User's Manual

AAC 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.1.00　2014.12

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。

また、本ミドルウェアに含まれる技術および特許の使用に際しては、お客様自身で別途、以下のライセンス契約を締結していただく必要があります。

Via Licensing Corporation (http://www.vialicensing.com/)

3. 関連マニュアル

AAC 関連資料

規格番号・タイトル	発行日
ISO/IEC 13818-7:2006 Information technology Generic coding of moving pictures and associated audio information Part 7: Advanced Audio Coding (AAC)-Forth Edition	2006/01/15
ISO/IEC 14496-3:2005 Information Technology - Coding of Audio-Visual Objects - Part 3: Audio-Third Edition	2005/12/01

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
AAC LC	AAC Low Complexity	低演算 AAC
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
bps	bits per second	転送速度を表す単位、ビット/秒
CRC	Cyclic Redundancy Check	巡回冗長検査
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
ETSI	European Telecommunications Standards Institute	欧州電気通信標準化機構
High Quality SBR	High Quality Spectral Band Replication	高品質スペクトル帯域複製
IEC	International Electrotechnical Commission	国際電気標準会議
ISO	International Organization for Standardization	国際標準化機構
LATM	Low overhead MPEG-4 audio transport multiplex	AAC の入力データ形式の一種
Low Power SBR	Low Power Spectral Band Replication	低消費電力スペクトル帯域複製
LSB	Least Significant Bit	最下位ビット
MDCT	Modified Discrete Cosine Transform	修正離散コサイン変換
MSB	Most Significant Bit	最上位ビット
PCM	Pulse Code Modulation	パルス符号変調
PS	Parametric Stereo	パラメトリックステレオ
QMF	Quadrature Mirror Filter	直交ミラーフィルタ
RAM	Random Access Memory	書き込みメモリ
ROM	Read Only Memory	読み出し専用メモリ
SBR	Spectral Band Replication	スペクトル帯域複製

目次

1. 概要

本章では、AAC デコーダの概要について説明します。

1.1 仕様概要

AACはAdvanced Audio Codingの略で、ISO/IEC 13818-7(MPEG-2 AAC)および14496-3(MPEG-4 AAC)で標準化されています。

MP3(MPEG Audio layer-3)との互換性を排除することにより、高品質でかつ高圧縮率を実現しています。

本ミドルウェアは、AAC の復号方式に対応し、入力された圧縮データをデコードして出力します。

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

<table>
<tr><th>項目</th><th>内容</th></tr>
<tr><td>入力データ形式</td><td>①MPEG-2 (AAC LC) : ISO/IEC 13818-7:2006(Fourth Edition)
②MPEG-4 (AAC LC) : ISO/IEC 14496-3:2005(Third Edition)</td></tr>
<tr><td>出力データ形式</td><td>16/32 ビット・リニア PCM</td></tr>
<tr><td>対応サンプリング周波数(Hz)</td><td>8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100, 48000, 64000, 88200, 96000</td></tr>
<tr><td>対応チャネル数</td><td>最大 2 チャネル</td></tr>
<tr><td>対応ビットレート(kbps)</td><td>チャネルあたりの最小ビットレート : 8kbps
チャネルあたりの最大ビットレート:
<table>
<tr><th>サンプリング周波数(Hz)</th><th>最大ビットレート [kbps]</th></tr>
<tr><td>8000</td><td>48</td></tr>
<tr><td>11025</td><td>66.15</td></tr>
<tr><td>12000</td><td>72</td></tr>
<tr><td>16000</td><td>96</td></tr>
<tr><td>22050</td><td>132.3</td></tr>
<tr><td>24000</td><td>144</td></tr>
<tr><td>32000</td><td>192</td></tr>
<tr><td>44100</td><td>264.6</td></tr>
<tr><td>48000</td><td>288</td></tr>
<tr><td>64000</td><td>384</td></tr>
<tr><td>88200</td><td>529.2</td></tr>
<tr><td>96000</td><td>576</td></tr>
</table></td></tr>
<tr><td>CRC</td><td>入力データ形式が ADTS の場合、誤り補償として、CRC を行います。</td></tr>
<tr><td>リエントラント</td><td>対応</td></tr>
<tr><td>制限事項</td><td>下記機能には非対応
・ゲイン・コントロール機能
（LC では未定義のため、ストリーム内に存在する場合エラーになります。）
・CCE(Coupling Channel Element)のデコード処理
（ストリーム中に CCE が存在した場合は、エラーになります。）
下記データは非対応
・LC 以外のプロファイル
・MPEG-4 コンテナ形式で包括されたデータ
・ADTS 形式で、1 フレーム内の RAW データ・ブロック数が 2 以上
・ADTS 形式で、エンファシス(2bit)が付加されたデータ(ISO/IEC 14496-3 初期規格)
・フロントチャネル以外のチャネルが存在するデータ
・2 チャネルを越えるデータ</td></tr>
</table>

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			65,896
		<内訳>	スタティック領域	<内訳>	37,192
			スクラッチ領域		28,680
		ユーザ・ワーク領域			10,870
		<内訳>	入力バッファ	<内訳>	1,536
			出力バッファ		4,096
			各種構造体		630
		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。

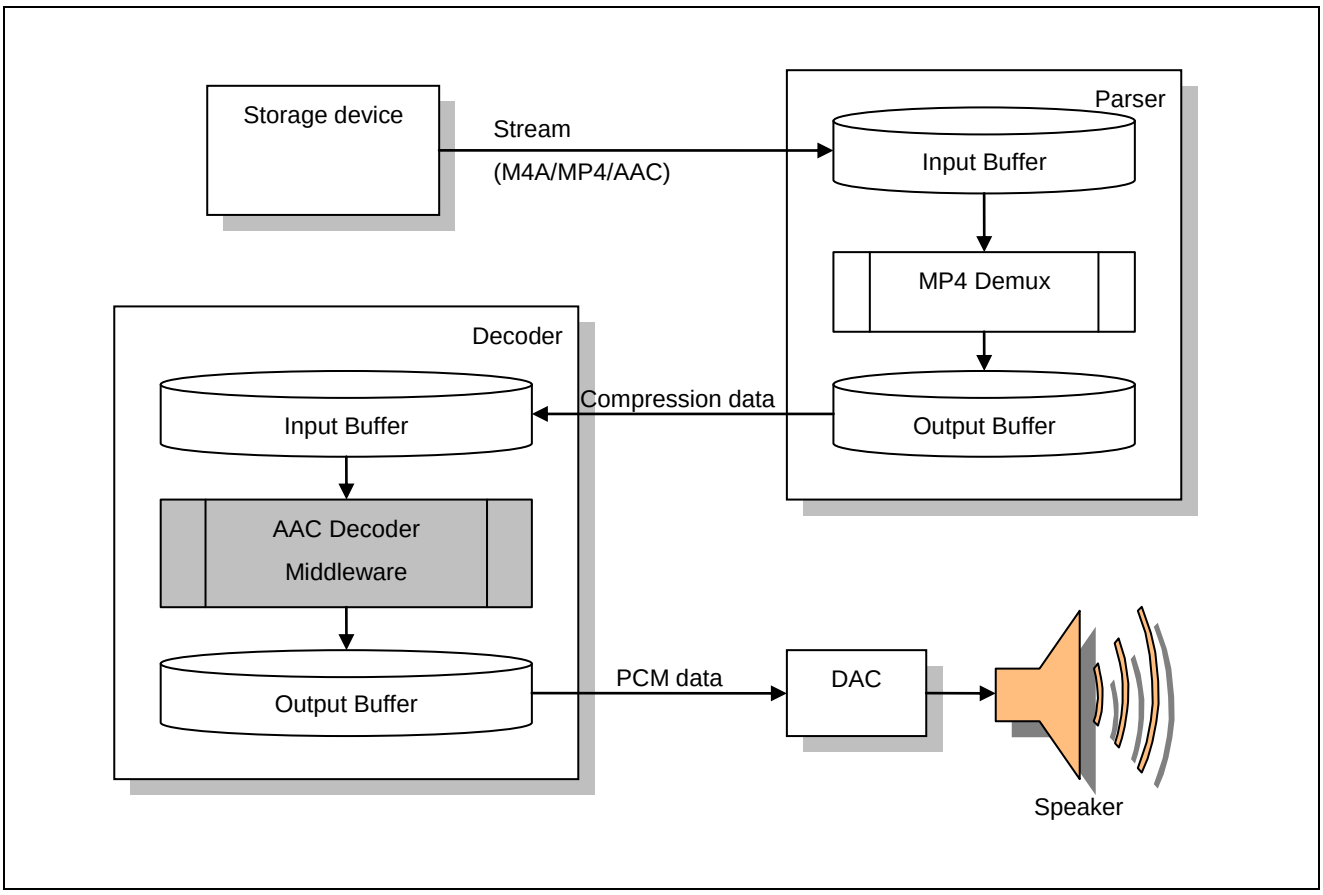

図1.1 デコード・システム構成例

1. M4A/MP4/AAC
   音楽配信サービスなどさまざまなサービスに採用されているデータです。

2. Parser
   M4A/MP4/AACファイルのデータの中から圧縮データを抜き出します。**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
   1フレームに切り出されたAACのビットストリーム・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16/32ビット・リニアPCMデータです。

6. DAC
   16/32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
aacd_GetMemorySize	必要メモリ・サイズ計算処理
aacd_Init	AACデコーダ初期化
aacd_Decode	AACフレーム・デコード処理
aacd_GetErrorFactor	エラー要因情報取得
aacd_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
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
エレメント情報構造体	PCE のエレメント情報を格納する構造体です。	I

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
AACD_RESULT_OK	0x00000000	処理結果が正常です。
AACD_RESULT_NG	0x00000001	処理結果が異常です。
AACD_RESULT_WARNING	0x00000002	処理継続可能な異常が発生しました。
AACD_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	aacd_XXXX
構造体名	aacd_XXXX
関数の返却値	AACD_RESULT_XXXX 【注】XXXXは全て大文字
エラー要因名	AACD_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	AACD_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。
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。

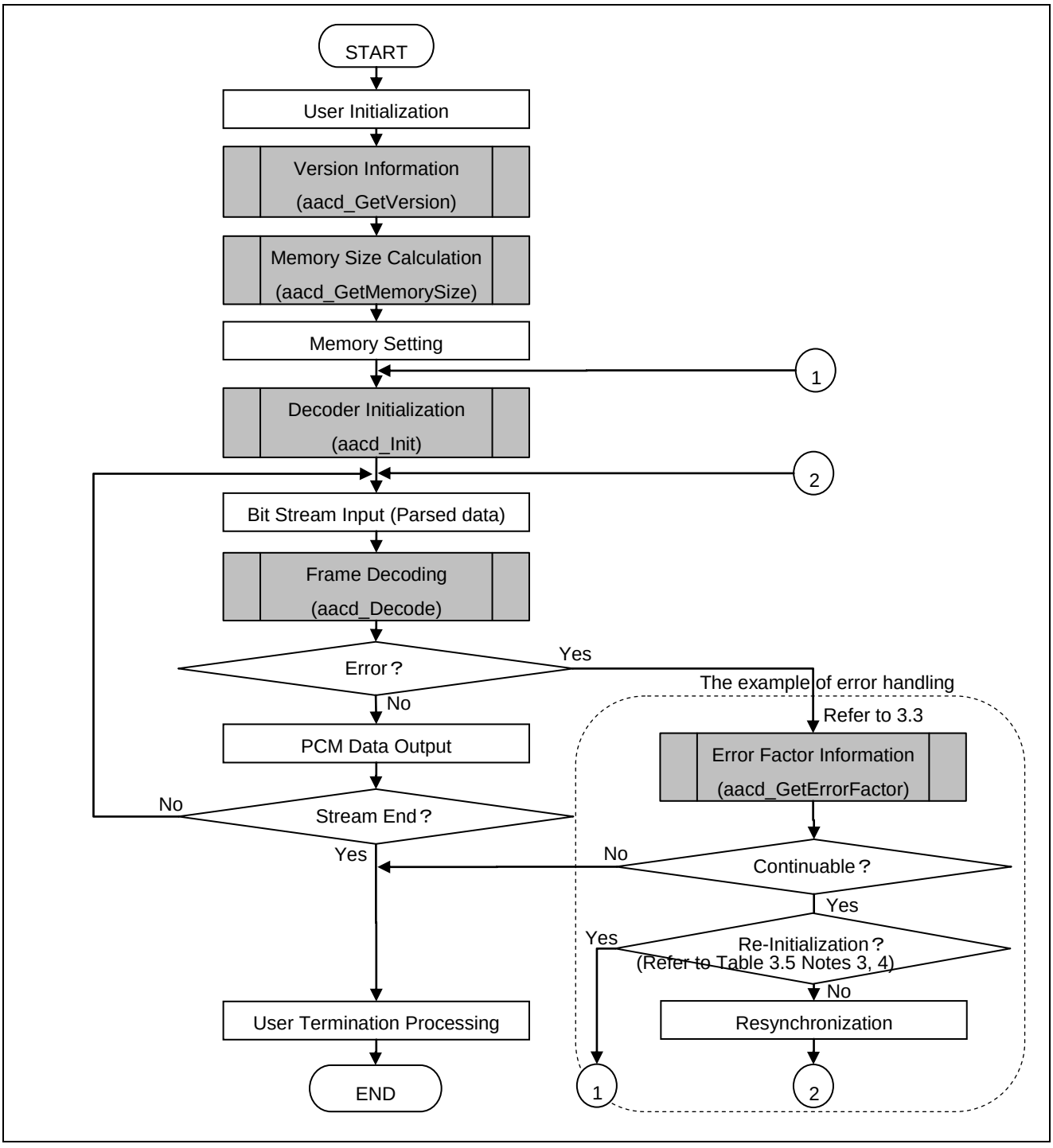

図2.1 アプリケーション処理フロー例

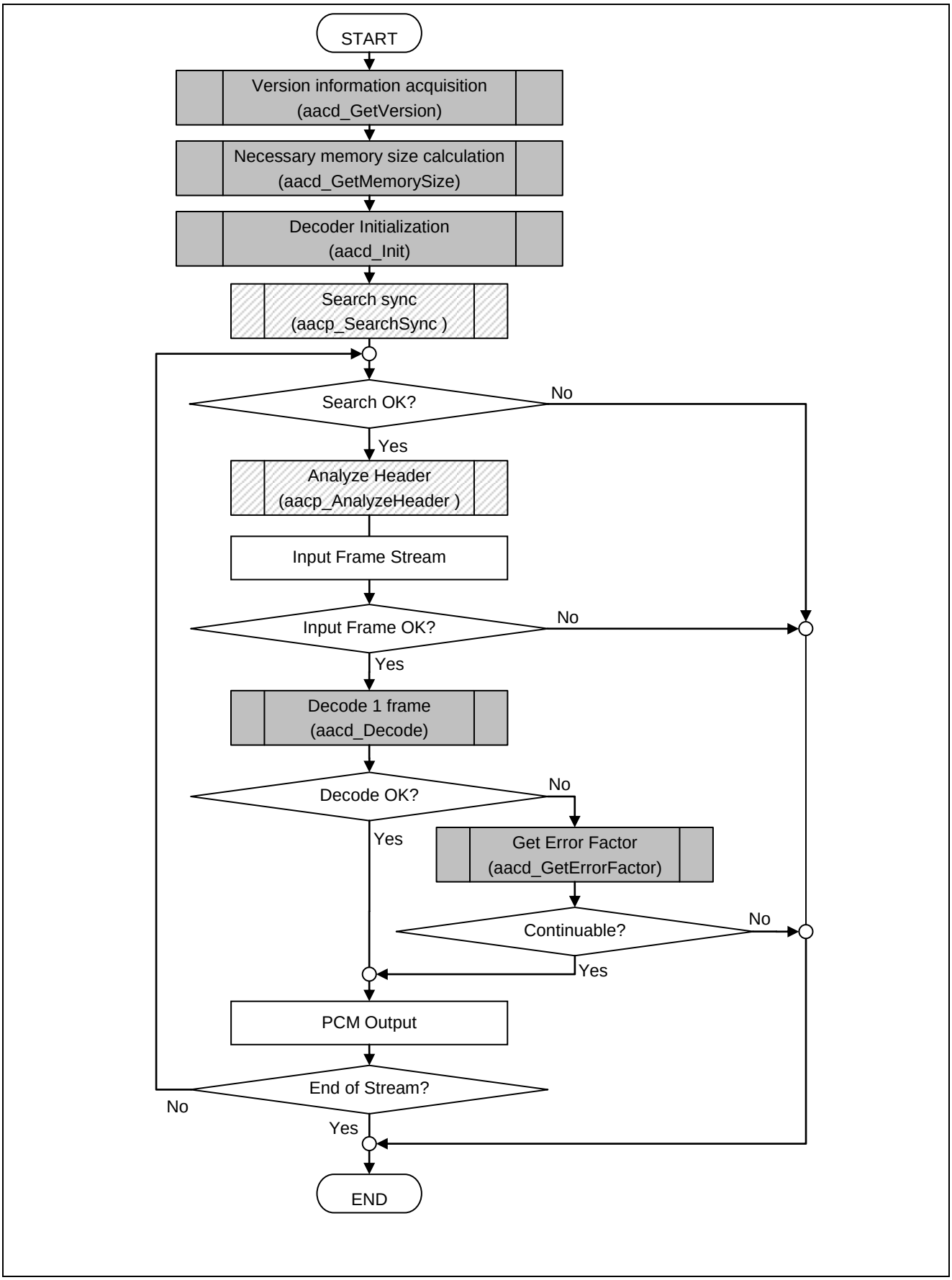

図2.2 サンプルプログラムのフローチャート

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】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 aacd_GetMemorySize 関数

概要	必要メモリ・サイズ計算処理		
構文	ACMW_INT32 aacd_GetMemorySize(const aacd_getMemorySizeConfigInfo * const pGetMemorySizeConfigInfo, aacd_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
aacd_getMemorySizeConfigInfo *pGetMemorySizeConfigInfo		I	メモリ・サイズ取得設定情報構造体
aacd_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3.3を参照
説明	aacd_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 aacd_Init 関数

概要	AACデコーダ初期化処理		
構文	ACMW_INT32 aacd_Init(const aacd_workMemoryInfo * const pWorkMemInfo, const aacd_initConfigInfo * const pInitConfigInfo);		
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
aacd_workMemoryInfo *pWorkMemInfo		I	ワーク・メモリ情報構造体
aacd_initConfigInfo *pInitConfigInfo		I	初期化設定情報構造体
戻り値	ACMW_INT32 errorCode		エラー・コード 詳細は、表3.3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 aacd_Decode 関数

概要	AACフレーム・デコード処理		
構文	ACMW_INT32 aacd_Decode(const aacd_workMemoryInfo * const pWorkMemInfo, const aacd_decConfigInfo * const pDecConfigInfo, const aacd_ioBufferConfigInfo * const pBuffConfigInfo, aacd_decStatusInfo * const pDecStatusInfo, aacd_ioBufferStatusInfo * const pBuffStatusInfo);		
機能	AAC LCフレーム・データをデコードします。		
引数		I/O	意味
aacd_workMemoryInfo *pWorkMemInfo		I	ワーク・メモリ情報構造体
aacd_decConfigInfo *pDecConfigInfo		I	デコード設定情報構造体
aacd_ioBufferConfigInfo *pBuffConfigInfo		I	バッファ・メモリ設定情報構造体
aacd_decStatusInfo *pDecStatusInfo		O	デコード結果情報構造体
aacd_ioBufferStatusInfo *pBuffStatusInfo		O	バッファ・メモリ結果情報構造体
戻り値	ACMW_INT32 errorCode		エラー・コード 詳細は、表3.3を参照
説明	1フレーム分のストリーム・データ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。		

3.1.4 aacd_GetErrorFactor 関数

概要	エラー要因情報取得処理		
構文	ACMW_UINT32 aacd_GetErrorFactor(const aacd_workMemoryInfo * const pWorkMemInfo);		
機能	直前に発生した、aacd_Init、aacd_Decode関数のエラー要因を返却します。		
引数		I/O	意味
aacd_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3.5を参照
説明	直前に発生した、aacd_Init、aacd_Decode関数のエラー要因を取得します。 aacd_Init、aacd_Decode関数以外のエラー、およびaacd_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aacd_Init、aacd_Decode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。		

3.1.5 aacd_GetVersion 関数

概要	バージョン情報取得処理		
構文	ACMW_UINT32 aacd_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】 aacd_getMemorySizeConfigInfo

【機能】 aacd_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_BOOL    bSbrDisableFlag;
    ACMW_BOOL    bPsDisableFlag;
    ACMW_BOOL    bDrcEnableFlag;
}aacd_getMemorySizeConfigInfo;
```

【メンバの説明】

メンバ変数名	内容	
bSbrDisableFlag	SBR強制OFFフラグ*1	
	0	SBR機能ON
	other	SBR機能強制OFF
bPsDisableFlag	PS強制OFFフラグ*1	
	0	PS機能ON
	other	PS機能強制OFF
bDrcEnableFlag	DRC適用フラグ	
	0	DRC OFF
	other	DRC ON

【備考】 領域の確保はユーザが行ってください。ユーザは、aacd_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。
*1：本ミドルウェアでは無効なパラメータです。(HE-AACデコード非対応のため)

3.2.2 メモリ・サイズ取得結果情報構造体

【構造体名】 aacd_getMemorySizeStatusInfo

【機能】 必要メモリ・サイズ計算処理(aacd_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;
}aacd_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]
nOutputBufferSize	出力バッファ領域の必要メモリ・サイズ[byte] *1
nStackSize	ミドルウェア・スタック領域の必要メモリ・サイズ[byte]

【備考】 領域の確保はユーザが行ってください。ユーザは、aacd_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：出力バッファ領域のメモリ・サイズは、1チャネル分のサイズを返します。また、初期化設定情報構造体のbOutBitsPerSample（3.2.4参照）に1を設定する場合、実際に必要な出力バッファ領域のメモリ・サイズは、返却値の2倍となります。

3.2.3 ワーク・メモリ情報構造体

【構造体名】 aacd_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;
}aacd_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

【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(aacd_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 】 aacd_initConfigInfo

【 機 能 】 初期化処理(aacd_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_BOOL   bSbrDisableFlag;
    ACMW_BOOL   bPsDisableFlag;
    ACMW_BOOL   bDownSampleSBRFlag;
    ACMW_BOOL   bDrcEnableFlag;
    ACMW_UINT16 nTargetRefLevel;
    ACMW_UINT16 nDefaultProgRefLevel;
    ACMW_UINT32 nCompress;
    ACMW_UINT32 nBoost;
    ACMW_BOOL   bOutputChMapType;
    ACMW_BOOL   bDisablePCEFlag;
    ACMW_BOOL   bOutBitsPerSample;
    ACMW_UINT16 nFormatType;
}aacd_initConfigInfo;
```

【メンバの説明】

メンバ変数名	内容	
bSbrDisableFlag	SBR強制OFFフラグ[*1]	
	0	SBR機能ON
	other	SBR機能強制OFF
bPsDisableFlag	PS強制OFFフラグ[*1]	
	0	PS機能ON
	other	PS機能強制OFF
bDownSampleSBRFlag	downsample SBRフラグ[*1]	
	0	通常SBRモード
	other	downsample SBRモード
bDrcEnableFlag	DRC適用フラグ	
	0	DRC OFF
	other	DRC ON
nTargetRefLevel	target reference level	
	0～127	0(dB)～31.75(dB)に対応します (1step 0.25dB)
	other	内部で127にクリッピングします
nDefaultProgRefLevel	program reference level	
	0～127	0(dB)～31.75(dB)に対応します (1step 0.25dB)
	other	内部で127にクリッピングします
nCompress	compress係数	
	0～0x7fffff	0.0～1.0に対応します
	other	内部で0x7fffffにクリッピングします
nBoost	boost係数	
	0～0x7fffff	0.0～1.0に対応します
	other	内部で0x7fffffにクリッピングします
bOutputChMapType	出力PCMのチャネルマッピング形式設定[*2]	
	0	デコード結果をそのまま出力します
	other	デコード結果を5.1chにマッピングして出力します L, R, C, LFE, Ls, Rsの順で出力します
bDisablePCEFlag	PCE読み飛ばしフラグ[*3]	
	0	PCEがあった場合は使用する
	other	PCEを読み飛ばす
bOutBitsPerSample	出力PCMの1サンプルあたりのビット数	
	0	16ビットPCM出力
	other	32ビットPCM出力
nFormatType	入力ビットストリームフォーマット	
	0x0000	ADTS
	0x0001	RAW DATA
	other	Reserved [*4]

【　備　考　】　領域の確保はユーザが行ってください。ユーザは、aacd_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　】　*1：本ミドルウェアでは無効なパラメータです。(HE-AACデコード非対応のため)
*2：本ミドルウェアでは無効なパラメータです。(6chデコード非対応のため)
*3：入力ストリームのRAWデータ部分に含まれるPCEにのみ有効です。
デコード設定情報構造体でPCEを設定した場合は、本フラグとは関係なく設定したPCEが有効となります。
*4：本ミドルウェアでは、RAW DATAが設定されたものとして動作します。

3.2.5 デコード設定情報構造体

【構造体名】 aacd_decConfigInfo

【機能】 aacd_Decode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_UINT32             nExtSamplingRate;
    ACMW_BOOL               bSetPceFlag;
    ACMW_aacd_elementInfo   sFrontElement;
    ACMW_aacd_elementInfo   sSideElement;
    ACMW_aacd_elementInfo   sBackElement;
    ACMW_aacd_elementInfo   sLfeElement;
}aacd_decConfigInfo;
```

【メンバの説明】

メンバ変数名	内容	
nExtSamplingRate	サンプリング周波数[Hz] *1	
bSetPceFlag	PCEデータ設定フラグ*2	
	0x0000	PCEデータを設定しない
	other	PCEデータを設定する
sFrontElement	フロントチャネル エレメント情報構造体（3.2.9参照）*3	
sSideElement	サイドチャネル エレメント情報構造体（3.2.9参照）*4	
sBackElement	バックチャネル エレメント情報構造体（3.2.9参照）*4	
sLfeElement	LFEチャネル エレメント情報構造体（3.2.9参照）*4	

【備考】 領域の確保はユーザが行ってください。ユーザは、aacd_Decode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】 *1：表3.17の周波数の値(Hz)を設定してください。
ただし、それ以外の値が設定された場合については、
ISO/IEC 14496-3： 4.5.1.1 Table4.82に従い、補正した周波数として動作します。
0指定時は、エラー（AACD_ERR_PARAMETER）となります。
*2：一度PCEデータを設定すると、再びPCEデータを設定するか、ストリーム内のRAWデータ部分のPCEを見つけるまで、内部でPCEデータの値を保持します。
*3：各エレメント情報構造体は、PCEデータ設定フラグを設定する時にのみ有効になります。
*4：本ミドルウェアでは無効なパラメータです。(6chデコード非対応のため)

3.2.6 デコード結果情報構造体

【 構 造 体 名 】 aacd_decStatusInfo

【 機 能 】 デコード処理(aacd_Decode関数)を行った結果を格納します。

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

【メンバの説明】

メンバ変数名	内容	
nSamplingRate	サンプリング周波数（AAC LC）　[Hz]	
nOutSamplingRate	出力サンプリング周波数(最終出力) [Hz]	
nOutSamplesPerFrame	1チャネル、1フレームあたりのサンプル数(最終出力)	
nChannelNum	チャネル数(AAC LC)	
nOutChannelNum	チャネル数(最終出力)	
nChannelConfig	チャネル構成情報(表3.2参照)*1	
nOutChannelConfig	出力チャネル構成情報*2 出力したチャネル位置に対応するビットが1に設定されます	
bDualChannelMode	ストリームがdualMono形式かを示す	
	0	dualMonoではない
	1	dualMono形式のストリーム
nDualChannelTag	dualMono時のelement_instance_tag情報 （図3.1参照）	
bCrcEnable	ストリーム上のCRC有無を示す	
	0	CRCなし
	1	CRCあり
bSbrFound	SBRデコードの有無を示す*3	
	0	SBRなし
	1	SBRあり
bPsFound	PSデコードの有無を示す*3	
	0	PSなし
	1	PSあり
bPceFound	ストリーム上のPCE有無を示す	
	0	PCEなし
	1	PCEあり
bMpegDmxCoefPresent	matrix_mixdown_idx_presentパラメータ*4	
nMpegDmxCoef	matrix_mixdown_idxパラメータ*4	
bPseudoSurroundEnable	pseudo_surround_enableパラメータ*4	
nDseNum	データ・ストリーム・バッファに格納したDSEの数	
aDseTag[16]	各データ・ストリーム・バッファに格納したDSEの element_instance_tagの値	

【　備　考　】　領域の確保はユーザが行ってください。ユーザは、aacd_Decode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：ADTSヘッダ情報であるchannel_configurationの値を出力します。初期化設定情報構造体のnFormatTypeにRAW DATAを指定した場合、値は0固定です。
*2：本ミドルウェアでは無効なパラメータです。常に0を返します。(6chデコード非対応のため)
*3：本ミドルウェアでは無効なパラメータです。常に0を返します。(HE-AACデコード非対応のため)
*4：ISO/IEC 14496-3：　4.4.1.1 Program config element参照
bPceFoundが1の時に有効となり、以降は最後に取得したPCE内の情報を保持します。

表3.2 チャネル構成情報

nChannelConfig	チャネル構成
0x0000	(PCEで定義)
0x0001	1
0x0002	2

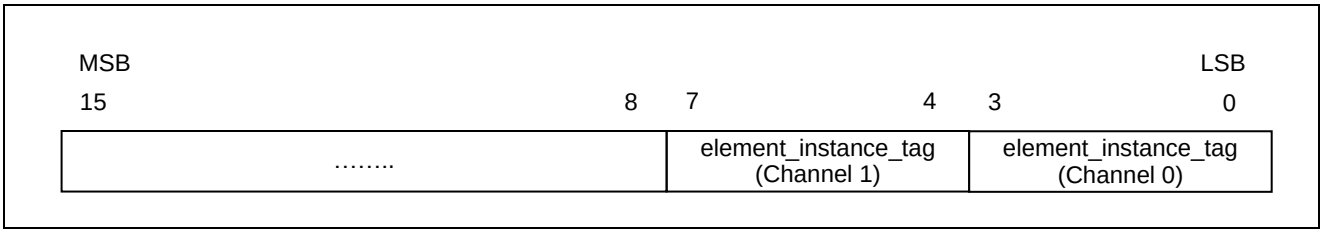

図3.1 dualMono 時の element_instance_tag 情報

【注】Channel 0、Channel 1は、それぞれ、バッファ・メモリ結果情報構造体のnOutBuffUsedDataSize [0]、[1]に対応しています。

3.2.7 バッファ・メモリ設定情報構造体

【構造体名】 aacd_ioBufferConfigInfo

【機能】 aacd_Decode関数で使用する入力バッファのパラメータを指定します。

【プロトタイプ】
```
typedef struct {
    ACMW_UINT8 *    pInBuffStart;
    ACMW_UINT32     nInBuffOffsetBits;
    ACMW_UINT32     nInBuffSetDataSize;
    void **         pOutBuffStart;
    ACMW_UINT32     nOutBuffSize;
    ACMW_UINT8      nDseBuffNum;
    ACMW_UINT8 **   pDseBuffStart;
    ACMW_UINT16     nDseBuffSize;
}aacd_ioBufferConfigInfo;
```

【メンバの説明】

メンバ変数名	内容
pInBuffStart	入力データの開始アドレス
nInBuffOffsetBits	入力開始ビットのオフセット (0～7)*1
nInBuffSetDataSize	入力データ・サイズ [byte]
pOutBuffStart	チャネルごとの出力データの開始アドレス
nOutBuffSize	各出力バッファのサイズ [byte]*2
nDseBuffNum	データ・ストリーム・バッファの数 (0～16)*3
pDseBuffStart	各データ・ストリーム・バッファの開始アドレス
nDseBuffSize	各データ・ストリーム・バッファのサイズ [byte]*4

【備考】 領域の確保はユーザが行ってください。ユーザは、aacd_Decode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。
入力バッファについては、3.4.5項を、出力バッファについては、3.4.6項を参照してください。

【備考2】 *1：範囲外の値を指定された場合は0とみなします。
*2：出力バッファのサイズは各チャネルで共通です。
*3：データ・ストリーム・バッファを使用しない場合は0を指定してください。範囲外の値を指定された場合は16とみなします。
*4：データ・ストリーム・バッファのサイズは共通です。

3.2.8 バッファ・メモリ結果情報構造体

【構造体名】 aacd_ioBufferStatusInfo

【機能】 aacd_Decode関数で使用するバッファ・メモリの情報の処理結果を格納します。

【プロトタイプ】
```
typedef struct {
    ACMW_UINT8 *        pInBuffLast;
    ACMW_UINT32         nInBuffUsedDataSize;
    void **             pOutBuffLast;
    ACMW_UINT32         nOutBuffUsedDataSize;
    ACMW_UINT16         aDseDataSize[16];
}aacd_ioBufferStatusInfo;
```

【メンバの説明】

メンバ変数名	内容
pInBuffLast	入力バッファの読み込み終了アドレス*1
nInBuffUsedDataSize	入力バッファの消費データ・サイズ [byte] *2
pOutBuffLast	チャネルごとの出力データの書き出し終了アドレス
nOutBuffUsedDataSize	各チャネルの出力バッファの消費データ・サイズ [byte] *3
aDseDataSize	各データ・ストリーム・バッファに格納したデータ・サイズ [byte]

【備考】 領域の確保はユーザが行ってください。ユーザは、aacd_Decode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：終了アドレスは次の開始アドレスを指し示しています。
*2：バッファ・メモリ設定情報構造体のnInBuffOffsetBitsが0以外の場合、入力バッファの消費データ・サイズは、実際の消費データ・ビット数にnInBuffOffsetBitsの値を足した値となります。（単位はバイト）
*3：出力バッファの消費データ・サイズは各チャネル共通です。

3.2.9 エレメント情報構造体

【構造体名】 aacd_elementInfo

【機能】 aacd_Decode関数で使用するエレメント情報を設定します。

【プロトタイプ】
```
typedef struct {
    ACMW_UINT32     nElement;
    ACMW_BOOL       aEleIsCpe[16];
    ACMW_UINT32     aEleTag[16];
}aacd_elementInfo;
```

【メンバの説明】

メンバ変数名	内容
nElement	num_front_channel_elements[*1]
aEleIsCpe[16]	front_element_is_cpe[*1]
aEleTag[16]	front_element_tag_select[*1]

【備考】 3.2.6項のデコード結果情報構造体のメンバとなっているため、デコード結果情報構造体を確保すると、本構造体領域も確保されます。ユーザは、aacd_Decode関数を呼び出すより前に、デコード結果情報構造体の確保を行ってください。

【備考2】 *1：ISO/IEC 14496-3： 4.4.1.1 Program config element参照

3.3 エラー処理

本ミドルウェアの関数は、表3.3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、aacd_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3.3 エラー・コード

エラー・コード (32bit)	値	説明	再初期化
① AACD_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
② AACD_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
③ AACD_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(aacd_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
④ AACD_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。aacd_GetErrorFactor 関数によるエラー要因取得はできません。	必要
⑤その他	上記以外	予約	ー

表3.4 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	aacd_GetMemorySize	aacd_Init	aacd_Decode	aacd_GetErrorFactor*1	aacd_GetVersion*2
AACD_RESULT_OK	○	○	○	ー	ー
AACD_RESULT_NG	ー	○	○	ー	ー
AACD_RESULT_WARNING	ー	ー	○	ー	ー
AACD_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

*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。AACD_RESULT_FATAL が発生した場合を除き、aacd_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 3.5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 3.6に示します。

表3.5 エラー要因一覧

errorFactor(32bit)	値	説明	表3.3	PCM
AACD_ERR_NONE	0x00000000	正常終了した。エラー要因は存在しない。	①	正常*1
AACD_ERR_POINTER	0x00000010	ポインタ値が不正だった。	②	なし
AACD_ERR_PARAMETER	0x00000020	パラメータが不正だった。	②	なし
AACD_ERR_SEQUENCE	0x00000040	本ライブラリ関数の実行順序が不正だった。*3	②	なし
AACD_ERR_INPUT_DATA_SIZE	0x00000100	入力バッファのエンプティを検出した。	②	なし
AACD_ERR_STREAM	0x00001000	入力ストリームのRAWデータ部分に異常を検出した。	②	なし
AACD_ERR_HEADER	0x00010000	ADTSヘッダのsyncwordが見つからなかった。もしくは不正だった。	②	なし
AACD_ERR_CHANGED_FS	0x00020000	前フレームとサンプリング周波数が異なる。*4	②	なし
AACD_ERR_CHANGED_CH	0x00040000	前フレームとチャネル数が異なる。*4	②	なし
AACD_ERR_PROGRAM_CONFIG	0x00080000	ProgramConfigのエラーを検出した。	②	なし
AACD_ERR_1ST2ND_FRAME	0x00100000	最初の2フレームを出力している。	③	あり
AACD_ERR_CRC	0x00200000	CRCエラーが発生した。	②	なし
AACD_ERR_DECODE	0x01000000	デコード処理中にエラーが発生した。	③	あり*2
その他	上記以外	予約	－	－

【注】 *1 データの破損部位によってはエラーの判定が不可能なため、AACD_ERR_NONEとみなし、デコード処理を継続し出力します。

*2 データの破損部位によってはデコード途中の前半でありPCMが作成されない時があります。

*3 実行順序が不正のため再初期化が必要です。

*4 リカバリ処理として、aacd_Init関数で初期化処理を行うと、同じ入力ストリーム開始位置からデコード処理が可能になります。

表3.6　　ライブラリ関数で設定されるエラー要因

関数 / エラー要因	aacd_GetMemorySize	aacd_Init	aacd_Decode	aacd_GetErrorFactor*1	aacd_GetVersion
AACD_ERR_NONE	ー	○	○	ー	ー
AACD_ERR_POINTER	ー	○	○	ー	ー
AACD_ERR_PARAMETER	ー	ー	○	ー	ー
AACD_ERR_SEQUENCE	ー	ー	○	○	ー
AACD_ERR_INPUT_DATA_SIZE	ー	ー	○	ー	ー
AACD_ERR_STREAM	ー	ー	○	ー	ー
AACD_ERR_HEADER	ー	ー	○	ー	ー
AACD_ERR_CHANGED_FS	ー	ー	○	ー	ー
AACD_ERR_CHANGED_CH	ー	ー	○	ー	ー
AACD_ERR_PROGRAM_CONFIG	ー	ー	○	ー	ー
AACD_ERR_1ST2ND_FRAME	ー	ー	○	ー	ー
AACD_ERR_CRC	ー	ー	○	ー	ー
AACD_ERR_DECODE	ー	ー	○	ー	ー

【注】○：設定される可能性がある、ー：設定されない

【注】*1　設定されたエラー要因を返します。
　　エラー要因を返せない場合はエラー・コードの AACD_RESULT_FATAL を返します。

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3.7 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、aacd_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3.8 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、aacd_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	4バイト境界に配置してください。

3.4.3 ミドルウェア・スタック領域

表3.9 ミドルウェア・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、aacd_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ミドルウェア・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3.10 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、ストリーム・データ（AAC 圧縮データ）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。 【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	1フレーム分のサイズを確保して下さい。 本ミドルウェアのaacd_GetMemorySize関数で得られるnInputBufferSize(メモリ･サイズ取得結果情報構造体のメンバ）は、推奨サイズです。
領域確保	ユーザが領域を確保します。 １フレームのデコード処理後に、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	制限はありません。

aacd_Decode 関数実行時に、バッファ・メモリ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 3.2に示します。

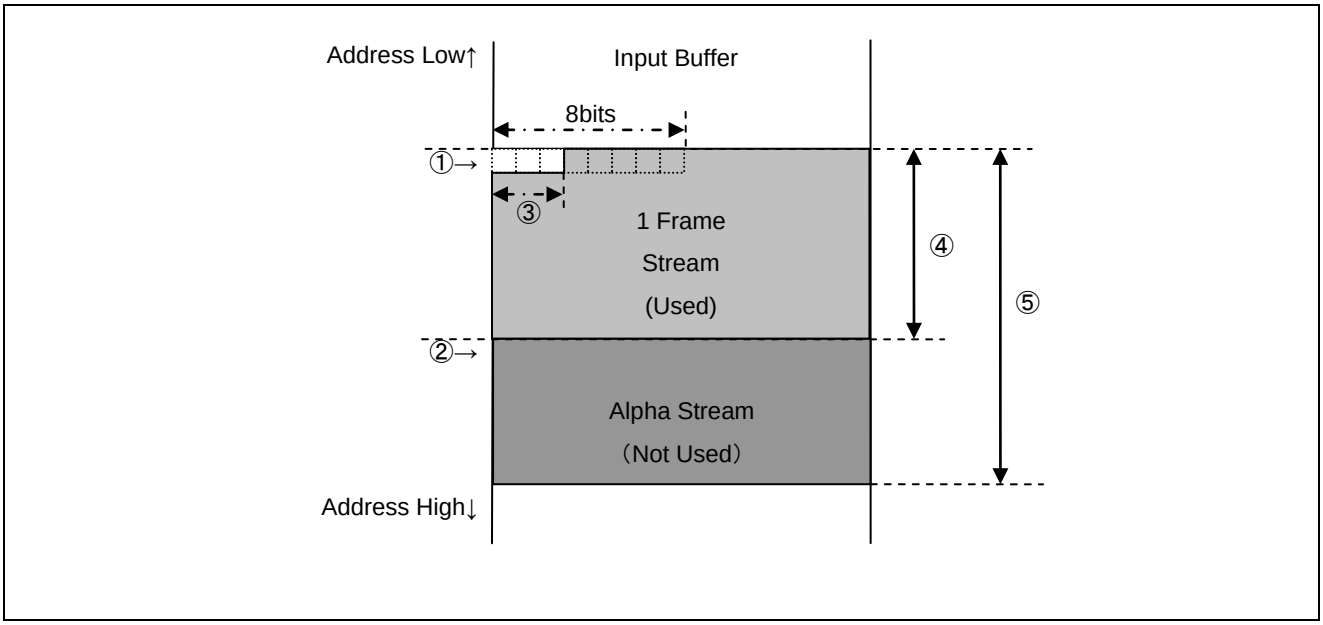

図3.2 入力バッファにおける各種構造体メンバの情報イメージ例

【注】入力バッファに1フレーム＋α分のデータを入れ、デコード処理を行った場合を示しています。

表3.11 入力バッファにおける各種構造体メンバの情報

①	pInBuffStart	(バッファ・メモリ設定情報構造体)	入力データ開始アドレス
②	pInBuffLast	(バッファ・メモリ結果情報構造体)	入力バッファ読み込み終了後アドレス
③	nInBuffOffsetBits	(バッファ・メモリ設定情報構造体)	入力開始ビットのオフセット
④	nInBuffUsedDataSize	(バッファ・メモリ結果情報構造体)	入力バッファ消費データ・サイズ
⑤	nInBuffSetDataSize	(バッファ・メモリ設定情報構造体):	入力データ・サイズ

【注】①～②はアドレスを示し、③～⑤はサイズを表しています。
ストリーム・データは、最小構成単位が8ビットになります。

また、ストリーム開始ビットがバイトアラインされていない場合、バッファ・メモリ設定情報構造体のnInBuffOffsetBits値でストリーム開始ビットを指定することができます。nInBuffOffsetBits=3の例を図3.3に示します。

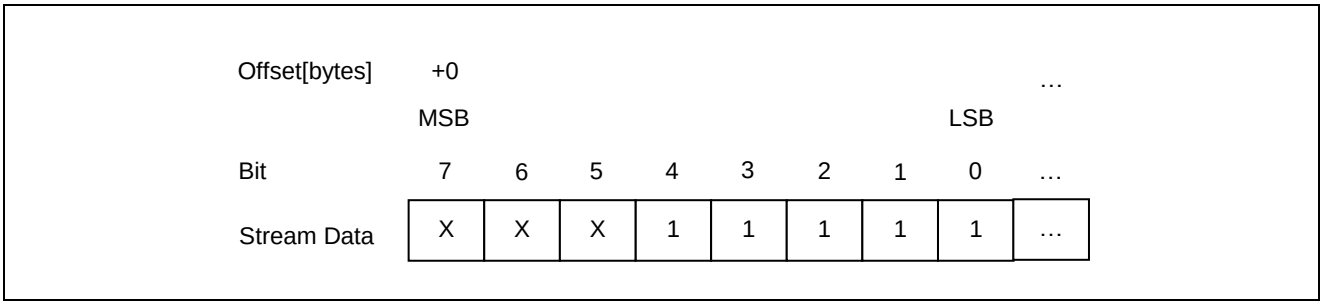

図3.3 nInBuffOffsetBits =3 の例

【注】“X”はストリーム・データではない任意の値

(1) 入力データ格納方式

入力データの格納方式を図 3.4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。

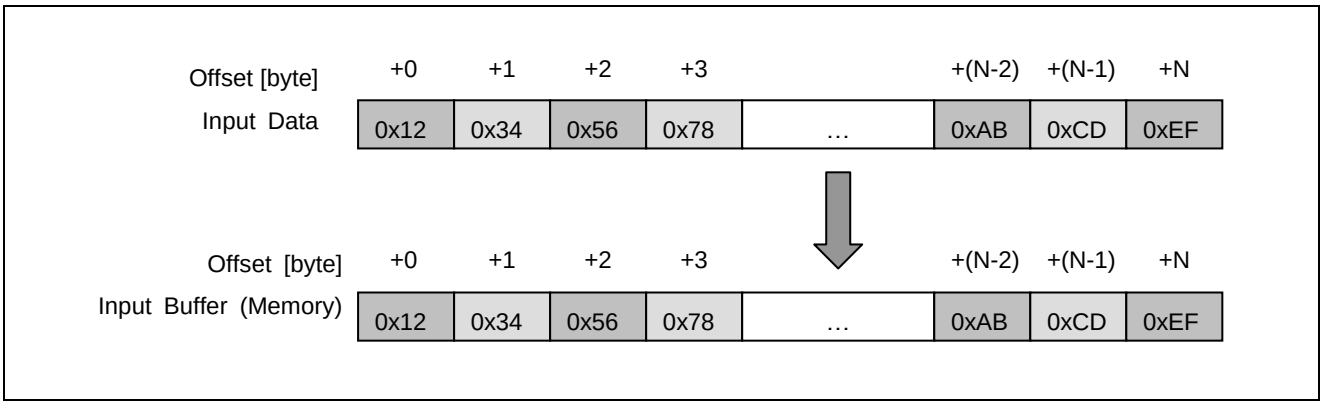

図3.4 入力データの格納方式

3.4.6 出力バッファ

表3.12 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、16/32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	必ずaacd_GetMemorySize関数で取得した必要メモリ・サイズ以上を確保してください。 【注】aacd_GetMemorySize関数で取得できるサイズは、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。 １フレームのデコード処理後に、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	16ビットPCMデータとして出力する場合は、2バイト境界に配置してください。 32ビットPCMデータとして出力する場合は、4バイト境界に配置してください。

aacd_Decode 関数実行時に、バッファ・メモリ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。出力バッファと各構造体メンバの関係を図 3.5に示します。

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、デコード結果情報構造体の nOutSamplesPerFrame を参照してください。

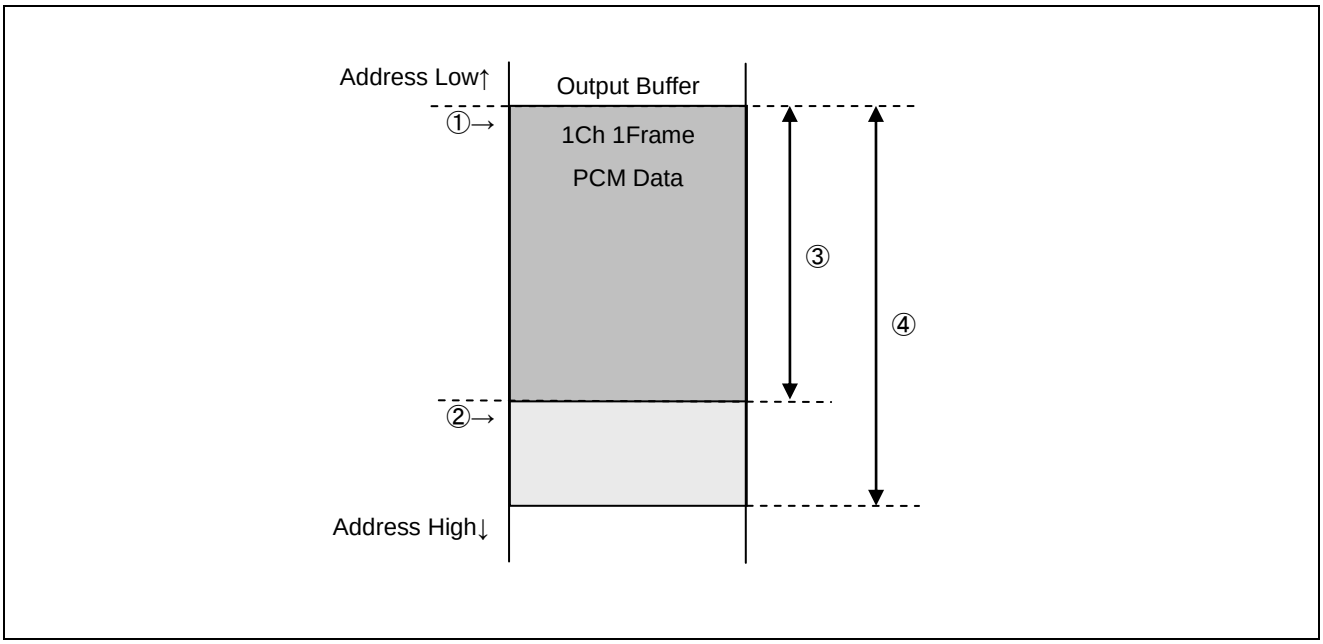

図3.5 出力バッファにおける各種構造体メンバの情報イメージ例

【注】デコード処理で出力バッファに1フレーム分の出力結果を入れた場合を示しています。

表3.13 出力バッファにおける各種構造体メンバの情報

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

出力バッファ(メモリ)には、2 バイト(16 ビット)または、4 バイト(32 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 3.6、図 3.7参照)は、リトル・エンディアンになります。

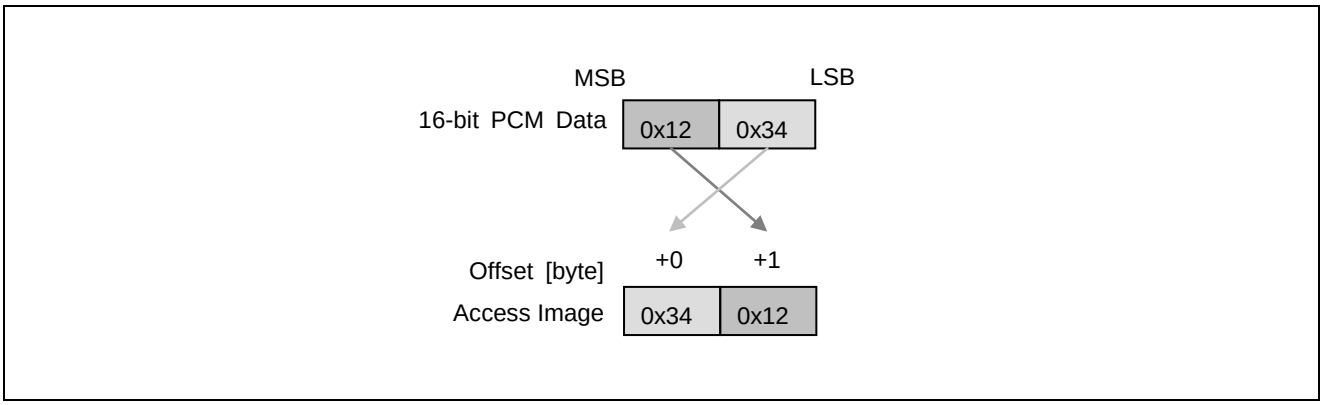

図3.6 PCM データ(リトル・エンディアン)アクセス(16 ビット)

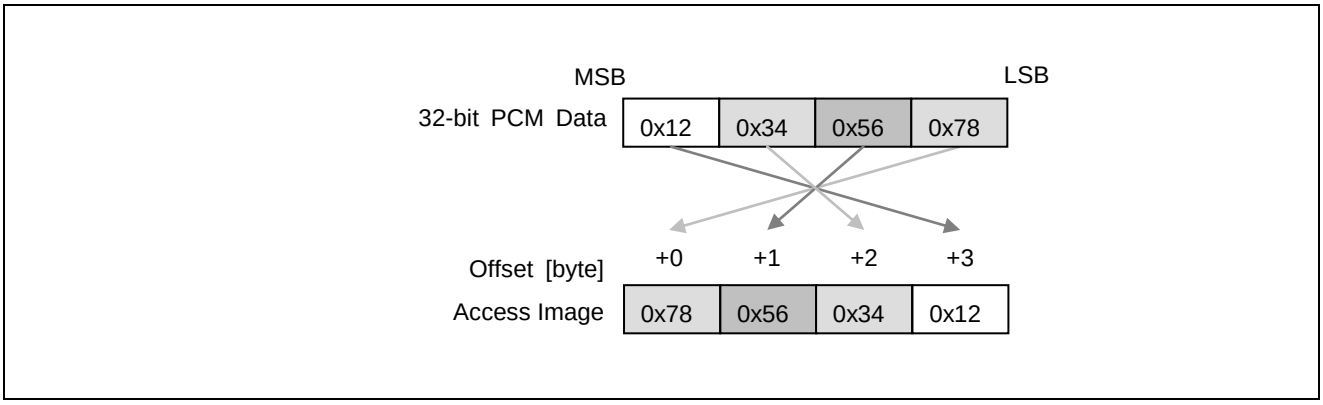

図3.7 PCM データ(リトル・エンディアン)アクセス(32 ビット)

図 3.8に PCM データのビット配置を示します。

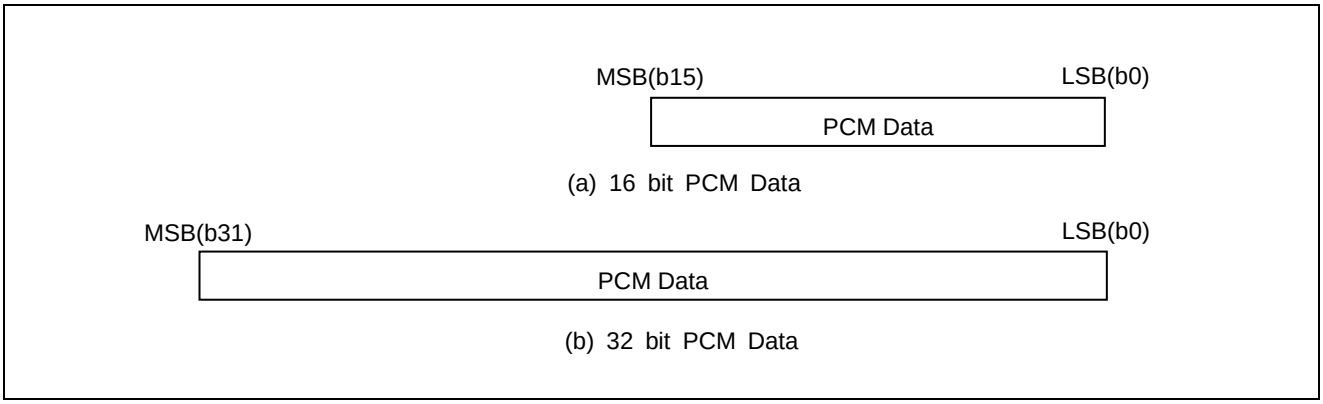

図3.8 PCM データのビット配置

3.4.7 データ・ストリーム・バッファ

データ・ストリーム・バッファは、DSE に含まれるバイト・データを格納する領域です。領域の確保は、ユーザが行ってください。DSE に含まれるバイト・データを取得する必要がなければ、確保する必要はありません。データ・ストリーム・バッファを確保しない場合、DSE に含まれるバイト・データは読み捨てられます。

DSE に含まれるバイト・データ最大サイズは、規格上 8160[byte](510[byte/instance] × 16[instance])です。ただし、確保したデータ・ストリーム・バッファ・サイズ以上のデータは読み捨てるため、必ずしも最大サイズを確保する必要はありません。ユーザが取得したい内容が十分に格納されるサイズの領域を確保してください。

データ・ストリーム・バッファと各構造体メンバの関係を図 3.9に示します。

図 3.9では、データ・ストリーム・バッファ 2 個の設定(バッファ・メモリ設定情報構造体の nDseBuffNum =2)で、デコード処理を行った場合のデータ・ストリーム・バッファ例を示しています。

最初にデコードされた DSE に含まれるバイト・データは、データ・ストリーム・バッファの先頭(①)から、データ・サイズ(③)分書き込まれます(バッファ 0)。同一フレームで異なる element_instance_tag を持つ DSE がデコードされた場合、この DSE に含まれるバイト・データは、データ・ストリーム・バッファの先頭(①)にデータ・ストリーム・バッファ・サイズ(②)を加算した位置から、データ・サイズ(④)分書き込まれます(バッファ 1)。DSE に含まれるバイト・データが、データ・サイズ(③)よりも大きい場合、先頭からデータ・サイズ(③)分までが書き込まれます。

各バッファに実際に書き込まれたデータ・サイズは、バッファ・メモリ結果情報構造体の aDseDataSize へ格納されます。格納した数と element_instance_tag の値は、それぞれデコード結果情報構造体の nDseNum と aDseTag が示しています。

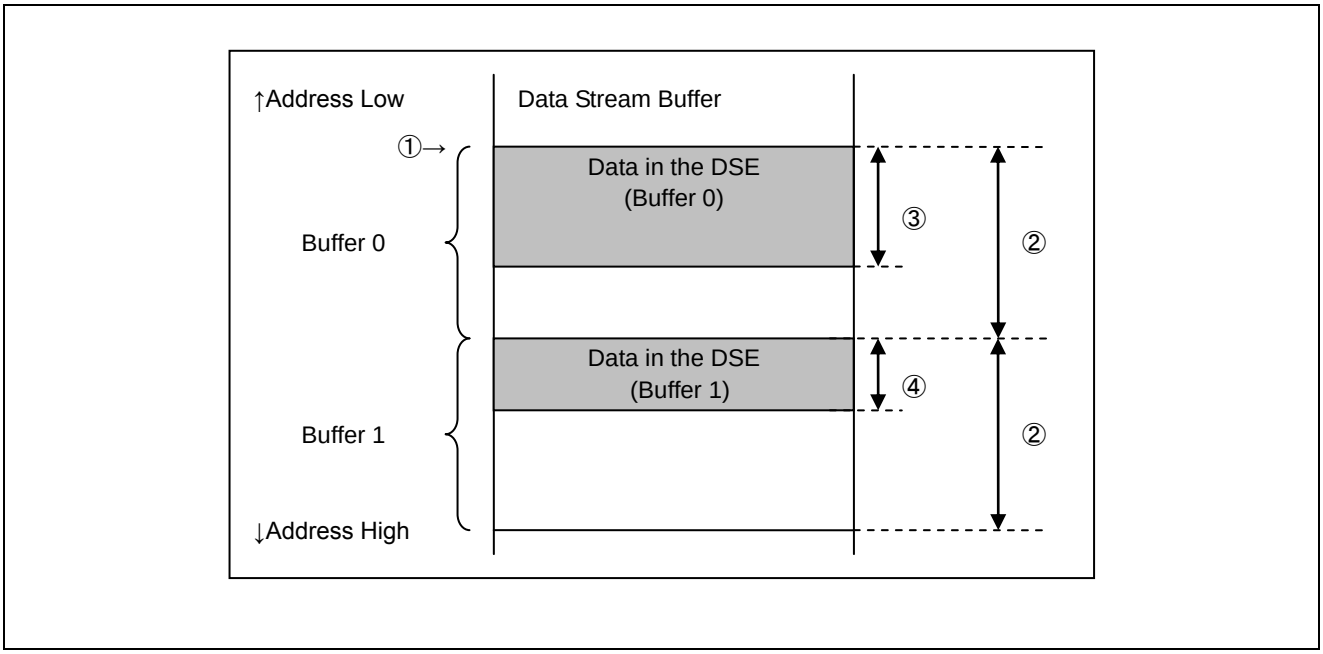

図3.9 データ・ストリーム・バッファ例

表3.14 データ・ストリーム・バッファにおける構造体メンバの情報

①	pDseBuffStart	(バッファ・メモリ設定情報構造体)	データ・ストリーム・バッファの開始アドレス
②	nDseBuffSize	(バッファ・メモリ設定情報構造体)	データ・ストリーム・バッファ・サイズ[byte]
③	aDseDataSize [0]	(バッファ・メモリ結果情報構造体)	データ・ストリーム・バッファに格納したデータ・サイズ (バッファ 0) [byte]
④	aDseDataSize [1]	(バッファ・メモリ結果情報構造体)	データ・ストリーム・バッファに格納したデータ・サイズ (バッファ 1) [byte]

3.5 入力データ

本ミドルウェアは、AAC LC 圧縮データを入力することが可能です。本節では、本ミドルウェアがサポートする ADTS 形式、RAW 形式を説明します。入力データの格納方式については3.4.5項を参照してください。

3.5.1 ADTS 形式

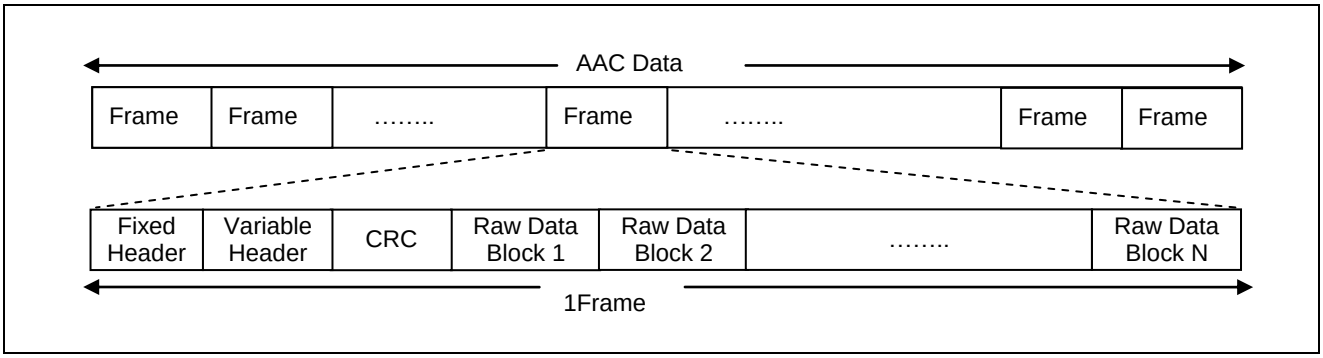

図3.10 圧縮データ・フォーマット

【Fixed Header】

任意の ADTS フレームにランダム・アクセスするためのヘッダで、各 ADTS フレームにおいて常に同じ値を設定します。syncword、サンプリング周波数、チャネル構成などの情報を含みます。

表3.15 ADTS 固定ヘッダ構造

	ビット	詳細
Syncword	12	同期ワード、'**1111 1111 1111**' のビット列
ID	1	MPEG の ID（0：MPEG-4 AAC、1：MPEG-2 AAC）
Layer	2	レイヤーの種類、'**00**'がセット
protection_absent	1	エラーチェックデータの有無（0：有り、1：無し）
Profile_ObjectType	2	ID に依存（表 3.16参照）
sampling_frequency_index	4	サンプリング周波数（表 3.17参照）
private_bit	1	私的使用の為のビット
channel_configuration	3	チャンネル配置（表 3.18参照）
original_copy	1	著作権保護（0：無し、1：有り）
home	1	オリジナルとコピーの区別（0：コピー、1：オリジナル）
Emphasis	2	初期規格のみ付加（本ミドルウェアでは未対応）

表3.16 Profile_ObjectType

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

【注】LC 以外のプロファイルは本ミドルウェアでは非対応です。

表3.17 sampling_frequency_index

サンプリング周波数（Hz）	値
96000	0000
88200	0001
64000	0010
48000	0011
44100	0100
32000	0101
24000	0110
22050	0111
16000	1000
12000	1001
11025	1010
8000	1011
7350	1100*1
（reserved）	1101*1
（reserved）	1110*1
（escape value）	1111*1

【注】*1：本ミドルウェアでは非対応です。

表3.18 channel_configuration

スピーカ数	値	audio syntactic elements, listed in order received	スピーカ位置
―	000	program_config_element	―
1	001	single_channel_element	center front speaker
2	010	channel_pair_element	left, right front speakers
3	011	single_channel_element channel_pair_element	center front speaker left, right front speakers
4	100	single_channel_element channel_pair_element single_channel_element	center front speaker left, right center front speakers, rear surround
5	101	single_channel_element channel_pair_element channel_pair_element	center front speaker left, right front speakers, left surround, right surround rear speakers
5+1	110	single_channel_element channel_pair_element channel_pair_element lfe_channel_element	center front speaker left, right front speakers, left surround, right surround rear speakers, front low frequency effects speaker
7+1	111	single_channel_element channel_pair_element channel_pair_element channel_pair_element lfe_channel_element	center front speaker left, right center front speakers, left, right outside front speakers, left surround, right surround rear speakers, front low frequency effects speaker

【Variable Header】

ADTS フレームごとに各情報の値を変えることができるヘッダです。ADTS フレームサイズや、ADTS フレーム内に存在する RAW データ・ブロック数などの情報を含みます。

表3.19 ADTS 可変ヘッダ構造

	ビット	詳細
copyright_identification_bit	1	72 ビット著作権識別フィールドの 1 ビット
copyright_identification_start	1	著作権ビットが最初の 1 ビットである事を示す
aac_frame_length	13	1 フレームの長さ（バイト）
adts_buffer_fullness	11	バッファ利用可能ビット数（0x7FF：VBR→適用不可）
no_raw_data_blocks_in_frame	2	フレーム内の RAW Data Block の個数 – 1

【CRC Check】

CRC エラー検出を行うためのデータです。ADTS Fixed Header 内のパラメータ’protection_absent’が 1 の場合は、本データは存在しません。

【RAW Data Block】 （3.5.2項参照）

符号化されたオーディオ・データと、これに関連する情報を含みます。1 つの ADTS フレーム内に、最大 4 個の RAW データ・ブロックが存在可能です。本ミドルウェアは、ADTS フレーム内に 2 個以上の RAW データがある場合は対応していません。

3.5.2 RAW 形式

RAW 形式は、複数の RAW データ・ブロックで構成され、ヘッダが存在しないものです。

各々のブロックの中身は ID の 3 ビットにより構成が分かれます。RAW データ・ブロックは関連情報他各種データ、符号化されたオーディオ・データを含んでいます。各 RAW データ・ブロックの終端には、terminate ID が必要です。

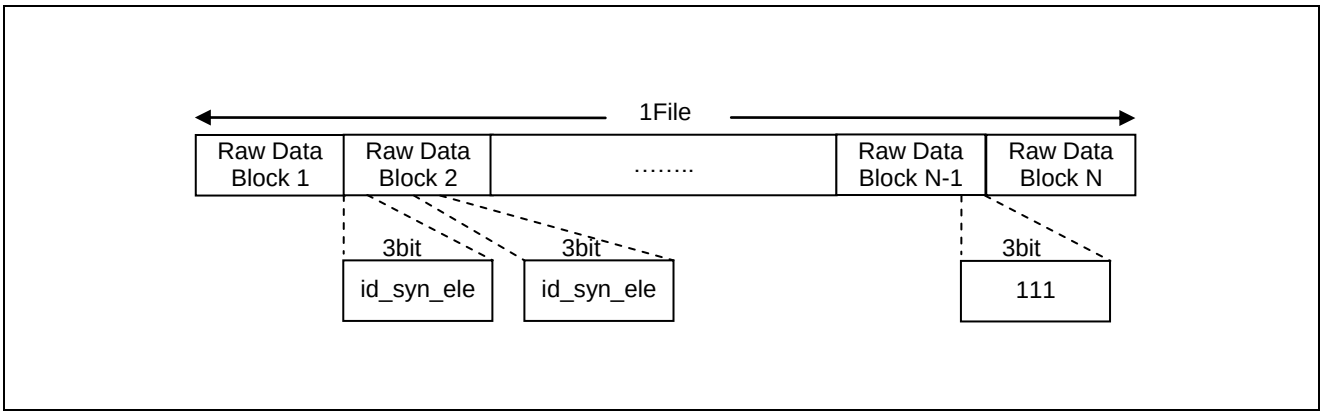

図3.11 RAW 形式

表3.20 id_syn_ele 一覧

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

3.6 出力データ

16/32ビット・リニアPCMデータを出力します。サンプル数は、1チャネル、1フレームあたり1024サンプルです。

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) aacd_GetMemorySize 関数

任意のタイミングで実行可能です。ただし、aacd_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) aacd_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) aacd_Decode 関数

1 フレーム分のストリーム・データをデコードするとき、この関数を実行してください。

(4) aacd_GetErrorFactor 関数

aacd_Init 関数実行後の任意のタイミングで実行可能です。

(5) aacd_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	AAC 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	2014.12.12	ー	初版発行

AAC Decode Middleware ユーザーズマニュアル

発行年月日　2014年12月12日 Rev.1.00

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/

AAC Decode Middleware

RENESAS

ルネサス エレクトロニクス株式会社