User's Manual

ARM AAC Encode Middleware for Linux

RTM0AC0000AEAACMZ1SL32C

User's Manual

RTM0AC0000AEAACMZ1SL32E-01

Renesas Electronics
www.renesas.com

Rev. 1.00 Aug. 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; anticrime 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)

How to Use This Manual

1. Purpose and Target Readers

This manual is designed to provide the user with an understanding of the interface specification of this middleware. It is targeted at people who wish to design application systems which use the middleware. Please refer to related documentations with this manual.

Use this middleware after carefully reading the precautions. The precautions are stated in the main text of each section, at the end of each section, and in the usage precaution section.

The revision history summarizes major corrections and additions to the previous version. It does not cover all the changes. For details, refer to this manual.

2. Use of This Product

To use the AAC 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/)

To use this product, you need to enter into a software license agreement with Renesas Electronics.
Any customer cannot redistribute a part of or the entire product unless otherwise separately agreed.
Products in which this software is embedded may fall under the categories of cargo and technology that are regulated by the Foreign Exchange and Foreign Trade Control Law. Before exporting the product (including carrying the product out of Japan), confirm if the product falls under the categories of the restriction cargo and technology, and follow the Foreign Exchange and Foreign Trade Control Law to obtain the appropriate license such as export admission from the Japanese Government.

3. Related Manuals

[1] Linux Interface Specification Yocto recipe Start-Up Guide

[2] R-Car Series, 2nd Generation User’s Manual: Hardware

4. Technical Terms and Abbreviation

Abbreviation	Full Form
AAC	Advanced Audio Coding
PCM	Pulse Code Modulation
CRC	Cyclic Redundancy Check
SCE	Single channel element
CPE	Channel Pair Element

Table of Contents

1. Overview

1.1. Basic Specifications

This Middleware converts 16 bit PCM format data to AAC(ISO/IEC 13818-7:2006, ISO/IEC 13818-7:2006/Amd.1:2007) coded bitstreams. The basic specifications of this middleware are shown in Table 1.1 and Table 1.2.

Table 1.1 Basic Specifications(1)

Item	Contents		
Product Type Name	RTM0AC0000AEAACMZ1SL32C		
Product Name	ARM AAC Encode Middleware for Linux		
Version	0x20010211(*1)		
Target CPU	ARM		
Encoding Specification	Conformity Standard		MPEG-2 Advanced Audio Coding + ISO/IEC13818-7:2006 Forth Edition + ISO/IEC13818-7:2006/Amd.1:2007
	Profile		ISO/IEC-13818-7:2006 Low Complexity
	Input Format		16 bit PCM
	Output Format		AAC bitstream(ADIF, ADTS, raw data)
	Channel Configuration		1: Monaural
			2: Stereo, Dual Monaural
	Sampling Frequency		8, 11.025, 12, 16, 22.05, 24, 32, 44.1, 48 [kHz]
	Bitrate (*2)		8k to 288k [bits/sec/ch]
		Sampling Frequency [kHz]	Bitrate [bits/sec/ch]
		8	8 k to 48 k
		11.025	8 k to 66.15 k
		12	8 k to 72 k
		16	8 k to 96 k
		22.05	8 k to 132.3 k
		24	8 k to 144 k
		32	8 k to 192 k
		44.1	8 k to 264.6 k
		48	8 k to 288 k

(*1) Version information that can be obtained by RSAACE_get_version function

(*2) It is necessary for the bit rate to be set at a good balance with the sound quality adequately.

Table 1.2 Basic Specifications(2)

Item	Contents	
Performance	26.10MHz(*1)	
Program Size	ROM	202626bytes
	RAM	56584 bytes
	STACK	2000 bytes
	Input Buffer	Recommended 4096 bytes or larger (allocated by application)
	Output Buffer	Recommended 1536 bytes or larger (allocated by application)
Use Form	C Language Interface (Library Function)	
Restrictions	+ CRC (Cyclic redundancy code) is not added to bitstream. + This middleware is reentrant.	

(*1) This value is not the definitive in the all cases.
It is the refference value when encoding to AAC bitstream of 128k[bits/sec] from the general musical CD which were recorded in PCM of 44.1[kHz] stereo on Cortex-A15 of R-Car M2.
The peak value is not a limit of the values above.

2. API Specifications

This chapter describes API functions and data structures provided by this middleware.

2.1. List of API Functions and Data Structures

Table 2.1 and Table 2.2 list API functions and data structures provided by this middleware respectively.

Table 2.1 API Functions

No.	Function Name	Description of Functions
1	RSAACE_Init	This function initializes this middleware.
2	RSAACE_Encode	This function encodes one frame of PCM data.
3	RSAACE_get_version	This function obtains the version number of this middleware.

Table 2.2 Data Structures

No.	Data Structure Name	Type	Description
1	RSAACE_AAC	Structure	Structure for work area to use in this middleware
2	RSAACE_AACInfo	Structure	Structure to store encode parameters

2.2. Details of API Functions

2.2.1. RSAACE_Init Function

This function initializes the work area of this middleware. When this function is executed without error, RSAACE_R_GOOD is returned as the return value, and the status code, RSAACE_NO _ERROR, is set to the member 'AAC_status' of the work structure. When an application program gives an incorrect parameter to this function, RSAACE_E_ERROR is returned as the return value, and a certain status code is set to the member AAC_status. Refer to Section 4.1 for details of the return values and the status codes.
This function shall be called before execution of the RSAACE_Encode function. And when subsequently encoding a string of sounds with different sampling frequency and channel mode, prior to the execution of the RSAACE_Encode function, in each case, this function must be called.

<Syntax>

```
RSAACE_TYPE_ERROR RSAACE_Init (
    RSAACE_AAC *work,
    RSAACE_AACInfo *info
);
```

<Arguments>

work	[Input/Output]	Pointer to the structure for work area to use in this middleware
info	[Input]	Pointer to the structure to which the application program sets the encode parameter

<Return Value>

(Normal)	
RSAACE_R_GOOD	Initialization completed successfully.
(Error)	
RSAACE_E_ERROR	This function has incorrect encode parameters. See Table 2.3 for the encode parameter. Refer to Section 4.1 for details of the status codes.

2.2.2. RSAACE_Encode Function

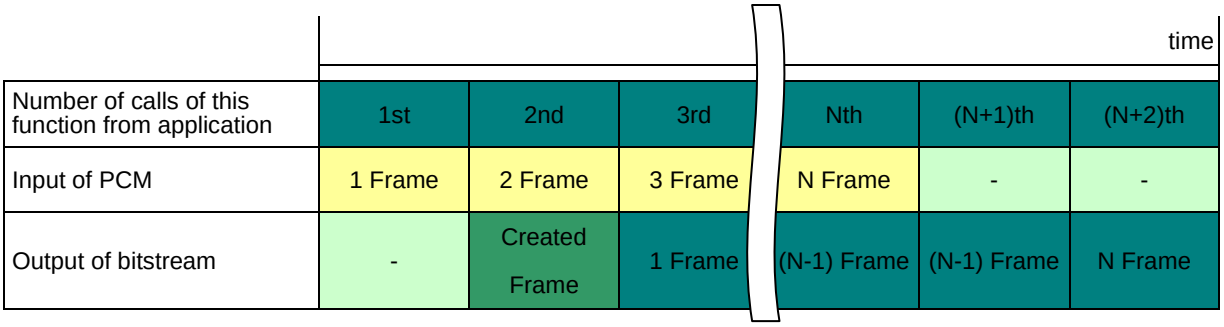
This function obtains PCM data from the address indicated by the argument 'input', encodes one frame of the data(2 frames before the input as shown in Figure 2.1), and then outputs bitstream to the output buffer specified by the argument 'output'. The number of bytes of the output bitstream is output to the address indicated by the argument 'bcnt'.

When the number of samples specified by the argument ‘pcnt’ is one frame or more, this function executes the encoding and returns RSAACE_R_GOOD as the return value, and sets the status code RSAACE_R_NO_ERROR to the member ‘AAC_status’ of the work structure.
However, as shown in Figure 2.1, the bitstream is not output in the first call ‘1st’ of this function just after the RSAACE _Init function is executed. (Zero (0) is output to the address indicated by the argument ‘bcnt’.) At this time, this function returns RSAACE_C_CHECK as the return value, and sets the status code RSAACE_C_PRE_ENCODE to the member ‘AAC_status’ of the work structure. And also, in the second call ‘2nd’, this funciton output the same as in the first call ‘1st’ when a value of member ‘gap’ of RSAACE_AACInfo structure, ‘*info’ has been set ‘0’, otherwise, when a value of member ‘gap’ of RSAACE_AACInfo structure, ‘*info’ has been set ‘1’, this function adds one silence frame of the top of PCM and generates a bitstream as the created frame.

If the number of samples specified by the argument 'pcnt' is insufficient for one frame, this function obtains PCM data with the insufficient sample values defined as 0, executes the encoding and returns RSAACE_C_CHECK as the return value, and sets the status code RSAACE_C_LACK_FRAME to the member 'AAC_status' of the work structure. PCM data should be sufficient for one frame encoding in the input buffer unless the input PCM data is the end. (Refer to Section 3.1)

If finishing to input PCM at the end of music etc., this function should be executed continuously with the number of samples specified by the argument ‘pcnt’ set zero (0). At this time, it should be confirmed by user program whether return code is RSAACE_C_CHECK and status code is finally set to RSAACE_C_ END_OF_FILE as show the below.
When finishing to input PCM at the end of music etc., by setting the number of samples specified by the argument ‘pcnt’ to zero (0), this function executes the encoding of one fame before final frame and returns RSAACE_C_CHECK as the return value, and sets the status code RSAACE_C_BEFORE_END_OF_FILE to the member 'AAC_status' of the work structure. However, as shown in Figure 2.1, in the first call ‘1st’ of this function, encoding is not executed and this function returns RSAACE_C_CHECK as the return value and set the status code RSAACE_C_END_OF _FILE to the member ‘AAC_status’ of the work structure.
After finishing to input PCM at the end of music etc., by setting the number of samples specified by the argument ‘pcnt’ to zero (0), this function executes the encoding of final frame and returns and returns RSAACE_C_CHECK as the return value, and sets the status code RSAACE_C_END_OF_ FILE to the member 'AAC_status' of the work structure.

Before this function is called, the RSAACE_Init function shall be executed.
When the status code is RSAACE_C_LACK_FRAME, RSAACE_C_BEFORE_END_OF_FILE or RSAACE_C_END_OF_FILE, encoding can be continued by supplying PCM data to the input buffer. However, it is recommended to execute RSAACE_Init function before this function is called next.



Number of calls of this function from application	1st	2nd	3rd	Nth	(N+1)th	(N+2)th
Input of PCM	1 Frame	2 Frame	3 Frame	N Frame	-	-
Output of bitstream	-	Created Frame	1 Frame	(N-1) Frame	(N-1) Frame	N Frame

[Note] N : PCM consists of N frames.
Created Frame : When the value of ‘gap’ equals ‘1’, at the same call(2) this adds one silence frame of the top of PCM and generates a bit stream.

Figure 2.1 Number of calls of this function, input of PCM, and output of bitstream

<Syntax>

```
RSAACE_TYPE_ERROR RSAACE_Encode (
    RSAACE_AAC *work,
    short *input,
    unsigned int *pcnt,
    unsigned char *output,
    unsigned int *bcnt,
);
```

<Arguments>

work	[input/output]	Pointer to the work structure for this middleware.	
input	[input]	Start address of input PCM data.	
pcnt	[input/output]	[input]	Pointer to the number of valid input PCM data samples. Refer to chapter 3 for details of input format.
		[output]	Pointer to the number of encoded PCM data samples.
output	[input]	Pointer to the output buffer where bitstream data is stored. Refer to chapter 3 for details of output formas.	
bcnt	[output]	Pointer to the number of bytes of the output bitstream data.	

<Return Value>

(Normal)	
RSAACE_R_GOOD	Encoding completed successfully.
(Error)	
RSAACE_E_ERROR	This function has incorrect arguments.
(Check)	
RSAACE_C_CHECK	Check the status code.

2.2.3. RSAACE_get_version Function

This function returns the version number of this middleware.

<Syntax>

unsigned int RSAACE_get_version (void);

<Return Value>

The return value is the version number of this middleware in hexadecimal 0xabbbccde.

a : Version number

bbb : Revision number

cc : Build number

d : Reserved

e : Reserved

2.3. Details of Data Structure

2.3.1. RSAACE_AAC Structure

RSAACE_AAC structure is the work structure used by this middleware. The contents of the member 'AAC' are closed. Do not refer and change the data in the structure.
The area of this structure should be aligned to an even alignment.

<Syntax>

```
#define WORKSIZE            13986
typedef struct _RSAACE_AAC
{
        int AAC_status;
        int AAC[WORKSIZE];
} RSAACE_AAC;
```

<Members>

AAC_status	Status code
AAC[WORKSIZE]	Work area for this middleware

2.3.2. RSAACE_AACInfo Structure

RSAACE_AACInfo structure is used to set the encode parameters from an application program to this middleware. After the encode parameters are set to each member of this structure, RSAACE_Init function shall be called. Table 2.3 lists the encode parameter values to be set to the RSAACE _AACInfo structure.

<Syntax>

```
typedef struct _RSAACE_AACInfo
{
    unsigned int    channelMode;
    unsigned int    sampleRate;
    unsigned int    bitRate;
    unsigned int    outputFormat;
    unsigned int    enable_cbr;
    unsigned int    gap;
    unsigned int    mode;
    unsigned int    home;
    unsigned int    original_copy;
    unsigned int    copyright_id_present;
    unsigned char   copyright_id[9];
} RSAACE_AACInfo;
```

<Members>

channelMode	Channel mode of PCM
sampleRate	Sampling frequency [Hz] of PCM
bitRate	Bit rate [bits/sec/ch] of bitstream
outputFormat	Format of bitstream
enable_cbr	Setting of constant bit rate (CBR)
gap	Setting of encoding silence frame to the head.
mode	Setting of speed priority mode
home	Bit indicating copy or original
original_copy	Bit indicating whether the copyright is protected
copyright_id_present	Bit indicating whether copyright_id[] exists
copyright_id[9]	Bit field to store identification code such as copyright conforming to international standards. e.g., ISAN(International Standard Audiovisual Number), etc.

Table 2.3 Value to be set to RSAACE_AACInfo structure

Parameter	Contents		Setting value
channelMode	Monaural		0（Initial value）
	Stereo		1
	Dual Monaural		2
sampleRate	Sampling Frequency [Hz]		8000
			11025
			12000
			16000
			22050
			24000
			32000
			44100
			48000
bitRate(*1)	Bitrate [bits/sec/ch]	fs = 8000	8000 to 48000
		fs = 11025	8000 to 66150
		fs = 12000	8000 to 72000
		fs = 16000	8000 to 96000
		fs = 22050	8000 to 132300
		fs = 24000	8000 to 144000
		fs = 32000	8000 to 192000
		fs = 44100	8000 to 264600
		fs = 48000	8000 to 288000
outputFormat	ADTS		0（Initial value）
	ADIF		1
	raw data		2
cbr_enable	Variable bitrate (VBR) (*2)		0（Initial value）
	Constant bitrate (CBR)		1
gap	Start frame of iput frame		0（Initial value）
	Start frame of added silence		1
mode	Normal		0（Initial value）
	speed priority mode		1
home	The bitstream is copy.		0（Initial value）
	The bitstream is original.		1
original_copy	The bitstream is uncopyrighted.		0（Initial value）
	The bitstream is copyrighted.		1
copyright_id_present	No copyright_id field exists.		0（Initial value）
	copyright_id field exists.		1
copyright_id[9](*3)	8-bit copyright_identifier, 64-bit copyright_number		Arbitrary bit string, or 0 Example) copyright_identifier: I.S.A.N (for audiovisual works) copyright_number: 1234567890123456 (ISAM number)

(*1) The member ‘bitRate’ must be set a rate value by a value [bits/sec] unit per the channel. In this table, the bottom value, the upper limit value in the parenthesis are listed every sampling frequency.

(*2) In the case of variable bit rate (VBR) ‘cbr_enable = 0’, there is the case that the mesured bit rate in the output stream doesn’t run out the value set in ‘bitRate’. This is dependent on each sound source.

(*3) The identification code is controlled with "Registration Authority". To get the code, the procedure of "Request of a Registered Identifier (RID)" is required. For more details, contact "Registration Authority".

3. I/O Data Format

3.1. Input Data Format

Figure 3.1 shows the input PCM format for this middleware. For the encode of one frame, 1024 samples are used. That is, the input buffer size is needed more than 2048 bytes, which is the minimum PCM data size when one frame is encoded in monaural.This is 4096 bytes when encoding stereo and dual monaural PCM data.

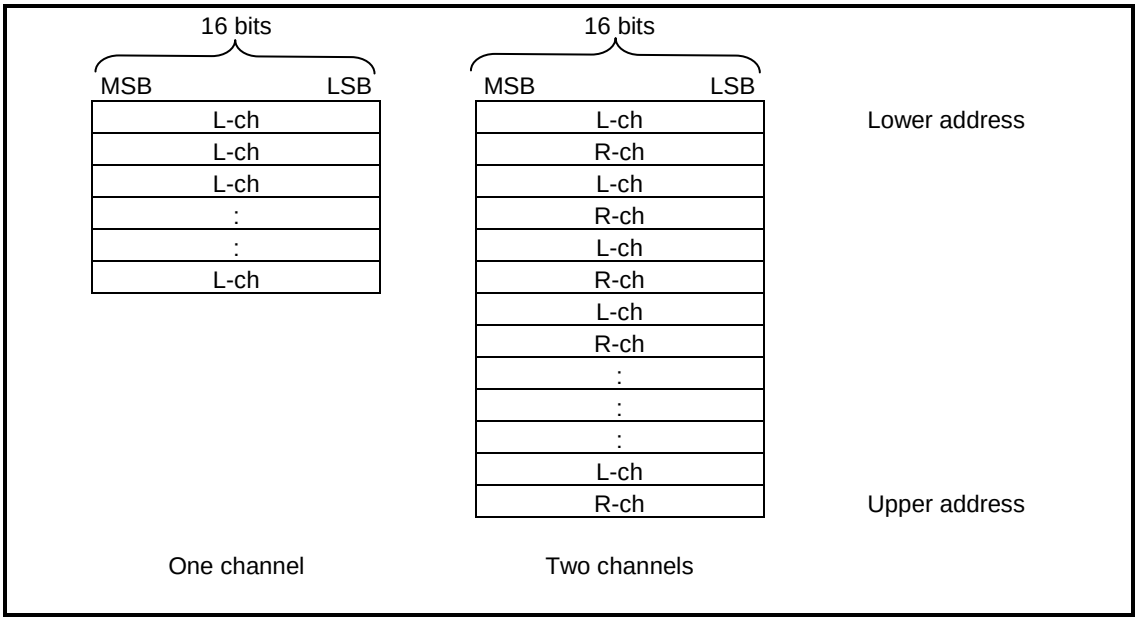


Figure 3.1 Format of Input PCM

3.2. Output Data Format

AAC bitstream consists of continuous frames. The AAC form is ADTS, ADIF, or raw data. The diagram below shows the structure of the bitstream.

3.2.1. ADTS

In each frame of ADTS, bitstream consists of only one block (a raw_data_block). In the first bit of the ADTS frame, 12-bit synchronous word should be inserted, and the ADTS header follows. Figure 3.2 shows the bitstream structure of ADTS. Element< FIL> may be formed for keeping the bit rate in the segment of the block only in the case of a constant bit rate.

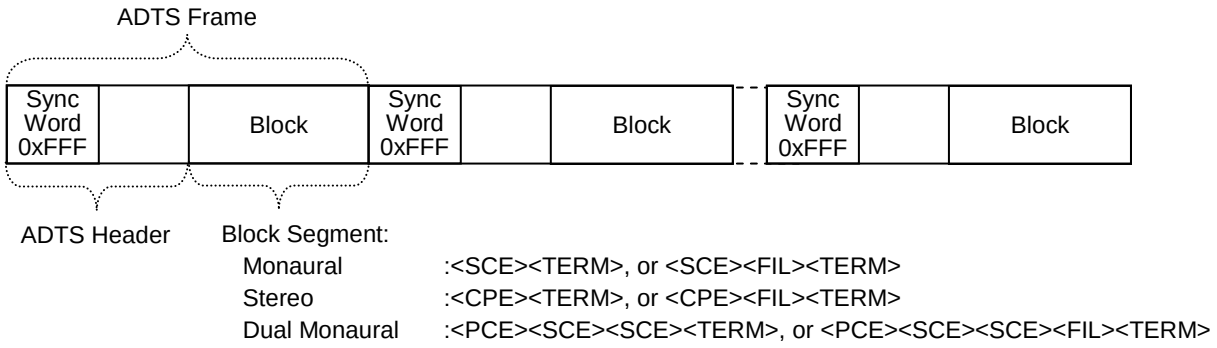


Figure 3.2 ADTS

3.2.2. ADIF

The bitstream structure of ADIF is as follows: ADIF header is aligned in the lead only, and without ADIF header blocks (raw_data_block) follow until the end. The ADIF header has one element <PCE (program_config_element)>. Figure 3.3 shows the bitstream structure of ADIF. Element< FIL> may be formed in the segment of the block only in the case of a constant bit rate.

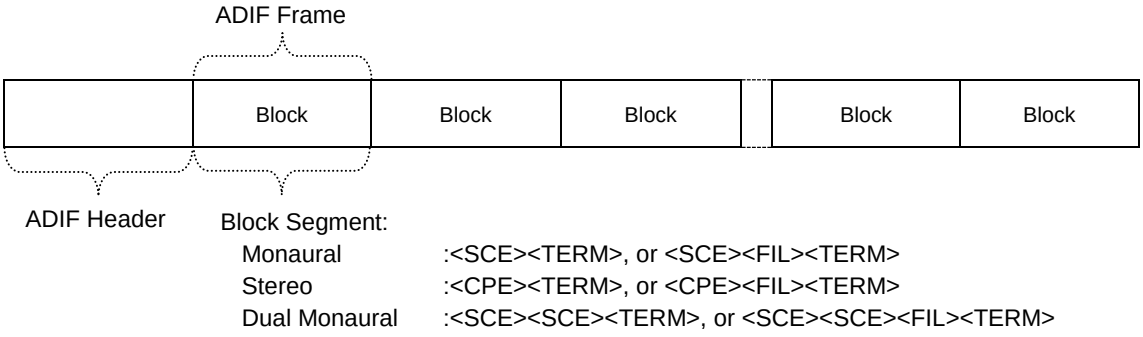


Figure 3.3 ADIF

3.2.3. raw data

The bitstream of raw data consists of blocks (raw_data_block) only. The structure is the same as ADIF except no ADIF header aligned as shown in Figure 3.3. In this form, the bitstream does not contain the number of channels and the header data of the sampling frequency. These data should be controlled with the application program.

3.3. Input Buffer (Input of 16-bit PCM Data)

The input buffer for this middleware shall be an array of data type short. This middleware does not support ring buffer for input. The input buffer size is needed more than 1024 samples. And also, the input buffer should be allocated on even byte boundary by application programs. If not, this middleware can't work correctly.

RSAACE_Encode function acquires PCM data for encoding one frame from the input buffer, according to the encode parameter values set to RSAACE_AACInfo structure. If PCM data is lack for one frame encoding in the input buffer, deficiency is complemented with silent data (0 data). To encode subsequently, PCM data for one frame or more should be provided to the input buffer before execution of RSAACE_Encode function, except for the end of the input PCM data.

3.4. Output Buffer (Output of Bitstream Data)

The output buffer for this middleware shall be an array of data type unsigned char. This middleware does not support ring buffer for output. The output buffer is needed to allocate by application programs.

The size of the output buffer is mandatory more than 1536 bytes(1536 bytes including one frame).

RSAACE_Encode function outputs the bitstream data in the top adderess of buffer specified by the argument.

4. Return Values and Error Recovery

4.1. Return Values and Status Codes of API Functions

The following macro definitions describe the return values and the status codes of the RSAACE_Init function and RSAACE_Encode function. The status codes are set to the member 'AAC_status' of the RSAACE_AAC structure.

<Definition>

```
// Return values
#define RSAACE_R_GOOD                    (  0)
#define RSAACE_C_CHECK                   (  1)
#define RSAACE_E_ERROR                   (  -1)

// Status codes
#define RSAACE_R_NO_ERROR                (  0)
#define RSAACE_C_END_OF_FILE             ( 100)
#define RSAACE_C_LACK_FRAME              ( 101)
#define RSAACE_C_PRE_ENCODE              ( 102)
#define RSAACE_C_BEFORE_END_OF_FILE      ( 103)
#define RSAACE_E_CHANNEL_MODE            (-100)
#define RSAACE_E_SAMPLE_RATE             (-101)
#define RSAACE_E_BIT_RATE                (-102)
#define RSAACE_E_STREAM_FORMAT           (-104)
#define RSAACE_E_ENABLE_CBR              (-105)
#define RSAACE_E_HOME                    (-107)
#define RSAACE_E_ORIGINAL_COPY           (-108)
#define RSAACE_E_COPY_ID_PRSNT           (-112)
#define RSAACE_E_GAP                     (-114)
#define RSAACE_E_MODE                    (-115)
#define RSAACE_E_NULL                    (-116)
```

4.1.1. Status Codes of RSAACE_Init Function

Table 4.1 lists the status codes of the RSAACE_Init function, their description, and reactions to them by an application program. When an error status code is set, make the application program set appropriate values to each member of the RSAACE_AACInfo structure with reference to Table 2.3.

Table 4.1 Status Codes of RSAACE_Init Function

Classifi-cation	Return Value	Status Code	Description	Processing by application program
Normal	RSAACE_R_GOOD	RSAACE_NO_ERROR	Initialization of work area succeeded.	The RSAACE_Encode function can be executed.
Error	RSAACE_E_ERROR	RSAACE_E_CHANNEL_MODE	The value set to channelMode is out of the specification.	As the RSAACE_Encode function can't be executed, with reference to Table 2.3, set appropriate values to each member of the RSAACE _AACInfo structure. And then, re-execute after specifing the correct.
		RSAACE_E_SAMPLE_RATE	The value set to sampleRate is out of the specification.	
		RSAACE_E_BIT_RATE	The value set to bitRate is out of the specification.	
		RSAACE_E_STREAM_FORMAT	The value set to outputFormat is out of the specification.	
		RSAACE_E_ENABLE_CBR	The value set to enable_cbr is out of the specification.	
		RSAACE_E_HOME	The value set to home is out of the specification.	
		RSAACE_E_ORIGINAL_COPY	The value set to original_copy is out of the specification.	
		RSAACE_E_COPY_ID_PRSNT	The value set to copyright_id_present is out of the specification.	
		RSAACE_E_GAP	The value set to gap is out of the specification.	
		RSAACE_E_MODE	The value set to mode is out of the specification.	
		RSAACE_E_NULL	Indicates that the NULL pointer was included in the parameter.	

[Note] If NULL is specified to the pointer to the RSAACE_AAC structure, this function returns RSAACE_E_ERROR, with no status code specified.

4.1.2. RSAACE_Encode Function Status Code

Table 4.2 lists the status codes of the RSAACE_Encode function, their description, and reactions to them by an application program.

Table 4.2 Status Codes of RSAACE_Encode Function

Classifi-cation	Return Value	Status Code	Description	Processing by application program
Normal	RSAACE_R_GOOD	RSAACE_R_NO_ERROR	Encoding of one frame successfully completed.	Bitstream data can be acquired from the output buffer.
Check	RSAACE_C_CHECK	RSAACE_C_END_OF_FILE	No PCM data exists in the input buffer. The encoding has been executed with being set zero (0) to the argument 'pcnt'. The output of the bitstream is the final frame.	Bitstream data can be acquired from the output buffer. To continue encoding, the RSAACE_Init function must be executed before this function being called again.
		RSAACE_C_LACK_FRAME	Encoding of one frame successfully completed. However, because the number of samples of PCM data is insufficient for one frame, silent data is complemented for encoding. It could be the last frame of the PCM data.	Bitstream data can be acquired from the output buffer. To continue encoding, set zero (0) to the argument 'pcnt' and re-execute this function if PCM data is the end, otherwise execute this function with supplying PCM data to the input buffer.
		RSAACE_C_PRE_ENCODE	When the value of "gap" in RSAACE_ AACInfo structure is set to" 0", the iput of first or second frame of PCM data just after RSAACE_Init function has been executed, and also when the value of "gap" equals "1", the iput of first frame has been executed.	Bitstream is not output to the output buffer. To continue encoding, re-execute this function with supplying PCM data to the input buffer.
		RSAACE_C_BEFORE_END_OF_FILE	No PCM data exists in the input buffer. The encoding has been executed with being set zero (0) to the argument 'pcnt'. The output of the bitstream is one frame before the last frame.	Bitstream data can be acquired from the output buffer. To continue encoding, set zero (0) to the argument 'pcnt' and execute this function.
Error	RSAACE_E_ERROR	RSAACE_E_NULL	Indicates that the NULL pointer was included in the parameter.	Re-execute again after specifing the correct to the argument.

[Note] If NULL is specified to the pointer to the RSAACE_AAC structure, this function returns RSAACE_E_ERROR, with no status code specified.

5. Processing Flows

Figure 5.1 shows an example of encoding with using the API functions. The application program with this middleware should determined whether it is the last frame of input data, whether the size of input data is one frame or more, and judgments of the return value and the status code.(*1)

(1) Set the encode parameter to RSAACE_AAInfo structure.

(2) Call RSAACE_Init function. (Initialization)

(3) Set PCM data and the number of samples.

(4) Call RSAACE_Encode function. (Encoding of one frame)

(5) Get the bitstream data.

(*1) Repeat steps (3) to (5). In step (3), set the last two frames as zero (0) at the end of PCM data, encode it in step (4), and after step (5), check the status code RSAACE_C_END_OF_FILE for completion.

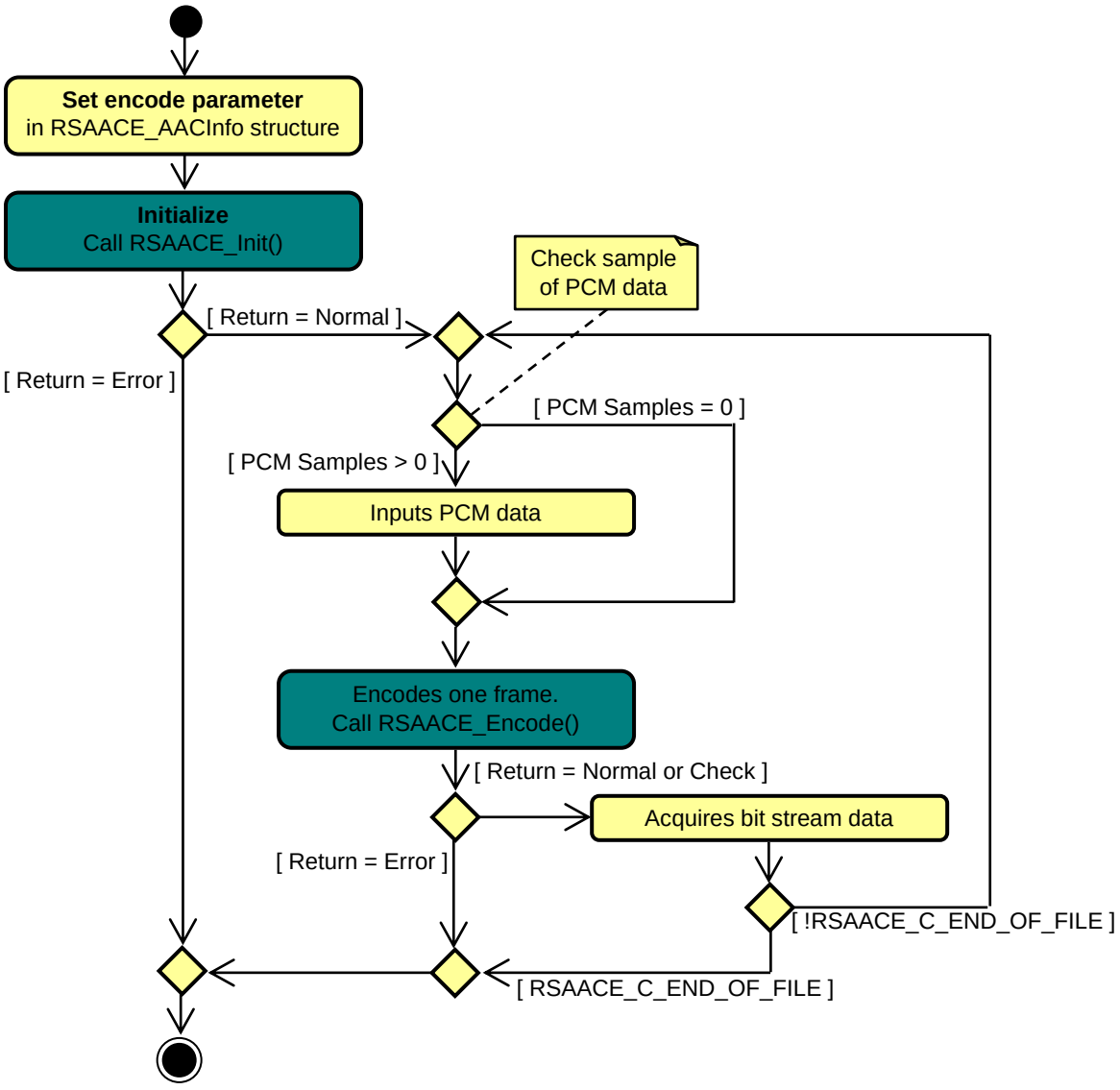


Figure 5.1 Encoding flow

6. Embedding Procedures

6.1. Product Configuration

This middleware is provided with CD-R. The files listed in Table 6.1 are stored in the CD-R.

Table 6.1 List of Files

No.	File name	Contents
1	RSAACE_AAC.h	Header file for API embedding
2	libRSAACELA_L.so.2.1	Dynamic link library
3	RTM0AC0000AEAACMZ1SL32J-01.pdf	User's Manual (Japanese)
4	RTM0AC0000AEAACMZ1SL32E-01.pdf	User's Manual (English)
5	RSAACE_sample.c	Sample program

6.2. Development Environment

To embed this middleware to the application program, use the development environment listed on Table 6.2 or any compatible environment.

Table 6.2 Development Environment

Items	Name/ Type name
OS	Linux kernel release 3.10
Compiler	cortexa15hf-vfp-neon-poky-linux-gnueabi gcc version 4.8.3 20140401 (prerelease) (Linaro GCC 4.8-2014.04)

6.3. Creating Application Program

Create the application program to call the API functions of this middleware. To call the API functions, include the header file provided with this middleware.

Table 6.3 lists the buffers and the structures which shall be allocated to the memory by the application program.

Table 6.3 Memory Areas Allocated by Application Program

	Name	Type	Size[bytes]	Remarks
1	Input buffer	short	4096	For Stereo
2	Output buffer	unsigned char	1536	
3	Stream information	RSAACE_AACInfo	52	
4	Work area	RSAACE_AAC	55948	

6.4. Including Standard Library

Include string.h as a standard library.

6.5. Linking this Middleware Library

Link this this middleware library in the appropriate to the user system.

6.6. Setting Compile Option

To embed this middleware, set the compiler options shown in Table 6.4. Use the default settings for the other options.

Table 6.4 Compile option

No	Compile option	Description
1	-fsigned-char	Set the char type to signed due to char is unsigned by default for gcc on ARM

7. Precautions

7.1. Reserved Words

The prefix "RSAACE_" is appended to the function and variable names and user open macro name of this middleware to distinguish from other middleware and application programs. To avoid competing, do not use symbols starting with "RSAACE_" in the application program using this middleware.

7.2. Reentrancy

This middleware guarantees reentrant. In the case of performing this middleware in plural task, each task needs each work region and buffers.

7.3. 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.

Revision History	ARM AAC Encode Middleware for Linux User's Manual

Rev.	Date	Description	
		Page	Summary
0.01	Aug. 20, 2014	-	First Edition issued
1.00	Aug. 22, 2014	-	How to Use This Manual Delete section 4. Notation of Numbers and Symbols
		19	Table 6.1 List of Files Change dynamic link library name.

ARM AAC Encode Middleware for Linux
RTM0AC0000AEAACMZ1SL32C
User's Manual

Publication Date: Aug. 20, 2014 Rev. 0.01
Aug. 22, 2014 Rev. 1.00

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

ARM AAC Encode Middleware for Linux

Renesas Electronics Corporation

RTM0AC0000AEAACMZ1SL32E-01