Record carrier, device for playing back a record carrier, method for playing back a record carrier, device for recording a record carrier and method for recording a record carrier.

The invention pertains to a record carrier.

The invention also pertains to a device for playing back a record carrier.

The invention further pertains to a method for playing back a record carrier.

Several standards are in development for digital storage and reproduction of audiovisual data such as the SVCD and DVD standard. SVCD describes storage of audiovisual data in the form of MPEG2-files, which may be reproduced under control of a so called Play Sequence Descriptor (PSD). The audiovisual data and the PSD are stored at the record carrier. The PSD is a set of control structures that enables the playback of preprogrammed sequences with user selection and interaction. User selection may for example take place by menus which are displayed on a display device, and from which the user may select an item by pointing at a position of the display device with a pointing device such as a mouse. Reading such a menu from the record carrier may delay the interaction speed.

It is an object of the invention to provide a record carrier, a device for reproducing audiovisual data from a record carrier and a method for reproducing audiovisual data from the record carrier which allow a higher interaction speed.

To that end a record carrier of the invention comprises information organized in a plurality of files containing audio visual data and playback control data for controlling playback of the audiovisual data on a playback device while enabling user interaction, the information further comprising priority information, indicating the relative priority with which the files are to be stored in a cache memory of the playback device.

A record carrier according to the invention is particularly suitable for playback on a playback device according to the invention. A playback device according to the invention comprises read means for reading the data from a record carrier, a cache memory for storing data from the record carrier, user input means for receiving user input, control

means for processing the control data, and reproduction means for reproducing the audio visual data, the device being adapted to read priority information and to store files into the cache memory with a priority determined by this priority information.

Preferably the priority information (priority) is related to an estimated frequency with which the said data is accessed. The priority information enables the playback device to store frequently accessed control data in its cache memory, while keeping less frequently accessed files at the record carrier. Such control data comprises for example menu items. This type of data is often used to request information from a user how to proceed with the playback. Loading such a menu item from the record carrier may take one or two seconds and therewith hampers the speed of interaction between the playback device and the user. By providing the priority information at the record carrier a playback device according to the invention is enabled to store this type of data in a cache memory and to therewith improve the interaction speed. The cache memory may be relatively small as the priority information indicates which information is the most relevant to store therein. The priority information preferably discerns at least 8 levels. This enables a playback device having only a relatively small cache memory to select only the most frequently accessed files for storage in the cache memory. If on the other hand such a record carrier is played back at a playback device having a greater cache memory, in addition to those files other files may be stored in the cache memory which are less frequently accessed, but still important.

It is possible to attach each item of priority information to the file to which it corresponds. Preferably however, the priority information is contained in a single file. This allows a playback device to retrieve the priority information with one file access.

An attractive embodiment of the playback device according to the invention is characterized by means for rewriting the priority information at the record carrier in accordance with the frequency with which the files are actually accessed by a user. This allows the playback device to adapt the record carrier to the profile of a particular user.

The invention also provides a method for playback of a record carrier comprising information organized in a plurality of files containing audiovisual data and playback control data for controlling playback of the audiovisual data while enabling user interaction, which method comprises the steps of

a. determining whether a cache memory does have sufficient unoccupied space,

aa. if it is determined that the cache memory does not have sufficient unoccupied space, then determining whether the cache memory does have space sufficiently large which

is occupied by one or more further data files having a priority value lower than the priority value of the first data file,

aaa. if the outcome of step aa is true, then overwriting said one or more further data files by the first data file.

A preferred embodiment of said method is characterised in that if the outcome of step a is true then

ab it is determined whether the first data file has a priority value higher than a predetermined value,

aba if the outcome of step ab is true then the first data unit is loaded from the record carrier into the cache memory.

The advantage of this embodiment is that it is prevented that files having a too low priority would be loaded into the cache memory. Such files generally would only stay relatively short into the cache memory as they would be overwritten by files having a higher priority.

The cache memory of a playback device according to the invention may also be loaded during an initialisation stage of the device by means of the following method for playback of a record carrier comprising information organized in a plurality of files containing audiovisual data and playback control data for controlling playback of the audiovisual data while enabling user interaction, which method comprises a procedure for loading files in a cache memory comprising the steps of

d setting a reference priority value,

e for a plurality of files examining whether a priority value assigned thereto is higher than the reference priority value,

ea if the outcome of step e is true examining whether the cache memory comprises sufficient space for storing the said data unit,

eaa if the outcome of step ea is true then loading said data unit into said space,

f reducing the reference priority value,

g determining whether the reference priority value is greater than or equal to a bottom priority value,

ga repeating steps d to g if the outcome of step g is true.

A record carrier according to the invention may be recorded with a device for recording a record carrier comprising information organized in a plurality of files containing audio visual data and playback control data for controlling playback of the audiovisual data

on a playback device while enabling user interaction, the information further comprising priority information, indicating the relative priority with which the files are to be stored in a cache memory of the playback device,

the playback device comprising

- means for obtaining the audiovisual data,
- means for composing the control data,
- priority determining means for determining the priority information,
- formatting means for formatting the audiovisual data and the control data into files,
- recording means for recording the information comprising the priority information, the audiovisual data and the control data at the record carrier.

A record carrier according to the invention may be recorded with a method for recording a record carrier comprising information organized in a plurality of files containing audio visual data and playback control data for controlling playback of the audiovisual data on a playback device while enabling user interaction, the information further comprising priority information, indicating the relative priority with which the files are to be stored in a cache memory of the playback device,

the method comprising the steps of

- obtaining the audiovisual data,
- composing the control data,
- determining the priority information,
- formatting the audiovisual data and the control data into files,
- storing the information comprising the priority information, the audiovisual data and the control data at the record carrier.

The playback control program may be recorded at the same information carrier together with the audiovisual data, but may otherwise be stored at a separate record carrier or in a ROM of the playbackdevice.

These and other aspects of the invention are described in more detail with reference to the drawing and with reference to an appendix: "Specification Proposal for a Video Disc Play Control Program (PCP)" which is incorporated by reference herein. In the drawing:

Figure 1 shows an embodiment of a record carrier according to the invention,

Figure 2 shows an embodiment of a device according to the invention,

Figure 3 shows an embodiment of a method according to the invention,

Figure 4 shows relations between several aspects of the invention,

Figure 5 shows a further embodiment of a method according to the invention.

Figure 1 shows a record carrier 1 which comprises information organized in a plurality of files 2 containing audiovisual data 11 and playback control data 12 for interactively controlling reproduction of the audiovisual data 11 on a playback device. The playback control data on the one hand controls playback of the audiovisual data and on the other hand allows for user interaction, for example by providing options from which the user can choose. In the embodiment disclosed here the files have a file structure as described in section 3 of the appendix. Part of the file structure is a file header (File_header) containing a file identification (File_ID). In an embodiment the audiovisual data maybe contained in one or more of the following four types of MPEG files. The first type is an MPEG movie file (File_ID = 10). This type of file contains a video stream, one or more audio streams and a private stream. A second type of file is an MPEG audio file (File_ID = 11). This type of file contains one or more audio streams, but no video stream. A third type of file is an MPEG picture file (File_ID = 12). It contains a video stream with one or more still pictures, but no audio stream. A fourth type of MPEG file (File_ID = 13), a so called MPEG show file comprising a video stream with one or more still pictures and one or more audio streams. Further the audiovisual data may comprise JPEG picture files (File_ID = 20) and bitmap image files denoted as BMP in the Figure (File_ID = 21). The playback control data 12 comprises one or more so called Play Control Program files (PCP, File_ID = 3), which may include a PCP-file for startup. Each PCP-file may contain one or more functional blocks FB. An FB on its turn contains one or more commands CMD1, ... CMDn, as described in section 4.5 of the appendix. Further the playback control data comprises a Play Item Table file (PIT-file, File_ID = 2). The structure of the PCP-file is described in detail in section 3.2 and the PIT-file is described in section 3.3 of the appendix. Furthermore the record carrier comprises a File Record Table file (FRT-file, File_ID = 1) as described in section 3.1 of the appendix. The FRT-file comprises information about all files at the record carrier 1, such as their size and address at the record carrier as well as general information about the record carrier. The record carrier is preferably a medium which is randomly accessible, such as a disc, as described in the appendix, or a card. The medium is for example an optical, a magnetic or a

magneto-optical medium. The information at the record carrier 1 further comprises priority information, indicating the relative priority with which the files 2 are to be stored in a cache memory of the playback device.

In the embodiment described here the priority information is contained in a single file, in this case the FRT-file. For each of the files 2, including itself, the FRT-file comprises an entry denoted as "priority" which has a value in the range of 0 to 15. The priority information is related to the estimated frequency with which the said data is accessed. A high value of priority represents a high estimated frequency. In the embodiment described here the control data is comprised in one or more control programs, the PCP-files which comprise references to menus, e.g. in the form of MPEG picture files. Preferably these files have a high priority, preferably the highest value (15).

Figure 2 shows a playback device according to the invention for playback of the record carrier 1 described above with reference to Figure 1. The playback device comprises read means 20 for reading the data from the record carrier 1. The read means 20 are provided with control means 21 to control the reading of data from the record carrier 1 upon instructions by processing means 30 for processing the control data. The data read from the record carrier 1 is transmitted via a bus 15. The device further comprises a cache memory 39 for storing data from the record carrier 1. In addition the device has a process memory 31 for storing data which is being used by the processing means 30. The device also comprises user input means 40, 41 for enabling user interaction. The user input means are for example a keyboard or a pointing device such as a mouse. The processing means 30 interactively select audiovisual data from the record carrier 1 and have these reproduced by audio reproduction means 53, 54 comprising an MPEG-audio decoder 53 and video reproduction means 51, 52 comprising an MPEG-video decoder 51. The device according to the invention is adapted to read priority information and to store files in the cache memory 39 in accordance with this priority information. To this end the device reads the priority information from the FRT-file and if the cache memory 39 is too small to store all files, the device stores those files in the cache memory 39 which have the highest priority value. The processing means 30 comprise interpreter means 32 for interpreting commands CMD1, ... in the functional blocks FB of the play control program file PCP. The interpreter 32 sequentially interprets the commands and writes/reads a plurality of Player API registers, also denoted as Registers, which function as an application programmers interface. These Registers and their function are described in section 4.3.2 of the appendix. For example upto 16 hotspot areas may be defined by writing the Registers 128-191. A next PCP-file may be selected by writing its file number in the

Register numbered 226. A play_item, for example an MPEG file may be selected by writing an identification of said item in Register 228. Note that most Registers correspond to a separate register for writing and for reading. By reading a Register status information returned by the executing means 34 may be retrieved. For example, if the Register numbered 228 is read a so called play event ID value may be retrieved. The latter indicates a cause for ending the playback of the MPEG-file. A more detailed description is given in section 4.4.6 of the appendix. The interpreting means and executing means of the processing means may be incorporated in software on a single processor or may be implemented as different processors. The mutual relation of the different files is shown in Figure 4. When a file number of a PCP-file is written in register 226, the execution means 34 retrieve the file address of said PCP-file form the FRT-file. If this file is not already available in the cache memory 39, it is loaded from the record carrier 1 into the process memory 31. If the Play Control Program file writes an identification of a Play-item in the register 228, control is passed to the execution means 34. These look up the corresponding filenumber of an audiovisual file and the entry point within said file in the Play Item Table PIT and have said play-item reproduced. The file address of said audio visual file is retrieved from its file number by means of the FRT-file. If control is passed to the execution means 34 then operation of the interpreting means 32 is suspended until an event is detected by event handler means 35. The event handler means 35 may detect at least one of the following events, the expiration of a time interval, an input by a user, the occurrence of trigger data in the audiovisual data. This is illustrated in Figure 1 of the appendix. A command set Timer, by writing a value to Register 252, has the effect that a timer 35 starts to count down from said value and that a timer event occurs if count down is completed. A trigger event occurs if a trigger detector 37 detects that trigger_data corresponding to a trigger_mask loaded in Register 249 occur in an MPEG stream which is read from the currently selected file. A user may cause an event in several ways. In the first place one or more hot spots may be defined by writing a selection or all of the registers 128 to 191. If the user points to a hotspot, a hotspot event occurs. Otherwise a userkey event occurs if a key is pressed which is enabled by the userkey_mask. The latter is assigned to Register 250. If the reproduction of the MPEG file is completed likewise an event, End_of_File event, occurs. If an event occurs the event handler means 35 enable the interpreter means 32 to continue the interpretation with a command (Command #n+1) succeeding the command (#n) at which the operation of the interpreter means 32 was suspended.

If a file is accessed, the procedure illustrated in Figure 3 is followed. First in step S1 it is examined whether the file to be accessed is already in the cache memory 39. If this is the case, then in step S2 the file is copied from the cache memory 39 into the process memory 31. Subsequently the file can be processed in step S3. If the file to be accessed is not available in the cache memory 39 then it is loaded from the record carrier 1 into the process memory 31 in step S4. Next it is examined in step S5 whether in the cache memory 39 enough free space is available to store the file. If the outcome is positive the file is copied from the process memory 31 into the free space of the cache memory 39 in step S7.If the outcome of the examination in step S5 is negative the program step S6 is executed. Therein it is examined whether enough space in the cache memory 39 is available which is occupied by files having a lower priority than the file which was accessed. If this is the case than that file is copied from process memory 31 into the said space in the cache memory 39 (step S7). Otherwise the said file is not copied into the cache memory 39.

An embodiment of the device according to the invention also includes a procedure for loading the cache memory 39 during initialization. Such a procedure is described with reference to Figure 5. In a first program step S10 a priority reference value PR is initialized at 15. In a second program step S11 a filenumber I is initialized at 1. Next in step S12 it is examined whether the file with file number I has a priority equal to PR. If this is not the case than the procedure continues with step S15. Otherwise the procedure continues with step S13 wherein it is examined whether the size of the file with filenumber I is less or equal than the free space in the cache memory 39. If this is true than said file is loaded into the cache memory 39 in step S14. If it is not true than the procedure continues with step S15. wherein the filenumber is increased by 1. Subsequently in step S16 it is examined whether the filenumber is less or equal than the total number of files. If this is true than the procedure continues with step S12. If this is not true than the priority reference value is decreased by one in step S17. Subsequently it is examined whether the new priority reference value is greater than or equal to 0 in step S18. If this is true than the procedure continues with step S11. If it is false then the procedure is completed.

It is remarked that the scope of protection of the invention is not restricted to the embodiments described herein. For example, while an embodiment of the apparatus according to the invention is described which comprises detection means to detect which type of record carrier is present, an other embodiment comprises input means which enable a user to provide that information. The apparatus may comprise recording means for recording an information stream on the record carrier in addition to the reading means. Neither is the scope

of protection restricted by the reference numerals included in the claims. The word 'comprising' does not exclude other parts than those mentioned in a claim. The word 'a(n)' preceding an element does not exclude a plurality of those elements. The invention further resides in each new feature or combination of features.

5

Specification Proposal for a Video Disc Play Control Program (PCP)

By Erik Schylander

version 3-Nov-99 11:43

A Play Control Program (PCP) is used for the user navigation when playing the disc. The user can select the parts to play from presented menus. It is applicable to any 2K sector addressable media, e.g. CD-ROM, DVD-ROM.

In the current VCD and SVCD specifications a Play Sequence Descriptor (PSD) file is used for the PCP with a fixed syntax and semantics. It has the drawback that it is static (pre-determined) and can not adopt to the input of user. No conditional testing or computations are supported. The Playback is programmed by recorded Play Lists and Selection Lists that define the MPEG streams to play. In SVCD an extension was introduced called Command Lists which allowed conditional testing and some limited computations.

This proposal is based on about the same concept, but without using Play Lists or Selection Lists. It also supports event handling triggered by a MPEG stream playback point, a timer or a user input.
This is also very different from DVD, where the control and navigation data is stored in the MPEG stream, which makes the authoring and testing very complicated.

Revision history

rev#1d - 26/10/99	NVRAM size allocated by Disc_info, and fixed location at Reg[0]. Fixed type errors, updated figure 1, outline changes
rev#2a – 27/10/99	Simplified Player registers to avoid MSB, LSB bytes (Hotspots.OGT. Audio_stream), and reallocated register numbers. Added JPEG and BMP file types values. Added as extension OGT rendering in a hotspot area.
Rev#2b – 1/11/99	Changed name of DRT to FRT, and PIN to PIT to use more correct wording. Added Player States (stopped, playing, paused, scanning). Added field in MPEG info record for PlayingTime. Added field in MPEG info record for average sector (bit) rate.
Rev#2c - 2/11/99	Filled in Error codes Added place holders for Systems Overview

Cc: D. de Jong, CP&T.

Table of Contents

1 GENERAL 12

1.1 NORMATIVE REFERENCES 12
1.2 CONVENTIONS 12
1.3 ABBREVIATIONS AND GENERAL DEFINITIONS 12
1.4 DATA TYPES AND STRUCTURES 13
1.5 SYSTEM OVERVIEW 14
1.5.1 Player Reference Model 14
1.5.2 Disc Reader Module 14
1.5.3 MPEG decoder 14
1.5.4 Overlay Graphics and Text (OGT) 14
1.5.5 Playback Controller 15
1.5.6 Cache Memory 15

2 DISC STRUCTURE 15

3 FILE STRUCTURE 15

3.1 FILE RECORD TABLE (FRT) FILE 16
3.2 PLAY ITEM TABLE (PIT) FILE 17
3.3 PLAY CONTROL PROGRAM (PCP) FILE 17
3.4 MPEG FILES 17
3.4.1 MPEG file characteristics 17

4 PLAY CONTROL PROGRAM (PCP) 18

4.1 PCP FILE STRUCTURE 18
4.2 PCP INTERPRETER 18
4.2.1 PCP structure 18
4.2.2 PCP Interpreter reference model 19
4.2.3 FB program flow structure 19
4.2.4 Player States 20
4.3 PCP REGISTERS 20
4.3.1 Variable registers, R[0..95] 20
4.3.2 Player API registers, R[128..255] 21
4.4 PCP PLAYER (API) FUNCTIONS 23
4.4.1 FB local variables memory(size) function 23
4.4.2 FB(address) functions 23
4.4.3 Goto_PCP(File_number) function 23
4.4.4 Goto_Disc(Disc_number) function 24
4.4.5 Play() function 24
4.4.6 Play_event() 25
4.4.7 MPEG file info function 26
4.4.8 OGT_select(channel) function 26
4.4.9 Audio_select(stream number) function 26
4.4.10 Audio_mixing(data) function 26
4.4.11 Hotspot() function 26
4.4.12 Userkey() functions 26
4.4.13 Timer(time) function 27
4.4.14 Error handling 27
4.4.15 OGT_rendering function 28
4.5 PCP COMMAND DEFINITIONS 28
4.5.1 Command codes 29
4.5.2 operand symbols 29
4.5.3 opcode conditions 29
4.5.4 opcode function definitions 30
4.5.5 Programming examples 30

General

Normative References

The following Standards contain provisions, which, through reference in this document, are used this document.

ISO 646	7-bit coded character set for information interchange
ISO 9660	Volume label and file structure (ISO9660) of CD-ROM for information interchange
ISO/IEC 11172-3:1993	Information technology—coding of moving picture and associated audio for digital storage media up to about 1,5Mbit/s Audio (MPEG1)
ISO/IEC:13818-1,-2,-3 2nd edition:1996/1997	Generic coding of moving picture and associated audio information (MPEG2); the 1997 edition shall apply for audio coding
ISO/IEC: 10918-1	Digital compression and coding of Still Images (JPEG)
IEC:13346	UDF 2.0 as defined by OSTA (Universal Disc File system)

Conventions

Bold terms are defined in this document
Values and comments are written in *italic*.
Hex values are preceded by a $ character
Binary values are preceded by % character
Strings are within double quotation marks, as "....."

Abbreviations and general definitions

The following abbreviations are used in this document.

 # or ## Abbreviation for " Number of ".

Album An **Album** is a collection of discs with the same **Album_ID**.

Album_ID An uniquely value that identifies all discs belonging to an Album.

Audio stream The MPEG stream can contain one or more audio streams numbered 1 to max. It is optional to support surround sound in the Extended stream.

Bitmap image An image representation where each pixel is represented by on or more bits. These bits represent different color values in a color lookup table (CLUT).

CLUT A Color Look-Up Table, defining a color value (RGB) for each entry.

FRT File Record Table, see 0

Hotspot A Hotspot is defined by the ID number and the area coordinates on the screen. Hotspots are used for user interaction on the screen, so a user can select a hotspot (by cursor or pointer control) to interact with the PCP. The selection of a Hotspot generates a User_input event. It can be used by (computer) playback systems that use pointing devices for user interaction, or for systems that can highlight selected areas. The area within the indicated co-ordinates should be highlighted. The player can chose frame thickness and color. Such highlighting is recommended to have two views, one when pointer inside area or one when selected. If more hotspot areas contain the point selected, (in the case of overlapping selection areas) then the selection area with the lowest Selection Number is selected.

Hotspot area The hotspot coordinate area defines a rectangular Selection Area on the screen. A scaled coordinate system is used such that the extreme upper left screen co-ordinate is 0,0 and the extreme lower right screen co-ordinate is 255,255.

I-picture	For I-field pictures, i.e. when a video frame is encoded in the form of two fields, the sector containing the start of the I-picture with the first of both fields shall be referred to (even when the 2nd field is closer to the expected grid).
JPEG	Joint Photographic Experts Group, See ISO/IEC 10918.
lsb	Least significant bit.
LSB	Least Significant Byte.
MPEG	Moving Picture Experts Group, see ISO/IEC 11172.
MPEG_PS	A MPEG-2 Program Stream, that can contain a video stream, zero or more Audio streams and a MPEG private stream.
MPEG private stream	A MPEG private stream that contains the OGT data and Trigger_data
msb	Most significant bit.
MSB	Most Significant Byte.
N/A.	Not Applicable.
NVRAM	A Non Volatile memory in the Playback device to permanent store a part of the Global variable registers for a disc, and shall be recalled when a disc belonging to the same album is played again. This is an optional feature of the playback device. unction that is called when a register is set. The playback device shall uniquely allocate the NV-RAM memory for each disc by using the Album ID as unique identification.
OGT	Data that define the Overlay Graphics and Text function for the MPEG decoder. The MPEG stream can contain in the Private stream up to 8 OGT channels numbered from 1 to max.
PIT	Play Items Table, a list of all MPEG file entry points (access sectors).
PCP	Playback Control program. The program is recorded in a form that is very compact and easy to implement. It's compiled and optimized during the authoring process. The playback device (player) uses the PCP for navigation and playback selection.
Playback device	A compliant device that contains a disc loader with optical pick-up unit and servo controller, MPEG_decoder, and Playback processor with user I/F, player control and PCP interpreter.
PS	A MPEG Program Stream, a collection of MPEG Elementary streams.
RT	Real Time, a classification for (MPEG) files that has to decode data for Real Time output.
R/W	Read and Write, used to define the characteristics of the PCP registers.

Data types and structures

The following data types are used in this specification:

Bit	A binary digit that can have value %0 or %1. Bits are numbered as b#*number*
n-bit data	A stream of n number of bits, where the first bit =b#*n-1* and the last bit=b#*0* (msb)
True	A logical state of a n-bit data value, where one or more bits = %1
False	A logical state of a n-bit data value, where all bits = %0
Byte	A 8-bit data value, that represent the values 0..255

Zero_byte	A byte with the value of 0 ($00)
Reserved_byte	A Zero_byte reserved for future use;
Character	A byte representing a Character, coded as ISO646
Digit	One of the Characters "0123456789"
Number	A positive value represented by a **String** of **Digit**
N-byte integer	A signed value of N bytes recorded in 2's complement format
N-byte word	An unsigned value of N bytes

Item	Any data as an array[0..*last_byte*] of byte, where the **Item_size** = *last_byte+1*;
<Item>	Optional items are enclosed by < >
Item_type	One of the following data types: Boolean: False..True Byte Character N-byte integer N-byte word String[] Array[] Record
Array[*0 .. last_item* **] of Item_type**	A collection of consecutive indexed items, from 0 to the last item, of the same Item_type.
Index	A n-byte word, used to indicate or reference an item in an array
String[*length* **]**	An array of up to max. *length* number of character's. The string is padded with Zero_byte's for not used characters. An empty string contains only Zero_byte's.
Field_item	A data structure defined as **Struc{ Item_name <.Item_name> of Item_type }**
Record	A data structure defined as **Struc{ Field_item <; Field_item> }** These items form a block of consecutive bytes, recorded in sequential order.
Table	An **Array[** 0.. last_record] of **Record**

System overview

This is a disc system to present full motion pictures with associated audio by using any ROM (2K addressable sector) disc format. The system uses the ISO/IEC MPEG2 standard definitions to compress the video, still picture and audio information for full screen TV quality pictures together with associated high quality audio. The disc also contain high quality still pictures (menus) and a Playback Control Program file for navigation and playback control of the disc.

Player Reference Model

The system consists of a disc and a compliant playback device (player).

Disc Reader Module

MPEG decoder

Overlay Graphics and Text (OGT)

Playback Controller

Cache Memory

This is an optional feature in a playback device. To access a file in a player can take up to 1 or 2 seconds, but by storing frequently used files in a Cache RAM memory the access time can be neglected. When a MPEG file is read for the first time, the playback device can decide to store the file in the cache depending on the priority value and how full the cache is. The priority value gives an estimated likelihood that the file will be used again. This is specially useful for Menu still picture files, which normally have a size of 1-200KB. The cache memory need to be 1MB or more, but with the falling memory prices this will only add a marginal cost to a player.

Disc structure

Disc	An **Array** [0..*last_sector*] of **Sector**.
Sector	An **Array**[0..2047] of **Byte**. User data is recorded in the Sector.
Sector_addres	A 3-byte word value representing the **Sector** index number in the **Disc** array.
Sector_size	The sector size is fixed to 2048 bytes.

File structure

File	User bytes are stored on the disc as an **Array**[0 .. *last_byte*] of byte, indexed by **File_byte_position**. The **File** is recorded in the consecutive sectors as defined by **Disc**[**file_start_address .. file_end_address**] of **Sector**. The last sector is padded with zero_byte's.
File_start_address	The **sector_address** value of the first sector of a file.
File_end_address	The **sector_address** value of the last sector of a file.
File_size	A 3-byte word value representing the number of **Sectors** recorded on the disc that the file occupy. The value is = (*File_end_address – File_start_address+1*). Note that the number of bytes in the file is = (File_size * Sector_size).
File_byte_position	An index of a byte in File[0 .. *last_byte*].
Structured_file	A file starting with a **File_header**, followed by an Array of Items, indexed from 0 to last_item. The item structure is depending on **File_ID** value, and is defined below for each specified File_ID. The file structure = **struc{ File_header**; array[0..*last_item*] of **Item** }
File_header	A **record** that contains information about the structured_file defined as **Struc{ File_ID, Item_offs, Item_size, Item_count** of 2-byte word; <, **Info** of **Array**[0..*last_byte*] of byte > }.
File_ID	A word identifying a specific file. Specified file_ID's are: 0 = unspecified file 1 = **File Record Table (FRT)** file 2 = **Play Item Table (PIT)** file 3 = **Play Control Program (PCP)** file 10 = **MPEG movie** file 11 = **MPEG audio** file 12 = **MPEG picture** file 13 = **MPEG show** file 20 = **JPEG** picture file 21 = **Bitmap image file** (.BMP) Other values are reserved for future use.
Item_offs	A 2-byte word for the **file_byte_position** of the first Item with index = 0.
Item_size	A 2-byte word for the number of bytes an Item contains.

Item_count	A 2-byte word for the number of items in the file = (*last_item*+1).
Info	An optional field to store specific information.

File Record Table (FRT) file

A structured file that contains a **Table** of **File_directory_records** indexed by a **File_number**. The record contains information about where the files are located on the disc. The default **File_number** of this file = 0.

UDF file name	= "FRT.BIN"
File_header	The file_header is defined as: **File_ID** = 1 **Item_offs** = 24 **Item_size** = 16 **Item_count** = number of **File_directory_records** **Info** = **Disc_info** of 16 bytes
Disc_info	A 16 byte Record that contain information about the disc, defined as **Struc{** **Album_ID** = 4-byte **word;** **Disc_ID** = 2-byte **word;** **Disc_count** = 2-byte **word;** **NVRAM_size** = 1-byte **word;** **Reserved** = 7 **Zero_bytes** }.
Album_ID	A unique number that is the same for all discs that belong to the same Album.
Disc_ID	A sequential number starting from 1 that identifies the sequential order of the disc in the Album.
Disc_count	A 2-byte word that contains the total number of discs in the Album.
NVRAM_size	A 1-byte value that defines the number of Global Variable registers, Reg[0 .. *NVRAM_size-1*], that shall be permanently stored into the **NVRAM** of the playback device when the disc is stopped by an **End()** or **Goto_disc()** function. At startup of a disc the **NVRAM** data shall be loaded into the same specified Global Variables again. If the **NVRAM_size** = 0 then no action required. The playback device support of this NVRAM feature is optional.
File_directory_record	A 16 byte record of 4 fields that contain file information data, defined as: **Struc{ File_address, File_size** of 3-byte word; **File_ID** of 2-byte word; **File_info** of 8 byte record }. The **File_ID** = the File_ID value of the referenced file.
File_address	A 3-byte word **sector_address** of the first sector of the file.
File_info	A 8 byte record that contains information about the file, defined as **Struc{** **Priority** = 4-bit **word** (0..15); **Restriction_category** = 4-bit item. (t.b.d) **Video_type** = 8-bit item; (*PAL / NTSC, full- / sub-picture*) **Audio_streams** = 8-bit item; (see 0 and 0) **OGT_streams** = 8-bit item; (see 0) **Playing_time** = 2 byte item; (see 0) **Sector_rate** = 1 byte item, that for RT files, indicates the average number of sectors (in units of 16K or 8 sectors) per second. For non RT files this value is 0. (see 0) *This value is used by the Playback controller to estimate the entry point into a MPEG file for a scan function or jump to time (seconds) function, by calculating the sector offset as (Sector_rate * time).* **Reserved** = 1 Zero_byte }.
Priority	A value 0..15 that indicates the frequency distribution a file is expected to be accessed, where the value of 15 is used for the most frequent files, e.g. Menu's. This value can be used to support a cache memory feature in the playback device to improve the playback performance, see 0.
File_number	A 2-byte word Item index of the **File_directory_record** in the **FRT** file

Play Item Table (PIT) file

A structured file that contains **Play_item_records.** The default **File_number** of this file = 1.

UDF file name	= "PIT.BIN"
File_header	The File_header is defined as: **File_ID** = 2 **Item_offs** = 8 **Item_size** = 5 **Item_count** = number of Play Items.
Play_item_record	A 5 byte record that contains location information of **Play_items**, defined as **Struc{** **File_number** of 2-byte word; **Access_Point** of 3-byte word }
PIN	A 2-byte word Item index of the PIT file. It is used by the PCP player function.

Play Control Program (PCP) file

A structured_file of sequentially stored **FBs** that together represent a Playback Control Program. For the PCP file structure see section 0. A **PCP** file is referenced by its **File_number.** The default **File_number** for the startup PCP file = 2.

UDF file name	= "PCPnnnnn.BIN", where nnnnn is a 5-digit zero preceding **Number** equal to the **File_number** in the **FRT** file
File_header	The file_header is defined as **File_ID** = 3 **Item_offs** = 8 **Item_size** = 5 **Item_count** = number of **Commands**. Item_count shall be less than 64K.

MPEG files

MPEG_file	A file containing a MPEG_PS.
MPEG movie file	A MPEG_file with a motion picture video stream and one or more audio streams and a private stream. The File_ID = 10
MPEG audio file	A MPEG_file with one or more audio streams, and no video stream. The File_ID = 11.
MPEG picture file	A MPEG_file with a video stream that contain one or more still pictures and no audio stream. The File_ID = 12.
MPEG show file	A MPEG_file with a video stream that contain one or more still pictures and one or more audio streams. The File_ID = 13.

MPEG file characteristics

Play_item	A part of a MPEG_file, starting at an access_point, and ending at the end of the MPEG PS. The playback of a Play_item can conditionally end at any trigger_point checked by the PCP interpreter. A Play Item can be the whole or part of a MPEG file containing: • MPEG Video; with or without MPEG Audio • One, or more, MPEG encoded Still Pictures; with or without MPEG Audio.

	• MPEG Audio only with no pictures or video. If a Play Item includes one or more still pictures, the last picture is displayed until it is replaced by a new Play Item.
Access_point	A 3-byte reference address to a **MPEG_file** sector that contains the beginning of a MPEG **I-picture** or for a **MPEG audio** file the beginning of a audio frame. *These are the entry points where the PCP program can start playing in a MPEG file.*
Trigger_point	A 3-byte reference address to a **MPEG_file** sector that contains **Trigger_data** stored in the Private Stream. When the playback_device is playing a MPEG_file and passes a **Trigger_point** then it should check if a **Trigger_event()** function should be called.
Trigger_data	A 16-bit data value in the MPEG private stream, which can be filtered by a **Trigger_mask** to decide if a **Trigger_event()** shall be called.

Play Control Program (PCP)

The PCP contains the navigation structure of the disc. The PCP is stored in PCP files. The file is divided into functional blocks called **FB**. At startup of the disc the PCP file with File_number = play, the **PCP interpreter** starts to interpret the first **FB** at address = 0, and continue until an **End()** function is called.

PCP file structure

Command	A 5-byte item that contains the PCP interpreter instruction code. Commands are referenced from 0 to last Command in the **FB**.
Command_index	A 2-byte word Item_index in a **FB**. The **Command_index** is = 0 for the first Command in a **FB**.
FB	A block of consecutive Commands, that represent a **FB function()** that can be called by setting the **FB_address** register. The size of a FB block shall be less than ?? Commands.
FB_address	A 2-byte word Item_index of the first Command in the **FB**. The address of the first **FB** in the **PCP** file = 0.

PCP Interpreter

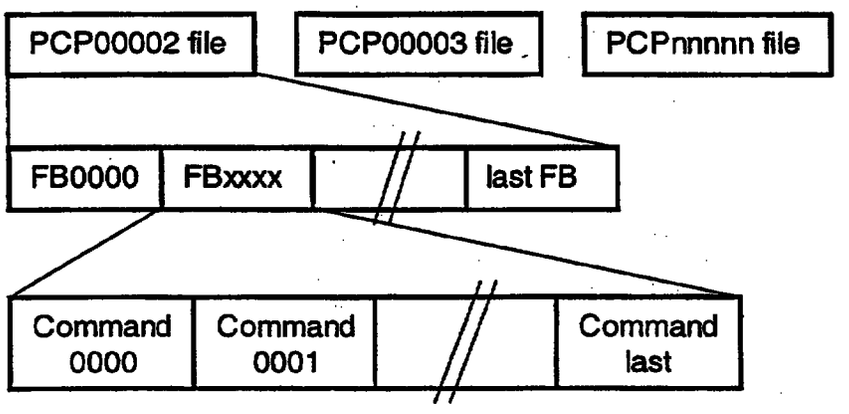
The **PCP interpreter** shall interpret and execute the Command instructions in the **FB**. These Commands shall be interpreted in sequential order one by one, until a **Return()** or an other function is called. If an error is detected the **Error_handler** shall be called. Most Commands will only be executed if a specified Command condition is fulfilled.

The **PCP interpreter** is easy to implement with a standard micro-controller, as it needs only to handle three simple type of operations:

- Read / Write of variable and register data
- Arithmetic operations on variables
- Command interpreter sequence management

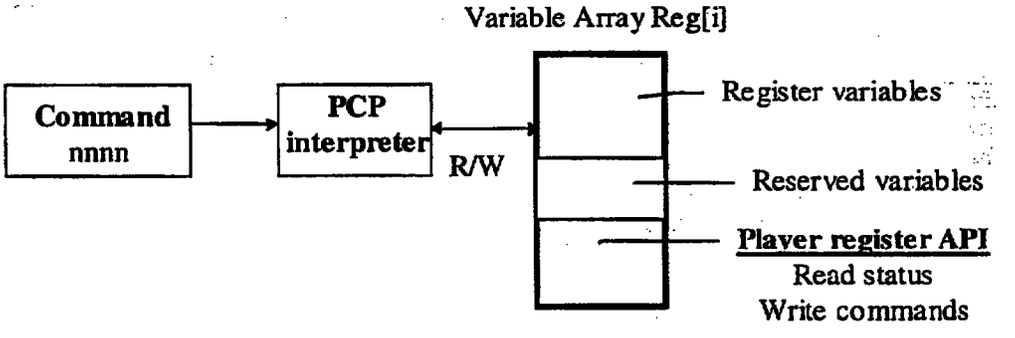
PCP structure

The relation between the different PCP data elements are shown in the figure below.



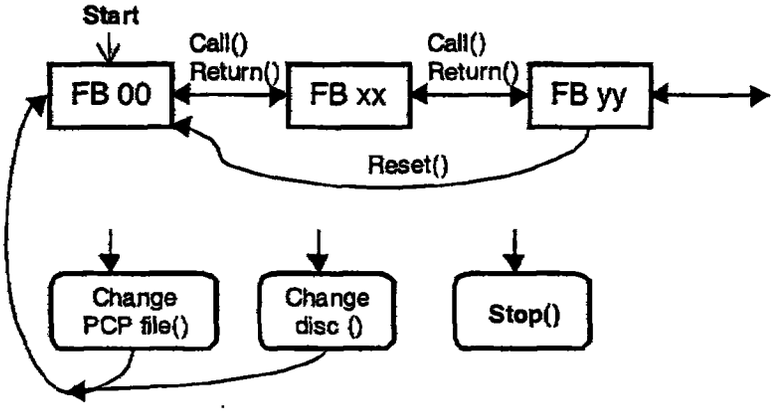
PCP Interpreter reference model

The PCP interpreter reads the commands in the FB one by one and executes the command instruction as defined in section 0. The command reads and writes data into the Variable Array. Values written into the register variables shall be stored in memory. If a value is written into the Player register part then a specific function() is called and shall be executed before the next command is interpreted. See the figure below.



FB program flow structure

The control flow structure between FB's, PCP files, and discs is shown in the figure below.



Player States

The MPEG player in the MPEG device can be in one of the states as described below. These states are important for the Play() function, see 0

State	Description
Stopped	The MPEG player stopped decoding the MPEG stream, either by a Play() command or by an End_of_File event. The last MPEG picture should remain on the screen.
Playing	The MPEG player is active decoding the current MPEG stream.
Paused	The MPEG player can be paused during Playing directly by the user. It does not effect the Interpreter as it is halted waiting for an Event(). When the user wants to continue the Player shall return to the Playing state.
Scanning	The MPEG player can scan the MPEG stream (file) forward or reverse during Playing directly by the user. It does not effect the Interpreter as it is halted waiting for an Event(). In the MPEG info record there is a field that gives the average bitrate to guide the player on how many sectors to jump per second. When the user wants to continue the Player shall return to the Playing state.

PCP registers

The **PCP interpreter** uses an **array R[0..255]** of 2-byte (16-bit) registers, that contain variables and player registers that functions as an API for the PCP interpreter to interface with the player functions and MPEG decoder. The PCP_registers can be read and written to by the PCP interpreter. The 256 registers are indexed from 0 to 255. The player shall set the defined player registers reflecting the current player status to be read by the PCP interpreter. At start up of the disc the default player values should be used. Some players allow the user to change the default values. The PCP can control the player by writing a value to a R/W player register, and the player has then to execute controls that would generate the same status. Note that a R/W player register actually consists of two different registers, one for read of the player status, and one for giving a command to the player. When the player is busy executing a register command, then these values could be different.

Variable registers, R[0..95]

Index	Register	Description
0 .. 31	**Global variables**	These 32 registers, indexed 0 to 31, are used for storing global accessible variables. At startup these variables shall not be set, except if the NV-RAM function is

		supported, those that has been stored in the NV-RAM.. At read return the set value. At write set value.
32 .. 63	**Parameters**	These 32 registers, indexed 32 to 63, are used for passing parameters to a FB function, and for returning parameters from a FB function. Before calling a FB function intermediate results should be saved in the local FB variables. At startup the variables should not be set, and shall keep its values during the change of a disc. At read return value. At write set value.
64 .. 95	**Local FB variables**	Up to 32 registers, indexed 64 to 95, can be allocated to be used locally by the called FB function or the error handler. These variables are allocated by setting the **FB var_size** register. Allocated local FB variables shall be saved in memory before the new FB function starts. Local FB variables of the calling FB function shall be restored from memory by the Return() function. Allocated variables shall be set to 0. At read return the set value. At write set value.
96 .. 127	**reserved**	Reserved for future extensions.

Player API registers, R[128..255]

Index	Register	Description
128 .. 143	**Hotspot_area_top** **ID = #128 .. #143**	At startup the register shall be set to -1 if Hotspots are supported by the playback device else it shall be set to -2. At read return the vertical position of the **top** line of the hotspot(#ID) selection area. ID values 128 ..143 are mapped to register values 128 .. 143. If the hotspot is disabled then return -1. See 0 At write set the **top** position for the hotspot(#ID). If this register is set to -1 then the hotspot is disabled.
144 .. 159	**Hotspot_area_bottom** **ID = #128 .. #143**	At startup the register shall be set to -1 if Hotspots are supported by the playback device else it shall be set to -2. At read return the vertical position of the **bottom** line of the hotspot(#ID) selection area. ID values 128 .. 143 are mapped to register values 176 .. 191. If the hotspot is disabled then return -1. See 0 At write set the **bottom** position for the hotspot(#ID). If this register is set to -1 then the hotspot is disabled. The hotspot **bottom** value shall be >= **top** value.
160 .. 175	**Hotspot_area_left** **ID = #128 .. #143**	At startup the register shall be set to -1 if Hotspots are supported by the playback device else it shall be set to -2. At read return the horizontal **left** most pixel position of the hotspot(#ID) selection area. ID values 128 ..143 are mapped to register values 192 .. 207. If the hotspot is disabled then return -1. See 0 At write set the horizontal **left** most pixel position for the hotspot(#ID). If this register is set to -1 then the hotspot is disabled.
176 .. 191	**Hotspot_area_right** **ID = #128 .. #143**	At startup the register shall be set to -1 if Hotspots are supported by the playback device else it shall be set to -2. At read return the horizontal **right** most pixel position of the hotspot(#ID) selection area. ID values 128 ..143 are mapped to register values 208 .. 223. If the hotspot is disabled then return -1. See 0 At write set the horizontal left most pixel position for the hotspot(#ID). If this register is set to -1 then the hotspot is disabled. The hotspot **right** value shall be >= **left** value.
192 .. 223	**reserved**	
224	**Locals_size**	At startup the register shall be set to 0. At read return the current FB local variables size. At write call the **FB local variables memory(size)** function. See 0.
225	**FB_address**	At startup the register shall be set to 0.

		At read return current **FB_address.** At write call the **FB(address)** function. See 0.
226	**File_Number**	At startup the register shall be set to 2. (*the first PCP file*) At read return the **File_Number** of the current **PCP_file.** At write call the **goto_PCP(File_Number)** function. See 0.
227	**Disc_Number**	At startup the register shall be set to 1. At read return the **Disc_Number** of the current disc. At write call the **goto_Disc(Disc_Number)** function. See 0.
228	**Play_item**	At startup the register shall be set to -1. At read return the **play event ID** value. At write call the **play(PIN)** function. See 0.
229	**MPEG_file_info**	At startup the register is set to –1. At read return current MPEG file number if the last write call was successful, else return -1. At write call the **MPEG_file_info(file_number)** function. See 0
230	**Playing_time**	At startup the register is set to 0. At read return the playing time (in seconds) of the current MPEG file updated by the **MPEG_file_info** function. At write do nothing.
231	**OGT_channels**	At startup set to 0. At read return the number of current available OGT channels of the current MPEG file updated by the **MPEG_file_info** function. See 0. At write do nothing.
232	**OGT_select**	At startup set to 0. At read return the current selected OGT channel. If OGT disabled then return 0. At write call the **OGT_select(channel)** function. See 0.
233	**Audio_streams**	At startup set to 0. At read return the number of current available Audio streams of the current MPEG file updated by the **MPEG_file_info** function. See 0. At write do nothing.
234	**Audio_select**	At startup set to 1. At read return the current selected Audio_stream number. If Audio disabled then return 0. At write call the **Audio_select(stream number)** function. See 0.
235	**Audio_mixing**	At startup the register set to playback device default value. (*stereo=3*) At read return the current Audio mixing setting At write call the **Audio_mixing**(data) function. See 0.
236	**OGT_image**	This is an optional extension. At startup the register shall be set to -1 if OGT rendering is supported by the playback device else it shall be set to -2. At read return the current File_Number of the Image file. If no image displayed then return -1. If this function is not supported then return -2. At write call the **OGT_rendering** function, see 0
237 .. 248	**reserved**	Reserved for future extensions.
249	**Trigger_mask**	This register is used to filter the Trigger_data in the MPEG stream to define if a trigger event shall occur. See 0. At startup this register is preset to –1 ($FFFF). At read return the current trigger mask. At write set the **Trigger_mask.** If the register is set to 0 then the Trigger function is disabled.
250	**Userkey_mask**	This register is used to enable and filter the User keys. Each bit represents one of the keys. If the bit is set then the key is enabled, else it's disabled. At startup this register is preset to –1 ($FFFF). At read return the current mask. At write set the **user key mask.** Bit#0 represents the user key with number = 0 and so on until bit #15 that represents the user key with number = 15. If the register is set to 0 then all user keys are disabled.
251	**Input_data**	This register returns the last pressed **User key** number or selected **Hotspot ID** number. For user key values see 0. and for Hotspot ID values see 0.

		At startup this register is set to -1. At read return the current **Input_data** (User key or Hotspot number). At write reset the **Input_data** to 0. If the value is in the range 128..143 then also highlight the corresponding hotspot area else no hotspot shall be highlighted.
252	**Timer_data**	At startup the register shall be set to 0. At read return the current **Timer** value (in units of 1/10 sec). At write call the **Timer(start_time) function**. See 0.
253	**NVRAM**	At startup the register shall be set to the **NVRAM_size** value in the **Disc_info** record. At read; If no Global registers could be restored form the NVRAM then return 0, else return the **NVRAM_size** value. At write no action.
254	**Error_handler_block**	At startup the register is set to -1. At read return current Error handler address. See 0 At write set the location of the PCP_block to execute by the PCP interpreter when an Error has been detected.
255	**Error_code**	At startup the register is set to 0. At read return current **Error_code**. See 0. At write the **Error_code** will be reset to 0.

Note: **Reserved** registers shall not be used. They are reserved for future use. Operations with reserved registers, shall generate an error.

PCP Player (API) functions

The Playback device shall support the following functions.

FB local variables memory(size) function

This function is called when the **FB var_size** register is set. It shall allocate the number of requested local FB variables in memory. The total memory is limited to 256 variables (512 bytes), so this function is to optimize the memory usage handling by the PCP interpreter. Up to 32 variables can be allocated for each FB function. The size value is reset to 0 when a new FB function is called, so the register must be set to 1..32 before local FB variables can be used, else an error will be detected.

FB(address) functions

These functions are called by setting the **FB_address** register with a value as defined below

Value	Function	Description
-1	**Return()**	Continue to interpret the Command following the Command that called the FB function or error handler.
0	**Reset()**	Clear the Local Variable stack, and the Return stack. Start interpret the first FB at FB_address = 0.
nnnn	**Function()**	If nnnn is a valid FB_address then call the **FB function** at address = value. Set the local FB variable size to 0.
else	**Error()**	Generate an Error

Goto_PCP(File_number) function

These function is called by setting the **File_number** register with a value as defined below

Value	Function	Description
2	**Restart()**	Load default FCP file with file_number = 2. Clear the Local Variable stack, and the Return stack. Start interpret the first FB at FB_address = 0.
nnnn	**Goto_PCP()**	If nnnn is a valid File_number for a PCP file, then load the FCP file with file_number = nnnn. Clear the Local Variable stack, and the Return stack. Start interpret the first FB at FB_address = 0.
else	**Error()**	Generate an Error

Goto_Disc(Disc_number) function

These function is called by setting the **Disc_number** register with a value as defined below

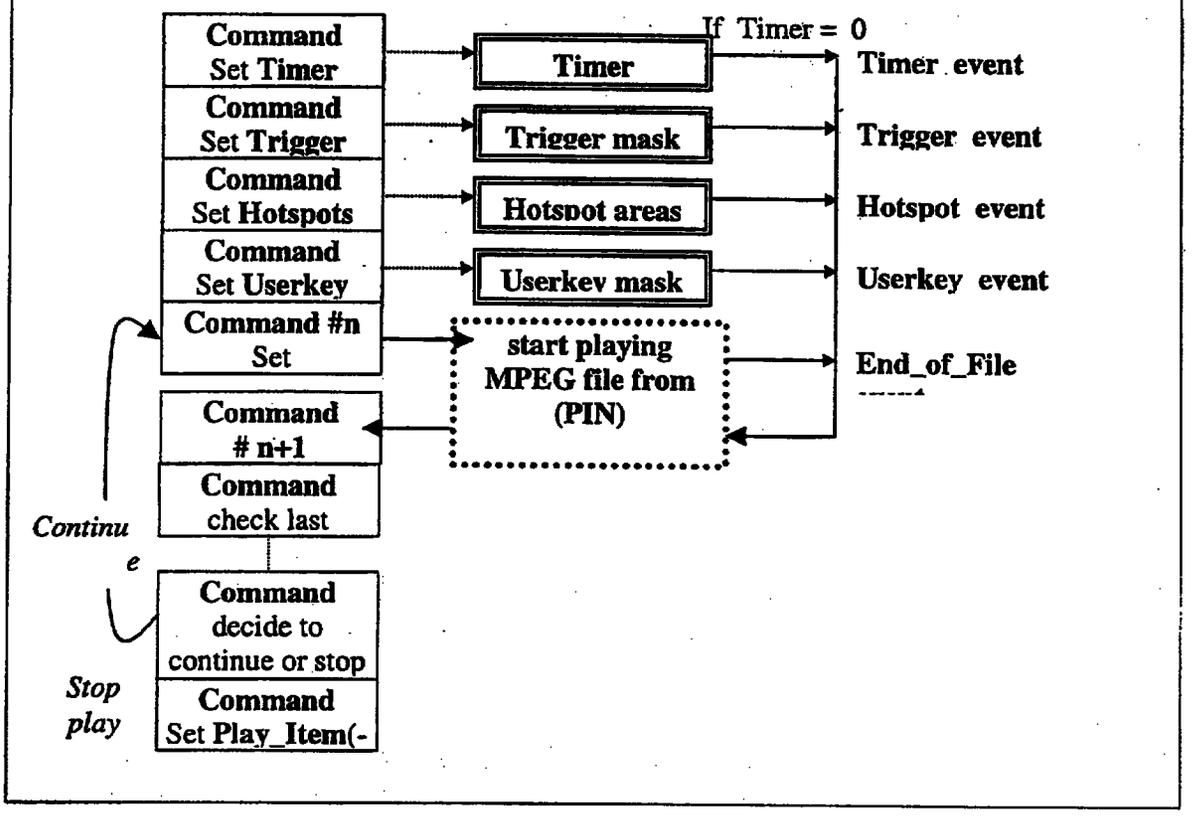
Value	Function	Description
0	**End()**	A function that ends the PCP playback (interpreter execution) and returns control to the playback device.
1 .. **Disc_count**	**Goto_Disc()**	Stop playing current disc. Change disc to a disc with this Disc_number. Start playing the disc. Note that the **Global variables** shall not change values during the disc change.
else	**Error**	Generate an Error

Play() function

This function is called by setting the **Play_Item** register with a value as defined below.

Value	Function	Description
-1	**Stop**	If the Player was **Playing** then the MPEG decoder shall be stopped and the last MPEG picture shall remain on the screen.
PIN	**Play**	If **PIN** value is valid and the player was **Stopped** then the playback device shall start playing the **Play_Item** indexed by the value in the **play_item** register. The PCP interpreter will wait to execute the Command until a **play_event()** has occurred. If **PIN** value is valid and the player was **Playing** then continue playing and halt the interpreter to wait for a **Play_event().**
else	**Error**	Generate an Error

Figure 1 Play() function program flow



Play_event()

A **Play_event** can only occur during the **Play()** function, and the corresponding **play_event_ID** value shall be returned in the **Play_item** register. The **play_events** are listed below:

Play_event ID code	Play_event	Description
-1	**End_of_file**	An event called from the MPEG player, to mark that the player has reached the end of the MPEG file.
-2	**Timer_terminate**	An event called from the player controller, to mark that the timer has reached the time = 0.
-3	**Trigger_point**	An event called from the MPEG decoder, to mark that a specific point in a MPEG_file has been passed during playback. The event shall be called if the logical result of (MPEG trigger_data & Trigger_mask) = True. *It is possible to use different kind of triggers by setting the msb bits as selectors and the lower bits as a trigger number.*
-4	**Userkey_input**	A function called from the player controller, to mark that a user key has been activated by the user. The event shall be called if the User key was enabled by the **userkey_mask**
-5	**Hotspot_select**	A function called from the player controller, to mark that a hotspot has been selected by the user. The event shall be called if the hotspot was enabled by valid area coordinates.

MPEG file info function

This function is called when the **MPEG_file_info** register is set to a **file_number** of a **File_directory_record** in the FRT file. The **Playing_time, OGT** and **Audio_stream** registers shall be set according to the **File_info** stored in the File_directory_record. This register shall return 0 if the **file_number** is not valid or the **File_ID** <> 10 (MPEG file), else it shall return –1.

OGT_select(channel) function

This function is called when the **OGT_select** register is set. Select the OGT_channel to the set channel. If set to 0 then disable display of OGT. If set to an invalid value then return 0 (disable).

Audio_select(stream number) function

This function is called when the **Audio_select** register is set. Select the Audio stream to the set number. If it is set to 0 then disable (mute) audio. If it is set to an invalid value then return 0 (disable).

Audio_mixing(data) function

This function is called when the **Audio_mixing** register is set. Set the Audio mixing as defined by the set data value as defined below:

Data value	Audio mixing setting description
0	Audio mute
1	Left channel only
2	Right channel only
3	Stereo
4	Surround sound
else	Do nothing

Hotspot() function

The hotspot function is used for user interaction on the screen, so a user can select a hotspot (by cursor or pointer control) to interact with the PCP. The **ID number** (128 .. 143) of a **selected** hotspot shall be stored in the **Input_data** register. The area within the indicated co-ordinates (**top, bottom, left, right**) shall be highlighted. Highlighting can be done e.g. by drawing a frame around the selected hotspot area, or by changing the contrast within the selected area. The playback device can chose frame thickness and color. Such highlighting is recommended to have two views, one when the pointer is inside the hotspot area or one when the hotspot is selected. If more hotspot areas contain the point selected, (in the case of overlapping selection areas) then the selection area with the lowest ID number shall be returned in the **Input_data** register. A hotspot is enabled when all area values are valid defined by (all values >=0 and within the screen area, and **bottom >= top**, and **right >= left**). If a hotspot is disabled then all hotspot area registers shall return -1. A hotspot area can be highlighted by writing a valid ID number for an enabled hotspot in the **Input_data** register. If the Input_data register is set to an invalid value then it shall return -1.

Userkey() functions

The Userkey() function is initiated by an user interaction pressing an enabled key. The key is enabled if the corresponding **Userkey_mask** bit is set to 1. The function returns the **Key number** value in the **Input_data** register as defined in the table below:

Key number	Userkey_mask bits (msb..lsb)	User Key description
0	%0000 0000 0000 0001	Numeric key 0
1	%0000 0000 0000 0010	Numeric key 1
2	%0000 0000 0000 0100	Numeric key 2
3	%0000 0000 0000 1000	Numeric key 3
4	%0000 0000 0001 0000	Numeric key 4
5	%0000 0000 0010 0000	Numeric key 5
6	%0000 0000 0100 0000	Numeric key 6
7	%0000 0000 1000 0000	Numeric key 7
8	%0000 0001 0000 0000	Numeric key 8
9	%0000 0010 0000 0000	Numeric key 9
10	%0000 0100 0000 0000	Up key ↑
11	%0000 1000 0000 0000	Down key ↓
12	%0001 0000 0000 0000	Left key ←
13	%0010 0000 0000 0000	Right key →
14	%0100 0000 0000 0000	Enter key ↵
15	%1000 0000 0000 0000	Trigger key •

Timer(time) function

This function is called when the **Timer_data** register is set. The timer is started when the **Timer_data** register is set to a value that is >0 (in units of 1/10 sec), and decrement the register value each 1/10 second until it reaches 0, when the timer is stopped. If the **Timer_data** register is set to 0, then the timer shall be stopped.

Error handling

Errors are handled by the **Error handler**, that can choose to skip the current command or to end the PCP playback. The location of the Error handler command block is stored in a dedicated register. The error handler returns to the next command (skip) by a **return** command, and ends the PCP playback by calling an **End()** function. Different types of errors that can occur during the PCP playback. The error code can be read from the **Error_code** register.

Error code	Error type	Description
0	**void**	No error
1	**Read error**	Disc damaged
2	**Access error**	Sector address not found
3	**File error**	Wrong PIN number
4	**Invalid File number**	An invalid **File_number** was set.
5	**Command code error**	Command code not valid
6	**Command index**	Command index out of range
7	**Execution error**	Divide by 0
8	**Invalid register**	A **reserved** register was addressed
9	**Invalid Disc number**	A **Disc_number** > **Disc_count** was set.
else	**reserved**	

OGT_rendering function

This function is an optional extension. If it is not supported then it shall return -2 in the **OGT_image** register.

This function is called when the **OGT_image** register is set to a **file_number** of a **File_directory_record** in the FRT file. The image data of the file shall be rendered into a selected Hotspot area. Parts of the image area outside the hotspot selection area shall be discarded (cropped).

Normally OGT can only be used when playing a movie file. This function can be used to overlay selectable OGT images on still pictures and menus. This is useful to interact with the user without loading a new picture and can also be selective on language preferences.

PCP Command definitions

All commands are 5 bytes. The first byte is an **opcode**, and the following 4 bytes shall be **operands.** The PCP interpreter shall execute the command as defined by the opcode and operands in the Table below. Most commands are conditional, depending on the value of one or two variables, meaning that the command shall only be executed if the condition is TRUE. A variable value or Register setting in the Variable Array can be written by a command, and reading a value is done by a conditional test. The registers are referenced by R[*index*]. Commands are referenced by Cmd[*index*], where the index is a value >0 relative to the beginning of the current Function Block **FB**.

Command codes

byte #1 opcode msb 7..6	5..3	lsb 2..0	byte #2	Byte #3	byte #4	byte #5	Opcode name	C code like description
00	cond1	xxx	i	j	dd	dd	calculate1 xxx	if cond1 { R[j]=R[j] opcodc(xxx) dddd
01	cond1	xxx	i	j	k	l	calculate2 xxx	if cond1 { R[j]=R[k] opcode(xxx) R[l] }
10	cond1	000	i	--	k	l	move	if cond1 { R[k]=R[l] }
10	cond2	001	i	j	k	l	move	if cond2 { R[k]=R[l] }
10	cond1	010	i	j	dd	dd	set const	if cond1 { R[j]=dddd }
10	cond1	011	i	j	dd	dd	set random	if cond1 { R[j]=random value; 0<=R[j]<=dddd }; if (dddd==$0000) {randomize/change seed}
10	000	100	i	j	dd	dd	fill	for (n=j; n<j+i; ++n) {R[n]=dddd }
11	cond1	000	i	--	idx	idx	jump	if cond1 {jump to Cmd[idx]}
11	cond2	001	i	j	idx	idx	jump	if cond2 {jump to Cmd[idx]}
11	cond1	100	i	--	idx	idx	loop1	if cond1 { --R[i]; jump to Cmd[idx] }
11	cond2	101	i	j	idx	idx	loop2	if cond2 { --R[i]; jump to Cmd[idx]}
11	cond1	110	i	j	idx	idx	loop3	if cond1 { --R[i]; --R[j]; jump to Cmd[idx]}

Note1: If an index or other value is out of range then an Exception_Error shall be executed.

Note2: bit 7 is msb and bit 0 is lsb.

Note3: if all bytes #1 to #5 are equal to 0 then execution skipped as a No Operation (NOP).

operand symbols

In the PCP opcode and operand definition table special symbols are used for different definitions as specified in the following table.

symbols	length	Meaning
i, j, k, l	1 byte	These letters are used to index variables (e.g. variable with index i is R[i]); If i identifies a is reserved variable then opcode instruction is not executed
dddd	2 bytes	a signed 16-bit constant. where byte#4 is the msb and byte#5 lsb
cond1 cond2	3 bits	Test for execution; if the test fails then jump to next Command else execute the opcode
idx	2 bytes	Used for jump to a command with index = idx
xxx	bits	See separate definition table for meaning
--	1 byte	Byte is not used (VOID)

opcode conditions

Reading Array variables is done by a conditional test. If a condition is not TRUE then the next command entry shall be executed. The **cond_1** test compares a variable R[i] compared to 0, and the **cond_2** test compares the relation between two variables R[i] and R[j] as defined below.

bit 5..3 msb..lsb	**cond_1 semantics**	**cond_2 semantics**
%000	True	R[i]==j
%001	R[i]>0	R[i]>R[j]
%010	R[i]<0	R[i]<R[j]
%011	R[i]==0	R[i]==R[j]
%101	R[i]>=0	R[i]>=R[j]
%110	R[i]<=0	R[i]<=R[j]

opcode function definitions

calculate_1 function -- If cond1 TRUE then variable R[j] indexed by byte #3 is set the outcome of calculation with the constant = dddd, defined by bytes #4 and #5, where byte #4 is MSB. The type of calculation is specified by the **opcode(xxx)** bits 2..0, as in table below.

calculate_2 function -- If cond2 TRUE then variable R[j] indexed by byte #3 is set the outcome of calculation of variable R[k] defined by byte #4 and R[l] defined by byte #5. The type of calculation is specified by the **opcode(xxx)** bits 2..0, as shown below.

xxx bits 2..0 msb ...lsb	Meaning	Note	example C-code	if R[k]=15 and R[l]=7 then
%000	**ADD**	addition	R[j]=R[k] + R[l]	R[j]= 22
%001	**SUB**	subtraction	R[j]=R[k] - R[l]	R[j]= 8
%010	**MUL**	multiplication	R[j]=R[k] * R[l]	R[j]= 105
%011	**DIV**	integer division	R[j]=R[k] / R[l]	R[j]= 2
%100	**MOD**	modulo division	R[j]=R[k] % R[l]	R[j]= 1
%101	**AND**	logical AND	R[j]=R[k] & R[l]	R[j]= 7
%110	**OR**	logical OR	R[j]=R[k] | R[l]	R[j]= 15
%111	**XOR**	logical XOR	R[j]=R[k] ^ R[l]	R[j]= 8

Note There is no carry flag for variable wrap around indication.

move function -- The variable R[k] indexed by byte #4 is set equal to the variable R[k] indexed by byte #5.

set function -- The variable R[j] indexed by byte #3 is set to value dddd defined by byte #4 and #5.

random function -- The variable R[j] indexed by byte #3 is set to a random value between 0 (zero) and dddd defined by byte #4 and #5. If dddd=0 then randomize the random generator.

fill function -- Fill N variables starting from R[j] with dddd defined by byte #4 and #5, where N is equal the value of R[i] indexed by byte #2

jump function -- Jump immediate to a command with index = **idx** defined by byte #4 and #5.

loop_1 function -- Decrement R[i], defined by byte #2, and cond#1 true then jump to the command with index = **idx** defined by byte #4 and #5.

loop_2 function -- Decrement R[i], defined by byte #2, and if cond#2 true then jump to the command with index = **idx** defined by byte #4 and #5.

loop_3 function -- Decrement R[i], defined by byte #2 and decrement R[j], defined by byte #3, and cond#1 true then jump to the command with index = **idx** defined by byte #4 and #5.

Programming examples

This section is informative.

CLAIMS:

1. Record carrier (1) comprising information organized in a plurality of files containing audio visual data (11) and playback control data (12) for controlling playback of the audiovisual data on a playback device while enabling user interaction, the information further comprising priority information (priority), indicating the relative priority with which the files are to be stored in a cache memory (39) of the playback device.

2. Record carrier according to claim 1, characterized in that the priority information (priority) is contained in a single file (FRT).

3. Record carrier according to claim 1, characterised in that the priority information (priority) is related to an estimated frequency with which the said data is accessed.

4. Record carrier according to claim 1, characterised in that the control data is comprised in a control program (PCP) which comprises references to menus, and that the data representing the control program and the menus has a relatively high priority.

5. Playback device for playback of a record carrier (1) comprising information organized in a plurality of files containing audiovisual data (11) and playback control data (12) for controlling playback of the audiovisual data while enabling user interaction, the playback device comprising read means (20) for reading the data from a record carrier, a cache memory (39) for storing data from the record carrier, user input means (40,41) for receiving user input, control means (30) for processing the control data, and reproduction means (51-54) for reproducing the audio visual data, the device being adapted to read priority information (priority) and to store files into the cache memory with a priority determined by this priority information.

6. Playback device according to claim 5, characterized by means for rewriting the priority information at the record carrier in accordance with the frequency with which the files are actually accessed by a user.

7. Method for playback of a record carrier (1) comprising information organized in a plurality of files containing audiovisual data and playback control data for controlling playback of the audiovisual data while enabling user interaction, which method comprises the steps of

a. determining whether a cache memory (39) does have sufficient unoccupied space (S5),

aa. if it is determined that the cache memory (39) does not have sufficient unoccupied space, then determining whether the cache memory does have space sufficiently large which is occupied by one or more further data files having a priority value lower than the priority value of the first data file (S6),

aaa. if the outcome of step aa is true, then overwriting said one or more further data files by the first data file (S7),

8. Method for playback according to claim 7, characterised in that if the outcome of step a is true then

ab determining whether the first data file has a priority value higher than a predetermined value,

aba if the outcome of step ab is true then loading the first data unit from the record carrier into the cache memory,

9. Method for playback of a record carrier comprising information organized in a plurality of files containing audiovisual data and playback control data for controlling playback of the audiovisual data while enabling user interaction, which method comprises a procedure for loading files in a cache memory comprising the steps of

d setting a reference priority value (PR),(step S10),

e for a plurality of files examining whether a priority value assigned thereto is higher than the reference priority value (step S12)

ea if the outcome of step e is true examining whether the cache memory comprises sufficient space for storing the said data unit (step S13),

eaa if the outcome of step ea is true then loading said data unit into said space (step S14),

f reducing the reference priority value (S17),

g determining whether the reference priority value is greater than or equal to a bottom priority value (S18),

ga repeating steps d to g if the outcome of step g is true.

10. Device for recording a record carrier (1) comprising information organized in a plurality of files containing audio visual data (11) and playback control data (12) for controlling playback of the audiovisual data on a playback device while enabling user interaction, the information further comprising priority information (priority), indicating the relative priority with which the files are to be stored in a cache memory (39) of the playback device,

the playback device comprising

means for obtaining the audiovisual data,

means for composing the control data,

priority determining means for determining the priority information,

formatting means for formatting the audiovisual data and the control data into files,

recording means for recording the information comprising the priority information, the audiovisual data and the control data at the record carrier.

11. Method for recording a record carrier (1) comprising information organized in a plurality of files containing audio visual data (11) and playback control data (12) for controlling playback of the audiovisual data on a playback device while enabling user interaction, the information further comprising priority information (priority), indicating the relative priority with which the files are to be stored in a cache memory (39) of the playback device,

the method comprising the steps of

- obtaining the audiovisual data,
- composing the control data,
- determining the priority information,
- formatting the audiovisual data and the control data into files,
- storing the information comprising the priority information, the audiovisual data and the control data at the record carrier.