AXC-002-070705

PLUG MAGIC シリーズ PIO アダプタ

AXC-PI01

ソフトウェアマニュアル

Pocket PC 2002／Pocket PC 2003／Windows Mobile 5.0 版

株式会社アドテックシステムサイエンス

― 目 次 ―

１．はじめに

本マニュアルでは、弊社の PLUG MAGIC シリーズ PIO アダプタカード「AXC-PI01」を Pocket PC 2002、Pocket PC 2003 および Windows Mobile 5.0 でご利用いただくためのドライバのインストール手順および添付ソフトウェアの使用方法などについて記述しています。

ハードウェアに関する詳細は、ユーザーズマニュアルを参照してください。

２．動作環境

本ソフトウェアは、下記の環境で動作致します。

PDA： Pocket PC 2002、Pocket PC 2003 および Windows Mobile 5.0 搭載機

OS ： Microsoft Pocket PC 2002 Software
Microsoft Pocket PC 2003 Software
Microsoft Windows Mobile 5.0 for Pocket PC

３．ファイル一覧

サポートディスク内の Software フォルダには、以下のファイルが収められています。

```
¥
├ PocketPC2002                Pocket PC 2002 用ファイルフォルダ
│  ├ Driver                   ドライバファイルフォルダ
│  │  ├ DrvSetup.exe          インストール実行ファイル
│  │  ├ DrvSetup.ini          インストール設定ファイル
│  │  └ Axcpidrv.cab          インストール cab ファイル
│  │
│  ├ CheckSoft                動作チェックアプリケーションフォルダ
│  │  ├ Setup.exe             インストール実行ファイル
│  │  ├ Setup.ini             インストール設定ファイル
│  │  ├ Axcpi01.cab           インストール cab ファイル
│  │  └ Source                ソースファイルフォルダ
│  │     ├ Axcpi01.vcw        プロジェクトワークスペース
│  │     ├ Axcpi01.vcp        プロジェクトファイル
│  │     ├ Axcpi01.rc         リソーステンプレート
│  │     ├ *.cpp              C++ソースファイル
│  │     ├ *.h                C ヘッダーファイル
│  │     └ res                リソースファイルフォルダ
│  │        ├ AXCPI01.rc2     リソーステンプレート
│  │        ├ *.bmp           ビットマップイメージ
│  │        └ *.ico           アイコン
│  │
│  └ Sample                   サンプルソースフォルダ
│     ├ eVC                  eVC サンプルソースフォルダ
│     │  ├ Axcpi01.h          DLL 定義ヘッダ
│     │  ├ Axcpi01w.h         ラッパー関数ヘッダ
│     │  ├ Axcpi01w.c         ラッパー関数
│     │  ├ Axcpi01s.c         ラッパー関数使用例
│     │  └ buildvc.txt        サンプルソース構築例
│     │
│     └ eVB                  eVB サンプルソースフォルダ
│        ├ Axcpi01.bas        DLL 定義
│        ├ Axcpi01w.bas       ラッパー関数
│        ├ Axcpi01s.bas       ラッパー関数使用例
│        └ buildvb.txt        サンプルソース構築例
```

≪次ページへ≫

≪前ページより≫

- PocketPC2003　Pocket PC 2003 用ファイルフォルダ
  - Driver　ドライバファイルフォルダ
    - DrvSetup.exe　インストール実行ファイル
    - DrvSetup.ini　インストール設定ファイル
    - Axcpidrv.cab　インストール cab ファイル
  - CheckSoft　動作チェックアプリケーションフォルダ
    - SetupPI.exe　インストール実行ファイル
    - SetupPI.ini　インストール設定ファイル
    - Axcpi01.cab　インストール cab ファイル
    - Source　ソースファイルフォルダ
      - Axcpi01.vcw　プロジェクトワークスペース
      - Axcpi01.vcp　プロジェクトファイル
      - Axcpi01.rc　リソーステンプレート
      - *.cpp　C++ソースファイル
      - *.h　C ヘッダーファイル
      - res　リソースファイルフォルダ
        - AXCPI01.rc2　リソーステンプレート
        - *.bmp　ビットマップイメージ
        - *.ico　アイコン
  - Sample　サンプルソースフォルダ
    - eVC　eVC サンプルソースフォルダ
      - Axcpi01.h　DLL 定義ヘッダ
      - Axcpi01w.h　ラッパー関数ヘッダ
      - Axcpi01w.c　ラッパー関数
      - Axcpi01s.c　ラッパー関数使用例
      - buildvc.txt　サンプルソース構築例
    - VBNET　VB.NET サンプルソースフォルダ
      - Axcpi01.vb　DLL 定義
      - Axcpi01w.vb　ラッパー関数
      - Axcpi01s.vb　ラッパー関数使用例
      - buildvbnet.txt　サンプルソース構築例

≪次ページへ≫

≪前ページより≫

- WindowsMobile5.0 WindowsMobile5.0 用ファイルフォルダ
  - Driver ドライバファイルフォルダ
    - DrvSetup.exe インストール実行ファイル
    - DrvSetup.ini インストール設定ファイル
    - AXCPI01_Drv.cab インストール cab ファイル
  - CheckSoft 動作チェックアプリケーションフォルダ
    - SetupPI.exe インストール実行ファイル
    - SetupPI.ini インストール設定ファイル
    - AXC-PI01.cab インストール cab ファイル
    - Source ソースファイルフォルダ
      - AXCPI01.sln ソリューションファイル
      - AXCPI01.ncb IntelliSense ファイル
      - AXCPI01.suo ソリューションユーザーオプションファイル
      - AXCPI01 ソースファイルフォルダ
        - AXCPI01.vcproj プロジェクトファイル
        - AXCPI01ppc.rc リソーステンプレート
        - ReadMe.txt プロジェクト概要説明
        - *.cpp C++ソースファイル
        - *.h C ヘッダーファイル
        - res リソースファイルフォルダ
          - AXCPI01ppc.rc2 リソーステンプレート
          - *.bmp ビットマップイメージ
          - *.ico アイコン
  - Sample サンプルソースフォルダ
    - VC VC サンプルソースフォルダ
      - Axcpi01.h DLL 定義ヘッダ
      - Axcpi01w.h ラッパー関数ヘッダ
      - Axcpi01w.c ラッパー関数
      - Axcpi01s.c ラッパー関数使用例
      - buildvc.txt サンプルソース構築例
    - VB VB サンプルソース
      - Axcpi01.vb DLL 定義
      - Axcpi01w.vb ラッパー関数
      - Axcpi01s.vb ラッパー関数使用例
      - buildvb.txt サンプルソース構築例
- axcpi01_ppc.pdf ソフトウェアマニュアル

４．ドライバ

本製品をご使用になる前に、ドライバをインストールする必要があります。

インストールに必要なファイルは、PocketPC2002、PocketPC2003 または WindowsMobile5.0 フォルダにそれぞれ収められています。ご使用になる環境に合わせてファイルを実行してください。

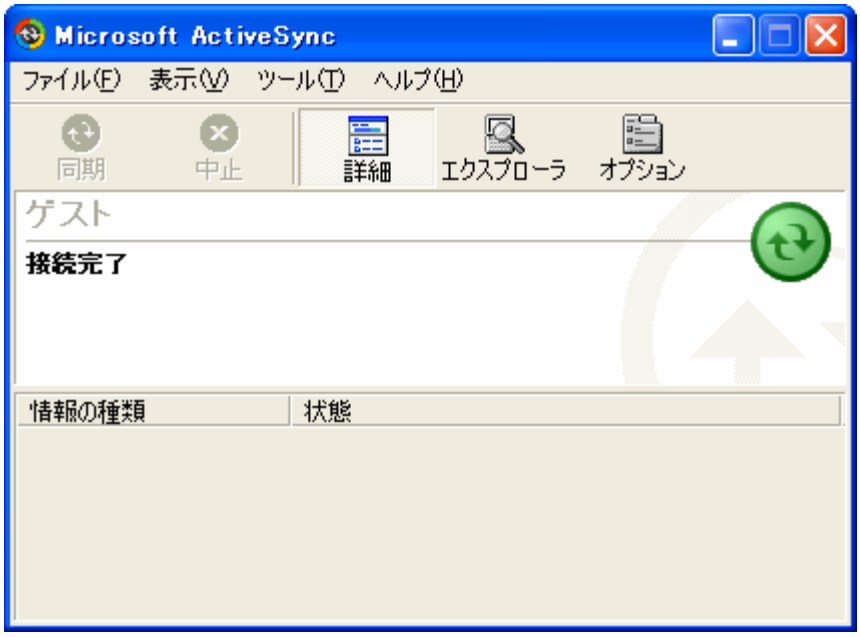
ここでは「AXC-PI01」を Pocket PC 2002 で使用する場合の手順について説明します。Pocket PC 2003 および Windows Mobile 5.0 で使用する場合は、画面の指示に従って適宜読み替えてください。

４－１．インストール前の準備

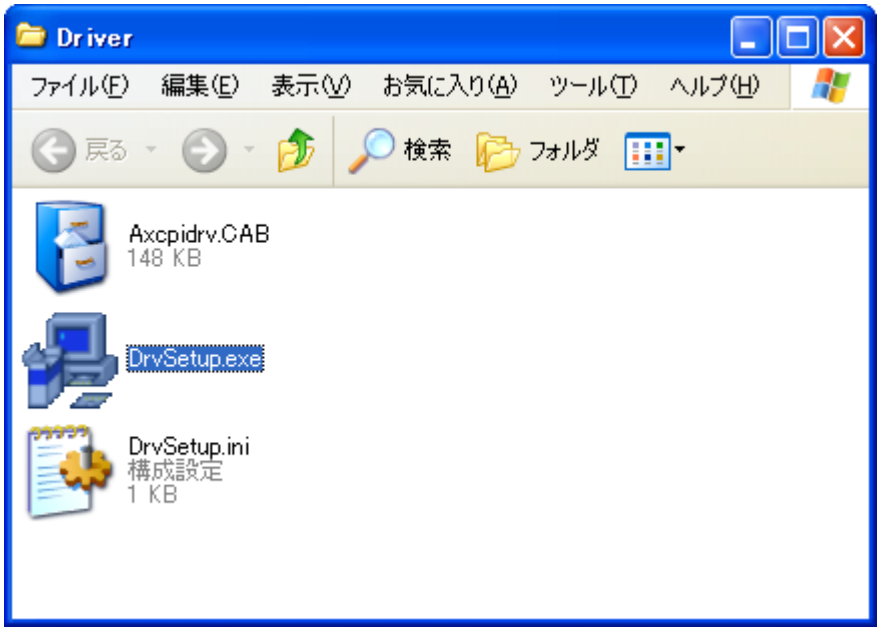
PDA に同梱されている通信ソフト「Microsoft ActiveSync」を PC にインストールしてください。PDA へのインストールは、PC 経由にて行います。

４－２．インストール

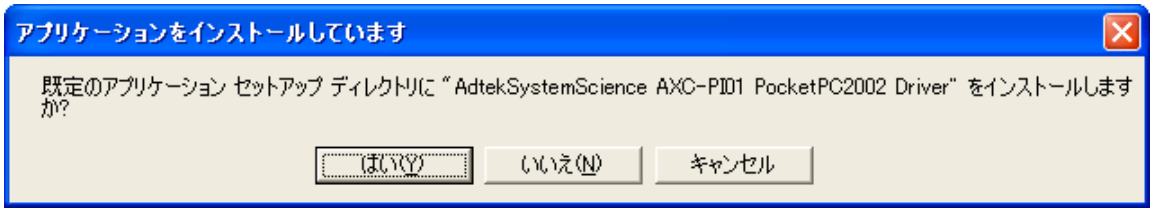
PC と PDA をシリアルケーブルもしくは USB ケーブルにて接続してください。PC にて、「Microsoft ActiveSync」が起動します。



エクスプローラ等を実行し、“ AXC-PI01 Driver ”のセットアップ用実行ファイル（PocketPC2002¥Driver¥DrvSetup.exe）を実行します。



「Microsoft ActiveSync」のウィンドウに、アプリケーションの追加と削除画面が表示されます。[既定のアプリケーション セットアップ ディレクトリに“ AdtekSystemScience AXC-PI01 PocketPC2002 Driver ”をインストールしますか？]とメッセージが表示されたら、[はい]をクリックしてください。

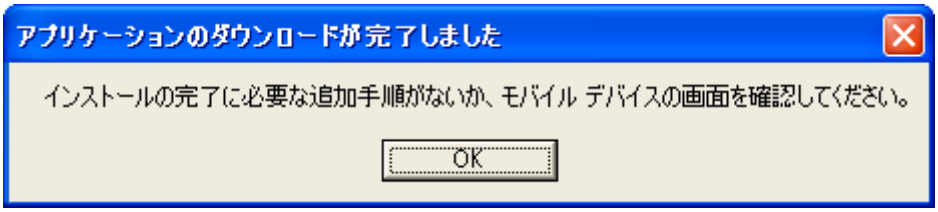


アプリケーションのインストールが開始されます。

下記のファイルが PDA の Windows フォルダにコピーされ、レジストリにデバイスの情報が登録されます。

- Axcpi01.dll　　ドライバファイル
- Axcpi01w.dll　DLL ファイル
- CardBeg.wav　カード挿入サウンドファイル
- CardEnd.wav　カード抜去サウンドファイル

［アプリケーションのダウンロードが完了しました］と表示され、PDA へのインストールは完了します。

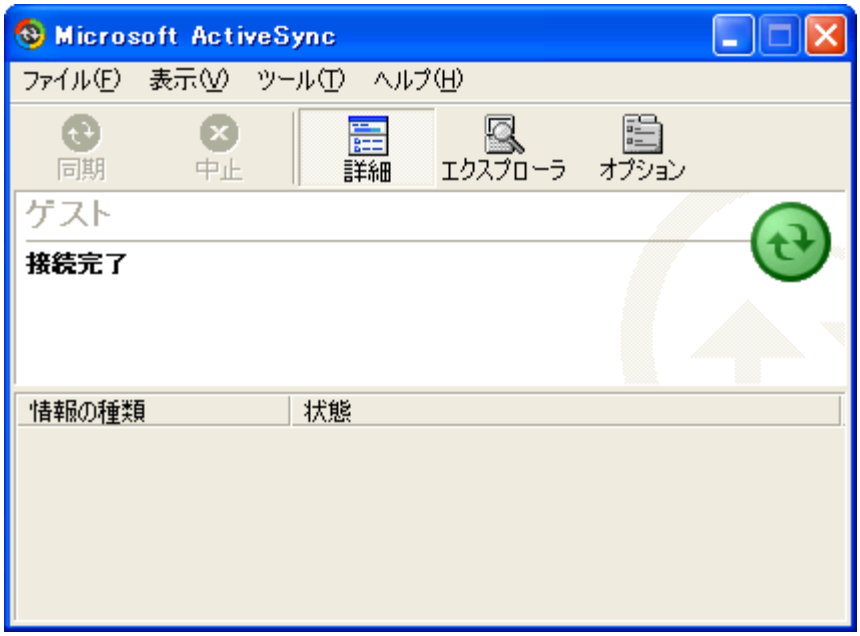


※　ここでの追加手順はありません。このメッセージで終了となります。

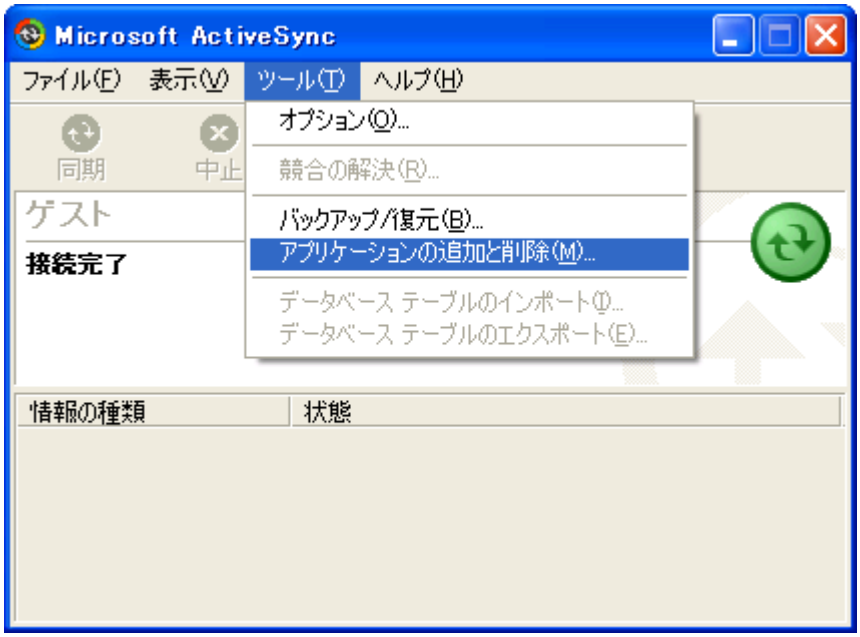
インストール完了後、「AXC-PI01」（以下「デバイス」）を PDA の CompactFlash カードスロットに挿入しますと、OS がデバイスを認識し、挿入サウンドが鳴ります。PDA からデバイスを抜きますと、抜去サウンドが鳴ります。

４－３．アンインストール

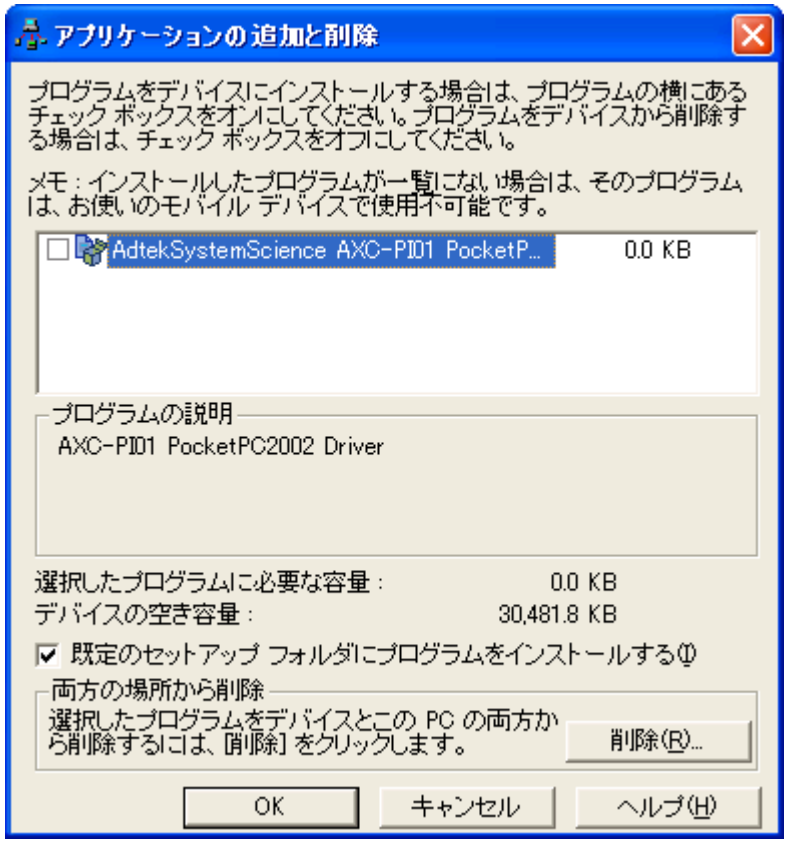
PDA を PC に接続し、「Microsoft ActiveSync」の画面が表示されることを確認してください。



メニューの［ツール］から［アプリケーションの追加と削除］を選択してください。



アプリケーションの追加と削除画面が表示されますので、[AdtekSystemScience AXC-PI01 PocketP…] のチェックを解除し、[OK] ボタンをクリックしてください（[プログラムの説明] には「AXC-PI01 PocketPC2002 Driver」と表示されています）。



下記のメッセージが表示され、自動的にアンインストールされます。

４－４．注意事項

デバイスを挿入した状態で PDA の電源を切り、再度電源を投入した場合、ドライバは一旦デバイスが抜かれ、再度挿入されたものと認識します。

また、Windows Mobile 5.0 では、デバイスを挿入した状態で電源投入を 2 回以上繰り返すと、デバイスが抜かれたままと同じ状態になる場合があります。その場合、一旦デバイスを抜き、再度挿入してください。

５．動作チェックソフト

５－１．概要

本アプリケーションソフトは、CF カード PIO アダプタ「AXC-PI01」の各機能の動作チェックを行うツールです。

「AXC-PI01」は CF カードサイズのパラレル入出力アダプタカードで PDA 等によるパラレル入出力（14 点）を行う事が可能です。

ソフト設定により 1 点毎に入出力、プルアップ、プルダウン、オープンドレインの設定が可能です。

５－２．インストールとアンインストール

インストールに必要なファイルは、PocketPC2002、PocketPC2003 または Windows Mobile 5.0 フォルダにそれぞれ収められています。ご使用になる環境に合わせてファイルを実行してください。

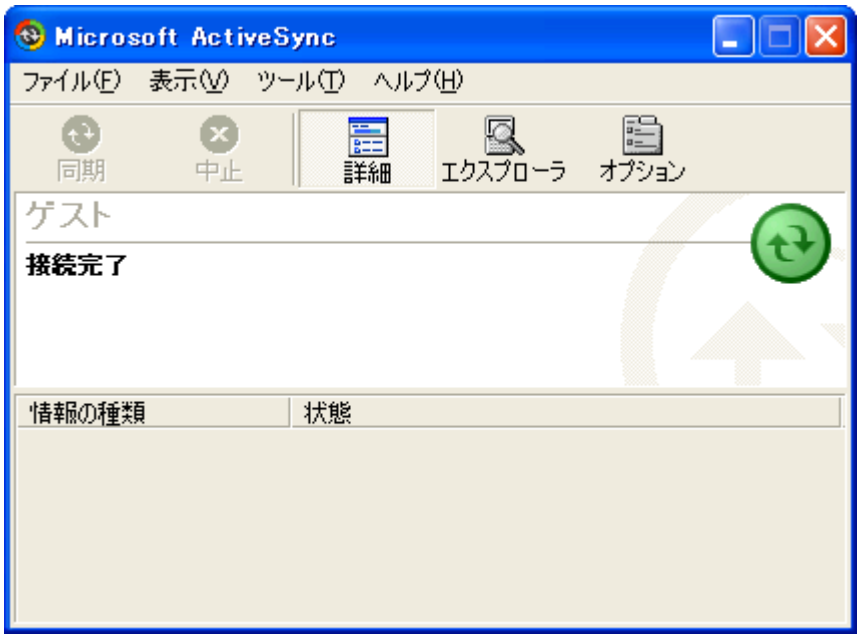
ここでは「AXC-PI01」を Pocket PC 2002 で使用する場合の手順について説明します。Pocket PC 2003 および Windows Mobile 5.0 で使用する場合は、画面の指示に従って適宜読み替えてください。

５－２－１．インストール前の準備

PDA に同梱されている通信ソフト「Microsoft ActiveSync」を PC にインストールしてください。PDA へのインストールは、PC 経由にて行います。

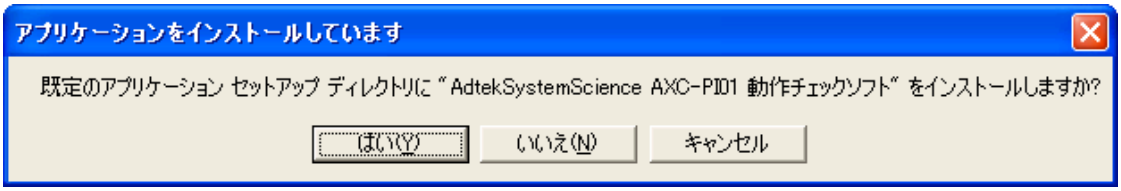
５－２－２．インストール

PC と PDA をシリアルケーブルもしくは USB ケーブルにて接続してください。PC にて、「Microsoft ActiveSync」が起動します。



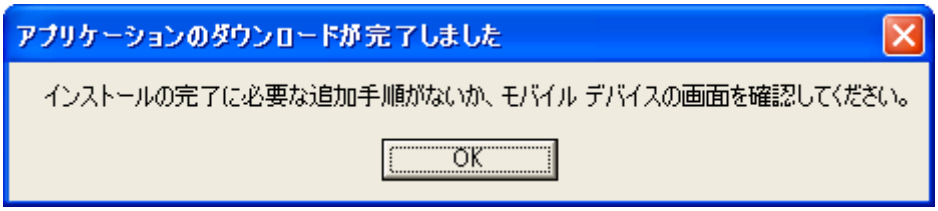
エクスプローラ等を実行し、“ AXC-PI01 動作チェックソフト ” のセットアップ用実行ファイル（PocketPC2002¥CheckSoft¥Setup.exe）を実行します。

「Microsoft ActiveSync」のウィンドウに、アプリケーションの追加と削除画面が表示されます。［既定のアプリケーション セットアップ ディレクトリに“AdtekSystemScience AXC-PI01 動作チェックソフト”をインストールしますか？］とメッセージが表示されたら、［はい］をクリックしてください。



アプリケーションのインストールが開始されます。

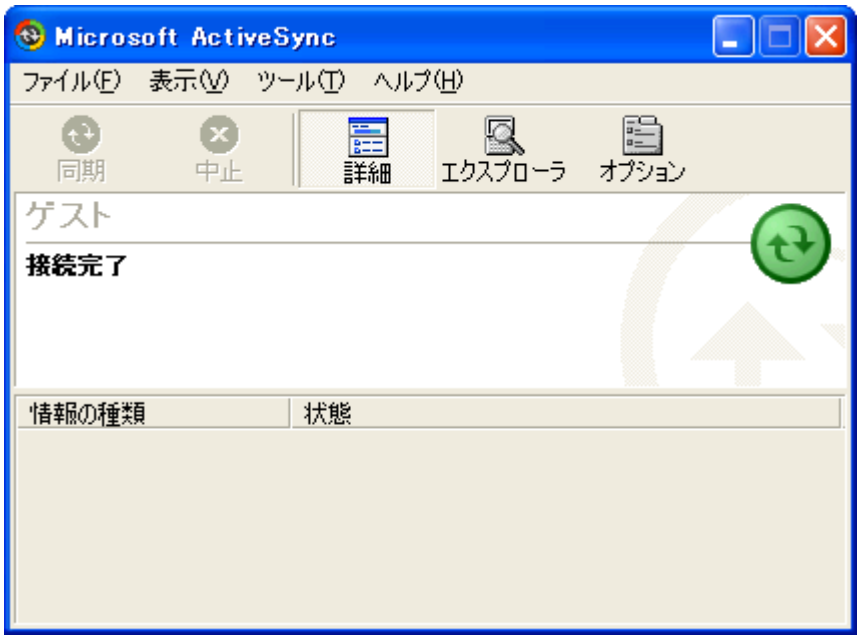
［アプリケーションのダウンロードが完了しました］と表示され、PDA へのインストールは完了します。



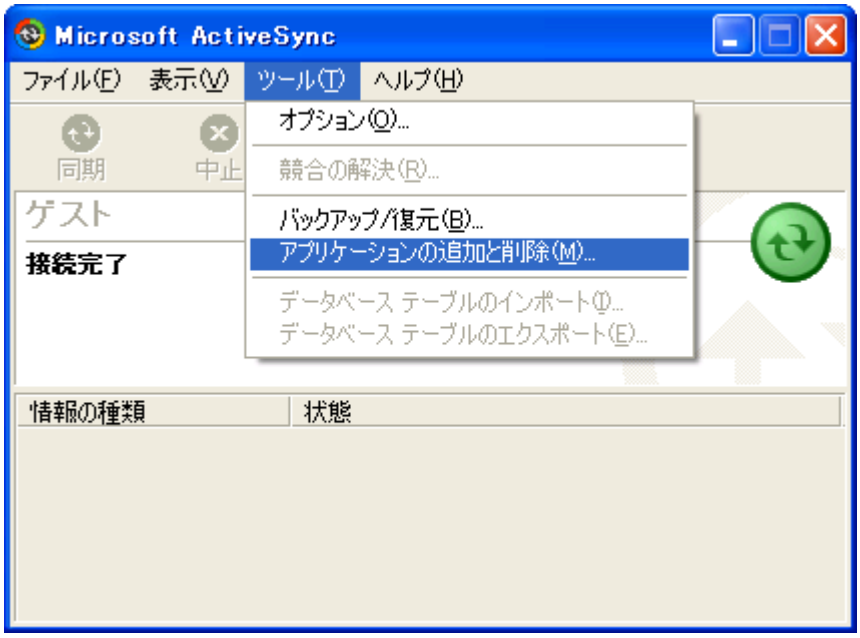
※　ここでの追加手順はありません。このメッセージで終了となります。

５－２－３．アンインストール

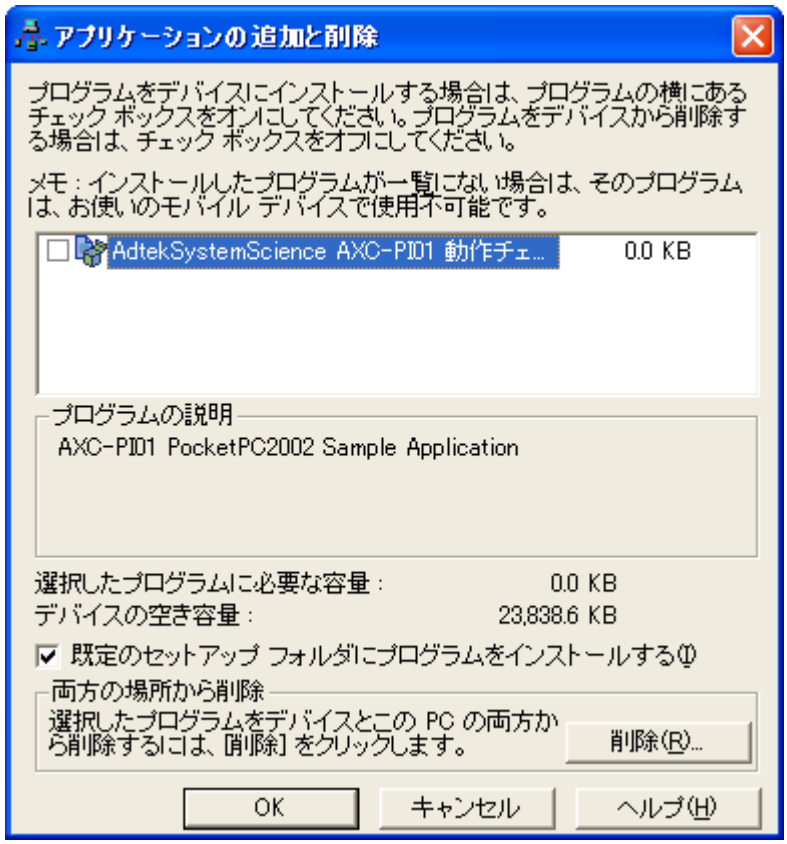
PDA を PC に接続し、「Microsoft ActiveSync」の画面が表示されることを確認してください。



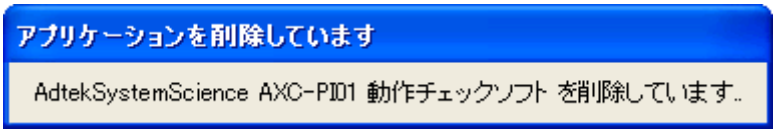
メニューの［ツール］から［アプリケーションの追加と削除］を選択してください。



アプリケーションの追加と削除画面が表示されますので、[AdtekSystemScience AXC-PI01 動作チェ...] のチェックを解除し、[OK] ボタンをクリックしてください ([プログラムの説明] には「AXC-PI01 PocketPC2002 Sample Application」と表示されています)。



下記のメッセージが表示され、自動的にアンインストールされます。



５－３．アプリケーションの起動と終了

５－３－１．起動前の準備

「AXC-PI01」（以下「デバイス」）を PDA の CompactFlash カードスロットに挿入してください。OS がデバイスを認識しますと、挿入サウンドが鳴ります。

５－３－２．起動

PDA の“スタート”メニュー－“プログラム”から本アプリケーションアイコン“AXC-PI01”をタップすることにより起動可能です。

注意！）

OS にデバイスが認識されていない場合（デバイス未挿入など）、アプリケーションエラーとなり強制終了します。

５－３－３．終了

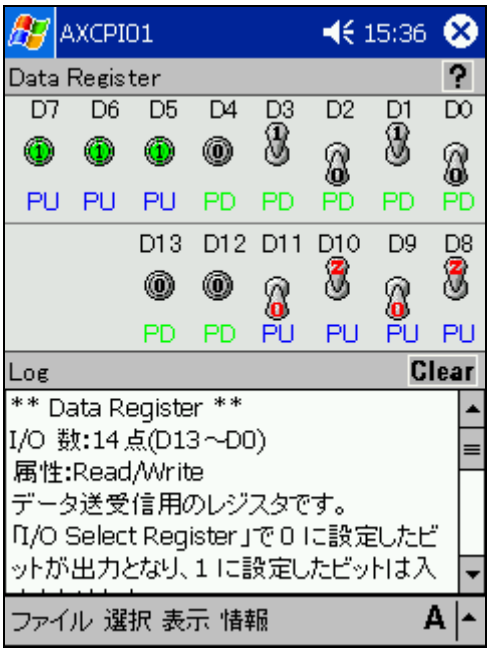
“ファイル”メニュー－“アプリケーションの終了”を選択することにより、本アプリケーションを終了することができます。

注意！）

本アプリケーション起動中に PDA からデバイスが抜かれた場合、アプリケーションエラーとなり強制終了します。

５－４．メイン画面

本アプリケーションを起動すると、下記の様な画面を表示します。



メイン画面の各表示情報は、次ページ以降の通りとなります。

５－４－１．「Data Register」

I/O 数：14 点（D13～D0）
属性：Read/Write
　データ送受信用レジスタです。
　「I/O Select Register」で“ 0 ”に設定したビットが出力となり、“ 1 ”に設定したビットは入力となります。
　データをライトした場合、そのデータは保持され、出力に設定したビットに対応する I/O コネクタの信号に反映されます。
　データをリードした場合、入出力の属性に関わらず I/O コネクタの信号の状態が読み込まれます。
　このレジスタの初期値はオール“ 1 ”です。

　画面上のイメージは各 I/O ポートの現在の状態を表します。
　各ビットの意味は以下の通りです。

（１） 入力時（「I/O Select Register」のビットデータが“ 1 ”の場合）

 　ビットデータ“ 0 ”

 　ビットデータ“ 1 ”

　100ms 周期でデータを入力し、ビットの状態を表示します。

（２） 通常出力時（「I/O Select Register」のビットデータが“ 0 ”、且つ「Output Mode Register」のビットデータが“ 1 ”の場合）

 　ビットデータ“ 0 ”

 　ビットデータ“ 1 ”

　イメージをタップするごとに  と  を切り替え、その値をレジスタに出力します。

（３） OpenDrain 出力時（「I/O Select Register」のビットデータが“ 0 ”、且つ「Output Mode Register」のビットデータが“ 0 ”の場合）

 　ビットデータ“ 0 ”

 　ビットデータ“ 1 ”

　イメージをタップするごとに  と  を切り替え、その値をレジスタに出力します。

（４） プルアップ時（「PU/PD Select Register」のビットデータが“１”の場合）

各ビットイメージの下部に“PU”と表示されます。

（５） プルダウン時（「PU/PD Select Register」のビットデータが“０”の場合）

各ビットイメージの下部に“PD”と表示されます。

５－４－２．Log

システムのログ情報（イベント情報）を表示します。表示内容は、下記の通りです。

（１） 入力データ

「Data Register」の入力値が変化したときに、その入力値を「ログ表示ボックス」に表示します。

表示されるビットは、右端から順に D0～D13 となります。

０ ： 入力値“０”

１ ： 入力値“１”

Ｘ ： 出力に設定されているビット

表示例）

```
** 入力データ **
Val : 0 0 - 0 X X X - 1 X 0 X - 0 1 1 1
```

（２） 割り込みメッセージ

割り込み発生時に割り込みメッセージを「ログ表示ボックス」に表示します。

表示例）

```
** 割り込み発生！ **
IRQIN1
```

（３） エラーメッセージ

エラー発生時にエラーメッセージを「ログ表示ボックス」に表示します。

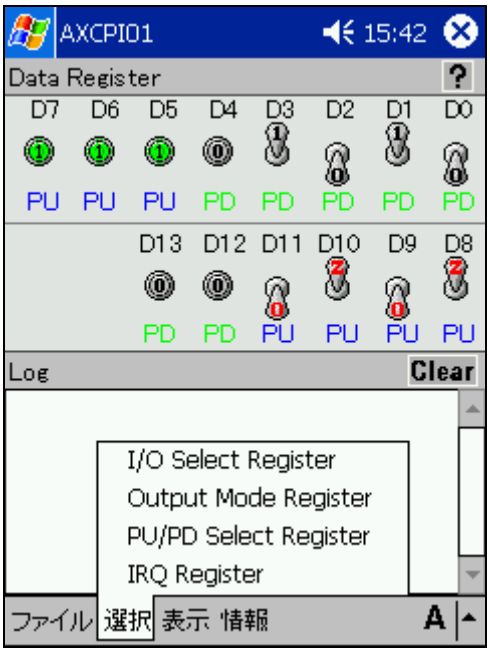
表示例）

```
** AXC-PIO1 Error **
Error code： xxxxxxxxH
使用可能なデバイスがありません。
```

※ 本ログ情報は、“Clear”ボタンによりクリアされます。また、“？”ボタンにより、「Data Register」に関するヘルプ情報が表示されます。

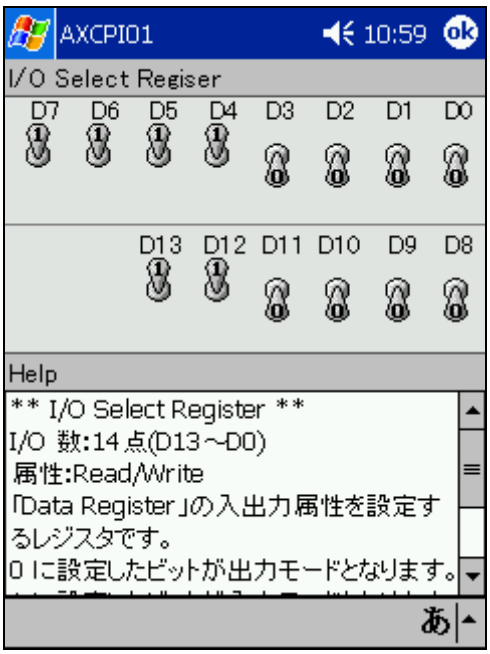
５－５．「選択」メニュー

“選択”メニューをクリックすると下記のメニューが表示されます。



各メニューの動作は、次ページ以降の通りとなります。

５－５－１．「I/O Select Register」



I/O 数：14 点（D13～D0）

属性：Read/Write

「Data Register」の入出力属性を設定するレジスタです。

“ 0 ”に設定したビットが出力モードとなります。“ 1 ”に設定したビットは入力モードとなります。

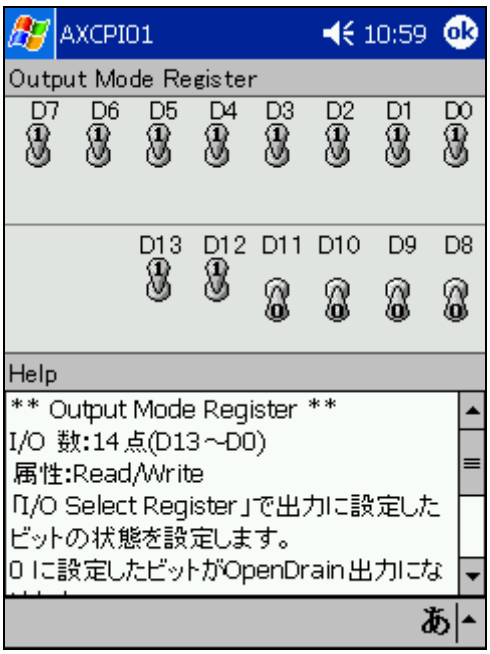
このレジスタの初期値はオール“ 1 ”です。

またこのレジスタはライトした値をリードバック可能です。

※　本アプリケーションでは、ビット D12,D13 を出力に設定すると、「IRQ Register」の割り込み許可ビット D0,D1 の値が自動的に“ 1 ”となり、割り込み禁止となります。

その後、直接「IRQ Register」のビット D0,D1 を操作し、割り込みを許可することは可能です。

５－５－２．「Output Mode Register」



I/O 数：14 点（D13～D0）

属性：Read/Write

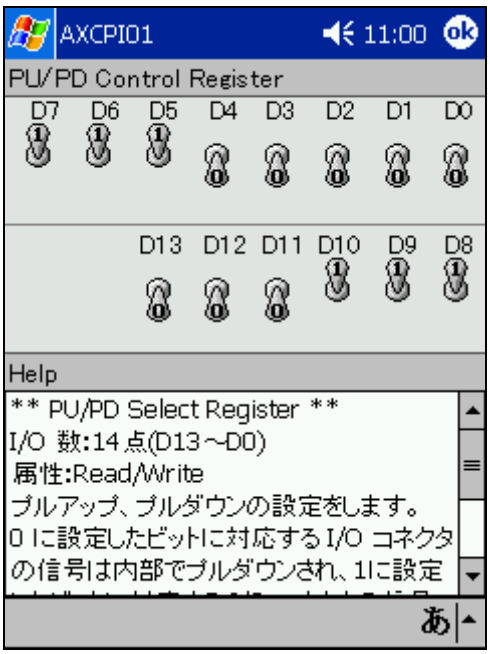
「I/O Select Register」で出力に設定したビットの状態を設定します。

“ 0 ”に設定したビットが OpenDrain 出力になります。“ 1 ”に設定したビットはトーテムポール出力となります。

このレジスタの初期値はオール“ 1 ”です。

またこのレジスタはライトした値をリードバック可能です。

５－５－３．「PU/PD Select Register」



I/O 数：14 点（D13～D0）

属性：Read/Write

プルアップ、プルダウンの設定をします。

“ 0 ”に設定したビットに対応する I/O コネクタの信号は内部でプルダウンされ、“ 1 ”に設定したビットに対応する I/O コネクタの信号は内部でプルアップされます。

このレジスタの初期値はオール“ 1 ”です。

またこのレジスタはライトした値をリードバック可能です。

５－５－４．「IRQ Register」

I/O 数：2 点（D1～D0）

D0：IRQEN1（IRQ1 Enable Bit）

D1：IRQEN2（IRQ2 Enable Bit）

属性：Read/Write

IRQEN1,IRQEN2 は割込み許可ビットです。

IRQEN1 を“ 0 ”に設定すると、IRQ1 割込みが有効になり、IRQEN2 を“ 0 ”に設定すると、IRQ2 割込みが有効になります。IRQ1 割込みは外部 I/O コネクタ 13Pin、IRQ2 割込みは外部 I/O コネクタ 14Pin が使用されます。

尚､この設定に関わらず｢Data Register｣の D12,D13 は信号の状態をリード可能です。

割込みは信号の立下りエッジで発生します。

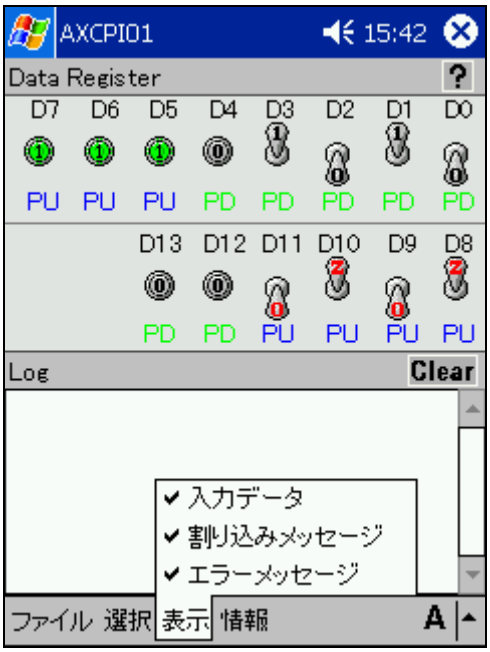
このレジスタの初期値はオール“ 1 ”です。

またこのレジスタはライトした値をリードバック可能です。

割込みを使用する場合は、「I/O Select Register」の割込みを使用する信号（D12 もしくは D13）を“ 1 ”に設定し、入力モードにする必要があります。

５－６．「表示」メニュー

“表示”メニューをクリックすると下記のメニューが表示されます。



５－６－１．「入力データ」

「入力データ」の表示をメイン画面「ログ表示ボックス」に行うかの選択を行います。
デフォルトでは、表示するとなります。

５－６－２．「割り込みメッセージ」

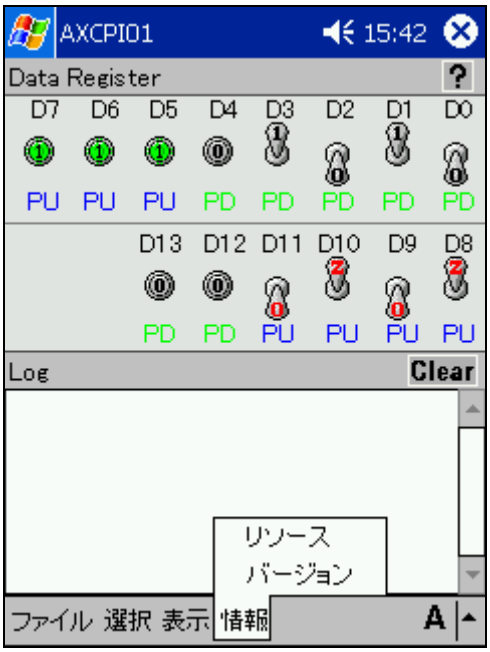
「割り込みメッセージ」の表示をメイン画面「ログ表示ボックス」に行うかの選択を行います。
デフォルトでは、表示するとなります。

５－６－３．「エラーメッセージ」

「エラーメッセージ」の表示をメイン画面「ログ表示ボックス」に行うかの選択を行います。
デフォルトでは、表示するとなります。

５－７．「情報」メニュー

“情報”メニューをクリックすると下記のメニューが表示されます。



５－７－１．「リソース」

以下のシステムリソース情報を、メイン画面「ログ表示ボックス」に表示します。

```
** リソース情報 **
I/O Address: xxxxxH - xxxxxH
```

５－７－２．「バージョン」

以下のドライバおよび DLL のバージョン情報を、メイン画面「ログ表示ボックス」に表示します。

```
** バージョン情報 **
Driver Version: xxxxxH
DLL Version: xxxxxH
```

５－８．その他

５－８－１．エラーメッセージについて

表示例）下記の様にエラー内容と、エラーコードを表示します。

```
** AXC-PIO1 Error **
Error code： xxxxxxxxH
無効な論理ソケット番号です
```

エラーコードにつきましては、以下の通りです。

00000000 H ： 異状なし（正常終了）
00000001 H ： Windows の GetLastError() をコールしてください
正常にドライバがロードされていない可能性があります
00000002 H ： 使用可能なデバイスがありません
00000003 H ： 指定のデバイスは使用中です
00000004 H ： 無効な論理ソケットです
00000005 H ： リソースエラー
00000006 H ： 不正なポートを要求しました
00000007 H ： 不正な引数を要求しました

５－８－２．注意事項

本アプリケーション起動中に PDA の電源を切り、再度電源を投入した場合、アプリケーションはデバイスが抜かれたものと認識し、強制終了します。詳しくは「５－３－３．終了」の項目を参照してください。

また、Windows Mobile 5.0 では、デバイスを挿入した状態で電源投入を 2 回以上繰り返すと、デバイスが抜かれたままと同じ状態になる場合があります。その場合、一旦デバイスを抜き、再度挿入してください。

６．サンプルソース

動作環境ごとに以下のサンプルソースがございます。

Pocket PC 2002：

eMbedded Visual C++ 3.0 版および eMbedded Visual Basic 3.0 版

Pocket PC 2003：

eMbedded Visual C++ 4.0 版および Visual Basic .NET 2003 版

Windows Mobile 5.0：

Visual C++ 2005 版および Visual Basic 2005 版

サンプルソースは各開発環境にて実行してお試しいただけます。

開発環境およびビルド方法の詳細については、各ディレクトリ内の buildxx.txt の例をご覧ください。

DLL 内の関数のインターフェースは、各関数のヘッダ、注釈および「７．API 仕様」を参照してください。

＜ラッパー関数＞

axcpi01w.* のように「w」が付いているファイルには、DLL 内の関数を簡単にコールするためのラッパー（Wrapper）関数が定義されています（DLL のロード/アンロード関数も含まれています）。

＜ラッパー関数使用例＞

axcpi01s.* のように「s」が付いているファイルには、ラッパー関数を使用してデバイスを制御する例が記されています。

※　サンプルソースのご利用については、各開発環境および OS・言語に対する充分な理解を前提としております。よって、これらそのものの使用方法に関するお問い合わせには一切お答えいたしかねますので、あらかじめご了承ください。

７．API 仕様

７－１．概要

本章では、AXC-PI01 用ユーザ公開 API を定義します。
デバイスドライバの詳細については触れていません。

７－２．プログラム構成

本章で定義される API は、Axcpi01w.dll およびデバイスドライバ Axcpi01.dll により実現されます。

７－３．API リファレンス

７－３－１．Axcpi01GetVersion

■機　能　バージョン情報取得

■形　式　eMbedded Visual C++ 3.0／4.0 および Visual C++ 2005

```
BOOL Axcpi01GetVersion
(
  PDWORD pdwDllVersion,  //DLL のバージョン
  PDWORD pdwDrvVersion   //DRIVER のバージョン
);
```

■形　式　eMbedded Visual Basic 3.0

```
Function Axcpi01GetVersion
(
  ByRef pdwDllVersion As Long,  'DLL のバージョン
  ByRef pdwDrvVersion As Long   'DRIVER のバージョン
) As Boolean
```

■形　式　Visual Basic .NET 2003

```
Function Axcpi01GetVersion
(
  ByRef pdwDllVersion As Integer,  'DLL のバージョン
  ByRef pdwDrvVersion As Integer   'DRIVER のバージョン
) As Boolean
```

■形　式　Visual Basic 2005

```
Function Axcpi01GetVersion
(
  ByRef pdwDllVersion As Integer,  'DLL のバージョン
  ByRef pdwDrvVersion As Integer   'DRIVER のバージョン
) As Boolean
```

■入　力　pdwDllVersion

DLL のバージョン番号を格納する領域へのポインタ。
NULL 可。

pdwDrvVersion

デバイスドライバのバージョン番号を格納する領域へのポインタ。
NULL 可。

■出　力　*pdwDllVersion

DLL のバージョン番号。

*pdwDrvVersion

デバイスドライバのバージョン番号。

■戻り値　API が正常終了したか、失敗したかを返します。

FALSE　　失敗
TRUE　　正常終了

■解　説　DLL とデバイスドライバのバージョン番号を取得します。
バージョン情報はリソースから取得せず、プログラムで指定します。
Axcpi01Create 処理をしなくてもバージョン番号を取得できます。

■エラー　AXCPI01_ERR_SYSTEM

//Windows の GetLastError() をコールしてください。
//所定のフォルダ（PDA の Windows フォルダ）にドライバ
//ファイル（Axcpi01.dll）がない可能性があります。

７－３－２．Axcpi01Create

■機　能　デバイスの使用を宣言

■形　式　eMbedded Visual C++ 3.0／4.0 および Visual C++ 2005

```
#define AXCPI01_SOCKET_AUTO  ((WORD) ~0U)
BOOL Axcpi01Create
(
   PWORD  pwLogSocket,  //論理ソケット指定
   HWND   hWnd,         //メッセージを受けるウインドウハンドル
   UINT   uWMBase,      //メッセージ ID のベース
   PVOID  pvInitArg     //予約
);
```

■形　式　eMbedded Visual Basic 3.0

```
Const AXCPI01_SOCKET_AUTO = &HFFFF
Function Axcpi01Create
(
   ByRef pwLogSocket As Integer,   '論理ソケット指定
   ByVal hWnd As Long,             'メッセージを受ける
                                   'ウインドウハンドル
   ByVal uWMBase As Long,          'メッセージ ID のベース
   ByRef pvInitArg As Any          '予約
) As Boolean
```

■形　式　Visual Basic .NET 2003

```
Const AXCPI01_SOCKET_AUTO As Short = &HFFFFs
Function Axcpi01Create
(
   ByRef pwLogSocket As Short,   '論理ソケット指定
   ByVal hWnd As Integer,        'メッセージを受ける
                                 'ウインドウハンドル
   ByVal uWMBase As Integer,     'メッセージ ID のベース
   ByRef pvInitArg As Integer    '予約
) As Boolean
```

■形　式　Visual Basic 2005

```
Const AXCPI01_SOCKET_AUTO As Short = &HFFFFs
Function Axcpi01Create
(
   ByRef pwLogSocket As Short,   '論理ソケット指定
   ByVal hWnd As Integer,        'メッセージを受ける
                                 'ウインドウハンドル
   ByVal uWMBase As Integer,     'メッセージ ID のベース
   ByRef pvInitArg As Integer    '予約
) As Boolean
```

■入　力　pwLogSocket

使用したいデバイスを指定します。論理ソケット番号は0から始まります。AXCPI01_SOCKET_AUTO を指定した場合、使用可能な論理ソケットを探します。

すでにアプリケーションにより使用されているデバイスはスキップします。

以後、アプリケーションはこの値でデバイスを識別します。

hWnd

ドライバまたは DLL が送出するメッセージを受け取るウィンドウのハンドルを指定します。メッセージを使用しない場合はNULL を指定します。

uWMBase

送出するメッセージ ID のベース値を指定します。デバイスから発生したイベントによって、uWMBase に特定の値を加えた ID のメッセージが送出されます。uWMBase の推奨値は 0x2000～0x6ff8 です。

pvInitArg

使用しません。Visual Basic の場合はダミーの変数を渡してください。

■出　力　*pwLogSocket

使用可能なデバイスが見つかった場合は、その論理ソケット番号を格納します。見つからなかった場合、この値は未定です。

■戻り値　API が正常終了したか、失敗したかを返します。

FALSE　失敗

TRUE　正常終了

■解　説　ソケットに存在しているであろうデバイスを、アプリケーションが使用することをデバイスドライバに通知します。ソケットにデバイスが存在していない場合は、エラーとなります。ソケット内のデバイスがすでに他のアプリケーションで使用されている場合もエラーとなります。これにより、1 つのデバイスは単一のアプリケーションから排他的に使用されます。他の API（Axcpi01GetVersion API を除く）を使う前に必ずこの API を使ってください。

デバイスの使用を宣言したままアプリケーションを終了する場合は、Axcpi01Close API を使用してデバイスを開放してください。

■注　意　一度アプリケーションに割り当てられたデバイスが抜去された場合は、それまでの割り当ては解除されます。デバイスが再度挿入された後、アプリケーションは再度この API を呼ぶ必要があります。その際には、本 API を呼び出す前に、Axcpi01Close API を使用する必要はありません。

■エラー　AXCPI01_ERR_SYSTEM

//Windows の GetLastError() をコールしてください。

AXCPI01_ERR_NO_DEVICE

//使用可能なデバイスがありません。

//（AXCPI01_SOCKET_AUTO を指定した場合）。

AXCPI01_ERR_IN_USE

//指定のデバイスは使用中です。

AXCPI01_ERR_INVALID_SOCKET

//無効な論理ソケットです。

//（0～15 及び AXCPI01_SOCKET_AUTO 以外の値の場合）。

この API が失敗した場合、Axcpi01GetLastError() の wLogSocket には AXCPI01_SOCKET_AUTO を指定してください。

７－３－３．Axcpi01Close

■機　能　デバイス開放

■形　式　eMbedded Visual C++ 3.0／4.0 および Visual C++ 2005

```
BOOL Axcpi01Close
(
  WORD wLogSocket    //論理ソケット指定
);
```

■形　式　eMbedded Visual Basic 3.0

```
Function Axcpi01Close
(
  ByVal wLogSocket As Integer  '論理ソケット指定
) As Boolean
```

■形　式　Visual Basic .NET 2003

```
Function Axcpi01Close
(
  ByVal wLogSocket As Short   '論理ソケット指定
) As Boolean
```

■形　式　Visual Basic 2005

```
Function Axcpi01Close
(
  ByVal wLogSocket As Short   '論理ソケット指定
) As Boolean
```

■入　力　wLogSocket

このデバイスを開放します。

■戻り値　API が正常終了したか、失敗したかを返します。

FALSE　失敗

TRUE　正常終了

■解　説　アプリケーションがデバイスの使用を終了し、デバイスを他のアプリケーションに開放することをデバイスドライバに通知します。
アプリケーションを終了する前に必ずこの API を呼び出してください。
この API を使用するには、事前に Axcpi01Create を使いデバイスを開いておく必要があります。

■エラー　AXCPI01_ERR_SYSTEM
//Windows の GetLastError() をコールしてください。
AXCPI01_ERR_INVALID_SOCKET
//無効な論理ソケットです（0～15 以外の値の場合)。
//またはデバイスがクリエイトされていません。

７－３－４．Axcpi01GetResource

■機　能　リソース情報取得

■形　式　eMbedded Visual C++ 3.0／4.0 および Visual C++ 2005

```
BOOL Axcpi01GetResource
(
  WORD    wLogSocket,        //論理ソケット指定
  PDWORD  pdwIOPortBase,     //I/O ポートベースアドレス
  PDWORD  pdwIOPortLength    //I/O ポート長
);
```

■形　式　eMbedded Visual Basic 3.0

```
Function Axcpi01GetResource
(
  ByVal wLogSocket As Integer,    '論理ソケット指定
  ByRef pdwIOPortBase As Long,    'I/O ポートベースアドレス
  ByRef pdwIOPortLength As Long   'I/O ポート長
) As Boolean
```

■形　式　Visual Basic .NET 2003

```
Function Axcpi01GetResource
(
  ByVal wLogSocket As Short,         '論理ソケット指定
  ByRef pdwIOPortBase As Integer,    'I/O ポートベースアドレス
  ByRef pdwIOPortLength As Integer   'I/O ポート長
) As Boolean
```

■形　式　Visual Basic 2005

```
Function Axcpi01GetResource
(
  ByVal wLogSocket As Short,         '論理ソケット指定
  ByRef pdwIOPortBase As Integer,    'I/O ポートベースアドレス
  ByRef pdwIOPortLength As Integer   'I/O ポート長
) As Boolean
```

■入　力　wLogSocket

対象のデバイスを論理ソケット番号で指定。

pdwIOPortBase

I/O ベースアドレスを格納する領域へのポインタ。

pdwIOPortLength

I/O ポート長を格納する領域へのポインタ。

■出　力　*pdwIOPortBase

割り当てられている I/O ベースアドレス。

*pdwIOPortLength

割り当てられている I/O ポート長。

■戻り値　API が正常終了したか、失敗したかを返します。

FALSE　失敗

TRUE　正常終了

■解　説　wLogSocket で指定されたデバイスに割り当てられているリソースを、表示用に取得します。この API を使用するには、事前に Axcpi01Create を使いデバイスを開いておく必要があります。

■エラー　AXCPI01_ERR_INVALID_SOCKET

//無効な論理ソケットです（0～15 以外の値の場合）。

//またはデバイスがクリエイトされていません。

AXCPI01_ERR_INVALID_ARGUMENT

//pdwIOPortBase, pdwIOPortLength が NULL です。

AXCPI01_ERR_RESOURCE

//リソース取得に失敗しました。

７－３－５．Axcpi01InPortB

■機　能　各ポートに対し 1 バイト入力

■形　式　eMbedded Visual C++ 3.0／4.0 および Visual C++ 2005

```
BOOL Axcpi01InPortB
(
  WORD  wLogSocket,  //論理ソケット指定
  DWORD dwPort,      //対象ポート指定
  PBYTE pbValue      //入力データ
);
```

■形　式　eMbedded Visual Basic 3.0

```
Function Axcpi01InPortB
(
  ByVal wLogSocket As Integer,  '論理ソケット指定
  ByVal dwPort As Long,         '対象ポート指定
  ByRef pbValue As Byte         '入力データ
) As Boolean
```

■形　式　Visual Basic .NET 2003

```
Function Axcpi01InPortB
(
  ByVal wLogSocket As Short,  '論理ソケット指定
  ByVal dwPort As Integer,    '対象ポート指定
  ByRef pbValue As Byte       '入力データ
) As Boolean
```

■形　式　Visual Basic 2005

```
Function Axcpi01InPortB
(
  ByVal wLogSocket As Short,  '論理ソケット指定
  ByVal dwPort As Integer,    '対象ポート指定
  ByRef pbValue As Byte       '入力データ
) As Boolean
```

■入　力　wLogSocket

対象のデバイスを論理ソケット番号で指定。

dwPort

対象のポートを数字または定数で指定（0～9）。

pbValue

入力したデータを格納する領域へのポインタ。

NULL 不可。

■出　力　*pbValue

入力バイトデータ。

■戻り値　API が正常終了したか、失敗したかを返します。

FALSE　失敗

TRUE　正常終了

■解　説　各ポートから 1 バイト入力します。

■エラー　AXCPI01_ERR_INVALID_SOCKET

//無効な論理ソケットです（0～15 以外の値の場合）。

//またはデバイスがクリエイトされていません。

AXCPI01_ERR_INVALID_ARGUMENT

//pbValue が NULL です。

AXCPI01_ERR_INVALID_PORT

//無効なポート番号（0～9 以外の値の場合）を要求しました。

７－３－６．Axcpi01OutPortB

■機　能　各ポートに対し 1 バイト出力

■形　式　eMbedded Visual C++ 3.0／4.0 および Visual C++ 2005

```
BOOL Axcpi01OutPortB
(
  WORD  wLogSocket,  //論理ソケット指定
  DWORD dwPort,      //対象ポート指定
  BYTE  bValue       //出力データ
);
```

■形　式　eMbedded Visual Basic 3.0

```
Function Axcpi01OutPortB
(
  ByVal wLogSocket As Integer,  '論理ソケット指定
  ByVal dwPort As Long,         '対象ポート指定
  ByVal bValue As Byte          '入力データ
) As Boolean
```

■形　式　Visual Basic .NET 2003

```
Function Axcpi01OutPortB
(
  ByVal wLogSocket As Short,  '論理ソケット指定
  ByVal dwPort As Integer,    '対象ポート指定
  ByVal bValue As Byte        '入力データ
) As Boolean
```

■形　式　Visual Basic 2005

```
Function Axcpi01OutPortB
(
  ByVal wLogSocket As Short,  '論理ソケット指定
  ByVal dwPort As Integer,    '対象ポート指定
  ByVal bValue As Byte        '入力データ
) As Boolean
```

■入　力　wLogSocket

対象のデバイスを論理ソケット番号で指定。

dwPort

対象のポートを数字または定数で指定（0～9）。

bValue

出力バイトデータ。

■戻り値　API が正常終了したか、失敗したかを返します。

FALSE　　失敗

TRUE　　正常終了

■解　説　各ポートに 1 バイト出力します。

■エラー　AXCPI01_ERR_INVALID_SOCKET

//無効な論理ソケットです（0～15 以外の値の場合）。

//またはデバイスがクリエイトされていません。

AXCPI01_ERR_INVALID_PORT

//無効なポート番号（0～9 以外の値の場合）を要求しました。

７－３－７．Axcpi01InPortW

■機　能　各ポートに対し 1 ワード入力

■形　式　eMbedded Visual C++ 3.0／4.0 および Visual C++ 2005

```
BOOL Axcpi01InPortW
(
  WORD  wLogSocket,  //論理ソケット指定
  DWORD dwPort,      //対象ポート指定
  PWORD pwValue      //入力データ
);
```

■形　式　eMbedded Visual Basic 3.0

```
Function Axcpi01InPortW
(
  ByVal wLogSocket As Integer,  '論理ソケット指定
  ByVal dwPort As Long,         '対象ポート指定
  ByRef pwValue As Integer      '入力データ
) As Boolean
```

■形　式　Visual Basic .NET 2003

```
Function Axcpi01InPortW
(
  ByVal wLogSocket As Short,  '論理ソケット指定
  ByVal dwPort As Integer,    '対象ポート指定
  ByRef pwValue As Short      '入力データ
) As Boolean
```

■形　式　Visual Basic 2005

```
Function Axcpi01InPortW
(
  ByVal wLogSocket As Short,  '論理ソケット指定
  ByVal dwPort As Integer,    '対象ポート指定
  ByRef pwValue As Integer    '入力データ
) As Boolean
```

■入　力　wLogSocket

対象のデバイスを論理ソケット番号で指定。

dwPort

対象のポートを数字または定数で指定。

Low 側のポート番号（0,2,4,6,8）のみ指定可能。

pwValue

入力したデータを格納する領域へのポインタ。

NULL 不可。

■出　力　*pwValue

入力ワードデータ。

■戻り値　API が正常終了したか、失敗したかを返します。

FALSE　失敗

TRUE　正常終了

■解　説　各ポートから 1 ワード入力します。

■注　意　High 側のポート番号（1,3,5,7,9）は指定できません。

■エラー　AXCPI01_ERR_INVALID_SOCKET

//無効な論理ソケットです（0～15 以外の値の場合）。

//またはデバイスがクリエイトされていません。

AXCPI01_ERR_INVALID_ARGUMENT

//pwValue が NULL です。

AXCPI01_ERR_INVALID_PORT

//無効なポート番号（0,2,4,6,8 以外の値の場合）を要求しました。

７－３－８．Axcpi01OutPortW

■機　能　各ポートに対し 1 ワード出力

■形　式　eMbedded Visual C++ 3.0／4.0 および Visual C++ 2005

```
BOOL Axcpi01OutPortW
(
  WORD  wLogSocket,  //論理ソケット指定
  DWORD dwPort,      //対象ポート指定
  WORD  wValue       //出力データ
);
```

■形　式　eMbedded Visual Basic 3.0

```
Function Axcpi01OutPortW
(
  ByVal wLogSocket As Integer,  '論理ソケット指定
  ByVal dwPort As Long,         '対象ポート指定
  ByVal wValue As Integer       '入力データ
) As Boolean
```

■形　式　Visual Basic .NET 2003

```
Function Axcpi01OutPortW
(
  ByVal wLogSocket As Short,  '論理ソケット指定
  ByVal dwPort As Integer,    '対象ポート指定
  ByVal wValue As Short       '入力データ
) As Boolean
```

■形　式　Visual Basic 2005

```
Function Axcpi01OutPortW
(
  ByVal wLogSocket As Short,  '論理ソケット指定
  ByVal dwPort As Integer,    '対象ポート指定
  ByVal wValue As Integer     '入力データ
) As Boolean
```

■入　力　wLogSocket

対象のデバイスを論理ソケット番号で指定。

dwPort

対象のポートを数字または定数で指定。

Low 側のポート番号（0,2,4,6,8）のみ指定可能。

wValue

出力ワードデータ。

■戻り値　API が正常終了したか、失敗したかを返します。

FALSE　失敗

TRUE　正常終了

■解　説　各ポートに 1 ワード出力します。

■注　意　High 側のポート番号（1,3,5,7,9）は指定できません。

■エラー　AXCPI01_ERR_INVALID_SOCKET

//無効な論理ソケットです（0～15 以外の値の場合）。

//またはデバイスがクリエイトされていません。

AXCPI01_ERR_INVALID_PORT

//無効なポート番号（0,2,4,6,8 以外の値の場合）を要求しました。

７－３－９．Axcpi01GetIRQStatus

■機　能　割り込みステータス情報取得

■形　式　eMbedded Visual C++ 3.0／4.0 および Visual C++ 2005

```
BOOL Axcpi01GetIRQStatus
(
  WORD  wLogSocket,  //論理ソケット指定
  PBYTE pbStatus     //割り込みステータス情報
);
```

■形　式　eMbedded Visual Basic 3.0

```
Function Axcpi01GetIRQStatus
(
  ByVal wLogSocket As Integer,  '論理ソケット指定
  ByRef pbStatus As Byte        '割り込みステータス情報
) As Boolean
```

■形　式　Visual Basic .NET 2003

```
Function Axcpi01GetIRQStatus
(
  ByVal wLogSocket As Short,  '論理ソケット指定
  ByRef pbStatus As Byte      '割り込みステータス情報
) As Boolean
```

■形　式　Visual Basic 2005

```
Function Axcpi01GetIRQStatus
(
  ByVal wLogSocket As Short,  '論理ソケット指定
  ByRef pbStatus As Byte      '割り込みステータス情報
) As Boolean
```

■入　力　wLogSocket

対象のデバイスを論理ソケット番号で指定。

■出　力　*pbStatus

割り込みステータス

IRQ1 発生時には戻り値の BIT0 が“1”となります。
IRQ2 発生時には戻り値の BIT1 が“1”となります。

未発生時　　：00h
IRQ1 発生時　：01h
IRQ2 発生時　：02h
IRQ1,IRQ2 発生時：03h

■戻り値　API が正常終了したか、失敗したかを返します。

FALSE　失敗
TRUE　正常終了

■解　説　発生した割り込みのステータス情報を取得します。

「IRQ Register」で許可した割り込みが発生した場合、本 API を呼び出すと、割り込みステータス情報を返します。
出力値 *pbStatus の BIT0,BIT1 がそれぞれ IRQ1,IRQ2 の割り込みに対応します。
割り込み発生後、本 API が呼び出されるまで、割り込みステータス情報は保持され、値を取得すると BIT0,BIT1 がそれぞれ“0”に初期化されます。

■注　意　本 API では、最初に発生した割り込みのステータス情報しか取得することができません。

したがって、本 API を呼び出す前に、連続で割り込みが発生した場合は、2 回目以降の割り込みのステータス情報は無視されます。

■エラー　AXCPI01_ERR_INVALID_SOCKET

//無効な論理ソケットです（0～15 以外の値の場合）。
//またはデバイスがクリエイトされていません。

AXCPI01_ERR_INVALID_ARGUMENT

//pbStatus が NULL です。

７－３－１０．Axcpi01GetLastError

■機　能　エラーコード取得

■形　式　eMbedded Visual C++ 3.0／4.0 および Visual C++ 2005

```
DWORD Axcpi01GetLastError
(
   WORD wLogSocket    //論理ソケット指定
);
```

■形　式　eMbedded Visual Basic 3.0

```
Function Axcpi01GetLastError
(
   ByVal wLogSocket As Integer  '論理ソケット指定
) As Long
```

■形　式　Visual Basic .NET 2003

```
Function Axcpi01GetLastError
(
   ByVal wLogSocket As Short   '論理ソケット指定
) As Integer
```

■形　式　Visual Basic 2005

```
Function Axcpi01GetLastError
(
   ByVal wLogSocket As Short   '論理ソケット指定
) As Integer
```

■入　力　wLogSocket

対象のデバイスを論理ソケット番号で指定。

■戻り値　エラーコード

■解　説　もっとも最近起こったエラーのコードを取得します。
デバイスに依存しないエラーは wLogSocket の番号に関わらず取得されます。
エラー情報はアプリケーション毎に管理されます。
エラー情報はアプリケーションが DLL にアタッチした時に
AXCPI01_SUCCESS で初期化されます。

７－４．定義

７－４－１．エラーコード

```
#define AXCPI01_SUCCESS                 0  // 異状なし（正常終了）
#define AXCPI01_ERR_SYSTEM              1
                                    // Windows の GetLastError() をコールしてください
                                    // 正常にドライバがロードされていない可能性があります
#define AXCPI01_ERR_NO_DEVICE           2  // 使用可能なデバイスがありません
#define AXCPI01_ERR_IN_USE              3  // 指定のデバイスは使用中です
#define AXCPI01_ERR_INVALID_SOCKET      4  // 無効な論理ソケットです
#define AXCPI01_ERR_RESOURCE            5  // リソースエラー
#define AXCPI01_ERR_INVALID_PORT        6  // 不正なポートを要求しました
#define AXCPI01_ERR_INVALID_ARGUMENT    7  // 不正な引数を要求しました
```

エラー内容は新たなエラーが発生するまで前回のものが保持されます。

７－４－２．定数

```
#define AXCPI01_SOCKET_AUTO             ((WORD) ~0U)
#define AXCPI01_MAX_SOCKETS             16

#define AXCPI01_PORT_DATA_L             0x0  // 入出力ポート Low
#define AXCPI01_PORT_DATA_H             0x1  // 入出力ポート High
#define AXCPI01_PORT_IO_SELECT_L        0x2  // 入出力選択ポート Low
#define AXCPI01_PORT_IO_SELECT_H        0x3  // 入出力選択ポート High
#define AXCPI01_PORT_OUTPUT_MODE_L      0x4  // 出力モード設定ポート Low
#define AXCPI01_PORT_OUTPUT_MODE_H      0x5  // 出力モード設定ポート High
#define AXCPI01_PORT_PUPD_SELECT_L      0x6  // PullUp/PullDown 選択ポート Low
#define AXCPI01_PORT_PUPD_SELECT_H      0x7  // PullUp/PullDown 選択ポート High
#define AXCPI01_PORT_IRQ_L              0x8  // 割込み設定ポート Low
#define AXCPI01_PORT_IRQ_H              0x9  // 割込み設定ポート High
```

７－５．メッセージ

ハードウェア割り込み等のイベントは､メッセージによってアプリケーションに通知されます。

Axcpi01Create 関数の uWMBase 引数に下記の値を加えた ID のメッセージが送出されます。

割り込み発生後は、デバイスドライバ内で「IRQ Register（割り込み設定レジスタ)」の IRQCLS1,IRQCLS2 に“ 0 ”を書き込み、割り込みをクリアしています。

IRQ1 の割り込みが発生した場合には、IRQCLS1 に“ 0 ”を書き込み、IRQ2 の割り込みが発生した場合には、IRQCLS2 に“ 0 ”を書き込みしています。

その後 IRQCLS1,IRQCLS2 に“ 1 ”を書き込み、初期化しています。

「IRQ Register」の各ビットの詳細につきましては、ユーザーズマニュアルを参照してください。

注意！)

- デバイスドライバ内で割り込みをクリアしているので、IRQIN1,IRQIN2 で割り込み入力の状態を読み出すことはできません。
- IRQCLS1,IRQCLS2 に値を書き込むと､割り込み処理が正常に動作しなくなる可能性があります。IRQ Register は IRQEN1,IRQEN2 以外は操作しないでください。
- Pocket PC 2002、Pocket PC 2003 および Windows Mobile 5.0 の場合、割り込みの即時性に若干問題がございます。
- 開発言語が eMbedded Visual Basic 3.0（以下「eVB」）の場合、eVB の制約上アプリケーション側でメッセージを受け取ることができません。割り込みのステータス取得は API 関数 Axcpi01GetIRQStatus を使用してください。使用方法につきましては、eVB のサンプルソースを参照してください。

```
#define AXCPI01_WM_OFS_CARD_REMOVAL   0   // カード抜去メッセージ
#define AXCPI01_WM_OFS_IN             1   // 外部入力割り込みメッセージ
```

メッセージハンドラへの引数は以下のようになります。

wParam

メッセージを送出する要因となったデバイスが存在する論理ソケット番号と、「AXCPI01_WM_OFS_IN」メッセージの場合は、発生した割り込み番号の情報が格納されています。

・下位 8 ビット（Low BYTE）

IRQ1 発生時には BIT0 が“ 1 ”となります。
IRQ2 発生時には BIT1 が“ 1 ”となります。

IRQ1 発生時 ： 01h
IRQ2 発生時 ： 02h
IRQ1,IRQ2 発生時 ： 03h

・上位 8 ビット（High BYTE）

論理ソケット番号：00h～0Fh

lParam

製造者コードと製品コードが格納されています。

・下位 16 ビット（Low WORD）

製品コード：7001h

・上位 16 ビット（High WORD）

製造者コード：c005h

７－５－１．AXCPI01_WM_OFS_CARD_REMOVAL + uWMBase

カードが抜去されたことにより発生します。

７－５－２．AXCPI01_WM_OFS_IN + uWMBase

外部入力割り込みで発生します。

製品のお問い合わせについて

◆ お買い求めいただいた製品に対する次のようなお問い合わせは、お求めの販売店又は**株式会社アドテックシステムサイエンス**の各営業所にご連絡ください。

- お求めの製品にご不審な点や万一欠品があったとき
- 製品の修理
- 製品の補充品や関連商品について
- 本製品を使用した特注製品についてのご相談

◆ 技術サポート ――― 技術的な内容のお問い合わせは、「ファックス」「郵送」「E-mail」のいずれかにて、下記までお問い合わせください。また、お問い合わせの際は、内容をできるだけ詳しく具体的にお書きくださるようお願いいたします。

――――― 技術的な内容のお問い合わせ先 ―――――

株式会社 アドテック システム サイエンス　テクニカルサポート
〒240-0005
神奈川県横浜市保土ヶ谷区神戸町 134　YBP ウエストタワー8F
E-mail　support@adtek.co.jp
Fax　045-331-7770

改訂履歴

発行年月日　2003 年 06 月 30 日　第 1 版

発行年月日　2004 年 09 月 07 日　改訂第 2 版
Pocket PC 2003 対応の記述を追加

発行年月日　2007 年 07 月 05 日　改訂第 3 版
Windows Mobile 5.0 対応の記述を追加

AXC-PI01
ソフトウェアマニュアル
Pocket PC 2002／Pocket PC 2003／Windows Mobile 5.0 版

第 3 版発行　2007 年 07 月 05 日
発行所　株式会社 アドテック システム サイエンス
〒240-0005　神奈川県横浜市保土ヶ谷区神戸町 134
YBP ウエストタワー8F
Tel　045-331-7575 ㈹　　Fax　045-331-7770

AXC-002-070705