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-B11

WMA デコーダ・ミドルウエア

対象デバイス

μPD77113A
μPD77114
μPD77210
μPD77213

資料番号 U15683JJ1V1UM00（第 1 版）
発行年月 January 2002 NS CP（N）

© NEC Corporation 2001

〔メ　モ〕

目次要約

Windows および Windows Media 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

本版で改訂された主な箇所

箇　所	内　容
p.28	**図** 2ー3　**スクラッチ領域確保の例** **プログラムを修正。**
p.36	4.3　**シミュレーション方法** **（2）**HSM77016**（ハイスピード・シミュレータ）を起動します。** **あとに続く例を削除。**

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，μPD77113A, 77114, 77210, 77213 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 A　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
μPD77113A	U12395J	U14373J	U14623J	U13116J	U11958J	U12021J
μPD77114						
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

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-B11 は，Windows Media ™ Audio（WMA）方式のデコード機能を提供するミドルウエアです。

1.2 WMA デコーダ

Windows Media Audio は，マイクロソフト社が推進するオーディオ／ビデオ・ストリーミング配信技術「Windows Media Technology（WMT）」のオーディオ規格で，マイクロソフト社標準の ASF（Advanced Systems Format）フォーマットに基づいており，MPEG-4（Moving Picture Experts Group），WMV（Windows Media Video）などと共に用いられています。

μSAP77016-B11 は，WMA の復号方式に従ったものです。

1.3 製品概要

1.3.1 特　　徴

- Windows Media Audio CODEC Version 2, 7, 8 のすべてのビット・レート，サンプリング周波数の圧縮データをデコード可能（**表 1−1　各コーデックの対応ビット・レートとサンプリング周波数**参照）
- デコード結果は 16 ビット・リニア PCM 形式で出力
- ASF ファイル形式に対応
- ビデオを含むデータからオーディオ・データのみを抜き出してのデコードが可能
- コンテンツ情報の読み出しが可能
- DRM（Digital Rights Management），スクリプト・コマンドおよびマーカ機能は未サポート

表1−1　各コーデックの対応ビット・レートとサンプリング周波数

ビット・レート [bps]	サンプリング周波数 [Hz]	チャネル数	WMA CODEC Version 8	WMA CODEC Version 7	WMA CODEC Version 2
192,000	48,000	2	○	−	−
160,000	48,000	2	○	○	○
128,000	48,000	2	○	○	○
192,000	44,100	2	○	○	−
160,000	44,100	2	○	○	○
128,000	44,100	2	○	○	○
96,000	44,100	2	○	○	○
80,000	44,100	2	○	○	○
64,000	44,100	2	○	○	○
48,000	44,100	2	○	−	○
32,000	44,100	2	○	−	−
48,000	44,100	1	○	−	−
32,000	44,100	1	○	○	○
64,000	32,000	2	−	−	○
48,000	32,000	2	○	○	○
44,000	32,000	2	−	−	○
40,000	32,000	2	○	○	○
36,000	32,000	2	−	−	○
32,000	32,000	2	○	○	○
22,000	32,000	2	−	−	○
32,000	32,000	1	−	−	○
20,000	32,000	1	○	○	○
32,000	22,050	2	○	○	○
22,000	22,050	2	○	○	○
20,000	22,050	2	○	○	○
20,000	22,050	1	○	○	○
16,000	22,050	1	○	○	○
20,000	16,000	2	○	○	○
16,000	16,000	2	○	○	○
16,000	16,000	1	○	○	○
12,000	16,000	1	○	○	○
10,000	16,000	1	○	○	○
10,000	11,025	1	○	○	○
8,000	11,025	1	○	○	○
12,000	8,000	2	○	○	○
8,000	8,000	1	○	○	○
6,000	8,000	1	○	○	○
5,000	8,000	1	○	○	○
128[注]	8,000	1	○	○	○

注　video のみのデータをエンコードする際などに選択されることがありますが，有効な Audio データは含まれません。

備考　○：対応，−：存在しない組み合わせ

1. 3. 2　動作環境

(1) 動作対象 DSP：

μPD77113A, 77114, 77210, 77213

(2) 必要メモリ・サイズ：

μSAP77016-B11 は，全ビット・レートに対応するために，次の表のメモリ・サイズを必要とします。

表 1－2　必要メモリ・サイズ

メ モ リ	種　　別	サイズ [K ワード]
命令メモリ	－	12.6
X メモリ	RAM	10.5
	ROM	17.5
Y メモリ	RAM	12.5
	ROM	9.6

注意 1.　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。

2.　PCM データおよびビット・ストリーム・データのバッファは含まれていません。また，対応するビット・レートを制限することで，必要なメモリ・サイズを削減することが可能です。詳細は，2. 5　メモリ構成を参照してください。

(3) ソフトウエア・ツール（Windows®**版）：**

表 1－3　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） LB77016（ライブラリアン）

1.3.3 性　能

WMA デコード処理をリアル・タイムに実行するために必要な MIPS 値を，表 1－4　WMA デコード処理に必要な MIPS 値に示します。

【測定条件】

- シミュレータ：HSM77016
- 評価結果：WMA ファイルをそれぞれデコードしたときの演算量を測定し，その平均値と最大値を求めます。
- デコード処理単位（サンプル数），出力バッファ・サイズは，表 2－5　推奨出力バッファ・サイズに示した値とします。
- MIPS 値はデコードに使用する wmad_FileDecodeData 関数と wmad_FileGetPCM 関数，および wmad_FileCBGetData 関数のみの演算量です。そのほか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。ユーザ定義の wmad_FileCBGetData 関数はサンプルに添付したものを使用します。システム構成によっては演算量が変化する場合があります。

表 1－4　WMA デコード処理に必要な MIPS 値

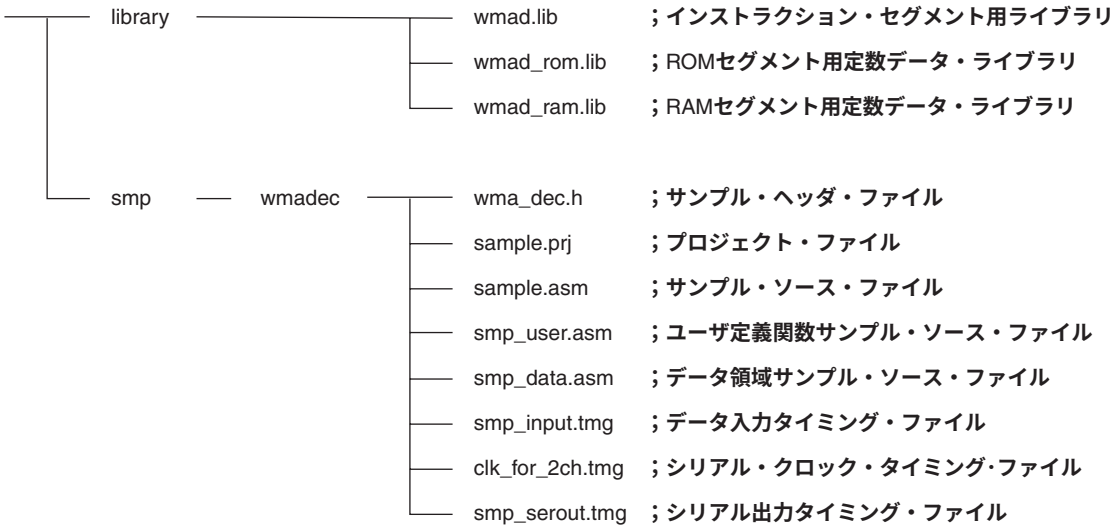
デコード条件			演算量	
ビット・レート [kbps]	サンプリング周波数 [kHz]	チャネル数	平均 MIPS 値 [MIPS]	最大 MIPS 値 [MIPS]
22	22	2	28	51
22	32	2	44	74
32	32	2	27	47
192	48	2	41	71

注意　データによっては最大 MIPS 値が表 1－4 の値よりも大きくなる可能性があります。
たとえば，デコード処理をリアルタイムに実行できなかった区間は無音を出力するなどの処置を，システム側で実装することを推奨します。

1.3.4 ディレクトリ構成

μSAP77016-B11 のディレクトリ構成を示します。

図 1－1　μSAP77016-B11 のディレクトリ構成



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。

- wmad.lib　：インストラクション・セグメント用ライブラリ・ファイル。
- wmad_rom.lib　：ROM セグメント用定数データ・ライブラリ・ファイル。
  定数データを ROM に配置する場合に選択してください。
- wmad_ram.lib　：RAM セグメント用定数データ・ライブラリ・ファイル。
  定数データを RAM に配置する場合に選択してください。

（2）smp/wmadec

サンプル・プログラムのソース・ファイル，ヘッダ・ファイル，シミュレート用のタイミング・ファイルを格納しています。タイミング・ファイルを使用することでハイスピード・シミュレータによるシミュレーションを行うことができます（**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**参照）。

第2章　ライブラリ仕様

この章では，μSAP77016-B11 の関数仕様と呼び出し規約について説明します。

2. 1　ライブラリ概要

μSAP77016-B11 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
wmad_FileDecodeInit	デコーダ初期化処理
wmad_FileDecodeData	デコード処理
wmad_FileGetPCM	デコード処理
wmad_GetVersion	バージョン情報取得
wmad_FileDecodeInfo	ファイル情報取得

また，μSAP77016-B11 の動作には，ユーザが次の関数を用意する必要があります。

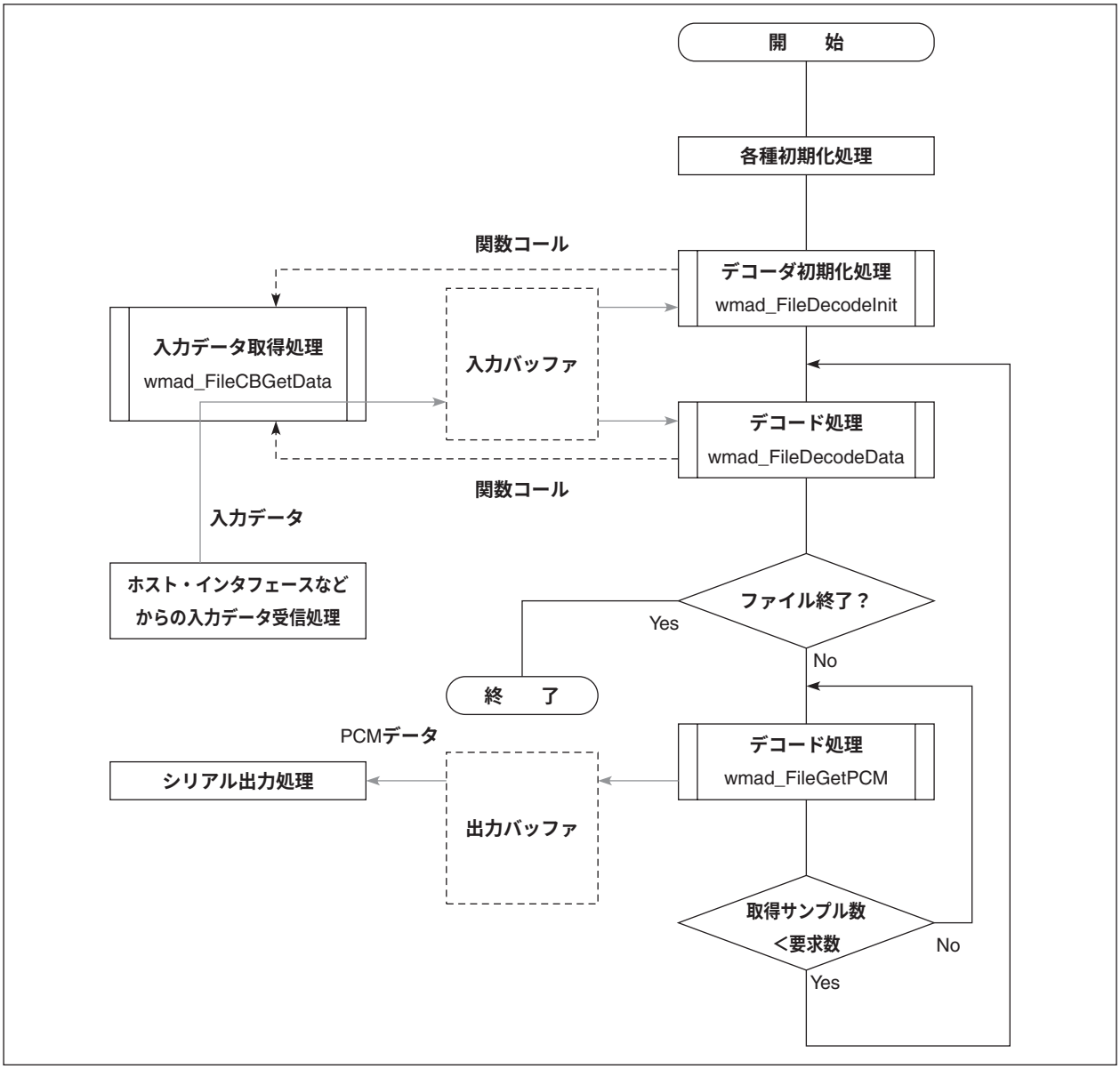
表 2－2　ユーザ定義関数一覧

関 数 名	機　　能
wmad_FileCBGetData	入力データ取得関数

2. 2　アプリケーション処理フロー

アプリケーション処理の例を次に示します。

図 2－1　アプリケーション処理フロー



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　wmad_FileDecodeInit 関数

【　分　類　】WMA デコーダ初期化処理関数

【 関 数 名 】wmad_FileDecodeInit

【 機 能 概 要 】μSAP77016-B11 で使用する RAM 領域の初期化および各種パラメータを設定します。

【　形　式　】call　wmad_FileDecodeInit

【　引　数　】

引　　数	説　　明
R0L	ユーザ定義の wmad_FileCBGetData 関数の先頭アドレス
R1	コンテンツ情報取得の有無 1: 取得する　0: 取得しない
R2L	コンテンツ情報を格納する構造体（X メモリ）の先頭アドレス

【 返 却 値 】

返 却 値	説　　明
R0	エラー・コード（エラー・コードの内容は，2. 4　エラー情報を参照）

【　機　能　】μSAP77016-B11 で使用する各種パラメータの設定，RAM 領域の初期化を行ないます。ビット・ストリーム・データがコンテンツ情報を持つ場合はコンテンツ情報を取得可能です。コンテンツ情報を格納する領域（構造体）は，表 2－7　コンテンツ情報構造体のメンバを参照して，あらかじめ用意してください。

【使用レジスタ】R0, R1, R2, R3, R4, R5, R6, R7,
DP0, DP1, DP2, DP3, DP4, DN0

【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	2
最大リピート回数	502
最大サイクル数	5.5×10^4

【　備　考　】この関数は wmad_FileCBGetData 関数をコールするため，wmad_FileCBGetData 関数でコール・スタックを使用する場合は，その分だけ最大スタック・レベルが増加します。また，最大リピート回数と最大サイクル数は，wmad_FileCBGetData 関数のリピート回数，サイクル数に依存します。上記の数値はサンプル・ソースの wmad_FileCBGetData 関数を使用し，すべてのコンテンツ情報を各 40 文字取得した場合の値です。最大サイクル数はコンテンツ情報の取得文字数，マーカ情報の有無などによっても変化します。

2. 3. 2　wmad_FileDecodeData 関数

【　分　類　】WMA デコード処理関数

【 関 数 名 】wmad_FileDecodeData

【機 能 概 要】ビット・ストリーム・データをデコードし，PCM データ作成に必要なデータを作成します。

【　形　式　】call　wmad_FileDecodeData

【　引　数　】なし

【 返 却 値 】

返却値	説　明
R0	エラー・コード（エラー・コードの内容は，2. 4　エラー情報を参照）

【　機　能　】ビット・ストリーム・データをデコードし，PCM データ作成に必要なデータをスタティック領域に格納します。デコード終了時にはエラー・コード cWMA_Failed または cWMA_NoMoreFrames を返します。なお，エラー・コードが cWMA_NoErr ではない場合，それ以降，wmad_FileGetPCM 関数を実行する必要はありません。この関数を再度実行する場合は，wmd_FileGetPCM 関数を実行し，そのときに作成できるすべての PCM データを作成したあとに行ってください。

【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, DN4, DN5, DN6, DN7

【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	25
最大サイクル数	2.7×10^6

【　備　考　】この関数は wmad_FileCBGetData 関数をコールするため，wmad_FileCBGetData 関数でコール・スタックを使用する場合は，その分だけ最大スタック・レベルが増加します。また，最大リピート回数と最大サイクル数は，wmad_FileCBGetData 関数のリピート回数，サイクル数に依存します。上記の数値はサンプル・ソースの wmad_FileCBGetData 関数を使用した場合の値です。

2. 3. 3　wmad_FileGetPCM 関数

【　分　類　】WMA デコード処理関数

【 関 数 名 】wmad_FileGetPCM

【 機 能 概 要 】wmad_FileDecodeData のデコード結果から PCM データを作成します。

【　形　式　】call　wmad_FileGetPCM

【　引　数　】

引　　数	説　　明
R0L	出力バッファ（X メモリ）の先頭アドレス
R1L	要求 PCM サンプル数（チャネル当たり）

【 返 却 値 】

返却値	説　　明
R1	取得 PCM サンプル数（チャネル当たり）

【　機　能　】スタティック領域に格納されたデコード結果を PCM 形式のデータに変換し，要求 PCM サンプル数×チャネル数の PCM データを X メモリの指定バッファに格納します。データが 2 チャネルの場合は L チャネル，R チャネルの順で交互に格納されます。引数の要求 PCM サンプル数は PCM バッファのサイズと同じか，それよりも小さいサンプル数（1 以上）を指定してください。返却値の取得 PCM サンプル数が要求 PCM サンプル数より小さい値であった場合は，スタティック領域に格納されたデータをすべてデコード終了しています。次の PCM データを取得するには，再度 wmad_FileDecodeData 関数を実行してください。なお，要求 PCM サンプル数を少なくすることで，ユーザ定義の出力バッファ・サイズを縮小することが可能です。

【使用レジスタ】R0, R1, R2, R3, R4, R5, R6, R7,
DP0, DP1, DP2, DP3, DP4, DP5,
DN0, DN2, DN5, DN7

【ハードウエア・リソースメント】

最大スタック・レベル	4
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	2.1×10^5

【　備　考　】最大サイクル数は，チャネル当たりの取得 PCM サンプル数が 2048 の場合の値です。

2. 3. 4　wmad_GetVersion 関数

【　分　類　】バージョン情報取得関数

【 関 数 名 】wmad_GetVersion

【機 能 概 要 】ライブラリのバージョンおよび対応する Windows Media Player のバージョンを返します。

【　形　式　】call　wmad_GetVersion

【　引　数　】なし

【 返 却 値 】

返却値	説　明
R0H	このライブラリのメージャ・バージョン番号
R0L	このライブラリのマイナ・バージョン番号
R1H	対応する Windows Media Player のメージャ・バージョン番号
R1L	対応する Windows Media Player のマイナ・バージョン番号

【　機　能　】このライブラリのバージョン番号と対応する Windows Media Player 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，ライブラリ・バージョン：V1.01

R1=0x00’0x0007’0x0000 の場合，Windows Media Player バージョン：V7.0

【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	10

2. 3. 5　wmad_FileDecodeInfo 関数

【　分　類　】ファイル情報取得獲得関数

【 関 数 名 】wmad_FileDecodeInfo

【 機 能 概 要 】WMA ファイルのファイル情報を取得します。

【　形　式　】call　wmad_FileDecodeInfo

【　引　数　】

引　　数	説　　明
R0L	ファイル情報を格納する構造体（X メモリ）の先頭アドレス

【 返 却 値 】

返却値	説　　明
R0	エラー・コード（エラー・コードの内容は，2. 4　エラー情報を参照）

【　機　能　】ビット・レート，サンプリング周波数などのファイル情報を取得し，ファイル情報構造体に結果を格納します。この関数は wmad_FileDecodeInit 関数を実行したあとに実行してください。その際，あらかじめファイル情報を格納する領域（構造体）を X メモリに用意し，その先頭アドレスを引数としてください。ファイル情報を格納する構造体の内容については，表 2－6　ファイル情報構造体のメンバを参照してください。

【使用レジスタ】R0, R1, R2, DP0

【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	41

2. 3. 6　wmad_FileCBGetData 関数（ユーザ定義関数）

【　分　類　】入力データ取得関数

【　関　数　名　】wmad_FileCBGetData

【　機　能　概　要　】デコードに必要なビット・ストリーム・データを入力バッファに補充します。

【　形　式　】wmad_FileDecodeInit 関数および wmad_FileDecodeData 関数が，この関数を call DP0 の形式でコールします。

【　引　数　】

引　数	説　明
R0	要求データ・サイズ [バイト]
R1	ビット・ストリーム・データの先頭からのオフセット [バイト]

【　返　却　値　】

返却値	説　明
R0	取得データ・サイズ [バイト]
R2L	入力バッファの先頭アドレス

【　機　能　】この関数は，wmad_FileDecodeInit関数およびwmad_FileDecodeData関数からそれぞれ複数回コールされ，WMAのデコードに必要なビット・ストリーム・データを入力バッファに補充します。

なお，この関数の先頭アドレスは wmad_FileDecodeInit 関数で設定してください。

【使用可能レジスタ】R0, R1, R2, R3, R4, R5, DP0, DP1

注意　上記以外のレジスタを使用する場合は，レジスタの内容をメモリに退避してからご使用ください。

【使用可能ハードウエア・リソースメント】

スタック・レベル	0〜7
ループ・スタック・レベル	1

【　備　考　】この関数内でリピート命令を使用する際，リピート回数が大きいと割り込み処理に遅れが発生する場合があります。また，この関数の実行サイクル数が大きいと，デコード処理をリアル・タイムに行えないことがあります。この関数のスタック・レベルは，wmad_FileCBGetData 関数，wmad_FileDecodeData 関数およびその他ユーザが使用するスタック・レベルとの合計が 15 以下になるように設定してください。

【関数の要求仕様】・ビット・ストリーム・データの先頭からのオフセット位置（引数：R1 バイト目）からデータ要求サイズ（引数：R0 バイト）のデータを，X メモリにあるユーザ定義の入力バッファに格納し，取得したデータのサイズをレジスタ R0 に，入力バッファの先頭アドレスをレジスタ R2L に設定してください（**図 2－2　入力バッファへのデータ設定方法**参照）。ビット・ストリーム・データを格納する入力バッファの必要最小サイズは 64 ワード（128 バイト）となります（**2. 5　メモリ構成**参照）。

・引数 R0 の値は，デコード処理に必要なデータを取得するときには，1 以上 128 以下となります。この値が 128 を越える場合は，デコード処理に必要のないビット・ストリーム・データを読み飛ばす場合ですので，入力バッファにデータを設定する必要はありません。ただし，返却値 R0 には取得したデータ・サイズを設定してください。

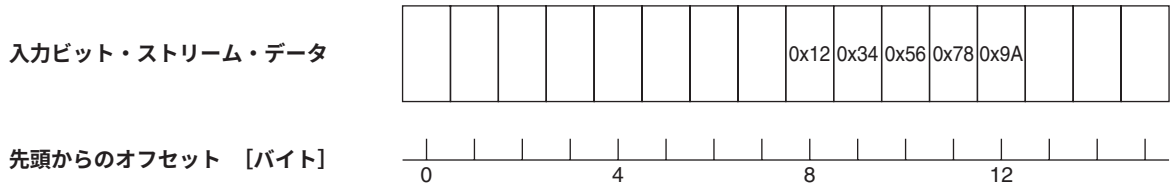
・引数 R1 の値は，前回 wmad_FileCBGetData 関数がコールされたときの R1 の値以下にはなりません。初めて wmad_FileCBGetData 関数がコールされたときの R1 の値は 0 となります。少なくとも，前回 wmad_FileCBGetData 関数がコールされたときの R1 の値までのビット・ストリーム・データはデコード処理が完了したことになります。

・引数 R1 の値が，前回 wmad_FileCBGetData 関数がコールされたときに読み込んだビット・ストリーム・データの終端（前回の引数 R1＋R0 の値）以下の場合（Case A）には，前回入力バッファに設定したデータのうち最初の 1 バイトを除いたものが必要になります[注]。したがって，wmad_FileCBGetData 関数がコールされたときに入力バッファに設定したデータのうち最初の 1 バイトを除いたものは，次回 wmad_FileCBGetData 関数がコールされるまで保存しておいてください。なお，μSAP77016-B11 が正常に動作していれば，ユーザの設定した入力バッファの内容を変更することはありません。

注　前回の引数 R0 が 128 を越えていた場合には，Case A になることはありません。

・引数 R1 の値が，前回 wmad_FileCBGetData 関数がコールされたときに読み込んだビット・ストリーム・データの終端（前回の引数 R1＋R0 の値）よりも大きい場合には，オフセットが R1 の値未満のデータを使用することはありません。

図 2－2　入力バッファへのデータ設定方法



引数が
オフセット = 8
要求サイズ = 3
の場合

入力バッファ

MSB	0x12	0x34	LSB
	0x56	＊＊	
	＊＊	＊＊	
	＊＊	＊＊	

＊＊は，任意

引数が
オフセット = 9
要求サイズ = 4
の場合

入力バッファ

MSB	0x34	0x56	LSB
	0x78	0x9A	
	＊＊	＊＊	
	＊＊	＊＊	

＊＊は，任意

2. 4　エラー情報

μSAP77016-B11 の関数が返却するエラー・コードの内容を次に示します。

表 2－3　エラー・コード一覧

エ ラ ー	値	内　　容
cWMA_NoErr	0	正常
cWMA_Failed	1	その他の異常
cWMA_BadArgument	2	初期化が正常に終了していません。
cWMA_BadAsfHeader	3	ASF Header が不正です。
cWMA_BadPacketHeader	4	PacketHeader が不正です。
cWMA_BrokenFrame	5	未使用
cWMA_NoMoreFrames	6	デコードすべきデータがありません。
cWMA_BadSamplingRate	7	未使用
cWMA_BadNumberOfChannels	8	未使用
cWMA_BadVersionNumber	9	未使用
cWMA_BadWeightingMode	10	未使用
cWMA_BadPacketization	11	未使用
cWMA_BadDRMType	12	未使用
cWMA_DRMFailed	13	未使用
cWMA_DRMUnsupported	14	DRM には対応していません。
cWMA_DemoExpired	15	未使用
cWMA_BadState	16	未使用
cWMA_Internal	17	内部エラー

2. 5　メモリ構成

スタティック領域やバッファの確保の仕方など，μSAP77016-B11 で使用するデータ･メモリの構成について説明します。

μSAP77016-B11 では，スクラッチ・メモリ領域とスタティック・メモリ領域を別々に定義する必要があります。それぞれのメモリ・サイズは表 2－4　各ビット・レートの必要メモリ・サイズを参照してください。

表 2－4　各ビット・レートの必要メモリ・サイズ（1/2）

ビット・レート	サンプリング	チャネル	WMA CODEC	X メモリ・サイズ［ワード］			Y メモリ・サイズ［ワード］		
[bps]	周波数 [Hz]	数	Version	static_x1	static_x2	scratch	static_y1	static_y2	scratch
192000	48000	2	8	741	0	1572	8192	4096	0
192000	44100	2	7, 8	741	0	1572	8192	4096	0
160000	48000	2	2, 7, 8	741	0	1572	8192	4096	0
160000	44100	2	2, 7, 8	741	0	1572	8192	4096	0
128000	48000	2	2, 7, 8	741	0	1572	8192	4096	0
128000	44100	2	2, 7, 8	741	0	1572	8192	4096	0
96000	44100	2	2, 7, 8	741	0	1572	8192	4096	0
80000	44100	2	2, 7, 8	741	0	1572	8192	4096	0
64000	44100	2	2, 7, 8	741	0	1572	8192	4096	0
64000	32000	2	2	741	0	1572	8192	4096	0
48000	44100	2	2, 8	741	0	1572	8192	4096	0
48000	32000	2	2, 7, 8	741	0	1572	8192	4096	0
44000	32000	2	2	972	0	1572	8192	4096	0
40000	32000	2	2, 7, 8	972	0	1572	8192	4096	0
36000	32000	2	2	972	0	1572	8192	4096	0
32000	44100	2	8	972	0	1572	8192	4096	0
48000	44100	1	8	741	0	1572	8192	2048	0
32000	44100	1	2, 7, 8	741	0	1572	8192	2048	0
32000	32000	2	2, 7, 8	972	0	1572	8192	4096	0
32000	32000	1	2	972	0	1572	8192	2048	0
32000	22050	2	2, 7, 8	741	0	1572	8192	2048	0
22000	32000	2	2	972	8192	1572	8192	4096	299
22000	22050	2	2, 7	972	4096	1572	8192	2048	299
22000	22050	2	8	972	0	1572	8192	2048	0
20000	32000	1	2, 7	972	4096	1572	8192	2048	299
20000	32000	1	8	972	0	1572	8192	2048	0
20000	22050	2	2, 7	972	4096	1572	8192	2048	299
20000	22050	2	8	972	0	1572	8192	2048	0
20000	22050	1	2, 7	972	2048	1572	8192	1024	299
20000	22050	1	8	972	0	1572	8192	1024	0
20000	16000	2	2, 7, 8	972	2048	1572	8192	1024	299
16000	22050	1	2, 7	972	2048	1572	8192	1024	299
16000	22050	1	8	972	0	1572	8192	1024	0
16000	16000	2	2, 7, 8	972	2048	1572	8192	1024	299
16000	16000	1	2, 7, 8	972	1024	1572	8192	512	299

表 2－4　各ビット・レートの必要メモリ・サイズ（2/2）

ビット・レート [bps]	サンプリング周波数 [hz]	チャネル数	WMA CODEC Version	X メモリ・サイズ［ワード］			Y メモリ・サイズ［ワード］		
				static_x1	static_x2	scratch	static_y1	static_y2	scratch
12000	16000	1	2, 7, 8	972	1024	1572	8192	512	299
12000	8000	2	2, 7, 8	972	2048	1572	8192	1024	299
10000	16000	1	2, 7, 8	972	1024	1572	8192	512	299
10000	11025	1	2, 7, 8	972	1024	1572	8192	512	299
8000	11025	1	2, 7, 8	972	1024	1572	8192	512	299
8000	8000	1	2, 7, 8	972	1024	1572	8192	512	299
6000	8000	1	2, 7, 8	972	1024	1572	8192	512	299
5000	8000	1	2, 7, 8	972	1024	1572	8192	512	299
128	8000	1	2, 7, 8	972	1024	1572	8192	512	299

2. 5. 1　スクラッチ領域

μSAP77016-B11 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-B11 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，ミドルウエア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，その設定情報は変更されることがありますので注意してください。

スクラッチ領域はレーベル名 wmad_lib_Scratch_x, wmad_lib_Scratch_y で確保してください。それぞれ align または at 指定は不要です。サイズは対応するビット・レートによって異なります。詳細は表 2－4　各ビット・レートの必要メモリ・サイズを参照してください。なお，定義したシンボルは必ず PUBLIC 宣言してください。

★ 図 2－3　スクラッチ領域確保の例

```
public      wmad_lib_Scratch_x
public      wmad_lib_Scratch_y

;すべてのビット・レートに対応する場合
#define WMAD_MAX_SCRATCH_X_SIZE 1572
#define WMAD_MAX_SCRATCH_Y_SIZE 299


__WMAD_LIB_SCRATCH_X     XRAMSEG
wmad_lib_Scratch_x:      DS  WMAD_MAX_SCRATCH_X_SIZE;

__WMAD_LIB_SCRATCH_Y     YRAMSEG
wmad_lib_Scratch_y:      DS  WMAD_MAX_SCRATCH_Y_SIZE;

end
```

2. 5. 2　スタティック領域

常にデータを保存しておく領域です。初期化処理以降にユーザがこの領域を操作した場合，このライブラリの正常な動作は保証されません。

スタティック領域はレーベル名 wmad_lib_Static_x1, wmad_lib_Static_x2, wmad_lib_Static_y1, wmad_lib_Static_y2 で確保してください。それぞれ，align または at 指定は不要です。サイズは対応するビット・レートによって異なります。詳細は表 2－4　各ビット・レートの必要メモリ・サイズを参照してください。なお，定義したシンボルは必ず PUBLIC 宣言してください。

図 2－4　スタティック領域確保の例

```
public      wmad_lib_Static_x1
public      wmad_lib_Static_x2
public      wmad_lib_Static_y1
public      wmad_lib_Static_y2

;すべてのビット・レートに対応する場合
#define WMAD_MAX_STATIC_X1_SIZE 972
#define WMAD_MAX_STATIC_X2_SIZE 8192
#define WMAD_MAX_STATIC_Y1_SIZE 8192
#define WMAD_MAX_STATIC_Y2_SIZE 4096

__WMAD_LIB_STATIC_X1     XRAMSEG
wmad_lib_Static_x1:      DS   WMAD_MAX_STATIC_X1_SIZE;
__WMAD_LIB_STATIC_X2     XRAMSEG
wmad_lib_Static_x2:      DS   WMAD_MAX_STATIC_X2_SIZE;

__WMAD_LIB_STATIC_Y1     YRAMSEG
wmad_lib_Static_y1:      DS   WMAD_MAX_STATIC_Y1_SIZE;
__WMAD_LIB_STATIC_Y2     YRAMSEG
wmad_lib_Static_y2:      DS   WMAD_MAX_STATIC_Y2_SIZE;

end
```

2. 5. 3　入出力バッファ

μSAP77016-B11 によりデコード処理を行うためには，ビット・ストリーム・データを入力する入力バッファ（X メモリ），およびデコード結果の PCM データを格納する出力バッファ（X メモリ）が必要です。これらのシンボル名は任意ですが，μSAP77016-B11 やほかのアプリケーションなどで使用されているものと重複しないように注意してください。

図 2－5　入出力バッファ確保の例

```
public      INPUT_BUFFER
public      HOST_IN_BUFF
public      OUTPUT_BUFFER
public      SERIAL_OUT_BUFF

I_O_BUFFER_X       XRAMSEG
INPUT_BUFFER:      DS     64;
HOST_IN_BUFF:      DS     1536;
OUTPUT_BUFFER:     DS     4096;
SERIAL_OUT_BUFF:   DS     4096;

end
```

入力バッファの必要最小サイズは，64 ワードです。また，出力バッファのサイズは，システム構成に合わせて設定します。推奨出力バッファ・サイズを**表 2－5　推奨出力バッファ・サイズ**に示します。

また，デコード処理をリアル・タイムに行う場合には，μSAP77016-B11 が使用する入出力バッファのほかに，たとえば，ビット・ストリーム・データをホスト CPU から受け取るための入力データ受信バッファ（X または Y メモリ）と，PCM データを DAC へ出力するためのシリアル出力バッファ（X または Y メモリ）などが必要となります。

シリアル出力バッファのサイズは，**表 2－5　推奨出力バッファ・サイズ**と同程度必要です。

入力データ受信バッファのサイズは次に示す平均入力データ数などを参考に，システムに合わせて設定してください。たとえば，推奨出力バッファ・サイズ×2 サンプルの出力結果を得るために必要な入力データを格納する場合，必要なサイズは，出力サンプル数として**表 2－5　推奨出力バッファ・サイズ**の値を代入して算出した平均入力データ数×2 ワード程度です。このように算出した平均入力データ数が最大となるのは 192 kbps, 44.1 kHz のデータの場合で，558 ワードとなります。

$$\text{平均入力データ数} = \frac{\dfrac{\text{ビット・レート [kbps]}/16}{\text{サンプリング周波数 [kHz]}}}{\text{出力サンプル数}} \text{[ワード]}$$

しかし，たとえば，Audio だけでなく Video 付きのデータも受信するような場合，ある程度多めにバッファを確保していても，バッファ内のデータが不足することが予測されます。この問題を回避するために，DSP 側から CPU へ入力データを要求するコマンドを発行できるようなシステムを構築することを推奨します。

表 2－5　推奨出力バッファ・サイズ

サンプリング周波数 [Hz]	サイズ [ワード]	
	mono	stereo
8000	512	1024
11025	512	1024
16000	512	1024
22050	1024	2048
32000	2048	4096
44100	2048	4096
48000	2048	4096

2. 5. 4　構 造 体

WMA ファイルのファイル情報やコンテンツ情報を取得したい場合には，次のような構造体を用意してください。

(1) ファイル情報構造体

WMA ファイルのファイル情報を取得したいときは次の構造体を X メモリに用意し，wmad_FileDecodeInfo 関数を実行してください。

表 2－6　ファイル情報構造体のメンバ

内　　容	サイズ [ワード]
WMA ファイル形式のバージョン[注]	1
サンプリング周波数 [Hz]	1
チャネル数	1
再生時間 [ms]	2
パケット・サイズ [バイト]	2
先頭パケットへのオフセット [バイト]	2
最終パケットへのオフセット [バイト]	2
DRM の有無　0: DRM なし，1: DRM あり	2
ビット・レート [bps]	2

注　μSAP77016-B11 は，WMA ファイル形式のバージョンが 2 のデータにのみ対応しています。WMA CODEC Version 2, 7, 8 でエンコードされたデータは，WMA ファイル形式のバージョンが 2 となります。

(2) コンテンツ情報構造体

曲のタイトル情報などを取得したいときは，次のコンテンツ情報構造体とコンテンツ文字列領域を X メモリに用意し，wmad_FileDecodeInit 関数を実行してください。

表 2－7　コンテンツ情報構造体のメンバ

内　　容	サイズ [ワード]
タイトル文字列の最大長 [バイト]	1
作成者文字列の最大長 [バイト]	1
著作者文字列の最大長 [バイト]	1
説明文文字列の最大長 [バイト]	1
規制文字列の最大長 [バイト]	1
タイトル文字列領域の先頭アドレス	1
作成者文字列領域の先頭アドレス	1
著作者文字列領域の先頭アドレス	1
説明文文字列領域の先頭アドレス	1
規制文字列領域の先頭アドレス	1

コンテンツ情報の最大長（要求文字数）は偶数で指定してください。ビット・ストリーム・データのコンテンツ情報が要求した文字数より短い場合は，構造体の文字列の最大長情報には実際に取得した文字列（終端コード“ 0x0000 ”を含む）の長さが格納されます。

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-B11（WMA デコーダ・ミドルウエア）のホスト・マシンへのインストール手順を次に示します。

（1）ホスト・マシンに作業用のディレクトリを作成します。

（2）提供媒体内のファイルおよびディレクトリをすべてホスト・マシンの作業用ディレクトリにコピーします。

3.2　サンプル作成手順

μSAP77016-B11 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。
エラー「Cannot Load system～*.model」が発生した場合は，「Options→Processor Model」で適切なモデル・ファイルを選択してください。
また，モデル・ファイルを変更した場合は，モデル・ファイルにあわせてプロジェクトの定数データ・ライブラリ・ファイルを変更してください。

例　ROM ありモデルの場合 wmad_ram.lib を wmad_rom.lib に変更します。

（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。

3.3　ロケーションの変更

μSAP77016-B11 が使用しているセグメント名を表 3－1　セグメント名に示します。LB77016（ライブラリアン）でライブラリ・ファイルからオブジェクト・ファイルを分離し，WB77016（ワークベンチ）のエディット・セグメント機能などを使用することで，ユーザのターゲットにあわせてロケーションを変更することが可能です。

なお，このライブラリおよびライブラリが使用するスタティック領域，スクラッチ領域は，外部メモリに配置しないでください。

表 3－1　セグメント名

セグメント名	説　　明
__WMAD_IMSEG＊	μSAP77016-B11 の命令セグメントです。
__WMAD_XROM＊	μSAP77016-B11 の X メモリ定数データ・セグメントです。
__WMAD_YROM＊	μSAP77016-B11 の Y メモリ定数データ・セグメントです。

備考　＊は任意の英数字とします。

3.4　シンボル命名規約

このライブラリ内で使用しているシンボルの命名規約を表 3－2　命名規約に示します。ほかのアプリケーションを組み合わせて使用するときは，重複しないように注意してください。

表 3－2　命名規約

分　　類	規　　約
関数名，変数名	wmad_XXXX
セグメント名	__WMAD_XXXX（先頭のアンダ・スコアは 2 つ）

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

サンプル・プログラムとサンプル・タイミング・ファイルを使用することで，デコード処理をシミュレーションすることが可能です。

【ソフトウエア環境例】

・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：smp_input.tmg, smp_serout.tmg, clk_for_2ch.tmg
・入力データ・ファイル：xxx.dat（作成方法は4. 2　入力データ・ファイルの作成参照）
・モデル・ファイル：uPD77113.model（サンプル・プログラム作成時のもの）

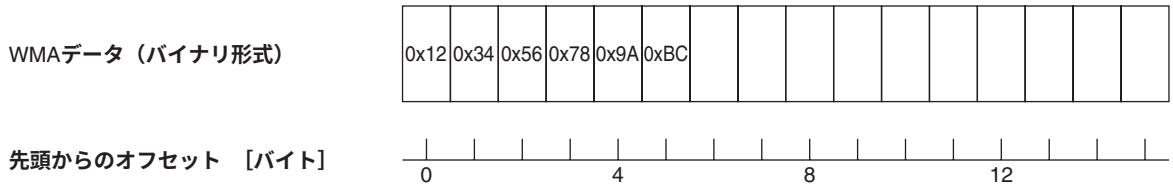
4. 2　入力データ・ファイルの作成

シミュレーションを行う際には入力データ・ファイルが必要です。入力データ・ファイルの作成手順を次に示します。

（1）任意の WMA データ（バイナリ形式）を用意します。

（2）用意した WMA データをテキスト・ファイルに変換します。テキスト・ファイルには，WMA データの先頭から順に 1 行に 2 バイトずつデータを記述してください。

図 4－1　入力データ・ファイル



入力データ・テキスト・ファイル
（テキスト形式）

4. 3　シミュレーション方法

次にシミュレーション方法について例を示します。

(1) 用意した入力データ・ファイルにあわせてタイミング・ファイルを編集します。（**A. 3　サンプル・タイミング・ファイル**参照）

★ (2) HSM77016（**ハイスピード・シミュレータ**）を起動します。

(3) **ターゲットにあわせてモデル・ファイルを選択します。**

例　「tools→Simulation Model」**でモデル・ファイルを選択します。**

(4) **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。**

(5) **次にタイミング・ファイル** smp_input.tmg, smp_serout.tmg, clk_for_2ch.tmg **を開きます。**

例　「file→open」**で各ファイルを指定します。**

(6) HSM77016（**ハイスピード・シミュレータ**）**の** CPU**，タイミング・ファイルなどをリセットします。**

例　「run→reset」**ですべての項目**（CPU and built-in I/O devices, time measurement, all timing files and restart execution）**を選択し，リセットします。**

(7) Run **で実行します。**

4.4　サンプル・プログラムの概要

4.4.1　サンプル・プログラムについて

μSAP77016-B11 のサンプル・プログラムは，入力データがホスト・インタフェース経由で DSP に送信されるシステムを想定して設計されています。その際，入力データは先頭からシーケンシャルに 16 ビット単位で送信されます。また，デコード処理単位（サンプル数）は**表 2−5　推奨出力バッファ・サイズ**の値とします。

4.4.2　ユーザ定義関数について

ユーザ定義関数サンプル・ソースの概要を示します。

【変数概要】

（1）変数 FileCBGetData_fp

ホスト・インタフェース経由で受信した入力データのバイト数を管理するための変数です。このシステムでは 2 バイトごとに受信するので，必ず偶数になります。

この変数の値 x は，今回の wmad_FileCBGetData 関数で，ホスト・インタフェース経由で最初に受信する入力データの先頭からのオフセットが x と x+1 のデータである，ということも意味します。なお，この変数の値は，「前回 wmad_FileCBGetData 関数がコールされたときの引数 R1＋R0 の値」ではありませんので，注意してください。

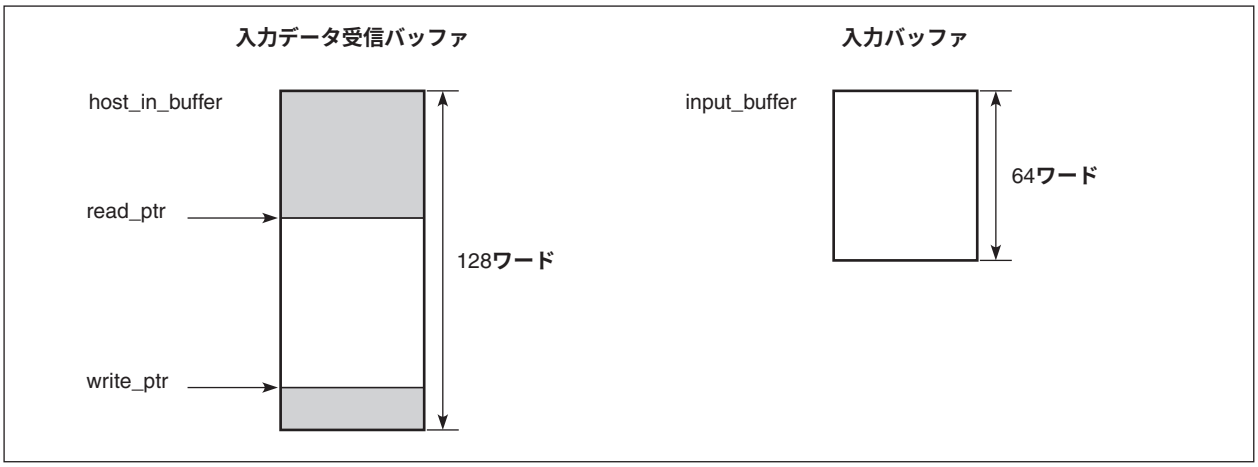
（2）変数 read_ptr

入力バッファにデータを設定するときに入力データ受信バッファからデータを読み出し始める位置を管理するための変数です。

（3）変数 write_ptr

ホスト・インタフェース経由で受信した入力データを入力データ受信バッファに書き込み始める位置を管理するための変数です。

図 4−2　サンプル・ユーザ定義関数のバッファ構成



【処理概要】

変数 FileCBGetData_fp の値と要求オフセット位置の値 r1 を比較し，

FileCBGetData_fp = r1 ならば，レーベル case000

FileCBGetData_fp＜r1 ならば，レーベル case001

FileCBGetData_fp＞r1 ならば，レーベル case002 へ，分岐します。

・レーベル case000：

最初にホスト・インタフェース経由で受信する 2 バイトの入力データのうちどちらかが，要求オフセットのデータと一致している場合です。そこで，変数 read_ptr に変数 write_ptr の値を設定します。また，レーベル get_data 以降の処理でホスト・インタフェース経由で受信するデータ・サイズを設定し，レーベル get_data へジャンプします。

・レーベル case001：

すでに受信した入力データよりも先の位置のデータを要求された場合です。そこで，要求オフセットのデータを受信する直前まで，ホスト・インタフェース経由でデータを受信します。ここで受信したデータは使用しませんので，入力データ受信バッファには保存しません。

次に，変数 read_ptr に変数 write_ptr の値を設定します。また，受信データ・サイズを設定し，レーベル get_data へジャンプします。

・レーベル case002：

この場合，要求オフセットのデータは，すでに受信しています。そこで，変数 read_ptr を設定しますが，すでに受信したデータを入力バッファに設定するため，変数 write_ptr の値から必要な数だけ巻き戻した位置を設定します。

次に，すでに受信したデータだけで要求サイズ分のデータを入力バッファに設定できるならば，レーベル set_data へ，すでに受信したデータだけでは要求サイズ分のデータを入力バッファに設定できないならば，新たに受信するデータ・サイズを設定し，レーベル get_data へジャンプします。

・レーベル get_data：

要求サイズが 128 バイト以下の場合，レーベル get_data_next へジャンプします。そうでない場合，ホスト・インタフェース経由で入力データを受信しますが，ここで受信したデータは使用しませんので，受信バッファには保存しません。受信処理が終了したあとは，レーベル finish へジャンプします。

・レーベル get_data_next：

受信すべきサイズが 0 の場合には，レーベル set_data へジャンプします。そうでない場合，ホスト・インタフェース経由で入力データを受信し，変数 write_ptr の位置から格納していきます。受信処理が終了したあとは，変数 write_ptr を更新し，レーベル set_data の処理を行います。

・レーベル set_data：

入力データ受信バッファに格納されている入力データを，変数 read_ptr の位置から読み出し，要求条件に合わせて入力バッファに設定します。次に，レーベル finish の処理を行います。

・レーベル finish：

返却値の設定，変数 FileCBGetData_fp の更新などを行い，wmad_FileCBGetData 関数の処理から呼び出し元へ復帰します。

4. 5　サンプル・プログラム処理フロー

サンプル・プログラムの処理フローを示します。

図 4－3　初期化，デコード処理フロー

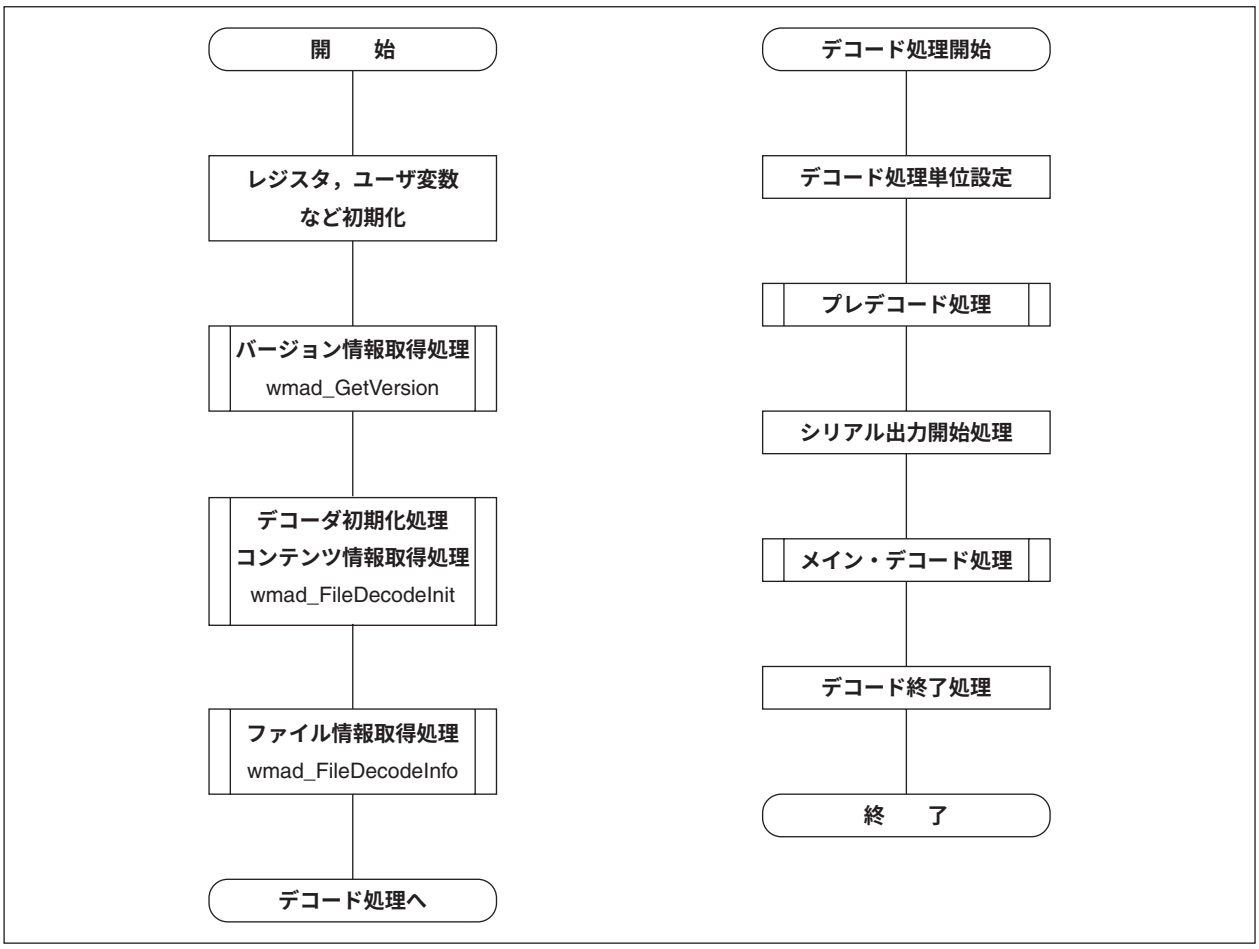


図 4−4　プレデコード処理フロー

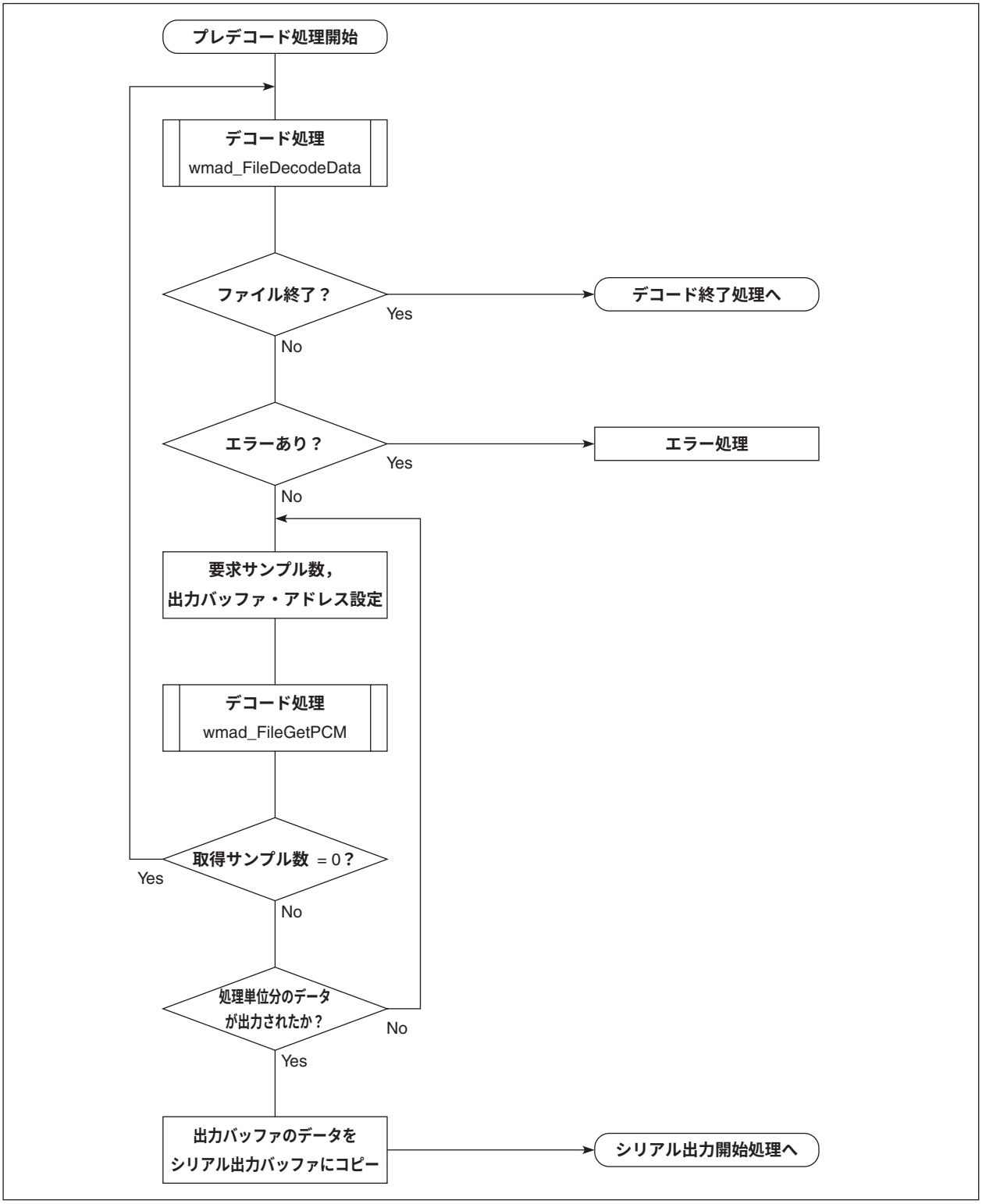
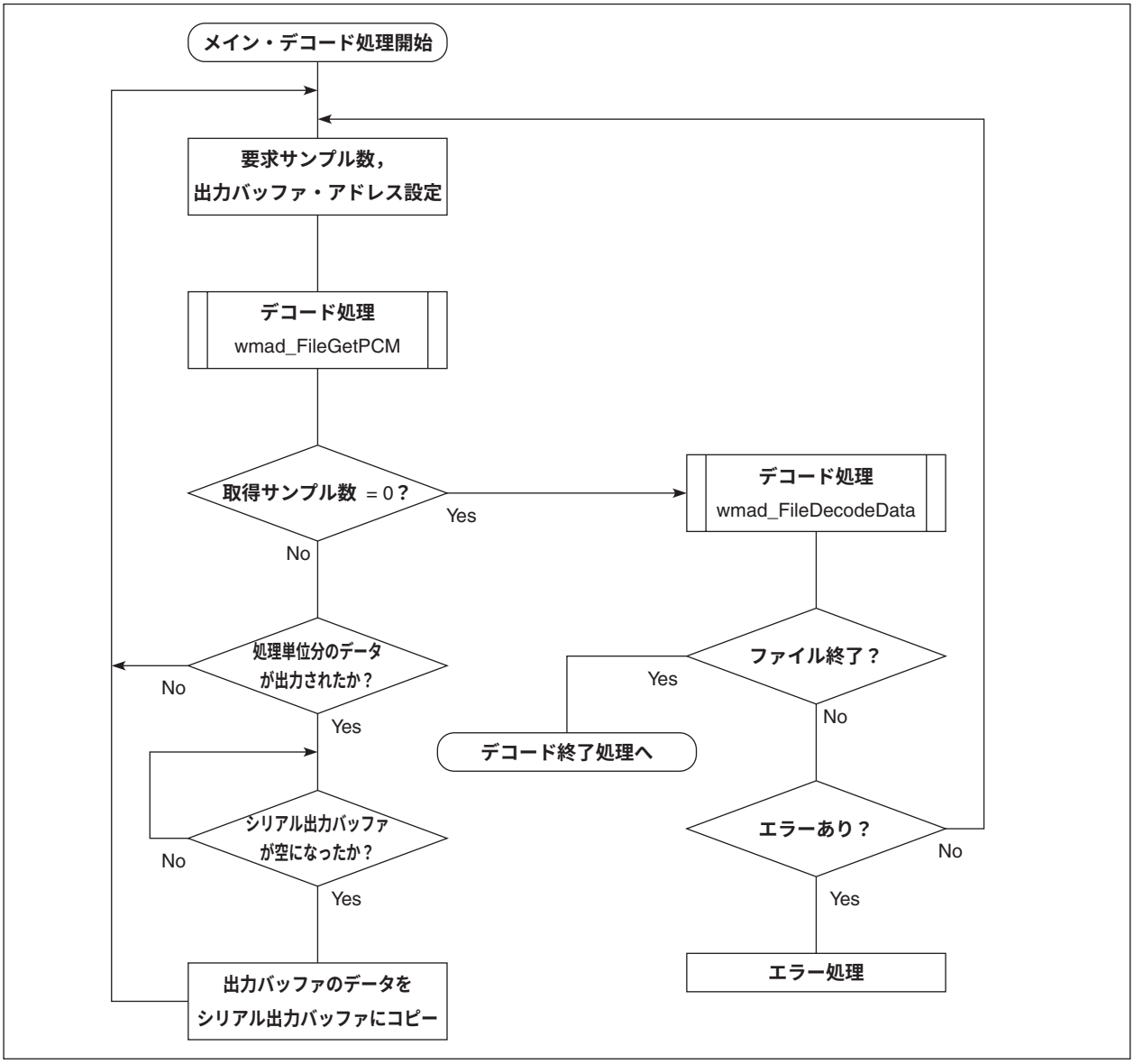


図 4−5　メイン・デコード処理フロー



付録 A　サンプル・プログラム・ソース

この章では，μSAP77016-B11 のサンプル・プログラム・ソースを示します。

A. 1　サンプル・ソース・ファイル

A. 1. 1　sample.asm

デコード処理全体を制御するためのファイルです。

(1/13)

```
/*-----------------------------------------------------------------*/
/*   File Information                                              */
/*-----------------------------------------------------------------*/
/*   Name      : sample.asm                                        */
/*   Type      : Assembler program module                          */
/*   Version   : 1.00                                              */
/*   Date      : 2001 July 10                                      */
/*   CPU       : uPD7701x Family                                   */
/*   Assembler : WB77016 Ver 2.4                                   */
/*   About     : sample main function                              */
/*                                                                 */
/*-----------------------------------------------------------------*/
/*   Copyright (C) NEC Corporation 2000, 2001                      */
/*   All rights reserved by NEC Corporation.                       */
/*   Use of copyright notice does not evidence publication         */
/*-----------------------------------------------------------------*/


/* ============================================================
 *   INCLUDE FILES
 * ============================================================ */
#include "wma_dec.h"

/* ============================================================
 *   PUBLIC
 * ============================================================ */
public Num_Channels                 ; for timing file
public sample_rate                  ; for timing file

/* ============================================================
 *   EXTERN FUNCTIONS
 * ============================================================ */
extrn  Init_FileCBGetData
extrn  wmad_FileCBGetData
```

(2/13)

```
/* ============================================================
 *   DEFINE & EQU
 * ============================================================ */
#define USE_SO       1              ; Use Serial Output interrupt
#define STRING_SIZE  64
#define MAX_PCMSIZE  2048
MAX_RINGSIZE   equ   MAX_PCMSIZE*2

/* ============================================================
 *      LOCAL VARIABLES AND BUFFER
 * ============================================================ */
__SAMPLE_X_RAM  XRAMSEG
ring_dn0:            ds      1              ;
ring_write_ptr:      ds      1              ;
ring_read_ptr:       ds      1              ;
ring_entries:        ds      1              ;
putin_temp:          ds      2              ;
save_regs:           ds      6              ;
g_ulOutputSample:    ds      2              ;
start_flag:          ds      1              ;
n_sample:            ds      1              ;
f_dec_unit_end:      ds      1              ;
n_get_pcm:           ds      1              ;


__SAMPLE_X_CONTENTS XRAMSEG
/*****************************************************
Contents Information
*****************************************************/
desc:
desc_title_len:              ds      1              ;
desc_author_len:             ds      1              ;
desc_copyright_len:          ds      1              ;
desc_description_len:        ds      1              ;
desc_rating_len:             ds      1              ;

desc_pTitle:                 ds      1              ;
desc_pAuthor:                ds      1              ;
desc_pCopyright:             ds      1              ;
desc_pDescription:           ds      1              ;
desc_pRating:                ds      1              ;

Title:                       ds      STRING_SIZE/2  ;
Author:                      ds      STRING_SIZE/2  ;
Copyright:                   ds      STRING_SIZE/2  ;
Description:                 ds      STRING_SIZE/2  ;
Rating:                      ds      STRING_SIZE/2  ;

StructFileInfo:
Version:                     ds      1              ;
```

(3/13)

```
sample_rate:         ds      1               ;
Num_Channels         ds      1               ;
duration:            ds      2               ;
packet_size:         ds      2               ;
first_packet_offset: ds      2               ;
last_packet_offset:  ds      2               ;
has_DRM:             ds      2               ;
bit_rate:            ds      2               ;

__SAMPLE_X_DEC_OUTPUT XRAMSEG
output_buffer:       ds      MAX_PCMSIZE*2   ;

__SAMPLE_X_SER_OUTPUT XRAMSEG align at 0
ser_out_buffer:      ds      MAX_RINGSIZE    ;



/* ============================================================
 *   VECTOR REGISTRATION
 * ============================================================ */
MAIN_V IMSEG at 0x200
     jmp star                                ; Regist start up routine
     nop                                     ;
     nop                                     ;
     nop                                     ;

; reserve vector 1
     nop                                     ;
     reti                                    ;
     nop                                     ;
     nop                                     ;

; reserve vector 2
     nop                                     ;
     reti                                    ;
     nop                                     ;
     nop                                     ;

; reserve vector 3
     nop                                     ;
     reti                                    ;
     nop                                     ;
     nop                                     ;

; int1 vector
     nop                                     ;
     reti                                    ;
     nop                                     ;
     nop                                     ;
```

(4/13)

```
; int2 vector
     nop                                ;
     reti                               ;
     nop                                ;
     nop                                ;

; int3 vector
     nop                                ;
     reti                               ;
     nop                                ;
     nop                                ;

; int4 vector
     nop                                ;
     reti                               ;
     nop                                ;
     nop                                ;

; SI1 vector
     nop                                ;
     reti                               ;
     nop                                ;
     nop                                ;

; SO1 vector
     jmp  _so_interrupt                 ; Regist SO1 handler
     reti                               ;
     nop                                ;
     nop                                ;

; SI2 vector
     nop                                ;
     reti                               ;
     nop                                ;
     nop                                ;

; SO2 vector
     nop                                ;
     reti                               ;
     nop                                ;
     nop                                ;

; HI vector
     nop                                ;
     reti                               ;
     nop                                ;
     nop                                ;

; HO vector
     nop                                ;
```

(5/13)

```
     reti                                ;
     nop                                 ;
     nop                                 ;

; Hardware signal vector
     nop                                 ;
     reti                                ;
     nop                                 ;
     nop                                 ;

; Timer vector
     nop                                 ;
     reti                                ;
     nop                                 ;
     nop                                 ;


/* ============================================================
 *   PROGRAM CODE
 * ============================================================ */
MAIN IMSEG AT 0x240                      ;
start:
     clr(r0)                             ;
     r0l = EIR                           ;
     r0  = r0 | 0x8000                   ;
     EIR = r0l                           ; disable interrupt

     ;;=========================================================;;
     ;;      Clear Register
     ;;=========================================================;;
     clr(r0)                             ;
     clr(r1)                             ;
     clr(r2)                             ;
     clr(r3)                             ;
     clr(r4)                             ;
     clr(r5)                             ;
     clr(r6)                             ;
     clr(r7)                             ;
     dp0 = r0l                           ;
     dp1 = r0l                           ;
     dp2 = r0l                           ;
     dp3 = r0l                           ;
     dp4 = r0l                           ;
     dp5 = r0l                           ;
     dp6 = r0l                           ;
     dp7 = r0l                           ;
     dn0 = r0l                           ;
     dn1 = r0l                           ;
     dn2 = r0l                           ;
     dn3 = r0l                           ;
```

(6/13)

```
dn4 = r0l                               ;
dn5 = r0l                               ;
dn6 = r0l                               ;
dn7 = r0l                               ;
DMX = r0l                               ;
DMY = r0l                               ;
;;=========================================================;;
;;      Initialize Register & Peripheral Units
;;=========================================================;;
r0l = 0x0000                            ;
*DWTR:x=r0l                             ;

r0l = 0x0081                            ;
*HST:x = r0l                            ;

r0l = 0x0202                            ;
*SST1:x = r0l                           ;

;;=========================================================;;
;;      Initialize Variables and Buffer
;;=========================================================;;
dp0 = ser_out_buffer                    ;
clr(r0)                                 ;
rep MAX_RINGSIZE                        ;
        *dp0++ = r0h                    ;
r0l = ser_out_buffer                    ;
*ring_write_ptr:x = r0l                 ;
*ring_read_ptr:x  = r0l                 ;
*ring_entries:x = r0h                   ; set 0
r0l = 1                                 ;
*ring_dn0:x = r0l                       ;

*g_ulOutputSample:x   = r0h             ;
*g_ulOutputSample+1:x = r0h             ;

r0l = Title                             ;
*desc_pTitle:x = r0l                    ;
r0l = Author                            ;
*desc_pAuthor:x = r0l                   ;
r0l = Copyright                         ;
*desc_pCopyright:x = r0l                ;
r0l = Description                       ;
*desc_pDescription:x = r0l              ;
r0l = Rating                            ;
*desc_pRating:x = r0l                   ;
r0l = STRING_SIZE                       ;
*desc_title_len:x = r0l                 ;
*desc_author_len:x = r0l                ;
*desc_copyright_len:x = r0l             ;
```

(7/13)

```
    *desc_description_len:x = r0l          ;
    *desc_rating_len:x = r0l                ;

    call Init_FileCBGetData                 ;


    ;;=========================================================;;
    ;;      Get Version Information
    ;;=========================================================;;
    call wmad_GetVersion                    ;

    ;;=========================================================;;
    ;;      Initialize WMA Decoder and Get Content Information
    ;;=========================================================;;
    r0l = wmad_FileCBGetData                ;
    clr(r1)                                 ;
    r1l = 0x0001                            ;
    r2l = desc                              ;
    call wmad_FileDecodeInit                ;
    r0 = r0 ^ cWMA_NoErr                    ;
    if(r0!=0) jmp _init_error               ;

    ;;=========================================================;;
    ;;      Get File Information
    ;;=========================================================;;
    r0l = StructFileInfo                    ;
    call wmad_FileDecodeInfo                ;


    ;;=========================================================;;
    ;;      Set Size of Unit for Decode Process
    ;;=========================================================;;
    clr(r0)                                 ;
    r0l = *sample_rate:x                    ;
    r1 = r0 - 32000                         ;
    if(r1>=0) jmp _pre_set_size_2048        ;
    r1 = r0 - 22050                         ;
    if(r1>=0) jmp _pre_set_size_1024        ;
            r1l = 512                       ;
            jmp _pre_set_size_end           ;
_pre_set_size_1024:
            r1l = 1024                      ;
            jmp _pre_set_size_end           ;
_pre_set_size_2048:
            r1l = 2048                      ;
_pre_set_size_end:
    *n_sample:x = r1l                       ; sample per ch

    clr(r0)                                 ;
    *start_flag:x = r0l                     ; set *start_flag:x = 0
```

(8/13)

```
     *f_dec_unit_end:x = r0l                ;
     *n_get_pcm:x = r0l                     ;


     ;;=========================================================;;
     ;;      Previous Decode routine
     ;;=========================================================;;
_pre_DecodeData:
     call    wmad_FileDecodeData            ;

     /*** decode is finished ? ***/
     r1 = r0 ^  cWMA_NoMoreFrames           ;
     if(r1==0) jmp finish                   ;
     r1 = r0 ^  cWMA_Failed                 ;
     if(r1==0) jmp finish                   ;

     /*** check error ***/
     r1 = r0 ^ cWMA_NoErr                   ;
     if(r1!=0) jmp _decode_error            ;

_pre_GetPCM:
     r2 = *n_get_pcm:x                      ;
     r3 = *n_sample:x                       ;
     r1 = r3 - r2                           ;
     r1 = r1 sra 16                         ; set r1
     r2  = r2 sra 16                        ;
     clr(r4)                                ;
     r4l = *Num_Channels:x                  ; Set r4 = Number of ch
     r4  = r4 - 2                           ;
     if(r4==0) r2 += r2                     ;
     r0l = output_buffer                    ;
     r0  = r0 + r2                          ; set r0

     call    wmad_FileGetPCM                ;
     if(r1==0) jmp _pre_DecodeData          ;

     clr(r2)                                ;
     r2h= *g_ulOutputSample:x               ;
     r2l= *g_ulOutputSample+1:x             ;
     r2 = r2 + r1                           ; count total sample per ch
     *g_ulOutputSample:x = r2h              ;
     *g_ulOutputSample+1:x = r2l            ;

     clr(r2)                                ;
     r2l = *n_get_pcm:x                     ;
     r2  = r2 + r1                          ;
     clr(r3)                                ;
     r3l = *n_sample:x                      ;
     r3  = r3 - r2                          ;
     if(r3>0) jmp _pre_GetPCM_end           ;
```

(9/13)

```
             r2l = 1                          ;
             *f_dec_unit_end:x = r2l           ; f_dec_unit_end = 1
             *start_flag:x = r2l               ; set *start_flag:x = 1
             clr(r2)                          ;
_pre_GetPCM_end:
     *n_get_pcm:x = r2l                       ;


     /* chekc decode unit is end ? */
     r2 = *f_dec_unit_end:x                   ;
     if(r2==0) jmp _pre_GetPCM                ;
#if USE_SO
     call _SetPCM                             ; set PCM
#else
     *f_dec_unit_end:x =  r2l                 ; f_dec_unit_end = 0
#endif

     ;;=========================================================;;
     ;;      Set interrupt mask and Start Serial Output
     ;;=========================================================;;
serout_start:
     *SDT1:x = r1h                            ; serial out start
     r0l = SR                                 ;
     r0  = r0 & 0x7fdf                        ;
     r0  = r0 | 0x0fdf                        ;
     nop                                      ;
     sr  = r0l                                ; enable interrupt (SO1)
     jmp _loop_GetPCM                         ;


     ;;=========================================================;;
     ;;      Main routine
     ;;=========================================================;;
_loop_DecodeData:
     call    wmad_FileDecodeData              ;

     /*** decode is finished ? ***/
     r1 = r0 ^  cWMA_NoMoreFrames             ;
     if(r1==0) jmp finish                     ;
     r1 = r0 ^  cWMA_Failed                   ;
     if(r1==0) jmp finish                     ;

     /*** check error ***/
     r1 = r0 ^ cWMA_NoErr                     ;
     if(r1!=0) jmp _decode_error              ;
     r1 = *ring_entries:x                     ;
     if(r1<0) jmp _mips_overflow              ;

_loop_GetPCM:
     r2 = *n_get_pcm:x                        ;
```

(10/13)

```
     r3 = *n_sample:x                        ;
     r1 = r3 - r2                            ;
     r1 = r1 sra 16                          ; set r1
     r2  = r2 sra 16                         ;
     clr(r4)                                 ;
     r4l = *Num_Channels:x                   ; Set r4 = Number of ch
     r4  = r4 - 2                            ;
     if(r4==0) r2 += r2                      ;
     r0l = output_buffer                     ;
     r0  = r0 + r2                           ; set r0

     call    wmad_FileGetPCM                 ;
     r2 = *ring_entries:x                    ;
     if(r2<0) jmp _mips_overflow             ;
     if(r1==0) jmp _loop_DecodeData          ;

     clr(r2)                                 ;
     r2h= *g_ulOutputSample:x                ;
     r2l= *g_ulOutputSample+1:x              ;
     r2 = r2 + r1                            ;  count total sample per ch
     *g_ulOutputSample:x = r2h               ;
     *g_ulOutputSample+1:x = r2l             ;

     clr(r2)                                 ;
     r2l = *n_get_pcm:x                      ;
     r2  = r2 + r1                           ;
     clr(r3)                                 ;
     r3l = *n_sample:x                       ;
     r3  = r3 - r2                           ;
     if(r3>0) jmp _loop_GetPCM_end           ;
             r2l = 1                         ;
             *f_dec_unit_end:x = r2l         ; f_dec_unit_end = 1
             clr(r2)                         ;
_loop_GetPCM_end:
     *n_get_pcm:x = r2l                      ;

     /* check decode unit is end ? */
     r2 = *f_dec_unit_end:x                  ;
     if(r2==0) jmp _loop_GetPCM              ;
     nop                                     ;

_loop_wait:
     r2 = *ring_entries:x                    ;
     if(r2<0) jmp _mips_overflow             ;
     if(r2!=0) jmp _loop_wait                ;
#if USE_SO
     call _SetPCM                            ; set PCM
#else
     *f_dec_unit_end:x =  r2l                ; f_dec_unit_end = 0
#endif
```

(11/13)

```
     jmp _loop_GetPCM                        ;


     ;;=========================================================;;
     ;;      Finish
     ;;=========================================================;;
finish:
     r2 = *ring_entries:x                    ;
     if(r2>0) jmp $-1                        ;
     r2 = *n_get_pcm:x                       ;
     if(r2==0) jmp serout_finish             ;
     *n_sample:x = r2h                       ;
#if USE_SO
     call _SetPCM                            ; set PCM
#else
     *f_dec_unit_end:x =  r2l                ; f_dec_unit_end = 0
#endif
     r2 = *ring_entries:x                    ;
     if(r2>0) jmp $-1                        ;
serout_finish:
     nop                                     ;
     jmp serout_finish                       ;


     ;;=========================================================;;
     ;;      Error
     ;;=========================================================;;
_init_error:
     nop                                     ;
     jmp $-1                                 ;

_decode_error:
     nop                                     ;
     jmp $-1                                 ;

_mips_overflow:
     nop                                     ;
     jmp $-1                                 ;




/* ================================================================
[Function Name] _SetPCM
 ================================================================*/
_SetPCM:
     dp0 = output_buffer                     ;
     r2l = *ring_write_ptr:x                 ;
     dp1 = r2l                               ;
     dn1 = 1                                 ;
```

(12/13)

```
     dmx = MAX_RINGSIZE-1                    ;

     clr(r4)                                 ;
     r4l = *Num_Channels:x                   ; Set r4 = Number of ch
     r4  = r4 - 1                            ;
     clr(r3)                                 ;
     r3l = *n_sample:x                       ;
     Loop r3l {                              ;
             r2 = *dp0++                     ;
             *dp1%% = r2h                    ;
             if(r4==0) jmp $+2               ; if ch = 1
             r2 = *dp0++                     ;
             *dp1%% = r2h                    ;
             nop                             ;
     }                                       ;
     r2l = dp1                               ;
     *ring_write_ptr:x = r2l                 ;

     r3 += r3                                ; always, output 2 ch
     clr(r2)                                 ;
     *f_dec_unit_end:x =  r2h                ; f_dec_unit_end = 0

     r2l  = EIR                              ;
     r2   = r2 | 0x8000                      ;
     EIR  = r2l                              ;
     nop                                     ;
     nop                                     ; wait disable interrupt
     nop                                     ;
     r2l = *ring_entries:x                   ;
     r2  = r2 + r3                           ;
     *ring_entries:x = r2l                   ;
     r2l = EIR                               ;
     r2  = r2 & 0x7fff                       ;
     EIR = r2l                               ; enable interrupt
     ret                                     ;


/* ================================================================
[Handler Name]  _so_interrupt:
 ================================================================*/
_so_interrupt:
     *save_regs+0:x = r0l                    ; push r0
     *save_regs+1:x = r0h                    ;
     *save_regs+2:x = r0e                    ;
     r0l = dp0                               ;
     *save_regs+3:x = r0l                    ; push dp0
     r0l = dn0                               ;
     *save_regs+4:x = r0l                    ; push dn0
     r0l = dmx                               ;
     *save_regs+5:x = r0l                    ; push dmx
```

(13/13)

```
     r0l = *ring_read_ptr:                ;
     dp0 = r0l                            ; set dp0
     r0l = *ring_dn0:x                    ;
     dn0 = r0l                            ; set dn0
     dmx = MAX_RINGSIZE-1                 ; set dmx
     r0  = *dp0%%                         ;
     *SDT1:x=r0h                          ; set output PCM data
     r0l = dp0                            ;
     *ring_read_ptr:x = r0l               ; set read ptr
     r0l = *ring_entries:x                ;
     r0  = r0 - 1                         ;
     *ring_entries:x = r0l                ; set ring_entries

     r0l = *save_regs+5:x                 ;
     dmx = r0l                            ; pop dmx
     r0l = *save_regs+4:x                 ;
     dn0 = r0l                            ; pop dn0
     r0l = *save_regs+3:x                 ;
     dp0 = r0l                            ; pop dp0
     r0e = *save_regs+2:x                 ;
     r0h = *save_regs+1:x                 ;
     r0l = *save_regs+0:x                 ; pop r0
     reti                                 ;

end
```

A. 1. 2　smp_data.asm

スタティック領域とスクラッチ領域の確保を行うファイルです。

```
/* ============================================================
 *      PUBLIC
 * ============================================================ */
public wmad_lib_static_x1
public wmad_lib_static_x2
public wmad_lib_static_y1
public wmad_lib_static_y2

public wmad_lib_scratch_x
public wmad_lib_scratch_y


/* ============================================================
 *      STATIC MEMORY
 * ============================================================ */

__WMAD_LIB_STATIC_X1        XRAMSEG
wmad_lib_static_x1:                 ds      972

__WMAD_LIB_STATIC_X2        XRAMSEG
wmad_lib_static_x2:                 ds      4096


__WMAD_LIB_STATIC_Y1        YRAMSEG
wmad_lib_static_y1:                 ds      8192

__WMAD_LIB_STATIC_Y2        YRAMSEG
wmad_lib_static_y2:                 ds      4096


/* ============================================================
 *      SCRATCH MEMORY
 * ============================================================ */

__WMAD_LIB_SCRATCH_X        XRAMSEG
wmad_lib_scratch_x:                 ds      1600

__WMAD_LIB_SCRATCH_Y        YRAMSEG
wmad_lib_scratch_y:                 ds      300

end
```

A. 1. 3　smp_user.asm

ユーザ定義関数 wmad_FileCBGetData **のソース・ファイルです。**

(1/6)

```
/*-----------------------------------------------------------------*/
/*   File Information                                              */
/*-----------------------------------------------------------------*/
/*   Name      : smp_user.asm                                      */
/*   Type      : Assembler program module                          */
/*   Version   : 1.00                                              */
/*   Date      : 2001 Jun 18                                       */
/*   CPU       : uPD7701x Family                                   */
/*   Assembler : WB77016 Ver 2.4                                   */
/*   About     : wmad_FileCBGetData function                       */
/*                                                                 */
/*-----------------------------------------------------------------*/
/*   Copyright (C) NEC Corporation 2000, 2001                      */
/*   All rights reserved by NEC Corporation                        */
/*   Use of copyright notice does not evidence publication         */
/*-----------------------------------------------------------------*/

/* ================================================================
 *   INCLUDE FILES
 * ================================================================ */
#include "wma_dec.h"

/* ================================================================
 *   PUBLIC FUNCTIONS
 * ================================================================ */
public Init_FileCBGetData
public wmad_FileCBGetData

/* ================================================================
 *   EXTERN FUNCTIONS
 * ================================================================ */


/* ================================================================
 *   DEFINE
 * ================================================================ */
#define CW_BUFF_SIZE     64
#define CB_BUFF_SIZE    128


/* ================================================================
 *      LOCAL MEMORY
 * ================================================================ */
```

(2/6)

```
__SMP_USER_XRAM__           XRAMSEG
FileCBGetData_fp:
     ds      2              ; size of obtained bitstream data [byte]
read_ptr:
     ds      1              ; read pointer to host_in_buffer
write_ptr:
     ds      1              ; write pointer to host_in_buffer
tmp_dn0:
     ds      1              ; for saving value of dn0 register
tmp_dmx:
     ds      1              ; for saving value of dmx register

__SMP_USER_BUFFER__ XRAMSEG align at 0
host_in_buffer:
     ds      CW_BUFF_SIZE*2  ; buffer for input data from Host I/F
input_buffer:
     ds      CW_BUFF_SIZE    ; buffer for input data to Middle Ware


/* ================================================================
 *      PROGRAM CODE
 * ================================================================ */
__SMP_USER__   IMSEG

/* ================================================================
[Function Name]      Init_FileCBGetData
[Argument]           r0  : data size
[Return]             non
[Call Function]      non
[Use Register]       r0, dp0, dp1
[Use Stacks]         loop stack: 1, call stack: 0, repeat: 0
 ================================================================*/
Init_FileCBGetData:
     clr(r0)                        ;
     *FileCBGetData_fp+0:x = r0l    ;
     *FileCBGetData_fp+1:x = r0l    ;
     dn0 = r0l                      ;
     r0l = host_in_buffer           ;
     *read_ptr:x = r0l              ;
     *write_ptr:x = r0l             ;
     dp0 = r0l                      ;
     r0l = input_buffer             ;
     dp1 = r0l                      ;
     Loop CW_BUFF_SIZE {            ; clear buffer
             *dp0++ = r0h           ;
             *dp0++ = r0h           ;
             *dp1++ = r0h           ;
     }                              ;
     ret                            ;
```

(3/6)

```
/* ============================================================
[Function Name]      wmad_FileCBGetData
[Argument]           r0  : required size [byte]
                     r1  : offset [byte]
[Return]             r0  : obtained size [byte]
                     r2l : address of input buffer
[Call Function]      non
[Use Register]       r0, r1, r2, r3, r4, r5, r6
                     dp0, dp1, dn0, dmx
[Use Stacks]         loop stack: 1, call stack: 0, repeat: 0
 ============================================================*/
wmad_FileCBGetData:
     r4l = dn0                              ;
     *tmp_dn0:x = r4l                       ;
     r4l = dmx                              ;
     *tmp_dmx:x = r4l                       ;

     r4l = 1                                ;
     dn0 = r4l                              ;
     r4l = 2*CW_BUFF_SIZE - 1               ;
     dmx = r4l                              ;

     r5 = *FileCBGetData_fp+0:x             ;
     r5l= *FileCBGetData_fp+1:x             ; r5 is always even.
     r4 = r5-r1                             ;

     if(r4==0) jmp case000                  ;
     if(r4>0)  jmp case002                  ;

     ;;=========================================================;;
     ;;      case001 ( r5 < r1 )
     ;;=========================================================;;
case001:
     r4 = -r4                               ;
     r4 = r4 sra 1                          ; r4 = read size [word]
     if(r4<=0) jmp case001a                 ;
     Loop r4l {                             ;
             %READ_HOST(R6,R6)              ;
             r5 = r5 + 2                    ;
             nop                            ; not stored
        }                                   ;
case001a:
     r6l = *write_ptr:x                     ;
     *read_ptr:x = r6l                      ; set read_ptr

     r3 = r0 + 1                            ;
     r3 = r3 srl 1                          ; set r3 = read size [word]
     jmp get_data                           ;
```

(4/6)

```
     ;;=========================================================;;
     ;;      case000 ( r5 = r1 )
     ;;=========================================================;;
case000:
     r6l = *write_ptr:x                    ;
     *read_ptr:x = r6l                     ; set read_ptr

     r3 = r0 + 1                           ;
     r3 = r3 srl 1                         ; set r3 = read size [word]
     jmp get_data                          ;


     ;;=========================================================;;
     ;;      case002 ( r5 > r1 )
     ;;=========================================================;;
case002:
     r3 = r4 + 1                           ;
     r3 = r3 sra 1                         ; set r3 = rewind size [word]

     clr(r6)                               ;
     r6l= *write_ptr:x                     ;
     r6 = r6 - r3                          ;

     r3 = r6 - host_in_buffer              ;
     if(r3>=0) jmp case002a                ;
     r6 = r3 + host_in_buffer + CB_BUFF_SIZE;

case002a:
     *read_ptr:x = r6l                     ; set read_ptr
     r3  = r0 - r4                         ;
     if(r3<=0) jmp set_data                ; No need to read data
     r3  = r3+1                            ;
     r3  = r3 srl 1                        ; set r3 = read size [word]


     ;;=========================================================;;
     ;;      Get Data
     ;;=========================================================;;
get_data:
     r4 = r0 - CB_BUFF_SIZE                ;
     if(r4<=0) jmp get_data_nex            ;
     r3 = r0 + 1                           ;
     r3 = r3 sra 1                         ;
     Loop r3l {                            ;
             %READ_HOST(R4,R4)             ;
             r5 = r5 + 2                   ;
             nop                           ; not stored
     }                                     ;
     jmp finish                            ;
```

(5/6)

```
get_data_next:
     if(r3==0) jmp set_data          ;
     r6l = *write_ptr:x              ;
     dp0 = r6l                       ;
     loop r3l {
             %READ_HOST(R4,R4)       ; get data
             *dp0%% = r4h            ;
             r5 = r5 + 2             ;
     }
     r6l = dp0                       ;
     *write_ptr:x = r6l              ;


     ;;=========================================================;;
     ;;      Set Data
     ;;=========================================================;;
set_data:
     r6l = *read_ptr:x               ;
     dp0 = r6l                       ;
     dp1 = input_buffer              ;

     r4 = r0+1                       ;
     r4 = r4 srl 1                   ;
     if(r4==0) jmp finish            ;

     r3 = r1 & 1                     ;
     if(r3==0) jmp simple_copy       ;
     r3l= *dp0%%                     ;
     r3 = r3 sll 8                   ;
     loop r4l {
             r3 = r3 sll 8           ; change allocation
             r3l= *dp0%%             ;
             r3 = r3 sll 8           ;
             *dp1++ = r3h            ; set data
     }                               ;
     jmp finish                      ;
simple_copy:
     loop r4l {                      ;
             r3l= *dp0%%             ;
             *dp1++ = r3l            ; set data
     }                               ;


     ;;=========================================================;;
     ;;      Finish
     ;;=========================================================;;
finish:
     *FileCBGetData_fp+0:x = r5h     ;
     *FileCBGetData_fp+1:x = r5l     ;
```

(6/6)

```
      r4l = *tmp_dn0:x                ;
      dn0 = r4l                       ;
      r4l = *tmp_dmx:x                ;
      dmx = r4l                       ;

      r2l= input_buffer               ; set r2l
      ret                             ;

END
```

A. 2　サンプル・ヘッダ・ファイル

A. 2. 1　wma_dec.h

サンプル・プログラム用のヘッダ・ファイルです。

(1/2)

```
/*-------------------------------------------------------------------*/
/*   File Information                                                */
/*-------------------------------------------------------------------*/
/*   Name       : wma_dec.h                                          */
/*   Type       : Assembler header file                              */
/*   Version    : 1.00                                               */
/*   Date       : 2001 Jun 18                                        */
/*   CPU        : uPD7701x Family                                    */
/*   Assembler  : WB77016 Ver 2.4                                    */
/*   About      : header file for sample source code                 */
/*                                                                   */
/*-------------------------------------------------------------------*/
/*   Copyright (C) NEC Corporation 2000, 2001                        */
/*   All rights reserved by NEC Corporation.                         */
/*   Use of copyright notice does not evidence publication           */
/*-------------------------------------------------------------------*/

/* ================================================================
 *   DEFINE
 * ================================================================ */
#define SDT1 0x3800
#define SST1 0x3801
#define SDT2 0x3802
#define SST2 0x3803
#define HDT  0x3806
#define HST  0x3807
#define DWTR 0x3808

/* ================================================================
 *   EXTERN FUNCTIONS
 * ================================================================ */
extrn wmad_FileDecodeInit
extrn wmad_FileDecodeData
extrn wmad_FileGetPCM
extrn wmad_FileDecodeInfo
extrn wmad_GetVersion

/* ================================================================
 *   MACRO
 * ================================================================ */
%define (READ_HOST(X,Y)) (
     Y@L = *HST:x                          ;
     Y   = Y & 1                           ;
```

(2/2)

```
     if(Y !=0) jmp $-2                      ;
     X   = *HDT:x                           ;
);

/* ============================================================
 *   DEFINE (ERROR CODE)
 * ============================================================ */
#define cWMA_NoErr                          0
#define cWMA_Failed                         1
#define cWMA_BadArgument                    2
#define cWMA_BadAsfHeader                   3
#define cWMA_BadPacketHeader                4
#define cWMA_BrokenFrame                    5
#define cWMA_NoMoreFrames                   6
#define cWMA_BadSamplingRate                7
#define cWMA_BadNumberOfChannels            8
#define cWMA_BadVersionNumber               9
#define cWMA_BadWeightingMode               10
#define cWMA_BadPacketization               11
#define cWMA_BadDRMType                     12
#define cWMA_DRMFailed                      13
#define cWMA_DRMUnsupported                 14
#define cWMA_DemoExpired                    15
#define cWMA_BadState                       16
#define cWMA_Internal                       17
```

A. 3　サンプル・タイミング・ファイル

A. 3. 1　smp_input.tmg

入力データ・ファイル（sample_20_16s.dat）からデータを取り出し，ホスト・インタフェース経由でデータを入力するタイミング・ファイルです。10 行目の入力データ・ファイル名は，ユーザの用意した入力データ・ファイルにあわせて変更してください。。

(1/2)

```
;;-------------------------------------------------------------------;;
;;   Declare Variables
;;-------------------------------------------------------------------;;
local data                          ; local variable receives data


;;-------------------------------------------------------------------;;
;;   File Open
;;-------------------------------------------------------------------;;
open input "sample_20_16s.dat"
output format showbase unsigned hex, ; select output format


;;-------------------------------------------------------------------;;
;;   Init
;;-------------------------------------------------------------------;;
     set pin hcs = 1                ; terminate any write access, which might
     set pin hwr = 1                ; be active
     set pin hrd = 1                ;

;;-------------------------------------------------------------------;;
;;   Main Input Loop
;;-------------------------------------------------------------------;;
  do                                ;
     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
     set port ha = 1                ; select higher byte of HDT
     wait 5ns                       ; delay
     set pin hwr = 0                ; start output
```

(2/2)

```
     set pin hcs = 0                ; perform 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
  enddo                             ;


;;-------------------------------------------------------------------;;
;;   File Close
;;-------------------------------------------------------------------;;
close input                         ;

break                               ;
end
```

A. 3. 2　smp_serout.tmg

シリアル・インタフェースから出力される 16 ビット・リニア PCM データをファイル（so_sample_20_16s_l.dat, so_sample_20_16s_r.dat）に保存するためのタイミング・ファイルです。9, 10 行目のファイル名は自由に設定してください。ステレオのデータの場合，出力ファイル#1, #2 にはそれぞれ L チャネル，R チャネルのデータが出力されます。モノラルのデータの場合，出力ファイル#1, #2 には同じデータが出力されます。

(1/3)

```
;;-------------------------------------------------------------------------;;
;;   Initialize
;;-------------------------------------------------------------------------;;
set pin soen1 = 0                   ; initialize SOEN1 line

;;-------------------------------------------------------------------------;;
;;   File Open
;;-------------------------------------------------------------------------;;
open output #1 "so_sample_20_16s_l.dat"     ;
open output #2 "so_sample_20_16s_r.dat"     ;
open output #3 "dummy_serial.dat"           ;
output format showbase unsigned hex,        ; select output format

;;-------------------------------------------------------------------------;;
;;   Dummy Output
;;-------------------------------------------------------------------------;;
  if pin soen1 == 0                 ; do only if new transmission start
     wait cond pin sorq1 == 1       ; wait for serial output request
     wait 5 ns                      ; logic delay
     set pin soen1 = 1              ; start serial output
  endif

  wait cond pin sorq1 == 0          ; wait for serial output start confirmation
  set pin soen1 = 0                 ;

  rept 15                           ; wait 15 clock cycles for one data frame
     wait cond pin sck1 == 0        ;   wait for rising edge (0->1)
     wait cond pin sck1 == 1        ;
  endrept                           ;

  wait 5ns                          ; make sure SO1 and SORQ1 are updated

     output #3 port so1&0xFFFF      ; write output data to file, mask sign bits

     if pin sorq1 == 1              ; request next output
             set pin soen1 = 1      ; start next serial output
     endif
```

(2/3)

```
  wait cond pin sck1 == 0           ; wait for rising edge (0->1)
  wait cond pin sck1 == 1           ;
close output #3


;;-------------------------------------------------------------------------;;
;;   Main Output Loop
;;-------------------------------------------------------------------------;;
do                                  ;

;;-------------------------------------------------------------------------;;
;;   For ch 1
;;-------------------------------------------------------------------------;;
  if pin soen1 == 0                 ; do only if new transmission start
     wait cond pin sorq1 == 1       ; wait for serial output request
     wait 5 ns                      ; logic delay
     set pin soen1 = 1              ; start serial output
  endif

  wait cond pin sorq1 == 0          ; wait for serial output start confirmation
  set pin soen1 = 0                 ;

  rept 15                           ; wait 15 clock cycles for one data frame
     wait cond pin sck1 == 0        ;       wait for rising edge (0->1)
     wait cond pin sck1 == 1        ;
  endrept                           ;

  wait 5ns                          ; make sure SO1 and SORQ1 are updated

  output #1 port so1&0xFFFF         ; write output data to file, mask sign bits

  if pin sorq1 == 1                 ; request next output
     set pin soen1 = 1              ; start next serial output
  endif

  wait cond pin sck1 == 0           ; wait for rising edge (0->1)
  wait cond pin sck1 == 1           ;

;;-------------------------------------------------------------------------;;
;;   For ch 2
;;-------------------------------------------------------------------------;;
  if pin soen1 == 0                 ; do only if new transmission start
     wait cond pin sorq1 == 1       ; wait for serial output request
     wait 5 ns                      ; logic delay
     set pin soen1 = 1              ; start serial output
  endif

  wait cond pin sorq1 == 0          ; wait for serial output start confirmation
  set pin soen1 = 0                 ;
```

(3/3)

```
  rept 15                          ; wait 15 clock cycles for one data frame
     wait cond pin sck1 == 0       ;       wait for rising edge (0->1)
     wait cond pin sck1 == 1       ;
  endrept                          ;

  wait 5ns                         ; make sure SO1 and SORQ1 are updated

  output #2 port so1&0xFFFF        ; write output data to file, mask sign bits

  if pin sorq1 == 1                ; request next output
     set pin soen1 = 1             ; start next serial output
  endif

  wait cond pin sck1 == 0          ; wait for rising edge (0->1)
  wait cond pin sck1 == 1          ;


;;-------------------------------------------------------------------------;;
;;   Serial Output is finished ?
;;-------------------------------------------------------------------------;;
  exit  ip == (MAIN.serout_finish & 0xffff)        ;
  exit  ip == ((MAIN.serout_finish+1) & 0xffff)    ;
enddo


;;-------------------------------------------------------------------------;;
;;   File Close
;;-------------------------------------------------------------------------;;
close output #1                    ; close data file
close output #2                    ; close data file


break                              ;
end
```

A. 3. 3　clk_for_2ch.tmg

シリアル出力の割り込みを発生させるためのクロック信号を生成するタイミング・ファイルです。入力データのサンプリング周波数に合わせ，適切な周期のクロック信号を生成します。常に 2 チャネル分のデータが出力されることを想定して記述しています。

```
;;-------------------------------------------------------------------------;;
;;   Declare Variables
;;-------------------------------------------------------------------------;;
LOCAL TM_pico_sec    ;
LOCAL fs             ;
LOCAL retern_addr    ;

;;-------------------------------------------------------------------------;;
;;   Wait
;;-------------------------------------------------------------------------;;
wait cond reg ip == ((MAIN.serout_start) & 0xffff)  ;

;;-------------------------------------------------------------------------;;
;;   Set Sampling Frequency and TM_pico_sec
;;-------------------------------------------------------------------------;;
set fs = *sample_rate:x & 0xffff                    ;
set TM_pico_sec = TIME_RESOLUTION / 16 / fs / 2     ; TIME_RESOLUTION = 10**12

;;-------------------------------------------------------------------------;;
;;   Generate Clock for 2 ch
;;-------------------------------------------------------------------------;;
do
  wait (TM_pico_sec/2) ps                   ;
  set pin sck1 = 0                          ;
  set pin sck2 = 0                          ;
  wait (TM_pico_sec/2) ps                   ;
  set pin sck1 = 1                          ;
  set pin sck2 = 1                          ;
enddo
```

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】

第一販売事業部
東　京　(03)3798−6106, 6107, 6108
大　阪　(06)6945−3178, 3200, 3208, 3212
広　島　(082)242−5504
仙　台　(022)267−8740

第二販売事業部
東　京　(03)3798−6110, 6111, 6112
立　川　(042)526−5981, 6167
松　本　(0263)35−1662
静　岡　(054)254−4794
金　沢　(076)232−7303
松　山　(089)945−4149

第三販売事業部
東　京　(03)3798−6151, 6155, 6586, 1622, 1623, 6156
水　戸　(029)226−1702
前　橋　(027)243−6060
鳥　取　(0857)27−5313
名古屋　(052)222−2170, 2190
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-B11　ユーザーズ・マニュアル

（U15683JJ1V1UM00（第 1 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（　　）					
（　　）					

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

キリトリ