User's Manual

FLAC Decode Software

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、本ソフト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。本ソフト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、概要、ソフトウェア仕様、ライブラリ関数仕様、使用上の注意で構成されています。

本ソフトウェアは、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. 本ソフトウェアのご使用について

FLAC はフリーの規格のため、ライセンサーによる認証のような形での性能保証、知的財産権に関する保証や争議調停に関する契約、大幅な規格変更がないことの確約等は一切なく、本ソフトウェアの使用は自己責任となりますのでご注意ください。

3. 関連マニュアル

FLAC 対応規格

規格番号・タイトル	発行日
FLAC 1.3.0 (26-May-2013)	2013/05/26

プロセッサ関連資料

別紙のソフトウェア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
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
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
LSB	Least Significant Bit	最下位ビット
MD5	Message Digest 5	メッセージダイジェストアルゴリズム
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

目次

1. 概要

本章では、FLAC デコーダの概要について説明します。

1.1 仕様概要

FLAC(Free Lossless Audio Codec)は、可逆オーディオ信号符号化方式の1つであり、比較的低い演算処理量で動作可能です。オープン・ソース・ソフトウェアとして開発、公開されています。本ソフトウェアは、こ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。

なお、FLAC はフリーの規格のため、ライセンサーによる認証のような形での性能保証、知的財産権に関する保証や争議調停に関する契約、大幅な規格変更がないことの確約等は一切なく、本ソフトウェアの使用は自己責任となりますのでご注意ください。

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、別紙のソフトウェア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	FLAC 1.3.0 (26-May-2013) ・STREAMINFO データ FLAC データ・ファイル内の METADATA に含まれるストリーム情報 ・FLAC FRAME データ FLAC データ・ファイル内のリニア PCM データを符号化した FLAC FRAME データ
出力データ形式	16/32 ビット・リニア PCM
対応サンプリング周波数(Hz)	8,000 ～ 192,000
対応チャネル数	1 ～ 5.1 チャネル
入力データ PCM ビット数	4 ～ 24 ビット/サンプル
CRC	対応
リエントラント	対応
制限事項	下記機能には非対応 ・メタデータ処理 ・MD5 チェック ・ダウンミックス処理 ・上記サンプリング周波数以外のストリーム ・上記チャネル数以外のストリーム ・上記入力データ PCM ビット数以外のストリーム ・ブロックサイズが固定長の場合、1 フレーム、1 チャネルあたり 1～4608 サンプル以外のストリーム ・ブロックサイズが可変長の場合、1 フレーム、1 チャネルあたり 16～4608 サンプル以外のストリーム 【注】ブロックサイズとは、1 フレームの FLAC FRAME データに含まれる、1 チャネルあたりの PCM データサンプル数を指します。

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	ソフトウェア・ワーク領域			113,448
		<内訳>	スタティック領域	<内訳>	2,856
			スクラッチ領域		110,592
		ユーザ・ワーク領域			194,668
		<内訳>	入力バッファ	<内訳>	83,968
			出力バッファ		110,592
			各種構造体		108
		スタック領域			8,192
		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、別紙のソフトウェアマニュアルを参照してください。

【注】各ワーク領域は、5.1 チャネル、32 ビット/サンプル時の最大サイズです。デコード条件による、各ワーク領域サイズは、表 1.3を参照してください。

表1.3 デコード条件による各ワーク領域サイズ

デコード条件			各ワーク領域サイズ[byte]		
入力データの最大チャネル数	出力チャネル数	出力 PCM の 1 サンプルあたりのビット数	スクラッチ領域	入力バッファ	出力バッファ
2ch	2ch	16bit	36,864	28,672	18,432
		32bit			36,864
	3ch	16bit			27,648
		32bit			55,296
	5.1ch	16bit			55,296
		32bit			110,592
5.1ch	2ch	16bit	110,592	83,968	18,432
		32bit			36,864
	3ch	16bit			27,648
		32bit			55,296
	5.1ch	16bit			55,296
		32bit			110,592

【注】デコード条件の詳細については、3.2.1 項を参照してください。

1.2 構成

本ソフト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。

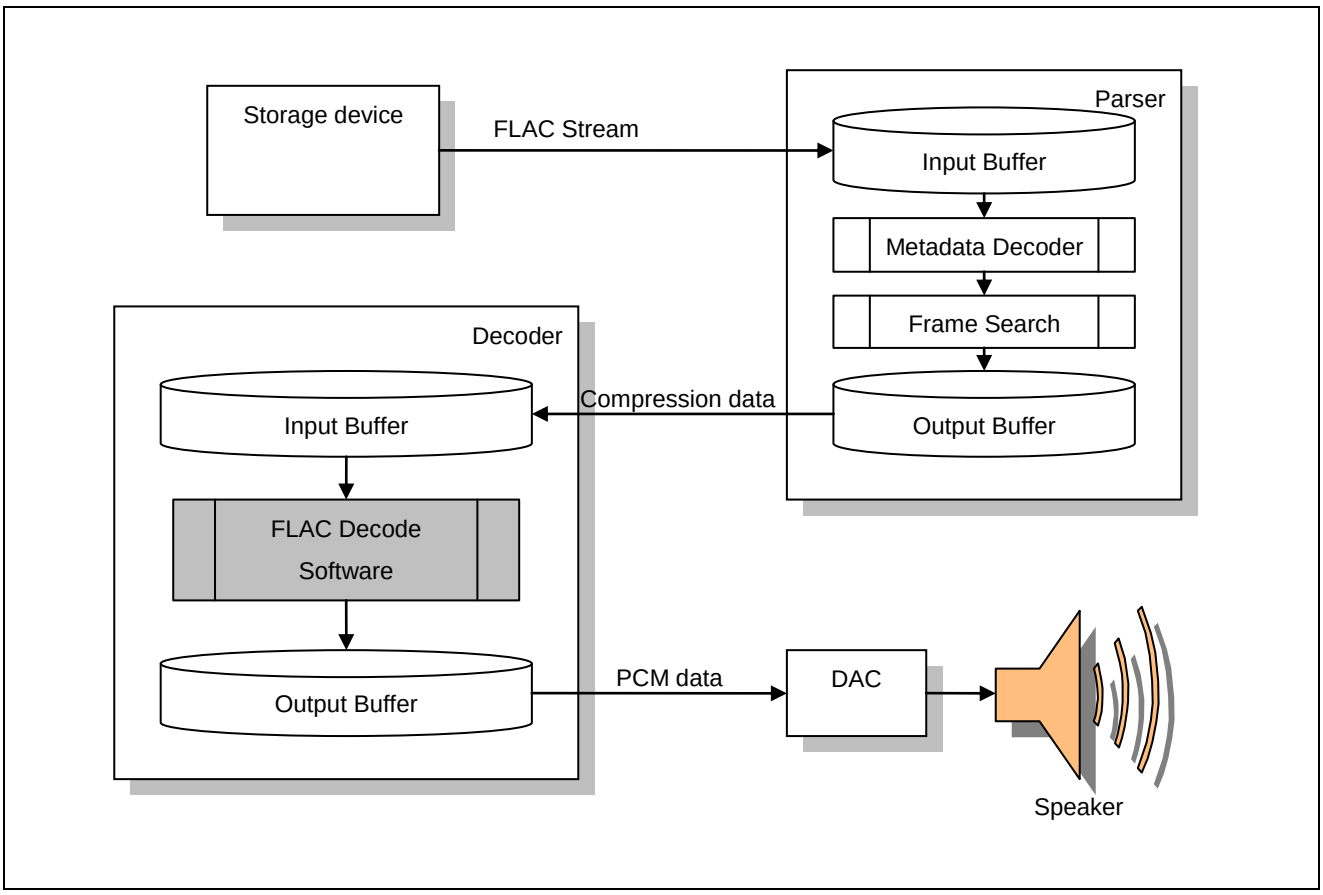

図1.1 デコード・システム構成例

1. FLAC Stream
   FLACは、サンプリングされたリニアPCMデータをFLAC規格で圧縮したデータです。サンプリング周波数、入力データPCMビット数については表1.1を参照してください。

2. Parser
   不要なメタデータなどを取り除き、フレーム単位で切り出します。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。(注意：本ソフトウェアには、含まれていません)

3. Compression data
   1フレームのFLACデータです。

4. Decoder
   Input Bufferに格納されたデータを、本ソフトウェアが処理し、Output Bufferへと出力します。

5. PCM data
   本ソフト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. ソフトウェア仕様

2.1 関数一覧

本ソフト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
flacd_GetMemorySize	必要メモリ・サイズ計算処理
flacd_Init	FLACデコーダ初期化
flacd_Decode	FLACフレーム・デコード処理
flacd_GetErrorFactor	エラー要因情報取得
flacd_GetVersion	バージョン情報取得

2.2 構造体一覧

本ソフトウェアで、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

2.3 マクロ定義

2.3.1 型定義一覧

本ソフト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

本ソフト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
FLACD_RESULT_OK	0x00000000	処理結果が正常です。
FLACD_RESULT_NG	0x00000001	処理結果が異常です。
FLACD_RESULT_WARNING	0x00000002	処理継続可能な異常が発生しました。
FLACD_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

本ソフト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	flacd_XXXX
構造体名	flacd_XXXX
関数の返却値	FLACD_RESULT_XXXX 【注】XXXXは全て大文字
エラー要因名	FLACD_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	FLACD_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

本ソフト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ソフト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のフローチャートを図 2.2に示します。網掛けした箇所は本ソフトウェアの関数が実行する処理です。斜線はサンプルパーサーの関数が実行する処理です。他の箇所はサンプルプログラムの処理です。

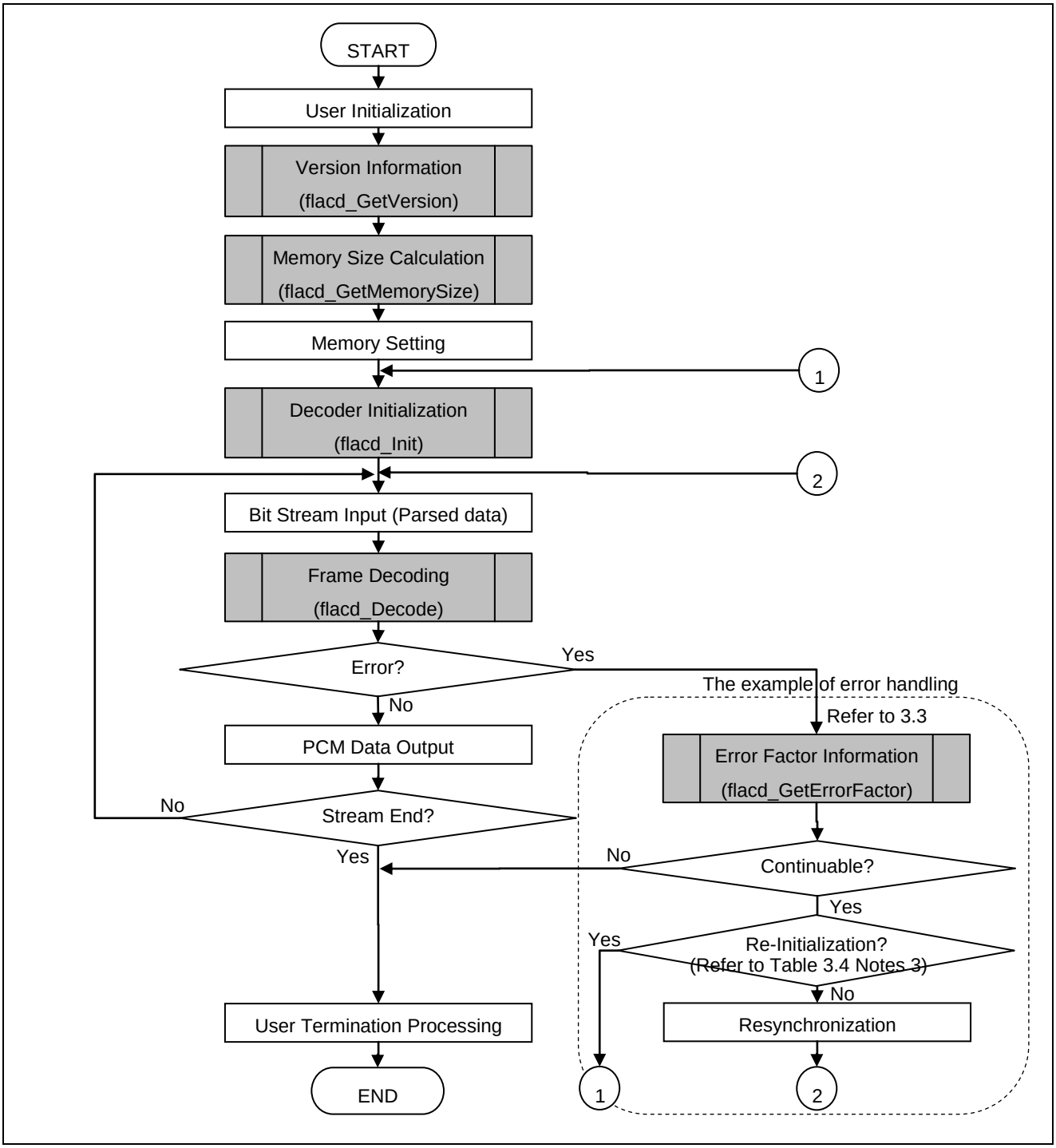

図2.1 アプリケーション処理フロー例

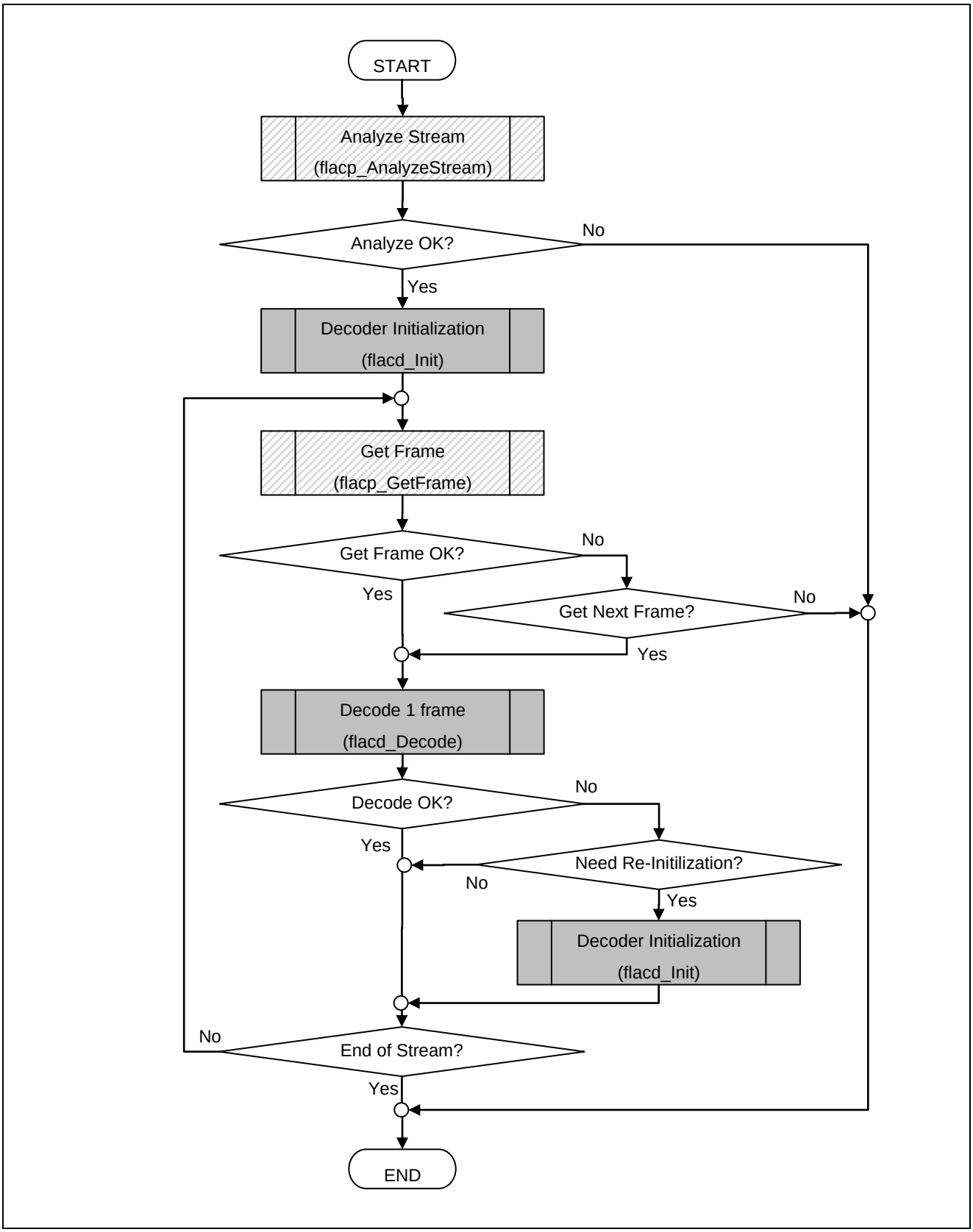

図2.2 サンプルプログラムのフローチャート

3. 関数仕様

3.1 関数仕様

次項から、本ソフト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。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 flacd_GetMemorySize 関数

概要	必要メモリ・サイズ計算処理		
構文	ACMW_INT32 flacd_GetMemorySize(const flacd_getMemorySizeConfigInfo * const pGetMemorySizeConfigInfo, flacd_getMemorySizeStatusInfo * const pGetMemorySizeStatusInfo);		
機能	本ソフト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
flacd_getMemorySizeConfigInfo *pGetMemorySizeConfigInfo		I	メモリ・サイズ取得設定情報構造体
flacd_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3.2を参照
説明	flacd_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 flacd_Init 関数

概要	FLACデコーダ初期化処理		
構文	ACMW_INT32 flacd_Init(const flacd_workMemoryInfo * const pWorkMemInfo, const flacd_initConfigInfo * const pInitConfigInfo);		
機能	本ソフト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
flacd_workMemoryInfo *pWorkMemInfo		I	ワーク・メモリ情報構造体
flacd_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3.2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 flacd_Decode 関数

概要	FLACフレーム・デコード処理		
構文	ACMW_INT32 flacd_Decode(const flacd_workMemoryInfo * const pWorkMemInfo, const flacd_decConfigInfo * const pDecConfigInfo, const flacd_ioBufferConfigInfo * const pBuffConfigInfo, flacd_decStatusInfo * const pDecStatusInfo, flacd_ioBufferStatusInfo * const pBuffStatusInfo);		
機能	FLAC FRAMEデータをデコードします。		
引数		I/O	意味
flacd_workMemoryInfo *pWorkMemInfo		I	ワーク・メモリ情報構造体
flacd_decConfigInfo *pDecConfigInfo		I	デコード設定情報構造体
flacd_ioBufferConfigInfo *pBuffConfigInfo		I	バッファ・メモリ設定情報構造体
flacd_decStatusInfo *pDecStatusInfo		O	デコード結果情報構造体
flacd_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3.2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 flacd_GetErrorFactor 関数

概要	エラー要因情報取得処理		
構文	ACMW_UINT32 flacd_GetErrorFactor(const flacd_workMemoryInfo * const pWorkMemInfo);		
機能	直前に発生した、flacd_Init、flacd_Decode関数のエラー要因を返却します。		
引数		I/O	意味
flacd_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3.4を参照
説明	直前に発生した、flacd_Init、flacd_Decode関数のエラー要因を取得します。 flacd_Init、flacd_Decode関数以外のエラー、およびflacd_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flacd_Init、flacd_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 flacd_GetVersion 関数

概要	バージョン情報取得処理		
構文	ACMW_UINT32 flacd_GetVersion(void);		
機能	本ソフト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	本ソフト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、本ソフト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】 flacd_getMemorySizeConfigInfo

【機能】 flacd_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_UINT16     nInputChannel;
    ACMW_UINT16     nOutputChannel;
    ACMW_UINT16     nOutBitsPerSample;
} flacd_getMemorySizeConfigInfo;
```

【メンバの説明】

メンバ変数名	内容	
nInputChannel	入力データの最大チャネル数	
	0x0000	2ch入力
	other	5.1(6)ch入力
nOutputChannel	出力データのチャネル数	
	0x0000	入力データが2chを超える場合は、L、Rchのみ出力する。(ダウンチャネル動作)
	0x0001	入力データが3、5、5.1(6)chの場合に、L、C、Rchのみ出力する。(ダウンミックス用) 入力データが4chの場合は、L、Rchのみ出力する。
	0x0002	入力データの各chを全て出力する。
	other	予約
nOutBitsPerSample	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

【備考】 領域の確保はユーザが行ってください。ユーザは、flacd_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 】 flacd_getMemorySizeStatusInfo

【 機 能 】 必要メモリ・サイズ計算処理(flacd_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;
} flacd_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最小メモリ・サイズ[byte]
nOutputBufferSize	出力バッファ領域の必要最小メモリ・サイズ[byte] *1
nStackSize	ソフトウェア・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、flacd_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サイズは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】 flacd_workMemoryInfo

【機能】 本ソフト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;
} flacd_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(flacd_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】 flacd_initConfigInfo

【機能】 初期化処理(flacd_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     nInputChannel;
    ACMW_UINT16     nOutputChannel;
    ACMW_UINT16     nOutBitsPerSample;
} flacd_initConfigInfo;
```

【メンバの説明】

メンバ変数名	内容	
nInputChannel	入力データの最大チャネル数 ここで指定したチャネル数を超える入力データを入力した場合は、FLACD_ERR_NOT_SUPPORTED_DATAエラーとなる。	
	0x0000	2ch入力
	other	5.1(6)ch入力
nOutputChannel	出力データのチャネル数	
	0x0000	入力データが2chを超える場合は、L、Rchのみ出力する。(ダウンチャネル動作)
	0x0001	入力データが3、5、5.1(6)chの場合に、L、C、Rchのみ出力する。(ダウンミックス用) 入力データが4chの場合は、L、Rchのみ出力する。
	0x0002	入力データの各chを全て出力する。
	other	予約
nOutBitsPerSample	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

【備考】 領域の確保はユーザが行ってください。ユーザは、flacd_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。

3.2.5 デコード設定情報構造体

【構造体名】 flacd_decConfigInfo

【機能】 flacd_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_UINT16     nMinBlockSize;
    ACMW_UINT16     nMaxBlockSize;
    ACMW_UINT32     nMinFrameSize;
    ACMW_UINT32     nMaxFrameSize;
    ACMW_UINT32     nSampleRate;
    ACMW_UINT16     nChannels;
    ACMW_UINT16     nBitsPerSample;
} flacd_decConfigInfo;
```

【メンバの説明】

メンバ変数名	内容
nMinBlockSize	1ブロックに含まれる最小サンプル数
nMaxBlockSize	1ブロックに含まれる最大サンプル数 最小サンプル数と最大サンプル数が同じ場合は、全てのブロックサイズが固定であることを示します。 本ソフトウェアでは、4608サンプルを超えるデータは非対応。
nMinFrameSize	1フレームの最小サイズ[バイト]
nMaxFrameSize	1フレームの最大サイズ[バイト]
nSampleRate	サンプリング周波数[Hz](8000～96000)
nChannels	チャネル数(1～6) 本ソフトウェアでは5.1(6)chを超えるデータは非対応。
nBitsPerSample	1サンプルあたりのビット数(4～24) 本ソフトウェアでは24ビット/サンプルを超えるデータは非対応。

【備考】 領域の確保はユーザが行ってください。ユーザは、flacd_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。
FLACデータ・ファイルには、最初のMETADATAとしてSTREAMINFOデータが含まれるため、STREAMINFOの内容をデコード設定情報構造体に設定してください。

3.2.6 デコード結果情報構造体

【 構 造 体 名 】 flacd_decStatusInfo

【 機 能 】 デコード処理(flacd_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   nSampleRate;
    ACMW_UINT32   nDecodedSamples;
    ACMW_UINT16   nChannels;
    ACMW_UINT16   nChannelInfo;
    ACMW_UINT16   nBitsPerSample;
} flacd_decStatusInfo;
```

【メンバの説明】

メンバ変数名	内容
nSampleRate	サンプリング周波数[Hz](8000～96000)*1
nDecodedSamples	1チャネルあたりの出力サンプル数
nChannels	出力チャネル数*1
nChannelInfo	出力チャネル構成情報（図3.1出力チャネル構成情報参照）
nBitsPerSample	サンプルあたりのビット数*1

【 備 考 】 領域の確保はユーザが行ってください。ユーザは、flacd_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 】 C言語では、この構造体を構成するメンバの間を、自動的にアライメントします。

【 備 考 3 】 *1：FLAC FRAMEデータを解析し、実際のデコードに使用した値を格納しています。

MSB LSB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					LFE	0	Rb	Lb	0	0	0	0	R	L	C

図3.1 出力チャネル構成情報

【注】C : front center, mono
L : (front) left
R : (front) right
Lb : back/surround left
Rb : back/surround right
LFE : low frequency effect

3.2.7 バッファ・メモリ設定情報構造体

【構造体名】 flacd_ioBufferConfigInfo

【機能】 flacd_Decode関数で使用する入出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     nInBuffSetDataSize;
    void **         pOutBuffStart;
    ACMW_UINT32     nOutBuffSize
} flacd_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
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]*1

【備考】 領域の確保はユーザが行ってください。ユーザは、flacd_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】 pOutBuffStartl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。
初期化設定情報構造体で、16ビットPCM出力を指定した場合、出力バッファのポインタには、2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。
初期化設定情報構造体で、32ビットPCM出力を指定した場合は、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。

【備考3】 *1：出力バッファのサイズは各チャネルで共通です。

3.2.8 バッファ・メモリ結果情報構造体

【構造体名】 flacd_ioBufferStatusInfo

【機能】 flacd_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;
} flacd_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]
pOutBuffLast	チャネルごとの出力データの書き出し終了アドレス*1
nOutBuffUsedDataSize	各チャネルの出力バッファの消費データ・サイズ [byte] *2

【備考】 領域の確保はユーザが行ってください。ユーザは、flacd_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：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。

3.3 エラー処理

本ソフトウェアの関数は、表3.2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、flacd_GetErrorFactor関数を実行してください。

3.3.1 エラー・コード

本ソフトウェアのエラー・コードについて説明します。

表3.2 エラー・コード

エラー・コード (32bit)	値	説明	再初期化
① FLACD_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 FLACD_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 FLACD_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(flacd_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 FLACD_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。flacd_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	―

表3.3 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	flacd_GetMemorySize	flacd_Init	flacd_Decode	flacd_GetErrorFactor*1	flacd_GetVersion*2
FLACD_RESULT_OK	○	○	○	―	―
FLACD_RESULT_NG	―	○	○	―	―
FLACD_RESULT_WARNING	―	―	○	―	―
FLACD_RESULT_FATAL	○	○	○	○	―

【注】○：出力する可能性がある、―：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。FLACD_RESULT_FATAL が発生した場合を除き、flacd_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 3.4に示します。

表3.4 エラー要因一覧

errorFactor(32bit)	値	説明	表3.2	PCM
FLACD_ERR_NONE	0x00000000	正常終了した。エラー要因は存在しない。	①	正常*1
FLACD_ERR_POINTER	0x00000010	ポインタ値が不正だった。	②	なし
FLACD_ERR_PARAMETER	0x00000020	パラメータが不正だった。	②	なし
FLACD_ERR_SEQUENCE	0x00000040	本ライブラリ関数の実行順序が不正だった。*2	②	なし
FLACD_ERR_SHORT_INPUT_DATA	0x00000100	入力データが不足していることを検出した。	②	なし
FLACD_ERR_NOT_SUPPORTED_DATA	0x00001000	サポートしていないデータであることを検出した。	②	なし
FLACD_ERR_LOST_SYNC	0x00010000	フレーム開始位置が見つからなかった。	②	なし
FLACD_ERR_CHANGE_FRAME_HEADER	0x00020000	ヘッダ情報が変更された。*3	②	なし
FLACD_ERR_CRC	0x00200000	CRCエラーが発生した。	③	あり
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、FLACD_ERR_NONEとみなし、デコード処理を継続し出力します。

*2 実行順序が不正のため再初期化が必要です。

*3 新しいフレーム情報で再生したい場合は再初期化が必要です。

表3.5 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	flacd_GetMemorySize	flacd_Init	flacd_Decode	flacd_GetErrorFactor*1	flacd_GetVersion
FLACD_ERR_NONE	ー	○	○	ー	ー
FLACD_ERR_POINTER	ー	○	○	ー	ー
FLACD_ERR_PARAMETER	ー	○	○	ー	ー
FLACD_ERR_SEQUENCE	ー	ー	○	○	ー
FLACD_ERR_SHORT_INPUT_DATA	ー	ー	○	ー	ー
FLACD_ERR_NOT_SUPPORTED_DATA	ー	ー	○	ー	ー
FLACD_ERR_LOST_SYNC	ー	ー	○	ー	ー
FLACD_ERR_CHANGE_FRAME_HEADER	ー	ー	○	ー	ー
FLACD_ERR_CRC	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 FLACD_RESULT_FATAL を返します。

3.4 メモリ仕様

本ソフト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3.6 スクラッチ領域情報

内容	本ソフトウェアを使用する場合、一時的に値を保存する領域です。 本ソフト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、本ソフト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、flacd_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。 本ソフト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、本ソフト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	4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3.7 スタティック領域情報

内容	本ソフト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、本ソフト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、flacd_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 ソフトウェア・スタック領域

表3.8 ソフトウェア・スタック領域情報

内容	本ソフトウェアが使用する、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、flacd_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。 本ソフト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

本ソフト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3.9 入力バッファ領域情報

内容	本ソフト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（FLAC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】本ソフトウェアは、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	必ずflacd_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。
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。

flacd_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。

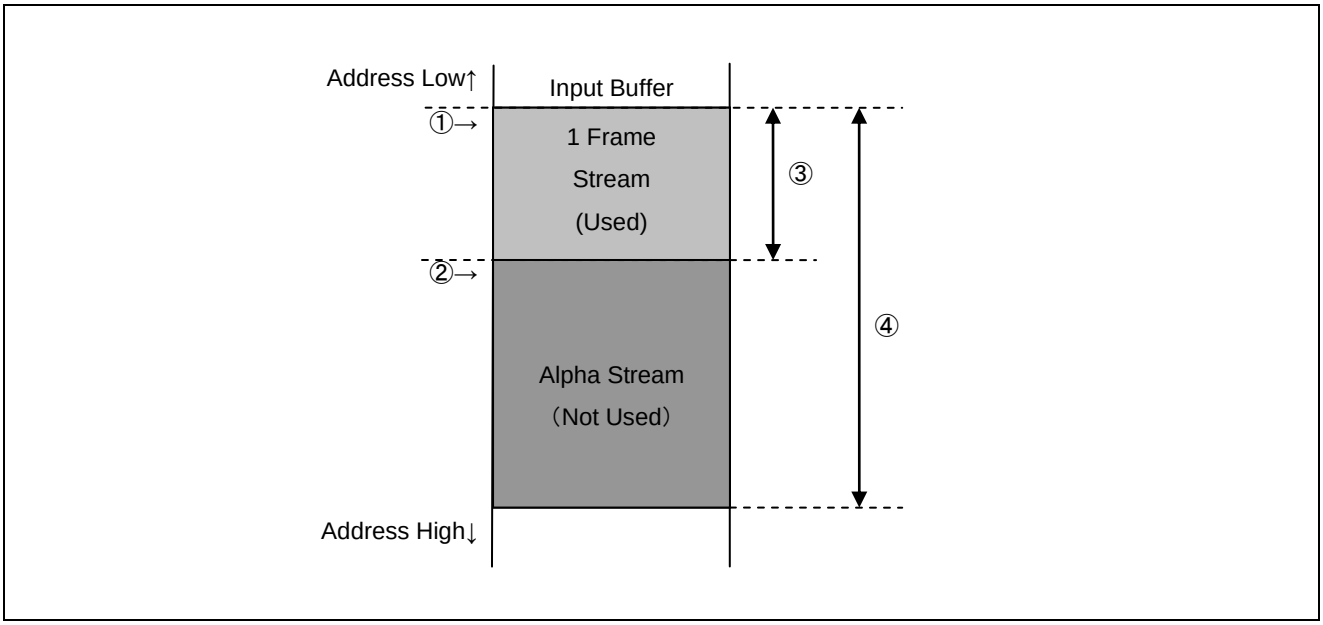

図3.2　入力バッファにおける各種構造体メンバの情報イメージ例

【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3.10　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	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
④	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、③～④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 3.3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。

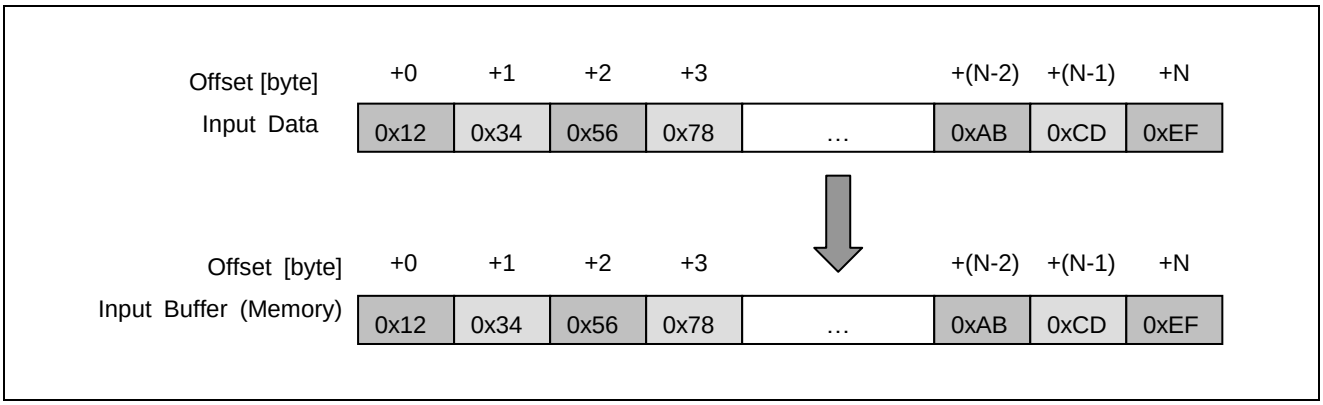

図3.3 入力データの格納方式

3.4.6 出力バッファ

表3.11 出力バッファ領域情報

内容	本ソフト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、ノン・インターリーブ形式で格納されます。 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flacd_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】flacd_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。

flacd_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 3.4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、入力データやデコード条件によって変化します。デコード結果情報構造体の nDecodedSamples を参照してください。

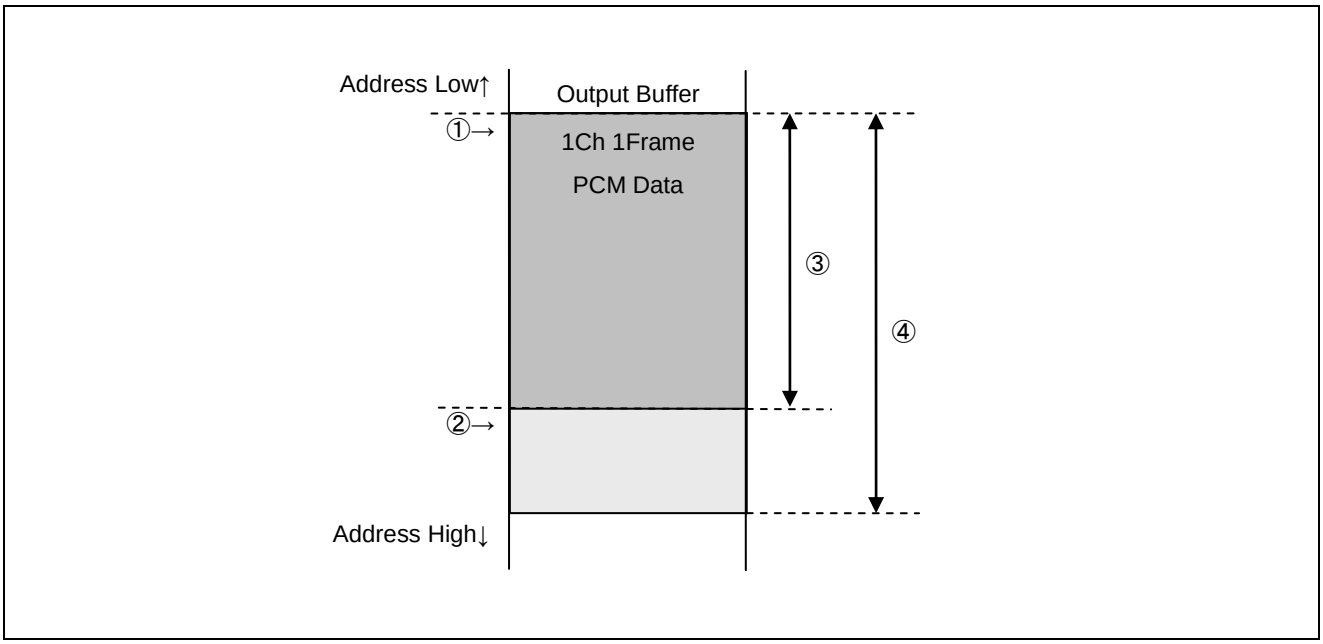

図3.4 出力バッファにおける各種構造体メンバの情報イメージ例

【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3.12 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。

FLAC 規格における、出力チャネル数と出力バッファの順番の関係を、表 3.15、表 3.14、表 3.15に示します。

表3.13 チャネル数と出力バッファの順番の関係 (nOutputChannel = 0 の場合)

入力データのチャネル数	pOutBuffStart					
	[0]	[1]	[2]	[3]	[4]	[5]
1	mono	-	-	-	-	-
2	left	right	-	-	-	-
3	left	right	-	-	-	-
4	front left	front right	-	-	-	-
5	front left	front right	-	-	-	-
6	front left	front right	-	-	-	-

表3.14 チャネル数と出力バッファの順番の関係 (nOutputChannel = 1 の場合)

入力データのチャネル数	pOutBuffStart					
	[0]	[1]	[2]	[3]	[4]	[5]
1	mono	-	-	-	-	-
2	left	right	-	-	-	-
3	left	right	center	-	-	-
4	front left	front right	-	-	-	-
5	front left	front right	front center	-	-	-
6	front left	front right	front center	-	-	-

表3.15 チャネル数と出力バッファの順番の関係 (nOutputChannel = 2 の場合)

入力データのチャネル数	pOutBuffStart					
	[0]	[1]	[2]	[3]	[4]	[5]
1	mono	-	-	-	-	-
2	left	right	-	-	-	-
3	left	right	center	-	-	-
4	front left	front right	back left	back right	-	-
5	front left	front right	front center	back/surround left	back/surround right	-
6	front left	front right	front center	LFE	back/surround left	back/surround right

(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 3.5、図 3.6参照)は、リトル・エンディアンになります。

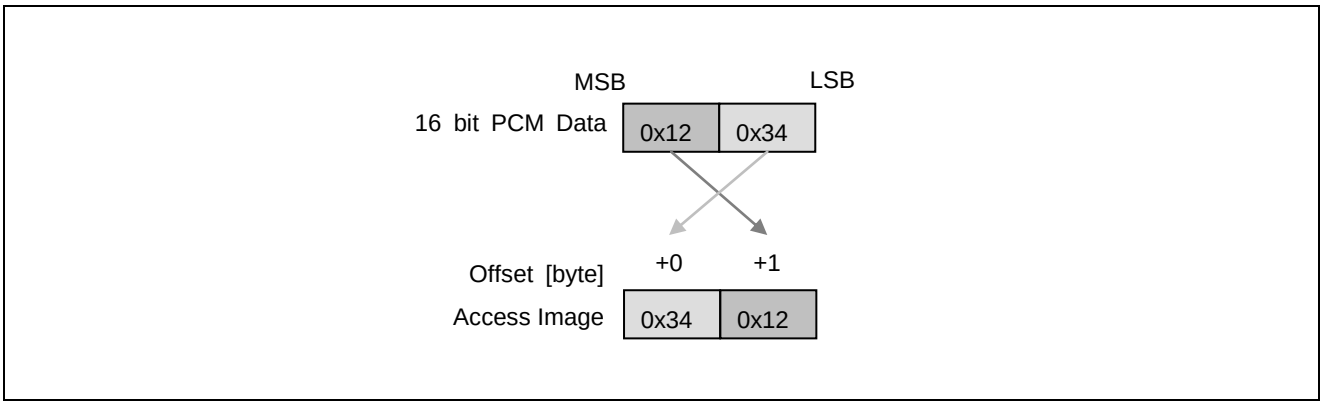

図3.5 PCM データ(リトル・エンディアン)アクセス(16 ビット)

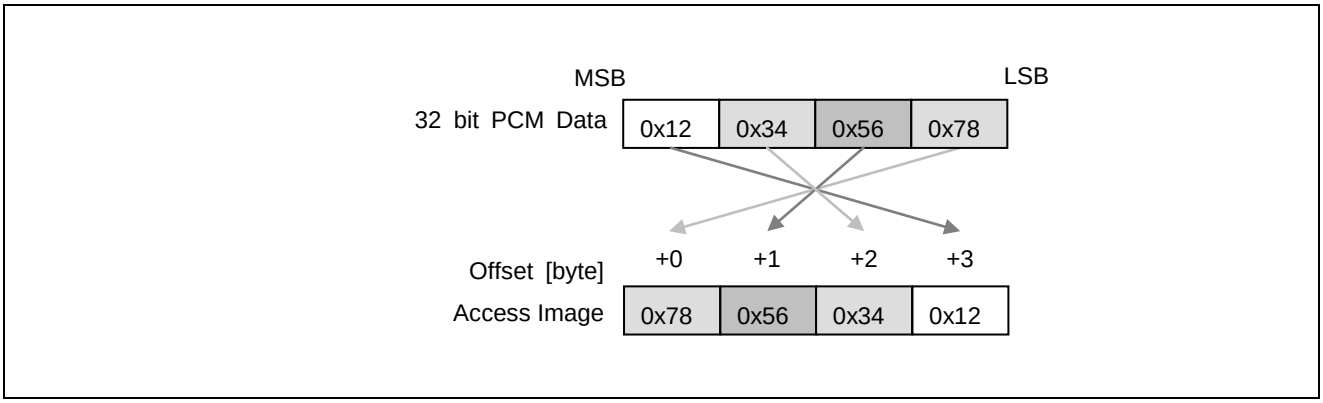

図3.6 PCM データ(リトル・エンディアン)アクセス(32 ビット)

図 3.7、図 3.8に PCM データのビット配置を示します。
本ソフトウェアは、入力データの PCM ビット数に関わらず、16/32 ビット・リニア PCM データとして出力します。入力データの PCM ビット数が出力 PCM ビット数に満たない場合は、MSB 側に格納され下位ビットは 0 埋めされます。

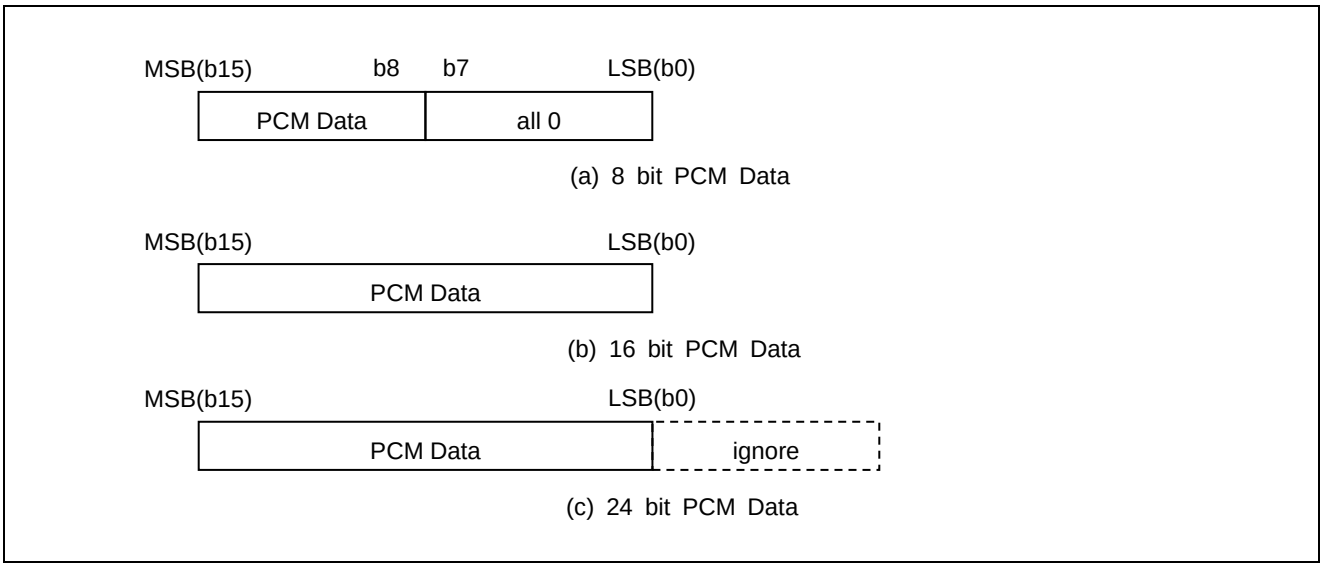

図3.7 PCM データのビット配置(16 ビット)

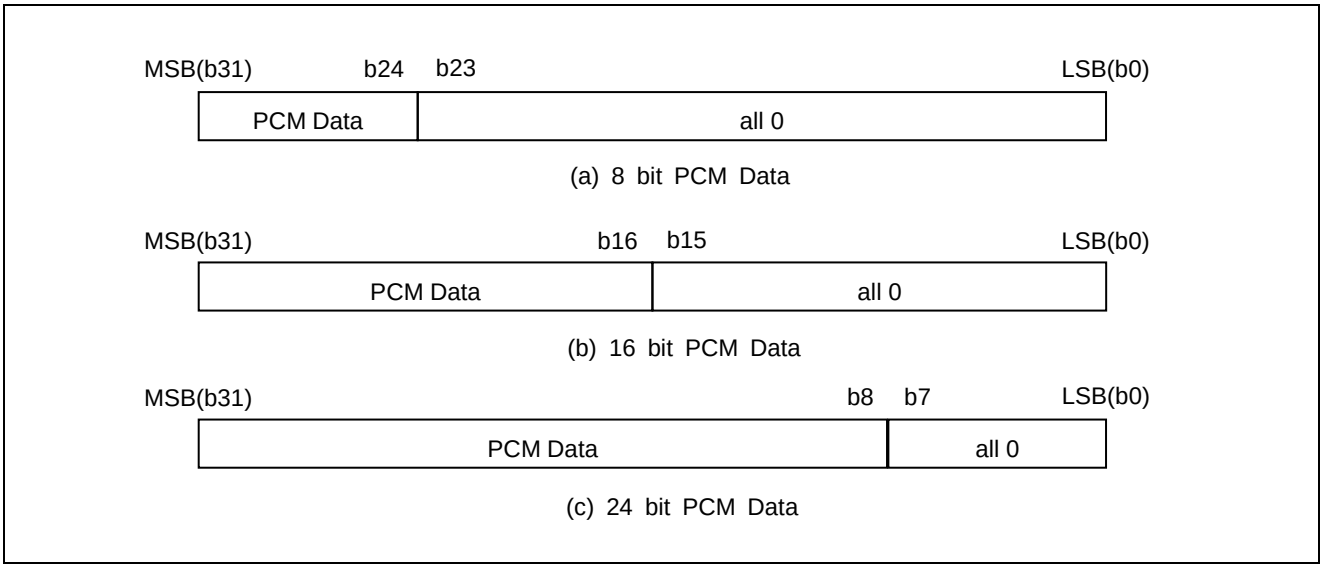

図3.8 PCM データのビット配置(32 ビット)

3.5 入力データ

本ソフトウェアは、FLAC FRAME データを入力することが可能です。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。

FLAC データ・ファイルは以下のように構成されています。FLAC データ・ファイルに関する詳細については、規格を参照してください。

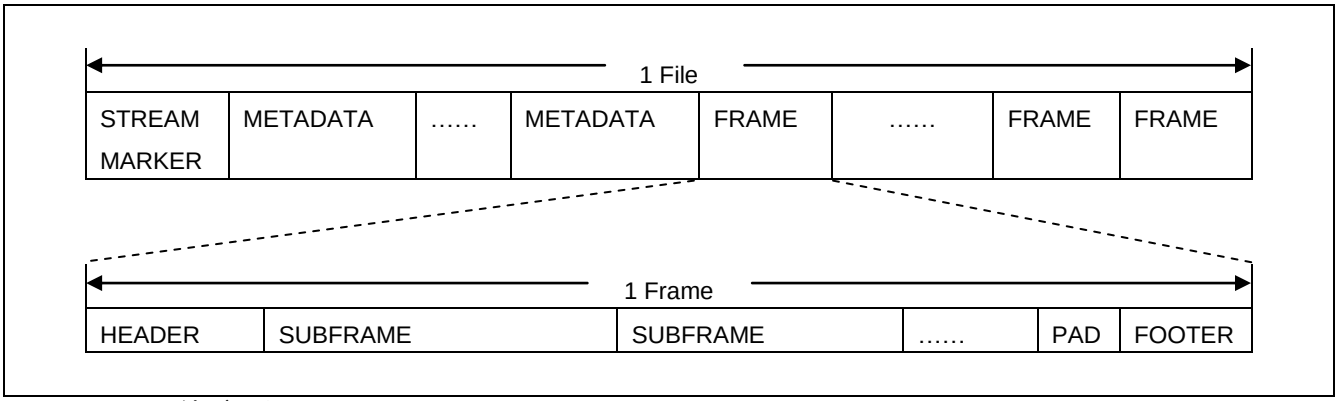

図3.9 圧縮データ・フォーマット

【STREAM MARKER】

FLACストリーム・データを示す“fLaC”の4バイトアスキーコード

【METADATA】

音声データ以外のストリームに関する情報(画像やコメント等)

必ず、最初のMETADATAとしてSTREAMINFOデータが含まれます。本ソフトウェアを使用する際には、STREAMINFOの内容を初期化設定情報構造体に設定し、flacd_Init関数を実行する必要があります。

【FRAME】

エンコードされた音声データが含まれます。本ソフトウェアでは、このFLAC FRAMEデータを入力してください。

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。出力データ形式については、3.4.6項を参照してください。出力されるサンプル数は入力データ内容に依存し、必ずしも出力バッファ・サイズ分のPCMデータが出力されるわけではありません。

また、本ソフトウェアでは、FLAC FRAMEデータをデコードして生成されるPCMデータに対する、ダウンミックス、レベル調整といった処理を行いません。このような処理が必要な場合は、ユーザが実装する必要があります。以下に、ダウンミックス処理の例を示します。

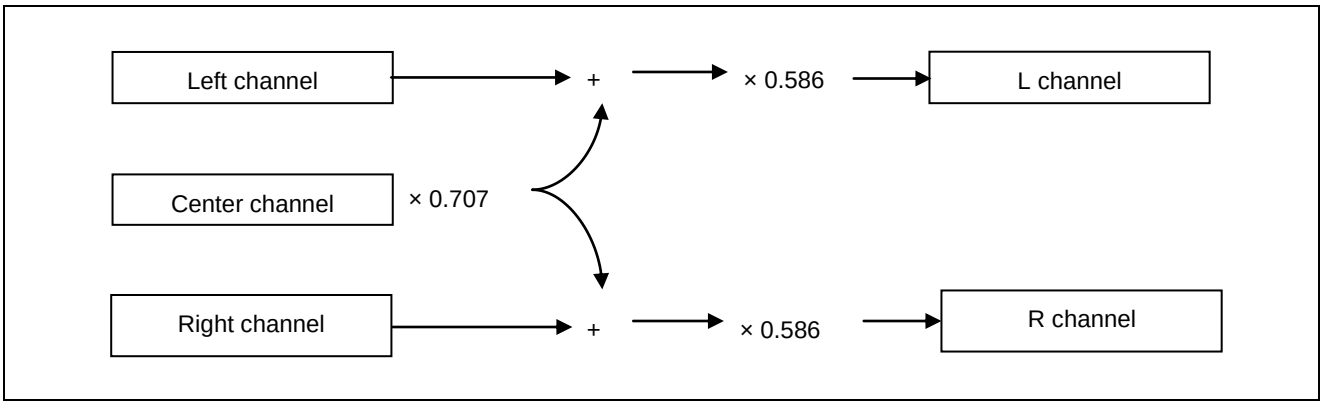

図3.10 ダウンミックス処理例

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

本ソフト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

本ソフト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) flacd_GetMemorySize 関数

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

flacd_Init 関数実行後の任意のタイミングで実行可能です。

(5) flacd_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

本ソフト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

本ソフトウェアは、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

本ソフト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 ソフトウェアの監視

本ソフト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ソフト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	FLAC Decode Software 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	2014.07.11	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
		44	「4.2.4 ミドルウェアの監視」の項目を追記

FLAC Decode Software 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/

FLAC Decode Software

RENESAS

ルネサス エレクトロニクス株式会社