お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。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。
3. 当社製品を改造、改変、複製等しないでください。
4. 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。
5. 輸出に際しては、「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。本資料に記載されている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の目的で使用しないでください。また、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に使用することができません。
6. 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。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されてい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生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   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

   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、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社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。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、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、社会的損害など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、機器またはシステムとしての出荷保証をお願いいたします。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、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、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。
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固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

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。

RENESAS

ユーザーズ・マニュアル

μSAP77016-B17

AAC オーディオ・エンコーダ・ミドルウエア

対象デバイス

μPD77110
μPD77113A
μPD77114
μPD77115
μPD77210
μPD77213

資料番号 U16314JJ1V0UM00（第 1 版）
発行年月 December 2002 NS CP（N）

〔メ　モ〕

目次要約

Windows は，米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。

- 本資料に記載されている内容は2002年12月現在のもので，今後，予告なく変更することがあります。量産設計の際には最新の個別データ・シート等をご参照ください。
- 文書による当社の事前の承諾なしに本資料の転載複製を禁じます。当社は，本資料の誤りに関し，一切その責を負いません。
- 当社は，本資料に記載された当社製品の使用に関連し発生した第三者の特許権，著作権その他の知的財産権の侵害等に関し，一切その責を負いません。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。
- 本資料に記載された回路，ソフトウエアおよびこれらに関する情報は，半導体製品の動作例，応用例を説明するものです。お客様の機器の設計において，回路，ソフトウエアおよびこれらに関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，一切その責を負いません。
- 当社は，当社製品の品質，信頼性の向上に努めておりますが，当社製品の不具合が完全に発生しないことを保証するものではありません。当社製品の不具合により生じた生命，身体および財産に対する損害の危険を最小限度にするために，冗長設計，延焼対策設計，誤動作防止設計等安全設計を行ってください。
- 当社は，当社製品の品質水準を「標準水準」，「特別水準」およびお客様に品質保証プログラムを指定していただく「特定水準」に分類しております。また，各品質水準は，以下に示す用途に製品が使われることを意図しておりますので，当社製品の品質水準をご確認ください。

  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

  特別水準：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，生命維持を目的として設計されていない医療機器

  特定水準：航空機器，航空宇宙機器，海底中継機器，原子力制御システム，生命維持のための医療機器，生命維持のための装置またはシステム等

当社製品のデータ・シート，データ・ブック等の資料で特に品質水準の表示がない場合は，標準水準製品であることを表します。意図されていない用途で当社製品の使用をお客様が希望する場合には，事前に当社販売窓口までお問い合わせください。

（注）

（1）本事項において使用されている「当社」とは，NECエレクトロニクス株式会社およびNEC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う。

（2）本事項において使用されている「当社製品」とは，（1）において定義された当社の開発，製造製品をいう。

M8E　02.11

はじめに

対象者　このマニュアルは，μPD77016 ファミリの応用システムを設計，開発するユーザを対象としています。

μPD77016 ファミリは，μPD7701×ファミリ（μPD77015, 77016, 77017, 77018A, 77019）と，μPD77111 ファミリ（μPD77110, 77111, 77112, 77113A, 77114, 77115），μPD77210 ファミリ（μPD77210, 77213）の総称です。
ただし，このマニュアルでは，μPD77110, 77113A, 77114, 77115, 77210, 77213 を対象デバイスにしています。

目的　このマニュアルは，μPD77016 ファミリの応用システムを設計，開発する際にサポートするミドルウエアを，ユーザに理解していただくことを目的としています。

構成　このマニュアルは，大きく分けて次の内容で構成されています。

- 概　説
- ライブラリ仕様
- インストレーション
- システム例
- サンプル・プログラム・ソース

読み方　このマニュアルの読者は，電気，論理回路やマイクロコンピュータ，C 言語に関する一般的知識が必要となります。

μPD77111 ファミリのハードウエア機能を知りたいとき
→**μPD77111 ファミリ　ユーザーズ・マニュアル　アーキテクチャ編**を参照してください。

μPD77210 ファミリのハードウエア機能を知りたいとき
→**μPD77210 ファミリ　ユーザーズ・マニュアル　アーキテクチャ編**を参照してください。

μPD77016 ファミリの命令機能を知りたいとき
→**μPD77016 ファミリ　ユーザーズ・マニュアル　命令編**を参照してください。

凡例

データ表記の重み	：左が上位桁，右が下位桁
アクティブ・ロウの表記	： $\overline{\times\times\times}$（端子，信号の名称に上線）
注	：本文中につけた注の説明
注意	：気をつけて読んでいただきたい内容
備考	：本文中の補足説明
数の表記	： 2 進数 … ××××または 0b××××
	10 進数 … ××××または 0t××××
	16 進数 … 0x××××

関連資料　関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。あらかじめご了承ください。

μPD77016 ファミリに関する資料

資料名 / 品名	パンフレット	データ・シート	ユーザーズ・マニュアル		アプリケーション・ノート	
			アーキテクチャ編	命令編	基本ソフトウエア編	ライブラリ編
μPD77110	U12395J	U12801J	U14623J	U13116J	U11958J	U12021J
μPD77111						
μPD77112						
μPD77113A		U14373J				
μPD77114						
μPD77115		U14867J				
μPD77210		U15203J	U15807J			
μPD77213						

開発ツールに関する資料

資 料 名		資料番号
HSM77016　ユーザーズ・マニュアル		U11602J
WB77016　ユーザーズ・マニュアル	言語編	U10078J
	操作編	U11506J
ID77016　ユーザーズ・マニュアル		U10118J
CC77016　ユーザーズ・マニュアル		U15037J
RX77016　ユーザーズ・マニュアル	機能編	U14397J
	コンフィギュレーション・ツール編	U14404J
RX77016　アプリケーション・ノート	HOST API 編	U14371J

ミドルウエアに関する資料

資 料 名	資料番号
μSAP77016-B08 ユーザーズ・マニュアル　（AAC デコーダ）	U15152J
μSAP77016-B17 ユーザーズ・マニュアル　（AAC エンコーダ）	このマニュアル

規格に関する資料

資 料 名	発行年月
ISO/IEC 13818-7 MPEG-2 Advanced Audio Coding, AAC	April, 1997

注意　上記関連資料は，予告なしに内容を変更することがあります。設計などには，必ず最新の資料をご使用ください。

目　次

図の目次

図番号　　タイトル，ページ

表の目次

表番号　　タイトル，ページ

第1章 概　　説

1.1　ミドルウエア

ミドルウエアとは，プロセッサの性能をできるだけ引き出せるようにチューニングされたソフトウエア群で，従来，ハードウエアが行っていた処理をソフトウエアで実現したものです。

DSP という高性能プロセッサの出現，そして DSP が手軽にシステムに組み込める環境がそろってきたため，ミドルウエアという概念が現実のものとなってきました。

当社は，μPD77016 ファミリ用にマルチメディア・システムを実現する要素技術を提供しています。たとえば音声コーデック，画像データの圧縮／伸長といったミドルウエアをタイムリに提供し，お客様のシステム開発を支援します。

μSAP77016-B17 は，AAC 方式のエンコード機能を提供するミドルウエアです。

1.2　AAC オーディオ・エンコーダ

このマニュアルでは，AAC を Advanced Audio Coding の略称として使用しています。

MPEG-2 AAC は，MPEG-1 オーディオとの互換性を排除することによって高品質・高圧縮率を達成したオーディオ符号化方式です。μSAP77016-B17 は，その符号方式に準拠したものです。

圧縮データ・フォーマットは，「ISO/IEC 13818-7 MPEG-2 Advanced Audio Coding, AAC」の規格書に準拠しています。取り扱うオーディオ・データは，8 kHz ～ 96 kHz の周波数（表 1-1参照）でサンプリングされた 16 ビット・リニア PCM データです。

表 1-1　サンプリング周波数

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

1. 2. 1　エンコーダ概略

図 1-1にμSAP77016-B17 を使用したエンコーダの構成例を示します。また，図 1-2にμSAP77016-B17 を使用したエンコーダのシステム構成例を示します。

図 1-1　エンコーダ構成例

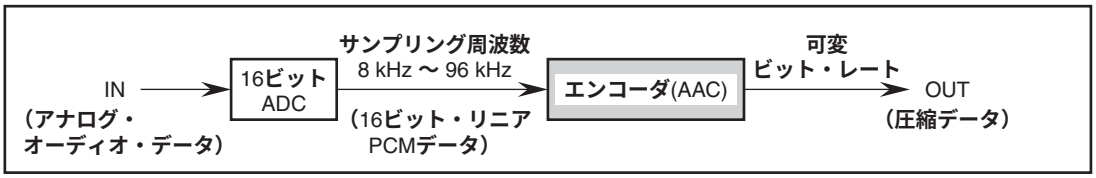
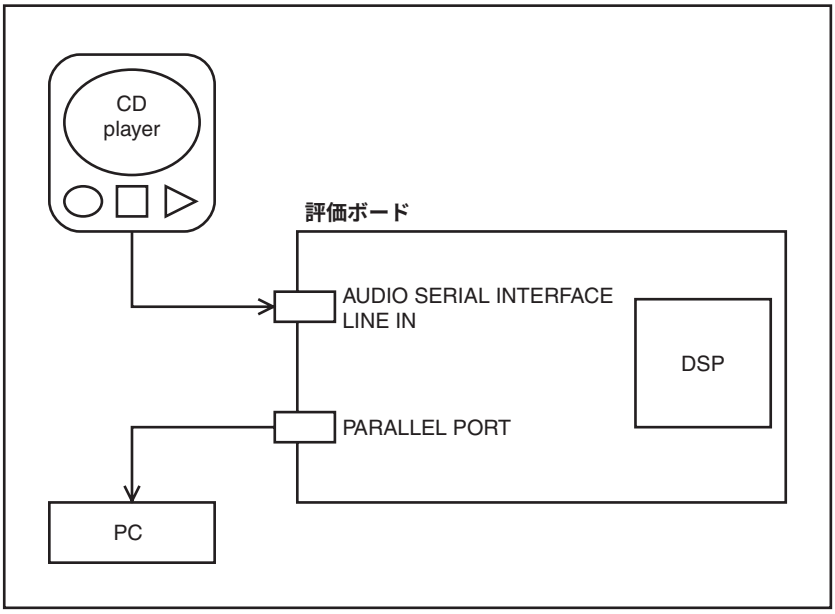


図 1-2　システム構成例



(1) 入力データ

8 kHz ～ 96 kHz（表 1-1参照）でサンプリングされた 16 ビット・リニア PCM データです。

(2) AAC オーディオ・エンコーダ

AAC オーディオ・エンコーダは，16 ビット・リニア PCM データを読み込み，設定されたビット・レートに対して符号量制御を行いながらデータを出力します。フレーム当りのビット・レートは可変です。

設定可能なビット・レートの最大値は，サンプリング周波数によって異なります。ビット・レートは，最大値までの任意の値をとります。ビット・レートの最大値を表 1-2に示します。

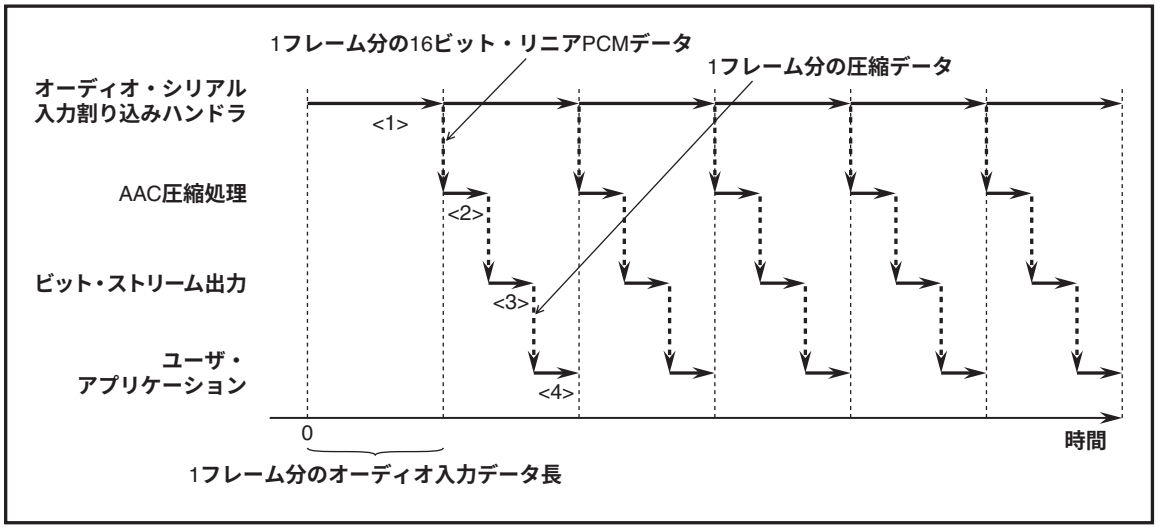
表 1-2　最大ビット・レート

サンプリング周波数 [Hz]	最大ビット・レート [bps/ch]
8000	48000
11025	66150
12000	72000
16000	96000
22050	132300
24000	144000
32000	192000
44100	264600
48000	288000
64000	384000
88200	529200
96000	576000

(3) タイミング・ダイアグラム

AAC オーディオ・エンコーダのタイミング・ダイアグラムを図 1-3に示します。

図 1-3　タイミング・ダイアグラム



<1> 1 フレーム分の 16 ビット・リニア PCM データを入力します。

<2> 1 フレーム分の 16 ビット・リニア PCM データをバッファリングして，圧縮します。

<3> 圧縮データをバッファリングして，出力します。

<4> ユーザが適宜処理を行います。

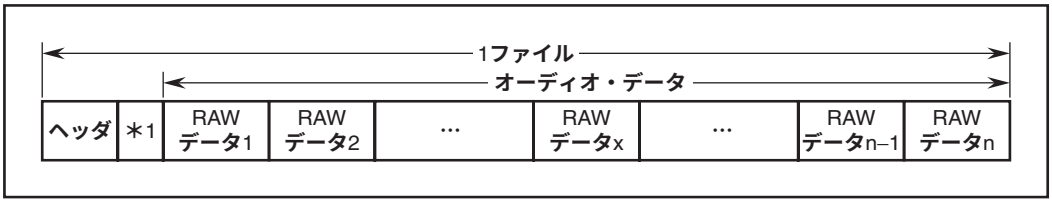
1.3 圧縮データ･フォーマット

圧縮データ･フォーマットは，「ISO/IEC 13818-7 MPEG-2 Advanced Audio Coding, AAC」の規格書に準拠しています。

1.3.1 ADIF フォーマット概要

図 1-4に ADIF フォーマットの概要を示します。

図 1-4 ADIF フォーマット

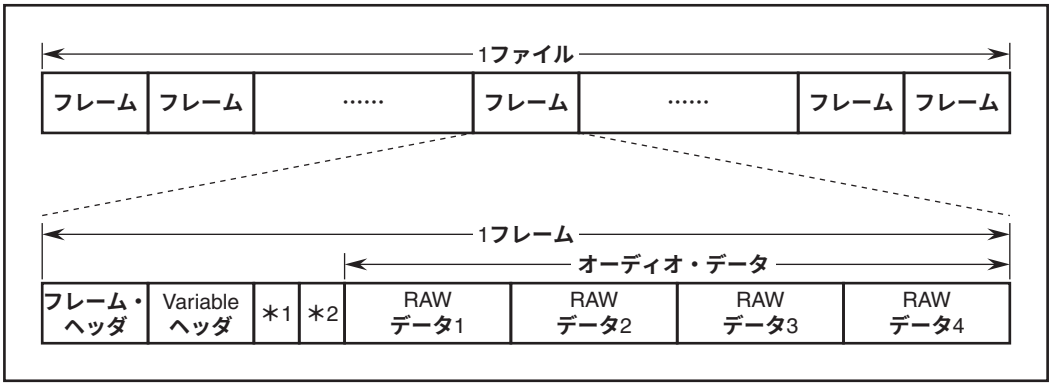


備考 ヘッダ：同期を取るためのサンプリング周波数，ビット・レート，モードなどの情報を含みます。
オーディオ・データ：オーディオ・サンプルに関する情報です。複数の RAW データで構成されています。
RAW データ：デコードを行う最小単位のビット・ストリームです。
＊1：ビット・アラインメント

1.3.2 ADTS フォーマット概要

図 1-5に ADTS フォーマットの概要を示します。

図 1-5 ADTS フォーマット



備考 フレーム・ヘッダ：同期を取るためのサンプリング周波数，ビット・レート，モードなどの情報を含みます。
Variable ヘッダ：オーディオ・データ中に含まれる RAW データの個数などのデコードに必要な情報です。
オーディオ・データ：オーディオ・サンプルに関する情報です。複数の RAW データで構成されています。
RAW データ：デコードを行う最小単位のビット・ストリームです。1 フレーム中には，最大 4 個まで存在します。
＊1：エラー・チェック
＊2：ビット・アラインメント

1.4 製品概要

1.4.1 特　徴

・MPEG-2 AAC（Advanced Audio Coding）LC（Low Complexity）プロファイルに対応。
・フロント 2 チャネルのみ対応（モノラル／ステレオ）。
・16 ビット・リニア PCM データ入力。
・設定ビット・レート（表 1-2　最大ビット・レート参照）に対して符号量制御（フレーム当りのビット・レートは可変）。
・サンプリング周波数は 8 kHz ～ 96 kHz（表 1-1　サンプリング周波数参照）。
・モノラル（1 チャネル）時，1024 サンプル／フレームの符号化。
・ステレオ（2 チャネル）時，2048 サンプル／フレームの符号化。
・圧縮データ・フォーマットは，ADTS, ADIF, RAW フォーマット。
・ショート・ブロック処理，TNS 処理，インテンシティ・ステレオ処理には未対応。

1.4.2 機　能

1 フレーム分の 16 ビット･リニア PCM データを，圧縮データに変換します。

1.4.3 動作環境

(1) 動作対象 DSP

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

(2) 必要メモリ・サイズ：

μSAP77016-B17 は表 1-3のメモリ・サイズを使用します。

表 1-3　必要メモリ・サイズ

メモリ	種　別		サイズ［Kword］
命令メモリ	－		4.8
X メモリ	RAM	スクラッチ領域	4.7
		スタティック領域	2.0
	ROM		3.4
Y メモリ	RAM	スクラッチ領域	4.0
		スタティック領域	0.1
		ライブラリ領域	0.1
	ROM		2.8

注意　ライブラリで使用する，X メモリおよび Y メモリ領域は，内蔵 ROM/RAM 空間に配置してください。
必要メモリ・サイズには，オーディオ・データおよびビット・ストリーム・データのバッファは含まれていません。2. 5. 3　入出力バッファを参照してください。

備考　命令メモリの 1 ワードは，32 ビットです。
X メモリ，Y メモリの 1 ワードは，16 ビットです。

(3) 必要 A/D コンバータ・スペック

2 チャネル，16 ビット分解能，表 1-1　サンプリング周波数で示したサンプリング周波数

(4) ソフトウエア・ツール（Windows®版）

表 1-4　ソフトウエア・ツール

対象 DSP	ソフトウエア・ツール
μPD77110 ファミリ	WB77016（ワークベンチ（アセンブラ／リンカ））
	HSM77016（ハイスピード・シミュレータ）
	ID77016（ディバッガ）
μPD77210 ファミリ	Atair Developer Studio（ワークベンチ（アセンブラ／リンカ））
	μPD7721x ハイスピード・シミュレータ
	μPD7721x ディバッガ

備考　これらの DSP 用ソフトウエア・ツールは Atair 社製です。

1.4.4 性　能

1 フレームの処理をリアルタイムに実行するために必要な MIPS 値（実測値）を表 1-5に示します。

・測定条件

シミュレータ：HSM77016（μPD77016 ハイスピード・シミュレータ）

サンプリング周波数：32 kHz, 44.1 kHz, 48 kHz

評価結果：ステレオ／モノラルのオーディオ・ファイルをそれぞれ圧縮したときの演算量を測定し，その平均値および最大値を求めます。

圧縮は，AACENC_Encode 関数のみの演算量になります。その他の関数および割り込みハンドラなどの演算量は含みません。

表 1-5　1 フレーム圧縮処理の MIPS 値（実測値）

サンプリング周波数 [kHz]		32			44.1			48		
設定ビット・レート [kbps]		64	96	128	64	96	128	64	96	128
平均値 [MIPS]	ステレオ	24.9	27.9	28.4	30.7	33.5	37.5	33.5	36.7	41.2
	モノラル	13.4	13.7	14.8	17.9	18.2	18.8	19.1	19.7	20.7
最大値 [MIPS]	ステレオ	42.2	49.8	51.2	50.3	58.5	67.7	54.6	63.8	73.8
	モノラル	24.9	26.9	28.5	33.1	34.6	36.4	35.3	36.5	39.5

備考　この MIPS 値は，当社評価時の実測値です。最大値は最悪値を保証するものではありません。

1.4.5 ディレクトリ構成

μSAP77016-B17 のパッケージ内容を示します。

各ディレクトリの概要は次のとおりです。

- library

  ライブラリ・ファイルを格納しています。

- smp/aacenc

  サンプル・プログラムのソース・ファイル，ヘッダ・ファイルおよびタイミング・ファイルを格納しています。

第2章 ライブラリ仕様

2.1 ライブラリ概要

μSAP77016-B17 では，表 2-1の 3 つの関数を用意しています。

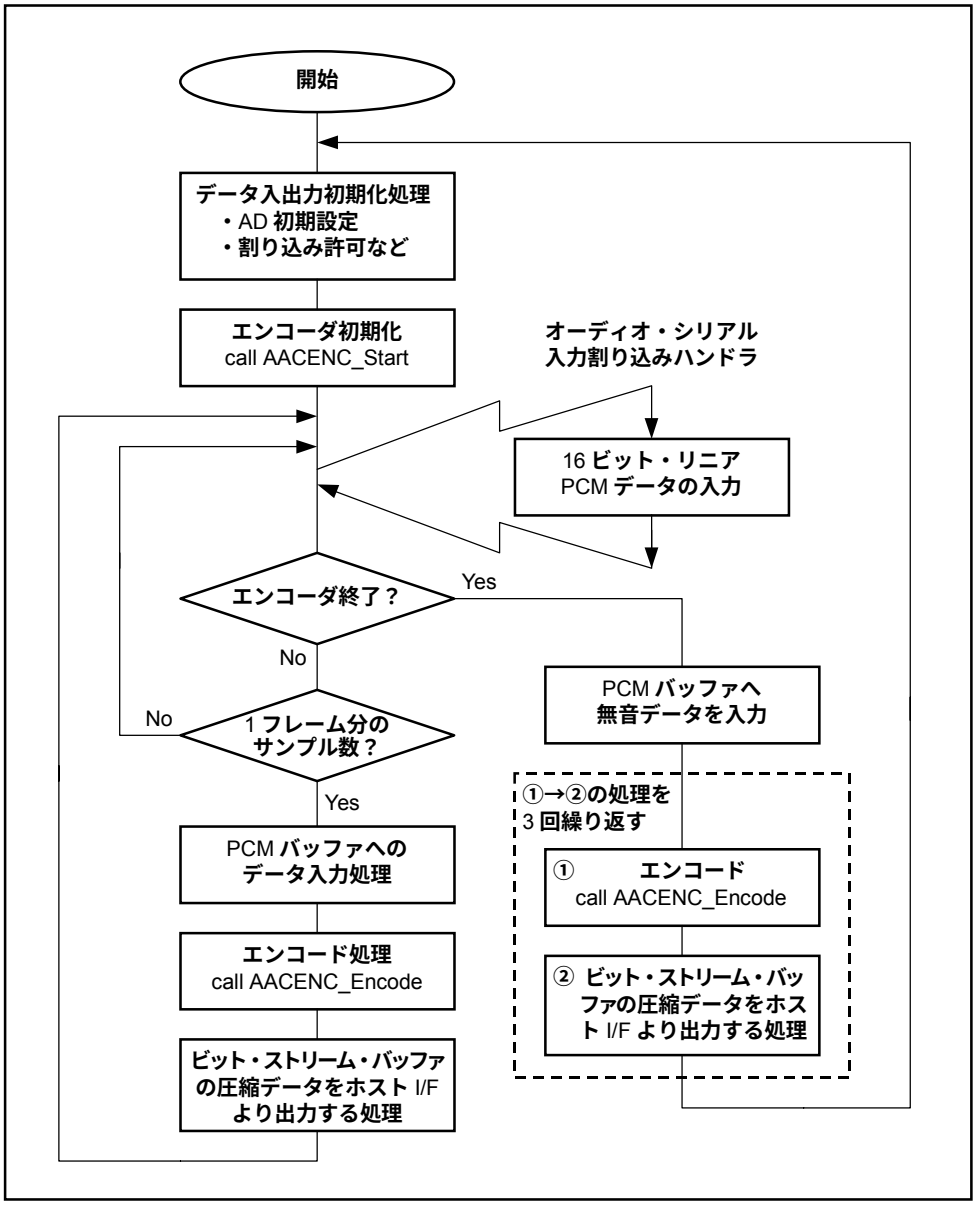
表 2-1 ライブラリ関数一覧

関 数 名	機　　能
AACENC_Start	初期化
AACENC_Encode	圧縮処理
AACENC_GetVersion	バージョン情報取得

2. 2　アプリケーション処理フロー

μSAP77016-B17 を使用したアプリケーションの処理の例をに示します。

図 2-1　アプリケーション処理フロー



備考　割り込みハンドラの 16 ビット・リニア PCM データ入力処理は，ターゲット・システムのハードウエアに依存する処理です。このため，入力処理部については，ユーザがターゲット・システムに合わせて設計してください。

2. 3　関数仕様

各ライブラリ関数を呼び出す際の仕様を次に示します。

2. 3. 1　AACENC_Start 関数

AACENC_Start 関数は，エンコーダが使用する各種パラメータを初期化します。また，ADIF フォーマットのときのみ，ADIF 形式の先頭ヘッダをビット・ストリーム・バッファに出力します。AACENC_Encode 関数を使用する前に，1 度だけ呼び出してください。

【　分　類　】AAC エンコーダ初期化処理

【 関 数 名 】AACENC_Start

【　機　能　】μSAP77016-B17 が使用する各種パラメータ（ビット・レート，オーディオ・チャネル数，サンプリング周波数，圧縮フォーマット，圧縮方式，μSAP77016-B17 が使用するバッファ・ポインタ）を初期化します。

圧縮フォーマットが ADIF のときのみは，ADIF 形式の先頭ヘッダをビット・ストリーム・バッファに出力します。

【　形　式　】call　AACENC_Start

【　引　数　】R0L: X メモリの圧縮に必要なパラメータ群からなる構造体の先頭アドレス[注1]。

R1L: X メモリのエンコーダの出力バッファ（ビット・ストリーム・バッファ）の先頭アドレス[注2]。

ADIF フォーマットのエンコードのときのみ，このビット・ストリーム・バッファに，ADIF ヘッダを出力します。

R2L: X メモリのスタティック領域の先頭アドレス[注2]。

R3L: X メモリのスクラッチ領域の先頭アドレス[注2]。

R4L: Y メモリのスタティック領域の先頭アドレス[注2]。

R5L: Y メモリのスクラッチ領域の先頭アドレス[注2]。

【 返 却 値 】R0L　0 以上とき：圧縮されたビット・ストリームのサイズ（バイト数）。

負のとき：エラー。

次の条件のときに，エラーを返します。

- 圧縮に必要なパラメータ群からなる構造体メンバの speaker_config に 0, 1 以外の値を設定したとき[注1]。
- 圧縮に必要なパラメータ群からなる構造体メンバの sampling_freq の上位 1 ワードおよび下位 1 ワードに，μSAP77016-B17 が対応したサンプリング周波数以外の値を設定したとき[注1]。

【使用レジスタ】r0, r1, r2, r3, r5, dp0, dp4

【ハードウエア・リソースメント】

最大スタック・レベル：	2
最大ループ・スタック・レベル：	1
最大リピート回数：	7
最大サイクル数：	13937

注 1. 圧縮に必要なパラメータ群については，2. 4　圧縮に必要なパラメータ群を参照してください。

2. メモリ領域や入出力バッファについては，2. 5　メモリ構造を参照してください。

注意　この関数を呼び出す前に，メモリ領域を確保しておいてください。

2. 3. 2　AACENC_Encode 関数

AACENC_Encode 関数は，16 ビット×2048 サンプル（ステレオ），または 16 ビット×1024 サンプル（モノラル）のオーディオ・データを指定したビット・レートで圧縮します。

【　分　類　】AAC エンコード処理

【 関 数 名 】AACENC_Encode

【　機　能　】PCM バッファ内の 16 ビット・リニア PCM データを圧縮したあと，圧縮データをビット・ストリーム・バッファに出力します。

【　形　式　】call　AACENC_Encode

【　引　数　】R0L: X メモリの圧縮に必要なパラメータ群からなる構造体の先頭アドレス[注1]。

R1L: X メモリのエンコーダへの入力バッファ（PCM バッファ）の先頭アドレス[注1]。

R2L: X メモリのエンコーダからの出力バッファ（ビット・ストリーム・バッファ）の先頭アドレス[注1]。

R3L: X メモリのスタティック領域の先頭アドレス[注2]。

R4L: Y メモリのスタティック領域の先頭アドレス[注2]。

【 返 却 値 】R0L　0 以上のとき：圧縮されたビット・ストリームのサイズ（バイト数）。

負のとき：エラー。

AACENC_Start 関数がエラーを返したあとに AACENC_Encode 関数を呼び出すと，AACENC_Encode 関数もエラーを返します。

【使用レジスタ】r0, r1, r2, r3, r4, r5,
dp0, dp1, dp2, dp3, dp4, dp5, dp6, dp7,
dn0, dn1, dn2, dn3, dn4, dn5, dn6, dn7,
dmx, dmy

【ハードウエア・リソースメント】

最大スタック・レベル	6
最大ループ・スタック・レベル	3
最大リピート回数	51
最大 MIPS 値	73.8 MIPS（48 kHz サンプリング，128 kbps，ステレオ）

注 1. 圧縮に必要なパラメータ群については，2. 4　圧縮に必要なパラメータ群を参照してください。

2. メモリ領域や入出力バッファについては，2. 5　メモリ構造を参照してください。

注意　エンコード終了時，ユーザは PCM バッファに無音データを 1 フレーム（2048 ワード（ステレオ）／1024 ワード（モノラル））分入力して，AACENC_Encode 関数を 3 回呼び出してください。これにより，次のオーディオ・データの圧縮時に，前のオーディオ・データの情報を引きずることを防止できます。

2. 3. 3　AACENC_GetVersion 関数

AACENC_GetVersion 関数は，μSAP77016-B17 のバージョン情報を返します。

【　分　類　】バージョン情報取得

【 関 数 名 】AACENC_GetVersion

【　機　能　】μSAP77016-B17 のバージョン番号を 32 ビットの値で返します。

例　R0=0x00’0x0001’0x0100 の場合，バージョン：V1.01

【　形　式　】call　AACENC_GetVersion

【　引　数　】なし

【 返 却 値 】R0H: メジャー・バージョン番号

R0L: マイナ・バージョン番号

【使用レジスタ】r0

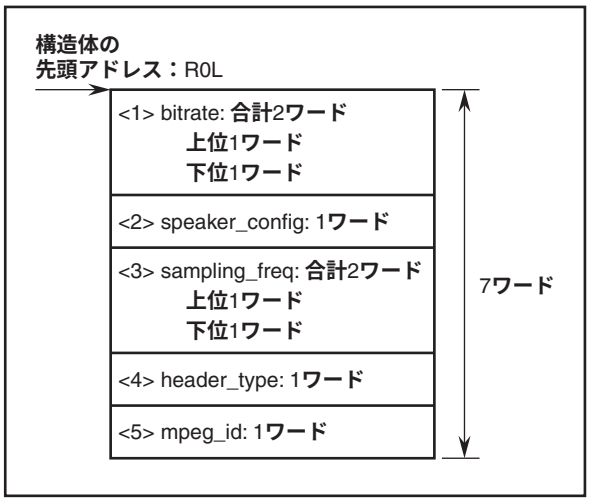
【ハードウエア・リソースメント】

最大スタック・レベル	0
最大ループ・スタック・レベル	0
最大リピート回数	0
最大サイクル数：	6

2. 4　圧縮に必要なパラメータ群

圧縮に必要なパラメータ群からなる構造体（図 2-2参照）を X メモリに確保してください。AACENC_Start 関数および，AACENC_Encoder 関数を呼び出す前に，構造体の各パラメータ情報を設定してください。

図 2-2　圧縮に必要なパラメータ群からなる構造体



<1>　bitrate（2 ワード）：ビット・レート（bps）を設定します。

例　ビット・レートを 264600bps（0t264600 = 0x40998）としたいときは，上位 1 ワードに 0x0004 を，下位 1 ワードに 0x0998 を設定します。

<2>　speaker_config（1 ワード）：オーディオ・チャネル数を設定します。0 または 1 以外の値を設定すると，AACENC_Start 関数はエラーを返します。

0: モノラル（1ch）

1: ステレオ（2ch）

例　ステレオ（2ch）にしたいときは，speaker_config に 0x0001 を設定します。

<3> sampling_freq（2 ワード）：サンプリング周波数（Hz）を指定します。指定できる周波数を表 2-2に示します。

表 2-2にない周波数を設定すると，AACENC_Start 関数はエラーを返します。

表 2-2　サンプリング周波数

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

例 ビット・レートを 96000 Hz（0t96000 = 0x17700）にしたいときは，sampling_freq の上位 1 ワードに 0x0001 を，下位 1 ワードに 0x7700 を設定します。

<4> header_type（1 ワード）：ヘッダ・タイプを指定します。指定できるタイプは次のとおりです。

0: RAW フォーマット（ヘッダなし）

1: ADIF フォーマット

2: ADTS フォーマット

例 ADTS フォーマットにしたいときは，header_type に 0x0002 を設定します。

<5> mpeg_id（1 ワード）：圧縮方式を指定します。μSAP77016-B17 では圧縮方式 MPEG-4/AAC には対応していません。このため，mpeg_id には必ず 0x0001 を設定してください。

0: 設定禁止（MPEG-4/AAC）

1: MPEG-2/AAC

2. 5　メモリ構造

μSAP77016-B17 では，処理に必要なメモリ領域と入出力バッファの領域をユーザが定義する必要があります。

スクラッチ・メモリ領域とスタティック・メモリ領域は別々に定義する必要があります。それぞれのメモリ・サイズについては，表 2-3を参照してください。

表 2-3　シンボル名／メモリ・サイズ

シンボル名	サイズ［ワード］	X/Y 面	説　明
scratch_x_area	4756	X	スクラッチ領域
scratch_y_area	4096	Y	スクラッチ領域
static_x_area	2048	X	スタティック領域
static_y_area	5	Y	スタティック領域

注意　ライブラリで使用する，X メモリおよび Y メモリ領域は，内蔵 ROM/RAM 空間に配置してください。スクラッチ・メモリ領域とスタティック・メモリ領域のサイズには，オーディオ・データおよびビット・ストリーム・データのバッファは含まれていません。2. 5. 3　入出力バッファを参照してください。

2. 5. 1　スクラッチ領域

スクラッチ領域とは，μSAP77016-B17 が使用していないとき，破棄可能なメモリ領域です。

1 フレームのエンコード処理のあと，ユーザはスクラッチ領域を自由に使用できます。

ただし，μSAP77016-B17 が再びこの領域を使用したとき，スクラッチ領域にユーザが設定していた情報は保証されません。

例
```
LIB_SCRATCH_X   xramseg
scratch_x_area:     ds   4756

LIB_SCRATCH_Y   yramseg
scratch_y_area:     ds   4096
```

2. 5. 2　スタティック領域

スタティック領域とは，μSAP77016-B17 が動作していないときにおいても，破棄することができないメモリ領域です。ユーザがスタティック領域を使用することはできません。

初期化処理以降にユーザがこの領域を操作した場合，μSAP77016-B17 の正常な動作を保証できません。

例
```
LIB_STATIC_X    xramseg
static_x_area:      ds    2048

LIB_STATIC_Y    yramseg
static_y_area:      ds    5
```

2. 5. 3　入出力バッファ

入出力バッファは，オーディオ・データ（16 ビット・リニア PCM データ）の入力およびビット・ストリーム・データの出力用の領域です。ユーザは，入出力バッファ用の領域を X メモリに確保してください。

1 フレームのエンコード処理のあと，ユーザは入出力バッファ用の領域を自由に使用できます。

エンコード処理中にオーディオ・データおよび，ビット・ストリーム・データの領域を操作した場合，正常な動作を保証できません。

例　必要な入出力バッファ

```
MAIN_X_WORK        xramseg
    sPCM:                   ds      2048
    __BitstreamBuffer:      ds      768
```

図 2-3　ユーザ定義の入力バッファ（PCM バッファ）

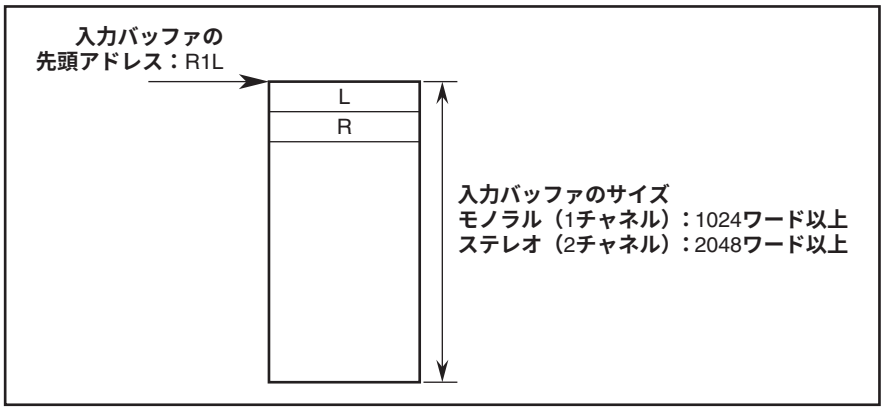
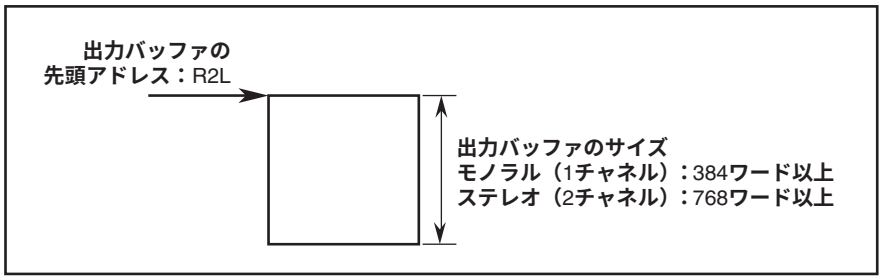


図 2-4　ユーザ定義の出力バッファ（ビット・ストリーム・バッファ）



注意　モノラル（チャネル数が 1）のときは，PCM バッファのサイズを 1024 ワード以上にしてください。また，ビット・ストリーム・バッファを 384 ワード以上確保してください。
ステレオ（チャネル数が 2）のときは，PCM バッファのサイズを 2048 ワード以上にしてください。また，ビット・ストリーム・バッファを 768 ワード以上確保してください。
必要量以上のバッファを確保していない場合，出力結果を保証できません。

第3章 インストレーション

3.1 インストレーション手順

μSAP77016-B17（AAC オーディオ・エンコーダ・ミドルウエア）の提供媒体は，CD-ROM です。ホスト・マシンへのインストール手順を次に示します。

① 提供媒体を CD-ROM ドライブにセットします。DSP ツール（WB77016，HSM77016 など）を使用しているディレクトリ（例：C:¥DSPTools）の下にファイルをコピーします。ここでは，Q ドライブから C ドライブへコピーした場合を示します。

```
Q:¥>xcopy /s *.* C:¥DSPTools<CR>
```

② ファイルがコピーされたことを確認します。各ディレクトリについては，1. 4. 5 ディレクトリ構成を参照してください。

```
C:¥>dir C:¥DSPTools<CR>
```

3. 2　サンプル作成手順

提供媒体の smp ディレクトリに，サンプル・プログラムを格納しています。

サンプル・プログラムは，ハイスピード・シミュレータ（HSM77016 Ver.2.32 以降）上で動作します。タイミング・ファイルを使用することで，データの入出力をシミュレーション可能にします。

タイミング・ファイルについては，**付　録　サンプル・プログラム・ソース**を参照してください。

次にμSAP77016-B17 のサンプル・プログラムのビルド方法について例を示します。

① ワークベンチ（WB77016 Ver.2.4 以降）を起動します。

② sample.prj プロジェクトを開きます。

　例　「Project → Open Project」で sample.prj を指定します。

③ ビルドを実行し，sample.lnk が生成されたことを確認します。

　例　「Make → Build All」を選択すると，sample.lnk ファイルが生成されます。

④ ハイスピード・シミュレータ（HSM77016 Ver.2.32 以降）を起動します。

⑤ sample.lnk を開きます。

　例　「file → open」で sample.lnk を指定します。

⑥ タイミング・ファイル（value.tmg, pcm_in.tmg, stream_out.tmg）を開きます。

　例　「file → open」で value.tmg を指定します。

3. 3　シンボル命名規約

μSAP77016-B17 で使用するシンボルの命名規約を表 3-1 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 3-1　シンボル命名規約

分　類	命名規則
関数名，コード・セグメント名（IMSEG）， 定数セグメント名（ROMSEG/RAMSEG）， 定数名，変数領域名	AACENC_XXXX

備考　XXXX は任意の英数字とします。

第4章 システム例

4.1 タイミング・ファイルを使用したシミュレーション環境

μSAP77016-B17 の圧縮処理のシミュレーション環境を次に示します。オーディオ・データ（16 ビット・リニア PCM データ）を入力し，1 フレーム単位で圧縮処理をしながら，圧縮データを出力します。

【ソフトウエア環境】

- ハイスピード・シミュレータ　：HSM77016 Ver.2.32 以降
- サンプル・プログラム　　：sample.lnk（3. 2　サンプル作成手順で作成したもの）
- タイミング・ファイル　　：value.tmg, pcm_in.tmg, stream_out.tmg

4.2 操作方法

① ハイスピード・シミュレータを起動します。

② 3. 2　サンプル作成手順で作成した sample.lnk を開きます。

　例　「file → open」で sample.lnk を指定します。

③ タイミング・ファイル（value.tmg, pcm_in.tmg, stream_out.tmg）を開きます。

　例　「file → open」で value.tmg を指定します。

④ Run で実行します。

付　録　サンプル・プログラム・ソース

付録.1　ヘッダ・ファイル（aacenc.h）

```
#ifndef   __aacenc_h
#define   __aacenc_h

#define   AACENC_STATIC_AREA_X_SIZE     2048
#define   AACENC_STATIC_AREA_Y_SIZE     5
#define   AACENC_SCRATCH_AREA_X_SIZE    4756
#define   AACENC_SCRATCH_AREA_Y_SIZE    4096

#define   HeaderSize         7

extrn     AACENC_Start
extrn     AACENC_Encode
extrn     AACENC_GetVersion

#endif
```

付録.2　サンプル・プログラム用インクルード・ファイル（sample.inc）

（1/2）

```
VECTOR  imseg at 0x200

%define (JmpVect(addr))
(
        jmp addr            ;
        nop                 ;
        nop                 ;
        nop                 ;
)

%define (NopVect)
(
        nop                 ;
        reti                ;
        nop                 ;
        nop                 ;
)

ivReset:
        %JmpVect(StartUp)   ;
        %NopVect            ;
        %NopVect            ;
        %NopVect            ;
ivINT1:
        %NopVect            ;
ivINT2:
        %NopVect            ;
ivINT3:
        %NopVect            ;
ivINT4:
        %NopVect            ;
ivINT5:
        %NopVect            ;
ivINT6:
        %NopVect            ;
ivINT7:
        %NopVect            ;
ivINT8:
        %NopVect            ;
ivINT9:
        %NopVect            ;
ivINT10:
        %NopVect            ;
ivINT11:
        %NopVect            ;
ivINT12:
        %NopVect            ;
```

(2/2)

```
%define (Initialize)
(
        clr (r0)                ;
        r1 = r0                 ;
        r2 = r0                 ;
        r3 = r0                 ;
        r4 = r0                 ;
        r5 = r0                 ;
        r6 = r0                 ;
        r7 = r0                 ;
        dp0 = r0l               ;
        dp1 = r0l               ;
        dp2 = r0l               ;
        dp3 = r0l               ;
        dp4 = r0l               ;
        dp5 = r0l               ;
        dp6 = r0l               ;
        dp7 = r0l               ;
        dn0 = r0l               ;
        dn1 = r0l               ;
        dn2 = r0l               ;
        dn3 = r0l               ;
        dn4 = r0l               ;
        dn5 = r0l               ;
        dn6 = r0l               ;
        dn7 = r0l               ;
        dmx = r0l               ;
        dmy = r0l               ;
)
```

付録.3　サンプル・ソース・ファイル（sample.asm）

（1/5）

```
#include "aacenc.h"

LIB_STATIC_X     xramseg
      static_x_area:              ds    AACENC_STATIC_AREA_X_SIZE

LIB_STATIC_Y     yramseg
      static_y_area:              ds    AACENC_STATIC_AREA_Y_SIZE

LIB_SCRATCH_X    xramseg
      scratch_x_area:             ds    AACENC_SCRATCH_AREA_X_SIZE

LIB_SCRATCH_Y    yramseg
      scratch_y_area:             ds    AACENC_SCRATCH_AREA_Y_SIZE

MAIN_X_WORKxramseg at 0x6000
      sPCM:                       ds    1024*2
      __BitstreamBuffer:          ds    2048/2

      Header:
      _Header_bitrate_H:          ds    1
      _Header_bitrate_L:          ds    1
      _Header_speaker_config:     ds    1
      _Header_sampling_freq_H:    ds    1
      _Header_sampling_freq_L:    ds    1
      _Header_header_type:        ds    1
      _Header_mpeg_id:            ds    1
      _OutputBufferFlag:          ds    1
      _valued:                    ds    1
      _frame_number:              ds    1

MAIN_CTRL_WORK   xramseg at 0x5ffc
      _s:                         ds    1      ; タイミング・ファイルで参照
      _stream_length:             ds    1      ; タイミング・ファイルで参照
      _FW_out_flag:               ds    1      ; タイミング・ファイルで参照
      _FW_run_flag:               ds    1      ; タイミング・ファイルで参照

#define  Header_bitrate_H        _Header_bitrate_H:x
#define  Header_bitrate_L        _Header_bitrate_L:x
#define  Header_speaker_config   _Header_speaker_config:x
#define  Header_sampling_freq_H  _Header_sampling_freq_H:x
#define  Header_sampling_freq_L  _Header_sampling_freq_L:x
#define  Header_header_type      _Header_header_type:x
#define  Header_mpeg_id          _Header_mpeg_id:x
#define  OutputBufferFlag        _OutputBufferFlag:x
#define  valued                  _valued:x
#define  frame_number            _frame_number:x

#define  s                       _s:x
#define  stream_length           _stream_length:x
#define  FW_out_flag             _FW_out_flag:x
#define  FW_run_flag             _FW_run_flag:x

#include " sample.inc"
```

(2/5)

```
MAIN  imseg at 0x240

StartUp:
      %Initialize                         ;

      clr (r0)                            ;
      dp0 = __BitstreamBuffer             ;
      rep 2048/2                          ;
                  *dp0++ = r0h            ;
      dp0 = sPCM                          ;
      rep 1024*2                          ;
                  *dp0++ = r0h            ;

      dp4 = 0x4000                        ;
      rep 0x2000                          ;
                  *dp4++ = r0h            ;

      *OutputBufferFlag = r0h             ;
      *frame_number = r0h                 ;
      *valued = r0h                       ;
      *stream_length = r0h                ;

      clr (r0)                            ;run_flag = 0
      *FW_run_flag = r0h                  ;
      r0 = *FW_run_flag                   ;while (run_flag == 0) {}
      if (r0 == 0) jmp $-1                ;

      ;; header解析 ;; (MSB) ;; 22+2+4+6+2+8 = 44byte -> 22word
      dp0 = sPCM                          ;
      rep 22/2                            ;fread (header, 1, 22, fi)
                  r0 = *dp0++             ;
                  r0 = *dp0++             ;fread (&s, 1, 2, fi)
      *s = r0h                            ;
      r1 = r0 sra 16                      ;Header.speaker_config = (s == 1) ? 0 : 1
      r1 = r1 - 1                         ;
      clr (r0)                            ;
      if (r1 != 0) r0 = r0 + 1            ;
      *Header_speaker_config = r0l        ;
                  r0 = *dp0++     ;fread (&i, 1, 4, fi)
                  r0l = *dp0++    ;
      *Header_sampling_freq_H = r0l       ;Header.sampling_freq = i
      *Header_sampling_freq_L = r0h       ;
                                          ;fread (tmp, 1, 6+2+8, fi)

      clr (r0)                            ;
      r0l = 64000                         ;64kbps
```

(3/5)

```
nop;
      *Header_bitrate_H = r0h            ;
      *Header_bitrate_L = r0l            ;
      r0l = 1                            ;MPEG-2 AAC
      *Header_mpeg_id = r0l              ;
      r0l = 2                            ;header_type = 2
      *Header_header_type = r0l          ;
      r0l = Header                       ;ret = AACENC_Start (&Header, BitstreamBuffer)
      r1l = __BitstreamBuffer            ;

      r2l = static_x_area                ;
      r3l = scratch_x_area               ;
      r4l = static_y_area                ;
      r5l = scratch_y_area               ;

      call AACENC_Start                  ;
      if (r0 < 0) jmp _exit              ;
      call CopyOutputBuffer              ;CopyOutputBuffer (ret, BitstreamBuffer)

_while1:                                 ;while (1) {
      clr (r0)                           ;   run_flag = 0
      *FW_run_flag = r0h                 ;
      r0 = *FW_run_flag                  ;   while (run_flag == 0) {}
      if (r0 == 0) jmp $-1               ;
      if (r0 < 0) jmp _break1            ;   if (run_flag < 0) break

      r0l = Header                       ;   ret = AACENC_Encode (&Header, sPCM, BitstreamBuffer)
      r1l = sPCM                         ;
      r2l = __BitstreamBuffer            ;

      r3l = static_x_area                ;
      r4l = static_y_area                ;

      call AACENC_Encode                 ;
      if (r0 < 0) jmp _exit              ;   if (ret < 0) exit(ret)
      call CopyOutputBuffer              ;   CopyOutputBuffer (ret, BitstreamBuffer)

      r1l = *frame_number                ;   frame_numfer++
      r1 = r1 + 1                        ;
      *frame_number = r1l                ;

      jmp _while1                        ;}
```

(4/5)

```
_break1:
      dp0 = sPCM                          ;sPCM領域の0クリア
      clr (r0)                            ;for (i = 0; i < 2048; i++) {
      rep 2048                            ;   sPCM[i] = 0
              *dp0++ = r0h                ;}

      r0l = Header                        ;ret = AACENC_Encode (&Header, sPCM, BitstreamBuffer)
      r1l = sPCM                          ;
      r2l = __BitstreamBuffer             ;

      r3l = static_x_area                 ;
      r4l = static_y_area                 ;

      call AACENC_Encode                  ;
      if (r0 < 0) jmp _exit               ;if (ret < 0) exit(ret)
      call CopyOutputBuffer               ;CopyOutputBuffer (ret, BitstreamBuffer)

      r0l = Header                        ;ret = AACENC_Encode (&Header, sPCM, BitstreamBuffer)
      r1l = sPCM                          ;
      r2l = __BitstreamBuffer             ;

      r3l = static_x_area                 ;
      r4l = static_y_area                 ;

      call AACENC_Encode                  ;
      if (r0 < 0) jmp _exit               ;if (ret < 0) exit(ret)
      call CopyOutputBuffer               ;CopyOutputBuffer (ret, BitstreamBuffer)

      r0l = Header                        ;ret = AACENC_Encode (&Header, sPCM, BitstreamBuffer)
      r1l = sPCM                          ;
      r2l = __BitstreamBuffer             ;

      r3l = static_x_area                 ;
      r4l = static_y_area                 ;

      call AACENC_Encode                  ;
      if (r0 < 0) jmp _exit               ;if (ret < 0) exit(ret)
      call CopyOutputBuffer               ;CopyOutputBuffer (ret, BitstreamBuffer)

      r0 = *OutputBufferFlag              ;
      if (r0 == 0) jmp _exit              ;
              r0 = *valued                ;
              *0x4000:y = r0h             ;
              r0l = 1                     ;
              *stream_length = r0l        ;

              clr (r1)                    ;
              *FW_out_flag = r1h          ;
              r1 = *FW_out_flag           ;
              if (r1 == 0) jmp $-1        ;
_exit:
      clr (r0)                            ;
      r0l = 0x2222                        ;length = 0x2222
      *stream_length = r0l                ;
      *FW_out_flag = r0h                  ;out_flag = 0
      *FW_run_flag = r0h                  ;run_flag = 0
      jmp StartUp                         ;
```

(5/5)

```
;; BitstreamBufferから出力バッファへコピーする ;;
CopyOutputBuffer:
      if (r0 == 0) ret                    ;
      dp4 = 0x4000                        ;wpt = 0x4000
      dp0 = __BitstreamBuffer             ;rpt = BitstreamBuffer
      r2 = *OutputBufferFlag              ;if (OutputBufferFlag == 0) {
      if (r2 != 0) jmp _L1                ;

      r1 = r0 sra 1                       ;   count = len >> 1
      *stream_length = r1l                ;
      r2 = r0 & 1                         ;
      loop r1l {                          ;   for ( ; count > 0; count--) {
                  r1 = *dp0++     ;           tmp = *rpt++
                  *dp4++ = r1h    ;           *wpt++ = tmp
      }                                   ;   }
      *OutputBufferFlag = r2l             ;
      if (r2 == 0) jmp _L2                ;   if (len & 1) {
                  r1 = *dp0               ;           valued = *rpt
                  *valued = r1h           ;   }

      jmp _L2                             ;}
_L1:                                      ;else {
      r1 = r0 sra 1                       ;   count = len >> 1
      r2 = r0 & 1                         ;   if (len & 1)
      if (r2 != 0) r1 = r1 + 1            ;           count++
      *stream_length = r1l                ;
      r0 = *valued                        ;   tmp = valued << 8
      r0 = r0 sra 8                       ;
      loop r1l {                          ;   for ( ; count > 0; count--) {
                  r0l = *dp0++            ;           tmp |= *rpt++
                  r0 = r0 sll 8           ;           tmp <<= 8
                  *dp4++ = r0h            ;           *wpt++ = (tmp >> 16)
                  r0 = r0 sll 8           ;           tmp <<= 8
      }                                   ;   }
      r0 = r0 sll 8                       ;
      *valued = r0h                       ;   valued = tmp >> 16
      clr (r1)                            ;   *wpt = valued
      if (r2 == 0) r1 = r1 + 1            ;
      *OutputBufferFlag = r1l             ;

_L2:                                      ;}

      clr (r0)                            ;
      *FW_out_flag = r0l                  ;
      r0 = *FW_out_flag                   ;
      if (r0 == 0) jmp $-1                ;

      ret                                 ;return
end
```

付録.4　パラメータ情報用タイミング・ファイル（value.tmg）

```
global  s, FW_run_flag, FW_out_flag, stream_length, pcm_start, stream_start

global  out44_128, out44_96, out44_80, out44_64
global  out32_96, out32_64, out32_48, out32_40

global  flag01, flag02, flag03, flag04, flag05
global  flag08, flag09, flag12, flag13, flag19
global  flag21

set s             = 0x5ffc
set stream_length = 0x5ffd
set FW_out_flag   = 0x5ffe
set FW_run_flag   = 0x5fff
set pcm_start     = 0x6000
set stream_start  = 0x4000

set out44_128  = 1 << 7
set out44_96   = 1 << 6
set out44_80   = 1 << 5
set out44_64   = 1 << 4
set out32_96   = 1 << 3
set out32_64   = 1 << 2
set out32_48   = 1 << 1
set out32_40   = 1 << 0

set flag01 = 0
set flag02 = 0
set flag03 = 0
set flag04 = 0
set flag05 = 0
set flag08 = 0
set flag09 = 0
set flag12 = 0
set flag13 = 0
set flag19 = 0
set flag21 = 0

set flag01 = out44_128 | out44_96 | out44_80 | out44_64 | out32_96 | out32_64 | out32_48 | out32_40
set flag02 = out44_128 | out44_96 | out44_80 | out44_64 | out32_96 | out32_64 | out32_48 | out32_40
set flag03 = out44_128 | out44_96 | out44_80 | out44_64 | out32_96 | out32_64 | out32_48 | out32_40
set flag04 = out44_128 | out44_96 | out44_80 | out44_64 | out32_96 | out32_64 | out32_48 | out32_40
set flag05 = out44_128 | out44_96 | out44_80 | out44_64 | out32_96 | out32_64 | out32_48 | out32_40
set flag08 = out44_128 | out44_96 | out44_80 | out44_64 | out32_96 | out32_64 | out32_48 | out32_40
set flag09 = out44_128 | out44_96 | out44_80 | out44_64 | out32_96 | out32_64 | out32_48 | out32_40
set flag12 = out44_128 | out44_96 | out44_80 | out44_64 | out32_96 | out32_64 | out32_48 | out32_40
set flag13 = out44_128 | out44_96 | out44_80 | out44_64 | out32_96 | out32_64 | out32_48 | out32_40
set flag19 = out44_128 | out44_96 | out44_80 | out44_64 | out32_96 | out32_64 | out32_48 | out32_40
set flag21 = out44_128 | out44_96 | out44_80 | out44_64 | out32_96 | out32_64 | out32_48 | out32_40

end
```

付録.5　データ入力用タイミング・ファイル（pcm_in.tmg）

（1/2）

```
global  s, FW_run_flag, FW_out_flag, stream_length, pcm_start
global  out44_128, out44_96, out44_80, out44_64
global  out32_96, out32_64, out32_48, out32_40
global  flag01, flag02, flag03, flag04, flag05
global  flag08, flag09, flag12, flag13, flag19
global  flag21
local   size, data, pcm
local   sz
local   bitrate
local   addr

set addr = 0x27c

sub hanten      ; (data)
        set data = ((data >> 8) & 0xff) | ((data & 0xff) << 8)
endsub

sub WriteData
        hanten
        set *pcm:x = data
        set pcm = pcm + 1
endsub

sub Writting    ; (size, sz)
        local   tmp

        set pcm = pcm_start
        if (size >= sz)
                rept sz
                        input data
                        WriteData
                endrept
                set size = size - sz
        else
                set tmp = sz - size
                rept size
                        input data
                        WriteData
                endrept
                set data = 0
                rept tmp
                        WriteData
                endrept
                set size = 0
        endif
endsub
```

(2/2)

```
sub wave2stream
        input format hex
        input size

        set *FW_run_flag:x = 1
        wait cond (*FW_run_flag:x != 1)

        set sz = 22
        Writting

        if (*s:x == 1)
                set sz = 1024
        else
                set sz = 2048
        endif

        set *FW_run_flag:x = 1
        wait cond ip == addr
        set r0 = bitrate

        do
                set *FW_run_flag:x = 1
                wait cond (*FW_run_flag:x != 1)

                exit (size < sz)

                Writting
        enddo

        set *FW_run_flag:x = -1
        wait cond (*FW_run_flag:x != -1)
endsub


;;; 初期化待ち ;;;
wait 1

if (flag02)
        if (flag02 & out32_40)
                ;;; 02.wav / 32kHz / 40kbps ;;;
                open input "02_32.dat"
                set bitrate = 40000
                wave2stream
                close input
        endif
endif

break
end
```

付録.6　データ出力用タイミング・ファイル（stream_out.tmg）

```
global  FW_out_flag, stream_length, stream_start
global  out44_128, out44_96, out44_80, out44_64
global  out32_96, out32_64, out32_48, out32_40
global  flag01, flag02, flag03, flag04, flag05
global  flag08, flag09, flag12, flag13, flag19
global  flag21

sub wave2stream2
        local  data, rpt, length

        output #10 format dec

        do
                set *FW_out_flag:x = 1
                wait cond *FW_out_flag:x != 1

                set length = *stream_length:x

                exit length >= 0x2000

                if (length)
                        set rpt = stream_start
                        rept length
                                set data = *rpt:y
                                output #10 data
                                set rpt = rpt + 1
                        endrept
                endif
        enddo
endsub


;;; main ;;;
wait 1
if (flag02)
        ;;; 02.wav ;;;
        if (flag02 & out32_40)
                open output #10 "02_32_40.dat"
                wave2stream2
                close output #10
        endif
endif

end
```

〔メ　モ〕

〔メ　モ〕

【発　行】

NEC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
電話（代表）：044(435)5111

【ホームページ】

NECエレクトロニクスの情報がインターネットでご覧になれます。
URL(アドレス)　http://www.necel.co.jp/

【営業関係お問い合わせ先】

下記のページに最新版のお問い合わせ先が記載されています。
URL(アドレス)　http://www.necel.com/ja/contact/contact_j.html

【技術的なお問い合わせ先】

半導体テクニカルホットライン
（電話：午前 9:00～12:00，午後 1:00～5:00）

電　話　：044−435−9494
FAX　：044−435−9608
E-mail　：info@lsi.nec.co.jp

【資料請求先】

NECエレクトロニクス特約店または上記ホームページ記載の営業関係お問い合わせ先へお申し付けください。
