

Application Note



5800 Series- 5800 AIDE Control of ADLink PXI Cards

Reference: AN513-01



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.

This note describes how ADLink PXI Cards 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).

The ADLink PXI Cards are controlled by an ADLink supplied Win32 DLL (D2K-Dask.dll) which is installed on the target PC controller, is common to all the ADLink PXI devices and provides a single API for user applications written in any of the common programming languages (C, C++, VB etc.).

In a .NET environment, access to ADLink PXI Card functionality can be provided through a common Aeroflex .NET Assembly that has been generated from the software provided by ADLink for this purpose. By adding direct references to the Aeroflex D2K-DaskLib.dll Assembly in an AIDE program the user can access all the available object types and methods necessary to communicate with the corresponding devices.

Installation

ADLink Installation

The user must first install the ADLink software and hardware supplied with their PXI device. This places all the hardware device drivers and documentation on the target PC Controller. Follow the ADLink instructions.

It is also possible to obtain the necessary software directly from the ADLink web site.

AIDE Installation Modification

In order for an AIDE test program to have access to the ADLink PXI hardware the ADLink supplied Win32 DLL (D2K-Dask.dll) and the Aeroflex supplied Assembly (D2K-DaskLib.dll) must be made accessible. This can be achieved by copying these DLLs to the AIDE installation \bin directory. Alternatively they can be placed in the same directory as the programs that require access to them.

Program Example – ADLink DAQ Devices

Create a new Program called, for example, ADLinkdaq.

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

```
Start\All Programs\D2K-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 D2K-DaskLib.dll Assembly. The AIDE also uses .NET Reflection to show the available functionality (see below).

This example assumes the presence of an ADLink PXI 2010 DAQ card that has both Analog and Digital I/O capabilities. The program is derived from a sample C program generated by the ADLink Code Creator tool to demonstrate using the 2010 to generate an analog waveform.

Configuration Data

Expand the Configuration Data section of the program and add the following .NET Assembly reference :-

Configuration Data

.NET Assembly Reference

```
Path:D:\Program Files\Aeroflex\5800 System Software\Bin\D2K-DaskLib.dll
```

Add the following .NET Namespace Reference:-

.NET Name space Reference

D2KDaskLib

Global Symbols

Create the Global Variables as shown below.

Global Symbols

Variables

```
Variable [ Short ] cardID InitialValue = -1
Variable [ Short ] err InitialValue = 0
Variable [ Byte ] fstop InitialValue = 0
Variable [ UInt ] WriteCnt InitialValue = 0
Variable [ UInt] MemSize InitialValue = 0
Variable [ UShort ] BufId InitialValue = -1
Variable [ Short[] ] OutBuf InitialValue = new Short[ 1000]
```

Create the Global Constants as shown below.

Global Symbols

Constants

```
Constant [ Int ] CardNumber = 0
Constant [ UShort ] DACHan = 0
Constant [ UInt ] CHUI = 40
Constant [ UInt ] Definite = 1
Constant [ UInt] Iteration = 1
Constant [ UInt ] UpdateCount = 1000
Constant [ UShort ] ConfigCtrl =
D2KdaskLib.D2KDask.DAQ2K_DA_WRSRC_Int
Constant [ UShort ] TrigCtrl =
[UShort] (D2KDaskLib.D2KDASK.DAQ2K_DA_TRGSRC_SOFT |
D2KDaskLib.D2KDASK.DAQ2K_DA_TRGMOD_POST)
Constant [ UShort ] ReTrgCnt = 0
Constant [ UShort ] Dly1Cnt = 0
Constant [ UShort ] Dly2Cnt = 1
Constant [ Boolean ] BufAutoReset = true
Constant [ UShort ] SyncMode =
D2KDaskLib.D2KDASK.ASYNCH_OP
Constant [ UShort ] StopMode =
D2KDaskLib.D2KDASK.DAQ2K_DA_TerminateImmediate
```

The available ADLink D2KDask constant data definitions can be found in the ADLink documentation and selected in AIDE by clicking the Type Viewer button in the Edit Constant Expression window, placing D2KDaskLib.D2KDASK in the Type Viewer Type list box and pressing the Show button. Scroll the subsequent

display contents until the required definition can be seen and selected.

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 ADLink DAQ hardware.

ADLink PXI 2010 Device

To communicate with the ADLink PXI 2010 device in a slot in the PXI rack create the Evaluate statement:-

```
Evaluate cardID =  
    D2KdaskLib.D2KDASK.D2K_Register_Card(D2Kdask  
    Lib.D2KDASK.DAQ_2010, [UShort] CardNumber)
```

This statement calls the static member function D2K_Register_Card which should return a valid card ID for the specified device. This should be a positive Short value. A negative value indicates an error code.

The ADLink PXI 2010 must now be initialized and set up for the required mode of operation. The available settings can be obtained from the installed ADLink documentation, see – D2K-DASK Function Reference Manual (D2KDASKFR.pdf) and D2K-DASK Data Acquisition Software Development Kit For DAQ-2000 Devices User's Guide (D2KDASK Manual.pdf).

```
Evaluate err =  
    D2KdaskLib.D2KDASK.D2K_AO_InitialMemory  
    Allocated([ UShort] cardID, out [ UInt] MemSize)
```

This statement calls the static member function D2K_AO_InitialMemoryAllocated to determine the memory available to hold the required analog output data pattern. This should output the memory size to the MemSize variable. We need to check this value as shown.

```
If (MemSize*1024 < UpdateCount*2)  
    Then  
        Evaluate MessageBox.Show("Insufficient  
        Memory Allocated","D2Kdask Error")  
Return
```

We now require to set up the DAQ Analog Output Channel to be used to generate the waveform:-

```
Evaluate err = D2KdaskLib.D2KDASK.D2K_AO_CH_  
Config(  
[ UShort] cardID,  
DACHan,  
D2KdaskLib.D2KDASK.DAQ2K_DA_BiPolar,  
D2KdaskLib.D2KDASK.DAQ2K_DA_Int_REF,  
10.0)
```

This statement calls the static member function D2K_AO_CH_Config to select DA Channel 0, Bi-Polar, Internal 10 V Reference (this is actually the default setting).

Now set up the DAQ Card to disable double-buffered data acquisition mode.

```
Evaluate err = D2KdaskLib.D2KDASK.D2K_AO_Async  
Db1BufferMode([ UShort] cardID, false)
```

Now configure the DAQ Channel to output a continuous waveform using the previously defined Constant parameters.

```
Evaluate err = D2KdaskLib.D2KDASK.D2K_AO_Config(  
[ UShort] cardID, ConfigCtrl, TrigCtrl, ReTrgCnt,  
Dly1Count, Dly2Count, AutoResetBuf)
```

Where :-

```
ConfigCtrl = DAQ2K_DA_WRSRC_Int
```

Set Timer Source to INTERNAL.

```
TrigCtrl = DAQ2K_DA_TRGSRC_SOFT|DAQ2K_DA_TRGMOD_  
POST
```

Set Trigger Source to SOFTWARE.

Set Trigger Mode to POST TRIGGER.

```
ReTrgCnt = 0
```

Set Re-Trigger Count to INFINITE (i.e. continuous output).

```
Dly1Cnt = 0
```

Set Post-Trigger Delay to 0 (i.e. output immediately on receipt of trigger).

```
Dly2Cnt = 1
```

Set Repeat Pattern Delay to 1 (i.e. delay 1 count on Timer Source between consecutive waveform generations).

```
AutoResetBuf = true
```

Set DA Buffer to RESET AUTOMATICALLY (i.e. start point is reset for next repeat of output waveform).

Now create the required output waveform – a single cycle of a SINE wave.

```
For (Int i=0 ; i<1000 ; i=i+1)  
    Evaluate OutBuf[i] =  
        [ Short] (Math.Sin(i*2*Math.PI/1000))
```

Now transfer the waveform pattern to the DAQ Channel output buffer memory.

```
Evaluate err = D2KdaskLib.D2KDASK.  
D2K_AO_ContBufferSetup ([ UShort] cardID,  
OutBuf, UpdateCount, out BufId)
```

Where :-

BufId = the index of the buffer currently set up.

The BufID value is output by this function (see manuals for details – different cards have different numbers of buffers available).

Now perform continuous D/A conversions on the specified analog output channel at a rate as close to the rate specified.

```
Evaluate err =  
D2KdaskLib.D2KDASK.D2K_AO_ContWriteChannel  
( [ UShort] cardID, DACHan, BufId, UpdateCount,  
Iterations, CHUI, Definite, SyncMode)
```

Where :-

```
DACHan = 0
```

The DACHan value specifies Analog output channel number.

BuflD = the index of the buffer set up to hold the waveform.

The BuflD value was output by the previous D2K_AO_Config function (see manuals for details – different cards have different numbers of buffers available).

```
UpdateCount = 1000
```

The UpdateCount value specifies the number of samples in the defined waveform.

```
Iterations = 1
```

The number of times the data in the buffer is to be output to the port. If the DA operation is performed synchronously, this argument must be set as 1. See manuals for Iterations description.

```
CHUI = 40
```

Set Channel Update Interval to 40 (see manuals).

```
Definite = 1
```

Set Definite to 1 (i.e. definite=1 rather than indefinitely=0).

```
SyncMode = D2KdaskLib.D2KDASK.ASYNCH_OP
```

Set SyncMode to ASYNCH_OP (i.e. asynchronous D/A conversion).

Now monitor the current status of the asynchronous analog output operation for completion or error.

```
While (fstop == 0)
```

```
Evaluate err = D2KdaskLib.D2KDASK.D2K_AO_Async  
  Check([UShort] cardID, out fstop, out  
  WriteCnt)
```

Where :-

```
fstop = current status
```

The fstop value is updated on each call to the D2K_AO_AsyncCheck function.

WriteCnt = number of output data values that have been written

The WriteCnt value is updated on each call to the D2K_AO_AsyncCheck function.

Finally stop the asynchronous analog output operation.

```
Evaluate err =  
  D2KdaskLib.D2KDASK.D2K_AO_AsyncClear([UShort]  
  cardID, out WriteCnt, StopMode)
```

Where:-

WriteCnt = number output data values that have been written

The WriteCnt value is the last value returned by the D2K_AO_AsyncCheck function.

```
StopMode = DAQ2K_DA_TerminateImmediate
```

The StopMode value is set to terminate immediately.

For each function call that returns an error code, the value should be checked and any failure condition reported.

```
If (CheckError(err))  
  Then  
    Return
```

An example error checking routine might be :-

```
Method CheckError [ Boolean] ([ Short] error)
```

```
  Local Symbols
```

```
  Method Parameters
```

```
  Method Parameter [ Short] error  
  DefaultValue=0
```

```
Variables
```

```
  Variable [ String] errorMsg
```

```
Method Code
```

```
  Switch ( error )
```

```
Case D2KdaskLib.D2KDASK.NoError:
```

```
  Evaluate errorMsg = "NoError"
```

```
Case D2KdaskLib.D2KDASK.ErrorAdTimeOut:
```

```
  Evaluate errorMsg = "Error - AdTimeOut"
```

```
...
```

```
Case Default:
```

```
  Evaluate errorMsg = "Unknown error code " +  
  error.ToString() + " detected"
```

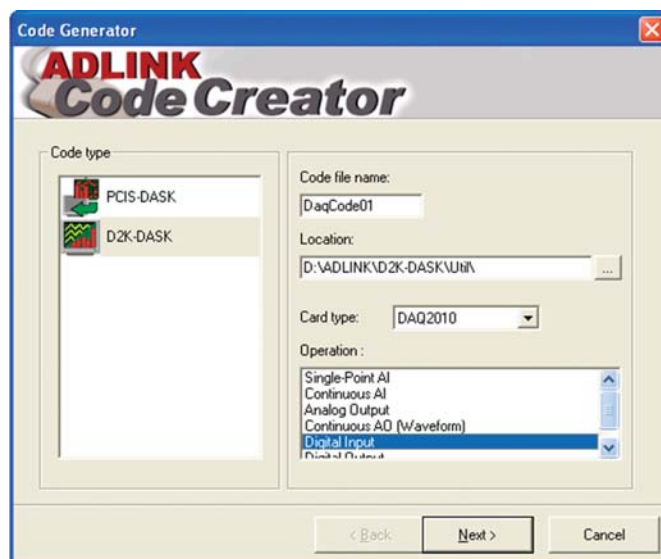
```
If ( error != 0)
```

```
  Then
```

```
    Evaluate MessageBox.Show(errorMsg, "D2K  
    Dask Error")
```

Program Development of ADLink DAQ Devices

The ADLink Code Creator tool can be used as a first step to investigating different hardware setups. It is then a relatively straight forward task to translate the generated example C code to the AIDE environment.



The ADLink Code Creator tool allows the programmer to specify the Card type and the Operation, Analog or Digital, Input or Output etc.

For a Digital Input or Output operation the user is offered either port or line Channel mode. In port mode the input or output data is 8-bits wide. In line mode the data is a single bit (0 or 1) on a designated input or output port.



A typical example of the C-code generated is shown below along with the equivalent AIDE program code.

The ADLink Code Creator tool Digital Input example program :-

```
// DAQ_2010 : Digital Input
/*****
```

The following code segment is generated according to your device setup. You can copy and insert it into your program to operate your device.

```
***** /
#include "d2kdask.h"

//constants definition
#define CardNumber 0
#define DIPort Channel_P1A
#define Direction INPUT_PORT
#define DILine 0

//variables definition
I16 cardID = -1;
I16 err=0;
U16 DIValue = 0; // returned DI value

cardID = D2K_Register_Card(DAQ_2010,
CardNumber);
if (cardID<0) {
//Error occurs !!
//ToDo : Handle error here
```

```
}
err = D2K_DIO_PortConfig(cardID, DIPort,
Direction);
err = D2K_DI_ReadLine(cardID, DIPort, DILine,
&DIValue);
if (err!=NoError) {
//Error occurs !!
//ToDo : Handle error here
}

D2K_Release_Card(cardID);
```

A typical translation to AIDE code is shown below :-

```
Evaluate cardID =
D2KdaskLib.D2KDASK.D2K_Register_Card(
D2KdaskLib.D2KDASK.DAQ_2010, [ UShort]
CardNumber)

If (cardID < 0)
Then
Evaluate MessageBox.Show("Card NOT FOUND",
"D2KDask Error")

Return

Evaluate err =
D2KdaskLib.D2KDASK.D2K_DIO_PortConfig
(cardID, DIPort, Direction)

Evaluate err =
D2KdaskLib.D2KDASK.D2K_DI_ReadLine
(cardID, DIPort, DILine, out DIValue)

If (err != 0)
Then
Evaluate MessageBox.Show("Card NOT FOUND",
"D2KDask Error")

Return

Evaluate
D2KdaskLib.D2KDASK.D2K_Release_Card
(cardID)
```

The correspondence can be clearly seen.

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@aeroflex.com



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