RX62N グループ
ADPCM 圧縮オーディオデータの録音と再生

R01AN0698JU0102
Rev.1.02
2011.08.10

要旨

このドキュメントは、適応差分パルス符号変調 ADPCM（Adaptive Differential Pulse Code Modulation）の内容と、これが RX62N をベースとした組み込み用プラットフォーム上のオーディオデータの圧縮と復号にどのように利用されるかを解説しています。このドキュメントには実例となるソースコードとプロジェクトファイルが含まれています。

以下の理由により、コストの大きなハードウェアを使用することなく組み込みアプリケーションにオーディオ再生機能を容易に追加することが可能となります。

- PWM 信号をフィルタに通すことで、再生のための D/A コンバータが不要となります。
- 4:1 ADPCM 圧縮方式を採用することで、必要なメモリが小さく押さえられます。
- シリアル接続のフラッシュメモリを使用することで、通常必要とされるフラッシュメモリを内部もしくは外部にパラレル接続で追加する必要がなくなります。
- 低コストのシリアルデバッグ・インタフェースとオーディオデータ関連のプログラミングを自動的に行う Hew Target Server のようなツールを使用することで、外部のオーディオ用メモリをサポートするためのプログラムの変更を容易に実現することができます。

動作確認デバイス

RX62N

このアプリケーションでは、RX62N MCU をベースとする R5F562N8BDFP チップ（100 pin、96kB RAM、512+32 kB フラッシュメモリ）を搭載した YRDKRX62N ボードを使用します。

関連ドキュメント

- REJ06J0085-0100、Rev.1.00、Jun.15.2010、 "M3S-S2-Tiny" for RX Family. Sound Data Compression/Expansion Software: このドキュメントにはオーディオデータの圧縮・展開ソフトウェアライブラリの使い方とこれを利用する簡単なプログラム例が記載されています。
- YRDK CD からソフトウェアがインストールされている場合は、スタート→すべてのプログラム→Renesas High Perf. Embedded Workshop にある Manual Navigator からクイックスタートガイド、RX ハードウェアマニュアル、ボードの回路図を含むドキュメント類を参照できます。

目次

1. 概要

このアプリケーションノートでは YRDKRX62N ボード上で ADPCM による圧縮・展開を行うプログラム例が提供されます。サンプルのワークスペースには、このドキュメントで解説されているソースコードとライブラリが含まれています。

2. ツールとドキュメント

YRDK62N CD をインストールしてください。これにより以下のものが提供されます。

- ソースコードプロジェクト生成ソフトとコンパイラを含む HEW。
- このボードで使用される Segger J-Link デバッガなど。
- このボードで使用されるドキュメント。これには回路図、MCU ハードウェアマニュアル、ユーザマニュアルとサードパーティ・デバイスの情報を含みます。次の図 1 をご覧ください。

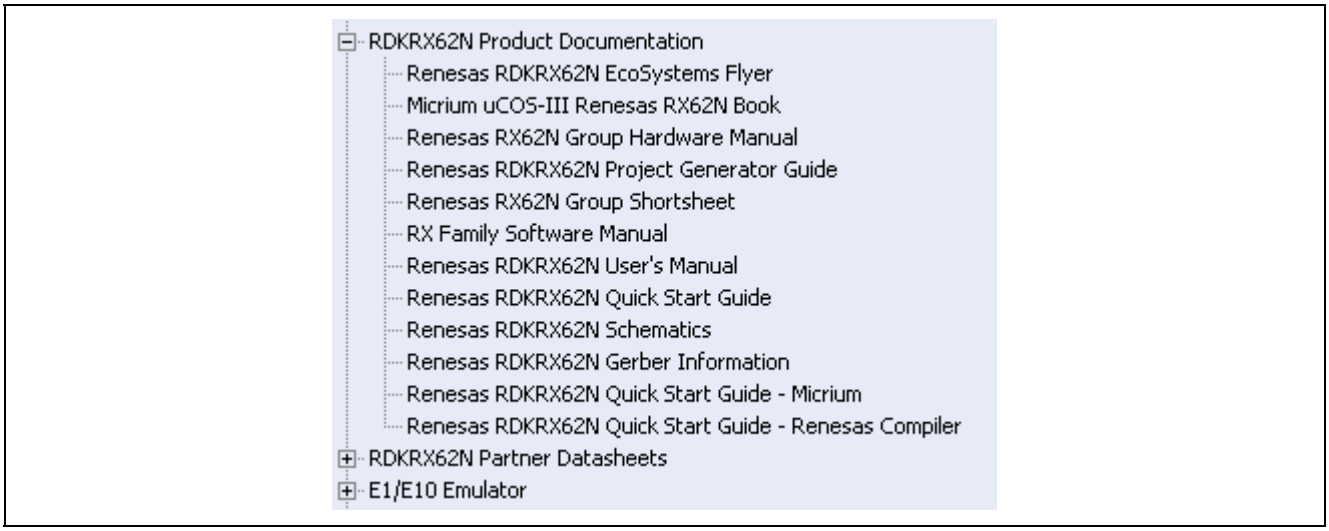


図1　YRDK62N CD をインストールすることで、このボードで使用されている MCU とサードパーティの周辺回路のドキュメントを参照できます。

2.1　ファイルディレクトリの構成

デモソフトを実行する方法については、ソースコードファイル
ADPCM_Demo¥sample_player¥main_ADPCM.c の先頭部分をご覧ください。

ADPCM 圧縮･展開ライブラリを作成するソースコードやプロジェクトファイルは複数のディレクトリにまたがって格納されています。このライブラリは録音・再生を行うプログラムの例で利用されています。

圧縮・展開ライブラリは次のように RX_RDK_ADPCM_Demo ディレクトリに格納されています。

```
RX_RDK_ADPCM_Demo       HEW の Workspace ディレクトリ
|
|-- ADPCM_Demo          HEW のプロジェクトディレクトリ
|     |
|     |-- ADPCM_audiodata   ADPCM オーディオデータ
|     |
|     |-- ADPCM_Lib       HEW の Workspace ディレクトリ
|     ||
|     ||-- ADPCM          ADPCM ライブラリ全体のプロジェクトディレクトリ
|     ||        |
|     ||        |-- lib   ADPCM 圧縮・展開ライブラリのライブラリディレクトリ
|     ||        |
|     ||        |-- Release   コンフィギュレーションディレクトリ
|     ||
|     ||-- ADPCM_Decode   展開機能のみの ADPCM ライブラリのプロジェクトディレクトリ
|     ||        |
|     ||        |-- lib   展開機能のみの ADPCM ライブラリのライブラリディレクトリ
|     ||        |
|     ||        |-- Release   コンフィギュレーションディレクトリ
|     ||
|     ||-- ADPCM_Encode   圧縮機能のみの ADPCM ライブラリのプロジェクトディレクトリ
|     ||        |
|     ||        |-- lib   圧縮機能のみの ADPCM ライブラリのライブラリディレクトリ
|     ||        |
|     ||        |-- Release   コンフィギュレーションディレクトリ
|     ||
|     ||-- src           ADCPM ライブラリのソースコード
|     |
|     |-- BSP           このボード専用のサポートルーチンのディレクトリ
|     |
|     |-- Debug         コンフィギュレーションディレクトリ
|     |
|     |-- RPDL          R-PDL 割り込みハンドラのソースコードとライブラリ
|     |
|     |-- sample_player   ADPCM 圧縮・展開デモプログラムのサンプルソースコード
|
|-- ADPCM_TOOL          ADPCM データ変換ツール
|     |
|     |-- bin           WAV ファイルから ADPCM 形式への変換ツール
|     |
|-- doc                 ADPCM データ変換ツールのドキュメント
```

2.1.1　各ディレクトリの内容

- ADPCM_audiodata
  ADPCM 圧縮されたサンプリングレート 11 kHz のサンプルオーディオファイルが格納されています。ファイル adpcm_sample.dat はデモソフトとリンクされてフラッシュメモリ内に置かれ、展開・再生をテストするオーディオデータとして使用されます。
- ADPCM_Lib
  ADPCM 圧縮・展開アルゴリズムに関するワークスペース、ソースコード、ライブラリファイルが置かれています。
- BSP
  YRDKRX62N ボードのためのボードサポート・パッケージが格納されています。これにはハードウェア初期化ルーチンや LCD 書き出し、スイッチ管理、RSPI バスアクセス調整のための API 層のドライバルーチンが含まれています。
- RPDL
  （以下の「3. RPDL」も参照してください）ライブラリ関数、サンプルの割り込みハンドラ、RPDL ライブラリファイルを参照するために必要なヘッダファイルが置かれています。RPDL はデモプログラムが録音・再生機能を実行する際に必要な種々の周辺回路のセットアップおよび変更などを行うために使用されます。
- sample_player
  デモプログラムのソースコードが置かれています。

このディレクトリツリーには 2 個のワークスペースが用意されています。ADPCM_Demo ワークスペースは YRDKRX62N ボード上で録音・再生を実行する際に ADPCM ライブラリがどのように利用されるかを示すサンプルプログラムです。他方の ADPCM_Lib は ADPCM ライブラリのバリエーション 3 種を生成するためのワークスペースです。

1. ADPCM　　データの圧縮と展開を行うフルセットのライブラリを生成します。
2. ADPCM_Decode　　デコード（展開）機能のみをもつライブラリを生成します。
3. ADPCM_Encode　　エンコード（圧縮）機能のみをもつライブラリを生成します。

3.　RPDL

ワークスペースには RPDL（Renesas Peripheral Driver Library）が含まれています。RPDL は所定の引数を伴う関数呼び出しで周辺回路のセットアップを行う場合、たとえばサンプルコードで録音を行う際の MTU の設定などを行うために使用されます。

RPDL のマニュアルはウィンドウのスタートメニューADPCM_Demo Workspace から参照できます。

4. オーディオの標本化と圧縮

今日では、オーディオの再生はほとんどがディジタルで行われています。これは、オーディオの録音と再生のために、元の波形をサンプリングし、その標本化されたデータを保存し、これをできるだけ忠実に元の信号に再現することが必要とされていることを意味しています。

サンプリング周波数が低すぎると、高い周波数が失われます。20 kHz を超える耳で聞こえない周波数帯が失われることは問題となりません。電話品質のオーディオは 8 kHz でサンプルされているため、再生できるもっとも高い周波数は 4 kHz となります。これがオーディオとしての最小限の品質といわれています。

オーディオをディジタル化する際のもうひとつの観点はサンプリングの幅です。サンプルのビット数が多くなるほど、再生される信号の品質は高くなります。このアプリケーションノートでは、オーディオ信号を 16 ビットで扱います。ただし用意されているサンプルデータでは 8 ビットのみが使用されています。

4.1 PCM（パルス符号変調）とデータ圧縮

データを圧縮する目的は、保存時のメモリ容量の節約と、通信時の帯域幅の縮小です。このアプリケーションノートでは ADPCM 方式を扱いますが、その前に PCM と DPCM について紹介しておきます。

4.1.1 PCM（パルス符号変調）

PCM（パルス符号変調）は直観的に理解しやすい方式です。個々のサンプリングされたデータは、前後の信号にかかわらず、それ自体が単純に数値化されます。最大振幅が 5V の系で 3V の信号であれば、16 ビット PCM の値は $3/5 * 2^{15} = 39322$（または $999A_H$）であり、8 ビット PCM では $3/5 * 2^8 = 153$ ($0x99_H$) という値となります。

4.1.2 DPCM（差分パルス符号変調）

直前のオーディオ信号のサンプル値が保存されていれば、信号の再現には直前の PCM 値との差を保存すれば充分です。もっとも単純な DPCM エンコードでは、オーディオ信号の値として直前の信号の値との差が保存されます。直前の信号の値との差のみが必要とされるため、データを保存するための容量は小さくなります。

よりスマートな DPCM エンコード方式では、エンコードとデコードの双方で次の PCM 値の予測がなされ、予測値と実際のサンプル値との差が保存されます。予測が正確であるほど信号を再現するために必要なビット数は少なくなります。

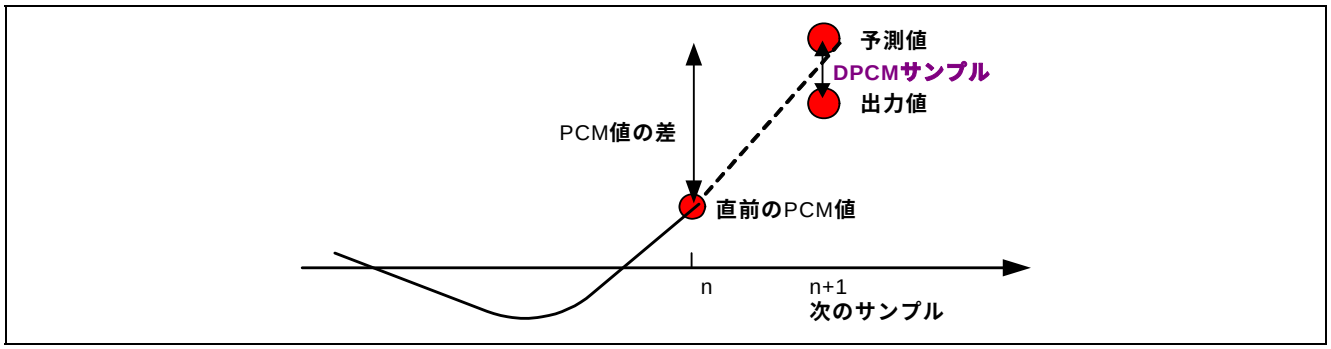


図2　DPCM 方式の変形ではエンコーダとデコーダの双方で PCM 値の予測がなされ、予測値との差がデータの値として保存されます。

4.1.3　ADPCM

ADPCM 方式を使えば保存されるデータはより小さくなります。圧縮比は 4:1 のとき、16 ビット PCM 値が 4 ビットの値となります。これで PCM 方式で 100 kB のオーディオファイルがメモリ上で 25 kB の容量となります。

ADPCM 方式では、サンプルごとに「重み」もしくは「ステップ幅」を変化させます。たとえば、エンコードされた値の 1 が、あるサンプルでは PCM 出力における差 40 を、別のサンプルでは差 72 を意味するならば、データのバンド幅は DPCM と比べてより小さくできます。

4 ビットのデータの値の意味は一定ではなく、サンプルごとに意味する差の値が異なります。たとえば、

ビット番号	3	2	1	0
符号	**保存値（0〜7）**			

個々のサンプルの PCM 値は次の ADPCM アルゴリズムで与えられます。

Output(n) = Output(n-1) +- stepsize x (b2 + b1 / 2 + b0 / 4 + 1/8)
ここで 符号+-は b3 **によって決められます。**

この回帰的な式を適用することで、デコードされた次の PCM 出力値がエンコードされたデータの値そのものとしてではなく stepsize に依存するものとして与えられます。

同時に、個々のサンプルにおける stepsize も、エンコードされたデータの値（bit 2〜0）と 2 個の表の参照で決められます。保存データの値は 0〜7 ですが、stepsize は常に変化つまり「適応」することになり、サンプル間の最終的な PCM 値の差を大きな値とすることができます。

最初の表（上）は stepsize 表（下）の中での移動数を与えます。

ADPCM **入力値**	Table **引数の変化**
7	8
6	6
5	4
4	2
3	-1
2	-1
1	-1
0	-1

```
static const  unsigned short   stepsizeTable[] = {
          5,    6,    7,    8,    9,   10,   11,   12,   14,   15,
         17,   18,   20,   22,   25,   27,   30,   33,   37,   40,
         44,   49,   54,   59,   65,   72,   79,   87,   95,  105,
        116,  127,  140,  154,  170,  187,  205,  226,  248,  273,
        301,  331,  364,  400,  440,  485,  533,  586,  645,  710,
        781,  859,  945, 1039, 1143, 1258, 1384, 1522, 1674, 1842,
       2026, 2228, 2451, 2697, 2966, 3263, 3589, 3948, 4343, 4777,
       5255, 5781, 6359, 6995, 7694, 8464, 9310,10242,11266,12392,
      13632,14995,16494,18144,19958,21954,24150,26565
};
```

2 つの表の使い方は、次の例のようになります。

あるサンプル時点で stepsize が 400 であるとします。(stepsizeTable[]の値 400 の位置に注目してください。）保存されていた次のサンプルの値（4 ビット）が 5 であれば、最初の表で黄色で示されているように、stepsize 表の中で 4 移動することが示され、stepsize は 586 となります。次の保存値（4 ビット）が 2 であれば stepsize 表での移動は-1 となり、586 から表の 1 つ小さい値に移動し、次のサンプルの stepsize は 533 となります。

5. オーディオデータのメモリ容量

オーディオデータは MCU に格納することもできますが、すぐにメモリ容量が不足する場合が多くあります。ADPCM で 4:1 に圧縮された 16 ビットの PCM オーディオデータでは、

オーディオメモリの 1 バイトに、4 ビットずつ 2 個のサンプルデータを格納することができます。電話品質の 8 kHz のサンプリングレートでは、

8000 samples/sec * ½ bytes/sample = 4000 bytes/sec

つまり、8 kHz のサンプルレートでは、1 秒のデータで 4 kB のメモリが必要となります。

メモリ容量が 1 MB の場合、理論上は

1000 kB / 4 kB/s = 250 seconds

つまり、8 kHz のサンプルレートでは 4 分以上のデータを保存できます。

6. ボードと MCU の考慮事項

6.1 ジャンパ設定

6.1.1 ボード上のスピーカからの出力

YRDKRX62N ボードはオーディオの再生を確認できるよう、便宜的に小型のスピーカを備えています。オーディオの右チャネルからの信号をスピーカにつなぐために、2 ピンジャンパ JP7 が備わっています。JP7 は PCB の手前右側にあり、SPK ENA とマーキングされています。

- オーディオをスピーカで再生するには、JP7 の二つのピンをジャンパで接続します。
- スピーカからのオーディオ出力を行わない時には、JP7 のジャンパを除きます。

【注】 1. **Rev.5 以降のボードには、ジャンパ** JP17 **が追加されており、**AUD_R **信号をスピーカのアンプに接続します。デフォルト（出荷時）はジャンパ** JP17 **のピン** 1-2 **は** PCB **の裏側でパターン接続されています。これは、このアプリケーションのためには妥当な接続です。もし** JP17 **のパターンが他のアプリケーションのために修正されている場合には、ピン** 1-2 **の間を（ジャンパプラグで）接続してください。**

2. **かすかな音量の音がボード上のスピーカから出力されます。これは近くで耳をそばだてて聞こえる程度です。**

6.1.2 ステレオ出力ジャック

YRDKRX62N ボードには 2 チャネルのオーディオアンプが備わっており、ボード上のステレオ出力ジャックに接続したステレオヘッドフォンを使用して、良質の音を聞くことができます。別の方法として、プリアンプを通したオーディオ信号をステレオオーディオケーブルで外部アンプに接続すると、より大きな音量を得られます。このとき、オーディオ出力は右チャネルからのものとなります。

Rev 5 以降のボードには 3 個のジャンパーJP17（AUD_R/SPK SEL）、JP18（AUD_R/AUDIO SELECT）、および JP19（AUD_L/AUDIO SELECT）が備わっています。これらはいずれも 3 本のピン（ピン 1-3）を持つジャンパで、デフォルト（出荷時）はピン 1-2 が PCB の裏側でパターンによりショートされています。これはこのアプリケーションでは適切な接続です。もし、他の目的のためにこのパターンが切断されているときには、JP17 から JP19 のピン 1-2 側にプラグをさし、これをショートしてください。（この際、3 ピンのヘッダを実装することが必要になるかもしれません。）

6.2 録音のためのボードの修正

REV 5 より前の YRDKRX62N ボードでは、録音データを RAM に書き込むためには若干のボードの手直しが必要です。フラッシュメモリからオーディオの再生のみを行う場合、この手直しは不要です。

手直し個所はボードの右下、ポテンションメータとスピーカの間にあります。手直しすべき個所は 2 個所で、次の回路図に示されています。

(a) **手直し個所** 1

- 未使用パッド R69 に 15K の抵抗を追加します。
- 未使用パッド R70 に 4.71K の抵抗を追加します。

(b) **手直し個所** 2

- ADC を中点にセットするため、下図のように、3V3A からの 100K のプルアップ抵抗を追加します。

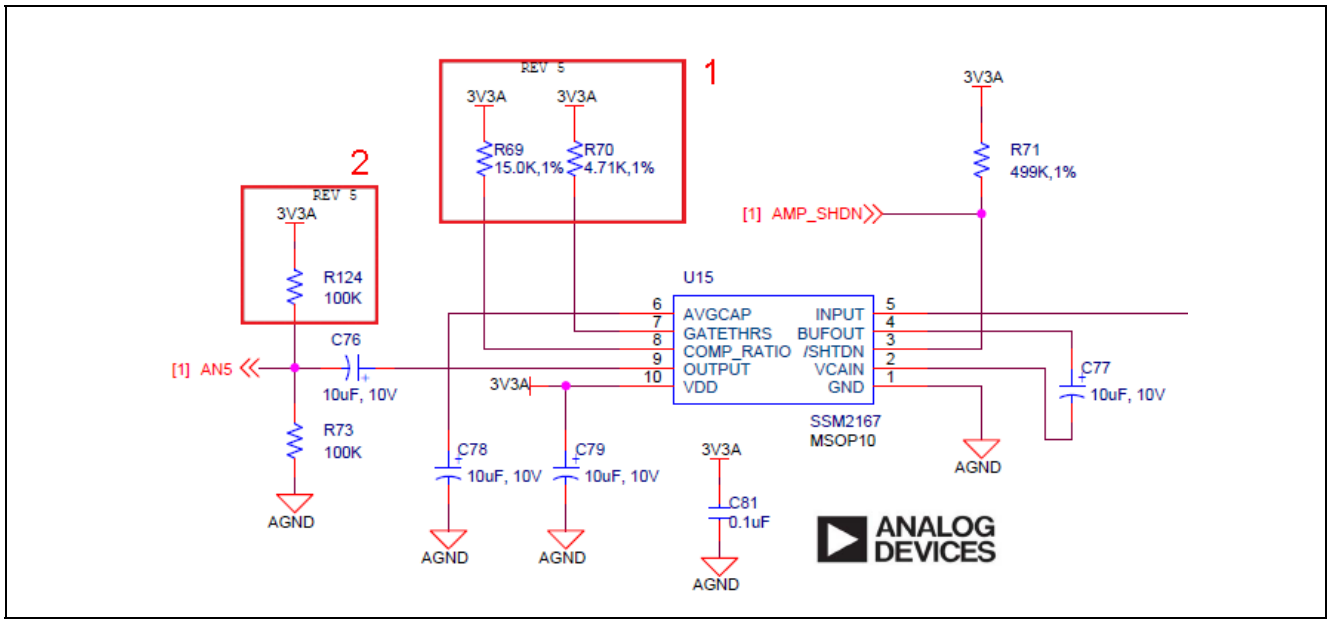


図3　RDK の録音回路を使用するために、この図に示されているように手直しを行うことが必要です。赤枠の中が手直し個所 1 と 2 に相当します。

6.3　MCU タイマの使用

使用するタイマを、次のサンプルコードのように syscfg.h ファイルで指定することができます。

```
/* MTU channel used for recording sample rate timing.     */
#define RECORD_SAMPLE_MTU      6

/* MTU channels used for carrier and sample rate timing.  */
#define SAMPLE_MTU             7
#define CARRIER_MTU            8
```

オーディオの右チャネルは MTIOC8A ピンに接続します。このピンには PWM モード 1 で動作しているタイマチャネル CARRIER_MTU の PWM 信号が出力されます。タイマジェネラルレジスタ TGRA が PWM の周期（PWM キャリア周波数）を、タイマジェネラルレジスタ TGRB がデューティ比（出力電圧）を制御します。

オーディオの左チャネルは MTIOC8B ピンに接続します。これは PWM モード 2 にのみ設定可能で、デモソフトではこの設定は行われていません。

「10.　付録 A」に記されたもうひとつのタイマに関するソフトウェア設定では、他の目的に DTC を使用できるよう、これを開放し MCU のリソースを節約できます。この方法には、左右両チャネル（ステレオ）で PWM モード 2 の MTU タイマを共有するよう設定することができる利点もあります。

6.4　DAC

オーディオの再生に、PWM ではなく DAC を利用することもできます。これに関しても「10.　付録 A」に記載されています。

7. 録音・再生のデモソフト

RX_RDK_ADPCM_Demo ワークスペースにある ADPCM_Demo プロジェクトでは録音・再生プログラムの例が提供されています。録音では、アナログ信号を集めて RX62N プロセッサ周辺回路の 12 ビット A/D コンバータに渡すために Analog Devices ADMP401 無指向性マイクが使用されます。信号は 11.025 kHz のレートでサンプルされて PCM データとなり、ADPCM 形式に圧縮されます。

再生動作は、PWM のハードウェアの制御を行うことが必要なため、多少複雑なプロセスとなります。再生動作は ADPCM データのセグメントをデコードして DTC（Data Transfer Controller）周辺回路への入力となる配列に格納することで開始されます。DTC は PCM データを PWM タイマに転送し、PWM タイマのデューティサイクルを変更します。DTC は定められた 11.025 kHz のサンプルレートで PWM タイマの設定を更新します。処理中の PCM データセグメントの転送が終わると、プロセッサに割り込みが入り、DTC にはデコードされた次のデータセグメントが渡されます。オーディオデータ全体の処理が終わるまで、このプロセスが繰り返されます。

7.1 デモソフトの状態遷移

次の図は、録音と再生の処理を管理するためにデモソフトに組み込まれている単純なステートマシンの状態を示しています。

ここでは 6 状態間の遷移が行われます。

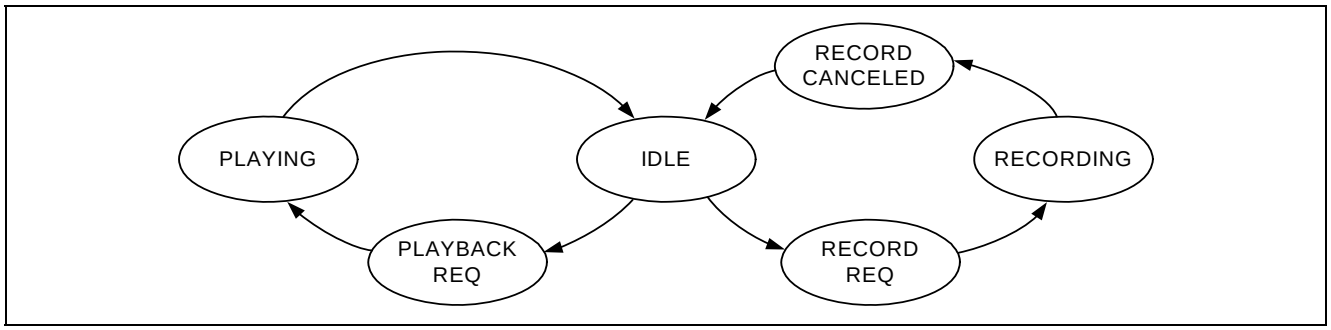


図4　ADPCM デモソフト状態遷移

SOUND_IDLE は ADPCM データの処理が行われておらず、ボタンなどが押されていないときのアイドル状態です。この状態には SOUND_PLAYING および SOUND_RECODING_CANCELLED のいずれかからのみ遷移します。

SOUND_PLAYBACK_REQUESTED はボード上の SW1 スイッチが押されたときに入る状態で、SOUND_IDLE からのみ遷移します。

SOUND_PLAYING は再生動作が開始されたときに入る状態で、再生動作中はここにとどまります。再生を中断する方法はありません。この状態は SOUND_PLAYBACK_REQUESTED からのみ遷移します。

SOUND_RECORD_REQUESTED 状態には SW2 スイッチが押されたときに入ります。この状態には SOUND_IDLE からのみ遷移します。

SOUND_RECORDING は録音動作が開始されたときに入る状態です。ステートマシンは、SW2 スイッチが押され、録音バッファが一杯になるまでの間、この状態にとどまります。この状態には SOUND_RECORD_REQUESTED からのみ遷移します。

SOUND_RECORDING_CANCELLED は実行中の録音処理が停止するときに入る状態です。SW2 スイッチを放した場合、または録音バッファが一杯になった場合にこの状態となります。この状態には SOUND_RECORDING からのみ遷移します。

7.2　再生音源の選択（RAM／フラッシュメモリ）

サンプルソフトでは、プログラム本体とリンクされフラッシュメモリ上に置かれている事前にエンコードされたデータと、このソフトの録音機能によって作られ一時的に RAM 上に置かれているオーディオデータのいずれも再生することができます。main_ADPCM.c（図 5 参照）の #if プリプロセッサ文で、ワークスペースに置かれておりリンクされてフラッシュメモリ上にあるオーディオデータを再生するか、RAM 上のオーディオデータを再生するかが選択されます。後者の RAM データの場合は、SW2 を押して録音を行いオーディオデータを作った後でのみ、再生が可能となります。

```
if( g_SrcFlag == PLAY_FLASH_FILE )
{
#if PLAY_FROM_RAM
    /* The following code will play back a recording saved in memory    */
    ADPCM_Playback( (uint8_t *)&g_EncodedData[0], fsize );
#else
     /* The following code will play the prerecorded clip built in flash */
    fsize = ADPCM_DATA_SIZE;
    ADPCM_Playback( (uint8_t *)ADPCM_ADDR, fsize );
#endif
}
```

図5　main_ADPCM.c の#if プリプロセッサ文によって、ワークスペースにありあらかじめリンクされフラッシュメモリ上に置かれているデータが再生音源として使用されるか、録音され RAM 上に置かれているデータが再生されるかを指定します。録音バッファは初期状態から何らかのデータが録音されるまで PLAY_FROM_RAM に 0 がセットされて再生可能なデータが存在しないことが示されます。

7.2.1　RAM からの再生動作

SW2 スイッチによる録音後には、RAM 上（g_EncodedData[] 内）にオーディオデータが置かれます。「7.2 再生音源の選択（RAM／フラッシュメモリ）」に記されているように、これを行うには PLAY_FROM_RAM が 1 にセットされていなければなりません。一時的に作られた ADPCM データを再生するには、まず SW2 を押して録音を行い、次に SW1 を押してこれを再生します。

7.2.2　フラッシュメモリからの再生動作

プログラムとリンクされたデータをフラッシュメモリから再生するには、main()を変更し、使用される ADPCM データのアドレスを g_EncodedData[]の代わりに ADPCM_ADDR に設定しなければなりません。

ADPCM_ADDR の値は ADPCM_audiodata ディレクトリにある adpcm_sample.dat ファイルで定義されています。このデモソフトでは、データの大きさは定数 ADPCM_DATA_SIZE としてソースコードに書き込まれています。

他のオーディオファイルを組み込むには、そのオーディオファイルを ADPCM_audiodata ディレクトリに置き、そのファイル名をビルド時に想定されているファイル名に変更するか、そのファイルがビルド時に取込まれるような変更を行わねばなりません。いずれの場合でも、定数 ADPCM_DATA_SIZE には組み込もうとするデータの大きさを設定することが必要です。

(1)　PC 上で新たな ADPCM ファイルを作るには

このパッケージでは新たな ADPCM ファイルを用意するために WAVE ファイルを ADPCM 形式にエンコードする ADPCM.exe という名前のツールが用意されています。このツールで組み込みたい WAVE ファイルを選択します。WAVE ファイルは次の仕様でなければなりません。

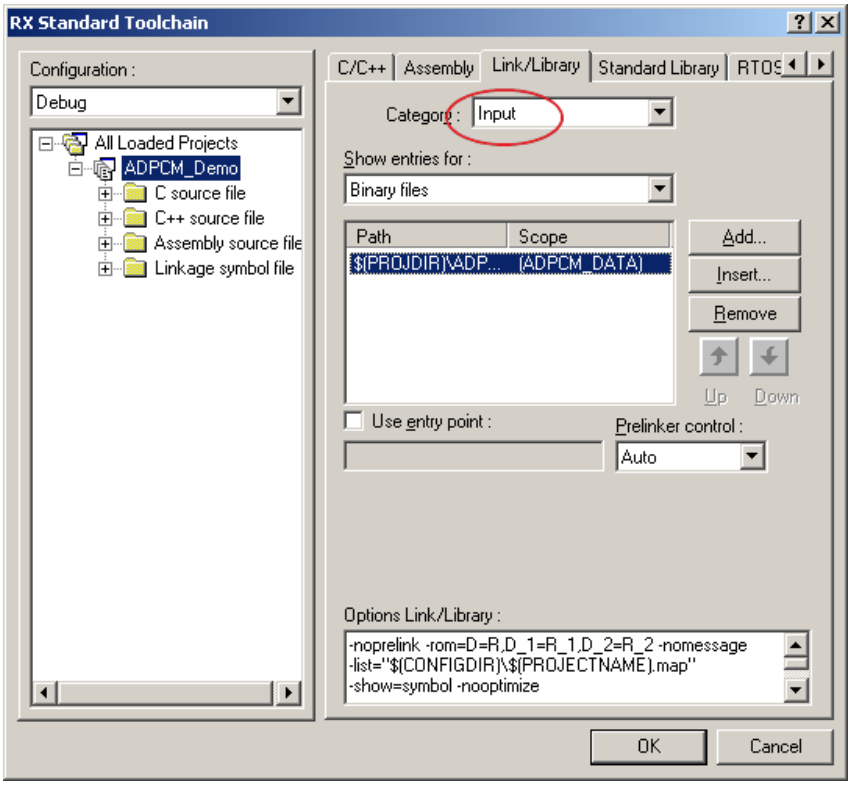
量子化ビット数:　16
チャネル数:　**モノラル**
サンプリング周波数:　11.025 kHz
ファイル形式:　WAV

ADPCM_TOOL¥bin ディレクトリにある ADPCM.exe ファイルをダブルクリックしツールをスタートさせます。ADPCM_TOOL¥doc ディレクトリにある PDF ファイルにファイルの作り方が説明されています。

(2)　フラッシュメモリに録音された ADPCM ファイルを組み込むには

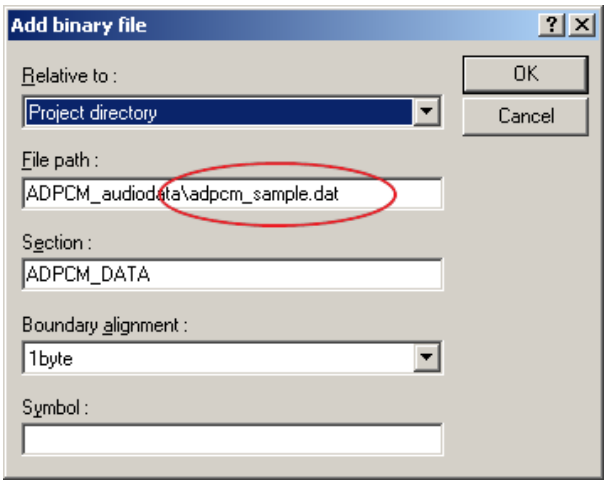
オーディオファイルを ADPCM 形式に変換した後、次のように、デモプログラムのビルドで ADPCM オーディオファイルの変更を行う必要があります。

まず「Build」を、次に「"RX Standard Toolchain…」を選択します。表示されたウィンドウの「Link/Library」タブをクリックします。



「Category」プルダウンメニューで「Input」を、次に「Show entries for」プルダウンメニューで「Binary files」をクリックします。

設定されているエントリを変更するには、これをダブルクリックします。（上図ではブルーで示されています）



次に、ファイルパスを組み込みたいファイルに変更し、「OK」を押します。

この変更を保存後、定数 ADPCM_DATA_SIZE を変更してあることを確認して、プロジェクトをリビルドします。

7.2.3　デコード処理の詳細（MTU と DTC)

SW1 が押されたことが検出されると、SwitchPressCallback()関数は状況に応じてステートマシンの状態を SOUND_PLAYBACK_REQUESTED に変更します。Main()関数はこれを受けて状態を SOUND_PLAYING に変更し、ADPCM ファイルを再生するために ADPCM_Playback()関数を呼び出します。

ADPCM_Playback()関数はデータファイルの一部をデコードし PWM 発生ルーチンに送る PCM データを入れた配列を用意します。この際にはフレキシブルな RX62N DTC 周辺回路が利用されます。DTC は固定長の PCM データ配列から PWM 信号を発生する MTU 8（CARRIER_MTU）に直接データを送るよう設定されます。

DTC の動作タイミングは MTU7（SAMPLE_MTU）によってサンプルレート（11.025 kHz）ごとに割り込みを発生するよう設定されます。MTU7 の割り込みが発生するごとに、DTC は次の PCM データを読出し、デューティ比を制御する MTU レジスタ（TGRB）に書き込みます。これらはすべてバックグランドで実行されるため、MCU はこの処理から解放され、アプリケーションで DTC に供給するデータの次の部分の準備を行うことができます。

DTC が使用中の配列内の最後のデータ要素を送り出すと、次の MTU7 割り込みが MCU により検出され、デコード済みのデータの次の部分が DTC に渡されます。この処理はデータがすべて再生されるまで続けられます。

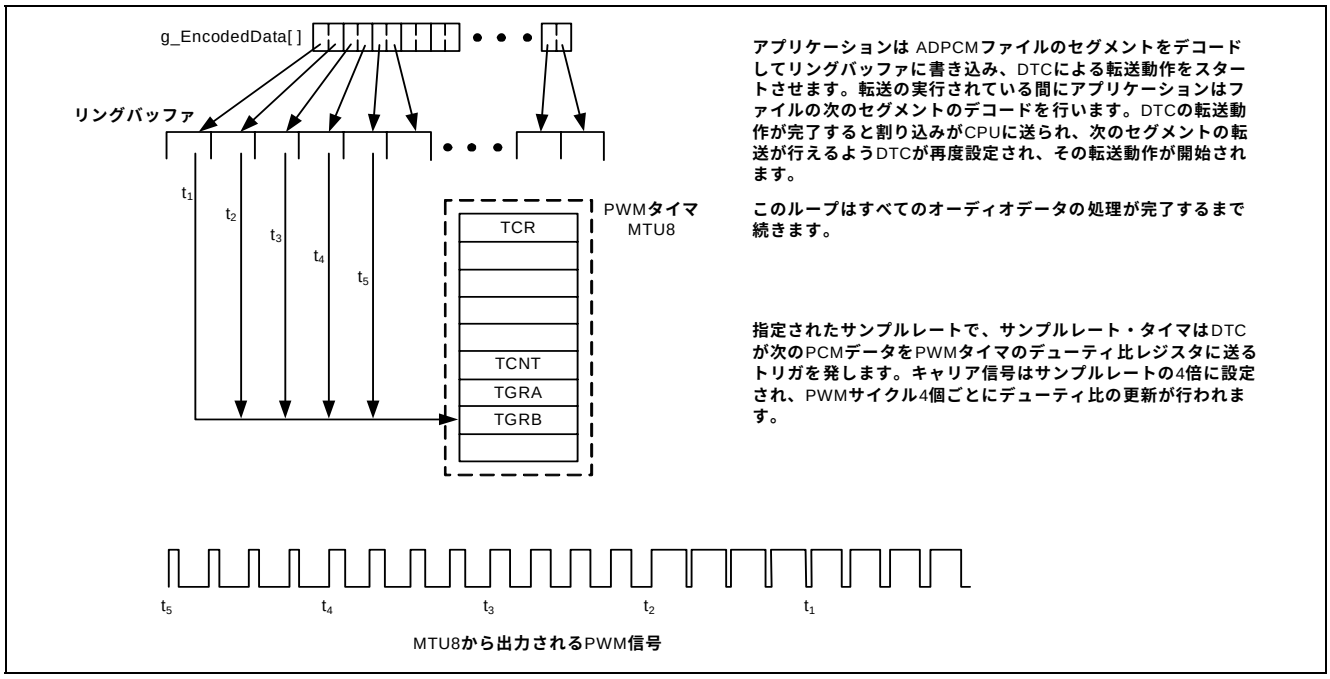


図6　再生動作の詳細

7.3　ボード上の RAM への録音

HEW ワークスペースを開き、「7.2　再生音源の選択（RAM／フラッシュメモリ）」で述べられているように PLAY_FROM_RAM に 1 をセットしてプロジェクトをビルドします。このイメージをロードし、「Reset/Go」を押します。LCD にはプロジェクト名のほか、録音で SW2 を使うための簡単な説明が表示されます。録音の準備ができた時点で、SW2 スイッチを押したままの状態で保持し、スピーカの横にあるマイクに向かって話し掛けます。録音が終わった時点で、SW2 を放します。なお、内部の録音用のメモリが一杯になったときにも録音動作は終了します。

録音動作が行われている間 LED14 が点灯し、録音が終わると消えます。

7.3.1　エンコード処理の詳細

SW2 スイッチが押されたことが検出されると、SwitchPressCallback()はステートマシンの状態を SOUND_RECORDING_REQUESTED とします。main()関数はこの要求を受け、ステートマシンの状態を SOUND_RECORDING に変更し、ADPCM_Record()を呼び出して、実際の録音動作を実行します。

録音時には、PWM 再生の場合と同じ MTU（MTU 8）が使われ、RPDL（「3.　RPDL」を参照）を利用してそのセットアップが行われます。この MTU は 11.025 kHz で、もしくは 91 us ごとに、プロセッサに割り込みます。MTU 割り込みが RPDL で生成され、ユーザが RPDL API に対して用意したコールバックルーチン、

デモソフトでは RecordSampleRateTimerCallback()をトリガします。コールバックがコントロールを得ると、12 ビット ADC 回路からデータを読み出し、値が 4 個揃うまでローカルな変数に格納します。これらは次に 16 ビットの PCM 値 4 個に変換され、グローバル配列 g_InputData[]に格納され、次に、圧縮すべきデータが揃ったことが ADPCM_Record()に通知されます。

ADPCM_Record()は通知を受け取ると ADPCM への変換を実行するために EncodeData()を呼び出します。これは ADPCM ライブラリの関数を呼び出すことによって行われます。

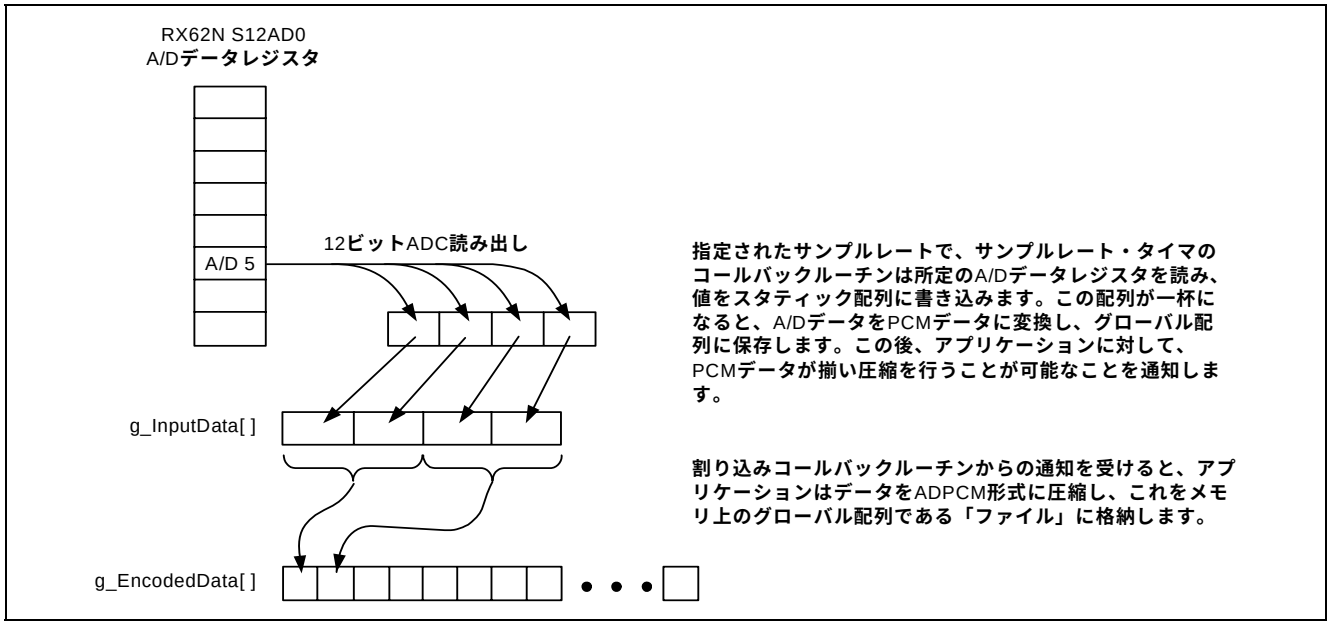


図7　データ配列 g_EncodedData[…]

ADPCM にエンコードされたデータは 27,563（0x6BAB）バイトの配列 g_EncodedData[]に書き込まれます。このサイズは 2*27,563=55,126 個の ADPCM データに相当しており、11.025 kHz では 5.00 秒間の録音ができます。録音時間を延長するには、定数 MAX_DATA_LENGTH を変更します。

7.4　ADPCM エンコード／デコードライブラリの選択

前にも述べられているように、3 種の ADPCM ライブラリが提供されています。

- デコードのみ
- エンコードのみ
- エンコードとデコードの両機能

デモソフトにはこのライブラリのいずれか 1 種がリンクされます。デフォルトは両機能（エンコードとデコード）をもつライブラリがリンクされています。

ライブラリを変更するには、まず「Build」を、次に「RX Standard Toolchain…」を選択し、表示されたウィンドウで「Link/Library」をクリックします。

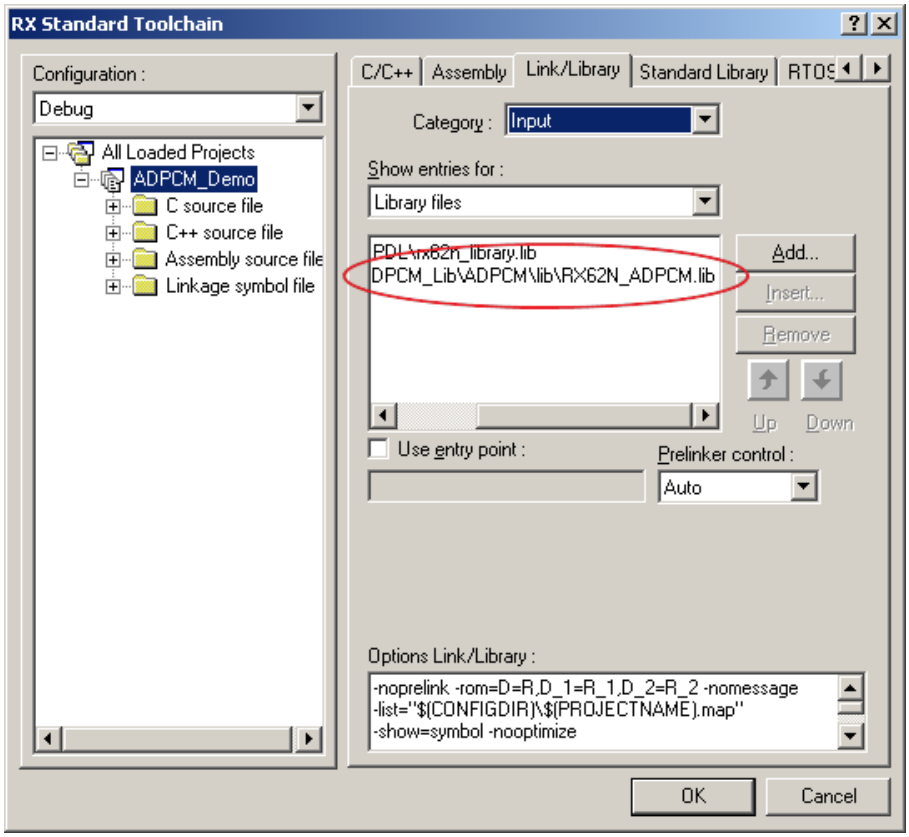


図8　「Category」プルダウンメニューで「Input」を選択し、次に「Show entries for」プルダウンメニューから「Library files」を選択します。上図の赤の円がついた項目をダブルクリックします。

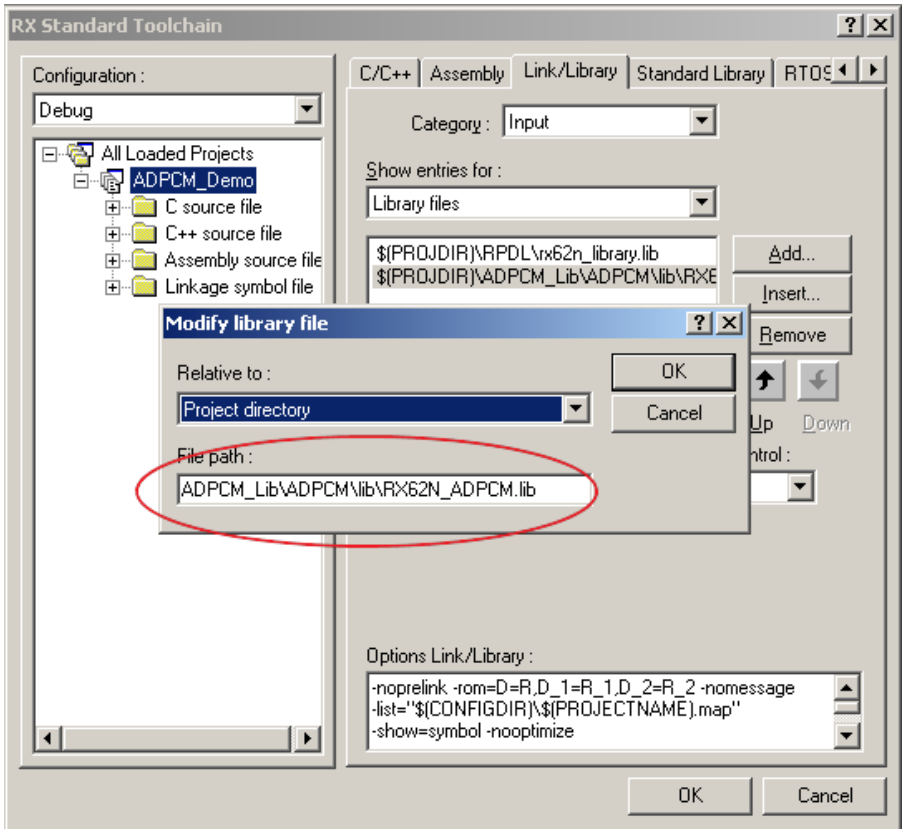


図9　「Modify library file」ポップアップでデモソフトにリンクされるライブラリのファイルパスおよびファイル名を変更します。

8. 「ADPCM_Lib」ワークスペース

あらかじめコンパイルされているライブラリ

ワークスペースではデモソフトとともに独自の ADPCM ライブラリを構築することができます。前にも述べられていますが、必要に応じてカスタムビルドを行うことができる 3 プロジェクトが用意されています。しかし、これらのライブラリは既にビルドを済ませた状態で出荷されます。

これらのライブラリの名前および場所は以下のとおりです。

- RX_RDK_ADPCM_Demo¥ADPCM_Lib¥ADCPM¥lib¥RX62N_ADCPM.lib
- RX_RDK_ADPCM_Demo¥ADPCM_Lib¥ADCPM_Decode¥lib¥RX62N_ADCPM_Decode.lib
- RX_RDK_ADPCM_Demo¥ADPCM_Lib¥ADCPM_Encode¥lib¥RX62N_ADCPM_Encode.lib

ADPCM_Demo プロジェクトは、それぞれがビルドされたディレクトリにあるビルド済みの RX62N_ADPCM.lib と直接にリンクされるよう設定されています。

ADPCM ソースコード

異なる ADPCM ライブラリを構築するためのソースコードは ADPCM_Lib¥src に置かれています。

核となる関数はアセンブラで記述されており、次のファイル内に存在します。

adpcm_encoder.src	圧縮ルーチン
adpcm_decoder.src	展開ルーチン
adpcm_table.src	圧縮と展開に共通して使用されるステップや表

ここには圧縮アルゴリズムの現時点でのバージョンに依存した version.c というファイルも置かれています。

8.1 エンコードライブラリの関数

8.1.1 初期化（R_apdcm_initEnc)

プロトタイプ

```
void R_adpcm_initEnc(adpcm_env *wenv);
```

説明

この関数は、圧縮されるサウンドデータを初期化します。

この関数は一連の ADPCM データを圧縮する前に、初期化を行うために一度だけ呼び出されます。

adpcm_env 構造体型はヘッダファイル r_s2_lib.h で宣言されています。この関数を実行する前に、引数として渡される adpcm_env 型の構造体変数が用意（宣言）されていなければなりません。この関数が実行される前に、値が引数に設定されている必要はありません。

戻り値

なし

8.1.2 圧縮（R_adpcm_encode)

プロトタイプ

```
int16_t R_adpcm_encode(int16_t smpln, adpcm_env *wenv);
```

説明

この関数は 16 ビット PCM データから 4 ビット ADPCM データへの変換（圧縮）を行います。エンコードされたデータは、関数 R_adpcm_initEnc で初期化されたメモリ領域内に格納されます。引数 smpln はエンコードされた ADPCM データが格納されるビット数を指定するもので、4 の倍数でなければなりません。メモリ上の出力される場所の大きさは smpln の値に依存します。

戻り値

int32_t - smpln の値が 4 の倍数でないときには-1 が戻され、その他は 0 が戻されます。

8.1.3　リフレッシュ（R_adpcm_refreshEnc)

プロトタイプ

```
void R_adpcm_refreshEnc(int16_t *inputAddr, uint8_t *outputAddr,
                        adpcm_env *wenv);
```

説明

この関数は内部の PCM 入力データバッファと ADPCM 出力データバッファをリフレッシュします。引数 inputAddr は入力データが格納されているメモリ領域を、引数 outputAddr は ADPCM データが格納されるメモリ領域を指定します。この関数と関数 R_adpcm_encode が続けて呼び出されると、R_adpcm_encode は inputAddr で示されるメモリ領域の先頭からデータを読み込んでエンコードを行い、エンコードされたデータを outputAddr で示されるメモリ領域の先頭から格納します。

表 1　初期化関数の引数

引数	型	説明
inputAddr	int16_t *	PCM データ領域の先頭アドレス
outputAddr	uint8_t *	ADPCM データ（圧縮されたデータ）領域の先頭アドレス
wenv	adpcm_env *	エンコード管理構造体へのポインタ

8.2　デコードライブラリの関数

8.2.1　初期化（R_adpcm_initDec)

プロトタイプ

```
void R_adpcm_initDec(adpcm_env *wenv);
```

説明

この関数は、展開されるサウンドデータを初期化します。

この関数は一連の ADPCM データを展開する前に、一度だけ実行されます。

adpcm_env 構造体型はヘッダファイル r_s2_lib.h で宣言されています。この関数を実行する前に、引数として渡される adpcm_env 型の構造体変数が用意（宣言）されていなければなりません。この関数が実行される前に、値が引数に設定されている必要はありません。

戻り値

なし

8.2.2　展開（R_adpcm_decode）

プロトタイプ

```
int16_t R_adpcm_decode( int16_t smpln, adpcm_env *wenv );
```

説明

この関数はエンコードされた 4 ビット ADPCM データをデコードし、16 ビット PCM データを生成します。デコードされたデータは関数 R_adpcm_refreshDec で初期化されたメモリ領域に格納されます。1 回の関数呼び出しでデコードされる 16 ビットのデータサンプルの数は smpln で指定されます。この値は 2 の倍数でなければなりません。

出力サンプルバッファ領域は、この関数を呼び出す前に宣言されていなければなりません。このバッファ領域の大きさは smpln の値に依存し、smpln ×16 ビットの出力メモリが必要とされます。

たとえば、サイズ（smpln）の値が 16 であれば、出力サンプルバッファ領域を用意する際に int16_t buffer[16] と定義します。

戻り値

変換される ADPCM データの数が奇数の場合には-1 が戻され、その他は 0 が戻されます。

8.2.3　リフレッシュ（R_adpcm_refreshDec)

プロトタイプ

```
void R_adpcm_refreshDec(uint8_t *inputAddr, int16_t *outputAddr,
adpcm_env *wenv);
```

説明

この関数は内部の ADPCM 入力データバッファと PCM 出力データバッファをリフレッシュします。引数 inputAddr は入力データが格納されているメモリ領域を、引数 outputAddr は PCM データが格納されるメモリ領域を指定します。この関数と関数 R_adpcm_decode が続けて呼び出されると、R_adpcm_decode は inputAddr で示されるメモリ領域の先頭からデータを読み込んでデコードを行い、デコードされたデータを outputAddr で示されるメモリ領域の先頭から格納します。

戻り値

なし

9.　ファイル単位のオーディオ形式変換 GUI

Windows の GUI ツールである ADPCM.exe が..¥R8CSPB_AN¥tools¥ADPCM-tool フォルダに用意されています。これには、ダウンロードされたこのアプリケーションノートも含まれています。このツールでは WAV ファイルから、エンコードされた ADPCM 形式のファイルを 1 個ずつ生成できます。また ADPCM ファイルをデコードし、WAV ファイルを作ることもできます。ADPCM-Tool のマニュアルをご覧ください。

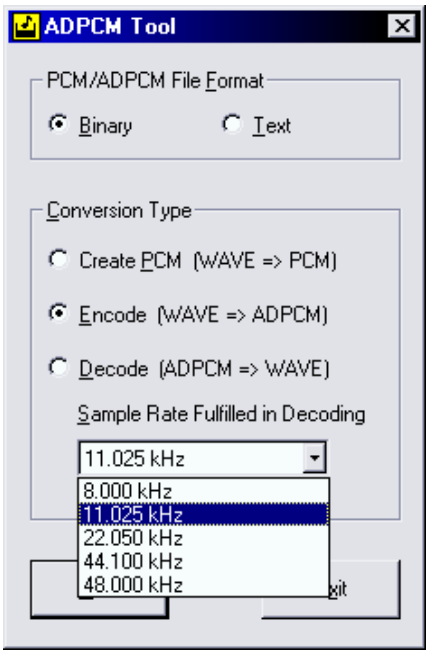


図10　ADPCM ツールは ADPCM 形式ファイルと WAV 形式ファイルの間の両方向の変換を行う GUI です。変換したいファイルの形式とサンプリング周波数を選択した後に、「Go」をクリックすることでファイルの変換が行われます。

ADPCM ツール GUI は、一度に 1 個のオーディオクリップの変換を行います。

他にもいろいろなオーディオ変換ツールをオンラインで入手することができますが、ADPCM 形式に相違があるため、必ずしも互換性があるとは限りません。

10. 付録 A
DTC の代わりとなる MTU 割り込み ISR - PWM モード 2 - ステレオ

REV 5 以降の YRDKRX62N ボードでは、もうひとつの、タイマをソフトウェアで設定する方法をとり、DTC を他の用途で利用できるようにすることで MCU のリソースを節約することができます。

標準のコードでは、MTU8 のコンペア・マッチ割り込み要求信号 TGIA8A が DTC をトリガするために使用されています。DTC はサンプリングレートでオーディオデータを PWM 出力タイマレジスタ（MTU8.TGRB）に書き込んでいます。DTC をトリガする代わりに、この TGIA8A 信号の割り込みサービスルーチンを利用することができます。加えて、DTC を使用しないことで、この方法では左右両チャネル（ステレオ）で同じ MTU タイマの設定と PWM モード（モード 2）を利用できる利点もあります。これにより両チャネルを使用する場合に、コードをより単純なものとすることができます。

PWM 周期の途中でではなく、各 PWM カウント周期のはじめで PWM 出力タイマレジスタに新たな値を書き込むために、タイマレジスタのバッファモードを使用することが推奨されます。周期の途中の書き込みでは、出力に聴取可能な障害が生じることがあります。

PWM デューティ比のために MTU6 と、PWM 周期のためにもうひとつのタイマ MTUx を使用する場合、タイマをどのようにセットアップするかを次に示します。

- TMDR レジスタをバッファモードに設定します。これで、TGRC が TGRA の、TGRD が EGRB のバッファとして使用されます。
- TMDR レジスタを PWM モード 2 に設定します。
- TBTM をバッファモードとコンペア・マッチモードにセットします。

PWM デューティ比：
- 左右両チャネルのデューティ比はバッファレジスタ MTU6.TGRC と MTU6.TGRD にセットされます。

PWM 周期：
- TSYR を同期動作にセットします。この設定で MTU6 ともうひとつの MTU、MTUx で TCNT が同期動作をします。
- いずれかの TGR（たとえば MTUx.TGRA）を使用して、MTUx が TCNT をクリアするようセットします。

この方法を行うためには、ジャンパ設定を変更しなければなりません。これは Rev 5 以降の YRDKRX62N ボードでのみ可能です。ジャンパ JP18 と JP19 が、MTU8 出力ピンの MTIOC8A および MTIOC8B ではなく、MTU6 出力ピンの MTIOC6A および MTIOC6B を使用するようセットされなければなりません。

11. 付録 B
PWM の代わりに DAC を使用するには

タイマによる PWM 信号を使う代わりに、チップ上の DAC を利用することもできます。

Rev. 5 以降の YRDKRX62N ボードを使用している場合にはボード上に JP17 の場所が確保されており、これでオーディオ右チャネルを PWM ではなく DA1 ピンに接続するよう設定します。DAC 出力を YRDKRX62N ボード上のスピーカに接続するには、JP17 のピン 1 とピン 2 をショートしている線を切断しなければなりません。DA のオーディオ出力は JP17 のピン 3 に出ていますので、これをピン 2 と接続し、DAC 出力をスピーカに接続します。3 ピンのジャンパヘッダを JP7 位置に取り付けることで、PWM と DAC のオーディオ出力の切り替えが容易に行えます。

DAC は PCM の右チャネルのデータストリームから抽出された同じデータを使用します。このため DAC のオーディオデータバッファのポインタは PWM 出力に与えるのと同じデータを指していることが必要です。（ただし、値のスケールを変える必要があるかもしれません。）オーディオデータは通常 16 ビットですので、DAC に合せて 10 ビットにスケール変更を行わないと、オーディオの振幅が大きな場合 DAC の変換レンジを越える可能性があります。この場合でもデータの値が 10 ビット以下であれば、オーディオ再生は正常に行われます。

DTC の初期化は同じように設定しますが、転送先アドレス（DAR）は DAC のデータレジスタとしておかねばなりません。

```
/* Define location where DTC is to write DUTY cycle values. */
#define DTC_DESTINATION (uint16_t *)&DA.DADR1
```

以下は DA 初期化の一例です。

```
/***********************************************************************
Function Name  : R_InitDA
* Declaration  : void R_InitDA(void)
* Description  : This function initializes DAC1
* Arguments    : none
* Return value : none
***********************************************************************/
static void R_InitDA( void )
{
  PORT0.DDR.BIT.B5 = 0;
  PORT0.ICR.BIT.B5 = 0;

  /* CancelD/AConverter Module Stop state. */
  SYSTEM.MSTPCRA.BIT.MSTPA19 = 0;

  /* InitD/AData Register. */
  DA.DADR1 = 0x0020;

  /*D/AControl Register. */
  DA.DACR.BYTE = 0x9F;   /* b7 DAOE1 -D/AOutput Enable 1 - enabled.
                            b6 DAOE0 -D/AOutput Enable 0 - disabled.
                            b5 DAE -D/AEnable 0 - independent on each channel
                            b4:b0 Reserved - Set to 1.*/

  /*D/AData Placement Register. */
  DA.DADPR.BYTE = 0x00;  /* b7 DPSEL - Data Placement Select aligned to LSB.
                            b6:b0 Reserved - Set to 0.*/
}
```

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/inquiry

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2011.06.07	—	初版発行
1.01	2011.07.25	9、10、12、14	6.1、6.3、7.2.1、7.3 を変更
1.02	2011.08.10	9	6.1、6.3 を変更

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部 ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

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。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル） (03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/inquiry

Colophon 1.0