User's Manual

WMA Decode Middleware

User's Manual

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (http://www.renesas.com).

Rev.1.01 August, 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.

(2012.4)

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.

This product is protected by certain intellectual property rights of Microsoft and cannot be used or further distributed without a license from Microsoft.

3. Related Documents

WMA-related Documents

Specifications Numbers and Titles	Date of issuance
ASF Specifications	
Advanced Systems Format (ASF) Specification Revision 01.20.02	June 15, 2004
WMA Standard Specifications	
Decoding WMA (Standard) An Overview of Windows Media Audio (Standard) Decoding	May 23, 2002

Processor-related Documents

Refer to attached product manual.

4. List of Abbreviations and Acronyms

Abbreviation	Full Form
ABR	Average Bit Rate
ANSI-C	American National Standards Institute - C
ASF	Advanced Systems Format
bps	bits per second
CBR	Constant Bit Rate
DAC	digital to analog converter
DRC	Dynamic Range Control
LSB	Least Significant Bit
MBR	Multiple Bit Rate
MSB	Most Significant Bit
PCM	Pulse Code Modulation
RAM	Random Access Memory
ROM	Read Only Memory
WMA	Windows Media Audio
WM DRM	Windows Media Digital Rights Management

Table of Contents

WMA Decode Middleware

Rev.1.01
August 29, 2014

1. Overview

This section provides an overview of the WMA decoder.

1.1 Specifications Outline

Windows Media Audio (WMA) is an audio codec developed by Microsoft for coding or decoding an audio signal. This middleware, which meets the WMA Standard Specifications, decodes compressed input data and outputs the decoding results. For these specifications, refer to Table 1.1. For basic specifications and performance, refer to attached product manual.

Table 1.1 Supported WMA Specifications

Item	Description
Input data format	WMA Standard Version 2, 7, 8, 9, 9.1, 9.2 (or later) All profile (L1/L2/L3)
Output data format	16-bit linear PCM
Sampling frequency (Hz) supported	Profile L1: 44,100 Profile L2/L3: 8,000 / 11,025 / 16,000 / 22,050 / 32,000 / 44,100 / 48,000
Number of channels supported	1 channel 2 channels
Bit rate (kbps) supported	Profile L1: 64 to 161 Profile L2: 161 or less to 161 Profile L3: 385 or less to 385 [Note] Operation of this middleware has been confirmed using the Microsoft test data (which is sent at minimum bit rates from 128 bps for Profile L2 to 95,224 bps for Profile L3). [Note] This middleware can decode data which is sent at fixed bit rate (CBR), variable bit rate (VBR), multi-bit rate (MBR) and average bit rate (ABR). To decode MBR data requires selecting one stream to be decoded and demultiplexing it.
Reentrant	This middleware is reentrant.
Restrictions	This middleware does not support these functions: ・ASF Demux function ・WM DRM Decrypt function ・DRC function

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			65,536
		Area breakdown	Static area	Size breakdown	49,152
			Scratch area		16,384
		User work area			53,424
		Area breakdown	Input buffer	Size breakdown	20,480
			Output buffer		32,768
			Structure		176
		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, Constant table area, and Other area refer to attached product manual.

1.2 Configuration

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

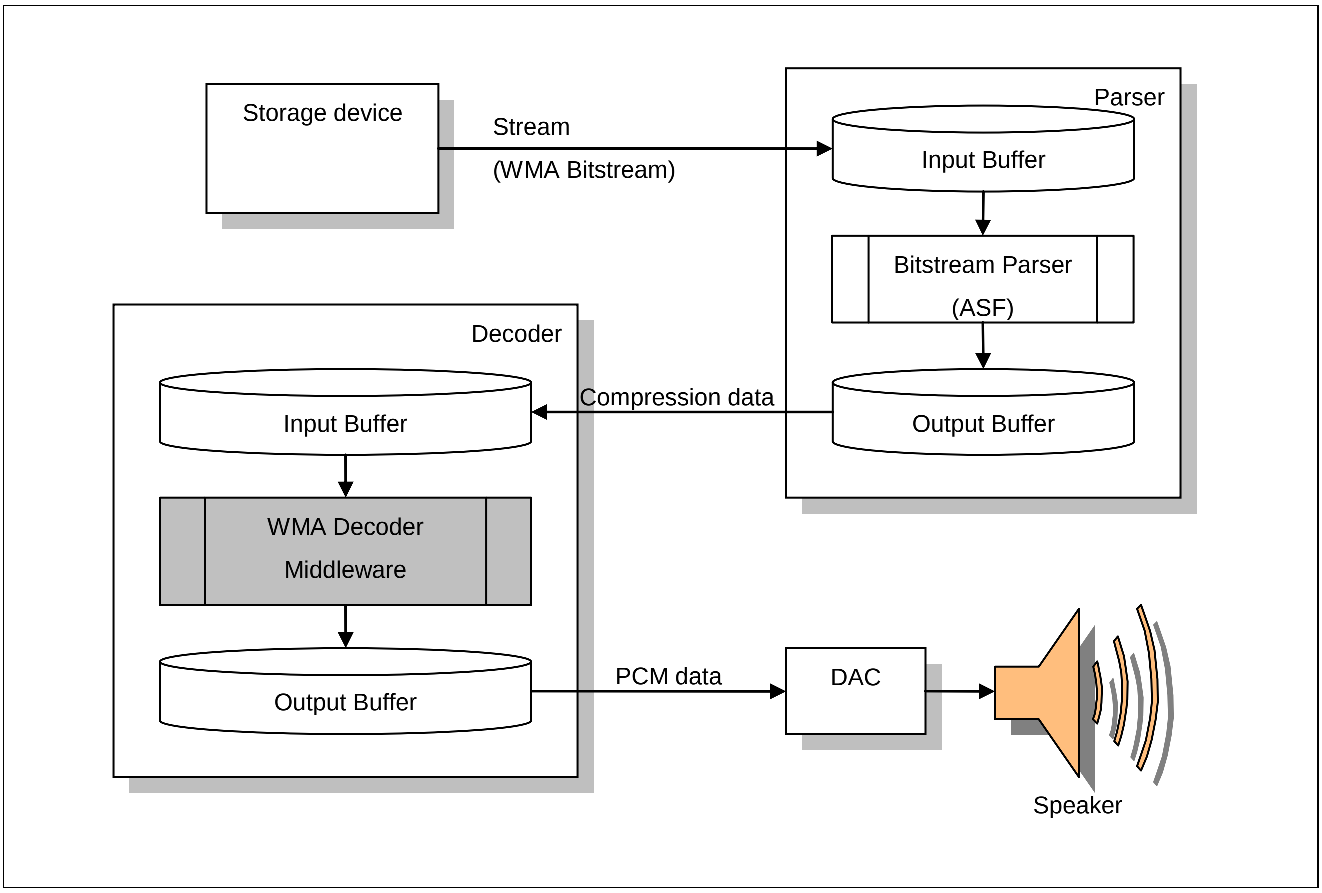


Figure1.1 Example of the Decode System Configuration

1. WMA Bitstream

   WMA Bitstream is a linear PCM data sample compressed according to the WMA specifications. For these specifications, refer to Table 1.1.

2. Parser

   The parser eliminates unnecessary parts from WMA Bitstream. Then, it outputs the remaining data in units hereinafter called data blocks as Payload Data or Sub-Palyload #N Data (n = 0, 1, etc.). The user should design the parser to suit the target system.

3. Compression data

   Output data in data blocks.

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

6. DAC

The DAC converts 16-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
wmastd_GetMemorySize	Calculates the required memory size.
wmastd_Init	Initializes the WMA decoder.
wmastd_AudecGetData	Inputs data.
wmastd_AudioDecode	Decodes data.
wmastd_ReconstructPcmData	Generates PCM data.
wmastd_GetErrorFactor	Obtains an error factor.
wmastd_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
Data input settings structure	Stores the parameters necessary for data input.	I
Data input results structure	Stores the results of data input.	O
Decode settings structure	Stores the parameters necessary for decoding.	I
Decode results structure	Stores the decoding results.	O
PCM data generation settings structure	Stores the parameters necessary for PCM data generation.	I
PCM data generation results structure	Stores the results of PCM data generation.	O

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
WMASTD_RESULT_OK	0x00000000	Processing results are normal.
WMASTD_RESULT_NG	0x00000001	Processing results are abnormal.
WMASTD_RESULT_WARNING	0x00000002	Abnormality has occurred, which does not prevent the process from continuing.
WMASTD_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	wmastd_XXXX
Structure names	wmastd_XXXX
Return values from functions	WMASTD_RESULT_XXXX [Note] XXXX consists only of upper-case letters.
Error factor names	WMASTD_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	WMASTD_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. Figure 2.2 shows a flow diagram of processing for guarding against errors which occur in the wmastd_Init, wmastd_AudecGetData, wmastd_AudioDecode, and wmastd_ReconstructPcmData functions.

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 2.3 and Figure 2.4 show a flow diagram of processing sample program and parser.

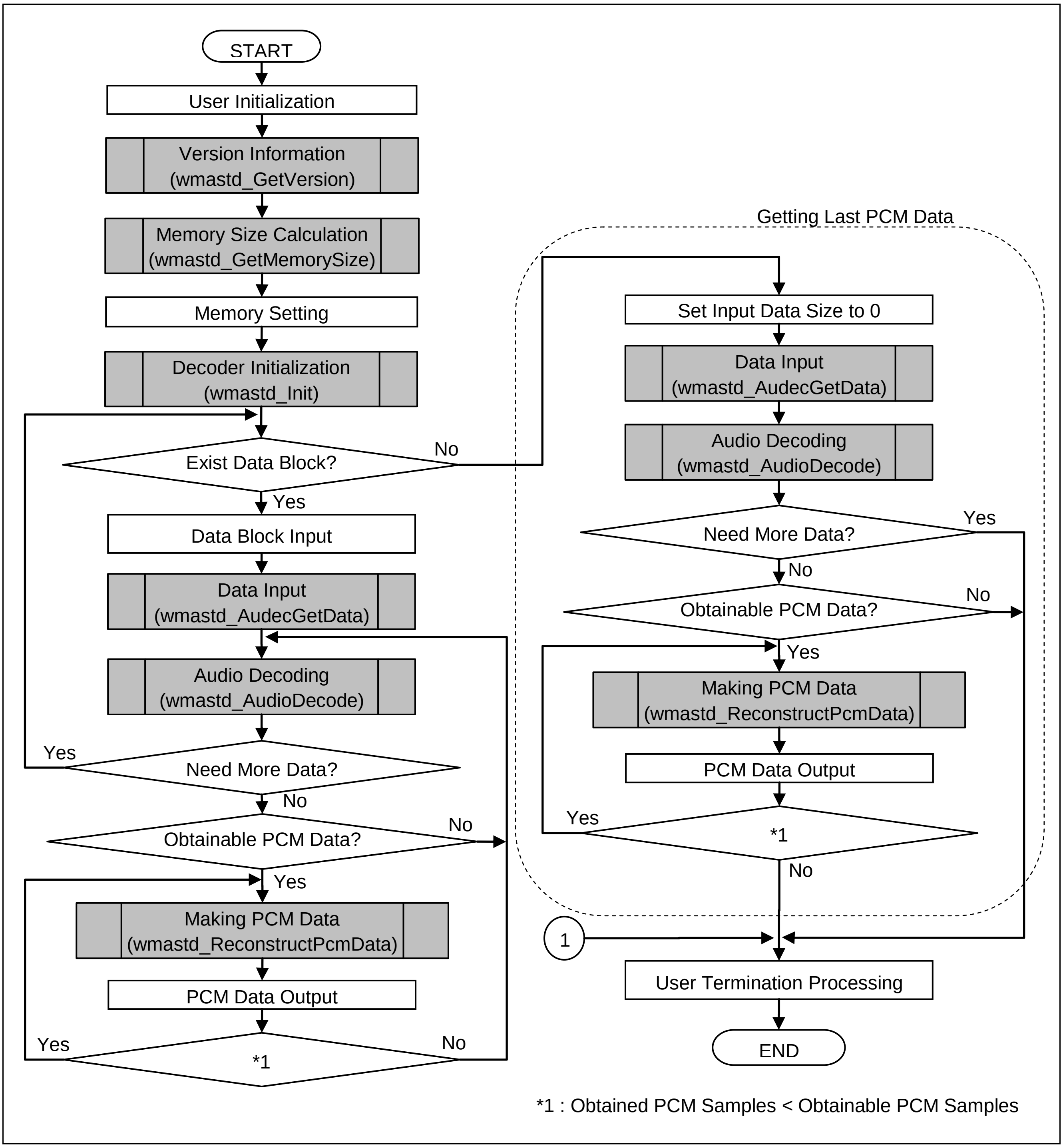


Figure 2.1 Example of the Application Processing Flow

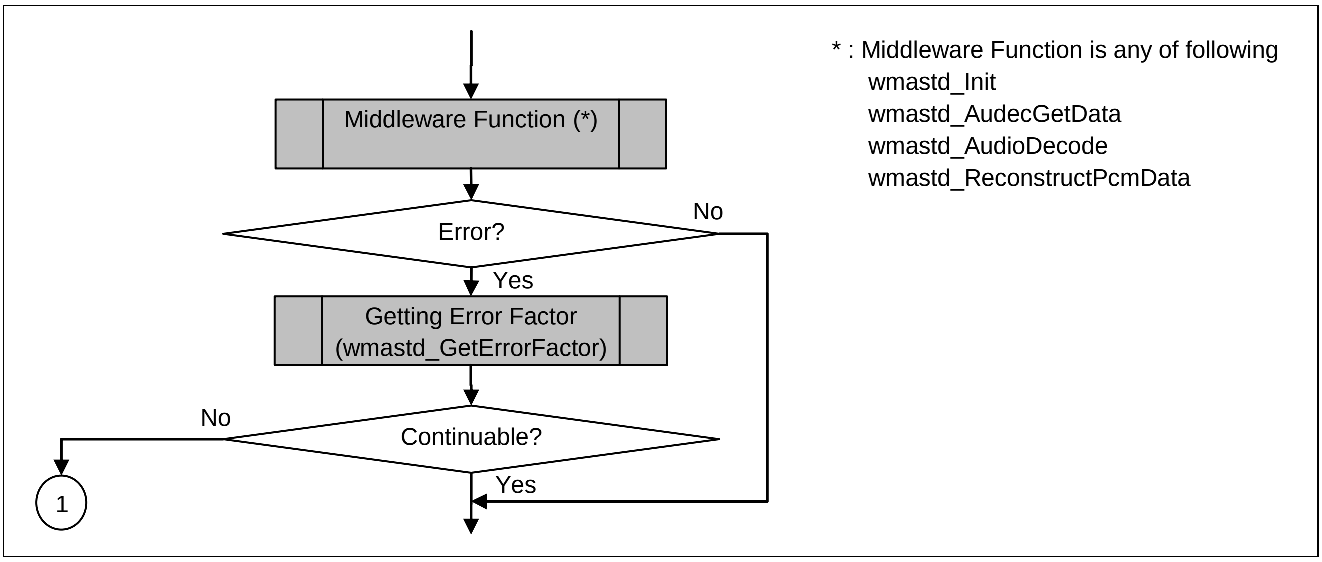


Figure 2.2 Example of the Error Processing Flow

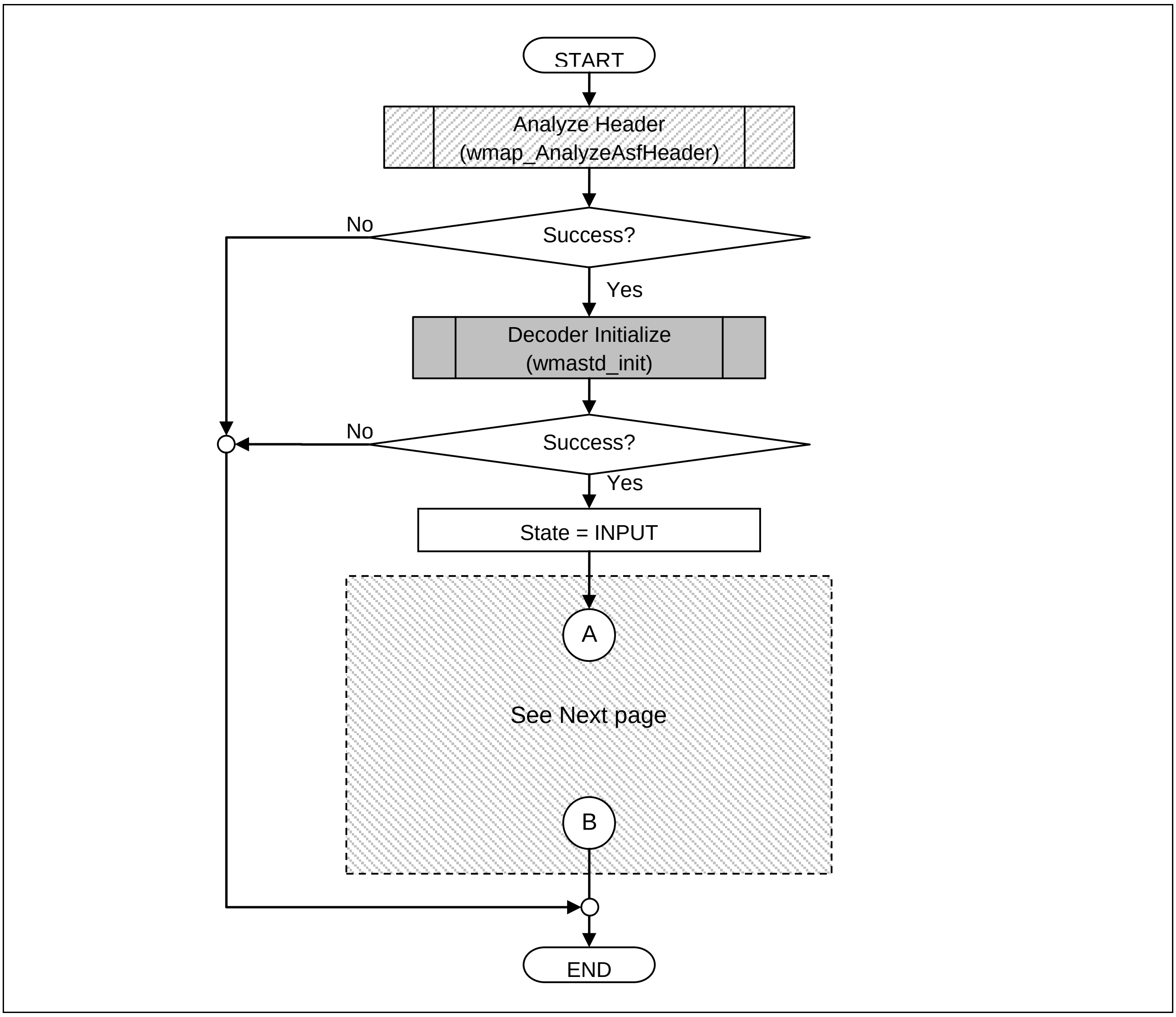


Figure 2.3 Example of the sample program and parser flow - 1

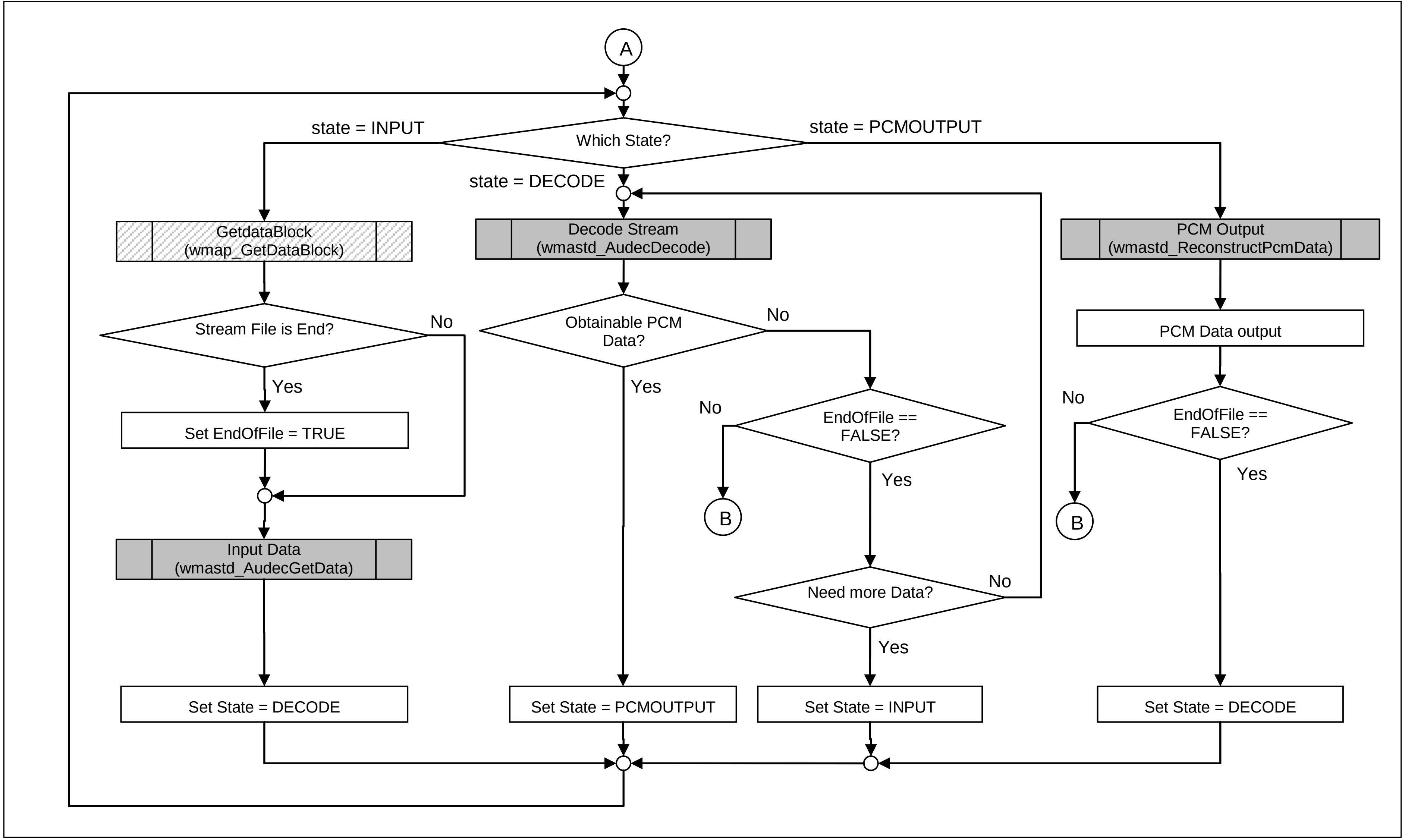


Figure 2.4 Example of the sample program and parser flow - 2

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 wmastd_GetMemorySize Function

Synopsis	Calculates the necessary memory size.	
Syntax	ACMW_INT32 wmastd_GetMemorySize(const wmastd_getMemorySizeConfigInfo * const pGetMemorySizeConfigInfo, wmastd_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
wmastd_getMemorySizeConfigInfo *pGetMemorySizeConfigInfo	I	Memory size acquisition settings structure
wmastd_getMemorySizeStatusInfo *pGetMemorySizeStatusInfo	O	Memory size acquisition results structure
Return value	ACMW_INT32 errorCode	Error code For details, refer to Table 3.4.
Description	Execute this function before wmastd_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 wmastd_Init Function

Synopsis	Initializes the WMA decoder.	
Syntax	ACMW_INT32 wmastd_Init(const wmastd_workMemoryInfo * const pWorkMemInfo, const wmastd_initConfigInfo * const pInitConfigInfo, wmastd_initStatusInfo * const pInitStatusInfo);	
Function	This function initializes the static and scratch areas which are used by this middleware. Also, it sets various parameters.	
Arguments	I/O	Meaning
wmastd_workMemoryInfo *pWorkMemInfo	I	Work memory information structure
wmastd_initConfigInfo *pInitConfigInfo	I	Initialization settings structure
wmastd_initStatusInfo *pInitStatusInfo	O	Initialization results structure
Return value	ACMW_INT32 errorCode	Error code For details, refer to Table 3.4.
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 wmastd_AudecGetData Function

Synopsis	Inputs data.	
Syntax	ACMW_INT32 wmastd_AudecGetData(const wmastd_workMemoryInfo * const pWorkMemInfo, const wmastd_audecGetDataConfigInfo * const pAudecGetDataConfigInfo, wmastd_audecGetDataStatusInfo * const pAudecGetDataStatusInfo);	
Function	This function inputs data blocks.	
Arguments	I/O	Meaning
wmastd_workMemoryInfo *pWorkMemInfo	I	Work memory information structure
wmastd_audecGetDataConfigInfo *pAudecGetDataConfigInfo	I	Data input settings structure
wmastd_audecGetDataStatusInfo *pAudecGetDataStatusInfo	O	Data input results structure
Return value	ACMW_INT32 errorCode	Error code For details, refer to Table 3.4.
Description	Execute this function to input data blocks to the middleware. [Note] If the size of a data block is 4 bytes or more, be sure to input 4 bytes or more of data to the input buffer. If the size of a data block is less than 4 bytes, input the entire data block at a time.	

3.1.4 wmastd_AudioDecode Function

Synopsis	Decodes data.	
Syntax	ACMW_INT32 wmastd_AudioDecode(const wmastd_workMemoryInfo * const pWorkMemInfo, const wmastd_audioDecodeConfigInfo * const pAudioDecodeConfigInfo, wmastd_audioDecodeStatusInfo * const pAudioDecodeStatusInfo);	
Function	This function decodes data blocks.	
Arguments	I/O	Meaning
wmastd_workMemoryInfo *pWorkMemInfo	I	Work memory information structure
wmastd_audioDecodeConfigInfo *pAudioDecodeConfigInfo	I	Decode settings structure
wmastd_audioDecodeStatusInfo *pAudioDecodeStatusInfo	O	Decode results structure
Return value	ACMW_INT32 errorCode	Error code For details, refer to Table 3.4.
Description	Execute this function to decode input data blocks. [Note] If an error occurs, the members of the decode results structure are indefinite. Do not reference these members.	

3.1.5 wmastd_ReconstructPcmData Function

Synopsis	Generates PCM data.	
Syntax	ACMW_INT32 wmastd_ReconstructPcmData(const wmastd_workMemoryInfo * const pWorkMemInfo, const wmastd_reconstructPcmDataConfigInfo * const pReconstructPcmDataConfigInfo, wmastd_reconstructPcmDataStatusInfo * const pReconstructPcmDataStatusInfo);	
Function	This function generates PCM data.	
Arguments	I/O	Meaning
wmastd_workMemoryInfo *pWorkMemInfo	I	Work memory information structure
wmastd_reconstructPcmDataConfigInfo *pReconstructPcmDataConfigInfo	I	PCM data generation settings structure
wmastd_reconstructPcmDataStatusInfo *pReconstructPcmDataStatusInfo	O	PCM data generation results structure
Return value	ACMW_INT32 errorCode	Error code For details, refer to Table 3.4.
Description	Execute this function to generate PCM data. [Note] If an error occurs, the members of the decode results structure are indefinite. Do not reference these members.	

3.1.6 wmastd_GetErrorFactor Function

Synopsis	Obtains error factors.	
Syntax	ACMW_UINT32 wmastd_GetErrorFactor(const wmastd_workMemoryInfo * const pWorkMemInfo);	
Function	This function returns any error factors related to the most recently executed wmastd_Init, wmastd_AudecGetData, wmastd_AudioDecode, and wmastd_ReconstructPcmData functions.	
Arguments	I/O	Meaning
wmastd_workMemoryInfo *pWorkMemInfo	I	Work memory information structure
Return value	ACMW_UINT32 errorFactor	Value indicating an error factor For details, refer to Table 3.6.
Description	This function returns any error factors related to the most recently executed wmastd_Init, wmastd_AudecGetData, wmastd_AudioDecode, and wmastd_ReconstructPcmData 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 wmastd_Init function initializes the static area. The error factors are overwritten next time you execute the wmastd_Init, wmastd_AudecGetData, wmastd_AudioDecode, and wmastd_ReconstructPcmData functions. So, if you need the error factors, execute this function before reexecuting the functions above.	

3.1.7 wmastd_GetVersion Function

Synopsis	Obtains version information.		
Syntax	ACMW_UINT32 wmastd_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] wmastd_getMemorySizeConfigInfo

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

[Prototype]

```
typedef struct {
        ACMW_INT32          reserve;
} wmastd_getMemorySizeConfigInfo;
```

[Member description]

Member Variable Name	Description
reserve	Reserved value (which should be 0).

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

3.2.2 Memory Size Acquisition Results Structure

[Structure name] wmastd_getMemorySizeStatusInfo

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

[Prototype]

```
typedef struct {
        ACMW_UINT32     nStaticSize;
        ACMW_UINT32     nScratchSize;
        ACMW_UINT32     nInputBufferSize;
        ACMW_UINT32     nOutputBufferSize;
        ACMW_UINT32     nStackSize;
        ACMW_INT32      reserve;
} wmastd_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 *1
nOutputBufferSize	Necessary memory size (in bytes) of the output buffer area *1 *2
nStackSize	Necessary memory size (in bytes) of the software stack area
reserve	Reserved

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

[Remarks2] *1: The recommended size is equal to the nInputBufferSize and nOutputBufferSize values. Specify the size which suits the target system.
*2: The size of the output buffer area is equal to that for each channel in non-interleaved format.

3.2.3 Work Memory Information Structure

[Structure name] wmastd_workMemoryInfo

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

[Prototype]

```
typedef struct {
        void *                  pStatic;
        void *                  pScratch;
        ACMW_INT32              reserve;
} wmastd_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.
reserve	Reserved value (which should be 0)

[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 (wmastd_GetMemorySize function).

3.2.4 Initialization Settings Structure

[Structure name] wmastd_initConfigInfo

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

[Prototype]

```
typedef struct {
        ACMW_UINT16     wFormatTag;
        ACMW_UINT16     nChannels;
        ACMW_UINT32     nSamplesPerSec;
        ACMW_UINT32     nAvgBytesPerSec;
        ACMW_UINT16     nBlockAlign;
        ACMW_UINT16     nValidBitsPerSample;
        ACMW_UINT8      codecSpecificData[18];
        ACMW_INT32      reserve;
} wmastd_initConfigInfo;
```

[Member description]

Member Variable Name	Description
wFormatTag	ID which indicates the type of CODEC. Codec ID/Format Tag which is stored in Type-Specific Data for Stream Properties Object (Audio). (The normal value is 0x0161 for this middleware.)
nChannels	Number of channels. Number of Channels which is stored in Type-Specific Data for Stream Properties Object (Audio). (The normal value is 1 or 2 for this middleware.)
nSamplesPerSec	Sampling frequency (Hz) Samples Per Second which is stored in Type-Specific Data for Stream Properties Object (Audio). (The normal value is 8000, 11025, 16000, 22050, 32000, 44100, or 48000 for this middleware.)
nAvgBytesPerSec	Average number of bytes per second Average Number of Bytes Per Second which is stored in Type-Specific Data for Stream Properties Object (Audio). (The normal value is 16 or more for this middleware. It is less than the nSamplesPerSec value multiplied by 2.)
nBlockAlign	Size (in bytes) of a data block Block Alignment which is stored in Type-Specific Data for Stream Properties Object (Audio). (The normal value is 1 or more for this middleware.)
nValidBitsPerSample	Number of bits per sample Bits Per Sample which is stored in Type-Specific Data for Stream Properties Object (Audio). (The normal value is 16 for this middleware.)
codecSpecificData	Codec Specific Data which is stored in Type-Specific Data for Stream Properties Object (Audio). (This middleware can process any values as normal values.)
reserve	Reserved value (which should be 0)

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

[Remarks2] For information about Stream Properties Object, refer to Section 3.3 “Stream Properties Object (mandatory, one per stream)” of the ASF Specifications (related document).
For information about Type-Specific Data, refer to Section 9.1 “Audio Media Type” of the ASF Specifications (related document).

Table 3.2 codecSpecificData[18] Settings

Offset (in bytes)	Type	Size (in bytes)	Setting
0	ACMW_UINT8	1	Codec Specific Data (1st element)
1	ACMW_UINT8	1	Codec Specific Data (2nd element)
2	ACMW_UINT8	1	Codec Specific Data (3rd element)
3	ACMW_UINT8	1	Codec Specific Data (4th element)
4	ACMW_UINT8	1	Codec Specific Data (5th element)
5	ACMW_UINT8	1	Codec Specific Data (6th element)
6	ACMW_UINT8	1	Codec Specific Data (7th element)
7	ACMW_UINT8	1	Codec Specific Data (8th element)
8	ACMW_UINT8	1	Codec Specific Data (9th element)
9	ACMW_UINT8	1	Codec Specific Data (10th element)
10	ACMW_UINT8	1	Reserved area (arbitrary value)
11	ACMW_UINT8	1	Reserved area (arbitrary value)
12	ACMW_UINT8	1	Reserved area (arbitrary value)
13	ACMW_UINT8	1	Reserved area (arbitrary value)
14	ACMW_UINT8	1	Reserved area (arbitrary value)
15	ACMW_UINT8	1	Reserved area (arbitrary value)
16	ACMW_UINT8	1	Reserved area (arbitrary value)
17	ACMW_UINT8	1	Reserved area (arbitrary value)

3.2.5 Initialization Results Structure

[Structure name] wmastd_initStatusInfo

[Function] This structure stores the results of the initialization process (wmastd_Init).

[Prototype]

```
typedef struct {
        ACMW_UINT16     nChannels;
        ACMW_UINT32     nSamplesPerSec;
        ACMW_INT32      reserve;
} wmastd_initStatusInfo;
```

[Member description]

Member Variable Name	Description
nChannels	Number of channels for output data
nSamplesPerSec	Sampling frequency (Hz) for output data
reserve	Reserved

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

3.2.6 Data Input Settings Structure

[Structure name] wmastd_audecGetDataConfigInfo

[Function] This structure specifies the processing conditions when the wmastd_AudecGetData function inputs data.

[Prototype]

```
typedef struct {
        ACMW_BOOL         bNewPacket;
        ACMW_UINT8 *      pInDataStart
        ACMW_UINT32       nInDataSize
        ACMW_INT32        reserve;
} wmastd_audecGetDataConfigInfo;
```

[Member description]

Member Variable Name	Description
bNewPacket	Flag indicating whether the input data is a new data block. 0 : The input data is not a new data block. other : The input data is a new data block.
pInDataStart	Start address of the input data
nInDataSize	Size (in bytes) of the input data
reserve	Reserved value (which should be 0)

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

3.2.7 Data Input Results Structure

[Structure name] wmastd_audecGetDataStatusInfo

[Function] This structure stores the results of data input performed by the wmastd_AudecGetData function.

[Prototype]

```
typedef struct {
        ACMW_INT32         reserve;
} wmastd_audecGetDataStatusInfo;
```

[Member description]

Member Variable Name	Description
reserve	Reserved

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

3.2.8 Decode Settings Structure

[Structure name] wmastd_audioDecodeConfigInfo

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

[Prototype]

```
typedef struct {
        ACMW_INT32          reserve;
} wmastd_audioDecodeConfigInfo;
```

[Member description]

Member Variable Name	Description
reserve	Reserved value (which should be 0)

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

3.2.9 Decode Results Structure

[Structure name] wmastd_audioDecodeStatusInfo

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

[Prototype]

```
typedef struct {
        ACMW_BOOL       bNoMoreInput;
        ACMW_UINT32     nDecodeSamples;
        ACMW_INT32      reserve;
} wmastd_audioDecodeStatusInfo;
```

[Member description]

Member Variable Name	Description
bNoMoreInput	Flag indicating whether there is data to be decoded. 0 : There is data to be decoded. 1 : There is not data to be decoded. other : Reserved
nDecodeSamples	Number of samples which can be generated per channel by executing the PCM data generation process (wmastd_ReconstructPcmData function).
reserve	Reserved

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

3.2.10 PCM Data Generation Settings Structure

[Structure name] wmastd_reconstructPcmDataConfigInfo

[Function] This structure specifies the processing conditions when the wmastd_ReconstructPcmData function generates PCM data.

[Prototype]

```
typedef struct {
        ACMW_UINT32       nRequestSamples;
        void **           pOutBuffStart;
        ACMW_UINT32       nOutBuffSize;
        ACMW_UINT16       nOutBuffFormat;
        ACMW_INT32        reserve;
} wmastd_reconstructPcmDataConfigInfo;
```

[Member description]

Member Variable Name	Description
nRequestSamples	Number of samples which should be generated per channel by executing the PCM data generation process (wmastd_ReconstructPcmData function).
pOutBuffStart	Start address of the output data per channel
nOutBuffSize	Size (in bytes) of the output buffer per channel
nOutBuffFormat	Data format for the output buffer 0 : Interleaved format (16 bits/sample) other : Non-interleaved format (32 bits/sample)
reserve	Reserved value (which should be 0)

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

[Remarks2] pOutBuffStart stores the start address of a pointer array for output buffers for output channels reserved by the user.
If you select a non-interleaved format with nOutBuffFormat, use the output buffer pointer to specify the address for alignment which enables access to 4-byte data in memory.
If you select an interleaved format with nOutBuffFormat, use pOutBuffStart[0] to specify the start address of the output buffer. In this case, alignment is not restricted.

[Remarks3] The nOutBuffSize value is the same for all channels.

3.2.11 PCM Data Generation Results Structure

[Structure name] wmastd_reconstructPcmDataStatusInfo

[Function] This structure stores the results of PCM data generation performed by the wmastd_ReconstructPcmData function.

[Prototype]

```
typedef struct {
    ACMW_UINT32     nReconstructedSamples;
    ACMW_UINT32     nChannels;
    void **         pOutBuffLast;
    ACMW_UINT32     nOutBuffUsedDataSize;
    ACMW_UINT16     nValidBitsPerSample;
    ACMW_UINT32     nChannelMask;
    ACMW_INT32      reserve;
} wmastd_reconstructPcmDataStatusInfo;
```

[Member description]

Member Variable Name	Description
nReconstructedSamples	Number of samples which are generated per channel by executing the PCM data generation process (wmastd_ReconstructPcmData function).
nChannels	Number of output channels
pOutBuffLast	Post-write address of the data which is output per channel by the PCM data generation process.
nOutBuffUsedDataSize	Consumed data size (in bytes) of an output buffer for each channel after the PCM data generation process
nValidBitsPerSample	Number of bits per sample
nChannelMask	Output channel information (see Table 3.3)
reserve	Reserved

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

[Remarks2] For pOutBuffLast, specify the start address of a user-reserved pointer array before calling the wmastd_ReconstructPcmData function.

[Remarks3] If you select 0 for member nOutBuffFormat of the PCM data generation settings structure, pOutBuffLast[0] is set to the post-write address of an output buffer and nOutBuffUsedDataSize to the consumed data size of an output buffer.
If you select a non-zero value for member nOutBuffFormat of the PCM data generation settings structure, pOutBuffLast[0 to nChannels] is set to the post-write address of an output buffer for each channel and nOutBuffUsedDataSize to the consumed data size of an output buffer for each channel.

Table 3.3 nChannelMask Bit Fields

Bit field	Numerical value	Meaning
bit 0	0x00000001	Front left channel
bit 1	0x00000002	Front right channel
bit 2	0x00000004	Front center channel/monaural channel
bit 3	0x00000008	Low Frequency Effect (LFE) channel (not used)
bit 4	0x00000010	Back left channel (not used)
bit 5	0x00000020	Back right channel (not used)
bit 6	0x00000040	Front left center channel (not used)
bit 7	0x00000080	Front right center channel (not used)
bit 8	0x00000100	Back center channel (not used)
bit 9	0x00000200	Side left channel (not used)
bit 10	0x00000400	Side right channel (not used)
bit11	0x00000800	Upper center channel (not used)
bit 12	0x00001000	Upper front left channel (not used)
bit 13	0x00002000	Upper front center channel (not used)
bit 14	0x00004000	Upper front right channel (not used)
bit 15	0x00008000	Upper back left channel (not used)
bit 16	0x00010000	Upper back center channel (not used)
bit 17	0x00020000	Upper back right channel (not used)
bit 18	0x00040000	Reserved
bit 19	0x00080000	Reserved
bit 20	0x00100000	Reserved
bit 21	0x00200000	Reserved
bit 22	0x00400000	Reserved
bit 23	0x00800000	Reserved
bit 24	0x01000000	Reserved
bit 25	0x02000000	Reserved
bit 26	0x04000000	Reserved
bit 27	0x08000000	Reserved
bit 28	0x10000000	Reserved
bit 29	0x20000000	Reserved
bit 30	0x40000000	Reserved
bit 31	0x80000000	Reserved

3.3 Error Processing

This middleware’s functions return the error codes listed in Table 3.4. To obtain details about the cause of an error, execute the wmastd_GetErrorFactor function.

3.3.1 Error codes

Below are the error codes for this middleware.

Table 3.4 Error Codes

Error code (32bit)	Value	Description	Reinitialization
WMASTD_RESULT_OK	0x00000000	The processing results are normal. The process has terminated normally.	Unnecessary
WMASTD_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
WMASTD_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 (wmastd_GetErrorFactor function).	Unnecessary
WMASTD_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 wmastd_GetErrorFactor function.	Necessary
Others	Other than the above	Reserved	-

Table 3.5 Error Codes Used by the Library Functions

Function / Error code	wmastd_GetMemorySize	wmastd_Init	wmastd_AudecGetData	wmastd_AudioDecode	wmastd_ReconstructPcmData	wmastd_GetErrorFactor [*1]	wmastd_GetVersion [*2]
WMASTD_RESULT_OK	o	o	o	o	o	-	-
WMASTD_RESULT_NG	-	o	o	o	o	-	-
WMASTD_RESULT_WARNING	-	-	o	o	o	-	-
WMASTD_RESULT_FATAL	o	o	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 wmastd_GetErrorFactor function except when WMASTD_RESULT_FATAL occurs. Table 3.6 lists the error factors. Table 3.7 shows the relationship of the library functions to the error factors and error codes.

Table 3.6 Error Factors

errorFactor(32bit)	Value	Meaning
WMASTD_ERR_NONE	0x00000000	The program has normally terminated. No error factor is available.
WMASTD_ERR_POINTER	0x00000010	Invalid pointer value
WMASTD_ERR_PARAMETER	0x00000020	Invalid parameter
WMASTD_ERR_SEQUENCE	0x00000040	The library functions were executed in invalid order.
WMASTD_ERR_LOST_PACKET	0x00000100	Lack of input data was detected.
WMASTD_ERR_BROKEN_FRAME	0x00000200	An abnormal input data structure was detected.
WMASTD_ERR_NOT_SUPPORTED_DATA	0x00010000	Unsupported data was detected.
WMASTD_ERR_BAD_ARGUMENT	0x01000000	An invalid argument for the internal function was detected.
WMASTD_ERR_MW_FAILED	0x02000000	An internal abnormality was detected.
WMASTD_ERR_OUT_OF_MEMORY	0x04000000	An invalid amount of memory used was detected.
WMASTD_ERR_WRONG_STATE	0x08000000	An invalid internal state was detected.
Others	Other than the above	Reserved

Table 3.7 Relationship of the Library Functions to the Error Factors and Error Codes

Function Error factor	wmastd_Init	wmastd_AudecGetData	wmastd_AudioDecode	wmastd_ReconstructPcmData
WMASTD_ERR_NONE	OK	OK	OK	OK
WMASTD_ERR_POINTER	NG	NG	NG	NG
WMASTD_ERR_PARAMETER	NG	-	-	NG
WMASTD_ERR_SEQUENCE	-	NG	NG	NG/WARNING
WMASTD_ERR_LOST_PACKET	NG	WARNING	-	NG
WMASTD_ERR_BROKEN_FRAME	NG	NG	NG	NG
WMASTD_ERR_NOT_SUPPORTED_DATA	NG	NG	-	NG
WMASTD_ERR_BAD_ARGUMENT	NG	NG	-	NG
WMASTD_ERR_MW_FAILED	NG	NG	WARNING	NG
WMASTD_ERR_OUT_OF_MEMORY	NG	NG	-	NG
WMASTD_ERR_WRONG_STATE	NG	NG	-	NG

[Note] The letters in the table above indicate error codes as follows.

“OK”: WMASTD_RESULT_OK

“NG”: WMASTD_RESULT_NG

“WARNING”: WMASTD_RESULT_WARNING

3.4 Memory Specifications

This section describes the memory areas used by this middleware.

3.4.1 Scratch Area

Table 3.8 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 wmastd_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 3.9 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 wmastd_GetMemorySize.
Area reservation	The user should reserve this area.
Allocation	This area is included in RAM.
Alignment	Align this area on an 8-byte boundary.

3.4.3 Software Stack Area

Table 3.10 Software Stack Area Description

Item	Stack area used by this middleware
Symbol name	- (freely defined by the user)
Size	Obtain the actually required size with wmastd_GetMemorySize.
Area reservation	The user should reserve this area. To use this middleware, reserve a software 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

Table 3.11 Input Buffer Description

Item	Area which stores inputs to this middleware. The input buffer contains data blocks extracted from WMA Bitstream by the parser. 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	Size is freely defined by the user. This size should not be less than 4 bytes. The recommended size is equal to the nInputBufferSize value obtained by this middleware's wmastd_GetMemorySize function. Specify the size which suits the target system.
Area reservation	The user should reserve this area. The user can freely use this area when the bNoMoreInput value in the decode results structure is 1 after the wmastd_AudioDecode function is executed.
Allocation	This area is included in RAM.
Alignment	Alignment is not restricted.

When executing the wmastd_AudecGetData function, specify the parameters in the data input settings structure. This stores the processing results into the buffer memory results structure. Figure 3.1 shows the relationship between the input buffer and structure members.

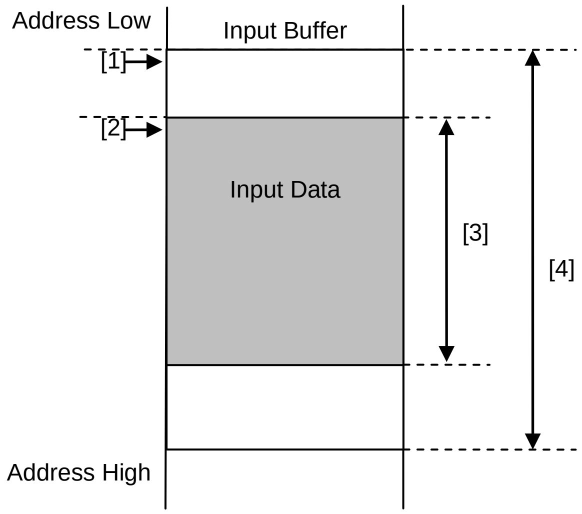


Figure 3.1 Structure Members in the Input Buffer

[Note] Data blocks are placed into the input buffer for decode processing.

Table 3.12 Structure Members in the Input Buffer

[1]	-	-	Input buffer start address
[2]	pInBuffStart	(Data input settings structure)	Input data start address
[3]	nInDataSize	(Data input settings structure)	Input data size
[4]	-	-	Input buffer size

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

(1) Input data storage method

Figure 3.2 shows how input data is stored. Store data given in bytes into the input buffer (memory).

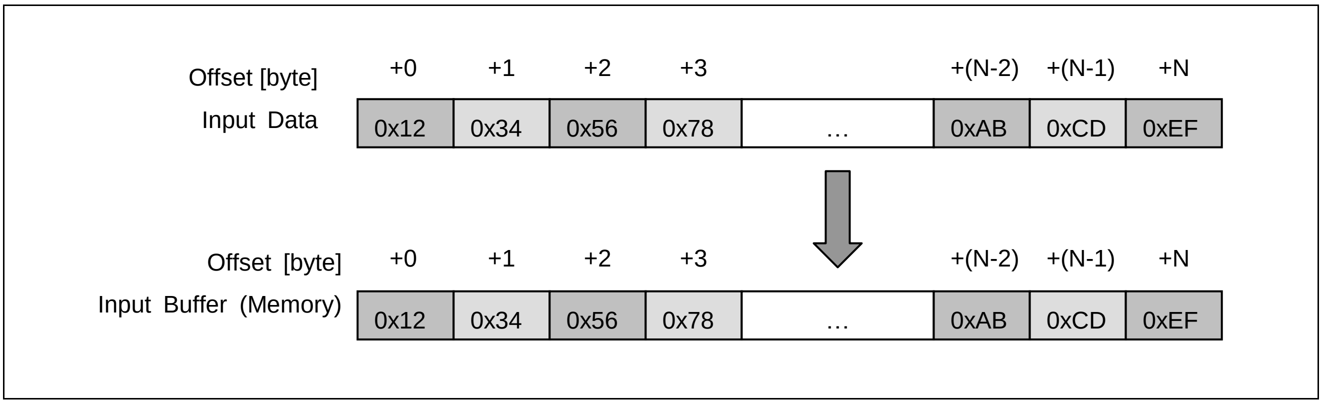


Figure 3.2 Input Data Storage Method

(2) Data input method

Input Payload Data or Sub-Payload #N Data (N = 0, 1, etc.) for each data block. The input data should contain the encoding results generated by WMA Standard CODEC.

When you input data blocks, store 4 or more bytes of data into the input buffer (except when the data block size is less than 4 bytes). (See Section 3.1.3.)

Section 3.5 describes the relationship of Payload Data or Sub-Payload #N Data to data blocks.

When Input Buffer Size less than Data Block Size:

You can split one data block into several portions and input them separately one after each other.

Store the maximum possible amount of data into the input buffer. Set bNewPacket (a member of the input data settings structure) to 1 only when switching one data block to another.

When Input Buffer Size not less than Data Block Size:

You can input one data block at a time.

Store one data block into the input buffer and always set bNewPacket (a member of the input data settings structure) to 1.

3.4.6 Output Buffer

Table 3.13 Output Buffer Description

Item	Area which stores outputs from this middleware. The output buffer contains 16-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.
Section name	- (freely defined by the user)
Symbol name	- (freely defined by the user)
Size	Size is freely defined by the user. This size should not be less than 4 bytes. The recommended size is equal to the nOutputBufferSize value obtained by this middleware's wmastd_GetMemorySize function. Specify the size which suits the target system. Normally, The nDecodedSamples value in decode result structure is 4,096 or less. If you reserve output buffer the size of 4,096 samples or more, could be generated all obtainable PCM data at a time. [Note] The size obtained by the wmastd_GetMemorySize function is equal to the size per channel in non-interleaved format.
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 non-interleaved format, align this area on a 4-byte boundary. If you output data in interleaved format, alignment is not restricted.

When executing the wmastd_ReconstructPcmData function, specify the parameters in the PCM data generation settings structure. This stores the processing results into the PCM data generation settings structure. Figure 3.3 shows the relationship between the output buffer and structure members.

If you output data in non-interleaved format, 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. The number of samples output to the output buffer varies depending on the input data and decode conditions. Refer to the description of nReconstructedSamples of the PCM data generation results structure.

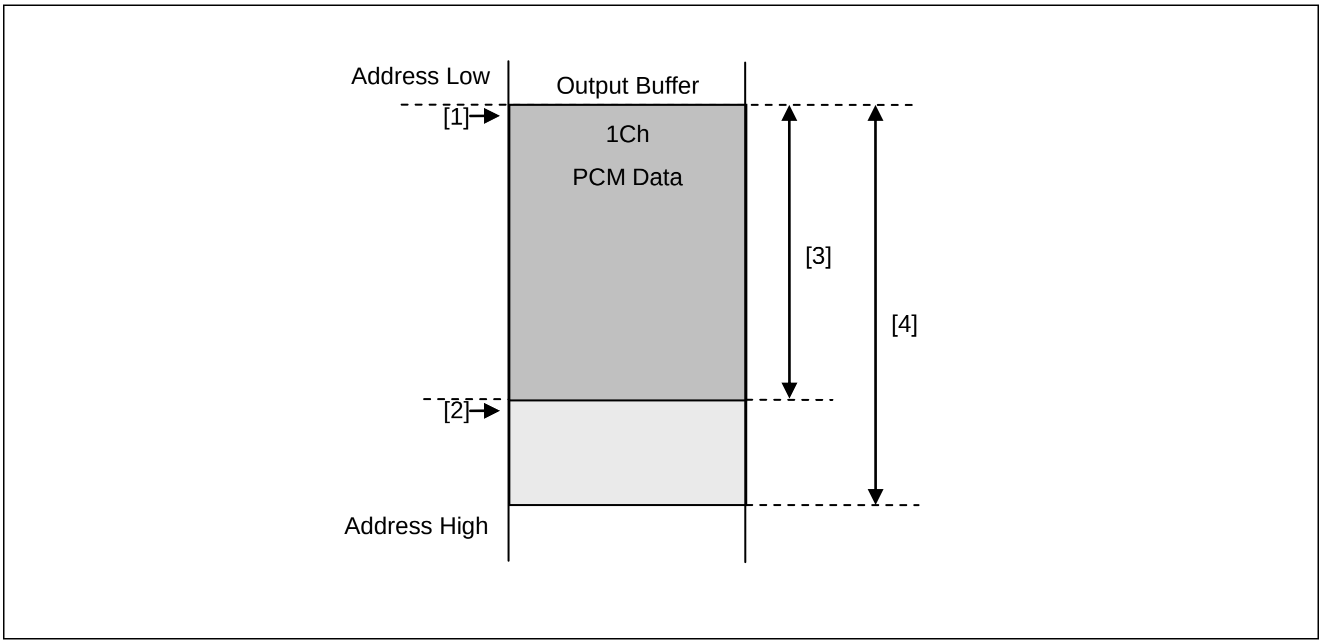


Figure 3.3 Structure Members in the Output Buffer

[Note] Output results are placed into the output buffer for PCM data generation.

Table 3.14 Structure Members in the Output Buffer

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

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

(1) Output data storage method (interleaved format)

Data is output as shown in Figure 3.5. The output buffer (memory) stores data in 2-byte (16-bit) units. The byte order for accessing the buffer is little endian (see Figure 3.4).

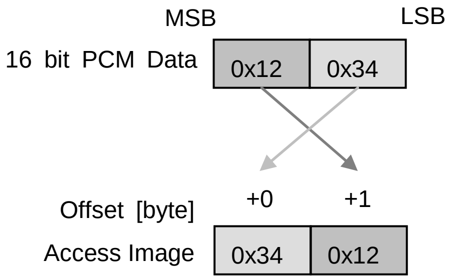
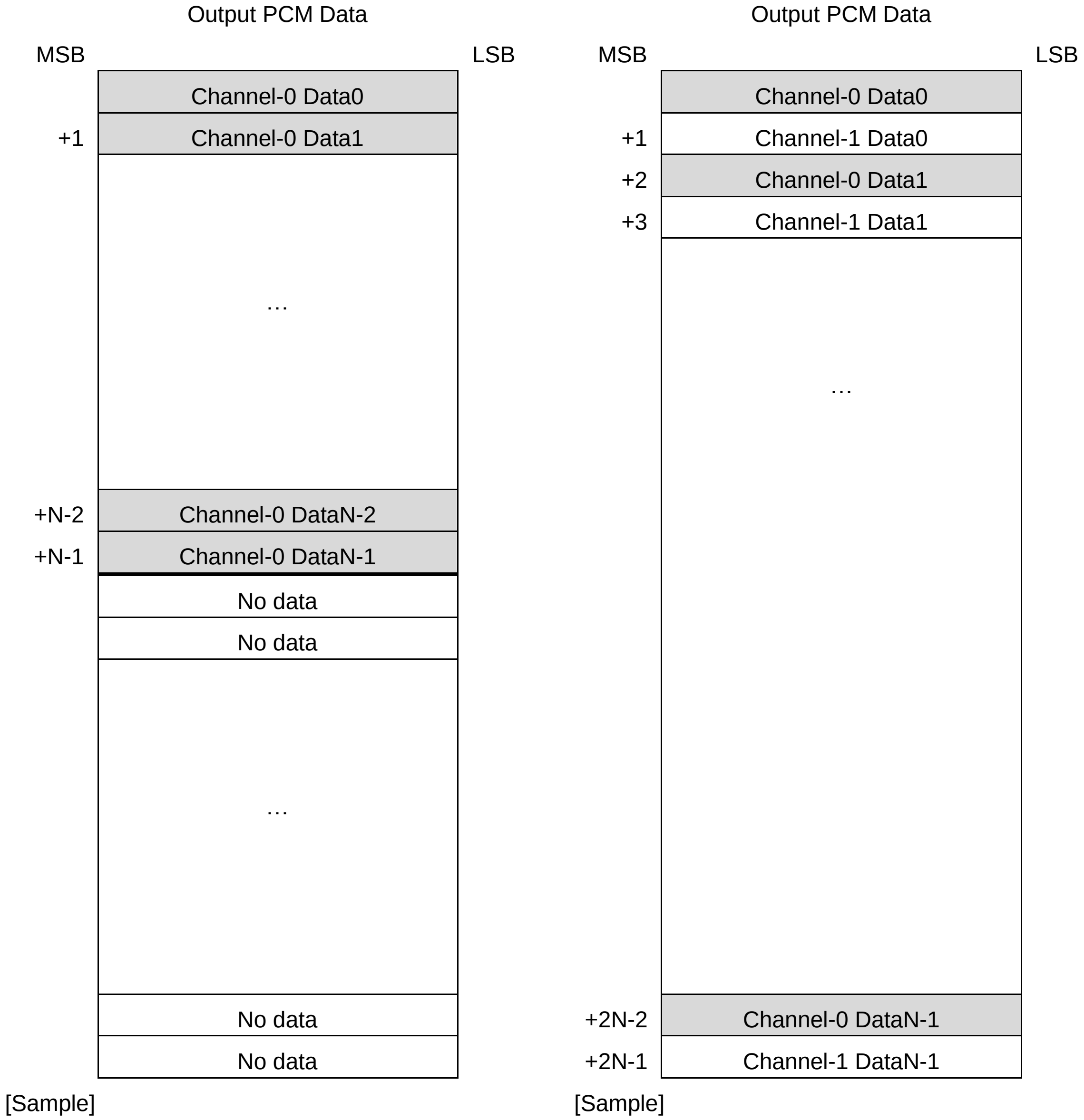


Figure 3.4 16-bit PCM Data Access (Little Endian Mode)



Mono Output Format

Stereo Output Format

Figure 3.5 Output PCM Data (Interleaved Format)

(2) Output data storage method (non-interleaved format)

Data for each channel is output as shown in Figure 3.8. The output buffer (memory) stores 4-byte (32-bit) data in the format shown in Figure 3.7. The byte order for accessing the buffer is little endian (see Figure 3.6).

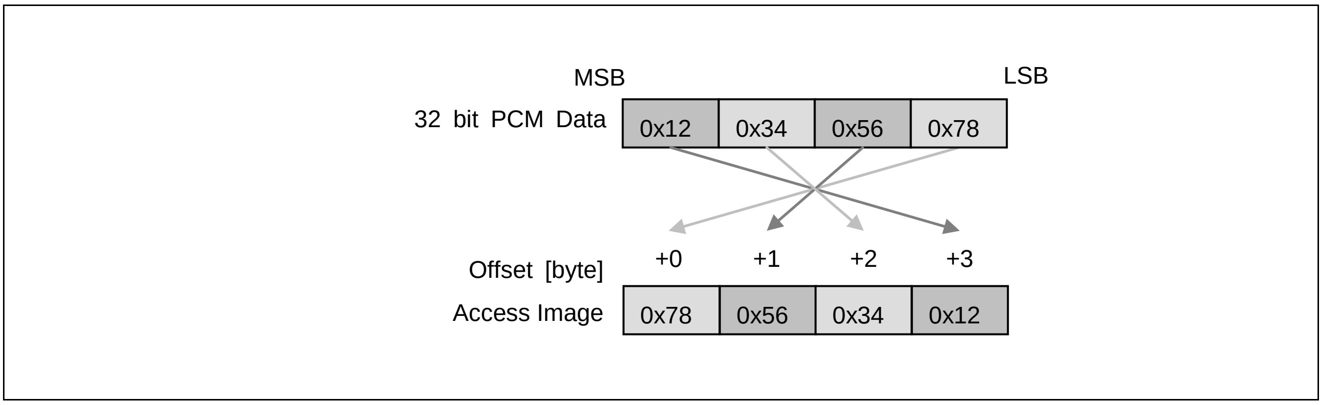


Figure 3.6 32-bit PCM Data Access (Little Endian Mode)

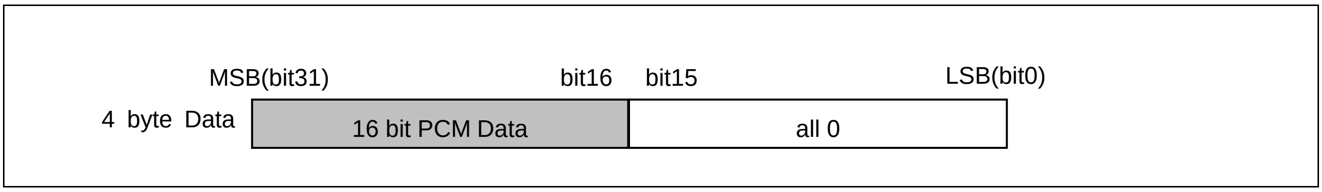
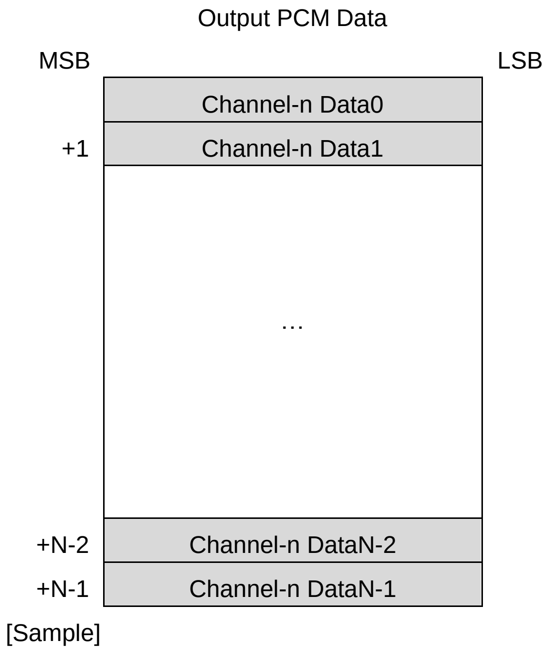


Figure 3.7 4-byte PCM Data Bit Positions



1ch Output Format

Figure 3.8 Output PCM Data (Non-interleaved Format)

[Note] The buffers for the channels can be allocated to inconsecutive addresses.

3.5 Input Data

Input Payload Data or Sub-Payload #N Data (N = 0, 1, etc.) for each data block to this middleware. The input data should contain the encoding results generated by WMA Standard CODEC.

For information about the input data storage and data input methods, refer to Section 3.4.5.
The relationship of Payload Data or Sub-Payload #N Data to data blocks is described below.

For further details about Payload Data or Sub-Payload #N Data, refer to the ASF Specifications (related document) released by Microsoft. Table 3.17 shows the relevant sections of the ASF Specifications.

Table 3.15 Sections Relevant to Input Data

Input data	Relevant section of ASF Specifications
Payload Data	5.2.3.1 Single payload 5.2.3.3 Multiple payloads
Sub-Payload #N Data	5.2.3.2 Single payload, compressed payload data 5.2.3.4 Multiple payload, compressed payload data

Relationship of Payload Data or Sub-Payload #N Data to Data Blocks

Payload Data or Sub-Payload #N Data consists of one or more data blocks. The size of each data block does not vary among files. This size is specified in Block Alignment stored in Type-Specific Data for Stream Properties Object (Audio).

For information about Stream Properties Object, refer to the ASF Specifications (related document) released by Microsoft.

3.6 Output Data

16-bit linear PCM data is output.

For a description of the output data storage method, refer to Section 3.4.6.

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) wmastd_GetMemorySize function

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

(2) wmastd_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) wmastd_AudecGetData function

Execute this function when you input data blocks to the middleware.

(4) wmastd_AudioDecode function

Execute this function when you decode an input data block.

(5) wmastd_ReconstructPcmData function

Execute this function when you create PCM data.

(6) wmastd_GetErrorFactor function

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

(7) wmastd_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	WMA Decode Middleware User’s Manual

Rev.	Date	Description	
		Page	**Summary**
1.00	July 31, 2014	-	First Edition issued
1.01	Aug. 29, 2014	3	Changed the format of “Table 1.2 Memory Size Requirements”
		52	Added Monitoring on the Performance in Section 4.2.4

WMA Decode Middleware User's Manual

Publication Date: August 29, 2014 Rev.1.01

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.
2880 Scott Boulevard Santa Clara, CA 95050-2554, 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: +44-1628-651-700, Fax: +44-1628-651-804

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, 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 Rd., Putuo District, Shanghai, China
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: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, 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-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 2.0

WMA Decode Middleware

Renesas Electronics Corporation