ICP DAS

Driver & SDK

UNiDAQ

Driver DLL User Manual

English Version

Supports 64-bit OS

Supports Windows 10

Supports most PCI I/O Boards

Warning

ICP DAS assumes no liability for any damage resulting from the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademarks

Names are used for identification purposes only and me be registered trademarks of their respective companies.

About this Manual

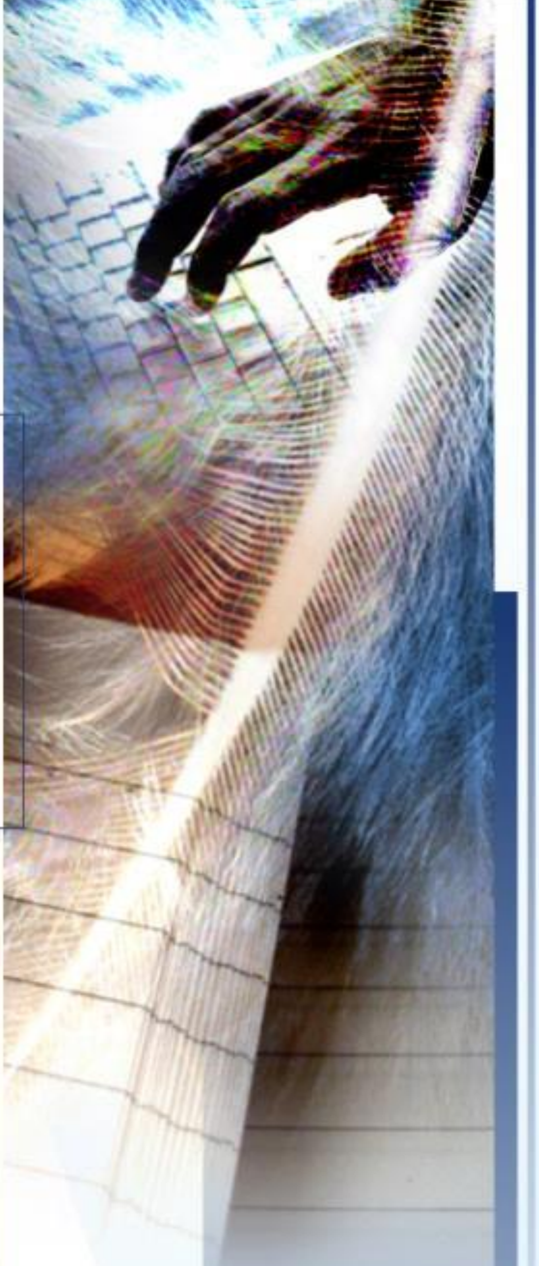
This manual contains the information you need to get started with the ICP DAS DLL Driver software package. The DLL Drivers allow you to easily perform vital I/O operations through the API, functions and structure.

The UniDAQ DLL drivers can be used to develop custom programs based on the VB, VC, BCB, Delphi, VB.NET, C#.NET, VC.NET, Console and other programming languages using Windows Systems. This manual also provides sample programs that can be modified to create custom applications that meet specific requirements.

If you have any questions, feel free to contact the ICP DAS Service Department via email at: service@icpdas.com

Table of Contents

Appendix A. Return Value and Configuration code 185



1. Introduction

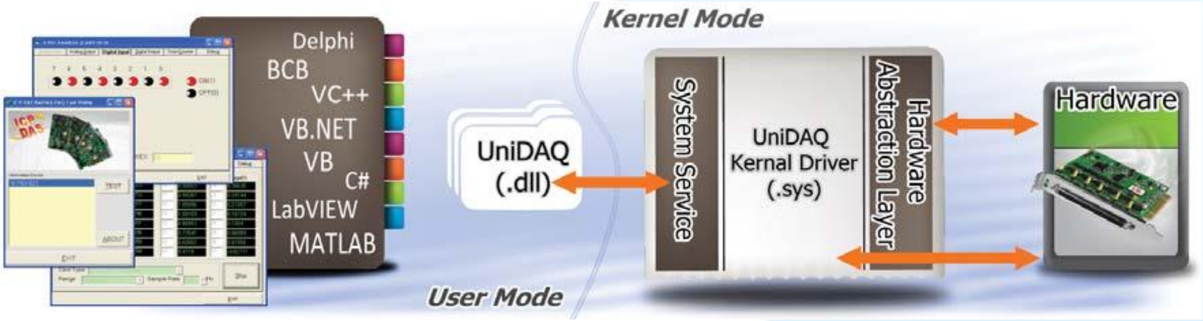
This chapter provides an overview of the functions and requirement for ICP DAS UniDAQ Driver DLL

1.1. Introducing the UniDAQ Driver DLL

The ICP DAS UniDAQ Driver DLL provides complete hardware functions and maximum performance. With the ICP DAS UniDAQ Driver DLL, there is no need to use hardware-specific register commands thanks to the powerful API function that can be used with a variety of programming environments and languages.

ICP DAS UniDAQ Driver DLL uses direct I/O techniques to promote API efficiency and I/O speed. It also provides interrupt and event notification functions, so that if an interrupt event occurs within the device, the user application will be notified via a callback function. Then, only the necessary actions need to be taken without needing to manually check the status of the hardware, which is more efficient and reduces the complexity of the application.

The ICP DAS UniDAQ Driver DLL supports Windows 2000 and both 32- and 64 bit versions Windows XP/2003/Vista/7/2008/8/2012/10.



1.2. Supported ICP DAS Products

The following is a summary of the ICP DAS products supported ICP DAS UniDAQ Driver DLL.

Model	Model
PIO-D24/D56/D24U/D56U、PEX-D24/D56	PIO-D48/D48U/D48SU、PEX-D48
PIO-D64/D64U	PIO-D96/D96U/D96SU、PEX-D96S
PIO-D144/D144U/D144LU、PEX-D144LS	PIO-D168/D168U
PCI-D96SU/D128SU	PIO-DA4/DA8/DA16/DA4U/DA8U/DA16U
PISO-DA4U/DA8U/DA16U	PEX-DA4/DA8/DA16
PIO-821L/821H/821LU/821HU	PISO-C64/C64U/P64/P64U
PEX-C64/P64	PISO-A64/A64U/P32A32/P32A32U/ P32A32U-5V、PEX-P32A32
PISO-P32C32/P32C32U/P32C32U-5V/P32S32WU	PEX-P32C32
PISO-P8R8/P8R8U	PISO-P8R8AC/P8R8DC
PISO-P16R16U、PEX-P16R16i/P8R8i	PISO-1730U
PISO-730/730A/730U/730AU、PEX-730/730A	PISO-725/725U
PISO-DA2/DA2U	PISO-813/813U
PCI-TMC12/TMC12A/TMC12AU、PEX-TMC12A	PCI-M128/M256/M512/M512U
PCI-P16R16/P16R16U/P16C16/P16C16U/ P16POR16/P16POR16U/P8R8/P8R8U	PEX-P16POR16i/P8POR8i
PCI-1002L/1002H/1002LU/1002HU	PCI-1202L/1202H/1202LU/1202HU
PEX-1002L/1002H	PEX-1202L/1202H
PCI-1602/1602U,PCI-1602F/1602FU	PCI-1800L/1800H/1800LU/1800HU
PCI-1802L/1802H/1802LU/1802HU	PCI-822LU/826LU
PCI-FC16U	PCI-2602U
PCIe-8620	PCIe-8622

1.3. System Requirements

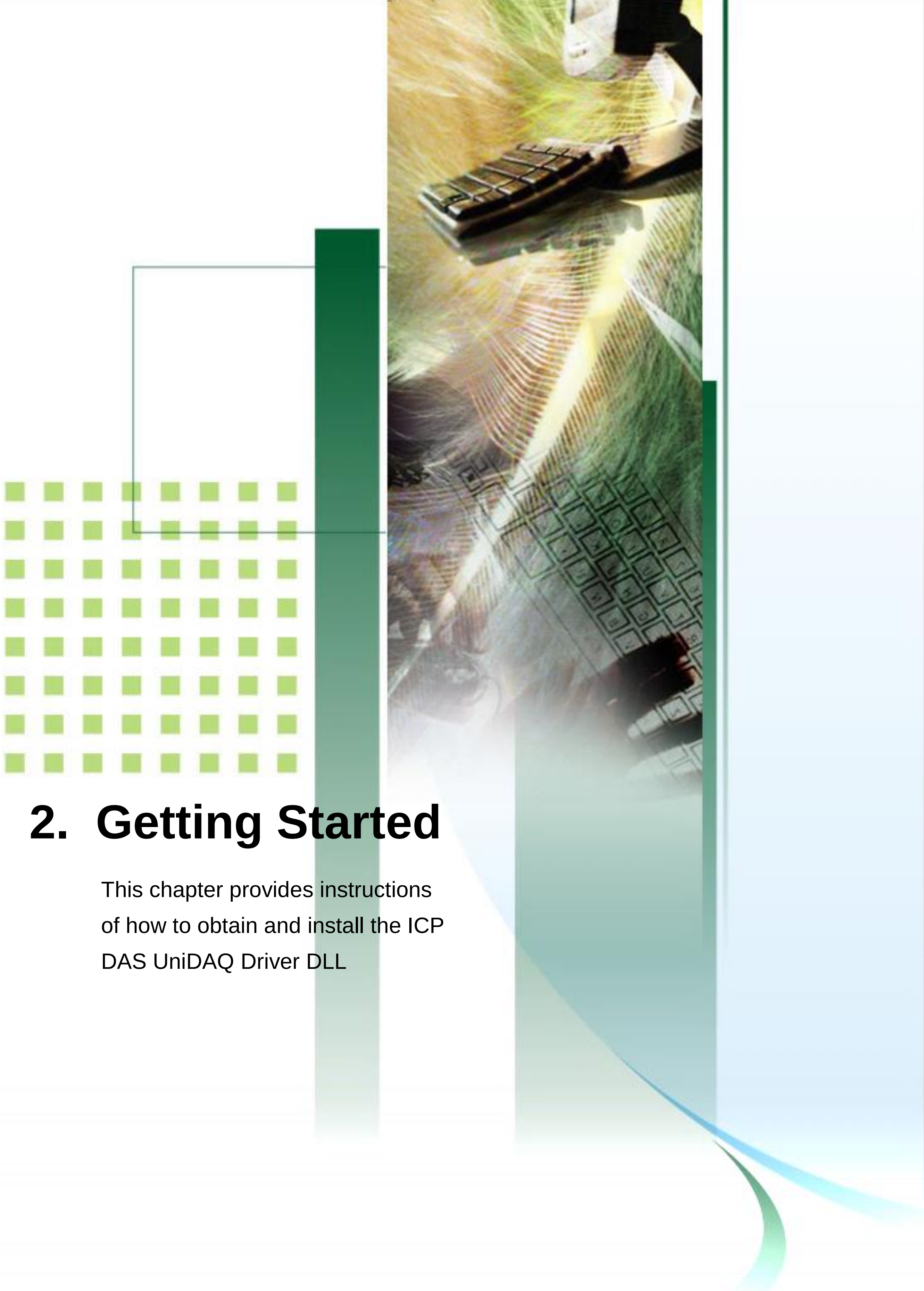
Minimum system requirements for ICP DAS UniDAQ Driver DLL are:

- 266 MHz 32-bit (x86) or 64-bit (x64) processor
- 64 MB of system memory
- Support for Super VGA graphics
- At least 20 MB of available space
- DVD/CD-ROM drive
- 32- or 64-bit Windows Operating System (Windows 2000 or later – see table below)

Operating system of Windows requirement

32-bit (x86)	64-bit (x64)
Windows 2000	**-**
Windows XP	**Windows XP**
Windows Server 2003	**Windows Server 2003**
Windows Vista	**Windows Vista**
Windows Server 2008	**Windows Server 2008**
Windows 7	**Windows 7**
-	**Windows Server 2012**
Windows 8/8.1	**Windows 8/8.1**
Windows 10	**Windows 10**

Note that Windows version 3.1,95,98,ME, and NT are not supported



2. Getting Started

This chapter provides instructions of how to obtain and install the ICP DAS UniDAQ Driver DLL

2.1. Obtaining the UniDAQ Driver DLL Installer package

The installer package for the ICP DAS UniDAQ Driver DLL can be found on the companion CD-ROM, or can be downloaded from the ICP DAS FTP site or the web site. The locations are:

CD:\\ NAPDOS\PCI\UniDAQ\DLL\Driver

http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/unidaq/dll/driver/

ftp://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/unidaq/dll/driver/

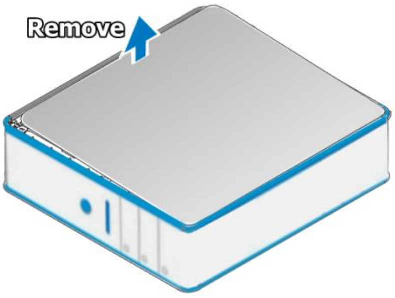
2.2. Installing the UniDAQ Driver DLL

Step 1 Install the DAQ Card

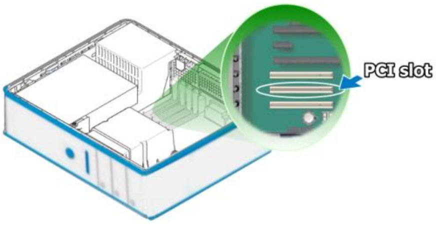
Install DAQ card by following the procedure described below:

Turn off computer
Turn Off

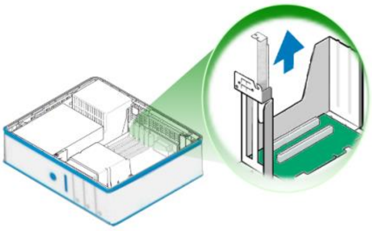
Correctly shut down and power off your computer, and then disconnect the power supply.



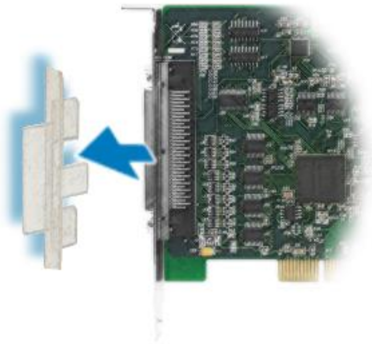
Remove the cover from the computer.



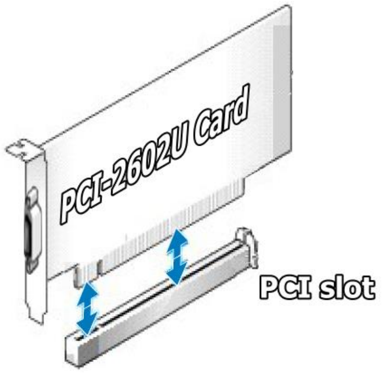
Select an empty PCI or PCIe slot.



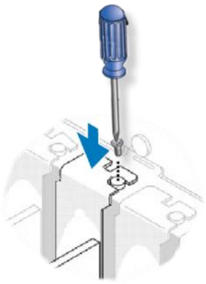
Remove the screw holding the cover for the PCI slot in place and then remove the slot cover from the PC. Ensure that you do not misplace the screw.



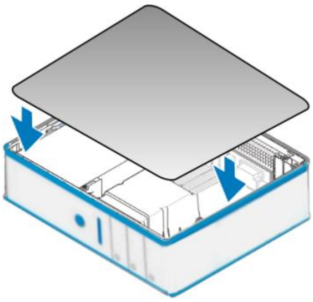
Remove the connector cover form the card.



Align the contacts of the card with the open slot on your motherboard and carefully insert your card into the PCI or PCIe slot.



Screw the mounting bracket screw into the new PCI or PCIe card bracket to secure the card in place.



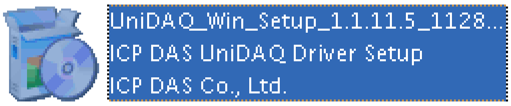
Re-attach cover for the computer and reconnect the power supply.

Power on the computer.

Step 2 Set up the ICP DAS UniDAQ Driver DLL

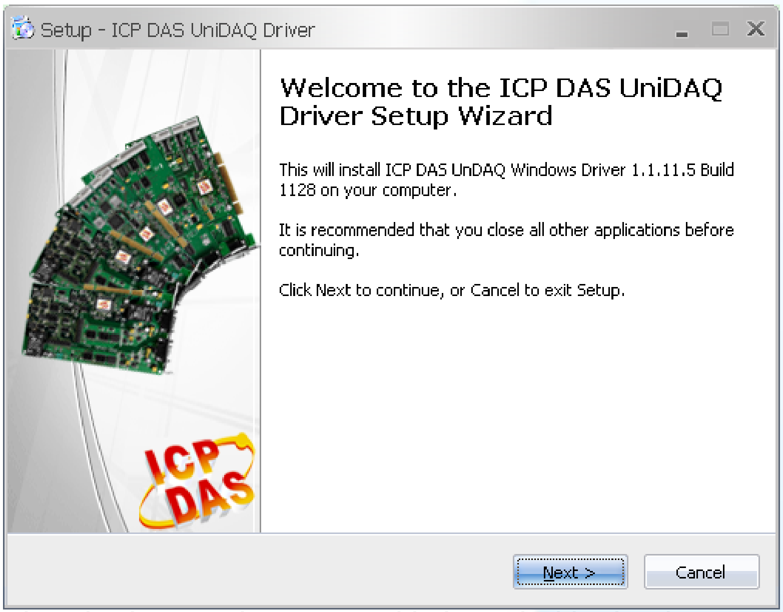
Install UniDAQ Driver DLL by following the procedure described below

1. Insert the companion CD into the CD-ROM drive on the computer, and then double-click the "UniDAQ_Win_Setup_x.x.x.x_xxxx.exe" file in the Driver folder.



UniDAQ_Win_Setup_1.1.11.5_1128...
ICP DAS UniDAQ Driver Setup
ICP DAS Co., Ltd.

2. When the "Welcome to the ICP DAS Driver Setup Wizard" screen is displayed, click the "Next>" button to start the installation.



Setup - ICP DAS UniDAQ Driver

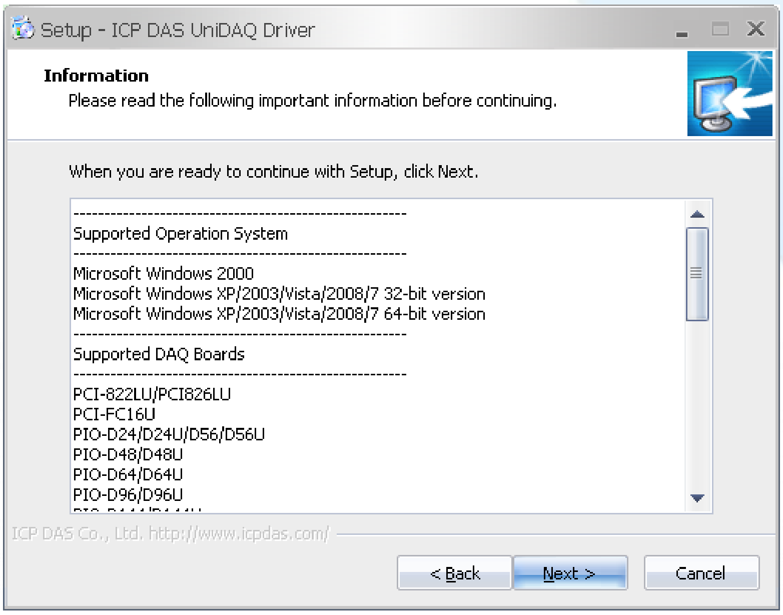
Welcome to the ICP DAS UniDAQ Driver Setup Wizard

This will install ICP DAS UnDAQ Windows Driver 1.1.11.5 Build 1128 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

3. Check that the installed DAQ Card is included in the list of supported devices, and then click the "Next>" button to continue.



Setup - ICP DAS UniDAQ Driver

Information

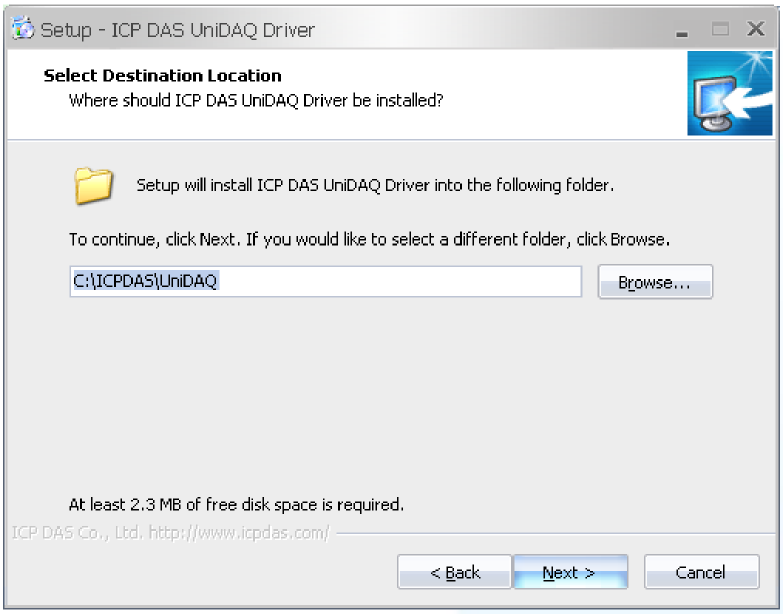
Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

```
-----------------------------------------------------
Supported Operation System
-----------------------------------------------------
Microsoft Windows 2000
Microsoft Windows XP/2003/Vista/2008/7 32-bit version
Microsoft Windows XP/2003/Vista/2008/7 64-bit version
-----------------------------------------------------
Supported DAQ Boards
-----------------------------------------------------
PCI-822LU/PCI826LU
PCI-FC16U
PIO-D24/D24U/D56/D56U
PIO-D48/D48U
PIO-D64/D64U
PIO-D96/D96U
```

ICP DAS Co., Ltd. http://www.icpdas.com/

4. Click the “Next>” button to install the software in the default folder, C:\ICPDAS\UniDAQ, or click the “Browse…” button to select the destination folder for the installation.



Setup - ICP DAS UniDAQ Driver

Select Destination Location

Where should ICP DAS UniDAQ Driver be installed?

Setup will install ICP DAS UniDAQ Driver into the following folder.

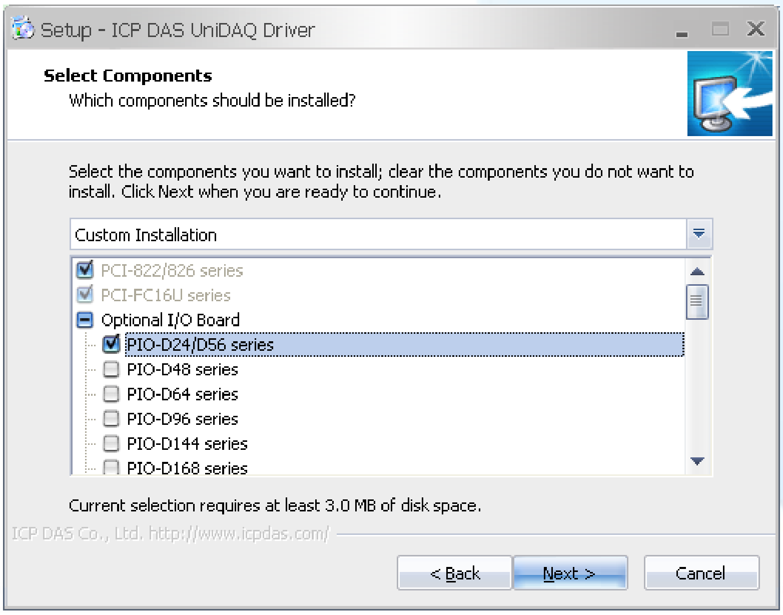
To continue, click Next. If you would like to select a different folder, click Browse.

C:\ICPDAS\UniDAQ

At least 2.3 MB of free disk space is required.

ICP DAS Co., Ltd. http://www.icpdas.com/

5. On the "Select Components" screen, check that the DAQ Card is included in the list of components to be installed, and then click the “Next >” button to continue.



Setup - ICP DAS UniDAQ Driver

Select Components

Which components should be installed?

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

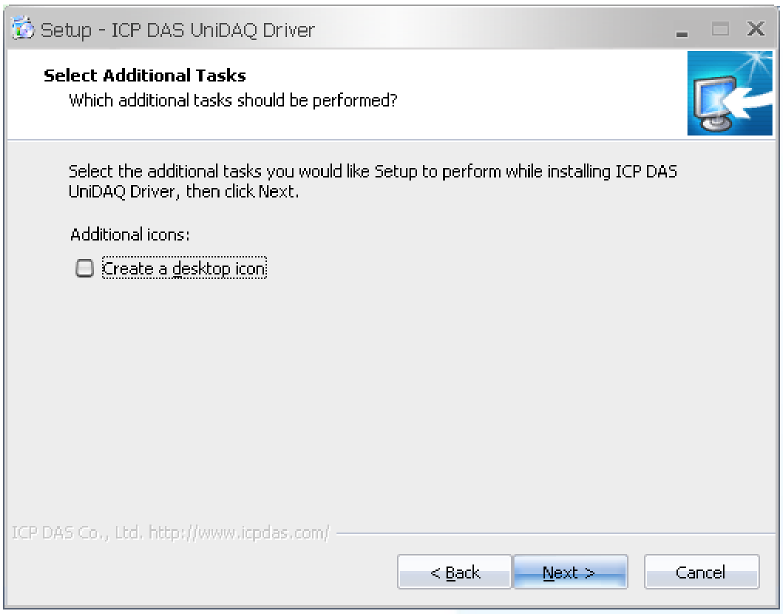
Custom Installation

- ☑ PCI-822/826 series
- ☑ PCI-FC16U series
- ⊟ Optional I/O Board
  - ☑ PIO-D24/D56 series
  - □ PIO-D48 series
  - □ PIO-D64 series
  - □ PIO-D96 series
  - □ PIO-D144 series
  - □ PIO-D168 series

Current selection requires at least 3.0 MB of disk space.

ICP DAS Co., Ltd. http://www.icpdas.com/

6. On the "Select Additional Tasks" screen, click the “Next >” button to continue.



Setup - ICP DAS UniDAQ Driver

Select Additional Tasks

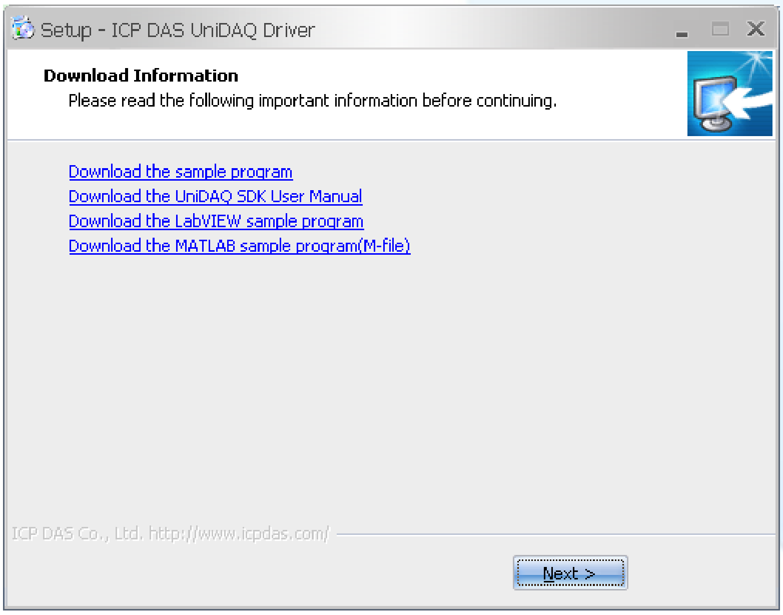
Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing ICP DAS UniDAQ Driver, then click Next.

Additional icons:

□ Create a desktop icon

7. If you wish to download the demo programs, click the relevant link on the "Download Information" screen, and then click the “Next >” button to continue.



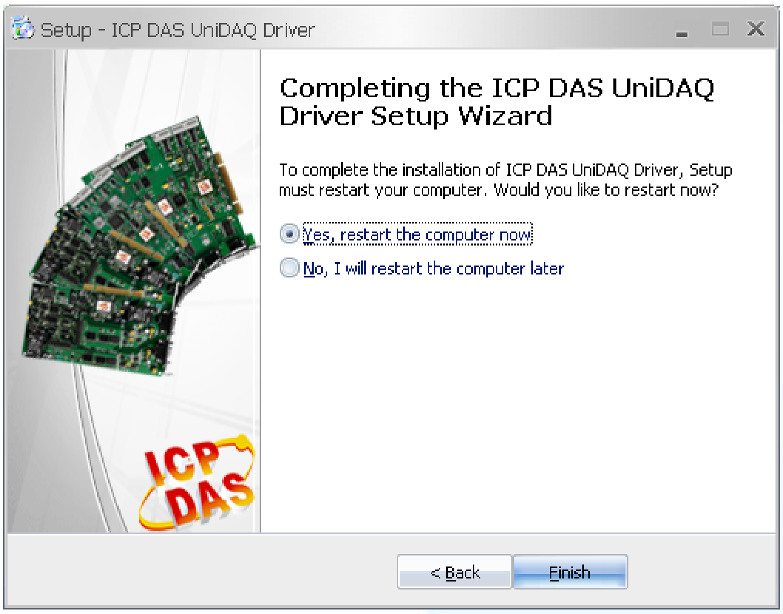
Setup - ICP DAS UniDAQ Driver

Download Information

Please read the following important information before continuing.

Download the sample program
Download the UniDAQ SDK User Manual
Download the LabVIEW sample program
Download the MATLAB sample program(M-file)

8. Select the “Yes, restart the computer now” radio button. Ensure that any open programs are closed and you have saved your work, and then click the “Finish” button. The system will then reboot to complete the installation of the ICP DAS UniDAQ Driver DLL.



Setup - ICP DAS UniDAQ Driver

Completing the ICP DAS UniDAQ Driver Setup Wizard

To complete the installation of ICP DAS UniDAQ Driver, Setup must restart your computer. Would you like to restart now?

● Yes, restart the computer now

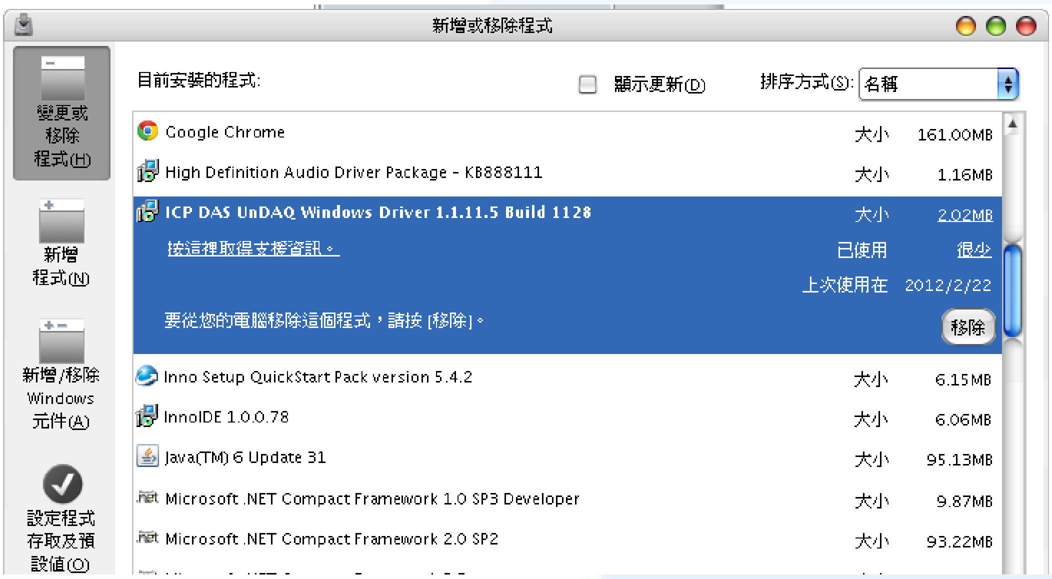
○ No, I will restart the computer later

2.3. Uninstalling the UniDAQ Driver DLL

The ICP DAS UniDAQ Driver DLL includes a utility that allows the software to be removed from your computer. To uninstall the software, follow the procedure described below:

1. Open the Control Panel by clicking “Start” button and then clicking “Control Panel”. Double-click the “Add/Remove Programs” icon to open “Add/Remove Programs” dialog.

2. In the “Add/Remove Programs” dialog, click the “Change or Remove Programs” tab, and then click the ”ICP DAS UniDAQ Windows Driver” item. Click the “Remove” button to begin the uninstall process.

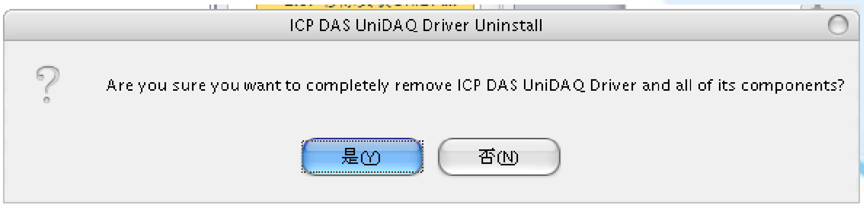


新增或移除程式

目前安裝的程式: □ 顯示更新(D) 排序方式(S): 名稱

Google Chrome	大小	161.00MB
High Definition Audio Driver Package - KB888111	大小	1.16MB
ICP DAS UnDAQ Windows Driver 1.1.11.5 Build 1128	大小	2.02MB
按這裡取得支援資訊。	已使用	很少
	上次使用在	2012/2/22
要從您的電腦移除這個程式，請按 [移除]。		移除
Inno Setup QuickStart Pack version 5.4.2	大小	6.15MB
InnoIDE 1.0.0.78	大小	6.06MB
Java(TM) 6 Update 31	大小	95.13MB
Microsoft .NET Compact Framework 1.0 SP3 Developer	大小	9.87MB
Microsoft .NET Compact Framework 2.0 SP2	大小	93.22MB

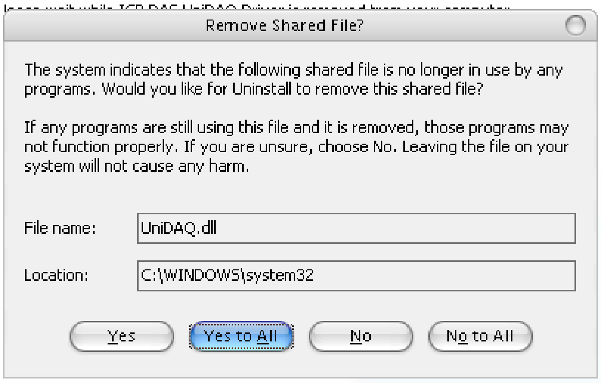
3. A prompt will be displayed asking you to confirm that you wish to remove the UniDAQ Windows Driver. Click the “Yes” button to continue.



ICP DAS UniDAQ Driver Uninstall

Are you sure you want to completely remove ICP DAS UniDAQ Driver and all of its components?

4. When the “Remove Shared Files” dialog is displayed, click the “Yes to All” button to continue.



Remove Shared File?

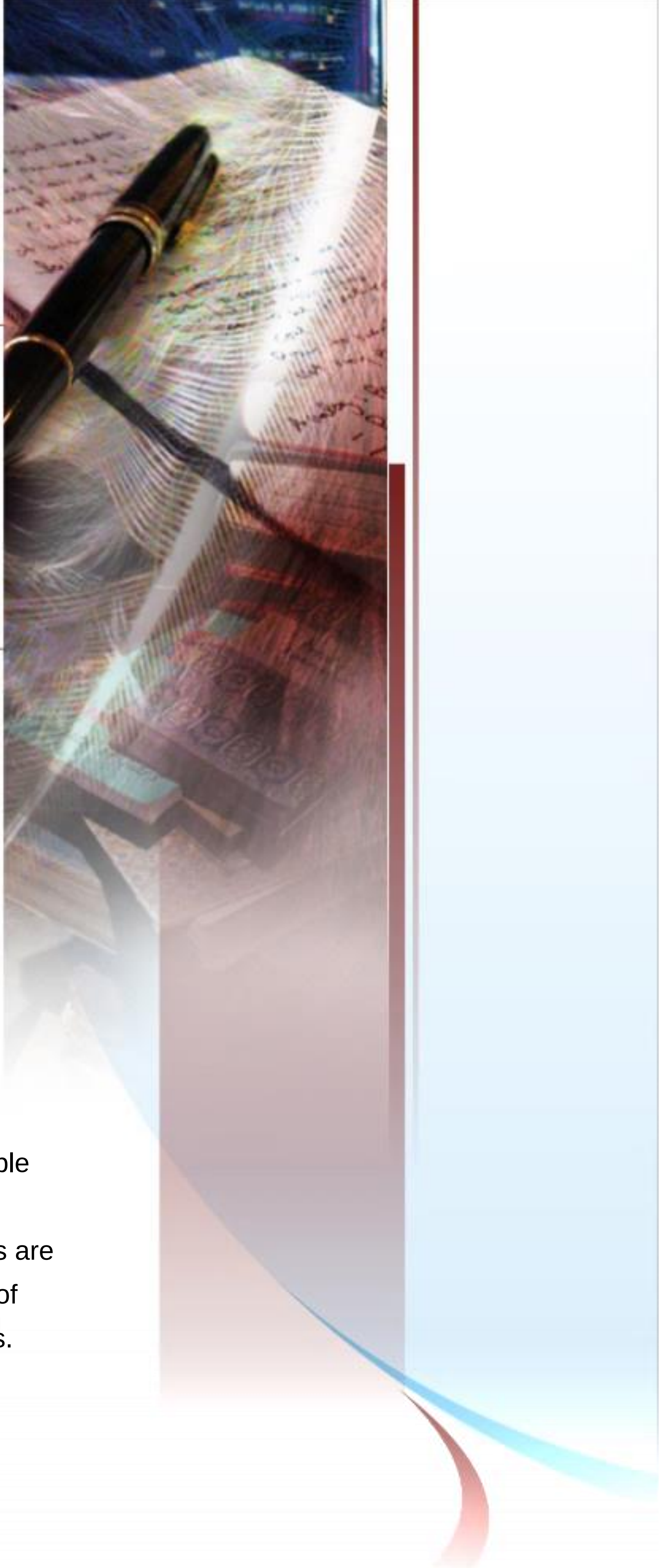
The system indicates that the following shared file is no longer in use by any programs. Would you like for Uninstall to remove this shared file?

If any programs are still using this file and it is removed, those programs may not function properly. If you are unsure, choose No. Leaving the file on your system will not cause any harm.

File name: UniDAQ.dll

Location: C:\WINDOWS\system32

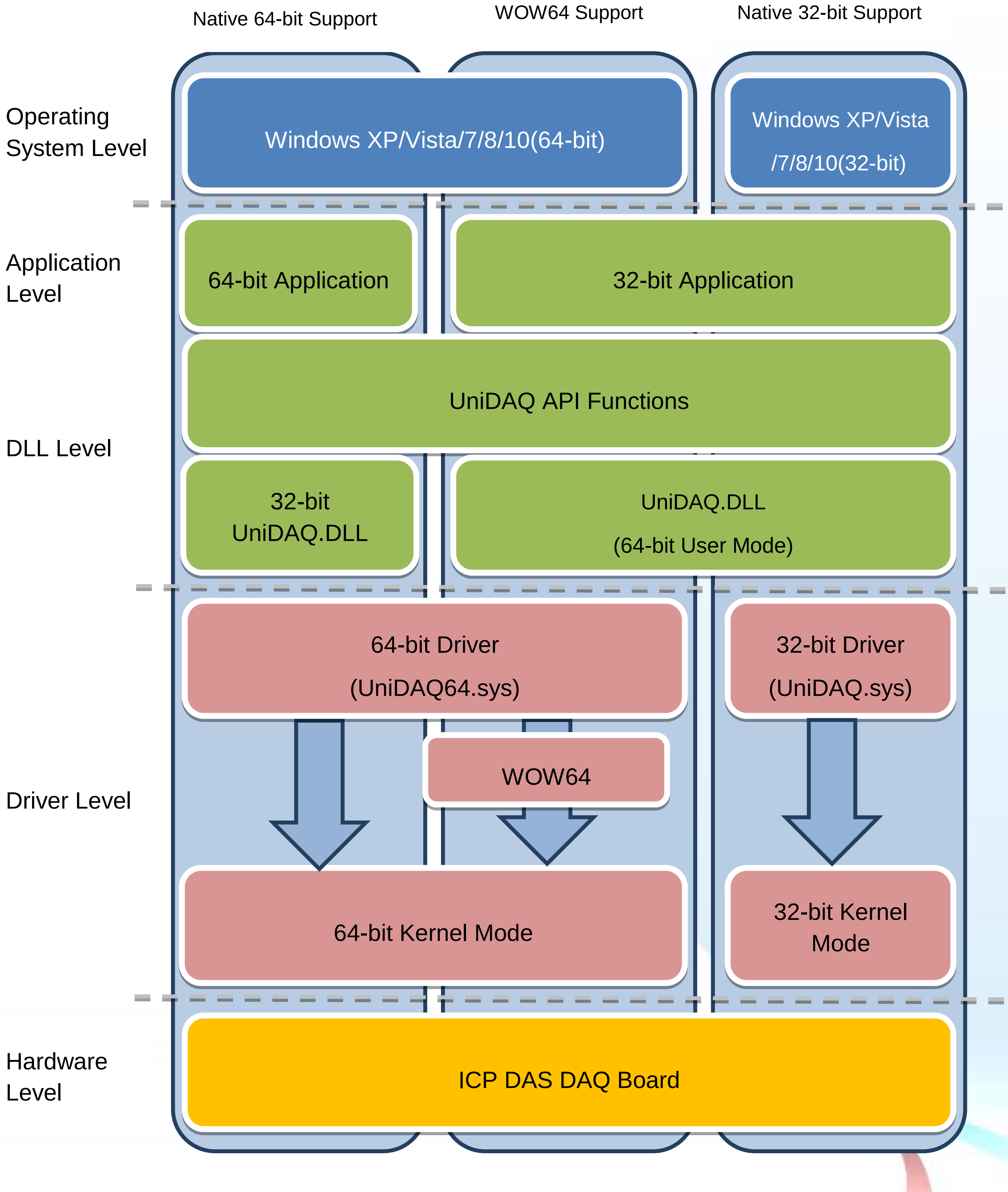
5. Once the removal process is complete, a dialog box will be displayed to notify that the UniDAQ Driver was successfully removed. Click the “OK” button to finish.



3. Tutorial

This chapter provides an overview of creating a simple application. Step-by-step implementation procedures are also included for a variety of development environments.

3.1. Application Structure

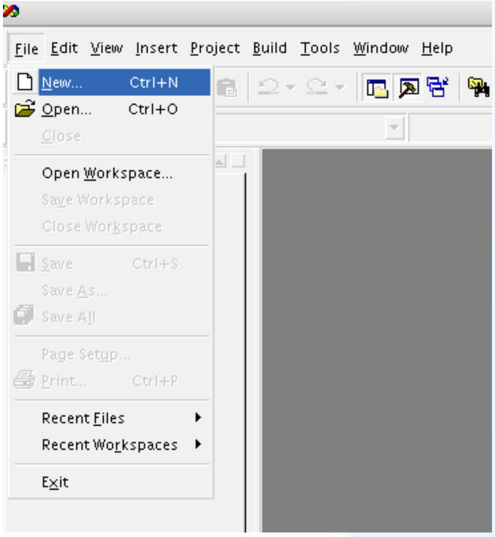


3.2. Creating a Win32 Console Application

The following procedure describes how to create a Win32 Console application based on the UniDAQ DLL. Note that this description is based on Microsoft Visual Studio 6.0.

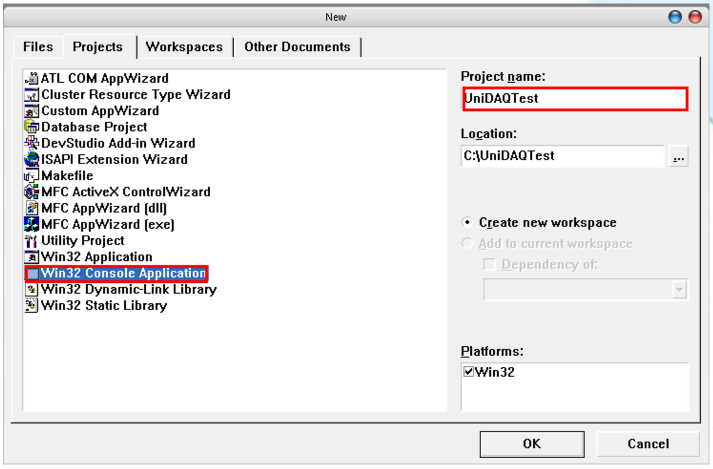
Creating the Application

1. Open Microsoft Visual Studio to create a new Visual C++ 6.0 project, and click File from the main menu, and then click New. Alternatively, press CTRL + N.



2. Click the Projects tab, and then specify the Project Name, Location, Workspace, Dependency, and Platforms options.

   Click the "Win32 Console Application" entry in the Projects List pane, an and enter “UniDAQTest” in the Project name field. The Location field indicates where the project files will be stored. Verify that the details are correct, and then click the "OK" button to continue.



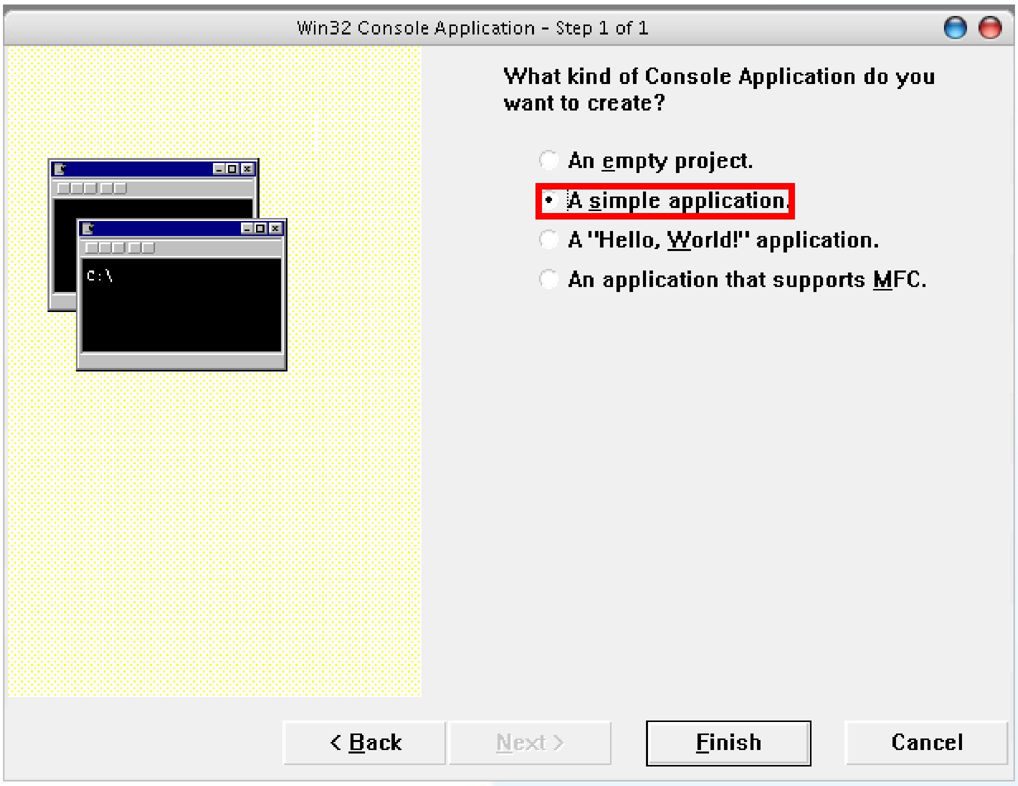
Project name: UniDAQTest

Location: C:\UniDAQTest

● Create new workspace

Platforms: ☑Win32

3. In Step 1 of the project creation wizard, specify the level of file support you want for the project. Click the “A simple application” option, and then click the "Finish" button. Visual Studio will then generate the folder structure and basic source code for the project.



Win32 Console Application - Step 1 of 1

What kind of Console Application do you want to create?

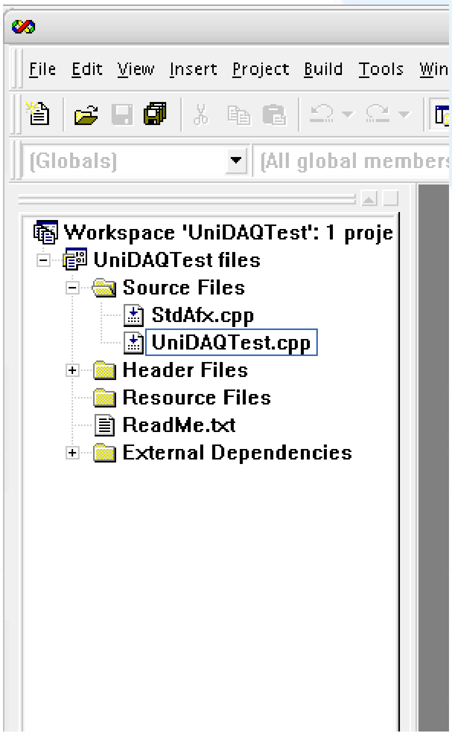
○ An empty project.

● A simple application.

○ A "Hello, World!" application.

○ An application that supports MFC.

4. Once the project has been created, open the "Source Files" folder in the Navigation pane, and double-click the UniDAQTest.cpp file to open the code editing window.



Workspace 'UniDAQTest': 1 proje
UniDAQTest files
Source Files
StdAfx.cpp
UniDAQTest.cpp
Header Files
Resource Files
ReadMe.txt
External Dependencies

5. Enter the following codes for the UniDAQTest.cpp.

```
#include "stdafx.h"
#include "stdio.h"
#include "UniDAQ.h" //Include the UniDAQ header file
#pragma comment(lib,"UniDAQ.lib") //Include the UniDAQ library file

WORD wRtn;
WORD wBoardNo;
WORD wTotalBoards;

int main(int argc, char* argv[])
{
 WORD wOutPortNo;

 //Initialize the resource and read total number of boards form driver
 wRtn=Ixud_DriverInit(&wTotalBoards);
 if (wRtn!=Ixud_NoErr)
 {
      printf("\nDriver Init Error(%d)",wRtn);
      return wRtn;
 }
 printf("Write the DO Value 0xFF");
 wBoardNo=0;
 wOutPortNo=0;
 //Write the DO
 wRtn = Ixud_WriteDO(wBoardNo,wOutPortNo,0xFF);
 //Release the resources from driver
 wRtn = Ixud_DriverClose();
 return 0;
}
```

Testing the application

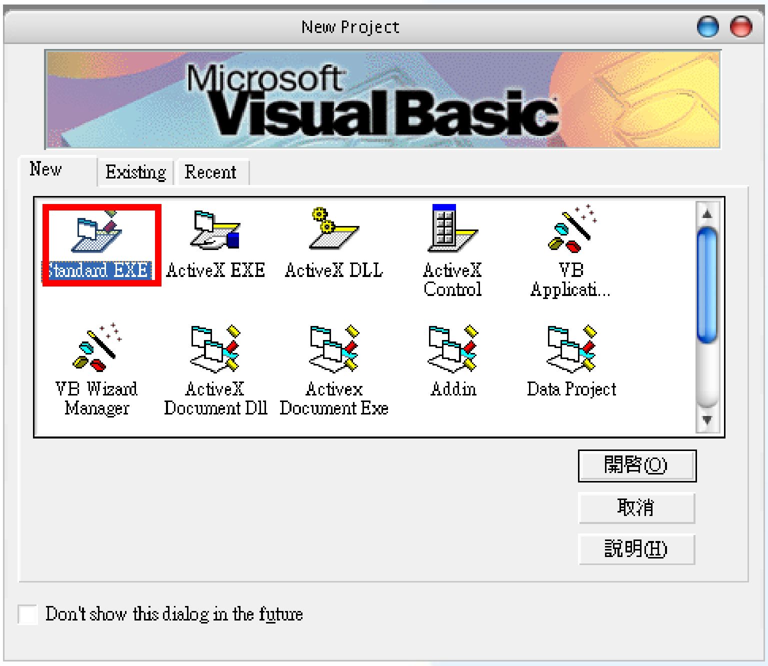
1. To compile your code, click Build from the main menu, and then click Compile, or press Ctrl + F7.
2. Execute the compiled application in a Command Prompt window.

3.3. Creating a Visual Basic Application

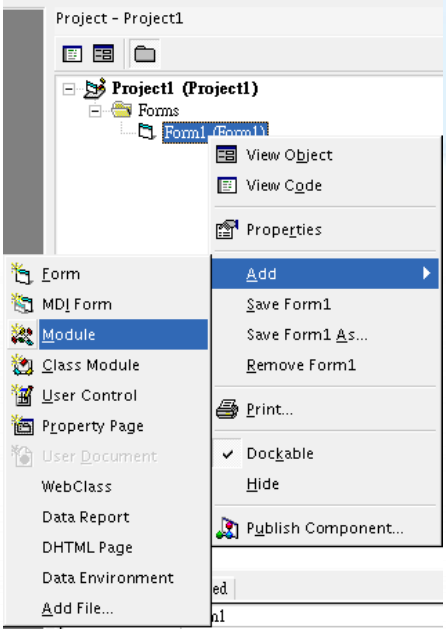
The following procedure describes how to create a Visual Basic application based on the UniDAQ DLL. Note that this description is based on Microsoft Visual Studio 6.0.

Creating the Application

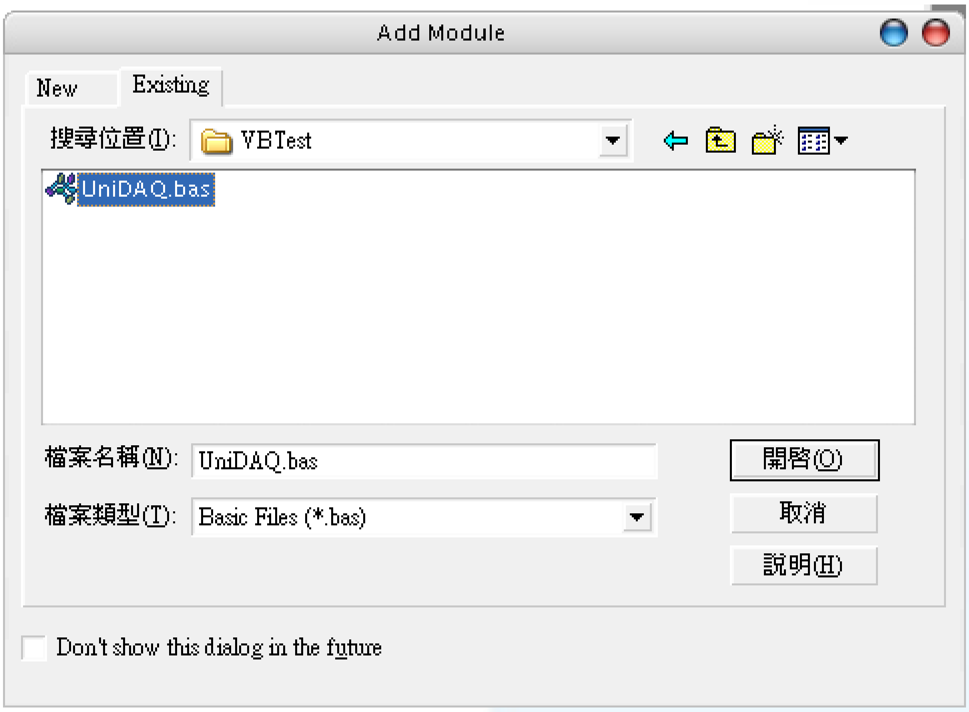
1. Open Microsoft Visual Studio to create a new Visual Basic project, and click the Standard.exe icon in the New Project window, and then click the Open button.



2. In the Project Explorer pane, right-click the name of the newly created form, point to Add in the menu, and then click Module to open the Add Module dialog box.



3. Add the UniDAQ.bas declaration file by clicking module by clicking on Add Module in the Project menu.



搜尋位置(I): VBTest

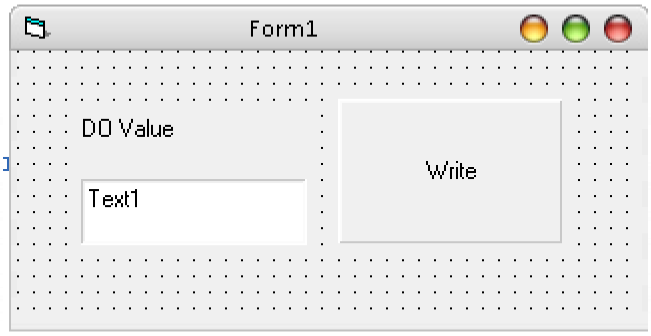
UniDAQ.bas

檔案名稱(N): UniDAQ.bas

檔案類型(T): Basic Files (*.bas)

□ Don't show this dialog in the future

4. The Form design screen will then be automatically displayed allowing you to design the Form. From the Toolbox, select a Label control and position it on the form. Click on the new control to open the Properties window for the Label, and then enter “DO Value” in the Caption field. Next, select a TextBox control from the Toolbox and position it on the Form. In the Properties window for the TextBox control, enter "txtDOVal". Finally, select a CommandButton control from the Toolbox and position it on the Form. In the Properties window for the CommandButton control, enter "cmdWrite" in the Name field, and enter "Write" in the Caption field. Your form should now look similar to the one shown in the image below:



5. Double click the CommandButton control on the Form to open the code editing window and then add the following code for the cmdWrite button:

```
Option Explicit
Dim wTotalBoards As Integer
Dim wBoardNo As Integer
Dim wOutPortNo As Integer
Dim wRtn As Integer

Private Sub cmdWrite_Click()

Dim wBoardIndex As Integer

‘//Initialize the resource and read total board number form driver
wRtn = Ixud_DriverInit(wTotalBoards)
If (wRtn) Then
    MsgBox ("Driver Initial Error.Error Code:" + Str(wRtn))
    End
End If

wBoardNo =0;
wOutportNo =0;

'//Write the DO Value
wRtn = Ixud_WriteDO(wBoardNo, wOutPortNo, Val(txtDOVal.Text))

‘//Release the resource form driver
wRtn = Ixud_DriverClose()
End Sub
```

Test the application

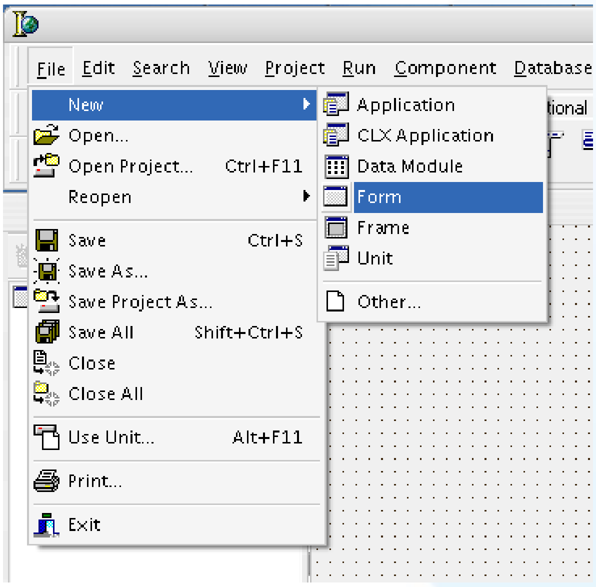
1. Run the application by either clicking the Start button on the toolbar, or by pressing F5.
2. Type "255" in the DO Value text box and then press the "Write" button to output a DO Value of 255.

3.4. Creating a Borland Delphi Application

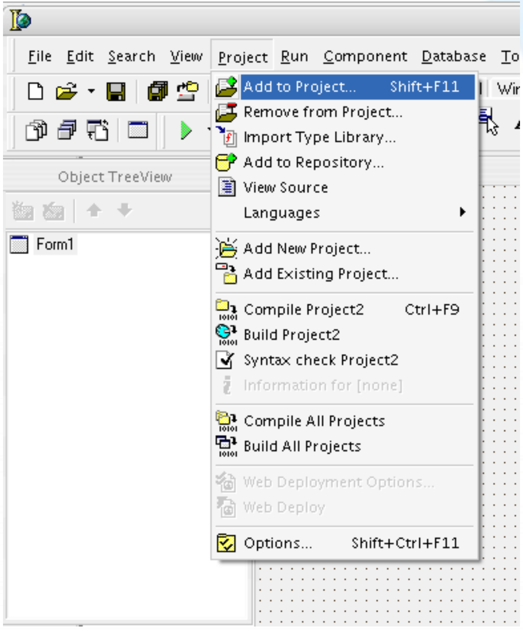
The following procedure describes how to create a Borland Delphi application based on the UniDAQ DLL. Note that this description is based on Borland Delphi version 6.

Creating the Application

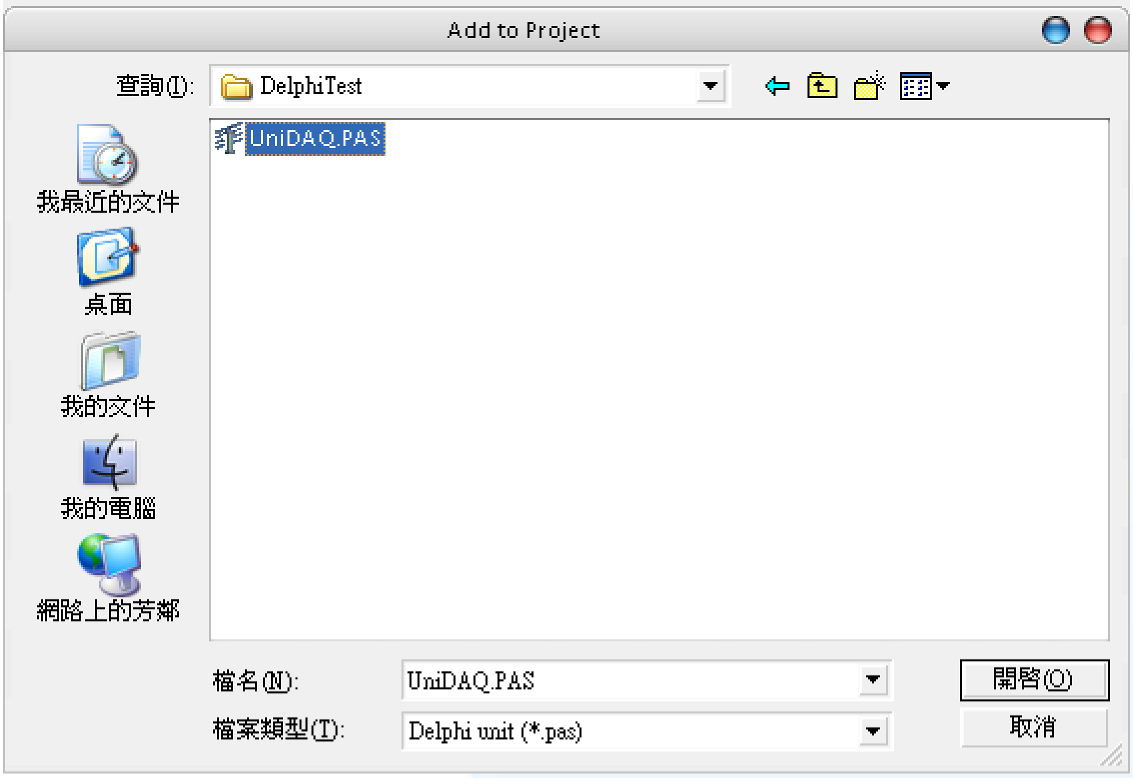
1. Open Borland Delphi 6, and click File from the main menu. Point to New and then click Form to create a new Delphi project.



2. From the main menu, click Project and then click Add to Project. Alternatively, press Shift + F11.



3. Add the UniDAQ.pas declaration file by clicking the name of the file and then clicking the Open button.



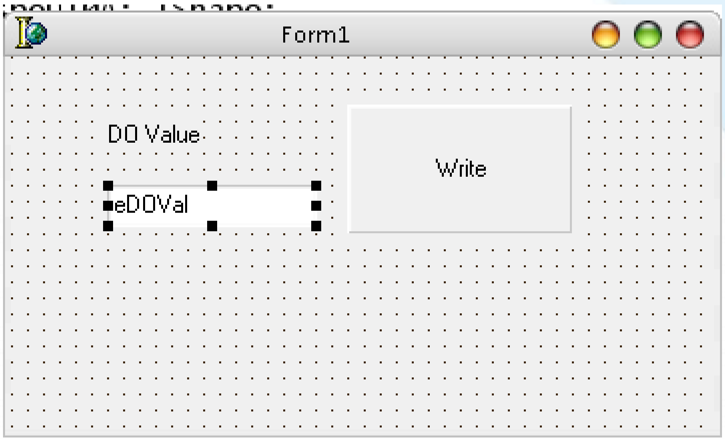
查詢(I): DelphiTest

UniDAQ.PAS

檔名(N): UniDAQ.PAS

檔案類型(T): Delphi unit (*.pas)

4. From the Component palette, select a Label control and position it on the form. Click on the new control to open the Object Inspector window for the Label, and then enter “DO Value” in the Caption field. Next, select an Edit control from the Component palette and position it on the Form. In the Object Inspector window for the Edit control, enter "eDOVal". Finally, select a Button control from the Component palette and position it on the Form. In the Object Inspector window for the Button control, enter "btnWrite" in the Name field, and enter "Write" in the Caption field. Your form should now look similar to the one shown in the image below:



5. Double click the btnWrite control on the Form to open the code editing window and then add the following code for the btnWrite button:

```
implementation
uses UniDAQ;

{$R *.dfm}

procedure TForm1.btnWriteClick(Sender: TObject);
var
   wTotalBoards,wRtn,wBoardNo,wOutportNo:Word;
   dwDOValue : LongInt;
begin
     //Initialize the resources and read the total number of boards from the driver
     wRtn := Ixud_DriverInit(wTotalBoards);
     If wRtn <> Ixud_NoErr Then
     begin
        Application.MessageBox('*** DriverInit Error! ***', 'Error' , IDOK);
        Exit;
     End;
     wBoardNo :=0;
     wOutportNo :=0;

     //Write the DO value
     wRtn:=Ixud_WriteDO(wBoardNo,wOutportNo,StrToInt(eDOVal.Text));

     //Release the resources from driver
     wRtn := Ixud_DriverClose;

end;

end.
```

Testing the application

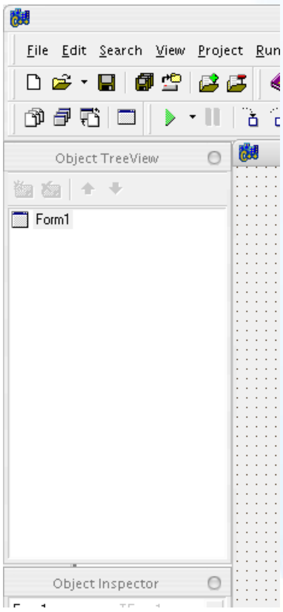
1. Run the application by either clicking the Start button on the toolbar, by clicking Run in the Run menu, or by pressing F9.
2. Type "255" in the DO Value text box and then press the "Write" button to output a DO Value of 255.

3.5. Creating a Borland C++ Builder Application

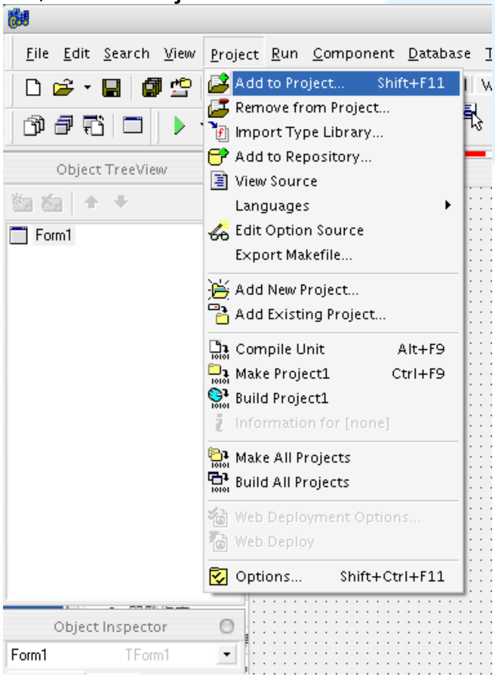
The following procedure describes how to create a Borland C++ application based on the UniDAQ DLL. Note that this description is based on Borland C++ Builder 6.

Creating the Application

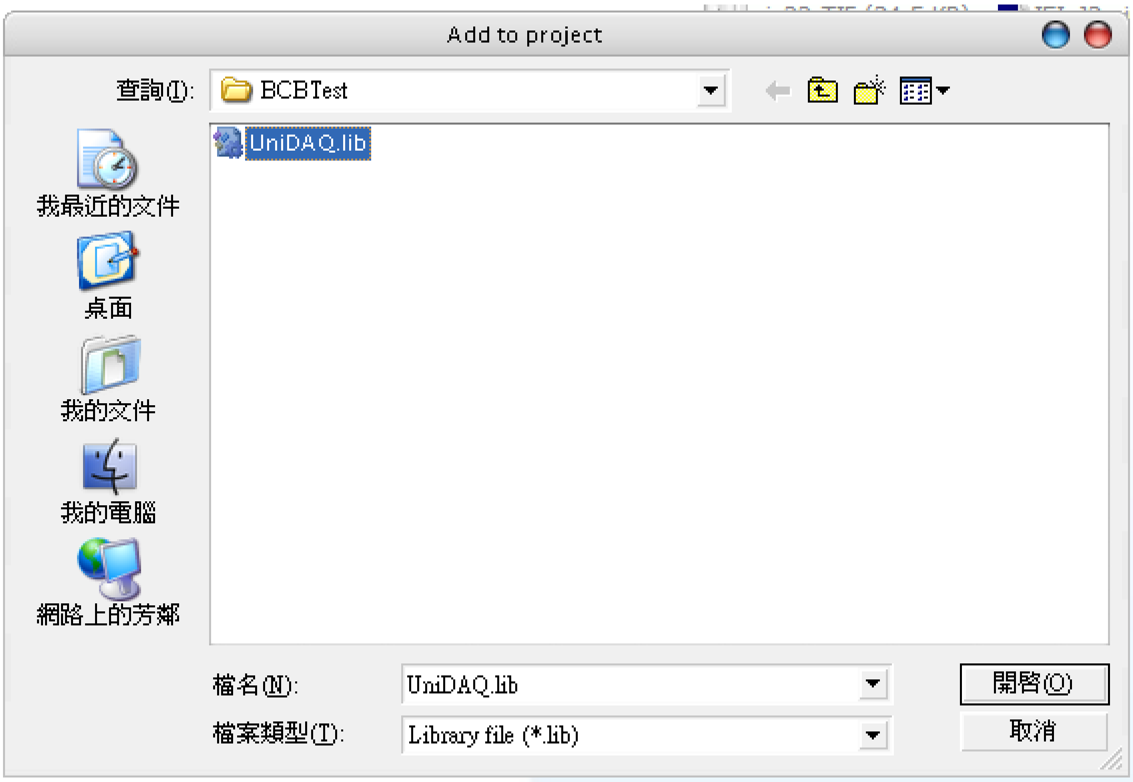
1. Open Borland C++ Builder 6, and click File from the main menu. Point to New and then click Form to create a new C++ project.



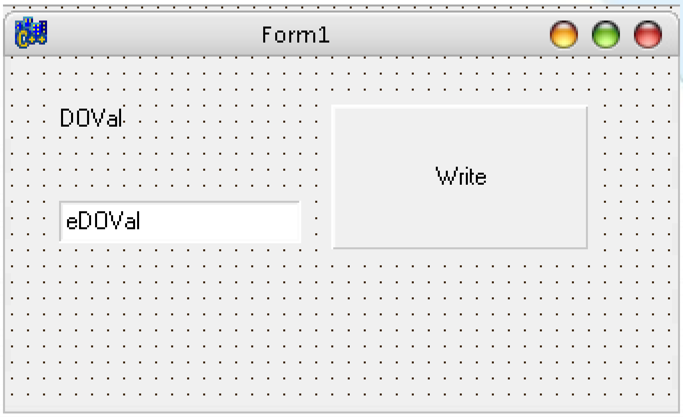
2. From the main menu, click Project and then click Add to Project. Alternatively, press Shift + F11.



3. Add the UniDAQ.lib declaration file by clicking the name of the file and then clicking the Open button.



4. The Form design screen will then be automatically displayed allowing you to design the Form. From the Component palette, select a Label control and position it on the form. Click on the new control to open the Object Inspector window for the Label, and then enter “DO Value” in the Caption field. Next, select an Edit control from the Component palette and position it on the Form. In the Object Inspector window for the Edit control, enter "eDOVal". Finally, select a Button control from the Component palette and position it on the Form. In the Object Inspector window for the Button control, enter "btnWrite" in the Name field, and enter "Write" in the Caption field. Your form should now look similar to the one shown in the image below:



5. Double click the btnWrite control on the Form to open the code editing window and then add the following code for the btnWrite button:

```
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "UniDAQ.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
__fastcall TForm1::TForm1(TComponent* Owner)
       : TForm(Owner)
{
}
void __fastcall TForm1::btnWriteClick(TObject *Sender)
{
Word wTotalBoard, wRtn ;
Word wOutPortNo;
Word wBoardNo;
//Initialize the resources and read the total number of boards from the driver
wRtn = Ixud_DriverInit(&wTotalBoard);
if ( wRtn != Ixud_NoErr )
{
    ShowMessage( "Driver Initial Err.Error Code:" + IntToStr(wRtn)) ;
}
wOutPortNo=0;
wBoardNo=0;
//Write the DO Value
wRtn=Ixud_WriteDO(wBoardNo,wOutPortNo,StrToInt(eDOVal->Text));

//Release the resources from driver
wRtn= Ixud_DriverClose();
}
```

Test the application

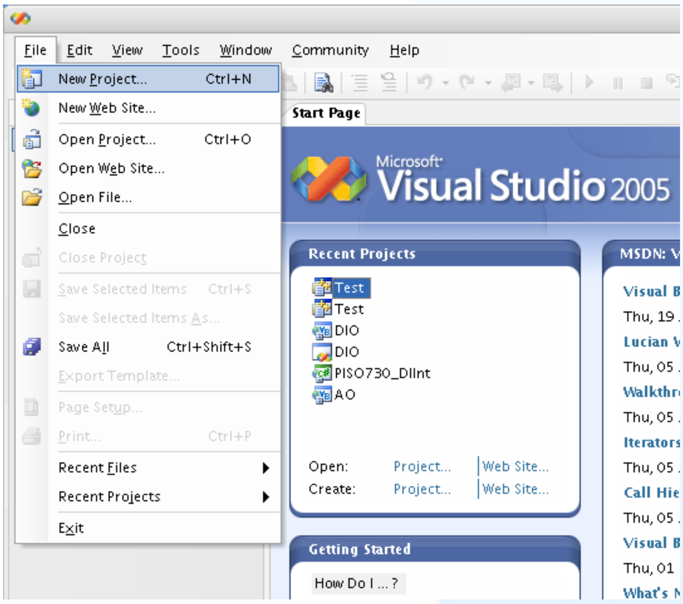
1. Run the application by either clicking the Start button on the toolbar, by clicking Run in the Run menu, or by pressing F9.
2. Type "255" in the DO Value text box and then press the "Write" button to output a DO Value of 255.

3.6. Creating a Visual C++.NET Application

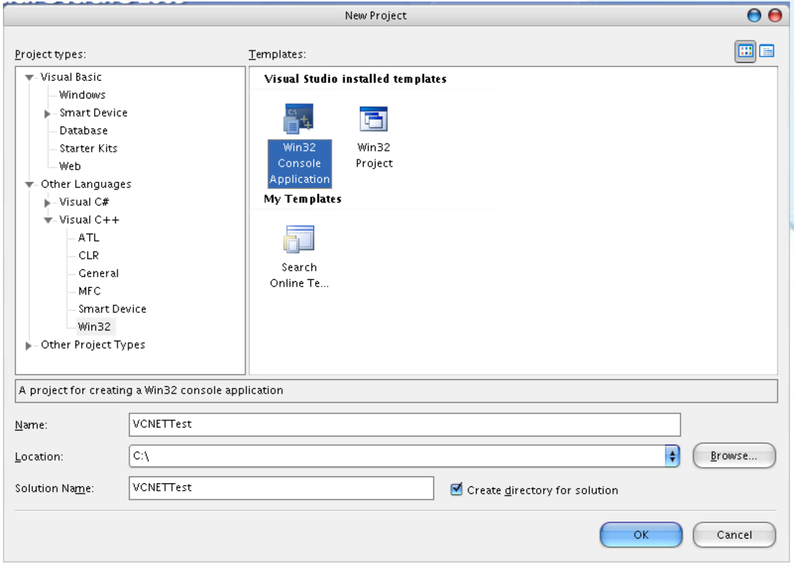
The following procedure describes how to create a Visual C++.NET application based on the UniDAQ DLL. Note that this description is based on Microsoft Visual Studio 2005.

Creating the Application

1. Open Microsoft Visual Studio 2005, and click File from the main menu and then click New Project to create a new Visual C++.NET project.



2. Once the New Project dialog box is displayed, click the "Other Languages" item in the Project types pane, click "Visual C++", and then click the "Win32" option. In the Templates pane, click the Win32 Console Application project template, enter “VCNETTest” in the Name field, and then click the OK button.



A project for creating a Win32 console application

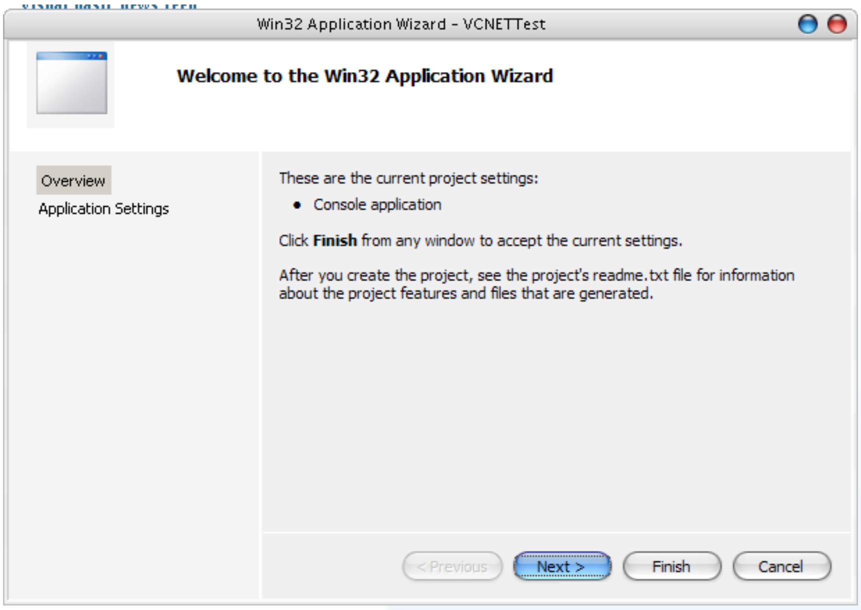
Name: VCNETTest

Location: C:\

Solution Name: VCNETTest

☑ Create directory for solution

3. When the Win32 Application Wizard is displayed indicating the current project settings. Click the "Finish" button to continue. Visual Studio will then generate the folder structure and basic source code for the project.



Welcome to the Win32 Application Wizard

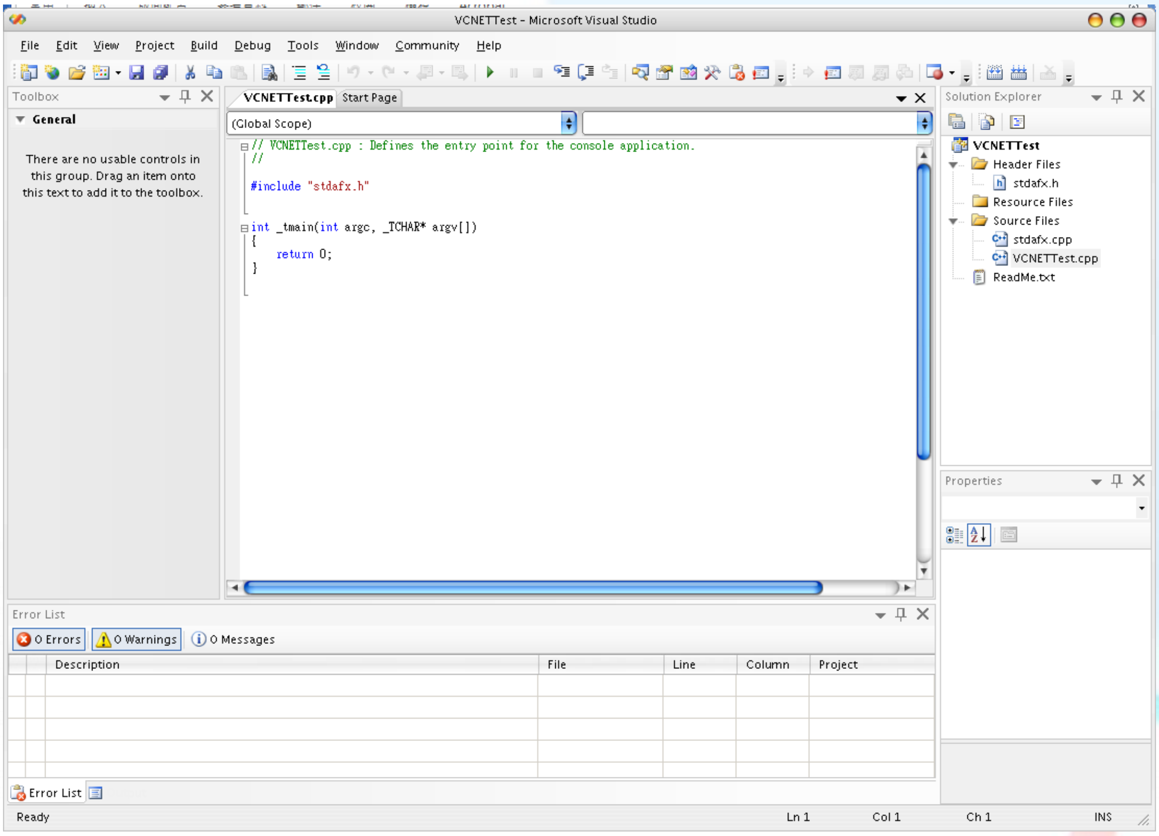
These are the current project settings:

- Console application

Click **Finish** from any window to accept the current settings.

After you create the project, see the project's readme.txt file for information about the project features and files that are generated.

4. Double click the VCNETTest.cpp of Solution Explorer to open the codes writing windows.



There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

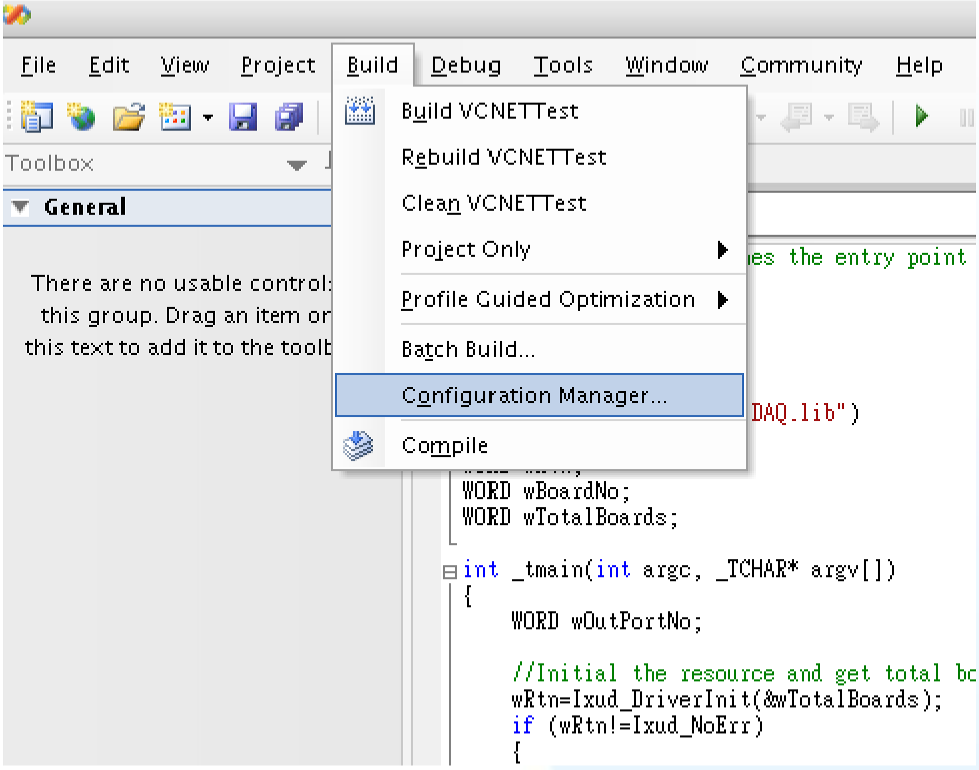
```
// VCNETTest.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"

int _tmain(int argc, _TCHAR* argv[])
{
    return 0;
}
```

5. In the code editing window, add the following code:

```
// VCNETTest.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include "stdio.h"
#include "UniDAQ.h"
#pragma comment(lib,"UniDAQ.lib")

WORD wRtn;
WORD wBoardNo;
WORD wTotalBoards;

int _tmain(int argc, _TCHAR* argv[])
{
 WORD wOutPortNo;

 //Initialize the resources and read total number of boards form  driver
 wRtn=Ixud_DriverInit(&wTotalBoards);
 if (wRtn!=Ixud_NoErr)
 {
      printf("\nDriver Initialization Error.(%d)",wRtn);
      return wRtn;
 }
 printf("Write DO Value 0xFF");
 wBoardNo=0;
 wOutPortNo=0;
 //Write the DO value
 wRtn = Ixud_WriteDO(wBoardNo,wOutPortNo,0xFF);
 //Release the resources from driver
 wRtn = Ixud_DriverClose();
 return 0;
}
```

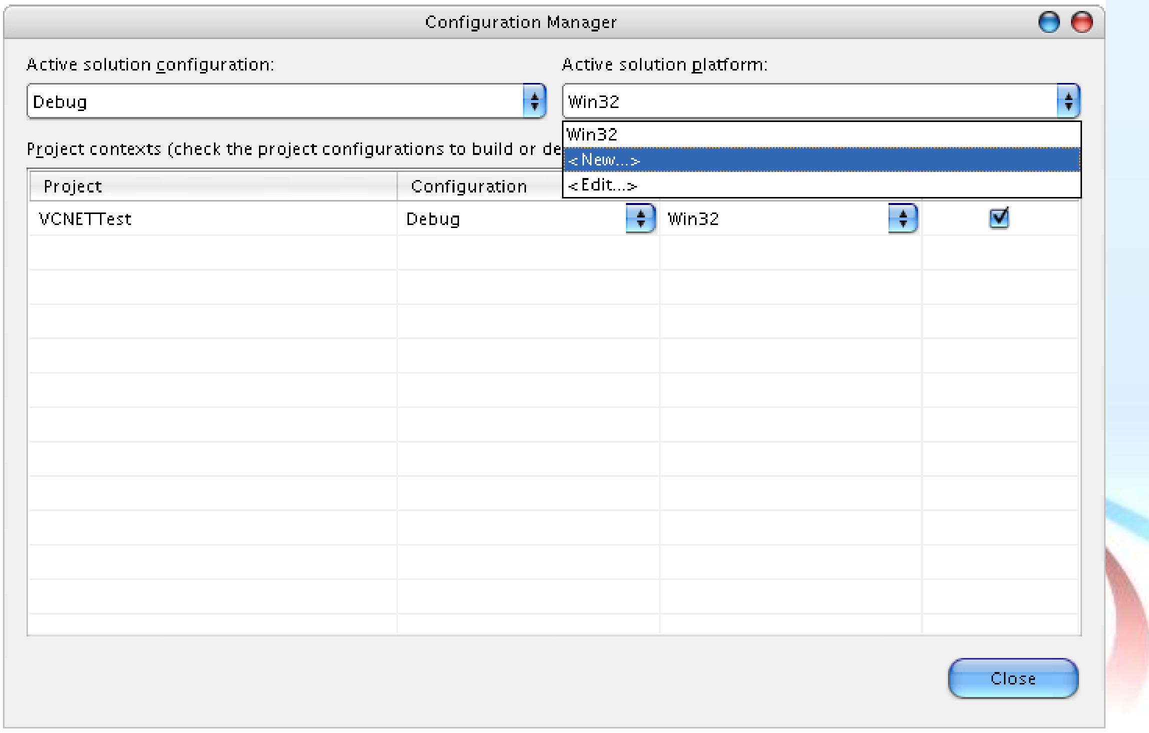
Compiling the Application

1. Click on Configuration Manager in the Build menu to open the Configuration Manager dialog box.



```
es the entry point

DAQ.lib")

WORD wBoardNo;
WORD wTotalBoards;

int _tmain(int argc, _TCHAR* argv[])
{
    WORD wOutPortNo;

    //Initial the resource and get total bo
    wRtn=Ixud_DriverInit(&wTotalBoards);
    if (wRtn!=Ixud_NoErr)
    {
```

2. In the Configuration Manager dialog box, select the <New...> option from the Active solution platform dropdown menu to open the New Solution Platform dialog box.



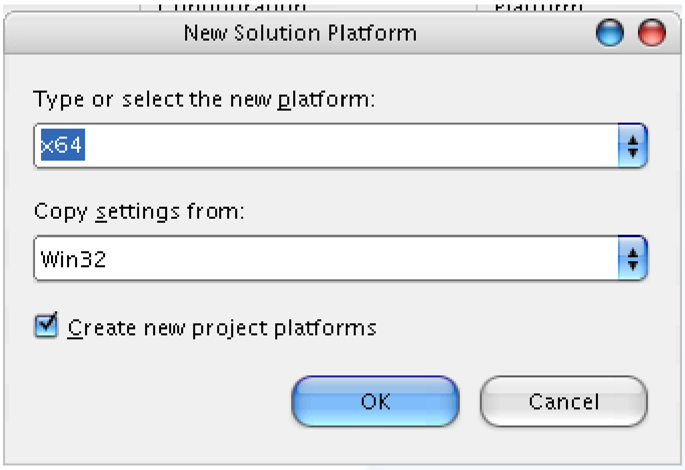
Active solution configuration: Debug

Active solution platform: Win32

Project contexts (check the project configurations to build or de

Project	Configuration		
VCNETTest	Debug	Win32	☑

3. In the New Solution Platform dialog box, select the required platform from the "Type or select the new platform" dropdown menu. Confirm the settings in the dialog then click the OK button to create a new configuration for the x64 platform and return to the Configuration Manager dialog box.



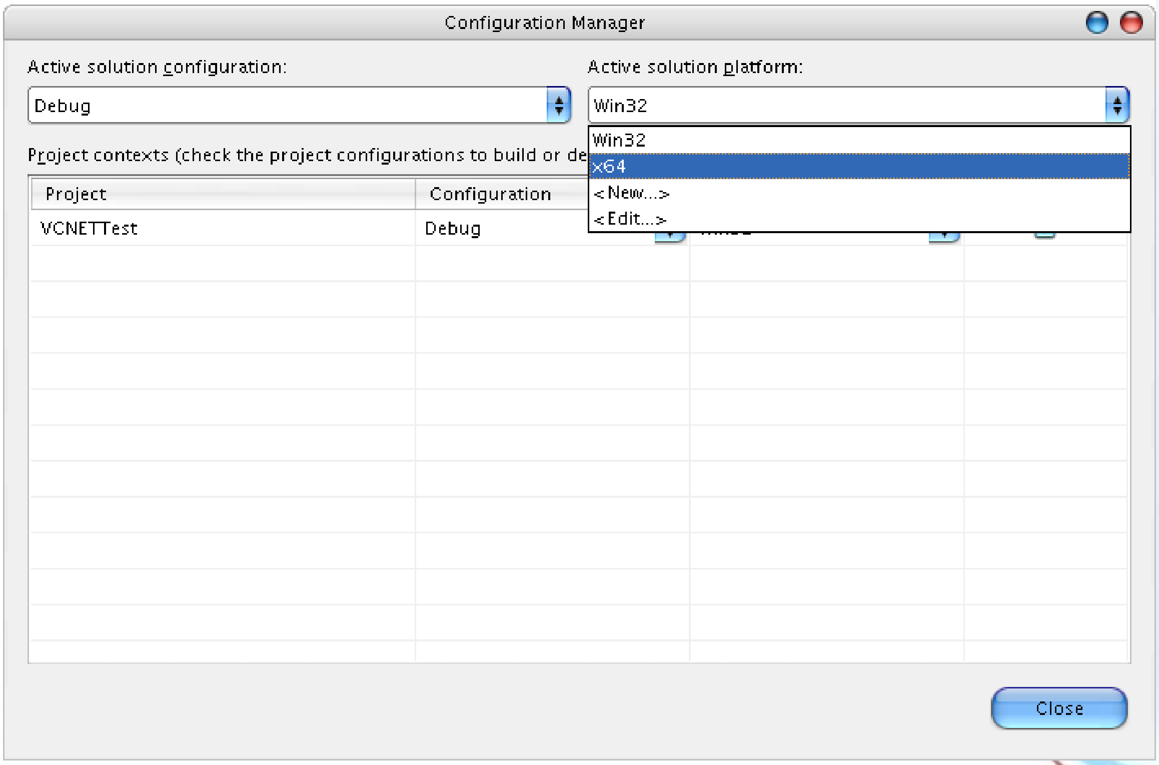
New Solution Platform

Type or select the new platform: x64

Copy settings from: Win32

☑ Create new project platforms

4. In the Configuration Manager dialog box, check that the details for the application configuration are correct. Note that if application is intended to be 64-bit, the x64 platform must selected. If the application is intended to be 32-bit, the Win32 (x86) platform must selected. Confirm the details and then click the Close button.



Configuration Manager

Active solution configuration: Debug

Active solution platform: Win32

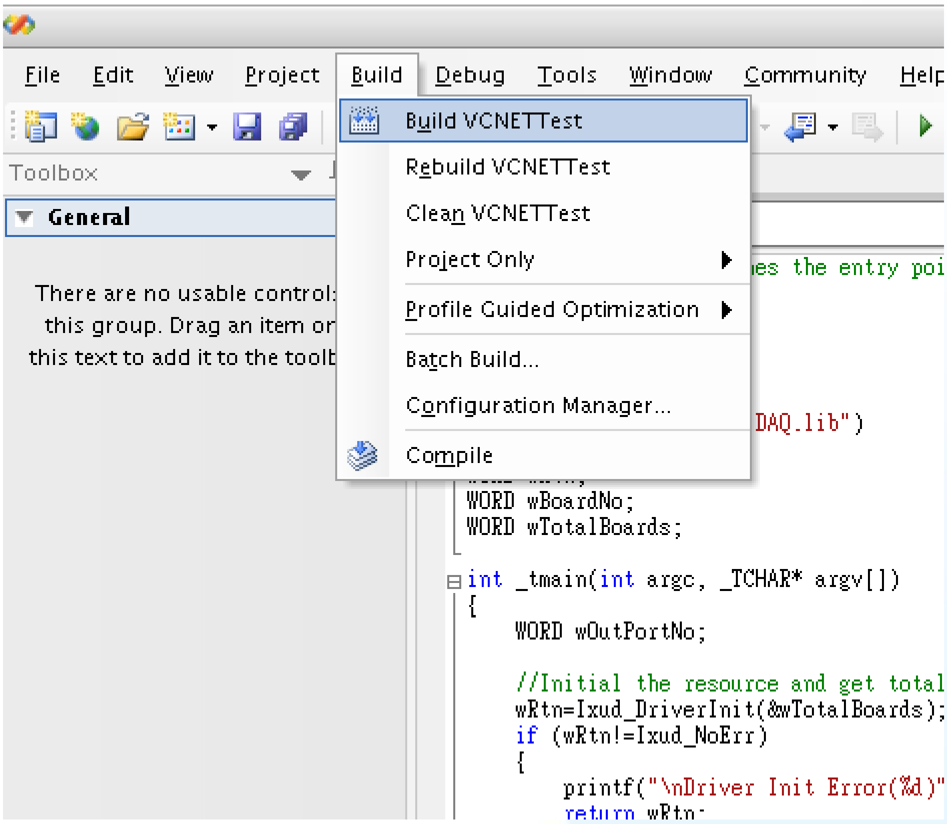
Project contexts (check the project configurations to build or de

Project	Configuration
VCNETTest	Debug

The 64-bit UniDAQ.lib file must be included for 64-bit applications

The 32-bit UniDAQ.lib file must be included for 32-bit applications

5. To build your VCNETTest application, click the Build VCNETTest option from the Build menu.



```
es the entry poi

DAQ.lib")

WORD wBoardNo;
WORD wTotalBoards;

int _tmain(int argc, _TCHAR* argv[])
{
    WORD wOutPortNo;

    //Initial the resource and get total
    wRtn=Ixud_DriverInit(&wTotalBoards);
    if (wRtn!=Ixud_NoErr)
    {
        printf("\nDriver Init Error(%d)"
        return wRtn;
```

Testing the Application

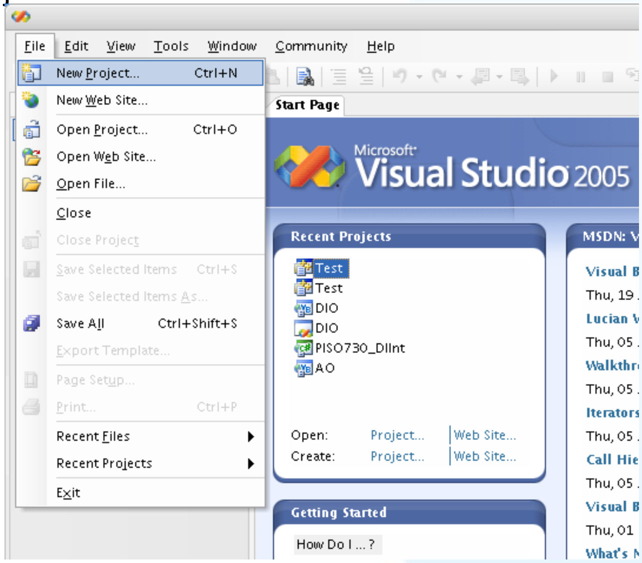
Execute the compiled application in a Command Prompt window.

3.7. Creating a Visual Basic.NET Application

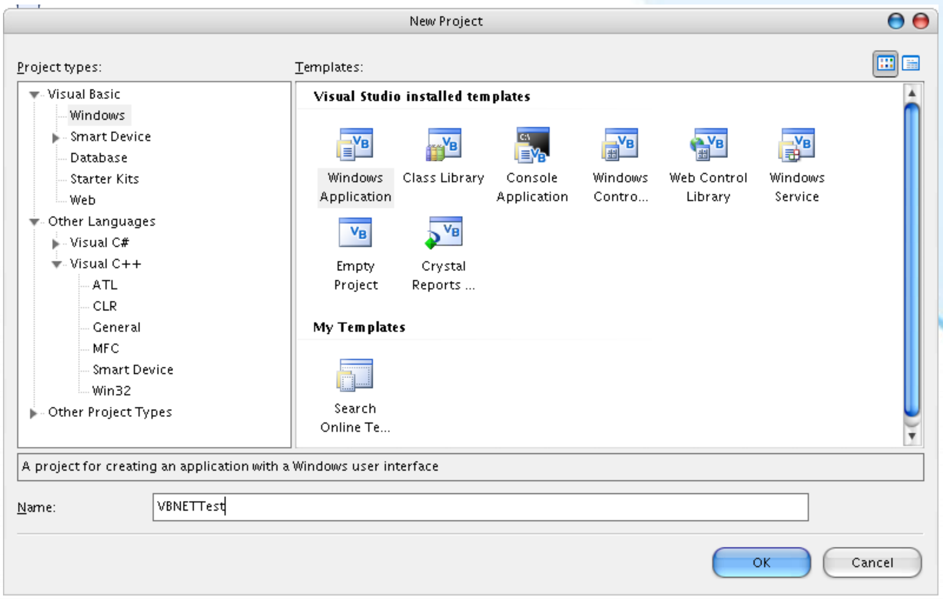
The following procedure describes how to create a Visual Basic.NET application based on the UniDAQ DLL. Note that this description is based on Microsoft Visual Studio 2005.

Creating the Application

1. Open Microsoft Visual Studio 2005, and click File from the main menu and then click New Project to create a new Visual Basic.NET.



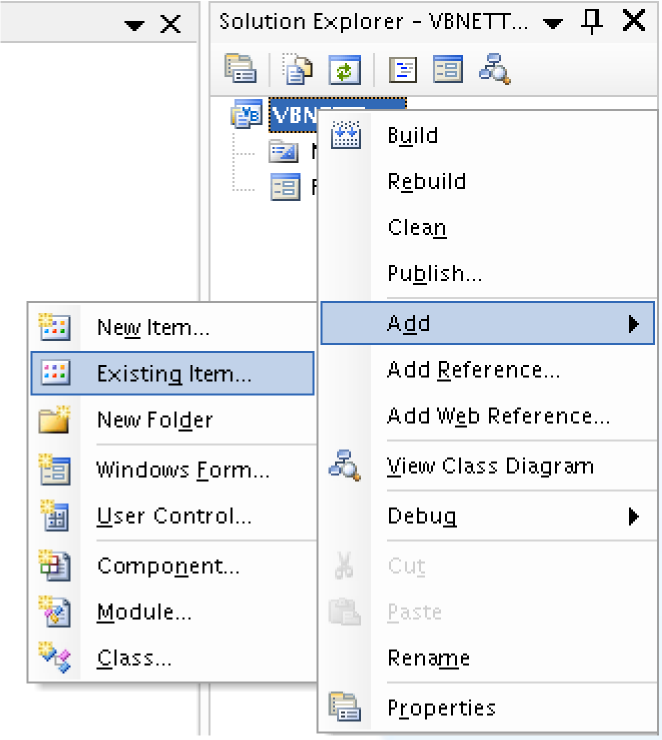
2. Once the New Project dialog box is displayed, click the "Visual Basic" item in the Project types pane, and then click the "Windows" option. In the Templates pane, click the Windows Application project template, enter “VBNETTest” in the Name field, and then click the OK button to create the new Visual Basic.NET project.



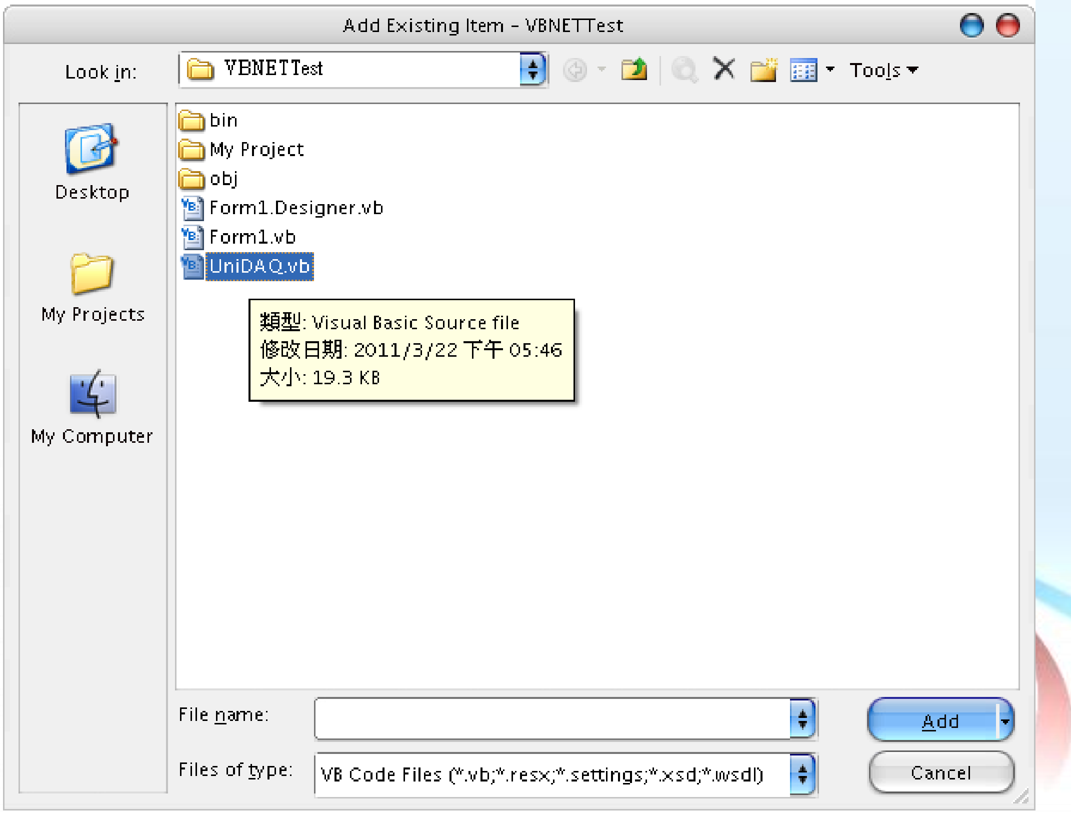
A project for creating an application with a Windows user interface

Name: VBNETTest

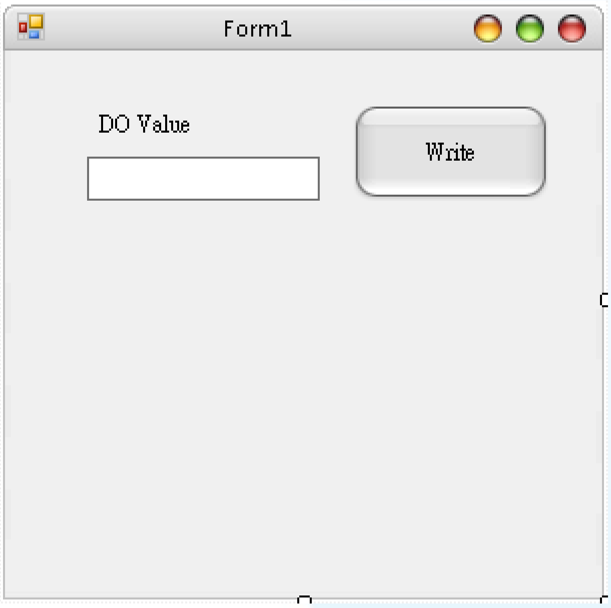
3. Once the project has been created, right-click the name of the newly created project in the Solutions Explorer pane, point to Add in the menu, and then click Existing Item option to open the Add Existing Item dialog box for the VBNETTest project.



4. Add the UniDAQ.vb declaration file by clicking the name of the file and then clicking the Add button.



5. D The Form design screen will then be automatically displayed allowing you to design the Form. From the Toolbox, select a Label control and position it on the form. Click on the new control to open the Properties window for the Label, and then enter “DO Value” in the Text field. Next, select a TextBox control from the Toolbox and position it on the Form. In the Properties window for the TextBox control, enter "txtDOVal". Finally, select a Button control from the Toolbox and position it on the Form. In the Properties window for the Button control, enter "btnWrite" in the Name field, and enter "Write" in the Text field. Your form should now look similar to the one shown in the image below:

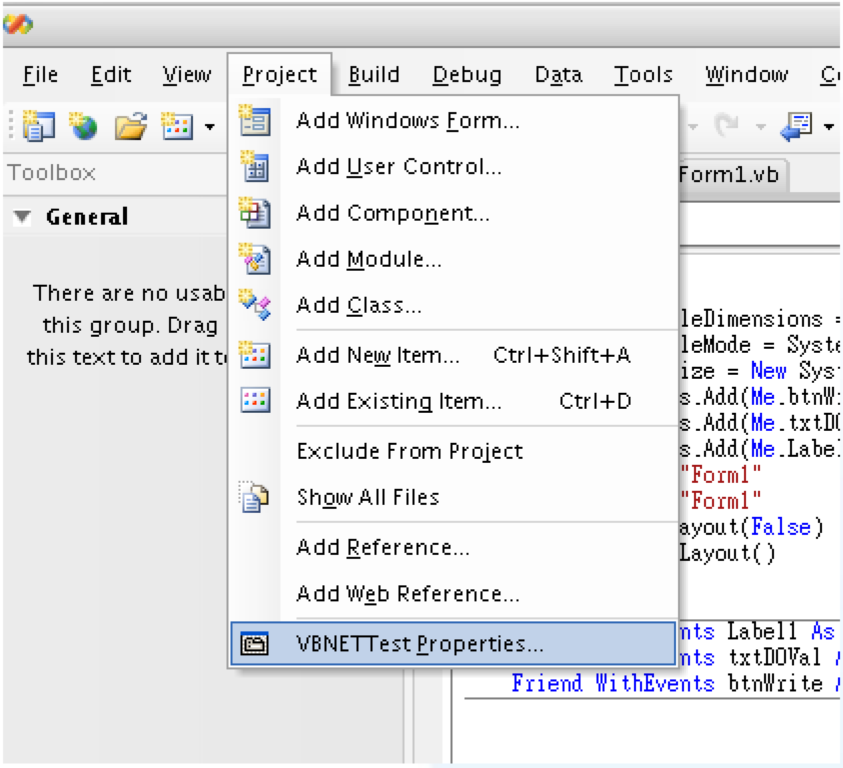


6. The btnWrite control on the Form to open the code editing window and then add the following code for the btnWrite button:

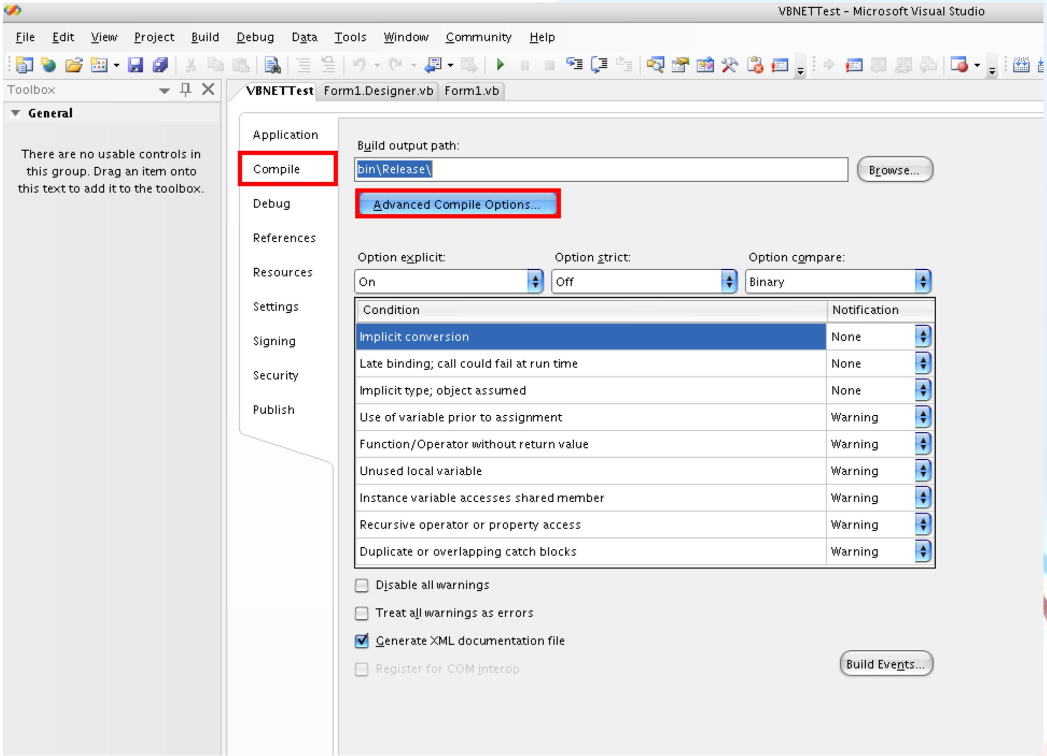
```
Private Sub btnWrite_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnWrite.Click
        Dim wTotalBoards As UInteger
        Dim wBoardNo As UInteger
        Dim wOutPortNo As UInteger
        Dim wRtn As UInteger

        '//Driver Initial
        wRtn = Ixud_DriverInit(wTotalBoards)
        If (wRtn) Then
            MsgBox("Driver Initial Error!!Error Code:" + Str(wRtn))
            End
        End If
        '//Write DO
        wRtn = Ixud_WriteDO(wBoardNo, wOutPortNo, Val(txtDOVal.Text))

        wRtn = Ixud_DriverClose()
End Sub
```

Compiling the Application

1. From the main menu, click Project, and then click “VBNETTest Properties” to display the Compile options dialog box.



2. Compile options dialog box, click the "Advanced Compile Options” button to open the “Advanced Compiler Settings” dialog box.



Build output path: bin\Release\

Option explicit: On

Option strict: Off

Option compare: Binary

Condition	Notification
Implicit conversion	None
Late binding; call could fail at run time	None
Implicit type; object assumed	None
Use of variable prior to assignment	Warning
Function/Operator without return value	Warning
Unused local variable	Warning
Instance variable accesses shared member	Warning
Recursive operator or property access	Warning
Duplicate or overlapping catch blocks	Warning

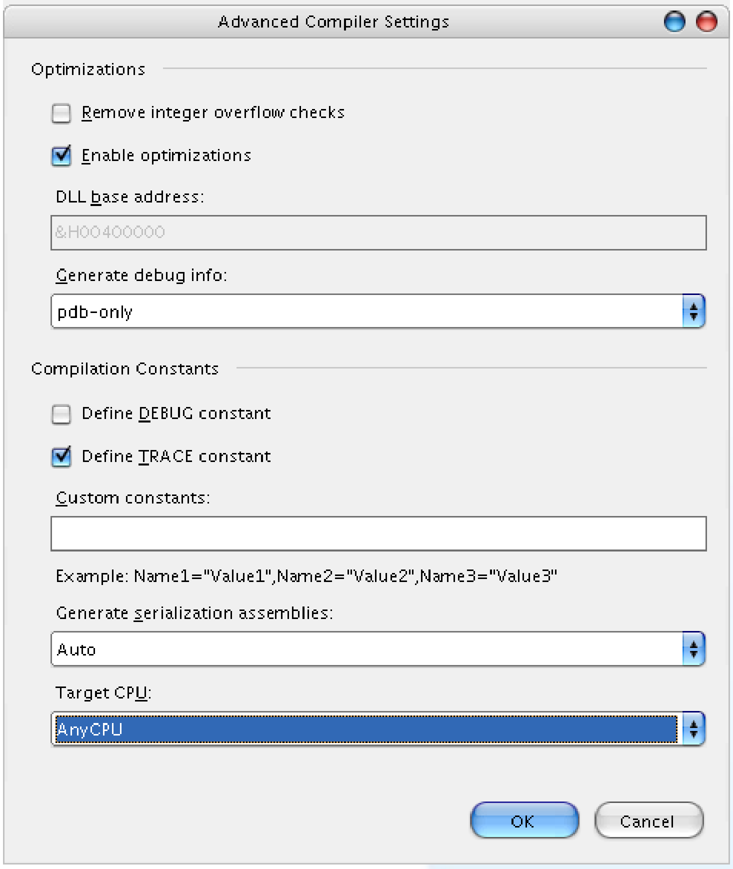
□ Disable all warnings

□ Treat all warnings as errors

☑ Generate XML documentation file

□ Register for COM interop

3. In the “Advanced Compiler Settings” dialog box, select the “Any CPU” option from the “Target CPU” section, and then click the OK button. For more details regarding the Target CPU options, refer to the important note below.



Advanced Compiler Settings

Optimizations

□ Remove integer overflow checks

☑ Enable optimizations

DLL base address: &H00400000

Generate debug info: pdb-only

Compilation Constants

□ Define DEBUG constant

☑ Define TRACE constant

Custom constants:

Example: Name1="Value1",Name2="Value2",Name3="Value3"

Generate serialization assemblies: Auto

Target CPU: AnyCPU

An important note regarding the Target CPU options:

Any CPU - The application will be compiled so that it will run natively on the CPU type is it currently running on, meaning that it will run as 64-bit on a 64-bit machine and 32-bit on a 32-bit machine. If you are compiling an executable file (.exe), it will run as an x64 process when loaded by an x64 version of the .Net Framework on an x64-based operating system. Otherwise the executable file will run as an x86 process.

x86 - The application will always run explicitly as an x86 process, regardless of the operating system or .Net Framework version.

x64 - The application will only load as an x64 process, regardless of the operating system or .Net Framework version. Attempting to run the an x64 application on a 32-bit Windows machine or attempting to call the application from a 32-bit process will result in a runtime error.

Testing the Application

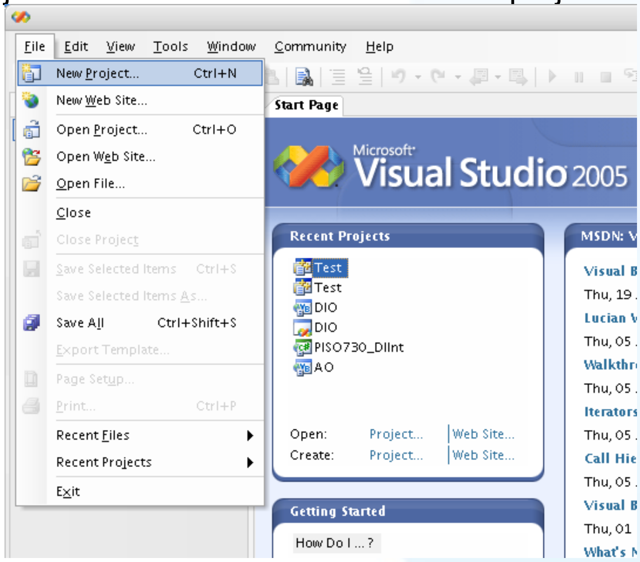
1. Run the application by either clicking the Start button on the toolbar, or by pressing F5.
2. Type "255" in the DO Value text box and then press the "Write" button to output a DO Value of 255.

3.8. Creating a Visual C#.NET Application

The following procedure describes how to create a Visual C#.NET application based on the UniDAQ DLL. Note that this description is based on Microsoft Visual Studio 2005.

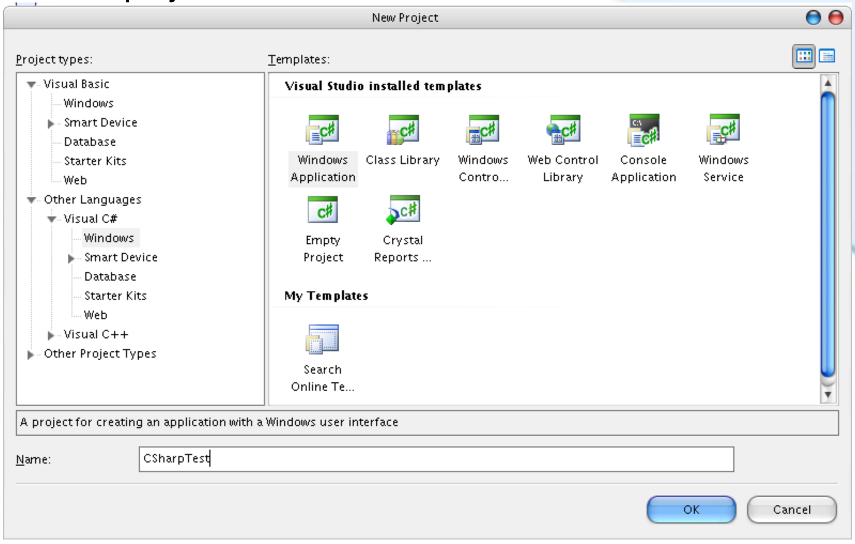
Creating the Application

1. Open Microsoft Visual Studio 2005, and click File from the main menu and then click New Project to create a new Visual C#.NET project.



2. Once the New Project dialog box is displayed, click the "Other Languages" item in the Project types pane, click "Visual C#", and then click the "Windows" option.

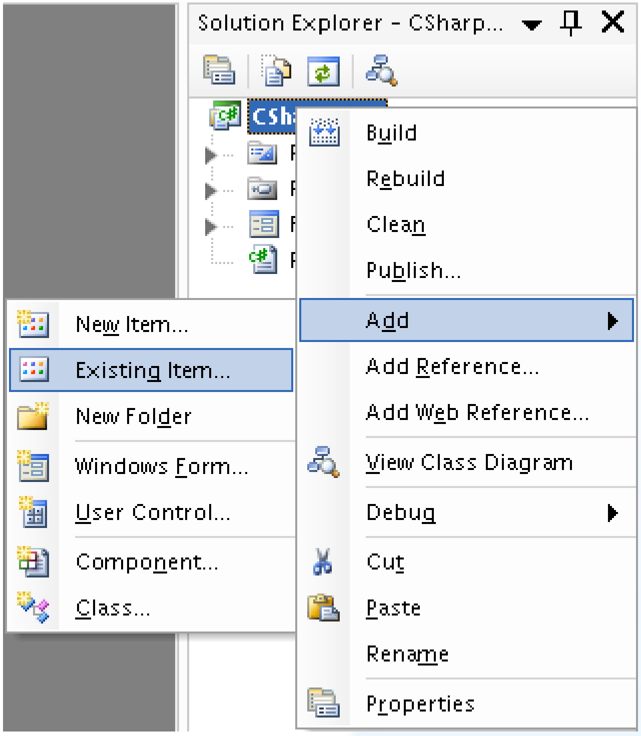
   In the Templates pane, click the Windows Application project template, enter “CSharpTest” in the Name field, and then click the OK button to create the new Visual C#.NET project.



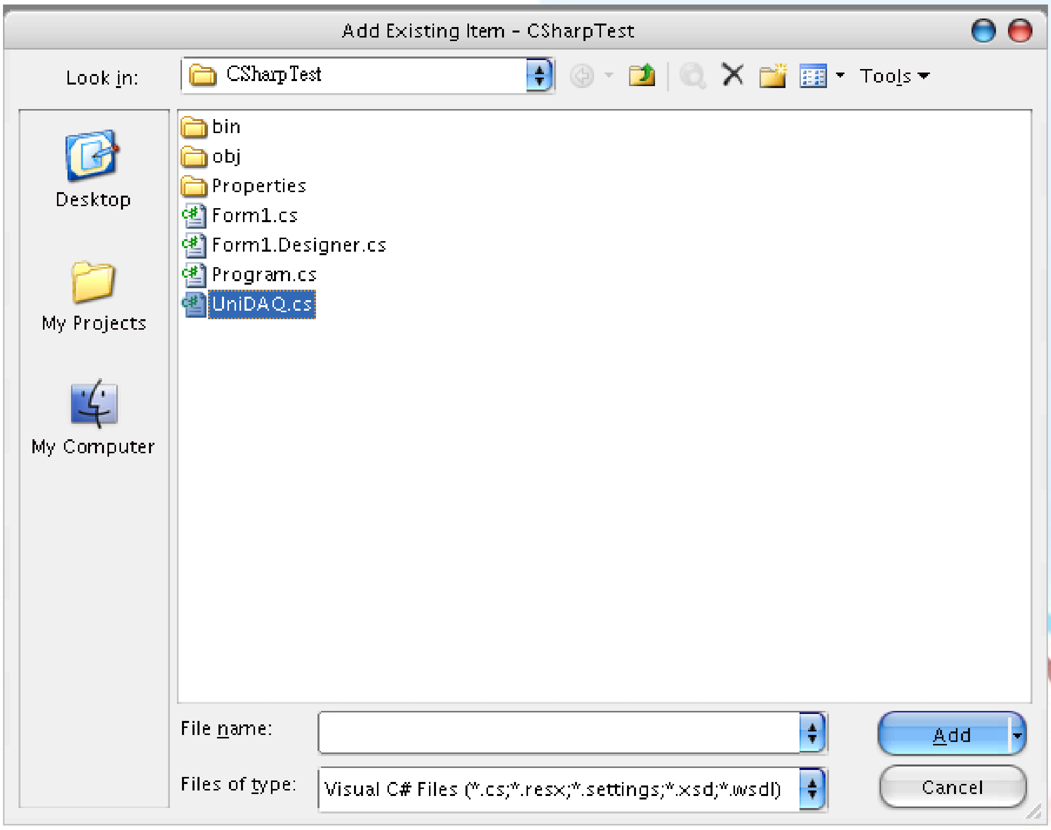
A project for creating an application with a Windows user interface

Name: CSharpTest

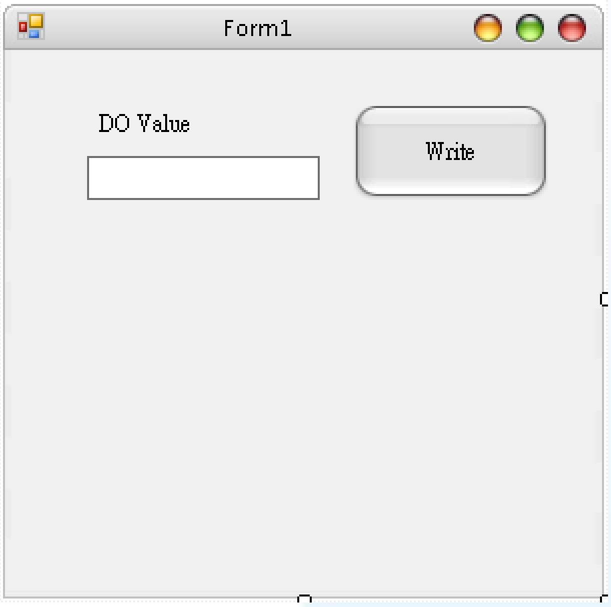
3. Once the project has been created, right-click the name of the newly created project in the Solutions Explorer pane, point to Add in the menu, and then click the Existing Item option to open the Add Existing Item dialog box for the CSharpTest project.



4. Add the UniDAQ.cs declaration file by clicking the name of the file and then clicking the Add button.



5. The Form design screen will then be automatically displayed allowing you to design the Form. From the Toolbox, select a Label control and position it on the form. Click on the new control to open the Properties window for the Label, and then enter “DO Value” in the Text field. Next, select a TextBox control from the Toolbox and position it on the Form. In the Properties window for the TextBox control, enter "txtDOVal". Finally, select a Button control from the Toolbox and position it on the Form. In the Properties window for the Button control, enter "btnWrite" in the Name field, and enter "Write" in the Text field. Your form should now look similar to the one shown in the image below:

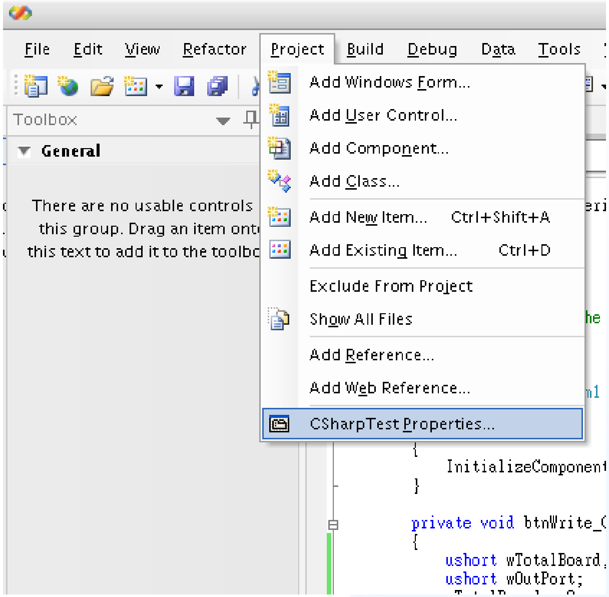


6. Double click the btnWrite control on the Form to open the code editing window and then add the following code to the Form.cs file:

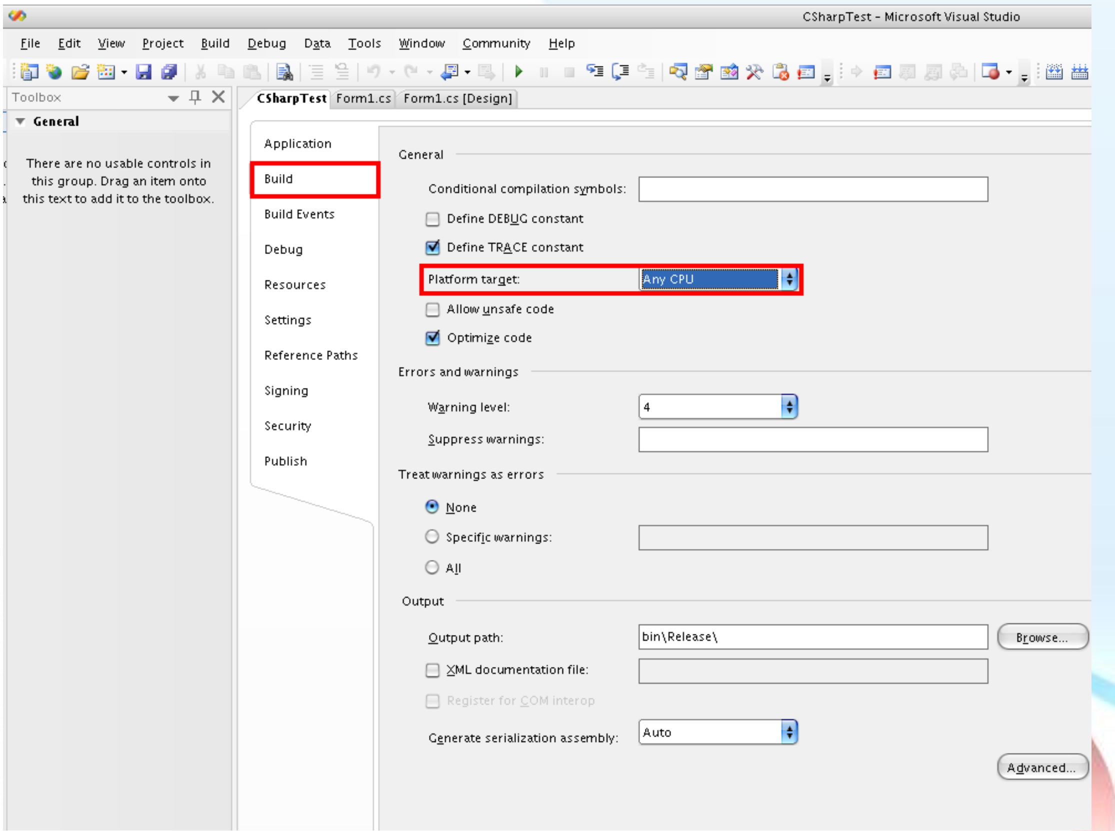
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using UniDAQ_Ns; //Include the UniDAQ namespace

namespace CSharpTest
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnWrite_Click(object sender, EventArgs e)
        {
            ushort wTotalBoard, wRtn, wBoardNo;
            ushort wOutPort;
            wTotalBoard = 0;
            //Initialize the resources and read the total number of boards form driver
            wRtn = UniDAQ.Ixud_DriverInit(ref wTotalBoard);
            if (wRtn != UniDAQ.Ixud_NoErr)
            {
                MessageBox.Show("Driver Initalization  Error.Error Code:" + 
wRtn.ToString());
                Close();
                return;
            }

            wBoardNo = 0;
            wOutPort = 0;
            //Write the DO Value
            wRtn = UniDAQ.Ixud_WriteDO(wBoardNo, wOutPort, 
Convert.ToUInt32(txtDOVal.Text));
            //Release the resources from the driver
            wRtn = UniDAQ.Ixud_DriverClose();

        }
    }
}
```

Compiling the Application

1. From the main menu, click Project, and then click "CSharpTest Properties” to display the Build options dialog box.



2. In the “General” section of the dialog box, select the “Any CPU” option from the “Platform target” dropdown menu. For more details regarding the Platform target options, refer to the important note below.



General

Conditional compilation symbols:

□ Define DEBUG constant

☑ Define TRACE constant

Platform target: Any CPU

□ Allow unsafe code

☑ Optimize code

Errors and warnings

Warning level: 4

Suppress warnings:

Treat warnings as errors

● None

○ Specific warnings:

○ All

Output

Output path: bin\Release\

□ XML documentation file:

□ Register for COM interop

Generate serialization assembly: Auto

⚠ An important note regarding the Platform target options:

Any CPU - The application will be compiled so that it will run natively on the CPU type is it currently running on, meaning that it will run as 64-bit on a 64-bit machine and 32-bit on a 32-bit machine. If you are compiling an executable file (.exe), it will run as an x64 process when loaded by an x64 version of the .Net Framework on an x64-based operating system. Otherwise the executable file will run as an x86 process.

x86 - The application will always run explicitly as an x86 process, regardless of the operating system or .Net Framework version.

x64 - The application will only load as an x64 process, regardless of the operating system or .Net Framework version. Attempting to run the an x64 application on a 32-bit Windows machine or attempting to call the application from a 32-bit process will result in a runtime error.

Testing the application

1. Run the application by either clicking the Start button on the toolbar, or by pressing F5.
2. Type "255" in the DO Value text box and then press the "Write" button to output a DO Value of 255.

3.9. Sample Programs and Related Documents

In addition to the UniDAQ Driver and DLL, ICP DAS provides a range of sample programs and source code that can be used in a Windows environment using a variety of programming languages, including Borland C++, Delphi, Visual Basic, Visual C, Visual Basic.NET, and Visual C#.NET.

The software, sample programs, and other related documentation can be accessed from the following locations:

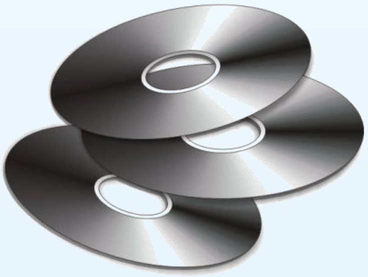
CD:\\ NAPDOS\PCI\UniDAQ\

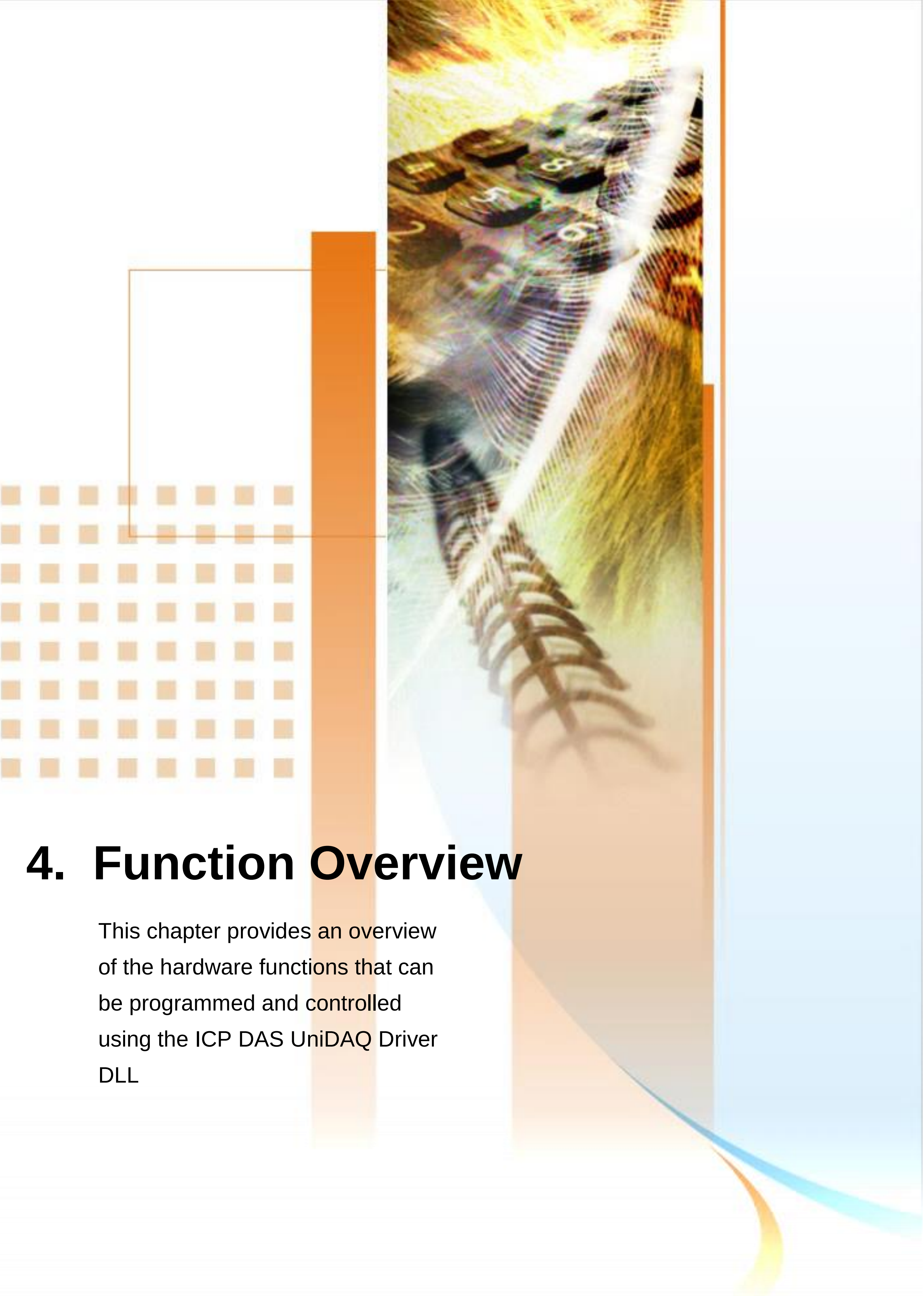
http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/unidaq/

ftp://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/unidaq/

The UniDAQ folder contains four sub-directories named DLL, Manual, LabView, and Matlab. An overview of the contents of each folder is given below.

- UniDAQ
  - DLL
    - The DLL folder contains the Driver installation files, the DLL, and the Header files and sample programs for a variety of programming languages.
  - Manual
    - The Manual folder contains the Driver DLL User Manual, i.e., this document.
  - LabView
    - The LabView folder contains sample programs that can be used in the LabView environment. Note that the UniDAQ Driver DLL must first be installed in order to use these sample programs.
  - Matlab
    - The Matlab folder contains sample programs that can be used in a MATLAB environment. Note that the UniDAQ Driver DLL must first be installed in order to use these sample programs.





4. Function Overview

This chapter provides an overview of the hardware functions that can be programmed and controlled using the ICP DAS UniDAQ Driver DLL

4.1. Introduction

ICP DAS UniDAQ Driver DLL contains a set of functions that can be used in a wide variety of applications for ICP DAS DAQ cards. The API functions support a range of development environments and programming languages, including Microsoft Visual C++, Microsoft Visual Basic, Borland Delphi, Borland C Builder++, Microsoft Visual C++.NET, Microsoft Visual C#.NET, and Microsoft Visual VB.NET.

Provides the following functions:

1. Driver Functionality: Initializes and releases device resources, and configures the device and accesses device information.

2. Digital I/O: Controls the Digital I/O functions for a specified channel.

3. Interrupt Event Functions: Provides support for DAQ cards that include interrupt functions, together with notifications that the Analog or Digital Input operations have been completed.

4. Analog Output: Provides the ability to convert DAC signals to output either voltage or current.

5. Analog Input: Provides the ability to convert single or multiple channels to acquire voltage, current, pressure, or strain data, etc.

6. Timer/Counter Functions: Provides the ability to perform event counting, frequency measurement and pulse output, etc.

7. Memory I/O: Provides the ability to control the memory I/O functions.

The UniDAQ Driver DLL supports the following programming languages:

- Microsoft Visual C++ version 4.0 or later
- Microsoft Visual Basic version 4.0 or later
- Borland Delphi version 2.0 or later
- Borland C++ Builder version 1.0 or later
- Microsoft Visual C++.NET version 2003 or later
- Microsoft Visual C#.NET version 2003 or later
- Microsoft Visual Basic.NET version 2003 or later

The following tables provide a summary of the function calls that can be accessed in custom applications using the UniDAQ Driver, each of which will be described in more detail later in this manual.

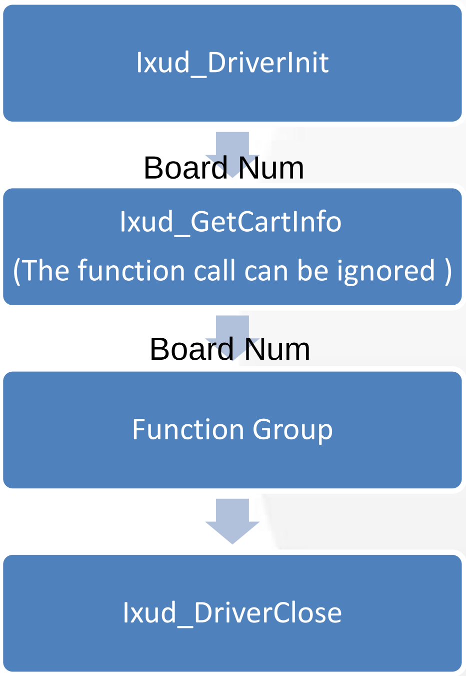
Driver Functions	Digital I/O	Interrupt Events	Analog Input
Ixud_GetDllVersion	Ixud_SetDIOModes32	Ixud_SetEventCallback	Ixud_ConfigAI
Ixud_OptionMode	Ixud_SetDIOMode	Ixud_RemoveEventCallback	Ixud_ConfigAIEx
Ixud_DriverInit	Ixud_ReadDI	Ixud_InstallIrq	Ixud_ClearAIBuffer
Ixud_DriverClose	Ixud_WriteDO	Ixud_RemoveIrq	Ixud_GetBufferStatus
Ixud_SearchCard	Ixud_ReadDIBit		Ixud_ReadAI
Ixud_GetBoardNoByCardID	Ixud_WriteDIBit		Ixud_ReadAIH
Ixud_GetCardInfo	Ixud_ReadDI32		Ixud_PollingAI
Ixud_ReadPort	Ixud_WriteDO32		Ixud_PollingAIH
Ixud_WritePort	Ixud_SoftwareReadbackDO		Ixud_PollingAIScan
Ixud_ReadPort32	Ixud_StartDI		Ixud_PollingAIScanH
Ixud_WritePort32	Ixud_StopDI		Ixud_StartAI
Ixud_ReadPhyMemory	Ixud_GetDIBufferH		Ixud_StartAIScan
Ixud_WritePhyMemory	Ixud_StartDO		Ixud_StartExtAI
	Ixud_StopDO		Ixud_StartExtAIScan
			Ixud_StartExtAnalogTrigger
			Ixud_GetAIBuffer
			Ixud_GetAIBufferH
			Ixud_StopAI

Analog Output	Timer/Counter	Memory I/O
Ixud_ConfigAO	Ixud_ReadCounter	Ixud_ReadMemory
Ixud_WriteAOVoltage	Ixud_SetCounter	Ixud_WriteMemory
Ixud_WriteAOVoltageH	Ixud_DisableCounter	Ixud_ReadMemory32
Ixud_WriteAOCurrent	Ixud_SetFCChannelMode	Ixud_WriteMemory32
Ixud_WriteAOCurrentH	Ixud_ReadFrequency	
Ixud_StartAOVoltage		
Ixud_StartAOVoltageH		
Ixud_StopAO		

4.2. Driver Functions

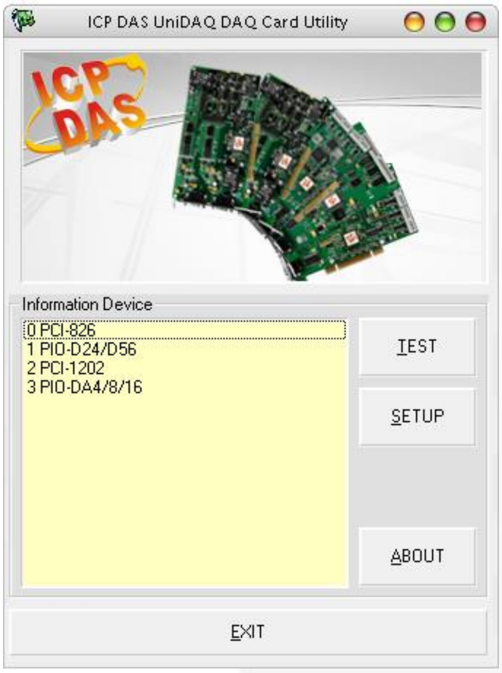
The figure below provides an overview of the common call flow for the ICP DAS UniDAQ Driver DLL

Call Flow



Board Num (Type: WORD, Size: 2 bytes)

The Board Num function specifies the DAQ board on which the I/O operations are to be performed. The value of Board Num depends on the Bus Num value and the Device number of the PCI Configuration space. The lower the Bus number and the Device number, the lower the Board Num value.



0 PCI-826
1 PIO-D24/D56
2 PCI-1202
3 PIO-DA4/8/16

The example shown in the image above indicates an entry for ”0 PCI-826”, which means that the Board Num value is equal to 0. This value can be used to directly assign the Board Num value to the function.

Ixud_DriverInit and Ixud_DriverClose

The Ixud_DriverInit function is used to allocate the resources for all boards installed in the system and to read the board number for each board. This function must be called when accessing the driver. The Ixud_DriverClose function is used to release the resources for board and must be called when ending access to the driver.

Ixud_GetCardInfo

This function is used to read the board name and hardware information. The function is optional and can be ignored if necessary.

4.3. Digital I/O

The Digital Input/Output function group is used to perform the Digital Input and Digital Output operations for the board. The Digital Input/Output lines on each data acquisition board are grouped into logical units called ports, and each port has 8, 16, or 32 lines or bits.

The Digital I/O ports for some data acquisition boards (e.g., the PIO-D24U/D56U/D48U/D96U/144U/168U) can be configured for either input or output. The Ixud_SetDIOModes32 or Ixud_SetDIOMode functions can be used to configure the specified port to be assigned for input or output.

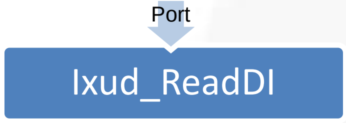
4.3.1. Digital Input

The Digital Input functions are used to perform Digital Input operations. The ICP DAS UniDAQ Driver DLL supports both Digital Input using software triggering, and Digital Input using interrupts.

Software triggering

The Ixud_ReadDI function can be used to read the status information from a port. The ICP DAS UniDAQ Driver DLL also includes the Ixud_ReadDIBit function that can be used to read a byte value from a specified bit.

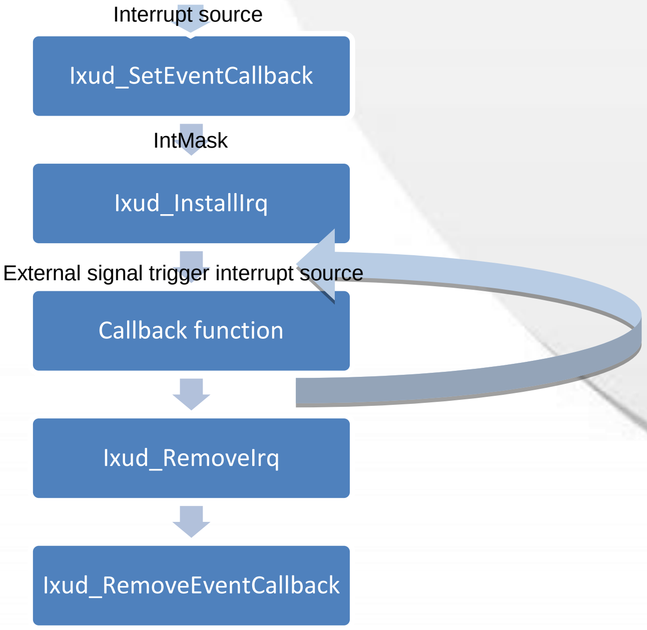
Call Flow



Interrupt Triggering

Interrupt Triggering allows the status of the Digital Input to be monitored. When the state changes from low to high or from high to low, the driver is acknowledged through a hardware interrupt, meaning that it is not necessary to periodically poll the Digital Input line.

Call Flow

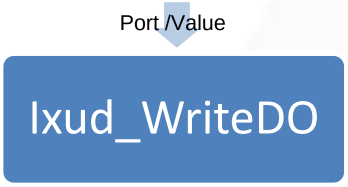


4.3.2. Digital Output

The digital output functions perform digital output operations.

User calls Ixud_WriteDO function to write a byte, word, dword value to a port. UniDAQ also provides Ixud_WriteDOBit function to set the state to the specified bit.

Call Flow



4.4. Analog Input

The analog input function group performs analog input functions. It can acquire single point data, multi-channels data, and waveform data with interrupt trigger. The analog input functions provide four kinds of operation according to the triggering mode and data transfer method.

Software Triggering

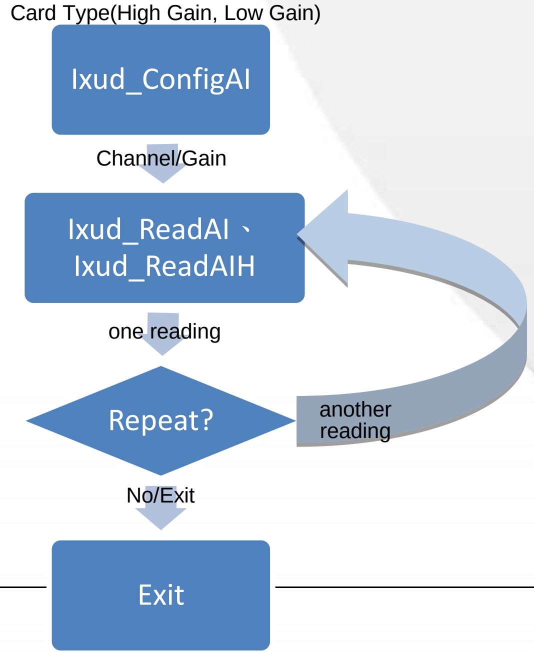
These functions trigger the data conversion by software. The UniDAQ provides three kinds of functions: one is for single point reading; the second is for multiple points reading; and the latest one is for multiple channel reading.

> ⚠ The sampling period of using software trigger on Windows platform is not as precise as using hardware trigger because of the effect from the multi-tasking system. It is recommended to use the software trigger function on low frequency measurement. (lower than 500 Hz)

Single Point Reading

If user wants to sample multiple data periodically by the functions, user can create a software timer to call the Ixud_ReadAI or Ixud_ReadAIH functions periodically.

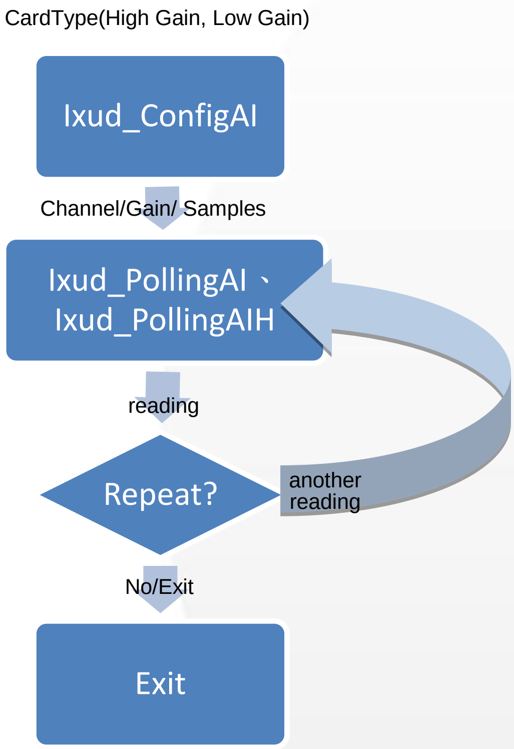
Call Flow



- **Multiple Points Reading**

The functions for single channel sampling are similar to that of multiple data reading.

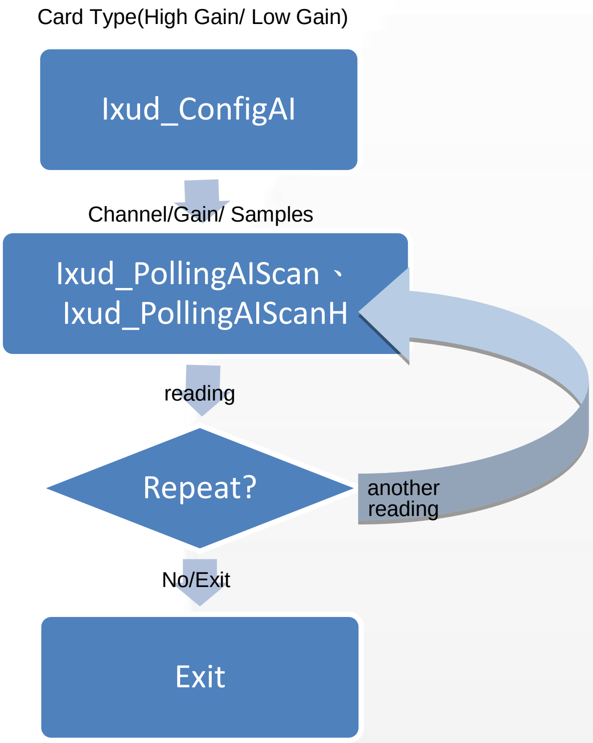
Call Flow



- **Multiple Channel Scan**

The functions for multiple channel sampling are similar to that data reading.

Call Flow



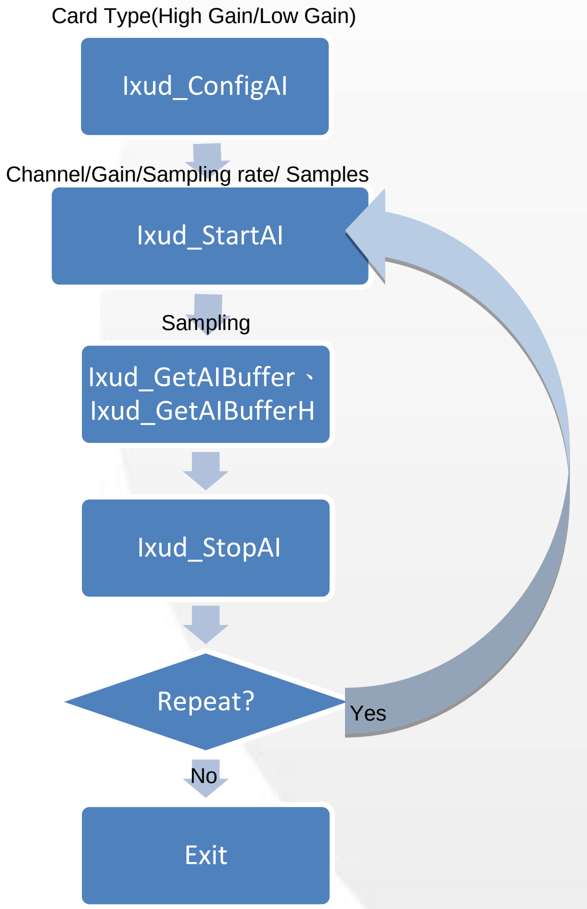
- **Waveform Data Reading**

The analog input function group provides many kinds of waveform data acquisition. The trigger mode is internal pacer trigger, interrupt trigger and external.

- **single-channel Internal Pacer trigger**

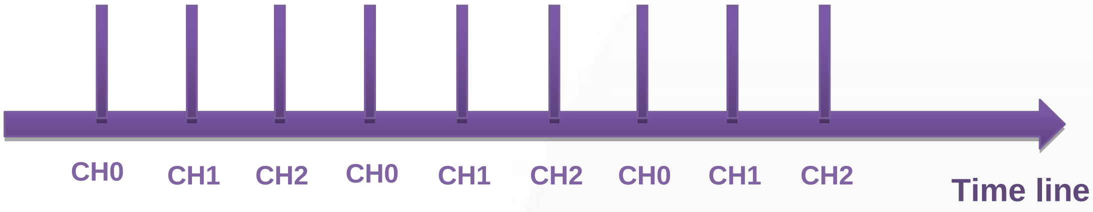
  Waveform data reading utilizes the on-board pacer to trigger the sampling operation and acknowledge the driver through a hardware interrupt or timer clock from single channel.

Call Flow

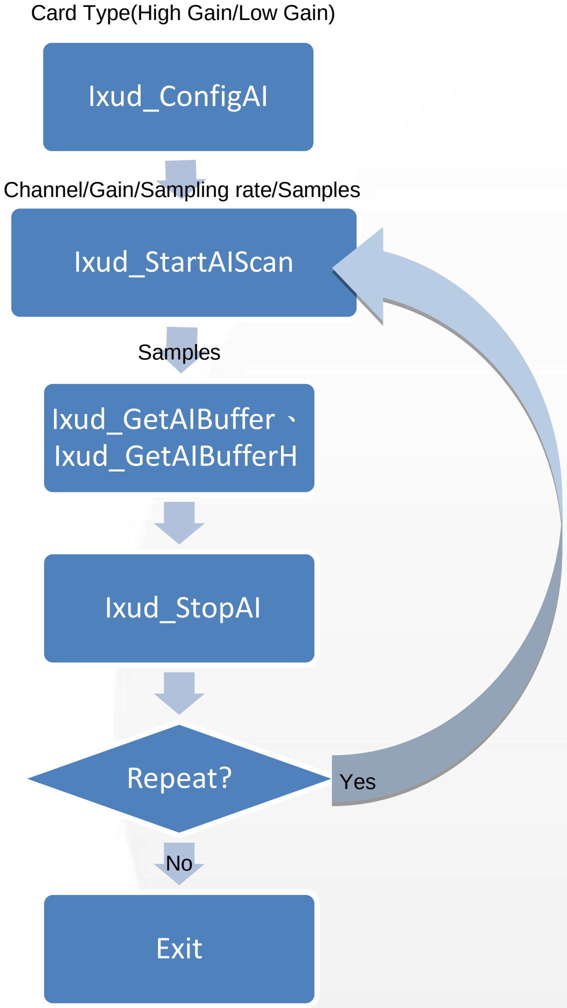


multi-channel Internal Pacer trigger

Waveform data reading utilizes the on-board pacer to trigger the sampling operation and acknowledge the driver through a hardware interrupt or timer clock from multi-channel.



Call Flow


Card Type(High Gain/Low Gain)
Ixud_ConfigAI
Channel/Gain/Sampling rate/Samples
Ixud_StartAIScan
Samples
Ixud_GetAIBuffer、Ixud_GetAIBufferH
Ixud_StopAI
Repeat?
Yes
No
Exit

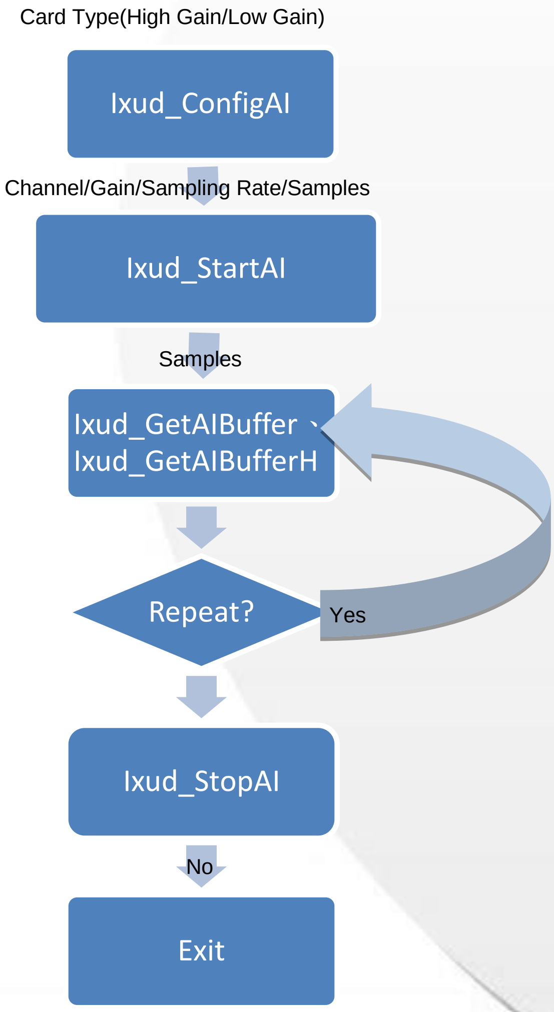
Long-term monitoring

The data buffer is configured as a big buffer (default is 2MB). The data acquisition will fill the buffer continuously. User can get data from this buffer infinite.

single channel continuous capture

Monitor the single form single channel to use continuous operation.

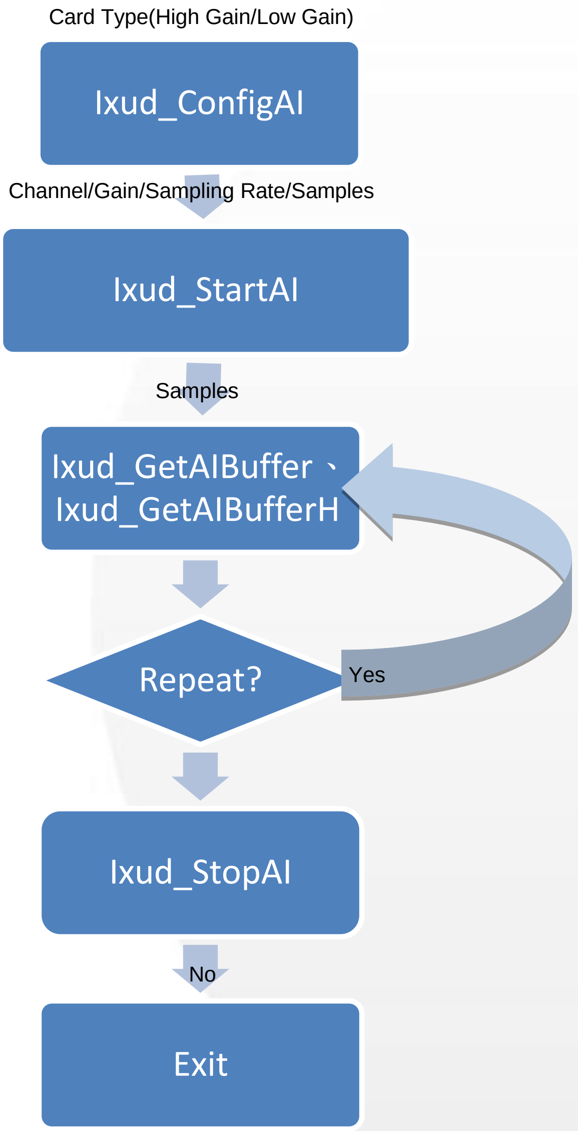
Call Flow


Card Type(High Gain/Low Gain)
Ixud_ConfigAI
Channel/Gain/Sampling Rate/Samples
Ixud_StartAI
Samples
Ixud_GetAIBuffer 、Ixud_GetAIBufferH
Repeat?
Yes
Ixud_StopAI
No
Exit

- multi-channel continuous capture

  Monitor the signal form multi-channel to use continuous operation.

Call Flow



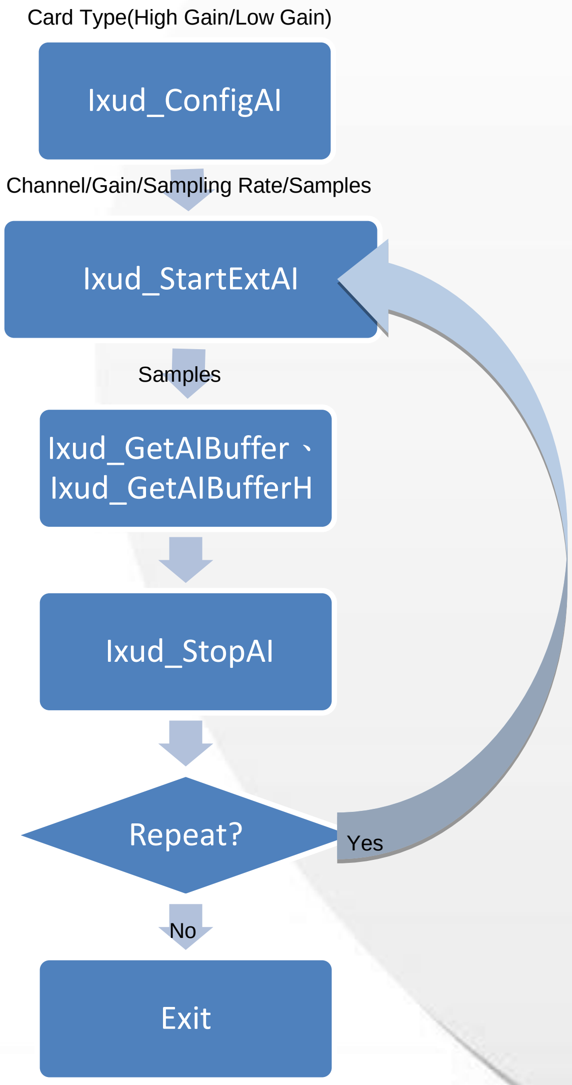
- External trigger operation

  The DAQ board may be triggered by TTL pulse received at the external pin. There are three kinds of external trigger operation for analog input. There are post-trigger, pre-trigger and mid-trigger.

  ■ single channel external trigger

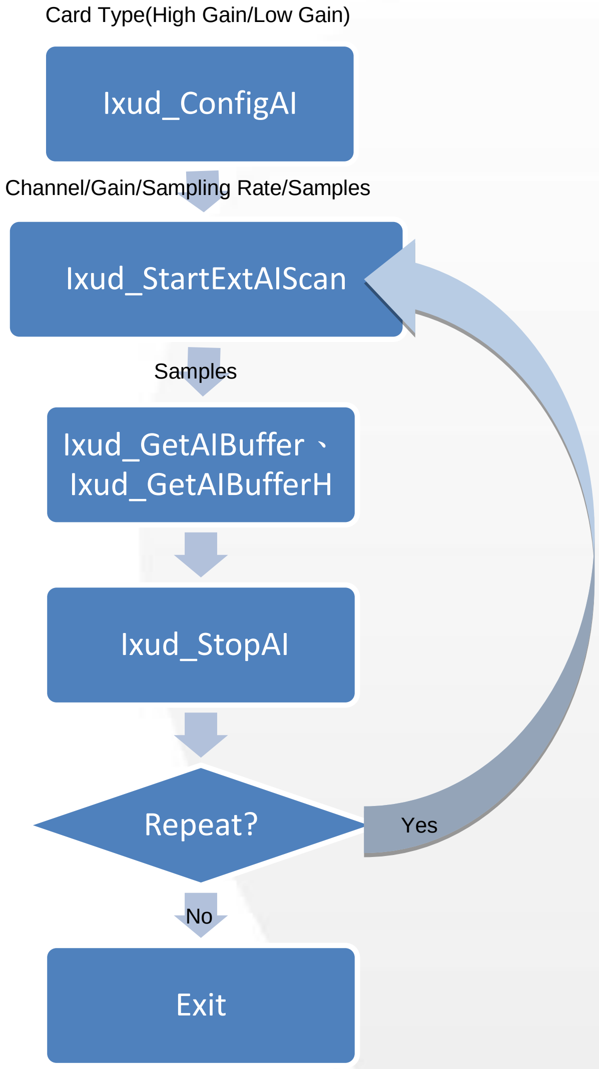
  Acquire the data by one channel on external trigger mode.

Call Flow


Card Type(High Gain/Low Gain)
Ixud_ConfigAI
Channel/Gain/Sampling Rate/Samples
Ixud_StartExtAI
Samples
Ixud_GetAIBuffer、Ixud_GetAIBufferH
Ixud_StopAI
Repeat?
Yes
No
Exit

- multiple-channel external trigger
  Acquire the data by multiple channels on external trigger mode.

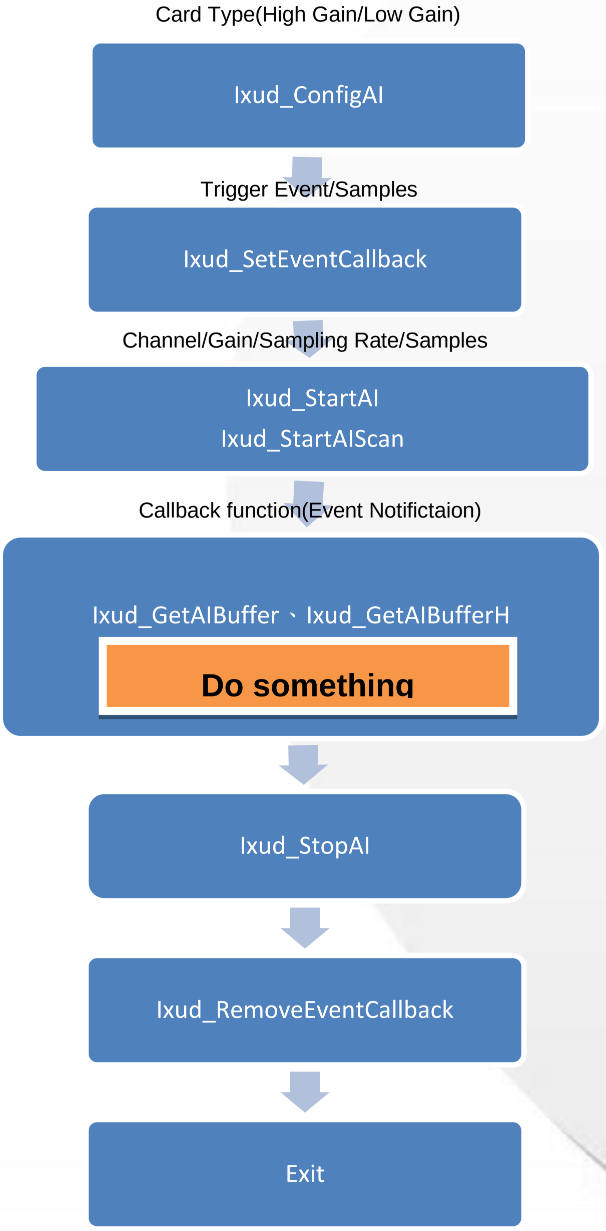
Call Flow


Card Type(High Gain/Low Gain)
Ixud_ConfigAI
Channel/Gain/Sampling Rate/Samples
Ixud_StartExtAIScan
Samples
Ixud_GetAIBuffer、Ixud_GetAIBufferH
Ixud_StopAI
Repeat?
Yes
No
Exit

- Analog input event trigger

  Some data acquisition operations run in the background, such as analog input with analog input with interrupt triggering. User can enable the event functions; the driver will trigger an event when data event occurs. User doesn't have to poll the status.

Call Flow

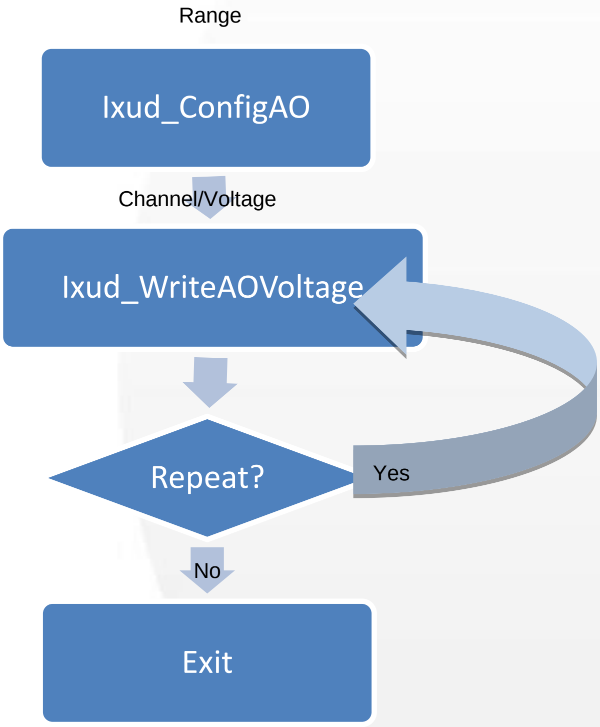

Card Type(High Gain/Low Gain)
Ixud_ConfigAI
Trigger Event/Samples
Ixud_SetEventCallback
Channel/Gain/Sampling Rate/Samples
Ixud_StartAI
Ixud_StartAIScan
Callback function(Event Notifictaion)
Ixud_GetAIBuffer、Ixud_GetAIBufferH
Do something
Ixud_StopAI
Ixud_RemoveEventCallback
Exit

4.5. Analog Output

The analog output function group performs analog output functions.

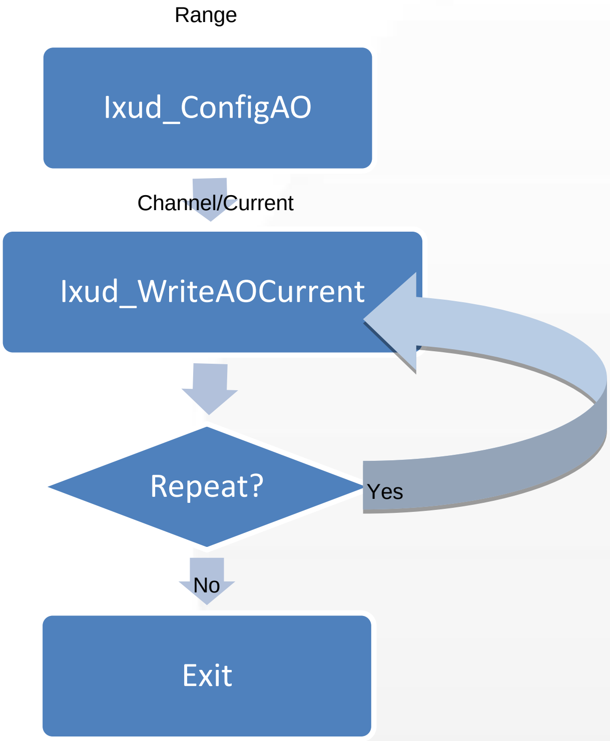
Static voltage output

Call Flow



Static current output

Call Flow

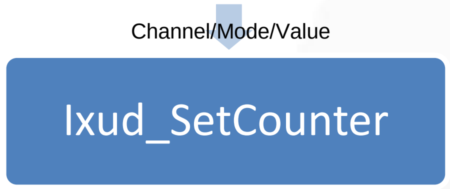


4.6. Timer/Counter

The timer/counter function group performs the counter operation.

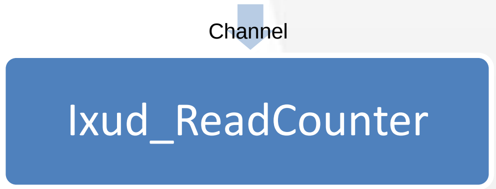
Write timer/counter

Call Flow



Read timer/counter

Call Flow

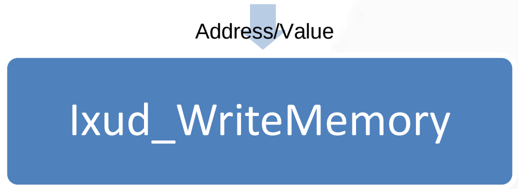


4.7. Memory R/W

The memory function group writes or reads by byte/word/dword data to a memory.

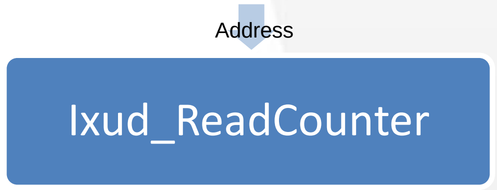
Writes memory

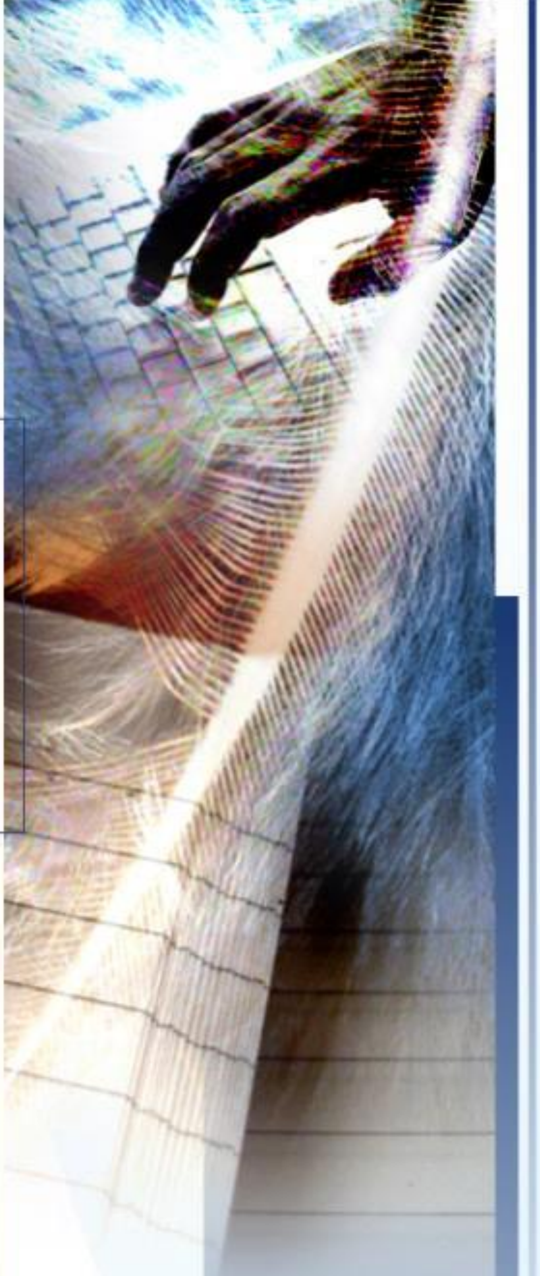
Call Flow



Reads memory

Call Flow





5. Function Reference

This chapter is a listing of all the functions and data structures that are supported by the ICP DAS UniDAQ Driver DLL. It shows what functions are supported by each ICP DAS's product.

5.1. Function Support List

Table 1

Function Name	Ixud_DriverInit Ixud_DriverClose	Ixud_SearchCard	Ixud_GetCardInfo	Ixud_GetBoardNoByCardID
PIO-D24/D24U/D56/D56U PEX-D24/D56	✔	✔	✔	✔
PIO-D48/D48U/D48SU PEX-D48	✔	✔	✔	✔
PIO-D64/D64U	✔	✔	✔	✔
PIO-D96/D96U/D96SU PEX-D96S	✔	✔	✔	✔
PIO-D144/D144U/D144LU PEX-D144LS	✔	✔	✔	✔
PIO-D168/D168U	✔	✔	✔	✔
PCI-D96SU/D128SU	✔	✔	✔	✔
PISO-DA2/DA2U	✔	✔	✔	✔
PIO-DA4/DA8/DA16 PIO- DA4U/DA8U/DA16U PISO-DA4U/DA8U/DA16U PEX-DA4/DA8/DA16	✔	✔	✔	✔
PISO-813/813U	✔	✔	✔	✔
PCI-P8R8/P8R8U	✔	✔	✔	✔
PCI-P16R16/P16R16U	✔	✔	✔	✔
PCI-P16C16/P16C16U	✔	✔	✔	✔
PCI-P16PRO16/P16POR16U PEX-P16POR16i/P8POR8i	✔	✔	✔	✔
PISO-P8R8UDC/AC	✔	✔	✔	✔
PISO-P8R8/P8R8U PEX-P8R8i	✔	✔	✔	✔
PISO-P16R16U PEX-P16R16i	✔	✔	✔	✔
PISO-P32C32/P32C32U/P32C32U-5V PEX-P32C32	✔	✔	✔	✔
PISO-P32A32/P32A32U/P32A32U-5V PEX-P32A32	✔	✔	✔	✔
PISO-P32S32WU	✔	✔	✔	✔

Function Name	Ixud_DriverInit Ixud_DriverClose	Ixud_SearchCard	Ixud_GetCardInfo	Ixud_GetBoardNoByCardID
PISO-P64/P64U PEX-P64	✔	✔	✔	✔
PISO-A64/A64U/C64/C64U PEX-C64	✔	✔	✔	✔
PISO-725/725U	✔	✔	✔	✔
PISO-730/730A/730AU PEX-730/730A	✔	✔	✔	✔
PISO-1730U	✔	✔	✔	✔
PCI-1002 series PEX-1002 series	✔	✔	✔	✔
PCI-1202 series PEX-1202 series	✔	✔	✔	✔
PCI-1602 series	✔	✔	✔	✔
PCI-1800/1802 series	✔	✔	✔	✔
PIO-821 Series	✔	✔	✔	✔
PCI-822LU/826LU	✔	✔	✔	✔
PCI-2602U	✔	✔	✔	✔
PCIe-8620	✔	✔	✔	✔
PCIe-8622	✔	✔	✔	✔
PCI-M512/M512U	✔	✔	✔	✔
PCI-FC16U	✔	✔	✔	✔
PCI-TMC12/TMC12A/TMC12AU PEX-TMC12A	✔	✔	✔	✔

Table 2

Function name	Ixud_ReadPort Ixud_ReadPort32	Ixud_WritePort Ixud_WritePort32	Ixud_SetDIOMode Ixud_SetDIOModes32
PIO-D24/D24U/D56/D56U PEX-D24/D56	✔	✔	✔
PIO-D48/D48U/D48SU PEX-D48	✔	✔	✔
PIO-D64/D64U	✔	✔	✔
PIO-D96/D96U/D96SU PEX-D96S	✔	✔	✔
PIO-D144/D144U/D144LU PEX-D144LS	✔	✔	✔
PIO-D168/D168U	✔	✔	✔
PCI-D96SU/D128SU	✔	✔	✔
PISO-DA2/DA2U	✔	✔	
PIO-DA4/DA8/DA16 PIO- DA4U/DA8U/DA16U PISO-DA4U/DA8U/DA16U PEX-DA4/DA8/DA16	✔	✔	
PISO-813/813U	✔	✔	
PCI-P8R8/P8R8U	✔	✔	
PCI-P16R16/P16R16U	✔	✔	
PCI-P16C16/P16C16U	✔	✔	
PCI-P16PRO16/P16POR16U PEX-P16POR16i/P8POR8i	✔	✔	
PISO-P8R8UDC/AC	✔	✔	
PISO-P8R8/P8R8U PEX-P8R8i	✔	✔	
PISO-P16R16U PEX-P16R16i	✔	✔	
PISO-P32C32/P32C32U/ P32C32U-5V PEX-P32C32	✔	✔	
PISO-P32A32/P32A32U/ P32A32U-5V PEX-P32A32	✔	✔	
PISO-P32S32WU	✔	✔	
PISO-P64/P64U PEX-P64	✔	✔	

Function name	Ixud_ReadPort Ixud_ReadPort32	Ixud_WritePort Ixud_WritePort32	Ixud_SetDIOMode Ixud_SetDIOModes32
PISO-A64/A64U/C64/C64U PEX-C64	✔	✔	
PISO-725/725U	✔	✔	
PISO-730/730A/730AU PEX-730/730A	✔	✔	
PISO-1730U	✔	✔	
PCI-1002 series PEX-1002 series	✔	✔	
PCI-1202 series PEX-1202 series	✔	✔	
PCI-1602 series	✔	✔	
PCI-1800/1802 series	✔	✔	
PIO-821 Series	✔	✔	
PCI-822LU/826LU	✔	✔	✔
PCI-2602U	✔	✔	✔
PCIe-8620	✔	✔	
PCIe-8622	✔	✔	
PCI-M512/M512U	✔	✔	
PCI-FC16U	✔	✔	✔
PCI-TMC12/TMC12A/TMC12AU PEX-TMC12A	✔	✔	

Table 3

Function Name	Ixud_ReadDI	Ixud_WriteDO	Ixud_ReadDIBit Ixud_ReadDI32	Ixud_WriteDOBit Ixud_WriteDO32
PIO-D24/D24U/D56/D56U PEX-D24/D56	✔	✔	✔	✔
PIO-D48/D48U/D48SU PEX-D48	✔	✔	✔	✔
PIO-D64/D64U	✔	✔	✔	✔
PIO-D96/D96U/D96SU PEX-D96S	✔	✔	✔	✔
PIO-D144/D144U/D144LU PEX-D144LS	✔	✔	✔	✔
PIO-D168/D168U	✔	✔	✔	✔
PCI-D96SU/D128SU	✔	✔	✔	✔
PISO-DA2/DA2U				
PIO-DA4/DA8/DA16 PIO- DA4U/DA8U/DA16U PISO-DA4U/DA8U/DA16U PEX-DA4/DA8/DA16	✔	✔	✔	✔
PISO-813/813U				
PCI-P8R8/P8R8U	✔	✔	✔	✔
PCI-P16R16/P16R16U	✔	✔	✔	✔
PCI-P16C16/P16C16U	✔	✔	✔	✔
PCI-P16PRO16/P16POR16U PEX-P16POR16i/P8POR8i	✔	✔	✔	✔
PISO-P8R8UDC/AC	✔	✔	✔	✔
PISO-P8R8/P8R8U PEX-P8R8i	✔	✔	✔	✔
PISO-P16R16U PEX-P16R16i	✔	✔	✔	✔
PISO-P32C32/P32C32U/P32C32U-5V PEX-P32C32	✔	✔	✔	✔
PISO-P32A32/P32A32U/ P32A32U-5V PEX-P32A32	✔	✔	✔	✔
PISO-P32S32WU	✔	✔	✔	✔
PISO-P64/P64U PEX-P64	✔		✔	

Function Name	Ixud_ReadDI	Ixud_WriteDO	Ixud_ReadDIBit Ixud_ReadDI32	Ixud_WriteDOBit Ixud_WriteDO32
PISO-A64/ A64U /C64/C64U PEX-C64		✔		✔
PISO-725/725U	✔	✔	✔	✔
PISO-730/730A/730AU PEX-730/730A	✔	✔	✔	✔
PISO-1730U	✔	✔	✔	✔
PCI-1002 series PEX-1002 series	✔	✔	✔	✔
PCI-1202 series PEX-1202 series	✔	✔	✔	✔
PCI-1602 series	✔	✔	✔	✔
PCI-1800/1802 series	✔	✔	✔	✔
PIO-821 Series	✔	✔	✔	✔
PCI-822LU/826LU	✔	✔	✔	✔
PCI-2602U	✔	✔	✔	✔
PCIe-8620	✔	✔	✔	✔
PCIe-8622	✔	✔	✔	✔
PCI-M512/M512U	✔	✔	✔	✔
PCI-FC16U	✔	✔	✔	✔
PCI-TMC12/TMC12A/TMC12AU PEX-TMC12A	✔	✔	✔	✔

Table 4

Function Name	Ixud_SoftwareReadbackDO	Ixud_SetEventCallback Ixud_RemoveEventCallback	Ixud_InstallIrq Ixud_RemoveIrq
PIO-D24/D24U/D56/D56U PEX-D24/D56	✔	✔	✔
PIO-D48/D48U/D48SU PEX-D48	✔	✔	✔
PIO-D64/D64U	✔	✔	✔
PIO-D96/D96U/D96SU PEX-D96S	✔	✔	✔
PIO-D144/D144U/D144LU PEX-D144LS	✔	✔	✔
PIO-D168/D168U	✔	✔	✔
PCI-D96SU/D128SU	✔	✔	✔
PISO-DA2/DA2U		✔	✔
PIO-DA4/DA8/DA16 PIO- DA4U/DA8U/DA16U PISO-DA4U/DA8U/DA16U PEX-DA4/DA8/DA16	✔	✔	✔
PISO-813/813U			
PCI-P8R8/P8R8U	✔		
PCI-P16R16/P16R16U	✔		
PCI-P16C16/P16C16U	✔		
PCI-P16PRO16/P16POR16U PEX-P16POR16i/P8POR8i	✔		
PISO-P8R8UDC/AC	✔		
PISO-P8R8/P8R8U PEX-P8R8i	✔		
PISO-P16R16U PEX-P16R16i	✔		
PISO-P32C32/P32C32U/P32C32U-5V PEX-P32C32	✔		
PISO-P32A32/P32A32U/ P32A32U-5V PEX-P32A32	✔		
PISO- P32S32WU	✔		
PISO-P64/P64U PEX-P64	✔		

Function Name	Ixud_SoftwareReadbackDO	Ixud_SetEventCallback Ixud_RemoveEventCallback	Ixud_InstallIrq Ixud_RemoveIrq
PISO-A64/A64U/C64/C64U PEX-C64	✔		
PISO-725/725U	✔	✔	✔
PISO-730/730A/730AU PEX-730/730A	✔	✔	✔
PISO-1730U	✔		
PCI-1002 series PEX-1002 series	✔	✔	
PCI-1202 series PEX-1202 series	✔		
PCI-1602 series	✔		
PCI-1800/1802 series	✔		
PIO-821 Series	✔	✔	
PCI-822LU/826LU	✔	✔	
PCI-2602U	✔		
PCIe-8620	✔	✔	
PCIe-8622	✔	✔	
PCI-M512/M512U	✔		
PCI-FC16U	✔		
PCI-TMC12/TMC12A/TMC12AU PEX-TMC12A	✔	✔	✔

Table 5

Function Name	Ixud_ConfigAI Ixud_ConfigAIEx	Ixud_ClearAIBuffer	Ixud_GetBufferStatus	Ixud_ReadAI Ixud_ReadAIH
PIO-D24/D24U/D56/D56U PEX-D24/D56				
PIO-D48/D48U/D48SU PEX-D48				
PIO-D64/D64U				
PIO-D96/D96U/D96SU PEX-D96S				
PIO-D144/D144U/D144LU PEX-D144LS				
PIO-D168/D168U				
PCI-D96SU/D128SU				
PISO-DA2/DA2U				
PIO-DA4/DA8/DA16 PIO- DA4U/DA8U/DA16U PISO-DA4U/DA8U/DA16U PEX-DA4/DA8/DA16				
PISO-813/813U	✔			✔
PCI-P8R8/P8R8U				
PCI-P16R16/P16R16U				
PCI-P16C16/P16C16U				
PCI-P16PRO16/P16POR16U PEX-P16POR16i/P8POR8i				
PISO-P8R8UDC/AC				
PISO-P8R8/P8R8U PEX-P8R8i				
PISO-P16R16U PEX-P16R16i				
PISO-P32C32/P32C32U/P32C32U-5V PEX-P32C32				
PISO-P32A32/P32A32U/P32A32U-5V PEX-P32A32				
PISO- P32S32WU				
PISO-P64/P64U PEX-P64				

Function Name	Ixud_ConfigAI Ixud_ConfigAIEx	Ixud_ClearAIBuffer	Ixud_GetBufferStatus	Ixud_ReadAI Ixud_ReadAIH
PISO-A64/A64U/C64/C64U PEX-C64				
PISO-725/725U				
PISO-730/730A/730AU PEX-730/730A				
PISO-1730U				
PCI-1002 series PEX-1002 series	✔	✔	✔	✔
PCI-1202 series PEX-1202 series	✔	✔	✔	✔
PCI-1602 series	✔	✔	✔	✔
PCI-1800/1802 series	✔	✔	✔	✔
PIO-821 Series	✔	✔	✔	✔
PCI-822LU/826LU	✔	✔	✔	✔
PCI-2602U	✔	✔	✔	✔
PCIe-8620	✔	✔	✔	✔
PCIe-8622	✔	✔	✔	✔
PCI-M512/M512U				
PCI-FC16U				
PCI-TMC12/TMC12A/TMC12AU PEX-TMC12A				

Table 6

Function Name	Ixud_PollingAI	Ixud_PollingAIH	Ixud_PollingAIScan	Ixud_PollingAIScanH
PIO-D24/D24U/D56/D56U PEX-D24/D56				
PIO-D48/D48U/D48SU PEX-D48				
PIO-D64/D64U				
PIO-D96/D96U/D96SU PEX-D96S				
PIO-D144/D144U/D144LU PEX-D144LS				
PIO-D168/D168U				
PCI-D96SU/D128SU				
PISO-DA2/DA2U				
PIO-DA4/DA8/DA16 PIO- DA4U/DA8U/DA16U PISO-DA4U/DA8U/DA16U PEX-DA4/DA8/DA16				
PISO-813/813U	✔	✔	✔	✔
PCI-P8R8/P8R8U				
PCI-P16R16/P16R16U				
PCI-P16C16/P16C16U				
PCI-P16PRO16/P16POR16U PEX-P16POR16i/P8POR8i				
PISO-P8R8UDC/AC				
PISO-P8R8/P8R8U PEX-P8R8i				
PISO-P16R16U PEX-P16R16i				
PISO-P32C32/P32C32U/P32C32U-5V PEX-P32C32				
PISO-P32A32/P32A32U/P32A32U-5V PEX-P32A32				
PISO- P32S32WU				
PISO-P64/P64U PEX-P64				

Function Name	Ixud_PollingAI	Ixud_PollingAIH	Ixud_PollingAIScan	Ixud_PollingAIScanH
PISO-A64/A64U/C64/C64U PEX-C64				
PISO-725/725U				
PISO-730/730A/730AU PEX-730/730A				
PISO-1730U				
PCI-1002 series PEX-1002 series	✔	✔	✔	✔
PCI-1202 series PEX-1202 series	✔	✔	✔	✔
PCI-1602 series	✔	✔	✔	✔
PCI-1800/1802 series	✔	✔	✔	✔
PIO-821 Series	✔	✔	✔	✔
PCI-822LU/826LU	✔	✔	✔	✔
PCI-2602U	✔	✔	✔	✔
PCIe-8620	✔	✔	✔	✔
PCIe-8622	✔	✔	✔	✔
PCI-M512/M512U				
PCI-FC16U				
PCI-TMC12/TMC12A/TMC12AU PEX-TMC12A				

Table 7

Function Name	Ixud_StartAI Ixud_StopAI	Ixud_StartAIScan	Ixud_StartExtAI Ixud_StartExtAIScan	Ixud_GetAIBuffer Ixud_GetAIBufferH
PIO-D24/D24U/D56/D56U PEX-D24/D56				
PIO-D48/D48U/D48SU PEX-D48				
PIO-D64/D64U				
PIO-D96/D96U/D96SU PEX-D96S				
PIO-D144/D144U/D144LU PEX-D144LS				
PIO-D168/D168U				
PCI-D96SU/D128SU				
PISO-DA2/DA2U				
PIO-DA4/DA8/DA16 PIO- DA4U/DA8U/DA16U PISO-DA4U/DA8U/DA16U PEX-DA4/DA8/DA16				
PISO-813/813U				
PCI-P8R8/P8R8U				
PCI-P16R16/P16R16U				
PCI-P16C16/P16C16U				
PCI-P16PRO16/P16POR16U PEX-P16POR16i/P8POR8i				
PISO-P8R8UDC/AC				
PISO-P8R8/P8R8U PEX-P8R8i				
PISO-P16R16U PEX-P16R16i				
PISO-P32C32/P32C32U/P32C32U-5V PEX-P32C32				
PISO-P32A32/P32A32U/P32A32U-5V PEX-P32A32				
PISO- P32S32WU				
PISO-P64/P64U PEX-P64				

Function Name	Ixud_StartAI Ixud_StopAI	Ixud_StartAIScan	Ixud_StartExtAI Ixud_StartExtAIScan	Ixud_GetAIBuffer Ixud_GetAIBufferH
PISO-A64/A64U/C64/C64U PEX-C64				
PISO-725/725U				
PISO-730/730A/730AU PEX-730/730A				
PISO-1730U				
PCI-1002 series PEX-1002 series	✔	✔		✔
PCI-1202 series PEX-1202 series	✔	✔		✔
PCI-1602 series	✔	✔		✔
PCI-1800/1802 series	✔	✔		✔
PIO-821 Series	✔			✔
PCI-822LU/826LU	✔	✔	✔	✔
PCI-2602U	✔	✔	✔	✔
PCIe-8620	✔	✔		✔
PCIe-8622	✔	✔	✔	✔
PCI-M512/M512U				
PCI-FC16U				
PCI-TMC12/TMC12A/TMC12AU PEX-TMC12A				

Table 8

Function Name	Ixud_ConfigAO	Ixud_WriteAOVoltage Ixud_WriteAOVoltageH	Ixud_WriteAOCurrent Ixud_WriteAOCurrentH
PIO-D24/D24U/D56/D56U PEX-D24/D56			
PIO-D48/D48U/D48SU PEX-D48			
PIO-D64/D64U			
PIO-D96/D96U/D96SU PEX-D96S			
PIO-D144/D144U/D144LU PEX-D144LS			
PIO-D168/D168U			
PCI-D96SU/D128SU			
PISO-DA2/DA2U	✔	✔	✔
PIO-DA4/DA8/DA16 PIO- DA4U/DA8U/DA16U PISO-DA4U/DA8U/DA16U PEX-DA4/DA8/DA16	✔	✔	✔
PISO-813/813U			
PCI-P8R8/P8R8U			
PCI-P16R16/P16R16U			
PCI-P16C16/P16C16U			
PCI-P16PRO16/P16POR16U PEX-P16POR16i/P8POR8i			
PISO-P8R8UDC/AC			
PISO-P8R8/P8R8U PEX-P8R8i			
PISO-P16R16U PEX-P16R16i			
PISO-P32C32/P32C32U/P32C32U-5V PEX-P32C32			
PISO-P32A32/P32A32U/P32A32U-5V PEX-P32A32			
PISO-P32S32WU			
PISO-P64/P64U PEX-P64			

Function Name	Ixud_ConfigAO	Ixud_WriteAOVoltage Ixud_WriteAOVoltageH	Ixud_WriteAOCurrent Ixud_WriteAOCurrentH
PISO-A64/A64U/C64/C64U PEX-C64			
PISO-725/725U			
PISO-730/730A/730AU PEX-730/730A			
PISO-1730U			
PCI-1002 series PEX-1002 series			
PCI-1202 series PEX-1202 series	✔	✔	✔
PCI-1602 series	✔	✔	✔
PCI-1800/1802 series	✔	✔	✔
PIO-821 Series	✔	✔	✔
PCI-822LU/826LU	✔	✔	✔
PCI-2602U	✔	✔	✔
PCIe-8620			
PCIe-8622	✔	✔	✔
PCI-M512/M512U			
PCI-FC16U			
PCI-TMC12/TMC12A/TMC12AU PEX-TMC12A			

Table 9

Function Name	Ixud_ReadCounter	Ixud_SetCounter	Ixud_DisableCounter
PIO-D24/D24U/D56/D56U PEX-D24/D56			
PIO-D48/D48U/D48SU PEX-D48	✔	✔	✔
PIO-D64/D64U	✔	✔	✔
PIO-D96/D96U/D96SU PEX-D96S			
PIO-D144/D144U/D144LU PEX-D144LS			
PIO-D168/D168U			
PCI-D96SU/D128SU			
PISO-DA2/DA2U	✔	✔	✔
PIO-DA4/DA8/DA16 PIO- DA4U/DA8U/DA16U PISO-DA4U/DA8U/DA16U PEX-DA4/DA8/DA16	✔	✔	✔
PISO-813/813U			
PCI-P8R8/P8R8U			
PCI-P16R16/P16R16U			
PCI-P16C16/P16C16U			
PCI-P16PRO16/P16POR16U PEX-P16POR16i/P8POR8i			
PISO-P8R8UDC/AC			
PISO-P8R8/P8R8U PEX-P8R8i			
PISO-P16R16U PEX-P16R16i			
PISO-P32C32/P32C32U/P32C32U-5V PEX-P32C32			
PISO-P32A32/P32A32U/P32A32U-5V PEX-P32A32			
PISO-P32S32WU			
PISO-P64/P64U PEX-P64			

Function Name	Ixud_ReadCounter	Ixud_SetCounter	Ixud_DisableCounter
PISO-A64/A64U/C64/C64U PEX-C64			
PISO-725/725U			
PISO-730/730A/730AU PEX-730/730A			
PISO-1730U			
PCI-1002 series PEX-1002 series			
PCI-1202 series PEX-1202 series			
PCI-1602 series			
PCI-1800/1802 series			
PIO-821 Series	✔	✔	✔
PCI-822LU/826LU			
PCI-2602U	✔	✔	✔
PCIe-8620			
PCIe-8622	✔	✔	✔
PCI-M512/M512U			
PCI-FC16U	✔		
PCI-TMC12/TMC12A/TMC12AU PEX-TMC12A	✔	✔	✔

Table 10

Function Name	PCI-M512U
Ixud_ReadMemory Ixud_WriteMemory	✔
Ixud_ReadMemory32 Ixud_WriteMemory32	✔

Table 11

Function Name	PCI-2602U	PCI-D96SU PCI-D128SU
Ixud_StartDO Ixud_StopDO	✔	✔
Ixud_StartDI Ixud_StopDI Ixud_GetDIBufferH	✔	
Ixud_StartExtAnalogTrigger	✔	
Ixud_StartAOVoltage Ixud_StartAOVoltageH Ixud_StopAO	✔	

5.2. Function Description

Please attend the following keyword before you reading this chapter.

Keyword	Set a value from Parameter	Returns a value in the Parameter
[Input]	Yes	No
[Output]	No	Yes

Every UniDAQ function is of the following form:

Status = FUNCTION_Name(Parameters 1, Parameters 2, …Parameters n)

Each function returns a value in the status variable that indicates the success or failure of the function as follows:

Status(Value)	Result
0	Function completed successfully
>0	Function failed due to error

Status is a 2-byte unsigned integer. For more information about the error code, please refer to A.1. Return Value

5.2.1. Driver Function Group

Ixud_GetDllVersion

Retrieves the version number of the DLL.

- **Syntax**

WORD Ixud_GetDllVersion(

DWORD ***dwDLLVer**

);

- **Parameters**

dwDLLVer

[Output] Retrieves the version number of the DLL.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_DriverInit

This function will request the system to allocate the resources, then search boards and initialize each board. Finally, it will retrieve the total number of boards. This function is the driver entry. It must be called before calling any function.

- **Syntax**

WORD Ixud_DriverInit(

WORD ***wTotalBoards**

);

- **Parameters**

  wTotalBoards

  [Output] Retrieves the total number of DAQ boards in the PC.

- **Return Value**

  Refer to Appendix A.1. Return Value.

Ixud_DriverClose

This function will release the resource to system. This function is the driver break. It must be called after calling any functions.

- **Syntax**

```
WORD Ixud_DriverClose(
        void
);
```

- **Parameters**

  None Parameters。

- **Return Value**

  Refer to Appendix A.1. Return Value.

Ixud_SearchCard

User calls this function to get the total board number for specific model. After this function is called, the board sequence will change for model.

- **Syntax**

```
WORD Ixud_SearchCard(
        WORD *wTotalBoards,
        DWORD dwModelNo
);
```

- **Parameters**

wTotalBoards

[Output] Retrieves the total board number for this board.

dwModelNo

[Input] Set the Model number, refer to Appendix A.2. 。

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_GetBoardNoByCardID

Use the parameters of Model number or Card ID to get the board number for this board.

- **Syntax**

```
WORD Ixud_GetBoardNoByCardID(
        WORD *wBoardNo,
        DWORD dwModelNumber,
        WORD wCardID
);
```

- **Parameters**

wBoardNo

[Output] Retrieves the board number.

dwModelNumber

[Input] Set the Model number, refer to Appendix A.2.

wCardID

[Input] Set the Card ID.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_GetCardInfo

Retrieves the hardware and software information and the model name of the board.

- **Syntax**

```
WORD Ixud_GetCardInfo(
        WORD wBoardNo,
        PIXUD_DEVICE_INFO sDevInfo,
        PIXUD_CARD_INFO sCardInfo,
        char *szModelName
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

sDevInfo

[Output] Retrieves the board information from the system. The data type is PIXUD_DEVICE_INFO.

sCardInfo

[Output] Retrieves the board hardware information. The data type is PIXUD_CARD_INFO.

szModelName[]

[Output] Retrieves the model name and is a string 20 char in length.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_ReadPort

Reads the byte/word/dword data from the specified I/O port.

- **Syntax**

```
WORD Ixud_ReadPort(
        DWORD dwAddress,
        WORD wSize,
        DWORD* dwVal
);
```

- **Parameters**

dwAddress

[Input] Sets the I/O port address.

wSize

[Input] Length of the data in bit.

wSize	length (bit)
8	8 (Byte)
16	16 (WORD)
32	32 (DWORD)

Table 5-1 wSize Parameters setting

dwVal

[Output] Retrieves the byte/word/dword data.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_WritePort

Writes the byte/word/dword date from the specified I/O port.

➢ Syntax

```
WORD Ixud_WritePort(
        DWORD dwAddress,
        WORD wSize,
        DWORD dwVal
);
```

➢ Parameters

dwAddress

[Input] Sets the I/O port address.

wSize

[Input] Length of the data in bit.

wSize	length (bit)
8	8 (Byte)
16	16 (WORD)
32	32 (DWORD)

Table 5-2 wSize Parameters setting

dwVal

[Input] Writes the byte/word/dword data.

➢ Return Value

Refer to Appendix A.1. Return Value.

Ixud_ReadPort32

Reads the dword data from the specified I/O port. User of Visual Basic 6.0 should use this function to read the dword data.

- **Syntax**

```
WORD Ixud_ReadPort32(
        DWORD dwAddress,
        DWORD* dwLow,
        DWORD* dwHigh
);
```

- **Parameters**

dwAddress

[Input] Sets the I/O port address.

dwLow

[Output] Retrieves the low part dword data.

dwHigh

[Output] Retrieves the high part dword data.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_WritePort32

Writes the dword data from the specified I/O port. User of Visual Basic 6.0 should use this function to read the dword data.

- **Syntax**

```
WORD Ixud_WritePort32(
        DWORD dwAddress,
        DWORD dwLow,
        DWORD dwHigh
);
```

- **Parameters**

dwAddress

[Input] Sets the I/O port address.

dwLow

[Input] Writes the low part data.

dwHigh

[Input] Writes the high part data.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_ReadPhyMemory

Reads the byte/word/dword data from the specified memory mapping I/O port.

➢ Syntax

```
WORD Ixud_ReadPhyMemory(
          DWORD dwAddress,
          WORD wSize,
          DWORD* dwValue
);
```

➢ Parameters

dwAddress

[Input] Sets the memory mapping I/O port address.

wSize

[Input] Length of the data in bit.

wSize	length (bit)
8	8 (Byte)
16	16 (WORD)
32	32 (DWORD)

Table 5-3 wSize Parameters setting

dwValue

[Output] Retrieves the byte/word/dword data.

➢ Return Value

Refer to Appendix A.1. Return Value.

Ixud_WritePhyMemory

Writes the byte/word/dword date from the specified memory mapping I/O port.

- **Syntax**

```
WORD Ixud_WritePhyMemory(
        DWORD dwAddress,
        WORD wSize,
        DWORD dwHigh
);
```

- **Parameters**

dwAddress

[Input] Sets the memory mapping I/O port address.

wSize

[Input] Length of the data in bit.

wSize	length (bit)
8	8 (Byte)
16	16 (WORD)
32	32 (DWORD)

Table 5-4 wSize Parameters setting

dwValue

[Input] Writes the byte/word/dword data.

- **Return Value**

Refer to Appendix A.1. Return Value.

5.2.2. Digital Input/Output Function Group

Ixud_SetDIOModes32

Sets the I/O mode for multiple ports. This function only supports the bi-direction I/O ports.

- **Syntax**

WORD Ixud_SetDIOModes32(

WORD wBoardNo**,**

DWORD dwDioModeMask

);

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

dwDioModeMask

[Input] Sets the bi-direction I/O port to input or output mode, each bit map to one port, it can set the 32 port at the same time. For detailed port mapping information, please refer to Appendix A.4. DI Port Number Definition and A.5. DO Port Number Definition。

Setting	I/O Mode
0	Input Mode
1	Output Mode

Table 5-5 I/O Mode Parameters Setting

- **Return Value**

Refer to Appendix A.1. Return Value.

- **Example**

```
wBoardNo = 0;//Sets the first board
dwDioModeMask = 5;//Sets the port0 and 2 is output mode, port 1 is input mode.
wRtn = Ixud_SetDIOModes32(wBoardNo, dwDioModeMask);
```

Ixud_SetDIOMode

Sets I/O mode for single port. This function only supports the bi-direction I/O ports.

➢ Syntax

```
WORD Ixud_SetDIOMode(
            WORD wBoardNo,
            WORD wPortNo,
            WORD wDioMode
);
```

➢ Parameters

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wPortNo

[Input] Sets port number

wDioMode

[Input] Sets bi-direction I/O port to input or output mode. For detailed port mapping information, please refer to Appendix A.4. DI Port Number Definition and A.5. DO Port Number Definition。

Setting	I/O Mode
0	Input Mode
1	Output Mode

Table 5-6 I/O Mode Parameters Setting

➢ Return Value

Refer to Appendix A.1. Return Value.

➢ Example

```
wRtn = Ixud_SetDIOMode(0, 1, 1);//Set Port 1 to Digital Output Mode
```

Ixud_ReadDI

Returns digital input data from the specified digital I/O port.

- **Syntax**

```
WORD Ixud_ReadDI(
            WORD wBoardNo,
            WORD wPortNo,
            DWORD *dwDIVal
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wPortNo

[Input] The user-assigned port number. For detailed port mapping information, please refer to Appendix A.4. DI Port Number Definition.

dwDIVal

[Output] 8/16/32-bit digital data read from the specified port.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_WriteDO

Writes the digital output data to specified digital I/O port.

- **Syntax**

```
WORD Ixud_WriteDO(
        WORD wBoardNo,
        WORD wPortNo,
        DWORD dwDOVal
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wPortNo

[Input] The user-assigned digital output port number. For detailed port mapping information, please refer to Appendix A.5. DO Port Number Definition.

dwDOVal

[Input] New digital logic state

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_ReadDIBit

Returns the bit state of digital input from the specified digital I/O port. If user must get more digital input channels status at the same time, please use the Ixud_ReadDI function that provides higher performance.

- **Syntax**

```
WORD Ixud_ReadDIBit(
          WORD wBoardNo,
          WORD wPortNo,
          WORD wBitNo,
          WORD *wDIVal
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wPortNo

[Input] The user-assigned port number. For detailed port mapping information, please refer to Appendix A.4. DI Port Number Definition.

wBitNo

[Input] The user-assigned channel number, where wBitNo =0 is the first channel and wBitNo=1 is the second channel, and so on.

wDIVal

[Output] bit data read from the specified port.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_WriteDOBit

Writes digital output bit data to the specified digital I/O port. If user must set more digital output channels status at the same time, please use Ixud_WriteDO function that provides the higher performance.

- **Syntax**

```
WORD Ixud_WriteDOBit(
            WORD wBoardNo,
            WORD wPortNo,
            WORD wBitNo,
            WORD wDOVal
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wPortNo

[Input] The user-assigned digital output port number. For detailed port mapping information, please refer to Appendix A.5. DO Port Number Definition.

wBitNo

[Input] The user-assigned channel number, where wBitNo =0 is the first channel and wBitNo=1 is the second channel, and so on.

wDOVal

[Input] Sets the digital output channel status.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_ReadDI32

Returns digital input 32-bit data from the specified digital I/O port. We suggest using this function when your programming language doesn't support unsigned Integer ex.Visual Basic 6.0.

➢ Syntax

```
WORD Ixud_ReadDI32(
        WORD wBoardNo,
        WORD wPortNo,
        DWORD* dwLow,
        DWORD* dwHigh,
);
```

➢ Parameters

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wPortNo

[Input] The user-assigned port number. For detailed port mapping information, please refer to Appendix A.4. DI Port Number Definition.

dwLow

[Output] Digital data of bit 0~15 read from the specified port.

dwHigh

[Output] Digital data of bit 16~31 read from the specified port.

➢ Return Value

Refer to Appendix A.1. Return Value.

Ixud_WriteDO32

Writes the digital output 32-bit data to specified digital I/O port. We suggest using this function when your programming language doesn’t support unsigned Integer ex.Visual Basic 6.0,

- **Syntax**

```
WORD Ixud_WriteDO32(
        WORD wBoardNo,
        WORD wPortNo,
        DWORD dwLow,
        DWORD dwHigh,
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wPortNo

[Input] The user-assigned digital output port number. For detailed port mapping information, please refer to Appendix A.5. DO Port Number Definition.

dwLow

[Input] New digital logic state for bit 0 ~ 15.

dwHigh

[Input] New digital logic state of bit 16 ~ 31.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_SoftwareReadbackDO

Returns the current digital output port status.(Non register-level).

- **Syntax**

WORD Ixud_SoftwareReadbackDO(

```
        WORD wBoardNo,
        WORD wPortNo,
        DWORD *dwDOVal
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wPortNo

[Input] The user-assigned digital output port number. For detailed port mapping information, please refer to Appendix A.5. DO Port Number Definition.

dwDOVal

[Output] Gets data from digital output port.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_StartDI

Initiates an asynchronous, the specified digital I/O port data acquisition operation with FIFO interrupt or without interrupt and stores its input in memory. It must call Ixud_GetDIBufferH function to get memory data, and call the Ixud_StopDI function to stop the acquisition operation.

⚠ Only support the PCI-2602U

⚠ When use this function to collect the data that will take up the CPU a short time. This time will depend on the amout of data and sampling rate.

- **Syntax**

WORD Ixud_StartDI(

```
        WORD wBoardNo,
        WORD wPortNo,
        DWORD dwReserved,
        float fSamplingRate,
        DWORD dwDataCount
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wPortNo

[Input] The user-assigned port number. For detailed port mapping information, please refer to Appendix A.4. DI Port Number Definition.

dwReserved

[Input] Reserved parameter.

fSamplingRate

[Input] Sampling rate in second. The fSamplingRate parameter specifies the rate for sampling one data in Hz. The driver uses it to program the on-board pacer.

dwDataCount

[Input] The sampled number. User must use the Ixud_StopDI to stop.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_StartDO

This function is used in PCI-2602U.It initiates the fast digital output operations by specifying the output count, the data buffer and the cyclic mode.

⚠ Only support the PCI-2602U / PCI-D96SU / PCI-D128SU

- **Syntax**

```
WORD Ixud_StartDO(
        WORD wBoardNo,
        WORD wPortNo,
        DWORD dwReserved,
        float fFrequency,
        DWORD dwDataCount,
        DWORD dwCycleNum,
        DWORD dwDOBuf[ ]
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wPortNo

[Input] The user-assigned digital output port number. For detailed port mapping information, please refer to Appendix A.5. DO Port Number Definition.

dwReserved

[Input] Reserved parameter.

fFrequency

[Input] Output frequency in second. The fFrequency parameter specifies the rate for output one data in Hz. The driver uses it to program the on-board pacer.

dwDataCount

[Input] The converted data count. The Max buffer size depends on the hardware property.

dwCycleNum

[Input] 0:Cyclic mode, the fast digital output operation will stop after user call Ixud_StopDO function.

dwDOBuf[]

[Input] The dwDOBuf[ ] to indicate the output data buffer. The load data is in time in order to avoid data under run.

➢ **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_GetDIBufferH

Gets the binary data for digital input data buffer. This function must be called after Ixud_StartDI function.

⚠ Only support the PCI-2602U

- **Syntax**

WORD Ixud_GetDIBufferH(

  WORD wBoardNo**,**

  WORD wPortNo,

  DWORD dwDataCount**,**

  DWORD hValue[ ]

);

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wPortNo

[Input] The user-assigned port number. For detailed port mapping information, please refer to Appendix A.4. DI Port Number Definition.

dwDataCount

[Input] The number of data from buffer

hValue[ ]

[Output] The measured raw data returned from buffer. Please declare the DWORD array, array size is dwDataCount

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_StopDI

Cancels the digital input data acquisition operation and reset the hardware and software.

⚠ Only support the PCI-2602U

- **Syntax**

```
WORD Ixud_StopDI(
        WORD wBoardNo,
        WORD wPortNo
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wPortNo

[Input] The user-assigned port number. For detailed port mapping information, please refer to Appendix A.4. DI Port Number Definition.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_StopDO

Cancels the digital output data acquisition operation and reset the hardware and software.

⚠ Only support the PCI-2602U / PCI-D96SU / PCI-D128SU

- **Syntax**

```
WORD Ixud_StopDO(
        WORD wBoardNo,
        WORD wPortNo
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wPortNo

[Input] The user-assigned digital output port number. For detailed port mapping information, please refer to Appendix A.5. DO Port Number Definition.

- **Return Value**

Refer to Appendix A.1. Return Value.

5.2.3. Interrupt Event Function Group

Ixud_SetEventCallback

Enable the callback function on interrupt event, when stop the callback function, it must call Ixud_RemoveEventCallback function to disable it.

- **Syntax**

```
WORD Ixud_SetEventCallback(
        WORD wBoardNo,
        WORD wEventType,
        WORD wInterruptSource,
        HANDLE *hEvent,
        PVOID CallbackFun,
        DWORD dwCallBackParameter
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wEventType

[Input] Sets notification event type, each bit can be enable one mode. About the detail setting, please refer to A.3.4. Interrupt Event Configuration Code

wInterruptSource

[Input] Sets interrupt source. About the detail interrupt source setting, please refer the following table:

wInterrupt Source	PIO-D24U PEX-D24 PIO-D56U PEX-D56	PIO-D48U PIO-D48SU PEX-D48	PIO-D64U	PIO-D96U PIO-D96SU PEX-D96S	PIO-D144U PIO-D144LU PIO-D168U PEX-D144LS	PCI-D96SU	PCI-D128SU
1	P2C0	P2C3/P2C7	EXTIRQ	P2C0	P2C0	Port 0	Port 0
2	P2C1	P5C3/P5C7	EVTIRQ	P5C0	P2C1	Port 1	Port 1
3	P2C2	COUT0	TMRIRQ	P8C0	P2C2	Port 2	Port 2
4	P2C3	COUT2	-	P11C0	P2C3		Port 3

wInterrupt Source	PIO-DA4U PIO-DA8U PIO-DA16U PISO-DA4U PISO-DA8U PISO-DA16U	PEX-DA4 PEX-DA8 PEX-DA16	PISO-730 PISO-730A PISO-730U PISO-730AU PEX-730 PEX-730A	PISO-725 PISO-725U	PCI-TMC12 PCI-TMC12A PCI-TMC12AU PEX-TMC12A	PIO-821 PCI-1002 PEX-1002	PCI-822LU PCI-826LU PCI-2602U PCIe-8620 PCIe-8622
1	COUT0	COUT0	DI0	IDI0	COUT3/6/9/12/Ext	AD Data	AD Data
2	COUT2	COUT2	DI1	IDI1	-	-	-
3	-		-	IDI2	-	-	-
4	-		-	IDI3	-	-	-
5	-		-	IDI4	-	-	-
6	-		-	IDI5	-	-	-
7	-		-	IDI6	-	-	-
8	-		-	IDI7	-	-	-

🟥 *Digital Input*

🟩 *Timer/Counter*

🟨 *Analog Input*

hEvent

[Input] Event pointer, please use Windows API CreateEvent(..) function create the event, when set to 0, system will create a event automatically.

CallbackFun

[Input] Sets Callback Function。

dwCallBackParameter

[Input] Sets the Parameters for Callback Function, when wEventType set to IXUD_APC_READY_INT, the parameters means the analog input data number.

- **Return Value**

Refer to Appendix A.1. Return Value.

- **Example**

DI Callback

```
//Set DI Callback function
// Use Source = 1 Event Type = Active High +Hardware Interrupt
wRtn = Ixud_SetEventCallback(wBoardNo, IXUD_HARDWARE_INT|IXUD_ACTIVE_HIGH , 1, hEvent0,
Callbackfun0, 0);

//Use Source = 3 Event Type = Active Low +Hardware Interrupt
wRtn = Ixud_SetEventCallback(wBoardNo, IXUD_HARDWARE_INT|IXUD_ACTIVE_LOW, 3, hEvent2,
Callbackfun2, 0);
```

AI Callback

```
// Set AI Callback function
// Use Source = 1 Event Type = APC Ready+Hardware Interrupt  Each AD data ready generate one Callback
Event
DataNum=1000;
wRtn = Ixud_SetEventCallback(wBoardNo, IXUD_HARDWARE_INT|IXUD_APC_READY_INT  , 1, hEvent0,
Callbackfun0,DataNum);
```

Ixud_RemoveEventCallback

Disable and remove the interrupt event and callback function. It must be called after calling Ixud_SetEventCallback function, before breaking callback function.

- **Syntax**

```
WORD Ixud_RemoveEventCallback(
          WORD wBoardNo,
          WORD wInterruptSource
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wInterruptSource

[Input] Sets interrupt source. About the detail interrupt source setting. When set to zero, it will remove all callback events.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_InstallIrq

Install the interrupt service routine, it supports to enable multiple interrupt source. Note: For Interrupt event of analog input, don't call this function.

- **Syntax**

```
WORD Ixud_InstallIrq(
          WORD wBoardNo,
          DWORD dwInterruptMask
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

dwInterruptMask

[Input] Interrupt source setting. Each bit enable one interrupt source, bit 0 is first interrupt source (INT_0), and so on.

Bit	PIO-D24U PEX-D24 PIO-D56U PEX-D56	PIO-D48U PIO-D48SU PEX-D48	PIO-D64U	PIO-D96U PIO-D96SU PEX-D96S	PIO-D144U PIO-D144LU PIO-D168U PEX-D144LS	PCI-D96SU	PCI-D128SU
0	P2C0	P2C3/P2C7	EXTIRQ	P2C0	P2C0	Port 0	Port 0
1	P2C1	P5C3/P5C7	EVTIRQ	P5C0	P2C1	Port 1	Port 1
2	P2C2	COUT0	TMRIRQ	P8C0	P2C2	Port 2	Port 2
3	P2C3	COUT2	-	P11C0	P2C3	-	Port 3

Bit	PIO-DA4U PIO-DA8U PIO-DA16U PISO-DA4U PISO-DA8U PISO-DA16U	PEX-DA4 PEX-DA8 PEX-DA16	PISO-730 PISO-730A PISO-730U PISO-730AU PEX-730 PEX-730A	PISO-725 PISO-725U	PCI-TMC12 PCI-TMC12A PCI-TMC12AU PEX-TMC12A
0	COUT0	COUT0	DI0	IDI0	COUT3/6/9/12/EXT
1	COUT2	COUT2	DI1	IDI1	-
2	-		-	IDI2	-
3	-		-	IDI3	-
4			-	IDI4	-
5			-	IDI5	-
6			-	IDI6	-
7			-	IDI7	-

🟥	*Digital Input*
🟩	*Timer/Counter*

- **Return Value**

Refer to Appendix A.1. Return Value.

- **Example**

```
dwInterruptMask = 0xF //(Enable INT_0,INT_1,INT_2 and INT_3)
wRtn=Ixud_InstallIrq(wBoardNo,dwInterruptMask);
```

Ixud_RemoveIrq

Disable the interrupt service routine.

- **Syntax**

WORD Ixud_RemoveIrq(

WORD wBoardNo

);

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

- **Return Value**

Refer to Appendix A.1. Return Value.

5.2.4. Analog Input Function Group

Ixud_ConfigAI

Configures the analog input settings for the specified analog input channel, it must be called before calling Analog Input Function Group.

- **Syntax**

```
WORD Ixud_ConfigAI(
        WORD wBoardNo,
        WORD wFIFOSizeKB,
        DWORD dwBufferSizeCount,
        WORD wCardType,
        WORD wDelaySettlingTime
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wFIFOSizeKB

[Input] Sets build-in FIFO size, the unit is Kbyte. When wFIFOSizeKB is 0, the driver will set the size automatically.

dwBufferSizeCount

[Input] Analog input buffer size in PC memory. The unit is DWORD. Default number is 524288 length (dwBufferSizeCount = 0), it spent about 2MB memory.

wCardType

[Input] analog input gain version type. Low gain version is 0, High gain version is 1.This setting will influence the accuracy and input range. The following table shows the detail setting:

wCardType	PISO-813	PIO-821	PCI-1002 PEX-1002	PCI-1202 PEX-1202	PCI-1602	PCI-1802 PCI-1800	PCI-822 PCI-826
0	JP1 = 20 V	PIO-821L	PCI-1002LU PEX-1002L	PCI-1202LU	PCI-1602U	PCI-1802LU PCI-1800LU	PCI-822LU PCI-826LU
1	JP1 = 10V	PIO-821H	PCI-1002HU PEX-1002H	PCI-1202HU	PCI-1602FU	PCI-1802HU PCI-1800HU	-

wCardType	PCI-2602U	PCIe-8620 PCIe-8622
0	PCI-2602U	PCIe-8620 PCIe-8622
1	-	

Table 5-7 wCardType Parameters setting

wDelaySettingTime

[Input] The analog input settling weight time, the unit is μs. This setting will influence the performance. We suggest setting 0(none delay weight time).

➢ **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_ConfigAIEx

Configures the analog input settings for the specified analog input channel and transfer mode, it must be called before calling Analog Input Function Group.

- **Syntax**

```
WORD Ixud_ConfigAIEx(
        WORD wBoardNo,
        WORD wFIFOSizeKB,
        DWORD dwBufferSizeCount,
        WORD wCardType,
        WORD wDelaySettlingTime,
        DWORD dwMode
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wFIFOSizeKB

[Input] Sets build-in FIFO size, the unit is Kbyte. When wFIFOSizeKB is 0, the driver will set the size automatically.

dwBufferSizeCount

[Input] Analog input buffer size in PC memory. The unit is DWORD. Default number is 524288 count(dwBufferSizeCount = 0), it spent about 2MB memory.

wCardType

[Input] analog input gain version type. Low gain version is 0, High gain version is 1.This setting will influence the accuracy and input range. The following table shows the detail setting:

wCardType	PISO-813	PIO-821	PCI-1002 PEX-1002	PCI-1202 PEX-1202	PCI-1602	PCI-1802 PCI-1800	PCI-822 PCI-826
0	JP1 = 20 V	PIO-821L	PCI-1002LU PEX-1002L	PCI-1202LU	PCI-1602U	PCI-1802LU PCI-1800LU	PCI-822LU PCI-826LU
1	JP1 = 10V	PIO-821H	PCI-1002HU PEX-1002H	PCI-1202HU	PCI-1602FU	PCI-1802HU PCI-1800HU	-

wCardType	PCI-2602U	PCIe-8620 PCIe-8622
0	PCI-2602U	PCIe-8620 PCIe-8622
1	-	

Table 5-8 wCardType Parameters setting

wDelaySettingTime

[Input] The analog input settling weight time, the unit is μs. This setting will influence the performance. We suggest setting 0(none delay weight time).

dwMode

[Input] The analog input data transfer mode.

dwMode	PCI-2602U PCIe-8620 PCIe-8622
ENABLEDMAAI	Use DMA Transfer

Table 5-9 dwMode Parameters setting

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_ClearAIBuffer

Clear the analog input buffer on system memory.

- **Syntax**

```
WORD Ixud_ClearAIBuffer(
        WORD wBoardNo
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_GetBufferStatus

Gets the status and data number from analog input buffer.

- **Syntax**

```
WORD Ixud_GetBufferStatus(
        WORD wBoardNo,
        WORD *wBufferStatus,
        DWORD *dwDataCount
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wBufferStatus

[Output] Gets analog input buffer status. The following table shows the description for value:

wBufferStatus	Status description
0	Empty, none data.
1	Normal, have data and no overflow
2	Buffer overflow
3	None allocate buffer
4	FIFO overflow
5	Unexpected, unknown status

Table 5-10 Analog input buffer status

dwDataCount

[Output] Get the analog input data number from buffer.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_ReadAI

Reads an analog input channel and returns one result scaled to a voltage (units = volts).

- **Syntax**

  WORD Ixud_ReadAI(

  WORD wBoardNo**,**

  WORD wChannel**,**

  WORD wConfig**,**

  float ***fValue**

  );

- **Parameters**

  wBoardNo

  [Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

  wChannel

  [Input] The sampled channel.

  wConfig

  [Input] Analog input range. Refer to A.3.1. AI Configuration Code. This setting will influence accuracy and input range.

  fValue

  [Output] float-point voltage reading from sampled channel. The unit is volts.

- **Return Value**

  Refer to Appendix A.1. Return Value.

Ixud_ReadAIH

Reads an analog input channel and returns one un-scaled result.

- **Syntax**

```
WORD Ixud_ReadAIH(
            WORD wBoardNo,
            WORD wChannel,
            WORD wConfig,
            DWORD *dwValue
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wChannel

[Input] The sampled channel.

wConfig

[Input] Analog input range. Refer to A.3.1. AI Configuration Code. This setting will influence accuracy and input range.

dwValue

[Output] raw data reading from sampled channel.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_PollingAI

Reads an analog input channel and returns the scaled to voltages (units=volts).

- **Syntax**

```
WORD Ixud_PollingAI(
        WORD wBoardNo,
        WORD wChannel,
        WORD wConfig,
        DWORD dwDataCount,
        float fValue[ ]
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wChannel

[Input] The sampled channel.

wConfig

[Input] Analog input range. Refer to A.3.1. AI Configuration Code. This setting will influence accuracy and input range.

dwDataCount

[Input] The number of the sampled data.

fValue[ ]

[Output] The measured voltages returned, scaled to units of volts. Please declare the float-point array, array size is dwDataCount.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_PollingAIH

Reads an analog input channel and returns the un-scaled results.

Syntax

```
WORD Ixud_PollingAIH(
            WORD wBoardNo,
            WORD wChannel,
            WORD wConfig,
            DWORD dwDataCount,
            DWORD dwValue[ ]
);
```

Parameters

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wChannel

[Input] The sampled channel.

wConfig

[Input] Analog input range. Refer to A.3.1. AI Configuration Code. This setting will influence accuracy and input range.

dwDataCount

[Input] The number of the sampled data.

dwValue[ ]

[Output] The measured raw data returned. Please declare the DWORD array, array size is dwDataCount.

Return Value

Refer to Appendix A.1. Return Value.

Ixud_PollingAIScan

Reads analog input channels and returns the scaled to voltages (units=volts).

- **Syntax**

```
WORD Ixud_PollingAIScan(
        WORD wBoardNo,
        WORD wChannels,
        WORD wChannelList[ ],
        WORD wConfigList[ ],
        DWORD dwDataCountPerChannel,
        float fValue[ ]
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wChannels

[Input] Number of channels.

wChannelList[ ]

[Input] Set the multiple of scan channels.

wConfigList[ ]

[Input] Analog input range array, set the analog input range for multiple of scan channels. Refer to A.3.1. AI Configuration Code.

dwDataCountPerChannel

[Input] The number of the sampled data for each channel.

fValue[ ]

[Output] The measured voltages returned, declare the float-point array, array size is wChannels multiply dwDataCountPerChannel. The sequence of array refers to Table 5-11.

- **Return Value**

Refer to Appendix A.1. Return Value.

- **Example**

```
DWORD dwDataCountPerChannel = 2 //Acquire two data from each sampled channel.
wChannels = 3                 //Number of channel is three.
float fValue[dwDataCounterPerChannel*wChannels]; //Declare the two multiply three array
wChannelList[0]= 5      //Acquire the channel 5 on first.
wChannelList[1]= 3      //Acquire the channel 3 on second
wChannelList[2]= 6      //Acquire the channel 6 on Third
wConfigList[0]= IXUD_BI_10V   //Input range of channel 5 is +/-10V
wConfigList[1]= IXUD_BI_5V    //Input range of channel 3 is +/-5V
wConfigList[2]= IXUD_BI_2V5   //Input range of channel 6 is +/-2.5V
wRtn = Ixud_PollingAIScan(wBoardNo, wChannels, wChannelList,wConfigList, dwDataCountPerChannel,
fValue)
```

Floating-point will storage to array(fValue[]), the sequence follows the table:

0	Channel 5	Value 0
1	Channel 3	Value 0
2	Channel 6	Value 0
3	Channel 5	Value 1
4	Channel 3	Value 1
5	Channel 6	Value 1

Table 5-11 Data sequence on array

Ixud_PollingAIScanH

Reads analog input channels and returns the un-scaled results.

- **Syntax**

WORD Ixud_PollingAIScanH(

WORD wBoardNo**,**

WORD wChannels**,**

WORD wChannelList[ ]**,**

WORD wConfigList[ ]**,**

DWORD dwDataCountPerChannel**,**

DWORD dwValue[ ]

);

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wChannels

[Input] Number of channels

wChannelList[ ]

[Input] Set the multiple of scan channels.

wConfigList[ ]

[Input] Analog input range array, set the analog input range for multiple of scan channels.

dwDataCountPerChannel

[Input] The number of the sampled data for each channel.

dwValue[ ]

[Output] The measured voltages returned, declare the dword array, array size is wChannels multiply dwDataCountPerChannel. The sequence of array refers to Table 5-12。

- **Return Value**

Refer to Appendix A.1. Return Value.

- **Example**

```
DWORD dwDataCountPerChannel = 2 //Acquire two data from each sampled channel.
wChannels = 3                   //Number of channel is three.
DWORD dwValue[dwDataCounterPerChannel*wChannels]; //Declare the two multiply three array
wChannelList[0]= 5      //Acquire the channel 5 on first.
wChannelList[1]= 3      //Acquire the channel 3 on second
wChannelList[2]= 6      //Acquire the channel 6 on Third
wConfigList[0]= IXUD_BI_10V   //Input range of channel 5 is +/-10V
wConfigList[1]= IXUD_BI_5V    //Input range of channel 3 is +/-5V
wConfigList[2]= IXUD_BI_2V5   //Input range of channel 6 is +/-2.5V
```

Floating-point will storage to array(dwValue[]), the sequence follows the table:

0	Channel 5	Val0
1	Channel 3	Val0
2	Channel 6	Val0
3	Channel 5	Val1
4	Channel 3	Val1
5	Channel 6	Val1

Table 5-12 Data sequence on array

Ixud_StartAI

Initiates an asynchronous, single-channel data acquisition operation with interrupt (support the ADC interrupt or FIFO interrupt) or without interrupt and stores its input in memory. It must call Ixud_GetAIBufferH or Ixud_GetAIBuffer function to get memory data, and call the Ixud_StopAI function to stop the acquisition operation.

⚠ When use this function to collect the data that will take up the CPU a short time. This time will depend on the amount of data and sampling rate.

- **Syntax**

```
WORD Ixud_StartAI(
        WORD wBoardNo,
        WORD wChannel,
        WORD wConfig,
        float fSamplingRate,
        DWORD dwDataCount
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wChannel

[Input] The sampled channel.

wConfig

[Input] Analog input range. Refer to A.3.1. AI Configuration Code. This setting will influence accuracy and input range.

fSamplingRate

[Input] Sampling rate in second. The fSamplingRate parameter specifies the rate for sampling one data in Hz. The driver uses it to program the on-board pacer.

dwDataCount

[Input] The sampled number. The dwDataCount =0 enable the continuous capture mode, User must use the Ixud_StopAI to stop.

⚠ Note of continuous capture mode:

1. When sampling rate is too fast, it is prone to develop FIFO overflow problem.
2. On continuous mode, analog input data will be stored in PC memory of user allocation. User must take the data on the suit time(Before the buffer overflow)
3. On data acquisition processing, user must reduce the CPU loading, ex. File processing etc.., otherwise, it will have the FIFO or buffer overflow.

➢ **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_StartAIScan

Initiates an asynchronous, multiple-channel data acquisition operation with interrupt (support the ADC interrupt or FIFO interrupt) or without interrupt and stores its input in memory and the gain codes for scan channel. It must call Ixud_GetAIBufferH or Ixud_GetAIBuffer function to get memory data, and call the Ixud_StopAI function to stop the acquisition operation.

⚠ When use this function to collect the data that will take up the CPU a short time. This time will depend on the amount of data and sampling rate.

- **Syntax**

```
WORD Ixud_StartAIScan(
        WORD wBoardNo,
        WORD wChannels,
        WORD wChannelList[ ],
        WORD wConfigList[ ],
        float fSamplingRate,
        DWORD dwDataCountPerChannel
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wChannels

[Input] Number of channels

wChannelList[ ]

[Input] Set the multiple of scan channels.

wConfigList[ ]

[Input] Analog input range array, set the analog input range for multiple of scan channels.

fSamplingRate

[Input] Sampling rate in second. The fSamplingRate parameter specifies the rate for sampling one data in Hz. The driver uses it to program the on-board pacer.

dwDataCountPerChannel

[Input] The number of the sampled data for each channel. The dwDataCountPerChannel =0 enable the continuous capture mode. User must use the Ixud_StopAI to stop.

⚠ Note of continuous capture mode:

1. When sampling rate is too fast, it is prone to develop FIFO overflow problem.
2. On continuous mode, analog input data will be stored in PC memory of user allocation. User must take the data on the suit time(Before the buffer overflow)
3. On data acquisition processing, user must reduce the CPU loading, ex. File processing etc.., otherwise, it will have the FIFO or buffer overflow.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_StartExtAI

Initiates an asynchronous, single-channel data acquisition operation with external signal trigger(TTL Level) and stores its input in memory. It must call Ixud_GetAIBufferH or Ixud_GetAIBuffer function to get memory data, and call the Ixud_StopAI function to stop the acquisition operation.

⚠ When use this function to collect the data that will take up the CPU a short time. This time will depend on the amount of data and sampling rate.

- **Syntax**

```
WORD Ixud_StartExtAI(
        WORD wBoardNo,
        WORD wActive,
        WORD wChannel,
        WORD wConfig,
        float fSamplingRate,
        DWORD dwPostDataCount
        DWORD dwPreDataCount
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wActive

[Input] It sets a specified trigger type.

dwActive	PCI-822LU PCI-826LU	PCI-2602U PCIe-8622
0	Failing Edge	Any Edge
1	Raising Edge	Any Edge

wChannel

[Input] The sampled channel.

wConfig

[Input] Analog input range. Refer to A.3.1. AI Configuration Code. This setting will influence accuracy and input range.

fSamplingRate

[Input] Sampling rate in second. The fSamplingRate parameter specifies the rate for sampling one data in Hz. The driver uses it to program the on-board pacer.

dwPostDataCount

[Input] The number of sampled data after the external trigger signal.

dwPreDataCount

[Input] The number of sampled data before the external trigger signal.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_StartExtAnalogTrigger

Initiates an asynchronous, single-channel data acquisition operation with external signal trigger(analog signal) and stores its input in memory. It must call Ixud_GetAIBufferH or Ixud_GetAIBuffer function to get memory data, and call the Ixud_StopAI function to stop the acquisition operation.

⚠ Only support the PCI-2602U

⚠ When use this function to collect the data that will take up the CPU a short time. This time will depend on the amount of data and sampling rate.

- **Syntax**

```
WORD Ixud_StartExtAnalogTrigger(
            WORD wBoardNo,
            WORD wActive,
            WORD wChannel,
            WORD wConfig,
            float fSamplingRate,
            DWORD dwDataCount,
            DWORD dwReserved,
            float fAboveTrgVoltage,
            float fBelowTrgVoltage
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wActive

[Input] It sets a specified analog trigger type.

dwActive	PCI-2602U
IXUD_ANALOGTRIGGER_ABOVE	Above High
IXUD_ANALOGTRIGGER_BELOW	Below Low
IXUD_ANALOGTRIGGER_LEAVE	Leave Region
IXUD_ANALOGTRIGGER_ENTRY	Entry Region

wChannel

[Input] The sampled channel.

wConfig

[Input] Analog input range. Refer to A.3.1. AI Configuration Code. This setting will influence accuracy and input range.

fSamplingRate

[Input] Sampling rate in second. The fSamplingRate parameter specifies the rate for sampling one data in Hz. The driver uses it to program the on-board pacer.

dwDataCount

[Input] The number of sampled data after the external trigger signal

dwReserved

[Input] Reserved parameter.

fAboveTrgVoltage

[Input] Above trigger voltage range.

fBelowTrgVoltage

[Input] Below trigger voltage range.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_StartExtAIScan

Initiates an asynchronous, multiple-channel data acquisition operation with external signal trigger(TTL level) and stores its input in memory and the gain codes for scan channel. It must call Ixud_GetAIBufferH or Ixud_GetAIBuffer function to get memory data, and call the Ixud_StopAI function to stop the acquisition operation.

> ⚠ When use this function to collect the data that will take up the CPU a short time. This time will depend on the amount of data and sampling rate.

- **Syntax**

```
WORD Ixud_StartExtAIScan(
          WORD wBoardNo,
          WORD wChannels,
          WORD wActive,
          WORD wChannelList[ ],
          WORD wConfigList[ ],
          float fSamplingRate,
          DWORD dwPostDataCountPerChannel,
          DWORD dwPreDataCountPerChannel
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wChannels

[Input] Number of channels.

wActive

[Input] It sets a specified trigger type.

dwActive	PCI-822LU PCI-826LU	PCI-2602U PCIe-8622
0	Failing Edge	Any Edge
1	Raising Edge	Any Edge

wChannelList[ ]

[Input] Set the multiple of scan channels.

wConfigList[ ]

[Input] Analog input range array, set the analog input range for multiple of scan channels.

fSamplingRate

[Input] Sampling rate in second. The fSamplingRate parameter specifies the rate for sampling one data in Hz. The driver uses it to program the on-board pacer.

dwPostDataCountPerChannel

[Input] The number of sampled data after the external trigger signal

dwPreDataCountPerChannel

[Input] The number of sampled data before the external trigger signal

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_GetAIBuffer

Gets the floating-point voltage value for analog data buffer. This function must be called after Ixud_StartAI, Ixud_StartAIScan, Ixud_StartExtAI or Ixud_StartExtAIScan function.

- **Syntax**

```
WORD Ixud_GetAIBuffer(
        WORD wBoardNo,
        DWORD dwDataCount,
        float fValue[ ]
        );
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

dwDataCount

[Input] The number of data from buffer

fValue[ ]

[Output] The measured voltages returned from buffer, scaled to units of volts. Please declare the float-point array, array size is dwDataCount.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_GetAIBufferH

Gets the binary data for analog data buffer. This function must be called after Ixud_StartAI, Ixud_StartAIScan, Ixud_StartExtAI or Ixud_StartExtAIScan function.

- **Syntax**

```
WORD Ixud_GetAIBufferH(
        WORD wBoardNo,
        DWORD dwDataCount,
        DWORD hValue[ ]
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

dwDataCount

[Input] The number of data from buffer

hValue[ ]

[Output] The measured raw data returned from buffer. Please declare the DWORD array, array size is dwDataCount

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_StopAI

Cancels the current data acquisition operation and reset the hardware and software.

- **Syntax**

```
WORD Ixud_StopAI(
        WORD wBoardNo
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

- **Return Value**

Refer to Appendix A.1. Return Value.

5.2.5. Analog Output Function Group

Ixud_ConfigAO

Records the output range for each analog output channel, it must be called before calling analog output function group.

- **Syntax**

```
WORD Ixud_ConfigAO(
        WORD wBoardNo,
        WORD wChannel,
        WORD wCfgCode
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wChannel

[Input] The output number

wCfgCode

[Input] Sets output range and polarity selected. Refer to A.3.2. AO Configuration Code(Voltage) and A.3.3. AO Configuration Code (Current). The setting will influence accuracy and input range.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_WriteAOVoltage

Accepts a floating-point voltage value, scales it to the proper binary number, and writes the number to an analog output channel to change the output voltage.

- **Syntax**

WORD Ixud_WriteAOVoltage(

WORD wBoardNo**,**

WORD wChannel**,**

float fValue

);

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wChannel

[Input] The output number

fValue

[Input] Floating-point value to be written, the unit is volts.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_WriteAOVoltageH

Writes a binary value to one of the analog output channels, changing the voltage produced at the channel.

- **Syntax**

```
WORD Ixud_WriteAOVoltageH(
        WORD wBoardNo,
        WORD wChannel,
        DWORD hValue
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wChannel

[Input] The output number

hValue

[Input] Binary data to be written

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_WriteAOCurrent

Accepts a floating-point current value, scales it to the proper binary number, and writes the number to an analog output channel to change the output current.

- **Syntax**

WORD Ixud_WriteAOCurrent(

```
        WORD wBoardNo,
        WORD wChannel,
        float fValue
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wChannel

[Input] The output number

fValue

[Input] Floating-point value to be written, the unit is mA.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_WriteAOCurrentH

Writes a binary value to one of the analog output channels, changing the voltage produced at the channel.

- **Syntax**

```
WORD Ixud_WriteAOCurrentH(
        WORD wBoardNo,
        WORD wChannel,
        DWORD hValue
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wChannel

[Input] The output number

hValue

[Input] Binary data to be written.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_StartAOVoltage

This function is used in PCI-2602U.It initiates the fast analog output operations by specifying the output count, the data (floating-point voltage value) buffer and the cyclic mode.

⚠ Only support the PCI-2602U

- **Syntax**

```
WORD Ixud_StartAOVoltage(
            WORD wBoardNo,
            WORD wChannel,
            WORD wCfgCode,
            float fFrequency,
            DWORD dwDataCount,
            DWORD dwCycleNum,
            float fAOBuf[ ]
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wChannel

[Input] The output number.

wCfgCode

[Input] Sets output range and polarity selected. Refer to A.3.2. AO Configuration Code(Voltage) and A.3.3. AO Configuration Code (Current). The setting will influence accuracy and input range.

fFrequency

[Input] Output frequency in second. The fFrequency parameter specifies the rate for output one data in Hz. The driver uses it to program the on-board pacer.

dwDataCount

[Input] The converted data count. The Max buffer size depends on the hardware property.

dwCycleNum

[Input] 0:Cyclic mode, the fast digital output operation will stop after user call Ixud_StopAO function.

fAOBuf[]

[Input] The fAOBuf[ ] to indicate the analog data buffer for floating-point voltage. The load data is in time in order to avoid data under run.

➢ **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_StartAOVoltageH

This function is used in PCI-2602U.It initiates the fast analog output operations by specifying the output count, the data (binary value) buffer and the cyclic mode.

⚠ Only support the PCI-2602U

➢ **Syntax**

```
WORD Ixud_StartAOVoltageH(
        WORD wBoardNo,
        WORD wChannel,
        WORD wCfgCode,
        float fFrequency,
        DWORD dwDataCount,
        DWORD dwCycleNum,
        DWORD dwAOBuf[ ]
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wChannel

[Input] The output number.

wCfgCode

[Input] Sets output range and polarity selected. Refer to A.3.2. AO Configuration Code(Voltage) and A.3.3. AO Configuration Code (Current). The setting will influence accuracy and input range.

fFrequency

[Input] Output frequency in second. The fFrequency parameter specifies the rate for output one data in Hz. The driver uses it to program the on-board pacer.

dwDataCount

[Input] The converted data count. The Max buffer size depends on the hardware property.

dwCycleNum

[Input] 0:Cyclic mode, the fast digital output operation will stop after user call Ixud_StopAO function.

dwAOBuf[]

[Input] The dwAOBuf[ ] to indicate the output data buffer. The load data is in time in order to avoid data under run.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_StopAO

Cancels the analog output data acquisition operation and reset the hardware and software.

⚠ Only support the PCI-2602U

- **Syntax**

WORD Ixud_StopAO(

    WORD wBoardNo,

    WORD wChannel

);

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wChannel

[Input] The output number.

- **Return Value**

Refer to Appendix A.1. Return Value.

5.2.6. Timer/Counter Function Group

Ixud_DisableCounter

Turns off the specified counter operation.

- **Syntax**

```
WORD Ixud_DisableCounter(
        WORD wBoardNo,
        WORD wChannel
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wChannel

[Input] Counter number, where wChannel=0 is first channel, and wChannel=1 is the second channel, and so on.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_ReadCounter

Reads the current counter total without disturbing the counting process and returns the count and overflow conditions.

- **Syntax**

```
WORD Ixud_ReadCounter(
        WORD wBoardNo,
        WORD wChannel,
        DWORD *dwValue
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wChannel

[Input] Counter number, where wChannel=0 is first channel, and wChannel=1 is the second channel, and so on.

dwValue

[Output] Counter value returned

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_ReadFrequency

Reads the frequency measurement.(Only support the PCI-FC16U)。

⚠ Only support the PCI-FC16U

- **Syntax**

```
WORD Ixud_ReadFrequency(
        WORD wBoardNo,
        WORD wChannel,
        float *fFrequency,
        DWORD dwTimeOutMs,
        WORD *wStatus
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wChannel

[Input] Counter number, where wChannel=0 is first channel, and wChannel=1 is the second channel, and so on.

fFrequency

[Output] Counter frequency returned, the units is Hz.

dwTimeOutMs

[Input] The delay time of counter, the units is ms.

wStatus

[Output] Counter status returned

wStatus	Description
0	Waiting the counter frequency
1	Timeout
2	Latch the frequency

Table 5-13 wStatus Parameters setting

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_SetCounter

Configures the specified counter for pulse output and starts the counter.

- **Syntax**

```
WORD Ixud_SetCounter(
        WORD wBoardNo,
        WORD wChannel,
        WORD wMode,
        DWORD dwValue
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wChannel

[Input] Counter number, where wChannel=0 is first channel, and wChannel=1 is the second channel, and so on.

wMode

[Input] Counter mode. The detail information, refer to Intel 8254 Datasheet.

wMode	Mode Definitions
0	Interrupt on terminal count
1	Hardware retriggerable one-shot
2	Rate generator
3	Square wave mode
4	Software triggered strobe
5	Hardware triggered strobe(Retriggerable)

Table 5-14 wMode Parameters Setting

dwValue

[Input] User input value for counter setting

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_SetFCChannelMode

Configures the counting mode for the specified counter

⚠ Only support the PCI-FC16U

- **Syntax**

```
WORD Ixud_SetFCChannelMode(
        WORD wBoardNo,
        WORD wChannel,
        WORD wMode,
        WORD wDelayMs
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

wChannel

[Input] Counter number, where wChannel=0 is first channel, and wChannel=1 is the second channel, and so on.

wMode

[Input] Counter mode

wMode	Description
0	-
1	-
2	down count mode
3	-
4	-
5	-

Table 5-15 wMode Parameters Setting

wDelayMs

[Input] Counter delay time. The unit is ms.

- **Return Value**

Refer to Appendix A.1. Return Value.

5.2.7. Memory Input/Output Function Group

Ixud_ReadMemory

Returns data from the specified memory.

- **Syntax**

```
WORD Ixud_ReadMemory(
        WORD wBoardNo,
        DWORD dwOffsetByte,
        WORD wSize,
        DWORD *dwValue
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

dwOffsetByte

[Input] Address offset

wSize

[Input] Data length

wSize	length
8	8-bit
16	16-bit
32	32-bit

Table 5-16 wSize Parameters Setting

dwValue

[Output] 8/16/32-bit digital data read from the specified memory.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_WriteMemory

Writes data to specified memory.

- **Syntax**

```
WORD Ixud_WriteMemory(
        WORD wBoardNo,
        DWORD dwOffsetByte,
        WORD wSize,
        DWORD dwValue
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

dwOffsetByte

[Input] Address offset

wSize

[Input] Data length

wSize	Length
8	8-bit
16	16-bit
32	32-bit

Table 5-17 wSize Parameters Setting

dwValue

[Input] new data state.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_ReadMemory32

Returns the 32-bit data from the specified memory. Suggest to use this function when your programming language doesn't support unsigned Integer ex.Visual Basic 6.0.

- **Syntax**

```
WORD Ixud_ReadMemory32(
        WORD wBoardNo,
        DWORD dwOffsetByte,
        DWORD *dwLow,
        DWORD *dwHigh
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

dwOffsetByte

[Input] Address offset

dwLow

[Output] Digital data of bit 0~15 read from the specified memory.

dwHigh

[Output] Digital data of bit 16~31 read from the specified memory.

- **Return Value**

Refer to Appendix A.1. Return Value.

Ixud_WriteMemory32

Writes the 32-bit data to the specified memory. Suggest to use this function when your programming language doesn't support unsigned Integer ex.Visual Basic 6.0.

- **Syntax**

```
WORD Ixud_WriteMemory32(
        WORD wBoardNo,
        DWORD dwOffsetByte,
        DWORD dwLow,
        DWORD dwHigh
);
```

- **Parameters**

wBoardNo

[Input] The user-assigned board number, where wBoardNo =0 is the first board, and wBoardNo=1 is the second board, and so on.

dwOffsetByte

[Input] Address offset

dwLow

[Input] New digital logic state for bit 0 ~ 15.

dwHigh

[Input] New digital logic state for bit 16 ~ 31.

- **Return Value**

Refer to Appendix A.1. Return Value.

5.3. Data Structure

PIXUD_DEVICE_INFO

- **Syntax**

```
typedef struct _IXUD_DEVICE_INFO_
{
    DWORD dwSize;
    WORD wVendorID;
    WORD wDeviceID;
    WORD wSubVendorID;
    WORD wSubDeviceID;
    DWORD dwBAR[6];
    UCHAR BusNo;
    UCHAR DevNo;
    UCHAR IRQ;
    UCHAR Aux;
    DWORD dwBarVirtualAddress[6];
}IXUD_DEVICE_INFO,*PIXUD_DEVICE_INFO;
```

- **Member**

dwSize

[Output] Structure size returned, unit is byte.

wVendorID

[Output] Vendor ID returned.

wDeviceID

[Output] Device ID returned.

wSubVendorID

[Output] Sub Vendor ID returned.

wSubDeviceID

[Output] Get Sub Device ID.

dwBAR[]

[Output] Get Base Address。

Base Address	dwBAR [Index]
Bar 0	dwBAR[0]
Bar 1	dwBAR[1]
Bar 2	dwBAR[2]
Bar 3	dwBAR[3]
Bar 4	dwBAR[4]
Bar 5	dwBAR[5]

BusNo

[Output] Bus number returned.

DevNo

[Output] Device number returned.

IRQ

[Output] IRQ number returned.

Aux

[Output] Aux ID returned.

dwBarVirtualAddress[]

[Output] Get virtual memory address for memory mapping I/O.

Virtual Memory Address	dwBAR [Index]
Bar 0	dwBarVirtualAddress [0]
Bar 1	dwBarVirtualAddress [1]
Bar 2	dwBarVirtualAddress [2]
Bar 3	dwBarVirtualAddress [3]
Bar 4	dwBarVirtualAddress [4]
Bar 5	dwBarVirtualAddress [5]

PIXUD_CARD_INFO

- **Syntax**

```
typedef struct _IXUD_CARD_INFO_
{
    DWORD dwSize;
    DWORD dwModelNo;
    UCHAR CardID;
    UCHAR wSingleEnded;
    WORD wAIOResolution;
    WORD wAIChannels;
    WORD wAOChannels;
    WORD wDIPorts;
    WORD wDOPorts;
    WORD wDIOPorts;
    WORD wDIOPortWidth;
    WORD wCounterChannels;
    WORD wMemorySize;
    DWORD dwReserved1[6];
}IXUD_CARD_INFO,*PIXUD_CARD_INFO;
```

- **Member**

dwSize

[Output] Structure size returned, unit is byte.

dwModelNo

[Output] Model number of board returned, detail information refer to A.2. Model number

CardID

[Output] Card ID returned. If returned value is 255(0xFF) that means unsupported this function.

wSingleEnded

[Output] Analog input type returned. Please refer the following table:

Value	Hex Value	Input Type
1	1	Single Ended(SE)
2	2	Differential(DIFF)
255	FF	Unsupported

wAIOResolution

[Output] Analog input and output resolution returned. High byte is analog input resolution((wAIOResolution>>8)&0xFF), low byte is analog output resolution.(wAIOResolution&0xFF)。

Value	Hex Value	Resolution
12	C	12-bit
14	E	14-bit
16	10	16-bit

wAIChannels

[Output] Number of the analog input channel returned.

wAOChannels

[Output] Number of the analog output channel returned.

wDIPorts

[Output] Number of the digital input port returned.

wDOPorts

[Output] Number of the digital output port returned.

wDIOPorts

[Output] Number of the bi-direction digital I/O port returned.

wDIOPortWidth

[Output] Bandwidth of digital input and output returned.

Value	Bandwidth
8	8-bit
16	16-bit
32	32-bit

wCounterChannels

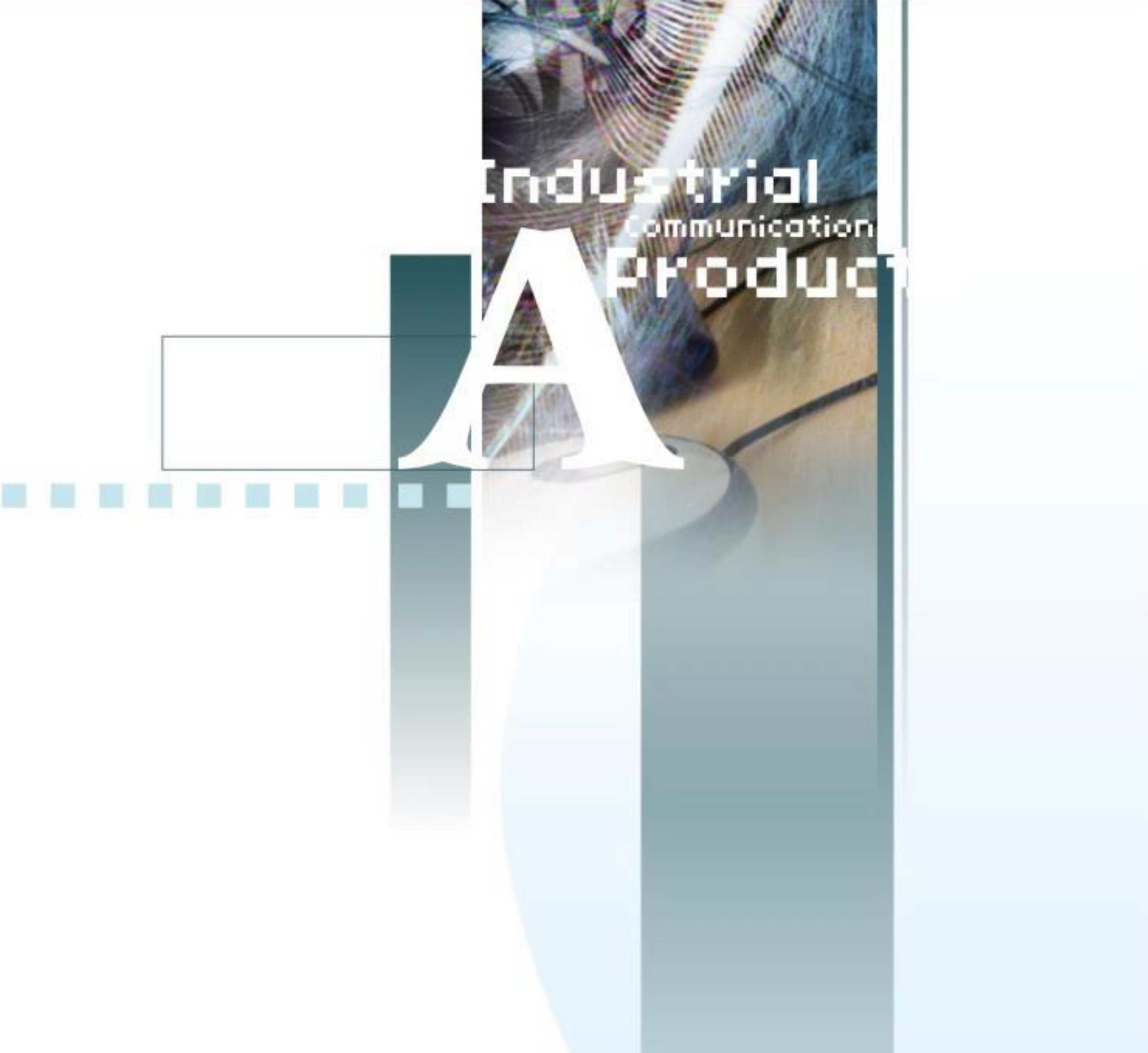
[Output] Number of counter returned.

wMemorySize

[Output] On-board memory size returned, unit is kByte.

dwReserved1[]

[Output] Reserved information



Appendix A. Return Value and Configuration code

The Appendix explains the return code and list the configuration code.

A.1. Return Value Definition

Explains the error code that might be returned when calling functions provide by the ICP DAS UniDAQ Driver DLL. Refer to this section when debugging your application.

Return Value	Error ID	Description (Error Message)
0	Ixud_NoErr	Successfully
1	Ixud_OpenDriverErr	Open Driver Failure
2	Ixud_PnPDriverErr	Plug&Play Failure
3	Ixud_DriverNoOpen	Driver was not open.
4	Ixud_GetDriverVersionErr	Get Driver Version Failure
5	Ixud_ExceedBoardNumber	Board number error
6	Ixud_FindBoardErr	Cannot Find Board
7	Ixud_BoardMappingErr	Board Mapping Error
8	Ixud_DIOModesErr	Configure DIO Port Failure
9	Ixud_InvalidAddress	Invalid Address
10	Ixud_InvalidSize	Invalid Size
11	Ixud_InvalidPortNumber	Invalid Port Number
12	Ixud_UnSupportedModel	Model Is Not Supported
13	Ixud_UnSupportedFun	Function Is Not Supported
14	Ixud_InvalidChannelNumber	Invalid Channel Number
15	Ixud_InvalidValue	Invalid Value
16	Ixud_InvalidMode	Invalid Mode
17	Ixud_GetAIStatusTimeOut	Data Not Ready
18	Ixud_TimeOutErr	Timeout
19	Ixud_CfgCodeIndexErr	Cannot Find Configuration Code Index
20	Ixud_ADCCTLTimeoutErr	ADC Timeout
21	Ixud_FindPCIIndexErr	Cannot Find Board Index
22	Ixud_InvalidSetting	Invaild Setting
23	Ixud_AllocateMemErr	Allocate Memory Space Failed
24	Ixud_InstallEventErr	Install Interrupt Event Failure
25	Ixud_InstallIrqErr	Install Interrupt IRQ Failure
26	Ixud_RemoveIrqErr	Remove Interrupt IRQ Failure
27	Ixud_ClearIntCountErr	Clear Interrupt Count Failure
28	Ixud_GetSysBufferErr	Get System Buffer Failure
29	Ixud_CreateEventErr	Call CreateEvent() Failed
30	Ixud_UnSupportedResolution	Resolution IS Not Supported
31	Ixud_CreateThreadErr	Call CreateThread() Failed

32	Ixud_ThreadTimeOutErr	Thread Timeout
33	Ixud_FIFOOverFlowErr	FIFO Overflow
34	Ixud_FIFOTimeOutErr	FIFO Timeout
35	Ixud_GetIntInstStatus	Get Installing IRQ Status Failure
36	Ixud_GetBufStatus	Get System Buffer Status Failture
37	Ixud_SetBufCountErr	Buffer Size Setting Failure
38	Ixud_SetBufInfoErr	Buffer Setting Failure
39	Ixud_FindCardIDErr	Cannot Find Card ID
40	Ixud_EventThreadErr	Event Thread Failure
41	Ixud_AutoCreateEventErr	Cannot Call CreateEvent() Automatically
42	Ixud_RegThreadErr	Register Thread Failure
43	Ixud_SearchEventErr	Cannot Find Event
44	Ixud_FifoResetErr	Cannot Clear FIFO
45	Ixud_InvalidBlock	Invalid EEPROM Block
46	Ixud_InvalidAddr	Invalid EEPROM Address
47	Ixud_AcqireSpinLock	Acquire Spin Lock Failure
48	Ixud_ReleaseSpinLock	Release Spin Lock Failure
49	Ixud_SetControlErr	Analog Input Setting Error
50	Ixud_InvalidChannels	Invalid Channel
51	Ixud_SearchCardErr	Search Card Failure
52	Ixud_SetMapAddressErr	Set Address Mapping Failure
53	Ixud_ReleaseMapAddressErr	Release Address Mapping Failure
54	Ixud_InvalidOffset	Invalid Offset
55	Ixud_ShareHandleErr	Open Share Memory Failed
56	Ixud_InvalidDataCount	Invalid number of data
57	Ixud_WriteEEPErr	Write EEPROM Failed
58	Ixud_CardIOErr	Use CardIO error
59	Ixud_IOErr	Use MemoryIO error
60	Ixud_SetScanChannelErr	Set channel scan number error
61	Ixud_SetScanConfigErr	Set channel scan configuration error
62	Ixud_GetMMIOMapStatus	Get Memory Mapping IO Status error

A.2. Model number

ID	Value(HEX)	Supported DAQ board
PIOD56	800140	PIO-D24/D56/D24U/D56U
PEXD56	800140	PEX-D24/D56
PIOD48	800130	PIO-D48/D48U/D48SU
PEXD48	800130	PEX-D48
PIOD64	800120	PIO-D64/D64U
PIOD96	800110	PIO-D96/D96U/D96SU
PEXD96	800110	PEX-D96S
PIOD144	800100	PIO-D144
PEXD144	800100	PEX-D144LS
PIOD168	800150	PIO-D168
PIODA	800400	PIO-DA4/DA8/DA16/DA4U/DA8U/DA16U/PISO-DA4U/DA8U/DA16U
PEXDA	800400	PEX-DA4/DA8/DA16
PIO821	800310	PIO-821 L/H/LU/HU
PISOP16R16U	1800FF	PISO-P16R16U/P16R16E
PEXP16R16	1800FF	PEX-P16R16i
PEXP8R8	1800FF	PEX-P8R8i
PISOC64	800800	PISO-C64
PEXC64	800800	PEX-C64
PISOP64	800810	PISO-P64
PEXP64	800810	PEX-P64
PISOA64	800850	PISO-A64/A64U
PISOP32C32	800820	PISO-P32C32/P32C32U/P32S32WU
PEXP32C32	800820	PEX-P32C32
PISO1730	800820	PISO-1730U
PISOP32A32	800870	PISO-P32A32/P32A32U/ P32A32U-5V
PEXP32A32	800870	PEX-P32A32
PISOP8R8	800830	PISO-P8R8/PISO-P8R8AC/PISO-P8R8DC/ P8SSR8AC
PISO730	800840	PISO-730
PEX730	800840	PEX-730
PISO730A	800880	PISO-730A/730AU
PEX730A	800880	PEX-730A
PISO725	8008FF	PISO-725/725U
PISODA2	800B00	PISO-DA2
PISO813	800A00	PISO-813/813U
PCITMC12	DF2962	PCI-TMC12/PCI-TMC12A/TMC12AU
PEXTMC12	DF2962	PEX-TMC12A
PCIM512	DE9562	PCI-M512
PCIM256	DE92A6	PCI-M256
PCIM128	DE9178	PCI-M128

PCIFC16	B13017	PCI-FC16U
PCID64	DE3513	PCI-D64
PCI822	DE3823	PCI-822 LU
PCI826	DE3827	PCI-826 LU
PCI2602	2CB656	PCI-2602U
PCI100X	341002	PCI-1002 LU/HU
PEX100X	341002	PEX-1002
PCI1202	345672	PCI-1202 L/H ,PCI-1202U L/H
PEX1202	345672	PEX-1202 L/H
PCI1602	345676	PCI-1602/1602U,PCI-1602 F
PCI180X	345678	PCI-1800 L/H, PCI-1802 L/H
PCIP8R8	D6102B	PCI-P8R8/P8R8U
PEXP8POR8	D6102B	PEX-P8POR8i
PCIP16R16	D61E39	PCI-P16R16/P16R16U/P16C16/ P16C16U/P16POR16/ P16POR16U
PEXP16POR16	D61E39	PEX-P16POR16i
PISO1730	800820	PISO-1730U
PCIE8620	658627	PCIe-8620
PCIE8622	658629	PCIe-8622
PCID96	80D096	PCI-D96SU
PCID128	80D128	PCI-D128SU

A.3. Configuration Code Definition

Configuration code can change the hardware setting. Ex. Change the analog input range then adjust the different input range to increase the accuracy.

A.3.1. AI Configuration Code

User can inquire the following table to set analog input range and polarity, each board have the different analog input range and polarity. For detailed information refer to hardware manual or ICPDAS Board Analog Input Configuration Code Supported Table.

Value	ID	Polarity	Range(Voltage)
0	IXUD_BI_10V	Bipolar	+/- 10V
1	IXUD_BI_5V	Bipolar	+/- 5V
2	IXUD_BI_2V5	Bipolar	+/- 2.5V
3	IXUD_BI_1V25	Bipolar	+/- 1.25V
4	IXUD_BI_0V625	Bipolar	+/- 0.625V
5	IXUD_BI_0V3125	Bipolar	+/- 0.3125V
6	IXUD_BI_0V5	Bipolar	+/- 0.5V
7	IXUD_BI_0V05	Bipolar	+/- 0.05V
8	IXUD_BI_0V005	Bipolar	+/- 0.005
9	IXUD_BI_1V	Bipolar	+/- 1V
10	IXUD_BI_0V1	Bipolar	+/- 0.1V
11	IXUD_BI_0V01	Bipolar	+/- 0.01V
12	IXUD_BI_0V001	Bipolar	+/- 0.001V
13	IXUD_UNI_20V	Unipolar	0 ~ 20V
14	IXUD_UNI_10V	Unipolar	0 ~ 10V
15	IXUD_UNI_5V	Unipolar	0 ~ 5V
16	IXUD_UNI_2V5	Unipolar	0 ~ 2.5V
17	IXUD_UNI_1V25	Unipolar	0 ~ 1.25V
18	IXUD_UNI_0V625	Unipolar	0 ~ 0.625V
19	IXUD_UNI_1V	Unipolar	0 ~ 1V
20	IXUD_UNI_0V1	Unipolar	0 ~ 0.1V
21	IXUD_UNI_0V01	Unipolar	0 ~ 0.01V
22	IXUD_UNI_0V001	Unipolar	0 ~ 0.001V
23	IXUD_BI_20V	Bipolar	+/- 20V

ICPDAS Board Analog Input Configuration Code Supported

Voltage Range	PIO-821L PIO-821LU	PIO-821H PIO-821HU	PISO-813 PIO-813U (JP1=10V)	PISO-813 PIO-813U (JP1=20V)	PCI-1002LU PEX-1002L	PCI-1002HU PEX-1002H
+/- 10V				✔	✔	✔
+/- 5V	✔	✔	✔	✔	✔	
+/- 2.5V	✔		✔	✔	✔	
+/- 1.25V	✔		✔	✔	✔	
+/- 0.625V	✔		✔	✔		
+/- 0.3125V						
+/- 0.5V		✔				
+/- 0.05V		✔				
+/- 0.005		✔				
+/- 1V						✔
+/- 0.1V						✔
+/- 0.01V						✔
+/- 0.001V						
0 ~ 20V						
0 ~ 10V			✔			
0 ~ 5V			✔			
0 ~ 2.5V			✔			
0 ~ 1.25V			✔			
0 ~ 0.625V			✔			
0 ~ 1V						
0 ~ 0.1V						
0 ~ 0.01V						
0 ~ 0.001V						

ICPDAS Board Analog Input Configuration Code Supported

Voltage Range	PCI-1202LU PCI-1800LU PCI-1802LU PEX-1202L	PCI-1202HU PCI-1800HU PCI-1802HU PEX-1202H	PCI-1602 PCI-1602U PCI-1602F PCI-1602FU	PCI-822LU PCI-826LU	PCI-2602U	PCIe-8620 PCIe-8622
+/- 10V	✔	✔	✔	✔	✔	✔
+/- 5V	✔	✔	✔	✔	✔	✔
+/- 2.5V	✔		✔	✔	✔	
+/- 1.25V	✔		✔	✔	✔	
+/- 0.625V	✔				✔	
+/- 0.3125V						
+/- 0.5V		✔				
+/- 0.05V		✔				
+/- 0.005		✔				
+/- 1V		✔				
+/- 0.1V		✔				
+/- 0.01V		✔				
+/- 0.001V						
0 ~ 20V						
0 ~ 10V	✔	✔				
0 ~ 5V	✔					
0 ~ 2.5V	✔					
0 ~ 1.25V	✔					
0 ~ 0.625V						
0 ~ 1V		✔				
0 ~ 0.1V		✔				
0 ~ 0.01V		✔				
0 ~ 0.001V						

PCI-2602U Analog Input Configuration Code

Voltage Setting	Voltage Range
+/- 10V	+/- 10.24V
+/- 5V	+/- 5.12V
+/- 2.5V	+/- 2.56V
+/- 1.25V	+/- 1.28V
+/- 0.625V	+/- 0.64V

A.3.2. AO Configuration Code(Voltage)

User can inquire the following table to set analog output range and polarity, each board have the different analog input range and polarity. For detailed information refer to hardware manual or ICPDAS Board Analog Input Configuration Code Supported Table.

Code	ID	Voltage Range
0	IXUD_AO_UNI_5V	0 ~ 5V
1	IXUD_AO_BI_5V	+/- 5V
2	IXUD_AO_UNI_10V	0 ~ 10V
3	IXUD_AO_BI_10V	+/- 10V
4	IXUD_AO_UNI_20V	0 ~ 20V
5	IXUD_AO_BI_20V	+/- 20V

ICPDAS Board Analog Output Configuration Code Supported

Code	Voltage Range	PIO-DA4U PIO-DA8U PIO-DA16U	PISO-DA4U PISO-DA8U PISO-DA16U	PIO-821L PIO-821H PIO-821LU PIO-821HU	PISO-DA2U	PCI-1202 PCI-1602 PCI-1800 PCI-1802 PEX-1202	PCI-822 PCI-826 PCI-2602U	PCIe-8622
0	0 ~ 5V	-	-	✔	✔	-	✔	✔
1	+/- 5V	-	-	✔	✔	✔	✔	✔
2	0 ~ 10V	-	-	-	✔	-	✔	✔
3	+/- 10V	✔	✔	-	✔	✔	✔	✔

A.3.3. AO Configuration Code (Current)

User can inquire the following table to set analog output range and polarity, each board have the different analog input range and polarity. For detailed information refer to hardware manual or ICPDAS Board Analog Input Configuration Code Supported Table.

Code	ID	Current Range
16	IXUD_AO_I_0_20_MA	0 ~ 20 mA
17	IXUD_AO_I_4_20_MA	4 ~ 20 mA

ICPDAS Board Analog Output Configuration Code Supported

Code	Current Range(mA)	PIO-DA4U PIO-DA8U PIO-DA16U	PISO-DA4U PISO-DA8U PISO-DA16U	PEX-DA4 PEX-DA8 PEX-DA16	PISO-DA2U
16	0 ~ 20	✔	✔	✔	✔
17	4~20	-	-		✔

A.3.4. Interrupt Event Configuration Code

Supported Event Types

Value	Type	Description
1	IXUD_HARDWARE_INT	Device generated a Hardware interrupt
2	IXUD_APC_READY_INT	Interrupt generated from analog input data ready.
4	IXUD_ACTIVE_LOW	Interrupt generated from digital input port failing edge
8	IXUD_ACTIVE_HIGH	Interrupt generated from digital input port raising edge.

A.4. DI Port Number Definition

DI Port No.	PIO-D24U PEX-D24	PIO-D56U PEX-D56	PIO-D48U PIO-D48SU PEX-D48	PIO-D64U	PIO-D96U PIO-D96SU PEX-D96S	PIO-D144 PIO-D144U PIO-D144LU PEX-D144LS	PIO-D168U	PISO-P64 PISO-P64U PEX-P64
0	**CN3 Port0**	**CN3 Port0**	**CN1 Port0**	CN2 DI 0 ~ 7	**CN1 Port0**	**CN1 Port0**	**CN1 Port0**	IDI 0 ~ 7
1	**CN3 Port1**	**CN3 Port1**	**CN1 Port1**	CN2 DI 8 ~ 15	**CN1 Port1**	**CN1 Port1**	**CN1 Port1**	IDI 8 ~ 15
2	**CN3 Port2**	**CN3 Port2**	**CN1 Port2**	CN4 DI 0 ~ 7	**CN1 Port2**	**CN1 Port2**	**CN1 Port2**	IDI 16 ~ 23
3	-	CN2 DI 0 ~ 7	**CN2 Port3**	CN4 DI 8 ~ 15	**CN2 Port3**	**CN2 Port3**	**CN2 Port3**	IDI 24 ~ 31
4	-	CN2 DI 8 ~ 15	**CN2 Port4**	-	**CN2 Port4**	**CN2 Port4**	**CN2 Port4**	IDI 32 ~ 39
5	-	-	**CN2 Port5**	-	**CN2 Port5**	**CN2 Port5**	**CN2 Port5**	IDI 40 ~ 47
6	-	-	-	-	**CN3 Port6**	**CN3 Port6**	**CN3 Port6**	IDI 48 ~ 55
7	-	-	-	-	**CN3 Port7**	**CN3 Port7**	**CN3 Port7**	IDI 56 ~ 63
8	-	-	-	-	**CN3 Port8**	**CN3 Port8**	**CN3 Port8**	
9	-	-	-	-	**CN4 Port9**	**CN4 Port9**	**CN4 Port9**	
10	-	-	-	-	**CN4 Port10**	**CN4 Port10**	**CN4 Port10**	
11	-	-	-	-	**CN4 Port11**	**CN4 Port11**	**CN4 Port11**	
12	-	-	-	-	-	**CN5 Port12**	**CN5 Port12**	
13	-	-	-	-	-	**CN5 Port13**	**CN5 Port13**	
14	-	-	-	-	-	**CN5 Port14**	**CN5 Port14**	
15	-	-	-	-	-	**CN6 Port15**	**CN6 Port15**	
16	-	-	-	-	-	**CN6 Port16**	**CN6 Port16**	
17	-	-	-	-	-	**CN6 Port17**	**CN6 Port17**	
18	-	-	-	-	-	-	**CN6 Port18**	
19	-	-	-	-	-	-	**CN6 Port19**	
20	-	-	-	-	-	-	**CN6 Port20**	

DI Port No.	PISO-P32A32U PISO-P32A32U-5V PISO-P32C32U PISO-P32S32WU PISO-1730U PEX-P32C32 PEX-P32A32	PISO-P16R16U PEX-P16R16i PCI-P16R16U	PISO-P8R8U PISO-P8SSR8AC PEX-P8R8i PCI-P8R8U PISO-725 PISO-725U	PISO-730 PISO-730U PISO-730A PISO-730AU PEX-730 PEX-730A	PCI-P8R8 PEX-P8POR8i	PCI-P16R16 PCI-P16C16 PCI-P16C16U PEX-P16POR16i PCI-P16POR16U
0	CN1 IDI 0 ~ 7	**CN1 IDI 0 ~ 7**	**CN1 IDI 0 ~ 7**	CN1 IDI 0 ~ 7	IDI 0 ~ 7	IDI 0 ~ 15
1	CN1 IDI 8 ~ 15	**CN2 IDI 8 ~ 15**	-	CN1 IDI 8 ~ 15	-	-
2	CN2 IDI 16 ~ 23	-	-	CN2 DI 0 ~ 7	-	-
3	CN2 IDI 24 ~ 31	-	-	CN2 DI 8 ~ 15	-	-

DI Port No.	PCI-822LU PCI-826LU PCI-FC16U	PIO-821L PIO-821H PIO-821LU PIO-821HU	PIO-DA4U PIO-DA8U PIO-DA16U	PISO-DA4U PISO-DA8U PISO-DA16U	PEX-DA4 PEX-DA8 PEX-DA16	PCI-1002 PEX-1002	PCI-1202 PEX-1202	PCI-1602 PCI-1800 PCI-1802
0	**PA 0 ~ 15**	DI 0~7	DI 0 ~ 7	DI 0 ~ 7	DI 0 ~ 7	DI 0 ~ 15	DI 0 ~ 15	DI 0 ~ 15
1	**PB 0 ~ 15**	DI 8~15	DI 8 ~ 15	DI 8 ~ 15	DI 8 ~ 15	-	-	-

DI Port No.	PCI-M512 PCI-M512U	PCI-TMC12 PCI-TMC12A PCI-TMC12AU PEX-TMC12A	PCI-2602U	PCI-D96SU	PCI-D128SU	PCIe-8620	PCIe-8622
0	DI 0 ~ 11	DI 0 ~ 15	**PA0~7** **PB0~7** **PC0~7** **PD0~7**	**CON1 Port0**	**CON1 Port0**	**DI 0~3**	**DI 0~11**
1				**CON1 Port1**	**CON1 Port1**		
2				**CON1 Port2**	**CON1 Port2**		
3					**CN1/2 Port3**		

🟩 **Bi-Direction digital I/O Port** 🟦 **Digital Input Port**

⚠ Bi-Direction digital I/O Port must use the Ixud_SetDIOModes32 or Ixud_SetDIOMode function to set the input mode.

A.5. DO Port Number Definition

DO Port No.	PIO-D24U PEX-D24	PIO-D56U PEX-D56	PIO-D48U PIO-D48SU PEX-D48	PIO-D64U	PIO-D96U PIO-D96SU PEX-D96S	PIO-D144 PIO-D144U PIO-D144LU PEX-D144LS	PIO-D168U	PISO-A64 PISO-A64U PISO-C64 PISO-C64U PEX-C64
0	**CN3 Port0**	**CN3 Port0**	**CN1 Port0**	CN1 DO 0 ~ 7	**CN1 Port0**	**CN1 Port0**	**CN1 Port0**	IDO 0 ~ 7
1	**CN3 Port1**	**CN3 Port1**	**CN1 Port1**	CN1 DO 8 ~ 15	**CN1 Port1**	**CN1 Port1**	**CN1 Port1**	IDO 8 ~ 15
2	**CN3 Port2**	**CN3 Port2**	**CN1 Port2**	CN3 DO 0 ~ 7	**CN1 Port2**	**CN1 Port2**	**CN1 Port2**	IDO 16 ~ 23
3	-	CN1 DO 0 ~ 7	**CN2 Port3**	CN3 DO 8 ~ 15	**CN2 Port3**	**CN2 Port3**	**CN2 Port3**	IDO 24 ~ 31
4	-	CN1 DO 8 ~ 15	**CN2 Port4**	-	**CN2 Port4**	**CN2 Port4**	**CN2 Port4**	IDO 32 ~ 39
5	-	-	**CN2 Port5**	-	**CN2 Port5**	**CN2 Port5**	**CN2 Port5**	IDO 40 ~ 47
6	-	-	-	-	**CN3 Port6**	**CN3 Port6**	**CN3 Port6**	IDO 48 ~ 55
7	-	-	-	-	**CN3 Port7**	**CN3 Port7**	**CN3 Port7**	IDO 56 ~ 63
8	-	-	-	-	**CN3 Port8**	**CN3 Port8**	**CN3 Port8**	
9	-	-	-	-	**CN4 Port9**	**CN4 Port9**	**CN4 Port9**	
10	-	-	-	-	**CN4 Port10**	**CN4 Port10**	**CN4 Port10**	
11	-	-	-	-	**CN4 Port11**	**CN4 Port11**	**CN4 Port11**	
12	-	-	-	-	-	**CN5 Port12**	**CN5 Port12**	
13	-	-	-	-	-	**CN5 Port13**	**CN5 Port13**	
14	-	-	-	-	-	**CN5 Port14**	**CN5 Port14**	
15	-	-	-	-	-	**CN6 Port15**	**CN6 Port15**	
16	-	-	-	-	-	**CN6 Port16**	**CN6 Port16**	
17	-	-	-	-	-	**CN6 Port17**	**CN6 Port17**	
18	-	-	-	-	-	-	**CN6 Port18**	
19	-	-	-	-	-	-	**CN6 Port19**	
20	-	-	-	-	-	-	**CN6 Port20**	

DO Port No.	PISO-P32A32U PISO-P32A32U-5V PISO-P32C32U PISO-P32S32WU PISO-1730U PEX-P32C32 PEX-P32A32	PISO-P16R16U PEX-P16R16i PCI-P16R16U	PISO-P8R8U PISO-P8SSR8AC PEX-P8R8i PCI-P8R8U PISO-725 PISO-725U	PISO-730 PISO-730A PISO-730U PISO-730AU PEX-730 PEX-730A	PCI-P8R8 PEX-P8POR8i	PCI-P16R16 PCI-P16C16 PCI-P16C16U PEX-P16POR16i PCI-P16POR16U
0	CN1 IDO 0 ~ 7	**CN1 IDO 0 ~ 7**	**CN1 IDO 0 ~ 7**	CN1 IDO 0 ~ 7	IDO 0 ~ 7	IDO 0 ~ 15
1	CN1 IDO 8 ~ 15	**CN2 IDO 8 ~ 15**	-	CN1 IDO 8 ~ 15	-	-
2	CN2 IDO 16 ~ 23	-	-	CN2 DO 0 ~ 7	-	-
3	CN2 IDO 24 ~ 31	-	-	CN2 DO 8 ~ 15	-	-

DO Port No.	PCI-822LU PCI-826LU PCI-FC16U	PIO-821L PIO-821H PIO-821LU PIO-821HU	PIO-DA4U PIO-DA8U PIO-DA16U	PISO-DA4U PISO-DA8U PISO-DA16U	PEX-DA4 PEX-DA8 PEX-DA16	PCI-1002 PEX-1002	PCI-1202 PEX-1202	PCI-1602 PCI-1802
0	**PA 0 ~ 15**	DO 0~7	DO 0 ~ 7	DO 0 ~ 7	DO 0 ~ 7	DO 0 ~ 15	DO 0 ~ 15	DO 0 ~ 15
1	**PB 0 ~ 15**	DO 8~15	DO 8 ~ 15	DO 8 ~ 15	DO 8 ~ 15	-	-	

DO Port No.	PCI-M512	PCI-TMC12 PCI-TMC12A PCI-TMC12AU PEX-TMC12A	PCI-2602U	PCI-D96SU	PCI-D128SU	PCIe-8620	PCIe-8622
0	DO 0 ~ 15	DO 0 ~ 15	**PA 0 ~ 7** **PB 0 ~ 7** **PC 0 ~ 7** **PD 0 ~ 7**	**CON1 Port0**	**CON1 Port0**	**DO 0~3**	**DO 0~11**
	-	-	-	**CON1 Port1**	**CON1 Port1**	-	-
	-	-	-	**CON1 Port2**	**CON1 Port2**	-	-
	-	-	-	-	**CN1/2 Port3**	-	-

Bi-Direction digital I/O Port must use the Ixud_SetDIOModes32 or Ixud_SetDIOMode function to set the output mode.

Appendix B. Other

This appendix will provide supplementary information.

B.1. FAQ

System and Install

Q. Does UniDAQ supports 64-bit Windows?

A. Yes, it supports 64-bit Windows XP/2003/Vista/7/2008/8.

Q. If I change the classic driver to UniDAQ driver. Do I need to modify the program?

A. Yes, the API function of the classic is different from the UniDAQ driver.

Q. I don’t know the driver that is the classic or UniDAQ driver.

A. Please check the device name on the device manager. If the device name have the key word -[UniDAQ] that means UniDAQ driver, otherwise is classic driver.

Q. If system must increase the new board to implement the new project, the old board uses the classic driver, the new board uses the UniDAQ driver. Because, user doesn’t modify the software for old board. Can user use the UniDAQ to develop the new board?

A. Yes, the old board uses the classic driver, the new board uses the UniDAQ driver.

Q. Does UniDAQ support the ISA bus board?

A. UniDAQ doesn’t support the ISA bus board.

Digital Input /Output

Q. When use PIO-D24U/D56U/D48U/D96U/D144U/D168U board, the digital output or input function doesn't work?

A. Because the digital I/O port of PIO-D24U/D56U/D48U/D96U/D144U/D168U is bi-direction digital I/O port. User must set the mode for port, please use the Ixud_SetDIOModes32 or Ixud_SetDIOMode function to set port mode at first.

Analog Outupt

Q. When use the PIO-DA4U/8U/16U or PISO-DA4U/8U/16U to output incorrect voltage or current on range = ±5V, 0 ~ 10V, 0 ~ 5V and 4~20mA.

A. The hardware design of thePIO-DA4U/8U/16U and PISO-DA4U/8U/16U only support the ±10V voltage and 0 ~ 20 mA, if user set the other range, it will output the incorrect voltage or current.

Q. When call the analog output function to output the incorrect voltage or current.

A. Please check your analog output range setting, it must call the Ixud_ConfigAO function to set the correct range and then call the Ixud_WriteAOVoltage or Ixud_WriteAOCurrent function to output voltage or current.

Troubleshooting for function return code

Q. Error code 1.

A. Please reinstall the UniDAQ driver or reboot the PC.

Q. Error code 2.

A. (1) Please call the Ixud_DriverInit function to initial the UniDAQ driver at first.

(2) Use the invalid BoardNo, please check the BoardNo for function parameter. The first board is wBoardNo =0.

Q. Error code 5.

A. Use the invalid BoardNo, please check the BoardNo for function parameter. The first board is wBoardNo =0.

Q. Error code 6.

A. If it doesn’t find any board, please install ICPDAS board and restart the program.

Q. Error code 13

A. This board doesn’t support this function.

Q. Error code 19.

A. Please set the correct analog input range.

B.2. Revision History

Revision	Date	Description
1.0	Sep. 2009	Initial issue
1.3	Sep. 2011	Add new function
2.0	Sep. 2012	Add starting, tutorial and function overview chapter.
2.1	Dec. 2012	Modify Interrupt Event Configuration Code Modify interrupt support list
2.2	May. 2013	Modify Cardtype parameter description for PISO-813 Add channel scan support description for PCI-1002
2.3	Feb. 2014	Add the new production Add the several new API function.
2.5	Aug. 2019	Modify some API error description Add the new production
2.6	Dec. 2019	Modify function support list and related table