

AXC-006-050331

PLUG MAGIC シリーズ プログラマブルロジック搭載 CF カード

AXC-PL01

ソフトウェアマニュアル

Pocket PC 2003 版

— 目 次 —

| | |
|----------------------------------|----|
| 1. はじめに..... | 1 |
| 2. 動作環境..... | 1 |
| 3. ファイル一覧..... | 2 |
| 4. ドライバ..... | 3 |
| 4-1. インストール前の準備..... | 3 |
| 4-2. インストール..... | 3 |
| 4-3. アンインストール..... | 6 |
| 4-4. 注意事項..... | 7 |
| 5. 動作チェックソフト..... | 8 |
| 5-1. 概要..... | 8 |
| 5-2. インストールとアンインストール..... | 8 |
| 5-2-1. インストール前の準備..... | 8 |
| 5-2-2. インストール..... | 8 |
| 5-2-3. アンインストール..... | 11 |
| 5-3. アプリケーションの起動と終了..... | 13 |
| 5-3-1. 起動前の準備..... | 13 |
| 5-3-2. 起動..... | 13 |
| 5-3-3. 終了..... | 13 |
| 5-4. メイン画面..... | 14 |
| 5-4-1. ベースアドレス..... | 15 |
| 5-4-2. Read/Write データ (BIN)..... | 16 |
| 5-4-2. Read/Write データ (HEX)..... | 17 |
| 5-4-3. Read コマンドボタン..... | 18 |
| 5-4-4. Write コマンドボタン..... | 19 |
| 5-4-5. SET コマンドボタン..... | 20 |
| 5-4-6. RESET コマンドボタン..... | 21 |
| 5-5. 「情報」メニュー..... | 22 |
| 5-5-1. 「バージョン情報」..... | 23 |
| 5-5-2. 「リソース情報」..... | 23 |
| 6. AXC-PI01 モード動作チェックソフト..... | 24 |
| 6-1. 概要..... | 24 |
| 6-2. インストールとアンインストール..... | 24 |
| 6-2-1. インストール前の準備..... | 24 |
| 6-2-2. インストール..... | 25 |
| 6-2-3. アンインストール..... | 28 |
| 6-3. アプリケーションの起動と終了..... | 30 |
| 6-3-1. 起動前の準備..... | 30 |

| | |
|--------------------------------------|----|
| 6-3-2. 起動 | 30 |
| 6-3-3. 終了 | 30 |
| 6-4. メイン画面 | 31 |
| 6-4-1. 「Data Register」 | 32 |
| 6-4-2. Log | 33 |
| 6-5. 「選択」メニュー | 34 |
| 6-5-1. 「I/O Select Register」 | 35 |
| 6-5-2. 「Output Mode Register」 | 36 |
| 6-5-3. 「PU/PD Select Register」 | 37 |
| 6-5-4. 「IRQ Register」 | 38 |
| 6-6. 「表示」メニュー | 39 |
| 6-6-1. 「入力データ」 | 39 |
| 6-6-2. 「割り込みメッセージ」 | 39 |
| 6-6-3. 「エラーメッセージ」 | 39 |
| 6-7. 「情報」メニュー | 40 |
| 6-7-1. 「リソース」 | 40 |
| 6-7-2. 「バージョン」 | 40 |
| 6-8. その他..... | 41 |
| 6-8-1. エラーメッセージについて | 41 |
| 6-8-2. 注意事項 | 41 |
| 7. サンプルソース | 42 |
| 8. API 仕様..... | 43 |
| 8-1. 概要 | 43 |
| 8-2. プログラム構成 | 43 |
| 8-3. API リファレンス | 44 |
| 8-3-1. Axcpl01GetVersion..... | 44 |
| 8-3-2. Axcpl01Create | 46 |
| 8-3-3. Axcpl01Close | 49 |
| 8-3-4. Axcpl01GetResource | 50 |
| 8-3-5. Axcpl01InPortB..... | 52 |
| 8-3-6. Axcpl01OutPortB..... | 54 |
| 8-3-7. Axcpl01InPortW..... | 56 |
| 8-3-8. Axcpl01OutPortW | 58 |
| 8-3-9. Axcpl01GetIRQStatus | 60 |
| 8-3-10. Axcpl01GetLastError | 62 |
| 8-4. 定義 | 63 |
| 8-4-1. エラーコード | 63 |
| 8-4-2. 定数 | 63 |
| 8-5. メッセージ | 64 |

| | |
|--|----|
| 8-5-1. AXCPL01_WM_OFS_CARD_REMOVAL + uWMBase | 65 |
| 8-5-2. AXCPL01_WM_OFS_IN + uWMBase..... | 65 |
| 製品のお問い合わせについて..... | 66 |
| 改訂履歴 | 67 |

1. はじめに

本マニュアルでは、弊社の PLUG MAGIC シリーズ プログラマブルロジック搭載 CF カード「AXC-PL01」を Pocket PC 2003 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 2003 搭載機

OS : Microsoft Pocket PC 2003 Software

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。

| | | |
|----|----------------------|--------------------------------|
| ¥ | | |
| ├─ | PocketPC2003 | PocketPC2003 用ファイルフォルダ |
| ├─ | Driver | ドライバファイルフォルダ |
| ├─ | DrvPLSetup.exe | インストール実行ファイル |
| ├─ | DrvPLSetup.ini | インストール設定ファイル |
| ├─ | Axcpldrv.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 |
| ├─ | SetupPL.exe | インストール実行ファイル |
| ├─ | SetupPL.ini | インストール設定ファイル |
| ├─ | Axcpl01.cab | インストール cab ファイル |
| ├─ | Source | ソースファイルフォルダ |
| ├─ | Axcpl01.vcw | プロジェクトワークスペース |
| ├─ | Axcpl01.vcp | プロジェクトファイル |
| ├─ | Axcpl01.rc | リソースファイル |
| ├─ | *.cpp | C++ソースファイル |
| ├─ | *.h | C ヘッダーファイル |
| ├─ | res | リソースフォルダ |
| ├─ | Axcpl01.ico | アイコン |
| ├─ | Axcpl01.rc2 | リソースファイル |
| ├─ | Cmdbar.bmp | ビットマップイメージ |
| ├─ | Axcpi01 | AXC-PI01 モード動作チェックアプリケーションフォルダ |
| ├─ | SetupPL_pi01mode.exe | インストール実行ファイル |
| ├─ | SetupPL_pi01mode.ini | インストール設定ファイル |
| ├─ | Axcpl_pi01mode.cab | インストール cab ファイル |
| ├─ | 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 |
| ├─ | Axcpl01.h | DLL 定義ヘッダ |
| ├─ | Axcpl01w.h | ラッパー関数ヘッダ |
| ├─ | Axcpl01w.c | ラッパー関数 |
| ├─ | Axcpl01s.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 |
| ├─ | Axcpl01.vb | DLL 定義 |
| ├─ | Axcpl01w.vb | ラッパー関数 |
| ├─ | Axcpl01s.vb | ラッパー関数使用例 |
| ├─ | buildvbnet.txt | サンプルソース構築例 |
| ├─ | axcpl01_ppc.pdf | ソフトウェアマニュアル |

4. ドライバ

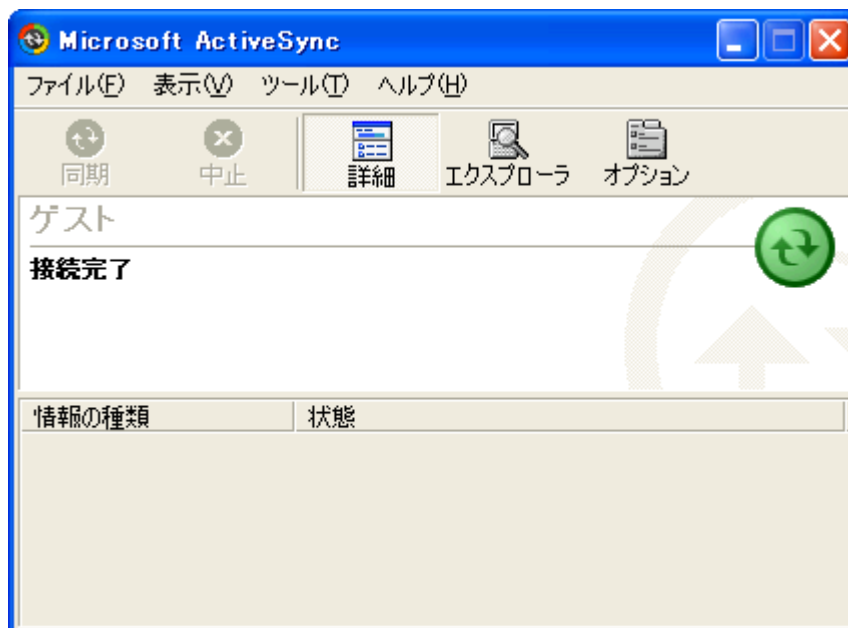
本製品をご使用になる前に、ドライバをインストールする必要があります。
インストールに必要なファイルは、PocketPC2003 フォルダに収められています。

4-1. インストール前の準備

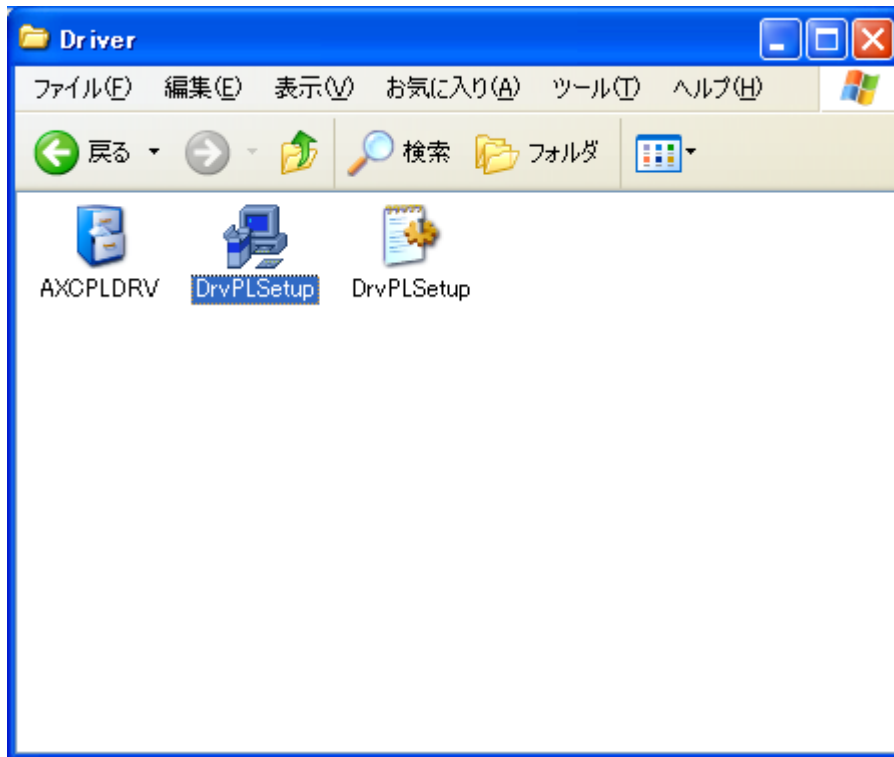
PDA に同梱されている通信ソフト「Microsoft ActiveSync」を PC にインストールしてください。PDA へのインストールは、PC 経由にて行います。

4-2. インストール

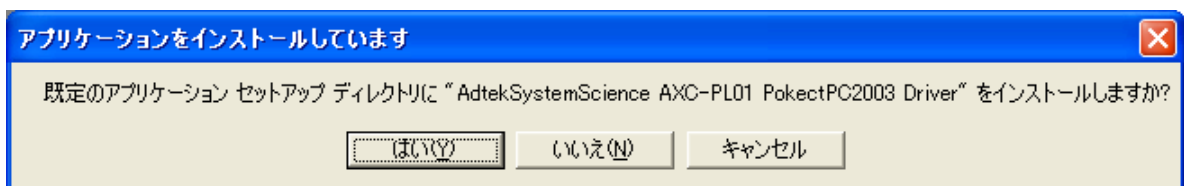
PC と PDA をシリアルケーブルもしくは USB ケーブルにて接続してください。PC にて、「Microsoft ActiveSync」が起動します。



エクスプローラ等を実行し、“AXC-PL01 Driver” のセットアップ用実行ファイル (PocketPC2003¥Driver¥DrvPLSetup.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-PL01 PocketPC2003 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。



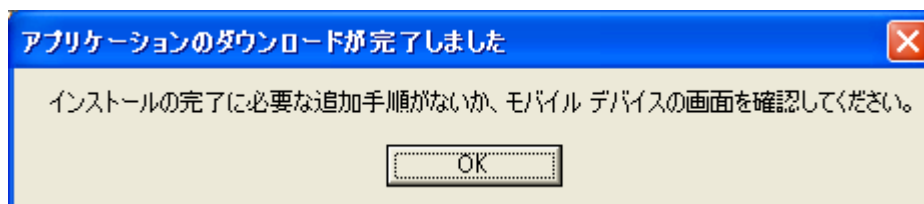
アプリケーションのインストールが開始されます。

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。

- Axcp101.dll ドライバファイル
- Axcp101w.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。

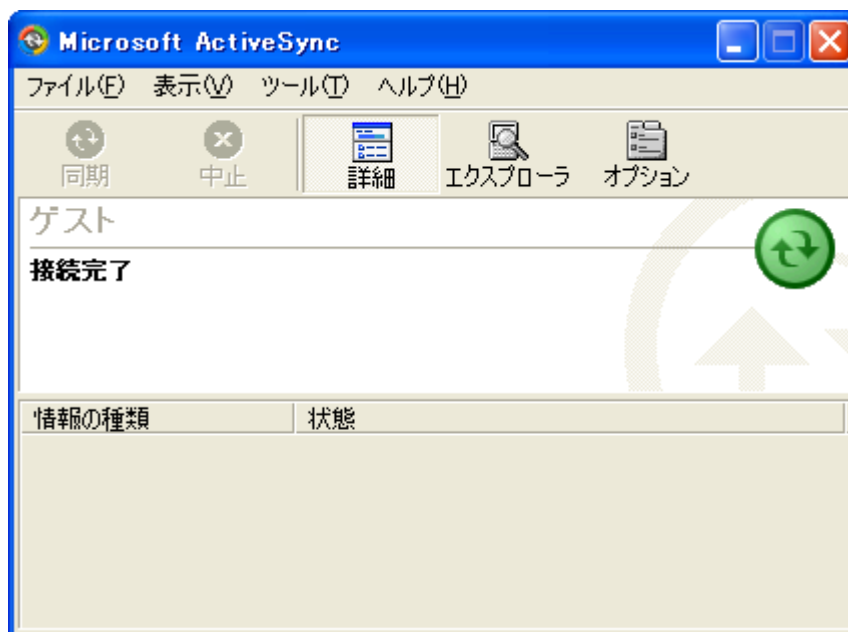


※ ここでの追加手順はありません。このメッセージで終了となります。

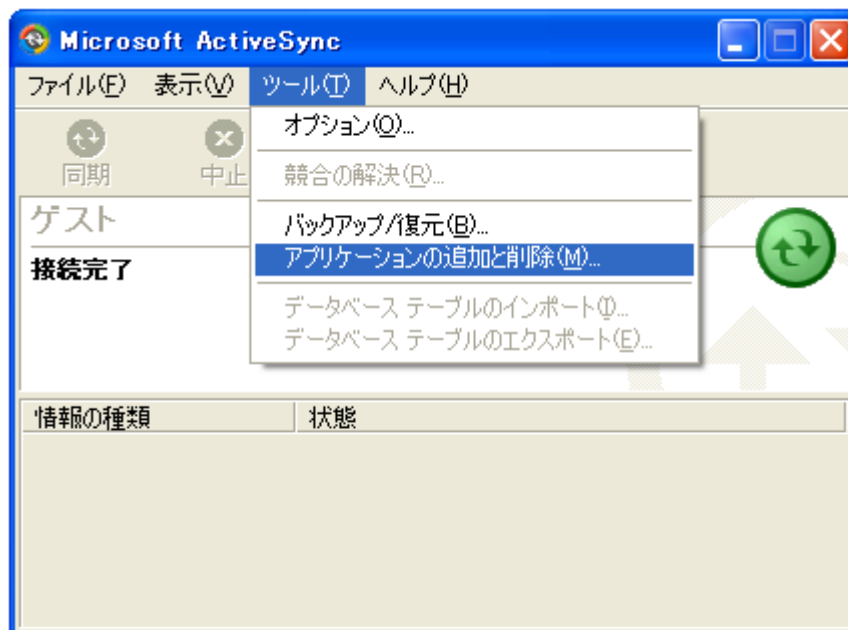
インストール完了後、「AXC-PL01」（以下「デバイス」）を PDA の CompactFlash カードスロットに挿入しますと、OS がデバイスを認識し、挿入サウンドが鳴ります。PDA からデバイスを抜きますと、抜去サウンドが鳴ります。

4-3. アンインストール

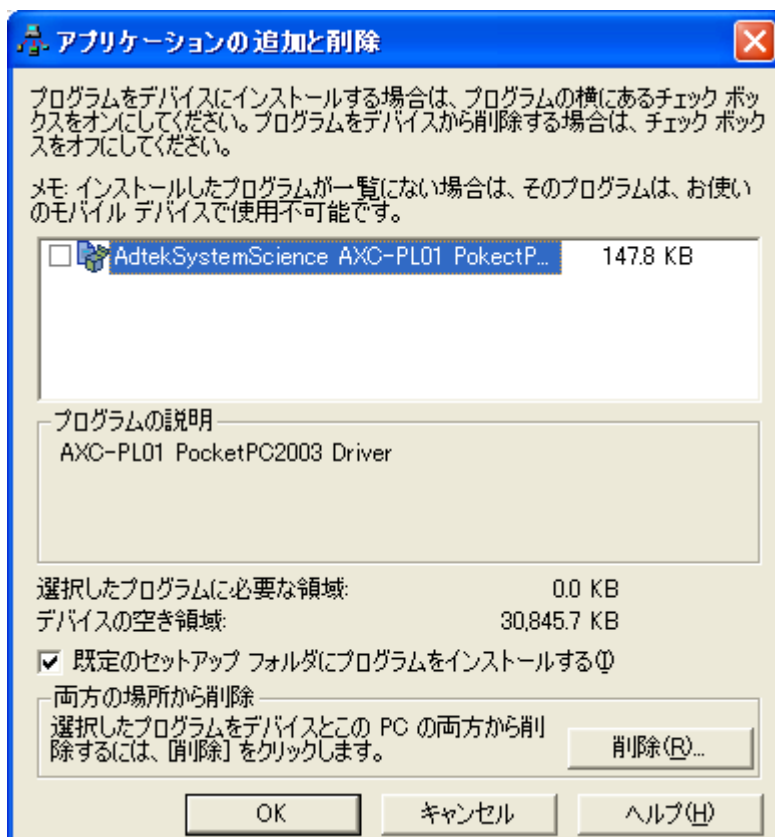
PDA を PC に接続し、「Microsoft ActiveSync」の画面が表示されることを確認してください。



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-PL01 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-PL01 PocketPC2003 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。

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、プログラマブルロジック搭載 CF カード「AXC-PL01」の内部レジスタの Read/Write を行うツールです。「AXC-PL01」の内部 CPLD が、添付のサンプルデータが書き込まれている場合にのみ使用可能です。

また、お客様がご自分で作成された CPLD データで本アプリケーションをお使いになる場合は、オフセットレジスタ等の設定をあわせていただく必要があります。

5-2. インストールとアンインストール

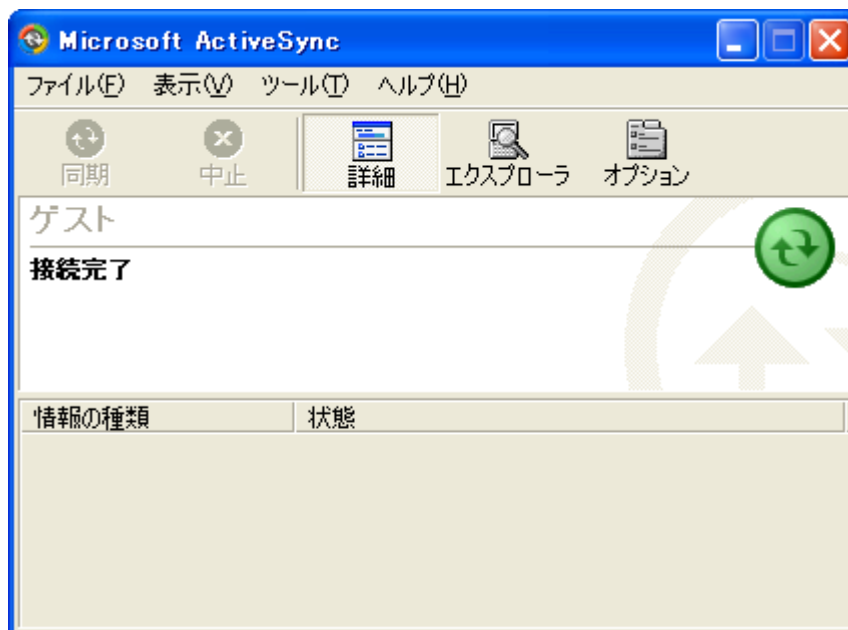
インストールに必要なファイルは、PocketPC2003 フォルダに収められています。

5-2-1. インストール前の準備

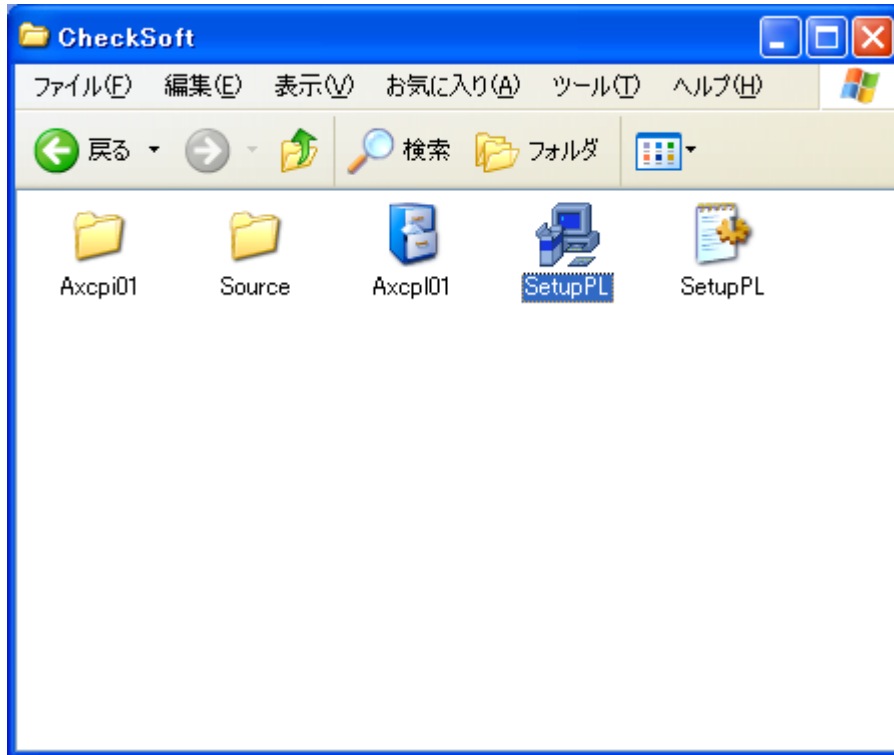
PDA に同梱されている通信ソフト「Microsoft ActiveSync」を PC にインストールしてください。PDA へのインストールは、PC 経由にて行います。

5-2-2. インストール

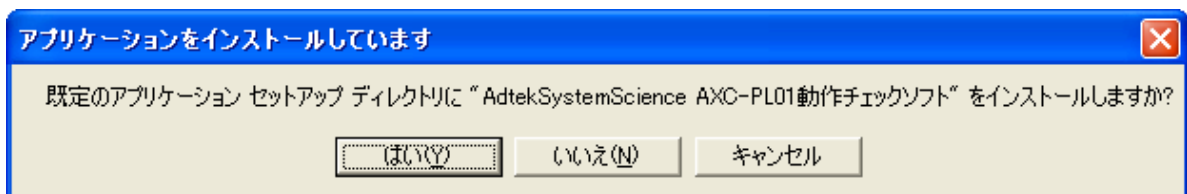
PC と PDA をシリアルケーブルもしくは USB ケーブルにて接続してください。PC にて、「Microsoft ActiveSync」が起動します。



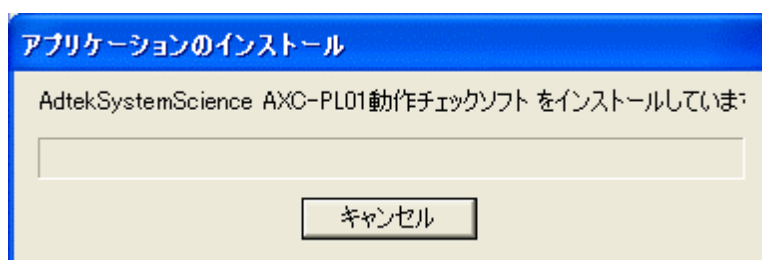
エクスプローラ等を実行し、“動作チェックソフト”のセットアップ用実行ファイル (PocketPC2003¥Checksoft¥SetupPL.exe) を実行します。



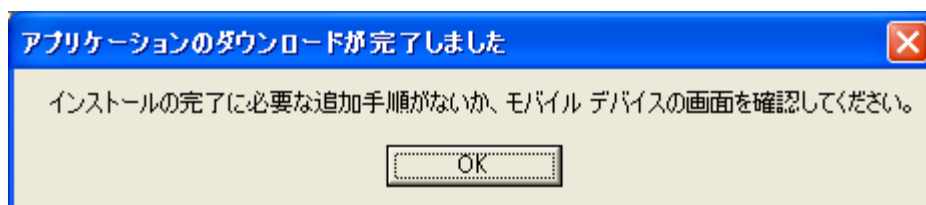
「Microsoft ActiveSync」のウィンドウに、アプリケーションの追加と削除画面が表示されます。[既定のアプリケーション セットアップ ディレクトリに “AdtekSystemScience AXC-PL01 動作チェックソフト” をインストールしますか?] とメッセージが表示されたら、[はい] をクリックしてください。



アプリケーションのインストールが開始されます。



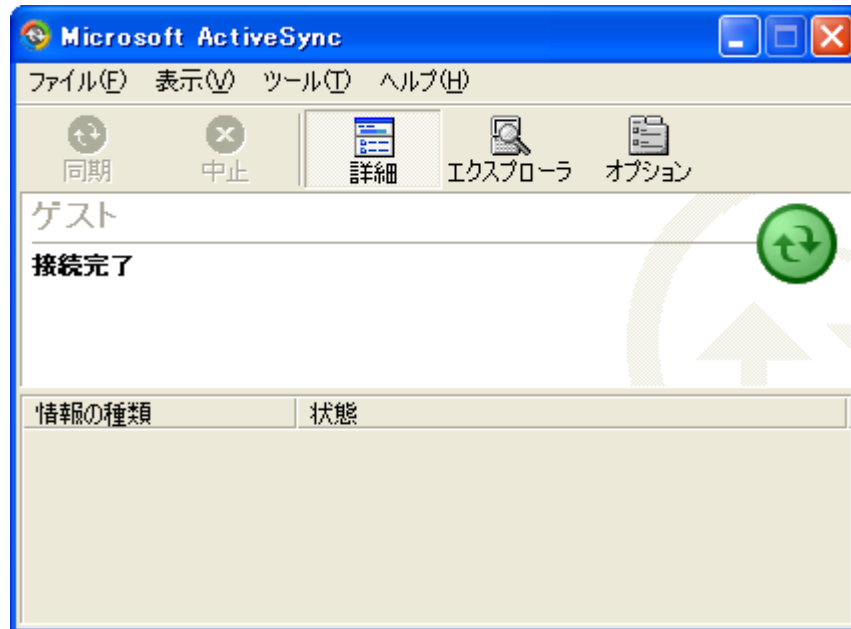
[アプリケーションのダウンロードが完了しました] と表示され、PDA へのインストールは完了します。



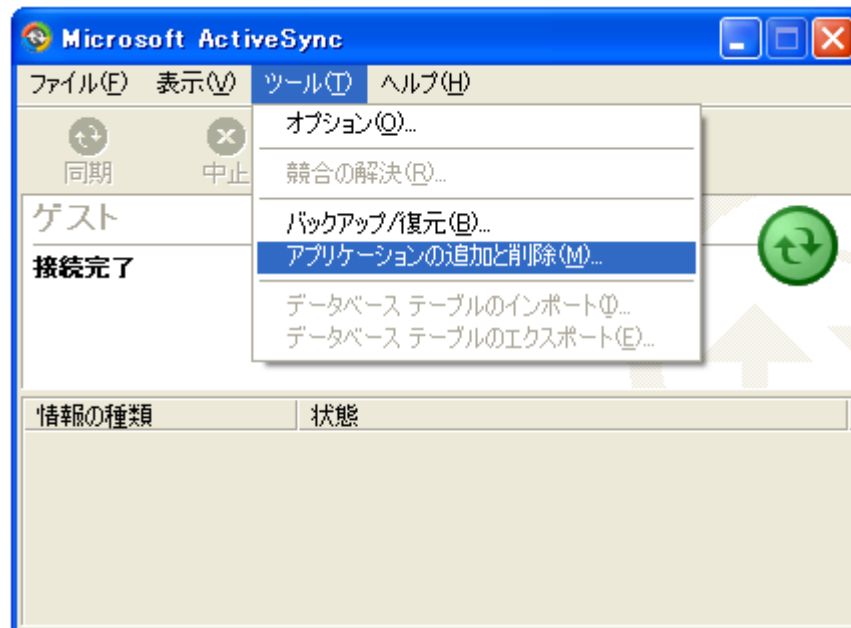
※ ここでの追加手順はありません。このメッセージで終了となります。

5-2-3. アンインストール

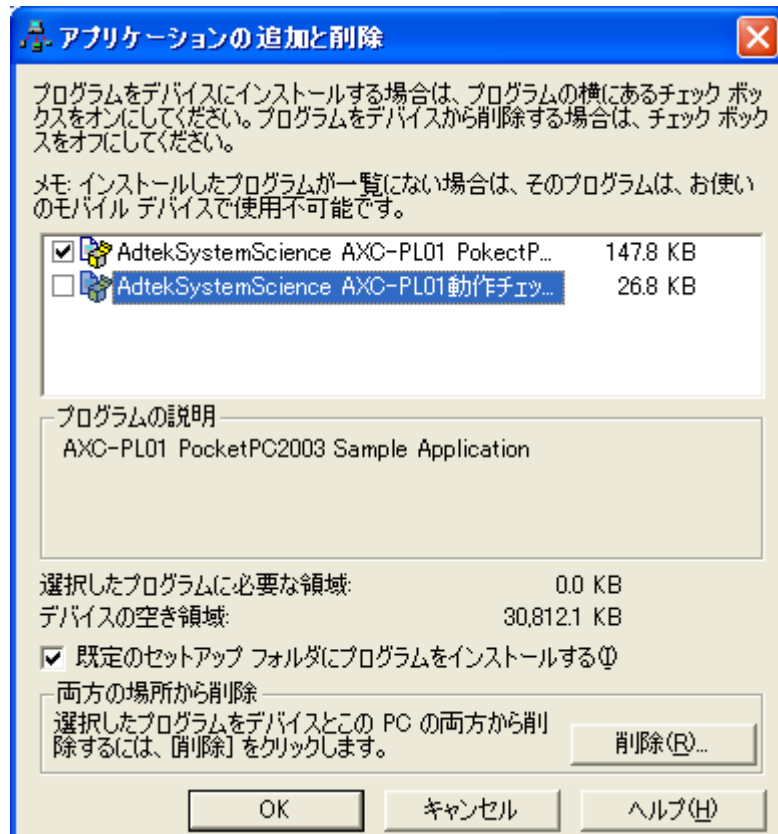
PDA を PC に接続し、「Microsoft ActiveSync」の画面が表示されることを確認してください。



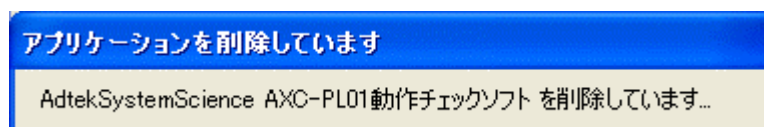
メニューの [ツール] から [アプリケーションの追加と削除] を選択してください。



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-PL01 動作チェッ...] 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-PL01 PocketPC2003 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-PL01」（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-PL01 Checker” 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

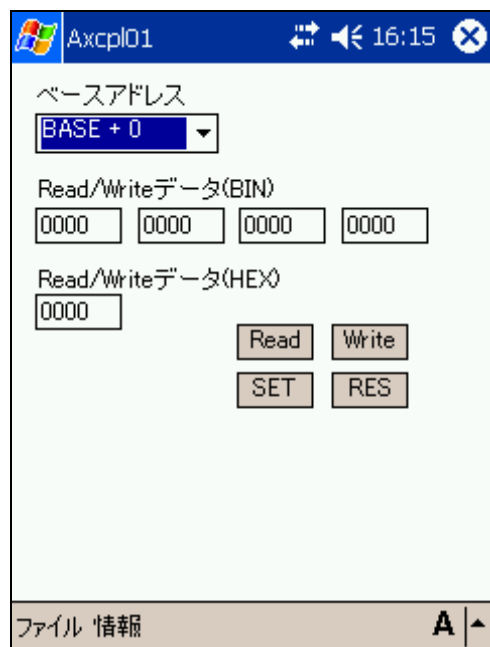
“ファイル” メニュー “アプリケーションの終了” を選択することにより、本アプリケーションを終了することができます。

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。

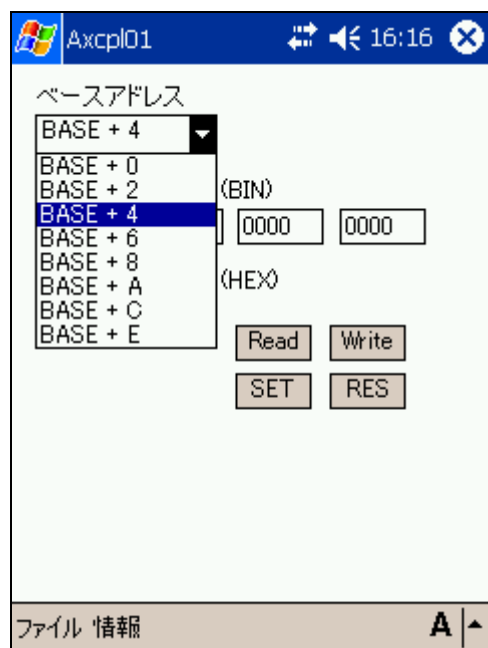
5-4. メイン画面

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. ベースアドレス



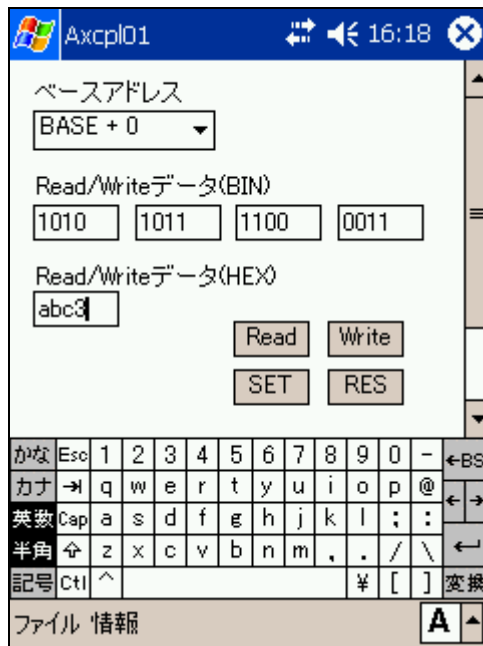
Read/Write する内部レジスタのベースアドレスがドロップダウンリストより選択できます。アドレス範囲は $\text{BASE} + 0\text{h}$ 、 $\text{BASE} + 2 \sim \text{BASE} + \text{Eh}$ までの 8 アドレスです。初期値は $\text{BASE} + 0$ です。

5-4-2. Read/Write データ (BIN)



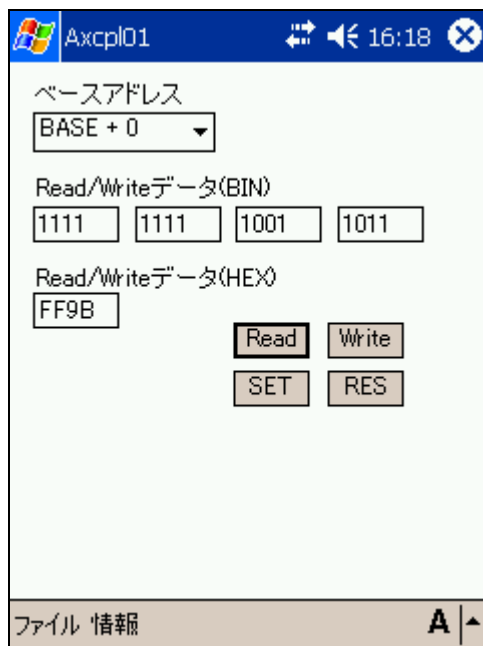
指定されたベースアドレスに書き込む 16 ビットのデータを指定します。2 進数で入力可能です。また、ベースアドレスから読み出された 16 ビットのデータが表示されます。初期値は全ビット“0”です。

5-4-2. Read/Write データ (HEX)



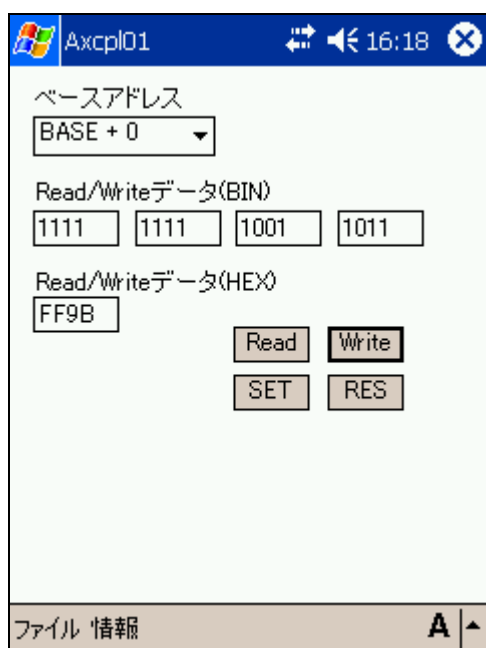
指定されたベースアドレスに書き込む 16 ビットのデータを指定します。16 進数で入力可能です。また、ベースアドレスから読み出された 16 ビットのデータが表示されます。初期値は 0000h です。

5-4-3. Read コマンドボタン



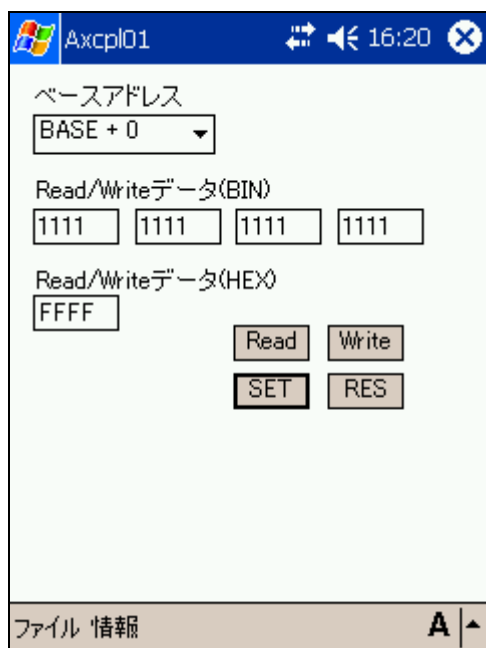
指定されたベースアドレスから 16 ビットのデータを読み込み、Read/Write データ (HEX) と Read/Write データ (BIN) に表示します。実装されていない内部レジスタから読み込んだ場合はエラーとなります。

5-4-4. Write コマンドボタン



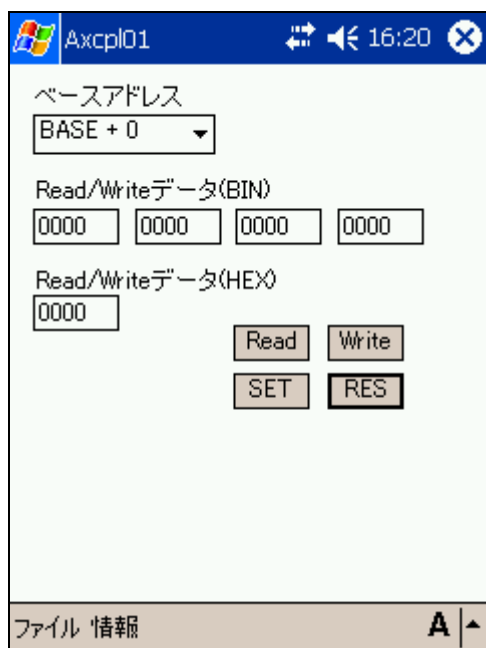
指定されたベースアドレスに 16 ビットのデータ (Read/Write データ (HEX) ならびに Read/Write データ (BIN) に表示されているデータ) を書き込みます。実装されていない内部レジスタに書き込んだ場合はエラーとなります。

5-4-5. SET コマンドボタン



Read/Write データ (HEX) を FFFFh とし、Read/Write データ (BIN) の全ビットを“1”にセットします。

5-4-6. RESET コマンドボタン



Read/Write データ (HEX) を 0000h とし、Read/Write データ (BIN) の全ビットを“0” にリセットします。

5-5. 「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。



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. 「バージョン情報」

以下のドライバおよび DLL のバージョン情報をダイアログボックスに表示します。



5-5-2. 「リソース情報」

以下のシステムリソース情報をダイアログボックスに表示します。



6. AXC-PI01 モード動作チェックソフト

6-1. 概要

プログラマブルロジック搭載 CF カード「AXC-PL01」は、製品出荷時の内部 CPLD が、パラレル入出力アダプタカードとしての機能を持った AXC-PI01 モードとなっています。

本アプリケーションソフトは、「AXC-PL01」の AXC-PI01 モードでの各機能の動作チェックを行うツールです。「AXC-PL01」の内部 CPLD が、製品出荷時（添付のサンプル 3 データが書き込まれている状態）の場合にのみ使用可能です。

AXC-PI01 モードでは 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。

6-2. インストールとアンインストール

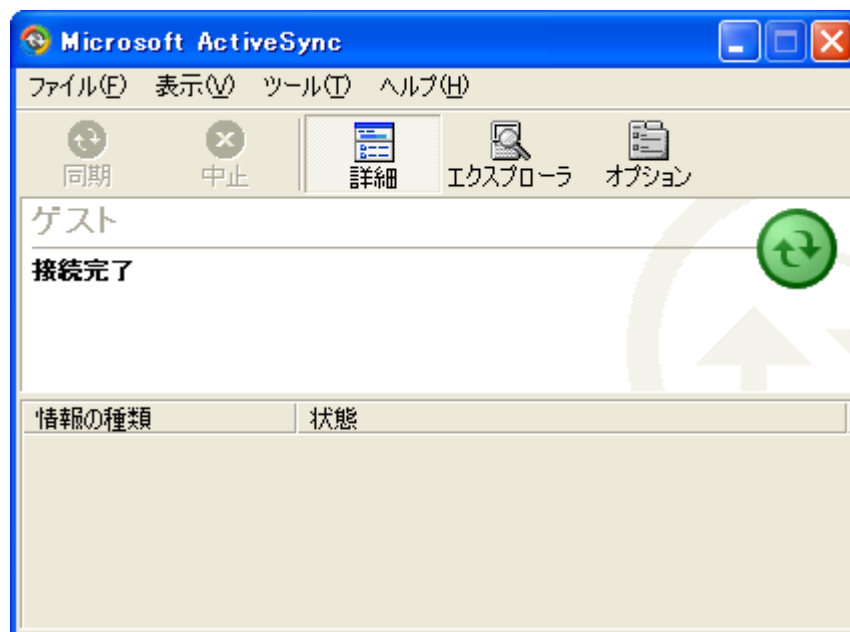
インストールに必要なファイルは、PocketPC2003 フォルダに収められています。

6-2-1. インストール前の準備

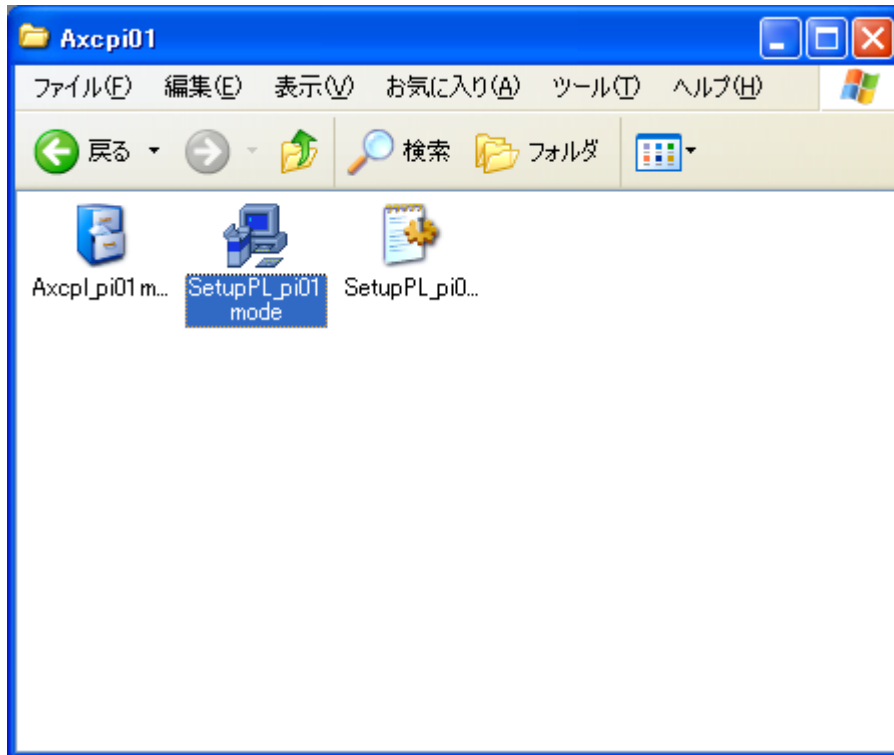
PDA に同梱されている通信ソフト「Microsoft ActiveSync」を PC にインストールしてください。PDA へのインストールは、PC 経由にて行います。

6-2-2. インストール

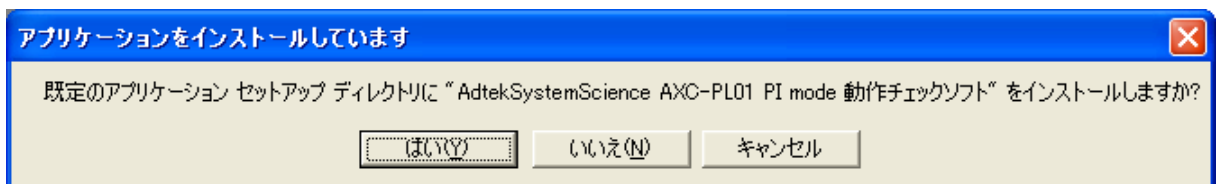
PC と PDA をシリアルケーブルもしくは USB ケーブルにて接続してください。PC にて、「Microsoft ActiveSync」が起動します。



エクスプローラ等を実行し、“AXC-PI01 モード動作チェックソフト”のセットアップ用実行ファイル（PocketPC2003¥CheckSoft¥Axcpi01¥SetupPL_pi01mode.exe）を実行します。



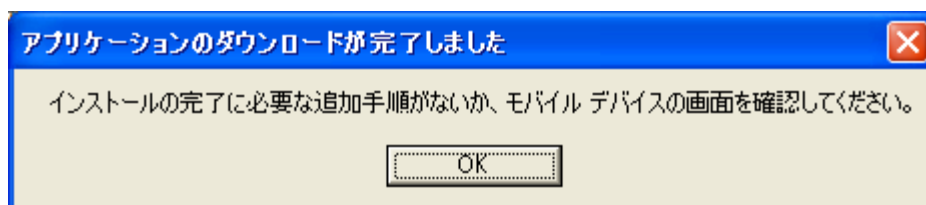
「Microsoft ActiveSync」のウィンドウに、アプリケーションの追加と削除画面が表示されます。[既定のアプリケーション セットアップ ディレクトリに “AdtekSystemScience AXC-PL01 PI mode 動作チェックソフト” をインストールしますか?] とメッセージが表示されたら、[はい] をクリックしてください。



アプリケーションのインストールが開始されます。



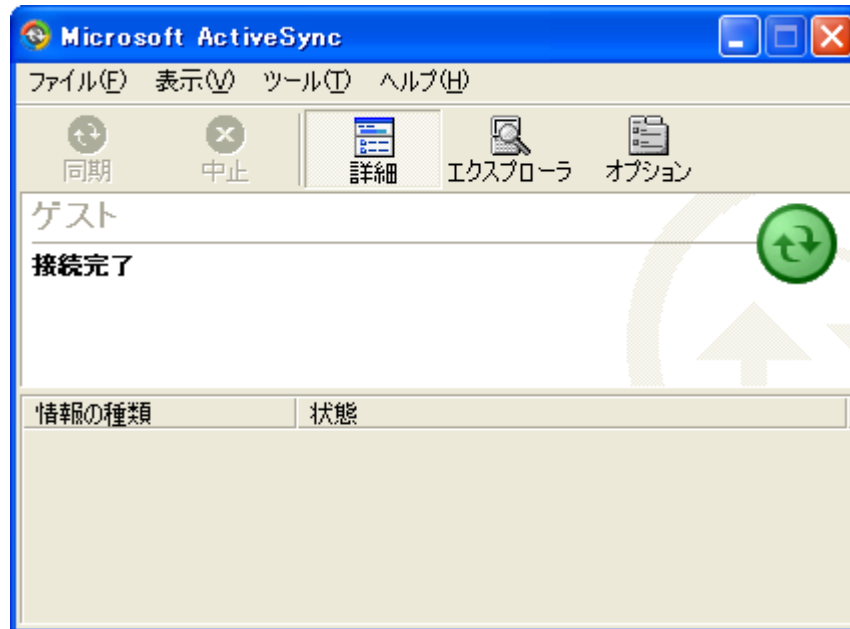
[アプリケーションのダウンロードが完了しました] と表示され、PDA へのインストールは完了します。



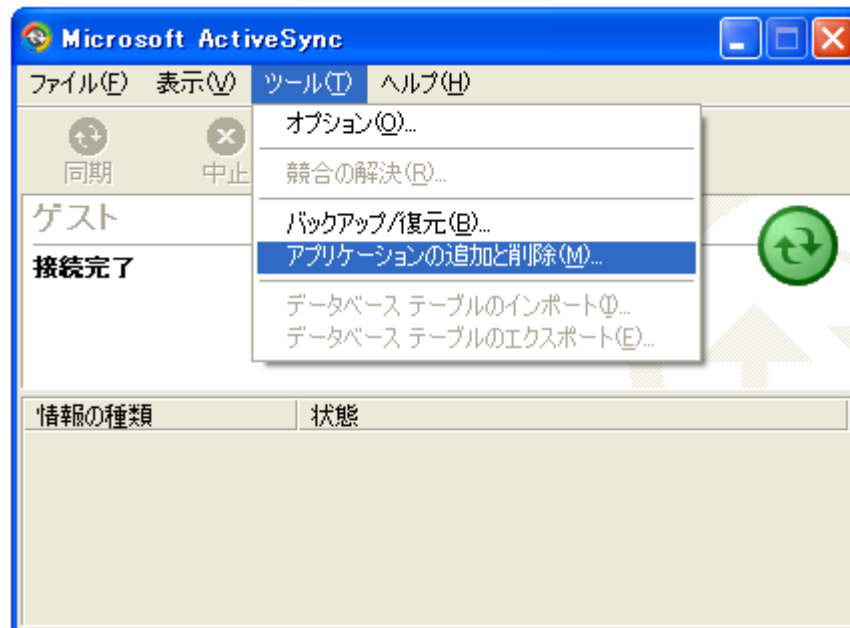
※ ここでの追加手順はありません。このメッセージで終了となります。

6-2-3. 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。



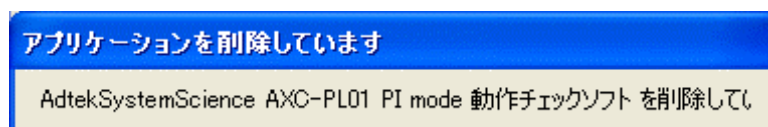
メニューの [ツール] から [アプリケーションの追加と削除] を選択してください。



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-PL01 PI mode...] 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-PL01 PI mode PocketPC2003 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。



6-3. アプリケーションの起動と終了

6-3-1. 起動前の準備

「AXC-PL01」（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。

6-3-2. 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-PL01 PI mode Checker” 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。

6-3-3. 終了

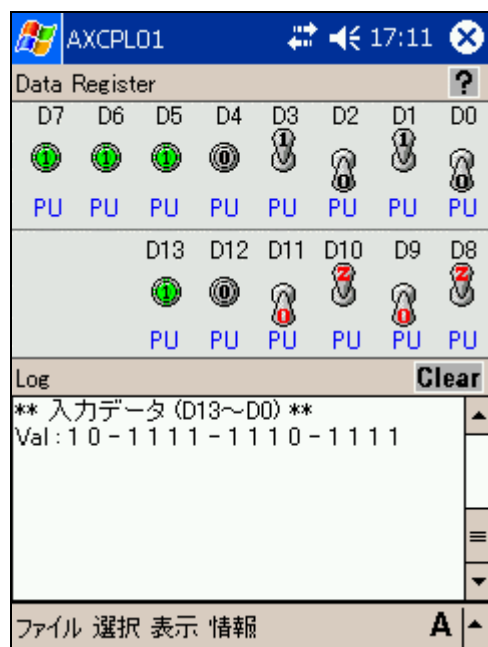
“ファイル” メニュー—“アプリケーションの終了” を選択することにより、本アプリケーションを終了することができます。

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。

6-4. 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。

6-4-1. 「Data Register」

I/O 数 : 14 点 (D13~D0)

属性 : Read/Write

データ送受信レジスタです。

「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。

6-4-2. 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-PL01 Error ****

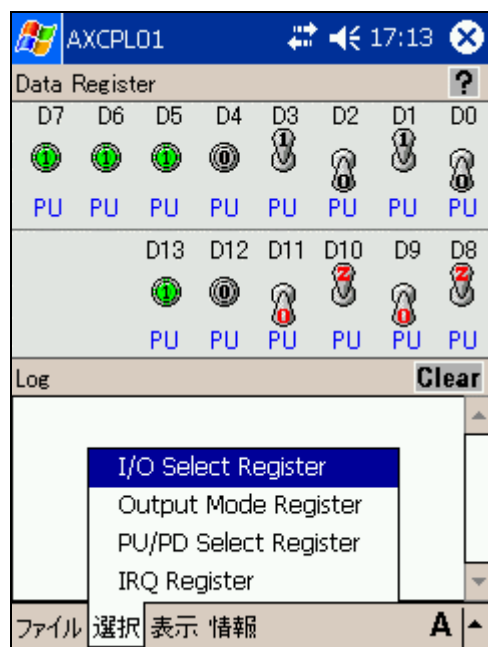
Error code : xxxxxxxxH

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。

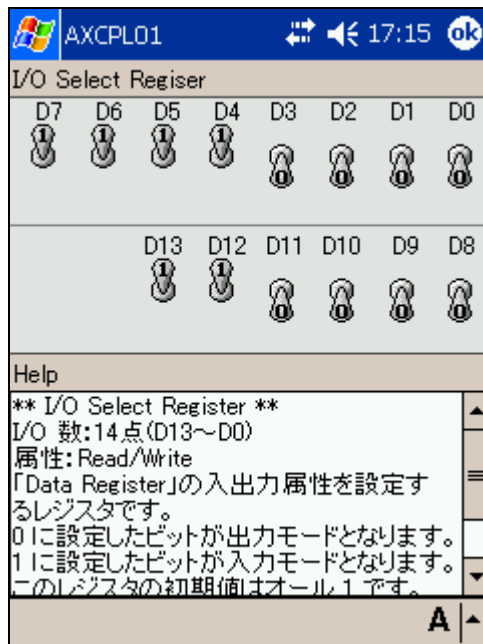
6-5. 「選択」メニュー

“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。

6-5-1. 「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。

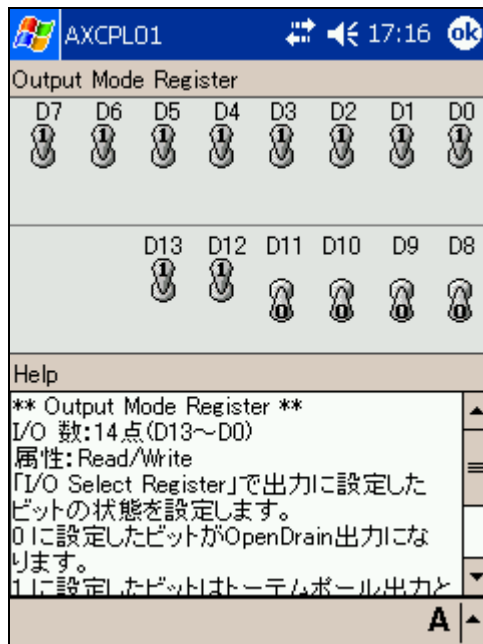
“0” に設定したビットが出力モードとなります。“1” に設定したビットは入力モードとなります。

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。

6-5-2. 「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タ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。

6-5-3. 「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。

6-5-4. 「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 は割り込み許可ビットです。

IRQEN1 を “0” に設定すると、IRQ1 割り込みが有効になり、IRQEN2 を “0” に設定すると、IRQ2 割り込みが有効になります。IRQ1 割り込みは外部 I/O コネクタ 13Pin、IRQ2 割り込みは外部 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。

割り込みは信号の立下りエッジで発生します。

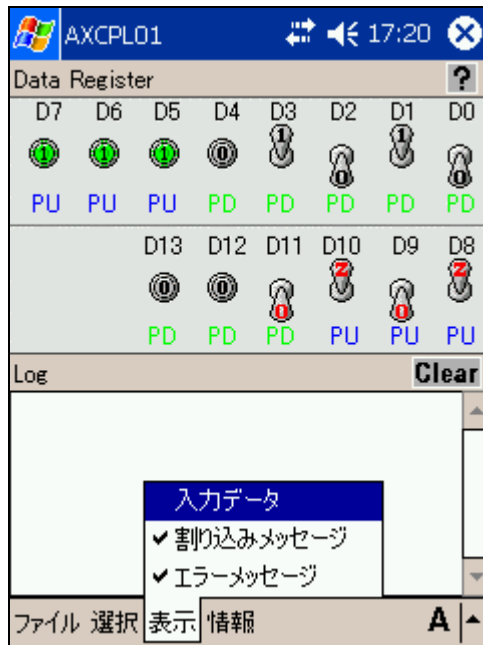
このレジスタの初期値はオール “1” です。

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。

割り込みを使用する場合は、「I/O Select Register」の割り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。

6-6. 「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。



6-6-1. 「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。

6-6-2. 「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。

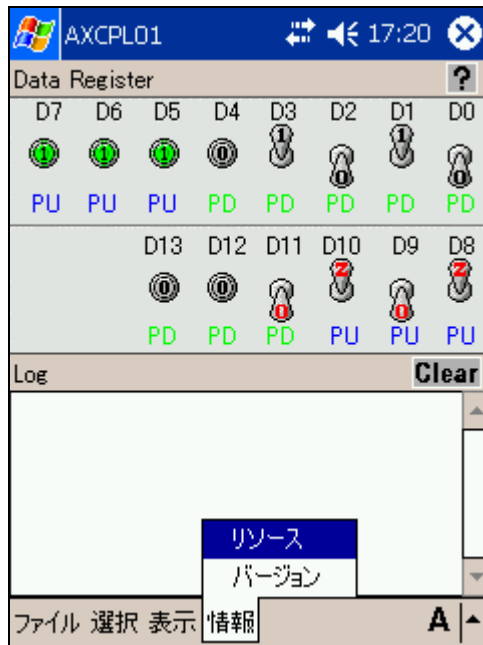
6-6-3. 「エラーメッセージ」

「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。

6-7. 「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。



6-7-1. 「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

6-7-2. 「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

6-8. その他

6-8-1. エラーメッセージについて

表示例) 下記のようにエラー内容と、エラーコードを表示します。

```
** AXG-PL01 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 : 異常なし (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
```

6-8-2. 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「6-3-3. 終了」の項目を参照してください。

7. サンプルソース

サンプルソースには、eMbedded Visual C++ (以下「eVC」) 版および Visual Basic .NET (以下「VB.NET」) 版がございます。

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「8. API 仕様」を参照してください。

<ラッパー関数>

`axcpl01w.*` 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>

`axcpl01s.*` 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・言語に対する十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。

8. API 仕様

8-1. 概要

本章では、AXC-PL01 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。

8-2. プログラム構成

本章で定義される API は、Axcpl01w.dll およびデバイスドライバ Axcpl01.dll により
実現されます。

8-3. API リファレンス

8-3-1. Axcpl01GetVersion

■機能 バージョン情報取得

■形式 eMbedded Visual C++

```
BOOL Axcpl01GetVersion  
(  
    PDWORD pdwDllVersion, //DLLのバージョン  
    PDWORD pdwDrvVersion //DRIVERのバージョン  
);
```

■形式 Visual Basic .NET

```
Function Axcpl01wGetVersion  
(  
    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。
Axcpl01Create 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 AXCPL01_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 (Axcpl01.dll) がない可能性があります。

8-3-2. Axcpl01Create

■機能 デバイスの使用を宣言

■形式 eMbedded Visual C++

```
#define AXCPLO1_SOCKET_AUTO ((WORD) ~0U)
BOOL Axcpl01Create
(
    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 Visual Basic .NET

```
Global Const AXCPLO1_SOCKET_AUTO = &HFFFF
Function Axcpl01Create
(
    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。AXCPL01_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

(Axcpl01GetVersion 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、Axcpl01Close 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、Axcpl01Close API を使用する必要はありません。

■エラー AXCPLO1_ERR_SYSTEM

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

AXCPLO1_ERR_NO_DEVICE

//使用可能なデバイスがありません。

// (AXCPLO1_SOCKET_AUTO を指定した場合)。

AXCPLO1_ERR_IN_USE

//指定のデバイスは使用中です。

AXCPLO1_ERR_INVALID_SOCKET

//無効な論理ソケットです。

// (0~15 及び AXCPLO1_SOCKET_AUTO 以外の値の場合)。

この API が失敗した場合、Axcpl01GetLastError() の wLogSocket には AXCPLO1_SOCKET_AUTO を指定してください。

8-3-3. Axcpl01Close

■機能 デバイス開放

■形式 eMbedded Visual C++
BOOL Axcpl01Close
(
 WORD wLogSocket //論理ソケット指定
);

■形式 Visual Basic .NET
Function Axcpl01Close
(
 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 Axcpl01Create を使いデバイスを開いておく必要があります。

■エラー AXCPLO1_ERR_SYSTEM
//Windows の GetLastError() をコールしてください。
AXCPLO1_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。

8-3-4. Axcpl01GetResource

■機能 リソース情報取得

■形式 eMbedded Visual C++

```
BOOL Axcpl01GetResource  
(  
    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 Visual Basic .NET

```
Function Axcpl01GetResource  
(  
    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 `Axcpl01Create` を使いデバ
イスを開いておく必要があります。

■エラー `AXCPL01_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。

`AXCPL01_ERR_INVALID_ARGUMENT`

//`pdwIOPortBase`, `pdwIOPortLength` が NULL です。

`AXCPL01_ERR_RESOURCE`

//リソース取得に失敗しました。

8-3-5. Axcpl01InPortB

■機能 各ポートに対し1バイト入力

■形式 eMbedded Visual C++

```
BOOL Axcpl01InPortB
(
    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 Visual Basic .NET

```
Function Axcpl01InPortB
(
    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~F)。

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 AXCPL01_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。

AXCPL01_ERR_INVALID_ARGUMENT //pbValue が NULL です。

AXCPL01_ERR_INVALID_PORT

//無効なポート番号 (0~F 以外の値の場合) を要求しました。

8-3-6. Axcpl01OutPortB

■機能 各ポートに対し1バイト出力

■形式 eMbedded Visual C++
BOOL Axcpl01OutPortB
(
 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 Visual Basic .NET
Function Axcpl01OutPortB
(
 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~F)。
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 AXCPL01_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。

AXCPL01_ERR_INVALID_PORT

//無効なポート番号 (0~F 以外の値の場合) を要求しました。

8-3-7. Axcpl01InPortW

■機能 各ポートに対し1ワード入力

■形式 eMbedded Visual C++

```
BOOL Axcpl01InPortW  
(  
    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 Visual Basic .NET

```
Function Axcpl01InPortW  
(  
    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
```

■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,A,C,E)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,B,D,F) は指定できません。

■エラー AXCPL01_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。

AXCPL01_ERR_INVALID_ARGUMENT //pwValue が NULL です。

AXCPL01_ERR_INVALID_PORT

//無効なポート番号 (0,2,4,6,8,A,C,E 以外の値の場合) を要求しました。

8-3-8. Axcpl01OutPortW

■機能 各ポートに対し1ワード出力

■形式 eMbedded Visual C++
BOOL Axcpl01OutPortW
(
 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 Visual Basic .NET
Function Axcpl01OutPortW
(
 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

■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,A,C,E)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,B,D,F)は指定できません。

■エラー AXCPL01_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。

AXCPL01_ERR_INVALID_PORT

//無効なポート番号 (0,2,4,6,8,A,C,E 以外の値の場合) を要求しました。

8-3-9. Axcpl01GetIRQStatus

■機能 割り込みステータス情報取得

■形式 eMbedded Visual C++

```
BOOL Axcpl01GetIRQStatus  
(  
    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 Visual Basic .NET

```
Function Axcpl01GetIRQStatus  
(  
    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 AXCPL01_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。
- AXCPL01_ERR_INVALID_ARGUMENT //pbStatus が NULL です。

8-3-10. Axcpl01GetLastError

■機能 エラーコード取得

■形式 eMbedded Visual C++

```
DWORD Axcpl01GetLastError  
(  
    WORD wLogSocket //論理ソケット指定  
);
```

■形式 Visual Basic .NET

```
Function Axcpl01GetLastError  
(  
    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
AXCPL01_SUCCESS で初期化されます。

8-4. 定義

8-4-1. エラーコード

```
#define AXCPL01_SUCCESS                0 // 異常なし (正常終了)
#define AXCPL01_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 AXCPL01_ERR_NO_DEVICE           2 // 使用可能なデバイスがありません
#define AXCPL01_ERR_IN_USE              3 // 指定のデバイスは使用中です
#define AXCPL01_ERR_INVALID_SOCKET     4 // 無効な論理ソケットです
#define AXCPL01_ERR_RESOURCE            5 // リソースエラー
#define AXCPL01_ERR_INVALID_PORT       6 // 不正なポートを要求しました
#define AXCPL01_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。

8-4-2. 定数

```
#define AXCPL01_SOCKET_AUTO            ((WORD) ~0U)
#define AXCPL01_MAX_SOCKETS           16

#define AXCPL01_PORT_BASE              0x00 // I/O ベースアドレス
#define AXCPL01_PORT_BASE_PLUS1       0x01 // I/O ベースアドレス+1
#define AXCPL01_PORT_BASE_PLUS2       0x02 // I/O ベースアドレス+2
#define AXCPL01_PORT_BASE_PLUS3       0x03 // I/O ベースアドレス+3
#define AXCPL01_PORT_BASE_PLUS4       0x04 // I/O ベースアドレス+4
#define AXCPL01_PORT_BASE_PLUS5       0x05 // I/O ベースアドレス+5
#define AXCPL01_PORT_BASE_PLUS6       0x06 // I/O ベースアドレス+6
#define AXCPL01_PORT_BASE_PLUS7       0x07 // I/O ベースアドレス+7
#define AXCPL01_PORT_BASE_PLUS8       0x08 // I/O ベースアドレス+8
#define AXCPL01_PORT_BASE_PLUS9       0x09 // I/O ベースアドレス+9
#define AXCPL01_PORT_BASE_PLUSA       0x0A // I/O ベースアドレス+A
#define AXCPL01_PORT_BASE_PLUSB       0x0B // I/O ベースアドレス+B
#define AXCPL01_PORT_BASE_PLUSC       0x0C // I/O ベースアドレス+C
#define AXCPL01_PORT_BASE_PLUSD       0x0D // I/O ベースアドレス+D
#define AXCPL01_PORT_BASE_PLUSE       0x0E // I/O ベースアドレス+E
```

8-5. 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。

Axcpl01Create 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 2003 の場合、割り込みの即時性に若干問題がございます。

```
#define AXCPL01_WM_OFS_CARD_REMOVAL    0 // カード抜去メッセージ
#define AXCPL01_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、「AXCPL01_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

8 - 5 - 1. AXCPL01_WM_OFS_CARD_REMOVAL + uWMBase

カードが抜去されたことにより発生します。

8 - 5 - 2. AXCPL01_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 ウェストタワー8F

E-mail support@adtek.co.jp

Fax 045-331-7770

改訂履歴

発行年月日 2005年03月31日 第1版

AXC-PL01
ソフトウェアマニュアル
Pocket PC 2003 版

第1版発行 2005年03月31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 (代) Fax 045-331-7770

不許複製

AXC-006-050331
© 2005 ADTEK SYSTEM SCIENCE Co.,Ltd.