To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. 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.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. 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.
5. When exporting the 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. You should not use Renesas Electronics products or the 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. 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.
6. 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.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. 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 categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. 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 an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "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.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. 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.
9. 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 system manufactured by you.
10. 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.
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.

User's Manual

μSAP77016-B08

AAC Audio Decoder Middleware

Target Device

μPD77110
μPD77113A
μPD77114
μPD77115
μPD77210
μPD77213

Document No. U15152EJ2V0UM00 (2nd edition)
Date Published October 2002 NS CP(N)

[MEMO]

Windows is either a trademark or registered trademark of Microsoft Corporation in the United States and/or other countries.

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics Inc. (U.S.)
Santa Clara, California
Tel: 408-588-6000
800-366-9782
Fax: 408-588-6130
800-729-9288

NEC do Brasil S.A.
Electron Devices Division
Guarulhos-SP, Brasil
Tel: 11-6462-6810
Fax: 11-6462-6829

NEC Electronics (Europe) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 01
Fax: 0211-65 03 327

- **Sucursal en España**
  Madrid, Spain
  Tel: 091-504 27 87
  Fax: 091-504 28 60

- **Succursale Française**
  Vélizy-Villacoublay, France
  Tel: 01-30-67 58 00
  Fax: 01-30-67 58 99

- **Filiale Italiana**
  Milano, Italy
  Tel: 02-66 75 41
  Fax: 02-66 75 42 99

- **Branch The Netherlands**
  Eindhoven, The Netherlands
  Tel: 040-244 58 45
  Fax: 040-244 45 80

- **Branch Sweden**
  Taeby, Sweden
  Tel: 08-63 80 820
  Fax: 08-63 80 388

- **United Kingdom Branch**
  Milton Keynes, UK
  Tel: 01908-691-133
  Fax: 01908-670-290

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Shanghai, Ltd.
Shanghai, P.R. China
Tel: 021-6841-1138
Fax: 021-6841-1137

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-2719-2377
Fax: 02-2719-5951

NEC Electronics Singapore Pte. Ltd.
Novena Square, Singapore
Tel: 253-8311
Fax: 250-3583

J02.4

Major Revisions in This Edition

Page	Description
p.13	Change of the product name μPD77113 of **1.3.3 (1) Operable DSPs** to μPD77113A. Addition of the μPD77210 and 77213.
p.13	Change of the size in **Table1-3 Required Memory Sizes**.
p.20	Modification of description of Arguments R0,R5 in **2.2.1 a2d_InitDec function**
p.23	Modification of description of Arguments R0 and Hardware resourcement in **2.2.3 a2d_Dec function**.
p.25	Modification of description of Return value R0 in **2.2.5 a2d_GetStatus function.**
p.26	Modification of description of Function table in **2.2.6 a2d_GetErrorStatus function.**
p.27	Modification of description of Argument, Return value and Registers used in **2.2.7 aac_dec_fillbits function.**
p.28	Modification of description **2.2.8 aac_read_1word function.**
p.38 to 45	Modification of description in **APENDIX SAMPLE PROGRAM SOURCE** • **sample.asm** a2d.h → a2d_dec.h, a2d_errh.h→a2d_err.h • **sam_call.asm** replaced • **sam_deta.asm** a2d.h → a2d_dec.h • **sam_int.asm** a2d.h → a2d_dec.h, a2d_errh.h→a2d_err.h

The mark ★ shows major revised points.

PREFACE

Target Readers

This manual is for users who design and develop μPD77016 Family application systems.

μPD77016 Family is the generic name for the μPD7701x family (μPD77015, 77016, 77017, 77018A, 77019), the μPD77111 Family (μPD77110, 77111, 77112, 77113A, 77114, 77115) and the μPD77210 Family (μPD77210, 77213). However, this manual is for μPD77110, 77113A, 77114, 77115, 77210 and 77213 devices.

Purpose

The purpose of this manual is to help users understand the supporting middleware when designing and developing μPD77016 Family application systems.

Organization

This manual consists of the following contents.

CHAPTER 1 OVERVIEW
CHAPTER 2 LIBRARY SPECIFICATIONS
CHAPTER 3 INSTALLATION
CHAPTER 4 SYSTEM EXAMPLES
APPENDIX SAMPLE PROGRAM SOURCE

How to Read This Manual

It is assumed that the reader of this manual has general knowledge in the fields of electrical engineering, logic circuits, microcontrollers, and the C language.

To learn about μPD77111 Family hardware functions
→ Refer to **μPD77111 Family User's Manual Architecture**.

To learn about μPD77210 Family hardware functions
→ Refer to **μPD77210 Family User's Manual Architecture**.

To learn about μPD77016 Family hardware functions
→ Refer to **μPD77016 Family User's Manual Instruction**.

Conventions

Data significance:	Higher digits on the left and lower digits on the right
Active low representation:	$\overline{XXX}$ (overscore over pin or signal name)
Note:	Footnote for item marked with **Note** in the text
Caution:	Information requiring particular attention
Remark:	Supplementary information
Numerical representation:	Binary ... XXXX or 0bXXXX
	Decimal ... XXXX
	Hexadecimal ... 0xXXXX

Related Documents The related documents indicated in this publication may include preliminary versions. However, preliminary versions are not marked as such.

Documents Related to Devices

Document Name / Name Part Number	Pamphlet	Data Sheet	User's Manual		Application Note
			Architecture	Instructions	Basic Software
μPD77110	U12395E	U12801E	U14623E	U13116E	U11958E
μPD77111					
μPD77112					
μPD77113		U14373E			
μPD77114					
μPD77115		U14867E			
μPD77210		U15203E	U15807E		
μPD77213					

Documents Related to Development Tools

Document Name		Document No.
RX77016 User's Manual	Function	U14397E
	Configuration Tool	U14404E
RX77016 Application Note	HOST API	U14371E

Caution The related documents listed above are subject to change without notice. Be sure to use the latest version of each document for designing.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 OVERVIEW

1.1 Middleware

Middleware is the name given to a group of software that has been tuned so that it draws out the maximum performance of the processor and enables processing that is conventionally performed by hardware to be performed by software.

The concept of middleware was introduced with the development of a new high-speed processor, the DSP, in order to facilitate operation of the environments integrated in the system.

By providing appropriate speech codec and image data compression/decompression-type middleware, NEC is offering users the kind of technology essential in the realization of a multimedia system for the μPD77016 Family, and is continuing its promotion of system development.

μSAP77016-B08 is middleware that provides AAC-technology decoding functions.

1.2 AAC Audio Decoder

AAC is a form of audio coding for multiple channels (up to 64 channels) that achieves high quality and high compression rates by excluding compatibility with MPEG-1 audio. The relevant standard is ISO/IEC 13818-7.

In this user's manual, AAC is an abbreviation for Advanced Audio Coding.

μSAP77016-B08 complies with this AAC-technology decoding method. The compressed data is data that encodes a digital signal converted to 16-bit linear PCM data after sampling an analog signal at frequencies shown in Table 1-1.

Note that μSAP77016-B08 decodes and outputs at most two front channels.

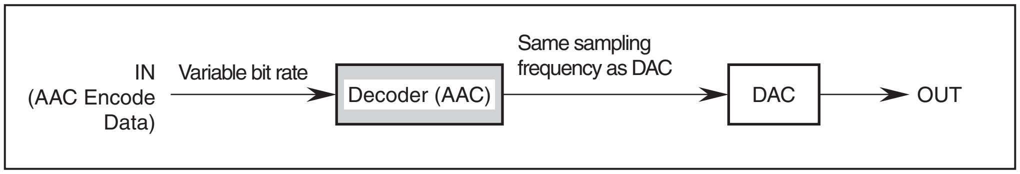
Table 1-1. Sampling Frequencies

Frequency [Hz]
8000
11025
12000
16000
22050
24000
32000
44100
48000
64000
88200
96000

1.2.1 Decoder outline

Figure 1-1 is a sample configuration of decoding using μSAP77016-B08.

Figure 1-1. Sample Configuration of Decoding



(1) AAC Encode Data

AAC Encode Data is data in which sampled 16-bit linear PCM data is encoded. For sampling frequencies, refer to **Table 1-1 Sampling Frequencies**. The maximum value of the bit rate differs according to the sampling frequency. The bit rate can have any value up to the maximum value. Table 1-2 shows the maximum values of bit rate.

Table 1-2. Maximum Bit Rates

Sampling Frequency [Hz]	Maximum Bit Rate [kbps/ch]
8000	48
11025	66.15
12000	72
16000	96
22050	132.3
24000	144
32000	192
44100	264.6
48000	288
64000	384
88200	529.2
96000	576

(2) AAC decoder (Decoder(AAC))

The AAC decoder (Decoder(AAC)) reads input data, performs decoding, and outputs 16-bit linear PCM data.
It decodes and outputs at most two front channels.
In ADTS format, it performs CRC processing as error compensation at the end.

(3) DAC

The DAC converts 16-bit linear PCM data to an analog signal.
The DAC must operate at the sampling frequency attached to the encoded data.
If the sampling frequency of the encoded data differs from the DAC, separate frequency conversion software (such as a rate converter) is needed.

1.3 Product Overview

1.3.1 Features

- Employs AAC decoder algorithm standardized by ISO/IEC
- Bit rates correspond to maximum bit rates shown in **Table 1-2 Maximum Bit Rates**
- Input data is encoded 16-bit linear PCM data sampled at a sampling frequency (**Table 1-1 Sampling Frequencies**)
- Output data is 16-bit linear PCM data of the same frequency as the sampling frequency of input data
- 1024 samples/frame/channel decoding
- Supports ADTS, ADIF, and RAW formats
- Provides CRC as error compensation (ADTS format only)
- Supports decoding of two front channels

1.3.2 Functions

(1) Decompression processing

Decompression processing converts compressed data to one RAW of 16-bit linear PCM data.

(2) Error compensation

CRC is performed as error compensation processing in ADTS format.

1.3.3 Operating environment

★ **(1) Operable DSPs:**

μPD77110, 77113A, 77114, 77115, 77210, 77213

(2) Required memory size:

μSAP77016-B0B requires memory sizes shown in the following table.

★ **Table 1-3. Required Memory Sizes**

Memory	Type		Size [kwords]
Instruction memory	–		8.6
X memory	RAM	Scratch area	2.0
		Static area	0.7
	ROM		4.0
Y memory	RAM	Scratch area	4.0
		Static area	4.1
	ROM		3.9

Caution One word of instruction memory is 32 bits.
One word of X memory or Y memory is 16 bits.

Scratch areas are memory areas that can be discarded when μSAP77016-B08 is not operating. The user should define scratch areas. A defined scratch area must be set using the a2d_InitDec function.

The user can use scratch areas when μSAP77016-B08 is not operating. However, caution is required when using these areas. Since μSAP77016-B08 will use these areas again when it operates, if a user has set information in scratch areas, the set information cannot be guaranteed.

Refer to Table 1-4 when using scratch areas.

Table 1-4. Scratch Areas

Memory	Public Symbol Name	Address	Size [words]
X memory	lib_Scratch_x	Specify at 0	2048
Y memory	lib_Scratch_y	Specify align at 0	4096

A static area is a memory area that cannot be discarded even when μSAP77016-B08 is not operating. A user cannot use a static area.

Besides the memories described in Table 1-3, input buffer (X memory) and output buffer (X memory) are required.

In the sample program described later, 1024 words are used as an input buffer and 2048 words as two output buffers, for a total of three buffers.

(3) Required D/A specs

D/A 2 ch, 16-bit resolution, shown in **Table 1-1 Sampling Frequencies**

(4) Software tools (Windows™ version)

Table 1-5. Software Tools

Relevant DSP	Software Tools
μPD77016 Family	DSP tools WB77016 (Workbench) HSM77016 (High-speed simulator)

1.3.4 Performance

Table 1-6 shows the MIPS values that are necessary in order to execute one frame processing in real time.

Measurement and Calculation Conditions DSP: μPD77110 (operating frequency: 75 MHz, 75 MIPS)
Sampling frequency: 44.1 kHz

Table 1-6. MIPS Needed in One Frame Decompression Processing

Measurement and Calculation Condition		Decompression Time [MIPS/ch]
Decompression	Logical maximum MIPS value[Note 1]	24.2
	Measured maximum MIPS value[Note 2]	13.5
	Measured average MIPS value[Note 2]	11.0
CRC (ADTS only)	Logical[Note 1] value	0.5

Notes 1. Logical means that the value is calculated by taking the maximum number of cycles for the number of loops, number of repeats, and algorithm processing route in a program.

2. Measured means that the MIPS value was measured by actually executing μSAP77016-B08 on a real machine and decompressing stereo (2-channel) encoded data at an average bit rate of 128 kbps.

1.3.5 Directory configuration

The directory configuration of μSAP77016-B08 is shown below.

<u>μPD77016 Family</u>

```
──┬─ library ──────────┬─ a2dram.lib:     Library for μPD77110, 77115, 77210
  │                    └─ a2drom.lib:     Library for μPD77113A, 77114, 77213
  └─ smp ─── aacdec ───┬─ a2d_dec.h:      Library header file (For decoder)
                       ├─ a2d_err.h:      Error status header file
                       ├─ sample.prj:     Project file
                       ├─ sample.h:       Header file for sample program
                       ├─ sample.asm:     Sample source file
                       ├─ sam_call.asm:   Source file for callback function
                       ├─ sam_data.asm:   Source file for data area
                       ├─ sam_int.asm:    Source file for interrupt servicing
                       ├─ sample.tmg:     Timing file
                       ├─ serclk.tmg:     Timing file for clocks input serially
                       └─ serout16.tmg:   Timing file for saving serial output in file
```

A summary of each directory is shown below.

(1) library

This directory contains library files.

(2) smp --- aacdec

This directory contains sample program source files and header files. It also provides timing files described later.

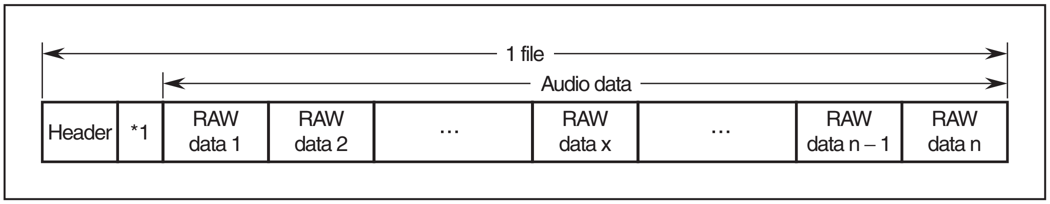
1.4 Compressed Data Format

For details of the compressed data format, refer to standards (**ISO/IEC 13818-7**).
μSAP77016-B08 specifications conform to standards.

1.4.1 ADIF format outline

Figure 1-2 shows the structure of ADIF format.

Figure 1-2. ADIF Format

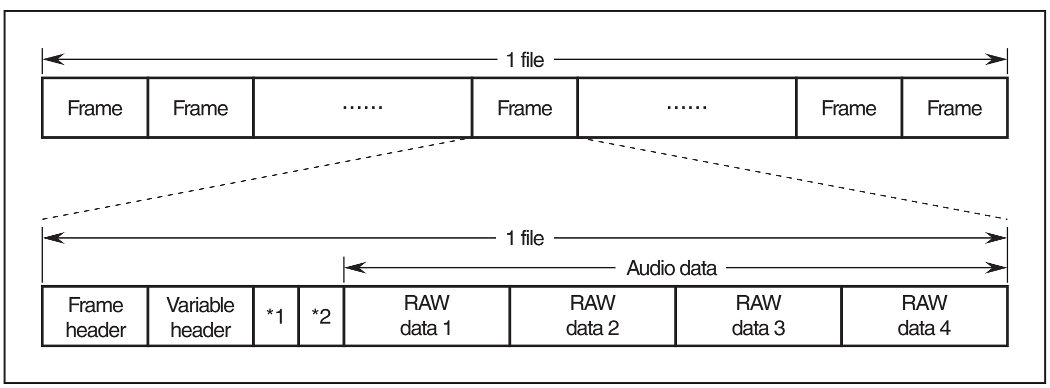


Remark Header: Contains information for synchronizing, such as sampling frequency, bit rate, and mode.
Audio data: This is information related to the audio sample. It consists of multiple RAW data bit streams.
RAW data: This is a bit stream of the smallest unit that is decoded.
*1: Bit alignment

1.4.2 ADTS format outline

Figure 1-3 shows the structure of ADTS format.

Figure 1-3. ADTS Format

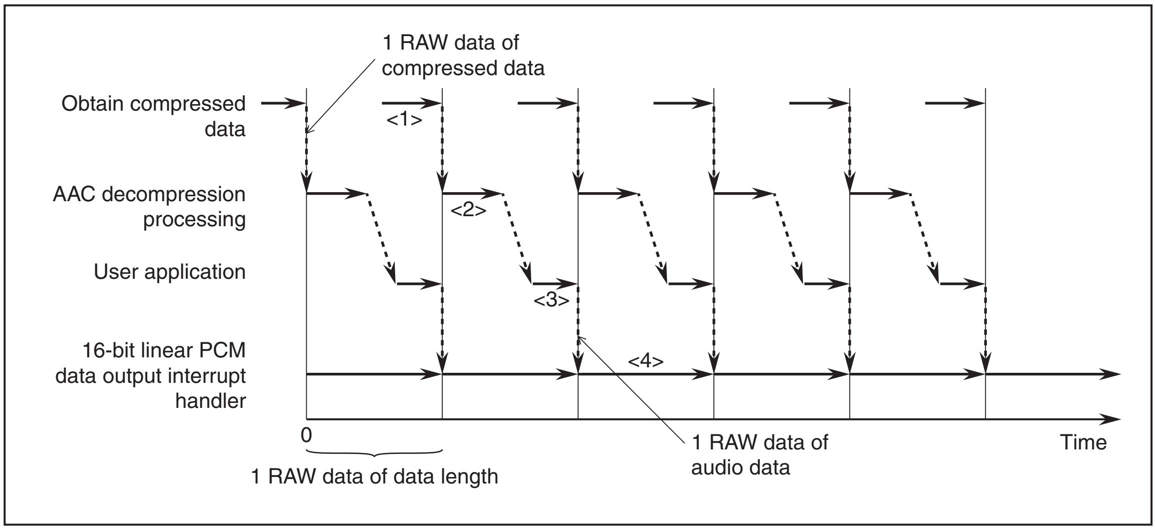


Remark Frame header: Contains information for synchronizing, such as sampling frequency, bit rate, and mode.
Variable header: This is information needed in decoding, such as the number of RAW data bit streams included in the audio data.
Audio data: This is information related to the audio sample. It consists of multiple RAW data bit streams.
RAW data: This is a bit stream of the smallest unit that is decoded. There are up to four in one frame.
*1: Error check
*2: Bit alignment

1.5 Timing Diagram

Figure 1-4 shows the timing diagram of Decoder.

Figure 1-4. Timing Diagram



<1> Read 1 RAW data of compressed data and pass it to decompression processing.

<2> Convert 1 RAW data of compressed data to 1 RAW data of decompressed data.

<3> Buffer decompressed data. Besides this, perform application processing.
If the sampling frequency differs from that of DAC, the user should execute Rate Conversion to convert to the same sampling frequency as DAC.

<4> Perform D/A conversion of 1 RAW data of 16-bit linear PCM data.

CHAPTER 2 LIBRARY SPECIFICATIONS

2.1 Library Overview

μSAP77016-B08 provides the following six functions.

Table 2-1. List of Library Functions

Function Name	Function
a2d_InitDec	Initialize decompression processing
a2d_set_adif_header	ADIF/RAW parameter initialization processing
a2d_Dec	Decompression processing
a2d_GetVersion	Obtain version information
a2d_GetStatus	Obtain status information
a2d_GetErrorStatus	Obtain error information

In addition, the functions in Table 2-2 must be defined to operate μSAP77016-B08. These functions need to be provided by the user.

Table 2-2. User-Defined Functions

Function Name	Function
aac_read_1word	Read frame header
aac_dec_fillbits	Obtain input data

2.2 Function Specifications

Specifications when calling each library function are shown below.

2.2.1 a2d_InitDec function

[Classification] AAC decoder initialization processing

[Function name] a2d_InitDec

[Summary of function] Initializes RAM areas and sets parameters used by μSAP77016-B08

[Format] call a2d_InitDec

[Arguments]

R0	Initialization mode		
	Bit 0	0: Decode RAW_DATA only	
		1: Decode format with header (ADIF/ADTS)	
	Bit 8	0: Normal	★
		1: Check header using aac_read_1word function on CRC error when ADTS decoding	
R1	Reserved		
R2	Reserved		
R3	Reserved		
R4	Address of callback function aac_read_1word		
R5	Address of callback function aac_dec_fillbits		★
R6	0x0000 fixed (Address of scratch area lib_Scratch_x)		
R7	Address of scratch area lib_Scratch_y		
DP0	Pointer to user-defined input buffer (see **Figure 2-1**)		

[Return value]

R0	0: Decoding error
	1: Decoded object was ADIF/RAW
	2: Decoded object was ADTS

[Function] Initialize RAM areas used by μSAP77016-B08 and set parameters.

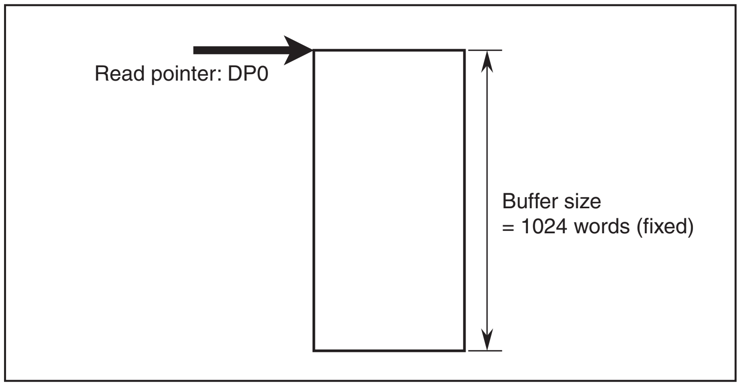
[Registers used] R0, R1, R2, R3, R4, R5, R6, R7, DP0, DP4

[Hardware resourcement]

Maximum stack level	3
Maximum loop stack level	2
Maximum number of repeats	1023
Maximum number of cycles	48413

Caution The a2d_InitDec function initializes only RAM areas that the μSAP77016-B08 uses. Initialization of user-defined RAM areas (such as I/O buffers) should be performed in a user program.

Figure 2-1. User-Defined Input Buffer



2.2.2 a2d_set_adif_header function

[Classification]	ADIF/RAW parameter initialization processing	
[Function name]	a2d_set_adif_header	
[Summary of function]	Sets parameters required for ADIF/RAW format decoding	
[Format]	call a2d_set_adif_header	
[Arguments]	DP0	Pointer to parameter save area
[Return value]	R0	0: Error Other than 0: Normal
[Function]	Set parameters required for ADIF/RAW format decoding.	
[Registers used]	R0, R2, R4, R5, R6, DP0, DP1, DP2	

[Hardware resourcement]

Maximum stack level	1
Maximum loop stack level	1
Maximum number of repeats	0
Maximum number of cycles	128

2.2.3 a2d_Dec function

[Classification] AAC decoder processing

[Function name] a2d_Dec

[Summary of function] Executes AAC decoding and obtains result of decoding

[Format] call a2d_Dec

[Arguments]

R0	0: Decode RAW_DATA only	
	1: Decode ADIF	★
	2: Decode ADTS	
R1	0: Perform normal decoding	
	1: Skip decoding for one frame	
DP0	Pointer to PCM data output area (see **Figure 2-2**)	

[Return value]

R0	0: Error
	Other than 0: Normal

[Function] Perform decoding in the mode specified in R0 and obtain a 2048-word decoding result. Output the decoding result in the output buffer specified in DP0.

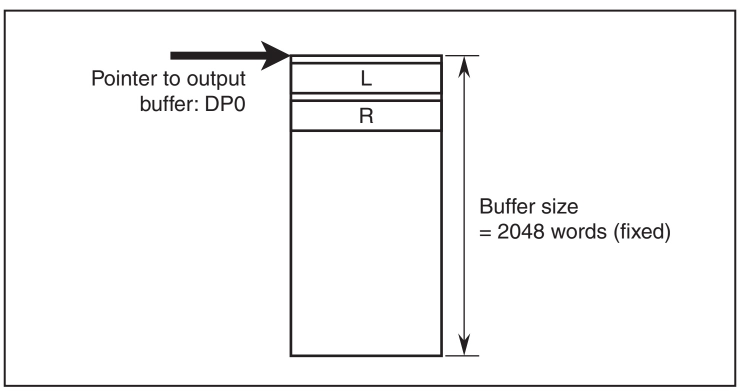
[Registers used] R0, R1, R2, R3, R4, R5, R6, R7, DP0, DP1, DP2, DP3, DP4, DP5, DP6, DP7, DN0, DN1, DN2, DN3, DN4, DN5, DN6, DN7, DMX, DMY

[Hardware resourcement]

★

Maximum stack level	6
Maximum loop stack level	4
Maximum number of repeats	176 (Measured maximum value)
Maximum MIPS value	31 MIPS (If there is no error encoding)
	31.5 MIPS (If there is error encoding)

Figure 2-2. User-Defined Output Buffer



2.2.4 a2d_GetVersion function

[Classification]	Version information acquisition	
[Function name]	a2d_GetVersion	
[Summary of function]	Returns the version of the library.	
[Format]	call a2d_GetVersion	
[Arguments]	None	
[Return value]	R0H	Major version number
	R0L	Minor version number
[Function]	Return the version number of the μSAP77016-B08 library in a 32-bit value.	
	Version when R0 = 0x00'0x0001'0x0100: V1.01	
[Registers used]	R0	
[Hardware resourcement]		
	Maximum stack level	0
	Maximum loop stack level	0
	Maximum number of repeats	0
	Maximum number of cycles	6

2.2.5 a2d_GetStatus function

[Classification] Status information acquisition

[Function name] a2d_GetStatus

[Summary of function] Obtains the status of the decoding result.
Information from the decoding performed just before this function was called is returned as the status.

[Format] call a2d_GetStatus

[Arguments] None

[Return value]

R0	0: Decode error
	★ 1: ADIF/RAW format
	2: ADTS format
R1	0: Result of decoding is monaural
	1: Result of decoding is stereo
R2	Index value of sampling frequency (See **Table 2-3**)
R3	0: Normal decoding
	Other than 0: Skip 1 frame of decoding
R4	Indicates the size [byte] of input data that is decoded by a2d_Dec just before this function

[Function] Obtain status information and set registers.

[Registers used] R0, R1, R2, R3, R4

[Hardware resourcement]

Maximum stack level	0
Maximum loop stack level	0
Maximum number of repeats	0
Maximum number of cycles	15

Table 2-3. Values of a2d_GetStatus() Return Value R2

Value of R2 (index value)		Sampling Frequency [Hz]
hex	bin	
0x00	‘0000’	96000
0x01	‘0001’	88200
0x02	‘0010’	64000
0x03	‘0011’	48000
0x04	‘0100’	44100
0x05	‘0101’	32000
0x06	‘0110’	24000
0x07	‘0111’	22050
0x08	‘1000’	16000
0x09	‘1001’	12000
0x0a	‘1010’	11025
0x0b	‘1011’	8000

2.2.6 a2d_GetErrorStatus function

[Classification] Error information acquisition

[Function name] a2d_GetErrorStatus

[Summary of function] Obtain AAC decoder error information.

[Format] call a2d_GetErrorStatus

[Arguments] None

[Return value] R0 Error status

[Function] Return the values below as AAC decoder error information.

Processing cannot be continued following the occurrence of an error.

Name	Value	Contents
_AAC_EBIT_FATAL_ERROR	0x0001	This is a fatal error.
_AAC_EBIT_CCE_FOUND	0x0002	A CCE[Note] was detected and ignored.
_AAC_EBIT_CRC_ERROR	0x0004	A CRC error occurred on ADTS format data.
★ _AAC_EBIT_NO_HEADER	0x0008	No header was found.

Note CCE : Coupling Channel Element

[Registers used] R0

[Hardware resourcement]

Maximum stack level 0

Maximum loop stack level 0

Maximum number of repeats 0

Maximum number of cycles 4

2.2.7 aac_dec_fillbits function

The aac_dec_fillbits function is a user function called by μSAP77016-B08 (callback function).
Create the function using the following specifications.

[Classification]	Input data acquisition function	
[Function name]	aac_dec_fillbits	
[Summary of function]	Fills input buffer with bit stream required for AAC decoding	
[Format]	a2d_Dec calls this function using the format call DP4.	
	Set the address of this function using the a2d_InitDec function.	
★ **[Arguments]**	a2d_Dec sets the following registers and calls the user-defined function.	
	R1	Fill request word count
	DP0	Input data write pointer
	DN0	1
	DMX	Input buffer size (1024 words fixed)
★ **[Return value]**	R0	Fixed to 0
	DP0	Input data write pointer
[Function]	Fill the input buffer with a bit stream required for AAC decoding.	
[Registers used]	If using registers other than those shown below, use them after saving register contents in memory.	
★	• R0, R2, R3	

★ 2.2.8 aac_read_1word function

The aac_read_1word function is a user function called by μSAP77016-B08 (callback function).
Create the function using the following specifications.

[Classification] Header seek function

[Function name] aac_read_1word

[Summary of function] Reads a header of the next frame to decode.

[Format] a2d_Dec calls this function using the format call DP4.
Set the address of this function using the a2d_InitDec function.

[Arguments] a2d_Dec sets the following register and calls the user-defined function.

R2	Size of the current frame to seek (byte)
R7	Information of frame position 0x000d: The beginning of the frame is higher byte (bit 15 to bit 8) of a word. 0x0005: The beginning of the frame is lower byte (bit 7 to bit 0) of a word.
DP1	Input buffer (user-defined) read pointer (This pointer has gone ahead just 6 bytes from the beginning of the current frame.)

[Return value]

R0	a header of the next frame **Remark**: If R0 is an expected header value (0xfff8 or 0xfff9), - a2d_Dec realize that the header recognition of the current frame is correct. - a2d_Dec output the error code (CRC error) as a return value. If R0 is other value, - a2d_Dec realize that the header recognition of the current frame is error. - a2d_Dec start seeking a header of the frame again.

[Function] In the case of CRC error in ADTS decoding, a2d_Dec calls this function. This function seeks and reads a header of the next frame to decode by frame size (R2), frame position (R7) and input buffer read pointer (DP1). This header information is used for distinction whether the header recognition of the current frame is correct in a2d_Dec.

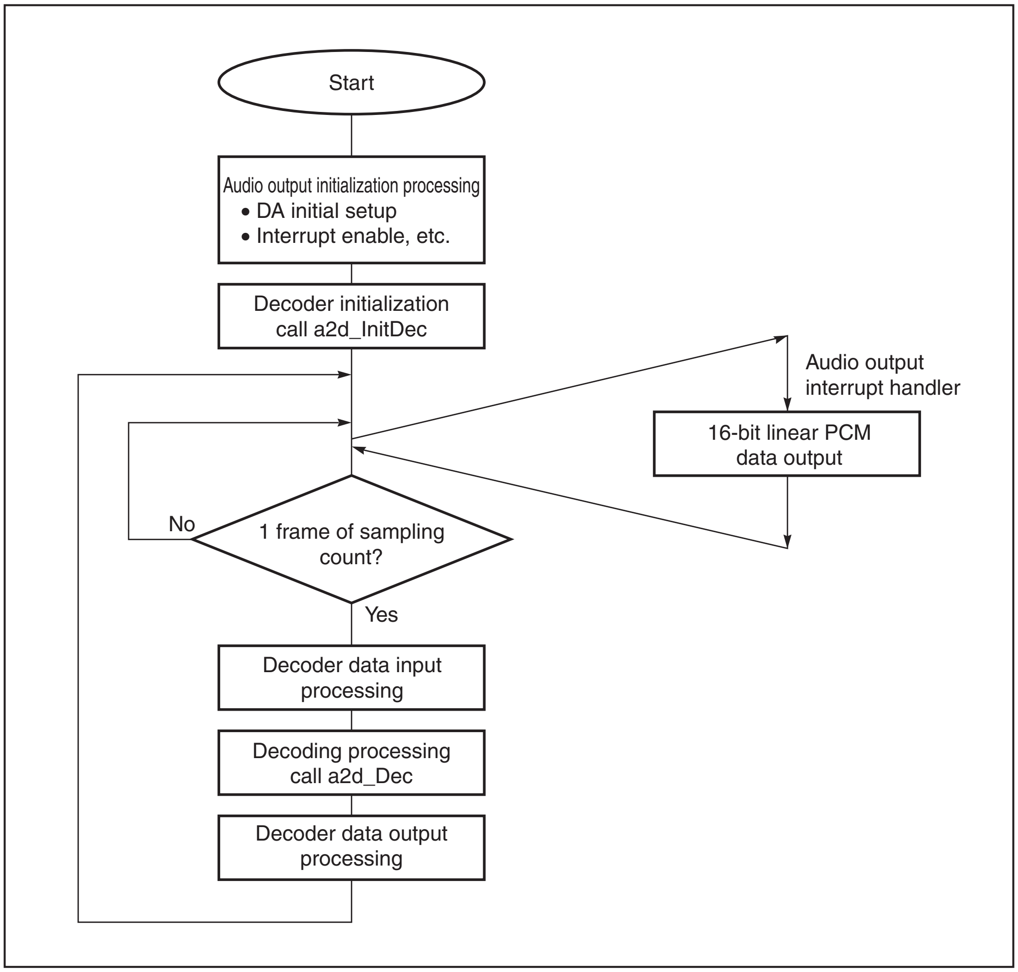
[Registers used] If using registers other than those shown below, use them after saving register contents in memory.

- R0, R1, R2, R7, DP1

2.3 Application Processing Flow

Figure 2-3 shows an example of the processing of an application that uses μSAP77016-B08.

Figure 2-3. Application Processing Flow (Decoder)

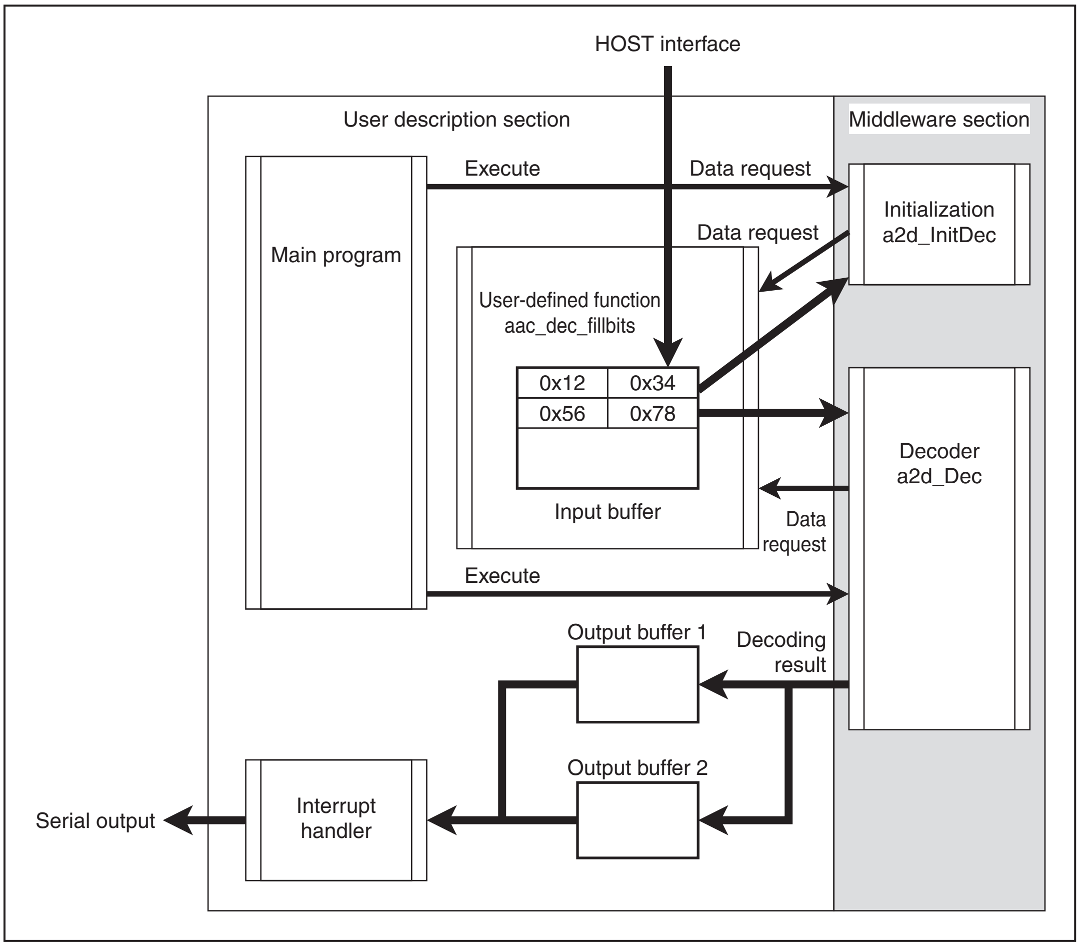


The audio data I/O processing section of the interrupt handler is processing that depends on the hardware of the target system. Consequently, the user should design it to suit the target system.

2.4 Data Flow

Figure 2-4 shows an example of the data flow when decoding.
Data in the input buffer must be set in order from MSB to LSB.
For example, place the data 0x1234, 0x5678, ... in the order 0x12, 0x34, 0x56, 0x78, ...

Figure 2-4. Sample Data Flow



CHAPTER 3 INSTALLATION

3.1 Installation Procedure

The μSAP77016-B08 (AAC decoder middleware) is supplied on a 3.5-inch floppy disk (1.44 MB). The procedure for installing the μSAP77016-B08 in the host machine is outlined below.

(1) Set the floppy disk in the floppy disk drive and copy the files to the directory where WB77016 and HSM77016 (DSP tools) are used (e.g. C:\DSPTools).
The following is an example of when files are copied from the A drive to the C drive.

```
A:\>xcopy /s *.* c:\DSPTools<CR>
```

(2) Confirm that the files have been copied. Refer to **1.3.5 Directory configuration** for details on the directories.

```
A:\>dir c:\DSPTools<CR>
```

3.2 Sample Program Creation Procedure

A sample program is stored in the smp directory.

The sample program operates on HSM77016 (high-speed simulator) Ver. 2.32 or later. Using the timing files described later makes it possible to simulate data I/O. Refer to **CHAPTER 4 SYSTEM EXAMPLE** regarding timing files.

The following is an explanation of how to build the AAC decoder middleware sample program.

(1) Start up the WB77016 (workbench) Ver.2.4 or later.
(2) Open the sample.prj project file.
Example) Specify sample.prj with the Open Project command on the Project menu.
(3) Execute Build and confirm that sample.lnk has been created.
Example) The sample.lnk file can be created by selecting the Build All command from the Make menu.
(4) Start up the HSM77016 (high-speed simulator) Ver.2.32 or later.
(5) Open the sample.lnk file.
Example) Specify sample.lnk with the Open command on the File menu.
(6) Open timing files (sample.tmg, serclk.tmg, serot16.tmg).
serclk.tmg and serot16.tmg are files provided by HSM77016 in Example.
Example) Specify sample.tmg with the Open command on the File menu.

3.3 Symbol Naming Regulations

The symbols used in this library are named according to the following regulations.

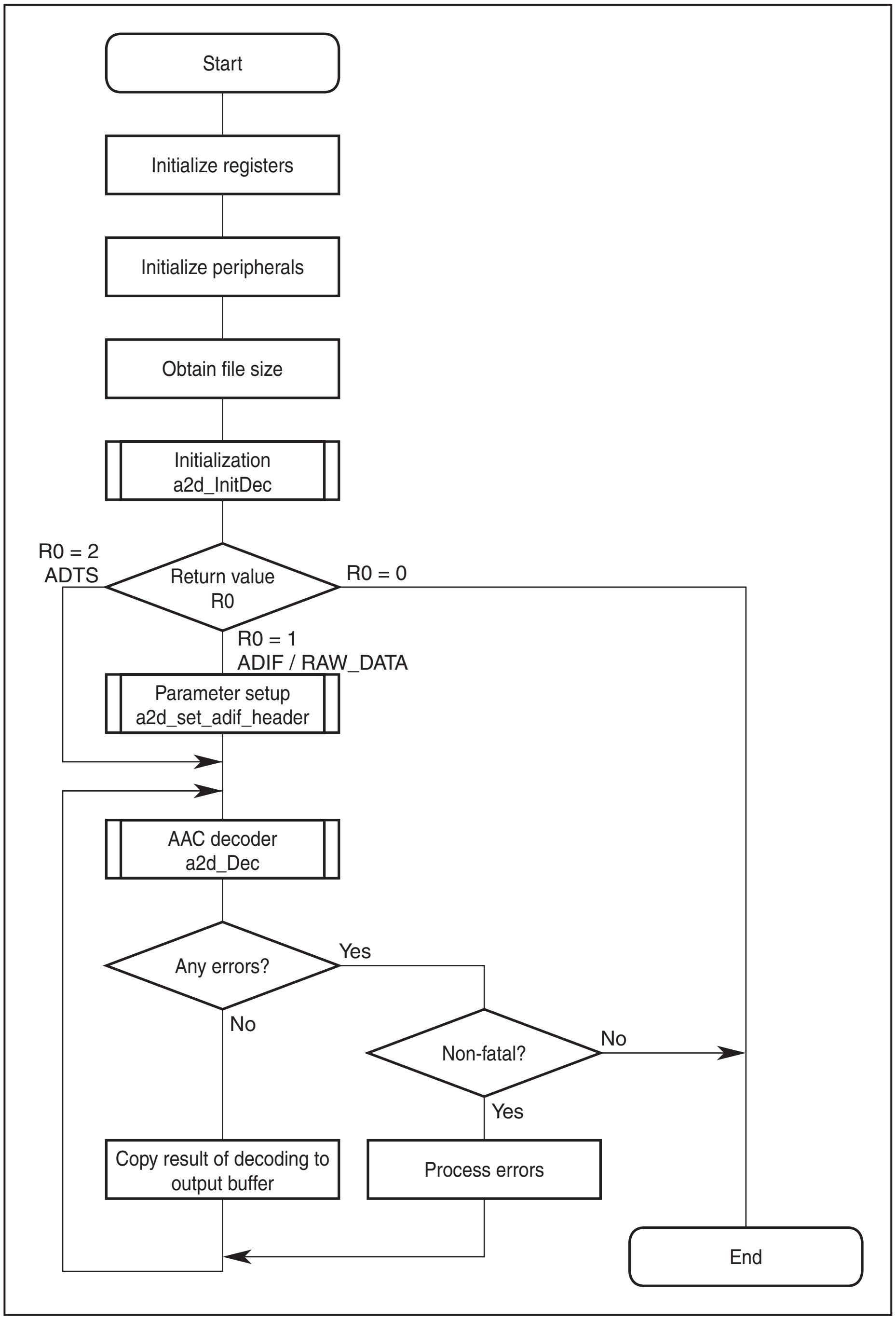
Table 3-1. Symbol Names

Classification	Regulation
Function name, variable name	a2d_xxxx
Macro, constant name	a2d_XXXX
Section name	_ _A2D_XXXX (Two leading underscores)

3.4 Sample Program Processing Flow

Figure shows the processing of a sample program that uses the AAC decoder.

Figure 3-1. Sample Program Processing Flow



CHAPTER 4 SYSTEM EXAMPLE

4.1 Simulation Environment Using Timing Files

An example in which an audio decoding decompression-processing simulator and timing files are used is shown below. Encoded data is input, and audio data is output frame by frame after each frame has undergone decompression processing.

[Software environment]

- High-speed simulator: HSM77016 Ver. 2.32 or later
- Sample program: sample.lnk (created in **3.2 Sample Program Creation Procedure**)
- Timing file: sample.tmg

4.2 Operation

<1> Start up the HSM77016 (High-speed simulator)

<2> Open sample.lnk created in **3.2 Sample Program Creation Procedure**.
Example Specify sample.lnk with Open command on the File menu.

<3> Open the timing files (sample.tmg, serclk.tmg, serot16.tmg).
serclk.tmg and serot16.tmg are files provided by HSM77016 in Example.
Example Specify sample.tmg with Open command on the File menu.

<4> Make the wait settings.
Example Set waits to the DWTR/IWTR registers in the setting windows opened by selecting Periphery Register on the Window menu.

<5> Execute with Run.

(a) Timing file sample.tmg

HSM77016 (high-speed simulator) provides a function for simulating external I/O using a timing file.

(b) Data file input (16-bit data)

Data is input from a file via the host interface. An example of the description format is shown below.

- Preparation

```
local data                   ; Compressed data storage variable
local size                   ; AAC file size (bytes)

set DEBUG_ID = 1             ; select target
open input "L144100.dat"     ; File name of AAC file converted to text data
set size = 65446             ; AAC file size (size of binary file in bytes)
```

- Input processing

(1/2)

```
          set pin hcs = 1                 ; terminate any write access, which might
          set pin hwr = 1                 ; be active
          set pin hrd = 1                 ;

; send AAC file size
             wait cond pin hwe == 0       ; wait till write is allowed
             wait cond pin hcs == 1       ; and no read is in progress
             set pin hcs = 0              ; perform the access...
             set port ha = 0              ; select higher byte of HDT
             set pin hwr = 0              ; start input
             set port hd = (size>>16)&0xFF ; input low byte to host port
             wait 100ns                   ; access duration
             set pin hwr = 1              ; end output
             set pin hcs = 1              ; terminate first access...
             wait 5ns                     ; delay
             set port ha = 1              ; select higher byte of HDT
             wait 5ns                     ; delay
             set pin hcs = 0              ; performe second access...
             set pin hwr = 0              ; start output
             set port hd = (size>>24)&0xFF ; input high byte to host port
             wait 100ns                   ; access duration
             set pin hwr = 1              ; end input
             set pin hcs = 1              ; end access

             wait 1                       ;

             wait cond pin hwe == 0       ; wait till write is allowed
             wait cond pin hcs == 1       ;      and no read is in progress
             set pin hcs = 0              ; perform the access...
             set port ha = 0              ; select higher byte of HDT
             set pin hwr = 0              ; start input
             set port hd = (size>>0)&0xFF  ; input low byte to host port
             wait 100ns                   ; access duration
             set pin hcs = 1              ; terminate first access...
             set pin hwr = 1              ; end output
             wait 5ns                     ; delay
             set port ha = 1              ; select higher byte of HDT
             wait 5ns                     ; delay
             set pin hcs = 0              ; performe second access...
             set pin hwr = 0              ; start output
             set port hd = (size>>8)&0xFF  ; input high byte to host port
             wait 100ns                   ; access duration
```

- Input processing

(2/2)

```
            set pin hwr = 1               ; end input
            set pin hcs = 1               ; end access

         do
            exit size<=0                  ;

            wait cond pin hwe == 0        ; wait till write is allowed
            wait cond pin hcs == 1        ;      and no read is in progress
            set pin hcs = 0               ; perform the access...
            set port ha = 0               ; select higher byte of HDT
            set pin hwr = 0               ; start input
            input data                    ; input host data to temp variable
            set port hd = data&0xFF       ; input low byte to host port
            wait 100ns                    ; access duration
            set pin hcs = 1               ; terminate first access...
            set pin hwr = 1               ; end output
            wait 5ns                      ; delay
            set port ha = 1               ; select higher byte of HDT
            wait 5ns                      ; delay
            set pin hwr = 0               ; start output
            set pin hcs = 0               ; performe second access...
            set port hd = (data>>8)&0xFF  ; input high byte to host port
            wait 100ns                    ; access duration
            set pin hwr = 1               ; end input
            set pin hcs = 1               ; end access
            set size = size-2             ;
         enddo

         close input

         wait cond (reg ip & 0xffff)==(MAIN.finish & 0xffff)
wait 1
```

- Termination

```
Break                                     ; Terminate
end
```

- **sample.h**

```
#define __OFFSET     4
#define RAW_DATA_STREAM 0

#define SDT1 0x3800
#define SST1 0x3801
#define SDT2 0x3802
#define SST2 0x3803
#define HDT  0x3806
#define HST  0x3807
#define DWTR 0x3808

extrn user_dec_mod
eextrn user_loaded_size
extrn user_decoded_size
extrn def_pce_setting
extrn user_input_bs
extrn user_output_buff1
extrn user_output_buff2
extrn user_skip_flag
extrn user_file_size
extrn user_stereo_flag
extrn user_frame_odd_flag
extrn user_int_output_ptr
extrn user_int_sv
extrn lib_Scratch_x
extrn lib_Scratch_y
```

- **sample.asm**

(1/3)

★
```
#include "a2d_dec.h"
#include "a2d_err.h"
#include "sample.h"

extrn user_wait
public start_up
extrn aac_read_1word
extrn aac_dec_fillbits

MAIN IMSEG at 0x300

start_up:

        r0l = 0x0401;
        *HST:x = r0l;

        dn0 = 0                                   ; for simulation
        dmx = 1                                   ; for simulation

        r0l = 0x8200                              ; lsb-first
                                                  ; continuous I/O mode
        *SST1:x=r0l                               ;
        r0l = user_output_buff1                   ;
        *user_int_output_ptr:x=r0l                ;
        clr(r0)                                   ;
        *user_frame_odd_flag:x = r0l              ;
        r0l = SR                                  ; enable interrupt
        r0  = r0 & 0x7fdf                         ;   SO1
        r0  = r0 | 0x0fdf                         ; disable other interrupt
        SR  = r0l                                 ;

        clr(r0)                                   ;
        *SDT1:x=r0l                               ; start output interrupt
        *SDT1:x=r0l                               ;

        call   fitst_data_read                    ; read file size

        r1eh = *user_file_size  :x                ;
        r1l  = *user_file_size+1:x                ;
        clr(r1)                                   ;
        *user_skip_flag:x  = r1l                  ;

#if RAW_DATA_STREAM    /* RAW_DATA_DEBUG */
        r4l = aac_read_1word                      ;
        r5l = aac_dec_fillbits                    ;
        r6l = lib_Scratch_x                       ;
        r7l = lib_Scratch_y                       ;
        clr(r0)                                   ; initialize only
        dp0 = user_input_bs                       ;
        call a2d_InitDec                          ;
        *user_dec_mode:x=r0l                      ;
        if(r0==0) jmp dec_error_fatal             ; initialize failed
        call a2d_GetStatus                        ;
        *user_decoded_size:x   = r4h              ;
        *user_decoded_size+1:x = r4l              ;
```

- **sample.asm**

(2/3)

```
        dp0 = def_pce_setting                    ;
        call a2d_set_adif_header                 ;
        if(r0==0) jmp dec_error_fatal            ; PCE element information invalid
#else
        r4l = aac_read_1word                     ;
        r5l = aac_dec_fillbits                   ;
        r6l = lib_Scratch_x                      ;
        r7l = lib_Scratch_y                      ;
        clr(r0)                                  ;
        r0l = _A2D_INI_DECODE_HEADER | _A2D_COMPREX_HEADER_SEARCH
                                                 ; initialize internal memory and read header
        dp0 = user_input_bs                      ;
        call a2d_InitDec                         ;
        *user_dec_mode:x=r0l                     ;
        if(r0==0) jmp dec_error_fatal            ; initialize failed
        call a2d_GetStatus                       ;
        *user_decoded_size:x   = r4h             ;
        *user_decoded_size+1:x = r4l             ;
#endif

test_loop:
        r1eh = *user_file_size:x                 ;
        r1l  = *user_file_size+1:x               ;
        r0eh = *user_decoded_size  :x            ;
        r0l  = *user_decoded_size+1:x            ;

        r1   = r1-r0                             ;
        r1   = r1-1                              ; all RAW_DATA_BLOCK larger than 1 byte
        if(r1<=0) jmp finish                     ;
        call user_wait                           ;
        clr(r1)                                  ;
        r1l = *user_skip_flag:x                  ;
        clr(r0)                                  ;
        r0l = *user_dec_mode:x                   ;
start_dec:
        call a2d_Dec                             ;
        if(r0==0) jmp dec_error                  ; if decoder failed jmp infinit loop
        call a2d_GetStatus                       ;
        r1eh = *user_decoded_size  :x            ;
        r1l  = *user_decoded_size+1:x            ;
        r1   = r1 + r4                           ;
        *user_decoded_size  :x  = r1h            ;
        *user_decoded_size+1:x  = r1l            ;

        if(r3!=0) jmp _start_next_decode         ;

_start_next_decode:
        jmp test_loop                            ;

finish:
        nop                                      ;
        jmp start_up                             ;
```

- **sample.asm**

(3/3)

```
dec_error:
        call a2d_GetErrorStatus                 ;
        r1  = r0 & _AAC_EBIT_FATAL_ERROR        ;
        if(r1!=0) jmp dec_error_fatal           ;
        r1  = r0 & _AAC_EBIT_CRC_ERROR          ;
        if(r1!=0) jmp dec_error_crc             ;
dec_error_fatal:                                ;
        jmp $                                   ;
dec_error_crc:
        nop                                     ; this code use only to DEBUG.
dec_error2:
        jmp $                                   ;
dec_error3:
        jmp $                                   ;

fitst_data_read:
        clr(r0)                                 ;
        *user_loaded_size+0:x = r0l             ;
        *user_loaded_size+1:x = r0l             ;
        *user_decoded_size  :x = r0l            ;
        *user_decoded_size+1:x = r0l            ;
        r0eh= *HDT:x                            ; read __OFFSET information
        r0l = *HDT:x                            ;

; get file size
        *user_file_size  :x=r0h                 ;
        *user_file_size+1:x=r0l                 ;
        ret                                     ;

        end
```

★ • **sam_call.asm**

(1/2)

```
#include "a2d_dec.h"
#include "a2d_err.h"
#include "sample.h"

public aac_dec_fillbits;
public aac_read_1word;


__USER_CALLBACK imseg
/***********************************
function
       aac_read_1word
input
       r2  frame size[byte]
       r7  remain size
       dp1 aac_input_buff read ptr

output
       r0l header data
***********************************/
aac_read_1word:
    r2  = r2 - 6                      ; 6byte already read.
    r7  = r7 srl 3                    ; 0x000d then MSB, 0x0005 then Not MSB
    if(r7 != 0) r2 = r2 - 1           ;
    r7  = r2 & 0x0001                 ;
    r0  = *user_loaded_size+0:x       ;
    r0l = *user_loaded_size+1:x       ; already loaded data size
    r1  = *user_decoded_size+0:x      ;
    r1l = *user_decoded_size+1:x      ; already decoded data size
    r0 -= r1                          ;
    r0 -= r2                          ;
    if(r0 < 0) jmp _not_read          ; lack of data -> jmp _not_read
    r2  = r2 sra 1                    ; convert byte to word

    r0l = dp1                         ;
    r0  = r0 + r2                     ;
    clr(r1)                           ;
    r1l = user_input_bs               ;
    r1  = r1 + INPUT_BUFSIZE          ;
    r1  = r1 - r0                     ;
if(r1 > 0) jmp _set_ptr           ;
    r0  = r0 - INPUT_BUFSIZE          ;

_set_ptr:
    dp1 = r0l                         ;
    nop                               ;
    r0l = *dp1%%                      ;

    if(r7 == 0) ret                   ;
    r0  = r0 sll 8                    ;
    r2l = *dp1                        ;
    r2  = r2 srl 8                    ;
```

- **sam_call.asm**

(2/2)

```
    r0  = r0 | r2                     ;
    ret                               ;

_not_read:
    clr(r0)                           ;
    ret                               ;

/************************************
function
        aac_dec_fillbits
input
        r1  req size [word]
        dp0 user_input_bs write ptr
        dn0 0x01
        dmx INPUT_BUFSIZE-1
output
        dp0 user_input_bs write ptr
************************************/
aac_dec_fillbits:

#if INPUT_BUFSIZE<(12288/16+1)
        Error too small INPUT_BUFSIZE
#endif

    r2  = *user_file_size+0:x         ;
    r2l = *user_file_size+1:x         ; total data size
    r0  = *user_loaded_size+0:x       ;
    r0l = *user_loaded_size+1:x       ; already loaded data size
r2  = r2 - r0                     ;
    if(r2 <= 0) jmp _load_end         ; if all data loaded return
    r2  = r2 + 1                      ;
    r2  = r2 sra 1                    ; conver byte to word
    r3  = r2 - r1                     ;
    if(r3 < 0) r1 = r2                ;

    loop r1l {                        ;
        r0 = *HDT:x                   ;
        *dp0%% = r0h                  ;
    }

    r0  = *user_loaded_size+0:x       ;
    r0l = *user_loaded_size+1:x       ; already loaded data size
    r1  = r1 sll 1                    ;
    r0 += r1                          ;
    *user_loaded_size+0:x = r0h       ;
    *user_loaded_size+1:x = r0l       ;
_load_end:
    clr(r0)                           ;
    ret                               ;

    end
```

●sam_data.asm

(1/2)

★

```
#include "a2d_dec.h"

public user_dec_mode
public user_loaded_size
public user_decoded_size
public user_file_size
public user_skip_flag
public user_stereo_flag

public user_input_bs
public user_output_buff1
public user_output_buff2

public user_frame_odd_flag
public user_int_output_ptr
public user_int_sv

public def_pce_setting

public lib_Scratch_x
public lib_Scratch_y

LIB_SCRATCH_X XRAMSEG AT 0x0000
lib_Scratch_x:        ds          2048
LIB_SCRATCH_Y YRAMSEG ALIGN AT 0
lib_Scratch_y:        ds          4096

MAINX XRAMSEG
user_dec_mode:        ds          1
user_loaded_size:     ds          2
user_decoded_size:    ds          2
user_file_size:       ds          2
user_skip_flag:       ds          1
user_stereo_flag:     ds          1
user_frame_odd_flag:  ds          1
user_int_output_ptr:  ds          1
user_int_sv:          ds          6

MAINRX XRAMSEG
def_pce_setting:
       DW       0x0441            ;  <profile               = LOW COMPLEXITY>
                                  ;  <sampling frequency = 44100>
                                  ;  <num_front_channel_elements=1>
       DW       0                 ;
       DW       0                 ;
                                  ;define front channel
       DW       0x0200            ;  <front_element_is_cpe[0]     =1>
                                  ;  <front_element_tag_select[0]=0>
       DS       7                 ;
                                  ;define side channel
       DS       8                 ;
                                  ;define back channel
       DS       1                 ;lfe
       DS       8                 ;
                                  ;define assoc
```

- **sam_data.asm**

(2/2)

```
        DS              2           ;
                                    ;define cce
        DS              8           ;
        DW              0xffff      ;end mark(only for product debugging)

__AAC_DEC_BS_XRAMSEG1  XRAMSEG ALIGN                ;
user_input_bs:          ds          INPUT_BUFSIZE    ;

__AAC_SAMPLE_OUTPUT_BUFFX          XRAMSEG ALIGN
user_output_buff1:      ds          OUTPUT_BUFSIZE   ;
;       even L ch       ;
;       odd  R ch       ;

user_output_buff2:      ds          OUTPUT_BUFSIZE   ;
;       even L ch       ;
;       odd  R ch       ;
        end
```

- **sam_int.asm**

(1/2)

★

```
#include "a2d_dec.h"
#include "a2d_err.h"
#include "sample.h"

extrn start_up
extrn aac_dec_fillbits;
extrn aac_read_1word

MAIN_VECTOR IMSEG at 0x200
        jmp start_up;

org 0x224
        jmp serial_out                  ;
        nop                             ;
        nop                             ;

public user_wait

#define _svR0L                          (user_int_sv+0)
#define _svR0H                          (user_int_sv+1)
#define _svR0E                          (user_int_sv+2)
#define _svDP0                          (user_int_sv+3)
#define _svDN0                          (user_int_sv+4)
#define _svDMX                          (user_int_sv+5)

serial_out:
        *_svR0L:x = r0l                 ;
        *_svR0H:x = r0h                 ;
        *_svR0E:x = r0e                 ;
        r0l = dp0                       ;
        *_svDP0:x = r0l                 ;
        r0l = dn0                       ;
        *_svDN0:x = r0l                 ;
        r0l = dmx                       ;
        *_svDMX:x = r0l                 ;

        r0l = *user_int_output_ptr:x    ;
        dp0 = r0l                       ;
        dn0 = 0x01                      ;
        dmx = 1024*4-1                  ;

        r0l = *dp0%%                    ; get left PCM
        *SDT1:x = r0l                   ;

        r0l = dp0                       ;
        *user_int_output_ptr:x=r0l      ;

        r0l = *_svDMX:x                 ;
        dmx = r0l                       ;
        r0l = *_svDN0:x                 ;
        dn0 = r0l                       ;
        r0l = *_svDP0:x                 ;
        dp0 = r0l                       ;
        r0e = *_svR0E:x                 ;
        r0h = *_svR0H:x                 ;
```

- **sam_int.asm**

(2/2)

```
        r0l = *_svR0L:x                         ;
        reti                                    ;

user_wait:
        clr(r0)                                 ;
        clr(r1)                                 ;
        r1l = *user_frame_odd_flag:x            ;
        r1  = r1 ^ 0x01                         ;
        *user_frame_odd_flag:x = r1l            ;
        if(r1==0) jmp _wait_frame_odd           ;

_wait_frame_even:
        clr(r0)                                 ;
;       halt                                    ;
        r0l = *user_int_output_ptr:x            ;
        r0  = r0 - user_output_buff2            ;
        if (r0 < 0 ) jmp _wait_frame_even       ;
        dp0 = user_output_buff1                 ;
        jmp _wait_frame_end                     ;

_wait_frame_odd:
        clr(r0)                                 ;
;       halt                                    ;
        r0l = *user_int_output_ptr:x            ;
        r0  = r0 - user_output_buff2            ;
        if (r0 > 0 ) jmp _wait_frame_odd        ;
        dp0 = user_output_buff2                 ;
;       jmp _wait_frame_end                     ;

_wait_frame_end:
        ret                                     ;
;

end
```

NEC

Facsimile Message

From:

Name

Company

Tel. FAX

Address

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

Thank you for your kind support.

North America
NEC Electronics Inc.
Corporate Communications Dept.
Fax: +1-800-729-9288
+1-408-588-6130

Europe
NEC Electronics (Europe) GmbH
Market Communication Dept.
Fax: +49-211-6503-274

South America
NEC do Brasil S.A.
Fax: +55-11-6462-6829

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Korea
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: +82-2-528-4411

P.R. China
NEC Electronics Shanghai, Ltd.
Fax: +86-21-6841-1137

Taiwan
NEC Electronics Taiwan Ltd.
Fax: +886-2-2719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Japan
NEC Semiconductor Technical Hotline
Fax: +81- 44-435-9608

I would like to report the following error/make the following suggestion:

Document title: ______________________

Document number: ______________________ Page number: ____________

____

____

____

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	❏	❏	❏	❏
Technical Accuracy	❏	❏	❏	❏
Organization	❏	❏	❏	❏