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-B19

MP3 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 U16553JJ1V0UM00（第 1 版）
発行年月 February 2003 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2003年1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-B07 ユーザーズ・マニュアル　（MP3 デコーダ）	U15134J
μSAP77016-B19 ユーザーズ・マニュアル　（MP3 エンコーダ）	このマニュアル

規格に関する資料

資 料 名
ISO/IEC 11172-3:1993 Information technology - Coding of moving pictures and associated audio for digital storage Media at up to about 1.5 Mbit/s - Part3
JIS X 4323-1995 日本工業規格 情報技術 - 約 1.5Mbit/s までの，ディジタル記憶媒体のためのビデオおよび付随するオーディオの符号化 - 第 3 部：オーディオ（MR-4284 1995.10.27）
ISO/IEC 13818-3 Information technology - Generic coding of moving pictures and associated audio information - Part 3: Audio
JIS X 4327 動画信号および付随する音響信号の汎用符号化 - 第 3 部 音響

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-B19 は，MP3 エンコード機能を提供するミドルウエアです。

1.2　MP3 オーディオ・エンコーダ

MP3 とは，記憶媒体用の高品質オーディオ信号の符号化および復号方式で，MPEG-1 Audio Layer-3 および MPEG-2 Audio Layer-3 LSF（Low Sampling Frequency）を称します。μSAP77016-B19 は，その符号方式に準拠したものです。

MPEG-1 Audio Layer-3 の圧縮データ・フォーマットは，「ISO/IEC 11172-3」の規格書に準拠しています。また，MPEG-2 Audio Layer-3 LSF の圧縮データ･フォーマットは，「ISO/IEC 13818-3」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，16 kHz～48 kHz（表 1-1参照）のサンプリング周波数で，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

MPEG-1 Audio Layer-3 [Hz]	MPEG-2 Audio Layer-3 LSF [Hz]
32000	16000
44100	22050
48000	24000

1. 2. 1　エンコーダ概略

図 1-1にμSAP77016-B19 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-B19 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

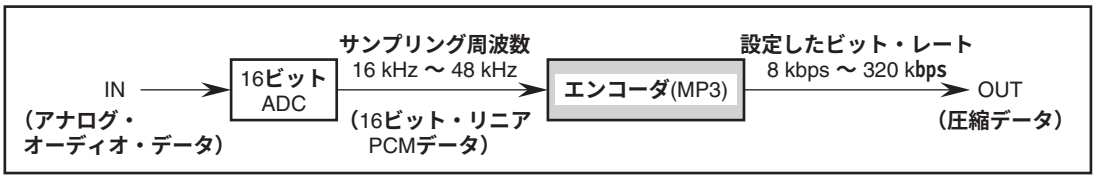
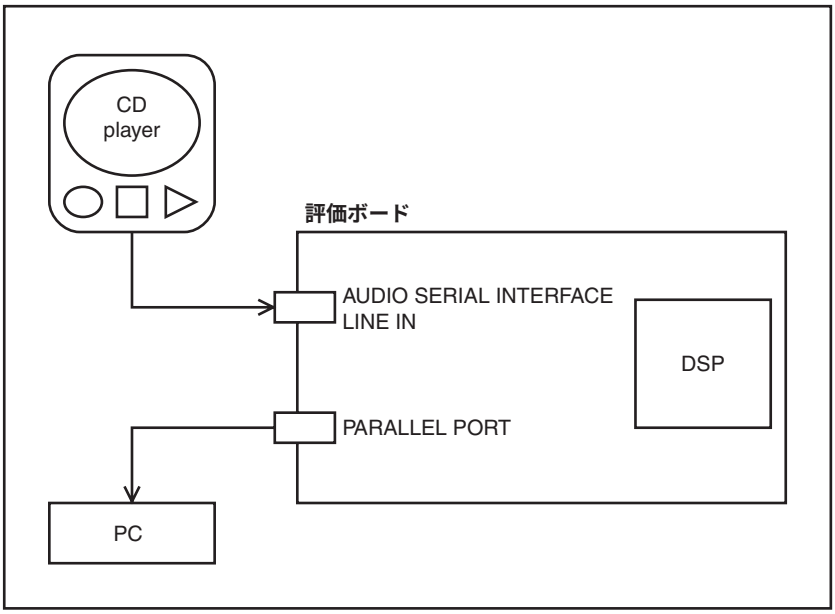


図 1-2　システム構成例



(1) 入力データ

16 kHz ～ 48 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。

1 フレーム分の入力データのワード数を表 1-2に示します。

表 1-2　入力データのワード数

圧縮方式	チャネル数	入力するオーディオ・データのワード数
MPEG-1	1	1152
Audio Layer-3	2	2304
MPEG-2	1	576
Audio Layer-3 LSF	2	1152

備考　1 ワード = 16 ビット

(2) MP3 オーディオ・エンコーダ

MP3 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・レートを表 1-3に示します。

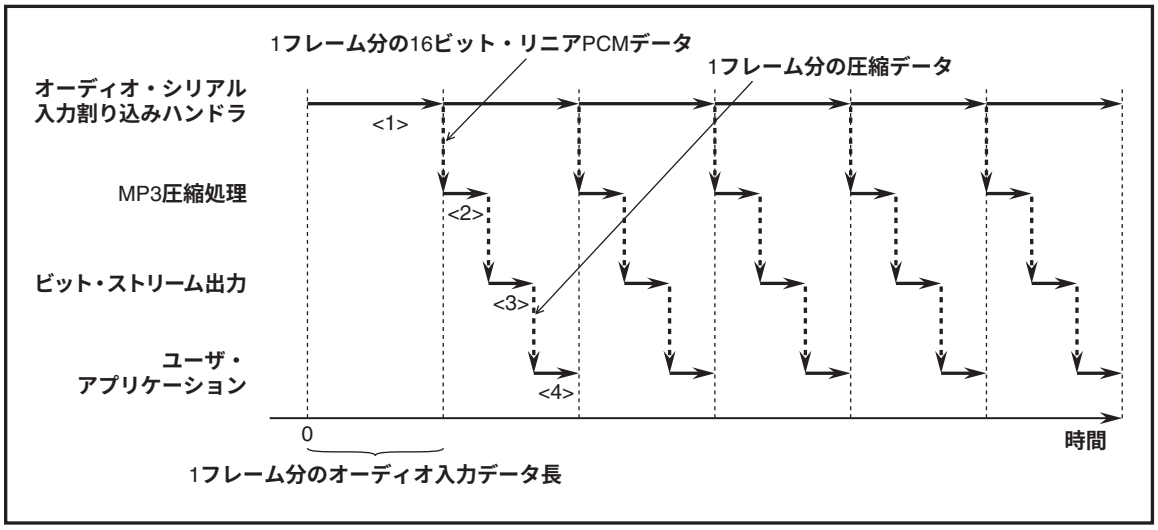
表 1-3　対応ビット・レート

MPEG-1 Audio Layer-3 [kbps]	MPEG-2 Audio Layer-3 LSF [kbps]
32	8
40	16
48	24
56	32
64	40
80	48
96	56
112	64
128	80
160	96
192	112
224	128
256	144
320	160

(3) タイミング・ダイアグラム

MP3 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

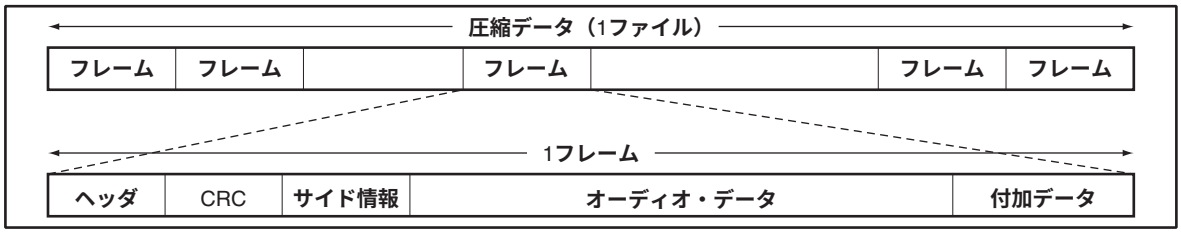
圧縮データ･フォーマットは，ISO/IEC 11172-3，JIS X 4323 の規格書に準拠しています。

ただし，μSAP77016-B19 は，bitrate_index が‘ 0000 ’の場合の自由形式（free format）に対応していません。

圧縮データは，複数のビット・ストリームから構成されており，1 つのビット・ストリームをフレームと呼びます。

圧縮データのフォーマット構造を，図 1-4に示します。

図 1-4　圧縮データ・フォーマット



1.3.1 ヘッダ

ヘッダには，同期を取るためのサンプリング周波数，ビット・レート，モードなどの情報が含まれています。

表 1-4 ヘッダ部の詳細

情報	使用ビット数	値
フレーム・シンク・ワード	12	‘1111 1111 1111’：固定値
ID	1	‘1’：MPEG-1 ‘0’：MPEG-2
レイヤ	2	‘11’：Layer1／‘10’：Layer2／‘01’：Layer3 （μSAP77016-B19 では，'01' 固定です。）
保護ビット	1	‘0’：CRC あり　‘1’：CRC なし （μSAP77016-B19 では，'1' 固定です。）
ビット・レート	4	表 1-5 ビット・レートとビットの値の関係を参照
サンプリング周波数	2	表 1-6 サンプリング周波数とビットの値の関係を参照
パディング・ビット	1	‘1’：フレームに 1 バイト追加あり[注1] ‘0’：フレームに 1 バイト追加なし
プライベート・ビット	1	未使用（μSAP77016-B19 では，'0' 固定です。）
モード	2	‘00’：stereo　‘01’：joint_stereo ‘10’：dual_channel　‘11’：single_channel （μSAP77016-B19 では，ステレオの場合は '01' 固定，モノラルの場合は '11' 固定です。）
モード拡張子[注2]	2	‘00’：is_off, ms_off　‘01’：is_on, ms_off ‘10’：is_off, ms_on　‘11’：is_on, ms_on （μSAP77016-B19 では，'10' 固定です。）
著作権	1	‘0’：no copyright　‘1’：copyright protected （μSAP77016-B19 では，'0' 固定です。）
オリジナル／コピー区別	1	‘0’：copy　‘1’：original （μSAP77016-B19 では，'1' 固定です。）
エンファシス	2	‘00’：no emphasis　‘01’：50/15 μs ‘10’：reserved　‘11’：CCITT J.17 （μSAP77016-B19 では，'00' 固定です。）

注 1. サンプリング周波数 44.1 kHz の場合，フレームの端数調整のために 1 バイトを追加します。

2. is は intensity stereo を，ms は MS stereo を示します。

表 1-5　ビット・レートとビットの値の関係

値	MPEG-1 Audio Layer-3 [kbps]	MPEG-2 Audio Layer-3 LSF [kbps]
‘0000’	free format 注	
‘0001’	32	8
‘0010’	40	16
‘0011’	48	24
‘0100’	56	32
‘0101’	64	40
‘0110’	80	48
‘0111’	96	56
‘1000’	112	64
‘1001’	128	80
‘1010’	160	96
‘1011’	192	112
‘1100’	224	128
‘1101’	256	144
‘1110’	320	160
‘1111’	設定禁止	

注　μSAP77016-B19 のサポート対象外

表 1-6　サンプリング周波数とビットの値の関係

値	MPEG-1 Audio Layer-3 [Hz]	MPEG-2 Audio Layer-3 LSF [Hz]
‘00’	44100	22050
‘01’	48000	24000
‘10’	32000	16000
‘11’	設定禁止	

1. 3. 2　CRC

ヘッダの保護ビットにおいて，CRC が有効を示している場合のみ，2 バイトの情報がヘッダの次に存在します。有効でない場合には，2 バイトの情報は存在しません。

μSAP77016-B19 では，CRC なしで出力されます。

1.3.3 サイド情報

オーディオ・データのデコードに必要な情報として，フレーム上のオーディオ・データの開始位置，デコード方法などの情報を含みます。

サイド情報のサイズは，表 1-7のようになります。

表 1-7　サイド情報のサイズ

	MPEG-1 Audio Layer-3 [byte]	MPEG-2 Audio Layer-3 LSF [byte]
モノラル	17	9
ステレオ	32	17

1.3.4 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。オーディオ・データの開始位置は，サイド情報に設定されています。オーディオ・データは，オーディオ・データの位置を示すサイド情報と同じフレームで始まる場合と，前のフレームから始まる場合があります。

また，オーディオ・データの先頭位置は，1 つ前のフレームとはかぎらず，2 つ前のフレームから始まる場合もあります。

オーディオ・データの詳細については，規格書（ISO/IEC 11172-3）を参照してください。

1.3.5 付加データ

ユーザが定義できるデータを載せる部分です。このデータは，フレームに存在しないこともあります。

μSAP77016-B19 では，このデータを付加しません。

1.4 製品概要

1.4.1 特　徴

・モノラル（Single Channel）／ステレオ（Joint Stereo）に対応。
・16 ビット・リニア PCM データ入力。
・設定ビット・レート（表 1-5　ビット・レートとビットの値の関係参照）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）。可変ビット・レート（VBR：Variable Bit Rate）は未対応。
・サンプリング周波数は，MPEG-1 Audio Layer-3 の 32 k/44.1 k/48 kHz および MPEG-2 Audio Layer-3 LSF の 16 k/22.05 k/24 kHz（表 1-6　サンプリング周波数とビットの値の関係参照）。
・モノラル（1 チャネル）時，MPEG-1 Audio Layer-3 では 1152 サンプル／フレームの符号化，MPEG-2 Audio Layer-3 LSF では 576 サンプル／フレームの符号化。
・ステレオ（2 チャネル）時，MPEG-1 Audio Layer-3 では 2304 サンプル／フレームの符号化，MPEG-2 Audio Layer-3 LSF では 1152 サンプル／フレームの符号化。
・MS stereo に対応，intensity stereo 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-B19 は表 1-8のメモリ・サイズを使用します。

表 1-8 必要メモリ・サイズ

メモリ	種　別		サイズ［Kword］
命令メモリ	−		6.6
X メモリ	RAM	スクラッチ領域	4.9
		スタティック領域	2.3
	ROM		3.1
Y メモリ	RAM	スクラッチ領域	2.5
		スタティック領域	3.2
		ライブラリ領域	0.2
	ROM		0.8

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。

μSAP77016-B19 は，スクラッチ領域とビット・ストリーム・バッファの領域とを共用する仕様になっています。

必要メモリ・サイズには，オーディオ・データの入力バッファ（PCM バッファ）は含まれていません。2.5.3 入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 1-9 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 1-10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）

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，MP3ENC_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 1-10　1 フレーム圧縮処理の MIPS 値（実測値）

(a) MPEG-1 Audio Layer-3

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	44.5	45.0	47.5	49.3	50.0	53.8	51.4	51.6	56.0
	モノラル	22.7	23.0	23.0	25.1	25.9	25.9	26.2	26.9	27.0
最大値［MIPS］	ステレオ	68.0	69.2	70.7	77.3	75.5	79.1	76.2	73.7	84.2
	モノラル	37.6	38.0	38.3	42.4	42.6	40.9	41.3	42.7	44.0

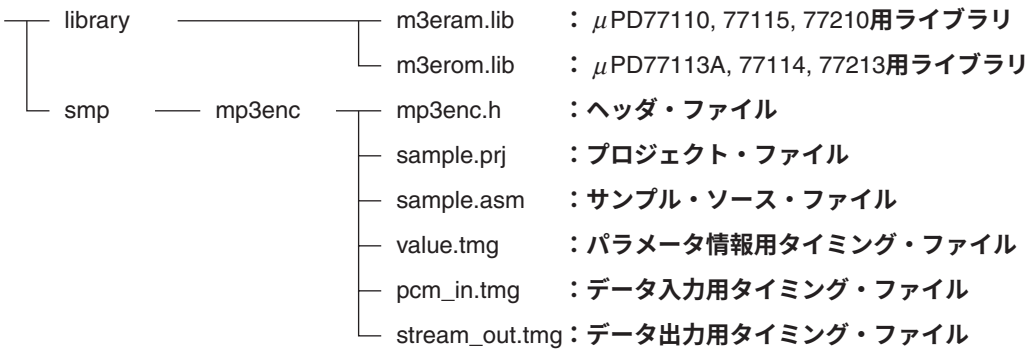
(b) MPEG-2 Audio Layer-3 LSF

サンプリング周波数［kHz］		16			22.05			24		
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	22.6	23.0	23.2	30.8	31.1	31.5	31.7	34.1	34.8
	モノラル	11.2	11.5	11.9	15.3	15.5	15.8	17.1	16.9	17.4
最大値［MIPS］	ステレオ	36.2	37.4	36.8	51.7	48.8	51.9	54.7	53.4	56.5
	モノラル	21.2	20.9	20.7	27.6	28.2	29.0	30.3	33.7	33.6

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-B19 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/mp3enc

  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-B19 では，表 2-1の 4 つの関数を用意しています。

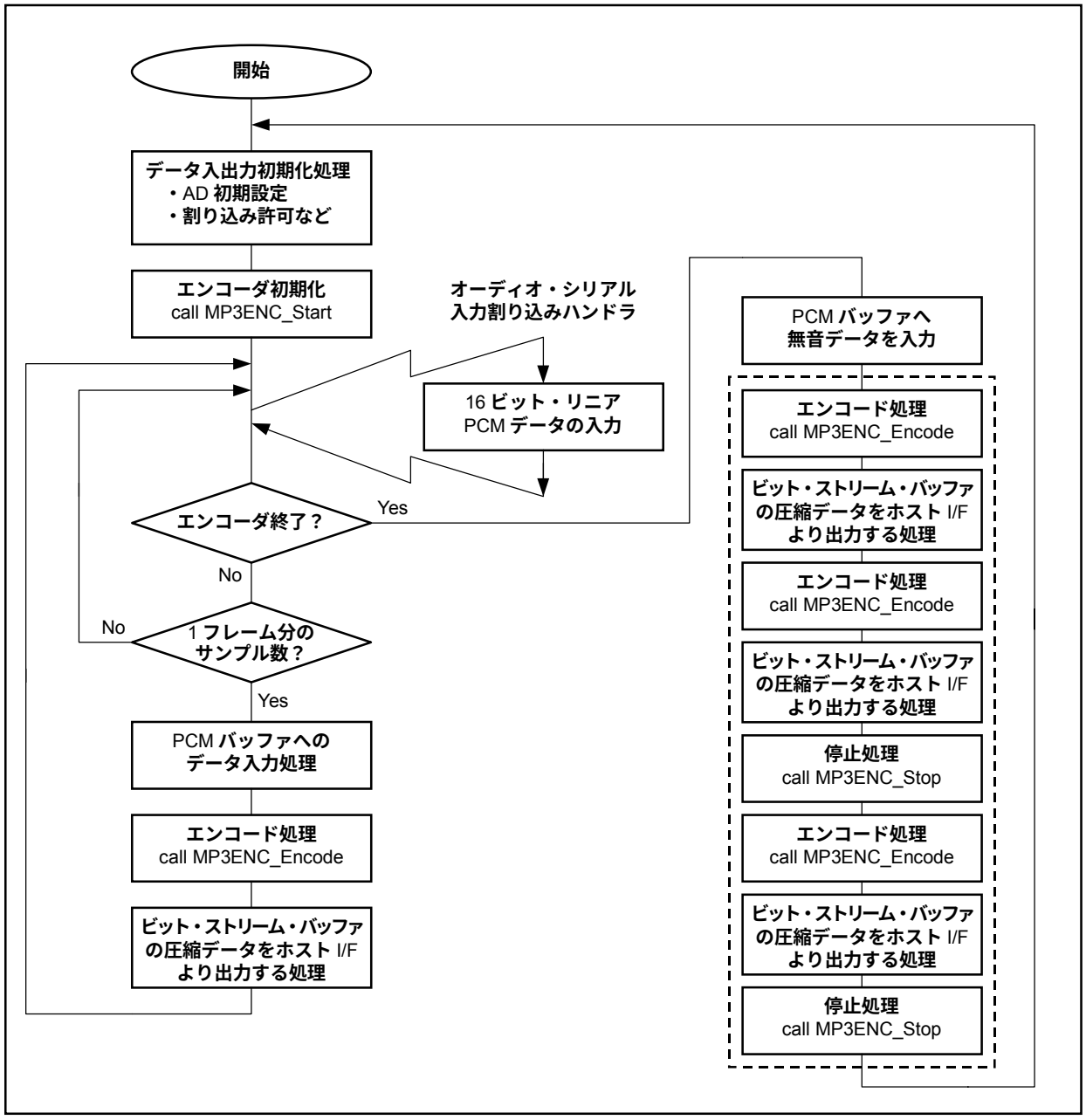
表 2-1 ライブラリ関数一覧

関 数 名	機　　能
MP3ENC_Start	初期化
MP3ENC_Encode	圧縮処理
MP3ENC_Stop	停止処理
MP3ENC_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-B19 を使用したアプリケーションの処理の例を図 2-1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　MP3ENC_Start 関数

MP3ENC_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。MP3ENC_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　】MP3 エンコーダ初期化処理

【 関 数 名 】MP3ENC_Start

【　機　能　】μSAP77016-B19 が使用する各種パラメータを初期化します。

【　形　式　】call　MP3ENC_Start

【　引　数　】R0L: X メモリのスタティック領域の先頭アドレス[注]。

R1L: X メモリのスクラッチ領域の先頭アドレス[注]。

R2L: Y メモリのスタティック領域の先頭アドレス[注]。

R3L: Y メモリのスクラッチ領域の先頭アドレス[注]。

【 返 却 値 】なし

【使用レジスタ】r0, r1, r2, r3, r4, r7,

dp0, dp4, dp5

【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：	3214
最大サイクル数：	11642

注　メモリ領域はユーザが定義する必要があります。メモリ領域については，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，スクラッチ・メモリおよびスタティック・メモリ領域を確保しておいてください。

2. 3. 2　MP3ENC_Encode 関数

MP3ENC_Encode 関数は，表 1-2　入力データのワード数に示すワード数のオーディオ・データを指定したビット・レートで圧縮します。

【　分　類　】MP3 エンコード処理

【 関 数 名 】MP3ENC_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　MP3ENC_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2]。

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2, 3]。

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。

【使用レジスタ】r0, r1, r2, r3, r4, r5, r6, r7

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	7
最大ループ・スタック・レベル	2
最大リピート回数	31
最大 MIPS 値	84.2 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。

3. 出力バッファ（ビット・ストリーム・バッファ）の先頭アドレスは，X メモリのスクラッチ領域の先頭アドレスと同じです。

注意　エンコード終了時，ユーザは次のフローを行ってください。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。

（1）PCM バッファに無音データを 1 フレーム分（表 1-2　入力データのワード数参照）入力する。

（2）MP3ENC_Encode 関数を 2 回呼び出す。

（3）MP3ENC_Stop 関数を 1 回呼び出す。

（4）MP3ENC_Encode 関数を 1 回呼び出す。

（5）MP3ENC_Stop 関数を 1 回呼び出す。

2. 3. 3　MP3ENC_Stop 関数

MP3ENC_Stop 関数は，エンコーダの停止処理を行います。エンコード終了時の処理フローについては，図 2-1　アプリケーション処理フローを参照してください。

【　分　類　】MP3 エンコーダ停止処理

【 関 数 名 】MP3ENC_Stop

【　機　能　】μSAP77016-B19 の停止処理を行います。

【　形　式　】call　MP3ENC_Stop

【　引　数　】R0L: 1（エンコード終了処理中，1 回目）

0（エンコード終了処理中，2 回目）

エンコード終了処理中，1 回目に MP3ENC_Stop 関数を呼ぶときは“ 1 ”を，2 回目に MP3ENC_Stop 関数を呼ぶときは“ 0 ”を設定してください。

【 返 却 値 】なし

【使用レジスタ】r0, r7,

dp4, dp5

【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：	0
最大サイクル数：	79

2. 3. 4　MP3ENC_GetVersion 関数

MP3ENC_GetVersion 関数は，μSAP77016-B19 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 】MP3ENC_GetVersion

【　機　能　】μSAP77016-B19 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　MP3ENC_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】

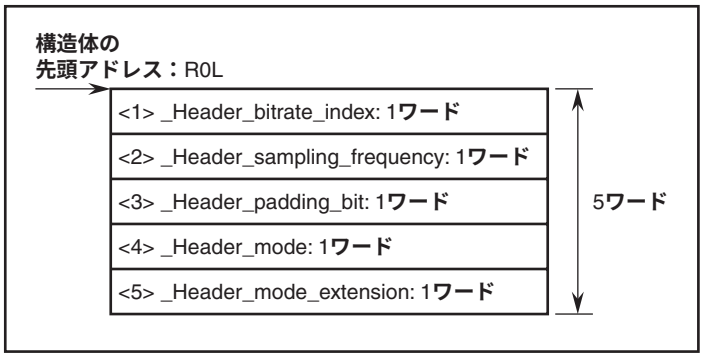
最大スタック・レベル	0
最大ループ・スタック・レベル	0
最大リピート回数	0
最大サイクル数：	6

2. 4　圧縮に必要なパラメータ群

ユーザは，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。

MP3ENC_Start 関数の実行と MP3ENC_Encoder 関数の実行の間に，1 回だけ，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>　_Header_bitrate_index（1 ワード）：ビット・レートを設定します。設定値については，**表 1-5　ビット・レートとビットの値の関係**を参照してください。

たとえば，ビット・レートを MPEG-2 Audio Layer-3 LSF 16 kbps としたいときは，0x2 を設定します。

<2>　_Header_sampling_frequency（1 ワード）：サンプリング周波数を設定します。このワードの設定値を表 2-2に示します。

たとえば，MPEG-2 Audio Layer-3 LSF 16000 Hz としたいときは，0x6 を設定します。

表 2-2　サンプリング周波数の設定値

圧縮方式	値	周波数［Hz］
MPEG-1 Audio Layer-3	0x0	44100
	0x1	48000
	0x2	32000
	0x3	設定禁止
MPEG-2 Audio Layer-3 LSF	0x4	22050
	0x5	24000
	0x6	16000
	0x7	設定禁止

<3>　_Header_padding_bit（1 ワード）：ユーザが設定する必要のないパラメータです。任意の値を設定してください。

<4>　_Header_mode（1 ワード）：チャネル数が 1 のとき 0x3 を，チャネル数が 2 のとき 0x1 を設定してください。

<5>　_Header_mode_extension（1 ワード）：ユーザが設定する必要のないパラメータです。任意の値を設定してください。

2. 5　メモリ構造

μSAP77016-B19 では，処理に必要なメモリ領域と入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。それぞれの領域のメモリ・サイズを，表 2-3に示します。

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	5000	X	スクラッチ領域
scratch_y_area	2533	Y	スクラッチ領域
static_x_area	2304	X	スタティック領域
static_y_area	3214	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。スクラッチ領域とスタティック領域のサイズには，入力バッファ（PCM バッファ）は含まれていません。2. 5. 3　入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-B19 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-B19 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
```
SCRATCH_X   xramseg
scratch_x_area:     ds   5000

SCRATCH_Y   yramseg
scratch_y_area:     ds   2533
```

注意　μSAP77016-B19 では，X メモリのスクラッチ領域をビット・ストリーム・バッファと共用する仕様になっています。そのため，1 フレームのエンコード処理が終了したあと，X メモリのスクラッチ領域（ビット・ストリーム・バッファ）に出力されたエンコード・データ（ビット・ストリーム）をほかの領域にコピーしてから，次のフレームのエンコード処理を行ってください。

2. 5. 2　スタティック領域

スタティック領域とは，μSAP77016-B19 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-B19 の正常な動作を保証できません。

例
```
STATIC_X    xramseg
static_x_area:    ds    2304

STATIC_Y    yramseg
static_y_area:    ds    3214
```

2. 5. 3　入力バッファ

入力バッファ（PCM バッファ）は，オーディオ・データ（16 ビット・リニア PCM データ）の入力用の領域です。ユーザは，入力バッファ用の領域を X メモリに確保してください。必要な入力バッファのサイズを表 2-4 に示します。

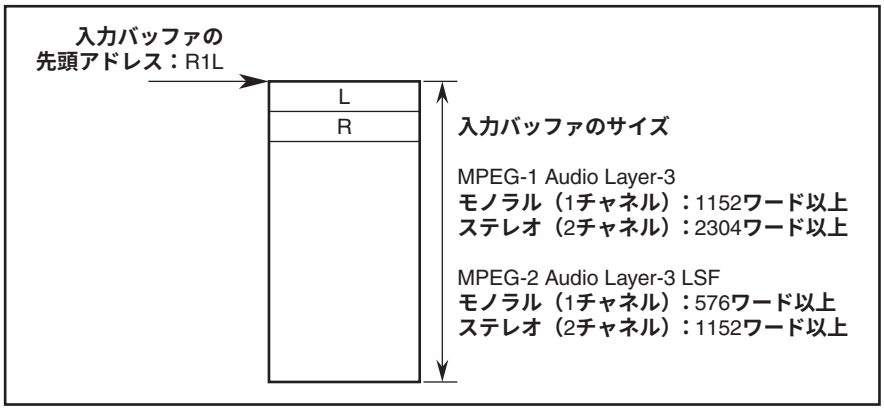
1 フレームのエンコード処理のあと，ユーザは入力バッファ用の領域を自由に使用できます。

エンコード処理中に入力バッファ用の領域を操作した場合，正常な動作を保証できません。

表 2-4　入力バッファのサイズ

圧縮方式	チャネル数	入力するオーディオ・データのワード数	必要な入力バッファのサイズ［ワード］
MPEG-1 Audio Layer-3	1	1152	1152 以上
	2	2304	2304 以上
MPEG-2 Audio Layer-3 LSF	1	576	576 以上
	2	1152	1152 以上

図 2-3　ユーザ定義の入力バッファ（PCM バッファ）



2. 5. 4　出力バッファ

出力バッファ（ビット・ストリーム・バッファ）は，ビット・ストリーム・データの出力用の領域です。

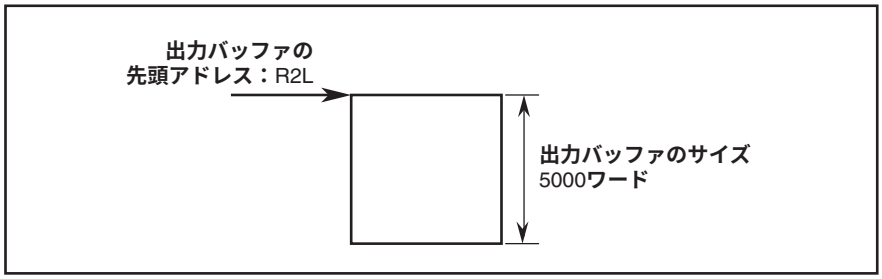
μSAP77016-B19 では，X メモリのスクラッチ領域をビット・ストリーム・バッファと共用する仕様になっています。X メモリのスクラッチ領域とは別の領域に，出力バッファを確保する必要はありません。

1 フレームのエンコード処理のあとに，出力されたエンコード・データ（ビット・ストリーム）[注]をほかの領域にコピーすることで，ユーザは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出力バッファ用の領域を操作した場合，正常な動作を保証できません。

注　1 フレームのエンコード時に出力されるビット・ストリーム・サイズの最大値は，1872 ワードです。

注意　**μSAP77016-B19 では，X メモリのスクラッチ領域をビット・ストリーム・バッファと共用する仕様になっています。そのため，1 フレームのエンコード処理が終了したあと，X メモリのスクラッチ領域（ビット・ストリーム・バッファ）に出力されたエンコード・データ（ビット・ストリーム）を他の領域にコピーしてから，次のフレームのエンコード処理を行ってください。**

図 2-4　出力バッファ（ビット・ストリーム・バッファ）



備考　出力バッファ（ビット・ストリーム・バッファ）の先頭アドレスは，X メモリのスクラッチ領域の先頭アドレスと同じです。

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-B19（MP3 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-B19 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-B19 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	MP3ENC_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-B19 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（mp3enc.h）

```
/*---------------------------------------------------------------------*/
/*     File Information                                                */
/*---------------------------------------------------------------------*/
/*     Name      : mp3enc.h                                            */
/*     Type      : header file                                         */
/*     Version   : 1.00                                                */
/*     Date      : 2002 Nov 26                                         */
/*     CPU       : uPD7701x Family                                     */
/*     Assembler : WB77016 Ver 2.4                                     */
/*     About     : Header                                              */
/*                                                                     */
/*---------------------------------------------------------------------*/
/*     Copyright (C) NEC Electronics Corporation 2002                  */
/*     NEC ELECTRONICS CONFIDENTIAL AND PROPRIETARY                    */
/*     All rights reserved by NEC Electronics Corporation.             */
/*     Use of copyright notice does not evidence publication           */
/*---------------------------------------------------------------------*/

#ifndef  __mp3enc_h
#define  __mp3enc_h

#define  MP3ENC_SCRATCH_AREA_X_SIZE    5000
#define  MP3ENC_SCRATCH_AREA_Y_SIZE    2533
#define  MP3ENC_STATIC_AREA_X_SIZE     2304
#define  MP3ENC_STATIC_AREA_Y_SIZE     3214

#define  HeaderSize   5
;typedef struct {
;         short   bitrate_index
;         short   sampling_frequency
;         short   padding_bit
;         short   mode
;         short   mode_extension
;}

extrn    MP3ENC_Start
extrn    MP3ENC_Encode
extrn    MP3ENC_Stop
extrn    MP3ENC_GetVersion

#endif
```

付録.2　サンプル・ソース・ファイル（sample.asm）


```
/*---------------------------------------------------------------------*/
/*     File Information                                                */
/*---------------------------------------------------------------------*/
/*     Name      : sample.asm                                          */
/*     Type      : Assembler program module                            */
/*     Version   : 1.00                                                */
/*     Date      : 2002 Nov 26                                         */
/*     CPU       : uPD7701x Family                                     */
/*     Assembler : WB77016 Ver 2.4                                     */
/*     About     : sample                                              */
/*                                                                     */
/*---------------------------------------------------------------------*/
/*     Copyright (C) NEC Electronics Corporation 2002                  */
/*     NEC ELECTRONICS CONFIDENTIAL AND PROPRIETARY                    */
/*     All rights reserved by NEC Electronics Corporation.             */
/*     Use of copyright notice does not evidence publication           */
/*---------------------------------------------------------------------*/

#include "mp3enc.h"

/*---------------------------------------------------------------------*/
#define    BitstreamBufferSize         MP3ENC_SCRATCH_AREA_X_SIZE
#define    PcmBufferSize               1152*2

#define    header_bitrate_index        _Header_bitrate_index:x
#define    header_sampling_frequency   _Header_sampling_frequency:x
#define    header_padding_bit          _Header_padding_bit:x
#define    header_mode                 _Header_mode:x
#define    header_mode_extension       _Header_mode_extension:x

#define    OutputBufferFlag            _OutputBufferFlag:x
#define    valued                      _valued:x
#define    BitstreamLength_H           _BitstreamLength:x
#define    BitstreamLength_L           _BitstreamLength+1:x
#define    frame                       _frame:x

#define    Channel                     _Channel:x
#define    bitrate_H                   _bitrate:x
#define    bitrate_L                   _bitrate+1:x
#define    stream_length               _stream_length:x
#define    FW_out_flag                 _FW_out_flag:x
#define    FW_run_flag                 _FW_run_flag:x
/*---------------------------------------------------------------------*/

MAIN_CTRL_WORK   xramseg at 0x0ff0
      _Channel:                  ds    1
      _bitrate:                  ds    2
      _OutputBufferFlag:         ds    1
      _valued:                   ds    1
```

```
      Header:
      _Header_bitrate_index:           ds    1
      _Header_sampling_frequency:      ds    1
      _Header_padding_bit:             ds    1
      _Header_mode:                    ds    1      ; use in timing file
      _Header_mode_extension:          ds    1

      _BitstreamLength:                ds    2
      _frame:                          ds    1

      _stream_length:                  ds    1      ; use in timing file
      _FW_out_flag:                    ds    1      ; use in timing file
      _FW_run_flag:                    ds    1      ; use in timing file

MAIN_X_WORK1     xramseg at 0x2000
      PCM:                             ds    PcmBufferSize

MAIN_Y_WORKyramseg at 0x2000
      OutBitstreamBuffer:              ds    0x1000

STATIC_X   xramseg
      static_x_area:                   ds    MP3ENC_STATIC_AREA_X_SIZE

SCRATCH_X  xramseg
      __BitstreamBuffer:               ds    MP3ENC_SCRATCH_AREA_X_SIZE

STATIC_Y   yramseg
      static_y_area:                   ds    MP3ENC_STATIC_AREA_Y_SIZE

SCRATCH_Y  yramseg
      scratch_y_area:                  ds    MP3ENC_SCRATCH_AREA_Y_SIZE


;-------------------------------------------------
VECTOR  imseg at 0x200

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

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

(3/9)

```
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                    ;


MAIN  imseg at 0x240

%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                   ;
```

(4/9)

```
      dn5 = r0l                   ;
      dn6 = r0l                   ;
      dn7 = r0l                   ;
      dmx = r0l                   ;
      dmy = r0l                   ;
      sp = r0l                    ;
      lsp = r0l                   ;
      r0l = 0xffff                ;
      sr = r0l                    ;
      eir = r0l                   ;
)

StartUp:
      %Initialize                     ;

      clr (r0)                        ;
      dp0 = __BitstreamBuffer         ;
      rep BitstreamBufferSize         ;
                  *dp0++ = r0h        ;
      dp0 = PCM                       ;
      rep PcmBufferSize               ;
                  *dp0++ = r0h        ;

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

      r0l = static_x_area             ;MP3ENC_Start (static_x_area, (scratch_y_area=)__BitstreamBuffer,
      r1l = __BitstreamBuffer         ;        static_y_area, scratch_y_area)
      r2l = static_y_area             ;
      r3l = scratch_y_area            ;
      call MP3ENC_Start               ;

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

      ;; header analysis ;; (MSB) ;; 22+2+4+16 = 44byte -> 22word
      dp0 = PCM                       ;
      rep 22/2                        ;fread (buff, 1, 22, fi)
                        r0 = *dp0++   ;
                        r0 = *dp0++   ;fread (&Channel, 1, 2, fi)
      *Channel = r0h                  ;
                        r0l = *dp0++  ;fread (&freq, 1, 4, fi)
                        r0eh = *dp0++ ;
                                      ;fread (buff, 1, 16, fi)

      clr (r1)                        ;index = 0
      r2 = r0 - 44100                 ;switch (freq) {
      if (r2 == 0) jmp _break         ;case 44100:   index = 0;   break
      r1 = r1 + 1                     ;
      r2 = r0 - 48000                 ;case 48000:   index = 1;   break
      if (r2 == 0) jmp _break         ;
      r1 = r1 + 1                     ;
      r2 = r0 - 32000                 ;case 32000:   index = 2;   break
```

(5/9)

```
      if (r2 == 0) jmp _break           ;
      r1 = r1 + 2                       ;
      r2 = r0 - 22050                   ;case 22050: index = 4;   break
      if (r2 == 0) jmp _break           ;
      r1 = r1 + 1                       ;
      r2 = r0 - 24000                   ;case 24000: index = 5;   break
      if (r2 == 0) jmp _break           ;
      r1 = r1 + 1                       ;
      r2 = r0 - 16000                   ;case 16000: index = 6;   break
      if (r2 == 0) jmp _break           ;
      jmp $                             ;defualt:    exit
_break:                                 ;}
      *header_sampling_frequency = r1l;header.sampling_frequency = index

      clr (r0)                          ;
      r1 = r0 + 1                       ;k = 1
      r0l = *Channel                    ;if (Channel != 2)
      r2 = r0 - 2                       ;      k += 2
      if (r2 != 0) r1 = r1 + 1          ;
      if (r2 != 0) r1 = r1 + 1          ;
      *header_mode = r1l                ;header.mode = k

      clr (r1)                          ;
      r0 = r1 + 64000                   ;64kbps
      nop                               ;
      *bitrate_H = r0h                  ;
      *bitrate_L = r0l                  ;

      r1l = *header_sampling_frequency;
      r2 = r1 - 4                       ;
      r0 = *bitrate_H                   ;
      r0l = *bitrate_L                  ;
      clr (r1)                          ;
      if (r2 >= 0) jmp _MPEG2           ;if (header.sampling_frequency < 4) {
_MPEG1:                                 ;
      r2 = r0 & 7                       ;      switch (bitrate) {
      if (r2 != 0) jmp _break2          ;
      r0 = r0 sra 3                     ;
      r2 = r0 - (32000/8)               ;      case 32000:
      r1 = r1 + 1                       ;             index = 1
      if (r2 == 0) jmp _break2          ;             break
      r2 = r0 - (40000/8)               ;      case 40000:
      r1 = r1 + 1                       ;             index = 2
      if (r2 == 0) jmp _break2          ;             break
      r2 = r0 - (48000/8)               ;      case 48000:
      r1 = r1 + 1                       ;             index = 3
      if (r2 == 0) jmp _break2          ;             break
      r2 = r0 - (56000/8)               ;      case 56000:
      r1 = r1 + 1                       ;             index = 4
      if (r2 == 0) jmp _break2          ;             break
      r2 = r0 - (64000/8)               ;      case 64000:
      r1 = r1 + 1                       ;             index = 5
      if (r2 == 0) jmp _break2          ;             break
      r2 = r0 - (80000/8)               ;      case 80000:
      r1 = r1 + 1                       ;             index = 6
      if (r2 == 0) jmp _break2          ;             break
```

(6/9)

```
      r2 = r0 - (96000/8)               ;      case 96000:
      r1 = r1 + 1                       ;             index = 7
      if (r2 == 0) jmp _break2          ;             break
      r2 = r0 - (112000/8)              ;      case 112000:
      r1 = r1 + 1                       ;             index = 8
      if (r2 == 0) jmp _break2          ;             break
      r2 = r0 - (128000/8)              ;      case 128000:
      r1 = r1 + 1                       ;             index = 9
      if (r2 == 0) jmp _break2          ;             break
      r2 = r0 - (160000/8)              ;      case 160000:
      r1 = r1 + 1                       ;             index = 10
      if (r2 == 0) jmp _break2          ;             break
      r2 = r0 - (192000/8)              ;      case 192000:
      r1 = r1 + 1                       ;             index = 11
      if (r2 == 0) jmp _break2          ;             break
      r2 = r0 - (224000/8)              ;      case 224000:
      r1 = r1 + 1                       ;             index = 12
      if (r2 == 0) jmp _break2          ;             break
      r2 = r0 - (256000/8)              ;      case 256000:
      r1 = r1 + 1                       ;             index = 13
      if (r2 == 0) jmp _break2          ;             break
      r2 = r0 - (320000/8)              ;      case 320000:
      r1 = r1 + 1                       ;             index = 14
      if (r2 == 0) jmp _break2          ;
      clr (r1)                          ;      default:
      jmp _break2                       ;             index = 0
                                        ;      }
_MPEG2:                                 ;} else {
      r2 = r0 & 3                       ;      switch (bitrate) {
      if (r2 != 0) jmp _break2          ;
      r0 = r0 sra 2                     ;
      r2 = r0 - (8000/4)                ;      case 8000:
      r1 = r1 + 1                       ;             index = 1
      if (r2 == 0) jmp _break2          ;             break
      r2 = r0 - (16000/4)               ;      case 16000:
      r1 = r1 + 1                       ;             index = 2
      if (r2 == 0) jmp _break2          ;             break
      r2 = r0 - (24000/4)               ;      case 24000:
      r1 = r1 + 1                       ;             index = 3
      if (r2 == 0) jmp _break2          ;             break
      r2 = r0 - (32000/4)               ;      case 32000:
      r1 = r1 + 1                       ;             index = 4
      if (r2 == 0) jmp _break2          ;             break
      r2 = r0 - (40000/4)               ;      case 40000:
      r1 = r1 + 1                       ;             index = 5
      if (r2 == 0) jmp _break2          ;             break
      r2 = r0 - (48000/4)               ;      case 48000:
      r1 = r1 + 1                       ;             index = 6
      if (r2 == 0) jmp _break2          ;             break
      r2 = r0 - (56000/4)               ;      case 56000:
      r1 = r1 + 1                       ;             index = 7
      if (r2 == 0) jmp _break2          ;             break
      r2 = r0 - (64000/4)               ;      case 64000:
      r1 = r1 + 1                       ;             index = 8
      if (r2 == 0) jmp _break2          ;             break
      r2 = r0 - (80000/4)               ;      case 80000:
```

(7/9)

```
      r1 = r1 + 1                         ;             index = 9
      if (r2 == 0) jmp _break2            ;             break
      r2 = r0 - (96000/4)                 ;      case 96000:
      r1 = r1 + 1                         ;             index = 10
      if (r2 == 0) jmp _break2            ;             break
      r2 = r0 - (112000/4)                ;      case 112000:
      r1 = r1 + 1                         ;             index = 11
      if (r2 == 0) jmp _break2            ;             break
      r2 = r0 - (128000/4)                ;      case 128000:
      r1 = r1 + 1                         ;             index = 12
      if (r2 == 0) jmp _break2            ;             break
      r2 = r0 - (144000/4)                ;      case 144000:
      r1 = r1 + 1                         ;             index = 13
      if (r2 == 0) jmp _break2            ;             break
      r2 = r0 - (160000/4)                ;      case 160000:
      r1 = r1 + 1                         ;             index = 14
      if (r2 == 0) jmp _break2            ;
      clr (r1)                            ;      default:
                                          ;             index = 0
                                          ;      }
_break2:                                  ;}
      *header_bitrate_index = r1l         ;

      clr (r0)                            ;
      *frame = r0h                        ;frame = 0
      *BitstreamLength_H = r0h            ;BitstreamLength = 0
      *BitstreamLength_L = r0l            ;

_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 = MP3ENC_Encode (&Header, PCM, BitstreamBuffer,
      r1l = PCM                           ;             static_x_area, static_y_area)
      r2l = __BitstreamBuffer             ;
      r3l = static_x_area                 ;
      r4l = static_y_area                 ;
      call MP3ENC_Encode                  ;
      if (r0 < 0) jmp _exit               ;      if (ret < 0) exit(ret)
      call CopyOutputBuffer               ;      CopyOutputBuffer (ret, BitstreamBuffer)

      r1l = *frame                        ;      frame++
      r1 = r1 + 1                         ;
      *frame = r1l                        ;

      jmp _while1                         ;}

_break1:
;; MP3ENC_Stop process ;;
      clr (r0)                            ;
      dp0 = PCM                           ;
      rep 1152*2                          ;
                  *dp0++ = r0h            ;
```

(8/9)

```
      r1 = r0 + PCM                     ;ret = MP3ENC_Encode (&Header, PCM, BitstreamBuffer,
      r2 = r0 + __BitstreamBuffer       ;      static_x_area, static_y_area)
      r0l = Header                      ;
      r3l = static_x_area               ;
      r4l = static_y_area               ;
      call MP3ENC_Encode                ;
      if (r0 < 0) jmp _exit             ;if (ret < 0) exit(ret)
      call CopyOutputBuffer             ;CopyOutputBuffer (ret, BitstreamBuffer)
      r0l = Header                      ;ret = MP3ENC_Encode (&Header, PCM, BitstreamBuffer,
      r1l = PCM                         ;      static_x_area, static_y_area)
      r2l = __BitstreamBuffer           ;
      r3l = static_x_area               ;
      r4l = static_y_area               ;
      call MP3ENC_Encode                ;
      if (r0 < 0) jmp _exit             ;if (ret < 0) exit(ret)
      call CopyOutputBuffer             ;CopyOutputBuffer (ret, BitstreamBuffer)

      r0l = 1                           ;FinalFrame = 1
      call MP3ENC_Stop                  ;

      r0l = Header                      ;ret = MP3ENC_Encode (&Header, PCM, BitstreamBuffer,
      r1l = PCM                         ;      static_x_area, static_y_area)
      r2l = __BitstreamBuffer           ;
      r3l = static_x_area               ;
      r4l = static_y_area               ;
      call MP3ENC_Encode                ;
      if (r0 < 0) jmp _exit             ;if (ret < 0) exit(ret)
      call CopyOutputBuffer             ;CopyOutputBuffer (ret, BitstreamBuffer)

      clr (r0)                          ;FinalFrame = 0
      call MP3ENC_Stop                  ;

      r0 = *OutputBufferFlag            ;
      if (r0 == 0) jmp _exit            ;
            r0 = *valued                ;
            *OutBitstreamBuffer: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                       ;

;; Copy from BitstreamBuffer to output buffer ;;
CopyOutputBuffer:
      if (r0 == 0) ret                  ;

      r1 = *BitstreamLength_H           ;
```

```
      r1l = *BitstreamLength_L          ;
      r1 += r0                          ;
      *BitstreamLength_H = r1h          ;
      *BitstreamLength_L = r1l          ;

      dp4 = OutBitstreamBuffer          ;wpt = OutBitstreamBuffer
      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 (r1)                          ;
      *FW_out_flag = r1l                ;
      r1 = *FW_out_flag                 ;
      if (r1 == 0) jmp $-1              ;

      ret                               ;return
end
```

付録.3　パラメータ情報用タイミング・ファイル（value.tmg）

（1/2）

```
;;-------------------------------------------------------------------------
;;  File Information
;;-------------------------------------------------------------------------
;;  Name      : value.tmg
;;  Type      : Timing File
;;  Version   : 1.00
;;  Date      : 2002 Nov 26
;;  CPU       : uPD7701x Family
;;  Assembler : WB77016 Ver2.4
;;  About     : for MP3 audio encoder Version 1.00
;;
;;-------------------------------------------------------------------------
;;  Copyright (C) NEC Electronics Corporation 2002
;;  NEC ELECTRONICS CONFIDENTIAL AND PROPRIETARY
;;  All rights reserved by NEC Corporation.
;;  Use of copyright notice does not evidence publication
;;-------------------------------------------------------------------------

global  Channel, FW_run_flag, FW_out_flag, stream_length, pcm_start, stream_start
global  InstAddrBitrate, InstAddrLSF, Sampling

; MPEG1 / 32kHz / stereo
global  s32_01, s32_02, s32_03

; bitrate
global  b320, b256, b224, b192      ; for MPEG1
global  b144, b024, b016, b008      ; for MPEG2
global  b160, b128, b112, b096, b080, b064, b056, b048, b040, b032       ; for MPEG1/2

local   MPEG1_MASK, MPEG2_MASK

;;; address setting ;;;
set     Channel          = 0x0ff0;;
set     Sampling         = 0x0ff6;;
set     stream_length    = 0x0ffd;;
set     FW_out_flag      = 0x0ffe;;
set     FW_run_flag      = 0x0fff;;
set     pcm_start        = 0x2000;;
set     stream_start     = 0x2000;;

set     InstAddrBitrate  = 0x0296;;
set     InstAddrLSF      = 0x02fd;;

;;; bitrate flag setting ;;;
set     b320 = 1 << 17
set     b256 = 1 << 16
set     b224 = 1 << 15
set     b192 = 1 << 14
set     b160 = 1 << 13
set     b144 = 1 << 12
set     b128 = 1 << 11
set     b112 = 1 << 10
set     b096 = 1 << 9
set     b080 = 1 << 8
```

(2/2)

```
set     b064 = 1 << 7
set     b056 = 1 << 6
set     b048 = 1 << 5
set     b040 = 1 << 4
set     b032 = 1 << 3
set     b024 = 1 << 2
set     b016 = 1 << 1
set     b008 = 1 << 0

;;; mask flag setting ;;;
set     MPEG1_MASK = b320|b256|b224|b192|b160|b128|b112|b096|b080|b064|b056|b048|b040|b032
set     MPEG2_MASK = b160|b144|b128|b112|b096|b080|b064|b056|b048|b040|b032|b024|b016|b008

sub FlagClear
        ;;; zero clear ;;;
        set     s32_01 = 0
        set     s32_02 = 0
        set     s32_03 = 0
endsub

sub FlagMask
        ;;; masking ;;;
        set     s32_01 = s32_01 & MPEG1_MASK
        set     s32_02 = s32_02 & MPEG1_MASK
        set     s32_03 = s32_03 & MPEG1_MASK
endsub


;;; main ;;;
FlagClear
set s32_02 = b040
FlagMask

end
```

付録.4　データ入力用タイミング・ファイル（pcm_in.tmg）

（1/3）

```
;;---------------------------------------------------------------------
;;  File Information
;;---------------------------------------------------------------------
;;  Name      : pcm_in.tmg
;;  Type      : Timing File
;;  Version   : 1.00
;;  Date      : 2002 Nov 26
;;  CPU       : uPD7701x Family
;;  Assembler : WB77016 Ver2.4
;;  About     : for MP3 audio encoder Version 1.00
;;
;;---------------------------------------------------------------------
;;  Copyright (C) NEC Electronics Corporation 2002
;;  NEC ELECTRONICS CONFIDENTIAL AND PROPRIETARY
;;  All rights reserved by NEC Corporation.
;;  Use of copyright notice does not evidence publication
;;---------------------------------------------------------------------
global  Channel, FW_run_flag, FW_out_flag, stream_length, pcm_start
global  InstAddrBitrate, InstAddrLSF, Sampling

; MPEG1 / 32kHz / stereo
global  s32_01, s32_02, s32_03

; bitrate
global  b320, b256, b224, b192       ; for MPEG1
global  b144, b024, b016, b008       ; for MPEG2
global  b160, b128, b112, b096, b080, b064, b056, b048, b040, b032       ; for MPEG1/2

global  size, sz
local   data, pcm
local   bitrate

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
```

(2/3)

```
                set tmp = sz - size
                rept size
                        input data
                        WriteData
                endrept
                set data = 0
                rept tmp
                        WriteData
                endrept
                set size = 0
        endif
endsub

sub wave2stream
        input format hex
        input size

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

        set sz = 22
        Writting

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

        wait cond ip == InstAddrLSF

        set sz = 576
        if (*Sampling:x < 4)
                set sz = sz << 1
        endif
        if (*Channel:x != 1)
                set sz = sz << 1
        endif

        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


;;; Wait for initialize ;;;
wait 1
```

(3/3)

```
if (s32_02)
        if (s32_02 & b040)
                open input "02u32.dat"
                set bitrate = 40000
                wave2stream
                close input
        endif
endif

break
end
```

付録.5　データ出力用タイミング・ファイル（stream_out.tmg）

（1/2）

```
;;---------------------------------------------------------------------
;;  File Information
;;---------------------------------------------------------------------
;;  Name      : stream_out.tmg
;;  Type      : Timing File
;;  Version   : 1.00
;;  Date      : 2002 Nov 26
;;  CPU       : uPD7701x Family
;;  Assembler : WB77016 Ver2.4
;;  About     : for MP3 audio encoder Version 1.00
;;
;;---------------------------------------------------------------------
;;  Copyright (C) NEC Electronics Corporation 2002
;;  NEC ELECTRONICS CONFIDENTIAL AND PROPRIETARY
;;  All rights reserved by NEC Corporation.
;;  Use of copyright notice does not evidence publication
;;---------------------------------------------------------------------
global  FW_out_flag, stream_length, stream_start

; MPEG1 / 32kHz / stereo
global  s32_01, s32_02, s32_03

; bitrate
global  b320, b256, b224, b192 ; for MPEG1
global  b144, b024, b016, b008 ; for MPEG2
global  b160, b128, b112, b096, b080, b064, b056, b048, b040, b032       ; for MPEG1/2

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
```

(2/2)

```
;;; main ;;;
wait 1
if (s32_02)
        if (s32_02 & b040)
                open output #10 "02u32u40.dat"
                wave2stream2
                close output #10
        endif
endif

end
```

〔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。
