NEC

# User's Manual

# μSAP77016-B11

## WMA Decoder Middleware

---

## Target Devices

μPD77113A
μPD77114
μPD77210
μPD77213

Document No. U15683EJ1V0UM00 (1st edition)
Date Published January 2002 N CP(N)

**[MEMO]**

**Windows and Windows Media are either trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries.**

- **The information in this document is current as of January, 2002. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC's data sheets or data books, etc., for the most up-to-date specifications of NEC semiconductor products. Not all products and/or types are available in every country. Please check with an NEC sales representative for availability and additional information.**
- No part of this document may be copied or reproduced in any form or by any means without prior written consent of NEC. NEC assumes no responsibility for any errors that may appear in this document.
- NEC 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 NEC semiconductor products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of customer's equipment shall be done under the full responsibility of customer. NEC assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC endeavours to enhance the quality, reliability and safety of NEC semiconductor products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC semiconductor products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment, and anti-failure features.
- NEC semiconductor products are classified into the following three quality grades:
  "Standard", "Special" and "Specific". The "Specific" quality grade applies only to semiconductor products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of a semiconductor product depend on its quality grade, as indicated below. Customers must check the quality grade of each semiconductor product before using it in a particular application.

  "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

  "Special": 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, life support systems and medical equipment for life support, etc.

  The quality grade of NEC semiconductor products is "Standard" unless otherwise expressly specified in NEC's data sheets or data books, etc. If customers wish to use NEC semiconductor products in applications not intended by NEC, they must contact an NEC sales representative in advance to determine NEC's willingness to support a given application.

  (Note)

  (1) "NEC" as used in this statement means NEC Corporation and also includes its majority-owned subsidiaries.

  (2) "NEC semiconductor products" means any semiconductor product developed or manufactured by or for NEC (as defined above).

M8E 00. 4

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

• 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

**NEC Electronics (France) S.A.**
Vélizy-Villacoublay, France
Tel: 01-3067-58-00
Fax: 01-3067-58-99

**NEC Electronics (France) S.A.**
**Representación en España**
Madrid, Spain
Tel: 091-504-27-87
Fax: 091-504-28-60

**NEC Electronics Italiana S.R.L.**
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

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

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

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


J01.12

# 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 μPD77113A, 77114, 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 INTRODUCTION**
**CHAPTER 2 LIBRARY SPECIFICATIONS**
**CHAPTER 3 INSTALLATION**
**CHAPTER 4 SYSTEM EXAMPLE**
**APPENDIX A 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 Architecture User's Manual***.

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

**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 / Part Number | Pamphlet | Data Sheet | User's Manual | | Application Note |
|---|---|---|---|---|---|
| | | | Architecture | Instructions | Basic Software |
| μPD77113A | U12395E | U14373E | U14623E | U13116E | U11958E |
| μPD77114 | | | | | |
| μPD77210 | | U15203E | To be prepared | | |
| 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 INTRODUCTION

## 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.

The μSAP77016-B11 introduced here is middleware that supplies decoding functions using Windows Media™ Audio (WMA) technology.

## 1.2 WMA Decoder

Windows Media Audio is the audio standard of the Windows Media Technology (WMT) audio/video streaming technology promoted by Microsoft Corporation. WMA is based on the Microsoft-standard Advanced Systems Format (ASF) and is used together with the MPEG-4 (Moving Picture Experts Group) and WMV (Windows Media Video) standards.

The μSAP77016-B11 performs decoding using WMA decoding technology.

## 1.3 Product Overview

### 1.3.1 Features

- All bit rates and sampling frequency data encoded by Windows Media Audio CODEC Versions 2, 7, and 8 can be decoded (refer to **Table 1-1 Bit Rates and Sampling Frequencies Supported by Each Codec**).
- Decoding results are output in 16-bit linear PCM format
- The ASF file format is supported
- Extracting and decoding only audio data from data including video is possible
- Contents information can be read out
- DRM (Digital Rights Management), script commands, and marker functions are not supported

**Table 1-1. Bit Rates and Sampling Frequencies Supported by Each Codec**

| Bit Rate [bps] | Sampling Frequency [Hz] | Number of Channels | WMA CODEC Version 8 | WMA CODEC Version 7 | WMA CODEC Version 2 |
|---|---|---|---|---|---|
| 192,000 | 48,000 | 2 | √ | – | – |
| 160,000 | 48,000 | 2 | √ | √ | √ |
| 128,000 | 48,000 | 2 | √ | √ | √ |
| 192,000 | 44,100 | 2 | √ | √ | – |
| 160,000 | 44,100 | 2 | √ | √ | √ |
| 128,000 | 44,100 | 2 | √ | √ | √ |
| 96,000 | 44,100 | 2 | √ | √ | √ |
| 80,000 | 44,100 | 2 | √ | √ | √ |
| 64,000 | 44,100 | 2 | √ | √ | √ |
| 48,000 | 44,100 | 2 | √ | – | √ |
| 32,000 | 44,100 | 2 | √ | – | – |
| 48,000 | 44,100 | 1 | √ | – | – |
| 32,000 | 44,100 | 1 | √ | √ | √ |
| 64,000 | 32,000 | 2 | – | – | √ |
| 48,000 | 32,000 | 2 | √ | √ | √ |
| 44,000 | 32,000 | 2 | – | – | √ |
| 40,000 | 32,000 | 2 | √ | √ | √ |
| 36,000 | 32,000 | 2 | – | – | √ |
| 32,000 | 32,000 | 2 | √ | √ | √ |
| 22,000 | 32,000 | 2 | – | – | √ |
| 32,000 | 32,000 | 1 | – | – | √ |
| 20,000 | 32,000 | 1 | √ | √ | √ |
| 32,000 | 22,050 | 2 | √ | √ | √ |
| 22,000 | 22,050 | 2 | √ | √ | √ |
| 20,000 | 22,050 | 2 | √ | √ | √ |
| 20,000 | 22,050 | 1 | √ | √ | √ |
| 16,000 | 22,050 | 1 | √ | √ | √ |
| 20,000 | 16,000 | 2 | √ | √ | √ |
| 16,000 | 16,000 | 2 | √ | √ | √ |
| 16,000 | 16,000 | 1 | √ | √ | √ |
| 12,000 | 16,000 | 1 | √ | √ | √ |
| 10,000 | 16,000 | 1 | √ | √ | √ |
| 10,000 | 11,025 | 1 | √ | √ | √ |
| 8,000 | 11,025 | 1 | √ | √ | √ |
| 12,000 | 8,000 | 2 | √ | √ | √ |
| 8,000 | 8,000 | 1 | √ | √ | √ |
| 6,000 | 8,000 | 1 | √ | √ | √ |
| 5,000 | 8,000 | 1 | √ | √ | √ |
| 128[Note] | 8,000 | 1 | √ | √ | √ |

**Note** Although this value may be selected when video-only data is encoded, valid audio data is not included.

**Remark** √: Supported, –: Non-existent combination

### 1.3.2 Operating environment

**(1) Operating target DSP:**

μPD77113A, 77114, 77210, 77213

**(2) Required memory size:**

The μSAP77016-B11 requires the memory sizes shown in the following table in order to support all the bit rates.

**Table 1-2. Required Memory Size**

| Memory | Type | Size [Kwords] |
|---|---|---|
| Instruction memory | – | 12.6 |
| X memory | RAM | 10.5 |
| | ROM | 17.5 |
| Y memory | RAM | 12.5 |
| | ROM | 9.6 |

**Cautions 1. One word is 32 bits in the instruction memory and 16 bits in the X and Y memories.**

**2. The memory size does not include PCM data and bit stream data buffers. Note also that the required memory size can be reduced by limiting the supported bit rates. For details, refer to 2.5 Memory Configuration.**

**(3) Software tools (Windows™ version):**

**Table 1-3. Software Tools**

| Target DSP | Software Tool |
|---|---|
| μPD77016 Family | DSP tools<br>WB77016 (workbench)<br>HSM77016 (high-speed simulator)<br>LB77016 (librarian) |

### 1.3.3 Performance

The MIPS values required to perform WMA decoding in real time are shown in **Table 1-4 MIPS Values Required for WMA Decode Processing**.

**[Measurement conditions]**

- Simulator: HSM77016
- Evaluation results: Measure the processing capacity when each WMA file is decoded to determine the typical and maximum values.
- Assign the values shown in **Table 2-5 Recommended Output Buffer Size** for the decode processing unit (number of samples) and output buffer size.
- The MIPS values only indicate the processing capacity of the wmad_FileDecodeData and wmad_FileGetPCM functions used for decoding and wmad_FileCBGetData function; the processing capacity of other functions and the interrupt handler, etc., is not included. Use the file attached as a sample for the user-defined wmad_FileCBGetData function. Note that the processing capacity may differ depending on the system configuration.

**Table 1-4. MIPS Values Required for WMA Decode Processing**

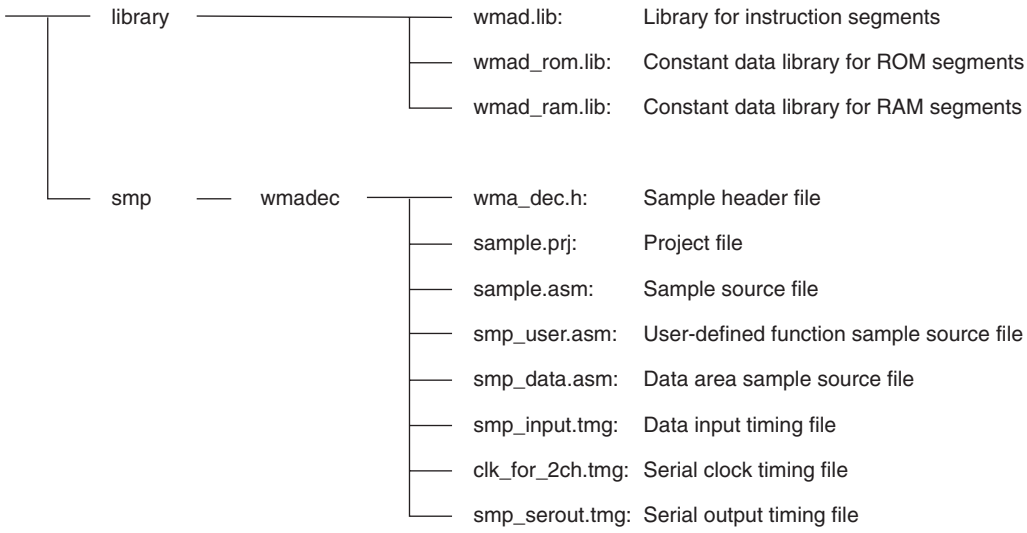
| Decoding Conditions | | | Processing Capacity | |
|---|---|---|---|---|
| Bit Rate [kbps] | Sampling Frequency [kHz] | Number of Channels | Typical MIPS Value [MIPS] | Maximum MIPS Value [MIPS] |
| 22 | 22 | 2 | 28 | 51 |
| 22 | 32 | 2 | 44 | 74 |
| 32 | 32 | 2 | 27 | 47 |
| 192 | 48 | 2 | 41 | 71 |

**Caution The maximum MIPS value may be larger than the values shown in Table 1-4 depending on the data.**
**It is recommended to implement measures such as outputting a silent sound for a section that cannot perform decode processing in real time, on the system side.**

### 1.3.4 Directory configuration

The directory configuration in the μSAP77016-B11 is shown below.

**Figure 1-1. μSAP77016-B11 Directory Configuration**




The directories are outlined below.

**(1) library**

Stores the library files.

- wmad.lib: Library file for instruction segments
- wmad_rom.lib: Constant data library file for ROM segments
  Select when allocating constant data to ROM.
- wmad_ram.lib: Constant data library file for RAM segments
  Select when allocating constant data to RAM.

**(2) smp/wmadec**

Stores the source, header, and simulation timing files of the sample program. Simulation can be performed using the high-speed simulator by utilizing these timing files (refer to **4.1 Environment Required for Simulation Using Timing Files**).

# CHAPTER 2 LIBRARY SPECIFICATIONS

This chapter describes the function specifications and function call regulations in the μSAP77016-B11.

## 2.1 Library Overview

The μSAP77016-B11 provides the following five functions.

**Table 2-1. Library Function List**

| Function Name | Description |
|---|---|
| wmad_FileDecodeInit | Decoder initialization processing |
| wmad_FileDecodeData | Decode processing |
| wmad_FileGetPCM | Decode processing |
| wmad_GetVersion | Version information acquisition |
| wmad_FileDecodeInfo | File information acquisition |

The following function must also be provided by the user in order to operate the μSAP77016-B11.

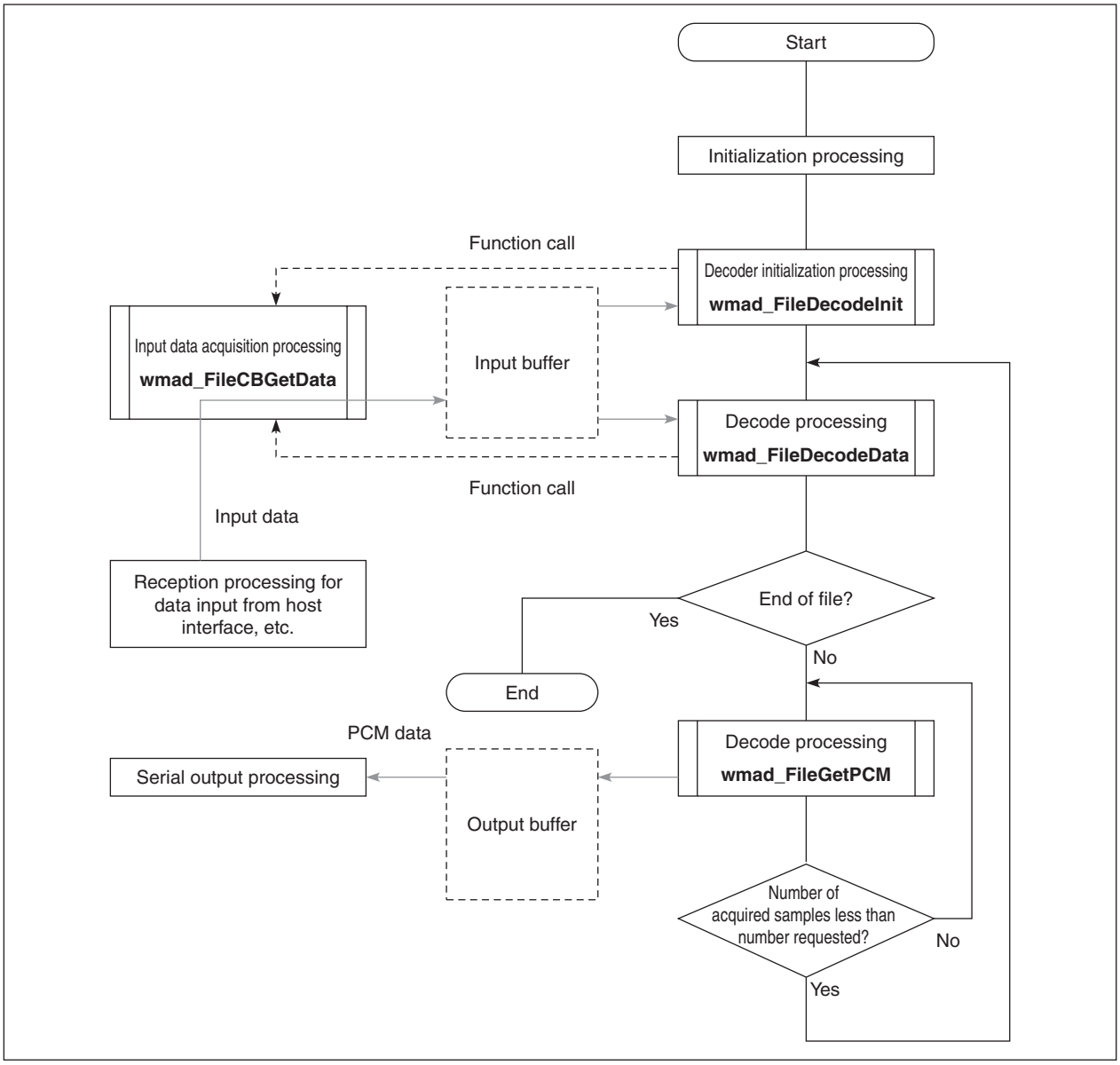
**Table 2-2. User-Defined Function List**

| Function Name | Description |
|---|---|
| wmad_FileCBGetData | Input data acquisition function |

## 2.2 Application Processing Flow

An example of the application processing flow is shown below.

**Figure 2-1. Application Processing Flow**



## 2.3 Function Specifications

The specifications for calling each library function are as follows.

### 2.3.1 wmad_FileDecodeInit function

[Classification] WMA decoder initialization processing function

[Function name] wmad_FileDecodeInit

[Summary of function] Initializes the RAM area and sets the parameters used by the μSAP77016-B11.

[Format] call wmad_FileDecodeInit

[Arguments]

| Argument | Description |
|---|---|
| R0L | Start address of user-defined wmad_FileCBGetData function |
| R1 | Contents information acquisition<br>1: Acquire 0: Do not acquire |
| R2L | Start address of structure storing contents information (X memory) |

[Return value]

| Return Value | Description |
|---|---|
| R0 | Error code (see **2.4 Error Information** for error code details) |

[Function] This function sets the parameters and initializes the RAM area used by the μSAP77016-B11. If the bit stream data contains contents information, this information can be acquired. An area (structure) for storing contents information must be prepared beforehand (refer to **Table 2-7 Members of Contents Information Structure**).

[Registers used] R0, R1, R2, R3, R4, R5, R6, R7,
DP0, DP1, DP2, DP3, DP4, DN0

[Hardware resources]

| | |
|---|---|
| Maximum stack level | 5 |
| Maximum loop stack level | 2 |
| Maximum number of repetitions | 502 |
| Maximum number of cycles | $5.5 \times 10^4$ |

[Remark] Because this function calls the wmad_FileCBGetData function, when using the call stack via the wmad_FileCBGetData function, the maximum stack level will increase by the amount of that call stack. Note also that the maximum number of repetitions and maximum number of cycles depend on the number of repetitions and cycles of the wmad_FileCBGetData function. The values above are for when the wmad_FileCBGetData function of the sample source is used and each item of contents information is acquired as 40 characters. The maximum number of cycles may also differ depending on factors such as the number of characters in the acquired contents information and whether marker information is included or not.

### 2.3.2 wmad_FileDecodeData function

[Classification] WMA decode processing function

[Function name] wmad_FileDecodeData

[Summary of function] Decodes the bit stream data and creates the data required for PCM data creation.

[Format] call wmad_FileDecodeData

[Arguments] None

[Return value]

| Return Value | Description |
|---|---|
| R0 | Error code (see **2.4 Error Information** for error code details) |

[Function] This function decodes the bit stream data and stores the data required for PCM data creation in the static area. When decoding has finished, the error code cWMA_Failed or cWMA_NoMoreFrames is returned. Note that it is not necessary to subsequently execute the wmad_FileGetPCM function when the error code is not cWMA_NoErr. This function should be re-executed only after the wmad_FileGetPCM function is executed and all the PCM data that can be created at that time is created.

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

[Hardware resources]

| | |
|---|---|
| Maximum stack level | 7 |
| Maximum loop stack level | 2 |
| Maximum number of repetitions | 25 |
| Maximum number of cycles | $2.7 \times 10^6$ |

[Remark] Because this function calls the wmad_FileCBGetData function, when using the call stack via the wmad_FileCBGetData function, the maximum stack level will increase by the amount of that call stack. Note also that the maximum number of repetitions and maximum number of cycles depend on the number of repetitions and cycles of the wmad_FileCBGetData function. The values above are for when the wmad_FileCBGetData function of the sample source is used.

### 2.3.3 wmad_FileGetPCM function

[Classification] WMA decode processing function

[Function name] wmad_FileGetPCM

[Summary of function] Creates PCM data from the results of decoding wmad_FileDecodeData.

[Format] call wmad_FileGetPCM

[Arguments]

| Argument | Description |
|---|---|
| R0L | Start address of output buffer (X memory) |
| R1L | Number of requested PCM samples (per channel) |

[Return value]

| Return Value | Description |
|---|---|
| R1 | Number of acquired PCM samples (per channel) |

[Function] This function converts the decoding results stored in the static area into PCM-format data and stores PCM data totaling the number of requested PCM samples multiplied by the number of channels in the specified buffer in the X memory. In the case of 2-channel data, data is stored alternately in the order of L channel then R channel. For the argument indicating the number of requested PCM samples, specify a value that is either the same as or smaller than the PCM buffer size (but at least 1). If the return value indicating the number of acquired PCM samples is smaller than the number of requested PCM samples, it indicates that decoding of all the data stored in the static area has finished. To acquire the next PCM data, re-execute the wmad_FileDecodeData function. Note that the size of the user-defined output buffer can be reduced by reducing the number of requested PCM samples.

[Registers used] R0, R1, R2, R3, R4, R5, R6, R7,
DP0, DP1, DP2, DP3, DP4, DP5,
DN0, DN2, DN5, DN7

[Hardware resources]

| | |
|---|---|
| Maximum stack level | 4 |
| Maximum loop stack level | 1 |
| Maximum number of repetitions | 0 |
| Maximum number of cycles | $2.1 \times 10^5$ |

[Remark] The maximum number of cycles is the value when the number of acquired PCM samples per channel is 2048.

### 2.3.4 wmad_GetVersion function

[Classification] Version information acquisition function

[Function name] wmad_GetVersion

[Summary of function] Returns the versions of the library and corresponding Windows Media Player.

[Format] call wmad_GetVersion

[Arguments] None

[Return value]

| Return Value | Description |
|---|---|
| R0H | Major version number of this library |
| R0L | Minor version number of this library |
| R1H | Major version number of the corresponding Windows Media Player |
| R1L | Minor version number of the corresponding Windows Media Player |

[Function] This function returns the version number of this library and the version number of the corresponding Windows Media Player as a 32-bit value.

**Example** When R0 = 0x00'0x0001'0x0100, the library version is V1.01
When R1 = 0x00'0x0007'0x0000, the Windows Media Player version is V7.0

[Registers used] R0, R1

[Hardware resources]

| | |
|---|---|
| Maximum stack level | 0 |
| Maximum loop stack level | 0 |
| Maximum number of repetitions | 0 |
| Maximum number of cycles | 10 |

### 2.3.5 wmad_FileDecodeInfo function

[Classification] File information acquisition function

[Function name] wmad_FileDecodeInfo

[Summary of function] Acquires WMA file information.

[Format] call wmad_FileDecodeInfo

[Arguments]

| Argument | Description |
|---|---|
| R0L | Start address of structure storing file information (X memory) |

[Return value]

| Return Value | Description |
|---|---|
| R0 | Error code (see **2.4 Error Information** for error code details) |

[Function] This function acquires file information such as the bit rate and sampling frequency and stores the result in the file information structure. Execute this function after executing the wmad_FileDecodeInit function. Note that at this time an area (structure) for storing file information must be prepared in the X memory beforehand using the start address of that area as the argument. Refer to **Table 2-6 Members of File Information Structure** for details of the structure for storing file information.

[Registers used] R0, R1, R2, DP0

[Hardware resources]

| | |
|---|---|
| Maximum stack level | 0 |
| Maximum loop stack level | 0 |
| Maximum number of repetitions | 0 |
| Maximum number of cycles | 41 |

### 2.3.6 wmad_FileCBGetData function (user-defined function)

[Classification] Input data acquisition function

[Function name] wmad_FileCBGetData

[Summary of function] Supplements the input buffer with the bit stream data required for decoding.

[Format] The wmad_FileDecodeInit and wmad_FileDecodeData functions call this function in call DP0 format.

[Arguments]

| Argument | Description |
|---|---|
| R0 | Size of requested data [bytes] |
| R1 | Offset from start of bit stream data [bytes] |

[Return value]

| Return Value | Description |
|---|---|
| R0 | Size of acquired data [bytes] |
| R2L | Start address of input buffer |

[Function] This function is called from the wmad_FileDecodeInit and wmad_FileDecodeData functions a number of times each to supplement the input buffer with the bit stream data required for WMA decoding.
Set the start address of this function using the wmad_FileDecodeInit function.

[Usable registers] R0, R1, R2, R3, R4, R5, DP0, DP1

**Caution When using registers other than the above, be sure to save their contents to memory before use.**

[Usable hardware resources]

| | |
|---|---|
| Stack level | 0 to 7 |
| Loop stack level | 1 |

[Remark] When using the repeat instruction in this function, the interrupt servicing may be delayed if the number of repetitions is large. Similarly, if the number of execution cycles of this function is large, decoding may not be able to be performed in real time. Also, be sure to set the stack level of this function so that the total stack level of the wmad_FileCBGetData function, wmad_FileDecodeData function, and other functions used by the user does not exceed 15.

[Function request specifications]

- Store data of the requested size (argument: R0 bytes) in the user-defined input buffer in the X memory starting from the position of the offset from the start of the bit stream data (argument: R1 byte number), and set the size of the acquired data to the R0 register and the start address of the input buffer to the R2L register (refer to **Figure 2-2 Setting Data in Input Buffer**). The input buffer that stores the bit stream data must have a capacity of at least 64 words (128 bytes) (refer to **2.5 Memory Configuration**).

- The value of the R0 argument is between 1 and 128 (inclusive) when acquiring the data required for decoding. When this value exceeds 128, bit stream data not required for decoding is skipped, so data does not need to be set in the input buffer. Be sure, however, to set the acquired data size for the R0 return value.

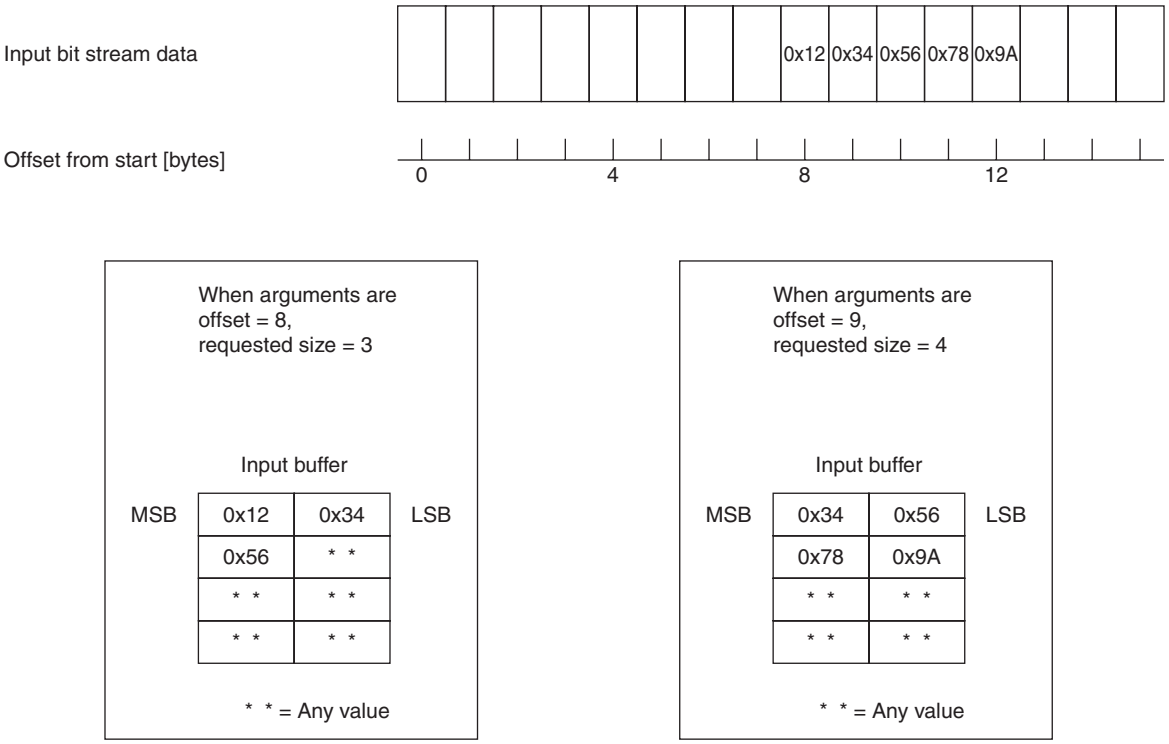
- The value of the R1 argument is always higher than the value of R1 when the wmad_FileCBGetData function was previously called. The R1 value when the wmad_FileCBGetData function is first called is 0. Therefore bit stream data up to the value of R1 when the wmad_FileCBGetData function was previously called is always completely decoded.

- When the value of the R1 argument is equal to or lower than the terminal position (value of previous argument R1 + R0) of the bit stream data read when the wmad_FileCBGetData function was previously called (case A), it is necessary to keep all but the first byte of the data previously set in the input buffer[Note]. Therefore, be sure to save all but the first byte of the data set in the input buffer when the wmad_FileCBGetData function was previously called until that function is next called. Note that if the $\mu$SAP77016-B11 is operating normally, the contents of the input buffer set by the user are never changed.

  **Note** If the previous value of the R0 argument exceeded 128, case A will not occur.

- When the value of the R1 argument is higher than the terminal position (value of previous argument R1 + R0) of the bit stream data read when the wmad_FileCBGetData function was previously called, data in which the offset is less than the value of R1 is never used.

**Figure 2-2. Setting Data in Input Buffer**



## 2.4 Error Information

Details of the error codes returned by the functions of the μSAP77016-B11 are shown in the table below.

**Table 2-3. Error Code List**

| Error | Value | Description |
|---|---|---|
| cWMA_NoErr | 0 | Normal |
| cWMA_Failed | 1 | Other abnormality |
| cWMA_BadArgument | 2 | Initialization was not completed normally |
| cWMA_BadAsfHeader | 3 | Illegal ASF Header |
| cWMA_BadPacketHeader | 4 | Illegal Packet Header |
| cWMA_BrokenFrame | 5 | Not used |
| cWMA_NoMoreFrames | 6 | There is no data to be decoded |
| cWMA_BadSamplingRate | 7 | Not used |
| cWMA_BadNumberOfChannels | 8 | Not used |
| cWMA_BadVersionNumber | 9 | Not used |
| cWMA_BadWeightingMode | 10 | Not used |
| cWMA_BadPacketization | 11 | Not used |
| cWMA_BadDRMType | 12 | Not used |
| cWMA_DRMFailed | 13 | Not used |
| cWMA_DRMUnsupported | 14 | DRM is not supported |
| cWMA_DemoExpired | 15 | Not used |
| cWMA_BadState | 16 | Not used |
| cWMA_Internal | 17 | Internal error |

## 2.5 Memory Configuration

The configuration of the data memory used by the μSAP77016-B11, including details of the static area and how to secure area for buffers, is described here.

With the μSAP77016-B11, the scratch memory area and static memory area must be defined separately. The respective sizes of these areas are shown in **Table 2-4 Memory Size Required for Each Bit Rate**.

**Table 2-4. Memory Size Required for Each Bit Rate (1/2)**

| Bit Rate [bps] | Sampling Frequency [Hz] | Number of Channels | WMA CODEC Version | X Memory Size [Words] | | | Y Memory Size [Words] | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | static_x1 | static_x2 | scratch | static_y1 | static_y2 | scratch |
| 192,000 | 48,000 | 2 | 8 | 741 | 0 | 1,572 | 8,192 | 4,096 | 0 |
| 192,000 | 44,100 | 2 | 7, 8 | 741 | 0 | 1,572 | 8,192 | 4,096 | 0 |
| 160,000 | 48,000 | 2 | 2, 7, 8 | 741 | 0 | 1,572 | 8,192 | 4,096 | 0 |
| 160,000 | 44,100 | 2 | 2, 7, 8 | 741 | 0 | 1,572 | 8,192 | 4,096 | 0 |
| 128,000 | 48,000 | 2 | 2, 7, 8 | 741 | 0 | 1,572 | 8,192 | 4,096 | 0 |
| 128,000 | 44,100 | 2 | 2, 7, 8 | 741 | 0 | 1,572 | 8,192 | 4,096 | 0 |
| 96,000 | 44,100 | 2 | 2, 7, 8 | 741 | 0 | 1,572 | 8,192 | 4,096 | 0 |
| 80,000 | 44,100 | 2 | 2, 7, 8 | 741 | 0 | 1,572 | 8,192 | 4,096 | 0 |
| 64,000 | 44,100 | 2 | 2, 7, 8 | 741 | 0 | 1,572 | 8,192 | 4,096 | 0 |
| 64,000 | 32,000 | 2 | 2 | 741 | 0 | 1,572 | 8,192 | 4,096 | 0 |
| 48,000 | 44,100 | 2 | 2, 8 | 741 | 0 | 1,572 | 8,192 | 4,096 | 0 |
| 48,000 | 32,000 | 2 | 2, 7, 8 | 741 | 0 | 1,572 | 8,192 | 4,096 | 0 |
| 44,000 | 32,000 | 2 | 2 | 972 | 0 | 1,572 | 8,192 | 4,096 | 0 |
| 40,000 | 32,000 | 2 | 2, 7, 8 | 972 | 0 | 1,572 | 8,192 | 4,096 | 0 |
| 36,000 | 32,000 | 2 | 2 | 972 | 0 | 1,572 | 8,192 | 4,096 | 0 |
| 32,000 | 44,100 | 2 | 8 | 972 | 0 | 1,572 | 8,192 | 4,096 | 0 |
| 48,000 | 44,100 | 1 | 8 | 741 | 0 | 1,572 | 8,192 | 2,048 | 0 |
| 32,000 | 44,100 | 1 | 2, 7, 8 | 741 | 0 | 1,572 | 8,192 | 2,048 | 0 |
| 32,000 | 32,000 | 2 | 2, 7, 8 | 972 | 0 | 1,572 | 8,192 | 4,096 | 0 |
| 32,000 | 32,000 | 1 | 2 | 972 | 0 | 1,572 | 8,192 | 2,048 | 0 |
| 32,000 | 22,050 | 2 | 2, 7, 8 | 741 | 0 | 1,572 | 8,192 | 2,048 | 0 |
| 22,000 | 32,000 | 2 | 2 | 972 | 8,192 | 1,572 | 8,192 | 4,096 | 299 |
| 22,000 | 22,050 | 2 | 2, 7 | 972 | 4,096 | 1,572 | 8,192 | 2,048 | 299 |
| 22,000 | 22,050 | 2 | 8 | 972 | 0 | 1,572 | 8,192 | 2,048 | 0 |
| 20,000 | 32,000 | 1 | 2, 7 | 972 | 4,096 | 1,572 | 8,192 | 2,048 | 299 |
| 20,000 | 32,000 | 1 | 8 | 972 | 0 | 1,572 | 8,192 | 2,048 | 0 |
| 20,000 | 22,050 | 2 | 2, 7 | 972 | 4,096 | 1,572 | 8,192 | 2,048 | 299 |
| 20,000 | 22,050 | 2 | 8 | 972 | 0 | 1,572 | 8,192 | 2,048 | 0 |
| 20,000 | 22,050 | 1 | 2, 7 | 972 | 2,048 | 1,572 | 8,192 | 1,024 | 299 |
| 20,000 | 22,050 | 1 | 8 | 972 | 0 | 1,572 | 8,192 | 1,024 | 0 |
| 20,000 | 16,000 | 2 | 2, 7, 8 | 972 | 2,048 | 1,572 | 8,192 | 1,024 | 299 |

**Table 2-4. Memory Size Required for Each Bit Rate (2/2)**

| Bit Rate [bps] | Sampling Frequency [Hz] | Number of Channels | WMA CODEC Version | X Memory Size [Words] | | | Y Memory Size [Words] | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | static_x1 | static_x2 | scratch | static_y1 | static_y2 | scratch |
| 16,000 | 22,050 | 1 | 2, 7 | 972 | 2,048 | 1,572 | 8,192 | 1,024 | 299 |
| 16,000 | 22,050 | 1 | 8 | 972 | 0 | 1,572 | 8,192 | 1,024 | 0 |
| 16,000 | 16,000 | 2 | 2, 7, 8 | 972 | 2,048 | 1,572 | 8,192 | 1,024 | 299 |
| 16,000 | 16,000 | 1 | 2, 7, 8 | 972 | 1,024 | 1,572 | 8,192 | 512 | 299 |
| 12,000 | 16,000 | 1 | 2, 7, 8 | 972 | 1,024 | 1,572 | 8,192 | 512 | 299 |
| 12,000 | 8,000 | 2 | 2, 7, 8 | 972 | 2,048 | 1,572 | 8,192 | 1,024 | 299 |
| 10,000 | 16,000 | 1 | 2, 7, 8 | 972 | 1,024 | 1,572 | 8,192 | 512 | 299 |
| 10,000 | 11,025 | 1 | 2, 7, 8 | 972 | 1,024 | 1,572 | 8,192 | 512 | 299 |
| 8,000 | 11,025 | 1 | 2, 7, 8 | 972 | 1,024 | 1,572 | 8,192 | 512 | 299 |
| 8,000 | 8,000 | 1 | 2, 7, 8 | 972 | 1,024 | 1,572 | 8,192 | 512 | 299 |
| 6,000 | 8,000 | 1 | 2, 7, 8 | 972 | 1,024 | 1,572 | 8,192 | 512 | 299 |
| 5,000 | 8,000 | 1 | 2, 7, 8 | 972 | 1,024 | 1,572 | 8,192 | 512 | 299 |
| 128 | 8,000 | 1 | 2, 7, 8 | 972 | 1,024 | 1,572 | 8,192 | 512 | 299 |

### 2.5.1 Scratch area

This is a memory area that can be freed up and used by the user when the μSAP77016-B11 is not operating. Note, however, that because the scratch area is used once more when the middleware starts operating again, if the user has set information in this area, the set information may be changed at this time.

Secure the scratch area under the label names wmad_lib_Scratch_x and wmad_lib_Scratch_y. It is not necessary to make align or at specifications. The size of the area differs depending on the supported bit rate. For details, refer to **Table 2-4 Memory Size Required for Each Bit Rate**. Be sure to make a PUBLIC declaration for defined symbols.

**Figure 2-3. Example of Securing Scratch Area**

```
public      wmad_lib_Scratch_x
public      wmad_lib_Scratch_y

;When all bit rates are supported
#define WMAD_MAX_SCRATCH_X_SIZE 1572
#define WMAD_MAX_SCRATCH_Y_SIZE 299


__WMAD_LIB_SCRATCH_X     XRAMSEG
wmad_lib_Scratch_x:      DS  WMAD_MAX_SCRATCH_X_SIZE;

__WMAD_LIB_SCRATCH_Y     YRAMSEG
wmad_lib_Scratch_y:      DS  WMAD_MAX_SCRATCH_Y_SIZE;

end
```

### 2.5.2 Static area

This area is used to store data permanently. If this area is manipulated by the user following initialization processing, the normal operation of this library cannot be guaranteed.

Secure the static area under the label names wmad_lib_Static_x1, wmad_lib_Static_x2, wmad_lib_Static_y1, and wmad_lib_Static_y2. It is not necessary to make align or at specifications. The size of the area differs depending on the supported bit rate. For details, refer to **Table 2-4 Memory Size Required for Each Bit Rate**. Be sure to make a PUBLIC declaration for defined symbols.

**Figure 2-4. Example of Securing Static Area**

```
public      wmad_lib_Static_x1
public      wmad_lib_Static_x2
public      wmad_lib_Static_y1
public      wmad_lib_Static_y2

;When all bit rates are supported
#define WMAD_MAX_STATIC_X1_SIZE 972
#define WMAD_MAX_STATIC_X2_SIZE 8192
#define WMAD_MAX_STATIC_Y1_SIZE 8192
#define WMAD_MAX_STATIC_Y2_SIZE 4096

__WMAD_LIB_STATIC_X1     XRAMSEG
wmad_lib_Static_x1:      DS   WMAD_MAX_STATIC_X1_SIZE;
__WMAD_LIB_STATIC_X2     XRAMSEG
wmad_lib_Static_x2:      DS   WMAD_MAX_STATIC_X2_SIZE;

__WMAD_LIB_STATIC_Y1     YRAMSEG
wmad_lib_Static_y1:      DS   WMAD_MAX_STATIC_Y1_SIZE;
__WMAD_LIB_STATIC_Y2     YRAMSEG
wmad_lib_Static_y2:      DS   WMAD_MAX_STATIC_Y2_SIZE;

end
```

### 2.5.3 I/O buffers

In order for the μSAP77016-B11 to perform decoding, an input buffer is required to input the bit stream data (X memory) and an output buffer is required to store the PCM data of the decoding results (X memory). Any symbol name can be assigned to these buffers, providing names used by the μSAP77016-B11 or other applications are not duplicated.

**Figure 2-5. Example of Securing I/O Buffers**

```
public      INPUT_BUFFER
public      HOST_IN_BUFF
public      OUTPUT_BUFFER
public      SERIAL_OUT_BUFF

I_O_BUFFER_X       XRAMSEG
INPUT_BUFFER:      DS     64;
HOST_IN_BUFF:      DS     1536;
OUTPUT_BUFFER:     DS     4096;
SERIAL_OUT_BUFF:   DS     4096;

end
```

The input buffer must have a capacity of at least 64 words and the size of the output buffer should be set to accord with the system configuration. Recommended output buffer sizes are shown in **Table 2-5 Recommended Output Buffer Size**.

Note that when performing decoding in real time, buffers such as an input data receive buffer to receive the bit stream data sent from the host CPU (X or Y memory), and a serial output buffer to output PCM data to the DAC (X or Y memory) are required in addition to the I/O buffers used by the μSAP77016-B11.

The size of the serial output buffer must be equivalent to the buffer size shown in **Table 2-5 Recommended Output Buffer Size**.

Set the size of the input data receive buffer to accord with the system configuration, based on the average amount of input data calculated from the equation below. For example, to store the input data required to obtain the output results of the recommended output buffer size × 2 samples, the size of the input data receive buffer should be the average input data amount calculated using the equation below, with the appropriate value from **Table 2-5 Recommended Output Buffer Size** as the number of output samples, × 2 words. The average input data amount calculated in this way will reach a maximum value of 558 words when the bit rate is 192 kbps and the sampling frequency is 44.1 kHz.

$$\text{Average input data amount} = \frac{\text{Bit rate [kbps]}/16}{\dfrac{\text{Sampling frequency [kHz]}}{\text{Number of output samples}}} \text{ [Words]}$$

Note, however, that if data that includes video data is received in addition to audio data, there may be insufficient data in the buffer, even if a larger buffer is secured. To counter this problem, it is recommended to construct a system in which a command requesting input data can be sent from the DSP side to the CPU.

**Table 2-5. Recommended Output Buffer Size**

| Sampling Frequency [Hz] | Size [Words] | |
|---|---|---|
| | Mono | Stereo |
| 8,000 | 512 | 1,024 |
| 11,025 | 512 | 1,024 |
| 16,000 | 512 | 1,024 |
| 22,050 | 1,024 | 2,048 |
| 32,000 | 2,048 | 4,096 |
| 44,100 | 2,048 | 4,096 |
| 48,000 | 2,048 | 4,096 |

### 2.5.4 Structures

The following structures must be prepared in order to acquire WMA file information or contents information.

#### (1) File information structure

To acquire WMA file information, prepare the following structure in the X memory and execute the wmad_FileDecodeInfo function.

**Table 2-6. Members of File Information Structure**

| Item | Size [Words] |
|---|---|
| Version of WMA file format[Note] | 1 |
| Sampling frequency [Hz] | 1 |
| Number of channels | 1 |
| Playback time [ms] | 2 |
| Packet size [bytes] | 2 |
| Offset to first packet [bytes] | 2 |
| Offset to last packet [bytes] | 2 |
| Use of DRM 0: DRM not used, 1: DRM used | 2 |
| Bit rate [bps] | 2 |

**Note** The μSAP77016-B11 only supports data of version 2 of the WMA file format. Data encoded using WMA CODEC Version 2, 7, or 8 is version 2 of the WMA file format.

#### (2) Contents information structure

To acquire information such as the title of a track, prepare the following contents information structure and contents character string area in the X memory and execute the wmad_FileDecodeInit function.

**Table 2-7. Members of Contents Information Structure**

| Item | Size [Words] |
|---|---|
| Maximum length of title character string [bytes] | 1 |
| Maximum length of creator character string [bytes] | 1 |
| Maximum length of writer character string [bytes] | 1 |
| Maximum length of explanatory note character string [bytes] | 1 |
| Maximum length of regulation character string [bytes] | 1 |
| Start address of title character string area | 1 |
| Start address of creator character string area | 1 |
| Start address of writer character string area | 1 |
| Start address of explanatory note character string area | 1 |
| Start address of regulation character string area | 1 |

Specify the maximum length (requested number of characters) of the contents information as an even number. If the contents information of the bit stream data is less than the requested number of characters, the length of the character string actually acquired (including the terminal code “0x0000”) is stored as the maximum length of the character string in the structure.

# CHAPTER 3 INSTALLATION

## 3.1 Installation Procedure

Install the μSAP77016-B11 (WMA decoder middleware) in the host machine following the procedure shown below.

(1) Create a work directory in the host machine.

(2) Copy all the files and directories on the supplied medium to the work directory in the host machine.

## 3.2 Sample Creation Procedure

An example of how to build the sample program of the μSAP77016-B11 is shown below.

(1) Start up the WB77016 (workbench).

(2) Open the sample.prj project file.

**Example** Select [Project → Open Project] and specify sample.prj.
If the error “Cannot Load system ~*.model” occurs, select [Options → Processor Model] and specify the appropriate model file.
If the model file has been changed, change the constant data library file of the project to accord with the model file.

**Example** In the case of a model with ROM, change wmad_ram.lib to wmad_rom.lib.

(3) Execute build and confirm that sample.lnk has been created.

**Example** The sample.lnk file is created by selecting [Make → Build All].

## 3.3 Change of Location

The segment names used by the μSAP77016-B11 are shown in **Table 3-1 Segment Names**. The location can be changed to accord with the user's target by separating the object files from the library files using the LB77016 (librarian) and utilizing the edit segment function of the WB77016 (workbench).

Note that this library and the static and scratch areas used by this library must not be allocated to external memory.

**Table 3-1. Segment Names**

| Segment Name | Description |
|---|---|
| _ _WMAD_IMSEG* | μSAP77016-B11 instruction segment |
| _ _WMAD_XROM* | μSAP77016-B11 X memory constant data segment |
| _ _WMAD_YROM* | μSAP77016-B11 Y memory constant data segment |

**Remark** An asterisk (*) indicates an arbitrary alphanumeric character.

## 3.4 Symbol Naming Conventions

The conventions that apply when naming symbols used in this library are shown in **Table 3-2 Naming Conventions**. Take care not to duplicate names when using the μSAP77016-B11 together with other applications.

**Table 3-2. Naming Conventions**

| Classification | Convention |
|---|---|
| Function name, variable name | wmad_XXXX |
| Segment name | _ _WMAD_XXXX (2 underscores at the start) |

**Remark** XXXX indicates arbitrary alphanumeric characters.

# CHAPTER 4 SYSTEM EXAMPLE

## 4.1 Environment Required for Simulation Using Timing Files

The decode processing can be simulated by using the sample program and sample timing files.

**[Example of software environment]**

- High-speed simulator: HSM77016 Ver.2.32 or later
- Sample program: sample.lnk (program created in **3.2 Sample Creation Procedure**)
- Timing files: smp_input.tmg, smp_serout.tmg, clk_for_2ch.tmg
- Input data file: xxx.dat (see **4.2 Input Data File Creation** for how to create this file)
- Model file: uPD77113.model (model file used when sample program was created)

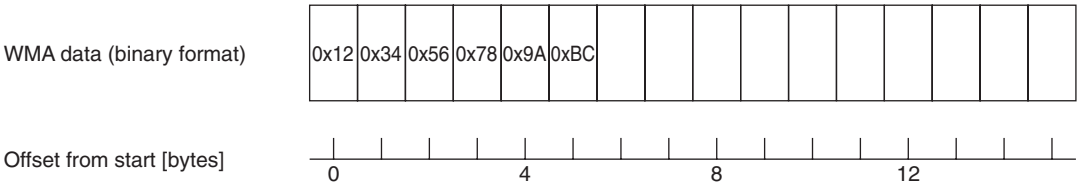
## 4.2 Input Data File Creation

An input data file is required to perform simulation. Create the input data file following the procedure shown below.

(1) Prepare arbitrary WMA data (binary format).

(2) Convert the prepared WMA data into a text file. In this text file, describe two bytes of WMA data per line in order from the start of the data.

**Figure 4-1. Input Data File**




Input data text file
(text format)

MSB

```
0x1234
0x5678
0x9ABC
:
:
```

LSB

## 4.3 Simulation

An example of how to perform simulation is shown below.

(1) Edit the timing file to accord with the prepared input data file (refer to **A.3 Sample Timing Files**).

(2) Start up the HSM77016 (high-speed simulator).

(3) Select the model file in accordance with the target.

**Example** Select [tools → Simulation Model] and specify the model file.

(4) Open sample.lnk created in **3.2 Sample Creation Procedure**.

**Example** Select [file → open] and specify sample.lnk.

(5) Open the timing files smp_input.tmg, smp_serout.tmg, and clk_for_2ch.tmg.

**Example** Select [file → open] and specify each file.

(6) Reset the CPU and timing files of the HSM77016 (high-speed simulator).

**Example** Select [run → reset], specify all the items (CPU and built-in I/O devices, time measurement, all timing files and restart execution), and reset.

(7) Select Run to execute.

## 4.4 Sample Program Outline

### 4.4.1 Sample program

The sample program of the μSAP77016-B11 is designed based on a system in which input data is transmitted to the DSP via a host interface. Input data is transmitted sequentially from the start of the data in 16-bit units. The decode processing unit (number of samples) is the value indicated in **Table 2-5 Recommended Output Buffer Size**.

### 4.4.2 User-defined functions

An outline of the user-defined function sample source is shown below.

**[Variables]**

**(1) FileCBGetData_fp variable**

This variable is used to manage the number of bytes of input data received via the host interface. In this system, input data is received in 2-byte units, so this will always be an even number.

The value of this variable, x, also means that the offset from the start of the first input data received via the host interface in the wmad_FileCBGetData function called this time is data of x and x + 1. Note that the value of this variable is not “the value of argument R1 + R0 when the wmad_FileCBGetData function was previously called”.

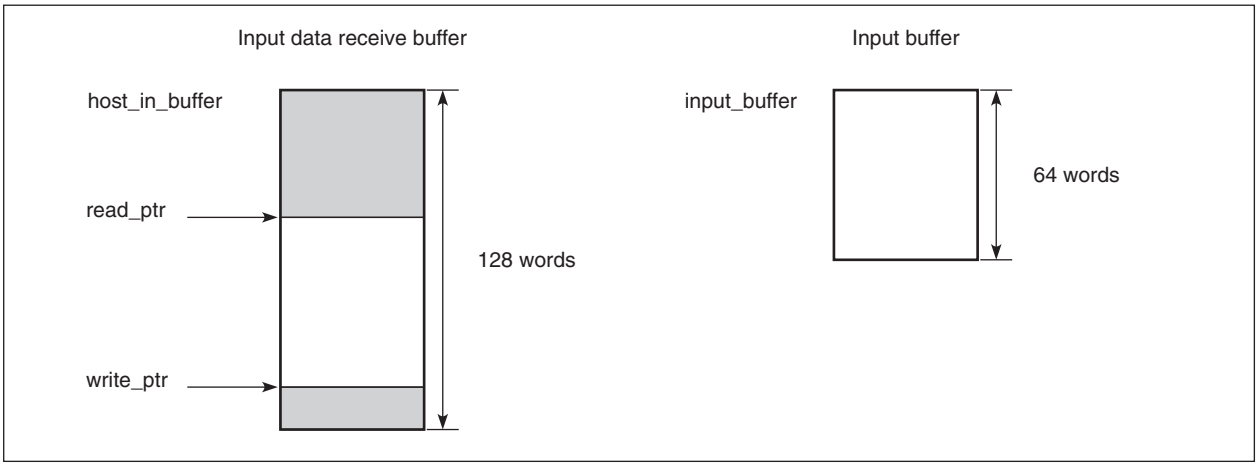
**(2) read_ptr variable**

This variable is used to manage the position at which data starts to be read from the input data receive buffer when data is set in the input buffer.

**(3) write_ptr variable**

This variable is used to manage the position at which input data received via the host interface starts to be written to the input data receive buffer.

**Figure 4-2. Configuration of Sample User-Defined Function Buffer**



**[Processing]**

The value of the FileCBGetData_fp variable is compared with the value r1 of the requested offset position and branching to the following label occurs depending on the result:

Branch to the case000 label if FileCBGetData_fp = r1
Branch to the case001 label if FileCBGetData_fp < r1
Branch to the case002 label if FileCBGetData_fp > r1

- case000 label:
  Branching to this label occurs if either of the first two bytes of input data received first via the host interface matches the requested offset data. The value of the write_ptr variable is then set to the read_ptr variable. The size of the data to be received via the host interface in the processing following the get_data label is also set, and the program jumps to the get_data label.

- case001 label:
  Branching to this label occurs if data in front of the input data already received is requested. In this case, the data up to the requested offset data is received via the host interface. The data received here is not used and is therefore not saved in the input data receive buffer.
  The value of the write_ptr variable is then set to the read_ptr variable. The size of the data to be received is also set, and the program jumps to the get_data label.

- case002 label:
  Branching to this label occurs if the requested offset data has already been received. In this case, the read_ptr variable is set, but because the data already received is set in the input buffer, the position that is the value of the write_ptr variable rewound by the required number is set.
  Next, if data of the requested size can be set in the input buffer from the data already received, the program jumps to the set_data label, and if data of the requested size cannot be set in the input buffer from the data already received, a new receive data size is set and the program jumps to the get_data label.

- get_data label:
  If the requested size is 128 bytes or less, the program jumps to the get_data_next label. In other cases, input data is received via the host interface, but this data is not used and is therefore not saved in the receive buffer. When the receive processing is finished, the program jumps to the finish label.

- get_data_next label:
  If the size of the data that should be received is 0, the program jumps to the set_data label. In other cases, the input data is received via the host interface and stored from the position indicated by the write_ptr variable. When the receive processing is finished, the write_ptr variable is updated and the program performs set_data label processing.

- set_data label:
  The input data stored in the input data receive buffer is read out from the position indicated by the read_ptr variable and set in the input buffer in accordance with the requested conditions. The program then performs finish label processing.

- finish label:
  After processing such as setting the return values and updating the FileCBGetData_fp variable is finished, the program returns to the caller from the wmad_FileCBGetData function processing.

## 4.5 Sample Program Processing Flow

The sample program processing flow is shown below.

**Figure 4-3. Initialization and Decode Processing Flow**

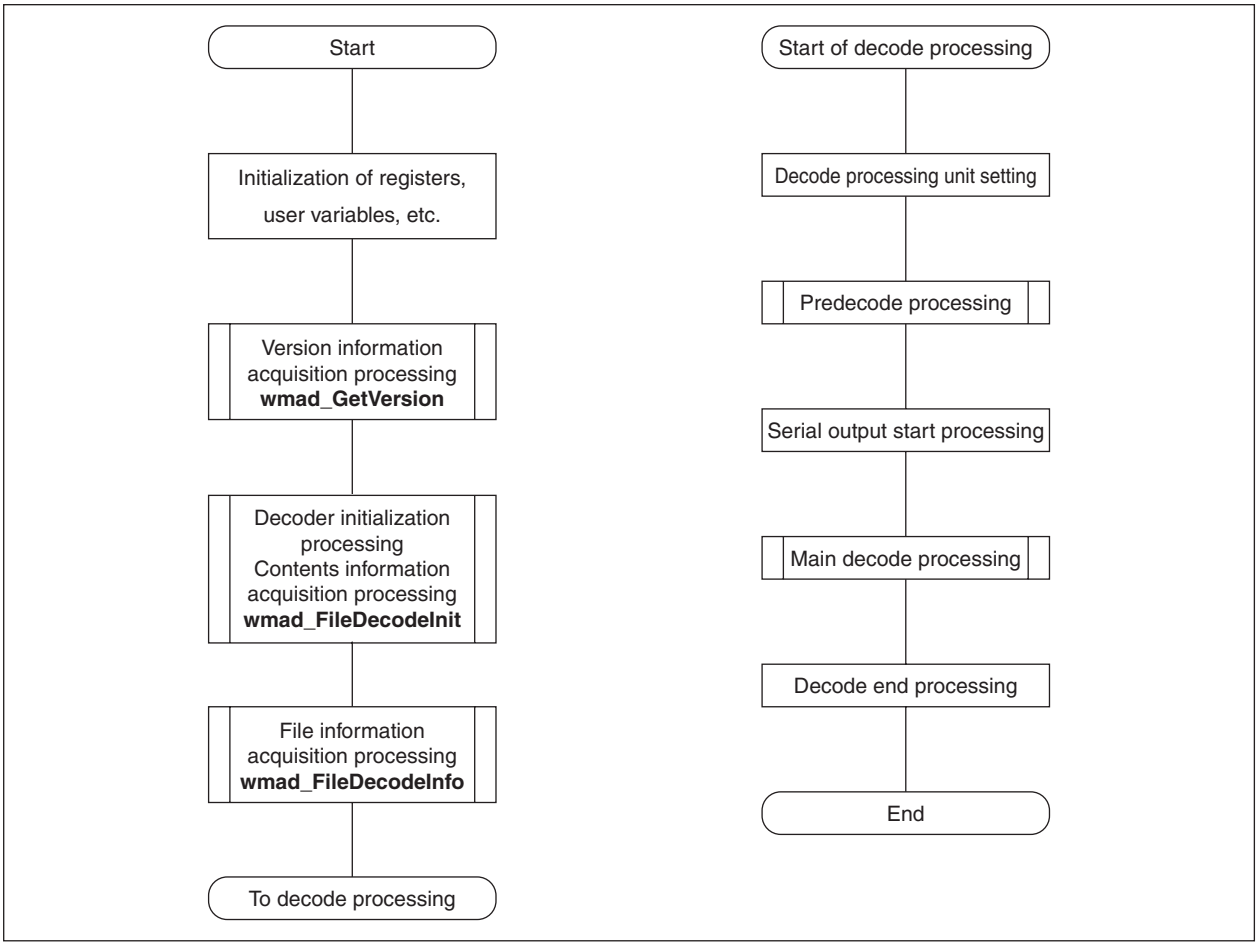


**Figure 4-4. Predecode Processing Flow**

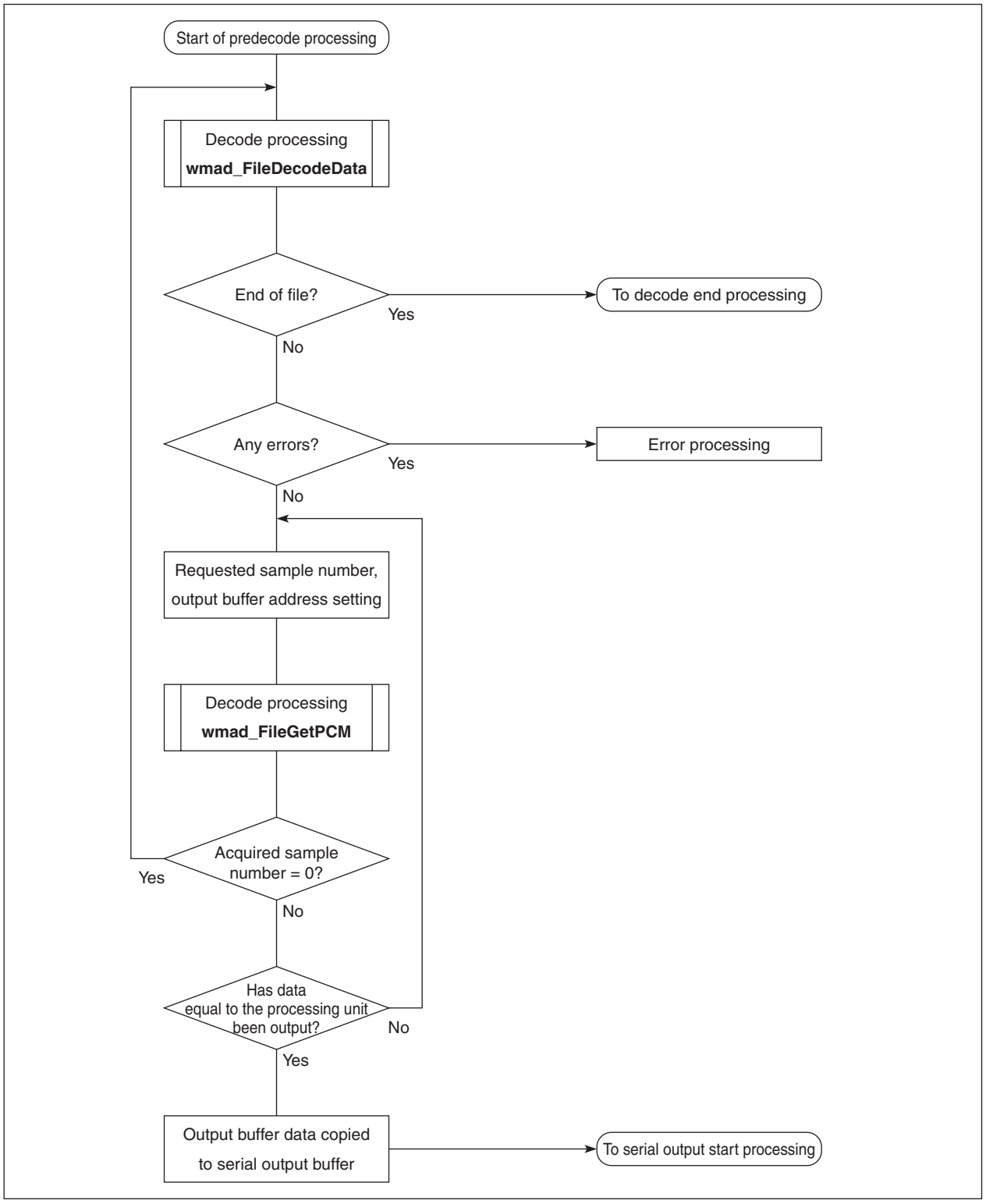
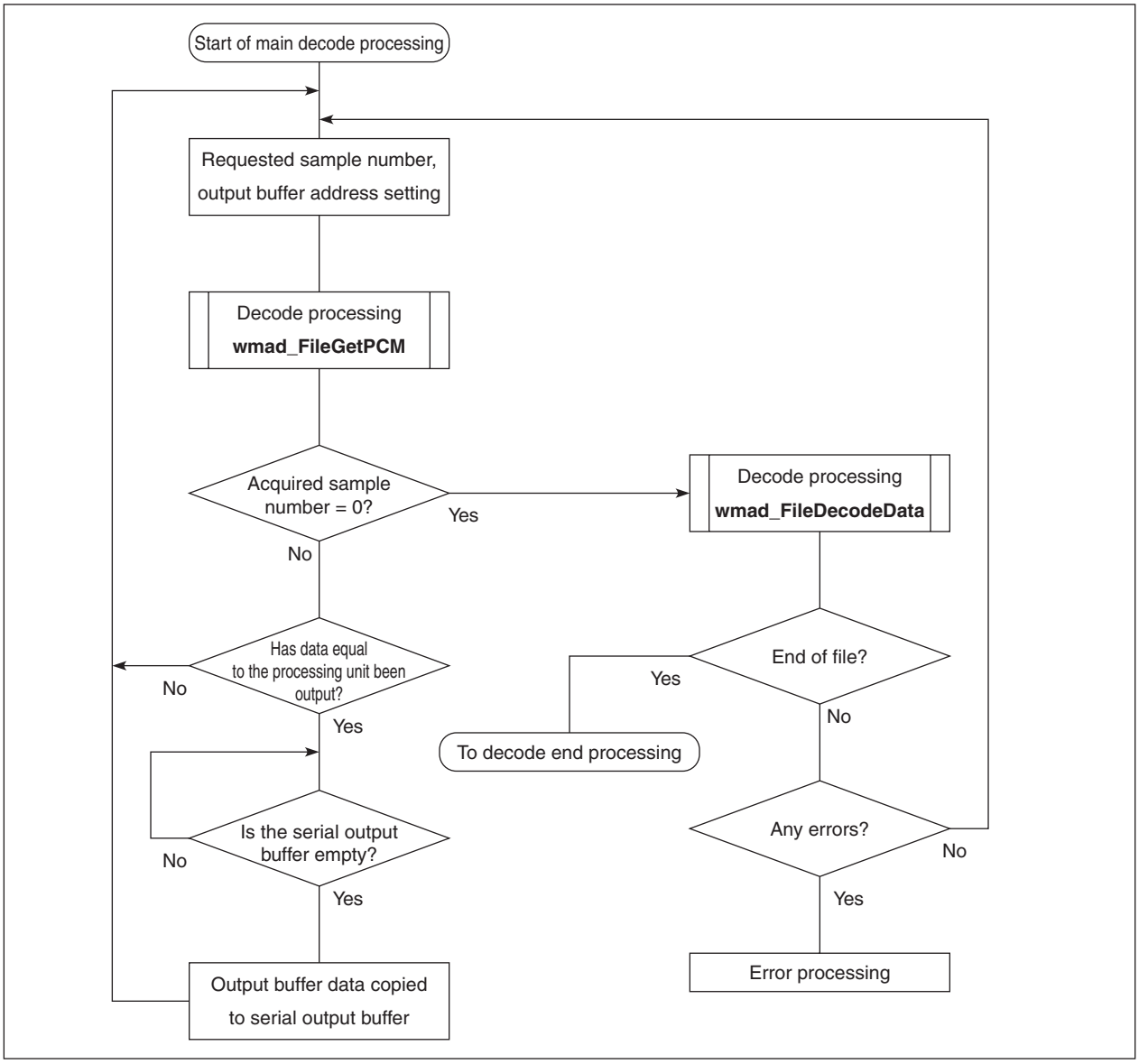


**Figure 4-5. Main Decode Processing Flow**



# APPENDIX A SAMPLE PROGRAM SOURCE

The sample program source of the μSAP77016-B11 is shown in this chapter.

## A.1 Sample Source Files

### A.1.1 sample.asm

This file is used to control the overall decode processing.

(1/13)

```
/*-----------------------------------------------------------------*/
/*   File Information                                              */
/*-----------------------------------------------------------------*/
/*   Name      : sample.asm                                        */
/*   Type      : Assembler program module                          */
/*   Version   : 1.00                                              */
/*   Date      : 2001 July 10                                      */
/*   CPU       : uPD7701x Family                                   */
/*   Assembler : WB77016 Ver 2.4                                   */
/*   About     : sample main function                              */
/*                                                                 */
/*-----------------------------------------------------------------*/
/*   Copyright (C) NEC Corporation 2000, 2001                      */
/*   All rights reserved by NEC Corporation.                       */
/*   Use of copyright notice does not evidence publication         */
/*-----------------------------------------------------------------*/


/* ================================================================
 *   INCLUDE FILES
 * ================================================================ */
#include "wma_dec.h"

/* ================================================================
 *   PUBLIC
 * ================================================================ */
public Num_Channels                ; for timing file
public sample_rate                 ; for timing file

/* ================================================================
 *   EXTERN FUNCTIONS
 * ================================================================ */
extrn  Init_FileCBGetData
extrn  wmad_FileCBGetData
```

(2/13)

```
/* ============================================================
 *   DEFINE & EQU
 * ============================================================ */
#define USE_SO       1              ; Use Serial Output interrupt
#define STRING_SIZE  64
#define MAX_PCMSIZE  2048
MAX_RINGSIZE   equ   MAX_PCMSIZE*2

/* ============================================================
 *      LOCAL VARIABLES AND BUFFER
 * ============================================================ */
__SAMPLE_X_RAM  XRAMSEG
ring_dn0:            ds      1              ;
ring_write_ptr:      ds      1              ;
ring_read_ptr:       ds      1              ;
ring_entries:        ds      1              ;
putin_temp:          ds      2              ;
save_regs:           ds      6              ;
g_ulOutputSample:    ds      2              ;
start_flag:          ds      1              ;
n_sample:            ds      1              ;
f_dec_unit_end:      ds      1              ;
n_get_pcm:           ds      1              ;


__SAMPLE_X_CONTENTS XRAMSEG
/*****************************************************
Contents Information
*****************************************************/
desc:
desc_title_len:              ds      1              ;
desc_author_len:             ds      1              ;
desc_copyright_len:          ds      1              ;
desc_description_len:        ds      1              ;
desc_rating_len:             ds      1              ;

desc_pTitle:                 ds      1              ;
desc_pAuthor:                ds      1              ;
desc_pCopyright:             ds      1              ;
desc_pDescription:           ds      1              ;
desc_pRating:                ds      1              ;

Title:                       ds      STRING_SIZE/2  ;
Author:                      ds      STRING_SIZE/2  ;
Copyright:                   ds      STRING_SIZE/2  ;
Description:                 ds      STRING_SIZE/2  ;
Rating:                      ds      STRING_SIZE/2  ;

StructFileInfo:
Version:                     ds      1              ;
```

(3/13)

```
sample_rate:         ds      1               ;
Num_Channels         ds      1               ;
duration:            ds      2               ;
packet_size:         ds      2               ;
first_packet_offset: ds      2               ;
last_packet_offset:  ds      2               ;
has_DRM:             ds      2               ;
bit_rate:            ds      2               ;

__SAMPLE_X_DEC_OUTPUT XRAMSEG
output_buffer:       ds      MAX_PCMSIZE*2   ;

__SAMPLE_X_SER_OUTPUT XRAMSEG align at 0
ser_out_buffer:      ds      MAX_RINGSIZE    ;



/* ============================================================
 *   VECTOR REGISTRATION
 * ============================================================ */
MAIN_V IMSEG at 0x200
     jmp star                                ; Regist start up routine
     nop                                     ;
     nop                                     ;
     nop                                     ;

; reserve vector 1
     nop                                     ;
     reti                                    ;
     nop                                     ;
     nop                                     ;

; reserve vector 2
     nop                                     ;
     reti                                    ;
     nop                                     ;
     nop                                     ;

; reserve vector 3
     nop                                     ;
     reti                                    ;
     nop                                     ;
     nop                                     ;

; int1 vector
     nop                                     ;
     reti                                    ;
     nop                                     ;
     nop                                     ;
```

(4/13)

```
; int2 vector
     nop                                ;
     reti                               ;
     nop                                ;
     nop                                ;

; int3 vector
     nop                                ;
     reti                               ;
     nop                                ;
     nop                                ;

; int4 vector
     nop                                ;
     reti                               ;
     nop                                ;
     nop                                ;

; SI1 vector
     nop                                ;
     reti                               ;
     nop                                ;
     nop                                ;

; SO1 vector
     jmp  _so_interrupt                 ; Regist SO1 handler
     reti                               ;
     nop                                ;
     nop                                ;

; SI2 vector
     nop                                ;
     reti                               ;
     nop                                ;
     nop                                ;

; SO2 vector
     nop                                ;
     reti                               ;
     nop                                ;
     nop                                ;

; HI vector
     nop                                ;
     reti                               ;
     nop                                ;
     nop                                ;

; HO vector
     nop                                ;
```

(5/13)

```
     reti                                ;
     nop                                 ;
     nop                                 ;

; Hardware signal vector
     nop                                 ;
     reti                                ;
     nop                                 ;
     nop                                 ;

; Timer vector
     nop                                 ;
     reti                                ;
     nop                                 ;
     nop                                 ;


/* ================================================================
 *    PROGRAM CODE
 * ================================================================ */
MAIN IMSEG AT 0x240                      ;
start:
     clr(r0)                             ;
     r0l = EIR                           ;
     r0  = r0 | 0x8000                   ;
     EIR = r0l                           ; disable interrupt

     ;;=========================================================;;
     ;;      Clear Register
     ;;=========================================================;;
     clr(r0)                             ;
     clr(r1)                             ;
     clr(r2)                             ;
     clr(r3)                             ;
     clr(r4)                             ;
     clr(r5)                             ;
     clr(r6)                             ;
     clr(r7)                             ;
     dp0 = r0l                           ;
     dp1 = r0l                           ;
     dp2 = r0l                           ;
     dp3 = r0l                           ;
     dp4 = r0l                           ;
     dp5 = r0l                           ;
     dp6 = r0l                           ;
     dp7 = r0l                           ;
     dn0 = r0l                           ;
     dn1 = r0l                           ;
     dn2 = r0l                           ;
     dn3 = r0l                           ;
```

(6/13)

```
dn4 = r0l                              ;
dn5 = r0l                              ;
dn6 = r0l                              ;
dn7 = r0l                              ;
DMX = r0l                              ;
DMY = r0l                              ;
;;=========================================================;;
;;      Initialize Register & Peripheral Units
;;=========================================================;;
r0l = 0x0000                           ;
*DWTR:x=r0l                            ;

r0l = 0x0081                           ;
*HST:x = r0l                           ;

r0l = 0x0202                           ;
*SST1:x = r0l                          ;

;;=========================================================;;
;;      Initialize Variables and Buffer
;;=========================================================;;
dp0 = ser_out_buffer                   ;
clr(r0)                                ;
rep MAX_RINGSIZE                       ;
        *dp0++ = r0h                   ;
r0l = ser_out_buffer                   ;
*ring_write_ptr:x = r0l                ;
*ring_read_ptr:x  = r0l                ;
*ring_entries:x = r0h                  ; set 0
r0l = 1                                ;
*ring_dn0:x = r0l                      ;

*g_ulOutputSample:x   = r0h            ;
*g_ulOutputSample+1:x = r0h            ;

r0l = Title                            ;
*desc_pTitle:x = r0l                   ;
r0l = Author                           ;
*desc_pAuthor:x = r0l                  ;
r0l = Copyright                        ;
*desc_pCopyright:x = r0l               ;
r0l = Description                      ;
*desc_pDescription:x = r0l             ;
r0l = Rating                           ;
*desc_pRating:x = r0l                  ;
r0l = STRING_SIZE                      ;
*desc_title_len:x = r0l                ;
*desc_author_len:x = r0l               ;
*desc_copyright_len:x = r0l            ;
```

(7/13)

```
     *desc_description_len:x = r0l          ;
     *desc_rating_len:x = r0l                ;

     call Init_FileCBGetData                ;


     ;;=========================================================;;
     ;;      Get Version Information
     ;;=========================================================;;
     call wmad_GetVersion                   ;

     ;;=========================================================;;
     ;;      Initialize WMA Decoder and Get Content Information
     ;;=========================================================;;
     r0l = wmad_FileCBGetData               ;
     clr(r1)                                ;
     r1l = 0x0001                           ;
     r2l = desc                             ;
     call wmad_FileDecodeInit               ;
     r0 = r0 ^ cWMA_NoErr                   ;
     if(r0!=0) jmp _init_error              ;

     ;;=========================================================;;
     ;;      Get File Information
     ;;=========================================================;;
     r0l = StructFileInfo                   ;
     call wmad_FileDecodeInfo               ;


     ;;=========================================================;;
     ;;      Set Size of Unit for Decode Process
     ;;=========================================================;;
     clr(r0)                                ;
     r0l = *sample_rate:x                   ;
     r1 = r0 - 32000                        ;
     if(r1>=0) jmp _pre_set_size_2048       ;
     r1 = r0 - 22050                        ;
     if(r1>=0) jmp _pre_set_size_1024       ;
             r1l = 512                      ;
             jmp _pre_set_size_end          ;
_pre_set_size_1024:
             r1l = 1024                     ;
             jmp _pre_set_size_end          ;
_pre_set_size_2048:
             r1l = 2048                     ;
_pre_set_size_end:
     *n_sample:x = r1l                      ; sample per ch

     clr(r0)                                ;
     *start_flag:x = r0l                    ; set *start_flag:x = 0
```

(8/13)

```
     *f_dec_unit_end:x = r0l                ;
     *n_get_pcm:x = r0l                     ;


     ;;=========================================================;;
     ;;      Previous Decode routine
     ;;=========================================================;;
_pre_DecodeData:
     call    wmad_FileDecodeData            ;

     /*** decode is finished ? ***/
     r1 = r0 ^  cWMA_NoMoreFrames           ;
     if(r1==0) jmp finish                   ;
     r1 = r0 ^  cWMA_Failed                 ;
     if(r1==0) jmp finish                   ;

     /*** check error ***/
     r1 = r0 ^ cWMA_NoErr                   ;
     if(r1!=0) jmp _decode_error            ;

_pre_GetPCM:
     r2 = *n_get_pcm:x                      ;
     r3 = *n_sample:x                       ;
     r1 = r3 - r2                           ;
     r1 = r1 sra 16                         ; set r1
     r2  = r2 sra 16                        ;
     clr(r4)                                ;
     r4l = *Num_Channels:x                  ; Set r4 = Number of ch
     r4  = r4 - 2                           ;
     if(r4==0) r2 += r2                     ;
     r0l = output_buffer                    ;
     r0  = r0 + r2                          ; set r0

     call    wmad_FileGetPCM                ;
     if(r1==0) jmp _pre_DecodeData          ;

     clr(r2)                                ;
     r2h= *g_ulOutputSample:x               ;
     r2l= *g_ulOutputSample+1:x             ;
     r2 = r2 + r1                           ; count total sample per ch
     *g_ulOutputSample:x = r2h              ;
     *g_ulOutputSample+1:x = r2l            ;

     clr(r2)                                ;
     r2l = *n_get_pcm:x                     ;
     r2  = r2 + r1                          ;
     clr(r3)                                ;
     r3l = *n_sample:x                      ;
     r3  = r3 - r2                          ;
     if(r3>0) jmp _pre_GetPCM_end           ;
```

(9/13)

```
            r2l = 1                         ;
            *f_dec_unit_end:x = r2l         ; f_dec_unit_end = 1
            *start_flag:x = r2l             ; set *start_flag:x = 1
            clr(r2)                         ;
_pre_GetPCM_end:
     *n_get_pcm:x = r2l                     ;


     /* chekc decode unit is end ? */
     r2 = *f_dec_unit_end:x                 ;
     if(r2==0) jmp _pre_GetPCM              ;
#if USE_SO
     call _SetPCM                           ; set PCM
#else
     *f_dec_unit_end:x =  r2l               ; f_dec_unit_end = 0
#endif

     ;;=========================================================;;
     ;;      Set interrupt mask and Start Serial Output
     ;;=========================================================;;
serout_start:
     *SDT1:x = r1h                          ; serial out start
     r0l = SR                               ;
     r0  = r0 & 0x7fdf                      ;
     r0  = r0 | 0x0fdf                      ;
     nop                                    ;
     sr  = r0l                              ; enable interrupt (SO1)
     jmp _loop_GetPCM                       ;


     ;;=========================================================;;
     ;;      Main routine
     ;;=========================================================;;
_loop_DecodeData:
     call    wmad_FileDecodeData            ;

     /*** decode is finished ? ***/
     r1 = r0 ^  cWMA_NoMoreFrames           ;
     if(r1==0) jmp finish                   ;
     r1 = r0 ^  cWMA_Failed                 ;
     if(r1==0) jmp finish                   ;

     /*** check error ***/
     r1 = r0 ^ cWMA_NoErr                   ;
     if(r1!=0) jmp _decode_error            ;
     r1 = *ring_entries:x                   ;
     if(r1<0) jmp _mips_overflow            ;

_loop_GetPCM:
     r2 = *n_get_pcm:x                      ;
```

(10/13)

```
     r3 = *n_sample:x                        ;
     r1 = r3 - r2                            ;
     r1 = r1 sra 16                          ; set r1
     r2  = r2 sra 16                         ;
     clr(r4)                                 ;
     r4l = *Num_Channels:x                   ; Set r4 = Number of ch
     r4  = r4 - 2                            ;
     if(r4==0) r2 += r2                      ;
     r0l = output_buffer                     ;
     r0  = r0 + r2                           ; set r0

     call    wmad_FileGetPCM                 ;
     r2 = *ring_entries:x                    ;
     if(r2<0) jmp _mips_overflow             ;
     if(r1==0) jmp _loop_DecodeData          ;

     clr(r2)                                 ;
     r2h= *g_ulOutputSample:x                ;
     r2l= *g_ulOutputSample+1:x              ;
     r2 = r2 + r1                            ;  count total sample per ch
     *g_ulOutputSample:x = r2h               ;
     *g_ulOutputSample+1:x = r2l             ;

     clr(r2)                                 ;
     r2l = *n_get_pcm:x                      ;
     r2  = r2 + r1                           ;
     clr(r3)                                 ;
     r3l = *n_sample:x                       ;
     r3  = r3 - r2                           ;
     if(r3>0) jmp _loop_GetPCM_end           ;
             r2l = 1                         ;
             *f_dec_unit_end:x = r2l         ; f_dec_unit_end = 1
             clr(r2)                         ;
_loop_GetPCM_end:
     *n_get_pcm:x = r2l                      ;

     /* check decode unit is end ? */
     r2 = *f_dec_unit_end:x                  ;
     if(r2==0) jmp _loop_GetPCM              ;
     nop                                     ;

_loop_wait:
     r2 = *ring_entries:x                    ;
     if(r2<0) jmp _mips_overflow             ;
     if(r2!=0) jmp _loop_wait                ;
#if USE_SO
     call _SetPCM                            ; set PCM
#else
     *f_dec_unit_end:x =  r2l                ; f_dec_unit_end = 0
#endif
```

(11/13)

```
     jmp _loop_GetPCM                        ;


     ;;=========================================================;;
     ;;      Finish
     ;;=========================================================;;
finish:
     r2 = *ring_entries:x                    ;
     if(r2>0) jmp $-1                        ;
     r2 = *n_get_pcm:x                       ;
     if(r2==0) jmp serout_finish             ;
     *n_sample:x = r2h                       ;
#if USE_SO
     call _SetPCM                            ; set PCM
#else
     *f_dec_unit_end:x =  r2l                ; f_dec_unit_end = 0
#endif
     r2 = *ring_entries:x                    ;
     if(r2>0) jmp $-1                        ;
serout_finish:
     nop                                     ;
     jmp serout_finish                       ;


     ;;=========================================================;;
     ;;      Error
     ;;=========================================================;;
_init_error:
     nop                                     ;
     jmp $-1                                 ;

_decode_error:
     nop                                     ;
     jmp $-1                                 ;

_mips_overflow:
     nop                                     ;
     jmp $-1                                 ;




/* ================================================================
[Function Name] _SetPCM
 ================================================================*/
_SetPCM:
     dp0 = output_buffer                     ;
     r2l = *ring_write_ptr:x                 ;
     dp1 = r2l                               ;
     dn1 = 1                                 ;
```

(12/13)

```
     dmx = MAX_RINGSIZE-1                    ;

     clr(r4)                                 ;
     r4l = *Num_Channels:x                   ; Set r4 = Number of ch
     r4  = r4 - 1                            ;
     clr(r3)                                 ;
     r3l = *n_sample:x                       ;
     Loop r3l {                              ;
            r2 = *dp0++                      ;
            *dp1%% = r2h                     ;
            if(r4==0) jmp $+2                ; if ch = 1
            r2 = *dp0++                      ;
            *dp1%% = r2h                     ;
            nop                              ;
     }                                       ;
     r2l = dp1                               ;
     *ring_write_ptr:x = r2l                 ;

     r3 += r3                                ; always, output 2 ch
     clr(r2)                                 ;
     *f_dec_unit_end:x =  r2h                ; f_dec_unit_end = 0

     r2l  = EIR                              ;
     r2   = r2 | 0x8000                      ;
     EIR  = r2l                              ;
     nop                                     ;
     nop                                     ; wait disable interrupt
     nop                                     ;
     r2l = *ring_entries:x                   ;
     r2  = r2 + r3                           ;
     *ring_entries:x = r2l                   ;
     r2l = EIR                               ;
     r2  = r2 & 0x7fff                       ;
     EIR = r2l                               ; enable interrupt
     ret                                     ;


/* ================================================================
[Handler Name]  _so_interrupt:
 ================================================================*/
_so_interrupt:
     *save_regs+0:x = r0l                    ; push r0
     *save_regs+1:x = r0h                    ;
     *save_regs+2:x = r0e                    ;
     r0l = dp0                               ;
     *save_regs+3:x = r0l                    ; push dp0
     r0l = dn0                               ;
     *save_regs+4:x = r0l                    ; push dn0
     r0l = dmx                               ;
     *save_regs+5:x = r0l                    ; push dmx
```

(13/13)

```
     r0l = *ring_read_ptr:                ;
     dp0 = r0l                            ; set dp0
     r0l = *ring_dn0:x                    ;
     dn0 = r0l                            ; set dn0
     dmx = MAX_RINGSIZE-1                 ; set dmx
     r0  = *dp0%%                         ;
     *SDT1:x=r0h                          ; set output PCM data
     r0l = dp0                            ;
     *ring_read_ptr:x = r0l               ; set read ptr
     r0l = *ring_entries:x                ;
     r0  = r0 - 1                         ;
     *ring_entries:x = r0l                ; set ring_entries

     r0l = *save_regs+5:x                 ;
     dmx = r0l                            ; pop dmx
     r0l = *save_regs+4:x                 ;
     dn0 = r0l                            ; pop dn0
     r0l = *save_regs+3:x                 ;
     dp0 = r0l                            ; pop dp0
     r0e = *save_regs+2:x                 ;
     r0h = *save_regs+1:x                 ;
     r0l = *save_regs+0:x                 ; pop r0
     reti                                 ;

end
```

### A.1.2 smp_data.asm

This file is used to secure the static and scratch areas.

```
/* ============================================================
 *      PUBLIC
 * ============================================================ */
public wmad_lib_static_x1
public wmad_lib_static_x2
public wmad_lib_static_y1
public wmad_lib_static_y2

public wmad_lib_scratch_x
public wmad_lib_scratch_y


/* ============================================================
 *      STATIC MEMORY
 * ============================================================ */

__WMAD_LIB_STATIC_X1        XRAMSEG
wmad_lib_static_x1:                 ds      972

__WMAD_LIB_STATIC_X2        XRAMSEG
wmad_lib_static_x2:                 ds      4096


__WMAD_LIB_STATIC_Y1        YRAMSEG
wmad_lib_static_y1:                 ds      8192

__WMAD_LIB_STATIC_Y2        YRAMSEG
wmad_lib_static_y2:                 ds      4096


/* ============================================================
 *      SCRATCH MEMORY
 * ============================================================ */

__WMAD_LIB_SCRATCH_X        XRAMSEG
wmad_lib_scratch_x:                 ds      1600

__WMAD_LIB_SCRATCH_Y        YRAMSEG
wmad_lib_scratch_y:                 ds      300

end
```

### A.1.3 smp_user.asm

This file is the source file of the user-defined function wmad_FileCBGetData.

(1/6)

```
/*-------------------------------------------------------------------*/
/*   File Information                                                */
/*-------------------------------------------------------------------*/
/*   Name      : smp_user.asm                                        */
/*   Type      : Assembler program module                            */
/*   Version   : 1.00                                                */
/*   Date      : 2001 Jun 18                                         */
/*   CPU       : uPD7701x Family                                     */
/*   Assembler : WB77016 Ver 2.4                                     */
/*   About     : wmad_FileCBGetData function                         */
/*                                                                   */
/*-------------------------------------------------------------------*/
/*   Copyright (C) NEC Corporation 2000, 2001                        */
/*   All rights reserved by NEC Corporation                          */
/*   Use of copyright notice does not evidence publication           */
/*-------------------------------------------------------------------*/

/* ================================================================
 *   INCLUDE FILES
 * ================================================================ */
#include "wma_dec.h"

/* ================================================================
 *   PUBLIC FUNCTIONS
 * ================================================================ */
public Init_FileCBGetData
public wmad_FileCBGetData

/* ================================================================
 *   EXTERN FUNCTIONS
 * ================================================================ */


/* ================================================================
 *   DEFINE
 * ================================================================ */
#define CW_BUFF_SIZE     64
#define CB_BUFF_SIZE    128


/* ================================================================
 *      LOCAL MEMORY
 * ================================================================ */
```

(2/6)

```
__SMP_USER_XRAM__           XRAMSEG
FileCBGetData_fp:
     ds      2              ; size of obtained bitstream data [byte]
read_ptr:
     ds      1              ; read pointer to host_in_buffer
write_ptr:
     ds      1              ; write pointer to host_in_buffer
tmp_dn0:
     ds      1              ; for saving value of dn0 register
tmp_dmx:
     ds      1              ; for saving value of dmx register

__SMP_USER_BUFFER__ XRAMSEG align at 0
host_in_buffer:
     ds      CW_BUFF_SIZE*2  ; buffer for input data from Host I/F
input_buffer:
     ds      CW_BUFF_SIZE    ; buffer for input data to Middle Ware


/* ================================================================
 *      PROGRAM CODE
 * ============================================================== */
__SMP_USER__   IMSEG

/* ================================================================
[Function Name]      Init_FileCBGetData
[Argument]           r0  : data size
[Return]             non
[Call Function]      non
[Use Register]       r0, dp0, dp1
[Use Stacks]         loop stack: 1, call stack: 0, repeat: 0
 ================================================================*/
Init_FileCBGetData:
     clr(r0)                        ;
     *FileCBGetData_fp+0:x = r0l    ;
     *FileCBGetData_fp+1:x = r0l    ;
     dn0 = r0l                      ;
     r0l = host_in_buffer           ;
     *read_ptr:x = r0l              ;
     *write_ptr:x = r0l             ;
     dp0 = r0l                      ;
     r0l = input_buffer             ;
     dp1 = r0l                      ;
     Loop CW_BUFF_SIZE {            ; clear buffer
             *dp0++ = r0h           ;
             *dp0++ = r0h           ;
             *dp1++ = r0h           ;
     }                              ;
     ret                            ;
```

(3/6)

```
/* ============================================================
[Function Name]      wmad_FileCBGetData
[Argument]           r0  : required size [byte]
                     r1  : offset [byte]
[Return]             r0  : obtained size [byte]
                     r2l : address of input buffer
[Call Function]      non
[Use Register]       r0, r1, r2, r3, r4, r5, r6
                     dp0, dp1, dn0, dmx
[Use Stacks]         loop stack: 1, call stack: 0, repeat: 0
 ============================================================*/
wmad_FileCBGetData:
     r4l = dn0                              ;
     *tmp_dn0:x = r4l                       ;
     r4l = dmx                              ;
     *tmp_dmx:x = r4l                       ;

     r4l = 1                                ;
     dn0 = r4l                              ;
     r4l = 2*CW_BUFF_SIZE - 1               ;
     dmx = r4l                              ;

     r5 = *FileCBGetData_fp+0:x             ;
     r5l= *FileCBGetData_fp+1:x             ; r5 is always even.
     r4 = r5-r1                             ;

     if(r4==0) jmp case000                  ;
     if(r4>0)  jmp case002                  ;

     ;;=========================================================;;
     ;;      case001 ( r5 < r1 )
     ;;=========================================================;;
case001:
     r4 = -r4                               ;
     r4 = r4 sra 1                          ; r4 = read size [word]
     if(r4<=0) jmp case001a                 ;
     Loop r4l {                             ;
             %READ_HOST(R6,R6)              ;
             r5 = r5 + 2                    ;
             nop                            ; not stored
        }                                   ;
case001a:
     r6l = *write_ptr:x                     ;
     *read_ptr:x = r6l                      ; set read_ptr

     r3 = r0 + 1                            ;
     r3 = r3 srl 1                          ; set r3 = read size [word]
     jmp get_data                           ;
```

(4/6)

```
     ;;=========================================================;;
     ;;      case000 ( r5 = r1 )
     ;;=========================================================;;
case000:
     r6l = *write_ptr:x                    ;
     *read_ptr:x = r6l                     ; set read_ptr

     r3 = r0 + 1                           ;
     r3 = r3 srl 1                         ; set r3 = read size [word]
     jmp get_data                          ;


     ;;=========================================================;;
     ;;      case002 ( r5 > r1 )
     ;;=========================================================;;
case002:
     r3 = r4 + 1                           ;
     r3 = r3 sra 1                         ; set r3 = rewind size [word]

     clr(r6)                               ;
     r6l= *write_ptr:x                     ;
     r6 = r6 - r3                          ;

     r3 = r6 - host_in_buffer              ;
     if(r3>=0) jmp case002a                ;
     r6 = r3 + host_in_buffer + CB_BUFF_SIZE;

case002a:
     *read_ptr:x = r6l                     ; set read_ptr
     r3  = r0 - r4                         ;
     if(r3<=0) jmp set_data                ; No need to read data
     r3  = r3+1                            ;
     r3  = r3 srl 1                        ; set r3 = read size [word]


     ;;=========================================================;;
     ;;      Get Data
     ;;=========================================================;;
get_data:
     r4 = r0 - CB_BUFF_SIZE                ;
     if(r4<=0) jmp get_data_nex            ;
     r3 = r0 + 1                           ;
     r3 = r3 sra 1                         ;
     Loop r3l {                            ;
             %READ_HOST(R4,R4)             ;
             r5 = r5 + 2                   ;
             nop                           ; not stored
     }                                     ;
     jmp finish                            ;
```

(5/6)

```
get_data_next:
     if(r3==0) jmp set_data          ;
     r6l = *write_ptr:x              ;
     dp0 = r6l                       ;
     loop r3l {
             %READ_HOST(R4,R4)       ; get data
             *dp0%% = r4h            ;
             r5 = r5 + 2             ;
     }
     r6l = dp0                       ;
     *write_ptr:x = r6l              ;


     ;;=========================================================;;
     ;;      Set Data
     ;;=========================================================;;
set_data:
     r6l = *read_ptr:x               ;
     dp0 = r6l                       ;
     dp1 = input_buffer              ;

     r4 = r0+1                       ;
     r4 = r4 srl 1                   ;
     if(r4==0) jmp finish            ;

     r3 = r1 & 1                     ;
     if(r3==0) jmp simple_copy       ;
     r3l= *dp0%%                     ;
     r3 = r3 sll 8                   ;
     loop r4l {
             r3 = r3 sll 8           ; change allocation
             r3l= *dp0%%             ;
             r3 = r3 sll 8           ;
             *dp1++ = r3h            ; set data
     }                               ;
     jmp finish                      ;
simple_copy:
     loop r4l {                      ;
             r3l= *dp0%%             ;
             *dp1++ = r3l            ; set data
     }                               ;


     ;;=========================================================;;
     ;;      Finish
     ;;=========================================================;;
finish:
     *FileCBGetData_fp+0:x = r5h     ;
     *FileCBGetData_fp+1:x = r5l     ;
```

(6/6)

```
      r4l = *tmp_dn0:x                ;
      dn0 = r4l                       ;
      r4l = *tmp_dmx:x                ;
      dmx = r4l                       ;

      r2l= input_buffer               ; set r2l
      ret                             ;

END
```

## A.2 Sample Header File

### A.2.1 wma_dec.h

This is the header file for the sample program.

(1/2)

```
/*-------------------------------------------------------------------*/
/*   File Information                                                */
/*-------------------------------------------------------------------*/
/*   Name       : wma_dec.h                                          */
/*   Type       : Assembler header file                              */
/*   Version    : 1.00                                               */
/*   Date       : 2001 Jun 18                                        */
/*   CPU        : uPD7701x Family                                    */
/*   Assembler  : WB77016 Ver 2.4                                    */
/*   About      : header file for sample source code                 */
/*                                                                   */
/*-------------------------------------------------------------------*/
/*   Copyright (C) NEC Corporation 2000, 2001                        */
/*   All rights reserved by NEC Corporation.                         */
/*   Use of copyright notice does not evidence publication           */
/*-------------------------------------------------------------------*/

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

/* ================================================================
 *   EXTERN FUNCTIONS
 * ================================================================ */
extrn wmad_FileDecodeInit
extrn wmad_FileDecodeData
extrn wmad_FileGetPCM
extrn wmad_FileDecodeInfo
extrn wmad_GetVersion

/* ================================================================
 *   MACRO
 * ================================================================ */
%define (READ_HOST(X,Y)) (
     Y@L = *HST:x                          ;
     Y   = Y & 1                           ;
```

(2/2)

```
     if(Y !=0) jmp $-2                    ;
     X   = *HDT:x                         ;
);

/* ============================================================
 *   DEFINE (ERROR CODE)
 * ============================================================ */
#define cWMA_NoErr                        0
#define cWMA_Failed                       1
#define cWMA_BadArgument                  2
#define cWMA_BadAsfHeader                 3
#define cWMA_BadPacketHeader              4
#define cWMA_BrokenFrame                  5
#define cWMA_NoMoreFrames                 6
#define cWMA_BadSamplingRate              7
#define cWMA_BadNumberOfChannels          8
#define cWMA_BadVersionNumber             9
#define cWMA_BadWeightingMode             10
#define cWMA_BadPacketization             11
#define cWMA_BadDRMType                   12
#define cWMA_DRMFailed                    13
#define cWMA_DRMUnsupported               14
#define cWMA_DemoExpired                  15
#define cWMA_BadState                     16
#define cWMA_Internal                     17
```

## A.3 Sample Timing Files

### A.3.1 smp_input.tmg

This timing file fetches data from the input data file (sample_20_16s.dat) and inputs data via the host interface. The input data file name on the 10th line should be changed to accord with the input data file prepared by the user.

(1/2)

```
;;-------------------------------------------------------------------;;
;;   Declare Variables
;;-------------------------------------------------------------------;;
local data                          ; local variable receives data


;;-------------------------------------------------------------------;;
;;   File Open
;;-------------------------------------------------------------------;;
open input "sample_20_16s.dat"
output format showbase unsigned hex, ; select output format


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

;;-------------------------------------------------------------------;;
;;   Main Input Loop
;;-------------------------------------------------------------------;;
  do                                ;
     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
```

(2/2)

```
    set pin hcs = 0                ; perform 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
  enddo                            ;


;;-------------------------------------------------------------------;;
;;   File Close
;;-------------------------------------------------------------------;;
close input                        ;

break                              ;
end
```

### A.3.2 smp_serout.tmg

This timing file is used to save the 16-bit linear PCM data output from the serial interface to a file (so_sample_20_16s_l.dat, so_sample_20_16s_r.dat). Set any name for the file names on the 9th and 10th lines. In the case of stereo data, L-channel and R-channel data is output to output files #1 and #2 respectively. In the case of monaural data, the same data is output to output files #1 and #2.

(1/3)

```
;;-------------------------------------------------------------------------;;
;;   Initialize
;;-------------------------------------------------------------------------;;
set pin soen1 = 0                   ; initialize SOEN1 line

;;-------------------------------------------------------------------------;;
;;   File Open
;;-------------------------------------------------------------------------;;
open output #1 "so_sample_20_16s_l.dat"     ;
open output #2 "so_sample_20_16s_r.dat"     ;
open output #3 "dummy_serial.dat"           ;
output format showbase unsigned hex,        ; select output format

;;-------------------------------------------------------------------------;;
;;   Dummy Output
;;-------------------------------------------------------------------------;;
  if pin soen1 == 0                 ; do only if new transmission start
     wait cond pin sorq1 == 1       ; wait for serial output request
     wait 5 ns                      ; logic delay
     set pin soen1 = 1              ; start serial output
  endif

  wait cond pin sorq1 == 0          ; wait for serial output start confirmation
  set pin soen1 = 0                 ;

  rept 15                           ; wait 15 clock cycles for one data frame
     wait cond pin sck1 == 0        ;   wait for rising edge (0->1)
     wait cond pin sck1 == 1        ;
  endrept                           ;

  wait 5ns                          ; make sure SO1 and SORQ1 are updated

     output #3 port so1&0xFFFF      ; write output data to file, mask sign bits

     if pin sorq1 == 1              ; request next output
             set pin soen1 = 1      ; start next serial output
     endif
```

(2/3)

```
  wait cond pin sck1 == 0           ; wait for rising edge (0->1)
  wait cond pin sck1 == 1           ;
close output #3


;;-------------------------------------------------------------------;;
;;   Main Output Loop
;;-------------------------------------------------------------------;;
do                                  ;

;;-------------------------------------------------------------------;;
;;   For ch 1
;;-------------------------------------------------------------------;;
  if pin soen1 == 0                 ; do only if new transmission start
     wait cond pin sorq1 == 1       ; wait for serial output request
     wait 5 ns                      ; logic delay
     set pin soen1 = 1              ; start serial output
  endif

  wait cond pin sorq1 == 0          ; wait for serial output start confirmation
  set pin soen1 = 0                 ;

  rept 15                           ; wait 15 clock cycles for one data frame
     wait cond pin sck1 == 0        ;       wait for rising edge (0->1)
     wait cond pin sck1 == 1        ;
  endrept                           ;

  wait 5ns                          ; make sure SO1 and SORQ1 are updated

  output #1 port so1&0xFFFF         ; write output data to file, mask sign bits

  if pin sorq1 == 1                 ; request next output
     set pin soen1 = 1              ; start next serial output
  endif

  wait cond pin sck1 == 0           ; wait for rising edge (0->1)
  wait cond pin sck1 == 1           ;

;;-------------------------------------------------------------------;;
;;   For ch 2
;;-------------------------------------------------------------------;;
  if pin soen1 == 0                 ; do only if new transmission start
     wait cond pin sorq1 == 1       ; wait for serial output request
     wait 5 ns                      ; logic delay
     set pin soen1 = 1              ; start serial output
  endif

  wait cond pin sorq1 == 0          ; wait for serial output start confirmation
  set pin soen1 = 0                 ;
```

(3/3)

```
  rept 15                          ; wait 15 clock cycles for one data frame
     wait cond pin sck1 == 0       ;       wait for rising edge (0->1)
     wait cond pin sck1 == 1       ;
  endrept                          ;

  wait 5ns                         ; make sure SO1 and SORQ1 are updated

  output #2 port so1&0xFFFF        ; write output data to file, mask sign bits

  if pin sorq1 == 1                ; request next output
     set pin soen1 = 1             ; start next serial output
  endif

  wait cond pin sck1 == 0          ; wait for rising edge (0->1)
  wait cond pin sck1 == 1          ;


;;-------------------------------------------------------------------------;;
;;   Serial Output is finished ?
;;-------------------------------------------------------------------------;;
  exit  ip == (MAIN.serout_finish & 0xffff)        ;
  exit  ip == ((MAIN.serout_finish+1) & 0xffff)    ;
enddo


;;-------------------------------------------------------------------------;;
;;   File Close
;;-------------------------------------------------------------------------;;
close output #1                    ; close data file
close output #2                    ; close data file


break                              ;
end
```

### A.3.3 clk_for_2ch.tmg

This timing file creates the clock signals used to generate the serial output interrupt. Create clock signals with an appropriate cycle in accordance with the input data sampling frequency. This file is described under the assumption that 2-channel data is always output.

```
;;-------------------------------------------------------------------------;;
;;   Declare Variables
;;-------------------------------------------------------------------------;;
LOCAL TM_pico_sec    ;
LOCAL fs             ;
LOCAL retern_addr    ;

;;-------------------------------------------------------------------------;;
;;   Wait
;;-------------------------------------------------------------------------;;
wait cond reg ip == ((MAIN.serout_start) & 0xffff)  ;

;;-------------------------------------------------------------------------;;
;;   Set Sampling Frequency and TM_pico_sec
;;-------------------------------------------------------------------------;;
set fs = *sample_rate:x & 0xffff                    ;
set TM_pico_sec = TIME_RESOLUTION / 16 / fs / 2     ; TIME_RESOLUTION = 10**12

;;-------------------------------------------------------------------------;;
;;   Generate Clock for 2 ch
;;-------------------------------------------------------------------------;;
do
  wait (TM_pico_sec/2) ps                   ;
  set pin sck1 = 0                          ;
  set pin sck2 = 0                          ;
  wait (TM_pico_sec/2) ps                   ;
  set pin sck1 = 1                          ;
  set pin sck2 = 1                          ;
enddo
```

NEC

# Facsimile Message

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.

From:

Name

Company

Tel. FAX

Address

*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

**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 | ❑ | ❑ | ❑ | ❑ |