

Application Note



5800 Series- Device Driver Instrument Control

Reference: AN502-02



Introduction

The Aeroflex Integrated Development Environment (AIDE) provides a comprehensive programming platform capable of controlling a vast array of devices in addition to the 5800 Automatic Test Equipment (ATE) hardware.

The AIDE can access devices present on any of the following standard interfaces :- PXI, compact PCI, PCI, Ethernet, GPIB (IEEE488), USB and Serial (RS232/422) plus any others which can be supported by the host PC platform.

These notes describe how devices with an appropriate Device Driver may be controlled from the AIDE in a .NET environment.

Functionality

The AIDE communicates with the 5800 ATE via the National Instruments (NI) Multisystem eXtension Interface bus (MXIbus). Control of this hardware is provided via the NI implementation of the Virtual Instrument Software Architecture (VISA) Application Programming Interface (API). To communicate with GPIB devices an NI IEEE488.2 card is added to the PC controller.

In a .NET environment, access to Instrument functionality is provided through .NET Assemblies. By adding direct references to these in an AIDE program the user can access all the available object types and methods necessary to communicate with the corresponding device.

Device Drivers

Very few devices have a .NET Assembly available. However, where a Device Driver is supplied, it is usually possible to create a .NET assembly wrapper that provides a mechanism for gaining the necessary access to the instrument.

General Device Drivers

The majority of device drivers come in the form of a Win32 Dynamic Link Library (DLL) that exposes a specific API for controlling that device. An instrument device driver then usually communicates with the target hardware via a “plug and play” Kernel Mode Device Driver that is started when the Operating System is booted. This “driver stack” is put in place during the vendor supplied installation process for the specific hardware and activated when the system is re-booted and the presence of the “new hardware” is detected (see the Microsoft Driver Developers Kit (DDK) documentation for a more detailed explanation).

Environment Specific Device Drivers

In addition there are environment specific device drivers for use with development tools and run-time-systems such as National Instruments LabWindows/CVI or LabView or Agilent’s VEE. These usually include a “soft front panel” that provides a visual control for displaying the available functionality and providing a means of selecting the desired features and generating code statements in the host environment. Behind such panels there are still Win32 DLLs.

Such drivers typically use the VISA API to communicate with their target hardware. In this case the Kernel Mode Device Driver that provides the low level hardware access is that supplied by NI e.g. for a PXI device, the access is via a hierarchy of device specific Win32 DLL to the NI-VISA API DLL, through to the NI-VISA layer then NI-MXI layer Kernel Mode Device Drivers. This class of device identifies itself to the host environment with the additional information supplied in a “.inf” file. This enables the National Instruments Measurement and Automation Explorer (NI-MAX) utility, for example, to display the device in the PXI Chassis Configuration.

The user can often be fooled into believing that a device that is physically plugged into a slot in a PXI backplane is NOT WORKING because it is NOT displayed in the NI-MAX Chassis Configuration. Instead the Device description for the occupied Slot is shown as <Unknown/Empty>. This is because the NI software has failed to find the appropriate “.inf” file for a device that is not specifically designed for the NI environment. The apparently strange thing is that these devices are still detected at

boot-up by the “plug and play” process looking for PXI and compact PCI devices via the NI-MXI card. This can be verified by checking with the Windows Device Manager. This will show all the devices that have been detected and for which the appropriate device drivers have been successfully started.

Vendor Specific - Class Device Drivers

Many vendors (including NI) supply a range of devices of a specific type on particular interfaces. In this case it is much more efficient to supply a single API that provides control of all of this Class of devices from that manufacturer. Typical examples are the NI-DAQ (National Instruments – Data Acquisition) and the ADLink Technology Inc. NuDAQ PCIS-DASK (Data Acquisition Software Kit) ranges of PCI, cPCI and PXI plug in cards.

This means that a single Win32 DLL provides the API for all the devices in the Class. A generic Kernel Mode Device Driver is then used to provide the necessary low level device access for all of these devices.

In the case of NI, there is not only a Win32 DLL for interfacing to Win32 application programs written in VB, C, C++ etc. but also a .NET Assembly NationalInstruments.DAQmx for the use of applications written in VB.NET and C#.

In the case of ADLink devices there is a Win32 DLL (PCI-DASK.dll) for interfacing to Win32 application programs written in VB, C, C++ etc. ADLink do not as yet provide a .NET Assembly, however, using the DLL Wrapper Tool developed specifically for the 5800 ATE environment, a .NET Assembly (PCIDaskWrapper.PCIDask) has been created that provides the required interface to the whole PCI-DASK API.

The DLL Wrapper Tool

The DLL Wrapper Tool developed specifically for the 5800 ATE environment can be used to produce a .NET Assembly that wraps the API of almost any available Win32 Device Driver. The details of this process can be rather complicated, are outside the scope of this Application Note, and are covered elsewhere (see DLL Wrapper Help in a 5800 Software Installation). Suffice to say that the tool is capable of creating such wrapper Assemblies.

Customers who feel they do not have the necessary time or experience to produce such wrapper Assemblies can contact Aeroflex ATE Customer Support about providing a suitable solution.

Program Example 1 – NI-DAQ Devices

Create a new Program called, for example, AIDEdaq.

Detailed information on the NI VISA API can be obtained from the installed documentation by selecting any of a number of files from:-

```
Start\All Programs\National Instruments\NI-DAQ\
```

Or by using a .NET Reflector tool (see Lutz Roeder’s at <http://www.aisto.com/roeder/dotnet/>) to examine the public contents of the NI Assemblies. The AIDE also uses .NET Reflection to show the available functionality.

This example assumes the presence of an NI-DAQ card such as the NI PXI 6025E which has both Analogue and Digital I/O capabilities.

Configuration Data

Expand the Configuration Data section of the program and add the following .NET Assembly References from the .NET Global Assembly Cache (GAC) or the NI-DAQ installation directories using the browse facilities provided:-

```
.NET Assembly Reference
    Assembly:NationalInstruments.DAQmx, ...
.NET Assembly Reference
    Assembly:NationalInstruments.VisaNS, ...
.NET Assembly Reference
    Assembly:NationalInstruments.Common, ...
```

Add the following .NET Namespace References:-

```
.NET Namespace Reference NationalInstruments.DAQmx
.NET Namespace Reference NationalInstruments.VisaNS
```

Global Symbols

Create a new Global Variable of type `AnalogSingleChannelReader` called `analogInput0` and assign it an `InitialValue = null`. The available NI-DAQ data types can be found and selected by clicking the button in the Type property window and pressing the `More Types...` button and scrolling the subsequent display contents until the required data type can be seen and selected.

Create another new Global Variable of type `double` called `result` and also assign an `InitialValue=null`.

Main Method

Create a new Method called `Main` and reference it in the Program `OnStart` Property.

We can now create Evaluate statements that can be used to communicate with the NI-DAQ hardware.

NI PXI 6025E Device

To communicate with the NI PXI 6025E device in a slot in the PXI rack create the Evaluate statement:-

```
Evaluate analogInput1 = new
    AnalogSingleChannelReader (Dev1/ai1)
```

This statement should assign Analog Input Channel 1 (ai1) from Device 1 (Dev1) to the newly created `analogInput1` instance of an `AnalogInputChannelReader`.

To take a measurement reading we insert a statement of the form:-

```
Evaluate result = analogInput1.ReadSingleChannel().
```

Program Example 2 – PCIS DASK Devices

Detailed information on the nuDAQ PCIS-DASK API can be obtained from the installed documentation by selecting any of a number of files from:-

```
Start\All Programs\PCIS-DASK\
```

Or by using a .NET Reflector tool (see Lutz Roeder's at <http://www.aisto.com/roeder/dotnet/>) to examine the public contents of the Aeroflex generated `PCIDaskWrapper.PCIDask`

Assembly. The AIDE also uses .NET Reflection to show the available functionality.

This example assumes the presence of an ADLink nuDAQ cPCI 7432 Opto-Isolated Digital I/O card.

Configuration Data

Expand the Configuration Data section of the program and add the following .NET Assembly References from the .NET Global Assembly Cache (GAC) or the Aeroflex nuDAQ installation directories using the browse facilities provided (use the file selection option and search for file `PCIDaskLib.dll`):-

```
.NET Assembly Reference
    Assembly:PCIDaskWrapper.PCIDask, ...
```

Or

```
.NET Assembly Reference
    Path:C:\ADLink\PCI-DASK\DLLWrapper\PCIDaskLib.dll
```

Add the following .NET Namespace References:-

```
.NET Namespace Reference PCIDaskWrapper.PCIDask
```

Global Symbols

Create new Global Variables as follows:-

Type	Name	InitialValue
Short	cardID	-1
Ushort	port	0
Uint	data	0

Create new Global Constants as follows:-

Type	Name	InitialValue
UShort	PCI_7432	16

Main Method

We can now create Evaluate statements that can be used to communicate with nuDAQ PCIS-DASK devices.

nuDAQ cPCI 7432

To communicate with the nuDAQ cPCI 7432 device in a slot in the PXI rack create the Evaluate statement:-

```
Evaluate cardID = Register_Card(PCI_7432,[ UShort] 0)
```

This statement should return the `cardID` of the first `PCI_7432` card in the system (`card 0`). In a system with only 1 `PCI-DASK` card present the value returned is 0. An error would return a negative value.

To Write (`DO = Digital Out`) or Read (`DI = Digital In`) the I/O pin state we insert statements of the form:-

```
Evaluate DO_WritePort ([ UShort] cardID, port, data)
```

and

```
Evaluate DI_ReadPort ([ UShort] cardID, port, ref
data) .
```

Note the use of the `[UShort]` cast operator on the `cardID` variable.

This has been done to get around a typical problem that can occur in the tightly typed .NET environment when communicating with methods defined in the somewhat slacker 'C' programming environment.

The specification of the return value of the Register_Card method is type short (signed 16 bit integer) as the actual value returned can be negative if an error occurs. The specification for the cardID parameter to the DO_WritePort and DO_ReadPort methods is however of type ushort (unsigned 16 bit integer), hence the cast operation.

The problem might have been solved had the variables used been defined as follows:-

Type	Name	InitialValue
ushort	cardID	0
ushort	port	0
uint	data	0
short	result	-1

The program statements could then be changed to:-

```
Evaluate result = Register_Card(PCI_7432,[ ushort] 0)
If ( result >= 0)
    Then
        Evaluate cardID = [ ushort] result
        Evaluate DO_WritePort(cardID, port, data)
        Evaluate DI_ReadPort(cardID, port, ref
        data) .
```

The programmer must always be aware that such problems can occur when using an API that did not originate within the .NET environment.

CHINA Beijing
 Tel: [+86] (10) 6539 1166
 Fax: [+86] (10) 6539 1778
CHINA Shanghai
 Tel: [+86] (21) 5109 5128
 Fax: [+86] (21) 5150 6112
CHINA Shenzhen
 Tel: [+86] (755) 3301 9358
 Tel: [+86] (755) 3301 9356
FINLAND
 Tel: [+358] (9) 2709 5541
 Fax: [+358] (9) 804 2441

FRANCE
 Tel: [+33] 1 60 79 96 00
 Fax: [+33] 1 60 77 69 22
GERMANY
 Tel: [+49] 89 99641 0
 Fax: [+49] 89 99641 160
HONG KONG
 Tel: [+852] 2832 7988
 Fax: [+852] 2834 5364
INDIA
 Tel: [+91] 80 [4] 115 4501
 Fax: [+91] 80 [4] 115 4502

JAPAN
 Tel: [+81] (3) 3500 5591
 Fax: [+81] (3) 3500 5592
KOREA
 Tel: [+82] (2) 3424 2719
 Fax: [+82] (2) 3424 8620
SCANDINAVIA
 Tel: [+45] 9614 0045
 Fax: [+45] 9614 0047
SINGAPORE
 Tel: [+65] 6873 0991
 Fax: [+65] 6873 0992

UK Stevenage
 Tel: [+44] (0) 1438 742200
 Fax: [+44] (0) 1438 727601
 Freephone: 0800 282388
USA
 Tel: [+1] (316) 522 4981
 Fax: [+1] (316) 522 1360
 Toll Free: 800 835 2352



As we are always seeking to improve our products, the information in this document gives only a general indication of the product capacity, performance and suitability, none of which shall form part of any contract. We reserve the right to make design changes without notice. All trademarks are acknowledged. Parent company Aeroflex, Inc. ©Aeroflex 2011.

www.aeroflex.com
info-test@eroflex.com



Our passion for performance is defined by three attributes represented by these three icons: solution-minded, performance-driven and customer-focused.