User's Manual

WMA Decode Middleware

ユーザーズマニュアル

ルネサス エレクトロニクス
www.renesas.com

Rev.1.01　2014.08

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、
   各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、
   家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：　輸送機器（自動車、電車、船舶等）、交通用信号機器、
   防災・防犯装置、各種安全装置等

   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 本資料に記載されている当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注１において定義された当社の開発、製造製品をいいます。

(2012. 4)

このマニュアルの使い方

1. 目的と対象者

このマニュアルは、本ミドルウェアのデコーダ機能と性能、使用方法をユーザに理解していただくためのマニュアルです。本ミドルウェアを用いた応用システムを設計するユーザを対象にしています。このマニュアルを使用するには、オーディオ、プログラミング言語、マイクロコンピュータに関する基本的な知識が必要です。

このマニュアルは、大きく分類すると、製品の概要、ミドルウェア仕様、ライブラリ関数仕様、使用上の注意で構成されています。

本ミドルウェアは、注意事項を十分確認の上、使用してください。注意事項は、各章の本文中、各章の最後、注意事項の章に記載しています。

改訂記録は旧版の記載内容に対して訂正または追加した主な箇所をまとめたものです。改訂内容すべてを記録したものではありません。詳細は、このマニュアルの本文でご確認ください。

2. 本製品のご使用について

本製品のご使用にあたっては、弊社とソフトウェア使用許諾 (ライセンス) 契約を取り交わして頂く必要があります。

3. 関連マニュアル

WMA 関連資料

規格番号・タイトル	発行日
ASF 規格	
Advanced Systems Format (ASF) Specification Revision 01.20.02	2004/06/15
WMA Standard 規格	
Decoding WMA (Standard) An Overview of Windows Media Audio (Standard) Decoding	2002/05/23

プロセッサ関連資料

別紙の製品マニュアルを参照してください。

4. 略語および略称の説明

略語／略称	英語名	日本語名
ABR	Average Bit Rate	平均ビットレート
ANSI-C	American National Standards Institute - C	米国標準規格協会が定めた C 言語標準規格
ASF	Advanced Systems Format	Microsoft 社開発のコンテナ・ファイル形式
bps	bits per second	転送速度を表す単位、ビット/秒
CBR	Constant Bit Rate	固定ビットレート
DAC	digital to analog converter	デジタル-アナログ変換回路
DRC	Dynamic Range Control	動的範囲制御
LSB	Least Significant Bit	最下位ビット
MBR	Multiple Bit Rate	マルチ・ビットレート
MSB	Most Significant Bit	最上位ビット
PCM	Pulse Code Modulation	パルス符号変調
RAM	Random Access Memory	書き込みメモリ
ROM	Read Only Memory	読み出し専用メモリ
WMA	Windows Media Audio	Microsoft 社開発の音声圧縮方式
WM DRM	Windows Media Digital Right Management	Microsoft 社の Windows Media 適用のデジタル著作権管理技術

目次

1. 概要

本章では、WMA デコーダの概要について説明します。

1.1 仕様概要

WMA は、Microsoft 社が開発したオーディオ信号符号化および復号化の方式です。本ミドルウェアは、WMA Standard 規格に対応し、入力された圧縮データをデコードして出力します。対応範囲については、表 1.1を参照してください。基本仕様、処理性能については、別紙の製品マニュアルを参照してください。

表1.1 対応規格

項目	内容
入力データ形式	WMA Standard Version 2, 7, 8, 9, 9.1, 9.2 (or later) All profile (L1/L2/L3)
出力データ形式	16 ビット・リニア PCM
対応サンプリング周波数(Hz)	Profile L1 : 44,100 Profile L2/L3 : 8,000 / 11,025 / 16,000 / 22,050 / 32,000 / 44,100 / 48,000
対応チャネル数	1 チャネル 2 チャネル
対応ビットレート(kbps)	Profile L1 : 64 ～ 161 Profile L2 : ～ 161 Profile L3 : ～ 385 【注】本ミドルウェアは、最低ビットレートとして Microsoft 社テストデータの範囲(Profile L2 にて 128bps、Profile L3 にて 95224bps まで)の動作確認を行っています。 【注】本ミドルウェアは、固定ビットレート(CBR)、可変ビットレート(VBR)、マルチ・ビットレート(MBR)、平均ビットレート(ABR)のデータをデコード可能ですが、MBR データの場合は、デコードするストリームを 1 つ選択し、Demux する必要があります。
リエントラント	対応
制限事項	下記機能には非対応 ・ASF Demux 機能 ・WM DRM Decrypt 機能 ・DRC 機能

表1.2 必要メモリ・サイズ

メモリ種別	配置	メモリ領域名		サイズ[byte]	
命令	ROM	命令領域			
データ		定数テーブル領域			—
		その他の領域(コンパイラ依存)			
	RAM	ミドルウェア・ワーク領域			65,536
		<内訳>	スタティック領域	<内訳>	49,152
			スクラッチ領域		16,384
		ユーザ・ワーク領域			53,424
		<内訳>	入力バッファ	<内訳>	20,480
			出力バッファ		32,768
			各種構造体		176
		スタック領域			2,048
		その他の領域(コンパイラ依存)			—

【注】配置欄に ROM と示してある領域は、RAM または ROM に配置することができます。

【注】配置欄に RAM と示してある領域は、RAM にのみ配置することができます。

【注】命令領域、定数テーブル領域、その他の領域のサイズは、別紙の製品マニュアルを参照してください。

1.2 構成

本ミドルウェアを使用したデコード・システムの構成例を図 1.1に示します。

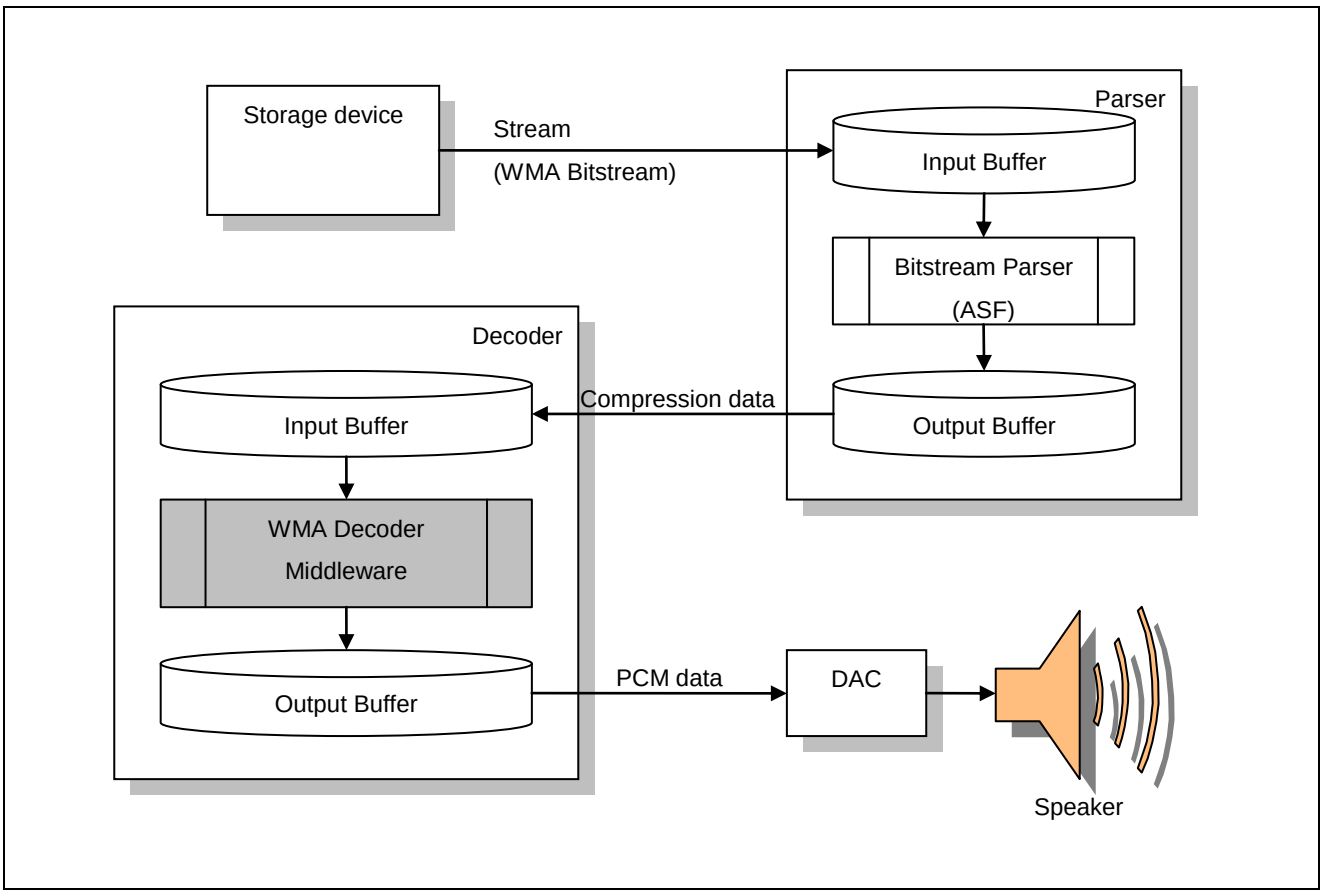

図1.1 デコード・システム構成例

1. WMA Bitstream
   WMA Bitstreamは、サンプリングされたリニアPCMデータをWMA規格で圧縮したデータです。
   対応規格については表1.1を参照してください。

2. Parser
   WMA Bitstreamを包括している不必要な部分などを取り除き、Payload Dataもしくは、Sub-Payload #N Data(n = 0, 1, …)単位(以下、データ・ブロックと表記)で切り出します。**ユーザがターゲット・システムにあわせて設計する必要があります。**

3. Compression data
   データ・ブロックに切り出されたデータです。

4. Decoder
   Input Bufferに格納されたデータを、本ミドルウェアが処理し、Output Bufferへと出力します。

5. PCM data
   本ミドルウェアによりデコードされた16ビット・リニアPCMデータです。

6. DAC
   16ビット・リニアPCMデータをアナログ信号に変換します。

2. ミドルウェア仕様

2.1 ライブラリ関数一覧

本ミドルウェアが提供する関数を表 2.1に示します。
関数の詳細な仕様については、3.1 節を参照してください。

表2.1 関数一覧

関数名	概要
wmastd_GetMemorySize	必要メモリ・サイズ計算処理
wmastd_Init	WMAデコーダ初期化
wmastd_AudecGetData	データ入力処理
wmastd_AudioDecode	デコード処理
wmastd_ReconstructPcmData	PCMデータ生成処理
wmastd_GetErrorFactor	エラー要因情報取得
wmastd_GetVersion	バージョン情報取得

2.2 構造体一覧

本ミドルウェアで、ユーザが領域確保を行う必要のある構造体を表 2.2に示します。
構造体の詳細な仕様については、3.2 節を参照してください。

表2.2 構造体一覧

構造体名	概要	I/O
メモリ・サイズ取得設定情報構造体	メモリ・サイズ取得に必要なパラメータを格納する構造体です。	I
メモリ・サイズ取得結果情報構造体	取得したメモリ・サイズを格納する構造体です。	O
ワーク・メモリ情報構造体	ワーク・メモリに関するパラメータを格納する構造体です。	I
初期化設定情報構造体	初期化に必要なパラメータを格納する構造体です。	I
データ入力設定情報構造体	データ入力処理に必要なパラメータを格納する構造体です。	I
データ入力結果情報構造体	データ入力処理の結果を格納する構造体です。	O
デコード設定情報構造体	デコードに必要なパラメータを格納する構造体です。	I
デコード結果情報構造体	デコード結果を格納する構造体です。	O
PCM生成処理設定情報構造体	PCM データ生成処理に必要なパラメータを格納する構造体です。	I
PCM生成処理結果情報構造体	PCM データ生成処理の結果を格納する構造体です。	O
バッファ・メモリ設定情報構造体	入出力バッファに関するパラメータを格納する構造体です。	I
バッファ・メモリ結果情報構造体	入出力バッファに関する処理結果を格納する構造体です。	O

2.3 マクロ定義

2.3.1 型定義一覧

本ミドルウェアで使用する型定義の一覧を表 2.3に示します。

表2.3 型定義一覧

型	サイズ[byte]	説明	
ACMW_INT8	1	符号あり8bit整数	-128 ～ 127
ACMW_INT16	2	符号あり16bit整数	-32768 ～ 32767
ACMW_INT32	4	符号あり32bit整数	-2147483648 ～ 2147483647
ACMW_UINT8	1	符号なし8bit整数	0 ～ 255
ACMW_UINT16	2	符号なし16bit整数	0 ～ 65535
ACMW_UINT32	4	符号なし32bit整数	0 ～ 4294967295
ACMW_BOOL	2	ブール値(符号あり16bit整数)	(0[FALSE] / 0以外[TRUE])

【注】ポインタは、全て同じサイズ(4byte)です。

2.3.2 共通シンボル一覧

本ミドルウェアで使用するシンボル定義の一覧を表 2.4に示します。

表2.4 共通シンボル一覧

共通シンボル	定義	説明
WMASTD_RESULT_OK	0x00000000	処理結果が正常です。
WMASTD_RESULT_NG	0x00000001	処理結果が異常です。
WMASTD_RESULT_WARNING	0x00000002	処理継続可能な異常が発生しました。
WMASTD_RESULT_FATAL	0x00000003	処理継続不可能な異常が発生しました。

2.4 予約語

本ミドルウェアで使用するシンボルの命名規約を表 2.5に示します。
他のアプリケーションを組み合わせて使用するときは、重複しないようにしてください。

表2.5 シンボル命名規約

分類	概要
関数名	wmastd_XXXX
構造体名	wmastd_XXXX
関数の返却値	WMASTD_RESULT_XXXX 【注】XXXXは全て大文字
エラー要因名	WMASTD_ERR_XXXX 【注】XXXXは全て大文字
基本型プレフィックス名	ACMW_XXXX 【注】XXXXは全て大文字
その他プレフィックス名	WMASTD_XXXX 【注】XXXXは全て大文字

【注】XXXX は任意の英数字とする

2.5 処理フロー

本ミドルウェアを使用したアプリケーションの処理の流れを図 2.1に示します。また、wmastd_Init, wmastd_AudecGetData, wmastd_AudioDecode, wmastd_ReconstructPcmData 関数において、エラーが発生した場合の流れを図 2.2に示します。

網掛けした箇所は本ミドルウェアの関数が実行する処理です。網掛けしていない箇所はユーザが定義する処理です。ターゲット・システムにあわせて設計してください。

サンプルプログラムとサンプルパーサーのフローチャートを図 2.3、図 2.4に示します。

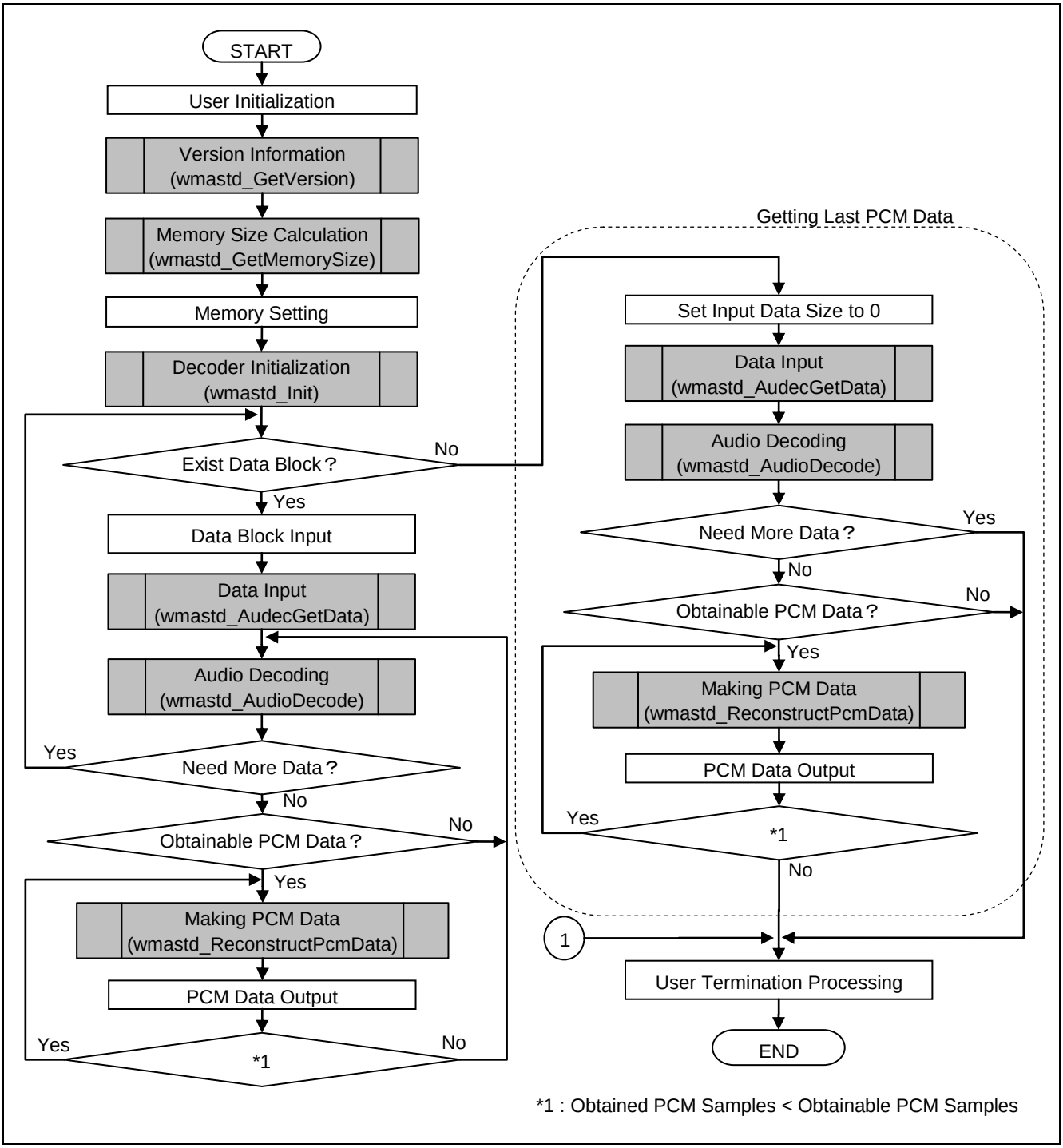

図2.1 アプリケーション処理フロー例

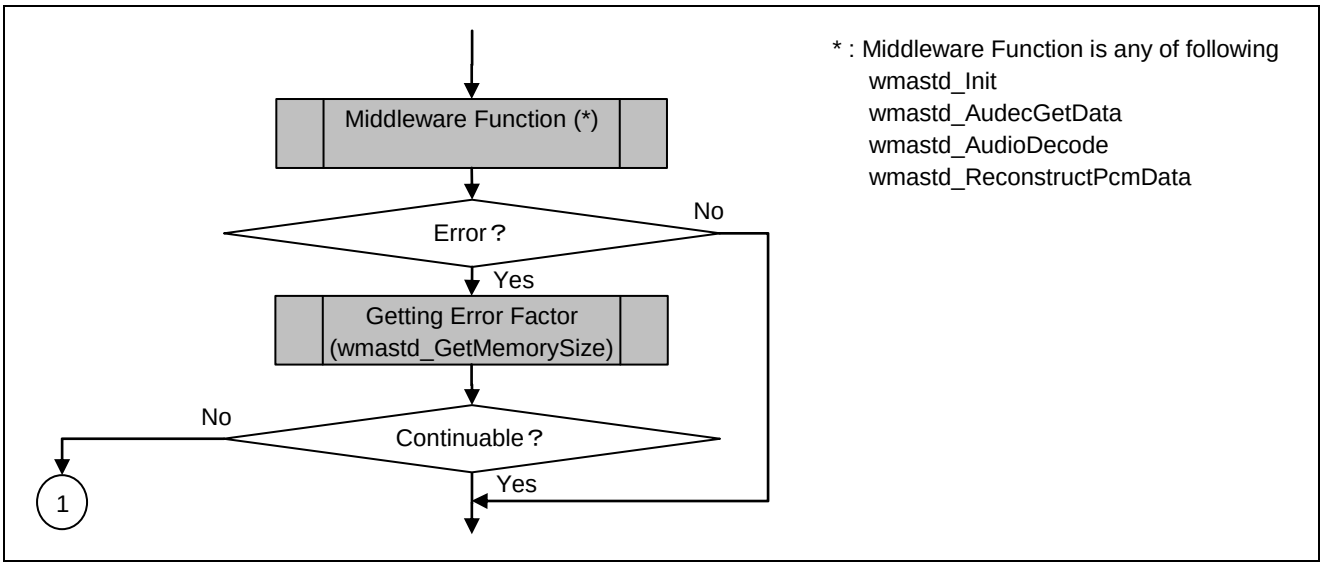

図2.2 エラー処理フロー例

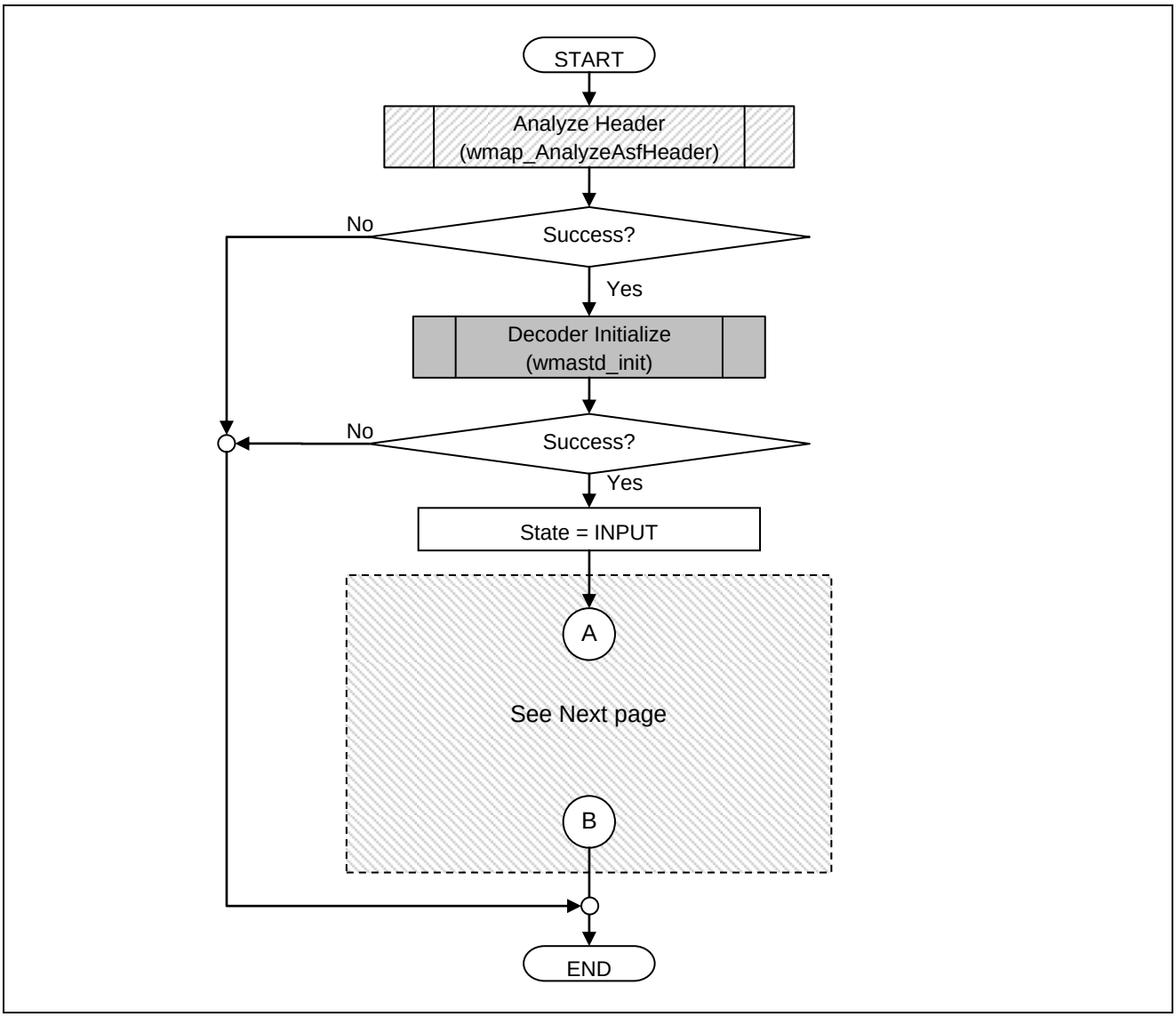

図2.3 サンプルプログラムとサンプルパーサーのフローチャート その 1

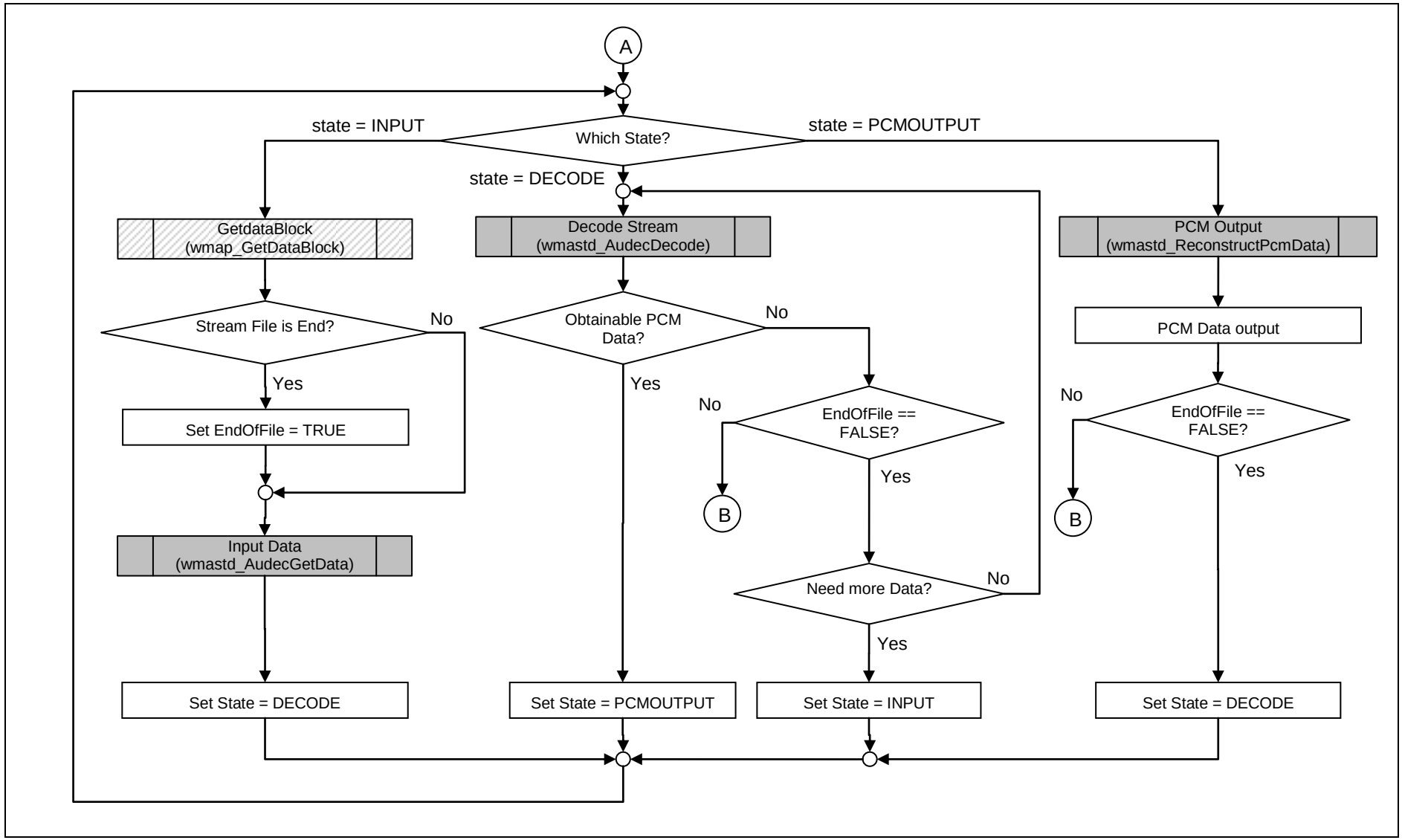

図2.4 サンプルプログラムとサンプルパーサーのフローチャート その２

3. ライブラリ関数仕様

3.1 関数仕様

次項から、本ミドルウェアが提供する関数について、以下の記述フォーマットに従って説明します。

概要	関数の概要を説明します。		
構文	関数の呼出し形式を説明します。		
機能	関数の機能を説明します。		
引数		I/O	関数の引数を説明します。
戻り値	型名		関数の返却値を説明します。
説明	関数を使用する際の注意点などについて説明します。		

【注】書式はANSI-Cに準拠します。数学関数を除いて、C言語規格上の標準Cライブラリ関数は使用しません。

3.1.1 wmastd_GetMemorySize 関数

概要	必要メモリ・サイズ計算処理		
構文	ACMW_INT32 wmastd_GetMemorySize(const wmastd_getMemorySizeConfigInfo * const pGetMemorySizeConfigInfo, wmastd_getMemorySizeStatusInfo * const pGetMemorySizeStatusInfo);		
機能	本ミドルウェアが使用するスタティック領域、スクラッチ領域、および入出力バッファに必要なメモリ・サイズを計算し、メモリ・サイズ取得結果情報構造体に格納します。		
引数		I/O	意味
wmastd_getMemorySizeConfigInfo *pGetMemorySizeConfigInfo		I	メモリ・サイズ取得設定情報構造体
wmastd_getMemorySizeStatusInfo *pGetMemorySizeStatusInfo		O	メモリ・サイズ取得結果情報構造体
戻り値	ACMW_INT32 errorCode		エラー・コード 詳細は、表3.4を参照
説明	wmastd_Init関数を実行する前に、本関数を実行し、必要なサイズのメモリを確保してください。本関数は、初期化処理を必要としないので、任意のタイミングで実行可能です。		

3.1.2 wmastd_Init 関数

概要	WMAデコーダ初期化処理		
構文	ACMW_INT32 wmastd_Init(const wmastd_workMemoryInfo * const pWorkMemInfo, const wmastd_initConfigInfo * const pInitConfigInfo, wmastd_initStatusInfo * const pInitStatusInfo);		
機能	本ミドルウェアが使用するスタティック領域、スクラッチ領域を初期化し、各種パラメータを設定します。		
引数		I/O	意味
wmastd_workMemoryInfo *pWorkMemInfo		I	ワーク・メモリ情報構造体
wmastd_initConfigInfo *pInitConfigInfo		I	初期化設定情報構造体
wmastd_initStatusInfo *pInitStatusInfo		O	初期化結果情報構造体
戻り値	ACMW_INT32 errorCode		エラー・コード 詳細は、表3.4を参照
説明	一連のデコード処理を開始する前に、この関数を1度だけ実行してください。ここで一連のデコード処理”とは、あるストリームのデコード開始からデコード終了をさします。		

3.1.3 wmastd_AudecGetData 関数

概要	データ入力処理		
構文	ACMW_INT32 wmastd_AudecGetData(const wmastd_workMemoryInfo * const pWorkMemInfo, const wmastd_audecGetDataConfigInfo * const pAudecGetDataConfigInfo, wmastd_audecGetDataStatusInfo * const pAudecGetDataStatusInfo);		
機能	データ・ブロックを入力します。		
引数		I/O	意味
wmastd_workMemoryInfo *pWorkMemInfo		I	ワーク・メモリ情報構造体
wmastd_audecGetDataConfigInfo *pAudecGetDataConfigInfo		I	データ入力設定情報構造体
wmastd_audecGetDataStatusInfo *pAudecGetDataStatusInfo		O	データ入力結果情報構造体
戻り値	ACMW_INT32 errorCode		エラー・コード 詳細は、表3.4を参照
説明	データ・ブロックをミドルウェアへ入力するとき、この関数を実行してください。 【注】データ・ブロックサイズが4byte以上の場合は、必ず入力バッファに4byte以上のデータを入力してください。4byte未満の場合は、データ・ブロック全体を一度に入力してください。		

3.1.4 wmastd_AudioDecode 関数

概要	デコード処理		
構文	ACMW_INT32 wmastd_AudioDecode(const wmastd_workMemoryInfo * const pWorkMemInfo, const wmastd_audioDecodeConfigInfo * const pAudioDecodeConfigInfo, wmastd_audioDecodeStatusInfo * const pAudioDecodeStatusInfo);		
機能	データ・ブロックをデコードします。		
引数		I/O	意味
wmastd_workMemoryInfo *pWorkMemInfo		I	ワーク・メモリ情報構造体
wmastd_audioDecodeConfigInfo *pAudioDecodeConfigInfo		I	デコード設定情報構造体
wmastd_audioDecodeStatusInfo *pAudioDecodeStatusInfo		O	デコード結果情報構造体
戻り値	ACMW_INT32 errorCode		エラー・コード 詳細は、表3.4を参照
説明	入力したデータ・ブロックをデコードするとき、この関数を実行してください。 【注】エラー発生時、デコード結果情報構造体の各メンバ値は不定です。参照しないでください。		

3.1.5 wmastd_ReconstructPcmData 関数

概要	PCMデータ生成処理		
構文	ACMW_INT32 wmastd_ReconstructPcmData(const wmastd_workMemoryInfo * const pWorkMemInfo, const wmastd_reconstructPcmDataConfigInfo * const pReconstructPcmDataConfigInfo, wmastd_reconstructPcmDataStatusInfo * const pReconstructPcmDataStatusInfo);		
機能	PCMデータを生成します。		
引数		I/O	意味
wmastd_workMemoryInfo *pWorkMemInfo		I	ワーク・メモリ情報構造体
wmastd_reconstructPcmDataConfigInfo *pReconstructPcmDataConfigInfo		I	PCM生成処理設定情報構造体
wmastd_reconstructPcmDataStatusInfo *pReconstructPcmDataStatusInfo		O	PCM生成処理結果情報構造体
戻り値	ACMW_INT32 errorCode		エラー・コード 詳細は、表3.4を参照
説明	PCMデータを生成するとき、この関数を実行してください。 【注】エラー発生時、デコード結果情報構造体の各メンバ値は不定です。参照しないでください。		

3.1.6 wmastd_GetErrorFactor 関数

概要	エラー要因情報取得処理		
構文	ACMW_UINT32 wmastd_GetErrorFactor(const wmastd_workMemoryInfo * const pWorkMemInfo);		
機能	直前に発生した、wmastd_Init、wmastd_AudecGetData, wmastd_AudioDecode, wmastd_ReconstructPcmData関数のエラー要因を返却します。		
引数		I/O	意味
wmastd_workMemoryInfo *pWorkMemInfo		I	ワーク・メモリ情報構造体
戻り値	ACMW_UINT32 errorFactor		エラー要因を示す値 詳細は、表3.6を参照
説明	直前に発生した、wmastd_Init、wmastd_AudecGetData, wmastd_AudioDecode, wmastd_ReconstructPcmData関数のエラー要因を取得します。 wmastd_Init、wmastd_AudecGetData, wmastd_AudioDecode, wmastd_ReconstructPcmData関数以外のエラー、およびwmastd_Init関数でスタティック領域の初期化が行われる前に発生したエラーについては、エラー要因を返却できません。 エラー要因は、次にwmastd_Init、wmastd_AudecGetData, wmastd_AudioDecode, wmastd_ReconstructPcmData関数が実行されると上書きされるため、エラー要因の取得が必要な場合は、これらの関数を実行する前に本関数を実行してください。		

3.1.7 wmastd_GetVersion 関数

概要	バージョン情報取得処理		
構文	ACMW_UINT32 wmastd_GetVersion(void);		
機能	本ミドルウェアのバージョン番号を返却します。		
引数		I/O	意味
無し		—	—
戻り値	ACMW_UINT32 versionCode		バージョン情報 例 ） 0x00000123の場合、バージョンは1.23となります 詳細は、表3.1を参照
説明	本ミドルウェアのバージョン番号を取得します。 任意のタイミングで実行可能です。		

表3.1　　versionCode の設定値

設定	値	説明
Customer ID (8bit)	0x00	標準版
	その他	予約
Release ID (8bit)	0x00	正式版
	0xA0 ～ 0xAF	α 版（機能制約のあるもの） 0xA1：　α1 ・・・ 0xA9：　α9 Other：　予約
	0xB0 ～ 0xBF	β 版（機能制約はないが、テスト未了のもの） 0xB1：　β1 ・・・ 0xB9：　β9 Other：　予約
	その他	予約
Major ID (8bit)	0xXY	Version XY.xy　（メジャー番号） X=0～9　かつ　Y=0～9　の場合 0x00：　Version 0.xy ・・・ 0x10：　Version10.xy ・・・ 0x99：　Version99.xy Other：　予約
Minor ID (8bit)	0xXY	Version xy.XY　（マイナー番号） X=0～9　かつ　Y=0～9　の場合 0x00：　Version xy.00 ・・・ 0x10：　Version xy.10 ・・・ 0x99：　Version xy.99 Other：　予約

3.2 構造体仕様

次項から、本ミドルウェアが提供する構造体について、以下の記述フォーマットに従って説明します。

【構造体名】 構造体名を説明します。

【機能】 構造体の機能を説明します。

【プロトタイプ】 構造体のプロトタイプを示します。

【メンバの説明】 構造体の各メンバについて説明します。

【備考】 構造体を使用する際の注意点などについて説明します。

3.2.1 メモリ・サイズ取得設定情報構造体

【構造体名】 wmastd_getMemorySizeConfigInfo

【機能】 wmastd_GetMemorySize関数により必要メモリ・サイズを取得する際に、必要メモリ・サイズ算出条件を指定します。

【プロトタイプ】
```
typedef struct {
    ACMW_INT32      reserve;
} wmastd_getMemorySizeConfigInfo;
```

【メンバの説明】

メンバ変数名	内容
reserve	予約 (0を設定してください)

【備考】 領域の確保はユーザが行ってください。ユーザは、wmastd_GetMemorySize関数を呼び出すより前に、領域の確保および値の設定を行ってください。

3.2.2 メモリ・サイズ取得結果情報構造体

【構造体名】 wmastd_getMemorySizeStatusInfo

【機能】 必要メモリ・サイズ計算処理(wmastd_GetMemorySize関数)により、必要メモリ・サイズ算出結果を格納します。

【プロトタイプ】
```
typedef struct {
    ACMW_UINT32     nStaticSize;
    ACMW_UINT32     nScratchSize;
    ACMW_UINT32     nInputBufferSize;
    ACMW_UINT32     nOutputBufferSize;
    ACMW_UINT32     nStackSize;
    ACMW_INT32      reserve;
} wmastd_getMemorySizeStatusInfo;
```

【メンバの説明】

メンバ変数名	内容
nStaticSize	スタティック領域の必要メモリ・サイズ[byte]
nScratchSize	スクラッチ領域の必要メモリ・サイズ[byte]
nInputBufferSize	入力バッファ領域の必要メモリ・サイズ[byte] *1
nOutputBufferSize	出力バッファ領域の必要メモリ・サイズ[byte] *1 *2
nStackSize	ソフトウェア・スタック領域の必要メモリ・サイズ[byte]
reserve	予約

【備考】 領域の確保はユーザが行ってください。ユーザは、wmastd_GetMemorySize関数を呼び出すより前に、領域の確保を行ってください。

【備考2】 *1：これらの値は推奨サイズです。ターゲット・システムにあわせて設計してください。
*2：出力バッファ領域のサイズは、ノン・インターリーブ形式で出力する場合の1チャネル分をあらわしています。

3.2.3 ワーク・メモリ情報構造体

【構造体名】 wmastd_workMemoryInfo

【機能】 本ミドルウェアで使用するワーク・メモリに関するアドレス情報を指定します。

【プロトタイプ】
```
typedef struct {
    void *          pStatic;
    void *          pScratch;
    ACMW_INT32      reserve;
} wmastd_workMemoryInfo;
```

【メンバの説明】

メンバ変数名	内容
pStatic	スタティック領域の先頭へのポインタ
pScratch	スクラッチ領域の先頭へのポインタ
reserve	予約 (0を設定してください)

【備考】 領域の確保はユーザが行ってください。ユーザは、引数として本構造体を必要とするライブラリ関数を呼び出すより前に、領域の確保および値の設定を行ってください。

【備考 2】 スタティック領域およびスクラッチ領域のサイズは、必要メモリ・サイズ計算処理(wmastd_GetMemorySize関数)で取得できます。

3.2.4 初期化設定情報構造体

【構造体名】 wmastd_initConfigInfo

【機能】 初期化処理(wmastd_Init関数)により初期化を行う際に、デコード条件を指定します。

【プロトタイプ】
```
typedef struct {
    ACMW_UINT16     wFormatTag;
    ACMW_UINT16     nChannels;
    ACMW_UINT32     nSamplesPerSec;
    ACMW_UINT32     nAvgBytesPerSec;
    ACMW_UINT16     nBlockAlign;
    ACMW_UINT16     nValidBitsPerSample;
    ACMW_UINT8      codecSpecificData[18];
    ACMW_INT32      reserve;
} wmastd_initConfigInfo;
```

【メンバの説明】

メンバ変数名	内容
wFormatTag	CODECの種類を示すID Stream Properties Object (Audio) の Type-Specific Dataに格納されている Codec ID / Format Tag (本ミドルウェアでの正常値は、0x0161です)
nChannels	チャネル数 Stream Properties Object (Audio) の Type-Specific Dataに格納されている Number of Channels (本ミドルウェアでの正常値は、1～2です)
nSamplesPerSec	サンプリング周波数 [Hz] Stream Properties Object (Audio) の Type-Specific Dataに格納されている Samples Per Second (本ミドルウェアでの正常値は、8000, 11025, 16000, 22050, 32000, 44100, 48000です)
nAvgBytesPerSec	平均バイト数 [byte/sec] Stream Properties Object (Audio) の Type-Specific Dataに格納されている Average Number of Bytes Per Second (本ミドルウェアでの正常値は、16以上、nSamplesPerSecの値の２倍未満です)
nBlockAlign	データ・ブロックのサイズ [byte] Stream Properties Object (Audio) の Type-Specific Dataに格納されている Block Alignment (本ミドルウェアでの正常値は、1以上です)
nValidBitsPerSample	サンプルあたりのビット数 [bit] Stream Properties Object (Audio) の Type-Specific Dataに格納されている Bits Per Sample (本ミドルウェアでの正常値は、16です)
codecSpecificData	Codec Specific Data Stream Properties Object (Audio) の Type-Specific Dataに格納されている Codec Specific Data (本ミドルウェアは、すべての値を正常値とみなして処理します。)
reserve	予約 (0を設定してください)

【備考】 領域の確保はユーザが行ってください。ユーザは、wmastd_Init関数を呼び出すより前に、領域の確保および値の設定を行ってください。

【 備 考 2 】 Stream Properties Objectについては、ASF仕様書(関連マニュアルを参照)の3.3 Stream Properties Object (mandatory, one per stream)を参照してください。
Type-Specific Dataについては、ASF仕様書(関連マニュアルを参照)の9.1 Audio media typeを参照してください。

表3.2 codecSpecificData[18]の設定方法

オフセット[byte]	型	サイズ[byte]	設定値
0	ACMW_UINT8	1	Codec Specific Data (1st element)
1	ACMW_UINT8	1	Codec Specific Data (2nd element)
2	ACMW_UINT8	1	Codec Specific Data (3rd element)
3	ACMW_UINT8	1	Codec Specific Data (4th element)
4	ACMW_UINT8	1	Codec Specific Data (5th element)
5	ACMW_UINT8	1	Codec Specific Data (6th element)
6	ACMW_UINT8	1	Codec Specific Data (7th element)
7	ACMW_UINT8	1	Codec Specific Data (8th element)
8	ACMW_UINT8	1	Codec Specific Data (9th element)
9	ACMW_UINT8	1	Codec Specific Data (10th element)
10	ACMW_UINT8	1	予約領域 (任意の値)
11	ACMW_UINT8	1	予約領域 (任意の値)
12	ACMW_UINT8	1	予約領域 (任意の値)
13	ACMW_UINT8	1	予約領域 (任意の値)
14	ACMW_UINT8	1	予約領域 (任意の値)
15	ACMW_UINT8	1	予約領域 (任意の値)
16	ACMW_UINT8	1	予約領域 (任意の値)
17	ACMW_UINT8	1	予約領域 (任意の値)

3.2.5 初期化結果情報構造体

【構造体名】 wmastd_initStatusInfo

【機能】 初期化処理(wmastd_Init)により初期化を行った結果を格納します。

【プロトタイプ】
```
typedef struct {
    ACMW_UINT16     nChannels;
    ACMW_UINT32     nSamplesPerSec;
    ACMW_INT32      reserve;
} wmastd_initStatusInfo;
```

【メンバの説明】

メンバ変数名	内容
nChannels	出力データのチャネル数
nSamplesPerSec	出力データのサンプリング周波数 [Hz]
reserve	予約

【備考】 領域の確保はユーザが行ってください。ユーザは、wmastd_Init関数を呼び出すより前に、領域の確保および値の設定を行ってください。

3.2.6 データ入力設定情報構造体

【構造体名】 wmastd_audecGetDataConfigInfo

【機能】 wmastd_AudecGetData関数によりデータ入力を行う際に、処理条件を指定します。

【プロトタイプ】
```
typedef struct {
    ACMW_BOOL       bNewPacket;
    ACMW_UINT8 *    pInDataStart
    ACMW_UINT32     nInDataSize
    ACMW_INT32      reserve;
} wmastd_audecGetDataConfigInfo;
```

【メンバの説明】

メンバ変数名	内容
bNewPacket	新しいデータ・ブロックになったことを示すフラグ 0 : 新しいデータ・ブロックではない other : 新しいデータ・ブロック
pInDataStart	入力データの開始アドレス
nInDataSize	入力データのサイズ [byte]
reserve	予約 (0を設定してください)

【備考】 領域の確保はユーザが行ってください。ユーザは、wmastd_AudecGetData関数を呼び出すより前に、領域の確保および値の設定を行ってください。

3.2.7 データ入力結果情報構造体

【構造体名】 wmastd_audecGetDataStatusInfo

【機能】 wmastd_AudecGetData関数によりデータ入力を行った結果を格納します。

【プロトタイプ】
```
typedef struct {
    ACMW_INT32    reserve;
} wmastd_audecGetDataStatusInfo;
```

【メンバの説明】

メンバ変数名	内容
reserve	予約

【備考】 領域の確保はユーザが行ってください。ユーザは、wmastd_AudecGetData関数を呼び出すより前に、領域の確保および値の設定を行ってください。

3.2.8 デコード設定情報構造体

【構造体名】 wmastd_audioDecodeConfigInfo

【機能】 wmastd_AudioDecode関数によりデコードを行う際に、処理条件を指定します。

【プロトタイプ】
```
typedef struct {
    ACMW_INT32      reserve;
} wmastd_audioDecodeConfigInfo;
```

【メンバの説明】

メンバ変数名	内容
reserve	予約 (0を設定してください)

【備考】 領域の確保はユーザが行ってください。ユーザは、wmastd_AudioDecode関数を呼び出すより前に、領域の確保および値の設定を行ってください。

3.2.9 デコード結果情報構造体

【構造体名】 wmastd_audioDecodeStatusInfo

【機能】 wmastd_AudioDecode関数によりデコードを行った結果を格納します。

【プロトタイプ】
```
typedef struct {
    ACMW_BOOL    bNoMoreInput;
    ACMW_UINT32  nDecodeSamples;
    ACMW_INT32   reserve;
} wmastd_audioDecodeStatusInfo;
```

【メンバの説明】

メンバ変数名	内容
bNoMoreInput	デコードすべきデータがないことを示すフラグ 0 : データあり 1 : データなし other : 予約
nDecodeSamples	PCMデータ生成処理(wmastd_ReconstructPcmData関数)を実行することによって生成可能なチャネルあたりのサンプル数
reserve	予約

【備考】 領域の確保はユーザが行ってください。ユーザは、wmastd_AudioDecode関数を呼び出すより前に、領域の確保および値の設定を行ってください。

3.2.10 PCM 生成処理設定情報構造体

【構造体名】 wmastd_reconstructPcmDataConfigInfo

【機能】 wmastd_ReconstructPcmData関数によりPCM生成を行う際に、処理条件を指定します。

【プロトタイプ】
```
typedef struct {
    ACMW_UINT32     nRequestSamples;
    void **         pOutBuffStart;
    ACMW_UINT32     nOutBuffSize;
    ACMW_UINT16     nOutBuffFormat;
    ACMW_INT32      reserve;
} wmastd_reconstructPcmDataConfigInfo;
```

【メンバの説明】

メンバ変数名	内容
nRequestSamples	PCMデータ生成処理(wmastd_ReconstructPcmData関数)を実行することによって生成したいチャネルあたりのサンプル数
pOutBuffStart	チャネルごとの出力データの開始アドレス
nOutBuffSize	チャネルごとの出力バッファのサイズ [byte]
nOutBuffFormat	出力バッファのデータ・フォーマット指定 0 : インターリーブ形式(16 [bit/sample]) other : ノン・インターリーブ形式(32 [bit/sample])
reserve	予約 (0を設定してください)

【備考】 領域の確保はユーザが行ってください。ユーザは、wmastd_ReconstructPcmData関数を呼び出すより前に、領域の確保および値の設定を行ってください。

【備考2】 pOutBuffStartには、ユーザが確保した、出力チャネル数分の出力バッファのポインタ配列の先頭アドレスを格納します。
nOutBuffFormatで、ノン・インターリーブ形式を指定した場合、出力バッファのポインタには、４バイト・データに対するメモリ・アクセス可能なアライメントのアドレスを指定してください。
nOutBuffFormatで、インターリーブ形式を指定した場合、pOutBuffStart[0]に出力バッファの開始アドレスを指定してください。この場合、アライメントに制限はありません。

【備考3】 nOutBuffSizeの値は、各チャネルで共通です。

3.2.11 PCM 生成処理結果情報構造体

【構造体名】 wmastd_reconstructPcmDataStatusInfo

【機能】 wmastd_ReconstructPcmData関数によりPCM生成を行った結果を格納します。

【プロトタイプ】
```
typedef struct {
    ACMW_UINT32     nReconstructedSamples;
    ACMW_UINT32     nChannels;
    void **         pOutBuffLast;
    ACMW_UINT32     nOutBuffUsedDataSize;
    ACMW_UINT16     nValidBitsPerSample;
    ACMW_UINT32     nChannelMask;
    ACMW_INT32      reserve;
} wmastd_reconstructPcmDataStatusInfo;
```

【メンバの説明】

メンバ変数名	内容
nReconstructedSamples	PCMデータ生成処理(wmastd_ReconstructPcmData関数)を実行することによって生成されたチャネルあたりのサンプル数
nChannels	出力チャネル数
pOutBuffLast	PCMデータ生成処理により、チャネルごとに出力したデータの書き出し終了後アドレス
nOutBuffUsedDataSize	PCMデータ生成処理後の各チャネルの出力バッファの消費データ・サイズ [byte]
nValidBitsPerSample	サンプルあたりのビット数 [bit]
nChannelMask	出力チャネル情報(表3.3参照)
reserve	予約

【備考】 領域の確保はユーザが行ってください。ユーザは、wmastd_ReconstructPcmData関数を呼び出すより前に、領域の確保および値の設定を行ってください。

【備考2】 pOutBuffLastには、wmastd_ReconstructPcmData関数を呼び出すより前に、ユーザが領域確保したポインタ配列の先頭アドレスを指定してください。

【備考3】 PCM生成処理設定情報構造体のメンバnOutBuffFormatに0を指定した場合、pOutBuffLast[0]、および、nOutBuffUsedDataSizeに出力バッファの書き込み終了アドレス、および、出力バッファの消費データ・サイズが出力されます。
PCM生成処理設定情報構造体のメンバnOutBuffFormatに0以外を指定した場合、pOutBuffLast[0～nChannels]、および、nOutBuffUsedDataSizeに各チャネルの出力バッファの書き込み終了アドレス、および、各チャネルの出力バッファの消費データ・サイズが出力されます。

表3.3 nChannelMask のビット・フィールドの意味

ビット・フィールド	数値	意味
bit 0	0x00000001	フロント左チャネル
bit 1	0x00000002	フロント右チャネル
bit 2	0x00000004	フロント・センター・チャネル　／　モノラル時のチャネル
bit 3	0x00000008	LFE(Low Frequency Effect)チャネル(未使用)
bit 4	0x00000010	バック左チャネル(未使用)
bit 5	0x00000020	バック右チャネル(未使用)
bit 6	0x00000040	フロント左センター・チャネル(未使用)
bit 7	0x00000080	フロント右センター・チャネル(未使用)
bit 8	0x00000100	バック・センター・チャネル(未使用)
bit 9	0x00000200	サイド左チャネル(未使用)
bit 10	0x00000400	サイド右チャネル(未使用)
bit11	0x00000800	上センター・チャネル(未使用)
bit 12	0x00001000	上フロント左チャネル(未使用)
bit 13	0x00002000	上フロント・センター・チャネル(未使用)
bit 14	0x00004000	上フロント右チャネル(未使用)
bit 15	0x00008000	上バック左チャネル(未使用)
bit 16	0x00010000	上バック・センター・チャネル(未使用)
bit 17	0x00020000	上バック右チャネル(未使用)
bit 18	0x00040000	予約
bit 19	0x00080000	予約
bit 20	0x00100000	予約
bit 21	0x00200000	予約
bit 22	0x00400000	予約
bit 23	0x00800000	予約
bit 24	0x01000000	予約
bit 25	0x02000000	予約
bit 26	0x04000000	予約
bit 27	0x08000000	予約
bit 28	0x10000000	予約
bit 29	0x20000000	予約
bit 30	0x40000000	予約
bit 31	0x80000000	予約

3.3 エラー処理

本ミドルウェアの関数は、表3.4のエラー・コードが返却されます。詳細なエラー要因情報を取得したい場合は、wmastd_GetErrorFactor関数を実行してください。

3.3.1 エラー・コード

本ミドルウェアのエラー・コードについて説明します。

表3.4 エラー・コード

エラー・コード (32bit)	値	説明	再初期化
WMASTD_RESULT_OK	0x00000000	処理結果が正常です。 処理が正常に終了したことを示します。	不要
WMASTD_RESULT_NG	0x00000001	処理結果が異常です。 構造体に指定されたパラメータが不正、もしくは、正しく動作ができませんでした。 PCM データは出力されません。構造体に正しいパラメータを指定するか、正しい手順により、再度実行可能です。	不要
WMASTD_RESULT_WARNING	0x00000002	処理継続可能な異常が発生しました。 デコーダがエラーを検出したが、PCM データは出力されました。その際、エラー・コンシールメントまたは MUTE 信号（すべて 0）が出力された可能性があります。その場合、エラー要因情報取得処理(wmastd_GetErrorFactor 関数)によるエラー要因取得によって確認してください。	不要
WMASTD_RESULT_FATAL	0x00000003	処理継続不可能な異常が発生しました。 処理の継続が不可能です。 PCM データは出力されません。プログラムの再初期化を行う必要があります。wmastd_GetErrorFactor 関数によるエラー要因取得はできません。	必要
その他	上記以外	予約	ー

表3.5 ライブラリ関数で使用されるエラー・コード

関数 / エラー・コード	wmastd_GetMemorySize	wmastd_Init	wmastd_AudecGetData	wmastd_AudioDecode	wmastd_ReconstructPcmData	wmastd_GetErrorFactor *1	wmastd_GetVersion *2
WMASTD_RESULT_OK	○	○	○	○	○	ー	ー
WMASTD_RESULT_NG	ー	○	○	○	○	ー	ー
WMASTD_RESULT_WARNING	ー	ー	○	○	○	ー	ー
WMASTD_RESULT_FATAL	○	○	○	○	○	○	ー

【注】○：出力する可能性がある、ー：使用しない

【注】*1 エラー要因を返します

【注】*2 バージョン情報を返します

3.3.2 エラー要因

エラー要因は、エラー発生時の詳細エラー情報を示します。WMASTD_RESULT_FATAL が発生した場合を除き、wmastd_GetErrorFactor 関数により、エラー要因を取得することができます。エラー要因の一覧を、表 3.6 に示します。また、ライブラリ関数とエラー要因、エラー・コードの対応を、表 3.7に示します。

表3.6 エラー要因一覧

errorFactor(32bit)	値	説明
WMASTD_ERR_NONE	0x00000000	正常終了した。エラー要因は存在しない。
WMASTD_ERR_POINTER	0x00000010	ポインタ値が不正だった。
WMASTD_ERR_PARAMETER	0x00000020	パラメータが不正だった。
WMASTD_ERR_SEQUENCE	0x00000040	本ライブラリ関数の実行順序が不正だった。
WMASTD_ERR_LOST_PACKET	0x00000100	入力データが欠落していることを検出した。
WMASTD_ERR_BROKEN_FRAME	0x00000200	入力データの構造が異常であることを検出した。
WMASTD_ERR_NOT_SUPPORTED_DATA	0x00010000	サポートしていないデータであることを検出した。
WMASTD_ERR_BAD_ARGUMENT	0x01000000	内部関数の引数が不正であることを検出した。
WMASTD_ERR_MW_FAILED	0x02000000	内部的に異常を検出した。
WMASTD_ERR_OUT_OF_MEMORY	0x04000000	メモリ使用量が不正であることを検出した。
WMASTD_ERR_WRONG_STATE	0x08000000	内部ステートが不正であることを検出した。
その他	上記以外	予約

表3.7 ライブラリ関数とエラー要因、エラー・コードの対応

関数 / エラー・コード	wmastd_Init	wmastd_AudecGetData	wmastd_AudioDecode	wmastd_ReconstructPcmData
WMASTD_ERR_NONE	OK	OK	OK	OK
WMASTD_ERR_POINTER	NG	NG	NG	NG
WMASTD_ERR_PARAMETER	NG	-	-	NG
WMASTD_ERR_SEQUENCE	-	NG	NG	NG/WARNING
WMASTD_ERR_LOST_PACKET	NG	WARNING	-	NG
WMASTD_ERR_BROKEN_FRAME	NG	NG	NG	NG
WMASTD_ERR_NOT_SUPPORTED_DATA	NG	NG	-	NG
WMASTD_ERR_BAD_ARGUMENT	NG	NG	-	NG
WMASTD_ERR_MW_FAILED	NG	NG	WARNING	NG
WMASTD_ERR_OUT_OF_MEMORY	NG	NG	-	NG
WMASTD_ERR_WRONG_STATE	NG	NG	-	NG

【注】表中の文字は、エラー・コードを示しています。

“OK” : WMASTD_RESULT_OK

“NG” : WMASTD_RESULT_NG

“WARNING” : WMASTD_RESULT_WARNING

3.4 メモリ仕様

本ミドルウェアが使用するメモリ領域について説明します。

3.4.1 スクラッチ領域

表3.8 スクラッチ領域情報

内容	本ミドルウェアを使用する場合、一時的に値を保存する領域です。 本ミドルウェア関数を呼び出し中に、割り込み処理などでこの領域を操作した場合、本ミドルウェアの正常な動作は保証されません。 1フレームのデコード処理の後、ユーザが自由に使用することができます。
シンボル名	—（ユーザが任意で指定）
サイズ	実際に必要なサイズは、wmastd_GetMemorySize で取得してください。
領域確保	ユーザが領域を確保してください。 本ミドルウェア関数から戻った後、ユーザが自由に使用することができます。ただし、ユーザが使用した後、本ミドルウェア関数を呼び出す場合、この領域に保存されたユーザが保存した値は上書きされます。
配置	RAM に配置
アライメント	8バイト境界に配置してください。

3.4.2 スタティック領域

表3.9 スタティック領域情報

内容	本ミドルウェアを使用する場合、常に値を保存する領域です。 初期化処理以降にユーザがこの領域を操作した場合、本ミドルウェアの正常な動作は保証されません。
シンボル名	—（ユーザが任意で指定）
サイズ	実際に必要なサイズは、wmastd_GetMemorySizeで取得してください。
領域確保	ユーザが領域を確保してください。
配置	RAM に配置
アライメント	8バイト境界に配置してください。

3.4.3 ソフトウェア・スタック領域

表3.10 ソフトウェア・スタック領域情報

内容	本ミドルウェアが使用する、スタック領域です。
シンボル名	—（ユーザが任意で指定）
サイズ	実際に必要なサイズは、wmastd_GetMemorySizeで取得してください。
領域確保	ユーザが領域を確保します。 本ミドルウェアを使用する場合、上記のサイズ以上にソフトウェア・スタック領域を確保してください。
配置	RAM に配置
アライメント	—

3.4.4 ヒープ領域

本ミドルウェアではヒープ領域は使用しません。

3.4.5 入力バッファ

表3.11 入力バッファ領域情報

内容	本ミドルウェアの入力データを格納するための領域です。 入力バッファに格納されるデータは、WMA BitstreamからParserにより切り出したデータ・ブロックです。 デコード処理中にこの領域を操作すると、正常動作を保証できません。 【注】本ミドルウェアは、リング・バッファ構造の入力バッファには対応しておりません。
シンボル名	—（ユーザが任意で指定）
サイズ	任意です。ただし、4byte未満のサイズで領域確保しないでください。 本ミドルウェアのwmastd_GetMemorySize関数で得られるnInputBufferSize値は、推奨サイズです。ターゲット・システムにあわせて設計してください。
領域確保	ユーザが領域を確保します。 wmastd_AudioDecode関数実行後、デコード結果情報構造体のbNoMoreInput値が1になったら、ユーザは、この領域を自由に使用できます。
配置	RAM に配置
アライメント	制限はありません。

wmastd_AudecGetData 関数実行時に、データ入力設定情報構造体へ各パラメータを設定し、処理結果がバッファ・メモリ結果情報構造体へ格納されます。入力バッファと各構造体メンバの関係を図 3.1に示します。

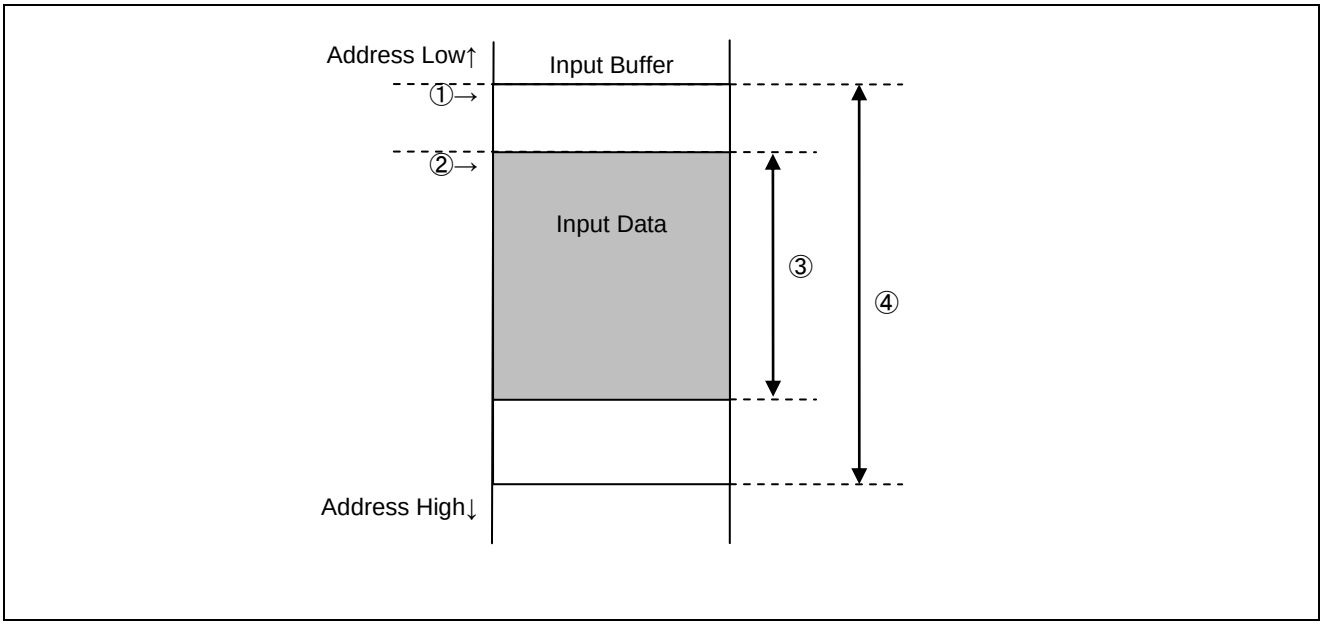

図3.1 入力バッファにおける各種構造体メンバの情報イメージ例

【注】入力バッファにデータ・ブロックを入れ、デコード処理を行った場合を示しています。

表3.12 入力バッファにおける各種構造体メンバの情報

①	-	-	入力バッファの開始アドレス
②	pInBuffStart	(データ入力設定情報構造体)	入力データの開始アドレス
③	nInDataSize	(データ入力設定情報構造体)	入力データ・サイズ
④	-	-	入力バッファ・サイズ

【注】①～②はアドレスを示し、③～④はサイズを表しています。

(1) 入力データ格納方式

入力データの格納方式を図 3.2に示します。入力バッファ(メモリ)には、1 バイト単位でデータを格納してください。

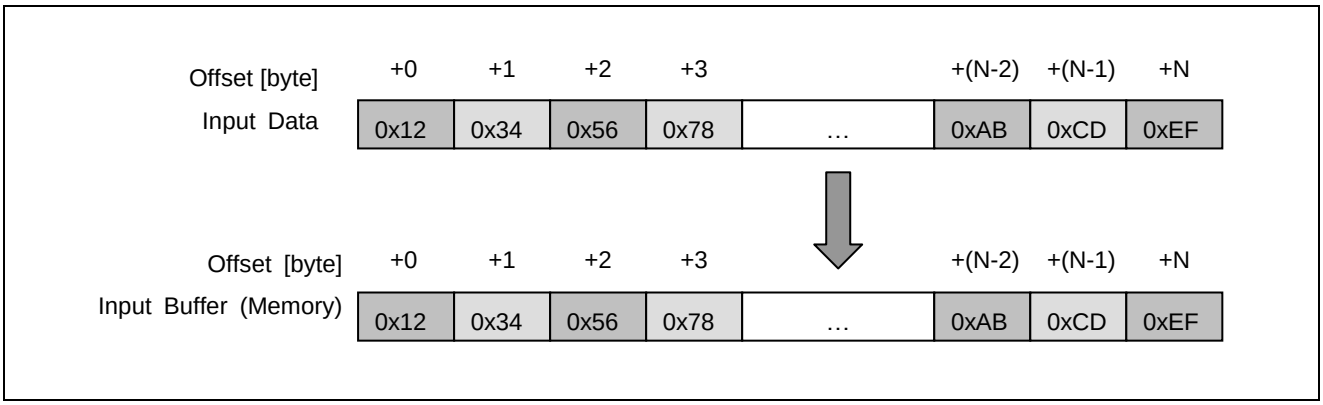

図3.2 入力データの格納方式

(2) データ入力方法

WMA Standard CODEC でエンコードされたデータが格納された Payload Data もしくは Sub-Payload #N Data を、データ・ブロックごとに入力してください(N = 0, 1, …)。

データ・ブロックを入力するときは、入力バッファに 4 バイト以上のデータを入力するようにしてください（データ・ブロックのサイズが 4 バイト未満の場合は除きます。3.1.3項を参照してください）。

Payload Data, Sub-Payload #N Data とデータ・ブロックとの関係については、3.5節を参照してください。

入力バッファのサイズ ＜ データ・ブロックのサイズ の場合

1 個のデータ・ブロックを複数回に分けて入力することが可能です。

入力バッファに格納できる分だけデータを格納し、データ・ブロックが切り替わるときにのみ、bNewPacket(入力データ設定情報構造体のメンバ)に値 1 を設定してください。

入力バッファのサイズ ≧ データ・ブロックのサイズ の場合

1 個のデータ・ブロックを 1 度に入力することが可能です。

入力バッファに、1 個のデータ・ブロックを全て格納し、bNewPacket(入力データ設定情報構造体のメンバ)には常に値 1 を設定してください。

3.4.6 出力バッファ

表3.13 出力バッファ領域情報

内容	本ミドルウェアの出力データを格納するための領域です。 出力バッファに格納されるデータは、16ビット・リニアPCMデータ（以下、PCMデータ）です。 デコード処理中にこの領域を操作すると、正常動作を保証できません。
シンボル名	—（ユーザが任意で指定）
サイズ	任意です。ただし、4byte未満のサイズで領域確保しないでください。 本ミドルウェアのwmastd_GetMemorySize関数で得られるnOutputBufferSize値は、推奨サイズです。ターゲット・システムにあわせて設計してください。 正常なデータをデコードした場合、デコード結果情報構造体のnDecodeSamplesの値は4096以下となるため、4096サンプル分のデータが格納できる出力バッファを確保すれば、生成可能なPCMデータを一度にすべて出力させることが可能です。 【注】wmastd_GetMemorySize関数で取得できるサイズは、ノン・インターリーブ形式で出力する場合の1チャネルあたりのサイズです。
領域確保	ユーザが領域を確保してください。 PCM生成処理後に、ユーザは、この領域を自由に使用できます。
配置	RAM に配置
アライメント	ノン・インターリーブ形式で出力する場合は、4バイト境界に配置してください。 インターリーブ形式で出力する場合は、制限はありません。

wmastd_ReconstructPcmData 関数実行時に、PCM 生成処理設定情報構造体へ各パラメータを設定し、処理結果が PCM 生成処理結果情報構造体へ格納されます。出力バッファと各構造体メンバの関係を図3.3に示します。

ノン・インターリーブ形式で出力する場合は、2 チャネル目以降も 1 チャネル目と同様に管理します。1 チャネル目と 2 チャネル目の出力バッファは連続していても、していなくても問題ありません。出力バッファに出力されるサンプル数は、入力データやデコード条件によって変化します。PCM 生成処理結果情報構造体の nReconstructedSamples を参照してください。

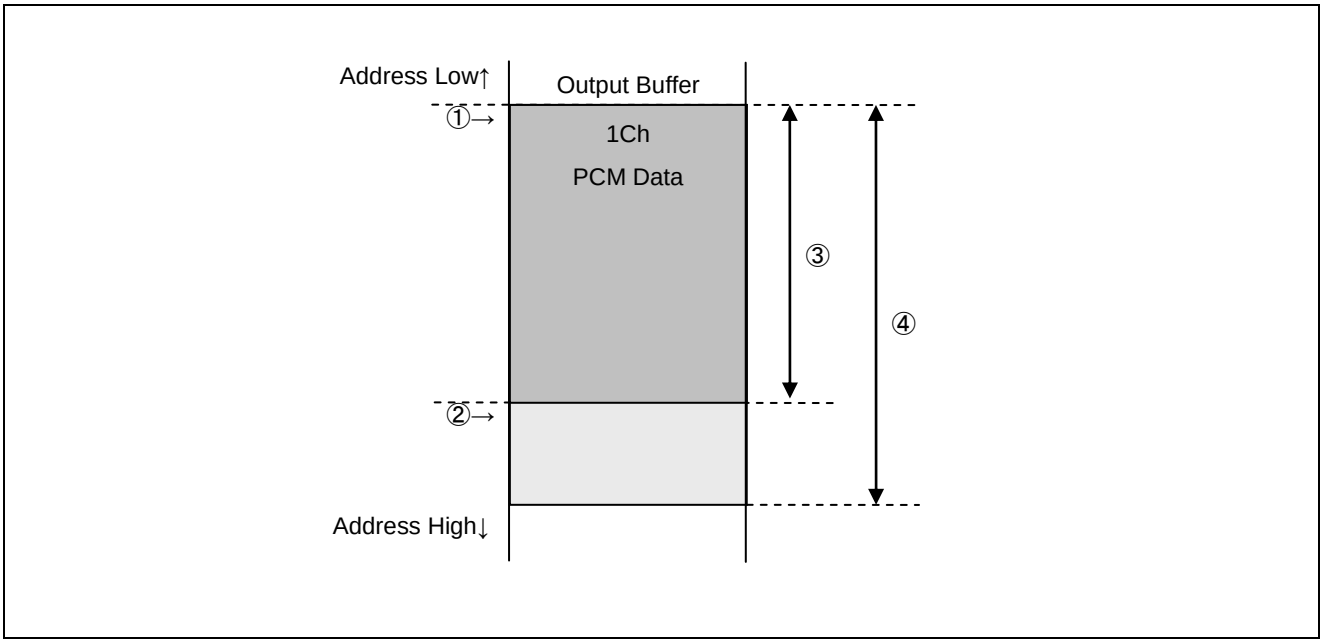

図3.3 出力バッファにおける各種構造体メンバの情報イメージ例

【注】PCM生成処理で出力バッファに出力結果を入れた場合を示しています。

表3.14 出力バッファにおける各種構造体メンバの情報

①	pOutBuffStart[0]	(PCM生成処理設定情報構造体)	1 チャネル目の出力データ開始アドレス
②	pOutBuffLast[0]	(PCM生成処理結果情報構造体)	1 チャネル目の書き出し終了後アドレス
③	nOutBuffUsedDataSize	(PCM生成処理結果情報構造体)	チャネルあたりの消費データ・サイズ
④	nOutBuffSize	(PCM生成処理設定情報構造体):	チャネルあたりの出力バッファ・サイズ

【注】①～②はアドレスを示し、③～④はサイズを表しています。

(1) 出力データ格納方式(インターリーブ形式の場合)

出力データは、図 3.5のような形式で出力します。出力バッファ(メモリ)には、2 バイト(16 ビット)単位でデータを格納します。バッファにアクセスする際のエンディアン(図 3.4参照)は、リトル・エンディアンになります。

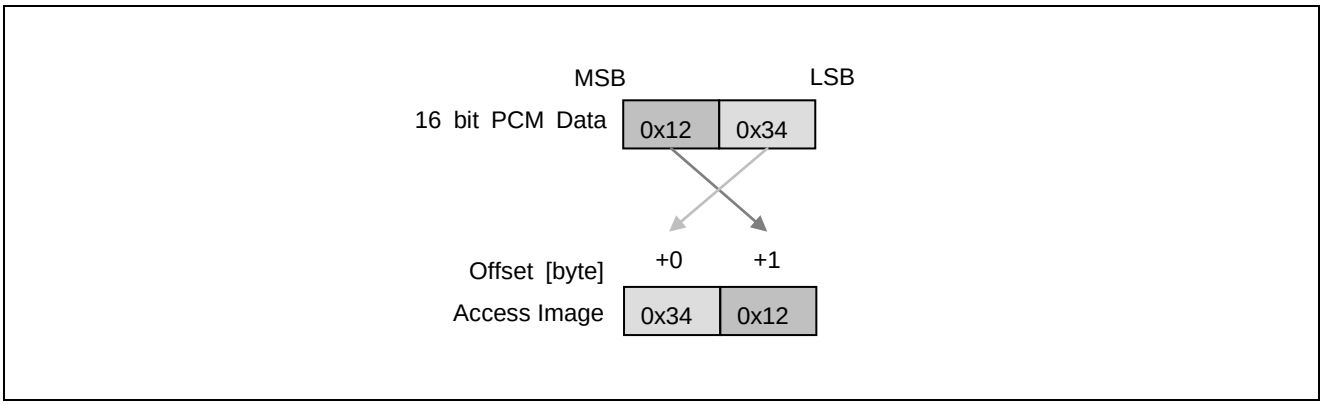

図3.4 16bit PCM データ(リトル・エンディアン)アクセス

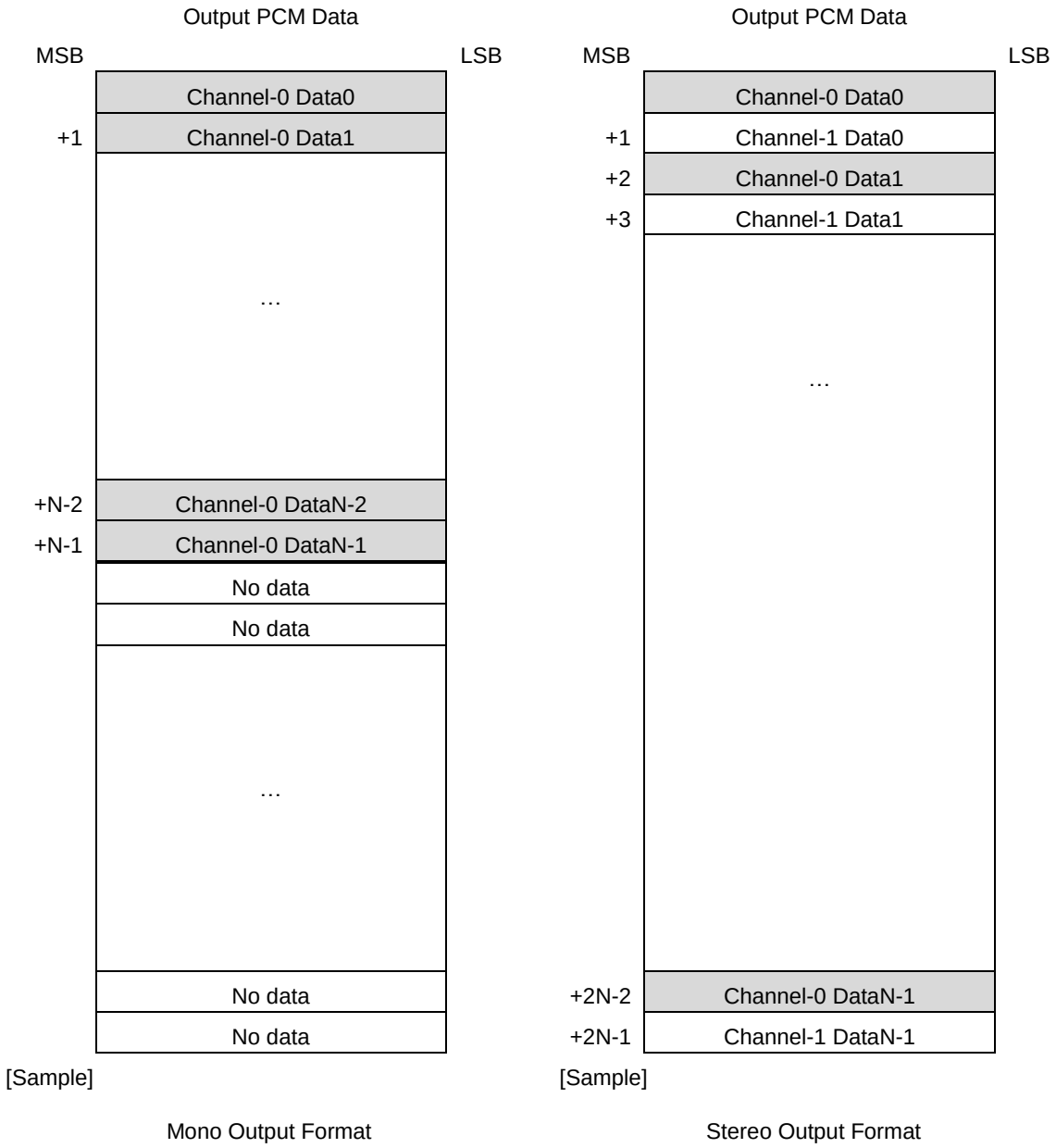

図3.5 出力 PCM データ(インターリーブ形式)の出力イメージ例

(2) 出力データ格納方式(ノン・インターリーブ形式の場合)

出力データは、チャネルごとに、図 3.8のような形式で出力します。出力バッファ(メモリ)には、図 3.7の形式で 4 バイト(32 ビット)化されたデータを格納します。バッファにアクセスする際のエンディアン(図 3.6参照)は、リトル・エンディアンになります。

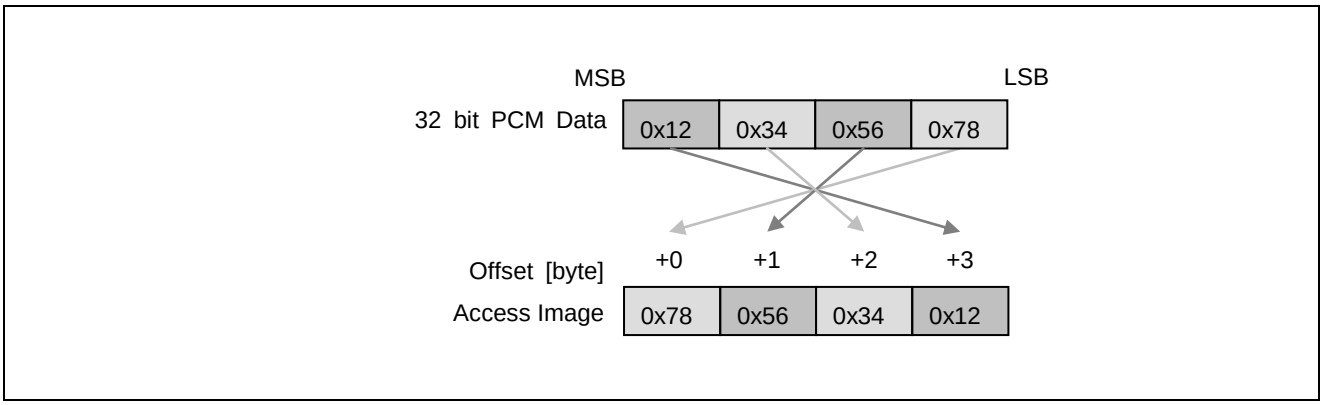

図3.6 32bit PCM データ(リトル・エンディアン)アクセス

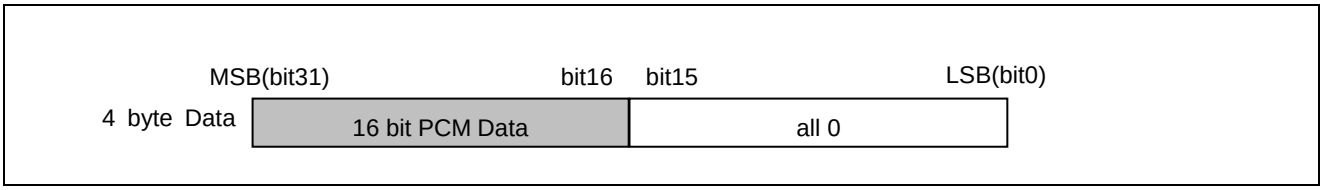

図3.7 4byte データ化による PCM データのビット配置

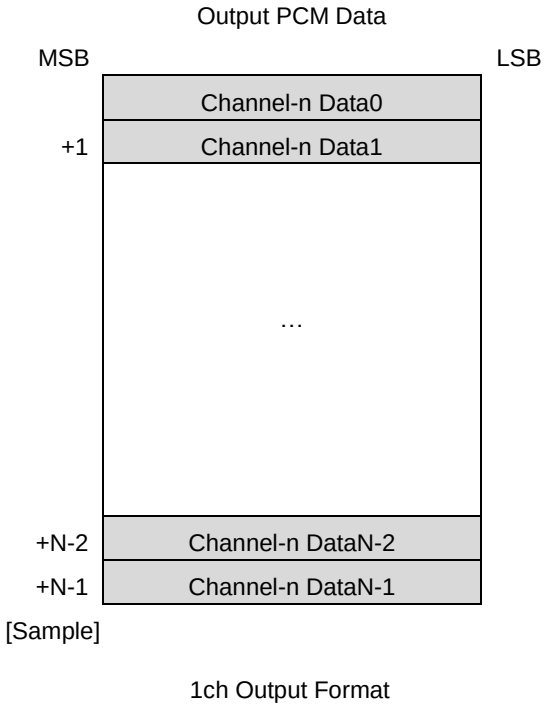

図3.8 出力 PCM データ(ノン・インターリーブ形式)の出力イメージ例

【注】各チャネルのバッファはバッファ同士が不連続なアドレスにも配置可能です

3.5 入力データ

本ミドルウェアでは、WMA Standard CODEC でエンコードされたデータが格納された Payload Data もしくは Sub-Payload #N Data を、データ・ブロックごとに入力してください(N = 0, 1, …)。

入力データの格納方式、データ入力方法については、3.4.5項を参照してください。
Payload Data, Sub-Payload #N Data とデータ・ブロックとの関係については、以下に示します。

Payload Data, Sub-Payload #N Data については、Microsoft 社が公開している ASF 仕様書(関連マニュアルを参照)を参照してください。ASF 仕様書内の説明箇所を、表 3.15に示します。

表3.15 入力データについての説明箇所

入力データ	ASF仕様書での説明箇所
Payload Data	5.2.3.1 Single payload 5.2.3.3 Multiple payloads
Sub-Payload #N Data	5.2.3.2 Single payload, compressed payload data 5.2.3.4 Multiple payload, compressed payload data

Payload Data, Sub-Payload #N Data とデータ・ブロックとの関係

Payload Data もしくは Sub-Payload #N Data は、1 以上（整数）個のデータ・ブロックから構成されています。データ・ブロックのサイズは、ファイルごとに固定で、Stream Properties Object (Audio)の Type-Specific Data に格納されている Block Alignment で指定されています。

Stream Properties Object については、Microsoft 社が公開している ASF 仕様書(関連マニュアルを参照)を参照してください。

3.6 出力データ

16ビット・リニアPCMデータを出力します。
出力データの格納方式については、3.4.6項を参照してください。

4. 注意事項

アプリケーション作成時の注意事項について示します。

4.1 関数呼び出しに関する注意事項

本ミドルウェアの関数を呼び出すユーザ・プログラムは、使用するコンパイラの呼び出しルールに従ってください。

4.1.1 関数実行タイミング

本ミドルウェアの関数を実行するタイミングについて説明します。

(1) wmastd_GetMemorySize 関数

任意のタイミングで実行可能です。ただし、wmastd_Init 関数を実行する前に、この関数を実行し、必要なサイズのメモリを確保してください。

(2) wmastd_Init 関数

一連のデコード処理を開始する前に、この関数を 1 度だけ実行してください。ここで “一連のデコード処理” とは、あるストリームのデコード開始からデコード終了をさします。

(3) wmastd_AudecGetData 関数

データ・ブロックをミドルウェアへ入力するとき、この関数を実行してください。

(4) wmastd_AudioDecode 関数

入力したデータ・ブロックをデコードするとき、この関数を実行してください。

(5) wmastd_ReconstructPcmData 関数

PCM データを生成するとき、この関数を実行してください。

(6) wmastd_GetErrorFactor 関数

wmastd_Init 関数実行後の任意のタイミングで実行可能です。

(7) wmastd_GetVersion 関数

任意のタイミングで実行可能です。

4.2 その他注意事項

4.2.1 メモリ領域の確保・配置

本ミドルウェアの関数を呼び出す前に、スタティック領域およびスクラッチ領域、入出力バッファ領域および各関数の引数で使用する各構造体を確保しておいてください。

4.2.2 範囲外メモリ・アクセス

本ミドルウェアは、確保した領域以外のメモリ空間へのメモリ・アクセスを行ないません。

4.2.3 他のアプリケーションとの組み合わせ

本ミドルウェアと他のアプリケーションを組み合わせて使用するときには、シンボル名の重複に注意してください。

4.2.4 ミドルウェアの監視

本ミドルウェア組み込み時は、システムがハングアップしないよう本ミドルウェアのデコード処理時間をタイマなどで監視し、上位プログラムにタイムアウト処理を実装して下さい。

改訂記録	WMA Decode Middleware ユーザーズマニュアル

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2014.07.07	ー	初版発行
1.01	2014.08.29	3	「表 1.2 必要メモリ・サイズ」のフォーマットを変更
		52	「4.2.4 ミドルウェアの監視」の項目を追記

WMA Decode Middleware ユーザーズマニュアル

発行年月日　2014年8月29日　Rev.1.01

発行　ルネサス エレクトロニクス株式会社
〒211-8668　神奈川県川崎市中原区下沼部1753

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社 〒100-0004 千代田区大手町2-6-2（日本ビル）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/

WMA Decode Middleware

RENESAS

ルネサス エレクトロニクス株式会社