WinDriver™ PCI/PCI Express/PCMCIA

クイックスタートガイド

5 分でデバイス ドライバを作成！

WinDriver を使用する開発者は？

(1) ハードウェア開発者 ― DriverWizard を使用し、新規のハードウェアをすぐにテストできます。

(2) ソフトウェア開発者 ― DriverWizard を使用し、ハードウェアを動作させるデバイス ドライバコードを生成できます。WinDriver でテストとデバッグができます。

サポートしているオペレーティング システムは？

(1) Windows 98/Me/2000/XP/Server 2003/Vista、Windows CE.NET、Windows Embedded CE v6.00、Windows Mobile 5.0/6.0、Linux および Solaris です。Windows NT 4.0 と VxWorks のサポートは、以前のバージョンで提供しています。

(2) WinDriver ベースのドライバは、対応するすべてのオペレーティング システム間でコードによる互換性があります。

どこで詳細および最新情報が入手できますか？

(1) WinDriver の日本語マニュアルおよび 30 日間の無料評価版は、エクセルソフト社の Web サイト (http://www.xlsoft.com/jp/products/download/download.html) から入手できます。

エクセルソフト株式会社

ドライバ開発の 7 つのステップ

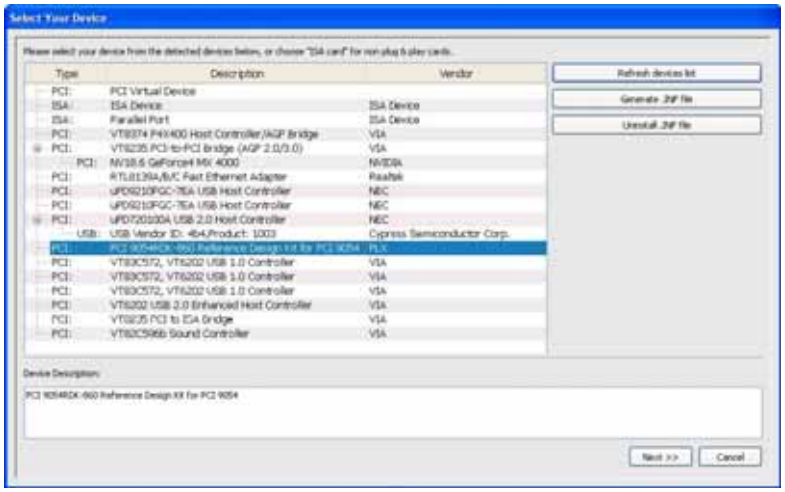
1. セットアップ

(1) PC にハードウェアデバイスを挿入します。
(2) WinDriver をインストールします。

2. デバイスの選択

(1) Windows のスタート メニューから WinDriver - DriverWizard を選択し、DriverWizard を起動します。または、インストール先のディレクトリより、WinDriver/wizard/wdwizard.exe を起動します。
(2) ダイアログ ボックスが表示されるので、New host driver project を選択します。

(3) DriverWizard はマシン上の plug and play カードをすべて表示します。
(4) Plug-and-Play デバイスの場合は、一覧から目的のハードウェア デバイスを選択します。Plug-and-Play デバイス以外 (ISA) デバイスの場合: ISA カード オプションを選択して、対象のデバイスのリソースを定義します。接続されていない PCI デバイス用のコードを生成するには、PCI: PCI Virtual Device オプションを選択します。

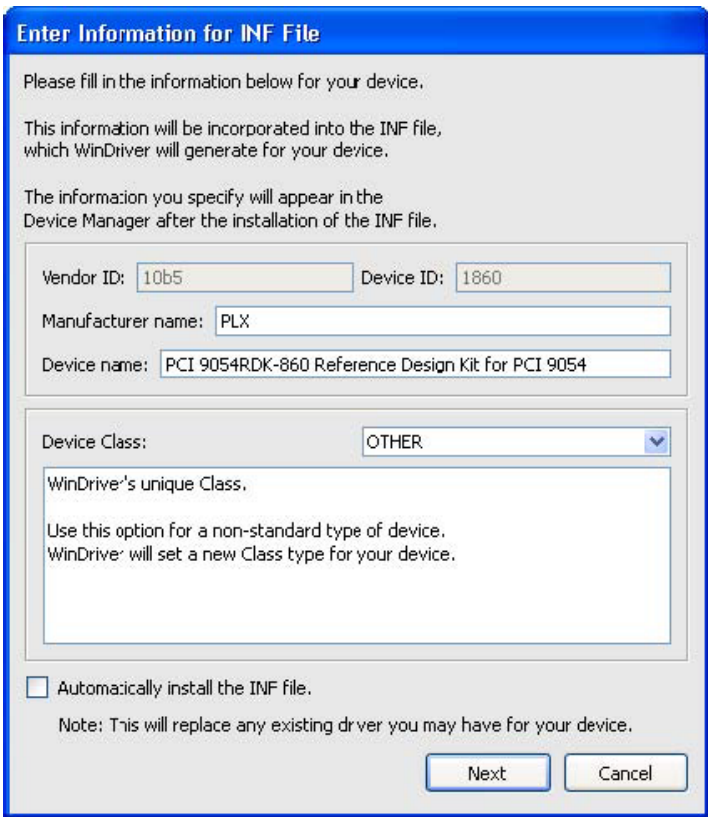


3. 対象の Plug-and-Play デバイスの INF ファイルの生成とインストール (Windows98/Me/2000/XP/Server2003/Vista の場合)

Plug-and-Play Windows OS (Windows 98/Me/2000/XP/Server 2003/Vista) で、Plug-and-Play デバイス (PCI/PCMCIA/CardBus) のドライバを開発する際に、WinDriver を使用して、正しくデバイスのリソースを検出し、デバイスと通信を 0 行うには、対象のデバイスが WinDriver と動作するように INF ファイルをインストールする必要があります。

DriverWizard で自動的に INF ファイルの生成とインストールを行います。
DriverWizard で INF ファイルを生成し、インストールするには、以下の手順で行います。

(1) Wizard の Select Your Device ダイアログで、Generate .INF file ボタンをクリックします。DriverWizard は、対象のデバイスの検出した情報を表示します。
Vendor ID、Device ID、Device Class、Manufacturer name、Device name。
また Manufacturer と Device name、Device Class の情報を編集できます。



(2) Windows 2000/XP/Server 2003/Vista では、DriverWizard の INF 生成ダイアログで Automatically Install the INF file オプションをチェックして、DriverWizard から自動的に INF ファイルをインストールすることができます。Windows 98/Me では、ハードウェアの追加ウィザードまたはハードウェアの更新ウィザードを使用して、手動で INF ファイルをインストールする必要があります。

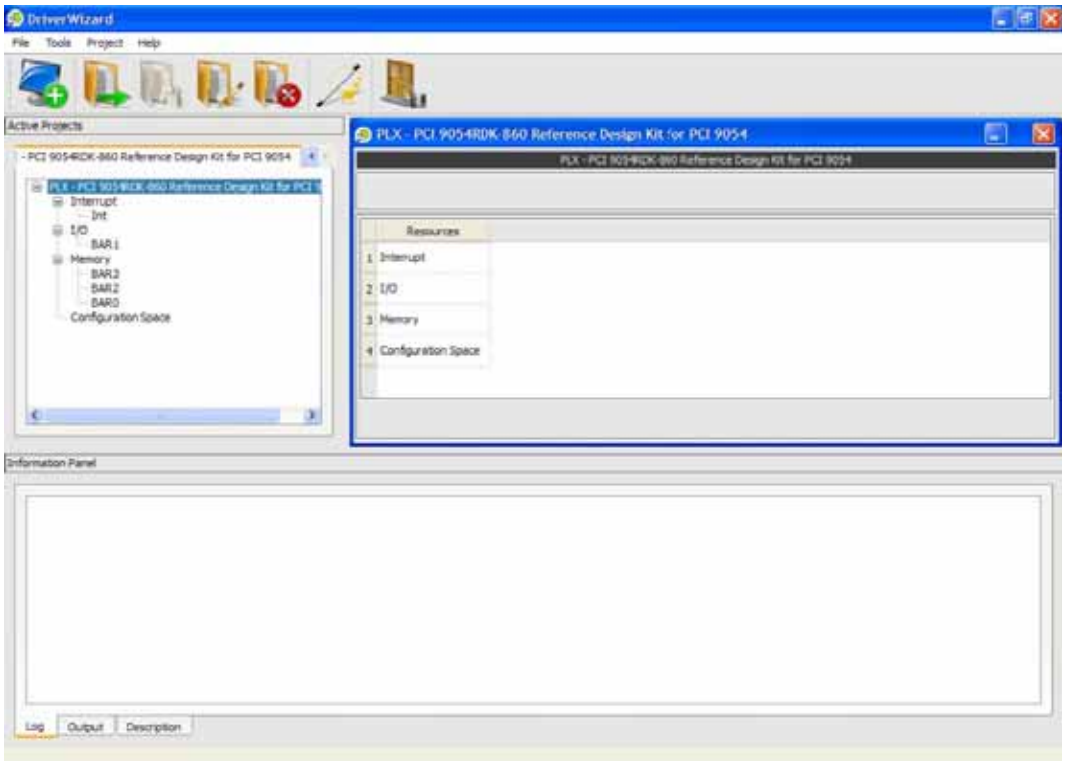
Windows 2000/XP/Server 2003/Vista で、INF ファイルの自動インストールに失敗した場合、DriverWizard がエラーの内容を表示し、その OS 用に手動のインストール手順をお知らせします。 INF 生成ダイアログで、Next をクリックして、INF ファイルを生成し、インストールします (選択した場合)。

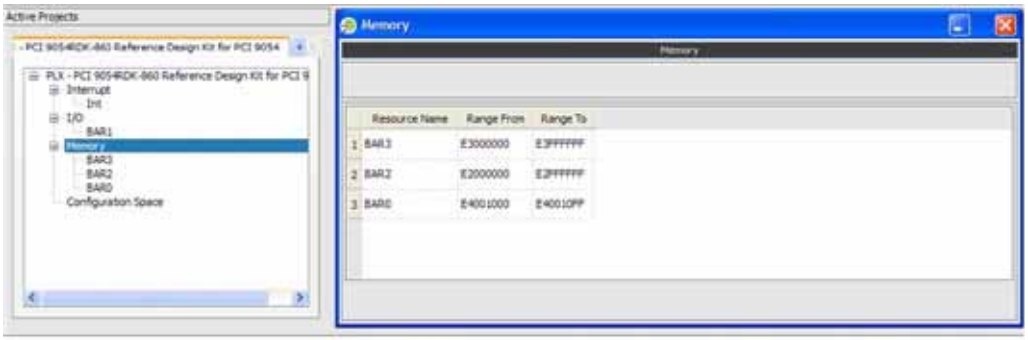
INF ファイルのインストールが終了したら、上記のステップ 2 のリストから対象のデバイスを選択し、開きます。

4. 対象のハードウェアのリソースの検出と定義

DriverWizard は、対象の Plug-and-Play ハードウェアのリソース (I/O 範囲、メモリ範囲、PCI configuration レジストおよび割り込み) を自動的に検出します。対象のデバイスのレジスタの定義など、追加の情報を定義できます。また、同様に定義したレジスタの read/write コマンドと割り込みを割り当てることもできます。

non-Plug-and-Play ハードウェア (ISA) の場合、対象のハードウェアのリソースを手動で定義します。




Active Projects
Memory
Resource Name
Range From
Range To
BAR3
BAR2
BAR0
Configuration Space


Active Projects
Interrupt
Interrupt Name
Interrupt Number
Type
Int
Level Sensitive


Active Projects
Configuration Space
Register
Offset
Size
Access Mode
Data

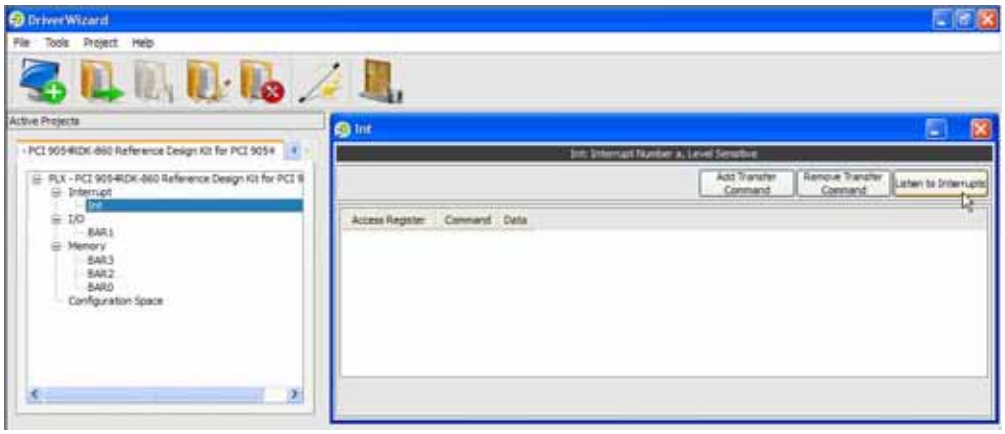

Register Information
Name
Register0
Auto Read
Resource Name
Access Mode
BAR0
Read/Write
Offset
Size
0
32 Bit
OK
Cancel

5. 対象のハードウェアのテスト

デバイス ドライバを記述する前に、ハードウェアが期待通りに動作するかを確認することは重要です。DriverWizard を使用して、対象のハードウェアを診断します。

• I/O ポート、メモリ範囲および定義したレジスタを Read および Write します。

• ハードウェアの割り込みを 'Listen' (確認) します。

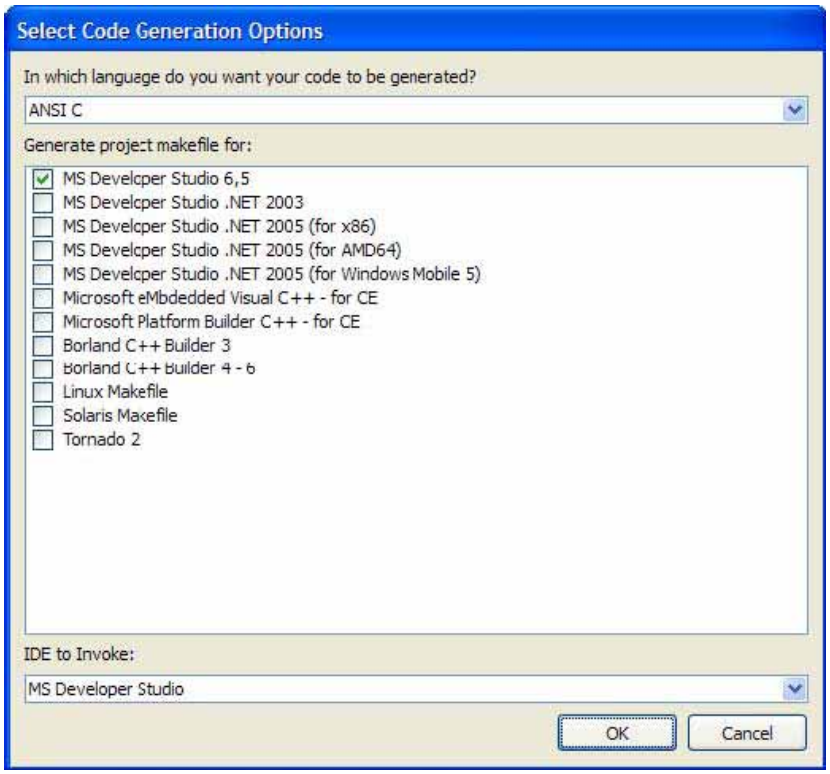


注意: PCI 割り込みなど、レベル センシティブな割り込みの場合、DriverWizard を使用して割り込みの確認をする前に、割り込みステータスのレジスタを定義し、割り込みを検知 (クリア) するための read/write コマンドを割り当てる必要があります。場合によっては OS がクラッシュする事があります。特定の割り込みを検知する情報は、ハードウェア独自の仕様です。

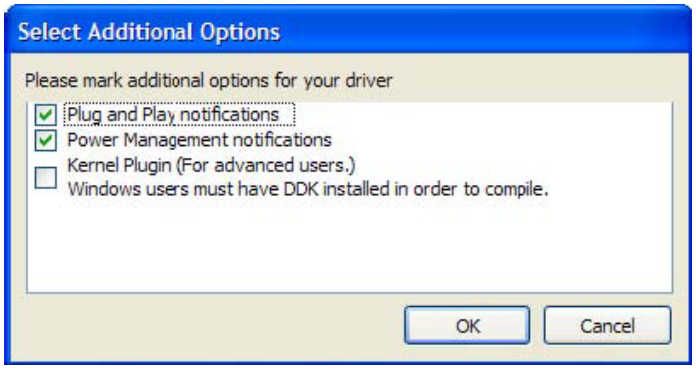
6. ドライバ コードの生成

(1) Generate Code ツールバー アイコンまたは Project | Generate Code メニューのいずれかを選択してコードを生成します。

(2) 以下のように、開発言語を選択して、作成するプロジェクトの開発環境を決めます。



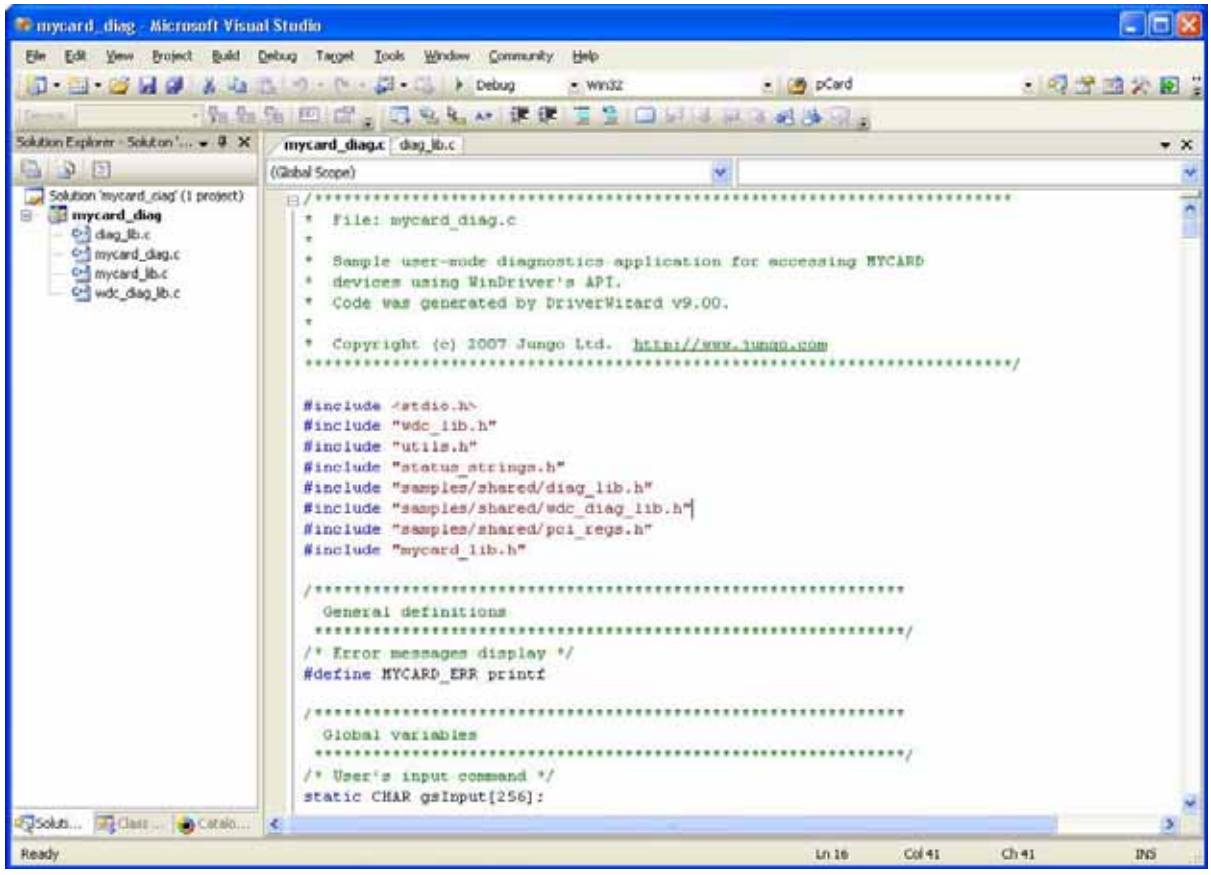
(3) 以下のように、ドライバ コード内から Plug-and-Play と power management イベントを処理するか、また、Kernel PlugIn コードを生成するか、選択します(注意: Kernel PlugIn ドライバをビルドするには、適切な Microsoft DDK がコンピュータにインストールされている必要があります)。



(4) OK をクリックします。DriverWizard が上記のステップ 6.2 で選択した開発環境を起動します。

7. コンパイルと実行

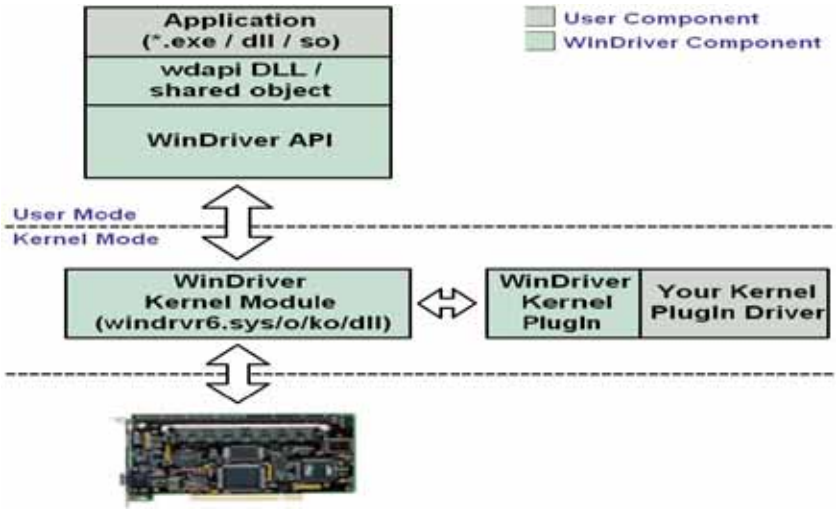
- 以下のコードが生成されます。
  - アプリケーション レベル (およびカーネル) から対象のハードウェアにアクセスする API
  - 対象のハードウェアにアクセスする上記の API を使用するサンプル アプリケーション
  - 選択したビルド環境用のすべての Project/make ファイル
  - 対象のデイバスの INF ファイル (Windows 98/Me/2000/XP/Server 2003/Vista の Plug-and-Play ハードウェア用)
- 選択したコンパイラで DriverWizard が生成した project/make ファイルを使用します。
- サンプルの診断アプリケーションをコンパイルし、実行します。このサンプルはドライバの強力な雛型となります。
- アプリケーションの仕様に応じて、サンプルのアプリケーションを修正します。または、WinDriver に付属の多くのサンプルの一つを使用することもできます。



Q & A:

Q: WinDriver はどのように動作しますか?

A: WinDriver を使用して、対象のデバイスのドライバをユーザーモードで開発します (アプリケーションの一部、または DLL として)。これによって、標準的な開発ツール (MSDEV/Visual C/C++、MSDEV .NET、Borland C++ Builder、Borland Delphi、Visual Basic 6.0、MS eMbedded Visual C++、MS Platform Builder C++、GCC など) を使用して、ドライバの開発およびデバッグが出来るので、開発時間を大幅に短縮できます。
WinDriver を使用して開発したデバイス ドライバ (YourApp.exe) は、標準的な WinDriver 関数を使用する WinDriver のカーネル モジュール (windrvr6/.sys/.o/.ko/.dll) を通して対象のハードウェアにアクセスします。



Q: WinDriver でパフォーマンスの最適化ができますか?

A: ドライバが完成後、パフォーマンスの最適化を行うには、ドライバ コードのパフォーマンスの重要な部分 (割り込みハンドラ、I/O ハンドラなど) を WinDriver の Kernel PlugIn (カーネルモード レベルで起動) に簡単に移行できます。
例えば、割り込みハンドラのコードをユーザーモードで記述します。ユーザーモードでこのコードをデバッグ後、コードを Kernel PlugIn に移行することができます。これによって、カーネル レベルで割り込みハンドラを実行でき、最大限のパフォーマンスを引き出します。
このアーキテクチャで、WinDriver の API を使用して、すべてのドライバ コードをユーザーモードで開発およびデバッグができ、シンプルな Kernel PlugIn のメカニズムで、コードのパフォーマンスの重要な部分のみをカーネル モードに移行できます。

実習:

以下の 5 分間演習で、WinDriver の機能の一部を紹介します。以下の Web サイトから WinDriver の 30 日間無料評価版をダウンロードし、次の演習をお試しください。
http://www.xlsoft.com/jp/products/download/download.html

演習#1: PCI メモリへの読み込みと書き込み

目的: PCI メモリ範囲への読み込みと書き込み、およびレジスタの定義の方法を練習します。

概要: この演習では、DriverWizard を使用して、PCI カードのメモリへ読み込みと書き込みの方法、および同じ処理を実行するアプリケーションの生成方法を説明します。それでは、PCI (または AGP) スクリーン カードへ読み込みと書き込みを行います。

演習ステップ:

Step #1: DriverWizard を起動して、New host driver project を選択します。既に Driver Wizard を起動している場合、New Device Driver Project ツールバー アイコンをクリックするか、File メニューからオプションを選択して、新しいプロジェクトを作成することができます。

Step #2: Plug-and-Play カードのリストから対象のスクリーン カードを選択します。スクリーン カードを見つけるには、カードの vendor 名をリストから選びます。

Step #3: 左のウィンドウの Memory をクリックします。右のウィンドウに対象のカードのメモリ範囲が表示されます。これらのメモリ範囲の一つがスクリーンにマップされています。つまり、メモリ範囲のバイトはスクリーンのピクセルに対応しています (これは通常、Bar 0 です。 最大のメモリ範囲を探します)。 左のウィンドウの関連する BAR を選択し、右のウィンドウの Read / Write Memory ボタンをクリックし、選択した BAR の offset 0 から読み込みます (スクリーンの左上)。ウィンドウをスクリーンの左上に移動し、色を変えて、再度、同じ offset を読み込みます。値が変わったら、これが正しいメモリ範囲です。この offset (FFFFFFFF と 00000000 を交互に試してください) へ書き込むと、ピクセルの色が変わったのが確認できます。

** 注意: 不正なメモリ範囲への書き込みによって、コンピュータがフリーズする可能性があります。

Step #4: スクリーンの左上を表す "TopLeft" というレジスタを定義し (正しいメモリ範囲とオフセット 0)、レジスタから/へ読み込みと書き込みをします。Offset FF の "Somewhere" というレジスタを定義します (スクリーンの他のピクセル)。

Step #5: Generate Code ツールバー アイコン、または Project | Generate Code メニュー オプションを選択して、コードを生成します。Driver Wizard が、対象のハードウェアのリソースにアクセスする関数を作成します。ユーザーモードのアプリケーションから直接これらの関数を呼ぶことができます。DriverWizard では、これらの関数を使用して対象のデバイスにアクセスするサンプル アプリケーションを作成します。

Step #6: サンプル アプリケーションをコンパイルし、実行します。このアプリケーションを使用し、スクリーン カードから読み込みと書き込みを行います。

** 注意: プロジェクトのソースを他の対応するオペレーティング システム (Windows 98/Me/2000/XP/Server 2003/Vista、Windows CE.NET、Windows Embedded CE v6.00、 Windows Mobile 5.0/6.0、Linux および Solaris) へコピーし、再コンパイルして実行することができます。

演習で DriverWizard が生成した API の一部です (screencard が、DriverWizard でコードを生成する際に、ドライバ プロジェクト用に選択した名前です)。

```
<screencard¥_lib.h>

/* SCREENCARD run-time registers */
/* [Values should correlate to the registers' indexes in the gSCREENCARD_Regs array] */
typedef enum {

SCREENCARD_TopLeft, /* TopLeft -This register represents the top left
pixel on the screen */
SCREENCARD_Somewhere, /* Somewhere -This register represents a pixel
somewhere on the screen */
SCREENCARD_REGS_NUM, /* Number of run-time registers */
} SCREENCARD_REGS;

DWORD SCREENCARD_LibInit(void);
DWORD SCREENCARD_LibUninit(void);

WDC_DEVICE_HANDLE SCREENCARD_DeviceOpen(const WD_PCI_CARD_INFO
*pDeviceInfo);
```

```
BOOL SCREENCARD_DeviceClose(WDC_DEVICE_HANDLE hDev);

DWORD SCREENCARD_IntEnable(WDC_DEVICE_HANDLE hDev,

SCREENCARD_INT_HANDLER funcIntHandler);
DWORD SCREENCARD_IntDisable(WDC_DEVICE_HANDLE hDev);
BOOL SCREENCARD_IntIsEnabled(WDC_DEVICE_HANDLE hDev);


DWORD SCREENCARD_EventRegister(WDC_DEVICE_HANDLE hDev,

SCREENCARD_EVENT_HANDLER funcEventHandler);
DWORD SCREENCARD_EventUnregister(WDC_DEVICE_HANDLE hDev);
BOOL SCREENCARD_EventIsRegistered(WDC_DEVICE_HANDLE hDev);

DWORD SCREENCARD_GetNumAddrSpaces(WDC_DEVICE_HANDLE hDev);
BOOL SCREENCARD_GetAddrSpaceInfo(WDC_DEVICE_HANDLE hDev,
SCREENCARD_ADDR_SPACE_INFO *pAddrSpaceInfo);

<screencard_diag.c>

/* -----------------------------------------------SCREENCARD run-time registers information

-----------------------------------------------*/
/* Run-time registers information array */
const WDC_REG gSCREENCARD_Regs[] = {

{ AD_PCI_BAR1, 0x0, WDC_SIZE_8, WDC_READ_WRITE, "TopLeft",
"This register represents the top left pixel on the" },
{ AD_PCI_BAR1, 0x50, WDC_SIZE_8, WDC_READ_WRITE, "Somewhere",
"This register represents a pixel somewhere on the " },
};
const WDC_REG *gpSCREENCARD_Regs = gSCREENCARD_Regs;
```

演習 #2: 割り込み処理

目的: 対象のハードウェアの割り込みのテスト方法、割り込みハンドラの作成方法を練習します。

概要: この演習では DriverWizard を使用してフロッピー ディスク ドライブが生成する割り込みを “Listen” (確認) します。そして DriverWizard を使用して、この割り込みを Listen (確認) し、ユーザー モードの割り込みハンドラで割り込みを処理するアプリケーションを生成します。

演習ステップ:

Step #1: DriverWizard を起動して、New host driver project を選択します。
既に DriverWizard を起動している場合、New Device Driver Project ツールバー アイコンをクリックするか、File メニューからオプションを選択して、新しいプロジェクトを作成することができます。

Step #2: DriverWizard がマシンに接続されている Plug-and-Play デバイスのリストを表示します。フロッピー ディスク ドライブを使用するので、メニューから ISA を選択します。

Step #3: 右のウィンドウの Add Resource ボタンをクリックします。1 つ以上のメモリ範囲を定義します (ダミーの範囲でも構いません)。Resource Type コンボボックスから Memory Resource を選択し、アドレス 0x0-0x0 のメモリ範囲を定義します。この値で、生成されたコードのコンパイルおよびビルドし、フロッピー ディスクドライブの割り込み処理をテストするには十分です。OK をクリックします。

Step #4: 左のウィンドウで ISA Device を選択し、右のウィンドウで Add Interrupt ボタンをクリックします。 Interrupt number として 6 を選択し、Type を選択して、Edge Triggered とし、Shared チェックボックスをチェックし、OK をクリックします。

Step #5: 作成した割り込みを選択し、Listen to Interrupts ボタンをクリックします。フロッピー ドライブの割り込みを確認するには、フロッピー ドライブにアクセスします (たとえば、DOS コンソール画面で "a:" と入力するか、またはファイル ブラウザでフロッピー ドライブを選択します)。

Step #6: Generate Code ツールバー アイコンをクリックするか、または Project | Generate Code メニュー オプションを選択して、コードを生成します。リソースにアクセスし、定義した割り込みを処理する関数を Driver Wizard が生成します。ユーザーモードでアプリケーションから直接これらの関数を呼ぶことができます。

DriverWizard はまた、これらの関数を使用して対象のデバイスにアクセスするサンプル アプリケーションを生成します。

Step #7: サンプル アプリケーションをコンパイルし、起動します。

Step #8: サンプル アプリケーション内から割り込みを有効にし、確認します。 フロッピードライブの割り込みを確認してください。後で、割り込みハンドラを修正し、独自の機能を挿入します。

** 注意: WinDriver の Kernel PlugIn 機能を使用すると、割り込みおよび IO 呼び出しをカーネルモードで処理できます。そのため、パフォーマンスの最適化を実現できます。