REX-5054U/B

A/D CONVERSION PC CARD

ユーザーズマニュアル

2002 年 2 月

第 6.0 版

RATOC Systems, Inc.
ラトックシステム株式会社

※ MS-DOS/Windows3.1でご利用のお客様は、製品添付FDの別冊マニュアル(PDF ファイル)をご参照ください。

第1章 はじめに

1-1. A/D カード概要仕様

REX-5054U･REX-5054B は DOS/V,PC-98 マルチ対応の PC カードタイプの A/D コンバータです。また、本製品は PC カードの標準規格である PCMCIA Release 2.1 / JEIDA 4.2 に対応した仕様になっており、PCMCIA Standard に準拠したカードスロットを有するパソコンで使用することができます。

本製品の主な仕様は下表の通りです。

	REX-5054U	REX-5054B
ADC コントローラ	National Semiconductor LM12H458	
入力チャンネル数	8 チャンネル	4 チャンネル
入力電圧範囲	0 ～ +2.5V	-5 ～ +5V
最高サンプリング周波数	50KHz(20μsec 間隔）/1 チャンネル時	
分解能	12 ビット	
変換方式	逐次変換方式	
I/O アドレス	任意のアドレスから連続 8 バイト	
割り込み番号	任意	
外形寸法	54mm(W)×85.6mm(D)×5mm(H)(TYPEⅡ)	
重量	28ｇ（ケーブルを除く）	
動作環境	温度 0～55℃、湿度 10～80%（ただし結露しないこと）	

本製品には、プログラム言語 Microsoft Visual C 16 ビットバージョン・32 ビットバージョンならびに Microsoft Visual BASIC16 ビットバージョン・32 ビットバージョン対応のライブラリとサンプルプログラムが添付されています。MS-DOS/Windows3.1、Windows95/98/Me/2000/XP 環境でのプログラム開発を容易に進めることができます。また、A/D ライブラリを使用しないで、カードのレジスタを自分で制御して変換プログラムを開発できるように、内部の詳細仕様について第４章で詳しく解説してありますので必要な方は参照して下さい。

1-2. 添付品

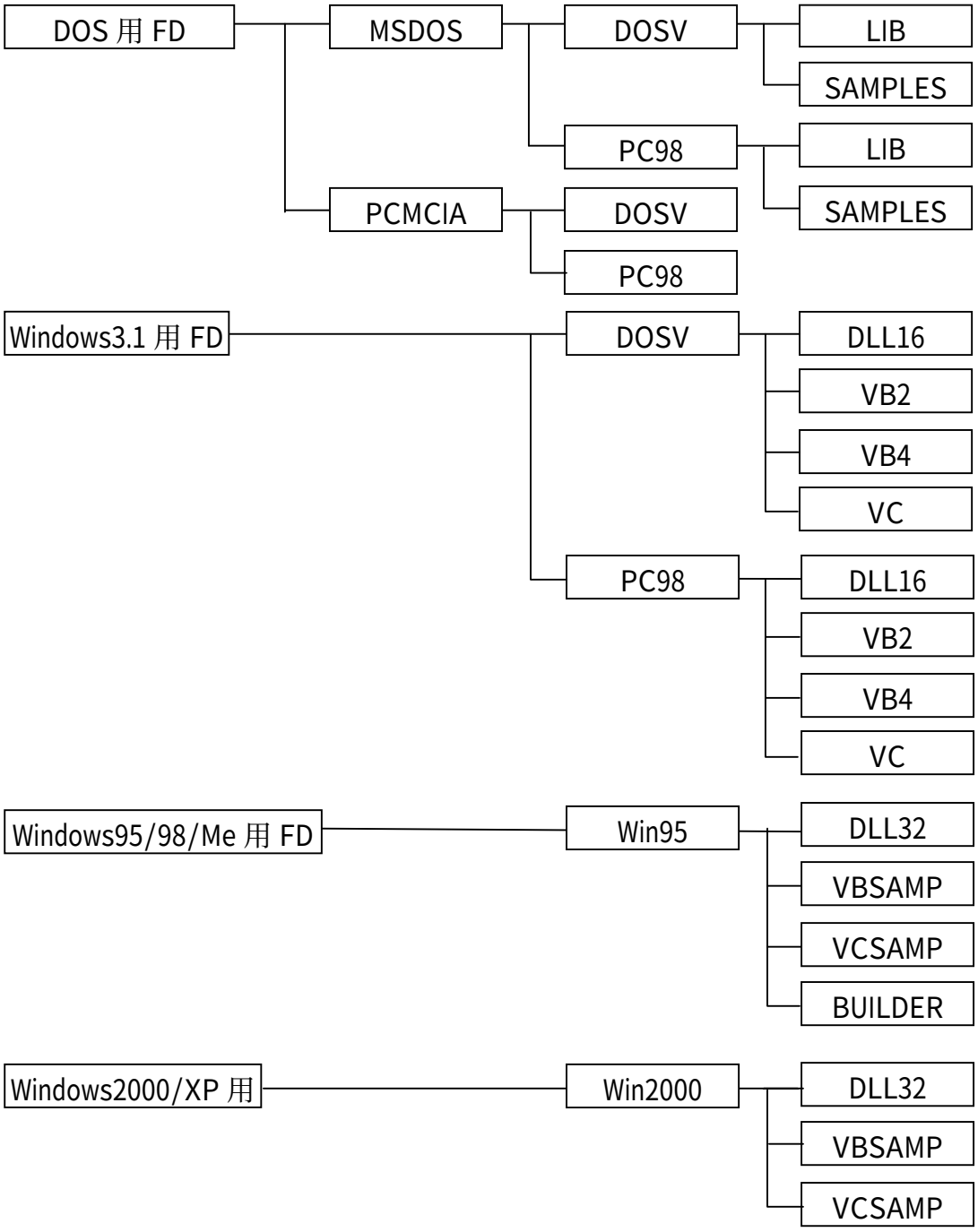
REX-5054 は PCMCIA スロットを持つパーソナルコンピュータ(DOS/V,PC-9800 シリーズ等)のための A/D PC カードで下記の製品より構成されています。開梱後、欠品がある場合には、すぐに御連絡ください。

- **REX-5054U/B A/D PCカード** 1枚
  パーソナルコンピュータ本体内の PCMCIA スロットに実装します。
- **添付ソフトウェア書き込み済ディスク (3.5", 1.44MB)** 4枚
  ライブラリの基本ソフトウェア、および応用プログラム例が書き込まれています。
- **REX-5054U/B ユーザーズマニュアル** 1冊
  本書のことです。REX-5054 を使用する上で必要な事項について述べてあります。
- **BNC ケーブル (50cm 長)** 1本
- **保証書兼ご愛用者登録ハガキ** 1枚

> ☞ご愛用者カードは保証書を切り離した後、必要事項を記入の上必ずご返送ください。ご返送頂けない場合、バージョンアップ等のサポートサービスは受けられませんのでご注意ください。

1-2-1. 添付フロッピーディスクライブラリ構成

AD ライブラリは、MS-DOS/Windows3.1、Windows95/98/Me/2000/XP 上での Microsoft C (Visual C++) および Visual BASIC によるアプリケーション開発を支援するためのライブラリです。添付フロッピーディスク (FD) の構成は下記のようになっています。



★★★ ご注意 ★★★

A/D カードご使用前に、添付フロッピーディスクのルートディレクトリーにある "README.TXT"ファイルをご一読下さい。最新の追加情報が記載されています。

1-3. A/D 変換入門

1-3-1. A/D 変換

時間と振幅が連続的に変化するアナログ信号を、コンピュータで処理可能なデジタル信号に離散化することを A/D 変換と言います。更に、アナログ信号の振幅量を離散化することを量子化、時間軸の離散化をサンプリング（標本化）と言います。

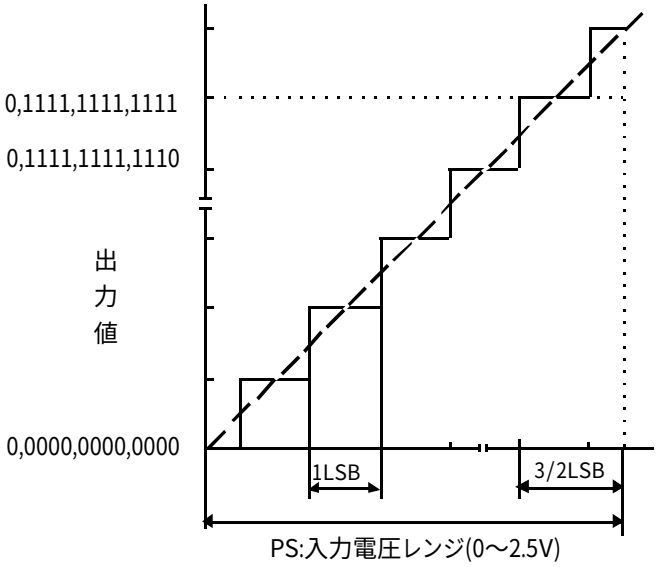
センサー等からの信号を A/D カードに入力する場合、出力電圧が A/D カードの入力レンジに較べて著しく小さい場合（大きい場合）があります。このような場合は、A/D カードに入力する前処理として信号の増幅（または減衰）が必要になります。

●量子化

REX5054U は入力電圧レンジが 0-2.5Volt、分解能 12Bit 分の A/D コンバータです。
従って、1LSB は次式で求められます。

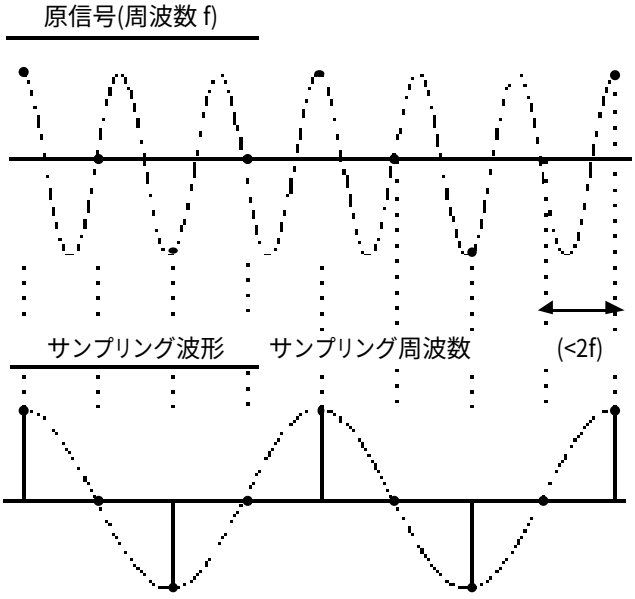
$$1LSB = \frac{2.5}{2^{12}}$$

これが入力電圧を量子化した時の誤差になります。



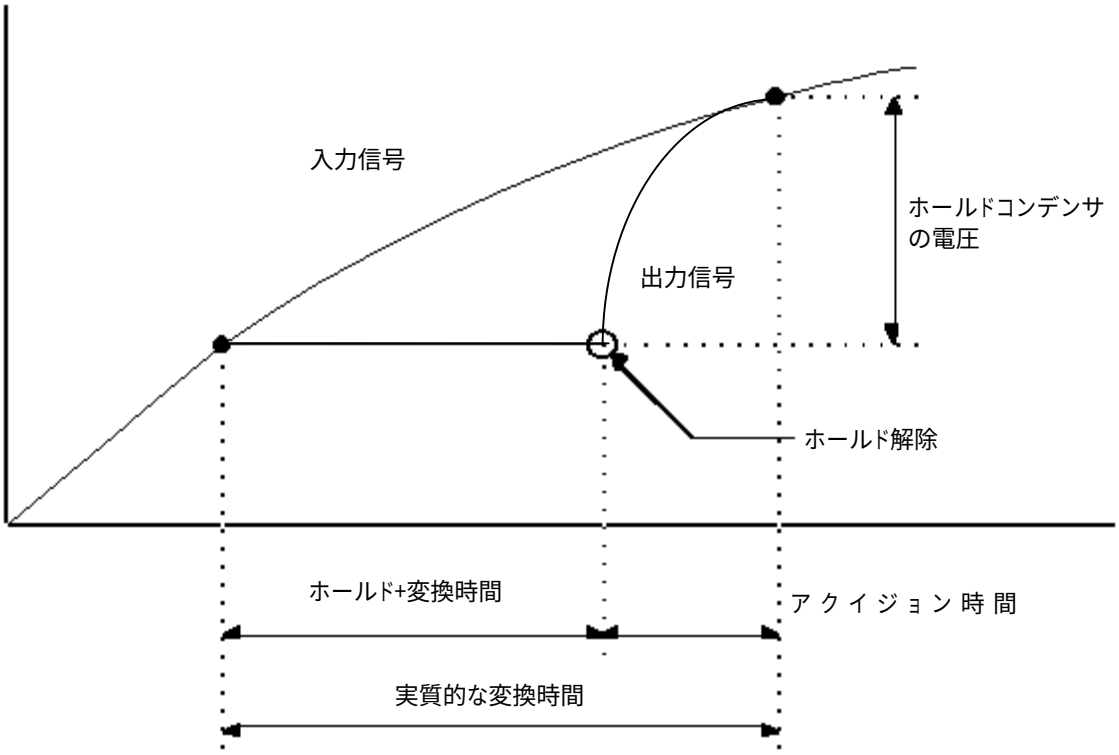
●サンプリング

原信号の持つ最高周波数（f Hz）の信号成分を失わないでサンプリングするためには、サンプリング周波数は 2f Hz に設定しなければなりません。これをサンプリングの定理と言います。右図のように、2f Hz 以下でサンプリングを行うと、サンプリング波形に原信号に存在しない周波数成分が現れてしまいます。一般に、このような現象をエイリアシングと言います。



1-3-2. サンプリング＆ホールド (S&H)

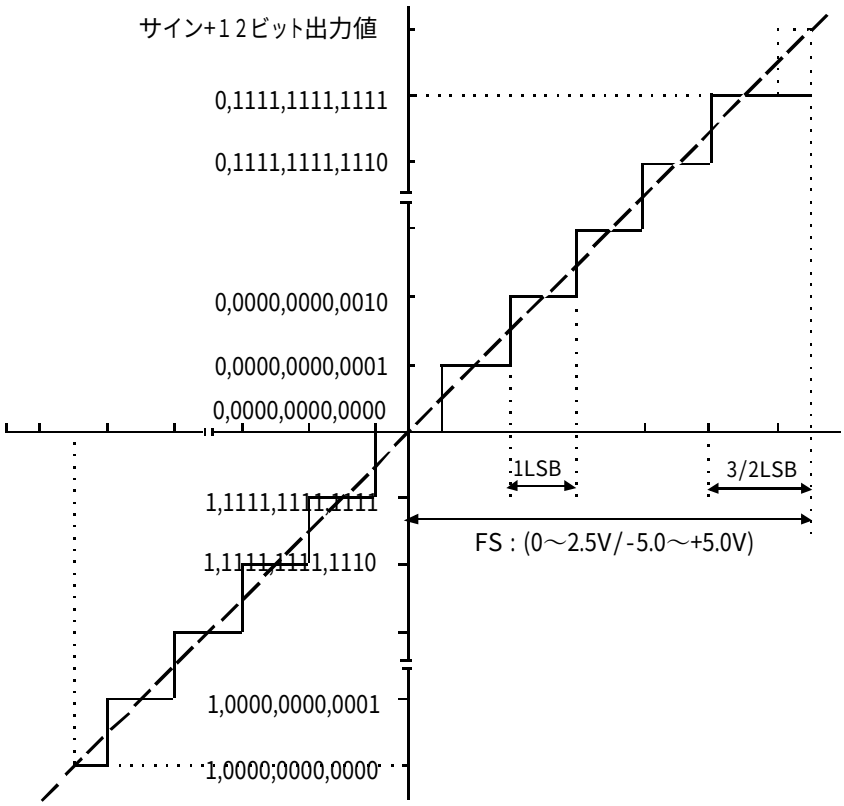
A/D 変換を行うには、変換中に入力信号が変動しても A/D コンバータへの出力信号が変動しないようにするために、入力信号をサンプリングしてホールドする必要があります。これを行うのが S&H 回路で、LM12458 に内蔵されています。



上図に示すように、実質的に A/D 変換にかかる時間はホールドに必要な時間と A/D 変換時間に加え、アクイジョン時間を加算した時間になります。アクイジョン時間は、変換が完了してホールドを解除した後、次の変換を行うためにホールドコンデンサーを入力信号の電圧まで充電するのに必要な時間を言います。

1-3-3. 出力コード

本製品の出力コードはコンプリメンタリバイナリコード仕様になっています。REX5054U 場合は入力レンジ FS (フルスケール) : 0～+2.5V が 0-4095 で出力され、REX5054B の場合は入力レンジ FS : -5.0～+5.0V が 0-4095 で出力されます。



A/D 変換データはコンプリメンタリバイナリコードで出力されますので、入力電圧値は下記のように算出することができます。

●REX-5054U の場合

```
Volt = (double)( ( AdVal   & 0x1FFF ) <<3 ) * 2.5 / 32768.0;
```

●REX-5054B の場合

```
Volt = (double)( ( AdVal   & 0x1FFF ) <<3 ) * 10.0 / 32768.0 - 5.0;
```

第2章 Windows95/98/Me解説

2-1. インストレーション

Windows95 OSR-2 (注 1) のリリースにより現在 Windows95 のバージョンには、Windows95 OSR-2 と OSR-2 以前のバージョンがあります。「マイコンピュータ」を右クリックし「プロパティ」情報を表示することによりどちらのバージョンがインストールされているか調べることができます。システム情報が「Microsoft Windows95 4.00.950 a」の場合は OSR-2 以前のバージョンになり、OSR-2 の場合は「Microsoft Windows95 4.00.950 B」となります。ご利用の Windows95 が OSR-2 かそれ以前のバージョンかによりインストールの方法が異なりますので注意して下さい。

（注1） OSR-2（OEM Service Release 2）は FAT32・CardBus 等の新機能がサポートされたバージョンです。

2-1-1. インストール方法

Windows95 OSR-2 でのインストール方法

【1】PC カードの挿入

PC カードをスロットに挿入すると、ハードウェアウィザードが起動し右のデバイスドライバーウィザードのインストールが表示されます。ここでは、次へを選択します。

【2】ドライバーファイル場所の指定

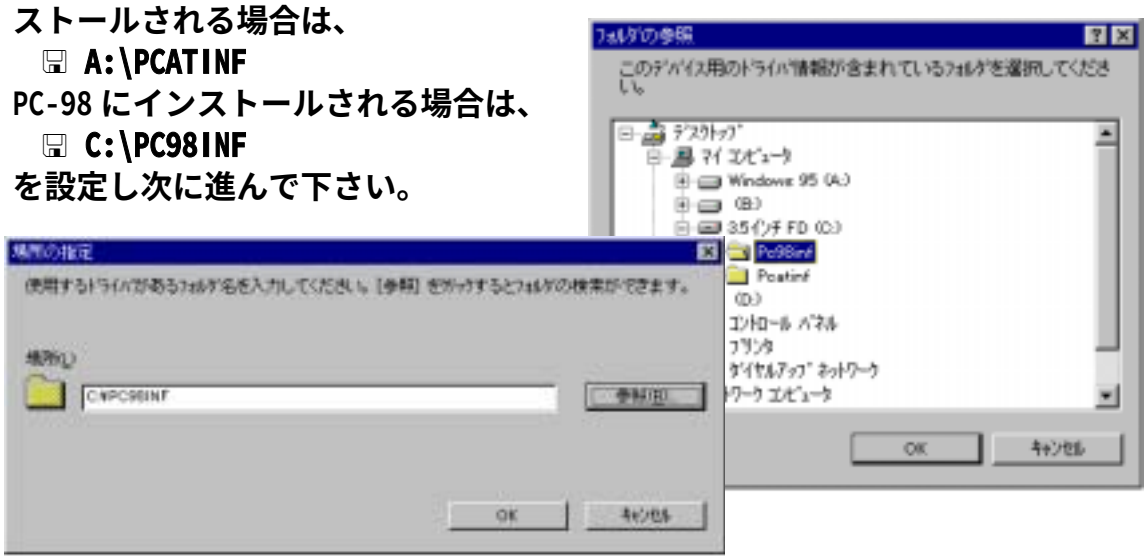
次にドライバーファイル（INF ファイル）の場所を指定します。PC/AT にインストールされる場合は、

A:\PCATINF

PC-98 にインストールされる場合は、

C:\PC98INF

を設定し次に進んで下さい。



【3】インストールの完了

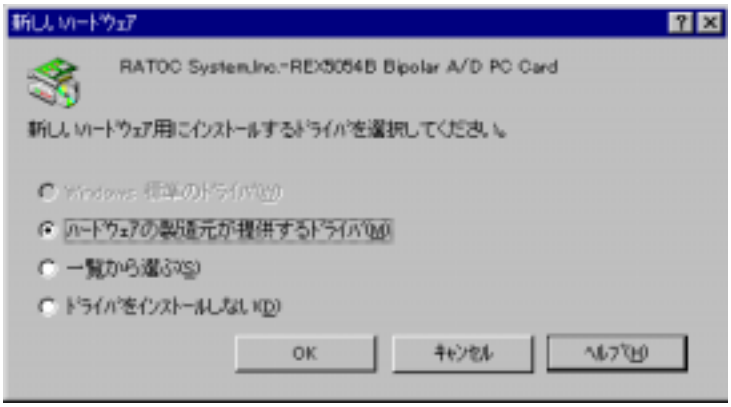
インストールが正常に完了した場合は、「このデバイス用に更新されたドライバーが見つかりました。」のメッセージが表示されますので確認して下さい。

以上でインストールは完了です。

Windows95（OSR-2 以前のバージョン）でのインストール方法

【1】PC カードの挿入

PC カードをスロットに挿入すると、右のハードウェアウィザードが起動します。ここでは「ハードウェアの製造元が提供するドライバ(M)」選択し次に進みます。



【2】配布ファイルコピー元の指定

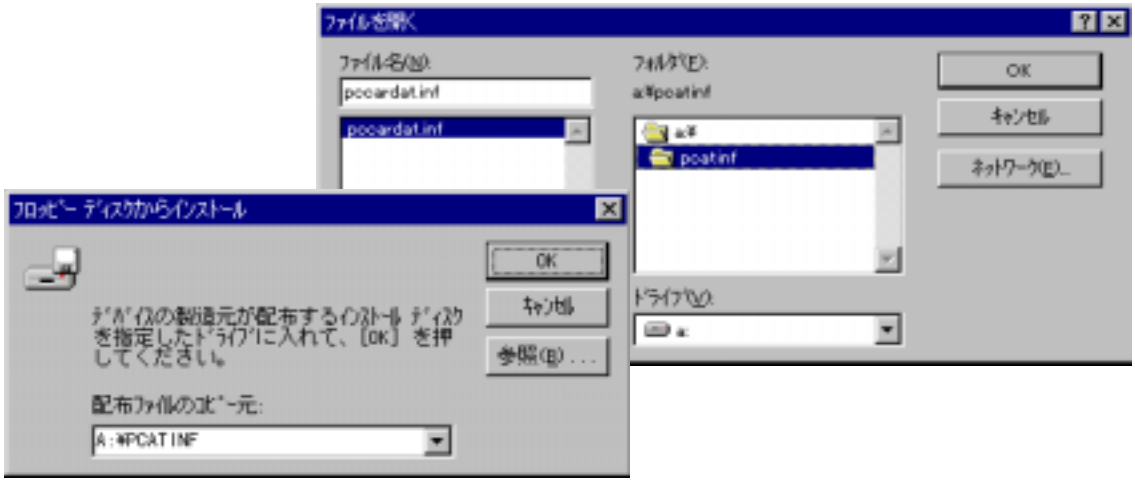
次にドライバーファイル（INF ファイル）の場所を指定します。PC/AT にインストールされる場合は、

A:\PCATINF

PC-98 にインストールされる場合は、

C:\PC98INF

を設定し次に進んで下さい。



【3】インストールの完了

インストールが正常に行われるとビープ音で完了が通知され、ハードウェアウィザードは自動的に終了します。

以上でインストールは完了です。

Windows98 でのインストール方法

【1】PC カードの挿入

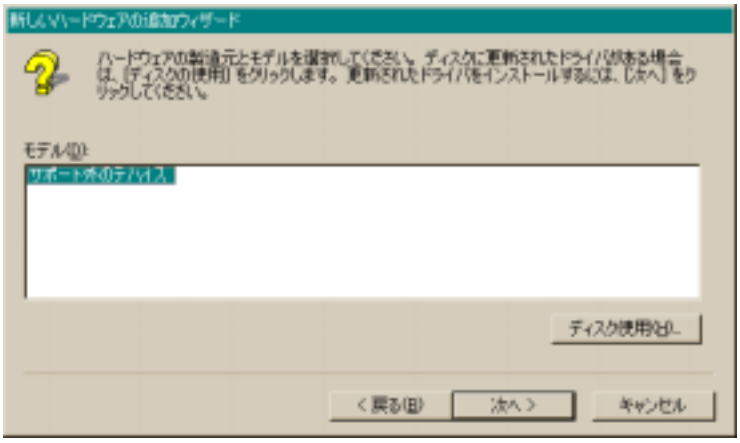
PC カードをスロットに挿入すると、ハードウェアウィザードが起動し右のデバイスドライバーウィザードのインストールが表示されます。ここでは、次へを押します。

ドライバの検索方法は「特定の場所にあるすべてのドライバの一覧を作成し、インストールするドライバを選択する。」を選択し、次へを押します。

デバイスの種類は「その他のデバイス」または「Otherdevices」を選択し、次へを押します。

【2】ドライバーファイル場所の指定

モデルの選択では「ディスク使用」を押します。



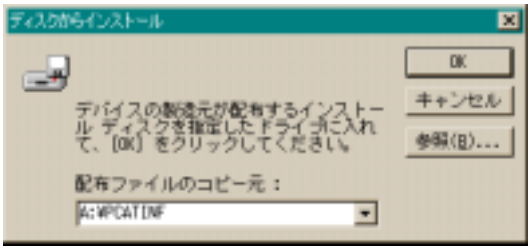
製品添付ディスク Windows95 セットアップファイルをフロッピーディスクドライブに挿入し、次にドライバーファイル（INF ファイル）の場所を指定します。
PC/AT にインストールされる場合は、

A:\PCATINF

PC-98 にインストールされる場合は、

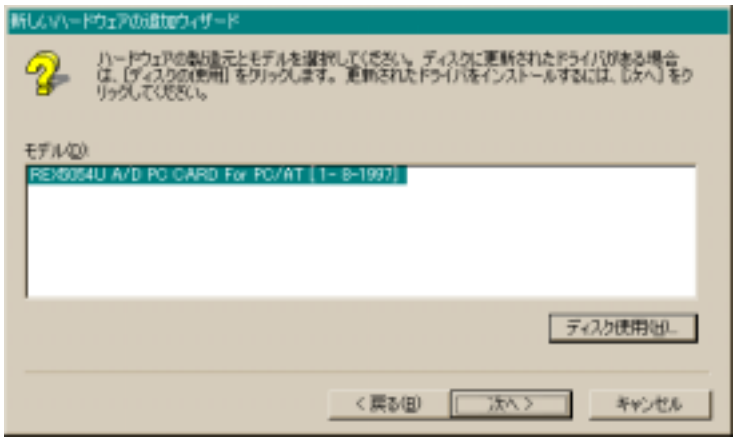
C:\PC98INF

と入力し、OK を押します。



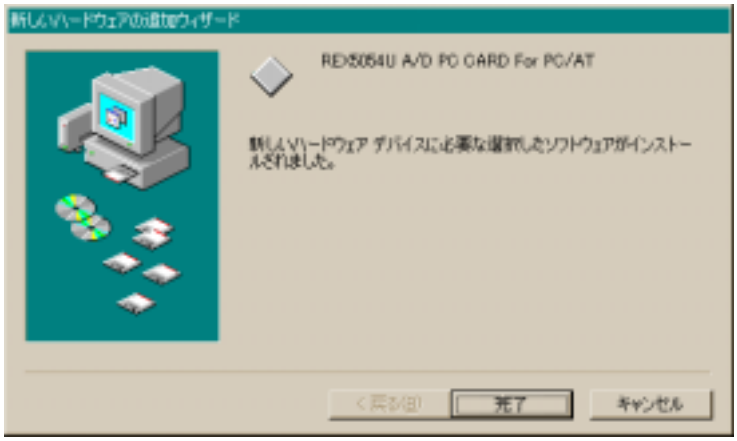
正しいモデル名「REX5054U A/D PC CARD for PC/AT」または「REX5054B A/D PC CARD for PC/AT」が表示されたら、次へを押します。

（注意）
NEC PC9821 シリーズをご利用の場合は、for PC/AT の表示が for PC98 になっていることを確認します。



インストール準備が完了したら、次へを押します。

インストール完了が表示されたら、完了を押してハードウェアウィザードを終了します。



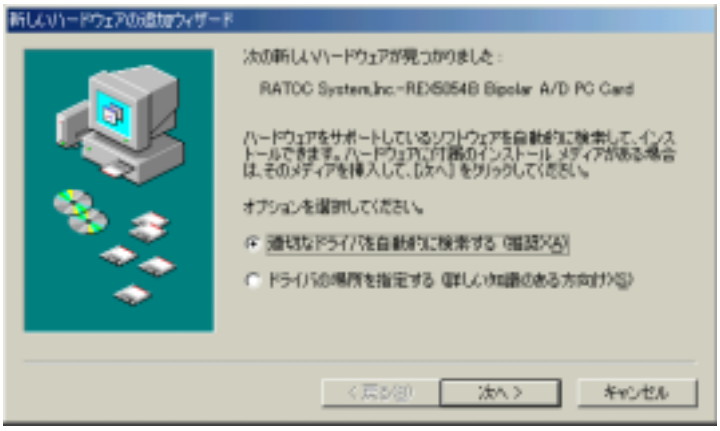
WindowsMe でのインストール方法

※以下のインストレーション画面は、REX-5054B を使用しておりますので REX-5054U をご使用のお客様は、「REX-5054B」の部分を「REX-5054U」にお読み替え下さい。

【1】PC カードの挿入

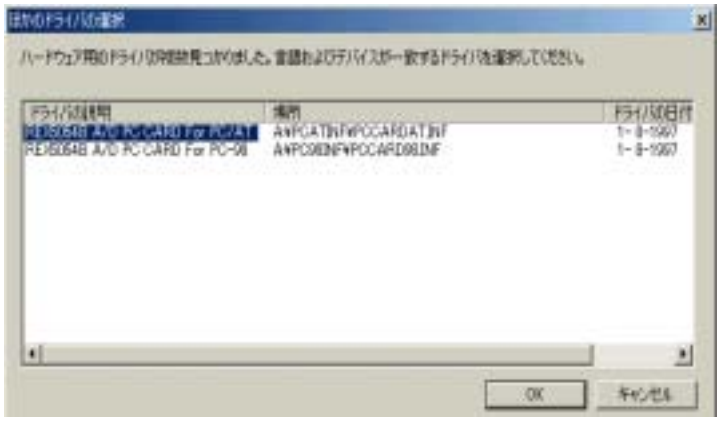
PC カードをスロットに挿入すると、右の新しいハードウェアの追加ウィザードが表示されますので、製品添付の Win95/98/Me 用フロッピーディスクを FD ドライブへ挿入してください。

次に、「適切なドライバを自動的に検索する （推奨）(A)」を選択し「次へ」ボタンを押します。

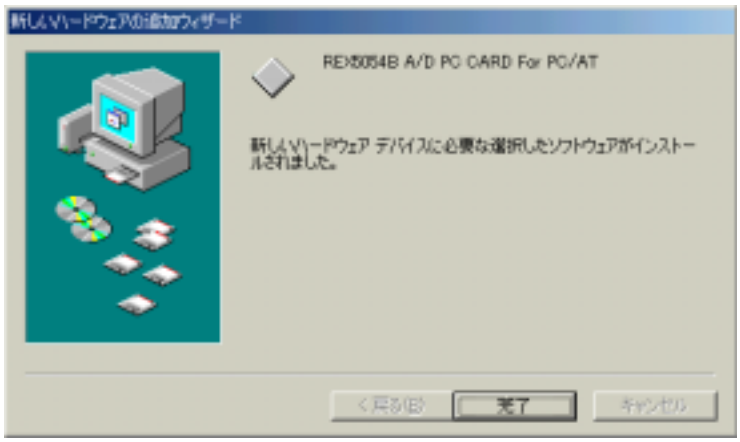


【2】ドライバーファイル場所の指定

右のようにセットアップ情報ファイル (inf ファイル) が、フロッピーディスク上から自動的に検索されますので、「REX-5054B A/D PC CARD For PC/AT を選択し、「OK」ボタンを押します。



右の画面が表示されましたら、「完了」ボタンを押します。



以上で、REX-5054 インストールは終了です。

2-1-2. アンインストール方法

カードが正しくインストールされなかった場合は以下の手順でカード情報の削除と INF ファイルの削除を行い、再度、2-1-1.の方法でインストールを行って下さい。

【1】カード情報の削除

PC カードを挿入した状態で、コントロールパネルからシステムを起動します。
「デバイスマネージャ」のタブを選択し、Otherdevices の下にある「REX5054U/B A/D PC CARD for PC/AT」を選択して「削除ボタン」で削除して下さい。

【2】INF ファイルの削除

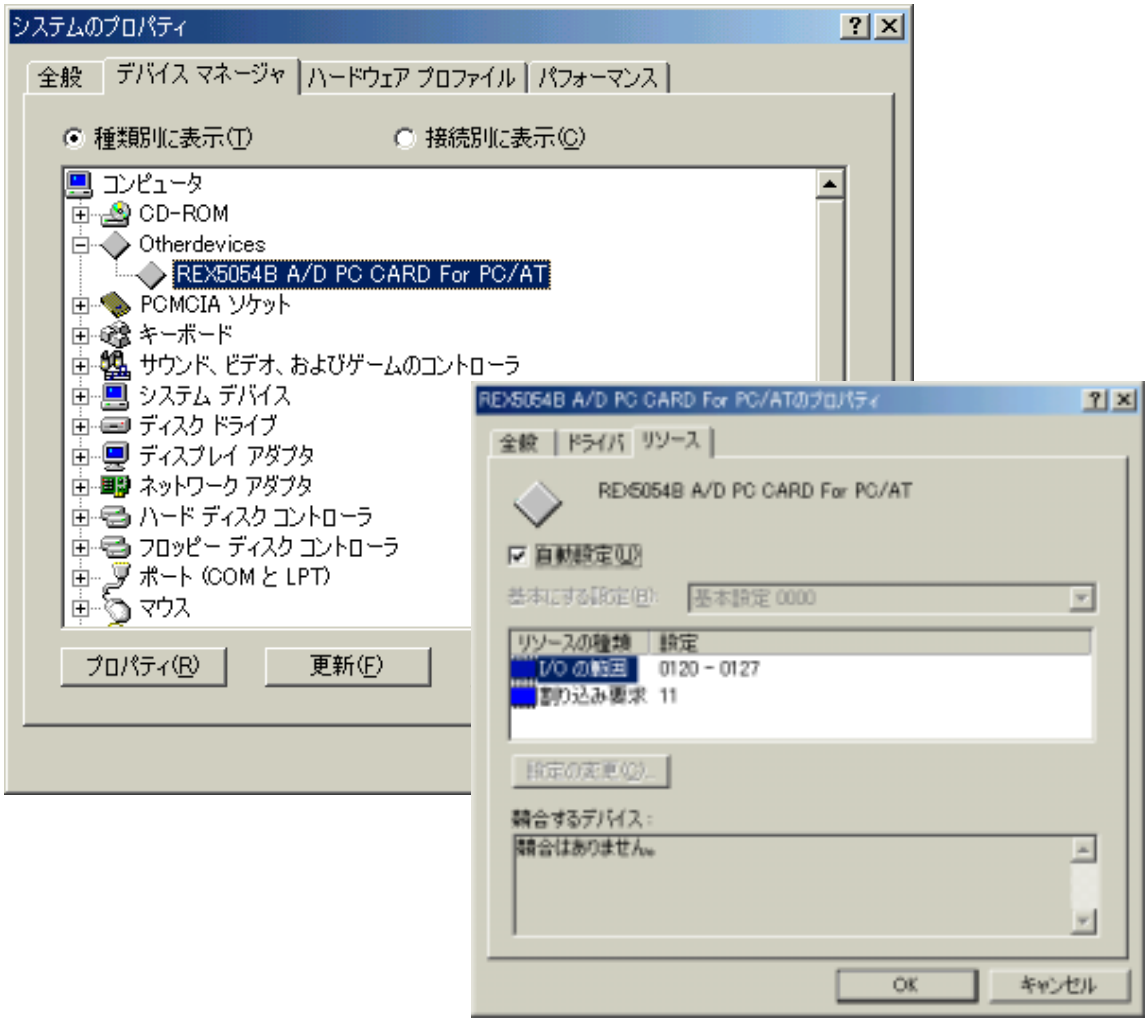
「エクスプローラ」を開いて\Windows\Inf\Other にある「RATOC System,Inc .PCCARDAT.INF」を削除して下さい。

2-2. PC カード設定内容の確認

→システムプロパティの起動

コントロールパネルのシステムを起動し、デバイスマネージャを選択します。カードの設定が正常に行われていれば、コンピュータのレジストリツリー「Otherdevices」の下に「REX5054U/B A/D PC CARD For PC/AT(または PC98)」が登録されます。

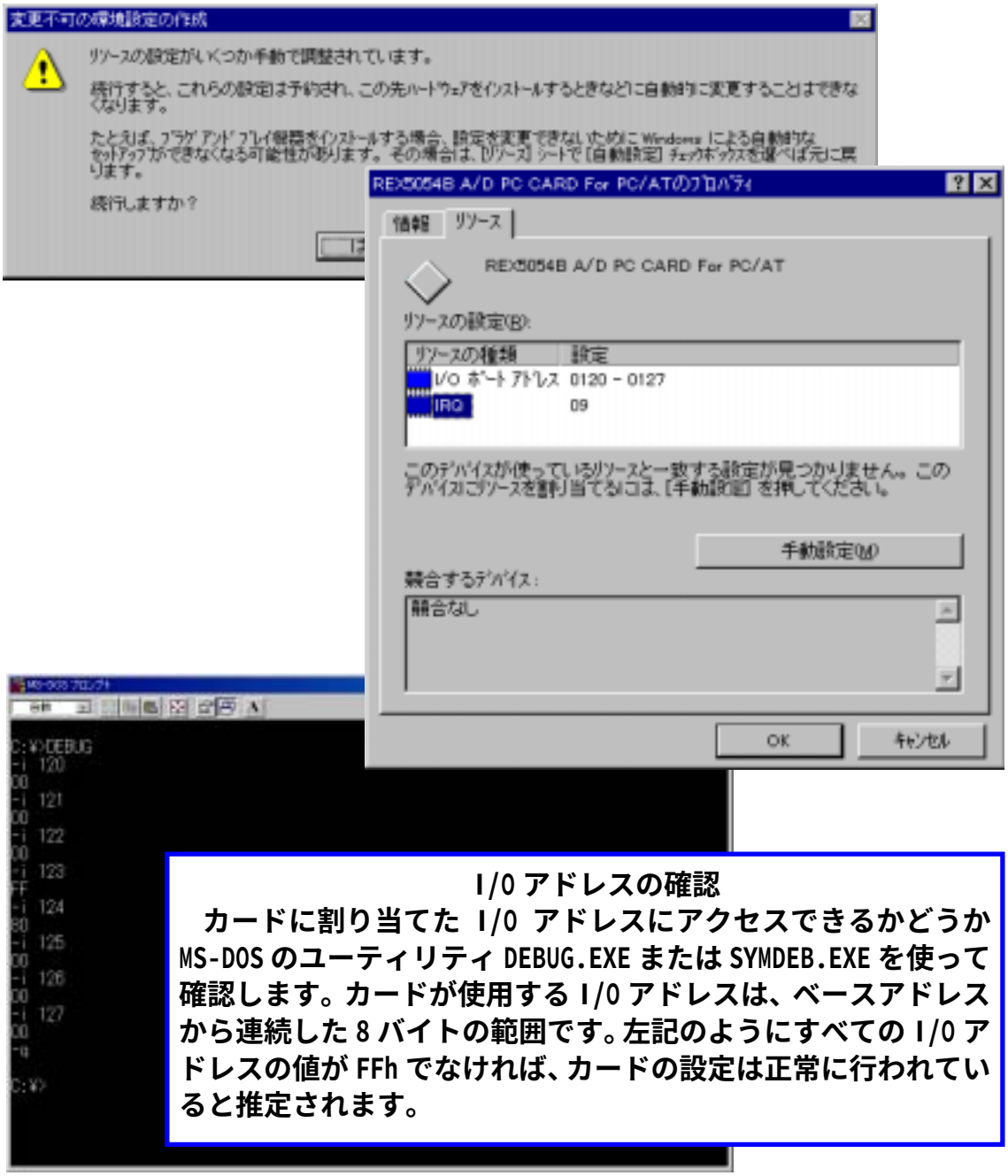
プロパティのリソースタグを選択して I/O ポートアドレスおよび IRQ の割り当てで競合していないことを確認して下さい。



→リソースの変更

リソースの変更は、自動設定を選択しないようにして、設定の登録名を別の設定に変更します。手動設定を行うと、「変更不可の環境設定の作成」ダイアログが表示されますが、続行「はい」を選択して下さい。

手動設定したリソースが他のデバイスと競合していなければ、「ピッポッ」というビープ音とともにシステムプロパティ画面に戻ります。もう一度、レジストリ「REX5054U/B A/D PC CARD For PC/AT」をダブルクリックし、手動設定したリソースが他のデバイスと競合していないことを確認して下さい。



I/O アドレスの確認

カードに割り当てた I/O アドレスにアクセスできるかどうか MS-DOS のユーティリティ DEBUG.EXE または SYMDEB.EXE を使って確認します。カードが使用する I/O アドレスは、ベースアドレスから連続した 8 バイトの範囲です。左記のようにすべての I/O アドレスの値が FFh でなければ、カードの設定は正常に行われていると推定されます。

2-3. Visual C 言語インターフェイス

2-3-1. DLL ライブラリ API 解説

本製品には 32 ビットアプリケーション開発に必要となる API インターフェースを提供する DLL ライブラリ“ADLIB32.DLL”と、特権レベルで Windows システムに対し割り込み要求等を行う“REXVPCD.VXD”仮想デバイスドライバーが添付されています。ユーザ側で作成したアプリケーションは“ADLIB32.DLL”の API を呼び出します。“ADLIB32.DLL”は自分で処理できない要求に対しては“REXVPCD.VXD”のファンクションを呼び出して処理を行います。

以下、“ADLIB32.DLL”が提供する API 関数について解説致します。

注）※(関数名)の印が付いている関数につきましては互換性のために残してある関数ですので新規でプログラムを作成する場合には呼び出す必要はありません。

アプリケーション作成上のアドバイス

① PC カードに割り当てられてるリソースを取得する

Windows95 では、PC カードが使用する I/O アドレス・IRQ 番号等のリソースは、カードが挿入された時点で動的な割り当てが行われ一元的に管理されています。今回リリースされた DLL では、現在自分のカードに割り当てられているリソース情報をシステムから取得するための AdGetCardResource()がサポートされています。アプリケーションの初期化部分等で AdGetCardResource()を呼び出し I/O アドレス・IRQ 番号を取得して下さい。

② PC カードへの入出力

Visual C では、I/O 入出力関数がサポートされていません。これは Windows95 の保護機能に基づいたものと考エラーれます。従って、PC カードへの I/O 入出力は DLL でサポートされている OutPort()･InPort()･wOutPort()･wInPort()を使って行います。

③ 割り込み制御

今回バージョンアップされた DLL では、従来のポーリングモードによる AD 変換に加え､割り込みサービスを使用した割り込みモードによる AD 変換がサポートされています。割り込みモードによる AD 変換には、

(1)PC カードからの割り込み要求が発生する毎に､ユーザ定義メッセージがドライバーから上位アプリケーションにポストされる方法

(2)ドライバー内部の割り込みハンドラ内で指定個数の変換が終了した時点で､ユーザ定義メッセージがドライバーから上位アプリケーションにポストする方法

があります。高速な処理が要求されるような場合は、(2)の方法を参照して下さい。

→ DLL 関数仕様

AdAllocDataMem　AD 変換データ格納用メモリのアロケーション

書 式　LPVOID **AdAllocDataMem**(HWND **hDlg**, DWORD **DataLength**)

機 能　指定データ個数の AD 変換データ格納用メモリブロックのアロケーションを要求します。メモリブロックは(指定データ個数+32)ワードサイズでドライバー内部にロックされて確保されます。
確保要求を行ったプログラムは、終了時必ず **AdFreeDataMem**()を呼び出しメモリブロックを解放して下さい。

引 数
- HWND **hDlg** ➢ 呼び出し元ウィンドウのハンドル
- DWORD **DataLength** ➢ 1 チャンネル当たりのサンプリングデータ個数

戻 値　メモリブロックへの VOID 型ポインタを返します。先にサンプリングチャンネルパラメータの設定が必要です。チャンネル設定が行われていない場合、または要求された大きさのメモリを確保できなかった場合は 0 を返します。

AdAllocMem　汎用メモリのアロケーション

書 式　LPVOID **AdAllocMem**(HWND **hDlg**, DWORD **AllocSize**)

機 能　指定バイト数のロックされたメモリブロックをドライバー内部にアロケーションします。確保要求を行ったプログラムは終了時に必ず **AdFreeMem**()を呼び出して確保したメモリブロックを解放して下さい。

引 数
- HWND **hDlg** ➢ 呼び出し元ウィンドウのハンドル
- DWORD **AllocSize** ➢ アロケーションサイズ(バイト単位)

戻 値　アロケーションしたメモリブロックへの VOID 型ポインタを返します。0 の場合はアロケーションできなかったことを示します。

AdAllocRingBuf　リングバッファのアロケーション

書 式　BOOL **AdAllocRingBuf**(HWND **hDlg**, DWORD **wAllocSize**, DWORD **DataLength**)

機 能　指定ワードサイズの AD 変換データ格納用リングバッファメモリの確保し、リングバッファ制御変数を初期化します。AD 変換はトータルの変換個数が（DataLength×指定チャンネル数）に達すると終了します。
確保要求元のプログラムは、終了時必ず **AdFreeRingBuf**()を呼び出して確保したリングバッファを解放して下さい。

引 数
- HWND **hDlg** ➢ 呼び出し元ウィンドウのハンドル
- DWORD **wAllocSize** ➢ ワード単位のアロケーションサイズ
- DWORD **DataLength** ➢ 1 チャンネル当たりの変換データ個数

戻 値
- 0 : リングバッファ初期化正常終了
- -1 : リングバッファ制御パラメータ構造体オブジェクト作成エラー
- -2 : リングバッファアロケーションエラー
- -3 : 二重設定エラー

AdAllocTrgBuf　　入力電圧リミットトリガーバッファのアロケーション

書 式　long **AdAllocTrgBuf**(WORD **SampTime**, WORD **TimeUnit**, WORD **AdChan**, WORD **PreTime**,
WORD **PostTime**)

機 能　リミットトリガを検出した時点を基準としてその前の PreTime 秒間の変換データとその後の PostTime 秒間の変換データを格納するためのバッファを確保します。呼び出し側プログラムは終了時 AdFreeTrgBuf()により確保されたバッファを解放して下さい。**AdSetLimitTrg**()で設定されたリミットトリガに対し、前後で取得する変換データ個数は下記 PreTime および PostTime より下式により計算されます（TimeUnit がミリ秒の場合）。

リミットトリガ以前の変換データ格納バッファワードサイズ
= (PreTime * 1000 / SampTime) * AdChan;

リミットトリガ以降の変換データ格納バッファワードサイズ
= (PostTime * 1000 / SampTime) * AdChan;

本バッファはリングバッファモードで変換データが格納されます。

引 数

- WORD **SampTime** ➢ A/D 変換サンプリング周期...AdSetFreq()での設定値を指定
  注）1msec より速くは変換不可。
- WORD **TimeUnit** ➢ サンプリング周期の単位を下記より指定
  sec(0):秒　msec(1):ミリ秒
- WORD **AdChan** ➢ 変換チャンネル総数指定
  ...AdSetChannel()での設定値を指定
- WORD **PreTime** ➢ リミットトリガ発生前何秒間のデータを保持するか
  秒単位で指定
- WORD **PostTime** ➢ リミットトリガ発生後何秒間のデータを保持するか
  秒単位で指定

戻 値　正常に変換データ格納用バッファが確保されると、確保されたバッファのワードサイズが返されます。

-1 ： リミットトリガー制御メモリーアロケーションエラー
-2 ： 変換データ格納バッファアロケーションエラー
-3 ： リミットトリガーモード変換時間設定エラー

※ (AdCheckStopTrig)　外部トリガーによる変換終了チェック

書 式　BOOL **AdCheckStopTrig**(HWND **hDlg**)

機 能　AD カードのステータスをリードし、外部トリガーにより AD 変換が終了しているかチェックします。本チェックルーチンは下記サンプルに示すようにタイマーイベントのコールバックルーチンから呼び出します。変換が停止している場合は、wParam に外部トリガーによる終了メッセージステータス“AD_MODE_PRESTOP”をセットし、lParam に変換終了までの全データ個数をセットしてユーザ定義メッセージ“WM_VXDEVENT”を呼び出し元ウィンドウにポストします。

引 数　HWND hDlg　➢ 呼び出し元ウィンドウのハンドル

戻 値　TRUE　: 外部トリガーにより変換終了
FALSE : 外部トリガー未検出

サンプル

```
LRESULT CALLBACK AdStartIrqHiSpeedUseTrigDlg
                ( HWND hDlg, UINT wMessage, WPARAM wParam, LPARAM lParam )
{
   switch( wMessage )
   {
   /* 割込ハンドラから変換終了メッセージ受信時の処理   */
   case WM_VXDEVENT:
      AdMode = wParam;          /* wParam -> AD 変換終了モード */
      AdDataCount = lParam;     /* lParam -> AD 変換データ数 */
      return TRUE;
   case WM_COMMAND:
      switch ( wParam )
      {
      case IDOK:
         /* 割込みによるＡＤ変換入力開始 */
         AdStartIrqVxdMode( hDlg, 0 );
         /* 外部トリガー-終了モード検出用にタイマーコールバックルーチン登録 */
         SetTimer( hDlg, MY_TIMER, 100, (TIMERPROC)CheckStopTrig);
         return TRUE;
      }
   }
   return FALSE;
}

VOID CALLBACK CheckStopTrig
              ( HWND hDlg, UINT wMessage, WPARAM wParam, LPARAM lParam )
{
   /* 外部トリガーによる変換終了時"TRUE"が返ります */
   if ( AdCheckStopTrig( hDlg ) == TRUE )
   {
      /* タイマーコールバック処理終了 */
      KillTimer( hDlg, MY_TIMER );
   }
}
```

AdFreeDataMem　AD 変換データ格納用メモリを解放

書 式　VOID APIENTRY **AdFreeDataMem**(VOID)

機 能　**AdAllocDataMem**()で確保した変換データ格納用を解放します。

引 数　なし

戻 値　なし

AdFreeMem　汎用メモリの解放

書 式　VOID **AdFreeMem**(LPVOID **pHeapMem**)

機 能　**AdAllocMem**()で確保したメモリブロックを解放する

引 数　LPVOID **pHeapMem**　➢ **AdAllocMem**()で取得したメモリポインタ

戻 値　なし

AdFreeRingBuf　リングバッファの解放

書 式　VOID **AdFreeRingBuf**(VOID)

機 能　**AdAllocRingBuf**()で確保したリングバッファおよびリングバッファ制御パラメータを解放します。

引 数　なし

戻 値　なし

AdFreeTrgBuf　入力電圧リミットトリガバッファの解放

書 式　VOID **AdFreeTrgBuf**(VOID)

機 能　**AdAllocTrgBuf**()で確保されたリミットトリガ用バッファを解放します。

引 数　なし

戻 値　なし

※ (AdGetCardNameInfo) PC カードの製品情報名を取得

書 式 BOOL **AdGetCardNameInfo**(HWND **hDlg**, WORD **SlotNo**, LPSTR **CopyBuf**,
WORD **CopyBufLen**)

機 能 指定スロットに挿入されている PC カードの製品情報名を取得します。

引 数

HWND	**hDlg**	➢ 呼び出し元ウィンドウのハンドル
WORD	**SlotNo**	➢ スロット番号
LPSTR	**CopyBuf**	➢ コピー先メモリバッファのアドレス
WORD	**CopyBufLen**	➢ コピー先メモリバッファのレングス

戻 値
0 : 正常終了
-1 : DEVICE I/O コントロールエラー
-2 : カードサービスドライバーバージョンエラー
-3 : GET_CARD_SERVICES_INFO ファンクションコールサービスエラー
-4 : GET_FIRST_TUPLE ファンクションコールサービスエラー
-5 : GET_TUPLE_DATA ファンクションコールサービスエラー
-6 : GET_CONFIG_INFO ファンクションコールサービスエラー
-7 : メモリアロケーションエラー
-8 : コピー先メモリバッファの長さがコピー元より短い

備 考 PC カード内部にはカードサービスドライバー等からカードの仕様を読み出すことが可能なように CIS 情報と呼ばれるデータが書き込まれています。CIS は複数のタプルの集合体から構成されており、タプルの一つにバージョン・プロダクツ情報タプルがあります。本製品のプロダクツ情報タプルは下記レイアウトの内容が記載されており、上記関数を使ってプロダクツネームを取得することができます。

項目	長さ	タプルデータ
タプル ID	1	15h
タプルリンク	1	33h 又は 32h
メジャーリビジョン	1	05h
マイナーリビジョン	1	00h
カンパニーネーム	11h	"RATOC System,Inc."
ターミネータ	1	00h
プロダクツネーム	1dh 1ch	"REX5054U Unipolar A/D PC Card" 又は "REX5054B Bipolar A/D PC Card"
ターミネータ	1	00h
プロダクツリビジョン	0	-
ターミネータ	1	00h
タプルエンド	1	FFh

AdGetCardResource　REX5054U/B に割り当てられているリソースの取得

書 式　BOOL **AdGetCardResource**(HWND **hWnd**, WORD **SlotNo**, LPWORD **pCardType**,
LPWORD **pIOBase**, LPWORD **pIrqNo**)

機 能　指定スロットに挿入されているカードが自分のカードなら、現在割り当てられている I/O ベースアドレスおよび IRQ 番号情報を返します。

引 数

HWND	**hWnd**	➢ 呼び出し元ウィンドウのハンドル
WORD	**SlotNo**	➢ PC カード挿入スロット番号
LPWORD	**pCardType**	➢ 出力）指定スロットで検出された AD カードの型式
LPWORD	**pIOBase**	➢ 出力）I/O ベースアドレス情報を格納する変数へのポインタ
LPWORD	**pIrqNo**	➢ 出力）IRQ 番号情報を格納する変数ポインタ

戻 値
0 : 正常終了
-1 : DEVICE I/O コントロールエラー
-2 : カードサービスドライバーバージョンエラー
-3 : GET_CARD_SERVICES_INFO ファンクションコールサービスエラー
-4 : GET_FIRST_TUPLE ファンクションコールサービスエラー
-5 : GET_TUPLE_DATA ファンクションコールサービスエラー
-6 : GET_CONFIG_INFO ファンクションコールサービスエラー
-7 : メモリアロケーションエラー
-8 : スロットにカードが挿入されていない
-9 : スロットに挿入されているカードは自分のカードと一致しない

AdGetParam　設定されているサンプリングパラメータを取得

書 式　VOID **AdGetParam**(LPWORD **pCardType**, LPWORD **pIOAddr**, LPWORD **pIRQNo**,
LPWORD **pStime**, LPWORD **pUnit**, LPWORD **pChannels**,
LPDWORD **pDataLen**, VOID **ppMem**)

機 能　現在設定されているサンプリングパラメータを取得

引 数

LPWORD	**pCardType**	➢ AD PC カード型式格納先変数へのポインタ
LPWORD	**pIOAddr**	➢ I/O ベースアドレス格納先変数へのポインタ
LPWORD	**pIRQNo**	➢ IRQ 番号格納先変数へのポインタ
LPWORD	**pStime**	➢ サンプリング間隔格納先変数へのポインタ
LPWORD	**pUnit**	➢ サンプリング間隔単位格納先変数へのポインタ
LPWORD	**pChannels**	➢ サンプリングチャンネル数格納先変数へのポインタ
LPDWORD	**pDataLen**	➢ サンプリング個数格納先変数へのポインタ
VOID	**ppMem**	➢ ドライバーで確保した変換データ格納先メモリへのポインタ

戻 値　なし

AdGetRingBufAdrs　リングバッファの先頭アドレスを取得

書 式　LPVOID **AdGetRingBufAdrs**(VOID)

機 能　リングバッファメモリブロックの先頭アドレスを取得します。

引 数　なし

戻 値　リングバッファメモリブロックの先頭アドレスを返します。0 の場合はリングバッファが確保されていないこと示します。

AdGetRingBuf　リングバッファから変換データを転送

書 式　BOOL **AdGetRingBuf**(LPVOID **pUserBuf**, BOOL **Offset**, BOOL **GetDataNum**)

機 能　リングバッファから指定個数の変換データをリードし指定のアドレスへ転送します。データの転送先アドレスは pUserBuf から Offset で示される変換データ個分オフセットした場所になります。
本関数は第三引数で指定した個数分の変換データがリングバッファにない場合は格納されている個数分を転送します。

引 数
- LPWORD **pUserBuf** ➢ データ格納先のアドレス
- BOOL **Offset** ➢ データ格納先のアドレスのオフセット
- BOOL **GetDataNum** ➢ 取得するデータ数

戻 値　転送したの変換データ個数を返します。 リングバッファが空の時は 0 を、リングバッファオーバーフロー発生時 -1 を返します。

AdGetRingBufConst　リングバッファから一定個数の変換データを転送

書 式　int **AdGetRingBufConst**(LPVOID **pUserBuf**, int **Offset**, int **GetDataNum**)

機 能　**GetDataNum**()で指定した個数分の変換データがリングバッファに格納された場合にリングバッファから指定個数の変換データをリードし指定のアドレスへ転送します。データの転送先アドレスは pUserBuf から Offset で示される変換データ個分オフセットした場所になります。
本関数は **AdGetRingBuf**()とは異なり、第三引数で指定した個数分の変換データがリングバッファにない場合は転送を行わず、戻り値として 0 を返します。

引 数
- LPWORD **pUserBuf** ➢ データ格納先のアドレス
- int **Offset** ➢ データ格納先のアドレスのオフセット
- int **GetDataNum** ➢ 取得するデータ数

戻 値　転送した変換データ個数を返します。
リングバッファが空の場合もしくは第三引数で指定した個数分の変換データがリングバッファにない場合は 0 を返します。リングバッファオーバーフロー発生時には-1 を返します。

AdGetTrgData 入力電圧リミットトリガバッファから変換データを転送

書 式 DWORD **AdGetTrgData**(PWORD **pStartMem**, PWORD **pCopyMem**)

機 能 **AdAllocTrgBuf()**で確保されたリミットトリガー用バッファから変換データを取り出します。
コピー先バッファの先頭には、リミットトリガ発生前何秒間のデータを保持するか指定した時点の変換データから順にセットされます。有効時間が経過する前にリミットポイントが検出された場合は、変換を開始した時点の変換データから順にセットされます。この場合は、コピーされるデータ個数は **AdAllocTrgBuf**()で返されたサイズより少なくなりますので注意願います。

引 数
- PWORD **pStartMem** ➢ リミットトリガーモード変換データ取り出し先頭アドレス
  通常、AD_MODE_STOP 受信時の追加情報 2 で渡されたアドレスをそのまま指定して下さい。
- PWORD **pCopyMem** ➢ コピー先バッファアドレス
  コピー先バッファのサイズは AdAllocTrgBuf()で返されたワードサイズ分確保しておいて下さい。

戻 値 コピーされたデータの個数が返されます。

AdGetVersion DLL ライブラリのバージョン表示

書 式 VOID **AdGetVersion**(HWND **hWnd**)

機 能 DLL ライブラリのバージョンダイアログボックスを表示します。

引 数 HWND **hWnd** ➢ 呼び出し元ウィンドウのハンドル

戻 値 なし

AdOneShot ワンショットモード AD 変換実行

書 式 BOOL **AdOneShot**(HWND **hDlg**, LPWORD **pMem**)

機 能 ワンショットモードによる AD 変換を行います。変換を実行する前に、**AdSetOneParam**()でパラメータ設定を完了しておいて下さい。

引 数
- HWND **hDlg** ➢ 呼び出し元ウィンドウのハンドル
- LPWORD **pMem** ➢ 変換データ格納先メモリへのポインタ

戻 値
0:正常終了
-1:ADC コントローラリセットエラー
-2:ADC コントローラフルキャリブレーションエラー
-3:分周回路設定エラー
-4:サンプリングクロックレジスタ設定エラー
-5:FIFO オーバーランエラー

※ (AdPlot) サンプリング波形のグラフ表示

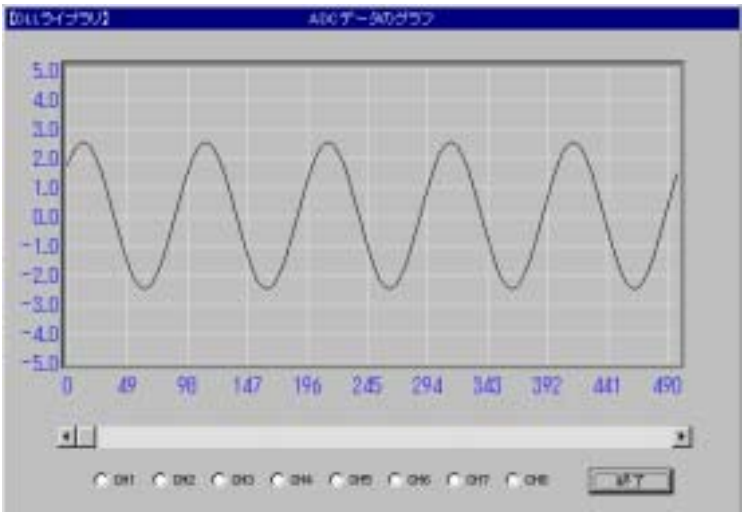
書 式 VOID **AdPlot**(HWND **hWnd**, LPWORD **lpAdData**)

機 能 lpAdData が示すメモリから変換データを取り出してグラフ表示します。本関数を呼び出す前に変換データが格納されたメモリを解放しないで下さい。

引 数
- HWND **hWnd** ➢ 呼び出し元ウィンドウのハンドル
- LPWORD **lpAdData** ➢ 変換データが格納されているメモリへのポインタ

戻 値 なし

備 考 画面仕様



AdSaveCsvFile　Excel CSV ファイル形式でデータ保存

書 式

```
BOOL AdSaveCsvFile( HWND hWnd, WORD MyCardType, WORD AdChannels,
                    WORD SampTime, WORD TimeUnitNo, LPWORD pMyApMem,
                    DWORD NumberOfAdData )
```

機 能

メモリに格納されている変換データをエクセルで読み込み可能な CSV 形式でファイル保存します。エクセルのセル上に 1 列目が時間、2 列目からチャンネル 1、3 列目からチャンネル 2…の順で展開されます。

引 数

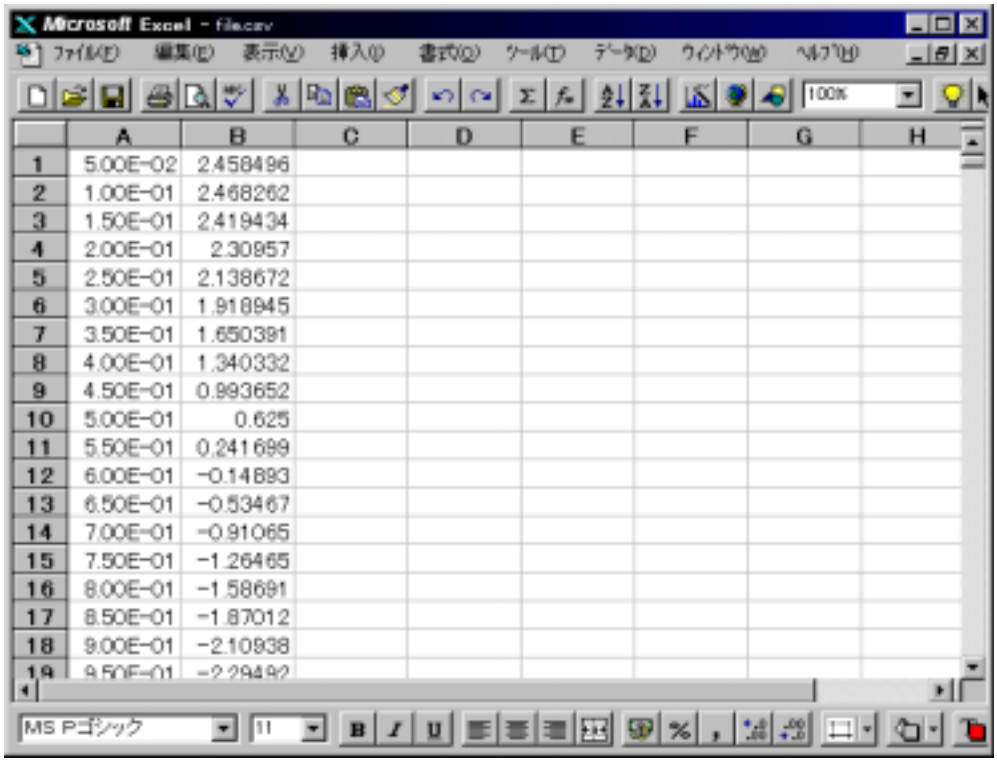
HWND	hWnd	➢ 呼び出し元ウィンドウのハンドル
WORD	MyCardType	➢ カード型式 0:REX5054U 1:REX5054B
WORD	AdChannels	➢ 変換データチャンネル数
WORD	SampTime	➢ 変換インターバル時間
WORD	TimeUnitNo	➢ 時間単位 0:sec 1:msec 2:usec
LPWORD	pMyApMem	➢ 変換データが格納されているメモリへのポインタ
DWORD	NumberOfAdData	➢ CSV ファイルに保存する変換データの個数

戻 値

0 ： 正常終了
-1 ： 書き込みエラー

備 考

エクセルにロードすると A 列にはサンプリング時間が B 列にはチャンネル 1 の変換データが表示されます。



	A	B
1	5.00E-02	2.458496
2	1.00E-01	2.468262
3	1.50E-01	2.419434
4	2.00E-01	2.30957
5	2.50E-01	2.138672
6	3.00E-01	1.918945
7	3.50E-01	1.650391
8	4.00E-01	1.340332
9	4.50E-01	0.993652
10	5.00E-01	0.625
11	5.50E-01	0.241699
12	6.00E-01	-0.14893
13	6.50E-01	-0.53467
14	7.00E-01	-0.91065
15	7.50E-01	-1.26465
16	8.00E-01	-1.58691
17	8.50E-01	-1.87012
18	9.00E-01	-2.10938
19	9.50E-01	-2.29492

AdSaveFile　　バイナリ形式で変換データファイル保存

書 式　BOOL **AdSaveFile**(HWND **hWnd**, LPWORD **pMyApMem**, DWORD **NumberOfAdcData**)

機 能　メモリに格納されている変換データをバイナリファイル形式でファイル保存します。1 回目の各チャンネル変換データ、2 回目の各チャンネル変換データの順でデータが格納されています。1 つの変換データのサイズは 2 バイトです。

引 数

HWND	**hWnd**	➢ 呼び出し元ウィンドウのハンドル
LPWORD	**pMyApMem**	➢ 変換データが格納されているメモリへのポインタ
DWORD	**NumberOfAdcData**	➢ バイナリファイルに保存する変換データの個数

戻 値　0 : 正常終了
-1 : 書き込みエラー

※ (AdSetAndStart)　　DLL ダイアログを使用したポーリングモード AD 変換実行

書 式　BOOL **AdSetAndStart**(HWND **hDlg**)

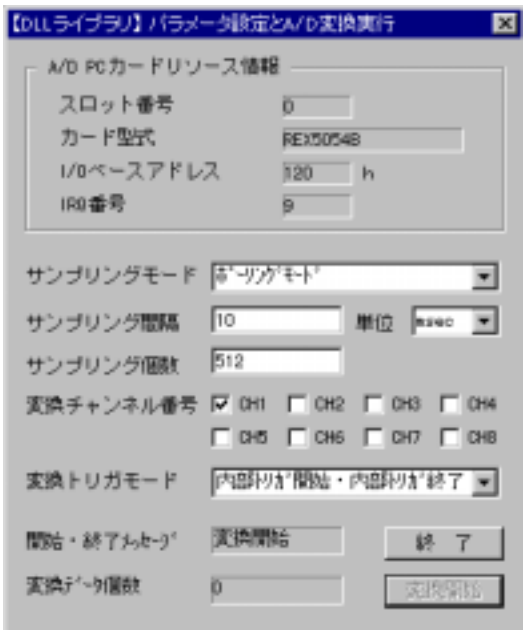
機 能　DLL が用意したパラメータ入力ダイアログを使ってポーリングモードで AD 変換を実行します。変換データが格納されているメモリのアドレスは **AdGetParam()** を呼び出して取得することができます。本関数は変換データを格納するためのメモリを **AdAllocDataMem()**を呼び出して確保しています。呼び出し元プログラムは、終了時必ず **AdFreeDataMem()**を呼び出し本関数で内部的に確保したメモリを解放して下さい。

引 数

HWND	**hDlg**	➢ 呼び出し元ウィンドウのハンドル

戻 値　変換が実行されていれば 0、何もしないで終了した場合は-1 を返す

備 考　実行時の画面を右に示します。



AdSetChannel 変換チャンネルパラメータの設定

書 式 BOOL **AdSetChannel**(WORD **Channels**, LPWORD **pSequence**)

機 能 **カード内部の A/D コンバージョンチップ（LM12458）に変換実行チャンネルの設定を行います。チャンネルシーケンスに設定されたチャンネル番号は昇順に並び換えます。同じチャンネルが複数指定されている場合は重複チャンネルを削除します。**

引 数
- WORD **Channels** ➢ AD 変換チャンネル数(1 オリジン)
- LPWORD **pSequence** ➢ チャンネルシーケンス（0 オリジン）

戻 値 **正常終了時、設定したチャンネル数を返します。それ以外はエラーになります。**

サンプル

```
Sequence[0] = 0;  Squence[1] = 1;  Sequence[2] = 2;  Sequence[3] = 3;
Channels = 4;
if ( AdSetChannel( Channels, Sequence ) != (BOOL)Channels )
{
  MessageBox (hDlg, "チャンネル設定誤り", " ﾊﾟﾗﾒｰﾀｾｯﾄ", MB_OK|MB_ICONSTOP);
  return -1;
}
```

AdSetDataCount サンプリング個数の設定

書 式 DWORD **AdSetDataCount**(DWORD **DataLength**)

機 能 **割り込みモードで割り込みに同期したユーザ定義メッセージを受け取りながら AD 変換を行うとき、本関数により変換を終了するデータ個数の設定を行います。また、ドライバー内部で必要となる 1 チャンネル当たりの変換個数および変換を停止するトータルの変換データ個数は、通常 AdAllocDataMem()を呼び出すことにより内部的に設定されますが、AdAllocDataMem()を使わないでプログラム側で変換データ格納用のメモリを確保する場合は本関数を使って 1 チャンネル当たりの変換個数を設定します。 AdAllocDataMem( )を呼び出した場合は、本関数によるサンプリング個数の設定は省略できます。本関数を呼び出す場合は、必ず先に AdSetChannel()により変換チャンネルの設定を行って下さい。**

引 数
- DWORD **DataLength** ➢ 1 チャンネル当たりのサンプリング個数

戻 値 **正常終了時、全変換個数（サンプリング個数×チャンネル数）を返します。戻り値が 0 の時はエラーです。**

AdSetExternalClock 外部クロック入力の設定

書 式 BOOL **AdSetExternalClock**(BOOL **Mode**, WORD **Counter0**, WORD **Counter1**)

機 能 AD カードの内部クロックを無効にし、外部クロック端子から入力された信号で変換を行います。外部クロックを使う場合、カード内部のタイマーカウンタで分周して変換のクロックとして使用するモードと分周しないで変換のクロックとして使用するモードがあります。分周を行う場合は、カウンタ#0・カウンタ#1 に分周値を設定して下さい。 分周値を大きくする場合は、カウンタ#0・カウンタ#1 の順で設定して下さい。分周値は（カウンタ#0×カウンタ#1）の値になります。

引 数
- BOOL **Mode** ➢ 外部クロック入力モード
  - EXTERNAL_DIRECT 分周しないモード
  - EXTERNAL_DIVIDE 分周するモード
- WORD **Counter0** ➢ カウンタ#0 のカウントデータ（2～65535 の範囲）
- WORD **Counter1** ➢ カウンタ#1 のカウントデータ（0～65535 の範囲）

戻 値 正常終了時 0 を返します。それ以外はエラーです。

AdSetFreq サンプリング間隔の設定

書 式 BOOL **AdSetFreq**(HWND **hWnd**, WORD **Stime**, WORD **Unit**)

機 能 サンプリング間隔と単位を設定します。

引 数
- HWND **hWnd** ➢ 呼び出し元ウィンドウのハンドル
- WORD **Stime** ➢ サンプリング間隔の設定値
- WORD **Unit** ➢ サンプリング間隔単位の設定値
  - sec：0 msec：1 usec：2
  - 入力電圧リミットトリガモードの場合は 1msec より遅く設定してください。

戻 値 正常終了時 0 を返します。それ以外はエラーです。

サンプル

```
if ( AdSetFreq( hDlg, 100, usec ) != 0 )
{
   MessageBox (hDlg, "サンプリング間隔設定誤り", "ﾊﾟﾗﾒｰﾀｾｯﾄ", MB_OK|MB_ICONSTOP);
   return -1;
}
```

AdSetInternalClock　内部クロックの選択とサンプリング間隔の手動設定

書 式　BOOL **AdSetInternalClock**(BOOL **UseClock**, WORD **Counter0**, WORD **Counter1**)

機 能　内部クロックの選択とサンプリング間隔を設定します。カードは 10MHz と 8.192MHz のクロックを内蔵しており、通常は 10MHz の内部クロックを使用しています。サンプリング間隔は、選択したクロックから計算した分周値をカウンタ#0･カウンタ#1 に設定することにより行います。分周値を大きくする場合は、カウンタ#0・カウンタ#1 の順で設定して下さい。分周値は（カウンタ#0×カウンタ#1）の値になります。

引 数　BOOL **UseClock**➢ 内部原発クロックの選択
INTERNAL_10MHZ (10MHz)
INTERNAL_8MHZ (8.129MHz)
WORD **Counter0** ➢ カウンタ#0 のカウントデータ
WORD **Counter1** ➢ カウンタ#1 のカウントデータ

戻 値　正常終了時 0 を返します。それ以外はエラーです。

AdSetLimitTrg　入力電圧リミットトリガ値の設定

書 式　BOOL **AdSetLimitTrg**(double **MaxLimit**, double **MinLimit**, WORD **TrgMode**)

機 能　リミットトリガモード A/D 変換のリミットトリガをかける電圧値を設定します。入力電圧が下記設定上限値もしくは下限値を上回った点でトリガがかかります。トリガが検出された前後の変換データを取得します。リミットトリガ発生の前後で取得する変換データ個数は、**AdAllocTrgBuf**()で設定します。

引 数　double **MaxLimit** ➢ トリガをかける入力電圧の上限値を電圧値で指定
double **MinLimit** ➢ トリガをかける入力電圧の下限値を電圧値で指定
WORD **TrgMode** ➢ 上限値・下限値のいづれを有効にするか下記から選択指定
UPPER_LIMIT :入力電圧が MaxLimit を上回った時、トリガをかける。
LOWER_LIMIT :入力電圧が MaxLimit を下回った時、トリガをかける。
UPPER_LOWER_LIMIT:上記の両方でトリガをかける。

戻 値　0 : 正常終了
-1 : A/D カード型式未設定
-2 : 上限下限値設定範囲エラー
-3 : リミットトリガーモード設定エラー

AdSetOneParam　　ワンショット AD 変換パラメータ設定

書 式　BOOL **AdSetOneParam**(HWND **hDlg**, WORD **CardType**, WORD **BaseAddr**)

機 能　ワンショットモード AD 変換実行時のパラメータを設定します。**AdOneShot**()を実行する前に本関数によりワンショットパラメータの設定を行って下さい。

引 数

HWND **hDlg**	➢ 呼び出し元ウィンドウのハンドル
WORD **CardType**	➢ カード型式（1：REX5054U，2：REX5054B）
WORD **BaseAddr**	➢ カードに割り当てられている I/O ベースアドレス

戻 値
0:正常
-1:型式エラー
-2:サンプリングモードエラー
-3:A/D 変換チャンネル数設定エラー

AdSetParam　カード情報の手動設定

書 式　BOOL **AdSetParam**(HWND **hWnd**, WORD **CardType**, BOOL **SampMode**, WORD **IOBase**, WORD **IrqNo**)

機 能　AD 変換実行に必要なパラメータを手動設定します。本関数を使用する場合は、予め AD カードが使用する I/O ベースアドレス及び割り込み番号を調べておく必要がありますが、**AdSetParamAuto**()を使えばそれらの情報は関数内部でシステムに問い合わせて自動で取得しますので AdSetParamAuto()の方を使用して下さい。また、 **AdSetAndStart**()を使った場合パラメータ設定は関数の内部で行いますのでパラメータ設定は必要ありません。

引 数

HWND **hWnd**	➢ 呼び出し元ウィンドウのハンドル	
WORD **CardType**	➢ A/D カード型式（1：**REX5054U**，2：**REX5054B**）	
BOOL **SampMode**	➢ サンプリング実行モード	
	ポーリングモード	**POLLING_MODE** ：0
	ワンショットモード	**ONESHOT_MODE** ：1
	ポストメッセージ版割込みモード	**INTPOSTMSG_MODE**：2
	リングバッファ割込みモード	**INTHISPEED_MODE**：3
WORD **IOBase**	➢ カードが使用する I/O ベースアドレス	
WORD **IrqNo**	➢ カードが使用する割り込み番号	

戻 値　正常終了時 0 を返します。それ以外はエラーです。

サンプル

```
if ( AdSetParam( hDlg, REX5054U, POLLING_MODE, 0x120, 9 ) != 0 )
{
   MessageBox (hDlg, "カード情報誤り", "パラメータセット", MB_OK|MB_ICONSTOP);
   return -1;
}
```

AdSetParamAuto　カード情報の自動設定

書 式　BOOL **AdSetParamAuto**(HWND **hDlg**, WORD **SampMode**)

機 能　AD カードの型式及びカードが使用する I/O アドレス・割り込み番号をシステムから取得してパラメータの自動設定を行います。

引 数

HWND **hDlg**	➢ 呼び出し元ウィンドウのハンドル	
BOOL **SampMode**	➢ サンプリング実行モード	
	ポーリングモード	**POLLING_MODE** ：0
	ワンショットモード	**ONESHOT_MODE** ：1
	ポストメッセージ版割込みモード	**INTPOSTMSG_MODE**：2
	リングバッファ割込みモード	**INTHISPEED_MODE**：3

戻 値　0；パラメータ設定正常終了
-1；自分のカードを検出できない
-2；サンプリングモードエラー

AdSetTrigger 外部トリガーによる変換開始･終了の設定

書 式 VOID **AdSetTrigger**(WORD **TriggerMode**, WORD **TimingMode**)

機 能 外部からのトリガー信号を使って AD 変換の開始及び終了を行うためのパラメータを設定します。

引 数

WORD **TriggerMode** ➢ 外部トリガーモードの指定

TRIGGER_DISABLE(0)	外部トリガーモードを無効にする
POST_TRIGGER(1)	外部トリガーで変換開始
PRE_TRIGGE(2)	外部トリガーで変換終了
POST_PRE_TRIGGER(3)	外部トリガーで変換を開始・終了

WORD **TimingMode** ➢ 立上がり・立下がりトリガーモードの指定

PULSE_FALLING(0)	信号の立下りでトリガー
PULSE_RISING(1)	信号の立上りでトリガー

戻 値 なし

※ (AdStartIrq) DLL ダイアログによる高速割り込みモード AD 変換実行

書 式 BOOL **AdStartIrq**(HWND **hDlg**, LPVOID **pMem**, WORD **IrqSetCount**)

機 能 DLL のパラメーター覧ダイアログを表示し高速割り込みモードの A/D 変換を実行します。
VXD ドライバーは指定個数の A/D 変換が終了するとユーザ定義メッセージを DLL へポストします。DLL のパラメーター覧ダイアログからグラフボタンをクリックするとグラフを表示します。
本関数は変換データを格納するためのメモリを **AdAllocDataMem**()を呼び出して確保しています。呼び出し元プログラムは、終了時必ず **AdFreeDataMem**()を呼び出し本関数で内部的に確保したメモリを解放して下さい。

引 数

HWND	**hDlg**	➢ 呼び出し元ウィンドウのハンドル
LPVOID	**pMem**	➢ 変換データ格納先メモリを示すポインタ
WORD	**IrqSetCount**	➢ 割り込み要求を行う FIFO 変換データ個数を指定 0 が指定されている場合は、ドライバーが最適な個数を自動設定します。手動設定する場合は 2～10 の範囲で指定して下さい。FIFO の長さは 32 ワードですが、ドライバーは一度の割り込み処理で 10 個以上は取り出しません(オーバーラン対策)。

戻 値 0；変換正常終了
-1；FIFO オーバーランによる変換終了

AdStartIrqPostMsgMode　割込み同期ユーザ-定義メッセージモード AD 変換実行

書 式　BOOL **AdStartIrqPostMsgMode**(HWND **hDlg**, WORD **IrqSetCount**)

機 能　割り込みに同期したユーザ-定義メッセージを受け取りながら AD 変換を実行します。FIFO の変換データが指定個数に達するとドライバーはユーザ側アプリケーションにメッセージをポストします。毎回呼び出し元プログラムへメッセージポストを行いますのでシステムのオーバーヘッドが大きくなります。変換スピードを上げて行くとメッセージ処理が遅れ最終的にシステムがクラッシュします。ユーザ定義メッセージ受け取り時、wParam に AD 変換メッセージコードが、lParam に FIFO から取り出した変換データ数がセットされています。

[メッセージコード] **AD_MODE_RUN** : 通常モードでの変換スタート
AD_MODE_TRGRUN : 開始トリガーにより変換スタート
AD_MODE_STOP : 指定個数の変換完了によるストップ
AD_MODE_PRESTOP : 終了トリガーによる変換ストップ
AD_MODE_OVRUN : オーバーランによる変換ストップ

呼び出し元プログラムは、終了時は必ず **AdStopIrqMode**()を呼び出して変換動作をクリアして下さい。

引 数　HWND **hDlg** ➢ ユーザ定義メッセージをポストするウィンドウのハンドル
WORD **IrqSetCount** ➢ 割り込み要求を行う FIFO 変換データ個数を指定
0 が指定されている場合は、ドライバーが最適な個数を自動設定します。手動設定する場合は 2～10 の範囲で指定して下さい。FIFO の長さは 32 ワードですが、ドライバーは一度の割り込み処理で 10 個以上は取り出しません(オーバーラン対策)。

戻 値　0 : 正常終了
-1 : AD コンバージョンチップ設定エラー
-4 : VPICD 割り込み登録エラー
-5 : 割り込み発生 FIFO データ個数設定エラー

AdStartIrqRingMode　リングバッファ割込み AD 変換実行

書 式　BOOL **AdStartIrqRingMode**(HWND **hDlg**, WORD **IrqSetCount**)

機 能　リングバッファを使った割り込みによる AD 変換入力を開始します。ドライバーは割り込み要求があると FIFO から変換データを取り出して指定のリングバッファに格納します。
リングバッファに格納された変換データは **AdGetRingBuf**()を呼び出していつでも取得することができます。
呼び出し元プログラムは、終了時は必ず **AdStopIrqMode**()を呼び出して変換動作をクリアして下さい。
変換の開始及び変換データ個数が指定個数に達するとドライバーから呼び出し元プログラムにメッセージがポストされます。この時、wParam に AD 変換開始終了メッセージコードが、lParam に終了時の変換データ数がセットされています。

[メッセージコード]
- **AD_MODE_RUN** : 通常モードでの変換スタート
- **AD_MODE_TRGRUN** : 開始トリガーにより変換スタート
- **AD_MODE_STOP** : 指定個数の変換完了によるストップ
- **AD_MODE_PRESTOP** : 終了トリガーによる変換ストップ
- **AD_MODE_OVRUN** : オーバーランによる変換ストップ

引 数
- HWND **hDlg** ➢ ユーザ定義メッセージをポストするウィンドウのハンドル
- WORD **IrqSetCount** ➢ 割り込み要求を行う FIFO 変換データ個数を指定
  0 が指定されている場合は、ドライバーが最適な個数を自動設定します。手動設定する場合は 2～10 の範囲で指定して下さい。FIFO の長さは 32 ワードですが、ドライバーは一度の割り込み処理で 10 個以上は取り出しません(オーバーラン対策)。

戻 値
- 0 : 正常終了
- -1 : AD コンバージョンチップ設定エラー
- -4 : VPICD 割り込み登録エラー
- -5 : 割り込み発生 FIFO データ個数設定エラー

AdStartIrqVxdMode	高速割込みモード AD 変換実行

書 式 BOOL **AdStartIrqVxdMode**(HWND **hDlg**, WORD **IrqSetCount**)

機 能 Windowsシステム側処理のオーバーヘッド伴わず高速に割り込みモードA/D変換の実行します。ドライバーは指定個数の A/D 変換が終了するまで呼び出し元プログラムに一切の通知を行いません。変換の開始及び変換データ個数が指定個数に達するとドライバーから呼び出し元プログラムに変換終了メッセージがポストされます。
この時、wParam に AD 変換開始終了メッセージコードが、lParam に終了時の変換データ数がセットされています。

[メッセージコード] **AD_MODE_RUN** : 通常モードでの変換スタート
AD_MODE_TRGRUN : 開始トリガーにより変換スタート
AD_MODE_STOP : 指定個数の変換完了によるストップ
AD_MODE_PRESTOP : 終了トリガーによる変換ストップ
AD_MODE_OVRUN : オーバーランによる変換ストップ

引 数
- HWND **hDlg** ➢ ユーザ定義メッセージをポストするウィンドウのハンドル
- WORD **IrqSetCount** ➢ 割り込み要求を行う FIFO 変換データ個数を指定
  0 が指定されている場合は、ドライバーが最適な個数を自動設定します。手動設定する場合は 2～10 の範囲で指定して下さい。FIFO の長さは 32 ワードですが、ドライバーは一度の割り込み処理で 10 個以上は取り出しません(オーバーラン対策)。

戻 値
0 : 正常終了
-1 : AD コンバージョンチップ設定エラー
-4 : VPICD 割り込み登録エラー
-5 : 割り込み発生 FIFO データ個数設定エラー

AdStartLimitTrgMode リミットトリガモード AD 変換実行

書 式 BOOL APIENTRY **AdStartLimitTrgMode**(HWND **hDlg**)

機 能 リミットトリガモードの A/D 変換を実行します。
繰り返し実行する場合は、**AdRestartLimitTrgMode()**を呼び出します。
呼び出し元プログラムは、終了時は必ず **AdStopLimitTrgMode()**を呼び出して変換動作をクリアして下さい。
リミットトリガモードの A/D 変換が開始するとユーザ定義メッセージ WM_VXDEVENT メッセージがユーザープログラムにポストされます。この時追加情報 1(wParam)には AD_MODE_RUN がセットされています。リミットポイント発生後指定時間の変換が終了すると、追加情報 1 に AD_MODE_STOP がセットされたユーザ定義メッセージがポストされます。この時、追加情報 2(lParam)には変換データを取り出す変換データ取り出し先頭アドレスがセットされています。
変換データを取り出す場合は、**AdGetTrgData()**に上記取り出し先頭アドレスを指定して取り出しを行って下さい。
FIFO オーバーランによる変換が終了した場合は、追加情報 1 に AD_MODE_OVRUN がセットされたユーザ定義メッセージがポストされます。

引 数 HWND **hDlg** ➢ ユーザ定義メッセージを受取るウィンドウハンドル

戻 値
0 : 正常終了
-1 : LM458 リセットエラー
-2 : LM458 フルキャリブレーションエラー
-3 : 分周回路設定エラー
-4 : サンプリングクロックレジスタの設定エラー
-5 : VPICD 割り込み登録エラー
-6 : DIO コントロールステータスエラー
-7 : 割り込みスタートエラー

AdRestartLimitTrgMode リミットトリガモード AD 変換繰返し実行

書 式 BOOL APIENTRY **AdRestartLimitTrgMode**(HWND **hDlg**)

機 能 リミットトリガモードの A/D 変換を繰り返し実行します。

引 数 HWND **hDlg** ➢ ユーザ定義メッセージを受取るウィンドウハンドル

戻 値
0 : 正常終了
-1 : LM458 リセットエラー
-2 : LM458 フルキャリブレーションエラー
-3 : 分周回路設定エラー
-4 : VPICD 割り込み登録エラー

※ (AdStartPol) DLL ダイアログを使ったポーリングモード AD 変換実行

書 式 BOOL **AdStartPol**(HWND **hWnd**, LPVOID **lpMem**)

機 能 設定されたパラメータの確認ダイアログを表示し、開始ボタン入力でポーリングモードによる AD 変換を実行します。本関数を使ったポーリングモード AD 変換では、CPU を独占してしまいます。CPU を独占したくない場合はチャイルドスレッドを使ったポーリングモード AD 変換 **AdStartPolModeThread**()が用意されていますのでそちらを呼び出して下さい。

引 数
- HWND **hWnd** ➢ 呼び出し元ウィンドウのハンドル
- LPVOID **lpMem** ➢ 変換データ格納先メモリへのポインタ

戻 値 正常終了時 0 を返します。それ以外はエラーです。

※ (AdStartPolMode) ポーリングモード AD 変換実行

書 式 BOOL **AdStartPolMode**(HWND **hWnd**, LPVOID **lpMem**)

機 能 DLL からは何も表示しないでポーリングモードの AD 変換を実行します。本関数を使ったポーリングモード AD 変換では、CPU を独占してしまいます。CPU を独占したくない場合はチャイルドスレッドを使ったポーリングモード AD 変換 **AdStartPolModeThread**()が用意されていますのでそちらを呼び出して下さい。

引 数
- HWND **hWnd** ➢ 呼び出し元ウィンドウのハンドル
- LPVOID **lpMem** ➢ 変換データ格納先メモリへのポインタ

戻 値
- 0 : 正常終了
- -1 : AD コンバージョンチップ設定エラー
- -2 : FIFO オーバーランエラー

※ (AdStartPolModeThread)　ポーリングモード AD 変換チャイルドスレッド

書 式　BOOL **AdStartPolModeThread**(HWND **hDlg**)

機 能　CPU を独占しないでポーリングによる AD 変換を行う場合に、ユーザ側アプリケーションから呼び出すポーリングモード AD 変換を行うチャイルドスレッドです。本チャイルドスレッドは入力変換の開始・終了時と呼び出し元プログラムにメッセージをポストします。
この時、wParam に AD 変換開始終了メッセージコードが、lParam に終了時の変換データ数がセットされています。

[メッセージコード] **AD_MODE_RUN** : 通常モードでの変換スタート
AD_MODE_TRGRUN : 開始トリガーにより変換スタート
AD_MODE_STOP : 指定個数の変換完了によるストップ
AD_MODE_PRESTOP : 終了トリガーによる変換ストップ
AD_MODE_OVRUN : オーバーランによる変換ストップ

引 数　HWND **hDlg**　➢ 呼び出し元ウィンドウのハンドル

戻 値　常に正常終了を返します。

サンプル

```
LRESULT CALLBACK AdStartPolMyDlg( HWND hDlg,UINT wMsg,WPARAM wParam,LPARAM lParam )
{
  switch( wMsg )
  {
  case WM_INITDIALOG:
    AdSetParamAuto( hDlg, POLLING_MODE );
    AdSetChannel( Channels, Sequence );
    AdSetFreq( hDlg, Stime, msec );
    lpMem = AdAllocDataMem( hDlg, 4096);
   /* チャイルドスレッドからのメッセージ受信時の処理 */
   case WM_VXDEVENT:
      AdcStat = wParam;      /* wParam -> AD 変換開始終了コード */
      AdCount = lParam;      /* lParam -> AD 変換データカウント数 */
      return TRUE;
   case WM_COMMAND:
      switch ( wParam )
      {
      case IDOK:
         /* ポーリングモード A/D 変換を行うスレッドを実行 */
         CreateThread( NULL, 0, MyChildThreadPol, hDlg, 0, &dwChildId   );
         return TRUE;
      }
      return FALSE;
}

DWORD WINAPI MyChildThreadPol( HWND hWnd )
{
   AdStartPolModeThread( hWnd );
   ExitThread( TRUE );
}
```

AdStopIrqMode 割り込み AD 変換停止とリソースを解除

書 式 VOID **AdStopIrqMode**(VOID)

機 能 割り込みモードによる変換を停止しシステムに登録した割り込みリソースを解除します。
AdStartIrqRingMode(),**AdStartIrqVxdMode**(),**AdStartIrqPostMsgMode**() を 呼 び出したプログラムは終了時は必ず割り込み登録をクリアして終了して下さい。

引 数 なし

戻 値 なし

AdStopLimitTrgMode リミットトリガモード AD 変換停止とリソース解除

書 式 VOID **AdStopLimitTrgMode**(VOID)

機 能 リミットトリガモードの A/D 変換を終了しリソースを解放します。

引 数 なし

戻 値 なし

HexToVal 変換データの電圧値換算

書 式 double **HexToVal**(WORD **BitData**)

機 能 AD 変換カードの型式を内部判別して 12 ビット AD 変換データを電圧値に換算します。

引 数 WORD **BitData** ➢ 12 ビット AD 変換データ

戻 値 電圧値

HexToValU/B カード型式別変換データの電圧値換算

書 式 double **HexToValU**(WORD **BitData**)
double **HexToValB**(WORD **BitData**)

機 能 12 ビット AD 変換データを電圧値に換算します。AD 変換カードのが REX-5054U の場合は **HexToValU**()を、REX-5054B の場合は **HexToValB**()を呼び出して下さい。

引 数 WORD **BitData** ➢ 12 ビット AD 変換データ

戻 値 電圧値

OutPort　1バイトをポート出力

書 式　VOID **OutPort**(WORD **IOAddr**, WORD **OutVal**)

機 能　**１バイトをポートに出力**

引 数　WORD **IOAddr** ➢ **ポート番号**
WORD **OutVal** ➢ **バイト出力値（上位バイトは無視）**

戻値　**なし**

wOutPort　1ワードをポート出力

書 式　VOID **wOutPort**(WORD **IOAddr**, WORD **OutVal**)

機 能　**１ワードをポートに出力**

引 数　WORD **IOAddr** ➢ **ポート番号**
WORD **OutVal** ➢ **ワード出力値**

戻値　**なし**

InPort　1バイトをポート入力

書 式　WORD **InPort**(WORD **IOAddr**)

機 能　**ポートから１バイト読み込む**

引 数　WORD **IOAddr** ➢ **ポート番号**

戻値　**ポートから読み込んだバイトデータ**

wInPort　1ワードをポート入力

書 式　WORD **wInPort**(WORD **IOAddr**)

機 能　**ポートから１ワード読み込む**

引 数　WORD **IOAddr** ➢ **ポート番号**

戻値　**ポートから読み込んだワードデータ**

VbGetRingBuf　リングバッファから変換データを転送 (Visual BASIC)

書 式　BOOL **VbGetRingBuf**(LPWORD **VbMem**, BOOL **dwOffset**, BOOL **dwGetDataNum**)

機 能　リングバッファから指定個数の変換データをリードし指定のアドレスへ転送します。データの転送先アドレスは VbMem から Offset で示される変換データ個分オフセットした場所になります。
本関数は第三引数で指定した個数分の変換データがリングバッファにない場合は格納されている個数分を転送します。

引 数
- LPWORD **VbMem** ➢ VB 内部で確保されているデータ格納先のアドレス
- BOOL **dwOffset** ➢ データ格納先のアドレスのオフセット
- BOOL **dwGetDataNum** ➢ 取得するデータ数

戻 値　転送したの変換データ個数を返します。 リングバッファが空の時は 0 を、リングバッファオーバーフロー発生時 -1 を返します。

VbGetRingBufConst　リングバッファから一定個数変換データを転送 (Visual BASIC)

書 式　BOOL **VbGetRingBufConst**(LPWORD **VbMem**, BOOL **dwOffset**, BOOL **dwGetDataNum**)

機 能　リングバッファから指定個数の変換データをリードし指定のアドレスへ転送します。データの転送先アドレスは VbMem から Offset で示される変換データ個分オフセットした場所になります。
本関数は **AdGetRingBuf()**とは異なり、第三引数で指定した個数分の変換データがリングバッファにない場合は転送を行わず、戻り値として 0 を返します。

引 数
- LPWORD **VbMem** ➢ VB 内部で確保されているデータ格納先のアドレス
- BOOL **dwOffset** ➢ データ格納先のアドレスのオフセット
- BOOL **dwGetDataNum** ➢ 取得するデータ数

戻 値　転送した変換データ個数を返します。
リングバッファが空の場合もしくは第三引数で指定した個数分の変換データがリングバッファにない場合は 0 を返します。リングバッファオーバーフロー発生時には－１を返します。

VbMemCopy　Visual BASIC の配列へのデータ転送

書 式　LPVOID **VbMemCopy**(LPVOID **VbMem**, LPVOID **DllMem**, DWORD **CopyByteSize**)

機 能　Visual BASIC で確保した配列へ DLL 内部に確保されているメモリから指定バイトサイズ転送します。

引 数

LPVOID	**VbMem**	➢ コピー先メモリアドレス
LPVOID	**DllMem**	➢ コピー元メモリアドレス
DWORD	**CopyByteSize**	➢ コピーバイト数

戻 値　コピー先メモリアドレスを返します。

※（VbPolModeChildThread）　ポーリングモード AD 変換チャイルドスレッド起動

書 式　VOID **VbPolModeChildThread**(HWND **hWnd**)

機 能　Visual BASIC アプリケーションからポーリングモードの A/D 変換を行うチャイルドスレッドを呼び出します。

引 数

HWND	**hWnd**	➢ 呼び出し元ウィンドウのハンドル

戻 値　なし

2-3-2. Visual C サンプルプログラム解説

REX-5054U/B AD PC カードを制御するアプリケーションを開発する場合は、本製品添付のソースプログラム（*.c）を参考にしてを下さい。
Visual C で作成したアプリケーションプログラムから 32Bit 版 DLL を呼び出すためには、以下の作業を行って下さい。

☞ アプリケーションの実行ディレクトリまたは \WINDOWS\SYSTEM に、ADLIB32.DLL ライブラリと REXVPCD.VXD ドライバーをコピーする。

添付のサンプルプログラムを使用してプログラム作成・修正する場合は、各サンプルのフォルダ内にあるファイル（*.C、*.RC、RESOURCE.H、REXICON.ICO）と DLL32 フォルダ内のファイル（ADLIB32.H、ADLIB32.LIB）を新規プロジェクトに追加してコンパイルを行ってください。

AD 変換を行うモードには大きく分けて次の二つがあります。

→ポーリングモード
プログラムから AD カードの FIFO に変換データがあるかどうか調べ、変換データがあればデータを取り出すモード
注）本モードは Windows システムパフォーマンス上の問題より、できる限り使用しないで割り込みモードを使用して下さい。

→割り込みモード
FIFO に設定した個数の変換データがセットされた時点で AD カードから割り込み要求を行い割り込み処理で FIFO からデータを取り出すモード

本製品には下記モードのサンプルプログラムが添付されています。

ポーリングモード	
OneShot	ワンショットモード AD 変換プログラム
PolMode	ポーリングモード AD 変換を行うプログラム
割り込みモード	
IntBase	割り込みモード AD 変換プログラム
IntPost	割り込み同期ポストメッセージモード AD 変換プログラム
RingMode	リングバッファを使った割り込みモード AD 変換プログラム
LimitTrg	リミットトリガーモード AD 変換プログラム

OneShot → ワンショットモード AD 変換プログラム

ワンショットモードでは、AdOneShot()を呼び出しポーリングモードで 1 回だけサンプリングを行い電圧値をダイアログ画面に表示します。繰り返し計測は、指定の繰返し周期でデータを取得するためにタイマー処理で AdOneShot()を呼び出し、指定回数分だけ変換を繰返します。

サンプルプログラム抜粋

```
BOOL Dlg_OnInitDialog (HWND hwnd, HWND hwndFocus, LPARAM lParam)
{
   /* スロット 0 またはスロット 1 に挿入されている自分のカードのリソース情報を取得する */
   for( SlotNo = 0; SlotNo < 4; SlotNo++ )
      if ( AdGetCardResource( hwnd, SlotNo, &MyCardType, &MyIOBase, &MyIrqNo ) == 0 )
         break;

   /* ワンショット A/D 変換のパラメータ設定 */
   if ( (Status = AdSetOneParam( hwnd, MyCardType, MyIOBase )) != 0 )
   {
       // エラー処理
       return FALSE;
   }
   return TRUE;
}

void Cmd_OnCmdStart ( HWND hwnd )
{
   /* マルチメディアタイマー起動 */
   StartMMTimer( hwnd, uPeriod );
}

int StartMMTimer( HWND hwnd, UINT uPeriod )
{
   /* 指定の周期でデータを取得する為にマルチメディア用ワンショットタイマーを使用する */
   /* この後、指定時間経過後 GetDataProc()がコールバックされる */
   timeSetEvent( uPeriod, uResolution, GetDataProc, (DWORD)hwnd, TIME_PERIODIC );
}
```

＜次のページにつづく＞

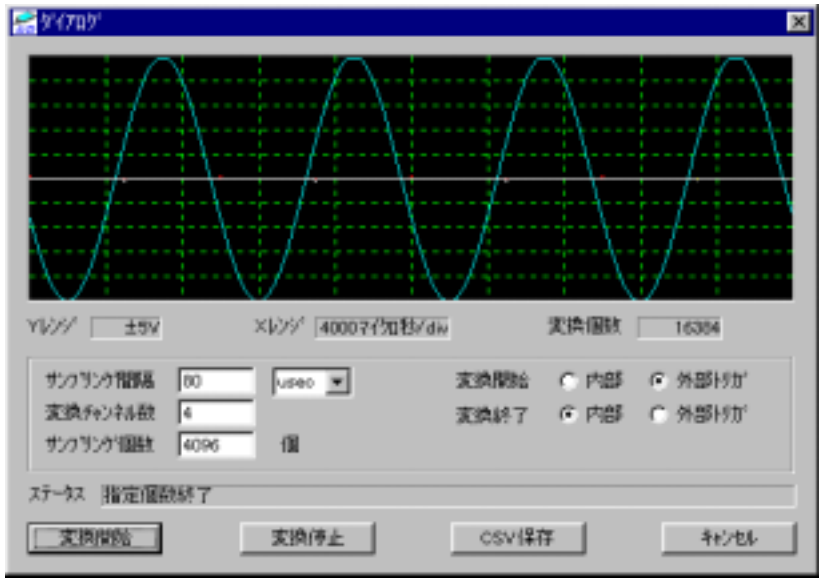
```
void CALLBACK GetDataProc( UINT nTimerId, UINT msg, DWORD dwUser, DWORD dwParam1, DWORD
dwParam2 )
{
   if ( MyAdOneShot ( hwnd ) != 0 )
   {
      StopMMTimer();
      return;
   }
}

int MyAdOneShot ( HWND hwnd )
{
   /* A/D 変換実行（常に全ﾁｬﾝﾈﾙの変換を行います） */
   if ( (Status = AdOneShot( hwnd, AdBuf )) != 0 )
   {  // エラー処理
      return -1;
   }
   /* 電圧値を表示 */
   switch( MyCardType )
   {
   case REX5054U:
      for ( Chloop = 0; Chloop < 8; Chloop++ )
      {
         AdVal = HexToVolt( AdBuf[Chloop] );
         sprintf( szBuf, "%+1.3f", AdVal );
         SetDlgItemText( hwnd, IDS_VOLTCH1+Chloop, szBuf );
      }
      break;
   case REX5054B:
      for ( Chloop = 0; Chloop < 4; Chloop++ )
      {
         // 上記と同処理
      }
      break;
   }
   return 0;
}
```

PolMode → ポーリングモード AD 変換を行うプログラム

DLL でイクスポートしているポーリングモードの A/D 変換ルーチンである**AdStartPolModeThread()**を呼び出すことにより割り込みを禁止して A/D 変換が開始します。AdStartPolModeThread()は変換の開始と終了時ユーザ定義メッセージを呼び出し元ウィンドウへポストします。呼び出されたアプリケーションはこの時の wParam・lParam からステータスと変換個数情報を受け取ります。

A/D 変換データは、**AdAllocDataMem()**でアロケーションしたメモリに格納されます。AdAllocDataMem()はアロケーションしたメモリへのポインター(LPVOID lpMem)を返しますので、ユーザ側のアプリケーションから変換データにアクセスすることが可能です。

変換開始・終了はボタン操作の他に外部トリガを使用することも可能です。



サンプルプログラム抜粋

```
BOOL Dlg_OnInitDialog (HWND hwnd, HWND hwndFocus, LPARAM lParam)
{
 /* スロット 0 またはスロット 1 に挿入されている自分のカードのリソース情報を取得する */
 for( SlotNo = 0; SlotNo < 4; SlotNo++ )
  if ( AdGetCardResource( hwnd, SlotNo, &Adc.MyCardType, &Adc.MyIOBase, &Adc.MyIrqNo ) == 0 )
      break;

 /* カードが挿入されていない場合のエラー処理 */
 if ( SlotNo >= 4 )
 {
  sprintf( szBuf, "REX-5054U/B AD カード検出エラー" );
  SetDlgItemText( hwnd, IDC_STATUS, szBuf );
  return FALSE;
 }
 return TRUE;
}
```

```
void Cmd_OnCmdStart ( HWND hwnd )
{
   /* AD 変換時間 */
   Adc.SampTime = (USHORT)GetDlgItemInt( hwnd, IDE_SAMPTIME, NULL, FALSE );
   /* サンプリング時間 */
   Adc.TimeUnitNo = (USHORT)SendDlgItemMessage( hwnd, IDCB_TIMEUNIT, CB_GETCURSEL, 0, 0L );
   /* AD 変換チャンネル数 */
   Adc.AdcChannel = (USHORT)GetDlgItemInt( hwnd, IDE_CHAN, NULL, FALSE );
   /* AD 変換個数 */
   Adc.dwSampCount = (DWORD)GetDlgItemInt( hwnd, IDE_SAMPCOUNT, NULL, FALSE );
   /* AD 変換パラメータ設定 */
   AdSetParam( hwnd, Adc.MyCardType, POLLING_MODE, Adc.MyIOBase, Adc.MyIrqNo );
   /* 変換チャンネル数設定 */
   AdSetChannel( Adc.AdcChannel, (LPWORD)Sequence );
   /* サンプリング間隔設定 */
   AdSetFreq( hwnd, Adc.SampTime, Adc.TimeUnitNo );
   /* AD データ格納メモリーをアロケート(1 チャンネル当たりの変換個数を指定する) */
   Adc.pData = (LPWORD)AdAllocDataMem( hwnd, Adc.dwSampCount );

   /* ポーリングモード AD 変換を行うスレッドを実行 */
   hThread = CreateThread( NULL, 0, PolChildThread, hwnd, 0, &dwChildId );
}
```

```
DWORD WINAPI PolChildThread( HWND hwnd )
{
   AdStartPolModeThread( hwnd );
   ExitThread( TRUE );
   return 0;       // コンパイルワーニングを除去するためのダミーコード
}
```

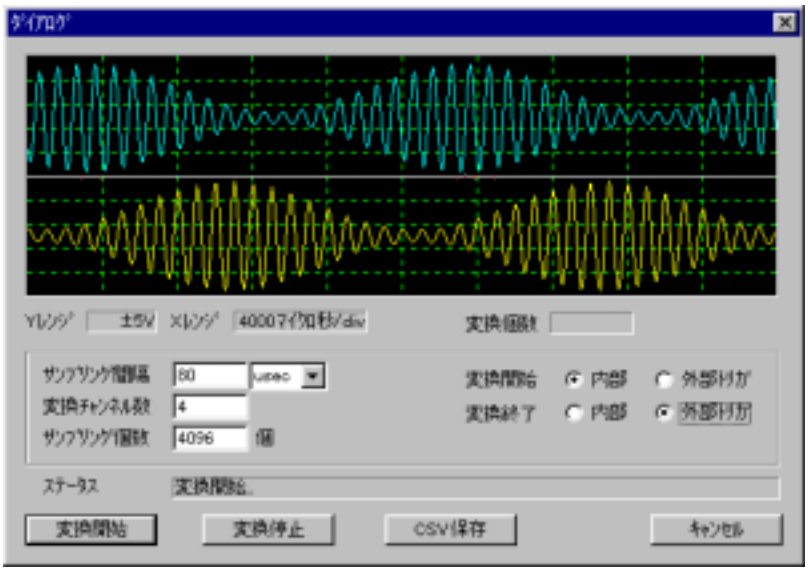
```
void Dlg_OnUserDefineMessage (HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
   AdcStat = wParam;         /* wParam -> AD 変換開始終了コード */
   AdCount = lParam;         /* lParam -> AD 変換データカウント数 */
   switch( AdcStat )
   {
   case AD_MODE_RUN:
      SetDlgItemText( hwnd, IDC_STATUS, "変換開始..." );
      break;
   case AD_MODE_TRGRUN:
      SetDlgItemText( hwnd, IDC_STATUS, "トリガー変換開始" );
      break;
   case AD_MODE_STOP:
      SetDlgItemText( hwnd, IDC_STATUS, "指定個数終了" );
      PlotGraph( hwnd );
      break;
   case AD_MODE_PRESTOP:
      SetDlgItemText( hwnd, IDC_STATUS, "トリガー終了" );
      PlotGraph( hwnd );
      break;
   }
}
```

IntBase → 割り込みモード AD 変換プログラム

AdStartIrqVxdMode()を呼び出すことによりシステム側処理のオーバーヘッドを伴わず高速に割り込みモード AD 変換を実行します。このモードではドライバは指定個数の A/D 変換が終了するまで呼び出し元プログラムに一切の通知を行いません。AdStartIrqVxdMode()は変換の開始及び変換データ個数が指定個数に達するとドライバから呼び出し元プログラムにメッセージをポストします。この時、wParam に AD 変換開始終了メッセージコードが、lParam に終了時の変換データ数がセットされています。

変換開始・終了はボタン操作の他に外部トリガを使用することも可能です。A/D 変換データは、**AdAllocDataMem()**でアロケーションしたメモリに格納されます。AdAllocDataMem()はアロケーションしたメモリへのポインター(LPVOID lpMem)を返しますので、ユーザ側のアプリケーションから変換データにアクセスすることができます。

本サンプルプログラムでは、変換終了後、再び AdStartIrqVxdMode()を呼び出して繰り返し次の割り込みモード A/D 変換を実行しています。



サンプルプログラム抜粋

```
BOOL Dlg_OnInitDialog (HWND hwnd, HWND hwndFocus, LPARAM lParam)
{
 /* スロット 0 またはスロット 1 に挿入されている自分のカードのリソース情報を取得する */
 for( SlotNo = 0; SlotNo < 4; SlotNo++ )
  if ( AdGetCardResource( hwnd, SlotNo, &Adc.MyCardType, &Adc.MyIOBase, &Adc.MyIrqNo ) == 0 )
      break;
 /* カードが挿入されていない場合のエラー処理 */
 if ( SlotNo >= 4 ) {
  sprintf( szBuf, "REX-5054U/B AD カード検出エラー" );
  SetDlgItemText( hwnd, IDC_STATUS, szBuf );
  return FALSE;
 }
 return TRUE;
}
```

```
void Cmd_OnCmdStart ( HWND hwnd )
{
   /* AD 変換時間取得、サンプリング時間取得、AD 変換チャンネル数取得、AD 変換個数取得 */
   /* 変換パラメータ設定 */
   AdSetParam( hwnd, Adc.MyCardType, INTHISPEED_MODE, Adc.MyIOBase, Adc.MyIrqNo );
   /* 変換チャンネル数設定 */
   AdSetChannel( Adc.AdcChannel, (LPWORD)Sequence );
   /* サンプリング間隔設定 */
   AdSetFreq( hwnd, Adc.SampTime, Adc.TimeUnitNo );
   /* AD データ格納メモリーをアロケート(1 チャンネル当たりの変換個数を指定する) */
   Adc.pData = (LPWORD)AdAllocDataMem( hwnd, Adc.dwSampCount );
   /* 割り込みモード AD 変換の実行 */
   fConversionStop = FALSE;
   AdStartIrqVxdMode( hwnd, 0 );
}
```

```
void Cmd_OnCmdStop ( HWND hwnd )
{
   // 本サンプルでは、ストップボタンが押されたことを示すフラグを用意し、
   // Dlg_OnUserDefineMessage()で次の変換を開始しないようにします。
   fConversionStop = TRUE;
}
```

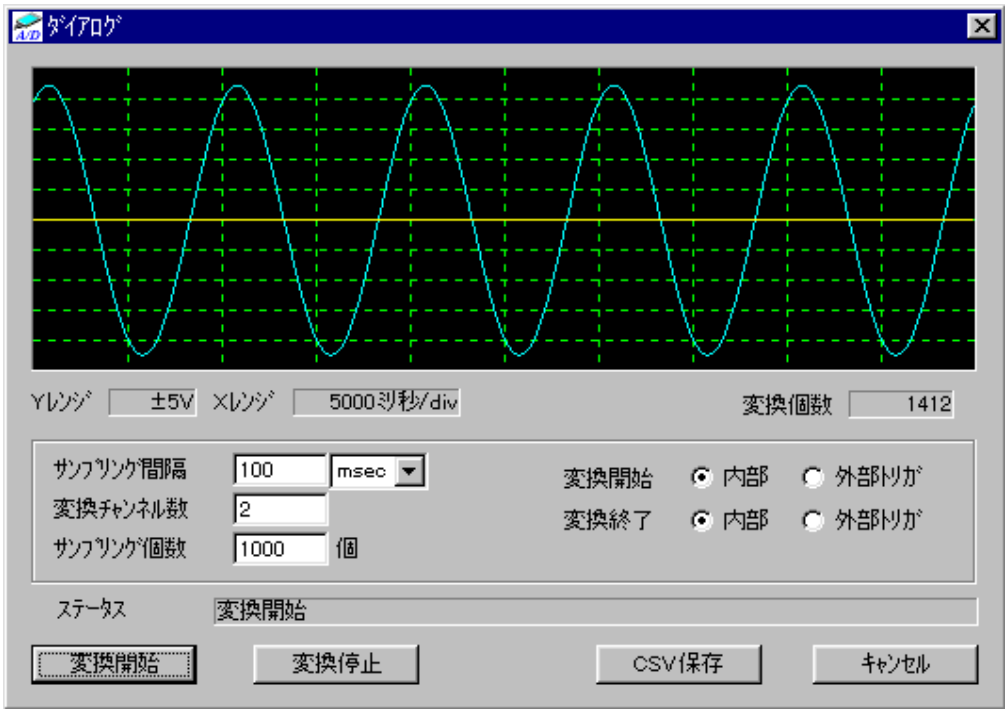
```
void Dlg_OnUserDefineMessage (HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
   AdcStat = wParam;          /* wParam -> AD 変換開始終了コード */
   AdCount = lParam;          /* lParam -> AD 変換データカウント数 */

   switch( AdcStat ){
   case AD_MODE_RUN:          // 変換開始メッセージ表示
      break;
   case AD_MODE_TRGRUN:       // トリガー変換開始メッセージ表示
      break;
   case AD_MODE_PRESTOP:      // トリガー変換終了メッセージ表示
      /* グラフ表示 */
      PlotGraph( hwnd );
      /* 変換を停止し割り込みリソース解放 */
      AdStopIrqMode();
      break;
   case AD_MODE_STOP:         // 変換終了メッセージ表示
      /* グラフ表示 */
      PlotGraph( hwnd );
      /* 変換停止ボタンが押されていないかチェック*/
      if( fConversionStop != TRUE ){
         /* 繰り返し次の割り込みモード AD 変換の実行 */
         AdStartIrqVxdMode( hwnd, 0 );
      }
      else{
         /* 変換を停止し割り込みリソース解放 */
         AdStopIrqMode();
      }
      break;
   }
}
```

IntPost → 割り込み同期ポストメッセージモード AD 変換プログラム

AdStartIrqPostMsgMode()を呼び出すことにより割り込みに同期したポストメッセージ A/D 変換を実行します。このモードでは毎回呼び出し元プログラムへメッセージポストを行いますのでシステムのオーバーヘッドが大きくなります。ドライバは FIFO の変換データが指定個数に達するとユーザ側アプリケーションにメッセージをポストします。この時、wParam に AD 変換開始終了メッセージコードが、lParam に FIFO から取り出した変換データ数がセットされています。

AD 変換データは、AdAllocDataMem()でアロケーションしたメモリに格納されます。AdAllocDataMem()はアロケーションしたメモリへのポインター(LPVOID lpMem)を返しますので、ユーザ側のアプリケーションから変換データにアクセス可能です。

変換開始・終了はボタン操作の他に外部トリガを使用することも可能です。本プログラムでは変換スピードを上げて行くとメッセージ処理が遅れ最終的にシステムがクラッシュする可能性がありますのでサンプリング間隔は 10msec 以上を設定してください。



サンプルプログラム抜粋

```
void Cmd_OnCmdStop ( HWND hwnd )
{
    // 本サンプルでは、ストップボタンが押されたことを示すフラグを用意し、
    // Dlg_OnUserDefineMessage()で次の変換を開始しないようにします。
    fConversionStop = TRUE;
}
```

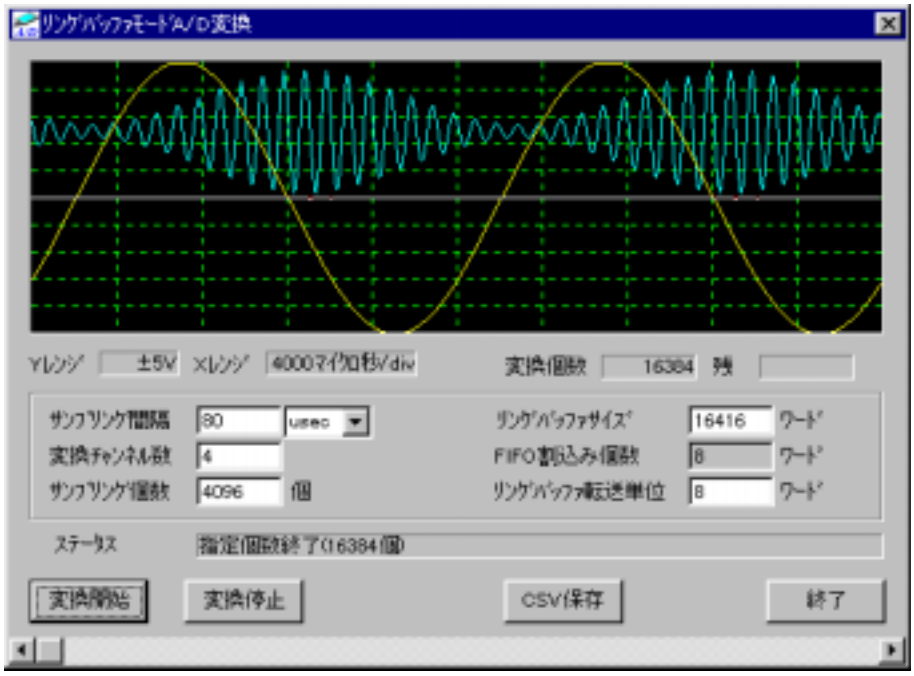
```
void Cmd_OnCmdStart ( HWND hwnd )
{
   /* AD 変換時間取得、サンプリング時間取得、AD 変換チャンネル数取得、AD 変換個数取得 */
   /* 変換パラメータ設定 */
   AdSetParam( hwnd, Adc.MyCardType, INTHISPEED_MODE, Adc.MyIOBase, Adc.MyIrqNo );
   /* 変換チャンネル数設定 */
   AdSetChannel( Adc.AdcChannel, (LPWORD)Sequence );
   /* サンプリング間隔設定 */
   AdSetFreq( hwnd, Adc.SampTime, Adc.TimeUnitNo );

   Adc.pMyBuf = (LPWORD)LocalAlloc( LPTR, Adc.dwSampCount*Adc.AdcChannel*sizeof(POINT) );
   AdSetDataCount( Adc.dwSampCount );

   /* 割り込みモード AD 変換の実行 */
   fConversionStop = FALSE;
   AdStartIrqPostMsgMode( hwnd, Adc.AdcChannel );
}
```

```
void Dlg_OnUserDefineMessage (HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
   /* 変換停止ボタンが押されていないかチェック*/
   if( fConversionStop == TRUE )
   {
      /* 変換を停止し割り込みリソース解放 */
      AdStopIrqMode();
      return;
   }
   AdcStat = wParam;         /* wParam -> AD 変換開始終了コード */
   AdCount = lParam;         /* lParam -> AD 変換データカウント数 */

   switch( AdcStat ){
   case AD_MODE_TRGRUN:      // トリガ変換開始メッセージ表示
      break;
   case AD_MODE_RUN:         // トリガ変換開始メッセージ表示
   case AD_MODE_STOP:
   case AD_MODE_PRESTOP:
      // ドライバーバッファから変換データ取り出し
      GetCount = AdGetRingBufConst( (LPVOID)Adc.pMyBuf, Adc.TotalNum, AdCount );
      // リングバッファから転送したデータを表示
      if ( GetCount > 0 ){
         // 次に書き込む位置（＝現在までに取り出した個数）を更新し表示
         Adc.TotalNum = Adc.TotalNum + GetCount;
         // 累計変換個数を更新
         SetDlgItemInt( hwnd, IDS_TOTALNUM, Adc.TotalNum, FALSE );
      }
      if ( AdcStat == AD_MODE_STOP || AdcStat == AD_MODE_PRESTOP ){
         AdStopIrqMode();       // 変換を停止し割り込みリソース解放
      }
      break;
   case AD_MODE_RINGOVER:
      break;
   case AD_MODE_OVRUN:
      break;
   }
}
```

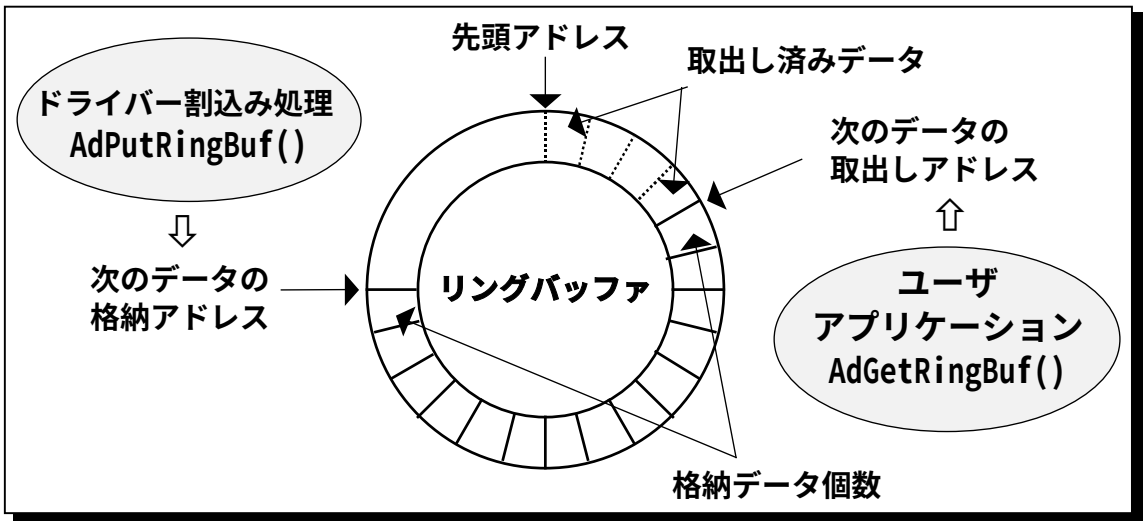
RingMode → リングバッファを使った割り込みモード AD 変換プログラム

リングバッファを使った割り込みモード AD 変換 **AdStartIrqRingMode()**をスタートすると A/D カードは FIFO に指定個数の変換データがセットされると割り込み要求を行います。ドライバーの割り込み処理では FIFO からデータを**AdAllocRingBuf()**で設定されたリングバッファへ転送します。本アプリケーションは、リングバッファからアプリケーションの内部バッファに変換データを転送するためのスレッド **GetRingDataThread()**を作成します。

GetRingDataThread()では終了個数に達するまで**AdGetRingBufConst()**を呼び出し、 指定個数単位で変換データを取得してグラフ表示します。ドライバーがリングバッファへ変換データを転送するレートに対し、アプリケーションがリングバッファからデータを取り出すスピードが遅れるとある時点でリングバッファがオーバーフローします。すなわち、ドライバーはリングバッファの最も古いデータに上書きしますのでそのデータは失われます。



☞リングバッファ構造



サンプルプログラム抜粋

```
BOOL Dlg_OnInitDialog (HWND hwnd, HWND hwndFocus, LPARAM lParam)
{
   /* スロット 0 またはスロット 1 に挿入されている自分のカードのリソース情報を取得する */
   for( SlotNo = 0; SlotNo < 4; SlotNo++ )
      if ( AdGetCardResource( hwnd, SlotNo, &Adc.MyCardType, &MyIOBase, &MyIrqNo ) == 0 )
         break;

   /* カードが挿入されていない場合のエラー処理 */
   if ( SlotNo >= 4 )
   {  /* エラーメッセージ表示 */
      /* 開始ボタンを無効にする処理 */
      return FALSE;
   }
   switch( Adc.MyCardType ){
   case REX5054U:
      sprintf( szBuf, "REX-5054U 正常検出 I/O ベース:0x%x IRQ 番号:%d", MyIOBase, MyIrqNo);
      SetDlgItemText( hwnd, IDC_STATUS, szBuf );
      sprintf( szBuf, "0-2.5V" );
      SetDlgItemText( hwnd, IDS_YRANGE, szBuf );
      break;
   case REX5054B:
      sprintf( szBuf, "REX-5054B 正常検出 I/O ベース:0x%x IRQ 番号:%d", MyIOBase, MyIrqNo);
      SetDlgItemText( hwnd, IDC_STATUS, szBuf );
      sprintf( szBuf, "±5V" );
      SetDlgItemText( hwnd, IDS_YRANGE, szBuf );
      break;
   }
   return TRUE;
}
```

```
void Cmd_OnCmdStart ( HWND hwnd )
{
   /* AD 変換時間取得、サンプリング時間取得、AD 変換チャンネル数取得、AD 変換個数取得 */
   /* ワード単位のリングバッファサイズ */
   /* 変換パラメータ設定 */
   /* カードリソース情報の自動設定 */
   AdSetParamAuto( hwnd, INTHISPEED_MODE )
   /* 変換チャンネル数設定 */
   AdSetChannel( Adc.AdcChannel, (LPWORD)Sequence );
   /* サンプリング間隔設定 */
   AdSetFreq( hwnd, Adc.SampTime, Adc.TimeUnitNo );
   /* RingBufSize ワードのリングバッファ作成 */
   AdAllocRingBuf( hwnd, RingBufSize, Adc.dwSampCount )

   /* リングバッファを使った割込みによる AD 変換開始 */
   AdStartIrqRingMode( hwnd, (USHORT)Adc.IrqUpNum )
   /* リングバッファからデータを転送するためのスレッドを実行 */
   /* スレッドではなくタイマーコールバック処理でデータを取れば CPU 負荷は少なくなります */
   hThread = CreateThread ( NULL, 0, GetRingDataThread, hwnd, 0, &dwChildId ) ;
   return;
}
```

```
DWORD WINAPI GetRingDataThread( HWND hwnd )
{
   while ( BreakFlag == FALSE )
   {
      /* リングバッファから取り出す残りのデータ個数を求める */
      Remain = Adc.MaxSampDataNum - Adc.TotalNum;
      /* リングバッファからデータを取得し自分のデータバッファへ転送する */
      if ( Remain >= (DWORD)Adc.RingGetDataCount )
         /* 残りが RingGetDataCount 以上の場合はアプリで指示された個数単位で転送 */
         AdGetRingBufConst( (LPVOID)Adc.pMyApMem, Adc.TotalNum, Adc.RingGetDataCount );
      else
         /* それ以下の場合は最後の残り個数を転送して終了 */
         AdGetRingBufConst( (LPVOID)Adc.pMyApMem, Adc.TotalNum, Remain );

      /* Polyline()によりグラフ描画処理 */

      /* 指定個数に達したらリングバッファリードを正常終了 */
      if ( (DWORD)Adc.TotalNum >= Adc.MaxSampDataNum )
         break;
   }
   BreakFlag = TRUE;
   return 0;       // コンパイルワーニングを除去するためのダミーコード
}
```

```
void Dlg_OnUserDefineMessage (HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
   /* wParam -> AD 変換終了モード */
   /* lParam -> AD 変換完了データ個数(チャンネル数×変換個数+α)カウント数 */
   NumberOfAdcData = (DWORD)lParam;
   AdcStat = wParam;
   switch ( wParam )
   {
   case AD_MODE_RUN:     sprintf( szBuf, "変換開始", 0 );                          break;
   case AD_MODE_TRGRUN:  sprintf( szBuf, "トリガー変換開始", 0 );                  break;
   case AD_MODE_STOP:    sprintf( szBuf, "指定個数終了(%u 個)", NumberOfAdcData ); break;
   case AD_MODE_PRESTOP: sprintf( szBuf, "トリガー終了(%u 個)", NumberOfAdcData ); break;
   case AD_MODE_RINGOVER:
      sprintf( szBuf, "リングバッファオーバーフロー停止(%u 個)", NumberOfAdcData );  break;
   case AD_MODE_OVRUN:
      sprintf( szBuf, "FIFO オーバーラン停止(%u 個)", NumberOfAdcData );           break;
   }
   SetDlgItemText( hwnd, IDC_STATUS, szBuf );
   return;
}
```

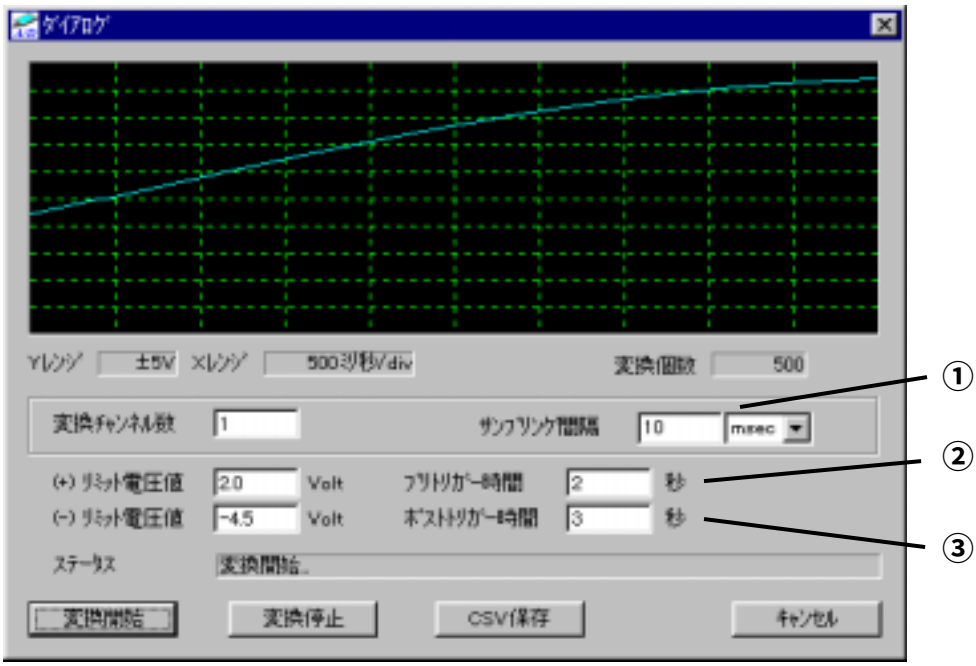
```
void Cmd_OnCmdStop ( HWND hwnd )
{
   BreakFlag = TRUE;              /* スレッドが生きている場合は終了させる */
   AdStopIrqMode();               /* 変換を強制的に止める */
   AdFreeRingBuf();               /* リングバッファを解放 */
   sprintf( szBuf, "変換終了", 0 );
   SetDlgItemText( hwnd, IDC_STATUS, szBuf );
}
```

LimitTrg → リミットトリガモード AD 変換プログラム

[変換開始]ボタンが押されると AdSetLimitTrg()が呼び出され、(+)(-)リミット電圧値で指定したリミットトリガ（入力電圧が設定上限値もしくは下限値を上回った点）をセットします。次に AdAllocTrgBuf()を呼び出してリミットポイント発生の前後で取得する変換データ個数とリミットトリガ用バッファを確保します。取得する変換データ個数はプリトリガー時間（トリガ検出前の時間）、ポストトリガー時間（トリガ検出後の時間）の指定が関係してきます。最後に AdStartLimitTrgMode()を呼び出してリミットトリガモードの A/D 変換を実行します。

リミットトリガモードの A/D 変換が開始するとユーザ定義メッセージがユーザプログラムにポストされます。この時、wParam には AD_MODE_RUN がセットされています。リミットトリガ発生後指定時間の変換が終了すると、AD_MODE_STOP がセットされたユーザ定義メッセージがポストされます。この時、lParam には変換データ取り出し先頭アドレスがセットされています。ここで、AdGetTrgData()を呼び出し、上記アドレスを指定して変換データの取り出しを行います。

本サンプルプログラムでは、AdGetTrgData()による変換データの取出しが完了すると AdRestartLimitTrgMode()を呼び出して繰り返し計測を実行します。



上記サンプルプログラムでは、+2.0V でリミットトリガを検出したときに検出前 2 秒間と検出後 3 秒間のデータを取得、グラフ描画したものです。

1 チャンネル当たりの変換個数 ＝ （ ②＋③ ）／ ①

リミットトリガ検出前の変換データについては②で設定した時間分のデータが存在しない場合、存在する個数しか取得しませんのでご注意ください。

また、③で設定した時間内で再度リミットトリガが発生しても、データ取得中はリミットトリガの検出は行いません。

サンプルプログラム抜粋

```
BOOL Dlg_OnInitDialog (HWND hwnd, HWND hwndFocus, LPARAM lParam)
{
 /* スロット 0 またはスロット 1 に挿入されている自分のカードのリソース情報を取得する */
 for( SlotNo = 0; SlotNo < 4; SlotNo++ )
  if ( AdGetCardResource( hwnd, SlotNo, &Adc.MyCardType, &Adc.MyIOBase, &Adc.MyIrqNo ) == 0 )
      break;

 /* AD 変換時間、AD 変換チャンネル数、プリトリガー時間、ポストトリガー時間の初期値を設定 */
 /* (+)側トリガー電圧、(-)側トリガー電圧の初期値を設定 */
```

```
void Cmd_OnCmdStart ( HWND hwnd )
{
   /* AD 変換時間 */
   Adc.SampTime = (USHORT)GetDlgItemInt( hwnd, IDE_SAMPTIME, NULL, FALSE );
   /* サンプリング時間 */
   Adc.TimeUnitNo = (USHORT)SendDlgItemMessage( hwnd, IDCB_TIMEUNIT, CB_GETCURSEL, 0, 0L );
   /* AD 変換チャンネル数 */
   Adc.AdcChannel = (USHORT)GetDlgItemInt( hwnd, IDE_CHAN, NULL, FALSE );
   /* プリトリガー時間 */
   Adc.PreTime = (USHORT)GetDlgItemInt( hwnd, IDE_PRETRGTIME, NULL, FALSE );
   /* ポストトリガー時間 */
   Adc.PostTime = (USHORT)GetDlgItemInt( hwnd, IDE_POSTTRGTIME, NULL, FALSE );
   /* (+)側トリガー電圧 */
   GetDlgItemText( hwnd, IDE_MAXLIMIT, szBuf, sizeof(szBuf) );
   Adc.MaxLimit = atof( szBuf );
   /* (-)側トリガー電圧 */
   GetDlgItemText( hwnd, IDE_MINLIMIT, szBuf, sizeof(szBuf) );
   Adc.MinLimit = atof( szBuf );

   AdSetParam( hwnd, Adc.MyCardType, INTHISPEED_MODE, Adc.MyIOBase, Adc.MyIrqNo );
   /* 変換チャンネル数設定 */
   AdSetChannel( Adc.AdcChannel, (LPWORD)Sequence );
   /* サンプリング間隔設定 */
   AdSetFreq( hwnd, Adc.SampTime, Adc.TimeUnitNo );
   /* リミットトリガーモードの設定 */
   AdSetLimitTrg( Adc.MaxLimit, Adc.MinLimit, UPPER_LOWER_LIMIT );
   /* リミットトリガー用バッファをアロケート */
   Adc.TrgBufWordSize = AdAllocTrgBuf( Adc.SampTime, Adc.TimeUnitNo, Adc.AdcChannel,
Adc.PreTime, Adc.PostTime );

   /* AD データ格納メモリーをアロケート */
   Adc.pData = LocalAlloc( LPTR, (DWORD)Adc.TrgBufWordSize * 2 );

   /* リミットトリガーAD 変換の実行 */
   AdStartLimitTrgMode( hwnd );
}
```

```
void Dlg_OnUserDefineMessage (HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
   pStartMem = (PWORD)lParam;          /* wParam -> AD 変換終了モード */
   AdcStat = wParam;                   /* lParam -> リングバッファ取り出し先頭アドレス */

   switch( AdcStat ){
   case AD_MODE_RUN:          // 変換開始メッセージ
      break;
   case AD_MODE_STOP:         // 変換終了メッセージ
      /* ﾃﾞｰﾀ取り出し */
      Adc.dwDataNum = AdGetTrgData( pStartMem, Adc.pData );
      if ( Adc.dwDataNum != 0 )
      {
         /* ｸﾞﾗﾌ表示 */

         /* ﾘﾝｸﾞﾊﾞｯﾌｧﾘｾｯﾄとｻﾝﾌﾟﾘﾝｸﾞ再開 */
         AdRestartLimitTrgMode( hwnd );
      }
      break;
   case AD_MODE_OVRUN:        // FIFO オーバーラン停止メッセージ
      break;
   }
}
```

```
void Cmd_OnCmdStop ( HWND hwnd )
{
   SetDlgItemText( hwnd, IDC_STATUS, "変換終了" );
   sprintf( szBuf, "" );
   SetDlgItemText( hwnd, IDS_TOTALNUM, szBuf );
   /* 割り込み登録解除 */
   AdStopLimitTrgMode();
   /* リングバッファを解放 */
   AdFreeTrgBuf();
}
```

2-4. Visual BASIC 言語インターフェイス

2-4-1. DLL ライブラリ API コール

本製品には 32 ビットアプリケーション開発に必要となる API インターフェースを提供する DLL ライブラリ“**ADLIB32.DLL**”と、特権レベルで Windows システムに対し割り込み要求等を行う“**REXVPCD.VXD**”仮想デバイスドライバーが添付されています。32 ビットバージョン Visual BASIC で作成したアプリケーションは“**ADLIB32.DLL**”の API を呼び出します。“**ADLIB32.DLL**”は自分で処理できない要求に対しては“**REXVPCD.VXD**”のファンクションを呼び出して処理を行います。

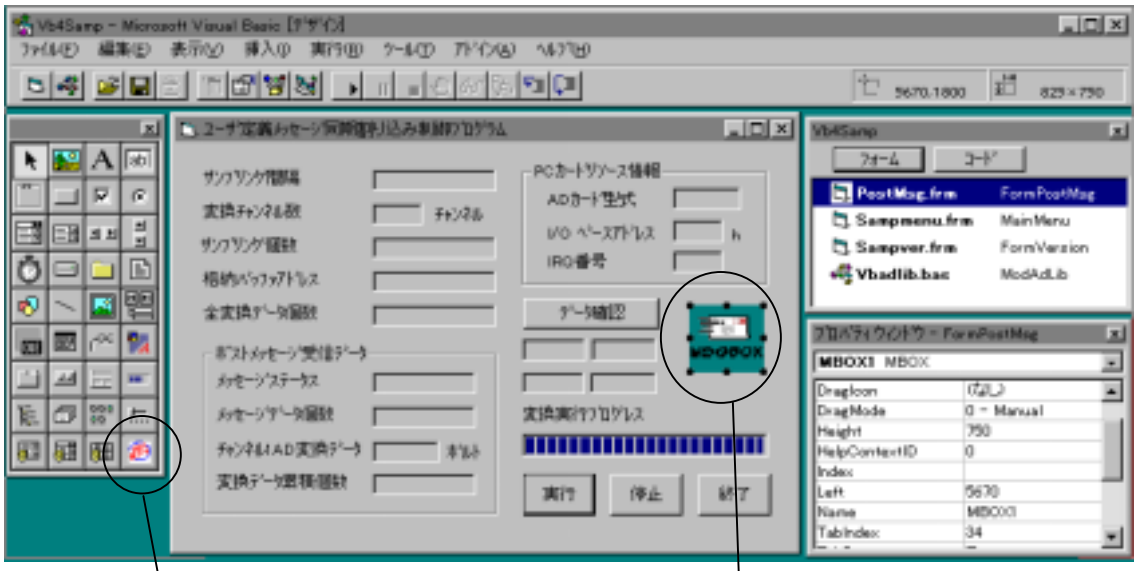
Visual BASIC でアプリケーションを作成する場合、次の二つの内容について理解し、必要となる設定作業を行って下さい。

① DLL ライブラリ API 関数コール

Visual BASIC から“**ADLIB32.DLL**”が提供する API 関数を呼び出すためにはモジュール定義ファイルで各 API 関数を Declare 宣言します。API 関数の Declare 宣言は、製品添付のサンプルプログラム“**VbAdLib.bas**”を参考にして必要部分をコピーして下さい。また、各 API 関数の仕様については「2-3-1.DLL ライブラリ API 解説」を参照して下さい。

② 割り込みモード AD 変換

割り込みモードの AD 変換を行う場合、割り込み要求に同期したユーザ定義メッセージを Visual BASIC で作成したアプリケーションで受け取る必要があります。Visual BASIC ではユーザ定義メッセージを受け取るコントロールは用意されていませんので、本製品に添付されている OLE カスタムコントロール(OCX)“**MBOX.OCX**”を使用します。



2-4-2. カスタムコントロール

Step.1 → 本製品添付ソフトのコピー

OLE カスタムコントロール：**MBOX.OCX** と 32 ビット版 DLL：**AdLib32.DLL** 及び仮想デバイスドライバー：**REXVPCD.VXD** を添付のフロッピーディスクからコピーします。Visual BASIC プログラム作成ディレクトリ名を“MyProg”とします。

```
>COPY “コピー元ドライブ名”:\Win95\Dll32\AdLib32.dll “コピー先ドライブ名”:\Windows\System
>COPY “コピー元ドライブ名”:\Win95\Dll32\Rexvpcd.vxd “コピー先ドライブ名”:\Windows\System
>COPY “コピー元ドライブ名”:\Win95\Dll32\Mbox.ocx “コピー先ドライブ名”:\MyProg
```

Step.2 → OCX のレジストリー登録 （割り込みサービス使用時必須）

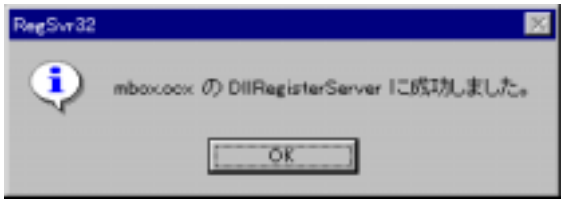
本製品添付の OCX “MBOX.OCX”を VB で使用するためには、VB の CD-ROM に添付されているツール“REGSVR32.EXE”を使って OCX のレジストリ登録を行います。“REGSVR32.EXE”は 32 ビットコンソールアプリケーションですので、Windows の DOS BOX から実行します。尚、“REGSVR32.EXE”は VB の CD-ROM の“TOOLS\PSS”に添付されています。

OCX をレジストリー登録するときは、下記構文で実行します。

```
>REGSVR32 “ドライブ名”:\MyProg\Mbox.ocx
```

OCX をレジストリー登録から削除するときは、“/U”を付けて下記構文で実行します。

```
>REGSVR32 /U “ドライブ名”:\MyProg\Mbox.ocx
```



登録成功メッセージ

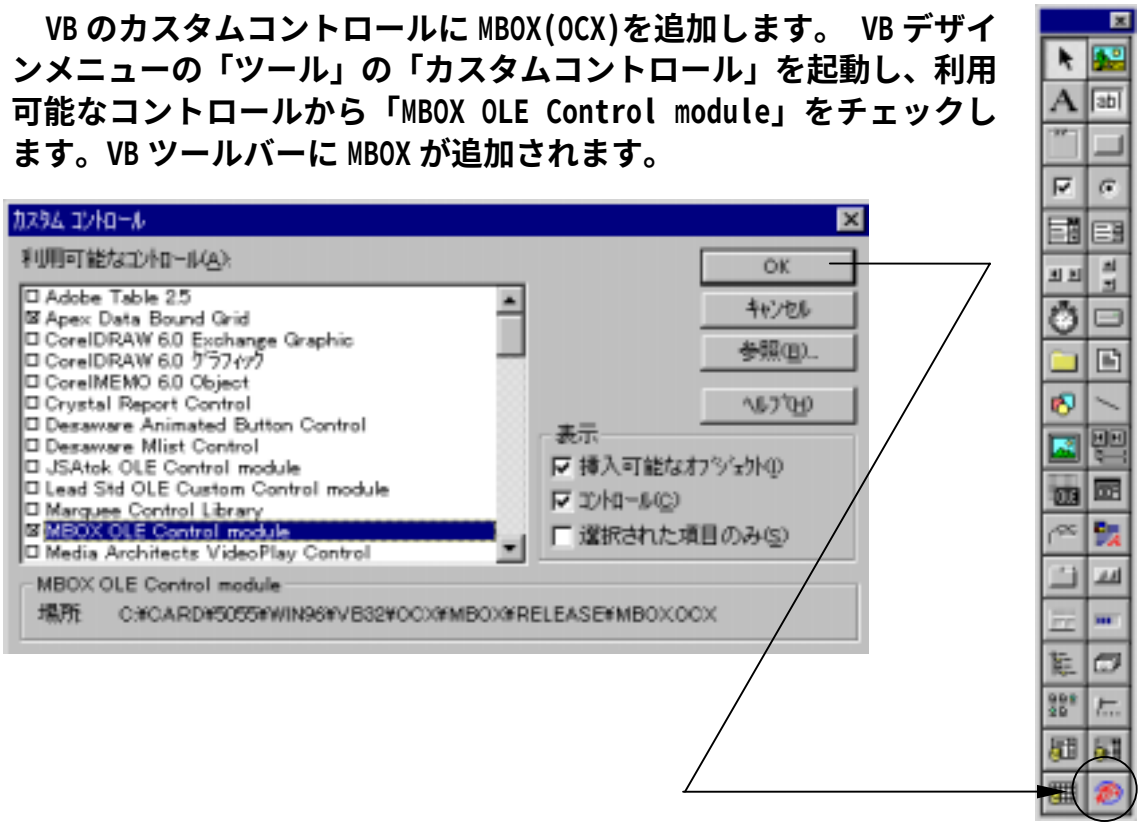
登録削除成功メッセージ

Step.3 → AdLib32.DLL API 関数の Declare 宣言

次に、VB プログラムの作成に入ります。VB デザインメニューから新規プロジェクトを作成し、「ファイル」の「挿入」から標準モジュールを追加します。追加した標準モジュールファイルで DLL 関数の参照宣言を行います。次ページに示す宣言部分を、サンプルプログラム“**VBADLIB.BAS**”からコピーして下さい。

Step.4 → MBOX OLE Control Module の追加　（割り込みサービス使用時必須）

VB のカスタムコントロールに MBOX(OCX)を追加します。 VB デザインメニューの「ツール」の「カスタムコントロール」を起動し、利用可能なコントロールから「MBOX OLE Control module」をチェックします。VB ツールバーに MBOX が追加されます。



☞ AdLib32.DLL 呼び出し関数の宣言

関数の仕様については「2-3-1.DLL ライブラリ API 解説」を参照して下さい。
※(関数名)の印がついた関数については、互換性のために残してある関数ですので新規でプログラムを作成する場合には呼び出す必要はありません。

★★★ ご注意 ★★★
DLL に確保要求して取得したメモリアドレスを渡す場合は
"ByVal MemAdrs As Long"宣言１）を有効にします
VB 内で確保した変数及び配列のアドレスを DLL へ渡す場合は
"VBArrayName As Any"宣言２）を有効にします

AdAllocDataMem	AD 変換データ格納用メモリのアロケーション
Declare Function AdAllocDataMem Lib "Adlib32.dll" (ByVal hwnd As Long, ByVal DataLength As Long) As Long	

AdAllocMem	汎用メモリのアロケーション
Declare Function AdAllocMem Lib "Adlib32.dll" (ByVal hwnd As Long, ByVal MemSize As Long) As Long	

AdAllocRingBuf	リングバッファのアロケーション
Declare Function AdAllocRingBuf Lib "Adlib32.dll" (ByVal hwnd As Long, ByVal wAllocSize As Long, ByVal DataLength As Long) As Long	

AdAllocTrgBuf	入力電圧リミットトリガバッファのアロケーション
Declare Function AdAllocTrgBuf Lib "Adlib32.dll" (ByVal SampTime As Integer, ByVal TimeUnit As Integer, ByVal AdChan As Integer, ByVal PreTime As Integer, ByVal PostTime As Integer) As Long	

※ (AdCheckStopTrig)	外部トリガーによる変換終了チェック
Declare Function AdCheckStopTrig Lib "Adlib32.dll" (ByVal hwnd As Long) As Long	

AdFreeDataMem	AD 変換データ格納用メモリを解放
Declare Sub AdFreeDataMem Lib "Adlib32.dll" ()	

AdFreeMem	汎用メモリの解放
Declare Sub AdFreeMem Lib "Adlib32.dll" (ByVal pHeapMem As Long)	

AdFreeRingBuf	リングバッファの解放
Declare Sub AdFreeRingBuf Lib "Adlib32.dll" ()	

AdFreeTrgBuf	入力電圧リミットトリガバッファの解放
Declare Sub AdFreeTrgBuf Lib "Adlib32.dll" ()	

※ (AdGetCardNameInfo)	PC カードの製品情報名を取得
Declare Function AdGetCardNameInfo Lib "Adlib32.dll" (ByVal hwnd As Long, ByVal SlotNo As Integer, CopyBuf As Any, ByVal CopyBufLen As Integer) As Long	

AdGetCardResource	REX5054U/B に割り当てられているリソースの取得
Declare Function AdGetCardResource Lib "Adlib32.dll" (ByVal hwnd As Long, ByVal SlotNo As Integer, pCardType As Any, pIOBase As Any, pIrqNo As Any) As Long	

AdGetParam	設定されているサンプリングパラメータを取得
Declare Sub AdGetParam Lib "Adlib32.dll" (lpCardType As Any, lpIOAddr As Any, lpIrqNo As Any, lpStime As Any, lpUnit As Any, lpChannels As Any, pDataLen As Any, ppMem As Any)	

AdGetRingBufAdrs	リングバッファの先頭アドレスを取得
Declare Function AdGetRingBufAdrs Lib "Adlib32.dll" () As Long	

AdGetTrgData	入力電圧リミットトリガバッファから変換データを転送
Declare Function AdGetTrgData Lib "Adlib32.dll" (ByVal pStartMem As Long, pCopyMem As Any) As Long	

AdGetVersion	DLL ライブラリのバージョン表示
Declare Sub AdGetVersion Lib "Adlib32.dll" (ByVal hwnd As Long)	

AdOneShot	ワンショットモード AD 変換実行
宣言1) Declare Function AdOneShot Lib "Adlib32.dll" (ByVal hwnd As long, ByVal lpMem As Long) As long **宣言2**) Declare Function AdOneShot Lib "Adlib32.dll" (ByVal hwnd As Long, AdData As Any) As Long	

※ (AdPlot)	サンプリング波形のグラフ表示
Declare Sub AdPlot Lib "Adlib32.dll" (ByVal hwnd As Long, ByVal pDataMem As Long)	

AdSaveCsvFile	Excel CSV ファイル形式でデータ保存
宣言1) Declare Function AdSaveCsvFile Lib "Adlib32.dll" (ByVal hwnd As Long, ByVal MyCardType As Integer, ByVal AdcChannel As Integer, ByVal SampTime As Integer, ByVal TimeUnitNo As Integer, ByVal pDllBuf As Long, ByVal NumberOfAdcData As Long) As Long･**宣言2**) Declare Function AdSaveCsvFile Lib "Adlib32.dll" (ByVal hwnd As Long, ByVal MyCardType As Integer, ByVal AdcChannel As Integer, ByVal SampTime As Integer, ByVal TimeUnitNo As Integer, VbBuf As Any, ByVal NumberOfAdcData As Long) As Long	

AdSaveFile　バイナリ形式で変換データファイル保存
宣言 1) Declare Function AdSaveFile Lib "Adlib32.dll" (ByVal hwnd As Long, ByVal pDllBuf As Long, ByVal NumberOfAdcData As Long) As Long **宣言 2**) Declare Function AdSaveFile Lib "Adlib32.dll" (ByVal hwnd As Long, VbBuf As Any, ByVal NumberOfAdcData As Long) As Long

※ (AdSetAndStart)　DLL ダイアログを使用したポーリングモード AD 変換実行
Declare Function AdSetAndStart Lib "Adlib32.dll" (ByVal hwnd As Long) As Long

AdSetChannel　変換チャンネルパラメータの設定
Declare Function AdSetChannel Lib "Adlib32.dll" (ByVal Channels As Integer, lpSequence As Any) As Long

AdSetDataCount　サンプリング個数の設定
Declare Function AdSetDataCount Lib "Adlib32.dll" (ByVal DataCount As Long) As Long

AdSetExternalClock　外部クロック入力の設定
Declare Function AdSetExternalClock Lib "Adlib32.dll" (ByVal Mode As Long, ByVal Counter0 As Integer, ByVal Counter1 As Integer) As Long

AdSetFreq　サンプリング間隔の設定
Declare Function AdSetFreq Lib "Adlib32.dll" (ByVal hwnd As Long, ByVal Stime As Integer, ByVal Unit As Integer) As Long

AdSetInternalClock　内部クロックの選択とサンプリング間隔の手動設定
Declare Function AdSetInternalClock Lib "Adlib32.dll" (ByVal UseClock As Long, ByVal Counter0 As Integer, ByVal Counter1 As Integer) As Long

AdSetLimitTrg　入力電圧リミットトリガ値の設定
Declare Function AdSetLimitTrg Lib "Adlib32.dll" (ByVal MaxLimit As Double, ByVal MinLimit As Double, ByVal TrgMode As Integer) As Long

AdSetOneParam　ワンショット AD 変換パラメータ設定
Declare Function AdSetOneParam Lib "Adlib32.dll" (ByVal hwnd As Long, ByVal hwnd As Integer, ByVal hwnd As Integer) As Long

AdSetParam　カード情報の手動設定
Declare Function AdSetParam Lib "Adlib32.dll" (ByVal hwnd As Long, ByVal CardType As Integer, ByVal SampMode As Long, ByVal IOBase As Integer, ByVal IrqNo As Integer) As Long

AdSetParamAuto	カード情報の自動設定
Declare Function AdSetParamAuto Lib "Adlib32.dll" (ByVal hwnd As Long, ByVal SampMode As Integer) As Long	

AdSetTrigger	外部トリガによる変換開始･終了の設定
Declare Sub AdSetTrigger Lib "Adlib32.dll" (ByVal TrgMode As Integer, ByVal RiseOrFall As Integer)	

※ (AdStartIrq)	DLL ダイアログによる高速割り込みモード AD 変換実行
宣言1) Declare Function AdStartIrq Lib "Adlib32.dll" (ByVal hwnd As Long, ByVal pMem As Long, ByVal IrqSetCount As Integer) As Long **宣言2**) Declare Function AdStartIrq Lib "Adlib32.dll" (ByVal hwnd As long, lpMem As Any, ByVal IrqSetCount As Integer) As long	

AdStartIrqPostMsgMode	割込み同期ユーザ−定義メッセージモード AD 変換実行
Declare Function AdStartIrqPostMsgMode Lib "Adlib32.dll" (ByVal hwnd As Long, ByVal IrqSetCount As Integer) As Long	

AdStartIrqRingMode	リングバッファ割込み AD 変換実行
Declare Function AdStartIrqRingMode Lib "Adlib32.dll" (ByVal hwnd As Long, ByVal IrqSetCount As Integer) As Long	

AdStartIrqVxdMode	割込みモード AD 変換実行
Declare Function AdStartIrqVxdMode Lib "Adlib32.dll" (ByVal hwnd As Long, ByVal IrqSetCount As Integer) As Long	

AdStartLimitTrgMode	リミットトリガモード AD 変換実行
Declare Function AdStartLimitTrgMode Lib "Adlib32.dll" (ByVal hwnd As Long) As Long	

AdRestartLimitTrgMode	リミットトリガモード AD 変換繰返し実行
Declare Function AdRestartLimitTrgMode Lib "Adlib32.dll" (ByVal hwnd As Long) As Long	

※ (AdStartPol)	DLL ダイアログを使ったポーリングモード AD 変換実行
宣言1) Declare Function AdStartPol Lib "Adlib32.dll" (ByVal hwnd As Long, ByVal lpMem As Long) As Long **宣言2**) Declare Function AdStartPol Lib "Adlib32.dll" (ByVal hwnd As long, AdData As Any) As long	

※ (AdStartPolMode)	ポーリングモード AD 変換実行
宣言1) Declare Function AdStartPolMode Lib "Adlib32.dll" (ByVal hwnd As Long, ByVal lpMem As Long) As Long **宣言2**) Declare Function AdStartPolMode Lib "Adlib32.dll" (ByVal hwnd As long, lpMem As Any) As long	

※ (AdStartPolModeThread)　　ポーリングモード AD 変換チャイルドスレッド
Declare Function AdStartPolModeThread Lib "Adlib32.dll" (ByVal hwnd As Long) As Long

AdStopIrqMode　　割り込み AD 変換停止とリソースを解除
Declare Sub AdStopIrqMode Lib "Adlib32.dll" ()

AdStopLimitTrgMode　　リミットトリガモード AD 変換停止とリソース解除
Declare Sub AdStopLimitTrgMode Lib "Adlib32.dll" ()

HexToVal　　変換データの電圧値換算
Declare Function HexToVal Lib "Adlib32.dll" (ByVal HexVal As Integer) As Double

HexToValU/B　　カード型式別変換データの電圧値換算
Declare Function HexToValU Lib "Adlib32.dll" (ByVal HexVal As Integer) As Double Declare Function HexToValB Lib "Adlib32.dll" (ByVal HexVal As Integer) As Double

OutPort　　バイトをポート出力
Declare Function OutPort Lib "Adlib32.dll" (ByVal IOAddr As Integer, ByVal OutVal As Integer) As Integer

wOutPort　　1ワードをポート出力
Declare Function wOutPort Lib "Adlib32.dll" (ByVal IOAddr As Integer, ByVal OutVal As Integer) As Integer

InPort　　1バイトをポート入力
Declare Function InPort Lib "Adlib32.dll" (ByVal IOAddr As Integer) As Integer

wInPort　　1ワードをポート入力
Declare Function wInPort Lib "Adlib32.dll" (ByVal IOAddr As Integer) As Integer

VbGetRingBuf　　リングバッファから変換データを転送
Declare Function VbGetRingBuf Lib "Adlib32.dll" (AdRing As Any, ByVal Offset As Long, ByVal GetDataNum As Long) As Long

VbGetRingBufConst　　リングバッファから一定個数変換データを転送
Declare Function VbGetRingBufConst Lib "Adlib32.dll" (AdRing As Any, ByVal Offset As Long, ByVal GetDataNum As Long) As Long

VbMemCopy　　Visual BASIC の配列へのデータ転送
Declare Function VbMemCopy Lib "Adlib32.dll" (VbMem As Any, ByVal DllMem As Long, ByVal CopyByteSize As Long) As Long

※（VbPolModeChildThread）ポーリングモード AD 変換チャイルドスレッド起動
Declare Sub VbPolModeChildThread Lib "Adlib32.dll" (ByVal hwnd As Long)

2-4-3. Visual BASIC サンプルプログラム解説

Visual BASIC 4.0 以上のバージョンを使って REX-5054U/B AD PC カードを制御するアプリケーションを開発する場合は、本製品に添付されているサンプルプログラムを参考にして下さい。
添付のサンプルプログラムを使用してプログラム作成・修正する場合は、各サンプルの FRM ファイルで新規プロジェクトを作成し、標準モジュール（VBADLIB.BAS）をプロジェクトに追加してコンパイルを行ってください。

AD 変換を行うモードには大きく分けて次の二つがあります。

∅**ポーリングモード**
プログラムから AD カードの FIFO に変換データがあるかどうか調べ、変換データがあればデータを取り出すモード
∅**割り込みモード**
FIFO に設定した個数の変換データがセットされた時点で AD カードから割り込み要求を行い割り込み処理で FIFO からデータを取り出すモード

以下のモードのサンプルプログラムが製品添付されています。

ポーリングモード	
OneShot	ワンショットモード AD 変換プログラム
PolMode	ポーリングモード AD 変換を行うプログラム
割り込みモード	
IntBase	割り込みモード AD 変換プログラム
RingMode	リングバッファを使った割り込みモード AD 変換プログラム
LimitTrg	リミットトリガーモード AD 変換プログラム

OneShot → ワンショットモード AD 変換プログラム

ワンショットモードでは、AdOneShot()を呼び出しポーリングモードで 1 回だけサンプリングを行い電圧値をダイアログ画面に表示します。繰り返し計測は、指定の繰返し周期でデータを取得するためにタイマー処理で AdOneShot()を呼び出し、指定回数分だけ変換を繰返します。

サンプルプログラム抜粋

```
Private Sub IDB_START_Click()

    'タイマー起動
    uPeriod = IDE_PERIOD.Text
    OneShotTimer.Interval = uPeriod

End Sub
```

```
Private Sub OneShotTimer_Timer()

    If MyAdOneShot(hwnd) <> 0 Then
        Exit Sub
    End If
End Sub
```

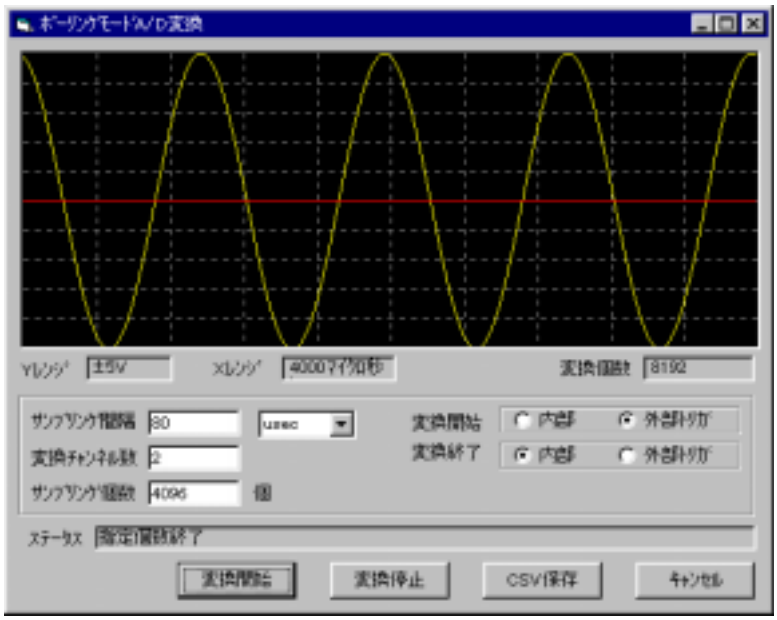
```
Private Function MyAdOneShot(ByVal hwnd As Long) As Long

    ' A/D 変換実行（常に全ﾁｬﾝﾈﾙの変換を行います）
    Status = AdOneShot(hwnd, AdBuf(0))
    If Status <> 0 Then
        IDS_STATUS.Caption = "ﾜﾝｼｮｯﾄ A/D 変換ｴﾗｰ [CODE:" + Str(Status) + "]"
        MyAdOneShot = -1    '戻り値
    End If

    ' 電圧値を表示
    Select Case MyCardType
    Case REX5054U
            AdVal = ((AdBuf(Chloop) And &H1FFF) * (2 ^ 3)) * 2.5 / 32768#
    Case REX5054B:
            AdVal = ((AdBuf(Chloop) And &H1FFF) * (2 ^ 3)) * 10# / 32768# - 5#
    End Select
    MyAdOneShot = 0   '戻り値
End Function
```

PolMode → ポーリングモード AD 変換を行うプログラム

DLL でイクスポートしているポーリングモードの A/D 変換ルーチンである **AdStartPolModeThread()**を呼び出すことにより割り込みを禁止して A/D 変換が開始します。AdStartPolModeThread()は変換の開始と終了時ユーザ定義メッセージを呼び出し元ウィンドウへポストします。呼び出されたアプリケーションはこの時の wParam・lParam からステータスと変換個数情報をコントロール MSGBOX を配置して受け取るようにします。

A/D 変換データは、**AdAllocDataMem()**でアロケーションしたメモリに格納されます。AdAllocDataMem()はアロケーションしたメモリへのポインターを返しますので、ユーザ側のアプリケーションから変換データにアクセスします。

VB アプリケーション側ではデータを保管するための配列を確保します。**VbMemCopy()**を呼び出してドライバがセットした変換データを自分のメモリへ転送保管します。



変換開始・終了はボタン操作の他に外部トリガを使用することも可能です。

サンプルプログラム抜粋

```
Private Sub PolTimer_Timer()
    Dim OleHandle As Long           ' MBOX.OCX ハンドル

    'OLE のウィンドウハンドル取得
    OleHandle = MBOX1.GetMboxWnd()

    'ポーリングモード AD 変換実行
    AdStartPolModeThread OleHandle
    'タイマー停止
    PolTimer.Interval = 0

End Sub
```

```
Private Sub IDB_START_Click()

    ' AD 変換時間、サンプリング時間、AD 変換チャンネル数、AD 変換個数の取得
    ' 変換パラメータのセット
    AdSetParam hwnd, Adc.MyCardType, INTHISPEED_MODE, Adc.MyIOBase, Adc.MyIrqNo

    ' 変換チャンネル数設定
    AdSetChannel Adc.AdcChannel, Sequence(1)
    ' 外部トリガで開始、指定個数で終了
    AdSetTrigger POST_TRIGGER, PULSE_RISING
    ' サンプリング間隔設定
    AdSetFreq hwnd, Adc.SampTime, Adc.TimeUnitNo
    ' VB アプリケーション側でデータを保管するための配列を確保
    Adc.MaxSampDataNum = Adc.AdcChannel * Adc.dwSampCount
    ReDim Adc.pData(Adc.MaxSampDataNum) As Integer

    ' AD データ格納メモリーをアロケート(1 チャンネル当たりの変換個数を指定する)
    DllMem = AdAllocDataMem(hwnd, Adc.dwSampCount)
    If DllMem = 0 Then
        Exit Sub
    End If
    '10msec 間隔で PolTimer を起動
    PolTimer.Interval = 10
End Sub
```

```
Private Sub MBOX1_OnMsgPost(ByVal wParam As Integer, ByVal lParam As Long)

    AdCount = lParam            ' lParam -> AD 変換データカウント数
    AdcStat = wParam            ' wParam -> AD 変換開始終了コード

    Select Case AdcStat
    Case AD_MODE_RUN            ' 変換開始メッセージ
        Exit Sub
    Case AD_MODE_TRGRUN         ' トリガー変換開始メッセージ
        Exit Sub
    Case AD_MODE_STOP           ' 指定個数終了メッセージ
    Case AD_MODE_PRESTOP        ' トリガー終了メッセージ
    Case AD_MODE_OVRUN       ' FIFO オーバーラン停止メッセージ
    End Select

    ' ドライバがセットした変換データを自分のメモリへ転送保管します
    ' 1 個の変換データは 2 バイトですので転送先のアドレスに注意して下さい
    VbMemCopy Adc.pData(0), DllMem, AdCount * 2
    ' グラフ描画処理
    PicGraph.Cls    '一旦消去
    PlotData
    'AD データ格納メモリー開放
    AdFreeDataMem

End Sub
```

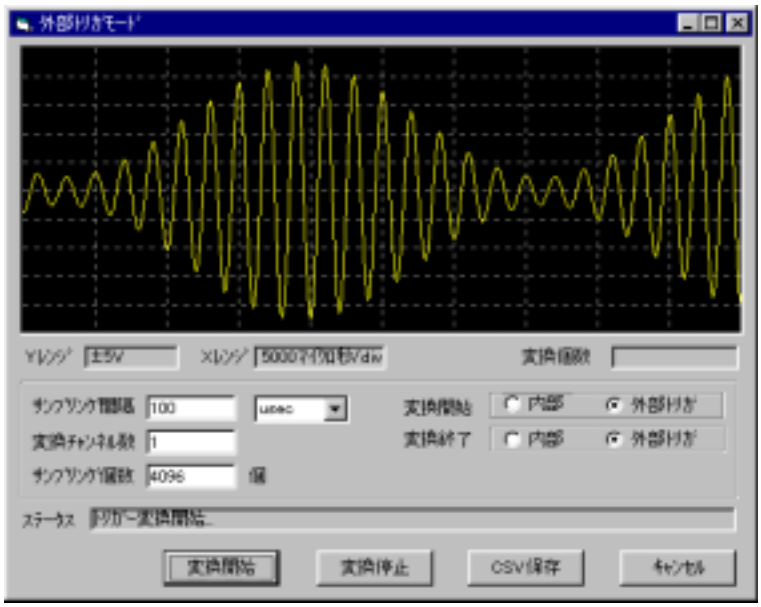
IntBase → 割り込みモード AD 変換プログラム

AdStartIrqVxdMode()を呼び出すことによりシステム側処理のオーバーヘッドを伴わず高速に割り込みモード AD 変換を実行します。このモードではドライバは指定個数の A/D 変換が終了するまで呼び出し元プログラムに一切の通知を行いません。AdStartIrqVxdMode()は変換の開始及び変換データ個数が指定個数に達するとドライバから呼び出し元プログラムにメッセージをポストします。この時、wParam に AD 変換開始終了メッセージコードが、lParam に終了時の変換データ数がセットされています。コントロール MSGBOX を配置してこれらを受け取るようにします。

A/D 変換データは、AdAllocDataMem()でアロケーションしたメモリに格納されます。AdAllocDataMem()はアロケーションしたメモリへのポインターを返しますので、ユーザ側のアプリケーションから変換データにアクセスすることができます。VB アプリケーション側ではデータを保管するための配列を確保します。VbMemCopy()を呼び出してドライバがセットした変換データを自分のメモリへ転送保管します。

本サンプルプログラムでは、変換終了後、再び AdStartIrqVxdMode() を呼び出して繰り返し次の割り込みモード A/D 変換を実行しています。

変換開始・終了はボタン操作の他に外部トリガを使用することも可能です。



サンプルプログラム抜粋

```
Private Sub IDB_START_Click()

    ' AD 変換時間、サンプリング時間、AD 変換チャンネル数、AD 変換個数の取得
    ' 変換パラメータのセット
    AdSetParam hwnd, Adc.MyCardType, INTHISPEED_MODE, Adc.MyIOBase, Adc.MyIrqNo

    ' 変換チャンネル数設定
    AdSetChannel Adc.AdcChannel, Sequence(1)
    ' 外部トリガで開始、指定個数で終了
    AdSetTrigger POST_TRIGGER, PULSE_RISING
    ' サンプリング間隔設定
    AdSetFreq hwnd, Adc.SampTime, Adc.TimeUnitNo
    ' VB アプリケーション側でデータを保管するための配列確保.必ず FIFO サイズ余分に確保
    Adc.MaxSampDataNum = Adc.AdcChannel * Adc.dwSampCount
    ReDim Adc.pData(Adc.MaxSampDataNum + 32) As Integer

    ' AD データ格納メモリーをアロケート(1 チャンネル当たりの変換個数を指定する)
    DllMem = AdAllocDataMem(hwnd, Adc.dwSampCount)
    If DllMem = 0 Then
        Exit Sub
    End If
    fConversionStop = False
    'OLE のウィンドウハンドル取得
    OleHandle = MBOX1.GetMboxWnd()
    ' 割り込みモード AD 変換の実行
    AdStartIrqVxdMode OleHandle, 0
End Sub
```

```
Private Sub IDB_STOP_Click()
    ' 繰り返し変換実行時は、ストップボタンが押されたとき、ドライバ内部で変換を
    ' 行っている場合は AdStopIrqMode()で割り込みリソースを解放しないようにする。
    ' 本サンプルでは、ストップボタンが押されたことを示すフラグを用意し、
    ' MBOX1_OnMsgPost()で次の変換を開始しないようにします。

    fConversionStop = True
End Sub
```

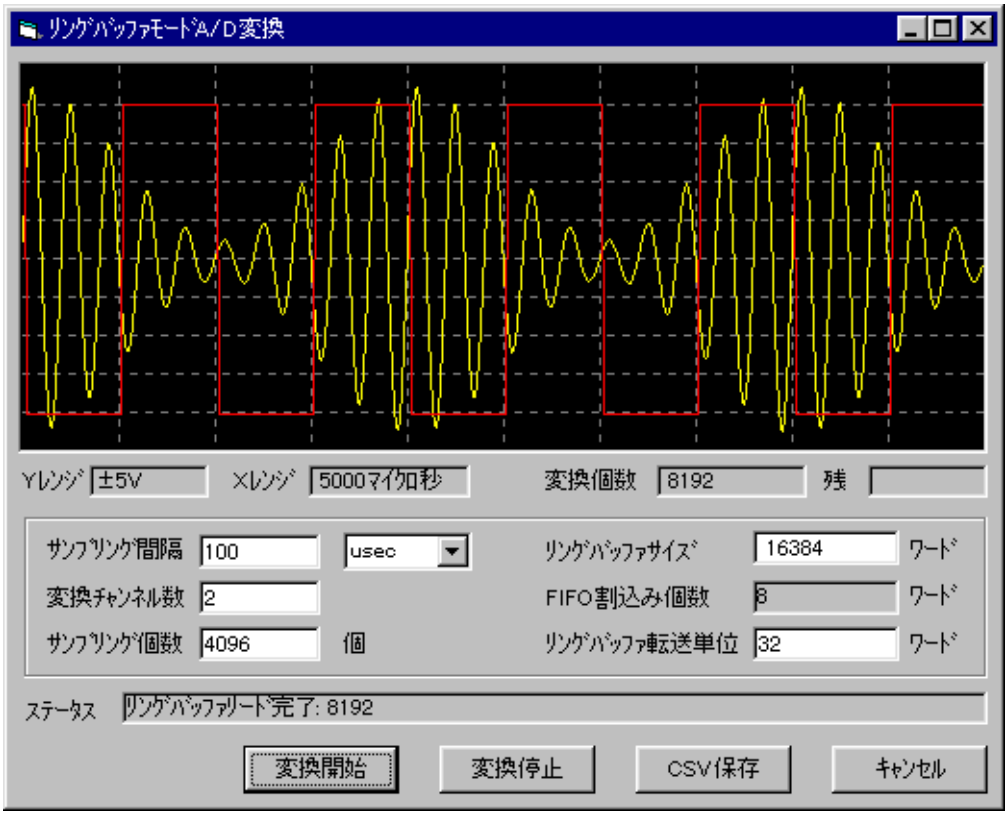
```
Private Sub MBOX1_OnMsgPost(ByVal wParam As Integer, ByVal lParam As Long)

    AdCount = lParam                    ' lParam -> AD 変換データカウント数
    AdcStat = wParam                    ' wParam -> AD 変換開始終了コード

    Select Case AdcStat
    Case AD_MODE_RUN                    ' 変換開始メッセージ

    Case AD_MODE_TRGRUN                 ' トリガー変換開始メッセージ

    Case AD_MODE_PRESTOP                ' トリガー終了メッセージ
      VbMemCopy Adc.pData(0), DllMem, AdCount * 2
      ' グラフ描画処理
      AdStopIrqMode                     ' 変換を停止し割り込みリソース解放
      AdFreeDataMem                     ' AD データ格納メモリー開放

    Case AD_MODE_STOP                   ' 指定個数終了メッセージ
      ' ドライバがセットした変換データを自分のメモリへ転送保管します
      ' 1 個の変換データは 2 バイトですので転送先のアドレスに注意して下さい
      VbMemCopy Adc.pData(0), DllMem, AdCount * 2
      ' グラフ描画処理
      ' 変換停止ボタンが押されていないかチェック
      If fConversionStop <> True Then
         ' 繰り返し次の割り込みモード AD 変換の実行
         AdStartIrqVxdMode OleHandle, 0
      Else
         AdStopIrqMode                  ' 変換を停止し割り込みリソース解放
         AdFreeDataMem                  ' AD データ格納メモリー開放
      End If

   Case AD_MODE_OVRUN                   ' FIFO オーバーラン停止メッセージ
      ' ｸﾞﾗﾌ表示
      AdStopIrqMode                     ' 変換を停止し割り込みリソース解放
   End Select
End Sub
```

RingMode → リングバッファを使った割り込みモード AD 変換プログラム

リングバッファを使った割り込みモード AD 変換 **AdStartIrqRingMode()**をスタートすると A/D カードは FIFO に指定個数の変換データがセットされると割り込み要求を行います。ドライバーの割り込み処理では FIFO からデータを **AdAllocRingBuf()**で設定されたリングバッファへ転送します。

VB アプリケーション側ではデータを保管するための配列を確保します。A/D 変換スピードに応じたリングバッファからの変換データ取り出し間隔と個数の目安値を設定してタイマー処理で **VbGetRingBufConst()**を呼び出してドライバがセットした変換データを自分のメモリへ転送保管します。タイマー処理ではグラフ表示も行います。

ドライバーがリングバッファへ変換データを転送するレートに対し、アプリケーションがリングバッファからデータを取り出すスピードが遅れるとある時点でリングバッファがオーバーフローします。すなわち、ドライバーはリングバッファの最も古いデータに上書きしますのでそのデータは失われます。



■サンプルプログラム抜粋

```
Private Sub IDB_START_Click()

    ' AD 変換時間、サンプリング時間、AD 変換チャンネル数、AD 変換個数の取得
    ' 割込みを発生する FIFO データ個数とリングバッファから転送するデータ個数設定
    InitRingParam    Adc.SampTime,    Adc.TimeUnitNo,    Adc.AdcChannel,    Adc.IrqUpNum,
Adc.RingGetDataCount

    ' カードリソース情報の自動設定
    AdSetParamAuto hwnd, INTHISPEED_MODE
    ' 変換チャンネル数設定
    AdSetChannel Adc.AdcChannel, Sequence(1)
    ' サンプリング間隔設定
    AdSetFreq hwnd, Adc.SampTime, Adc.TimeUnitNo
    ' RingBufSize ワードのリングバッファ作成
    AdAllocRingBuf hwnd, RingBufSize, Adc.dwSampCount

    'OLE のウィンドウハンドル取得
    OleHandle = MBOX1.GetMboxWnd()
    ' リングバッファを使った割込みによる AD 変換開始
    RetCode = AdStartIrqRingMode(OleHandle, Adc.IrqUpNum)
    If RetCode <> 0 Then
        AdFreeRingBuf
        Exit Sub
    End If

    ' VB アプリケーション側でデータを保管するためのメモリーを確保
    Adc.MaxSampDataNum = Adc.dwSampCount * Adc.AdcChannel
    ReDim Adc.pData(Adc.MaxSampDataNum) As Integer

    BreakFlag = False
    ' リングバッファデータ取得インターバルタイマー起動
    RingTimer.Interval = 10
End Sub
```

```
Private Sub IDB_STOP_Click()
    BreakFlag = True
    ' 変換を強制的に止める
    AdStopIrqMode
    ' ﾘﾝｸﾞﾊﾞｯﾌｧを解放
    AdFreeRingBuf

    RingTimer.Interval = 0
End Sub
```

```
' AD 変換開始終了のメッセージ、変換個数の通知
Private Sub MBOX1_OnMsgPost(ByVal wParam As Integer, ByVal lParam As Long)

    ' lParam -> AD 変換完了データ個数(チャンネル×変換個数+α)カウント数
    NumberOfAdcData = lParam
    AdcStat = wParam                ' wParam -> AD 変換終了ﾓｰﾄﾞ

    Select Case wParam
        Case AD_MODE_RUN           ' 変換開始メッセージ
        Case AD_MODE_TRGRUN        ' トリガー変換開始メッセージ
        Case AD_MODE_STOP          ' 指定個数終了メッセージ
        Case AD_MODE_PRESTOP       ' トリガー変換終了メッセージ
        Case AD_MODE_OVRUN         ' FIFO オーバーラン停止メッセージ
    End Select
End Sub
```

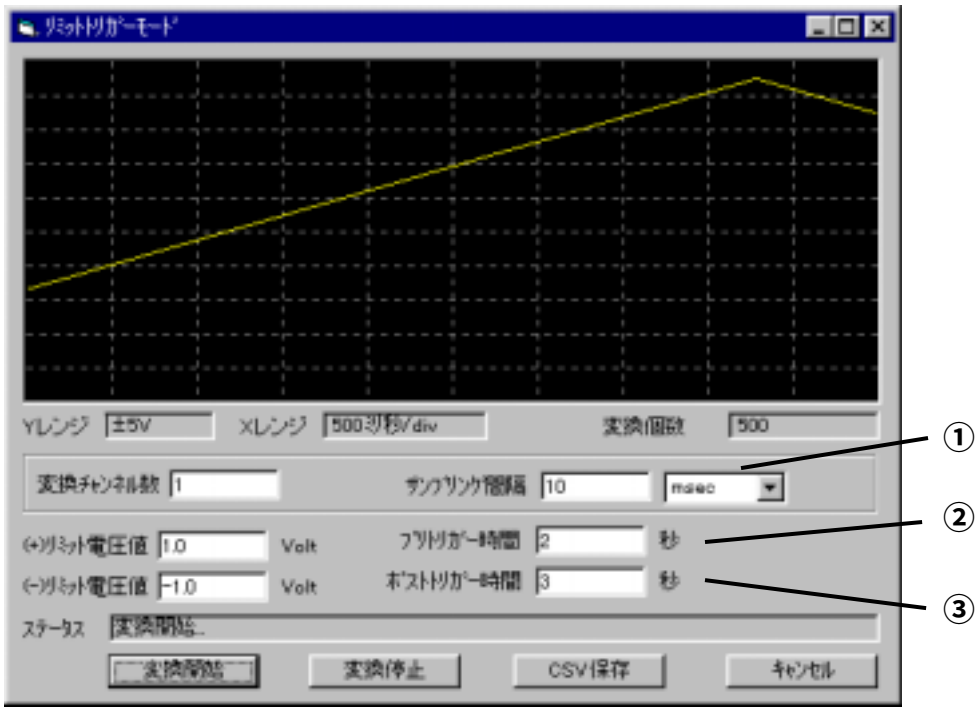
```
Private Sub RingTimer_Timer()

    ' 変換停止・終了を判別。BreakFlag = True の時（停止・終了時）はタイマー処理を終了。
    If BreakFlag = True Then
        RingTimer.Interval = 0
        Exit Sub
    End If

    ' リングバッファにある残りのデータ個数を求める
    Remain = Adc.MaxSampDataNum - Adc.TotalNum
    ' リングバッファからデータを取得し自分のデータバッファへ転送する
    If Remain >= Adc.RingGetDataCount Then
        GetCount = VbGetRingBufConst(Adc.pData(0), Adc.TotalNum, Adc.RingGetDataCount)
    Else
        GetCount = VbGetRingBufConst(Adc.pData(0), Adc.TotalNum, Remain)
    End If


    If GetCount = -1 Then
        ' リングバッファがオーバーフローしたので変換を停止します
        RingTimer.Interval = 0  'タイマー停止
        AdStopIrqMode
        AdStartFlag = False
    ElseIf GetCount > 0 Then
        ' リングバッファから転送したデータを表示
        ' グラフ描画処理
    End If

    ' 指定個数に達したら     ' リングバッファリードを停止
    If Adc.TotalNum >= Adc.MaxSampDataNum Then
        RingTimer.Interval = 0
        BreakFlag = True
    End If
End Sub
```

LimitTrg → リミットトリガモード AD 変換プログラム

[変換開始]ボタンが押されると AdSetLimitTrg()が呼び出され、(+)(-)リミット電圧値で指定したリミットトリガ（入力電圧が設定上限値もしくは下限値を上回った点）をセットします。次に AdAllocTrgBuf()を呼び出してリミットトリガ発生の前後で取得する変換データ個数とリミットトリガ用バッファを確保します。取得する変換データ個数はプリトリガー時間、ポストトリガー時間の指定が関係してきます。最後に AdStartLimitTrgMode()を呼び出してリミットトリガモードの A/D 変換を実行します。

A/D 変換が開始するとユーザ定義メッセージがユーザプログラムにポストされます。この時、wParam には AD_MODE_RUN がセットされています。リミットトリガ発生後指定時間の変換が終了すると、AD_MODE_STOP がセットされたユーザ定義メッセージがポストされます。この時、lParam には変換データ取り出し先頭アドレスがセットされています。ここで、AdGetTrgData()を呼び出し、上記アドレスを指定して変換データの取り出しを行います。

本サンプルプログラムでは、AdGetTrgData()による変換データの取出しが完了すると AdRestartLimitTrgMode()を呼び出して繰り返し計測を実行します。



上記サンプルプログラムでは、+2.0V でリミットトリガを検出したときに検出前 2 秒間と検出後 3 秒間のデータを取得、グラフ描画したものです。

1 チャンネル当たりの変換個数 ＝ （ ②+③ ）／ ①

リミットトリガ検出前の変換データについては②で設定した時間分のデータが存在しない場合、存在する個数しか取得しませんのでご注意ください。

また、③で設定した時間内で再度リミットトリガが発生しても、データ取得中はリミットトリガの検出は行いません。

サンプルプログラム抜粋

```
Private Sub IDB_START_Click()

    ' AD 変換時間、サンプリング時間、AD 変換チャンネル数の取得
    ' プリトリガー時間、ポストトリガー時間、(+)側トリガー電圧、(-)側トリガー電圧の取得

    AdSetParam hwnd, Adc.MyCardType, INTHISPEED_MODE, Adc.MyIOBase, Adc.MyIrqNo
    ' 変換チャンネル数設定
    AdSetChannel Adc.AdcChannel, Sequence(1)
    ' サンプリング間隔設定
    AdSetFreq hwnd, Adc.SampTime, Adc.TimeUnitNo

    ' リミットトリガーモードの設定
    AdSetLimitTrg Adc.MaxLimit, Adc.MinLimit, UPPER_LIMIT
    '取得する変換データ個数とバッファを確保
    Adc.TrgBufWordSize  =  AdAllocTrgBuf(Adc.SampTime,  Adc.TimeUnitNo,  Adc.AdcChannel,
Adc.PreTime, Adc.PostTime)
    If Adc.TrgBufWordSize <= 0 Then
        Exit Sub
    End If
    ReDim Adc.pData(Adc.TrgBufWordSize * 2)

    'OLE のウィンドウハンドル取得
    OleHandle = MBOX1.GetMboxWnd()
    ' リミットトリガーモード AD 変換の実行
    AdStartLimitTrgMode OleHandle

End Sub
```

```
Private Sub IDB_STOP_Click()
    ' 割り込み登録解除
    AdStopLimitTrgMode
    ' リングバッファを解放
    AdFreeTrgBuf
    IDS_TOTALNUM.Caption = ""
    IDC_STATUS.Caption = "変換終了"

End Sub
```

```
Private Sub MBOX1_OnMsgPost(ByVal wParam As Integer, ByVal lParam As Long)

    pStartMem = lParam          ' lParam -> リングバッファ取り出し先頭アドレス
    AdcStat = wParam               ' wParam -> AD 変換終了モード

    Select Case AdcStat
        Case AD_MODE_RUN           ' 変換開始メッセージ
        Case AD_MODE_STOP          ' 指定個数終了メッセージ
            ' データ取り出し
            Adc.dwDataNum = AdGetTrgData(pStartMem, Adc.pData(0))
            If Adc.dwDataNum <> 0 Then
                IDS_TOTALNUM.Caption = Str(Adc.dwDataNum)
                ' グラフ表示処理
                ' リングバッファリセットとサンプリング再開
                RetCode = AdRestartLimitTrgMode(OleHandle)
                If RetCode <> 0 Then
                    IDC_STATUS.Caption = "AD 変換再開ｴﾗｰ(ｺｰﾄﾞ:" + Str(RetCode) + ")"
                End If
            End If
        Case AD_MODE_OVRUN      ' FIFO オーバーラン停止メッセージ
    End Select

End Sub
```

空白ページ

第3章 Windows2000/XP解説

3-1. インストレーション

この章では Windows2000 および WindowsXP での REX-5054U/B セットアップについて解説致しております。

3-1-1.インストール方法

Windows2000 でのインストール方法

※以下のインストレーション画面は、REX-5054U を使用しておりますので REX-5054B をご使用のお客様は、「REX-5054U」の部分を「REX-5054B」にお読み替え下さい。

【1】 PC カードの挿入

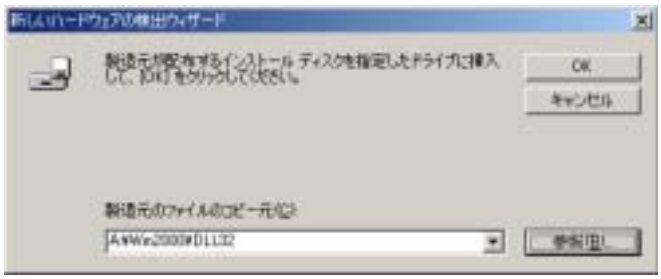
PC カードを挿入すると、右の「新しいハードウェアの検出ウィザード」が起動します。
ここでは、「次へ(N)>」ボタンを押して次に進んでください

検索方法の指定では、「デバイスに最適なドライバを検索する (推奨)(S)」にチェックを入れて「次へ(N)>」ボタンを押します。

[2] ドライバファイル場所の指定

「ドライバファイルの特定」で「場所を指定(S)」にチェックを入れて「次へ(N)>」ボタンを押します。

製品添付の Windows2000/XP 用フロッピーディスクをＦＤドライブに挿入します。製造元のファイルのコピー元(C)で inf ファイルの場所を指定し、「ＯＫ」ボタンを押します。



「ドライバファイルの検索」が始まりドライバを検索します。検索完了後、「次へ(N)>」ボタンを押します。

「新しいハードウェアの検出ウィザードの完了」で「REX-5054.SYS for REX-5054 and REX5054B」が表示されます。
「完了」ボタンを押してください。

以上で REX-5054 ドライバのインストールは終了です。

WindowsXP でのインストール方法

※以下のインストレーション画面は、REX-5054B を使用しておりますので REX-5054U をご使用のお客様は、「REX-5054B」の部分を「REX-5054U」にお読み替え下さい。

【1】 PC カードの挿入

PC カードをスロットに挿入すると、右図の「新しいハードウェアの検出ウィザード」が表示されますので、製品添付の Win2000/XP 用フロッピーディスクを FD ドライブへ挿入してください。

次に、「ソフトウェアを自動的にインストールする (推奨) (I)」を選択し「次へ」ボタンを押します。

セットアップ情報ファイル (inf ファイル) が、フロッピーディスク上から検索され、自動的にインストールが行われます。

右の画面が表示されましたら、「完了」ボタンを押します。

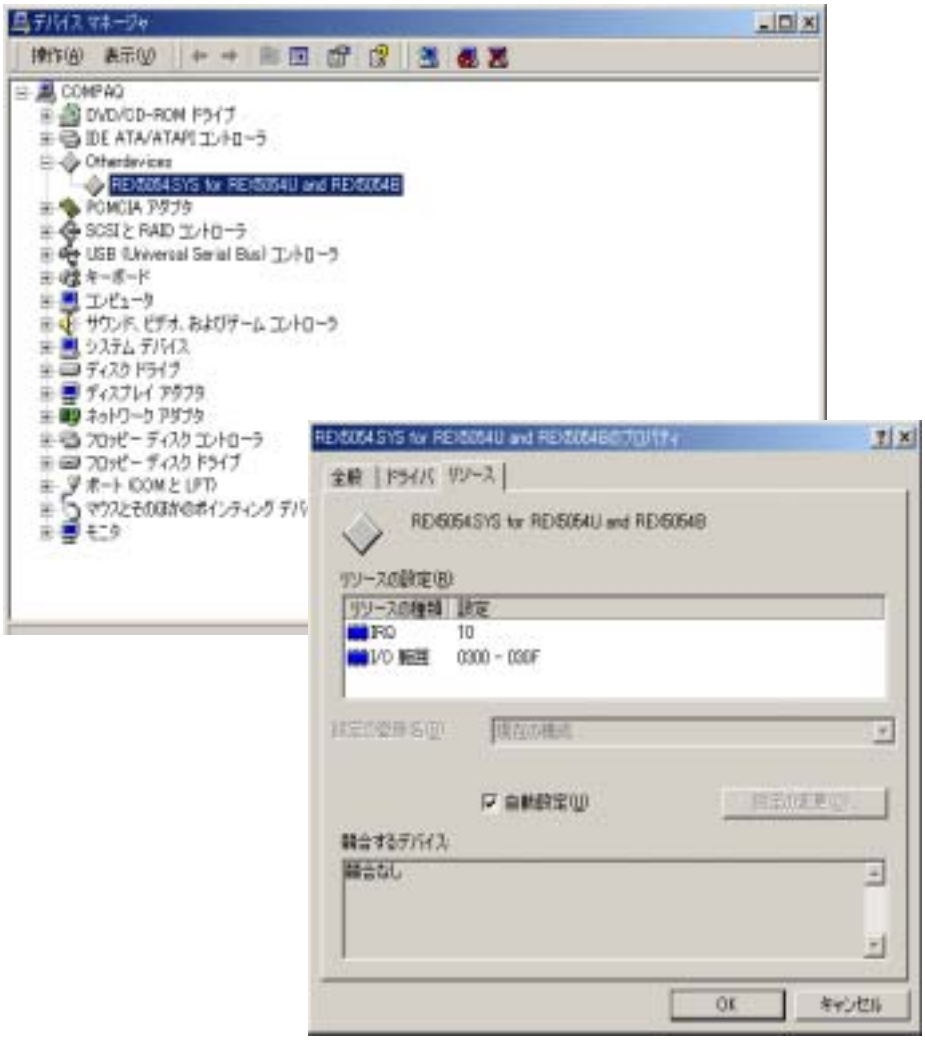
「完了」ボタンを押した後、右下の画面が出ますので、「はい (Y)」を押してパソコンを再起動してください。

以上で、REX-5054 インストールは終了です。

3-1-2. PC カード設定内容の確認

コントロールパネルのシステムを起動します。「システムのプロパティ」のハードウェアのタブから「デバイスマネージャ(D)」ボタンを押します。「OtherDevices」をクリックして新しく REX5054.SYS for REX5054U and REX5054B が追加されているのを確認してください。

また、「プロパティ」でリソースが正しく割当てられているかを確認してください。デバイスの競合が発生した場合は「自動設定(U)」のチェックを外し、競合が起こらない値に設定を変更してください。



3-1-3. アンインストール方法

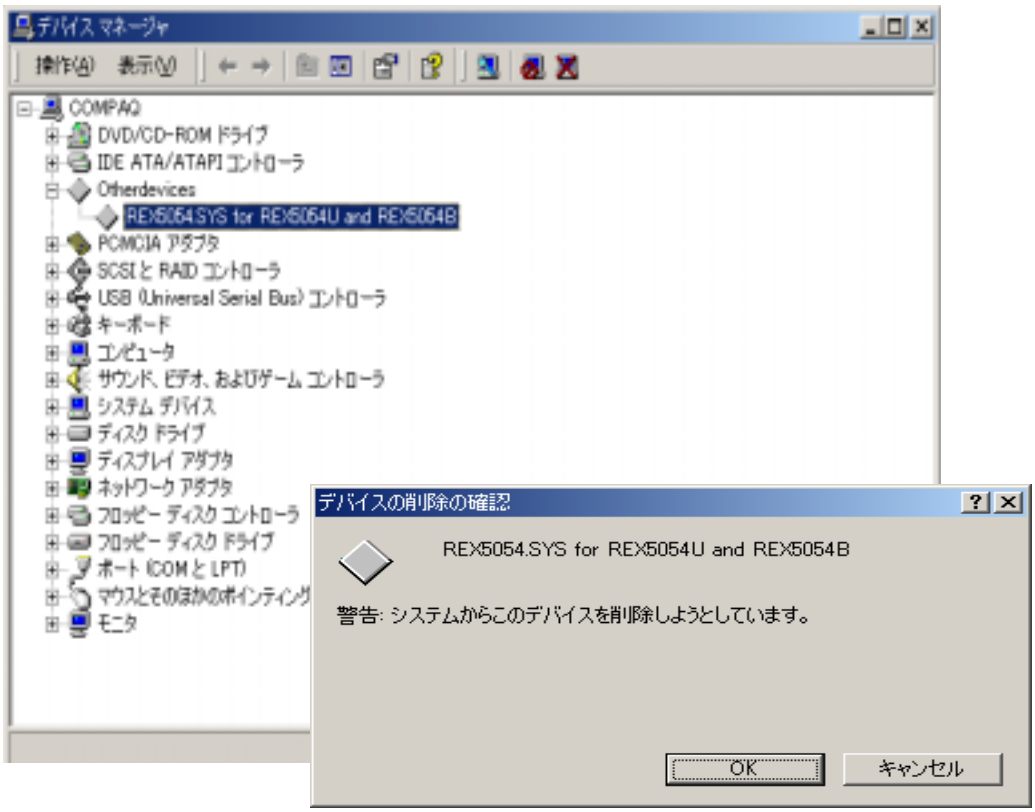
Windows2000 および WindowsXP でのアンインストール方法

インストールした内容を削除する方法について説明します。
削除は、
(1)デバイスの削除
(2)INF ファイルの削除
の手順で行います。

[1] デバイスの削除

PC カードを挿入した状態で、コントロールパネルのシステムを起動します。「システムのプロパティ」のハードウェアのタグから「デバイスマネージャ(D)」ボタンを押します。「Otherdevices」をクリックして REX-5054.SYS for REX5054U and REX5054B を表示させクリックします。

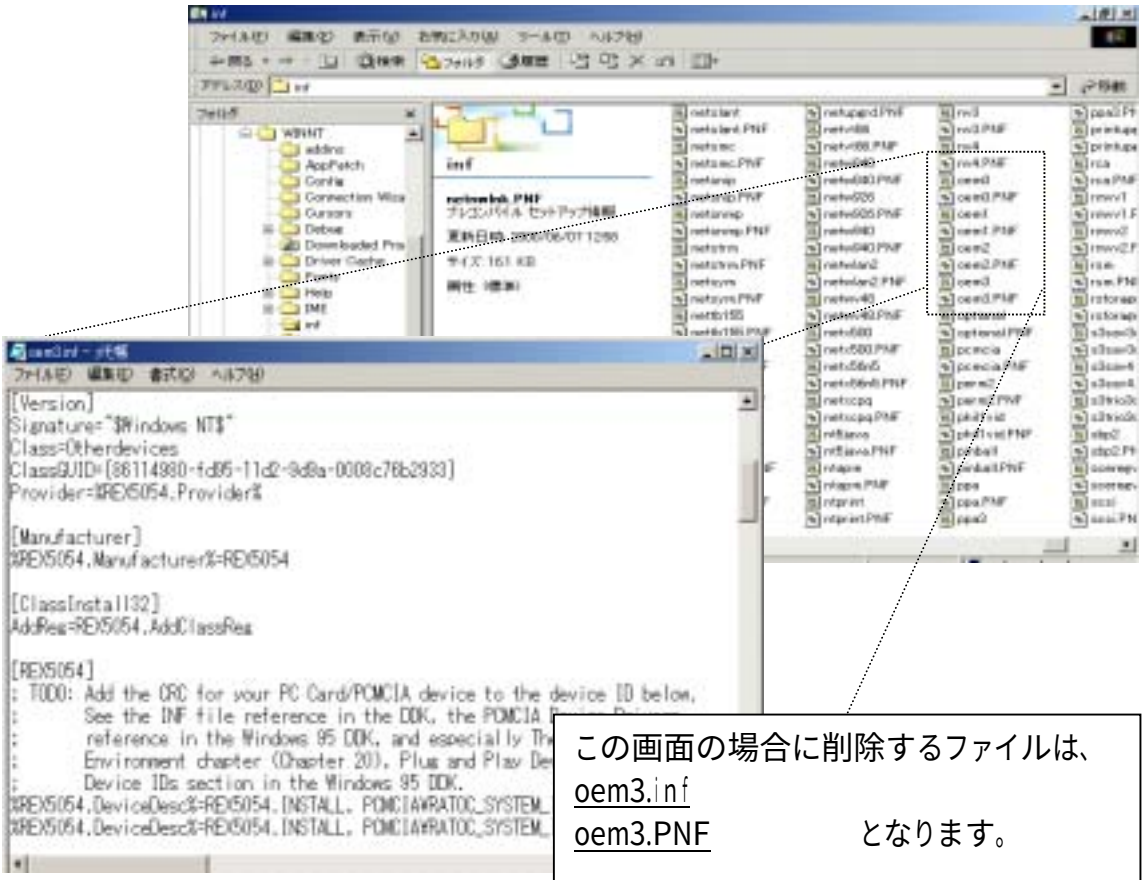
メニューバーより「操作(A)」ー「削除(U)」を選択します。デバイスの削除の確認で「ＯＫ」ボタンを押し削除してください。



【2】INF ファイルの削除

エクスプローラから Windows フォルダ内の「inf フォルダ」を開き、oemX.inf ファイル(X=数字)を検索し、例えば oem0.inf が１つだけの場合は、oem0.inf と拡張子のみ異なる oem0.PNF を削除してください。oemX.inf が複数ある場合（ oem0.inf, oem1.inf・・・）は、メモ帳などでそれぞれの inf ファイルを開いて、その内容の[Manufacturer]セクションが %REX5054, Manufacturer%=REX5054 となっているファイルと拡張子のみ異なる PNF ファイルを削除してください。

エクスプローラの設定が「全てのファイルを表示」になっていないとフォルダ「C:\WINDOWS\INF」は表示されません。設定の変更は、エクスプローラメニューの「ツール(T)」から「フォルダオプション(O)」を選択して変更します。



この画面の場合に削除するファイルは、
oem3.inf
oem3.PNF　　となります。

以上の操作でアンインストール完了です。カードスロットより、REX-5054 を抜きパソコンを再起動してください。

再度、インストールされる場合はパソコンを再起動後、Page.3-1 のインストレーションをご参照ください。

3-2. Visual C 言語インターフェイス

3-2-1. DLL ライブラリ API 解説

本製品には Windows2000/XP での 32 ビットアプリケーション開発に必要となる API インターフェースを提供する DLL ライブラリ“ADLIB2K.DLL”が添付されています。ユーザ側で作成したアプリケーションは“ADLIB2K.DLL”の API を呼び出します。

次ページでは、“ADLIB2K.DLL”が提供する関数について解説致します。

アプリケーション作成上のアドバイス

○ PC カードに割り当てられてるリソースを取得する

Windows2000/XP では、Windows95/98/Me と同様に **AdGetCardResource()**がサポートされています。アプリケーションの初期化部分等で AdGetCardResource()を呼び出し I/O アドレス・IRQ 番号を取得して下さい。

○ Windows95/98/Me で作成したアプリケーションを Windows2000/XP で使用する場合、Windows2000/XP 用ヘッダファイル **ADLIB2K.H** とライブラリファイル **ADLIB2K.LIB** を新規プロジェクトに追加し、作成したソースファイルにインクルード後、コンパイルすることによって使用可能になります。

但し、以下の関数を使用されている場合は、ADLIB2K.DLL ではサポートしておりませんのでご注意ください。

AdCheckStopTrig()
AdGetCardNameInfo()
AdGetParam()
AdGetRingBufAdrs()
AdGetVersion()
AdPlot()
AdSaveCsvFile()
AdSaveFile()
AdSetAndStart()
AdstartIrqPostMsgMode()
AdStartIrq()
AdStartPol()
AdStartPolMode()
HexToVal()※1
HexToValU/B
OutPort()
wOutPort()
InPort()
wInPort()

※ Windows95/98/Me のサンプルプログラム(ONESHOT.C)に関数仕様を載せておりますのでご参照ください。

→ *DLL 関数仕様*

AdAllocDataMem　AD 変換データ格納用メモリのアロケーション

書 式　LPVOID **AdAllocDataMem**(HWND **hwnd**, DWORD **SampNumPerChan**)

機 能　指定データ個数の AD 変換データ格納用メモリブロックのアロケーションを要求します。メモリブロックは(指定データ個数+32)ワードサイズでドライバー内部にロックされて確保されます。
確保要求を行ったプログラムは、終了時必ず **AdFreeDataMem**()を呼び出しメモリブロックを解放して下さい。

引 数　HWND **hwnd** ➢ 呼び出し元ウィンドウのハンドル
DWORD **SampNumPerChan** ➢ 1 チャンネル当たりのサンプリングデータ個数

戻 値　メモリブロックへの VOID 型ポインタを返します。先にサンプリングチャンネルパラメータ **AdSetChannel** の設定が必要です。チャンネル設定が行われていない場合、または要求された大きさのメモリを確保できなかった場合は 0 を返します。

AdAllocMem　汎用メモリのアロケーション

書 式　LPVOID **AdAllocMem**(HWND **hwnd**, DWORD **ByteToAllocate**)

機 能　指定バイト数のロックされたメモリブロックをドライバー内部にアロケーションします。確保要求を行ったプログラムは終了時に必ず **AdFreeMem**()を呼び出して確保したメモリブロックを解放して下さい。

引 数　HWND **hwnd** ➢ 呼び出し元ウィンドウのハンドル
DWORD **ByteToAllocate** ➢ アロケーションサイズ(バイト単位)

戻 値　アロケーションしたメモリブロックへの VOID 型ポインタを返します。0 の場合はアロケーションできなかったことを示します。

AdAllocRingBuf　リングバッファのアロケーション

書 式　BOOL **AdAllocRingBuf**(HWND **hwnd**, DWORD **wAllocSize**, DWORD **SampNumPerChan**)

機 能　指定ワードサイズの AD 変換データ格納用リングバッファメモリの確保し、リングバッファ制御変数を初期化します。AD 変換はトータルの変換個数が（**SampNumPerChan**×指定チャンネル数）に達すると終了します。
確保要求元のプログラムは、終了時必ず **AdFreeRingBuf**()を呼び出して確保したリングバッファを解放して下さい。

引 数
- HWND **hwnd** ➢ 呼び出し元ウィンドウのハンドル
- DWORD **wAllocSize** ➢ ワード単位のアロケーションサイズ
- DWORD **SampNumPerChan** ➢ 1 チャンネル当たりの変換データ個数

戻 値
- 0 : リングバッファ初期化正常終了
- -2 : リングバッファアロケーションエラー
- -3 : 二重設定エラー
- -4 : アロケーションサイズ及び変換データ個数取得エラー
- -10 : ドライバ呼び出しエラー

AdAllocTrgBuf　　入力電圧リミットトリガーバッファのアロケーション

書 式　long **AdAllocTrgBuf**(WORD **SampTime**, WORD **TimeUnit**, WORD **AdChan**, WORD **PreTime**, WORD **PostTime**)

機 能　リミットトリガを検出した時点を基準としてその前の PreTime 秒間の変換データとその後の PostTime 秒間の変換データを格納するためのバッファを確保します。呼び出し側プログラムは終了時 AdFreeTrgBuf()により確保されたバッファを解放して下さい。AdSetLimitTrg()で設定されたリミットトリガに対し、前後で取得する変換データ個数は下記 PreTime および PostTime より下式により計算されます（TimeUnit がミリ秒の場合）。

リミットトリガ以前の変換データ格納バッファワードサイズ
= (PreTime * 1000 / SampTime) * AdChan;

リミットトリガ以降の変換データ格納バッファワードサイズ
= (PostTime * 1000 / SampTime) * AdChan;

本バッファはリングバッファモードで変換データが格納されます。

引 数

- WORD **SampTime** ➢ A/D 変換サンプリング周期...AdSetFreq()での設定値を指定
  注）1msec より速くは変換不可。
- WORD **TimeUnit** ➢ サンプリング周期の単位を下記より指定
  sec(0):秒　msec(1):ミリ秒
- WORD **AdChan** ➢ 変換チャンネル総数指定
  ...AdSetChannel()での設定値を指定
- WORD **PreTime** ➢ リミットトリガ発生前何秒間のデータを保持するか秒単で指定
- WORD **PostTime** ➢ リミットトリガ発生後何秒間のデータを保持するか秒単で指定

戻 値　正常に変換データ格納用バッファが確保されると、確保されたバッファのワードサイズが返されます。

-2 ： 変換データ格納バッファアロケーションエラー
-3 ： リミットトリガーモード変換時間設定エラー
-10 ： ドライバ呼び出しエラー

AdFreeDataMem　AD 変換データ格納用メモリを解放

書 式　VOID **AdFreeDataMem**(VOID)

機 能　**AdAllocDataMem**()で確保した変換データ格納用を解放します。

引 数　なし

戻 値　なし

AdFreeMem　汎用メモリの解放

書 式　VOID **AdFreeMem**(LPVOID **pHeapMem**)

機 能　**AdAllocMem**()で確保したメモリブロックを解放します。

引 数　LPVOID **pHeapMem**　➢ **AdAllocMem**()で取得したメモリポインタ

戻 値　なし

AdFreeRingBuf　リングバッファの解放

書 式　VOID **AdFreeRingBuf**(VOID)

機 能　**AdAllocRingBuf**()で確保したリングバッファおよびリングバッファ制御パラメータを解放します。

引 数　なし

戻 値　なし

AdFreeTrgBuf　入力電圧リミットトリガバッファの解放

書 式　VOID **AdFreeTrgBuf**(VOID)

機 能　**AdAllocTrgBuf()**で確保されたリミットトリガ用バッファを解放します。

引 数　なし

戻 値　なし

AdGetCardResource　REX5054U/B に割り当てられているリソースの取得

書 式　BOOL **AdGetCardResource**(HWND **hwnd**, WORD **SlotNo**, LPWORD **pCardType**, LPWORD **pIOBase**, LPWORD **pIrqNo**)

機 能　指定スロットに挿入されているカードが自分のカードなら、現在割り当てられている I/O ベースアドレスおよび IRQ 番号情報を返します。

引 数

HWND	**hwnd**	➢ 呼び出し元ウィンドウのハンドル
WORD	**SlotNo**	➢ PC カード挿入スロット番号
LPWORD	**pCardType**	➢ 出力）指定スロットで検出された AD カードの型式
LPWORD	**pIOBase**	➢ 出力）I/O ベースアドレス情報を格納する変数へのポインタ
LPWORD	**pIrqNo**	➢ 出力）IRQ 番号情報を格納する変数ポインタ

戻 値
- 0 : 正常終了
- -1 : カードリソース取得エラー
- -10 : ドライバ呼び出しエラー

AdGetRingBuf　リングバッファから変換データを転送

書 式　int **AdGetRingBuf**(LPVOID **pUserBuf**, int **Offset**, int **GetDataNum**)

機 能　リングバッファから指定個数の変換データをリードし指定のアドレスへ転送します。データの転送先アドレスは **pUserBuf** から **Offset** で示される変換データ個分オフセットした場所になります。
本関数は第三引数で指定した個数分の変換データがリングバッファにない場合は格納されている個数分を転送します。

引 数

LPVOID	**pUserBuf**	➢ データ格納先のアドレス
int	**Offset**	➢ データ格納先のアドレスのオフセット
int	**GetDataNum**	➢ 取得するデータ数

戻 値　転送したの変換データ個数を返します。 リングバッファが空の時は 0 を、リングバッファオーバーフロー発生時 -1 を返します。また、ドライバ呼び出しエラーは-10 を返します。

AdGetRingBufConst　　リングバッファから一定個数の変換データを転送

書 式　int **AdGetRingBufConst**(LPVOID **pUserBuf**, int **Offset**, int **GetDataNum**)

機 能　**GetDataNum** で指定した個数分の変換データがリングバッファに格納された場合にリングバッファから指定個数の変換データをリードし指定のアドレスへ転送します。データの転送先アドレスは **pUserBuf** から **Offset** で示される変換データ個分オフセットした場所になります。
本関数は **AdGetRingBuf()**とは異なり、第三引数で指定した個数分の変換データがリングバッファにない場合は転送を行わず、戻り値として０を返します。

引 数
- LPVOID **pUserBuf** ➢ データ格納先のアドレス
- int **Offset** ➢ データ格納先のアドレスのオフセット
- int **GetDataNum** ➢ 取得するデータ数

戻 値　転送した変換データ個数を返します。
第三引数で指定した個数分の変換データがリングバッファにない場合は 0 を返します。リングバッファオーバーフロー発生時には－１を返します。また、ドライバ呼び出しエラーは-10 を返します。

AdGetTrgData　　入力電圧リミットトリガバッファから変換データを転送

書 式　DWORD **AdGetTrgData**(PWORD **pStartMem**, PWORD **pCopyMem**)

機 能　**AdAllocTrgBuf()**で確保されたリミットトリガー用バッファから変換データを取り出します。
コピー先バッファの先頭には、リミットトリガ発生前何秒間のデータを保持するか指定した時点の変換データから順にセットされます。有効時間が経過する前にリミットポイントが検出された場合は、変換を開始した時点の変換データから順にセットされます。この場合は、コピーされるデータ個数は AdAllocTrgBuf()で返されたサイズより少なくなりますので注意願います。

引 数
- PWORD **pStartMem** ➢ リミットトリガーモード変換データ取り出し先頭アドレス
  通常、AD_MODE_STOP 受信時の追加情報 2 で渡されたアドレスをそのまま指定して下さい。
- PWORD **pCopyMem** ➢ コピー先バッファアドレス
  コピー先バッファのサイズは AdAllocTrgBuf()で返されたワードサイズ分確保しておいて下さい。

戻 値　コピーされたデータの個数が返されます。

AdOneShot　ワンショットモード AD 変換実行

書 式　BOOL **AdOneShot**(HWND **hwnd**, LPWORD **pMem**)

機 能　ワンショットモードによる AD 変換を行います。変換を実行する前に、**AdSetOneParam**()でパラメータ設定を完了しておいて下さい。

引 数　HWND **hwnd** ➢ 呼び出し元ウィンドウのハンドル
LPWORD **pMem** ➢ 変換データ格納先メモリへのポインタ

戻 値　0 : 正常終了
-1 : ドライバ呼び出しエラー
-2 : ドライバシグナル状態応答エラー
-4 : 変換開始エラー
-5 : データ転送エラー

AdSetChannel　変換チャンネルパラメータの設定

書 式　**int AdSetChannel**(WORD **Channels**, LPWORD **pSequence**)

機 能　カード内部の A/D コンバージョンチップ（LM12458）に変換実行チャンネルの設定を行います。チャンネルシーケンスに設定されたチャンネル番号は昇順に並び換えます。同じチャンネルが複数指定されている場合は重複チャンネルを削除します。本関数を呼び出す場合は、必ず先に AdSetParam() 又は AdSetParamAuto()によりカード情報の設定を行ってください。

引 数　WORD **Channels** ➢ AD 変換チャンネル数(1 オリジン)
LPWORD **pSequence** ➢ チャンネルシーケンス（0 オリジン）

戻 値　正常終了時、設定したチャンネル数を返します。チャンネル設定が違う時は-1 を返します。ドライバ呼び出しエラーは-10 を返します。

AdSetDataCount　サンプリング個数の設定

書 式　DWORD **AdSetDataCount**(DWORD **SampNumPerChan**)

機 能　割り込みモードで割り込みに同期したユーザ定義メッセージを受け取りながら AD 変換を行うとき、本関数により変換を終了するデータ個数の設定を行います。また、ドライバー内部で必要となる 1 チャンネル当たりの変換個数および変換を停止するトータルの変換データ個数は、通常 **AdAllocDataMem()**を呼び出すことにより内部的に設定されますが、AdAllocDataMem()を使わないでプログラム側で変換データ格納用のメモリを確保する場合は本関数を使って 1 チャンネル当たりの変換個数を設定します。 AdAllocDataMem()を呼び出した場合は、本関数によるサンプリング個数の設定は省略できます。本関数を呼び出す場合は、必ず先に **AdSetChannel()**により変換チャンネルの設定を行って下さい。

引 数　DWORD **SampNumPerChan** ➢ 1 チャンネル当たりのサンプリング個数

戻 値　0 : 正常終了
-1 : チャンネル未設定エラー
-10 : ドライバ呼び出しエラー

AdSetExternalClock　外部クロック入力の設定

書 式　BOOL **AdSetExternalClock**(BOOL **Mode**, WORD **Counter0**, WORD **Counter1**)

機 能　AD カードの内部クロックを無効にし、外部クロックを使って変換を行います。外部クロックを A/D カード内部のプログラマブルタイマーカウンタで分周して A/D 変換のクロックとするモードと、そのまま A/D 変換のクロックとするモードがあります。分周を行う場合は、カウンタ#0・カウンタ#1 に分周値を設定して下さい。
分周を行う場合は、Counter0 または Counter1 に分周値を設定します。分周値が小さい場合は、Counter0 に値を設定し Counter1 には 0 を渡します。Counter0 には 2～65535 の範囲の値を指定できます。分周値が大きい場合は、Counter0/Counter1 の順で値を設定します。分周値は（Counter0×Counter1）になります。

引 数　BOOL **Mode**　➢ 外部クロック入力モード
EXTERNAL_DIVIDE : 内部で分周を行う
EXTERNAL_DIRECT : 分周を行わない
WORD **Counter0**　➢ 分周回路 0 に設定する値（2～65535 の範囲）
WORD **Counter1**　➢ 分周回路 1 に設定する値（0～65535 の範囲）

戻 値　0 : 正常終了
-10 : ドライバ呼び出しエラー

AdSetFreq　サンプリング間隔の設定

書 式　BOOL **AdSetFreq**(HWND **hwnd**, WORD **Stime**, WORD **Unit**)

機 能　サンプリング間隔と単位を設定します。

引 数　HWND **hwnd**　➢ 呼び出し元ウィンドウのハンドル
WORD **Stime**　➢ サンプリング間隔の設定値
WORD **Unit**　➢ サンプリング間隔単位の設定値
sec：0　msec：1　usec：2
入力電圧リミットトリガモードの場合は 1msec より遅く設定してください。

戻 値　0 : 正常終了
-1 : チャンネル数指定エラー
-2 : サンプリング間隔単位設定エラー
-3 : サンプリング間隔設定エラー
-10 : ドライバ呼び出しエラー

AdSetInternalClock　内部クロックの選択とサンプリング間隔の手動設定

書 式　BOOL **AdSetInternalClock**(BOOL **UseClock**, WORD **Counter0**, WORD **Counter1**)

機 能　内部クロックの選択とサンプリング間隔の設定を行います。A/D カードは 10MHz と 8.192MHz の二つのクロックを内蔵しており、通常は 10MHz の内部クロックを使用しています。サンプリング間隔は、選択したクロックから分周値を Counter0 または Counter1 に設定することにより行います。分周値を大きくする場合は、Counter0・Counter1 の順で設定して下さい。分周値は（Counter0×Counter1）の値になります。

引 数　BOOL **UseClock**　➢ 内部原発クロックの選択
INTERNAL_10MHZ : 内部クロック 10MHz 選択
INTERNAL_8MHZ : 内部クロック 8.192MHz 選択
WORD **Counter0**　➢ 分周回路 0 に設定する値
WORD **Counter1**　➢ 分周回路 1 に設定する値

戻 値　0 : 正常終了
-10 : ドライバ呼び出しエラー

AdSetLimitTrg 入力電圧リミットトリガ値の設定

書 式 ini **AdSetLimitTrg**(double **MaxLimit**, double **MinLimit**, WORD **TrgMode**)

機 能 リミットトリガモード A/D 変換のリミットトリガをかける電圧値を設定します。入力電圧が下記設定上限値もしくは下限値を上回った点でトリガがかかります。トリガが検出された前後の変換データを取得します。リミットトリガ発生の前後で取得する変換データ個数は、AdAllocTrgBuf()で設定します。

引 数

double **MaxLimit**	➢ トリガをかける入力電圧の上限値を電圧値で指定	
double **MinLimit**	➢ トリガをかける入力電圧の下限値を電圧値で指定	
WORD **TrgMode**	➢ 上限値・下限値のいづれを有効にするか下記から選択指定	
	UPPER_LIMIT	:入力電圧が MaxLimit を上回った時、トリガをかける。
	LOWER_LIMIT	:入力電圧が MaxLimit を下回った時、トリガをかける。
	UPPER_LOWER_LIMIT	:上記の両方でトリガをかける。

戻 値
- 0 : 正常終了
- -1 : A/D カード型式未設定
- -2 : 上限下限値設定範囲エラー
- -3 : リミットトリガーモード未設定エラー
- -10 : ドライバ呼び出しエラー

AdSetOneParam ワンショット AD 変換パラメータ設定

書 式 BOOL **AdSetOneParam**(HWND **hwnd**, WORD **wCardType**, WORD **wUseIOAddr**)

機 能 ワンショットモード AD 変換実行時のパラメータを設定します。AdOneShot()を実行する前に本関数によりワンショットパラメータの設定を行って下さい。

引 数

HWND **hwnd**	➢ 呼び出し元ウィンドウのハンドル
WORD **wCardType**	➢ カード型式（1：REX5054U，2：REX5054B）
WORD **wUseIOAddr**	➢ カードに割り当てられている I/O ベースアドレス

戻 値
- 0 : 正常終了
- -1 : カードタイプ設定エラー
- -2 : カードステータスレジスタリードエラー
- -3 : A/D 変換チャンネル数エラー
- -4 : サンプリング間隔エラー
- -10 : ドライバ呼び出しエラー

AdSetParam　カード情報の手動設定

書 式　BOOL **AdSetParam**(HWND **hwnd**, WORD **CardType**, BOOL **SampMode**, WORD **BaseAddr**, WORD **IRQNo**)

機 能　AD 変換実行に必要なパラメータを手動設定します。本関数を使用する場合は、予め AD カードが使用する I/O ベースアドレス及び割り込み番号を調べておく必要がありますが、AdSetParamAuto()を使えばそれらの情報は関数内部でシステムに問い合わせて自動で取得しますので AdSetParamAuto()の方を使用して下さい。

引 数

HWND **hwnd**	➢ 呼び出し元ウィンドウのハンドル		
WORD **CardType**	➢ A/D カード型式（1：REX5054U，2：REX5054B）		
BOOL **SampMode**	➢ サンプリング実行モード		
	ポーリングモード	POLLING_MODE	：0
	ワンショットモード	ONESHOT_MODE	：1
	ポストメッセージ版割込みモード	INTPOSTMSG_MODE	：2
	リングバッファ割込みモード	INTHISPEED_MODE	：3
WORD **BaseAddr**	➢ カードが使用する I/O ベースアドレス		
WORD **IRQNo**	➢ カードが使用する割り込み番号		

戻 値

0	：正常終了
-1	：サンプリング実行モード未設定エラーまたはカードタイプ設定エラー
-2	：カードステータスレジスタリードエラー
-10	：ドライバ呼び出しエラー

AdSetParamAuto　カード情報の自動設定

書 式　BOOL **AdSetParamAuto**(HWND **hwnd**, WORD **SampMode**)

機 能　AD カードの型式及びカードが使用する I/O アドレス・割り込み番号をシステムから取得してパラメータの自動設定を行います。

引 数　HWND **hwnd** ➢ 呼び出し元ウィンドウのハンドル
BOOL **SampMode** ➢ サンプリング実行モード

ポーリングモード	POLLING_MODE	：0
ワンショットモード	ONESHOT_MODE	：1
ポストメッセージ版割込みモード	INTPOSTMSG_MODE	：2
リングバッファ割込みモード	INTHISPEED_MODE	：3

戻 値　0 ： 正常終了
-1 ： カードタイプ指定エラー
-2 ： カードステータスレジスタリードエラー
-3 ： サンプリングモード未設定エラー
-10 ： ドライバ呼び出しエラー

AdSetTrigger　外部トリガーによる変換開始･終了の設定

書 式　int **AdSetTrigger**(WORD **TriggerMode**, WORD **TimingMode**)

機 能　外部からのトリガー信号を使って AD 変換の開始及び終了を行うためのパラメータを設定します。

引 数　WORD **TriggerMode** ➢ 外部トリガーモードの指定

TRIGGER_DISABLE(0)	:外部トリガーモードを無効にする
POST_TRIGGER(1)	:外部トリガーで変換開始
PRE_TRIGGE(2)	:外部トリガーで変換終了
POST_PRE_TRIGGER(3)	:外部トリガーで変換を開始・終了

WORD **TimingMode** ➢ 立上がり・立下がりトリガーモードの指定

PULSE_FALLING(0)	:信号の立下りでトリガー
PULSE_RISING(1)	:信号の立上りでトリガー

戻 値　0 ： 正常終了
-10 ： その他のエラー

AdStartIrqRingMode　リングバッファ割込み AD 変換実行

書 式　BOOL **AdStartIrqRingMode**(HWND **hwnd**, WORD **IrqSetCount**)

機 能　リングバッファを使った割り込みによる AD 変換入力を開始します。ドライバーは割り込み要求があると FIFO から変換データを取り出して指定のリングバッファに格納します。
リングバッファに格納された変換データは AdGetRingBuf()を呼び出していつでも取得することができます。
呼び出し元プログラムは、終了時は必ず AdStopIrqMode()を呼び出して変換動作をクリアして下さい。
変換の開始及び変換データ個数が指定個数に達するとドライバーから呼び出し元プログラムにメッセージがポストされます。この時、wParam に AD 変換開始終了メッセージコードが、lParam に終了時の変換データ数がセットされています。

[メッセージコード] **AD_MODE_RUN** : 通常モードでの変換スタート
AD_MODE_TRGRUN : 開始トリガーにより変換スタート
AD_MODE_STOP : 指定個数の変換完了によるストップ
AD_MODE_PRESTOP : 終了トリガーによる変換ストップ
AD_MODE_OVRUN : オーバーランによる変換ストップ

引 数　HWND **hwnd** ➢ ユーザ定義メッセージをポストするウィンドウのハンドル
WORD **IrqSetCount** ➢ 割り込み要求を行う FIFO 変換データ個数を指定
0 が指定されている場合は、ドライバーが最適な個数を自動設定します。手動設定する場合は 2～10 の範囲で指定して下さい。FIFO の長さは 32 ワードですが、ドライバーは一度の割り込み処理でに 10 個以上は取り出しません(オーバーラン対策)。

戻 値　0 : 正常終了
-1 : バッファ確保エラー
-2 : FIFO データ個数設定エラー
-3 : 変換開始エラー

AdStartIrqVxdMode	高速割込みモード AD 変換実行

書 式 BOOL **AdStartIrqVxdMode**(HWND **hwnd**, WORD **IrqSetCount**)

機 能 Windowsシステム側処理のオーバーヘッド伴わず高速に割り込みモードA/D変換の実行します。ドライバーは指定個数の A/D 変換が終了するまで呼び出し元プログラムに一切の通知を行いません。変換の開始及び変換データ個数が指定個数に達するとドライバーから呼び出し元プログラムに変換終了メッセージがポストされます。
この時、**wParam** に AD 変換開始終了メッセージコードが、**lParam** に終了時の変換データ数がセットされています。

[メッセージコード] **AD_MODE_RUN** : 通常モードでの変換スタート
AD_MODE_TRGRUN : 開始トリガーにより変換スタート
AD_MODE_STOP : 指定個数の変換完了によるストップ
AD_MODE_PRESTOP : 終了トリガー-による変換ストップ
AD_MODE_OVRUN : オーバーランによる変換ストップ

引 数
- HWND **hwnd** ➢ ユーザ定義メッセージをポストするウィンドウのハンドル
- WORD **IrqSetCount** ➢ 割り込み要求を行う FIFO 変換データ個数を指定
  0 が指定されている場合は、ドライバーが最適な個数を自動設定します。手動設定する場合は 2～10 の範囲で指定して下さい。FIFO の長さは 32 ワードですが、ドライバーは一度の割り込み処理でに 10 個以上は取り出しません(オーバーラン対策)。

戻 値
- 0 : 正常終了
- -1 : バッファ確保エラー
- -2 : (IrqSetCount=1)個数セットエラー
- -3 : 変換開始エラー

AdStartLimitTrgMode　リミットトリガモード AD 変換実行

書 式　BOOL **AdStartLimitTrgMode**(HWND **hwnd**)

機 能　リミットトリガモードの A/D 変換を実行します。
繰り返し実行する場合は、AdRestartLimitTrgMode()を呼び出します。
呼び出し元プログラムは、終了時は必ず AdStopLimitTrgMode()を呼び出して変換動作をクリアして下さい。
リミットトリガモードの A/D 変換が開始するとユーザ定義メッセージ WM_VXDEVENT メッセージがユーザープログラムにポストされます。この時追加情報 1(wParam)には AD_MODE_RUN がセットされています。リミットポイント発生後指定時間の変換が終了すると、追加情報 1 に AD_MODE_STOP がセットされたユーザ定義メッセージがポストされます。この時、追加情報 2(lParam)には変換データを取り出す変換データ取り出し先頭アドレスがセットされています。
変換データを取り出す場合は、AdGetTrgData()に上記取り出し先頭アドレスを指定して取り出しを行って下さい。
FIFO オーバーランによる変換が終了した場合は、追加情報 1 に AD_MODE_OVRUN がセットされたユーザ定義メッセージがポストされます。

引 数　HWND **hwnd**　➢ ユーザ定義メッセージを受取るウィンドウハンドル

戻 値　0 : 正常終了
-1 : バッファ確保エラー
-2 : 変換開始エラー

AdRestartLimitTrgMode　リミットトリガモード AD 変換繰返し実行

書 式　BOOL **AdRestartLimitTrgMode**(HWND **hwnd**)

機 能　AdStartLimitTrgMode()を繰り返し実行します。

引 数　HWND **hwnd**　➢ 呼び出し元ウィンドウのハンドル

戻 値　0 : 正常終了
-1 : バッファ確保エラー
-2 : トリガー開始エラー

AdStartPolModeThread　　ポーリングモード AD 変換チャイルドスレッド

書 式　BOOL **AdStartPolModeThread**(HWND **hwnd**)

機 能　CPU を独占しないでポーリングによる AD 変換を行う場合に、ユーザ側アプリケーションから呼び出すポーリングモード AD 変換を行うチャイルドスレッドです。本チャイルドスレッドは入力変換の開始・終了時と呼び出し元プログラムにメッセージをポストします。
この時、wParam に AD 変換開始終了メッセージコードが、lParam に終了時の変換データ数がセットされています。

[メッセージコード] AD_MODE_RUN　：通常モードでの変換スタート
AD_MODE_TRGRUN　：開始トリガーにより変換スタート
AD_MODE_STOP　：指定個数の変換完了によるストップ
AD_MODE_PRESTOP　：終了トリガー-による変換ストップ
AD_MODE_OVRUN　：オーバーランによる変換ストップ

引 数　HWND **hwnd**　➢ 呼び出し元ウィンドウのハンドル

戻 値　0 ： 正常終了
-1 ： バッファ確保エラー

AdStopIrqMode 割り込み AD 変換停止とリソースを解除

書 式 VOID **AdStopIrqMode**(VOID)

機 能 割り込みモードによる変換を停止しシステムに登録した割り込みリソースを解除します。
AdStartIrqRingMode(),AdStartIrqVxdMode(),AdStartIrqPostMsgMode()を呼び出したプログラムは終了時は必ず割り込み登録をクリアして終了して下さい。

引 数 なし

戻 値 なし

AdStopLimitTrgMode リミットトリガモード AD 変換停止とリソース解除

書 式 VOID **AdStopLimitTrgMode**(VOID)

機 能 リミットトリガモードの A/D 変換を終了しリソースを解放します。

引 数 なし

戻 値 なし

GetRingBufNum リングバッファから変換データ個数を転送

書 式 DWORD **GetRingBufNum**(VOID)

機 能 リングバッファから変換データ個数を転送します。

引 数 なし

戻 値 リングバッファから変換データ個数を返します。リングバッファからの転送に失敗した場合は、0 を返します。

3-2-2. Visual C サンプルプログラム解説

REX-5054U/B AD PC カードを制御するアプリケーションを開発する場合は、本製品添付のソースプログラム（*.c）を参考にして下さい。

添付のサンプルプログラムを使用してプログラム作成・修正する場合は、各サンプルのフォルダ内にあるファイル（*.C、*.RC、RESOURCE.H、REX.ICO）と DLL32 フォルダ内のファイル（ADLIB2K.H、ADLIB2K.LIB）を新規プロジェクトに追加してコンパイルを行ってください。

AD 変換を行うモードには大きく分けて次の二つがあります。

→ポーリングモード
プログラムから AD カードの FIFO に変換データがあるかどうか調べ、変換データがあればデータを取り出すモード
注）本モードは Windows システムパフォーマンス上の問題より、できる限り使用しないで割り込みモードを使用して下さい。

→割り込みモード
FIFO に設定した個数の変換データがセットされた時点で AD カードから割り込み要求を行い割り込み処理で FIFO からデータを取り出すモード

本製品には下記モードのサンプルプログラムが添付されています。

ポーリングモード	
OneShot	ワンショットモード AD 変換プログラム
PolMode	ポーリングモード AD 変換を行うプログラム
割り込みモード	
IntBase	割り込みモード AD 変換プログラム
RingMode	リングバッファを使った割り込みモード AD 変換プログラム
LimitTrg	リミットトリガーモード AD 変換プログラム

OneShot → ワンショットモード AD 変換プログラム

ワンショットモードでは、AdOneShot()を呼び出しポーリングモードで 1 回だけサンプリングを行い電圧値をダイアログ画面に表示します。繰り返し計測は、指定の繰返し周期でデータを取得するためにタイマー処理で AdOneShot()を呼び出し、指定回数分だけ変換を繰返します。

サンプルプログラム抜粋

```
BOOL Dlg_OnInitDialog (HWND hwnd, HWND hwndFocus, LPARAM lParam)
{
   /* スロットに挿入されている自分のカードのリソース情報を取得する */
   if ( AdGetCardResource( hwnd, 0, &MyCardType, &MyIOBase, &MyIrqNo ) != 0 )
   {
       // エラー処理
       return FALSE;
   }
   /* ワンショット A/D 変換のパラメータ設定 */
   if ( (Status = AdSetOneParam( hwnd, MyCardType, MyIOBase )) != 0 )
   {
       // エラー処理
       return FALSE;
   }
   return TRUE;
}

void Cmd_OnCmdStart ( HWND hwnd )
{
   /* マルチメディアタイマー起動 */
   StartMMTimer( hwnd, uPeriod );
}

int StartMMTimer( HWND hwnd, UINT uPeriod )
{
   /* 指定の周期でデータを取得する為にマルチメディア用ワンショットタイマーを使用する */
   /* この後、指定時間経過後 GetDataProc()がコールバックされる */
   timeSetEvent( uPeriod, uResolution, GetDataProc, (DWORD)hwnd, TIME_PERIODIC );
}
```

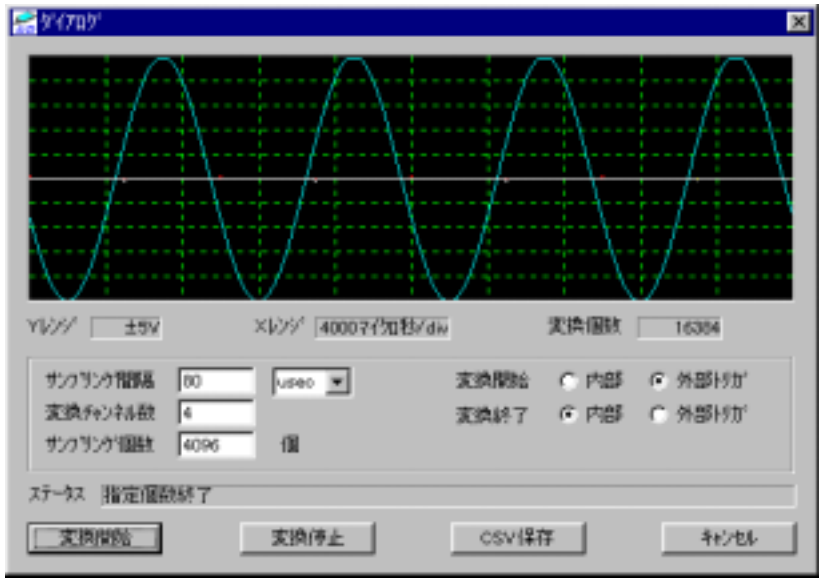
＜次のページにつづく＞

```
void CALLBACK GetDataProc( UINT nTimerId, UINT msg, DWORD dwUser, DWORD dwParam1, DWORD
dwParam2 )
{
   if ( MyAdOneShot ( hwnd ) != 0 )
   {
      StopMMTimer();
      return;
   }
}

double HexToVolt( USHORT Val )
{
   double RetVal;

   switch ( MyCardType )
   {
   case REX5054U:
      RetVal = (double)((short)( ( Val & 0x1fff ) << 3 ) ) * 2.5 / 32768.0;
      return RetVal;
   case REX5054B:
      RetVal = (double)((short)( ( Val & 0x1fff) << 3 ) ) * 10.0 / 32768.0 - 5.0;
      return RetVal;
   }
   return 0.0;
}

int MyAdOneShot ( HWND hwnd )
{
   /* A/D 変換実行（常に全ﾁｬﾝﾈﾙの変換を行います） */
   if ( (Status = AdOneShot( hwnd, AdBuf )) != 0 )
   {  // エラー処理
      return -1;
   }
   /* 電圧値を表示 */
   switch( MyCardType )
   {
   case REX5054U:
      for ( Chloop = 0; Chloop < 8; Chloop++ )
      {
         AdVal = HexToVolt( AdBuf[Chloop] );
         sprintf( szBuf, "%+1.3f", AdVal );
         SetDlgItemText( hwnd, IDS_VOLTCH1+Chloop, szBuf );
      }
      break;
   case REX5054B:
      for ( Chloop = 0; Chloop < 4; Chloop++ )
      {
         // 上記と同処理
      }
      break;
   }
   return 0;
}
```

PolMode → ポーリングモード AD 変換を行うプログラム

DLL でイクスポートしているポーリングモードの A/D 変換ルーチンである**AdStartPolModeThread()**を呼び出すことにより割り込みを禁止して A/D 変換が開始します。AdStartPolModeThread()は変換の開始と終了時ユーザ定義メッセージを呼び出し元ウィンドウへポストします。呼び出されたアプリケーションはこの時の wParam・lParam からステータスと変換個数情報を受け取ります。A/D 変換データは、**AdAllocDataMem()**でアロケーションしたメモリに格納されます。AdAllocDataMem()はアロケーションしたメモリへのポインター(LPVOID lpMem)を返しますので、ユーザ側のアプリケーションから変換データにアクセスすることが可能です。

変換開始・終了はボタン操作の他に外部トリガを使用することも可能です。



サンプルプログラム抜粋

```
BOOL Dlg_OnInitDialog (HWND hwnd, HWND hwndFocus, LPARAM lParam)
{
  /* スロットに挿入されている自分のカードのリソース情報を取得する */
  if ( AdGetCardResource( hwnd, 0, &Adc.MyCardType, &Adc.MyIOBase, &Adc.MyIrqNo ) != 0 )
  {
   /* ｶｰﾄﾞが挿入されていない場合のｴﾗｰ処理 */
   sprintf( szBuf, "REX-5054U/B ADｶｰﾄﾞ検出ｴﾗｰ" );
   SetDlgItemText( hwnd, IDC_STATUS, szBuf );
   return FALSE;
  }

  return TRUE;
}
```

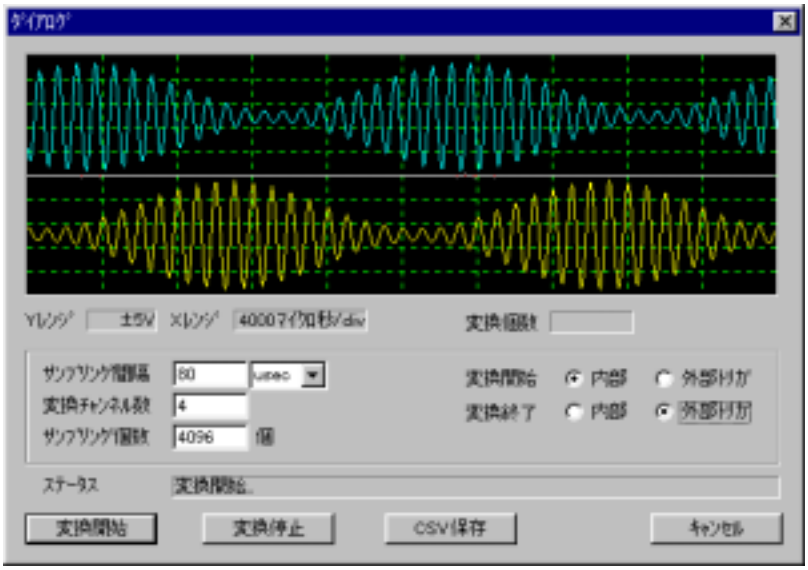
```
void Cmd_OnCmdStart ( HWND hwnd )
{
   /* AD 変換時間 */
   Adc.SampTime = (USHORT)GetDlgItemInt( hwnd, IDE_SAMPTIME, NULL, FALSE );
   /* サンプリング時間 */
   Adc.TimeUnitNo = (USHORT)SendDlgItemMessage( hwnd, IDCB_TIMEUNIT, CB_GETCURSEL, 0, 0L );
   /* AD 変換チャンネル数 */
   Adc.AdcChannel = (USHORT)GetDlgItemInt( hwnd, IDE_CHAN, NULL, FALSE );
   /* AD 変換個数 */
   Adc.dwSampCount = (DWORD)GetDlgItemInt( hwnd, IDE_SAMPCOUNT, NULL, FALSE );
   /* AD 変換パラメータ設定 */
   AdSetParam( hwnd, Adc.MyCardType, POLLING_MODE, Adc.MyIOBase, Adc.MyIrqNo );
   /* 変換チャンネル数設定 */
   AdSetChannel( Adc.AdcChannel, (LPWORD)Sequence );
   /* サンプリング間隔設定 */
   AdSetFreq( hwnd, Adc.SampTime, Adc.TimeUnitNo );
   /* AD データ格納メモリーをアロケート(1 チャンネル当たりの変換個数を指定する) */
   Adc.pData = (LPWORD)AdAllocDataMem( hwnd, Adc.dwSampCount );

   /* ポーリングモード AD 変換を行うスレッドを実行 */
   hThread = CreateThread( NULL, 0, PolChildThread, hwnd, 0, &dwChildId );
}
```

```
DWORD WINAPI PolChildThread( HWND hwnd )
{
   AdStartPolModeThread( hwnd );
   ExitThread( TRUE );
   return 0;       // コンパイルワーニングを除去するためのダミーコード
}
```

```
void Dlg_OnUserDefineMessage (HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
   AdcStat = wParam;         /* wParam -> AD 変換開始終了コード */
   AdCount = lParam;         /* lParam -> AD 変換データカウント数 */
   switch( AdcStat )
   {
   case AD_MODE_RUN:
      SetDlgItemText( hwnd, IDC_STATUS, "変換開始..." );
      break;
   case AD_MODE_TRGRUN:
      SetDlgItemText( hwnd, IDC_STATUS, "トリガー変換開始" );
      break;
   case AD_MODE_STOP:
      SetDlgItemText( hwnd, IDC_STATUS, "指定個数終了" );
      PlotGraph( hwnd );
      break;
   case AD_MODE_PRESTOP:
      SetDlgItemText( hwnd, IDC_STATUS, "トリガー終了" );
      PlotGraph( hwnd );
      break;
   }
}
```

IntBase → 割り込みモード AD 変換プログラム

AdStartIrqVxdMode()を呼び出すことによりシステム側処理のオーバーヘッドを伴わず高速に割り込みモード AD 変換を実行します。このモードではドライバは指定個数の A/D 変換が終了するまで呼び出し元プログラムに一切の通知を行いません。AdStartIrqVxdMode()は変換の開始及び変換データ個数が指定個数に達するとドライバから呼び出し元プログラムにメッセージをポストします。この時、wParam に AD 変換開始終了メッセージコードが、lParam に終了時の変換データ数がセットされています。
変換開始・終了はボタン操作の他に外部トリガを使用することも可能です。
A/D 変換データは、**AdAllocDataMem()**でアロケーションしたメモリに格納されます。AdAllocDataMem()はアロケーションしたメモリへのポインター(LPVOID lpMem)を返しますので、ユーザ側のアプリケーションから変換データにアクセスすることができます。
本サンプルプログラムでは、変換終了後、再び AdStartIrqVxdMode()を呼び出して繰り返し次の割り込みモード A/D 変換を実行しています。



サンプルプログラム抜粋

```
BOOL Dlg_OnInitDialog (HWND hwnd, HWND hwndFocus, LPARAM lParam)
{
 /* スロットに挿入されている自分のカードのリソース情報を取得する */
 if ( AdGetCardResource( hwnd, 0, &Adc.MyCardType, &Adc.MyIOBase, &Adc.MyIrqNo ) != 0 )
 {
  sprintf( szBuf, "REX-5054U/B AD カード検出エラー" );
  SetDlgItemText( hwnd, IDC_STATUS, szBuf );
  return FALSE;
 }

 return TRUE;
}
```

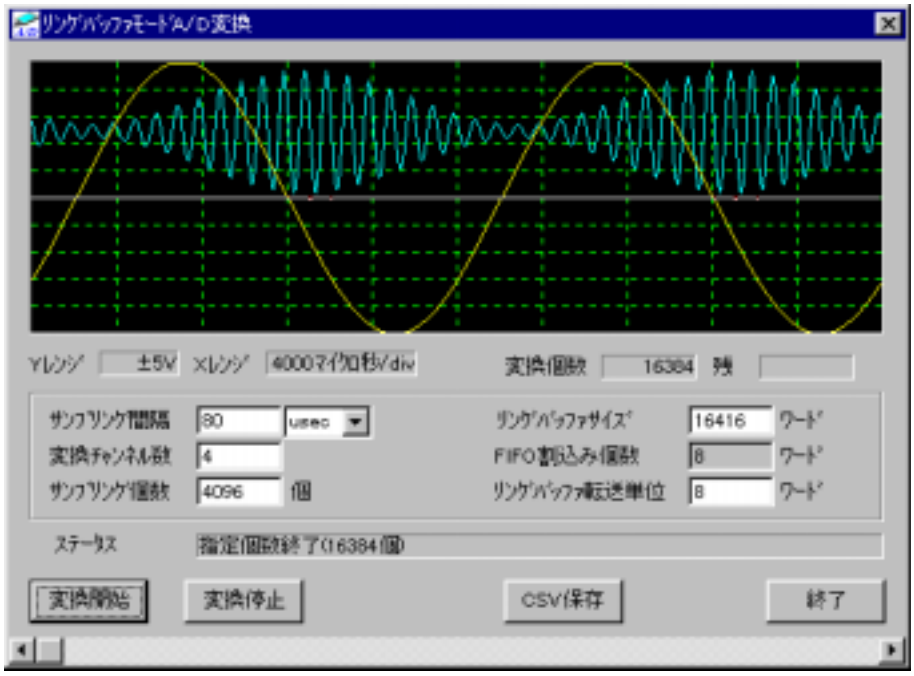
```
void Cmd_OnCmdStart ( HWND hwnd )
{
   /* AD 変換時間取得、サンプリング時間取得、AD 変換チャンネル数取得、AD 変換個数取得 */
   /* 変換パラメータ設定 */
   AdSetParam( hwnd, Adc.MyCardType, INTHISPEED_MODE, Adc.MyIOBase, Adc.MyIrqNo );
   /* 変換チャンネル数設定 */
   AdSetChannel( Adc.AdcChannel, (LPWORD)Sequence );
   /* サンプリング間隔設定 */
   AdSetFreq( hwnd, Adc.SampTime, Adc.TimeUnitNo );
   /* AD データ格納メモリーをアロケート(1 チャンネル当たりの変換個数を指定する) */
   Adc.pData = (LPWORD)AdAllocDataMem( hwnd, Adc.dwSampCount );
   /* 割り込みモード AD 変換の実行 */
   fConversionStop = FALSE;
   AdStartIrqVxdMode( hwnd, 0 );
}
```

```
void Cmd_OnCmdStop ( HWND hwnd )
{
   // 本サンプルでは、ストップボタンが押されたことを示すフラグを用意し、
   // Dlg_OnUserDefineMessage()で次の変換を開始しないようにします。
   fConversionStop = TRUE;
}
```

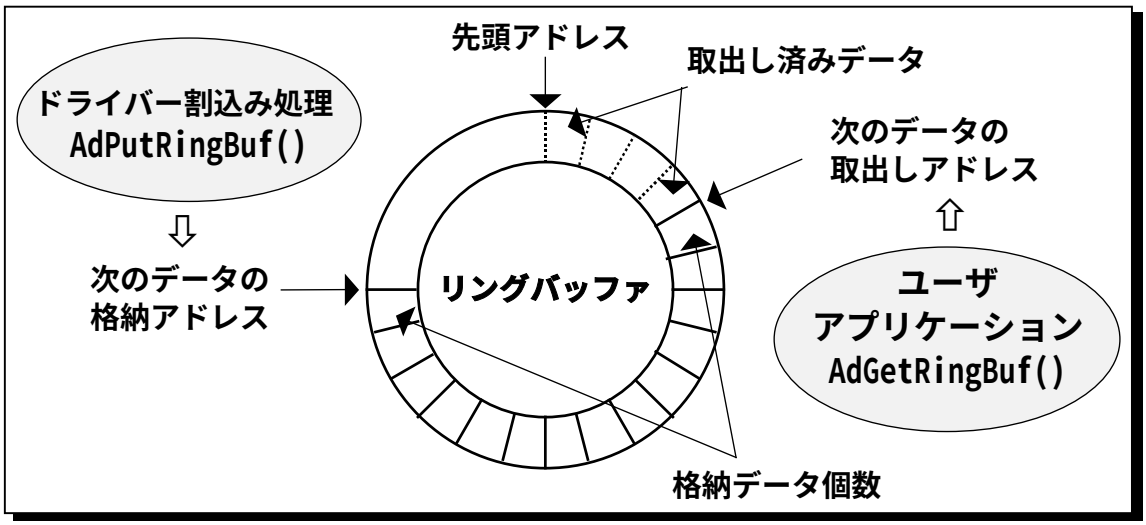
```
void Dlg_OnUserDefineMessage (HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
   AdcStat = wParam;          /* wParam -> AD 変換開始終了コード */
   AdCount = lParam;          /* lParam -> AD 変換データカウント数 */

   switch( AdcStat ){
   case AD_MODE_RUN:          // 変換開始メッセージ表示
      break;
   case AD_MODE_TRGRUN:       // トリガー変換開始メッセージ表示
      break;
   case AD_MODE_PRESTOP:      // トリガー変換終了メッセージ表示
      /* グラフ表示 */
      PlotGraph( hwnd );
      /* 変換を停止し割り込みリソース解放 */
      AdStopIrqMode();
      break;
   case AD_MODE_STOP:         // 変換終了メッセージ表示
      /* グラフ表示 */
      PlotGraph( hwnd );
      /* 変換停止ボタンが押されていないかチェック*/
      if( fConversionStop != TRUE ){
         /* 繰り返し次の割り込みモード AD 変換の実行 */
         AdStartIrqVxdMode( hwnd, 0 );
      }
      else{
         /* 変換を停止し割り込みリソース解放 */
         AdStopIrqMode();
      }
      break;
   }
}
```

RingMode → リングバッファを使った割り込みモード AD 変換プログラム

リングバッファを使った割り込みモード AD 変換 **AdStartIrqRingMode()**をスタートすると A/D カードは FIFO に指定個数の変換データがセットされると割り込み要求を行います。ドライバーの割り込み処理では FIFO からデータを**AdAllocRingBuf()**で設定されたリングバッファへ転送します。本アプリケーションは、リングバッファからアプリケーションの内部バッファに変換データを転送するためのスレッド **GetRingDataThread()**を作成します。
GetRingDataThread()では終了個数に達するまで**AdGetRingBufConst()**を呼び出し、 指定個数単位で変換データを取得してグラフ表示します。ドライバーがリングバッファへ変換データを転送するレートに対し、アプリケーションがリングバッファからデータを取り出すスピードが遅れるとある時点でリングバッファがオーバーフローします。すなわち、ドライバーはリングバッファの最も古いデータに上書きしますのでそのデータは失われます。



☞リングバッファ構造



サンプルプログラム抜粋

```
BOOL Dlg_OnInitDialog (HWND hwnd, HWND hwndFocus, LPARAM lParam)
{
   /* スロットに挿入されている自分のカードのリソース情報を取得する */
   if ( AdGetCardResource( hwnd, SlotNo, &Adc.MyCardType, &MyIOBase, &MyIrqNo ) != 0 )
   {
      /* ｶｰﾄﾞが挿入されていない場合のｴﾗｰ処理 */
      sprintf( szBuf, "REX-5054U/B ADｶｰﾄﾞ検出ｴﾗｰ" );
      SetDlgItemText( hwnd, IDC_STATUS, szBuf );
      /* 開始ボタンを無効にする処理 */
      return FALSE;
   }

   switch( Adc.MyCardType ){
   case REX5054U:
      sprintf( szBuf, "REX-5054U 正常検出 I/O ベース:0x%x IRQ 番号:%d", MyIOBase, MyIrqNo);
      SetDlgItemText( hwnd, IDC_STATUS, szBuf );
      sprintf( szBuf, "0-2.5V" );
      SetDlgItemText( hwnd, IDS_YRANGE, szBuf );
      break;
   case REX5054B:
      sprintf( szBuf, "REX-5054B 正常検出 I/O ベース:0x%x IRQ 番号:%d", MyIOBase, MyIrqNo);
      SetDlgItemText( hwnd, IDC_STATUS, szBuf );
      sprintf( szBuf, "±5V" );
      SetDlgItemText( hwnd, IDS_YRANGE, szBuf );
      break;
   }
   return TRUE;
}
```

```
void Cmd_OnCmdStart ( HWND hwnd )
{
   /* AD 変換時間取得、サンプリング時間取得、AD 変換チャンネル数取得、AD 変換個数取得 */
   /* ワード単位のリングバッファサイズ */
   /* 変換パラメータ設定 */
   /* カードリソース情報の自動設定 */
   AdSetParamAuto( hwnd, INTHISPEED_MODE )
   /* 変換チャンネル数設定 */
   AdSetChannel( Adc.AdcChannel, (LPWORD)Sequence );
   /* サンプリング間隔設定 */
   AdSetFreq( hwnd, Adc.SampTime, Adc.TimeUnitNo );
   /* RingBufSize ワードのリングバッファ作成 */
   AdAllocRingBuf( hwnd, RingBufSize, Adc.dwSampCount )

   /* リングバッファを使った割込みによる AD 変換開始 */
   AdStartIrqRingMode( hwnd, (USHORT)Adc.IrqUpNum )
   /* リングバッファからデータを転送するためのスレッドを実行 */
   /* スレッドではなくタイマーコールバック処理でデータを取れば CPU 負荷は少なくなります */
   hThread = CreateThread ( NULL, 0, GetRingDataThread, hwnd, 0, &dwChildId ) ;
   return;
}
```

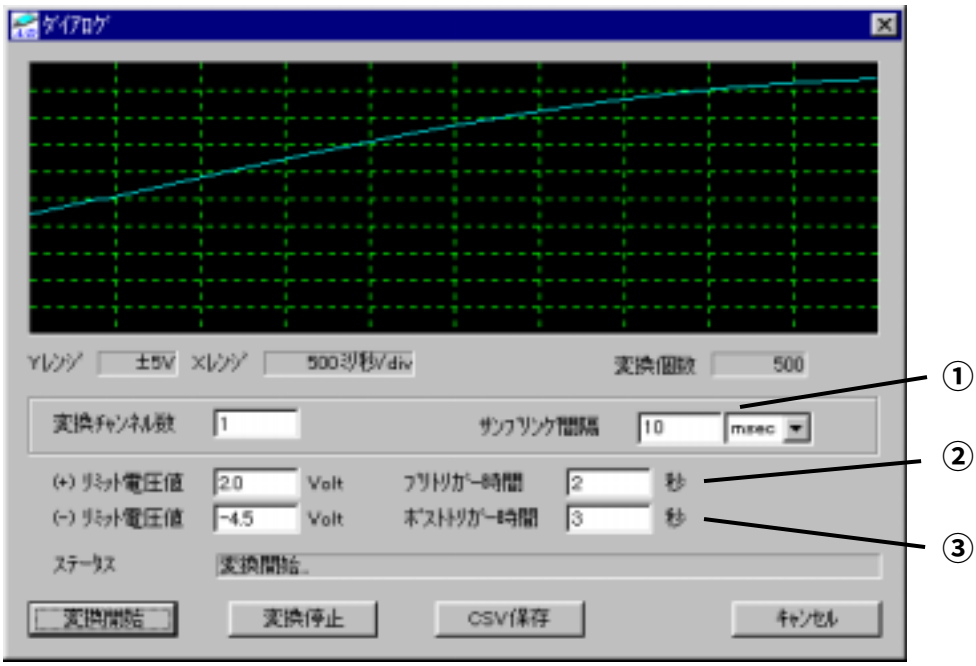
```
DWORD WINAPI GetRingDataThread( HWND hwnd )
{
   while ( BreakFlag == FALSE )
   {
      /* リングバッファから取り出す残りのデータ個数を求める */
      Remain = Adc.MaxSampDataNum - Adc.TotalNum;
      /* リングバッファからデータを取得し自分のデータバッファへ転送する */
      if ( Remain >= (DWORD)Adc.RingGetDataCount )
         /* 残りが RingGetDataCount 以上の場合はアプリで指示された個数単位で転送 */
         AdGetRingBufConst( (LPVOID)Adc.pMyApMem, Adc.TotalNum, Adc.RingGetDataCount );
      else
         /* それ以下の場合は最後の残り個数を転送して終了 */
         AdGetRingBufConst( (LPVOID)Adc.pMyApMem, Adc.TotalNum, Remain );

      /* Polyline()によりグラフ描画処理 */

      /* 指定個数に達したらリングバッファリードを正常終了 */
      if ( (DWORD)Adc.TotalNum >= Adc.MaxSampDataNum )
         break;
   }
   BreakFlag = TRUE;
   return 0;       // コンパイルワーニングを除去するためのダミーコード
}
```

```
void Dlg_OnUserDefineMessage (HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
   /* wParam -> AD 変換終了モード */
   /* lParam -> AD 変換完了データ個数(チャンネル数×変換個数+α)カウント数 */
   NumberOfAdcData = (DWORD)lParam;
   AdcStat = wParam;
   switch ( wParam )
   {
   case AD_MODE_RUN:     sprintf( szBuf, "変換開始", 0 );                          break;
   case AD_MODE_TRGRUN:  sprintf( szBuf, "トリガー変換開始", 0 );                  break;
   case AD_MODE_STOP:    sprintf( szBuf, "指定個数終了(%u 個)", NumberOfAdcData ); break;
   case AD_MODE_PRESTOP: sprintf( szBuf, "トリガー終了(%u 個)", NumberOfAdcData ); break;
   case AD_MODE_RINGOVER:
      sprintf( szBuf, "リングバッファオーバーフロー停止(%u 個)", NumberOfAdcData );  break;
   case AD_MODE_OVRUN:
      sprintf( szBuf, "FIFO オーバーラン停止(%u 個)", NumberOfAdcData );           break;
   }
   SetDlgItemText( hwnd, IDC_STATUS, szBuf );
   return;
}
```

```
void Cmd_OnCmdStop ( HWND hwnd )
{
   BreakFlag = TRUE;              /* スレッドが生きている場合は終了させる */
   AdStopIrqMode();               /* 変換を強制的に止める */
   AdFreeRingBuf();               /* リングバッファを解放 */
   sprintf( szBuf, "変換終了", 0 );
   SetDlgItemText( hwnd, IDC_STATUS, szBuf );
}
```

LimitTrg → リミットトリガモード AD 変換プログラム

[変換開始]ボタンが押されると AdSetLimitTrg()が呼び出され、(+)(-)リミット電圧値で指定したリミットトリガ（入力電圧が設定上限値もしくは下限値を上回った点）をセットします。次に AdAllocTrgBuf()を呼び出してリミットポイント発生の前後で取得する変換データ個数とリミットトリガ用バッファを確保します。取得する変換データ個数はプリトリガー時間（トリガ検出前の時間）、ポストトリガー時間（トリガ検出後の時間）の指定が関係してきます。最後に AdStartLimitTrgMode()を呼び出してリミットトリガモードの A/D 変換を実行します。
リミットトリガモードの A/D 変換が開始するとユーザ定義メッセージがユーザプログラムにポストされます。この時、wParam には AD_MODE_RUN がセットされています。リミットトリガ発生後指定時間の変換が終了すると、AD_MODE_STOP がセットされたユーザ定義メッセージがポストされます。この時、lParam には変換データ取り出し先頭アドレスがセットされています。ここで、AdGetTrgData()を呼び出し、上記アドレスを指定して変換データの取り出しを行います。
本サンプルプログラムでは、AdGetTrgData()による変換データの取出しが完了すると AdRestartLimitTrgMode()を呼び出して繰り返し計測を実行します。



上記サンプルプログラムでは、+2.0V でリミットトリガを検出したときに検出前 2 秒間と検出後 3 秒間のデータを取得、グラフ描画したものです。
1 チャンネル当たりの変換個数 ＝ （ ②+③ ）／ ①
リミットトリガ検出前の変換データについては②で設定した時間分のデータが存在しない場合、存在する個数しか取得しませんのでご注意ください。
また、③で設定した時間内で再度リミットトリガが発生しても、データ取得中はリミットトリガの検出は行いません。

サンプルプログラム抜粋

```
BOOL Dlg_OnInitDialog (HWND hwnd, HWND hwndFocus, LPARAM lParam)
{
 /* スロットに挿入されている自分のカードのリソース情報を取得する */
 if ( AdGetCardResource( hwnd, SlotNo, &Adc.MyCardType, &Adc.MyIOBase, &Adc.MyIrqNo ) != 0 )
      return FALSE;

 /* AD 変換時間、AD 変換チャンネル数、プリトリガー時間、ポストトリガー時間の初期値を設定 */
 /* (+)側トリガー電圧、(-)側トリガー電圧の初期値を設定 */
```

```
void Cmd_OnCmdStart ( HWND hwnd )
{
   /* AD 変換時間 */
   Adc.SampTime = (USHORT)GetDlgItemInt( hwnd, IDE_SAMPTIME, NULL, FALSE );
   /* サンプリング時間 */
   Adc.TimeUnitNo = (USHORT)SendDlgItemMessage( hwnd, IDCB_TIMEUNIT, CB_GETCURSEL, 0, 0L );
   /* AD 変換チャンネル数 */
   Adc.AdcChannel = (USHORT)GetDlgItemInt( hwnd, IDE_CHAN, NULL, FALSE );
   /* プリトリガー時間 */
   Adc.PreTime = (USHORT)GetDlgItemInt( hwnd, IDE_PRETRGTIME, NULL, FALSE );
   /* ポストトリガー時間 */
   Adc.PostTime = (USHORT)GetDlgItemInt( hwnd, IDE_POSTTRGTIME, NULL, FALSE );
   /* (+)側トリガー電圧 */
   GetDlgItemText( hwnd, IDE_MAXLIMIT, szBuf, sizeof(szBuf) );
   Adc.MaxLimit = atof( szBuf );
   /* (-)側トリガー電圧 */
   GetDlgItemText( hwnd, IDE_MINLIMIT, szBuf, sizeof(szBuf) );
   Adc.MinLimit = atof( szBuf );

   AdSetParam( hwnd, Adc.MyCardType, INTHISPEED_MODE, Adc.MyIOBase, Adc.MyIrqNo );
   /* 変換チャンネル数設定 */
   AdSetChannel( Adc.AdcChannel, (LPWORD)Sequence );
   /* サンプリング間隔設定 */
   AdSetFreq( hwnd, Adc.SampTime, Adc.TimeUnitNo );
   /* リミットトリガーモードの設定 */
   AdSetLimitTrg( Adc.MaxLimit, Adc.MinLimit, UPPER_LOWER_LIMIT );
   /* リミットトリガー用バッファをアロケート */
   Adc.TrgBufWordSize = AdAllocTrgBuf( Adc.SampTime, Adc.TimeUnitNo, Adc.AdcChannel,
Adc.PreTime, Adc.PostTime );

   /* AD データ格納メモリーをアロケート */
   Adc.pData = LocalAlloc( LPTR, (DWORD)Adc.TrgBufWordSize * 2 );

   /* リミットトリガーAD 変換の実行 */
   AdStartLimitTrgMode( hwnd );
}
```

```
void Dlg_OnUserDefineMessage (HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
   pStartMem = (PWORD)lParam;          /* wParam -> AD 変換終了モード */
   AdcStat = wParam;                   /* lParam -> リングバッファ取り出し先頭アドレス */

   switch( AdcStat ){
   case AD_MODE_RUN:          // 変換開始メッセージ
      break;
   case AD_MODE_STOP:         // 変換終了メッセージ
      /* ﾃﾞｰﾀ取り出し */
      Adc.dwDataNum = AdGetTrgData( pStartMem, Adc.pData );
      if ( Adc.dwDataNum != 0 )
      {
         /* ｸﾞﾗﾌ表示 */

         /* ﾘﾝｸﾞﾊﾞｯﾌｧﾘｾｯﾄとｻﾝﾌﾟﾘﾝｸﾞ再開 */
         AdRestartLimitTrgMode( hwnd );
      }
      break;
   case AD_MODE_OVRUN:        // FIFO オーバーラン停止メッセージ
      break;
   }
}
```

```
void Cmd_OnCmdStop ( HWND hwnd )
{
   SetDlgItemText( hwnd, IDC_STATUS, "変換終了" );
   sprintf( szBuf, "" );
   SetDlgItemText( hwnd, IDS_TOTALNUM, szBuf );
   /* 割り込み登録解除 */
   AdStopLimitTrgMode();
   /* リングバッファを解放 */
   AdFreeTrgBuf();
}
```

3-3. Visual BASIC 言語インターフェイス

3-3-1. DLL ライブラリ API コール

本製品には Windows2000/XP での 32 ビットアプリケーション開発に必要となる API インターフェースを提供する DLL ライブラリ“ADLIB2K.DLL”が添付されています。32 ビットバージョン Visual BASIC で作成したアプリケーションは“ADLIB2K.DLL”の API を呼び出します。

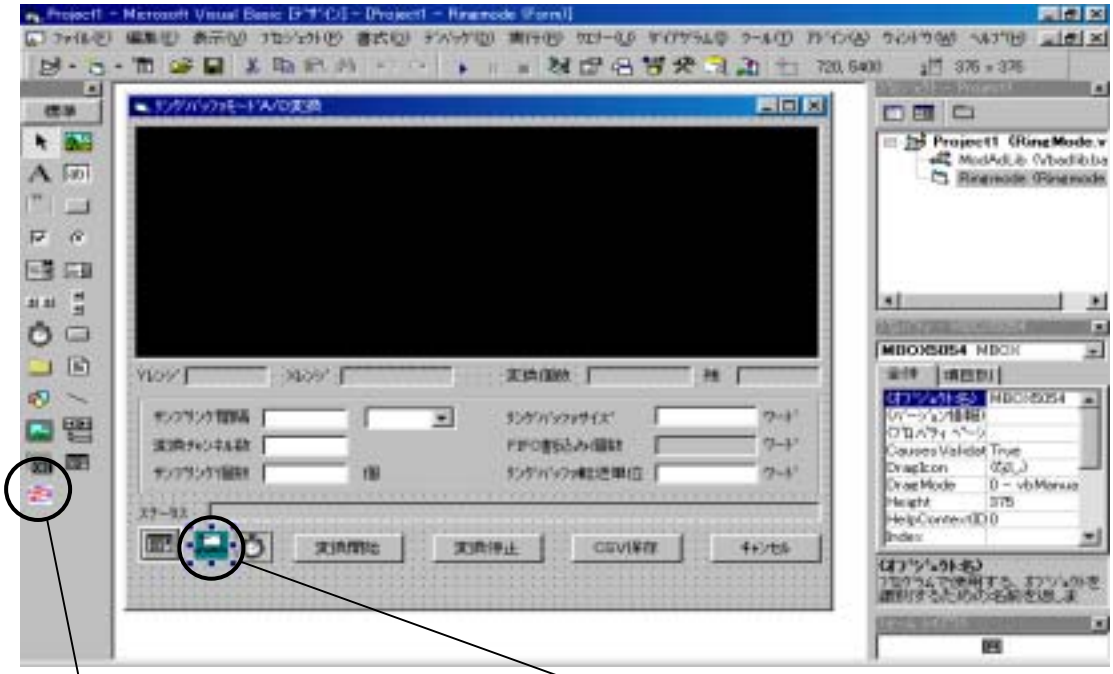
Visual BASIC でアプリケーションを作成する場合、次の二つの内容について理解し、必要となる設定作業を行って下さい。

① DLL ライブラリ API 関数コール

Visual BASIC から“ADLIB2K.DLL”が提供する API 関数を呼び出すためにはモジュール定義ファイルで各 API 関数を Declare 宣言します。API 関数の Declare 宣言は、製品添付のサンプルプログラム“VBADLIB.BAS”を参考にして必要部分をコピーして下さい。

② 割り込みモード AD 変換

割り込みモードの AD 変換を行う場合、割り込み要求に同期したユーザ定義メッセージを Visual BASIC で作成したアプリケーションで受け取る必要があります。Visual BASIC ではユーザ定義メッセージを受け取るコントロールは用意されていませんので、本製品に添付されている OLE カスタムコントロール(OCX)“MBOX5054.OCX”を使用します。



3-3-2. カスタムコントロール

Step.1 → OCX のレジストリー登録 （割り込みサービス使用時必須）

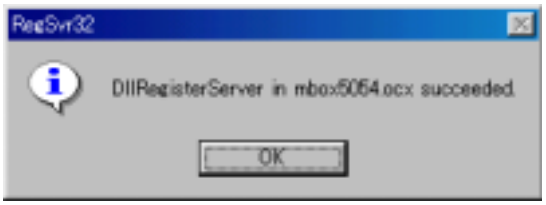
本製品添付の OCX “ MBOX5054.OCX ” を VB で使用するためには、VB の CD-ROM に添付されているツール “ REGSVR32.EXE ” を使って OCX のレジストリ登録を行います。“ REGSVR32.EXE ” は 32 ビットコンソールアプリケーションですので、Windows の DOS BOX から実行します。尚、“ REGSVR32.EXE ” は VB の CD-ROM に添付されています。

OCX をレジストリー登録するときは、下記構文で実行します。

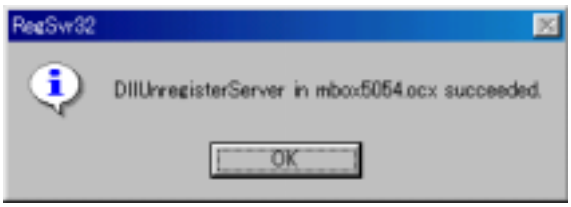
```
>REGSVR32 “ドライブ名”:\WinNT\System\Mbox5054.ocx
```

OCX をレジストリー登録から削除するときは、“ /U ” を付けて下記構文で実行します。

```
>REGSVR32 /U “ドライブ名”:\WinNT\System\Mbox5054.ocx
```



登録成功メッセージ



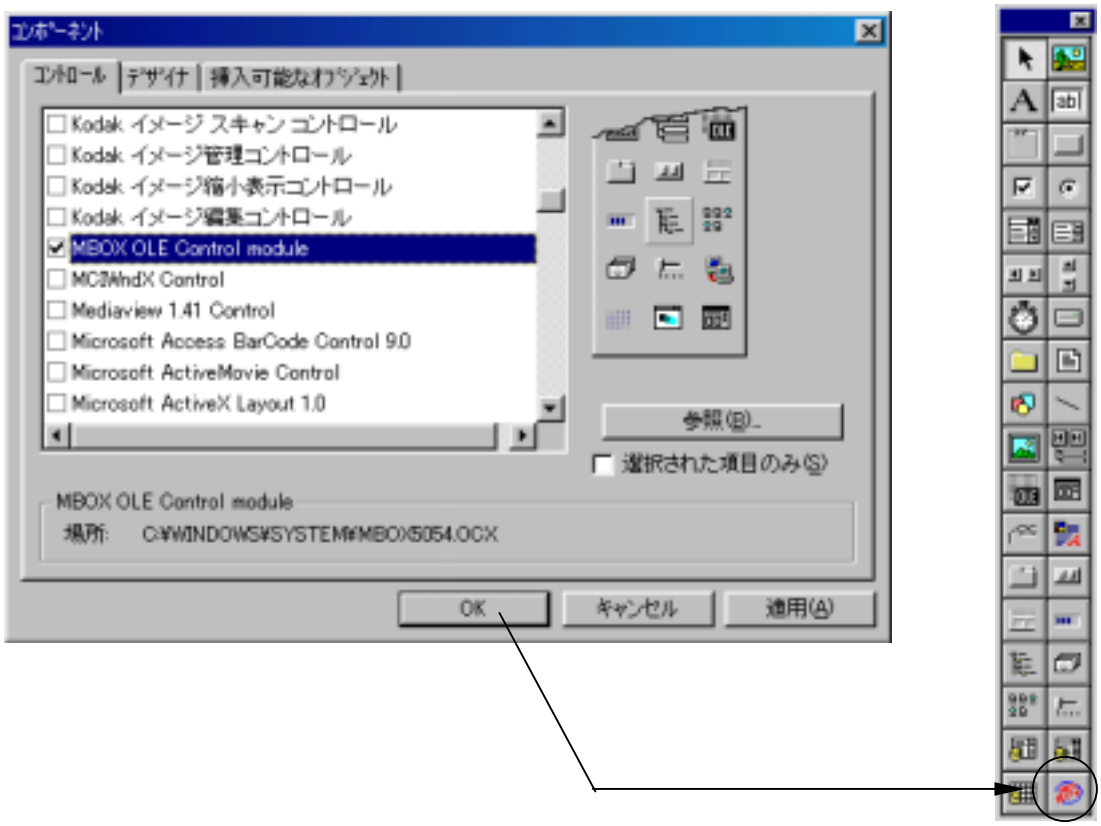
登録削除成功メッセージ

Step.2 → AdLib2K.DLL API 関数の Declare 宣言

次に、VB プログラムの作成に入ります。VB デザインメニューから新規プロジェクトを作成し、「プロジェクト」の「標準モジュールの追加」よりモジュールを追加します。追加した標準モジュールファイルで DLL 関数の参照宣言を行います。宣言部分を、サンプルプログラム “ VBADLIB.BAS ” からコピーして下さい。

Step.3 → MBOX OLE Control Module の追加 （割り込みサービス使用時必須）

VB のカスタムコントロールに MBOX(OCX)を追加します。 VB デザインメニューの「プロジェクト」の「コンポーネント」を起動し、利用可能なコントロールから「MBOX OLE Control module」をチェックします。VB ツールバーに MBOX が追加されます。



3-3-3. DLL ライブラリの Declare 宣言

関数の仕様については、「*3-2-1. DLL ライブラリ API 解説*」を参照してください。

★★★ ご注意 ★★★
DLL に確保要求して取得したメモリアドレスを渡す場合は
"ByVal MemAdrs As Long"宣言１）を有効にします
VB 内で確保した変数及び配列のアドレスを DLL へ渡す場合は
"VBArrayName As Any"宣言２）を有効にします

AdAllocDataMem AD 変換データ格納用メモリのアロケーション
```
Declare Function AdAllocDataMem Lib "Adlib2k.dll" (ByVal hwnd As Long, ByVal
 DataLength As Long) As Long
```

AdAllocMem 汎用メモリのアロケーション
```
Declare Function AdAllocMem Lib "Adlib2k.dll" (ByVal hwnd As Long, ByVal MemSize
As Long) As Long
```

AdAllocRingBuf リングバッファのアロケーション
```
Declare Function AdAllocRingBuf Lib "Adlib2k.dll" (ByVal hwnd As Long, ByVal
wAllocSize As Long, ByVal DataLength As Long) As Long
```

AdAllocTrgBuf 入力電圧リミットトリガーバッファのアロケーション
```
Declare Function AdAllocTrgBuf Lib "Adlib2k.dll" (ByVal SampTime As Integer, ByVal
TimeUnit As Integer, ByVal AdChan As Integer, ByVal PreTime As Integer, ByVal
PostTime As Integer) As Long
```

AdFreeDataMem AD 変換データ格納用メモリを解放
```
Declare Sub AdFreeDataMem Lib "Adlib2k.dll" ()
```

AdFreeMem 汎用メモリの解放
```
Declare Sub AdFreeMem Lib "Adlib2k.dll" (ByVal pHeapMem As Long)
```

AdFreeRingBuf リングバッファの解放
```
Declare Sub AdFreeRingBuf Lib "Adlib2k.dll" ()
```

AdFreeTrgBuf 入力電圧リミットトリガバッファの解放
```
Declare Sub AdFreeTrgBuf Lib "Adlib2k.dll" ()
```

AdGetCardResource　　REX5054U/B に割り当てられているリソースの取得

```
Declare Function AdGetCardResource Lib "Adlib2k.dll" (ByVal hwnd As Long, ByVal
SlotNo As Integer, pCardType As Any, pIOBase As Any, pIrqNo As Any) As Long
```

AdGetTrgData　　入力電圧リミットトリガバッファから変換データを転送

```
Declare Function AdGetTrgData Lib "Adlib2k.dll" (ByVal pStartMem As Long, pCopyMem
As Any) As Long
```

AdOneShot　　ワンショットモード AD 変換実行

```
' 宣言１） Declare Function AdOneShot Lib "Adlib2k.dll" (ByVal hwnd As long, ByVal
lpMem As Long) As long
' 宣言２）
Declare Function AdOneShot Lib "Adlib2k.dll" (ByVal hwnd As Long, AdData As Any)
As Long
```

AdSetChannel　　変換チャンネルパラメータの設定

```
Declare Function AdSetChannel Lib "Adlib2k.dll" (ByVal Channels As Integer,
lpSequence As Any) As Long
```

AdSetDataCount　　サンプリング個数の設定

```
Declare Function AdSetDataCount Lib "Adlib2k.dll" (ByVal DataCount As Long) As Long
```

AdSetExternalClock　　外部クロック入力の設定

```
Declare Function AdSetExternalClock Lib "Adlib2k.dll" (ByVal Mode As Long, ByVal
Counter0 As Integer, ByVal Counter1 As Integer) As Long
```

AdSetFreq　　サンプリング間隔の設定

```
Declare Function AdSetFreq Lib "Adlib2k.dll" (ByVal hwnd As Long, ByVal Stime As
Integer, ByVal Unit As Integer) As Long
```

AdSetInternalClock　　内部クロックの選択とサンプリング間隔の手動設定

```
Declare Function AdSetInternalClock Lib "Adlib2k.dll" (ByVal UseClock As Long, ByVal
Counter0 As Integer, ByVal Counter1 As Integer) As Long
```

AdSetLimitTrg　　入力電圧リミットトリガ値の設定

```
  Declare Function AdSetLimitTrg Lib "Adlib2k.dll" (ByVal MaxLimit As Double, ByVal
MinLimit As Double, ByVal TrgMode As Integer) As Long
```

AdSetOneParam　　ワンショット AD 変換パラメータ設定

```
Declare Function AdSetOneParam Lib "Adlib2k.dll" (ByVal hwnd As Long, ByVal
wCardType As Integer, ByVal wUseIOAddr As Integer) As Long
```

AdSetParam カード情報の手動設定

```
Declare Function AdSetParam Lib "Adlib2k.dll" (ByVal hwnd As Long, ByVal CardType As Integer, ByVal SampMode As Long, ByVal IOBase As Integer, ByVal IrqNo As Integer) As Long
```

AdSetParamAuto カード情報の自動設定

```
Declare Function AdSetParamAuto Lib "Adlib2k.dll" (ByVal hwnd As Long, ByVal SampMode As Integer) As Long
```

AdSetTrigger 外部トリガーによる変換開始･終了の設定

```
Declare Sub AdSetTrigger Lib "Adlib2k.dll" (ByVal TrgMode As Integer, ByVal RiseOrFall As Integer)
```

AdStartIrqRingMode リングバッファ割込み AD 変換実行

```
Declare Function AdStartIrqRingMode Lib "Adlib2k.dll" (ByVal hwnd As Long, ByVal IrqSetCount As Integer) As Long
```

AdStartIrqVxdMode 高速割込みモード AD 変換実行

```
Declare Function AdStartIrqVxdMode Lib "Adlib2k.dll" (ByVal hwnd As Long, ByVal IrqSetCount As Integer) As Long
```

AdStartLimitTrgMode リミットトリガモード AD 変換実行

```
Declare Function AdStartLimitTrgMode Lib "Adlib2k.dll" (ByVal hwnd As Long) As Long
```

AdRestartLimitTrgMode リミットトリガモード AD 変換繰返し実行

```
Declare Function AdReStartLimitTrgMode Lib "Adlib2k.dll" (ByVal hwnd As Long) As Long
```

AdStartPolModeThread ポーリングモード AD 変換チャイルドスレッド

```
Declare Function AdStartPolModeThread Lib "Adlib2k.dll" (ByVal hwnd As Long) As Long
```

AdStopIrqMode 割り込み AD 変換停止とリソースを解除

```
Declare Sub AdStopIrqMode Lib "Adlib2k.dll" ()
```

AdStopLimitTrgMode リミットトリガモード AD 変換停止とリソース解除

```
Declare Sub AdStopLimitTrgMode Lib "Adlib2k.dll" ()
```

GetRingBufNum リングバッファから変換データ個数を転送

```
Declare Function GetRingBufNum Lib "Adlib2k.dll" () As Long
```

VbGetRingBuf　リングバッファから変換データを転送

書 式　Declare Function **VbGetRingBuf** Lib "Adlib2k.dll" (ByVal **AdRing** As Any, ByVal **Offset** As Long, ByVal **GetDataNum** As Long) As Long

機 能　リングバッファから指定個数の変換データをリードし指定のアドレスへ転送します。データの転送先アドレスは AdRing から Offset で示される変換データ個分オフセットした場所になります。
本関数は第三引数で指定した個数分の変換データがリングバッファにない場合は格納されている個数分を転送します。

引 数
- **AdRing** ➢ VB 内部で確保されているデータ格納先のアドレス
- **Offset** ➢ データ格納先のアドレスのオフセット
- **GetDataNum** ➢ 取得するデータ数

戻 値　転送した変換データ個数を返します。リングバッファオーバーフロー発生時は -1 を返し、ドライバ呼び出しエラーは、-10 を返します。

VbGetRingBufConst　リングバッファからの一定個数変換データを転送

書 式　Declare Function **VbGetRingBufConst** Lib "Adlib2k.dll" (ByVal **AdRing** As Any, ByVal **Offset** As Long, ByVal **GetDataNum** As Long) As Long

機 能　リングバッファから指定個数の変換データをリードし指定のアドレスへ転送します。データの転送先アドレスは AdRing から Offset で示される変換データ個分オフセットした場所になります。
本関数は **VbGetRingBuf**()とは異なり、第三引数で指定した個数分の変換データがリングバッファにない場合は転送を行わず、戻り値として 0 を返します。

引 数
- **AdRing** ➢ VB 内部で確保されているデータ格納先のアドレス
- **Offset** ➢ データ格納先のアドレスのオフセット
- **GetDataNum** ➢ 取得するデータ数

戻 値　転送した変換データ個数を返します。
正常終了時、0 を返します。リングバッファオーバーフロー発生時には-1 を返し、ドライバ呼び出しエラーは、-10 を返します。

VbMemCopy　Visual BASIC の配列へのデータ転送

書 式　Declare Function **VbMemCopy** Lib "Adlib2k.dll" (ByVal **VbMem** As Any, ByVal **DllMem** As Long, ByVal **CopyByteSize** As Long) As Long

機 能　**Visual BASIC** で確保した配列へ **DLL** 内部に確保されているメモリから指定バイトサイズ転送します。

引 数
- **VbMem** ➢ **コピー先メモリアドレス**
- **DllMem** ➢ **コピー元メモリアドレス**
- **CopyByteSize** ➢ **コピーバイト数**

戻 値　**コピー先メモリアドレスを返します。**

3-3-4. Visual BASIC サンプルプログラム解説

Visual BASIC 4.0 以上のバージョンを使って REX-5054U/B AD PC カードを制御するアプリケーションを開発する場合は、本製品に添付されているサンプルプログラムを参考にして下さい。
添付のサンプルプログラムを使用してプログラム作成・修正する場合は、各サンプルの FRM ファイルで新規プロジェクトを作成し、標準モジュール（VBADLIB.BAS）をプロジェクトに追加してコンパイルを行ってください。

AD 変換を行うモードには大きく分けて次の二つがあります。

→ ポーリングモード
プログラムから AD カードの FIFO に変換データがあるかどうか調べ、変換データがあればデータを取り出すモード
→ 割り込みモード
FIFO に設定した個数の変換データがセットされた時点で AD カードから割り込み要求を行い割り込み処理で FIFO からデータを取り出すモード

以下のモードのサンプルプログラムが製品添付されています。

ポーリングモード	
OneShot	ワンショットモード AD 変換プログラム
PolMode	ポーリングモード AD 変換を行うプログラム
割り込みモード	
IntBase	割り込みモード AD 変換プログラム
RingMode	リングバッファを使った割り込みモード AD 変換プログラム
LimitTrg	リミットトリガーモード AD 変換プログラム

OneShot → ワンショットモード AD 変換プログラム

ワンショットモードでは、AdOneShot()を呼び出しポーリングモードで 1 回だけサンプリングを行い電圧値をダイアログ画面に表示します。繰り返し計測は、指定の繰返し周期でデータを取得するためにタイマー処理で AdOneShot()を呼び出し、指定回数分だけ変換を繰返します。

サンプルプログラム抜粋

```
Private Sub IDB_START_Click()

    'タイマー起動
    uPeriod = IDE_PERIOD.Text
    OneShotTimer.Interval = uPeriod

End Sub
```

```
Private Sub OneShotTimer_Timer()

    If MyAdOneShot(hwnd) <> 0 Then
        Exit Sub
    End If
End Sub
```

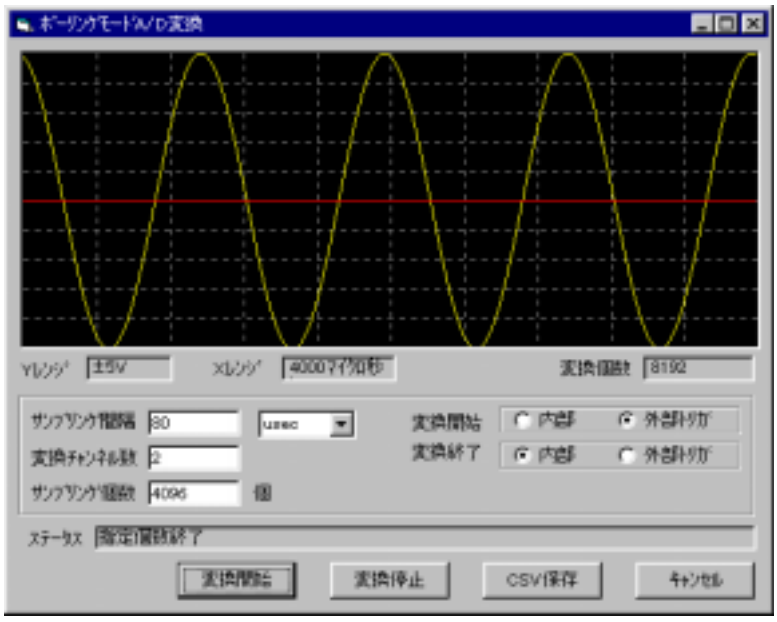
```
Private Function MyAdOneShot(ByVal hwnd As Long) As Long

    ' A/D 変換実行（常に全ﾁｬﾝﾈﾙの変換を行います）
    Status = AdOneShot(hwnd, AdBuf(0))
    If Status <> 0 Then
        IDS_STATUS.Caption = "ﾜﾝｼｮｯﾄ A/D 変換ｴﾗｰ [CODE:" + Str(Status) + "]"
        MyAdOneShot = -1    '戻り値
    End If

    ' 電圧値を表示
    Select Case MyCardType
    Case REX5054U
            AdVal = ((AdBuf(Chloop) And &H1FFF) * (2 ^ 3)) * 2.5 / 32768#
    Case REX5054B:
            AdVal = ((AdBuf(Chloop) And &H1FFF) * (2 ^ 3)) * 10# / 32768# - 5#
    End Select
    MyAdOneShot = 0   '戻り値
End Function
```

PolMode → ポーリングモード AD 変換を行うプログラム

DLL でイクスポートしているポーリングモードの A/D 変換ルーチンである **AdStartPolModeThread()**を呼び出すことにより割り込みを禁止して A/D 変換が開始します。AdStartPolModeThread()は変換の開始と終了時ユーザ定義メッセージを呼び出し元ウィンドウへポストします。呼び出されたアプリケーションはこの時の wParam・lParam からステータスと変換個数情報をコントロール MSGBOX を配置して受け取るようにします。

A/D 変換データは、**AdAllocDataMem()**でアロケーションしたメモリに格納されます。AdAllocDataMem()はアロケーションしたメモリへのポインターを返しますので、ユーザ側のアプリケーションから変換データにアクセスします。

VB アプリケーション側ではデータを保管するための配列を確保します。**VbMemCopy()**を呼び出してドライバがセットした変換データを自分のメモリへ転送保管します。



変換開始・終了はボタン操作の他に外部トリガを使用することも可能です。

サンプルプログラム抜粋

```
Private Sub PolTimer_Timer()
    Dim OleHandle As Long           ' MBOX.OCX ハンドル

    'OLE のウィンドウハンドル取得
    OleHandle = MBOX5054.GetMboxWnd()

    'ポーリングモード AD 変換実行
    AdStartPolModeThread OleHandle
    'タイマー停止
    PolTimer.Interval = 0

End Sub
```

```
Private Sub IDB_START_Click()

    ' AD 変換時間、サンプリング時間、AD 変換チャンネル数、AD 変換個数の取得
    ' 変換パラメータのセット
    AdSetParam hwnd, Adc.MyCardType, INTHISPEED_MODE, Adc.MyIOBase, Adc.MyIrqNo

    ' 変換チャンネル数設定
    AdSetChannel Adc.AdcChannel, Sequence(1)
    ' 外部トリガで開始、指定個数で終了
    AdSetTrigger POST_TRIGGER, PULSE_RISING
    ' サンプリング間隔設定
    AdSetFreq hwnd, Adc.SampTime, Adc.TimeUnitNo
    ' VB アプリケーション側でデータを保管するための配列を確保
    Adc.MaxSampDataNum = Adc.AdcChannel * Adc.dwSampCount
    ReDim Adc.pData(Adc.MaxSampDataNum) As Integer

    ' AD データ格納メモリーをアロケート(1 チャンネル当たりの変換個数を指定する)
    DllMem = AdAllocDataMem(hwnd, Adc.dwSampCount)
    If DllMem = 0 Then
        Exit Sub
    End If
    '10msec 間隔で PolTimer を起動
    PolTimer.Interval = 10
End Sub
```

```
Private Sub MBOX5054_OnMsgPost(ByVal wParam As Integer, ByVal lParam As Long)

    AdCount = lParam            ' lParam -> AD 変換データカウント数
    AdcStat = wParam            ' wParam -> AD 変換開始終了コード

    Select Case AdcStat
    Case AD_MODE_RUN            ' 変換開始メッセージ
        Exit Sub
    Case AD_MODE_TRGRUN         ' トリガー変換開始メッセージ
        Exit Sub
    Case AD_MODE_STOP           ' 指定個数終了メッセージ
    Case AD_MODE_PRESTOP        ' トリガー終了メッセージ
    Case AD_MODE_OVRUN       ' FIFO オーバーラン停止メッセージ
    End Select

    ' ドライバがセットした変換データを自分のメモリへ転送保管します
    ' 1 個の変換データは 2 バイトですので転送先のアドレスに注意して下さい
    VbMemCopy Adc.pData(0), DllMem, AdCount * 2
    ' グラフ描画処理
    PicGraph.Cls    '一旦消去
    PlotData
    'AD データ格納メモリー開放
    AdFreeDataMem

End Sub
```

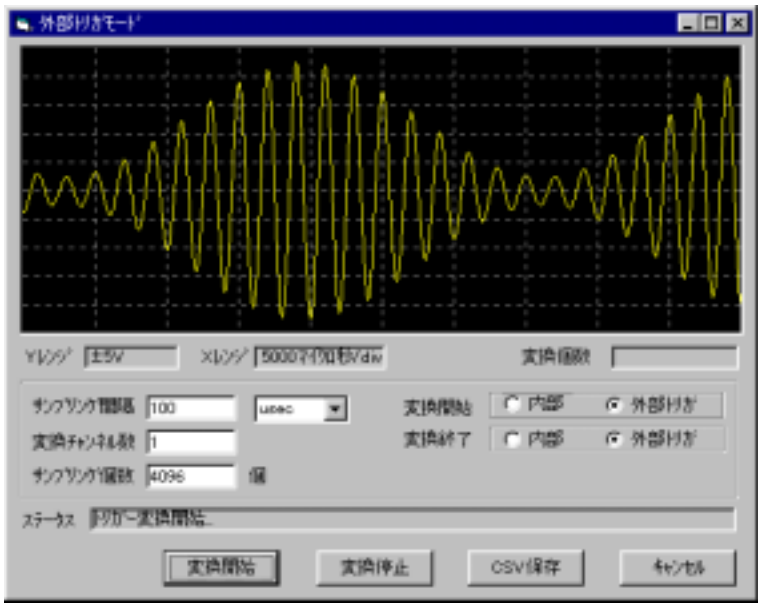
IntBase → 割り込みモード AD 変換プログラム

AdStartIrqVxdMode()を呼び出すことによりシステム側処理のオーバーヘッドを伴わず高速に割り込みモード AD 変換を実行します。このモードではドライバは指定個数の A/D 変換が終了するまで呼び出し元プログラムに一切の通知を行いません。AdStartIrqVxdMode()は変換の開始及び変換データ個数が指定個数に達するとドライバから呼び出し元プログラムにメッセージをポストします。この時、wParam に AD 変換開始終了メッセージコードが、lParam に終了時の変換データ数がセットされています。コントロール MSGBOX を配置してこれらを受け取るようにします。

A/D 変換データは、**AdAllocDataMem()**でアロケーションしたメモリに格納されます。AdAllocDataMem()はアロケーションしたメモリへのポインターを返しますので、ユーザ側のアプリケーションから変換データにアクセスすることができます。VB アプリケーション側ではデータを保管するための配列を確保します。**VbMemCopy()**を呼び出してドライバがセットした変換データを自分のメモリへ転送保管します。

本サンプルプログラムでは、変換終了後、再び AdStartIrqVxdMode() を呼び出して繰り返し次の割り込みモード A/D 変換を実行しています。

変換開始・終了はボタン操作の他に外部トリガを使用することも可能です。



サンプルプログラム抜粋

```
Private Sub IDB_START_Click()

    ' AD 変換時間、サンプリング時間、AD 変換チャンネル数、AD 変換個数の取得
    ' 変換パラメータのセット
    AdSetParam hwnd, Adc.MyCardType, INTHISPEED_MODE, Adc.MyIOBase, Adc.MyIrqNo

    ' 変換チャンネル数設定
    AdSetChannel Adc.AdcChannel, Sequence(1)
    ' 外部トリガで開始、指定個数で終了
    AdSetTrigger POST_TRIGGER, PULSE_RISING
    ' サンプリング間隔設定
    AdSetFreq hwnd, Adc.SampTime, Adc.TimeUnitNo
    ' VB アプリケーション側でデータを保管するための配列確保.必ず FIFO サイズ余分に確保
    Adc.MaxSampDataNum = Adc.AdcChannel * Adc.dwSampCount
    ReDim Adc.pData(Adc.MaxSampDataNum + 32) As Integer

    ' AD データ格納メモリーをアロケート(1 チャンネル当たりの変換個数を指定する)
    DllMem = AdAllocDataMem(hwnd, Adc.dwSampCount)
    If DllMem = 0 Then
        Exit Sub
    End If
    fConversionStop = False
    'OLE のウィンドウハンドル取得
    OleHandle = MBOX5054.GetMboxWnd()
    ' 割り込みモード AD 変換の実行
    AdStartIrqVxdMode OleHandle, 0
End Sub
```

```
Private Sub IDB_STOP_Click()
    ' 繰り返し変換実行時は、ストップボタンが押されたとき、ドライバ内部で変換を
    ' 行っている場合は AdStopIrqMode()で割り込みリソースを解放しないようにする。
    ' 本サンプルでは、ストップボタンが押されたことを示すフラグを用意し、
    ' MBOX5054_OnMsgPost()で次の変換を開始しないようにします。

    fConversionStop = True
End Sub
```

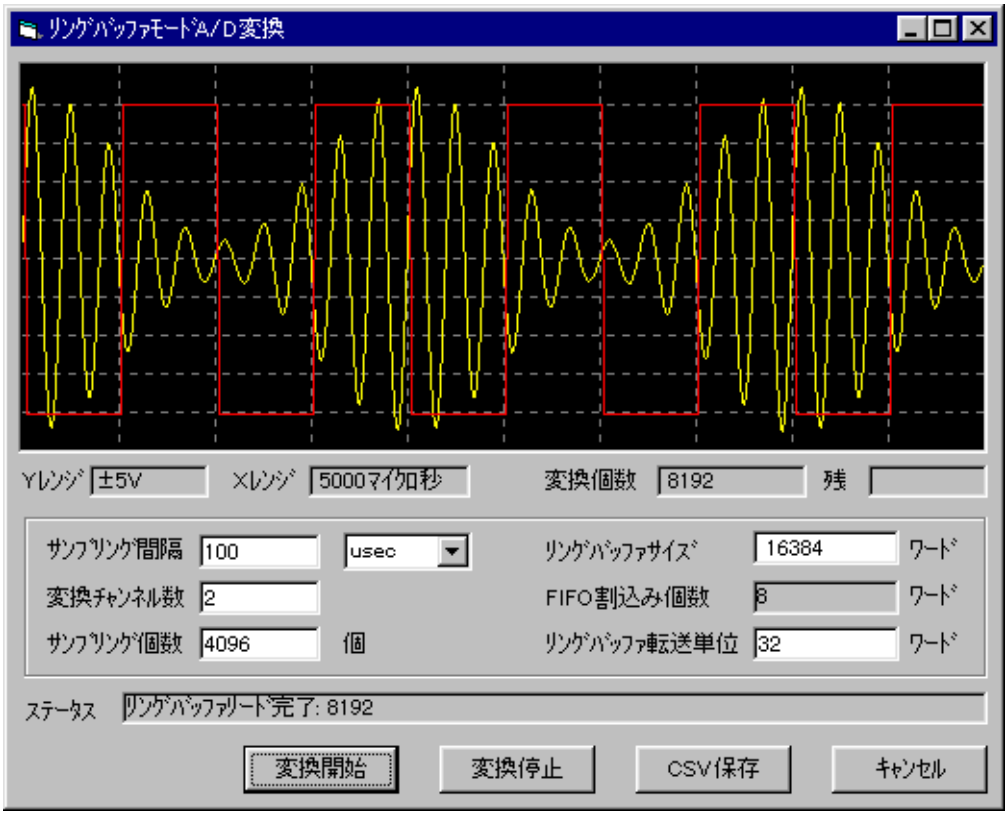
```
Private Sub MBOX5054_OnMsgPost(ByVal wParam As Integer, ByVal lParam As Long)

   AdCount = lParam                    ' lParam -> AD 変換データカウント数
   AdcStat = wParam                    ' wParam -> AD 変換開始終了コード

   Select Case AdcStat
   Case AD_MODE_RUN                    ' 変換開始メッセージ

   Case AD_MODE_TRGRUN               ' トリガー変換開始メッセージ

   Case AD_MODE_PRESTOP                ' トリガー終了メッセージ
      Beep
      ' ﾄﾞﾗｲﾊﾞがｾｯﾄした変換ﾃﾞｰﾀを自分のﾒﾓﾘへ転送保管します
      ' 1 個の変換ﾃﾞｰﾀは 2 ﾊﾞｲﾄですので転送先のｱﾄﾞﾚｽに注意して下さい
      VbMemCopy Adc.pData(0), DllMem, AdCount * 2

      '##### 必要ならここでﾌｧｲﾙ保存 #####
      ' グラフ描画処理
      AdStopIrqMode                    ' 変換を停止し割り込みリソース解放
      AdFreeDataMem                    ' AD データ格納メモリー開放
   Case AD_MODE_STOP                   ' 指定個数終了メッセージ
      Beep
      ' ドライバがセットした変換データを自分のメモリへ転送保管します
      ' 1 個の変換データは 2 バイトですので転送先のアドレスに注意して下さい
      VbMemCopy Adc.pData(0), DllMem, AdCount * 2

      '##### 必要ならここでﾌｧｲﾙ保存 #####
      ' グラフ描画処理
      ' 変換停止ボタンが押されていないかチェック
      If fConversionStop <> True Then
         ' 繰り返し次の割り込みモード AD 変換の実行
         AdStartIrqVxdMode OleHandle, 0
      Else
         AdStopIrqMode                    ' 変換を停止し割り込みリソース解放
         AdFreeDataMem                    ' AD データ格納メモリー開放
      End If

   Case AD_MODE_OVRUN                  ' FIFO オーバーラン停止メッセージ
      ' ｸﾞﾗﾌ描画処理
      AdStopIrqMode                    ' 変換を停止し割り込みリソース解放
   End Select
End Sub
```

RingMode → リングバッファを使った割り込みモード AD 変換プログラム

リングバッファを使った割り込みモード AD 変換 AdStartIrqRingMode()をスタートすると A/D カードは FIFO に指定個数の変換データがセットされると割り込み要求を行います。ドライバーの割り込み処理では FIFO からデータを AdAllocRingBuf()で設定されたリングバッファへ転送します。

VB アプリケーション側ではデータを保管するための配列を確保します。A/D 変換スピードに応じたリングバッファからの変換データ取り出し間隔と個数の目安値を設定してタイマー処理で VbGetRingBufConst()を呼び出してドライバがセットした変換データを自分のメモリへ転送保管します。タイマー処理ではグラフ表示も行います。

ドライバーがリングバッファへ変換データを転送するレートに対し、アプリケーションがリングバッファからデータを取り出すスピードが遅れるとある時点でリングバッファがオーバーフローします。すなわち、ドライバーはリングバッファの最も古いデータに上書きしますのでそのデータは失われます。



サンプルプログラム抜粋

```
Private Sub IDB_START_Click()

    ' AD 変換時間、サンプリング時間、AD 変換チャンネル数、AD 変換個数の取得
    ' 割込みを発生する FIFO データ個数とリングバッファから転送するデータ個数設定
    InitRingParam Adc.SampTime, Adc.TimeUnitNo, Adc.AdcChannel, Adc.IrqUpNum,
Adc.RingGetDataCount

    ' カードリソース情報の自動設定
    AdSetParamAuto hwnd, INTHISPEED_MODE
    ' 変換チャンネル数設定
    AdSetChannel Adc.AdcChannel, Sequence(1)
    ' サンプリング間隔設定
    AdSetFreq hwnd, Adc.SampTime, Adc.TimeUnitNo
    ' RingBufSize ワードのリングバッファ作成
    AdAllocRingBuf hwnd, RingBufSize, Adc.dwSampCount

    'OLE のウィンドウハンドル取得
    OleHandle = MBOX1.GetMboxWnd()
    ' リングバッファを使った割込みによる AD 変換開始
    RetCode = AdStartIrqRingMode(OleHandle, Adc.IrqUpNum)
    If RetCode <> 0 Then
        AdFreeRingBuf
        Exit Sub
    End If

    ' VB アプリケーション側でデータを保管するためのメモリーを確保
    Adc.MaxSampDataNum = Adc.dwSampCount * Adc.AdcChannel
    ReDim Adc.pData(Adc.MaxSampDataNum) As Integer

    BreakFlag = False
    ' リングバッファデータ取得インターバルタイマー起動
    RingTimer.Interval = 10
End Sub
```

```
Private Sub IDB_STOP_Click()
    BreakFlag = True
    ' 変換を強制的に止める
    AdStopIrqMode
    ' ﾘﾝｸﾞﾊﾞｯﾌｧを解放
    AdFreeRingBuf

    RingTimer.Interval = 0
End Sub
```

```
' AD 変換開始終了のメッセージ、変換個数の通知
Private Sub MBOX1_OnMsgPost(ByVal wParam As Integer, ByVal lParam As Long)

    ' lParam -> AD 変換完了データ個数(チャンネル×変換個数+α)カウント数
    NumberOfAdcData = lParam
    AdcStat = wParam                ' wParam -> AD 変換終了ﾓｰﾄﾞ

    Select Case wParam
        Case AD_MODE_RUN           ' 変換開始メッセージ
        Case AD_MODE_TRGRUN        ' トリガー変換開始メッセージ
        Case AD_MODE_STOP          ' 指定個数終了メッセージ
        Case AD_MODE_PRESTOP       ' トリガー変換終了メッセージ
        Case AD_MODE_OVRUN         ' FIFO オーバーラン停止メッセージ
    End Select
End Sub
```

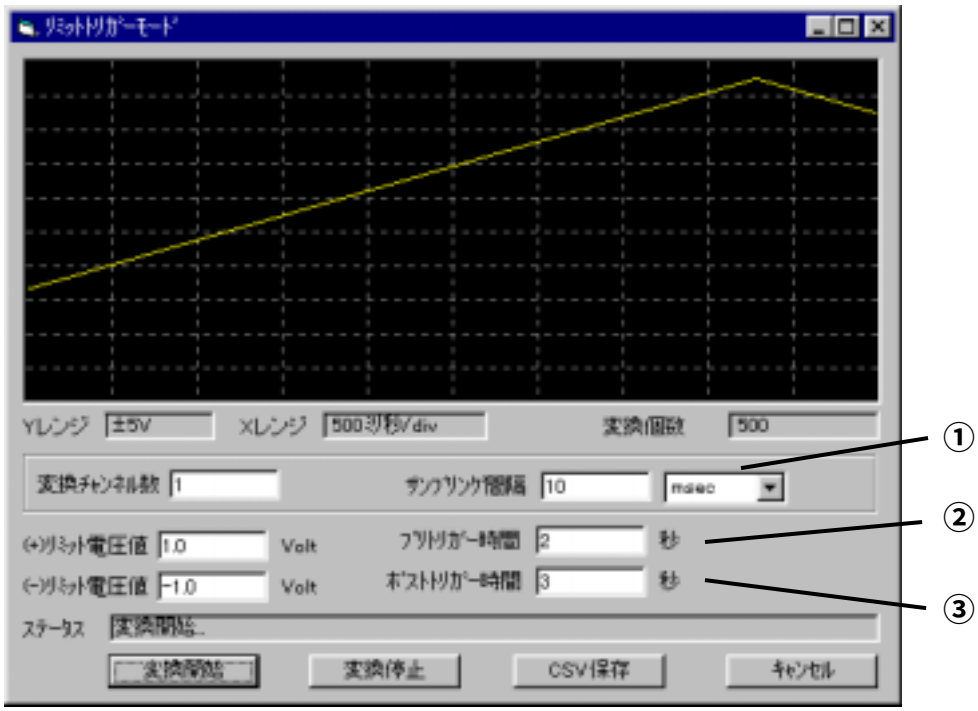
```
Private Sub RingTimer_Timer()

    ' 変換停止・終了を判別。BreakFlag = True の時（停止・終了時）はタイマー処理を終了。
    If BreakFlag = True Then
        RingTimer.Interval = 0
        Exit Sub
    End If

    ' リングバッファにある残りのデータ個数を求める
    Remain = Adc.MaxSampDataNum - Adc.TotalNum
    ' リングバッファからデータを取得し自分のデータバッファへ転送する
    If Remain >= Adc.RingGetDataCount Then
        GetCount = VbGetRingBufConst(Adc.pData(0), Adc.TotalNum, Adc.RingGetDataCount)
    Else
        GetCount = VbGetRingBufConst(Adc.pData(0), Adc.TotalNum, Remain)
    End If


    If GetCount = -1 Then
        ' リングバッファがオーバーフローしたので変換を停止します
        RingTimer.Interval = 0  'タイマー停止
        AdStopIrqMode
        AdStartFlag = False
    ElseIf GetCount > 0 Then
        ' リングバッファから転送したデータを表示
        ' グラフ描画処理
    End If

    ' 指定個数に達したら     ' リングバッファリードを停止
    If Adc.TotalNum >= Adc.MaxSampDataNum Then
        RingTimer.Interval = 0
        BreakFlag = True
    End If
End Sub
```

LimitTrg → リミットトリガモード AD 変換プログラム

[変換開始]ボタンが押されると AdSetLimitTrg()が呼び出され、(+)(-)リミット電圧値で指定したリミットトリガ（入力電圧が設定上限値もしくは下限値を上回った点）をセットします。次に AdAllocTrgBuf()を呼び出してリミットトリガ発生の前後で取得する変換データ個数とリミットトリガ用バッファを確保します。取得する変換データ個数はプリトリガー時間、ポストトリガー時間の指定が関係してきます。最後に AdStartLimitTrgMode()を呼び出してリミットトリガモードの A/D 変換を実行します。

A/D 変換が開始するとユーザ定義メッセージがユーザプログラムにポストされます。この時、wParam には AD_MODE_RUN がセットされています。リミットトリガ発生後指定時間の変換が終了すると、AD_MODE_STOP がセットされたユーザ定義メッセージがポストされます。この時、lParam には変換データ取り出し先頭アドレスがセットされています。ここで、AdGetTrgData()を呼び出し、上記アドレスを指定して変換データの取り出しを行います。

本サンプルプログラムでは、AdGetTrgData()による変換データの取出しが完了すると AdRestartLimitTrgMode()を呼び出して繰り返し計測を実行します。



上記サンプルプログラムでは、+2.0V でリミットトリガを検出したときに検出前 2 秒間と検出後 3 秒間のデータを取得、グラフ描画したものです。

1 チャンネル当たりの変換個数 ＝ （ ②+③ ）／ ①

リミットトリガ検出前の変換データについては②で設定した時間分のデータが存在しない場合、存在する個数しか取得しませんのでご注意ください。

また、③で設定した時間内で再度リミットトリガが発生しても、データ取得中はリミットトリガの検出は行いません。

サンプルプログラム抜粋

```
Private Sub IDB_START_Click()

    ' AD 変換時間、サンプリング時間、AD 変換チャンネル数の取得
    ' プリトリガー時間、ポストトリガー時間、(+)側トリガー電圧、(-)側トリガー電圧の取得

    AdSetParam hwnd, Adc.MyCardType, INTHISPEED_MODE, Adc.MyIOBase, Adc.MyIrqNo
    ' 変換チャンネル数設定
    AdSetChannel Adc.AdcChannel, Sequence(1)
    ' サンプリング間隔設定
    AdSetFreq hwnd, Adc.SampTime, Adc.TimeUnitNo

    ' リミットトリガーモードの設定
    AdSetLimitTrg Adc.MaxLimit, Adc.MinLimit, UPPER_LIMIT
    '取得する変換データ個数とバッファを確保
    Adc.TrgBufWordSize  =  AdAllocTrgBuf(Adc.SampTime,  Adc.TimeUnitNo,  Adc.AdcChannel,
Adc.PreTime, Adc.PostTime)
    If Adc.TrgBufWordSize <= 0 Then
        Exit Sub
    End If
    ReDim Adc.pData(Adc.TrgBufWordSize * 2)

    'OLE のウィンドウハンドル取得
    OleHandle = MBOX1.GetMboxWnd()
    ' リミットトリガーモード AD 変換の実行
    AdStartLimitTrgMode OleHandle

End Sub
```

```
Private Sub IDB_STOP_Click()
    ' 割り込み登録解除
    AdStopLimitTrgMode
    ' リングバッファを解放
    AdFreeTrgBuf
    IDS_TOTALNUM.Caption = ""
    IDC_STATUS.Caption = "変換終了"

End Sub
```

```
Private Sub MBOX5054_OnMsgPost(ByVal wParam As Integer, ByVal lParam As Long)

    pStartMem = lParam          ' lParam -> リングバッファ取り出し先頭アドレス
    AdcStat = wParam               ' wParam -> AD 変換終了モード

    Select Case AdcStat
        Case AD_MODE_RUN           ' 変換開始メッセージ
        Case AD_MODE_STOP          ' 指定個数終了メッセージ
            ' データ取り出し
            Adc.dwDataNum = AdGetTrgData(pStartMem, Adc.pData(0))
            If Adc.dwDataNum <> 0 Then
                IDS_TOTALNUM.Caption = Str(Adc.dwDataNum)
                ' グラフ表示処理
                ' リングバッファリセットとサンプリング再開
                RetCode = AdRestartLimitTrgMode(OleHandle)
                If RetCode <> 0 Then
                    IDC_STATUS.Caption = "AD 変換再開ｴﾗｰ(ｺｰﾄﾞ:" + Str(RetCode) + ")"
                End If
            End If
        Case AD_MODE_OVRUN      ' FIFO オーバーラン停止メッセージ
    End Select

End Sub
```

第4章 A/Dカード詳細仕様

4-1. 概要仕様

REX5054U/B は A/D コンバータにナショナルセミコンダクター社のデータアクイジョンシステム LM12H458 を採用しています。内部クロックとして 10ＭＨｚと 8.192ＭＨｚの二つのクロックを搭載しており、どちらかを選択することができます。プログラマブルタイマカウンタμＰＤ７１０５４の分周回路でクロック分周を行い、A/D 変換のソースクロックとして出力されます。また、外部クロック端子から入力した信号をプログラマブルタイマカウンタを通して A/D 変換のソースクロックとすることも可能です。A/D 変換の開始および終了は通常ソフトウェアで制御して行いますが、外部ﾄﾘｶﾞ端子から入力した信号で制御することも可能です。

	REX-5054U	REX-5054B
入力チャンネル数	8 チャンネル	4 チャンネル
入力電圧範囲	0～＋2.5Ｖ	－5 ～＋5Ｖ
許容最大入力電圧	－3～＋5.5Ｖ	－8 ～＋8Ｖ
設定可能サンプリングクロック	内部クロック 10 ＭＨｚ 8.192ＭＨｚ 外部クロック（外部端子使用）	
最高サンプリング周波数	50ＫＨｚ（20μｓｅｃ間隔）	
分解能	サイン符号＋12 ビット	
A/D 変換出力コード	オフセットバイナリ形式	
A/D 変換方式	逐次比較方式	
変換絶対誤差	±0.1%以下	
トリガー	ソフトウェアトリガー ハードウェアトリガー（外部端子使用：入力信号 TTL レベル）	
I/O アドレス占有サイズ	8 バイト連続	
動作電圧	5Ｖ	
外形寸法	PCMCIA 2.1 TYPEII 準拠	

4-2. A/D カードレジスタ仕様

4-2-1. REX-5054U レジスタ仕様

OFFSET | **REX5054U レジスタ仕様**

BASE+0

A/D コンバータ IC アドレスレジスタ [WRITE]

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	A4	A3	A2	A1

A1-A4：LM12458 のレジスタアドレスをセットすることにより、A/D コンバータ IC データレジスタを通して選択した LM12458 のレジスタに入出力することができます。
注１）LM12458 は１６ビットバスモードで動作しています。
●LM12458 のレジスタレイアウト

アドレス				LM12458 レジスタ仕様
A4	A3	A2	A1	
0000-0111				**Instruction RAM (RAM Pointer = 00)**
0000-0111				**Instruction RAM (RAM Pointer = 01)**
0000-0111				**Instruction RAM (RAM Pointer = 10)**
1	0	0	0	Configuration Register
1	0	0	1	Interrupt Enable Register
1	0	1	0	Interrupt Status Register
1	0	1	1	Timer Register
1	1	0	0	Conversion FIFO
1	1	0	1	Limit Status Register

OFFSET	REX5054U　レジスタ仕様
BASE+1	サンプリングクロック設定レジスタ［WRITE］

b7	b6	b5	b4	b3	b2	b1	b0
CLKPLA	TRGPLA	0	DIVS1	DIVS0	0	CS1	CS0

CLKPLA：外部から TTL レベルのクロック信号を入力する場合に、パルスの立ち上がりエッジと立ち下がりエッジのどちらを有効にするか指定します。
０：立ち下がり　１：立ち上がり
になります。内部クロックを使用する場合は、意味を持ちません。

TRGPLA：外部から TTL レベルのトリガ信号を入力する場合に、パルスの立ち上がりエッジと立ち下がりエッジのどちらを有効にするか指定します。
０：立ち下がり　１：立ち上がり
になります。ソフトウェアトリガを使用する場合は、意味を持ちません。

DIVS1/S0：プログラマブルタイマカウンタの分周モードを設定します。

DVIS1	DVIS0	分周モード
0	0	カウンタ#０
0	1	カウンタ#０→カウンタ#１
1	0	ダイレクト
1	1	設定不可

CS1/CS0：ソースクロックモードを設定します。

CS1	CS0	クロックモード
0	0	内部１０ＭＨｚ
0	1	内部８．１９２ＭＨｚ
1	0	設定不可
1	1	外部クロック

OFFSET	REX5054U　レジスタ仕様
BASE+2	コントロールレジスタ [WRITE]

b7	b6	b5	b4	b3	b2	b1	b0
SETCLK	TRGSTR	TRGEND	INTEN	OVRCLR	0	CA1	CA0

SETCLK：LM12458 へ出力するクロック信号を制御します。クロック信号をスタートすることにより A/D 変換が開始します。
０：クロック停止　１：クロックスタート
注１）サンプリング終了時には必ずリセットして下さい。

TRGEND：外部トリガ停止モードを設定します。
０：外部トリガによる A/D 変換停止モード無効
１：外部トリガによる A/D 変換停止モード有効
注１)サンプリング終了時には必ずリセットして下さい。
注２)TRGSTR/TRGEND を両方セットした場合、最初のエッジで変換がスタートし次のエッジで終了します。

TRGSTR：外部トリガ開始モードを設定します。
０：外部トリガによる A/D 変換開始モード無効
１：外部トリガによる A/D 変換開始モード有効
注１)サンプリング終了時には必ずリセットして下さい。

INTEN： 割り込み要求モードを設定します。
０：割り込み要求を出力しません。
１：割り込み要求を出力します。
注１)割り込み要因は、LM12458 の内部レジスタで設定します。

OVRCLR：ステータスレジスタのビット０：オーバーランフラグをリセットします。
注１)FIFO オーバーランが発生したときは、１をセットしてステータスレジスタのオーバーランフラグをクリアして下さい。

CA1/CA0：プログラマブルタイマカウンタのアドレスをセットすることにより、[BASE+3]のタイマカウンタデータレジスタを通してタイマカウンタμＰＤ７１０５４のレジスタにリード・ライトできます。

CA1	CA0	μＰＤ７１０５４レジスタ
0	0	カウンタ#０
0	1	カウンタ#１
1	0	カウンタ＃２
1	1	コントロールワードレジスタ

<table>
<tr><th>OFFSET</th><th>REX5054U　レジスタ仕様</th></tr>
<tr><td>BASE+3</td><td>タイマカウンタデータレジスタ［READ/WRITE］
<table><tr><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th></tr><tr><td>CD7</td><td>CD6</td><td>CD5</td><td>CD4</td><td>CD3</td><td>CD2</td><td>CD1</td><td>CD0</td></tr></table>
CD7-CD0：プログラマブルタイマカウンタμＰＤ７１０５４のレジスタにデータをリード・ライトします。レジスタの指定はコントロールレジスタのビット１・ビット０で行います。
注１)ワードアクセスでアドレスとデータを同時に書き込んではいけません。</td></tr>
<tr><td>BASE+4</td><td>ステータスレジスタ［READ］
<table><tr><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th></tr><tr><td>DTRDY</td><td>ADBUSY</td><td>-</td><td>-</td><td>TRGSTR</td><td>TRGEND</td><td>-</td><td>OVRUN</td></tr></table>
DTRDY：データレディフラグ
このビットが１の時、変換が完了し LM12458 Conversion FIFO にデータがあることを示します。FIFO に何個あるかは、Interrupt Status Register から取得します。
ADBUSY：ＡＤビジーフラグ
このビットが１の時、A/D コンバータが変換動作中であることを示します。変換動作中は絶対に LM12458 Conversion FIFO にアクセスしないようにプログラム側で保証して下さい。ＡＤビジーが１の時は、０になるまで待って Conversion FIFO からデータを読み出します。
TRGSTR：トリガスタートフラグ
このビットが１の時、外部トリガによる A/D 変換開始待ち状態にあることを示します。TRGEND：トリガエンドフラグ
このビットが１の時、外部トリガによる A/D 変換終了待ち状態にあることを示します。
OVRUN：オーバーランフラグ
このビットが１の時、LM12458 の FIFO がオーバーフローしたことを示します。オーバーラン発生後の A/D 変換動作は保証されません。オーバーランが発生しないようにプログラム側で保証して下さい。</td></tr>
</table>

OFFSET	REX5054U　レジスタ仕様
BASE+5	RESERVED
BASE+6,7	A/D コンバータ IC データレジスタ [READ/WRITE] **b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0** **D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0** D15-D0：A/D コンバータ IC レジスタにリード･ライトするデータをセットします。このレジスタは必ずワ ードアクセスして下さい。A/D コンバータ IC のレジスタ選択は、A/D コンバータ IC アドレスレジスタ [BASE+0]で行います。

4-2-2. REX-5054B レジスタ仕様

REX5054B は REX505 4U のレジスタレイアウトに対して、**BASE+2** と **BASE+3** が反対になっていますでの注意して下さい。

OFFSET	REX5054B レジスタ仕様
BASE+0	A/D コンバータ IC アドレスレジスタ [WRITE]
BASE+1	サンプリングクロック設定レジスタ [WRITE]
BASE+2	**タイマカウンタデータレジスタ [READ/WRITE]**
BASE+3	**コントロールレジスタ [WRITE]**
BASE+4	ステータスレジスタ [READ]
BASE+5	RESERVED
BASE+6,7	A/D コンバータ IC データレジスタ [READ/WRITE]

4-3. LM12458 データアクイジョンシステム解説

本セクションでは、LM 12458 を使った基本的な A/D 変換プログラムを作成する上で必要となる部分を、ナショナルセミコンダクタ社のデータシートから抜粋して解説します。詳細はデータシートを参照願います。

4-3-1. LM12458 の機能

●12ビット(サイン符号付き)分解能で変換結果は2の補数形式で出力
●32ワード×16ビット FIFO 内蔵
●命令 RAM に8個までの実行命令を格納可能
●フルキャリブレーション機能内蔵
A/D カードまたはライブラリでは LM 12458 の持つ以下の機能は使いません。
◯ウォッチドッグモード
◯LM12458 内蔵タイマー
◯DMA転送機能

4-3-2. 内部ユーザプログラマブルレジスタ

(1) 命令 RAM (Instruction RAM)

命令 RAM は、順次実行される命令を最大で8つまで保持できます。48ビット長の命令は、16ビットの3セクションに分割されます。命令アドレスと構成レジスタの2ビット「RAM Pointer」により、各16ビットのセクションに READ 命令と WRITE 命令を送ることができます。0000～0111の命令アドレスに8つの命令が格納され、ランダムにアクセスしプログラムすることが可能です。

命令 RAM の READ や WRITE などの命令は、シーケンサの動作に影響を与えることがあります。命令 RAM の READ か WRITE 命令を実行する前に、実行中の命令が終了するのを待って、RESET ビットを1にセットして、シーケンサの実行を停止して下さい。

ソフト RESET は、命令 RAM への READ か WRITE 命令の後で、構成レジスタの RESET ビットを1にセットして行います。

命令 RAM の3セクションは、構成レジスタの2ビット「RAM Pointer」(ビット D8,D9) によって選択します。最初の16ビットの命令 RAM セクションは「RAM Pointer:00」を指定して選択します。このセクションは、マルチプレクサ・チャンネルの選択、分解能、アクイジション時間などに関係します。2番目の16ビットセクションは、ウォッチドッグのリミット#1とその符号、および入力信号がプログラムされたリミット値を上回るとき、または下回るときの割り込み発生を示すインジケータを保持します。3番目の16ビットセクションは、ウォッチドッグのリミット#2とその符号、および入力信号がプログラムされたリミット値を上回るとき、または下回るときの割り込み発生を示すインジケータを保持します。

●レジスタレイアウト

ｱﾄﾞﾚｽ A4-A0	Type	レジスタ内容 D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0000 ～ 0111	R/W	命令 RAM：Instruction RAM(RAM Pointer = 00)															
		Acquisition Time				Watch Dog	8/12	Timer	Sync	Vin- (MIXOUT-)			Vin+ (MIXOUT+)			Pause	Loop
0000 ～ 0111	R/W	命令 RAM：Instruction RAM(RAM Pointer = 01)															
		Don't Care						>/<	Sign	Limit #1							
0000 ～ 0111	R/W	命令 RAM：Instruction RAM(RAM Pointer = 10)															
		Don't Care						>/<	Sign	Limit #2							
1000	R/W	構成レジスタ:Configuration Register															
		Don't Care				Diag	Test = 0	Ram Pointer		I/O Set	Auto Zero	Chan Mask	Stand -by	Full CAL	Auto Zero	Reset	Start
1001	R/W	割り込みイネーブルレジスタ：Interrupt Enable Register															
		Number of Conversions in Conversion FIFO In Generate INT2					Sequencer Address to Generate INT1			INT 7	INT 6	INT 5	INT 4	INT 3	INT 2	INT 1	INT 0
1010	R	割り込みステータスレジスタ：Interrupt Status Register															
		Actual Number of Conversion Results In Conversion FIFO					Address of Seqencer Instruction being Excluded			INT 7	INT 6	INT 5	INT 4	INT 3	INT 2	INT 1	INT 0
1011	R/W	タイマレジスタ：Timer Register															
		Timer Preset High Byte								Timer Preset Low Byte							
1100	R	FIFO レジスタ：Conversion FIFO															
		Address of Sign			sign	Convention Data: MSB's				Convention Data: LSB's							
1101	R	リミットステータスレジスタ：Limit Status Register															
		Limit #2:Status								Limit #1:Status							

命令 RAM：00

ビット	内容
BIT0	LOOP ビットです。このビットが１にセットされているときは、命令シーケンスで実行される最後の命令を示します。実行される次の命令は、命令０になります。
BIT1	PAUSE ビットです。このビットは、シーケンサの動作を制御します。PAUSE ビットが１にセットされている時は、シーケンサはその命令を読み込んだ後、その命令を実行する前に停止します。構成レジスタの START ビットは、自動的に０にリセットされます。PAUSE をセットすると割り込みが発生します。構成レジスタのビット０（START ビット）を１にセットすると、シーケンサがリスタートします。 命令 RAM がプログラムされ、RESET ビットが１にセットされた後、シーケンサは命令 000 を読み込み、解読して、構成レジスタの START ビットが１になるのを待ちます。 START ビットが０であると、シーケンサがスタートした時の命令 000 の PAUSE ビットの値は無効です。一旦スタートすると、シーケンサは命令 000 を実行し、残っている命令を読み込み、解読して実行します。PAUSE ビットが１にセットされた命令 000 をシーケンサが最初に実行しても PAUSE 割り込み（INT5）は発生しません。シーケンサが LOOP ビットを検出するか、全ての８命令が終了した時、再び命令 000 が読み込まれ解読されますが、命令 000 の PAUSE ビットはこの時には命令を実行する前にシーケンサを停止させます。
BIT2-4	８つの入力チャンネル（IN0～IN7 の場合は"000"～"111"）のうち、どれを LM12458 の A/D コンバータへの非反転入力として構成するかを選択するためのビットです。また、４つの入力チャンネル（IN0～IN3 の場合は"000"～"011"）のうち、どれを LM12458 の A/D コンバータへの非反転入力として構成するかを選択するためにも使用されます。
BIT5-7	７つの入力チャンネル（IN1～IN7 の場合は"001"～"111"）のうち、どれを LM12458 の A/D コンバータへの反転入力として構成するかを選択するためのビットです。また、３つの入力チャンネル（IN1～IN3 の場合は"001"～"011"）のうち、どれを LM12458 の A/D コンバータへの非反転入力として構成するかを選択するためにも使用されます。２つのマルチプレクサチャンネルを選択することにより、完全差動入力動作を行うことができます。一方のチャンネルは非反転モードで動作し、もう一方のチャンネルは反転モードで動作します。コード"000"は、シングルエンド動作の反転入力としてグランドを選択することになります。

Channel Selection Data	Normal Mode	
	VIN+	VIN-
000	IN0	GND
001	IN1	IN1
010	IN2	IN2
011	IN3	IN3
100	IN4	IN4
101	IN5	IN5
110	IN6	IN6
111	IN7	IN7

ビット	内容
BIT8	SYNC ビットです。BIT8 を１にセットすると、シーケンサは内部 S/H のアクイションサイクルの終わりで動作を中断し、SYNC 端子の立ち上がりエッジを待ちます。立ち上がりエッジが現れると、内部 S/H が入力信号の振幅を取り込み、A/D コンバータがクロックの次の立ち上がりでエッジで変換を開始します。SYNC 端子を入力端子として使用する場合は、構成レジスタの I/OSel ビット（ビット７）を０にセットします。SYNC 端子を入力端子として構成すると、変換の開始を外部イベントに同期させることができます。この機能は、正確な変換タイミングが必要なデジタル信号処理などのアプリケーションに有用です。
BIT9	タイマビットです。BIT9 が１にセットされると、シーケンサは内部の１６ビットタイマがゼロにカウントダウンするまで停止します。この停止期間中は、ウォッチドッグによる比較や A/D 変換は行われません。
BIT10	A/D コンバータの分解能を選択するためのビットです。BIT10 を１にセットすると８ビット＋サインモードが選択され、０をセットすると１２ビット＋サインモードが選択されます。
BIT11	ウォッチドッグ比較モードのイネーブルビットです。BIT11 を１にセットすると、選択されたアナログ入力信号が格納されている２つのリミット値と比較されます。０にセットすると、命令 RAM"00"の BIT10 の状態に応じて入力信号の８ビット＋サインまたは１２ビットサインモードの変換が開始します。
BIT12-15	ユーザプログラマブルなアクイジョン時間の値を格納します。シーケンサは、ビット１２～１５の設定値の２倍に相当する可変クロックサイクル数を加えた一定のクロックサイクル期間中（１２ビット＋サイン変換では９クロックサイクル・８ビット＋サイン変換やウォッチドッグ変換では２クロックサイクル）、内部 S/H をアクイジョンモードにします。従って、S/H のアクイジョン時間は、１２ビット＋サイン変換では（9+2D）クロックサイクルになり、８ビット＋サイン変換やウォッチドッグ変換では（2+2D）クロックサイクルになります。（D）はビット１２～１５の設定値を示します。

命令 RAM：01

2番目の命令 RAM セクションは、構成レジスタのビット8と9を"01"にセットして選択します。

ビット	内容
BIT0-7	ウォッチドッグリミット値＃１を保持します。命令 RAM"00"のビット１１が１にセットされると、サンプリングしたアナログ入力信号をウォッチドッグモードでリミット値＃１の値と比較し、次にこのアナログ入力信号をリミット値＃２の値と比較します。
BIT8	リミット値＃１のサインを保持します。
BIT9	ウォッチドッグモードの割り込みを発生するリミット条件が決まります。１の状態では電圧リミット＃１を超えると割り込みが発生し、０では電圧がリミット値＃１以下になると割り込みが発生します。
BIT10-15	使用しません。

命令 RAM：10

3 番目の命令 RAM セクションは、構成レジスタのビット 8 と 9 を" 10"にセットして選択します。

ビット	内容
BIT0-7	ウォッチドッグリミット値＃２を保持します。命令 RAM"00"のビット１１が１にセットされると、サンプリングしたアナログ入力信号をウォッチドッグモードでリミット値＃１の値と比較し、次にこのアナログ入力信号をリミット値＃２の値と比較します。
BIT8	リミット値＃２のサインを保持します。
BIT9	ウォッチドッグモードの割り込みを発生するリミット条件が決まります。１の状態では電圧リミット＃２を超えると割り込みが発生し、０では電圧がリミット値＃２以下になると割り込みが発生します。
BIT10-15	使用しません。

構成レジスタ

構成レジスタ１０００は、リード･ライト機能を持った１６ビット制御レジスタです。このレジスタは、シーケンサのスタート･ストップ、リセット、セルフキャリブレーション、スタンバイ命令などのグローバルな情報を保持しLM12458の制御パネルとして動作します。

ビット	内容
BIT0	シーケンサのスタート・ストップビットです。ビット０をリードするとシーケンサのステータスを表示します。０はシーケンサが停止して、次の命令の実行を待っていることを示します。１はシーケンサが動作していることを示します。０をセットすると、実行中の命令が終了した時にシーケンサが停止します。次に実行される命令は、ステータスレジスタにある命令ポインタによって指示されます。１をセットすると、命令ポインタによって現在指示されている命令により、シーケンサがリスタートします（割り込みステータスレジスタビット８～１０の項を参照）。
BIT1	LM12458のシステムリセットビットです。ビット１に１をライトすると、シーケンサが停止し（構成レジスタのスタート・ストップビットをリセット）、命令ポインタが"000"に戻り（ステータスレジスタで示される）、変換結果格納FIFOがクリアされ、全ての割り込みフラグがリセットされます。このリセットビットは、構成レジスタのスタンバイビットに１を書き込み強制的にHIGHにしない限り、２クロックサイクル後に０に戻ります。リセット信号は、最初の電源投入時に内部生成されます。リセットビットが０になるまで、いかなる動作も開始しません。
BIT2	１をセットすると、オートゼロ・オフセット電圧キャリブレーションが開始します。ビット２は、フル・キャリブレーションプロセスで実行される８サンプルのオートゼロ・キャリブレーションとは異なり、オフセットを一度サンプリングして補正率を求めることにより、短いオートゼロを開始します（フル・キャリブレーションの場合は、補正率を求めるときにコンバータのオフセット電圧のサンプル８個から平均を計算する）。ビット２が１にセットされてシーケンサが動作している場合、現在実行中の命令が終了すると、ただちにオートゼロが開始します。ビット２は、オートゼロ（７６クロックサイクル）が終了すると、自動的に０にリセットされ、割り込みフラグ（割り込みステータスレジスタのビット３）がセットされます。シーケンサは、オートゼロ・キャリブレーションが終了すると、命令RAMポインタが示している次の命令をフェッチし、実行を再開します。シーケンサが停止している場合には、オートゼロが要求された時点ですぐに実行されます。

ビット	内容
BIT3	１をセットすると、長いオートゼロ・オフセット電圧の補正を含む完全なキャリブレーションプロセス（このキャリブレーションでは、補正率を求めるときにコンパレータ・オフセット電圧の８個のサンプルから平均値を計算）を実行し、次に A/D コンバータの直線性キャリブレーションを実行します。この完全なキャリブレーションは、ビット３が１にセットされてシーケンサが動作している場合に、現在実行中の命令が終了してから開始されます。このキャリブレーション・プロセス（4944 クロックサイクル）が終了した時点で、ビット３は自動的に０にリセットされ、割り込みフラグ（割り込みステータスレジスタのビット４）が生成されます。 シーケンサは、フル・オートゼロと直線性キャリブレーションが終了すると、命令 RAM ポインタが示している次の命令をフェッチし、実行を再開します。フルキャリブレーションは、シーケンサが停止している場合には、要求された時点ですぐに実行されます。
BIT4	スタンバイビットです。ビット４を１にセットすると、LM12458 はただちにスタンバイモードになります。ビット４を０にリセットすると、通常モードに戻ります。スタンバイコマンド１を実行すると、外部クロックを内部回路から切り離し、LM12458 の内部アナログ回路の電源電流を下げ、内部 RAM の全てのデータを保存します。スタンバイビットを０にセットすると、LM12458 は、リセットビットをセットした時と同じ動作状態になります。パワーアップ完了後にスタンバイ完了割り込みが出力され、これによってアナログ回路が安定します。スタンバイ完了割り込みがかかってからシーケンサがリスタートするようにして下さい。LM12458 がスタンバイモードの間も、リード・ライト動作によって命令 RAM にアクセスできます。
BIT5	チャンネルアドレスマスクです。ビット５を１にセットすると、変換結果格納 FIFO レジスタのビット１３～１５は、変換データのサインビット（ビット１２）と等しくなります。ビット５を０にリセットすると、変換データのビット１３～１５は、変換データが属している命令の命令ポインタ値を保持します。
BIT6	各変換毎の短いオートゼロ補正を選択する際に使用します。ビット６が１にセットされている場合は、シーケンサは各変換またはウォッチドッグモードで比較を行う前に、自動的にオートゼロを挿入します。ビット６が０にセットされていると、いかなる自動補正も実行されません。 LM12458 のキャリブレーション後のオフセット電圧ドリフトは、－４０℃～＋８０℃の温度範囲で、０.１LSB（TYP）になります。この小さいドリフト値は、各変換動作にオートゼロを使用した時に生じるオフセット値の変動幅より小さくなっています。これは、補正値を求めるためにオフセット電圧を一度サンプリングしたことによるものです。フルキャリブレーションモードでは変動幅が小さくなりますが、これは、補正値を得るためにオフセット電圧を８回サンプリングし、その平均値を計算するのに使用したためです。

ビット	内容
BIT7	SYNC 端子を、入力端子または出力端子のどちらかで動作させるようにプログラムするときに使用します。この SYNC 端子は、ビット７を１にすると出力端子になり、０にすると入力端子になります。SYNC 端子を入力端子としてプログラムすると、ピン２９に加わる論理信号の立ち上がりエッジで、変換またはウォッチドッグモードによる比較が開始します。出力ピンとしてプログラムされると、変換またはウォッチドッグモードによる比較の開始時点でピン２９が H レベルになり、どちらかの動作が終了するまでその状態を維持します。命令 RAM"00"のビット８の項を参照して下さい。
BIT8-9	４８ビット命令 RAM の各１６ビットの３セクションを、リード・ライトの動作中に選択するために使用する RAM ポインタを形成します。"00"にすると命令 RAM の第一セクションが選択され、"01"にすると第２セクションが選択され、"10"にすると第３セクションが選択されます。
BIT10	製造時の試験でテストモードを起動するためのビットです。このビットは常に０をセットしておきます。
BIT11	診断用のビットです。このビットは１にするとアクティブになります（テストビットは０にしておきます）。診断モードで命令が正しく選択されていれば、LM12458 の A/D コンバータが正しく動作しているか検証することができます。

割り込みレジスタ

LM12458 には同一優先順位の 8 個の割り込みがあります。これらの割り込みのどれが発生しても、割り込みレジスタによってマスクされていない限り、ハードウェア割り込みが INT 端子に現れます。次にステータスレジスタが読み込まれ、8 つの割り込みのどれが発生したかが判別されます。

割り込みイネーブルレジスタへのライト後、割り込みステータスレジスタ (A4-A1 : 1010) を読み出しクリアしなければなりません。これにより、割り込みイネーブルレジスタのアクセス中に INT 端子に発生するスプリアス割り込みを取り除くことができます。

割込み番号	内容
Interrupt 0	LM12458 がウォッチドッグ比較モード動作中、選択されたマルチプレクサチャンネルのアナログ入力電圧がリミット値を超えるたびに発生します。
Interrupt 1	シーケンサが、割り込みレジスタのビット 8～10 で指定の命令カウント値に達すると発生します。このフラグは、命令の実行前にセットされます。
Interrupt 2	変換結果格納 FIFO レジスタ内の変換回数の数値が、割り込みイネーブルレジスタのビット 11～15 にセットされているプログラム値に等しくなると発生します。
Interrupt 3	短い 1 回のサンプリングでオートゼロキャリブレーションが終了すると発生します。
Interrupt 4	フルオートゼロと直線性セルフキャリブレーションが終了すると発生します。
Interrupt 5	シーケンサが、1 にセットされたポーズビット (命令 RAM"00"のビット 1) を含む命令を検出すると発生します。
Interrupt 6	4 V 以下の電源電圧を検出すると発生します。この割り込みは、LM12458 によって返されたデータが破壊されている可能性があることを示します。
Interrupt 7	LM12458 が、構成レジスタのビット 4 をセットしてスタンバイモードからアクティブ動作に復帰してから 10 ms に発生します。この短い遅延の間に、内部のアナログ回路の動作が十分に安定して高精度の変換結果が得られます。

割り込みイネーブルレジスタ

アドレス１００１にある割り込みイネーブルレジスタは、リード･ライトの機能を持っています。個々の外部割り込みは、該当するビットアドレスを1にセットして行います。割り込みイネーブルレジスタのビット状態にかかわらず、内部割り込みが発生すると、対応する割り込みステータスレジスタ内のビットが1にセットされます。

ビット	内容
BIT0	内部ウォッチドッグ比較リミットの割り込みが発生すると、外部割り込みをイネーブルにします。
BIT1	シーケンサが、割り込みイネーブルレジスタのビット８～１０に格納されているアドレスに達すると、外部割り込みをイネーブルにします。
BIT2	割り込みイネーブルレジスタのビット１１～１５に格納されている変換 FIFO レジスタのリミット値に達すると、外部割り込みをイネーブルにします。
BIT3	１回のサンプリングのオートゼロキャリブレーションが終了すると、外部割り込みをイネーブルにします。
BIT4	フルオートゼロと直線性セルフキャリブレーションが終了すると、外部割り込みをイネーブルにします。
BIT5	ポーズ割り込みが発生すると、外部割り込みをイネーブルにします。
BIT6	低電圧条件（４Ｖ以下）によって割り込みが発生すると、外部割り込みをイネーブルにします。
BIT7	パワーダウンからアクティブモードに復帰すると、外部割り込みをイネーブルにします。
BIT8-10	ユーザによるプログラム値を格納します。この値は、シーケンサのアドレスと比較されます。シーケンサが、ビット８から１０の格納値に等しいアドレスに達すると、内部割り込みが発生して割り込みステータスレジスタのビット１がセットされます。割り込みイネーブルレジスタのビット１が１にセットされると、外部割り込みをイネーブルにします。 ビット８～１０の格納値０００～１１１は、命令 RAM 内の０～７の命令を表します。命令 RAM のプログラミングを行い、リセットビットを１にセットした後、構成レジスタのスタートビットを１にセットするとシーケンサが開始します。INT1 のトリガ値の００００にセットしても、シーケンサによる１回目の命令０００のフェッチとデコーディングだけでは INT1 は発生しません。シーケンサは、命令０００を検出した後で２回目に（割り込みステータスレジスタのビット１を１にセットして）INT1 を発生させます。内部または外部で命令の割り込み（INT1）が発生しても、シーケンサは動作し続けるので注意して下さい。シーケンサの停止は、ポーズビットを１にセットする（命令の実行前に停止する）か、構成レジスタのスタートビットを０にするか、構成レジスタのリセットビットを１にする場合のみです。
BIT11-15	内部割り込みを発生させるために、変換結果格納 FIFO レジスタに格納すべき変換数を保持します。内部割り込みが発生すると、割り込みステータスレジスタのビット２がセットされます。割り込みイネーブルレジスタのビット２が１にセットされると、外部割り込みが出力されます。

割り込みステータスレジスタ

このリード専用レジスタは、アドレス１０１０に設定されています。割り込みイネーブルレジスタで割り込みがイネーブルされていなくても、割り込みが発生すると対応する割り込み状態フラグが常に HIGH ("1") になります。このレジスタが読み出されるか、あるいはデバイスリセットが送出されると、アクティブ (" 1") になっている割り込みステータスレジスタのフラグは全て"0"にリセットされます。

ビット	内容
BIT0	ウォッチドッグ比較リミットの割り込みが発生すると、"1"にセットされます。
BIT1	シーケンサが、割り込みイネーブルレジスタのビット８～１０に格納されているアドレスに達すると、"1"にセットされます。
BIT2	割り込みイネーブルレジスタのビット１１～１５に格納されている変換 FIFO レジスタのリミット値に達すると、"1"にセットされます。
BIT3	１回のサンプリングのオートゼロキャリブレーションが終了すると、"1"にセットされます。
BIT4	フルオートゼロと直線性セルフキャリブレーションが終了すると、"1"にセットされます。
BIT5	ポーズ割り込みが発生すると、"1"にセットされます。
BIT6	低電圧条件（４Ｖ以下）が成立すると、"1"にセットされます。
BIT7	パワーダウンからアクティブモードに復帰すると、"1"にセットされます。
BIT8-10	シーケンサが動作中に、現行アドレスを保持します。
BIT11-15	シーケンサが動作中に、変換 FIFO レジスタに格納されている現行変換数を保持します。

リミットステータスレジスタ

このリード専用レジスタは、アドレス１１０１に設定されています。このレジスタは、命令 RAM のリミット＃１およびリミット＃２のレジスタとともに使用されます。入力電圧が対応するリミットレジスタにセットされているリミット値を超えると、リミットステータスレジスタの命令番号に対応するビットがセットされます。このレジスタが読み出されるか、デバイスリセットが送出されると、アクティブ ("1") になっているリミットステータスレジスタのフラグは全て"0"にリセットされます。

ビット	内容
BIT0-7	**リミット＃１レジスタの状態を示します。**
BIT8-15	**リミット＃２レジスタの状態を示します。**

タイマレジスタ

LM12458 は、5ビットのプリスケーラを含む１６ビットタイマを内蔵しています。このタイマはピン２３に入力されるクロックを使用し、2^5 刻みで 0～2^{21} クロックサイクルまでのタイムインターバルを発生することができます。このタイムインターバルは、命令の実行を遅らせたり、変換レートを遅くするためにも使用でき、この結果コントローラによって読み出される FIFO の冗長データ量を減らすことができます。

タイマで使用するユーザ定義のタイミング値は、１６ビットのリード･ライトタイマレジスタのアドレス１０１１に格納され、自動的にプリロードされます。ビット0～7はプリセット値の下位バイトを、ビット8～１５はプリセット値の上位バイトを保持します。タイマは、現行命令のビット9が1にセットされている場合に限り、シーケンサによって起動できます。

FIFO レジスタ

各変換結果は、内部のリード専用 FIFO レジスタに格納されます。FIFO レジスタは、アドレス１１００に設定されています。このレジスタには、１６ビット幅のアドレスロケーションが３２あり、各ロケーションには１３ビットのデータが保持できます。ビット０～３は、１２ビット＋サインモードの４LSB を保持します。ビット４～１１は８MSB を保持し、ビット１２はサインビットを保持します。ビット１３～１５は、レジスタの２の補数のデータ形式をフルサイズの１６ビットに拡張したサインビットか、または変換とその結果のデータを生成した命令アドレスのいずれかを保持します。これらのモードは、構成レジスタのビット５の論理状態に応じて選択されます。

FIFO に格納されている変換結果の数を知るために、変換結果を読み出す前に FIFO の状態を割り込みステータスレジスタ（ビット１１～１５）から読み出す必要があります。これにより、リード数が FIFO 内の変換結果の数より大きい場合に生じる不正な変換データを防止することができます。FIFO が空の時に FIFO リードを行うと、FIFO に書き込み中の新しいデータが更新されることがあります。A/D コンバータを用いて FIFO に変換データを３２件以上書き込むと、最初の変換データが失われます。従って、データの欠落を防止するために、LM12458 の割り込み機能を使用して、システムのコントローラに FIFO が満杯であることを通知する必要があります。

ビット	内容
BIT0-12	**１２ビット＋サインの変換データを保持します。８ビット＋サイン分解能使用時には、ビット０～３が１１１０（LSB）になります。**
BIT13-15	**変換データに関する命令か、またはサインビットのどちらかを保持します。どちらのモードにするかは、構成レジスタのビット５で選択します。** **オーバーフローなしに FIFO の全アドレスロケーションを使用するためには、割り込みイネーブルレジスタのビット１１～１５の値を１１１１に、ビット２を１にセットすることが必要です。これにより、３１回目の変換データが FIFO に取り込まれると外部割り込みが発生します。ホストプロセッサーは、この割り込みを使用して LM12458 の構成レジスタのスタートビットを０にセットし、３２回目の変換終了前に A/D コンバータに動作停止命令を送ります。シーケンサは現行（３２回目）の変換が終了した後で停止します。FIFO の３２番目のアドレスロケーションに変換データが取り込まれ、格納されます。FIFO のオーバーフローを防止するには、３２回目の変換開始前に、構成レジスタのスタートビットを０にセットしてシーケンサを停止することが必要です。内部または外部で FIFO の割り込み（INT2）が発生しても、シーケンサは動作し続けるので注意して下さい。シーケンサの停止は、ポーズビット１にセットする（命令の実行前に停止）か、あるいは構成レジスタのリセットビットを１にセットします。**

4-4. プログラマブルタイマカウンタ解説

REX5054U/B はプログラマブルタイマカウンタμPD71054相当品を搭載しています。タイマカウンタにはカウンタ＃0・カウンタ＃1・カウンタ＃2・コントロールワードの4つのレジスタがあり、コントロールレジスタの CA0/CA1 にアドレスをセットすることにより、タイマカウンタデータレジスタを通してリード・ライトすることができます。

カウンタ分周モードとして、クロックを分周しないでダイレクトに A/D 変換の信号として入力するモード、カウンタ＃0で分周した信号を A/D 変換の信号として入力するモード、カウンタ＃0をカウンタ＃1のプリスケーラとして使用し A/D 変換の信号として入力するモードをサポートしています。サンプリングクロック設定レジスタの DIVS0/DIVS1 でどのモードに設定するか選択して下さい。

カウンタ＃2は ADBUSY 信号発生のために A/D カード内部で使用しています。必ず、4-3-2.記載の設定を行って下さい。

4-4-1. タイマカウンタの設定方法

タイマカウンタμPD71054は6つの動作モードを持っています。タイマカウンタを動作させるには、動作モードを選択してからカウント値を書き込むという設定が必要になります。以下に、カウンタ＃0・＃1をモード2のレートジェネレータに設定し、カウンタ＃0をカウンタ＃1のプリスケーラとする場合のコーディング例を示します。

```
mov     dx,ADC_CONTROL
mov     al,3
out     dx,al

mov     dx,ADC_COUNTER
mov     al,34h
out     dx,al
mov     al,74h
out     dx,al

mov     dx,ADC_CONTROL
mov     al,0
out     dx,al

mov     dx,ADC_COUNTER
mov     ax,DIV0_VAL
out     dx,al
mov     al,ah
out     dx,al
```

コントロールレジスタに3をライトし、タイマカウンタデータレジスタをμPD71054のコントロールワードレジスタにマッピング

カウンタ＃0を動作モード2で初期化

カウンタ＃1を動作モード2で初期化

タイマカウンタデータレジスタをμPD71054のカウンタ＃0にマッピング

カウンタ＃0にカウンタ値 DIV0_VAL を下位バイト・上位バイトの順で設定

```
mov     dx,ADC_CONTROL
mov     al,1
out     dx,al

mov     dx,ADC_COUNTER
mov     ax,DIV1_VAL
out     dx,al
mov     al,ah
out     dx,al
```

タイマカウンタデータレジスタをμＰＤ７１０５４のカウンタ＃１にマッピング

カウンタ＃１にカウンタ値DIV1_VALを下位バイト・上位バイトの順で設定

4-4-2. カウンタ＃２の設定

カウンタ＃２はリトリガラブルワンショットモードで次の式で計算した値を設定して下さい。

●設定値＝４４×チャンネル数＋１１×（チャンネル数－１）

コーディング例を以下に示します。

```
mov     dx,ADC_CONTROL
mov     al,3
out     dx,al

mov     dx,ADC_COUNTER
mov     al,b2h
out     dx,al

mov     dx,ADC_CONTROL
mov     al,2
out     dx,al

mov     dx,ADC_COUNTER
mov     ax,CHANNEL_NUM
mov     ah,55
mul     ah
sub     ax,11

out     dx,al
mov     al,ah
out     dx,al
```

コントロールレジスタに3をライトし、タイマカウンタデータレジスタをμＰＤ７１０５４のコントロールワードレジスタにマッピング

カウンタ＃2 を動作モード１で初期化

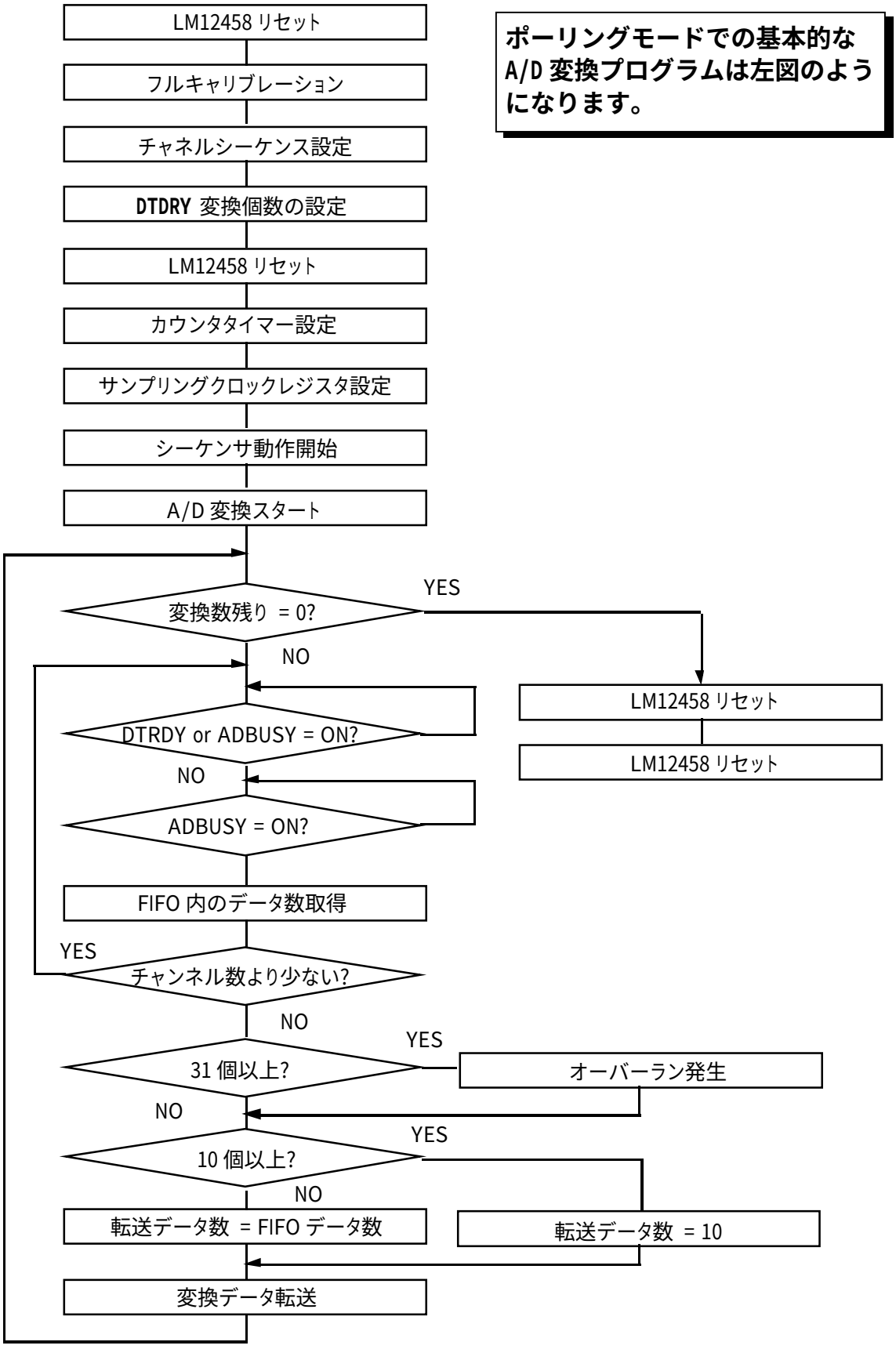
タイマカウンタデータレジスタをμＰＤ７１０５４のカウンタ＃２にマッピング

上記の計算値を求める

カウンタ＃２に計算値を下位バイト・上位バイトの順で設定

4-5. ポーリングモードでのプログラム

ポーリングモードでの基本的な A/D 変換プログラムは左図のようになります。



フローチャートに従って各手続きをプログラム内容について説明します。REX5054U A/D カードを I/O アドレス 300h から割り付けた場合、レジスタマップは下記のようになります。

```
ADC_ADDRESS   equ   300h
ADC_CLOCK     equ   301h
ADC_CONTROL   equ   302h
ADC_COUNTER   equ   303h
ADC_STATUS    equ   304h
ADC_DATA      equ   306h
LM_INST_RAM0 equ   00h
LM_INST_RAM1 equ   01h
LM_INST_RAM2 equ   02h
LM_INST_RAM3 equ   03h
LM_INST_RAM4 equ   04h
LM_INST_RAM5 equ   05h
LM_INST_RAM6 equ   06h
LM_INST_RAM7 equ   07h
LM_CONFIG     equ   08h
LM_INTENABLE equ   09h
LM_INTSTAT    equ   0ah
LM_TIMER      equ   0bh
LM_CONVFIFO   equ   0ch
LM_LIMITSTAT equ   0dh
```

A/D コンバータ IC レジスタ	BASE +0	**W**
サンプリングクロック設定レジスタ	BASE +1	**W**
コントロールレジスタ	BASE +2	**W**
タイマカウンタデータレジスタ	BASE +3	**R/W**
ステータスレジスタ	BASE +4	**R**
A/D コンバータ IC データレジスタ	BASE +6	**R/W**
LM12458 命令 RAM0	A1-A4	0000
LM12458 命令 RAM1	A1-A4	0001
LM12458 命令 RAM2	A1-A4	**0010**
LM12458 命令 RAM3	A1-A4	**0011**
LM12458 命令 RAM4	A1-A4	**0100**
LM12458 命令 RAM5	A1-A4	**0101**
LM12458 命令 RAM6	A1-A4	**0110**
LM12458 命令 RAM7	A1-A4	**0111**
LM12458 構成レジスタ	A1-A4	**1000**
LM12458 割り込みイネーブルレジスタ	A1-A4	**1001**
LM12458 割り込みステータスレジスタ	A1-A4	**1010**
LM12458 タイマレジスタ	A1-A4	**1011**
LM12458 FIFO レジスタ	A1-A4	**1100**
LM12458 リミットステータスレジスタ	A1-A4	**1101**

(1) LM12458 リセット手順

```
mov     dx,ADC_ADDRESS
mov     al,LM_CONFIG_REG
out     dx,al

mov     dx,ADC_DATA
mov     ax,00000010b
out     dx,al

mov     cx,50000
@@:
in      ax,dx
or      ax,ax
jz      jmp     short @f
loop    short @b
; Reset Error
@@:
; Normal End
```

A/D コンバータ IC アドレスに LM12458 構成レジスタのアドレスをセットし、A/D コンバータ IC データレジスタを LM12458 構成レジスタに設定します。

LM12458 構成レジスタのリセットビットを立てます。

LM12458 構成レジスタのリセットビットがオフになり、リセットが完了するのを待ちます。

(2) フルキャリブレーション

```
mov     dx,ADC_ADDRESS
mov     al,LM_CONFIG_REG
out     dx,al

mov     dx,ADC_DATA
mov     ax,00001000b
out     dx,al

mov     cx,50000
@@:
in      ax,dx
or      ax,ax
jz      jmp     short @f
loop    short @b
; Calibration Error
@@:
; Calibration End
```

A/D コンバータ IC アドレスに LM12458 構成レジスタのアドレスをセットし、A/D コンバータ IC データレジスタを LM12458 構成レジスタに設定します。

LM12458 構成レジスタのフルキャリブレーションビットを立てます。

LM12458 構成レジスタのフルキャリブレーションビットがオフになり、フキャリブレーションが完了するのを待ちます。

(3) チャンネルシーケンス設定

チャンネル1とチャンネル2の2チャンネルについて変換を実行するときのシーケンス設定例を下記に示します。チャンネル1は、S&H アクイジョンサイクルの終わりで動作を中断し SYNC 端子の立ち上がりエッジをを待ち、エッジを検出したら入力信号を取り込み、A/D コンバータが次のクロックエッジで変換を開始するよに設定します。チャンネル2以降は、S&H と A/D 変換を直ちに実行するように設定します。また、命令シーケンスのターミネーションをチャンネル2に設定します。

```
mov dx,ADC_ADDRESS
mov al,LM_CONFIG_REG
out dx,al

mov dx,ADC_DATA
mov ax,0h
out dx,ax

mov dx,ADC_ADDRESS
mov al,LM_INST_RAM0
out dx,al

mov dx,ADC_DATA
mov ax,00000001 00000000b
out dx,ax
```

LM12458 構成レジスタを選択します。

LM12458 命令 RAM00 (RAM Pointer 00) を選択します。

命令 RAM00 のアドレス"0000"を選択します。

アドレス"0000"にチャンネル1のシーケンス命令を書き込みます。

```
mov     dx,ADC_ADDRESS
mov     al,LM_CONFIG_REG
out     dx,al

mov     dx,ADC_DATA
mov     ax,0h
out     dx,ax

mov     dx,ADC_ADDRESS
mov     al,LM_INST_RAM1
out     dx,al

mov     dx,ADC_DATA
mov     ax,00000000
00000101b
out     dx,ax
```

LM12458 構成レジスタを選択します。

LM12458 命令 RAM00（RAM Pointer 00）を選択します。

命令 RAM00 のアドレス"0001"を選択します。

アドレス"0001"にチャンネル 2 のシーケンス命令を書き込みます。VIN+に<001>を入力し、LOOP ビットに<1>命令シーケンスの終了をセットします。

（4）DTRDY 変換個数の設定

FIFO レジスタに何個データがたまったら DTRDY を発生させるか割り込みイネーブルレジスタに設定します。通常はサンプリングチャンネル数に等しい数を設定しますが、FIFO を有効に使用するにはある程度デ ータ数（8 個程度）をまとめて転送するようにプログラムします。

```
mov     dx,ADC_ADDRESS
mov     al,LM_INTENABLE
out     dx,al

mov     dx,ADC_DATA
mov     ax,2
mov     cl,11
shl     ax,cl
or      ax,4h
out     dx,ax
```

LM12458 割り込みイネーブルレジスタを選択します。

D15-D11 に DTRDY を発生させる FIFO 内のデータ数を設定します（左の例では 2 個に設定しています）。

（5）カウンタタイマー設定

4-4.プログラマブルタイマカウンタの解説を参照して下さい。

(6) サンプリングクロックレジスタ設定

１０ＭＨｚの内部クロックを選択し、分周回路としてカウンタ＃０と＃１の両方を使うように設定する場合の例を示します。

```
mov     dx,ADC_CLOCK
mov     al,00001000b
out     dx,al
```

(7) LM12458 のシーケンサ動作開始

```
mov     dx,ADC_ADDRESS
mov     al,LM_CONFIG_REG
out     dx,al

mov     dx,ADC_DATA
mov     ax,1h
out     dx,ax
```

LM12458 構成レジスタを選択します。

構成レジスタのスタートビットをセットします。

(8) A/D 変換スタート

コントロールレジスタにビット7に1をセットすることにより A/D コンバータへのクロック供給が開始し、A/D 変換がスタートします。ポーリングモードでのサンプリングの場合は、割り込み要求ビット4を立てません。また、オーバーランフラグはリセットしておきます。

```
mov     dx,ADC_CONTROL
mov     al,10001000b
out     dx,al
```

(9) A/D 変換メインルーチン

A/D 変換個数が２５６個の2チャンネル (合計：５１２個) の場合のプログラム例を示します。FIFO からデータを転送する場合、絶対に ADBUSY のタイミングで読み出さないような工夫が必要になります。また、ADBUSY の時でも一度に１０個以上転送しようとすると、途中で次の変換が開始し ADBUSY のタイミングで FIFO をリードしてしまいます。１０個を超えて一度にリードしないようにします。変換データの転送先アドレスは、es:[di]に設定してあるものとします。転送完了個数を"ad_count"変数で管理して下さい。

```
ADC_MAIN:
;  while(ad_count<256*2)
;
mov     dx,ADC_STATUS
@@:
in      al,dx
test    al,11000000b
jnz     short   @f
jmp     short   @b

@@:
pushf
cli

in      al,dx
test    al,01000000b
jnz     short   @f
jmp     short   @b

@@:
in      al,dx
test    al,01000000b
jnz     short   @b

mov     dx,ADC_ADDRESS
mov     al,LM_INTSTAT
out     dx,al

in      ax,dx
and     ax,0f800h
shr     ax,11
cmp     ax,2
jge     short   @f
; 必要に応じてオーバーラン処理記述
jmp     ADC_MAIN
```

ステータスレジスタを選択します。

ADBUSY がオンであれば変換が終了した時点で、DTRDY がオンになります。

割り込みを禁止します。

ADBUSY がオンのタイミングでアクセスしないようにするための処理です。
ADBUSY がオフの場合、ADBUSY がオンになるまで待ちます。

さらに ADBUSY がオフになるまで待ちます。

LM12458 割り込みステータスレジスタを選択します。

FIFO 内のデータ数がチャンネル数 (この場合は2チャンネル) より少ない場合は、再度ステータスレジスタのリードに戻ります。

```
@@:
cmp     ax,1fh
jne     short   @
; 必要に応じてオーバーラン
; 処理記述

@@:
cmp     ax,10
jle     short   @f
mov     ax,10

@@:
mov     cx,ax
push    cx

@@:
mov     dx,ADC_ADDRESS
mov     al,LM_CONVFIFO
out     dx,al

in      ax,dx
stosw
loop    short   @b

pop     cx
add     ad_count,cx

popf
jmp     ADC_MAIN
;
; End of while
ADC_EXIT:
```

FIFO 内のデータ数が３１個以上の場合はオーバーランしています。

一度に１０個以上転送しないようにします。

転送カウント数を退避します。

LM12458 FIFO レジスタを選択します。

変換データを es:[di]に転送し di+=2 インクリメントします。

転送カウント数を復帰し、現在までの転送完了数を更新します。

割り込みを元の状態に戻します。

(10) シーケンサ停止

```
mov     dx,ADC_CONTROL
xor     al,al
out     dx,al

mov     dx,ADC_ADDRESS
mov     al,LM_CONFIG_REG
out     dx,al

mov     dx,ADC_DATA
mov     ax,2h
out     dx,ax
```

コントロールレジスタによりクロックを停止します。

LM12458 構成レジスタを選択します。

構成レジスタのリセットビットをセットします。

発行　　ラトックシステム株式会社
2002 年 2 月 14 日 第 6.0 版 第１刷発行

製品に対するお問い合わせ

REX-5054U/B の技術的なご質問やご相談の窓口を用意していますのでご利用ください。

ラトックシステム株式会社
I&L サポートセンター
〒556-0012
大阪市浪速区敷津東 1-6-14　朝日なんばビル
TEL.06-6633-6741
FAX.06-6633-3553
<サポート受付時間>
月曜－金曜（祝祭日は除く）10:00～13:00
14:00～17:00

また、インターネットのホームページでも受け付けています。

HomePage ➲ http://www.ratocsystems.com

ご注意

☑本書の内容については、将来予告なしに変更することがあります。
☑本書の内容につきましては万全を期して作成しましたが、万一ご不審な点や誤りなどお気づきになられましたらご連絡願います。
☑本製品および本製品添付のマニュアルに記載されている会社名および製品名は、各社の商品または登録商標です。
☑運用の結果につきましては、責任を負いかねますので、予めご了承願います。