Confidential Information

Super Audio CD System Description

Part 2 Audio Specification

Version 2.0

March 2004

SONY PHILIPS

COPYRIGHT

DISCLAIMER

The information contained herein is believed to be accurate as of the date of publication, however, neither Royal Philips Electronics, nor Sony Corporation will be liable for any damages, including indirect or consequential, from use of the System Description Super Audio CD or reliance on the accuracy of this document.

LICENSING

Application of the System Description Super Audio CD in both disc and equipment products requires a separate license from Philips.

CLASSIFICATION

The information contained in this document is marked as confidential and shall be treated as confidential according to the provisions of the Agreement through which the document has been obtained.

NOTICE

For any further explanation of the contents of this document, or in case of any perceived inconsistency or ambiguity of interpretation, or for any information regarding the Super Audio CD patent license program, please consult:

Philips Intellectual Property & Standards
Business Support
Building WAH
Prof. Holstlaan 6
5656 AA Eindhoven
The Netherlands

Fax.: +31 40 27 32113
Internet: www.licensing.philips.com
Email: info.licensing@philips.com

Table of Contents Page

Table of Figures Page

This page is intentionally left blank.

1. General

1.1 Scope

This document describes the Audio part of the Super Audio CD system. The Super Audio CD System Description consists of three parts:

- Part 1, Physical Specification
- Part 2, Audio Specification
- Part 3, Copy Protection Specification

1.2 Main Features

- Hybrid disc concept which makes Super Audio CD discs possible that can be played both on existing CD Audio players and new Super Audio CD players.
- Independent super high quality 2-channel stereo and multichannel audio on one disc. It is recommended that a multichannel disc contains the same content for 2-channel stereo and for multichannel audio. Every Super Audio CD disc must have a 2-Channel Stereo Area with Audio Tracks. It is recommended that a hybrid disc contains the same content for the CD Layer and for 2-channel stereo on the high density layer.
- Playing time of 70 to 80 minutes for both 2-channel stereo and multichannel audio on one disc.
- Multichannel audio with maximum six, full quality, audio channels.
- Multichannel loudspeaker arrangements according to ITU recommendations.
- The main access mechanism for a Super Audio CD is the Table Of Contents structure. Optionally a file system can be used for access.
- The Table Of Contents contains information on four levels, album information, disc information, area information and track information.
- In the Table Of Contents eight Text Channels are available. Both normal and phonetic text can be used.
- Support for text and pictures in supplementary data, multiplexed with the audio signal.
- Separate Extra Data area, for optional data related to the audio. The information in this Extra Data area can be retrieved via a file system.
- Advanced copy protection system, specified in Super Audio CD System Description Part 3.
- Bonus Tracks.
- JPEG pictures, sub-pictures and coded text for slide shows and for browsable information.
- Super Audio CD players developed according to this version of the specification do not need to read the specification version number (see chapter 3.1.1.1.2) to be able to play back discs according to specification version numbers 1.0, 1.1, 1.2 and 1.3.
- General audio parameters:

A/D conversion process	1-bit Delta-Sigma (DSD) Optional DST Coding
Sampling Frequency	64 times 44.1 kHz

1.3 References and Conformance

Super Audio CD conforms to all mandatory requirements specified in this document. All parts in this document are mandatory unless they are specially defined as recommended or optional or informative.
Super Audio CD also conforms to the applicable parts of the System Descriptions or international standards that are listed below:

BIG5	Big5 character set.
CCIR 601	ITU-R Recommendation BT.601-5, Studio Encoding Parameters of Digital television for Standard 4:3 and Wide-Screen 16:9 Aspect Ratios (1982 - 1995), [formerly CCIR Rec.601] (Geneva: ITU)
CCIR 709	ITU-R Recommendation BT.709, Basic Parameter Values for the HDTV standard for the Studio and for International Programme Exchange (1990), [formerly CCIR Rec.709] (Geneva: ITU, 1990)
CD-DA	Compact Disc Digital Audio, specified in the System Description Compact Disc Digital Audio (“Red Book”), Royal Philips Electronics and Sony Corporation.
EXIF	JEIDA-49-1997, Digital Still Camera Image File Format Standard (Exchangeable image file format for Digital Still Camera: EXIF) Version 2.0, JEIDA, Nov. 1997.
IEC 908 / IEC 60908	Compact disc digital audio system.
IEEE 1394	Standard for a High Performance Serial Bus. Ref. No. IEEE 1394-1995. Also see 1394 Trade Association, www.1394ta.org.
ISO 639	Code for the representation of names of languages. Ref. No. ISO 639.
ISO/IEC 646	Information technology - ISO 7-bit coded character set for information interchange. Ref. No. ISO/IEC 646.
ISO/IEC 2022	Information technology - Character code structure and extension techniques. Ref. No. ISO/IEC 2022.
ISO 3901	Documentation - International Standard Recording Code (ISRC). Ref. No. ISO 3901.
ISO/IEC 8859-1	Information technology - 8-bit single-byte coded graphic character sets -- Part 1: Latin alphabet No. 1. Ref. No. ISO/IEC 8859-1.
ISO 9660	Information processing - volume and file structure of CD-ROM for information interchange. Ref. No. ISO 9660.
ISO/IEC 10918-1	Information technology -- Digital compression and coding of continuous-tone still images: Requirements and guidelines. Ref. No. ISO/IEC 10918-1.
ITU-R BS.775	Multi-channel stereophonic sound system with and without accompanying picture. ITU-R Recommendations BS Series. ITU-R BS.775-1.
JPEG	ISO/IEC 10918-1

RIS 504	Online data exchange format for records. Published by Recording Industry Association of Japan.
RIS 506	Character set for records. Published by Recording Industry Association of Japan.
Super Audio CD Part 1	Scarlet Book Part 1. Super Audio CD System Description Part 1, Physical Specification, Royal Philips Electronics and Sony Corporation.
Super Audio CD Part 3-General	Scarlet Book Part 3-General. Super Audio CD System Description Part 3, Copy Protection Specification: General Introduction and Specification, Royal Philips Electronics and Sony Corporation.
Super Audio CD Part 3-Player	Scarlet Book Part 3-Player, Super Audio CD System Description, Copy Protection Specification: Player Specification, Royal Philips Electronics and Sony Corporation.
UDF	Universal Disk Format™ Specification. Published by OSTA, Optical Storage Technology Association, see www.osta.org.

1.4 Definitions

The following definitions are used in this specification.

Access List	The Access List is a table providing the relation between Time Code and Sector Address for a DST coded Audio Area.
Album	An Album consists of one or more discs. All discs in an Album must have the same Album_Catalog_Number (see chapter 3.1.1.2.3).
Audio Channel	The stream of DSD bits intended for one loudspeaker.
Audio Frame	A Frame containing Audio data.
Audio Channel Number	The sequence number assigned to an Audio Channel. Audio Channel Numbers are contiguously numbered starting with one.
CCI	Copy Control Information
Constant Byte Rate	The number of Sectors needed to store three Audio Frames is fixed.
Copy Control Information	Information included on Super Audio CD discs that pertains to the number of copies and/or quality of such copies that may be made from permitted digital outputs from Super Audio CD players and from connections from and within Integrated Super Audio CD Player/Recorders and that may include other fields.
Display Track	Display Track[n+Track_Offset] consists of Pause[n] and Track[n] and is used for display purposes.
Display Track Number	Display Track Number = Track Number + Track_Offset. Track_Offset is given in the Area TOC. The maximum value for Display Track Number is 510.
DSD	Direct Stream Digital, the one bit audio signal used on this specification. A DSD signal can either be DST Coded DSD, or Plain DSD.

DTCP	Digital Transmission Content Protection Specification Version 1.2, (Information Version), Volume 1. See www.dtcp.com.
EKB	Enabling Key Block. A data structure that encodes the revocation state of the Super Audio CD copy protection system, i.e. the Device Keys that have been withdrawn by the Key Issuance Centre.
EOS, eos	End Of Sector.
Elementary Frame	An Audio Frame or a Supplementary Data Frame, or a Padding Frame.
Frame	A block of data belonging to a certain Time Code. The data can be either Audio Data, Supplementary Data, Padding or it can be Multiplexed. The playing time of a Frame is 1/75 Sec.
Hybrid Disc	A Super Audio CD disc with a high density layer and a CD-DA layer.
Index	A subdivision of a Track.
Index Number	An Index Number is the sequence number of an Index within a Track. The start position of Index one is equal to the start position of the Track. The maximum number of Indexes within a Track is 255.
LSN	Logical Sector Number, used to address the Sectors on the disc.
Multiplexed Frame	The part of the multiplexed stream belonging to a certain Time Code. See Frame.
Padding Frame	A Frame containing Padding data.
Pause	Area before the start of a Track. A Pause length of zero is allowed.
Plain DSD	The Direct Stream Digital one bit audio signal that has not been DST coded.
Reserved	All fields labeled Reserved are reserved for future standardization. All Reserved fields must be set to zero.
Sector	The 2048 bytes of Main Data in a Data Frame, see Super Audio CD Part 1 chapter 4.2.2 (≠ Physical Sector as defined in Super Audio CD Part 1 chapter 4.2.7).
Silence Pattern	A digitally generated DSD pattern with the following properties: • All Audio Bytes (see 5.6.1) have the same value • Each Audio Byte must contain 4 bits equal to zero and 4 bits equal to one.
Super Audio CD	The disc as defined in this specification.
Supplementary Data	Additional data multiplexed with the audio signal.
Supplementary Data Frame	A Frame containing Supplementary Data.
Text Channel	Text in the TOC using one language / character set combination. One Text Channel can only use one character set.

Text Channel Number	The sequence number assigned to a Text Channel. Text Channel Numbers are contiguously numbered starting with one.
TNO, tno	Track Number.
TOC	Table Of Contents. Album and disc related information is stored in the Master TOC. Area and track related information is stored in the Area TOC.
Track	A Track is a contiguous area on the disc with audio information and with one and the same Track number.
Track Number	A Track Number is the sequence number of a Track. The first Track Number in an Audio Area is one. The maximum number of Tracks in an Audio Area is 255.
Variable Byte Rate	The number of Sectors needed to store Audio Frames is variable.

1.5 Definitions for DST

The following definitions are used in the DST part of this specification.

Direct Stream Transfer	The lossless coding technique used for DSD signals in Super Audio CD.
DST	See Direct Stream Transfer.
Half Probability	Half Probability defines for each Audio Channel in an Audio Frame whether the first DSD bits are arithmetically encoded using the Ptable values, or using a probability equal to ½.
Mapping	Mapping defines, for each Segment, the Prediction Filter and Probability Table that is used.
Prediction Filter	A Prediction Filter is a transversal filter used to predict the value of the next DSD bit. A Prediction Filter is characterized by a prediction order and by coefficients.
Probability Table	A probability Table contains the probability that the value of a DSD bit is predicted erroneously for a given output of the prediction filter.
Ptable	Probability Table
Segmentation	Each Audio Channel in an Audio Frame is partitioned into Segments.

1.6 Conventions

In this version of the Super Audio CD Specification the conventions as described in this chapter are used.

1.6.1 Arithmetic and bit operations

a>>b	Right shift a over b bits. The new msb bits are set to '0'.
a<<b	Left shift a over b bits. The new lsb bits are set to '0'.
a | b	Bitwise OR of a and b.
a & b	Bitwise AND of a and b.
min(a,b)	Minimum value of a and b.
max(a,b)	Maximum value of a and b.
a mod b	Value of a modulo b.
trunc(a)	Value of a, rounded downwards.
roundup(a)	Value of a, rounded upwards.
| a |	Absolute value of a.
a==b	Evaluate if a is equal to b.
a!=b	Evaluate if a is not equal to b.
a=b	Variable a is set to the value of b.
a++	a = a + 1
a -= b	a = a – b
a += b	a = a + b
a^b	Bitwise Exclusive OR of a and b.

1.6.2 Bit ordering

The graphical representation of all multiple-bit quantities is such that the most significant bit (msb) is on the left, and the least significant bit (lsb) is on the right. Figure 1-1 defines the bit position in a Byte.

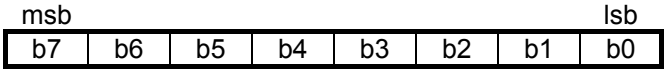

Figure 1-1 : Bit ordering in a Byte

1.6.3 Bit sequence

In all places where a bit sequence is used, a most significant bit first notation is used.

1.6.4 Byte position

The bytes in a series of n Sectors are successively numbered 0 .. n*2048-1. Figure 1-2 defines the Byte position (BP) in a series of n Sectors.

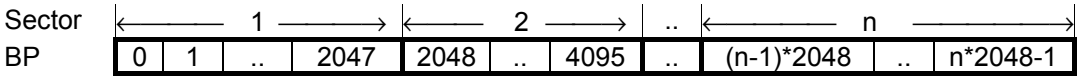

Figure 1-2 : Byte position in a series of n Sectors

1.6.5 Decimal notation

All Decimal values are preceded by a blank space or the range indicator (..) when included in a range. The most significant digit is on the left, the least significant digit is on the right.

1.6.6 DSD bit order

The first sampled DSD bit is stored in the most significant bit of a byte. See chapter 5.6.1.

1.6.7 DSD Polarity

A DSD bit equal to one means "plus". A DSD bit equal to zero means "minus".

1.6.8 Hex notation

All Hexadecimal values are preceded by a $. The most significant nibble is on the left, the least significant nibble is on the right.

1.6.9 Range

Constant_1..Constant_2 denotes the range from and including Constant_1 up to and including Constant_2, in increments of 1.

1.6.10 Until

Until is used in figures to indicate that for a structure Byte Positions are used upto (not including) a given value.
At Byte Position B1, the expression “until B2” specifies B2-B1 bytes. At Byte Position B1, the expression “until eos” specifies the number of bytes from B1 upto and including the last byte of the current Sector. Note that Byte Position is specified relative to the start of the current, or a previous, Sector.

1.7 Basic Types

In this version of the Super Audio CD Specification the Basic Types as described in this chapter are used.

1.7.1 Simple types

1.7.1.1 BsMsbf

Bit Sequence, Most Significant Bit First, must be interpreted as a Bit String.

1.7.1.2 Char

A one-byte character, encoded according to ISO 646. The NUL ($00) character is not allowed for Char.

1.7.1.3 SiMsbf

Bit sequence, Most Significant Bit First, must be interpreted as Signed Integer using two’s complement notation.

1.7.1.4 UiMsbf

Bit sequence, Most Significant Bit First, must be interpreted as Unsigned Integer.

1.7.1.5 Uint*n*

An n bit, binary encoded, unsigned numerical value.

1.7.1.6 Uint8

An 8 bit, binary encoded, unsigned numerical value. A Uint8 value must be recorded in a one-byte field.

1.7.1.7 Uint16

A 16-bit, binary encoded, unsigned numerical value. A Uint16 value, represented by the hexadecimal representation $wxyz, must be recorded in a two-byte field as $wx $yz (most significant byte first).

1.7.1.8 Uint32

A 32-bit, binary encoded, unsigned numerical value. A Uint32 value, represented by the hexadecimal representation $stuvwxyz, must be recorded in a four-byte field as $st $uv $wx $yz (most significant byte first).

1.7.2 Complex types

1.7.2.1 Date

The syntax of Date is defined in Figure 1-3.

	# bytes	format	value
Date(){			
Year	2	Uint16	
Month	1	Uint8	0..12
Day	1	Uint8	0..31
}			

Figure 1-3 : Syntax of Date

The value of the Year field has a range of 0..65535. The value of the Month field has a range of 0..12, with 1..12 meaning January .. December. The value of the Day field has a range of 0..31. The value zero for Year, Month and Day is only allowed if a valid Date is not available.

1.7.2.2 Genre_Code

The syntax of Genre_Code is defined in Figure 1-4.

	# bytes	format	value
Genre_Code(){			
Genre_Table	1	Genre_Table	
Reserved	1	Uint8	0
Genre_Index	2	Uint16	
}			

Figure 1-4 : Syntax of Genre_Code

If a Genre_Code is not used, Genre_Table and Genre_Index must be set to zero.

1.7.2.2.1 Genre_Table

Genre_Table identifies the table used by Genre_Index. The definition of the values for Genre_Table is given in Figure 1-5.

Value	Table format
0	Not used.
1	General Genre Table. See Annex B.
2	Japanese Genre Table. See RIS504.
3..255	Reserved for future standardization.

Figure 1-5 : Definition of Genre_Table

1.7.2.2.2 Genre_Index

Genre_Index contains the index to the selected Genre_Table that will give the actual genre definition. If Genre_Table is equal to zero, then Genre_Index must be set to zero.

1.7.2.3 Genre4

The syntax of Genre4 is defined in Figure 1-6.

```
                                    # bytes        format
Genre4(){
   for (i=1; i<=4; i++){
       Genre_Code[i]                  4           Genre_Code
   }
}
```

Figure 1-6 : Syntax of Genre4

If used, Genre_Code fields are available, they must start at i=1. The unused Genre_Code fields must follow the used Genre_Code fields.

1.7.2.4 Spec_Version

Spec_Version is a 2 byte number showing the version number of the format, used in this disc. The syntax of Spec_Version is defined in Figure 1-7.

```
                                    # bytes        format              value
Spec_Version() {
   Major_Version                      1           Uint8                 2
   Minor_Version                      1           Uint8                 0
}
```

Figure 1-7 : Syntax of Spec_Version

Major_Version must contain the major version number of this specification. Minor_Version must contain the minor version number of this specification. For discs according to this specification, Major_Version is equal to 2 and Minor_Version is equal to 0.

1.7.2.5 Special_Char

Special_Char is a text character with a length of one or two bytes, depending on the character set used for Special_Char.

1.7.2.6 Special_String

Special_String is a sequence of Special_Char one or two byte characters, terminated by a zero Special_Char. The syntax of Special_String is defined in Figure 1-8.

```
                                    # bytes        format              value
Special_String(){
   for (i=1; i<=n; i++){
       Character[i]                   1..2        Special_Char
   }
   Zero_Special_Char                  1..2        Special_Char          0
}
```

Figure 1-8 : Syntax of Special_String

All Characters in a Special_String must belong to one and the same character set. If the Characters are from a one byte character set, the length of Zero_Special_Char is one byte. If the Characters are from a two byte character set, the length of Zero_Special_Char is two bytes. A zero length Special_String is allowed. A zero length Special_String only contains a Zero_Special_Char.

1.7.2.7 String

A string of characters encoded according to ISO 646 IRV. Control Characters are not allowed. The syntax of String is defined in Figure 1-9.

```
                                    # bytes         format
String(){
   for (i=1; i<=n; i++){
       Character[i]                   1            Char
   }
}
```

Figure 1-9 : Syntax of String

1.7.2.8 Text_Channels

The syntax of Text_Channels is defined in Figure 1-10.

```
                                    # bytes         format                  value
Text_Channels(){
   N_Text_Channels                     1            Uint8                    0..8
   Reserved                            7            Uint8                     0
   for (c=1; c<=N_Text_Channels; c++)
   {
       Language_Code[c]                2            String
       Character_Set_Code[c]           1            Character_Set_Code       1..7
       Reserved                        1            Uint8                     0
   }
   Reserved                 32-4*N_Text_Channels    Uint8                     0
}
```

Figure 1-10 : Syntax of Text_Channels

The used Text Channels must be encoded starting with Text Channel Number c equal to one. All unused Text Channels must have a Text Channel Number c, higher than the Text Channel Number of all used Text Channels. It is recommended that an Super Audio CD player uses Text Channel one as the default Text Channel.
If the same data combination is used in the Text Channels of both the Master TOC and the Area TOC, it is recommended to use the same Text Channel Number order for the Master TOC and for the Area TOC.
If N_Text_Channels is not equal to zero, at least one text item must be present in each used Text Channel.

1.7.2.8.1 N_Text_Channels

N_Text_Channels contains the number of Text Channels used. The maximum allowed value for N_Text_Channels is 8. A value of zero is allowed for N_Text_Channels. N_Text_Channels must be equal to the number of used Text Channels.

1.7.2.8.2 Language_Code

Language_Code[c] must contain the ISO 639 Language Code that is used with Text Channel c. All text in Text Channel c must be according to this Language Code. The value $0000 is not allowed for Language_Code[c].

1.7.2.8.3 Character_Set_Code

Character_Set_Code[c] defines the character set used for Text Channel c. The definition of Character_Set_Code is given in Figure 1-11. All text in Text Channel c must be encoded using this Character Set. The value zero is not allowed for Character_Set_Code[c].

Code	Bytes per character	Description
0	--	Not used
1	1	ISO 646 International Reference Version (IRV), escape sequences are not allowed
2	1	ISO 8859-1, escape sequences are not allowed
3	2	RIS 506
4	2	Korean KSC 5601-1989[149]
5	2	Chinese GB 2312-80[58]
6	2	Big5
7	1	ISO 8859-1, escape sequences to single byte character sets are allowed
8..255	--	Reserved for future standardization

Note: Numbers in between square brackets [ ] are the ISO 2022 registration numbers

Figure 1-11 : Definition of Character Set

For the usage of escape sequences see ISO 2022. Note that one Text Channel can only use one Character_Set_Code. Note that in a 2 byte character set each character has a length of 2 bytes.

1.7.2.8.3.1 Control Characters

With ISO 646 and ISO 8859-1 it is not allowed to use the DEL character ($7F) and the C0 set of Control Characters, with the following exceptions:

- The ESC ($1B) character with ISO 8859-1 and Character_Set_Code equal to seven.
- The LF ($0A) character with ISO 646 and ISO 8859-1 when used in Supplementary Data or in the Extension Area. The LF character has the functionality to position next character on a new line. In the Master TOC and in the Area TOC the LF character is not allowed.

1.7.2.9 Time_Code

The syntax of Time_Code is defined in Figure 1-12.

```
                                    # bytes        format                value
Time_Code(){
   Minutes                            1           Uint8                 0..255
   Seconds                            1           Uint8                  0..59
   Frames                             1           Uint8                  0..74
}
```

Figure 1-12 : Syntax of Time_Code

The value of the Minutes field has a range of 0..255. The value of the Seconds field has a range of 0..59. The value of the Frames field has a range of 0..74. If both the Minutes, Seconds and Frame fields are set to $FF, a valid Time_Code is not available.

1.7.2.10 TOC_Text

The length of a TOC_Text structure must be a multiple of 4 bytes. The syntax of TOC_Text is defined in Figure 1-13.

```
                                    # bytes        format                  value
TOC_Text(){
   Text_Type                          1           Text_Type
   Padding1                           1           Char                     $20
   Sp_String                                      Special_String
   Padding2                         0..3          Uint8                     0
}
```

Figure 1-13 : Syntax of TOC_Text

1.7.2.10.1 Text_Type

Text_Type gives the type of text encoded in this TOC_Text. The definition of Text_Type is given in Figure 1-14.

Value	Meaning
0	Reserved
1	Title
2	Name(s) of the performer(s)
3	Name(s) of the songwriter(s)
4	Name(s) of the composer(s)
5	Name(s) of the arranger(s)
6	Message(s) from the content provider and/or the artist
7	Extra message(s)
8	Copyright
9..127	Reserved
128	Reserved
129	Title, phonetic text
130	Name(s) of the performer(s), phonetic text
131	Name(s) of the songwriter(s), phonetic text
132	Name(s) of the composer(s), phonetic text
133	Name(s) of the arranger(s), phonetic text
134	Message(s) from the content provider and/or the artist, phonetic text
135	Extra message(s), phonetic text
136	Copyright, phonetic text
137..255	Reserved

Figure 1-14 : Definition of Text_Type

1.7.2.10.2 Padding1

The Padding1 field must contain one space character ($ 20).

1.7.2.10.3 Sp_String

Sp_String contains the text information for the TOC_Text structure. The definition of Special_String is given in chapter 1.7.2.6.

1.7.2.10.4 Padding2

The Padding2 field must be used to adjust the length of the TOC_Text structure to a multiple of four bytes. All bytes in the Padding2 field must be set to zero.

1.7.2.11 Weblink_String

Weblink_String is a sequence of Char followed by a zero byte. A zero length Weblink_String is allowed. A zero length Weblink_String contains one Zero_Byte. The maximum length of Weblink_String is 128 bytes. The definition of Weblink_String is given in Figure 1-15.

	# bytes	format	value
Weblink_String(){			
for (i=1; i<=n; i++){			
Character[i]	1	Char	
}			
Zero_Byte	1	Uint8	0
}			

Figure 1-15 : Syntax of Weblink_String

Weblink_String must contain a full URL, an example of a valid Weblink_String is “http://www.weblink.net”,0.

2. Disc Layout

2.1 Volume Space

The High Density layer of a Super Audio CD disc is used as one single logical volume. The definition of the physical format is defined in Super Audio CD Part 1, Physical Specification.

The length of a Logical Sector must be 2048 bytes, which is equal to the length of a Physical Sector. Each Logical Sector of a volume is identified by a unique Logical Sector Number. Logical Sector Numbers must be consecutive integers assigned in ascending order to the Physical Sectors on the disc. For the High Density Layer of a Hybrid or Single Layer Super Audio CD disc, the relation between Physical Sector Number and Logical Sector Number is defined in Figure 2-1. For a Dual Layer Super Audio CD disc, the relation between Physical Sector Number and Logical Sector Number is defined in Figure 2-2.

Logical Sector Number 0 must be assigned to Sector Start PSN of Physical Layer 0. The value of Start PSN is equal to the value given by Bytes 5 to 7 of Application Data Zone allocation, see Super Audio CD Part 1 chapter 4.3.4.5.1.

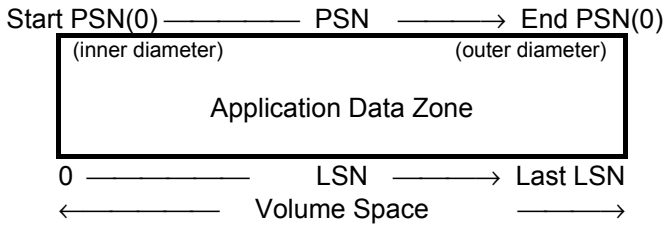

Figure 2-1 : Volume Space of a disc with one High Density Layer

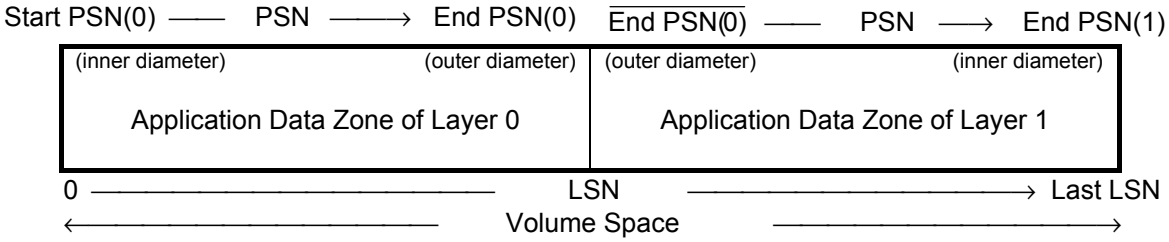

Figure 2-2 : Volume Space of a Dual Layer disc

Start PSN(0)	The first Physical Sector number of the Application Data Zone of Layer 0.
End PSN(i)	The end Physical Sector number of the Application Data Zone of Layer i (i=0, 1).
$\overline{\text{End PSN(0)}}$	Is calculated such that each bit of End PSN(0) is inverted (ones are replaced with zeros and vice versa).
Last LSN	The last Logical Sector Number of the Logical Volume.

2.2 Structure of the Volume Space

The Volume Space of a disc according to this version of the Super Audio CD Specification is split into: File System Area, DTCP Area, EKB1 Area, Master TOC Area, Rev TOC Area, 2-Channel Stereo Area, Multi Channel Area, Extension Area, EKB2 Area, Revocation Data Area and Extra Data Area. In discs according to the Super Audio CD Specification Version 1.3 or lower, the EKB1 Area, the Rev TOC Area, the Extension Area, the EKB2 Area and the Revocation Data Area do not exist. The structure of the Volume Space is shown in Figure 2-3.

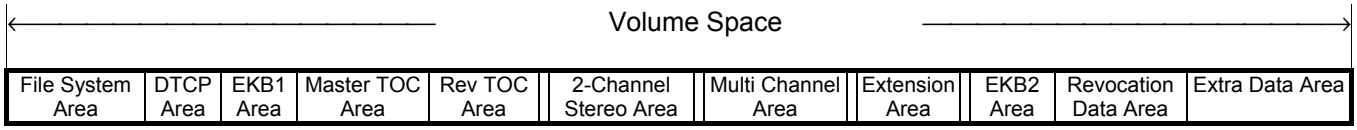

Figure 2-3 : Structure of the Volume Space

The 2-Channel Stereo Area and the Multi Channel Area are called Audio Areas. Every Super Audio CD disc according to this version of the Super Audio CD Specification must contain the Master TOC Area, the DTCP Area, the EKB1 Area, the Rev TOC Area, the EKB2 Area, the Revocation Data Area and minimally one Audio Area. The 2-Channel Stereo Area must be present on discs according to the Super Audio CD Specification Version 2.0 or higher. Note that the 2-Channel Stereo Area is optionally present on discs according to the Super Audio CD Specification Version 1.3 or lower. The presence of the Extension Area and the Extra Data Area is optional.
Gaps between the 2-Channel Stereo Area, the Multi Channel Area, the Extension Area and the EKB2 Area are allowed. There is a gap of three Sectors between the Rev TOC Area and the 2-Channel Stereo Area. The content of all Sectors in the gaps is reserved for future use and must be set to zero.

2.2.1 File System Area

A Super Audio CD disc can optionally contain the ISO 9660 and/or the UDF file systems. The File System Area starts at LSN 0 and ends at LSN 399. It is recommended that all file system structures are stored in the File System Area. When needed file system structures also can be stored in the Extra Data Area. The content of all not used Sectors in the File System Area must be set to zero. If no file system is used, the content of all Sectors in the File System Area must be set to zero.

2.2.2 DTCP Area

The DTCP Area contains DTCP information, see chapter 3.3. The DTCP area starts at LSN 400 and has a length of 48 Sectors.

2.2.3 EKB1 Area, EKB2 Area

The EKB1 Area and the EKB2 Area each contain one copy of EKB_Data. EKB_Data is defined in chapter 3.4. The start address of the EKB1 Area is given in the Master_TOC (see chapter 3.1.1.6.11). The start address of the EKB2 Area is given in the Master_TOC (see chapter 3.1.1.6.12). Both the EKB1 Area and the EKB2 Area have a length of 62 Sectors. The EKB1 Area and the EKB2 Area are not present in discs according to the Super Audio CD Specification version 1.3 or lower.

2.2.4 Master TOC Area

The Master TOC Area contains three identical copies of the Master TOC. The Master TOC describes the High Density layer of a Super Audio CD. The three instances of the Master TOC are stored starting at LSN 510, 520 and 530. Figure 2-4 shows the structure of the Master TOC Area.

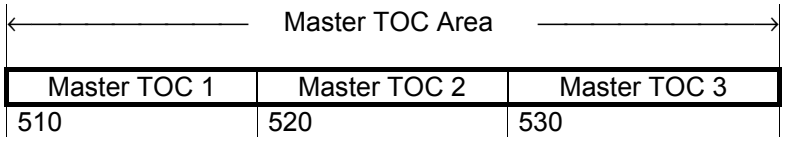

Figure 2-4 : Structure of the Master TOC Area

2.2.4.1 Master TOC

The Master TOC has a fixed size of 10 Sectors. The content of the Master TOC is defined in chapter 3.1. Figure 2-5 shows the structure of the Master TOC.

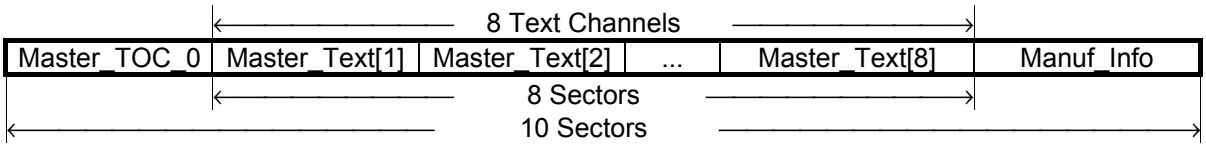

Figure 2-5 : Structure of the Master TOC

2.2.5 Rev TOC Area

The Rev TOC Area contains the Revocation_TOC, see chapter 3.5. The Rev TOC Area starts at LSN 540, the length of the Rev TOC Area is one Sector. The content of the Revocation_TOC is defined in chapter 3.5. The Rev TOC Area is not present in discs according to the Super Audio CD Specification version 1.3 or lower.

2.2.6 Audio Areas

Each Audio Area according to this version of the Super Audio CD Specification contains Area TOC-1, the Track Area, Area TOC-2 and optionally Area TOC-3 and Area TOC-4. Area TOC-2 is a copy of Area TOC-1, Area TOC-4 is a copy of Area TOC-3. Note that discs according to the Super Audio CD Specification Version 1.3 or lower only contain Area TOC-1 and Area TOC-2.

The format of Area TOC-1 and Area TOC-2 is according to Version 1.3 of the Super Audio CD Specification. Area TOC-3 and Area TOC-4 contain the information needed by Super Audio CD players according to the Super Audio CD Specification Version 2.0 or higher. Both Area TOC-1, Area TOC -2, Area TOC -3 and Area TOC -4 contain control information for the Audio Area.

The Track Area contains the Tracks with audio information.
The audio information is stored in Audio Tracks. All 2-Channel Stereo Tracks are grouped in the 2-Channel Stereo Area. All Multi Channel Tracks are grouped in the Multi Channel Area. Each Audio Area contains Area TOC-1, a Track Area with Audio Tracks and Area TOC-2. Each Audio Area optionally contains Area TOC-3 and Area TOC-4.
Gaps between the Track Area and both Area TOC-3 and Area TOC-4 are allowed. Gaps between Area TOC-1 and Area TOC-3 and between Area TOC-4 and Area TOC-2 are allowed. The content of all Sectors in the gaps is Reserved.

If a 2-Channel Stereo Area is present, the 2-Channel Stereo Area must start at LSN 544. If a 2-Channel Stereo Area is not present, the Multi Channel Area must start at LSN 544. Note that the 2-Channel Stereo Area must be present on discs according to the Super Audio CD Specification Version 2.0 or higher, and that the 2-Channel Stereo Area optionally is present on discs according to the Super Audio CD Specification Version 1.3 or lower.

The structure of an Audio Area is given in Figure 2-6.

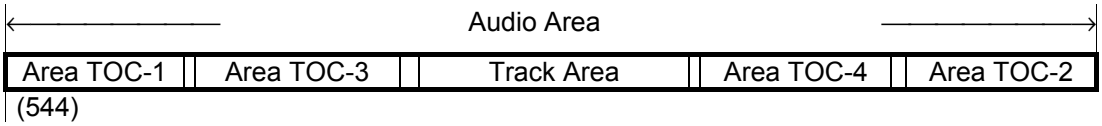

Figure 2-6 : Structure of an Audio Area

2.2.6.1 Area TOC

The content of Area TOC-1 is equal to the content of Area TOC-2. Area TOC-1 and Area TOC-2 are referred to as Area TOC-A. The content of Area TOC-3 is equal to the content of Area TOC-4. Area TOC-3 and Area TOC-4 are referred to as Area TOC-B.

Area TOC-A contains information according to the definitions in Super Audio CD Part 2 Version 1.3. Area TOC-B contains features not defined in Super Audio CD Part 2 Version 1.3 or lower. Area TOC-B is intended to be used by players according to version 2.0 or higher of the Super Audio CD specification. Area TOC-A is intended for players according to Super Audio CD specification version 1.3 or lower. Note that both Area TOC-A and Area TOC-B must have the same value of Spec_Version.

The start location and length of Area TOC-1, Area TOC-2, Area TOC-3 and Area TOC-4 is for each Audio Area given in the Master TOC, see chapter 3.1.1.6. The content of the Area TOC-1, Area TOC-2, Area TOC-3 and Area TOC-4 is defined in chapter 3.2. Figure 2-7 shows the structure of Area TOC-1, Area TOC-2, Area TOC-3 and Area TOC-4.

Area name	Number of Sectors
Area_TOC_0	1
Track_List_1	1
Track_List_2	1
ISRC_and_Genre_List	2
Access_List	p
Track_Text	q
Index_List	r
Track_WebLink_List	s
Track_List_3	t
Set_Of_PlayLists	u

Figure 2-7 : Structure of Area TOC-1, Area TOC-2, Area TOC-3 and Area TOC-4

Area_TOC_0, Track_List_1, Track_List_2 and ISRC_and_Genre_List must always be present in an Area TOC. The presence of Track_Text, Index_List, Track_WebLink_List and Set_Of_PlayLists are optional. Track_List_3 is conditionally present, see chapter 3.2.9. If Access_List and/or Track_Text and/or Index_List and/or Track_WebLink_List and/or Track_List_3 and/or Set_Of_PlayLists are present in an Area TOC, Area_TOC_0 contains pointers to these lists. Access_List must only be present if an Audio Area is DST coded (see chapter 3.2.1.2.3.1). The values of p, q, r, s, t and u are given in chapter 3.2.
In Area TOC-A, Track_List_3 and Set_Of_PlayLists are not allowed. In Area TOC-A the values of t and u must be equal to zero.

2.2.6.2 Track Area

Where the term Track Area is used, this refers to the Tracks in Track Area-A for Area TOC-A and to the Tracks in Track Area-B for Area TOC-B.
The start and end position of a Track Area are given in the Area TOC for that Audio Area (see chapters 3.2.1.2.13 and 3.2.1.2.14). The last Sector of the Track Area is the last Sector used by the last Track of that Track Area. Maximum 255 Tracks are allowed in a Track Area. The layout of a Track Area is given in Figure 2-8.

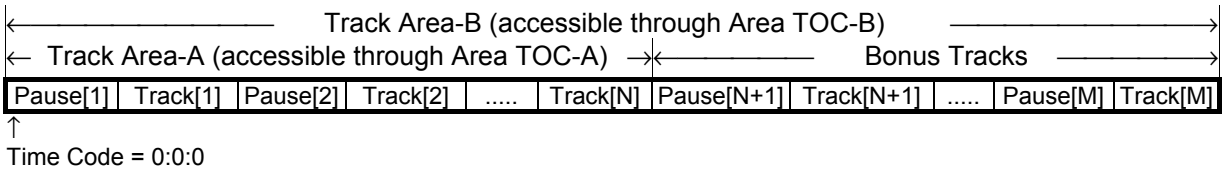

Figure 2-8 : Layout of the Track Area

For Track[1] to Track[N], the start location, length and other information is given in the Area TOC-A and in Area TOC-B (see chapter 3.2) of the Audio Area where the Track belongs to.
For Track[N+1] to Track[M], the start location, length and other information is given only in Area TOC-B (see chapter 3.2) of the Audio Area where the Track belongs to. Track[N+1] to Track[M] are called Bonus Tracks. The presence of Bonus Tracks is optional.
The value of N_Tracks in Area TOC-A is equal to the value of N. The value of N_Tracks in Area TOC-B is equal to the value of M. The value of M must be greater than or equal to the value of N. The presence of Bonus Tracks is indicated by the value of First_Bonus_Track_Number in Area Data of Area TOC-B, see chapter 3.2.1.2.12. The total playing time of Track Area-A must minimally be equal to 80% of the total playing time of Track Area-B.

2.2.6.2.1 Time Code

The Track Area contains a Time Code. The Time Code starts with zero (0:0:0) at the start of Pause[1]. The Time Code increments in steps of 1/75 Second to the end of the Track Area. A time unity of 1/75 Second is called a Frame. The Time Code is encoded in the Multiplexed Frames, see chapter 5.3.1.1.5.1.

2.2.6.2.2 Track

All Tracks in a Track Area are consecutively numbered starting with one. It is not allowed for Tracks to overlap. However, the first Frame of a Track can be located in a Sector that contains the end of the last Frame of the previous Track. The minimum length of a Track is 1 Second. A Track is subdivided into Indexes, see Figure 2-9.

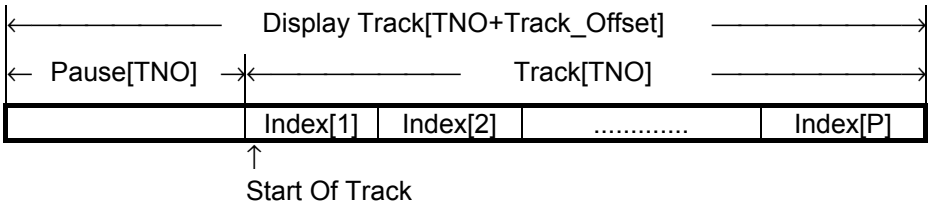

Figure 2-9 : Layout of a Track

Start Of Track for Track[TNO] is the Track_Start_Time_Code[TNO] as defined in Track_List_2 of the Area TOC, see chapter 3.2.3.
Note: Pause[TNO] is not encoded as Index[0].

2.2.6.2.2.1 Index

The first Index in a Track is Index[1]. Indexes must be numbered consecutively to the end of the Track. Maximum 255 Indexes are allowed per Track. The start of Index[1] is equal to the start of the Track. For Index[2] and following Indexes, the start Time Code is given in the Area TOC, see chapter 3.2.7. The minimum length of an Index is 1 Frame.

2.2.6.2.3 Pause

Each Track is preceded by a Pause. The Pause area that precedes Track[i] is called Pause[i]. The minimum length of a Pause is zero Seconds.

2.2.6.2.4 Display Time

It is recommended that an Super Audio CD player calculates a Time Code relative to the Start Of Track (see Figure 2-9), for display purposes.

2.2.6.2.5 Display Track

It is recommended that the Track Number displayed by an Super Audio CD player is equal to the value of TNO+Track_Offset (see chapter 3.2.1.2.10), from the start of Pause[TNO] to the end of Track[TNO].

2.2.6.2.6 General Audio Encoding

Depending on the Area, the encoded audio information in the Audio Tracks and in the preceding Pauses can be either Plain DSD, or DST coded DSD. The data in the Audio Tracks and Pauses will be multiplexed with Supplementary Data.
The audio information in the 2-Channel Stereo Area can be either Plain DSD or DST coded DSD. If the 2-Channel Stereo Area contains Bonus Tracks, the 2-Channel Stereo Area must be DST coded DSD. The audio information in the Multi Channel Area must be DST coded DSD.
The multiplexed audio information in an Area with Plain DSD has a Constant Byte Rate. The multiplexed audio information in an Area with DST coded DSD has a Variable Byte Rate.

2.2.7 Extension Area

The contents of the Extension Area is defined in chapter 6. The Extension Area starts with the Extension_TOC followed by Extension_Data. A gap between the Extension_TOC and Extension_Data is allowed. The structure of the Extension Area is shown in Figure 2-10.

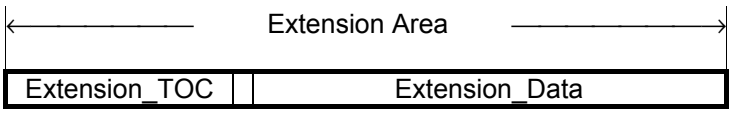

Figure 2-10 : Structure of the Extension Area

The start locations of Extension_TOC and Extension_Data, the length of the Extension_TOC, and the end location of Extension_Data are given in Disc_Info_2 in the Master TOC, see chapter 3.1.1.6.

2.2.8 Revocation Data Area

The Revocation Data Area contains data to be used by Approved Secure Digital Interfaces, see Super Audio CD Part 3-General. The Revocation TOC (see chapter 3.5) contains pointers to the actual revocation data. The Revocation Data Area has a fixed size of 1024 Sectors. The Revocation Data Area is not present on discs according to the Super Audio CD Specification version 1.3 or lower.

2.2.9 Extra Data Area

The Extra Data Area can optionally be used to store additional audio related information.

3. TOC structure

The TOC of the high density layer has a two level structure, the highest level is the Master TOC, (see chapter 3.1) and the lower level is formed by the Area TOC (see chapter 3.2). The Master TOC contains album information and disc information. The Area TOC contains area information and track information. The Extra Data Area does not contain an Area TOC.

3.1 Master TOC

The Master TOC has a fixed length of 10 Sectors. The first Sector of the Master TOC is called Master_TOC_0. Master_TOC_0 contains general information on the disc, such as the size and location of the Audio Areas, album information, disc catalog number, disc genre and disc date.
Master_TOC_0 is followed by eight Sectors with maximum eight Text Channels that contain general information on the disc. For each Text Channel a language code and a character set code is defined in Master_TOC_0.
The syntax of the Master TOC is defined in Figure 3-1.

```
                                    # bytes        format
Master_TOC(){
   Master_TOC_0()                     2048         Master_TOC_0
   for (c=1; c<=8; c++)
   {
       Master_Text()[c]               2048         Master_Text
   }
   Manuf_Info()                       2048         Manuf_Info
}
```

Figure 3-1 : Syntax of the Master TOC

3.1.1 Master_TOC_0

The syntax of Master_TOC_0 is defined in Figure 3-2.

```
                                    # bytes        format                 value
Master_TOC_0(){
   M_TOC_0_Header()                   16          M_TOC_0_Header
   Album_Info()                       48          Album_Info
   Disc_Info()                        64          Disc_Info
   Text_Channels()                    40          Text_Channels
   Disc_WebLink_Info()               128          Disc_WebLink_Info
   Disc_Info_2()                      64          Disc_Info_2
   Reserved                       until 2048      Uint8                   0
}
```

Figure 3-2 : Syntax of Master_TOC_0

3.1.1.1 M_TOC_0_Header

The syntax of M_TOC_0_Header is defined in Figure 3-3.

```
                                    # bytes        format                 value
M_TOC_0_Header(){
   Master_TOC_Signature               8           String              "SACDMTOC"
   Spec_Version                       2           Spec_Version
   Reserved                           6           Uint8                   0
}
```

Figure 3-3 : Syntax of Master_TOC_0_Header

3.1.1.1.1 Master_TOC_Signature

Master_TOC_Signature is an 8 byte string identifying the Master TOC. The value of Master_TOC_Signature must be “SACDMTOC“ ($53 $41 $43 $44 $4D $54 $4F $43).

3.1.1.1.2 Spec_Version

Spec_Version must contain the Spec_Version as is defined in chapter 1.7.2.4.

3.1.1.2 Album_Info

The syntax of Album_Info is defined in Figure 3-4.

```
                                    # bytes        format                  value
Album_Info(){
   Album_Set_Size                     2           Uint16                1..65535
   Album_Sequence_Number              2           Uint16           1..Album_Set_Size
   Reserved                           4           Uint8                    0
   Album_Catalog_Number              16           String
   Album_Genre                       16           Genre4
   Reserved                           8           Uint8                    0
   /*** Album text fields are in Master_Text[c] with c=1..8 ***/
}
```

Figure 3-4 : Syntax of Album_Info

3.1.1.2.1 Album_Set_Size

Album_Set_Size must contain the total number of discs in this album. The minimum allowed value for Album_Set_Size is one. All discs in one Album must have the same value of Album_Set_Size.

3.1.1.2.2 Album_Sequence_Number

Album_Sequence_Number must contain the sequence number of this disc within the album. The Album_Sequence_Number must be numbered consecutively over all discs in an Album. The first disc from an Album must have set the Album_Sequence_Number equal to one.
The minimum allowed value for Album_Sequence_Number is one. The maximum allowed value of Album_Sequence_Number is the value of Album_Set_Size.

3.1.1.2.3 Album_Catalog_Number

Album_Catalog_Number contains the catalog number of this album. All discs in one Album must have the same Album_Catalog_Number. It is recommended to use the UPC/EAN catalog number. The Album_Catalog_Number field must be padded at the end with bytes with space characters ($20). If an Album_Catalog_Number is not used, all bytes must be set to zero.

3.1.1.2.4 Album_Genre

The syntax of Album_Genre is defined in chapter 1.7.2.3. Album_Genre contains minimum zero and maximum four genres associated with this Super Audio CD album. It is recommended that all discs in one Album have the same Album_Genre.

3.1.1.3 Disc_Info

The syntax of Disc_Info is defined in Figure 3-5.

```
                                    # bytes        format                  value
Disc_Info(){
   2CH_TOC_1_Address                   4           Uint32                  0, 544
   2CH_TOC_2_Address                   4           Uint32
   MC_TOC_1_Address                    4           Uint32
   MC_TOC_2_Address                    4           Uint32
   Disc_Flags                          1           Disc_Flags
   Reserved                            3           Uint8                     0
   2CH_TOC_Length                      2           Uint16                  0, 5..
   MC_TOC_Length                       2           Uint16                 0, 37..
   Disc_Catalog_Number                16           String
   Disc_Genre                         16           Genre4
   Disc_Date                           4           Date
   Reserved                            4           Uint8                     0
   /*** Disc text fields are in Master_Text[c] with c=1..8 ***/
}
```

Figure 3-5 : Syntax of Disc_Info

3.1.1.3.1 2CH_TOC_1_Address

2CH_TOC_1_Address is a 4 byte integer containing the Logical Sector Number (LSN) of the first Sector of Area TOC-1 in the 2-Channel Stereo Area. If the 2-Channel Stereo Area is not present, the value of 2CH_TOC_1_Address must be zero. If the 2-Channel Stereo Area is present, the value of 2CH_TOC_1_Address must be 544.

3.1.1.3.2 2CH_TOC_2_Address

2CH_TOC_2_Address is a 4 byte integer containing the LSN of the first Sector of Area TOC-2 in the 2-Channel Stereo Area. If the 2-Channel Stereo Area is not present, the value of 2CH_TOC_2_Address must be zero.

3.1.1.3.3 MC_TOC_1_Address

MC_TOC_1_Address is a 4 byte integer containing the LSN of the first Sector of Area TOC-1 in the Multi Channel Area. If the Multi Channel Area is not present, the value of MC_TOC_1_Address must be zero.

3.1.1.3.4 MC_TOC_2_Address

MC_TOC_2_Address is a 4 byte integer containing the LSN of the first Sector of Area TOC-2 in the Multi Channel Area. If the Multi Channel Area is not present, the value of MC_TOC_2_Address must be zero.

3.1.1.3.5 Disc_Flags

The format of Disc_Flags must be as defined in Figure 3-6.

b7	b6	b5	b4	b3	b2	b1	b0
Hybr	Reserved = 0						

Figure 3-6 : Format of Disc_Flags

3.1.1.3.5.1 Hybr bit

The Hybr bit must be set to one on a Hybrid Disc. The Hybr bit must be set to zero on a not-Hybrid Disc.

3.1.1.3.6 2CH_TOC_Length

2CH_TOC_Length must contain the length in Sectors of Area TOC-A in the 2-Channel Stereo Area. If the 2-Channel Stereo Area is not present, the value of 2CH_TOC_Length must be zero.

3.1.1.3.7 MC_TOC_Length

MC_TOC_Length must contain the length in Sectors of Area TOC-A in the Multi Channel Area. If the Multi Channel Area is not present, the value of MC_TOC_Length must be zero.

3.1.1.3.8 Disc_Catalog_Number

Disc_Catalog_Number is a 16 byte String that uniquely identifies each disc in an Album. The string is padded at the end with space characters ($20). If a Disc_Catalog_Number is not used, all bytes must be set to zero.

3.1.1.3.9 Disc_Genre

The syntax of Disc_Genre is defined in chapter 1.7.2.3. Disc_Genre contains minimum zero and maximum four genres associated with this Super Audio CD disc.

3.1.1.3.10 Disc_Date

Disc_Date contains the creation date of the disc. For the encoding of Disc_Date, see chapter 1.7.2.1. If Disc_Date is not used, all fields of Disc_Date must be set to zero.

3.1.1.4 Text_Channels

Text_Channels contains the definition of the Text Channels used in the Master TOC. All text in Master_Text[c] must be according to the definitions in Text_Channels for Text Channel c. For the definition of Text_Channels, see chapter 1.7.2.8.

3.1.1.5 Disc_WebLink_Info

The syntax of Disc_WebLink_Info is defined in Figure 3-7.

```
                                    # bytes        format                  value
Disc_WebLink_Info(){
   Disc_WebLink                       n           WebLink_String
   Reserved                         128-n         Uint8                     0
}
```

Figure 3-7 : Syntax of Disc_WebLink_Info

3.1.1.5.1 Disc_WebLink

Disc_WebLink contains a WebLink_String (see chapter 1.7.2.11) that points to a web page with information about this disc.

3.1.1.6 Disc_Info_2

The syntax of Disc_Info_2 is defined in Figure 3-8.

	# bytes	format	value
Disc_Info_2(){			
2CH_TOC_3_Address	4	Uint32	
2CH_TOC_4_Address	4	Uint32	
MC_TOC_3_Address	4	Uint32	
MC_TOC_4_Address	4	Uint32	
Reserved	4	Uint8	0
2CH_TOC_B_Length	2	Uint16	
MC_TOC_B_Length	2	Uint16	
E_TOC_Address	4	Uint32	
E_TOC_Length	2	Uint16	
Reserved	2	Uint8	0
E_Data_Start_Address	4	Uint32	
E_Data_End_Address	4	Uint32	
EKB1_Area_Address	4	Uint32	
EKB2_Area_Address	4	Uint32	
Rev_Area_Start_Address	4	Uint32	
Rev_Area_End_Address	4	Uint32	
Reserved	8	Uint8	0
}			

Figure 3-8 : Syntax of Disc_Info_2

3.1.1.6.1 2CH_TOC_3_Address

2CH_TOC_3_Address is a 4 byte integer containing the Logical Sector Number (LSN) of the first Sector of Area TOC-3 in the 2-Channel Stereo Area. A value of zero for 2CH_TOC_3_Address means that Area TOC-3 is not available in the 2-Channel Stereo Area.

3.1.1.6.2 2CH_TOC_4_Address

2CH_TOC_4_Address is a 4 byte integer containing the LSN of the first Sector of Area TOC-4 in the 2-Channel Stereo Area. A value of zero for 2CH_TOC_4_Address means that Area TOC-4 is not available in the 2-Channel Stereo Area.

3.1.1.6.3 MC_TOC_3_Address

MC_TOC_3_Address is a 4 byte integer containing the LSN of the first Sector of Area TOC-3 in the Multi Channel Area. A value of zero for MC_TOC_3_Address means that Area TOC-3 is not available in the Multi Channel Area.

3.1.1.6.4 MC_TOC_4_Address

MC_TOC_4_Address is a 4 byte integer containing the LSN of the first Sector of Area TOC-4 in the Multi Channel Area. A value of zero for MC_TOC_4_Address means that Area TOC-4 is not available in the Multi Channel Area.

3.1.1.6.5 2CH_TOC_B_Length

2CH_TOC_B_Length must contain the length in Sectors of Area TOC-B in the 2-Channel Stereo Area. If Area TOC-B is not present in the 2-Channel Stereo Area, or if the 2-Channel Stereo Area is not present, the value of 2CH_TOC_B_Length must be zero.

3.1.1.6.6 MC_TOC_B_Length

MC_TOC_B_Length must contain the length in Sectors of Area TOC-B in the Multi Channel Area. If Area TOC-B is not present in the Multi Channel Area, or if the Multi Channel Area is not present, the value of MC_TOC_B_Length must be zero.

3.1.1.6.7 E_TOC_Address

E_TOC_Address is a 4 byte integer containing the Logical Sector Number (LSN) of the first Sector of the Extension_TOC. If the Extension Area is not present, the value of E_TOC_Address must be zero.

3.1.1.6.8 E_TOC_Length

E_TOC_Length is a 2 byte integer containing the length in Sectors of the Extension_TOC, see chapter 6.1. If the Extension Area is not present, the value of E_TOC_Length must be zero.

3.1.1.6.9 E_Data_Start_Address

E_Data_Start_Address is a 4 byte integer containing the Logical Sector Number (LSN) of the first Sector of Extension_Data. If the Extension Area is not present, the value of E_Data_Start_Address must be zero.

3.1.1.6.10 E_Data_End_Address

E_Data_End_Address is a 4 byte integer containing the Logical Sector Number (LSN) of the last Sector of the Extension_Data. If the Extension Area is not present, the value of E_Data_End_Address must be zero.

3.1.1.6.11 EKB1_Area_Address

EKB1_Area_Address is a 4 byte integer containing the Logical Sector Number (LSN) of the first Sector of the EKB1 Area. In discs according to this version of the Super Audio CD Specification the value of EKB1_Area_Address must be 448.
A value of zero for EKB1_Area_Address means the EKB1 Area is not present, the EKB1 Area is not present in discs according to the Super Audio CD Specification version 1.3 or lower.

3.1.1.6.12 EKB2_Area_Address

EKB2_Area_Address is a 4 byte integer containing the Logical Sector Number (LSN) of the first Sector of the EKB2 Area. A value of zero for EKB2_Area_Address means the EKB2 Area is not present, the EKB2 Area is not present in discs according to the Super Audio CD Specification version 1.3 or lower.

3.1.1.6.13 Rev_Area_Start_Address

Rev_Area_Start_Address is a 4 byte integer containing the LSN of the first Sector of the Revocation Data Area. If the Revocation Data Area is not present, the value of Rev_Area_Start_Address must be zero. The Revocation Data Area is not present in discs according to the Super Audio CD Specification version 1.3 or lower.

3.1.1.6.14 Rev_Area_End_Address

Rev_Area_End_Address is a 4 byte integer containing the LSN of the last Sector of the Revocation Data Area. If the Revocation Data Area is not present, the value of Rev_Area_End_Address must be zero. The Revocation Data Area is not present in discs according to the Super Audio CD Specification version 1.3 or lower.

3.1.2 Master_Text

Master_Text[c] contains all general text information that is related with the Album and with the Disc, for Text Channel c. Every Text Channel has an associated Master Text structure. The syntax of Master_Text is defined in Figure 3-9. The size of Master_Text is one Sector. For all Master_Text[c] with c>N_Text_Channels (see chapter 3.1.1.4 and chapter 1.7.2.8), all fields of Master_Text[c], except Master_Text_Signature, must be set to zero.

```
                                    # bytes          format                    value
Master_Text()[c] {
   Master_Text_Signature              8              String                  "SACDText"
   Reserved                           8              Uint8                        0
   Album_Title_Ptr                    2              Uint16
   Album_Artist_Ptr                   2              Uint16
   Album_Publisher_Ptr                2              Uint16
   Album_Copyright_Ptr                2              Uint16
   Album_Title_Phonetic_Ptr           2              Uint16
   Album_Artist_Phonetic_Ptr          2              Uint16
   Album_Publisher_Phonetic_Ptr       2              Uint16
   Album_Copyright_Phonetic_Ptr       2              Uint16

   Disc_Title_Ptr                     2              Uint16
   Disc_Artist_Ptr                    2              Uint16
   Disc_Publisher_Ptr                 2              Uint16
   Disc_Copyright_Ptr                 2              Uint16
   Disc_Title_Phonetic_Ptr            2              Uint16
   Disc_Artist_Phonetic_Ptr           2              Uint16
   Disc_Publisher_Phonetic_Ptr        2              Uint16
   Disc_Copyright_Phonetic_Ptr        2              Uint16
   Reserved                          16              Uint8                        0

   if (Album_Title_Ptr!=0)
       { Album_Title                                 Special_String }
   if (Album_Artist_Ptr!=0)
       { Album_Artist                                Special_String }
   if (Album_Publisher_Ptr!=0)
       { Album_Publisher                             Special_String }
   if (Album_Copyright_Ptr!=0)
       { Album_Copyright                             Special_String }
   if (Album_Title_Phonetic_Ptr!=0)
       { Album_Title_Phonetic                        Special_String }
   if (Album_Artist_Phonetic_Ptr!=0)
       { Album_Artist_Phonetic                       Special_String }
   if (Album_Publisher_Phonetic_Ptr!=0)
       { Album_Publisher_Phonetic                    Special_String }
   if (Album_Copyright_Phonetic_Ptr!=0)
       { Album_Copyright_Phonetic                    Special_String }

   if (Disc_Title_Ptr!=0)
       { Disc_Title                                  Special_String }
   if (Disc_Artist_Ptr!=0)
       { Disc_Artist                                 Special_String }
   if (Disc_Publisher_Ptr!=0)
       { Disc_Publisher                              Special_String }
   if (Disc_Copyright_Ptr!=0)
       { Disc_Copyright                              Special_String }
   if (Disc_Title_Phonetic_Ptr!=0)
       { Disc_Title_Phonetic                         Special_String }
   if (Disc_Artist_Phonetic_Ptr!=0)
       { Disc_Artist_Phonetic                        Special_String }
   if (Disc_Publisher_Phonetic_Ptr!=0)
       { Disc_Publisher_Phonetic                     Special_String }
   if (Disc_Copyright_Phonetic_Ptr!=0)
       { Disc_Copyright_Phonetic                     Special_String }
   Reserved                          Until 2048      Uint8                        0
}
```

Figure 3-9 : Syntax of Master_Text

It is recommended that all discs in one Album contain the same Album related text. If Album_Set_Size (see chapter 3.1.1.2.1) is equal to one, it is recommended that the Album related text is equal to the Disc related text.

3.1.2.1 Master_Text_Signature

Master Text Signature is an 8 byte string identifying the text Sectors of the Master TOC. The value of Master_Text_Signature must be “SACDText“ ($53 $41 $43 $44 $54 $65 $78 $74). All eight Master_Text Sectors must have a Master_Text_Signature.

3.1.2.2 Album_Title_Ptr

Album_Title_Ptr contains a pointer to Album_Title. The pointer value is the byte position in the Sector of the first character of Album_Title. If Album_Title is not used, the Album_Title field is not present and Album_Title_Ptr must be set to zero. If Album_Title is used, the Album_Title field is present and Album_Title_Ptr must have the value 64.

3.1.2.3 Album_Artist_Ptr

Album_Artist_Ptr contains a pointer to Album_Artist. The pointer value is the byte position in the Sector of the first character of Album_Artist. If Album_Artist is not used, the Album_Artist field is not present and Album_Artist_Ptr must be set to zero.

3.1.2.4 Album_Publisher_Ptr

Album_Publisher_Ptr contains a pointer to Album_Publisher. The pointer value is the byte position in the Sector of the first character of Album_Publisher. If Album_Publisher is not used, the Album_Publisher field is not present and Album_Publisher_Ptr must be set to zero.

3.1.2.5 Album_Copyright_Ptr

Album_Copyright_Ptr contains a pointer to Album_Copyright. The pointer value is the byte position in the Sector of the first character of Album_Copyright. If Album_Copyright is not used, the Album_Copyright field is not present and Album_Copyright_Ptr must be set to zero.

3.1.2.6 Album_Title_Phonetic_Ptr

Album_Title_Phonetic_Ptr contains a pointer to Album_Title_Phonetic. The pointer value is the byte position in the Sector of the first character of Album_Title_Phonetic. If Album_Title_Phonetic is not used, the Album_Title_Phonetic field is not present and Album_Title_Phonetic_Ptr must be set to zero.

3.1.2.7 Album_Artist_Phonetic_Ptr

Album_Artist_Phonetic_Ptr contains a pointer to Album_Artist_Phonetic. The pointer value is the byte position in the Sector of the first character of Album_Artist_Phonetic. If Album_Artist_Phonetic is not used, the Album_Artist_Phonetic field is not present and Album_Artist_Phonetic_Ptr must be set to zero.

3.1.2.8 Album_Publisher_Phonetic_Ptr

Album_Publisher_Phonetic_Ptr contains a pointer to Album_Publisher_Phonetic. The pointer value is the byte position in the Sector of the first character of Album_Publisher_Phonetic. If Album_Publisher_Phonetic is not used, the Album_Publisher_Phonetic field is not present and Album_Publisher_Phonetic_Ptr must be set to zero.

3.1.2.9 Album_Copyright_Phonetic_Ptr

Album_Copyright_Phonetic_Ptr contains a pointer to Album_Copyright_Phonetic. The pointer value is the byte position in the Sector of the first character of Album_Copyright_Phonetic. If Album_Copyright_Phonetic is not used, the Album_Copyright_Phonetic field is not present and Album_Copyright_Phonetic_Ptr must be set to zero.

3.1.2.10 Disc_Title_Ptr

Disc_Title_Ptr contains a pointer to Disc_Title. The pointer value is the byte position in the Sector of the first character of Disc_Title. If Disc_Title is not used, the Disc_Title field is not present and Disc_Title_Ptr must be set to zero.

3.1.2.11 Disc_Artist_Ptr

Disc_Artist_Ptr contains a pointer to Disc_Artist. The pointer value is the byte position in the Sector of the first character of Disc_Artist. If Disc_Artist is not used, the Disc_Artist field is not present and Disc_Artist_Ptr must be set to zero.

3.1.2.12 Disc_Publisher_Ptr

Disc_Publisher_Ptr contains a pointer to Disc_Publisher. The pointer value is the byte position in the Sector of the first character of Disc_Publisher. If Disc_Publisher is not used, the Disc_Publisher field is not present and Disc_Publisher_Ptr must be set to zero.

3.1.2.13 Disc_Copyright_Ptr

Disc_Copyright_Ptr contains a pointer to Disc_Copyright. The pointer value is the byte position in the Sector of the first character of Disc_Copyright. If Disc_Copyright is not used, the Disc_Copyright field is not present and Disc_Copyright_Ptr must be set to zero.

3.1.2.14 Disc_Title_Phonetic_Ptr

Disc_Title_Phonetic_Ptr contains a pointer to Disc_Title_Phonetic. The pointer value is the byte position in the Sector of the first character of Disc_Title_Phonetic. If Disc_Title_Phonetic is not used, the Disc_Title_Phonetic field is not present and Disc_Title_Phonetic_Ptr must be set to zero.

3.1.2.15 Disc_Artist_Phonetic_Ptr

Disc_Artist_Phonetic_Ptr contains a pointer to Disc_Artist_Phonetic. The pointer value is the byte position in the Sector of the first character of Disc_Artist_Phonetic. If Disc_Artist_Phonetic is not used, the Disc_Artist_Phonetic field is not present and Disc_Artist_Phonetic_Ptr must be set to zero.

3.1.2.16 Disc_Publisher_Phonetic_Ptr

Disc_Publisher_Phonetic_Ptr contains a pointer to Disc_Publisher_Phonetic. The pointer value is the byte position in the Sector of the first character of Disc_Publisher_Phonetic. If Disc_Publisher_Phonetic is not used, the Disc_Publisher_Phonetic field is not present and Disc_Publisher_Phonetic_Ptr must be set to zero.

3.1.2.17 Disc_Copyright_Phonetic_Ptr

Disc_Copyright_Phonetic_Ptr contains a pointer to Disc_Copyright_Phonetic. The pointer value is the byte position in the Sector of the first character of Disc_Copyright_Phonetic. If Disc_Copyright_Phonetic is not used, the Disc_Copyright_Phonetic field is not present and Disc_Copyright_Phonetic_Ptr must be set to zero.

3.1.2.18 Album_Title

If present, Album_Title is a variable length string coded with the character set specified by Character_Set_Code[c]. Album_Title contains text with the album title. The definition of Special_String is given in chapter 1.7.2.6.

3.1.2.19 Album_Artist

If present, Album_Artist is a variable length string coded with the character set specified by Character_Set_Code[c]. Album_Artist contains text with the name of the main artist(s) for the album. The definition of Special_String is given in chapter 1.7.2.6.

3.1.2.20 Album_Publisher

If present, Album_Publisher is a variable length string coded with the character set specified by Character_Set_Code[c]. Album_Publisher contains text with the name of the album publisher. The definition of Special_String is given in chapter 1.7.2.6.

3.1.2.21 Album_Copyright

If present, Album_Copyright is a variable length string coded with the character set specified by Character_Set_Code[c]. Album_Copyright contains text with copyright information on the album. The definition of Special_String is given in chapter 1.7.2.6.

3.1.2.22 Album_Title_Phonetic

If present, Album_Title_Phonetic is a variable length string coded with the character set specified by Character_Set_Code[c]. Album_Title_Phonetic contains phonetic text with the album title. The definition of Special_String is given in chapter 1.7.2.6.

3.1.2.23 Album_Artist_Phonetic

If present, Album_Artist_Phonetic is a variable length string coded with the character set specified by Character_Set_Code[c]. Album_Artist_Phonetic contains phonetic text with the name of the main artist(s) for the album. The definition of Special_String is given in chapter 1.7.2.6.

3.1.2.24 Album_Publisher_Phonetic

If present, Album_Publisher_Phonetic is a variable length string coded with the character set specified by Character_Set_Code[c]. Album_Publisher_Phonetic contains phonetic text with the name of the album publisher. The definition of Special_String is given in chapter 1.7.2.6.

3.1.2.25 Album_Copyright_Phonetic

If present, Album_Copyright_Phonetic is a variable length string coded with the character set specified by Character_Set_Code[c]. Album_Copyright_Phonetic contains phonetic text with copyright information on the album. The definition of Special_String is given in chapter 1.7.2.6.

3.1.2.26 Disc_Title

If present, Disc_Title is a variable length string coded with the character set specified by Character_Set_Code[c]. Disc_Title contains text with the disc title. The definition of Special_String is given in chapter 1.7.2.6.

3.1.2.27 Disc_Artist

If present, Disc_Artist is a variable length string coded with the character set specified by Character_Set_Code[c]. Disc_Artist contains text with the name of the main artist(s) for the disc. The definition of Special_String is given in chapter 1.7.2.6.

3.1.2.28 Disc_Publisher

If present, Disc_Publisher is a variable length string coded with the character set specified by Character_Set_Code[c]. Disc_Publisher contains text with the name of the disc publisher. The definition of Special_String is given in chapter 1.7.2.6.

3.1.2.29 Disc_Copyright

If present, Disc_Copyright is a variable length string coded with the character set specified by Character_Set_Code[c]. Disc_Copyright contains text with copyright information on the disc. The definition of Special_String is given in chapter 1.7.2.6.

3.1.2.30 Disc_Title_Phonetic

If present, Disc_Title_Phonetic is a variable length string coded with the character set specified by Character_Set_Code[c]. Disc_Title_Phonetic contains phonetic text with the disc title. The definition of Special_String is given in chapter 1.7.2.6.

3.1.2.31 Disc_Artist_Phonetic

If present, Disc_Artist_Phonetic is a variable length string coded with the character set specified by Character_Set_Code[c]. Disc_Artist_Phonetic contains phonetic text with the name of the main artist(s) for the disc. The definition of Special_String is given in chapter 1.7.2.6.

3.1.2.32 Disc_Publisher_Phonetic

If present, Disc_Publisher_Phonetic is a variable length string coded with the character set specified by Character_Set_Code[c]. Disc_Publisher_Phonetic contains phonetic text with the name of the disc publisher. The definition of Special_String is given in chapter 1.7.2.6.

3.1.2.33 Disc_Copyright_Phonetic

If present, Disc_Copyright_Phonetic is a variable length string coded with the character set specified by Character_Set_Code[c]. Disc_Copyright_Phonetic contains phonetic text with copyright information on the disc. The definition of Special_String is given in chapter 1.7.2.6.

3.1.3 Manuf_Info

Manuf_Info can contain information stored by the disc manufacturer. The syntax of Manuf_Info is defined in Figure 3-10.

```
                                    # bytes        format                value
Manuf_Info(){
   Manuf_Info_Signature               8           String              "SACD_Man"
   Information()                     2040
}
```

Figure 3-10 : Syntax of Manuf_Info

3.1.3.1 Manuf_Info_Signature

Manuf_Info_Signature is an 8 byte string identifying the Sector with the manufacturer information in the Master TOC. The value of Manuf_Info_Signature must be “SACD_Man“ ($53 $41 $43 $44 $5F $4D $61 $6E).

3.1.3.2 Information

The content and the format of the data in the Information field is decided by the disc manufacturer. If manufacturer information is not stored in this Sector, all bytes in the Information field must be set to zero.

3.2 Area TOC

Two versions of the Area TOC exist, one version (Area TOC-A) is stored in Area TOC-1 and Area TOC-2, the other version (Area TOC-B) is stored in Area TOC-3 and Area TOC-4. Area TOC-A is mandatory and Area TOC-B is optional. Area TOC-1 and Area TOC-3 are stored at the start of an Audio Area. Area TOC-2 and Area TOC-4 are stored at the end of an Audio Area.
The start location of each Area TOC is given in the Master TOC (see chapters 3.1.1.3 and 3.1.1.6). The Area TOC contains information on the Tracks in the Track Area.
The content of Area TOC-A is according to Super Audio CD Part 2 Version 1.3. In addition to the content of Area TOC-A, Area TOC-B can contain information on play list, bonus tracks, text and pictures. Whenever Area TOC-B is equal to Area TOC-A, Area TOC-B can be omitted from the disc.
In this chapter 3.2, the Area TOC data refers to the Tracks in Track Area-A for Area TOC-A and to the Tracks in Track Area-B for Area TOC-B.
The syntax of Area TOC is defined in Figure 3-11.

Area_TOC(){	# bytes	# Sectors	format	Mandatory/ Optional
Area_TOC_0()	2048	1	Area_TOC_0	M
Track_List_1()	2048	1	Track_List_1	M
Track_List_2()	2048	1	Track_List_2	M
ISRC_and_Genre_List()	2*2048	2	ISRC_and_Genre_List	M
Access_List()	0 or 32*2048	0, 32	Access_List	-/M
Track_Text()	q*2048	<= 32	Track_Text	O
Index_List()	r*2048	<= 10	Index_List	O
Track_WebLink_List()	s*2048	<= 4	Track_WebLink_List	O
Track_List_3()	0 or 8*2048	0, 8	Track_List_3	O
Set_Of_PlayLists()	u*2048	<= 5	Set_Of_PlayLists	O
}				

Figure 3-11 : Syntax of Area TOC

If the Audio Area is DST coded, the Access_List must be present. If the Audio Area is Plain DSD coded, the Access_List is not present. Access_List occupies zero or 32 Sectors, the value of p (see chapter 2.2.6.1) is equal to zero or 32.
The fields Track_Text, Index_List and Track_WebLink_List are optional. The fields Track_List_3 and Set_Of_PlayLists are not present in Area TOC-A. The fields Track_List_3 and Set_Of_PlayLists are optional in Area TOC-B. The presence of Track_Text, Index_List, Track_WebLink_List, Track_List_3 and Set_Of_PlayLists is indicated in the List_Pointers field of Area_TOC_0.
Track_Text occupies maximum 32 Sectors (q <= 32). Index_List occupies maximum 10 Sectors (r <= 10). Track_WebLink_List occupies maximum 4 Sectors (s <= 4). Track_List_3 occupies zero or 8 Sectors (t = 0 or t = 8). Set_Of_PlayLists occupies maximum 5 Sectors (u <= 5).
The maximum size of Area TOC-A is 79 Sectors. The maximum size of Area TOC-B is 96 Sectors. In future versions of Super Audio CD Part 2 additional lists can be defined, which can increase the maximum size of Area TOC-B.
Super Audio CD Part 3 can define additional Lists at the end of the Area_TOC.
Track_List_1 contains access information. Track_List_2 contains Time Code information and Track attributes. Track_List_3 contains CCI as well as text and pictures related information.
Note that Area TOC-B is not present on Super Audio CD version 1.3 or lower discs and that Area TOC-B is conditionally present on version 2.0 or higher discs.

3.2.1 Area_TOC_0

The syntax of Area_TOC_0 is defined in Figure 3-12.

	# bytes	format	value
Area_TOC_0(){			
A_TOC_0_Header()	16	A_TOC_0_Header	
Area_Data()	112	Area_Data	
List_Pointers()	16	List_Pointers	
Area_Text()	1904	Area_Text	
}			

Figure 3-12 : Syntax of Area_TOC_0

3.2.1.1 A_TOC_0_Header

The syntax of A_TOC_0_Header is defined in Figure 3-13.

	# bytes	format	value
A_TOC_0_Header(){			
Area_TOC_Signature	8	String	"TWOCHTOC" or "MULCHTOC"
Spec_Version	2	Spec_Version	
Area_TOC_Length	2	Uint16	5..
Reserved	4	Uint8	0
}			

Figure 3-13 : Syntax of A_TOC_0_Header

3.2.1.1.1 Area_TOC_Signature

Area_TOC_Signature is an 8 byte string identifying the first Sector of the Area TOC. For the Area TOC in the 2-Channel Stereo Area, the value of Area_TOC_Signature must be “TWOCHTOC“ ($54 $57 $4F $43 $48 $54 $4F $43). For the Area TOC in the Multi Channel Area, the value of Area_TOC_Signature must be “MULCHTOC“ ($4D $55 $4C $43 $48 $54 $4F $43).

3.2.1.1.2 Spec_Version

Spec_Version must contain the Spec_Version as is defined in chapter 1.7.2.4.

3.2.1.1.3 Area_TOC_Length

Area_TOC_Length contains the length of the Area_TOC in Sectors.
Note: In case additional Lists are defined in Super Audio CD Part 3, these Lists are part of the Area_TOC and therefore their lengths must be included in Area_TOC_Length.

3.2.1.2 Area_Data

The syntax of Area_Data is defined in Figure 3-14.

```
                                  # bytes        format                  value
Area_Data(){
   Max_Byte_Rate                     4           Uint32
   FS_Code                           1           FS_Code                   4
   Area_Flags                        1           Area_Flags
   Reserved                         10           Uint8                     0

   N_Channels                        1           Uint8                   2,5,6
   Area_Config                       1           Area_Config
   Max_Available_Channels            1           Uint8                    0..6
   Area_Mute_Flags                   1           Area_Mute_Flags
   Reserved                         12           Uint8                     0

   Area_Copy_Management              1           Area_Copy_Management
   Reserved                         15           Uint8                     0

   Total_Area_Play_Time              3           Time_Code
   Reserved                          1           Uint8                     0
   Track_Offset                      1           Uint8                   0..255
   N_Tracks                          1           Uint8                   1..255

   First_Bonus_Track_Number          1           Uint8
   Reserved                          1           Uint8                     0

   Track_Area_Start_Address          4           Uint32
   Track_Area_End_Address            4           Uint32

   Text_Channels()                  40           Text_Channels
   Reserved                      until 128       Uint8                     0
}
```

Figure 3-14 : Syntax of Area_Data

3.2.1.2.1 Max_Byte_Rate

Max_Byte_Rate contains the highest Average Byte Rate of the Multiplexed Frames (see chapter 5) as used in this Track Area. The Average Byte Rate must be calculated over 15 Multiplexed Frames. The highest value for Average Byte Rate in the Track Area must be encoded in Max_Byte_Rate. The value of Max_Byte_Rate is expressed in bytes per second.
If the Frame_Format (see chapter 3.2.1.2.3.1) is equal to 0, the Average Byte Rate must be calculated in the following way:

$$\text{Average Byte Rate} = 2048 \times \frac{75}{15} \times \left(\text{Start\_Address}\left(n+15 \right) - \text{Start\_Address}\left(n \right) \right)$$

Where: n is a Frame number between 0 and (Total_Area_Play_Time - 15 - 1).
Start_Address(n) is the LSN of the Sector that contains the first byte of Multiplexed Frame n.

If the Frame_Format is equal to 0 and N_Channels is equal to 2, the maximum allowed value of Max_Byte_Rate is 839680. If the Frame_Format is equal to 0 and N_Channels is equal to 5 or 6, the maximum allowed value of Max_Byte_Rate is 1873920.

If the Frame_Format is equal to 2, the value of Max_Byte_Rate must be 716800. If the Frame_Format is equal to 3, the value of Max_Byte_Rate must be 819200.

3.2.1.2.2 FS_Code

The FS_Code field contains a code for the sampling frequency used for the current Audio Area. The definition of FS_Code is given in Figure 3-15.

FS_Code	Sample_Frequency
0..3	Reserved for future use
4	64 * 44100 Hz
5..255	Reserved for future use

Figure 3-15 : Definition of FS_Code

3.2.1.2.3 Area_Flags

The format of Area_Flags field is defined Figure 3-16.

b7	b6	b5	b4	b3	b2	b1	b0
Reserved = 0				Frame_Format			

Figure 3-16 : Format of Area_Flags

3.2.1.2.3.1 Frame_Format

Frame_Format defines the Frame structure used by the multiplexed audio signal, see chapter 5. The value of Frame_Format is defined in Figure 3-17.

Value	Meaning
0	DST coded. Flexible format, see chapter 5
1	Reserved
2	Fixed format. 2-Channel Stereo, Plain DSD, 3 Frames in 14 Sectors. See chapter 5
3	Fixed format. 2-Channel Stereo, Plain DSD, 3 Frames in 16 Sectors. See chapter 5
4..15	Reserved for future use

Figure 3-17 : Definition of Frame_Format

3.2.1.2.4 N_Channels

N_Channels contains the number of audio channels that are encoded in each Frame of the current Audio Area. N_Channels is equal to the maximum number of Audio Channels that are available in the current Audio Area. For the 2-Channel Stereo Area the value of N_Channels must be set to 2. For the Multi Channel Area the allowed values of N_Channels are 5 and 6.
The relation between N_Channels and the Audio Channel number is defined in chapters 3.2.1.2.4.1 to 3.2.1.2.4.3. Except for the LFE Channel, the Channel Mapping defined in chapters 3.2.1.2.4.1 to 3.2.1.2.4.3 refers to the loudspeaker positions as defined in ITU-R BS.775-1.
For the Audio Channel Numbers used in this chapter, see Channel_Nr in chapters 5.6.3.1 and 5.6.3.2.

3.2.1.2.4.1 Channel Mapping for a 2-Channel Stereo Area (N_Channels=2)

If N_Channels is equal to 2, the Channel Mapping is defined in Figure 3-18.

Audio Channel Number	Channel Mapping
1	Left
2	Right

Figure 3-18 : Definition of Channel Mapping for N_Channels = 2

3.2.1.2.4.2 Channel Mapping for a Multi Channel Area (N_Channels=5)

If N_Channels is equal to 5, the Channel Mapping is defined in Figure 3-19.

Audio Channel Number	Channel Mapping
1	Left
2	Right
3	Center
4	Left Surround
5	Right Surround

Figure 3-19 : Definition of Channel Mapping for N_Channels = 5

3.2.1.2.4.3 Channel Mapping for a Multi Channel Area (N_Channels=6)

If N_Channels is equal to 6, the Channel Mapping is defined in Figure 3-20.

Audio Channel Number	Channel Mapping
1	Left
2	Right
3	Center
4	LFE
5	Left Surround
6	Right Surround

Figure 3-20 : Definition of Channel Mapping for N_Channels = 6

Note: All Audio Channels, including LFE, are always full bandwidth and have equal audio gain.

3.2.1.2.5 Area_Config

The definition of Area_Config is given in Figure 3-21.

b7	b6	b5	b4	b3	b2	b1	b0
Extra_Setting			Loudspeaker_Config				

Figure 3-21 : Definition of Area_Config

3.2.1.2.5.1 Extra_Setting

If N_Channels = 6, Extra_Setting defines the usage of Audio Channel 4 for the current Audio Area. If N_Channels = 2 or N_Channels = 5, the value of Extra_Setting must be zero. If N_Channels = 6, the definition of Extra_Setting is given in Figure 3-22.

Value	Meaning
0	Audio Channel 4 is used for a LFE loudspeaker
1..7	Reserved for future use

Figure 3-22 : Definition of Extra_Setting

Audio Channel 4 should contain an audio signal that is compatible with a LFE loudspeaker system.

3.2.1.2.5.2 Loudspeaker_Config

Loudspeaker_Config gives the global loudspeaker set-up for this Area as used for the recording. The definition of Loudspeaker_Config is given in Figure 3-23.

Value	Meaning
0	2-Channel Stereo set-up
1..2	Reserved for future use
3	5-Channels, set-up according to ITU-R BS.775
4	6-Channels. 5-Channel set-up according to ITU-R BS.775, plus additional Low Frequency Enhancement (LFE) loudspeaker. Also known as “5.1 channel”.
5..31	Reserved for future use

Figure 3-23 : Definition of Loudspeaker_Config

The value 0 for Loudspeaker_Config is only allowed if N_Channels (see chapter 3.2.1.2.4) is equal to 2. The value 3 of Loudspeaker_Config is only allowed if N_Channels is equal to 5. The value 4 of Loudspeaker_Config is only allowed if N_Channels is equal to 6.

3.2.1.2.6 Max_Available_Channels

Max_Available_Channels encodes the maximum number of Audio Channels available per Track of the current Track Area. The minimum value of Max_Available_Channels is zero, the maximum value of Max_Available_Channels is 6. If Max_Available_Channels is equal to zero, the number of Audio Channels available in the current Track Area is equal to the value of N_Channels. If Max_Available_Channels is in the range of 1..6, the number of Audio Channels available in the current Track Area is equal to the value of Max_Available_Channels.

3.2.1.2.7 Area_Mute_Flags

Area_Mute_Flags indicate which Audio Channels are not available in the current Track Area. If an Audio Channel is not available for a Track, this Audio Channel must contain a Silence Pattern during the whole Track. If an Audio Channel is not available in the current Track Area, this Audio Channel must contain a Silence Pattern during all Tracks. The format of the Area_Mute_Flags field is defined Figure 3-24.

b7	b6	b5	b4	b3	b2	b1	b0
Reserved	AMF4	AMF3	AMF2	AMF1	Reserved		

Figure 3-24 : Format of Area_Mute_Flags

AMF1 must be set to one if TMF1[tno] (see 3.2.3.5.2) is set to one for all Tracks in the current Track Area, in all other cases AMF1 must be set to zero. AMF2 must be set to one if TMF2[tno] is set to one for all Tracks in the current Track Area, in all other cases AMF2 must be set to zero. AMF3 must be set to one if TMF3[tno] is set to one for all Tracks in the current Track Area, in all other cases AMF3 must be set to zero. AMF4 must be set to one if TMF4[tno] is set to one for all Tracks in the current Track Area, in all other cases AMF4 must be set to zero.
The values of AMF1 .. AMF4 can be calculated in the following way:

```
AMF1=1; AMF2=1; AMF3=1; AMF4=1;
For (tno=1; tno<=N_Tracks; tno++)
{
    AMF1 = AMF1 & TMF1[tno];
    AMF2 = AMF2 & TMF2[tno];
    AMF3 = AMF3 & TMF3[tno];
    AMF4 = AMF4 & TMF4[tno];
}
```

For the relation between TMF*k* and Audio Channel Numbers, see Figure 3-34.
Note: If N_Channels is equal to 2, both AMF1, AMF2, AMF3 and AMF4 must be zero. If N_Channels is equal to 5, AMF4 must be zero.

3.2.1.2.8 Area_Copy_Management

The format of Area_Copy_Management is defined in Figure 3-25.

b7	b6	b5	b4	b3	b2	b1	b0
Reserved				Track_Attribute			

Figure 3-25 : Format of Area_Copy_Management

3.2.1.2.8.1 Track_Attribute

For the definition of Track_Attribute see Super Audio CD Part 3-General chapter 3.1.

3.2.1.2.9 Total_Area_Play_Time

Total_Area_Play_Time contains the total playing time of the current Track Area. The definition of Time_Code is given in chapter 1.7.2.9. The maximum value of Total_Area_Play_Time is 255:59:74.
The values of Total_Area_Play_Time in Area TOC-A and in Area TOC-B must fulfil the following requirement:

(Total_Area_Play_Time in Area TOC-A) >= 0.8 * roundup(Total_Area_Play_Time in Area TOC-B)

Note: Total_Area_Play_Time = Track_Start_Time_Code[N_Tracks] + Track_Time_Length[N_Tracks]. The Total_Area_Play_Time is equal to the last encoded Time_Code in the Track_Area plus one Frame.

3.2.1.2.10 Track_Offset

Track_Offset contains the Track Number offset for all Tracks in the current Track Area, for display purposes. The Display Track Number is obtained if the Track_Offset is added to the Track Number as given in Track_List_1 and Track_List_2. The first disc from an Album (Album_Sequence_Number = 1, see chapter 3.1.1.2.2) must have a Track_Offset equal to zero. The Display Track Number = Physical Track Number + Track_Offset. The maximum value of Display Track Number is 510. An example for the use of Track_Offset is given in Figure 3-26.

Album_Sequence_Number	N_Tracks	Track_Offset	Track Number	Display Track Number
1	5	0	1	1
			2	2
			3	3
			4	4
			5	5
2	7	5	1	6
			2	7
			3	8
			4	9
			5	10
			6	11
			7	12
3	3	12	1	13
			2	14
			3	15

Figure 3-26 : Example of the use of Track_Offset

3.2.1.2.11 N_Tracks

N_Tracks contains the value of the number of Tracks in the current Track Area. The minimum allowed value for N_Tracks is 1. The maximum allowed value for N_Tracks is 255.
Note: In Area TOC-A the value of N_Tracks excludes the Bonus Tracks, see chapter 2.2.6.2.

3.2.1.2.12 First_Bonus_Track_Number

First_Bonus_Track_Number must contain the Track Number of the first Bonus Track in the current Track Area. If no Bonus Tracks are present in the current Track Area, First_Bonus_Track_Number must be equal to zero.
Notes: In Area TOC-A, First_Bonus_Track_Number must be equal to zero.
The Bonus Tracks occupy maximum 20% of an Audio Area, see chapter 3.2.1.2.9.

3.2.1.2.13 Track_Area_Start_Address

Track_Area_Start_Address must contain the Logical Sector Number (LSN) of the first Sector in the Track Area (see chapter 2.2.6.2). The Sector addressed by Track_Area_Start_Address must contain the Audio_Frame with the Time_Code 0:0:0.

3.2.1.2.14 Track_Area_End_Address

Track_Area_End_Address must contain the Logical Sector Number (LSN) of the last Sector in the Track Area (see chapter 2.2.6.2).
Note: Track_Area_End_Address = Track_Start_Address[N_Tracks] + Track_Length[N_Tracks] - 1

3.2.1.2.15 Text_Channels

Text_Channels contains the definition of the Text Channels used in the Area TOC of the current Track Area. All text in Area_Text (see chapter 3.2.1.4) and Track_Text (see chapter 3.2.6) must be according to the definitions in Text_Channels. For the definition of Text_Channels, see chapter 1.7.2.8. It is recommended that the contents of Text_Channels is the same for Area TOC-A and Area TOC-B.

3.2.1.3 List_Pointers

List_Pointers contains pointers to various lists in the current Area TOC. The value of a pointer gives the Sector offset, from the first Sector of the Area_TOC, to the first Sector of a list. If an offset value is set to zero, the corresponding list is not present. The syntax of List_Pointers is defined Figure 3-27.

	# bytes	format	value
List_Pointers(){			
Track_Text_Ptr	2	Uint16	0,5,37
Index_List_Ptr	2	Uint16	
Access_List_Ptr	2	Uint16	0,5
Track_WebLink_List_Ptr	2	Uint16	
Track_List_3_Ptr	2	Uint16	
Set_Of_PlayLists_Ptr	2	Uint16	
Reserved	4	Uint8	0
}			

Figure 3-27 : Syntax of List_Pointers

Super Audio CD Part 3 can define additional pointers in the Reserved field of List_Pointers.
Note that the order of Track_Text, Index_List and Access_List in the Area TOC (see chapter 3.2) is different from the order of the corresponding pointers in List_Pointers.

3.2.1.3.1 Track_Text_Ptr

Track_Text_Ptr contains a relative pointer to the Sector position of Track_Text. The value Track_Text_Ptr is the offset in Sectors from the start of the current Area_TOC to the first Sector of Track_Text. If the Track_Text is not present, Track_Text_Ptr must be set to zero. If N_Text_Channels (see chapter 3.2.1.2.15 and chapter 1.7.2.8) is zero, Track_Text is not present, and therefore Track_Text_Ptr must be set to zero. For the current version of the Super Audio CD Specification, the allowed values for Track_Text_Ptr are 0, 5 and 37.

3.2.1.3.2 Index_List_Ptr

Index_List_Ptr contains a relative pointer to the Sector position of Index_List. The value of Index_List_Ptr is the offset in Sectors from the start of the current Area_TOC to the first Sector of Index_List. The Index_List is only present if the Track Area contains minimum one Track with two or more Indexes. If the Index_List is not present, Index_List_Ptr must be set to zero.

3.2.1.3.3 Access_List_Ptr

Access_List_Ptr contains a relative pointer to the Sector position of Access_List. The value of Access_List_Ptr is the offset in Sectors from the start of the current Area_TOC to the first Sector of Access_List. The Access_List must be present if the Audio Area is DST coded. If the current Audio Area is Plain DSD, the Access_List is not allowed to be present. If the Access_List is not present, Access_List_Ptr must be set to zero. For the current version of the Super Audio CD Specification, the allowed values for Access_List_Ptr are 0 and 5.

3.2.1.3.4 Track_WebLink_List_Ptr

Track_WebLink_List_Ptr contains a relative pointer to the Sector position of Track_WebLink_List. The value of Track_WebLink_List_Ptr is the offset in Sectors from the start of the current Area_TOC to the first Sector of Track_WebLink_List. The Track_WebLink_List is only present if the Track Area contains minimum one Track with a WebLink. If the Track_WebLink_List is not present, Track_WebLink_List_Ptr must be set to zero.

3.2.1.3.5 Track_List_3_Ptr

Track_List_3_Ptr contains a relative pointer to the Sector position of Track_List_3. The value of Track_List_3_Ptr is the offset in Sectors from the start of the current Area_TOC to the first Sector of Track_List_3. If the Track_List_3 is not present, Track_List_3_Ptr must be set to zero. For Area TOC-A ,Track_List_3_Ptr must be equal to zero.

3.2.1.3.6 Set_Of_PlayLists_Ptr

Set_Of_PlayLists_Ptr contains a relative pointer to the Sector position of Set_Of_PlayLists. The value of Set_Of_PlayLists_Ptr is the offset in Sectors from the start of the current Area_TOC to the first Sector of Set_Of_PlayLists. Set_Of_PlayLists only is present if there is minimally one PlayList for the Track_Area, see chapter 3.2.10. If Set_Of_PlayLists is not present, Set_Of_PlayLists_Ptr must be set to zero. For Area TOC-A, Set_Of_PlayLists_Ptr must be equal to zero.

3.2.1.4 Area_Text

Area_TOC_0 must always contain Area_Text. Area_Text can contain general text for the current Track Area. All unused fields in Area_Text must be set to zero. The syntax of Area_Text is defined in Figure 3-28.

```
                                              # bytes       format          value
Area_Text(){
   for (c=1; c<=N_Text_Channels; c++)
   {
       Area_Description_Ptr[c]                    2          Uint16
       Area_Copyright_Ptr[c]                      2          Uint16
       Area_Description_Phonetic_Ptr[c]           2          Uint16
       Area_Copyright_Phonetic_Ptr[c]             2          Uint16
   }
   Reserved                                  until 208       Uint8             0
   for (c=1; c<=N_Text_Channels; c++)
   {
       if (Area_Description_Ptr[c]!=0)
           { Area_Description[c]                             Special_String }
       if (Area_Copyright_Ptr[c]!=0)
           { Area_Copyright[c]                               Special_String }
       if (Area_Description_Phonetic_Ptr[c]!=0)
           { Area_Description_Phonetic[c]                    Special_String }
       if (Area_Copyright_Phonetic_Ptr[c]!=0)
           { Area_Copyright_Phonetic[c]                      Special_String }
   }
   Reserved                                  until 2048      Uint8             0
}
```

Figure 3-28 : Syntax of Area_Text

The value of N_Text_Channels is encoded in Text_Channels of Area_Data, see chapter 3.2.1.2.15.

3.2.1.4.1 Area_Description_Ptr

Area_Description_Ptr[c] contains the byte position for the first byte of Area_Description for Text Channel c. If Area_Description[c] is not used, Area_Description[c] is not present and Area_Description_Ptr[c] must be set to zero.

3.2.1.4.2 Area_Copyright_Ptr

Area_Copyright_Ptr[c] contains the byte position for the first byte of Area_Copyright for Text Channel c. If Area_Copyright[c] is not used, Area_Copyright[c] is not present and Area_Copyright_Ptr[c] must be set to zero.

3.2.1.4.3 Area_Description_Phonetic_Ptr

Area_Description_Phonetic_Ptr[c] contains the byte position for the first byte of Area_Description_Phonetic for Text Channel c. If Area_Description_Phonetic[c] is not used, Area_Description_Phonetic[c] is not present and Area_Description_Phonetic_Ptr[c] must be set to zero.

3.2.1.4.4 Area_Copyright_Phonetic_Ptr

Area_Copyright_Phonetic_Ptr[c] contains the byte position for the first byte of Area_Copyright_Phonetic for Text Channel c. If Area_Copyright_Phonetic[c] is not used, Area_Copyright_Phonetic[c] is not present and Area_Copyright_Phonetic_Ptr[c] must be set to zero.

3.2.1.4.5 Area_Description

Area_Description[c] contains a description of the current Track Area in Text Channel c. The character set and language code for Text Channel c is defined in Text_Channels of Area_Data, see chapter 3.2.1.2.15. The definition of Special_String is given in chapter 1.7.2.6.

3.2.1.4.6 Area_Copyright

Area_Copyright[c] contains copyright information of the current Track Area in Text Channel c. The character set and language code for Text Channel c is defined in Text_Channels of Area_Data, see chapter 3.2.1.2.15. The definition of Special_String is given in chapter 1.7.2.6.

3.2.1.4.7 Area_Description_Phonetic

Area_Description_Phonetic[c] contains a phonetic description of the current Track Area in Text Channel c. The character set and language code for Text Channel c is defined in Text_Channels of Area_Data, see chapter 3.2.1.2.15. The definition of Special_String is given in chapter 1.7.2.6.

3.2.1.4.8 Area_Copyright_Phonetic

Area_Copyright_Phonetic[c] contains phonetic copyright information of the current Track Area in Text Channel c. The character set and language code for Text Channel c is defined in Text_Channels of Area_Data, see chapter 3.2.1.2.15. The definition of Special_String is given in chapter 1.7.2.6.

3.2.2 Track_List_1

Track_List_1 contains the start Sector address and length for all Tracks in the current Track Area. Track_List_1 must always be present in an Area TOC. The length of Track_List_1 is always one Sector. The syntax of Track_List_1 is defined in Figure 3-29.

```
                                    # bytes         format                   value
Track_List_1(){
   Track_List_1_Signature              8            String                "SACDTRL1"
   for (tno=1; tno<=N_Tracks; tno++)
   {
       Track_Start_Address[tno]        4            Uint32
   }
   Reserved                         until 1028      Uint8                     0

   for (tno=1; tno<=N_Tracks; tno++)
   {
       Track_Length[tno]               4            Uint32
   }
   Reserved                         until 2048      Uint8                     0
}
```

Figure 3-29 : Syntax of Track_List_1

The value of N_Tracks is encoded in Area_Data of Area_TOC_0, see chapter 3.2.1.2.

3.2.2.1 Track_List_1_Signature

Track_List_1_Signature is an 8 byte string identifying the Sector with Track_List_1. The value of Track_List_1_Signature must be “SACDTRL1“ ($53 $41 $43 $44 $54 $52 $4C $31).

3.2.2.2 Track_Start_Address

Track_Start_Address[tno] contains the Logical Sector Number (LSN) for the first Sector of Track[tno]. The first Sector of Track[tno] is the Sector that contains the first byte of the first Multiplexed Frame of that Track. The Track_Start_Address of a Track must be in the Track Area (see chapter 2.2.6.2).

Note: For 1 ≤ tno < N_Tracks the following equation must be true:
Track_Start_Address[tno+1] ≥ Track_Start_Address[tno] + Track_Length[tno] - 1

3.2.2.3 Track_Length

Track_Length[tno] contains the length in Sectors of Track[tno]. A Track_Length of zero is not allowed. The Last Sector of Track[tno] is the Sector that contains the last byte of the last Multiplexed Frame of that Track.
Track_Length can be calculated by using the following formula:

Track_Length[tno] = LSN(Last Sector[tno]) - Track_Start_Address[tno] + 1

Note that the sum of the Track_Length of all Tracks in a Track Area does not have to be equal to the size of that Track Area. The reasons are:

- One Sector can be both the last Sector of a Track and the first Sector of the next Track.
- One or more Tracks can have a preceding Pause.

3.2.3 Track_List_2

Track_List_2 contains the start Time Code, Playing Time and Track Flags for all Tracks in the current Track Area. Track_List_2 must always be present in an Area TOC. The length of Track_List_2 is always one Sector. The syntax of Track_List_2 is defined in Figure 3-30.

	# bytes	format	value
Track_List_2(){			
Track_List_2_Signature	8	String	"SACDTRL2"
for (tno=1; tno<=N_Tracks; tno++)			
{			
Track_Start_Time_Code[tno]	3	Time_Code	
Track_Mode[tno]	1	Track_Mode	
}			
Reserved	until 1028	Uint8	0
for (tno=1; tno<=N_Tracks; tno++)			
{			
Track_Time_Length[tno]	3	Time_Code	
Track_Flags[tno]	1	Track_Flags	
}			
Reserved	until 2048	Uint8	0
}			

Figure 3-30 : Syntax of Track_List_2

The value of N_Tracks is encoded in Area_Data of Area_TOC_0, see chapter 3.2.1.2.

3.2.3.1 Track_List_2_Signature

Track_List_2_Signature is an 8 byte string identifying the Sector with Track_List_2. The value of Track_List_2_Signature must be "SACDTRL2" ($53 $41 $43 $44 $54 $52 $4C $32).

3.2.3.2 Track_Start_Time_Code

Track_Start_Time_Code[tno] gives the start Time Code of Track[tno]. The definition of Time_Code is given in chapter 1.7.2.9. Track_Start_Time_Code[tno] must be encoded in the Audio_Header of the Sector addressed by Track_Start_Address[tno].

Note : For 1 ≤ tno < N_Tracks the following equation must be true:
Track_Start_Time_Code[tno+1] ≥ Track_Start_Time_Code[tno] + Track_Time_Length[tno]

3.2.3.3 Track_Mode

The format of Track_Mode[tno] is defined in Figure 3-31.

b7	b6	b5	b4	b3	b2	b1	b0
Extra_Use			Reserved				

Figure 3-31 : Format of Track_Mode

3.2.3.3.1 Extra_Use

If N_Channels = 6, Extra_Use[tno] defines the usage of Audio Channel 4. If N_Channels = 2 or N_Channels = 5, the value of Extra_Use[tno] must be zero. If N_Channels = 6, the definition of Extra_Use[tno] is given in Figure 3-32.

Value	Meaning
0	Audio Channel 4 is used for a LFE loudspeaker
1..7	Reserved for future use

Figure 3-32 : Definition of Extra_Use

3.2.3.4 Track_Time_Length

Track_Time_Length[tno] contains the playing time of Track[tno]. The definition of Time_Code is given in chapter 1.7.2.9. The minimum allowed value for Track_Time_Length is 1 second.

3.2.3.5 Track_Flags

The format of Track_Flags[tno] is defined in Figure 3-33.

b7	b6	b5	b4	b3	b2	b1	b0
ILP[tno]	TMF4[tno]	TMF3[tno]	TMF2[tno]	TMF1[tno]	Reserved		

Figure 3-33 : Format of Track_Flags

3.2.3.5.1 Index List Present (ILP)

ILP[tno] indicates whether for Track[tno] an Index_List is present or not. If Index_Ptr[tno] (see chapter 3.2.7.2) is not equal to zero, ILP[tno] must be set to one. If Index_List is not present or Index_Ptr[tno] is equal to zero, ILP[tno] must be set to zero.

3.2.3.5.2 TMF1, TMF2, TMF3, TMF4

The Track mute flags TMF1[tno] .. TMF4[tno] indicate which Audio Channels are not available for Track[tno].
If an Audio Channel is not available for a Track, this Audio Channel must contain a Silence Pattern during the whole Track. In case an Audio Channel is not available in a Track, it is not required to have a Silence Pattern at the start of that Track only if the same Audio Channel is available for the previous Track. In case an Audio Channel is not available in a Track, it is not required to have a Silence Pattern at the end of that Track only if the same Audio Channel is available for the next Track.
If contents is available for an Audio Channel, the audio contents of this Audio Channel is not restricted.
If N_Channels is equal to 2, TMF1[tno], TMF2[tno], TMF3[tno] and TMF4[tno] all must be set to zero.
TMF1[tno] set to one indicates that audio contents is not available in Audio Channels 1 and 2.
TMF2[tno] set to one indicates that audio contents is not available in Audio Channels 4 and 5 (if N_Channels = 5) or in Audio Channels 5 and 6 (if N_Channels = 6).
If N_Channels is equal to 5 or 6, TMF3[tno] set to one indicates that audio contents is not available in Audio Channel 3.
If N_Channels is equal to 6, TMF4[tno] set to one indicates that audio contents is not available in Audio Channel 4. If N_Channels is equal to 5, TMF4[tno] must be set to zero.
Figure 3-34 gives an overview of the relation between N_Channels and TMF1, TMF2, TMF3, TMF4.
Figure 3-35 gives an overview of the allowed combinations of TMF1, TMF2, TMF3 and TMF4.
For the relation between Audio Channel Number and the loudspeaker position, see the Channel Mappings defined in chapter 3.2.1.2.4.

Field		N_Channels = 6	N_Channels = 5	N_Channels = 2
name	value			
TMF1	0	Channel 1 and Channel 2 are available	Channel 1 and Channel 2 are available	Channel 1 and Channel 2 are available
	1	Value not allowed	Value not allowed	Value not allowed
TMF3	0	Channel 3 is available	Channel 3 is available	Not used
	1	Channel 3 contains a Silence Pattern	Channel 3 contains a Silence Pattern	Value not allowed
TMF4	0	Channel 4 is available	Not used	Not used
	1	Channel 4 contains a Silence Pattern	Value not allowed	Value not allowed
TMF2	0	Channel 5 and Channel 6 are available	Channel 4 and Channel 5 are available	Not used
	1	Channel 5 and Channel 6 contain a Silence Pattern	Channel 4 and Channel 5 contain a Silence Pattern	Value not allowed

Figure 3-34 : Relation between N_Channels and TMF1, TMF2, TMF3 and TMF4

For the Multichannel Area the following restrictions shall apply:
- Channel 1 and Channel 2 are always available (TMF1 is always equal to zero)
- Minimally three Audio Channels are always available
- If N_Channels is equal to 6 and TMF4[tno] is equal to zero, minimally four Audio Channels are always available.

Note: It is recommended that the jacket of the disc contains information on the actual Audio Channels used if minimally one of TMF2[tno], TMF3[tno] or TMF4[tno] is equal to 1.

N_Channels	Available channels in Track [tno]	TMFx[tno]			
		4	3	2	1
6	Channel 1, 2, 3, 4, 5, 6	0	0	0	0
	Channel 1, 2, 3, 5, 6	1	0	0	0
	Channel 1, 2, 4, 5, 6	0	1	0	0
	Channel 1, 2, 5, 6	1	1	0	0
	Channel 1, 2, 3, 4	0	0	1	0
	Channel 1, 2, 3	1	0	1	0
5	Channel 1, 2, 3, 4, 5	0	0	0	0
	Channel 1, 2, 4, 5	0	1	0	0
	Channel 1, 2, 3	0	0	1	0
2	Channel 1, 2	0	0	0	0

Figure 3-35 : Allowed combinations of TMF1, TMF2, TMF3 and TMF4

3.2.4 ISRC_and_Genre_List

ISRC_and_Genre_List must always be present in an Area TOC. The length of ISRC_and_Genre_List is always two Sectors. The syntax of ISRC_and_Genre_List is defined in Figure 3-36.

```
                                          # bytes          format          value
ISRC_and_Genre_List(){
   ISRC_and_Genre_Signature                  8             String       "SACD_IGL"
   for (tno=1; tno<=N_Tracks; tno++)
   {
       ISRC()[tno]                          12             ISRC
   }
   Reserved                               until 3072       Uint8            0
   for (tno=1; tno<=N_Tracks; tno++)
   {
       Genre[tno]                           4              Genre_Code
   }
   Reserved                               until 4096       Uint8            0
}
```

Figure 3-36 : Syntax of ISRC_and_Genre_List

The value of N_Tracks is encoded in Area_Data of Area_TOC_0, see chapter 3.2.1.2.

3.2.4.1 ISRC_and_Genre_Signature

ISRC_and_Genre_Signature is an 8 byte string identifying the first Sector of ISRC_and_Genre_List. The value of ISRC_and_Genre_Signature must be “SACD_IGL“ ($53 $41 $43 $44 $5F $49 $47 $4C).

3.2.4.2 ISRC

It is recommended that ISRC[tno] contains the ISRC code for Track[tno]. If the ISRC code is not available for Track[tno], all fields in ISRC[tno] must be set to NUL characters ($00). If the ISRC code is used for a Track, it must be used according to ISO 3901. The syntax of ISRC is defined in Figure 3-37.

```
                                    #bytes        format              value
ISRC()[tno]{
   Country_Code[tno]                  2           String
   Owner_Code[tno]                    3           String
   Recording_Year[tno]                2           String
   Designation_Code[tno]              5           String
}
```

Figure 3-37 : Syntax of ISRC

3.2.4.2.1 Country_Code

Country_Code[tno] must contain the ISO 646 coded Country code for Track[tno], see ISO 3901 section 4.1.

3.2.4.2.2 Owner_Code

Owner_Code[tno] must contain the ISO 646 coded First owner code for Track[tno], see ISO 3901 section 4.2.

3.2.4.2.3 Recording_Year

Recording_Year[tno] must contain the Year-of-recording code for Track[tno], see ISO 3901 section 4.3. Each digit is coded as one ISO 646 character.

3.2.4.2.4 Designation_Code

Designation_Code[tno] must contain the concatenation of Recording code and Recording-item code for Track[tno], see ISO 3901 sections 4.4 and 4.5. Each digit is coded as one ISO 646 character.

3.2.4.3 Genre

It is recommended that Genre[tno] contains the Genre Code for Track[tno]. If the Genre Code is not available for Track[tno], all fields in Genre[tno] must be set to zero. If the Genre Code is used for a Track, it must be used according to the definitions in chapter 1.7.2.2.

3.2.5 Access_List

Access_List contains a table with Start Addresses associated with Time Codes for a DST coded Track Area. Access_List must be present if the Audio Area is DST coded. If the current Audio Area is Plain DSD, the Access_List is not allowed to be present. The Start Address for a Time Code T, is the Logical Sector Number of the Sector that contains the first byte of Multiplexed Frame T. Access_List has a fixed size of 65536 Bytes (32 Sectors). The syntax of Access_List is defined in Figure 3-38.

	# bytes	format	value
Access_List(){			
Access_List_Signature	8	String	"SACD_ACC"
N_Entries	2	Uint16	1..6550
Main_Step_Size	1	Uint8	
Reserved	5	Uint8	0
Main_Acc_List()		Main_Acc_List	
Reserved	until 32768	Uint8	0
Detailed_Access()		Detailed_Access	
Reserved	until 65536	Uint8	0
}			

Figure 3-38 : Syntax of Access_List

3.2.5.1 Access_List_Signature

Access_List_Signature is an 8 byte string identifying the first Sector of Access_List. The value of Access_List_Signature must be “SACD_ACC“ ($53 $41 $43 $44 $5F $41 $43 $43).

3.2.5.2 N_Entries

N_Entries contains the number of Entries in Main_Acc_List. The relation between N_Entries, Total_Area_Play_Time (see chapter 3.2.1.2) and Main_Step_Size is defined in the following formula:

$$N\_Entries = 1 + \mathrm{trunc}\left(\frac{Total\_Area\_Play\_Time - 1}{Main\_Step\_Size} \right)$$

Where in this formula Total_Area_Play_Time must be expressed in Frames. The maximum allowed value of N_Entries is 6550.

3.2.5.3 Main_Step_Size

The Track_Area is divided into Intervals of Main_Step_Size Frames, with Interval[N] being the (N+1)th Interval in the Track Area. For every Main_Step_Size Frames, the Start Address of the Multiplexed Frame is encoded in Main_Acc_List. Main_Step_Size must be a multiple of 10. The minimum allowed value of Main_Step_Size is 10. The maximum allowed value of Main_Step_Size is 180.

3.2.5.4 Main_Acc_List

Main_Acc_List contains the Start Addresses of the Multiplexed Frames at intervals of Main_Step_Size Frames. For each encoded Start Address, Main_Acc_List contains either a correction factor to estimate the intermediate Start Addresses, or a pointer to a detailed access List (Detailed_List). The syntax of Main_Acc_List is defined in Figure 3-39.

```
                                    # bytes        format                  value
Main_Acc_List(){
   for (N=0; N<N_Entries; N++)
   {
       Access_Flags()[N]              2           Access_Flags
       Entry[N]                       3           Uint24
   }
}
```

Figure 3-39 : Syntax of Main_Acc_List

3.2.5.4.1 Access_Flags

Access_Flags[N] contains for Interval[N] either a pointer to a detailed Access List (Detailed_List), or the Access_Margin that is needed to estimate the Start Address for a given Time Code. The syntax of Access_Flags is defined in Figure 3-40.

```
                                    # bits         format                  value
Access_Flags()[N] {
   Detailed[N]                       1           Uint1
   if (Detailed[N] == 1)
   {
       Detailed_List_Ptr[N]         15           Uint15
   }
   else
   {
       Access_Margin[N]             15           Uint15
   }
}
```

Figure 3-40 : Syntax of Access_Flags

Note that the allocation of the fields in Access_Flags is defined in bits and not in bytes.

3.2.5.4.1.1 Detailed

If Detailed[N] is set to one, for Interval[N] a Detailed_List is encoded in Detailed_Access. If a Detailed_List is encoded, the Start Address for a Time Code in Interval[N] can be estimated by using Detailed_List[N]. If Detailed[N] is set to zero, in Interval[N] the Start Address is estimated by linear interpolation modified with Access_Margin[N]. For the last Interval in a Track Area, Detailed[N] must be set to zero.
Note: Main_Acc_List can contain maximum 1213 Access_Flags with Detailed[N] set to one.

3.2.5.4.1.2 Detailed_List_Ptr

Detailed_List_Ptr[N] contains the byte position of Detailed_List[N] within Detailed_Access. The first Detailed_List_Ptr in Main_Acc_List must have the value zero.

3.2.5.4.1.3 Access_Margin

Access_Margin[N] contains a correction factor for the calculation of the estimated Start Address between Entry[N] and Entry[N+1]. The estimated Start Address for a given Time Code T, can be calculated with the following formulas:

$$\text{Estimated Start Address}[T] = \max(\text{Entry}[N], \text{Interp\_Address}[T] - \text{Access\_Margin}[N])$$

where: $N = \text{trunc}\left(\frac{T}{\text{Main\_Step\_Size}}\right)$

T = The Time Code expressed in Frames.

$$\text{Interp\_Address}[T] = \text{Entry}[N] + \text{trunc}\left(\frac{(T \bmod \text{Main\_Step\_Size}) \times (\text{Entry}[N+1] - \text{Entry}[N])}{\text{Main\_Step\_Size}}\right)$$

Access_Margin[N] must contain the largest value of (Interp_Address[T] - Start_Address[T]) for all frames in the Interval[N]. Where Start Address[T] is the Start Address of Multiplexed Frame T. For the last Interval of a Track Area, a different calculation method for Interp_Address[T] is used:

$$\text{Interp\_Address}[T] = \text{Entry}[N_{last}] + \text{trunc}\left(\frac{(T \bmod \text{Main\_Step\_Size}) \times (\text{Track\_Area\_End\_Address} - \text{Entry}[N_{last}])}{1 + ((\text{Total\_Area\_Play\_Time} - 1) \bmod \text{Main\_Step\_Size})}\right)$$

where: Total_Area_Play_Time is the total playing time of the Track Area expressed in Frames.

3.2.5.4.2 Entry

Entry[N] contains the Start Address of Multiplexed Frame[N * Main_Step_Size].

3.2.5.5 Detailed_Access

Detailed_Access contains one Detailed_List for each Entry[N] in Main_Acc_List with Detailed[N] set to one. If a Detailed_List is encoded for Interval[N], the Start Address can be retrieved from Detailed_List[N] for 9 Time Codes in Interval[N]. The syntax of Detailed_Access is defined in Figure 3-41.

```
                                    # bytes        format                  value
Detailed_Access(){
   for (N=0; N<N_Entries; N++)
   {
       if (Detailed[N] == 1)
       {
           Detailed_List()[N]          27          Detailed_List
       }
   }
}
```

Figure 3-41 : Syntax of Detailed_Access

Note: Detailed_Access can contain maximum 1213 Detailed_List.

3.2.5.5.1 Detailed_List

Detailed_List[N] contains the Start Address for 9 Frames between Entry[N] and Entry[N+1]. The syntax of Detailed_List is defined in Figure 3-42.

```
                                        # bytes      format                    value
Detailed_List()[N] {
   for (M=1; M<10; M++)
   {
       Sub_Entry[N][M]                     3         Uint24
   }
}
```

Figure 3-42 : Syntax of Detailed_List

3.2.5.5.1.1 Sub_Entry

Sub_Entry[N][M] contains the Start Address of Multiplexed Frame[N*Main_Step_Size+ M*Main_Step_Size/10]. Sub_Entry[N][0] is equal to Entry[N] and is not encoded in Detailed_List[N].

3.2.6 Track_Text

Track_Text is only present if Track_Text_Ptr (see chapter 3.2.1.3) is not set to zero. Track_Text contains text associated with the Tracks in the current Track Area. The length of Track_Text is an integer number of Sectors. The maximum length of Track_Text is 32 Sectors. The syntax of Track_Text is defined in Figure 3-43.

```
                                                    # bytes     format     value
Track_Text(){
   Track_Text_Signature                                8        String  "SACDTTxt"
   for (c=1; c<=N_Text_Channels; c++)
   {
       for (tno=1; tno<=N_Tracks; tno++)
       {
           Track_Text_Item_Ptr[c][tno]                 2        Uint16
       }
   }
   Reserved                                          until 4096   Uint8        0

   for (c=1; c<=N_Text_Channels; c++)
   {
       for (tno=1; tno<=N_Tracks; tno++)
       {
           if (Track_Text_Item_Ptr[c][tno] != 0)
           {
               N_Items[c][tno]                         1        Uint8      1..10
               Reserved                                3        Uint8        0
               for (item=1; item<=N_Items[c][tno]; item++)
               {
                   Text_Item[c][tno][item]                      TOC_Text
               }
           }
       }
       Reserved                                       until eos   Uint8        0
   }
}
```

Figure 3-43 : Syntax of Track_Text

The values of N_Text_Channels and N_Tracks are encoded in Text_Channels of Area_Data, see chapter 3.2.1.2.15.
If Track_Text is present, it must contain minimally one Text_Item.

3.2.6.1 Track_Text_Signature

Track_Text_Signature is an 8 byte string identifying the first Sector of Track_Text. The value of Track_Text_Signature must be “SACDTTxt“ ($53 $41 $43 $44 $54 $54 $78 $74).

3.2.6.2 Track_Text_Item_Ptr

Track_Text_Item_Ptr[c][tno] contains the byte offset from the first byte of Track_Text to N_Items[c][tno]. If no Text_Item is available for the combination of Text Channel c and Track[tno], Track_Text_Item_Ptr[c][tno] must be set to zero. If minimally one Text_Item is available, the first non zero Track_Text_Item_Ptr in Track_Text must have the value 4096.

3.2.6.3 N_Items

N_Items[c][tno] contains the number of text items available for Track[tno] in Text Channel c. The minimum allowed value for N_Items is one. The maximum allowed value for N_Items is 10.

3.2.6.4 Text_Item

Text_Item[c][tno][item] contains a text item for Track[tno] in Text Channel c. The Character Set and Language Code for Text Channel c is defined in Text_Channels of Area_Data, see chapter 3.2.1.2.15. The definition of TOC_Text is given in chapter 1.7.2.10. Text_Items must be stored in increasing order of Text_Type, see chapter 1.7.2.10.1.

3.2.7 Index_List

Index_List is only allowed to be present if the Track Area contains minimum one Track with two or more Indexes. For all Tracks with Indexes above the value of one, Index_List contains the start Time Code of Indexes 2 and higher. The maximum length of Index_List is 10 Sectors. The syntax of Index_List is defined in Figure 3-44.

```
                                                  # bytes       format        value
Index_List(){
   Index_List_Signature                              8          String     "SACD_Ind"
   for (tno=1; tno<=N_Tracks; tno++)
   {
       Index_Ptr[tno]                                2          Uint16
   }
   Reserved                                       until 524     Uint8           0
   for (tno=1; tno<=N_Tracks; tno++)
   {
       if (Index_Ptr[tno] > 0)
       {
           N_Stored_Indexes[tno]                     1          Uint8        1..254
           for (x=1; x<=N_Stored_Indexes[tno]; x++)
           {
               Index_Start_TC[tno][x+1]              3          Time_Code
           }
       }
   }
   Reserved                                       until eos     Uint8           0
}
```

Figure 3-44 : Syntax of Index_List

The value of N_Tracks is encoded in Area_Data of Area_TOC_0, see chapter 3.2.1.2.
If Index_List is present, it must contain minimally one Index_Start_TC.

3.2.7.1 Index_List_Signature

Index_List_Signature is an 8 byte string identifying the first Sector of Index_List. The value of Index_List_Signature must be “SACD_Ind“ ($53 $41 $43 $44 $5F $49 $6E $64).

3.2.7.2 Index_Ptr

Index_Ptr[tno] contains the byte offset from the first byte of Index_List to N_Stored_Indexes[tno]. If no Index_Start_TC[tno][ ] are stored for Track[tno], the value of Index_Ptr[tno] must be zero. If Index_List is present, the first non zero Index_Ptr in Index_List must have the value 524.

3.2.7.3 N_Stored_Indexes

N_Stored_Indexes[tno] contains the number of Indexes that is stored in the Index_List for Track[tno]. The minimum allowed value for N_Stored_Indexes is one. The maximum allowed value for N_Stored_Indexes is 254.
Note: N_Stored_Indexes is equal to the number of Indexes in a Track minus one.

3.2.7.4 Index_Start_TC

Index_Start_TC[tno][x+1] contains the start Time_Code for Index x+1 in Track[tno]. The start Time_Code is relative to the start of the Track Area, see chapter 2.2.6.2. The start position of Index[1] is equal to the start position of Track[tno], and is not stored in the Index_List. The first stored Index_Start_TC points to the start of Index 2. The encoding of Time_Code is defined in chapter 1.7.2.9.

3.2.8 Track_WebLink_List

Track_WebLink_List is only allowed to be present if the Track Area contains minimum one Track with a WebLink. The maximum length of Track_WebLink_List is 4 Sectors. The syntax of Track_WebLink_List is defined in Figure 3-45.

```
                                              # bytes       format          value
Track_WebLink_List(){
   Track_WebLink_List_Signature                  8          String       "SACD_WLL"
   for (tno=1; tno<=N_Tracks; tno++)
   {
       Track_WL_Start_Ptr[tno]                   2          Uint16
   }
   Reserved                                  until 524      Uint8            0
   for (tno=1; tno<=N_Tracks; tno++)
   {
       if (Track_WL_Start_Ptr[tno] != 0)
       {
           Track_WebLink[tno]                             WebLink_String
       }
   }
   Reserved                                  until eos      Uint8            0
}
```

Figure 3-45 : Syntax of Track_WebLink_List

The value of N_Tracks is encoded in Area_Data of Area_TOC_0, see chapter 3.2.1.2.

3.2.8.1 Track_WebLink_List_Signature

Track_WebLink_List_Signature is an 8 byte string identifying the first Sector of Track_WebLink_List. The value of Track_WebLink_List_Signature must be “SACD_WLL“ ($53 $41 $43 $44 $5F $57 $4C $4C).

3.2.8.2 Track_WL_Start_Ptr

Track_WL_Start_Ptr[tno] contains the byte offset from the first byte of Track_WebLink_List to Track_WebLink[tno]. If Track_WebLink_List is present, the first non zero Track_WL_Start_Ptr in Track_WebLink_List must have the value 524. If Track_WL_Start_Ptr[tno] is equal to zero, Track_WebLink[tno] is not present.

3.2.8.3 Track_WebLink

Track_WebLink[tno] contains a WebLink_String that points to a web page containing information about Track[tno]. If Disc_WebLink (see chapter 3.1.1.5.1) can not be used for Track[tno], Track_WebLink can contain a WebLink_String for Track[tno]. If Track_WebLink[tno] is not present, Disc_WebLink can be used for Track[tno].

3.2.9 Track_List_3

Track_List_3 contains information about the contents of the Supplementary Data Stream in the current Audio Area, in addition Track_List_3 contains CCI information. If, in the current Audio Area, the Supplementary Data Stream contains SD_Units as specified in chapter 5.7.1.1, Track_List_3 must be present in Area TOC-B. Track_List_3 shall not be present in Area TOC-A. Note that Track_List_3 can be present even if Supplementary Data is not present. The length of Track_List_3 is 8 Sectors. The syntax of Track_List_3 is defined in Figure 3-46.

```
                                          # bytes       format            value
Track_List_3(){
   Track_List_3_Signature                    8          String          "SACDTRL3"
   SD_Info()                              4088          SD_Info
   Audio_CCI()                            4096          Audio_CCI
   SD_Slide_List()                        8192          SD_Slide_List
   Reserved                           until 4*4096      Uint8               0
}
```

Figure 3-46 : Syntax of Track_List_3

The value of N_Tracks is encoded in Area_Data of Area_TOC_0, see chapter 3.2.1.2.

3.2.9.1 Track_List_3_Signature

Track_List_3_Signature is an 8 byte string identifying the first Sector of Track_List_3. The value of Track_List_3_Signature must be “SACDTRL3“ ($53 $41 $43 $44 $54 $52 $4C $33).

3.2.9.2 SD_Info

SD_Info contains information about the Supplementary Data in the current Audio Area. The syntax of SD_Info is defined in Figure 3-47.

```
                                          # bytes       format            value
SD_Info(){
   Area_Flags_2                              1          Area_Flags_2
   Reserved                                  7          Reserved            0
   for (tno=1; tno<=N_Tracks; tno++)
   {
       PreLoad_Entry[tno]                    7          PreLoad_Entry

       Track_Pic_Presence[tno]               1          Track_Pic_Presence
       for (c=1; c<=N_Text_Channels; c++)
       {
           Track_Txt_Sub_Presence[tno][c]    1          Track_Txt_Sub_Presence
       }
       Reserved                      8-N_Text_Channels  Uint8               0
   }
   Reserved                           4080-16*N_Tracks  Uint8               0
}
```

Figure 3-47 : Syntax of SD_Info

3.2.9.2.1 Area_Flags_2

The format of Area_Flags_2 is defined in Figure 3-48.

b7	b6	b5	b4	b3	b2	b1	b0
Pic_Pr	Sub_Pr	Txt_Pr	Reserved				

Figure 3-48 : Format of Area_Flags_2

3.2.9.2.1.1 Pictures Present (Pic_Pr)

Pic_Pr indicates whether JPEG pictures are present or not in Supplementary Data of the current Audio Area, see chapter 5.7.1.1.1.2. If JPEG pictures are present in Supplementary Data of the current Audio Area, Pic_Pr must be set to one. If JPEG pictures are not present in Supplementary Data of the current Audio Area, Pic_Pr must be set to zero.

3.2.9.2.1.2 Sub_Pictures Present (Sub_Pr)

Sub_Pr indicates whether Sub_Pictures are present or not in Supplementary Data of the current Audio Area, see chapter 5.7.1.1.1.3. If Sub_Pictures are present in Supplementary Data of the current Audio Area, Sub_Pr must be set to one. If Sub_Pictures are not present in Supplementary Data of the current Audio Area, Sub_Pr must be set to zero.

3.2.9.2.1.3 Text Present (Txt_Pr)

Txt_Pr indicates whether encoded text is present or not in Supplementary Data of the current Audio Area, see chapter 5.7.1.1.1.4. If encoded text is present in Supplementary Data of the current Audio Area, Txt_Pr must be set to one. If encoded text is not present in Supplementary Data of the current Audio Area, Txt_Pr must be set to zero.

3.2.9.2.2 PreLoad_Entry

PreLoad_Entry[tno] contains the Time Code and the LSN at which the PreLoad Area (see chapter 5.7.1.1.4.1) of Track[tno] starts. The syntax of PreLoad_Entry is defined in Figure 3-49.

```
                                    # bytes        format                  value
PreLoad_Entry()[tno]{
   PreLoad_Start_Address[tno]          4           Uint32
   PreLoad_Start_TC[tno]               3           Time_Code
}
```

Figure 3-49 : Syntax of PreLoad_Entry

3.2.9.2.2.1 PreLoad_Start_Address

The Sector with address PreLoad_Start_Address[tno] must contain the first byte of the Frame with Time Code equal to PreLoad_Start_TC[tno].

3.2.9.2.2.2 PreLoad_Start_TC

PreLoad_Start_TC[tno] contains the Time Code at which the PreLoad Area of Track[tno] starts, see chapter 5.7.1.1.4.1. If the PreLoad Area of Track[tno] does not exist, PreLoad_Start_TC[tno] must be equal to Track_Start_Time_Code[tno], see chapter 3.2.3.2.
Notes:

- For (tno > 1), the PreLoad_Start_TC[tno] can be located in Track[tno-1] or in Pause[tno].
- For Track[1], the PreLoad_Start_TC[1] must be located in Pause[1]. The length of Pause[1] must be such that PreLoad_Start_TC[1] is not negative.

3.2.9.2.3 Track_Pic_Presence

Track_Pic_Presence[tno] indicates the presence of pictures for Track[tno]. The format of Track_Pic_Presence[tno] is defined in Figure 3-50.

b7	b6	b5	b4	b3	b2	b1	b0
Pic_SS[tno]	Pic_BA[tno]	Pic_BB[tno]	Pic_BC[tno]	Reserved			Resol[tno]

Figure 3-50 : Format of Track_Pic_Presence

3.2.9.2.3.1 Slide Show Picture Present (Pic_SS)

Pic_SS[tno] indicates whether Slide Show pictures are present or not for Track[tno]. If one or more Slide Show pictures with a Start_PTC (see chapter 5.7.1.1.4) in Track[tno] are present, Pic_SS[tno] must be set to one. If no Slide Show pictures with a Start_PTC in Track[tno] are present, Pic_SS[tno] must be set to zero.

3.2.9.2.3.2 Background Picture A Present (Pic_BA)

Pic_BA[tno] indicates whether Background_Picture_A is present or not for Track[tno]. If Background_Picture_A is present for Track[tno], Pic_BA[tno] must be set to one. If Background_Picture_A is not present for Track[tno], Pic_BA[tno] must be set to zero.

3.2.9.2.3.3 Background Picture B Present (Pic_BB)

Pic_BB[tno] indicates whether Background_Picture_B is present or not for Track[tno]. If Background_Picture_B is present for Track[tno], Pic_BB[tno] must be set to one. If Background_Picture_B is not present for Track[tno], Pic_BB[tno] must be set to zero.

3.2.9.2.3.4 Background Picture C Present (Pic_BC)

Pic_BC[tno] indicates whether Background_Picture_C is present or not for Track[tno]. If Background_Picture_C is present for Track[tno], Pic_BC[tno] must be set to one. If Background_Picture_C is not present for Track[tno], Pic_BC[tno] must be set to zero.

3.2.9.2.3.5 Resol

Resol[tno] indicates whether the JPEG_Pictures and Sub_Pictures in Track[tno] have a resolution of 1440x1080 or a resolution of 1920x1080. If the resolution of the JPEG_Pictures and Sub_Pictures in Track[tno] is 1920x1080 (hxv), Resol[tno] must be set to zero. If the resolution of the JPEG_Pictures and Sub_Pictures in Track[tno] is 1440x1080 (hxv), Resol[tno] must be set to one. If there are no JPEG_Pictures and no Sub_Pictures available in Track[tno], Resol[tno] must be set to zero.

3.2.9.2.4 Track_Txt_Sub_Presence

Track_Txt_Sub_Presence[tno][c] indicates the presence of Sub Pictures and Text in Supplemantary Data for Track[tno] with Text_Channel[c]. The format of Track_Txt_Sub_Presence[tno][c] is defined in Figure 3-51.

b7	b6	b5	b4	b3	b2	b1	b0
Sub_St[tno][c]	Sub_Ti[tno][c]	Sub_Ly[tno][c]	Sub_Li[tno][c]	Txt_St[tno][c]	Txt_Ti[tno][c]	Txt_Ly[tno][c]	Txt_Li[tno][c]

Figure 3-51 : Format of Track_Txt_Sub_Presence

3.2.9.2.4.1 Streaming_Info SubPicture Present (Sub_St)

Sub_St[tno][c] indicates whether Streaming_Info SubPictures are present or not for Track[tno] with Text Channel[c]. If one or more Streaming_Info SubPictures for Text_Channel[c] with a Start_PTC (see chapter 5.7.1.1.4) in Track[tno] are present, Sub_St[tno][c] must be set to one. If no Streaming_Info SubPictures for Text_Channel[c] with a Start_PTC in Track[tno] are present for Track[tno][c], Sub_St[tno][c] must be set to zero.

3.2.9.2.4.2 Title SubPicture Present (Sub_Ti)

Sub_Ti[tno][c] indicates whether any Title SubPicture is present or not for Track[tno] with Text Channel[c]. If Title SubPictures are present for Track[tno][c], Sub_Ti[tno][c] must be set to one. If Title SubPictures are not present for Track[tno][c], Sub_Ti[tno][c] must be set to zero.

3.2.9.2.4.3 Lyrics SubPicture Present (Sub_Ly)

Sub_Ly[tno][c] indicates whether Lyrics SubPictures are present or not for Track[tno] with Text Channel[c]. If Lyrics SubPictures are present for Track[tno][c], Sub_Ly[tno][c] must be set to one. If Lyrics SubPictures are not present for Track[tno][c], Sub_Ly[tno][c] must be set to zero.

3.2.9.2.4.4 Liner_Notes SubPicture Present (Sub_Li)

Sub_Li[tno][c] indicates whether Liner_Notes SubPictures are present or not for Track[tno] with Text Channel[c]. If Liner_Notes SubPictures are present for Track[tno][c], Sub_Li[tno][c] must be set to one. If Liner_Notes SubPictures are not present for Track[tno][c], Sub_Li[tno][c] must be set to zero.

3.2.9.2.4.5 Streaming_Info Text Present (Txt_St)

Txt_St[tno][c] indicates whether Streaming_Info Text is present or not for Track[tno] with Text Channel[c]. If Streaming_Info Text is present for Track[tno][c], Txt_St[tno][c] must be set to one. If Streaming_Info Text is not present for Track[tno][c], Txt_St[tno][c] must be set to zero.

3.2.9.2.4.6 Title Text Present (Txt_Ti)

Txt_Ti[tno][c] indicates whether Title Text is present or not for Track[tno] with Text Channel[c]. If Title Text is present for Track[tno][c], Txt_Ti[tno][c] must be set to one. If Title Text is not present for Track[tno][c], Txt_Ti[tno][c] must be set to zero.

3.2.9.2.4.7 Lyrics Text Present (Txt_Ly)

Txt_Ly[tno][c] indicates whether Lyrics Text is present or not for Track[tno] with Text Channel[c]. If Lyrics Text is present for Track[tno][c], Txt_Ly[tno][c] must be set to one. If Lyrics Text is not present for Track[tno][c], Txt_Ly[tno][c] must be set to zero.

3.2.9.2.4.8 Liner_Notes Text Present (Txt_Li)

Txt_Li[tno][c] indicates whether Liner_Notes Text is present or not for Track[tno] with Text Channel[c]. If Liner_Notes Text is present for Track[tno][c], Txt_Li[tno][c] must be set to one. If Liner_Notes Text is not present for Track[tno][c], Txt_Li[tno][c] must be set to zero.

3.2.9.3 Audio_CCI

Audio_CCI contains Copy Control Information for all Tracks in the current Audio Area. For the definition of Audio_CCI, see Super Audio CD Part 3-General.

3.2.9.4 SD_Slide_List

SD_Slide_List contains the addresses of all sectors with the first byte of a Multiplexed Frame that contains the SD_Unit (see chapter 5.7.1.1) with the start of a Slide_Show_Picture (see chapter 5.7.1.1.7.2). The syntax of SD_Slide_List is defined in Figure 3-52. SD_Slide_List must be present if minimally one Slide_Show_Picture is present in Supplementary Data of the current Audio Area.

```
                                          # bytes       format            value
SD_Slide_List(){
   N_SD_Slides                               2          Uint16           0..2047
   Reserved                                  2          Uint8               0
   for (n=1; n<=N_SD_Slides; n++)
   {
       SD_Slide_Address[n]                   4          Uint32
   }
   Reserved                               until 8192    Uint8               0
}
```

Figure 3-52 : SD_Slide_List

3.2.9.4.1 N_SD_Slides

N_SD_Slides contains the number of Slide_Show_Pictures in Supplementary Data of the current Audio Area. A value of zero for N_SD_Slides is allowed. The maximum allowed value of N_SD_Slides is 2047.

3.2.9.4.2 SD_Slide_Address

SD_Slide_Address[n] contains the address of a sector with the first byte of a Multiplexed Frame that contains the SD_Unit (see chapter 5.7.1.1) with the start of a Slide_Show_Picture[n] (see chapter 5.7.1.1.7.2) encoded in Supplementary Data of the current Audio Area. For 1 ≤ n ≤ N_SD_Slides-1 the following must be true:
SD_Slide_Address[n] ≤ SD_Slide_Address[n+1]

3.2.10 Set_Of_PlayLists

If Set_Of_PlayLists is present it contains one or more PlayLists. Each PlayList gives an alternative playback order and/or a different selection for the Audio Tracks on the Super Audio CD disc. Set_Of_PlayLists in the 2 Channel Stereo Area can only contain Tracks from the 2 Channel Stereo Area. Set_Of_PlayLists in the Multi Channel Area can only contain Tracks from the Multi Channel Area.
The presence of the Set_Of_PlayLists in the Area TOC is optional. Set_Of_PlayLists shall not be present in Area TOC-A. The maximum length of Set_Of_PlayLists is 5 Sectors. The syntax of Set_Of_PlayLists is defined in Figure 3-53.

```
                                          # bytes          format          value
Set_Of_PlayLists(){
   Set_Of_PlayLists_Signature                8             String       "SACDPLAY"
   N_PlayLists                               1             Uint8           1..16
   Reserved                                  7             Uint8             0
   for (c=1; c<=N_Text_Channels; c++)
   {
       for (p=1; p<=N_PlayLists; p++)
       {
           PlayList_Name_Ptr[c][p]           2             Uint16
       }
   }
   Reserved                                until 272       Uint8             0

   for (c=1; c<=N_Text_Channels; c++)
   {
       for (p=1; p<=N_PlayLists; p++)
       {
           PlayList_Name[c][p]                             Special_String
       }
   }
   Reserved                                until 2048      Uint8             0

   for (p=1; p<=N_PlayLists; p++)
   {
       PlayList()[p]                          512          PlayList
   }
   Reserved                                 until eos       Uint8             0
}
```

Figure 3-53 : Syntax of Set_Of_PlayLists

The value of N_Text_Channels is encoded in Text_Channels of Area_Data, see chapter 3.2.1.2.

3.2.10.1 Set_Of_PlayLists_Signature

Set_Of_PlayLists_Signature is an 8 byte string identifying the first Sector of Set_Of_PlayLists. The value of Set_Of_PlayLists_Signature must be “SACDPLAY“ ($53 $41 $43 $44 $50 $4C $41 $59).

3.2.10.2 N_PlayLists

N_PlayLists contains the number of PlayLists encoded in Set_Of_PlayLists. If Set_Of_PlayLists exists, the minumum value of N_PlayLists is equal to one. The maximum value of N_PlayLists is 16.

3.2.10.3 PlayList_Name_Ptr

PlayList_Name_Ptr[c][p] contains the byte position in Set_Of_PlayLists for the first byte of PlayList_Name[c][p]. If PlayList_Name[c][p] is not used, PlayList_Name[c][p] is not present and PlayList_Name_Ptr[c][p] must be set to zero.

3.2.10.4 PlayList_Name

PlayList_Name[c][p] contains the identification of PlayList[p] encoded for Text_Channel c. It is recommended that PlayList_Name[c][p] is present if PlayList[p] exists.

3.2.10.5 PlayList

The syntax of PlayList[p] is defined in Figure 3-54.

```
                                          # bytes          format          value
PlayList()[p]{
   PL_N_Entries[p]                           1             Uint8          1..255
   Reserved                                  1             Uint8             0
   for (t=1; t<=PL_N_Entries[p]; t++)
   {
       PL_Track_Value[p][t]                  1             Uint8
       PL_Index_Value[p][t]                  1             Uint8
   }
   Reserved                      510-2*PL_N_Entries[p]     Uint8             0
}
```

Figure 3-54 : Syntax of PlayList

The value of t defines the playback order for PlayList[p].

3.2.10.5.1 PL_N_Entries

PL_N_Entries[p] contains the value of the number of entries in PlayList[p]. The maximum value of PL_N_Entries[p] is equal to 255. The value of zero is not allowed for PL_N_Entries[p].

3.2.10.5.2 PL_Track_Value, PL_Index_Value

The combination of PL_Track_Value[p][t] and PL_Index_Value[p][t] is called PlayList_Entry[p][t]. PlayList_Entry[p][t] contains Track/Index combination t of PlayList[p].
The minimum value of PL_Track_Value[p][t] is one. The maximum value of PL_Track_Value[p][t] is equal to the value of N_Tracks in the current Audio Area. The maximum value of PL_Index_Value[p][t] is equal to the value of N_Stored_Indexes[PL_Track_Value[p][t]]+1 in the current Audio Area. A value of zero for PL_Index_Value[p][t] means that the whole Track[PL_Track_Value[p][t]] must be played. A non-zero value of PL_Index_Value[p][t] means that only Index[PL_Index_Value[p][t]] of Track[PL_Track_Value[p][t]] must be played.
It is recommended to play seamless between two successive instances of PlayList_Entry in a PlayList, if these two instances of PlayList_Entry refer to two adjacent Track/Index combinations, this includes situations where the second Track/Index combination is preceded by a Pause.

3.3 DTCP

The DTCP Area contains the DTCP_SRM_List. The first Sector of the DTCP Area is Sector 400. The length of the DTCP Area is 48 Sectors.

3.3.1 DTCP_SRM_List

DTCP_SRM_List contains information for full authentication as defined in the DTCP specification. The first byte of the DTCP_SRM_List is the first byte of Sector 400. If the DTCP_SRM_List is not present, all bytes of all Sectors in the DTCP Area must be set to zero.
The syntax of DTCP_SRM_List is defined in Figure 3-55.

```
                                        # bytes              format          value
DTCP_SRM_List(){
   DTCP_SRM0_List_Signature                8                 String       "SACDSRM0"
   DTCP_SRM[0]                                               SRM0
   Reserved                          Until 48*2048           Uint8            0
}
```

Figure 3-55 : Syntax of DTCP_SRM_List

3.3.1.1 DTCP_SRM0_List_Signature

DTCP_SRM0_List_Signature is an 8 byte string identifying the first Sector of DTCP_SRM0_List. If DTCP_SRM[0] exists, the value of DTCP_SRM0_List_Signature must be “SACDSRM0“ ($53 $41 $43 $44 $53 $52 $4D $30).

3.3.1.2 SRM0

The content of SRM0 is equal to the content of the SRM Type=0 as defined in the DTCP Specification.

3.4 EKB_Data

EKB_Data contains authorization data for SACD players. Two copies of EKB_Data are stored, the first copy in the EKB1 Area, the second copy in the EKB2 Area. The syntax of EKB_Data is defined in Figure 3-56.

```
                                        # bytes              format          value
EKB_Data(){
   EKB_Signature                           8                 String       "SACD_EKB"
   EKB_Length                              4                 Uint32
   Reserved                                4                 Uint8            0
   EKB                                                       EKB

   Reserved                          until 62*2048           Uint8            0
}
```

Figure 3-56 : Syntax of EKB_Data

Note: EKB_Data is not available on discs according to the Super Audio CD Specification Version 1.3 or lower.

3.4.1 EKB_Signature

EKB_Signature is an 8 byte string identifying the first Sector of EKB_Data. The value of EKB_Signature must be “SACD_EKB“ ($53 $41 $43 $44 $5F $45 $4B $42).

3.4.2 EKB_Length

EKB_Length contains the length of EKB in bytes.

3.4.3 EKB

The use of EKB is defined in Super Audio CD Part 3-Player.

3.5 Revocation_TOC

The Revocation_TOC is stored in the Rev TOC Area. The Revocation_TOC contains pointers to data to be used by Approved Secure Digital Interfaces, see Super Audio CD Part 3-General. The Revocation_TOC is stored in Sector 540. The Revocation_TOC has a fixed length of one Sector. The syntax of the Revocation_TOC is defined in Figure 3-57.

```
                                        # bytes          format          value
Revocation_TOC(){
   Rev_TOC_Signature                       8              String       "SACDRTOC"

   N_Revocation_Systems                    1              Uint8          0..169
   Reserved                                7              Uint8            0
   for (n=1; n<=N_Revocation_Systems; n++)
   {
       Revocation_ID[n]                    2              Uint16
       Rev_Data_Length[n]                  2              Uint16
       Rev_Data_Start_Address[n]           4              Uint32
       Rev_Attribute[n]                    4              Uint32
   }
   Reserved                             until 2048        Uint8            0
}
```

Figure 3-57 : Syntax of Revocation_TOC

Note: The Revocation_TOC is not available in discs according to the Super Audio CD Specification Version 1.3 or lower.

3.5.1.1 Rev_TOC_Signature

Rev_TOC_Signature is an 8 byte string identifying the Revocation_TOC. The value of Rev_TOC_Signature must be "SACDRTOC" ($53 $41 $43 $44 $52 $54 $4F $43).

3.5.1.2 N_Revocation_Systems

N_Revocation_Systems contains the number of Revocation Systems used in the Revocation_TOC. A value of zero is allowed for N_Revocation_Systems. The maximum allowed value of N_Revocation_Systems is 169.

3.5.1.3 Revocation_ID

Revocation_ID[n] identifies the Revocation System using the Revocation Data pointed to by Rev_Data_Start_Address[n]. A list of Revocation_IDs will be defined in Super Audio CD Part 3-General Annex D. Revocation_ID equal to zero is not allowed in the Revocation_TOC. Each value of Revocation_ID can be stored more than once in the Revocation_TOC.
For the storage of Revocation_ID[n] the following rule applies:

Revocation_ID[n] >= Revocation_ID[n-1] for n>1 and n<=N_Revocation_Systems

3.5.1.4 Rev_Data_Length

Rev_Data_Length[n] contains the length in Sectors of the Revocation Data starting at Rev_Data_Start_Address[n]. The following condition must be fulfilled:

Rev_Data_Start_Address[n] + Rev_Data_Length[n] <= Rev_Area_End_Address +1

3.5.1.5 Rev_Data_Start_Address

Rev_Data_Start_Address[n] contains the Sector address of the first Sector containing Revocation Data for the Revocation System with Revocation_ID[n]. The following condition must be fulfilled:

Rev_Area_Start_Address <= Rev_Data_Start_Address[n] <= Rev_Area_End_Address

Rev_Area_Start_Address and Rev_Area_End_Address are stored in Disc_Info_2, see chapter 3.1.1.6.

3.5.1.6 Rev_Attribute

The content of Rev_Attribute[n] is defined by the Revocation System identified by Revocation_ID[n]. If the Revocation System identified by Revocation_ID[n] does not define a value for Rev_Attribute[n], Rev_Attribute[n] must be set to zero. For each value of Revocation_ID, the storage of Rev_Attribute[n] must be in ascending order of the value of Rev_Attribute[n].

4. File System

A Super Audio CD disc optionally contains the UDF file system. A Super Audio CD disc optionally contains the ISO 9660 file system. If both UDF and ISO 9660 are used on a disc, both file systems must represent the same files and directories. The requirements in chapter 4 are only valid if at least one file system is present on the disc.

The minimum required directories on a Super Audio CD disc are: Root directory and one of 2C_AUDIO or MC_AUDIO. Other files and/or directories may exist outside the required Super Audio CD directory structure.

An example of the directory structure of a Super Audio CD disc is given in Figure 4-1.

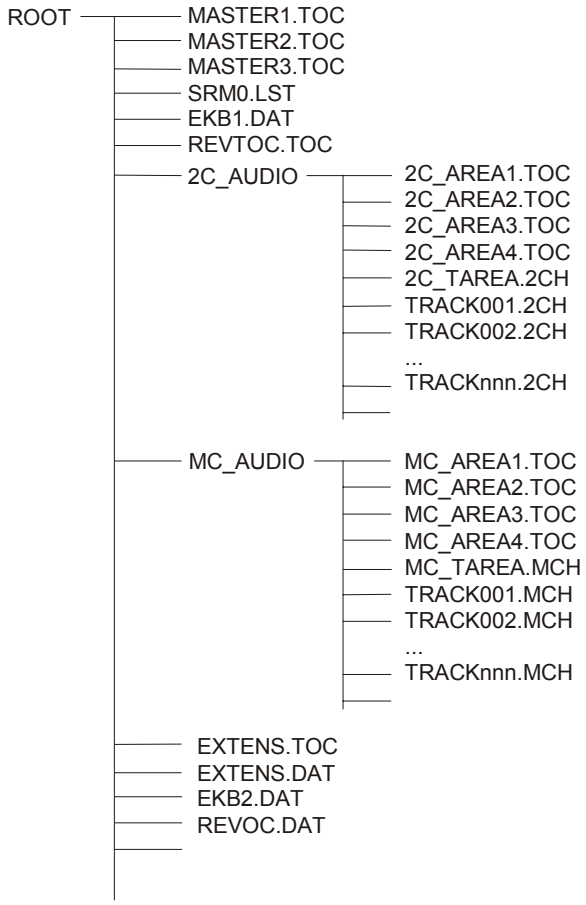

Figure 4-1 : Example of the Super Audio CD directory structure

4.1 UDF Requirements

If UDF is recorded on a Super Audio CD disc, it must be recorded such that the following requirements are fulfilled:

- UDF version 1.02 or higher shall be used on a Super Audio CD disc.
- The Minimum UDF Read Revision must be set to $0102, see UDF chapter 2.2.6.4.
- All UDF structures on the disc must be recorded such that they are in conformance with the UDF version 1.02 read requirement.
- The Anchor Volume Descriptor Pointer shall be recorded at Sector 256 and one of Sectors N and N-256, where Sector N is the last Sector in the Volume Space of the disc.

4.2 Root Directory

The required files in the ROOT directory are: MASTER1.TOC, MASTER2.TOC, MASTER3.TOC, SRM0.LST, EKB1.DAT, EKB2.DAT, REVTOC.TOC and REVOC.DAT. Note that on discs according to Version 1.3 or lower of the Super Audio CD Specification the files EKB1.DAT, EKB2.DAT, REVTOC.TOC and REVOC.DAT are not present.

4.2.1 Files MASTER1.TOC, MASTER2.TOC and MASTER3.TOC

The files MASTER1.TOC, MASTER2.TOC and MASTER3.TOC each must contain a copy of the Master TOC, see chapters 2.2.4.1 and 3.1. MASTER1.TOC must start at LSN 510. The name of the file MASTER1.TOC must be “MASTER1.TOC”. MASTER2.TOC must start at LSN 520. The length of the files MASTER1.TOC, MASTER2.TOC and MASTER3.TOC must be 20480 bytes each. The name of the file MASTER2.TOC must be “MASTER2.TOC”. MASTER3.TOC must start at LSN 530. The name of the file MASTER3.TOC must be “MASTER3.TOC”.

4.2.2 File SRM0.LST

The file SRM0.LST contains the DTCP_SRM_List as defined in chapter 3.3.1. SRM0.LST must start at LSN 400. The name of SRM0.LST must be "SRM0.LST". The length of the file SRM0.LST must be 48*2048 bytes.

4.2.3 Files EKB1.DAT and EKB2.DAT

The files EKB1.DAT and EKB2.DAT each contain one copy of EKB_Data as defined in chapter 3.4. EKB1.DAT must start at LSN 448. The name of EKB1.DAT must be “EKB1.DAT”, the name of EKB2.DAT must be “EKB2.DAT”. The length of both EKB1.DAT and EKB2.DAT is 62*2048 bytes.

4.2.4 File REVTOC.TOC

The file REVTOC.TOC contains the Revocation_TOC as defined in chapter 3.5. REVTOC.TOC must start at LSN 540. The name of REVTOC.TOC must be “REVTOC.TOC”. The length of REVTOC.TOC is 2048 bytes.

4.2.5 File EXTENS.TOC

The file EXTENS.TOC contains the Extension_TOC as defined in chapter 6.1. The name of EXTENS.TOC must be “EXTENS.TOC”. The length of EXTENS.TOC is variable and must be equal to 2048*E_TOC_Length bytes (see chapter 3.1.1.6.8). If E_TOC_Length is equal to zero, EXTENS.TOC shall not be present. Note that on discs according to Version 1.3 or lower of the Super Audio CD Specification EXTENS.TOC is not present.

4.2.6 File EXTENS.DAT

The file EXTENS.DAT contains the Extension_Data as defined in chapter 6.2. The name of EXTENS.DAT must be “EXTENS.DAT”. The length of EXTENS.DAT is variable and must be equal to 2048*(E_Data_End_Address - E_Data_Start_Address + 1) bytes, see chapter 3.1.1.6. If EXTENS.TOC is not present, EXTENS.DAT shall not be present. Note that on discs according to Version 1.3 or lower of the Super Audio CD Specification EXTENS.DAT is not present.

4.2.7 File REVOC.DAT

The file REVOC.DAT contains revocation data for Approved Digital Interfaces. The name of REVOC.DAT must be “REVOC.DAT”. The start address of REVOC.DAT must be equal to the value of Rev_Area_Start_Address, see chapter 3.1.1.6. The length of REVOC.DAT must be equal to 1024*2048 bytes.

4.3 Directory 2C_AUDIO

If and only if the 2-Channel Stereo Area (see chapter 2.2) is present on a Super Audio CD, the directory 2C_AUDIO must be present. The name of the 2C_AUDIO directory must be “2C_AUDIO”. The required files in the 2C_AUDIO directory are: 2C_AREA1.TOC, 2C_AREA2.TOC, 2C_TAREA.2CH and a file TRACKnnn.2CH for each Track in the 2-Channel Stereo Area.

4.3.1 Files 2C_AREA1.TOC and 2C_AREA2.TOC

The files 2C_AREA1.TOC and 2C_AREA2.TOC each must contain a copy of the Area TOC-A for the 2-Channel Stereo Area, see chapters 2.2.6.1 and 3.2. The name of the file 2C_AREA1.TOC must be "2C_AREA1.TOC". The name of the file 2C_AREA2.TOC must be "2C_AREA2.TOC". 2C_AREA1.TOC must start at the first Sector of the 2-Channel Stereo Area. If the 2C_AUDIO directory is present, the 2-Channel Stereo Area must start at LSN 544. The length of each file 2C_AREA1.TOC and 2C_AREA2.TOC must be equal to 2048*2CH_TOC_Length bytes (see chapter 3.1.1.3.6).

4.3.2 Files 2C_AREA3.TOC and 2C_AREA4.TOC

The files 2C_AREA3.TOC and 2C_AREA4.TOC each must contain a copy of the Area TOC-B for the 2-Channel Stereo Area, see chapters 2.2.6.1 and 3.2. The name of the file 2C_AREA3.TOC must be "2C_AREA3.TOC". The name of the file 2C_AREA4.TOC must be "2C_AREA4.TOC". The length of each file 2C_AREA3.TOC and 2C_AREA4.TOC must be equal to 2048*2CH_TOC_B_Length bytes (see chapter 3.1.1.6.5). If 2CH_TOC_B_Length is equal to zero, 2C_AREA3.TOC and 2C_AREA4.TOC shall not exist.

4.3.3 File 2C_TAREA.2CH

The file 2C_TAREA.2CH represents the Track Area of the 2-Channel Audio Area. The name of the file 2C_TAREA.2CH must be "2C_TAREA.2CH". The first Sector of 2C_TAREA.2CH must be equal to the Track_Area_Start_Address (see chapter 3.2.1.2.13) of the 2-Channel Stereo Area. The last Sector of 2C_TAREA.2CH must be equal to the Track_Area_End_Address (see chapter 3.2.1.2.14) as defined in Area TOC-B, or in Area TOC-A if Area TOC-B does not exist, for the 2-Channel Stereo Area. The length of the file 2C_TAREA.2CH must be a multiple of 2048 bytes.

4.3.4 Files TRACKnnn.2CH

The file TRACKnnn.2CH represents Track nnn in the 2-Channel Stereo Area, see chapter 2.2.6.2. The name of the files TRACKnnn.2CH must be "TRACKnnn.2CH", where "nnn" is the Track number of the represented Track, encoded in ISO 646 as a three digit decimal number with leading zeros. The highest value of "nnn" must be equal to the value of N_Tracks as defined in Area TOC-B, or in Area TOC-A if Area TOC-B does not exist, for the 2-Channel Stereo Area. TRACKnnn.2CH starts at the Track_Start_Address[nnn] (see chapter 3.2.2.2) of the 2-Channel Stereo Area. The size in Sectors of TRACKnnn.2CH must be equal to Track_Length[nnn] (see chapter 3.2.2.3) in the 2-Channel Stereo Area. The length of each TRACKnnn.2CH file must be a multiple of 2048 bytes. All files TRACKnnn.2CH overlap with the file 2C_TAREA.2CH.

4.4 Directory MC_AUDIO

If and only if the Multi Channel Area (see chapter 2.2) is present on a Super Audio CD, the directory MC_AUDIO must be present. The name of the MC_AUDIO directory must be "MC_AUDIO". The required files in the MC_AUDIO directory are: MC_AREA1.TOC, MC_AREA2.TOC, MC_TAREA.MCH and a file TRACKnnn.MCH for each Track in the Multi Channel Area.

4.4.1 Files MC_AREA1.TOC and MC_AREA2.TOC

The files MC_AREA1.TOC and MC_AREA2.TOC each must contain a copy of the Area TOC-A for the Multi Channel Stereo Area, see chapters 2.2.6.1 and 3.2. The name of the file MC_AREA1.TOC must be "MC_AREA1.TOC". The name of the file MC_AREA2.TOC must be "MC_AREA2.TOC". MC_AREA1.TOC must start at the first Sector of the Multi Channel Area. If the 2C_AUDIO directory is not present, the Multi Channel Area must start at LSN 544. The length of each file MC_AREA1.TOC and MC_AREA2.TOC must be equal to 2048*MC_TOC_Length bytes (see chapter 3.1.1.3.7).

4.4.2 Files MC_AREA3.TOC and MC_AREA4.TOC

The files MC_AREA3.TOC and MC_AREA4.TOC each must contain a copy of the Area TOC-B for the Multi Channel Stereo Area, see chapters 2.2.6.1 and 3.2. The name of the file MC_AREA3.TOC must be “MC_AREA3.TOC”. The name of the file MC_AREA4.TOC must be “MC_AREA4.TOC”. The length of each file MC_AREA3.TOC and MC_AREA4.TOC must be equal to 2048*MC_TOC_B_Length bytes (see chapter 3.1.1.6.6). If MC_TOC_B_Length is equal to zero, MC_AREA3.TOC and MC_AREA4.TOC shall not exist.

4.4.3 File MC_TAREA.MCH

The file MC_TAREA.MCH represents the Track Area of the Multi Channel Audio Area. The name of the file MC_TAREA.MCH must be “MC_TAREA.MCH”. The first Sector of MC_TAREA.MCH must be equal to the Track_Area_Start_Address (see chapter 3.2.1.2.13) of the Multi Channel Area. The last Sector of MC_TAREA.MCH must be equal to the Track_Area_End_Address (see chapter 3.2.1.2.14) as defined in Area TOC-B, or in Area TOC-A if Area TOC-B does not exist, for the Multi Channel Area. The length of the file MC_TAREA.MCH must be a multiple of 2048 bytes.

4.4.4 Files TRACKnnn.MCH

The file TRACKnnn.MCH represents Track nnn in the Multi Channel Area, see chapter 2.2.6.2. The name of the files TRACKnnn.MCH must be “TRACKnnn.MCH”, where “nnn” is the Track number of the represented Track, encoded in ISO 646 as a three digit decimal number with leading zeros. The highest value of “nnn” must be equal to the value of N_Tracks in as defined in Area TOC-B, or in Area TOC-A if Area TOC-B does not exist, for the Multi Channel Area. TRACKnnn.MCH starts at the Track_Start_Address[nnn] (see chapter 3.2.2.2) of the Multi Channel Area. The size in Sectors of TRACKnnn.MCH must be equal to Track_Length[nnn] (see chapter 3.2.2.3) of the Multi Channel Area. The length of each TRACKnnn.MCH file must be a multiple of 2048 bytes. All files TRACKnnn.MCH overlap with the file MC_TAREA.MCH.

4.5 Permissions

If a Super Audio CD disc contains the ISO 9660 file system and if Extended Attributes (see ISO 9660 chapter 9.5) are used, it is recommended to use the following Permissions (ISO 9660 9.5.3):

Files: System, Group, Owner, Any User: read is allowed
Directories: System, Group, Owner, Any User: read and execute are allowed.

If a Super Audio CD disc contains the UDF file system, it is recommended to use the following Permissions in the File Entry (ECMA 167 4/14.9.5):

Files: Other, Group and Owner: read is allowed
Directories: Other, Group and Owner: read and execute are allowed.

5. Audio Data Format

The information in a Track Area (see chapter 2.2.6.2) must be stored in one Byte Stream.
Note that for the definition of the Audio Data Format the Track Area includes Bonus Tracks if present.
A Byte Stream is stored in an integer number of Sectors. The Sectors used by a Byte Stream are called Audio Sectors. The format of Audio Sectors is defined in chapter 5.1.
A Byte Stream must contain an integer number of Multiplexed Frames. The format of a Multiplexed Frame is defined in chapter 5.2. The syntax of a Byte Stream is defined in chapter 5.3. An example of a Byte Stream is given in Figure 5-1.

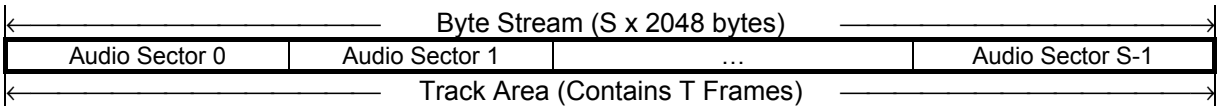

Figure 5-1 : Example of a Byte Stream

5.1 Audio Sector Format

The structure of the Sectors in a Byte Stream must be as defined in Figure 5-2. Each Audio Sector must start with an Audio_Header. The Audio_Header must be followed by one or more Packets. If the last Packet in a Sector does not end at the last byte of the Audio Sector, Stuffing bytes must be added upto the last byte of the Audio Sector. The syntax of an Audio Sector is defined in chapter 5.3.1.

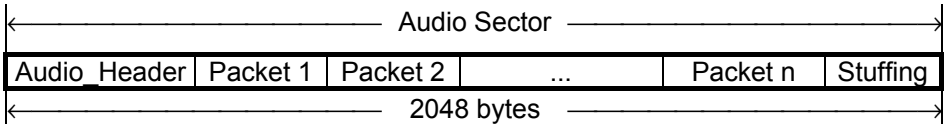

Figure 5-2 : Structure of an Audio Sector

Audio_Header, Packets and Stuffing are all contained in an integer number of bytes.

5.1.1 Audio_Header

The Audio_Header contains information on the Packets in the Sector, and on the Frames (see chapter 5.2) that start in the Sector. The syntax of the Audio_Header is defined in chapter 5.3.1.1.

5.1.2 Packet

One Packet only can contain one type of data, Audio Data, Supplementary Data or Padding Data. A Packet with Audio Data is called an Audio Packet. A Packet with Supplementary Data is called a Supplementary Data Packet. A Packet with Padding Data is called a Padding Packet. A Packet can only belong to one Audio Sector. An Audio Sector must contain minimum one Packet. An Audio Sector contains maximum seven Packets.

5.2 Multiplexed Frame Format

A Multiplexed Frame consists of an integer number of Packets. A Multiplexed Frame must contain minimum one Audio Packet. A Multiplexed Frame can contain zero or more Supplementary Data and Padding Packets. The first Packet of a Multiplexed Frame must always be an Audio Packet, both for Plain DSD and for DST coded Multiplexed Frames. A Multiplexed Frame is contained in maximum 16 Sectors, see chapter 5.3.1.1.5.3.

Each Multiplexed Frame has a time length of 1/75 Second. The concatenation of all Audio Packets in a Multiplexed Frame is called an Audio Frame. The concatenation of all Supplementary Data Packets in a Multiplexed Frame is called a Supplementary Data Frame. The concatenation of all Padding Packets in a Multiplexed Frame is called a Padding Frame. Audio Frames, Supplementary Data Frames and Padding Frames are called Elementary Frames.

Each Multiplexed Frame has a Time Code expressed in Minutes (0..255), Seconds (0..59), Frames (0..74). The Time Code must start with 0:0:0 at the first Audio Frame in the Track Area of an Audio Area. The Time Code must increase contiguously throughout the whole Track Area.

An example of a Multiplexed Frame is given in Figure 5-3. An example of the structure of Audio Sectors with Packets and Frames is given in Figure 5-4. An example of a Byte Stream is given in Figure 5-5.

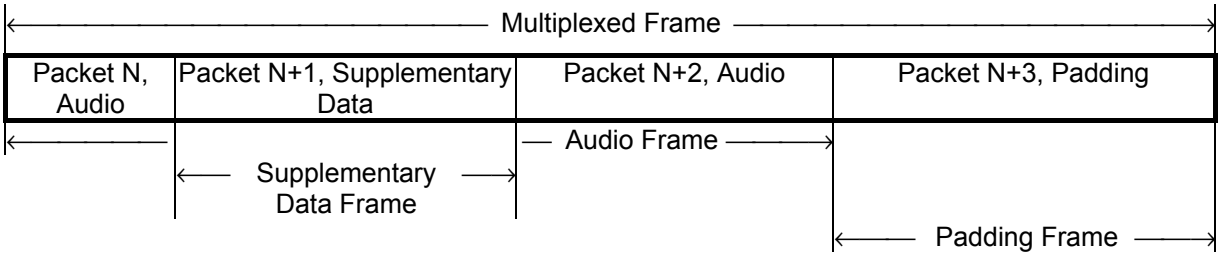

Figure 5-3 : Example of a Multiplexed Frame

Audio Sector M	Audio Sector M+1	Audio Sector M+2	Audio Sector M+3	Audio Sector M+4
Audio Header	Audio Header	Audio Header	Audio Header	Audio Header
Packet 2 of Audio Frame N-1	Packet 1 of Supplementary Data Frame N	Packet 3 of Audio Frame N	Packet 4 of Audio Frame N	Packet 2 of Audio Frame N+2
Packet 1 of Audio Frame N	Packet 2 of Audio Frame N		Packet 2 of Supplementary Data Frame N	
			Packet 1 of Audio Frame N+1	
			Padding Frame N+1	
			Packet 1 of Audio Frame N+2	

Figure 5-4 : Example of the structure of Audio Sectors

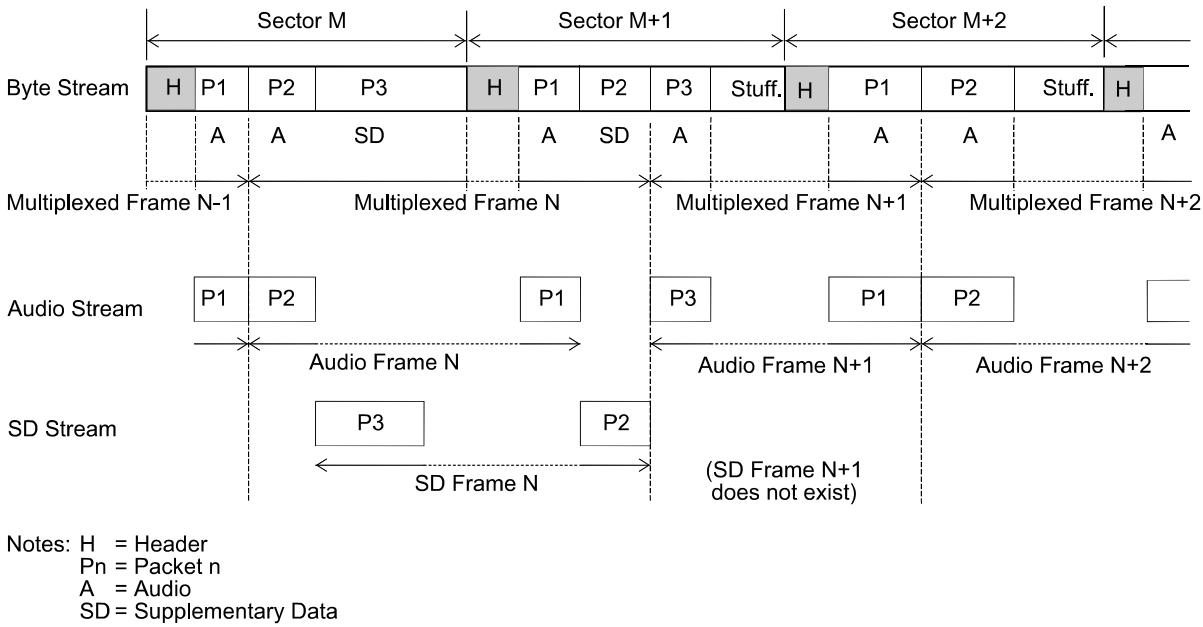

Figure 5-5 : Example of a Byte Stream

5.3 Byte Stream Syntax

A Byte Stream is the concatenation of all Multiplexed Frames in an Audio Area. The syntax of a Byte Stream is defined in Figure 5-6.

```
                                          #bytes          Format          value
Byte Stream(){
   for (i=0;i<Stream_Sector_Len; i++)
   {
       Audio_Sector()[i]                   2048           Audio_Sector
   }
}
```

Figure 5-6 : Syntax of Byte Stream

Stream_Sector_Len is the length in Sectors of the Byte Stream.

5.3.1 Audio_Sector

The Syntax of an Audio_Sector is defined in Figure 5-7.

```
                                          #bytes          format          value
Audio_Sector(){
   Audio_Header()                                         Audio_Header
   for (p=1; p<=N_Packets; p++)
   {
       Packet[p]                                          Packet
   }
   Stuffing                             until eos         Uint8             0
}
```

Figure 5-7 : Syntax of Audio_Sector

The value of N_Packets is defined in Audio_Header, see chapter 5.3.1.1.

5.3.1.1 Audio_Header

Audio_Header contains information on the content of the Packets in the Audio_Sector. The syntax of the Audio_Header is defined in Figure 5-8.

```
                                          #bits          format          value
Audio_Header(){
   N_Packets                                3            Uint3           1..7
   N_Frame_Starts                           3            Uint3           0..7
   Reserved                                 1            Uint1            0
   DST_Coded                                1            Uint1
   for (p=1; p<=N_Packets; p++)
   {
       Packet_Info()[p]                    16            Packet_Info
   }
   for (f=1; f<=N_Frame_Starts; f++)
   {
       Frame_Info()[f]                  24 or 32         Frame_Info
   }
}
```

Figure 5-8 : Syntax of Audio_Header

Note that the allocation of the fields in Audio_Header is defined in bits and not in bytes.

5.3.1.1.1 N_Packets

N_Packets contains the number of Packets in the Audio_Sector. The minimum allowed value for N_Packets is one.

5.3.1.1.2 N_Frame_Starts

N_Frame_Starts contains the number of Frames that start in this Audio_Sector.

5.3.1.1.3 DST_Coded

DST_Coded defines whether the Track Area is DST Coded or not. If Frame_Format (see chapter 3.2.1.2.3.1) is zero, DST_Coded must be set to one. If Frame_Format is equal to 2 or 3, DST_Coded must be set to zero.

5.3.1.1.4 Packet_Info

Packet_Info[p] contains control information for Packet[p] in the Audio_Sector. The syntax of Packet_Info[p] is defined in Figure 5-9.

```
                                          #bits          format          value
Packet_Info()[p] {
   Frame_Start[p]                           1            Uint1           0..1
   Reserved                                 1            Uint1            0
   Data_Type[p]                             3            Data_Type       2,3,7
   Packet_Length[p]                        11            Uint11         1..2045
}
```

Figure 5-9 : Syntax of Packet_Info

5.3.1.1.4.1 Frame_Start

Frame_Start indicates that a Frame starts in this Packet. If a Frame starts in this Packet, Frame_Start must be set to one, else Frame_Start must be set to zero.

5.3.1.1.4.2 Data_Type

Data_Type[p] defines the content of Packet[p] in the Audio Sector. The definition of Data_Type is given in Figure 5-10.

Value	Definition
0 .. 1	Reserved
2	Audio Packet
3	Supplementary Data Packet
4 .. 6	Reserved
7	Padding

Figure 5-10 : Definition of Data_Type

5.3.1.1.4.3 Packet_Length

Packet_Length[p] contains the length in bytes of Packet[p]. The minimum allowed length of a Packet is one byte. The maximum length of a Packet is 2045 bytes.

5.3.1.1.5 Frame_Info

For each Frame that starts in the Audio Sector, there is a Frame_Info[f]. For each Frame, the Time Code is given. In addition, if the Frame_Format is zero, the number of Sectors used by the Frame and the number of Audio Channels is given. The syntax of Frame_Info is defined in Figure 5-11.

```
                                                #bits          format          value
Frame_Info()[f] {
   Time_Code[f]                                   24          Time_Code
   if (Frame_Format == 0 /* == DST */)
   {
       Ch_Bit_1[f]                                 1          Uint1
       N_Sectors[f]                                5          Uint5           1..16
       Ch_Bit_2[f]                                 1          Uint1
       Ch_Bit_3[f]                                 1          Uint1
   }
}
```

Figure 5-11 : Syntax of Frame_Info

The value of Frame_Format is given in the Area TOC, see chapter 3.2.1.2.3.1. Note that if DST is used, the length of Frame_Info is 32 bits, else the length is 24 bits.

5.3.1.1.5.1 Time_Code

Time_Code[f] gives the Time Code for all Elementary Frames in Multiplexed Frame[f]. The syntax of Time_Code is defined in chapter 1.7.2.9.

5.3.1.1.5.2 Ch_Bit_n

The combination of Ch_Bit_1[f], Ch_Bit_2[f] and Ch_Bit_3[f] defines the number of audio channels used in Audio Frame[f]. The usage of Ch_Bit_1, Ch_Bit_2 and Ch_Bit_3 is defined in Figure 5-12. The number of audio channels encoded in Ch_Bit_1, Ch_Bit_2 and Ch_Bit_3 must be equal to the value of N_Channels in the Area TOC, see chapter 3.2.1.2.4.

Ch_Bit_1	Ch_Bit_2	Ch_Bit_3	Meaning
0	0	0	N_Channels = 2
0	0	1	N_Channels = 5
0	1	0	N_Channels = 6
All other combinations			Reserved for future use

Figure 5-12 : Definition of Ch_Bit_n

5.3.1.1.5.3 N_Sectors

N_Sectors[f] is the number of Sectors used by Multiplexed Frame[f]. If the start of the first Packet of a Multiplexed Frame is located in Sector X, and the end of the last Packet of the same Multiplexed Frame is located in Sector Y, then the value of N_Sectors is equal to Y-X+1. Two examples with N_Sectors = 3 is given in Figure 5-13.

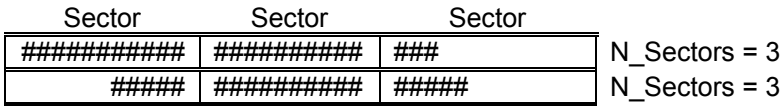

Figure 5-13 : Examples of the value of N_Sectors

If N_Channels (see chapter 3.2.1.2.4) is equal to 2, the value of N_Sectors must be between 1 and 7. If N_Channels is equal to 5, the value of N_Sectors must be between 1 and 14. If N_Channels is equal to 6, the value of N_Sectors must be between 1 and 16.

5.3.1.2 Packet

The length of every Packet must be an integer number of bytes. An Audio Packet contains bytes from the Audio Stream, see chapter 5.6. A Supplementary Data Packet contains Supplementary Data, see chapter 5.7. All bytes in a Padding Packet must be set to zero.

5.3.1.3 Stuffing

If the last byte of the last Packet in an Audio_Sector is not the last byte of that Audio_Sector, Stuffing bytes must be added until the end of that Audio_Sector.
If an Audio Area contains Bonus Tracks, the following rules apply:

- The first Packet of the Pause of the first Bonus Track (Pause[N+1] in Figure 2-8) must be the first Packet of an Audio_Sector. Note that if the length of the Pause is equal to zero, the first Packet of the first Bonus Track must be the first Packet of an Audio_Sector.
- If the last byte of the last Packet of the last Track preceding the first Bonus Track is not the last byte of an Audio_Sector, Stuffing bytes must be added until the end of that Audio_Sector.

The content of Stuffing bytes must be zero.

5.4 2-Channel Stereo Area

The Audio Frames in the 2-Channel Stereo Area can be either Plain DSD coded or DST coded. If the 2-Channel Stereo Area contains Bonus Tracks, the 2-Channel Stereo Area must be DST coded.

5.4.1 Plain DSD

If the 2-Channel Stereo Area is Plain DSD coded (see chapter 3.2.1.2.3.1), the Audio Sectors must use one out of two available fixed formats. In the fixed formats, a fixed pattern of Audio Packets, Supplementary Data Packets and Padding Packets is used. The two fixed formats are: 3 Frames in 14 Sectors, and 3 Frames in 16 Sectors.
The last Sector of a Track Area, must be a Sector that contains the last Audio Packet of an Audio Frame.

5.4.1.1 3 Frames in 14 Sectors

With the 3 Frames in 14 Sectors format, the Byte Stream is blocked in groups of 14 Sectors. Each group of 14 Sectors contains 3 Multiplexed Frames. For each block of 3 Frames, 365 bytes of Supplementary Data are available. For each Sector in a group of 14 Sectors, the length of all Packets is defined in Figure 5-14. For each Sector in a group of 14 Sectors, the Packet sequence is defined in Figure 5-15.

Sector number	Audio Header length	Supplementary Data Packet length	Audio Packet length
n	8	24	2016
n+1	5	27	2016
n+2	5	27	2016
n+3	5	27	2016
n+4	10	22	1344, 672
n+5	5	27	2016
n+6	5	27	2016
n+7	5	27	2016
n+8	5	27	2016
n+9	10	22	672, 1344
n+10	5	27	2016
n+11	5	27	2016
n+12	5	27	2016
n+13	5	27	2016

Figure 5-14 : Packet length for the 3 Frames in 14 Sectors format

Sector number	Audio Header	Supplementary Data Packets	Audio Packets	
n	Audio Header	SD Packet[0][5]	Audio Packet[1][1]	
n+1	Audio Header	SD Packet[1][1]	Audio Packet[1][2]	
n+2	Audio Header	SD Packet[1][2]	Audio Packet[1][3]	
n+3	Audio Header	SD Packet[1][3]	Audio Packet[1][4]	
n+4	Audio Header	SD Packet[1][4]	Audio Packet[1][5]	Audio Packet[2][1]
n+5	Audio Header	SD Packet[2][1]	Audio Packet[2][2]	
n+6	Audio Header	SD Packet[2][2]	Audio Packet[2][3]	
n+7	Audio Header	SD Packet[2][3]	Audio Packet[2][4]	
n+8	Audio Header	SD Packet[2][4]	Audio Packet[2][5]	
n+9	Audio Header	SD Packet[2][5]	Audio Packet[2][6]	Audio Packet[3][1]
n+10	Audio Header	SD Packet[3][1]	Audio Packet[3][2]	
n+11	Audio Header	SD Packet[3][2]	Audio Packet[3][3]	
n+12	Audio Header	SD Packet[3][3]	Audio Packet[3][4]	
n+13	Audio Header	SD Packet[3][4]	Audio Packet[3][5]	
		SD Packet[3][5]		

Figure 5-15 : Packet sequence for the 3 Frames in 14 Sectors format

In Figure 5-15, SD Packet[a][b] indicates Supplementary Data Packet b of Frame a. In Figure 5-15, Audio Packet[a][b] indicates Audio Packet b of Frame a.
SD Packet[0][5] is the last SD Packet of the previous group of 14 Sectors. The last SD Packet of Frame 3, is the first SD Packet in the next group of 14 Sectors.

The first Sector of a Track can only be Sector n, Sector n+4 or Sector n+9, see chapter 3.2.2.2. The first Sector of a Track Area must be Sector n. In the first Sector of a Track Area, the content of SD Packet[0][5] must be set to zero.

The last Sector of a Track can only be Sector n, Sector n+4 or Sector n+9. The last Sector of a Track Area can be Sector n, Sector n+4 or Sector n+9.
The last Sector of a Track Area must contain a dummy start of a new Frame. The Audio Header in the last Sector of a Track Area must contain a Time_Code with the content of all fields set to $FF. After the last Packet of the last Multiplexed Frame, the last Sector of a Track Area must contain one Padding Packet, see chapters 5.3.1.1.4.2 and 5.3.1.2. This Padding Packet forms the content of the Multiplexed Frame with the Time_Code set to $FF:$FF:$FF.

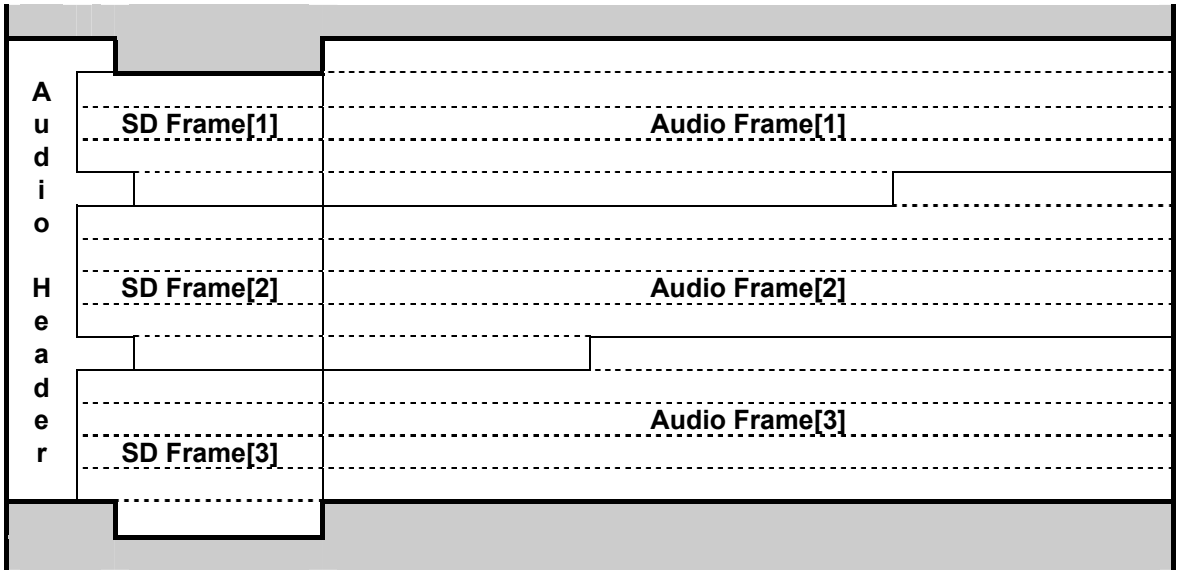

Figure 5-16 : Layout of the 3 Frames in 14 Sectors format

5.4.1.2 3 Frames in 16 Sectors

With the 3 Frames in 16 Sectors format, the Byte Stream is blocked in groups of 16 Sectors. Each group of 16 Sectors contains 3 Multiplexed Frames. For each block of 3 Frames, 4451 bytes of Supplementary Data are available. For each Sector in a group of 16 Sectors, the length of all Packets is defined in Figure 5-17. For each Sector in a group of 16 Sectors, the Packet sequence is defined in Figure 5-18.

Sector number	Audio Header length	Supplementary Data Packet length	Audio Packet length
n	8	276	1764
n+1	5	279	1764
n+2	5	279	1764
n+3	5	279	1764
n+4	5	279	1764
n+5	10	274	588, 1176
n+6	5	279	1764
n+7	5	279	1764
n+8	5	279	1764
n+9	5	279	1764
n+10	10	274	1176, 588
n+11	5	279	1764
n+12	5	279	1764
n+13	5	279	1764
n+14	5	279	1764
n+15	5	279	1764

Figure 5-17 : Packet length for the 3 Frames in 16 Sectors format

Sector number	Audio Header	Supplementary Data Packets	Audio Packets
n	Audio Header	SD Packet[0][6]	Audio Packet[1][1]
n+1	Audio Header	SD Packet[1][1]	Audio Packet[1][2]
n+2	Audio Header	SD Packet[1][2]	Audio Packet[1][3]
n+3	Audio Header	SD Packet[1][3]	Audio Packet[1][4]
n+4	Audio Header	SD Packet[1][4]	Audio Packet[1][5]
n+5	Audio Header	SD Packet[1][5]	Audio Packet[1][6] | Audio Packet[2][1]
n+6	Audio Header	SD Packet[2][1]	Audio Packet[2][2]
n+7	Audio Header	SD Packet[2][2]	Audio Packet[2][3]
n+8	Audio Header	SD Packet[2][3]	Audio Packet[2][4]
n+9	Audio Header	SD Packet[2][4]	Audio Packet[2][5]
n+10	Audio Header	SD Packet[2][5]	Audio Packet[2][6] | Audio Packet[3][1]
n+11	Audio Header	SD Packet[3][1]	Audio Packet[3][2]
n+12	Audio Header	SD Packet[3][2]	Audio Packet[3][3]
n+13	Audio Header	SD Packet[3][3]	Audio Packet[3][4]
n+14	Audio Header	SD Packet[3][4]	Audio Packet[3][5]
n+15	Audio Header	SD Packet[3][5]	Audio Packet[3][6]
		SD Packet[3][6]	

Figure 5-18 : Packet sequence for the 3 Frames in 16 Sectors format

In Figure 5-18, SD Packet[a][b] indicates Supplementary Data Packet b of Frame a. In Figure 5-18, Audio Packet[a][b] indicates Audio Packet b of Frame a.
SD Packet[0][6] is the last SD Packet of the previous group of 16 Sectors. The last SD Packet of Frame 3, is the first SD Packet in the next group of 16 Sectors.

The first Sector of a Track can only be Sector n, Sector n+5 or Sector n+10, see chapter 3.2.2.2. The first Sector of a Track Area must be Sector n. In the first Sector of a Track Area, the content of SD Packet[0][6] must be set to zero.

The last Sector of a Track can only be Sector n, Sector n+5 or Sector n+10. The last Sector of a Track Area can be Sector n, Sector n+5 or Sector n+10.
The last Sector of a Track Area must contain a dummy start of a new Frame. The Audio Header in the last Sector of a Track Area must contain a Time_Code with the content of all fields set to $FF. After the last Packet of the last Multiplexed Frame, the last Sector of a Track Area must contain one Padding Packet, see chapters 5.3.1.1.4.2 and 5.3.1.2. This Padding Packet forms the content of the Multiplexed Frame with the Time_Code set to $FF:$FF:$FF.

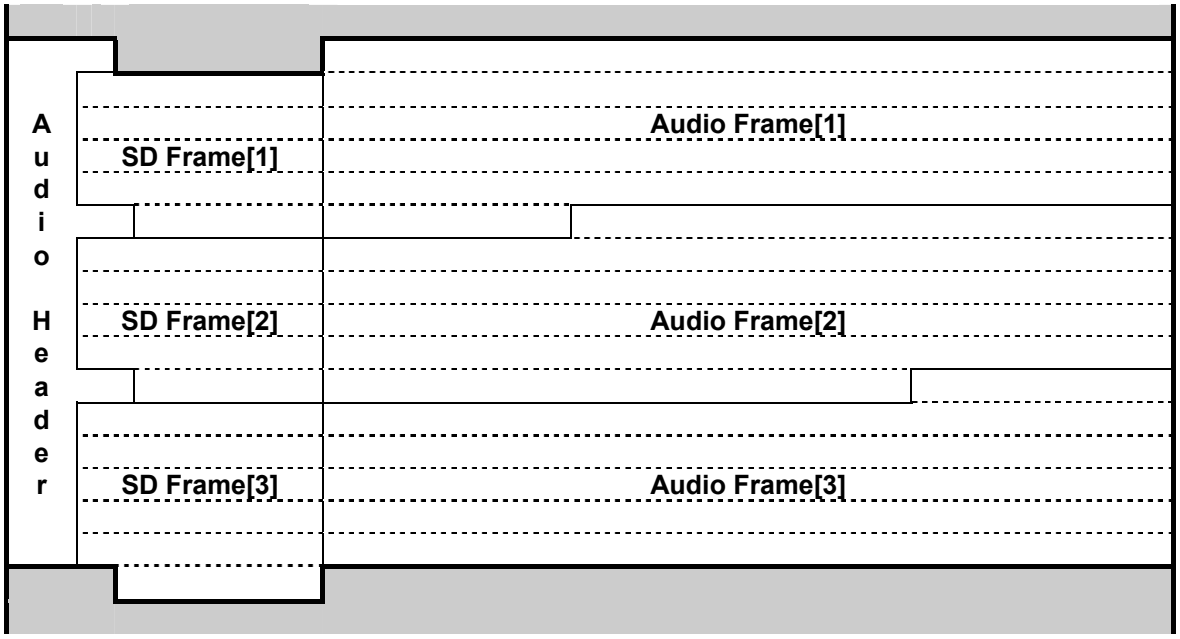

Figure 5-19 : Layout of the 3 Frames in 16 Sectors format

5.4.2 DST Coded DSD

If the 2-Channel Stereo Area is DST coded (see chapter 3.2.1.2.3.1), the Audio Sectors use a flexible format. In the flexible format, the first Packet in a Multiplexed Frame must be an Audio Packet. After the first Packet, the pattern of Audio Packets, Supplementary Data Packets and Padding Packets can be chosen freely.

5.5 Multi Channel Area

The Audio signal in the Multi Channel Area must be DST coded.
The Audio Sectors used in the Multi Channel Area must use the flexible format. In the flexible format, the first Packet in a Multiplexed Frame must be an Audio Packet. After the first Packet, the pattern of Audio Packets, Supplementary Data Packets and Padding Packets can be chosen freely.

5.6 Audio Streams

An Audio Stream contains the DSD audio signal, Plain DSD or DST coded DSD. An Audio Stream is the concatenation of all Audio Frames (see chapter 5.2) in a Byte Stream, see Figure 5-5. Note that if the Time Code of a Frame is not valid (see chapter 1.7.2.9), that Frame does not contain valid audio information.

5.6.1 DSD Sampled Bit Stream Data

For a 2-Channel Audio signal, the DSD Sampled Bit sequence is defined as follows: L_0, L_1, L_2, L_3, L_4, L_5, L_6, L_7, R_0, R_1, R_2, R_3, R_4, R_5, R_6, R_7, L_8, L_9, L_{10}, … Where L_0 is the first sampled bit of the left channel of an Audio Frame.
Per Audio Channel, eight successively sampled bits are grouped into one Audio Byte. The most significant bit of an Audio Byte is the first sampled bit of that byte.

5.6.2 Structure of the Audio Stream

Plain DSD Audio Frames, for the 2-Channel Stereo Area, are structured according to the specification in chapter 5.4.1. DST coded Audio Frames have a variable length. The reference model of the DST Decoder is defined in Annex C.

5.6.3 Audio_Frame

Audio_Frame contains the DST or Plain DSD coded audio information for one Frame. The maximum size of an Audio_Frame is equal to the size of a Plain DSD coded Audio_Frame plus one byte. The syntax of an Audio_Frame is defined in Figure 5-20.

```
                                                    # bits   format      value
Audio_Frame() {
   if ((Frame_Format==2) || (Frame_Format==3))
   {
       DSD()                                                DSD
   }
   if (Frame_Format==0)
   {
       DST()                                                DST
   }
}
```

Figure 5-20 : Syntax of Audio_Frame

The value of Frame_Format is defined in the Area_TOC_0, see chapter 3.2.1.2.3.

5.6.3.1 DSD

DSD contains the audio data for one Plain DSD Audio_Frame. The syntax of DSD is defined in Figure 5-21. An example of a 5 channel DSD Frame is given in Figure 5-22.

```
                                                        # bytes     format
DSD() {
   for (Byte_Nr=0; Byte_Nr<Frame_Length; Byte_Nr++)
   {
       for (Channel_Nr=1; Channel_Nr<=N_Channels;

       {
           DSD_Byte[Channel_Nr][Byte_Nr]                    1       Audio_Byte
       }
   }
}
```

Figure 5-21 : Syntax of DSD

The definition of Audio_Byte is given in chapter 5.6.1.
Frame_Length is the length of an Audio Frame in bytes per audio channel. The Frame_Length can be calculated from the Sample_Frequency (see chapter 3.2.1.2.2). For a Sample_Frequency of 64*44100 Hz, Frame_Length can be calculated with the following formula:

$$\text{Frame\_Length} = \frac{64 * 44100}{75 * 8} = 4704 \text{ bytes per Audio Channel}$$

Channel_Nr is the number of the Audio Channel, for the Channel Mapping see chapter 3.2.1.2.4.
N_Channels is the number of audio channels used. The value of N_Channels is given in the Area_TOC_0, see chapter 3.2.1.2.

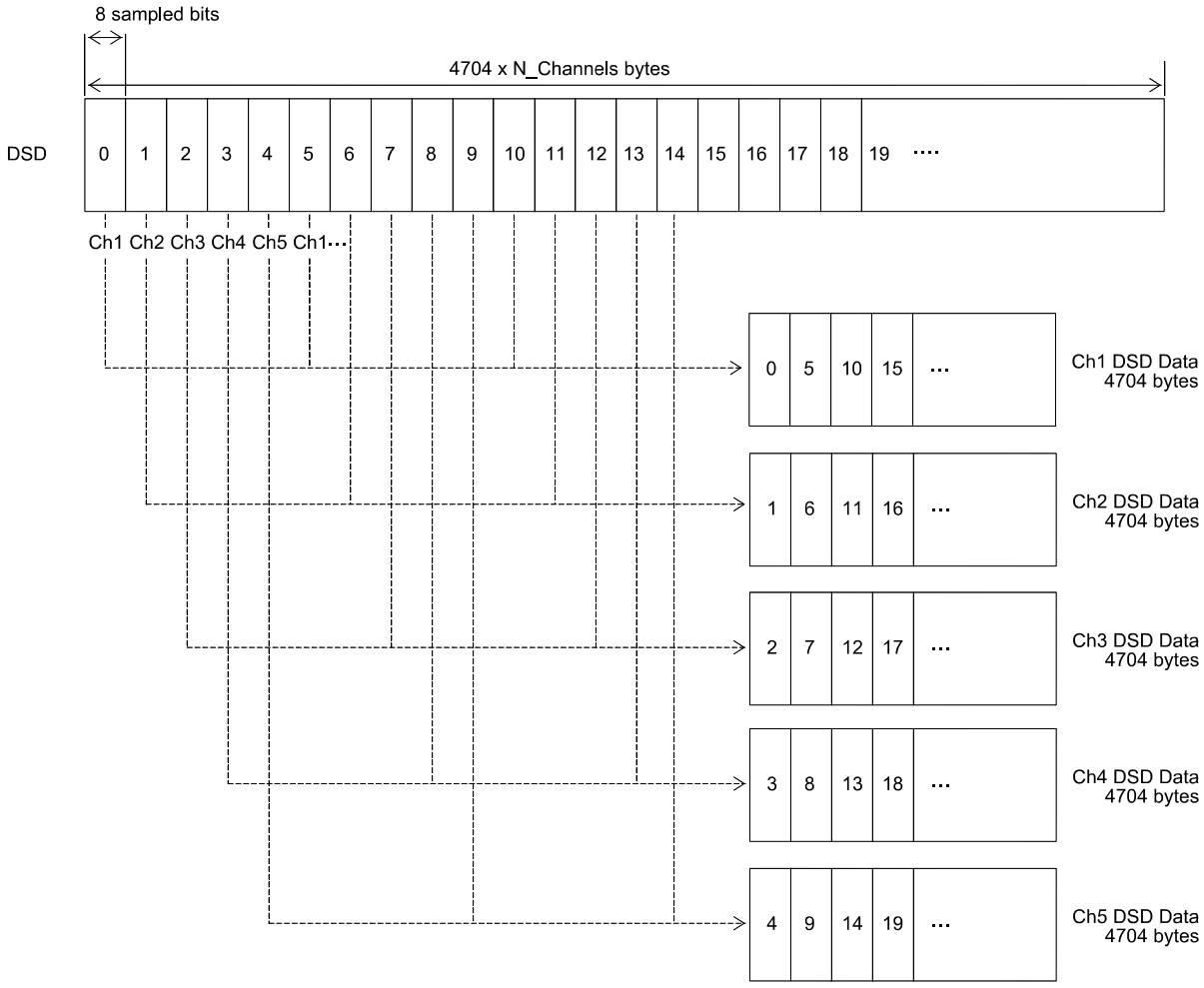

Figure 5-22 : Example of a 5 channel DSD Frame

5.6.3.1.1 DSD_Byte

DSD_Byte[Channel_Nr][Byte_Nr] contains the DSD signal as defined in chapter 5.6.1.

5.6.3.2 DST

DST contains the audio data for one DST coded Audio_Frame. The syntax of DST is defined in Figure 5-23.

```
                                          # bits       format                  value
DST() {
   Processing_Mode                           1         BsMsbf
   if (Processing_Mode == 0)
   {
       DST_X_Bit                             1         BsMsbf
       Reserved                              6         BsMsbf                    0
       DSD()                                           DSD
   }
   else
   {
       Segmentation()                                  Segmentation
       Mapping()                                       Mapping
       Half_Probability()                              Half_Probability
       Filter_Coef_Sets()                              Filter_Coef_Sets
       Probability_Tables()                            Probability_Tables
       Arithmetic_Coded_Data()                         Arithmetic_Coded_Data
   }
}
```

Figure 5-23 : Syntax of DST

5.6.3.2.1 Processing_Mode

If the Processing_Mode bit is set to one, the Audio_Frame contains the DST_X_Bit and the DSD signal in a lossless coded form. If the Processing_Mode bit is set to zero, the Audio_Frame contains the DST_X_Bit and the DSD signal without lossless coding.

5.6.3.2.2 DST_X_Bit

If Frame_Format (see chapter 3.2.1.2.3.1) is equal to zero, the Track Area is DST coded and each Audio_Frame contains one DST_X_Bit.
On the disc two patterns are allowed for the DST_X_Bit:
Pattern A: DST_X_Bit = 0
Pattern B: Successive DST_X_Bits within an Audio Area shall be repetitively filled with the pattern $7E 00 00 00 00 00 00 00 00 00 00 00 00 on discs according to this version of the specification. DST_X_Bits must be encoded on the disc with the most significant bit of the most significant byte first.
The pattern $7E 00 00 00 00 00 00 00 00 00 00 00 00 is called the Basic Pattern. Any successive sequence of bits from the Basic Pattern with length < 104 bits is called an Incomplete Basic Pattern. The disc must be encoded such that there is maximum one Incomplete Basic Pattern in between two Basic Patterns.
More encoding patterns for the DST_X_Bit can be defined in future versions of the Super Audio CD Specification. It is allowed that one Audio Area contains both Pattern A and Pattern B. Players according to this version of the Super Audio CD Specification shall ignore the content of the DST_X_Bit.

5.6.3.2.2.1 P(DST_X_Bit) (Multiplexing/Demultiplexing)

P(DST_X_Bit) shall be derived from Coef[0][0] (see chapter 5.6.3.2.7) as follows. If Coef[0][0] = $\%c_8c_7c_6c_5c_4c_3c_2c_1c_0$, then P(DST_X_Bit) shall be equal to $\%0c_0c_1c_2c_3c_4c_5c_6$ + 1 of type Uint8. c_x represents the value of individual bits, with x is in the range 0..8. The value of P(DST_X_Bit) is in the range 1..128.

5.6.3.2.3 DSD

See chapter 5.6.3.1.

5.6.3.2.4 Segmentation

For each Audio Channel, the Audio Frame is partitioned into one or more Segments for Filters, and one or more Segments for Ptables. Each Segment may use a different Prediction Filter / Ptable. An example of Segmentation is shown in Figure 5-24. The syntax of Segmentation is defined in Figure 5-25.

	← Audio Frame →			
Channel number	Segments			
1	Segment 1	Segment 2	Segment 3	Segment 4
2	Segment 1	Segment 2	Segment 3	Segment 4
3	Segment 1			
4	Segment 1		Segment 2	
5	Segment 1			
6	Segment 1	Segment 2	Segment 3	

Figure 5-24 : Example of Segmentation

```
                                                  # bits      format
Segmentation() {
   Same_Segmentation                                 1
   if(Same_Segmentation == 0)
   {
       Filter_Segmentation()                                 Segment_Alloc
       Ptable_Segmentation()                                 Segment_Alloc
   }
   else
   {
       Filter_And_Ptable_Segmentation()                      Segment_Alloc
   }
}
```

Figure 5-25 : Syntax of Segmentation

Filter_Segmentation
For each Audio Channel, the Audio Frame is partitioned into one or more Segments for Prediction Filters. Each Segment may use a different Prediction Filter. The variables Nr_Of_Segments[ ] and Segment_Length[ ][ ] from Segment_Alloc (see chapter 5.6.3.2.4.2) used for Filter_Segmentation are referred to as:

- Filters.Nr_Of_Segments[Channel_Nr] and
- Filters.Segment_Length[Channel_Nr][1..Filters.Nr_Of_Segments[Channel_Nr]].

With Channel_Nr = 1..N_Channels.

Ptable_Segmentation
For each Audio Channel, the Audio Frame is partitioned into one or more Segments for Ptables. Each Segment may use a different Ptable. The variables Nr_Of_Segments[ ] and Segment_Length[ ][ ] from Segment_Alloc (see chapter 5.6.3.2.4.2) used for Ptable_Segmentation are referred to as:

- Ptables.Nr_Of_Segments[Channel_Nr] and
- Ptables.Segment_Length[Channel_Nr][1..Ptables.Nr_Of_Segments[Channel_Nr]].

With Channel_Nr = 1..N_Channels.

Filter_And_Ptable_Segmentation
For each Audio Channel, the Audio Frame is partitioned into one or more Segments. Each Segment may use a different combination of Prediction Filter and Ptable. For each Audio Channel, the following equations must be true:

Filters.Nr_Of_Segments[Channel_Nr] = Ptables.Nr_Of_Segments[Channel_Nr] = Nr_Of_Segments[Channel_Nr]
Filters.Segment_Length[Channel_Nr][ ] = Ptables.Segment_Length[Channel_Nr][ ] = Segment_Length[Channel_Nr][ ]

With Channel_Nr = 1..N_Channels.

5.6.3.2.4.1 Same_Segmentation

If Same_Segmentation is one, the Ptables and Prediction Filters use one and the same Segmentation. If Same_Segmentation is zero, the partitioning for Prediction Filters is independent from the partitioning for Ptables, for the Audio Frame.

5.6.3.2.4.2 Segment_Alloc

Segment_Alloc defines the Segmentation for the Prediction Filters and/or the Ptables. The syntax of Segment_Alloc is defined in Figure 5-26.
For each Audio Channel, the variables Nr_Of_Segments and Segment_Length[1..Nr_Of_Segments] from Channel_Segmentation (see chapter 5.6.3.2.4.2.2) are referred to as:

- Nr_Of_Segments[Channel_Nr]
- Segment_Length[Channel_Nr][1..Nr_Of_Segments[Channel_Nr]]

In the syntax diagrams, syntax variables are shown in *italics*.

```
                                                    # bits    format
Segment_Alloc() {
   Resolution_Read = false
   Same_Segm_For_All_Channels                          1
   if(Same_Segm_For_All_Channels == 0)
   {
       for (Channel_Nr=1; Channel_Nr<=N_Channels;
            Channel_Nr++)
       {
           Channel_Segmentation()[Channel_Nr]              Channel_Segmentation
       }
   }
   else
   {
       Channel_Segmentation()[1]                           Channel_Segmentation
   }
}
```

Figure 5-26 : Syntax of Segments

Resolution_Read indicates whether Resolution from Channel_Segmentation, see chapter 5.6.3.2.4.2.2, has been read. Resolution_Read is set to true in the Channel_Segmentation of the first Audio Channel with more than one Segment. Note that if Prediction Filters and Ptables use independent Segmentation, they also use an independent Resolution_Read.
Channel_Nr is a local index variable.
N_Channels is the number of audio channels used. The value of N_Channels is given in the Area_TOC_0, see chapter 3.2.1.2.

5.6.3.2.4.2.1 Same_Segm_For_All_Channels

If Same_Segm_For_All_Channels is one, only the Segmentation for the first Audio Channel is stored and Channel_Segmentation()[Channel_Nr] = Channel_Segmentation()[1] for all Audio Channels. If Same_Segm_For_All_Channels is zero, the Audio Frame is partitioned into segments independent for each Audio Channel.

5.6.3.2.4.2.2 Channel_Segmentation

Channel_Segmentation defines the Segmentation of the Prediction Filters and/or the Ptables. The syntax of Channel_Segmentation is defined in Figure 5-27.
The following variables are used in the syntax of Channel_Segmentation:

- Nr_Of_Segments
- Start[1..Nr_Of_Segments]
- Segment_Length[1..Nr_Of_Segments]

```
                                                                  # bits    format
Channel_Segmentation() {
   Nr_Of_Segments = 1
   Start[1] = 0
   End_Of_Channel_Segm                                             1
   while(End_Of_Channel_Segm == 0)
   {
       if (Resolution_Read == false)
       {
           Resolution                                             13     UiMsbf
           Resolution_Read = true
       }
       Scaled_Length[Nr_Of_Segments]                             1..13   UiMsbf
       Segment_Length[Nr_Of_Segments] =
           Resolution * Scaled_Length[Nr_Of_Segments]
       Start[Nr_Of_Segments+1] =
           Start[Nr_Of_Segments] + Segment_Length[Nr_Of_Segments]
       Nr_Of_Segments++
       End_Of_Channel_Segm                                         1
   }
   Segment_Length[Nr_Of_Segments] =
           Frame_Length - Start[Nr_Of_Segments]
}
```

Figure 5-27 : Syntax of Channel_Segmentation

Nr_Of_Segments is the number of Segments for the current Audio Channel. The maximum number of Segments is MAXNRSEGS. MAXNRSEGS must be 4 for the Filter_Segmentation, 8 for the Ptable_Segmentation, and 4 for the Filter_And_Ptable_Segmentation.
Resolution_Read indicates whether the variable Resolution has been read in this or a previous Channel_Segmentation. Resolution_Read is set to true in the Channel_Segmentation of the first Audio Channel with more than one Segment. Note that if Prediction Filters and Ptables use independent Segmentation, they also use an independent Resolution_Read.
Segment_Length[Seg_Nr] contains the length of the Segment in bytes, where:
1 <= Seg_Nr <= Nr_Of_Segments
Start[Seg_Nr] is the starting position in bytes of Segment[Seg_Nr].
Frame_Length, see chapter 5.6.3.1.

5.6.3.2.4.2.2.1 End_Of_Channel_Segm

If End_Of_Channel_Segm is zero, one or more values for Scaled_Length will follow. If End_Of_Channel_Segm is one, the Channel_Segmentation structure ends.

5.6.3.2.4.2.2.2 Resolution

Each value of Scaled_Length is multiplied by Resolution to get the Segment length in bytes. Resolution is stored only once, at the beginning of the first Audio Channel with more than one Segment. If all Audio Channels have only one Segment, Resolution is not encoded.
Resolution must be in the range of 1 to Frame_Length - MINSEGLEN. MINSEGLEN must be 128 bytes for Filter_Segmentation, 4 bytes for Ptable_Segmentation, and 128 bytes for Filter_And_Ptable_Segmentation.

5.6.3.2.4.2.2.3 Scaled_Length

For each Segment, except the last one, a value of Scaled_Length is encoded. The length in bytes of a Segment is calculated with the following formula:

Segment_Length[Seg_Nr] = Resolution * Scaled_Length[Seg_Nr]
Where:
1 <= Seg_Nr < Nr_Of_Segments

The minimum Segment length of each Segment is MINSEGLEN, see chapter 5.6.3.2.4.2.2.2.
For Ptable_Segmentation the length of the first Segment of each Audio Channel must be at least (Pred_Order[Filter[Channel_Nr][1]]+7)/8 bytes. For the definition of Filter[ ][ ] see chapter 5.6.3.2.5. For the definition of Pred_Order[ ] see chapter 5.6.3.2.7.
The number of bits needed to encode Scaled_Length[Seg_Nr] depends on the value of Range. Range must be calculated with the following formula:

$$\text{Range} = \text{Trunc}\left(\frac{\text{Frame\_Length} - \text{Start}[\text{Seg\_Nr}] - \text{MINSEGLEN}}{\text{Resolution}}\right)$$

If $2^{n-1} \le \text{Range} < 2^n$, n bits must be used to encode Scaled_Length[Seg_Nr], see Figure 5-28. The minimum value of Range is 1. The length of the last Segment is not encoded on the disc. The length of the last Segment can be calculated from the Frame Length and the start position of the last Segment with the following formula:

Segment_Length[Nr_Of_Segments] = Frame_Length - Start[Nr_Of_Segments]

Range	bits used
1	1
2..3	2
4..7	3
8..15	4
16..31	5
32..63	6
64..127	7

Range	bits used
128..255	8
256..511	9
512..1023	10
1024..2047	11
2048..4095	12
4096..8191	13

Figure 5-28 : Bits used to encode Scaled_Length

5.6.3.2.5 Mapping

Mapping defines the Prediction Filters and Ptables used with the Segments specified in chapter 5.6.3.2.4. An example of Prediction Filter allocation for the Segmentation example in Figure 5-24 is shown in Figure 5-29. The syntax of Mapping is defined in Figure 5-30.

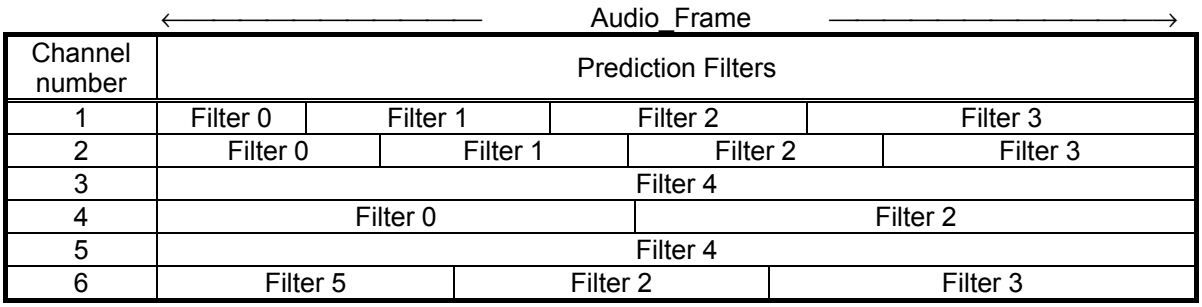

Figure 5-29 : Example of filter allocation

```
                                                          # bits        format
Mapping() {
   Same_Mapping                                             1
   if(Same_Mapping == 0)
   {
       Filter_Mapping()                                                 Maps
       Ptable_Mapping()                                                 Maps
   }
   else
   {
       Filter_And_Ptable_Mapping()                                      Maps
   }
}
```

Figure 5-30 : Syntax of Mapping

Filter_Mapping

For each Audio Channel and each Segment, a Prediction Filter number is encoded. For Filter_Mapping, the variable Element[ ][ ] from Channel_Mapping (see chapter 5.6.3.2.5.2.2) contains the Prediction Filter numbers. For Filter_Mapping, Element[ ][ ] is referred to as:

Filter[Channel_Nr][1..Filters.Nr_Of_Segments[Channel_Nr]].

For Filter_Mapping, the variable Nr_Of_Elements (see chapter 5.6.3.2.5.2.2) is referred to as Nr_Of_Filters.

Ptable_Mapping

For each Audio Channel and each Segment, a Ptable number is encoded. For Ptable_Mapping, the variable Element[ ][ ] from Channel_Mapping (see chapter 5.6.3.2.5.2.2) contains the Ptable numbers. For Ptable_Mapping, Element[ ][ ] is referred to as:

Ptable[Channel_Nr][1..Ptables.Nr_Of_Segments[Channel_Nr]].

For Ptable_Mapping, the variable Nr_Of_Elements (see chapter 5.6.3.2.5.2.2) is referred to as Nr_Of_Ptables.

Filter_And_Ptable_Mapping
For each Audio Channel and each Segment, a common Prediction Filter and Ptable number is encoded. For Filter_and_Ptable_Mapping, the variable Element[ ][ ] from Channel_Mapping (see chapter 5.6.3.2.5.2.2) contains the common Prediction Filter and Ptable numbers. For Filter_and_Ptable_Mapping, Element[ ][ ] is referred to as:

Filter[Channel_Nr][1..Filters.Nr_Of_Segments[Channel_Nr]]

as well as:

Ptable[Channel_Nr][1..Ptables.Nr_Of_Segments[Channel_Nr]].

For Filter_and_Ptable_Mapping, the variable Nr_Of_Elements (see chapter 5.6.3.2.5.2.2) is referred to as Nr_Of_Filters as well as Nr_Of_Ptables.

5.6.3.2.5.1 Same_Mapping

If Same_Mapping is one, the Ptables and Prediction Filters use one and the same Mapping. If Same_Mapping is zero, the mapping for Prediction Filters is independent from the mapping for Ptables, for the Audio Frame.

5.6.3.2.5.2 Maps

Maps defines the mapping of Prediction Filters and Ptables to the Segments defined in chapter 5.6.3.2.4. The syntax of Maps is defined in Figure 5-31.
The following variables are used in the syntax of Maps:

- For each Audio Channel, Element[Channel_Nr][1..Nr_Of_Segments[Channel_Nr]]
- Nr_Of_Elements

```
                                                    # bits      format
Maps() {
   Nr_Of_Elements = 0
   Same_Maps_For_All_Channels                          1
   if(Same_Maps_For_All_Channels == 0)
   {
       for (Channel_Nr=1; Channel_Nr<=N_Channels;
                                  Channel_Nr++)
       {
           Channel_Mapping()[Channel_Nr]                          Channel_Mapping
       }
   }
   else
   {
       Channel_Mapping()[1]                                       Channel_Mapping
   }
}
```

Figure 5-31 : Syntax of Maps

Nr_Of_Elements is the total number of Prediction Filter and/or Ptables, used in Maps. Nr_Of_Elements must be in the range 1 .. 2 * N_Channels.
Channel_Nr is a local index variable, used in Figure 5-31 and Figure 5-32.
N_Channels is the number of audio channels used. The value of N_Channels is given in the Area_TOC_0, see chapter 3.2.1.2.

5.6.3.2.5.2.1 Same_Maps_For_All_Channels

If Same_Maps_For_All_Channels is one, only Element[1][ ] is stored and each Audio Channel uses the same array Element[Channel_Nr][ ] = Element[1][ ]. If Same_Maps_For_All_Channels is equal to zero, Element[Channel_Nr][ ] is stored independent for each Audio Channel. If Nr_Of_Segments[Channel_Nr] does not have the same value for all Audio Channels, Same_Maps_For_All_Channels must be zero.

5.6.3.2.5.2.2 Channel_Mapping

Channel_Mapping contains per Audio Channel the Prediction Filter and/or Ptable numbers used for each Segment. The syntax of Channel_Mapping is defined in Figure 5-32.

```
                                                                  # bits    format
Channel_Mapping() {
   for(Seg_Nr=1; Seg_Nr<=Nr_Of_Segments[Channel_Nr]; Seg_Nr++)
   {
       Element[Channel_Nr][Seg_Nr]                                 0..4    UiMsbf
       if (Element[Channel_Nr][Seg_Nr] == Nr_Of_Elements)
       {
           Nr_Of_Elements++
       }
   }
}
```

Figure 5-32 : Syntax of Channel_Mapping

Nr_Of_Elements is the total number of Prediction Filters and/or Ptables for all channels. Nr_Of_Elements is initialized in Maps, see Figure 5-31.
Channel_Nr is the index variable from Maps, see Figure 5-31.
Seg_Nr is a local index variable.
Nr_Of_Segments[Channel_Nr] is the total number of Segments used in the current Audio Frame for Audio Channel Channel_Nr.

5.6.3.2.5.2.2.1 Element

Element is the Prediction Filter and/or Ptable number used in the Segment. The number of bits used to encode Element depends on the value of Nr_Of_Elements. In each iteration, Element must be <= Nr_Of_Elements. Therefore, Element[1][1] is always zero and is not stored (#bits = 0). For all other Audio Channels and Segments, Nr_Of_Elements > 0 and the number of bits needed to store Element is n with: $2^{n-1} \leq Nr\_Of\_Elements < 2^n$, see Figure 5-33.

Nr_Of_Elements	#bits used
0	0
1	1
2..3	2
4..7	3
8..12	4

Figure 5-33 : Bits used to encode Element

5.6.3.2.6 Half_Probability

The syntax of Half_Probability is defined in Figure 5-34.

```
                                                                # bits    format
Half_Probability() {
   for (Channel_Nr=1; Channel_Nr<=N_Channels; Channel_Nr++)
   {
       Half_Prob[Channel_Nr]                                      1      BsMsbf
   }
}
```

Figure 5-34 : Syntax of Half_Probability

Channel_Nr is a local index variable.
N_Channels is the number of audio channels used. The value of N_Channels is given in the Area_TOC_0, see chapter 3.2.1.2.

5.6.3.2.6.1 Half_Prob

Half_Prob is used to encode, for each Audio Channel, which method will be used for applying a probability value to the arithmetic decoder. The definition of Half_Prob is given in Figure 5-35.

Half_Prob[ ]	Probability to use during first Pred_Order[ ] bits of the Audio Channel
0	Use the entries from the Ptable.
1	Use p=½ (corresponds to P_one = 128).

Figure 5-35 : Definition of Half_Prob

For optimum coding gain it is desired that the next residual bit in E has the value that has the greatest probability. If a probability is applied that reflects a high chance of the next E bit being 1 while the next E bit is a 0, then more than 1 bit is required in the arithmetic code to send this bit.
The prediction filter is initially filled with an initialization pattern. During the first Pred_Order[Filter[Channel_Nr][1]] samples in Audio Channel Channel_Nr, the Prediction Filter is gradually filled with real DSD data. As a consequence the probability distribution can be quite different from the rest of the frame, and the combination of applied E and P for these bits results in more bits than desired. When applying a probability of ½ during encoding, each bit will also cost only one bit in the arithmetic code.
Therefore Half_Prob is available for each channel separately to be able to overrule a bad combination of E and P at the start of a frame.

5.6.3.2.7 Filter_Coef_Sets

For each Segment in each Audio Channel, the DST decoder uses a Prediction Filter. In case two or more Prediction Filters are equal, the corresponding filter coefficients may be encoded only once. The variables used in the syntax of Filter_Coef_Sets are:

- Pred_Order[Filter_Nr]
- Coef[Filter_Nr][0..Pred_Order[Filter_Nr]-1]

With Filter_Nr = 0..Nr_Of_Filters-1.
All Prediction Filter coefficients are encoded in the disc. Per Prediction Filter, the order of prediction (the number of coefficients) and the coefficients are encoded. The Prediction Filter coefficients can be coded using simple linear prediction and Rice coding. Rice coding is a variable length coding technique (special case of Huffman coding) which is used to decrease the number of bits required for a certain “message”, without loss of information. The syntax of Filter_Coef_Sets is defined in Figure 5-36.
The least significant bit of Coef[0][0] is called DST_Y_Bit, see chapter 5.6.3.2.7.1.

```
                                                                  # bits    format
Filter_Coef_Sets() {
   for (Filter_Nr=0; Filter_Nr<Nr_Of_Filters; Filter_Nr++)
   {
       Coded_Pred_Order                                             7      UiMsbf
       Pred_Order[Filter_Nr]=Coded_Pred_Order+1
       Coded_Filter_Coef_Set                                        1      BsMsbf
       if (Coded_Filter_Coef_Set==0)
       {
           for (Coef_Nr=0; Coef_Nr<Pred_Order[Filter_Nr];
                   Coef_Nr++)
           {
               Coef[Filter_Nr][Coef_Nr]                             9      SiMsbf
           }
       }
       else
       {
           CC_Method                                                2      BsMsbf
           for (Coef_Nr=0; Coef_Nr<CCPO; Coef_Nr++)
           {
               Coef[Filter_Nr][Coef_Nr]                             9      SiMsbf
           }
           CCM                                                      3      UiMsbf
           for (Coef_Nr=CCPO; Coef_Nr<Pred_Order[Filter_Nr];
                                                  Coef_Nr++)
           {
               Run_Length=0
               do
               {
                   RL_Bit                                           1      BsMsbf
                   if (RL_Bit==0)
                   {
                       Run_Length++
                   }
               } while (RL_Bit==0)
               LSBs                                               0..6     UiMsbf
               Delta=(Run_Length<<CCM)+LSBs
               if (Delta!=0)
               {
                   Sign                                             1      BsMsbf
                   if (Sign==1)
                   {
                       Delta = -Delta
                   }
               }
               Coef[Filter_Nr][Coef_Nr] = Delta
               Delta8 = 0
               for (Tap_Nr=0; Tap_Nr<CCPO; Tap_Nr++)
               {
                   Delta8 += 8*CCPC[Tap_Nr]*Coef[Filter_Nr][Coef_Nr-Tap_Nr-1]
               }
               if (Delta8>=0)
               {
                   Coef[Filter_Nr][Coef_Nr] -= trunc((Delta8+4)/8)
               }
               else
               {
                   Coef[Filter_Nr][Coef_Nr] += trunc((-Delta8+3)/8)
               }
           }
       }
   }
}
```

Figure 5-36 : Syntax of Filter_Coef_Sets

Nr_Of_Filters is the value calculated in Mapping, see chapter 5.6.3.2.5.
Pred_Order[ ] is an array that contains the prediction order for each Prediction Filter. Where Pred_Order[Filter_Nr] = Coded_Pred_Order + 1, for Filter_Nr∈(0..Nr_Of_Filters-1). The allowed range of the prediction order is: 1 ≤ Pred_Order[Filter_Nr] ≤ 128.
Coef[ ][ ] is a two-dimensional array that contains all coefficients of all Prediction Filters. Each entry of Coef[ ][ ] must be in the range of -256 to +255. The first (left) index is the Filter_Nr and ranges from 0 through Nr_Of_Filters-1. The second (right) index is the coefficient number and ranges from 0 through Pred_Order[Filter_Nr]-1.
CCPO is the Coefficient Coding Prediction Order (CCPO). The relation between CC_Method and CCPO is defined in Figure 5-37.

CC_Method	CCPO
'00'	1
'01'	2
'10'	3
'11'	Not used

Figure 5-37 : Relation between CC_Method and CCPO

The restriction CCPO < Pred_Order[Filter_Nr] applies.
Run_Length is a help variable to count the number of zeros in the run length code that is part of the Rice code.
Delta is a help variable to calculate the Rice coded Number.
Delta8 is a help variable to calculate the Rice coded Number.
CCPC[] is an array that contains the Coefficient Coding Prediction Coefficients (CCPC) that are used for the linear prediction of the Filter coefficients. The relation between CC_Method and CCPC[] is defined in Figure 5-38.

CC_Method	CCPC[0]	CCPC[1]	CCPC[2]
'00'	-1	-	-
'01'	-2	1	-
'10'	-9/8	-5/8	6/8
'11'	Not used	Not used	Not used

Figure 5-38 : Relation between CC_Method and CCPC[]

The linear prediction requires rounding, as specified in the syntax of Figure 5-36.

5.6.3.2.7.1 DST_Y_Bit

DST_Y_Bit is the least significant bit of Coef[0][0]. The DST_Y_Bit must be set to 1 in discs according to this version of the specification. It must be taken into account that the DST_Y_Bit may also be set to zero in later versions of the Super Audio CD specification. Players according to this version of the Super Audio CD Specification shall ignore the content of the DST_Y_Bit.

5.6.3.2.7.2 Coded_Pred_Order

Coded_Pred_Order is a 7 bit unsigned integer that contains the coded prediction order of the current Prediction Filter.

5.6.3.2.7.3 Coded_Filter_Coef_Set

Coded_Filter_Coef_Set indicates whether the Prediction Filter coefficients are predicted and Rice coded. Coded_Filter_Coef_Set is set to zero if the Prediction Filter coefficients are stored directly. Coded_Filter_Coef_Set is set to one if the Prediction Filter coefficients are predicted and Rice coded.

The maximum number of bits permitted for a single Prediction Filter inside Filter_Coef_Sets is:

7+1+Pred_Order[ ]*9

where 7 counts the bits for Coded_Pred_Order, 1 counts the Coded_Filter_Coef_Set bit and the rest the Pred_Order[] coefficients of 9 bit each.

5.6.3.2.7.4 CC_Method

CC_Method is a 2 bit code that identifies the Coefficient Coding Method of the current Prediction Filter.

5.6.3.2.7.5 CCM

CCM is a 3 bit unsigned integer that contains the Coefficient Coding M parameter that is used for Rice decoding the coefficients of the current Prediction Filter. The minimum allowed value for CCM is zero. The maximum allowed value for CCM is 6.

5.6.3.2.7.6 RL_Bit

RL_Bit is used to retrieve the single bits of the run length code, that consists of zeros with a terminating one. The shortest run length code is ‘1’.

5.6.3.2.7.7 LSBs

CCM least significant bits of the absolute value of the predicted coefficient are read from the stream directly and are stored in LSBs.

5.6.3.2.7.8 Sign

Sign is a bit that indicates if the predicted coefficient is positive (Sign=‘0’) or negative (Sign=‘1’).

5.6.3.2.8 Probability_Tables

For each Segment in each Audio Channel, the decoder uses a Probability Table (Ptable). In case two or more probability tables are equal, the corresponding probability table entries may be available from the stream only once. The variables used in the syntax of Probability_Tables are:

- Ptable_Len[Ptable_Nr]
- P_One[Ptable_Nr][0.. Ptable_Len[Ptable_Nr]-1]

With Ptable_Nr = 0..Nr_Of_Ptables-1.
In Probability_Tables all probability table entries are encoded. Per probability table, the length of the table (= the number of entries) and the entries are encoded. The Ptable entries can be coded using simple linear prediction and Rice coding. The syntax of Probability_Tables is defined in Figure 5-39.

```
                                                                  # bits    format
Probability_Tables() {
   for (Ptable_Nr=0; Ptable_Nr<Nr_Of_Ptables; Ptable_Nr++)
   {
       Coded_Ptable_Len                                             6      UiMsbf
       Ptable_Len[Ptable_Nr] = Coded_Ptable_Len+1
       if (Ptable_Len[Ptable_Nr] == 1)
       {
           P_one[Ptable_Nr][0] = 128
       }
       else
       {
           Coded_Ptable                                             1      BsMsbf
           if (Coded_Ptable==0)
           {
               for (Entry_Nr=0; Entry_Nr<Ptable_Len[Ptable_Nr];
                     Entry_Nr++)
               {
                   Coded_P_one                                      7      UiMsbf
                   P_one[Ptable_Nr][Entry_Nr] = Coded_P_one+1
               }
           }
           else
           {
               PC_Method                                            2      BsMsbf
               for (Entry_Nr=0; Entry_Nr<PCPO; Entry_Nr++)
               {
                   Coded_P_one                                      7      UiMsbf
                   P_one[Ptable_Nr][Entry_Nr] = Coded_P_one+1
               }
               PCM                                                  3      UiMsbf
               for (Entry_Nr=PCPO;Entry_Nr<Ptable_Len[Ptable_Nr];
                       Entry_Nr++)
               {
                   Run_Length=0
                   do
                   {
                       RL_Bit                                       1      BsMsbf
                       if (RL_Bit==0)
                       {
                           Run_Length++
                       }
                   } while (RL_Bit==0)
                   LSBs                                           0..4     UiMsbf
                   Delta = (Run_Length<<PCM)+LSBs
                   if (Delta != 0)
                   {
                       Sign                                         1      BsMsbf
                       if (Sign==1)
                       {
                           Delta = - Delta
                       }
                   }
                   P_one[Ptable_Nr][Entry_Nr] = Delta
                   for (Tap_Nr=0; Tap_Nr<PCPO; Tap_Nr++)
                   {
                       P_one[Ptable_Nr][Entry_Nr] -=
                           PCPC[Tap_Nr]*P_one[Ptable_Nr][Entry_Nr-Tap_Nr-1]
                   }
               }
           }
       }
   }
}
```

Figure 5-39 : Syntax of Probability_Tables

Nr_Of_Ptables is the value calculated in Mapping, see chapter 5.6.3.2.5.
Ptable_Len[ ] is an array that contains the probability table length for each Ptable. Where Ptable_Len[Ptable_Nr] = Coded_Ptable_Len + 1, for Ptable_Nr∈{0..Nr_Of_Ptables-1}. The allowed range of Ptable length: 1 ≤ Ptable_Len[Ptable_Nr] ≤ 64.
P_one[ ][ ] is a two-dimensional array that finally contains all entries of all probability tables. The first (left) index is the Ptable_Nr and ranges from 0 through Nr_Of_Ptables-1. The second (right) index is the entry number and ranges from 0 through Ptable_Len[Ptable_Nr]-1. Each entry of P_one[ ][ ] is in the range of 1 to 128, corresponding to a probability of 1/256 to 128/256 of the next error bit (bit E, See Figure C-1) being a one.
PCPO is the Ptable Coding Prediction Order (PCPO). The relation between PC_Method and PCPO is defined in Figure 5-40.

PC_Method	PCPO
'00'	1
'01'	2
'10'	3
'11'	Not used

Figure 5-40 : Relation between PC_Method and PCPO

The restriction PCPO < Ptable_Len[Ptable_Nr] applies.
Run_Length is a help variable to count the number of zeros in the run length code that is part of the Rice code.
Delta is a help variable to calculate the Rice decoded Number.
PCPC[] is an array that contains the Ptable Coding Prediction Coefficients (PCPC) that are used for the linear prediction of the Ptable entries. The relation between PC_Method and PCPC[] is defined in Figure 5-41.

PC_Method	PCPC[0]	PCPC[1]	PCPC[2]
'00'	-1	-	-
'01'	-2	1	-
'10'	-3	3	-1
'11'	Not used	Not used	Not used

Figure 5-41 : Relation between PC_Method and PCPC[]

5.6.3.2.8.1 Coded_Ptable_Len

Coded_Ptable_Len is a 6 bit unsigned integer that contains the coded probability table length.

5.6.3.2.8.2 Coded_Ptable

Coded_Ptable indicates whether the Ptable entries are predicted and Rice coded. Coded_Ptable is set to zero if the Ptable entries are stored directly. Coded_Ptable is set to one if the Ptable entries are predicted and Rice coded.

The maximum number of bits permitted for a single Ptable inside Probability_Tables is:

6+1+Ptable_Len[ ]*7

where 6 counts the bits for Coded_Ptable_Len, 1 counts the Coded_Ptable bit and the rest the Ptable_Len[] coded Ptable entries of 7 bit each.

5.6.3.2.8.3 Coded_P_one

Coded_P_one is a 7 bit unsigned integer that contains the coded value of the next entry of the current Ptable.

5.6.3.2.8.4 PC_Method

PC_Method is a 2 bit field that identifies the Ptable Coding Method of the current Ptable.

5.6.3.2.8.5 PCM

PCM is a 3 bit unsigned integer that contains the Ptable Coding M parameter that is used for Rice decoding of the Ptable entries of the current Ptable. The minimum allowed value for PCM is zero. The maximum allowed value for PCM is 4.

5.6.3.2.8.6 RL_Bit

RL_Bit contains the single bits of the run length code, that consists of zeros with a terminating one. The shortest run length code is '1'.

5.6.3.2.8.7 LSBs

PCM least significant bits of the absolute value of the predicted entry are stored in LSBs.

5.6.3.2.8.8 Sign

Sign is a bit that indicates if the predicted entry is positive (Sign='0') or negative (Sign='1').

5.6.3.2.9 Arithmetic_Coded_Data

The syntax of Arithmetic_Coded_Data is defined in Figure 5-42.

```
                                                        # bits      format
Arithmetic_Coded_Data() {
   j=0
   do
   {
       A_Data[j]                                          1        BsMsbf
       j++
   } until end of Audio_Frame
}
```

Figure 5-42 : Syntax of Arithmetic_Coded_Data

Note that the length of Arithmetic_Coded_Data is not encoded.

5.6.3.2.9.1 A_Data

A_Data[] contains two parts:

- the Arithmetic Code
- Stuffing Bits

The Stuffing Bits are appended at the end of the Arithmetic Code to align the Audio_Frame to a byte boundary. The number of Stuffing Bits is 0..7. The value of the Stuffing Bits must be zero.
A_Data[] is used by the function "Input next bit D" as described in Annex C. The minimum length of A_Data is zero bits. If the length of A_Data is not equal to zero, A_Data[0] must have the value zero. The maximum length of Arithmetic Code is the number of bits processed by "Input next bit D". It is allowed that trailing zeros of Arithmetic Code are not encoded in A_Data[].

5.7 Supplementary Data Stream

A Supplementary Data Stream can contain additional information that is multiplexed with the audio information. A Supplementary Data Stream is the concatenation of all Supplementary Data Frames (see chapter 5.2) in an Audio Area. Both the 2-Channel Stereo Area and the Multi Channel Area contain one Supplementary Data Stream each.

5.7.1 Supplementary_Data_Frame

Every Supplementary_Data_Frame has an associated Time Code (see chapter 5.3.1.1.5.1). A Supplementary_Data_Frame contains an integer number of Supplementary Data Units (SD_Unit). The maximum allowed length of a Supplementary_Data_Frame is 1500 bytes. The syntax of a Supplementary_Data_Frame is defined in Figure 5-43.

	# bytes	format	value
Supplementary_Data_Frame(){			
for (i=1; i<=N; i++)			
{			
SD_Unit()		SD_Unit	
}			
SD_Stuffing	until end of Supplementary Data Frame	Uint8	0
}			

Figure 5-43 : Syntax of Supplementary Data Frame

N is the number of SD_Units in a Supplementary_Data_Frame.

5.7.1.1 SD_Unit

The syntax of an SD_Unit is defined in Figure 5-44. All SD_Units in a Supplementary_Data_Frame are associated with the Time Code of that Supplementary_Data_Frame. The length of an SD_Unit must be an even number of bytes.

```
                                                # bits          format          value
SD_Unit(){
   SD_Type                                        3            Uint3
   IU_Start                                       1            IU_Start
   IU_End                                         1            IU_End
   SD_Length                                     11            Uint11         0..1498
   if (IU_Start == 1)
   {
       Start_PTC                                 24            Time_Code
       End_PTC                                   24            Time_Code

       Immediate_Start                            1            Immediate_Start
       Immediate_End                              1            Immediate_End
       Reserved                                  22            Uint22            0

       if (SD_Type == JPEG_Picture)
       {
           Pic_Sub_Type                           8            Pic_Sub_Type
       }
       else if ((SD_Type == Sub_Picture) || (SD_Type == Text))
       {
           Text_Channel_Mapping                   8            Text_Channel_Mapping
           T_Sub_Type                             2            Uint2
           Page_Nr                                6            Uint6
           P_Ref                                  2            Uint2
           Nm_Pages                               6            Uint6
       }
   }
   SD_Data                                  8*SD_Length        SD_Data
   SD_Unit_Stuffing                           0 or 8           Uint8             0
}
```

Figure 5-44 : Syntax of SD_Unit

5.7.1.1.1 SD_Type

SD_Type defines the global type of the information stored in the SD_Unit. The definition of SD_Type is given in Figure 5-45.

SD_Type	Definition
0	Not Allowed
1	JPEG_Picture
2	Sub_Picture
3	Text
4..7	Reserved

Figure 5-45 : Definition of SD_Type

5.7.1.1.1.1 Not Allowed

A value of zero is not allowed for SD_Type. If the byte at the location of SD_Type is equal to zero, this location belongs to SD_Stuffing of a Supplementary_Data_Frame.

5.7.1.1.1.2 JPEG_Picture

If SD_Type is equal to JPEG_Picture, the concatenation of all SD_Data fields in this Information Unit (see chapter 5.7.1.1.2) must contain one JPEG Picture.
JPEG Pictures in Supplementary Data must use one of the following resolutions: 1440x1080 (hxv) or 1920x1080 (hxv). All JPEG_Pictures and Sub_Pictures in one Track must have the same resolution. It is recommended that all JPEG_Pictures and Sub_Pictures in one Track Area have the same resolution.
Additional restrictions and definitions for JPEG pictures are defined in chapter 7.

5.7.1.1.1.3 Sub_Picture

If SD_Type is equal to Sub_Picture, the concatenation of all SD_Data fields in this Information Unit (see chapter 5.7.1.1.2) must contain one Sub Picture encoded according to the definitions in chapter 8. All JPEG_Pictures and Sub_Pictures in one Track must have the same resolution. It is recommended that all JPEG_Pictures and Sub_Pictures in one Track Area have the same resolution.

5.7.1.1.1.4 Text

If SD_Type is equal to Text, the concatenation of all SD_Data fields in this Information Unit (see chapter 5.7.1.1.2) must contain one Special_String (see chapter 1.7.2.6). The Language Code and the Character_Set_code that must be used for the Special_String is defined by Text_Channel_Mapping (see chapter 5.7.1.1.8).

5.7.1.1.2 Information Unit: IU_Start, IU_End

One Information Unit contains all SD_Data bytes for the information associated with one Start_PTC / End_PTC combination. All SD_Units in one Information Unit must have the same value of SD_Type and the same value of P_Sub_Type or T_Sub_Type. IU_Start must be set to one if the first byte of SD_Data is the first byte of an Information Unit. IU_End must be set to one if the last byte of SD_Data is the last byte of an Information Unit. Information Units with a different value of SD_Type can be interleaved. Each Information Unit can occupy maximum one SD_Unit per Supplementary_Data_Frame. For the definition of IU_Start and IU_End, see Figure 5-46. Figure 5-47 gives an example of the de-interleaving of Information Units.

IU_Start	IU_End	Description
0	0	A continuation SD_Unit, contains SD_Data bytes belonging to one SD_Type. Does not contain the first or the last SD_Data byte of an Information Unit.
1	0	A SD_Unit containing the first SD_Data byte of an Information Unit.
0	1	A SD_Unit containing the last SD_Data byte of an Information Unit.
1	1	A SD_Unit containing both the first and the last SD_Data byte of an Information Unit.

Figure 5-46 : Definition of IU_Start and IU_End

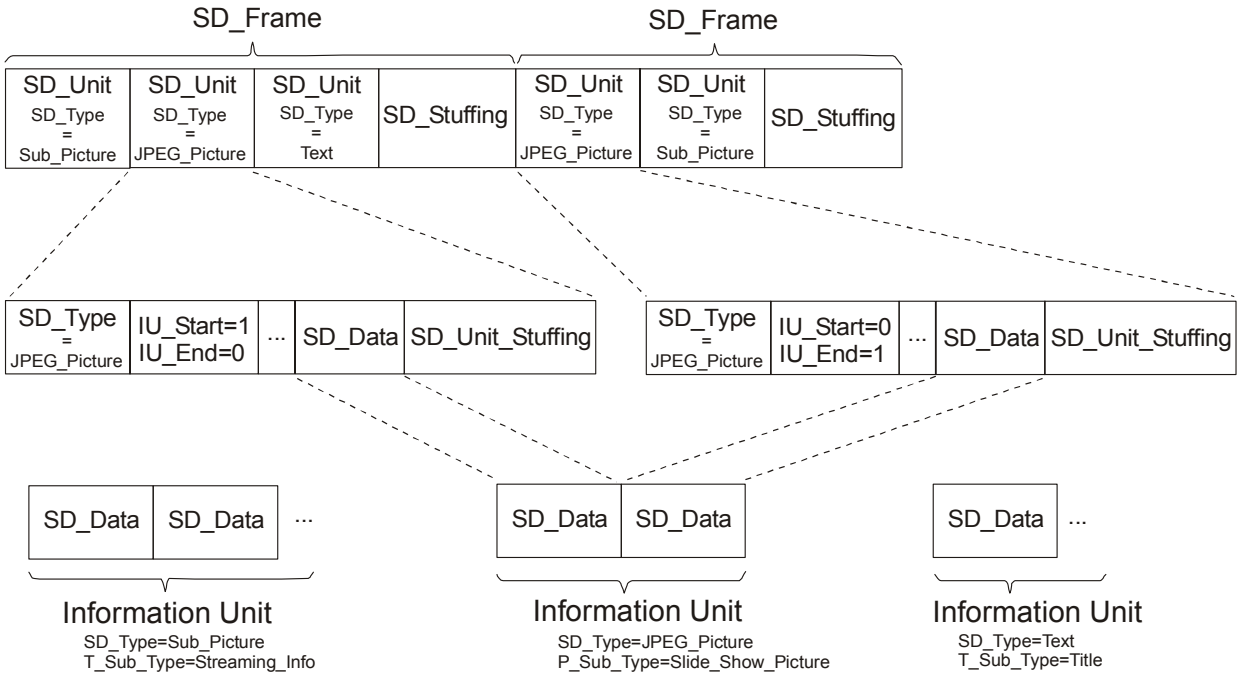

Figure 5-47 : Example of the de-interleaving of Information Units

5.7.1.1.3 SD_Length

SD_Length contains the length of the SD_Data field in bytes. If IU_Start is equal to zero, the minimum value of SD_Length is one. If IU_Start is equal to one, the minimum value of SD_Length is zero.
If SD_Length is even, no SD_Unit_Stuffing byte must be added following SD_Data. If SD_Length is odd, one SD_Unit_Stuffing byte must be added following SD_Data.

5.7.1.1.4 Start_PTC, End_PTC, IU_Start_TC, IU_End_TC

Start_PTC contains the Time Code from which the data in the Information Unit could be displayed, and before which the data in this Information Unit should not be displayed, also see chapter 5.7.1.1.5. End_PTC contains the Time Code at which the presentation of data in the Information Unit should be stopped, also see chapter 5.7.1.1.6. The IU_End_TC of an Information Unit is equal to the Time Code associated with the SD_Unit that contains the last byte of that Information Unit.
The IU_Start_TC of an Information Unit is equal to the Time Code associated with the SD_Unit that contains the first byte of that Information Unit. The IU_Start_TC should be as close as possible to the Start_PTC of that Information Unit. Figure 5-48 shows the relation between IU_Start_TC, IU_End_TC, Start_PTC and End_PTC.

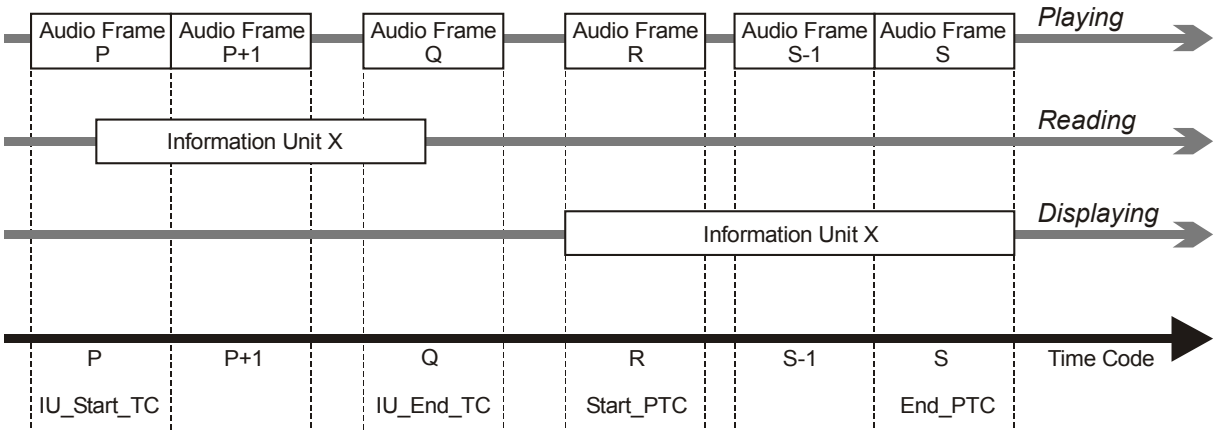

Figure 5-48 : Relation between IU_Start_TC, IU_End_TC, Start_PTC and End_PTC

5.7.1.1.4.1 PreLoad Area

PreLoad Area[tno] is an area before Track_Start_Time_Code[tno] and is used to store Supplementary Data with contents that should be displayed at or close after Track_Start_Time_Code[tno]. The start of PreLoad Area[tno] is given in Track_List_3, see chapter 3.2.9.2.2. The minimum length of the PreLoad Area is 00:00:00. The maximum length of any PreLoad Area is 00:02:00. The size of the PreLoad Area should be as small as possible. All Supplementary Data information in PreLoad Area[tno] shall have a Start_PTC in Track[tno]. The layout of the PreLoad Area is presented in Figure 5-49.

5.7.1.1.4.2 Load Area

The Load Area of a Track consists of the first 5 seconds of a Track. All Pages 1 of the Browsable Text and Browsable Sub-Pictures (see chapter 5.7.1.1.9) with associated JPEG Pictures must be stored in the combined PreLoad Area and Load Area. The layout of the Load Area is presented in Figure 5-49.

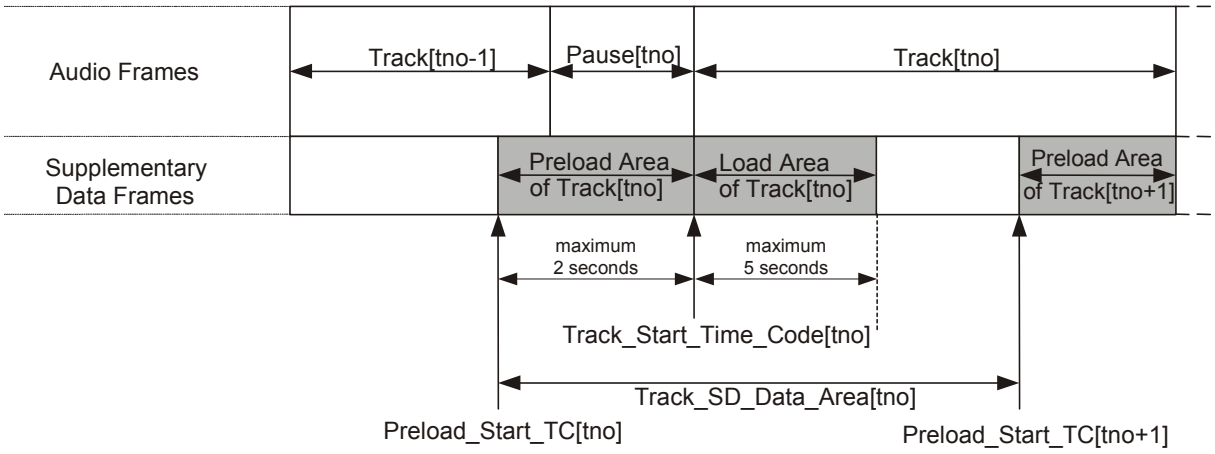

Figure 5-49 : Layout of PreLoad Area and Load Area

5.7.1.1.4.2.1 Track_SD_Data_Area

Track_SD_Data_Area[tno] can contain Browsable Information and other Supplementary Data information for Track[tno]. The area from PreLoad_Start_TC[tno] to PreLoad_Start_TC[tno+1] - 00:00:01 is called the Track_SD_Data_Area[tno]. If tno is equal to the value of N_Tracks, Track_SD_Data_Area[tno] is the area from PreLoad_Start_TC[tno] to Total_Area_Play_Time - 00:00:01.

5.7.1.1.4.3 Index_Load_Area

The Index_Load_Area consists of the first 7 seconds after the Index_Start_TC (see chapter 3.2.7.4) of Index 2 or higher in Track[tno]. It is allowed to repeat the Browsable Information (see chapter 5.7.1.1.9) and Background_Picture_A/_B/_C from the combined PreLoad Area[tno] and Load Area[tno] in the Index_Load_Areas of Track[tno].

5.7.1.1.5 Immediate_Start

Immediate_Start indicates that the Information Unit should be displayed immediately at the Start_PTC or that a transition effect at the Start_PTC is allowed. A value of zero for Immediate_Start means that a transition effect is allowed. If the value of Immediate_Start is equal to one, the presentation of the information should start instantaneously at the time code given by Start_PTC.
Only if one of the following conditions is fulfilled it is allowed to set Immediate_Start to one:

- SD_Type is equal to JPEG_Picture and P_Sub_Type is equal to Slide_Show_Picture
- SD_Type is equal to Text or Sub_Picture and T_Sub_Type is equal to Streaming_Info

Note: Transition effects can for example be a fade-in, or, if the End_PTC of the previous information is close, a cross-over effect.

5.7.1.1.6 Immediate_End

Immediate_End indicates that display of the Information Unit should stop at the End_PTC or that a transition effect at the End_PTC is allowed. A value of zero for Immediate_End means that a transition effect is allowed. If the value of Immediate_End is equal to one, the presentation of the information should end instantaneously at the time code given by End_PTC.
Only if one of the following conditions is fulfilled it is allowed to set Immediate_End to one:

- SD_Type is equal to JPEG_Picture and P_Sub_Type is equal to Slide_Show_Picture
- SD_Type is equal to Text or Sub_Picture and T_Sub_Type is equal to Streaming_Info

Note: Transition effects can for example be a fade-out, or, if the Start_PTC of the information following is close, a cross-over effect.

5.7.1.1.7 Pic_Sub_Type

Pic_Sub_Type gives additional information on the data in the Information Unit in case SD_Type is equal to JPEG_Picture. The format of Pic_Sub_Type is given in Figure 5-50.

b7	b6	b5	b4	b3	b2	b1	b0
Reserved						P_Sub_Type	

Figure 5-50 : Format of Pic_Sub_Type

5.7.1.1.7.1 P_Sub_Type

The definition of P_Sub_Type is given in Figure 5-51.

P_Sub_Type	Definition
0	Slide_Show_Picture
1	Background_Picture_A
2	Background_Picture_B
3	Background_Picture_C

Figure 5-51 : Definition of P_Sub_Type

5.7.1.1.7.2 Slide_Show_Picture

If P_Sub_Type is equal to Slide_Show_Picture, the picture encoded in this Information Unit is part of a series of pictures in a slide show. The Slide_Show_Picture is intended to be displayed from Start_PTC to End_PTC. One Audio Area can contain maximum 2047 Slide_Show_Pictures.
There should not be a gap in the presentation between two successive Slide_Show_Pictures if the difference between the End_PTC of a Slide_Show_Picture and the Start_PTC of the following Slide_Show_Picture is equal to one Frame.
For Slide_Show_Pictures the following restrictions apply:

- The Start_PTC must be higher than the End_PTC of the immediately preceding Slide_Show_Picture.
- Start_PTC[n+1] > Start_PTC[n] + 20 Frames, where n is the sequence number of an Information Unit containing a Slide_Show_Picture.
- The highest allowed value for End_PTC is Total_Area_Play_Time – 00:00:01.
- Start_PTC - IU_Start_TC <= 375 Frames
- The amount of bytes in the Information Units containing Slide_Show Pictures from IU_Start_TC[n] to Start_PTC[n+1] must be less than 1125000 bytes.
- Start_PTC > IU_End_TC + 10 Frames

5.7.1.1.7.3 Background_Picture_A .. C

If P_Sub_Type is equal to Background_Picture_A..C, the picture encoded in this Information Unit is intended as background for text information. Per Track maximum one picture labelled as Background_Picture_A and one labelled as Background_Picture_B and one labelled as Background_Picture_C is allowed. Background_Picture_A..C are valid from Start_PTC to End_PTC.
For Background_Picture_A..C in Track_SD_Data_Area[tno] (see chapter 5.7.1.1.4.2) the following restrictions apply:

- The Start_PTC must be equal to the Track_Start_Time_Code[tno] (see chapter 3.2.3.2).
- If Track[tno+1] exists, Track_Start_Time_Code[tno] + Track_Time_Length[tno] – 00:00:01 <= End_PTC <= Track_Start_Time_Code[tno+1] – 00:00:01
- If Track[tno+1] does not exist, End_PTC = Track_Start_Time_Code[tno] + Track_Time_Length[tno] – 00:00:01
- The total number of bytes in the Information Units containing Background_Picture_A plus Background_Picture_B plus Background_Picture_C must be less than 1500000 bytes for each Track.
- Start_PTC <= IU_End_TC is allowed.

5.7.1.1.8 Text_Channel_Mapping

In case SD_Type is equal to Sub_Picture or Text, Text_Channel_Mapping defines the Text Channels used with the SD_Data in the Information Unit. The format of Text_Channel_Mapping is defined in Figure 5-52.

b7	b6	b5	b4	b3	b2	b1	b0
Ch[8]	Ch[7]	Ch[6]	Ch[5]	Ch[4]	Ch[3]	Ch[2]	Ch[1]

Figure 5-52 : Format of Text_Channel_Mapping

Ch[n]=1 means Text Channel n is used. Ch[n]=0 means Text Channel n is not used. Minimally one of Ch[1] to Ch[8] must be set to one.
The character set and language code associated with the Text Channels are defined by Text_Channels in the Area TOC of the current Audio Area, see chapter 3.2.1.2.15. A value of Ch[n]=1 only is allowed for n <= N_Text_Channels.
If SD_Type is equal to Text, all Text Channels with Ch[n] set to one in a single Text_Channel_Mapping must use one value of Character_Set_Code.
At any Time Code it is allowed to have maximum one reference to each Text Channel for each T_Sub_Type and Page_Nr combination. For each Time_Code, a Text_Channel may only appear once in the Text_Channel_Mapping for each Page and T_Sub_Type combination.

5.7.1.1.9 T_Sub_Type

The definition of T_Sub_Type is given in Figure 5-53.

T_Sub_Type	Definition
0	Streaming_Info
1	Title
2	Lyrics
3	Liner_Notes

Figure 5-53 : Definition of T_Sub_Type

Information with T_Sub_Type in the range of 1..3 is called Browsable Information.

5.7.1.1.9.1 Streaming_Info

If T_Sub_Type is equal to Streaming_Info, the Text or Sub_Picture encoded in this Information Unit is intended to be displayed as a sequence of Special_Strings or Sub_Pictures synchronized with the audio information.
The Start_PTC of this Information Unit must be higher than the End_PTC of the previous Information Unit containing Streaming_Info data.
There should not be a gap in the presentation between two successive Streaming_Info Sub_Pictures if the difference between the End_PTC of a Streaming_Info Sub_Picture and the Start_PTC of the following Streaming_Info Sub_Picture is equal to one Frame.
For Streaming_Info the following restrictions apply:

- The Start_PTC must be higher than the End_PTC of the immediately preceding Streaming_Info with the same SD_Type and the same Text_Channel_Mapping.
- Start_PTC[n+1] > Start_PTC[n] + 3 Frames for Streaming_Info with the same SD_Type and the same Text_Channel_Mapping, where n is the sequence number of an Information Unit containing Streaming_Info.
- The highest allowed value for End_PTC is Total_Area_Play_Time – 00:00:01.
- For Sub_Pictures, Start_PTC - IU_Start_TC <= 75 Frames
- For Text, Start_PTC - IU_Start_TC <= 30 Frames
- The amount of bytes in the Information Units containing Streaming_Info Sub_Pictures from IU_Start_TC[n] to Start_PTC[n+1] must be less than 225000 bytes for each Text Channel.
- The amount of bytes in the Information Units containing Streaming_Info Text from IU_Start_TC[n] to Start_PTC[n+1] must be less than 90000 bytes for each Text Channel.
- Start_PTC > IU_End_TC + 3 Frames.

5.7.1.1.9.2 Title

If T_Sub_Type is equal to Title, the Text or Sub_Picture encoded in this Information Unit is intended to be displayed as one page with title information.
For Title in Track_SD_Data_Area[tno] (see chapter 5.7.1.1.4.2) the following restrictions apply:

- The Start_PTC must be equal to the Track_Start_Time_Code[tno] (see chapter 3.2.3.2).
- The End_PTC must be equal to Track_Start_Time_Code[tno] + Track_Time_Length[tno] – 00:00:01.
- The total number of bytes in the Information Units containing Sub_Pictures with Title plus Lyrics plus Liner_Notes must be less than 7000000 bytes for each Track for each Text Channel.
- The total number of bytes in the Information Units containing Text with Title plus Lyrics plus Liner_Notes must be less than 200000 bytes for each Track for each Text Channel.
- Start_PTC <= IU_End_TC is allowed

5.7.1.1.9.3 Lyrics

If T_Sub_Type is equal to Lyrics, the Text or Sub_Picture encoded in this Information Unit is intended to be displayed as one page with lyrics text.
For Lyrics in Track_SD_Data_Area[tno] (see chapter 5.7.1.1.4.2) the following restrictions apply:

- The Start_PTC must be equal to the Track_Start_Time_Code[tno] (see chapter 3.2.3.2).
- The End_PTC must be equal to Track_Start_Time_Code[tno] + Track_Time_Length[tno] – 00:00:01.
- The total number of bytes in the Information Units containing Sub_Pictures with Title plus Lyrics plus Liner_Notes must be less than 7000000 bytes for each Track for each Text Channel.
- The total number of bytes in the Information Units containing Text with Title plus Lyrics plus Liner_Notes must be less than 200000 bytes for each Track for each Text Channel.
- Start_PTC <= IU_End_TC is allowed

5.7.1.1.9.4 Liner_Notes

If T_Sub_Type is equal to Liner_Notes, the Text or Sub_Picture encoded in this Information Unit is intended to be displayed as one page with liner notes.
For Liner_Notes in Track_SD_Data_Area[tno] (see chapter 5.7.1.1.4.2) the following restrictions apply:

- The Start_PTC must be equal to the Track_Start_Time_Code[tno] (see chapter 3.2.3.2).
- The End_PTC must be equal to Track_Start_Time_Code[tno] + Track_Time_Length[tno] – 00:00:01.
- The total number of bytes in the Information Units containing Sub_Pictures with Title plus Lyrics plus Liner_Notes must be less than 7000000 bytes for each Track for each Text Channel.
- The total number of bytes in the Information Units containing Text with Title plus Lyrics plus Liner_Notes must be less than 200000 bytes for each Track for each Text Channel.
- Start_PTC <= IU_End_TC is allowed

5.7.1.1.10 Page_Nr

Page_Nr contains the Page Number of an Information Unit. The Page Number is the sequence number of this Information Unit for the current T_Sub_Type in the current Track. The value of Page_Nr must be in the range of 1..Nm_Pages.
For each Text_Channel_Mapping, SD_Type, T_Sub_Type combination, the following restrictions apply to the order of Page_Nr on the disc:

- Page_Nr shall start at one in the beginning of a Track_SD_Data_Area.
- For Browsable Information in a Track_SD_Data_Area, Page_Nr shall increment by one if Page_Nr<Nm_Pages; Page_Nr only is allowed to restart at one if Page_Nr=Nm_Pages.
- The value of Page_Nr must be equal to Nm_Pages in the last page of the current Track_SD_Data_Area.

If T_Sub_Type is equal to Streaming_Info or Title, Page_Nr must be equal to one.

5.7.1.1.11 P_Ref

P_Ref contains the P_Sub_Type of the background picture that should be used with the Sub_Picture or Text in this Information Unit. If a P_Sub_Type with the value of P_Ref does not exist in the current Track or in the Preload Area (see chapter 5.7.1.1.4.1) of the current Track, the value of P_Ref must be ignored and a black background should be used by the player. If SD_Type is equal to Text, it is allowed for the player to ignore P_Ref.

5.7.1.1.12 Nm_Pages

Nm_Pages contains the total number of pages for each Text_Channel_Mapping, SD_Type, T_Sub_Type combination in the current Track_SD_Data_Area. A value of zero is not allowed for Nm_Pages. If T_Sub_Type is equal to Streaming_Info or Title, Nm_Pages must be equal to one.

5.7.1.1.13 SD_Data

SD_Data contains SD_Length bytes of information.

5.7.1.1.14 SD_Unit_Stuffing

If SD_Length is not even, eight SD_Unit_Stuffing bits shall be added. The content of SD_Unit_Stuffing bits must be zero.

5.7.1.2 SD_Stuffing

The content of SD_Stuffing bytes must be zero.

5.7.2 Supplementary Data player guidelines

5.7.2.1 Player controls

If a player supports the display of JPEG_Pictures, minimally the following user selections must be available:

- Display the Slide_Show_Picture.

If a player supports the display of Sub_Pictures or Text, minimally the following user selections must be available:

- Display Streaming_Info.
- Display Title.
- Display Lyrics.
- Display Liner_Notes.

5.7.2.2 Display of Background pictures

When the display of either a Sub_Picture or Text is selected and if a background picture can be displayed, the JPEG picture referred to by P_Ref should be used as a background for the selected Text or Sub_Picture.

The last displayed background picture (referenced via P_Ref) should be displayed until minimally one of the following conditions is fulfilled:

- The End_PTC of the displayed background picture is reached.
- A new background picture is selected via a new value of P_Ref.
- The user de-selects the currently displayed Sub_Picture or Text.

6. Extension Area

The presence of the Extension Area is optional. If present, the Extension Area contains the Extension TOC followed by Extension_Data. The Extension TOC describes the actual browsable picture and text information stored in Extension_Data.
The first Sector of the Extension TOC is the first Sector of the Extension Area. For the structure of the Extension Area see chapter 2.2.7.

6.1 Extension_TOC

Extension_TOC describes the pictures and text information stored in the Extension Area. The start location of the Extension_TOC is given in the Master TOC, see chapter 3.1.1.6. The syntax of the Extension_TOC is defined in Figure 6-1.

```
                                    # bytes        format
Extension_TOC() {
   Extension_TOC_0()                  2048         Extension_TOC_0

   2CH_Track_ENodes_Info()           a*2048        Track_ENodes_Info (see 6.1.4)
   MCH_Track_ENodes_Info()           b*2048        Track_ENodes_Info

   Disc_ENode_List()                 c*2048        ENode_List (see 6.1.8)
   2CH_ENode_List()                  d*2048        ENode_List
   MCH_ENode_List()                  e*2048        ENode_List
}
```

Figure 6-1 : Syntax of Extension_TOC

The value of a is equal to 0 or 16. The value of b is equal to 0 or 16. The value of b must be zero if the Multi Channel Area is not present. The value of c is equal to roundup(32*Disc_N_ENodes/2048), see chapter 6.1.1.5. The value of d is equal to roundup(32*2CH_N_ENodes/2048), see chapter 6.1.1.7. The value of e is equal to roundup(32*MCH_N_ENodes/2048), see chapter 6.1.1.9.
Minimally one of Disc_Enode_List, 2CH_ENode_List, MCH_ENode_List must contain a reference to a ENode which points to a Sub_Picture, Text and/or a Main_Picture in Extension_Data..
Note: Additional Lists can be added after MCH_ENode_List in future versions of this specification.

6.1.1 Extension_TOC_0

The syntax of Extension_TOC_0 is defined in Figure 6-2.

```
                                  # bytes          format              value
Extension_TOC_0() {
   E_TOC_Signature                   8             String            "SACDETOC"
   Ext_TOC_Length                    2             Uint16
   Reserved                          4             Uint8                0
   Picture_Flags                     1             Uint8
   Reserved                          1             Uint8

   Text_Channels()                  40             Text_Channels
   Reserved                          8             Uint8                0

/* disc group */
   Disc_N_ENodes                     2             Uint16
   Reserved                          6             Uint8                0
   Disc_ENode_Info()               128             Disc_ENode_Info

/* 2ch group */
   2CH_N_ENodes                      2             Uint16
   2CH_N_Tracks                      1             Uint8              1..255
   Reserved                          1             Uint8                0

/* mch group */
   MCH_N_ENodes                      2             Uint16
   MCH_N_Tracks                      1             Uint8              0..255
   Reserved                          1             Uint8                0

   Reserved                         16             Uint8                0
   Ext_Data_Start_Address            4             Uint32
   Ext_Data_End_Address              4             Uint32
   Reserved                         24             Uint8                0

   E_List_Pointers()                32             E_List_Pointers

   Reserved                    Until 2048          Uint8                0
}
```

Figure 6-2 : Syntax of Extension_TOC_0

6.1.1.1 E_TOC_Signature

E_TOC_Signature is an 8 byte string identifying the Sector with Extension_TOC_0. The value of E_TOC_Signature must be "SACDETOC" ($53 $41 $43 $44 $45 $54 $4F $43).

6.1.1.2 Ext_TOC_Length

Ext_TOC_Length contains the length in Sectors of Extension_TOC.

6.1.1.3 Picture_Flags

The format of Picture_Flags is defined in Figure 6-3.

b7	b6	b5	b4	b3	b2	b1	b0
4x3	16x9	Reserved = 0					

Figure 6-3 : Format of Picture_Flags

6.1.1.3.1 4x3 Bit

The 4x3 Bit must be set to one if the Extension Area contains Pictures (see chapter 6.2.4) and/or Thumbnail Pictures (see chapter 6.2.3) with a picture aspect ratio of 4:3. The 4x3 Bit must be set to zero if the Extension Area does not contain Pictures or Thumbnail Pictures with a picture aspect ratio of 4:3.

6.1.1.3.2 16x9 Bit

The 16x9 Bit must be set to one if the Extension Area contains Pictures (see chapter 6.2.4) and/or Thumbnail Pictures (see chapter 6.2.3) with a picture aspect ratio of 16:9. The 16x9 Bit must be set to zero if the Extension Area does not contain Pictures or Thumbnail Pictures with a picture aspect ratio of 16:9.

6.1.1.4 Text_Channels

Text_Channels contains the definition of the Text Channels, see chapter 1.7.2.8, as used in the Extension Area. If the Extension Area is present, it is recommended that Text_Channels is identical in the Extension Area, in the Master TOC (see chapter 3.1.1.4) and in the Area TOC (see chapter 3.2.1.2.15) of both the 2-Channel Stereo Area and the Multi Channel Area.

6.1.1.5 Disc_N_ENodes

Disc_N_ENodes contains the number of ENode (see chapter 6.1.8.1) structures stored in Disc_ENode_List, see chapter 6.1.5.

6.1.1.6 Disc_ENode_Info

Disc_ENode_Info contains the ENode_Nr of the first ENode structures of an Enode_Group (see chapters 6.1.8.1.1 to 6.1.8.1.3) that describe the contents of the whole disc. The syntax of Disc_ENode_Info is defined in Figure 6-4.

```
                                          # bytes             format        value
Disc_ENode_Info() {
   for (c=1; c<=N_Text_Channels; c++)
   {
       Disc_Jacket_ENode_Nr[c]                 2              ENode_Nr
       Disc_Liner_Notes_ENode_Nr[c]            2              ENode_Nr
       Discography_ENode_Nr[c]                 2              ENode_Nr
       Copyright_Info_ENode_Nr[c]              2              ENode_Nr
       Reserved                                8              Uint8          0
   }
   Reserved                          128-16*N_Text_Channels   Uint8          0
}
```

Figure 6-4 : Syntax of Disc_ENode_Info

The value of N_Text_Channels is equal to the value of N_Text_Channels as used in Text_Channels, see chapters 6.1.1.4 and 1.7.2.8. Each ENode_Nr in Disc_ENode_Info refers to an ENode in Disc_ENode_List (see chapter 6.1.5).

6.1.1.6.1 ENode_Nr

ENode_Nr contains the Index Number of an ENode (see chapter 6.1.8.1) in the ENode_List (see chapter 6.1.8). The Index Number of an ENode is equal to the value of the variable n in Figure 6-7. ENode_Nr is equal to zero or ENode_Nr must contain the Index Number of the first ENode of an ENode Group, see chapter 6.1.8.1.1. A value of zero for ENode_Nr means that no ENodes are avalable for the current field.

6.1.1.6.2 Disc_Jacket_ENode_Nr

Disc_Jacket_ENode_Nr[c] contains the Enode_Nr of the first ENode (see chapter 6.1.8.1) of an Enode_Group with the jacket information of the disc for Text Channel c. All ENodes in the Enode_Group referred to by Disc_Jacket_ENode_Nr[c] must have a TC (see chapter 6.1.8.1.4) equal to $FF:$FF:$FF. A value of zero for Disc_Jacket_ENode_Nr[c] means there is no Enode with jacket information of the disc in Text Channel c.

6.1.1.6.3 Disc_Liner_Notes_ENode_Nr

Disc_Liner_Notes_ENode_Nr[c] contains the Enode_Nr of the first ENode (see chapter 6.1.8.1) of an Enode_Group with the liner notes information of the disc for Text Channel c. All ENodes in the Enode_Group referred to by Disc_Liner_Notes_ENode_Nr[c] must have a TC (see chapter 6.1.8.1.4) equal to $FF:$FF:$FF. A value of zero for Disc_Liner_Notes_ENode_Nr[c] means there is no Enode with liner notes information of the disc in Text Channel c.

6.1.1.6.4 Discography_ENode_Nr

Discography_ENode_Nr[c] contains the Enode_Nr of the first ENode (see chapter 6.1.8.1) of an Enode_Group with the discography information of the disc for Text Channel c. All ENodes in the Enode_Group referred to by Discography_ENode_Nr[c] must have a TC (see chapter 6.1.8.1.4) equal to $FF:$FF:$FF. A value of zero for Discography_ENode_Nr[c] means there is no Enode with discography information of the disc in Text Channel c.

6.1.1.6.5 Copyright_Info_ENode_Nr

Copyright_Info_ENode_Nr[c] contains the Enode_Nr of the first ENode (see chapter 6.1.8.1) of an Enode_Group with the copyright information of the disc for Text Channel c. All ENodes in the Enode_Group referred to by Copyright_Info_ENode_Nr[c] must have a TC (see chapter 6.1.8.1.4) equal to $FF:$FF:$FF. A value of zero for Copyright_Info_ENode_Nr[c] means there is no Enode with copyright information of the disc in Text Channel c.

6.1.1.7 2CH_N_ENodes

2CH_N_ENodes contains the number of ENode structures stored in 2Ch_ENode_List, see chapter 6.1.6.

6.1.1.8 2CH_N_Tracks

2CH_N_Tracks contains the value of the number of Tracks in the 2-Channel Stereo Area. If Area TOC-B is present in the 2-Channel Stereo Area, the content of 2CH_N_Tracks must be equal to the value of N_Tracks (see chapter 3.2.1.2.11) in Area TOC-B. If Area TOC-B is not present in the 2-Channel Stereo Area, the content of 2CH_N_Tracks must be equal to the value of N_Tracks in Area TOC-A.

6.1.1.9 MCH_N_ENodes

MCH_N_ENodes contains the number of ENode structures stored in Mch_ENode_List, see chapter 6.1.7.

6.1.1.10 MCH_N_Tracks

MCH_N_Tracks contains the value of the number of Tracks in the Multi Channel Area. If Area TOC-B is present in the Multi Channel Area, the content of MCH_N_Tracks must be equal to the value of N_Tracks (see chapter 3.2.1.2.11) in Area TOC-B. If Area TOC-B is not present in the Multi Channel Area, the content of MCH_N_Tracks must be equal to the value of N_Tracks in Area TOC-A. If the Multi Channel Area is not present, the value of MCH_N_Tracks must be zero.

6.1.1.11 Ext_Data_Start_Address

Ext_Data_Start_Address is a 4 byte integer containing the Logical Sector Number (LSN) of the first Sector of Extension_Data. The content of Ext_Data_Start_Address must be equal to the content of E_Data_Start_Address in the Master TOC, see chapter 3.1.1.6.

6.1.1.12 Ext_Data_End_Address

Ext_Data_End_Address is a 4 byte integer containing the Logical Sector Number (LSN) of the last Sector of Extension_Data. The content of Ext_Data_End_Address must be equal to the content of E_Data_End_Address in the Master TOC, see chapter 3.1.1.6.

6.1.1.13 E_List_Pointers

E_List_Pointers contains pointers to various lists in the Extension TOC. The value of a pointer gives the Sector offset, from the first Sector of the Extension_TOC, to the first Sector of a list. If an offset value is set to zero, the corresponding list is not present. The syntax of E_List_Pointers is defined in Figure 6-5.

	# bytes	format	value
E_List_Pointers() {			
2CH_Track_ENodes_Info_Ptr	2	Uint16	
MCH_Track_ENodes_Info_Ptr	2	Uint16	
Disc_ENode_List_Ptr	2	Uint16	
2CH_ENode_List_Ptr	2	Uint16	
MCH_ENode_List_Ptr	2	Uint16	
Reserved	22	Uint8	0
}			

Figure 6-5 : Syntax of E_List_Pointers

Additional pointers can be defined in future versions of the specification.

6.1.1.13.1 2CH_Track_ENodes_Info_Ptr

2CH_Track_ENodes_Info_Ptr contains a relative pointer to the first Sector of 2CH_Track_ENodes_Info. The value 2CH_Track_Enodes_Info_Ptr is the offset in Sectors from the start of the Extension_TOC to the first Sector of 2CH_Track_ENodes_Info. A value of zero for 2CH_Track_ENodes_Info_Ptr means 2CH_Track_ENodes_Info is not present.

6.1.1.13.2 MCH_Track_ENodes_Info_Ptr

MCH_Track_ENodes_Info_Ptr contains a relative pointer to the first Sector of MCH_Track_ENodes_Info. The value MCH_Track_Enodes_Info_Ptr is the offset in Sectors from the start of the Extension_TOC to the first Sector of MCH_Track_ENodes_Info. A value of zero for MCH_Track_ENodes_Info_Ptr means MCH_Track_ENodes_Info is not present.

6.1.1.13.3 Disc_ENode_List_Ptr

Disc_ENode_List_Ptr contains a relative pointer to the first Sector of Disc_ENode_List. The value Disc_ENode_List_Ptr is the offset in Sectors from the start of the Extension_TOC to the first Sector of Disc_ENode_List. A value of zero for Disc_ENode_List_Ptr means Disc_ENode_List is not present. If Disc_N_ENodes (see chapter 6.1.1.5) is equal to zero, Disc_ENode_List_Ptr must be equal to zero.

6.1.1.13.4 2CH_ENode_List_Ptr

2CH_ENode_List_Ptr contains a relative pointer to the first Sector of 2CH_ENode_List. The value 2CH_ENode_List_Ptr is the offset in Sectors from the start of the Extension_TOC to the first Sector of 2CH_ENode_List. A value of zero for 2CH_ENode_List_Ptr means 2CH_ENode_List is not present. If 2CH_N_ENodes (see chapter 6.1.1.7) is equal to zero, 2CH_ENode_List_Ptr must be equal to zero.

6.1.1.13.5 MCH_ENode_List_Ptr

MCH_ENode_List_Ptr contains a relative pointer to the first Sector of MCH_ENode_List. The value MCH_ENode_List_Ptr is the offset in Sectors from the start of the Extension_TOC to the first Sector of MCH_ENode_List. A value of zero for MCH_ENode_List_Ptr means MCH_ENode_List is not present. If MCH_N_ENodes (see chapter 6.1.1.9) is equal to zero, MCH_ENode_List_Ptr must be equal to zero.

6.1.2 2CH_Track_ENodes_Info

2CH_Track_ENodes_Info contains a Track_ENodes_Info structure (see chapter 6.1.4) with the ENode_Nr of all Track related information for the 2-Channel Stereo Area.

6.1.3 MCH_Track_Enodes_Info

MCH_Track_ENodes_Info contains a Track_ENodes_Info structure (see chapter 6.1.4) with the ENode_Nr of all Track related information for the Multi Channel Area.

6.1.4 Track_ENodes_Info

Track_ENodes_Info contains the ENode_Nr of all first ENode structures of Enode_Groups that describe the contents of all Tracks in the current Audio Area for all Text Channels. The syntax of Track_ENodes_Info is defined in Figure 6-6.

```
                                                # bytes             format     value
Track_ENodes_Info() {
   for (c=1; c<=N_Text_Channels; c++)
   {
       for (tno=1; tno<=N_Tracks; tno++)
       {
           Track_Title_ENode_Nr[c][tno]            2                ENode_Nr
           Track_Liner_Notes_ENode_Nr[c][tno]      2                ENode_Nr
           Track_Lyrics_ENode_Nr[c][tno]           2                ENode_Nr
           Track_Slide_Show_ENode_Nr[c][tno]       2                ENode_Nr
           Reserved                                8                Uint8        0
       }
       Reserved                              (256-N_Tracks)*16      Uint8        0
   }
   Reserved                               (8-N_Text_Channels)*4096  Uint8        0
}
```

Figure 6-6 : Syntax of Track_ENodes_Info

The value of N_Text_Channels is equal to the value of N_Text_Channels in Text_Channels, see chapters 6.1.1.4 and 1.7.2.8. For the definition of ENode_Nr see chapter 6.1.1.6.1. For 2CH_Track_ENodes_Info, ENode_Nr refers to a ENode from the 2CH_ENode_List, see chapter 6.1.6. For MCH_Track_ENodes_Info, ENode_Nr refers to a ENode from the MCH_ENode_List, see chapter 6.1.7. For 2CH_Track_Enodes_Info the value of N_Tracks is equal to the value of 2CH_N_Tracks (see chapter 6.1.1.8). For MCH_Track_Enodes_Info the value of N_Tracks is equal to the value of MCH_N_Tracks (see chapter 6.1.1.10).

6.1.4.1 Track_Title_ENode_Nr

Track_Title_ENode_Nr[c][tno] contains the ENode_Nr (see chapter 6.1.1.6.1) of the first ENode (see chapter 6.1.8.1) of an ENode Group with the title information for Track[tno] in the current Audio Area for Text Channel c. A value of 0 for Track_Title_ENode_Nr[c][tno] means that Track title information is not available for Track[tno] and Text Channel c.
The ENode referred to by Track_Title_ENode_Nr[c][tno] must have the value of TC set to Track_Start_Time_Code[tno] (see chapter 3.2.3.2). The ENode referred to by Track_Title_ENode_Nr[c][tno] must have the value of Num_Pages set to one.

6.1.4.2 Track_Liner_Notes_ENode_Nr

Track_Liner_Notes_ENodes_Nr[c][tno] contains the ENode_Nr (see chapter 6.1.1.6.1) of the first ENode (see chapter 6.1.8.1) of an ENode Group with the liner notes information for Track[tno] in the current Audio Area for Text Channel c. A value of 0 for Track_Liner_Notes_ENode_Nr[c][tno] means that Track liner notes information is not available for Track[tno] and Text Channel c.
The ENode referred to by Track_Liner_Notes_ENode_Nr[c][tno] must have the value of TC set to Track_Start_Time_Code[tno] (see chapter 3.2.3.2).

6.1.4.3 Track_Lyrics_ENode_Nr

Track_Lyrics_ENode_Nr[c][tno] contains the ENode_Nr (see chapter 6.1.1.6.1) of the first ENode (see chapter 6.1.8.1) of an ENode Group with the lyrics information for Track[tno] in the current Audio Area for Text Channel c. A value of 0 for Track_Lyrics_ENode_Nr[c][tno] means that Track lyrics information is not available for Track[tno] and Text Channel c.
The ENode referred to by Track_Lyrics_ENode_Nr[c][tno] must have the value of TC set to a valid Time Code in Track[tno] or set to $FF:$FF:$FF.

6.1.4.4 Track_Slide_Show_ENode_Nr

Track_Slide_Show_ENode_Nr[c][tno] contains the ENode_Nr (see chapter 6.1.1.6.1) of the first ENode (see chapter 6.1.8.1) of an ENode Group with the slide show information for Track[tno] in the current Audio Area for Text Channel c. A value of 0 for Track_Slide_Show_ENode_Nr[c][tno] means that Track slide show information is not available for Track[tno] and Text Channel c.
The ENode referred to by Track_Slide_Show_ENode_Nr[c][tno] must have the value of TC set to a valid Time Code in Track[tno]. A value of $FF:$FF:$FF is not allowed.

6.1.5 Disc_ENode_List

Disc_ENode_List contains the ENode_List (see chapter 6.1.8) for all text and images with disc related information.

6.1.6 2CH_ENode_List

2CH_ENode_List contains the ENode_List (see chapter 6.1.8) for all text and images with information for the 2-Channel Stereo Area.

6.1.7 MCH_ENode_List

MCH_ENode_List contains the ENode_List (see chapter 6.1.8) for all text and images with information for the Multi Channel Area.

6.1.8 ENode_List

ENode_List contains ENode data for the text and images in the Extension Area. ENode_List has a variable size. The syntax of ENode_List is defined in Figure 6-7.

```
                                          # bytes          format       value
ENode_List() {
   for (n=1; n<=N_ENodes; n++}
   {
       ENode()[n]                           32             ENode
   }
   Reserved                               until eos        Uint8          0
}
```

Figure 6-7 : Syntax of ENode_List

With Disc_ENode_List, N_ENodes is equal to the value of Disc_N_ENodes (see chapter 6.1.1.5). With 2CH_ENode_List, N_ENodes is equal to the value of 2CH_N_ENodes (see chapter 6.1.1.7). With MCH_ENode_List, N_ENodes is equal to the value of MCH_N_ENodes (see chapter 6.1.1.9).

6.1.8.1 ENode

ENode defines one image / text combination in the Extension Area. Each ENode has a fixed size of 32 bytes. The syntax of ENode is defined in Figure 6-8.

```
                                        # bytes        format            value
ENode() {
   Page_Number                             1           Uint8
   Num_Pages                               1           Uint8
   TC                                      3           Time_Code
   Reserved                                3           Uint8               0

   Text_Ptr                                4           Uint32
   Sub_Pic_Ofs                             4           Uint32
   Thumb_Pic_Ofs                           4           Uint32
   Main_Pic_Ofs                            4           Uint32

   Text_Byte_Len                           2           Uint16
   Sub_Pic_Len                             2           Uint16
   Thumb_Pic_Len                           2           Uint16
   Main_Pic_Len                            2           Uint16
}
```

Figure 6-8 : Syntax of ENode

6.1.8.1.1 ENode Group

One ENode Group consists of Num_Pages consecutive ENode structures.

6.1.8.1.2 Page_Number

Page_Number contains the sequence number of this ENode in an ENode Group. The minimum allowed value of Page_Number is one. The value of Page_Number is equal to one in the first ENode of an ENode Group. The value of Page_Number is equal to Num_Pages in the last ENode of an ENode Group. The value of Page_Number must increment by one between two consecutive ENode structures in one Enode Group.

6.1.8.1.3 Num_Pages

Num_Pages contains the total number of ENode structures in this ENode Group. A value of zero is not allowed for Num_Pages.

6.1.8.1.4 TC

TC can contain a Time Code of the Audio that is related with the information pointed to by the ENode. A TC of $FF:$FF:$FF means this ENode does not have a Time Code.

6.1.8.1.5 Text_Ptr

Text_Ptr contains the Byte offset from the start of Extension Area to the first Byte of the Text_Information structure (see chapter 6.2.1.1) referred to by this ENode. If the value of Text_Ptr is equal to zero, Text is not available for this ENode. Text_Ptr in different ENode structures can point to the same Text_Information structure.

6.1.8.1.6 Sub_Pic_Ofs

Sub_Pic_Ofs contains the Sector offset from the start of Extension Area to the first Sector of the Sub_Pic_Information (see chapter 6.2.2.1) referred to by this ENode. If the value of Sub_Pic_Ofs is equal to zero, a Sub Picture is not available for this ENode. Sub_Pic_Ofs in different ENode structures can point to the same Sub_Pic_Information.

6.1.8.1.7 Thumb_Pic_Ofs

Thumb_Pic_Ofs contains the Sector offset from the start of Extension Area to the first Sector of the Thumbnail_Picture (see chapter 6.2.3.1) referred to by this ENode. If the value of Thumb_Pic_Ofs is equal to zero, a Thumbnail Picture is not available for this ENode. Thumb_Pic_Ofs in different ENode structures can point to the same Thumbnail_Picture.

6.1.8.1.8 Main_Pic_Ofs

Main_Pic_Ofs contains the Sector offset from the start of Extension Area to the first Sector of the Main_Picture (see chapter 6.2.4.1) referred to by this ENode. If the value of Main_Pic_Ofs is equal to zero, a Picture is not available for this ENode. Main_Pic_Ofs in different ENode structures can point to the same Main_Picture.

6.1.8.1.9 Text_Byte_Len

Text_Byte_Len contains the length in bytes of the Text information referred to by Text_Ptr.

6.1.8.1.10 Sub_Pic_Len

Sub_Pic_Len contains the length in Sectors of the Sub Picture referred to by Sub_Pic_Ofs.

6.1.8.1.11 Thumb_Pic_Len

Thumb_Pic_Len contains the length in Sectors of the Thumbnail Picture referred to by Thumb_Pic_Ofs.

6.1.8.1.12 Main_Pic_Len

Main_Pic_Len contains the length in Sectors of the Main Picture referred to by Main_Pic_Ofs.

6.2 Extension_Data

Extension_Data contains all Text, Sub_Pictures, Thumbnail Pictures and Main Pictures referred to by the Extension_TOC. The start location and end location of Extension_Data is given in the Master TOC (see chapter 3.1.1.6) and in the Extension_TOC (see chapter 6.1). The syntax of Extension_Data is defined in Figure 6-9.

```
                                          # bytes       format            value
Extension_Data() {
   for (c=1; c<=N_Text_Channels; c++)
   {
       Text_Data()[c]                    p[c]*2048     Text_Data
   }
   for (c=1; c<=N_Text_Channels; c++)
   {
       Sub_Pic_Data()[c]                 q[c]*2048     Sub_Pic_Data
   }
   Thumbnail_Data()                        r*2048      Thumbnail_Data
   Main_Picture_Data()                     s*2048      Main_Picture_Data
}
```

Figure 6-9 : Syntax of Extension_Data

The value of N_Text_Channels is equal to the value of N_Text_Channels in Text_Channels, see chapters 6.1.1.4 and 1.7.2.8. A value of zero is allowed for p[c], q[c], r and s.

6.2.1 Text_Data

Text_Data[c] contains the text referred to by Text_Ptr from all ENode structures for Text Channel c. The syntax of Text_Data[c] is defined in Figure 6-10.

```
                                          # bytes        format            value
Text_Data()[c] {
   for (n=0; n<N_Text_Information; n++)
   {
       Text_Information[c][n]                           Special_String
   }
   Reserved                               until eos      Uint8               0
}
```

Figure 6-10 : Syntax of Text_Data

N_Text_Information is the number of Special_Strings in this Text_Data structure.

6.2.1.1 Text_Information

Text_Information[c][n] contains the actual text data pointed to by the Extension_TOC.

6.2.2 Sub_Pic_Data

Sub_Pic_Data[c] contains the Sub_Pictures referred to by Sub_Pic_Ofs from all ENode structures for Text Channel c. The syntax of Sub_Pic_Data[c] is defined in Figure 6-11. Every Sub_Picture must start at a Sector boundary.

```
                                       # bytes        format              value
Sub_Pic_Data()[c] {
   for (n=0; n<N_Sub_Pic_Information; n++)
   {
       Sub_Pic_Information[c][n]                     Sub_Picture    see chapter 8
       Reserved                        until eos     Uint8                0
   }
}
```

Figure 6-11 : Syntax of Sub_Pic_Data

N_Sub_Pic_Information is the number of Sub_Pictures in this Sub_Pic_Data structure.

6.2.2.1 Sub_Pic_Information

Sub_Pic_Information[c][n] contains the actual data for one Sub_Picture formatted according to the definitions in chapter 8.

6.2.3 Thumbnail_Data

Thumbnail_Data contains the Thumbnail Pictures referred to by Thumb_Pic_Ofs from all ENode structures. The syntax of Thumbnail_Data is defined in Figure 6-12. Every Thumbnail Picture must start at a Sector boundary.

```
                                          # bytes        format            value
Thumbnail_Data() {
   for (n=0; n<N_Thumbnails; n++)
   {
       Thumbnail_Picture[n]                             Thumbnail_Picture
       TD_Stuffing                        until eos     Uint8               0
   }
}
```

Figure 6-12 : Syntax of Thumbnail_Data

N_Thumbnails is the number of Thumbnail Pictures in this Thumbnail_Data structure.

6.2.3.1 Thumbnail_Picture

Every Thumbnail Picture must be a JPEG Picture and must use one of the following resolutions: 180x135 (hxv) or 240x135 (hxv). Additional restrictions and definitions for JPEG pictures are defined in chapter 7.

6.2.3.2 TD_Stuffing

All bytes in TD_Stuffing shall be zero.

6.2.4 Main_Picture_Data

Main_Picture_Data contains the Main Pictures referred to by Main_Pic_Ofs from all ENode structures. The syntax of Main_Picture_Data is defined in Figure 6-13. Every Main Picture must start at a Sector boundary.

```
                                          # bytes        format            value
Main_Picture_Data() {
   for (n=0; n<N_Main_Pictures; n++)
   {
       Main_Picture[n]                                   Main_Picture
       MP_Stuffing                        until eos      Uint8               0
   }
}
```

Figure 6-13 : Syntax of Main_Picture_Data

N_Main_Pictures is the number of Main Pictures in this Main_Picture_Data structure.

6.2.4.1 Main_Picture

Every Main Picture must be a JPEG Picture and must use one of the following resolutions: 1440x1080 (hxv) or 1920x1080 (hxv). Additional restrictions and definitions for JPEG pictures are defined in chapter 7.

6.2.4.2 MP_Stuffing

All bytes in MP_Stuffing shall be zero.

7. JPEG Definitions and Restrictions

7.1 Requirements for JPEG images

JPEG images shall:

- Use 1:1 pixel aspect ratio
- Be "baseline process" “interchange format” compressed images according to ISO/IEC 10918-1
- Use one or three image components
  - Use YCbCr components in case of three image components
  - Use 4:2:2, 4:2:0 or 4:4:4 subsampling in case of three image components
  - Use Y in case of one image component

The use of the DNL marker segments in JPEG images is not allowed.

7.2 Recommendations for JPEG images

It is recommended to generate the JPEG images according to the default colour parameters of EXIF (primary colours, white point, gamma, R’G’B’ to Y’C_bC_r conversion, colour signal range).

7.3 Notes for Super Audio CD players

- Super Audio CD players should not gamma correct the RGB components calculated by linear conversion from Y’C_bC_r nor the Y’C_bC_r signals for output to a display device.
- The R’G’B’ to Y’C_bC_r transformation in EXIF as well as in SDTV is according to CCIR 601.
- The R’G’B’ to Y’C_bC_r transformation in HDTV is according to CCIR 709.
- The colour difference signals Y’C_bC_r have a different range in EXIF and in both the CCIR 601 and CCIR 709 specifications.
- Super Audio CD players may ignore the content of marker segments that are not related to the JPEG baseline process. In particular, the content of the application marker segments may be ignored.
- It is recommended that Super Audio CD players support the RSTm and DRI marker segments.

8. Sub Picture Definitions

The syntax of a Sub_Picture is defined in Figure 8-1.

```
                                        # bytes        format
Sub_Picture()
{
   Sub_Picture_Header()                               Sub_Picture_Header
   Color_LUT()                                        Color_LUT
   Encoded_Sub_Picture()                              Encoded_Sub_Picture
}
```

Figure 8-1 : Syntax of Sub_Picture

8.1 Sub_Picture_Header

The syntax of Sub_Picture_Header is defined in Figure 8-2.

```
                                        # bytes        format                  value
Sub_Picture_Header()
{
   Sub_Picture_Signature                   2          String                   "SP"
   Sub_Picture_Flags                       1          Sub_Picture_Flags
   Reserved                                1          Uint8                     0
}
```

Figure 8-2 : Syntax of Sub_Picture_Header

8.1.1 Sub_Picture_Signature

Sub_Picture_Signature is a 2 byte string identifying the start of a Sub Picture. The value of Sub_Picture_Signature must be "SP" ($53 $50).

8.1.2 Sub_Picture_Flags

The format of Sub_Picture_Flags is given in Figure 8-3.

b7	b6	b5	b4	b3	b2	b1	b0
Sub_Pic_Type				Reserved			

Figure 8-3 : Format of Sub_Picture_Flags

8.1.2.1 Sub_Pic_Type

Sub_Pic_Type defines the Resolution, the Encoding Method, the CLUT Type and the Color Space used for the Sub Picture. The definition of Sub_Pic_Type is given in Figure 8-4.

Sub_Pic_Type	Resolution(hxv)	Encoding Method	CLUT Type	Color Space
0	1920x1080	A	Color_LUT_00	YCbCr
1	1440x1080	A	Color_LUT_00	YCbCr
2..15	Reserved	Reserved	Reserved	Reserved

Figure 8-4 : Definition of Sub_Pic_Type

8.2 Color_LUT

Color_LUT contains the Color Look-Up Table used for this Sub Picture. For this version of the Super Audio CD Specification only Color Look-Up Table type Color_LUT_00 is used. Other Color Look-Up Table types can be defined in future versions of the Super Audio CD Specification. The syntax of Color_LUT_00 is defined in Figure 8-5.

```
                                        # bytes              format
Color_LUT_00()
{
   for(cl=0; cl<4; cl++)
   {
       YCbCrT()[cl]                       4                 YCbCrT
   }
}
```

Figure 8-5 : Syntax of Color_LUT_00

8.2.1 YCbCrT

The syntax of YCbCrT is defined in Figure 8-6.

```
                                        # bytes              format             value
YCbCrT()
{
   P_Y                                    1                 Uint8              0..255
   P_Cb                                   1                 Uint8              0..255
   P_Cr                                   1                 Uint8              0..255
   Transparency                           1                 Uint8              0..128
}
```

Figure 8-6 : Syntax of YCbCrT

8.2.1.1 P_Y

P_Y must contain the Luminance signal "Y" as defined in EXIF.

8.2.1.2 P_Cb

P_Cb must contain the Colour-difference signal "Cb" as defined in EXIF.

8.2.1.3 P_Cr

P_Cr must contain the Colour-difference signal "Cr" as defined in EXIF.

8.2.1.4 Transparency

Transparency defines the Blending ratio between JPEG_Picture and Sub_Picture.
The Blending is calculated according to the following formula:

(Transparency*JPEG_Picture + (128-Transparency)*Sub_Picture)/128

For the calculation of the Blending the 3 least significant bits of Transparency may be ignored by a player.

8.3 Encoded_Sub_Picture

Encoded_Sub_Picture contains Encoded_Sub_Picture_A, the data for the Sub Picture encoded by Encoding Method A. Other Encoding Methods are reserved for future use. The syntax of Encoded_Sub_Picture_A is defined in Figure 8-7.

```
                                                                  #bits      format
Encoded_Sub_Picture_A() {
   Ver=0
   Rep1=0
   if (Sub_Pic_Type == 0)
   {
       Line=1920
   }
   else
   {
       Line=1440
   }
   do
   {
       Hor=0
       Raw=1
       do
       {
           Pixel_Run()
       } while (Hor != Line)
       for (h=0; h<Line; h++)
       {
           if (Ver == 0)
           {
               S_Pix[Ver][h]=Dif_Pix[h]
           }
           else
           {
               S_Pix[Ver][h]=Dif_Pix[h] ^ S_Pix[Ver-1][h]
           }
       }
       Ver++
   } while (Ver != 1080)
   Stuffing_Bits                                                  0..7       BsMsbf
}
```

Figure 8-7 : Syntax of Encoded_Sub_Picture_A

Ver is the index variable indicating the vertical position of S_Pix[][].
Rep1 is a help variable to count the repetition of Repeat1.
Line holds the horizontal resolution defined by Sub_Pic_Type. The value of Line is 1920 or 1440.
Hor is the index variable indicating the horizontal position of Dif_Pix[].
S_Pix[][] is a two-dimensional array that contains cl (see chapter 8.2) of each Sub_Picture pixel. The first (left) index is the vertical position and ranges from 0 through 1079. The second (right) index is the horizontal position and ranges from 0 through Line-1. The value of S_Pix[][] is 0 or 1 or 2 or 3. S_Pix[0][0] is the left-top pixel of Sub_Picture. S_Pix[1079][0] is the left-bottom pixel of Sub_Picture. S_Pix[0][Line-1] is the right-top pixel of Sub_Picture. S_Pix[1079][Line-1] is the right-bottom pixel of Sub_Picture.
Dif_Pix[] is an array that contains the inter-line Exclusive OR values of cl (see chapter 8.2). The index is the horizontal position and ranges from 0 through Line-1. The value of Dif_Pix[] is 0 or 1 or 2 or 3.

8.3.1 Pixel_Run

Pixel_Run defines the variable length codeword and it's interpretation for one run of pixels. The syntax of Pixel_Run is defined in Figure 8-8.

```
                                                                    #bits format
Pixel_Run()
{
   Code_Head=0
   Code_Head_Len=0
   if (Rep1 != 0)
   {
       do
       {
           Dif_Pix[Hor]=0
           Hor++
       } while (Hor != Line)
       Rep1--
   }
   else
   {
       do
       {
           Code_Bit                                                  1   BsMsbf
           Code_Head=(Code_Head<<1) + Code_Bit
           Code_Head_Len++
       } while ((Code_Bit==1) && (Code_Head_Len<8))
       if (Code_Head == 0)
       {
           Trans_Table()
           Dif_Pix[Hor]=Table_Out
           Hor++
           Raw=0
       }
       else if (Code_Head == 2)
       {
           Code_Head2=0
           do
           {
               Code_Bit2                                             1   BsMsbf
               Code_Head2=(Code_Head2<<1) + Code_Bit2
           } while ((Code_Head2!=0) && (Code_Head2!=2) && (Code_Head2!=3))
           if (Code_Head2 == 0)
           {
               do
               {
                   Dif_Pix[Hor]=0
                   Hor++
               } while (Hor != Line)
           }
           else if (Code_Head2 == 2)
           {
               for (i=0; i<128; i++)
               {
                   Dif_Pix[Hor]=0
                   Hor++
               }
               Raw=1
           }
           else
           {
               Repeat2                                               4   UiMsbf
               for (r=0; r<Repeat2; r++)
               {
                   for (i=0; i<128; i++)
                   {
                       Dif_Pix[Hor]=0
                       Hor++
                   }
               }
```

```
            Raw=1
        }
    }
    else if (Code_Head != 255)
    {
        RLength=0
        for (i=0; i<Code_Head_Len-2; i++)
        {
            RLen_Bit                                                    1   BsMsbf
            RLength=(RLength<<1) + RLen_Bit
        }
        RLength += 1<<(Code_Head_Len-2)
        Trans_Table()
        for(l=0; l<RLength; l++)
        {
            Dif_Pix[Hor]=Table_Out
            Hor++
        }
        if ((RLength==127) && (Dif_Pix[Hor-1]!=0))
        {
            Raw=1
        }
        else
        {
            Raw=0
        }
    }
    else
    {
        Repeat1                                                         8   UiMsbf
        Rep1=Repeat1
        do
        {
            Dif_Pix[Hor]=0
            Hor++
        } while (Hor != Line)
        Rep1--
    }
  }
}
```

Figure 8-8 : Syntax of Pixel_Run

Code_Head is a help variable to register the unique header part of the code word. When Code_Head is equal to '10', Code_Head2 follows Code_Head. When Code_Head is equal to '11111111', Repeat1 (see chapter 8.3.1.6) follows Code_Head.
Code_Head_Len is a help variable to count the bit length of Code_Head.
Raw is a help variable to select the calculation method of Table_Out. If Raw is equal to zero, the pixel value is calculated. If Raw is equal to one, the pixel value is taken from the variable length codeword.
Table_Out is a help variable that contains the value for Dif_Pix[Hor]. When Raw is equal to zero and Hor is not equal to zero, Table_Out is calculated in Trans_Table using Trans_Table_Bit1, Trans_Table_Bit2 and Dif_Pix[Hor-1].
Code_Head2 is a help variable used when Code_Head is equal to '10'. Code_Head2 registers the code word following Code_Head.
RLength is a help variable to register the runlength value part of the code word.

8.3.1.1 Code_Bit

Code_Bit is used to retrieve the single bits of Code_Head.

8.3.1.2 Trans_Table

The syntax of Trans_Table is defined in Figure 8-9.

```
                                                                    #bits  format
Trans_Table() {
   if (Raw==1)
   {
       Raw_Pix_Data                                                   2    UiMsbf
       Table_Out=Raw_Pix_Data
   }
   else
   {
       Trans_Table_bit1                                               1    BsMsbf
       if (Trans_Table_bit1 == 0)
       {
           if (Dif_Pix[Hor-1] == 0)
           {
               Table_Out=1
           }
           else
           {
               Table_Out=0
           }
       }
       else
       {
           Trans_Table_bit2                                           1    BsMsbf
           if (Trans_Table_bit2 == 0)
           {
               if ((Dif_Pix[Hor-1]==0) || (Dif_Pix[Hor-1]==1))
               {
                   Table_Out=2
               }
               else
               {
                   Table_Out=1
               }
           }
           else
           {
               if (Dif_Pix[Hor-1] == 3)
               {
                   Table_Out=2
               }
               else
               {
                   Table_Out=3
               }
           }
       }
   }
}
```

Figure 8-9 : Syntax of Trans_Table

8.3.1.2.1 Raw_Pix_Data

Raw_Pix_Data is a 2 bit code that is used for the value of Dif_Pix[].

8.3.1.2.2 Trans_Table_Bit1, Trans_Table_Bit2

Trans_Table_Bit1 and Ttans_Table_Bit2 are used to determine the value of Table_Out by using Dif_Pix[Hor-1].

8.3.1.3 Code_Bit2

Code_Bit2 is used to retrieve the single bits of Code_Head2.

8.3.1.4 Repeat2

Repeat2 is a 4 bit unsigned integer that contains the repetition time of 128 pixel run of Dif_Pix[]=0. Repeat2 ranges from 2 through 14. Repeated Runs must not go beyond the end of a pixel line.

8.3.1.5 RLen_Bit

RLen_Bit is used to retrieve the single bits of RLength.

8.3.1.6 Repeat1

Repeat1 is a 8 bit unsigned integer that contains the repetition time of "Until End Of Line" run of Dif_Pix[]=0. Repeat1 ranges from 6 through 255.

8.3.2 Stuffing_Bits

Stuffing_Bits are appended at the end of Encoded_Sub_Picture_A until a byte boundary. All bits of Stuffing_Bits must be zero.

Annex A: Play Back Requirements (Normative)

This Annex contains requirements for Super Audio CD players.

A.1 Player Reference Model

The global block diagram of a Super Audio CD Reference Player is given in Figure A-1. This description of the Super Audio CD player reference model consists of:

- Super Audio CD player block diagram
- Description of each block

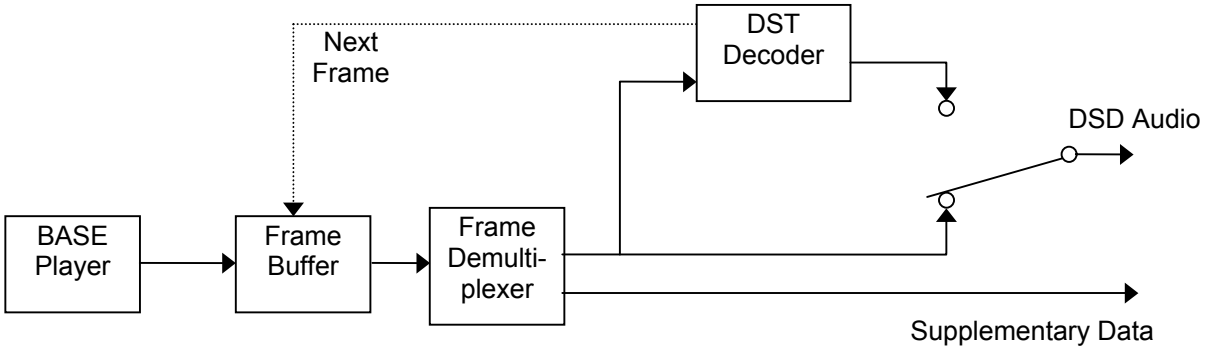

Figure A-1 : Global Block Diagram of the Super Audio CD Reference Player

BASE Player:
The BASE Player is responsible for functions like:

- Optical readout
- Servo Systems
- Demodulation
- Error Correction
- Delivering a stream of Sectors with a constant rate.

Part 3
Every Super Audio CD player must fulfill the requirements defined in Super Audio CD Part 3.

Frame Buffer:
The Frame Buffer accumulates the disc data into Frames, which will be passed to the Frame Demultiplexer at a rate of 75 Frames per second.

Frame Demultiplexer:
The Frame Demultiplexer separates the Elementary Frames from a Multiplexed Frame.

DST Decoder:
The DST Decoder decodes the DST Coded Audio Frames into a Plain DSD Audio Stream.

Fade-in and fade-out times:
It is recommended that both the fade-in and the fade-out times of a Super Audio CD player are less then 50 mS.

A.2 Supplementary Data Buffer Model

The player model for demultiplexing and buffering Supplementary Data is given in Figure A-2.

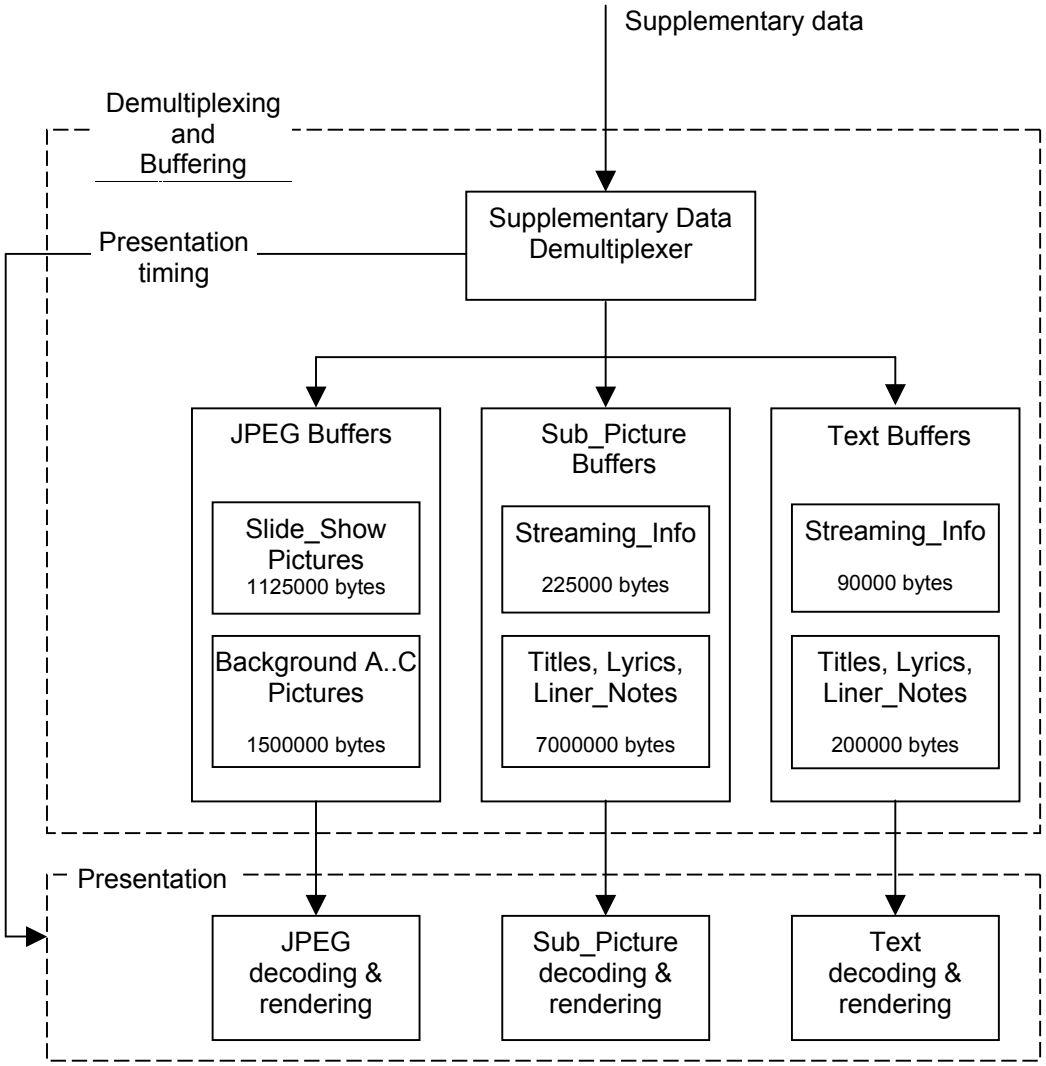

Figure A-2 : Supplementary Data Buffer Model

Annex B: General Genre Table (Normative)

For the General Genre Table, the values for Genre_Index are defined in Figure B-1.

Genre_Index	Definition
0	Reserved
1	Not defined
2	Adult Contemporary
3	Alternative Rock
4	Children's Music
5	Classical
6	Contemporary Christian
7	Country
8	Dance
9	Easy Listening
10	Erotic
11	Folk
12	Gospel
13	Hip Hop
14	Jazz
15	Latin
16	Musical
17	New Age
18	Opera
19	Operetta
20	Pop Music
21	RAP
22	Reggae
23	Rock Music
24	Rhythm & Blues
25	Sound Effects
26	Sound Track
27	Spoken Word
28	World Music
29	Blues
30.. 65535	Reserved

Figure B-1 : Definition of Genre_Index

Annex C: DST Decoder Reference Model (Normative)

C.1 DST Decoder Block Diagrams

In Figure C-1, the block diagram of a DST Decoder for a mono DSD signal is given. Figure C-2 is a short hand notation for Figure C-1. Figure C-3 shows the diagram that is applicable when C-channels plus the DST_X_Bit have to be decoded.

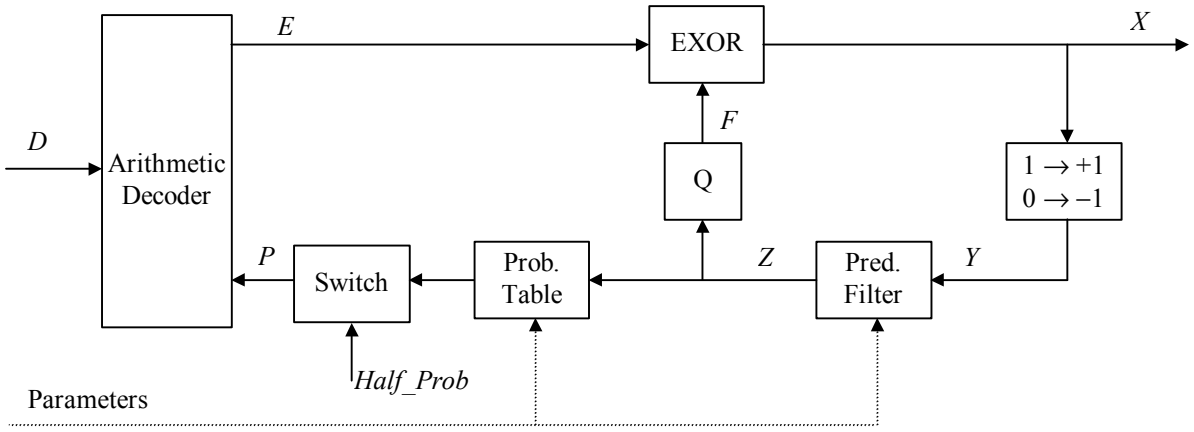

Figure C-1 : Block diagram of the DST decoder for a mono DSD signal

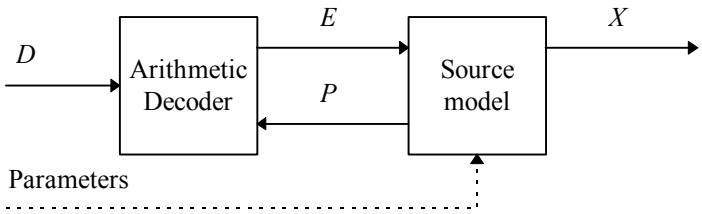

Figure C-2 : Global diagram of the single channel DST decoder

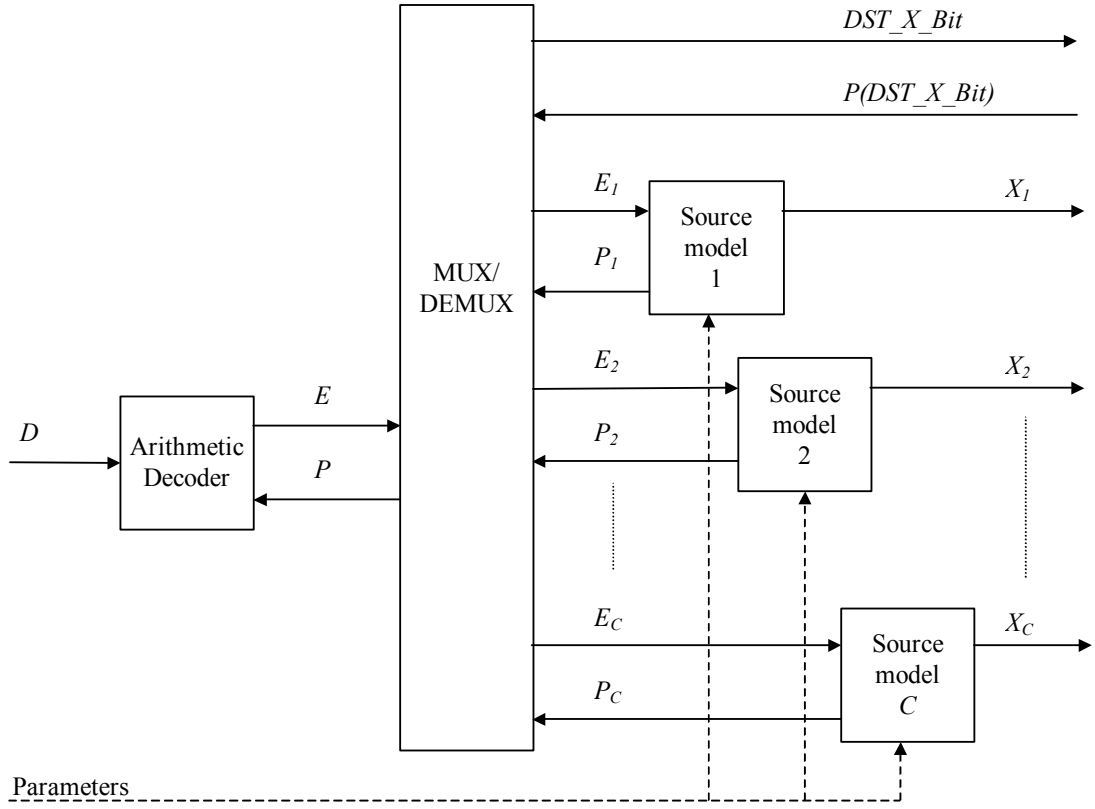

Figure C-3 : Global diagram of the C-channel DST decoder

C.2 DST Decoding Processes

The parameters processed and extracted from the stream are used to decode the DST coded frame. This chapter explains the decoding processes needed for DST coded frames.

C.2.1 Introduction

In Figure C-3, three functions are distinguished: the arithmetic decoder, the multiplexer/demultiplexer and a number of source models. The arithmetic decoder receives the sequence of bits (*D*=A_Data) and the sequence of probabilities (*P*) and generates the sequence of bits (*E*). The *E* and *P* sequences are assigned to the source models in a cyclic order that is controlled by the multiplexer/demultiplexer. Every source model receives the required parameters like prediction filter coefficients and probability table entries from the stream as specified in syntax and semantics.
Source model S corresponds with channel S. The output X of Source model S is the DSD signal for channel S.

C.2.2 Arithmetic Decoder

Arithmetic coding is a variable length coding technique to compress data near to its entropy. The coded data is represented as a number. The number uses as many digits as are required to uniquely identify the original data. For more information about arithmetic coding see:

[1] I.H. Witten, R.M. Neal, and J.G. Cleary, “Arithmetic coding for data compression”, Communications ACM, vol. 30, pp. 520-540, June 1987.
[2] P.G. Howard, and J.S. Vitter, “Arithmetic coding for data compression”, Proc. IEEE, vol. 82, no. 6, p. 857-65, June 1994.

Figure C-4 defines the variables used in Figure C-5, Figure C-6, Figure C-7 and Figure C-8.

Name	Characteristics	Description
A	12 bit register	This unsigned integer represents the current interval size of the arithmetic decoder.
C	12 bit register	This unsigned integer holds part of the arithmetic code bits.
K	4 bit variable	This unsigned integer is a 4 bit approximation of A.
P	8 bit variable	This unsigned integer is the probability value applied to the arithmetic decoder.
Q	12 bit variable	This unsigned integer is the multiplication of K and P.

Figure C-4 : Variables used in Figures C-5, C-6, C-7 and C-8

Figure C-5 shows the overall flow chart of the DST decoding algorithm.

The initialization process is shown in Figure C-6 and is required at the start of decoding each frame. It consists of loading the first 13 bits of the arithmetic code into register C and resetting register A to 4095. The first bit read into C will be overwritten, this is intended because the first bit is always 0.
The function “Input next bit D” in Figures C-5, C-6 and C-8 means bit D is taken from A_Data[ ], starting with the first bit. After all bits from A_Data[ ] have been read, the function “Input next bit D” sets bit D to 0.

The reading of probability information is just retrieving a *P* from the source model (see chapter C.2.3), except for the first bit, where a DST_X_Bit probability is read instead.

Figure C-7 illustrates the decoding of one bit of *E* and updating of register A and C. First the current approximated interval size, *K*, is calculated. Then the multiplication of the approximated interval size *K* and the applied probability value *P* is stored in *Q*. If C is greater than or equal to A-Q, then the arithmetic code lies in the upper part of the interval which means that the original bit encoded, *E*, was a ‘1’ bit; otherwise a ‘0’ bit was transmitted. A and C must be adjusted in the same way as in the encoder.

The renormalizing process is shown in Figure C-8. Renormalizing is required when the value of A is too small. If A is too small, both A and C are shifted one bit left, and a new bit of the arithmetic code, *D*, is read into the least significant bit of C.

It is possible that the last bits of A_Data[ ] are not used by the function “Input next bit D” for decoding of the Audio Frame. These unused bits are stuffing bits for alignment of the Audio Frame on a byte boundary.

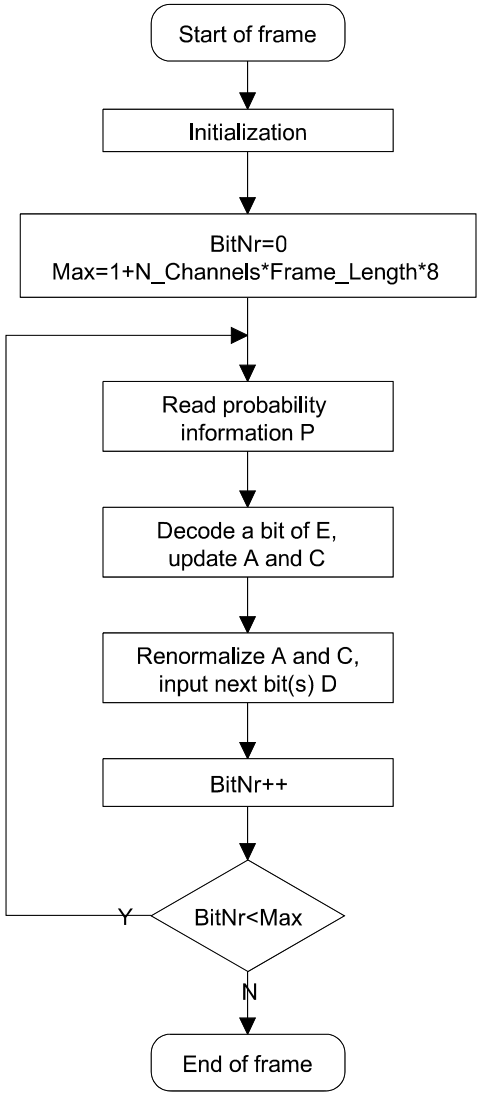

Figure C-5 : Flowchart of the arithmetic decoder

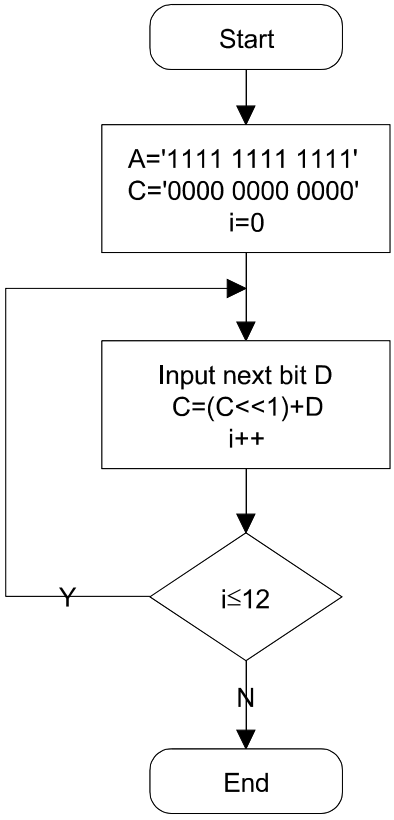

Figure C-6 : Initialization

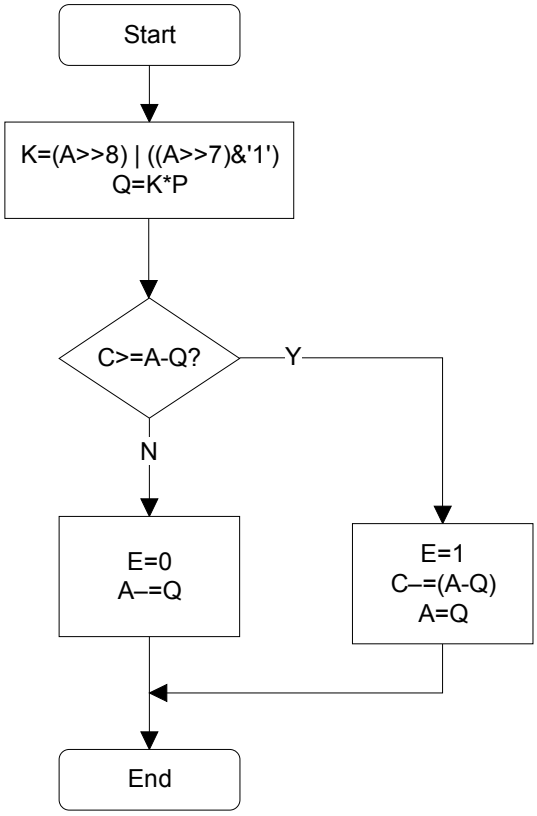

Figure C-7 : Decode a bit of *E*, update A and C

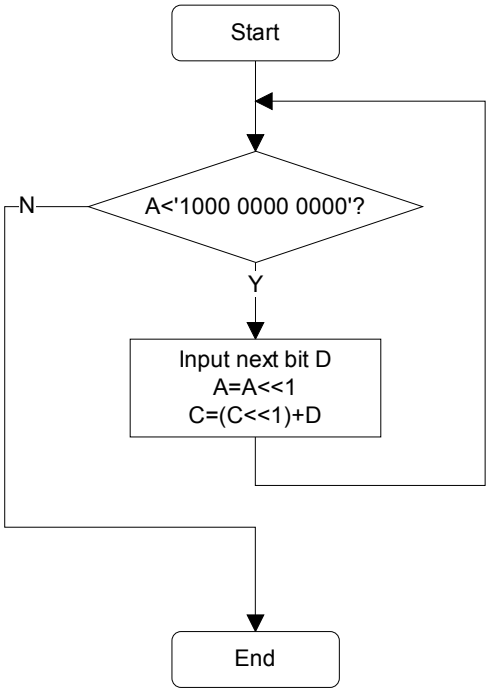

Figure C-8 : Renormalize A and C, input next bit(s) D

C.2.3 Source Model

The decoding process in the source model is described for one channel only, as it is equivalent for all channels. Figure C-9 shows the block diagram of the source model.

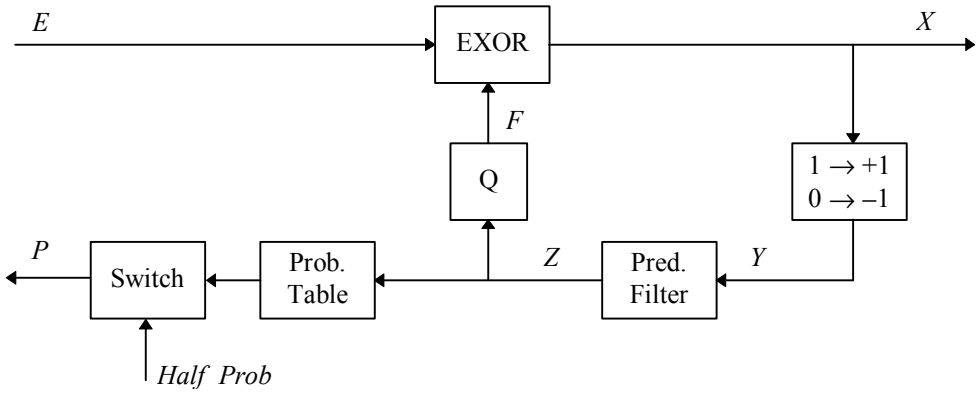

Figure C-9 : Source model

Segmentation and Mapping determine which Prediction Filter and which Ptable must be used to decode the next bit of an Audio Channel. The two functions Filter_N(n) and Ptable_N(n) return the Prediction Filter and Ptable number used for decoding bit n of an Audio Channel. The functions Filter_N(n) and Ptable_N(n) use Segmentation and Mapping information. Filter_N(n) is defined as follows:

Filters.Start[1] = 0
Filters.Start[Seg+1] = Filters.Start[Seg] + Filters.Segment_Length[Channel_Nr][Seg]
Where: Seg = 1 .. Filters.Nr_Of_Segments[Channel_Nr]
and Seg is the Segment number of Audio Channel Channel_Nr.

For bit n, the variable Seg can be determined via:

Filters.Start[Seg] <= (n >> 3) < Filters.Start[Seg+1]

Filter_N(n) can be found by using the following formula:

Filter_N(n) = Filter[Channel_Nr][Seg]

The function Ptable_N(n) is defined using the same mechanism by:

- Replacing Filters by Ptables
- Replacing Filter_N by Ptable_N

Filters.Segment_Length[ ][ ] and Ptables.Segment_Length[ ][ ] are defined in chapter 5.6.3.2.4.
Filters.Nr_Of_Segments[ ] and Ptables.Nr_Of_Segments[ ] are defined in chapter 5.6.3.2.4.
Filter[ ][ ] and Ptable[ ][ ] are defined in chapter 5.6.3.2.5.

Filter_N(n) and Ptable_N(n) are used in the following definitions, which are used in the next section:

N	Pred_Order[Filter_N(n)]	Prediction order of the Prediction Filter that the current Segment uses.
H [ ]	Coef [Filter_N(n)] [ ]	Coefficients of the Prediction Filter that the current Segment uses.
L	Ptable_Len[Ptable_N(n)]	Length of the Ptable that the current Segment uses.
T [ ]	P_one [Ptable_N(n)] [ ]	Entries of the Ptable that the current Segment uses.
n	bit sequence number, range: 0 .. 8*Frame_Length - 1	Variable that runs through all bits of the current Audio Channel.

Frame_Length is defined in chapter 5.6.3.1.
Pred_Order[ ] and Coef[ ][ ] are defined in chapter 5.6.3.2.7.
Ptable_Len[ ] and P_One[ ][ ] are defined in chapter 5.6.3.2.8.

C.2.3.1 Initialization

The initialization of the prediction filter at the start of each frame is defined as:

$$Y[m] = (-1)^m \text{ for } -N \le m < 0$$

The output value of the prediction filter is defined as:

$$Z[n] = \sum_{i=0}^{N-1} Y[n-1-i] \cdot H[i]$$

The Q-function transforms Z into F as follows:

$$F[n] = \begin{cases} 1 & : \quad Z[n] \ge 0 \\ 0 & : \quad Z[n] < 0 \end{cases}$$

There are two methods for applying probability values to the arithmetic decoder. In case *Half_Prob* [*Channel_Nr*] = '0' the probability value is determined by:

$$P[n] = T\Big[\min\big(|Z[n]| >> 3, \quad L-1\big)\Big]$$

In case *Half_Prob* [*Channel_Nr*] = '1' the probability value is determined by:

$$P[n] = \begin{cases} 128 & : \quad 0 \le n < N \\ T\Big[\min\big(|Z[n]| >> 3, \quad L-1\big)\Big] & : \quad n \ge N \end{cases}$$

This value of *P* [*n*] is applied to the arithmetic decoder that will return the value of *E* [*n*]. Next the output value *X* [*n*] (DSD sample) is the exclusive OR of *E* [*n*] and *F* [*n*]:

$$X[n] = E[n] \oplus F[n]$$

The logical value of X[n] is converted to a numerical value of Y[n] as follows:

$$Y[n] = \begin{cases} +1 & : \quad X[n] = 1 \\ -1 & : \quad X[n] = 0 \end{cases}$$

C.2.4 Multiplexing/Demultiplexing

The multiplexing/demultiplexing device connects each source model to the arithmetic decoder at the correct instant.

Some new variables are introduced for readability of the equations:
C N_Channels, see chapter 3.2.1.2
n bit sequence number, range: 0 .. 8*Frame_Length - 1

For $1 \le i \le C$ we have:

$$DST\_X\_Bit = E[0]$$
$$P[0] = P(DST\_X\_Bit)$$
$$E_i[n] = E[C \cdot n + i]$$
$$P[C \cdot n + i] = P_i[n]$$

For a definition of DST_X_Bit and P(DST_X_Bit), see chapter 5.6.3.2.2.

Annex D: Audio Signal Requirements (Normative)

D.1 Super Audio CD Audio Level measuring condition

Super Audio CD Audio levels must be measured after a 50 kHz Butterworth 30 dB/Oct low pass filter.

D.2 Super Audio CD Zero dB Audio Reference Level

The Super Audio CD Zero dB Audio Reference Level, referred to as “0 dB Super Audio CD”, corresponds to a sine wave with a peak amplitude equal to 50% of the theoretical maximum DSD signal level.

D.3 Maximum Super Audio CD Audio Peak Level

The maximum Super Audio CD audio peak level is determined by the maximum allowable DSD Modulation Level (see D.3.1) and by the maximum allowable L-Level (see D.3.2).

D.3.1 Maximum DSD Modulation Level

The DSD Modulation Level is equal to $\frac{|28-2N|}{28}$, where N is the number of bits set to one within any 28 consecutive bits of the DSD stream, and $4 \le N \le 24$. The maximum allowed value of the DSD Modulation Level is 20/28. A DSD Modulation Level of 20/28 corresponds to the maximum Super Audio CD Audio Peak Level of +3.10 dB Super Audio CD. Peak signal levels above +3.10 dB Super Audio CD are not allowed.

D.3.2 Maximum Long term average modulation Level (L-Level)

To calculate the Long term average modulation Level (L-Level), the following algorithm must be used:
A) Peak-hold with slow-down (time constant 258 ms) the DSD Modulation Level.
B) Low pass filter the result of A) using a filter with a time constant of 64 ms.
Figure D-1 shows a block diagram to measure the L-Level. The maximum allowed value for L-Level is +1.51 dB Super Audio CD.

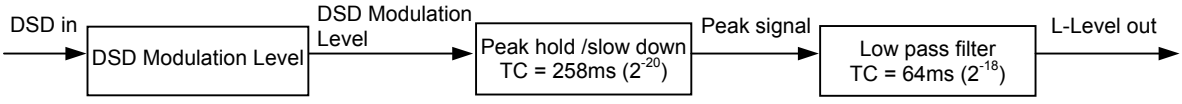

Figure D-1 : Meter for L-Level

D.4 High Frequency DSD Signal + Noise Level

The accumulated RMS signal + noise level of the DSD signal, measured after a 40 kHz Butterworth 30 dB/Oct high pass filter and a 100 kHz Butterworth 30dB/Oct low pass filter, is maximally equal to the RMS level of an input sinewave with a peak amplitude of -20 dB Super Audio CD (see D.2).
The averaging filter used to calculate the RMS level must be a first order unity gain IIR filter with a coefficient of 1/524288 (2^{-19}), corresponding to an IIR filter with a cutoff frequency of about 0.85 Hz.

Annex E: Audio Signal Recommendations (Informative)

E.1 Analog Output Level

The recommended analog output level of a 1 kHz 0 dB Super Audio CD sine wave is 2 V RMS ±1 dB. For a Super Audio CD player with CD-DA playback capability it is recommended that the analog CD-DA output of a fully modulated signal is equal to the analog Super Audio CD output for a 0 dB Super Audio CD signal.
Note: the maximum allowed peak signal level is +3.10 dB Super Audio CD, see D.3.

E.2 Analog Post-filter

To protect analog amplifiers and loudspeakers, it is recommended that a Super Audio CD player contain at its output an analog low pass filter with a cut-off frequency of maximum 50 kHz and a slope of minimum 30 dB/Oct. For use with wide-band audio equipment, filters with a cut-off frequency of over 50 kHz can be used.

E.3 DSD DC Offset

It is recommended that the DC Offset of the DSD signal on the disc is less than -50 dB Super Audio CD.

E.4 DSD polarity

It is recommended that a DSD bit equal to one translates into a rising output voltage at the output of a Super Audio CD player.

Annex F: Restrictions to DST coded Audio_Frames (Normative)

To allow an optimum DST decoder design, the following restrictions must be imposed on DST coded Audio Frames.

F.1 Limited number of erroneously predicted samples

The maximum allowed number of erroneously predicted samples (see signal E in Annex C) in a DST coded Audio_Frame is half of the number of DSD samples in a Frame.

$$N\_Errors\_max = \frac{N\_Channels * Frame\_Length * 8}{2} = 18816 * N\_Channels$$

The total number of erroneous predicted samples (N_Errors) in a Frame is the sum of the number of erroneous predicted samples per Audio Channel:

$$N\_Errors = \sum_{i=1}^{N\_Channels} \left(\sum_{n=0}^{Frame\_Length * 8 - 1} E_i[n] \right)$$

where $E_i[n]$ is either 0 (good prediction) or 1 (wrong prediction) (see Figure C-7).

For each DST coded Audio Frame the following rule must apply:

$$N\_Errors \leq N\_Errors\_max$$

F.2 Probability table design requirement

The restrictions defined in this paragraph must be applied to the Ptables.

For each Ptable that is used in a Frame the contents is determined with the following algorithm:

- Take all samples of the frame into account that use the Ptable in question and count for these samples how many times which Ptable entry is used (CA[ ][ ]), and how many times signal E is equal to 1 for this Ptable entry (CW[ ][ ]).

```
for (Ptable_Nr=0; Ptable_Nr<Nr_Of_Ptables; Ptable_Nr++)
{
   for (Entry_Nr=0; Entry_Nr<Ptable_Len[Ptable_Nr]; Entry_Nr++)
   {
      CA[Ptable_Nr][Entry_Nr] = 0;
      CW[Ptable_Nr][Entry_Nr] = 0;
   }
}
for (Channel_Nr=1; Channel_Nr<=N_Channels; Channel_Nr++)
{
   if (Half_Prob[Channel_Nr]==0)
   {
      Start = 0;
   }
   else
   {
      Start = Pred_Order[Filter[Channel_Nr][1]];
   }
   Stop = 0;
   for (Seg_Nr=1; Seg_Nr<=Ptables.Nr_Of_Segments[Channel_Nr]; Seg_Nr++)
   {
      Stop += 8*Ptables.Segment_Length[Channel_Nr][Seg_Nr];
      for (Bit_Nr=Start; Bit_Nr<Stop; Bit_Nr++)
      {
          Ptable_Nr = Ptable[Channel_Nr][Seg_Nr];
          Entry_Nr = min(|Z[Channel_Nr][Bit_Nr]|>>3, Ptable_Len[Ptable_Nr]-1);
          CA[Ptable_Nr][Entry_Nr]++;
          CW[Ptable_Nr][Entry_Nr] += E_Channel_Nr[Bit_Nr];
      }
      Start = Stop;
   }
}
```

- For each Entry of each Ptable the probability for wrong prediction is derived from these numbers in the following way:

```
for (Ptable_Nr=0; Ptable_Nr<Nr_Of_Ptables; Ptable_Nr++)
{
   for (Entry_Nr=0; Entry_Nr<Ptable_Len[Ptable_Nr]; Entry_Nr++)
   {
      if (CA[Ptable_Nr][Entry_Nr] == 0)
      {
         P_min[Ptable_Nr][Entry_Nr] = 1;
      }
      else
      {
                  ⎛ 512*CW[Ptable_Nr][Entry_Nr]+CA[Ptable_Nr][Entry_Nr] ⎞
         p=trunc  ⎜ ──────────────────────────────────────────────────── ⎟;
                  ⎝              2*CA[Ptable_Nr][Entry_Nr]              ⎠

         P_min[Ptable_Nr][Entry_Nr] = min(max(p, 1), 128);
      }
   }
}
```

- P_min[ ][ ] are the minimum allowed probability values for the entries of the Ptables. For each Entry of each Ptable the probabilities actually used for encoding (P_one[ ][ ]) must meet the following condition:

$$P\_min[Ptable\_Nr][Entry\_Nr] \leq P\_one[Ptable\_Nr][Entry\_Nr] \leq 128$$

List of Changes

Changes from "Super Audio CD System Description Part 2, Audio Specification, Version 1.3" to "Super Audio CD System Description Part 2, Audio Specification, Version 2.0"

The main changes from Version 1.3 to Version 2.0 are:

- The term "SACD" is changed into "Super Audio CD" throughout the document.
- The 2-Channel Stereo Area is made mandatory on Super Audio CD discs.
- Content provider defined playlists added.
- Bonus Tracks feature added.
- Picture and text features added in Supplementary Data and in the new Extension Area.
- Track_List_3 with Copy Control Information and information for pictures and text added.
- Enhanced copy protection system with player authorization.
- The length of the DTCP area changed.
- Revocation data for approved interfaces can be added to the disc.
- Two extended versions of the Area TOC added (Area TOC-3 and Area TOC-4). The existing Area TOC-1 and Area TOC-2 are compatible with Super Audio CD Version 1.3.
- Relaxation of the rule for a silence pattern at the start and end of a Track in case an audio channel is not available.
- Definition of the DST_X_Bit and the DST_Y_Bit moved from Super Audio CD Part 3 to Super Audio CD Part 2.
- New audio signal requirement added to Annex D.3.
- Clarifications and editorial changes.

Unless stated otherwise, the chapter and figure numbers in the following list refer to the numbers found in Version 2.0.

Chapter	Version 1.3	Version 2.0	Remarks
many	SACD	Super Audio CD	Editorial.
1.2	--	Every Super Audio CD disc must have a 2-Channel Stereo Area with Audio Tracks.	Mandatory 2-Channel Stereo Area.
1.2	Support for scalable supplementary data, synchronous with the audio signal.	Support for text and pictures in supplementary data, multiplexed with the audio signal.	Clarification.
1.2	--	Bonus Tracks.	Bonus Tracks.
1.2	--	JPEG pictures, sub-pictures and coded text for slide shows and for browsable information.	Picture and text features.
1.2	--	Super Audio CD players developed according to this version ... and 1.3.	Clarification.
1.3	--	CCIR 601 ...	Reference added.
1.3	--	CCIR 709 ...	Reference added.
1.3	--	EXIF ...	Reference added.
1.3	--	ISO/IEC 10918-1 ...	Reference added.
1.3	--	JPEG ...	Reference added.
1.3	SACD Part 3 ...	Super Audio CD Part 3-General ...	Reference corrected.
1.3	--	Super Audio CD Part 3-Player ...	Reference added.
1.4	--	CCI ...	Reference added.
1.4	--	Copy Control Information ...	Reference added.
1.4	--	EKB ...	Definition added.

Chapter	Version 1.3	Version 2.0	Remarks
1.4	EOS ...	EOS, eos ...	Editorial.
1.4	--	Sector ...	Definition added.
1.4	Supplementary Data Additional data synchronized with the audio signal.	Supplementary Data Additional data multiplexed with the audio signal.	Clarification.
1.6.1	--	roundup(a) ...	Definition added.
1.6.1	--	a^b Bitwise Exclusive OR of a and b.	Definition added.
1.7.2.4 Fig. 1-7	Major_Version ... 1 Minor_Version ... 30	Major_Version ... 2 Minor_Version ... 0	Version number changed.
1.7.2.4	... Major_Version is equal to 1 and Minor_Version is equal to 30.	... Major_Version is equal to 2 and Minor_Version is equal to 0.	Version number changed.
1.7.2.7	... encoded according to ISO 646 IRV.	... encoded according to ISO 646 IRV. Control Characters are not allowed.	Clarification.
1.7.2.8.3.1	--	Control Characters ...	Chapter added. Clarification.
2.2	The Volume Space of a SACD disc is ... and Extra Data Area.	The Volume Space of a disc according to this version ... Revocation Data Area do not exist.	Areas added due to new features.
2.2 Fig. 2-3	--	EKB1 Area Rev TOC Area Extension Area EKB2 Area Revocation Data Area	Areas added. Enhanced copy protection system. Revocation data for interfaces. Picture and text features.
2.2	Every SACD disc must contain ... gaps must be set to zero.	Every Super Audio CD disc according to ... and must be set to zero.	Areas added due to new features. 2-Channel Stereo Area mandatory now.
2.2.1	In case the File System Area is not big enough, additional file system structures can be stored in the Extra Data Area.	When needed file system structures also can be stored in the Extra Data Area.	Clarification
2.2.3	--	EKB1 Area, EKB2 Area ...	Areas added. Enhanced copy protection system.
2.2.5	--	Rev TOC Area ...	Area added. Revocation data for interfaces.
2.2.6	An Audio Area contains two copies of the Area TOC and one Track Area. The Area TOC contains control information for the Audio Area.	Each Audio Area according to this version ... and Area TOC-4 contain control information for the Audio Area.	Area TOC-3 and Area TOC-4.
2.2.6	--	Each Audio Area optionally contains Area TOC-3 and Area TOC-4.	Area TOC-3 and Area TOC-4.

Chapter	Version 1.3	Version 2.0	Remarks
2.2.6	Gaps between the Track Area and both Area TOC-1 and Area TOC-2 are allowed.	Gaps between the Track Area and both Area TOC-3 and Area TOC-4 are allowed. Gaps between Area TOC-1 and Area TOC-3 and between Area TOC-4 and Area TOC-2 are allowed.	Area TOC-3 and Area TOC-4.
2.2.6	The content of all Sectors in the gaps must be set to zero.	The content of all Sectors in the gaps is Reserved.	Clarification.
2.2.6	If a 2-Channel Stereo Area is not present, the Multi Channel Area must start at LSN 544.	Note that the 2-Channel Stereo Area must be present ... optionally is present on discs according to the Super Audio CD Specification Version 1.3 or lower.	Mandatory 2-Channel Stereo Area.
2.2.6 Fig. 2-6	--	Area TOC-3 Area TOC-4	Area TOC-3 and Area TOC-4.
2.2.6.1	Two identical copies of the Area TOC are stored in Area TOC-1 and in Area TOC-2.	The content of Area TOC-1 is equal to ... the same value of Spec_Version.	Area TOC-3 and Area TOC-4.
2.2.6.1	The start location and length of Area TOC-1 and Area TOC-2 is for each Audio Area ...	The start location and length of Area TOC-1, Area TOC-2, Area TOC-3 and Area TOC-4 is for each Audio Area ...	Area TOC-3 and Area TOC-4.
2.2.6.1	The content of the Area TOC-1 and Area TOC-2 is defined ...	The content of the Area TOC-1, Area TOC-2, Area TOC-3 and Area TOC-4 is defined ...	Area TOC-3 and Area TOC-4.
2.2.6.1	... the structure of the Area TOC.	... the structure of Area TOC-1, Area TOC-2, Area TOC-3 and Area TOC-4.	Area TOC-3 and Area TOC-4.
2.2.6.1 Fig. 2-7	--	Track_List_3 Set_of_PlayLists	Figure redrawn. Playlists. Track_List_3.
2.2.6.1	The presence of Track_Text, Index_List, and Track_WebLink_List is optional. ... values of p, q, r and s are given in chapter 3.2.	The presence of Track_Text, Index_List, ... In Area TOC_A values of t and u must be equal to zero.	Area TOC-3 and Area TOC-4. Track_List_3. Playlists.
2.2.6.2	--	Where the term Track Area is used ... for Area TOC-B.	Area TOC-3 and Area TOC-4.
2.2.6.2 Fig. 2-8	-- -- -- --	Track Area-B ... Track Area-A ... Bonus Tracks Pause[n+1] ... Track[m]	Bonus Tracks. Area TOC-3 and Area TOC-4.
2.2.6.2	For each Track, ... for that Audio Area ...	For Track[1] to Track[N], ... and in Area TOC-B ...	Bonus Tracks.
2.2.6.2	--	For Track[N+1] to Track[M], the start ... playing time of Track Area-B.	Bonus Tracks.
2.2.6.2.6	--	If the 2-Channel Stereo Area contains Bonus Tracks, the 2-Channel Stereo Area must be DST coded DSD.	Rule added. Bonus Tracks.
2.2.7	--	Extension Area ...	Picture and text features.

Chapter	Version 1.3	Version 2.0	Remarks
2.2.8	--	Revocation Data Area ...	Area added. Revocation data for interfaces.
2.2.9	The Extra Data Area can optionally ... needed for the file system(s).	The Extra Data Area can optionally be used to store additional audio related information.	Editorial.
3.1.1 Fig. 3-2	--	Disc_Info_2()	Structure added.
3.1.1.3.6	... length in Sectors of the Area TOC ...	... length in Sectors of Area TOC-A ...	Area TOC-3 and Area TOC-4 added.
3.1.1.3.7	... length in Sectors of the Area TOC ...	... length in Sectors of Area TOC-A ...	Area TOC-3 and Area TOC-4 added.
3.1.1.6	--	Disc_Info_2 ...	Structure added.
3.2	Two copies of the Area TOC are ... the Tracks in the Audio Area.	Two versions of the Area TOC exist ... the Tracks in Track Area-B for Area TOC-B.	Area TOC-3 and Area TOC-4 added.
3.2 Fig. 3-11	--	Track_List_3() ...	Track_List_3 added.
3.2 Fig. 3-11	--	Set_Of_PlayLists() ...	Playlists added.
3.2	The presence of Track_Text, Index_List ... Time Code information and Track attributes.	The fields Track_List_3 and ... is conditionally present on version 2.0 or higher discs.	Track_List_3 added.
3.2.1.1.3	Note: Additional Lists defined in SACD Part 3 are part ...	Note: In case additional Lists are defined in Super Audio CD Part 3, these Lists are part ...	Clarification.
3.2.1.2.1	... Audio Area ...	... Track Area ...	Editorial, 2 times.
3.2.1.2.6	... Audio Area ...	... Track Area ...	Editorial, 3 times.
3.2.1.2.7	... Audio Area ...	... Track Area ...	Editorial, 6 times.
3.2.1.2.9	... Audio Area ...	... Track Area ...	Editorial.
3.2.1.2.9	--	The values of Total_Area_Play_Time in ... in Area TOC-B)	Bonus Tracks.
3.2.1.2.10	... Audio Area ...	... Track Area ...	Editorial.
3.2.1.2.11	... Audio Area ...	... Track Area ...	Editorial.
3.2.1.2.11	--	Note: In Area TOC-A the value of ...	Note added. Bonus Tracks.
3.2.1.2.12	--	First_Bonus_Track_Number ...	Chapter added. Bonus Tracks.
3.2.1.2.15	... Audio Area ...	... Track Area ...	Editorial.
3.2.1.2.15		It is recommended that ... Area TOC-B.	Recommendation added. Area TOC-3 and Area TOC-4.
3.2.1.3 Fig. 3-27	--	Track_List_3_Ptr	Track_List_3 added.
3.2.1.3 Fig. 3-27	--	Set_Of_PlayLists_Ptr	List added. Playlists.
3.2.1.3.1	... from the start of the current Area_TOC.	... from the start of the current Area_TOC to the first Sector of Track_Text.	Clarification.
3.2.1.3.2	... from the start of the current Area_TOC.	... from the start of the current Area_TOC to the first Sector of Index_List.	Clarification.

Chapter	Version 1.3	Version 2.0	Remarks
3.2.1.3.2	... the Audio Area contains ...	... the Track Area contains ...	Editorial.
3.2.1.3.3	... from the start of the current Area_TOC.	... from the start of the current Area_TOC to the first Sector of Access_List.	Clarification.
3.2.1.3.4	... from the start of the current Area_TOC.	... from the start of the current Area_TOC to the first Sector of Track_WebLink_List.	Clarification.
3.2.1.3.4	... the Audio Area contains ...	... the Track Area contains ...	Editorial.
3.2.1.3.5	--	Track_List_3_Ptr ...	Track_List_3 added.
3.2.1.3.6	--	Set_Of_PlayLists_Ptr ...	List added. Playlists.
3.2.1.4.5	... current Audio Area ...	... current Track Area ...	Editorial.
3.2.1.4.6	... current Audio Area ...	... current Track Area ...	Editorial.
3.2.1.4.7	... current Audio Area ...	... current Track Area ...	Editorial.
3.2.1.4.8	... current Audio Area ...	... current Track Area ...	Editorial.
3.2.2	... current Audio Area.	... current Track Area.	Editorial.
3.2.3	... current Audio Area.	... current Track Area.	Editorial.
3.2.3.2	... Time_Code ...	... Time Code ...	Editorial.
3.2.3.5.2	--	In case an Audio Channel is not available ... same Audio Channel is available for the next Track.	Relaxed silence pattern rule.
3.2.3.5.2	--	Figure 3-35 gives an overview of the allowed combinations of TMF1, TMF2, TMF3 and TMF4.	Figure added.
3.2.3.5.2 Fig. 3-35	--	Figure 3-35 : Allowed combinations of TMF1, TMF2, TMF3 and TMF4 ...	Clarification. Figure added.
3.2.4.1	... the first Sector with ...	... the first Sector of ...	Editorial.
3.2.5	... DST coded Audio Area.	... DST coded Track Area.	Editorial.
3.2.5.4.1.1	... last Interval in an Audio Area, ...	... last Interval in a Track Area, ...	Editorial.
3.2.5.4.1.3	... of the Audio Area expressed in Frames.	... of the Track Area expressed in Frames.	Editorial.
3.2.5.5.1	Detailled_List[N]()	Detailled_List()[N]	Editorial.
3.2.6	... in the current Audio Area.	... in the current Track Area.	Editorial.
3.2.6.1	... identifying the Sector with ...	... identifying the first Sector of ...	Correction.
3.2.7	... if the Audio Area contains ...	... if the Track Area contains ...	Editorial.
3.2.7.1	... identifying the Sector with ...	... identifying the first Sector of ...	Correction.
3.2.7.4	... start of the Audio Area ...	... start of the Track Area ...	Editorial.
3.2.8	... if the Audio Area contains ...	... if the Track Area contains ...	Editorial.
3.2.8.1	... identifying the sector with Index_List.	... identifying the first Sector of Track_WebLink_List.	Correction.
3.2.8.1	The value of Track_WebLink_Signature must be ...	The value of Track_WebLink_List_Signature must be ...	Correction.
3.2.9	--	Track_List_3 ...	Track_List_3 added.
3.2.10	--	Set_Of_PlayLists ...	Playlists added.

Chapter	Version 1.3	Version 2.0	Remarks
3.3	The length of the DTCP Area is 110 Sectors.	The length of the DTCP Area is 48 Sectors.	Length of DTCP Area changed.
3.3.1	The definition of DTCP_SRM_List is given...	The syntax of DTCP_SRM_List is defined ...	Editorial.
3.3.1 Fig. 3-54	Until 110*2048	Until 48*2048	Length of DTCP Area changed.
3.3.1 Fig. 3-54	Definition of DTCP_SRM_List	Syntax of DTCP_SRM_List	Editorial.
3.4	--	EKB_Data ...	Enhanced copy protection system.
3.5	--	Revocation_TOC ...	Revocation data for approved interfaces added.
4. Fig.4-1	Example of the SACD directory structure ...	Example of the Super Audio CD directory structure ...	New files added.
4.2	The required files in the ROOT directory are ... and SRM0.LST.	The required files in the ROOT directory are ... and REVOC.DAT are not present.	New files added.
4.2.2	The length of the file SRM0.LST must be 110*2048 bytes.	The length of the file SRM0.LST must be 48*2048 bytes.	Length of DTCP Area changed.
4.2.3	--	Files EKB1.DAT and EKB2.DAT ...	New files added.
4.2.4	--	File REVTOC.TOC ...	New file added.
4.2.5	--	File EXTENS.TOC ...	New file added.
4.2.6	--	File EXTENS.DAT ...	New file added.
4.2.7	--	File REVOC.DAT ...	New file added.
4.3.1	... copy of the Area TOC ...	... copy of the Area TOC-A ...	Area TOC-3 and Area TOC-4.
4.3.1	... equal to 2048*2CH_TOC_Length ...	... equal to 2048*2CH_TOC_Length bytes ...	Editorial.
4.3.2	--	Files 2C_AREA3.TOC and 2C_AREA4.TOC ...	New files added.
4.3.3	...of the 2-Channel Stereo Area.	... as defined in Area TOC-B, or in Area TOC-A if Area TOC-B does not exist, for the 2-Channel Stereo Area.	Area TOC-3 and Area TOC-4.
4.3.4	--	The highest value of “nnn” must be equal to ... 2-Channel Stereo Area.	Definition added. Area TOC-3 and Area TOC-4.
4.4.2	--	Files MC_AREA3.TOC and MC_AREA4.TOC ...	New files added.
4.4.3	...of the Multi Channel Area.	... as defined in Area TOC-B, or in Area TOC-A if Area TOC-B does not exist, for the Multi Channel Area.	Area TOC-3 and Area TOC-4.
4.4.4	--	The highest value of “nnn” must be equal to ... Multi Channel Area.	Definition added. Area TOC-3 and Area TOC-4.
5.	--	Note that for the definition of the Audio Data Format the Track Area includes Bonus Tracks if present.	Note added. Bonus Tracks.
5.3.1.3	... Audio Sector ...	... Audio_Sector ...	Editorial.

Chapter	Version 1.3	Version 2.0	Remarks
5.3.1.3	... must be added to the end of that Audio Sector.	... must be added until the end of that Audio Sector.	Editorial
5.3.1.3	--	If an Audio Area contains Bonus Tracks, the following rules apply ...	Rules for Bonus Tracks added.
5.4	--	If the 2-Channel Stereo Area contains Bonus Tracks, the 2-Channel Stereo Area must be DST coded.	Rules for Bonus Tracks added.
5.6.3.2.2	For the definition of DST_X_Bit, see SACD Part 3.	On the disc two patterns are allowed for the DST_X_Bit ...	Definition of the DST_X_Bit moved.
5.6.3.2.2.1	--	P(DST_X_Bit) (Multiplexing/Demultiplexing) ...	Definition of the DST_X_Bit moved.
5.6.3.2.7.1	For the definition of DST_Y_Bit, see SACD Part 3.	The DST_Y_Bit must be set ... shall ignore the content of the DST_Y_Bit.	Definition of the DST_Y_Bit moved.
5.7	...information that is synchronized with the audio ...	... information that is multiplexed with the audio ...	Editorial.
5.7	... concatenation of all Supplementary Data Frames (see chapter 5.2) in a Byte Stream, see Figure 5-5.	... concatenation of all Supplementary Data Frames (see chapter 5.2) in an Audio Area. Both the 2-Channel Stereo Area and the Multi Channel Area contain one Supplementary Data Stream each.	Editorial.
5.7.1 Fig. 5-43	Stuffing ...	SD_Stuffing ...	Name changed. Editorial.
5.7.1	--	N is the number of SD_Units ... of N is 107.	Clarification.
5.7.1.1	SD_Unit ...	SD_Unit ...	Chapter and sub-chapters added. Picture and text features in Supplementary Data added.
5.7.1.2	--	SD_Stuffing ...	Chapter added. Picture and text features in Supplementary Data added.
5.7.2	--	Supplementary Data player guidelines ...	Chapter added. Picture and text features in Supplementary Data added.
6.	--	Extension Area ...	Chapter and sub-chapters added. Extension Area added.
7.	--	JPEG Definitions and Restrictions ...	Chapter and sub-chapters added. Picture definitions.
8.	--	Sub Picture Definitions ...	Chapter and sub-chapters added. Text and pictures.

Chapter	Version 1.3	Version 2.0	Remarks
Annex A.2	--	Supplementary Data Buffer Model ...	Buffer model added. Picture and text features in Supplementary Data.
Annex C.2.4	For a definition DST_X_Bit and P(DST_X_Bit), see Part 3.	For a definition of DST_X_Bit and P(DST_X_Bit), see chapter 5.6.3.2.2.	Definition of the DST_X_Bit and DST_Y_Bit moved.
Annex D.3	The maximum SACD audio peak level is determined by the maximum allowable DSD modulation level.	The maximum Super Audio CD audio peak level is determined by the maximum allowable DSD Modulation Level (see D.3.1) and by the maximum allowable L-Level (see D.3.2).	Audio signal requirement added.
Annex D.3.1	--	Content of the old Annex D.3	Audio signal requirement added.
Annex D.3.2	--	Maximum Long term average modulation Level (L-Level) ...	Audio signal requirement added.

**Changes from "Super Audio CD System Description Part 2, Audio Specification, Tentative Version 1.9"
to "Super Audio CD System Description Part 2, Audio Specification, Version 2.0"**

The main changes from Tentative Version 1.9 to Version 2.0 are:

- SD_Slide_List with the start addresses of all slide show pictures in Supplementary Data added.
- Clarifications and editorial changes.

Unless stated otherwise, the chapter and figure numbers in the following list refer to the numbers found in Version 2.0.

Chapter	Tentative Version 1.9	Version 2.0	Remarks
1.7.2.4 Fig. 1-7	Major_Version ... 1 Minor_Version ... 90	Major_Version ... 2 Minor_Version ... 0	Version number changed.
1.7.2.4	... Major_Version is equal to 1 and Minor_Version is equal to 90.	... Major_Version is equal to 2 and Minor_Version is equal to 0.	Version number changed.
2.2.6.2	--	The total playing time of Track Area-A must minimally be equal to 80% of the total playing time of Track Area-B.	Clarification.
3.1.1.6.13	--	The Revocation Data Area is not present in discs according to the Super Audio CD Specification version 1.3 or lower.	Clarification.
3.1.1.6.14	--	The Revocation Data Area is not present in discs according to the Super Audio CD Specification version 1.3 or lower.	Clarification.
3.2.1.2.9	--	The values of Total_Area_Play_Time in ... in Area TOC-B)	Clarification.
3.2.1.2.12	Note: In Area ... to zero.	Notes: In Area ... see chapter 3.2.1.2.9.	Note added. Clarification.
3.2.9 Fig. 3-46	Reserved 16 ... Reserved until 4*4096	SD_Slide_List() 8192 ... Reserved until 4*4096	SD_Slide_List added.
3.2.9.2.3.1	If Slide Show pictures are present ... must be set to zero.	If one or more Slide Show pictures with a ... must be set to zero.	Clarification.
3.2.9.2.4.1	If Streaming_Info SubPictures are present ... must be set to zero.	If one or more Streaming_Info SubPictures for Text_Channel[c] ... must be set to zero.	Clarification.
3.2.9.4	--	SD_Slide_List ...	SD_Slide_List added.
5.7.1.1.2	--	Figure 5-47 gives an example of the de-interleaving of Information Units.	Figure added. Clarification.
5.7.1.1.2 Fig 5-47	--	Figure 5-47 ...	Figure added. Clarification.
5.7.1.1.4	... not be displayed.	... not be displayed, also see chapter 5.7.1.1.5.	Clarification.

Chapter	Tentative Version 1.9	Version 2.0	Remarks
5.7.1.1.4	... should be stopped, possibly with using a transition effect.	... should be stopped, also see chapter 5.7.1.1.6.	Clarification.
5.7.1.1.4 Fig 5-48	Figure 5-47 ...	Figure 5-48 ...	New figure. Clarification.
5.7.1.1.4.3	... repeat the Browsable Text and Browsable Sub_Pictures from ...	... repeat the Browsable Information (see chapter 5.7.1.1.9) and Background_Picture_A/_B/_C from ...	Clarification.
5.7.1.1.7.2	--	One Audio Area can contain maximum 2047 Slide_Show_Pictures.	Clarification.
5.7.1.1.7.2	... + 20 Frames.	... + 20 Frames, where n is the sequence number of an Information Uinit containing a Slide_Show_Picture.	Clarification.
5.7.1.1.8	The use of Text_Channel_Mapping ... combination exists.	For each Time_Code ... T_Sub_Type combination.	Clarification.
5.7.1.1.9.1	Start_PTC[N+1] > ... same Text_Channel_Mapping.	Start_PTC[N+1] > ... containing Streaming_Info.	Clarification.
6.1.1.6.2	--	All ENodes in the Enode_Group ... equal to $FF:$FF:$FF.	Clarification.
6.1.1.6.3	--	All ENodes in the Enode_Group ... equal to $FF:$FF:$FF.	Clarification.
6.1.1.6.4	--	All ENodes in the Enode_Group ... equal to $FF:$FF:$FF.	Clarification.
6.1.1.6.5	--	All ENodes in the Enode_Group ... equal to $FF:$FF:$FF.	Clarification.
6.1.4.1	... the value of TC set to the PreLoad_Start_TC (see chapter 3.2.9.2.2.2) of Track[tno].	... the value of TC set to Track_Start_Time_Code[tno] (see chapter 3.2.3.2).	Clarification.
6.1.4.2	... the value of TC set to the PreLoad_Start_TC (see chapter 3.2.9.2.2.2) of Track[tno].	... the value of TC set to Track_Start_Time_Code[tno] (see chapter 3.2.3.2).	Clarification.
6.1.4.4	... Time Code in Track[tno], a value of $FF:$FF:$FF is not allowed.	... Time Code in Track[tno]. A value of $FF:$FF:$FF is not allowed.	Editorial.