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-B03

G.729 音声コーデック・ミドルウエア

対象デバイス

μPD77016
μPD77017
μPD77018
μPD77018A
μPD77019
μPD77110
μPD77111
μPD77112
μPD77113
μPD77114
μPD77116

資料番号　U13373JJ3V1UM00（第 3 版）
発行年月　December　1999 NS CP（K）

［メ　モ］

目次要約

第１章　概　　説　…　13

第２章　ライブラリ仕様　…　19

第３章　インストレーション　…　41

付　録　サンプル・ソース　…　45

［メ　モ］

M7A 98.8

本版で改訂された主な箇所

箇　所	内　　容
全般	対象デバイスにμPD77110，77111，77112，77113，77114，77116を追加
全般	G.729 ANNEX B マルチチャネル版に関する内容を追加
p.15	1.5.2（3）ソフトウエア・ツール（Windows™ 版）のバージョンを更新
p.31	2.3　関数仕様（マルチチャネル版）を追加
p.38	2.5　マルチチャネル版外部インタフェースを追加
p.58	付.2　マルチチャネル版用サンプル・ソース（samplebm.asm）を追加

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　本マニュアルは，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，μPD77015，77016，77017，77018，77018A，77019，77110，77111，77112，77113，77114，77116[注]の総称です。ただし，このマニュアルでは，μPD77015 を対象デバイスにしていませんので，注意してください。

注　開発中

目　的　μPD77016 ファミリの応用，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・マニュアルでは，基本的な数値演算プログラムなどについて説明しています。

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
付　録	サンプル・ソース

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

μPD77016 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。

μPD77016 ファミリの命令機能を知りたいとき
→μPD7701×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。

デバイスに関する資料

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	U12395J	U10891J	U10503J	U13116J	U11958J	U12021J
μPD77015		U10902J				
μPD77017						
μPD77018						
μPD77018A		U11849J				
μPD77019						
μPD77019-013		U13053J				
μPD77110		U12801J	作成中			
μPD77111						
μPD77112						
μPD77113		U14373J				
μPD77114						

開発ツールに関する資料

資　料　名		資料番号
SM77016　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
IE-77016-98，IE-77016-PC　ユーザーズ・マニュアル	ハードウエア編	U13044J
μPD77016　スタータ・キット　ユーザーズ・マニュアル		U13032J
IE-77016-CM-LC　ユーザーズ・マニュアル		U14139J
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

第１章　概　　説　　… 13

1.1 ミドルウエア … 13
1.2 G.729 音声コーデック … 13
1.3 G.729 ANNEX A 音声コーデック … 14
1.4 G.729 ANNEX B 音声コーデック … 14
1.5 システム概要 … 14
1.5.1 特　　徴 … 14
1.5.2 動作環境 … 14
1.5.3 性　　能 … 15
1.5.4 ディレクトリ構成 … 16

第２章　ライブラリ仕様… 19

2.1 G.729 音声コーデック処理フロー … 19
2.2 関数仕様（シングルチャネル版） … 22
2.2.1 エンコーダ初期化関数 … 22
2.2.2 デコーダ初期化関数 … 24
2.2.3 エンコーダ関数 … 26
2.2.4 デコーダ関数 … 28
2.2.5 バージョン取得関数 … 30
2.3 関数仕様（マルチチャネル版） … 31
2.3.1 スクラッチ領域初期化関数 … 31
2.3.2 エンコーダ初期化関数 … 32
2.3.3 デコーダ初期化関数 … 33
2.3.4 エンコーダ関数 … 34
2.3.5 デコーダ関数 … 35
2.4 コントロール／ステータス・レジスタ（シングルチャネル版） … 36
2.4.1 G729E_CMD_STS（G729AE_CMD_STS，
G729BE_CMD_STS，G729ABE_CMD_STS）レジスタ … 36
2.4.2 G729D_CMD_STS（G729AD_CMD_STS，
G729BD_CMD_STS，G729ABD_CMD_STS）レジスタ … 37
2.5 マルチチャネル版外部インタフェース … 38
2.5.1 G.729 ANNEX B マルチチャネル版用関数の引き渡しパラメータ … 38
2.6 圧縮データ・フォーマット … 39
2.6.1 有音データの圧縮フォーマット … 39
2.6.2 SID フレームの圧縮フォーマット … 40

第３章　インストレーション　…　41

3.1　インストレーション手順　…　41
3.2　サンプル作成手順　…　42
3.3　シンボル名規約　…　43

付録　サンプル・ソース　…　45

付.1　シングルチャネル版用サンプル・ソース（sample.asm）　…　45
付.2　マルチチャネル版用サンプル・ソース（samplebm.asm）　…　58
付.2.1　samplebm.asm 用ヘッダ・ファイル（sysconf.h）　…　75

図の目次

図番号　　タイトル，ページ

2－1　**アプリケーション処理フロー（エンコーダ）**　…　20
2－2　**アプリケーション処理フロー（デコーダ）**　…　21

3－1　**サンプル・プログラム評価システム**　…　42

表の目次

表番号　　タイトル，ページ

1－1　**必要メモリの容量**　…　15

2－1　G729E_CMD_STS **レジスタ**　…　36
2－2　G729D_CMD_STS **レジスタ**　…　37
2－3　G.729 ANNEX B **マルチチャネル版用関数の引き渡しパラメータ**　…　38
2－4　**有音圧縮データのビット割り当て**　…　39
2－5　**有音圧縮データの意味**　…　39
2－6　**無音圧縮データのビット割り当て**　…　40
2－7　**無音圧縮データの意味**　…　40

3－1　**シンボル名**　…　43

［メ　　モ］

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整ってきたために，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-B03 は，ITU-T [注] 勧告 G.729（付属勧告 ANNEX A，ANNEX B を含む）音声圧縮伸長機能を提供するミドルウエアです。このマニュアルでは、特に指定のない限り，G.729 音声コーデックとして使用した場合を例にして説明しています。

注　International Telecommunication Union - Telecommunication Standardization Sector

1.2　G.729 音声コーデック

G.729 音声コーデックは，ITU-T で勧告された 8 Kbps の音声圧縮伸長コーデックで，CS-ACELP [注]（共役構造－代数的符号励振線形予測）を使用した音声信号を符号化するためのアルゴリズムです。

G.729 音声コーデックは，電話帯域フィルタ（ITU-T 勧告 G.712）により帯域制限されたアナログ入力信号を 8 kHz で標本化し，次に 16 ビット・リニア PCM データに変換することによって得られたディジタル信号を，符号器の入力として動作するように設計されています。同様に，復号器の出力もアナログ信号に戻される必要があります。

ITU-T 勧告 G.711 で規定される 64 Kbps PCM データのようなほかの入出力形式の信号は，符号器の前で 16 ビット・リニア PCM へ，また復号器のあとで 16 ビット・リニア PCM から適当な形式へ変換する必要があります。符号器から復号器へ渡されるビット列は ITU-T 勧告 G.729 で規定されています。

注　Conjugate Structure-Algebraic Code Excited Linear Prediction

1.3 G.729 ANNEX A 音声コーデック

ITU-T 勧告 G.729 の付属勧告 ANNEX A は，G.729 音声コーデックの低演算量版であり，このコーデックは標準の G.729 音声コーデックと相互接続可能です。低演算量版音声コーデックは，音声とデータを同時に利用するマルチメディア・アプリケーションのために開発されたものですが，特にこのアプリケーションに限定されるものではありません。

1.4 G.729 ANNEX B 音声コーデック

ITU-T 勧告 G.729 の付属勧告 ANNEX B は，G.729 音声コーデックの無音圧縮機能であり，標準の G.729 音声コーデックまたは，ANNEX A に付加して使用します。無音圧縮機能を付加した音声コーデックは，付加していない音声コーデックとの相互接続ができません。この無音圧縮機能は，無音部分の圧縮率を上げることによるビット・レートの軽減を目的としています。

1.5 システム概要

1.5.1 特　　徴

- 8 Kbps に圧縮符号化。
- 高品質音声符号化（32 Kbps ADPCM と同等の音質）。
- 8 kHz のサンプリング周波数で，80 サンプルを 1 フレームとして符号／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。

1.5.2 動作環境

★ (1) 動作対象 DSP

μPD77016, 77017 注1, 77018, 77018A, 77019, 77110, 77111, 77112, 77113, 77114, 77116 注2

注 1. ANNEX B を付加する場合，μPD77017 は動作対象外です。
2. 開発中

注意　μPD77015 は動作対象デバイスから除きます。

(2) 必要メモリ

★ 表 1−1 必要メモリの容量

種別 / メモリ[ワード]		G.729	ANNEX A	ANNEX B	ANNEX AB 注	ANNEX B
		シングルチャネル版				マルチチャネル版
命令メモリ	−	8.8 K	7.7 K	12.2 K	11.2 K	7.5 K
X メモリ	RAM	1.8 K	1.7 K	1.9 K	1.8 K	942×N+1153
	ROM	3.1 K	2.8 K	3.1 K	2.8 K	2.6 K
Y メモリ	RAM	1.9 K	1.7 K	2.0 K	1.8 K	914×N+1388
	ROM	2.0 K	1.8 K	2.4 K	2.2 K	2.0 K

注 ANNEX AB：ANNEX A＋ANNEX B

備考 N：チャネル数

★ (3) ソフトウエア・ツール（Windows™版）

DSP ツール： ワークベンチ WB77016 Ver2.32

ハイスピード・シミュレータ HSM77016 Ver2.3

1.5.3 性 能

【条 件】 DSP ：μPD77016 ファミリ（動作周波数 33 MHz，33 MIPS）

【1フレームの処理をリアルタイム（10 ms）に実行するために必要な MIPS 値】

（G.729） 圧縮処理：15.7 MIPS

伸長処理：3.3 MIPS

（G.729 ANNEX A） 圧縮処理：9.0 MIPS

伸長処理：2.2 MIPS

（G.729 ANNEX B） 圧縮処理：16.5 MIPS

伸長処理：3.5 MIPS

（G.729 ANNEX A ＋ANNEX B） 圧縮処理：9.5 MIPS

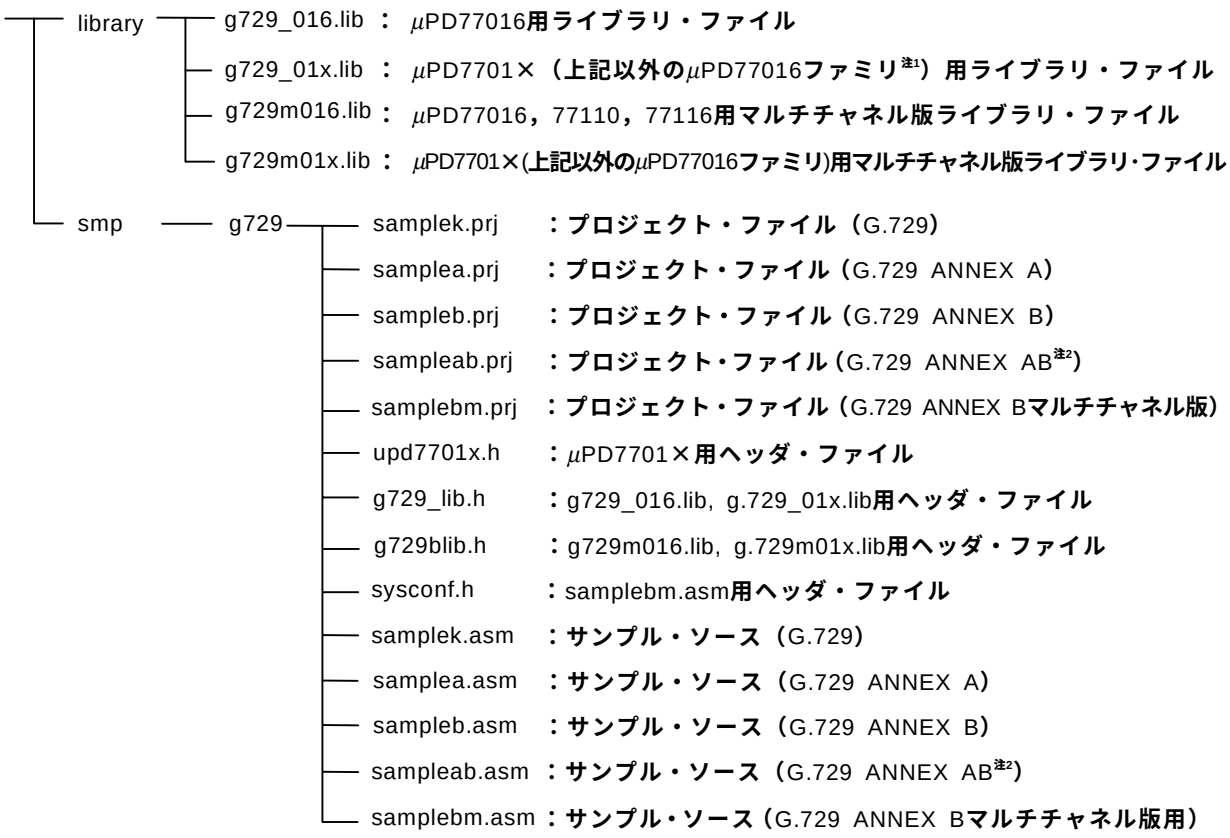
伸長処理：3.7 MIPS

★ （G.729 ANNEX B マルチチャネル版） 圧縮処理：13.9×N MIPS （N：チャネル数）

伸長処理：2.8×N MIPS （N：チャネル数）

★ 1.5.4 ディレクトリ構成

μSAP77016-B03 のディレクトリ構成を次に示します。



注1. μPD77016, 77110, 77116 **は対象外ですので注意してください。**g729_01x.lib **は，**μPD77017, 77018, 77018A, 77019, 77111, 77112, 77113, 77114 **のためのファイルです。**

2. ANNEX AB：ANNEX A＋ANNEX B

ライブラリ・ファイル g729_016.lib **および** g729_01x.lib **は，シングルチャネル版用です。次のオブジェクト・ファイルを含んでいます。**

- g729kenc.rel （G.729 **エンコーダ**）
- g729kdec.rel （G.729 **デコーダ**）
- g729aenc.rel （G.729 ANNEX A **エンコーダ**）
- g729adec.rel （G.729 ANNEX A **デコーダ**）
- g729benc.rel （G.729 ANNEX B **エンコーダ**）
- g729bdec.rel （G.729 ANNEX B **デコーダ**）
- g729abenc.rel （G.729 ANNEX A＋ANNEX B **エンコーダ**）
- g729abdec.rel （G.729 ANNEX A＋ANNEX B **デコーダ**）
- g729getv.rel （**バージョン取得関数**）

ライブラリ・ファイル g729m016.lib および g729m01x.lib は，マルチチャネル版用です。次のオブジェクト・ファイルを含んでいます。

- g729bmen.rel （G.729 ANNEX B マルチチャネル版エンコーダ）
- g729bmde.rel （G.729 ANNEX B マルチチャネル版デコーダ）
- g729bmco.rel （G.729 ANNEX B マルチチャネル版共通パート）

注意 g729m016.lib は，μPD77016, 77110, 77116 用のファイルです。g729bm01x.lib は，μPD77018, 77018 A, 77019, 77111, 77112, 77113, 77114 用のファイルです。

［メ　モ］

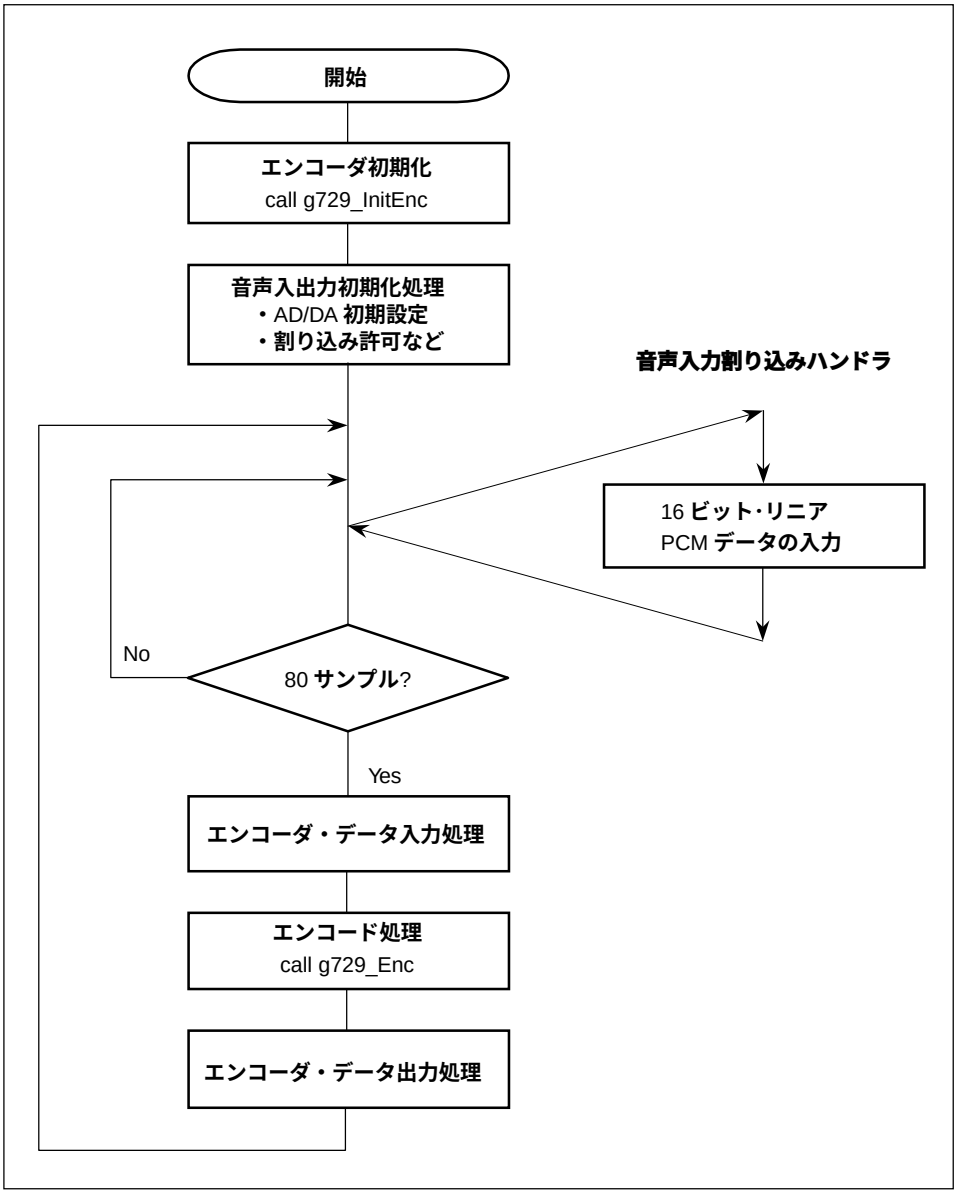
第２章　ライブラリ仕様

この章では，G.729 音声コーデックの関数仕様と呼び出し規約について説明します。

2.1　G.729 音声コーデック処理フロー

G.729 音声コーデックを使用したアプリケーションの処理の流れを図 2－1，2－2 に示します。

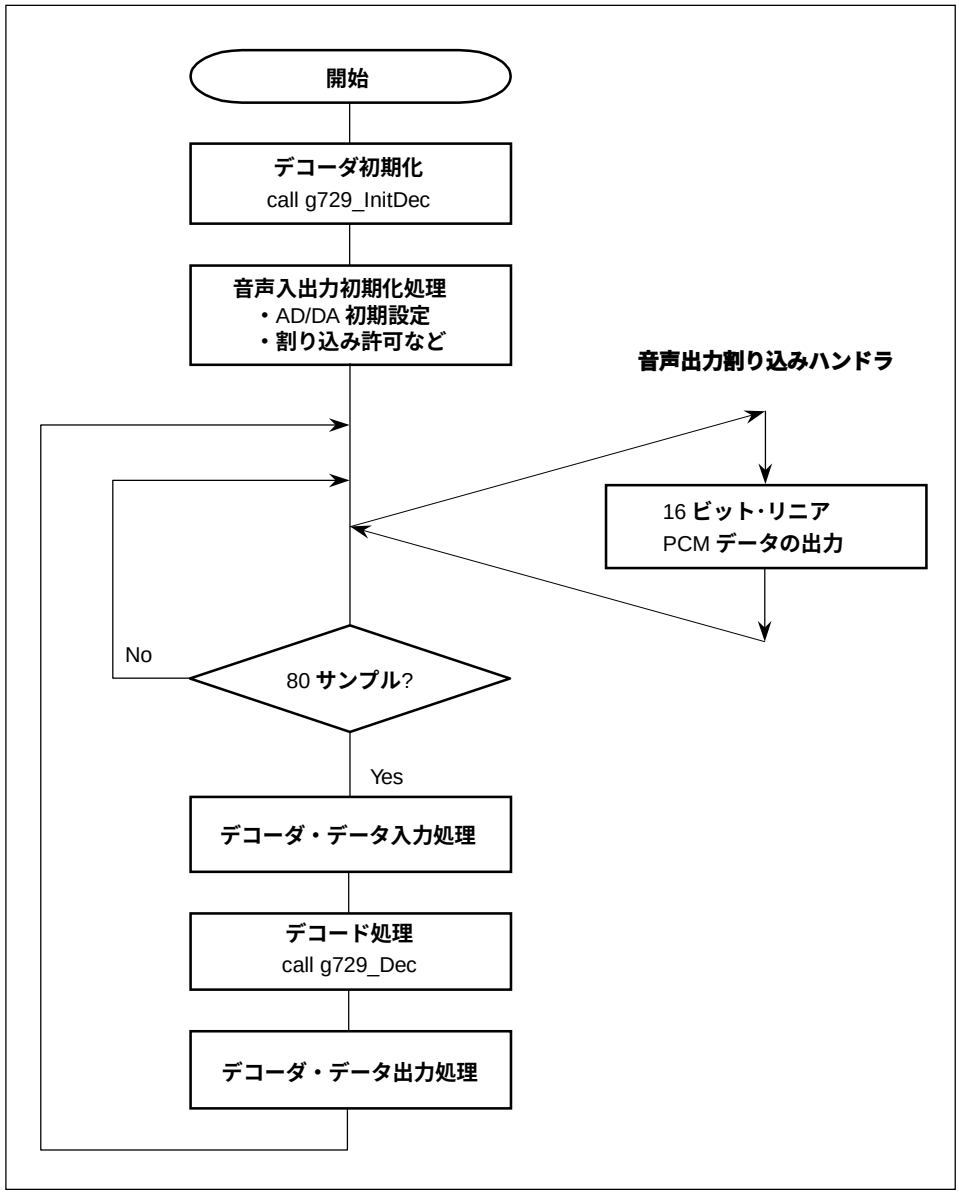
図２−１　アプリケーション処理フロー（エンコーダ）



備考　G.729 ANNEX A，ANNEX B，ANNEX A＋ANNEX B の場合は，G.729 用関数名の“ g729_ ”の部分がそれぞれ“ g729a_ ”，“ g729b_ ”，“ g729ab_ ”になります。

★　また，G.729 ANNEX B マルチチャネル版の場合は，開始直後に“ g729b_Start Codec ”を，“ g729_InitEnc ”の部分で“ g729b_InitEncM ”を，“ g729_Enc ”の部分で “ g729b_EncM ”をそれぞれコールしてください。

図２−２　アプリケーション処理フロー（デコーダ）



備考　G.729 ANNEX A，ANNEX B，ANNEX A＋ANNEX B の場合は，G.729 用関数名の“ g729_ ”の部分がそれぞれ“ g729a_ ”，“ g729b_ ”，“ g729ab_ ”になります。

★　また，G.729 ANNEX B マルチチャネル版の場合は，開始直後に“ g729b_Start Codec ”を，“ g729_IniDec ”の部分で“ g729b_InitDecM ”を，“ g729_Dec ”の部分で “ g729b_DecM ”をそれぞれコールしてください。

2.2　関数仕様（シングルチャネル版）

2.2.1　エンコーダ初期化関数

g729_InitEnc（ g729a_InitEnc，g729b_InitEnc，g729ab_InitEnc ）関数は，エンコーダ各定数の設定，係数テーブル，遅延バッファの初期化を行う関数です。

（1）G.729 用エンコーダ初期化関数

【　分　類　】　エンコーダ初期化処理
【 関 数 名 】　g729_InitEnc
【機 能 概 要】　G.729 エンコーダの使用する RAM 領域の初期化および各種パラメータを設定する。
【　形　式　】　call g729_InitEnc
【 引 き 数 】　なし
【 返 り 値 】　なし
【　機　能　】　G.729 エンコーダの初期化，各種パラメータの設定などを行う。
【使用レジスタ】　R0, DP0, DP1, DP2, DP3, DP4, DP7, DN0, DN2, DMX
【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	240
最大サイクル数	842

（2）G.729 ANNEX A 用エンコーダ初期化関数

【　分　類　】　エンコーダ初期化処理
【 関 数 名 】　g729a_InitEnc
【機 能 概 要】　G.729 ANNEX A エンコーダの使用する RAM 領域の初期化および各種パラメータを設定する。
【　形　式　】　call g729a_InitEnc
【 引 き 数 】　なし
【 返 り 値 】　なし
【　機　能　】　G.729 ANNEX A エンコーダの初期化，各種パラメータの設定などを行う。
【使用レジスタ】　R0, DP0, DP1, DP2, DP3, DP4, DP7, DN0, DN2, DMX
【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	240
最大サイクル数	860

(3) G.729 ANNEX B 用エンコーダ初期化関数

【 分　類 】 エンコーダ初期化処理
【 関 数 名 】 g729b_InitEnc
【機 能 概 要】 G.729 ANNEX B エンコーダの使用する RAM 領域の初期化および各種パラメータを設定する。
【 形　式 】 call g729b_InitEnc
【 引 き 数 】 なし
【 返 り 値 】 なし
【 機　能 】 G.729 ANNEX B エンコーダの初期化，各種パラメータの設定などを行う。
【使用レジスタ】 R0, R1, R2, DP0, DP1, DP2, DP3, DP4, DP7, DN0, DN2, DMX
【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	240
最大サイクル数	1177

(4) G.729 ANNEX A＋ANNEX B 用エンコーダ初期化関数

【 分　類 】 エンコーダ初期化処理
【 関 数 名 】 g729ab_InitEnc
【機 能 概 要】 G.729 ANNEX A＋ANNEX B エンコーダの使用する RAM 領域の初期化および各種パラメータを設定する。
【 形　式 】 call g729ab_InitEnc
【 引 き 数 】 なし
【 返 り 値 】 なし
【 機　能 】 G.729 ANNEX A＋ANNEX B エンコーダの初期化，各種パラメータの設定などを行う。
【使用レジスタ】 R0, R1, R2, DP0, DP1, DP2, DP3, DP4, DP7, DN0, DN2, DMX
【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	240
最大サイクル数	1195

2.2.2　デコーダ初期化関数

g729_InitDec（ g729a_InitDec，g729b_InitDec，g729ab_InitDec）関数は，デコーダ各定数の設定，係数テーブル，遅延バッファの初期化を行う関数です。

（1）G.729 用デコーダ初期化関数

【 分　類 】 デコーダ初期化処理
【 関 数 名 】 g729_InitDec
【機 能 概 要】 G.729 デコーダの使用する RAM 領域の初期化および各種パラメータを設定する。
【 形　式 】 call g729_InitDec
【 引 き 数 】 なし
【 返 り 値 】 なし
【 機　能 】 G.729 デコーダの初期化，各種パラメータの設定などを行う。
【使用レジスタ】 R0, R1, R2, DP0, DP1, DP2, DP4, DP7, DN0, DN2, DMX
【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	549

（2）G.729 ANNEX A 用デコーダ初期化関数

【 分　類 】 デコーダ初期化処理
【 関 数 名 】 g729a_InitDec
【機 能 概 要】 G.729 ANNEX A デコーダの使用する RAM 領域の初期化および各種パラメータを設定する。
【 形　式 】 call g729a_InitDec
【 引 き 数 】 なし
【 返 り 値 】 なし
【 機　能 】 G.729 ANNEX A デコーダの初期化，各種パラメータの設定などを行う。
【使用レジスタ】 R0, R1, R2, DP0, DP1, DP2, DP4, DP7, DN0, DN2, DMX
【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	183
最大サイクル数	939

（3）G.729 ANNEX B 用デコーダ初期化関数

【 分　類 】 デコーダ初期化処理
【 関 数 名 】 g729b_InitDec
【機 能 概 要】 G.729 ANNEX B デコーダの使用する RAM 領域の初期化および各種パラメータを設定する。
【 形　式 】 call g729b_InitDec
【 引 き 数 】 なし
【 返 り 値 】 なし
【 機　能 】 G.729 ANNEX B デコーダの初期化，各種パラメータの設定などを行う。
【使用レジスタ】 R0, R1, R2, DP0, DP1, DP2, DP4, DP7, DN0, DN2, DMX
【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	760

（4）G.729 ANNEX A＋ANNEX B 用デコーダ初期化関数

【 分　類 】 デコーダ初期化処理
【 関 数 名 】 g729ab_InitDec
【機 能 概 要】 G.729 ANNEX A＋ANNEX B デコーダの使用する RAM 領域の初期化および各種パラメータを設定する。
【 形　式 】 call g729ab_InitDec
【 引 き 数 】 なし
【 返 り 値 】 なし
【 機　能 】 G. 729 ANNEX A＋ANNEX B デコーダの初期化，各種パラメータの設定などを行う。
【使用レジスタ】 R0, R1, R2, DP0, DP1, DP2, DP4, DP7, DN0, DN2, DMX
【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	183
最大サイクル数	1120

2.2.3　エンコーダ関数

エンコーダ関数は，入力した 80 サンプルの音声信号を 80 ビットまたは 16 ビットに圧縮した信号を生成します。

（1）G.729 用エンコーダ関数

【　分　類　】　エンコード処理部
【 関 数 名 】　g729_Enc
【機 能 概 要】　80 サンプル×16 ビットを 80 ビットに圧縮する。
【　形　式　】　call g729_Enc
【 引 き 数 】　G729E_PCM_BUF[80](X)　入力データ
　　*G729E_CMD_STS:y　コントロール／ステータス・レジスタ
【 返 り 値 】　G729E_ANA_BUF[5](Y)　圧縮された信号
【　機　能　】　コーデックからの入力信号（80 サンプル×16 ビット）を 80 ビットに圧縮する。
【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
【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	2
最大リピート回数	240
最大 MIPS 値	15.7

（2）G.729 ANNEX A 用エンコーダ関数

【　分　類　】　エンコード処理部
【 関 数 名 】　g729a_Enc
【機 能 概 要】　80 サンプル×16 ビットを 80 ビットに圧縮する。
【　形　式　】　call g729a_Enc
【 引 き 数 】　G729AE_PCM_BUF[80](X)　入力データ
　　*G729AE_CMD_STS:y　コントロール／ステータス・レジスタ
【 返 り 値 】　G729AE_ANA_BUF[5](Y)　圧縮された信号
【　機　能　】　コーデックからの入力信号（80 サンプル×16 ビット）を 80 ビットに圧縮する。
【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
【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	2
最大リピート回数	240
最大 MIPS 値	9.0

（3）G.729 ANNEX B 用エンコーダ関数

【 分　類 】 エンコード処理部

【 関 数 名 】 g729b_Enc

【機 能 概 要】 80 サンプル×16 ビットを 80 ビットまたは 16 ビットに圧縮する。

【 形　式 】 call g729b_Enc

【 引 き 数 】

G729BE_PCM_BUF[80](X)	入力データ
*G729BE_CMD_STS:y	コントロール／ステータス・レジスタ
*G729Bframe:y	frame 数のカウンタ（0-0x7fff） （0x7fff の後 0x100 にリセット）

【 返 り 値 】 G729BE_ANA_BUF[5](Y)　圧縮された信号

【 機　能 】 コーデックからの入力信号（80 サンプル×16 ビット）を 80 ビットまたは 16 ビットに圧縮する。

【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

【ハードウエア・リソースメント】

最大スタック・レベル	6
最大ループ･スタック・レベル	3
最大リピート回数	240
最大 MIPS 値	16.5

（4）G.729 ANNEX A＋ANNEX B 用エンコーダ関数

【 分　類 】 エンコード処理部

【 関 数 名 】 g729ab_Enc

【機 能 概 要】 80 サンプル×16 ビットを 80 ビットまたは 16 ビットに圧縮する。

【 形　式 】 call g729ab_Enc

【 引 き 数 】

G729ABE_PCM_BUF[80](X)	入力データ
*G729ABE_CMD_STS:y	コントロール／ステータス・レジスタ
*G729ABframe:y	frame 数のカウンタ（0-0x7fff） （0x7fff の後 0x100 にリセット）

【 返 り 値 】 G729ABE_ANA_BUF[5](Y)　圧縮された信号

【 機　能 】 コーデックからの入力信号（80 サンプル×16 ビット）を 80 ビットまたは 16 ビットに圧縮する。

【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

【ハードウエア・リソースメント】

最大スタック・レベル	6
最大ループ･スタック・レベル	3
最大リピート回数	240
最大 MIPS 値	9.5

2.2.4　デコーダ関数

デコーダ関数は，80 **ビットまたは** 16 **ビットに圧縮された信号を** 80 **サンプル×**16 **ビットの音声データに伸長します。**

（1）G.729 用デコーダ関数

【　分　類　】　**デコード処理部**

【 関 数 名 】　g729_Dec

【機 能 概 要】　80 **ビットを** 80 **サンプル×**16 **ビットに伸長する。**

【　形　式　】　call g729_Dec

【 引 き 数 】　G729D_ANA_BUF[5](Y)　**入力データ**

*G729D_CMD_STS:y　**コントロール／ステータス・レジスタ**

【 返 り 値 】　G729D_PCM_BUF[80](X)　**伸長された信号**

【　機　能　】　80 **ビットに圧縮されたデータを音声信号（**80 **サンプル×**16 **ビット）に伸長する。**

【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

【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	2
最大リピート回数	40
最大 MIPS **値**	3.3

（2）G.729 ANNEX A 用デコーダ関数

【　分　類　】　**デコード処理部**

【 関 数 名 】　g729a_Dec

【機 能 概 要】　80 **ビットを** 80 **サンプル×**16 **ビットに伸長する。**

【　形　式　】　call g729a_Dec

【 引 き 数 】　G729AD_ANA_BUF[5](Y)　**入力データ**

*G729AD_CMD_STS:y　**コントロール／ステータス・レジスタ**

【 返 り 値 】　G729AD_PCM_BUF[80](X)　**伸長された信号**

【　機　能　】　80 **ビットに圧縮されたデータを音声信号（**80 **サンプル×**16 **ビット）に伸長する。**

【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

【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	2
最大リピート回数	40
最大 MIPS **値**	2.2

(3) G.729 ANNEX B 用デコーダ関数

【 分　類 】 デコード処理部

【 関 数 名 】 g729b_Dec

【機 能 概 要】 80 ビットまたは 16 ビットを 80 サンプル×16 ビットに伸長する。

【 形　式 】 call g729b_Dec

【 引 き 数 】 G729BD_ANA_BUF[5](Y)　入力データ

*G729BD_CMD_STS:y　コントロール／ステータス・レジスタ

【 返 り 値 】 G729BD_PCM_BUF[80](X)　伸長された信号

【 機　能 】 80 ビットまたは 16 ビットに圧縮されたデータを音声信号（80 サンプル×16 ビット）に伸長する。

【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

【ハードウエア・リソースメント】

最大スタック・レベル	6
最大ループ･スタック・レベル	3
最大リピート回数	40
最大 MIPS 値	3.5

(4) G.729 ANNEX A＋ANNEX B 用デコーダ関数

【 分　類 】 デコード処理部

【 関 数 名 】 g729ab_Dec

【機 能 概 要】 80 ビットまたは 16 ビットを 80 サンプル×16 ビットに伸長する。

【 形　式 】 call g729ab_Dec

【 引 き 数 】 G729ABD_ANA_BUF[5](Y)　入力データ

*G729ABD_CMD_STS:y　コントロール／ステータス・レジスタ

【 返 り 値 】 G729ABD_PCM_BUF[80](X)　伸長された信号

【 機　能 】 80 ビットまたは 16 ビットに圧縮されたデータを音声信号（80 サンプル×16 ビット）に伸長する。

【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

【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	3
最大リピート回数	40
最大 MIPS 値	3.7

2.2.5　バージョン取得関数

バージョン取得関数は，ライブラリのバージョンを返します。

【　分　類　】　**バージョン情報取得**

【 関 数 名 】　g729_GetVersion

【 機 能 概 要 】　**ライブラリのバージョンを返す。**

【　形　式　】　call g729_GetVersion

【 引 き 数 】　**なし**

【 返 り 値 】　R0H　**メジャー・バージョン番号**

R0L　**マイナ・バージョン番号**

【　機　能　】　G.729 **音声コーデック・ライブラリのバージョン番号を** 32 **ビットの値で返します。**

例　R0 = 0x00’0x0001’0x0100 **の場合**

バージョン：V1.01

【使用レジスタ】　R0

【ハードウエア・リソースメント】

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

★ 2.3　関数仕様（マルチチャネル版）

2.3.1　スクラッチ領域初期化関数

スクラッチ領域初期化関数は，マルチチャネル仕様に対応する本ミドルウエアのスクラッチ領域の初期化を行う関数です。

（1）スクラッチ領域初期化関数

【　分　類　】　スクラッチ領域初期化処理

【 関 数 名 】　g729b_StartCodec

【機 能 概 要】　G.729 ANNEX B マルチチャネル版で使用するスクラッチ領域を 0 クリアする。

【　形　式　】　call G729b_Start_Codec

【 引 き 数 】　なし

【 返 り 値 】　なし

【　機　能　】　G.729 ANNEX B マルチチャネル版で使用するスクラッチ領域を 0 クリアする。

【使用レジスタ】　R0,DP0,DP4,DP5

【ハードウエア・リソースメント】

最大スタック・レベル	1
最大ループ･スタック・レベル	1
最大リピート回数	597
最大サイクル数	1072

2.3.2　エンコーダ初期化関数

g729b_InitEncM 関数は，エンコーダ各定数の設定，係数テーブル，遅延バッファの初期化を行う関数です。

（1）G.729 ANNEX B マルチチャネル版用エンコーダ初期化関数

【　分　類　】　エンコーダ初期化処理

【関 数 名 】　g729b_InitEncM

【機能概要】　G.729 ANNEX B マルチチャネル版エンコーダの初期化および各種パラメータの設定などを行う。

【　形　式　】　call g729b_InitEncM

【引 き 数 】　入出力バッファの先頭アドレスおよび各種パラメータ

例

```
/*---*/         dp0=g729b_IO_Table   ;
clr(r1)                            ;
r0l=STATIC_X1                      ;
/*---*/         *dp0++=r0l           ;
r0l=STATIC_Y1                      ;
/*---*/         *dp0++=r0l           ;
r0l=EncPcmData1                    ;
/*---*/         *dp0++=r0l           ;
r0l=EncPrmData1                    ;
/*---*/         *dp0++=r0l           ;
rep 4                              ;
   /*...*/      *dp0++=r1h           ;
r0l=DecPcmData1                    ;
/*---*/         *dp0++=r0l           ;
r0l=DecPrmData1                    ;
/*---*/         *dp0++=r0l           ;
r0l= 1                             ;
    /*   */     *dp0++=r0l           ; fParityCheck = 1
rep 5                              ;
    /*...*/     *dp0++=r1h           ;
/*   */         *dp0++=r0l           ; fVadEnable = 1
rep 3                              ;
    /*...*/     *dp0++=r1h           ;
/*---*/         dp0=g729b_IO_Table   ;
```

【返り値】　なし

【　機　能　】　G.729 ANNEX B マルチチャネル版エンコーダの初期化，各種パラメータの設定などを行う。

【使用レジスタ】　R0,R1,DP0, DP1,DP2,DP3, DP4,DP5,DP6, DP7,DN0

【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	240
最大サイクル数	2642

2.3.3　デコーダ初期化関数

g729b_InitDncM 関数は，デコーダ各定数の設定，係数テーブル，遅延バッファの初期化を行う関数です。

(1) G.729 ANNEX B マルチチャネル版用デコーダ初期化関数

【 分　類 】 デコーダ初期化処理

【 関 数 名 】 g729b_InitDecM

【機 能 概 要】 G.729 ANNEX B マルチチャネル版デコーダの初期化および各種パラメータの設定などを行う。

【 形　式 】 call g729b_InitDecM

【 引 き 数 】 入出力バッファの先頭アドレスおよび各種パラメータ

例

```
/*---*/          dp0=g729b_IO_Table ;
clr(r1)                           ;
r0l=STATIC_X1                     ;
/*---*/          *dp0++=r0l         ;
r0l=STATIC_Y1                     ;
/*---*/          *dp0++=r0l         ;
r0l=EncPcmData1                   ;
/*---*/          *dp0++=r0l         ;
r0l=EncPrmData1                   ;
/*---*/          *dp0++=r0l         ;
rep 4                             ;
   /*...*/       *dp0++=r1h         ;
r0l=DecPcmData1                   ;
/*---*/          *dp0++=r0l         ;
r0l=DecPrmData1                   ;
/*---*/          *dp0++=r0l         ;
r0l= 1                            ;
   /*   */       *dp0++=r0l         ; fParityCheck = 1
rep 5                             ;
   /*...*/       *dp0++=r1h         ;
/*   */           *dp0++=r0l         ; fVadEnable = 1
rep 3                             ;
   /*...*/       *dp0++=r1h         ;
/*---*/          dp0=g729b_IO_Table ;
```

【 返 り 値 】 なし

【 機　能 】 G.729 ANNEX B マルチチャネル版用デコーダの初期化，各種パラメータの設定などを行う。

【使用レジスタ】 R0, R1, DP0, DP1, DP2, DP3, DP4, DP5, DP6, DN0

【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	234
最大サイクル数	1741

2.3.4　エンコーダ関数

エンコーダ関数は，入力した 80 サンプルの音声信号を 80 ビットまたは 16 ビットに圧縮した信号を生成します。

(1) G.729 ANNEX B マルチチャネル版用エンコーダ関数

【　分　類　】　エンコード処理部

【 関 数 名 】　g729b_EncM

【機 能 概 要】　80 サンプル×16 ビットを 80 ビットまたは 16 ビットに圧縮する。

【　形　式　】　call g729b_EncM

【 引 き 数 】　入出力バッファの先頭アドレスおよび各種パラメータ

例

```
/*---*/          dp0=g729b_IO_Table  ;
clr(r1)                              ;
r0l=STATIC_X1                        ;
/*---*/          *dp0++=r0l          ;
r0l=STATIC_Y1                        ;
/*---*/          *dp0++=r0l          ;
r0l=EncPcmData1                      ;
/*---*/          *dp0++=r0l          ;
r0l=EncPrmData1                      ;
/*---*/          *dp0++=r0l          ;
rep 4                                ;
   /*...*/       *dp0++=r1h          ;
r0l=DecPcmData1                      ;
/*---*/          *dp0++=r0l          ;
r0l=DecPrmData1                      ;
/*---*/          *dp0++=r0l          ;
r0l= 1                               ;
   /*   */       *dp0++=r0l          ; fParityCheck = 1
rep 5                                ;
   /*...*/       *dp0++=r1h          ;
/*   */          *dp0++=r0l          ; fVadEnable = 1
rep 3                                ;
   /*...*/       *dp0++=r1h          ;
/*---*/          dp0=g729b_IO_Table  ;
```

【 返 り 値 】　圧縮データ：引き数で指定した出力バッファ

【　機　能　】　コーデックからの入力信号（80 サンプル×16 ビット）を 80 ビットまたは 16 ビットに圧縮する。

【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

【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	1
最大リピート回数	223
最大 MIPS 値	13.9

2.3.5　デコーダ関数

デコーダ関数は，80 ビットまたは 16 ビットに圧縮された信号を 80×16 ビットの音声データに伸長します。

(1) G.729 ANNEX B マルチチャネル版用デコーダ関数

【　分　類　】　デコード処理部

【 関 数 名 】　g729b_DecM

【機 能 概 要】　80 ビットまたは 16 ビットを 80 サンプル×16 ビットに伸長する。

【　形　式　】　call g729b_DecM

【 引 き 数 】　入出力バッファの先頭アドレスおよび各種パラメータ

例

```
/*---*/          dp0=g729b_IO_Table  ;
clr(r1)                              ;
r0l=STATIC_X1                        ;
/*---*/          *dp0++=r0l          ;
r0l=STATIC_Y1                        ;
/*---*/          *dp0++=r0l          ;
r0l=EncPcmData1                      ;
/*---*/          *dp0++=r0l          ;
r0l=EncPrmData1                      ;
/*---*/          *dp0++=r0l          ;
rep 4                                ;
   /*...*/       *dp0++=r1h          ;
r0l=DecPcmData1                      ;
/*---*/          *dp0++=r0l          ;
r0l=DecPrmData1                      ;
/*---*/          *dp0++=r0l          ;
r0l= 1                               ;
   /*   */       *dp0++=r0l          ; fParityCheck = 1
rep 5                                ;
   /*...*/       *dp0++=r1h          ;
/*   */          *dp0++=r0l          ; fVadEnable = 1
rep 3                                ;
   /*...*/       *dp0++=r1h          ;
/*---*/          dp0=g729b_IO_Table  ;
```

【 返 り 値 】　圧縮データ：引き数で指定した出力バッファ

【　機　能　】　80 ビットまたは 16 ビットを 80 サンプル×16 ビットに伸長する。

【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

【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	203
最大 MIPS 値	2.8

2.4　コントロール／ステータス・レジスタ（シングルチャネル版）

G729E_CMD_STS（G729AE_CMD_STS，G729BE_CMD_STS，G729ABE_CMD_STS）は，G.729 エンコーダ，G729D_CMD_STS（G729AD_CMD_STS，G729BD_CMD_STS，G729ABD_CMD_STS）はデコーダのコントロールおよびステータス・レジスタです。上位 8 ビットはステータス・ワードで読み出しのみ可能です。下位 8 ビットはコントロール・ワードで読み出し／書き込み可能です。

2.4.1　G729E_CMD_STS（G729AE_CMD_STS，G729BE_CMD_STS，G729ABE_CMD_STS）レジスタ

G729E_CMD_STS （G729AE_CMD_STS，G729BE_CMD_STS，G729ABE_CMD_STS）レジスタはエンコーダの動作コントロールおよびステータス表示を行うメモリマップト・レジスタです。変更は，g729_Enc（ g729a_Enc，g729b_Enc，g729ab_Enc ）関数のコール前に行ったものが有効です。

ただし，ビット D9，D8，D5 は ANNEX B 対応のビットのため，G.729 および G.729 ANNEX A を使用する場合，（D9, D8, D5）＝（0, 1, 0）を入力してください。

表 2－1　G729E_CMD_STS レジスタ

ビット	名称	初期値	0	1	備　考
D15	Busy Flag	-	起動されていない	起動中	エンコーダ（g729_Enc/g729a_Enc/g729b_Enc/g729ab_Enc）が起動されている状態か否かを示す。
D14-D10	Reserved	-	-	-	-
D9	Ftyp1	-	注	注	-
D8	Ftyp0	-	注	注	-
D7	Operation	-	処理しない	処理する	エンコード処理を行うか否かを決定する。
D6	Test Mode	-	ノーマル	テスト	ROM 内のテスト・データをエンコーダに入力する。
D5	VAD/DTX	-	処理しない	処理する	無音圧縮機能のオン／オフを指定する。
D4	Muting	-	ノーマル	無音	無音データ（0）をエンコーダに入力する。
D3-D0	Reserved	-	-	-	-

注　Ftyp(1,0) = 00：Untransmitted frame
01：Active Speech frame
10：SID frame
11：N/A

2.4.2　G729D_CMD_STS（G729AD_CMD_STS，G729BD_CMD_STS，G729ABD_CMD_STS）レジスタ

G729D_CMD_STS （G729AD_CMD_STS， G729BD_CMD_STS，G729ABD_CMD_STS）レジスタはデコーダの動作コントロールおよびステータス表示を行うメモリマップト・レジスタです。変更は，g729_Dec（ g729a_Dec，g729b_Dec，g729ab_Dec ）関数のコール前に行ったものが有効です。

ただし，ビット D5，D1，D0 は ANNEX B 対応ビットのため，G.729 および G.729 ANNEX A を使用する場合，（D5，D1，D0）＝（0,0,1）を入力してください。

表 2ー2　G729D_CMD_STS レジスタ

ビット	名称	初期値	0	1	備　考
D15	Busy Flag	-	起動されていない	起動中	デコーダ（g729_Dec/g729a_Dec/g729b_Dec/g729ab_Dec）が起動されている状態か否かを示す。
D14	Parity Error	-	エラーなし	エラーあり	パリティ・チェックの結果を示す。
D13-D8	Reserved	-	-	-	-
D7	Operation	-	処理しない	処理する	デコード処理を行うか否かを決定する。
D6	Test Mode	-	ノーマル	テスト	ROM 内のテスト・データをデコーダに入力する。
D5	DTX/CNG	-	処理しない	処理する	無音圧縮機能のオン／オフを指定する。
D4	Muting	-	ノーマル	無音	無音データ（0）をデコーダから出力する。
D3	Frame Eras.	-	処理しない	処理する	消失フレームに対する処理を行うか否かを決定する。
D2	Parity Check	-	チェックなし	チェックあり	パリティ・チェックを行うか否かを決定する。
D1	Ftyp1	-	注	注	-
D0	Ftyp0	-	注	注	-

注　Ftyp(1,0) = 00：Untransmitted frame
01：Active Speech frame
10：SID frame
11：N/A

★ 2.5　マルチチャネル版外部インタフェース

ここでは，G.729 ANNEX B マルチチャネル版用関数 G729b_EncM，g729_DecM のコール時に引き渡すパラメータを定義します。

2.5.1　G.729 ANNEX B マルチチャネル版用関数の引き渡しパラメータ

表 2－3 にテーブル内容を示します。

表 2－3　G.729 ANNEX B マルチチャネル版の引き渡しパラメータ

Offset	内　　容	アクセス・タイミング		
		初期化	Encode	Decode
0x00	Static 変数領域 x のアドレス（初期化処理以降，変更しないこと）	C	C	C
0x01	Static 変数領域 y のアドレス（初期化処理以降，変更しないこと）	C	C	C
0x02	符号化用 PCM データ入力バッファのアドレス		C	
0x03	符号化用符号データ出力バッファのアドレス		C	
0x04	符号化フレーム・カウント 設定値：最初だけ 0x0001-0x7fff (フレームごとに＋1)，以降は， 0x0100-0x7fff（フレームごとに＋1）を繰り返すこと		C	
0x05-0x07	予備			
0x08	復号化用 PCM データ出力バッファのアドレス			C
0x09	復号化用符号データ入力バッファのアドレス			C
0x0a	パリティ・チェック指定 （0：パリティ・チェックなし，1：パリティ・チェックあり）			C
0x0b	パリティ・エラー （0：パリティ・エラーなし，1：パリティ・エラーあり）			S
0x0c	消失フレーム処理指定（0：通常フレーム，1：消失）			C
0x0d-0x0f	予備			
0x10	符号化用無音フレーム生成指定（0：生成なし，1：生成あり）		C	
0x11	符号化出力フレーム・タイプ		S	
0x12	復号化入力フレーム・タイプ			S
0x13	予備			

注意　スクラッチ領域の確保は次のとおりに行ってください。スクラッチ領域 x には ’align at 4’ が必要です。
スクラッチ領域 x：X メモリ上に“ lib_Scratch_x”というレーベル名で 1153 ワードの大きさ
スクラッチ領域 y：Y メ　モリ上に“ lib_Scratch_y”というレーベル名で 1257 ワードの大きさ

備考　アクセスの種類　C：制御情報
S：結果ステータス

2.6　圧縮データ・フォーマット

2.6.1　有音データの圧縮フォーマット

エンコーダ関数の出力，およびデコーダ関数の入力となる，80 ビット・データのフォーマットを表 2－4， 2－5 に示します。圧縮データに関する情報の詳細については，ITU-T 勧告を参照してください。

表 2－4　有音圧縮データのビット割り当て

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
word0	L0	L1	L1	L1	L1	L1	L1	L1	L2	L2	L2	L2	L2	L3	L3	L3
word1	L3	L3	P1	P1	P1	P1	P1	P1	P1	P1	P0	C1	C1	C1	C1	C1
word2	C1	C1	C1	C1	C1	C1	C1	C1	S1	S1	S1	S1	GA1	GA1	GA1	GB1
word3	GB1	GB1	GB1	P2	P2	P2	P2	P2	C2	C2	C2	C2	C2	C2	C2	C2
word4	C2	C2	C2	C2	C2	S2	S2	S2	S2	GA2	GA2	GA2	GB2	GB2	GB2	GB2

表 2－5　有音圧縮データの意味

シンボル	内　　容	ビット数
L0	Switched MA predictor index of LSP quantizer	1
L1	1st stage vector of LSP quantizer	7
L2	2nd stage lower vector of LSP quantizer	5
L3	2nd stage higher vector of LSP quantizer	5
P1	Pitch delay 1st subframe	8
P0	Parity bit for pitch delay	1
C1	Fixed codebook 1st subframe	13
S1	Signs of fixed-codebook pulses 1st subframe	4
GA1	Gain codebook (stage 1) 1st subframe	3
GB1	Gain codebook (stage 2) 1st subframe	4
P2	Pitch delay 2nd subframe	5
C2	Fixed codebook 2nd subframe	13
S2	Signs of fixed-codebook pulses 2nd subframe	4
GA2	Gain codebook (stage 1) 2nd subframe	3
GB2	Gain codebook (stage 2) 2nd subframe	4

2.6.2　SID フレームの圧縮フォーマット

エンコーダ関数の出力，およびデコーダ関数の入力となる，16 ビット無音データのフォーマットを表 2－6， 2－7 に示します。圧縮データに関する情報の詳細については，ITU-T 勧告を参照してください。

表 2－6　無音圧縮データのビット割り当て

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word	L0	L1	L1	L1	L1	L1	L2	L2	L2	L2	E	E	E	E	E	0

表 2－7　無音圧縮データの意味

シンボル	内　　容	ビット数
L0	Switched predictor index of LSF quantizer	1
L1	1st stage vector of LSF quantizer	5
L2	2nd stage vector of LSF quantizer	4
E	Gain (Energy)	5

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

本ミドルウエアの供給媒体は，3.5 インチ・フロッピィ・ディスク（1.44 MB）です。ホスト・マシンへのインストールの手順を次に示します。

（1）供給媒体をフロッピィ・ディスク・ドライブにセットします。ソフトウエア・ツールが使用しているディレクトリ（例：C:¥DSPTools）の下にファイルをコピーします。ここでは，A ドライブから C ドライブへコピーした場合を示します。

```
a:¥>xcopy /s *.* c:¥DSPTools <CR>
```

（2）ファイルがコピーされたことを確認します。各ディレクトリについては，**1.5.4　ディレクトリ構成**を参照してください。

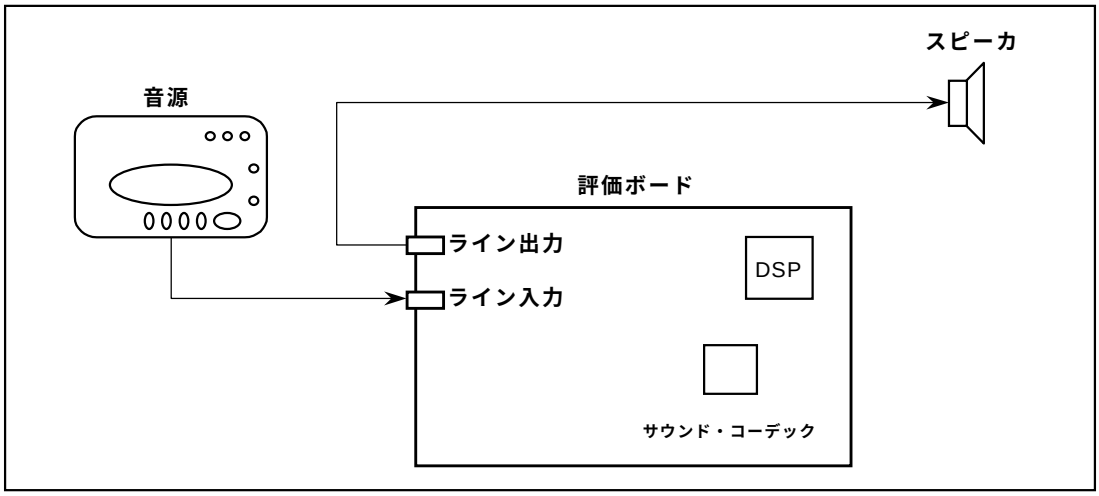
```
a:¥>dir c:¥DSPTools <CR>
```

3.2　サンプル作成手順

sample のディレクトリ（smp）に，サンプル・プログラムを格納しています（sample.asm のソース・プログラムについては，付録 サンプル・ソース参照）。

サンプル・プログラムは，実際に CD や DAT（Digital Audio Tape）などの音源，スピーカなどを接続して，リアルタイムで音源の圧縮伸長を行うことができます。なお，サンプル・プログラムはμPD77016 音声コーデック評価ボード上で動作します。

図 3－1　サンプル・プログラム評価システム



★　注意　ただし，G.729 ANNEX B マルチチャネル版用のサンプル・プログラムは，この評価システム上では動作しません。

次に本ミドルウエアのサンプル・プログラムのビルド方法について示します。

（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 メニューの Open Project コマンドで 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 メニューの Build All コマンドを選択すると，sample.lnk が生成されます。

（4）IE77016（ディバッガ）を使用して，ターゲットにダウンロードして実行します。

3.3　シンボル名規約

本ライブラリで使用するシンボル名は，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３－1 シンボル名**

種　別	G.729	ANNEX A	ANNEX B	ANNEX AB 注	ANNEX B
	シングル・チャネル				マルチチャネル
関数名	g729_xxxx	g729a_xxxx	g729b_xxxx	g729ab_xxxx	g729b_xxxx
マクロ，定数名	G729xxxx	G729Axxxx	G729Bxxxx	G729ABxxxx	g729bxxxx
セクション名	__G729_xxxx	__G729A_xxxx	__G729B_xxxx	__G729AB_xxxx	__G729B_xxxx

注　ANNEX AB：ANNEX＋ANNEX B

［メ　モ］

付録　サンプル・ソース

付.1　シングルチャネル版用サンプル・ソース（sample.asm）

次に示すシングルチャネル版用サンプル・ソースは，samplek.asm（G.729 用のサンプル・ソース）です。samplea.asm（G.729 ANNEX A 用のサンプル・ソース）では，G729 の部分が G729A，g729_の部分が g729a_になります。sampleb.asm（G.729 ANNEX B 用のサンプル・ソース）では，G729 の部分が G729B，g729_の部分が g729b_になります。sampleab.asm（G.729 ANNEX A＋ANNEXB 用のサンプル・ソース）では，G729 の部分が G729AB，g729_の部分が g729ab_になります。

また，sampleb.asm，sampleab.asm では，フレーム数カウンタ（G729Bframe，G729ABframe）に関する部分が追加されます。

```
;; ************************************************************************
;;  CS-CELP[G.729] CODEC control program Ver1.0
;; ************************************************************************

#DEFINE    MODEB_TYPE
#IFDEF     MODEB_TYPE
#DEFINE    MODE_BIT_SET
#ENDIF

#include "g729_lib.h"
#include "upd7701x.h"

#IF CHIPTYPE == TYPE77017
#DEFINE IN_OR_EXT
#ELSE
#DEFINE IN_OR_EXT   EXTERNAL
#ENDIF

;; ************************************************************************
;;  G729 Encoder, Decoder processing core Parts
;; ************************************************************************
EXTRN   g729_Enc                ;; G729 Encoder Main
EXTRN   g729_InitEnc            ;; G729 Encoder Initialize
EXTRN   g729_Dec                ;; G729 Decoder Main
EXTRN   g729_InitDec            ;; G729 Decoder Initialize

EXTRN   G729E_ANA_BUF           ;; Encoder Output Buffer
EXTRN   G729D_ANA_BUF           ;; Decoder Input Buffer
EXTRN   G729E_PCM_BUF           ;; Serial Input Buffer
EXTRN   G729D_PCM_BUF           ;; Serial Output Buffer
EXTRN   G729E_CMD_STS           ;; Encoder status
EXTRN   G729D_CMD_STS           ;; Decoder Status
EXTRN   G729ana                 ;; Encoder internal output
;; ************************************************************************

;; ======================================
;;  X/Y Data ROM Table (NEC)
;; ======================================
__G729_COMROMY YROMSEG IN_OR_EXT
;; --------------------------------------
;;  Encoder Mute Data (for ana[])
;; --------------------------------------
```

注 1. sampleb.asm（sampleab.asm）**では，次のようになります。ただし，**sampleab.asm **では** G729B **の部分が** G729AB **になります。**

```
    EXTRN   G729Bana        ;; Encoder internal output
    EXTRN   G729Bframe      ;;
;; ************************************************************
```

```
E_MUTE_ANA:                     ;;
    DW  0b01111000              ;; ana[0]
    DW  0b0011111010            ;;    1
    DW  0b10111101              ;;    2
    DW  0b0                     ;;    3
    DW  0b0000000000000         ;;    4
    DW  0b1111                  ;;    5
    DW  0b1010110               ;;    6
    DW  0b11011                 ;;    7
    DW  0b0000000000000         ;;    8
    DW  0b1111                  ;;    9
    DW  0b1010110               ;;   10

;; ============================================
;;  Register Store Area
;; ============================================

__G729_X_REG_SAVE_SEG  XRAMSEG

SaveRegAreaX:
__R0_SAVE:      DS  3      ;; SaveRegAreaX+0x0000
__R1_SAVE:      DS  3      ;; SaveRegAreaX+0x0003
__R2_SAVE:      DS  3      ;; SaveRegAreaX+0x0003
__R3_SAVE:      DS  3      ;; SaveRegAreaX+0x0003
__R4_SAVE:      DS  3      ;; SaveRegAreaX+0x0003
__R5_SAVE:      DS  3      ;; SaveRegAreaX+0x0003
__R6_SAVE:      DS  3      ;; SaveRegAreaX+0x0003
__R7_SAVE:      DS  3      ;; SaveRegAreaX+0x0003
__DP0_SAVE:     DS  1      ;; SaveRegAreaX+0x0006
__DP1_SAVE:     DS  1      ;; SaveRegAreaX+0x0006
__DP2_SAVE:     DS  1      ;; SaveRegAreaX+0x0006
__DP3_SAVE:     DS  1      ;; SaveRegAreaX+0x0006
__DP4_SAVE:     DS  1      ;; SaveRegAreaX+0x0007
__DP5_SAVE:     DS  1      ;; SaveRegAreaX+0x0006
__DP6_SAVE:     DS  1      ;; SaveRegAreaX+0x0006
__DP7_SAVE:     DS  1      ;; SaveRegAreaX+0x001f

;; ======================================
;;  SI1 Save/Recover registers MACRO
;; ======================================

%DEFINE (G729SAVE_REG)(
    *SaveRegAreaX+0x0000:X = R0L       ;;
    *SaveRegAreaX+0x0001:X = R0H       ;;
    *SaveRegAreaX+0x0002:X = R0E       ;;

    *SaveRegAreaX+0x0003:X = R1L       ;;
    *SaveRegAreaX+0x0004:X = R1H       ;;
    *SaveRegAreaX+0x0005:X = R1E       ;;

    R0L = DP0                          ;;
    *SaveRegAreaX+0x00018:X = R0L      ;;
    R0L = DP1                          ;;
    *SaveRegAreaX+0x00019:X = R0L      ;;
)

%DEFINE (G729RECOVER_REG)(
    R0L = *SaveRegAreaX+0x0019:X       ;;
    DP1 = R0L                          ;;
    R0L = *SaveRegAreaX+0x0018:X       ;;
```

```
    DP0 = R0L                          ;;

    R1E = *SaveRegAreaX+0x0005:X       ;;
    R1H = *SaveRegAreaX+0x0004:X       ;;
    R1L = *SaveRegAreaX+0x0003:X       ;;

    R0E = *SaveRegAreaX+0x0002:X       ;;
    R0H = *SaveRegAreaX+0x0001:X       ;;
    R0L = *SaveRegAreaX+0x0000:X       ;;
)

;; ============================================
;; ENCODER SYMBOL
;; ============================================


__G729_ENC_DSEG_Y  YRAMSEG

E_SI1_CNT:     DS  1          ;; E_SI1_BUF index
E_FRAME_CNT:   DS  1          ;; Frame Number
E_SKIP_FLAG:   DS  1          ;; Skip flag for Encoder Startup
E_NEW_DATA:    DS  1          ;; New Data input flag

__G729_ENC_DSEG_X  XRAMSEG

E_SI1_BUF:     DS  80         ;; Serial input buffer  (Speech Signal)

;; ---------------------------------------
;;  DECODER SYMBOL
;; ---------------------------------------


__G729_DEC_DSEG_Y  YRAMSEG
D_SO1_CNT:     DS  1          ;; D_SO1_BUF index
D_NEW_DATA:    DS  1          ;; Input New Data flag
D_SKIP_FLAG:   DS  1          ;; Skip Flag for Startup Decoder
D_MUTE_DATA:   DS  1          ;; Mute Data

__G729_DEC_DSEG_X  XRAMSEG

D_SO1_BUF:     DS  80         ;; Output CODEC Frame Buffer

;; ***************************************************************************
;;  G729 TEXT SEGMENT
;; ***************************************************************************

SPX_START   IMSEG   at 0x200
;; ---------------------------------------
;;  SPX Interrupt Vector Code
;; ---------------------------------------
Reset_Entry:   ;;
    JMP Main   ;; Jump to Main
    NOP        ;;
    NOP        ;;
    NOP        ;;
    ;
    ; Reserved_Vector_Area [0x0204 - 0x020F]
    ;
    NOP        ;;
    NOP        ;;
```

```
    NOP         ;;
    NOP         ;;
    NOP         ;;
    NOP         ;;
    NOP         ;;
    NOP         ;;
    NOP         ;;
    NOP         ;;
    NOP         ;;
    NOP         ;;
INT1_Entry:             ;; -----------------------------------
    JMP FATAL_INT       ;; [INT1:External]
    NOP         ;;
    NOP         ;;
    NOP         ;;
INT2_Entry:             ;; -----------------------------------
    JMP FATAL_INT       ;; [INT2:External]
    NOP         ;;
    NOP         ;;
    NOP         ;;
INT3_Entry:             ;; -----------------------------------
    JMP FATAL_INT       ;; [INT3:External]
    NOP         ;;
    NOP         ;;
    NOP         ;;
INT4_Entry:             ;; -----------------------------------
    JMP FATAL_INT       ;; [INT4:External]
    NOP         ;;
    NOP         ;;
    NOP         ;;
SI1_Entry:              ;; -----------------------------------
    JMP Enc_SI1_Handler;; [SI1:Internal]
    NOP         ;;
    NOP         ;;
    NOP         ;;
SO1_Entry:              ;; -----------------------------------
    JMP FATAL_INT       ;; [SO1:Internal]
    NOP         ;;
    NOP         ;;
    NOP         ;;
SI2_Entry:              ;; -----------------------------------
    JMP FATAL_INT       ;; [SI2:Internal]
    NOP         ;;
    NOP         ;;
    NOP         ;;
SO2_Entry:              ;; -----------------------------------
    JMP FATAL_INT       ;; [SO2:Internal]
    NOP         ;;
    NOP         ;;
    NOP         ;;
HI_Entry:               ;; -----------------------------------
    JMP FATAL_INT       ;; [HI:Internal]
    NOP         ;;
    NOP         ;;
    NOP         ;;
HO_Entry:               ;; -----------------------------------
    JMP FATAL_INT       ;; [HO:Internal]
    NOP         ;;
    NOP         ;;
    NOP         ;;
```

```
    ;
    ; Reserved_Vector_Area [0x0238 - 0x023F]
    ;
    NOP             ;;
    NOP             ;;
    NOP             ;;
    NOP             ;;
    NOP             ;;
    NOP             ;;
    NOP             ;;
    NOP             ;;

;; ***************************************************************************
;;
;;

NECMAIN IMSEG at 0x240
Main:
    ;; ------------------------------
    ;; Initialize SPX
    ;; ------------------------------
    R0L = 0xF3FF        ;; Disable All Interrupt
    SR  = R0L           ;;
    CLR(R0)             ;; Clear General Registers
    CLR(R1)             ;;
    CLR(R2)             ;;
    CLR(R3)             ;;
    CLR(R4)             ;;
    CLR(R5)             ;;
    CLR(R6)             ;;
    CLR(R7)             ;;
    DP0 = R0L           ;;
    DN0 = R0L           ;;
    DP1 = R0L           ;;
    DN1 = R0L           ;;
    DP2 = R0L           ;;
    DN2 = R0L           ;;
    DP3 = R0L           ;;
    DN3 = R0L           ;;
    DP4 = R0L           ;;
    DN4 = R0L           ;;
    DP5 = R0L           ;;
    DN5 = R0L           ;;
    DP6 = R0L           ;;
    DN6 = R0L           ;;
    DP7 = R0L           ;;
    DN7 = R0L           ;;
    DMX = R0L           ;;
    DMY = R0L           ;;
```

```
    *SST1:X = R0L       ;; SI1/SO1 MODE <= 16bit/MSB first
    *SST2:X = R0L       ;; SI2/SO2 MODE <= 16bit/MSB first
    *DWTR:X = R0L       ;; All Data Memory is NO WAIT
    *IWTR:X = R0L       ;; All Inst Memory is NO WAIT
    R1L = 0x300         ;;
    *HST:X  = R1L       ;; Disable HDT In/Out                          } 注2

    ;; -------------------------------
    ;; Clear Memory[X/Y:0x0000-0x07FF]
    ;; -------------------------------
    DP0 = R0L                          ;; X Memory Pointer <= 0x0000
    DP4 = R0L                          ;; Y Memory Pointer <= 0x0000

    REP 0x800                          ;; -- loop [clear] --
           *DP0++=R0H  *DP4++=R0H      ;; Clear X/Y
                                       ;; -- loop end -------------------
    CLR(R0)                            ;;
    R3L = 0x55FF                       ;; R3L = A-law Mute Data
    *D_MUTE_DATA:y = R3L               ;; D_MUTE_DATA = R3L

;; *************************************************************************
;;
;; *************************************************************************
MAIN2:

_ENC_INI:
    ;; -------------------------------
    ;;  Encoder Initialize
    ;; -------------------------------

     CALL    Initialize_Encoder ;;

    ;; -------------------------------
    ;;  SST1[BIT-9,8] <- [1,0]
    ;; -------------------------------
    R1L = 0x200                ;;
    *SST1:X = R1L              ;;

_DEC_INI:
    ;; -------------------------------
    ;;  Decoder Initialize
    ;; -------------------------------

     CALL    Initialize_Decoder ;;

    ;; -------------------------------
    ;;  SO1 Dummy W
    ;; -------------------------------
    R1L = *D_MUTE_DATA:y       ;; Read D_MUTE_DATA
    *SDT1:X = R1L              ;; Dummy Write to SO1
```

注2. sampleb.asm（sampleab.asm）では，次のようになります。ただし，sampleab.asm では G729B の部分が G729AB になります。

```
    *HST:X  = R1L        ;; Disable HDT In/Out
    *G729Bframe:y = R0L  ;;

    ;; -------------------------------
```

```
    ;; ------------------------------
    ;;  Enable Interrupt
    ;; ------------------------------
    R0L = SR             ;; READ Status Register

    R0 = R0 & 0x7FEF     ;; Open Current Int Mask
    SR = R0L             ;;
    nop                  ;;

;; *************************************************************************
;;  G729 Encoder/Decoder Main Loop
;; *************************************************************************

MainLoop:
    ;; ------------------------------
    ;;  Check Mode
    ;; ------------------------------
    CLR(R1)                     ;;

    ;; ################################
    ;;   Normal Process
    ;; ################################

    ;; ------------------------------
    ;;  Check E_SKIP_FLAG
    ;; ------------------------------
    R0 = *E_SKIP_FLAG:y                 ;; IF E_SKIP_FLAG == ON then SKIP
    IF (R0 == 0) JMP chk_E_NEW_DATA     ;;

    ;; ------------------------------
    ;;  Encoder init
    ;; ------------------------------
Enc_Skip:                       ;;
    Call g729_InitEnc           ;;
    DP4 = E_MUTE_ANA            ;;
    DP0 = G729ana               ;;
       R3=*DP4++                ;;
    Rep 10                      ;;
       R3=*DP4++   *DP0++=R3H   ;;
              *DP0++=R3H        ;;

    ;; ------------------------------
    ;;  Set E_SKIP_FLAG
    ;; ------------------------------
    CLR(R0)                     ;;
    *E_SKIP_FLAG:y = R0L        ;; Clear E_SKIP_FLAG

    JMP EndEncStage             ;;

    ;; ------------------------------
    ;;  Check E_NEW_DATA
    ;; ------------------------------
chk_E_NEW_DATA:                 ;;
    R0 = *E_NEW_DATA:y          ;;
    IF (R0 == 0) JMP EndEncStage ;;
```

```
    ;; -------------------------------
    ;;  Run Encoder
    ;; -------------------------------                                  } 注3
    CALL    g729_Enc                ;; Do Encoder

    CLR ( R0 )                      ;;
    *E_NEW_DATA:y = R0L             ;;
    R0L = 0x1                       ;;
    *D_NEW_DATA:y = R0L             ;;

EndEncStage:                        ;;

    DP4 = G729E_ANA_BUF             ;;
    DP5 = G729D_ANA_BUF             ;;
    loop 5 {                        ;;
        R2 = *DP4++                 ;;
        *DP5++=R2H                  ;;
    }                               ;;


    ;; -------------------------------
    ;;  Check D_SKIP_FLAG
    ;; -------------------------------
    R0 = *D_SKIP_FLAG:y             ;; IF D_SKIP_FLAG == ON then SKIP;
    IF (R0 == 0) JMP chk_D_NEW_DATA;;

    ;; -------------------------------
    ;;  Decoder Init
    ;; -------------------------------
Dec_Skip:                           ;;

    Call    g729_InitDec            ;;
    R0 = *D_MUTE_DATA:y             ;;
    DP1 = G729D_PCM_BUF             ;;
    Rep 80                          ;;
        *DP1++=R0H                  ;;
    jmp EndDecStage                 ;;
```

注3. sampleb.asm（sampleab.asm）**では，次のようになります。ただし，**sampleab.asm **では** G729B **の部分が** G729AB**，**g729B_**の部分が** g729ab_**になります。**

```
    ;; -------------------------------
    CLR(R0)                         ;;
        R0L = *G729Bframe:y         ;;
        R0 = R0 + 1                 ;;
        R1 = R0 - 0x7fff            ;;
        IF(R1 < 0)JMP over_frm_cnt ;;
        R0L = 0x100                 ;;
ovr_frm_cnt:
        *G729Bframe:y = R0L         ;;

    CALL    g729B_Enc               ;; Do Encoder
```

```
    ;; ------------------------------
    ;;  Check D_NEW_DATA
    ;; ------------------------------
chk_D_NEW_DATA:                 ;;
    R0 = *D_NEW_DATA:y          ;;
    IF (R0 == 0) JMP EndDecStage ;;

    ;; ------------------------------
    ;;  Run Decoder
    ;; ------------------------------
    CALL    g729_Dec            ;; Do Decoder
    CLR ( R0 )                  ;;
    *D_NEW_DATA:y = R0L         ;;

EndDecStage:

    ;; ------------------------------
    ;;  Set D_SKIP_FLAG
    ;; ------------------------------
    CLR(R0)                     ;; Clear D_SKIP_FLAG;
    *D_SKIP_FLAG:y = R0L        ;;

    JMP MainLoop                ;; Infinite Loop


;; ************************************************
;; **** G729 Encoder/Decoder Subroutine Segment ****
;; ************************************************
SUBR_SEG    IMSEG
;; --------------------------------------
;;  Encoder Initialize
;; --------------------------------------
Initialize_Encoder:
                                ;; note! R0,R1,DP0,DP4 ONLY
    R0L = 1                     ;;
    *E_SKIP_FLAG:y = R0L        ;; E_SKIP_FLAG = 1

    R0L = 0x0080                ;;
    *G729E_CMD_STS:y = R0L      ;;
    CLR(R0)                     ;;
    *E_NEW_DATA:y  = R0L        ;;

    R0L = 3                     ;;
    *E_FRAME_CNT:y = R0L        ;; E_FRAME_CNT = 3

    R0L = 0                     ;;
    *D_SO1_CNT:y   = R0L        ;; D_SO1_CNT   = 0

    R0L = 0                     ;;
    *E_SI1_CNT:y   = R0L        ;; E_SI1_CNT   = 0

    RET                         ;;

;; --------------------------------------
;;  Decoder Initialize
;; --------------------------------------

Initialize_Decoder:
                                ;; note! R0,R1,DP0,DP4 ONLY
    R0L = 0x0085                ;;
```

```
    *G729D_CMD_STS:y = R0L       ;;
    CLR(R0)                      ;;
    *D_NEW_DATA:y  = R0L         ;; D_NEW_DATA  = 0
    R0L = 1                      ;;
    *D_SKIP_FLAG:y = R0L         ;; D_SKIP_FLAG = 1

    R0 = *D_MUTE_DATA:y          ;;
    DP1 = G729D_PCM_BUF          ;;
    Rep 80                       ;;
        *DP1++=R0H               ;;

    RET                          ;;

;; ***************************************************************************
;;  Serial Import CH-1 Interrupt Handler Segment (Encoder)
;; ***************************************************************************
EncSI1SEG   IMSEG   at  0x4000
;; --------------------------------------
;;  Encoder Serial-In Interrupt Flow
;; --------------------------------------
Enc_SI1_Handler:
    ;; ------------------------------
    ;;  Save Registers
    ;;    R0/R1/DP0/DP4
    ;; ------------------------------
    %G729SAVE_REG                ;;

    ;; ------------------------------
    ;;  Check D_SO1_CNT
    ;; ------------------------------
    clr(R0)                      ;;
    R0L = *D_SO1_CNT:y           ;;

    R0 = R0 - 80                 ;; 0 <= D_SO1_CNT <= 79
    if(R0 < 0 ) jmp skip_D_SO1_RST   ;;
    clr( R0)                     ;;
    *D_SO1_CNT:y = R0L           ;;

skip_D_SO1_RST:

    ;; ------------------------------
    ;;  Check E_SI1_CNT
    ;; ------------------------------
    clr(r1)                      ;;
    R1L = *E_SI1_CNT:y           ;;
    R1 = R1 - 80                 ;;
    if(R1 < 0 )  jmp not_full_SMPL   ;;

    *E_SI1_CNT:y = R1L           ;; E_SI1_CNT=0

    ;; ------------------------------
    ;;  E_SI1_BUF -> E_PCM_BUF
    ;; ------------------------------
    DP1 = E_SI1_BUF              ;;
    R1L = G729E_PCM_BUF          ;;
    DP0 = R1L                    ;;

    Loop 80 {                    ;;
       R1=*DP1++                 ;;
       *DP0++=R1H                ;;
```

```
    }                                ;;

    ;; -----------------------------;;
    ;;  Output Data Copy
    ;; -----------------------------;;
    DP0 = G729D_PCM_BUF              ;;
    DP1 = D_SO1_BUF                  ;;
    Loop 80 {                        ;;
            R0 = *DP0++              ;;
            *DP1++ = R0H             ;;
    }                                ;;

    ;; -------------------------------
    ;;  E_NEW_DATA <- 1
    ;; -------------------------------
    R1L = 1                          ;;
    *E_NEW_DATA:y = R1L              ;;


not_full_SMPL:
    ;; -------------------------------
    ;;  Input Serial Port
    ;; -------------------------------
    R0L = *E_SI1_CNT:y               ;;
    R1 = R0 + E_SI1_BUF              ;;
    DP0 = R1L                        ;;
    R1L = *SDT1:X                    ;;
    *DP0 = R1L                       ;;

    ;; -----------------------------;;
    ;;  Output Serial Port
    ;; -----------------------------;;

    R0L = *D_SO1_CNT:y             ;;

    R0 = R0 + D_SO1_BUF            ;;
    DP0 = R0L                      ;;

    nop                            ;;
    R0L = *DP0                     ;;

    *SDT1:X = R0L                  ;;

    ;; -------------------------------
    ;;  Increment E_SI1_CNT
    ;; -------------------------------
    R0L = *E_SI1_CNT:y             ;;
    R0 = R0 + 1                    ;;
    *E_SI1_CNT:y = R0L             ;;

    ;; -------------------------------
    ;;  D_SO1_CNT:Update
    ;; -------------------------------
    clr(R0)                        ;;
    R0L = *D_SO1_CNT:y             ;;
    R0 = R0 + 1                    ;;
    *D_SO1_CNT:y = R0L             ;;

    ;; -------------------------------
    ;;  Recover Registers
```

```
    ;; ------------------------------
    %G729RECOVER_REG              ;;
    RETI                          ;;


;; ============================================================================
;;  Fatal Error!!
;;  Disable All Interrupt & infinite loop
;;  without reset Watch Dog Timer.
;; ============================================================================
%DEFINE (FATAL_ERROR(Label))
(
Label:  R7L = 0xFFFF                    ;; Disable Interrupt
    SR  = R7L                           ;;
    JMP $-2                             ;; Infinite Loop
)

;;/************************************************/


%FATAL_ERROR(ERR_SO2_CNT)               ;;
%FATAL_ERROR(ERR_SI1_CNT)               ;;
%FATAL_ERROR(ERR_E_BUSY)                ;;

%FATAL_ERROR(ERR_D_BUSY)                ;;

%FATAL_ERROR(ERR_E_COD_CNT)             ;;
%FATAL_ERROR(ERR_D_COD_CNT)             ;;
%FATAL_ERROR(ERR_E_EXT_CNT)             ;;
%FATAL_ERROR(ERR_D_EXT_CNT)             ;;
%FATAL_ERROR(ERR_MEM)                   ;;
%FATAL_ERROR(ERR_MODE)                  ;;
%FATAL_ERROR(ERR_MODE2)                 ;;
%FATAL_ERROR(ERR_HDT1)                  ;;
%FATAL_ERROR(ERR_HDT2)                  ;;

%FATAL_ERROR(ERR_E_FRAME_CNT)           ;;
%FATAL_ERROR(ERR_E_SI1_CNT)             ;;

;; ============================================================================
;;  Illegal interruption handling routine. (DI HALT)
;; ============================================================================
FATAL_INT:                              ;;
    R0L = 0xFFFF                        ;; Disable interrupt
    SR = R0L                            ;;
FATAL_INT_LOOP:                         ;;
    HALT                                ;; Halt loop
    JMP FATAL_INT_LOOP                  ;;

    END
;; ----------------------- end of file ---------------------------------------
```

付.2　マルチチャネル版用サンプル・ソース（samplebm.asm）

このサンプル・ソースは，ホストインタフェースを使った 4 チャネルのデータの送受信を行うものです。送られてきたデータのチャネル・ナンバとエンコーダ（6）とデコーダ（7）の実行コマンドが必要です。この実行コマンドについては、付.2.1　samplebm.asm 用ヘッダ・ファイル（sysconf.h）を参照してください。

```
;; ***************************************************************************
;;  CS-ACELP[G.729 multi-channel] CODEC control program for 4-channel Ver3.0
;; ***************************************************************************

#include "g729blib.h"
#include "sysconf.h"              /* Application definition file */
#include "upd7701x.h"             /* SPX definition file */
public     lib_Scratch_x, lib_Scratch_y
#define L_FRAME    80                    ; Frame size.
#define PRM_SIZE    5                    ; Size of vector of analysis parameters.
#define UPD77116
extrn   g729b_StartCodec

/*---- Memory allocation-----------*/
__G729B_SCRATCHX    xramseg    align at 4
lib_Scratch_x:
             ds     G729B_SCRATCH_X_BUFSIZE

__G729B_SCRATCHY    yramseg
lib_Scratch_y:
             ds     G729B_SCRATCH_Y_BUFSIZE

USER_RAMX      xramseg    Internal
/*-- channel 1 --*/
g729b_IO_Table1:
             ds     G729B_IOTABLE_SIZE
STATIC_X1:   ds     G729B_STATIC_X_BUFSIZE

EncPcmData1: ds     G729B_ENCODE_PCM_BUFSIZE
EncPrmData1: ds     G729B_ENCODE_PRM_BUFSIZE
DecPcmData1: ds     G729B_DECODE_PCM_BUFSIZE           ;
DecPrmData1: ds     G729B_DECODE_PRM_BUFSIZE           ;

/*-- channel 2 --*/
g729b_IO_Table2:
             ds     G729B_IOTABLE_SIZE
STATIC_X2:   ds     G729B_STATIC_X_BUFSIZE

EncPcmData2: ds     G729B_ENCODE_PCM_BUFSIZE
EncPrmData2: ds     G729B_ENCODE_PRM_BUFSIZE
DecPcmData2: ds     G729B_DECODE_PCM_BUFSIZE           ;
DecPrmData2: ds     G729B_DECODE_PRM_BUFSIZE           ;

/*-- channel 3 --*/
g729b_IO_Table3:
             ds     G729B_IOTABLE_SIZE
STATIC_X3:   ds     G729B_STATIC_X_BUFSIZE

EncPcmData3: ds     G729B_ENCODE_PCM_BUFSIZE
EncPrmData3: ds     G729B_ENCODE_PRM_BUFSIZE
DecPcmData3: ds     G729B_DECODE_PCM_BUFSIZE           ;
DecPrmData3: ds     G729B_DECODE_PRM_BUFSIZE           ;
```

```
/*-- channel 4 --*/                              ;
g729b_IO_Table4:
               ds     G729B_IOTABLE_SIZE
STATIC_X4:     ds     G729B_STATIC_X_BUFSIZE

EncPcmData4:   ds     G729B_ENCODE_PCM_BUFSIZE
EncPrmData4:   ds     G729B_ENCODE_PRM_BUFSIZE
DecPcmData4:   ds     G729B_DECODE_PCM_BUFSIZE
DecPrmData4:   ds     G729B_DECODE_PRM_BUFSIZE


USER_RAMY     yramseg
/*-- channel 1 --*/
STATIC_Y1:     ds     G729B_STATIC_Y_BUFSIZE

/*-- channel 2 --*/
STATIC_Y2:     ds     G729B_STATIC_Y_BUFSIZE

/*-- channel 3 --*/
STATIC_Y3:     ds     G729B_STATIC_Y_BUFSIZE

/*-- channel 4 --*/
STATIC_Y4:     ds     G729B_STATIC_Y_BUFSIZE


%define(MakeIOTable(ch))
(
   /*---*/             dp0=g729b_IO_Table@ch@        ;
   clr(r1)                                           ;
   r0l=STATIC_X@ch@                                  ;
   /*---*/             *dp0++=r0l                    ;
   r0l=STATIC_Y@ch@                                  ;
   /*---*/             *dp0++=r0l                    ;
   r0l=EncPcmData@ch@                                ;
   /*---*/             *dp0++=r0l                    ;
   r0l=EncPrmData@ch@                                ;
   /*---*/             *dp0++=r0l                    ;
   rep 4                                             ;
      /*...*/          *dp0++=r1h                    ;
   r0l=DecPcmData@ch@                                ;
   /*---*/             *dp0++=r0l                    ;
   r0l=DecPrmData@ch@                                ;
   /*---*/             *dp0++=r0l                    ;
   r0l= 1                                            ;
      /*   */          *dp0++=r0l                    ; fParityCheck = 1
   rep 5                                             ;
      /*...*/          *dp0++=r1h                    ;
   /*   */             *dp0++=r0l                    ; fVadEnable = 1
   rep 3                                             ;
      /*...*/          *dp0++=r1h                    ;
)

USER_TEXT_SEG    imseg
g729b_MkIO_Table:
      %MakeIOTable(1)                                ;
      %MakeIOTable(2)                                ;
      %MakeIOTable(3)                                ;
      %MakeIOTable(4)                                ;
      ret                                            ;
```

```
/*---- Symbol definition----*/
%define ( DefVectNop ) (        /* Macro for vector definition (no processing) */
    reti;
    nop;
    nop;
    nop;
    )
%define ( DefVectJmp(iproc) ) ( /* Macro for vector definition (jmp iproc) */
    jmp  iproc;
    nop;
    nop;
    nop;
    )

/*---- Functions entry----*/
public  ClrIntMask             /* Interrupt mask change clear */
public  SetIntMask             /* Interrupt mask change set */
public  ErrorDump              /* Error data display  */

$EJECT
/**/
/*************************************************************************/
/* Make interrupt vector settings                                        */
/*************************************************************************/
Sys_V  imseg at 0x0200
    %DefVectJmp(StartUp)               /*- Reset --------------------------*/
    %DefVectNop                        /*- Reserved 1----------------------*/
    %DefVectNop                        /*- Reserved 2----------------------*/
    %DefVectNop                        /*- Reserved 3----------------------*/
    %DefVectJmp(Codec1CK)              /*- INT1  --------------------------*/
    %DefVectJmp(Codec1FM)              /*- INT2  --------------------------*/
    %DefVectJmp(Codec2CK)              /*- INT3  --------------------------*/
    %DefVectJmp(HostIntProc)           /*- INT4  --------------------------*/
    %DefVectJmp(SI1Proc)               /*- SI1   --------------------------*/
    %DefVectJmp(SO1Proc)               /*- SO1   --------------------------*/
    %DefVectJmp(SI2Proc)               /*- SI2   --------------------------*/
    %DefVectJmp(SO2Proc)               /*- SO2   --------------------------*/
    %DefVectJmp(HostIn)                /*- HI    --------------------------*/
    %DefVectJmp(HostOut)               /*- HO    --------------------------*/
    %DefVectNop                        /*- Reserved 4----------------------*/
    %DefVectNop                        /*- Reserved 5----------------------*/

/*************************************************************************/
/* Register save are for vectored interrupt processing                   */
/*************************************************************************/
Sys_XE xramseg
HipSave:
    ds  4;

/*************************************************************************/
/* Vectored interrupt processing according to PIO value                  */
/*************************************************************************/
Sys_II1 imseg
HostIntProc:
    *HipSave+0:x = r0e;                /* Register save */
    *HipSave+1:x = r0h;
    *HipSave+2:x = r0l;
    r0l= dp0;
    *HipSave+3:x = r0l;
```

```
   r0l= *PDT:x;                    /* PIO value registration */
   r0 = r0 & MAX_HIP_NO;           /* Execution of registered interrupt processing */
   r0 = r0 + HipJumpTbl;
   /*   */           dp0 = r0l;
   call dp0;

   r0l= *HipSave+3:x;              /* Register restore */
   /*   */           dp0 = r0l;
   r0l= *HipSave+2:x;
   r0h= *HipSave+1:x;
   r0e= *HipSave+0:x;
   reti;

  HipJumpTbl:
    jmp  StartErrDump;             /* For error processing */
    jmp  HintProc1;                 /* jump to registered interrupt processing */
    jmp  HintProc2;
    jmp  HintProc3;
    jmp  HintProc4;
    jmp  HintProc5;
    jmp  HintProc6;
    jmp  HintProc7;

$EJECT
/**/
/*************************************************************************/
/* Perform initialization processing, wait for command                   */
/*************************************************************************/
Sys_II2 imseg
StartUp:
   call InitMode;              /* mode */
   call InitWait;              /* Wait controller setting  */
   call InitPort;              /* General-purpose I/O port setting  */
   call InitInt;               /* Interrupt initialization  */
   call SetHifBusy;            /* Busy setting */
   WaitCom:
      call InitPort;           /* General-purpose I/O port setting  */

      call SetHifReady;        /* Ready setting */

      clr(r1);                 /* Command ID input */
      r1l= *HDT:x;
      call SetHifBusy;         /* Busy setting */

      r0 = r1 - 11;            /* Command ID check */
      if( r0>0 ) call ErrorDump;

      r0 = r1 + ComTbl;        /* Call command processing */
      /*---*/          dp0=r0l;
      nop;
      /*   */          r0l=*dp0;
      /*---*/          dp0=r0l;
      call dp0;

      jmp  WaitCom;


/*************************************************************************/
/* Function table used to perform command processing                     */
```

```
/*************************************************************************/
#ifdef  UPD77116
Sys_XC     xramseg
#else
Sys_XC     xromseg
#endif
ComTbl:
    dw  NoProc                  /* ComId=0  : No processing */
    dw  PowerDown               /* ComId=1  : Power down  */
    dw  SetData                 /* ComId=2  : Data set */
    dw  PutData                 /* ComId=3  : Data put */
    dw  Com4                    /* ComId=4  : Virtual parameter */
    dw  Com5                    /* ComId=5  : Virtual parameter */
    dw  Com6                    /* ComId=6  : Virtual parameter */
    dw  Com7                    /* ComId=7  : Virtual parameter */
    dw  Com8                    /* ComId=8  : Virtual parameter */
    dw  Com9                    /* ComId=9  : Virtual parameter */
    dw  Com10                   /* ComId=10 : Virtual parameter */
    dw  Com11                   /* ComId=11 : Virtual parameter */



/*************************************************************************/
/*                                                                       */
/*************************************************************************/
Sys_II3 imseg
InitMode:
    r0l= 0x440;
    *HST:x=r0l;
;    r0l= 0x0F0;             Use HWE,HRE
;    *ICR:x = r0l;
    ret;

/*************************************************************************/
/* Initialize wait controller                                            */
/*************************************************************************/
Sys_II3 imseg
InitWait:
    r0l= DWTR_WAIT0;            /*  External data memory setting */
    *DWTR:x = r0l;              /*  External instruction memory setting */
    r0l= IWTR_WAIT0;
    *IWTR:x = r0l;
    ret;

/*************************************************************************/
/* Set general-purpose I/O ports (all input)                             */
/*************************************************************************/
Sys_II4 imseg
InitPort:
    r0l= PIO_ALL_IN;
    *PCD:x = r0l;
    ret;

$EJECT
/**/
/*************************************************************************/
/* Initialize interrupt mask flag etc.                                   */
/*                                                                       */
/*  others: r_     [*, , , , , , , ]   dmx,dmy       [ , ]               */
/*          dp_    [ , , , , , , , ]   loops/stacks  [0/0]               */
```

```
/*         dn_    [ , , , , , , , ]   cycles        8                        */
/*****************************************************************************/
Sys_II5 imseg
InitInt:
   clr(r0);
   r0l= 0xFFFF;                /* EIR initialization */
   EIR= r0l;

   fint;                       /* Interrupt request annulled */

   r0l= SR;                    /* Interrupt enable flags change */
   r0 = r0 | 0x03FF;           /*  All disabled  */
   SR = r0l;
   ret;

/*****************************************************************************/
/* Set interrupt mask flag                                                   */
/* Change interrupt status                                                   */
/*                                                                           */
/*  input : r0l   change-bit setting                                         */
/*          r1l    Interrupt status setting 0x8000: Change, 0x0000: No change   */
/*  others: r_    [ , ,*,*, , , , ]   dmx,dmy       [ , ]                   */
/*          dp_   [ , , , , , , , ]   loops/stacks  [0/0]                   */
/*          dn_   [ , , , , , , , ]   cycles        10                      */
/*****************************************************************************/
Sys_II6 imseg
SetIntMask:
   r2l= EIR;                   /* Interrupt disable setting */
   r3 = r2 | 0x8000;
   EIR= r3l;
   r3 = r2 ^ r1;
   r2l= SR;                    /* Interrupt enable flag change */
   r2 = r2 | r0;
   SR = r2l;
   EIR= r3l;                   /* Interrupt status change  */
   ret;

/*****************************************************************************/
/* Clear interrupt mask flag                                                 */
/* Change interrupt status                                                   */
/*                                                                           */
/*  input : r0l    Change-bit setting                                        */
/*          r1l    Interrupt status setting 0x8000: Change, 0x0000: No change   */
/*  others: r_     [*, ,*,*, , , , ]   dmx,dmy       [ , ]                  */
/*          dp_    [ , , , , , , , ]   loops/stacks  [0/0]                  */
/*          dn_    [ , , , , , , , ]   cycles        11                     */
/*****************************************************************************/
Sys_II7 imseg
ClrIntMask:
   clr(r2);
   r2l= EIR;                   /* Interrupt disable setting */
   r3 = r2 | 0x8000;
   EIR= r3l;
   r3 = r2 ^ r1;
   r2l= SR;                    /* Interrupt enable flag change */
   r0 = r0 ^ 0xFFFF;
   r2 = r2 & r0;
   SR = r2l;
   EIR= r3l;                   /* Interrupt status change */
   ret;
```

```
$EJECT
/**/
/*****************************************************************************/
/* Set host IF to Busy                                                       */
/*                                                                           */
/*  others: r_    [*, , , , , , , ]   dmx,dmy        [ , ]                   */
/*          dp_   [ , , , , , , , ]   loops/stacks   [0/0]                   */
/*          dn_   [ , , , , , , , ]   cycles         6                       */
/*****************************************************************************/
Sys_II8 imseg
SetHifBusy:
   r0l= *HST:x;                /*  Flag-bit save */
   R0 = R0 & 0x003F;           /*  16-bit, with wait, busy */
   r0 = r0 | 0x0440;
   *HST:x = r0l;
   ret;

/*****************************************************************************/
/* Set host IF to Ready                                                      */
/*  * Caution: HST-related error flag will be cleared                        */
/*                                                                           */
/*  others: r_     [*, , , , , , , ]   dmx,dmy        [ , ]                   */
/*          dp_    [ , , , , , , , ]   loops/stacks   [0/0]                   */
/*          dn_    [ , , , , , , , ]   cycles         4                       */
/*****************************************************************************/
Sys_II9 imseg
SetHifReady:
   r0l= HST_WAIT;              /*  16-bit, with wait, ready */
   *HST:x = r0l;
   ret;

$EJECT
/**/
/*****************************************************************************/
/* Download to external instruction and data memories                        */
/*                                                                           */
/*  others: r_     [*, , , , , , ,*]    dmx,dmy         [ , ]                */
/*          dp_    [*, , ,*,*, , , ]    loops/stacks    [2/1]                */
/*          dn_    [ , , , , , , , ]    cycles          ...                  */
/*****************************************************************************/
Sys_II10 imseg
DownLoadProc:
   clr(r7);                          /* Data set to external instruction memory */
   r7l= *HDT:x;                      /*  Instruction count */
   if( r7==0 ) jmp Xmem;             /*                    */
   r0l= *HDT:x;                      /*  Segment address   */
   /*---*/          dp3=r0l;        /*                    */
   call HostReBoot;

  Xmem:
   clr(r0);
   r0l= *HDT:x;                      /* Data set to x memory */
   if( r0==0 ) jmp  Ymem;            /*  Segment count */
   loop r0l {                        /*               */
      r0l= *HDT:x;                   /*  Segment data count */
      r1l= *HDT:x;                   /*                     */
      /*---*/       dp0=r1l;        /*  Segment address */
      loop r0l {                     /*                  */
         r0l= *HDT:x;                /*  Setting data */
```

```
          /*   */    *dp0++=r0l;
       }
       clr(r0);
   }

  Ymem:
   r0l= *HDT:x;                        /* Data set to y memory */
   if( r0==0 ) ret;                    /*  Segment count */
   loop r0l {                          /*             */
      r0l= *HDT:x;                     /*  Segment data count */
      r1l= *HDT:x;                     /*                  */
      /*---*/        dp4=r1l;         /*  Segment address */
      loop r0l {                       /*               */
         r0l= *HDT:x;                  /*  Setting data */
         /*   */    *dp4++=r0l;
      }
      nop;
   }
   ret;

/*************************************************************************/
/*  Check operation status                                               */
/*                                                                       */
/*  others: r_     [*,*, , , , , , ]   dmx,dmy       [ , ]              */
/*          dp_    [ , , , , , , , ]   loops/stacks  [0/0]              */
/*          dn_    [ , , , , , , , ]   cycles        14                 */
/*************************************************************************/
Sys_IE1 imseg
HealthCheck:
   clr(r0);
   r0l= ESR;
   if( r0!=0 ) call  ErrorDump;

   r0l= LSP;
   if( r0!=0 ) call  ErrorDump;

   r0l= EIR;
   r1 = r0 - 0xFFFF;
   if( r1!=0 ) call  ErrorDump;

   r0l= SP;
   r1 = r0 - 0x0001;
   if( r1!=0 ) call  ErrorDump;

   r0l= *HST:x;                        /* Wait for host to */
   r1 = r0 & HST_ERRMASK;              /* read out parameter */
   if( r1!=0 ) call  ErrorDump;
   r1 = r0 & HST_SENMASK;
   if( r1!=0 ) jmp  $-4;

   nop;
   ret;

$EJECT
/**/
/*************************************************************************/
/* For error Dump                                                        */
/*************************************************************************/
Sys_XE xramseg
EdLock:
```

```
    ds  1;
CoreImage:
    ds  0x20;

/*************************************************************************/
/*  Dump error data, end command processing                              */
/*************************************************************************/
Sys_IE2 imseg
ErrorDump:
    *CoreImage+0x02:x = r0l;            /* Interrupt disable setting */
    *CoreImage+0x03:x = r0h;            /* Register value dump */
    call InitInt;
    *CoreImage+0x04:x = r1l;
    *CoreImage+0x05:x = r1h;
    *CoreImage+0x06:x = r2l;
    *CoreImage+0x07:x = r2h;
    *CoreImage+0x08:x = r3l;
    *CoreImage+0x09:x = r3h;
    *CoreImage+0x0A:x = r4l;
    *CoreImage+0x0B:x = r4h;
    *CoreImage+0x0C:x = r5l;
    *CoreImage+0x0D:x = r5h;
    *CoreImage+0x0E:x = r6l;
    *CoreImage+0x0F:x = r6h;
    *CoreImage+0x10:x = r7l;
    *CoreImage+0x11:x = r7h;
    r0l= dp0;
    *CoreImage+0x12:x = r0l;
    r0l= dp1;
    *CoreImage+0x13:x = r0l;
    r0l= dp2;
    *CoreImage+0x14:x = r0l;
    r0l= dp3;
    *CoreImage+0x15:x = r0l;
    r0l= dp4;
    *CoreImage+0x16:x = r0l;
    r0l= dp5;
    *CoreImage+0x17:x = r0l;
    r0l= dp6;
    *CoreImage+0x18:x = r0l;
    r0l= dp7;
    *CoreImage+0x19:x = r0l;
    r0l= STK;                           /* Error occurrence address dump */
    *CoreImage+0x00:x = r0l;
    r0l= ESR;                           /* ESR value dump */
    *CoreImage+0x01:x = r0l;

    r0l= *HST:x;                        /* Error flag set */
    r0 = r0 & 0x003F;                   /*  Flag-bit save */
    r0 = r0 | 0x04C0;                   /*  16 bit, with wait, busy, error */
    *HST:x = r0l;

    clr(r0);                            /* Initialize stack(etc.) */
    SP = r0l;
    LSP= r0l;
    ESR= r0l;

    clr(r0);                            /* Wait for INT4(Vect:7) */
    *EdLock:x = r0l;
    r0l= INTFLG_INT4;
```

```
    r1l= 0x8000;
    call ClrIntMask;
    nop;
    r0 = *EdLock:x;
    if( r0==0 ) jmp  $-2;
    call InitInt;

    /*---*/           dp0=CoreImage;    /* Error data reported to Host */
    loop 0x1A {
        /*...*/       r0l=*dp0++;
        *HDT:x = r0l;
    }

    r0l= *HST:x;                        /* Clear host input */
    r0 = r0 & HST_LENMASK;              /* Return to command wait status */
    if( r0!=0 ) jmp  WaitCom;
    r0l= *HDT:x;
    jmp  WaitCom;

StartErrDump:                           /* INT2(Vect:7) processing */
    r0l= 1;
    *EdLock:x = r0l;
    ret;

$EJECT
/**/
/*************************************************************************/
/* No processing <for interrupt processing>                              */
/*************************************************************************/
Sys_IE3 imseg
IntNoProc:
    reti;


/*************************************************************************/
/* No processing                                                         */
/*************************************************************************/
Sys_IE4 imseg
NoProc:
    ret;


/*************************************************************************/
/* Power down                                                            */
/*  *1) No return to normal operation from power down mode               */
/*************************************************************************/
Sys_IE5 imseg
PowerDown:
    r0l= EIR;                   /* All interrupts:disabled */
    r0 = r0 | 0x8000;           /* Disable Power Down release */
    EIR= r0l;
    nop;

     PdLoop:
       halt;                    /* PowerDown */
       jmp  PdLoop;

$EJECT
/**/
/*************************************************************************/
```

```
/* Read data from host IF, store in specified area                          */
/*  1. Receive stored parameters from host IF                               */
/*     Memory classification 0) x memory 1) y memory                        */
/*     Stored addresses                                                     */
/*     Stored data count                                                    */
/*  2. Receive stored data from host IF, store in specified memory          */
/*                                                                          */
/*  others: r_     [*,*,*,*, , , , ]   dmx,dmy        [ , ]                */
/*          dp_    [*, , , ,*, , , ]    loops/stacks   [1/1]                */
/*          dn_    [ , , , , , , , ]   cycles         ...                  */
/***************************************************************************/
Sys_IE6 imseg
SetData:
   clr(r2);                          /* Stored parameter introduction  */
   clr(r3);                          /*                             */
   r1  = *HDT:x;                     /*  Memory classification */
   r2l = *HDT:x;                     /*  Stored address */
   r3l = *HDT:x;                     /*  Stored data count */

   if( r3==0 ) ret;                  /* Stored parameter analysis */
   if( r1!=0 ) jmp  SetYdata;

  SetXdata:
    /*---*/       dp0=r2l;          /* Stored to x memory */
    loop r3l {
       r0  = *HDT:x;
       /*...*/    *dp0++=r0h;
    }
    ret;

  SetYdata:
    /*---*/       dp4=r2l;          /* Stored to y memory */
    loop r3l {
       r0  = *HDT:x;
       /*...*/    *dp4++=r0h;
    }
    ret;

/***************************************************************************/
/* Output data stored in specified area from host IF                        */
/*  1. Receive stored area specification value from host IF                 */
/*      Memory classification 0) x memory 1) y memory                       */
/*      Stored addresses                                                    */
/*      Output data count                                                  */
/*  2. Output stored data to host IF                                        */
/*                                                                          */
/*  others: r_     [*,*,*,*, , , , ]  dmx,dmy       [ , ]                 */
/*          dp_    [*, , , ,*, , , ]   loops/stacks  [1/1]                 */
/*          dn_    [ , , , , , , , ]   cycles         ...                  */
/***************************************************************************/
Sys_IE7 imseg
PutData:
   clr(r2);                          /* Stored parameter introduction */
   clr(r3);                          /*                            */
   r1  = *HDT:x;                     /*  Memory classification */
   r2l = *HDT:x;                     /*  Store addresses */
   r3l = *HDT:x;                     /*  Stored data count */

   if( r3==0 ) ret;                  /* Stored parameter analysis */
   if( r1!=0 ) jmp  PutYdata;
```

```
  PutXdata:
    /*---*/       dp0=r2l;              /* Output from x memory */
    loop r3l {
       /*...*/    r0l=*dp0++;
       *HDT:x = r0l;
    }
    ret;

  PutYdata:
    /*---*/       dp4=r2l;             /* Output from y memory  */
    loop r3l {
       /*...*/    r0l=*dp4++;
       *HDT:x = r0l;
    }
    ret;
/*-------------------------------------------------------------------------*/
/*   Function setting definition                                           */
/*-------------------------------------------------------------------------*/
#define Dec              ; Use decoder function
#define Enc              ; Use encoder function

#ifdef  Enc
extrn   g729b_InitEncM
extrn   g729b_EncM
#endif

#ifdef  Dec
extrn   g729b_InitDecM
extrn   g729b_DecM
#endif

;public  Encode
;public  Decode

;public  InitG729b

Mchannel_X  xramseg
cur_ch:    ds  1

/*************************************************************************/
/* Input processing of PCM data for Coder                                */
/*************************************************************************/
Ld8k_Ix  imseg
GetPcmData:
    r0l= dp0                          ;
    r0 = r0+2                         ;
    /*---*/           dp1=r0l        ;
    nop                               ;
    /*---*/           r0l=*dp1       ;
    /*---*/           dp1=r0l        ;
    loop L_FRAME {                    ;

       r0l= *HDT:x                    ;
       /*   */        *dp1++ =r0l     ;
    }                                 ;
    ret                               ;

/*************************************************************************/
/* Output processing of compression data for Coder                       */
```

```
/*************************************************************************/
Ld8k_Ix  imseg
PutPrmData:
    r0l= dp0                            ;
    r0 = r0 + 3                         ;
    /*---*/           dp1=r0l           ;
    nop                                 ;
    /*---*/           r1l=*dp1          ;
    /*---*/           dp1=r1l           ;
    r0 = r0 + 14                        ; dp0 + 3+14 -> dp0 + 0x11
    dp2 = r0l                           ;
    nop                                 ;
    r0l = *dp2                          ; Get FTYP
    *HDT:x  = r0l                       ; put FTYP

    loop PRM_SIZE {                     ;
        /*   */        r0l=*dp1++       ;
        *HDT:x = r0l                    ;
    }                                   ;
    ret                                 ;

/*************************************************************************/
/* Output processing of PCM data for Decode                              */
/*************************************************************************/
Ld8k_Ix imseg;
PutPcmData:
    r0l= dp0                            ;
    r0 = r0 + 8                         ;
    /*---*/           dp1=r0l           ;
    nop                                 ;
    /*---*/           r0l=*dp1          ;
    /*---*/           dp1=r0l           ;
    loop L_FRAME {                      ;
        /*   */        r0l=*dp1++       ;
        *HDT:x = r0l                    ;
    }                                   ;
    ret                                 ;

/*************************************************************************/
/* Input processing of compression data for Decode                       */
/*************************************************************************/
Ld8k_Ix imseg
GetPrmData:
    r0l= dp0                            ;
    r0 = r0 + 9                         ;
    /*---*/           dp1=r0l           ;
    nop                                 ;
    /*---*/           r0l=*dp1          ;
    /*---*/           dp1=r0l           ;
    loop PRM_SIZE {                     ;
        r0l= *HDT:x                     ;
        /*   */        *dp1++=r0l       ;
    }                                   ;

    ret                                 ;


/*************************************************************************/
/* Initialization                                                        */
/*************************************************************************/
```

```
Ld8k_Ix imseg
InitG729b:
   call g729b_MkIO_Table               ;
#ifdef  Enc
   call   g729b_StartCodec             ;
   /*---*/        dp0=g729b_IO_Table1  ;
   call   g729b_InitEncM               ;
#endif
#ifdef  Dec
   /*---*/        dp0=g729b_IO_Table1  ;
   call   g729b_InitDecM               ;
#endif
#ifdef  Enc
   /*---*/        dp0=g729b_IO_Table2  ;
   call   g729b_InitEncM               ;
#endif
#ifdef  Dec
   /*---*/        dp0=g729b_IO_Table2  ;
   call   g729b_InitDecM               ;
#endif
#ifdef  Enc
   /*---*/        dp0=g729b_IO_Table3  ;
   call   g729b_InitEncM               ;
#endif
#ifdef  Dec
   /*---*/        dp0=g729b_IO_Table3  ;
   call   g729b_InitDecM               ;
#endif

#ifdef  Enc
   /*---*/        dp0=g729b_IO_Table4  ;
   call   g729b_InitEncM               ;
#endif
#ifdef  Dec
   /*---*/        dp0=g729b_IO_Table4  ;
   call   g729b_InitDecM               ;
#endif

;   call   InitCycleDt                  ; *
;   call   StMipsDt                     ; *
   ret                                 ;

#ifdef  Enc
/*****************************************************************************/
/* Encode 1 frame                                                            */
/*****************************************************************************/
Encode:
   r0l= *HDT:x                         ; Get channel No.
   *cur_ch:x=r0l                       ;
   call getCurIO_Table                 ;
   r0l= *HDT:x                         ; Get Vad Mode
   call setVadMode                     ;
   call IncFrameCount                  ;
   call getCurIO_Table                 ;
   call GetPcmData                     ;
   call getCurIO_Table                 ;
;   call ClrMipsDt                     ; *
   call g729b_EncM                     ;
;   call PutMipsDt                     ; *
   call getCurIO_Table                 ;
```

```
   call PutPrmData                    ;
  ret                                 ;

/*************************************************************************/
/*  setup VAD flag(Encode)                                               */
/*************************************************************************/
setVadMode:
   r1l= dp0                           ;
   r1 = r1 + 16                       ;
   /*---*/           dp1=r1l          ;
   nop                                ;
   /*---*/           *dp1=r0l         ;
   ret                                ;

/*************************************************************************/
/*  Incriment frame counter(Encode)                                      */
/*************************************************************************/
IncFrameCount:
   r0l= dp0                           ;
   r0 = r0 + 4                        ;
   /*---*/           dp1=r0l          ;
   nop                                ;
   /*...*/           r0 = *dp1        ;
   r0 = r0 sra 16                     ;
   r1 = r0 - 32767                    ;
   if(r1 != 0) jmp $+2                ;
      r0l = 255                       ;
   r0 = r0 + 1                        ;
   /*...*/            *dp1=r0l        ;
   ret                                ;
#else
Encode:
   ret                                ; Dummy function
#endif

#ifdef  Dec
/*************************************************************************/
/*  Decode 1 frame                                                       */
/*************************************************************************/
Decode:
   r0 = *HDT:x                        ; ch No.
   *cur_ch:x = r0h                    ;
   call getCurIO_Table                ;
   r0 = *HDT:x                        ; erase
   r1 = *HDT:x                        ; ftype
   call DecSetup                      ;

   call GetPrmData                    ;
;    call ClrMipsDt                   ;
   call g729b_DecM                    ;
;    call PutMipsDt                   ;
   call getCurIO_Table                ;
   call PutPcmData                    ;
   ret                                ;

/*************************************************************************/
/*  setup decoder parameter                                              */
/*************************************************************************/
DecSetup:
   r2l= dp0                           ;
```

```
    /*---*/           dp1=r2l           ;
    nop                               ;
    /*...*/           r2 =*dp1##12      ;
    /*...*/           *dp1##6=r0h       ;
    /*...*/           *dp1=r1h          ;
    ret                               ;

#else
Decode:
    ret                               ; Dummy function
#endif
/*****************************************************************************/
/*  get current IO Control Table pointer                                     */
/*****************************************************************************/
;; Step 2-6 (d)
getCurIO_Table:
    r0 = *cur_ch:x                    ;

        clr(r1);
        r1l = g729b_IO_Table2;
        r1 = r1 - g729b_IO_Table1;
;;      r1l = g729b_IO_Table2 - g729b_IO_Table1;
        r1 = r1 sll 16;
        r0 = r0h * r1h;
        r0 = r0 sra 1;
        r0 = r0 + g729b_IO_Table1;
        dp0 = r0l;
        ret;

#if 0
    r0 = r0 sra 16                    ;
    if(r0 != 0) jmp $+3               ;
       dp0=g729b_IO_Table1            ;
       ret                            ;
    r0 = r0 - 1                       ;
    if(r0 != 0) jmp $+3               ;
       dp0=g729b_IO_Table2            ;
       ret                            ;
    dp0=g729b_IO_Table3               ;
    ret                               ;
#endif


$EJECT
/**/
/*****************************************************************************/
/* Data for Debug                                                            */
/*****************************************************************************/
Debug_Ye yramseg
NextAdr:    ds  1
DebDat:     ds  1024

/*****************************************************************************/
/* Initialization for Debug                                                  */
/*****************************************************************************/
Debug_Ix imseg
IniDebDat:
    r0l= DebDat;
    *NextAdr:y = r0l;
    ret;
```

```
/*****************************************************************************/
/* Debug data output                                                         */
/*****************************************************************************/
Debug_Ix imseg
PutDebDat:
    clr(r0);
    r0l= *NextAdr:y;
    r0 = r0 - DebDat;
    *HDT:x = r0l;
    if( r0==0 ) ret;
    /*---*/              dp4=DebDat;
    loop r0l {
       /*   */           r0l=*dp4++;
       *HDT:x = r0l;
    }
    r0l= DebDat;
    *NextAdr:y = r0l;
    ret;

end
```

付.2.1　samplebm.asm 用ヘッダ・ファイル（sysconf.h）

```
/*************************************************************************/
/* SPX/evaluation board, application definition file                      */
/*************************************************************************/

/*---- Parameter definition ----*/
#define SI1Proc   IntNoProc    /* SIO1(IN)       <Interrupt processing>  */
#define SO1Proc   IntNoProc    /* SIO1(OUT)      <Interrupt processing>  */
#define SI2Proc   IntNoProc    /* SIO2(IN)       <Interrupt processing>  */
#define SO2Proc   IntNoProc    /* SIO2(OUT)      <Interrupt processing>  */
#define HostIn    IntNoProc    /* Host IF input  <Interrupt processing>  */
#define HostOut   IntNoProc    /* Host IF output <Interrupt processing>  */
#define Codec1CK  IntNoProc    /* INT1           <Interrupt processing>  */
#define Codec1FM  IntNoProc    /* INT2           <Interrupt processing>  */
#define Codec2CK  IntNoProc    /* INT3           <Interrupt processing>  */

#define MAX_HIP_NO      3      /* When [P0..2] is host programmable 7    */
                               /* When [P0..1] is host programmable 3    */

#define HintProc1  NoProc    /* INT4(Vect:1) processing  */
#define HintProc2  NoProc    /* INT4(Vect:2) processing  */
#define HintProc3  NoProc    /* INT4(Vect:3) processing  */
#define HintProc4  NoProc    /* INT4(Vect:4) processing. Use prohibited when [P0..1] */
#define HintProc5  NoProc    /* INT4(Vect:5) processing. Use prohibited when [P0..1] */
#define HintProc6  NoProc    /* INT4(Vect:6) processing. Use prohibited when [P0..1] */
#define HintProc7  NoProc    /* INT4(Vect:7) processing. Use prohibited when [P0..1] */

#define Com4       NoProc       /* ComID=4      processing      */
#define Com5       NoProc       /* ComID=5      processing      */
#define Com6       Encode       /* ComID=6      processing      */
#define Com7       Decode       /* ComID=7      processing      */
#define Com8       NoProc       /* ComID=8      processing      */
#define Com9       NoProc       /* ComID=9      processing      */
#define Com10      NoProc       /* ComID=10     processing      */
#define Com11      InitG729b    /* ComID=11     processing      */
/* uPD77116 only */
ICR     equ 0x3828              /* Interrupt control register   */


/*---- Register setting value definition ----*/
DWTR_WAIT0      equ 0x0000      /* Wait setting value (data)            0wait   */
DWTR_WAIT1      equ 0x5454      /*                                      1wait   */
DWTR_WAIT3      equ 0xA8A8      /*                                      3wait   */
DWTR_WAIT7      equ 0xFCFC      /*                                      7wait   */
IWTR_WAIT0      equ 0x0000      /* Wait setting value (instruction)     0wait   */
IWTR_WAIT1      equ 0x0054      /*                                      1wait   */
IWTR_WAIT3      equ 0x00A8      /*                                      3wait   */
IWTR_WAIT7      equ 0x00FC      /*                                      7wait   */

PIO0_INP        equ 0x2001      /* General-purpose I/O register setting value   */
PIO0_HIGH       equ 0xF001
PIO0_LOW        equ 0xB001
PIO1_INP        equ 0x2002
PIO1_HIGH       equ 0xF102
PIO1_LOW        equ 0xB102
PIO2_INP        equ 0x2004
PIO2_HIGH       equ 0xF204
PIO2_LOW        equ 0xB204
```

```
PIO3_INP        equ 0x2008
PIO3_HIGH       equ 0xF308
PIO3_LOW        equ 0xB308
PIO_ALL_IN      equ 0x200F

HST_WAIT        equ 0x0400      /* Host status register setting value   */
HST_NOWAIT      equ 0x0000
HST_MASK        equ 0x00FF
HST_LENMASK     equ 0x0001
HST_SENMASK     equ 0x0002
HST_ERRMASK     equ 0x003C

SST_WAIT        equ 0x0F00      /* Serial status register setting value */
SST_NOWAIT      equ 0x0300
SST_HALT        equ 0x0000
SST_ERRMASK     equ 0x000C
SST_LENMASK     equ 0x0001
SST_SENMASK     equ 0x0002
INTFLG_INT1     equ 0x0001      /* Interrupt register setting value     */
INTFLG_INT2     equ 0x0002
INTFLG_INT3     equ 0x0004
INTFLG_INT4     equ 0x0008
INTFLG_SI1      equ 0x0010
INTFLG_SO1      equ 0x0020
INTFLG_SI2      equ 0x0040
INTFLG_SO2      equ 0x0080
INTFLG_HI       equ 0x0100
INTFLG_HO       equ 0x0200

/*---- External bus definition ----*/

/*---- Function entry ----*/
HostReBoot      equ 0x0005      /* Host reboot */

$LIST
```

［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（インフォメーションセンター）
（電話：午前 9:00～12:00，午後 1:00～5:00）

電　話　：044−548−8899
FAX　：044−548−7900
E-mail　：s-info@saed.tmg.nec.co.jp

【営業関係お問い合わせ先】

半導体第一販売事業部 半導体第二販売事業部 半導体第三販売事業部	〒108-8001　東京都港区芝５－７－１　（日本電気本社ビル）		(03)3454−1111
中部支社	半導体第一販売部 半導体第二販売部	〒460-8525　愛知県名古屋市中区錦１－17－１　（日本電気中部ビル）	(052)222−2170 (052)222−2190
関西支社	半導体第一販売部 半導体第二販売部 半導体第三販売部	〒540-8551　大阪府大阪市中央区城見１－４－24　（日本電気関西ビル）	(06)6945−3178 (06)6945−3200 (06)6945−3208

北海道支社	札　幌	(011)231−0163	甲府支店	甲　府	(055)224−4141	京都支社	京　都	(075)344−7824
東北支社	仙　台	(022)267−8740	長野支社	松　本	(0263)35−1662	神戸支社	神　戸	(078)333−3854
岩手支店	盛　岡	(019)651−4344	静岡支社	静　岡	(054)254−4794	中国支社	広　島	(082)242−5504
郡山支店	郡　山	(024)923−5511	立川支社	立　川	(042)526−5981,6167	鳥取支店	鳥　取	(0857)27−5311
長岡支店	長　岡	(0258)36−2155	埼玉支社	大　宮	(048)649−1415	岡山支店	岡　山	(086)225−4455
水戸支店	水　戸	(029)226−1717	千葉支社	千　葉	(043)238−8116	四国支社	松　山	(089)945−4149
群馬支店	高　崎	(027)326−1255	神奈川支社	横　浜	(045)682−4524	九州支社	福　岡	(092)261−2806
太田支店	太　田	(0276)46−4011	三重支店	津	(059)225−7341			
宇都宮支店	宇都宮	(028)621−2281	北陸支社	金　沢	(076)232−7303			

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-B03　ユーザーズ・マニュアル
（U13373JJ3V1UM00（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，NEC 半導体ソリューション技術本部員，
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。

NEC 半導体テクニカルホットライン
FAX：（044）548-7900

キリトリ