User's Manual

ARM 5.1ch aacPlus V2 Decode Middleware for Linux

RTM0AC0000ADAAPMZ1SL32C

User's Manual

RTM0AC0000ADAAPMZ1SL32E-02

Renesas Electronics
www.renesas.com

Rev.1.01 December 2014

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(2012.4)

Notes regarding this middleware

1. Customers wishing to use this product must bear the responsibility for complying with the Foreign Exchange and Foreign Trade Control Law, and any other related regulations and procedures, and for determining whether a permit from the Japanese Government is required.

2. Renesas Electronics Corporation neither warrants nor grants licenses of any rights of Renesas Electronics Corporation’s or any third party’s patent, copyright, trademark, or other intellectual property rights for information contained in this document. Renesas Electronics Corporation bears no responsibility for problems that may arise with third party’s rights, including intellectual property rights, in connection with use of the information contained in this document.

3. Products and product specifications may be subject to change without notice. Confirm that you have received the latest product standards or specifications before final design, purchase or use.

4. Renesas Electronics Corporation cannot accept responsibility for primary or secondary damage arising from modifications to the product by the customer. Renesas Electronics Corporation bears no responsibility for failure and damage arising from the system or media including this products.

5. Renesas Electronics Corporation makes every attempt to ensure that its products are of high quality and reliability. However, contact Renesas Electronics Corporation’s sales office before using the product in an application that demands especially high quality and reliability or where its failure or malfunction may directly threaten human life or cause risk of bodily injury, such as aerospace, aeronautics, nuclear power, combustion control, transportation, traffic, safety equipment or medical equipment for life support.

6. Design your application so that the product is used within the ranges guaranteed by Renesas Electronics Corporation particularly for maximum rating, operating supply voltage range, heat radiation characteristics, installation conditions and other characteristics. Renesas Electronics Corporation bears no responsibility for failure or damage when used beyond the guaranteed ranges. Even within the guaranteed ranges, consider normally foreseeable failure rates or failure modes in semiconductor devices and employ systemic measures such as fail-safes, so that the equipment incorporating Renesas Electronics Corporation product does not cause bodily injury, fire or other consequential damage due to operation of the Renesas Electronics Corporation product.

7. Renesas Electronics Corporation owns the copyright of this product and this document. It may not be duplicated, analyzed, or reproduced in any form, either in part or in its entirety, without written approval from Renesas Electronics Corporation.

8. No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without written approval from Renesas Electronics Corporation.

9. Renesas Electronics Corporation permits the use of this product only in a single computer. Renesas Electronics Corporation does not permit the transfer, loan, or rental of this product without written approval from Renesas Electronics Corporation.

10. Contact Renesas Electronics Corporation’s sales office for any questions regarding this document or Renesas Electronics Corporation semiconductor products.

How to Use This Manual

1. Purpose and Target Readers

This manual is designed to provide the user with an understanding of the user interface specifications of the middleware product. It is intended for users designing application systems incorporating the middleware product. Please refer to the related documents with this product.

Particular attention should be paid to the precautionary notes when using the manual. These notes occur within the body of the text, at the end of each section, and in the Usage Notes section.

The revision history summarizes the locations of revisions and additions. It does not list all revisions. Refer to the text of the manual for details.

2. Restrictions on the Use of this Middleware

Any customer who wishes to use this middleware must obtain a software license from Renesas Electronics.

The middleware may not be copied, except to make single copies for archival purpose, and may not be decompiled or reverse engineered, except as permitted by mandatory legal provisions under applicable law.
HE AAC by Dolby Labs Inc. (http://www.dolby.com)

To use the AAC and aacPlus V1/V2 techniques and patents contained in this middleware, you must separately enter into the following license agreement by yourself.

Via Licensing Corporation (http://www.vialicensing.com/)

3. Related Manuals

[1] Linux Interface Specification Yocto recipe Start-Up Guide
[2] R-Car Series, 2nd Generation User's Manual: Hardware

Standards
ISO/IEC 13818-7:2006
ISO/IEC 14496-3:2009
ARIB (The Association of Radio Industries and Businesses) Standard B-32, B-21 Rev. 5.2, Rev. 5.3
ETSI TS 101 154 V1.9.1: 2009
ETSI TS 102 563 V1.2.1 (2010-05)
DTG D-Book Issue 6.2.1: May 2010
Brazilian ISDB-T: ABNT NBR 15602-2:2007 and ABNT NBR 1560

4. Technical Terms and Abbreviations

Abbreviation	Full Form
AAC	Advanced Audio Coding
HE-AAC V1	AAC stream has SBR extension
HE-AAC V2	AAC stream has SBR and PS extension
SBR	Spectral Band Replication
PS	Parametric Stereo
LOAS	Low Overhead Audio Stream
LATM	Low overhead MPEG-4 Audio Transport Multiplex
PCM	Pulse Code Modulation
DRC	Dynamic Range Control
CRC	Cyclic Redundancy Check
SCE	Single Channel Element
CPE	Channel Pair Element
CCE	Coupling Channel Element
LFE	LFE Channel Element
PCE	Program Config Element
DSE	Data Stream Element
I/O	Input/Output

Table of Contents

1 Overview

This middleware is the embedded middleware for ARM that decodes AAC and HE-AAC (aacPlus) V1/V2 coded bit stream data compliant with the international standards, ISO/IEC 13818-7:2006 and 14496-3:2009.

Hereinafter, for the coded bit stream type (profile), HE-AAC (aacPlus) is referred to as aacPlus.

Table 1-1shows the basic specifications of this decode middleware.

Table 1-1 Basic Specification

Item		Description
Product Name		ARM 5.1ch aacPlus V2 Decode Middleware for Linux
Product Type Name		RTM0AC0000ADAAPMZ1SL32C
Target CPU		ARM
OS		Linux kernel release 3.10
Memory usage (size) **(Estimation)** (Note 1)		ROM section : 172 Kbytes RAM section : 2 Kbytes stack : 5 Kbytes work : 225 Kbytes (allocated by user) Input buffer : more than 1 byte Output buffer : more than 2048 bytes per channel(AAC-LC) : more than 4096 bytes per channel(HE-AAC) Input and output buffers are allocated by user
Interface		C-language interface (library functions)
Decoding specifications	**Standards**	ISO/IEC 13818-7:2006 Fourth edition ISO/IEC 14496-3:2009 Fourth edition Brazilian ISDB-T: ABNT NBR 15602-2:2007 and ABNT NBR 1560 ETSI TS 101 154 V1.9.1:2009 (DVB-T Standard) ESTI TS 102 563 V1.2.1:2010 (DAB Standard) ARIB Standard B-32, B-21 Rev. 5.2, B-21 Rev. 5.3 DTG D-Book Issue 6.2.1: May 2010
	Profile	AAC-LC HE-AAC V1 (aacPlus V1) HE-AAC V2 (aacPlus V2)
	Input format	ADIF, ADTS, LOAS / LATM (Note 2), RawDataStream format supported
	Output format	16-bit linear non-interleaved PCM (Note 3)
	Number of channels supported	1ch (Monaural), 2ch (Stereo, Dual monaural), 3ch (3/0, 2/1), 4ch (3/1, 2/2), 5ch (3/2), 5.1ch (3/2 + LFE) * "/" indicates the number of channels of the front/rear speakers
	Downmix	3 ch (3/0, 2/1), 4 ch (3/1, 2/2), 5 ch (3/2), and 5.1 ch (3/2+LFE) can be downmixed to monaural or stereo

Table 1-1 Basic Specification(2)

Item		Description	
Decoding specifications	**Supported sampling frequency**	AAC	8, 11.025, 12, 16, 22.05, 24, 32, 44.1, 48, 64, 88.2, 96 [kHz]
		HE-AAC V1/V2 (aacPlus V1/V2)	16, 22.05, 24, 32, 44.1, 48 [kHz] (Note 4)
	Supported bit rate	AAC	8 to 576 [kbits/sec]
		HE-AAC V1/V2 (aacPlus V1/V2)	8 to 128 [kbits/sec]
CPU Load (processing performance) (Estimation) (Note 5)/(Note 6)		AAC	Avg. 11.9 MHz (sampling frequency:48kHz, bit rate:128kbits/sec, Stereo) Avg. 26.4 MHz (sampling frequency:48kHz, bit rate:384kbits/sec, 5.1ch)
		HE-AAC V1 (aacPlus V1)	Avg. 21.1 MHz (sampling frequency:48kHz, bit rate:48kbits/sec, Stereo) Avg. 60.1 MHz (sampling frequency:48kHz, bit rate:128kbits/sec, 5.1ch)
		HE-AAC V2 (aacPlus V2)	Avg. 12.0 MHz (samplingfrequency:48kHz, bit rate:48kbits/sec, Stereo)
Endian		Little	
Reentrancy		Supported	

(Note 1) The memory capacities are shown on condition that K equals to 1024.

(Note 2) This middleware does not detect CRC error for LOAS format.

(Note 3) 2 channels (L/R) data can be converted to Interleave by function RSACPD_InterleavePCM.

(Note 4) The sampling frequencies for HE-AAC(aacPlus) coded bit stream indicate the output sampling frequencies.

(Note 5) The performance of the frame whose bit rate is locally higher would be worse. Ex. The frame including attack signal.

(Note 6) The value is measured on R-Car H2 Evaluation Board (ARM Cortex-A15) and it is not guaranteed for the every case.

2 Input/Output Data Format

2.1 Input data format

This decode middleware retrieves the header information, side information to be arguments for decoding, and the main data structured by frequency components (to be decoding target), from the input AAC/aacPlus V1/aacPlus V2 coded bit stream.

2.1.1 ADTS-format

In the ADTS-format, each ADTS frame[1] consists of one or more blocks (raw_data_block). The number of blocks can be obtained by "number_of_raw_data_block_in_frame" in the ADTS header. The ADTS frame must start with a 12-bit syncword, followed by the ADTS header.

Figure below shows the bit stream structure in the ADTS-format as an example.

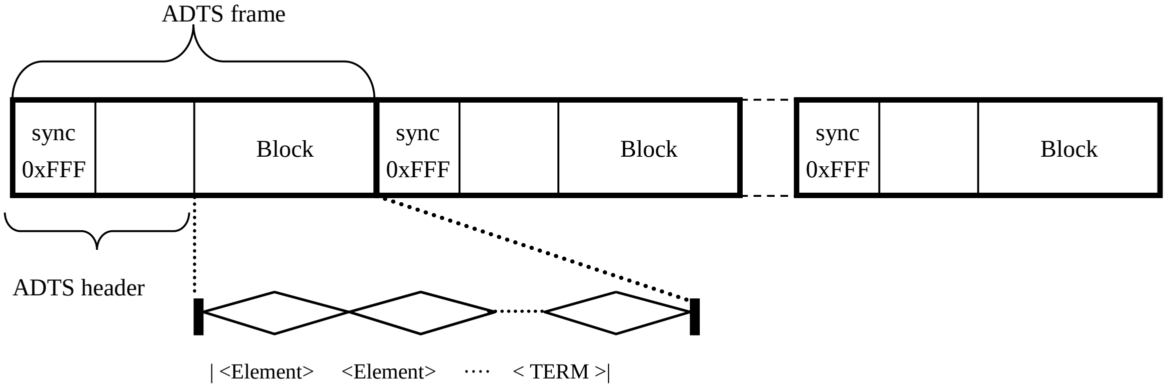


Figure 2.1 Structure of AAC coded bit stream (ADTS-format)

2.1.2 ADIF-format

For the structure of the ADIF-format bit stream, an ADIF header locates at the head of the bit stream once. No more headers locate, and only blocks (raw_data_block) follow through the end.

The ADIF header contains one PCE (program_config_element) or more.

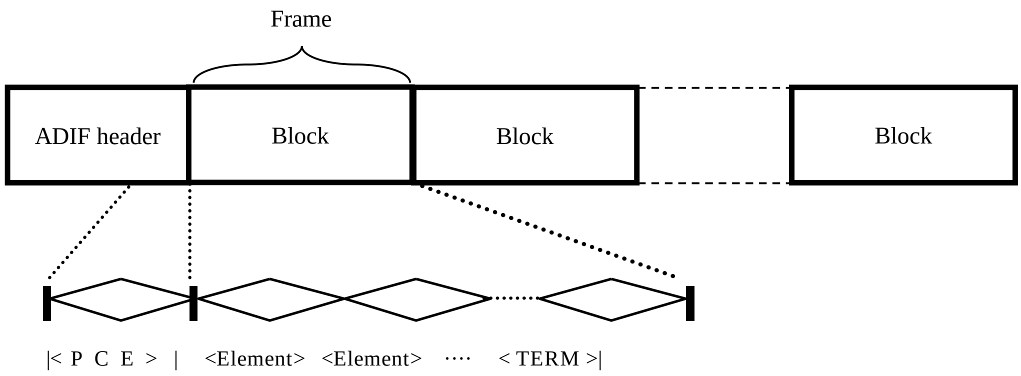


Figure 2.2 Structure of AAC coded bit stream (ADIF-format)

[1]In this manual, "frame" and "ADTS frame" are separately used. "1 frame" indicates the output unit of the audio data of 1024/2048 or 960/1920 samples/channel. "ADTS frame" consists of one ADTS header and one or more raw_data_block(s).

2.1.3 LOAS/LATM format

Fig 2.3 shows the concept of MPEG-4 Audio transport. The transport mechanism uses a two-layer approach, namely a multiplex layer and a synchronization layer.

The multiplex layer (Low-overhead MPEG-4 Audio Transport Multiplex: LATM) manages multiplexing of several MPEG-4 Audio payloads and their AudioSpecificConfig() elements.

The synchronization layer specifies a self-synchronized syntax of the MPEG-4 Audio transport stream which is called Low Overhead Audio Stream (LOAS). The interface format to a transmission layer depends on the conditions of the underlying transmission layer.

This middleware supports only kind of bit stream which includes both multiplex layer (LATM) and synchronization layer (LOAS). Thus, the term “LOAS” is used to indicate LOAS/LATM format in this manual.

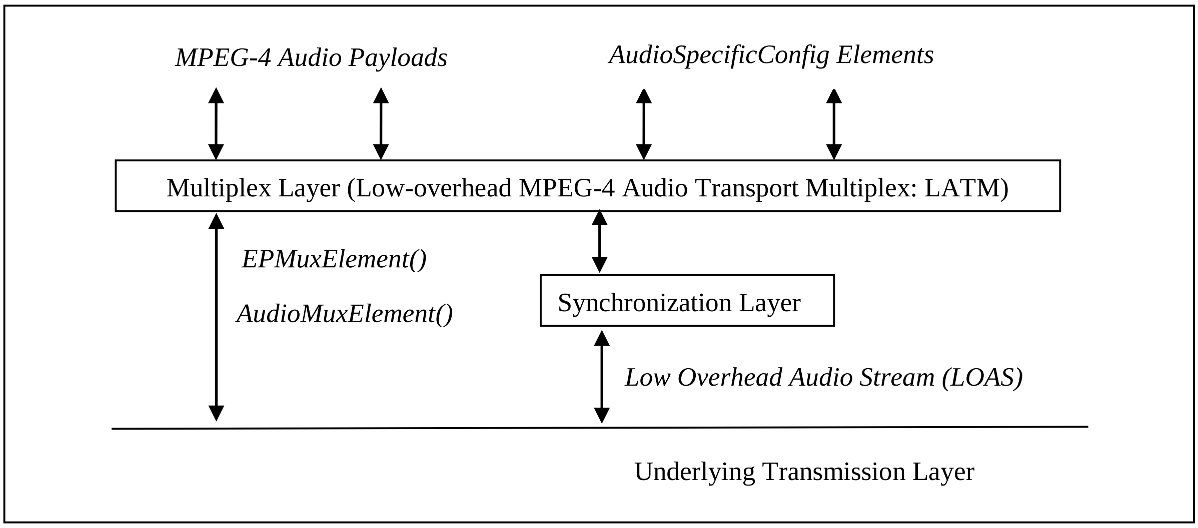


Figure 2.3 Concept of MPEG-4 Audio Transport

In LOAS format, the header is started by a 11-bit syncword, followed by one or more sub frames (raw_data_block).

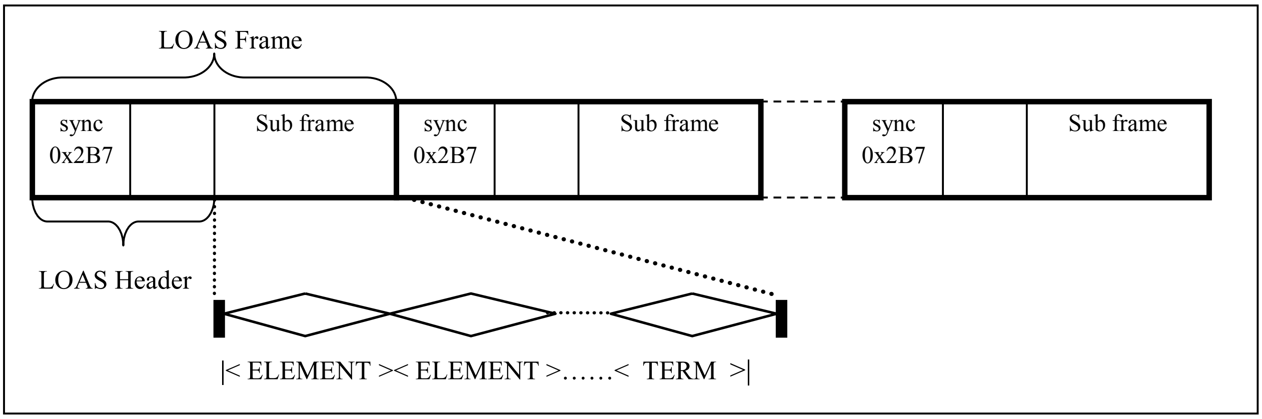


Figure 2.4 Structure of AAC coded bit stream (LOAS format)

2.1.4 RawDataStream-format

A bit stream in the RawDataStream-format consists only of blocks (raw_data_block). Therefore, the bit stream has the same structure as that in the ADIF-format except that it does not have the ADIF header shown in Figure 2.2.

2.1.5 raw_data_block

Each block (raw_data_block) consists of seven types of elements. The types of the elements are listed below:

<Element>

SCE	(single_channel_element)
CPE	(channel_pair_element)
CCE	(coupling_channel_element)
LFE	(lfe_channel_element)
PCE	(program_config_element)
FILL	(fill_element)
DSE	(data_stream_element)
TERM	(ID_END)

2.2 Output Data Format

The PCM data output by this decode middleware is a 16-bit signed integer type. The output starts from the lowest address of the output buffer array. The output data is in the same endian as the target CPU.

This middleware decodes a bitstream by 1 block (raw_data_block) and output PCM data. The number of output words varies depending on the input bit stream type (AAC/aacPlus) and decode mode (AAC upsample/downsample SBR). For the details, see Section 3.2.7 “RSACPD_Decode”

As the output PCM, 1024 or 2048 words (960 or 1920 words) will be output from the addresses indicated by the pointers of the RSACPD_OUT_INFO type structure members (pcm_cf, pcm_lf, pcm_rf, pcm_ls, pcm_rs and pcm_lfe).

The figure below shows an example of output PCM format for 5.1ch.

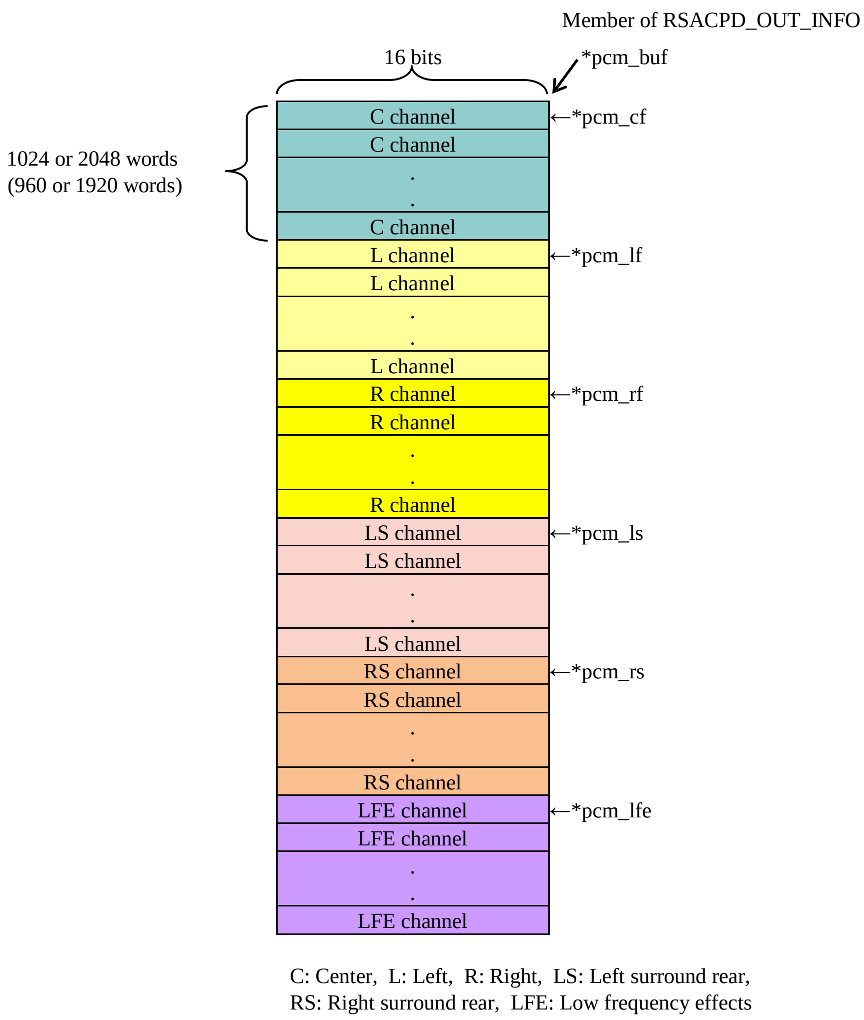


Figure 2.5 Structure of output PCM data (Example of 5.1ch)

3 API Specifications

3.1 API function list

Table 3-1 shows the definitions of API functions, the functional overviews and the necessity of execution of API functions for decoding of a bit stream.

Table 3-1 API function list

No.	Definition of function	Description	Necessary/arbitrary
1	int RSACPD_Open(RSACPD_AAC *aac, unsigned char *buf_adr, int buf_len, unsigned int (*RSACPD_GetData), int key)	Initialize this decode middleware.	Necessary
2	int RSACPD_SetPCEArea(RSACPD_AAC *aac, RSACPD_PCE *pce, int pce_cnt)	Set the area to obtain the PCE information.	Necessary in the case of following PCE channel configuration
3	int RSACPD_GetAdifHeader(RSACPD_AAC *aac, RSACPD_AdifHeader *header, int *bcnt)	Acquire the header of the bit stream in the ADIF-format.	Necessary in the case of ADIF-format
4	int RSACPD_GetAdtsHeader(RSACPD_AAC *aac, RSACPD_AdtsHeader *header, int *bcnt)	Acquire the header of the bit stream in the ADTS-format.	Necessary in the case of ADTS-format
5	int RSACPD_GetLoasInfo(RSAPCD_AAC *aac, RSACPD_LoasInfo *header, int *bcnt)	Acquire the header of the bit stream in the LOAS format	Necessary in the case of LOAS format
6	int RSACPD_SetFormat(RSACPD_AAC *aac, int sampleRateIdx, int decodingType)	Set the information to decode the bit stream in the RawDataStream-format.	Necessary in the case of RawDataStream-format
7	int RSACPD_Decode(RSACPD_AAC *aac, int *bcnt, RSACPD_OUT_INFO *outInfo, int *pnum)	Decode one block of the bit stream.	Necessary
8	int RSACPD_Skip(RSACPD_AAC *aac, int *bcnt)	Skip one block of the bit stream.	Arbitrary
9	int RSACPD_GetStatusCode(RSACPD_AAC *aac)	Return the status code after execution of the API function.	Arbitrary
10	int RSACPD_DecodeStatus(RSACPD_AAC *aac, int *decodeStatus)	Check the decoding of a block of the bit stream, or the decode status after skipping.	Arbitrary
11	int RSACPD_SetDecOpt(RSACPD_AAC *aac, int decopt)	Set the option for decoding function.	Arbitrary
12	int RSACPD_get_version(void)	Acquire the version of this decode middleware.	Arbitrary

13	int RSACPD_SetDSE (RSACPD_AAC *aac, RSACPD_DSE *dse, int dse_cnt)	Set the area for acquiring DSE (data_stream_element) contained in a bit stream.	Arbitrary
14	int RSACPD_InterleavePCM(short *pcm_l, short *pcm_r, int pcnt, short *outpcm)	Convert the non-interleaved PCM data output to each channel to a 2ch interleaved PCM data.	Arbitrary
15	int RSACPD_MatrixMixdown(RSACPD_AAC *aac, int* sel_std, int* mixdown_mode, RSACPD_OUT_INFO* outInfo, int scale)	Downmix a multi-channel to 2-channel stereo or monaural.	Arbitrary
16	int RSACPD_SetSAC(RSACPD_AAC *aac, RSACPD_SAC *sac)	Set the area for acquiring spatial audio coding side information necessary for MPEG Surround.	Arbitrary
17	int RSACPD_SetDRC(RSACPD_AAC *aac, RSACPD_DRC *drc, int hi, int lo, int ref_level)	Set the Dynamic Range Control information	Arbitrary

The formal arguments are set to make the explanation easily understood. They can be freely set.

< Input/Output (I/O) identification of arguments >

This user's manual identifies the input/output (I/O) of the arguments of API functions in accordance with the following rules:

I : The API function refers to the contents in the relevant argument or the area designated by the relevant argument (in the case of a pointer variable) (read only).

O : The API function sets the contents in the area designated by the relevant argument (pointer variable).

I/O : The API function refers/sets (updates) the contents in the area designated by the relevant argument (pointer variable).

3.2 Details of API function

This section describes the API functions of this middleware.

3.2.1 RSACPD_Open

<table>
<tr><td>Synopsis</td><td colspan="3">int RSACPD_Open(
    RSACPD_AAC *aac,
    unsigned char *buf_adr,
    int buf_len,
    unsigned int (*RSACPD_GetData),
    int key
)</td></tr>
<tr><td>Function</td><td colspan="3">Execute initializing process of the work area for this middleware.</td></tr>
<tr><td colspan="2">Argument</td><td>I/O</td><td>Description</td></tr>
<tr><td colspan="2">RSACPD_AAC *aac</td><td>O</td><td>Pointer to the RSACPD_AAC type structure</td></tr>
<tr><td colspan="2">unsigned char *buf_adr</td><td>I</td><td>Input buffer initial address</td></tr>
<tr><td colspan="2">int buf_len</td><td>I</td><td>Input buffer size</td></tr>
<tr><td colspan="2">unsigned int (*RSACPD_GetData)</td><td>I</td><td>Pointer to the user created function (call back function)</td></tr>
<tr><td colspan="2">int key</td><td>I</td><td>Reserved (set to 0)</td></tr>
<tr><td>Return value</td><td colspan="2">Macro name</td><td>Description</td></tr>
<tr><td>0</td><td colspan="2">RSACPD_RTN_GOOD</td><td>Successfully completed.</td></tr>
<tr><td>-1</td><td colspan="2">RSACPD_RTN_ERROR</td><td>Abnormally ends.</td></tr>
<tr><td>Description</td><td colspan="3"><Execution of this function>
This function shall be executed before using this middleware. This function initializes the RSACPD_AAC type structure of the work area of this middleware.

<Details of function>
The fourth argument is the pointer to the user created call back function. User can use independent user created call back functions for each decoding task.

The fifth argument is reserved and need to set to 0.

<Notes>
(1) Set the input buffer size to 1 byte or more, or the function will end with an error.
(2) This function shall be called whenever a different bit stream is to be decoded. (For example, to playback different songs, this function shall be called.)
(3) The work area (RSACPD_AAC type structure) and the input buffer must be reserved by the application program. For the details, see Section 7.1 “RSACPD_AAC type structure”.
(4) If an error occurs on the RSACPD_Decode() or the RSACPD_Skip(), make sure to call this API function to continue decode processing.</td></tr>
</table>

3.2.2 RSACPD_SetPCEArea

Synopsis	int RSACPD_SetPCEArea(RSACPD_AAC *aac, RSACPD_PCE *pce, int pce_cnt)
Function	Set the area to obtain the PCE information.

Argument	I/O	Description
RSACPD_AAC *aac	I/O	Pointer to the RSACPD_AAC type structure
RSACPD_PCE *pce	O	Pointer to the RSAPCD_PCE type structure
int pce_cnt	I	Number of available PCE information (setting value: 1 to 16)

Return value	Macro name	Description
0	RSACPD_RTN_GOOD	Successfully completed.
-1	RSACPD_RTN_ERROR	Abnormally ends.

Description

<Execution of this function>
Execute this function just after executing RSACPD_Open() to set memory area for PCE information.

<Details of function>
PCE information exists in the PCE element of raw_data_block, in the ADIF header or LOAS header.

PCE has been detected during execution of RSACPD_GetAdifHeader(), RSACPD_GetLoasInfo(), RSACPD_Decode() or RSACPD_Skip() function, PCE information will be output to an RSACPD_PCE type structure, pce.
Whether or not to use the PCE information acquired by this middleware for decoding depends on the input data format and user's selection of PCE by RSACPD_SetDecOpt() function (refer to Section 3.2.11 RSACPD_SetDecOpt for how to select PCE). The behaviour related to PCE is explained below (Refer to Table 3-2 Referred PCE and bitstream format)

(1) ADIF format

- If user selects PCE by RSACPD_SetDecOpt() function, the decoder refers to the selected PCE. If PCE is not selected, the first PCE is referred.
- If the selected PCE is not matched with any PCEs in the ADIF header, the decoder ignores all PCEs and follows default channel configuration.

(2) LOAS format

- If PCE exists in the LOAS header (unique PCE), the decoder always follows the PCE information
- If PCEs exists in the raw_data_block, the decoder follows the matched PCE selected by RSACPD_SetDecOpt() function or follows the first PCE (if user does not select PCE). If the selected PCE is not matched with any PCEs, the decoder ignores all PCEs and follows default channel configuration.

(3) ADTS format

- If PCEs exists in the raw_data_block, the decoder follows the matched PCE selected by RSACPD_SetDecOpt() function or follows the first PCE (if user does not select PCE). If the selected PCE is not matched with any PCEs, the decoder ignores all PCEs and follows default channel configuration.

<Notes>
(1) For information about default channel configuration, please refer to Table 5-1.
(2) For the RSACPD_PCE type structure, an area will be reserved by the application program. For the details, see Section 7.2.
  - Declare the RSACPD_PCE structure, pce, of the second argument as an array of the number of elements, pce_cnt, of the third argument.
  - For ADTS/LOAS/RawDataStream-format, the third argument pce_cnt should be set 1 (prepare 1 RSACPD_PCE type strucure) because the middleware always stores only one PCE information in the format and the remainings are ignored.

	- If the value of the third argument pce_cnt is more than 16 or less than 1, this function will end with an error (Status code: RSACPD_ERR_PCECNT). - If bit stream has more than one PCE, PCE can be chosen by setting the value (0 to 15) of the instance tag of the PCE to be used by the RSACPD_SetDecOpt() function (Section 3.2.11). - If the number of PCEs in the ADIF header is larger than the value of the third argument pce_cnt in the case of the ADIF-format, the RSACPD_GetAdifHeader() function will end with an error (Status code: RSACPD_ERR_PCECNT). (3) This middleware supports only the matrix mixdown function. When detecting other downmix information, it executes the following operation. - mono_mixdown_present should be 0. If another setting is detected, the RSACPD_Decode() function will end with an error. (Status code: RSACPD_ERR_AUDIO_MODE). - stereo_mixdown_present should be 0. If another setting is detected, the RSACPD_Decode() function will end with an error. (Status code: RSACPD_ERR_AUDIO_MODE).

Table 3-2 Referred PCE and bitstream format

Bitstream format	**PCE Location**	**User's Selection of PCE Instance Tag**	**User's Setting is Matched with PCE in Stream**	**Got PCE**	**Note**
ADIF	header (Many PCE)	No	-	1st PCE	-
		Yes	Yes	Matched PCE	-
			No	Ignore ALL PCE	-
	header and raw_data_block	-	-	-	ERROR
LOAS	header (1 PCE)	-	-	Unique PCE	-
	raw_data_block	No	-	1st PCE	-
		Yes	Yes	Matched PCE	-
			No	Ignore ALL PCE	-
	header and raw_data_block	-	-	-	ERROR
ADTS RawDataStream	raw_data_block	No	-	1st PCE	-
		Yes	Yes	Matched PCE	-
			No	Ignore ALL PCE	-

- : it does not affect the behavior.

3.2.3 RSACPD_GetAdifHeader

<table>
<tr><td>Synopsis</td><td colspan="3">int RSACPD_GetAdifHeader(
    RSACPD_AAC *aac,
    RSACPD_AdifHeader *header,
    int *bcnt
)</td></tr>
<tr><td>Function</td><td colspan="3">Acquire the header of the bit stream in the ADIF-format.</td></tr>
<tr><td colspan="2">Argument</td><td>I/O</td><td>Description</td></tr>
<tr><td colspan="2">RSACPD_AAC *aac</td><td>I/O</td><td>Pointer to the RSACPD_AAC type structure</td></tr>
<tr><td colspan="2">RSACPD_AdifHeader *header</td><td>O</td><td>Pointer to the RSACPD_AdifHeader type structure</td></tr>
<tr><td colspan="2">int *bcnt</td><td>O</td><td>Byte count of input data used for acquiring header information</td></tr>
<tr><td>Return value</td><td colspan="2">Macro name</td><td>Description</td></tr>
<tr><td>0</td><td colspan="2">RSACPD_RTN_GOOD</td><td>Successfully completed.</td></tr>
<tr><td>-1</td><td colspan="2">RSACPD_RTN_ERROR</td><td>Abnormally ends.</td></tr>
<tr><td>Description</td><td colspan="3"><Execution of this function>
In the case of an ADIF-format bit stream, execute this function without fail. For the decoding flowchart, see Section 6.3 “Decoding in ADIF-format”.

<Details of function>
This function acquires the ADIF header information and stores it in the RSACPD_AdifHeader type structure.

<Notes>
(1) The area for the RSACPD_AdifHeader type structure will be secured with the application program. For more details, see Section 7.3 RSACPD_AdifHeader type structure”.
(2) Before executing this function, execute the RSACPD_SetPCEArea() function to set the area to store the PCE information. Please refer to Section 3.2.2 RSACPD_SetPCEArea for details of PCE behavior.</td></tr>
</table>

3.2.4 RSACPD_GetAdtsHeader

Synopsis	int RSACPD_GetAdtsHeader(
	    RSACPD_AAC *aac,
	    RSACPD_AdtsHeader *header,
	    int *bcnt
	    )		

Function	Acquire the header of the bit stream in the ADTS-format.		
Argument		I/O	Description
RSACPD_AAC *aac		I/O	Pointer to the RSACPD_AAC type structure
RSACPD_AdtsHeader *header		O	Pointer to the RSACPD_AdtsHeader type structure
int *bcnt		O	Byte count of input data used for acquiring header information
Return value	Macro name		Description
0	RSACPD_RTN_GOOD		Successfully completed.
1	RSACPD_RTN_CHECK		Warning
-1	RSACPD_RTN_ERROR		Abnormally ends.
Description	<Execution of this function> In the case of an ADTS-format bit stream, execute this function without fail before the RSACPD_Decode() function or RSACPD_Skip() function after the RSACPD_Open() function. After once the RSACPD_Decode() function or the RSACPD_Skip() function is executed, execute this function after confirming if the ADTS frame does not contain unprocessed blocks (raw_data_block) by the RSACPD_DecodeStatus() function. For the decoding flowchart, see Section 6.2 “Decoding in ADTS-format”. <Details of function> This function acquires the ADTS header information and stores it in the RSACPD_AdtsHeader type structure. <Notes> (1) The area for the RSACPD_AdtsHeader type structure must be reserved by the application program. For the details, see Section 7.4 “RSACPD_AdtsHeader type structure”. (2) If a pseudo-syncword[2] is detected, the next normal syncword cannot be detected, and the middleware may end with an error. (3) This function ends abnormally (Status code : RSACPD_ERR_STREAM_DATA), when it does not find a syncword after reading more than 4610 bytes. (4) The ADTS format is in conformance with standards beginning from ISO/IEC 14496-3:2001 / Cor 2:2002. Operation cannot be guaranteed for inputs of formats which predate the above standard dates.		

[2]bit strings of 0xFFF in a bit stream such as spectral-data which is the same as the sync-word but not sync-word

3.2.5 RSACPD_GetLoasInfo

Synopsis	int RSACPD_GetLoasInfo(RSACPD_AAC *aac, RSACPD_LoasInfo *header, int *bcnt)
Function	Acquire the header of the bit stream in LOAS format.

Argument	I/O	Description
RSACPD_AAC *aac	I/O	Pointer to the RSACPD_AAC type structure
RSACPD_LoasInfo *header	O	Pointer to the RSACPD_LoasInfo type structure
int *bcnt	O	Byte count for input data used for acquiring header information

Return value	Macro name	Description
0	RSACPD_RTN_GOOD	Successfully completed.
1	RSACPD_RTN_CHECK	Warning
-1	RSACPD_RTN_ERROR	Abnormally ends.

Description

<Execution of this function>
In the case of an LOAS-format bit stream, execute this function without fail before the RSACPD_Decode() function or RSACPD_Skip() function after the RSACPD_Open() function.
For the decoding flowchart, see Section 6.4 “Decoding in LOAS-format”.

<Details of function>
This function acquires the LOAS header information and stores it in the RSACPD_LoasInfo type structure

<Notes>
(1) The area for the RSACPD_LoasInfo type structure must be reserved by the application program. For the details, see Section 7.5 “RSACPD_LoasInfo type structure”.
(2) In case of LOAS format, the header may include or not include PCE. If the header includes PCE, this middleware supports only 1 PCE. If the header includes more than 1 PCE, this function will return error (Status code: RSACPD_ERR_LOAS_INFO). In case the header includes 1 PCE, any PCE found in the raw_data_block will make RSACPD_Decode() or RSACPD_Skip() function return error (Status code: RSACPD_ERR_PCE_LOC). Please refer to Section 3.2.2 RSACPD_SetPCEArea for details of PCE behaviour.
(3) This function ends abnormally (Status code: RSACPD_ERR_STREAM_DATA), when it does not find a syncword after reading more than 4610 bytes.

3.2.6 RSACPD_SetFormat

Synopsis	int RSACPD_SetFormat(RSACPD_AAC *aac, int sampleRateIdx, int decodingType)
Function	Set the information to decode the bit stream in the RawDataStream-format.

Argument	I/O	Description
RSACPD_AAC *aac	I/O	Pointer to the RSACPD_AAC type structure
int sampleRateIdx	I	Setting of sampling_frequency_index (see Table 3-3)
int decodingType	I	Decoding type to 1024/2048 or 960/1920 PCM samples per frame (Setting value: 0 or 1) 0: Number of output PCM samples per frame is 1024 or 2048 per channel 1: Number of output PCM samples per frame is 960 or 1920 per channel

Return value	Macro name	Description
0	RSACPD_RTN_GOOD	Successfully completed.
1	RSACPD_RTN_CHECK	Warning
-1	RSACPD_RTN_ERROR	Abnormally ends.

Description

<Execution of this function>
In the case of a RawDataStream-format bit stream, execute this function without fail. For the decoding flowchart, see Section 6.1 "Decoding in RawDataStream-format”.

<Details of function>
This function sets the sampling frequency necessary for decoding or skipping of a RawDataStream -format bit stream.
This function will choose the 1024/2048 or 960/1920 samples output for the RawDataStream format bit stream base on the third argument decodingType. If 0 is set to this argument, the decoder will choose the 1024/2048 output samples (per channel) decoding type. If 1 is set to this argument, the decoder will choose the 960/1920 output samples (per channel) decoding type.

<Notes>
(1) The application program must be informed of the sampling frequency of the bit stream in advance.
(2) As the value in the second argument sampleRateIdx, specify a value conforming to the contents of the RawDataStream-format bit stream to be decoded.
(3) If a PCE element is detected during decoding, priority will be given to the sampling frequency value in the PCE information.
(4) If the third argument is passed with other values than 0 or 1, the decoder will output a warning message (Status code: RSACPD_WARN_INVALID_DECODE_TYPE) and choose the default 1024/2048 output samples decoding type.

Table 3-3 Sampling frequency list (Sampling_frequency_index)

Value of Sampling_frequency_index	Sampling frequency	
	AAC coded bit stream	aacPlus coded bit stream
0x0	96000	Not supported.
0x1	88200	Not supported.
0x2	64000	Not supported.
0x3	48000	Not supported.
0x4	44100	Not supported.
0x5	32000	Not supported.
0x6	24000	24000
0x7	22050	22050
0x8	16000	16000
0x9	12000	12000
0xa	11025	11025
0xb	8000	8000
0xc	Not supported.	Not supported.
0xd	Not supported.	Not supported.
0xe	Not supported.	Not supported.
0xf	Not supported.	Not supported.

(Note 1) The aacPlus coded bit stream is upsampled during decoding. Therefore, the sampling frequency for output PCM data is twice the shown frequency. However, in the downsample SBR mode, sampling frequency is the same as the shown sampling frequency.

(Note 2) When the AAC upsample mode is used for decoding an AAC coded bit stream, the sampling frequency of output PCM data is twice the shown frequency.

3.2.7 RSACPD_Decode

Synopsis	int RSACPD_Decode(RSACPD_AAC *aac, int *bcnt, RSACPD_OUT_INFO *outInfo, int *pnum)
Function	Decode one block (raw_data_block) of the bit stream.

Argument	I/O	Description
RSACPD_AAC *aac	I/O	Pointer to the RSACPD_AAC type structure
int *bcnt	O	Byte count of input data used for decoding
RSACPD_OUT_INFO *outInfo	I/O	Pointer to the RSACPD_OUT_INFO type structure
int *pnum	O	Number of output words of PCM data per channel

Return value	Macro name	Description
0	RSACPD_RTN_GOOD	Successfully completed.
1	RSACPD_RTN_CHECK	Warning
-1	RSACPD_RTN_ERROR	Abnormally ends.

Description

<Execution of this function>
To decode a bit stream, execute this function without fail.
For the decoding flowchart, see Section 6 “Decoding Flowchart”

<Details of function>
This function decodes a bit stream in block (raw_data_block) units and outputs PCM data.
Before calling this function, output PCM data area of necessary size (2048 × a number of audio channels) should be allocated by the application program and set the initial address of the PCM data area to the *pcm_buf, member of the RSACPD_OUT_INFO type structure which is also allocated by the application program.

After decoding is completed, the number of channels of output PCM data, channel mode and output PCM data of each channel are set to the member of RSACPD_OUT_INFO type structure in accordance with channel configuration.
For the output PCM data structure, see Section 2.2 "Output Data Format”.
For the output contents of RSACPD_OUT_INFO type structure, see Table 7-6.

The number of output words of the PCM data is shown in the table below. The number of output words of the PCM data varies depending on the coded bit stream type specified by RSACPD_SetDecOpt() function (Refer to Section 3.2.11 “RSACPD_SetDecOpt”). In the case of abnormal termination, 0 or undefined value is set.

Bit stream type	Decode mode	Value of *pnum	Output sampling frequency
AAC	Normal	1024/960	Input sampling frequency
	AAC upsample	2048/1920	Input sampling frequency × 2
aacPlus V1/V2	Normal	2048/1920	Input sampling frequency × 2
	Downsample SBR	1024/960	Input sampling frequency
	Forced AAC	1024/960	Input sampling frequency

During decoding aacPlus V1/V2 bit stream with normal mode, if decoding of AAC ends normally and an error, such as the SBR-CRC error, is detected during decoding of SBR, upsampled PCM data with the concealed error will be automatically output.

<Notes>
(1) All PCE elements in raw_data_block must appear before other elements. If any PCE element is detected after any other element, the middleware will end abnormally.
(Status code: RSACPD_ERR_PCE_LOC).
(2) In the case of the ADTS-format, if the value of the RSACPD_AdtsHeader type structure member, frame_length, does not conform to the actually decoded ADTS frame length, a warning is returned (Status code: RSACPD_WARN_ERR_ADTS_LEN). In this case, the PCM data is output, but the output results are not guaranteed.

	(3) This middleware supports the ADTS-format compliant with the ISO/IEC 14496-3:2005 or later. The operation cannot be guaranteed if the format earlier than those listed above is input. (4) If PCE element is detected in raw_data_block of ADIF stream, this function will end abnormally (Status code: RSACPD_ERR_PCE_LOC). If PCE element is detected in both LOAS header and raw_data_block, this function will also end abnormally (Status code: RSACPD_ERR_PCE_LOC). (5) If the RSACPD_SetPCEArea() function is not called, PCE information will be ignored.

3.2.8 RSACPD_Skip

<table>
<tr><td>Synopsis</td><td colspan="3">int RSACPD_Skip(
    RSACPD_AAC *aac,
    int *bcnt
)</td></tr>
<tr><td>Function</td><td colspan="3">Skip one block (raw_data_block) of the bit stream.</td></tr>
<tr><td colspan="2">Argument</td><td>I/O</td><td>Description</td></tr>
<tr><td colspan="2">RSACPD_AAC *aac</td><td>I/O</td><td>Pointer to the RSACPD_AAC type structure</td></tr>
<tr><td colspan="2">int *bcnt</td><td>O</td><td>Byte count of input data used for skip processing</td></tr>
<tr><td>Return value</td><td colspan="2">Macro name</td><td>Description</td></tr>
<tr><td>0</td><td colspan="2">RSACPD_RTN_GOOD</td><td>Successfully completed.</td></tr>
<tr><td>1</td><td colspan="2">RSACPD_RTN_CHECK</td><td>Warning</td></tr>
<tr><td>-1</td><td colspan="2">RSACPD_RTN_ERROR</td><td>Abnormally ends.</td></tr>
<tr><td>Description</td><td colspan="3"><Execution of this function>
To skip a bit stream, execute this function without fail.

<Details of function>
This function only analyzes the bit stream, does not generate and output PCM data. This function is used to fast-forward audio data by one block (raw_data_block).

<Notes>
(1) All PCE elements in raw_data_block must appear before other elements. If any PCE element is detected after any other element, the middleware will ends abnormmaly (Status code: RSACPD_ERR_PCE_LOC).
(2) In the case of the ADTS-format, if the value of the RSACPD_AdtsHeader type structure member, frame_length, does not conform to the actually decoded ADTS frame length, a warning is returned. (Status code: RSACPD_WARN_ERR_ADTS_LEN)
(3) This middleware supports the ADTS-format compliant with the ISO/IEC 14496-3:2005 or later. The operation cannot be guaranteed if the format earlier than those listed above is input.
(4) If PCE element is detected in raw_data_block of ADIF stream, this function will end abnormally (Status code: RSACPD_ERR_PCE_LOC).
If PCE element is detected in both LOAS header and raw_data_block, this function will also end abnormally (Status code: RSACPD_ERR_PCE_LOC).
(5) If the RSACPD_SetPCEArea() function is not called, PCE information will be ignored.</td></tr>
</table>

3.2.9 RSACPD_GetStatusCode

Synopsis	int RSACPD_GetStatusCode(RSACPD_AAC *aac)
Function	Return the status code after execution of an API function.

Argument	I/O	Description
RSACPD_AAC *aac	I	Pointer to the RSACPD_AAC type structure

Return value	Description
See Section 8 “List of status codes”	Status code

Description	<Execution of this function> To acquire detailed status information when the return value of an API function is an abnormal termination (RSACPD_RTN_ERROR) or a warning termination (RSACPD_RTN_CHECK), execute this function. <Details of function> This function is designed to check the status code after execution of an API function.

3.2.10 RSACPD_DecodeStatus

<table>
<tr><td>Synopsis</td><td colspan="3">int RSACPD_DecodeStatus(
RSACPD_AAC *aac,
int *decodeStatus
)</td></tr>
<tr><td>Function</td><td colspan="3">Check the decode status after decoding or skipping one block (raw_data_block) of the bit stream.</td></tr>
<tr><td colspan="2">Argument</td><td>I/O</td><td>Description</td></tr>
<tr><td colspan="2">RSACPD_AAC *aac</td><td>I</td><td>Pointer to the RSACPD_AAC type structure</td></tr>
<tr><td colspan="2">int *decodeStatus</td><td>O</td><td>Flags to set decoding status
Sets every status in a bit field. The values to set to the bit field are listed in Table 3-4.</td></tr>
<tr><td>Return value</td><td colspan="2">Macro name</td><td>Description</td></tr>
<tr><td>0</td><td colspan="2">RSACPD_RTN_GOOD</td><td>Successfully completed.</td></tr>
<tr><td>-1</td><td colspan="2">RSACPD_RTN_ERROR</td><td>Abnormally ends.</td></tr>
<tr><td>Description</td><td colspan="3"><Execution of this function>
To acquire the input bit stream type, execute this function.
For the decoding flowchart, see Section 6 “Decoding Flowchart”.

<Details of function>
This function checks the decode status after RSACPD_Decode() or RSACPD_Skip() is executed. The function can check the following decode statuses.

(1) raw_data_block decode continuation check bit
If raw_data_block which has not been decoded exists in the ADTS frame after the completion of raw_data_block decoding (or skipping), 1 is set in the bit field 4 of the second argument.
In the case of decoding in a format other than the ADTS format, normally, 0 is set.

(2) PCE acquisition flag
If PCE exits in raw_data_block of ADTS, LOAS and RawDataStream format, 1 is set in the bit field 5.

(3) SBR detection flag
When SBR data is detected in the input stream, 1 is set in the bit field 13.
When this flag is set to 1, the input stream is an aacPlus coded bit stream. When it is 0, the input stream is an AAC coded bit stream.

(4) Parametric Stereo (PS) detection flag
If PS data is detected in the input stream, 1 is set in the bit field 14.
When this flag is 1, the input stream is an aacPlus V2 coded bit stream. When it is 0, the input stream conforms to the value of the SBR detection flag.

<Notes>
(1) The number of raw_data_block(s) contained in one ADTS frame can be obtained by referring to the number_of_raw_data_blocks_in_frame member in the RSACPD_AdtsHeader structure. Therefore, it is possible to decode the block by executing RSACPD_GetAdtsHeader() on the application program side without executing this function.
(2) In the case of decoding in ADTS format, RSACPD_Decode() function or RSACPD_Skip() function may not be executed with the condition that raw_data_block decode continuation check bit is 0, it ends with an error (Status code: RSACPD_ERR_NO_RAW_ DATA_BLOCK). In this case, the value of this bit is unstable.</td></tr>
</table>

Table 3-4 Bit fields of decode status

Bit field	Flag	Value
14	Parametric Stereo (PS) detection flag	0: No PS data 1: With PS data
13	SBR detection flag	0:No SBR data 1:With SBR data
5	PCE acquisition flag	0:PCE not acquired 1: PCE acquired
4	raw_data_block decode continuation check bit (in ADTS format)	0: No unprocessed raw_data_block in ADTS frame, or bit stream not in ADTS-format 1: ADTS frame containing unprocessed raw_data_block
Others	Reserved.	0

31 30 … … 18 17 16 15	14	13	12 11 … … … 6	5	4	3 2 1 0
Reserved (*)			Reserved (*)			Reserved (*)

(*):Reservation bit field ("0" must be set.)

3.2.11 RSACPD_SetDecOpt

<table>
<tr><td>Synopsis</td><td colspan="3">int RSACPD_SetDecOpt(
    RSACPD_AAC *aac,
    int decopt
)</td></tr>
<tr><td>Function</td><td colspan="3">Set the option for decoding function.</td></tr>
<tr><td colspan="2">Argument</td><td>I/O</td><td>Description</td></tr>
<tr><td colspan="2">RSACPD_AAC *aac</td><td>I/O</td><td>Pointer to the RSACPD_AAC type structure</td></tr>
<tr><td colspan="2">int decopt</td><td>I</td><td>Flags to set decode option
Every option is set in bit field. The values to set to the bit field are listed in Table 3-5.</td></tr>
<tr><td>Return value</td><td colspan="2">Macro name</td><td>Description</td></tr>
<tr><td>0</td><td colspan="2">RSACPD_RTN_GOOD</td><td>Successfully completed.</td></tr>
<tr><td>1</td><td colspan="2">RSACPD_RTN_CHECK</td><td>Warning</td></tr>
<tr><td>-1</td><td colspan="2">RSACPD_RTN_ERROR</td><td>Abnormally ends.</td></tr>
<tr><td>Description</td><td colspan="3"><Execution of this function>
To set decode options, execute this function before executing RSACPD_Decode() for the first time after executing RSACPD_Open() without fail.
The user is responsible for switching decode options for each decoding. If this function is executed in each case of decoding, the function will return a warning after making decode options enable.

<Details of function>
This function sets for the decode option. For the details, see Table 3-5.
The following is the settable decode option:

(1) Forced AAC decode mode
If the forced AAC decode mode is enabled, decoded data for aacPlus will be ignored, aacPlus decoding will not be executed, and only ACC decoding will be executed.

(2) Downsample SBR mode
When the input stream is an aacPlus coded stream, if the Downsample SBR mode setting flag is made effective, filtering without upsampling will be executed during aacPlus decoding.
Note that if the forced AAC decode mode is effective, the downsample SBR mode setting is ineffective.

(3) AAC upsample mode
When the input stream is an AAC or aacPlus coded stream and the forced AAC decode mode is effective, if the AAC upsample mode setting flag is made effective, filtering with upsampling will be executed during AAC decoding. This option is available when the sampling frequency of the input bit stream is less than 32 kHz

(4) Enabling selection of program
If this setting is enabled, it is possible to select a program to use when multi-programs exist in the bit stream. Please refer “(5) Selection of program to be used” to set the program number

(5) Selection of program to be used
If “(4) Enabling selection of program” is enabled, it is possible to specify the program (PCE information) to be used by setting the instance tag (0 to 15) of the PCE to be used in the bit fields [11:8] of the decode option setting flag.
For details of PCE behavior, please refer to Section 3.2.2 RSACPD_SetPCEArea.

(6) Selection of SBR processing mode
When the input stream is an aacPlus coded stream, the SBR processing mode can be selected by making the SBR processing mode selection flag effective.
When the SBR processing mode selection flag (bit 13 is set to 0 and bit 12 is set 1), decoding will be performed normally in the HQ-SBR (High Quality SBR: high-quality band expansion method) mode. When (bit 13 is set to 0 and bit 12 is set 0), decoding will be automatically switched according to the data type.

(Note 1) The case that SBR processing mode setting flag (bit 13) is set to 1 is reserved and not covered under warranty.</td></tr>
</table>

(7) Error conceal mode setting

If the error conceal mode setting flag is made effective, the function will not end with an error even if an error is detected in the input stream, and it will execute the error conceal processing and RSACPD_Decode() function outputs PCM data.
The output PCM data contains the data in the overlap buffer in the immediately preceding frame.

When error conceal mode is effective and RSACPD_Decode() function and RSACPD_Skip() function detect error, return values are as follows.

Input data format	Return Value
ADTS-format	RSACPD_RTN_CHECK
ADIF-format	RSACPD_RTN_ERROR
LOAS-format	RSACPD_RTN_CHECK
RawDataStream-format	RSACPD_RTN_ERROR

Note that when the end of the input data detected, RSCADP_RTN_ERROR is always returned.

RSACDP_Decode() function and RSACPD_Skip() function behaves like below, when it detects the end of the input data (data empty) under error conceal mode is effective.

- If data empty is found in the beginning of bit stream, error conceal is not done and RSCAPD_Decode() function does not output PCM data.
- If data empty is found on the way of decoding, error conceal is done and RSACPD_Decode() function outputs PCM data.

Though PCM data is output at an error detected frame when error conceal mode is effective, it is impossible to continue decoding after the error frame for ADIF/RawDataStream-format. In case of ADTS-format and LOAS format, it is possible to decode continuously by executing RSACPD_GetAdtsHeader() and RSACPD_GetLoasInfo() function until it ends normally.
Note that some frames might be read through without decoding or pseudo-syncword might be detected.

(Note 1) Error conceal is not be done at the first frame or the frame just after skipping and PCM data is not output.

(Note 2) When decoding parametric stereo (PS) data and error conceal is done, RSAPD_Decode() function outputs monaural PCM data. To get stereo data, it is necessary to copy PCM data of L channel to R channel or execute the RSACPD_InterleavePCM() function.

<Note>

This API function must be executed to change the default operations of RSAPD_Decode() function. When this function is not executed, RSACPD_Decode() function operates by default as shown below.

Decode option	Default setting
(1) Forced AAC decode mode	Ineffective
(2) Downsample SBR mode	Ineffective
(3) AAC upsample mode	Ineffective
(4) Enabling selection of program	Ineffective
(5) Selection of program to be used	Ineffective
(6) SBR processing mode setting	Automatic switching
(7) Error conceal mode setting	Ineffective

Table 3-5 Bit fields of decode options

Bit field	Decode option	Value (meaning)
31-17	Reserved.	(Be sure to set 0.)
16	Error conceal mode setting flag	0: Error conceal processing will not be executed. 1: Error conceal processing will be executed.
15-14	Reserved.	(Be sure to set 0.)
13-12	SBR processing mode selection flag	00: SBR decoding will be automatically switched according to the data type. In the monaural or parametric stereo mode, decoding will be performed in the HQ-SBR (High-Quality-SBR) mode. In other cases, SBR decoding will be performed in the LP-SBR (Low-Power-SBR) mode. 01: Normally, SBR decoding in HQ-SBR mode for all channel mode (bit 13 is set 0, bit 12 is set 1) 10 – 11 : Reserved
11-8	Selection of program to be used	Value of instance tag of PCE to be used
7	Flag to set up-sampling for AAC bit stream or forced AAC mode	0: Disable to do up-sampling for AAC coded bit stream. 1: Enable to do up-sampling for AAC coded bit stream up to 24kHz (input)
6-5	Reserved.	(Be sure to set 0.)
4	Downsample SBR mode setting flag	0:Downsample SBR mode is ineffective. 1:Downsample SBR mode is effective.
3-2	Reserved.	(Be sure to set 0.)
1	Enabling selection of program	0: Disable selection of PCE 1: Enable selection of PCE
0	Forced AAC decode mode setting flag	0: Forced AAC decode mode is ineffective. 1: Forced AAC decode mode is effective.

31	30	…	…	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved (*)								Reserved (*)									Reserved (*)			Reserved (*)			

(*) Make sure to set 0 to the reservation bit field. If any other value is set, the operation is not covered under warranty.

Table 3-6 Decode options and decoding operations

Input stream type \ Setting bit								
	AAC upsample mode	O	-	-	O	O	O	-
	Downsample SBR mode	-	O	-	O	-	O	O
	Forced AAC decode mode	-	-	O	-	O	O	O
AAC	fs ≤ 24kHz	AAC upsample mode	---	---	AAC upsample mode	AAC upsample mode	AAC upsample mode	---
	32kHz ≤ fs ≤ 48kHz	---	---	---	---	---	---	---
	64kHz ≤ fs ≤ 96kHz	---	---	---	---	---	---	---
aacPlus V1/V2	fs ≤ 24kHz	---	Downsample SBR mode	Forced AAC decode mode	Downsample SBR mode	Forced AAC decode mode and AAC upsample mode	Forced AAC decode mode and AAC upsample mode	Forced AAC decode mode (Note 1)
	32kHz ≤ fs ≤ 48kHz	Downsample SBR mode (Note 2)	Downsample SBR mode (Note 2)	Forced AAC decode mode	Downsample SBR mode (Note 2)	Forced AAC decode mode	Forced AAC decode mode (Note 1)	Forced AAC decode mode (Note 1)
	64kHz ≤ fs ≤ 96kHz	Forced AAC decode mode (Note 3)	Forced AAC decode mode (Note 3)	Forced AAC decode mode	Forced AAC decode mode (Note 3)	Forced AAC decode mode	Forced AAC decode mode	Forced AAC decode mode

* The sampling frequencies (fs) indicate the input bit stream sampling frequencies (sampling frequencies in the AAC part).

(Note 1) When the forced AAC mode and the downsample SBR mode are simultaneously specified, the downsample mode is ineffective.

(Note 2) When the sampling frequency in the SBR part is higher than 48 kHz, decoding will be executed in the downsample SBR mode irrespective of the downsample SBR mode setting. A warning will be set in the status code.

(Note 3) When the sampling frequency in the SBR part is higher than 96 kHz, decoding will be executed in the forced AAC decode mode irrespective of the forced AAC decode mode setting. A warning will be set in the status code.

O : Option is made effective.
- : Option is made ineffective.
--- : Option setting does not affect the decoding operation.

Table 3-7 Decode modes and output sampling frequencies

Input stream type	Setting bit								
	AAC upsample mode	-	O	-	-	O	O	O	-
	Downsample SBR mode	-	-	O	-	O	-	O	O
	Forced AAC decode mode	-	-	-	O	-	O	O	O
AAC	24 kHz	24 kHz	**48 kHz**	24 kHz	24 kHz	**48 kHz**	**48 kHz**	**48 kHz**	24 kHz
	48 kHz	48 kHz	48 kHz	48 kHz	48 kHz	48 kHz	48 kHz	48 kHz	48 kHz
	96 kHz	96 kHz	96 kHz	96 kHz	96 kHz	96 kHz	96 kHz	96 kHz	96 kHz
aacPlus V1/V2	24 kHz (Note 1)	48 kHz	48 kHz	**24 kHz**	**24 kHz**	**24 kHz**	48 kHz	48 kHz	**24 kHz**
	48 kHz (Note 2)	48 kHz (Note 3)	48 kHz (Note 3)	48 kHz (Note 3)	48 kHz	48 kHz (Note 3)	48 kHz	48 kHz	48 kHz
	96 kHz	96 kHz (Note 4)	96 kHz (Note 4)	96 kHz (Note 4)	96 kHz	96 kHz (Note 4)	96 kHz	96 kHz	96 kHz

* The sampling frequencies indicate the input bit stream sampling frequencies (sampling frequencies in the AAC part).

(grey) : The output sampling frequencies of AAC and aacPlus streams at the same input sampling frequency are different.

Bold : The decode option affects the output sampling frequency.

Underline : The output sampling frequency is automatically affected regardless of the decode option setting.

(Note 1) Sampling frequency in SBR part: 48 kHz

(Note 2) Sampling frequency in SBR part: 96 kHz

(Note 3) When the sampling frequency in the SBR part is higher than 48 kHz, decoding will be executed in the downsample SBR mode irrespective of the downsample SBR mode setting. A warning will be set in the status code.

(Note 4) When the sampling frequency in the SBR part is higher than 96 kHz, decoding will be executed in the forced AAC decode mode irrespective of the forced AAC decode mode setting. A warning will be set in the status code.

O : Option is made effective.
\- : Option is made ineffective.

3.2.12 RSACPD_get_version

<table>
<tr><td>Synopsis</td><td colspan="2">int RSACPD_get_version(void)</td></tr>
<tr><td>Function</td><td colspan="2">Return the version number of this middleware.</td></tr>
<tr><td>Argument</td><td>I/O</td><td>Description</td></tr>
<tr><td>None</td><td>–</td><td>–</td></tr>
<tr><td colspan="2">Return value</td><td>Description</td></tr>
<tr><td colspan="2">0x10010112</td><td>Returns the version number in 0xabbbccdd
a :Version number
bbb :Revision number
cc :Build number
dd :Reserved</td></tr>
<tr><td>Description</td><td colspan="2"><Execution of this function>
To acquire the version number of this middleware, execute this function.

<Details of function>
This function returns the version number of this middleware as the return value.</td></tr>
</table>

3.2.13 RSACPD_SetDSE

Synopsis	int RSACPD_SetDSE (
	        RSACPD_AAC *aac,
	        RSACPD_DSE *dse,
	        int dse_cnt
	        )

Function	Set the area for acquiring DSEs (data_stream_element) contained in a bit stream.

Argument	I/O	Description
RSACPD_AAC *aac	I/O	Pointer to the RSACPD_AAC type structure
RSACPD_DSE *dse	O	Pointer to the RSACPD_DSE type structure
int dse_cnt	I	Number of obtainable DSEs (setting value: 1 to 16)

Return value	Macro name	Description
0	RSACPD_RTN_GOOD	Successfully completed.
-1	RSACPD_RTN_ERROR	Abnormally ends.

Description

<Execution of this function>
When it is necessary to acquire downmix information conforming to DVB-T standard in DSE (data_stream_element) in a bit stream, execute this function before executing RSACPD_Decode() or RSACPD_Skip() for the first time after executing RSACPD_Open() or RSACPD_SetDecOpt().

<Details of function>
When DSEs are detected during execution of RSACPD_Decode() or RSACPD_Skip(), the DSEs information is output to the RSACPD_DSE structure, dse.

<Notes>

(1) The area for the RSACPD_DSE type structure must be secured by the application program. For the details, see Section 7.7 “RSACPD_DSE type structure”.
  - Declare the RSACPD_DSE structure, dse, of the second argument as an array of the number of elements, dse_cnt, of the third argument.
  - The DSE information is stored in the RSACPD_DSE type structure in order in which the data appears in one block.
  - If the number of detected DSEs is larger than the value set in the third argument (dse_cnt), the number of DSEs set in dse_cnt is stored in the RSACPD_DSE type structure, but the following DSEs are skipped.
  - If the value of the third argument dse_cnt is more than 16 or less than 1, middleware will end with error (Status code : RSACPD_ERR_DSECNT).

(2) If this function is not executed, DSEs (data_stream_element) cannot be acquired, but decoding can be continued.

(3) If DSEs are not contained in the block after block (raw_data_block) from which DSEs (data_stream_element) have been acquired, the RSACPD_DSE type structure member, present, is set to 0, but other members are not cleared.

(4) In case that DSE does not follow DVB-T standard for downmix or this function is not executed, the decoder will ignore DSE information.

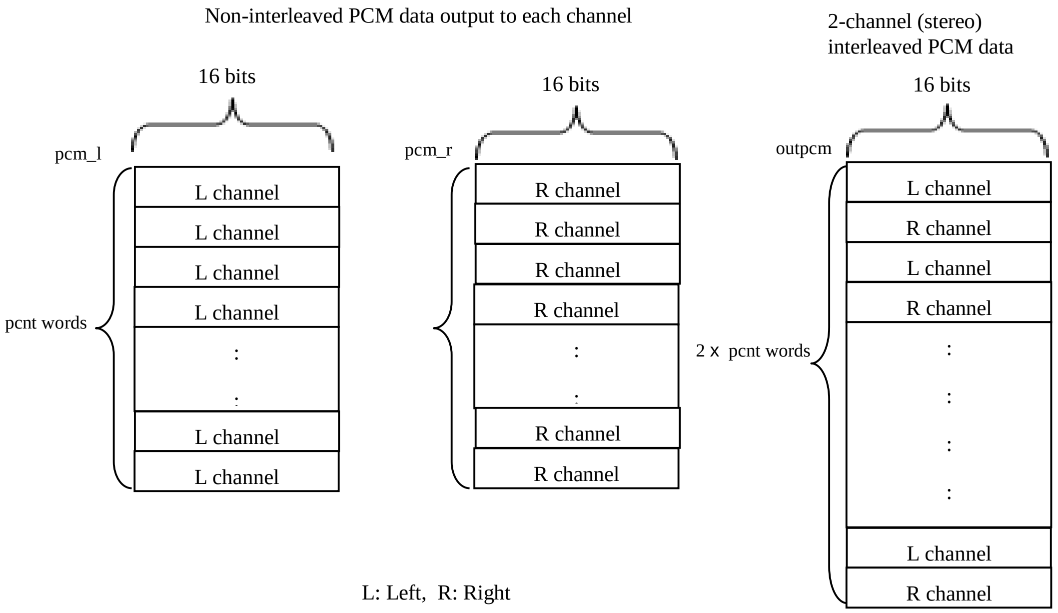
(5) If there are one more DSEs exists in a raw_data_block, the first detected DSE will be referred.

3.2.14 RSACPD_InterleavePCM

Synopsis	int RSACPD_InterleavePCM (short *pcm_l, short *pcm_r, int pcnt, short *outpcm)
Function	Convert the non-interleaved PCM data output to each channel to 2-channel (stereo) interleaved PCM data.

Argument	I/O	Description
short *pcm_l	I	Pointer to PCM data storing buffer for L channel
short *pcm_r	I	Pointer to PCM data storing buffer for R channel
int pcnt	I	Number of words in PCM data per channel (setting value: 1024/2048 or 960/1920)
short *outpcm	O	Pointer to buffer to which 2-channel (stereo) interleaved PCM data will be input.

Return value	Macro name	Description
0	RSACPD_RTN_GOOD	Successfully completed.
-1	RSACPD_RTN_ERROR	Abnormally ends.

Description	<Execution of this function> To use the 2-channel (stereo) interleaved PCM data conversion function, execute this function after the RSACPD_Decode() function terminates normally. <Details of function> This function converts the non-interleaved PCM data output to each channel to a 2-channel (stereo) interleaved PCM data. For the number of words in the PCM data, pcnt, specify the same value as the value in pcnt of RSACPD_Decode(). For the fourth argument outpcm, specify the pointer to a continuous 2 x pcnt words sized buffer secured by an application program.  Specify a continuous pcnt (the third argument) words sized all-zero buffer in pcm_r (the second argument) and monaural data can be converted to stereo data (all R channels are 0). In the same manner, specify the same pointer as pcm_l in pcm_r, and monaural data can be converted to stereo data (the data of L channel are copied onto R channel).

3.2.15 RSACPD_MatrixMixdown

Synopsis	int RSACPD_MatrixMixdown(RSACPD_AAC *aac, int* sel_std, int* mixdown_mode, RSACPD_OUT_INFO* outInfo, int scale)
Function	Downmix a multi channel stream to 2-channel stereo or monaural.

Argument	I/O	Description
RSACPD_AAC *aac	I	Pointer to the RSACPD_AAC type structure
int* sel_std	I	Applicable standard 0: Conforming to ISO/IEC 13818-7 or ISO/IEC 14496-3 1: Conforming to ARIB STD-B21 Rev5.2 2: Conforming to ARIB STD-B21 Rev5.3 regardless of overflow 3: Conforming to ARIB STD-B21 Rev5.3 regarding of overflow
	O	Executed standard 0: Conforming to ISO/IEC 13818-7 or ISO/IEC 14496-3 1: Conforming to ARIB STD-B21 Rev5.2 2: Conforming to ARIB STD-B21 Rev5.3 regardless of overflow 3: Conforming to ARIB STD-B21 Rev5.3 regarding of overflow
int* mixdown_mode	I	Downmix mode setting 0: Downmix to stereo (not to downmix for external pseudo-surround processor) 1: Downmix to stereo (to downmix for external pseudo-surround processor) 2: Downmix to monaural 3: Downmix to stereo (based on DVB-T STD) with volume normalization 4: Downmix to stereo (based on DVB-T STD) without volume normalization and regardless of overflow 5: Downmix to stereo (based on DVB-T STD) without volume normalization and regarding of overflow
	O	Executed downmix mode 0: Downmix to stereo (downmix for external pseudo-surround processor was not executed) 1: Downmix to stereo (downmix for external pseudo-surround processor was executed) 2: Downmix to monaural 3: Downmix to stereo (based on DVB-T STD) with volume normalization 4: Downmix to stereo (based on DVB-T STD) without volume normalization and regardless of overflow 5: Downmix to stereo (based on DVB-T STD) without volume normalization and regarding of overflow
RSACPD_OUT_INFO* outInfo	I/O	Pointer to the RSACPD_OUT_INFO type structure (After the completion of decoding, the number of output channels, audio mode and initial address of the output buffer of each channel PCM data will be set.)
int scale	I	Scale value to prevent overflow due to downmix (setting value: 0 to 2)

Return value	Macro name	Description
0	RSACPD_RTN_GOOD	Successfully completed.
1	RSACPD_RTN_CHECK	Warning
-1	RSACPD_RTN_ERROR	Abnormally ends.

Description

<Execution of this function>
To use the downmix function, execute this function as a post-process after the RSACPD_Decode() function terminates normally.

<Details of function>
This function downmixes a multi-channel stream in the channel mode 3 ch (3/0, 2/1), 4 ch (3/1, 2/2), 5 ch (3/2), and 5.1 ch (3/2+LFE) to 2-channel stereo or monaural. If this function is invoked in another channel mode, the function will end abnormally.
When downmix processing to stereo ends normally, the mixed-down PCM data will be output to pcm_lf and pcm_rf of the RSACPD_OUT_INFO type structure. When downmix processing to monaural ends normally, the mixed-down PCM data will be output from the address designated by pcm_cf.
The below table shows the downmix formulas for each standard.
The channels left, right, center, left surround rear, right surround rear are denoted by L, R, C, LS, RS
The output downmix channels are denoted by L', R', M.
A and B are coefficients based on each standard.

Formula		Standard
(1)	L' = 1 / (1 + 1/sqrt(2) + A) * (L + 1/sqrt(2)*C + A*LS) R' = 1 / (1 + 1/sqrt(2) + A) * (R + 1/sqrt(2)*C + A*RS)	ISO/IEC 14496-3, ISO/IEC 13818-7
(2)	L' = 1 / (1 + 1/sqrt(2) + 2*A) * [L + 1/sqrt(2)*C – A*(LS + RS)] R' = 1 / (1 + 1/sqrt(2) + 2*A) * [R + 1/sqrt(2)*C + A*(LS + RS)]	
(3)	M = 1 / (3 + 2*A) * [L + C + R + A*(LS + RS)]	
(4)	L' = 1/sqrt(2) * (L + 1/sqrt(2)*C + A*LS) R' = 1/sqrt(2) * (R + 1/sqrt(2)*C + A*RS)	ARIB B21 Rev5.2
(5)	L' = 1/sqrt(2) * [L + 1/sqrt(2)*C – A*(LS + RS)] R' = 1/sqrt(2) * [R + 1/sqrt(2)*C + A*(LS + RS)]	
(6)	L' = L + 1/sqrt(2)*C + A*LS R' = R + 1/sqrt(2)*C + A*RS	ARIB B21 Rev5.3
(7)	L' = L + 1/sqrt(2)*C – A*(LS + RS) R' = R + 1/sqrt(2)*C + A*(LS + RS)	
(8)	L' = 1 / (1 + B + A) * (L + B*C + A*LS) R' = 1 / (1 + B + A) * (R + B*C + A*RS)	DVB-T with volume normalization
(9)	L' = L + B*C + A*LS R' = R + B*C + A*RS	DVB-T without volume normalization

(1) Applicable standard:

- If the second argument **sel_std is set to 0**, downmix processing conforming to ISO/IEC 13818-7 or ISO/IEC 14496-3 will be executed.
- If **sel_std is set to 1**, ARIB STD-B21 Rev5.2 is specified, the sound volume generated from multi-channel stream are decoded to make the sound volume generated through downmix as identical as possible. In this case, the downmix computation may result in clipping due to an overflow. Such overflow can be avoided by specifying the value of the fifth argument **scale** to 1 or 2 to acquire a ½ or ¼ input PCM data scale before performing the downmix.
- If **sel_std is set to 2 or 3**, ARIB STD-B21 Rev5.3 regardless of overflow or regarding of overflow is specified, the volume normalization coefficient is not multiplied to the downmix PCM.
    - When regardless of overflow is specified (**sel_std = 2**) and overflow occurs, the decoder will output overflowed PCM and end with RSACPD_RTN_CHECK (Status code: RSACPD_WARN_MIXDOWN_OVF).
    - When regarding of overflow is specified (**sel_std = 3**) and overflow occurs, this function will change this standard to ARIB B21 Rev5.2 (**sel_std = 1**) and output the downmix PCM multiplied by the volume normalization coefficient. From the sample where overflow occurs, output PCM is downmixed with new formula of new standard.
- Whether or not the selecting standard has been changed can be checked according to the value of sel_std after this function is executed.

(2) Downmix mode setting:

- If the third argument **mixdown_mode is set to 2** (downmix to monaural), monaural downmix processing conforming to ISO/IEC 13818-7 or ISO/IEC 14496-3 will be executed regardless of the value of the second argument sel_std.
- If **mixdown_mode is set to 1**, downmix for external pseudo_surround processor can be executed when PCE has acquired and the RSACPD_PCE type structure members **matrix_mixdown_idx_present and pseudo_surround_enable are 1**. If mixdown_mode is 1 when these requirements are not met, the processing will be executed in the same manner as when mixdown_mode is 0.
- This middleware can support downmix based on DVB-T STD. DVB-T STD uses the ISO/IEC STD downmix matrix with higher resolution and refers the downmix information in DSE to calculate the downmix coefficients, and it supports only downmix from multi-channel to stereo. In order to enable downmix on DVB-T, RSACPD_SetDSE() function must be executed (Section 3.2.13), **mixdown_mode should be set to 3, 4 or 5**; and **0 should be set to the argument sel_std**. Note that, the decoder does not find downmix information in DSE, or "0" is not set to the argument sel_std, it will use the default downmix matrix (**mixdown_mode = 0**) and follow the selected STD in the second argument sel_std.
- There are two formulas for DVB-T STD:
  - With volume normalization (**mixdown_mode = 3**)
  - Without volume normalization (**mixdown_mode = 4 or 5**)
    - If regardless of overflow is specified (**mixdown_mode = 4**) and overflow occurs, the decoder will output overflowed PCM and end with RSACPD_RTN_CHECK (Status code: RSACPD_WARN_MIXDOWN_OVF).
    - If regarding of overflow is specified (**mixdown_mode = 5**) and overflow occurs, this function will change **mixdown_mode to 3**. From the sample where overflow occurs, output PCM is downmixed with new formula of new downmix mode.
- In case of downmix based on DVB-T STD, once the downmix information in DSE is acquired, it will be used for the next frames (even though DSE does not present in these frames) until new DSE is acquired.
- Whether or not the mixdown mode has been changed can be checked according to the value of **mixdown_mode** after this function is executed.

<Notes>

(1) When this function ends normally (or ends with a warning), the PCM data output before downmix processing will be overwritten with the PCM data output after downmix processing. In the same manner, other members of the RSACPD_OUT_INFO type structure will be overwritten after downmix processing.
(2) The PCM data to be output by downmix to stereo is in the non-interleaved PCM format. Therefore, to convert it to its equivalent in the 2-channel (stereo) interleaved PCM format, execute RSACPD_InterleavePCM() function after executing this function.
(3) The modes in which the applied standards are automatically changed for overflow case (mixdown_mode = 5 or sel_std = 3) are Renesas original modes.
(4) In two above modes, because the left channel of downmixed output PCM is calculated first, if overflow occurs in the left channel, the right channel is calculated with new standard, but if overflow occurs in the right channel, the left channel is kept with old standard.
(5) If a raw data block has more than 1 DSE, the decoder will use downmix information in the first one.
(6) In case that the applicable standard is set to ARIB STD-B21 Rev5.3 (sel_std is 3), if overflow still occurs after applying new sel_std ARIB STD-B21 Rev5.2 (sel_std is 1), this function will output overflowed PCM and end with RSACPD_RTN_CHECK (Status code: RSACPD_WARN_MIXDOWN_OVF).

3.2.16 RSACPD_SetSAC

Synopsis	int RSACPD_SetSAC(RSACPD_AAC *aac, RSACPD_SAC *sac)
Function	Set the area for acquiring spatial audio coding side information necessary for MPEG Surround contained in a bit stream.

Argument	I/O	Description
RSACPD_AAC *aac	I/O	Pointer to the RSACPD_AAC type structure
RSACPD_SAC *sac	I	Pointer to the RSACPD_SAC type structure

Return value	Macro name	Description
0	RSACPD_RTN_GOOD	Successfully completed.
-1	RSACPD_RTN_ERROR	Abnormally ends.

Description	<Execution of this function> When it is necessary to acquire spatial audio coding side information necessary for MPEG Surround contained in a bit stream, execute this function before executing RSACPD_Decode() or RSACPD_Skip() for the first time after executing RSACPD_Open(). <Details of function> When spatial audio coding side information is detected during execution of RSACPD_Decode() or RSACPD_Skip(), spatial audio coding side information is output to the RSACPD_SAC type structure, sac. <Notes> (1) This function is not applicable to MPEG Surround. For MPEG Surround, see ISO/IEC 23003-1. (2) The area for the RSACPD_SAC type structure must be reserved by the application program. For the details, see Section 7.8 “RSACPD_SAC type structure”. (3) If this function is not executed, spatial audio coding side information cannot be acquired, but decoding can be continued. (4) If spatial audio coding side information is not contained in the block after the block (raw_data_block) from which spatial audio coding side information has been acquired, the RSACPD_SAC type structure member, present, is set to 0, but other data are not cleared. (5) When more than one piece of spatial audio coding side information is detected in a raw_data_block, the last acquired spatial audio coding side information is output to the RSACPD_SAC type structure

3.2.17 RSACPD_SetDRC

<table>
<tr><td>Synopsis</td><td colspan="3">int RSACPD_SetDRC(
RSACPD_AAC *aac,
RSACPD_DRC *drc,
int hi,
int lo,
int ref_level
)</td></tr>
<tr><td>Function</td><td colspan="3">Set Dynamic Range Control information.</td></tr>
<tr><td colspan="2">Argument</td><td>I/O</td><td>Description</td></tr>
<tr><td colspan="2">RSACPD_AAC *aac</td><td>I/O</td><td>Pointer to the RSACPD_AAC type structure</td></tr>
<tr><td colspan="2">RSACPD_DRC *drc</td><td>O</td><td>Pointer to the RSACPD_DRC type structure</td></tr>
<tr><td colspan="2">int hi</td><td>I</td><td>DRC cut scale
(setting value: 0 to 100)</td></tr>
<tr><td colspan="2">int lo</td><td>I</td><td>DRC boost scale
(setting value: 0 to 100)</td></tr>
<tr><td colspan="2">int ref_level</td><td>I</td><td>DRC output level
(setting value: 0 to 127, -1)</td></tr>
<tr><td>Return value</td><td colspan="2">Macro name</td><td>Description</td></tr>
<tr><td>0</td><td colspan="2">RSACPD_RTN_GOOD</td><td>Successfully completed.</td></tr>
<tr><td>-1</td><td colspan="2">RSACPD_RTN_ERROR</td><td>Abnormally ends.</td></tr>
<tr><td>Description</td><td colspan="3"><Execution of this function>
If user wants to apply DRC, execute this function after RSACPD_Open() without fail to set DRC information for the decoder.

<Details of function>
The DRC can adjust the dynamic range in the signal based on DRC data in the bit stream. The volume of the PCM might be cut or boosted at some or all frames.
It is possible to scale cut and boost coefficient in the bit stream by setting the value between 0 and 100 to the third and fourth arguments. If 100 is set, the decoder will refer the full scaled coefficient in the bit stream as it is and the output will be max cut or minimum boosted. If 0 is set, the decoder will not refer DRC coefficient in the bit stream and do not cut or boost the audio data.
Also if a value between 0 and 127 is set to the fifth argument “ref_level”, the DRC program reference level in the bit stream will be adjusted. If -1 is set to this argument, the decoder will not refer DRC program reference level in the bit stream. Only the cut and boost coefficient will be referred.

<Notes>
(1) If a value other than the setting value (0 to 100) is set to third and fourth arguments, 100 is set by the middleware and continue decoding.
(2) If a value of 128 or more is set to the fifth argument “ref_level”, the value is set to 127 and middleware will continue decoding. If a value of -2 or less is set to “ref_level”, the value is set to -1 and middleware will continue decoding.
(3) Up to three DRC threads can be supported in a raw_data_block. The fourth and following DRC will be ignored.</td></tr>
</table>

4 User-created function

4.1 User-created function

In this middleware, the user must create the following function.

The user-created function is invoked by this middleware to feed the bit stream data in the input buffer to the middleware.

<table>
<tr><td>Synopsis</td><td colspan="3">unsigned int RSACPD_GetData(
    unsigned char *wpt,
    int size
)</td></tr>
<tr><td>Function</td><td colspan="3">Replenishes the input buffer with bit stream data.</td></tr>
<tr><td colspan="2">Argument</td><td>I/O</td><td>Description</td></tr>
<tr><td colspan="2">unsigned char *wpt</td><td>I</td><td>Pointer to the input buffer</td></tr>
<tr><td colspan="2">int size</td><td>I</td><td>Number of the maximum data that can be input to the buffer (in bytes)</td></tr>
<tr><td>Return value</td><td colspan="3">Number of bytes input to the input buffer</td></tr>
<tr><td>Description</td><td colspan="3"><Execution of this function>
This function is invoked by this middleware when bit stream data is emptied out of the input buffer.

<Details of function>
Input bit stream data to addresses after the address specified by the argument wpt. Argument size used to specify number of byte need to get. This function will return the size of data to be input. If there is no data to be input, set 0 in the return value.
When the return value of this function is 0, this middleware considers that the input bit stream has ended and terminates abnormally with the status RSACPD_ERR_DATA_EMPTY. With the aid of this function, the user can forcibly stop the decode processing.
To the first argument of this function, the second argument (input buffer initial address) of RSACPD_Open() is delivered.
The third argument (input buffer size) of RSACPD_Open() is delivered to the second argument of this function.

<Note>
The operation cannot be guaranteed if the return value of this function is bigger than the second argument (size).</td></tr>
</table>

Note: The name of user-created function is determined by user. This document calls it RSACPD_GetData.

4.2 Outline of operation

The outline of the operation of the user-created function is shown in Figure 4.1.

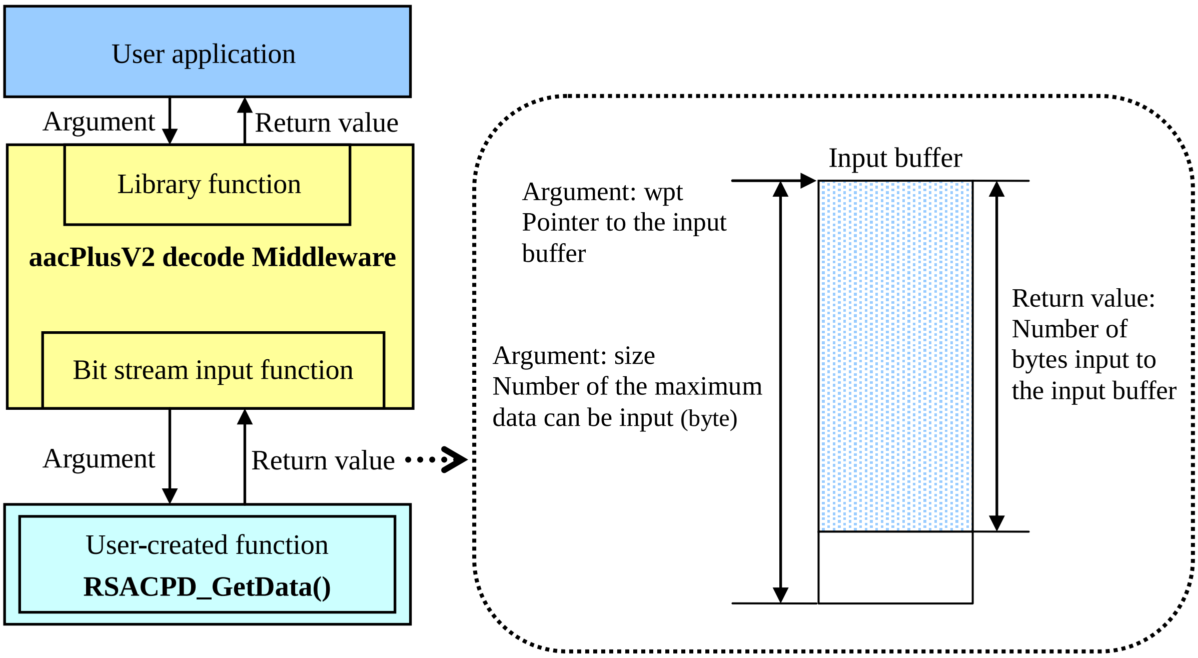


Figure 4.1 Outline of operation

5 Channel Configuration

5.1 Definition of channel count

Table 5-1 and 5-2 show the PCM data output destination (members of the RSACPD_OUT_INFO type structure) corresponding to every audio mode supported by this middleware. After decoding, use the information set to this structure to retrieve the PCM data corresponding to the channel structure.

Table 5-1 Channel definition and PCM data output destination (1)

Audio mode	ADTS header configuration	pcm_cf	pcm_lf	pcm_rf	pcm_ls	pcm_rs	pcm_lfe	Element appearance order				channel Mode
								1	2	3	4	
Monaural	(1)	M						SCE				0
Stereo	(2)		L	R				CPE				1
3/0	(3)	C	L	R				SCE	CPE			4
3/1	(4)	C	L	R	MS			SCE	CPE	SCE		5
3/2	(5)	C	L	R	LS	RS		SCE	CPE	CPE		6
3/2 & LFE	(6)	C	L	R	LS	RS	LFE	SCE	CPE	CPE	LFE	7
Dual Mono	(0)		M	M				SCE	SCE			2
2/1	(0)		L	R	MS			CPE	SCE			8
2/2	(0)		L	R	LS	RS		CPE	CPE			9

* "ARIB STD-B32 2.1" standard specifications are referred to define channels.

(Note 1) "/" in audio mode indicates the number of channels of the front/rear speakers.

Example: 3/1 = front 3ch + rear 1ch

C: Center L: Left R: Right M: Monaural

MS: Monaural Surround Rear (Rear Surround) LS: Left Surround Rear

RS: Right Surround Rear LFE: Low Frequency Effects

(Note 2) ChannelMode is a member of RSACPD_OUT_INFO type structure.

(Note 3) The members of RSACPD_OUT_INFO type structure which don't contain PCM data will be assigned to NULL.

In an audio mode other than those defined in Table 5-1, RSACPD_Decode() function returns a warning (Status code: RSACPD_WARN_UNSUPPORTED_CH_CFG) and normally outputs the PCM data, if the total number of channels is supported. In this case, the output destination of PCM data is described in Table 5-2.

When PCE is acquired with executing the RSACPD_SetPCEArea() function, PCM data is output according to the PCE information. See "3.2.2 RSACPD_SetPCEArea"

When the channel information (PCE information, channel configuration in ADTS header and elements include in a raw_data_block) is changed during decode processing, this middleware follows the new information.

If ADTS channel configuration and PCE appears in the same frame, the decoder will follow PCE.

PCE information is ignored if the channel count information contained in PCE differs from the actual number of channels configured by every element included in the actual raw_data_block; if ADTS channel configuration differs from the actual number of channels, a warning message is output (Status code: RSACPD_WARN_ILLEGAL_CHAN_CONFIG).

Table 5-2 Channel definition and PCM data output destination (2)

Sound mode	Element appearance order / Channel mapping					channelMode
3/2	SCE	SCE	SCE	SCE	SCE	-1
	pcm_cf	pcm_lf	pcm_rf	pcm_ls	pcm_rs	
3/2	SCE	SCE	SCE	CPE		-1
	pcm_cf	pcm_lf	pcm_rf	pcm_ls	pcm_rs	
3/2	SCE	SCE	CPE		SCE	-1
	pcm_cf	pcm_lf	pcm_ls	pcm_rs	pcm_rf	
3/2	SCE	CPE		SCE	SCE	-1
	pcm_cf	pcm_lf	pcm_rf	pcm_ls	pcm_rs	
3/2	CPE		SCE	SCE	SCE	-1
	pcm_lf	pcm_rf	pcm_cf	pcm_ls	pcm_rs	
3/2	CPE		SCE	CPE		-1
	pcm_lf	pcm_rf	pcm_cf	pcm_ls	pcm_rs	
3/2	CPE		CPE		SCE	-1
	pcm_lf	pcm_rf	pcm_ls	pcm_rs	pcm_cf	
3/1	SCE	SCE	SCE	SCE	-	-1
	pcm_lf	pcm_rf	pcm_ls	pcm_rs	-	
3/1	SCE	SCE	CPE		-	-1
	pcm_lf	pcm_rf	pcm_ls	pcm_rs	-	
3/1	CPE		SCE	SCE	-	-1
	pcm_lf	pcm_rf	pcm_ls	pcm_rs		
3/0	SCE	SCE	SCE	-	-	-1
	pcm_cf	pcm_lf	pcm_rf	-	-	

(Note 1) PCM of LFE is always output to pcm_lfe (omitted from the table above).

(Note 2) CPE must be output as a pair of pcm_lf and pcm_rf, or pcm_ls and pcm_rs.

(Note 3) For three SCEs and onc CPE (SCE SCE SCE CPE), the CPE is output as a pair of pcm_ls and pcm_rs.

6 Decoding Flowchart

The input bit stream is in RawDataStream, ADTS, ADIF or LOAS format.

The decoding flowchart in each data format is shown below.

6.1 Decoding in RawDataStream-format

Figure 6.1 shows the procedure of decoding bit stream in RawDataStream-format. After the middleware is initialized, decoding is repeated by frame.

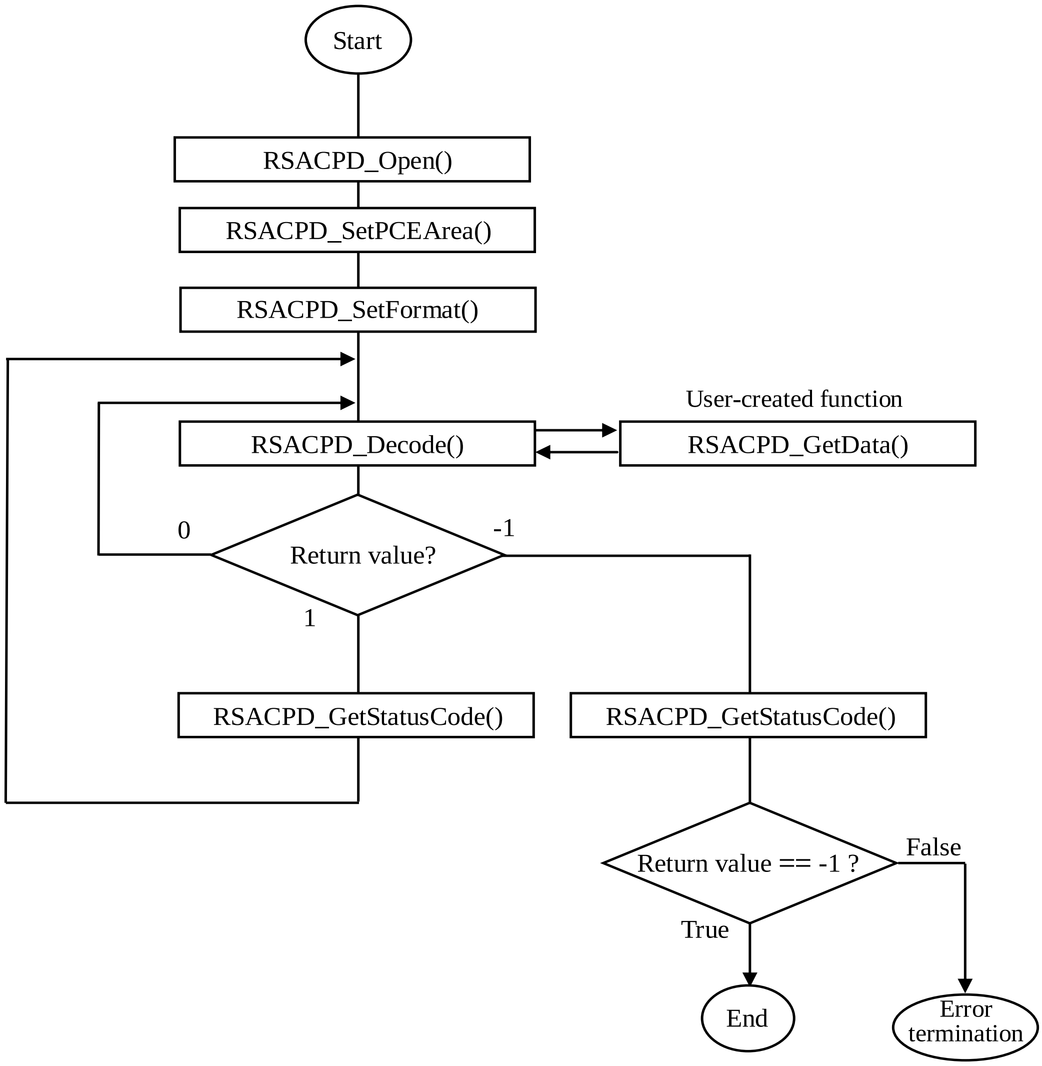


Figure 6.1 Flowchart of decoding in RawDataStream-format (example)

6.2 Decoding in ADTS-format

Figure 6.2 is the flowchart showing the procedure for decoding a bit stream in the ADTS-format. In the head of each ADTS frame in the ADTS-format bit stream, a header (ADTS header) beginning with a syncword. Each ADTS frame is configured with some raw_data_block. After the middleware is initialized, the ADTS header information is acquired, and decoding is repeated by raw_data_block.

When some raw_data_block(s) in the ADTS frame are not decoded, "1" is set to the fourth bit of the decode status to be acquired with the RSACPD_DecodeStatus().

The number of raw_data_block(s) contained in the ADTS frame can be acquired with "number_of_raw_data_blocks_in_frame" of the ADTS header information.

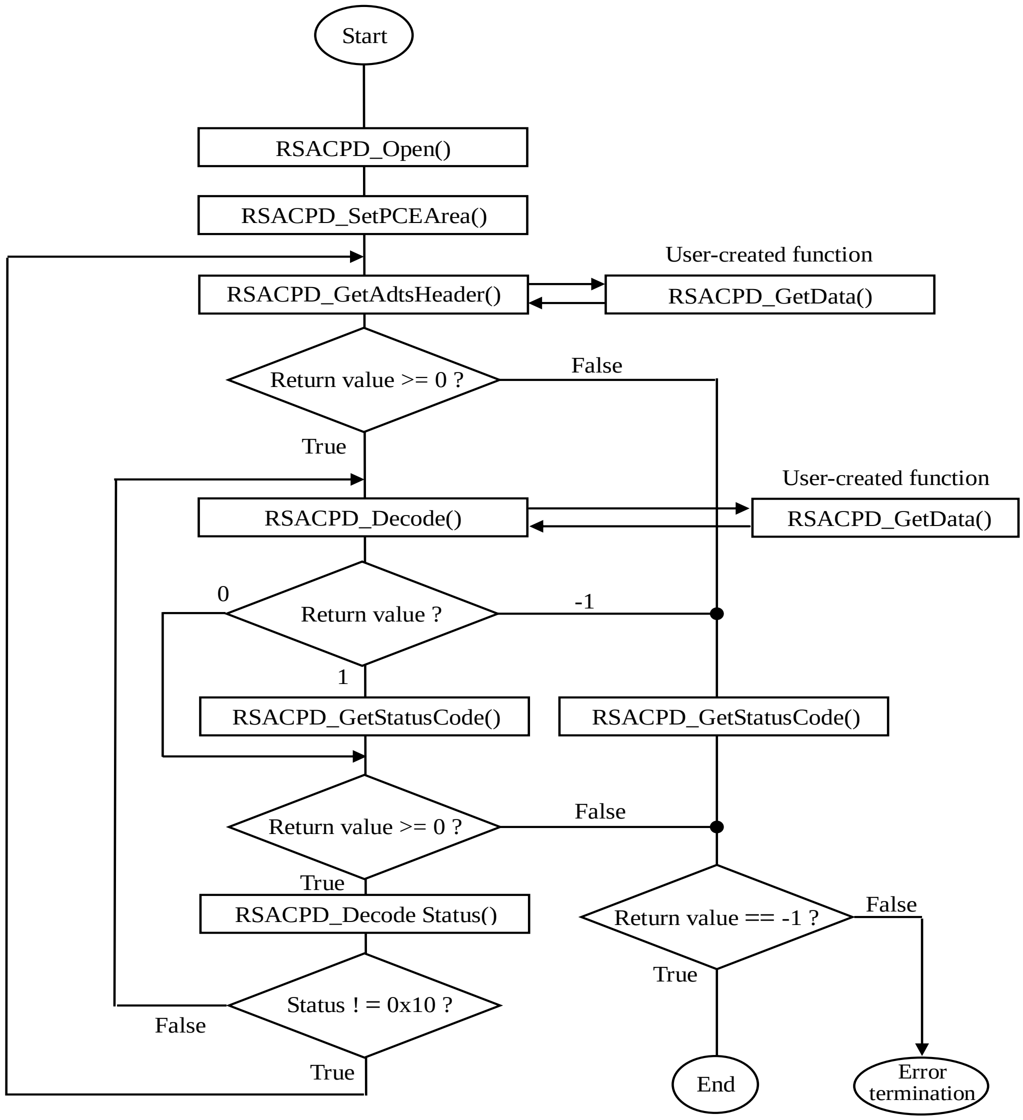


Figure 6.2 Flowchart of decoding in ADTS-format (example)

6.3 Decoding in ADIF-format

Figure 6.3 is the flowchart showing the procedure for decoding a bit stream in the ADIF-format. The bit stream in the ADIF-format has the header information at its head. After the middleware is initialized, the ADIF header information is acquired, and decoding is repeated by frame.

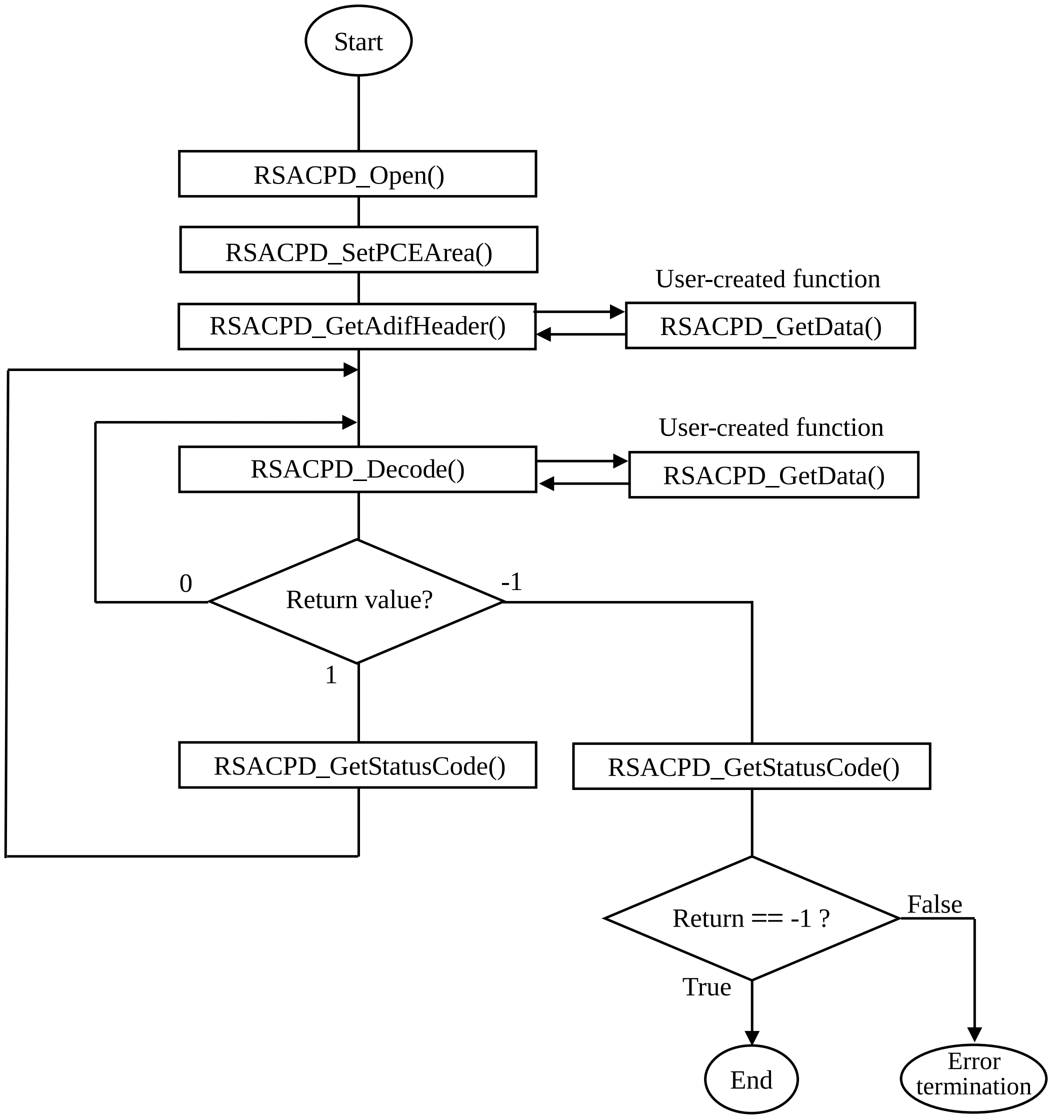


Figure 6.3 Flowchart of decoding in ADIF-format (example)

6.4 Decoding in LOAS-format

Figure 6.4 is the flowchart showing the procedure for decoding a bit stream in the LOAS-format. A header (LOAS header) which begins with a syncword is located at the beginning of each LOAS frame in LOAS format bit streams. After the decoder has been initialized, the LOAS header information is acquired, and decoding is executing.

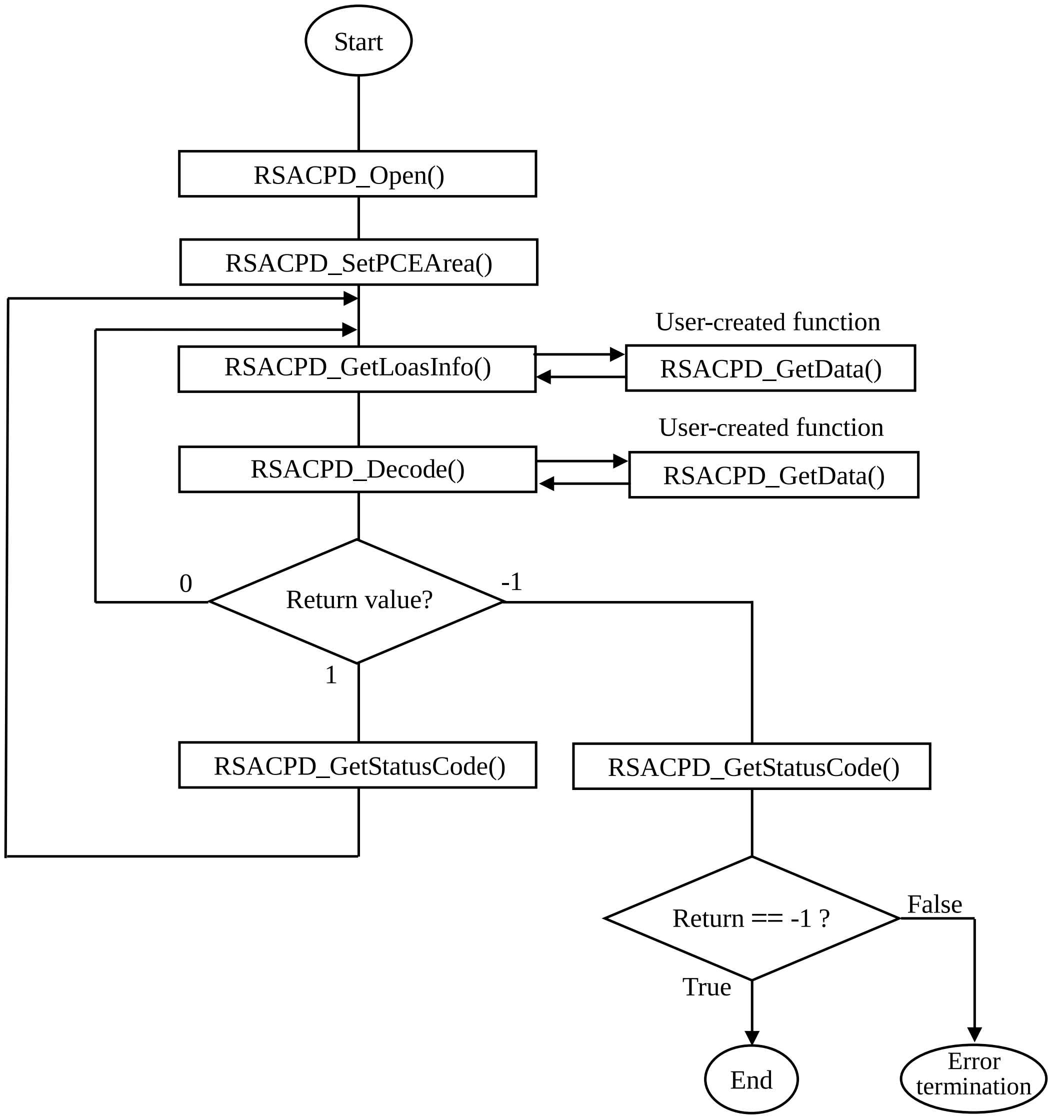


Figure 6.4 Flowchart of decoding in LOAS-format (example)

7 Structure

Table below shows the structures for which areas must be reserved by the application program when this middleware is incorporated. The type of each structure is defined in the library header file (RSACPD_ADL.h). The contents of these structures are fixed after the middleware ends normally.

Because these types of structure are the work area to be used by the middleware, do not change the contents of the structure by the application program.

Table 7-1 List of structure

No.	Structure definition	Size	Remarks
1	RSACPD_AAC	Approx. 225 Kbytes	Required
2	RSACPD_PCE	413 bytes	Required
3	RSACPD_AdifHeader	28 bytes	Required for decoding in the ADIF-format.
4	RSACPD_AdtsHeader	18 bytes	Required for decoding in the ADTS-format.
5	RSACPD_LoasInfo	36 bytes	Required for decoding in the LOAS-format.
6	RSACPD_OUT_INFO	52 bytes	Required
7	RSACPD_DSE	24 bytes	Arbitrary
8	RSACPD_SAC	1096 bytes	Arbitrary
9	RSACPD_DRC	2108 bytes	Arbitrary

7.1 RSACPD_AAC type structure

The RSACPD_AAC type structure is the work area used by this middleware. When using this middleware, secure the area with the application program. It's not necessary to refer to this area because it only contains the internal variables and working buffers of the middleware, so this structure is described in the Appendix.

Make sure not to change the value of this area with the application program.

7.2 RSACPD_PCE type structure

The RSACPD_PCE type structure is the area which the application program acquires the PCE element in the raw_data_block and the PCE information of the ADIF header. When using this middleware, secure the area with the application program. The RSACPD_PCE type structure is arrayed as necessary so that multiple PCE information can be used. The data structure of the RSACPD_PCE type structure is shown below.

Table 7-2 RSACPD_PCE type structure information

Member name	Description
unsigned char element_instance_tag	Element instance tag
unsigned char profile	00: Main 01: LC (Required) 10: SSR 11: Reserved (for MPEG2), LTP (for MPEG4)
unsigned char sampling_frequency_index	See Table 3-3.
unsigned char num_front_channel_elements	Number of front audio structure elements
unsigned char num_side_channel_elements	Number of side audio structure elements
unsigned char num_back_channel_elements	Number of rear audio structure elements
unsigned char num_lfe_channel_elements	Number of LFE audio structure elements
unsigned char num_assoc_data_elements	Number of related data elements
unsigned char num_valid_cc_elements	Number of CCE elements to be added to audio data
unsigned char mono_mixdown_present	Monaural downmix flag
unsigned char mono_mixdown_element_number	Number of monaural downmix elements
unsigned char stereo_mixdown_present	Stereo downmix flag
unsigned char stereo_mixdown_element_number	Number of stereo downmix flag of CPE elements
unsigned char matrix_mixdown_idx_present	Matrix mixdown information flag
unsigned char matrix_mixdown_idx	Surround downmix coefficient index
unsigned char pseudo_surround_enable	Pseudo sound downmix flag
unsigned char front_element_is_cpe[16]	Front specification flag of SCE and CPE
unsigned char front_element_tag_select[16]	SCE/CPE instant tag to be processed as front element
unsigned char side_element_is_cpe[16]	Side element specification flag of SCE and CPE
unsigned char side_element_tag_select[16]	SCE/CPE instant tag to be processed as side element
unsigned char back_element_is_cpe[16]	Rear element specification flag of SCE and CPE
unsigned char back_element_tag_select[16]	SCE/CPE instant tag to be processed as rear element
unsigned char lfe_element_tag_select[4]	LFE instance tag
unsigned char assoc_data_element_tag_select[8]	DSE instance tag
unsigned char cc_element_is_ind_sw[16]	Independent CCE flag
unsigned char valid_cc_element_tag_select[16]	CCE instance tag
unsigned char comment_field_bytes	Byte count of the following comment field
unsigned char comment_field_data[256]	Comment field data

7.3 RSACPD_AdifHeader type structure

The RSACPD_AdifHeader type structure is the area to acquire the ADIF header information from the bit stream in the ADIF-format. When using this middleware, secure the area with the application program. The data structure is shown below.

Table 7-3 RSACPD_AdifHeader type structure information

Member name	Description
unsigned char adif_id[4]	“ADIF” (0x41, 0x44, 0x49, 0x46)
unsigned char copyright_id_present	0: Appends the copyright ID. 1: Does not append the copyright ID.
unsigned char copyright_id[9]	Copyright ID
unsigned char original_copy	0: Copy 1: Original
unsigned char home	Reserved.
unsigned char bitstream_type	0: Fixed bit rate 1: Variable bit rate
unsigned long bitrate	Bit rate, if fixed bit rate Maximum bit rate, if Variable bit rate
unsigned char num_program_config_element	Number of program_config_elements is equal to num_program_config_element + 1. (The value 0 is indicating 1 program_config_element)

7.4 RSACPD_AdtsHeader type structure

The RSACPD_AdtsHeader type structure is the area to acquire the ADTS header information by ADTS frame from the bit stream in the ADTS-format. When using this middleware, secure the area with the application program. The data structure is shown below.

Table 7-4 RSACPD_AdtsHeader type structure information

Member name	Description
unsigned char ID	0:MPEG-4 1:MPEG-2
unsigned char layer	00: Reserved, 01: Layer 3, 10: Layer 2, 11: Layer 1 (00 is required for AAC/aacPlus coded bit stream)
unsigned char protection_absent	0: Adds correction of error detection 1: Does not append the copyright ID
unsigned char profile	00: Main 01: LC (Required) 10: SSR 11: Reserved (for MPEG2), LTP (for MPEG4)
unsigned char sampling_frequency_index	See Table 3-3.
unsigned char private_bit	Private bit
unsigned char channel_configuration	Channel configuration
unsigned char original_copy	0: Copy 1: Original
unsigned char home	Reserved.
unsigned char copyright_identification_bit	0: Appends the copyright 1: Does not append the copyright ID
unsigned char copyright_identification_start	0: Copyright information does not exist 1: Copyright information exists
unsigned short frame_length	ADTS frame length (Number of bytes of an ADTS frame)
unsigned short adts_buffer_fullness	State of the bit reservoir
unsigned char number_of_rawdata_blocks_in_frame	Number of blocks (raw_data_block) Note: One block for the value "0" (Number of blocks (block count+1) exists.)

7.5 RSACPD_LoasInfo type structure

The RSACPD_LoasInfo type is the area to acquire the LOAS header information by LOAS frame from the bit stream in the LOAS-format. When using this middleware, secure the area with the application program. The data structure is shown below.

Table 7-5 RSACPD_LoasInfo type structure information

Member name	Description
unsigned short audioMuxLengthBytes	A data element indicating the byte length of the subsequent AudioMuxElement() with byte alignment.
unsigned char useSameStreamMux	A flag indicating whether the multiplex configuration in the previous frame is applied to the current frame.
unsigned char audioMuxVersion	A data element to signal the used multiplex syntax. 0: Default. 1: Support the transmission of a taraBufferFullness and the transmission of the lengths of individual AudioSpecificConfig() data function.
unsigned char audioMuxVersionA	A data element to signal the used syntax version 0: Default. 1: Reserved for future extensions (Not supported).
unsigned char allStreamSameTimeFraming	A data element indicating whether all payloads, which are multiplexed in PayloadMux(), share a common time base. 0: Not share (Not supported). 1: Share.
unsigned char numSubFrames	A data element indicating how many PayloadMux() frames are multiplexed (numSubFrames+1). The minimum value is 0 indicating 1 subframe.
unsigned char numSubFramesIndex	Current PayloadMux() frame index.
unsigned char numProgram	A data element indicating how many programs are multiplexed (numProgram+1). The middleware only supports the value 0 (indicate 1 program).
unsigned char numLayer	A data element indicating how many scalable layers are multiplexed (numLayer+1). The middleware only supports the value 0 (indicate 1 layer).
unsigned char otherDataPresent	A flag indicating the presence of the other data than audio payloads. 0: The other data than audio payload otherData is not multiplexed. 1: The other data than audio payload otherData is multiplexed.
unsigned char audioObjectType	A data element indicating the audio Object type of the bit stream: 2: AAC Low Complexity (LC) profile 5: SBR profile 29: PS profile others: Not supported
unsigned long otherDataLenBits	A data element indicating the length in bits of other data
unsigned char crcCheckPresent	A data element indicating the presence of CRC check bits for the StreamMuxConfig() data function. 0: CRC check bits are not present. 1: CRC check bits are present.
unsigned char crcCheckSum	CRC error detection data. This CRC uses the generation polynomial CRC8 and covers the entire StreamMuxConfig() upto but excluding the crcCheckPresent bit. Note: This middleware does not detect CRC error for LOAS format

unsigned char frameLengthType	A data element indicating the frame length type of the payload. 0: Payload with variable frame length. The payload length in bytes is directly specified with 8-bit codes in PayloadLengthInfo(). 1: Payload with fixed frame length. The payload length in bits is specified with frameLength in StreamMuxConfig() Others: Not supported
unsigned char frameLengthFlag	A data element indicating the length of the frame, number of spectral, respective. 0: A 1024/128 lines IMDCT is used and frame length is set to 1024 sample. 1: A 960/120 lines IMDCT is used and frame length is set to 960 sample.
unsigned short frameLength	A data element indicating the frame length of the payload with frameLengthType of 1. The payload length in bits is specified as 8 * (frameLength+20).
unsigned char channelConfiguraton	A four bit indicating the audio output channel configuration.
unsigned char samplingFrequencyIndex	A four bit indicating the sampling rate used. See Table 3-3 for details
unsigned long samplingFrequency	The sampling frequency used for this audio object in case of samplingFrequencyIndex is 0xF
unsigned long extensionSamplingFrequency	The output sampling frequency of the extension tool corresponding to the extensionAudioObjectType in case of extensionSamplingFrequencyIndex is 0xF.
unsigned char extensionFlag	A data element indicating the presence of extension data
unsigned char extensionSamplingFrequencyIndex	A four bit indicating the output sampling frequency of the extension tool corresponding to the extensionAudioObjectType. See Table 3-3 for more details

For “Not supported” cases, the decoder will return error with status code RSACPD_ERR_LOAS_INFO. For more detail, refer to Section 8.2.5.”

7.6 RSACPD_OUT_INFO type structure

The RSACPD_OUT_INFO type is the structure for this middleware to store channel mode, number of channels of output PCM data and the output PCM data.

Before executing this middleware, secure the area with the application program.

The data structure of the RSACPD_OUT_INFO type structure is shown below.

Table 7-6 RSACPD_OUT_INFO type structure

Member name	Description
int channelMode	0: Monaural 1: Stereo 2: Dual monaural 3: Parametric stereo 4:3/0 5:3/1 6:3/2 7:3/2 + LFE (5.1) 8:2/1 9:2/2 -1: Others (decoding abnormal ends or audio mode other than those defined in Table 5-1)
int ChannelNumber	Number of all channels
int nfch	Number of front channels
int nsch	Number of side channels (Must be set by “0”)
int nbch	Number of rear channels
int nlch	Number of LFE channel
short *pcm_buf	Pointer to PCM data area allocated by application
short *pcm_cf	When nfch = 1 or 3 - Pointer to center channel output PCM data
short *pcm_lf	When nfch = 2 or 3 Pointer to either of the following output PCM data - Left channel - Dual monaural first channel (when channel mode = 2)
short *pcm_rf	When nfch = 2 or 3 Pointer to either of the following output PCM data - Right channel - Dual monaural second channel (when channel mode = 2)
short *pcm_ls	When nbch = 1 or 2 Pointer to either of the following output PCM data - Left surround rear channel - Monaural surround rear channel (when channel mode = 5 or 8)
short *pcm_rs	When nbch = 2 - Pointer to right surround rear channel output PCM data
short *pcm_lfe	When nlch = 1 - Pointer to LFE channel output PCM data

7.7 RSACPD_DSE type structure

The RSACPD_DSE type structure is an area for storing data_stream_element information embedded in the DSE element. If it is necessary to acquire data_stream_element information, reserve an area with the application program when incorporating this middleware. If necessary, more than one DSE can be acquired by arraying the RSACPD_DSE type structure. If the area is not reserved, DSE will be skipped.

For more details, see Section 3.2.13 "RSACPD_SetDSE"

The data structure of the RSACPD_DSE type structure is shown below.

Table 7-7 RSACPD_DSE type structure information

Member name	Description
unsigned int count	Number of bytes in DSE
unsigned char present	1 : DSE(data_stream_element) is acquired. 0 : DSE(data_stream_element) is not acquired.
unsigned char element_instance_tag	Element instance tag
unsigned char ancillary_data_sync	Data field indicating the availability of the extended ancillary data 0xBC : The ancillary data is present others : The ancillary data is not present
unsigned char mpeg_audio_type	MPEG audio type
unsigned char dolby_surround_mode	Dolby surround mode
unsigned char downmixing_levels_MPEG4_status	Data field indicating the presence of downmixing_levels_MPEG4 1 : The MPEG4 downmixing level exists 0 : The MPEG4 downmixing level does not exist
unsigned char center_mix_level_on	1 : The center_mix_level_value is used 0 : The center_mix_level_value is not used
unsigned char center_mix_level_value	Matrix mixdown level
unsigned char surround_mix_level_on	1 : The surround_mix_level_value is used 0 : The surround_mix_level_value is not used
unsigned char surround_mix_level_value	Matrix mixdown level
unsigned char audio_coding_mode_and_compression_status	1 : The audio_coding_mode is present 0 : The audio_coding_mode is not present
unsigned char compression_on	1 : The compression_value is used 0 : The compression_value is not used
unsigned char compression_value	The heavy compression factor used for monophonic downmix reproduction
unsigned char coarse_grain_timecode_status	1 : The coarse_grain_timecode is present 0 : The coarse_grain_timecode is not present
unsigned short coarse_grain_timecode	Coarse grain timecode value
unsigned short fine_grain_timecode	1 : The fine_grain_timecode is present 0 : The fine_grain_timecode is not present
unsigned char fine_grain_timecode_status	Fine grain timecode value

7.8 RSACPD_SAC type structure

The RSACPD_SAC type structure is an area for storing spatial audio coding side information necessary for MPEG Surround embedded in extension_type of EXT_SAC_DATA element. If it is necessary to acquire the data, reserve an area with the application program when incorporating this middleware.

Table 7-8 RSACPD_SAC type structure information

Member name	Description
int present	1: SAC (special) information is acquired. 0: SAC (special) information is not acquired.
int ancType	Indicates type of ncillary data: 0: MPEG Surround frame 1: MPEG Surround header and MPEG Surround frame Othres : Reserved
int ancStart	Indicates if data segment begins a data block.
int ancStop	Indicates if data segment ends a data block.
int ancDataSegmentByte[269]	Spatial audio data. Note that the low-order 8 bits are significant.
int count	Number of bytes acquired in ancDataSegmentByte [269] (= number of significant words)

7.9 RSACPD_DRC type structure

The RSACPD_DRC type structure is an area for storing the DRC information. If it is necessary acquire the data, reserved an area with the application program when incorporating this middleware.

Table 7-9 RSACPD_DRC type structure information

Member name	Description
int enable	0 : DRC processing is not executed 1 : DRC processing is executed
int hi	DRC cut scale (setting value: 0 to 100)
int lo	DRC boost scale (setting value: 0 to 100)
int digital_norm	0 : Program level normalization in digital domain is not executed 1 : Program level normalization in digital domain is executed
int target_ref_level	The target reference level
int prog_ref_level	The program reference level
int thread	Number of DRC threads
RSACPD_DRC_Bitstream drc_bit[3]	DRC bitstreams buffer
RSACPD_DRC_Info drc_info[6]	DRC information

Note that structure information of RSACPD_DRC_Bitstream and RSACPD_DRC_Info are described in Appendix

8 List of status codes

The API functions of this middleware end normally, return a warning or return an abnormal end error code. When any function gives a warning or ends abnormally, the detailed status can be checked with RSACPD_ GetStatusCode(). Table 8-1 is a list of status codes.

For the processing of the application program including error recovery of each API function, see Section 8.2 “API functions and status codes”.

When more than one status is detected, the status code is set as stated below.

(1) When a warning status code has been set:
The status code is overwritten with an error status. It is not overwritten with a warning status except 3 cases following:

- If warning code is RSACPD_WARN_DISAGREE_INPUT_FS, it is overwritten with RSACPD_WARN_FI_MIX_AAC or RSACPD_WARN_AUTO_DS_SBR.
- If warning code is RSACPD_WARN_FI_MIX_AAC, it is overwritten with RSACPD_WARN_AUTO_DS_SBR.
- If warning code is RSACPD_RTN_FI_MULTI_CCE or RSACPD_WARN_ILLEGAL_CHAN_CONFIG, it is overwritten with new code.

(2) When an error status code has been set (only when the error conceal mode has been enabled by RSACPD_SetDecOpt()):
The status code is not overwritten, and the first detected error status code is kept set.

8.1 List of status codes

Table below shows the status codes of this middleware.

Table 8-1 List of status codes

Classification	Value	Code name	Description
Normal	0	RSACPD_RTN_GOOD	Normally ends.
Warning	5	RSACPD_WARN_PCE	Only PCE has been detected in a block. (Audio data is not included.)
	7	RSACPD_WARN_NO_AUDIO_DATA	No audio data has been detected in the frame. (For only PCE, the code 5 is returned.)
	10	RSACPD_WARN_ERR_ADTS_LEN	The value of the ADTS header frame_length does not correspond to the actually decoded ADTS frame size.
	12	RSACPD_WARN_UNSUPPORTED_CH_CFG	Channel configuration are not supported
	13	RSACPD_WARN_FI_MULTI_CCE	Multiple CCEs have been detected
	21	RSACPD_WARN_FI_GARBAGE	Detection of garbage data before a syncword when the ADTS header is acquired.
	22	RSACPD_WARN_FI_MIX_AAC	Detection of switching of input bit stream. (AAC <-> aacPlus)
	23	RSACPD_WARN_MIXDOWN_OVF	Overflow occurs during downmix
	25	RSACPD_WARN_DISAGREE_SBR_DATA	The SCE/CPE count is not identical to the SBR_DATA count.
	26	RSACPD_WARN_ERR_SBR_CRC_CHECK	Detection of SBR-CRC error.
	27	RSACPD_WARN_AUTO_DS_SBR	Execution of automatic downsampling.
	28	RSACPD_WARN_SEQUENCE	The sequence of execution of API functions is not supported.
	29	RSACPD_WARN_NOT_SUPPORT_SBR_FS	The sampling frequency of SBR is not supported.
	30	RSACPD_WARN_LFE_RESTRICT_ERR	Detection of constraint violation of LFE
	31	RSACPD_WARN_SBR_HEADER_ERR	Detection of SBR header error.
	33	RSACPD_WARN_DISAGREE_INPUT_FS	The sampling frequency of the previous input frame is different from the sampling frequency of the current input frame.
	39	RSACPD_WARN_AVG_PERFORMANCE	Reserved
	40	RSACPD_WARN_ILLEGAL_CHAN_CONFIG	Channel configuration is different from channel number.
	100	RSACPD_WARN_INVALID_DECODE_TYPE	The decoding type is invalid
Abnormal	-1	RSACPD_ERR_DATA_EMPTY	The end of input data has been detected.
	-2	RSACPD_ERR_NO_RAW_DATA_BLOCK	There is no raw_data_block to be decoded.

	-3	RSACPD_ERR_PARAM	An invalid value has been specified for the input argument.
	-10	RSACPD_ERR_NO_ADIF	ADIF header ID ("ADIF") cannot be detected when acquiring the ADIF header.
	-60	RSACPD_ERR_ADTS_DATA	Incorrect ADTS header format has been detected when acquiring the ADTS header.
	-90	RSACPD_ERR_SAMPLE_INDEX	An invalid value has been specified for the argument sampling frequency index of the RSACPD_SetFormat() function.
	-100	RSACPD_ERR_NOT_READY	Necessary information for decoding has not been acquired.
	-110	RSACPD_ERR_STREAM_DATA	Incorrect format has been detected.
	-111	RSACPD_ERR_SFB_TBL	An error has been detected in the SFB (scale factor) table data.
	-112	RSACPD_ERR_HUFFMAN	No Huffman code has been detected.
	-200	RSACPD_ERR_AAC_LC	An unsupported function other than the AAC-LC profile has been detected.
	-201	RSACPD_ERR_NOT_SUPPORT	A format other than AAC has been detected when acquiring the ADTS header.
	-202	RSACPD_ERR_CRC_CHECK	A CRC error in ADTS frame has been detected.
	-212	RSACPD_ERR_FI_GAIN	An unsupported function (gain control) has been detected.
	-213	RSACPD_ERR_FI_PREDI	An unsupported function (prediction) has been detected.
	-220	RSACPD_ERR_SEQUENCE	An incorrect API calling sequence has been detected.
	-403	RSACPD_ERR_AUDIO_MODE	Audio mode which is not supported
	-404	RSACPD_ERR_PCECNT	An invalid value has been set for the number of the available PCE information.
	-405	RSACPD_ERR_PCE_LOC	PCE has been detected after other elements
	-410	RSACPD_ERR_DRC_THREAD	Invalid DRC data has been detected
	-501	RSACPD_ERR_DISAGREE_OUTPUT_FS	The output sampling frequency of the immediately preceding frame is different from the current frame.
	-510	RSACPD_ERR_MIXDOWN_PARAM	Improper downmix argument
	-511	RSACPD_ERR_MIXDOWN_CH	Number of channels inappropriate to downmix
	-512	RSACPD_ERR_DSECNT	Improper setting of number of pieces of obtainable DSE information
	-700	RSACPD_ERR_LOAS_INFO	The LOAS header data is invalid or not supported

8.2 API functions and status codes

8.2.1 RSACPD_Open

Table 8-2 Status codes of RSACPD_Open() function

Classification	Code name	Output: RSACPD_AAC structure	Description	Processing by application program
Normal	RSACPD_RTN_GOOD	Defined	Initializing of the RSACPD_AAC structure and work area has been successfully completed.	Execute the next functions of this middleware according to the decode sequence.
Abnormal	RSACPD_ERR_PARAM	Undefined	Indicates that NULL pointer has been included in the argument or the input buffer length has been 0 or less.	Specify the correct arguments and re-execute this API function

8.2.2 RSACPD_SetPCEArea

Table 8-3 Status codes of RSACPD_SetPCEArea() function

Classification	Code name	Output: RSACPD_AAC structure	Output: RSACPD_PCE structure	Description	Processing by application program
Normal	RSACPD_RTN_GOOD	Defined	Undefined (Note 1)	The area for the RSACPD_PCE structure has been set normally.	Execute the next functions of this middleware according to the decode sequence.
Abnormal	RSACPD_ERR_PARAM	Undefined	Undefined	Indicates that NULL pointer has been included in the argument.	Specify the correct arguments and re-execute this API function
	RSACPD_ERR_SEQUENCE	Undefined	Undefined	The RSACPD_Open() function has not been successfully executed.	Execute the RSACPD_Open() function.
	RSACPD_ERR_PCECNT	Undefined	Undefined	An invalid value has been set for the number of the available PCE information.	Number of available PCE information is 1 to 16. Set the correct value, and secure the area.

(Note 1) Only area is secured. The content is not defined until the PCE elements are decoded.

8.2.3 RSACPD_GetAdifHeader

Table 8-4 Status codes of RSACPD_GetAdifHeader() function

Classification	Code name	Output			Description	Processing by application program
		RSACPD_AAC structure	RSACPD_AdifHeader structure	bcnt		
Normal	RSACPD_RTN_GOOD	Defined	Defined	ADIF header byte count	The ADIF header information has been stored in the structure successfully.	Execute the next functions of this middleware according to the decode sequence.
Abnormal	RSACPD_ERR_DATA_EMPTY	Undefined	Undefined	0	The end of the input data has been detected.	None
	RSACPD_ERR_PARAM	Undefined	Undefined	Undefined	Indicates that NULL pointer has been included in the argument.	Specify the correct arguments and re-execute this API function
	RSACPD_ERR_NO_ADIF	Undefined	Undefined	0	The ADIF header ID (“ADIF”) has not been detected.	Check the data format of the bit stream.
	RSACPD_ERR_STREAM_DATA	Undefined	Undefined	0	The bit stream format is abnormal.	Decoding cannot be continued further due to illegal bit stream. To continue decoding, check the bit stream, and execute the RSACPD_Open() function.
	RSACPD_ERR_SEQUENCE	Undefined	Undefined	Undefined	The RSACPD_Open() or function has not been successfully executed.	Execute the RSACPD_Open() function.
	RSACPD_ERR_AUDIO_MODE	Undefined	Undefined	Undefined	Unsupported audio mode	Decoding cannot be continued further due to unsupported function. To continue decoding, check the bit stream and execute RSACPD_Open() function.
	RSACPD_ERR_PCECNT	Undefined	Undefined	Undefined	There is a possibility that RSACPD_SetPCEArea() function has not been executed, or a number of pieces of PCE information more than the available number of pieces of PCE information set with RSACPD_SetPCEArea() has been detected.	Execute RSACPD_SetPCEArea() function, and set correctly the available number of pieces of PCE information.

8.2.4 RSACPD_GetAdtsHeader

Table 8-5 Status codes of RSACPD_GetAdtsHeader() function

Classification	Code name	Output			Description	Processing by application program
		RSACPD_AAC structure	RSACPD_AdtsHeader structure	bcnt		
Normal	RSACPD_RTN_GOOD	Defined	Defined	ADTS header byte count	The ADTS header information has been stored in the structure successfully.	Execute the next functions of this middleware according to the decode sequence.
Warning	RSACPD_WARN_FI_GARBAGE	Defined	Defined	ADTS header byte count + size of garbage data	Detection of garbage data before syncword	Execute the next functions of this middleware according to the decode sequence.
Abnormal	RSACPD_ERR_DATA_EMPTY	Undefined	Undefined	0	The end of the input data has been detected.	None
	RSACPD_ERR_PARAM	Undefined	Undefined	Undefined	Indicates that NULL pointer has been included in the argument.	Specify the correct arguments and re-execute this API function
	RSACPD_ERR_ADTS_DATA	Undefined	Undefined	Number of read bytes of header data	The value of the member sampling_frequency_index of the ADTS header is invalid. (value larger than 0xb)	Decoding cannot be continued further due to unsupported function.
	RSACPD_ERR_STREAM_DATA	Undefined	Undefined	0	An incorrect bit stream data has been detected.	Decoding cannot be continued further due to illegal bit stream. To continue decoding, check the bit stream and execute the RSACPD_Open() function.
	RSACPD_ERR_AAC_LC	Undefined	Undefined	Number of read bytes of header data	The value of the member profile of the ADTS header is invalid. (Not AAC-LC)	Decoding cannot be continued further due to unsupported function.
	RSACPD_ERR_NOT_SUPPORT	Undefined	Undefined	Number of read bytes of header data	The value of the member layer of the ADTS header is invalid. (Not 0x00)	Decoding cannot be continued further due to unsupported function.
	RSACPD_ERR_CRC_CHECK	Undefined	Undefined	0	A CRC error has been detected.	Decoding cannot be continued further due to illegal bit stream.
	RSACPD_ERR_SEQUENCE	Undefined	Undefined	Undefined	The RSACPD_Open() function has not been successfully executed.	Execute the RSACPD_Open() function.

8.2.5 RSACPD_GetLoasInfo

Table 8-6 Status codes of RSACPD_GetLoasInfo() function

Classification	Code name	Output: RSACPD_ AAC structure	Output: RSACPD_ LoasInfo structure	Output: bcnt	Description	Processing by application program
Normal	RSACPD_RTN_GOOD	Defined	Defined	LOAS header byte count	The LOAS header information has been stored in the structure successfully.	Execute the next functions of this middleware according to the decode sequence.
Warning	RSACPD_WARN_FI_GARBAGE	Defined	Defined	LOAS header byte count + size of garbage data	Detection of garbage data before syncword	Execute the next functions of this middleware according to the decode sequence.
Abnormal	RSACPD_ERR_DATA_EMPTY	Undefined	Undefined	0	The end of the input data has been detected.	None
	RSACPD_ERR_PARAM	Undefined	Undefined	Undefined	Indicates that NULL pointer has been included in the argument.	Specify the correct arguments and re-execute this API function
	RSACPD_ERR_LOAS_INFO	Undefined	Undefined	0	The LOAS header data is invalid or not supported.	.Decoding cannot be continued further due to unsupported function.
	RSACPD_ERR_STREAM_DATA	Undefined	Undefined	0	An incorrect bit stream data has been detected.	Decoding cannot be continued further due to illegal bit stream. To continue decoding, check the bit stream and execute the RSACPD_Open() function.
	RSACPD_ERR_AAC_LC	Undefined	Undefined	0	The value of the member profile of the LOAS header is invalid. (Not AAC-LC)	Decoding cannot be continued further due to unsupported function.
	RSACPD_ERR_AUDIO_MODE	Undefined	Undefined	0	Unsupported audio mode	Decoding cannot be continued further due to unsupported function. To continue decoding, check the bit stream and execute RSACPD_Open() function.
	RSACPD_ERR_SEQUENCE	Undefined	Undefined	Undefined	The RSACPD_Open() function has not been successfully executed.	Execute the RSACPD_Open() function.

8.2.6 RSACPD_SetFormat

Table 8-7 Status codes of RSACPD_SetFormat() function

Classification	Code name	Output RSACPD_AAC structure	Description	Processing by application program
Normal	RSACPD_RTN_GOOD	Defined	Information for decoding a bit stream in the RawDataStream-format has been correctly set.	Execute the next functions of this middleware according to the decode sequence.
Warning	RSACPD_WARN_INVALID_DECODE_TYPE	Defined	An invalid decoding type has been specified.	Execute the next functions of this middleware according to the decode sequence with decoding type is 0.
Abnormal	RSACPD_ERR_SAMPLE_INDEX	Undefined	An unsupported sampling frequency index has been specified.	Specify the correct sampling frequency index by referring Table 3-3 “Sampling frequency list (Sampling_frequency_index).”
	RSACPD_ERR_SEQUENCE	Undefined	The RSACPD_Open() function has not been successfully executed.	Execute the RSACPD_Open() and function. Note that this function should be called before the RSACPD_Decode() and the RSACPD_Skip() function.

8.2.7 RSACPD_Decode

Table 8-8 Status codes of RSACPD_Decode() function

Classification	Code name	Output			Description	Processing by application program
		RSACPD_AAC /RSACPD_OUT_INFO structure	bcnt	pnum		
Normal	RSACPD_RTN_GOOD	Defined	Number of bytes of current block	1024/2048 or 960/1920	Decode processing has been normally completed.	Execute the next functions of this middleware according to the decode sequence.
Warning	RSACPD_WARN_PCE	Defined	Number of bytes of current block	0	Decoding of blocks which contain only PCE has been completed.	The blocks containing only PCE are out of the standard. To continue decoding, check the bit stream. (Note 1)
	RSACPD_WARN_NO_AUDIO_DATA	Defined	Number of bytes of current block	0	No audio data has been detected in the block.	It is a bit stream out of the standard. To continue decoding, check the bit stream. (Note 1)
	RSACPD_WARN_ERR_ADTS_LEN	Defined	Number of bytes of current block	1024/2048 or 960/1920	The RSACPD_AdtsHeader type structure member frame_length does not correspond to the actually read ADTS frame length.	Decode processing has been normally completed, but the frame_length is incorrect. The output PCM data may be invalid.
	RSACPD_WARN_UNSUPPORTED_CH_CFG	Defined	Number of bytes of current block	1024/204 or 960/1920	The channel configuration which are not supported	The channel configuration is not supported. For more details, see 5.1 Definition of channel count".
	RSACPD_WARN_FI_MULTI_CCE	Defined	Number of bytes of current block	1024/2048 or 960/1920	Because two or more CCEs have been detected, only the firstly appeared CCE has been coupling-decoded.	Only one coupling channel is supported.
	RSACPD_WARN_FI_MIX_AAC	Defined	Number of bytes of current block	1024/2048 or 960/1920	The input bit stream has been switched (AAC <-> aacPlus)	Execute the next functions of this middleware according to the decode sequence.
	RSACPD_WARN_DISAGREE_SBR_DATA	Defined	Number of bytes of current block	1024/2048 or 960/1920	The SCE/CPE count is not identical to the SBR_DATA count.	When SBR_DATA is sufficient, decoding is executed normally. When SBR_DATA is insufficient, the upsampled PCM data is output. To continue decoding, execute the next functions of this middleware according to the decode sequence.
	RSACPD_WARN_ERR_SBR_CRC_CHECK	Defined	Number of bytes of current block	1024/2048 or 960/1920	Detection of SBR-CRC error.	The upsampled PCM data is output. To continue decoding, execute the next functions of this middleware according to the decode sequence.

	RSACPD_WARN_AUTO_DS_SBR	Defined	Number of bytes of current block	1024 or 960	Automatic downsampling has been executed.	Execute the next functions of this middleware according to the decode sequence.
	RSACPD_WARN_NOT_SUPPORT_SBR_FS	Defined	Number of bytes of current block	1024 or 960	Sampling frequency of unsupported SBR	SBR decoding is not executed, and only AAC PCM data is output. To continue decoding, execute the next functions of this middleware according to the decode sequence.
	RSACPD_WARN_LFE_RESTRICT_ERR	Defined	Number of bytes of current block	1024/2048 or 960/1920	Detection of constraint violation of LFE	The output PCM may be invalid. To continue decoding, execute the next function according to the decode sequence
	RSACPD_WARN_SBR_HEADER_ERR	Defined	Number of bytes of current block	1024/2048 or 960/1920	Detection of SBR header error	The upsampled PCM data is output. To continue decoding, execute the next functions of this middleware according to the decode sequence.
	RSACPD_WARN_DISAGREE_INPUT_FS	Defined	Number of bytes of current block	1024/2048 or 960/1920	Detection the difference of the input frequency of each frame	To continue decoding, execute the next function according to the decode sequence
	RSACPD_WARN_ILLEGAL_CHAN_CONFIG	Defined	Number of bytes of current block	1024/2048 or 960/1920	The channel configuration in ADTS header is not consistent to actual channel configuration.	The channel configuration is not supported. For more details, see Section 5.1"Definition of channel count".
Abnormal	RSACPD_ERR_DATA_EMPTY	Undefined	0 or Number of read bytes of current block	0	The end of the input data has been detected.	None
	RSACPD_ERR_NO_RAW_DATA_BLOCK	Undefined	0	0	There is no raw_data_block to be decoded.	None
	RSACPD_ERR_PARAM	Undefined	Undefined	Undefined	Indicates that NULL pointer has been included in the argument.	Specify the correct arguments and re-execute this API function
	RSACPD_ERR_NOT_READY	Undefined	Number of read bytes of current block	0	No information for decoding has been specified.	Execute the RSACPD_SetFormat() function to specify the information necessary for decoding when decoding in the RawDataStream-format.
	RSACPD_ERR_STREAM_DATA	Undefined	Number of read bytes of current block	0	An incorrect bit stream data has been detected.	Decoding cannot be continued further due to illegal bit stream.
	RSACPD_ERR_SFB_TBL	Undefined	Number of read bytes of current block	0	An incorrect scale factor value has been detected.	Decoding cannot be continued further due to illegal bit stream.

RSACPD_ERR_HUFFMAN	Undefined	Number of read bytes of current block	0	Huffman code error has been detected.	Decoding cannot be continued further due to illegal bit stream.
RSACPD_ERR_AAC_LC	Undefined	Number of read bytes of current block	0	The value of the member profile of the PCE element is invalid. (Not AAC-LC)	Decoding cannot be continued further due to illegal bit stream.
RSACPD_ERR_CRC_CHECK	Undefined	Number of read bytes of current block	0	A CRC error has been detected.	Decoding cannot be continued further due to illegal bit stream.
RSACPD_ERR_FI_GAIN	Undefined	Number of read bytes of current block	0	An unsupported function (gain control) has been detected.	Decoding cannot be continued further due to unsupported function. The gain control function has not been supported by AAC-LC profile.
RSACPD_ERR_FI_PREDI	Undefined	Number of read bytes of current block	0	An unsupported function (prediction) has been detected.	Decoding cannot be continued further due to unsupported function. The prediction function has not been supported by AAC-LC profile.
RSACPD_ERR_SEQUENCE	Undefined	Undefined	Undefined	The RSACPD_Open() has not been successfully executed.	Execute the RSACPD_Open() function.
RSACPD_ERR_AUDIO_MODE	Undefined	0 or Number of read bytes of current block	0	Unsupported audio mode.	Decoding cannot be continued further due to unsupported function.
RSACPD_ERR_PCECNT	Undefined	0 or Number of read bytes of current block	0	The PCE element count is invalid. (value larger than 16)	Decoding cannot be continued further due to illegal bit stream.
RSACPD_ERR_PCE_LOC	Undefined	Number of read bytes of current block	0	PCE has been detected after other elements.	Decoding cannot be continued further due to illegal bit stream.
RSACPD_ERR_DRC_THREAD	Undefined	0 or Number of read bytes of current block	0	Invalid DRC thread information.	Decoding cannot be continued further due to illegal bit stream.
RSACPD_ERR_DISAGREE_OUTPUT_FS	Undefined	Number of read bytes of current block	0	The output sampling frequency of the immediately preceding frame is different from the current frame.	Decoding cannot be continued further due to illegal bit stream

(Note 1) The bit stream can be decoded, when the subsequent bit stream is correct, although the bit stream is out of the standard.

8.2.8 RSACPD_Skip

Table 8-9 Status codes of RSACPD_Skip() function

Classification	Code name	Output		Description	Processing by application program
		RSACPD_ AAC structure	bcnt		
Normal	RSACPD_RTN_GOOD	Defined	Number of bytes of current block	Skip processing has been normally completed.	Execute the next functions of this middleware according to the decode sequence.
Warning	RSACPD_WARN_PCE	Defined	Number of bytes of current block	Decoding of blocks which contain only PCE has been completed.	The blocks containing only PCE are out of the standard. To continue decoding, check the bit stream. (Note 1)
	RSACPD_WARN_NO_AUDIO_DATA	Defined	Number of bytes of current block	No audio data has been detected in the block.	It is a bit stream out of the standard. To continue decoding, check the bit stream. (Note 1)
	RSACPD_WARN_ERR_ADTS_LEN	Defined	Number of bytes of current block	The RSACPD_ AdtsHeader type structure member frame_length does not correspond to the actually read ADTS frame length.	Decode processing has been normally completed, but the frame_length is incorrect. The output PCM data may be invalid.
	RSACPD_WARN_UNSUPPORTED_CH_CFG	Defined	Number of bytes of current block	Channel configuration which are not supported	Channel configuration is not supported. For more details, see Section 5.1 Definition of channel count".
	RSACPD_WARN_FI_MULTI_CCE	Defined	Number of bytes of current block	Because two or more CCEs have been detected, only the firstly appeared CCE has been coupling-decoded.	Only one coupling channel is supported.
	RSACPD_WARN_ILLEGAL_CHAN_CONFIG	Defined	Number of bytes of current block	Channel_configuration in ADTS header is not consistent to actual channel configuration.	Channelconfiguration is not supported. For more details, see Section 5.1"Definition of channel count".
Abnormal	RSACPD_ERR_DATA_EMPTY	Undefined	0 or Number of read bytes of current block	The end of the input data has been detected.	None
	RSACPD_ERR_NO_RAW_DATA_BLOCK	Undefined	0	There is no raw_data_block to be decoded.	None
	RSACPD_ERR_PARAM	Undefined	Undefined	Indicates that NULL pointer has been included in the argument.	Specify the correct arguments and re-execute this API function
	RSACPD_ERR_NOT_READY	Undefined	Number of read bytes of current block	No information for decoding has been specified.	Execute the RSACPD_SetFormat() function to specify the information necessary for decoding when decoding in the RawDataStream-format.

	RSACPD_ERR_STREAM_DATA	Undefined	Number of read bytes of current block	An incorrect bit stream data has been detected.	Decoding cannot be continued further due to illegal bit stream.
	RSACPD_ERR_SFB_TBL	Undefined	Number of read bytes of current block	An incorrect scale factor value has been detected.	Decoding cannot be continued further due to illegal bit stream.
	RSACPD_ERR_HUFFMAN	Undefined	Number of read bytes of current block	Huffman decode error has been detected.	Decoding cannot be continued further due to illegal bit stream.
	RSACPD_ERR_AAC_LC	Undefined	Number of read bytes of current block	The value of the member profile of the PCE element is invalid. (Not AAC-LC)	Decoding cannot be continued further due to illegal bit stream.
	RSACPD_ERR_CRC_CHECK	Undefined	Number of read bytes of current block	A CRC error has been detected.	Decoding cannot be continued further due to illegal bit stream.
	RSACPD_ERR_FI_GAIN	Undefined	Number of read bytes of current block	An unsupported function (gain control) has been detected.	Decoding cannot be continued further due to unsupported function. The gain control function has not been supported by AAC-LC profile.
	RSACPD_ERR_FI_PREDI	Undefined	Number of read bytes of current block	An unsupported function (prediction) has been detected.	Decoding cannot be continued further due to unsupported function. The prediction function has not been supported by AAC-LC profile.
	RSACPD_ERR_SEQUENCE	Undefined	Undefined	The RSACPD_Open() function has not been successfully executed.	Execute the RSACPD_Open() function.
	RSACPD_ERR_AUDIO_MODE	Undefined	0 or Number of read bytes of current block	Unsupported audio mode.	Decoding cannot be continued further due to unsupported function.
	RSACPD_ERR_PCECNT	Undefined	0 or Number of read bytes of current block	The PCE element count is invalid. (value larger than 16)	Decoding cannot be continued further due to illegal bit stream.
	RSACPD_ERR_PCE_LOC	Undefined	Number of read bytes of current block	PCE has been detected after other elements.	Decoding cannot be continued further due to illegal bit stream.
	RSACPD_ERR_DRC_THREAD	Undefined	Number of read bytes of current block	Invalid DRC thread information.	Decoding cannot be continued further due to illegal bit stream.

(Note 1) The bit stream can be decoded, when the subsequent bit stream is correct, although the bit stream is out of the standard.

8.2.9 RSACPD_DecodeStatus

Table 8-10 Status codes of RSACPD_DecodeStatus() function

Classification	Code name	Output Status	Description	Processing by application program
Normal	RSACPD_RTN_GOOD	Defined	Checking of the decode status has been normally executed.	Execute the next functions of this middleware according to the decode sequence.
Abnormal	RSACPD_ERR_PARAM	Undefined	Indicates that NULL pointer has been included in the argument.	Specify the correct arguments and re-execute this API function.
	RSACPD_ERR_SEQUENCE	Undefined	Illegal API calling sequence has been found.	Execute the RSACPD_Open() function. Note that this function should be called after the RSACPD_Decode() or the RSACPD_Skip() function.

8.2.10 RSACPD_SetDecOpt

Table 8-11 Status codes of RSACPD_SetDecOpt() function

Classification	Code name	Decode option	Description	Processing by application program
Normal	RSACPD_RTN_GOOD	Defined	The decode option has been set correctly.	Execute the next functions of this middleware according to the decode sequence.
Warning	RSACPD_WARN_SEQUENCE	Defined	The sequence of execution of API functions is not supported.	The user is responsible for switching decode options for each decoding.
Abnormal	RSACPD_ERR_SEQUENCE	Undefined	The RSACPD_Open() function has not been successfully executed.	Execute the RSACPD_Open() function.

8.2.11 RSACPD_SetDSE

Table 8-12 Status codes of RSACPD_SetDSE() function

Classification	Code name	Output		Description	Processing by application program
		RSACPD_ AAC structure	RSACPD_ DSE structure		
Normal	RSACPD_RTN_GOOD	Defined	Undefined (Note 1)	The area for the RSACPD_DSE structure has been set normally.	Execute the next functions of this middleware according to the decode sequence.
Abnormal	RSACPD_ERR_SEQUENCE	Undefined	Undefined	The RSACPD_Open() function has not been successfully executed.	Execute the RSACPD_Open() function.
	RSACPD_ERR_DSECNT	Undefined	Undefined	Improper setting of number of pieces of obtainable DSE information.	Specify the correct value to the argument dse_cnt (from 1 to 16)
	RSACPD_ERR_PARAM	Undefined	Undefined	Indicates that NULL pointer has been included in the argument.	Specify the correct arguments and re-execute this API function.

(Note 1) Only area is secured. The content is not defined until the DSE elements are decoded.

8.2.12 RSACPD_MatrixMixdown

Table 8-13 Status codes of RSACPD_MatrixMixdown() function

Classification	Code name	Output: RSACPD_AAC / RSACPD_OUT_INFO structure	Output: mixdown_mode / sel_std	Description	Processing by application program
Normal	RSACPD_RTN_GOOD	Defined	Defined	Downmix processing has been normally completed.	Execute the next functions of this middleware according to the decode sequence.
Warning	RSACPD_WARN_MIXDOWN_OVF	Defined	Defined	Downmix output is overflowed	User is responsible for switching downmix standards and modes when decoding
Abnormal	RSACPD_ERR_SEQUENCE	Undefined	Undefined	Illegal API calling sequence has been found.	This function should be called after the RSACPD_Decode() function.
	RSACPD_ERR_MIXDOWN_PARAM	Undefined	Undefined	Improper set values of arguments	Set correct values.
	RSACPD_ERR_PARAM	Undefined	Undefined	Indicates that NULL pointer has been included in the argument.	Specify the correct arguments and re-execute this API function.
	RSACPD_ERR_MIXDOWN_CH	Undefined	Undefined	Number of channels out of scope of downmix	The function is not applicable to downmix processing of channel configuration other than 3/2, 3/2 and LFE, 3/1, 2/2, 3/0, 2/1

8.2.13 RSACPD_SetSAC

Table 8-14 Status codes of RSACPD_SetSAC() function

Classification	Code name	Output		Description	Processing by application program
		RSACPD_AAC structure	RSACPD_SAC structure		
Normal	RSACPD_RTN_GOOD	Defined	Undefined (Note1)	The area for the RSACPD_SAC structure has been set normally.	Execute the next functions of this middleware according to the decode sequence.
Abnormal	RSACPD_ERR_SEQUENCE	Undefined	Undefined	The RSACPD_Open() function has not been successfully executed.	Execute the RSACPD_Open() function.
	RSACPD_ERR_PARAM	Undefined	Undefined	Indicates that NULL pointer has been included in the argument.	Specify the correct arguments and re-execute this API function.

(Note 1) Only area is secured. The content is not defined until the SAC elements are decoded.

8.2.14 RSACPD_SetDRC

Table 8-15 Status codes of RSACPD_SetDRC() function

Classification	Code name	Output		Description	Processing by application program
		RSACPD_AAC structure	RSACPD_DRC structure		
Normal	RSACPD_RTN_GOOD	Defined	Undefined (Note1)	The area for the RSACPD_DRC structure has been set normally.	Execute the next functions of this middleware according to the decode sequence.
Abnormal	RSACPD_ERR_SEQUENCE	Undefined	Undefined	The RSACPD_Open() function has not been successfully executed.	Execute the RSACPD_Open() function.
	RSACPD_ERR_PARAM	Undefined	Undefined	Indicates that NULL pointer has been included in the argument.	Specify the correct arguments and re-execute this API function.

(Note 1) Only area is secured. The content is not defined until the DRC elements are decoded.

9 Embedding Procedure

9.1 System configuration

Figure 9.1 shows an example of the system configuration of this middleware. The part enclosed in a dotted line corresponds to this middleware. The application program shall execute to read the memory of the bit stream and read from the PCM data memory.

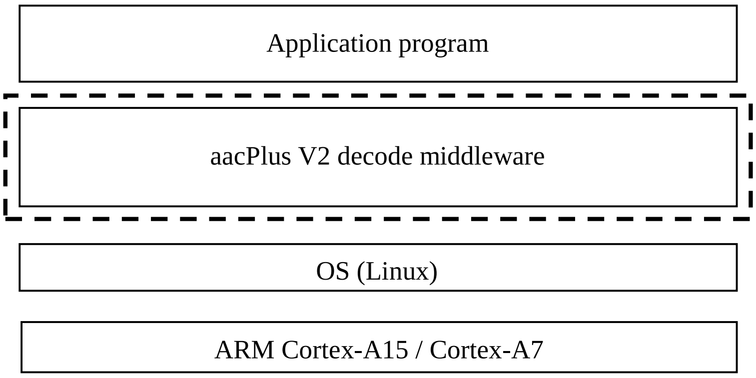


Figure 9.1 An example of system configuration

9.2 Development and evaluation environment

To embed this middleware to the application program, use the development tool listed in Table 9-1 or any compatible development environment.

Table 9-1 Development environment

Items	Description	Remarks
Target OS	Linux kernel release 3.10	See also related manual [1]
Board support package	Linux Interface Specification Yocto recipe	
Tool chain	arm-poky-linux-gnueabi-gcc (Linaro GCC 4.8-2014.04) 4.8.3 20140401 (prerelease)	

9.3 Contents of middleware

The middleware consists of the files listed below.

Table 9-2 Contents of middleware

	File Name	Contents
1	libRSACPDLA_L.so.1.1	Dynamic-link library
2	RSACPD_ADL.h	Header file
3	RTM0AC0000ADAAPMZ1SL32E-02.pdf	User's manual (English)
4	RTM0AC0000ADAAPMZ1SL32J-02.pdf	User's manual (Japanese)
5	sample_main.c	Sample program in C language (Note 1)

(Note 1) The sample program does not guarantee the behavior in user system.

9.4 Creating User Program

Create the user program which calls API functions of this middleware and the User-created function (see Section 4 "User-created function"). Include the below header files:

Table 9-3 List of header files of user program

Header file	Description
RSACPD_ADL.h	Provided with the middleware
string.h	Standard library header

9.5 Setting Compile Option

Table 9-4 shows the compile option to be set when compile this middleware library. Please confirm Table 9-4 when link this middleware to the user system.

Table 9-4 Compile Option

	Compile Option	Setting value	Description
1	Optimization	-O2	Set O2 optimization level
2	Character signed	-fsigned-char	Set char to signed char

10 Notes

This Section describes that precaution of using this middleware.

The notes to use the each decoder API are described as <Notes> of each API function in Section 3.2

10.1 Reserved word

For this middleware, "RSACPD_" is added to the head of function names and macro names to distinguish this application program from other application programs. To avoid competition, do not use functions and variables starting with "RSACPD_" for other application programs which use this middleware.

10.2 Restoration after termination with error

After RSACPD_Decode() or RSACPD_Skip() has ended with an error or before a different stream is decoded, perform initialization with RSACPD_Open().

10.3 Monitoring on the Performance

The products embedding this middleware shall observe performance of the middleware periodically with Watch Dog timer or such functions in order not to damage system performance.

10.4 Decoding of mixed aacPlus and AAC coded bit streams

When this middleware detects switching a type of a coded bit stream such as AAC to aacPlus during decoding, this middleware resets inner procedure. Therefore muting period might be occurred in a half of a frame in maximum.

When decoding mixed aacPlus and AAC coded bit stream, the aacPlus frame and the AAC frame differ in the number of output words (the rate of the AAC frame is half of that of the aacPlus frame). When decoding mixed coded bit streams, it is necessary to check the number of words in the output PCM data (see Section 3.2.7 "RSACPD_Decode").

10.5 Operation when SBR header information has not been acquired

When the middleware decodes an aacPlus coded bit stream, it executes upsampling without performing aacPlus decoding until it receives SBR header information from the bit stream. Therefore, the output sampling frequency will not be changed after the SBR header is acquired (except when a bit stream not containing aacPlus data is decoded).

10.6 Decoding of mixed channel configuration

10.6.1 Change of the number of output channels

It is possible to continue decoding if the number of channels is varied during decoding a bit stream. The number of output channels is the same as input data. Therefore, application program may observe the number of channels of output PCM data and manage it properly in case of varying the number of channels such as from monaural to stereo.

When decoding the bit stream which includes parametric stereo (PS) data, the number of channels of output PCM data might be changed by frame according to the condition of SBR header existence and PS data, it is recommended to observe the number of output channels and manage it properly.

10.6.2 Change of the channel configuration

When channel configuration in an input bit stream is changed, this middleware resets inner procedure. Therefore muting period might be occurred in a half of a frame in maximum.

10.7 Alternation of output sampling frequency

When the alternation of output sampling frequency occurs due to a change of sampling frequency in a bit stream, type of a coded bit stream (AAC/aacPlus V1/V2) or decode option, this middleware operates like below.

- In case that error conceal mode setting flag is effective
  The error concealed PCM data with the sampling frequency of previous frame is output.
  Though the error conceal is proceeded continuously after the frame, the output PCM data might be silent when the output sampling frequency differs from the previous frame which ends normally.
- In case that error conceal option is ineffective
  This middleware ends abnormally.

Note that if one or more frames are skipped by RSACPD_Skip() function during decoding, this middleware continues decoding independently of setting frequency is changed.

Application program may observe the output sampling frequency and conduct necessary operation.

Appendix

The RSACPD_AAC structure type information

RSACPD_AAC type structure information

Member name	Description
unsigned char *BsBuf	Pointer to the beginning of input buffer
unsigned char *BsBufIdx	Indicate the address to read current raw data block
unsigned long UseBitCount	The number of bit are used
unsigned long nNoUseBit	The number of remained bit in Bs4Byte
unsigned long Bs4Byte	Buffer to read the necessary information
unsigned long BsBufSize	Size of the read data in input buffer
RSACPD_PCE *pcebuf	Pointer to the PCE buffer
RSACPD_SAC *sacbuf	Pointer to the SAC buffer
RSACPD_DRC *drcbuf	Pointer to the DRC buffer
RSACPD_PCE *ppcebuf[RSACPD_MAX_ELE_TAG]	Pointer to each PCE element buffer
RASCPD_DSE *pdsebuf[RSACPD_MAX_ELE_TAG]	Pointer to each DSE element buffer
char AudioObjectType	The audio object type of the input bit stream
char sampling_frequency_index	The index of sampling frequency of the input bit stream
unsigned char id	ID of the current element
unsigned char common_window	The flag indicating two individual_channel_stream share a common ics_info or not
struct RSACPD_ChannelData1 *pCD1	Pointer to the channel data CD1
struct RSACPD_ChannelData2 *pCD2	Pointer to the channel data CD2
struct RSACPD_ChannelData1 *pCDCW1	Pointer to the saved channel data
struct RSACPD_CCE CCE	CCE working data
SPEC *spec	MDCT spectral working data
int sequence_number	Sequence number
int ahcod[MAX_CHANNEL_NUM][1024]	Help element: decoder working data
int x_quant[1024]	Help element: decoder working data
int is_position[120]	Help element: decoder working data
int *hcod	Pointer to the ahcod when in decoding process
int default_config	Default configuration
int implicit_ch_cfg	Flag indicating the implicit channel configure
int current_program	The current program
int adts_channel_config	The ADTS header channel configuration
int first_block	Help element: temporary working data
int max_pce_cnt	Max PCE count in the current bit stream
int element_skip[RSACPD_Chans]	Flag mark the element is skipped in decoding process
SPEC *channel_spec[MAX_CHANNEL_NUM]	Pointer to the spectral buffer of each channel
struct RSACPD_ChannelData1 CD1[MAX_CHANNEL_NUM]	Channel data information 1 of each channel
struct RSACPD_ChannelData2 CD2[MAX_CHANNEL_NUM]	Channel data information 2 of each channel
char ID	ADTS header ID data
char header_type	Header type
unsigned int (*UserFunc)(unsigned char *, int)	Function pointer to user defined read data function
int ApiSeqNum	API sequence number
long RandomSeed[8][64]	The random vector seed use for decoding process
long CurrentSeed	The current seed number
void *crc	Pointer to the CRC data (RSACPD_CRCDATA)
unsigned char CrcFlag	CRC control flag 0 : Disable CRC reading 1 : Enable CRC reading
unsigned short FileCRC	The bit stream CRC value, checking with CalcCRC
unsigned short CalcCRC	The calculated CRC value, checking with FileCRC
unsigned short frame_length	ADTS frame length
RSACPD_CRC RSACPD_CRCDATA	The CRC working data
int audioCountSCE	SCE counter

int audioCountCPE	CPE counter
int audioCountLFE	LFE counter
int audioCountCCE	CCE counter
int audioCountPCE	PCE counter
int audioCountDSE	DSE counter
int audioCountSAC	SAC counter
int audioCountDRC	DRC counter
int flag_AAC_reset	Flags indicating the reset step is required
int curr_AAC_DecodeMode	Current AAC decode mode
unsigned int all_spectraldata_bit	The number of spectral data bit is used
int get_pce_cnt	The PCE count number acquired from ADIF/LOAS header
int crc_check_flag	The CRC checking flag 0 : No CRC checking 1 : CRC checking
int frame_len_cnt	ADTS frame length counter
int sac_enable	SAC status flag 0 : SAC disable 1 : SAC enable
int max_dse_cnt	Max DSE element count
int drc_scale	DRC scale
int drc_enable	DRC enable flag 0 : DRC disable 1 : DRC enable
int SkipFlag	Flag indicating skip is effective
short *PcmBuf[MAX_CHANNEL_NUM]	Output PCM buffer
short ChannelNumber	The number channel of output data
int PcmLen	The number of PCM samples per channel
int ch_Index	Number of current channel index is processed
int next_ch_Index	Number of the next channel index will be processed
int DErrorCode	The decode error code
RSACPD_OUT_INFO *pOutInfo	Pointer to the RSACPD_OUT_INFO data
RSACPD_OUT_INFO prevOutInfo	The value of the previous RSACPD_OUT_INFO data
RSACPD_MC_Info MC_Info	Multichannel information
RSACPD_MC_Info prev_mc_info	Previous frame channel information
RSACPD_MC_Info save_mc_info	Saved multichannel information
SPEC overLapBuffer[1024*MAX_CHANNEL_NUM]	Overlapping spectral buffer
unsigned long buf_len	Size of input buffer
char RSACPD_flag	SBR decoding flag
int channelMode	Channel mode value
REENTRANCY1 entrancy1	Reentrancy data
int dec_mode	Decoding mode
int flag_error_conceal	Flag indicating error conceal is executing
int sbr_behavior	SBR decoding behavior
int prev_SBR_DecodeMode	Previous SBR decode mode
int curr_SBR_DecodeMode	Current SBR decode mode
int prev_sbr_behavior	Previous SBR decoding behavior
int prev_PcmLen	Previous frame PCM output samples number
int curr_sbrDataNum	The number of current SBR data number
int prev_sbrDataNum	The number of previous SBR data number
int prev_FS	The previous sampling frequency
int curr_FS	The current sampling frequency
int cc_enable	Coupling channel enable flag
int decode_cce	The decoding CCE status
int indicate_pce_flag	Flag indicating PCE is defined
int prev_cc_enable	Previous coupling channel enable flag
int curr_pce_detected	Flag indicating PCE is detected
unsigned char bUpSample;	Upsample SBR flag 0 : Upsample AAC off 1 : Upsample AAC on
int bDownSample	Down-sampling SBR flag 0 : Down-sampling SBR off

	1 : Down-sampling SBR on
int ForceSbrOff	Force AAC flag 0 : SBR process is off 1 : SBR process is on
unsigned char useHqSbr	The HQ-SBR usage data
SBRBITSTREAM streamSBR	SBR stream data
SBRDECODER RSACPD_sbrDecoderInfo	SBR decoding information
struct SBR_DECODER_INSTANCE SBR_DECODER_INSTANCE_DATA	SBR decoding working data
int sbr_header_err_detect	SBR header error detected flag
int sbr_crc_err_detect	SBR CRC error detected flag
int curr_ps_detected	PS data detected flag
int flag_force_SBR_Up	SBR force up-sampling flag
int force_SBR_Up_enable	SBR force up-sampling enable flag
RSACPD_LoasInfo prev_header	The previous LOAS frame header data
unsigned long UseBitCount_LOAS	The number of LOAS header data is used
unsigned short frame_size	The number output samples per frames (1024/960)
LONG32 sbr_drcFactorVector[6][26][32]	The DRC Factor vector using for calculation DRC

The RSACPD_DRC_Bitstream structure type information

Member name	Description
int excl_chn_present;	One bit indicating that excluded channels are present
int excl_chn_mask[RSACPD_MAX_CHAN];	Boolean array indicating the audio channels of a program that are excluded from DRC processing using this DRC information
int num_bands;	The number of bands greater than one if there is multi-band DRC information
int band_top[RSACPD_MAX_DRC_BANDS];	Indicates top of i-th DRC band in units of 4 spectral lines.
int prog_ref_level_present;	One bit indicating that reference level is present
int prog_ref_level;	Reference level. A measure of long-term program audio level for all channels combined.
int drc_sgn[RSACPD_MAX_DRC_BANDS];	Dynamic range control sign information. One bit indicating the sign of drc_mag (0 if positive, 1 if negative)
int drc_mag[RSACPD_MAX_DRC_BANDS];	Dynamic range control magnitude information
int drc_interp_scheme;	Indicates which interpolation scheme is used for the DRC data in the SBR QMF domain

The RSACPD_DRC_Info structure type information

Member name	Description
int num_bands;	The number of bands greater than one if there is multi-band DRC information
int drc_interp_scheme;	Indicates which interpolation scheme is used for the DRC data in the SBR QMF domain
int band_top[RSACPD_MAX_DRC_BANDS];	Indicates top of i-th DRC band in units of 4 spectral lines.
int drc_sgn[RSACPD_MAX_DRC_BANDS];	Dynamic range control sign information. One bit indicating the sign of drc_mag (0 if positive, 1 if negative)
int drc_mag[RSACPD_MAX_DRC_BANDS];	Dynamic range control magnitude information

REVISION HISTORY	ARM 5.1ch aacPlus V2 Decode Middleware for Linux User's Manual

Rev.	Date	Description	
		Page	**Summary**
1.00	Sep. 30, 2014	-	First Edition issued
1.01	Dec. 05, 2014	28	Update RSACPD_get_version return value.
		57-70	Update API functions and status codes.
		72	Update the List of files.
		84	Add the member "prev_mc_info".

ARM 5.1ch aacPlus V2 Decode Middleware for Linux
RTM0AC0000ADAAPMZ1SL32C
User's Manual

Publication Date: Rev.1.01 December 05, 2014
Published by: Renesas Electronics Corporation

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Ku, Seoul, 135-920, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 3.0

ARM 5.1ch aacPlus V2 Decode Middleware for Linux

RTM0AC0000ADAAPMZ1SL32C

Renesas Electronics Corporation