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-B07

MP3 オーディオ・デコーダ・ミドルウエア

対象デバイス

μPD77110
μPD77113A
μPD77114
μPD77115

資料番号 U15134JJ4V0UM00（第 4 版）
発行年月 September 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。

- **本資料の内容は予告なく変更することがありますので，最新のものであることをご確認の上ご使用ください。**
- **文書による当社の承諾なしに本資料の転載複製を禁じます。**
- **本資料に記載された製品の使用もしくは本資料に記載の情報の使用に際して，当社は当社もしくは第三者の知的財産権その他の権利に対する保証または実施権の許諾を行うものではありません。上記使用に起因する第三者所有の権利にかかわる問題が発生した場合，当社はその責を負うものではありませんのでご了承ください。**
- **本資料に記載された回路，ソフトウエア，及びこれらに付随する情報は，半導体製品の動作例，応用例を説明するためのものです。従って，これら回路・ソフトウエア・情報をお客様の機器に使用される場合には，お客様の責任において機器設計をしてください。これらの使用に起因するお客様もしくは第三者の損害に対して，当社は一切その責を負いません。**

M7A 98.8

本版で改版された主な箇所

箇　所	内　容
p.27	**2.2.5　mp3_GetErrorStatus 関数の説明を変更。**

本文欄外の★印は，本版で改訂された主な箇所を示しています。

巻末にアンケート・コーナを設けております。このドキュメントに対するご意見をお気軽にお寄せください。

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），
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 を対象デバイスにしています。

目　的　このユーザーズ・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　このユーザーズ・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。

- 第 1 章　概　説
- 第 2 章　ライブラリ仕様
- 第 3 章　インストレーション
- 第 4 章　システム例
- 付　録　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。

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…××××
	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。

μ PD77016 **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				

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

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。

NEC では，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-B07 は，MP3 方式のデ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・デコーダ

このマニュアルでは，オーディオ信号の符号化および復号方式である，MPEG-1 Audio Layer-3 および MPEG-2 Audio Layer-3 LSF（Low Sampling Frequency）を MP3 と称します。これらに関しては ISO/IEC 11172-3, ISO/IEC 13818-3 で規格化されています。

μSAP77016-B07 は，その復号方式に従ったものです。取り扱う圧縮データは，アナログ信号を表 1－1 で示す周波数でサンプリングし，16 ビット・リニア PCM データに変換されたディジタル信号を符号化したデータです。

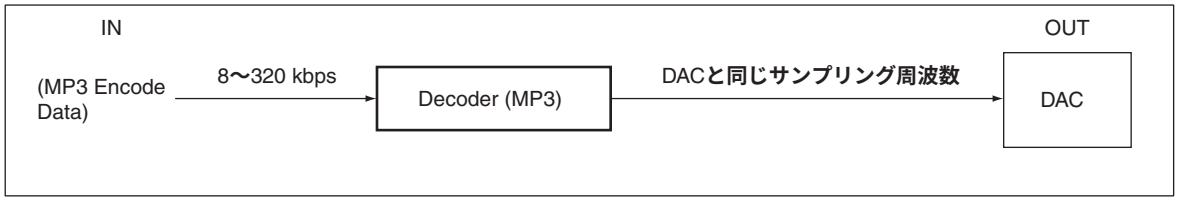
表1－1　サンプリング周波数

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

1.2.1 デコーダ概略

図 1－1 は，μSAP77016-B07 を使用したデコードの構成例です。

図1－1　デコードの構成例



（1）MP3 Encode Data

表 1－1 で示す周波数でサンプリングされたデータ（16 ビット・リニア PCM データ）を，表 1－2 の固定ビット・レートまたは可変ビット・レートで符号化したデータです。

表1－2　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

（2）Decoder

入力データを読み込み，復号処理を行い，16 ビット・リニア PCM データを出力します。

μSAP77016-B07 は，最大 2 チャネルをデコードして出力します。

最後に，誤り補償として CRC 処理を行ないます。

（3）DAC

16 ビット・リニア PCM データをアナログ信号に変換します。

DAC は，符号化データに付加されたサンプリング周波数での動作が必要となります。

DAC と符号化データのサンプリング周波数が異なる場合は，別途，周波数変換ソフト（レート・コンバータなど）が必要になります。

1.3 製品概要

1.3.1 特　徴

(1) ISO/IEC で標準化された MP3 デコーダ・アルゴリズムを採用
(2) ビット・レートは，表 1－2 の固定ビット・レートおよび可変ビット・レートに対応
(3) 入力データは，表 1－1 で示す周波数でサンプリングされた 16 ビット・リニア PCM データを表 1－2 の固定ビット・レートまたは可変ビット・レートで符号化したデータ
(4) 出力データは，入力データのサンプリング周波数と同じ周波数の 16 ビット・リニア PCM データ
(5) 1152 サンプル／フレーム／チャネル（MPEG1），576 サンプル／フレーム／チャネル（MPEG2）の復号化
(6) 誤り補償として CRC をもつ
(7) 自由形式（free format）に対応しない
(8) 2 チャネルのデコードに対応

1.3.2 機　能

(1) 伸長処理

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

(2) 誤り補償

誤り補償処理として，CRC を行います。

1.3.3 動作環境

(1) 動作対象 DSP：

μPD77110, 77113A, 77114, 77115

(2) 必要メモリ・サイズ：

μSAP77016-B07 は，次の表のメモリ・サイズを必要とします。

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	－		6.4
X メモリ	RAM	スクラッチ領域	2.3
		スタティック領域	2.2
	ROM		1.7
Y メモリ	RAM	スクラッチ領域	0.6
		スタティック領域	3.7
	ROM		5.4

注意　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。

スクラッチ領域とは，μSAP77016-B07 が動作していないときに破棄可能なメモリ領域です。ユーザは，μSAP77016-B07 が動作していないときにスクラッチ領域を使用することができます。

ただし，この領域の使用には注意が必要です。μSAP77016-B07 が動作すると再びこの領域を使用するため，スクラッチ領域にユーザが情報を設定していた場合，この設定情報は保証できなくなります。

スクラッチ領域を使用する場合は，表 1－4 を参考にしてください。

表1－4　スクラッチ領域

メモリ	Public シンボル名	サイズ[word]
X メモリ	lib_Scratch_x	2304
Y メモリ	lib_Scratch_y	576

スクラッチ領域はユーザが確保して，mp3_InitDec 関数で設定してください。lib_Scratch_x, lib_Scratch_y ともに宣言時の align/at 指定は不要です。

スタティック領域とは，μSAP77016-B07 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。ユーザは，スタティック領域を使用することはできません。

そのほかに，入力データ・バッファ，出力バッファが必要です。

後述のサンプル・プログラムにおいては，入力データ・バッファとして 1442 ワード，出力バッファとして 2304 ワードを 2 つ，合計 3 つのバッファを使用しています。

入力データ・バッファのサイズを大きくすることは可能ですが，小さくすることは動作に影響する可能性がありますので，行わないでください。

出力バッファのサイズの変更はできません。ただし，バッファの個数は変更可能です。

(3) 必要 D/A スペック

D/A 2ch，16 ビット分解能，表 1－1 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1－5　ソフトウエア・ツール

対象 DSP	ソフトウエア・ツール
μPD77016 ファミリ	DSP ツール WB77016（ワークベンチ） HSM77016（ハイスピード・シミュレータ）

1.3.4 性　能

1 フレームの伸長処理をリアルタイムに実行するために必要な MIPS 値を，表 1－6 に示します。

表1－6　1 フレームの伸長処理に必要な MIPS 数

条件		DSP：μPD77110（動作周波数 75 MHz, 75 MIPS）		
伸長	サンプリング周波数	32 kHz [MIPS]	44.1 kHz [MIPS]	48 kHz [MIPS]
	理論上[注1]の最大 MIPS 値	約 26	約 35	約 37
	実測[注2]での最大 MIPS 値	22.6	31.1	33.9
	実測[注2]での平均 MIPS 値	19.2	26.4	28.7
CRC	理論上[注1]の値	0.4	0.5	0.5

注 1. 理論上とは，プログラム上のループ回数，リピート回数，アルゴリズムの処理ルートが最大サイクル数をとるものとして算出した値。

2. 実測値とは，ビット・レートを 320 kbps，ステレオ（2 チャネル）の符号化データを用いて，実機上でμSAP77016-B07 を実際に実行し，計測した値。

1.3.5 ディレクトリ構成

μSAP77016-B07 のディレクトリ構成を示します。

```
┬ library ┬ m3dram.lib          ：μPD77110, 77115用ライブラリ
│         └ m3drom.lib          ：μPD77113A, 77114用ライブラリ
│
└ smp     ── mp3dec ┬ m3d_dec.h     ：ライブラリ・ヘッダ・ファイル（デコーダ用）
                    ├ m3d_err.h     ：エラー情報のためのヘッダ・ファイル
                    ├ sample.prj    ：プロジェクト・ファイル
                    ├ sample.h      ：サンプル・プログラム用ヘッダ・ファイル
                    ├ sample.asm    ：サンプル・ソース・ファイル
                    ├ sam_data.asm  ：データ領域ファイル
                    ├ sam_int.asm   ：割り込み処理用ソース・ファイル
                    ├ sample.tmg    ：タイミング・ファイル
                    ├ serclk.tmg    ：シリアルに入力するクロック用のタイミング・ファイル
                    └ serout16.tmg  ：シリアルからの出力をファイルに保存するための
                                      タイミング・ファイル
```

次に，各ディレクトリの概要を示します。

（1）library

ライブラリ・ファイルを格納しています。

（2）smp ── mp3dec

サンプル・プログラムのソース・ファイル，ヘッダ・ファイルを格納しています。

また，後述するタイミング・ファイルを用意しています。

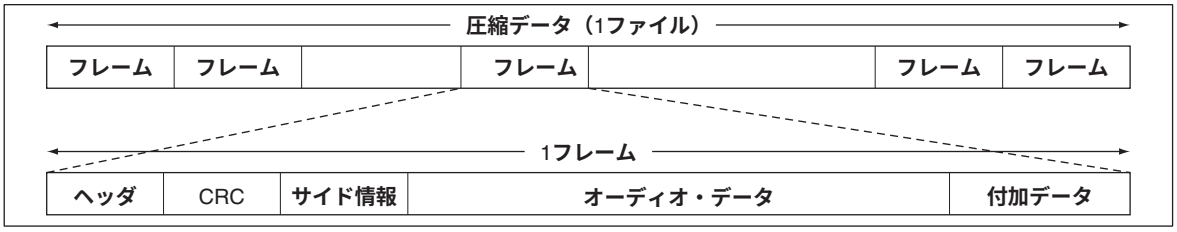
1.4 圧縮データ・フォーマット

圧縮データ･フォーマットの詳細は，**規格書（ISO/IEC 11172-3，JIS X 4323）**を参照してください。

μSAP77016-B07 の仕様は，規格書に従います。ただし，bitrate_index が，‘0000’の場合の自由形式（free format）については，μSAP77016-B07 は対応しません。

圧縮データのフォーマット構造を，図 1－2 に示します。

図1－2　圧縮データ・フォーマット



1.4.1 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1－7　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
保護ビット	1	‘0’：CRC あり　‘1’：CRC なし
ビット・レート	4	表 1－8　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 1－9　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
モード	2	‘00’：stereo　‘01’：joint_stereo ‘10’：dual_channel　‘11’：single_channel
モード拡張子	2	‘00’：is_off, ms_off　‘01’：is_on, ms_off ‘10’：is_off, ms_on　‘11’：is_on, ms_on[注2]
著作権	1	‘0’：no copyright　‘1’：copyright protected
オリジナル／コピー区別	1	‘0’：copy　‘1’：original
エンファシス	2	‘00’：no emphasis　‘01’：50/15 μs ‘10’：reserved　‘11’：CCITT J.17

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1－8　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-B07 のサポート対象外

表1－9　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. 4. 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。

1. 4. 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 1－10 のようになります。

表1－10　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

1.4.4 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，ISO/IEC 11172-3 を参照してください。

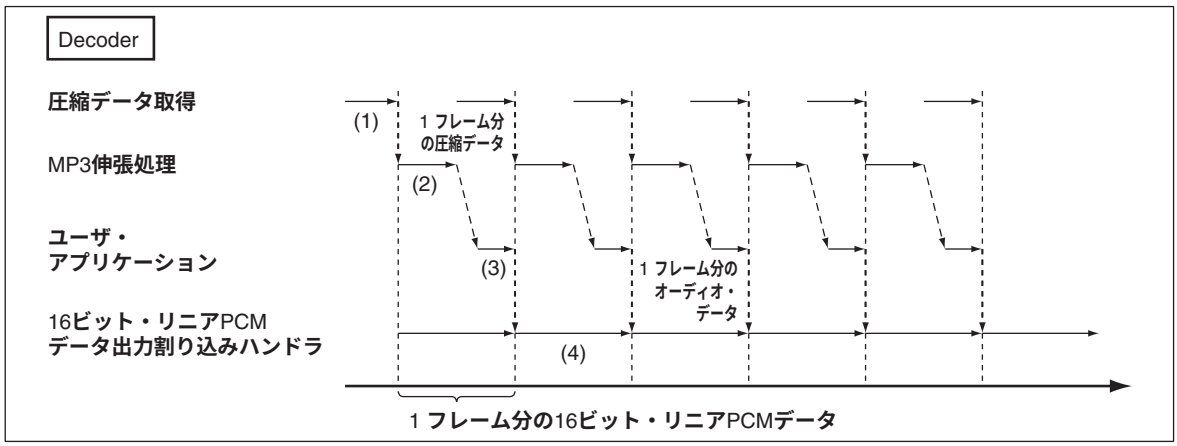
1.4.5 付加データ

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-B07 では，このデータに対して，何も処理を行いません。また，読み込みも行いません。

1.5 タイミング・ダイアグラム

図1－3　デコーダのタイミング・ダイアグラム



(1) 1 フレーム分の圧縮データを読み出し，伸長処理へ引き渡します。

(2) 1 フレーム分の圧縮データを 1 フレーム分の伸長したデータに変換します。

(3) 伸長したデータをバッファリングします。その他に，レート・コンバージョンのようなアプリケーション処理を行います。

(4) 1 フレーム分の 16 ビット・リニア PCM データを D/A 変換します。

第2章　ライブラリ仕様

2.1　ライブラリ概要

μSAP77016-B07 では，次の 5 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
mp3_InitDec	伸長処理初期化
mp3_Dec	伸長処理
mp3_GetVersion	バージョン情報取得
mp3_GetStatus	Status 情報取得
mp3_GetErrorStatus	エラー情報取得

2. 2　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。

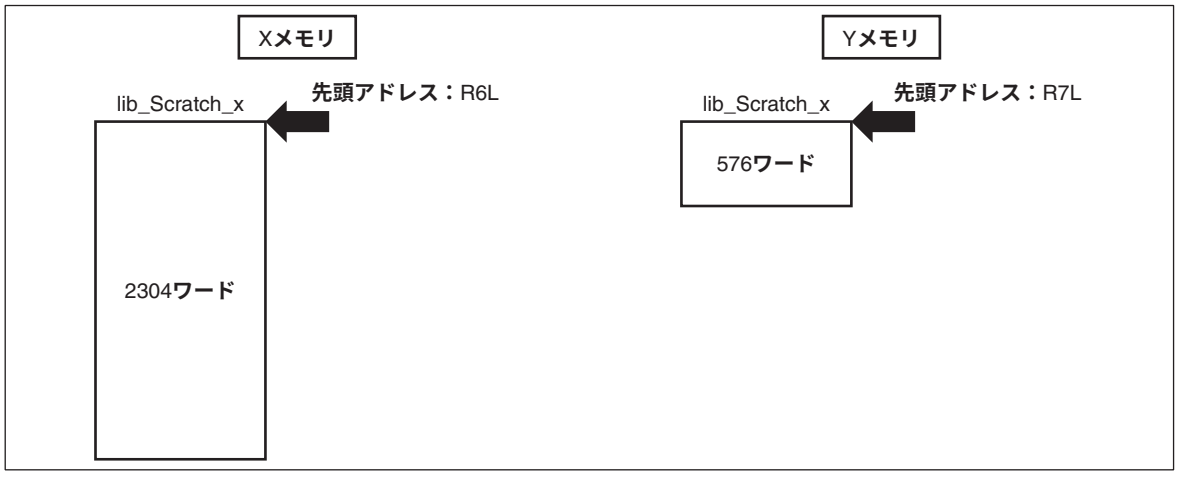
2. 2. 1　mp3_InitDec 関数

【　分　類　】MP3 デコーダ初期化処理
【 関 数 名 】mp3_InitDec
【 機 能 概 要 】μSAP77016-B07 で使用する RAM 領域の初期化およびパラメータの設定
【　形　式　】call　mp3_InitDec
【　引　数　】R0　　入力 MP3 データの長さ[byte]
R1-R5　　予約
R6L　　lib_Scratch_x の先頭アドレス
R7L　　lib_Scratch_y の先頭アドレス
【 返 却 値 】なし
【　機　能　】MP3 デコーダで使用するサブバンド・フィルタ，IMDCT，逆量子化，ステレオ，CRC，メイン処理のパラメータの設定，RAM 領域の初期化を行ないます。
【使用レジスタ】R0, R6, R7, 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	0
最大リピート回数	576
最大サイクル数	4636

注意　mp3_InitDec関数は，MP3デコーダが使用するRAM領域しか初期化を行いません。ユーザが定義するRAM領域（入出力バッファなど）の初期化は，ユーザのプログラムで行うようにしてください。また，lib_Scratch_x, lib_Scratch_yの各領域はあらかじめユーザが確保してください。

図2－1　スクラッチ領域メモリ・イメージ



2. 2. 2　mp3_Dec 関数

【　分　類　】MP3 デコーダ処理

【 関 数 名 】mp3_Dec

【機 能 概 要】指定された圧縮データを，1 フレーム分のオーディオ・データに伸長します。

【　形　式　】call　mp3_Dec

【　引　数　】R0　　0：デコードします

1：1 フレーム分デコードをスキップします

R7　　入力バッファ（ユーザ定義）のリード・ポインタ（図 2−2 参照）における同一アドレスの読み込み順序を設定します

0：DP2 の指すアドレスの上位バイトから使用

1：DP2 の指すアドレスの下位バイトから使用

DP2　　入力バッファ（ユーザ定義）のリード・ポインタ（図 2−2 参照）を設定します

DP3　　入力バッファ（ユーザ定義）のライト・ポインタ（図 2−2 参照）を設定します

DP4　　出力データ・バッファ（ユーザ定義）のポインタに設定します

DMX　　入力バッファ（ユーザ定義）のサイズ−1 の値に設定します

DMY　　出力バッファ（ユーザ定義）のサイズ−1 の値に設定します

【 返 却 値 】DP2　　入力バッファ（ユーザ定義）のリード・ポインタ（図 2−2 参照）を返却します

次回のデコードに使用します

R7　　入力バッファ（ユーザ定義）のリード・ポインタ（図 2−2 参照）の 0：上位バイト／1：下位バイトを返却します

次回のデコードに使用します

【　機　能　】mp3_Dec は，DP2 と R7 で示されるアドレスを入力データの先頭として DP3 で指定されたアドレス直前までの圧縮データに対して伸張処理を行います。

通常，引数の DP2 と R7 は前回のデコードの返却値 DP2, R7 を使用します。

付属のサンプル・プログラムでは，2 フレーム分の出力バッファを持ちますが，これは mp3_Dec が 1 フレームを出力している間にもう一方のフレームのデータを D/A に出力できるようにするためです。圧縮データはビット単位のストリームです。

【使用レジスタ】R0, R1, R2, R3, R4, R5, R6, R7

DP0, DP1, DP2, DP3, DP4, DP5, DP6, DP7

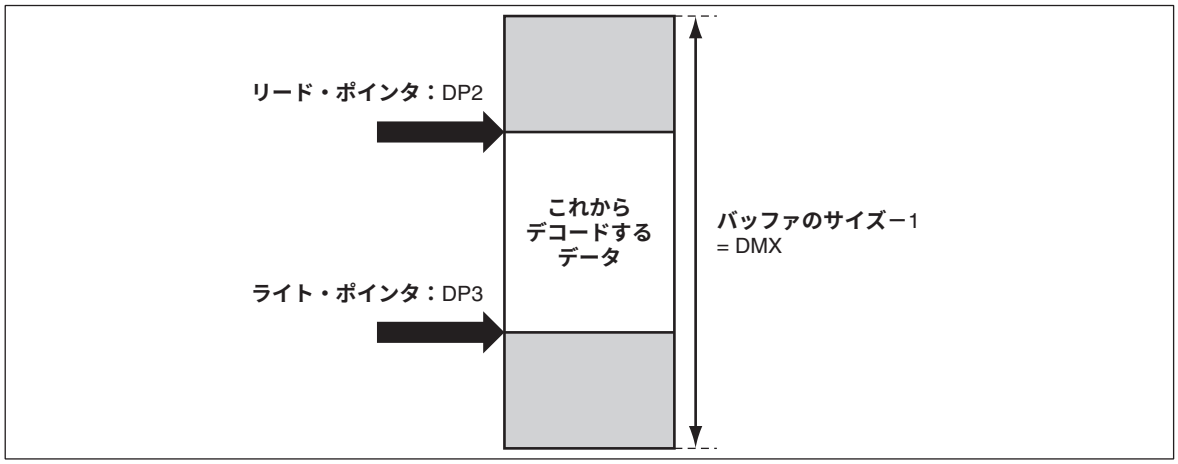
DN0, DN1, DN2, DN3, DN4, DN5, DN6, DN7, DMX, DMY

【ハードウエア・リソースメント】

最大スタック・レベル	5
最大ループ・スタック・レベル	3
最大リピート回数	256
最大 MIPS 値	37 MIPS（エラー符号なしの場合）
	37.5 MIPS（エラー符号ありの場合）

（1）ユーザ定義の入力バッファ

図2－2　ユーザ定義の入力バッファ



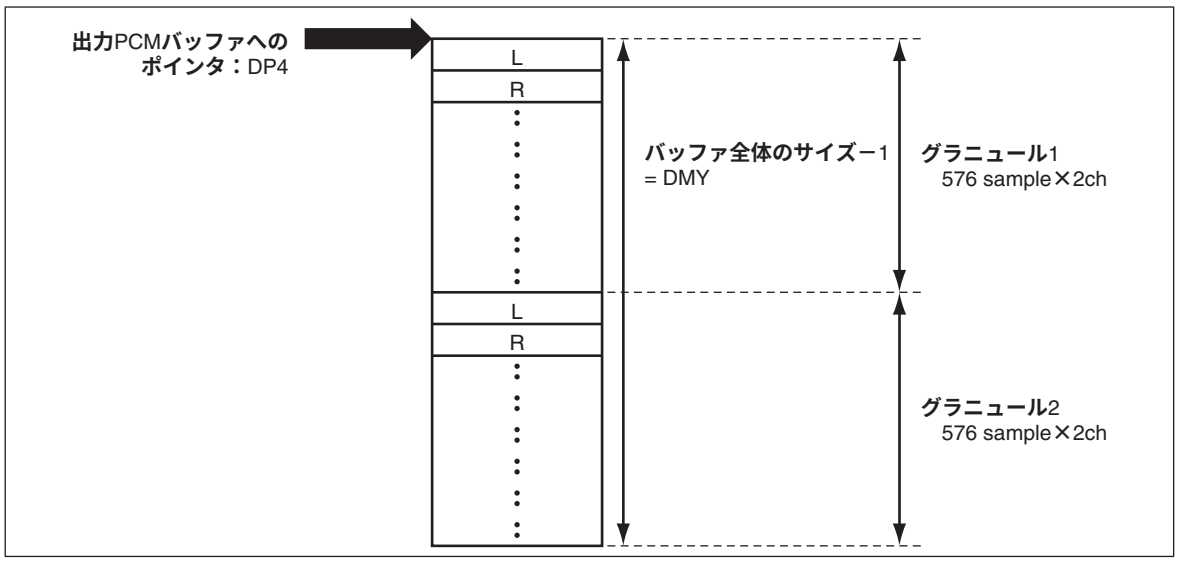
（2）ユーザ定義の出力バッファ

グラニュール： MP3 デコーダ内部の処理単位です。通常，1 フレーム中に 2 グラニュールをデコードします。ただし，LSF 時には，1 フレーム中に 1 グラニュールをデコードします。

sample：　出力する 16 ビット・リニア PCM データの最小単位です。

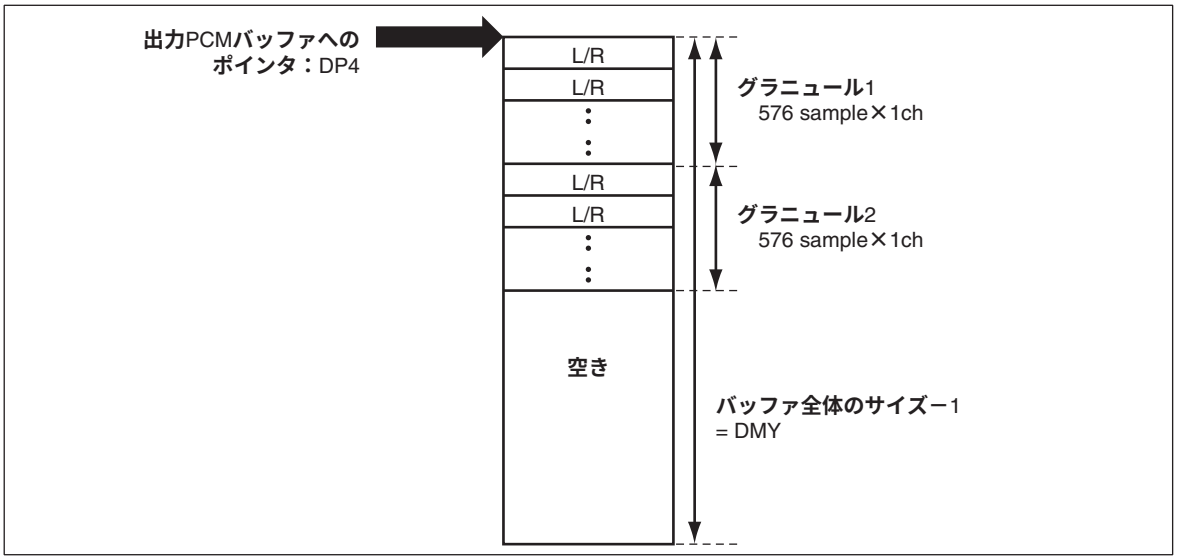
（i）stereo channel の場合

図2－3　stereo channel の場合のユーザ定義出力バッファ



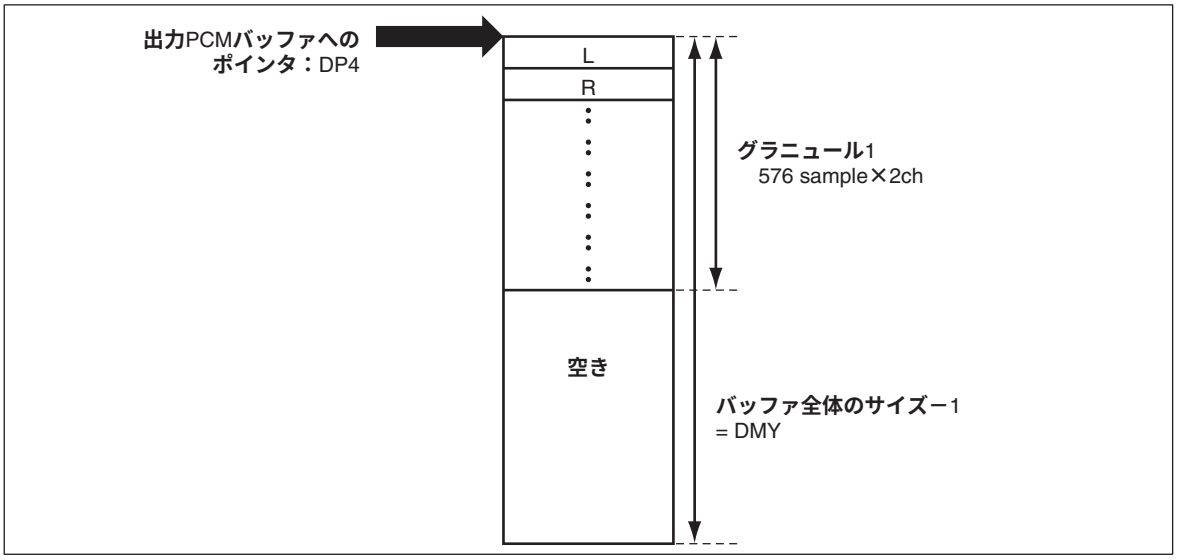
（ii）single channel の場合

図2－4　single channel の場合のユーザ定義出力バッファ



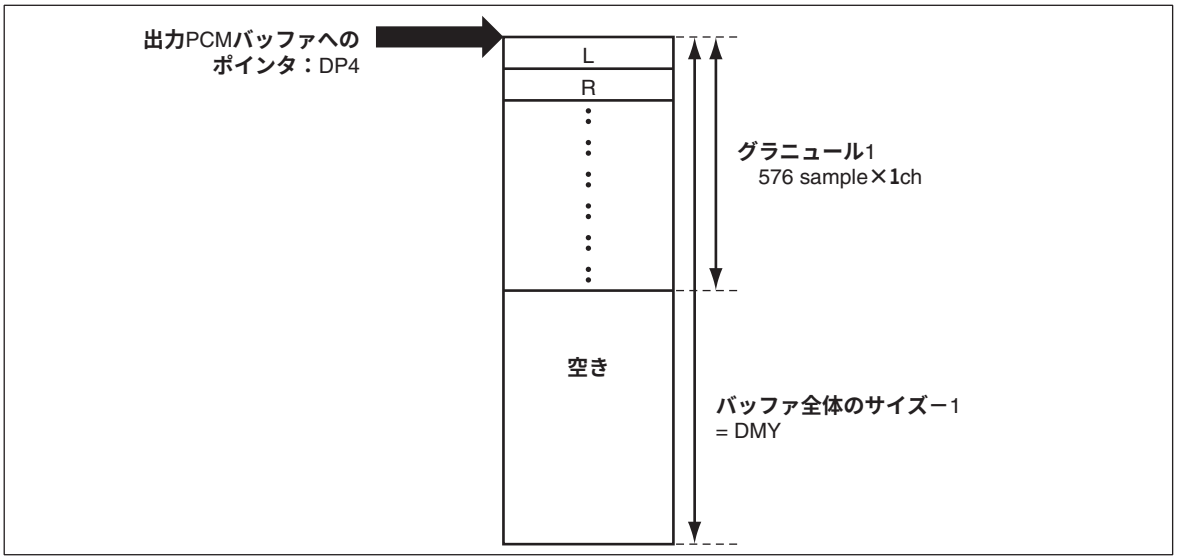
（iii）LSF かつ stereo channel の場合

図2－5　LSF かつ stereo channel の場合のユーザ定義出力バッファ



（iv）LSF かつ single channel の場合

図2－6　LSF かつ single channel の場合のユーザ定義出力バッファ



2. 2. 3　mp3_GetVersion 関数

【　分　類　】バージョン情報取得

【 関 数 名 】mp3_GetVersion

【機 能 概 要】ライブラリのバージョンを返します。

【　形　式　】call　mp3_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

【　機　能　】μSAP77016-B07 ライブラリのバージョン番号を 32 ビットの値で返します。

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

【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. 2. 4　mp3_GetStatus 関数

【　分　類　】Status 情報取得

【 関 数 名 】mp3_GetStatus

【 機 能 概 要 】MP3 デコーダのステータスを取得します。この関数をコールした直前に行ったデコードの情報がステータスとして反映されます。

【　形　式　】call　mp3_GetStatus

【　引　数　】なし

【 返 却 値 】

R0	0：MPEG-2 Audio Layer-3 LSF	
	other：MPEG-1 Audio Layer-3	
R1	0：モノラル	
	other：ステレオ	
R2	サンプリング周波数の Index 値（表 2－2 参照）	
R3	0：通常のデコード	
	other：1 フレーム分デコードをスキップ	
R4	ビット・レートの Index 値（表 2－3 参照）	

【　機　能　】Status 情報を取得しレジスタに設定します。

【使用レジスタ】R0, R1, R2, R3, R4

【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	15

表2－2　mp3_GetStatus( ) の返却値 R2 の周波数

R2 の値	MPEG-1 Audio Layer-3 [Hz]	MPEG-2 Audio Layer-3 LSF [Hz]
0x00	44100	22050
0x01	48000	24000
0x02	32000	16000

表2－3　mp3_GetStatus( ) の返却値 R4 のビット・レート

R4 の値	MPEG-1 Audio Layer-3 [kbps]	MPEG-2 Audio Layer-3 LSF [kbps]
0x00	free format 注	
0x01	32	8
0x02	40	16
0x03	48	24
0x04	56	32
0x05	64	40
0x06	80	48
0x07	96	56
0x08	112	64
0x09	128	80
0x0a	160	96
0x0b	192	112
0x0c	224	128
0x0d	256	144
0x0e	320	160

注　μSAP77016-B07 **のサポート対象外**

2. 2. 5　mp3_GetErrorStatus 関数

【　分　類　】エラー情報取得

【 関 数 名 】mp3_GetErrorStatus

【機 能 概 要】MP3 デコーダのエラー情報を取得します。

【　形　式　】call mp3_GetErrorStatus

【　引　数　】なし

【 返 却 値 】R0L　　エラー・ステータス

R1L　　スキップ・サイズ（バイト数）

【　機　能　】MP3 デコーダのエラー情報を R0L, R1L を使用して取得します。この関数は mp3_Dec 関数をコールしたあとに使用してください。

・ エラー情報として R0L に次の値を返します。

名　前	値	内　容	
MP3_DEC_NO_ERROR	0x0000	正常	続行可能
MP3_DEC_CRC_ERROR	0x0001	CRC エラー	続行可能
MP3_DEC_HEADER_FOUND_ERROR	0x0002	ヘッダ検出エラー	続行可能
★ MP3_DEC_BITSTREAM_ERROR	0x0003	ビット・ストリーム異常 ①入力バッファのビット・ストリーム・データ・サイズが空（0 バイト）です。 ②MP3 データの main_data_begin が指す位置にメイン・データが存在しません。ビット・ストリーム・データが異常です。	続行可能

(i)　CRC エラー

CRC エラーです。このときの DP2, R7（リード・ポインタ）はデコード対象となったフレーム終端の次の位置に移動した状態になります。

(ii)　ヘッダ検出エラー

mp3 のファイルでないか，ファイルが壊れている可能性があります。ユーザはデコードを中断するか，または，mp3_Dec 関数がヘッダを検出するまでビット・ストリームを入力バッファに補充し続けてください。

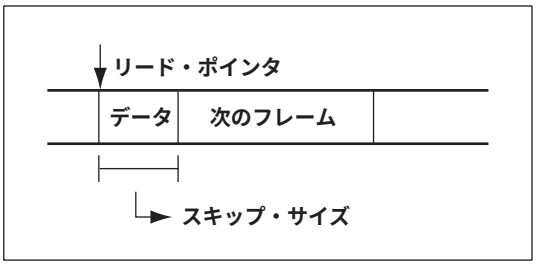
★ (iii) ビット・ストリーム異常

デコードに必要なビット・ストリームが不足している，または，ビット・ストリーム・データが異常です。ユーザはビット・ストリームを補充する必要があります。リード・ポインタとライト・ポインタを適切な値に設定して，再度 mp3_Dec 関数を実行してください。

・返却値のスキップ・サイズについて

フレーム・ヘッダを検出した際に，リード・ポインタから次のフレーム・ヘッダまでスキップしたサイズです。リード・ポインタがフレーム・ヘッダを指していない場合は，スキップした値をバイト・サイズで返します。

図 2−7　リード・ポインタのスキップ・サイズ



【使用レジスタ】 R0, R1

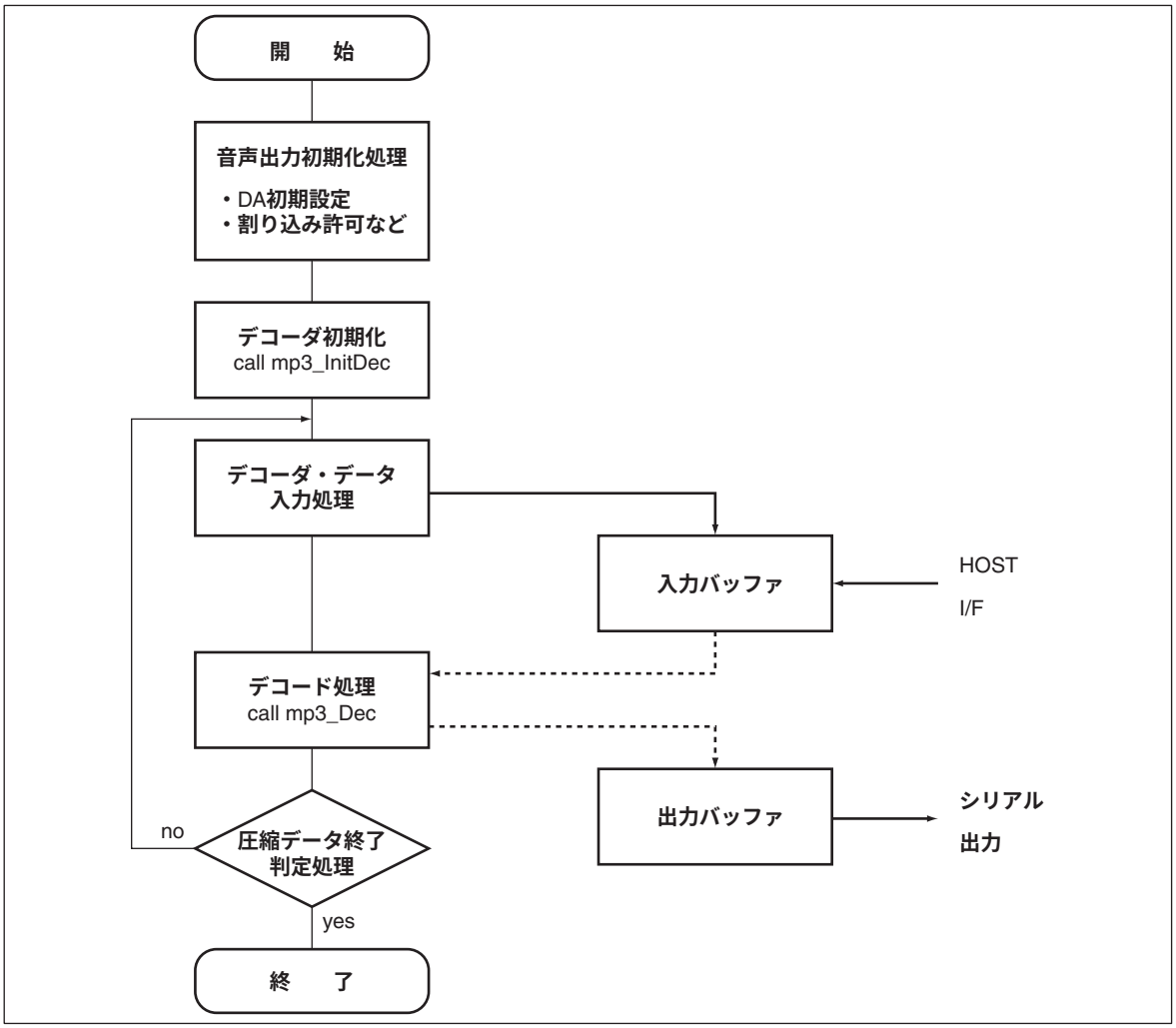
【ハードウエア・リソースメント】

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	5

2. 3　アプリケーション処理フロー

MP3 デコーダを使用したアプリケーションの処理の例を図 2－8 に示します。

図2－8　アプリケーション処理フロー（デコーダ）



割り込みハンドラのオーディオ・データ入出力処理部は，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ので，ユーザがターゲット･システムにあわせて設計してください。

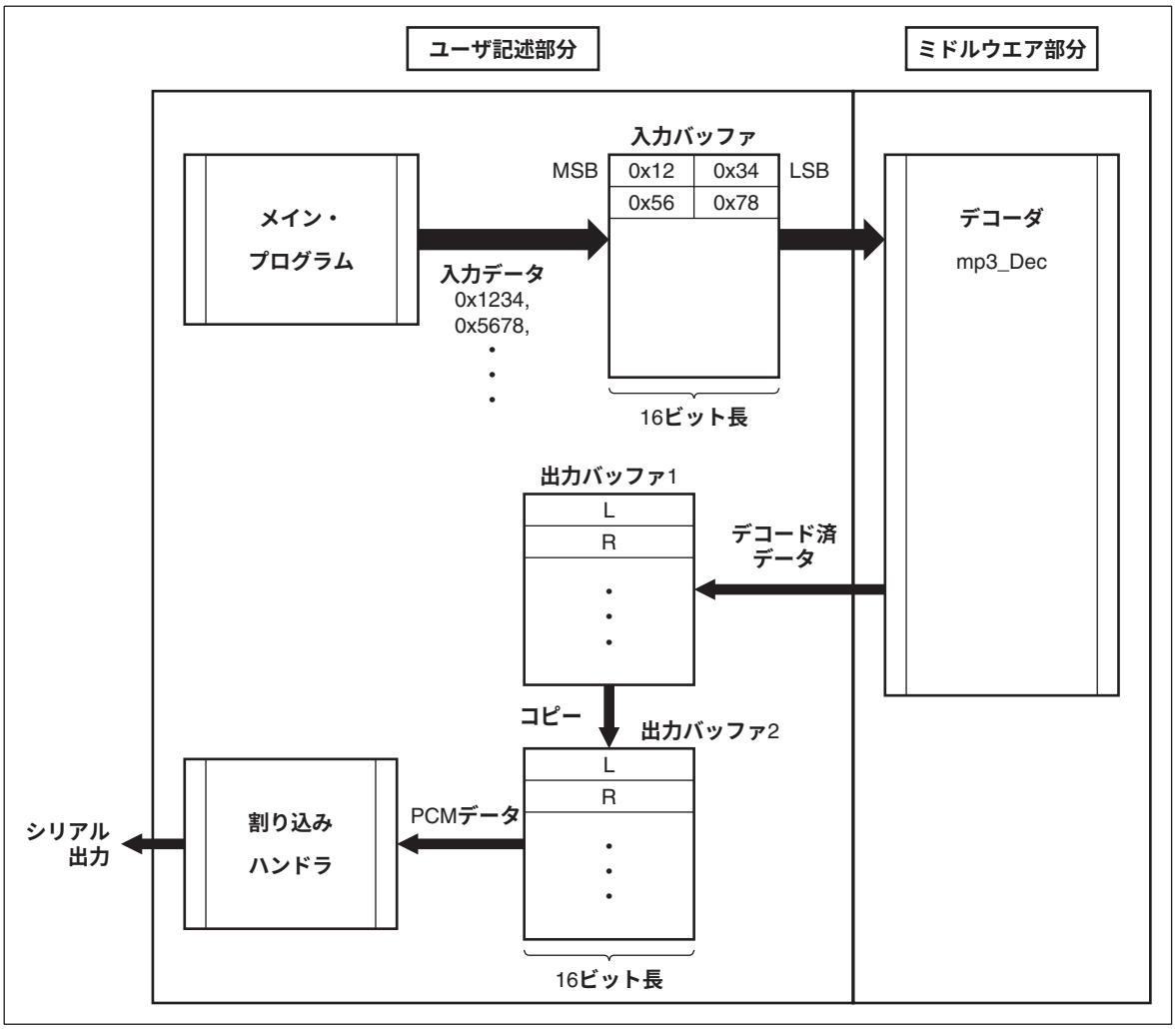
2. 3. 1　データ・フロー

デコード時のデータ・フローの例を図 2－9 に示します。

入力バッファのデータは，MSB から LSB の順にデータを設定する必要があります。

たとえば，データが，0x1234, 0x5678, …というデータは，0x12, 0x34, 0x56, 0x78, …の順に配置します。

図2－9　データ・フロー例



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-B07（MP3 デコーダ・ミドルウエア）の提供媒体は，3.5 インチ・フロッピィ・ディスク（1.44 MB）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) 提供媒体をフロッピィ・ディスク・ドライブにセットします。DSP ツール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，A ドライブから C ドライブへコピーした場合を示します。

```
A:¥>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ては，1. 3. 5　ディレクトリ構成を参照してください。

```
A:¥>dir c:¥DSPTools<CR>
```

3.2　サンプル・プログラム作成手順

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， HSM77016（ハイスピード・シミュレータ） Ver.2.32 以降上で動作します。後述するタイミング・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，第 4 章　システム例を参照してください。
次に MP3 デコーダ・ミドルウエアのサンプル・プログラムのビルド方法について例を示します。

(1) WB77016（ワークベンチ）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) HSM77016（ハイスピード・シミュレータ）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) 次にタイミング・ファイル（sample.tmg, serclk.tmg, serot16.tmg）を開きます。
serclk.tmg, serot16.tmg は，HSM77016（ハイスピード・シミュレータ）が Example で提供するファイルです。

例　「file → open」で sample.tmg を指定します。

3. 3　シンボル命名規約

このライブラリ内で使用しているセクション名は，次のとおりです。

分　類	規　約
関数名，変数名	mp3_xxxx
マクロ，定数名	mp3_XXXX
セクション名	__MP3_XXXX（先頭のアンダスコアは 2 つ）

3. 4　サンプル・プログラム処理フロー

MP3 デコーダを使用したサンプル・プログラムの処理を図 3－1 と図 3－2 に示します。

図3－1　サンプル・プログラム処理フロー（その 1）

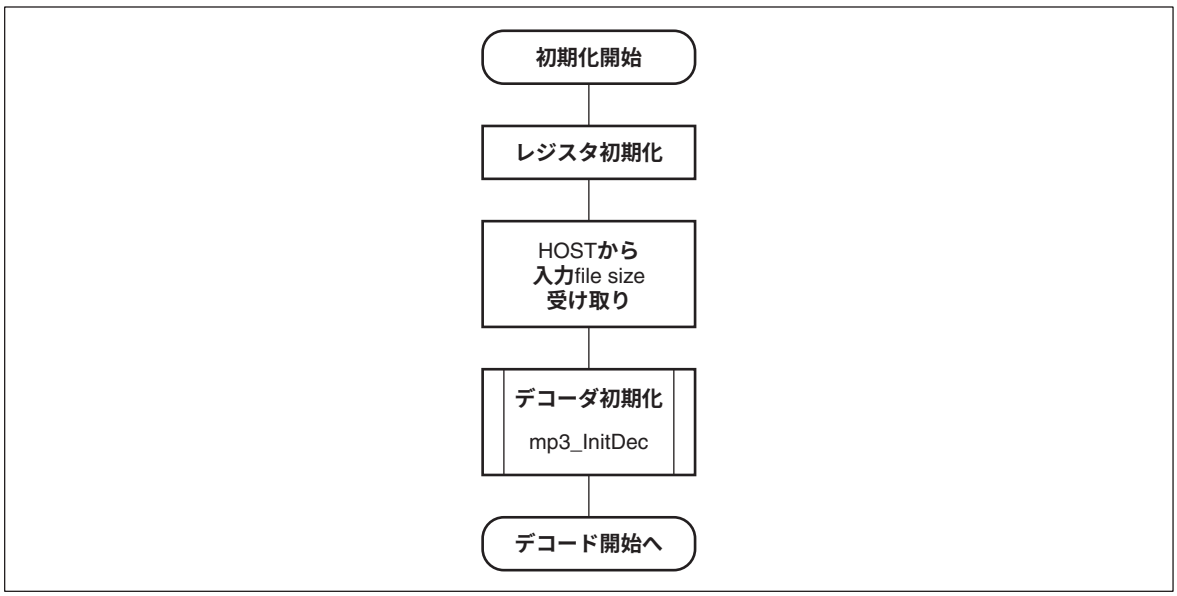
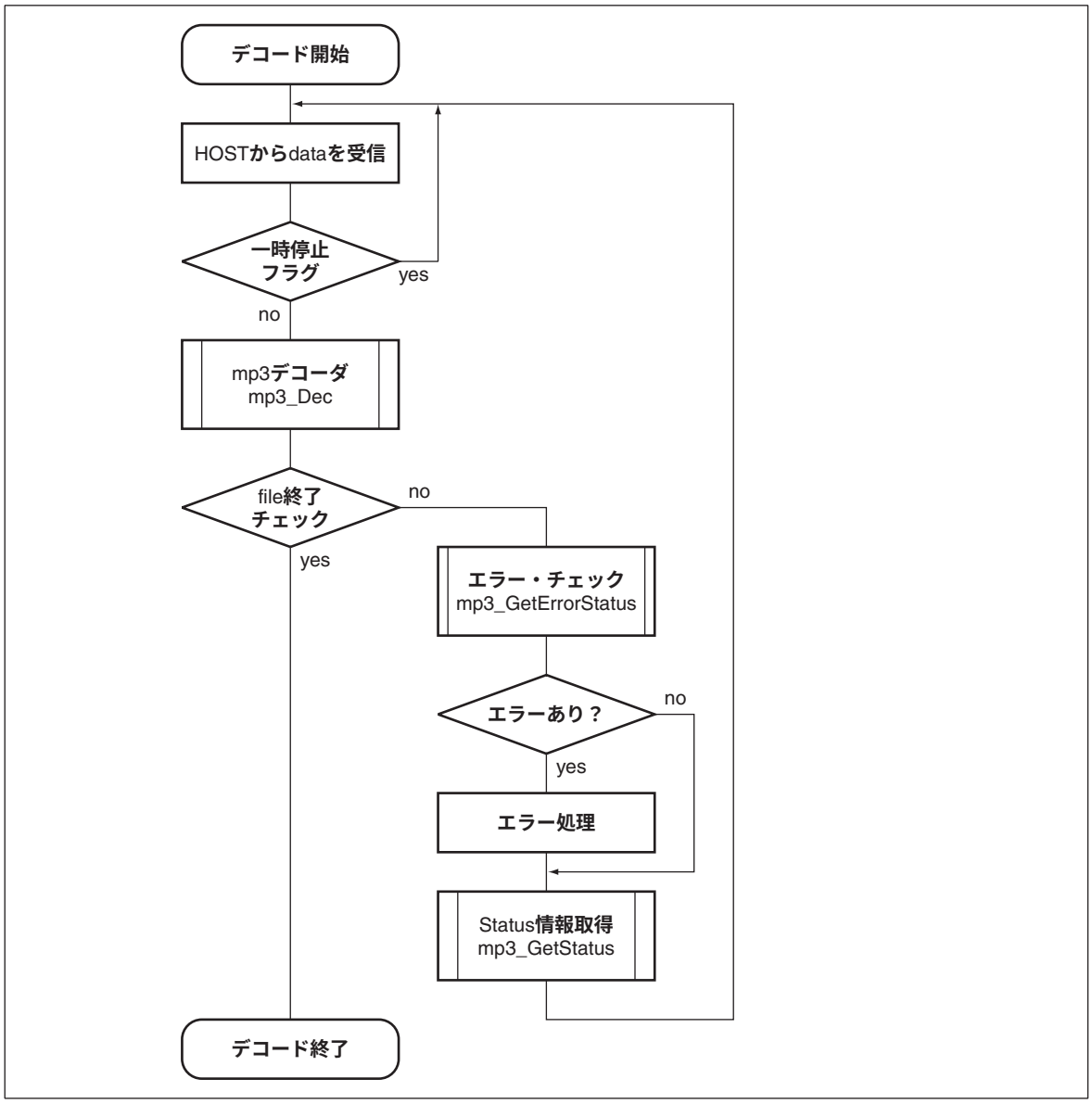


図3－2　サンプル・プログラム処理フロー（その 2）



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

オーディオ・デコードの伸長処理のシミュレータおよびタイミング・ファイルを使用した例を次に示します。符号化データを入力し，1 フレーム単位で圧縮／伸長処理をしながら，オーディオ・データを出力させます。

【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・プログラム作成手順で作成したもの）
・タイミング・ファイル：sample.tmg

4.2　操作方法

① HSM77016（ハイスピード・シミュレータ）を起動します。

②3. 2　サンプル・プログラム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 次にタイミング・ファイル（sample.tmg, serclk.tmg, serot16.tmg）を開きます。
serclk.tmg, serot16.tmg は，HSM77016（ハイスピード・シミュレータ）が Example で提供するファイルです。

例　「file → open」で sample.tmg を指定します。

④ wait 設定を行います。

例　「Window → Periphery Register」で開く設定ウインドウにて，DWTR, IWTR の各レジスタに設定します。

⑤ Run で実行します。

(a) タイミング・ファイル sample.tmg について
HSM77016（ハイスピード・シミュレータ）には，タイミング・ファイルを用いて，外部との入出力をシミュレーションする機能があります。
詳細については，HSM77016　ユーザーズ・マニュアル（U11602J）を参照してください。

(b) データ・ファイルの入力（16 ビット・データ）
ファイルからデータを入力する場合には，ホスト・インタフェースによって行います。次に記述例を示します。

・準備

```
local data                                  ;圧縮データ格納変数
local size                                  ;MP3 ファイルサイズ（byte）

set DEBUG_ID = 1                            ;select target
open input "mp3data.dat"                    ;MP3 ファイルをテキストデータに変換したファイル名
set size = 888058                           ;MP3 ファイルサイズ（バイナリファイルのバイトサイズ）
```

・入力処理（16 ビット・データ）

(1/2)

```
            set pin hcs = 1                 ; terminate any write access, which might
            set pin hwr = 1                 ; be active
            set pin hrd = 1                 ;
; send MP3 file size
             wait cond pin hwe == 0         ; wait till write is allowed
             wait cond pin hcs == 1         ; and no read is in progress
             set pin hcs = 0                ; perform the access...
             set port ha = 0                ; select higher byte of HDT
             set pin hwr = 0                ; start input
             set port hd = (size>>16)&0xFF  ; input low byte to host port
             wait 100ns                     ; access duration
             set pin hwr = 1                ; end output
             set pin hcs = 1                ; terminate first access...
             wait 5ns                       ; delay
             set port ha = 1                ; select higher byte of HDT
             wait 5ns                       ; delay
             set pin hcs = 0                ; performe second access...
             set pin hwr = 0                ; start output
             set port hd = (size>>24)&0xFF  ; input high byte to host port
             wait 100ns                     ; access duration
             set pin hwr = 1                ; end input
             set pin hcs = 1                ; end access

             wait 1                         ;

             wait cond pin hwe == 0         ; wait till write is allowed
             wait cond pin hcs == 1         ; and no read is in progress
             set pin hcs = 0                ; perform the access...
             set port ha = 0                ; select higher byte of HDT
             set pin hwr = 0                ; start input
             set port hd = (size>>0)&0xFF   ; input low byte to host port
             wait 100ns                     ; access duration
             set pin hcs = 1                ; terminate first access...
             set pin hwr = 1                ; end output
             wait 5ns                       ; delay
             set port ha = 1                ; select higher byte of HDT
             wait 5ns                       ; delay
             set pin hcs = 0                ; performe second access...
             set pin hwr = 0                ; start output
             set port hd = (size>>8)&0xFF   ; input high byte to host port
             wait 100ns                     ; access duration
             set pin hwr = 1                ; end input
             set pin hcs = 1                ; end access

            do
             exit size<=0 ;
             wait cond pin hwe == 0         ; wait till write is allowed
             wait cond pin hcs == 1         ; and no read is in progress
             set pin hcs = 0                ; perform the access...
             set port ha = 0                ; select higher byte of HDT
             set pin hwr = 0                ; start input
             input data                     ; input host data to temp variable
             set port hd = data&0xFF        ; input low byte to host port
             wait 100ns                     ; access duration
             set pin hcs = 1                ; terminate first access...
             set pin hwr = 1                ; end output
             wait 5ns                       ; delay
```

(2/2)

```
            set port ha = 1                    ; select higher byte of HDT
            wait 5ns                           ; delay
            set pin hwr = 0                    ; start output
            set pin hcs = 0                    ; performe second access...
            set port hd = (data>>8)&0xFF       ; input high byte to host port
            wait 100ns                         ; access duration
            set pin hwr = 1                    ; end input
            set pin hcs = 1                    ; end access
            set size = size-2                  ;
          enddo

          close input

          wait cond (reg ip & 0xffff)==(mp3_mon_input_bs_finish & 0xffff)
```

mp3_mon_input_bs_finish **はラベル名で，サンプル・プログラム・ソースの先頭部分で定義しています。**

・終了

```
Break                                          ;終了
end
```

付　録　サンプル・プログラム・ソース

- **sample.h**

```
#define MP3_SAMPLE_INPUT_BUFF_SIZE (1440+2)

#define HDT  0x3806
#define HST  0x3807
#define SDT1 0x3800
#define SST1 0x3801
#define SDT2 0x3802
#define SST2 0x3803

;Scratch area
extrn lib_Scratch_x
;XRAM, size= 2304 word
extrn lib_Scratch_y
;XRAM, size= 576 word

extrn mp3_sample_input_read_ptr             ;
extrn mp3_sample_input_read_byte_flag       ;
extrn mp3_sample_input_write_ptr            ;

extrn mp3_sample_dat_length                 ;data length
extrn mp3_sample_remain_length              ;
extrn mp3_sample_decode_length              ;decoded data length
extrn mp3_sample_error_frame_count          ;error counter

extrn mp3_sample_command_stop_flag          ;
extrn mp3_sample_command_skip_flag          ;
extrn mp3_sample_lsf_frame_odd              ;

extrn mp3_sample_input_buff                 ;
extrn mp3_sample_pcm_buff1                  ;
extrn mp3_sample_pcm_buff2                  ;

extrn mp3_sample_int_so1_output_ptr         ;
extrn mp3_sample_int_so1_output_dmx         ;
extrn mp3_sample_int_so1_output_dn          ;

extrn mp3_sample_int_saver0e                ;
extrn mp3_sample_int_saver0h                ;
extrn mp3_sample_int_saver0l                ;

extrn mp3_sample_int_savedp4                ;
extrn mp3_sample_int_savedn4                ;
extrn mp3_sample_int_savedmy                ;

extrn mp3_sample_int1_saver0e               ;
extrn mp3_sample_int1_saver0h               ;
extrn mp3_sample_int1_saver0l               ;
```

• **sample.asm**

(1/5)

```
/*******************************************************/
/*                                                     */
/*       MP3 DECODER MIDDLE WARE                       */
/*                                                     */
/*         SAMPLE.ASM                                  */
/*                                                     */
/*                                                     */
/*******************************************************/

#include "m3d_dec.h"
#include "sample.h"
#include "m3d_err.h"

;for evaluation
public mp3_mon_input_bs_finish                          ;debug symbol
public mp3_mon_decode_result                            ;debug symbol

extrn mp3_sample_copy_mono                              ;
extrn mp3_sample_copy_stereo                            ;

/*******************************************************
mp3_Dec define constant
MP3_DEC_OUTPUT_PCM_BUFFER_SIZE

user define constant
MP3_SAMPLE_INPUT_BUFF_SIZE

user define variable
mp3_sample_input_write_ptr user input buffer write pointer
mp3_sample_input_read_ptr user input buffer read pointer   (mp3_Dec use)
mp3_sample_input_read_byte_flag user input buffer read byte flag (mp3_Dec use)

mp3_sample_dat_length  file length
mp3_sample_remain_length remain file length
mp3_sample_decode_length decoded data length

mp3_sample_error_frame_count error counter

user define buffer
mp3_sample_input_buff ds MP3_SAMPLE_INPUT_BUFF_SIZE input buffer
mp3_sample_pcm_buff1 ds MP3_DEC_OUTPUT_PCM_BUFFER_SIZE output buffer page1
mp3_sample_pcm_buff2 ds MP3_DEC_OUTPUT_PCM_BUFFER_SIZE output buffer page2

*******************************************************/
__sample_start_main imseg at 0x200                      ;
        jmp mp3_sample_init_start                       ; jmp sample main module

__sample_main  imseg                                    ;

mp3_sample_init_start:

/******************  initial routine   ***************/
 r0l = 0x0401                                           ;
 *HST:x = r0l                                           ;

/******************  init register    ***************/
;       Init Reg
        clr(r0)                                         ;
        clr(r1)                                         ;
        clr(r2)                                         ;
        clr(r3)                                         ;
        clr(r4)                                         ;
        clr(r5)                                         ;
        clr(r6)                                         ;
        clr(r7)                                         ;
        dn0 = 0x01                                      ;
        dn1 = 0x01                                      ;
        dn2 = 0x01                                      ;
        dn3 = 0x01                                      ;
        dn4 = 0x01                                      ;
        dn5 = 0x01                                      ;
        dn6 = 0x01                                      ;
        dn7 = 0x01                                      ;
        dp0 = 0x0                                       ;
        dp1 = 0x0                                       ;
        dp2 = 0x0                                       ;
        dp3 = 0x0                                       ;
        dp4 = 0x0                                       ;
        dp5 = 0x0                                       ;
        dp6 = 0x0                                       ;
        dp7 = 0x0                                       ;
        dmx = 0x01                                      ;
        dmy = 0x01                                      ;
```

(2/5)

```
/************* INIT interrupt handler variable ************/
        r0l = mp3_sample_pcm_buff2                          ;
        *mp3_sample_int_so1_output_ptr:y = r0l              ;

/***************   INIT output buffer    **************/
        clr(r0)                                             ;
        dp4 = mp3_sample_pcm_buff2                          ;
        rep MP3_DEC_OUTPUT_PCM_BUFFER_SIZE * 4              ;
        *dp4++ = r0l                                        ;
        dp4 = mp3_sample_pcm_buff1                          ;
        rep MP3_DEC_OUTPUT_PCM_BUFFER_SIZE * 4              ;
        *dp4++ = r0l                                        ;

/******************* INIT SST register  ****************/
        R0L = 0x200                                         ;
        *SST1:X = R0L                                       ;
        *SST2:X = R0L                                       ;
        NOP                                                 ;

/******************* init interrupt mask ****************/
        clr(R0)                                             ;
        R0L = SR                                            ;
        r0 = r0 | 0x03ff                                    ;
;       HO/HI/SO1/INT1/INT2 enable
        R0 = R0 & 0x7fdc                                    ;
        SR = R0L                                            ;
        NOP                                                 ;

        *SDT1:X = R0H                                       ;
        *SDT2:X = R0H                                       ;

/************* receive file length from HOST ************/
        r0L= *HST:x                                         ;
        r0 = r0 & 1                                         ;
        if(r0!=0) jmp $-2                                   ;
        clr(R1)                                             ;
        r2=*HDT:x                                           ;

        r0L= *HST:x                                         ;
        r0 = r0 & 1                                         ;
        if(r0!=0) jmp $-2                                   ;
        r2l=*HDT:x                                          ;

/***************   INIT user variable     **************/
        *mp3_sample_dat_length:x = r2h                      ; data length[byte]
        *mp3_sample_dat_length+1:x = r2l                    ;
        *mp3_sample_remain_length:x = r2h                   ;
        *mp3_sample_remain_length+1:x = r2l                 ;

        clr(r0)                                             ;
        *mp3_sample_decode_length:x  = r0h                  ;decoded length[byte]
        *mp3_sample_decode_length+1:x = r0l                 ;

        *mp3_sample_error_frame_count:x = r0l               ; error counter

        clr(r0)                                             ;
        *mp3_sample_lsf_frame_odd:y = r0l                   ;
        r0l = mp3_sample_input_buff                         ;
        *mp3_sample_input_write_ptr:y = r0l                 ; user input buffer write ptr
        *mp3_sample_input_read_ptr:y = r0l                  ; user input buffer read ptr
        *mp3_sample_input_read_byte_flag:y = r0l            ;
        *mp3_sample_command_stop_flag:y = r0l               ;
        *mp3_sample_command_skip_flag:y = r0l               ;

/***************    init input buffer   **************/
        dp0 = mp3_sample_input_buff                         ;
        clr(r0)                                             ;
        rep MP3_SAMPLE_INPUT_BUFF_SIZE                      ;
        *dp0++ = r0l                                        ;

/***************    init output PCM buffer   **************/
        dp4 = mp3_sample_pcm_buff2                          ;
        rep MP3_DEC_OUTPUT_PCM_BUFFER_SIZE * 4              ;
        *dp4++ = r0l                                        ;
         dp4 = mp3_sample_pcm_buff1                         ;
        rep MP3_DEC_OUTPUT_PCM_BUFFER_SIZE * 4              ;
        *dp4++ = r0l                                        ;

/******************  Init Decoder   ****************/
        clr(r0)                                             ;
        r0 = *mp3_sample_dat_length:x                       ;
        r0l = *mp3_sample_dat_length+1:x                    ;

        r6l = lib_Scratch_x                                 ;
        r7l = lib_Scratch_y                                 ;

        call mp3_InitDec                                    ;
```

(3/5)

```
mp3_sample_main:                                        ; sample main module

/***************     DECODE START     *****************/
        clr(r0)                                         ;
        r0l = *mp3_sample_input_write_ptr:y             ; bit_stream write ptr
        dp2 = r0l                                       ;

        dmx = MP3_SAMPLE_INPUT_BUFF_SIZE-1              ;
        dn2 = 1                                         ;

/*************** calc input buffer blank size *****************/
        clr(r1)                                         ;
        r1l = *mp3_sample_input_read_ptr:y              ;
        r0  = r1 - r0                                   ;
        if(r0>0) jmp $ + 2                              ;
        r0  = r0 + MP3_SAMPLE_INPUT_BUFF_SIZE           ; r0[word]
        r0 = r0 - 1                                     ;
        r0  = r0 sll 0x01                               ; r0[byte]

/************** calc receive data size *****************/
        r2  = *mp3_sample_remain_length:x               ;
        r2l = *mp3_sample_remain_length+1:x             ;
        r1 = r0 - r2                                    ;
                                                        ;
        if( r1 > 0 ) r0 = r2                            ;

/*********** if get size == 0 jmp mp3_Dec **********/
        if (r0 <= 0 ) jmp skip_read_bs                  ;

/*************** Get data from HOST ****************/
        r0 = r0 + 1                                     ;
        r0 = r0 sra 1                                   ; r0[byte] -> r0[word]
        clr(r1)                                         ;
        loop r0l {                                      ;
                r1L= *HST:x                             ;
                r1 = r1 & 1                             ;
                if(r1!=0) jmp $-2                       ;
                r1l=*HDT:x                              ;
                *dp2%%=r1l                              ;
                nop                                     ;
                nop                                     ;
                nop                                     ;
        }

/***************** remain_length Update ******************/
        r0 = r0 sll 1                                   ; r0[word] -> r0[byte]
        r2 = r2 - r0                                    ;
        *mp3_sample_remain_length:x = r2h               ;remain file length Update
        *mp3_sample_remain_length+1:x = r2l             ;

/********* *mp3_sample_inpup_write_ptr:y Update *********/
                                                        ;bit_stream write ptr Update
        r0l = dp2                                       ;
        *mp3_sample_input_write_ptr:y=r0l               ;

/****************** check stop mode ******************/
        clr(r0)                                         ;
        r0l = *mp3_sample_command_stop_flag:y           ;
        if(r0 != 0 ) jmp mp3_sample_main                ;

skip_read_bs:

/****************** set register  ******************/
;       dp2 = read_ptr dmx = INPUT_BUFF_SIZE -1
;       dp3 = write_ptr
;       dmx =
;       r7l = byte_flag
;       dp4 = output_pcm
        r0l = dp2                                       ;
        dp3 = r0l                                       ;write_ptr

        r0l = *mp3_sample_input_read_ptr:y              ;
        dp2 = r0l                                       ;
        dmx = MP3_SAMPLE_INPUT_BUFF_SIZE -1             ;

        dp4 = mp3_sample_pcm_buff1                      ;
        dmy = (MP3_DEC_OUTPUT_PCM_BUFFER_SIZE*4) -1     ;
        clr(r7)                                         ;
        r7l = *mp3_sample_input_read_byte_flag:y        ;
        clr(r0)                                         ;
        r0l = *mp3_sample_command_skip_flag:y           ;

/****************** EXEC mp3 DECODE ******************/
        call mp3_Dec                                    ;  call main program
```

(4/5)

```
/**********   mp3_sample_decode_length Update   **********/
        clr(r0)                                         ;
        r0l = *mp3_sample_input_read_ptr:y              ;
        clr(r1)                                         ;
        r1l = dp2                                       ;
        r0 = r1 - r0                                    ;
        *mp3_sample_input_read_ptr:y = r1l              ;
        if(r0 > 0) jmp $ + 2                            ;
        r0 = r0 + MP3_SAMPLE_INPUT_BUFF_SIZE            ;
        r0 = r0 sll 0x01                                ;word->byte
        r3 = *mp3_sample_decode_length:x                ;
        r3l = *mp3_sample_decode_length+1:x             ;
        r3 = r3 + r0                                    ;
        *mp3_sample_decode_length:x = r3h               ;
        *mp3_sample_decode_length+1:x = r3l             ;
        *mp3_sample_input_read_byte_flag:y = r7l        ;

/************* check remain file size  *************/
        r0 = *mp3_sample_dat_length:x                   ;
        r0l = *mp3_sample_dat_length+1:x                ;
        r0 = r0 - r3                                    ;

/****************  jump END_OF_FILE check routine   ****************/
        if(r0 <= 0) jmp input_bs_end_chk                ;

/*****************  error routine  *********************/
        call mp3_GetErrorStatus                         ;
;       r0: 0:ok     other:error
        if (r0 != 0) jmp mon_error                      ;jump error routine

        call mp3_GetStatus                              ;
; r0  0:lsf other:normal
; r1  mono/stereo 0:mono other:stereo
; r2  sample freq (index)
; r3  skip        0:normal other:skip
; r4  bitrate

;copy decoded PCM data
        dmy = (MP3_DEC_OUTPUT_PCM_BUFFER_SIZE*4) -1 ;
        dn5 = 0x01                                      ;
        dn4 = 0x01                                      ;
        if(r0 != 0 ) jmp _chk_copy_normal               ;
_chk_copy_lsf:
        clr(r0)                                         ;
        r0l = *mp3_sample_lsf_frame_odd:y               ;
        dp4 = mp3_sample_pcm_buff1                      ;
        if(r0 != 0 ) jmp _frame_odd                     ;
_frame_even:
        dp5 = mp3_sample_pcm_buff2                      ;
        r0l = 0x01                                      ;
        *mp3_sample_lsf_frame_odd:y = r0l               ;
_wait_lsf_even:
        clr(r0)                                         ;
        r0l = *mp3_sample_int_so1_output_ptr:y          ;
        r0  = r0 - (mp3_sample_pcm_buff2 + 576*2)       ;
        if(r0 < 0 ) jmp _wait_lsf_even                  ;
        jmp _copy_lsf                                   ;
_frame_odd:
        dp5 = mp3_sample_pcm_buff2 + 576*2              ;
        r0l = 0x00                                      ;
        *mp3_sample_lsf_frame_odd:y = r0l               ;
_wait_lsf_odd:
        clr(r0)                                         ;
        r0l = *mp3_sample_int_so1_output_ptr:y          ;
        r0  = r0 - (mp3_sample_pcm_buff2 + 576*2)       ;
        if(r0 >= 0 ) jmp _wait_lsf_odd                  ;

_copy_lsf:
        if(r1 == 0) call mp3_sample_copy_mono           ;
        if(r1 != 0) call mp3_sample_copy_stereo         ;
        jmp mp3_mon_decode_result                       ;

_chk_copy_normal:
        dp4 = mp3_sample_pcm_buff1                      ;
        dp5 = mp3_sample_pcm_buff2                      ;
_wait_gr1:
        clr(r0)                                         ;
        r0l = *mp3_sample_int_so1_output_ptr:y          ;
        r0  = r0 - (mp3_sample_pcm_buff2 + 576*2)       ;
        if(r0 < 0 ) jmp _wait_gr1                       ;
        if(r1 == 0) call mp3_sample_copy_mono           ;
        if(r1 != 0) call mp3_sample_copy_stereo         ;
_wait_gr2:
        clr(r0)                                         ;
        r0l = *mp3_sample_int_so1_output_ptr:y          ;
        r0  = r0 - (mp3_sample_pcm_buff2 + 576*2)       ;
        if(r0 >= 0 ) jmp _wait_gr2                      ;
        if(r1 == 0) call mp3_sample_copy_mono           ;
        if(r1 != 0) call mp3_sample_copy_stereo         ;
```

(5/5)

```
/****************  jump mp3_sample_main  ****************/
mp3_mon_decode_result:
        jmp mp3_sample_main                             ;

/****************** check end of file *******************/
input_bs_end_chk:
        r2 = *mp3_sample_dat_length:x                   ;
        r2l = *mp3_sample_dat_length+1:x                ;
        r1 = r2 - r3                                    ;
        if(r1 > 0) jmp skip_read_bs                     ;


/****************** decode finish  *******************/
mp3_mon_input_bs_finish:

/***************    clear output buffer   **************/
        clr(r0)                                         ;
        dp4 = mp3_sample_pcm_buff1                      ;
        rep MP3_DEC_OUTPUT_PCM_BUFFER_SIZE*4            ;
        *dp4++ = r0l                                    ;
        dp4 = mp3_sample_pcm_buff2                      ;
        rep MP3_DEC_OUTPUT_PCM_BUFFER_SIZE*4            ;
        *dp4++ = r0l                                    ;

input_bs_finish:
        clr(r0)                                         ;
        stk = r0l                                       ;
        r0l = stk                                       ;
        nop                                             ;
;       stop                                            ;
        nop                                             ;
        jmp 0x200                                       ;

/******************  error routine  *********************/

/******** *mp3_sample_error_frame_count:x += 1 *********/
mon_error:
        clr(r1)                                         ;
        r1l = *mp3_sample_error_frame_count:x           ;
        r1 = r1 + 1                                     ;
        *mp3_sample_error_frame_count:x = r1l           ;

; goto error type routine
        r1 = r0 ^ MP3_DEC_CRC_ERROR                     ;
        if(r1 ==0) jmp _error_return                    ;

        r1 = r0 ^ MP3_DEC_HEADDER_FOUND_ERROR           ;
        if(r1 ==0) jmp _header_error                    ;restartable error
; if(r1 ==0) jmp _error_finish                          ;fatal error

        r1 = r0 ^ MP3_DEC_BITSTREAM_ERROR               ;
        if(r1 ==0) jmp _error_return                    ;restartable error

        r1 = r0 ^ MP3_DEC_BIT_RATE_ERROR                ;
        if(r1 ==0) jmp _error_finish                    ;fatal error

        r1 = r0 ^ MP3_DEC_FREE_FORMAT_ERROR             ;
        if(r1 ==0) jmp _error_finish                    ;fatal error

        r1 = r0 ^ MP3_DEC_LAYER_ERROR                   ;
        if(r1 ==0) jmp _error_finish                    ;fatal error

        r1 = r0 ^ MP3_DEC_SAMPLE_RATE_ERROR             ;
        if(r1 ==0) jmp _error_finish                    ;fatal error

mp3_mon_error_finish:
_error_finish:

/****************  fatal error routine  *****************/
;       finish routine
        stop                                            ;
        halt                                            ;

/****************  restartable error ****************/
_header_error:
        r0  = *mp3_sample_remain_length:x               ;
        r0l = *mp3_sample_remain_length+1:x             ;
        r0  = r0 - (17*2)                               ;
        if(r0<=0) jmp input_bs_finish                   ;
_error_return:
;       if( r0 ==0 ) call 1frame_repeat                 ;1frame repeat
;       if( r0 !=0 ) call 1frame_skip                   ;1frame skip
;       call 1frame_mute                                ;
;       call mp3_sample_pcmbuf_wait_with_crc_error      ;

        jmp mp3_sample_main                             ;

END                                                     ;
```

- **sam_data.asm**

```
#include "m3d_dec.h"
#define MP3_SAMPLE_INPUT_BUFF_SIZE ( 1440 + 2 )


public mp3_sample_input_read_ptr                    ;user input buffer read pointer   (mp3_Dec use)
public mp3_sample_input_read_byte_flag              ;user input buffer read byte flag (mp3_Dec use)
public mp3_sample_input_write_ptr                   ;user input buffer write pointer

public mp3_sample_dat_length                        ;file length
public mp3_sample_remain_length                     ;remain data length
public mp3_sample_decode_length                     ;decoded data length
public mp3_sample_error_frame_count                 ;error counter

public mp3_sample_command_stop_flag                 ;user skop mode flag
public mp3_sample_command_skip_flag                 ;user skip mode flag

public mp3_sample_input_buff                        ;user input buffer
public mp3_sample_pcm_buff1                         ;user output buffer page1
public mp3_sample_pcm_buff2                         ;user output buffer page2

public mp3_sample_int_so1_output_ptr                ;user interrupt handler output pointer
public mp3_sample_int_so1_output_dmx                ;user interrupt handler output dmx
public mp3_sample_int_so1_output_dn                 ;user interrupt handler output dn
public mp3_sample_lsf_frame_odd                     ;user lsf mode page flag

public mp3_sample_int_saver0e                       ;
public mp3_sample_int_saver0h                       ;
public mp3_sample_int_saver0l                       ;

public mp3_sample_int_savedp4                       ;
public mp3_sample_int_savedn4                       ;
public mp3_sample_int_savedmy                       ;

public mp3_sample_int1_saver0e                      ;
public mp3_sample_int1_saver0h                      ;
public mp3_sample_int1_saver0l                      ;

__MP3_SAMPLE_INPUT_BUFF xramseg align
mp3_sample_input_buff: ds MP3_SAMPLE_INPUT_BUFF_SIZE ;

__MP3_SAMPLE_OUTPUT_PCM_BUFF1 yramseg align
mp3_sample_pcm_buff1:  ds MP3_DEC_OUTPUT_PCM_BUFFER_SIZE*4
__MP3_SAMPLE_OUTPUT_PCM_BUFF2 yramseg align
mp3_sample_pcm_buff2:  ds MP3_DEC_OUTPUT_PCM_BUFFER_SIZE*4
__MP3_SAMPLE_DATA_Y1 yramseg
mp3_sample_input_read_ptr: ds 1                     ;
mp3_sample_input_read_byte_flag:ds 1                ;
mp3_sample_input_write_ptr: ds 1                    ;


mp3_sample_command_stop_flag: ds 1                  ;
mp3_sample_command_skip_flag: ds 1                  ;

mp3_sample_int_so1_output_ptr: ds 1                 ;
mp3_sample_int_so1_output_dmx: ds 1                 ;
mp3_sample_int_so1_output_dn: ds 1                  ;
mp3_sample_lsf_frame_odd: ds 1                      ;

__MP3_SAMPLE_DATA_X1 xramseg

mp3_sample_dat_length:  ds 2                        ;file length
mp3_sample_remain_length: ds 2                      ;
mp3_sample_decode_length: ds 2                      ;decoded data length
mp3_sample_error_frame_count: ds 1                  ;error counter

mp3_sample_int_saver0e:  ds 1                       ;
mp3_sample_int_saver0h:  ds 1                       ;
mp3_sample_int_saver0l:  ds 1                       ;

mp3_sample_int_savedp4:  ds 1                       ;
mp3_sample_int_savedn4:  ds 1                       ;
mp3_sample_int_savedmy:  ds 1                       ;
mp3_sample_int1_saver0e: ds 1                       ;
mp3_sample_int1_saver0h: ds 1                       ;
mp3_sample_int1_saver0l: ds 1                       ;



end
```

• **sam_int.asm**

(1/3)

```
#include "m3d_dec.h"
#include "sample.h"
#include "m3d_err.h"


public mp3_sample_copy_mono                         ;
public mp3_sample_copy_stereo                       ;


_mp3_sample_intvector imseg at 0x0210
mp3_sample_int_INT1:
                call mp3_sample_command_stop        ;
                reti                                ;
                nop                                 ;
                nop                                 ;
mp3_sample_int_INT2:
                call mp3_sample_command_skip        ;
                reti                                ;
                nop                                 ;
                nop                                 ;
mp3_sample_int_INT3:
                nop                                 ;
                reti                                ;
                nop                                 ;
                nop                                 ;
mp3_sample_int_INT4:
                nop                                 ;
                reti                                ;
                nop                                 ;
                nop                                 ;

mp3_sample_int_si1:
;               call mp3_sample_int_si1_main        ;
                nop                                 ;
                reti                                ;
                nop                                 ;
                nop                                 ;
mp3_sample_int_so1:
                call mp3_sample_int_si1_main        ;
                reti                                ;
                nop                                 ;
                nop                                 ;
mp3_sample_int_si2:
                nop                                 ;
                nop                                 ;
                nop                                 ;
                nop                                 ;
mp3_sample_int_so2:
                nop                                 ;
                nop                                 ;
                nop                                 ;
                nop                                 ;
mp3_sample_int_HI:
                nop                                 ;
                reti                                ;
                nop                                 ;
                nop                                 ;
mp3_sample_int_HO:
                nop                                 ;
                reti                                ;
                nop                                 ;
                nop                                 ;



_mp3_sample_int imseg


mp3_sample_int_si1_main:
;       interrupt for PCM data output
;       save register
        *mp3_sample_int_saver0e:x = r0e             ;
        *mp3_sample_int_saver0h:x = r0h             ;
        *mp3_sample_int_saver0l:x = r0l             ;

        r0l = dp4                                   ;
        *mp3_sample_int_savedp4:x = r0l             ;

        r0l = dn4                                   ;
        *mp3_sample_int_savedn4:x = r0l             ;

        r0l = dmy                                   ;
        *mp3_sample_int_savedmy:x = r0l             ;
```

(2/3)

```
        r0l = *mp3_sample_int_so1_ptr:y              ;
        r0l = *mp3_sample_int_so1_output_ptr:y       ;
        dp4 = r0l                                    ;

;       r0l = *mp3_sample_int_so1_dmy:y              ;
;       dmy = r0l    ;
        dmy = (MP3_DEC_OUTPUT_PCM_BUFFER_SIZE*4) -1  ;
;       r0l = *mp3_sample_int_so1_dn4:y              ;
;       dn4 = r0l                                    ;
        dn4 = 0x01                                   ;

; output PCM DATA to SDT1 SDT2 port
        r0l = *dp4%%                                 ;
        *SDT1:y = r0l                                ;

        r0l = dp4                                    ;
;       *mp3_sample_int_so1_ptr:y = r0l              ;
        *mp3_sample_int_so1_output_ptr:y= r0l        ;


;       load register
        r0l = *mp3_sample_int_savedmy:x              ;
        dmy = r0l                                    ;

        r0l = *mp3_sample_int_savedn4:x              ;
        dn4 = r0l                                    ;

        r0l = *mp3_sample_int_savedp4:x              ;
        dp4 = r0l                                    ;

        r0l = *mp3_sample_int_saver0l:x              ;
        r0h = *mp3_sample_int_saver0h:x              ;
        r0e = *mp3_sample_int_saver0e:x              ;

        ret                                          ;

/****************     user command      *****************/

/****************  stop command   *****************/
mp3_sample_command_stop:
        *mp3_sample_int1_saver0e:x = r0e             ;
        *mp3_sample_int1_saver0h:x = r0h             ;
        *mp3_sample_int1_saver0l:x = r0l             ;

        clr(r0)                                      ;
        r0l = *mp3_sample_command_stop_flag:y        ;
        if(r0 == 0) jmp _stop_off                    ;
_stop_on:
        r0l = 0x01                                   ;
        *mp3_sample_command_stop_flag:y = r0l        ;
; stop output interrupt
        r0l = SR                                     ;
        r0 = r0 | 0x00f0                             ; mask si1 so1 si2 so2
        sr = r0l                                     ;
        jmp _end_command                             ;
_stop_off:
        clr(r0)                                      ;
        *mp3_sample_command_stop_flag:y = r0l        ;
; run output interrupt
        r0l = SR                                     ;
        r0 = r0 & 0xff0f                             ; mask canncel si1 so1 si2 so2
        sr = r0l                                     ;
        jmp _end_command                             ;

/*****************  skip command  *****************/
mp3_sample_command_skip:
        *mp3_sample_int1_saver0e:x = r0e             ;
        *mp3_sample_int1_saver0h:x = r0h             ;
        *mp3_sample_int1_saver0l:x = r0l             ;

        clr(r0)                                      ;
        r0l = *mp3_sample_command_skip_flag:y        ;
        if(r0 != 0) jmp _skip_off                    ;
_skip_on:
        r0l = 0x01                                   ;
        *mp3_sample_command_skip_flag:y = r0l        ;
        jmp _end_command                             ;

_skip_off:
        clr(r0)                                      ;
        *mp3_sample_command_skip_flag:y = r0l        ;

_end_command:                                        ;
        r0l = *mp3_sample_int1_saver0l:x             ;
        r0h = *mp3_sample_int1_saver0h:x             ;
        r0e = *mp3_sample_int1_saver0e:x             ;

        ret                                          ;
```

(3/3)

```
/*****************  copy decoded PCM data  *****************/

mp3_sample_copy_mono:
;       copy data 1gra
;       mono to stereo モノラルなので、同じデータを2こずつコピー
;       dp4 = mp3_sample_pcm_buff1                      ;
;       dp5 = mp3_sample_pcm_buff2                      ;
;       dn5 = 0x01                                      ;
;       dmy = 576 * 2                                   ;
;
        loop 576 {                                      ;
                r0l = *dp4++                            ;
        *dp5%% = r0l                                    ;
        *dp5%% = r0l                                    ;
        }                                               ;
        ret                                             ;


mp3_sample_copy_stereo:
;       copy 1 gra pcm data
;
;       dp4 = mp3_sample_pcm_buff1  ;
;       dp5 = mp3_sample_pcm_buff2  ;
;       dn5 = 0x01                                      ;
;       dmy = 576 * 2                                   ;
;
        loop 576*2 {                                    ;
                r0l = *dp4++                            ;
                *dp5%% = r0l                            ;
        }                                               ;
        ret                                             ;

end
```

〔メ　モ〕

〔メ　モ〕

〔メ　モ〕

お問い合わせ先

【技術的なお問い合わせ先】

ＮＥＣ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

【営業関係お問い合わせ先】

システムLSI第一営業事業部

東　京	(03)3798−6106, 6107, 6108, 6155
大　阪	(06)6945−3178, 3200, 3208
名古屋	(052)222−2375
仙　台	(022)267−8740
水　戸	(029)226−1702
広　島	(082)242−5504
鳥　取	(0857)27−5313
松　山	(089)945−4149

システムLSI第二営業事業部

東　京	(03)3798−6110, 6111, 6112, 6151, 6156
名古屋	(052)222−2170, 2190
松　本	(0263)35−1662
前　橋	(027)243−6060
立　川	(042)526−5981
静　岡	(054)254−4794
金　沢	(076)232−7303
福　岡	(092)261−2806

【資料の請求先】

上記営業関係お問い合わせ先またはNEC特約店へお申しつけください。

【NECエレクトロンデバイス　ホームページ】

NECエレクトロンデバイ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.ic.nec.co.jp/

アンケート記入のお願い

お手数ですが，このドキュメントに対するご意見をお寄せください。今後のドキュメント作成の参考にさせていただきます。

［ドキュメント名］ μSAP77016-B07　ユーザーズ・マニュアル

（U15134JJ4V0UM00（第 4 版））

［お名前など］（さしつかえのない範囲で）

御社名（学校名，その他）　（　　）
ご住所　（　　）
お電話番号　（　　）
お仕事の内容　（　　）
お名前　（　　）

1．ご評価（各欄に○をご記入ください）

項　　目	大変良い	良　い	普　通	悪　い	大変悪い
全体の構成					
説明内容					
用語解説					
調べやすさ					
デザイン，字の大きさなど					
その他（　　）					
（　　）					

キリトリ

2．わかりやすい所（第　章，第　章，第　章，第　章，その他　　）

理由［　　］

3．わかりにくい所（第　章，第　章，第　章，第　章，その他　　）

理由［　　］

4．ご意見，ご要望

5．このドキュメントをお届けしたのは

NEC 販売員，特約店販売員， その他（　　）

ご協力ありがとうございました。

下記あてに FAX で送信いただくか，最寄りの販売員にコピーをお渡しください。

日本電気（株） NEC エレクトロンデバイス
半導体テクニカルホットライン
FAX：（044）435-9608

2000.6