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-B06

## AMR 音声コーデック・ミドルウエア

---

## 対象デバイス

μPD77018A
μPD77019
μPD77110
μPD77111
μPD77112
μPD77113A
μPD77114

**資料番号** U15165JJ3V0UM00（第 3 版）
**発行年月** May 2002 NS CP（N）

〔メ　モ〕

# 目次要約

**Windows，Windows NT 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 |
|---|---|
| 全　般 | 対象デバイスからμPD77018, 77113 を削除。 |
| p.14 | 1. 3. 1　特徴を変更。 |
| p.14 | 1. 3. 2（2）必要メモリを変更。 |
| p.15 | 表 1-3　AMR 音声 CODEC の演算量を変更。 |
| p.16 | 1. 3. 4　ディレクトリ構成を変更。 |
| p.17 | 1. 3. 5　ライブラリの組み合わせを変更。 |
| p.22 | 2. 2. 3　amr_EncodeFrame 関数<br>ハードウエア・リソースメントの値を変更。 |
| p.23 | 2. 2. 4　amr_InitDecoder 関数<br>ハードウエア・リソースメントの値を変更。 |
| p.24 | 2. 2. 5　amr_ResetDecoder 関数<br>ハードウエア・リソースメントの値を変更。 |
| p.25 | 2. 2. 6　amr_DecodeFrame 関数<br>引数，ハードウエア・リソースメントの値を変更。 |
| p.25 | 表 2-3　受信フレーム・タイプを変更。 |
| p.27 | 表 2-4　送信フレーム・タイプを変更。 |
| p.28 | 2. 2. 10　amr_TX_to_RX 関数<br>使用レジスタ，ハードウエア・リソースメントの値を変更。 |
| p.28 | 表 2-5　送受信フレーム・タイプを変更。 |
| p.30 | 2. 2. 13　amr_GetVersion 関数<br>機能を変更。 |
| p.44 | 付録 A　サンプル・プログラム・ソースを変更。 |

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）と，μPD77111 ファミリ（μPD77110, 77111, 77112, 77113A, 77114, 77115）の総称です。
ただし，このマニュアルでは，μPD77018A, 77019, 77110, 77111, 77112, 77113A, 77114 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　ビット・ストリーム・フォーマット**
**第 4 章　インストレーション**
**付録 A　サンプル・プログラム・ソース**
**付録 B　関連文書**

**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。

μPD7701×ファミリのハードウエア機能を知りたいとき
→**μPD7701×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。

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。

**μPD77016 ファミリに関する資料**

| 資料名 / 品名 | パンフレット | データ・シート | ユーザーズ・マニュアル | | アプリケーション・ノート | |
|---|---|---|---|---|---|---|
| | | | アーキテクチャ編 | 命令編 | 基本ソフトウエア編 | ライブラリ編 |
| μPD77016 | − | U10891J | U10503J | U13116J | U11958J | U12021J |
| μPD77015 | | U10902J | | | | |
| μPD77017 | | | | | | |
| μPD77018A | | U11849J | | | | |
| μPD77019 | | | | | | |
| μPD77019-013 | | U13053J | | | | |
| μPD77110 | U12395J | U12801J | U14623J | | | |
| 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。

## 1.2　AMR 音声 CODEC

AMR 音声 CODEC は，3GPP（3rd Generation Partnership Project）で規格された，4.75 kbps～12.2 kbps の音声圧縮／伸長コーデックです。AMR 音声 CODEC は，マルチレート・スピーチ CODER，無音圧縮機能，エラー・コンシルメント機能から構成されます。マルチレート・スピーチ CODER は，MRDTX 注1 を除く，8 つのビット・レート（表 1-1参照）からビット・レートを選択することができ，1 フレーム（20 ms）ごとにビット・レートを切り替えることが可能です。

**表 1-1　AMR 音声 CODEC のビット・レート**

| コーデック・モード | ビット・レート |
|---|---|
| MR122 | 12.20 kbps（GSM EFR 注2) |
| MR102 | 10.20 kbps |
| MR795 | 7.95 kbps |
| MR74 | 7.40 kbps（IS-641 注3) |
| MR67 | 6.70 kbps（PDC-EFR 注4) |
| MR59 | 5.90 kbps |
| MR515 | 5.15 kbps |
| MR475 | 4.75 kbps |
| MRDTX 注1 | 1.80 kbps |

**注 1.** 無音圧縮機能を ON にした場合，エンコーダが自動的に MRDTX モードで圧縮します。

**2.** GSM EFR: ETSI GSM 06.90 Enhanced Full Rate Speech Codec

**3.** IS-641: TIA/EIA IS-641 TDMA Enhanced Full Rate Speech Codec

**4.** PDC-EFR: ARIB 6.7 kbps Enhanced Full Rate Speech Codec

**備考**　各規格の詳細については，**付録 B　関連文書**に記載されている勧告書を参照してください。

# 1.3 システム概要

## 1.3.1 特　徴

★
- 3GPP の AMR 音声 CODEC　Version 7.6.0 に対応（チャネル・コーデックは含まれません）。
- 圧縮/伸長は，8 つのビット・レートに対応（表 1-1　AMR 音声 CODEC のビット・レート参照）。
- 無音圧縮機能対応（VAD1 オプション，VAD2 オプションに対応）。
- サンプリング周波数 8 kHz で 160 サンプル／フレームの符号化，復号化を行う。
- 音声入出力データはすべて 16 ビット・リニア PCM データ注。

注　μSAP77016-B06 のライブラリ内部では，MSB から 13 ビットを使用します。

## 1.3.2 動作環境

★　(1) 動作対象 DSP

- μPD77018A, 77019, 77110, 77111, 77112, 77113A, 77114

★　(2) 必要メモリ

表 1-2　必要メモリの容量

| メモリ空間 | 種　別 | サイズ［ワード］ | | |
|---|---|---|---|---|
| | | Codec | Encoder | Decoder |
| 命令メモリ | − | 14.1 K | 10.2 K | 4.4 K |
| X メモリ | RAM | 2.9 K | 2.5 K | 2.4 K |
| | ROM | 11.2 K | 11.1 K | 10.7 K |
| Y メモリ | RAM | 2.4 K | 1.9 K | 1.5 K |
| | ROM | 3.6 K | 3.2 K | 2.8 K |

備考 1. μSAP77016-B06 のライブラリで使用する，X メモリおよび Y メモリ領域は，内蔵 ROM/RAM 空間に配置してください。

2. 必要メモリの容量には，音声データおよびビット・ストリーム・データ入出力のバッファは含まれていません。詳細は，2.3　メモリ構造を参照してください。

(3) ソフトウエア・ツール（Windows®版）

| | | |
|---|---|---|
| DSP ツール： | ワークベンチ | WB77016 |
| | ハイスピード・シミュレータ | HSM77016 |
| | ディバッガ | ID77016 |

### 1.3.3 性　能

1 フレームの処理をリアルタイム（20 ms）に実行するために必要な MIPS 値は，表 1-3のようになります。

【測定条件】

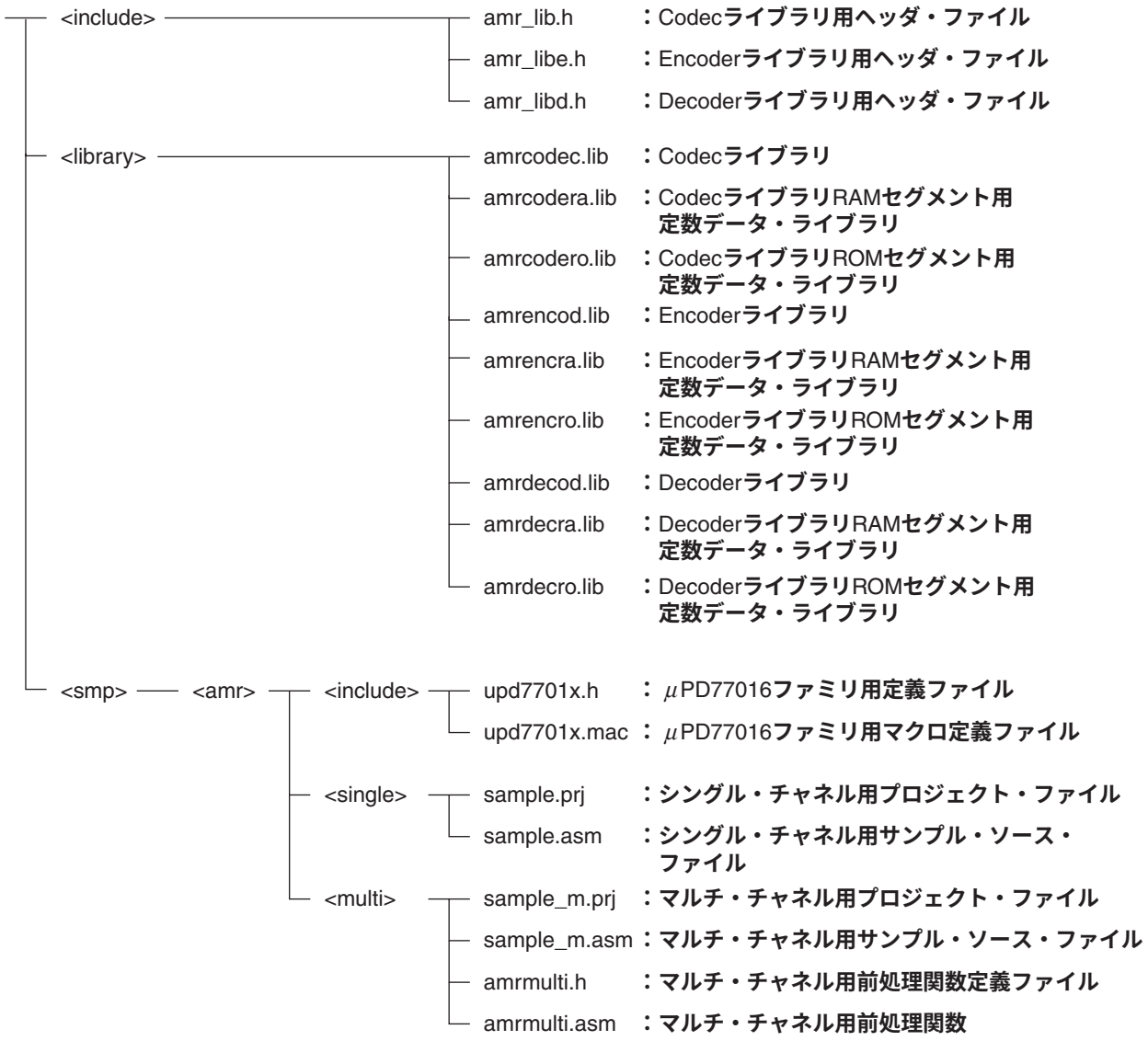
- ハイスピード・シミュレータ：HSM77016
- 評価音声：3GPP　Test Sequences（TS 26.074）の音声ファイルを使用します。
- 評価結果：音声ファイルをそれぞれ圧縮/伸長したときの演算量を測定し，その最悪値を求めます。
- 圧縮は，amr_EncodeFrame 関数のみの演算量になり，伸長は amr_DecodeFrame 関数のみの演算量になります。そのほかの関数，および割り込みハンドラなどの演算量は含みません。

★ 表 1-3　AMR 音声 CODEC の演算量

| コーデック・モード | VAD OFF [MIPS] | | VAD1 ON [MIPS] | | VAD2 ON [MIPS] | |
|---|---|---|---|---|---|---|
| | 圧縮 | 伸長 | 圧縮 | 伸長 | 圧縮 | 伸長 |
| MR475 | 15.116 | 2.658 | 15.658 | 2.658 | 16.749 | 2.683 |
| MR515 | 12.000 | 2.675 | 12.542 | 2.675 | 13.635 | 2.675 |
| MR59 | 14.901 | 2.674 | 15.425 | 2.674 | 16.552 | 2.661 |
| MR67 | 19.331 | 2.740 | 19.817 | 2.740 | 20.970 | 2.745 |
| MR74 | 17.371 | 2.408 | 17.864 | 2.408 | 18.992 | 2.407 |
| MR795 | 18.318 | 2.783 | 18.830 | 2.783 | 19.934 | 2.835 |
| MR102 | 18.548 | 2.495 | 19.054 | 2.495 | 20.161 | 2.495 |
| MR122 | 19.997 | 2.489 | 20.510 | 2.489 | 21.626 | 2.489 |

## ★ 1. 3. 4 ディレクトリ構成

μSAP77016-B06 のディレクトリ構成を次に示します。



## ★ 1.3.5 ライブラリの組み合わせ

使用するライブラリとヘッダ・ファイルの組み合わせは，表 1-4のようになります。

Codec ライブラリを使用する場合の DSP とライブラリの組み合わせは，表 1-5のようになります。

表 1-4　ヘッダ・ファイルとライブラリの組み合わせ

| ライブラリ | ヘッダ・ファイル | コード | RAM 用定数 | ROM 用定数 |
|---|---|---|---|---|
| Codec ライブラリ | amr_lib.h | amrcodec.lib | amrcodra.lib | amrcodro.lib |
| Encoder ライブラリ | amr_libe.h | amrencod.lib | amrencra.lib | amrencro.lib |
| Decoder ライブラリ | amr_libd.h | amrdecod.lib | amrdecra.lib | amrdecro.lib |

表 1-5　ターゲット・デバイスとライブラリの組み合わせ（Codec ライブラリの場合）

| デバイス名 | コード<br>（amrcodec.lib） | RAM 用定数<br>（amrcodra.lib） | ROM 用定数<br>（amrcodro.lib） |
|---|---|---|---|
| μPD77015 | × | × | × |
| μPD77016 | × | × | × |
| μPD77017 | × | × | × |
| μPD77018A | ○ | × | ○ |
| μPD77019 | ○ | × | ○ |
| μPD77110 | ○ | ○ | × |
| μPD77111 | ○ | × | ○ |
| μPD77112 | ○ | × | ○ |
| μPD77113A | ○ | ○注 | ○注 |
| μPD77114 | ○ | ○注 | ○注 |
| μPD77115 | × | × | × |

注　amrcodra.lib，または amrcodro.lib のどちらか一方をリンクしてください。

# 第2章 ライブラリ仕様

この章では，μSAP77016-B06 の関数仕様と呼び出し規約について説明します。

## 2. 1 アプリケーション処理フロー

μSAP77016-B06 を使用したアプリケーションの処理の例を図 2-1，図 2-2に示します。

図 2-1 アプリケーション処理フロー例（エンコーダ）

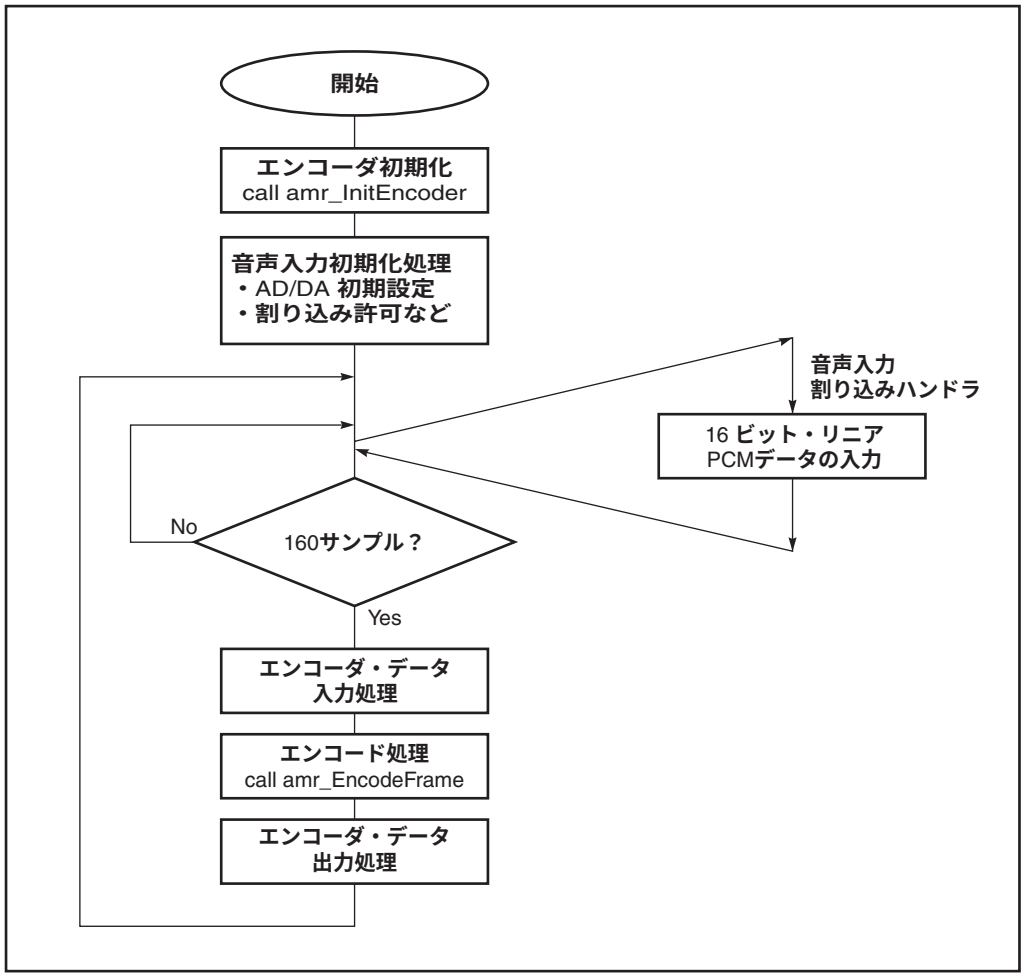
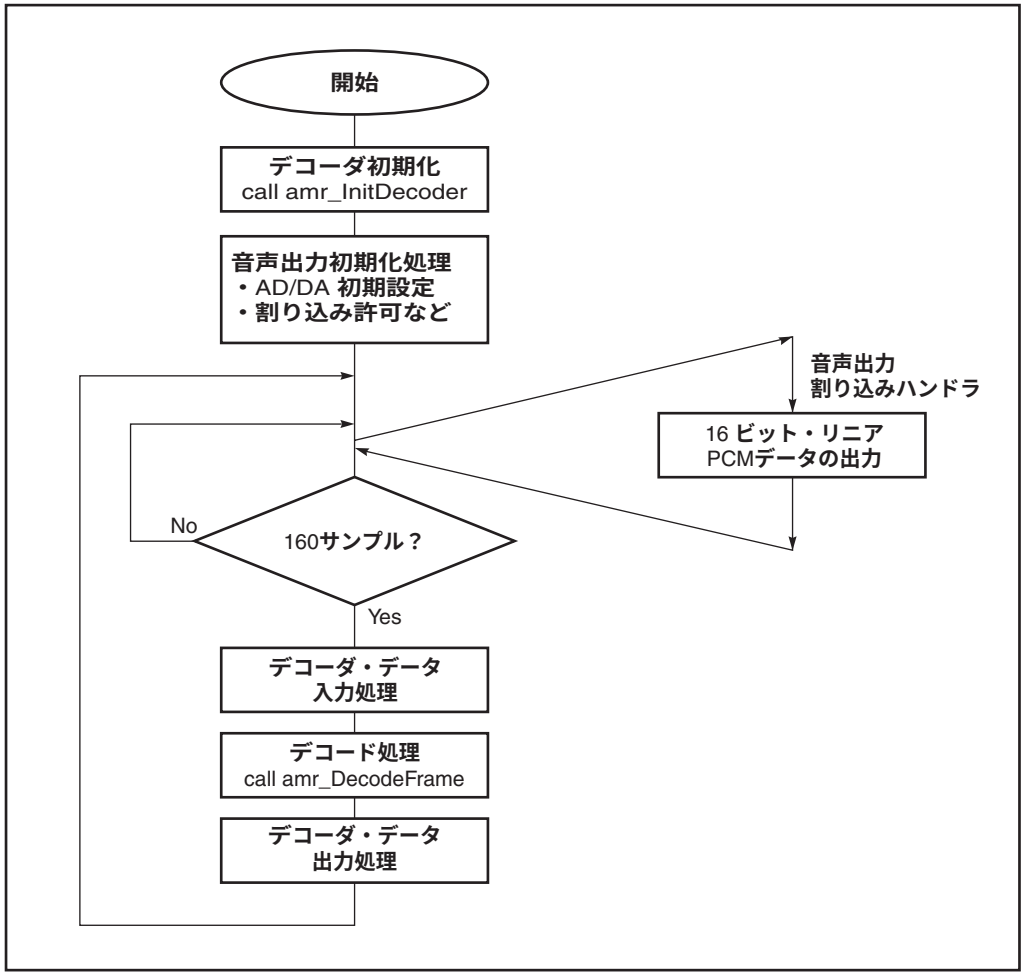


**図 2-2　アプリケーション処理フロー例（デコーダ）**



# 2. 2　関数仕様

## 2. 2. 1　amr_InitEncoder 関数

amr_InitEncoder 関数は，エンコーダで使用する各種定数，係数テーブル，バッファの初期化を行う関数です。エンコーダ使用時に 1 度だけ呼び出してください。

【　分　類　】エンコーダ初期化関数

【 関 数 名 】amr_InitEncoder

【機 能 概 要】AMR 音声 CODEC エンコーダの処理に必要な RAM 領域の初期化および VAD モードの設定を行い，amr_ResetEncoder 関数を呼び出し各種パラメータを初期設定します。

【　形　式　】call amr_InitEncoder

【　引　数　】

| 型 | 引　数 | 説　　明 |
|---|---|---|
| Register | R0L | DTX（無音圧縮機能）の ON/OFF（表 2-1参照） |

【 返 却 値 】なし

【使用レジスタ】R0, R1, DP0, DP1, DP4, DP5

【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 | 320 |
| 最大サイクル数 | 1256 |

**備考**　この関数を呼び出すときには，あらかじめスタティック・メモリおよびスクラッチ・メモリ領域を確保しておいてください。詳細は，**2. 3　メモリ構造**を参照してください。

**表 2-1　DTX モード**

| DTX モード | 値 | 説　　明 |
|---|---|---|
| DTX_OFF | 0 | 無音圧縮機能 OFF |
| DTX_ON_VAD1 | 1 | 無音圧縮機能 ON（VAD1 オプション） |
| DTX_ON_VAD2 | 2 | 無音圧縮機能 ON（VAD2 オプション） |

## 2. 2. 2　amr_ResetEncoder 関数

amr_ResetEncoder 関数は，エンコーダで使用する各種定数，係数テーブル，バッファを初期状態に戻します。

【　分　類　】エンコーダ・リセット関数
【 関 数 名 】amr_ResetEncoder
【機 能 概 要】AMR 音声 CODEC エンコーダの処理に必要な，各種パラメータを初期設定に戻します。
【　形　式　】call amr_ResetEncoder
【　引　数　】なし
【 返 却 値 】なし
【使用レジスタ】R0, R1, DP0, DP1, DP4, DP5
【ハードウエア・リソースメント】

| | |
|---|---|
| 最大スタック・レベル | 3 |
| 最大ループ・スタック・レベル | 1 |
| 最大リピート回数 | 320 |
| 最大サイクル数 | 1248 |

**備考**　この関数を呼び出すときには，あらかじめスタティック・メモリおよびスクラッチ・メモリ領域を確保しておいてください。詳細は，**2. 3　メモリ構造**を参照してください。

## 2. 2. 3　amr_EncodeFrame 関数

amr_EncodeFrame 関数は，16 ビット×160 サンプルの音声データを指定したビット・レートで圧縮します。また，初期化時に無音圧縮機能を ON に設定している場合，入力音声に応じて自動的に無音圧縮機能が有効になります。

【　分　類　】エンコード処理関数

【 関 数 名 】amr_EncodeFrame

【機 能 概 要】1 フレーム（160 サンプル）の音声を指定したビット・レートで圧縮します。

【　形　式　】call amr_EncodeFrame

【　引　数　】

| 型 | 引　数 | 説　　明 |
|---|---|---|
| Register | R0L | エンコード・モード（表 2-2参照，MRDTX は除く） |
| Register | DP4 | 音声入力バッファの先頭アドレス |
| Register | DP0 | ビット・ストリーム出力バッファの先頭アドレス |

【 返 却 値 】

| 型 | 返却値 | 説　　明 |
|---|---|---|
| Register | R0L | 圧縮に使用したエンコード・モード（表 2-2参照） |

【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 | 239 |
| ★ 最大 MIPS 数 | 21.7 |

**備考**　この関数を呼び出すときには，あらかじめスタティック・メモリおよびスクラッチ・メモリ領域を確保しておいてください。詳細は，2. 3　メモリ構造を参照してください。

**表 2-2　エンコーダ／デコーダの動作モード**

| コーデック・モード | 値 | 説　　明 |
|---|---|---|
| MR475 | 0 | 4.75 kbps での圧縮／伸長モード |
| MR515 | 1 | 5.15 kbps での圧縮／伸長モード |
| MR59 | 2 | 5.90 kbps での圧縮／伸長モード |
| MR67 | 3 | 6.70 kbps での圧縮／伸長モード |
| MR74 | 4 | 7.40 kbps での圧縮／伸長モード |
| MR795 | 5 | 7.95 kbps での圧縮／伸長モード |
| MR102 | 6 | 10.20 kbps での圧縮／伸長モード |
| MR122 | 7 | 12.20 kbps での圧縮／伸長モード |
| MRDTX | 8 | 無音圧縮機能が有効になったときの圧縮／伸長モード |

## 2. 2. 4　amr_InitDecoder 関数

amr_InitDecoder 関数は，デコーダで使用する各種定数，係数テーブル，バッファの初期化を行う関数です。デコーダ使用時に 1 度だけ呼び出してください。

【　分　類　】デコーダ初期化関数

【 関 数 名 】amr_InitDecoder

【機 能 概 要】AMR 音声 CODEC デコーダの処理に必要な RAM 領域の初期化を行い， amr_ResetDecoder 関数を呼び出し各種パラメータを初期設定します。

【　形　式　】call amr_InitDecoder

【　引　数　】なし

【 返 却 値 】なし

【使用レジスタ】R0, R1, R2, DP0, DP1, DP2, DP4, DP5, DP6, DN5

【ハードウエア・リソースメント】

| | | |
|---|---|---|
| | 最大スタック・レベル | 3 |
| | 最大ループ・スタック・レベル | 1 |
| | 最大リピート回数 | 154 |
| ★ | 最大サイクル数 | 2545 |

**備考**　この関数を呼び出すときには，あらかじめスタティック・メモリおよびスクラッチ・メモリ領域を確保しておいてください。詳細は，2. 3　**メモリ構造**を参照してください。

## 2. 2. 5　amr_ResetDecoder 関数

amr_ResetDecoder 関数は，デコーダで使用する各種定数，係数テーブル，バッファを初期状態に戻します。

【　分　類　】デコーダ・リセット関数

【 関 数 名 】amr_ResetDecoder

【 機 能 概 要 】AMR 音声 CODEC デコーダの処理に必要な，各種パラメータを初期設定に戻します。

【　形　式　】call amr_ResetDecoder

【　引　数　】なし

【 返 却 値 】なし

【使用レジスタ】R0, R1, R2, DP0, DP1, DP2, DP4, DP5, DP6, DN5

【ハードウエア・リソースメント】

| | | |
|---|---|---|
| | 最大スタック・レベル | 2 |
| | 最大ループ・スタック・レベル | 1 |
| | 最大リピート回数 | 154 |
| ★ | 最大サイクル数 | 1009 |

**備考**　この関数を呼び出すときには，あらかじめスタティック・メモリおよびスクラッチ・メモリ領域を確保しておいてください。詳細は，2. 3　**メモリ構造**を参照してください。

## 2. 2. 6　amr_DecodeFrame 関数

amr_DecodeFrame 関数は，圧縮されたビット・ストリーム・データを 16 ビット×160 サンプルの音声データに伸長します。

【　分　類　】デコード処理関数

【 関 数 名 】amr_DecodeFrame

【機 能 概 要】ビット・ストリームから，1 フレーム（160 サンプル）の音声を伸長します。

【　形　式　】call amr_DecodeFrame

【　引　数　】

| 型 | 引　数 | 説　　明 |
|---|---|---|
| Register | R0L | デコード・モード[注]（表 2-2　エンコーダ／デコーダの動作モード参照。MRDTX は除く） |
| Register | R1L | 受信フレーム・タイプ（表 2-3参照） |
| Register | DP4 | 音声出力バッファの先頭アドレス |
| Register | DP0 | ビット・ストリーム入力バッファの先頭アドレス |

注　受信フレーム・タイプが RX_NO_DATA の場合は，直前フレームのデコード・モードを引数としてください。

【 返 却 値 】なし

【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

【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 | 170 |
| 最大 MIPS 数 | 2.9 |

備考　この関数を呼び出すときには，あらかじめスタティック・メモリおよびスクラッチ・メモリ領域を確保しておいてください。詳細は，2. 3　メモリ構造を参照してください。

表 2-3　受信フレーム・タイプ

| RX_TYPE | 値 | 説　　明 |
|---|---|---|
| RX_SPEECH_GOOD | 0 | 音声フレーム（エラーなし） |
| RX_SPEECH_DEGRADED | 1 | 音声フレーム（少なくとも 1 ビット・エラーが含まれている） |
| RX_ONSET | 2 | ONSET フレーム |
| RX_SPEECH_BAD | 3 | 音声フレーム（エラーが含まれている） |
| RX_SID_FIRST | 4 | SID フレームの開始 |
| RX_SID_UPDATE | 5 | SID フレームのアップデート（エラーなし） |
| RX_SID_BAD | 6 | SID フレームのアップデート（エラーが含まれている） |
| RX_NO_DATA | 7 | データなしフレーム |

## 2. 2. 7　amr_sid_sync_init 関数

amr_sid_sync_init 関数は，SID（Silence Descriptor）同期機能で使用する各種定数，係数テーブル，バッファの初期化を行う関数です。エンコーダ使用時に 1 度だけ呼び出してください。

【　分　類　】SID 同期初期化関数
【 関 数 名 】amr_sid_sync_init
【 機 能 概 要 】SID 同期機能に必要な RAM 領域の初期化を行い，amr_sid_sync_reset 関数を呼び出し各種パラメータを初期設定します。
【　形　式　】call amr_sid_sync_init
【　引　数　】なし
【 返 却 値 】なし
【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 | 1 |
| 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 |

**備考**　この関数を呼び出すときには，あらかじめスタティック・メモリおよびスクラッチ・メモリ領域を確保しておいてください。詳細は，2. 3　**メモリ構造**を参照してください。

## 2. 2. 8　amr_sid_sync_reset 関数

amr_sid_sync_reset 関数は，SID（Silence Descriptor）同期機能で使用する各種定数，係数テーブル，バッファを初期状態に戻します。

【　分　類　】SID 同期初期化関数
【 関 数 名 】amr_sid_sync_reset
【 機 能 概 要 】SID 同期機能に必要な，各種パラメータを初期状態に戻します。
【　形　式　】call amr_sid_sync_reset
【　引　数　】なし
【 返 却 値 】なし
【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 | 8 |

**備考**　この関数を呼び出すときには，あらかじめスタティック・メモリおよびスクラッチ・メモリ領域を確保しておいてください。詳細は，2. 3　**メモリ構造**を参照してください。

## 2. 2. 9　amr_sid_sync 関数

amr_sid_sync 関数は，圧縮に使用したエンコード・モードに応じて送信フレーム・タイプを決定します。1 フレームのエンコード処理ごとに呼び出してください。

【　分　類　】SID 同期関数

【 関 数 名 】amr_sid_sync

【機 能 概 要 】圧縮に使用したエンコード・モードからフレーム・タイプを決定します。

【　形　式　】call amr_sid_sync

【　引　数　】

| 型 | 引　数 | 説　　明 |
|---|---|---|
| Register | R0L | 圧縮に使用したエンコード・モード（表 2-2　エンコーダ／デコーダの動作モード参照） |

【 返 却 値 】

| 型 | 返却値 | 説　　明 |
|---|---|---|
| Register | R0L | 送信フレーム・タイプ（表 2-4参照） |

【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 | 30 |

**備考**　この関数を呼び出すときには，あらかじめスタティック・メモリおよびスクラッチ・メモリ領域を確保しておいてください。詳細は，2. 3　メモリ構造を参照してください。

★

表 2-4　送信フレーム・タイプ

| TX_TYPE | 値 | 説　　明 |
|---|---|---|
| TX_SPEECH_GOOD | 0 | 音声フレーム |
| TX_SID_FIRST | 1 | SID フレームの開始 |
| TX_SID_UPDATE | 2 | SID フレームのアップデート |
| TX_NO_DATA | 3 | データなしフレーム |

## 2. 2. 10 amr_TX_to_RX 関数

amr_TX_to_RX 関数は，送信フレーム・タイプ（TX_TYPE）を受信フレーム・タイプ（RX_TYPE）に変換します。

【　分　類　】フレーム・タイプ変換関数

【 関 数 名 】amr_TX_to_RX

【機 能 概 要 】送信フレーム・タイプを受信フレーム・タイプに変換します。

【　形　式　】call amr_TX_to_RX

【　引　数　】

| 型 | 引　数 | 説　　明 |
|---|---|---|
| Register | R1L | 送信フレーム・タイプ（TX_TYPE） |

【 返 却 値 】

| 型 | 返却値 | 説　　明 |
|---|---|---|
| Register | R1L | 受信フレーム・タイプ（RX_TYPE） |

★ 【使用レジスタ】R1, R2, R3

【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 | 30 |

**備考**　各送信フレーム・タイプに対応する受信フレーム・タイプを表 2-5に示します。（　）内は各送受信フレーム・タイプの値です。これ以外の送信フレーム・タイプの場合には，0xFFFF を返却します。

★ **表 2-5　送受信フレーム・タイプ**

| 送信フレーム・タイプ | 受信フレーム・タイプ |
|---|---|
| TX_SPEEC_GOOD（= 0） | RX_SPEECH_GOOD（= 0） |
| TX_SID_FIRST（= 1） | RX_SID_FIRST（= 4） |
| TX_SID_UPDATE（= 2） | RX_SID_UPDATE（= 5） |
| TX_NO_DATA（= 3） | RX_SID_NO_DATA（= 7） |
| TX_SPEECH_DEGRADE（= 4） | RX_SPEECH_DEGRADE（= 1） |
| TX_SPEECH_BAD（= 5） | RX_SPEECH_BAD（= 3） |
| TX_SID_BAD（= 6） | RX_SID_BAD（= 6） |
| TX_ONSET（= 7） | RX_ONSET（= 2） |

## 2. 2. 11 amr_ehf_test 関数

amr_ehf_test 関数は，入力された音声信号がエンコーダのホーミング・フレームかどうか判定します。

【　分　類　】エンコーダ・ホーミング・フレーム・テスト関数

【 関 数 名 】amr_ehf_test

【機 能 概 要】音声データに対してホーミング・フレームの判定を行います。

【　形　式　】call amr_ehf_test

【　引　数　】

| 型 | 引　数 | 説　　明 |
|---|---|---|
| Register | DP4 | 音声バッファの先頭アドレス |

【 返 却 値 】

| 型 | 返却値 | 説　　明 |
|---|---|---|
| Register | R0L | 判定結果<br>0：ホーミング・フレームなし<br>1：ホーミング・フレームあり |

【使用レジスタ】R0, R1, DP4

【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 | 1 |
| 最大リピート回数 | 0 |
| 最大サイクル数 | 967 |

## 2. 2. 12 amr_dhf_test 関数

amr_dhf_test 関数は，入力されたビット・ストリームがデコーダのホーミング・フレームかどうか判定します。

【　分　類　】デコーダ・ホーミング・フレーム・テスト関数

【 関 数 名 】amr_dhf_test

【機 能 概 要】ビット・ストリーム・データに対してホーミング・フレームの判定を行います。

【　形　式　】call amr_dhf_test

【　引　数　】

| 型 | 引　数 | 説　　明 |
|---|---|---|
| Register | DP0 | ビット・ストリーム・バッファの先頭アドレス |
| Register | R6L | デコード・モード（表 2-2　エンコーダ／デコーダの動作モード参照，MRDTX は除く） |

【 返 却 値 】

| 型 | 返却値 | 説　　明 |
|---|---|---|
| Register | R0L | 判定結果<br>0：ホーミング・フレームなし<br>1：ホーミング・フレームあり |

【使用レジスタ】R0, R1, R2, R3, R4, R5, R6, R7, DP0, DP1, DP2, DP4, DP5, DP6, DP7

【ハードウエア・リソースメント】

| | |
|---|---|
| 最大スタック・レベル | 2 |
| 最大ループ・スタック・レベル | 1 |
| 最大リピート回数 | 0 |
| 最大サイクル数 | 1858 |

## 2. 2. 13 amr_GetVersion 関数

amr_GetVersion 関数は，AMR 音声 CODEC ライブラリのバージョン情報を返します。

【　分　類　】バージョン情報獲得関数

【 関 数 名 】amr_GetVersion

【機 能 概 要】バージョン情報を返します。

【　形　式　】call amr_GetVersion

【　引　数　】なし

【 返 却 値 】

| 型 | 返却値 | 説　明 |
|---|---|---|
| Register | R0H | ライブラリのメジャー・バージョン番号 |
| Register | R0L | ライブラリのマイナ・バージョン番号 |
| Register | R1H | AMR 音声 CODEC のメジャー・バージョン番号 |
| Register | R1L | AMR 音声 CODEC のマイナ・バージョン番号 |

【　機　能　】ライブラリのバージョンと AMR 音声 CODEC のバージョンを返します。

★ 例　R0=0x00’0x0002’0x0000 → ライブラリ・バージョン Ver 2.00.00

R1=0x00’0x0007’0x0600 → AMR 音声 CODEC のバージョン Ver 7.6.0

【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　メモリ構造

μSAP77016-B06 のライブラリで必要とするメモリの構造について説明します。

μSAP77016-B06 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。定義したシンボル名は，必ず PUBLIC 擬似命令を使用してください。それぞれのメモリ・サイズについては，表 2-6を参照してください。

### (1) スクラッチ・メモリ領域

保存の必要のない領域です。1 フレームのエンコード／デコード処理のあと，ユーザが自由に使用することができます。

例

```
public    lib_Scratch_x
public    lib_Scratch_y

SCRATCH_X            XRAMSEG
lib_Scratch_x:       DS        AMR_MAX_SCRATCH_X_SIZE

SCRATCH_Y            YRAMSEG
lib_Scratch_y:       DS        AMR_MAX_SCRATCH_Y_SIZE
```

### (2) スタティック・メモリ領域

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，μSAP77016-B06 のライブラリの正常な動作は保証されません。

例

```
public      amr_Static_enc_x
public      amr_Static_enc_y
public      amr_Static_dec_x
public      amr_Static_dec_y

AMR_STATIC_X         XRAMSEG
amr_Static_enc_x:    DS        AMR_MAX_STATIC_ENC_X_SIZE
amr_Static_dec_x:    DS        AMR_MAX_STATIC_DEC_X_SIZE

AMR_STATIC_Y         YRAMSEG
amr_Static_enc_y:    DS        AMR_MAX_STATIC_ENC_Y_SIZE
amr_Static_dec_y:    DS        AMR_MAX_STATIC_DEC_Y_SIZE
```

表 2-6　メモリのシンボル名とメモリ・サイズ

| シンボル名 | サイズ［ワード］ | X/Y 面 | 説　明 |
|---|---|---|---|
| lib_Scratch_x | 2025 | X | エンコーダ／デコーダ共通スクラッチ領域 |
| lib_Scratch_y | 909 | Y | エンコーダ／デコーダ共通スクラッチ領域 |
| amr_Static_enc_x | 509 | X | エンコーダ用スタティック領域 |
| amr_Static_enc_y | 955 | Y | エンコーダ用スタティック領域 |
| amr_Static_dec_x | 402 | X | デコーダ用スタティック領域 |
| amr_Static_dec_y | 525 | Y | デコーダ用スタティック領域 |

備考　メモリ領域は，amr_lib.h で定義されているマクロで確保できます（2. 4　マクロ参照）。

### (3) 入出力バッファ

音声データおよびビット・ストリーム・データの入出力用の領域です。エンコーダ／デコーダが使用する入出力バッファは，1 フレームのエンコード／デコード処理のあと，ユーザが自由に使用できます。エンコード／デコード処理中にそれらの領域を操作すると正常な動作は保証されません。

・エンコーダに必要な入出力バッファ

| | |
|---|---|
| ビット・ストリーム・データの出力バッファ | X メモリ空間に 16 ワード |
| 音声データの入力バッファ | Y メモリ空間に 160 ワード |

例

```
I_O_BUFFER_X        XRAMSEG
bitstream_out:      DS      16
I_O_BUFFER_Y        YRAMSEG
speech_in:          DS      160
```

・デコーダに必要な入出力バッファ

| | |
|---|---|
| ビット・ストリーム・データの入力バッファ | X メモリ空間に 16 ワード |
| 音声データの出力バッファ | Y メモリ空間に 160 ワード |

例

```
I_O_BUFFER_X        XRAMSEG
bitstream_in:       DS      16
I_O_BUFFER_Y        YRAMSEG
speech_out:         DS      160
```

**備考**　一般的には，圧縮／伸長をリアルタイムで行う場合，音声データの入出力バッファをダブル・バッファ構成にします。その場合，エンコーダ／デコーダが使用する入出力バッファとして Y メモリ空間に 320 ワード，シリアル入出力などに使用する入出力バッファとして X または Y メモリ空間に 320 ワードのデータ・メモリが必要になります。

**【シミュレータにおける注意点】**

μSAP77016-B06 のライブラリでは，ポインタ処理の高速化のため，確保した領域以外のメモリ空間にリード・アクセスしています。したがって，メモリ・アロケーションの状態によっては，シミュレータで警告が出ることがあります。

**表 2-7　メモリ・アクセスの範囲**

| 領域名 | 説　明 |
|---|---|
| amr_Static_enc_x | この領域以外のメモリ空間にアクセスしません。 |
| amr_Static_enc_y | この領域以外のメモリ空間にアクセスしません。 |
| amr_Static_dec_x | この領域以外のメモリ空間にアクセスしません。 |
| amr_Static_dec_y | この領域以外のメモリ空間にアクセスしません。 |
| lib_Scratch_x | この領域の終端から＋5 ワードまでの範囲のメモリ・アクセスを行います。 |
| lib_Scratch_y | この領域の先頭から－1 ワードまでの範囲のメモリ・アクセスを行います。 |

# 2. 4　マクロ

## 2. 4. 1　AMR_AllocateScratchMemory マクロ

【　分　類　】スクラッチ・メモリ確保

【マクロ名】AMR_AllocateScratchMemory

【機能概要】AMR 音声 CODEC の処理に必要なスクラッチ・メモリ領域を確保し，シンボル名を宣言します。

【　形　式　】%AMR_AllocateScratchMemory

【　引　数　】なし

【返却値】なし

【定義シンボル】lib_Scratch_x, lib_Scratch_y

## 2. 4. 2　AMR_AllocateStaticMemoryForEncoder マクロ

【　分　類　】エンコーダ用スタティック・メモリ確保

【マクロ名】AMR_AllocateStaticMemoryForEncoder

【機能概要】AMR 音声 CODEC のエンコード処理に必要なスタティック・メモリ領域を確保し，シンボル名を宣言します。

【　形　式　】% AMR_AllocateStaticMemoryForEncoder

【　引　数　】なし

【返却値】なし

【定義シンボル】amr_Static_enc_x, amr_Static_enc_y

## 2. 4. 3　AMR_AllocateStaticMemoryForDecoder マクロ

【　分　類　】デコーダ用スタティック・メモリ確保

【マクロ名】AMR_AllocateStaticMemoryForDecoder

【機能概要】AMR 音声 CODEC のデコード処理に必要なスタティック・メモリ領域を確保し，シンボル名を宣言します。

【　形　式　】% AMR_AllocateStaticMemoryForDecoder

【　引　数　】なし

【返却値】なし

【定義シンボル】amr_Static_dec_x, amr_Static_dec_y

**備考**　μSAP77016-B06 のライブラリで使用する，マクロは amr_lib.h で定義されています。使用するためには，amr_lib.h をインクルードしてください。

# 第3章 ビット・ストリー ム・フォーマット

μSAP77016-B06 の各モードにおけるビット・ストリームのビット・アロケーションを表 3-1から表 3-9に示します。

表 3-1　MR122 モードのビット・アロケーション

| ビット位置 | 説　　明 |
|---|---|
| b1 - b7 | index of 1st LSF submatrix |
| b8 - b15 | index of 2nd LSF submatrix |
| b16 - b23 | index of 3rd LSF submatrix |
| b24 | sign of 3rd LSF submatrix |
| b25 - b32 | index of 4th LSF submatrix |
| b33 - b38 | index of 5th LSF submatrix |
| 第 1 サブフレーム | |
| b39 - b47 | adaptive codebook index |
| b48 - b51 | adaptive codebook gain |
| b52 | sign information for 1st and 6th pulses |
| b53 - b55 | position of 1st pulse |
| b56 | sign information for 2nd and 7th pulses |
| b57 - b59 | position of 2nd pulse |
| b60 | sign information for 3rd and 8th pulses |
| b61 - b63 | position of 3rd pulse |
| b64 | sign information for 4th and 9th pulses |
| b65 - b67 | position of 4th pulse |
| b68 | sign information for 5th and 10th pulses |
| b69 - b71 | position of 5th pulse |
| b72 - b74 | position of 6th pulse |
| b75 - b77 | position of 7th pulse |
| b78 - b80 | position of 8th pulse |
| b81 - b83 | position of 9th pulse |
| b84 - b86 | position of 10th pulse |
| b87 - b91 | fixed codebook gain |
| 第 2 サブフレーム | |
| b92 - b97 | adaptive codebook index (relative) |
| b98 - b141 | same description as b48 - b91 |
| 第 3 サブフレーム | |
| b142 - b194 | same description as b39 - b91 |
| 第 4 サブフレーム | |
| b195 - b244 | same description as b92 - b141 |

表 3-2　MR102 モードのビット・アロケーション

| ビット位置 | 説　　明 |
|---|---|
| b1 - b8 | index of 1st LSF subvector |
| b9 - b17 | index of 2nd LSF subvector |
| b18 - b26 | index of 3rd LSF subvector |
| 第 1 サブフレーム | |
| b27 - b34 | adaptive codebook index |
| b35 | sign information for 1st and 5th pulses |
| b36 | sign information for 2st and 6th pulses |
| b37 | sign information for 3rd and 7th pulses |
| b38 | sign information for 4th and 8th pulses |
| b39 - b48 | position of 1st, 2nd, and 5th pulse |
| b49 - b58 | position of 3rd, 6th, and 7th pulse |
| b59 - b65 | position of 4th and 8th pulse |
| b66 - b72 | codebook gains |
| 第 2 サブフレーム | |
| b73 - b77 | adaptive codebook index (relative) |
| b78 - b115 | same description as b35 - b72 |
| 第 3 サブフレーム | |
| b116 - b161 | same description as b27 - b72 |
| 第 4 サブフレーム | |
| b162 - b204 | same description as b73 - b115 |

表 3-3　MR795 モードのビット・アロケーション

| ビット位置 | 説　　明 |
|---|---|
| b1 - b9 | index of 1st LSF subvector |
| b10 - b18 | index of 2nd LSF subvector |
| b19 - b27 | index of 3rd LSF subvector |
| 第 1 サブフレーム | |
| b28 - b35 | adaptive codebook index |
| b36 - b39 | position of 4th pulse |
| b40 - b42 | position of 3rd pulse |
| b43 - b45 | position of 2nd pulse |
| b46 - b48 | position of 1st pulse |
| b49 | sign information for 4th pulse |
| b50 | sign information for 3rd pulse |
| b51 | sign information for 2nd pulse |
| b52 | sign information for 1st pulse |
| b53 - b56 | adaptive codebook gain |
| b57 - b61 | fixed codebook gain |
| 第 2 サブフレーム | |
| b62 - b67 | adaptive codebook index (relative) |
| b68 - b93 | same description as b36 - b61 |
| 第 3 サブフレーム | |
| b94 - b127 | same description as b28 - b61 |
| 第 4 サブフレーム | |
| b128 - b159 | same description as b62 - b93 |

表 3-4　MR74 モードのビット・アロケーション

| ビット位置 | 説　　明 |
|---|---|
| b1 - b8 | index of 1st LSF subvector |
| b9 - b17 | index of 2nd LSF subvector |
| b18 - b26 | index of 3rd LSF subvector |
| 第１サブフレーム | |
| b27 - b34 | adaptive codebook index |
| b35 - b38 | position of 4th pulse |
| b39 - b41 | position of 3rd pulse |
| b42 - b44 | position of 2nd pulse |
| b45 - b47 | position of 1st pulse |
| b48 | sign information for 4th pulse |
| b49 | sign information for 3rd pulse |
| b50 | sign information for 2nd pulse |
| b51 | sign information for 1st pulse |
| b52 - b58 | codebook gains |
| 第 2 サブフレーム | |
| b59 - b63 | adaptive codebook index (relative) |
| b64 - b87 | same description as b35 - b58 |
| 第 3 サブフレーム | |
| b88 - b119 | same description as b27 - b58 |
| 第 4 サブフレーム | |
| b120 - b148 | same description as b59 - b87 |

表 3-5　MR67 モードのビット・アロケーション

| ビット位置 | 説　　明 |
|---|---|
| b1 - b8 | index of 1st LSF subvector |
| b9 - b17 | index of 2nd LSF subvector |
| b18 - b26 | index of 3rd LSF subvector |
| 第１サブフレーム | |
| b27 - b34 | adaptive codebook index |
| b35 - b38 | position of 3rd pulse |
| b39 - b42 | position of 2nd pulse |
| b43 - b45 | position of 1st pulse |
| b46 | sign information for 3rd pulse |
| b47 | sign information for 2nd pulse |
| b48 | sign information for 1st pulse |
| b49 - b55 | codebook gains |
| 第 2 サブフレーム | |
| b56 - b59 | adaptive codebook index (relative) |
| b60 - b80 | same description as b35 - b55 |
| 第 3 サブフレーム | |
| b81 - b109 | same description as b27 - b55 |
| 第 4 サブフレーム | |
| b110 - b134 | same description as b56 - b80 |

表 3-6　MR59 モードのビット・アロケーション

| ビット位置 | 説　　明 |
|---|---|
| b1 - b8 | index of 1st LSF subvector |
| b9 - b17 | index of 2nd LSF subvector |
| b18 - b26 | index of 3rd LSF subvector |
| 第１サブフレーム | |
| b27 - b34 | adaptive codebook index |
| b35 - b39 | position of 2nd pulse |
| b40 - b43 | position of 1st pulse |
| b44 | sign information for 2nd pulse |
| b45 | sign information for 1st pulse |
| b46 - b51 | codebook gains |
| 第 2 サブフレーム | |
| b52 - b55 | adaptive codebook index(relative) |
| b56 - b72 | same description as b35 - b51 |
| 第 3 サブフレーム | |
| b73 - b97 | same description as b27 - b51 |
| 第 4 サブフレーム | |
| b98 - b118 | same description as b52 - b72 |

表 3-7　MR515 モードのビット・アロケーション

| ビット位置 | 説　　明 |
|---|---|
| b1 - b8 | index of 1st LSF subvector |
| b9 - b16 | index of 2nd LSF subvector |
| b17 - b23 | index of 3rd LSF subvector |
| 第 1 サブフレーム | |
| b24 - b31 | adaptive codebook index |
| b32 | position subset |
| b33 - b35 | position of 2nd pulse |
| b36 - b38 | position of 1st pulse |
| b39 | sign information for 2nd pulse |
| b40 | sign information for 1st pulse |
| b41- b46 | codebook gains |
| 第 2 サブフレーム | |
| b47 - b50 | adaptive codebook index (relative) |
| b51 - b65 | same description as b32 - b46 |
| 第 3 サブフレーム | |
| b66 - b84 | same description as b47 - b65 |
| 第 4 サブフレーム | |
| b85 - b103 | same description as b47 - b65 |

表 3-8　MR475 モードのビット・アロケーション

| ビット位置 | 説　　明 |
|---|---|
| b1 - b8 | index of 1st LSF subvector |
| b9 - b16 | index of 2nd LSF subvector |
| b17 - b23 | index of 3rd LSF subvector |
| 第 1 サブフレーム | |
| b24 - b31 | adaptive codebook index |
| b32 | position subset |
| b33 - b35 | position of 2nd pulse |
| b36 - b38 | position of 1st pulse |
| b39 | sign information for 2nd pulse |
| b40 | sign information for 1st pulse |
| b41- b48 | codebook gains |
| 第 2 サブフレーム | |
| b49 - b52 | adaptive codebook index(relative) |
| b53 - b61 | same description as b32 - b40 |
| 第 3 サブフレーム | |
| b62 - b65 | same description as b49 - b52 |
| b66 - b82 | same description as b32 - b48 |
| 第 4 サブフレーム | |
| b83 - b95 | same description as b49 - b61 |

**表 3-9　MRDTX モードのビット・アロケーション**

| ビット位置 | 説　　明 |
|---|---|
| b1 - b3 | index of reference vector |
| b4 - b11 | index of 1st LSF subvector |
| b12 - b20 | index of 2nd LSF subvector |
| b21 - b29 | index of 3rd LSF subvector |
| b30 - b35 | index of logarithmic frame energy |

**表** 3-1**～表** 3-9**における，ビット・アロケーション**（b1 - b242）**とワード・データとの関係を次に示します。**

**図 3-1　ビット・アロケーションとワードの関係**

| Offset | MSB | | | | | | | | | | | | | | | LSB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| +0 | b1 | b2 | b3 | b4 | b5 | b6 | b7 | b8 | b9 | b10 | b11 | b12 | b13 | b14 | b15 | b16 |
| +1 | b17 | | | | | | • | • | • | • | | | | | | b32 |
| +2 | b33 | | | | | | • | • | • | • | | | | | | b48 |
| +3 | b49 | | | | | | • | • | • | • | | | | | | b64 |
| +4 | b65 | | | | | | • | • | • | • | | | | | | b80 |
| +5 | b81 | | | | | | • | • | • | • | | | | | | b96 |
| +6 | b97 | | | | | | • | • | • | • | | | | | | b112 |
| +7 | b113 | | | | | | • | • | • | • | | | | | | b128 |
| +8 | b129 | | | | | | • | • | • | • | | | | | | b144 |
| +9 | b145 | | | | | | • | • | • | • | | | | | | b160 |
| +10 | b161 | | | | | | • | • | • | • | | | | | | b176 |
| +11 | b177 | | | | | | • | • | • | • | | | | | | b192 |
| +12 | b193 | | | | | | • | • | • | • | | | | | | b208 |
| +13 | b209 | | | | | | • | • | • | • | | | | | | b224 |
| +14 | b225 | | | | | | • | • | • | • | | | | | | b240 |
| +15 | b241 | | | | | | • | • | • | • | | | | | | b256 |

# 第4章 インストレーション

## 4.1 インストレーション手順

μSAP77016-B06 の提供媒体は，3.5 インチ・フロッピィ・ディスク（1.44MB）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。なお，この作業を行う場合には，Windows95，98，2000 または Windows NT® 4.0 以上の OS がインストールされたホスト・マシンを使用してください。

① 提供媒体をフロッピィ・ディスク・ドライブにセットしてください。

② フロッピィ・ディスク内の amr_mw.exe を実行してください（μSAP77016-B06 のファイル群は，自己解凍形式で圧縮されています）。ここで，A ドライブをフロッピィ・ディスク・ドライブ，C ドライブをインストール先のハード・ディスクとします。

```
A:¥>amr_mw.exe<CR>
```

③ インストールするディレクトリを指定するダイアログ・ボックスが表示されます。
インストールするディレクトリを指定してください。これは任意のディレクトリで構いませんが，ここでは，C:¥DSPTools を指定したとします。

④ 「OK」ボタンをクリックすると，ファイル群が展開されます。

⑤ ファイル群が展開されたことを確認してください。各ディレクトリにつては，1. 3. 4 ディレクトリ構成を参照してください。

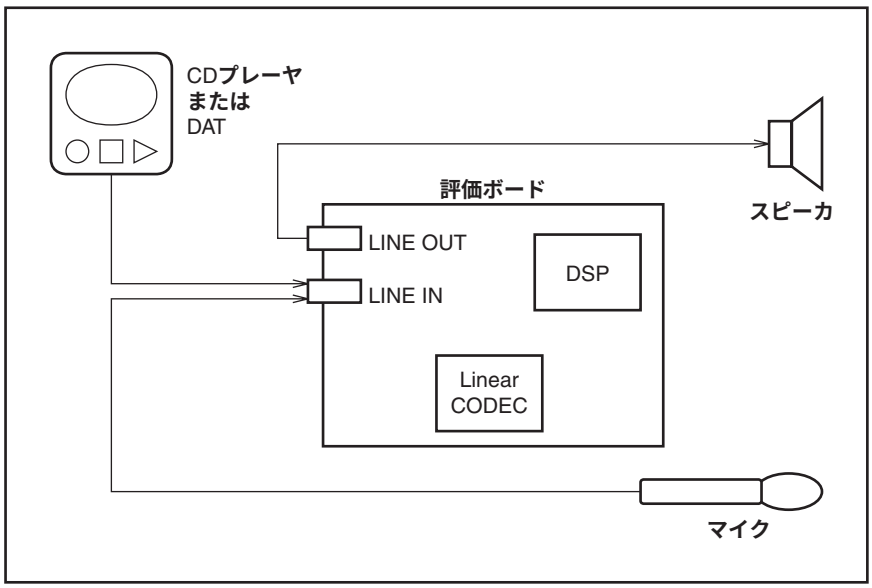
```
A:¥>dir C:¥DSPTools<CR>
```

## 4. 2　サンプル作成手順

μSAP77016-B06 では，smp ディレクトリに，シングル・チャネルおよびマルチ・チャネル（2 チャネル）用のサンプル・プログラムを格納しています（1. 3. 4　ディレクトリ構成参照）。シングル・チャネル用のソース・プログラムについては，付録 A　サンプル・プログラム・ソースを参照してください。

サンプル・プログラムを利用することで，実際に CD プレーヤまたは DAT やマイク，スピーカを接続して AMR 音声 CODEC システムを評価することができます。ただし，必要に応じてシステム依存部などのコードを追加してください。

図 4-1　サンプル・プログラム評価システム例




次にμSAP77016-B06 のサンプル・プログラム（シングル・チャネル用）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）プロジェクトファイル「sample.prj」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）ID77016（ディバッガ）を使用して，ターゲットに sample.lnk をダウンロードして実行します。

## 4. 3　ロケーションの変更

μSAP77016-B06 のライブラリでは，表 4-1に示すセクション名が付けられています。ユーザのターゲットに合わせて，ロケーションを変更できます。

表 4-1　セクション名

| セクション名 | タイプ | 内　　容 |
|---|---|---|
| __AMR_CONST_X__ | XROMSEG/XRAMSEG | 定数データ格納領域 |
| __AMR_CONST_Y__ | YROMSEG/YRAMSEG | 定数データ格納領域 |
| __AMR_CODEC__ | IROMSEG/IRAMSEG | AMR 音声 CODEC プログラム |

## 4. 4　シンボル名規約

μSAP77016-B06 のライブラリで使用するシンボル名などは，次に示す規約に従っています。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，重複しないようにしてください。

表 4-2　シンボル命名規約

| 分　　類 | 命名規則 |
|---|---|
| 関数名 | amr_XXXX |
| 定数シンボル名 | amr_cnst_XXXX |
| スタティック RAM 領域シンボル名 | amr_Static_[enc\|dec]_[x\|y] |
| スクラッチ RAM 領域シンボル名 | lib_Scratch_[x\|y] |
| コード・セグメント名（IMSEG） | __AMR_CODEC__ |
| 定数セグメント名（ROMSEG / RAMSEG） | __AMR_CONST_[X\|Y]__ |
| マクロ，定数名（#define，equ 宣言など） | AMR_XXXX |
| ファイル名 | amrXXXX.* |

備考　“ XXXX ”は任意の英数字とします。

# ★ 付録 A　サンプル・プログラム・ソース

μSAP77016-B06（AMR 音声 CODEC ミドルウエア）のサンプル・ソースを次に示します。このサンプル・ソースは，3GPP（3rd Generation Partnership Project）のサンプル・ソースを参考に作成しています。各処理の詳細は，3GPP のサンプル・ソースを参照してください。

・sample.asm　(1/7)

```
/*------------------------------------------------------------------------*/
/*     File Information                                                   */
/*------------------------------------------------------------------------*/
/*     Name    : sample.asm                                               */
/*     Type    : Assembler program module                                 */
/*     Version : 2.00                                                     */
/*     Date    : 2002 APR 24                                              */
/*     CPU     : uPD7701x Family                                          */
/*     Assembler: WB77016                                                 */
/*     About   : GSM AMR speech codec   Version 7.6.0                     */
/*                                                                        */
/*------------------------------------------------------------------------*/
/*     Copyright (C) NEC Corporation 1999, 2000, 2001, 2002               */
/*     NEC CONFIDENTIAL AND PROPRIETARY                                   */
/*     All rights reserved by NEC Corporation.                            */
/*     Use of copyright notice does not evidence publication              */
/*------------------------------------------------------------------------*/

/* ========================================================================
 *  Include files
 * ======================================================================== */
#include "upd7701x.mac"
#include "amr_lib.h"

/* ========================================================================
 *  Memory allocate for AMR CODEC
 * ======================================================================== */
   %AMR_AllocateScratchMemory
   %AMR_AllocateStaticMemoryForEncoder
   %AMR_AllocateStaticMemoryForDecoder

/* ========================================================================
 *  Local variables and tables
 * ======================================================================== */
#define DTX        DTX_OFF        /* set DTX mode.    DTX_OFF, DTX_ON_VAD1, DTX_ON_VAD2  */
#define MULTI      0              /* set MULTI mode.  0: normal mode. 1:multi mode       */

#define L_FRAME    160

WORK_X  XRAMSEG at 0x0000
   f_run:          ds  1            ; codec start flag, "1" is start codec.
   f_reset:        ds  1            ; test sequence reset flag, "-1" is reset.
   f_multi:        dw  MULTI        ; multi mode flag, not "0" is multi rate mode.
   frame_type:     ds  1            ; RX/TX frame type.
   used_mode:      ds  1            ; used AMR Encoder mode.
   bitstream:      ds  16           ; bitstream load/store buffer.
   e_mode:         dw  MR475        ; AMR Encoder mode.
```

• sample.asm　(2/7)

```
   d_mode:        ds  1            ; AMR Decoder mode.
   prev_d_mode:   ds  1            ; AMR previous codec mode
   dtx_mode:      dw  DTX          ; DTX mode.
   f_e_reset:     ds  1            ; encoder reset flag.
   f_d_reset:     ds  1            ; decoder reset flag.
   f_d_reset_old: ds  1            ; decoder old reset flag.
   r7save:        ds  3            ; r7 register load/sotre buffer for interrupt handler.
   dp7save:       ds  1            ; dp7 register load/sotre buffer for interrupt handler.
   mode_cnt:      ds  1            ; coder mode count for multi rate mode.
   frame_cnt:     ds  1            ; frame count.

WORK_Y  YRAMSEG at 0x0000
   si1_ptr:       ds  1            ;
   si1_buff:      ds  L_FRAME      ;
   so1_buff:      ds  L_FRAME      ;
   speech_in:     ds  L_FRAME      ;
   speech_out:    ds  L_FRAME      ;

#define F_RUN                *f_run:x
#define F_RESET              *f_reset:x
#define F_MULTI              *f_multi:x
#define TX_TYPE              *frame_type:x
#define RX_TYPE              *frame_type:x
#define USED_MODE            *used_mode:x

#define E_MODE               *e_mode:x
#define D_MODE               *d_mode:x
#define DTX_MODE             *dtx_mode:x
#define PREV_D_MODE          *prev_d_mode:x
#define E_RESET_FLAG         *f_e_reset:x
#define D_RESET_FLAG         *f_d_reset:x
#define D_RESET_FLAG_OLD     *f_d_reset_old:x

#define MODE_COUNT           *mode_cnt:x
#define FRAME_COUNT          *frame_cnt:x

/* ===========================================================================
 *  Vector registration
 * ===========================================================================*/
%BeginVector(StartUp)              ;Regist start up routine
   %NotUseVector(VectorINT1)       ;
   %NotUseVector(VectorINT2)       ;
   %NotUseVector(VectorINT3)       ;
   %NotUseVector(VectorINT4)       ;
   %RegistVector(VectorSI1, SI1Handler)    ;Regist SI1 handler
   %NotUseVector(VectorSO1)        ;
   %NotUseVector(VectorSI2)        ;
   %NotUseVector(VectorSO2)        ;
   %NotUseVector(VectorHI)         ;
   %NotUseVector(VectorHO)         ;
%EndVector
```

• sample.asm　(3/7)

```
/* ========================================================================
 *  Programe code section
 * ======================================================================*/
MAIN_CODE   IMSEG at 0x4000
StartUp:
   ;;=============================================================;;
   ;;  Initialize uPD7701x                                        ;;
   ;;=============================================================;;
   %InitializeSystemRegister      ;Initialize system register
   %ClearAllRegister              ;Clear all uPD7701x register

   ;;=============================================================;;
   ;;  Initialize Register & Peripheral Units                     ;;
   ;;=============================================================;;
   r0l = 0x0200                   ;
   *SST1:x = r0l                  ;

   ;;=============================================================;;
   ;;  Initialize AMR CODEC module                                ;;
   ;;=============================================================;;
   /* Initialize AMR encoder */
   clr(r0)                        ;
   r0l = DTX_MODE                 ;set DTX mode
   call amr_InitEncoder           ;
   call amr_sid_sync_init         ;
   /* Initialize AMR decoder */
   call amr_InitDecoder           ;

   ;;=============================================================;;
   ;;  Initialize buffer                                          ;;
   ;;=============================================================;;
   /* clear serial input/output buffer */
   r0l = si1_buff                 ;Initialize serial i/o buffer pointer.
   *si1_ptr:y = r0l               ;
   dp4 = si1_buff                 ;Initialize serial i/o buffer.
   rep L_FRAME*2                  ;
      *dp4++ = r0h                ;
   /* clear speech input/output buffer */
   dp4 = speech_in                ;
   dp5 = speech_out               ;
   loop L_FRAME {                 ;
      *dp4++ = r0h                ;
      *dp5++ = r0h                ;
   }                              ;
   /* clear flags */
   clr(r0)                        ;
   F_RUN = r0l                    ;clear codec start flag.
   F_RESET = r0l                  ;clear codec rest  flag.

   TX_TYPE = r0l                  ;clear TX type.
   RX_TYPE = r0l                  ;clear RX type.
   USED_MODE = r0l                ;clear used AMR encoder mode.

   D_MODE = r0l                   ;clear AMR decoder mode.
   PREV_D_MODE = r0l              ;clear prev AMR decoder mode.

   E_RESET_FLAG = r0l             ;clear encoder reset flag.
   D_RESET_FLAG = r0l             ;clear decoder reset flag.
```

• sample.asm　(4/7)

```
    MODE_COUNT = r0l                ;clear mode count.
    FRAME_COUNT = r0l               ;clear frame count.

    r0l = 1                         ;
    D_RESET_FLAG_OLD = r0l          ;set decoder old reset flag.

    ;;=============================================================;;
    ;;  Set interrupt mask                                         ;;
    ;;=============================================================;;
    %DisableInterrupt               ;
    %SetMask(SR_ALL)                ;
    %UnSetMask(SR_SI1)              ;
    %EnableInterrupt                ;

    ;;=============================================================;;
    ;;  Main routine                                               ;;
    ;;=============================================================;;
MainLoop:
    /* check reset codec flag */
    r0 = F_RESET                    ;
    if(r0 < 0) jmp StartUp          ;
    /* check start codec flag */
    r0 = F_RUN                      ;
    if(r0 == 0) jmp MainLoop        ;
    clr(r0)                         ;
    F_RUN = r0l                     ;clear codec start flag.
    /* increment frame count */
    r0l = FRAME_COUNT               ;
    r0 = r0 + 1                     ;
    FRAME_COUNT = r0l               ;
    /* Copy input/output PCM data */
    dp4 = speech_in                 ;
    dp5 = speech_out                ;
    dp6 = si1_buff                  ;
    loop L_FRAME {                  ;
        r0l = *dp6++                ;read input PCM data from si1_buff.
        r0 = r0 & 0xfff8            ;
        *dp4++ = r0l                ;store PCM data to speech_in.
    }                               ;
    loop L_FRAME {                  ;
        r0l = *dp5++                ;read output PCM data from speech_out.
        r0 = r0 & 0xfff8            ;
        *dp6++ = r0l                ;store PCM data to so1_buff.
    }                               ;
    /* Encode process */
    call Proc_AMR_Encoder           ;
    /* Decode process */
    call Proc_AMR_Decoder           ;
    jmp MainLoop                    ;

/* ============================================================================
[Function Name] Proc_AMR_Encoder
 ============================================================================*/
Proc_AMR_Encoder:
    /* check for homing frame */
    dp4 = speech_in                 ;set speech buffer address.
    call amr_ehf_test               ;test homing frame.
    E_RESET_FLAG = r0l              ;save reset flag.
```

• sample.asm　(5/7)

```
    /* multi rate mode */
    r0 = F_MULTI                    ;
    if(r0 != 0) call MultiRateMode  ;

    /* encode speech */
    clr(r0)                         ;
    r0l = E_MODE                    ;set AMR encoder mode.
    dp4 = speech_in                 ;set speech buffer address.
    dp0 = bitstream                 ;set bitstream buffer address.
    call amr_EncodeFrame            ;encode speech data.
    USED_MODE = r0l                 ;save used AMR encoder mode.

    /*include frame type and mode information in serial bitstream */
    clr(r0)                         ;
    r0l = USED_MODE                 ;set used AMR encoder mode.
    call amr_sid_sync               ;
    TX_TYPE = r0l                   ;save TX frame type.
    r0 = r0 - TX_NO_DATA            ;
    if( r0 != 0 ) jmp proc_enc_noupdate_mode;
        clr(r0)                     ;
        r0 = r0 - 1                 ;
        D_MODE = r0l                ;
        jmp proc_enc_next           ;
proc_enc_noupdate_mode:
        clr(r0)                     ;
        r0l = E_MODE                ;
        D_MODE = r0l                ;

proc_enc_next:
    /* perform homing if homing frame was detected at encoder input */
    r0 = E_RESET_FLAG               ;
    if(r0 == 0) ret                 ;
        call amr_ResetEncoder       ;
        call amr_sid_sync_reset     ;
        ret                         ;

/* ========================================================================
[Function Name] Proc_AMR_Decoder
 ========================================================================*/
Proc_AMR_Decoder:
    /* Convert TX frame type to RX frame type*/
    clr(r1)                         ;
    r1l = TX_TYPE                   ;set TX frame type.
    call amr_TX_to_RX               ;
    RX_TYPE = r1l                   ;save RX frame type.

    clr(r1)                         ;
    r1l = RX_TYPE                   ;
    r0 = r1 - RX_NO_DATA            ;
    if(r0 != 0) jmp proc_dec_set_prev_mode;
        r0l = PREV_D_MODE           ;
        D_MODE = r0l                ;
        jmp proc_dec_set_prev_mode_end;
proc_dec_set_prev_mode:
        r0l = D_MODE                ;
        PREV_D_MODE = r0l           ;
proc_dec_set_prev_mode_end:
```

• sample.asm　(6/7)

```
    /* if homed: check if this frame is another homing frame */
    r0 = D_RESET_FLAG_OLD           ;
    if(r0 == 0) jmp proc_dec_next_1 ;
        /* only check until end of first subframe */
        clr(r6)                     ;
        R6l = D_MODE                ;set AMR decoder mode.
        dp0 = bitstream             ;set bitstream buffer address.
        call amr_dhf_test           ;test homing frame.
        D_RESET_FLAG = r0l          ;save reset flag.

proc_dec_next_1:
    /* produce encoder homing frame if homed & input=decoder homing frame */
    r0 = D_RESET_FLAG               ;
    if(r0 == 0) jmp proc_dec_start  ;
    r0 = D_RESET_FLAG_OLD           ;
    if(r0 == 0) jmp proc_dec_start  ;
        r0l = EHF_MASK              ;
        dp4 = speech_out            ;
        rep L_FRAME                 ;
            *dp4++ = r0l            ;
        jmp proc_dec_next_2         ;

proc_dec_start:
    /* decode frame */
    clr(r0)                         ;
    clr(r1)                         ;
    r0l = D_MODE                    ;set AMR decoder mode.
    r1l = RX_TYPE                   ;set RX_TYPE = RX_SPEECH.
    dp0 = bitstream                 ;set bitstream buffer address.
    dp4 = speech_out                ;set speech buffer address.
    call amr_DecodeFrame            ;decoder bitstream.

    /* if not homed: check whether current frame is a homing frame */
proc_dec_next_2:
    r0 = D_RESET_FLAG_OLD           ;
    if(r0 != 0) jmp proc_dec_next_3 ;
        /* check whole frame */
        clr(r6)                     ;
        R6l = D_MODE                ;set AMR decoder mode.
        dp0 = bitstream             ;set bitstream buffer address.
        call amr_dhf_test           ;test homing frame.
        D_RESET_FLAG = r0l          ;save reset flag.

    /* reset decoder if current frame is a homing frame */
proc_dec_next_3:
    r0 = D_RESET_FLAG               ;
    if(r0 != 0) call amr_ResetDecoder;
    r0 = D_RESET_FLAG               ;
    D_RESET_FLAG_OLD = r0h          ;
    ret                             ;
```

• sample.asm　(7/7)

```
/* ============================================================================
[Function Name] MultiRateMode
 ============================================================================*/
MultiRateMode:
    clr(r0)                         ;
    r0l = MODE_COUNT                ;
    r1 = r0 & 0x7                   ;
    E_MODE = r1l                    ;
    r1 = r1 + 1                     ;
    MODE_COUNT = r1l                ;
    ret                             ;

/* ============================================================================
[Handler Name]   SI1Handler
[RAM]            si1_buff, so1_buff, si1_ptr, f_run
[Use Register]   r7, dp7
[MIPS]           0.200[MIPS] (25*159+27*1[cycle])
[Use Stacks]     loop stack: 0, call stack: 0, repeat: 0
 ============================================================================*/
SI1Handler:
    /* Save r7, dp7 register */
    *r7save+0:x = r7l               ;save r7l
    *r7save+1:x = r7h               ;save r7h
    *r7save+2:x = r7e               ;save r7e
    r7l = dp7                       ;save dp7
    *dp7save:x = r7l                ;
    /* input/output PCM data */
    r7l = *si1_ptr:y                ;
    dp7 = r7l                       ;
    r7h = *SDT1:x                   ;
    *dp7##160 = r7h                 ;
    r7h = *dp7##-159                ;
    *SDT1:x = r7h                   ;
    /* check frame count */
    clr(r7)                         ;
    r7l = dp7                       ;
    *si1_ptr:y = r7l                ;
    r7 = r7 - ( si1_buff + L_FRAME ) ;
    if(r7 != 0) jmp si1_end         ;
    r7l = 1                         ;
    F_RUN = r7l                     ;
    r7l = si1_buff                  ;
    *si1_ptr:y = r7l                ;
si1_end:
    /* Restore r7, dp7 register */
    r7l = *dp7save:x                ;
    dp7 = r7l                       ;load dp7
    r7e = *r7save+2:x               ;load r7e
    r7h = *r7save+1:x               ;load r7h
    r7l = *r7save+0:x               ;load r7l
    reti                            ;

/* End of file */
END
```

# 付録 B　関連文書

3GPP から発行された AMR 音声 CODEC 関連の勧告書を次に示します。

- 3GPP TS 26.071　AMR speech Codec; General description
- 3GPP TS 26.073　AMR speech Codec; C-source code
- 3GPP TS 26.074　AMR speech Codec; Test sequences
- 3GPP TS 26.090　AMR speech Codec; Transcoding Functions
- 3GPP TS 26.091　AMR speech Codec; Error concealment of lost frames
- 3GPP TS 26.092　AMR speech Codec; comfort noise
- 3GPP TS 26.093　AMR speech Codec; Source Controlled Rate operation
- 3GPP TS 26.094　AMR speech Codec; Voice Activity Detector
- 3GPP TS 26.101　AMR speech Codec; Frame Structure
- 3GPP TS 26.102　AMR speech Codec; Interface to Iu and Uu

〔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-B06　ユーザーズ・マニュアル

（U15165JJ3V0UM00（第 3 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