ICP DAS

PCIe-LM4

Driver DLL User Manual

English Version

Supports 64-bit OS

Supports Windows 10

Warning

Copyright

Trademarks

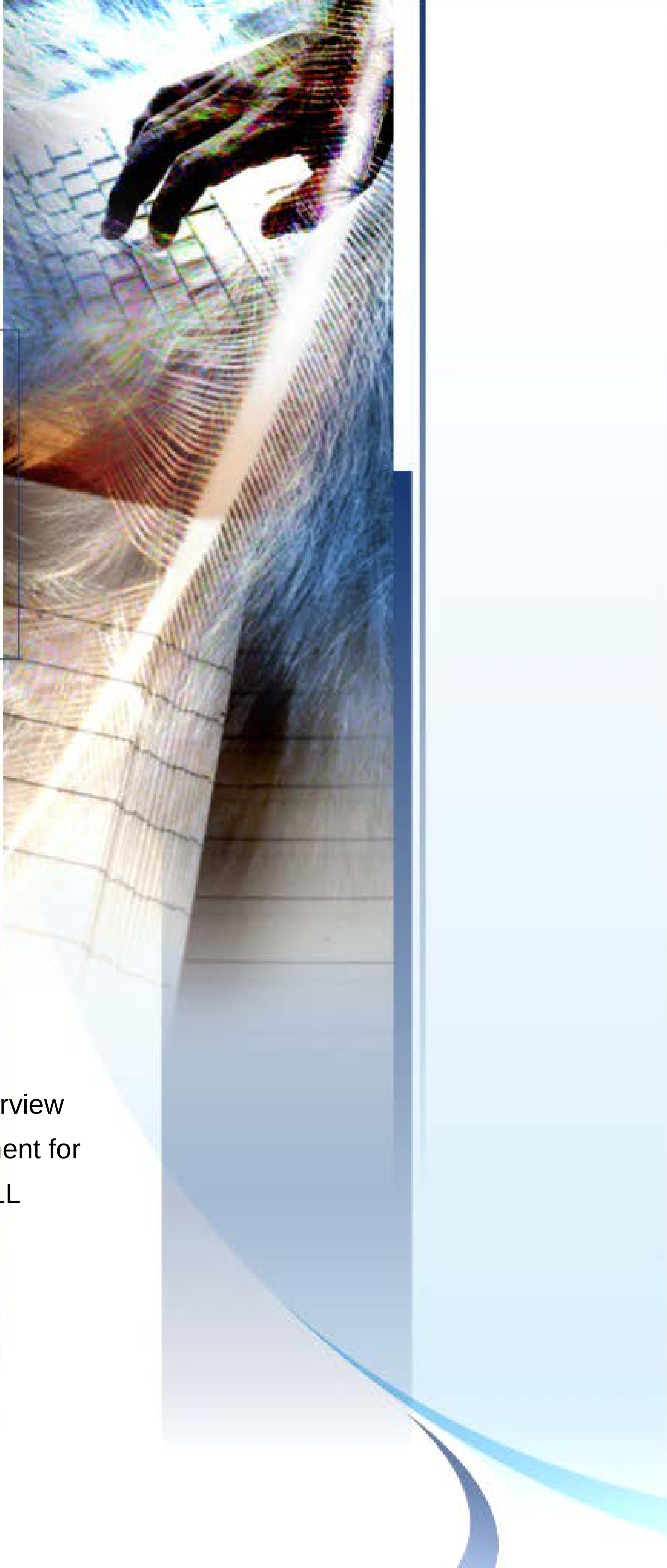
About this Manual

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

The PCIELM4 DLL drivers can be used to develop custom programs based on the VC, VB, VB.NET, C#.NET, VC.NET, Console and other programming languages using Windows Systems. This manual also provides sample programs that can be modified to create custom applications that meet specific requirements.

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

Table of Contents



1. Introduction

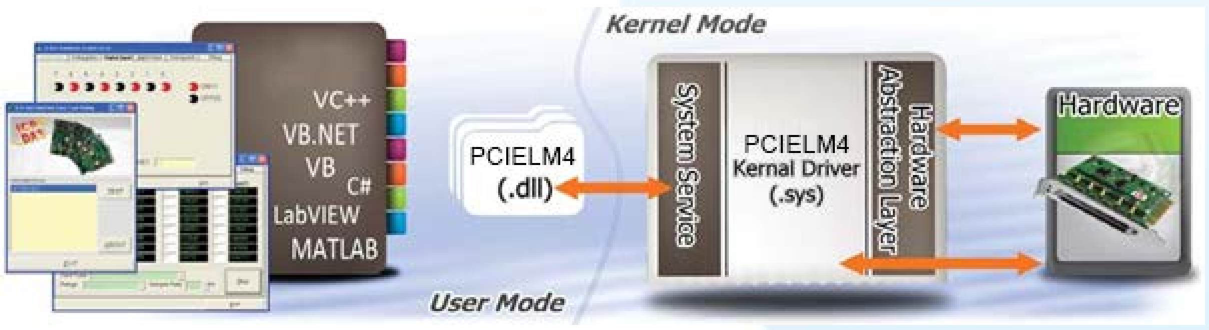
This chapter provides an overview of the functions and requirement for ICP DAS PCIELM4 Driver DLL

1.1. Introducing the PCIELM4 Driver DLL

The ICP DAS PCIELM4 Driver DLL provides complete hardware functions and maximum performance. With the ICP DAS PCIELM4 Driver DLL, there is no need to use hardware-specific register commands thanks to the powerful API function that can be used with a variety of programming environments and languages.

ICP DAS PCIELM4 Driver DLL uses direct I/O techniques to promote API efficiency and I/O speed. It also provides interrupt and event notification functions, so that if an interrupt event occurs within the device, the user application will be notified via a callback function. Then, only the necessary actions need to be taken without needing to manually check the status of the hardware, which is more efficient and reduces the complexity of the application.

The ICP DAS PCIELM4 Driver DLL supports Windows 2000 and both 32- and 64 bit versions Windows XP/2003/Vista/7/2008/8/2012/10.



1.2. Supported ICP DAS Products

The following is a summary of the ICP DAS products supported ICP DAS PCIELM4 Driver DLL.

Model
PCIe-LM4

1.3. System Requirements

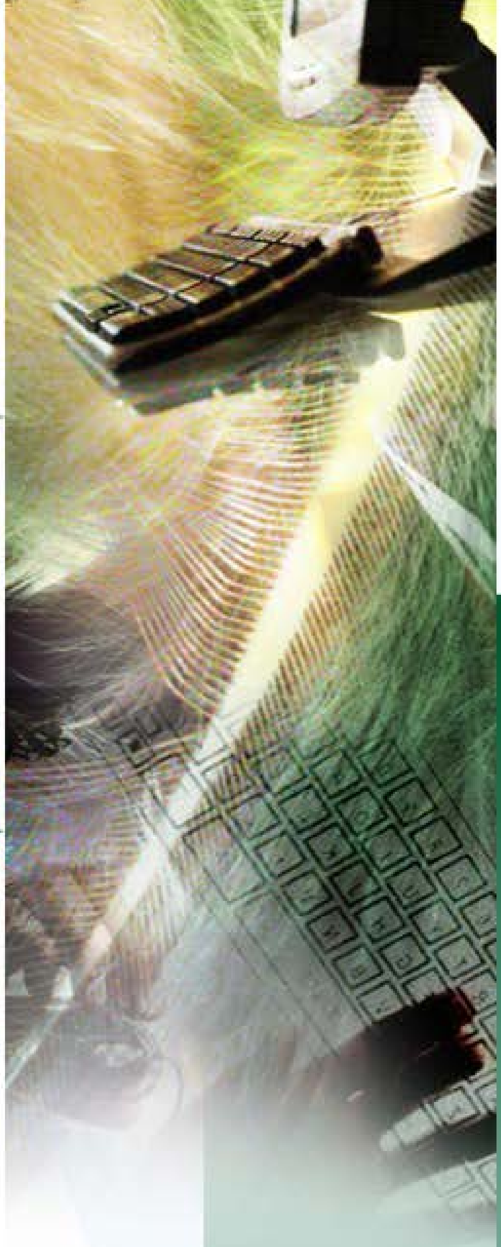
Minimum system requirements for ICP DAS PCIELM4 Driver DLL are:

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

Operating system of Windows requirement

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

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



2. Getting Started

This chapter provides instructions of how to obtain and install the ICP DAS PCIELM4 Driver DLL

2.1. Obtaining the PCIELM4 Driver DLL Installer package

The installer package for the ICP DAS PCIELM4 Driver DLL can be found on the web site. The locations are:

http://www.icpdas.com/en/download/index.php?model=PCIe-LM4

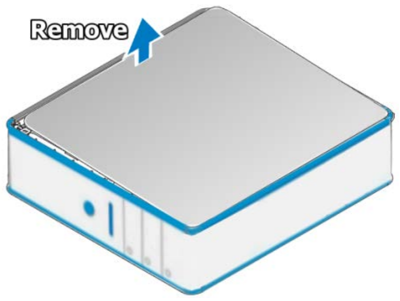
2.2. Installing the PCIELM4 Driver DLL

Step 1 Install the DAQ Card

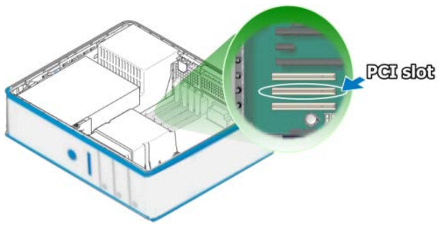
Install DAQ card by following the procedure described below:

Turn off computer
Turn Off

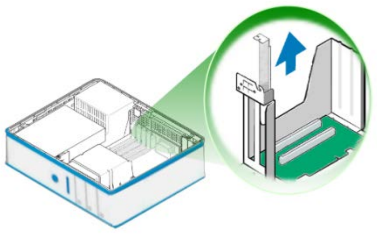
Correctly shut down and power off your computer, and then disconnect the power supply.



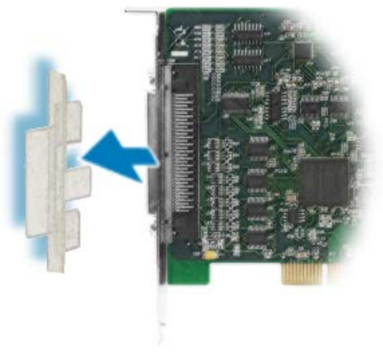
Remove the cover from the computer.



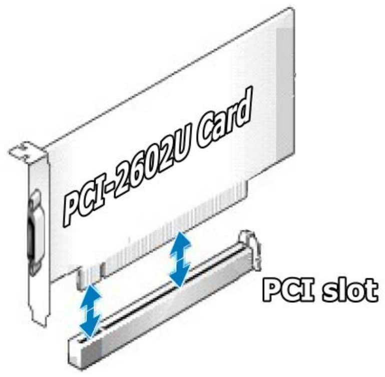
Select an empty PCI or PCIe slot.



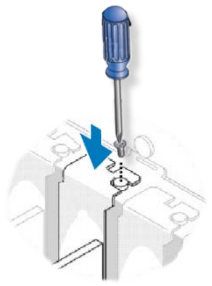
Remove the screw holding the cover for the PCI slot in place and then remove the slot cover from the PC. Ensure that you do not misplace the screw.



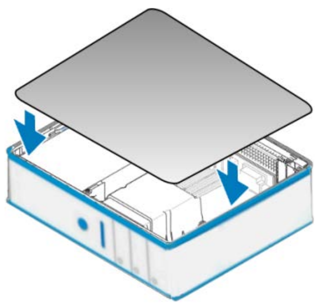
Remove the connector cover form the card.



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



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



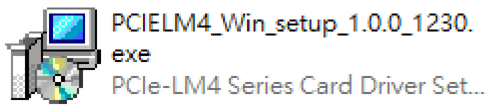
Re-attach cover for the computer and reconnect the power supply.

Power on the computer.

Step 2 Set up the ICP DAS PCIELM4 Driver DLL

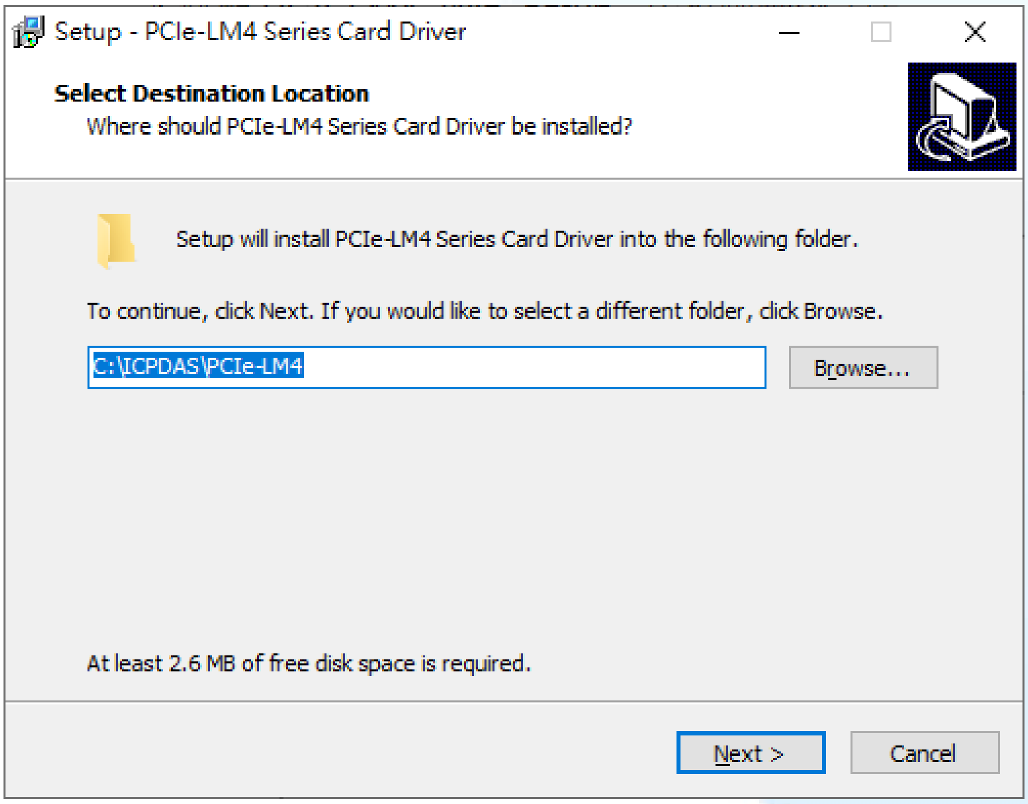
Install PCIELM4 Driver DLL by following the procedure described below

1. Double-click the “PCIELM4_Win_setup_x.x.x.x_xxxx.exe” file to install.



PCIELM4_Win_setup_1.0.0_1230.exe
PCIe-LM4 Series Card Driver Set...

2. Click the “Next>” button to install the software in the default folder, C:\ICPDAS\PCIe-LM4, or click the “Browse…” button to select the destination folder for the installation.



Select Destination Location
Where should PCIe-LM4 Series Card Driver be installed?

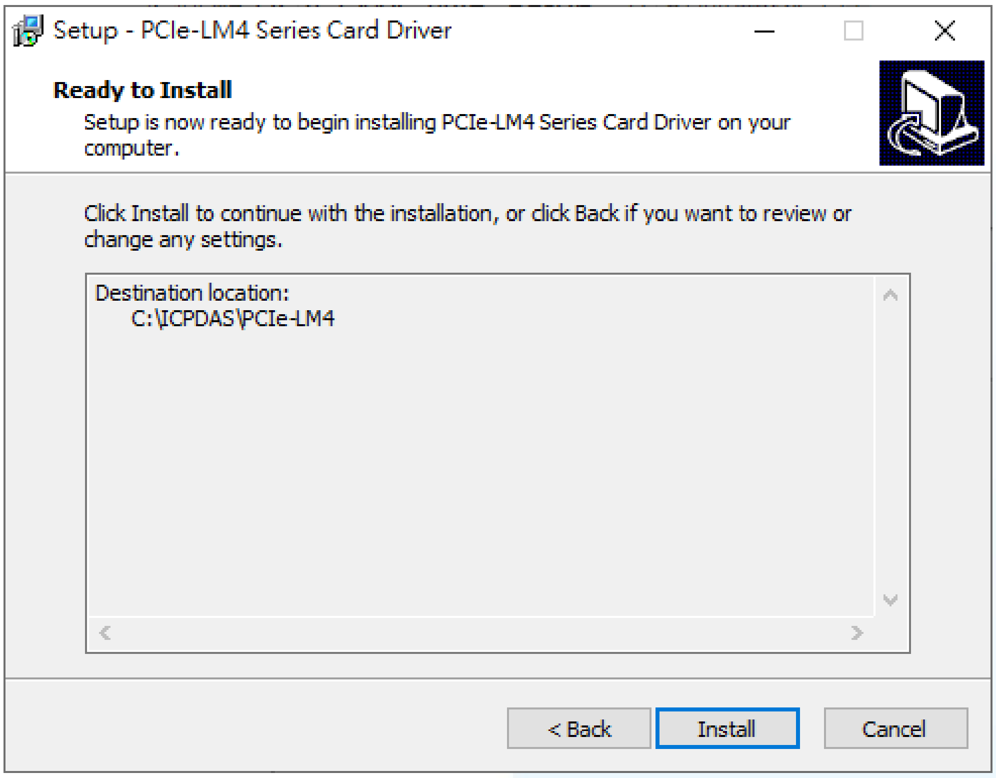
Setup will install PCIe-LM4 Series Card Driver into the following folder.

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

C:\ICPDAS\PCIe-LM4

At least 2.6 MB of free disk space is required.

3. Click “Install” button to continue.



Setup - PCIe-LM4 Series Card Driver

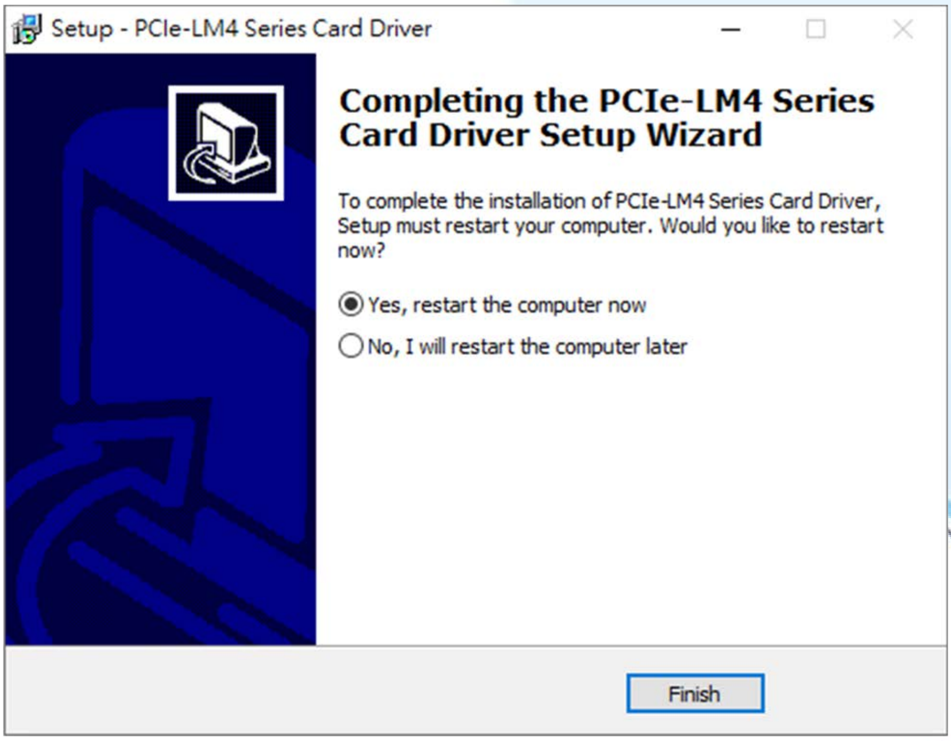
Ready to Install

Setup is now ready to begin installing PCIe-LM4 Series Card Driver on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

Destination location:
C:\ICPDAS\PCIe-LM4

4. Select the “Yes, restart the computer now” radio button. Ensure that any open programs are closed and you have saved your work, and then click the “Finish” button. The system will then reboot to complete the installation of the ICP DAS PCIELM4 Driver DLL.



Setup - PCIe-LM4 Series Card Driver

Completing the PCIe-LM4 Series Card Driver Setup Wizard

To complete the installation of PCIe-LM4 Series Card Driver, Setup must restart your computer. Would you like to restart now?

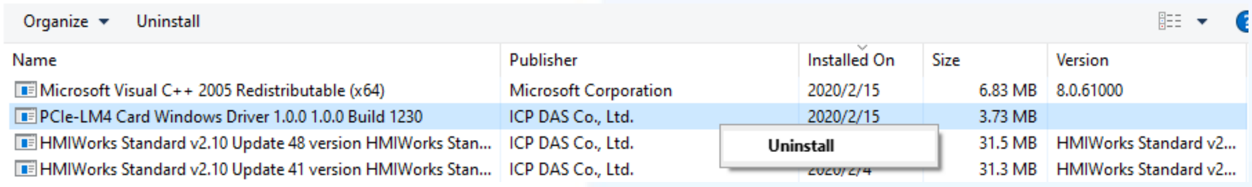
[✓ Yes, restart the computer now]
No, I will restart the computer later

2.3. Uninstalling the PCIELM4 Driver DLL

The ICP DAS PCIELM4 Driver DLL includes a utility that allows the software to be removed from your computer. To uninstall the software, follow the procedure described below:

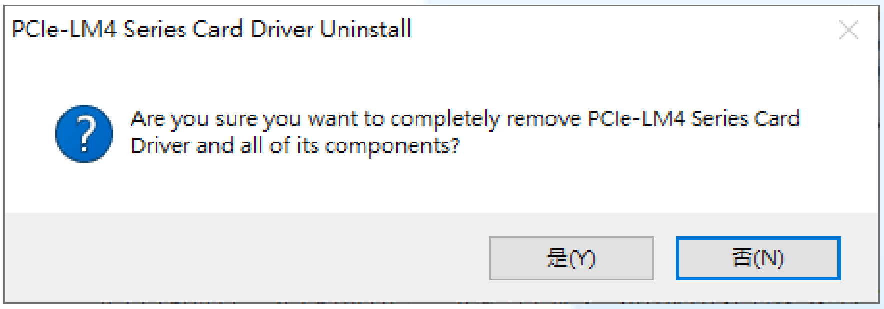
1. Open the Control Panel by clicking "Start" button and then clicking "Control Panel". Double-click the "Add/Remove Programs" icon to open "Add/Remove Programs" dialog.

2. In the "Add/Remove Programs" dialog, click the "Change or Remove Programs" tab, and then click the "PCIe-LM4 Card Windows Driver" item. Click the "Uninstall" button to begin the uninstall process.



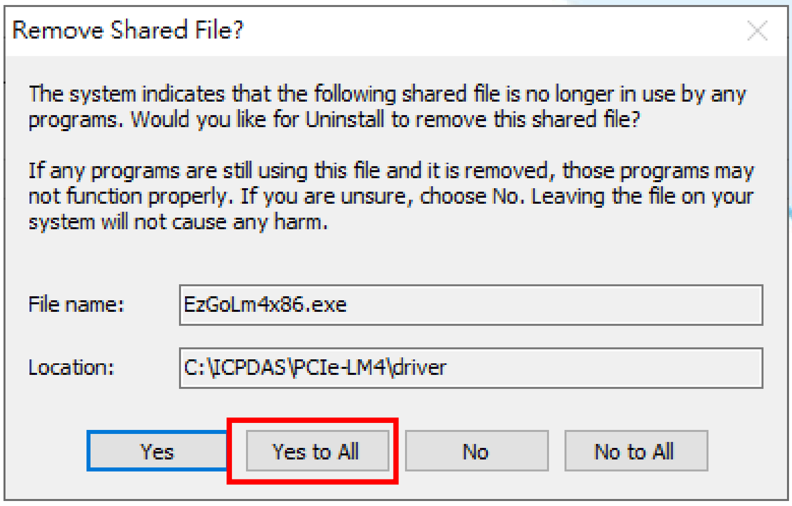
Name	Publisher	Installed On	Size	Version
Microsoft Visual C++ 2005 Redistributable (x64)	Microsoft Corporation	2020/2/15	6.83 MB	8.0.61000
PCIe-LM4 Card Windows Driver 1.0.0 1.0.0 Build 1230	ICP DAS Co., Ltd.	2020/2/15	3.73 MB	
HMIWorks Standard v2.10 Update 48 version HMIWorks Stan...	ICP DAS Co., Ltd.		31.5 MB	HMIWorks Standard v2...
HMIWorks Standard v2.10 Update 41 version HMIWorks Stan...	ICP DAS Co., Ltd.	2020/2/4	31.3 MB	HMIWorks Standard v2...

3. A prompt will be displayed asking you to confirm that you wish to remove the PCIELM4 Windows Driver. Click the "Yes" button to continue.



Are you sure you want to completely remove PCIe-LM4 Series Card Driver and all of its components?

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

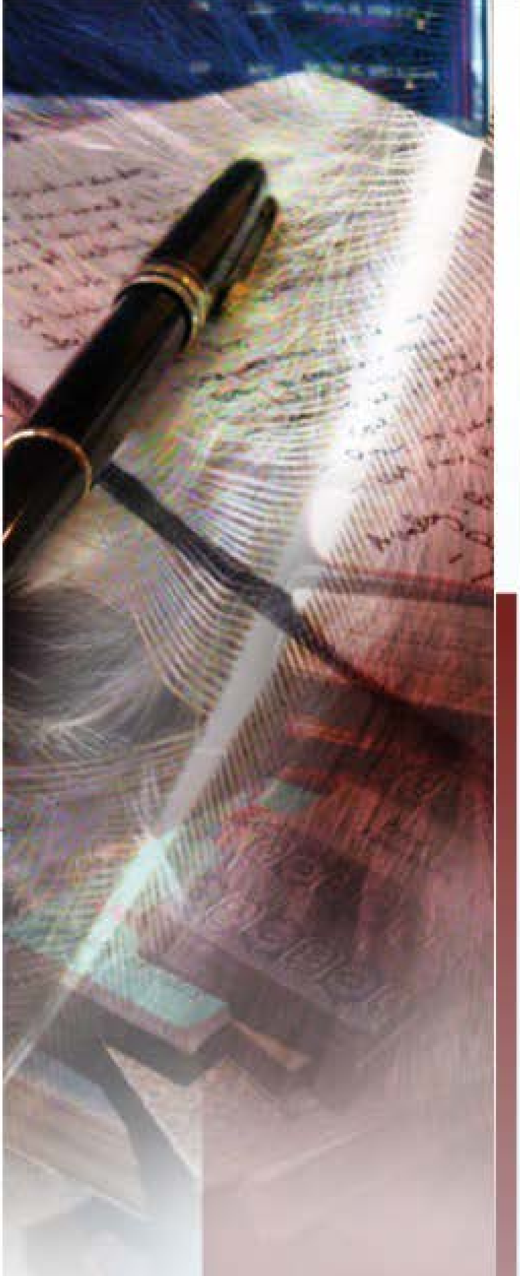


The system indicates that the following shared file is no longer in use by any programs. Would you like for Uninstall to remove this shared file?

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

File name: EzGoLm4x86.exe

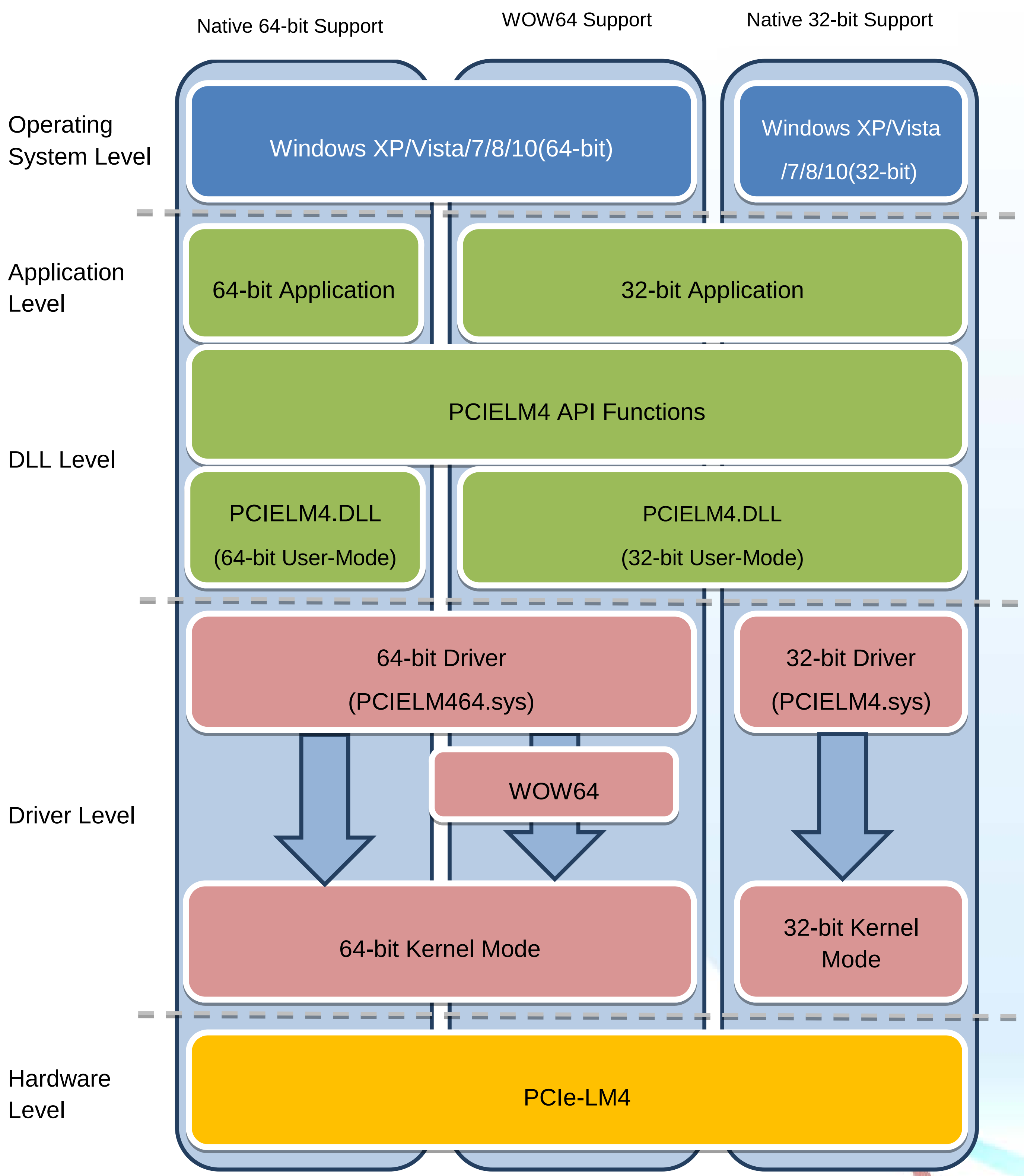
Location: C:\ICPDAS\PCIe-LM4\driver



3. Tutorial

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

3.1. Application Structure

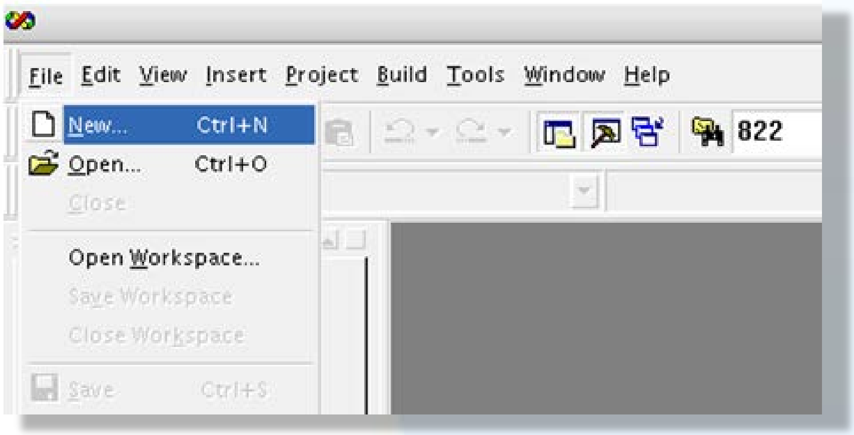


3.2. Creating a Win32 Console Application

The following procedure describes how to create a Win32 Console application based on the PCIELM4 DLL. Note that this description is based on Microsoft Visual Studio 6.0.

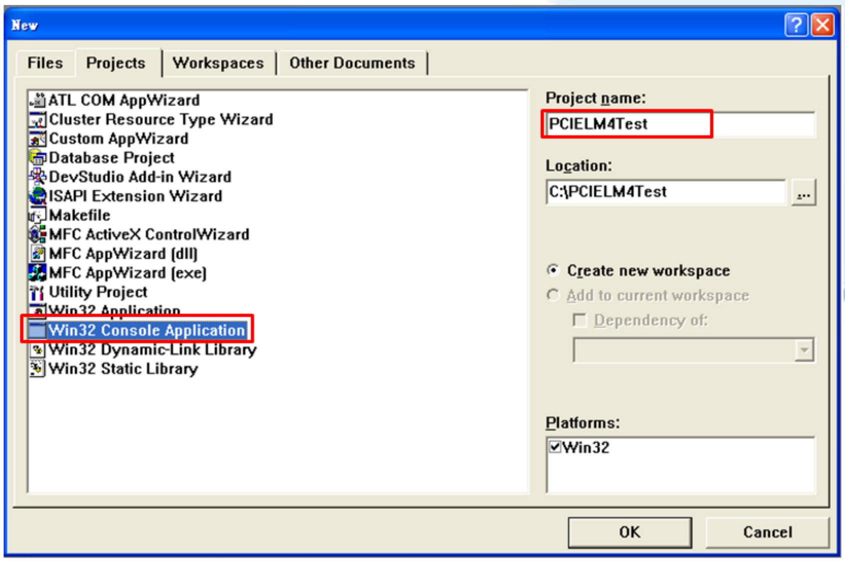
Creating the Application

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



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

   Click the "Win32 Console Application" entry in the Projects List pane, an and enter “PCIELM4Test” in the Project name field. The Location field indicates where the project files will be stored. Verify that the details are correct, and then click the "OK" button to continue.

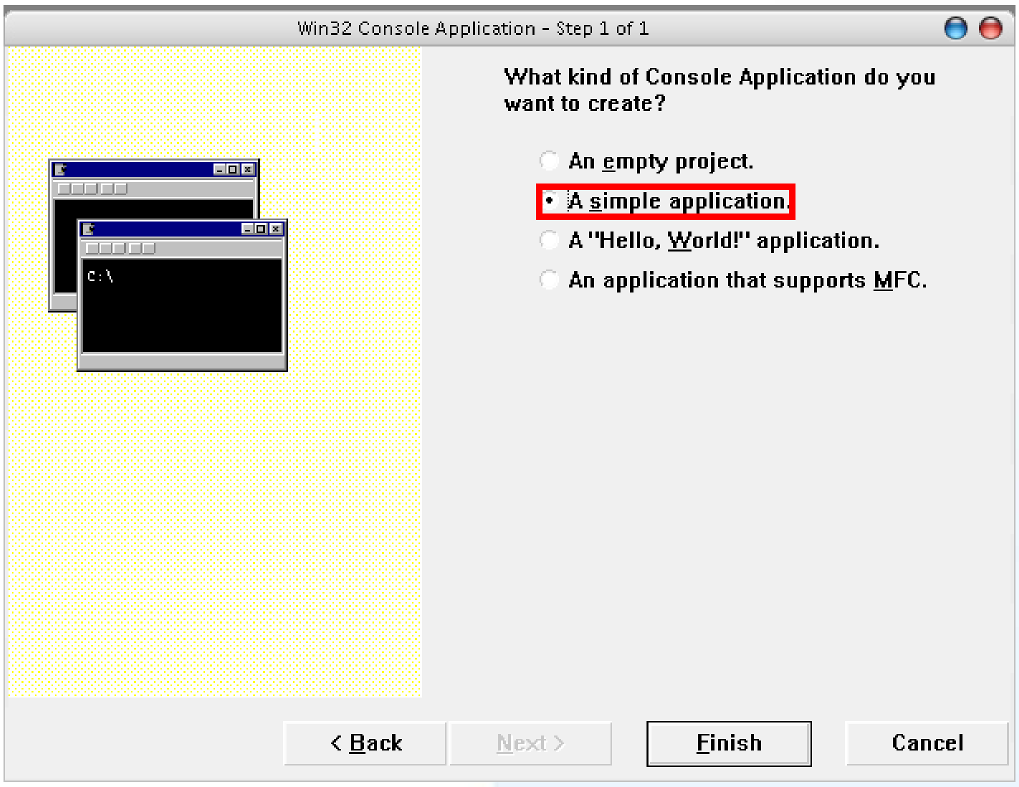


Project name: PCIELM4Test

Location: C:\PCIELM4Test

Platforms: ☑ Win32

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

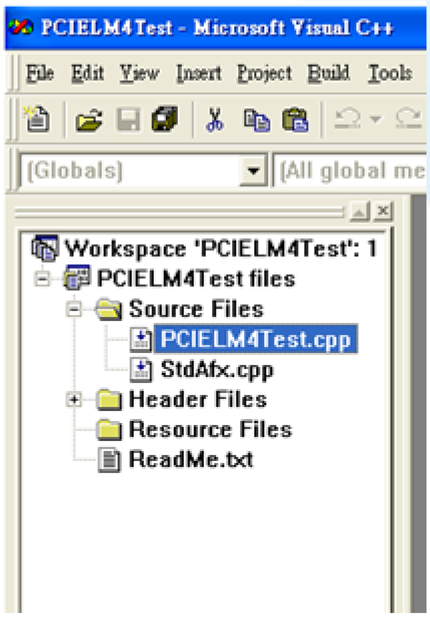


Win32 Console Application - Step 1 of 1

What kind of Console Application do you want to create?

- An empty project.
- [✓ A simple application.]
- A "Hello, World!" application.
- An application that supports MFC.

4. Once the project has been created, open the "Source Files" folder in the Navigation pane, and double-click the PCIELM4Test.cpp file to open the code editing window.



PCIELM4Test - Microsoft Visual C++

```
Workspace 'PCIELM4Test': 1
  PCIELM4Test files
    Source Files
      PCIELM4Test.cpp
      StdAfx.cpp
    Header Files
    Resource Files
    ReadMe.txt
```

5. Enter the following codes for the PCIELM4Test.cpp.

```
#include "stdafx.h"
#include "stdio.h"
#include "PCIELM4.h" //Include the PCIELM4 header file
#pragma comment(lib,"PCIELM4.lib") //Include the PCIELM4 library file

WORD wRtn;
WORD wBoardNo;
WORD wTotalBoards;

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

 //Initialize the resource and read total number of boards form driver
 wRtn= PCIELM4_DriverInit(&wTotalBoards);
 if (wRtn!=PCIELM4_NoErr)
 {
      printf("\nDriver Init Error(%d)",wRtn);
      return wRtn;
 }
 printf("Write the DO Value 0xFF");
 wBoardNo=0;
 wOutPortNo=0;
 //Write the DO
 wRtn = PCIELM4_WriteDO(wBoardNo, 0xFF);
 //Release the resources from driver
 wRtn = PCIELM4_DriverClose();
 return 0;
}
```

Testing the application

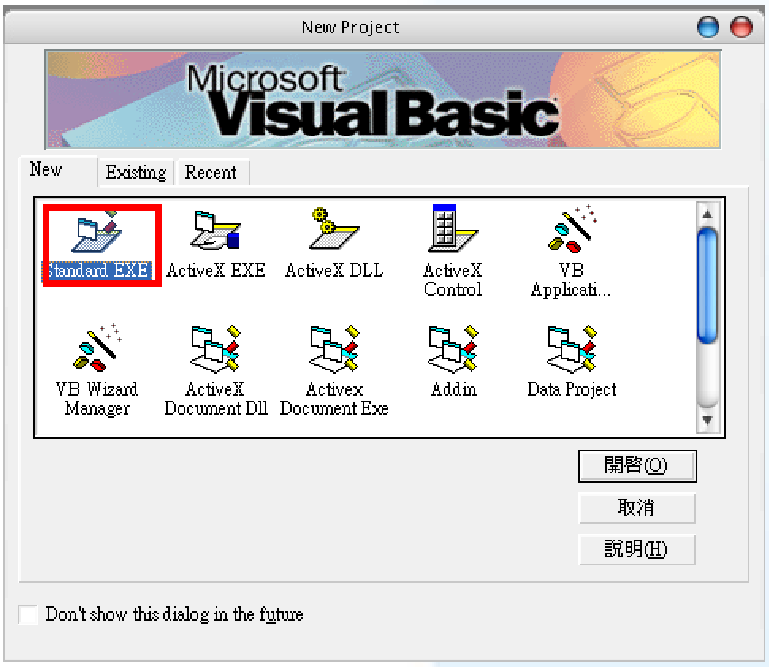
1. To compile your code, click Build from the main menu, and then click Compile, or press Ctrl + F7.
2. Execute the compiled application in a Command Prompt window.

3.3. Creating a Visual Basic Application

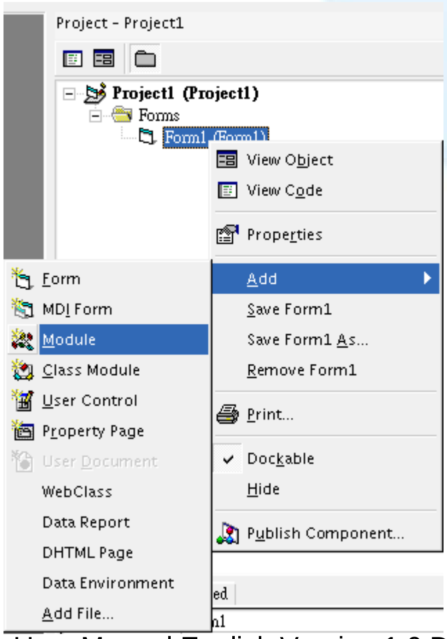
The following procedure describes how to create a Visual Basic application based on the PCIELM4 DLL. Note that this description is based on Microsoft Visual Studio 6.0.

Creating the Application

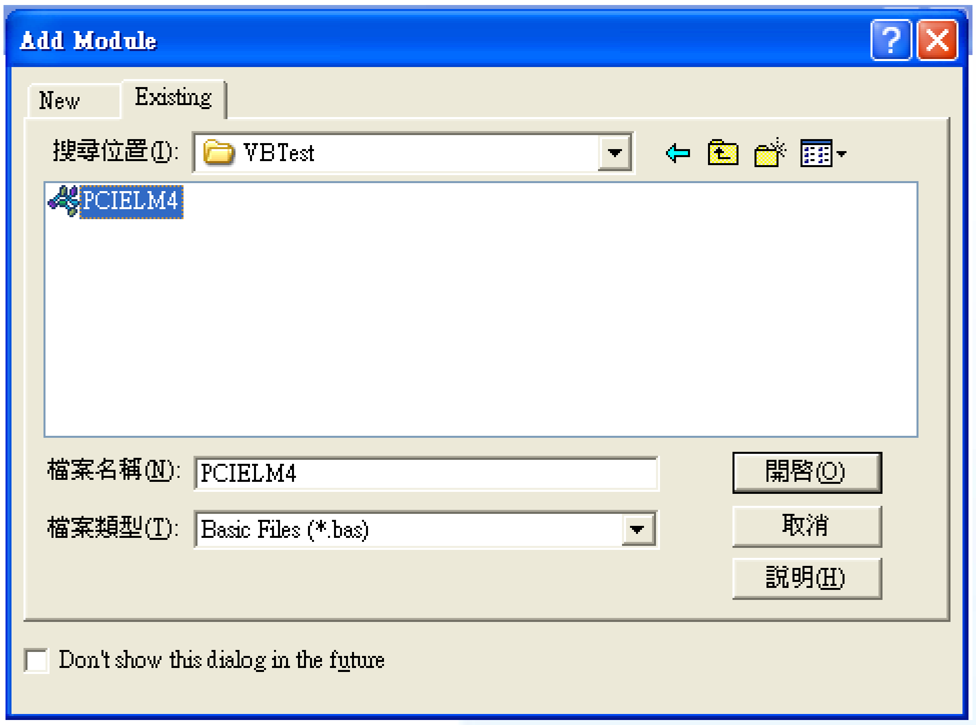
1. Open Microsoft Visual Studio to create a new Visual Basic project, and click the Standard.exe icon in the New Project window, and then click the Open button.



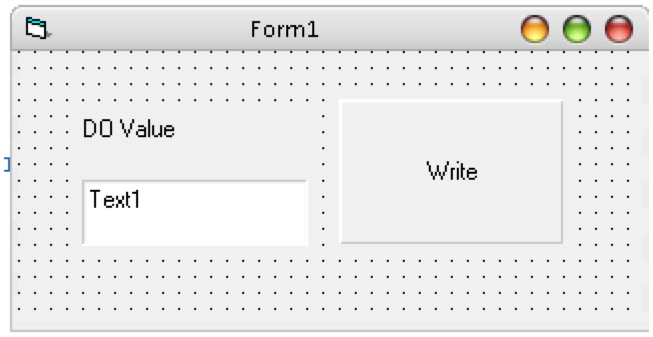
2. In the Project Explorer pane, right-click the name of the newly created form, point to Add in the menu, and then click Module to open the Add Module dialog box.



3. Add the PCIELM4.bas declaration file by clicking module by clicking on Add Module in the Project menu.



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



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

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

Private Sub cmdWrite_Click()

Dim wBoardIndex As Integer

‘//Initialize the resource and read total board number form driver
wRtn = PCIELM4_DriverInit(wTotalBoards)
If (wRtn) Then
    MsgBox ("Driver Initial Error.Error Code:" + Str(wRtn))
    End
End If

wBoardNo =0;
wOutportNo =0;

'//Write the DO Value
wRtn = PCIELM4_WriteDO(wBoardNo, Val(txtDOVal.Text))

‘//Release the resource form driver
wRtn = PCIELM4_DriverClose()
End Sub
```

Test the application

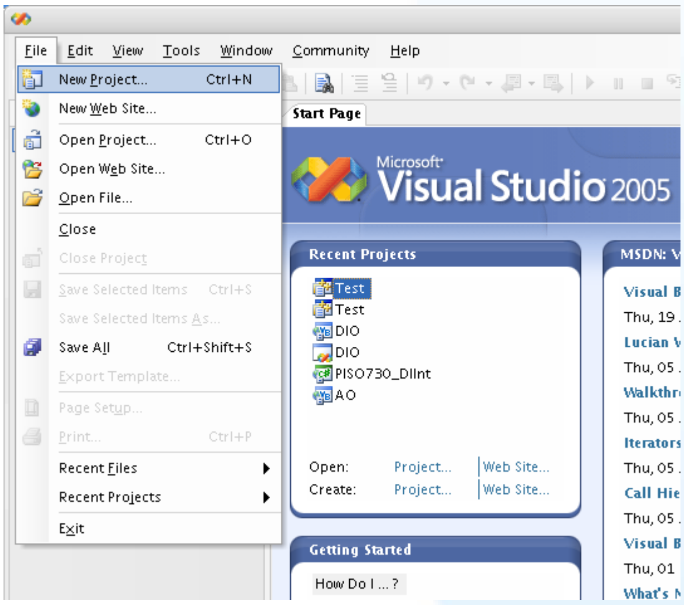
1. Run the application by either clicking the Start button on the toolbar, or by pressing F5.
2. Type "255" in the DO Value text box and then press the "Write" button to output a DO Value of 255.

3.4. Creating a Visual C++.NET Application

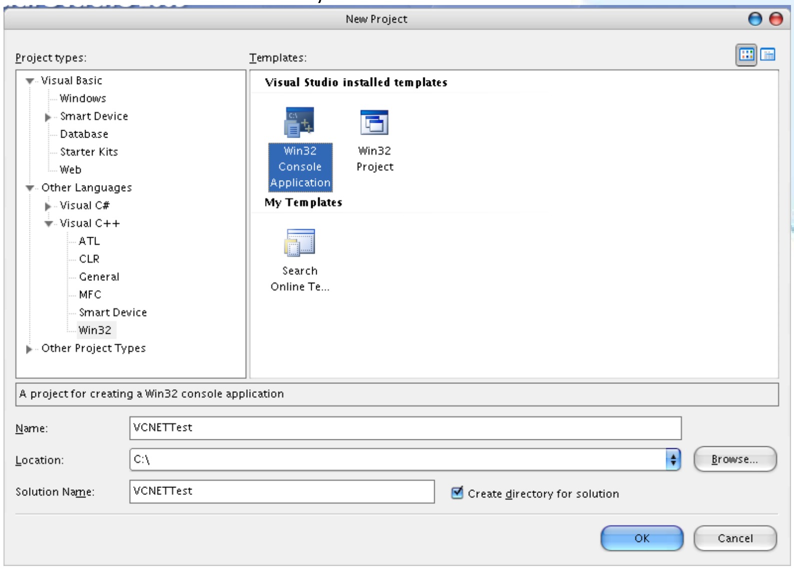
The following procedure describes how to create a Visual C++.NET application based on the PCIELM4 DLL. Note that this description is based on Microsoft Visual Studio 2005.

Creating the Application

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



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



A project for creating a Win32 console application

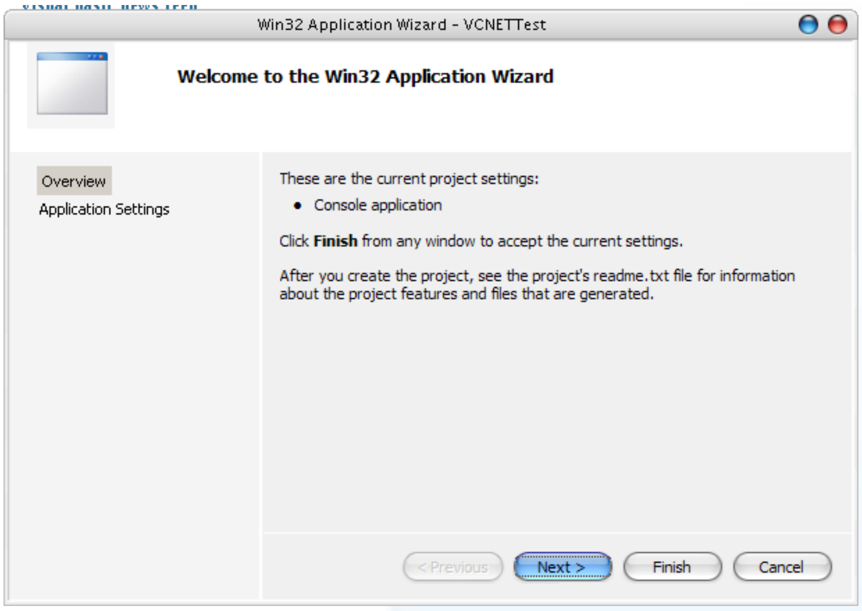
Name: VCNETTest

Location: C:\

Solution Name: VCNETTest

☑ Create directory for solution

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



Win32 Application Wizard - VCNETTest

Welcome to the Win32 Application Wizard

Overview
Application Settings

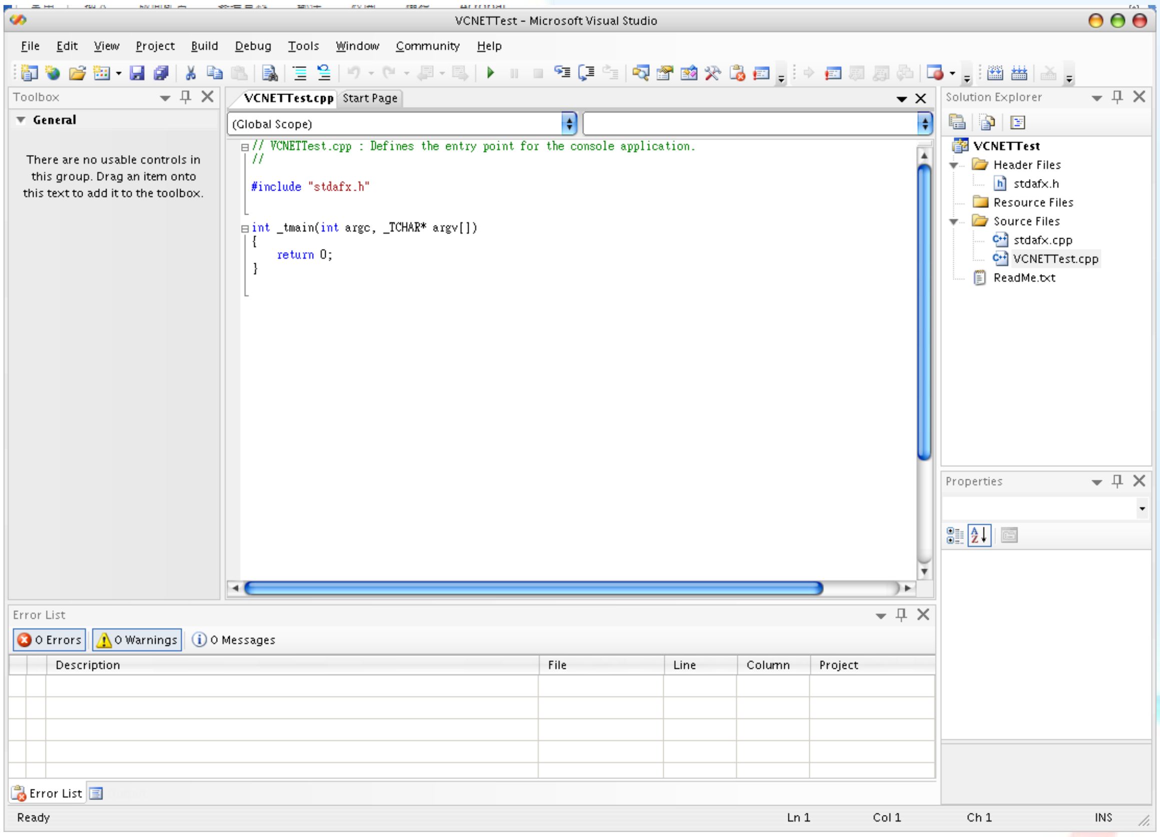
These are the current project settings:

- Console application

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

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

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



VCNETTest - Microsoft Visual Studio

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

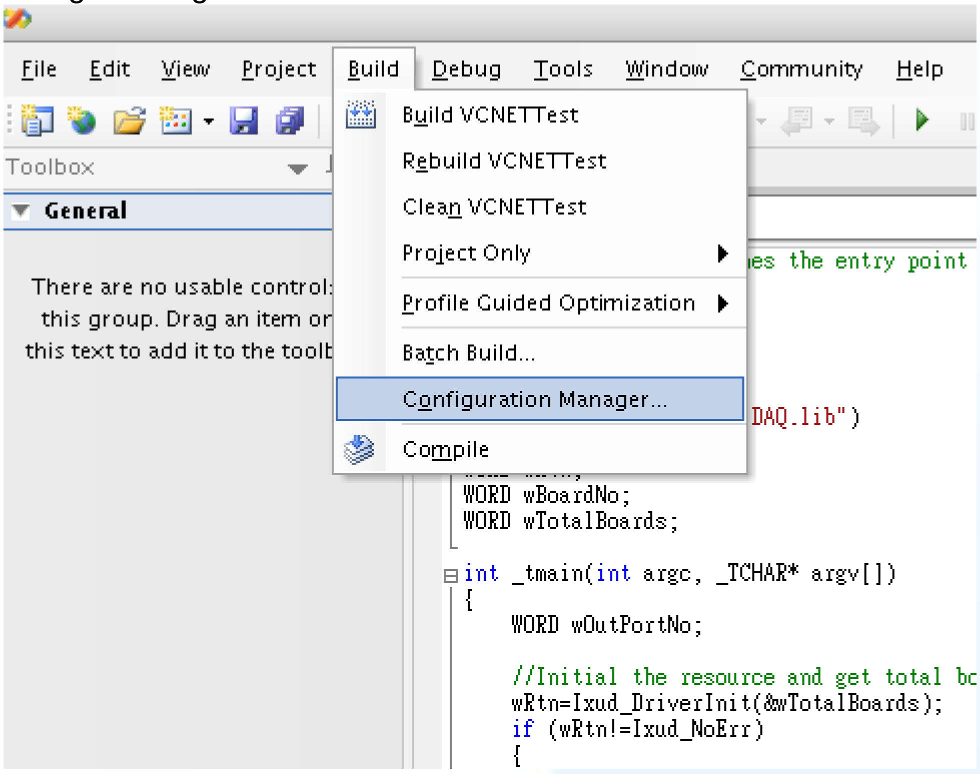
```
// VCNETTest.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"

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

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

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

#include "stdafx.h"
#include "stdio.h"
#include "PCIELM4.h"
#pragma comment(lib,"PCIELM4.lib")

WORD wRtn;
WORD wBoardNo;
WORD wTotalBoards;

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

 //Initialize the resources and read total number of boards form  driver
 wRtn=PCIELM4_DriverInit(&wTotalBoards);
 if (wRtn!=PCIELM4_NoErr)
 {
      printf("\nDriver Initialization Error.(%d)",wRtn);
      return wRtn;
 }
 printf("Write DO Value 0xFF");
 wBoardNo=0;
 wOutPortNo=0;
 //Write the DO value
 wRtn = PCIELM4_WriteDO(wBoardNo,0xFF);
 //Release the resources from driver
 wRtn = PCIELM4_DriverClose();
 return 0;
}
```

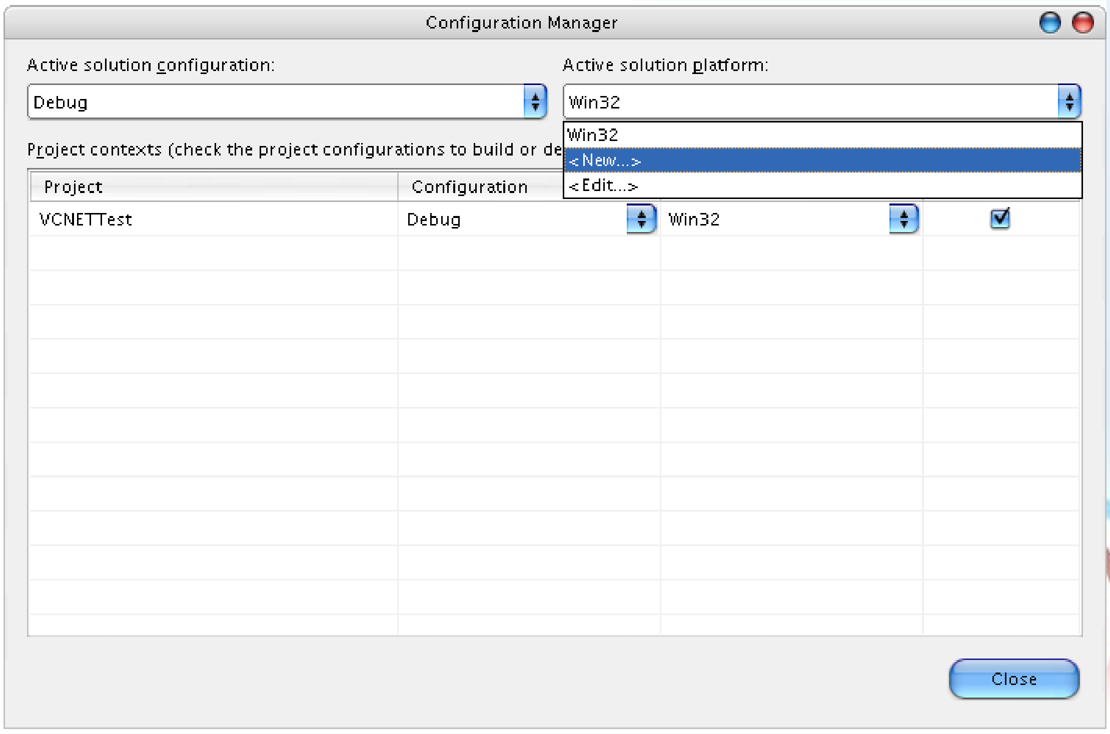
Compiling the Application

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



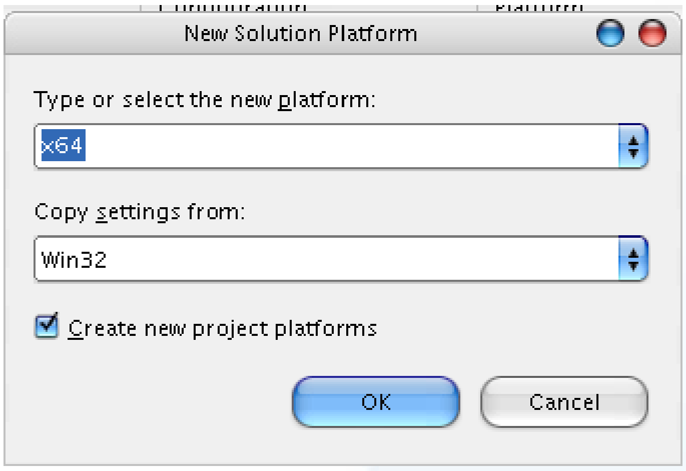
```
...es the entry point
...DAQ.lib")
WORD wBoardNo;
WORD wTotalBoards;

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

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

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


Configuration Manager
Active solution configuration:
Debug
Active solution platform:
Win32
Win32
<New...>
<Edit...>
Project contexts (check the project configurations to build or de
Close

Project	Configuration		
VCNETTest	Debug	Win32	✓

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

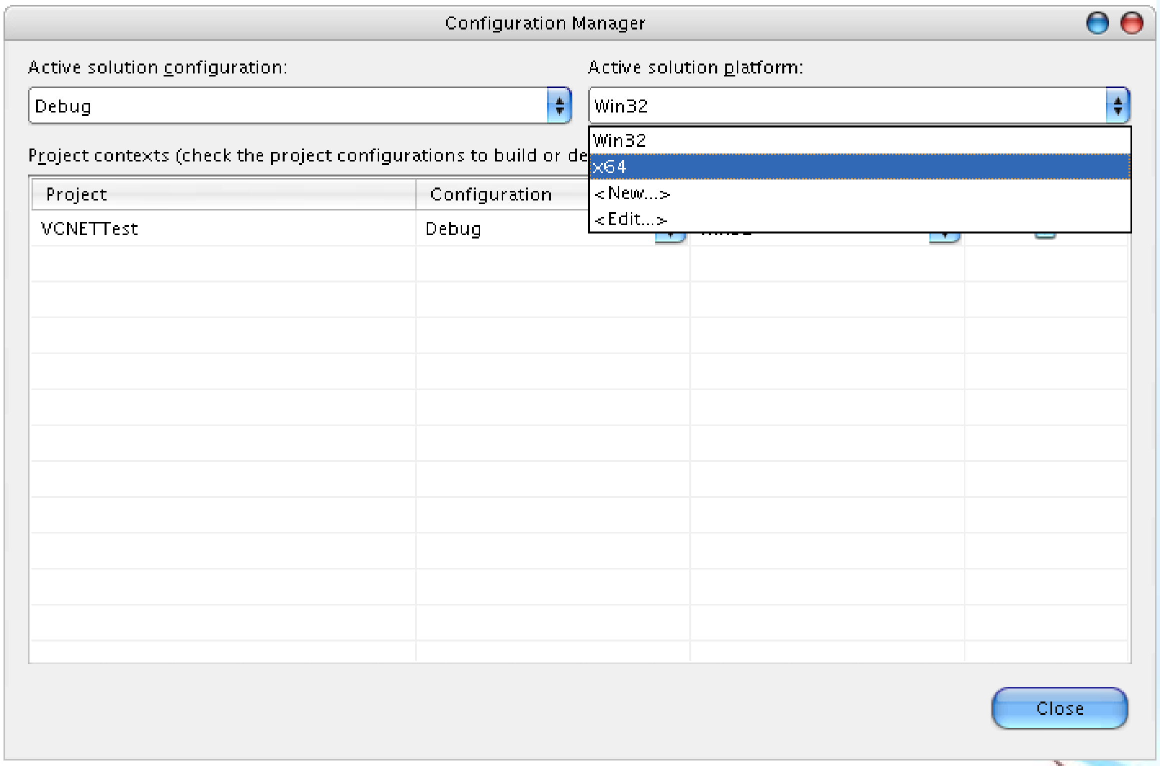


Type or select the new platform: x64

Copy settings from: Win32

☑ Create new project platforms

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



Active solution configuration: Debug

Active solution platform: Win32

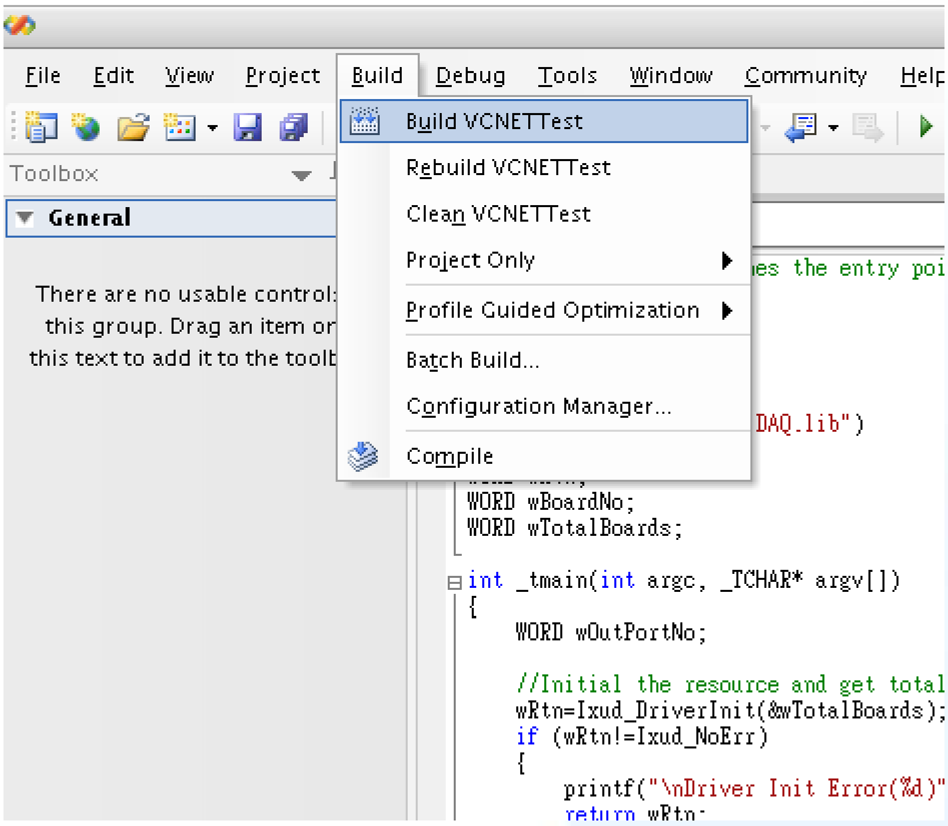
Project contexts (check the project configurations to build or de

Project	Configuration
VCNETTest	Debug

The 64-bit PCIELM4.lib file must be included for 64-bit applications

The 32-bit PCIELM4.lib file must be included for 32-bit applications

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



```
                                  es the entry poi

                                  DAQ.lib")

WORD wBoardNo;
WORD wTotalBoards;

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

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

Testing the Application

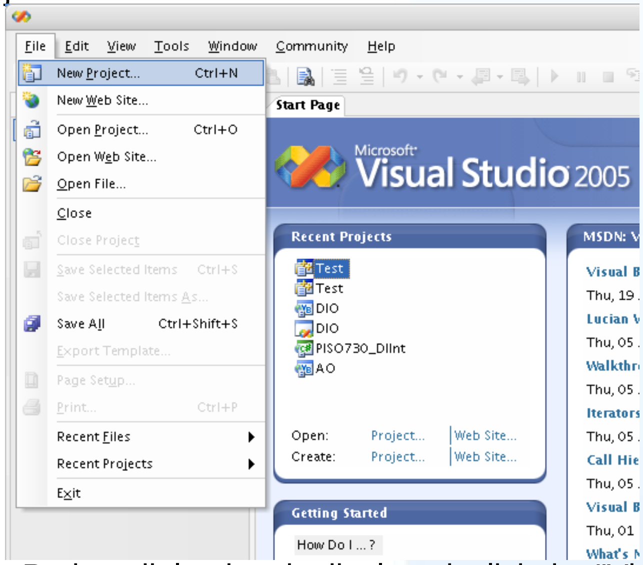
Execute the compiled application in a Command Prompt window.

3.5. Creating a Visual Basic.NET Application

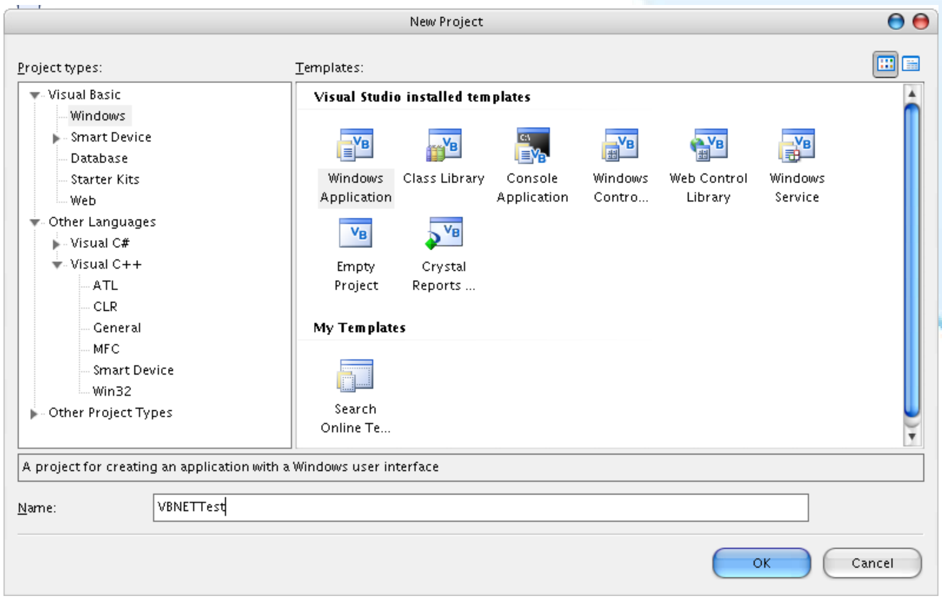
The following procedure describes how to create a Visual Basic.NET application based on the PCIELM4 DLL. Note that this description is based on Microsoft Visual Studio 2005.

Creating the Application

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



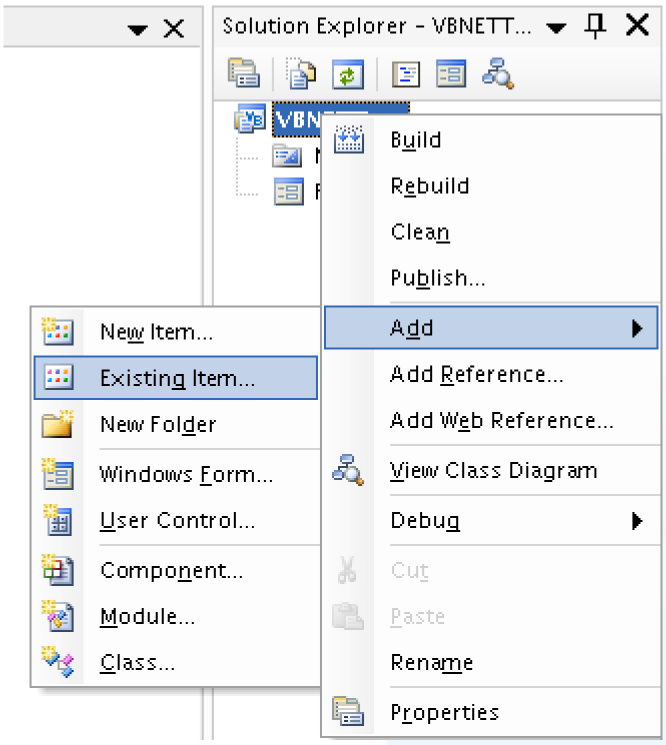
2. Once the New Project dialog box is displayed, click the "Visual Basic" item in the Project types pane, and then click the "Windows" option. In the Templates pane, click the Windows Application project template, enter “VBNETTest” in the Name field, and then click the OK button to create the new Visual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



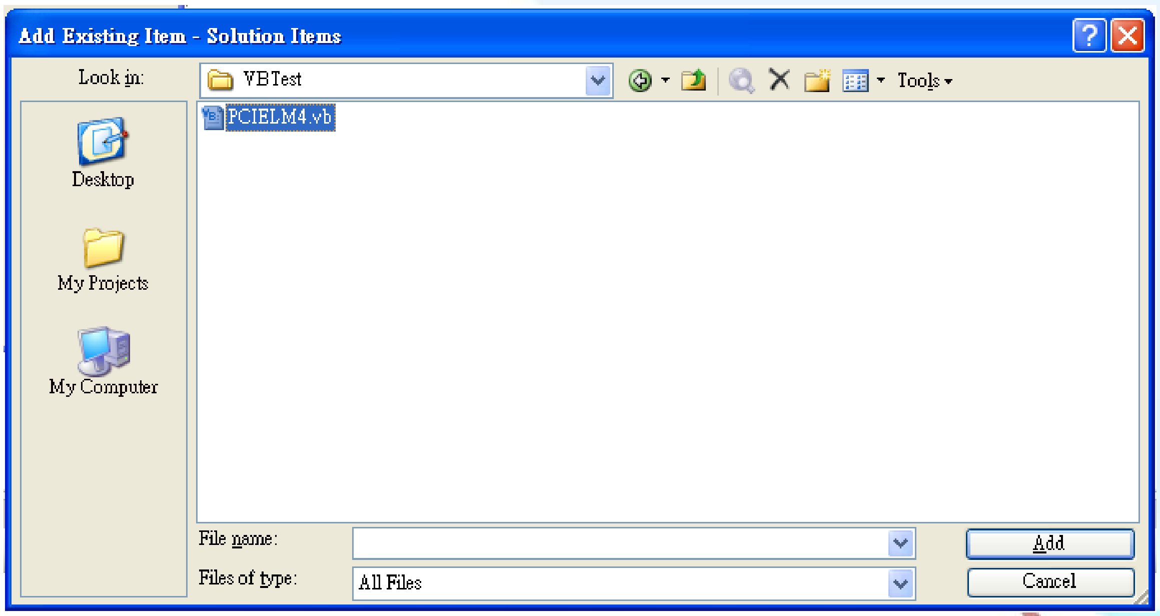
A project for creating an application with a Windows user interface

Name: VBNETTest

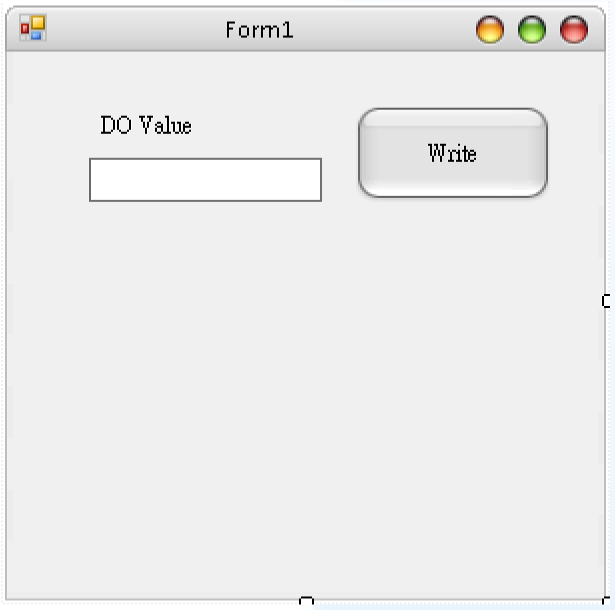
3. Once the project has been created, right-click the name of the newly created project in the Solutions Explorer pane, point to Add in the menu, and then click Existing Item option to open the Add Existing Item dialog box for the VBNETTest project.



4. Add the PCIELM4.vb declaration file by clicking the name of the file and then clicking the Add button.



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



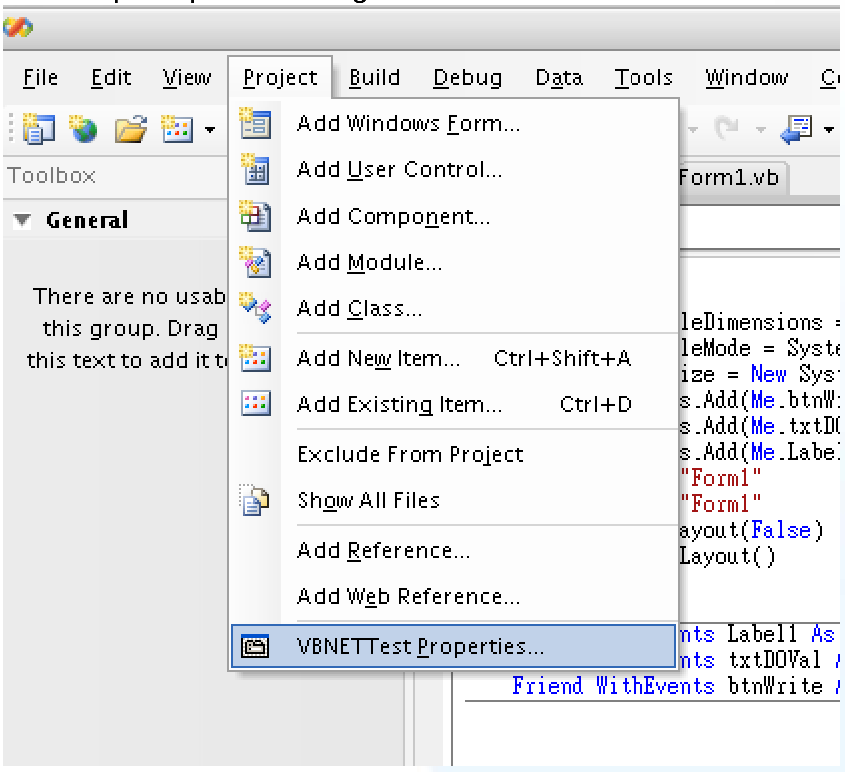
6. The btnWrite control on the Form to open the code editing window and then add the following code for the btnWrite button:

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

       '//Driver Initial
       wRtn = PCIELM4_DriverInit(wTotalBoards)
       If (wRtn) Then
           MsgBox("Driver Initial Error!!Error Code:" + Str(wRtn))
           End
       End If
```

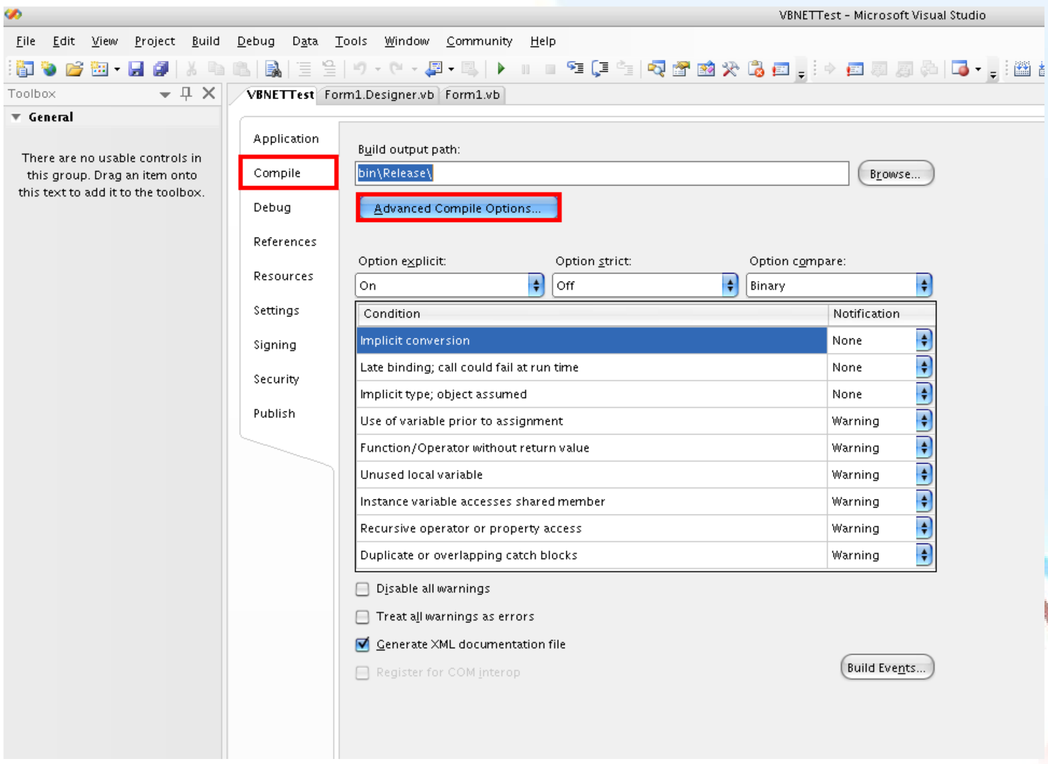
```
        '//Write DO
        wRtn = PCIELM4_WriteDO(wBoardNo, Val(txtDOVal.Text))

        wRtn = PCIELM4_DriverClose()
End Sub
```

Compiling the Application

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



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



Build output path: bin\Release\

Option explicit: On

Option strict: Off

Option compare: Binary

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

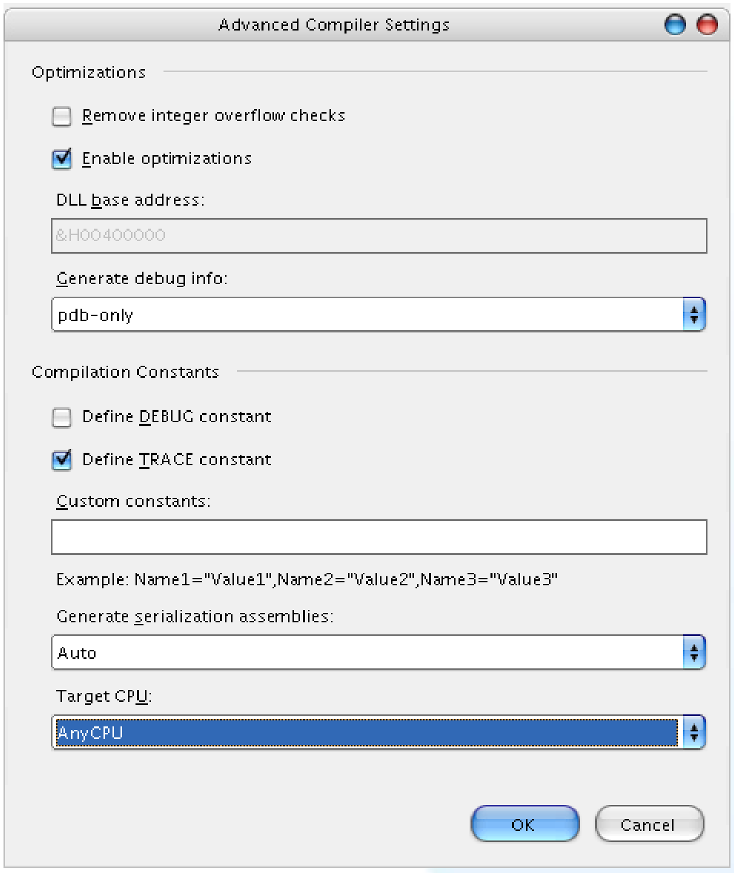
☐ Disable all warnings

☐ Treat all warnings as errors

☑ Generate XML documentation file

☐ Register for COM interop

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



☐ Remove integer overflow checks
☑ Enable optimizations
DLL base address: &H00400000
Generate debug info: pdb-only
☐ Define DEBUG constant
☑ Define TRACE constant
Custom constants:
Example: Name1="Value1",Name2="Value2",Name3="Value3"
Generate serialization assemblies: Auto
Target CPU: AnyCPU

An important note regarding the Target CPU options:

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

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

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

Testing the Application

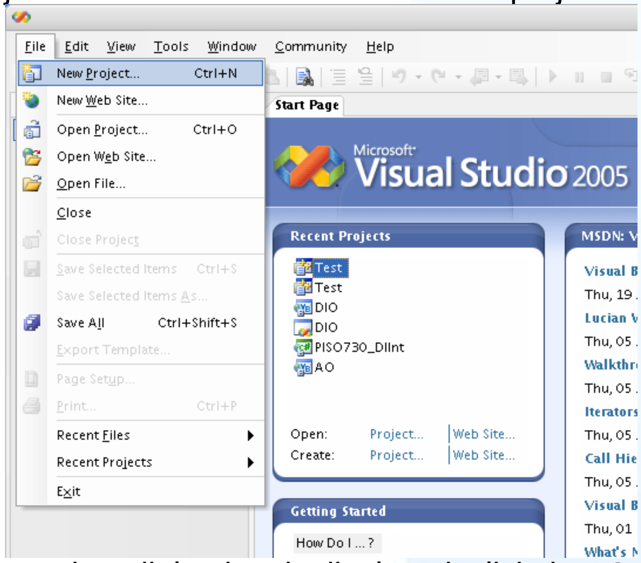
1. Run the application by either clicking the Start button on the toolbar, or by pressing F5.
2. Type "255" in the DO Value text box and then press the "Write" button to output a DO Value of 255.

3.6. Creating a Visual C#.NET Application

The following procedure describes how to create a Visual C#.NET application based on the PCIELM4 DLL. Note that this description is based on Microsoft Visual Studio 2005.

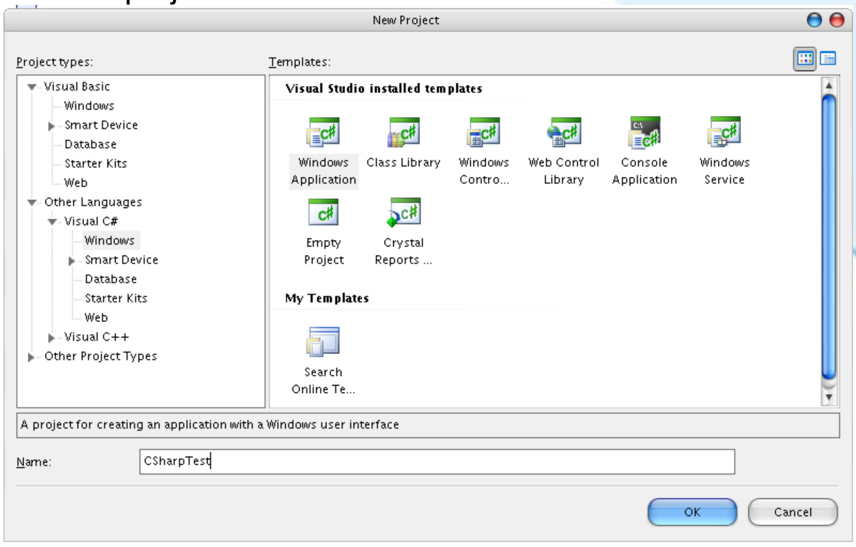
Creating the Application

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



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

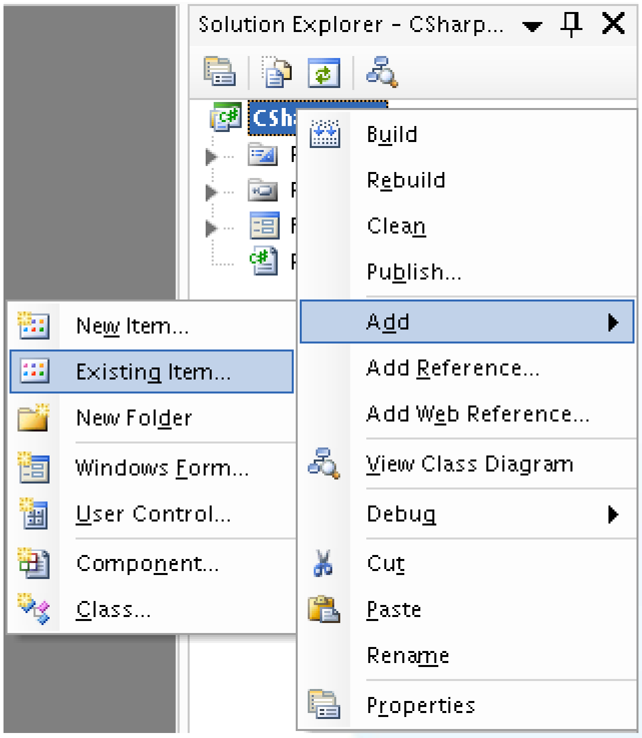
   In the Templates pane, click the Windows Application project template, enter “CSharpTest” in the Name field, and then click the OK button to create the new Visual C#.NET project.



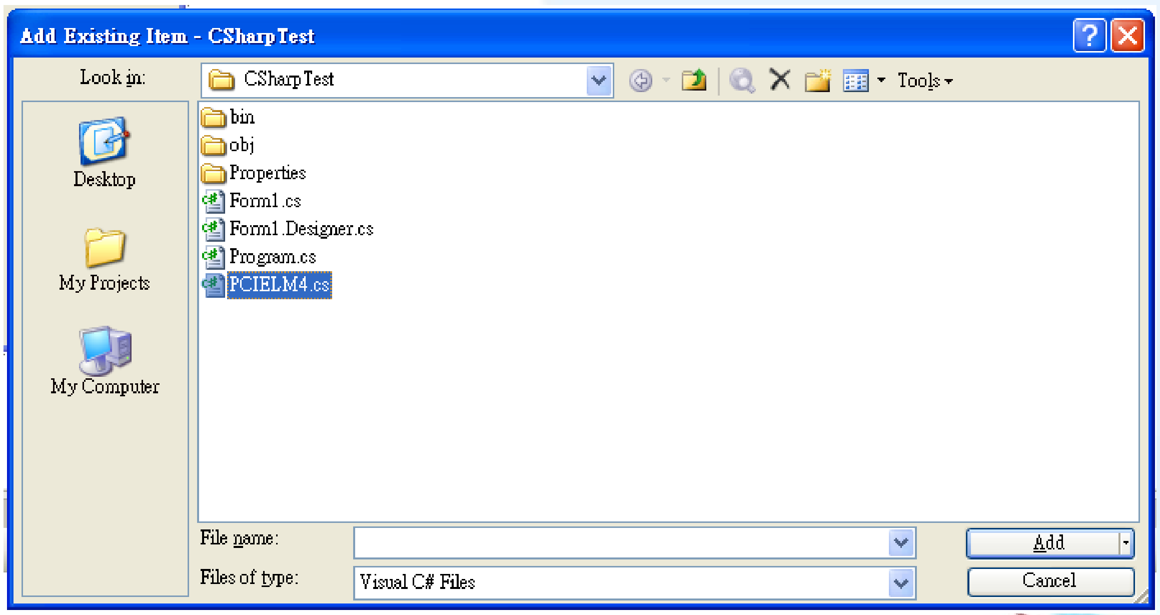
A project for creating an application with a Windows user interface

Name: CSharpTest

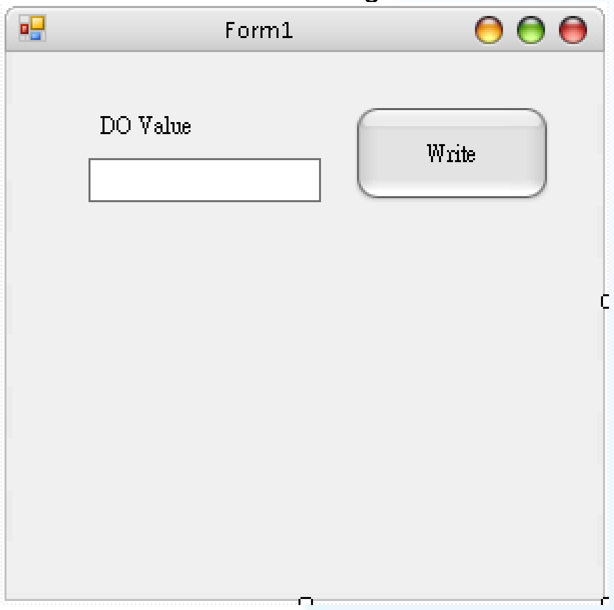
3. Once the project has been created, right-click the name of the newly created project in the Solutions Explorer pane, point to Add in the menu, and then click the Existing Item option to open the Add Existing Item dialog box for the CSharpTest project.



4. Add the PCIELM4.cs declaration file by clicking the name of the file and then clicking the Add button.



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

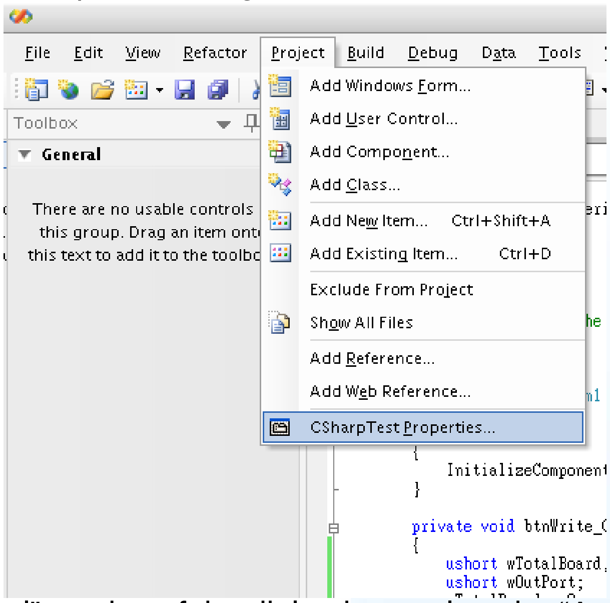


6. Double click the btnWrite control on the Form to open the code editing window and then add the following code to the Form.cs file:

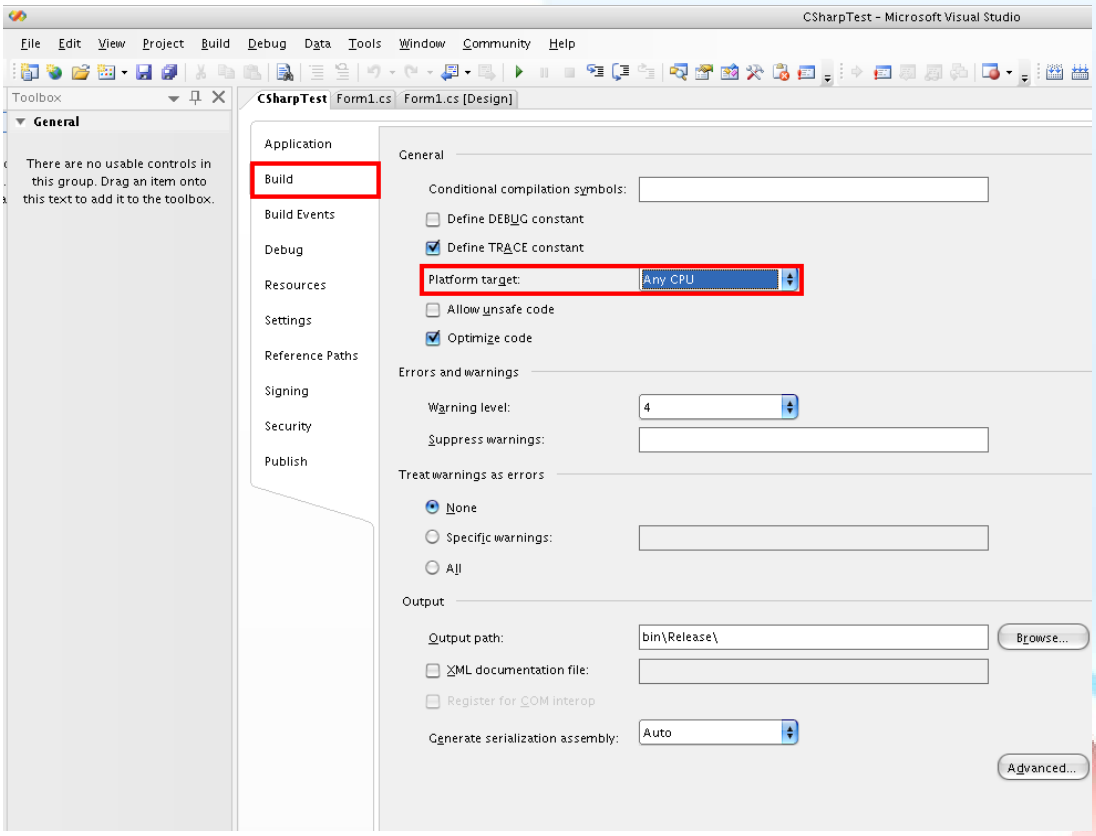
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using PCIELM4_Ns; //Include the PCIELM4 namespace

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

        private void btnWrite_Click(object sender, EventArgs e)
        {
            ushort wTotalBoard, wRtn, wBoardNo;
            ushort wOutPort;
            wTotalBoard = 0;
            //Initialize the resources and read the total number of boards form driver
            wRtn = PCIELM4_DLL.PCIELM4_DriverInit(ref wTotalBoard);
            if (wRtn != PCIELM4_DLL.PCIELM4_NoErr)
            {
                MessageBox.Show("Driver Initalization  Error.Error Code:" + 
wRtn.ToString());
                Close();
                return;
            }

            wBoardNo = 0;
            wOutPort = 0;
            //Write the DO Value
            wRtn = PCIELM4_DLL.PCIELM4_WriteDO(wBoardNo, 
Convert.ToUInt32(txtDOVal.Text));
            //Release the resources from the driver
            wRtn = PCIELM4_DLL.PCIELM4_DriverClose();

        }
    }
}
```

Compiling the Application

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


File Edit View Refactor Project Build Debug Data Tools
Toolbox
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Add Windows Form...
Add User Control...
Add Component...
Add Class...
Add New Item... Ctrl+Shift+A
Add Existing Item... Ctrl+D
Exclude From Project
Show All Files
Add Reference...
Add Web Reference...
CSharpTest Properties...

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


CSharpTest - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Window Community Help
Toolbox
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
CSharpTest Form1.cs Form1.cs [Design]
Application
Build
Build Events
Debug
Resources
Settings
Reference Paths
Signing
Security
Publish
Browse...
Advanced...

General

Conditional compilation symbols:

☐ Define DEBUG constant

☑ Define TRACE constant

Platform target: Any CPU

☐ Allow unsafe code

☑ Optimize code

Errors and warnings

Warning level: 4

Suppress warnings:

Treat warnings as errors

● None

○ Specific warnings:

○ All

Output

Output path: bin\Release\

☐ XML documentation file:

☐ Register for COM interop

Generate serialization assembly: Auto

An important note regarding the Platform target options:

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

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

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

Testing the application

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

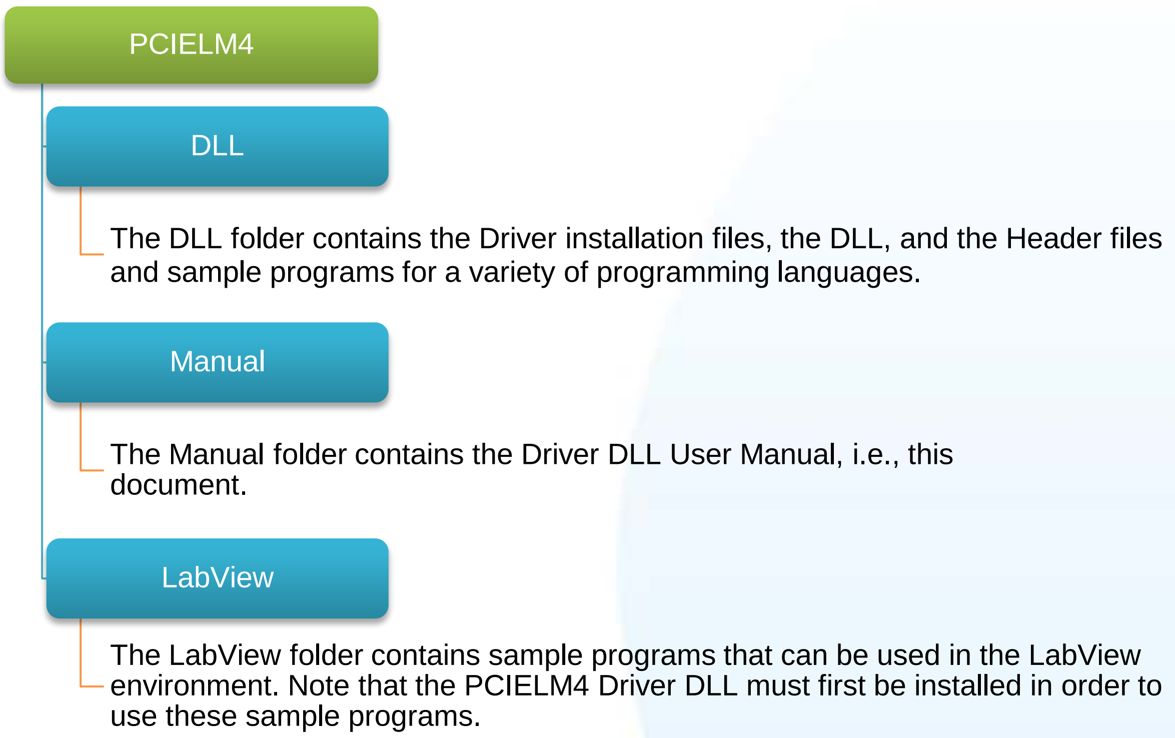
3.7. Sample Programs and Related Documents

In addition to the PCIELM4 Driver and DLL, ICP DAS provides a range of sample programs and source code that can be used in a Windows environment using a variety of programming languages, including Visual Basic, Visual C, Visual Basic.NET, and Visual C#.NET.

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

💿	CD:\\ NAPDOS\PCI\PCIe-LM4\
HTTP	http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pcie-lm4/
FTP	ftp://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pcie-lm4/

The PCIELM4 folder contains four sub-directories named DLL, Manual, LabView, and Matlab. An overview of the contents of each folder is given below.



The DLL folder contains the Driver installation files, the DLL, and the Header files and sample programs for a variety of programming languages.

The Manual folder contains the Driver DLL User Manual, i.e., this document.

The LabView folder contains sample programs that can be used in the LabView environment. Note that the PCIELM4 Driver DLL must first be installed in order to use these sample programs.

4. Function Overview

This chapter provides an overview of the hardware functions that can be programmed and controlled using the ICP DAS PCIELM4 Driver DLL

4.1. Introduction

ICP DAS PCIELM4 Driver DLL contains a set of functions that can be used in a wide variety of applications for PCIE-LM4 card. The API functions support a range of development environments and programming languages, including Microsoft Visual C++, Microsoft Visual Basic, Microsoft Visual C++.NET, Microsoft Visual C#.NET, and Microsoft Visual VB.NET.

Provides the following functions:

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

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

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

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

The PCIELM4 Driver DLL supports the following programming languages:

- Microsoft Visual C++ version 4.0 or later
- Microsoft Visual Basic version 4.0 or later
- Microsoft Visual C++.NET version 2003 or later
- Microsoft Visual C#.NET version 2003 or later
- Microsoft Visual Basic.NET version 2003 or later

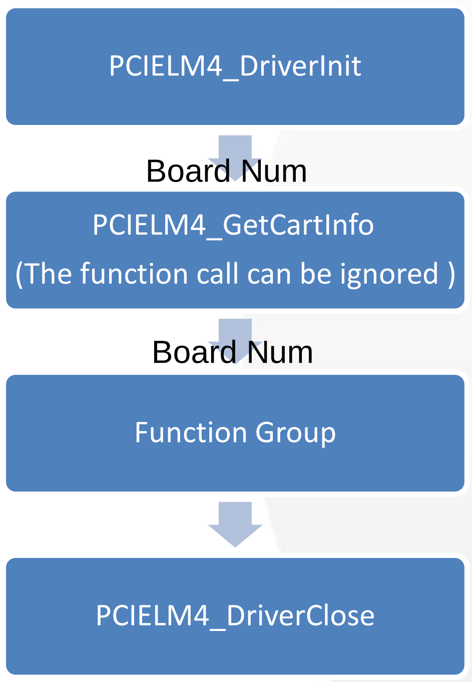
The following tables provide a summary of the function calls that can be accessed in custom applications using the PCIELM4 Driver, each of which will be described in more detail later in this manual.

Driver Functions	Digital I/O	Analog Input	Analog Output
PCIELM4_DriverInit	PCIELM4_ReadDI	PCIELM4_ConfigAI	PCIELM4_ConfigAO
PCIELM4_DriverClose	PCIELM4_WriteDO	PCIELM4_PollingAI	PCIELM4_WriteAOVoltage
PCIELM4_GetCardInfo		PCIELM4_PollingAIH	PCIELM4_WriteAOVoltageH
		PCIELM4_ConfigAIAutoZero	PCIELM4_StartAOVoltageALL
		PCIELM4_SaveAIAutoZeroVal	PCIELM4_StartAOVoltageALLH
		PCIELM4_AIHex2Vol	PCIELM4_StopAOALL

4.2. Driver Functions

The figure below provides an overview of the common call flow for the ICP DAS PCIELM4 Driver DLL

Call Flow



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

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

PCIELM4_DriverInit and PCIELM4_DriverClose

The PCIELM4_DriverInit function is used to allocate the resources for all boards installed in the system and to read the board number for each board. This function must be called when accessing the driver. The PCIELM4_DriverClose function is used to release the resources for board and must be called when ending access to the driver.

PCIELM4_GetCardInfo

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

4.3. Digital I/O

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

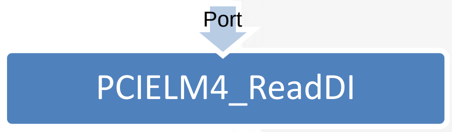
4.3.1. Digital Input

The Digital Input functions are used to perform Digital Input operations.

Software triggering

The PCIELM4_ReadDI function can be used to read the status information from a port.

Call Flow

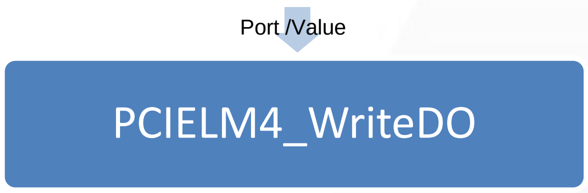


4.3.2. Digital Output

The digital output functions perform digital output operations.

User calls PCIELM4_WriteDO function to write a dword value to a port.

Call Flow



4.4. Analog Input

The analog input function group performs analog input functions. It can acquire multi-data in a single-channel.

Software Triggering

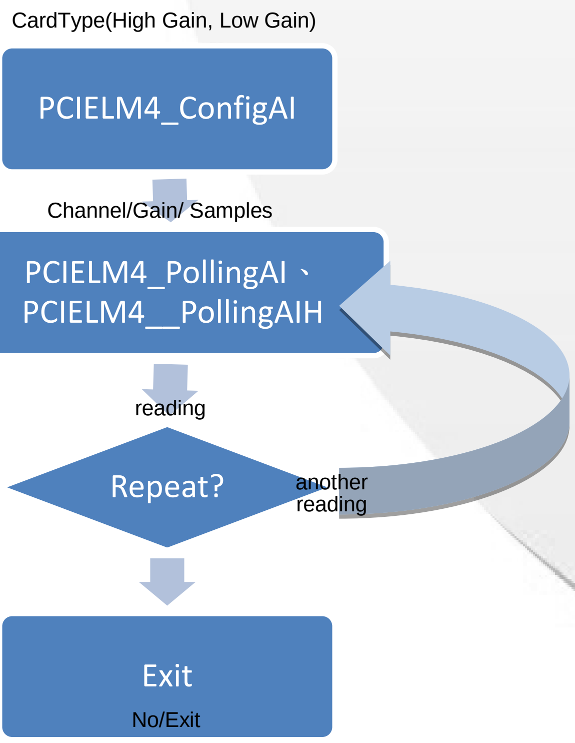
These functions trigger the data conversion by software. The PCIELM4 provides multiple points reading function.

> ⚠ 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)

- **Multiple Points Reading**

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

Call Flow

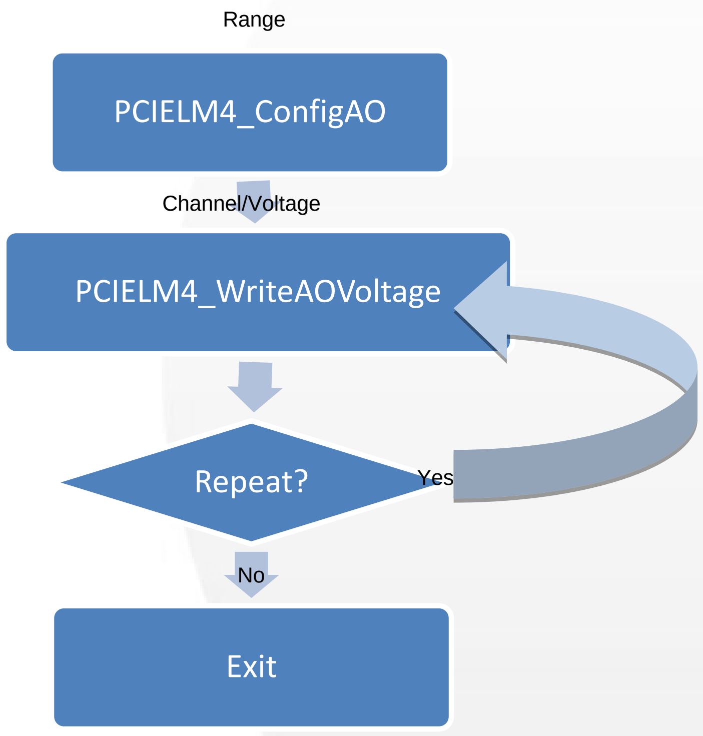


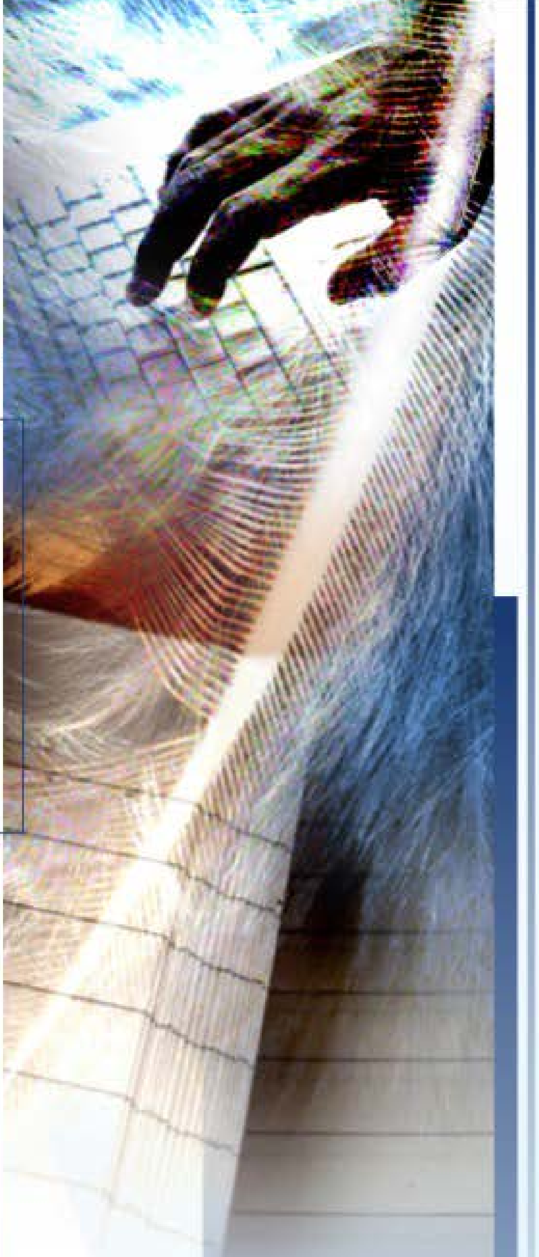
4.5. Analog Output

The analog output function group performs analog output functions.

Static voltage output

Call Flow





5. Function Reference

This chapter is a listing of all the functions and data structures that are supported by the ICP DAS PCIELM4 Driver DLL. It shows what functions are supported by each ICP DAS's product.

5.1. 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 PCIELM4 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

5.1.1. Driver Function Group

PCIELM4_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 PCIELM4_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.

PCIELM4_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 PCIELM4_DriverClose(

  void

);

- **Parameters**

None Parameters.

➢ **Return Value**

Refer to Appendix A.1. Return Value.

PCIELM4_GetCardInfo

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

➢ **Syntax**

WORD PCIELM4_GetCardInfo(

WORD wBoardNo**,**

PPCIELM4_DEVICE_INFO sDevInfo**,**

PPCIELM4_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 PPCIELM4_DEVICE_INFO.

sCardInfo

[Output] Retrieves the board hardware information. The data type is PPCIELM4_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.

5.1.2. Digital Input/Output Function Group

PCIELM4_ReadDI

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

- **Syntax**

```
WORD PCIELM4_ReadDI(
        WORD wBoardNo,
        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.

dwDIVal

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

- **Return Value**

Refer to Appendix A.1. Return Value.

PCIELM4_WriteDO

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

- **Syntax**

```
WORD PCIELM4_WriteDO(
        WORD wBoardNo,
        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.

dwDOVal

[Input] New digital logic state

- **Return Value**

Refer to Appendix A.1. Return Value.

5.1.3. Analog Input Function Group

PCIELM4_ConfigAI

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

- **Syntax**

```
WORD PCIELM4_ConfigAI(
        WORD wBoardNo,
        WORD wFIFOSizeKB,
        DWORD dwBufferSizeCount,
        WORD wCardType,
        WORD wDelaySettingTime,
        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] Reserved parameter.

dwBufferSizeCount

[Input] Reserved parameter.

wCardType

[Input] Reserved parameter.

wDelaySettingTime

[Input] Reserved parameter.

dwMode

[Input] Reserved parameter.

- **Return Value**

  Refer to Appendix A.1. Return Value.

PCIELM4_PollingAI

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

- **Syntax**

```
WORD PCIELM4_PollingAI(
        WORD wBoardNo,
        WORD wChannel,
        WORD wConfig,
        DWORD dwSampleRate,
        DWORD dwDataCount,
        double dValue[ ]
);
```

- **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.

dwSampleRate

[Input] Sampling rate in second. The dwSamplingRate parameter specifies the rate for sampling one data in Hz.

dwDataCount

[Input] The number of the sampled data.

dValue[ ]

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

➢ **Return Value**

Refer to Appendix A.1. Return Value.

PCIELM4_PollingAIH

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

➢ **Syntax**

```
WORD PCIELM4_PollingAIH(
        WORD wBoardNo,
        WORD wChannel,
        WORD wConfig,
        DWORD dwSampleRate,
        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.

dwSampleRate

[Input] Sampling rate in second. The dwSamplingRate parameter specifies the rate for sampling one data in Hz.

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.

PCIELM4_AIHex2Vol

Translates the un-scaled data to the double precision floating point type.

- **Syntax**

```
WORD PCIELM4_AIHex2Vol(
        WORD wBoardNo,
        WORD wChannel,
        WORD wConfig,
        DWORD dwValue,
        double *dVal
);
```

- **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

[Input] Please input an un-scaled data(DWORD).

dVal

[Output] The translated double precision floating point data from dwValue.

- **Return Value**

  Refer to Appendix A.1. Return Value.

PCIELM4_ConfigAIAutoZero

This function only supports the Load Cell channel. The user can enable or disable the calibration offset function.

- **Syntax**

```
WORD PCIELM4_ConfigAIAutoZero(
        WORD wBoardNo,
        WORD wChannel,
        WORD wEnableAutoZero
);
```

- **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.

wEnableAutoZero

[Input] The user can enable or disable the calibration offset function. (1=Enable, 0=Disable)

➢ **Return Value**

Refer to Appendix A.1. Return Value.

PCIELM4_SaveAIAutoZeroVal

This function only supports the Load Cell channel. The user can input an un-scaled data as the calibration offset value.

➢ **Syntax**

```
WORD PCIELM4_SaveAIAutoZeroVal(
        WORD wBoardNo,
        WORD wChannel,
        DWORD dwSampleRate,
        DWORD dwAutoZeroValue
);
```

➢ **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.

dwSamplingRate

[Input] Sampling rate in second. The fSamplingRate parameter specifies the rate for sampling one data in Hz.

dwAutoZeroValue

[Input] Inputs an un-scaled data(DWORD) as the calibration offset value.

➢ **Return Value**

Refer to Appendix A.1. Return Value.

5.1.4. Analog Output Function Group

PCIELM4_ConfigAO

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

- **Syntax**

```
WORD PCIELM4_ConfigAO(
            WORD wBoardNo,
            WORD wChannel,
            WORD wMode,
            WORD wAORange
);
```

- **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

dwMode

[Input] Reserved parameter.

wAORange

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

- **Return Value**

Refer to Appendix A.1. Return Value.

PCIELM4_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 PCIELM4_WriteAOVoltage(
            WORD wBoardNo,
            WORD wChannel,
            float fAOValue
);
```

- **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

fAOValue

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

- **Return Value**

Refer to Appendix A.1. Return Value.

PCIELM4_WriteAOVoltageH

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

- **Syntax**

```
WORD PCIELM4_WriteAOVoltageH(
        WORD wBoardNo,
        WORD wChannel,
        DWORD dwAOValue
);
```

- **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

dwAOValue

[Input] Binary data to be written

- **Return Value**

Refer to Appendix A.1. Return Value.

PCIELM4_StartAOVoltageALL

It initiates the fast analog output operations by specifying the output count, the data (floating-point voltage value) buffer and the cyclic mode.

➢ **Syntax**

```
WORD PCIELM4_StartAOVoltageALL(
        WORD wBoardNo,
        float fSamplingRate,
        DWORD dwDataCount,
        DWORD dwCycleNum,
        float fValueCH0[ ],
        float fValueCH1[ ]
);
```

➢ **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.

fSampleRate

[Input] Output rate in second. The fSamplingRate parameter specifies the rate for output one data in Hz.

dwDataCount

[Input] The numbers of data for outputting a waveform.

dwCycleNum

[Input] 0:Cyclic mode, the fast digital output operation will stop after user call PCIELM4_StopAOALL function.

fValueCH0[]

[Input] The fValueCH0[ ] to indicate the analog data buffer of the channel0 for floating-point voltage.

fValueCH1[]

[Input] The fValueCH1[ ] to indicate the analog data buffer of the channel1 for floating-point voltage.

- **Return Value**

Refer to Appendix A.1. Return Value.

PCIELM4_StartAOVoltageALLH

It initiates the fast analog output operations by specifying the output count, the data (binary voltage value) buffer and the cyclic mode.

- **Syntax**

```
WORD PCIELM4_StartAOVoltageALLH(
        WORD wBoardNo,
        float fSamplingRate,
        DWORD dwDataCount,
        DWORD dwCycleNum,
        DWORD dwValueCH0[ ],
        DWORD dwValueCH1[ ]
);
```

- **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.

fSampleRate

[Input] Output rate in second. The fSamplingRate parameter specifies the rate for output one data in Hz.

dwDataCount

[Input] The numbers of data for outputting a waveform.

dwCycleNum

[Input] 0:Cyclic mode, the fast digital output operation will stop after user call PCIELM4_StopAOALL function.

dwValueCH0[]

[Input] The dwValueCH0[ ] to indicate the analog data buffer of the channel0 for binary voltage.

dwValueCH1[]

[Input] The dwValueCH1[ ] to indicate the analog data buffer of the channel1 for binary voltage.

➢ **Return Value**

Refer to Appendix A.1. Return Value.

PCIELM4_StopAOALL

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

➢ **Syntax**

```
WORD PCIELM4_StopAOALL(
        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. Data Structure

PPCIELM4_DEVICE_INFO

➢ **Syntax**

```
typedef struct _PCIELM4_DEVICE_INFO_
{
    DWORD dwSize;
    WORD wVendorID;
    WORD wDeviceID;
    WORD wSubVendorID;
    WORD wSubDeviceID;
    DWORD dwBAR[6];
    UCHAR BusNo;
    UCHAR DevNo;
    UCHAR IRQ;
    UCHAR Aux;
    ULONGLONG dwBarVirtualAddress [6];
}PCIELM4_DEVICE_INFO,*PPCIELM4_DEVICE_INFO;
```

➢ **Member**

dwSize

[Output] Structure size returned, unit is byte.

wVendorID

[Output] Vendor ID returned.

wDeviceID

[Output] Device ID returned.

wSubVendorID

[Output] Sub Vendor ID returned.

wSubDeviceID

[Output] Get Sub Device ID.

dwBAR[]

[Output] Get Base Address。

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

BusNo

[Output] Bus number returned.

DevNo

[Output] Device number returned.

IRQ

[Output] IRQ number returned.

Aux

[Output] Aux ID returned.

dwBarVirtualAddress[]

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

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

PPCIELM4_CARD_INFO

- **Syntax**

```
typedef struct _PCIELM4_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];
}PCIELM4_CARD_INFO,*PPCIELM4_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. The Value is 2 at the PCIe-LM4. It means Differential(DIFF) type.

wAIOResolution

[Output] Reserved information. The AI resolution of the PCIe-LM4 is 24-bit, and the AO resolution of the PCIe-LM4 is 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. PCIe-LM4 is 16-bit。

wCounterChannels

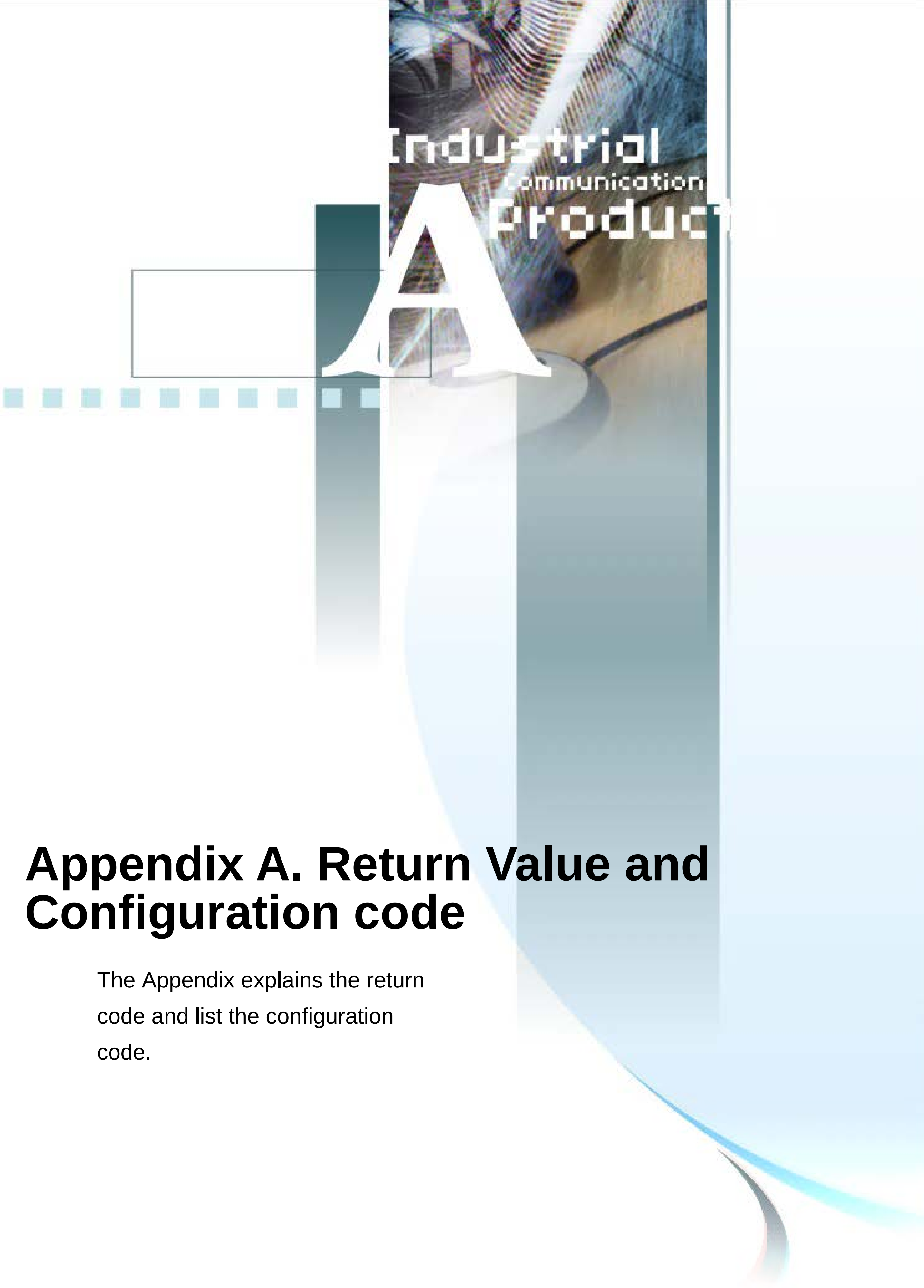
[Output] Number of counter returned.

wMemorySize

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

dwReserved1[]

[Output] Reserved information



Appendix A. Return Value and Configuration code

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

A.1. Return Value Definition

Explains the error code that might be returned when calling functions provide by the ICP DAS PCIELM4 Driver DLL. Refer to this section when debugging your application.

Return Value	Error ID	Description (Error Message)
0	PCIELM4_NoErr	Successfully
1	PCIELM4_OpenDriverErr	Open Driver Failure
2	PCIELM4_PnPDriverErr	Plug&Play Failure
3	PCIELM4_DriverNoOpen	Driver was not open.
4	PCIELM4_GetDriverVersionErr	Get Driver Version Failure
5	PCIELM4_ExceedBoardNumber	Board number error
6	PCIELM4_FindBoardErr	Cannot Find Board
7	PCIELM4_BoardMappingErr	Board Mapping Error
8	PCIELM4_DIOModesErr	Configure DIO Port Failure
9	PCIELM4_InvalidAddress	Invalid Address
10	PCIELM4_InvalidSize	Invalid Size
11	PCIELM4_InvalidPortNumber	Invalid Port Number
12	PCIELM4_UnSupportedModel	Model Is Not Supported
13	PCIELM4_UnSupportedFun	Function Is Not Supported
14	PCIELM4_InvalidChannelNumber	Invalid Channel Number
15	PCIELM4_InvalidValue	Invalid Value
16	PCIELM4_InvalidMode	Invalid Mode
17	PCIELM4_GetAIStatusTimeOut	Data Not Ready
18	PCIELM4_TimeOutErr	Timeout
19	PCIELM4_CfgCodeIndexErr	Cannot Find Configuration Code Index
20	PCIELM4_ADCCTLTimeoutErr	ADC Timeout
21	PCIELM4_FindPCIIndexErr	Cannot Find Board Index
22	PCIELM4_InvalidSetting	Invaild Setting
23	PCIELM4_AllocateMemErr	Allocate Memory Space Failed
24	PCIELM4_InstallEventErr	Install Interrupt Event Failure
25	PCIELM4_InstallIrqErr	Install Interrupt IRQ Failure
26	PCIELM4_RemoveIrqErr	Remove Interrupt IRQ Failure
27	PCIELM4_ClearIntCountErr	Clear Interrupt Count Failure
28	PCIELM4_GetSysBufferErr	Get System Buffer Failure
29	PCIELM4_CreateEventErr	Call CreateEvent() Failed
30	PCIELM4_UnSupportedResolution	Resolution IS Not Supported

31	PCIELM4_CreateThreadErr	Call CreateThread() Failed
32	PCIELM4_ThreadTimeOutErr	Thread Timeout
33	PCIELM4_FIFOOverFlowErr	FIFO Overflow
34	PCIELM4_FIFOTimeOutErr	FIFO Timeout
35	PCIELM4_GetIntInstStatus	Get Installing IRQ Status Failure
36	PCIELM4_GetBufStatus	Get System Buffer Status Failture
37	PCIELM4_SetBufCountErr	Buffer Size Setting Failure
38	PCIELM4_SetBufInfoErr	Buffer Setting Failure
39	PCIELM4_FindCardIDErr	Cannot Find Card ID
40	PCIELM4_EventThreadErr	Event Thread Failure
41	PCIELM4_AutoCreateEventErr	Cannot Call CreateEvent() Automatically
42	PCIELM4_RegThreadErr	Register Thread Failure
43	PCIELM4_SearchEventErr	Cannot Find Event
44	PCIELM4_FifoResetErr	Cannot Clear FIFO
45	PCIELM4_InvalidBlock	Invalid EEPROM Block
46	PCIELM4_InvalidAddr	Invalid EEPROM Address
47	PCIELM4_AcqireSpinLock	Acquire Spin Lock Failure
48	PCIELM4_ReleaseSpinLock	Release Spin Lock Failure
49	PCIELM4_SetControlErr	Analog Input Setting Error
50	PCIELM4_InvalidChannels	Invalid Channel
51	PCIELM4_SearchCardErr	Search Card Failure
52	PCIELM4_SetMapAddressErr	Set Address Mapping Failure
53	PCIELM4_ReleaseMapAddressErr	Release Address Mapping Failure
54	PCIELM4_InvalidOffset	Invalid Offset
55	PCIELM4_ShareHandleErr	Open Share Memory Failed
56	PCIELM4_InvalidDataCount	Invalid number of data
57	PCIELM4_WriteEEPErr	Write EEPROM Failed
58	PCIELM4_CardIOErr	Use CardIO error
59	PCIELM4_IOErr	Use MemoryIO error
60	PCIELM4_SetScanChannelErr	Set channel scan number error
61	PCIELM4_SetScanConfigErr	Set channel scan configuration error
62	PCIELM4_GetMMIOMapStatus	Get Memory Mapping IO Status error
63	PCIELM4_InvalidEEPCmd	Invalid EEPROM command
64	PCIELM4_CheckEEPCRCErr	Check EEPROM CRC error
65	PCIELM4_CtlEEPFail	Control EEPROM fail
66	PCIELM4_UnknownEEPErr	Unknown EEPROM error
67	PCIELM4_SetAIMuxErr	Set AI Multiplexer error
68	PCIELM4_SetAICONErr	Set AI CON error
69	PCIELM4_SetAIRATEErr	Set AI Rate error

70	PCIELM4_WriteAOBufErr	Write AO buffer error
71	PCIELM4_TimeOut1Err	Timeout 1
72	PCIELM4_TimeOut2Err	Timeout 2
73	PCIELM4_TimeOut3Err	Timeout 3
74	PCIELM4_TimeOut4Err	Timeout 4
75	PCIELM4_TimeOut5Err	Timeout 5
76	PCIELM4_TimeOut6Err	Timeout 6
77	PCIELM4_SetAIUnknowErr	Set AI unknown error

A.2. Model number

ID	Value(HEX)	Supported DAQ board
PCIELM4	357701C4	PCIe-LM4

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	PCIELM4_AI_BI_227MV	Bipolar	+/- 227 mV
0	PCIELM4_AI_BI_10V	Bipolar	+/- 10V
1	PCIELM4_AI_BI_5V	Bipolar	+/- 5V
2	PCIELM4_AI_BI_2V5	Bipolar	+/- 2.5V
3	PCIELM4_AI_BI_1V25	Bipolar	+/- 1.25V

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.

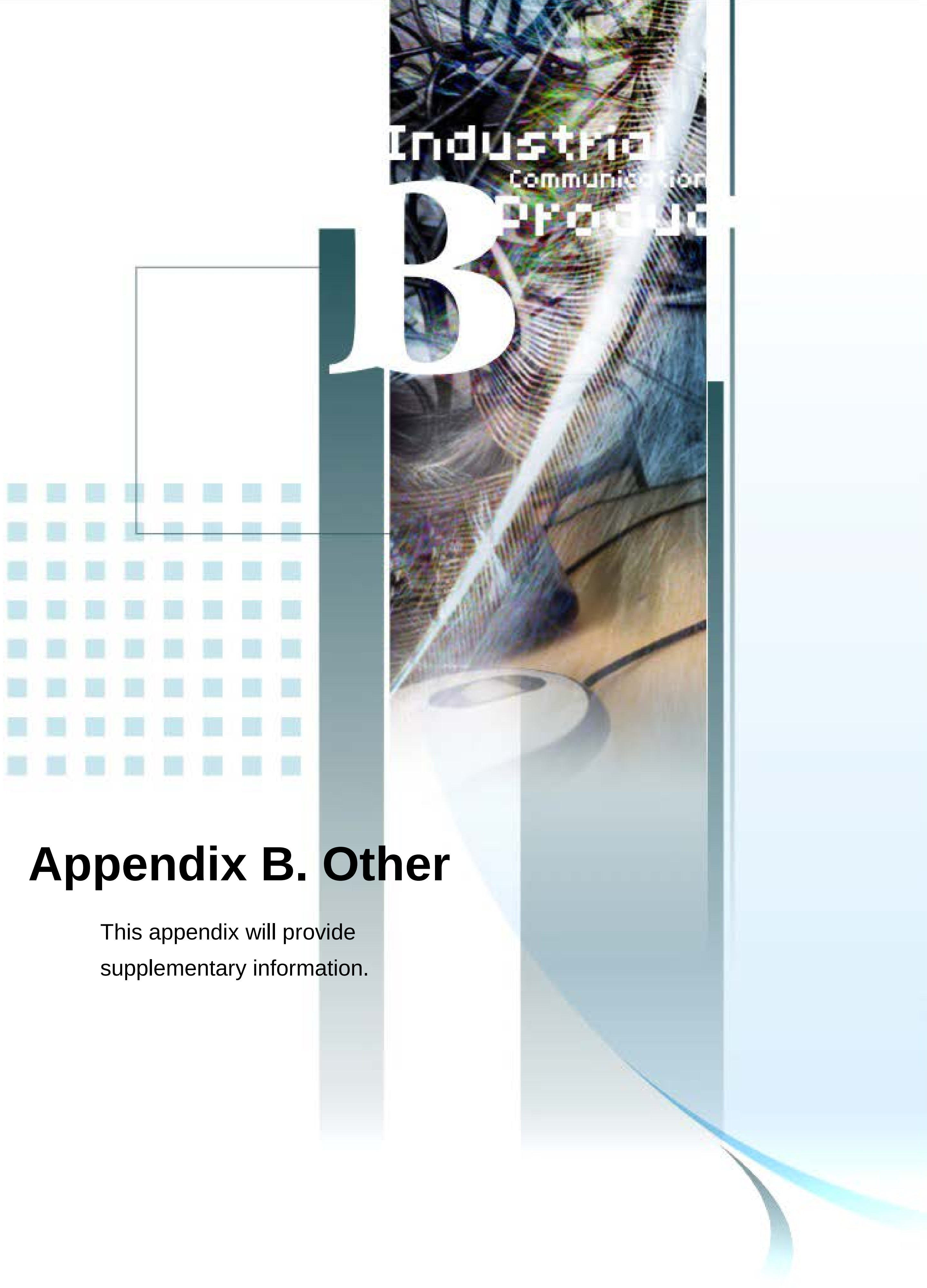
Code	ID	Voltage Range
0	PCIELM4_AO_UNI_5V	0 ~ 5V
1	PCIELM4_AO_UNI_10V	0 ~ 10V
2	PCIELM4_AO_BI_5V	+/- 5V
3	PCIELM4_AO_BI_10V	+/- 10V

A.4. DI Port Number Definition

DI Port No.	PCIe-LM4
0	DI 0 ~ 15

A.5. DO Port Number Definition

D0 Port No.	PCIe-LM4
0	D0 0 ~ 15



Appendix B. Other

This appendix will provide supplementary information.

B.1. FAQ

System and Install

Q. Does PCIELM4 supports 64-bit Windows?

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

Q. Does PCIELM4 support the ISA bus board?

A. PCIELM4 doesn’t support the ISA bus board.

Analog Outupt

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

A. Please check your analog output range setting, it must call the PCIELM4_ConfigAO function to set the correct range and then call the PCIELM4_WriteAOVoltage function to output voltage.

Troubleshooting for function return code

Q. Error code 1.

A. Please reinstall the PCIELM4 driver or reboot the PC.

Q. Error code 2.

A. (1) Please call the PCIELM4_DriverInit function to initial the PCIELM4 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	Feb. 2020	Initial issue