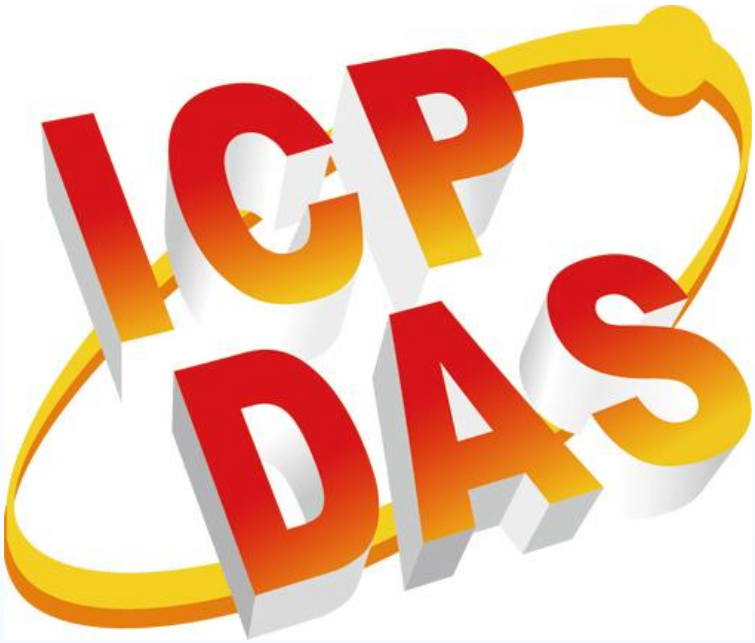


ICP DAS

Driver & SDK

UNiDAQ

Driver DLL User Manual

English Version

Support 64-bit OS

Support Windows 8

Support most PCI I/O Board

➢ 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, not for any infringements of patents or other rights of third parties resulting from its use.

➢ Copyright

➢ Trademark

The names used for identification only may be registered trademarks of their respective companies.

About

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

User can use UniDAQ DLL drivers to develop program with VB, VC, BCB, Delphi, VB.NET, C#.NET, VC.NET, Console and other programming languages on Windows System. This manual also provides the sample program. Use can modify these sample applications to suit your needs.

If you have any question, please feel free to contact ICPDAS engineer.

Email: service@icpdas.com

Table of Contents

Table Of Contents

Table Of Contents

Table Of Contents

Table Of Contents

1. Introduction

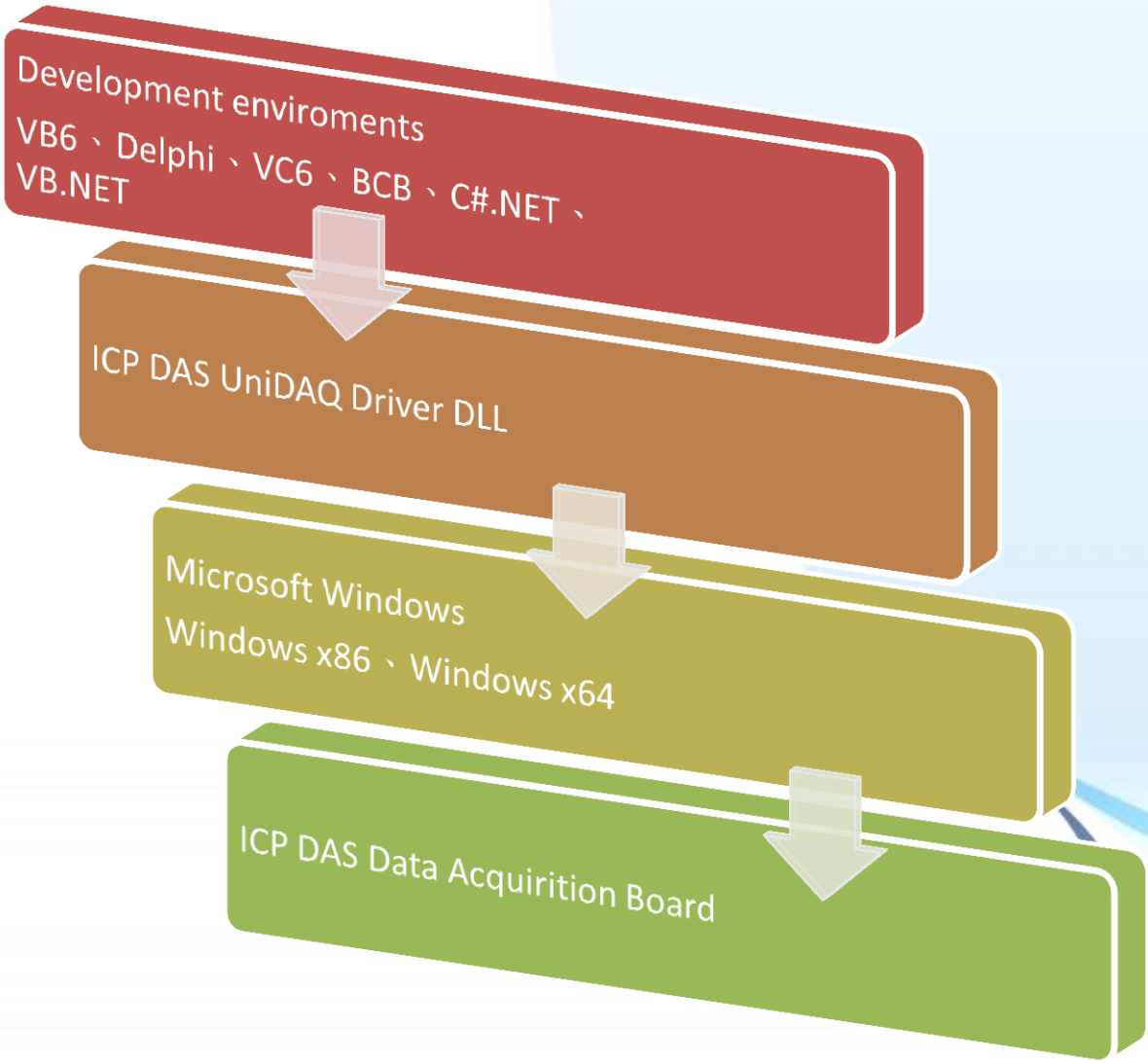
Introduce the functions and requirement for ICP DAS UniDAQ Driver DLL

1.1. About the UniDAQ Driver DLL

The ICP DAS UniDAQ Driver DLL provides complete hardware functions and maximum performance. With the ICP DAS UniDAQ Driver DLL, user doesn't have to use hardware-specific register commands, it provides user a powerful API function for use with a variety of programming environments and language.

ICP DAS UniDAQ Driver DLL use direct I/O techniques to promote API efficiency and I/O speed. It also supports interrupt and event notification functions. When interrupt events occur within the device, it notifies user application by posting callback function. User only has to take the necessary actions without checking hardware status manually. It is more efficient and reduces the complexity of the application.

ICP DAS UniDAQ Driver DLL supports Windows 2000 and 32/64 bit Windows XP/2003/Vista/7/2008/8.



1.2. Support in ICP DAS Product

The following table shows ICP DAS hardware that supports ICP DAS UniDAQ Driver DLL.

Product model	Product Model
PIO-D24/D56/D24U/D56U、PEX-D24/D56	PIO-D48/D48U/D48SU、PEX-D48
PIO-D64/D64U	PIO-D96/D96U/D96SU
PIO-D144/D144U	PIO-D168/D168U
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/P32A32/P32A32U	PISO-P32C32/P32C32U/P32S32WU
PEX-P32C32	PISO-P8R8/P8R8U
PISO-P8R8AC/P8R8DC	PISO-P16R16U、PEX-P16R16i/P8R8i
PISO-730/730A/730U/730AU	PISO-725
PISO-DA2/DA2U	PISO-813/813U
PCI-TMC12/PCI-TMC12A	PCI-M512/M256/M128/M512U
PCI-P16R16/P16C16/P16POR16/P8R8	PEX-P16ROR16i/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

1.3. System requirement

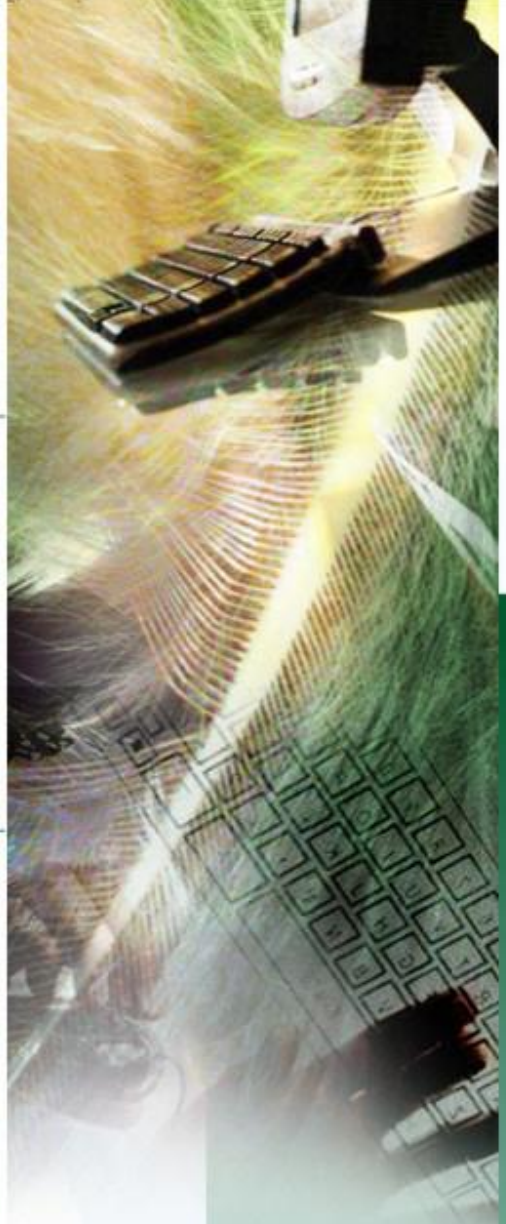
Minimum system requirements for ICP DAS UniDAQ Driver DLL are given below:

- 266MHz 32-bit(x86) or 64-bit(x64) processor
- 64 MB of system memory
- Support for Super VGAgraphics
- At least 20 MB of available space
- DVD/CD-ROM driver
- Microsoft Windows 2000 or later(32-bit or 64-bit Windows Operating System)

Operating system of Windows requirement

32-bit(x86)	64-bit(x64)
Microsoft Windows 2000	**-**
Microsoft Windows XP 32-bit	**Microsoft Windows XP 64-bit**
Microsoft Windows 2003 32-bit	**Microsoft Windows 2003 64-bit**
Microsoft Windows Vista 32-bit	**Microsoft Windows Vista 64-bit**
Microsoft Windows 7 32-bit	**Microsoft Windows 7 64-bit**
Microsoft Windows 2008 32-bit	**Microsoft Windows 2008 64-bit**
Microsoft Windows 8 32-bit	**Microsoft Windows 8 64-bit**
-	**Microsoft Windows 2012 64-bit**

PS： Microsoft Windows 3.1/95/98/ME/NT not supported



2. Starting

Introduces to get and install the ICP DAS UniDAQ Driver DLL

2.1. Getting the UniDAQ Driver DLL Installer package

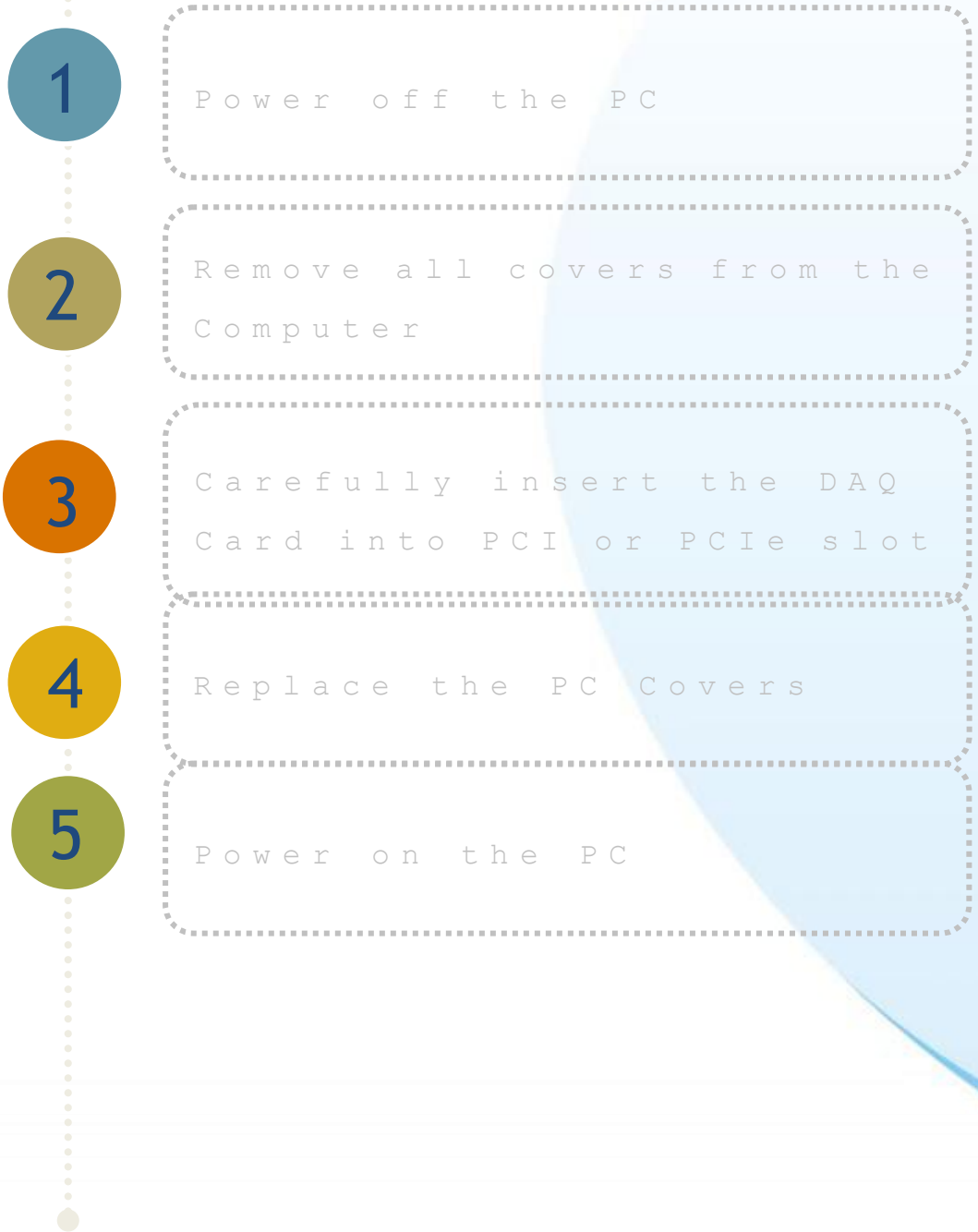
User can get the installer of the ICP DAS UniDAQ Driver DLL from CD-ROM, FTP site and web site, the file path is as below:

	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. Install UniDAQ Driver DLL

Step1 Install the DAQ Card

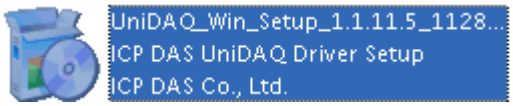
Please follow the following steps to install DAQ card:



Step2 Setup the ICP DAS UniDAQ Driver DLL

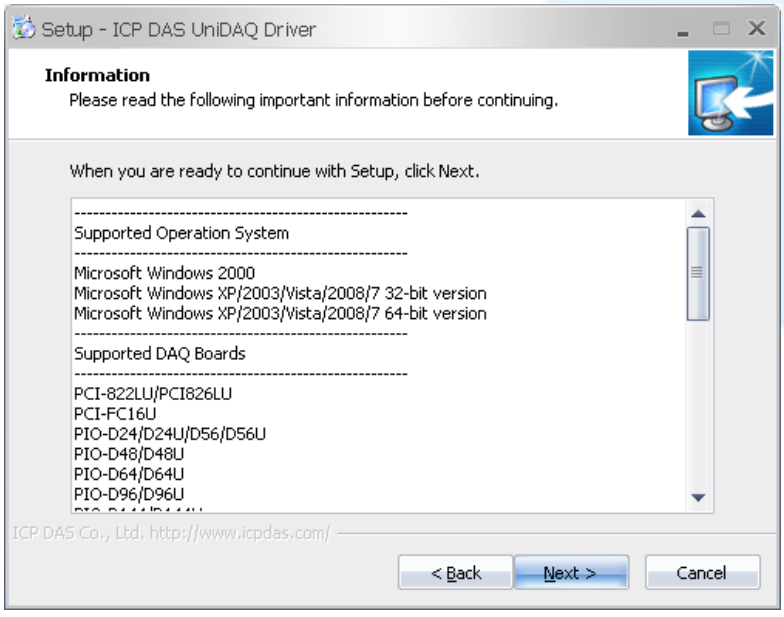
Please follow the following steps to setup software

1. Double Click the UniDAQ_Win_Setupxxx.exe to setup it.

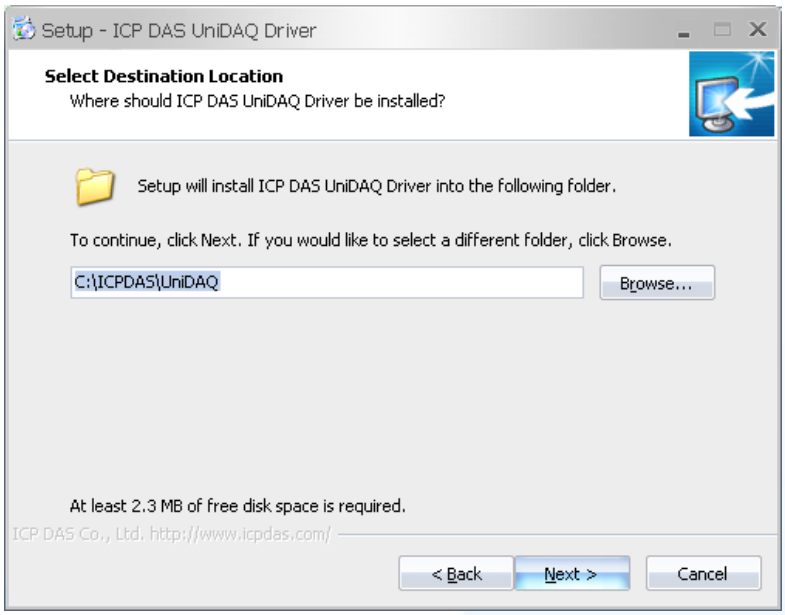


2. Click the Next> button

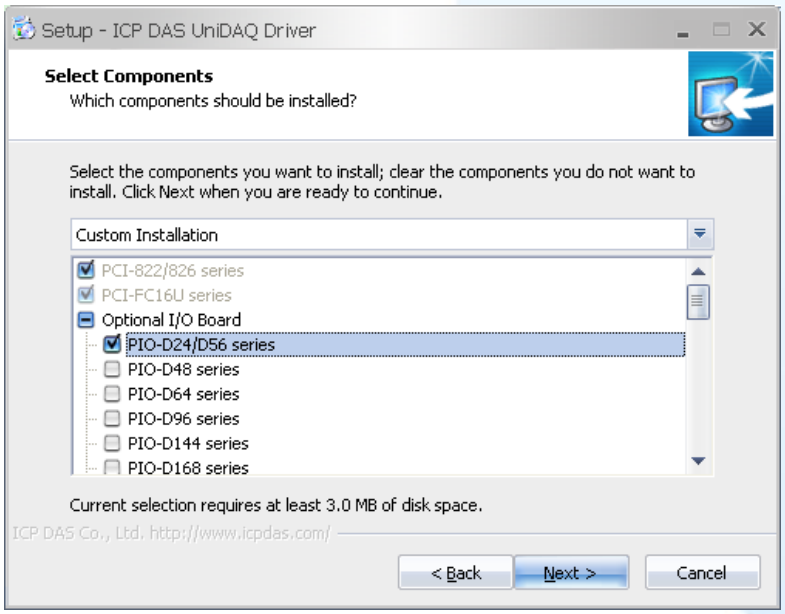
3. Check your DAQ Card is or not on supported list, click the Next> button



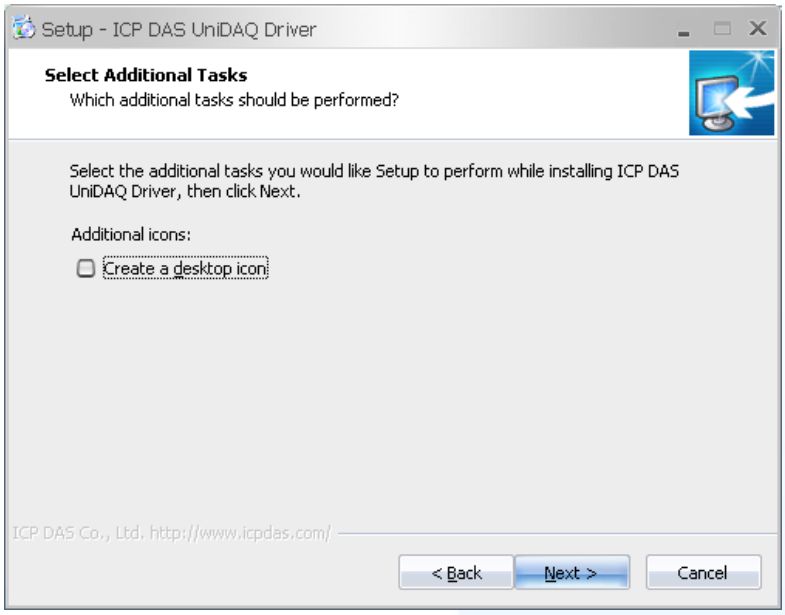
4. Select the installation folder, the default path is C:\ICPDAS\UniDAQ, confirm and click then Next> button



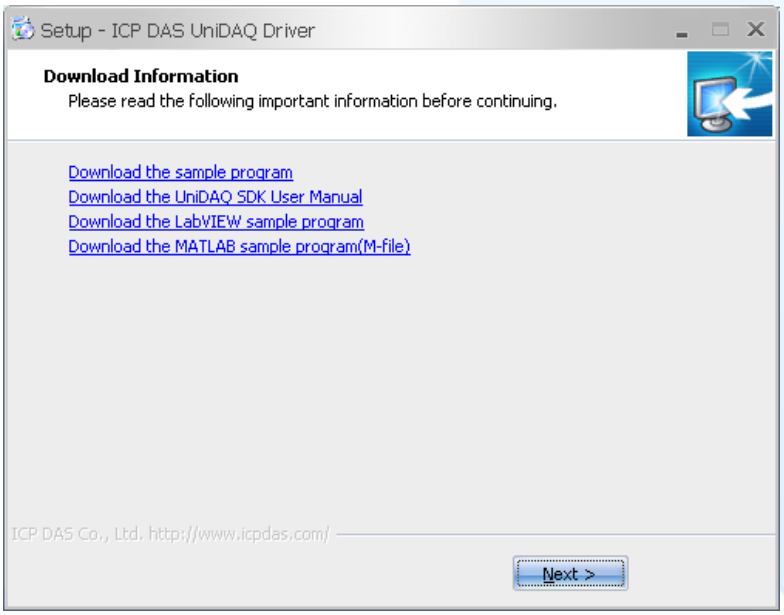
5. Check your DAQ card on the list, then click the Next> button



6. Click the Next> button

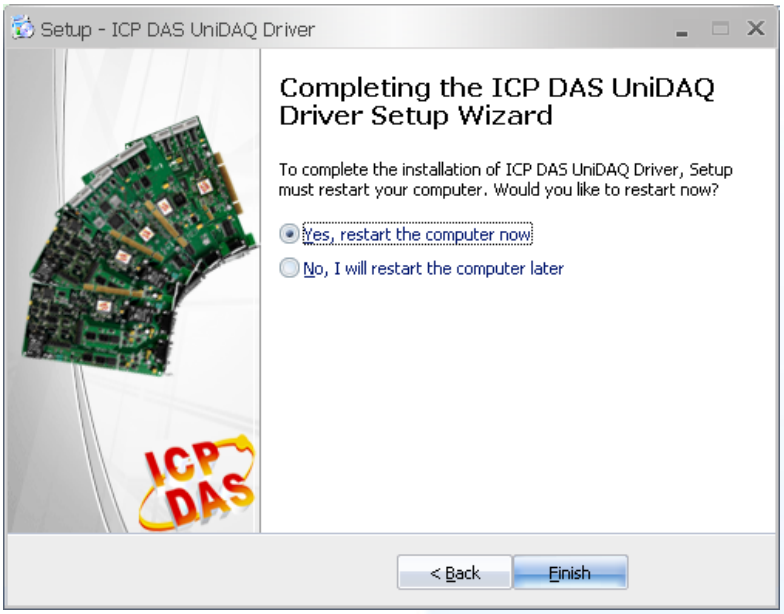


7. Click the Next> button



8. Select the item “Yes，restart the computer now”, press the Finish button. System will reboot.

Complete the ICP DAS UniDAQ Driver DLL setup after reboot the PC.

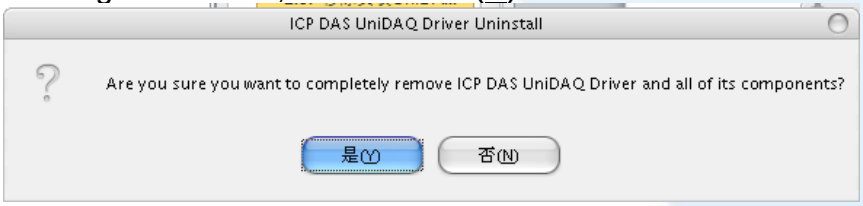


2.3. Uninstalling the UniDAQ Driver DLL

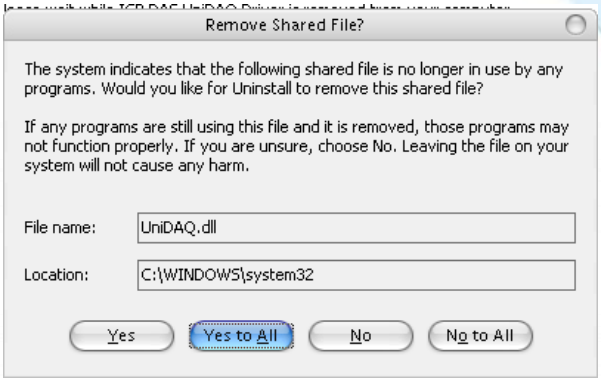
ICP DAS UniDAQ Driver DLL includes an uninstall utility to help user remove the software from your computer. To uninstall the software, complete the following procedures:

1. Select Settings|Control Panel|Add/Remove Programs from the Windows Start menu.
2. Click the Install/Uninstall tab and highlight the item ICP DAS UniDAQ Windows Driver and then click the remove button.

3. When the message box loads, click the Yes(Y) button to uninstall the software.



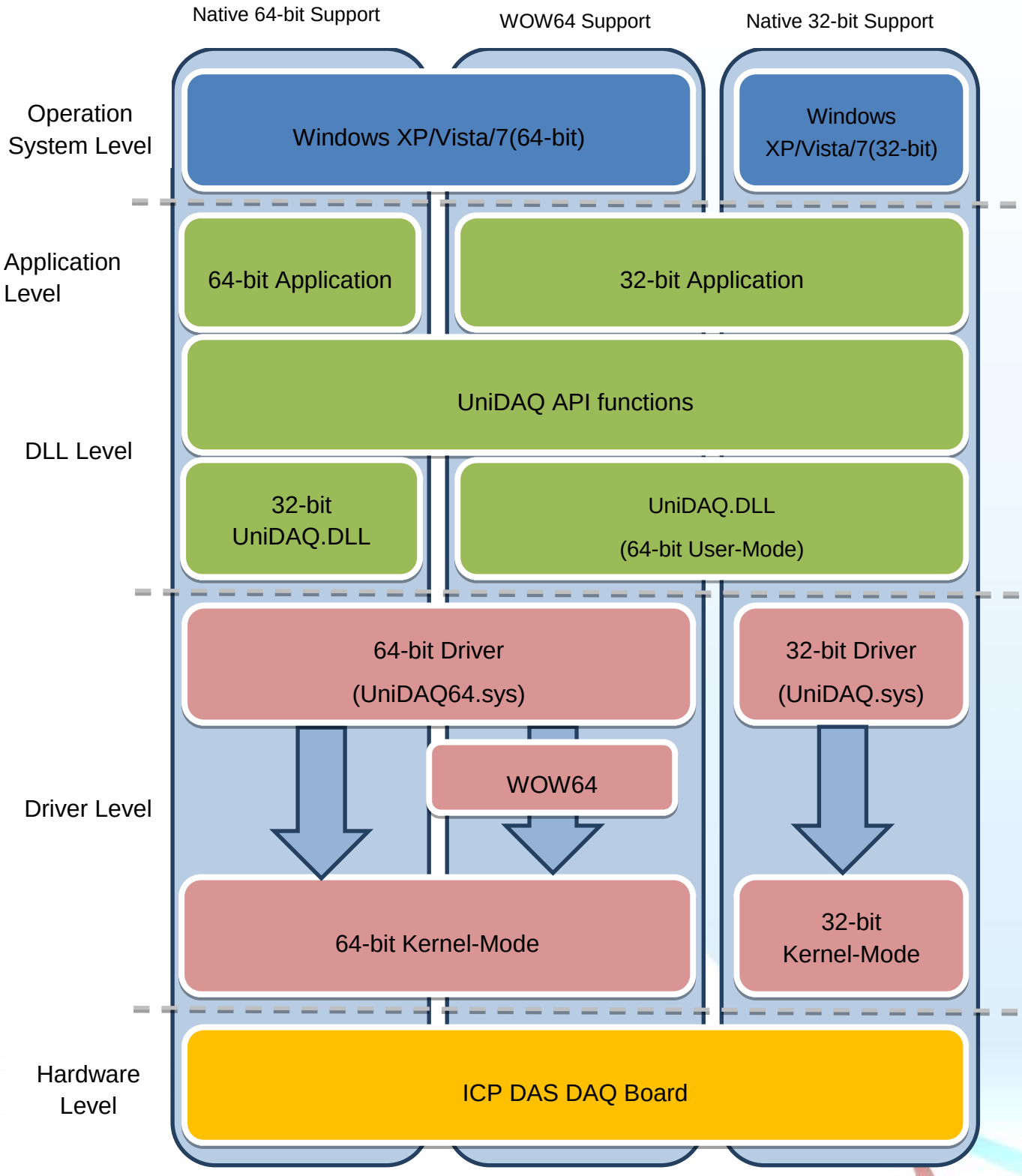
4. Click the Yes to All button to remove all UniDAQ.dll file



3. Tutorial

Gives the new user a walk-through introduction in creating simple application and provides Step-by-step procedures under many development environments.

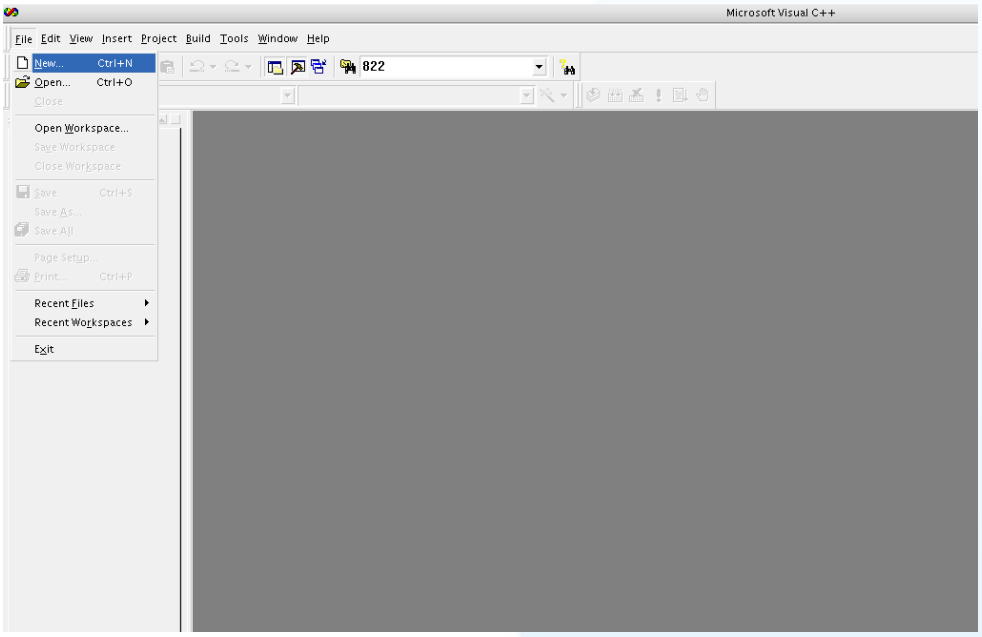
3.1. Application Structure



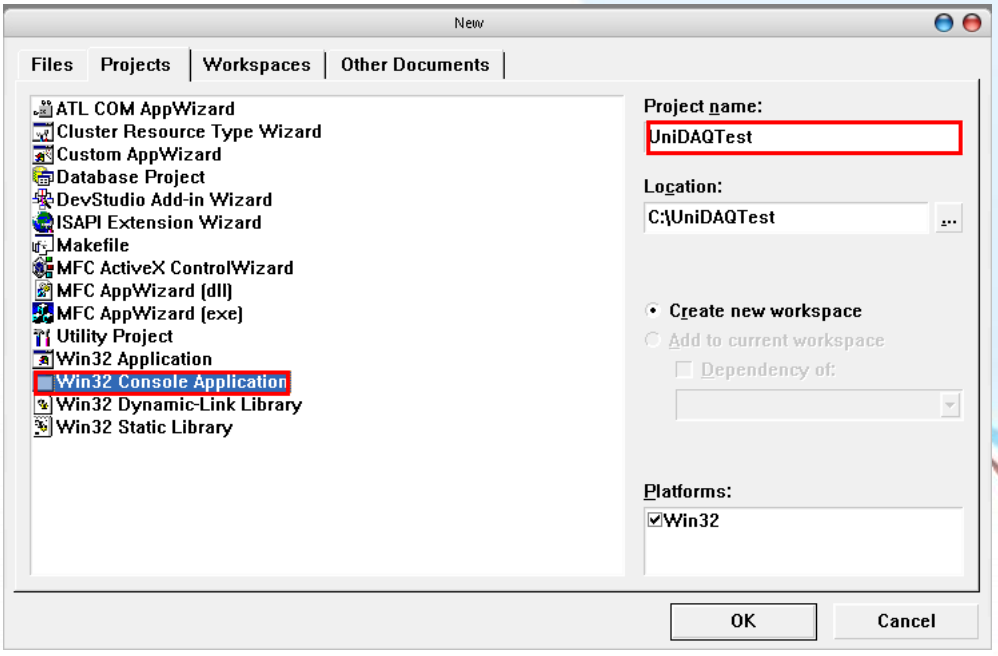
3.2. For Win32 Console

Step1: Write application with UniDAQ DLL

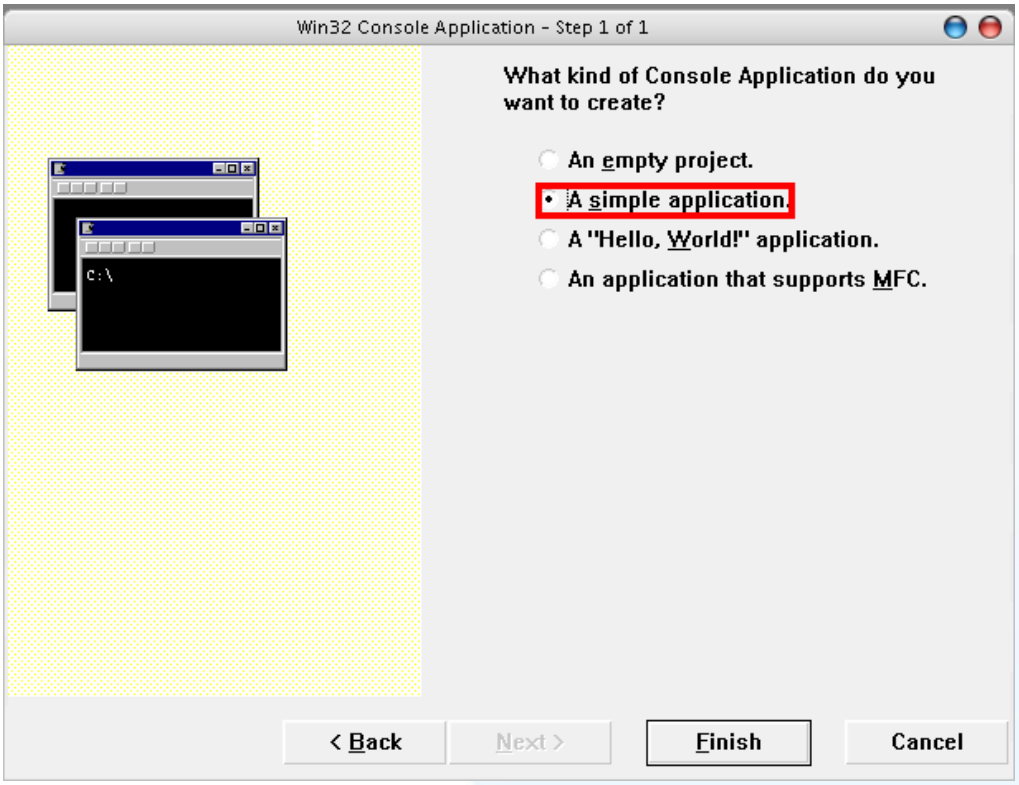
1. Go into all programs and click Microsoft Visual C++ 6.0 on the Microsoft Visual Studio 6.0.
2. Select File|New… from the main menu.



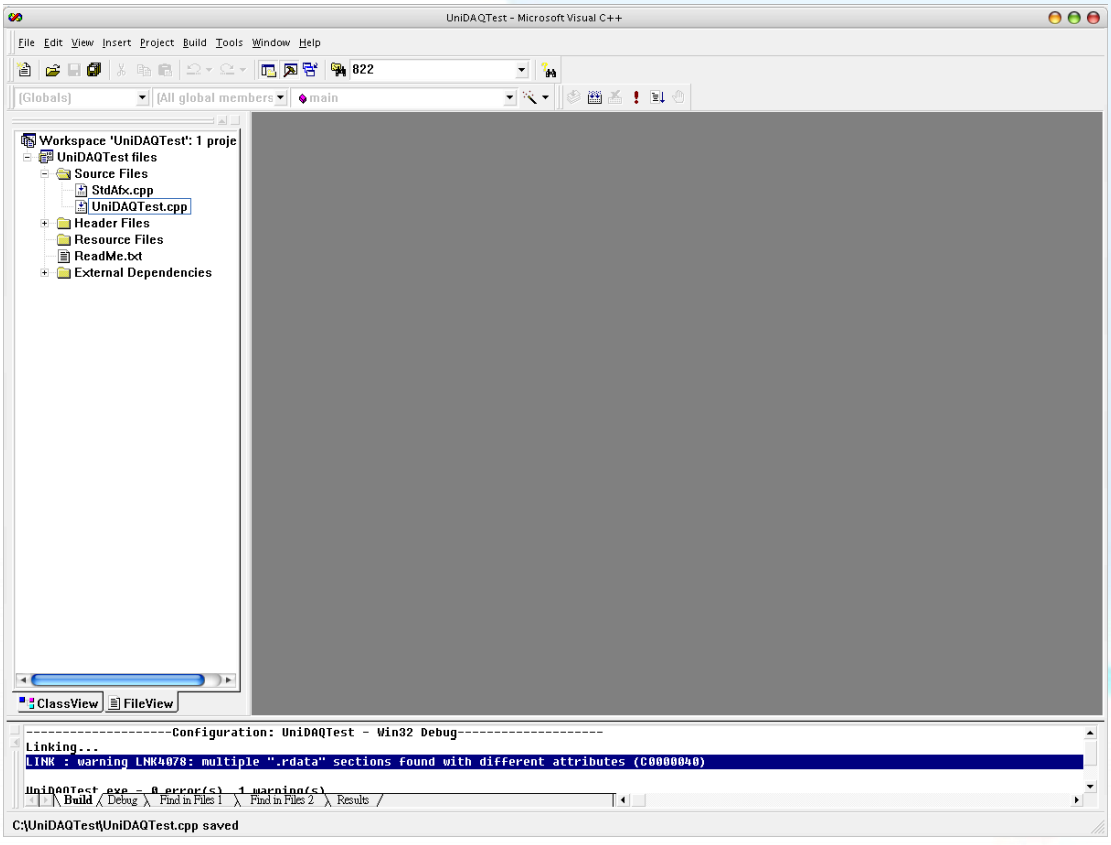
3. The following dialog box will appear. Click on the Win32 Console Application entry in the list and enter “UniDAQTest” in the Project name field. Then press the OK button.



4. Select the “A simple application” option, click the Finish button, and then some skeleton code will be generated.



5. Double click the UniDAQTest.cpp to open the codes writing windows.



6. Write codes for UniDAQTest.cpp as follows:

```
#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;

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

Step2: Test application

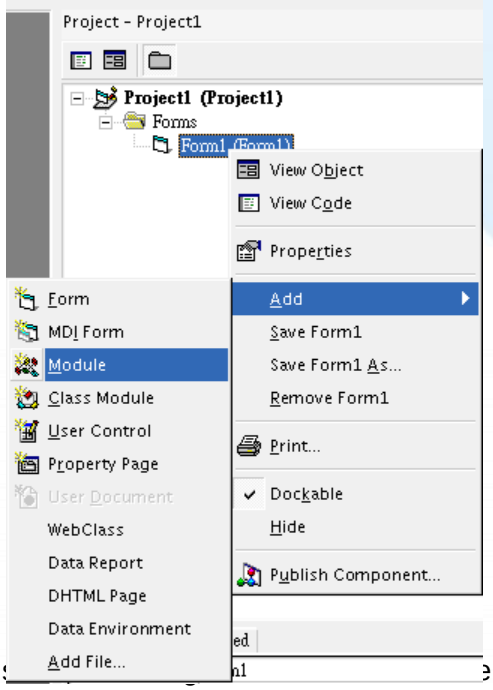
1. Click on Compile under the Build menu to compile your code.
2. Run it under a DOS Prompt.

3.3. For Visual Basic 6.0

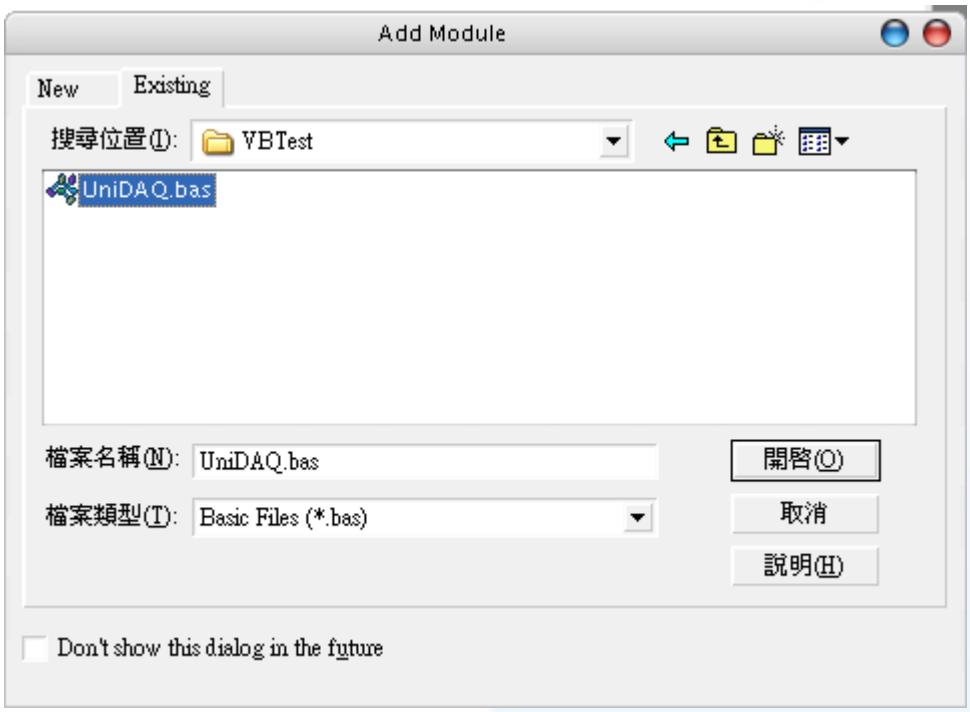
Step1: Write application with UniDAQ DLL

1. Go into all program and click Microsoft Visual Basic 6.0 on the Microsoft Visual Studio 6.0
2. Select the Standard EXE icon and press Open button. A new project will be created.

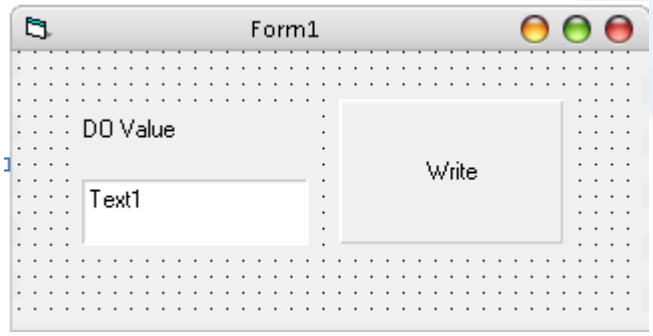
3. Click on the Project Explorer in the view menu.



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



5. Design form place a Label control on Form1and enter “DO Value” as its Caption field. Then place a TextBox control on Form1. Switch to the Property Window and enter txtDOVal. At last, place a CommandButton control on the form. Enter cmdWrite as its Name property, and enter Write as the Caption property. Your form should look similar to the one shown below:



6. Write code for the cmdWrite button as below:

```
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

‘//Initial the resource and get 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 DO
wRtn = Ixud_WriteDO(wBoardNo, wOutPortNo, Val(txtDOVal.Text))

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

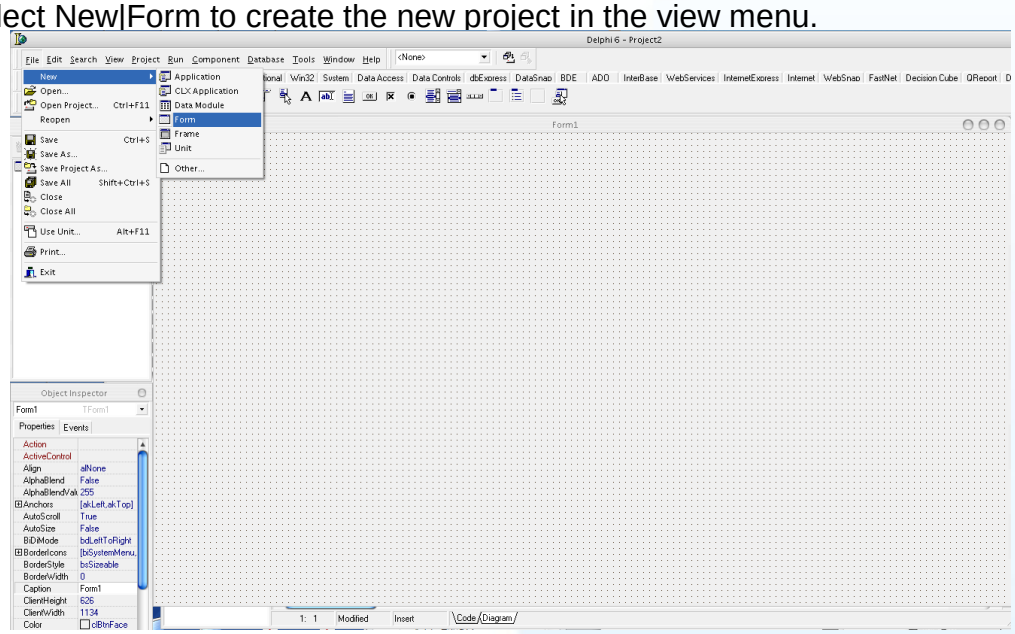
Step2: Test application

1. Press F5 to run program.
2. Then enter 255 on DO Value field.
3. Press the Write button to output DO Value 255.

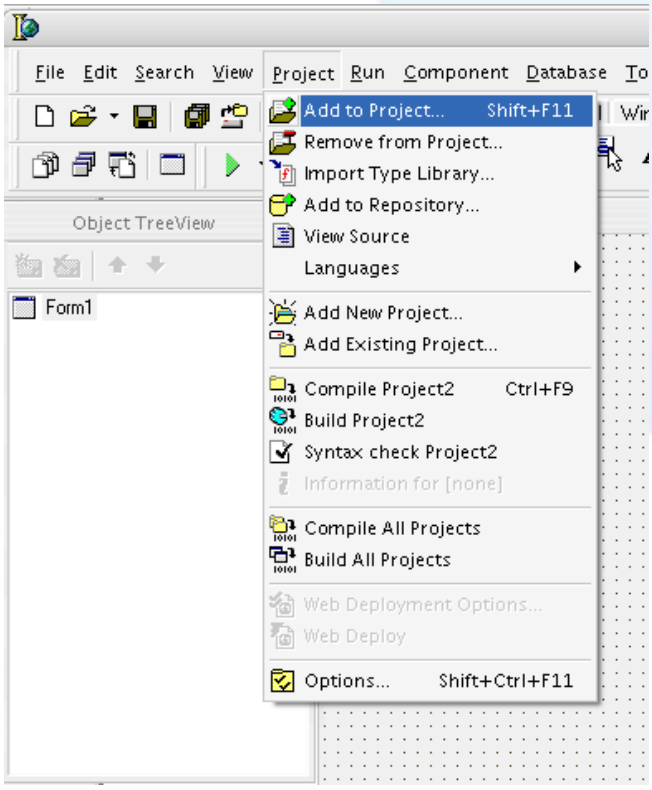
3.4. For Borland Delphi

Step1: Write application with UniDAQ DLL

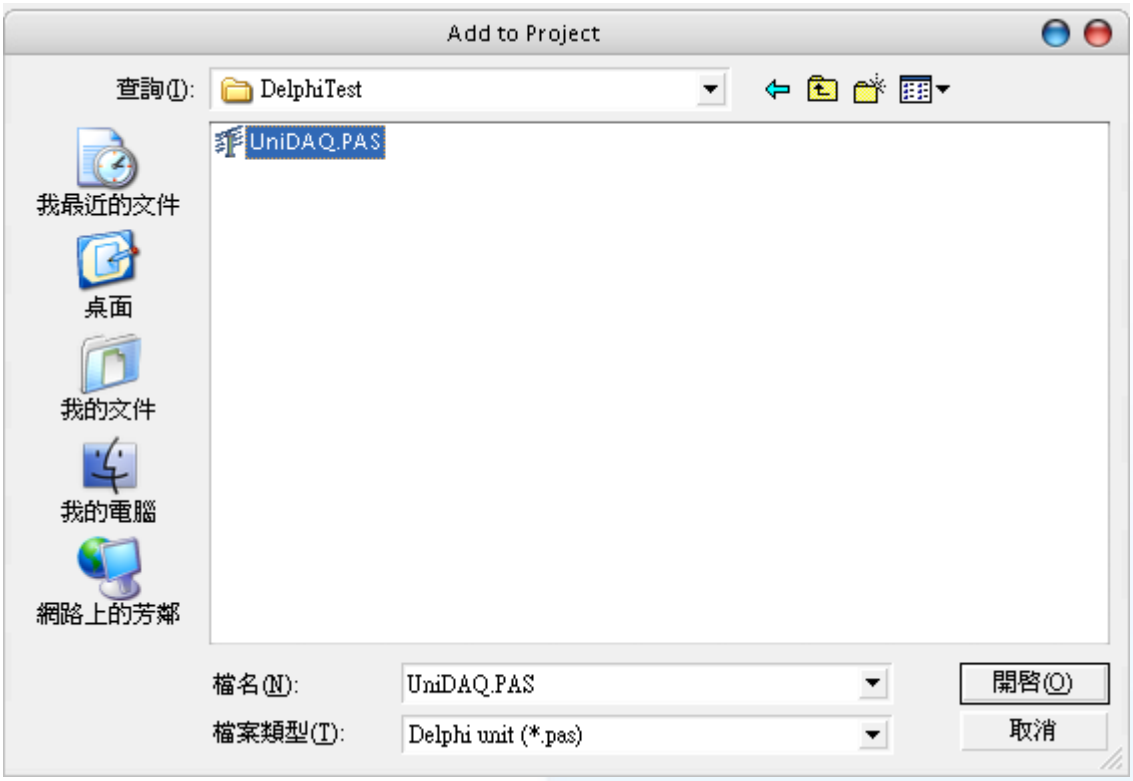
1. Go into the Start menu and click on the Delphi 6.0 icon in the Borland Delphi 6 folder.
2. Select New|Form to create the new project in the view menu.



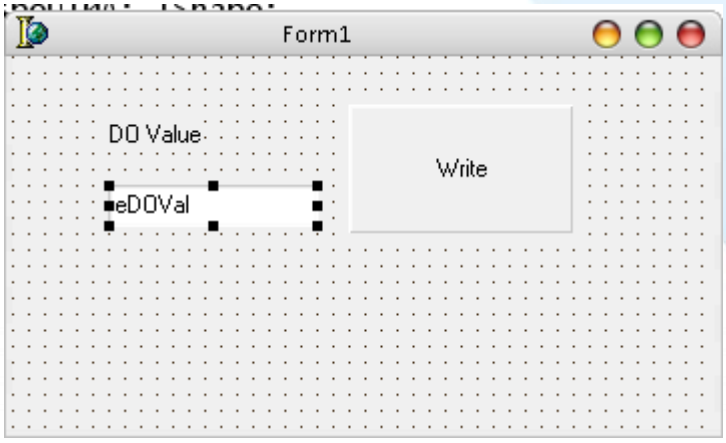
3. Click Project|Add to open the “Add to Project “ in the view menu.



4. Add the declaration file, UniDAQ.pas file by clicking the “Add to Project” in the Project menu.



5. Design form, place a Label control on Form1and enter “DO Value” as its Caption field. Then place an Edit control on Form1. Switch to the Property Window and enter eDOVal. At last, place a Button control on the form. Enter btnWrite as its Name property, and enter Write as the Caption property. Your form should look similar to the one shown below:



6. Write the code for the btnWrite button as below:

```
implementation
uses UniDAQ;

{$R *.dfm}

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

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

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

end;

end.
```

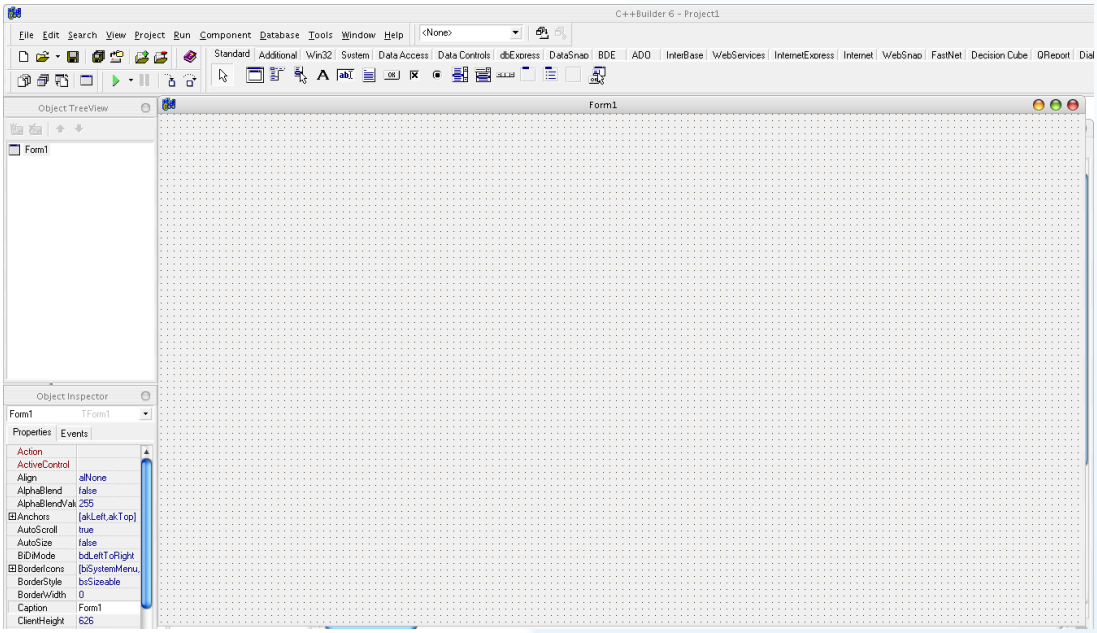
Step2: Test application

1. Press F9 to run program.
2. Then enter 255 on DO Value field.
3. Press the Write button to output DO Value 255.

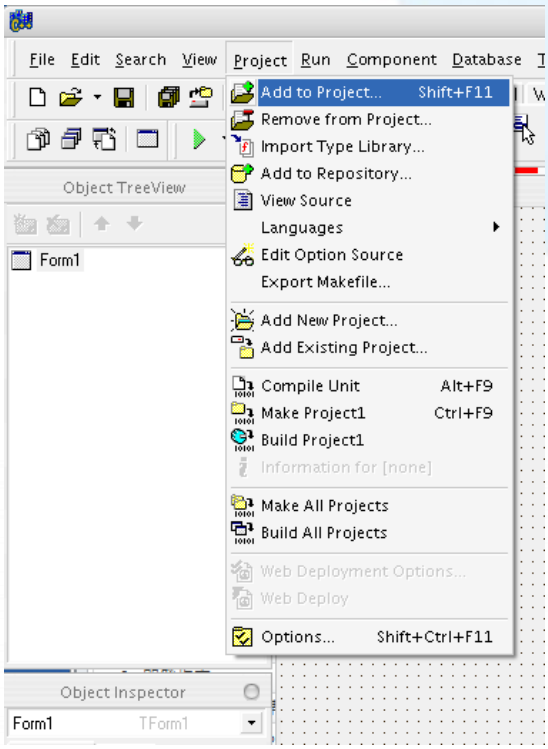
3.5. For Borland C++ Builder

Step1: Write application with UniDAQ DLL

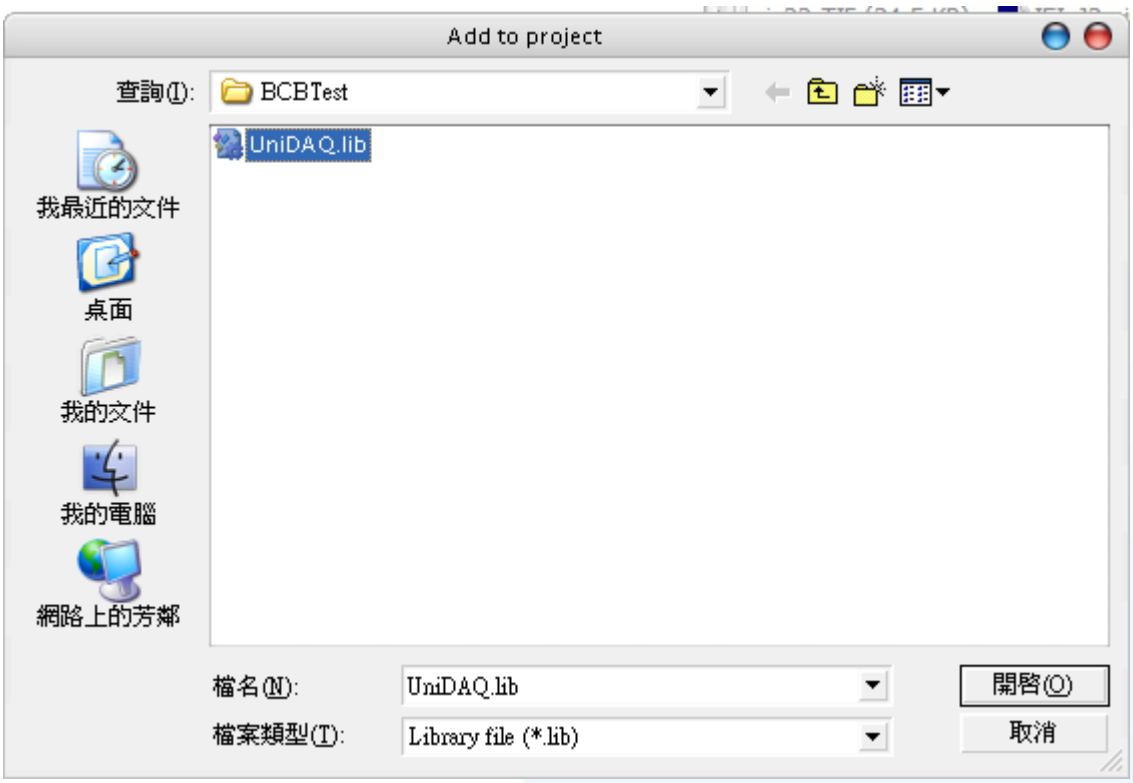
1. Go into the Start menu and click on the C++ Builder 6 icon in the Borland C+ Builder 6 folder.
2. Select New|Form to create the new project in the view menu.



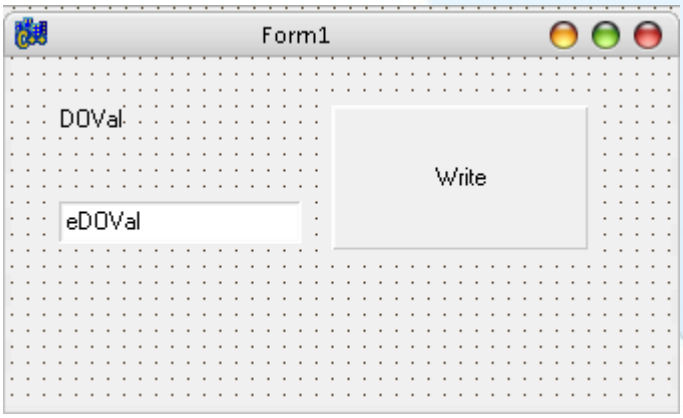
3. Click Project|Add to open the “Add to Project “ in the view menu.



4. Add the declaration file, UniDAQ.lib file by clicking the “Open” in the Project menu.



5. Design form, place a Label control on Form1and enter “DO Value” as its Caption field. Then place an Edit control on Form1. Switch to the Property Window and enter eDOVal. At last, place a Button control on the form. Enter btnWrite as its Name property, and enter Write as the Caption property. Your form should look similar to the one shown below:



6. Write the code for the btnWrite button as below:

```
#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;
//Initial the resource and get the total board number from driver
wRtn = Ixud_DriverInit(&wTotalBoard);
if ( wRtn != Ixud_NoErr )
{
    ShowMessage( "Driver Initial Err!!Error Code:" + IntToStr(wRtn)) ;
}
wOutPortNo=0;
wBoardNo=0;
wRtn=Ixud_WriteDO(wBoardNo,wOutPortNo,StrToInt(eDOVal->Text));

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

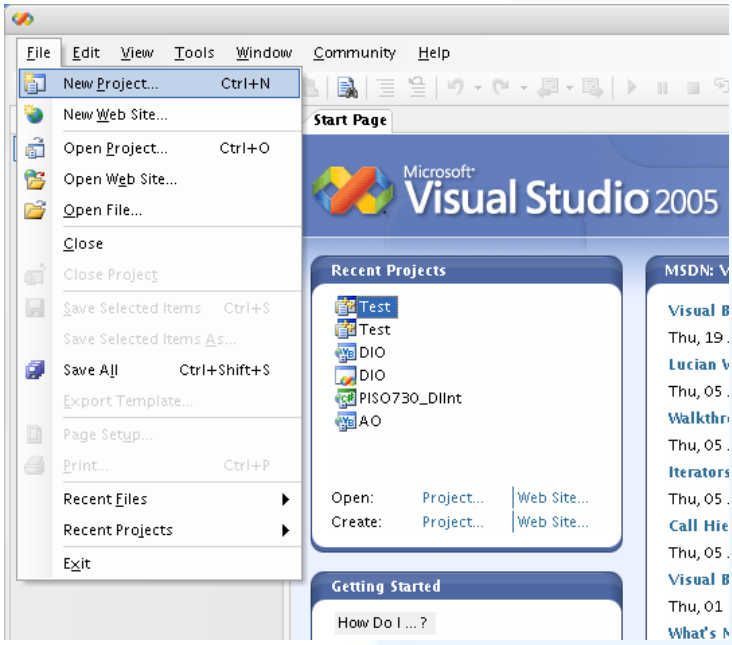
Step2: Test application

1. Press F9 to run program.
2. Then enter 255 on DO Value field.
3. Press the Write button to output DO Value 255.

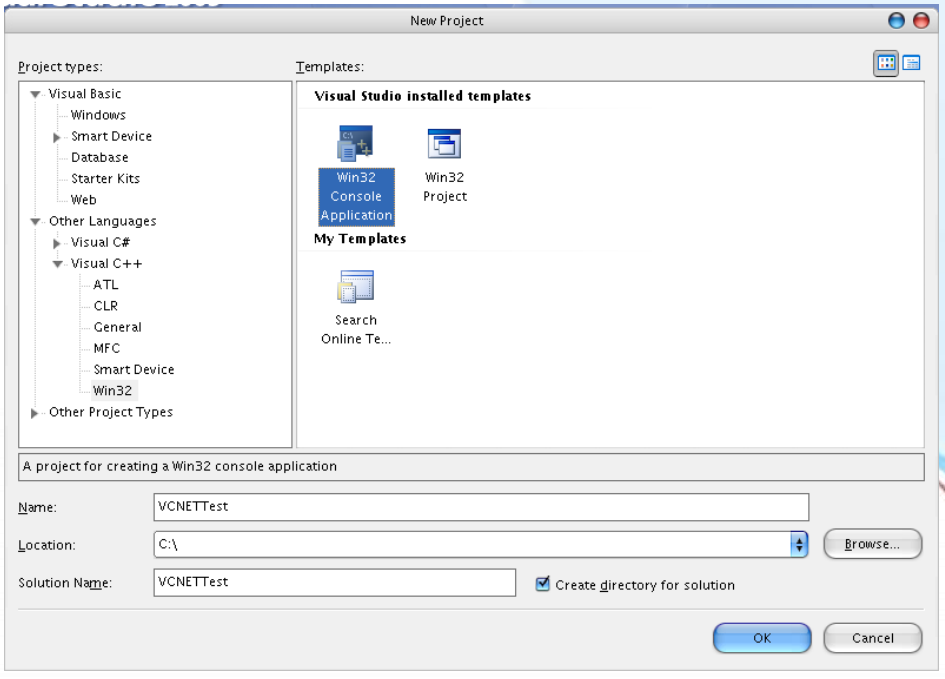
3.6. For Visual C++.NET

Step1: Write application with UniDAQ DLL

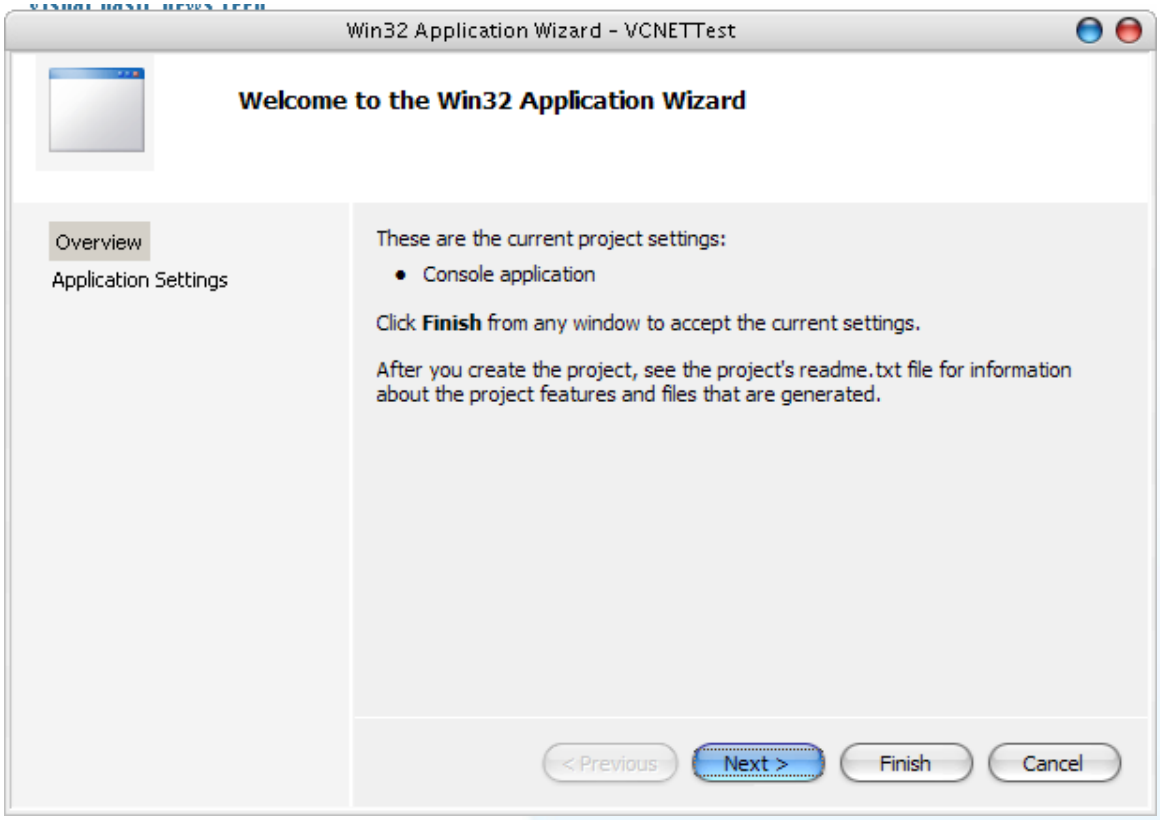
1. Go into all programs and click Microsoft Visual Studio 2005.
2. Select File|New Project… from the main menu



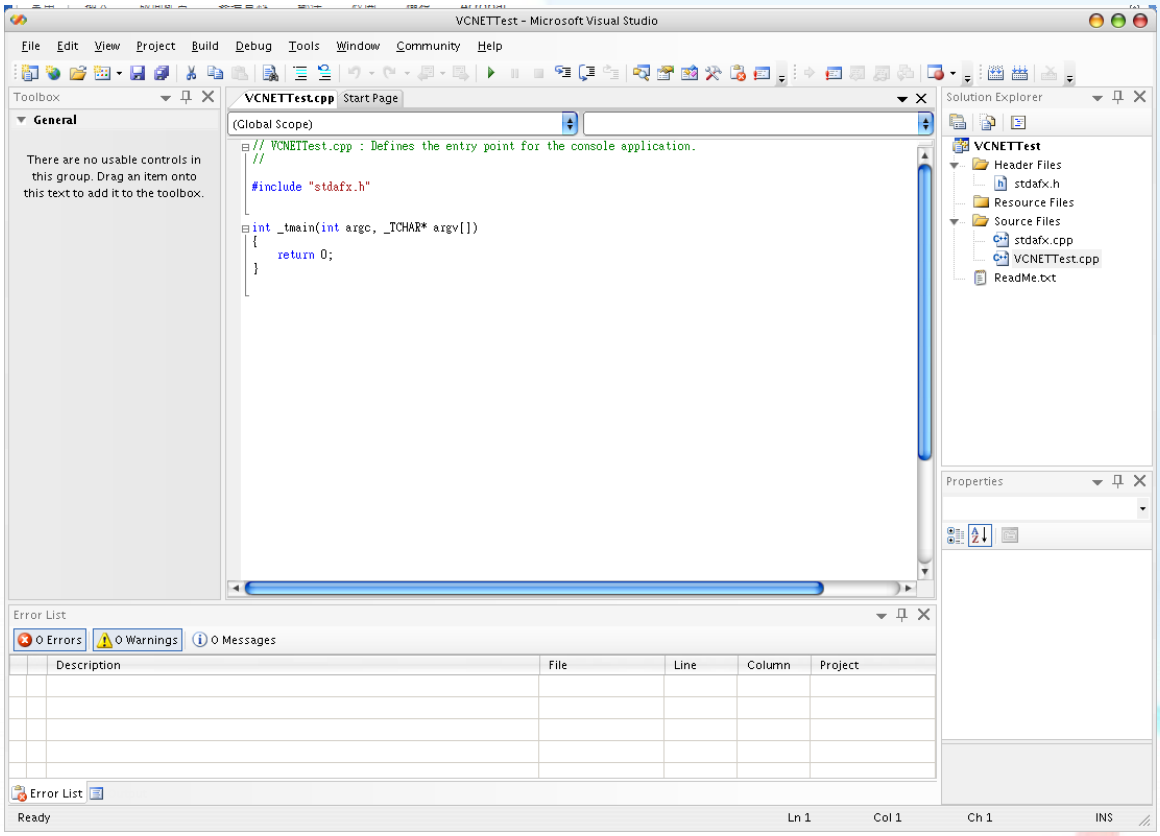
3. The following dialog box will appear. Click on the Win32 Console Application entry in the list and enter “VCNETTest” in the Project name field. Then press the OK button.



4. Click the Finish button, and then some skeleton code will be generated.



5. Double click the VCNETTest.cpp to open the codes writing windows.

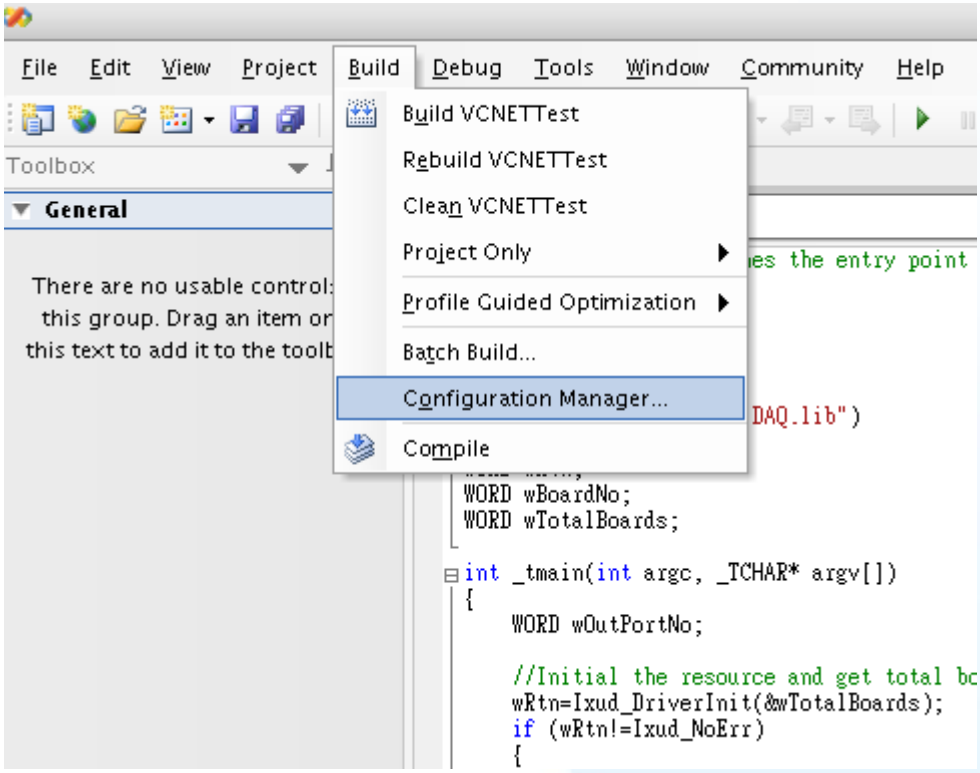


6. Write codes for VCNETTest.cpp as follows:

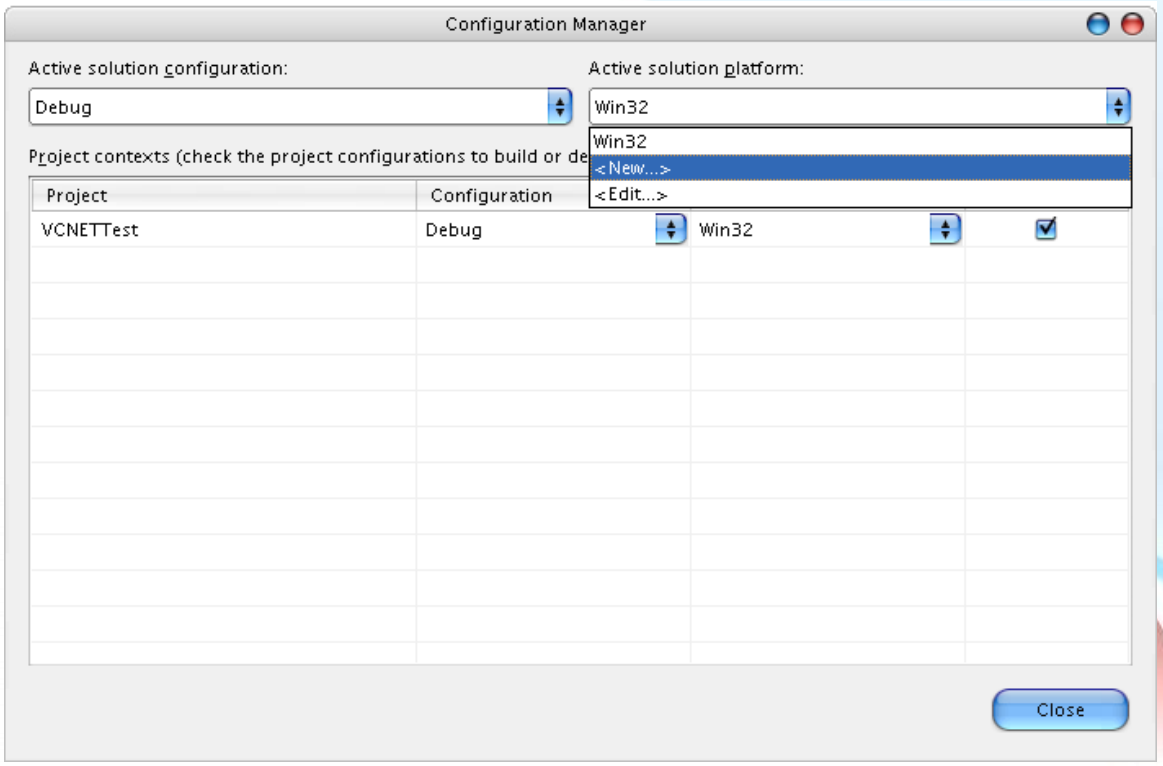
```
// VCNETTest.cpp : Defines the entry point for the console application.
//

#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;

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

Step2: Compiler application

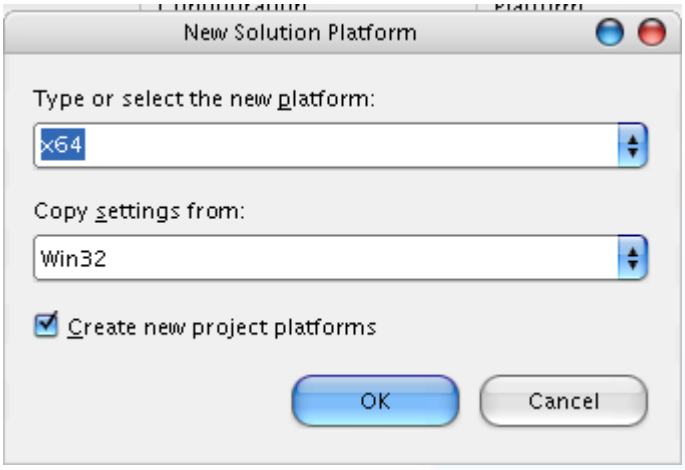
1. Click on “Configuration Manager…” under Build menu.



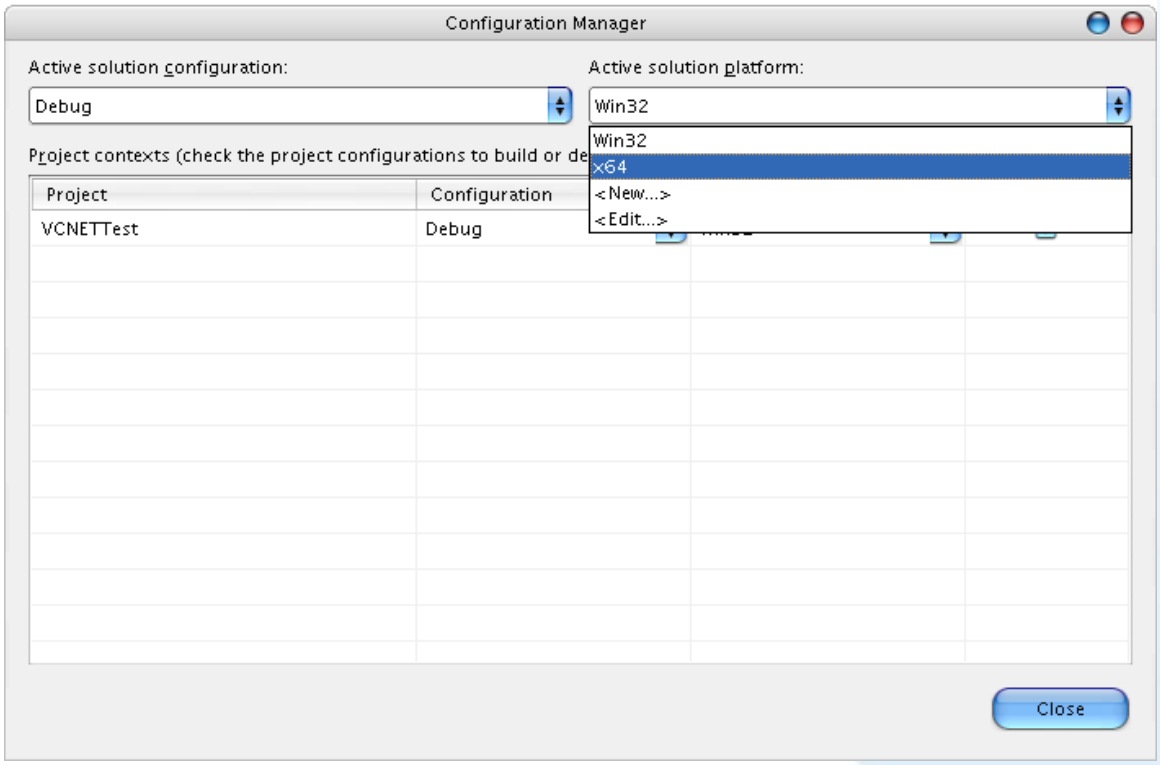
2. The following dialog box will appear. Select the <New…>。



3. Select the x64 on Type or select the new platform field and click on OK button to create x64 platform.



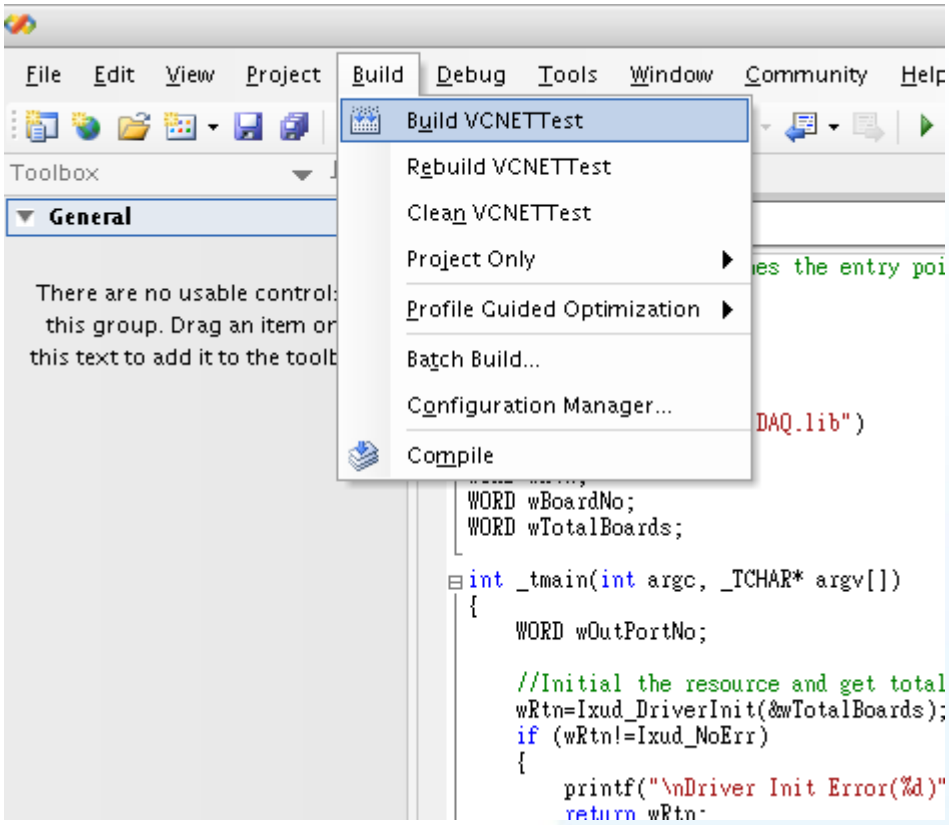
4. Check your application request .If application is 64-bit, it must select x64 platform. If application is 32-bit, it must select Win32 platform. Then click close button.



User must include the 64-bit UniDAQ.lib for 64-bit application

User must include the 32-bit UniDAQ.lib for 32-bit application

5. Click on “Build VCNETTest “ under build menu to compile your code.



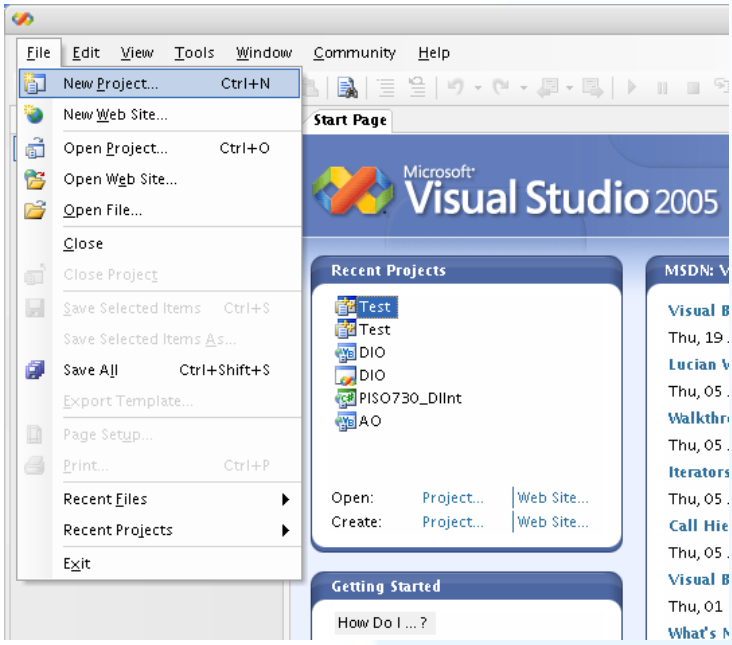
Step3: Test application

Run it under a DOS Prompt.

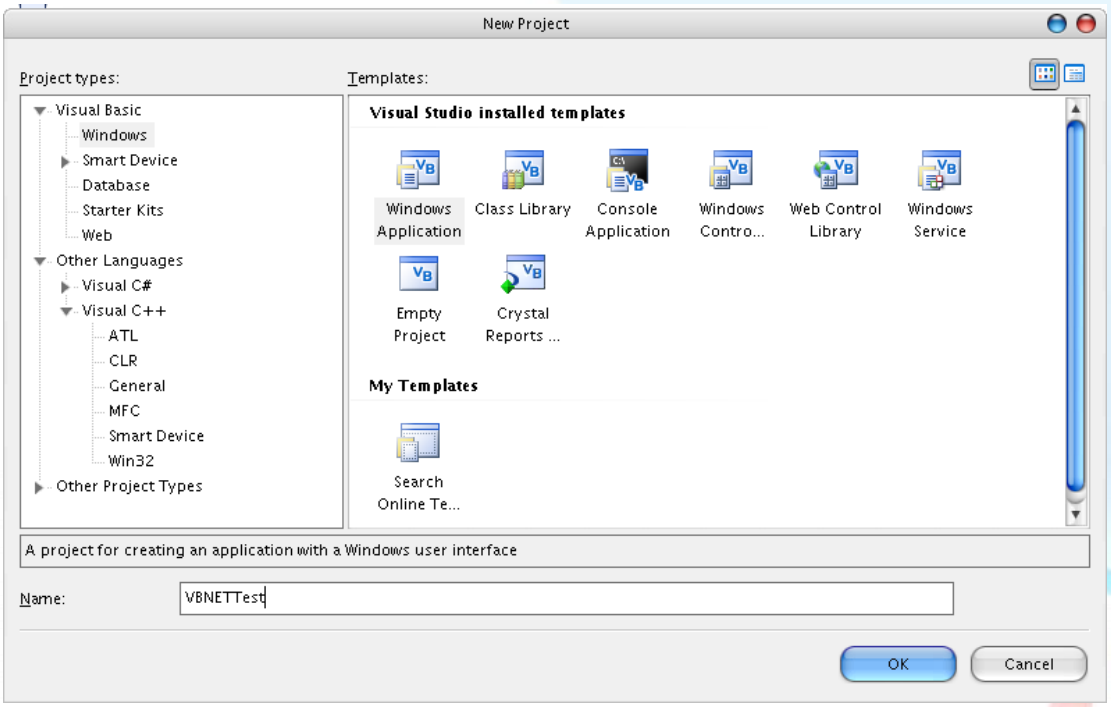
3.7. For Visual Basic.NET

Step1: Write application with UniDAQ DLL

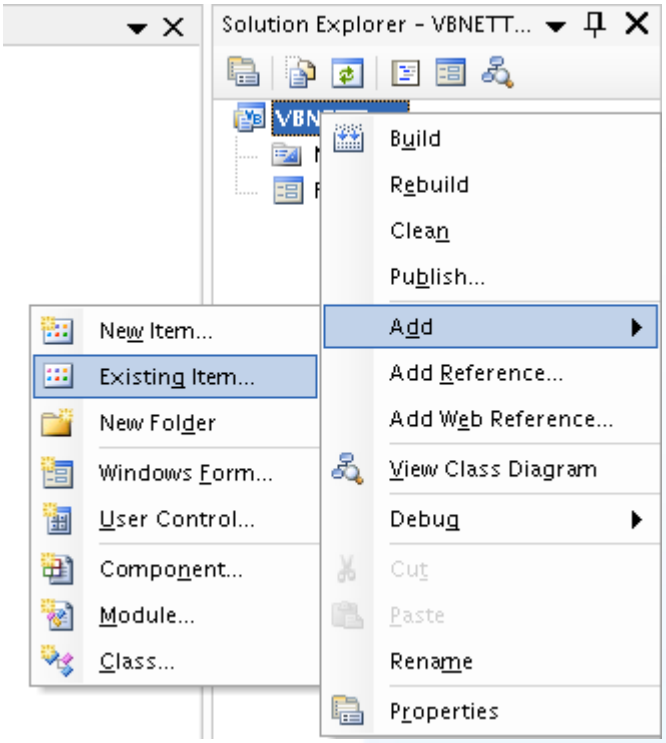
1. Go into all programs and click Microsoft Visual Studio 2005.
2. Select File|New Project… from the main menu



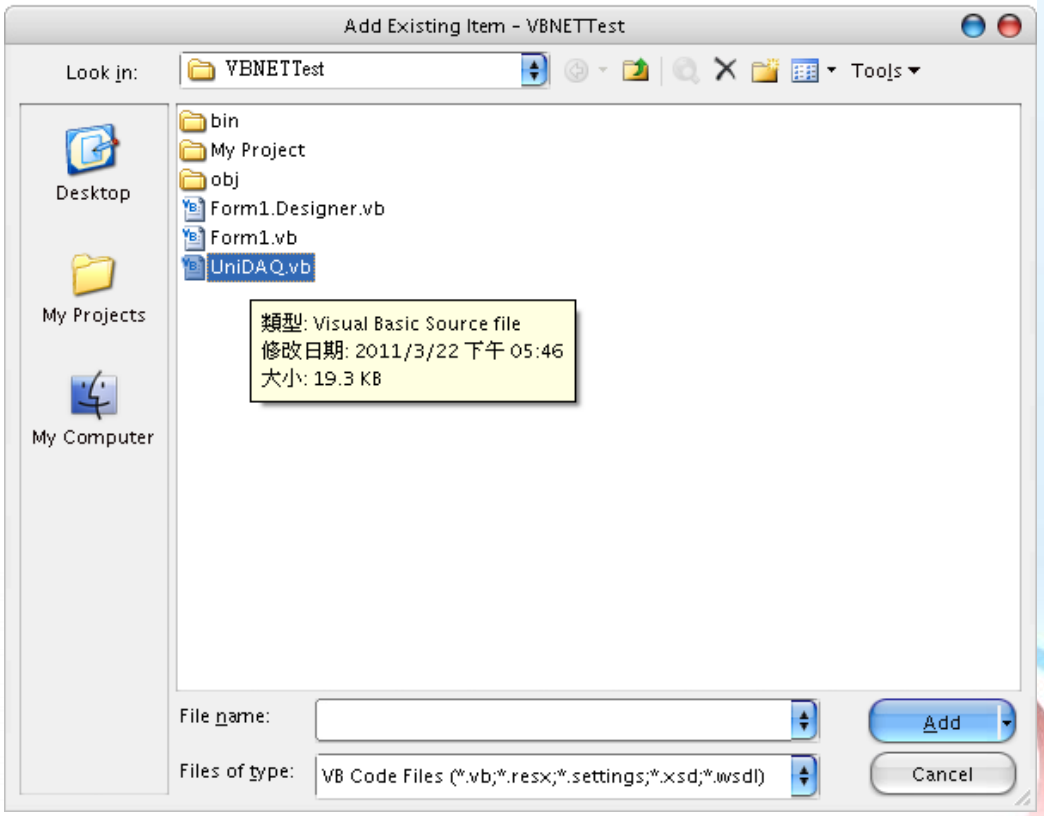
3. The following dialog box will appear. Click on the Windows Application entry in the list and enter “VBNETTest” in the Project name field. Then press the OK button.



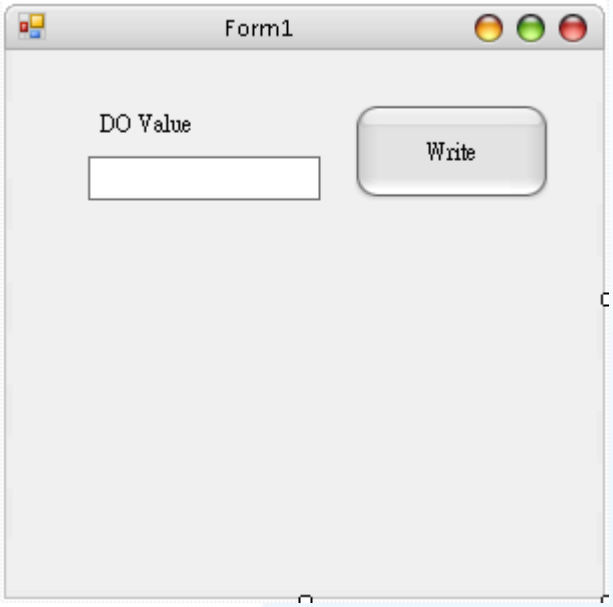
4. Select the Add|Existion item… on Solution Explorer.



5. Add the declaration file, UniDAQ.vb file by clicking the “Add to Existing item” in the Project menu.



6. Design form, place a Label control on Form1and enter “DO Value” as its Text field. Then place a TextBox control on Form1. Switch to the Property Window and enter txtDOVal. At last, place a Button control on the form. Enter btnWrite as its Name property, and enter Write as the Text property. Your form should look similar to the one shown below:

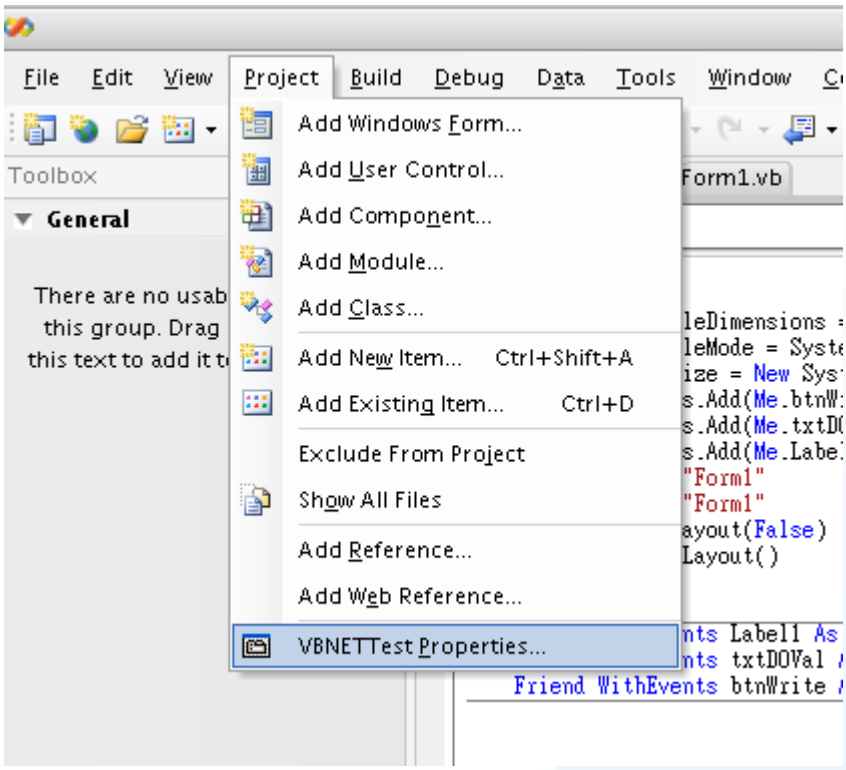


7. Write the code for the btnWrite button as below:

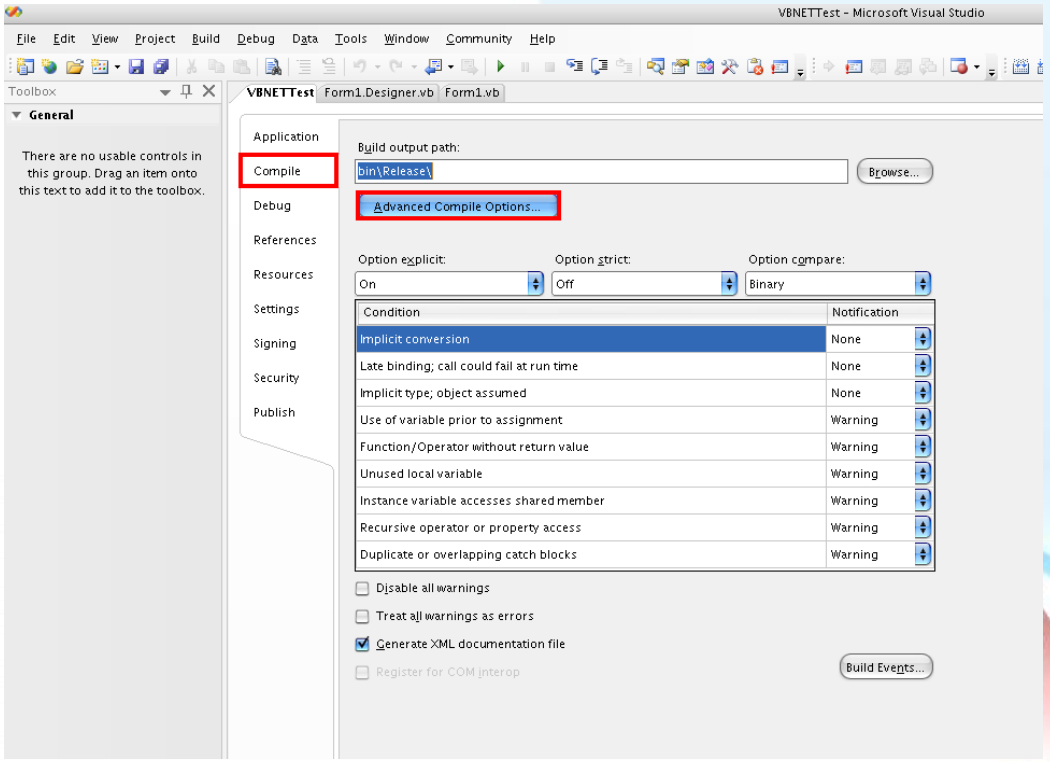
```
Private Sub btnWrite_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnWrite.Click
        Dim wTotalBoards As UInteger
        Dim wBoardNo As UInteger
        Dim wOutPortNo As UInteger
        Dim wRtn As UInteger

        '//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
```

Step2: Compiler application

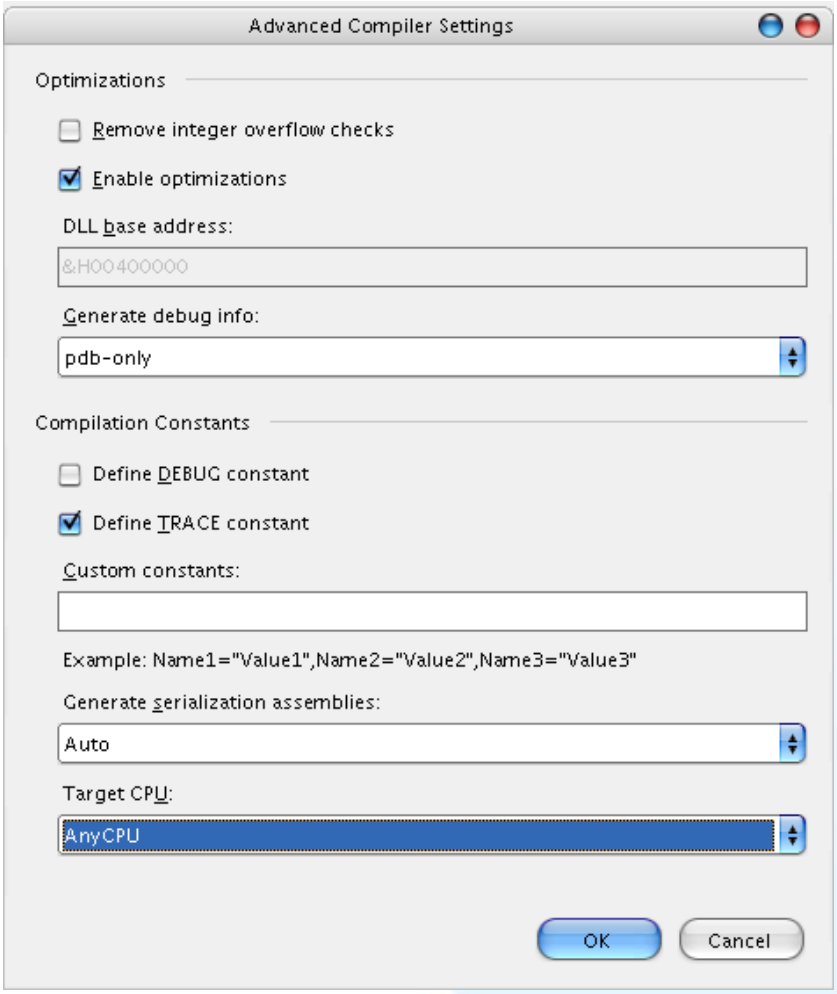
1. Click on “VBNETTest Properties” under Project menu.



2. The following dialog box will appear. Click on “Advanced Compiler Option” Button to enter “Advanced Compiler Setting”.



3. Select the “Any CPU” option under “Target CPU”.



Any CPU option– The assembly is compiled in a CPU-independent manner. If you are compiling an EXE, the EXE will run as an x64 processes when loaded by an x64 version of the .Net Framework on an x64 operating system. Otherwise the EXE will run as an x86 process.

x86 option– The assembly will always load as an x86 assembly, regardless of the operating system or .Net Framework version.

x64 option– The assembly will always load as an x64 assembly, regardless of the operating system or .Net Framework version.

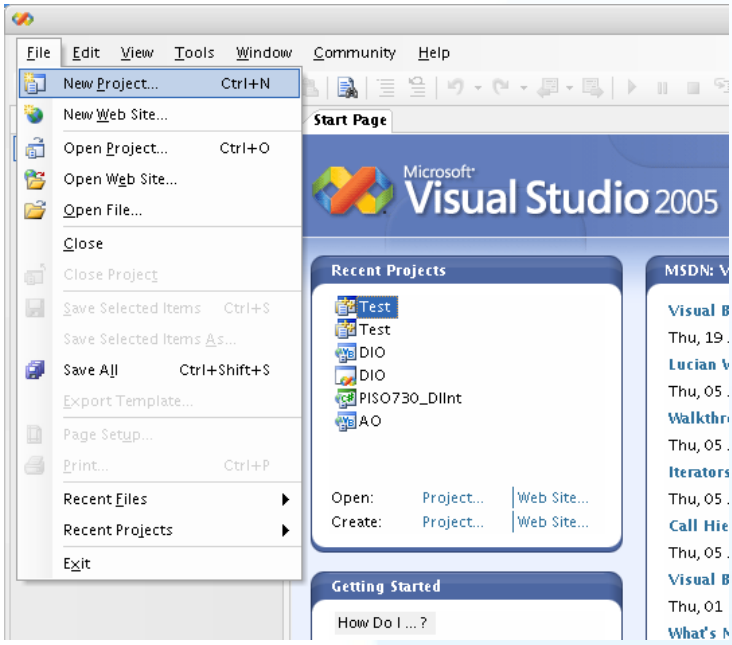
Step3: Test application

1. Press F5 to run program.
2. Then enter 255 on DO Value field.
3. Press the Write button to output DO Value 255.

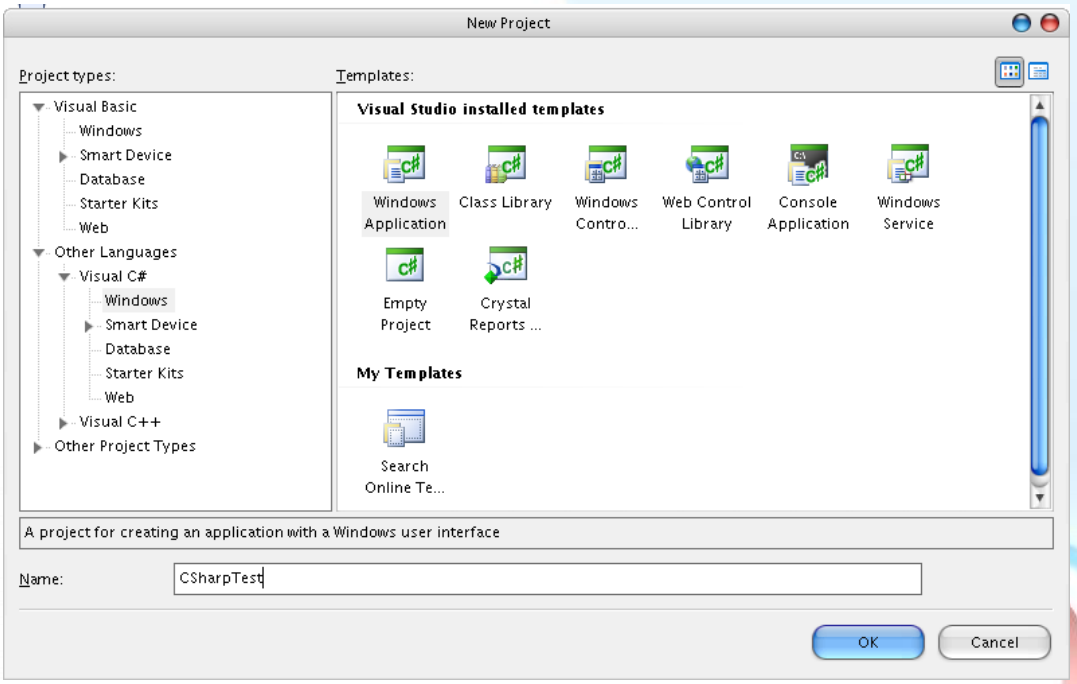
3.8. For Visual C#.NET

Step1: Write application with UniDAQ DLL

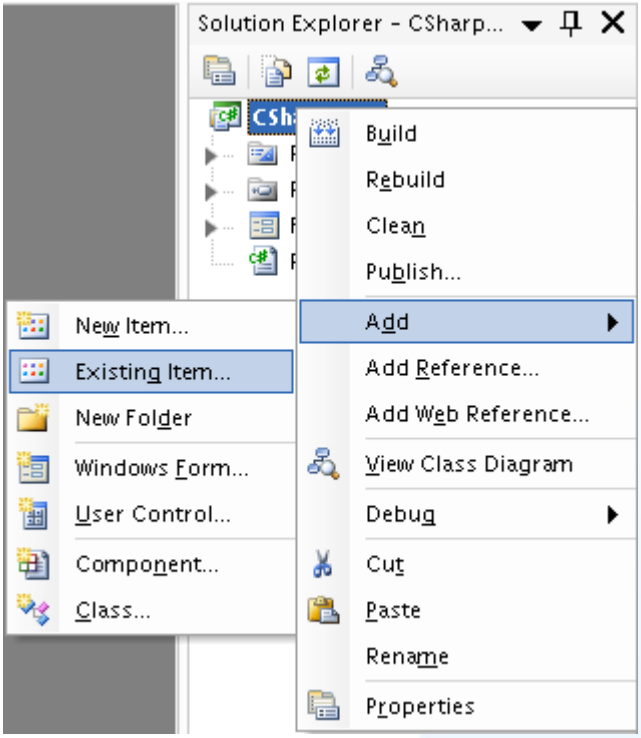
1. Go into all programs and click Microsoft Visual Studio 2005.
2. Select File|New Project… from the main menu



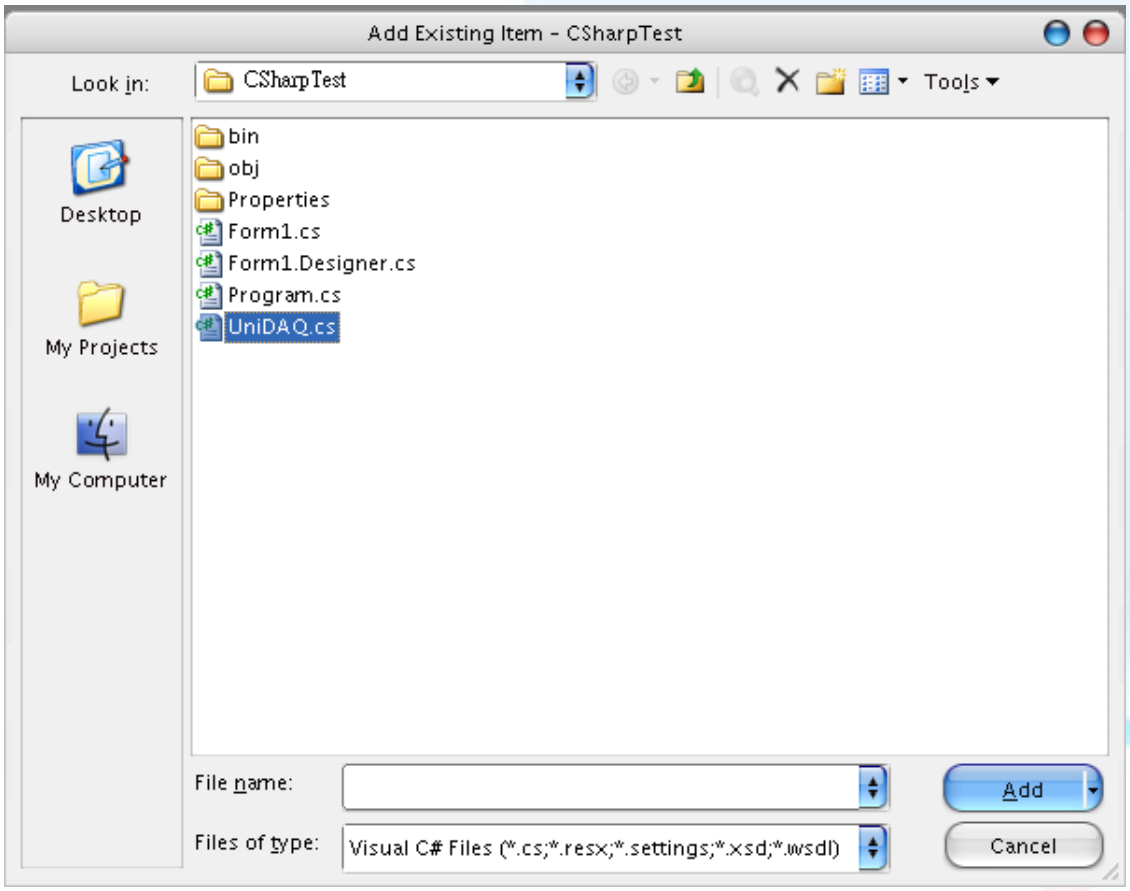
3. The following dialog box will appear. Click on the Windows Application entry in the list and enter “CSharpTest” in the Project name field. Then press the OK button.



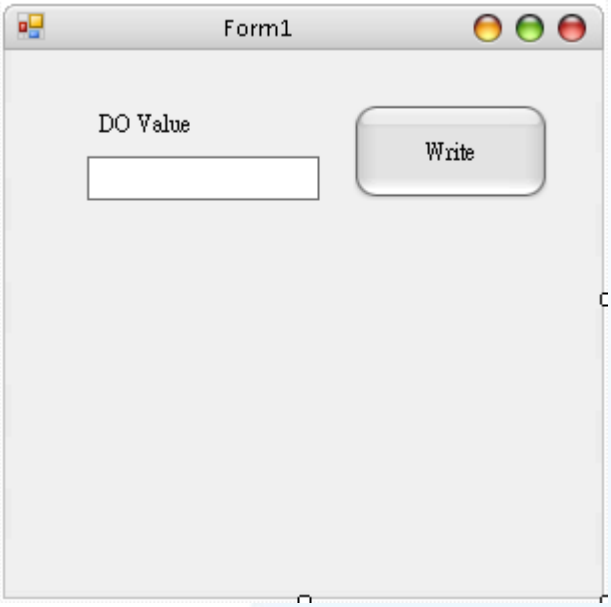
4. Select the Add|Existion item… on Solution Explorer.



5. Add the declaration file, UniDAQ.vb file by clicking the “Add to Existing item” in the Project menu。



6. Design form, place a Label control on Form1and enter “DO Value” as its Text field. Then place a TextBox control on Form1. Switch to the Property Window and enter txtDOVal. At last, place a Button control on the form. Enter btnWrite as its Name property, and enter Write as the Text property. Your form should look similar to the one shown below:

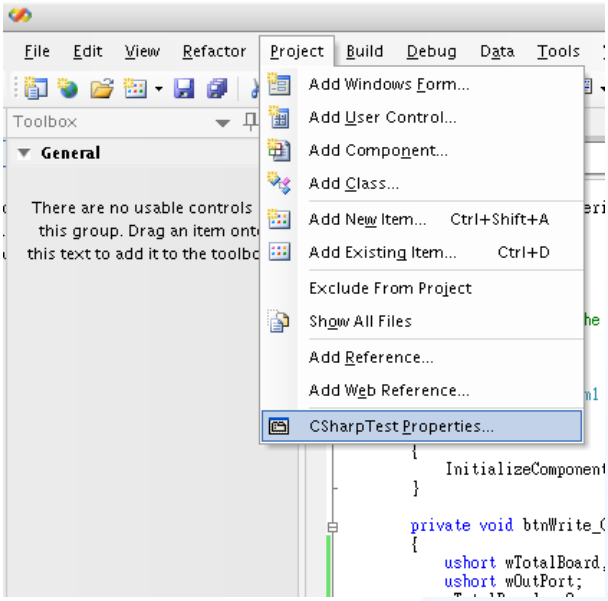


7. Write the code for the Form.cs as below:

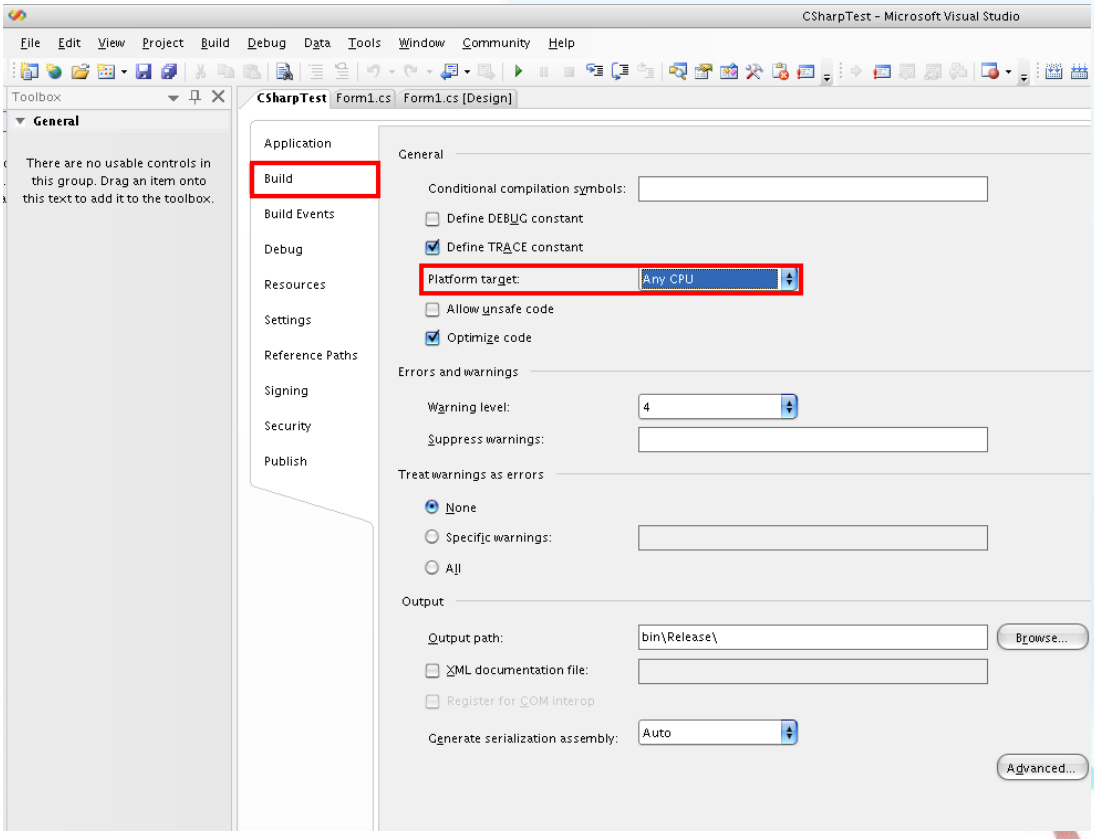
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using UniDAQ_Ns; //Include the UniDAQ namespace

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;
            //Initial the resource and get total board number form Driver
            wRtn = UniDAQ.Ixud_DriverInit(ref wTotalBoard);
            if (wRtn != UniDAQ.Ixud_NoErr)
            {
                MessageBox.Show("Driver Inital Error!!Error Code:" + wRtn.ToString());
                Close();
                return;
            }

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

        }
    }
}
```

Step2: Compiler application

1. Click on “CSharpTest Properties” under Project menu.



2. The following dialog box will appear. Select the “Any CPU” option under “Platform Target”.



Any CPU option– The assembly is compiled in a CPU-independent manner. If you are compiling an EXE, the EXE will run as an x64 processes when loaded by an x64 version of the .Net Framework on an x64 operating system. Otherwise the EXE will run as an x86 process.

x86 option– The assembly will always load as an x86 assembly, regardless of the operating system or .Net Framework version.

x64 option– The assembly will always load as an x64 assembly, regardless of the operating system or .Net Framework version.

Step3: Test application

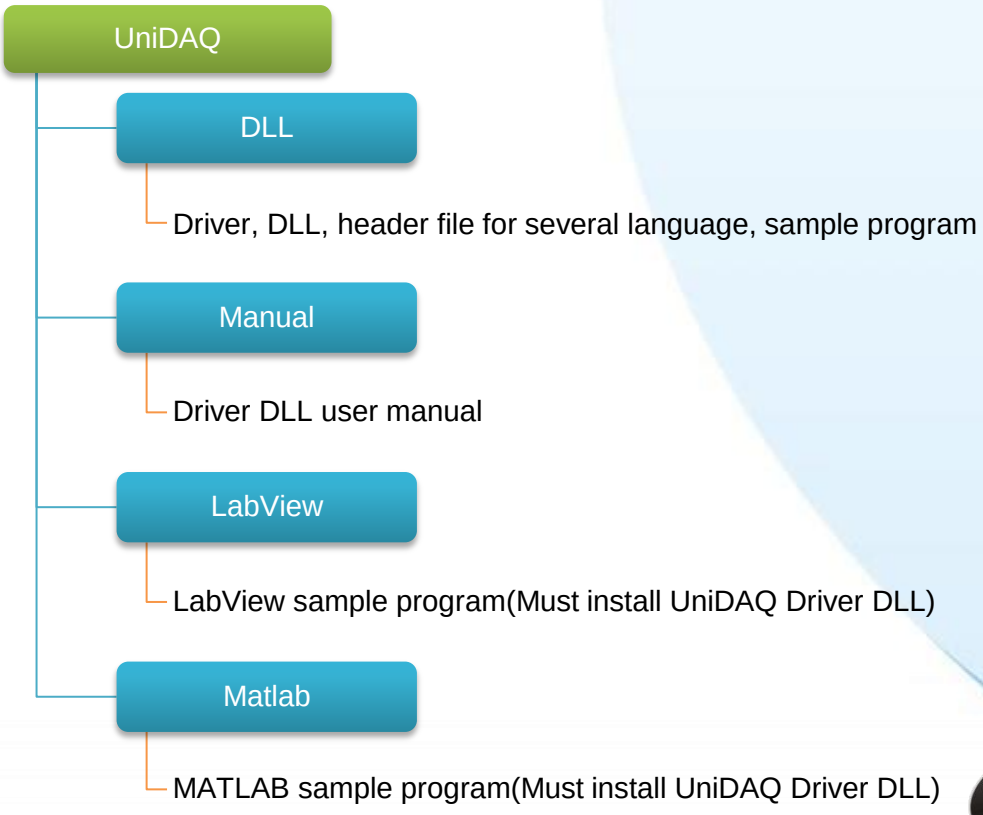
1. Press F5 to run program.
2. Then enter 255 on DO Value field.
3. Press the Write button to output DO Value 255.

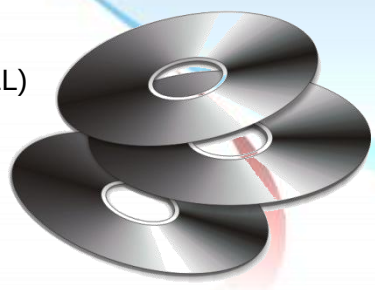
3.9. Sample program and Document

UniDAQ Software, Document location:

	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/

UniDAQ folder directory tree





4. Function Overview

Introduces the hardware functions that can be programmed by using the ICP DAS UniDAQ Driver DLL

4.1. Introduction

ICP DAS UniDAQ Driver DLL contains a set of functions. It can be used in various application programs for ICP DAS DAQ Card. The API functions supports many development environments and programming languages, including Microsoft Visual C++，Visual Basic，Borland Delphi，Borland C Builder++，Microsoft Visual C++.NET，Microsoft Visual C#.NET，Microsoft Visual VB.NET.

It provides the following functions:

1. Driver function: initializes and release device resource and configures the device and get device information.
2. Digital I/O: Controls digital I/O for the specified channel.
3. Interrupt Event function: Support the DAQ card that contains the interrupt functions, send notification when the analog input or digital input operation is completed.
4. Analog Output: Converts DAC to output the voltage or current.
5. Analog Input: Converts single and multiple-channel to acquire the voltage, current, pressure, strain and so on.
6. Timer/counter: performs event-counting, frequency measurement and pulse output.
7. Memory I/O: Controls memory I/O.

Supported programming language：

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

The driver provides a set of function calls for applications as shown in the following tables.

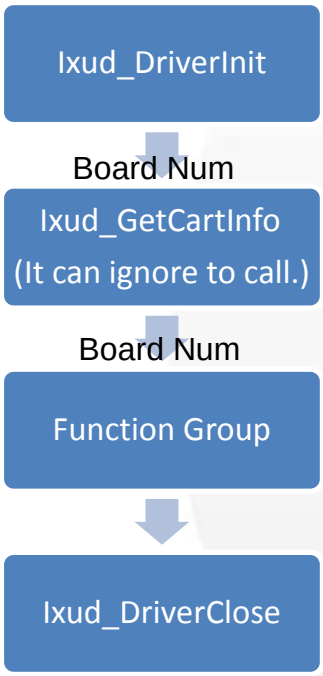
Driver functions	Digital I/O	Interrupt Event	Analog Input
lxud_GetDllVersion	lxud_SetDIOModes32	lxud_SetEventCallback	lxud_ConfigAI
lxud_OptionMode	lxud_SetDIOMode	lxud_RemoveEventCallback	lxud_ConfigAIEx
lxud_DriverInit	lxud_ReadDI	lxud_InstallIrq	lxud_ClearAIBuffer
lxud_DriverClose	lxud_WriteDO	lxud_RemoveIrq	lxud_GetBufferStatus
lxud_SearchCard	lxud_ReadDIBit		lxud_ReadAI
lxud_GetBoardNoByCardID	lxud_WriteDIBit		lxud_ReadAIH
lxud_GetCardInfo	lxud_ReadDI32		lxud_PollingAI
lxud_ReadPort	lxud_WriteDO32		lxud_PollingAIH
lxud_WritePort	lxud_SoftwareReadbackDO		lxud_PollingAIScan
lxud_ReadPort32	lxud_StartDI		lxud_PollingAIScanH
lxud_WritePort32	lxud_StopDI		lxud_StartAI
lxud_ReadPhyMemory	lxud_GetDIBufferH		lxud_StartAIScan
lxud_WritePhyMemory	lxud_StartDO		lxud_StartExtAI
	lxud_StopDO		lxud_StartExtAIScan
			lxud_StartExtAnalogTrigger
			lxud_GetAIBuffer
			lxud_GetAIBufferH
			lxud_StopAI

Analog Output	Timer/Counter	Memory I/O
lxud_ConfigAO	lxud_ReadCounter	lxud_ReadMemory
lxud_WriteAOVoltage	lxud_SetCounter	lxud_WriteMemory
lxud_WriteAOVoltageH	lxud_DisableCounter	lxud_ReadMemory32
lxud_WriteAOCurrent	lxud_SetFCChannelMode	lxud_WriteMemory32
lxud_WriteAOCurrentH	lxud_ReadFrequency	
lxud_StartAOVoltage		
lxud_StartAOVoltageH		
lxud_StopAO		

4.2. Driver functions

The following figure describes the common call flow of the ICP DAS UniDAQ Driver DLL

Call Flow



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

The Board Num specifies the DAQ board that user want to perform the I/O operations. The Board Num depends on Bus num and Device number of PCI Configuration space. If the smaller Bus number and Device number, the Board Number is smaller.

For the entry of the device”0 PCI-826”, Board Num is equal to 0, user can use the Board Num to perform the configuration directly.

Ixud_DriverInit and Ixud_DriverClose function

Ixud_DriverInit allocate the resource for all board and get the board number. User must call this function on driver entry.Ixud_DriverClose release the resource for board. User must call on driver exit.

Ixud_GetCardInfo function

This function provides the board name and hardware information. User can ignore to call it.

4.3. Digital I/O

The Digital Input/Output function group performs digital input and output operations. The digital input/output lines on each data acquisition board are grouped into logical units called ports. Each port has eight, sixteen or thirty-two lines or bits.

The digital I/O port of some data acquisition board (e.g., PIO-D24U/D56U/D48U/D96U/144U/168U) can be configured for input or output. User can use the Ixud_SetDIOModes32 or Ixud_SetDIOMode function to configure the specified port for input or output.

4.3.1. Digital Input

The digital input functions perform digital input operations.ICP DAS UniDAQ Driver DLL supports digital input with software triggering, and digital input with interrupt.

- **Software triggering**

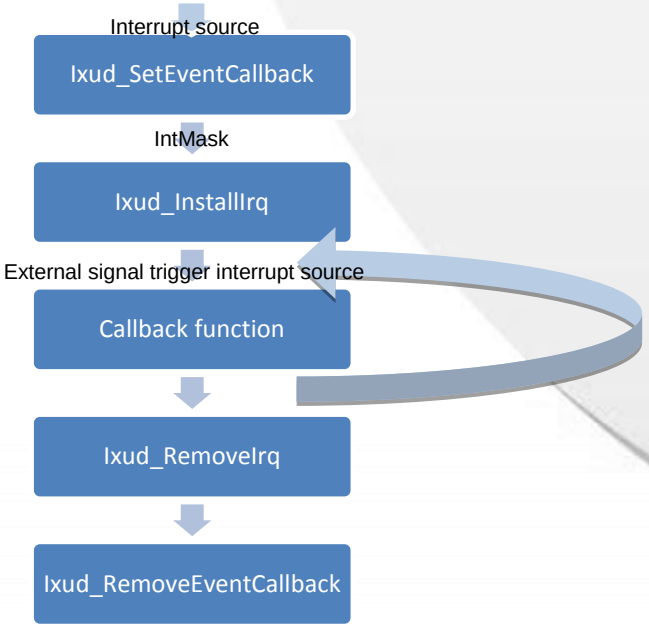
User just calls Ixud_ReadDI function to read the state from a port. UniDAQ also provide the Ixud_ReadDIBit function to read a byte value from the specified bit.

Call Flow

- **Interrupt Triggering**

Allow user to monitor the status of the digital input line. When the state changes from low to high or/and high to low, it would acknowledge the driver through a hardware interrupt. You don't have to poll the digital input line periodically.

Call Flow

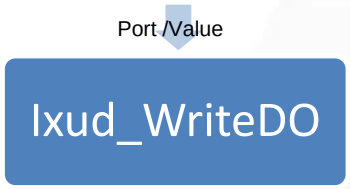


4.3.2. Digital Output

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**

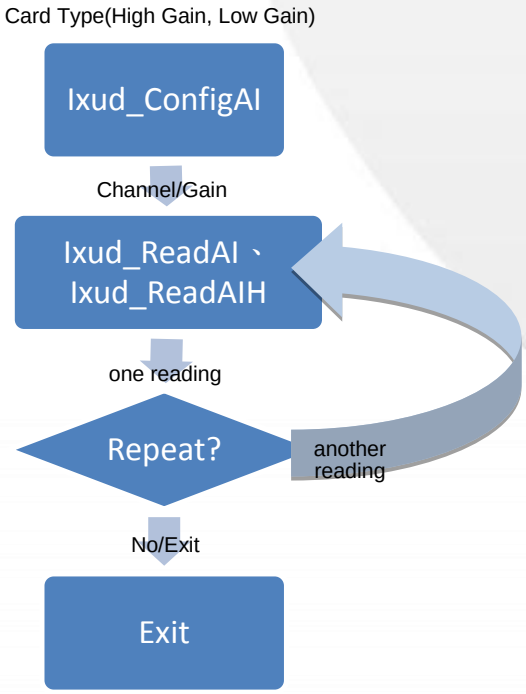
These functions trigger the data conversion by software. The UniDAQ provides three kinds of functions: one is for single point reading; the second is for multiple points reading; and the latest one is for multiple channel reading.

> 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.

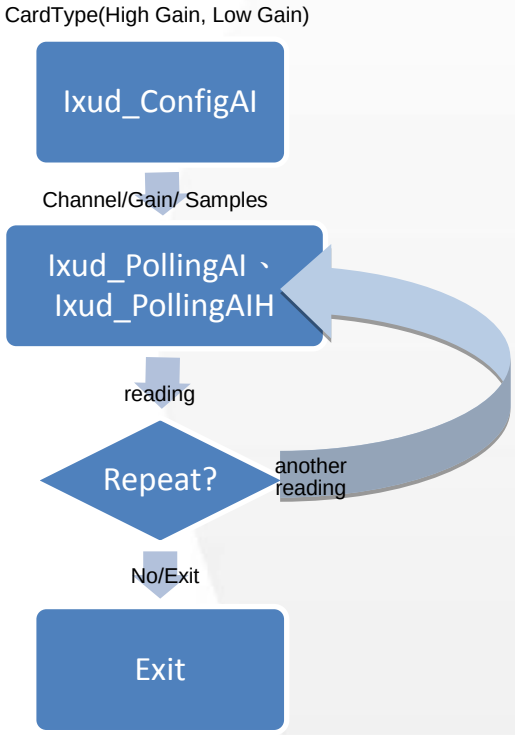
Call Flow



■ Multiple Points Reading

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

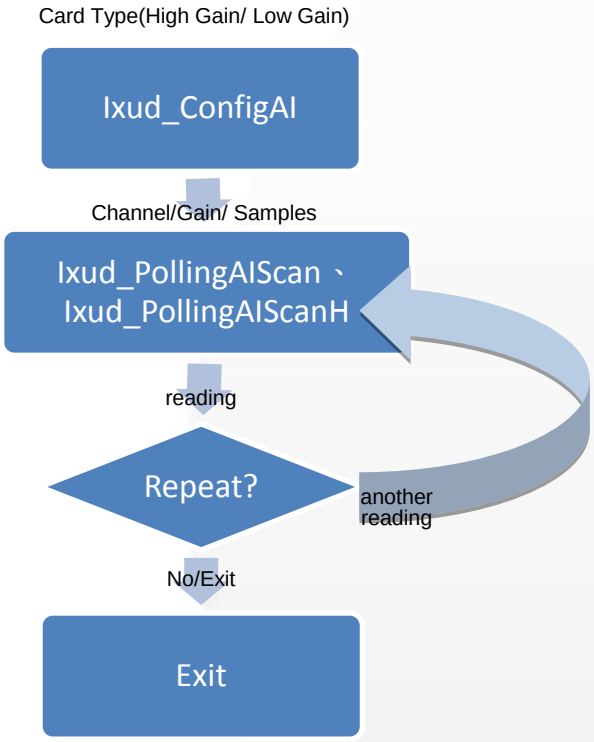
Call Flow



■ Multiple Channel Scan

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

Call Flow



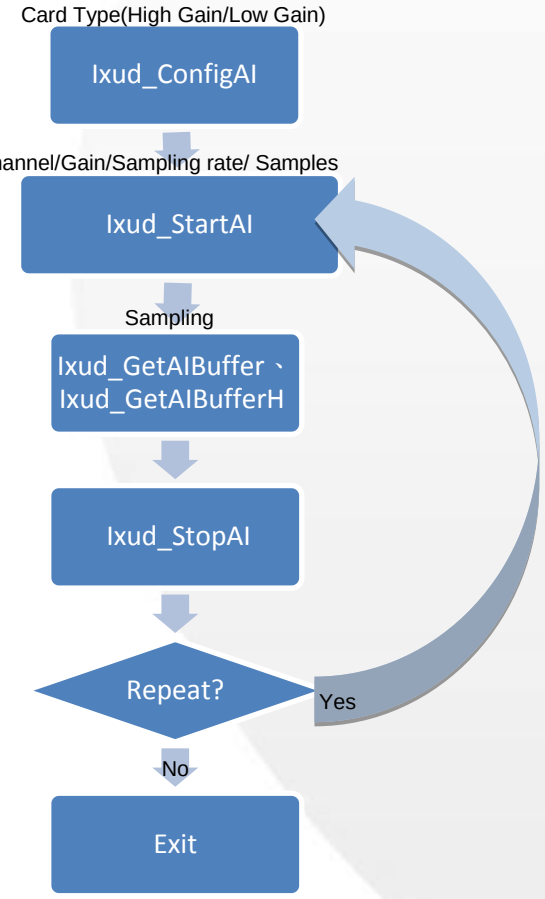
- **Waveform Data Reading**

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.

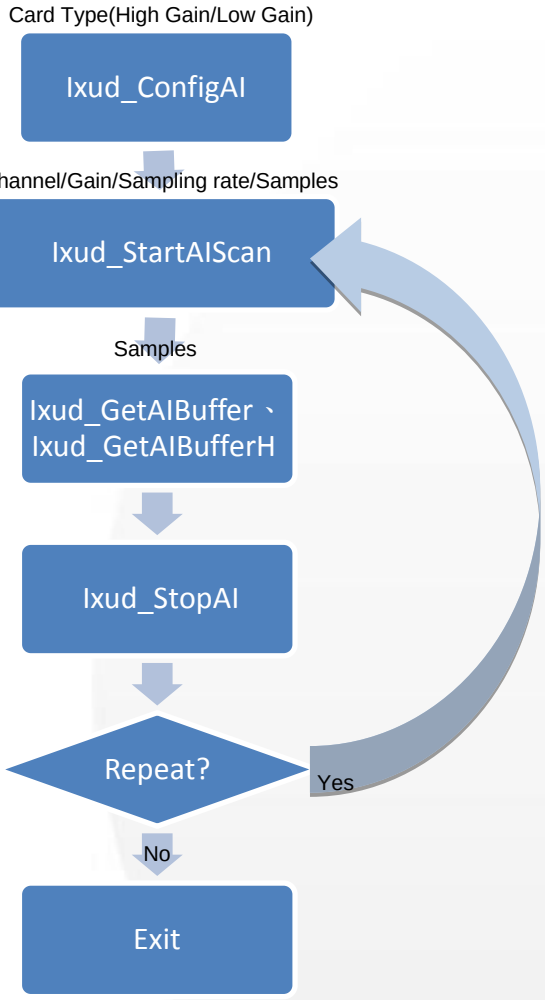
Call Flow



- **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



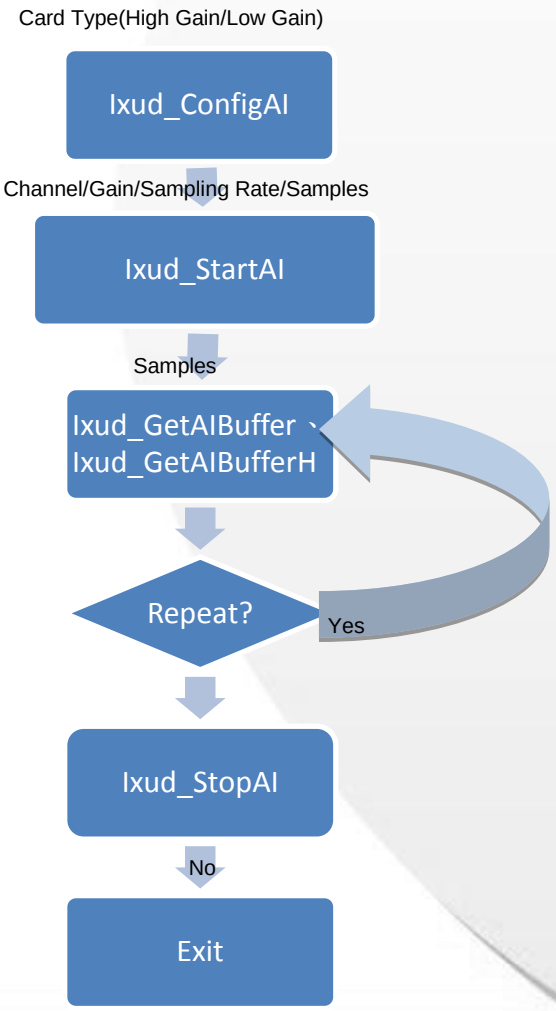
- **Long-term monitoring**

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

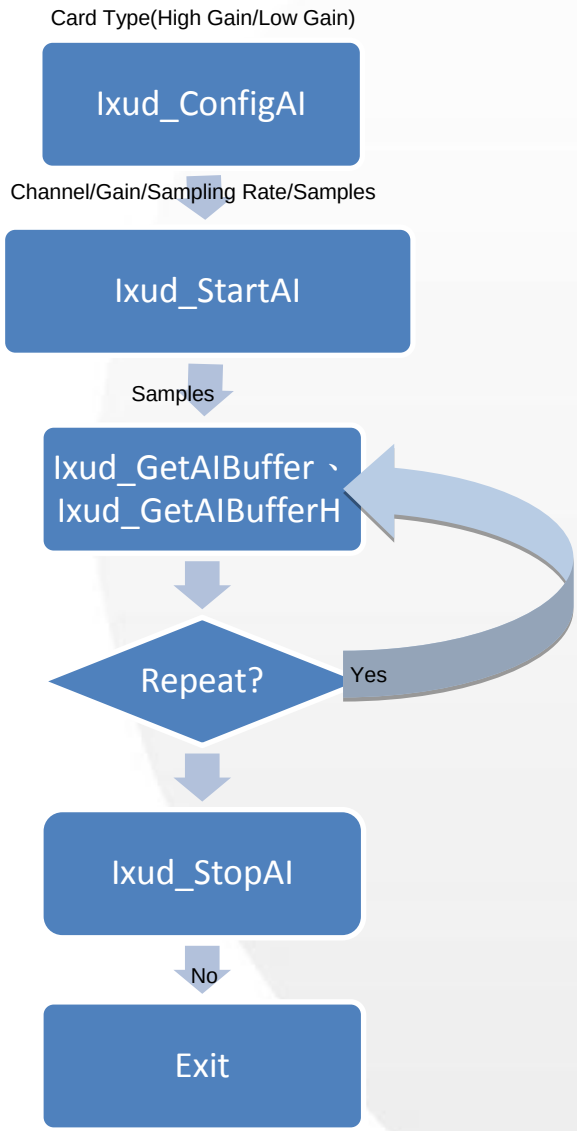
  Monitor the single form single channel to use continuous operation.

Call Flow



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

Call Flow



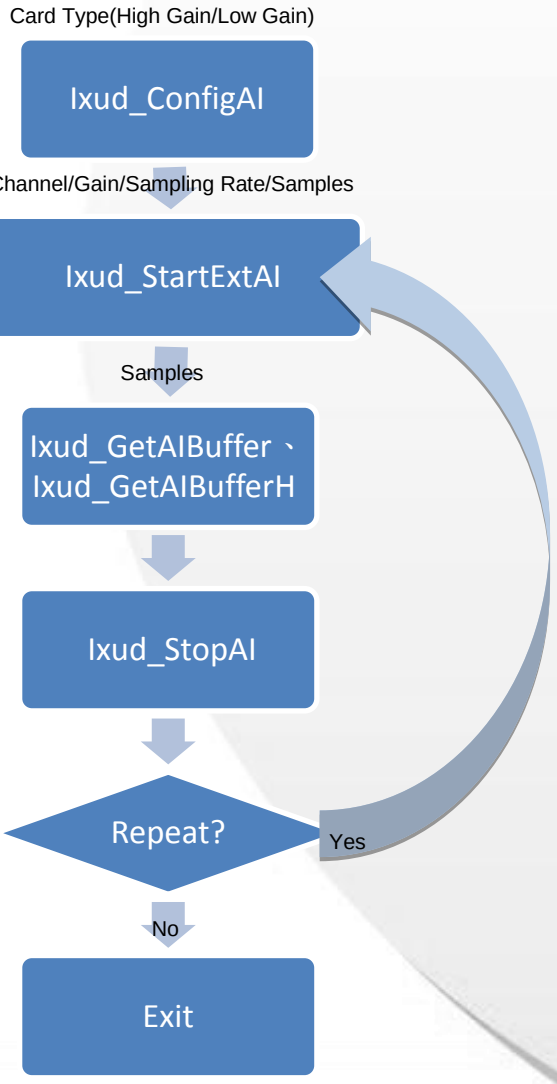
- External trigger operation

  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

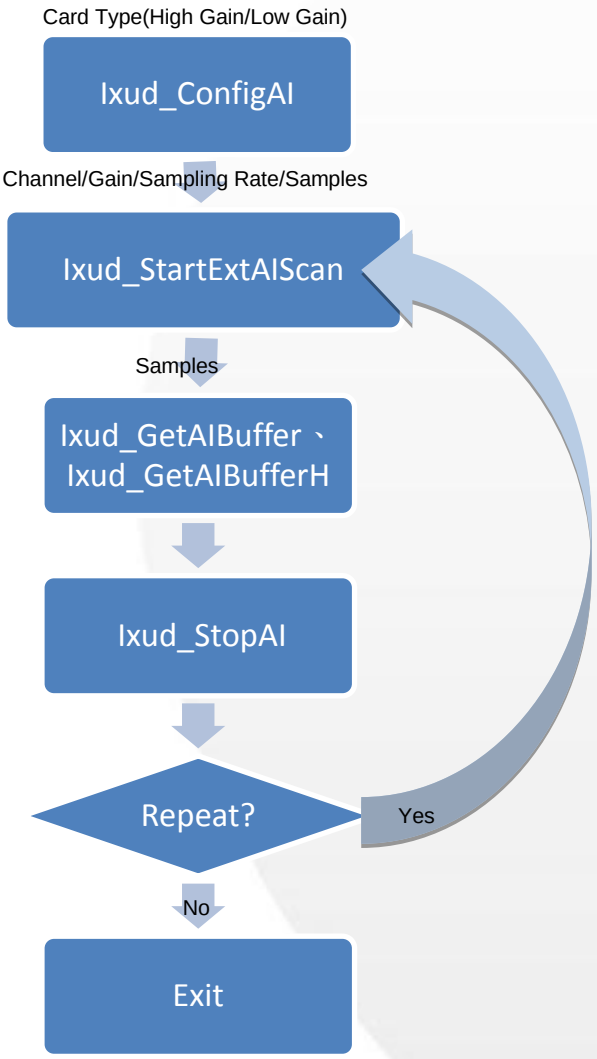
    Acquire the data by one channel on external trigger mode.

Call Flow



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

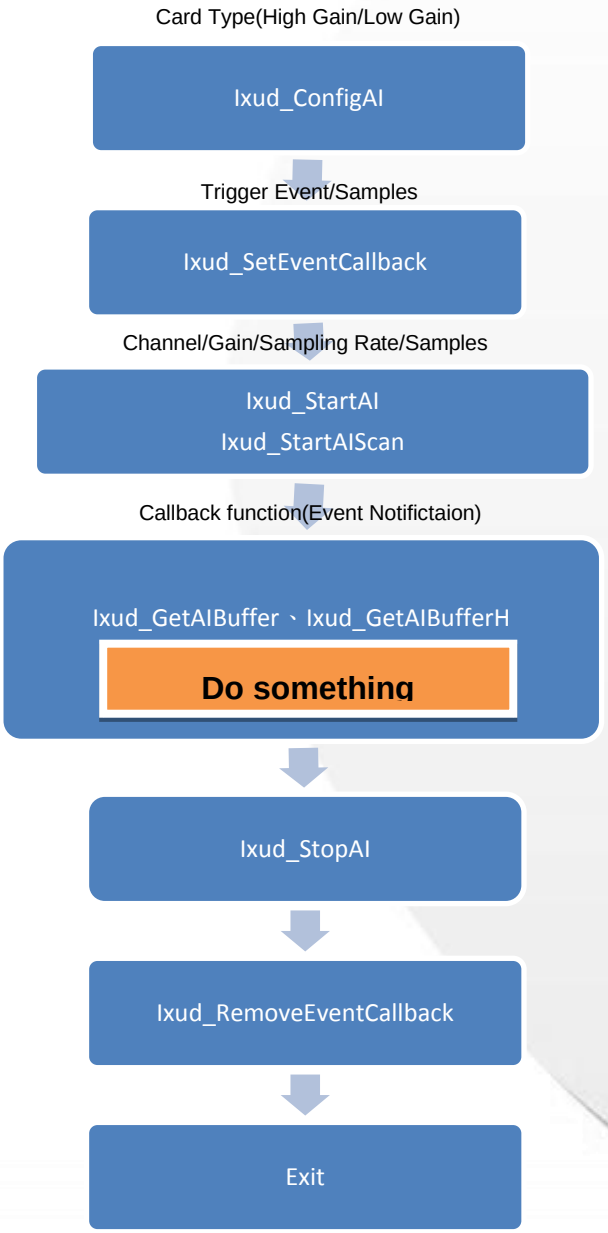
Call Flow



- 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

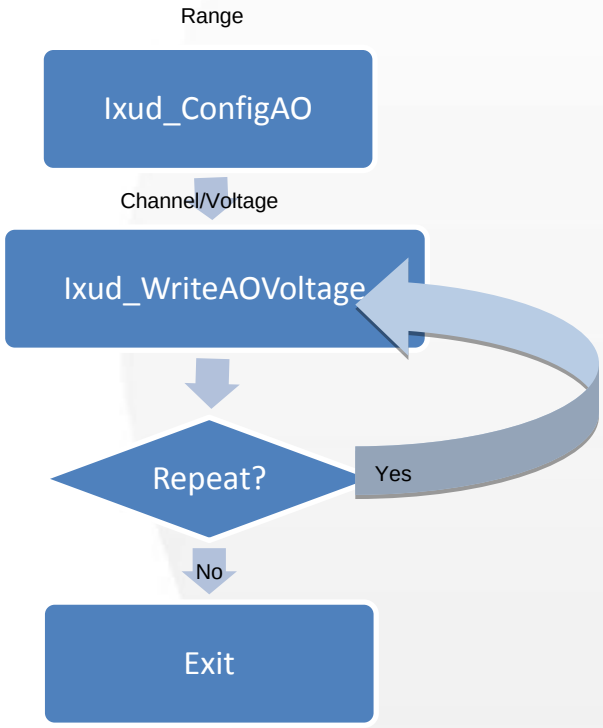


4.5. Analog Output

The analog output function group performs analog output functions.

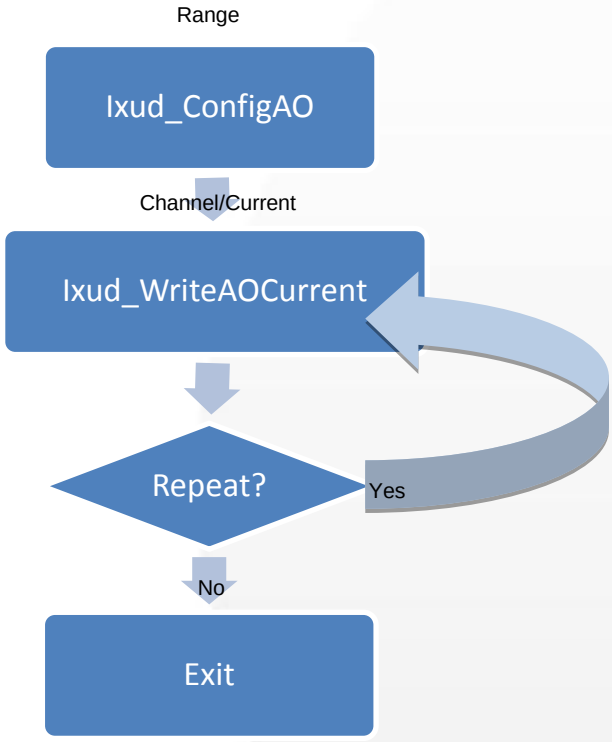
- Static voltage output

Call Flow



- Static current output

Call Flow

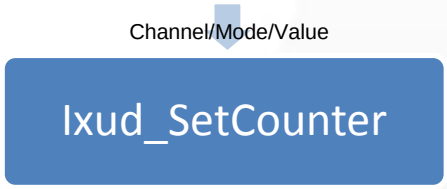


4.6. Timer/Counter

The timer/counter function group performs the counter operation.

- Write timer/counter

Call Flow



- Read timer/counter

Call Flow

4.7. Memory R/W

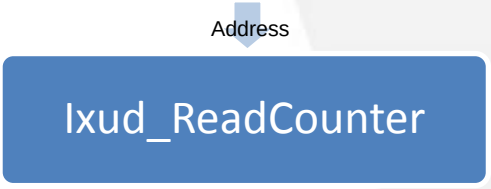
The memory function group writes or reads by byte/word/dword data to a memory.

- 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

Function name	Ixud_DriverInit	Ixud_DriverClose	Ixud_SearchCard	Ixud_GetCardInfo	Ixud_GetBoardNoByCardID
PIO-D24/D24U/D56/D56U	✔	✔	✔	✔	✔
PIO-D48/D48U/D48SU	✔	✔	✔	✔	✔
PIO-D64/D64U	✔	✔	✔	✔	✔
PIO-D96/D96U/D96SU	✔	✔	✔	✔	✔
PIO-D144/D144U/D144LU	✔	✔	✔	✔	✔
PIO-D168/D168U	✔	✔	✔	✔	✔
PEX-D24/D56	✔	✔	✔	✔	✔
PEX-D48	✔	✔	✔	✔	✔
PIO-DA4/DA4U, PEX-DA4	✔	✔	✔	✔	✔
PIO-DA8/DA8U, PEX-DA8	✔	✔	✔	✔	✔
PIO-DA16/DA16U,PEX-DA16	✔	✔	✔	✔	✔
PIO-821 Series	✔	✔	✔	✔	✔
PISO-P64/P64U, PEX-P64	✔	✔	✔	✔	✔
PISO-A64/C64/C64U, PEX-C64	✔	✔	✔	✔	✔
PISO-P32C32/P32C32U/P32S32WU	✔	✔	✔	✔	✔
PEX-P32C32	✔	✔	✔	✔	✔
PISO-P32A32/P32A32U	✔	✔	✔	✔	✔
PISO-725/730/730A	✔	✔	✔	✔	✔
PISO-P16R16U	✔	✔	✔	✔	✔
PEX-P16R16i/P8R8i	✔	✔	✔	✔	✔
PISO-P8R8/P8R8U	✔	✔	✔	✔	✔
PISO-P8R8UDC/AC	✔	✔	✔	✔	✔
PISO-DA4U/DA8U/DA16U	✔	✔	✔	✔	✔
PISO-DA2/DA2U	✔	✔	✔	✔	✔
PISO-813/813U	✔	✔	✔	✔	✔
PCI-TMC12A	✔	✔	✔	✔	✔
PCI-M512/M512U	✔	✔	✔	✔	✔
PCI-P16R16/P16POR16	✔	✔	✔	✔	✔
PCI-P16C16/P8R8	✔	✔	✔	✔	✔
PEX-P16POR16i/P8POR8i	✔	✔	✔	✔	✔
PCI-1002 series, PEX-1002 series	✔	✔	✔	✔	✔
PCI-1202 series, PEX-1202 series	✔	✔	✔	✔	✔
PCI-1602 series	✔	✔	✔	✔	✔
PCI-1800/1802 series	✔	✔	✔	✔	✔
PCI-822LU/826LU	✔	✔	✔	✔	✔
PCI-2602U	✔	✔	✔	✔	✔
PCI-FC16U	✔	✔	✔	✔	✔

Function name	Ixud_ReadPort	Ixud_WritePort	Ixud_ReadPort32	Ixud_WritePort32	Ixud_SetDIOModes32
PIO-D24/D24U/D56/D56U	✔	✔	✔	✔	✔
PIO-D48/D48U/D48SU	✔	✔	✔	✔	✔
PIO-D64/D64U	✔	✔	✔	✔	✔
PIO-D96/D96U/D96SU	✔	✔	✔	✔	✔
PIO-D144/D144U/D144LU	✔	✔	✔	✔	✔
PIO-D168/D168U	✔	✔	✔	✔	✔
PEX-D24/D56	✔	✔	✔	✔	✔
PEX-D48	✔	✔	✔	✔	✔
PIO-DA4/DA4U, PEX-DA4	✔	✔	✔	✔	
PIO-DA8/DA8U, PEX-DA8	✔	✔	✔	✔	
PIO-DA16/DA16U,PEX-DA16	✔	✔	✔	✔	
PIO-821 Series	✔	✔	✔	✔	
PISO-P64/P64U, PEX-P64	✔	✔	✔	✔	
PISO-A64/C64/C64U, PEX-C64	✔	✔	✔	✔	
PISO-P32C32/P32C32U/P32S32WU	✔	✔	✔	✔	
PEX-P32C32	✔	✔	✔	✔	
PISO-P32A32/P32A32U	✔	✔	✔	✔	
PISO-725/730/730A	✔	✔	✔	✔	
PISO-P16R16U	✔	✔	✔	✔	
PEX-P16R16i/P8R8i	✔	✔	✔	✔	
PISO-P8R8/P8R8U	✔	✔	✔	✔	
PISO-P8R8UDC/AC	✔	✔	✔	✔	
PISO-DA4U/DA8U/DA16U	✔	✔	✔	✔	
PISO-DA2/DA2U	✔	✔	✔	✔	
PISO-813/813U	✔	✔	✔	✔	
PCI-TMC12A	✔	✔	✔	✔	
PCI-M512/M512U	✔	✔	✔	✔	
PCI-P16R16/P16POR16	✔	✔	✔	✔	
PCI-P16C16/P8R8	✔	✔	✔	✔	
PEX-P16POR16i/P8POR8i	✔	✔	✔	✔	
PCI-1002 series, PEX-1002 series	✔	✔	✔	✔	
PCI-1202 series, PEX-1202 series	✔	✔	✔	✔	
PCI-1602 series	✔	✔	✔	✔	
PCI-1800/1802 series	✔	✔	✔	✔	
PCI-822LU/826LU	✔	✔	✔	✔	✔
PCI-2602U	✔	✔	✔	✔	✔
PCI-FC16U	✔	✔	✔	✔	✔

Function name	Ixud_SetDIOMode	Ixud_ReadDI	Ixud_WriteDO	Ixud_ReadDIBit Ixud_ReadDI32	Ixud_WriteDOBit Ixud_WriteDO32
PIO-D24/D24U/D56/D56U	✔	✔	✔	✔	✔
PIO-D48/D48U/D48SU	✔	✔	✔	✔	✔
PIO-D64/D64U	✔	✔	✔	✔	✔
PIO-D96/D96U/D96SU	✔	✔	✔	✔	✔
PIO-D144/D144U/D144LU	✔	✔	✔	✔	✔
PIO-D168/D168U	✔	✔	✔	✔	✔
PEX-D24/D56	✔	✔	✔	✔	✔
PEX-D48	✔	✔	✔	✔	✔
PIO-DA4/DA4U, PEX-DA4		✔	✔	✔	✔
PIO-DA8/DA8U, PEX-DA8		✔	✔	✔	✔
PIO-DA16/DA16U,PEX-DA16		✔	✔	✔	✔
PIO-821 Series		✔	✔	✔	✔
PISO-P64/P64U, PEX-P64		✔		✔	
PISO-A64/C64/C64U, PEX-C64			✔		✔
PISO-P32C32/P32C32U/P32S32WU		✔	✔	✔	✔
PEX-P32C32		✔	✔	✔	✔
PISO-P32A32/P32A32U		✔	✔	✔	✔
PISO-725/730/730A		✔	✔	✔	✔
PISO-P16R16U		✔	✔	✔	✔
PEX-P16R16i/P8R8i		✔	✔	✔	✔
PISO-P8R8/P8R8U		✔	✔	✔	✔
PISO-P8R8UDC/AC		✔	✔	✔	✔
PISO-DA4U/DA8U/DA16U		✔	✔	✔	✔
PISO-DA2/DA2U					
PISO-813/813U					
PCI-TMC12A		✔	✔	✔	✔
PCI-M512/M512U		✔	✔	✔	✔
PCI-P16R16/P16POR16		✔	✔	✔	✔
PCI-P16C16/P8R8		✔	✔	✔	✔
PEX-P16POR16i/P8POR8i		✔	✔	✔	✔
PCI-1002 series, PEX-1002 series		✔	✔	✔	✔
PCI-1202 series, PEX-1202 series		✔	✔	✔	✔
PCI-1602 series		✔	✔	✔	✔
PCI-1800/1802 series		✔	✔	✔	✔
PCI-822LU/826LU	✔	✔	✔	✔	✔
PCI-2602U	✔	✔	✔	✔	✔
PCI-FC16U	✔	✔	✔	✔	✔

Function name	Ixud_SoftwareReadbackDO	Ixud_SetEventCallback Ixud_RemoveEventCallback	Ixud_InstallIrq Ixud_RemoveIrq
PIO-D24/D24U/D56/D56U	✔	✔	✔
PIO-D48/D48U/D48SU	✔	✔	✔
PIO-D64/D64U	✔	✔	✔
PIO-D96/D96U/D96SU	✔	✔	✔
PIO-D144/D144U/D144LU	✔	✔	✔
PIO-D168/D168U	✔	✔	✔
PEX-D24/D56	✔	✔	✔
PEX-D48	✔	✔	✔
PIO-DA4/DA4U, PEX-DA4	✔	✔	✔
PIO-DA8/DA8U, PEX-DA8	✔	✔	✔
PIO-DA16/DA16U,PEX-DA16	✔	✔	✔
PIO-821 Series	✔	✔	
PISO-P64/P64U, PEX-P64	✔		
PISO-A64/C64/C64U, PEX-C64	✔		
PISO-P32C32/P32C32U/P32S32WU	✔		
PEX-P32C32	✔		
PISO-P32A32/P32A32U	✔		
PISO-725/730/730A	✔	✔	✔
PISO-P16R16U	✔		
PEX-P16R16i/P8R8i	✔		
PISO-P8R8/P8R8U	✔		
PISO-P8R8UDC/AC	✔		
PISO-DA4U/DA8U/DA16U	✔	✔	✔
PISO-DA2/DA2U		✔	✔
PISO-813/813U			
PCI-TMC12A	✔	✔	✔
PCI-M512/M512U	✔		
PCI-P16R16/P16POR16	✔		
PCI-P16C16/P8R8	✔		
PEX-P16POR16i/P8POR8i	✔		
PCI-1002 series, PEX-1002 series	✔	✔	
PCI-1202 series, PEX-1202 series	✔		
PCI-1602 series	✔		
PCI-1800/1802 series	✔		
PCI-822LU/826LU	✔	✔	
PCI-2602U	✔		
PCI-FC16U	✔		

Function name	Ixud_ConfigAI Ixud_ConfigAIEx	Ixud_ClearAIBuffer	Ixud_GetBufferStatus	Ixud_ReadAI	Ixud_ReadAIH
PIO-D24/D24U/D56/D56U					
PIO-D48/D48U/D48SU					
PIO-D64/D64U					
PIO-D96/D96U/D96SU					
PIO-D144/D144U/D144LU					
PIO-D168/D168U					
PEX-D24/D56					
PEX-D48					
PIO-DA4/DA4U, PEX-DA4					
PIO-DA8/DA8U, PEX-DA8					
PIO-DA16/DA16U,PEX-DA16					
PIO-821 Series	✔	✔	✔	✔	✔
PISO-P64/P64U, PEX-P64					
PISO-A64/C64/C64U, PEX-C64					
PISO-P32C32/P32C32U/P32S32WU					
PEX-P32C32					
PISO-P32A32/P32A32U					
PISO-725/730/730A					
PISO-P16R16U					
PEX-P16R16i/P8R8i					
PISO-P8R8/P8R8U					
PISO-P8R8UDC/AC					
PISO-DA4U/DA8U/DA16U					
PISO-DA2/DA2U					
PISO-813/813U	✔			✔	✔
PCI-TMC12A					
PCI-M512/M512U					
PCI-P16R16/P16POR16					
PCI-P16C16/P8R8					
PEX-P16POR16i/P8POR8i					
PCI-1002 series, PEX-1002 series	✔	✔	✔	✔	✔
PCI-1202 series, PEX-1202 series	✔	✔	✔	✔	✔
PCI-1602 series	✔	✔	✔	✔	✔
PCI-1800/1802 series	✔	✔	✔	✔	✔
PCI-822LU/826LU	✔	✔	✔	✔	✔
PCI-2602U	✔	✔	✔	✔	✔
PCI-FC16U					

Function name	lxud_PollingAI	lxud_PollingAIH	lxud_PollingAIScan	lxud_PollingAIScanH
PIO-D24/D24U/D56/D56U				
PIO-D48/D48U/D48SU				
PIO-D64/D64U				
PIO-D96/D96U/D96SU				
PIO-D144/D144U/D144LU				
PIO-D168/D168U				
PEX-D24/D56				
PEX-D48				
PIO-DA4/DA4U, PEX-DA4				
PIO-DA8/DA8U, PEX-DA8				
PIO-DA16/DA16U,PEX-DA16				
PIO-821 Series	✔	✔	✔	✔
PISO-P64/P64U, PEX-P64				
PISO-A64/C64/C64U, PEX-C64				
PISO-P32C32/P32C32U/P32S32WU				
PEX-P32C32				
PISO-P32A32/P32A32U				
PISO-725/730/730A				
PISO-P16R16U				
PEX-P16R16i/P8R8i				
PISO-P8R8/P8R8U				
PISO-P8R8UDC/AC				
PISO-DA4U/DA8U/DA16U				
PISO-DA2/DA2U				
PISO-813/813U	✔	✔	✔	✔
PCI-TMC12A				
PCI-M512/M512U				
PCI-P16R16/P16POR16				
PCI-P16C16/P8R8				
PEX-P16POR16i/P8POR8i				
PCI-1002 series, PEX-1002 series	✔	✔	✔	✔
PCI-1202 series, PEX-1202 series	✔	✔	✔	✔
PCI-1602 series	✔	✔	✔	✔
PCI-1800/1802 series	✔	✔	✔	✔
PCI-822LU/826LU	✔	✔	✔	✔
PCI-2602U	✔	✔	✔	✔
PCI-FC16U				

Function name	Ixud_StartAI Ixud_StopAI	Ixud_StartAIScan	Ixud_StartExtAI	Ixud_StartExtAIScan	Ixud_GetAIBuffer Ixud_GetAIBufferH
PIO-D24/D24U/D56/D56U					
PIO-D48/D48U/D48SU					
PIO-D64/D64U					
PIO-D96/D96U/D96SU					
PIO-D144/D144U/D144LU					
PIO-D168/D168U					
PEX-D24/D56					
PEX-D48					
PIO-DA4/DA4U, PEX-DA4					
PIO-DA8/DA8U, PEX-DA8					
PIO-DA16/DA16U,PEX-DA16					
PIO-821 Series	✔				✔
PISO-P64/P64U, PEX-P64					
PISO-A64/C64/C64U, PEX-C64					
PISO-P32C32/P32C32U/P32S32WU					
PEX-P32C32					
PISO-P32A32/P32A32U					
PISO-725/730/730A					
PISO-P16R16U					
PEX-P16R16i/P8R8i					
PISO-P8R8/P8R8U					
PISO-P8R8UDC/AC					
PISO-DA4U/DA8U/DA16U					
PISO-DA2/DA2U					
PISO-813/813U					
PCI-TMC12A					
PCI-M512/M512U					
PCI-P16R16/P16POR16					
PCI-P16C16/P8R8					
PEX-P16POR16i/P8POR8i					
PCI-1002 series, PEX-1002 series	✔	✔			✔
PCI-1202 series, PEX-1202 series	✔	✔	✔	✔	✔
PCI-1602 series	✔	✔	✔	✔	✔
PCI-1800/1802 series	✔	✔	✔	✔	✔
PCI-822LU/826LU	✔	✔	✔	✔	✔
PCI-2602U	✔	✔	✔	✔	✔
PCI-FC16U					

Function name	Ixud_ConfigAO	Ixud_WriteAOVoltage Ixud_WriteAOVoltageH	Ixud_WriteAOCurrent Ixud_WriteAOCurrentH
PIO-D24/D24U/D56/D56U			
PIO-D48/D48U/D48SU			
PIO-D64/D64U			
PIO-D96/D96U/D96SU			
PIO-D144/D144U/D144LU			
PIO-D168/D168U			
PEX-D24/D56			
PEX-D48			
PIO-DA4/DA4U, PEX-DA4	✔	✔	✔
PIO-DA8/DA8U, PEX-DA8	✔	✔	✔
PIO-DA16/DA16U,PEX-DA16	✔	✔	✔
PIO-821 Series	✔	✔	✔
PISO-P64/P64U, PEX-P64			
PISO-A64/C64/C64U, PEX-C64			
PISO-P32C32/P32C32U/P32S32WU			
PEX-P32C32			
PISO-P32A32/P32A32U			
PISO-725/730/730A			
PISO-P16R16U			
PEX-P16R16i/P8R8i			
PISO-P8R8/P8R8U			
PISO-P8R8UDC/AC			
PISO-DA4U/DA8U/DA16U	✔	✔	✔
PISO-DA2/DA2U	✔	✔	✔
PISO-813/813U			
PCI-TMC12A			
PCI-M512/M512U			
PCI-P16R16/P16POR16			
PCI-P16C16/P8R8			
PEX-P16POR16i/P8POR8i			
PCI-1002 series, PEX-1002 series			
PCI-1202 series, PEX-1202 series	✔	✔	✔
PCI-1602 series	✔	✔	✔
PCI-1800/1802 series	✔	✔	✔
PCI-822LU/826LU	✔	✔	✔
PCI-2602U	✔	✔	✔
PCI-FC16U			

Function name	Ixud_ReadCounter	Ixud_SetCounter	Ixud_DisableCounter
PIO-D24/D24U			
PIO-D56/D56U			
PIO-D48/D48U	✔	✔	✔
PIO-D64/D64U	✔	✔	✔
PIO-D96/D96U			
PIO-D144/D144U			
PIO-D168/D168U			
PEX-D24/D56			
PEX-D48	✔	✔	✔
PIO-DA4/DA4U	✔	✔	✔
PIO-DA8/DA8U	✔	✔	✔
PIO-DA16/DA16U	✔	✔	✔
PIO-812L/H	✔	✔	✔
PISO-P64/P64U			
PISO-A64/C64/C64U			
PISO-P32C32/P32C32U			
PISO-P32S32WU			
PISO-P32A32/P32A32U			
PISO-725/730/730A			
PISO-P16R16U			
PEX-P16R16i/P8R8i			
PISO-P8R8/P8R8U			
PISO-P8R8UDC/AC			
PISO-DA4U/DA8U/DA16U	✔	✔	✔
PISO-DA2/DA2U	✔	✔	✔
PISO-813			
PCI-TMC12A	✔	✔	✔
PCI-M512/M512U			
PCI-P16R16/P16POR16			
PCI-P16C16/P8R8			
PEX-P16POR16/P8POR8			
PCI-1002 series			
PCI-1202 series			
PCI-1602 series			
PCI-1800/1802 series			
PCI-822LU/826LU			
PCI-2602U			
PCI-FC16U			

Function name	PCI-M512U
Ixud_ReadMemory Ixud_WriteMemory	✔
Ixud_ReadMemory32 Ixud_WriteMemory32	✔

Function name	PCI-2602U
Ixud_StartDO Ixud_StopDO	✔
Ixud_StartDI Ixud_StopDI Ixud_GetDIBufferH	✔
Ixud_ConfiAIEx 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 wModelNo
);
```

- **Parameters**

  wTotalBoards

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

wModelNo

[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 wSizeParameters 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_WriteReadPort32(
    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. ◦

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. ◦

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

Writesdigital 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_WriteDO(
        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.

➢ **Syntax**

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

Only support the PCI-2602U

➢ **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 fSamplingRate 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_GetAIBufferH(
        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_StopAI(
        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

- **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 wEventTypeMask,

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	PIO-D144U PIO-D144LU PIO-D168U
1	P2C0	P2C3/P2C7	EXTIRQ	P2C0	P2C0
2	P2C1	P5C3/P5C7	EVTIRQ	P5C0	P2C1
3	P2C2	COUT0	TMRIRQ	P8C0	P2C2
4	P2C3	COUT2	-	P11C0	P2C3

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	PISO-725	PCI-TMC12A	PIO-821 PCI-1002 PEX-1002	PCI-822LU PCI-826LU PCI-2602U
1	COUT0	COUT0	DI0	IDI0	COUT3/6/9/12/Ext	AD Data	AD Data
2	COUT2	COUT2	DI1	IDI1	-	-	-
3	-		-	IDI2	-	-	-
3	-		-	IDI3	-	-	-
	-		-	IDI4	-	-	-
	-		-	IDI5	-	-	-
	-		-	IDI6	-	-	-
	-		-	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 call edafter 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_Installlrq

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_Installlrq(

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 soruce (INT_0), and so on.

Bit	PIO-D24U PEX-D24 PIO-D56U PEX-D56	PIO-D48U PIO-D48S PEX-D48	PIO-D64U	PIO-D96U PIO-D96S	PIO-D144U PIO-D144LU PIO-D168U	PIO-DA4U PIO-DA8U PIO-DA16U PISO-DA4U PISO-DA8U PISO-DA16U	PEX-DA4 PEX-DA8 PEX-DA16
0	P2C0	P2C3/P2C7	EXTIRQ	P2C0	P2C0	COUT0	COUT0
1	P2C1	P5C3/P5C7	EVTIRQ	P5C0	P2C1	COUT2	COUT2
2	P2C2	COUT0	TMRIRQ	P8C0	P2C2	-	
3	P2C3	COUT2	-	P11C0	P2C3	-	

Bit	PISO-730 PISO-730A PISO-730U PISO-730AU	PISO-725	PCI-TMC12A
0	DI0	IDI0	COUT3/6/9/12/ EXT
1	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.

➢ **Eample**

```
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 BufferSizeCount,
        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.

wBufferSizeCount

[Input] Analog input buffer size in PC memory. The unit is DWORD. Default number is 524288 length (wBufferSizeCount = 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
0	PCI-2602U
1	-

Table 5-7 wCardTypeParameters 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 BufferSizeCount,
        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.

wBufferSizeCount

[Input] Analog input buffer size in PC memory.The unit is DWORD. Default number is 524288 count(wBufferSizeCount = 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
0	PCI-2602U
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
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, unknow 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 unscaled 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 = lxud_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 amout 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:

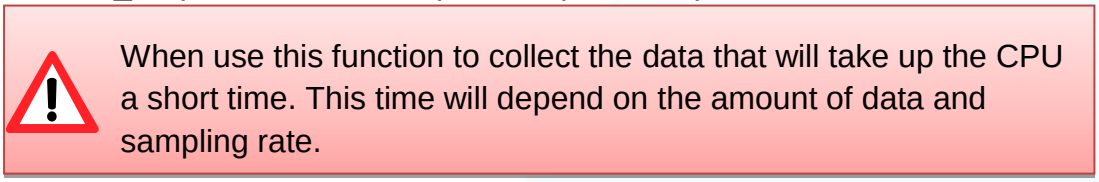
1. When sampling rate is too fast, it is prone to develop FIFO overflow problem.
2. On continuous mode, analog input data will be stored in PC memory of user allocation. User must take the data on the suit time(Before the buffer overflow)
3. On data acquisition processing, user must reduce the CPU loading,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



➢ **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 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_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 dwPerDataCount

);

➢ **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
0	Failing Edge	Any Edge
1	Rasing 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

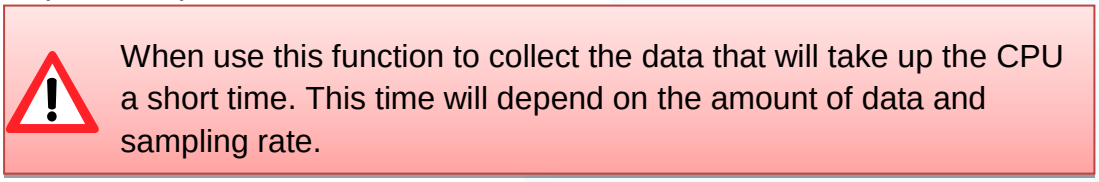
[Input] Below trigger voltage range.

- **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.



➢ **Syntax**

```
WORD Ixud_StartExtAIScan(
        WORD wBoardNo,
        WORD wActive,
        WORD wChannels,
        WORD wChannelList[ ],
        WORD wConfigList[ ],
        float fSamplingRate,
        DWORD dwDataCountPostChannel,
        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.

wActive

[Input] It sets a specified trigger type.

dwActive	PCI-822LU PCI-826LU	PCI-2602U
0	Failing Edge	Any Edge
1	Rasing Edge	Any Edge

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.

dwDataPostCountPostChannel

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

dwDataPreCountPerChannel

[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 binrary 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 arrary, 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 fSamplingRate parameter specifies therate 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 fSamplingRate 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.

wTimeOutMs

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

wStatusy

[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 wSizeParameterssSetting

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 wSizeParameters 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,
        WORD *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.

dwOffset

[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 dwOffset**,**

WORD 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.

dwOffset

[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.

AuxNo

[Output] Aux ID returned.

dwBarVirtualAddress[]

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

Virual 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	Unspported

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.

wCounterChannels

[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/D96SU
PIOD144	800100	PIO-D144
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
PISOP32C32	800820	PISO-P32C32/P32C32U/P32S32WU
PEXP32C32	800820	PEX-P32C32
PISO1730	800820	PISO-1730U
PISOP32A32	800870	PISO-P32A32
PISOP8R8	800830	PISO-P8R8/PISO-P8R8AC/PISO-P8R8DC
PISO730	800840	PISO-730
PISO730A	800880	PISO-730A
PISO725	8008FF	PISO-725
PISODA2	800B00	PISO-DA2
PISO813	800A00	PISO-813/813U
PCITMC12	DF2962	PCI-TMC12/PCI-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
PEXP8POR8	D6102B	PEX-P8POR8i
PCIP16R16	D61E39	PCI-P16R16/P16C16/P16POR16
PEXP16POR16	D61E39	PEX-P16POR16i

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
+/- 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
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	PIO-D144U	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-P32C32U PISO-P32S32WU PISO-1730U PEX-P32C32	PISO-P16R16U PEX-P16R16i	PISO-P8R8U PEX-P8R8i PISO-725	PISO-730 PISO-730U PISO-730A PISO-730AU PEX-730	PCI-P8R8 PEX-P8POR8i	PCI-P16R16 PCI-P16C16 PEX-P16POR16i
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-TMC12A	PCI-2602U
0	DI 0 ~ 11	DI 0 ~ 15	**PA0~7** **PB0~7** **PC0~7** **PD0~7**

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	PIO-D144U PIO-D144LU	PIO-D168U	PISO-A64 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-P32C32U PISO-P32S32WU PISO-1730U PEX-P32C32	PISO-P16R16U PEX-P16R16i	PISO-P8R8U PEX-P8R8i PISO-725	PISO-730 PISO-730A PISO-730U PISO-730AU	PCI-P8R8 PEX-P8POR8i	PCI-P16R16 PCI-P16C16 PEX-P16POR16i
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 ~ 15	DO 0 ~ 15	
1	**PB 0 ~ 15**	DO 8~15	DO 8 ~ 15	DO 8 ~ 15		-	-	

DO Port No.	PCI-M512	PCI-TMC12A	PCI-2602U
0	DO 0 ~ 15	DO 0 ~ 15	**PA 0 ~ 7** **PB 0 ~ 7** **PC 0 ~ 7** **PD 0 ~ 7**

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.