User's Manual

AAC Decode Middleware

User's Manual

Rev.1.00 December, 2014

Renesas Electronics
www.renesas.com

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.

How to Use This Manual

1. Purpose and Target Readers

This manual is intended to give users of the middleware an understanding of the decoder functionality, performance, and usage of the middleware. It is targeted at people who wish to design application systems which use the middleware. It assumes readers hold general knowledge in the fields of audio technology, programming languages, and microcontrollers.

This manual is broadly divided into the following sections:

-Product overview

-Middleware specifications

-Library function specifications

-Usage precautions

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 this product, you need to enter into a software license agreement with Renesas Electronics.

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/)

3. Related Documents

AAC-related Documents

Specifications Numbers and Titles	Date of issuance
ISO/IEC 13818-7:2006 Information technology Generic coding of moving pictures and associated audio information Part 7: Advanced Audio Coding (AAC)-Forth Edition	2006/01/15
ISO/IEC 14496-3:2005 Information Technology - Coding of Audio-Visual Objects - Part 3: Audio-Third Edition	2005/12/01

Processor-related Documents

Refer to attached product manual.

4. List of Abbreviations and Acronyms

Abbreviation	Full Form
AAC LC	AAC Low Complexity
ANSI-C	American National Standards Institute - C
bps	bits per second
CRC	Cyclic Redundancy Check
DAC	digital to analog converter
DRC	Dynamic Range Control
ETSI	European Telecommunications Standards Institute
High Quality SBR	High Quality Spectral Band Replication
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
LATM	Low overhead MPEG-4 audio transport multiplex
Low Power SBR	Low Power Spectral Band Replication
LSB	Least Significant Bit
MDCT	Modified Discrete Cosine Transform
MSB	Most Significant Bit
PCM	Pulse Code Modulation
PS	Parametric Stereo
QMF	Quadrature Mirror Filter
RAM	Random Access Memory
ROM	Read Only Memory
SBR	Spectral Band Replication

Table of Contents

1. Overview 1

1.1 Specifications Outline 1

1.2 Configuration 4

2. Middleware Specifications 6

2.1 Library Functions 6

2.2 Structures 6

2.3 Macro Definitions 7

2.3.1 Type Definitions 7

2.3.2 Common Symbols 7

2.4 Reserved Words 8

2.5 Processing Flow 9

3. Library Function Specifications 11

3.1 Function Specifications 11

3.1.1 aacd_GetMemorySize Function 12

3.1.2 aacd_Init Function 13

3.1.3 aacd_Decode Function 14

3.1.4 aacd_GetErrorFactor Function 15

3.1.5 aacd_GetVersion Function 16

3.2 Structure Specifications 18

3.2.1 Memory Size Acquisition Settings Structure 19

3.2.2 Memory Size Acquisition Results Structure 20

3.2.3 Work Memory Information Structure 21

3.2.4 Initialization Settings Structure 22

3.2.5 Decode Settings Structure 24

3.2.6 Decode Results Structure 25

3.2.7 Buffer Memory Settings Structure 28

3.2.8 Buffer Memory Results Structure 29

3.2.9 Element Information Structure 30

3.3 Error Processing 31

3.3.1 Error codes 31

3.3.2 Error Factors 32

3.4 Memory Specifications 34
3.4.1 Scratch Area 34
3.4.2 Static Area 34
3.4.3 Middleware Stack Area 35
3.4.4 Heap Area 35
3.4.5 Input Buffer 36
3.4.6 Output Buffer 40
3.4.7 Data Stream Buffer 44
3.5 Input Data 46
3.5.1 ADTS Format 46
3.5.2 RAW Format 49
3.6 Output Data 50

4. Precautions 51

4.1 Precautions in Calling a Function 51
4.1.1 Function Execution Timing 51
4.2 Other Precautions 52
4.2.1 Reserving and Allocating Memory Areas 52
4.2.2 Access Outside a Memory Range 52
4.2.3 Combination with Other Applications 52
4.2.4 Monitoring on the Performance 52

1. Overview

This section provides an overview of the AAC decoder.

1.1 Specifications Outline

AAC stands for Advanced Audio Coding. Standard ISO/IEC 13818-7 specifies MPEG-2 AAC. Standard ISO/IEC 14496-3 specifies MPEG-4 AAC.

AAC achieves an improved audio quality and an enhanced compression ratio by eliminating its compatibility with MP3 (MPEG Audio layer-3).

This middleware, which supports AAC decoding, decodes compressed input data and outputs decode results.

For basic specifications and performance, refer to attached product manual.

Table 1.1 Supported AAC Specifications

<table>
<tr><th>Item</th><th>Description</th></tr>
<tr><td>Input data format</td><td>MPEG-2 (AAC LC) : ISO/IEC 13818-7:2006(Fourth Edition)
MPEG-4 (AAC LC) : ISO/IEC 14496-3:2005(Third Edition)</td></tr>
<tr><td>Output data format</td><td>16/32-bit linear PCM</td></tr>
<tr><td>Sampling frequency (Hz) supported</td><td>8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100, 48000, 64000, 88200, 96000</td></tr>
<tr><td>Number of channels supported</td><td>Up to two channels</td></tr>
<tr><td>Bit rate (kbps) supported</td><td>Minimum bit rate for each channel: 8 kbps
Maximum bit rate for each channel:
<table>
<tr><th>Sampling frequency (Hz)</th><th>Max. bit rate (kbps)</th></tr>
<tr><td>8000</td><td>48</td></tr>
<tr><td>11025</td><td>66.15</td></tr>
<tr><td>12000</td><td>72</td></tr>
<tr><td>16000</td><td>96</td></tr>
<tr><td>22050</td><td>132.3</td></tr>
<tr><td>24000</td><td>144</td></tr>
<tr><td>32000</td><td>192</td></tr>
<tr><td>44100</td><td>264.6</td></tr>
<tr><td>48000</td><td>288</td></tr>
<tr><td>64000</td><td>384</td></tr>
<tr><td>88200</td><td>529.2</td></tr>
<tr><td>96000</td><td>576</td></tr>
</table></td></tr>
<tr><td>CRC</td><td>When the input data format is ADTS, CRC is performed for error compensation.</td></tr>
<tr><td>Reentrant</td><td>Supported</td></tr>
<tr><td>Restrictions</td><td>The program does not support the following:
- Gain control functionality
(Gain control is not defined for LC. Thus, if the stream contains gain control information, an error occurs.)
- CCE (Coupling Channel Element) decode processing
(If the stream contains CCE, an error occurs.)
The program does not handle the following data:
- Profiles other than LC
- Data in MPEG-4 container format
- One frame consisting of two or more raw data blocks in ADTS format
- Data with an emphasis (2 bits) added in ADTS format (according to the ISO/IEC 14496-3 initial specifications)
- Data for channels other than the front channel
- Data in more than two channels</td></tr>
</table>

Table 1.2 Memory Size Requirements

Memory type	Location	Memory area name		Size (in bytes)	
Instruction	ROM	Instruction area			
Data		Constant table area			---
		Other area(Depended on the compiler)			
	RAM	Middleware work area			65,896
		Area breakdown	Static area	Size breakdown	37,192
			Scratch area		28,680
		User work area			10,870
		Area breakdown	Input buffer	Size breakdown	1,536
			Output buffer		4,096
			Structure		630
		Stack area			2,048
		Other area(Depended on the compiler)			---

[Note] Area whose location is shown as ROM in the location column can be included in RAM or ROM.

[Note] Area whose location is shown as RAM in the location column can be included in RAM only.

[Note] For Instruction area and Constant table area refer to attached product manual.

1.2 Configuration

Figure1.1 shows an example of the decode system configuration which uses this middleware.

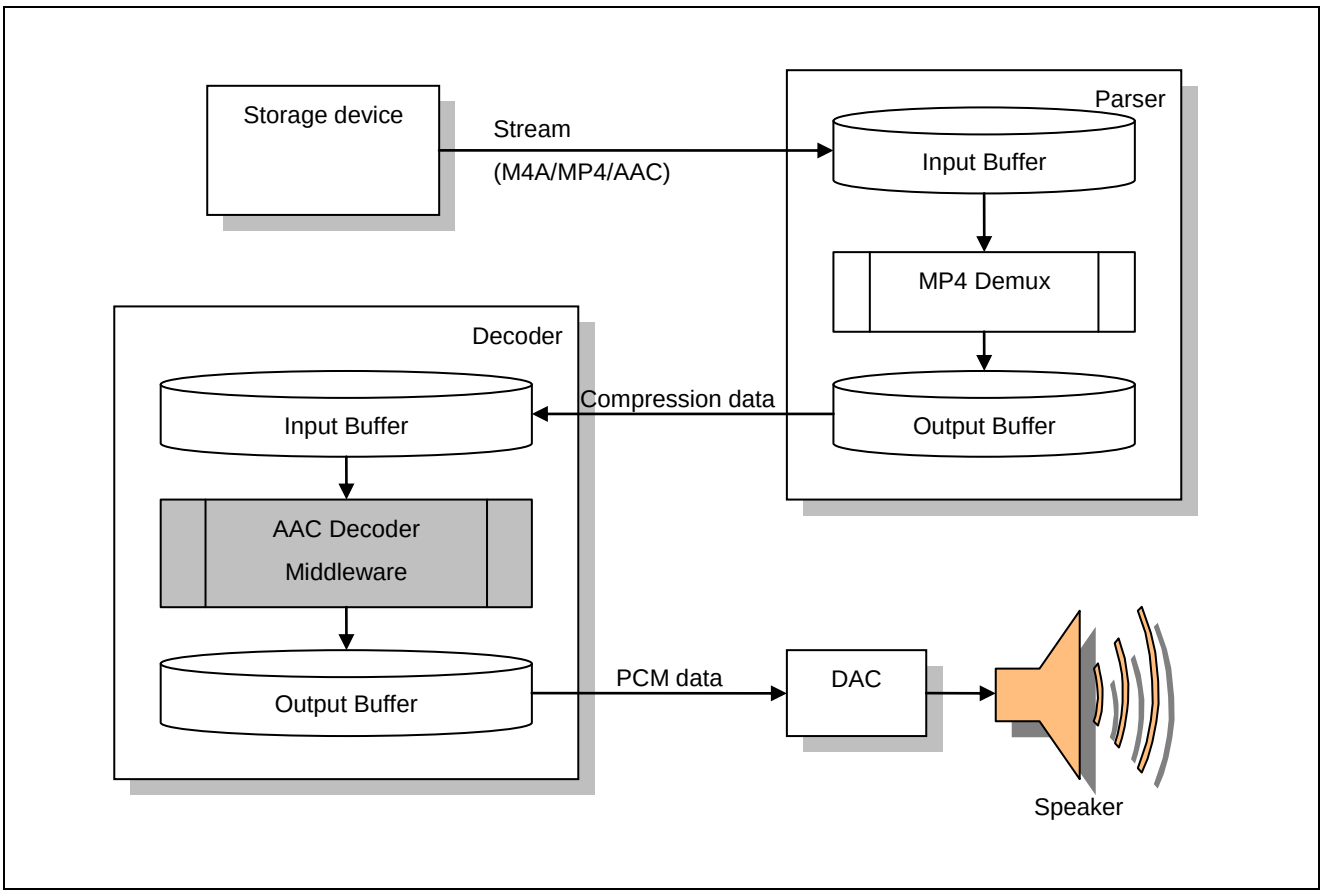

Figure1.1 Example of the Decode System Configuration

1. M4A/MP4/AAC

   Data used for various services such as delivering music.

2. Parser

   Extracts compressed data from M4A/MP4/AAC files. **The user should design the parser to suit the target system.**

3. Compression data

   AAC bit stream data in frames.

4. Decoder

   This middleware processes the data stored in the input buffer and outputs the processing results to the output buffer.

5. PCM data

   16/32-bit linear PCM data which is a decoding result generated by this middleware.

6. DAC

   The DAC converts 16/32-bit linear PCM data into an analog signal.

2. Middleware Specifications

2.1 Library Functions

Table 2.1 lists the functions offered by this middleware. For detailed specifications of these functions, refer to Section 3.1.

Table 2.1 Functions

Function name	Outline
aacd_GetMemorySize	Calculates the required memory size.
aacd_Init	Initializes the AAC decoder.
aacd_Decode	Decodes AAC frame data.
aacd_GetErrorFactor	Obtains an error factor.
aacd_GetVersion	Obtains version information.

2.2 Structures

Table 2.2 lists the structures for this middleware. The user should reserve areas required for these structures. For detailed specifications of these structures, refer to Section 3.2.

Table 2.2 Structures

Structure name	Outline	I/O
Memory size acquisition settings structure	Stores the parameters necessary for memory size acquisition.	I
Memory size acquisition results structure	Stores the acquired memory sizes.	O
Work memory information structure	Stores the parameters related to work memory.	I
Initialization settings structure	Stores the parameters necessary for initialization.	I
Decode settings structure	Stores the parameters necessary for decoding.	I
Decode results structure	Stores the decoding results.	O
Buffer memory settings structure	Stores the parameters related to the input/output buffer.	I
Buffer memory results structure	Stores the processing results related to the input/output buffer.	O
Element information structure	Stores PCE element information.	I

2.3 Macro Definitions

2.3.1 Type Definitions

Table 2.3 lists the type definitions available in this middleware.

Table 2.3 Type Definitions

Type	Size in bytes	Description	
ACMW_INT8	1	8-bit signed integer	-128 to 127
ACMW_INT16	2	16-bit signed integer	-32768 to 32767
ACMW_INT32	4	32-bit signed integer	-2147483648 to 2147483647
ACMW_UINT8	1	8-bit unsigned integer	0 to 255
ACMW_UINT16	2	16-bit unsigned integer	0 to 65535
ACMW_UINT32	4	32-bit unsigned integer	0 to 4294967295
ACMW_BOOL	2	Boolean value (16-bit signed integer)	Zero (false)/Non-zero (true)

[Note] All the pointers have the same size (4 bytes).

2.3.2 Common Symbols

Table 2.4 lists the symbol definitions available in this middleware.

Table 2.4 Common Symbols

Common symbol	Definition	Description
AACD_RESULT_OK	0x00000000	Processing results are normal.
AACD_RESULT_NG	0x00000001	Processing results are abnormal.
AACD_RESULT_WARNING	0x00000002	Abnormality has occurred, which does not prevent the process from continuing.
AACD_RESULT_FATAL	0x00000003	Abnormality has occurred, which prevents the process from continuing.

2.4 Reserved Words

Table 2.5 lists the naming rules for the symbols available in this middleware.

When you use this middleware together with other applications, be careful to avoid the duplication of symbol names.

Table 2.5 Naming Rules for Symbols

Classification	Outline
Function names	aacd_XXXX
Structure names	aacd_XXXX
Return values from functions	AACD_RESULT_XXXX [Note] XXXX consists only of upper-case letters.
Error factor names	AACD_ERR_XXXX [Note] XXXX consists only of upper-case letters.
Basic type prefix names	ACMW_XXXX [Note] XXXX consists only of upper-case letters.
Other prefix names	AACD_XXXX [Note] XXXX consists only of upper-case letters.

[Note] XXXX can be any alphanumeric string.

2.5 Processing Flow

Figure 2.1 shows a flow diagram of processing performed by an application which uses this middleware.

The steps executed by the functions of this middleware are shaded. The steps defined by the user are white. Design the process to suit the target system.

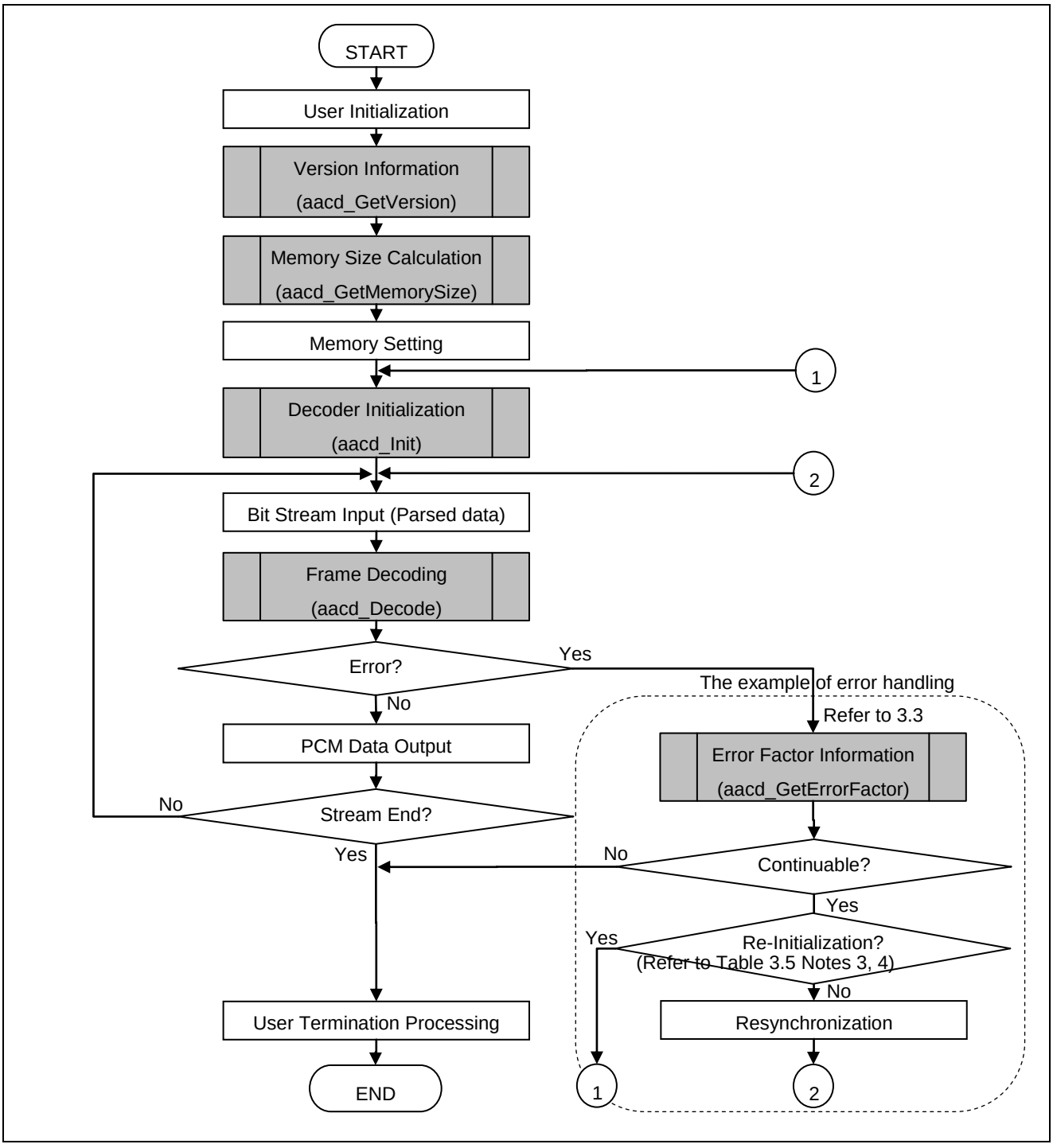

Figure 2.1 Example of the Application Processing Flow

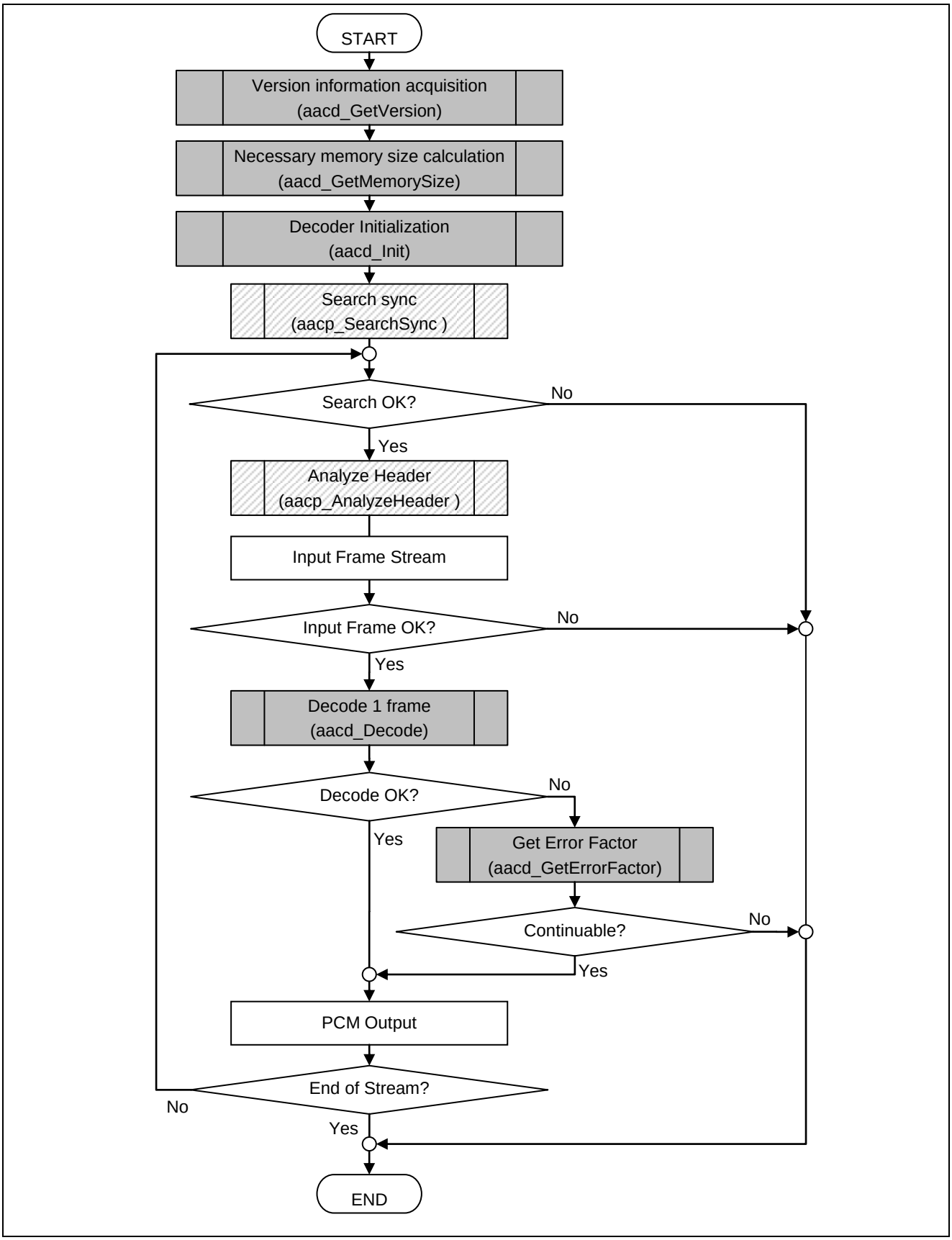

Figure 2.2 Example of the sample program and parser flow

3. Library Function Specifications

3.1 Function Specifications

The next sections describe this middleware's functions by using the description format below.

Synopsis	Outlines the function.		
Syntax	Describes the syntax for calling the function.		
Function	Describes what the function does.		
Arguments		I/O	Describes the arguments for the function.
Return value	Type name		Describes the return values from the function.
Description	Provides information such as precautions in using the function.		

[Note] This syntax format complies with ANSI-C. It does not use to standard C libraries of functions with C language standard other than the mathematical functions.

3.1.1 aacd_GetMemorySize Function

Synopsis	Calculates the necessary memory size.		
Syntax	ACMW_INT32 aacd_GetMemorySize(const aacd_getMemorySizeConfigInfo * const pGetMemorySizeConfigInfo, aacd_getMemorySizeStatusInfo * const pGetMemorySizeStatusInfo);		
Function	This function calculates the necessary memory size for the static area, scratch area, and input/output buffer which are used by this middleware. Then, it stores the calculation results into the memory size acquisition results structure.		
Arguments		I/O	Meaning
aacd_getMemorySizeConfigInfo *pGetMemorySizeConfigInfo		I	Memory size acquisition settings structure
aacd_getMemorySizeStatusInfo *pGetMemorySizeStatusInfo		O	Memory size acquisition results structure
Return value	ACMW_INT32 errorCode		Error code For details, refer to Table 3.3.
Description	Execute this function before aacd_Init function and then reserve the required amount of memory space. You can execute this function at any time because it does not need to be initialized.		

3.1.2 aacd_Init Function

Synopsis	Initializes the AAC decoder.		
Syntax	ACMW_INT32 aacd_Init(const aacd_workMemoryInfo * const pWorkMemInfo, const aacd_initConfigInfo * const pInitConfigInfo);		
Function	This function initializes the static and scratch areas which are used by this middleware. Also, it sets various parameters.		
Arguments		I/O	Meaning
aacd_workMemoryInfo *pWorkMemInfo		I	Work memory information structure
aacd_initConfigInfo *pInitConfigInfo		I	Initialization settings structure
Return value	ACMW_INT32 errorCode		Error code For details, refer to Table 3.3
Description	Execute this function only once before starting a series of decode process steps. They make up a process from the start to the end of decoding a certain stream.		

3.1.3 aacd_Decode Function

Synopsis	Decodes data.		
Syntax	ACMW_INT32 aacd_Decode(const aacd_workMemoryInfo * const pWorkMemInfo, const aacd_decConfigInfo * const pDecConfigInfo, const aacd_ioBufferConfigInfo * const pBuffConfigInfo, aacd_decStatusInfo * const pDecStatusInfo, aacd_ioBufferStatusInfo * const pBuffStatusInfo);		
Function	This function decodes AAC LC frame data.		
Arguments		I/O	Meaning
aacd_workMemoryInfo *pWorkMemInfo		I	Work memory information structure
aacd_decConfigInfo *pDecConfigInfo		I	Decode settings structure
aacd_ioBufferConfigInfo *pBuffConfigInfo		I	Buffer memory settings structure
aacd_decStatusInfo *pDecStatusInfo		O	Decode results structure
aacd_ioBufferStatusInfo *pBuffStatusInfo		O	Buffer memory results structure
Return value	ACMW_INT32 errorCode		Error code For details, refer to Table 3.3.
Description	Execute this function when you decode one frame of stream data. [Note] If an error occurs, the members of the decode results structure are indefinite. Do not reference these members.		

3.1.4 aacd_GetErrorFactor Function

Synopsis	Obtains error factors.		
Syntax	ACMW_UINT32 aacd_GetErrorFactor(const aacd_workMemoryInfo * const pWorkMemInfo);		
Function	This function returns any error factors related to the most recently executed aacd_Init and aacd_Decode functions.		
Arguments		I/O	Meaning
aacd_workMemoryInfo *pWorkMemInfo		I	Work memory information structure
Return value	ACMW_UINT32 errorFactor		Value indicating an error factor For details, refer to Table 3.5.
Description	This function returns any error factors related to the most recently executed aacd_Init and aacd_Decode functions. It cannot return error factors related to any other functions. Nor can it return error factors for any errors that have occurred before the aacd_Init function initializes the static area. The error factors are overwritten next time you execute the aacd_Init and aacd_Decode functions. So, if you need the error factors, execute this function before reexecuting the functions above.		

3.1.5 aacd_GetVersion Function

Synopsis	Obtains version information.		
Syntax	ACMW_UINT32 aacd_GetVersion(void);		
Function	This function returns the version number of this middleware.		
Arguments		I/O	Meaning
None		-	-
Return value	ACMW_UINT32 versionCode		Version information Example: If the return code is 0x00000123, the version number is 1.23. For details, refer to Table 3.1 .
Description	This function obtains the version number of this middleware. You can execute this function at any time.		

Table 3.1 versionCode Settings

Setting	Value	Description
Customer ID (8bit)	0x00	Standard version
	Others	Reserved
Release ID (8bit)	0x00	Authorized version
	0xA0 to 0xAF	alpha version (restricted in functionality) 0xA1 : alpha 1 ... 0xA9 : alpha 9 Other: Reserved
	0xB0 to 0xBF	beta version (not restricted in functionality, but not completely tested) 0xB1 : beta 1 ... 0xB9 : beta 9 Other: Reserved
	Others	Reserved
Major ID (8bit)	0xXY	Version XY.xy (major number) When X=0 to 9 and Y=0 to 9: 0x00 : Version 0.xy ... 0x10 : Version10.xy ... 0x99 : Version99.xy Other: Reserved
Minor ID (8bit)	0xXY	Version xy.XY (minor number) When X=0 to 9 and Y=0 to 9: 0x00 : Version xy.00 ... 0x10 : Version xy.10 ... 0x99 : Version xy.99 Other: Reserved

3.2 Structure Specifications

The next sections describe this middleware's structures by using the description format below.

[Structure name]	Name of the structure
[Function]	Describes what the structure does.
[Prototype]	Prototype of the structure
[Member description]	Describes the members of the structure.
[Remarks]	Provides information such as precautions in using the structure.

3.2.1 Memory Size Acquisition Settings Structure

[Structure name] aacd_getMemorySizeConfigInfo

[Function] This structure specifies the conditions for calculating the necessary memory size when aacd_GetMemorySize function obtains that size.

[Prototype]

```
typedef struct {
    ACMW_BOOL    bSbrDisableFlag;
    ACMW_BOOL    bPsDisableFlag;
    ACMW_BOOL    bDrcEnableFlag;
}aacd_getMemorySizeConfigInfo;
```

[Member description]

Member Variable Name	Description	
bSbrDisableFlag	SBR disable flag*1	
	0	SBR ON
	other	SBR Forced OFF
bPsDisableFlag	PS disable flag*1	
	0	PS ON
	other	PS Forced OFF
bDrcEnableFlag	DRC enable flag	
	0	DRC OFF
	other	DRC ON

[Remarks] The user should reserve the necessary areas. The user should reserve the areas and set the values before calling the aacd_GetMemorySize function.
*1: Invalid parameter for this middleware. (It is invalid because this middleware does not support HE-AAC decoding).

3.2.2 Memory Size Acquisition Results Structure

[Structure name] aacd_getMemorySizeStatusInfo

[Function] This structure stores the necessary-memory-size calculation results by using the necessary-memory-size calculation process (aacd_GetMemorySize function).

[Prototype]
```
typedef struct {
    ACMW_UINT32     nStaticSize;
    ACMW_UINT32     nScratchSize;
    ACMW_UINT32     nInputBufferSize;
    ACMW_UINT32     nOutputBufferSize;
    ACMW_UINT32     nStackSize;
}aacd_getMemorySizeStatusInfo;
```

[Member description]

Member Variable Name	Description
nStaticSize	Necessary memory size (in bytes) of the static area
nScratchSize	Necessary memory size (in bytes) of the scratch area
nInputBufferSize	Necessary memory size (in bytes) of the input buffer area
nOutputBufferSize	Necessary memory size (in bytes) of the output buffer area *1
nStackSize	Necessary memory size (in bytes) of the middleware stack area

[Remarks] The user should reserve the necessary areas before calling the aacd_GetMemorySize function.

[Remarks2] *1: Output buffer area size for one channel is returned. If bOutBitsPerSample of the initialization settings structure (Section 3.2.4) is set to 1, the actually required output buffer area size is twice the return value.

3.2.3 Work Memory Information Structure

[Structure name] aacd_workMemoryInfo

[Function] This structure specifies the addresses in the work memory used by this middleware.

[Prototype]
```
typedef struct {
    void *      pStatic;
    void *      pScratch;
}aacd_workMemoryInfo;
```

[Member description]

Member Variable Name	Description
pStatic	Pointer to the beginning of the static area.
pScratch	Pointer to the beginning of the scratch area.

[Remarks] The user should reserve the necessary areas. The user should reserve the areas and set the values before calling the library function which requires this structure as the arguments.

[Remarks2] You can obtain the sizes of the static and scratch areas by using the necessary-memory-size calculation process (aacd_GetMemorySize function).

3.2.4 Initialization Settings Structure

[Structure name] aacd_initConfigInfo

[Function] This structure specifies the decoding conditions for the initialization process (aacd_Init function).

[Prototype]

```
typedef struct {
    ACMW_BOOL   bSbrDisableFlag;
    ACMW_BOOL   bPsDisableFlag;
    ACMW_BOOL   bDownSampleSBRFlag;
    ACMW_BOOL   bDrcEnableFlag;
    ACMW_UINT16 nTargetRefLevel;
    ACMW_UINT16 nDefaultProgRefLevel;
    ACMW_UINT32 nCompress;
    ACMW_UINT32 nBoost;
    ACMW_BOOL   bOutputChMapType;
    ACMW_BOOL   bDisablePCEFlag;
    ACMW_BOOL   bOutBitsPerSample;
    ACMW_UINT16 nFormatType;
}aacd_initConfigInfo;
```

[Member description]

Member Variable Name	Description	
bSbrDisableFlag	SBR disable flag[*1]	
	0	SBR ON
	other	SBR OFF
bPsDisableFlag	PS disable flag[*1]	
	0	PS ON
	other	PS OFF
bDownSampleSBRFlag	Downsample SBR flag[*1]	
	0	Normal SBR mode
	other	Downsample SBR mode
bDrcEnableFlag	DRC enable flag	
	0	DRC OFF
	other	DRC ON
nTargetRefLevel	Target reference level	
	0 to 127	Correspond to 0 (dB) to 31.75 (dB). (0.25 dB/step)
	other	Internally clipped to 127.
nDefaultProgRefLevel	Program reference level	
	0 to 127	Correspond to 0 (dB) to 31.75 (dB). (0.25 dB/step)
	other	Internally clipped to 127.
nCompress	Compress factor	
	0 to 0x7fffff	Correspond to 0.0 to 1.0.
	other	Internally clipped to 0x7fffff.
nBoost	Boost factor	
	0 to 0x7fffff	Correspond to 0.0 to 1.0.
	other	Internally clipped to 0x7fffff.
bOutputChMapType	Output PCM channel mapping format[*2]	
	0	Outputs the decode results without modifying them.
	other	Outputs the decode results by mapping them to 5.1 ch. Outputs L, R, C, LFE, Ls, and Rs in this order.
bDisablePCEFlag	PCE reading skip flag[*3]	
	0	Uses PCE if present.
	other	Skips PCE reading.
bOutBitsPerSample	Number of bits per output PCM sample	
	0	16-bit PCM output
	other	32-bit PCM output
nFormatType	Input bit stream format	
	0x0000	ADTS
	0x0001	RAW DATA
	other	Reserved [*4]

[Remarks] The user should reserve the necessary areas. The user should reserve the areas and set the values before calling the aacd_Init function.

[Remarks2]

*1: Invalid parameter for this middleware. (It is invalid because this middleware does not support HE-AAC decoding.)
*2: Invalid parameter for this middleware. (It is invalid because this middleware does not support 6-ch decoding.)
*3: Only valid for PCE included in raw data in the input stream.
PCE, if set in the decode settings structure, becomes valid regardless of this flag.
*4: This middleware operates assuming that raw data is set.

3.2.5 Decode Settings Structure

[Structure name] aacd_decConfigInfo

[Function] This structure specifies the processing conditions when the aacd_Decode function decodes data.

[Prototype]
```
typedef struct {
    ACMW_UINT32             nExtSamplingRate;
    ACMW_BOOL               bSetPceFlag;
    ACMW_aacd_elementInfo   sFrontElement;
    ACMW_aacd_elementInfo   sSideElement;
    ACMW_aacd_elementInfo   sBackElement;
    ACMW_aacd_elementInfo   sLfeElement;
}aacd_decConfigInfo;
```

[Member description]

Member Variable Name	Description	
nExtSamplingRate	Sampling frequency (Hz) *1	
bSetPceFlag	PCE data setting flag*2	
	0x0000	Does not set PCE data.
	other	Sets PCE data.
sFrontElement	Front channel element information structure (see Section 3.2.9)*3	
sSideElement	Side channel element information structure (see Section 3.2.9)*4	
sBackElement	Back channel element information structure (see Section 3.2.9)*4	
sLfeElement	LFE channel element information structure (see Section 3.2.9)*4	

[Remarks] The user should reserve the necessary areas. The user should reserve the areas and set the values before calling the aacd_Decode function.

[Remarks2]

*1: Set one of the frequencies (Hz) listed in Table3.17.
If you set any other value, it is corrected according to ISO/IEC 14496-3: 4.5.1.1 Table 4.82, and the corrected frequency is used for operation.
If you specify 0, an error (AACD_ERR_PARAMETER) occurs.

*2: Once you set PCE data, it is internally held until PCE data is set again or until PCE of raw data portion in the stream is detected.

*3: Each element information structure is valid only when the PCE data setting flag is configured.

*4: Invalid parameter for this middleware. (It is invalid because this middleware does not support 6-ch decoding.)

3.2.6 Decode Results Structure

[Structure name] aacd_decStatusInfo

[Function] This structure stores the results of decoding performed by the aacd_Decode function.

[Prototype]

```
typedef struct {
    ACMW_UINT32     nSamplingRate;
    ACMW_UINT32     nOutSamplingRate;
    ACMW_UINT16     nOutSamplesPerFrame;
    ACMW_UINT16     nChannelNum;
    ACMW_UINT16     nOutChannelNum;
    ACMW_UINT16     nChannelConfig;
    ACMW_UINT16     nOutChannelConfig;
    ACMW_BOOL       bDualChannelMode;
    ACMW_UINT16     nDualChannelTag;
    ACMW_BOOL       bCrcEnable;
    ACMW_BOOL       bSbrFound;
    ACMW_BOOL       bPsFound;
    ACMW_BOOL       bPceFound;
    ACMW_BOOL       bMpegDmxCoefPresent;
    ACMW_UINT16     nMpegDmxCoef;
    ACMW_BOOL       bPseudoSurroundEnable;
    ACMW_UINT16     nDseNum;
    ACMW_UINT16     aDseTag[16];
}aacd_decStatusInfo;
```

[Member description]

Member Variable Name	Description	
nSamplingRate	Sampling frequency (Hz) (AAC LC)	
nOutSamplingRate	Output sampling frequency (Hz) (for the last output)	
nOutSamplesPerFrame	Number of samples per channel or frame (for the last output)	
nChannelNum	Number of channels (AAC LC)	
nOutChannelNum	Number of channels (for the last output)	
nChannelConfig	Channel configuration information (see Table3.2)*1	
nOutChannelConfig	Output channel configuration information*2. The bit corresponding to the output channel position is set to 1.	
bDualChannelMode	Specifies whether the stream is in dualMono format.	
	0	No
	1	Yes
nDualChannelTag	element_instance_tag information for dualMono (see Figure 3.1)	
bCrcEnable	Specifies whether there is CRC in the stream.	
	0	No
	1	Yes
bSbrFound	Specifies whether there is SBR to be decoded*3	
	0	No
	1	Yes
bPsFound	Specifies whether there is PS to be decoded.	
	0	No
	1	Yes
bPceFound	Specifies whether there is PCE in the stream.	
	0	No
	1	Yes
bMpegDmxCoefPresent	matrix_mixdown_idx_present parameter*4	
nMpegDmxCoef	matrix_mixdown_idx parameter*4	
bPseudoSurroundEnable	pseudo_surround_enable parameter*4	
nDseNum	Number of DSEs stored in the data stream buffer	
aDseTag[16]	element_instance_tag value of DSE stored in each data stream buffer	

[Remarks] The user should reserve the necessary areas. The user should reserve the areas and set the values before calling the aacd_Decode function.

[Remarks2]

*1: The value of channel_configuration as ADTS header information is output. This value is always 0 if raw data is specified for nFormatType in the Initialization Settings Structure.
*2: Invalid parameter for this middleware. 0 is always returned. (The parameter is invalid because this middleware does not support 6-ch decoding.)
*3: Invalid parameter for this middleware. 0 is always returned. (The parameter is invalid because this middleware does not support HE-AAC decoding.)
*4: Refer to ISO/IEC 14496-3: 4.4.1.1 Program config element. This parameter becomes valid when bPceFound is 1. After that, it holds the last information acquired which is included in PCE.

Table3.2 Channel Configuration Information

nChannelConfig	Channel configuration
0x0000	(Defined in PCE)
0x0001	1
0x0002	2

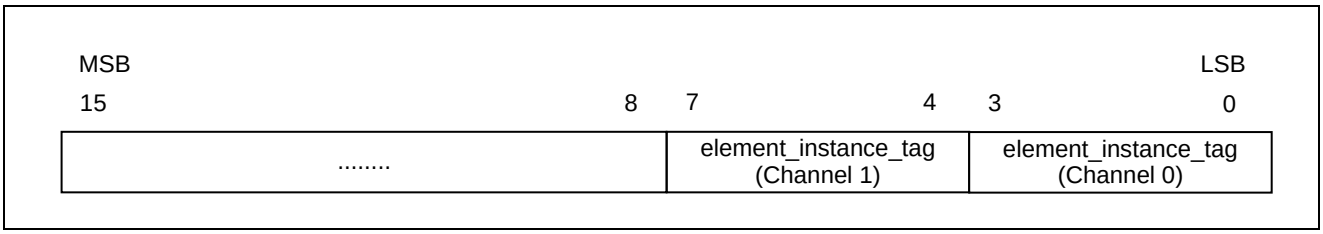

Figure 3.1 element_instance_tag Information for dual Mono

[Note]Channel 0 corresponds to nOutBuffUsedDataSize [0] in the buffer memory results structure. Channel 1 corresponds to nOutBuffUsedDataSize [1] in that structure.

3.2.7 Buffer Memory Settings Structure

[Structure name] aacd_ioBufferConfigInfo

[Function] Stores the parameters related to the input/output buffer.

[Prototype]
```
typedef struct {
    ACMW_UINT8 *    pInBuffStart;
    ACMW_UINT32     nInBuffOffsetBits;
    ACMW_UINT32     nInBuffSetDataSize;
    void **         pOutBuffStart;
    ACMW_UINT32     nOutBuffSize;
    ACMW_UINT8      nDseBuffNum;
    ACMW_UINT8 **   pDseBuffStart;
    ACMW_UINT16     nDseBuffSize;
}aacd_ioBufferConfigInfo;
```

[Member description]

Member Variable Name	Description
pInBuffStart	Start address of input data
nInBuffOffsetBits	Offset for the input start bit*1. (A value from 0 to 7)
nInBuffSetDataSize	Input data size (in bytes)
pOutBuffStart	Start address of output data for each channel
nOutBuffSize	Size (in bytes) of each output buffer*2
nDseBuffNum	Number of data stream buffers*3. (A value from 0 to 16)
pDseBuffStart	Start address of each data stream buffer
nDseBuffSize	Size (in bytes) of each data stream buffer*4

[Remarks] The user should reserve the necessary areas. The user should reserve the areas and set the values before calling the aacd_Decode function.
For information about the input buffer, refer to Section 3.4.5. For information about the output buffer, refer to Section 3.4.6.

[Remarks2]
*1: If a value outside the range is specified, it is assumed to be 0.
*2: The output buffer size is the same for all channels.
*3: Specify 0 if you do not use a data stream buffer. If a value outside the range is specified, it is assumed to be 16.
*4: The data stream buffer size is the same for all channels.

3.2.8 Buffer Memory Results Structure

[Structure name] aacd_ioBufferStatusInfo

[Function] Stores the processing results related to the input/output buffer.

[Prototype]
```
typedef struct {
    ACMW_UINT8 *    pInBuffLast;
    ACMW_UINT32     nInBuffUsedDataSize;
    void **         pOutBuffLast;
    ACMW_UINT32     nOutBuffUsedDataSize;
    ACMW_UINT16     aDseDataSize[16];
}aacd_ioBufferStatusInfo;
```

[Member description]

Member Variable Name	Description
pInBuffLast	Post-read address of an input buffer*1
nInBuffUsedDataSize	Consumed data size (in bytes) of an input buffer *2
pOutBuffLast	Post-write address of output data for each channel
nOutBuffUsedDataSize	Consumed data size (in bytes) of an output buffer for each channel*3
aDseDataSize	Size (in bytes) of data stored in each data stream buffer

[Remarks] The user should reserve the necessary areas. The user should reserve the areas and set the values before calling the aacd_Decode function.

[Remarks2]
*1: The post-read address indicates the next start address.
*2: If nInBuffOffsetBits in the buffer memory settings structure is not 0, the consumed data size (in bytes) of an input buffer is equal to the sum of the actual number of consumed data bits and the nInBuffOffsetBits value.
*3: The consumed data size of an output buffer is the same for all channels.

3.2.9 Element Information Structure

[Structure name] aacd_elementInfo

[Function] This structure contains element information used by the aacd_Decode function.

[Prototype]
```
typedef struct {
    ACMW_UINT32     nElement;
    ACMW_BOOL       aEleIsCpe[16];
    ACMW_UINT32     aEleTag[16];
}aacd_elementInfo;
```

[Member description]

Member Variable Name	Description
nElement	num_front_channel_elements*1
aEleIsCpe[16]	front_element_is_cpe*1
aEleTag[16]	front_element_tag_select*1

[Remarks] The members of this structure are the same as those of the decode results structure described in Section 3.2.6. Thus, the element information structure is reserved when the decode results structure is reserved. The user should reserve the decode results structure before calling the aacd_Decode function.

[Remarks2] *1: Refer to ISO/IEC 14496-3: 4.4.1.1 Program config element.

3.3 Error Processing

This middleware's functions return the error codes listed in Table 3.3. To obtain details about the cause of an error, execute the aacd_GetErrorFactor function.

3.3.1 Error codes

Below are the error codes for this middleware.

Table 3.3 Error Codes

Error code (32bit)	Value	Description	Reinitialization
[1]AACD_RESULT_OK	0x00000000	The processing results are normal. The process has terminated normally.	Unnecessary
[2]AACD_RESULT_NG	0x00000001	The processing results are abnormal. An invalid parameter is specified in the structure. Or else, the program execution is incorrect. PCM data is not output. Specify the valid parameter in the structure or reexecute the program by using the correct procedure.	Unnecessary
[3]AACD_RESULT_WARNING	0x00000002	Abnormality has occurred, which does not disable process continuation. The decoder detected an error, but PCM data was output. At this time, the error concealment or MUTE signal (all 0's) might have been output. Check the error by using the error factor acquisition process (aacd_GetErrorFactor function).	Unnecessary
[4]AACD_RESULT_FATAL	0x00000003	Abnormality has occurred, which disables process continuation. The process cannot continue. PCM data is not output. Reinitialization the program. An error factor cannot be obtained through the aacd_GetErrorFactor function.	Necessary
[5]Others	Other than the above	Reserved	-

Table 3.4 Error Codes Used by the Library Functions

Function / Error code	aacd_GetMemorySize	aacd_Init	aacd_Decode	aacd_GetErrorFactor*1	aacd_GetVersion*2
AACD_RESULT_OK	o	o	o	-	-
AACD_RESULT_NG	-	o	o	-	-
AACD_RESULT_WARNING	-	-	o	-	-
AACD_RESULT_FATAL	o	o	o	o	-

[Note] o : Error code might be output. - : Error code is not used.

[Note] *1 : Returns an error factor.

*2 : Returns version information.

3.3.2 Error Factors

An error factor provides details about an error which has occurred. You can obtain error factors with the aacd_GetErrorFactor function except when AACD_RESULT_FATAL occurs. Table 3.5 lists the error factors. Table 3.6 shows the relationship of the library functions to the error factors and error codes.

Table 3.5 Error Factors

errorFactor (32 bits)	Value	Description	Table 3.3	PCM
AACD_ERR_NONE	0x00000000	The program has normally terminated. No error factor is available.	[1]	Normal*1
AACD_ERR_POINTER	0x00000010	Invalid pointer value	[2]	Unavailable
AACD_ERR_PARAMETER	0x00000020	Invalid parameter	[1]	Unavailable
AACD_ERR_SEQUENCE	0x00000040	This and other library functions were executed in invalid order.*3	[2]	Unavailable
AACD_ERR_INPUT_DATA_SIZE	0x00000100	The input buffer was empty.	[2]	Unavailable
AACD_ERR_STREAM	0x00001000	Abnormal raw data in the input stream was detected.	[2]	Unavailable
AACD_ERR_HEADER	0x00010000	The ADTS header's syncword was not detected. Or else, it was invalid.	[2]	Unavailable
AACD_ERR_CHANGED_FS	0x00020000	The sampling frequency is different from the one for a previous frame.*4	[2]	Unavailable
AACD_ERR_CHANGED_CH	0x00040000	The channel count is different from the one for a previous frame.*4	[1]	Unavailable
AACD_ERR_PROGRAM_CONFIG	0x00080000	A ProgramConfig error was detected.	[2]	Unavailable
AACD_ERR_1ST2ND_FRAME	0x00100000	The first two frames have been output.	[3]	Available
AACD_ERR_CRC	0x00200000	A CRC error has occurred.	[2]	Unavailable
AACD_ERR_DECODE	0x01000000	An error has occurred during decode processing.	[3]	Available*2
Others	Other than the above	Reserved	-	-

[Note] *1 : Error determination may be impossible depending on which part of data is damaged. In this case, the error factor is assumed to be AACD_ERR_NONE, the decode processing continues, and the decoding results are output.

*2 : PCM data generation may be prevented during the first half of the decode processing depending on which part of data is damaged.

*3 : Reinitialization is necessary because the execution sequence is invalid.

*4 : If you use the aacd_Init function to perform initialization for recovery, you can start decoding from the same position in the input stream it previously started from.

Table 3.6 Relationship of the Library Functions to the Error Factors

Function / Error factor	aacd_GetMemorySize	aacd_Init	aacd_Decode	aacd_GetErrorFactor*1	aacd_GetVersion
AACD_ERR_NONE	-	o	o	-	-
AACD_ERR_POINTER	-	o	o	-	-
AACD_ERR_PARAMETER	-	-	o	-	-
AACD_ERR_SEQUENCE	-	-	o	o	-
AACD_ERR_INPUT_DATA_SIZE	-	-	o	-	-
AACD_ERR_STREAM	-	-	o	-	-
AACD_ERR_HEADER	-	-	o	-	-
AACD_ERR_CHANGED_FS	-	-	o	-	-
AACD_ERR_CHANGED_CH	-	-	o	-	-
AACD_ERR_PROGRAM_CONFIG	-	-	o	-	-
AACD_ERR_1ST2ND_FRAME	-	-	o	-	-
AACD_ERR_CRC	-	-	o	-	-
AACD_ERR_DECODE	-	-	o	-	-

[Note] o : Error factor might be output. - : Error factor is not used.

[Note] *1 : Returns an error factor.In the case of could not returns an error factor, returns AACD_RESULT_FATAL as error code.

3.4 Memory Specifications

This section describes the memory areas used by this middleware.

3.4.1 Scratch Area

Table 3.7 Scratch Area Description

Item	Area which temporarily contains values when this middleware is used. If the user manipulates this area for interrupt or other processing while a function in this middleware is being called, the correct execution of this middleware cannot be ensured. The user can freely use this area after decoding one frame.
Symbol name	- (freely defined by the user)
Size	Obtain the actually required size with aacd_GetMemorySize.
Area reservation	The user should reserve this area. The user can freely use this area after returning from a function in this middleware. Note that if the user calls a function in this middleware after using this area, the value stored in this area is overwritten.
Allocation	This area is included in RAM.
Alignment	Align this area on an 8-byte boundary.

3.4.2 Static Area

Table 3.8 Static Area Description

Item	Area which always holds values when this middleware is used. If the user manipulates this area after initialization, the correct execution of this middleware is not ensured.
Symbol name	- (freely defined by the user)
Size	Obtain the actually required size with aacd_GetMemorySize.
Area reservation	The user should reserve this area.
Allocation	This area is included in RAM.
Alignment	Align this area on a 4-byte boundary.

3.4.3 Middleware Stack Area

Table 3.9 Middleware Stack Area Description

Item	Stack area used by this middleware
Symbol name	- (freely defined by the user)
Size	Obtain the actually required size with aacd_GetMemorySize.
Area reservation	The user should reserve this area. To use this middleware, reserve a middleware stack area which exceeds the size above.
Allocation	This area is included in RAM.
Alignment	-

3.4.4 Heap Area

This middleware does not use a heap area.

3.4.5 Input Buffer

Table3.10 Input Buffer Description

Item	Area which stores inputs to this middleware. The input buffer contains stream data (AAC-compressed data). If the user manipulates this area during decode processing, the normal execution of the program cannot be ensured. [Note]This middleware does not support an input buffer which is a ring buffer.
Symbol name	- (freely defined by the user)
Size	Reserve an amount of memory equal to the size of one frame. The recommended size is the size (nInputBufferSize member of the memory size acquisition information structure) obtained by this middleware's aacd_GetMemorySize function.
Area reservation	The user should reserve this area. The user can freely use this area after decoding one frame.
Allocation	This area is included in RAM.
Alignment	Alignment is not restricted.

When executing the aacd_Decode function, specify the parameters in the buffer memory settings structure. This stores the processing results into the buffer memory results structure. Figure 3.2 shows the relationship between the input buffer and structure members.

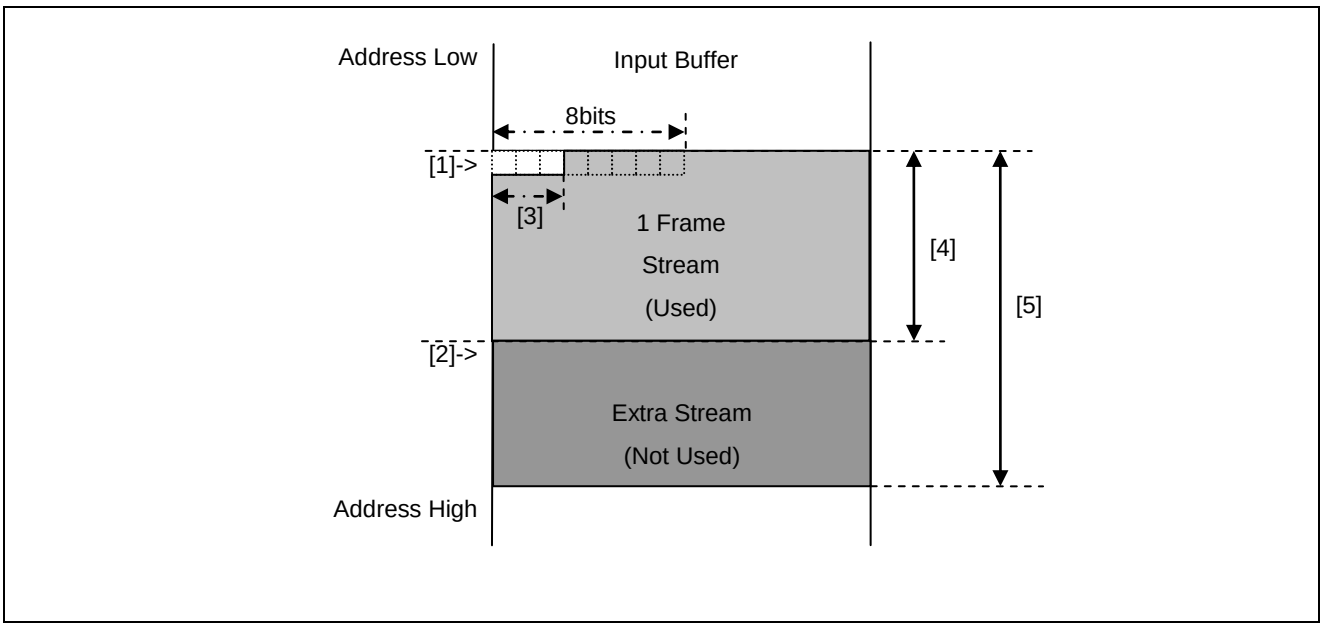

Figure 3.2 Structure Members in the Input Buffer

[Note] Data exceeding the size of one frame is placed into the input buffer for decode processing.

Table3.11 Structure Members in the Input Buffer

[1]	pInBuffStart	(Buffer memory settings structure)	Input data start address
[2]	pInBuffLast	(Buffer memory results structure)	Input buffer post-read address
[3]	nInBuffOffsetBits	(Buffer memory settings structure)	Input start bit offset
[4]	nInBuffUsedDataSize	(Buffer memory results structure)	Input buffer consumed-data size
[5]	nInBuffSetDataSize	(Buffer memory settings structure):	Input data size

[Note] Items [1] and [2] indicate addresses. Items [3], [4] and [5] indicate sizes.
The smallest unit of stream data is 8 bits.

If the stream start bit is not byte-aligned, you can specify the stream start bit with the nInBuffOffsetBits value in the buffer memory settings structure. Figure 3.3 shows a sample nInBuffOffsetBits value of 3.

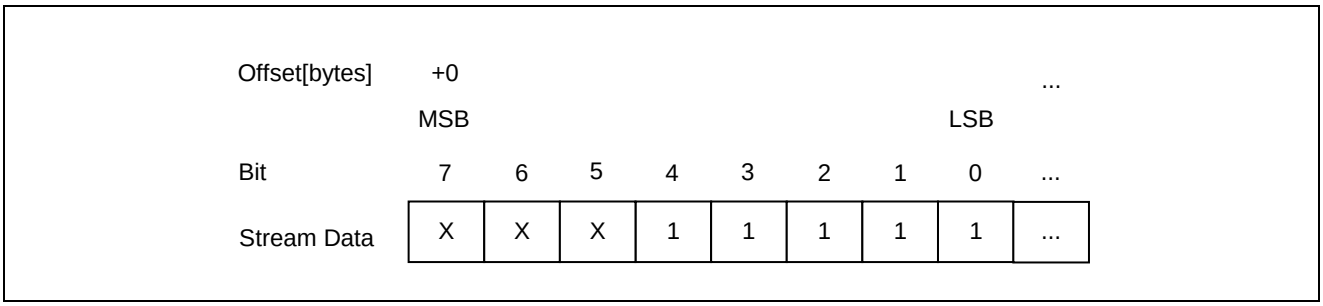

Figure 3.3 nInBuffOffsetBits = 3

[Note] "X" is an arbitrary value which is not stream data.

(1) Input data storage method

Figure 3.4 shows how input data is stored. Store data given in bytes into the input buffer (memory).

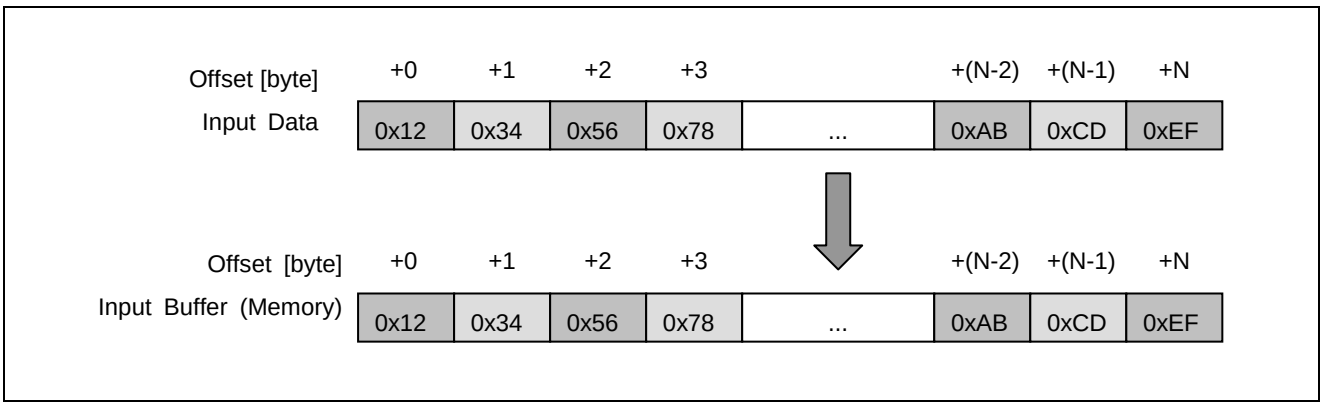

Figure 3.4 Input Data Storage Method

3.4.6 Output Buffer

Table 3.12 Output Buffer Description

Item	Area which stores outputs from this middleware. The output buffer contains 16/32-bit linear PCM data (hereinafter called PCM data). If the user manipulates this area during decode processing, the normal execution of the program cannot be ensured.
Symbol name	- (freely defined by the user)
Size	Size is freely defined by the user. This size should not be less than the size obtained by the aacd_GetMemorySize function. [Note] The size obtained by the aacd_GetMemorySize function is equal to the size per channel.
Area reservation	The user should reserve this area. The user can freely use this area after PCM data generation.
Allocation	This area is included in RAM.
Alignment	To output data in 16-bit PCM data, align this area on a 2-byte boundary. To output data in 32-bit PCM data, align this area on a 4-byte boundary.

When executing the aacd_Decode function, specify the parameters in the buffer memory settings structure. This stores the processing results into the buffer memory results structure. Figure 3.5 shows the relationship between the output buffer and the structure members.

The second and subsequent channels are managed just like the first channel. It does not matter whether the output buffers for the first and second channels are consecutive or not. For the number of samples output to the output buffer, refer to the description of nOutSamplesPerFrame of the decode results structure.

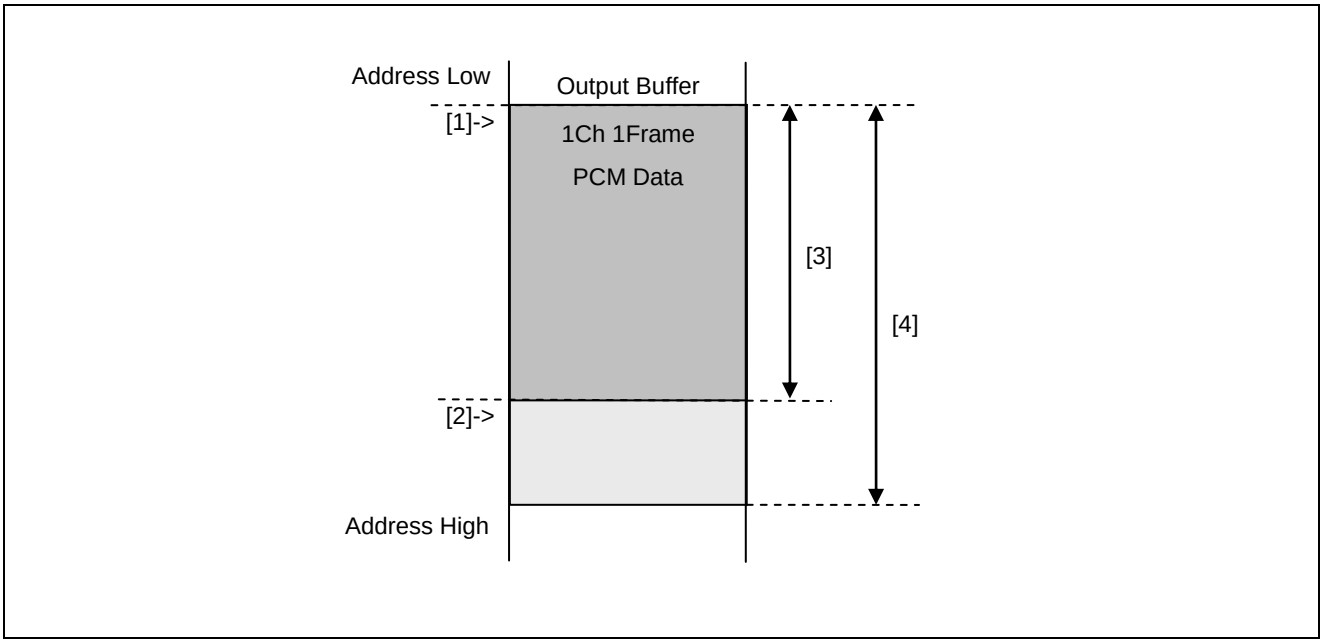

Figure 3.5 Structure Members in the Output Buffer

[Note] Output results are placed into the output buffer for decode.

Table3.13 Structure Members in the Output Buffer

[1]	pOutBuffStart[0]	(Buffer memory settings structure)	Output data start address for the first channel
[2]	pOutBuffLast[0]	(Buffer memory results structure)	Post-write address for the first channel
[3]	nOutBuffUsedDataSize	(Buffer memory results structure)	Consumed data size per channel
[4]	nOutBuffSize	(Buffer memory settings structure)	Output buffer size per channel

[Note] Items [1] and [2] indicate addresses. Items [3] and [4] indicate sizes.

(1) Output data storage method

The output buffer (memory) stores data in 2-byte (16-bit) or 4-byte (32-bit) units. The byte order for accessing the buffer is little endian (see Figure 3.6, Figure 3.7).

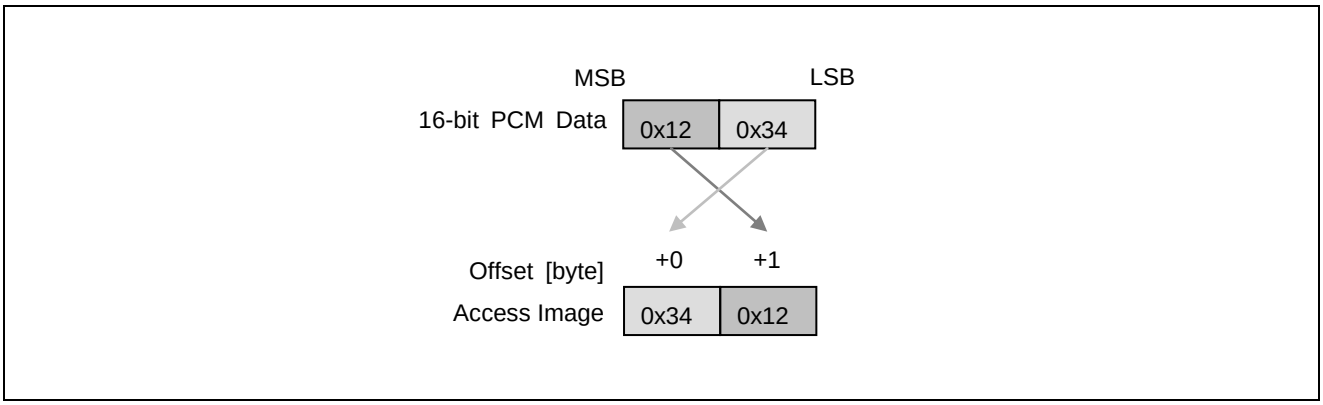

Figure 3.6 16-bit PCM Data Access (Little Endian Mode)

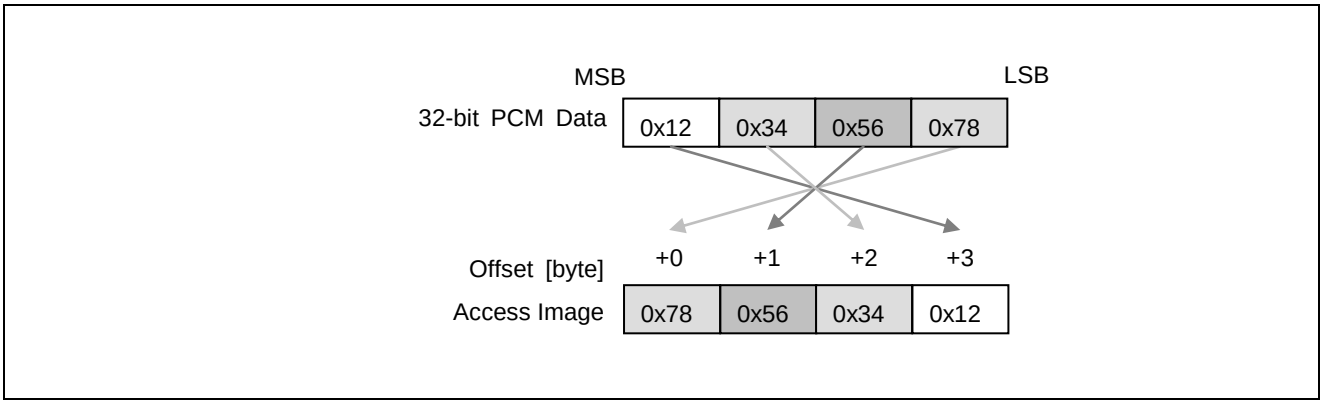

Figure 3.7 32-bit PCM Data Access (Little Endian Mode)

Figure 3.8 shows the PCM data bit positions.

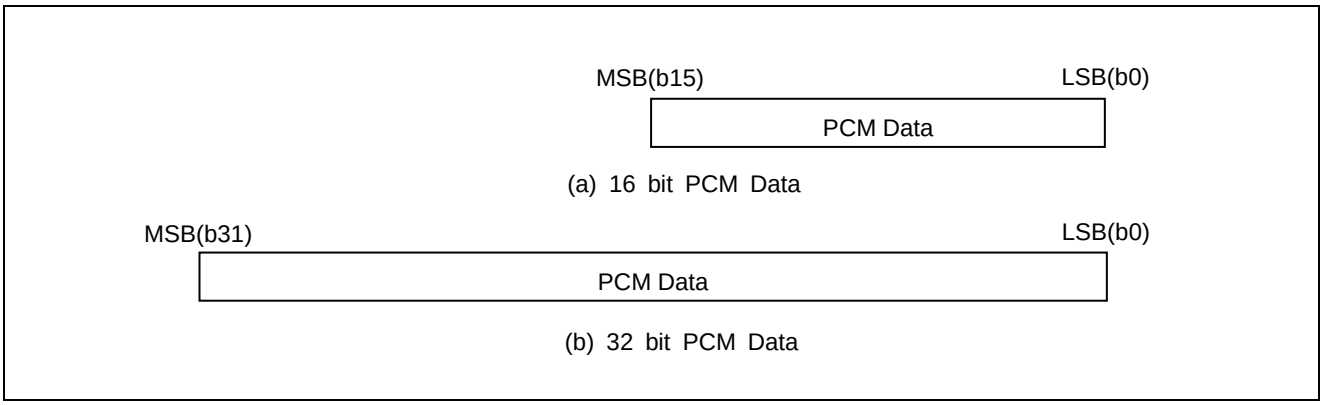

Figure 3.8 PCM Data Bit Positions

3.4.7 Data Stream Buffer

The data stream buffer is an area which stores byte data included in DSE. The user should reserve this area. Doing this is unnecessary if there is no need to obtain byte data included in DSE. If the user does not reserve this area, byte data included in DSE is discarded.

The maximum size of byte data included in DSE should be 8,160 bytes (510 bytes/instance * 16 instances). If byte data exceeds the reserved data stream buffer size, the excess data is discarded. Thus, you should not always reserve an area of the maximum size. Reserve an area large enough to contain data necessary for the user.

Figure 3.9 shows the relationship between the data stream buffers and structure members.

The buffers shown in Figure 3.9 are two sample data stream buffers for decode processing (nDseBuffNum = 2 in the buffer memory settings structure). Byte data of the data size ([3]), included in the first DSE decoded, is stored starting at the beginning ([1]) of the data stream buffer (buffer 0). If DSE which has different element_instance_tag's for the same frame is decoded, byte data of the data size ([4]), included in this DSE, is stored starting at the position indicated by the offset equal to the data stream buffer size ([2]) from the beginning ([1]) of the data stream buffer. If byte data included in DSE exceeds the data size ([3]), an amount of data equal to the data size ([3]) is stored starting at the beginning of the data stream buffer.

The size of data stored in each buffer is set in aDseDataSize of the buffer memory settings structure. nDseNum of the decode results structure indicates the number of bytes stored. aDseTag of this structure indicates the element_instance_tag value.

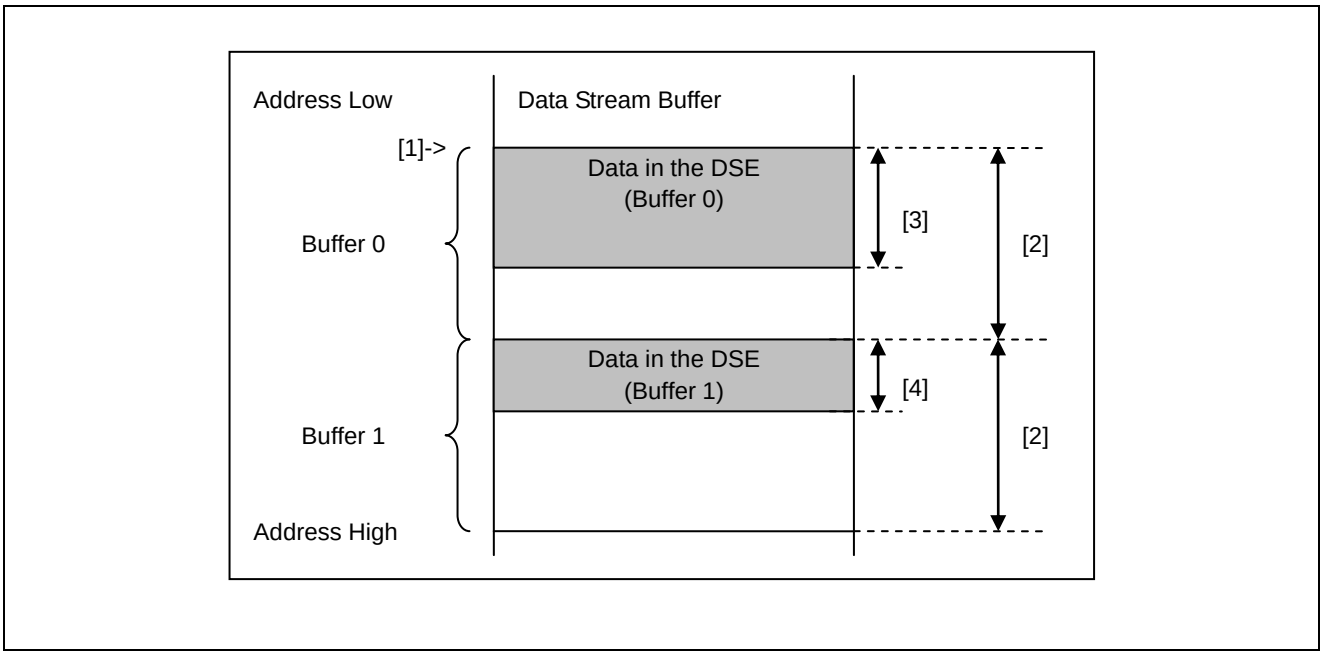

Figure 3.9 Sample Data Stream Buffers

Table 3.14 Structure Members for Data Stream Buffers

[1]	pDseBuffStart	(Buffer memory settings structure)	Data stream buffer start address
[2]	nDseBuffSize	(Buffer memory settings structure)	Data stream buffer size (in bytes)
[3]	aDseDataSize [0]	(Buffer memory results structure)	Size (in bytes) of data stored in the data stream buffer (buffer 0)
[4]	aDseDataSize [1]	(Buffer memory results structure)	Size (in bytes) of data stored in the data stream buffer (buffer 1)

3.5 Input Data

You can input AAC LC compressed data to this middleware. This section describes the ADTS and RAW formats supported by this middleware. For information about storing input data, refer to Section 3.4.5.

3.5.1 ADTS Format

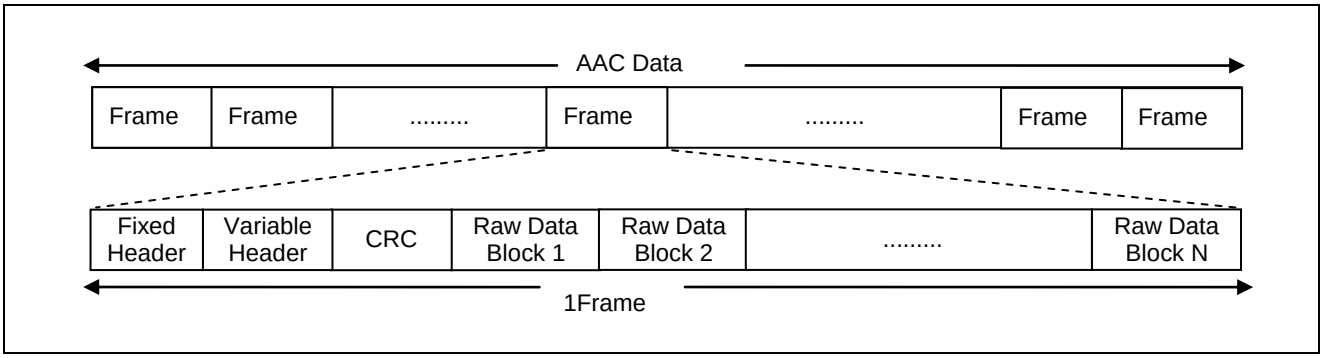

Figure 3.10 Compressed Data Format

[Fixed Header]

A header which can be used for random access to any ADTS frames. This header always contains the same value for all ADTS frames. It contains information such as syncword, sampling frequency, channel configuration, etc.

Table 3.15 ADTS Fixed Header Configuration

	Bit	Details
Syncword	12	Synchronization word which is the bit string "1111 1111 1111".
ID	1	MPEG ID (0: MPEG-4 AAC, 1: MPEG-2 AAC)
Layer	2	Layer type. "00" is set.
protection_absent	1	Specifies whether there is error check data. (0: Yes, 1: No)
Profile_ObjectType	2	Dependent on ID (see Table3.16)
sampling_frequency_index	4	Sampling frequency (see Table3.17)
private_bit	1	Bit for private purposes
channel_configuration	3	Channel configuration (see Table3.18)
original_copy	1	Specifies whether the copyright is protected or not. (0: No, 1: Yes)
home	1	Discriminates the original from the copy. (0: Copy, 1: Original)
Emphasis	2	Adds only the initial specifications. (Not supported by this middleware.)

Table3.16 Profile_ObjectType

MPEG-2 AAC profile(ID == 1)	Value	MPEG-4 AAC object types (ID == 0)	Value
AAC Main	00	AAC Main	00
AAC LC	01	AAC LC	01
AAC SSR	10	AAC SSR	10
(reserved)	11	AAC LTP	11

[Note]This middleware supports only the AAC LC profile.

Table3.17 sampling_frequency_index

Sampling frequency (Hz)	Value
96000	0000
88200	0001
64000	0010
48000	0011
44100	0100
32000	0101
24000	0110
22050	0111
16000	1000
12000	1001
11025	1010
8000	1011
7350	1100*1
(Reserved)	1101*1
(Reserved)	1110*1
(Escape value)	1111*1

[Note]*1 : Not supported by this middleware.

Table3.18 channel_configuration

Number of speakers	Value	Audio syntactic elements, listed in order received	Speaker position
-	000	program_config_element	-
1	001	single_channel_element	Center front speaker
2	010	channel_pair_element	Left, right front speakers
3	011	single_channel_element channel_pair_element	Center front speaker Left, right front speakers
4	100	single_channel_element channel_pair_element single_channel_element	Center front speaker Left, right center front speakers, Rear surround
5	101	single_channel_element channel_pair_element channel_pair_element	Center front speaker Left, right front speakers, Left surround, right surround rear speakers
5+1	110	single_channel_element channel_pair_element channel_pair_element lfe_channel_element	Center front speaker Left, right front speakers, Left surround, right surround rear speakers, Front low frequency effects speaker
7+1	111	single_channel_element channel_pair_element channel_pair_element channel_pair_element lfe_channel_element	Center front speaker Left, right center front speakers, Left, right outside front speakers, Left surround, right surround rear speakers, Front low frequency effects speaker

[Variable Header]

A header which can be used to change the values of information for each ADTS frame. This header contains information such as the size of an ADTS frame, the number of raw data blocks in the ADTS frame, etc.

Table3.19 ADTS Variable Header Configuration

	Bit	Details
copyright_identification_bit	1	One bit in the 72-bit copyright identification field
copyright_identification_start	1	Indicates that the copyright identification bit is the first one bit.
aac_frame_length	13	Length (in bytes) of one frame
adts_buffer_fullness	11	Number of bits available to the buffer (0x7FF: VBR->Unavailable)
no_raw_data_blocks_in_frame	2	Number of raw data blocks in the frame - 1

[CRC Check]

Data for CRC error detection. This data does not exist if the parameter "protection_absent" in the ADTS fixed header is 1.

[RAW Data Block](See Section 3.5.2)

A raw data block contains encoded audio data and its related information. One ADTS frame can contain up to four raw data blocks. This middleware cannot handle an ADTS frame which contains two or more raw data blocks.

3.5.2 RAW Format

The RAW format consists of multiple raw data blocks and does not include a header.

The raw data block configuration varies depending on the three ID bits. Raw data blocks contain not only related information but also various data such as encoded audio data. Each raw data block should end with a terminate ID.

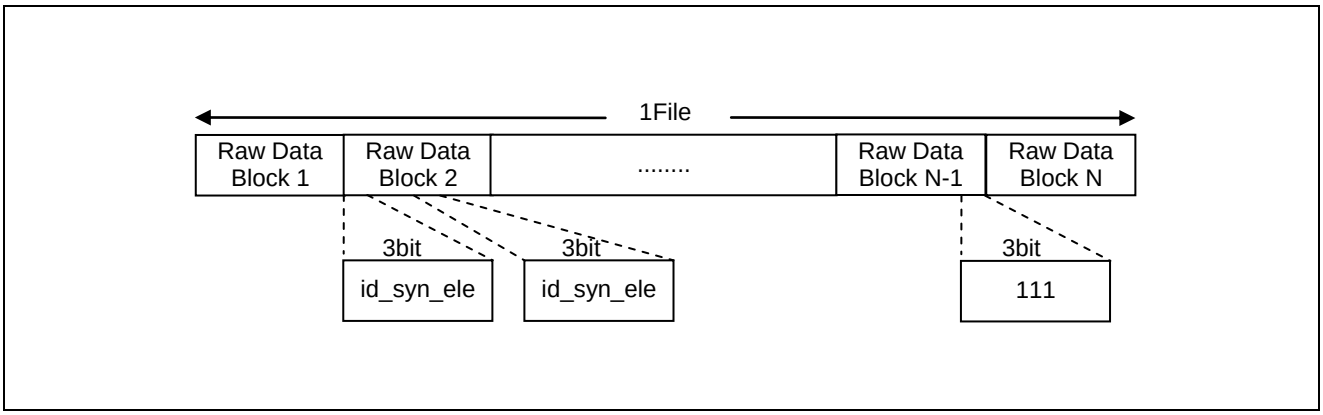

Figure 3.11 RAW Format

Table 3.20 id_syn_ele

ID name	Value	Syntactic Element
ID_SCE	000	single_channel_element
ID_CPE	001	channel_pair_element
ID_CCE	010	coupling_channel_element
ID_LFE	011	lfe_channel_element
ID_DSE	100	data_stream_element
ID_PCE	101	program_config_element
ID_FIL	110	fill_element
ID_END	111	terminator

3.6 Output Data

16/32-bit linear PCM data is output. The number of samples is 1,024 for each channel or frame.

4. Precautions

This section provides precautions in creating an application.

4.1 Precautions in Calling a Function

The user program which calls a function in this middleware should follow the calling rules for the compiler used.

4.1.1 Function Execution Timing

The following describes the timing of executing functions in this middleware.

(1) aacd_GetMemorySize function

You can execute this function at any time before executing the aacd_Init function. Execute the aacd_GetMemorySize function to reserve the necessary amount of memory.

(2) aacd_Init function

Execute this function only once before starting a series of decode process steps. They make up a process from the start to the end of decoding a certain stream.

(3) aacd_Decode function

Execute this function when you decode one frame of stream data.

(4) aacd_GetErrorFactor function

You can execute this function at any time after executing the aacd_Init function.

(5) aacd_GetVersion function

You can execute this function at any time.

4.2 Other Precautions

4.2.1 Reserving and Allocating Memory Areas

Before calling a function in this middleware, reserve a static area, a scratch area, an input/output buffer area, and areas for structures which should hold the arguments of functions.

4.2.2 Access Outside a Memory Range

This middleware does not access memory space outside the reserved areas.

4.2.3 Combination with Other Applications

If you use this middleware together with other applications, be careful to avoid the duplication of symbol names.

4.2.4 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	AAC Decode Middleware User's Manual

Rev.	Date	Description	
		Page	Summary
1.00	Dec 12, 2014	-	First Edition issued

AAC Decode Middleware User's Manual

Publication Date: December 12, 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

Colophon 3.0

AAC Decode Middleware

Renesas Electronics Corporation