of the hardware of the receiver/decoder 2020 and software stored therein.

Writing/updating of software stored in the receiver/decoder may be performed:

- at the request of the user of the receiver/decoder 2020;
- at the request of an application stored in the receiver/decoder 2020; or
- if the software previously stored in the receiver/decoder 2020 (referred to as "resident" software) has become corrupted.

To determine whether resident software has become corrupted, software written by the manufacturer of the receiver/decoder 2020 and stored therein calculates a checksum on the resident software data and compares this to a checksum written in the resident software. If the two values of the checksum are not equal, the resident software has become corrupted.

The FLASH memory 69 and EEPROM 68 of the receiver/decoder 2020 contain parameters which enable the Bootstrap loader 100 to download a loader in the form of native code from the bitstream. Parameters may be stored in the Bootstrap loader 100 itself, that is, in FLASH memory 69, or in EEPROM 68. Examples of parameters which may be stored in the FLASH memory 69 include:

- the frequency to which the transponder 2014 is tuned;

order to update the resident software.

The software downloaded into the FLASH memory 69 of the receiver/decoder 2020 may also contain a loader, referred to as a "Resident" loader. This loader should at least be able to perform a writing/update of software from the MPEG bitstream, and may offer other features, such as updates from local ports, and may allow the video and audio data in the MPEG bitstream to be decoded. The Resident loader is loaded from the bitstream at the request of an application, for example, to complement the loader which performs the downloading of the Instream loader, or for loading data from the bitstream. For example, if writing/updating is requested by an application stored in the receiver/decoder 2020 and the resident software is not corrupted, the Resident loader is used to perform the update instead of an Instream loader. This can reduce the time taken to update software in the receiver/decoder. At least part of the Resident loader is in the form of native code.

The various MPEG tables included in the MPEG bitstream which allow a receiver/decoder 2020 to locate and download the required software will now be described with reference to Figures 11 and 12.

The MPEG bitstream includes at least one hardware directory table 200 and a plurality of loader directories 300.

A hardware directory table 200 enables the Bootstrap loader 100 to locate the correct versions of the Instream loader and the software to be downloaded for a number of different versions of the receiver/decoder 2020. With reference to Figure 11, a hardware directory table 200 includes a TID 202 of D0 and a TID extension 204 of 0000, which values are previously stored in the EEPROM 68 of the receiver/decoder 2020 to enable, for example, the Bootstrap loader 100 to locate and download the hardware directory table 200.

The hardware directory table 200 includes:

a version number, HVERSION 206, of the hardware directory 200. The version

to the loader directory 300 associated with the HVN 210 of the receiver/decoder 2020, the Instream loader and the software to be downloaded.

With reference to Figure 12, each loader directory 300 associated with the HVN 210 of the receiver/decoder 2020 includes:

a version number, LVERSION 302, of the loader directory 300. The version number is increased each time there is a change in the content of the loader directory 300;

the number, NL 304, of MPEG tables of the Instream loader;

the version number, LVERS 306, of the Instream loader;

the number, NS 308, of MPEG tables of the software to be downloaded;

the version number, SVERS 310, of the software to be downloaded;

for each MPEG table of the Instream loader:

the identification, SEG_ID 312 of that MPEG table;

the TID extension, TID_EXT 314, of that MPEG table;

two redundant bytes, RES 316;

a maximum size, SECTION_LEN 318, of an MPEG section of that MPEG table;

a value, TIME_OUT 320, of the time-out for the loading of that MPEG table; and

the value, SGN_SIGN 322, of the signature of that MPEG table;

for each MPEG table of the software:

the identification, SEG_ID 324 of that MPEG table;

the TID extension, TID_EXT 326, of that MPEG table;

two redundant bytes, RES 328;

a maximum size, SECTION_LEN 330, of an MPEG section of that MPEG table;

a value, TIME_OUT 332, of the time-out for the loading of that MPEG table; and

the value, SIGN_SIGN 334, of the signature of that MPEG table;

an identification, KEY 336, of the private key used for the authentication of the loader directory 300; and

and the modules of the updated software are downloaded and mounted.

A front panel LED display 4038 of the receiver/decoder 2020 is adapted to display messages to the user of the receiver/decoder 2020 during the downloading of data. For example, the four following messages are specified in a parameter field stored in the FLASH memory 69 of the receiver/decoder 2020:

- a "LOAD" message, indicating that writing/updating is proceeding in a "normal" state, that is, at the request of an application;
- a "NATIV" message, indicating that updating is proceeding in a "native" state, that is, because the resident software has become corrupted;
- a "OOO" message, indicating that the Bootstrap loader 100 is unable to perform the writing/updating because it is unable to locate consistent or valid parameters (such as the frequency to which the MPEG tuner 4028 is to be set or the PID of the MPEG bitstream) in the memory of the receiver/decoder 2020; and
- an "ERRL" message, indicating that an error, other than those specified with reference to the OOO message, has occurred during writing/updating.

As an alternative to static messages, messages in the form of an animated cartoon may be displayed on the receiver/decoder.

The steps in, for example, the updating of resident software will now be described with reference to Figure 13.

In step S101, software stored in the receiver/decoder checks the integrity of any resident software by performing the checksum calculation and comparing the result of that calculation with the value of the checksum stored in the resident software. If the two values are different, then updating continues in the native state; if the two values are the same, or if no resident software is located, then updating continues in the normal state.

In this native state, it is then determined whether a previous update request is still pending in step S102. If such an update request from an application is pending, that request is erased in step S103 and step S102 is repeated. If no such update request is

downloaded before the time-out is reached, or if the loader directory is not authenticated (as an error has occurred during the downloading), the updating is aborted and the ERRL message displayed.

If downloading and authentication are successful, the Bootstrap loader 100 downloads the Instream loader from the MPEG bitstream into the RAM 72 of the receiver/decoder 2020 in step S110. If the Instream loader is not downloaded before the time-out is reached, or if the Instream loader is not authenticated (as an error has occurred during the downloading), the updating is aborted and the ERRL message displayed.

If the Instream loader is successfully loaded and authenticated, the Instream loader is executed in step S111 and, in step S112, the corrupted portion of the resident software is erased and the segments of the software to be downloaded are downloaded by the Instream loader, authenticated and written in the appropriate address location in FLASH memory 69. If the software is not downloaded before the time-out is reached; or if the software is not authenticated (as an error has occurred during the downloading) or if an error occurs during the writing of the software into the FLASH memory 69, the updating is aborted and the ERRL message displayed.

If the resident software is successfully updated, the writing of the report is stopped in step S113 and the receiver/decoder 2020 reset to enable a further updating to be commenced.

In any steps at which the updating is aborted, the step may alternatively be reperformed a prescribed number of times until successfully completed or until a time-out for performing that step is reached.

If updating is to proceed in the normal state, the Bootstrap loader 100 determines in step S201 whether an update request from an application is already pending. If not, the updating continues as normal. If there is already a pending update request, then the pending request is processed first.

the hardware directory 200 in step S208. If the hardware directory 200 is not downloaded before the time-out is reached, if the hardware directory is not authenticated (as an error has occurred during the downloading) or, depending on an option selected by the application requesting the updating, if a successful updating has been carried out using a hardware directory having the same HVERSION 206, the updating is aborted and the ERRL message displayed.

If downloading and authentication are successful and the updating is enabled by the application, the Bootstrap or Resident loader searches for an HVN 210 corresponding to the version number of the receiver/decoder 2020, as defined in a parameters field. If such an HVN is not located, the updating is aborted and the ERRL message displayed.

If such an entry is located, the Bootstrap or Resident loader reads the TID 212 of the MPEG tables used for the loader directory 300 associated with that HVN 210, the Instream loader and the software to be downloaded and, in step S209, downloads and authenticates the correct loader directory 300.

If the loader directory 300 is not downloaded before the time-out is reached, if the loader directory is not authenticated (as an error has occurred during the downloading), or, depending on an option selected by the application requesting the updating, if a successful updating has been carried out using a loader directory having the same LVERS 306, the updating is aborted and the ERRL message displayed.

If downloading and authentication are successful and the updating is enabled by the application, the Bootstrap loader 100 downloads the Instream loader from the MPEG bitstream into the RAM 72 of the receiver/decoder 2020 in step S210. If the Instream loader is not downloaded before the time-out is reached, or if the Instream loader is not authenticated (as an error has occurred during the downloading), the updating is aborted and the ERRL message displayed.

If the Instream loader is successfully loaded and authenticated, the Instream loader is

CLAIMS

1. A method of downloading data to a receiver/decoder, comprising the steps, at the receiver/decoder, of:

receiving a bitstream including the data;

downloading a loader specific at least in part for the hardware platform of the receiver/decoder, for loading the data from the bitstream into the receiver/decoder; and

downloading said data from the bitstream using said downloaded data loader.

2. A method according to Claim 1, wherein the downloaded data loader is deleted from the receiver/decoder after the data has been downloaded from the bitstream.

3. A method according to Claim 1, wherein the downloaded data loader is subsequently stored in non-volatile memory of the receiver/decoder.

4. A method according to Claim 3, wherein the non-volatile memory is a Flash memory volume of the receiver/decoder.

5. A method according to any preceding claim, wherein a portion only of data stored in the receiver/decoder is replaced by a corresponding portion of data downloaded by the downloaded data loader.

6. A method according to any preceding claim, wherein the bitstream includes at least one data loader, said method further comprising the steps, at a transmitting system, of:

for the or each data loader, dividing the data loader into a plurality of modules; and

for the or each data loader, dividing the data into a respective plurality of modules, each plurality of data modules being associated with a respective plurality of data loader modules.

7. A method according to Claim 6, further comprising the steps, at the transmitting system, of:

for the or each data loader, formatting each of the modules as a respective table, the tables having the same respective table identification ("TID") and respective different table identification extensions ("TID-extensions"); and

for the or each plurality of modules of data, formatting each of the modules of data as a respective table, the tables having the same respective TID as the tables of the data loader modules associated therewith and respective different TID-extensions.

8. A method according to Claim 7, comprising, in said downloading steps, of downloading module tables having the same TID.

9. A method according to Claim 8, wherein said tables have respective different TID-extensions other than a predetermined TID-extension; and further comprising the step, at the transmitting system, of generating a respective directory table for the or each plurality of modules having the same TID, the or each directory table having said predetermined TID-extension and that TID, the directory table containing for each of the modules a name of that module and the respective TID-extension.

transmitting system, of;

transmitting a second data loader included in said bitstream; and

at the receiver/decoder, of:

downloading the second data loader; and

downloading one of the first-mentioned data loader and the data, said second data loader performing the downloading of one of the first-mentioned data loader and the data.

18. A method according to Claim 17, wherein at least part of the second data loader is in the form of code which is specific to the hardware of the receiver/decoder.

19. A receiver/decoder comprising:

a receiver for receiving a bitstream including data;

storage means; and

downloading means for downloading from the bitstream into the storage means a loader specific at least in part to the hardware platform of the receiver/ decoder, for loading the data from the bitstream into the receiver/decoder.

20. A receiver/decoder according to Claim 19, further comprising means for deleting the downloaded data loader from the storage means after the data has been downloaded from the bitstream.

21. A receiver/decoder according to Claim 19, further comprising a non-volatile memory for storing the downloaded data loader after the data has been downloaded from the bitstream.

22. A receiver/decoder according to Claim 21, wherein the non-volatile memory is a Flash memory volume of the receiver/decoder.

23. A receiver/decoder according to any of Claims 19 to 22, wherein the downloaded data loader is adapted to perform the downloading of data from the bitstream.

24. A receiver/decoder according to any of Claims 19 to 23, wherein the downloaded data loader is adapted to replace a portion only of data stored in the receiver/decoder by a corresponding portion of data downloaded thereby.

25. A receiver/decoder according to Claims 19 to 24, arranged to download tables.

26. A receiver/decoder according to Claim 25, wherein said downloading means is arranged to download a table having a table identification ("TID") and a predetermined table identification extension ("TID-extension") so as to download a directory table, to determine from the content of the directory table the TID-extensions of module tables having the same TID as the directory table, and to download the module tables having the same TID as that of the downloaded directory table and TID-extensions determined from the downloaded directory table so as to download said loader.

27. A receiver/decoder according to Claim 25 or 26, wherein said downloading means is arranged to download a directory table having a predetermined TID and containing, for each of a plurality of version identifications of a receiver/decoder, a respective TID associated with that version identification, to determine the version identification of the receiver/decoder, and to download a directory table having a TID associated with the version number of the

37. A combination of a receiver/decoder according to any of Claims 19 to 30 and a transmission system according to any of Claims 31 to 36

38. A signal including at least one loader for loading data into a receiver/decoder wherein said at least one loader is specific at least in part to the hardware platform of the receiver/ decoder, and data associated with the or each loader, the or each loader being divided into a plurality of modules and the data associated with the or each loader being divided into a respective plurality of modules.

39. A method of downloading data to a receiver/decoder substantially as herein described.

40. A receiver/decoder substantially as herein described.

41. A transmission system substantially as herein described.