XPAC/WinPAC/ViewPAC
Standard API User Manual

(For WinCE-Based Platform)

Version 1.3.2, Sep. 2024

Service and usage information for

XPAC-8000 **WinPAC-8000** **ViewPAC-2000** **WinPAC-5000**

Written by Sean

Edited by Anna Huang

Warranty

All products manufactured by ICP DAS are under warranty regarding defective materials for a period of one year, beginning from the date of delivery to the original purchaser.

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.

Contact US

If you have any problem, please feel free to contact us.
You can count on us for quick response.

Email: service@icpdas.com

Contents

1. About this Guide

This manual is intended for software developers who want to integrate the functionality of standard PAC family into their applications.

What Models are covered in this Manual?

The following PAC models are covered in this manual:

WinPAC family for Arm AM335x platform series

WP-9000 series (Metal Case)	
WP-9221-CE7	WinCE 7.0 Based Standard WinPAC with 1 I/O slot
WP-9421-CE7	WinCE 7.0 Based Standard WinPAC with 4 I/O slots
WP-9821-CE7	WinCE 7.0 Based Standard WinPAC with 8 I/O slots
WP-8000 series	
WP-8121-CE7	WinCE 7.0 Based Standard WinPAC with 1 I/O slot
WP-8421-CE7	WinCE 7.0 Based Standard WinPAC with 4 I/O slots
WP-8821-CE7	WinCE 7.0 Based Standard WinPAC with 8 I/O slots
WP-5000 series	
WP-5231-CE7	WinCE 7.0 Based palm-sized WinPAC
WP-5231M-CE7	WinCE 7.0 Based palm-sized WinPAC with Metal Case
WP-5231PM-3GWA-CE7	WinCE 7.0 Based palm-sized WinPAC with 3G modem
WP-5231PM-4GE-CE7 WP-5231PM-4GC-CE7	WinCE 7.0 Based palm-sized WinPAC with 4G modem

WinPAC family for Arm AM335x platform series

ViewPAC series	
VP-1231-CE7	WinCE 7.0 Based 5.7" ViewPAC with 3 I/O Slots
VP-4231-CE7	WinCE 7.0 Based 10.4" ViewPAC with 3 I/O Slots
VP-6231-CE7	WinCE 7.0 Based 15" ViewPAC with 3 I/O Slots
VP-2201-CE7	WinCE 7.0 Based 7" ViewPAC without I/O Slots
VP-3201-CE7	WinCE 7.0 Based 8.4" ViewPAC without I/O Slots
VP-4201-CE7	WinCE 7.0 Based 10.4" ViewPAC without I/O Slots
VP-5201-CE7	WinCE 7.0 Based 12.1" ViewPAC without I/O Slots
VP-6201-CE7	WinCE 7.0 Based 15" ViewPAC without I/O Slots

XPAC family for x86 platform series

XP-8x4x-CE series	
XP-8041-CE6	WinCE 6.0 Based Standard XPAC with 0 I/O slot
XP-8341-CE6	WinCE 6.0 Based Standard XPAC with 3 I/O slots
XP-8741-CE6	WinCE 6.0 Based Standard XPAC with 7 I/O slots
XP-8x4x-Atom-CE6 series	
XP-8141-Atom-CE6	WinCE 6.0 Based Standard XPAC with 1 I/O slot
XP-8341-Atom-CE6	WinCE 6.0 Based Standard XPAC with 3 I/O slots
XP-8741-Atom-CE6	WinCE 6.0 Based Standard XPAC with 7 I/O slots
XP-8x3x-CE6 series	
XP-8131-CE6	WinCE 6.0 Based Standard XPAC with 1 I/O slot
XP-8331-CE6	WinCE 6.0 Based Standard XPAC with 3 I/O slots
XP-8731-CE6	WinCE 6.0 Based Standard XPAC with 7 I/O slots

WinPAC family for Arm PAX270 platform series

WP-8000 series	
WP-8131	WinCE 5.0 Based Standard WinPAC with 1 I/O slot
WP-8431	WinCE 5.0 Based Standard WinPAC with 4 I/O slots
WP-8831	WinCE 5.0 Based Standard WinPAC with 8 I/O slots
WP-8141	WinCE 5.0 Based Standard WinPAC with 1 I/O slot
WP-8441	WinCE 5.0 Based Standard WinPAC with 4 I/O slots
WP-8841	WinCE 5.0 Based Standard WinPAC with 8 I/O slots
WP-5000 series	
WP-5141	WinCE 5.0 Based palm-sized WinPAC
WP-5141-OD	WinCE 5.0 Based palm-sized WinPAC with Audio
ViewPAC series	
VP-23W1	WinCE 5.0 Based 3.5" ViewPAC
VP-25W1	WinCE 5.0 Based 5.7" ViewPAC
VP-4131	WinCE 5.0 Based 10.4" ViewPAC

Related Information

For additional information about your PAC that can be obtained from CD or by downloading the latest version from ICP DAS web site.

WinPAC family - AM335x series

WP-9000 series:

CD:\WinPAC_AM335x\wp-9000

https://www.icpdas.com/en/download/index.php?model=WP-9421-CE7

WP-8000 series:

CD:\WinPAC_AM335x\wp-8x2x

https://www.icpdas.com/en/download/index.php?model=WP-8421-CE7

WP-5000 series:

CD:\WinPAC_AM335x\Wp-5231

https://www.icpdas.com/en/download/index.php?model=WP-5231-CE7

ViewPAC series:

CD:\WinPAC_AM335x\VP-x231

https://www.icpdas.com/en/download/index.php?model=VP-1231-CE7

CD:\WinPAC_AM335x\VP-x201

https://www.icpdas.com/en/download/index.php?model=VP-2201-CE7

XPAC family - x86 series

XP-8x4x-CE series:

CD:\XP-8000-CE6\Document\

http://www.icpdas.com/products/PAC/xpac/download/xpac_ce6/download_documents.htm

XP-8x4x-Atom-CE6 series:

CD:\XPAC-ATOM-CE6\Document\

http://www.icpdas.com/products/PAC/xpac/download/xpac_atom_ce6/download_documents.htm

XP-8x3x-CE6 series:

CD:\XPAC-8x3x-CE6\Document\

https://www.icpdas.com/en/download/index.php?model=XP-8031-CE6

WinPAC family - PXA270 series

WP-8000 series:

CD:\Napdos\wp-8x4x_ce50\document\

https://www.icpdas.com/en/download/index.php?model=WP-8141-EN

WP-5000 series:

CD:\Napdos\wp-5000_ce50\Document\

https://www.icpdas.com/en/download/index.php?model=WP-5141-EN

ViewPAC series:

CD:\Napdos\vp-2000_ce50\Document\

https://www.icpdas.com/en/download/index.php?model=VP-25W1-en

How to contact us?

For support for this or any ICP DAS product, you can contact ICP DAS Customer Support in one of the following ways:

- Visit the ICP DAS Storage Manager technical support Web site at: https://www.icpdas.com/en/faq/index.php?model=WP-8421-CE7

- Submit a problem management record (PMR) electronically from our Web site at: http://www.icpdas.com/sevices/contact_customerservice.htm

- Send e-mail to: service@icpdas.com

Revision History

The table below lists the revision history.

Revision	Date	Description
1.0.1	August 2012	Initial issue
1.1.0	January 2013	Added the information about the defined slots and COM ports in appendix F.
1.2.0	June 2014	Added the support for WP-5231, VP-6641 Added 2 API functions(pac_GetBackLight/pac_SetBackLight)
1.2.1	August 2017	Modified UART_GetDataSize API functions Added pac_SetModulePowerOnEnStatus/ pac_GetModulePowerOnEnStatus functions
1.31	April 2021	Fix the e.g. demo code error.
1.3.2	Sep. 2024	Added function pac_ReadDICNTAll /pac_ClearDICNTAll

2. Getting Started

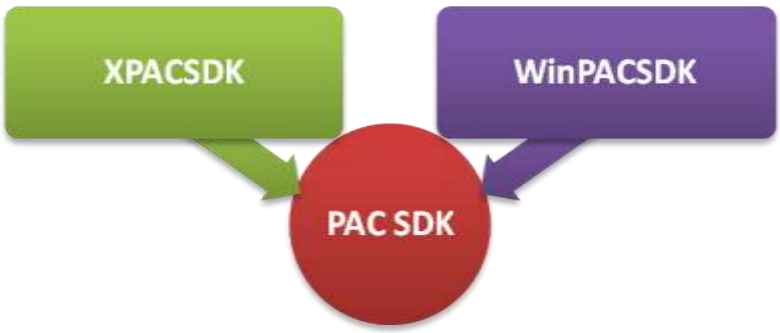
This chapter provides a guided tour that describes the steps needed to know, download, install and configure of the basic procedures for user working with the PACSDK for the first time.

2.1. Introducing the PACSDK

PACSDK are software development kits that contain header files, libraries, documentation and tools required to develop applications for XPAC, WinPAC and ViewPAC series.

PACSDK has replaced XPACSDK and WinPACSDK

ICP DAS has released a new SDK, PAC SDK, which merged with and replaced the XPACSDK and the WinPACSDK.



The XPAC/WinPAC SDK has been unified and renamed PACSDK. The new PACSDK.dll provides support for two platforms, one being designed for the WinPAC series (ARM platforms) and the other for the XPAC series (x86 platforms).

PACSDK.dll (x86) is linked to C programs for the XPAC series to replace the previous SDK, XPACSDK_CE.dll, and PACSDK.dll(ARM) is linked to C programs for the WinPAC series to replace the previous SDK, WinPACSDK.dll.

The PACNET.dll is used for .Net CF programs (C#, VB) for both the XPAC and WinPAC series to replace the previous SDKs (XPacNet.dll and WinPacNet.dll).

New/Previous SDK files comparison

Items	WinPACSDK	XPACSDK (CE6)	PACSDK
Header files	WinPacSDK.h	XPacSDK_CE.h	PACSDK.h PACSDK_PWM.h
Library files	WinPacSDK.lib	XPacSDK_CE.lib	PACSDK.lib PACSDK_PWM.lib
Target device Native DLL files	WinPacSDK.dll	XPacSDK_CE.dll	PACSDK.dll PACSDK_PWM.dll
Target device .NET CF DLL files	WinPacNet.dll	XpacNet.dll	PACNET.dll

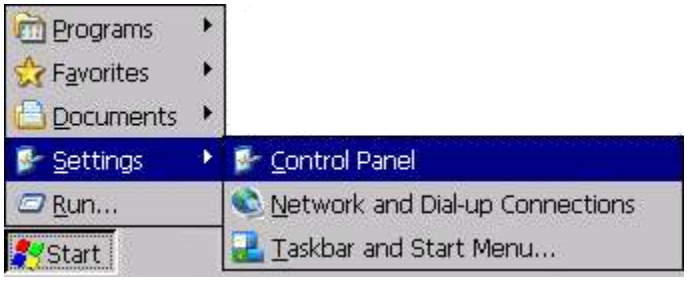
Benefits of the unified SDK include

- Easily migrates custom WinPAC programs to the XPAC series
- Easily migrates custom XPAC programs to the WinPAC series

A suite of PACSDK APIs is almost same as the previous SDK, (WinPACSDK.dll and XPACSDK_CE.dll) but there are some modifications and updates. Refer to the Appendix C for more details.

2.2. Installing the PACSDK

The installation package of the XPAC/WinPAC platform SDK which supports PACSDK library are available to enable users to develop the applications for the XPAC and WinPAC series.



To install the new installation package

Get the latest version of the installation package of WinPAC/XPAC platform SDK which supports PACSDK library.

XPAC (CE6) platform SDK:

The latest version of the installation package from FTP site isted as following FTP:

http://ftp.icpdas.com/pub/cd/xp-8x3x-ce6/sdk/platformsdk/
http://ftp.icpdas.com/pub/cd/xp-8000-ce6/sdk/platformsdk/
http://ftp.icpdas.com/pub/cd/xpac-atom-ce6/sdk/platformsdk/

File name: xpacsdk_ce_n.n.n_vsxxxx.msi
n.n.n : platform sdk version number
xxxx: 2005 indicates VS2005, 2008 indicates VS2008)

Tips & Warnings

The version number of SDK installation package provided the PACSDK library must be later than or equal to 1.4.0, such as PacSDK_CE_1.4.0_VS2008.msi or PacSDK_CE_1.4.0_VS2005.msi

WinPAC (CE5) platform SDK:

The latest version of the installation package from FTP site listed as following FTP:

http://ftp.icpdas.com/pub/cd/winpac/napdos/wp-8x4x_ce50/sdk/

File name: pac270_sdk_yyyymmdd.msi

yyyymmdd : platform sdk released date

Tips & Warnings

The released date of SDK installation package that provides the PACSDK library must be later than or equal to 2012/10/15, such as PAC270_SDK_20121015.msi

WinPAC (CE7) platform SDK:

The latest version of the installation package from FTP site listed as following FTP:

http://ftp.icpdas.com/pub/cd/winpac_am335x/wp-5231/sdk/platformsdk/

File name: AM335x_WINCE7_SDK_yyyymmdd.msi

yyyymmdd : platform sdk released date

Run the installation packge (*.msi) on PC and follow the prompts until the installation is complete

After the package of WinPAC platform SDK has installed to PC, the PACSDK library files will are copied to the default location on PC

C:\Program Files\Windows CE Tools\wce500\PAC270\Include\Armv4i

C:\Program Files\Windows CE Tools\wce500\PAC270\Lib\ARMV4I

After the package of XPAC platform SDK has installed to PC, the PACSDK library files will are copied to the default location on PC

C:\Program Files\Windows CE Tools\wce600\XPacSDK_CE\Include\X86

C:\Program Files\Windows CE Tools\wce600\XPacSDK_CE\Lib\X86

After the package of AM335x WinCE7 platform SDK has installed to PC, the PACSDK library files will are copied to the default location on PC

C:\Program Files\Windows CE Tools\SDKs\AM335x_WINCE7_SDK\Include\Armv4i

C:\Program Files\Windows CE Tools\SDKs\AM335x_WINCE7_SDK\Lib\Armv4i

To Update the XPACSDK/WinPACSDK to PACSDK

The documents, w6-10_How_to_update_to_PACSDK_library_from_WinPacSDK_library_en.pdf and w6-10_How_to_update_to_PACSDK_library_from_WinPacSDK_library_tc.pdf describe how update PACSDK library to replace WinPACSDK library on user's program.
It located at
http://ftp.icpdas.com/pub/cd/winpac/napdos/wp-8x4x_ce50/document/faq/sdk/

The documents, X6-10_How_to_update_to_PACSDK_library_from_XPacSDK_library_en.pdf and X6-10_How_to_update_to_PACSDK_library_from_XPacSDK_library_tc.pdf describe how update PACSDK library to replace XPacSDK library on user's program.
It located at
http://ftp.icpdas.com/pub/cd/xp-8000-ce6/document/faq/sdk/
or
http://ftp.icpdas.com/pub/cd/xpac-atom-ce6/document/faq/sdk/

2.3. Setting up the Development Environment

The OS of XPAC series is Windows CE 6.0, and OS of WinPAC series is Windows CE5.0. Different from development tools of XPAC and WinPAC series. XPAC supports Visual Studio 2005/2008, and in addition to support Visual studio 2005/2008, WinPAC supports Embedded Visual C++ (eVC). PAC controller with Windows embedded compact 7.0 OS (such as WP-5231) only supports Visual studio 2008.

2.3.1. C/C++/MFC based on Visual Studio(XPAC series)

Applied platforms

- All WinPAC series
- All ViewPAC series
- All XPAC series

Required header and library files

The following list lists the libraries, header files or DLL files you will need to include developing a PAC application or plug-in

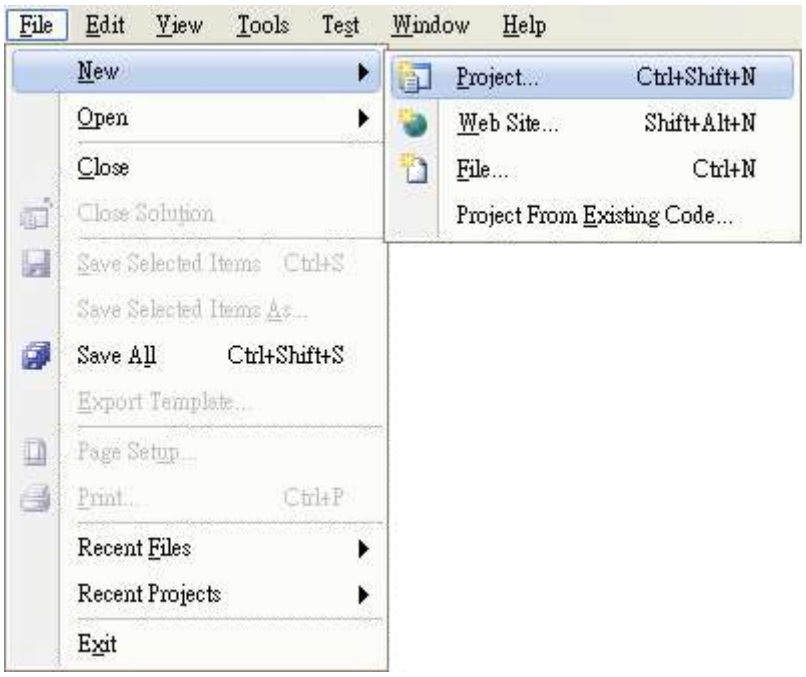
- PACSDK.h
- PACSDK.lib

If I-7K/I-87K PWM modules is used on PAC series device, you need to include or plug-in the files below

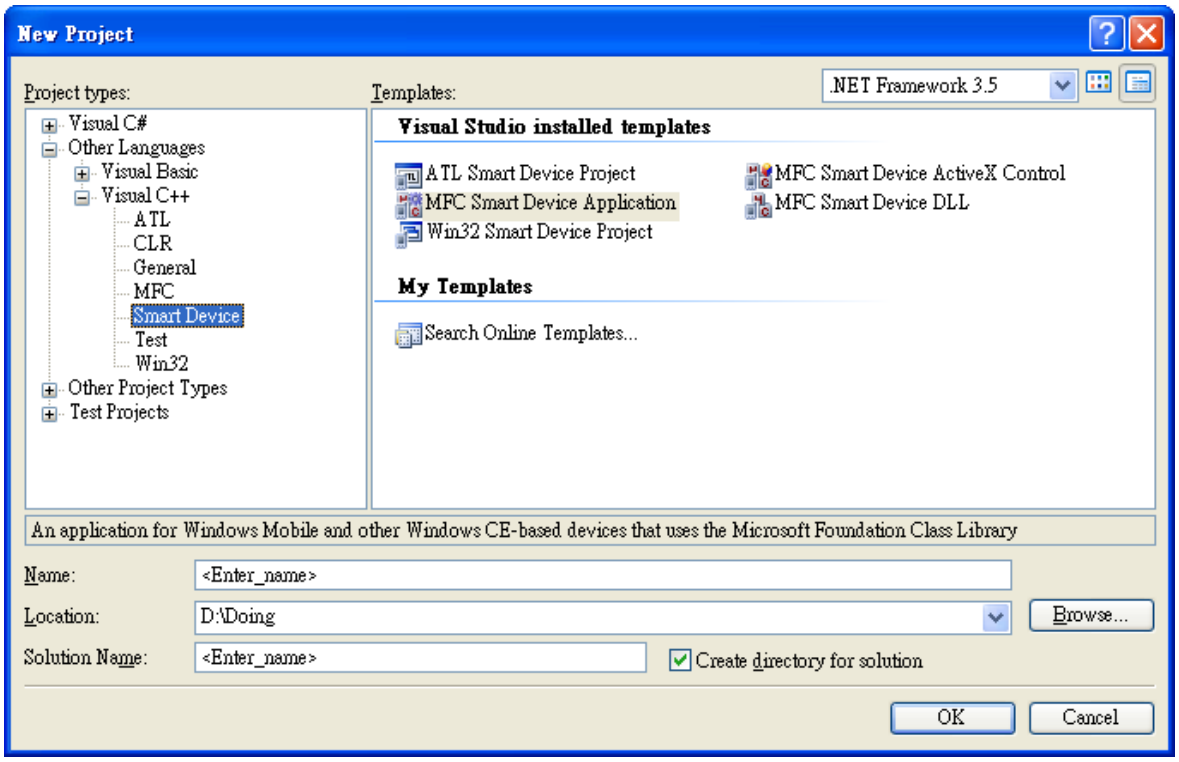
- PACSDK_PWM.h
- PACSDK_PWM.lib

How to create a program with new SDK using Visual Studio 2005/2008 (VS2005/VS2008)

1. Create a new project by using Visual Studio 2005/2008

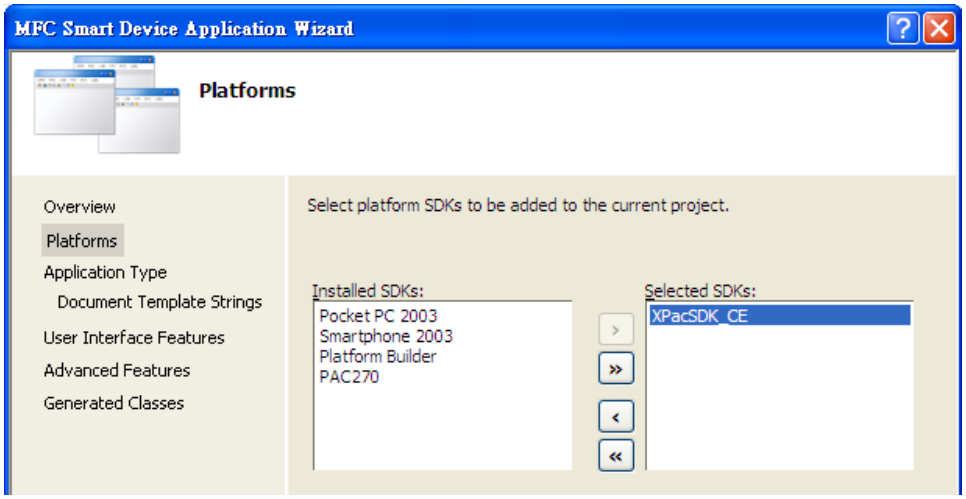


2. Select Smart Device



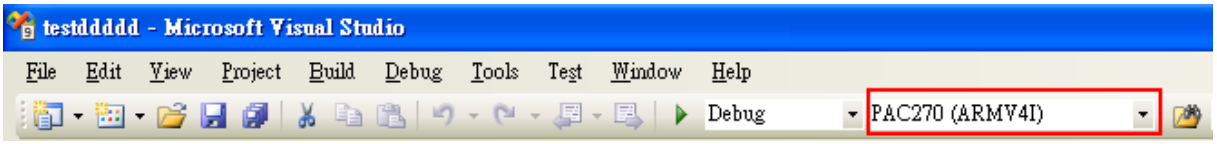
3. Selected SDKs

XPAC series (x86 CPU) - Select platform, XPacSDK_CE, to be added to the current project



4. On the configuration toolbar

Select the XPacSDK_CE(x86) for XPAC series(x86 CPU)



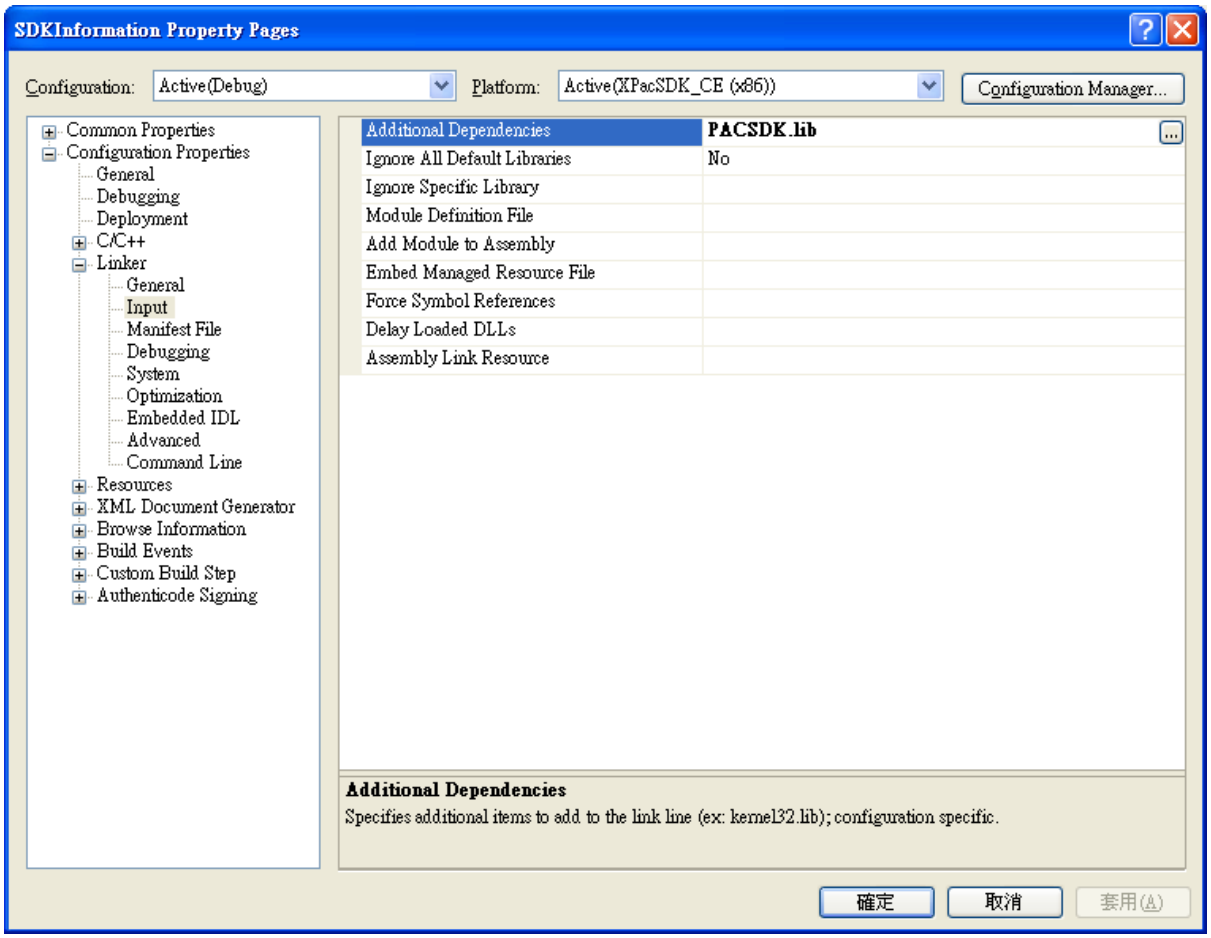
5. Include PACSDK.h

#include “PACSDK.h”

```
#include "stdafx.h"
#include "SDKInformation.h"
#include "SDKInformationDlg.h"
#include "PACSDK.H"
```

6. Include PACSDK.lib

In the right pane, choose the PACSDK.lib in the Additional Dependencies item



2.3.2. C/C++/MFC based on eVC(PXA270 Series)

Applied platforms

WP-8x3x/WP-8x4x series

WP-5000 series (except WP-5231)

VP-23Wx/VP-25Wx/VP-413x series

Required header and library files

The following list lists the libraries, header files or DLL files you will need to include developing a PAC application or plug-in

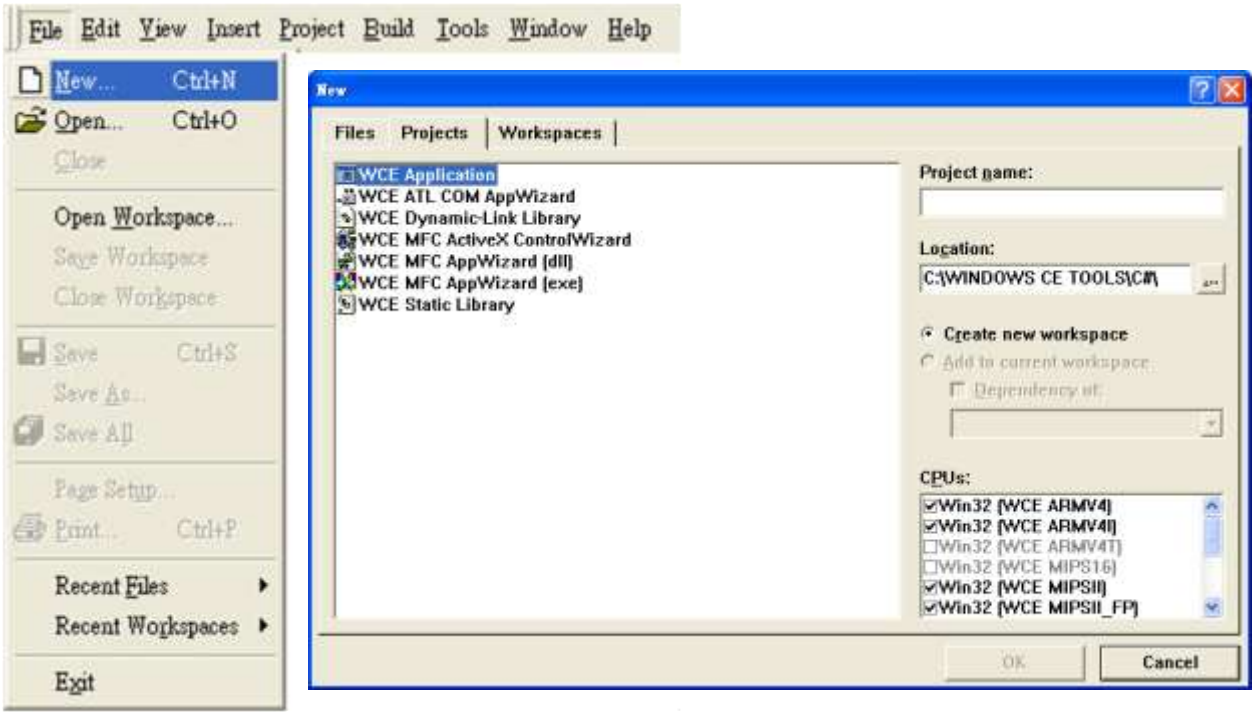
- PACSDK.h
- PACSDK.lib

If I-7K/I-87K PWM modules is used on PAC series device, you need to include or plug-in the files below

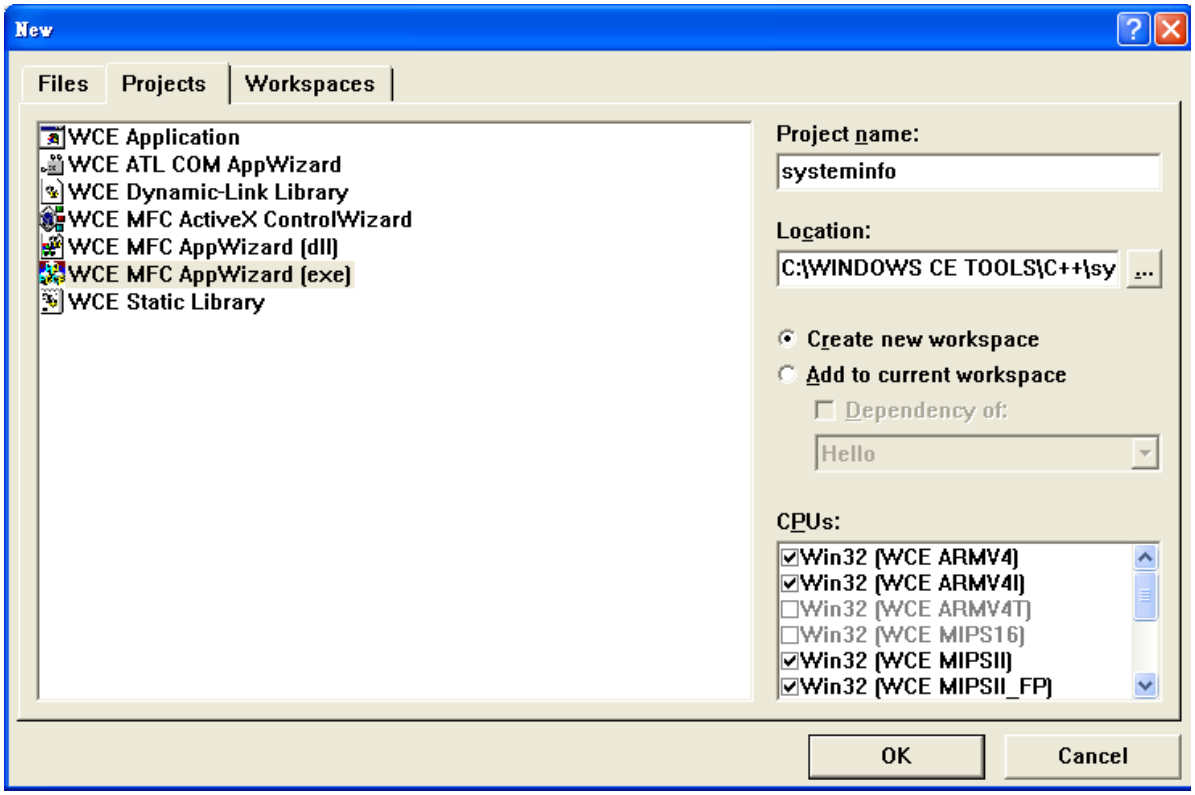
- PACSDK_PWM.h
- PACSDK_PWM.lib

How to create a program with new SDK using Microsoft Embedded Visual C++ (eVC)

1. Create a new project by using eVC



2. Select WCE MFC AppWizard (exe)



3. **Select platform, Win32 [WCE ARMV4I], to be added to the current project**

4. **On the configuration toolbar, select the "Win32 [WCE ARMV4] Release"**

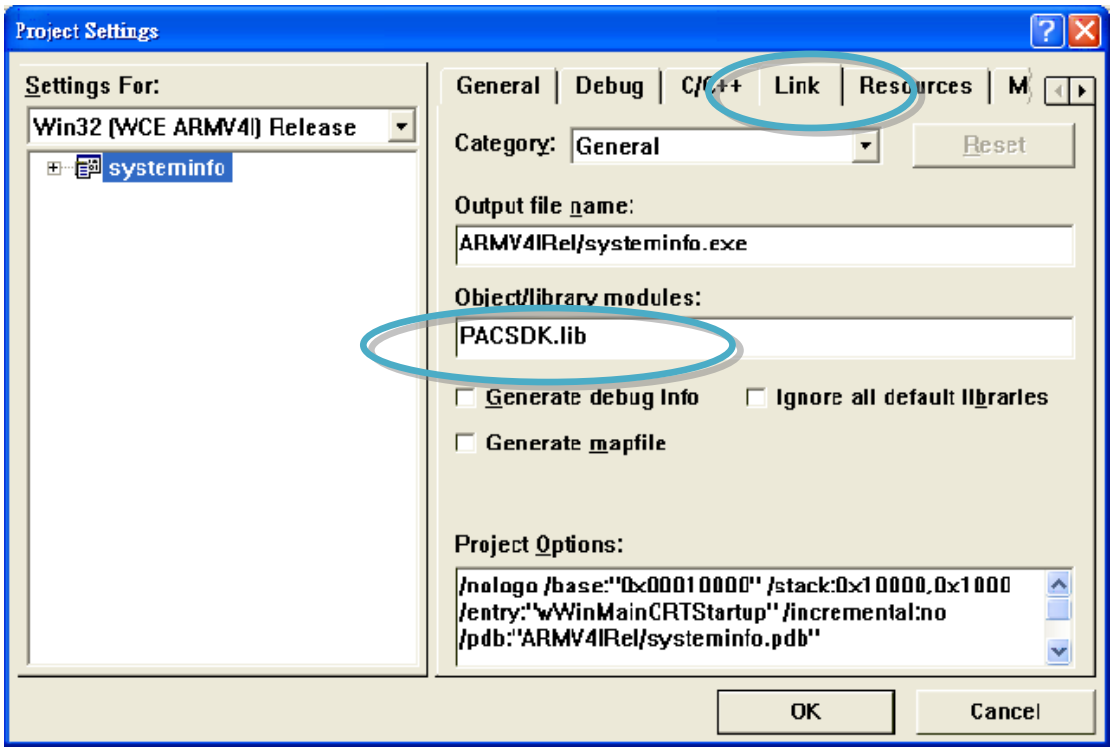
5. **Step 5: Include PACSDK.h**

   #include "PACSDK.h"

```
#include "stdafx.h"
#include "SDKInformation.h"
#include "SDKInformationDlg.h"
#include "PACSDK.H"
```

6. **Include PACSDK.lib**

   In the right pane, type the PACSDK.lib in the textbox



2.3.3. C/C++/MFC based on Visual Studio(PXA270 Series)

Applied platforms

- All WinPAC series
- All ViewPAC series
- All XPAC series

Required header and library files

The following list lists the libraries, header files or DLL files you will need to include developing a PAC application or plug-in

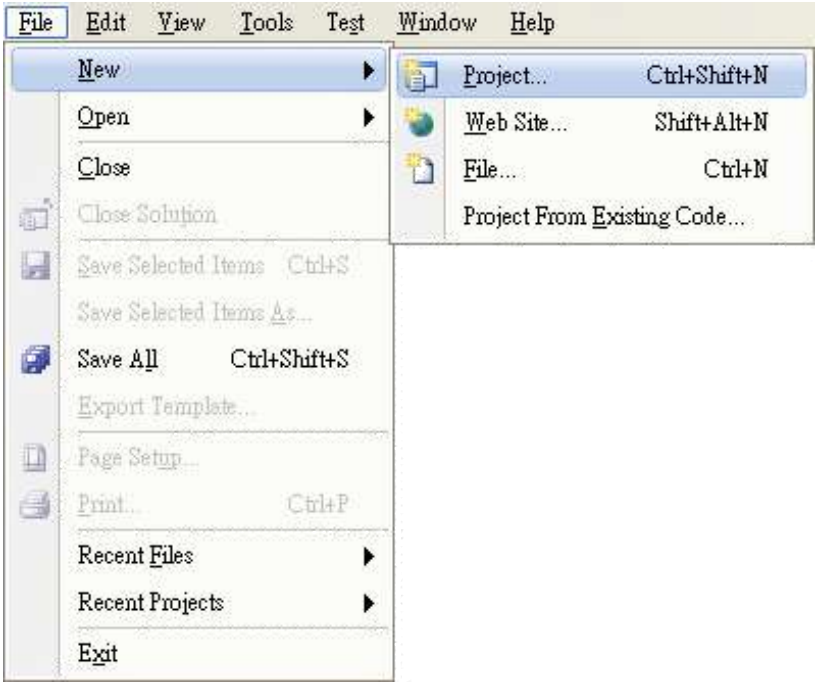
- PACSDK.h
- PACSDK.lib

If I-7K/I-87K PWM modules is used on PAC series device, you need to include or plug-in the files below

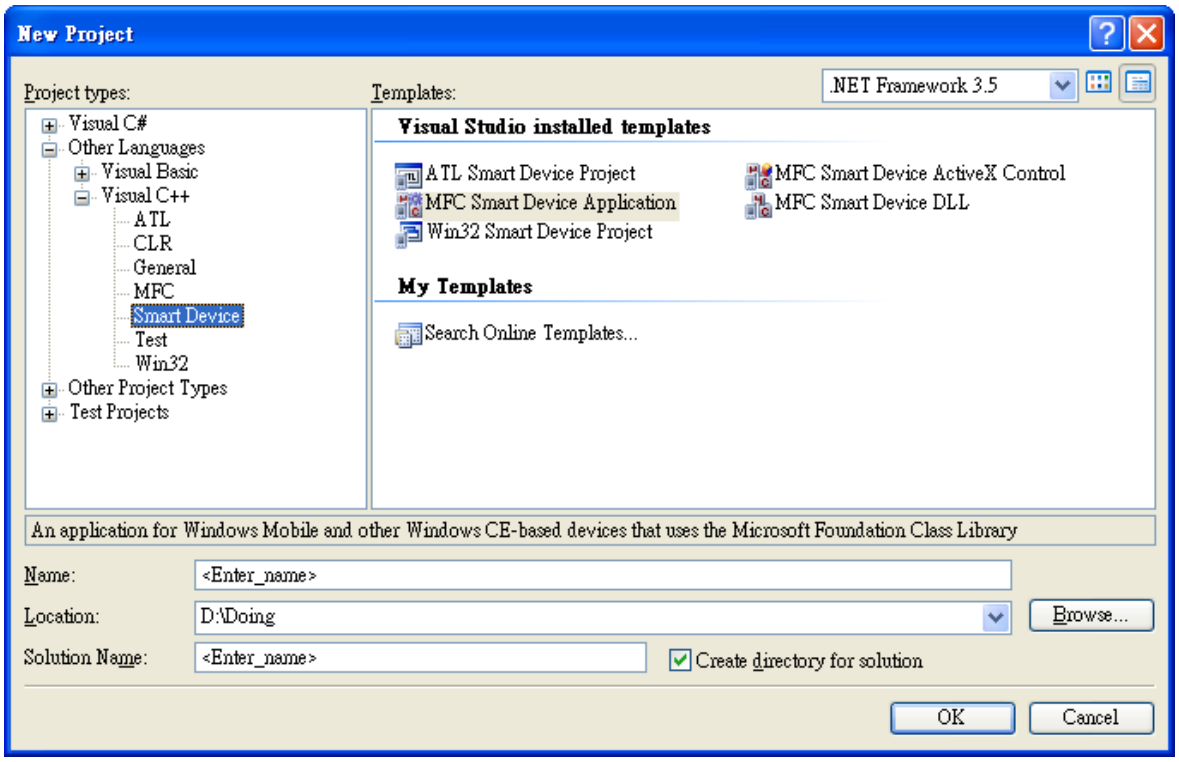
- PACSDK_PWM.h
- PACSDK_PWM.lib

How to create a program with new SDK using Visual Studio 2005/2008 (VS2005/VS2008)

1. Create a new project by using Visual Studio 2005/2008

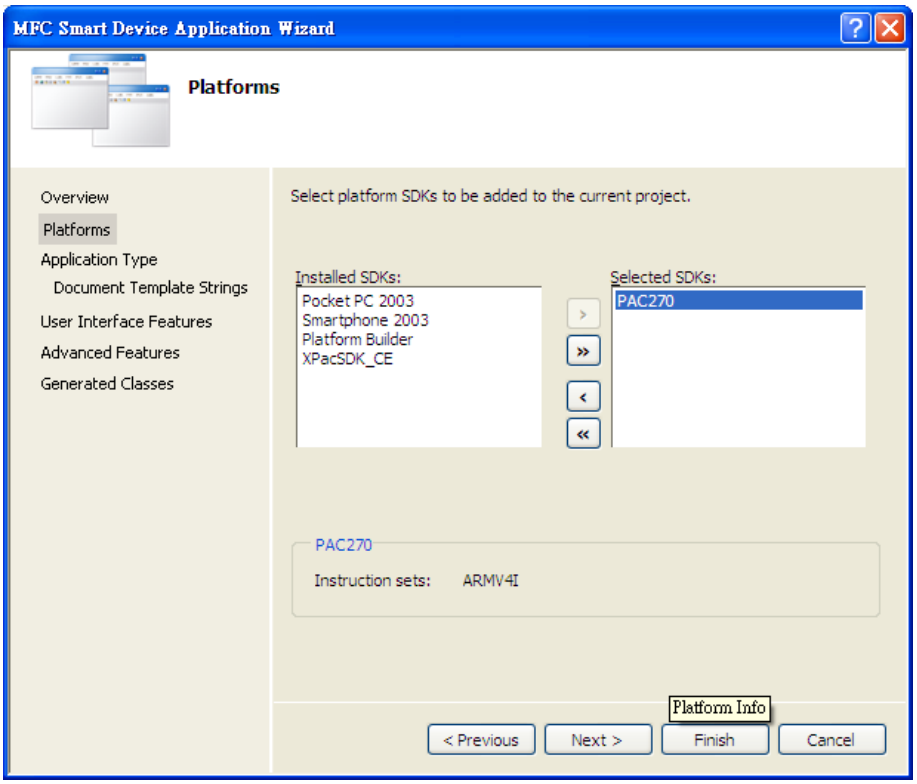


2. Select Smart Device



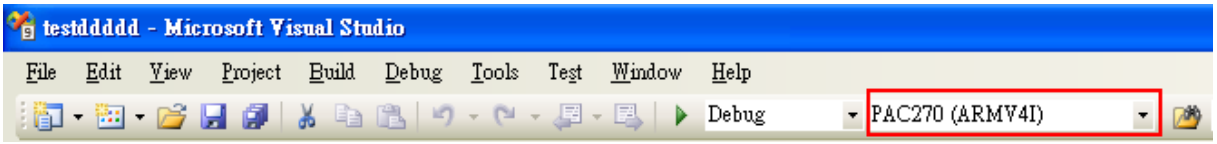
3. Selected SDKs

WinPAC series (PAX270 CPU) - Select platform, PAC270, to be added to the current project



4. On the configuration toolbar

Select the PAC270(ARMV4I) for WinPAC series (PXA270 CPU)



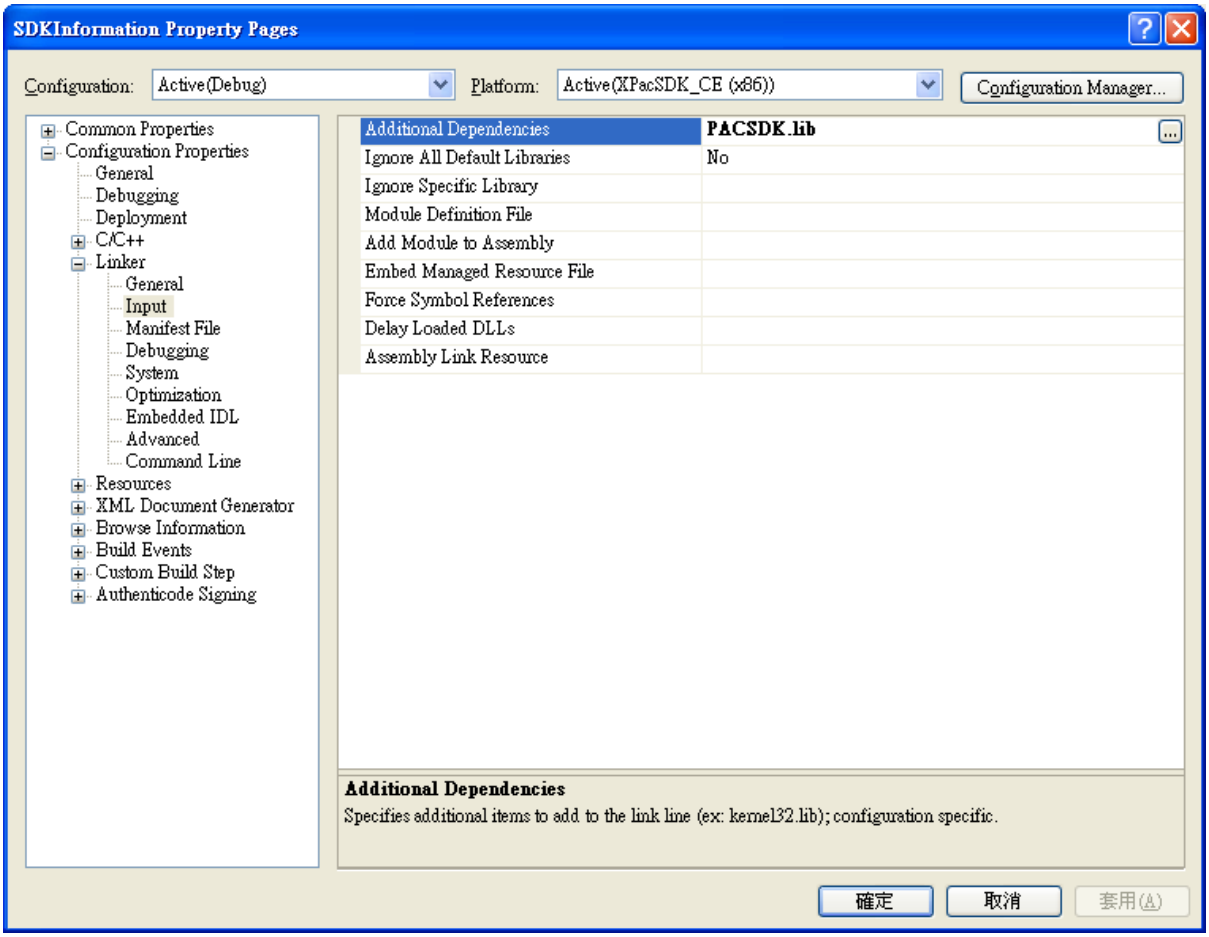
5. Include PACSDK.h

#include “PACSDK.h”

```
#include "stdafx.h"
#include "SDKInformation.h"
#include "SDKInformationDlg.h"
#include "PACSDK.H"
```

6. Include PACSDK.lib

In the right pane, choose the PACSDK.lib in the Additional Dependencies item



2.3.4. C/C++/MFC based on Visual Studio(AM335x Series)

Applied platforms

- All WinPAC series
- All ViewPAC series
- All XPAC series

Required header and library files

The following list lists the libraries, header files or DLL files you will need to include developing a PAC application or plug-in

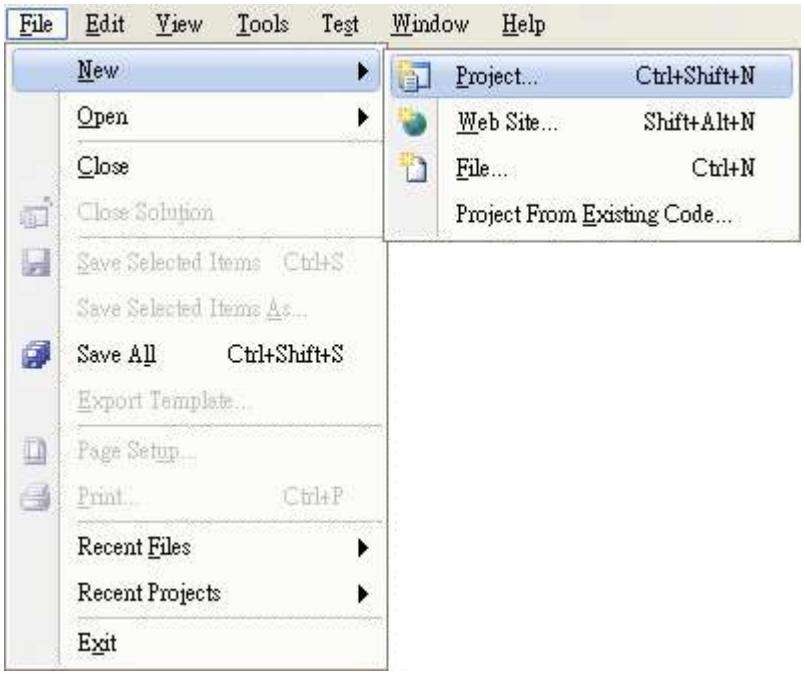
- PACSDK.h
- PACSDK.lib

If I-7K/I-87K PWM modules is used on PAC series device, you need to include or plug-in the files below

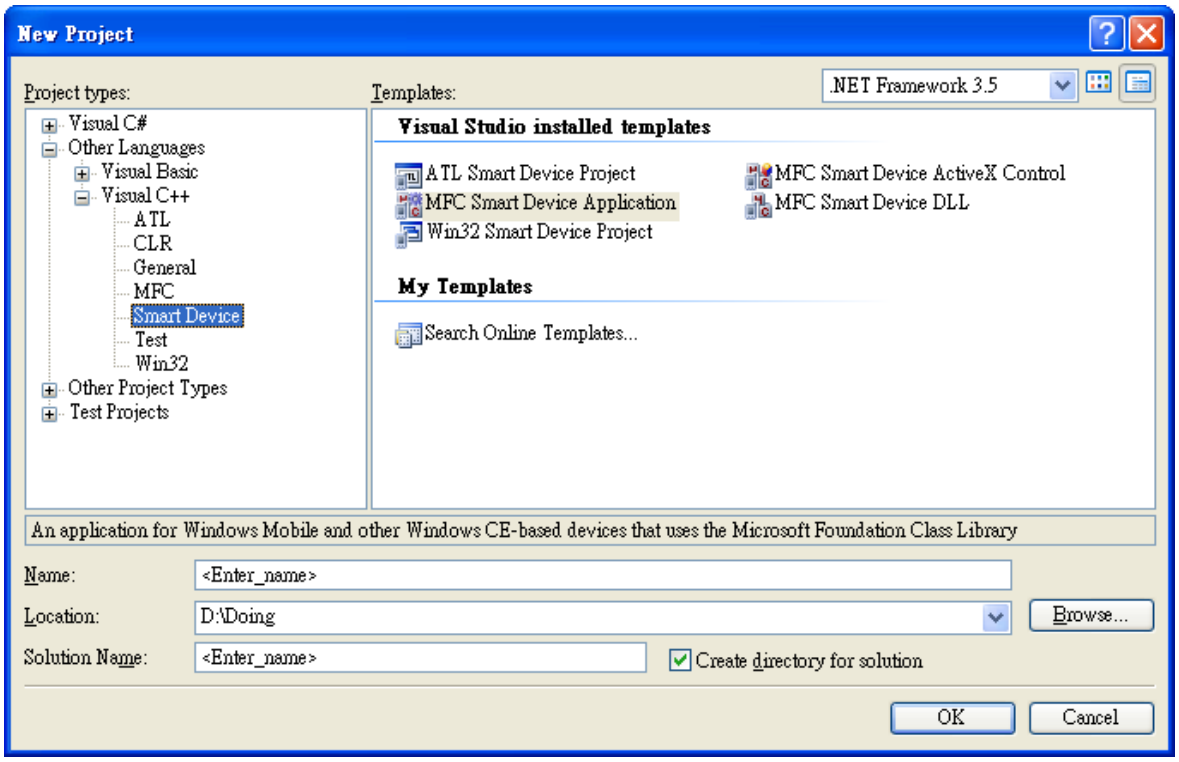
- PACSDK_PWM.h
- PACSDK_PWM.lib

How to create a program with new SDK using Visual Studio 2005/2008 (VS2005/VS2008)

1. Create a new project by using Visual Studio 2005/2008

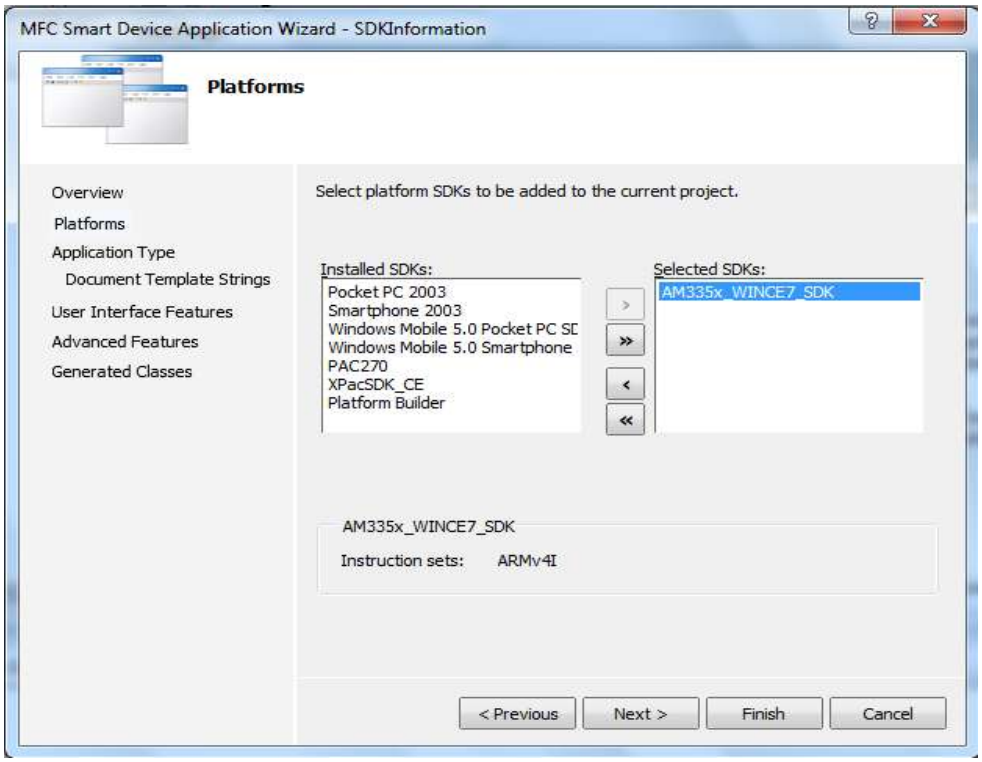


2. Select Smart Device



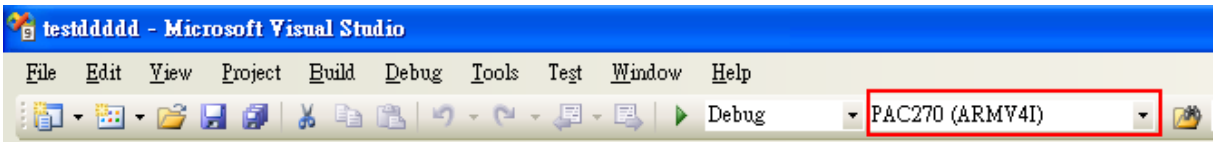
3. Selected SDKs

WP-5000 series (AM335x CPU) - Select platform, AM335x_WINCE7_SDK, to be added to the current project



4. On the configuration toolbar

Select AM335x_WINCE7_SDK(ARMv4I) for WinPaC series(AM335x CPU)



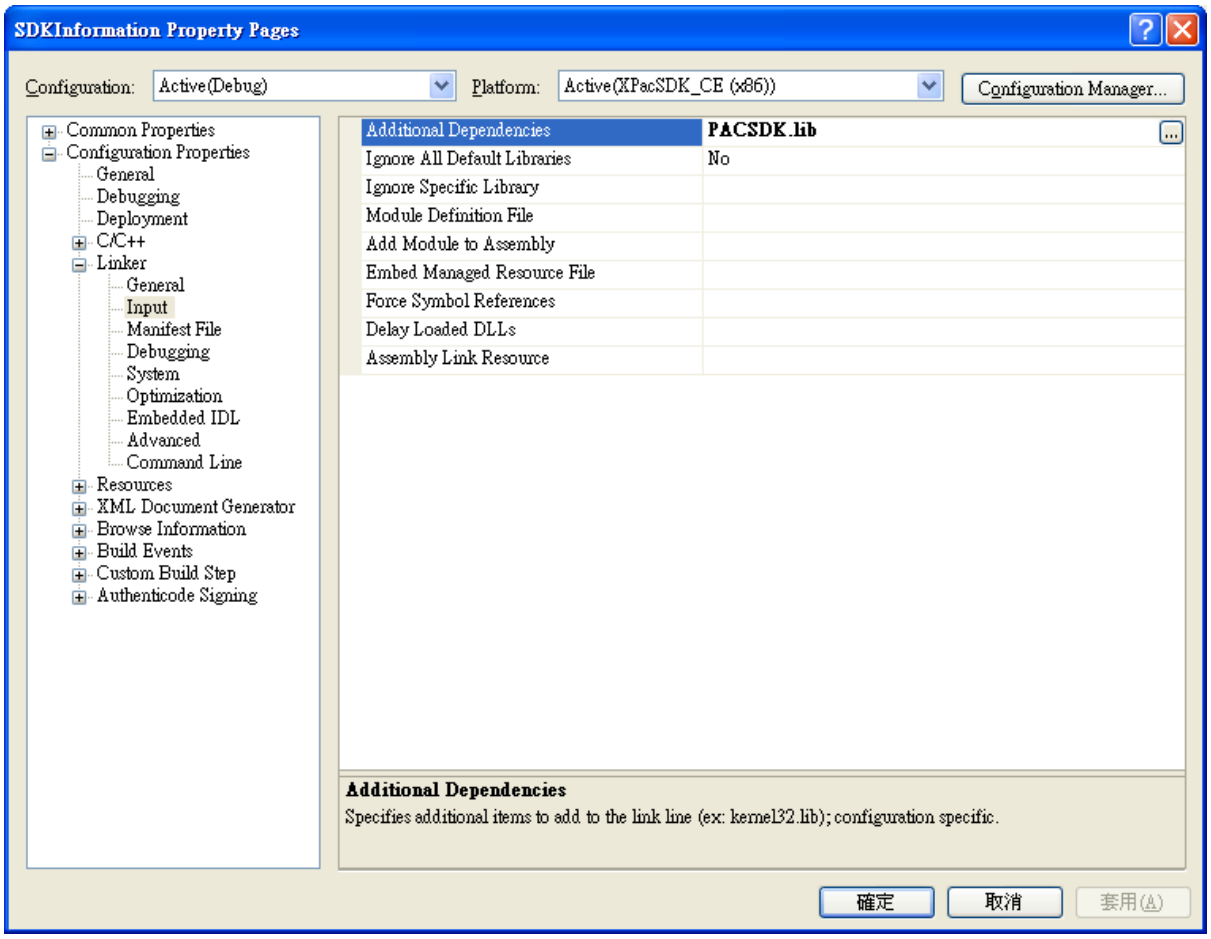
5. Include PACSDK.h

#include “PACSDK.h”

```
#include "stdafx.h"
#include "SDKInformation.h"
#include "SDKInformationDlg.h"
#include "PACSDK.H"
```

6. Include PACSDK.lib

In the right pane, choose the PACSDK.lib in the Additional Dependencies item



2.3.5. C# (XPAC/WinPAC Series)

Applied platforms

- All WinPAC series
- All ViewPAC series
- All XPAC series

Required library files

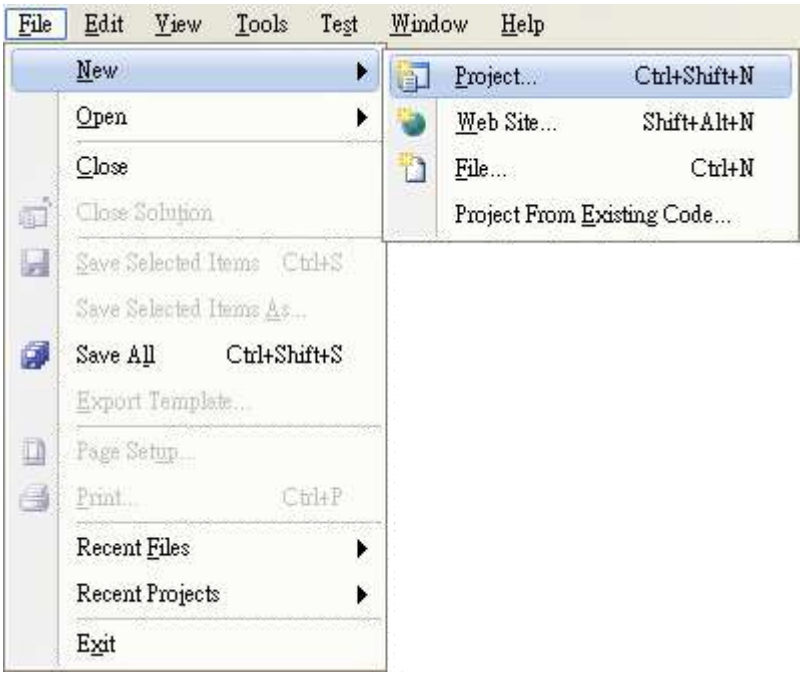
The following DLL files is needed to include to develop a PAC application or plug-in

- PACSDK.dll
- PACSDK_PWM.dll (If I-7K/I-87K PWM modules is used on the device)

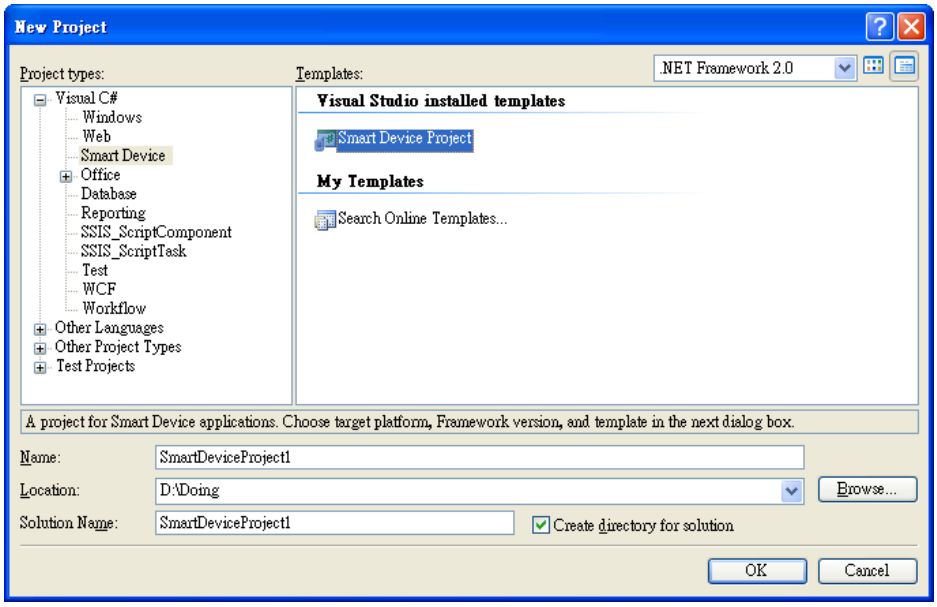
How to create a program with new SDK using Visual Studio 2005/2008 (VS2005/VS2008)

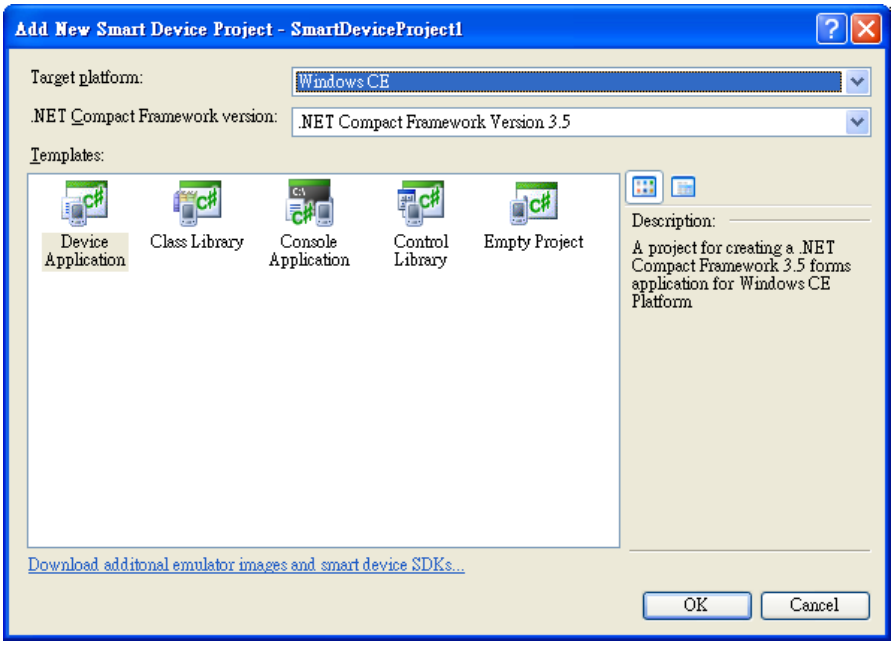
☑ Using Dll Import:

1. Create a new project by using Visual Studio 2005/2008



2. Select Smart Device





3. In order to use “DllImport”, you should using System.Runtime.InteropServices, and then implement the function which you want to call

For example of using pac_WriteDO in .NET project.

[The function defined In PACSDK.h file]

XPAC_API BOOL pac_WriteDO(HANDLE hPort, int slot, int iDO_TotalCh, DWORD lDO_Value);

[How to use in your .NET project]

- Added this line in your project:

```
using System.Runtime.InteropServices;
```

- Declare this function as following:

```
[DllImport("PACSDK.dll", EntryPoint = "pac_WriteDO")]
public extern static bool pac_WriteDO(IntPtr hPort, int slot, int iDO_TotalCh, uint lDO_Value);
```

- Then you can use this function, pac_WriteDO, in your .NET project.

  [Code Snippet]

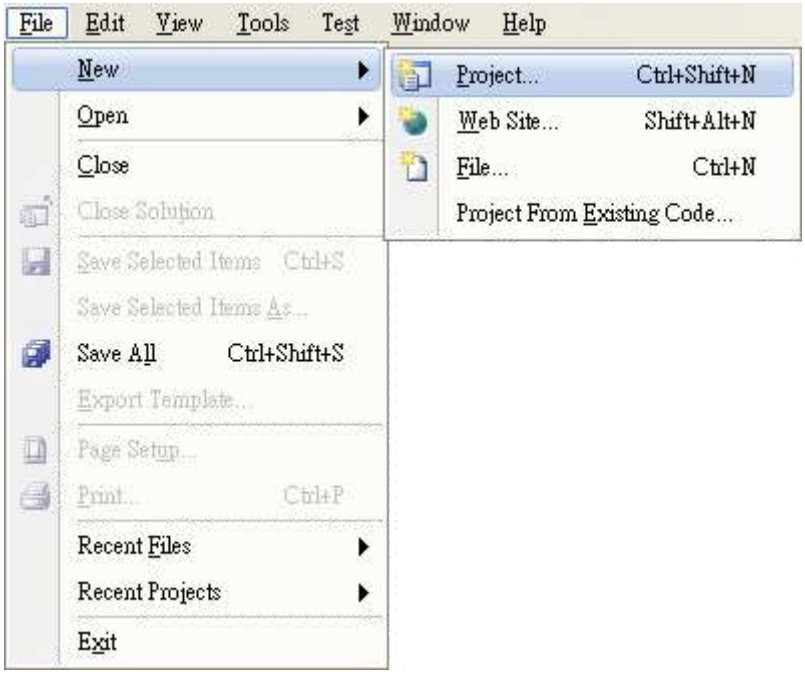
```
using System.Windows.Forms;
using System.Runtime.InteropServices;
namespace WindowsFormsApplication2
{
    public partial class Form1 : Form
    {
        [DllImport("PACSDK.dll", EntryPoint = "pac_WriteDO")]
        public extern static bool pac_WriteDO(IntPtr hPort, int slot, int
        iDO_TotalCh, uint lDO_Value);
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            pac_WriteDO((IntPtr)0, 1, 16, 0xff);
        }
    }
}
```

☑ Using PACNET.dll

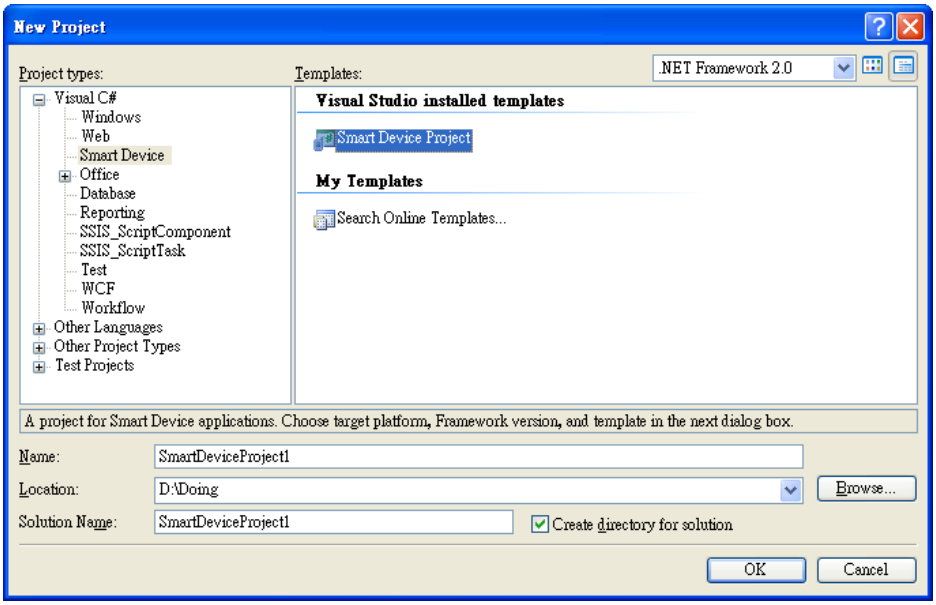
PACNET.dll is a .net Compact framework SDK and PACNET.dll isn't used for C# program but also used for VB.net program.
PACNET.dll (the execution file should be put in the same directory of the PACNET.dll)

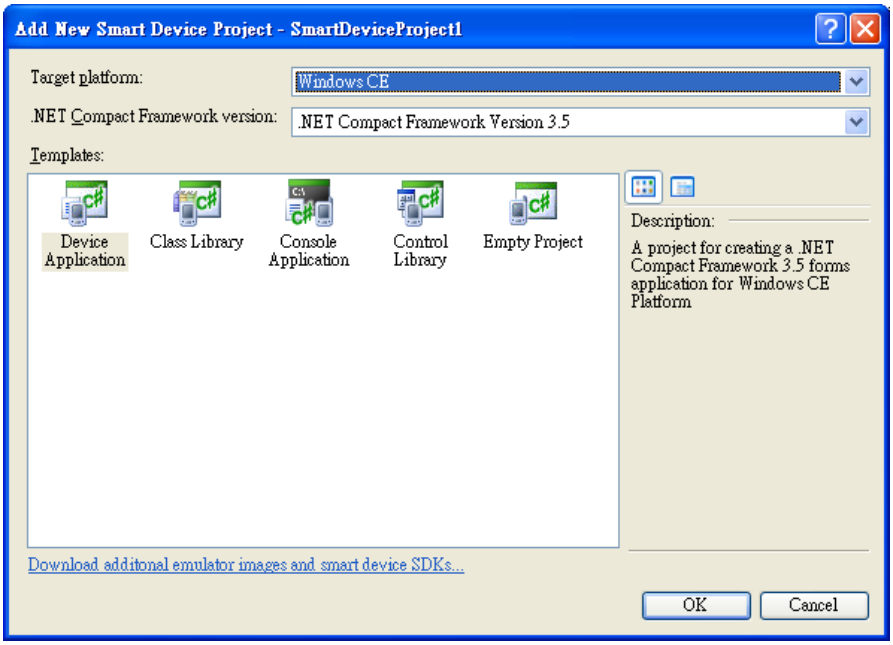
The latest version of this library is located at:
CD root\XP-8000-CE6\SDK\XPacNET (in the companion CD)
ftp://ftp.icpdas.com/pub/cd/xp-8000-ce6/sdk/xpacnet

1. Create a new project by using Visual Studio 2005/2008

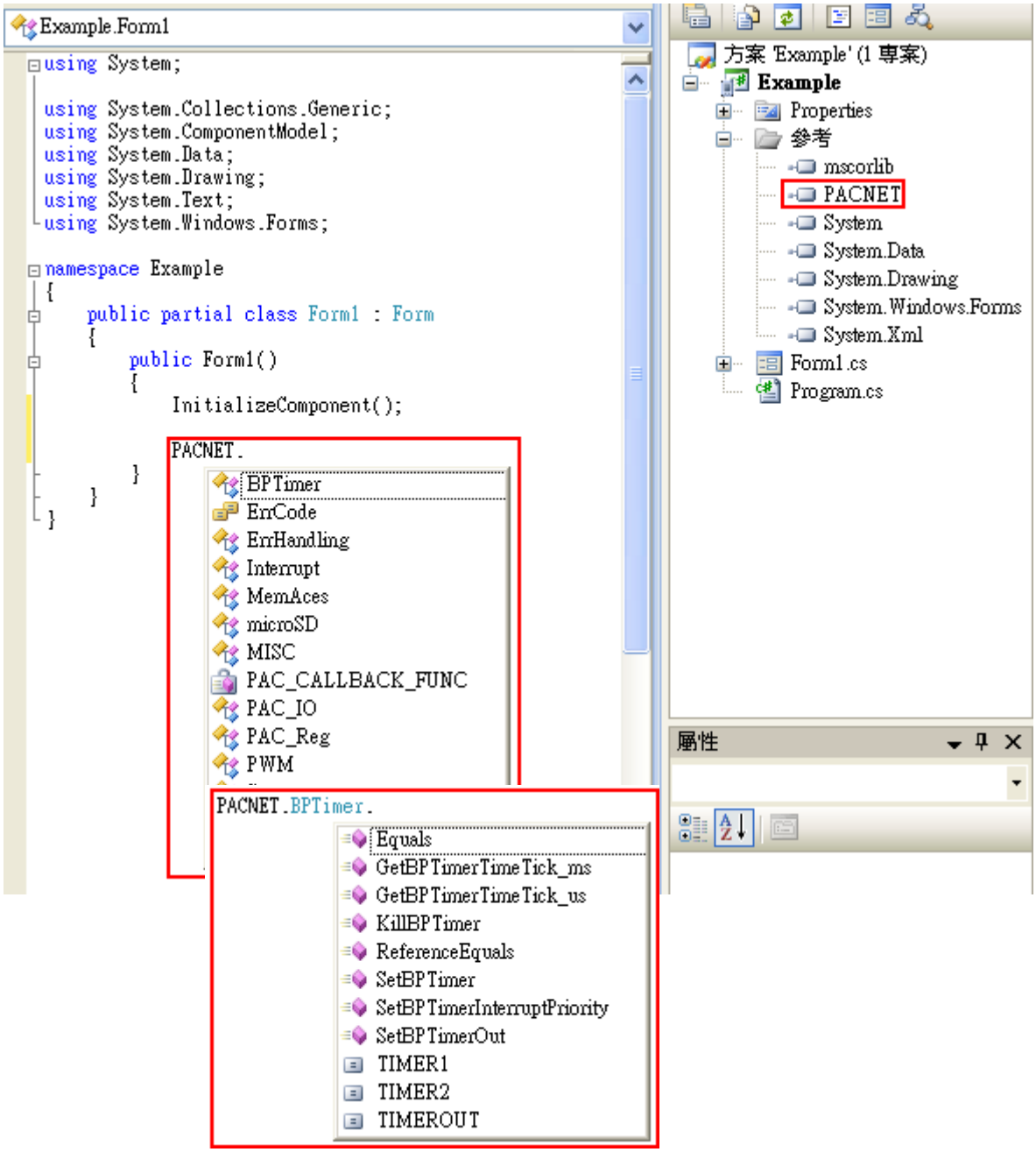


2. Select Smart Device





3. Add the PACNET.dll into the references of the project, and using PACNET



2.3.6. VB.NET

Applied platforms

- All XPAC series
- All WinPAC series
- All ViewPAC series

Required library files

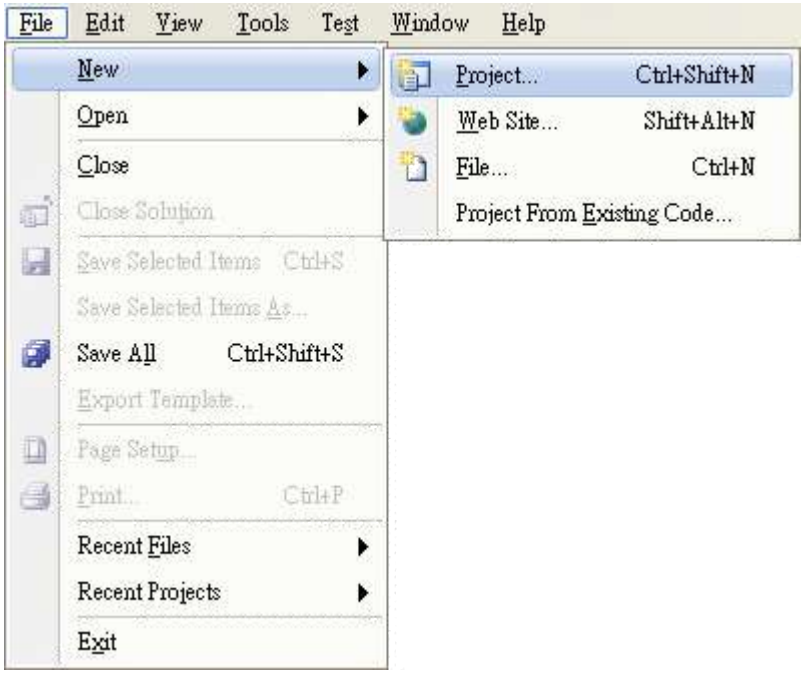
The following DLL files need to be included to develop a XPAC/WinPAC application or plug-in.

- PACSDK.dll
- PACSDK_PWM.dll (If I-7K/I-87K PWM modules is used on the device)

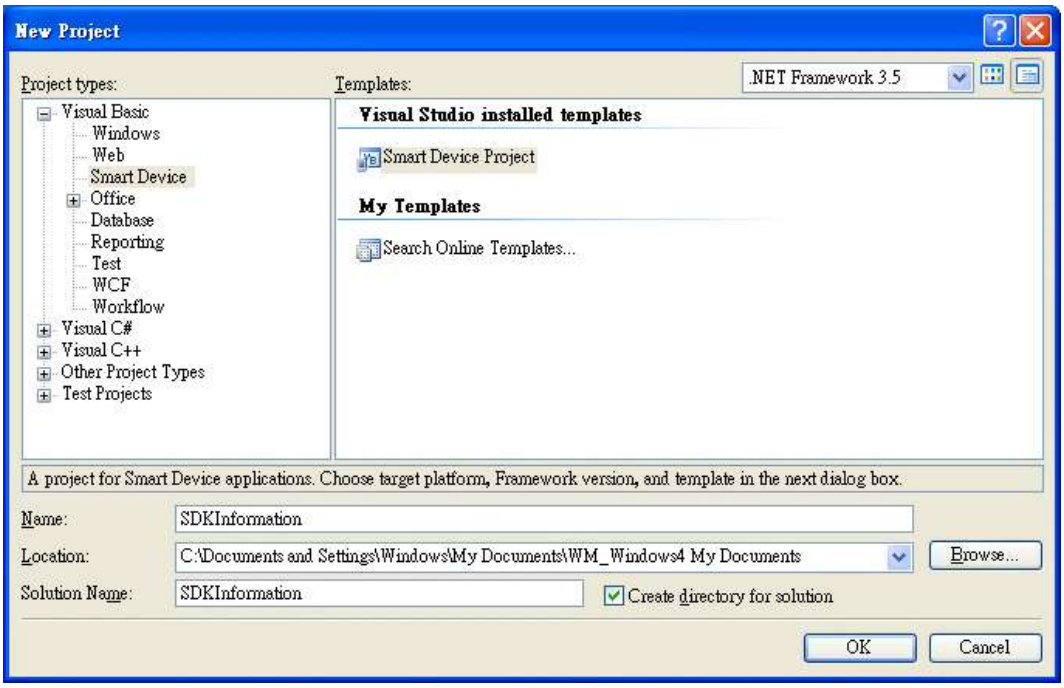
How to create a program with new SDK using Visual Studio 2005/2008 (VS2005/VS2008)

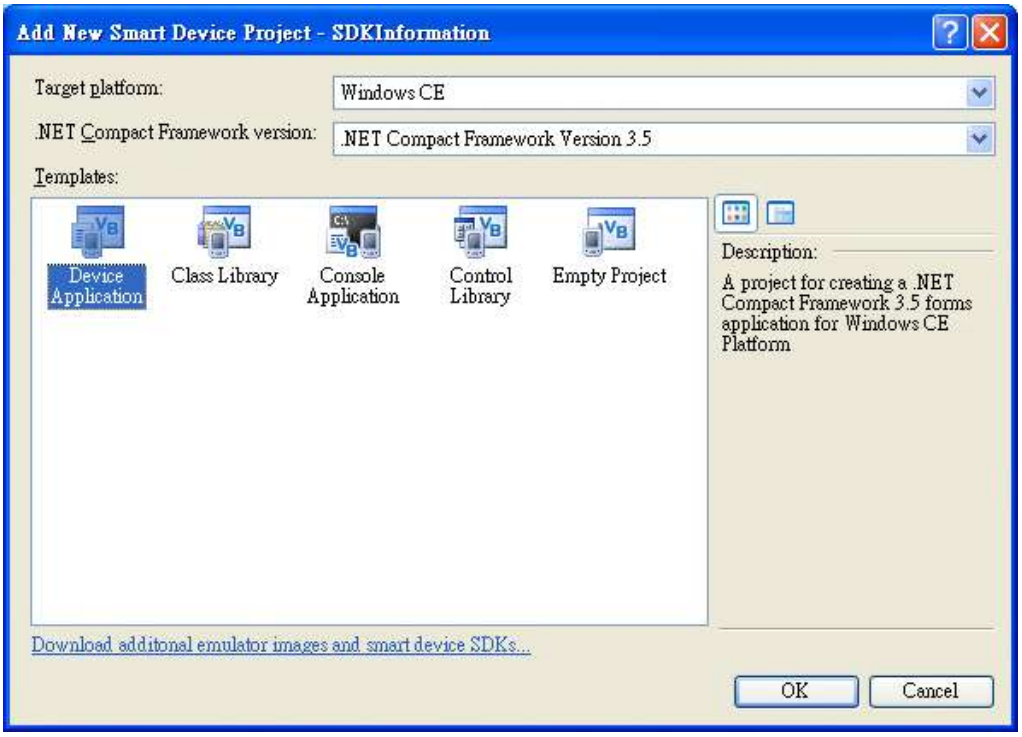
☑ Using Dll Import:

1. Create a new project by using Visual Studio 2005/2008



2. Select Smart Device





3. In order to use “DllImport”, you should using System.Runtime.InteropServices, and then **implement the function which you want to call**

For example of using pac_WriteDO in .NET project.

[The function defined In PACSDK.h file]

XPAC_API BOOL pac_WriteDO(HANDLE hPort, int slot, int iDO_TotalCh, DWORD lDO_Value);

[How to use in your .NET project]

- Added this line in your project:

```
Imports System.Runtime.InteropServices
```

- Declare this function as following:

```
<DllImport("PACSDK.dll", EntryPoint := "pac_WriteDO")> _
Public Shared Function pac_WriteDO(hPort As IntPtr, slot As Integer, iDO_TotalCh As Integer, lDO_Value As UInteger) As Boolean
End Function
```

- Then you can use this function, pac_WriteDO, in your .NET project.

[Code Snippet]

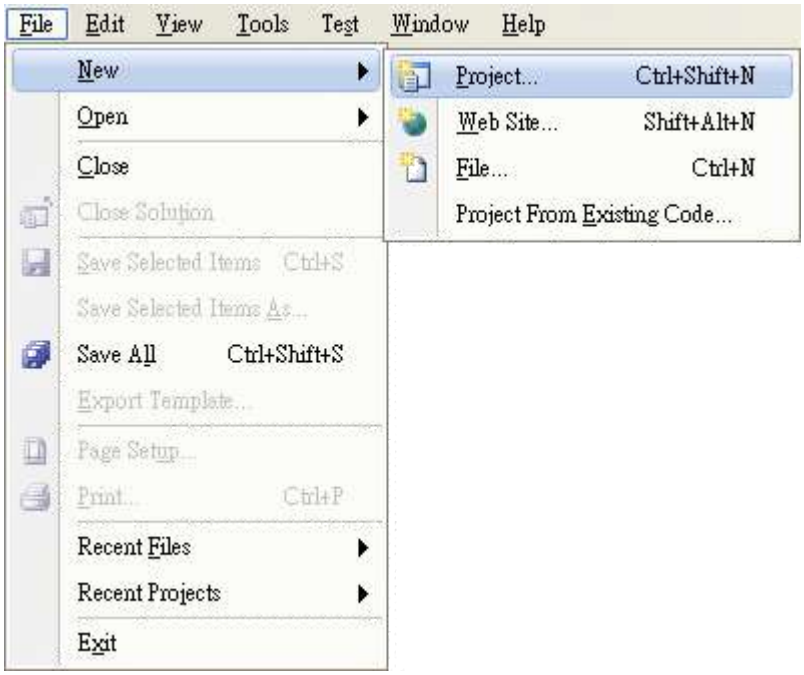
```
Imports System.Windows.Forms
Imports System.Runtime.InteropServices
Namespace WindowsFormsApplication2
    Public Partial Class Form1
        Inherits Form
            <DllImport("PACSDK.dll", EntryPoint := "pac_WriteDO")> _
            Public Shared Function pac_WriteDO(hPort As IntPtr, slot As Integer,
            iDO_TotalCh As Integer, lDO_Value As UInteger) As Boolean
        End Function

        Private Sub button1_Click(sender As Object, e As System.EventArgs) Handles
        button1.Click
            pac_WriteDO(CType(0, IntPtr), 1, 16, &Hff)
        End Sub
    End Class
End Namespace
```

☑ Using PACNET.dll

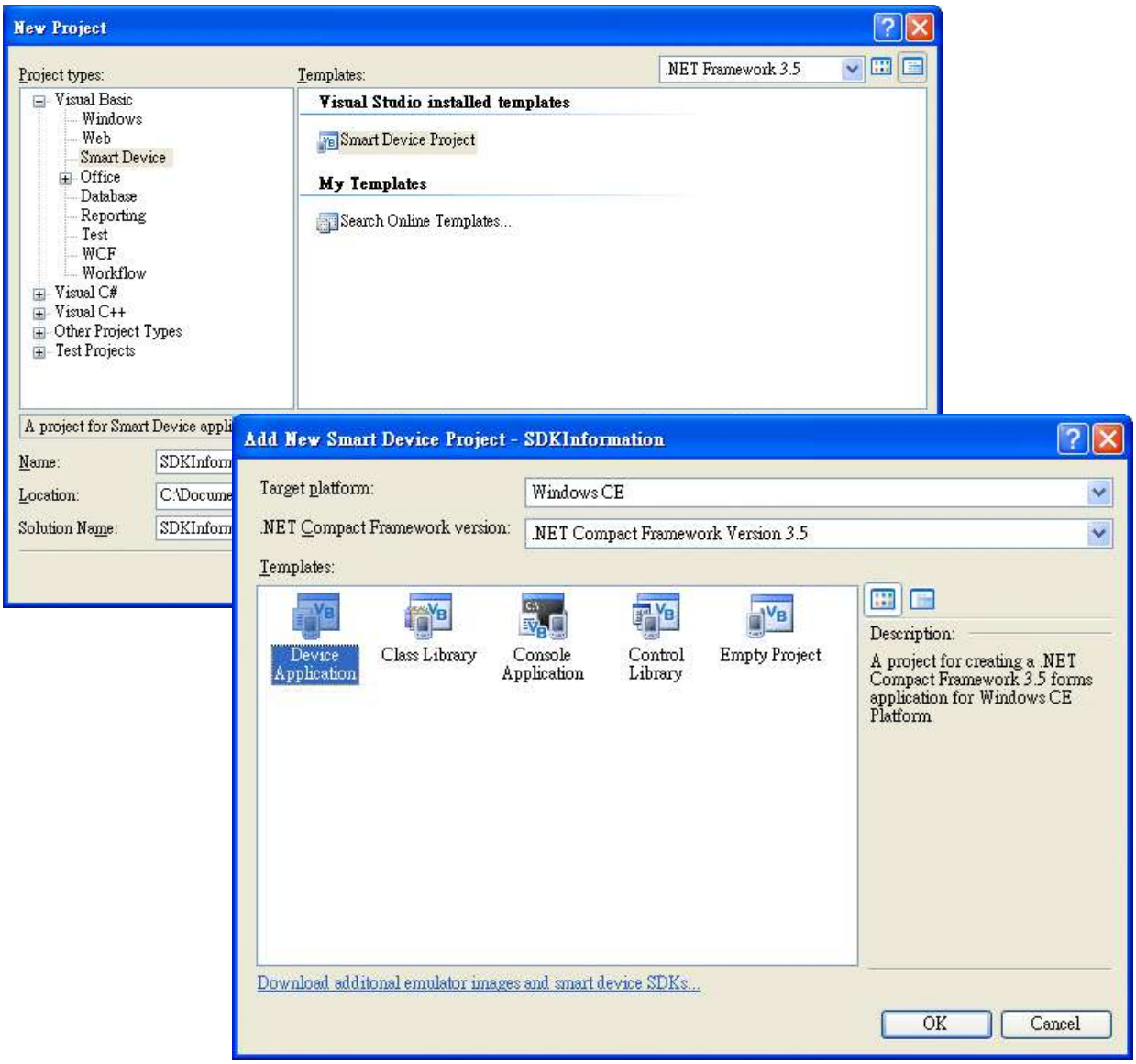
PACNET.dll is a .net Compact framework SDK and PACNET.dll isn't used for C# program but also used for VB.net program.
PACNET.dll (the execution file should be put in the same directory of the PACNET.dll)

The latest version of the library is located at:
CD root\XP-8000-CE6\SDK\XPacNET (in the companion CD)
ftp://ftp.icpdas.com/pub/cd/xp-8000-ce6/sdk/xpacnet

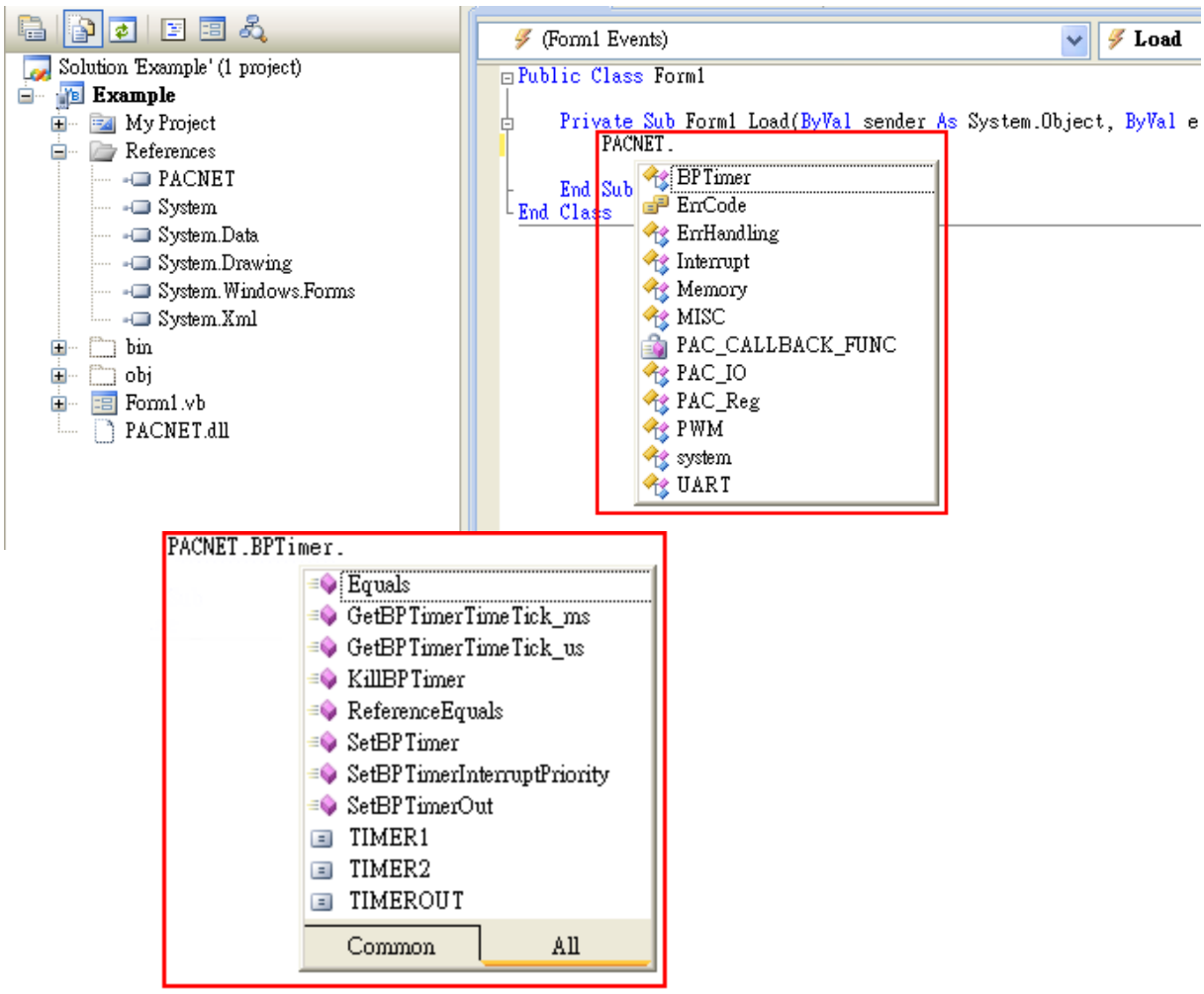
1. **Create a new project by using Visual Studio 2005/2008**



2. **Select Smart Device**



3. Add the PACNET.dll into the references of the project, and using PACNET



The following references illustrate how to develop the programs for a variety of platforms step by step.

XP-8x4x-Atom-CE6

Refer to the chapter 4 on xp-8000-Atom-ce6 user manual for more details

ftp://ftp.icpdas.com/pub/cd/xpac-atom-ce6/document/user_manual/

XP-8x3x-CE6

Refer to the chapter 4 on xp-8000-Atom-ce6 user manual for more details

ftp://ftp.icpdas.com/pub/cd/xp-8x3x-ce6/document/user_manual/

WP-8000 (CE5.0)

Refer to the chapter 4 on wp-8000 user manual for more details

ftp://ftp.icpdas.com/pub/cd/winpac/napdos/wp-8x4x_ce50/document/

WP-5000(CE5.0)

Refer to the chapter 4 on wp-5000 user manual for more details

ftp://ftp.icpdas.com/pub/cd/winpac/napdos/wp-5000_ce50/document/

VP-2xWx(CE5.0)

Refer to the chapter 4 on ViewPAC user manual for more details

ftp://ftp.icpdas.com/pub/cd/winpac/napdos/vp-2000_ce50/document/

WP-9000 (CE7.0)

Refer to the chapter 4 on wp-9000 user manual for more details

ftp://ftp.icpdas.com/pub/cd/winpac_am335x/wp-9000/document/

WP-8000 (CE7.0)

Refer to the chapter 4 on wp-8000 user manual for more details

ftp://ftp.icpdas.com/pub/cd/winpac_am335x/wp-8x2x/document/

WP-5000(CE7.0)

Refer to the chapter 4 on wp-5000 user manual for more details

ftp://ftp.icpdas.com/pub/cd/winpac_am335x/wp-5231/document/

ViewPAC(CE7.0)

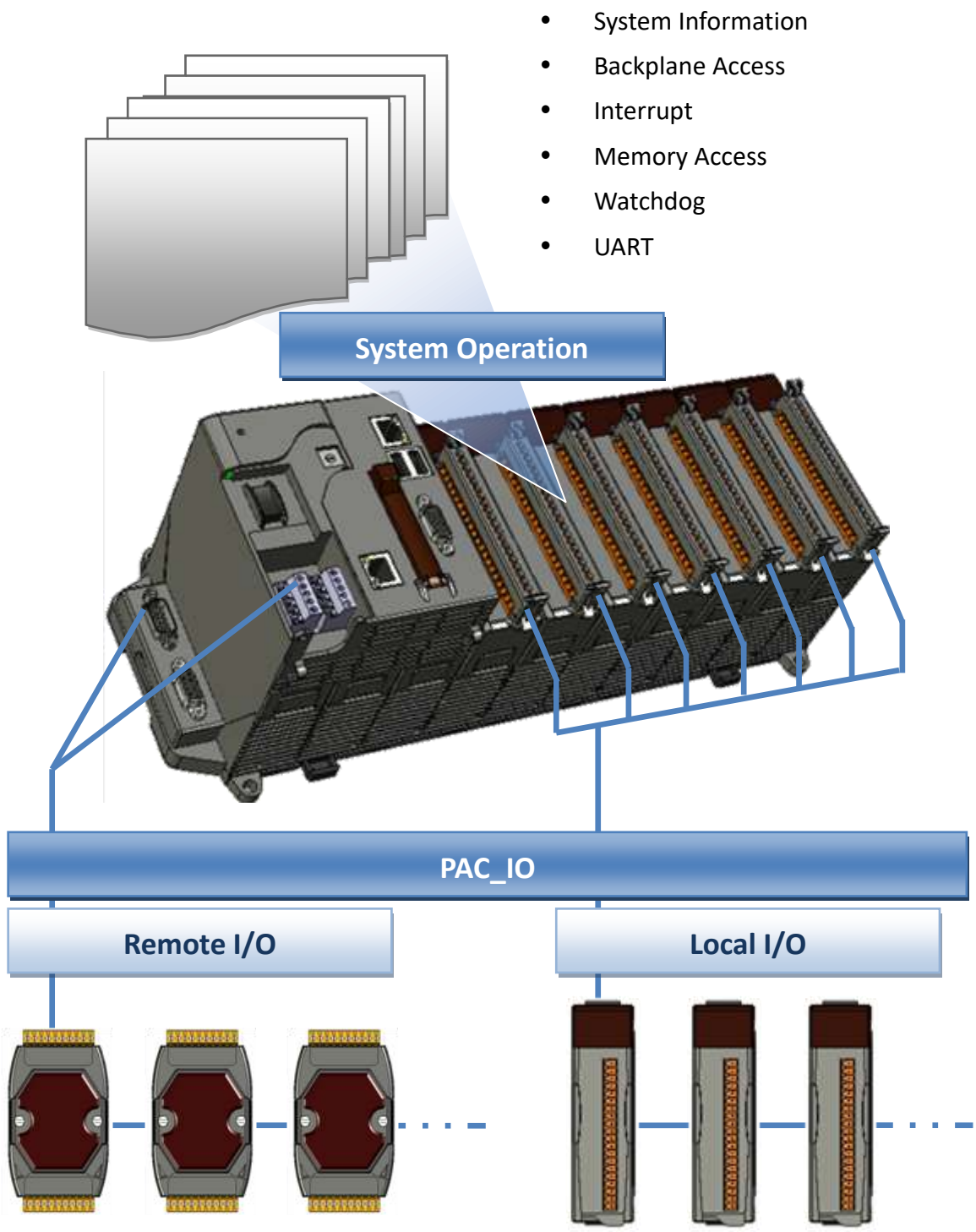
Refer to the chapter 4 on ViewPAC user manual for more details

ftp://ftp.icpdas.com/pub/cd/winpac_am335x/vp-x231/document/

3. PAC API Functions

The diagram below shows the set of each PAC API provided in the PACSDK.

The PACSDK API consists of the following APIs and functional categories



3.1. System Information API (System API)

The system information functions and messages describe or change the system configuration, settings, and attributes.

Supported PACs

The table below lists the system information functions that are supported by each PAC.

Models / Functions	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_GetModuleName	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetRotaryID	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetSerialNumber	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetSDKVersion	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_ChangeSlot	Y	-	Y	-	Y	Y	Y	-	Y
pac_CheckSDKVersion	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_ModuleExists	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetOSVersion	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetMacAddress	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_ReBoot	Y	Y	Y	Y	Y	Y	Y	Y	Y

Models / Functions	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_GetCPUVersion	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_EnableLED	Y	Y	Y	Y	-	-	Y	Y	Y
pac_EnableLEDs	-	-	-	-	-	Y	-	-	-
pac_BackwardCompatible()	Y	-	-	-	-	-	Y	-	-
pac_GetEbootVersion	Y	Y	Y	Y	-	-	Y	Y	Y
pac_GetComMapping	Y	Y	-	Y	-	-	Y	Y	-
pac_GetModuleType	Y	Y	Y	-	Y	Y	Y	Y	Y
pac_GetPacNetVersion	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_BuzzerBeep	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetBuzzerFreqDuty	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SetBuzzerFreqDuty	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_StopBuzzer	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetDIPSwitch	Y	-	-	-	Y	Y	Y	-	-
pac_GetSlotCount	Y	-	Y	-	Y	Y	Y	-	Y
pac_EnableRetrigger	Y	-	Y	-	Y	Y	Y	-	Y
pac_GetBackplaneID	Y	-	Y	-	Y	Y	Y	-	Y
pac_GetBatteryLevel	Y	-	Y	-	Y	Y	Y	-	Y
pac_RegistryHotPlug (Beta version)	-	-	-	-	-	-	-	-	-

Models / Functions	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_UnregistryHotPlug (Beta version)	-	-	-	-	-	-	-	-	-
pac_SetBackLight	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetBackLight	Y	Y	Y	Y	Y	Y	Y	Y	Y

System Information Functions

The table below lists the functions that are used to retrieve or set system information.

PACSDK Functions	PACNET Functions	Description
pac_GetModuleName	Sys.GetModuleName	retrieves the name of the I/O module plugged into the specified I/O slot.
pac_GetRotaryID	Sys.GetRotaryID	retrieves the value of the rotary switch.
pac_GetSerialNumber	Sys.GetSerialNumber	retrieves the hardware ID/serial number.
pac_GetSDKVersion	Sys.GetSDKVersion	retrieves the version number of the current PACSDK.dll.
pac_ChangeSlot	Sys.ChangeSlot	handles the slot from one to another.
pac_CheckSDKVersion	Sys.CheckSDKVersion	checks if the specified version is the installed version of the PACSDK.
pac_ModuleExists	Sys.ModuleExists	specifies whether the I/O module exist or not, and is at local or remote.
pac_GetOSVersion	Sys.GetOSVersion	retrieves the version number of the current operating system (OS).
pac_GetMacAddress	Sys.GetMacAddress	retrieves the MAC address of Ethernet adapter.
pac_ReBoot	Sys.ReBoot	reboots the OS.
pac_GetCPUVersion	Sys.GetCPUVersion	retrieves the version number of the CPU.
pac_EnableLED	Sys.EnableLED	sets the state of the LED.
pac_EnableLEDs	Sys.EnableLEDs	sets the state of the specified LED.
pac_BackwardCompatible()	Sys.BackwardCompatible	determines whether the operating system work in backward compatible mode or not.
pac_GetEbootVersion	Sys.GetEbootVersion	retrieves the version number of the eboot.
pac_GetComMapping	Sys.GetComMapping	retrieves the index mapping of COM port.
pac_GetModuleType	Sys.GetModuleType	retrieves the type of I/O modules

PACSDK Functions	PACNET Functions	Description
pac_GetPacNetVersion	Sys.GetPacNetVersion	retrieves the version number of the PACNET.dll.
pac_BuzzerBeep	Sys.BuzzerBeep	generates simple tones on the speaker.
pac_GetBuzzerFreqDuty	Sys.GetBuzzerFreqDuty	retrieves the frequency value and duty cycle value of the buzzer.
pac_SetBuzzerFreqDuty	Sys.SetBuzzerFreqDuty	sets the frequency value and duty cycle value of the buzzer.
pac_StopBuzzer	Sys.StopBuzzer	stops the buzzer.
pac_GetDIPSwitch	Sys.GetDIPSwitch	retrieves the dip switch.
pac_GetSlotCount	Sys.GetSlotCount	retrieves the total number of the I/O slot.
pac_EnableRetrigger	Sys.EnableRetrigger	determines the retrigger status.
pac_GetBackplaneID	Sys.GetBackplaneID	retrieves the backplane ID.
pac_GetBatteryLevel	Sys.GetBatteryLevel	retrieves the battery status of the backplane.
pac_RegistryHotPlug(Beta version)	Sys.RegistryHotPlug (Beta version)	registers the registry after turning on the hot plug.
pac_UnregistryHotPlug(Beta version)	Sys.UnregistryHotPlug (Beta version)	deletes the registry key after turning off the hot plug.
pac_SetBackLight	Sys.SetBackLight	sets the value of the backlight.
pac_GetBackLight	Sys.GetBackLight	retrieves the value of the backlight.

3.1.1. pac_GetModuleName

This function retrieves the name of the I/O module plugged into the specified I/O slot on XPAC/WinPAC platform.

Syntax

C++

```
int pac_GetModuleName(
    BYTE slot,
    LPSTR strName
);
```

Parameters

Slot

[in] Specifies the slot number that I/O module is plugged into.

strName

[out] A pointer to a buffer that receives the name of the specified I/O module.

Return Value

If the 8K I/O module is undefined, the return value is undefined.

Examples

[C]

```
byte slot = 1;
char strName[10] ;
pac_GetModuleName(slot, strName);
```

[C#]

```
byte slot = 1;
string strName;
int ModuleType = 0;
ModuleType = PACNET.Sys.GetModuleName(slot, ref strName);
//ref. because of calling by reference, you should add key word, ref.
```

```
// For this API, there are two overloading for .NET. First is above, another is below,
// whose return value is Module Name, not Module type.

byte slot = 1;
string strName;
strName = PACNET.Sys.GetModuleName(slot);
```

3.1.2. pac_GetRotaryID

This function retrieves the value of the rotary switch.

Syntax

C++

```
int pac_GetRotaryID();
```

Parameters

This function has no parameters.

Return Value

If the function succeeds, the return value is the position number of the rotary switch. If the function fails, the return value is invalid value. To get extended error information, call pac_GetLastError.

Examples

[C]

```
int RotaryID;
RotaryID = pac_GetRotaryID();
```

[C#]

```
int RotaryID;
RotaryID = PACNET.Sys.GetRotaryID();
```

3.1.3. pac_GetSerialNumber

This function retrieves the hardware ID/serial number of XPAC/WinPAC platform.

Syntax

C++

```
void pac_GetSerialNumber(
  LPSTR SerialNumber
);
```

Parameters

SerialNumber

[out] The hardware ID/serial number.

Return Value

This function does not return a value.

Examples

[C]

```
char SN [32] ;
pac_GetSerialNumber(SN);
```

[C#]

```
string SN;
SN = PACNET.Sys.GetSerialNumber();
```

Remarks

If the retrieved value is null, means the function executes failure or the device is not valid product.

3.1.4. pac_GetSDKVersion

This function retrieves the version number of the current PACSDK.dll.

Syntax

C++

```
void pac_GetSDKVersion(
    LPSTR sdk_version
);
```

Parameters

sdk_version

[out] The version number of the PACSDK.

Return Value

This function does not return a value.

Examples

[C]

```
char SDK[32];
pac_GetSDKVersion(SDK);
```

[C#]

```
string PacSDK;
string PacNET;
PacSDK = PACNET.Sys.GetPacSDKVersion(); //retrieving PACSDK.dll version
PacNET = PACNET.Sys.GetPacNetVersion(); //retrieving PACNET.dll version
//In .net, ths API is different with VC. And there are two API, //pac_GetPacSDKVersion and pac_GetPacNetVersion.
```

3.1.5. pac_ChangeSlot

This function handles the slot from one to another 87k module. The 87K module on the slot 0~x can be controlled after use the function.

Syntax

C++

```
void pac_ChangeSlot(
    BYTE slotNo
);
```

Parameters

slotNo

[in] Specifies the slot number that 87K module is plugged into.

Return Value

This function does not return a value.

Examples

[C]

```
BYTE slot;
HANDLE hPort;
BOOL RET;
char buf[Length] ;
hPort = uart_Open("");
pac_ChangeSlot(slot);
// Change to the slot that 87k module is plugged into
RET = uart_SendCmd(hPort, "$ 00M", buf);
// $00M: ask the device name
uart_Close(hPort);
```

[C#]

```
BYTE slot;
IntPtr hPort;
bool ret;
string BUF;
hPort = PACNET.UART.Open(PACNET.MISC.AnsiString(""));
PACNET.Sys.ChangeSlot(slot);
// Change to the slot that 87k module is plugged into
RET = PACNET.UART.SendCmd(hPort, PACNET.MISC.AnsiString("$ 00M"), BUF);
PACNET.UART.Close(hPort);
```

Remarks

When you use uart API and the I/O modules(87K series) located as slots.

You have to call pac_ChangeSlot to change the slot.

Besides, other low level operations may use pac_ChangeSlot to change the slot.

If you just use 8K series module, you needn't care about this.

3.1.6. pac_CheckSDKVersion

This function checks if the specified version is the installed version of the PACSDK.

This function does not support all versions of XPACSDK and WinPACSDK.

Syntax

C++

```
BOOL pac_CheckSDKVersion(
    DWORD Version
);
```

Parameters

version

[in] The version number of the PACSDK.

If the version number is 1.0.0.1. or above, this parameter must be **0x01000001**

Return Value

If the specified version number is eailer than the currently used PACSDK.dll, , the return value is TRUE.

If the specified version number is later than the currently used PACSDK.dll, the return value is FALSE.

Examples

[C]

```
BOOL bVersion;
bVersion = pac_CheckSDKVersion(0x01000001);
//if your application should use newer than version 1.0.0.1
If(!bVersion)
{
   MessageBox("The XPacSDK.dll version is wrong");
   //display some warning and close the application
}
```

[C#]

```
BOOL bVersion;
bVersion = PACNET.Sys.CheckSDKVersion(0x01000001);
//if your application should use newer than version 1.0.0.1
If(!bVersion)
{
   MessageBox.show("The XPacSDK.dll version is wrong");
   //display some warning and close the application
}
```

3.1.7. pac_ModuleExists

This functions specifies whether the I/O module exist or not, and is at local or remote.

Syntax

C++

```
BOOL pac_ModuleExists(
    HANDLE hPort,
    BYTE slot
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open().

If the I/O module is at local, this parameter must be **0**.

slot

[in] The slot in which module is to check exists or not.

If the I/O module is at remote, this parameter is a value of I/O module ID.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call pac_GetLastError.

Examples

[C]

```
// check a module which is in the slot 5
BOOL bExist;
bExist = pac_ModuleExists(0, 5); // check a 8k/9k module which is in the slot 5
HANDLE hPort= uart_Open("");
bExist = pac_ModuleExists(hPort, 5); // check a 87k/97k module which is in the slot 5
uart_Close(hPort);
if(bExist)
{
   MessageBox("The module exist!");
}
else
{
   MessageBox("The module exist!");
}
```

[C#]

```
//check a module which is in the slot 5
BOOL bExist;
bExist = PACNET.Sys.ModuleExists(0, 5); // check a 8k/9k module which is in the slot 5
IntPtr hPort= PACNET.UART.Open(PACNET.MISC.AnsiString(""));
bExist = PACNET.Sys.ModuleExists(hPort , 5); // check a 87k/97k module which is in the slot 5

PACNET.UART.Close(hPort);
if(bExist)
{
   MessageBox.show("The module exist!");
}
else
{
   MessageBox.show("The module exist!");
}
```

3.1.8. pac_GetOSVersion

This function retrieves the version number of the current operating system (OS).

Syntax

C++

```
void pac_GetOSVersion(
    LPSTR OS_VERSION
);
```

Parameters

OS_VERSION

[out] The version number of the OS.

Return Value

This function does not return a value.

Examples

[C]

```
char OS[32] ;
pac_GetOSVersion(OS);
```

[C#]

```
string OS;
OS = PACNET.Sys.GetOSVersion();
```

3.1.9. pac_GetMacAddress

This function retrieves the MAC address of Ethernet adapter.

Syntax

C++

```
void pac_GetMacAddress(
    BYTE LAN,
    LPSTR MacAddr
);
```

Parameters

LAN

[in] Specifies the LAN number which you want to use.

MacAddr

[out] Retrieves the MAC address.

Return Value

This function does not return a value.

Examples

[C]

```
char MAC [32] ;
BYTE LAN= 1;
pac_GetMacAddress(LAN, MAC);
```

[C#]

```
byte LAN = 1;
String MAC;
MAC = PACNET.Sys.GetMacAddress(LAN);
```

3.1.10. pac_ReBoot

This function reboots the OS (**warm restatr**).

Syntax

C++

```
void pac_ReBoot();
```

Parameters

This function has no parameters.

Return Value

This function does not return a value.

Examples

[C]

```
pac_ReBoot();
```

[C#]

```
PACNET.Sys.Reboot();
```

3.1.11. pac_GetCPUVersion

This function retrieves the version number of the CPU.

Syntax

C++

```
void pac_GetCPUVersion(
    LPSTR cpu_version
);
```

Parameters

cpu_version

[out] The version number of the CPU.

Return Value

This function does not return a value.

Examples

[C]

```
char CPU [32] ;
pac_GetCPUVersion(CPU);
```

[C#]

```
string CPU = PACNET.Sys.GetCPUVersion();
```

3.1.12. pac_EnableLED

This function sets the on/off state of the LED.

Syntax

C++

```
void pac_EnableLED(
    BOOL bFlag
);
```

Parameters

bFlag

Specifies the mode of the LED.

True: Turn on the LED

False: Turn off the LED

Return Value

This function does not return a value.

Examples

[C]

```
pac_EnableLED(true);
```

[C#]

```
PACNET.Sys.EnableLED(true);
```

3.1.13. pac_EnableLEDs

This function sets the on/off state of the LEDs.

Syntax

C++

```
void pac_EnableLEDs(
    INT pin,
    BOOL bFlag
);
```

Parameters

pin

Specifies the LED.

0: L1 LED

1: L2 LED

bFlag

Specifies the mode of the LED.

True: Turn on the LED

False: Turn off the LED

Return Value

This function does not return a value.

Examples

[C]

```
pac_EnableLEDs(0, TRUE);
```

[C#]

```
PACNET.Sys.EnableLEDs(0, TRUE);
```

3.1.14. pac_BackwardCompatible()

This function determines whether the operating system work in backward compatible mode or not.

Syntax

C++

```
Bool pac_BackwardCompatible();
```

Parameters

This function has no parameters

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call pac_GetLastError.

Remark

The function is only applied to the WP-8000 sereis and it is used for back-compatible with old PAC controller, WinCon series.

Examples

[C]

```
Bool BBC;
BBC = pac_BackwardCompatible;
```

[C#]

```
Bool BBC;
BBC = PACNET.Sys.BackwardCompatible();
```

3.1.15. pac_GetEbootVersion

This function retrieves the version number of the eboot.

Syntax

C++

```
void pac_GetEbootVersion(
    LPSTR eboot_version
);
```

Parameters

eboot_version

[out] The version number of the eboot.

Return Value

This function does not return a value.

Examples

[C]

```
char Eboot[32] ;
pac_GetEbootVersion(Eboot);
```

[C#]

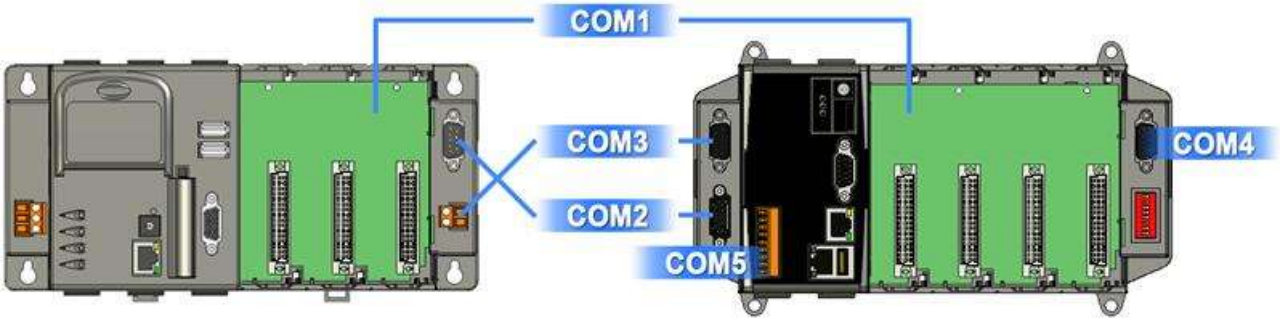
```
string Eboot;
EBOOT = PACNET.Sys.GetEbootVersion();
```

3.1.16. pac_GetComMapping

This function retrieves the index mapping of COM port.

[Normal mode]

[Backward-Compatible mode]



Syntax

C++

```
int pac_GetComMapping(
    int NDX
);
```

Parameters

ndx

[in] Specifies which slot COM port map between backward compatible or not.

Return Value

Return the index of COM port under the backward compatible, if backward compatible is running. Otherwise, return the index of COM port in the normal mode.

Remarks

The function is only applied to the WP-8000 sereis and it is used for back-compatible with old PAC controller, WinCon series.

Examples

[eVC]

```
int currentCOM; // current com port
int normalCOM; // com port index in normal mode
currentCOM = pac_GetComMapping(normalCOM);
// If the device is running on normal mode, then the return value, currentCOM,
//equals normalCOM
// ex: normalCOM = 0; after this API, the currentCOM = 0, too.
// Otherwise, if the device is running on backward compatible mode, then the
//return value, currentCOM, is backward compatible mapping index
// ex: normalCOM = 0; after this API, the currentCOM = 1
```

[C#]

```
int currentCOM;
int normalCOM;
currentCOM = PACNET.Sys.GetComMapping(normalCOM);
```

3.1.17. pac_GetModuleType

This function retrieves the type of I/O modules which plugged into the WinPAC/XPAC series devices.

Syntax

C++

```
int pac_GetModuleType(
    int slot
);
```

Parameters

slot

[in] Specifies the slot number that I/O module is plugged into.

Return Value

For WinPAC-8000/ViewPAC Series

The following table shows the defined values.

Value	Description
0	No module existed
0x80	Genernal I-8000W module
0x81	I-8000RW module (R version: Provide PowerOn and Safe value)
0xE3	I-8000W module with 32 DI channels
0xE0	I-8000W module with 32 DO channels
0xE2	I-8000W module with 16 DI channels and 16 DO channels
0xC3	I-8000W module with 16 DI channels
0xC0	I-8000W module with 16 DO channels
0xC2	I-8000W module with 8 DI channels and 8 DO channels
0x40	No module defined

For WinPAC-5000 Series

The following table shows the defined values.

Value	Description
0	No XW board existed
0x80	Genernal XW board
0xE3	XW board with 32 DI channels
0xE0	XW board with 32 DO channels
0xE2	XW board with 16 DI channels and 16 DO channels
0xC3	XW board with 16 DI channels
0xC0	XW board with 16 DO channels
0xC2	XW board with 8 DI channels and 8 DO channels
0x40	No XW board defined

Examples

[C]

```
int iDIO_Slot = 1;
int Type= 0;
Type = pac_GetModuleType(iDIO_Slot);
if(Type == 0xE2 | | Type == 0xC2){
   //The module is DIO module
   ...
}
```

[C#]

```
byte slot = 1;
int ModuleType = 0;
ModuleType = PACNET.Sys.GetModuleType(slot);
if(ModuleType == 0xE2 | | ModuleType == 0xC2){
   //The module is DIO module
   ...
}
```

3.1.18. pac_GetPacNetVersion

This function retrieves the version number of the PACNET.dll.

Syntax

C++

```
string pac_GetPacNetVersion();
```

Parameters

dll_version

[out] The version number of the PACNET.dll.

Return Value

This function does not return a value.

Examples

[C#]

```
string PacNet;
PacNet = PACNET.Sys.GetPacNetVersion();
```

3.1.19. pac_BuzzerBeep

This function generates simple tones on the speaker.

Syntax

C++

```
void pac_BuzzerBeep(
    WORD count,
    DWORD milliseconds
);
```

Parameters

count

[in] Specifies the times of beep.

milliseconds

[in] Specifies the interval time, in milliseconds.

Return Value

This function does not return a value.

Examples

[C]

```
pac_BuzzerBeep(1, 100);
```

[C#]

```
PACNET.Sys.Buzzer.BuzzerBeep(1, 100);
```

3.1.20. pac_GetBuzzerFreqDuty

This function retrieves the frequency value and duty cycle value of the buzzer.

Syntax

C++

```
void pac_GetBuzzerFreqDuty(
    int *freq,
    int *duty
);
```

Parameters

freq

[out] The frequency of the sound.

duty

[out] The duration of the sound.

Return Value

This function does not return a value.

Examples

[C]

```
INT FQ = 0;
INT DU = 0;
pac_GetBuzzerFreqDuty(&fq, &du);
```

[C#]

```
INT FQ = 0;
INT DU = 0;
PACNET.Sys.Buzzer.GetBuzzerFreqDuty(ref fq, ref du);
```

3.1.21. pac_SetBuzzerFreqDuty

This function sets the frequency value and duty cycle value of the buzzer.

Syntax

C++

```
void pac_SetBuzzerFreqDuty(
    int freq,
    int duty
);
```

Parameters

freq

[out] The frequency of the sound.

duty

[out] The duration of the sound.

Return Value

This function does not return a value.

Examples

[C]

```
INT FQ = 500;
INT DU = 20;
pac_GetBuzzerFreqDuty(FQ , DU);
```

[C#]

```
INT FQ = 500;
INT DU = 20;
PACNET.Sys.Buzz.GetBuzzerFreqDuty(FQ , DU);
```

3.1.22. pac_StopBuzzer

This function stops the buzzer.

Syntax

C++

```
void pac_StopBuzzer();
```

Parameters

This function has no parameters.

Return Value

This function does not return a value.

Examples

[C]

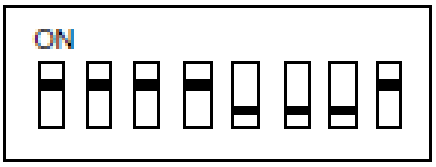
```
pac_StopBuzzer();
```

[C#]

```
PACNET.Sys.Buzzer.StopBuzzer();
```

3.1.23. pac_GetDIPSwitch

This function retrieves the dip switch.



Syntax

C++

```
int pac_GetDIPSwitch();
```

Parameters

This function has no parameters.

Return Value

The return value specifies the dip switch.

Examples

[C]

```
int iDipSwitch;
iDipSwitch = pac_GetDIPSwitch();
```

[C#]

```
int iDipSwitch;
iDipSwitch = PACNET.Backplane.GetDIPSwitch();
```

3.1.24. pac_GetSlotCount

This function retrieves the total numbers of the I/O slot.

Syntax

C++

```
int pac_GetSlotCount();
```

Parameters

This function has no parameters.

Return Value

The return value is the numbers of the I/O slot.

Examples

[C]

```
int wSlot;
wSlot = pac_GetSlotCount();
```

[C#]

```
int wSlot;
wSlot = PACNET.Backplane.GetSlotCount();
```

3.1.25. pac_EnableRetrigger

This function determines the retrigger status.

Syntax

C++

```
void pac_EnableRetrigger(
    BYTE iValues
);
```

Parameters

iValues

[in] Specifies the retrigger value, 0~255, unit= 10 microsecond. (0 means disable retrigger function)

Return Value

This function has does not return a value.

Examples

This function has no examples.

Remarks

The retrigger mechanism is used when the below situation occurred.

If an interrupt is sent but not be serviced, the retrigger function will send an interrupt again. This operation will continue until the interrupt has been serviced.

3.1.26. pac_GetBackplaneID

This function retrieves the backplane ID.

Syntax

C++

```
void pac_GetBackplaneID(
    LPSTR backplane_version
);
```

Parameters

backplane_version

[out] Retrieves the backplane ID.

Return Value

This function has does not return a value.

Examples

[C]

```
char Backplane[32] ;
pac_GetBackplaneID(Backplane);
```

[C#]

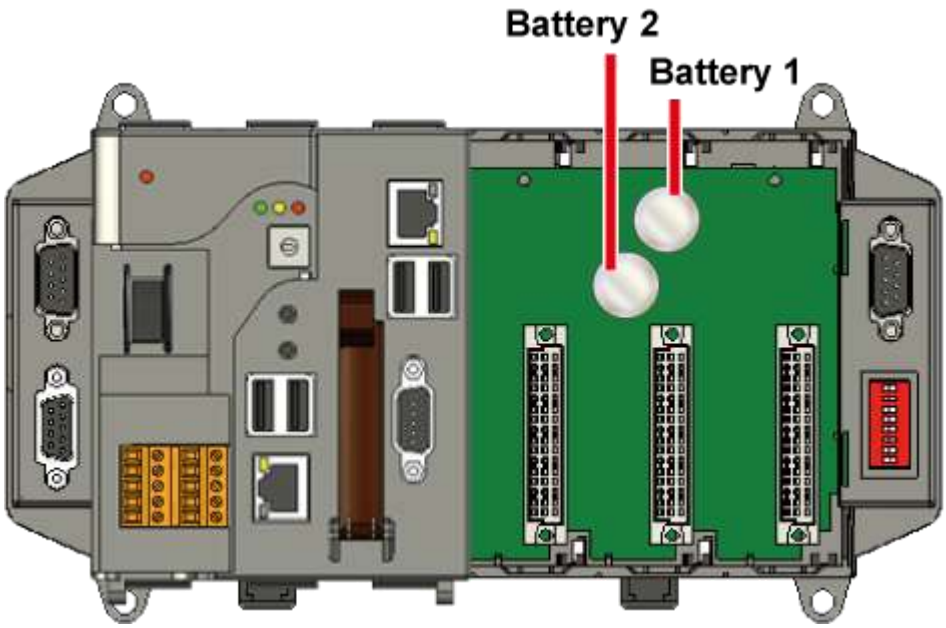
```
string Backplane= PACNET.Backplane.GetBackplaneID();
```

3.1.27. pac_GetBatteryLevel

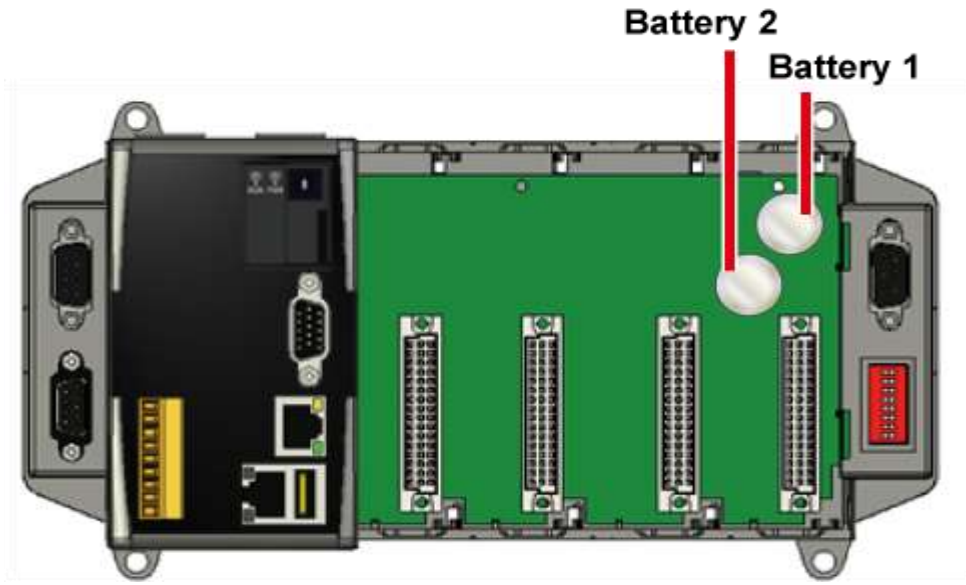
This function retrieves the battery status of the backplane.

This function supports the following series models.

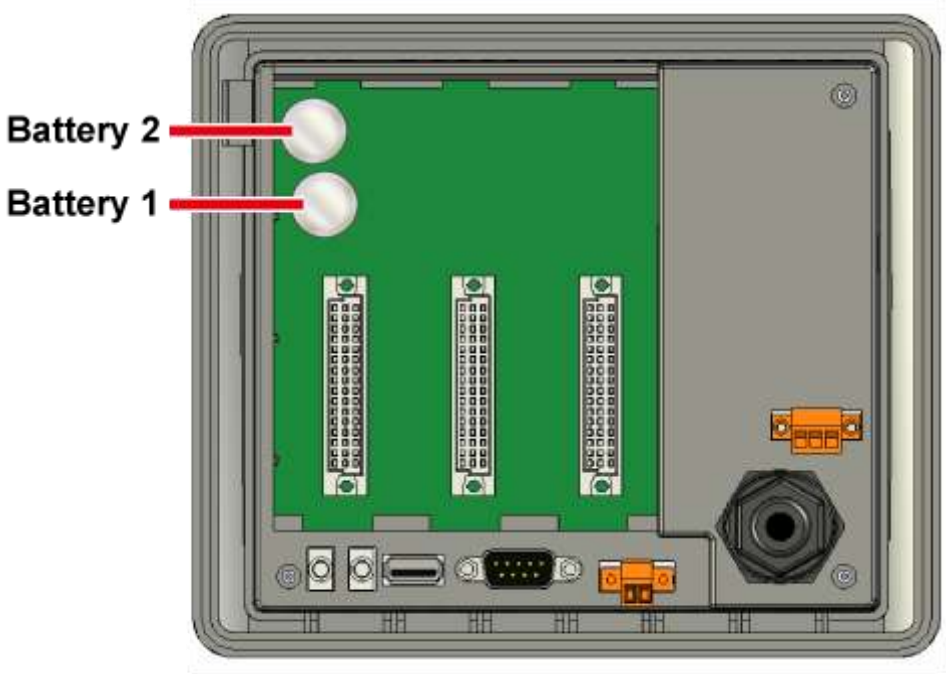
XPAC



WinPAC



ViewPAC



Syntax

C++

```
int pac_ GetBatteryLevel(
    int nBattery
);
```

Parameters

nBattery

[in] Specifies the index of battery.

1 means first battery.

2 means second battery.

3 means RTC battery. (For XPAC_Atom series only)

Return Value

1 means high voltage.

0 means low voltage. (for XP-8000-CE6 and XP-8000-Atom-CE6 series only)

2 means low voltage. (for WP-8000 series only)

Examples

[C]

```
int nBattery;
INT index= 1;
nBattery = pac_GetBatteryLevel(index);
```

[C#]

```
int nBattery;
INT index= 1;
nBattery = PACNET.Backplane.GetBatteryLevel(index);
```

3.1.28. pac_RegistryHotPlug(Beta Version)

This function registers the registry after turning on the hot plug.

Syntax

C++

```
void pac_RegistryHotPlug(
    DWORD 的 hWnd ·
    DWORD MSGID
);
```

Parameters

hWnd

[in] Specifies the handle ID.

Return Value

This function does not return a value.

Examples

This function has no examples.

3.1.29. pac_UnregistryHotPlug(Beta Version)

This function deletes the registry key after turning off the hot plug.

Syntax

C++

```
void pac_UnregistryHotPlug(
    DWORD hWnd
);
```

Parameters

hWnd

[in] Specifies the handle ID.

Return Value

This function does not return a value.

Examples

This function has no examples.

3.1.30. pac_SetBackLight

This function sets the back-light value of the "ViewPAC" series.

Syntax

C++

```
void pac_SetBackLight(
    int level
);
```

Parameters

level

[out] bright value: 0 ~ 100.

Return Value

This function has does not return a value.

Examples

[C]

```
pac_SetBackLight(level);
```

[C#]

```
PACNET.Sys.pac_SetBackLight(level)
```

3.1.31. pac_GetBackLight

This function retrieves the back-light value of the "ViewPAC" series.

Syntax

C++

```
DWORD pac_GetBackLight( );
```

Parameters

This function has no parameters.

Return Value

The return value is the value of the backlight.

Examples

[C]

```
DWORD Bright;
Bright = pac_GetBackLight();
```

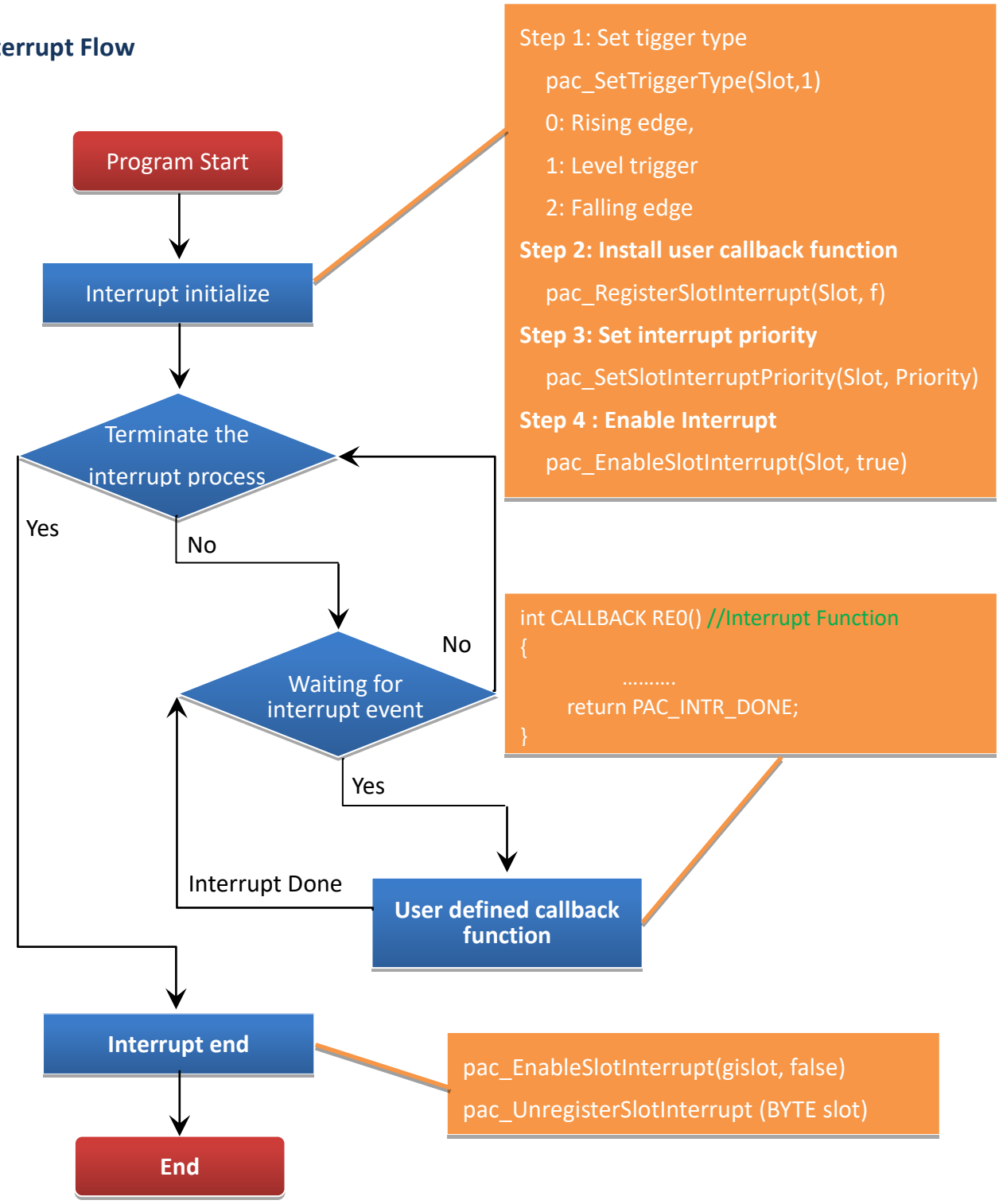
[C#]

```
Int Bright;
Bright = PACNET.Sys.pac_GetBackLight();
```

3.2. Interrupt API (System API)

The Interrupt functions provide the slot interrupt that may be used for counting, timing, detecting external events, and sending and receiving data using the serial interface

Interrupt Flow



Supported PACs

The table below lists the interrupt functions that are supported by each PAC.

Functions \ Models	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_RegisterSlotInterrupt	Y	-	Y	-	Y	Y	Y	-	Y
pac_UnregisterSlotInterrupt	Y	-	Y	-	Y	Y	Y	-	Y
pac_EnableSlotInterrupt	Y	-	Y	-	Y	Y	Y	-	Y
pac_SetSlotInterruptPriority	Y	-	Y	-	Y	Y	Y	-	Y
pac_InterruptInitialize	Y	-	Y	-	Y	Y	Y	-	Y
pac_GetSlotInterruptEvent	Y	-	Y	-	Y	Y	Y	-	Y
pac_SetSlotInterruptEvent	Y	-	Y	-	Y	Y	Y	-	Y
pac_SetTriggerType	Y	-	Y	-	Y	Y	Y	-	Y
pac_GetSlotInterruptID	Y	-	Y	-	Y	Y	Y	-	Y
pac_InterruptDone	Y	-	Y	-	Y	Y	Y	-	Y

Interrupt Functions

The following functions are used to retrieve or set the slot interrupt.

PACSDK Functions	PACNET Functions	Description
pac_RegisterSlotInterrupt	Interrupt.RegisterSlotInterrupt	registers the slot interrupt service route after turning on the slot interrupt.
pac_UnregisterSlotInterrupt	Interrupt.UnregisterSlotInterrupt	unregisters slot interrupt service route and disables a hardware interrupt as specified by its interrupt identifier.
pac_EnableSlotInterrupt	Interrupt.EnableSlotInterrupt	performs hardware operations necessary to enable the specified hardware interrupt.
pac_SetSlotInterruptPriority	Interrupt.SetSlotInterruptPriority	sets the priority for a real-time thread on a thread by thread basis.
pac_InterruptInitialize	Interrupt.InterruptInitialize	initializes a slot interrupt with the kernel. This initialization allows the slot to register an event and enable the interrupt.
pac_GetSlotInterruptEvent	Interrupt.GetSlotInterruptEvent	retrieves the slot event handle which registered by pac_InterruptInitialize.
pac_SetSlotInterruptEvent	Interrupt.SetSlotInterruptEvent	sets the priority for a real-time thread on a thread by thread basis.
pac_SetTriggerType	Interrupt.SetTriggerType	assigns the pulse trigger type for separate slot.
pac_GetSlotInterruptID	Interrupt.GetSlotInterruptID	retrieves the ID of the slot interrupt.
pac_InterruptDone	Interrupt.InterruptDone	signals to the kernel that interrupt processing has been completed.

3.2.1. pac_RegisterSlotInterrupt

This function registers the slot interrupt service route after turning on the slot interrupt.

Syntax

C++

```
BOOL pac_RegisterSlotInterrupt(
    BYPTE slot,
    PAC_CALLBACK_FUNC f
);
```

Parameters

slot

[in] Specifies the index of slot.

f

A call back function.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call pac_GetLastError.

Examples

[C]

```
Int slot= 3; //slot3
int CALLBACK slot_callback_proc()
{
   return true;
// if return true, SDK will do pac_InterruptDone automatically
// else, users should do pac_InterruptDone by themselves if needed.
// if interrupt type is level trigger, no matter return true or flase,
// needn't add pac_InterruptDone and it will work correctly.

}
void CIntrDlg :: OnButton1()
{
   pac_RegisterSlotInterrupt(slot , slot_callback_proc);
   pac_EnableSlotInterrupt(slot , TRUE); // enable slot interrupt
}

void CIntrDlg :: OnButton2()
{
   pac_EnableSlotInterrupt(slot , FALSE); // disable slot interrupt
   pac_UnregisterSlotInterrupt(slot); // unregister slot interrupt
}
```

Remarks (for XPAC series only)

Default trigger type is level trigger.

For XP-8000 series, only support level trigger type.

3.2.2. pac_UnregisterSlotInterrupt

This function unregisters slot interrupt service route and disables a hardware interrupt as specified by its interrupt identifier.

Syntax

C++

```
BOOL pac_UnregisterSlotInterrupt(
    BYTE slot
);
```

Parameters

slot

[in] Specifies the index of slot.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call pac_GetLastError.

Examples

[C]

```
Int slot= 3; //slot3
int CALLBACK slot_callback_proc()
{
   // do something
   pac_InterruptDone(slot);
   return false;
   // if return true, SDK will do pac_InterruptDone automatically
   //else, users should do pac_InterruptDone by themselves if needed
   // if interrupt type is level trigger, no matter return true or flase,
   // needn't add pac_InterruptDone and it will work correctly.
}
void CIntrDlg :: OnButton1()
{
   pac_RegisterSlotInterrupt(slot , slot_callback_proc);
   pac_EnableSlotInterrupt(slot , true); // enable slot interrupt
}
void CIntrDlg :: OnButton2()
{
   pac_EnableSlotInterrupt(slot , false); // disable slot interrupt
   pac_UnregisterSlotInterrupt(slot); // unregister slot interrupt
}
```

3.2.3. pac_EnableSlotInterrupt

This function performs hardware operations necessary to enable the specified hardware interrupt.

Syntax

C++

```
void pac_EnableSlotInterrupt(
    BYTE slot,
    BOOL bEnable
);
```

Parameters

slot

[in] Specifies the index of slot to enable interrupt or disable.

bEnable

[in] Specifies the Slot interrupt turning on or not.

Return Value

This function has does not return a value.

Examples

[C]

```
int slot= 3; //slot3
int CALLBACK slot_callback_proc()
{
   // do something
   pac_InterruptDone(slot);
   return true;
   // if return true, SDK will do pac_InterruptDone automatically
   //else, users should do pac_InterruptDone by themselves if needed
   // if interrupt type is level trigger, no matter return true or flase,
   // needn't add pac_InterruptDone and it will work correctly.

}
void CIntrDlg :: OnButton1()
{
   pac_RegisterSlotInterrupt(slot , slot_callback_proc);
   pac_EnableSlotInterrupt(slot , true); // enable slot interrupt
}

void CIntrDlg :: OnButton2()
{
   pac_EnableSlotInterrupt(slot , false); // disable slot interrupt
   pac_UnregisterSlotInterrupt(slot); // unregister slot interrupt
}
```

Remarks (for XPAC series only)

Default trigger type is level trigger.

For XP-8000 series, only support level trigger type.

3.2.4. pac_SetSlotInterruptPriority

This function sets the priority for a real-time thread on a thread by thread basis.

Syntax

C++

```
BOOL pac_SetSlotInterruptPriority(
    BYTE slot,
    int nPriority
);
```

Parameters

slot

[in] Specifies the index of slot to set priority.

nPriority

[in] Priorities to set for the thread.

This value can range from 0 through 255, with 0 as the highest priority.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call pac_GetLastError.

Examples

This function has no examples.

3.2.5. pac_InterruptInitialize

This function initializes a slot interrupt with the kernel. This initialization allows the slot to register an event and enable the interrupt.

Syntax

C++

```
BOOL pac_InterruptInitialize(
    BYTE slot
);
```

Parameters

slot

[in] Specify the index of slot to initialize.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call pac_GetLastError.

Examples

This function has no examples.

Remarks

Default trigger type is level trigger.

For XP-8000 series, only support level trigger type.

If you want to get the registered event handle, please call this API, pac_GetSlotInterruptEvent.

3.2.6. pac_GetSlotInterruptEvent

This function retrieves the slot event handle which registered by pac_InterruptInitialize.

Syntax

C++

```
HANDLE pac_GetSlotInterruptEvent(
  BYTE slot
);
```

Parameters

slot

[in] Specifies the index of slot to retrieve the event handle.

Return Value

If the function succeeds, this program handles the event object.

If the function fails, the return value is NULL. To get extended error information, call pac_GetLastError.

Examples

This function has no examples.

3.2.7. pac_SetSlotInterruptEvent

This function allows a device driver to assign the slot event handle.

Syntax

C++

```
void pac_SetSlotInterruptEvent(
    BYTE slot,
    HANDLE hEvent
);
```

Parameters

slot

[in] Specifies the index of slot to retrieve the event handle.

hEvent

[in] Event to be signaled.

Return Value

This function has does not return a value.

Examples

This function has no examples.

3.2.8. pac_SetTriggerType

This function assigns the pulse trigger type for separate slot.

Syntax

C++

```
void pac_SetTriggerType(
    BYTE slot,
    int ITYPE
);
```

Parameters

iType

[in] Specifies the pulse trigger type.

0: Rising edge trigger(default)

1: Level trigger

2: Falling edge trigger

Return Value

This function has does not return a value.

Examples

This function has no examples.

Remarks

For XP-8000 series, only support level trigger type.

3.2.9. pac_GetSlotInterruptID

This function retrieves the ID of the slot interrupt.

Syntax

C++

```
DWORD pac_GetSlotInterruptID(
  BYTE slot
);
```

Parameters

slot

[in] Specifies the slot.

Return Value

If the function succeeds, the return value is the ID of the slot interrupt.

If the function fails, the return value is FALSE. To get extended error information, call pac_GetLastError.

Examples

This function has no examples.

3.2.10. pac_InterruptDone

This function signals to the kernel that interrupt processing has been completed.

Syntax

C++

```
Void pac_InterruptDone(
    BYTE slot
);
```

Parameters

slot

[in] Specifies the slot to clear trigger.

Return Value

This function has does not return a value.

Examples

[C]

```
HANDLE hIntr;
BOOL bExit = FALSE;
BYTE slot= 0;
DWORD INTP_Thread(PVOID PCONTEXT){
while(bExit)
{
   WaitForSingleObject(hIntr , INFINITE);
   //Wait the interrupt trigger
   pac_InterruptDone(slot);
}
   pac_EnableSlotInterrupt(slot , false);
   pac_SetSlotInterruptEvent(slot , NULL);
   CloseHandle(pac_GetSlotInterruptEvent(slot));
   return 0;
}

void CInterruptDlg :: OnButton1()
{
   bExit = TRUE;
   pac_InterruptInitialize(slot);
   pac_EnableSlotInterrupt(slot , true);
   hIntr = pac_GetSlotInterruptEvent(slot);
   CreateThread(NULL , 0 , INTP_Thread , &slot , 0 , NULL);
}
```

3.3. Memory Access API

The memory access functions provide the memory management that may be used for reading, writing EEPROM or SRAM, or mounting, ummounting MicroSD.

EEPROM and SRAM application:

Write some data to the EEPROM or SRAM.
The data will keep save after power off.

Supported PACs

The table below lists the Memory Access functions that are supported by each PAC.

Functions \ Models	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_GetMemorySize	Y	Y▲	Y	Y	Y	Y	Y	Y▲	Y
pac_ReadMemory	Y	Y▲	Y	Y	Y	Y	Y	Y▲	Y
pac_WriteMemory	Y	Y▲	Y	Y	Y	Y	Y	Y▲	Y
pac_EnableEEPROM	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SDExists	Y	Y	Y	Y	-	-	Y	Y	Y
pac_SDMount	Y	Y	Y	Y	-	-	Y	Y	Y
pac_SDOnside	Y	Y	Y	Y	-	-	Y	Y	Y
pac_SDUnmount	Y	Y	Y	Y	-	-	Y	Y	Y

▲ WP-5xxx only supports the memory type 1 (EEPROM), not type 0 (SRAM).

Memory Access Functions

The following functions are used to retrieve or set the memory

PACSDK Functions	PACNET Functions	Description
pac_GetMemorySize	Memory.GetMemorySize	retrieves the size of the specified memory.
pac_ReadMemory	Memory.ReadMemory	retrieves data from the specified memory.
pac_WriteMemory	Memory.WriteMemory	stores data in the specified memory.
pac_EnableEEPROM	Memory.EnableEEPROM	sets the states of the EEPROM.
pac_SDExists	Memory.SDExists	checks that the micro SD whether has been standby ready or not.
pac_SDMount	Memory.SDMount	mounts the micro SD if the micro SD has been inserting to the device without mounting.
pac_SDOnside	Memory.SDOnside	checks the Micro SD whether on-side or not.
pac_SDUnmount	Memory.SDUnmount	dismounts Micro SD if the Micro SD has been mounted.

3.3.1. pac_GetMemorySize

This function retrieves the size of the specified memory.

Syntax

C++

```
DWORD pac_GetMemorySize(
    int mem_type
);
```

Parameters

mem_type

[in] Handle to a currently type memory.

PAC_MEM_SRAM 0 (WP-5000 series doesn't support SRAM)

PAC_MEM_EEPROM 1

Return Value

The return value specifies the memory size.

Examples

[C]

```
DWORD mem_size;
mem_size = pac_GetMemorySize(PAC_MEM_SRAM);
```

[C#]

```
uint mem_size;
mem_size = PACNET.Memory.GetMemorySize(0);
```

3.3.2. pac_ReadMemory

This function retrieves data from the specified memory.

Syntax

C++

```
BOOL pac_ReadMemory(
    DWORD address,
    LPBYTE lpBuffer,
    DWORD dwLength,
    int mem_type
);
```

Parameters

address

[in] Specifies the memory address where read from.

EEPROM

0 ~0x1FFF (8KB) for users

0x2000~0x3FFF (8KB) is reserved for the system

SRAM

The size of the input range for the SRAM is only 0 ~0x6FFFF (448KB), with another 64KB of SRAM is reserved for use by the system.

lpBuffer

[in] Receives the memory data.

dwLength

[in] Number of characters to be read.

mem_type

[in] Handle to a currently type memory.

PAC_MEM_SRAM 0 (WP-5000 series doesn't support SRAM)

PAC_MEM_EEPROM 1

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call pac_GetLastError.

A nonzero error code defined in PACERROR.h indicates failure. To get a generic description of the error, call pac_GetErrorMessage. The message resource is optional; therefore, if you call pac_GetErrorMessage it could fail.

Examples

[C]

```
#define LENGTH 2
bool ret;
DWORD address = 0;
BYTE Buffer[LENGTH] ;
ret = pac_ReadMemory(address, Buffer, LENGTH,0);
```

[C#]

```
bool ret;
uint address = 0;
byte[] Buffer = new byte[2] ;
ret = PACNET.Memory.ReadMemory(address, Buffer, 2, 0);
```

Remarks

If an older program is coded to write data to the 0x2000 ~ 0x3FFF address of the EEPROM, or to the last segment of the SRAM using the SDK version 2.0.1.0 or earlier, the program may fail to write the data to the EEPROM or the SRAM using the PACSDK.dll or PACNET.dll.

There are two ways to fix the problem

1. Modify the program so that the data is writhen to the 0~0x1FFF address of the EEPROM or the 0 ~ 0x6FFFF address of the SRAM.

2. Ask for the previous SDK from ICPDAS.

3.3.3. pac_WriteMemory

This function stores data in the specified memory.

Syntax

C++

```
BOOL pac_WriteMemory(
    DWORD address,
    LPBYTE lpBuffer,
    DWORD dwLength,
    int mem_type
);
```

Parameters

Address

[in] Specifies the memory address where write from.

EEPROM

0 ~0x1FFF (8KB) for users

0x2000~0x3FFF (8KB) is reserved for the system

SRAM

The size of the input range for the SRAM is only 0 ~0x6FFFF (448KB), with another 64KB of SRAM is reserved for use by the system.

lpBuffer

[in] A pointer to the buffer containing the data to be written to the memory.

dwLength

[in] Number of characters to be written.

mem_type

[in] Handle to a currently type memory.

PAC_MEM_SRAM 0 (WP-5000 series doesn't support SRAM)

PAC_MEM_EEPROM 1

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call pac_GetLastError.

A nonzero error code defined in PACERROR.h indicates failure. To get a generic description of the error, call pac_GetErrorMessage. The message resource is optional; therefore, if you call pac_GetErrorMessage it could fail.

Examples

[C]

```
#define LENGTH 2
bool ret;
DWORD address = 0;
BYTE Buffer[LENGTH] ;
Buffer[0] = 10;
Buffer[1] = 20;
ret = pac_WriteMemory(address, Buffer, LENGTH, PAC_MEM_SRAM);
```

[C#]

```
int LENGTH=2;
bool ret;
uint address = 0;
byte[] Buffer = new byte[2] { 10, 20 };
ret = PACNET.Memory.WriteMemory(address, Buffer, LENGTH, PAC_MEM_SRAM);
```

Remarks

If an older program is coded to write data to the 0x2000 ~ 0x3FFF address of the EEPROM, or to the last segment of the SRAM using the SDK version 2.0.1.0 or earlier, the program may fail to write the data to the EEPROM or the SRAM using the PACSDK.dll or PACNET.dll..

There are two ways to fix the problem

1. Modify the program so that the data is writhen to the 0~0x1FFF address of the EEPROM or the 0 ~ 0x6FFFF address of the SRAM.
2. Ask for the previous SDK from ICPDAS.

3.3.4. pac_EnableEEPROM

This function disable write protection of EEPROM.

Syntax

C++

```
void pac_EnableEEPROM(
    BOOL bEnable
);
```

Parameters

bEnable

[in] Specifies the mode of the EEPROM.

True: disable write protection of EEPROM

False: enable write protection of EEPROM

Return Value

This function has does not return a value.

Examples

[C]

```
#define LENGTH 2
int ret;
DWORD address = 0;
BYTE Buffer[LENGTH] ;
Buffer[0] =0xAB;
Buffer[1] =0xCD;
pac_EnableEEPROM(true);
ret = pac_WriteMemory(address, Buffer, LENGTH, PAC_MEM_EEPROM);
pac_EnableEEPROM(false) ;
```

[C#]

```
bool ret;
uint address = 0;
byte[] Buffer = new byte[2] {0xAB,0xCD };
PACNET.Memory.EnableEEPROM(true);
ret = PACNET.Memory.WriteMemory(address, Buffer, (uint)Buffer.Length,
PAC_MEM_EEPROM );
PACNET.Memory.EnableEEPROM(false);
```

Remarks

Before writing EEPROM, need turn on the EEPROM write protectio; after written EEPROM, need turn off the EEPROM write protectio.

3.3.5. pac_SDExists

This function checks that the micro SD whether has been standby ready or not.

Syntax

C++

```
bool pac_SDExists();
```

Parameters

This function has no parameters.

ReturnValue

Return TRUE: micro SD card is exist.

Return FALSE: micro SD card is not exist.To get extended error information, call pac_GetLastError.

Examples

[C]

```
bool bExist;
bExist = pac_SDExists();
```

[C#]

```
bool bExist;
bExist = PACNET.Memory.SDExists();
```

3.3.6. pac_SDMount

This function mounts the micro SD.

(if the micro SD has been inserting to the device without mounting.)

Syntax

C++

```
bool pac_SDMount(
    LPTSTR szPartitionName
);
```

Parameters

szPartitionName

[in] Name of the partition. Example Part00.

ReturnValue

If mounts Micro SD card succeed, the return value is TRUE.

If mounts Micro SD card fail, the return value is FALSE. To get extended error information, call pac_GetLastError.

Examples

[C]

```
bool ret;
ret = pac_SDMount(TEXT("Part00"));
```

[C#]

```
bool ret;
ret = PACNET.Memory.SDMount("Part00");
```

3.3.7. pac_SDOnside

This function checks the Micro SD whether on-side or not.

Syntax

C++

```
bool pac_SDOnside();
```

Parameters

szPartitionName

[in] Name of the partition. Example Part00.

ReturnValue

If the Micro SD card on-side, the return value is TRUE.

If the Micro SD card not on-side, the return value is FALSE. To get extended error information, call pac_GetLastError.

Examples

[C]

```
bool ret;
ret = pac_SDOnside();
```

[C#]

```
bool ret;
ret = PACNET.Memory.SDOnside();
```

3.3.8. pac_SDUnmount

This function unmounts Micro SD if the Micro SD has been mounted.

Syntax

C++

```
bool pac_SDUnmount();
```

Parameters

This function has no parameters.

ReturnValue

If unmounts Micro SD succeed, the return value is TRUE.

If unmounts Micro SD fail, the return value is FALSE. To get extended error information, call pac_GetLastError.

Examples

[C]

```
bool ret;
ret = pac_SDUnmount();
```

[C#]

```
bool ret;
ret = PACNET.Memory.SDUnmount();
```

3.4. Watchdog API

A watchdog timer is an electronic timer used to detect and recover from faults in the PAC. In normal, the computer PAC resets the watchdog timer to prevent it from timeout.

If due to hardware failure or program error. The program will not be able to reset the watchdog again. The timer will timeout and generates a timeout signal . This timeout signal will reset the system.

Watchdog operations include basic management operations, such as turning on and refreshing. The following topics describe how you can operate watchdog programmatically using the watchdog functions.

"WDT" usage scenarios and applications:

Typical applications of Watchdog Timer ("WDT"):
Preventing the microprocessor deadlock is a typical application of "WDT". Usually the software has a "**main loop**" program, which is used to call subroutines to achieve different tasks. The "WDT" will **refresh once every loop**. If the loop fails for any reason, the watchdog timer will timeout and reset the system. It should be noted that "WDT" cannot detect instantaneous system faults.

So, the processor will reset only when the "WDT" counter reaches a predetermined time interval. The user necessary to select a shortest timeout period so that the "WDT" can generate a reset before the system is out of control, so that the system can resume the normal operation.

Supported PACs

The table below lists the watchdog functions that are supported by each PAC.

Models / Functions	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_EnableWatchDog	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_DisableWatchDog	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RefreshWatchDog	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetWatchDogState	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetWatchDogTime	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SetWatchDogTime	Y	Y	Y	Y	Y	Y	Y	Y	Y

Watchdog Functions

The following functions are used to retrieve or set the Watchdog.

PACSDK Functions	PACNET Functions	Description
pac_EnableWatchDog	Sys.WDT.EnableWatchDog	starts a watchdog operation.
pac_DisableWatchDog	Sys.WDT.DisableWatchDog	stops a watchdog operation.
pac_RefreshWatchDog	Sys.WDT.RefreshWatchDog	refreshes the watchdog.
pac_GetWatchDogState	Sys.WDT.GetWatchDogState	retrieves the watchdog state.
pac_GetWatchDogTime	Sys.WDT.GetWatchDogTime	retrieves the watchdog time.
pac_SetWatchDogTime	Sys.WDT.SetWatchDogTime	starts a watchdog operation, or set the WDT parameters in the WDT enable status .

3.4.1. pac_EnableWatchDog

This function starts a watchdog operation.

Syntax

C++

```
BOOL pac_EnableWatchDog(
    int WDT,
    DWORD value
);
```

Parameters

wdt

[in] Specifies the name of watchdog:

PAC_WDT_HW 0

PAC_WDT_OS 1

value

[in] Specifies the watchdog time.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call pac_GetLastError.

A nonzero error code defined in PACERROR.h indicates failure. To get a generic description of the error, call pac_GetErrorMessage. The message resource is optional; therefore, if you call pac_GetErrorMessage it could fail.

Examples

[C]

```
DWORD second = 1000;
bool ret;
ret = pac_EnableWatchDog(PAC_WDT_OS, second);
```

[C#]

```
uint second = 1000;
bool ret_err;
ret_err = PACNET.Sys.WDT.EnableWatchDog(PAC_WDT_OS, second);
```

Remarks

The unit of the second parameter of the OS watchdog is second. In addition, the unit cannot be zero.

PAC_WDT_OS WDT:

Generation the restart signal by the CPU watchdog module, when "WDT" triggers.
The OS watchdog second parameter is unit between 0~1321 seconds.(About 22 minutes)

PAC_WDT_HW WDT:

Generation the restart signal by the independent watchdog module, when "WDT" triggers.

(for XPAC series only)

The hardware watchdog second parameter is a value which is between 1~63 unit.
A unit is about 0.5 seconds. 1 means the shortest timeout, otherwise 63 is longest and it takes about 30 seconds.

(for WinPAC series only)

The hardware watchdog second Parameter is a value which between 1~31 unit.
A unit is about 200 milliseconds. 1 means the shortest timeout, otherwise 31 is longest and it takes about 6.2 seconds.

3.4.2. pac_DisableWatchDog

This function stops watchdog.

Syntax

C++

```
void pac_DisableWatchDog(
    int WDT
);
```

Parameters

wdt

[in] Specifies the Watchdog type:

PAC_WDT_HW 0

PAC_WDT_OS 1

Return Value

This function has does not return a value.

Examples

[C]

```
pac_DisableWatchDog(PAC_WDT_OS);
```

[C#]

```
PACNET.Sys.WDT.DisableWatchDog(PAC_WDT_OS);
```

3.4.3. pac_RefreshWatchDog

This function refreshes the watchdog.

Note:

The function must be placed in the main loop code of the program.

To make sure the watchdog reset OS system during a crash.

Syntax

C++

```
void pac_RefreshWatchDog(
    int WDT
);
```

Parameters

wdt

[in] Specifies the Watchdog type:

PAC_WDT_HW 0

PAC_WDT_OS 1

Return Value

This function has does not return a value.

Examples

[C]

```
pac_RefreshWatchDog(PAC_WDT_OS);
```

[C#]

```
PACNET.Sys.WDT.RefreshWatchDog(PAC_WDT_OS);
```

3.4.4. pac_GetWatchDogState

This function retrieves the watchdog state.

Syntax

C++

```
BOOL pac_GetWatchDogState(
    int WDT
);
```

Parameters

wdt

[in] Specifies the Watchdog type:

PAC_WDT_HW 0

PAC_WDT_OS 1

Return Value

If the watchdog is turning on, the return value is TRUE.

If the watchdog is turning off, the return value is FALSE.

To get extended error information, call pac_GetLastError.

Examples

[C]

```
BOOL bState;
bState = pac_GetWatchDogState(PAC_WDT_OS);
```

[C#]

```
bool bState;
bState = PACNET.Sys.WDT.GetWatchDogState(PAC_WDT_OS);
```

3.4.5. pac_GetWatchDogTime

This function retrieves the watchdog time.

Syntax

C++

```
DWORD pac_GetWatchDogTime(
    int WDT
);
```

Parameters

wdt

[in] Specifies the Watchdog type:

PAC_WDT_HW 0

PAC_WDT_OS 1

Return Value

The return value is the watchdog time which has been assigned by pac_EnableWatchDog or pac_SetWatchDogTime.

The OS watchdog unit for return value is second.

The hardware watchdog return value is between 0~63 unit.

Examples

[C]

```
DWORD dwTime;
dwTime = pac_GetWatchDogTime(PAC_WDT_OS);
```

[C#]

```
uint uTime;
uTime = PACNET.Sys.WDT.GetWatchDogTime(PAC_WDT_OS);
```

3.4.6. pac_SetWatchDogTime

This function starts a watchdog operation, or set the WDT parameters in the WDT enable status.

Syntax

C++

```
bool pac_SetWatchDogTime(
    int wdt,
    DWORD value
);
```

Parameters

wdt

[in] Specifies the name of watchdog:

PAC_WDT_HW 0

PAC_WDT_OS 1

value

[in] Specifies the watchdog time.

Return Value

This function has does not return a value.

Examples

[C]

```
DWORD dwTime = 1000;
pac_SetWatchDogTime(PAC_WDT_OS , dwTime);
```

[C#]

```
UINT UTIME = 1000;
PACNET.Sys.WDT.SetWatchDogTime(PAC_WDT_OS , UTIME);
```

Remarks

The unit of the second parameter of the OS watchdog is second. In addition, the unit cannot be zero.

PAC_WDT_OS WDT:

Generation the restart signal by the CPU watchdog module, when "WDT" triggers.

The OS watchdog second parameter is unit between 0~1321 seconds.(About 22 minutes)

PAC_WDT_HW WDT:

Generation the restart signal by the independent watchdog module, when "WDT" triggers.

(for XPAC series only)

The hardware watchdog second parameter is a value which is between 1~63 unit.

A unit is about 0.5 seconds. 1 means the shortest timeout, otherwise 63 is longest and it takes about 30 seconds.

(for WinPAC series only)

The hardware watchdog second Parameter is a value which between 1~31 unit.

A unit is about 200 milliseconds. 1 means the shortest timeout, otherwise 31 is longest and it takes about 6.2 seconds.

3.5. Registry API

Registry operations include basic management operations, such as reading from and writing to the registry. The following topics describe how you can create, delete, or modify registry keys programmatically using the registry functions.

Supported PACs

The table below lists the registry functions that are supported by each PAC.

Models / Functions	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_RegCountKey	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegCountValue	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegCreateKey	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegDeleteKey	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegDeleteValue	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegGetDWORD	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegGetKeyByIndex	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegGetKeyInfo	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegGetString	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegGetValueByIndex	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegKeyExist	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegSave	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegSetString	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegSetDWORD	Y	Y	Y	Y	Y	Y	Y	Y	Y

Registry Functions

The following functions are used to retrieve or set the registry.

PACSDK Functions	PACNET Functions	Description
pac_RegCountKey	PAC_Reg.RegCountKey	retrieves the specified registry key which has how many sub keys.
pac_RegCountValue	PAC_Reg.RegCountValue	retrieves the specified registry key which has how many values.
pac_RegCreateKey	PAC_Reg.RegCreateKey	creates the specified registry key.
pac_RegDeleteKey	PAC_Reg.RegDeleteKey	deletes a subkey from the specified registry key.
pac_RegDeleteValue	PAC_Reg.RegDeleteValue	removes a value from the specified registry key.
pac_RegGetDWORD	PAC_Reg.RegGetDWORD	retrieves DWORD value of the specified registry key.
pac_RegGetKeyByIndex	PAC_Reg.RegGetKeyByIndex	retrieves the name of specified index of registry key.
pac_RegGetKeyInfo	PAC_Reg.RegGetKeyInfo	retrieves the type of specified index of registry key.
pac_RegGetString	PAC_Reg.RegGetString	retrieves string value of the specified registry key.
pac_RegGetValueByIndex	PAC_Reg.RegGetValueByIndex	retrieves the value of specified index of registry key.
pac_RegKeyExist	PAC_Reg.RegKeyExist	determinants the specified registry key exist or not.
pac_RegSave	PAC_Reg.RegSave	writes all the attributes into the registry.
pac_RegSetString	PAC_Reg.RegSetString	assigns the specified registry key data whose type is string.
pac_RegSetDWORD	PAC_Reg.RegSetDWORD	assigns the specified registry key data whose type is DWORD.

3.5.1. pac_RegCountKey

retrieves the specified registry key which has how many sub keys.

Syntax

C++

```
DWORD pac_RegCountKey(
    LPCTSTR KEYNAME
);
```

Parameters

KeyName

[in] Specifies the absolute path of registry key which you want to count.

Return Value

Return the number of subkeys contained by the specified key.

Examples

[C]

```
DWORD I;
I = pac_RegCountKey(TEXT("HKEY_USERS\\myKey \\"));
```

[C#]

```
Uint i;
I = PACNET.PAC_Reg.RegCountKey("HKEY_USERS \\myKey \\");
```

3.5.2. pac_RegCountValue

retrieves the specified registry key which has how many values.

Syntax

C++

```
DWORD pac_RegCountValue(
    LPCTSTR KEYNAME
);
```

Parameters

KeyName

[in] Specific the absolute path of registry key which you want to count.

Return Value

Return the number of values associated with the key.

Examples

[C]

```
DWORD I;
I = pac_RegCountValue(TEXT("HKEY_USERS\\myKey\\"));
```

[C#]

```
Uint i;
I = PACNET.PAC_Reg.RegCountValue("HKEY_USERS\\myKey\\");
```

3.5.3. pac_RegCreateKey

This function creates the specified registry key.

Syntax

C++

```
Bool pac_RegCreateKey(
    LPCTSTR KEYNAME
);
```

Parameters

KeyName

[in] Specific the path of registry key which you want to create.

Return Value

Return true if creates key success, otherwise false. To get an error code, call pac_GetLastError.

A nonzero error code defined in PACERROR.h indicates failure.

To get a generic description of the error, call pac_GetErrorMessage.

The message resource is optional; therefore, if you call pac_GetErrorMessage it could fail.

Examples

[C]

```
Bool RET;
RET = pac_RegCreateKey(TEXT("HKEY_USERS\\myKey\\"));
```

[C#]

```
Bool RET;
RET = PACNET.PAC_Reg.RegCreateKey("HKEY_USERS\\myKey\\");
```

3.5.4. pac_RegDeleteKey

This function deletes a subkey from the specified registry key.

Syntax

C++

```
Bool pac_RegDeleteKey(
    LPCTSTR KEYNAME
);
```

Parameters

KeyName

[in] Specifics the absolute path of registry key which you want to delete.

Return Value

Return true if success, otherwise false. To get an error code, call pac_GetLastError. A nonzero error code defined in PACERROR.h indicates failure. To get a generic description of the error, call pac_GetErrorMessage. The message resource is optional; therefore, if you call pac_GetErrorMessage it could fail.

Examples

[C]

```
Bool RET;
RET = pac_RegDeleteKey(TEXT("HKEY_USERS\\myKey \\"));
```

[C#]

```
Bool RET;
RET = PACNET.PAC_Reg.RegDeleteKey("HKEY_USERS\\myKey \\");
```

Remarks

If the function succeeds, the function will delete the specified key including all of its subkeys and values. An application cannot call RegDeleteKey for a key that an application currently has open.

3.5.5. pac_RegDeleteValue

This function removes a value from the specified registry key.

Syntax

C++

```
Bool pac_RegDeleteValue(
    LPCTSTR KEYNAME
);
```

Parameters

KeyName

[in] Specifics the absolute path of registry key which you want to delete.

Return Value

Return true if success, otherwise false. To get an error code, call pac_GetLastError. A nonzero error code defined in PACERROR.h indicates failure. To get a generic description of the error, call pac_GetErrorMessage. The message resource is optional; therefore, if you call pac_GetErrorMessage it could fail.

Examples

[C]

```
Bool RET;
RET = pac_RegDeleteValue(TEXT("HKEY_USERS\\myKey \\value"));
```

[C#]

```
Bool RET;
RET = PACNET.PAC_Reg.RegDeleteValue(TEXT("HKEY_USERS\\myKey \\value"));
```

Remarks

The function could be used only in leaf key. And the function, pac_RegDeleteKey, can be used in any registry key except leaf key. If you would delete a leaf key, you should call pac_RegDeleteValue.

3.5.6. pac_RegGetDWORD

This function retrieves DWORD value of the specified registry key.

Syntax

C++

```
DWORD pac_RegGetDWORD(
    LPCTSTR KEYNAME
);
```

Parameters

KeyName

[in] Specific the absolute path of registry key.

Return Value

Return the DWORD value of the specific key.

Examples

[C]

```
DWORD dwValue;
dwValue = pac_RegGetDWORD(TEXT("HKEY_USERS\\myKey \\value"));
```

[C#]

```
UINT uValue;
uValue = PACNET.PAC_Reg.RegGetDWORD("HKEY_USERS\\myKey \\value");
```

3.5.7. pac_RegGetKeyByIndex

This function retrieves the name of specified index of registry key.

Syntax

C++

```
Bool pac_RegGetKeyByIndex(
    LPCTSTR keyname,
    DWORD dwIndex,
    LPTSTR lpName
);
```

Parameters

KeyName

[in] Specific the absolute path of registry key.

dwIndex

[in] Specific the index of registry key.

lpName

[out] Assign a buffer to retrieves the specific the key name.

Return Value

Return true if success, otherwise false. To get an error code, call pac_GetLastError. A nonzero error code defined in PACERROR.h indicates failure. To get a generic description of the error, call pac_GetErrorMessage. The message resource is optional; therefore, if you call pac_GetErrorMessage it could fail.

Examples

[C]

```
Bool RET;
DWORD index= 0;
TCHAR strName[10] ;
RET = pac_RegGetKeyByIndex(TEXT("HKEY_USERS\\myKey \\"), Index, strName);
```

[C#]

```
Bool RET;
UINT index= 0;
String strName =new String('\ 0' · 10);
RET = PACNET.PAC_Reg.RegGetKeyByIndex("HKEY_USERS\\myKey \\", index, strName);
```

3.5.8. pac_RegGetKeyInfo

This function retrieves the type of specified index of registry key.

Syntax

C++

```
DWORD pac_RegGetKeyInfo(LPCTSTR KeyName);
```

Parameters

KeyName

[in] Specific the path of registry key.

Return Value

We define four types about the return value:

PKT_NONE 0

PKT_KEY 1

PKT_STRING 2

PKT_DWORD 3

Examples

[C]

```
DWORD dwType;
dwType = pac_RegGetKeyInfo(TEXT("HKEY_USERS\\myKey \\value"));
```

[C#]

```
UINT UTYPE;
UTYPE = PACNET.PAC_Reg.RegGetKeyInfo("HKEY_USERS \\myKey \\value");
```

3.5.9. pac_RegGetString

This function retrieves string value of the specified registry key.

Syntax

C++

```
Bool pac_RegGetString(
  LPCTSTR keyname,
  LPTSTR lpData,
  DWORD dwLength
);
```

Parameters

KeyName

[in] Specific the absolute path of registry key.

lpData

[out] Pointer to a string buffer that receives the value's data.

dwLength

[in] Specific the size of data.

Return Value

Return true if success, otherwise false. To get an error code, call pac_GetLastError. A nonzero error code defined in PACERROR.h indicates failure. To get a generic description of the error, call pac_GetErrorMessage. The message resource is optional; therefore, if you call pac_GetErrorMessage it could fail.

Examples

[C]

```
Bool RET;
TCHAR strName[10] ;
RET = pac_RegGetString(TEXT( "HKEY_USERS\\myKey \\value" ) , strName ,
sizeof(strName));
```

[C#]

```
Bool RET;
String strName =new String('\ 0' , 10);
RET = PACNET.PAC_Reg.RegGetString( "HKEY_USERS \\myKey \\value" , strName ,
(UINT)strName.Length);
```

3.5.10. pac_RegGetValueByIndex

This function retrieves the value of specified index of registry key.

Syntax

C++

```
Bool pac_RegGetValueByIndex(
    LPCTSTR keyname,
    DWORD dwIndex,
    LPTSTR lpName
);
```

Parameters

KeyName

[in] Specific the absolute path of registry key.

dwIndex

[in] Specific the index of value.

lpName

[out] Pointer to a buffer that receives the value's data.

Return Value

Return true if success, otherwise false. To get an error code, call pac_GetLastError. A nonzero error code defined in PACERROR.h indicates failure. To get a generic description of the error, call pac_GetErrorMessage. The message resource is optional; therefore, if you call pac_GetErrorMessage it could fail.

Examples

[C]

```
Bool RET;
DWORD index= 0;
TCHAR strName[10] ;
RET = pac_RegGetValueByIndex(TEXT(“HKEY_USERS\\myKey \\”)，index，strName);
```

[C#]

```
Bool RET;
UINT index = 0;
String strName =new String('\ 0'，10);
RET = PACNET.PAC_Reg.RegGetValueByIndex(“HKEY_USERS\\myKey \\”，index，strName);
```

3.5.11. pac_RegKeyExist

This function determinants the specified registry key exist or not.

Syntax

C++

```
Bool pac_RegKeyExist(
    LPCTSTR KEYNAME
);
```

Parameters

KeyName

[in] Specific the absolute path of registry key which you want to check whether it exists or not.

Return Value

Return true if success, otherwise false. To get an error code, call pac_GetLastError. A nonzero error code defined in PACERROR.h indicates failure. To get a generic description of the error, call pac_GetErrorMessage. The message resource is optional; therefore, if you call pac_GetErrorMessage it could fail.

Examples

[C]

```
Bool bExist;
bExist = pac_RegKeyExist(TEXT("HKEY_USERS \\myKey \\"));
```

[C#]

```
Bool bExist;
bExist = PACNET.PAC_Reg.RegKeyExist("HKEY_USERS \\myKey \\");
```

3.5.12. pac_RegSave

This function writes all the attributes into the registry..

Syntax

C++

```
Bool pac_RegSave(
    LPCTSTR KEYNAME
);
```

Parameters

KEYNAME

[in] :

HKEY_CLASSES_ROOT

HKEY_CURRENT_USER

HKEY_LOCAL_MACHINE

HKEY_USERS

Return Value

Return true if success, otherwise false. To get an error code, call pac_GetLastError. A nonzero error code defined in PACERROR.h indicates failure. To get a generic description of the error, call pac_GetErrorMessage. The message resource is optional; therefore, if you call pac_GetErrorMessage it could fail.

Examples

[C]

```
Bool RET;
RET = pac_RegSave(TEXT("HKEY_USERS"));
```

[C#]

```
Bool RET;
RET = PACNET.PAC_Reg.RegSave("HKEY_USERS");
```

3.5.13. pac_RegSetString

This function assigns the specified registry key data whose type is string.

Syntax

C++

```
Bool pac_RegSetString(
    LPCTSTR keyname,
    LPCTSTR assignStr,
    DWORD dwLength
);
```

Parameters

KeyName

[in] Specific the absolute path of registry key which you want to assign string data.

assignStr

[in] Specific the string data.

dwLength

[in] Specific the size of string data.

Return Values

Return true if success, otherwise false. To get an error code, call pac_GetLastError. A nonzero error code defined in PACERROR.h indicates failure. To get a generic description of the error, call pac_GetErrorMessage. The message resource is optional; therefore, if you call pac_GetErrorMessage it could fail.

Examples

[C]

```
Bool RET;
RET = pac_RegSetString(TEXT("HKEY_USERS\\myKey\\value"), TEXT("HELLO.EXE"), 2 * wcslen(TEXT("HELLO.EXE"))); //size of(TCHAR)= 2
```

[C#]

```
Bool RET;
RET = PACNET.PAC_Reg.RegSetString("HKEY_USERS \\myKey \\value", "HELLO.EXE", 2 *(UINT) "HELLO.EXE".lenght);
```

3.5.14. pac_RegSetDWORD

This function assigns the specified registry key data whose type is DWORD.

Syntax

C++

```
Bool pac_RegSetDWORD(
    LPCTSTR keynamee,
    DWORD assignVal
);
```

Parameters

KeyName

[in] Specific the absolute path of registry key which you want to assign DWORD data.

assignStr

[in] Specific the DWORD data.

Return Values

Return true if success, otherwise false. To get an error code, call pac_GetLastError. A nonzero error code defined in PACERROR.h indicates failure. To get a generic description of the error, call pac_GetErrorMessage. The message resource is optional; therefore, if you call pac_GetErrorMessage it could fail.

Examples

[C]

```
bool ret;
ret = pac_RegSetDWORD(TEXT("HKEY_USERS\\myKey\\value"), 40);
```

[C#]

```
bool ret;
ret = PACNET.PAC_Reg.RegSetDWORD("HKEY_USERS\\myKey\\value", 40);
```

3.6. UART API

Uart operations include basic management operations, such as opening, sending, receiving, and closing. The following topics describe how you can operate uart programmatically using the uart functions.

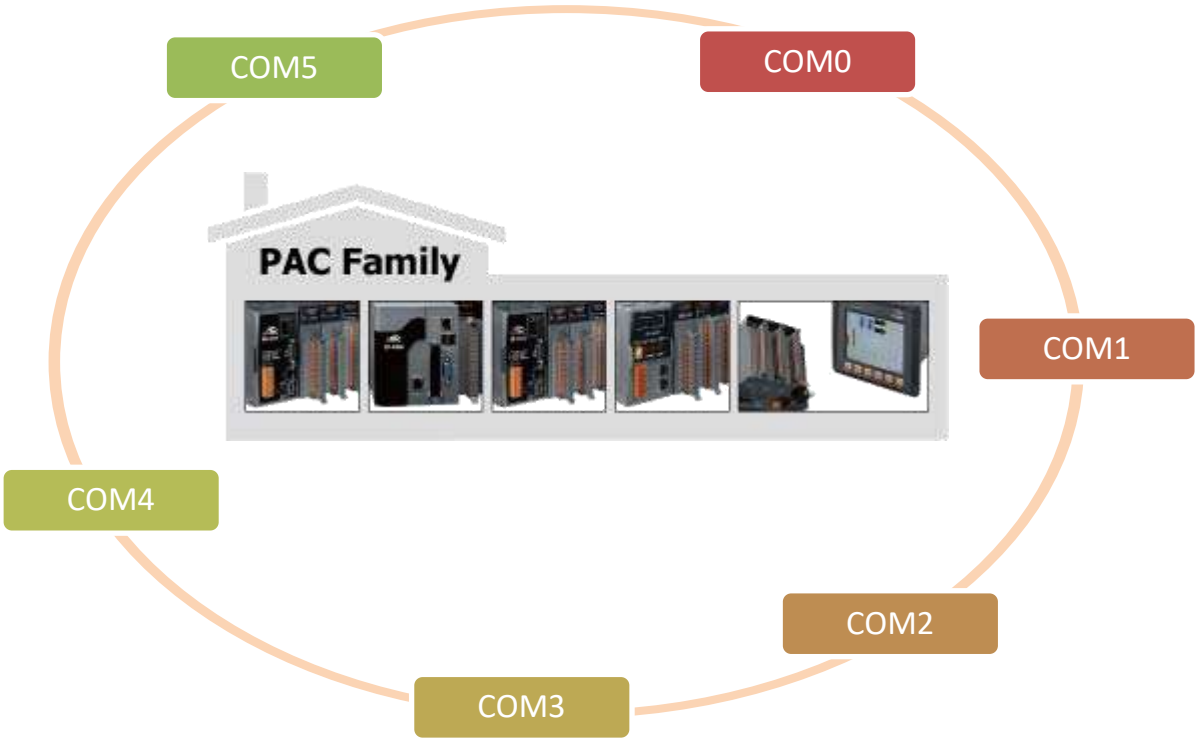
Remarks

We provide several COM port functions (uart_Send/uart_Recv…) to communicate with ICPDAS devices (I-87K series, I-811xW/I-814xW series, I-7000 series).

All the functions are based on standard COM port API functions in C++ (CreateFile/CloseHandle/WriteFile/ReadFile /GetCommModemStatus…..).

The pre-defined COM ports in each PACs

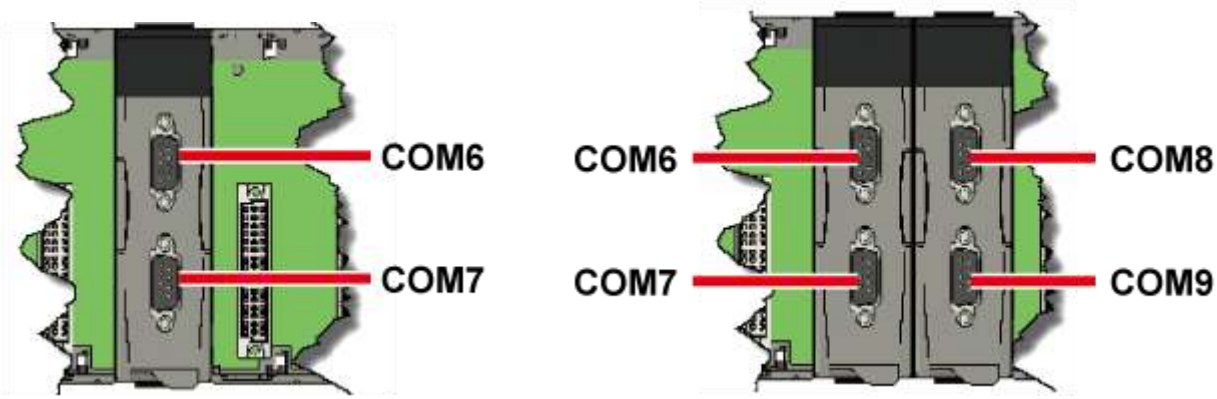
The pre-definition of the COM port in each PACs are listed in Appendix F



The expanded COM ports in each PACs

Some PACs has some I/O slot that can be used to expand the communication port.

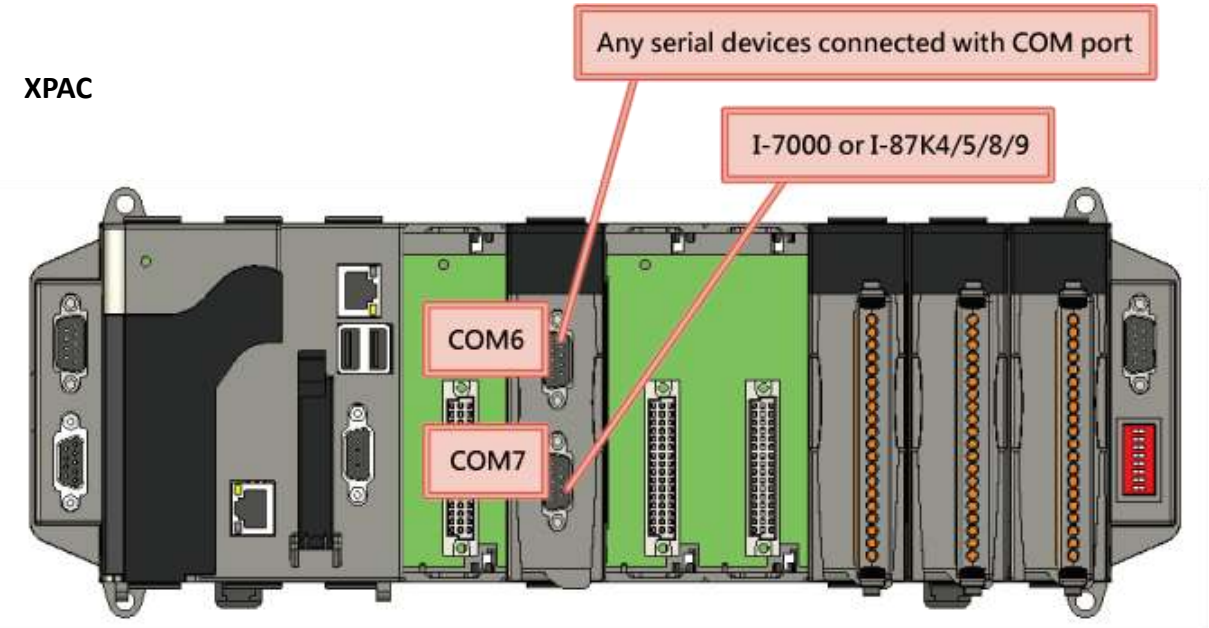
WP-8000 / ViewPAC (WinCE)



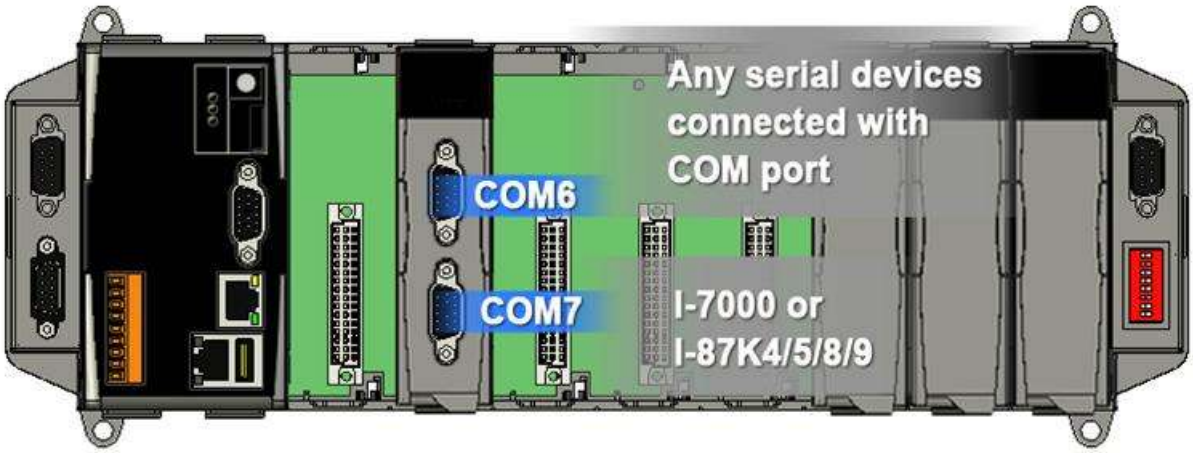
When a I-87K is plugged in slot, please call the function, pac_ChangeSlot, to change to the specific slot before doing other operations. Please refer to demo “87k_Basic”.
About I-87K commands (DCON protocol), please refer
http://ftp.icpdas.com/pub/cd/8000cd/napdos/dcon/io_module/87k_high_profile_modules.htm

Although user can use UART API to set and read values for I-87K series modules, we provide a more convenient API to do it. Please refer to Section 7 PAC_IO API
Use these functions of this section to communicate with external devices by I-811xW/I-814xW serises modules.

XPAC



WP-8000 / ViewPAC (WinCE)



WP-5000

The WP-5000 series has no slots for plugging the I-8K and I-87K series, but the UART API on this section can also be used for COM1/COM2/COM3/COM4 of WP-5000 series.

In addition of COM1/COM2/COM3/COM4, all of the functions can be used to communicate with external device by XW5xxx/XV5xxx(COM0) expansion board.

To see more information, please reference user manual - Chapter 5 API and Demo Reference.

Supported PACs

The table below lists the UART functions that are supported by each PAC.

Models / Functions	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
uart_Close	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_SendExt	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_Send	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_RecvExt	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_Recv	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_SendCmdExt	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_SetTimeOut	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_EnableCheckSum	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_SetTerminator	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_BinSend	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_BinRecv	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_BinSendCmd	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_GetLineStatus	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_GetDataSize	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_SetLineStatus	Y	Y	Y	Y	Y	Y	Y	Y	Y

UART Functions

The following functions are used to retrieve or set the UART.

PACSDK Functions	PACNET Functions	Description
uart_Open	UART.Open	opens the COM port and specifies the baud rate, parity bits, data bits, and stop bits.
uart_Close	UART.Close	closes the COM port which has been opened.
uart_SendExt	UART.SendExt	sends data as a string through the COM port which has been opened.
uart_Send	UART.Send	sends data through the COM port which have been opened.
uart_RecvExt	UART.RecvExt	receives a string+0x0D. A [0x0D] character is assigned to terminate the string.
uart_Recv	UART.Recv	retrieves data through the COM port which have been opened.
uart_SendCmdExt	UART.SendCmdExt	sends commands through the COM port which has been opened.
uart_SetTimeOut	UART.SetTimeOut	sets the time out timer.
uart_EnableCheckSum	UART.EnableCheckSum	turns on the check sum or not.
uart_SetTerminator	UART.SetTerminator	sets the terminate characters.
uart_BinSend	UART.BinSend	sends out command string with or without null character under the consideration of the command length.
uart_BinRecv	UART.BinRecv	receives the response string data with or without null character under the consideration of receiving length.
uart_BinSendCmd	UART.BinSendCmd	sends binary command and receive binary data with the fixed length.
uart_GetLineStatus	UART.GetLineStatus	retrieves the specifiesa pin status of COM port.
uart_GetDataSize	UART.GetDataSize	retrieves the number of bytes received by the serial provider but not yet read by a uart_Recv operation, or of user data remaining to transmitted for write operations.
uart_SetLineStatus	UART.SetLineStatus	sets the status of modem line.

3.6.1. uart_Open

This function opens the COM port and specifies the baud rate, parity bits, data bits, and stop bits.

Syntax

C++

```
HANDLE uart_Open(
  LPCSTR ConnectionString
);
```

Parameters

connectionString

[in] Specifies the COM port, baud rate, parity bits, data bits, and stop bits.

The default setting is COM1, 115200, N, 8, 1.

The format of ConnectionString is as follows:

"com_port, baud_rate, parity_bits, data_bits, stop_bits"

Warning: there is no blank space between each parameter.

Com_port:

XPAC: COM1, COM2……

WinPAC: COM0, COM1……

baud_rate:

1200/2400/4800/9600/19200/38400/57600/115200

parity_bits:

'N' = NOPARITY 'O' = ODDPARITY

'E' = EVENPARITY 'M' = MARKPARITY

'S' = SPACEPARITY

Data_bits:

5/6/7/8

Stop_bits:

"1" = ONESTOPBIT "2" = TWOSTOPBITS

"1.5" = ONE5STOPBITS

Return Values

A handle to the open COM port.

Nonzero indicates success.

If the function fails, the return value is INVALID_HANDLE_VALUE. (INVALID_HANDLE_VALUE should be 0xffffffff in C/C++/MFC. INVALID_HANDLE_VALUE should be -1 in .NET.)

To get extended error information, call pac_GetLastError. To get a generic description of the error, call pac_GetErrorMessage.

The message resource is optional; therefore, if you call pac_GetErrorMessage it could fail.

Examples

[C]

```
HANDLE hOpen;
hOpen = uart_Open("COM1,9600,N,8,1");
```

[C#]

```
IntPtr hOpen;
hOpen = PACNET.UART.Open("COM1,9600,N,8,1");
```

Remarks

The uart_Open function does not open the specified COM port if the COM port has been opened.

[Use I-811xW/I-814xW series modules]

The COM port name is COM6/COM7/MSA1/MSB1…., MSAx/MSBx is an earlier old usage. The new usage is COMx.

For example:

uart_Open("COM6:,9600,N,8,1");

uart_Open("MSA1:,9600,N,8,1");

About how to set I-811xW/I-814xW series modules, Please refer to the manual below:

w1-007-1_how_to_set_up_a_communication_module(I-8112_I-8114_I-8142_I-8144)_use_COM_english.pdf

w1-007-2_how_to_set_up_a_communication_module(I-8112_I-8114_I-8142_I-8144)_use_MSA(B..)_english.pdf

[Use I-87K series modules]

Only use COM0 to communicate with I-87K series modules. Please refer to Sec.8 UART API.

3.6.2. uart_Close

This function closes the COM port which has been opened.

Syntax

C++

```
BOOL uart_Close(
    HANDLE hPort
);
```

Parameters

hPort

[in] Handle to the open COM port to close.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
BOOL ret;
HANDLE hOpen;
hOpen = uart_Open("COM1,9600,N,8,1");
ret = uart_Close(hOpen);
```

[C#]

```
bool ret;
IntPtr hOpen;
hOpen = PACNET.UART.Open("COM1,9600,N,8,1");
ret = PACNET.UART.Close(hOpen);
```

Remarks

The function for a specified COM port should not be used after it has been closed.

3.6.3. uart_SendExt

This function sends data as a string through the COM port which has been opened.

When the checksum is enabled by using uart_EnableCheckSum function, the two bytes of the checksum is automatically added to the string, and the character [0x0D] is added to the end of the string to terminate the string (buf).

This function replaces the uart_Send function.

Syntax

C++

```
BOOL uart_SendExt(
    HANDLE hPort,
    LPSTR BUF,
    DWORD out_Len
);
```

Parameters

hPort

[in] Handle to the open COM port.

buf

[in] A pointer to a buffer that receives data.

out_Len

[in] A pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the buf.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FLASE.

Examples

[C]

```
BOOL ret;
HANDLE hOpen;
char buf[Length] ;
sprintf(buf,"abcd");
hOpen = uart_Open("COM1,9600,N,8,1");
ret = uart_SendExt(hOpen, buf, Length);
uart_Close(hPort);
```

[C#]

```
bool ret;
IntPtr hOpen;
string buf;
buf = "abcd"
byte[] result = new byte[64] ;
result= PACNET.MISC.AnsiString(buf);
hOpen = PACNET.UART.Open("COM1,9600,N,8,1");
ret = PACNET.UART.SendExt(hOpen, result, 64);
PACNET.UART.Close(hPort);
```

Remarks

A string for “buf” cannot include space character within the string. Otherwise, the string will be stoped by space character. For example: “$01M 02 03” of the user defined string. Howerver, the actual string sent out is “$01M”+0x0D.

The terminate characters is 0x0D. (Refer to uart_SetTerminator function to change.)

This function will call PurgeComm() to clear serial COM port output buffer.

This function sends data with a terminate character 0x0D. For example:

Check sum is disabled. The “buf” are five bytes (ABCD+0x0). This function will send five bytes (ABCD+0x0D).

3.6.4. uart_Send

This function sends data through the COM port which have been opened.

This function will send a string. If the checksum is enabled by the uart_EnableCheckSum function, this function automatically adds the two checksum bytes to the string. And then the end of sending string is further added [0x0D] to mean the termination of the string (buf).

Syntax

C++

```
BOOL uart_Send(
    HANDLE hPort,
    LPSTR BUF
);
```

Parameters

hPort

[in] Handle to the open COM port.

buf

[in] A pointer to a buffer that receives data.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FLASE.

Examples

[C]

```
BOOL ret;
HANDLE hPort;
char buf[4] ;
sprintf(buf,"abcd");
hPort = uart_Open("COM2:,9600,N,8,1");
ret = uart_Send(hPort, buf);
uart_Close(hPort);
```

[C#]

```
bool ret;
IntPtr hPort;
string buf;
buf = "abcd";
hPort = PACNET.UART.Open("COM2:,9600,N,8,1");
ret = PACNET.UART.Send(hPort, PACNET.MISC.AnsiString(buf));
PACNET.UART.Close(hPort);
```

Remarks

A string for “buf” cannot include space character within the string. Otherwise, the string will be stoped by space character. For example: “$01M 02 03” of the user defined string. Howerver, the actual string sent out is “$01M”+0x0D.

The terminate characters is 0x0D. (Refer to uart_SetTerminator function to change.)

This function will call PurgeComm() to clear serial COM port output buffer.

This function sends data with a terminate character 0x0D.

For example:

Check sum is disabled. The “buf” are five bytes (ABCD+0x0). This function will send five bytes (ABCD+0x0D).

3.6.5. uart_RecvExt

This function receives a string+0x0D. A [0x0D] character is assigned to terminate the string.

This function is not called when the checksum is enabled by using uart_EnableCheckSum function which includes the terminate character [0x0D].

This function replaces uart_Recv. The uart_Recv can cause the buffer overflow in some situation.

Syntax

C++

```
BOOL uart_RecvExt(
    HANDLE hPort,
    LPSTR BUF,
    DWORD out_Len
);
```

Parameters

hPort

[in] Handle to the open COM port.

buf

[in] A pointer to a buffer that receives data.

out_Len

[in] A pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the buf.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

If the function doesn't receive a character 0x0D, the other data still store to "buf" but the return value is FALSE. Calling pac_GetLastError function will get an error code (PAC_ERR_UART_READ_TIMEOUT)

If this function to receive the actual data size is larger than the buffer length of buf, it will return FALSE. Calling pac_GetLastError function will get an error code

(PAC_ERR_UART_INTERNAL_BUFFER_OVERFLOW)

Examples

[C]

```
BOOL ret;
HANDLE hOpen;
char buf[Length] ;
hOpen = uart_Open("COM1,9600,N,8,1");
ret = uart_RecvExt(hOpen, buf, Length);
uart_Close(hPort);
```

[C#]

```
bool ret;
IntPtr hOpen;
byte[] result = new byte[64] ;
hOpen = PACNET.UART.Open("COM1,9600,N,8,1");
ret = PACNET.UART.RecvExt(hOpen, result, 64);
PACNET.UART.Close(hPort):
```

Remarks

The terminate characters is 0x0D. (Refer to uart_SetTerminator function to change.)

For example:

a. Check sum is disabled. This function receives five bytes (ABCD+0x0D). The “buf” will be five bytes (ABCD+0x0).

b. Check sum is disabled. This function receives four bytes (ABCD). The “buf” will be four bytes (ABCD). But the reurn value is 0.

3.6.6. uart_Recv

This function retrieves data through the COM port which have been opened.

This function will receive a string+0x0D. Wait a character [0x0D] to mean the termination of a string. And then if the checksum is enabled by the uart_EnableCheckSum function, this function automatically check the two checksum bytes to the string. This function will provides a string without the last byte[0x0D].

Syntax

C++

```
BOOL uart_RecvExt(
    HANDLE hPort,
    LPSTR BUF,
);
```

Parameters

hPort

[in] Handle to the open COM port.

buf

[in] A pointer to a buffer that receives data.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

If this function doesn't receive a character0x0D, the other data still store to "buf" but the return value is 0. Calling pac_GetLastError function will get an error code (pac_ERR_uart_READ_TIMEOUT).

Examples

[C]

```
BOOL ret;
HANDLE hPort;
char buf[10] ;
hPort = uart_Open("COM2,9600,N,8,1");
ret = uart_Recv(hPort, buf);
uart_Close(hPort);
```

[C#]

```
bool ret;
IntPtr hPort;
string buf;
hPort = PACNET.UART.Open("COM2:,9600,N,8,1");
ret = PACNET.UART.Recv(hPort, PACNET.MISC.AnsiString(buf));
PACNET.UART.Close(hPort);
```

Remarks

The terminate characters is 0x0D. (Refer to uart_SetTerminator function to change.)

For example:

a. Check sum is disabled. This function receives five bytes (ABCD+0x0D). The “buf” will be five bytes (ABCD+0x0).

b. Check sum is disabled. This function receives four bytes (ABCD). The “buf” will be four bytes (ABCD). But the reurn value is 0.

3.6.7. uart_SendCmdExt

This function sends commands through the COM port which has been opened.

This function is a combination of uart_SendExt and uart_RecvExt.

The operation for sending a command is the same as uart_SendExt.

The operation for receiving a response is the same as uart_RecvExt.

This function replaces uart_SendCmd. The uart_SendCmd can cause the buffer overflow in some situation.

Syntax

C++

```
BOOL uart_SendCmdExt(
    HANDLE hPort,
    LPCSTR CMD,
    DWORD out_Len,
    LPSTR szResult,
    DWORD in_Len
);
```

Parameters

hPort

[in] Handle to the open COM.

cmd

[in] A pointer to a command.

out_Len

[in] A pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the buf.

szResult

[out] A pointer to a buffer that receives data.

in_Len

[in] A pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the szResult.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FLASE.

Examples

[C]

```
HANDLE hOpen;
char buf[Length] ;
hOpen = uart_Open("COM1,9600,N,8,1");
ret = uart_SendCmdExt(hOpen,"$00M", 4, buf, Length); // $00M: ask the device name
uart_Close(hPort);
```

[C#]

```
bool ret;
IntPtr hOpen;
string buf=”$01M”;
byte[] cmd = new byte[64] ;
byte[] result = new byte[64] ;
hOpen = PACNET.UART.Open("COM1,9600,N,8,1");
cmd= PACNET.MISC.AnsiString(buf);
ret = PACNET.UART.SendCmdExt(hOpen,cmd, 64, result, 64);
```

Remarks

This function calls PurgeComm() to clear serial COM port input and output buffer.

Refer to Remarks of uart_SendExt/uart_RecvExt for more details.

3.6.8. uart_SetTimeOut

This function sets the timeout timer.

Syntax

C++

```
void uart_SetTimeOut(
    HANDLE hPort,
    DWORD msec,
    int ctoType
);
```

Parameters

hPort

[in] Handle to the open COM port.

msec

[in] Millisecond to the timer.

ctoType

[in] Specifies the time out timer type as following:

CTO_TIMEOUT_ALL 0

CTO_READ_RETRY_TIMEOUT 1

CTO_READ_TOTAL_TIMEOUT 2

CTO_WRITE_TOTAL_TIMEOUT 3

Return Value

This function has does not return a value.

Examples

[C]

```
HANDLE hOpen;
DWORD mes;
hOpen = uart_Open("COM1,9600,N,8,1");
mes = 300;
uart_SetTimeOut(hOpen, mes, CTO_TIMEOUT_ALL);
uart_Close(hOpen);
```

[C#]

```
IntPtr hOpen;
uint msc;
hOpen = PACNET.UART.Open("COM1,9600,N,8,1");
mes = 300;
PACNET.UART.SetTimeOut(hOpen, msc, 0);
PACNET.UART.Close(hOpen);
```

Remarks

CTO_READ_TOTAL_TIMEOUT:

A constant used to calculate the total time-out period for read operations, in milliseconds.

A value of zero for the CTO_READ_TOTAL_TIMEOUT indicates that total time-outs are not used for read operations.

CTO_WRITE_TOTAL_TIMEOUT:

A constant used to calculate the total time-out period for write operations, in milliseconds.

A value of zero for the CTO_WRITE_TOTAL_TIMEOUT indicates that total time-outs are not used for write operations.

CTO_READ_RETRY_TIMEOUT:

A constant used to calculate the time-out period for read operations, in system tick count.

CTO_TIMEOUT_ALL:

A constant used to calculate the total time-out period for write and read operations, in milliseconds.

A value of zero for the CTO_TIMEOUT_ALL indicates that total time-outs are not used for write and read operations.

3.6.9. uart_EnableCheckSum

This function turns on the check sum or not.

Add two checksum bytes to the end of data which is used to produce checksum.

Syntax

C++

```
void uart_EnableCheckSum(
    HANDLE hPort,
    BOOL bEnable
);
```

Parameters

hPort

[in] Handle to the open COM port.

bEnable

[in] Decide the check sum turning on or not.

Default is disabling.

Return Value

This function has does not return a value.

Examples

[C]

```
HANDLE hUart;
char result[32] ;
hUart = uart_Open("");
uart_EnableCheckSum(hUart , true);
pac_ChangeSlot(1);
uart_SendCmd(hUart, "$00M", result);
uart_Close(hOpen);
```

[C#]

```
byte[] result = new byte[32] ;
IntPtr hPort = PACNET.UART.Open("");
PACNET.UART.EnableCheckSum(hPort, true);
PACNET.Sys.ChangeSlot(1);
PACNET.UART.SendCmd(hPort, XPac.AnsiString("$00M"), result);
string str = PACNET.MISC.WideString(result);
PACNET.UART.Close(hOpen);
```

3.6.10. uart_SetTerminator

This function sets the terminate characters.

Syntax

C++

```
void uart_SetTerminator(
    HANDLE hPort,
    LPCSTR szTerm
);
```

Parameters

hPort

[in] Handle to the open COM port.

szTerm

[in] Pointer the terminate characters.

Default is CR(0x0D).

Return Value

This function has does not return a value.

Examples

[C]

```
HANDLE hPort;
char result[32] ;
hPort = uart_Open(""); //Open COM0, data format: 115200,N,8,1
uart_SetTerminator(hPort, "\r");
pac_ChangeSlot(0); //A I-87K module is in slot0
uart_SendCmd(hPort, "$00M", result); // $00M: ask the device name,DCON
uart_Close(hPort);
```

[C#]

```
byte[] result = new byte[32] ;
IntPtr hPort = PACNET.UART.Open(""); // Open COM0, data format: 115200,N,8,1
PACNET.UART.SetTerminator(hPort, PACNET.MISC.AnsiString("\r"));
PACNET. Sys.ChangeSlot(0); //A I-87K module is in slot0.
PACNET.UART.SendCmd(hPort, PACNET.MISC.AnsiString("$00M"), result); // $00M: ask the device name,DCON
string str = PACNET.MISC.WideString(result);
PACNET.UART.Close(hPort);
```

Remarks

This function relates to uart_Send/uart_Recv/uart_SendCmd.

3.6.11. uart_BinSend

Send out the command string by fix length, which is controlled by the Parameter “in_Len”. The difference between this function and uart_Send is that uart_BinSend terminates the sending process by the string length “in_Len” instead of the character "CR"(Carry return). Therefore, this function sends out command string with or without null character under the consideration of the command length. Besides, because of this function without any error checking mechanism (Checksum, CRC, LRC... etc.), users have to add the error checking information to the raw data by themselves if communication checking system is required.

Syntax

C++

```
Bool uart_BinSend(
  HANDLE hPort,
  LPCSTR BUF,
  DWORD in_Len
);
```

Parameters

hPort

[in] Handle to the open COM port.

buf

[in] A pointer to a buffer that send the data.

in_Len

[in] The length of result string.

Return Value

If the function succeeds, the return value is 1.

If the function fails, the return value is 0.

Examples

[C]

```
bool ret;
HANDLE hPort;
char buf[2] ;
buf[0] = 0x41;
buf[1] = 0x42;
hPort = uart_Open("COM4,9600,N,8,1");
ret = uart_BinSend(hPort, buf, 2);
uart_Close(hPort);
```

[C#]

```
bool ret;
IntPtr hPort;
string buf = "AB";
hPort = PACNET.UART.Open("COM4,9600,N,8,1");
ret = PACNET.UART.BinSend(hPort, XPac.AnsiString(buf), 2);
PACNET.UART.Close(hPort);
```

Remarks

Note that this function is usually applied to communicate with the other device, but not for ICPDAS DCON (I-7000/8000/87K) series modules.

This function will call PurgeComm() to clear serial COM port output buffer.

3.6.12. uart_BinRecv

This function is applied to receive the fix length response. The length of the receiving response is controlled by the Parameter “in_Len”. The difference between this function and uart_Recv is that uart_BinRecv terminates the receiving process by the string length “in_Len” instead of the character "CR"(Carry return). Therefore, this function receives the response string data with or without null character under the consideration of receiving length. Besides, because of this function without any error checking mechanism (checksum, CRC, LRC... etc.), users have to remove the error checking information from the raw data by themselves if communication checking system is used.

Syntax

C++

```
Bool uart_BinRecv(
    HANDLE hPort,
    LPSTR BUF,
    DWORD in_Len
);
```

Parameters

hPort

[in] Handle to the open COM port.

buf

[out] A pointer to a buffer that receives the data.

in_Len

[in] The length of result string.

Return Value

If the function succeeds, the return value is 1.

If the function fails, the return value is 0.

Examples

[C]

```
bool ret;
HANDLE hPort;
char buf[2] ;
hPort = uart_Open("COM4,9600,N,8,1");
ret = uart_BinSend(hPort, "AB", 2);
ret = uart_BinRecv(hPort, buf, 2);
uart_Close(hPort);
```

[C#]

```
bool ret;
IntPtr hPort;
byte[] buf = new byte[100] ;
hPort = PACNET.UART.Open("COM4,9600,N,8,1");
ret = PACNET.UART.BinSend(hPort, XPac(WinPac).AnsiString("AB"), 2);
ret = PACNET.UART.Recv(hPort, buf);
PACNET.UART.Close(hPort);
```

Remarks

Note that this function is usually applied to communicate with the other device, but not for ICPDAS DCON (I-7000/8000/87K) series modules.

3.6.13. uart_BinSendCmd

This function sends binary command and receive binary data with the fixed length.

This function is a combination of uart_BinSend and uart_BinRecv.

The operation for sending a command is the same as uart_BinSend.

The operation for receiving a response is the same as uart_BinRecv.

Syntax

C++

```
Bool uart_BinSendCmd(
    HANDLE hPort,
    LPCSTR ByteCmd,
    DWORD in_Len,
    LPSTR ByteResult,
    DWORD out_Len
);
```

Parameters

hPort

[in] Handle to the open COM.

ByteCmd

[in] A pointer to a command.

in_Len

[in] The length of command string.

ByteResult

[out] A pointer to a buffer that receives the data.

out_Len

[in] The length of result string.

Return Value

If the function succeeds, the return value is 1.

If the function fails, the return value is 0.

Examples

[C]

```
bool ret;
HANDLE hPort;
char buf[2] ;
char cmd[2] ;
hPort = uart_Open("COM4,9600,N,8,1");
cmd[0] = 0x41;
cmd[1] = 0x42;
ret = uart_BinSendCmd( hPort, cmd, 2, buf, 2);
uart_Close(hPort);
```

[C#]

```
bool ret;
byte[] cmd = new byte[2] ;
IntPtr hPort;
byte[] buf = new byte[2] ;
cmd[0] = 0x41;
cmd[1] = 0x42;
hPort = PACNET.UART.Open("COM4,9600,N,8,1");
ret = PACNET.UART.BinSendCmd(hPort, cmd, 2, buf, 2);
PACNET.UART.Close(hPort);
```

Remarks

This function will call PurgeComm() to clear serial COM port output and input buffer.

3.6.14. uart_GetLineStatus

This function retrieves the specifiesa pin status of COM port.

Syntax

C++

```
BOOL uart_GetLineStatus(
    HANDLE hPort,
    INT pin
);
```

Parameters

hPort

[in] Handle to the open COM port.

pin

[in] A variable specifies pin of the COM port.

This parameter can be following values:

```
#define    CTS    0
#define    DSR    1
#define    RI     2
#define    CD     3
```

Return Value

TRUE indicates the pin's state is ON. 0 indicates OFF.

Examples

[C]

```
HANDLE hPort = uart_Open("COM5,115200,N,8,1");
BOOL ret = uart_GetLineStatus(hPort, DSR);    // for example the pin, DSR
if(ret)
   printf("The status of DSR is ON\n");
else
   printf("The status of DSR is OFF \n");
uart_Close(hPort);
```

[C#]

```
IntPtr hPort = PACNET.UART.Open( "COM5,115200,N,8,1" );
bool ret = PACNET.UART.GetLineStatus(hPort, DSR); // for example the pin, DSR
if(ret)
   Console.WriteLine("The status of DSR is ON");
else
   Console.WriteLine ("The status of DSR is OFF");
PACNET.UART.Close(hPort);
```

3.6.15. uart_GetDataSize

This function retrieves the number of bytes received by the serial provider but not yet read by a uart_Recv operation, or of user data remaining to transmitted for write operations.

Syntax

C++

```
BOOL uart_GetDataSize(
    HANDLE hPort,
    int DATA_TYPE
);
```

Parameters

hPort

[in] Handle to the open COM port.

data_type

[in] A value specifies to retrieve in or out buffer.

This parameter can be following values:

```
#define    IN_DATA    0
#define    OUT_DATA    1
```

Return Value

The number of bytes in/out buffer but not yet read/write.

3.6.16. uart_SetLineStatus

This function sets the specifiesa pin status of COM port .

Syntax

C++

```
DWORD uart_SetLineStatus(
    HANDLE hPort,
    INT pin,
    INT mode,
);
```

Parameters

hPort

[in] Handle to the open COM port.

pin

[in] A variable specifies pin of the COM port.

This parameter can be following values:

#define DTR 1

#define RTS 2

#define DTR + RTS 3

mode

[in] 0: Disable, Set the pin signal is OFF

1: Enable, Set the pin signal is ON

Return Value

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero.

To get an error code, call pac_GetLastError. A nonzero error code defined in PACERROR.h indicates failure. If error code getting from pac_GetLastError is PAC_ERR_UART_GET_COMM_STATUS_ERROR, call the GetLastError function to obtain the last error ocde of Windows API.

Examples

[C]

```
HANDLE hPort = uart_Open(“COM5, 9600, N, 8,1”); //DTR pin on COM5 of XPAC
//HANDLE hPort = uart_Open(“COM4, 9600, N, 8,1”); // DTR pin on COM4 of WinPAC
uart_SetLineStatus(hPort, 1, 1); // set DTR to ON
uart_Close(hPort);
```

[C#]

```
IntPtr 的 hPort = PACNET.UART.Open(“COM5, 9600, N, 8,1”); //DTR pin on COM5 of XPAC
//IntPtr 的 hPort = PACNET.UART.Open(“COM4, 9600, N, 8,1”); // DTR pin on COM4 of WinPAC
PACNET.UART。SetLineStatus(hPort, 1,1); // set DTR to ON
PACNET.UART.Close(hPort);
```

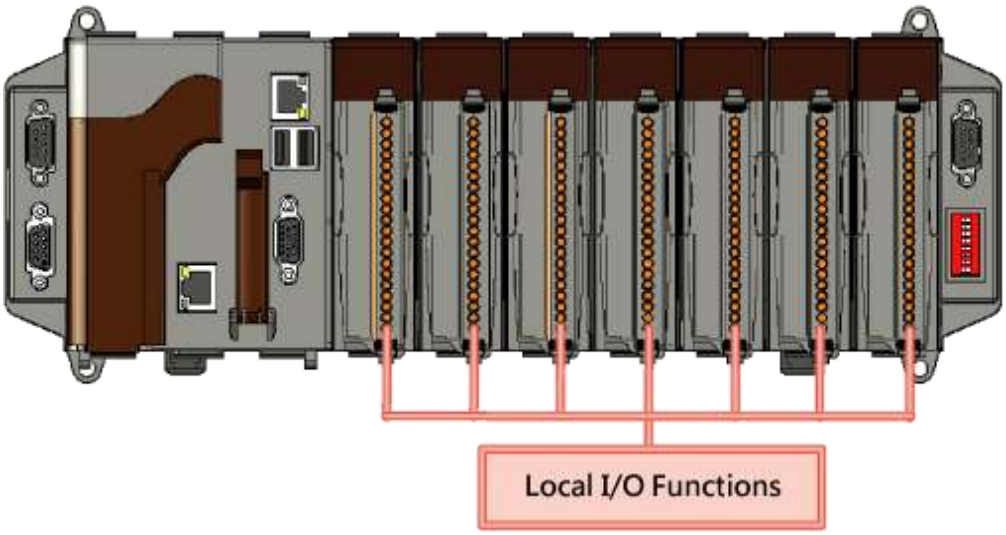
3.7. PAC_IO API

PAC_IO API supports to operate I/O modules, which can be divided into the following parts:

Local (I/O in slot)

In the local mode, the slot range is from 0 to 7, the same the iSlot as follow.

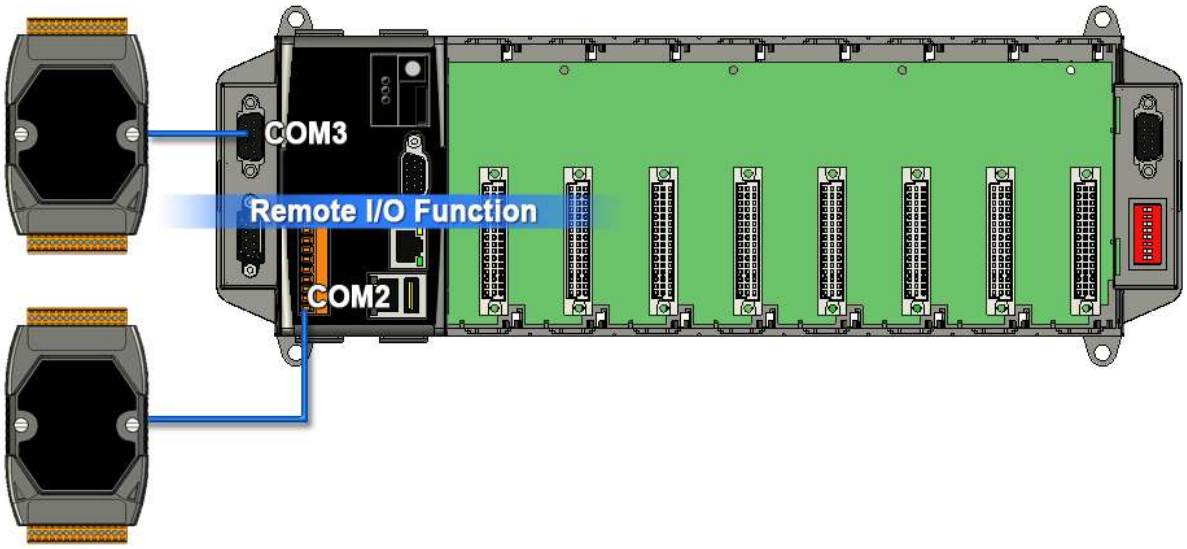
```
hPort = uart_Open("");
//Clear DO
pac_WriteDO( hPort, iSlot, iDo_TotalCh, 0);
```



Remote

If remote mode, the address need call a macro, PAC_REMOTE_IO. And its range is from 0 to 255. For example as follow:

```
//Write DO value to remote module
HANDLE hPort = uart_Open(ConnectionString);
if( !hPort ) AfxMessageBox( _T("Open Com Error"));

pac_WriteDO( hPort, PAC_REMOTE_IO(iAddr), m_iDOCHs, lDO_Value);
```



In PACSDK.dll, modify the processing to send the DCON commands without determining the module name, the new PAC_IO API functions can support to access the I-87K/I-8K, I-7K, I-8000 units, tM series, and etc DCON modules.

You also need to know the expansion capacities in order to choose the best expansion module for achieving maximal efficiency.

For more information about expansion modules that are compatible with the XPAC/WinPAC series, please refer to

I-8K/I-87K series

https://www.icpdas.com/en/product/guide+Remote__I_O__Module__and__Unit+PAC__%EF%BC%86amp;__Local__I_O__Modules+I-8K_I-87K__Series__(High__Profile)

I-7K series

https://www.icpdas.com/en/product/guide+Remote__I_O__Module__and__Unit+RS-485__I_O__Modules+I-7000%23464

I-8K units

https://www.icpdas.com/en/product/guide+Remote__I_O__Module__and__Unit+RS-485__I_O__Modules+IO__Expansion__Unit

TM series

https://www.icpdas.com/en/product/guide+Remote__I_O__Module__and__Unit+RS-485__I_O__Modules+tM__series

API functions for the Multi-function DCON modules

PAC_IO API has provided 2 types functions.

One type which includes pac_WriteDO, pac_ReadDIO, pac_ReadDI, pac_ReadDO, pac_ReadDIO_DIBit, pac_ReadDIO_DOBit, pac_ReadDIBit, pac_ReadDOBit, pac_ReadDICNT, pac_ClearDICNT, pac_ReadDICNTAll, pac_ClearDICNTAll functions is used to access the pure DIO DCON modules, which are defeind as modules that only has DI/DO or DIO channels.

The other type which includes pac_WriteDO_MF, pac_ReadDIO_MF and pac_ReadDI_MF, etc functions is used to access the Multi-function DCON modules, which are defined as modules that mainly act as AIO or Counters but are equipped with DIO channels. Such as the I-87005W/I-87016W/I-87082W/I-7016/I-7088, etc.)

The instructions of two fucntions (i.e. pac_WriteDO and pac_WriteDO_MF) are placed on the same section because of the definition of the parameters and Return Value of this pair of functions are the same..

The functions used to access the pure DIO DCON modules cannot be used to access Multi-function DCON modules. If use the function to access Multi-function DCON modules and vice versa. The function will return 0x14003 meaning of “Uart response error “.

Supported PACs

The table below lists the PAC IO functions that are supported by each PAC.

Models / Functions	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_GetBit	Y	-	Y	-	Y	Y	Y	-	Y
pac_WriteDO/pac_WriteDO_MF	Y	-	Y	-	Y	Y	Y	-	Y
pac_WriteDOBit	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadDO/pac_ReadDO_MF	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadDI/pac_ReadDI_MF	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadDIO/pac_ReadDIO_MF	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadDILatch	Y	-	Y	-	Y	Y	Y	-	Y
pac_ClearDILatch	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadDIOLatch	Y	-	Y	-	Y	Y	Y	-	Y
pac_ClearDIOLatch	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadDICNT/pac_ReadDICNT_MF	Y	-	Y	-	Y	Y	Y	-	Y
pac_ClearDICNT/pac_ClearDICNT_MF	Y	-	Y	-	Y	Y	Y	-	Y

Models / Functions	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_ReadDICNTAll	Y	-	Y	-	Y	Y	Y	-	Y
pac_ClearDICNTAll	Y	-	Y	-	Y	Y	Y	-	Y
pac_WriteAO/pac_WriteAO_MF	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadAO	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadAI	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadAIHex	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadAIAllExt	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadAIAll	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadAIAllHexExt	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadAIAllHex	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadCNT	Y	-	Y	-	Y	Y	Y	-	Y
pac_ClearCNT	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadCNTOverflow	Y	-	Y	-	Y	Y	Y	-	Y
pac_WriteModuleSafeValueDO/pac_WriteModuleSafeValueDO_MF	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadModuleSafeValueDO/pac_ReadModuleSafeValueDO_MF	Y	-	Y	-	Y	Y	Y	-	Y

Models / Functions	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_WriteModulePowerOnValueDO/pac_WriteModulePowerOnValueDO_MF	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadModulePowerOnValueDO/pac_ReadModulePowerOnValueDO_MF	Y	-	Y	-	Y	Y	Y	-	Y
pac_WriteModuleSafeValueAO	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadModuleSafeValueAO	Y	-	Y	-	Y	Y	Y	-	Y
pac_WriteModulePowerOnValueAO	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadModulePowerOnValueAO	Y	-	Y	-	Y	Y	Y	-	Y
pac_GetModuleLastOutputSource	Y	-	Y	-	Y	Y	Y	-	Y
pac_GetModuleWDTStatus	Y	-	Y	-	Y	Y	Y	-	Y
pac_GetModuleWDTConfig	Y	-	Y	-	Y	Y	Y	-	Y
pac_SetModuleWDTConfig	Y	-	Y	-	Y	Y	Y	-	Y
pac_ResetModuleWDT	Y	-	Y	-	Y	Y	Y	-	Y
pac_RefreshModuleWDT	Y	-	Y	-	Y	Y	Y	-	Y

Functions \ Models	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_InitModuleWDTInterrupt	Y	-	Y	-	Y	Y	Y	-	Y
pac_GetModuleWDTInterruptStatus	Y	-	Y	-	Y	Y	Y	-	Y
pac_SetModuleWDTInterruptStatus	Y	-	Y	-	Y	Y	Y	-	Y

PAC_IO Functions

The following functions are used to retrieve or set the I/O modules.

PACSDK Functions	PACNET Functions	Description
pac_GetBit	PAC_IO.GetBit	Retrieves the value which in specific bit to check the DI specific channel status.
pac_WriteDO/pac_WriteDO_MF	PAC_IO.WriteDO\PAC_IO.WriteDO_MF	Writes the DO values to DO modules.
pac_WriteDOBit	PAC_IO.WriteDOBit	Writes a single bit of value to the DO module, that is, only the channel corresponding to the bit is changed.
pac_ReadDO/pac_ReadDO_MF	PAC_IO.ReadDO\PAC_IO.ReadDO_MF	Reads the DO value of the DO module.
pac_ReadDI/pac_ReadDI_MF	PAC_IO.ReadDI\PAC_IO.ReadDI_MF	Reads the DI value of the DI module.
pac_ReadDIO/pac_ReadDIO_MF	PAC_IO.ReadDIO\PAC_IO.ReadDIO_MF	Reads the DI and the DO values to the DIO module.
pac_ReadDILatch	PAC_IO.ReadDILatch	Reads the DI latch value of the DI module.
pac_ClearDILatch	PAC_IO.ClearDILatch	Clears the latch value of the DI module.
pac_ReadDIOLatch	PAC_IO.ReadDIOLatch	Reads the latch values of the DI and DO channels of the DIO module.
pac_ClearDIOLatch	PAC_IO.ClearDIOLatch	clears the latch values of DI and DO channels of the DIO module.
pac_ReadDICNT/pac_ReadDICNT_MF	PAC_IO.ReadDICNT\PAC_IO.ReadDICNT_MF	Reads the counts of the DI channels of the DI module.
pac_ClearDICNT/pac_ClearDICNT_MF	PAC_IO.ClearDICNT\PAC_IO.ClearDICNT_MF	Clears the counter value of the DI channel of the DI module.
pac_ReadDICNTAll	PAC_IO.ReadDICNTAll	Reads the counts of all DI channels of the DI module.

PACSDK Functions	PACNET Functions	Description
pac_ClearDICNTAll	PAC_IO.ClearDICNTAll	Clears the counter value of all DI channels of the DI module.
pac_WriteAO/pac_WriteAO_MF	PAC_IO.WriteAO\PAC_IO.WriteAO_MF	Writes the AO value to the AO modules.
pac_ReadAO	PAC_IO.ReadAO	Reads the AO value of the AO module
pac_ReadAI	PAC_IO.ReadAI	Reads the engineering-mode AI value of the AI module.
pac_ReadAIHex	PAC_IO.ReadAIHex	Reads the 2's complement-mode AI value of the AI module.
pac_ReadAIAllExt	PAC_IO.ReadAIAllExt	Reads all the AI values of all channels in engineering-mode of the AI module
pac_ReadAIAll	PAC_IO.ReadAIAll	Reads all the AI values of all channels in engineering-mode of the AI module.
pac_ReadAIAllHexExt	PAC_IO.ReadAIAllHexExt	Reads all the AI values of all channels in 2's complement-mode of the AI module
pac_ReadAIAllHex	PAC_IO.ReadAIAllHex	Reads all the AI values of all channels in 2's complement-mode of the AI module.
pac_ReadCNT	PAC_IO.ReadCNT	Reads the counter values of the counter/frequency modules.
pac_ClearCNT	PAC_IO.ClearCNT	pac_ClearCNTclears the counter values of the counter/frequency modules.
pac_ReadCNTOverflow	PAC_IO.ReadCNTOverflow	Clears the counter overflow value of the counter/frequency modules.
pac_WriteModuleSafeValueDO pac_WriteModuleSafeValueDO_MF	PAC_IO.WriteModuleSafeValueDO PAC_IO.WriteModuleSafeValueDO_MF	Writes the DO safe values to DO modules.

PACSDK Functions	PACNET Functions	Description
pac_ReadModuleSafeValueDO pac_ReadModuleSafeValueDO_MF	PAC_IO.ReadModuleSafeValueDO PAC_IO.ReadModuleSafeValueDO_MF	Reads the safe value of the DO modules.
pac_WriteModulePowerOnValueDO pac_WriteModulePowerOnValueDO_MF	PAC_IO.WriteModulePowerOnValueDO PAC_IO.WriteModulePowerOnValueDO_MF	Writes the DO power on values to DO modules.
pac_ReadModulePowerOnValueDO pac_ReadModulePowerOnValueDO_MF	PAC_IO.ReadModulePowerOnValueDO PAC_IO.ReadModulePowerOnValueDO_MF	Reads the power on value of the DO modules.
pac_WriteModuleSafeValueAO	PAC_IO.WriteModuleSafeValueAO	Writes the AO safe value of the AO modules.
pac_ReadModuleSafeValueAO	PAC_IO.ReadModuleSafeValueAO	Reads the AO safe value of the AO module.
pac_WriteModulePowerOnValueAO	PAC_IO.WriteModulePowerOnValueAO	Writes the AO power on value to the AO modules.
pac_ReadModulePowerOnValueAO	PAC_IO.ReadModulePowerOnValueAO	Reads the AO power on value of the AO module.
pac_GetModuleLastOutputSource	PAC_IO.GetModuleLastOutputSource	Reads the last output source of a module.
pac_GetModuleWDTStatus	PAC_IO.GetModuleWDTStatus	Reads the module watchdog status of a module.
pac_GetModuleWDTConfig	PAC_IO.GetModuleWDTConfig	Reads the module watchdog status of a module.
pac_SetModuleWDTConfig	PAC_IO.SetModuleWDTConfig	Reads the module watchdog status of a module.
pac_ResetModuleWDT	PAC_IO.ResetModuleWDT	Resets the module watchdog timeout status of a module.
pac_RefreshModuleWDT	PAC_IO.RefreshModuleWDT	Refreshes the module watchdog.
pac_InitModuleWDTInterrupt	PAC_IO.InitModuleWDTInterrupt	Initializes and enables interrupt of a module watchdog.
pac_GetModuleWDTInterruptStatus	PAC_IO.GetModuleWDTInterruptStatus	Reads interrupt status of a module watchdog.
pac_SetModuleWDTInterruptStatus	PAC_IO.SetModuleWDTInterruptStatus	Enables/Disables interrupt of a module watchdog.

3.7.1. pac_GetBit

The function retrieves the value which in specific bit to check the DI specific channel status..

Syntax

C++

```
BOOL pac_GetBit(
    int V,
    int NDX
);
```

Parameters

v

Which I/O result wants to get bit.

ndx

Specific bit to retrieve.

Return Value

The value of specific index.

Examples

[C]

```
BYTE bit3;
int index= 3;
BYTE iSlot = 2;
int iDI_TotalCh = 8;
DWORD lDI_Value;
HANDLE hPort;
hPort = uart_Open("");
BOOL IRET = pac_ReadDI(hPort, iSlot, iDI_TotalCh, lDI_Value);
bit3= pac_GetBit(lDI_Value, index);
uart_Close(hPort);
```

[C#]

```
bool bit3;
int index= 3;
Byte iSlot = 2;
Byte iSlot = 2;
int iDI_TotalCh = 8;
UINT lDI_Value = 0;
IntPtr hPort;
hPort = PACNET.UART.Open("");
Bool IRET = PACNET.PAC_IO.ReadDI(hPort, iSlot, iDI_TotalCh, ref lDI_Value);
bit3= PACNET.PAC_IO.GetBit(lDI_Value, index);
PACNET.UART.Close(hPort);
```

Remarks

The function is used the same as v & (1<<index).

3.7.2. pac_WriteDO/pac_WriteDO_MF

This function writes the DO values to DO modules.

Syntax

C++

```
pac_WriteDO
    BOOL pac_WriteDO(
    HANDLE hPort,
    int slot,
    int iDO_TotalCh,
    DWORD lDO_Value
);
```

C++

```
pac_WriteDO_MF
    BOOL pac_WriteDO_MF(
    HANDLE hPort,
    int slot,
    int iDO_TotalCh,
    DWORD lDO_Value
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iDO_TotalCh

[in] The total number of DO channels of the DO modules.

iDO_Value

[in] A 8-digit hexadecimal value, where bit 0 corresponds to DO0, bit 31 corresponds to DO31, etc. When the bit is 1, it denotes that the digital output channel is on, and 0 denotes that the digital output channel is off.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Remarks

The definition of the parameters and Return Value of pac_WriteDO and pac_WriteDO_MF functions are the same. The different is that pac_WriteDO is applied to the pure DIO DCON modules and pac_WriteDO_MF is applied to the Multi-function DCON modules.

Examples

[C] pac_WriteDO

Example 1：

```
// If the module is remote
HANDLE hPort;
hPort = uart_Open( "COM2, 9600, N, 8, 1" );
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //turn on the channel two
BOOL RET = pac_WriteDO(hPort, PAC_REMOTE_IO(1), TOTAL_CHANNEL, do_value);
uart_Close(hPort);
```

Example 2：

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //turn on the channel two
BOOL RET = pac_WriteDO(hPort, 1, TOTAL_CHANNEL, do_value);
uart_Close(hPort);
```

Example 3：

```
// If the module is 8k local
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //turn on the channel two
BOOL RET = pac_WriteDO(0, 1, TOTAL_CHANNEL, do_value);
```

[C] pac_WriteDO_MF

Example 1：

```
// If the module is remote
HANDLE hPort;
hPort = uart_Open( "COM2, 9600, N, 8, 1" );
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //turn on the channel two
BOOL RET = pac_WriteDO_MF(hPort, PAC_REMOTE_IO(1), TOTAL_CHANNEL, do_value);
uart_Close(hPort);
```

Example 2：

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //turn on the channel two
BOOL RET = pac_WriteDO_MF(hPort, 1, TOTAL_CHANNEL, do_value);
uart_Close(hPort);
```

[C#] pac_WriteDO

Example 1：

```
// If the module is remote
IntPtr hPort;
hPort = PACNET.UART.Open( “COM1, 9600, N, 8, 1” );
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //turn on the channel two
Bool RET = PACNET.PAC_IO.WriteDO(hPort, PACNET.PAC_IO.REMOTE_IO(1),
TOTAL_CHANNEL, do_value);
PACNET.UART.Close(hPort);
```

Example 2：

```
// If the module is 87k local
IntPtr hPort; hPort = PACNET.UART.Open(“”);
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //turn on the channel two
boolRET = PACNET.PAC_IO.WriteDO(hPort, 1, TOTAL_CHANNEL, do_value);
PACNET.UART.Close(hPort);
```

Example 3：

```
// If the module is 8k local
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //turn on the channel two
bool 型 RET = PACNET.PAC_IO.WriteDO(0, 1, TOTAL_CHANNEL, do_value);
```

[C#] pac_WriteDO_MF

Example 1：

```
// If the module is remote
IntPtr hPort; hPort = PACNET.UART.Open( "COM1, 9600, N, 8, 1" );
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //turn on the channel two
boolRET = PACNET.PAC_IO.WriteDO_MF(hPort, PACNET.PAC_IO.REMOTE_IO(1),
TOTAL_CHANNEL, do_value);
PACNET.UART.Close(hPort);
```

Example 2：

```
// If the module is 87k local
IntPtr hPort; hPort = PACNET.UART.Open("");
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //turn on the channel two
boolRET = PACNET.PAC_IO.WriteDO_MF(hPort, 1, TOTAL_CHANNEL, do_value);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.

3.7.3. pac_WriteDOBit

This function writes a single bit of value to the DO module, that is, only the channel corresponding to the bit is changed.

Syntax

C++

```
BOOL pac_WriteDOBit(
    HANDLE hPort,
    int slot,
    int iDO_TotalCh,
    int iChannel,
    int iBitValue
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iDO_TotalCh

[in] The total number of DO channels of the DO modules.

iChannel

[in]The DO channel to be change.

iBitValue

[in] 1 is to turn on the DO channel; 0 is off.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

Example 1：

```
HANDLE hPort; // If the module is 87k local
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
int iDO_TotalCh = 8;
int iBitValue = 1;
BOOL RET = pac_WriteDOBit(hPort, iSlot, iChannel, miDO_TotalCh, iBitValue);
uart_Close(hPort);
```

Example 2：

```
BYTE iSlot = 1; // If the module is 8k local
int iChannel = 2;
int iDO_TotalCh = 8;
int iBitValue = 1;
BOOL RET = pac_WriteDOBit(0, iSlot, iChannel, miDO_TotalCh, iBitValue);
```

[C#]

```
IntPtrhPort; // If the module is 87k local
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
int iDO_TotalCh = 8;
int iBitValue = 1;
boolRET = PACNET.PAC_IO.WriteDOBit(hPort, iSlot, iChannel, miDO_TotalCh, iBitValue);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.

3.7.4. pac_ReadDO/pac_ReadDO_MF

This function reads the DO value of the DO module.

Syntax

C++ - pac_ReadDO

```
BOOL pac_ReadDO(
    HANDLE hPort,
    int slot,
    int iDO_TotalCh,
    DWORD * lDO_Value
);
```

C++ - pac_ReadDO_MF

```
BOOL pac_ReadDO_MF(
    HANDLE hPort,
    int slot,
    int iDO_TotalCh,
    DWORD * lDO_Value
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iDO_TotalCh

[in] The total number of DO channels of the DO modules.

lDO_Value

[in] The pointer of the DO value to read from the DO module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C] pac_ReadDO

Example 1：

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
byte slot= 1;
int TOTAL_CHANNEL = 8;
DWORD do_value;
BOOL RET = pac_ReadDO(hPort, slot, TOTAL_CHANNEL, &do_value);
uart_Close(hPort);
```

Example 2：

```
// If the module is 8k local
byte slot= 1;
int TOTAL_CHANNEL = 8;
DWORD do_value;
BOOL RET = pac_ReadDO(0, slot, TOTAL_CHANNEL, &do_value);
```

[C] pac_ReadDO_MF

Example 1：

```
HANDLE hPort; // If the module is 87k local
hPort = uart_Open("");
byte slot= 1;
int TOTAL_CHANNEL = 8;
DWORD do_value;
BOOL RET = pac_ReadDO_MF(hPort, slot, TOTAL_CHANNEL, &do_value);
uart_Close(hPort);
```

[C#] pac_ReadDO

```
IntPtr hPort; // If the module is 87k local
hPort = PACNET.UART.Open("");
byte slot= 1;
int TOTAL_CHANNEL = 8;
UINT do_value;
boolRET = PACNET.PAC_IO.ReadDO(hPort, slot, TOTAL_CHANNEL, ref do_value);
PACNET.UART.Close(hPort);
```

[C#] pac_ReadDO_MF

```
IntPtr hPort; // If the module is 87k local
hPort = PACNET.UART.Open("");
byte slot= 1;
int TOTAL_CHANNEL = 8;
UINT do_value;
boolRET = PACNET.PAC_IO.ReadDO_MF(hPort, slot, TOTAL_CHANNEL, ref do_value);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.

3.7.5. pac_ReadDI/pac_ReadDI_MF

This function reads the DI value of the DI module.

Syntax

C++ - pac_ReadDI

```
BOOL pac_ReadDI(
    HANDLE hPort,
    int slot,
    int iDI_TotalCh,
    DWORD * lDI_Value
);
```

C++ - pac_ReadDI_MF

```
BOOL pac_ReadDI_MF(
    HANDLE hPort,
    int slot,
    int iDI_TotalCh,
    DWORD * lDI_Value
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iDI_TotalCh

[in] The total channels of the DI module.

lDI_Value

[out] The pointer to DI value to read back.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C] pac_ReadDI

Example 1：

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 2;
int iDI_TotalCh = 8;
DWORD lDI_Value;
BOOL IRET = pac_ReadDI(hPort, iSlot, iDI_TotalCh, &lDI_Value);
uart_Close(hPort);
```

Example 2：

```
// If the module is 8k local
BYTE iSlot = 2;
int iDI_TotalCh = 8;
DWORD lDI_Value;
BOOL IRET = pac_ReadDI(0, iSlot, iDI_TotalCh,  &lDI_Value);
```

[C] pac_ReadDI_MF

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 2;
int iDI_TotalCh = 8;
DWORD lDI_Value;
BOOL IRET = pac_ReadDI_MF(hPort, iSlot, iDI_TotalCh, &lDI_Value);
uart_Close(hPort);
```

[C#] pac_ReadDI

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 2;
int iDI_TotalCh = 8;
UINT lDI_Value;
boolIRET = PACNET.PAC_IO.ReadDI(hPort, iSlot, iDI_TotalCh, ref lDI_Value);
PACNET.UART.Close(hPort);
```

[C#] pac_ReadDI_MF

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 2;
int iDI_TotalCh = 8;
UINT lDI_Value;
boolIRET = PACNET.PAC_IO.ReadDI_MF(hPort, iSlot, iDI_TotalCh, ref lDI_Value);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.

3.7.6. pac_ReadDIO/pac_ReadDIO_MF

This function reads the DI and the DO values of the DIO module.

Syntax

C++ - pac_ReadDIO

```
BOOL pac_ReadDIO(
    HANDLE hPort,
    int slot,
    int iDI_TotalCh,
    int iDO_TotalCh,
    DWORD * lDI_Value,
    DWORD * lDO_Value
);
```

C++ - pac_ReadDIO_MF

```
BOOL pac_ReadDIO_MF(
    HANDLE hPort,
    int slot,
    int iDI_TotalCh,
    int iDO_TotalCh,
    DWORD * lDI_Value,
    DWORD * lDO_Value
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iDI_TotalCh

[in] The total number of DI channels of the DIO module.

iDO_TotalCh

[in] The total number of DO channels of the DIO module.

lDI_Value

[out] The pointer to the value of DI read back.

lDO_Value

[out] The pointers to the value of DO read back.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C] pac_ReadDIO

Example 1：

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iDI_TotalCh = 8;
int iDO_TotalCh = 8;
DWORD lDI_Value;
DWORD lDO_Value;
BOOL IRET = pac_ReadDIO(hPort, iSlot, iDI_TotalCh, iDO_TotalCh, &lDI_Value, &lDO_Value);
uart_Close(hPort);
```

Example 2：

```
// If the module is 8k local
BYTE iSlot = 1;
int iDI_TotalCh = 8;
int iDO_TotalCh = 8;
DWORD lDI_Value;
DWORD lDO_Value;
BOOL IRET = pac_ReadDIO(0, iSlot, iDI_TotalCh, iDO_TotalCh, &lDI_Value, &lDO_Value);
```

[C] pac_ReadDIO_MF

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iDI_TotalCh = 8;
int iDO_TotalCh = 8;
DWORD lDI_Value;
DWORD lDO_Value;
BOOL IRET = pac_ReadDIO(hPort, iSlot, iDI_TotalCh, iDO_TotalCh, &lDI_Value, &lDO_Value);
uart_Close(hPort);
```

[C#] pac_ReadDIO

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iDI_TotalCh = 8;
int iDO_TotalCh = 8;
UINT lDI_Value;
UINT lDO_Value;
boolIRET = PACNET.PAC_IO.ReadDIO(hPort, iSlot, iDI_TotalCh, iDO_TotalCh, ref lDI_Value, ref lDO_Value);
PACNET.UART.Close(hPort);
```

[C#] pac_ReadDIO_MF

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iDI_TotalCh = 8;
int iDO_TotalCh = 8;
UINT lDI_Value;
UINT lDO_Value;
boolIRET = PACNET.PAC_IO.ReadDIO_MF(hPort, iSlot, iDI_TotalCh, iDO_TotalCh, ref
lDI_Value, ref lDO_Value);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO (0...255), which range is from 0 to 255.

3.7.7. pac_ReadDILatch

This function reads the DI latch value of the DI module.

Syntax

C++

```
BOOL pac_ReadDILatch(
    HANDLE hPort,
    int slot,
    int iDI_TotalCh,
    int iLatchType,
    DWORD * lDI_Latch_Value
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro,PAC_REMOTE_IO(0...255).

iDI_TotalCh

[in] The total number of the DI channels of the DI module.

iLatchType

[in] The latch type specified to read latch value back.

1 = latched high status

0 = latched low status

lDI_Latch_Value

[out] The pointer to the latch value read back from the DI module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

Example 1：

```
HANDLE hPort; // If the module is 87k local
hPort = uart_Open("");
BYTE iSlot = 1;
int iDI_TotalCh = 8;
int iLatchType = 0;
DWORD lDI_Latch_Value;
BOOL IRET = pac_ReadDILatch(hPort, iSlot, iDI_TotalCh, iLatchType, &lDI_Latch_Value);
uart_Close(hPort);
```

Example 2：

```
BYTE iSlot = 1; // If the module is 8k local
int iDI_TotalCh = 8;
int iLatchType = 0;
DWORD lDI_Latch_Value;
BOOL IRET = pac_ReadDILatch(0, iSlot, iDI_TotalCh, iLatchType,  &lDI_Latch_Value);
```

[C#]

```
IntPtr hPort; // If the module is 87k local
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iDI_TotalCh = 8;
int iLatchType = 0;
UINT lDI_Latch_Value;
boolIRET = PACNET.PAC_IO.ReadDILatch(hPort, iSlot, iDI_TotalCh, iLatchType, ref
lDI_Latch_Value);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.

3.7.8. pac_ClearDILatch

This function clears the latch value of the DI module.

Syntax

C++

```
BOOL pac_ClearDILatch(
    HANDLE hPort,
    int slot
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

Example 1：

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
BOOL IRET = pac_ClearDILatch(hPort, iSlot);
uart_Close(hPort);
```

Example 2：

```
// If the module is 8k local
BYTE iSlot = 1;
BOOL IRET = pac_ClearDILatch(0, iSlot);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
boolIRET = PACNET.PAC_IO.ClearDILatch(hPort, iSlot);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.

3.7.9. pac_ReadDIOLatch

This function reads the latch values of the DI and DO channels of the DIO module.

Syntax

C++

```
BOOL pac_ReadDIOLatch(
    HANDLE hPort,
    int slot,
    int iDI_TotalCh,
    int iDO_TotalCh,
    int iLatchType,
    DWORD * lDI_Latch_Value,
    DWORD * lDO_Latch_Value
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iDI_TotalCh

[in] The total number of the DI channels of the DIO module.

iDO_TotalCh

[in] The total number of the DO channels of the DIO module.

iLatchType

[in] The type of the latch value read back.

1 = latched high status

0 = latched low status

lDI_Latch_Value

[out] The pointer to the DI latch value read back.

lDO_Latch_Value

[out] The pointer to the DO latch value read back.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

Example 1：

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iDI_TotalCh = 8;
int iDO_TotalCh = 8;
int iLatchType = 0;
DWORD lDI_Latch_Value;
DWORD lDO_Latch_Value;
BYTE cDI_Latch_BitValue;
BYTE cDO_Latch_BitValue;
BOOL IRET = pac_ReadDIOLatch(hPort, iSlot, iDI_TotalCh, iDO_TotalCh, iLatchType,
&lDI_Latch_Value, &lDO_Latch_Value);
uart_Close(hPort);
```

Example 2：

```
// If the module is 8k local
BYTE iSlot = 1;
int iDI_TotalCh = 8;
int iDO_TotalCh = 8;
int iLatchType = 0;
DWORD lDI_Latch_Value;
DWORD lDO_Latch_Value;
BYTE cDI_Latch_BitValue;
BYTE cDO_Latch_BitValue;
BOOL lRET = pac_ReadDIOLatch(0, iSlot, iDI_TotalCh, iDO_TotalCh, iLatchType, &
lDI_Latch_Value, &lDO_Latch_Value);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iDI_TotalCh = 8;
int iDO_TotalCh = 8;
int iLatchType = 0;
UINT lDI_Latch_Value;
UINT lDO_Latch_Value;
byte cDI_Latch_BitValue;
byte cDO_Latch_BitValue;
boollRET = PACNET.PAC_IO.ReadDIOLatch(hPort, iSlot, iDI_TotalCh, iDO_TotalCh, iLatchType,
ref lDI_Latch_Value, ref lDO_Latch_Value);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.

3.7.10. pac_ClearDIOLatch

This function clears the latch values of DI and DO channels of the DIO module.

Syntax

C++

```
BOOL pac_ClearDIOLatch(
    HANDLE hPort,
    int slot
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO (0...255).

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

Example 1：

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
BOOL IRET = pac_ClearDIOLatch(hPort, iSlot);
uart_Close(hPort);
```

Example 2：

```
// If the module is 8k local
BYTE iSlot = 1;
BOOL IRET = pac_ClearDIOLatch(0, iSlot);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
boolIRET = PACNET.PAC_IO.ClearDIOLatch(hPort, iSlot);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO (0...255), which range is from 0 to 255.

3.7.11. pac_ReadDICNT/pac_ReadDICNT_MF

This function reads the counts of the DI channels of the DI module.

Syntax

C++ - pac_ReadDICNT

```
BOOL pac_ReadDICNT(
    HANDLE hPort,
    int slot,
    int iChannel,
    int iDI_TotalCh,
    DWORD * lCounter_Value);
```

C++ - pac_ReadDICNT_MF

```
BOOL pac_ReadDICNT_MF(
    HANDLE hPort,
    int slot,
    int iChannel,
    int iDI_TotalCh,
    DWORD * lCounter_Value);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] The channel that the counter value belongs.

iDI_TotalCh

[in] Total number of the DI channels of the DI module.

lCounter_Value

[out] The pointer to the counter value.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C] pac_ReadDICNT

Example 1：

```
HANDLE hPort; // If the module is 87k local
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
int iDI_TotalCh = 8;
DWORD lCounter_Value;
BOOL IRET = pac_ReadDICNT(hPort, iSlot, iChannel, iDI_TotalCh, &lCounter_Value);
uart_Close(hPort);
```

Example 2：

```
BYTE iSlot = 1; // If the module is 8k local
int iChannel = 2;
int iDI_TotalCh = 8;
DWORD lCounter_Value;
BOOL IRET = pac_ReadDICNT(0, iSlot, iChannel, iDI_TotalCh,  &lCounter_Value);
```

[C] pac_ReadDICNT_MF

```
HANDLE hPort; // If the module is 87k local
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
int iDI_TotalCh = 8;
DWORD lCounter_Value;
BOOL IRET = pac_ReadDICNT_MF(hPort, iSlot, iChannel, iDI_TotalCh, &lCounter_Value);
uart_Close(hPort);
```

[C#] pac_ReadDICNT

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
int iDI_TotalCh = 8;
UINT lCounter_Value;
boolIRET = PACNET.PAC_IO.ReadDICNT(hPort, iSlot, iChannel, iDI_TotalCh, ref
lCounter_Value);
PACNET.UART.Close(hPort);
```

[C#] pac_ReadDICNT_MF

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
int iDI_TotalCh = 8;
UINT lCounter_Value;
boolIRET = PACNET.PAC_IO.ReadDICNT_MF(hPort, iSlot, iChannel, iDI_TotalCh, ref
lCounter_Value);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.

3.7.12. pac_ClearDICNT/pac_ClearDICNT_MF

This function clears the counter value of the DI channel of the DI module.

Syntax

C++ -pac_ClearDICNT

```
BOOL pac_ClearDICNT(
    HANDLE hPort,
    int slot,
    int iChannel,
    int iDI_TotalCh
);
```

C++ - pac_ClearDICNT_MF

```
BOOL pac_ClearDICNT_MF(
    HANDLE hPort,
    int slot,
    int iChannel,
    int iDI_TotalCh
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] The channel that the counter value belongs.

iDI_TotalCh

[in] Total number of the DI channels of the DI module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C] pac_ClearDICNT

Example 1：

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
int iDI_TotalCh = 8;
BOOL IRET = pac_ClearDICNT(hPort, iSlot, iChannel, iDI_TotalCh);
uart_Close(hPort);
```

Example 2：

```
// If the module is 8k local
BYTE iSlot = 1;
int iChannel = 2;
int iDI_TotalCh = 8;
BOOL IRET = pac_ClearDICNT(0, iSlot, iChannel, iDI_TotalCh);
```

[C] pac_ClearDICNT_MF

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
int iDI_TotalCh = 8;
BOOL IRET = pac_ClearDICNT_MF(hPort, iSlot, iChannel, iDI_TotalCh);
uart_Close(hPort);
```

[C#] pac_ClearDICNT

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
int iDI_TotalCh = 8;
boolIRET = PACNET.PAC_IO.ClearDICNT(hPort, iSlot, iChannel, iDI_TotalCh);
PACNET.UART.Close(hPort);
```

[C#] pac_ClearDICNT_MF

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
int iDI_TotalCh = 8;
boolIRET = PACNET.PAC_IO.ClearDICNT_MF(hPort, iSlot, iChannel, iDI_TotalCh);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.

3.7.13. pac_ReadDICNTAll

This function reads the counts of all DI channels of the DI module.

Syntax

C++ - pac_ReadDICNTAll

```
BOOL pac_ReadDICNTAll(
    HANDLE hPort,
    int slot,
    int iDI_TotalCh,
    DWORD    lCounter_Value[]
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iDI_TotalCh

[in] Total number of the DI channels of the DI module.

lCounter_Value[]

[out] The array pointer to the counter value.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C] pac_ReadDICNTAll

Example 1：

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
int iDI_TotalCh = 8;
DWORD lCounter_Value[8];
BOOL IRET = pac_ReadDICNTAll(hPort, iSlot, iDI_TotalCh, lCounter_Value);
uart_Close(hPort);
```

[C#] pac_ReadDICNTAll

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
int iDI_TotalCh = 8;
UINT[] lCounter_Value=new UINT[8];
boolIRET = PACNET.PAC_IO.ReadDICNTAll(hPort, iSlot, iDI_TotalCh, lCounter_Value);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.

3.7.14. pac_ClearDICNTAll

This function clears the counter value of all DI channels of the DI module.

Syntax

C++ -pac_ClearDICNTAll

```
BOOL pac_ClearDICNT(
    HANDLE hPort,
    int slot,
    int iDI_TotalCh
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iDI_TotalCh

[in] Total number of the DI channels of the DI module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C] pac_ClearDICNTAll

Example 1：

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iDI_TotalCh = 8;
BOOL IRET = pac_ClearDICNT(hPort, iSlot, iDI_TotalCh);
uart_Close(hPort);
```

[C#] pac_ClearDICNTAll

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iDI_TotalCh = 8;
boolIRET = PACNET.PAC_IO.ClearDICNT(hPort, iSlot, iDI_TotalCh);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.

3.7.15. pac_WriteAO/pac_WriteAO_MF

This function writes the AO value to the AO modules.

Syntax

C++ - pac_WriteAO

```
BOOL pac_WriteAO(
    HANDLE hPort,
    int slot,
    int iChannel,
    int iAO_TotalCh,
    float fValue
);
```

C++ - pac_WriteAO_MF

```
BOOL pac_WriteAO_MF(
    HANDLE hPort,
    int slot,
    int iChannel,
    int iAO_TotalCh,
    float fValue
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] The channel that is written thee AO value to.

iAO_TotalCh

[in] The total number of the AO channels of the AO module.

float fValue

[in] The AO value to write to the AO module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C] pac_WriteAO

Example 1：

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
int iAO_TotalCh = 8;
float fValue= 5;
BOOL IRET = pac_WriteAO(hPort, iSlot, iChannel, iAO_TotalCh, fValue);
uart_Close(hPort);
```

Example 2：

```
// If the module is 8k local
BYTE iSlot = 1;
int iChannel = 2;
int iAO_TotalCh = 8;
float fValue= 5;
BOOL IRET = pac_WriteAO(0, iSlot, iChannel, iAO_TotalCh, fValue);
```

[C#] pac_WriteAO

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
int iAO_TotalCh = 8;
float fValue= 5;
boolIRET = PACNET.PAC_IO.WriteAO(hPort, iSlot, iChannel, iAO_TotalCh, fValue);
PACNET.UART.Close(hPort);
```

Remarks

1. The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO (0...255), which range is from 0 to 255.
2. The comparison table of pac_WriteAO / pac_WriteAO_MF Functions and available modules are as following:

pac_Write AO	pac_WriteAO_MF
I-87024W/CW/DW/RW, I-87024	I-87026PW
I-87028CW/UW	-
I-87022	-
I-87026	-
I-7021, I-7021P	-
I-7022	-
I-7024, I-8024R	-

3.7.16. pac_ReadAO

This function reads the AO value of the AO module.

Syntax

C++

```
BOOL pac_ReadAO(
    HANDLE hPort,
    int slot,
    int iChannel,
    int iAO_TotalCh,
    float* fValue
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] Read the AO value from the channel.

iAO_TotalCh

[in] The total number of the AO channels of the AO module.

float fValue

[in] The pointer to the AO value that is read back from the AO module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

Example 1：

```
HANDLE hPort; // If the module is 87k local
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
int iAO_TotalCh = 8;
float fValue;
BOOL IRET = pac_ReadAO(hPort, iSlot, iChannel, iAO_TotalCh,  &fValue);
uart_Close(hPort);
```

Example 2：

```
BYTE iSlot = 1; // If the module is 8k local
int iChannel = 2;
int iAO_TotalCh = 8;
float fValue;
BOOL IRET = pac_ReadAO(0, iSlot, iChannel, iAO_TotalCh,  &fValue);
```

[C#]

```
IntPtr hPort; // If the module is 87k local
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
int iAO_TotalCh = 8;
float fValue;
boolIRET = PACNET.PAC_IO.ReadAO(hPort, iSlot, iChannel, iAO_TotalCh, ref fValue);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO (0...255), which range is from 0 to 255.

3.7.17. pac_ReadAI

This function reads the engineering-mode AI value of the AI module.

Syntax

C++

```
BOOL pac_ReadAI(
    HANDLE hPort,
    int slot,
    int iChannel,
    int iAI_TotalCh,
    float* fValue
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] Read the AI value from the channel.

iAI_TotalCh

[in] The total number of the AI channels of the AI module.

fValue

[in] The pointer to the AI value that is read back from the AI module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

Example 1：

```
HANDLE hPort; // If the module is 87k local
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
int iAI_TotalCh = 8;
float fValue;
BOOL IRET = pac_ReadAI(hPort, iSlot, iChannel, iAI_TotalCh,  &fValue);
uart_Close(hPort);
```

Example 2：

```
BYTE iSlot = 1; // If the module is 8k local
int iChannel = 2;
int iAI_TotalCh = 8;
float fValue;
BOOL IRET = pac_ReadAI(0, iSlot, iChannel, iAI_TotalCh,  &fValue);
```

[C#]

```
IntPtr hPort; // If the module is 87k local
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
int iAI_TotalCh = 8;
float fValue;
boolIRET = PACNET.PAC_IO.ReadAI(hPort, iSlot, iChannel, iAI_TotalCh, ref fValue);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO (0...255), which range is from 0 to 255.

3.7.18. pac_ReadAIHex

This function reads the 2's complement-mode AI value of the AI module.

Syntax

C++

```
BOOL pac_ReadAIHex(
    HANDLE hPort,
    int slot,
    int iChannel,
    int iAI_TotalCh,
    int * iValue
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] Read the AI value from the channel.

iAI_TotalCh

[in] The total number of the AI channels of the AI module.

iValue

[in] The pointer to the AI value that is read back from the AI module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

Example 1：

```
HANDLE hPort; // If the module is 87k local
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
int iAI_TotalCh = 8;
INT iValue;
BOOL IRET = pac_ReadAIHex(hPort, iSlot, iChannel, iAI_TotalCh,  &iValue);
uart_Close(hPort);
```

Example 2：

```
BYTE iSlot = 1; // If the module is 8k local
int iChannel = 2;
int iAI_TotalCh = 8;
INT iValue;
BOOL IRET = pac_ReadAIHex(0, iSlot, iChannel, iAI_TotalCh,  &iValue);
```

[C#]

```
IntPtr hPort; // If the module is 87k local
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
int iAI_TotalCh = 8;
INT iValue;
boolIRET = PACNET.PAC_IO.ReadAIHex(hPort, iSlot, iChannel, iAI_TotalCh, ref iValue);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO (0...255), which range is from 0 to 255.

3.7.19. pac_ReadAIAllExt

This function reads all the AI values of all channels in engineering-mode of the AI module.

This function replaces pac_ReadAIAll.

Syntax

C++

```
BOOL pac_ReadAIAllExt(
    HANDLE hPort,
    int slot,
    float fValue[] ,
    DWORD Buff_Len,
    DWORD *Channel
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

fValue[]

[out] The array contains the AI values that read back from the AI module.

Buff_Len

[in] A pointer to a variable that specifies the size of the buffer pointed to by the fvalue.

Channel

[out] The pointer to a variable that specifies the total available channels number of AI module.

This total channels number is only valid if the return value is TRUE.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

Example 1：

```
HANDLE hPort; // If the module is 87k local
int ichannelnumber = 0;
hPort = uart_Open("");
BYTE iSlot = 1;
float fValue[8] ;
BOOL IRET = pac_ReadAIAllExt(hPort, iSlot, fValue, 8,  &ichannelnumber);
uart_Close(hPort);
```

Example 2：

```
BYTE iSlot = 1; // If the module is 8k local
int ichannelnumber = 0;
float fValue[8] ;
BOOL IRET = pac_ ReadAIAllExt(0, iSlot, fValue, 8,  &ichannelnumber);
```

[C#]

```
IntPtr hPort; // If the module is 87k local
int channelnumber = 0;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
float fValue[8] ;
boolIRET = PACNET.PAC_IO.ReadAIAllExt(hPort, iSlot, fValue, 8, ref channelnumber);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO (0...255), which range is from 0 to 255.

3.7.20. pac_ReadAIAll

This function reads all the AI values of all channels in engineering-mode of the AI module.

The function maybe causes the buffer overflow in some situation.

Syntax

C++

```
BOOL pac_ReadAIAll(
    HANDLE hPort,
    int slot,
    float fValue[]
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

fValue[]

[out] The array contains the AI values that read back from the AI module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

Example 1：

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
float fValue[8] ;
BOOL IRET = pac_ReadAIAll(hPort, iSlot, fValue);
uart_Close(hPort);
```

Example 2：

```
// If the module is 8k local
BYTE iSlot = 1;
float fValue[8] ;
BOOL IRET = pac_ReadAIAll(0, iSlot, fValue);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
float fValue[8] ;
boolIRET = PACNET.PAC_IO.ReadAIAll(hPort, iSlot, fValue);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO (0...255), which range is from 0 to 255.

3.7.21. pac_ReadAIAllHexExt

This function reads all the AI values of all channels in 2's complement-mode of the AI module.

This function replaces pac_ReadAIAllHex.

Syntax

C++

```
BOOL pac_ReadAIAllHexExt(
    HANDLE hPort,
    int slot,
    INT iValue [] ,
    DWORD Buff_Len,
    DWORD *Channel
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO (0...255).

iValue[]

[out] The array contains the AI values that read back from the AI module.

Buff_Len

[in] A pointer to a variable that specifies the size of the buffer pointed to by the iValue.

Channel

[out] The pointer to a variable that specifies the total available channels number of AI module.

This total channels number is only valid if the return value is TRUE.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

Example 1：

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iValue [8] ;
int ichannelnumber = 0;
BOOL IRET = pac_ReadAIAllHexExt(hPort, iSlot, iValue, 8,  &ichannelnumber);
uart_Close(hPort);
```

Example 2：

```
// If the module is 8k local
BYTE iSlot = 1;
int ichannelnumber = 0;
int iValue [8] ;
BOOL IRET = pac_ReadAIAllHexExt(0, iSlot, iValue, 8,  &ichannelnumber);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int ichannelnumber = 0;
int iValue [8] ;
boolIRET = PACNET.PAC_IO.ReadAIAllHexExt(hPort, iSlot, iValue, 8, ref ichannelnumber);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO (0...255), which range is from 0 to 255.

3.7.22. pac_ReadAIAllHex

This function reads all the AI values of all channels in 2's complement-mode of the AI module.

The function maybe causes the buffer overflow in some situation.

Syntax

C++

```
BOOL pac_ReadAIAllHex(
    HANDLE hPort,
    int slot,
    int iValue []
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO (0...255).

iValue[]

[out] The array contains the AI values that read back from the AI module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

Example 1：

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iValue [8] ;
BOOL IRET = pac_ReadAIAllHex(hPort, iSlot, iValue);
uart_Close(hPort);
```

Example 2：

```
// If the module is 8k local
BYTE iSlot = 1;
int iValue [8] ;
BOOL IRET = pac_ReadAIAllHex(0, iSlot, iValue);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iValue [8] ;
boolIRET = PACNET.PAC_IO.ReadAIAllHex(hPort, iSlot, iValue);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO (0...255), which range is from 0 to 255.

3.7.23. pac_ReadCNT

This function reads the counter values of the counter/frequency modules.

Syntax

C++

```
BOOL pac_ReadCNT(
    HANDLE hPort,
    int slot,
    int iChannel,
    DWORD * lCounter_Value
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] The channel that reads the counter value back from the counter/frequency module.

lCounter_Value

[out] The pointer to the counter value that reads back from the counter/frequency module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

Example 1：

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 0;
DWORD lCounter_Value;
BOOL IRET = pac_ReadCNT(hPort, iSlot, iChannel, &lCounter_Value);
uart_Close(hPort);
```

Example 2：

```
// If the module is 8k local
BYTE iSlot = 1;
int iChannel = 0;
DWORD lCounter_Value;
BOOL IRET = pac_ReadCNT(0, iSlot, iChannel,  &lCounter_Value);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 0;
UINT lCounter_Value;
boolIRET = PACNET.PAC_IO.ReadCNT(hPort, iSlot, iChannel, ref lCounter_Value);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO (0...255), which range is from 0 to 255.

3.7.24. pac_ClearCNT

This function clears the counter values of the counter/frequency modules.

Syntax

C++

```
BOOL pac_ClearCNT(
    HANDLE hPort,
    int slot,
    int iChannel
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] The channel that clears the counter value back from the counter/frequency modules.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

Example 1：

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 0;
BOOL IRET = pac_ClearCNT(hPort, iSlot, iChannel);
uart_Close(hPort);
```

Example 2：

```
// If the module is 8k local
BYTE iSlot = 1;
int iChannel = 0;
BOOL IRET = pac_ClearCNT(0, iSlot, iChannel);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 0;
boolIRET = PACNET.PAC_IO.ClearCNT(hPort, iSlot, iChannel);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO (0...255), which range is from 0 to 255.

3.7.25. pac_ReadCNTOverflow

This function clears the counter overflow value of the counter/frequency modules.

Syntax

C++

```
BOOL pac_ReadCNTOverflow(
    HANDLE hPort,
    int slot,
    int iChannel,
    int * iOverflow
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

0, if the module is 8k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO (0...255).

iChannel

[in] The channel that reads the counter overflows value back from the counter/frequency module.

iOverflow

[out] The pointer to the counter overflow that is read back from the counter/frequency module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[eVC/VC/VS]

Example 1：

```
HANDLE hPort; // If the module is 87k local
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 0;
int iOverflow;
BOOL IRET = pac_ReadCNTOverflow(hPort, iSlot, iChannel, &iOverflow);
uart_Close(hPort);
```

Example 2：

```
BYTE iSlot = 1; // If the module is 8k local
int iChannel = 0;
int iOverflow;
BOOL IRET = pac_ReadCNTOverflow(0, iSlot, iChannel, &iOverflow);
```

[C#]

```
IntPtr hPort;       // If the module is 87k local
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 0;
int iOverflow;
boolIRET = PACNET.PAC_IO.ReadCNTOverflow(hPort, iSlot, iChannel, ref iOverflow);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO (0...255), which range is from 0 to 255.

3.7.26. pac_WriteModuleSafeValueDO/pac_WriteModuleSafeValueDO_MF

This function writes the DO safe values to DO modules.

Syntax

C++ - pac_WriteModuleSafeValueDO

```
BOOL pac_WriteModuleSafeValueDO(
    HANDLE hPort,
    int slot,
    int iDO_TotalCh,
    DWORD lValue
);
```

C++ - pac_WriteModuleSafeValueDO_MF

```
BOOL pac_WriteModuleSafeValueDO_MF(
    HANDLE hPort,
    int slot,
    int iDO_TotalCh,
    DWORD lValue
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iDO_TotalCh

[in] The total number of DO channels of the DO modules.

iValue

[in] A 8-digit hexadecimal value, where bit 0 corresponds to DO0, bit 31 corresponds to DO31, etc. When the bit is 1, it denotes that the digital output channel is on, and 0 denotes that the digital output channel is off.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C] pac_WriteModuleSafeValueDO

Example 1：

```
HANDLE hPort; // If the module is remote
hPort = uart_Open( "COM2, 9600, N, 8, 1" );
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //turn on the channel two
BOOL RET = pac_WriteModuleSafeValueDO(hPort, PAC_REMOTE_IO(1), TOTAL_CHANNEL,
do_value); uart_Close(hPort);
```

Example 2：

```
HANDLE hPort; // If the module is 87k local
hPort = uart_Open("");
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //turn on the channel two
BOOL RET = pac_WriteModuleSafeValueDO(hPort, 1, TOTAL_CHANNEL, do_value);
uart_Close(hPort);
```

[C] pac_WriteModuleSafeValueDO_MF

Example 1：

```
// If the module is remote
HANDLE hPort;
hPort = uart_Open( "COM2, 9600, N, 8, 1" );
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //turn on the channel two
BOOL RET = pac_WriteModuleSafeValueDO_MF(hPort, PAC_REMOTE_IO(1),
TOTAL_CHANNEL, do_value); uart_Close(hPort);
```

Example 2：

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //turn on the channel two
BOOL RET = pac_WriteModuleSafeValueDO_MF(hPort, 1, TOTAL_CHANNEL, do_value);
uart_Close(hPort);
```

[C#] pac_WriteModuleSafeValueDO

Example 1：

```
// If the module is remote
IntPtr hPort; hPort = PACNET.UART.Open( "COM1, 9600, N, 8, 1" );
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //turn on the channel two
boolRET = PACNET.PAC_IO.pac_WriteModuleSafeValueDO(hPort,
PACNET.PAC_IO.REMOTE_IO(1), TOTAL_CHANNEL, do_value);
PACNET.UART.Close(hPort);
```

Example 2：

```
// If the module is 87k local
IntPtr hPort; hPort = PACNET.UART.Open("");
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //turn on the channel two
boolRET = PACNET.PAC_IO.pac_WriteModuleSafeValueDO(hPort, 1, TOTAL_CHANNEL,
do_value);
PACNET.UART.Close(hPort);
```

[C#] pac_WriteModuleSafeValueDO_MF

Example 1：

```
// If the module is remote
IntPtr hPort; hPort = PACNET.UART.Open( "COM1, 9600, N, 8, 1" );
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //turn on the channel two
boolRET = PACNET.PAC_IO.pac_WriteModuleSafeValueDO_MF(hPort,
PACNET.PAC_IO.REMOTE_IO(1), TOTAL_CHANNEL, do_value);
PACNET.UART.Close(hPort);
```

Example 2：

```
// If the module is 87k local
IntPtr hPort; hPort = PACNET.UART.Open("");
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //turn on the channel two
boolRET = PACNET.PAC_IO.pac_WriteModuleSafeValueDO_MF(hPort, 1, TOTAL_CHANNEL,
do_value);
PACNET.UART.Close(hPort);
```

Remarks

The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.

3.7.27. pac_ReadModuleSafeValueDO/pac_ReadModuleSafeValueDO_MF

This function reads the safe value of the DO modules.

Syntax

C++ - pac_ReadModuleSafeValueDO

```
BOOL pac_ReadModuleSafeValueDO(
    HANDLE hPort,
    int slot,
    int iDO_TotalCh,
    unsigned long* lValue
);
```

C++ - pac_ReadModuleSafeValueDO_MF

```
BOOL pac_ReadModuleSafeValueDO_MF(
    HANDLE hPort,
    int slot,
    int iDO_TotalCh,
    unsigned long* lValue
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

Slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO (0...255).

iChannel

[in] The total number of DO channels of the DO modules.

lValue

[in] The pointer of the DO safe value to read from the DO module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C] pac_ReadModuleSafeValueDO

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
byte slot= 1;
int TOTAL_CHANNEL = 8;
DWORD do_value;
BOOL RET = pac_ReadModuleSafeValueDO(hPort, slot, TOTAL_CHANNEL, &do_value);
uart_Close(hPort);
```

[C] pac_ReadModuleSafeValueDO_MF

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
byte slot= 1;
int TOTAL_CHANNEL = 8;
DWORD do_value;
BOOL RET = pac_ReadModuleSafeValueDO_MF(hPort, slot, TOTAL_CHANNEL, &do_value);
uart_Close(hPort);
```

[C#] pac_ReadModuleSafeValueDO

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte slot= 1;
int TOTAL_CHANNEL = 8;
UINT do_value;
boolRET = PACNET.PAC_IO.pac_ReadModuleSafeValueDO(hPort, slot, TOTAL_CHANNEL, ref do_value);
PACNET.UART.Close(hPort);
```

[C#] pac_ReadModuleSafeValueDO_MF

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte slot= 1;
int TOTAL_CHANNEL = 8;
UINT do_value;
boolRET = PACNET.PAC_IO.pac_ReadModuleSafeValueDO_MF(hPort, slot, TOTAL_CHANNEL, ref do_value);
PACNET.UART.Close(hPort);
```

Remarks

1. The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.
2. I-7K/I-87K series modules with power-on or safety value function can support this "API" function. I-8K series modules only provide this function for I-8041RW, and I-9K series DO modules support this function.

3.7.28. pac_WriteModulePowerOnValueDO/pac_WriteModulePowerOnValueDO_MF

This function writes the DO power on values to DO modules.

Syntax

C++ - pac_WriteModulePowerOnValueDO

```
BOOL pac_WriteModulePowerOnValueDO(
    HANDLE hPort,
    int slot,
    int iDO_TotalCh,
    DWORD lValue
);
```

C++ - pac_WriteModulePowerOnValueDO_MF

```
BOOL pac_WriteModulePowerOnValueDO_MF(
    HANDLE hPort,
    int slot,
    int iDO_TotalCh,
    DWORD lValue
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iDO_TotalCh

[in] The total number of DO channels of the DO modules.

iValue

[in] A 8-digit hexadecimal value, where bit 0 corresponds to DO0, bit 31 corresponds to DO31, etc. When the bit is 1, it denotes that the digital output channel is on, and 0 denotes that the digital output channel is off.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C] pac_WriteModulePowerOnValueDO

Example 1：

```
// If the module is remote
HANDLE hPort;
hPort = uart_Open( "COM1, 9600, N, 8, 1" );
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //turn on the channel two
BOOL RET = pac_WriteModulePowerOnValueDO(hPort, PAC_REMOTE_IO(1),
TOTAL_CHANNEL, do_value); uart_Close(hPort);
```

Example 2：

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //turn on the channel two
BOOL RET = pac_WriteModulePowerOnValueDO(hPort, 1, TOTAL_CHANNEL, do_value);
uart_Close(hPort);
```

[C] pac_WriteModulePowerOnValueDO_MF

Example 1：

```
// If the module is remote
HANDLE hPort;
hPort = uart_Open( “COM1, 9600, N, 8, 1” );
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //turn on the channel two
BOOL RET = pac_WriteModulePowerOnValueDO_MF(hPort, PAC_REMOTE_IO(1),
TOTAL_CHANNEL, do_value); uart_Close(hPort);
```

Example 2：

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open(“”);
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //turn on the channel two
BOOL RET = pac_WriteModulePowerOnValueDO(hPort, 1, TOTAL_CHANNEL, do_value);
uart_Close(hPort);
```

[C#] pac_WriteModulePowerOnValueDO

Example 1：

```
// If the module is remote
IntPtr 的 hPort = PACNET.UART.Open( “COM1, 9600, N, 8, 1” );
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //turn on the channel two
boolRET = PACNET.PAC_IO.pac_WriteModulePowerOnValueDO(hPort,
PACNET.PAC_IO.REMOTE_IO(1), TOTAL_CHANNEL, do_value);
PACNET.UART.Close(hPort);
```

Example 2：

```
// If the module is 87k local
IntPtr 的 hPort = PACNET.UART.Open( “” );
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //turn on the channel two
boolRET = PACNET.PAC_IO.pac_WriteModulePowerOnValueDO(hPort, 1, TOTAL_CHANNEL,
do_value);
PACNET.UART.Close(hPort);
```

[C#] pac_WriteModulePowerOnValueDO_MF

Example 1：

```
// If the module is remote
IntPtr 的 hPort = PACNET.UART.Open( “COM1, 9600, N, 8, 1” );
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //turn on the channel two
boolRET = PACNET.PAC_IO.pac_WriteModulePowerOnValueDO_MF(hPort,
PACNET.PAC_IO.REMOTE_IO(1), TOTAL_CHANNEL, do_value);
PACNET.UART.Close(hPort);
```

Example 2：

```
// If the module is 87k local
IntPtr 的 hPort = PACNET.UART.Open( “” );
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //turn on the channel two
boolRET = PACNET.PAC_IO.pac_WriteModulePowerOnValueDO_MF(hPort, 1,
TOTAL_CHANNEL, do_value);
PACNET.UART.Close(hPort);
```

Remarks

1. The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.
2. I-7K/I-87K series modules with power-on or safety value function can support this "*API*" function. I-8K series modules only provide this function for I-8041RW, and I-9K series DO modules support this function.

3.7.29. pac_ReadModulePowerOnValueDO/pac_ReadModulePowerOnValueDO_MF

This function reads the power on value of the DO modules.

Syntax

C++ - pac_ReadModulePowerOnValueDO

```
BOOL pac_ReadModulePowerOnValueDO(
    HANDLE hPort,
    int slot,
    int iDO_TotalCh,
    unsigned long* lValue
);
```

C++ - pac_ReadModulePowerOnValueDO_MF

```
BOOL pac_ReadModulePowerOnValueDO_MF(
    HANDLE hPort,
    int slot,
    int iDO_TotalCh,
    unsigned long* lValue
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

Slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO (0...255).

iChannel

[in] The total number of DO channels of the DO modules.

lValue

[in] The pointer of the DO power on value to read from the DO module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C] pac_ReadModulePowerOnValueDO

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
byte slot= 1;
int TOTAL_CHANNEL = 8;
DWORD do_value;
BOOL RET = pac_ReadModulePowerOnValueDO(hPort, slot, TOTAL_CHANNEL, &do_value);
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte slot= 1;
int TOTAL_CHANNEL = 8;
UINT do_value;
boolRET = PACNET.PAC_IO.pac_ReadModulePowerOnValueDO(hPort, slot, TOTAL_CHANNEL,
ref do_value);
PACNET.UART.Close(hPort);
```

[C] pac_ReadModulePowerOnValueDO_MF

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
byte slot= 1;
int TOTAL_CHANNEL = 8;
DWORD do_value;
BOOL RET = pac_ReadModulePowerOnValueDO_MF(hPort, slot, TOTAL_CHANNEL,
&do_value);
uart_Close(hPort);
```

[C#] pac_ReadModulePowerOnValueDO_MF

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte slot= 1;
int TOTAL_CHANNEL = 8;
UINT do_value;
boolRET = PACNET.PAC_IO.pac_ReadModulePowerOnValueDO_MF(hPort, slot,
TOTAL_CHANNEL, ref do_value);
PACNET.UART.Close(hPort);
```

Remarks

1. The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.
2. I-7K/I-87K series modules with power-on or safety value function can support this "*API*" function. I-8K series modules only provide this function for I-8041RW, and I-9K series DO modules support this function.

3.7.30. pac_WriteModuleSafeValueAO

This function writes the AO safe value to the AO modules.

Syntax

C++

```
BOOL pac_WriteModuleSafeValueAO(
    HANDLE hPort,
    int slot,
    int iChannel,
    int iAO_TotalCh,
    float fValue
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] The channel that is written thee AO value to.

iAO_TotalCh

[in] The total number of the AO channels of the AO module.

float fValue

[in] The AO value to write to the AO module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
int iAO_TotalCh = 8;
float fValue= 5;
BOOL IRET = pac_WriteModuleSafeValueAO(hPort, iSlot, iChannel, iAO_TotalCh, fValue);
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
int iAO_TotalCh = 8;
float fValue= 5;
boolIRET = PACNET.PAC_IO.WriteModuleSafeValueAO(hPort, iSlot, iChannel, iAO_TotalCh,
fValue);
PACNET.UART.Close(hPort);
```

Remarks

1. The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.
2. I-7K/I-87K series modules with power-on or safety value function can support this "*API*" function. I-8K series modules only provide this function for I-8041RW, and I-9K series DO modules support this function.

3.7.31. pac_ReadModuleSafeValueAO

This function reads the AO safe value of the AO module.

Syntax

C++

```
BOOL pac_ReadModuleSafeValueAO(
    HANDLE hPort,
    int slot,
    int iChannel,
    int iAO_TotalCh,
    float* fValue
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] Read the AO value from the channel.

iAO_TotalCh

[in] The total number of the AO channels of the AO module.

float fValue

[in] The pointer to the AO safe value that is read back from the AO module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
int iAO_TotalCh = 8;
float fValue;
BOOL IRET = pac_ReadModuleSafeValueAO(hPort, iSlot, iChannel, iAO_TotalCh, &fValue);
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
int iAO_TotalCh = 8;
float fValue;
boolIRET = PACNET.PAC_IO.ReadModuleSafeValueAO(hPort, iSlot, iChannel, iAO_TotalCh, ref fValue);
PACNET.UART.Close(hPort);
```

Remarks

1. The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.
2. I-7K/I-87K series modules with power-on or safety value function can support this "*API*" function. I-8K series modules only provide this function for I-8041RW, and I-9K series DO modules support this function.

3.7.32. pac_WriteModulePowerOnValueAO

This function writes the AO power on value to the AO modules.

Syntax

C++

```
BOOL pac_WriteModulePowerOnValueAO(
    HANDLE hPort,
    int slot,
    int iChannel,
    int iAO_TotalCh,
    float fValue
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] The channel that is written thee AO value to.

iAO_TotalCh

[in] The total number of the AO channels of the AO module.

float fValue

[in] The AO value to write to the AO module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
int iAO_TotalCh = 8;
float fValue= 5;
BOOL IRET = pac_WriteModulePowerOnValueAO(hPort, iSlot, iChannel, iAO_TotalCh, fValue);
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
int iAO_TotalCh = 8;
float fValue= 5;
boolIRET = PACNET.PAC_IO.WriteModulePowerOnValueAO(hPort, iSlot, iChannel,
iAO_TotalCh, fValue);
PACNET.UART.Close(hPort);
```

Remarks

1. The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.
2. I-7K/I-87K series modules with power-on or safety value function can support this "*API*" function. I-8K series modules only provide this function for I-8041RW, and I-9K series DO modules support this function.

3.7.33. pac_ReadModulePowerOnValueAO

This function reads the AO power on value of the AO module.

Syntax

C++

```
BOOL pac_ReadModulePowerOnValueAO(
    HANDLE hPort,
    int slot,
    int iChannel,
    int iAO_TotalCh,
    float* fValue
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

iSlot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] Read the AO value from the channel.

iAO_TotalCh

[in] The total number of the AO channels of the AO module.

float fValue

[in] The pointer to the AO power on value that is read back from the AO module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
int iAO_TotalCh = 8;
float fValue;
BOOL IRET = pac_ReadModulePowerOnValueAO(hPort, iSlot, iChannel, iAO_TotalCh, &
fValue);
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
int iAO_TotalCh = 8;
float fValue;
boolIRET = PACNET.PAC_IO.ReadModulePowerOnValueAO(hPort, iSlot, iChannel,
iAO_TotalCh, ref fValue);
PACNET.UART.Close(hPort);
```

Remarks

1. The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.
2. I-7K/I-87K series modules with power-on or safety value function can support this "*API*" function. I-8K series modules only provide this function for I-8041RW, and I-9K series DO modules support this function.

3.7.34. pac_GetModuleLastOutputSource

Reads the last output source of a module.

Syntax

C++

```
short pac_GetModuleLastOutputSource(
    HANDLE hPort,
    int slot
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

回傳值

For I-8K(I-8kRw)

0：no action

1：power on value

2：Safe value

3：DO command

For I-87K/I-7K

0：Other(power on value or DO command)

2：Safe value

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
int iSlot = 0;
int lastOutput = 0;
hPort = uart_Open("");
lastOutput = pac_GetModuleLastOutputSource(hPort, iSlot);
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
int iSlot = 0;
hPort = PACNET.UART.Open("");
int lastOutput = PACNET.PAC_IO.GetModuleLastOutputSource(hPort, iSlot);
PACNET.UART.Close(hPort);
```

Remarks

1. The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.
2. I-7K/I-87K series modules with power-on or safety value function can support this "*API*" function. I-8K series modules only provide this function for I-8041RW, and I-9K series DO modules support this function.

3.7.35. pac_GetModuleWDTStatus

This function reads the status of watchdog on the module.

Syntax

C++

```
bool pac_GetModuleWDTStatus(
    HANDLE hPort,
    int slot
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

Return Value

If the return value is TRUE, it means watchdog timeout occurred

If the return value is FALSE, it means watchdog timeout doesn't occur.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
int iSlot = 0;
boolBSTATUS = 0;
hPort = uart_Open(“”);
BSTATUS = pac_GetModuleWDTStatus(hPort, iSlot);
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
int iSlot = 0;
hPort = PACNET.UART.Open(“”);
bool BSTATUS = PACNET.PAC_IO.GetModuleWDTStatus(hPort, iSlot);
PACNET.UART.Close(hPort);
```

Remarks

1. The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.
2. I-7K/I-87K series modules with power-on or safety value function can support this "*API*" function. I-8K series modules only provide this function for I-8041RW, and I-9K series DO modules support this function.

3.7.36. pac_GetModuleWDTConfig

This function reads the host watchdog config of a module.

Syntax

C++

```
bool pac_GetModuleWDTConfig(
    HANDLE hPort,
    int slot,
    short* enStatus,
    unsigned long* wdtTimeout,
    int * ifWDT_Overwrite
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

enStatus

[out] 1: the host watchdog is enabled

0: the host watchdog is disabled

wdtTimeout

[out] The unit of return value is 100ms.

ifWDT_Overwrite (only for i-8k)

[out] 1: the host watchdog does overwrite

0: the host watchdog does not overwrite

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
int iSlot = 0;
short sStatus = 0;
unsigned longulWDTtime = 0;
int iOverwrite = 0;
hPort = uart_Open("");
pac_GetModuleWDTConfig(hPort, iSlot, &sStatus, &ulWDTtime, &iOverwrite);
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
int iSlot = 0;
short sStatus = 0;
unsigned longulWDTtime = 0;
int iOverwrite = 0;
hPort = PACNET.UART.Open("");
PACNET.PAC_IO.GetModuleWDTConfig(hPort, iSlot, ref sStatus, ref ulWDTtime, ref
iOverwrite);
PACNET.UART.Close(hPort);
```

Remarks

1. The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.
2. I-7K/I-87K series modules with power-on or safety value function can support this "*API*" function. I-8K series modules only provide this function for I-8041RW, and I-9K series DO modules support this function.

3.7.37. pac_SetModuleWDTConfig

This function enables/disables the host watchdog and sets the host watchdog timeout value of a module.

Syntax

C++

```
bool pac_SetModuleWDTStatus(
    HANDLE hPort,
    int slot,
    short enStatus,
    unsigned longwdtTimeout,
    int ifWDT_Overwrite
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

enStatus

[in] 1: the host watchdog is enabled

0: the host watchdog is disabled

longwdtTimeout

[in] The unit of return value is 100ms.

ifWDT_Overwrite (only for i-8k)

[in] 1: the host watchdog does overwrite

0: the host watchdog does not overwrite

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
int iSlot = 0;
short sStatus = 0;
unsigned longulWDTtime = 0;
int iOverwrite = 0;
hPort = uart_Open("");
pac_SetModuleWDTConfig(hPort, iSlot, sStatus, ulWDTtime, iOverwrite);
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
int iSlot = 0;
short sStatus = 0;
unsigned longulWDTtime = 0;
int iOverwrite = 0;
hPort = PACNET.UART.Open("");
PACNET.PAC_IO.SetModuleWDTConfig(hPort, iSlot, sStatus, ulWDTtime, iOverwrite);
PACNET.UART.Close(hPort);
```

Remarks

1. The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.
2. I-7K/I-87K series modules with power-on or safety value function can support this "*API*" function. I-8K series modules only provide this function for I-8041RW, and I-9K series DO modules support this function.

3.7.38. pac_ResetModuleWDT

This function resets the host watchdog timeout status of a module.

Syntax

C++

```
bool pac_ResetModuleWDT(
    HANDLE hPort,
    int slot
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
int iSlot = 0;
hPort = uart_Open("");
pac_ResetModuleWDT(hPort, iSlot);
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
int iSlot = 0;
hPort = PACNET.UART.Open("");
PACNET.PAC_IO.ResetModuleWDT(hPort, iSlot);
PACNET.UART.Close(hPort);
```

Remarks

1. The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.
2. I-7K/I-87K series modules with power-on or safety value function can support this "*API*" function. I-8K series modules only provide this function for I-8041RW, and I-9K series DO modules support this function.

3.7.39. pac_RefreshModuleWDT

This function refresh the host watchdog of a module.

Syntax

C++

```
bool pac_RefreshModuleWDT(
    HANDLE hPort,
    int slot
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
int iSlot = 0;
hPort = uart_Open("");
pac_RefreshModuleWDT(hPort, iSlot);
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
int iSlot = 0;
hPort = PACNET.UART.Open("");
PACNET.PAC_IO.RefreshModuleWDT(hPort, iSlot);
PACNET.UART.Close(hPort);
```

Remarks

1. The function can support for Local or Remote. When the module is local, the second Parameter's range is from 0 to 7. If remote, the second Parameter need use the macro, PAC_REMOTE_IO(0...255), which range is from 0 to 255.
2. I-7K/I-87K series modules with power-on or safety value function can support this "*API*" function. I-8K series modules only provide this function for I-8041RW, and I-9K series DO modules support this function.

3.7.40. pac_InitModuleWDTInterrupt

This function initializes and enables interrupt of a module watchdog.

Syntax

C++

```
bool pac_RefreshModuleWDT(
    int slot,
    PAC_CALLBACK_FUNC f
);
```

Parameters

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

f

[in] A call back function..

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
int CALLBACK slot_callback_proc()
{
// do something
   return true;
}
int iSlot = 0;
pac_InitModuleWDTInterrupt(iSlot, slot_callback_proc);
```

[C#]

```
PACNET.CALLBACK_FUNC slot_callback_proc; //global
int slot_callback_proc()
{
// do something
   return 0;
}
int iSlot = 0;
PACNET.PAC_IO.InitModuleWDTInterrupt(iSlot, slot_callback_proc);
```

Remarks

I-8K series modules with power-on or safety value function can support this "*API*" function. I-8K series modules only provide this function for I-8041RW, and I-9K series DO modules support this function.

3.7.41. pac_GetModuleWDTInterruptStatus

This function reads interrupt status of a module watchdog.

Syntax

C++

```
short pac_GetModuleWDTInterruptStatus(
    int slot
);
```

Parameters

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

Return Value

Interrupt status.

Examples

[C]

```
int iSlot = 0;
short sStatus = 0;
sStatus = pac_GetModuleWDTInterruptStatus(iSlot);
```

[C#]

```
int iSlot = 0;
short sStatus = 0;
sStatus = PACNET.PAC_IO.GetModuleWDTInterruptStatus(iSlot);
```

Remarks

I-8K series modules with power-on or safety value function can support this "*API*" function. I-8K series modules only provide this function for I-8041RW, and I-9K series DO modules support this function.

3.7.42. pac_SetModuleWDTInterruptStatus

This function enables/disables interrupt of a module watchdog.

Syntax

C++

```
bool pac_SetModuleWDTInterruptStatus(
    int slot,
    short enStatus
);
```

Parameters

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

enStatus

[in] Interrupt status.

1: enable

0: disable

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
int iSlot = 0;
short sStatus = 0;
pac_SetModuleWDTInterruptStatus(iSlot, sStatus);
```

[C#]

```
int iSlot = 0;
short sStatus = 0;
PACNET.PAC_IO.SetModuleWDTInterruptStatus(iSlot, sStatus);
```

Remarks

I-8K series modules with power-on or safety value function can support this "*API*" function. I-8K series modules only provide this function for I-8041RW, and I-9K series DO modules support this function.

3.8. PWM API (Pulse-width modulation module control)

PWM API only supports to operate I-7K/I-87K PWM modules.

Before using the PWM API functions, refer to the previous chapter, PAC_IO Reference first for more details regarding of the slot definition in local and how to use remote I/O module.

In developing C/C++ program for I-7K/I-87K PWM modules connected or plugged to/on the the WinPAC/XPAC series device, in addition to link PACSDK.lib and it needs to link PACSDK_PWM.lib to the user's project.

Besides, the built executable file placed in the WinPAC/XPAC series device must work with PACSDK.dll and PACSDK_PWM.dll.

In developing .net CF program, the project only refer to PACNET.dll and the built executable file placed in the WinPAC series device only works with PACNET.dll and PACSDK.dll.

For more information about I-7K/I-87K PWM modules that are compatible with the XPAC/WinPAC series, please refer to

I-87K series
https://www.icpdas.com/en/product/guide+Remote__I_O__Module__and__Unit+PAC__%EF%BC%86amp;__Local__I_O__Modules+I-8K_I-87K__Series__(High__Profile)#482
PWM module (such as **I-87088W** module)

I-7K series
https://www.icpdas.com/en/product/guide+Remote__I_O__Module__and__Unit+RS-485__I_O__Modules+I-7000#464
(such as **I-7088**)

The suit of the PWM API functions isn't applied to the I-8K PWM module.

Supported PACs

The table below lists the PWM functions that are supported by each PAC.

Functions \ Models	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_SetPWMDuty	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetPWMDuty	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SetPWMFrequency	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetPWMFrequency	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SetPWMMode	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetPWMMode	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SetPWMDITriggerConfig	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetPWMDITriggerConfig	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SetPWMStart	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SetPWMSynChannel	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetPWMSynChannel	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SyncPWMStart	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SavePWMConfig	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetPWMDIOStatus	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SetPWMPulseCount	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetPWMPulseCount	Y	Y	Y	Y	Y	Y	Y	Y	Y

PWM Functions

The following functions are used to retrieve or set the PWM.

PACSDK Functions	PACNET Functions	Description
pac_SetPWMDuty	PWM.SetPWMDuty	Sets the duty cycle value for a specified channel.
pac_GetPWMDuty	PWM.GetPWMDuty	Reads the duty cycle value for a specific channel.
pac_SetPWMFrequency	PWM.SetPWMFrequency	Sets the frequency value for a specific channel.
pac_GetPWMFrequency	PWM.GetPWMFrequency	Reads the frequency value for a specific channel.
pac_SetPWMMode	PWM.SetPWMMode	Sets the continuous mode for a specific channel.
pac_GetPWMMode	PWM.GetPWMMode	Reads the continuous mode for a specific channel.
pac_SetPWMDITriggerConfig	PWM.SetPWMDITriggerConfig	Sets the hardware trigger for a specific channel.
pac_GetPWMDITriggerConfig	PWM.GetPWMDITriggerConfig	Reads the hardware trigger for a specific channel.
pac_SetPWMStart	PWM.SetPWMStart	Sets the status of the PWM output port.
pac_SetPWMSynChannel	PWM.SetPWMSynChannel	Sets the PWM synchronization status for a specific channel.
pac_GetPWMSynChannel	PWM.GetPWMSynChannel	Reads the PWM synchronization status for a specific channel.
pac_SyncPWMStart	PWM.SyncPWMStart	Starts the PWM synchronization.
pac_SavePWMConfig	PWM.SavePWMConfig	Saves the PWM configuration.
pac_GetPWMDIOStatus	PWM.GetPWMDIOStatus	Reads the status of the PWM output port and the digital output/input port.
pac_SetPWMPulseCount	PWM.SetPWMPulseCount	Sets the PWM step value for a specific channel.
pac_GetPWMPulseCount	PWM.GetPWMPulseCount	Reads the PWM step value for a specific channel.

3.8.1. pac_SetPWMDuty

This function sets the duty cycle value for a specified channel.

Syntax

C++

```
bool pac_SetPWMDuty(
    HANDLE PORT,
    int slot,
    short chIndex,
    float duty
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

chIndex

[in] Set the duty cycle value from the channel.

duty

[in] The duty cycle value to write to the PWM module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FLASE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
float fValue= 1.23;
BOOL IRET = pac_SetPWMDuty(hPort , iSlot , iChannel , fValue);
uart_Close(hPort);
```

[C#]

```
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
float fValue= 1.23;
boolIRET = PACNET.PWM.SetPWMDuty(hPort , iSlot , iChannel , fValue);
PACNET.UART.Close(hPort);
```

3.8.2. pac_GetPWMDuty

This function reads the duty cycle value for a specific channel.

Syntax

C++

```
bool pac_SetPWMDuty(
    HANDLE PORT,
    int slot,
    short chIndex,
    float*duty
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

chIndex

[in] Get the duty cycle value from the channel.

duty

[out] The duty cycle value to read to the PWM module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FLASE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
float fValue;
BOOL IRET = pac_GetPWMDuty(hPort , iSlot , iChannel , &fValue);
uart_Close(hPort);
```

[C#]

```
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
float fValue;
boolIRET = PACNET.PWM.GetPWMDuty(hPort , iSlot , iChannel , ref fValue);
PACNET.UART.Close(hPort);
```

3.8.3. pac_SetPWMFrequency

This function reads the frequency value for a specific channel.

Syntax

C++

```
bool pac_SetPWMFrequency(
    HANDLE PORT,
    int slot,
    short chIndex,
    unsigned long freq
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

chIndex

[in] Set the duty cycle value from the channel.

freq

[in] The frequency value to read to the PWM module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FLASE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
unsigned longulfreq = 1;
BOOL IRET = pac_SetPWMFrequency(hPort , iSlot , iChannel , ulfreq);
uart_Close(hPort);
```

[C#]

```
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
ULONG ulfreq = 1;
boolIRET = PACNET.PWM.SetPWMFrequency(hPort , iSlot , iChannel , ulfreq);
PACNET.UART.Close(hPort);
```

3.8.4. pac_GetPWMFrequency

This function sets the frequency value for a specific channel.

Syntax

C++

```
bool pac_GetPWMFrequency(
    HANDLE PORT,
    int slot,
    short chIndex,
    unsigned long*freq
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

chIndex

[in] Set the duty cycle value from the channel.

freq

[in] The frequency value to set to the PWM module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FLASE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
unsigned longulfreq;
BOOL IRET = pac_GetPWMFrequency(hPort , iSlot , iChannel , &ulfreq);
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
ULONG ulfreq = 1;
boolIRET = PACNET.PWM.GetPWMFrequency(hPort , iSlot , iChannel , ref ulfreq);
PACNET.UART.Close(hPort);
```

3.8.5. pac_SetPWMMode

This function sets the continuous mode for a specific channel.

Syntax

C++

```
bool pac_SetPWMMode(
    HANDLE PORT,
    int slot,
    short chIndex,
    long mode
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

chIndex

[in] Set the duty cycle value from the channel.

mode

[in] 0: Disables the PWM continuous mode

1: Enables the PWM continuous mode

(If the PWM continuous mode is enabled, the step value for PWM will be automatically set to 1)

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FLASE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
long mode= 0;
BOOL IRET = pac_SetPWMMode(hPort, iSlot, iChannel, mode);
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
ULONG mode = 0;
boolIRET = PACNET.PWM.SetPWMMode(hPort, iSlot, iChannel, mode);
PACNET.UART.Close(hPort);
```

3.8.6. pac_GetPWMMode

This function reads the continuous mode for a specific channel.

Syntax

C++

```
bool pac_GetPWMMode(
    HANDLE PORT,
    int slot,
    short chIndex,
    long *mode
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

chIndex

[in] Set the duty cycle value from the channel.

mode

[out] 0: Disables the PWM continuous mode

1: Enables the PWM continuous mode

(If the PWM continuous mode is enabled, the step value for PWM will be automatically set to 1)

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FLASE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
long mode;
BOOL IRET = pac_GetPWMMode(hPort, iSlot, iChannel, &mode);
uart_Close(hPort);
```

[C#]

```
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
ulong mode;
boolIRET = PACNET.PWM.GetPWMMode(hPort, iSlot, iChannel, REF mode);
PACNET.UART.Close(hPort);
```

3.8.7. pac_SetPWMDITriggerConfig

This function sets the hardware trigger for a specific channel.

Syntax

C++

```
bool pac_SetPWMDITriggerConfig(
    HANDLE PORT,
    int slot,
    short chIndex,
    short config
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

chIndex

[in] Set the duty cycle value from the channel.

config

[in] 0: Disables the hardware trigger

1: Enables the trigger start

2: Enables the trigger stop

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FLASE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
short mode = 0;
BOOL IRET = pac_SetPWMDITriggerConfig(hPort, iSlot, iChannel, mode );
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
short mode = 0;
boolIRET = PACNET.PWM.SetPWMDITriggerConfig(hPort, iSlot, iChannel, mode );
PACNET.UART.Close(hPort);
```

3.8.8. pac_GetPWMDITriggerConfig

This function reads the hardware trigger for a specific channel.

Syntax

C++

```
bool pac_GetPWMDITriggerConfig(
    HANDLE PORT,
    int slot,
    short chIndex,
    short* config
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

chIndex

[in] Set the duty cycle value from the channel.

config

[out] 0: Disables the hardware trigger

1: Enables the trigger start

2: Enables the trigger stop

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FLASE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
short mode;
BOOL IRET = pac_GetPWMDITriggerConfig(hPort, iSlot, iChannel, &mode );
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
short mode;
boolIRET = PACNET.PWM.GetPWMDITriggerConfig(hPort, iSlot, iChannel, REF MODE);
PACNET.UART.Close(hPort);
```

3.8.9. pac_SetPWMStart

This function sets the status of the PWM output port.

Syntax

C++

```
bool pac_SetPWMStart(
    HANDLE PORT,
    int slot,
    short enStatus
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

enStatus

[in] bit 0 corresponds to PWM channel 0, and bit 1 corresponds to PWM channel 1, etc. When the bit is 0, it denotes that the PWM output port is off, and 1 denotes that the PWM output port is o

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FLASE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
short Status= 0x01;
BOOL IRET = pac_SetPWMStart(hPort, iSlot, Status);
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
short Status= 0x01;
boolIRET = PACNET.PWM.SetPWMStart(hPort, iSlot, Status);
PACNET.UART.Close(hPort);
```

3.8.10. pac_SetPWMSynChannel

This function sets the PWM synchronization status for a specific channel.

Syntax

C++

```
bool pac_SetPWMSynChannel(
    HANDLE PORT,
    int slot,
    short chIndex,
    short enStatus
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

chIndex

[in] Set the duty cycle value from the channel.

enStatus

[in] 0: Disables the PWM synchronization

1: Enables the PWM synchronization

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FLASE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
short mode= 0;
BOOL IRET = pac_SetPWMSynChannel(hPort, iSlot, iChannel, mode );
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
short mode= 0;
boolIRET = PACNET.PWM.SetPWMSynChannel(hPort, iSlot, iChannel, mode );
PACNET.UART.Close(hPort);
```

3.8.11. pac_GetPWMSynChannel

This function reads the PWM synchronization status for a specific channel.

Syntax

C++

```
bool pac_GetPWMSynChannel(
    HANDLE PORT,
    int slot,
    short chIndex,
    short* enStatus
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

chIndex

[in] Set the duty cycle value from the channel.

enStatus

[out] 0: Disables the PWM synchronization

1: Enables the PWM synchronization

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FLASE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
short mode;
BOOL IRET = pac_GetPWMSynChannel(hPort, iSlot, iChannel, &mode );
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
short mode;
boolIRET = PACNET.PWM.GetPWMSynChannel(hPort, iSlot, iChannel, REF MODE);
PACNET.UART.Close(hPort);
```

3.8.12. pac_SyncPWMStart

This function starts the PWM synchronization.

Syntax

C++

```
bool pac_SyncPWMStart(
    HANDLE PORT,
    int slot,
    short enStatus
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

enStatus

[in] 0: Stops the PWM synchronization

1: Starts the PWM synchronization

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FLASE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
short Status= 0;
BOOL IRET = pac_SyncPWMStart(hPort, iSlot, Status);
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
short Status= 0;
boolIRET = PACNET.PWM.SyncPWMStart(hPort, iSlot, Status);
PACNET.UART.Close(hPort);
```

3.8.13. pac_SavePWMConfig

This function saves the PWM configuration.

Syntax

C++

```
bool pac_SavePWMConfig(
    HANDLE PORT,
    int slot,
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FLASE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
BOOL IRET = pac_SavePWMConfig(hPort, iSlot);
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
boolIRET = PACNET.PWM.SavePWMConfig(hPort, iSlot);
PACNET.UART.Close(hPort);
```

3.8.14. pac_GetPWMDIOStatus

This function reads the status of the PWM output port and the digital outout/input port.

Syntax

C++

```
bool pac_GetPWMDIOStatus(
    HANDLE PORT,
    int slot,
    unsigned char pwmBitArr [] ,
    unsigned char diBitArr []
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

pwmBitArr

[out] where array[0] corresponds to PWM channel 0, and array[1] corresponds to PWM channel 1, etc. When the array is 0, it denotes that the PWM is inactive and 1 denotes that the PWM is active.

diBitArr

[out] where array[0] corresponds to DI/DO channel 0, and array[1] corresponds to DI/DO channel 1, etc. When the bit is 0, it denotes that the DI/DO is inactive and 1 denotes that the DI is active.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FLASE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
unsigned char PWM [32] ;
unsigned char di[32] ;
BOOL IRET = pac_GetPWMDIOStatus(hPort, iSlot, PWM, di);
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
byte [] PWM =new byte [32] ;
byte [] di=new byte [32] ;
boolIRET = PACNET.PWM.GetPWMDIOStatus(hPort, iSlot, ref PWM, ref di);
PACNET.UART.Close(hPort);
```

3.8.15. pac_SetPWMPulseCount

This function sets the PWM step value for a specific channel.

Syntax

C++

```
bool pac_SetPWMPulseCount(
    HANDLE PORT,
    int slot,
    short chIndex,
    LONG CNT
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

chIndex

[in] Set the duty cycle value from the channel.

cnt

[in] The PWM steps (0x0001 to 0xFFFF)

(When set to more than 1 step, the PWM continuous mode will be automatically set to disabled)

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FLASE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
long lCNT = 1;
BOOL IRET = pac_SetPWMPulseCount(hPort, iSlot, iChannel, LCNT);
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
long lCNT = 1;
boolIRET = PACNET.PWM.SetPWMPulseCount(hPort, iSlot, iChannel, LCNT);
PACNET.UART.Close(hPort);
```

3.8.16. pac_GetPWMPulseCount

This function reads the PWM step value for a specific channel.

Syntax

C++

```
bool pac_GetPWMPulseCount(
    HANDLE PORT,
    int slot,
    short chIndex,
    long* CNT
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 87k modules in local.

slot

[in] The slot in which module is to receive the command. Default is local.

If the I/O module is remote, please use the macro, PAC_REMOTE_IO(0...255).

chIndex

[in] Set the duty cycle value from the channel.

cnt

[out] The PWM steps (0x0001 to 0xFFFF)

(When set to more than 1 step, the PWM continuous mode will be automatically set to disabled)

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FLASE.

Examples

[C]

```
// If the module is 87k local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 2;
long lCNT ;
BOOL IRET = pac_GetPWMPulseCount(hPort, iSlot, iChannel, &lCNT);
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 2;
long lCNT ;
boolIRET = PACNET.PWM.GetPWMPulseCount(hPort, iSlot, iChannel, ref lCNT);
PACNET.UART.Close(hPort);
```

3.9. Backplane Timer API

Backplane timer API supports to hardware timer including timerout/timer1/timer2.

The timer of the development program is used on the WinCE platform

Because these timers (VC/C#/VB program) cannot adjust the priority of thread execution, their accuracy will be affected by the system scheduling and have an error of more than millisecond. So, they cannot realize the requirements of real-time.

BPtimer is a hardware timer provided by ICPDAS PAC. It provides a high-resolution timer (unit microseconds). The priority of the timer thread can be adjusted to meet real-time requirements.

Blackplane API function:

(1).pac_SetBPTimerOut
(2).pac_SetBPTimer
(3).pac_KillBPTimer
(4).pac_SetBPTimerInterruptPriority

[Operating Restrictions]

1. The BPtimer provides two unit modes(Only one timer can be used at the same time.):
   timer 1 :1 microsecond
   timer 2 : 10 microseconds.

2. The BPtimer shares interrupts with the backplane slot and some COM ports.
   The interrupts will affect each other when used at the same time.
   Do not use the BPTimer with other interrupted modules or COM ports at the same time.

Supported PACs

The table below lists the backplane timer functions that are supported by each PAC.

Models / Functions	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_GetBPTimerTimeTick_ms	Y	-	Y	-	Y	Y	Y	-	Y
pac_GetBPTimerTimeTick_us	Y	-	Y	-	Y	Y	Y	-	Y
pac_SetBPTimer	Y	-	Y	-	Y	Y	Y	-	Y
pac_SetBPTimerOut	Y	-	Y	-	Y	Y	Y	-	Y
pac_SetBPTimerInterruptPriority	Y	-	Y	-	Y	Y	Y	-	Y
pac_KillBPTimer	Y	-	Y	-	Y	Y	Y	-	Y

Backplane Timer Functions

The following functions are used to retrieve or set the backplane timer.

PACSDK Functions	PACNET Functions	Description
pac_GetBPTimerTimeTick_ms	BPTimer.GetBPTimerTimeTick_ms	returns the number of milliseconds have elapsed since the system was started, excluding any time that the system was suspended.
pac_GetBPTimerTimeTick_us	BPTimer.GetBPTimerTimeTick_us	returns the number of microsecond have elapsed since the system was started, excluding any time that the system was suspended.
pac_SetBPTimer	BPTimer.SetBPTimer	creates a hardware timer with the specified time-out value.
pac_SetBPTimerOut	BPTimer.SetBPTimerOut	creates a hardware timer with the specified time-out value of high/low wave.
pac_SetBPTimerInterruptPriority	BPTimer.SetBPTimerInterruptPriority	sets the priority for a real-time thread of the backplane timer.
pac_KillBPTimer	BPTimer.KillBPTimer	destroys the specified timer event identified by type, set by an earlier call to pac_SetBPTimer.

3.9.1. pac_GetBPTimerTimeTick_ms

This function returns the number of milliseconds have elapsed since the system was started, excluding any time that the system was suspended.

Syntax

C++

```
DWORD pac_GetBPTimerTimeTick_ms();
```

Parameters

This function has no parameters.

Return Value

The number of milliseconds indicates success.

Examples

This function has no examples.

3.9.2. pac_GetBPTimerTimeTick_us

This function returns the number of microsecond have elapsed since the system was started, excluding any time that the system was suspended.

Syntax

C++

```
DWORD pac_GetBPTimerTimeTick_us();
```

Parameters

This function has no parameters.

Return Value

The number of microseconds indicates success.

Examples

This function has no examples.

3.9.3. pac_SetBPTimer

This function creates a hardware timer with the specified time-out value.

A time-out value is specified, and every time a time-out occurs, the system posts an interrupt signal to the system and the pass the message to an application-defined callback function.

Syntax

C++

```
Bool pac_SetBPTimer(int type,unsigned int uElapse, pac_TIMEROUT_CALLBACK_FUNC f );
```

Parameters

type

[in] Specifies the type of the timer.

1 (Timer 1): 1 microsecond timer

2 (Timer 2): 10 microsecond timer

Others: Not applicable

uElapse

[in] Specifies the elapsed time.

Timer 1: A value of a timerout signal as integer from 0~65535, in 1 microsecond.

Timer 2: A value for a timerout signal as integer from0~65535, in 10 microseconds.

f

Specifies the address of the application-supplied f callback function.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is zero.

Examples

[C]

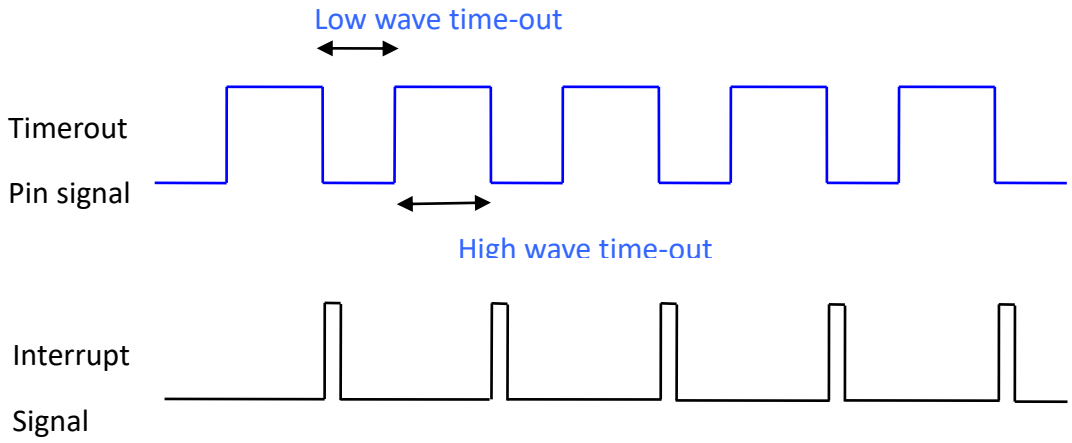
```
int CALLBACK TIMER()    //Interrupt Function
{
  /*Add the user control code here*/
  return 0;    // Interrupt done
}
// Set timer1 with 200 microsecond interval
pac_SetBPTimer(1, 200, TIMER);
```

3.9.4. pac_SetBPTimerOut

This function creates a hardware timer with the specified time-out value of high/low wave.

The time-out value of high/low wave are specified, and every time the time-out of high wave and low wave occur, the system posts an interrupt signal to the system and the pass the message to an application-defined callback function.

The timerout pin on each slot will be triggered while a timerout signal has been outputted. The timeourput pin can be used to acquire the synchronized data on each slot.



Syntax

C++

```
bool pac_SetBPTimerOut(unsigned int uHighElapse,unsigned int uLOwElapse,
pac_TIMEROUT_CALLBACK_FUNC f );
```

Parameters

uHighElapse

[in] Specifies the elapsed time value for a high wave of the timerout signal as integer from 0~65535, in microseconds.

uLOwElapse

[in] Specifies the elapsed time value for a low wave of the timerout signal as integer from 0~65535, in microseconds.

f

Specifies the address of the application-supplied f callback function.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, The return value is zero.

Examples

[C]

```
int CALLBACK TIMER()    //Interrupt Function
{
  /*Add the user control code here*/
  return 0;    // Interrupt done
}
// Set timer with 200 microsecond interval
pac_SetBPTimerOut(200, 300, TIMER);
```

3.9.5. pac_SetBPTimerInterruptPriority

This function sets the priority for a real-time thread of the backplane timer.

Syntax

C++

```
bool pac_SetBPTimerInterruptPriority(
    int type,
    int nPriority
);
```

Parameters

type

[in] Specifies the backplane timer.

0: Timerout

1: Timer 1

2: Timer 2

nPriority

[in] Priority to set for the thread

This value can range from 0 through 255, with 0 as the highest priority.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FLASE.

Examples

[C]

```
int CALLBACK TIMER()    //Interrupt Function
{
  return 0;    // Interrupt done
}
pac_SetBPTimer (1, 200, TIMER); // Set timer1 with 200 microsecond interval
pac_SetBPTimerInterruptPriority(1, 100); // Set the priority of timer 1 to 100
```

3.9.6. pac_KillBPTimer

This function destroys the specified timer event identified by type, set by an earlier call to pac_SetBPTimer.

Syntax

C++

```
void pac_KillBPTimer(int type);
```

Parameters

type

[in] Specifies the timer.

0 (Timerout)

1 (Timer 1): 1 microsecond timer

2 (Timer 2): 10 microsecond timer

Return Value

This function has does not return a value.

Examples

[C]

```
// Destroy the timer 1
pac_KillBPTimer(1);
```

3.10. Error Handling API

The error handling functions enable you to receive and display error information for your application.

Supported PACs

The table below lists the error handling functions that are supported by each PAC.

Functions \ Models	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_GetLastError	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SetLastError	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetErrorMessage	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_ClearLastError	Y	Y	Y	Y	Y	Y	Y	Y	Y

Error Handling Functions

The following functions are used with error handling.

PACSDK Functions	PACNET Functions	Description
pac_GetLastError	ErrHandling.GetLastError	Retrieves the last-error code value.
pac_SetLastError	ErrHandling.SetLastError	sets the last-error code.
pac_GetErrorMessage	ErrHandling.GetErrorMessage	Enter the error code to retrieves a error message string.
pac_ClearLastError	ErrHandling.ClearLastError	clears the last-error code.

3.10.1. pac_GetLastError

Retrieves the last-error code value.

If the development program occurs error.

It determines conveniently what kind of problem caused the error.

Syntax

C++

```
DWORD pac_GetLastError();
```

Parameters

This function has no parameters.

Return Value

The Return Value section of each function page notes the conditions under which the function sets the last-error code.

Examples

This function has no examples.

Remarks

You should call the pac_GetLastError function immediately when a function's return value indicates that such a call will return useful data. That is because some functions call pac_SetLastError(0) when they succeed, wiping out the error code set by the most recently failed function.

For an example, please refer to pac_GetErrorMerrage in this chapter.

To obtain an error string for XPAC error codes, use the pac_GetErrorMessage function. For a complete list of error codes, see Appendix A. System Error Code.

The following table lists the system error codes ranges for each function reference.

Error Type	Explanation	Range
PAC_ERR_SUCCESS	No error, success	0x00000
PAC_ERR_UNKNOWN	A error which is undefined	0x00001
Basic	Defined common error conditions	0x10000～
Memory	About memory access	0×11000～
Watchdog	About watchdog	0x12000～
Interrupt	About interrupt	0x13000～
UART	About uart protocol	0x14000～
IO	About I/O modules	0x15000～
Users	For user	0x20000～

3.10.2. pac_SetLastError

This function sets the last-error code.

If the development program occurs error.

It determines conveniently what kind of problem caused the error.

Syntax

C++

```
void pac_SetLastError(
    DWORD errno
);
```

Parameters

errno

[in] Specifies the last-error code.

Return Value

This function has does not return a value.

Examples

This function has no examples.

Remarks

Applications can optionally retrieve the value set by this function by using the pac_GetLastError function.

The error codes are defined as DWORD values. If you are defining an error code, ensure that your error code does not conflict with any XPacSDK-defined error codes.

We recommend that your error code should be greater than 0x20000.

For more information about the definition of error codes, please refer to pac_GetLastError in this document.

3.10.3. pac_GetErrorMessage

Enter the error code to retrieves a error message string.

Syntax

C++

```
void pac_GetErrorMessage(
    DWORD dwMessageID,
    LPTSTR lpBuffer
);
```

Parameters

dwMessageID

[in] Specifies the 32-bit message identifier for the requested message.

lpBuffer

[out] A pointer to a buffer that receives the error message.

Return Value

This function has does not return a value.

Examples

[C]

```
int main(int argc, char* argv[] )
{
    if(argc < 3)
    {
        printf("usage: ReadMemory [ address ] [ dwLength ] [ mem_type ] \n\n");
        printf("where\n");
        printf("     address:\n");
        printf("          - the memory address where read from.\n");
        printf("     dwLength:\n");
        printf("          - number of characters to be read.\n");
        printf("     mem_type:\n");
        printf("          - 0     SRAM\n");
        printf("          - 1     EEPROM\n");
    }
    else
    {
        BYTE buffer[4096] ;
        BOOL err;
        char strErr[32] ;
        memset(buffer, 0, 4096);
        if(atoi(argv[3] ) == 0)
        {
            printf("The size of SRAM is %d\n", pac_GetMemorySize(atoi(argv[3] )));
            err = pac_ReadMemory(atoi(argv[1] ), buffer, atoi(argv[2] ), atoi(argv[3] ));
            if(err == FALSE)
            {
                pac_GetErrorMessage(pac_GetLastError(), strErr);
                printf("Read SRAM failure!. The error code is %x\n", pac_GetLastError());
                printf("%s", strErr);
                return 0;
            }
            printf("%s\n", buffer);
        }
        else
```

```
        {
            printf("The size of EEPROM is %d\n", pac_GetMemorySize(atoi(argv[3] )));
            err = pac_ReadMemory(atoi(argv[1] ), buffer, atoi(argv[2] ), atoi(argv[3] ));
            if(err == FALSE)
            {
                pac_GetErrorMessage(pac_GetLastError(), strErr);
                printf("Read EEPROM failure!. The error code is %x\n", pac_GetLastError());
                printf("%s", strErr);
                return 0;
            }
            printf("%s\n", buffer);
        }
    }
    return 0;
}
```

[C#]

```
class Program
{
static void Main(string[] args)
   {
        if (args.Length < 3){
            Console.WriteLine("pac_WriteDO for 8000 modules\n\n");
            Console.WriteLine("usage: pac_WriteDO [ Slot ] [ total channel ] [ DO's value ]
\n\n");
            Console.WriteLine("where\n");
            Console.WriteLine("Slot:\n");
            Console.WriteLine(" - number of slot for local modules\n");
            Console.WriteLine("total channel:\n");
            Console.WriteLine(" - number of DO's channel\n");
            Console.WriteLine("DO's value:\n");
            Console.WriteLine(" - 1 is to turn on the DO channel; 0 is off.\n");
        }
        else{
            bool err;
            err = PACNET.PAC_IO.WriteDO(IntPtr.Zero,Convert.ToInt32(args[1] ),
                                Convert.ToInt32(args[2] ), Convert.ToUInt32(args[3] ));
            if (err == false)
            {
                Cnsole.WriteLine("Write DO's Error: " + PACNET.
ErrHandling.GetErrorMessage(XPac.GetLastError()) + ". The error code is " +
PACNET.ErrHandling.GetLastError().ToString() + "\n");
                return;
            }
        }
   }
}
```

Remarks

The pac_GetErrorMessage function can be used to obtain error message strings for the XPac error codes returned by pac_GetLastError, as shown in the following example.

```
TCHAR Buffer[32];
pac_GetErrorMessage(pac_GetLastError(), Buffer);
MessageBox( NULL, Buffer, L"Error", MB_OK | MB_ICONINFORMATION );
```

3.10.4. pac_ClearLastError

This function clears the last-error code.

Syntax

C++

```
void pac_ClearLastError();
```

Parameters

This function has no parameters.

Return Value

This function has does not return a value.

Examples

This function has no examples.

Remarks

The pac_ClearLastError function clears the last error.

3.11. Misc API

Supported PACs

The table below lists the Misc functions that are supported by each PAC.

Functions \ Models	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
byte AnsiString	Y	Y	Y	Y	-	-	Y	Y	Y
WideString	Y	Y	Y	Y	-	-	Y	Y	Y
pac_AnsiToWideString	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_WideToAnsiString/pac_WideStringToAnsi	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_DoEvent/pac_DoEvents	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetCurrentDirectory	Y	Y	Y	Y	-	-	Y	Y	Y
pac_GetCurrentDirectoryW	Y	Y	Y	Y	-	-	Y	Y	Y
byte AnsiString	Y	Y	Y	Y	-	-	Y	Y	Y
WideString	Y	Y	Y	Y	-	-	Y	Y	Y
pac_AnsiToWideString	Y	Y	Y	Y	Y	Y	Y	Y	Y

Misc Functions

The following functions are used to retrieve or set the memory.

PACSDK Functions	PACNET Functions	Description
byte AnsiString	MISC.AnsiString	converts a unicode string to an ANSI byte array.
WideString	MISC.WideString	converts an ANSI byte array to Unicode string.
pac_AnsiToWideString	N/A	converts an ANSI string to a Unicode string.
pac_WideToAnsiString pac_WideStringToAnsi	N/A	converts a Unicode string to an ANSI string.
pac_DoEvent/pac_DoEvents	N/A	handles all events.
pac_GetCurrentDirectory	MISC.GetCurrentDirectory	retrieves the current directory of the Char data type for the current application.
pac_GetCurrentDirectoryW	N/A	retrieves the current directory of the TCHAR data type for the current application

3.11.1. byte AnsiString

This function converts a unicode string to an ANSI byte array.

Syntax

C#

```
byte [] AnsiString(
    string str
);
```

Parameters

str

[in] Points to the Unicode string to be converted.

Return Value

Returns the ANSI byte array.

Examples

[C#]

```
byte[] result = new byte[32] ;
IntPtr hPort = PACNET.UART.Open("COM1,115200,N,8,1");
PACNET.Sys.ChangeSlot(Convert.ToByte(1));
PACNET.UART.SendCmd(hPort, PACNET.MISC.AnsiString("$00M"), result);
string str = PACNET.MISC.WideString(result);
```

Remarks

In .NET, if we want to convert a Unicode string to ANSI or vice versa, we should convert through byte array.

3.11.2. WideString

This function converts an ANSI byte array to Unicode string.

Syntax

C#

```
String WideString(
    byte [] CharStr
);
```

Parameters

CharStr

[in] Points to the ANSI byte array to be converted.

Return Value

Returns the Unicode string.

Examples

[C#]

```
byte[] result = new byte[32] ;
IntPtr hPort = PACNET.UART.Open("COM1,115200,N,8,1");
PACNET.Sys.ChangeSlot(Convert.ToByte(1));
PACNET.UART.SendCmd(hPort, PACNET.MISC.AnsiString("$00M"), result);
string str = PACNET.MISC.WideString(result);
```

Remarks

In .NET, if we want to convert a Unicode string to ANSI, or vice versa, we should convert through byte array.

3.11.3. pac_AnsiToWideString

This function converts an ANSI string to a Unicode string.

Syntax

C++

```
void pac_AnsiToWideString(
    LPCSTR ASTR,
    LPTSTR WSTR
);
```

Parameters

astr

[in] Points to the ANSI string to be converted.

wstr

[in] A pointer to a buffer location that receives the converted Unicode string.

Return Value

This function has does not return a value.

Examples

[C]

```
char ansiString[128] = "This is an ansi string";
TCHAR uniString[128] ;
pac_AnsiToWideString(ansiString, uniString);
// The string "This is an ansi string" will show in the messagebox correctly
MessageBox(NULL, uniString, NULL, MB_OK);
```

Remarks

The maximum size of the string buffer is 2 Kbytes.

3.11.4. pac_WideToAnsiString/pac_WideStringToAnsi

This function converts a Unicode string to an ANSI string.

Syntax

C++

```
void pac_WideToAnsiString(
    LPCTSTR WSTR,
    LPSTR ASTR
);
```

Parameters

wstr

[in] Points to the Unicode string to be converted.

astr

[in] A pointer to a buffer location that receives the converted ANSI string.

Return Value

This function has does not return a value.

Examples

[C]

```
TCHAR uniString[128] = TEXT("This is a unicode string");
char ansiString[128] ;
pac_WideStringToAnsi(uniString, ansiString);
printf("%s", ansiString); // The string "This is a unicode string" will show the console mode correctly
```

Remarks

The maximum size of the string buffer is 2 kbytes.

3.11.5. pac_DoEvent/pac_DoEvents

This function handles all events.

When you run a Windows Form, it creates the new form, which then waits for events to handle. Each time the form handles an event, it processes all the code associated with that event. All other events wait in the queue. While your code handles the event, your application does not respond. If you call pac_DoEvents in your code, your application can handle the other events.

Syntax

C++

```
void pac_DoEvents();
```

Parameters

This function has no parameters.

Return Value

This function has does not return a value.

Examples

[C]

```
int counter = 0;
char buf[10] ;
bFlag = true;
while(bFlag)
{
   pac_DoEvents();
   sprintf(buf, %d", counter);
   SetDlgItemText(IDC_EDIT1, buf);
   counter++;
}
```

3.11.6. pac_GetCurrentDirectory

This function retrieves the current directory of the Char data type for the current application.

Syntax

C++

```
bool pac_GetCurrentDirectory(
    char* cBuffer,
    DWORD nSize
);
```

Parameters

cBuffer

[out] A pointer to a buffer that receives the current directory.

nSize

[in] A pointer to a variable that specifies the size, in chars, of the buffer.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call GetLastError to obtain the last error ocde of Windows API.

Examples

[C]

```
char buf[1024] ;
pac_GetCurrentDirectory(buf, 1024);
```

[C#]

```
string str;
str = PACNET.MISC. GetCurrentDirectory();
```

3.11.7. pac_GetCurrentDirectoryW

This function retrieves the current directory of the TCHAR data type for the current application.

Syntax

C++

```
bool pac_GetCurrentDirectoryW(
    LPTSTR cBuffer,
    DWORD nSize
);
```

Parameters

LPTSTR

[out] A pointer to a buffer that receives the current directory.

nSize

[in] A pointer to a variable that specifies the size, in TCHARs, of the buffer.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call GetLastError to obtain the last error code of Windows API.

Examples

[C]

```
TCHAR buf[1024] ;
pac_GetCurrentDirectory(buf, 1024);
```

Appendix.

A. System Error Codes

This following table provides a list of system error code that is intended to be used by programmers so that the software they write can better deal with errors.

The error codes/error messages are returned by the pac_GetLastError/pac_GetErrorMessage function when one of the PAC services functions fail.

Error Code	Error Message
0x00001	Unknow Error
0x10001	Slot registered error
0x10002	Slot not registered error
0x10003	Unknown Module
0x10004	Module doesn't exist
0x10005	Invalid COM port number
0x10006	Function not supported
0x10007	Module doesn't exist
0x10008	Slot not registered error
0x11001	EEPROM accesses invalid address
0x11002	SRAM accesses invalid address
0x11003	SRAM accesses invalid type
0x11004	NVRAM accesses invalid address
0x11005	EEPROM write protection
0x11006	EEPROM write fail
0x11007	EEPROM read fail
0x12001	The input value is invalid
0x12002	The wdt doesn't exist
0x12003	The wdt init error
0x13001	Create interrupt's event failure
0x14001	Uart check sum error
0x14002	Uart read timeout
0x14003	Uart response error
Error Code	Error Message

0x14004	Uart under input range
0x14005	Uart exceed input range
0x14006	Uart open filed
0x14007	Uart get Comm Modem status error
0x14008	Uart get wrong line status
0x14009	Uart internal buffer overflow
0x15001	I/O card does not support this API function
0x15002	API unsupport this I/O card
0x15003	Slot's value exceeds its range
0x15004	Channel's value exceeds its range
0x15005	Gain's value exceeds its range
0x15006	Unsupported interrupt mode
0x15007	I/O value is out of the range
0x15008	I/O channel is out of the range
0x1500A	DIO channel can't overwrite
0x1500B	AI/O channel can't overwrite
0x16001	Backplane Timer registed
0x16002	Backplane Timer not registed

B. API Comparison

The following tables give a brief summary of the capabilities of each API function, where **Y/ X** means **supported/unsupported**

System Information API

Functions \ Models	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_GetModuleName	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetRotaryID	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetSerialNumber	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetSDKVersion	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_ChangeSlot	Y	-	Y	-	Y	Y	Y	-	Y
pac_CheckSDKVersion	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_ModuleExists	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetOSVersion	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetMacAddress	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_ReBoot	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetCPUVersion	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_EnableLED	Y	Y	Y	Y	-	-	Y	Y	Y
pac_EnableLEDs	-	-	-	-	-	Y	-	-	-
pac_BackwardCompatible()	Y	-	-	-	-	-	Y	-	-

System Information API

Functions \ Models	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_GetEbootVersion	Y	Y	Y	Y	-	-	Y	Y	Y
pac_GetComMapping	Y	Y	-	Y	-	-	Y	Y	-
pac_GetModuleType	Y	Y	Y	-	Y	Y	Y	Y	Y
pac_GetPacNetVersion	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_BuzzerBeep	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetBuzzerFreqDuty	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SetBuzzerFreqDuty	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_StopBuzzer	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetDIPSwitch	Y	-	-	-	Y	Y	Y	-	-
pac_GetSlotCount	Y	-	Y	-	Y	Y	Y	-	Y
pac_EnableRetrigger	Y	-	Y	-	Y	Y	Y	-	Y
pac_GetBackplaneID	Y	-	Y	-	Y	Y	Y	-	Y
pac_GetBatteryLevel	Y	-	Y	-	Y	Y	Y	-	Y
pac_RegistryHotPlug(Beta)	-	-	-	-	-	-	-	-	-
pac_UnregistryHotPlug(Beta)	-	-	-	-	-	-	-	-	-
pac_SetBackLight	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetBackLight	Y	Y	Y	Y	Y	Y	Y	Y	Y

Interrupt API

Functions \ Models	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_RegisterSlotInterrupt	Y	-	Y	-	Y	Y	Y	-	Y
pac_UnregisterSlotInterrupt	Y	-	Y	-	Y	Y	Y	-	Y
pac_EnableSlotInterrupt	Y	-	Y	-	Y	Y	Y	-	Y
pac_SetSlotInterruptPriority	Y	-	Y	-	Y	Y	Y	-	Y
pac_InterruptInitialize	Y	-	Y	-	Y	Y	Y	-	Y
pac_GetSlotInterruptEvent	Y	-	Y	-	Y	Y	Y	-	Y
pac_SetSlotInterruptEvent	Y	-	Y	-	Y	Y	Y	-	Y
pac_SetTriggerType	Y	-	Y	-	Y	Y	Y	-	Y
pac_GetSlotInterruptID	Y	-	Y	-	Y	Y	Y	-	Y
pac_InterruptDone	Y	-	Y	-	Y	Y	Y	-	Y

Memory Access API

Models / Functions	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_GetMemorySize	Y	Y▲	Y	Y	Y	Y	Y	Y▲	Y
pac_ReadMemory	Y	Y▲	Y	Y	Y	Y	Y	Y▲	Y
pac_WriteMemory	Y	Y▲	Y	Y	Y	Y	Y	Y▲	Y
pac_EnableEEPROM	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SDExists	Y	Y	Y	Y	-	-	Y	Y	Y
pac_SDMount	Y	Y	Y	Y	-	-	Y	Y	Y
pac_SDOnside	Y	Y	Y	Y	-	-	Y	Y	Y
pac_SDUnmount	Y	Y	Y	Y	-	-	Y	Y	Y

▲ WP-5xxx only supports the memory type 1 (EEPROM), not type 0 (SRAM).

Watchdog API

Functions \ Models	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_EnableWatchDog	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_DisableWatchDog	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RefreshWatchDog	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetWatchDogState	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetWatchDogTime	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SetWatchDogTime	Y	Y	Y	Y	Y	Y	Y	Y	Y

Registry API

Models / Functions	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_RegCountKey	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegCountValue	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegCreateKey	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegDeleteKey	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegDeleteValue	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegGetDWORD	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegGetKeyByIndex	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegGetKeyInfo	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegGetString	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegGetValueByIndex	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegKeyExist	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegSave	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegSetString	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_RegSetDWORD	Y	Y	Y	Y	Y	Y	Y	Y	Y

UART API

Functions \ Models	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
uart_Close	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_SendExt	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_Send	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_RecvExt	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_Recv	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_SendCmdExt	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_SetTimeOut	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_EnableCheckSum	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_SetTerminator	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_BinSend	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_BinRecv	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_BinSendCmd	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_GetLineStatus	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_GetDataSize	Y	Y	Y	Y	Y	Y	Y	Y	Y
uart_SetLineStatus	Y	Y	Y	Y	Y	Y	Y	Y	Y

PAC_IO API

Functions \ Models	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_GetBit	Y	-	Y	-	Y	Y	Y	-	Y
pac_WriteDO/pac_WriteDO_MF	Y	-	Y	-	Y	Y	Y	-	Y
pac_WriteDOBit	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadDO/pac_ReadDO_MF	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadDI/pac_ReadDI_MF	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadDIO/pac_ReadDIO_MF	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadDILatch	Y	-	Y	-	Y	Y	Y	-	Y
pac_ClearDILatch	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadDIOLatch	Y	-	Y	-	Y	Y	Y	-	Y
pac_ClearDIOLatch	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadDICNT/pac_ReadDICNT_MF	Y	-	Y	-	Y	Y	Y	-	Y
pac_ClearDICNT/pac_ClearDICNT_MF	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadDICNTAll	Y	-	Y	-	Y	Y	Y	-	Y

PAC_IO API

Models / Functions	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_ClearDICNTAll	Y	-	Y	-	Y	Y	Y	-	Y
pac_WriteAO/pac_WriteAO_MF	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadAO	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadAI	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadAIHex	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadAIAllExt	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadAIAll	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadAIAllHexExt	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadAIAllHex	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadCNT	Y	-	Y	-	Y	Y	Y	-	Y
pac_ClearCNT	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadCNTOverflow	Y	-	Y	-	Y	Y	Y	-	Y
pac_WriteModuleSafeValueDO/pac_WriteModuleSafeValueDO_MF	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadModuleSafeValueDO/pac_ReadModuleSafeValueDO_MF	Y	-	Y	-	Y	Y	Y	-	Y

PAC_IO API

Functions \ Models	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_WriteModulePowerOnValueDO/pac_WriteModulePowerOnValueDO_MF	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadModulePowerOnValueDO/pac_ReadModulePowerOnValueDO_MF	Y	-	Y	-	Y	Y	Y	-	Y
pac_WriteModuleSafeValueAO	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadModuleSafeValueAO	Y	-	Y	-	Y	Y	Y	-	Y
pac_WriteModulePowerOnValueAO	Y	-	Y	-	Y	Y	Y	-	Y
pac_ReadModulePowerOnValueAO	Y	-	Y	-	Y	Y	Y	-	Y
pac_GetModuleLastOutputSource	Y	-	Y	-	Y	Y	Y	-	Y
pac_GetModuleWDTStatus	Y	-	Y	-	Y	Y	Y	-	Y
pac_GetModuleWDTConfig	Y	-	Y	-	Y	Y	Y	-	Y
pac_SetModuleWDTConfig	Y	-	Y	-	Y	Y	Y	-	Y
pac_ResetModuleWDT	Y	-	Y	-	Y	Y	Y	-	Y

PAC_IO API

Models / Functions	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_RefreshModuleWDT	Y	-	Y	-	Y	Y	Y	-	Y
pac_InitModuleWDTInterrupt	Y	-	Y	-	Y	Y	Y	-	Y
pac_GetModuleWDTInterrupt Status	Y	-	Y	-	Y	Y	Y	-	Y
pac_SetModuleWDTInterrupt Status	Y	-	Y	-	Y	Y	Y	-	Y

PWM API

Functions \ Models	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_SetPWMDuty	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetPWMDuty	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SetPWMFrequency	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetPWMFrequency	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SetPWMMode	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetPWMMode	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SetPWMDITriggerConfig	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetPWMDITriggerConfig	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SetPWMStart	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SetPWMSynChannel	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetPWMSynChannel	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SyncPWMStart	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SavePWMConfig	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetPWMDIOStatus	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_SetPWMPulseCount	Y	Y	Y	Y	Y	Y	Y	Y	Y

Backplane Timer API

Models / Functions	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
pac_GetBPTimerTimeTick_ms	Y	-	Y	-	Y	Y	Y	-	Y
pac_GetBPTimerTimeTick_us	Y	-	Y	-	Y	Y	Y	-	Y
pac_SetBPTimer	Y	-	Y	-	Y	Y	Y	-	Y
pac_SetBPTimerOut	Y	-	Y	-	Y	Y	Y	-	Y
pac_SetBPTimerInterruptPriority	Y	-	Y	-	Y	Y	Y	-	Y
pac_KillBPTimer	Y	-	Y	-	Y	Y	Y	-	Y

Misc API

Functions \ Models	CE7				CE6		CE5		
	WP-9x2x WP-8x2x	WP-5231 WP-5231M WP-5231PM-3GWA	VP-x231	VP-x201	XP-8x4x	XP-8x4x-Atom XP-8x3x	WP-8x3x WP-8x4x	WP-51x1-OD WP-51x1	VP-23W1 VP-25W1 VP-4131
byte AnsiString	Y	Y	Y	Y	-	-	Y	Y	Y
WideString	Y	Y	Y	Y	-	-	Y	Y	Y
pac_AnsiToWideString	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_WideToAnsiString/pac_WideStringToAnsi	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_DoEvent/pac_DoEvents	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetCurrentDirectory	Y	Y	Y	Y	-	-	Y	Y	Y
pac_GetCurrentDirectoryW	Y	Y	Y	Y	-	-	Y	Y	Y
byte AnsiString	Y	Y	Y	Y	-	-	Y	Y	Y
WideString	Y	Y	Y	Y	-	-	Y	Y	Y
pac_AnsiToWideString	Y	Y	Y	Y	Y	Y	Y	Y	Y

C. What's New in PACSDK

PACSDK is the next version of XPACSDK and WinPACSDK. It builds on the features of XPACSDK and WinPACSDK library by providing the following:

C.1. PACSDK.dll modifications and updates

The new PACSDK.dll provides support for two platforms, one being designed for the WinPAC series (ARM platforms) and the other for the XPAC series (x86 platforms).
However, there are a number of modifications and updates that are included in the new PACSDK, which are listed below. (Note: Compared to the previous WinPAC/XPAC SDK, these modification and updates need to be made to the previously implemented WinPAC/XPAC programs so that it will work with the new SDK)

1. pac_EnableLED

The original **pac_EnableLED (bool bFlag)** function can be used only for the WinPAC series in the previous SDK, and the original **pac_EnableLED (int pin, bool bFlag)** function can be used only for the XPAC series in the previous SDK.
Consequently, this API function cannot be integrated to the PACSDK.dll because of the conflicting parameters Conflict function parameters

As a result, the function in PACSDK.dll has been changed to
pac_EnableLED (bool bFlag) is been reserved and
a new API function has been added,
pac_EnableLEDs (int pin,bool bFlag) .

2. Add Registry API for XPAC series

The suite of the API functions listed below doesn't been provided in the previous SDK, XPACSDK_CE.dl, and supported in the new PACSDK.dll for the XPAC series.
(The previous SDK, WinPACSDK.dll for the WinPAC series and the new SDK, PACSDK.dll for the WinPAC series have provided the support of all the functions below)

pac_RegCountKey

pac_RegCountValue

pac_RegCreateKey

pac_RegDeleteKey

pac_RegDeleteValue

pac_RegGetDWORD

pac_RegGetKeyByIndex

pac_RegGetKeyInfo

pac_RegGetString

pac_RegGetValueByIndex

pac_RegKeyExist

pac_RegSave

pac_RegSetString

pac_RegSetDWORD

3. Add I/O WDT, PowerOn/Safe Value API for pure DIO modules

The new PACSDK.dll provides the support of I/O WDT, Power On and Safe value functions for pure DIO DCON modules. (Refer to Note 1) These functions aren't supported for the previous SDK, WinPacSDK.dll and XPacSDK_CE.dll.

pac_GetModuleLastOutputSource
pac_GetModuleWDTStatus
pac_GetModuleWDTConfig
pac_SetModuleWDTConfig
pac_ResetModuleWDT
pac_RefreshModuleWDT
pac_InitModuleWDTInterrupt
pac_SetModuleWDTInterruptStatus
pac_GetModuleWDTInterruptStatus
pac_ReadModuleSafeValueDO
pac_WriteModuleSafeValueDO
pac_ReadModuleSafeValueAO
pac_WriteModuleSafeValueAO
pac_ReadModulePowerOnValueDO
pac_WriteModulePowerOnValueDO
pac_ReadModulePowerOnValueAO
pac_WriteModulePowerOnValueAO

Notes

1. The each of API function is used for the DCON module which is provided with Power ON or Safe value function.
2. I-7K/I-87K series modules provided with Power ON or Safe Value function can support the API functions above. I-8K series module provide the functions is only I-8041RW.

4. Add I/O accessing API functions for the Multi-function modules

The new PACSDK.dll provides the support of I/O accessing functions (including Write/Read DIO, AIO, Read DI counter and I/O WDT, Power On and Safe value function for the Multi-function DCON modules. (Refer to Note 2 regarding of the definition of Multi-function modules) These functions aren't supported for the previous SDK, WinPAC.dll and XPACSDK_CE.dll.

pac_WriteAO_MF(Note:5)
pac_WriteModulePowerOnValueAO_MF
pac_WriteModuleSafeValueAO_MF
pac_WriteDO_MF
pac_ReadDIO_MF
pac_ReadDI_MF
pac_ReadDO_MF
pac_ReadDIO_DIBit_MF
pac_ReadDIO_DOBit_MF
pac_ReadDIBit_MF
pac_ReadDOBit_MF
pac_ReadDICNT_MF
pac_ClearDICNT_MF
pac_ReadModulePowerOnValueDO_MF
pac_WriteModulePowerOnValueDO_MF
pac_ReadModuleSafeValueDO_MF
pac_WriteModuleSafeValueDO_MF

Notes

1. The functions pac_WriteDO, pac_ReadDIO, pac_ReadDI, pac_ReadDO, pac_ReadDIO_DIBit, pac_ReadDIO_DOBit, pac_ReadDIBit, pac_ReadDOBit, pac_ReadDICNT, pac_ClearDICNT, pac_ReadDICNTAll and pac_ClearDICNTAll, which were supported in the previous SDK, are used to read and write the DIO channels for pure DIO DCON modules, which are defined as modules that only have DI, DO or DIO channels.
2. In addition to providing support for the API functions described above, the PACSDK also provides the support for the Multi-function API that is used to read and write the DIO channels for the Multi-function DCON modules, which are defined as modules that mainly act as AIO or Counters but are equipped with DIO channels. Such as the I-87005W/I-87016W/I-87082W/I-7016/I-7088, etc.)
3. The functions mentioned above (i.e., pac_WriteDO/ pac_ReadDIO, etc.) cannot be used to access Multi-function DCON modules. Only the pac_xxx_MF API allows access to Multi-function DCON modules.
4. In the PACSDK.dll, the process sing can be modified to send DCON commands without needing to determine the module name, which means that a the PAC_IO API functions can support access to the I-87K/I-8K series modules, I-7K series modules, I-8000 series modules units, tM series modules, and other OEM/ODM DCON modules.
5. The comparison table of pac_WriteAO / pac_WriteAO_MF Functions and available modules are as following:

Since November 1, 2012

pac_Write AO	pac_WriteAO_MF
I-87024W/CW/DW/RW, I-87024	I-87026PW
I-87028CW/UW	
I-87022	
I-87026	
I-7021,I -7021P	
I-7022	
I-7024, I-8024R	

5. Add Misc. API function for PACSDK

The new PACSDK.dll provides 2 miscellaneous API functions below.

pac_GetCurrentDirectory

pac_GetCurrentDirectoryW

6. Add the reserved memory section for XPAC series

In order to reserve some memory sections of EEPROM and SRAM for the use by the system, the reserved section of the pac_ReadMemory and pac_WriteMemory function must be changed. Thee reserved section is same with the WinPAC SDK. The definition of the items included in the reserved section is

EEPROM

0 ~0x1FFF (8KB) for users

0x2000~0x3FFF (8KB) is reserved for the system

SRAM

The size of the input range for the SRAM is only 0 ~0x6FFFF (448KB), with another 64KB of SRAM is reserved for use by the system.

In the previous XPAC SDK (XPacSDK_CE.dll), all memory space (0~0x3FFF, 16KB) of EEPROM is available for the use by the user, and all memory space (0~0x80000, 512KB) of SRAM is available for the use by the user.

7. Using the new SDK (PACSDK) in a C program

To use the new PACSDK in a C-based program, some code needs to be changed in the program.

Replace the previous header file by PACSDK.h

```
#include "WinPacSDK.h"
```

Changed as

```
#include " PACSDK.h"
```

WinPacSDK.h is used for both WinPAC or ViewPAC series program and it must be replaced by PACSDK.h

```
#include "XPacSDK_CE.h”
```

Changed as

```
#include " PACSDK.h"
```

XPacSDK_CE.h is used for the XPAC series program and it must be replaced by PACSDK.h

Replace the previous library file by PACSDK.lib

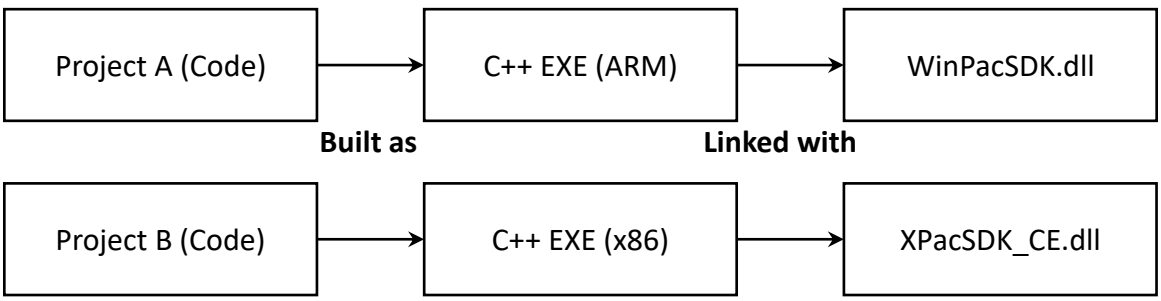
```
WinpacSDK.lib      // WinPAC or ViewPAC series
XPacSDK_CE.lib     // XPAC series
```

Changed as

```
PACSDK.lib
```

WinPacSDK.lib used for WinPAC or ViewPAC series and XPacSDK_CE.lib used for XPAC series are replaced by PACSDK.lib

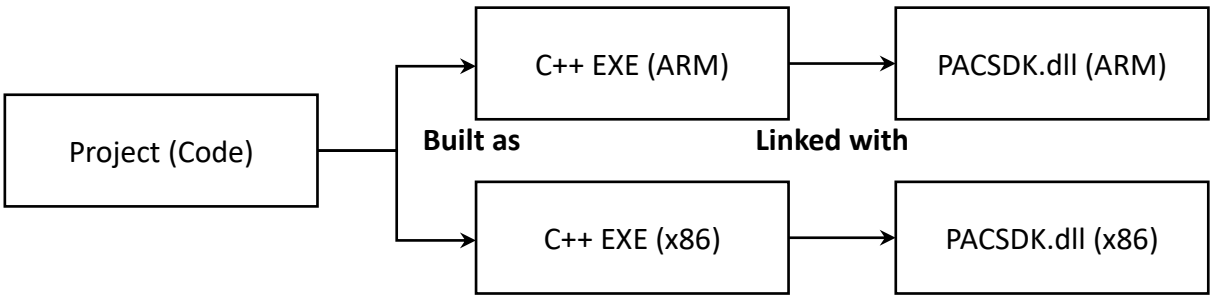
The original flowchart for a C program that is calling the previous SDK is illustrated below



Even if Project A applied to WinPAC series modules and Project B applied to XPAC series modules are functionally identical. The source code using the previous SDK cannot be exactly the same because of using the different header file and the few function names and error code defined in the previous SDK are different. So Project A and Project B are regarded as separate programs, cannot share the source code

The results of the above are
Project A is built as an ARM-based executable program and it must be run with WinPacSDK.dll.
Project B is built as an x86-based executable program and it must be run with XPacSDK_CE.dll.

The flowchart for a C program that is now calling the new SDK (PACSDK.dll) is as follows:



The benefits of using the new SDK:
A program applied to WinPAC series modules and the other program applied to XPAC series modules are functionally identical, because using the same header file and the API functions and error code on the library are exactly the same, the source code can be shared for two programs. The Project with the shared source code can be built as two different platform executable programs selecting the different Platform settings in the development environment while build the project.

The results of the above are
Project is built as an ARM-based executable program which runs with the ARM-based PaCSDK.dll and it's also built as an x86-based executable program which runs with x86-based PaCSDK.dll.

C.2. PACNET SDK modifications and updates

The .NET Compact Framework environment allows multiple high-level languages (C#, VB) to be used on different platforms without needing to be rewritten for specific architectures. The new PACNET.dll replaces the previous .NETCF SDK, WinPacNet.dll and XPacNet.dll files which means that NET CF programs linking to the PACNET.dll on a WinPAC device can be migrated to a XPAC device without needing to rewrite the code or rebuild the project and vice versa.

1. API function classification

All API functions for the WinPacNet.dll or the XPacNet.dll are placed in a single WinPacNet.WinPAC.xxx/XPacNET.XPac.xxx class, but the API functions for the PACNET.dll are classified as PACNET.sys, PACNET.Memory, and PACNET.Interrupt, etc.
The classifications applied to the API functions for the PACNET.dll as defined in the API user manual are as follows.

Classification in the API Manual	Class Name in PACNET.dll
2.1. System Information API	Sys
2.1. System Information API	Sys.Buzzer
2.2. Interrupt API	Interrupt
2.3. Memory Access API	Memory
2.4. Watchdog API	Sys.WDT
2.5. Registry API	PAC_Reg
2.6. UART API	UART
2.7. PAC_IO API	PAC_IO
2.8. PWM API	PWM
2.9. Backplane Timer API	BPTimer
2.10. Error Handling API	ErrHandling
2.11. Misc API	MISC

2. API functions modification

LED control API function (pac_EnableLED)

Refer to “pac_EnableLED” reference of PACSDK.dll modifications and updates for more details.
The modification in PACNET SDK, XPacNet.XPac.pac_EnableLED(pin, bFlag) function defined in XPacNet.dll has been changed as PACNET.Sys.EnableLEDs(pin, bFlag) PACNET.dll.

Add Registry API for XPAC series

Refer to “Add Registry API for XPAC series” reference of PACSDK.dll modifications and updates for more details.
The suite of the Registry API functions is placed in PACNET.PAC_Reg class.

Add I/O WDT, PowerOn/Safe Value API for pure DIO modules

Refer to “Add I/O WDT, PowerOn/Safe Value API for pure DIO modules” reference of PACSDK.dll modifications and updates for more details.
The suite of the I/O WDT, PowerOn/Safe Value API functions for pure DIO modules is placed in PACNET.PAC_IO class.

Add I/O WDT, PowerOn/Safe Value API for the Multi-function modules

Refer to “Add I/O WDT, PowerOn/Safe Value API for Multi-function modules” reference of PACSDK.dll modifications and updates for more details.
The suite of the I/O WDT, PowerOn/Safe Value API functions for Multi-function modules is also placed in PACNET.PAC_IO class.

Add Misc. API function for PACSDK

Refer to “Add Misc. API function for PACSDK” reference of PACSDK.dll modifications and updates for more details.
The suite of misc. API function is placed in PACNET.MISC class.

3. Enumerate the error codes

Add a function to enumerate all the error codes for PACSDK

The code snippet is as follows (The code is applicable to every C#/VB demo file)

```
uint ec = PACNET.ErrHandling.GetLastError();
MessageBox.Show(((PACNET.ErrCode)ec).ToString() + "\nError Code: 0x" + ec.ToString("X"));
```

The sample code is used to show the error code number and its e***numerated definition.***

If the last error code, 0x10001 is happened on the user's program.

The message box with "PAC_ERR_UNKNOWN Error Code:0x10001" caption will be shown.

4. Using the new SDK (PACNET) in a C# or VB.net program

To use the new PACNET in a C# or VB.net program, some code needs to be changes in the program.

In a C# program

Modify the code for using XPAC series devices, "using XPacNET" to "using PACNET".

```
using XPacNet;
```

Changed as

```
using PACNET;
```

Modify the code for using XPAC series devices, "using WinPacNet" to "using PACNET".

```
using WinPacNet;
```

Changed as

```
using PACNET;
```

In a VB.net program

Modify the code for using XPAC series devices, "Imports XPacNET" to "Imports PACNET".

```
Imports XpacNet
```

Changed as

```
Imports PACNET
```

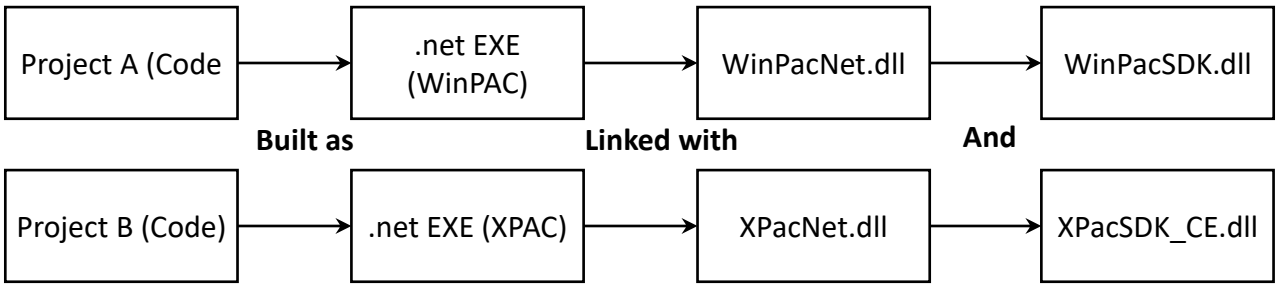
Modify the code for using XPAC series devices, "Imports WinPacNet" to "Imports PACNET".

```
Imports WinPacNet
```

Changed as

```
Imports PACNET
```

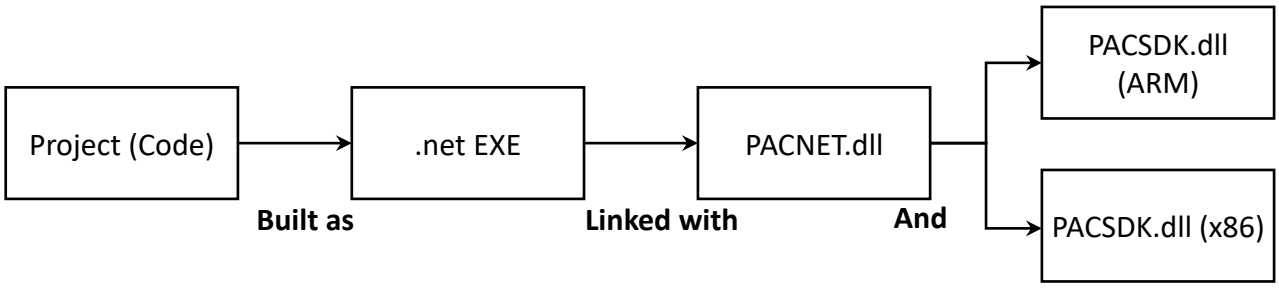
With the previous .NETCF library (WinPacNet.dll or XPacNet.dll), the flowchart was as follows:



Project A applied to WinPAC series modules and Project B applied to XPAC series modules are functionally identical, but the source code cannot be exactly the same because of using the different .NET CF library and few function name and error code are different. So Project A and Project B are regarded as separate programs, no relevance.

Project A for WinPAC series is built as an executable program which must be run with WinPacNet.dll and WinPacSDK.dll.
Project B for XPAC is built as an executable program which must be run with XpacNet.dll and XPacSDK_CE.dll.

With the new .NETCF library (PACNET.dll) and the flowchart becomes:



The benefits of using the new SDK

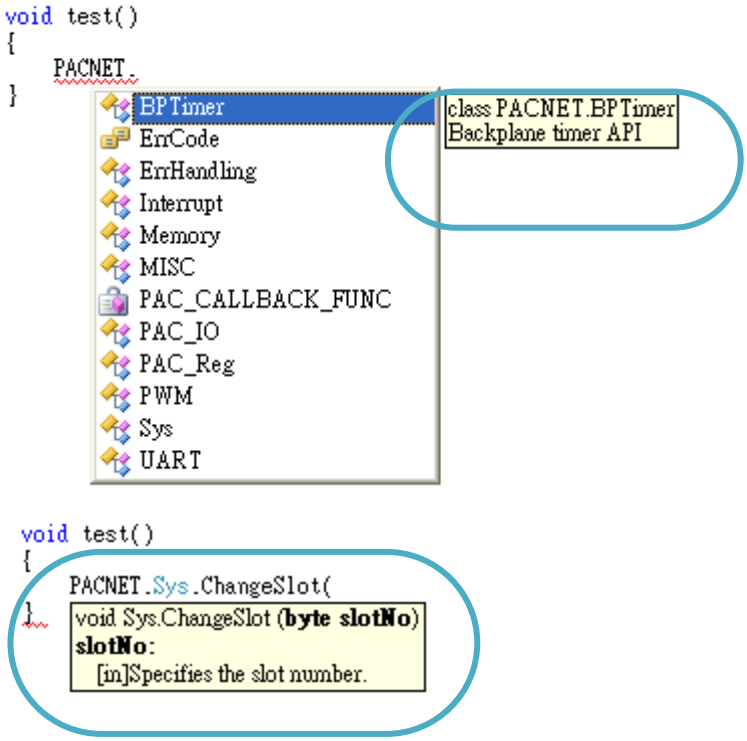
A program applied to WinPAC series modules and the other program applied to XPAC series modules are functionally identical, because of using the same .NET CF library and the API functions and error code on the library are exactly the same, the source code can be shared for two programs.
One shared source code can be built as an executable programs and link the same .NET CF library (PACNET.dll). The only change is that links different platform native SDK. (PACSDK.dll (ARM) is used on WinPAC series and PACSDK.dll(x86) is used for XPAC series)

Notes

PACNET.dll has been developed using the .Net CF V2.0 environment and can be used on all XPAC and WinPAC series devices.

5. Show a tooltip for the classes of PACNET.dll

When developing the programs in VS2005/VS2008 IDE, typing a reference to a system class or namespace or roll over class, the tooltips pop up on your cursor line giving not only the parameters and variables of methods, but also some descriptions for these methods, classes and namespaces. Those description of tooltips are same on the PAC API manual. (Refer to the following figure)



Note

The PACNET.dll referred to the project and PACNET.xml must be placed at the same folder and the tooltip will show in the Visual studio IDE well.

C.3. Error code modifications and updates

1. For WinPAC series

Modify

The error code, PAC_ERR_EEP_ACCESS_RESTRICTION and PAC_ERR_SRAM_INVALID_TYPE defined in WinPacSDK.h are modified as PAC_ERR_EEP_INVALID_ADDRESS and PAC_ERR_MEMORY_INVALID_TYPE defined in PACSDK.h.

Error code (PAC_ERR_MEMORY_BASE + 1)
PAC_ERR_EEP_ACCESS_RESTRICTION
Changed to
PAC_ERR_EEP_INVALID_ADDRESS

Error code (PAC_ERR_MEMORY_BASE + 3)
PAC_ERR_SRAM_INVALID_TYPE
Changed to
PAC_ERR_MEMORY_INVALID_TYPE

Add

```
//Basic
PAC_ERR_MODULE_UNEXISTS                 (PAC_ERR_BASE + 7)
PAC_ERR_INVALID_SLOT_NUMBER             (PAC_ERR_BASE + 8)
//Interrupt
PAC_ERR_INTR_BASE                       0x13000
PAC_ERR_INTR_CREATE_EVENT_FAILURE       (PAC_ERR_INTR_BASE + 1)
//UART
PAC_ERR_UART_INTERNAL_BUFFER_OVERFLOW   (PAC_ERR_UART_BASE+9)
//IO
PAC_ERR_IO_DO_CANNOT_OVERWRITE          (PAC_ERR_IO_BASE+10)
PAC_ERR_IO_AO_CANNOT_OVERWRITE          (PAC_ERR_IO_BASE+11)
```

2. For XPAC series

Modify

The error code,PAC_ERR_INTR_CRATE_EVENT_FAILURE defined in XPacSDK_CE.h is misspelled, and it is corrected in PACSDK.h as PAC_ERR_INTR_CREATE_EVENT_FAILURE

//Interrup

Error code (PAC_ERR_INTR_BASE + 1)

PAC_ERR_INTR_CRATE_EVENT_FAILURE

Changed to

PAC_ERR_INTR_CREATE_EVENT_FAILURE

//Basic

PAC_ERR_MODULE_UNEXISTS

Original Errorcode: PAC_ERR_BASE + 4

Changed to

PAC_ERR_BASE + 7

Add

//Basic

PAC_ERR_INVALID_MAC	(PAC_ERR_BASE + 4)
PAC_ERR_INVALID_COMPORT_NUMBER	(PAC_ERR_BASE + 5)
PAC_ERR_FUNCTION_NOT_SUPPORT	(PAC_ERR_BASE + 6)
PAC_ERR_INVALID_SLOT_NUMBER	(PAC_ERR_BASE + 8)

//Memory Access

PAC_ERR_NVRAM_INVALID_ADDRESS	(PAC_ERR_MEMORY_BASE + 4)
PAC_ERR_EEP_WRITE_PROTECT	(PAC_ERR_MEMORY_BASE + 5)
PAC_ERR_EEP_WRITE_FAIL	(PAC_ERR_MEMORY_BASE + 6)
PAC_ERR_EEP_READ_FAIL	(PAC_ERR_MEMORY_BASE + 7)

//UART

PAC_ERR_UART_INTERNAL_BUFFER_OVERFLOW	(PAC_ERR_UART_BASE+9)

//IO

PAC_ERR_IO_DO_CANNOT_OVERWRITE	(PAC_ERR_IO_BASE+10)
PAC_ERR_IO_AO_CANNOT_OVERWRITE	(PAC_ERR_IO_BASE+11)

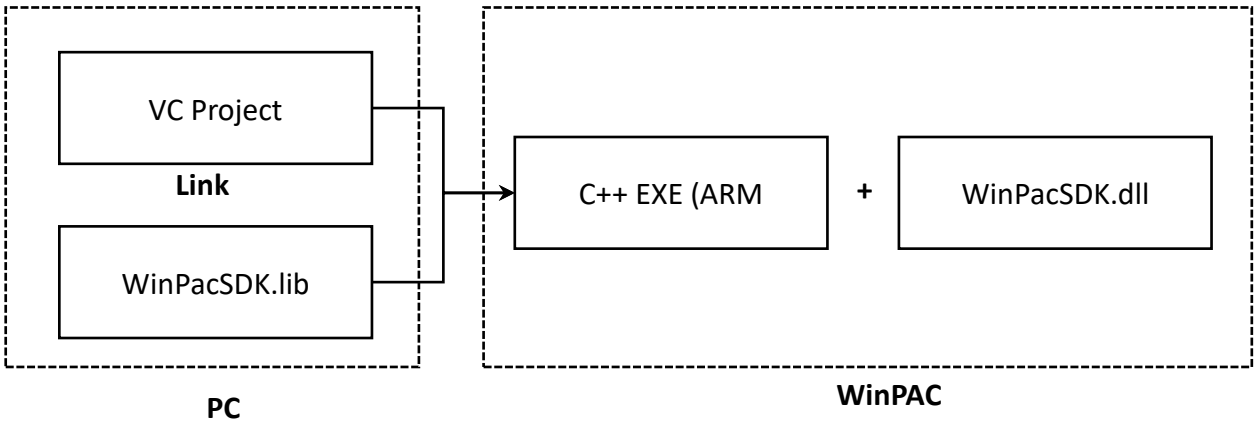
D. Using the Multi-function DCON module

D. 1. On WinPAC devices

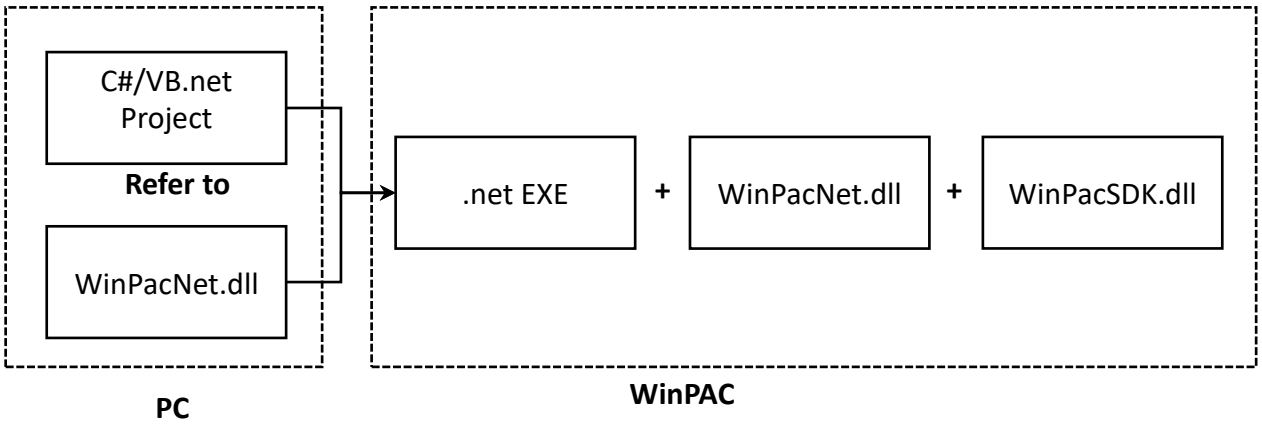
1. The users have used WinPAC series devices and their programs is based on the old SDK (WinPacSDK.dll/WinPacNet.dll) working with the old DCON modules (Note 2) on WinPAC device and without using multi-function DCON modules (Note 1).

   The user's program can continue to use the old library without needing to be modified.
   (The Old SDK **will continue to maintain** (Fix the **bugs**) **and released** regularly, **but will not add new features**)

   Use the old SDK as following flowchart:
   The VC project required to link WinPacSDK.lib while building, and the built executable file placed in the WinPAC series device must work with WinPacSDK.dll



The C#/VB.net project required to refer to WinPacNet.dll while building, and the built executable file placed in the WinPAC series device must work with WinPacNet.dll and WinPacSDK.dll.



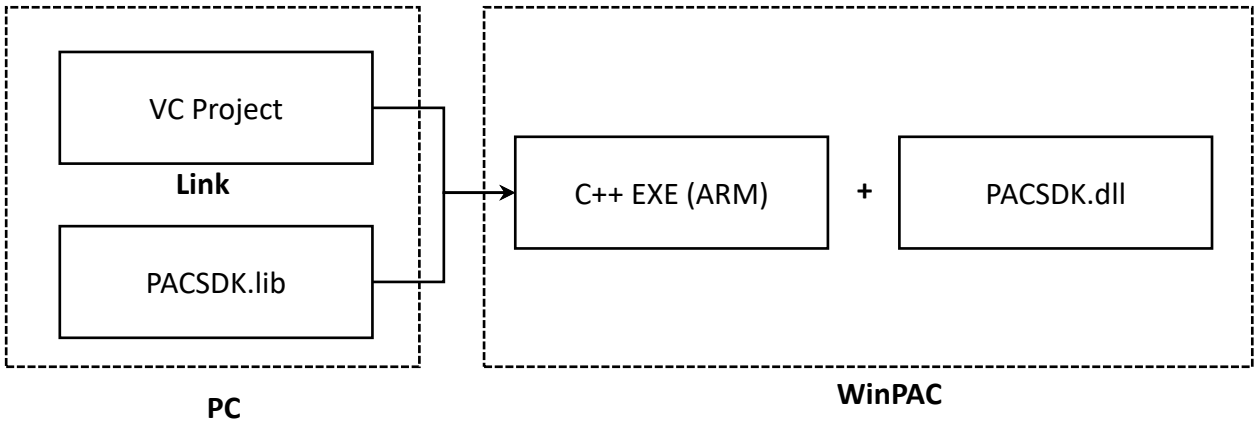
2. The users have used WinPAC series devices and their programs is based on the old SDK (WinPacSDK.dll) working with the old DCON modules and multi-function DCON modules on WinPAC device. The new PACSDK.dll provides pac_xxx_MF API functions that allow access to Multi-function modules, **so the code must be updated in order to use the new PACSDK.dll in the program.**

   (Refer to How-to document, w6-10_How_to_update_to_PACSDK_library_from_WinPacSDK_library_EN.pdf for more details)

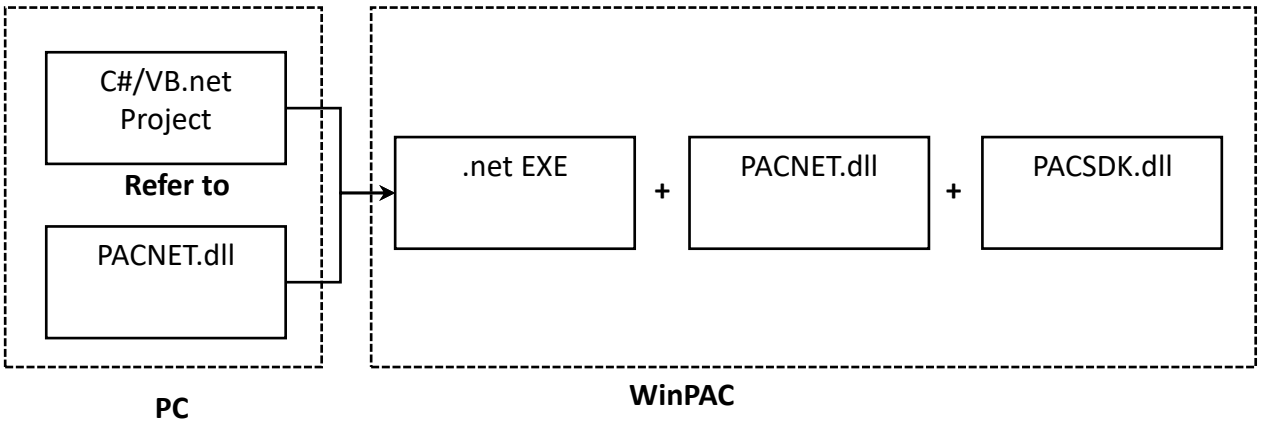
3. The users have never used WinPAC series devices. Their program will be based on the new SDK working with an old DCON module or a Multi-function module. Our API Manual give instructions for the PACSDK.dll and the demo programs included on the shipped CD/FTP are linked with the new PACSDK.dll, so users should refer to the demo programs and follow the API instructions when developing new programs based on the new PACSDK.dll, rather than those for the WinPACSDK.

Use the new SDK as following flowchart:

The VC project required to link PACSDK.lib while building, and the built executable file placed in the WinPAC series device must work with PACSDK.dll



The C#/VB.net project required to refer to PACNET.dll while building, and the built executable file placed in the WinPAC series device must work with PACNET.dll and PACSDK.dll.



Notes

Multi-function DCON modules are defined as modules that mainly act as AIO or Counters but are equipped with DIO channels. Such as the I-87005W/I-87016W/I-87082W/I-7016/I-7088, etc.)

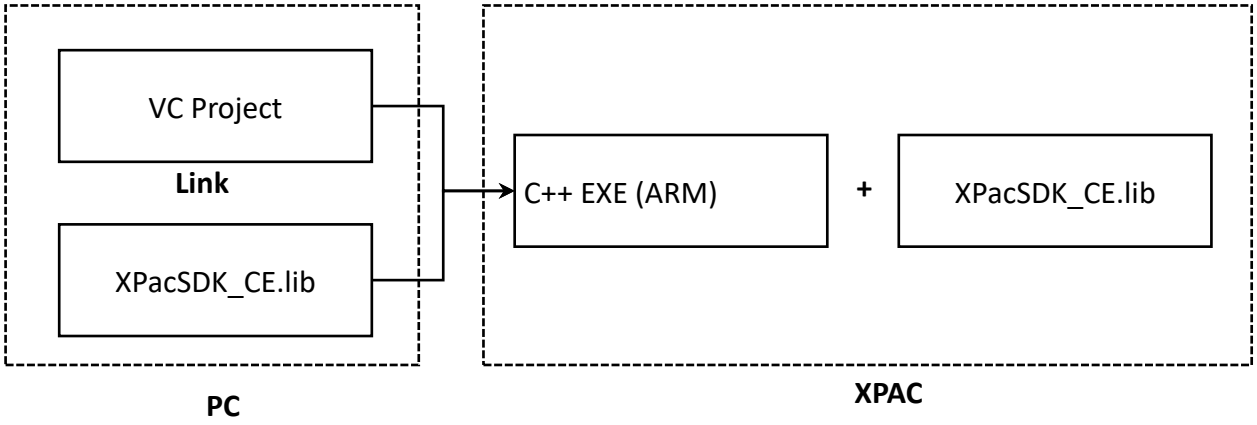
Old DCON module definition: Non multi-function DCON modules are defined as Old DCON modules.

D.2. On XPAC devices

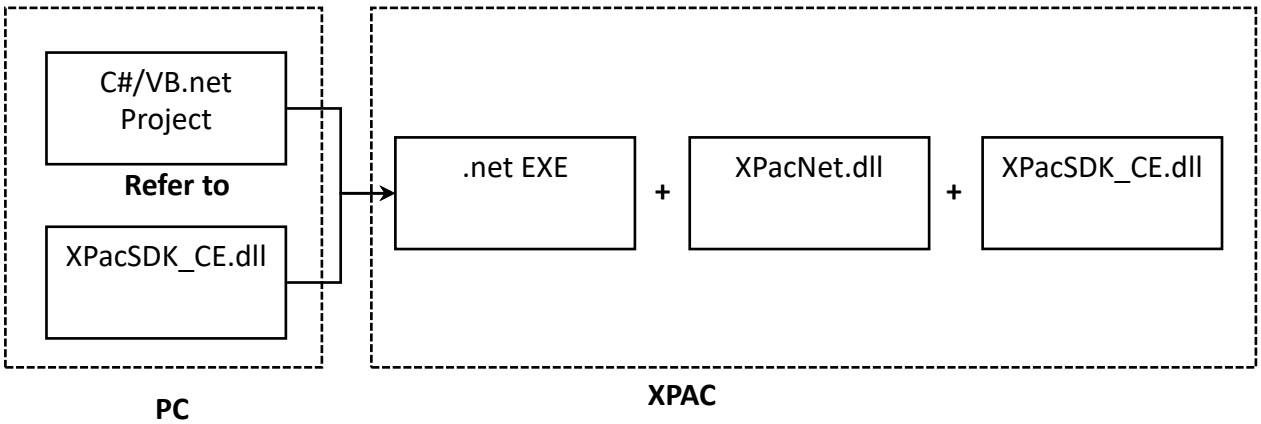
1. The users have used XPAC series devices and their programs is based on the old SDK (XPacSDK_CE.dll/XPacNet.dll) working with the old DCON modules (*Note 2*) on XPAC device and without using multi-function DCON modules (*Note 1*).

   The user's program can continue to use the old library without needing to be modified.
   (The Old SDK **will continue to maintain** (Fix the b**ugs) and released regularly**, **but will not add new features)**

   Use the old SDK as following flowchart:
   The VC project required to link XPacSDK_CE.lib while building, and the built executable file placed in the XPAC series device must work with XPacSDK_CE.dll



   The C#/VB.net project required to refer to XPacNet.dll while building, and the built executable file placed in the XPAC series device must work with XPacNet.dll and XPacSDK_CE.dll.

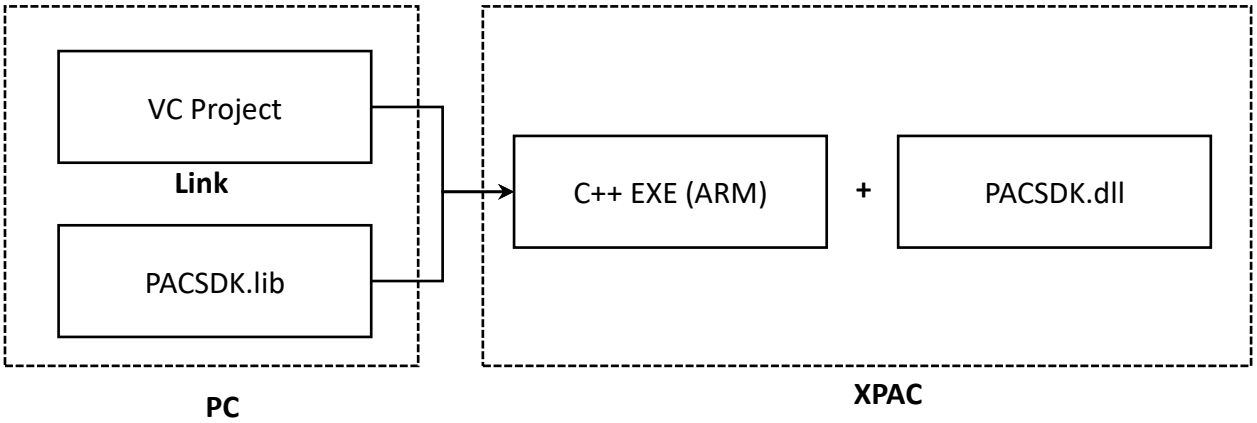


2. The users have used XPAC series devices and their programs is based on the old SDK (XPacSDK_CE.dll) working with the old DCON modules and multi-function DCON modules on XPAC device. The new PACSDK.dll provides pac_xxx_MF API functions that allow access to Multi-function modules, **so the code must be updated in order to use the new PACSDK.dll in the program.** (Refer to How-to document, x6-10_How_to_update_to_PACSDK_library_from_XPacSDK_library_EN.pdf for more details)

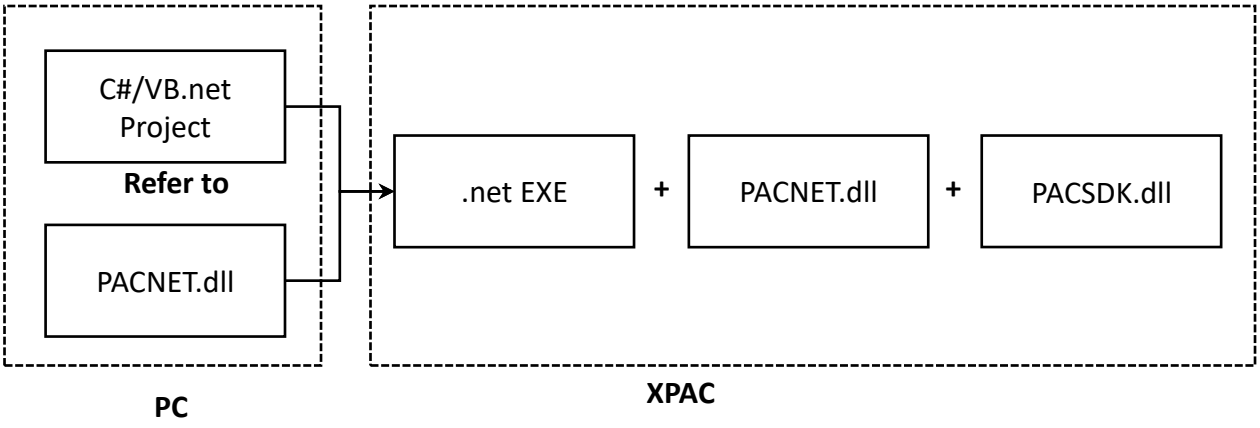
3. The users have never used XPAC series devices. Their program will be based on the new SDK working with an old DCON module or a Multi-function module. Our API Manual give instructions for the PACSDK.dll and the demo programs included on the shipped CD/FTP are linked with the new PACSDK.dll, so users should refer to the demo programs and follow the API instructions when developing new programs based on the new PACSDK.dll, rather than those for the XPACSDK.

Use the new SDK as following flowchart:

The VC project required to link PACSDK.lib while building, and the built executable file placed in the XPAC series device must work with PACSDK.dll



The C#/VB.net project required to refer to PACNET.dll while building, and the built executable file placed in the XPAC series device must work with PACNET.dll and PACSDK.dll.



E. How to update to the PACSDK Library from the WinPACSDK/XPACSDK Library

Questions related to updating the PACSDK library from the WinPacSDK library and the solutions

Refer to w6-10_How_to_update_to_PACSDK_library_from_WinPacSDK_library_en.pdf located at http://ftp.icpdas.com/pub/cd/winpac/napdos/wp-8x4x_ce50/document/faq/sdk/w6-010_how_to_update_to_pacsdk_library_from_winpacsdk_library_en.pdf

Questions related to updating to PACSDK library from the XPacSDK library and solutions

Refer to x6-10_How_to_update_to_PACSDK_library_from_XPacSDK_library_en.pdf located at http://ftp.icpdas.com/pub/cd/xpac-atom-ce6/document/faq/sdk/x6-10_how_to_update_to_pacsdk_library_from_xpacsdk_library_en.pdf

F. Comparison of Defined Slots and COM Ports

Each PAC has its own definition and corresponding communication ports and slots whose parameters are defined as below.

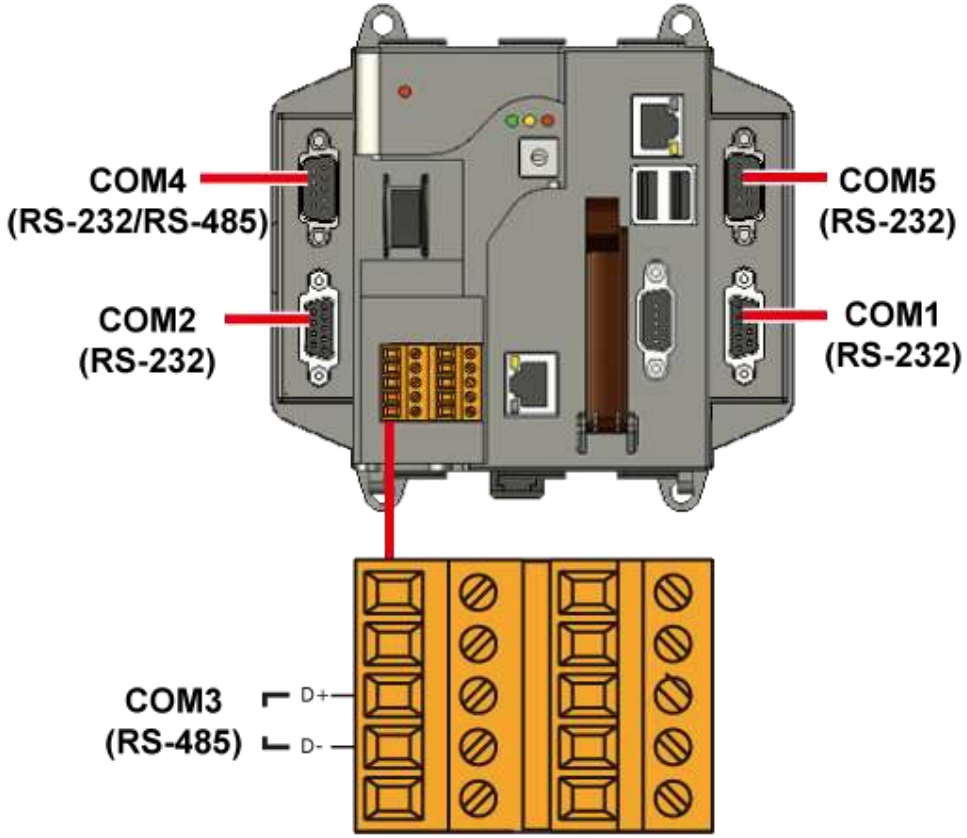
As a result, apply the corresponding slot and COM port number on the API functions in writing program.

<table>
<tr><th></th><th>Slot number</th><th>COM0</th><th>COM1</th><th>COM2</th><th>COM3</th><th>COM4</th><th>COM5</th></tr>
<tr><td>XP-8131-CE6</td><td>1 (1)</td><td rowspan="3">-</td><td rowspan="3">Backplane</td><td rowspan="3">RS-232</td><td rowspan="3">RS-485</td><td rowspan="3">RS-232/RS-485</td><td rowspan="3">RS-232</td></tr>
<tr><td>XP-8331-CE6</td><td>3 (1 ~ 3)</td></tr>
<tr><td>XP-8731-CE6</td><td>7 (1 ~ 7)</td></tr>
<tr><td>XP-8041-CE6</td><td>0</td><td rowspan="3">-</td><td>RS-232</td><td rowspan="3">RS-232</td><td rowspan="3">RS-485</td><td rowspan="3">RS-232/RS-485</td><td rowspan="3">RS-232</td></tr>
<tr><td>XP-8341-CE6</td><td>3 (1 ~ 3)</td><td rowspan="2">Backplane</td></tr>
<tr><td>XP-8741-CE6</td><td>7 (1 ~ 7)</td></tr>
<tr><td>XP-8141-Atom-CE6</td><td>1 (1)</td><td rowspan="3">-</td><td rowspan="3">Backplane</td><td rowspan="3">RS-232</td><td rowspan="3">RS-485</td><td rowspan="3">RS-232/RS-485</td><td rowspan="3">RS-232</td></tr>
<tr><td>XP-8341-Atom-CE6</td><td>3 (1 ~ 3)</td></tr>
<tr><td>XP-8741-Atom-CE6</td><td>7 (1 ~ 7)</td></tr>
<tr><td>WP-9221-CE7</td><td>2(0 ~ 1)</td><td rowspan="3">Backplane</td><td rowspan="3">RS-232/RS-485</td><td rowspan="3">RS-485</td><td rowspan="3">RS-232/RS-485</td><td rowspan="3">RS-232</td><td rowspan="3">-</td></tr>
<tr><td>WP-9421-CE7</td><td>4 (0 ~ 3)</td></tr>
<tr><td>WP-9821-CE7</td><td>8 (0 ~ 7)</td></tr>
<tr><td>WP-8121-CE7</td><td>1 (0)</td><td rowspan="3">Backplane</td><td rowspan="3">RS-232</td><td rowspan="3">RS-485</td><td>-</td><td>-</td><td rowspan="3">-</td></tr>
<tr><td>WP-8421-CE7</td><td>4 (0 ~ 3)</td><td rowspan="2">RS-232/RS-485</td><td rowspan="2">RS-232</td></tr>
<tr><td>WP-8821-CE7</td><td>8 (0 ~ 7)</td></tr>
</table>

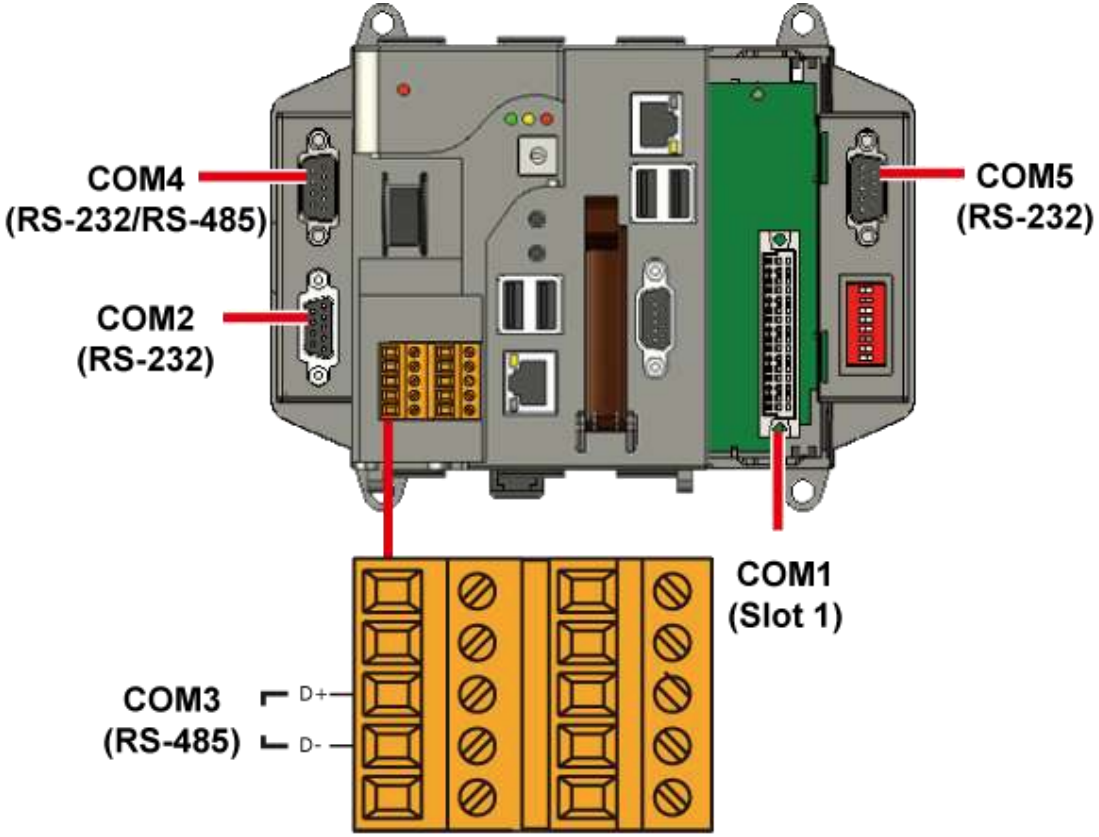
	Slot number	COM0	COM1	COM2	COM3	COM4	COM5
WP-8131	1 (0)	Backplane	RS-232	RS-485	-	-	-
WP-8431	4 (0 ~ 3)				RS-232/RS-485	RS-232	
WP-8831	8 (0 ~ 7)						
WP-8141	1 (0)	Backplane	RS-232	RS-485	-	-	-
WP-8441	4 (0 ~ 3)				RS-232/RS-485	RS-232	
WP-8841	8 (0 ~ 7)						
WP-5141/WP-5141-OD	-	-	RS-232	RS-485	RS-232	-	-
WP-5151/WP-5151-OD			RS-485				
WP-5231-CE7 WP-5231M-CE7 WP-5231PM-3GWA-CE7 WP-5231PM-4GE-CE7 WP-5231PM-4GC-CE7		XVboard	RS-232	RS-232	RS-485	RS-485	
VP-23W1	3 (0 ~ 2)	Backplane	-	RS-485	RS-232	-	-
VP-25W1							
VP-4131							
VP-1231-CE7	3 (0 ~ 2)	Backplane	-	RS-485	RS-232	-	-
VP-4231-CE7							
VP-6231-CE7							

Backplane: It’s RS232 interface used for accessing the I-87K module only.

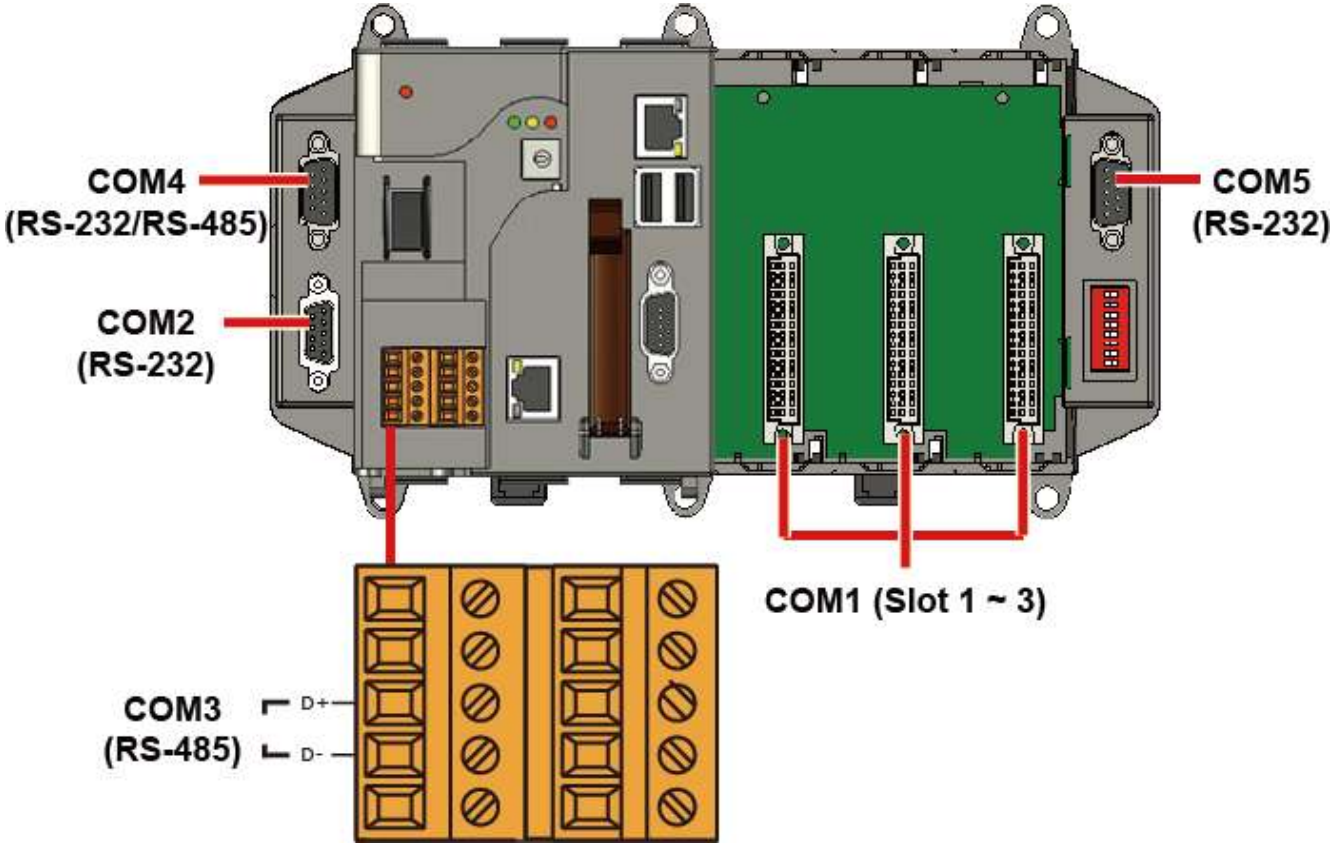
F.1. XP-8041-CE6



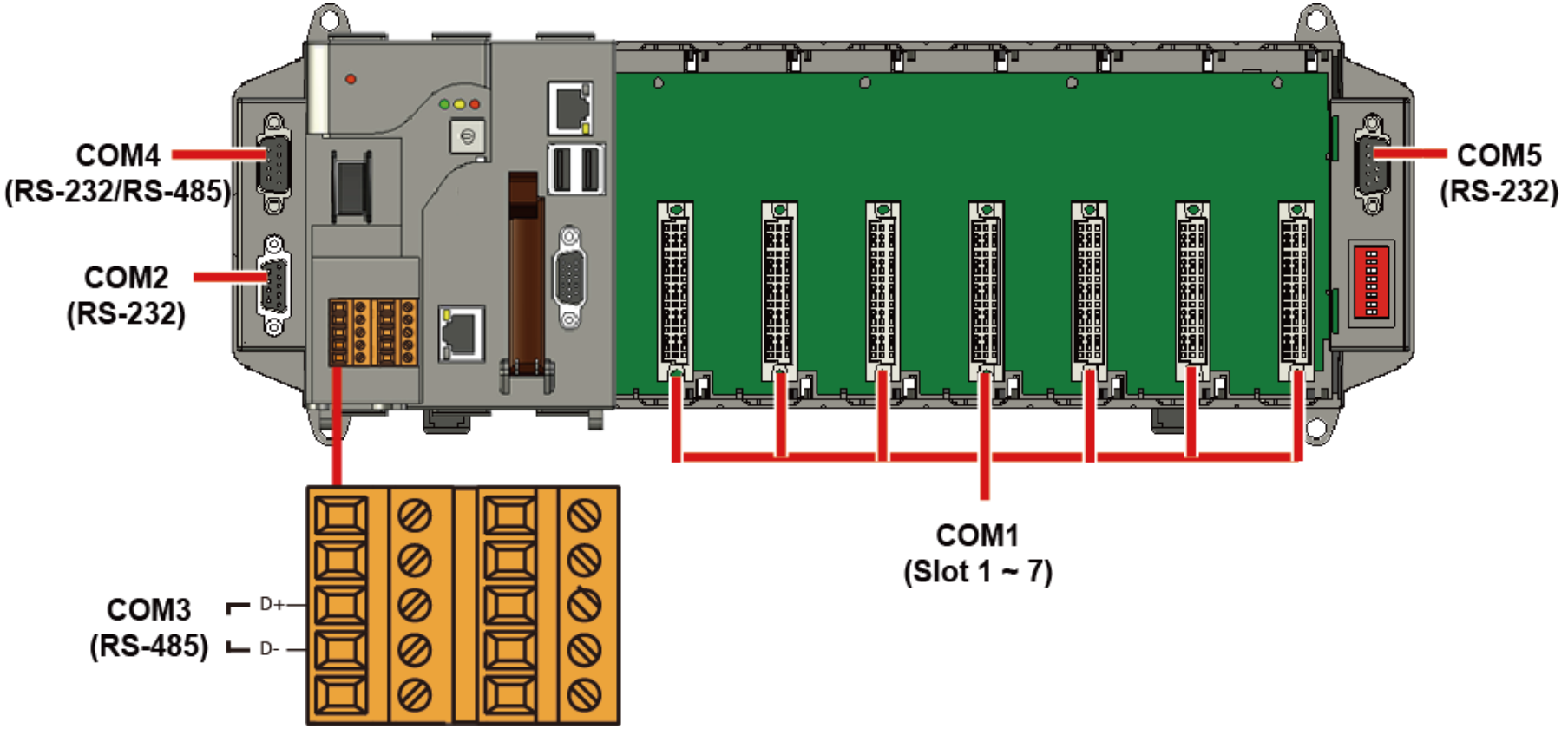
F.2. XP-8131-CE6/XP-8141-Atom-CE6



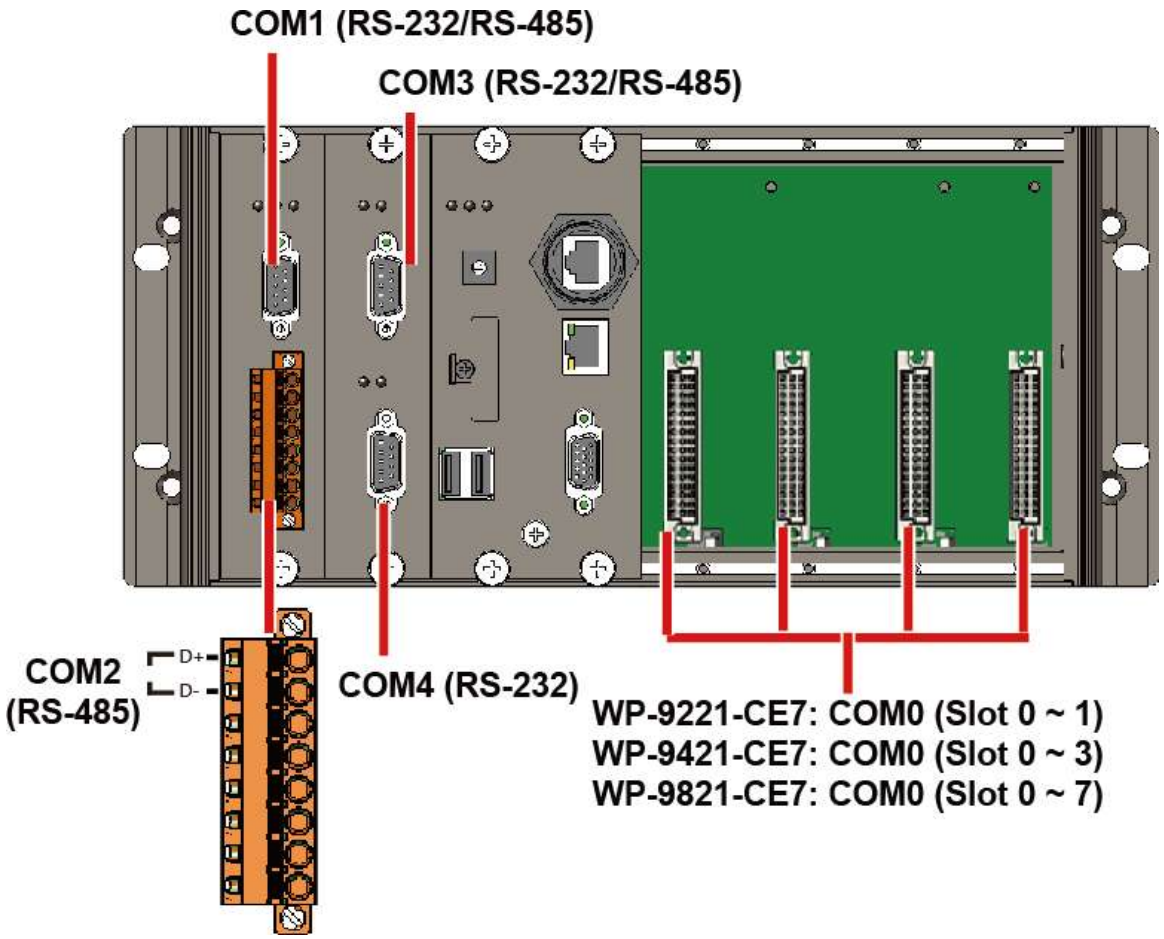
F.3. XP-8331-CE6/XP-8341-CE6/XP-8341-Atom-CE6



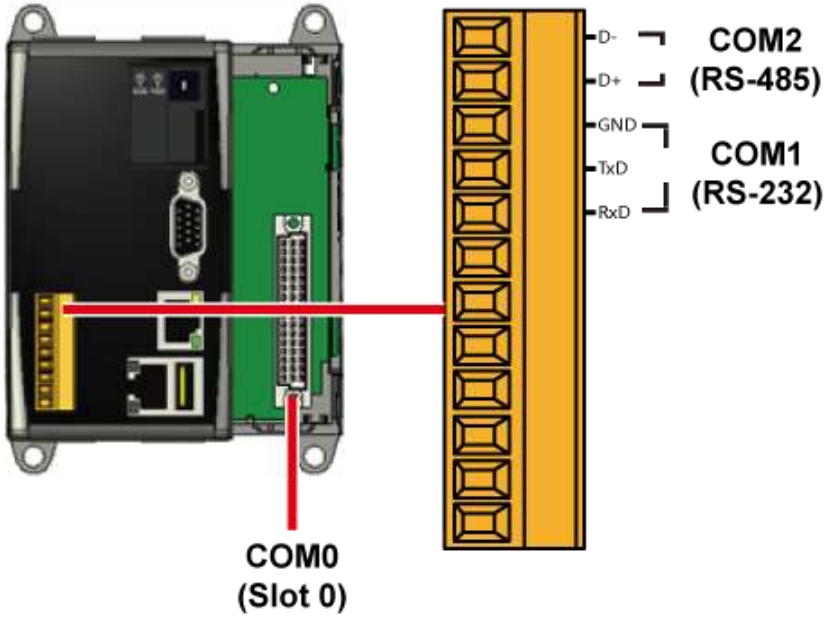
F.4. XP-8731-CE6/XP-8741-CE6/XP-8741-Atom-CE6



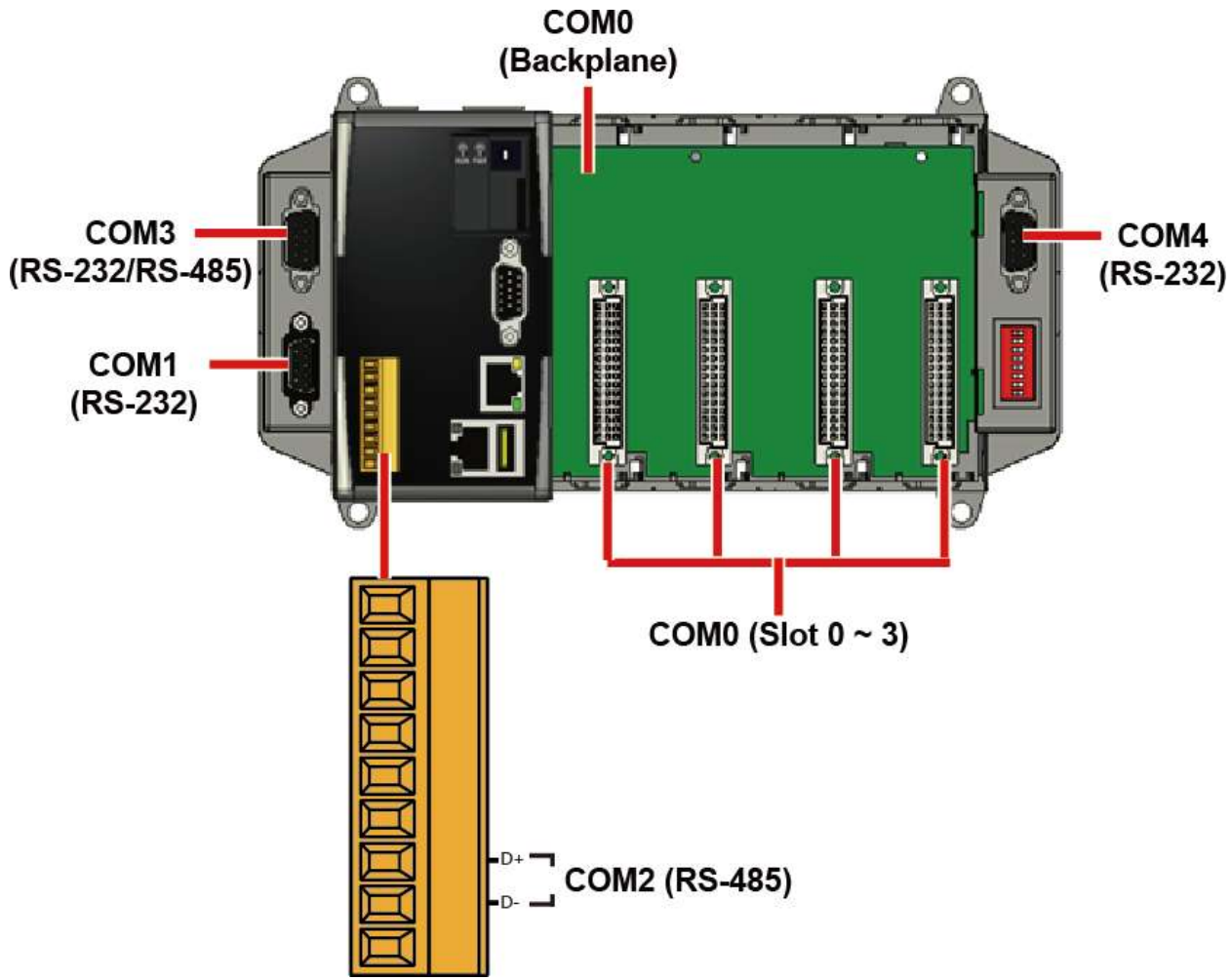
F.5. WP-9221-CE7/WP-9421-CE7/WP-9821-CE7



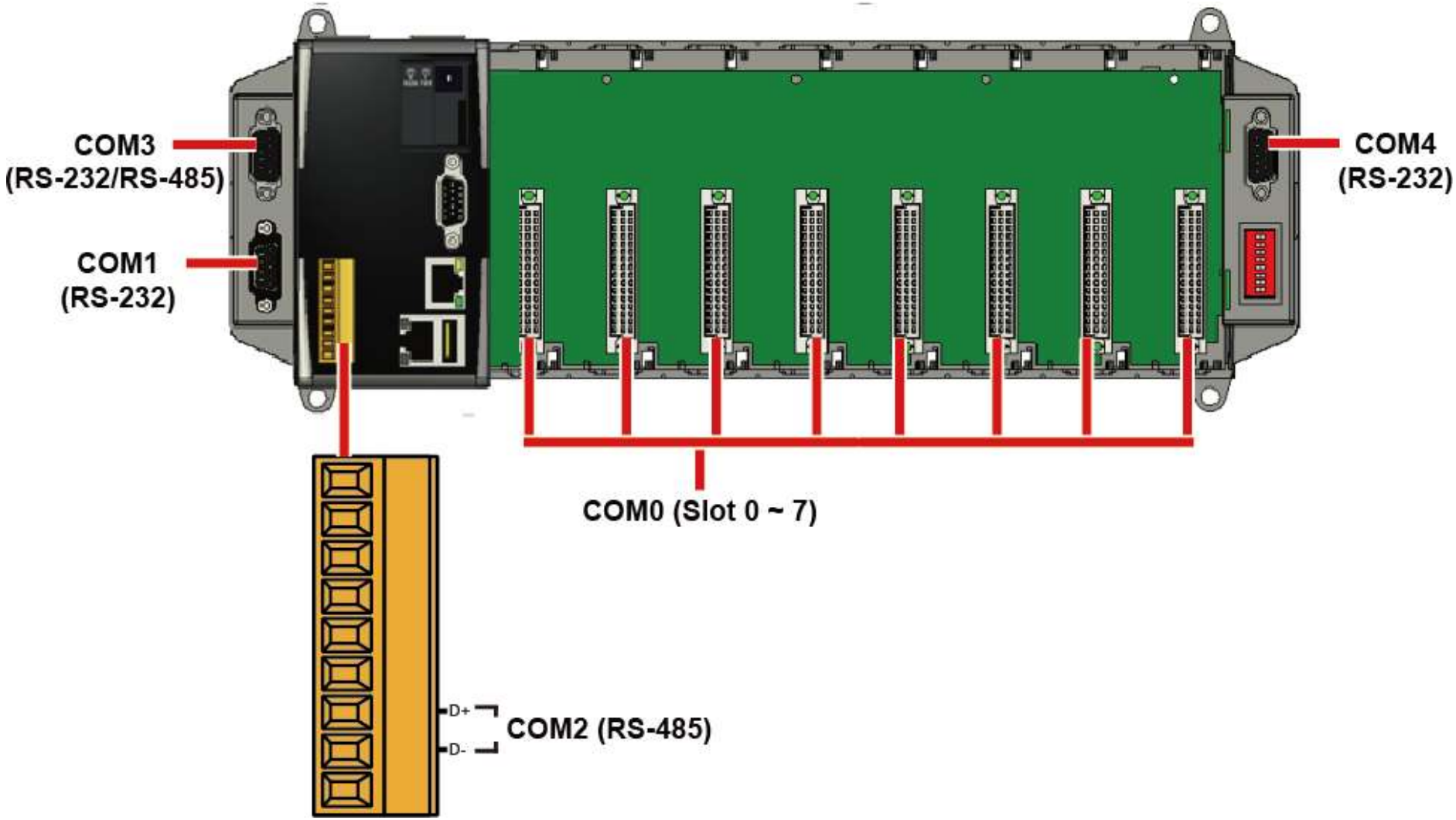
F.6. WP-8121-CE7/WP-8131/WP-8141



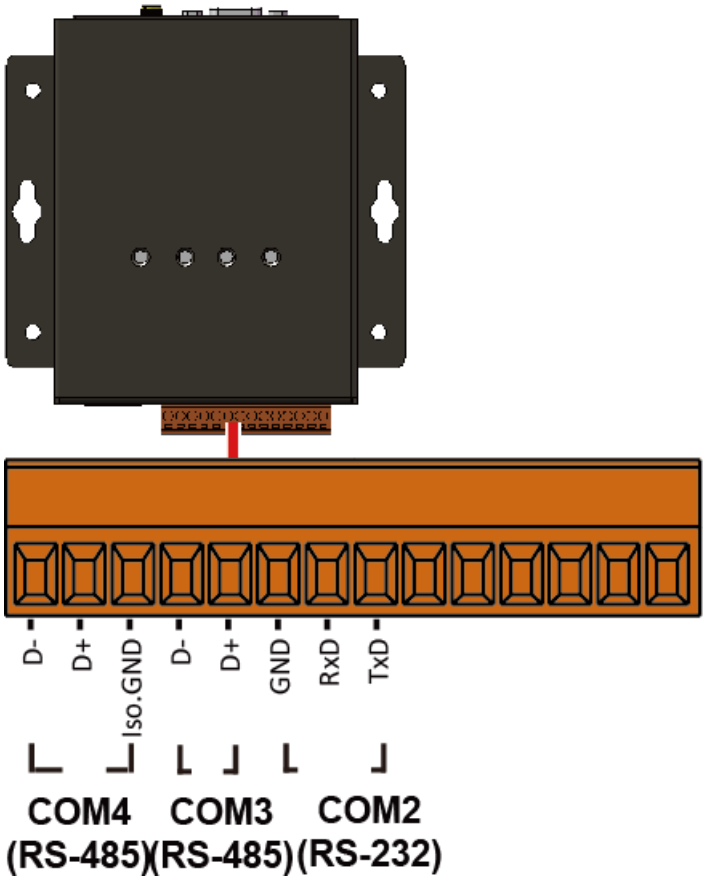
F.7. WP-8421-CE7/WP-8431/WP-8441



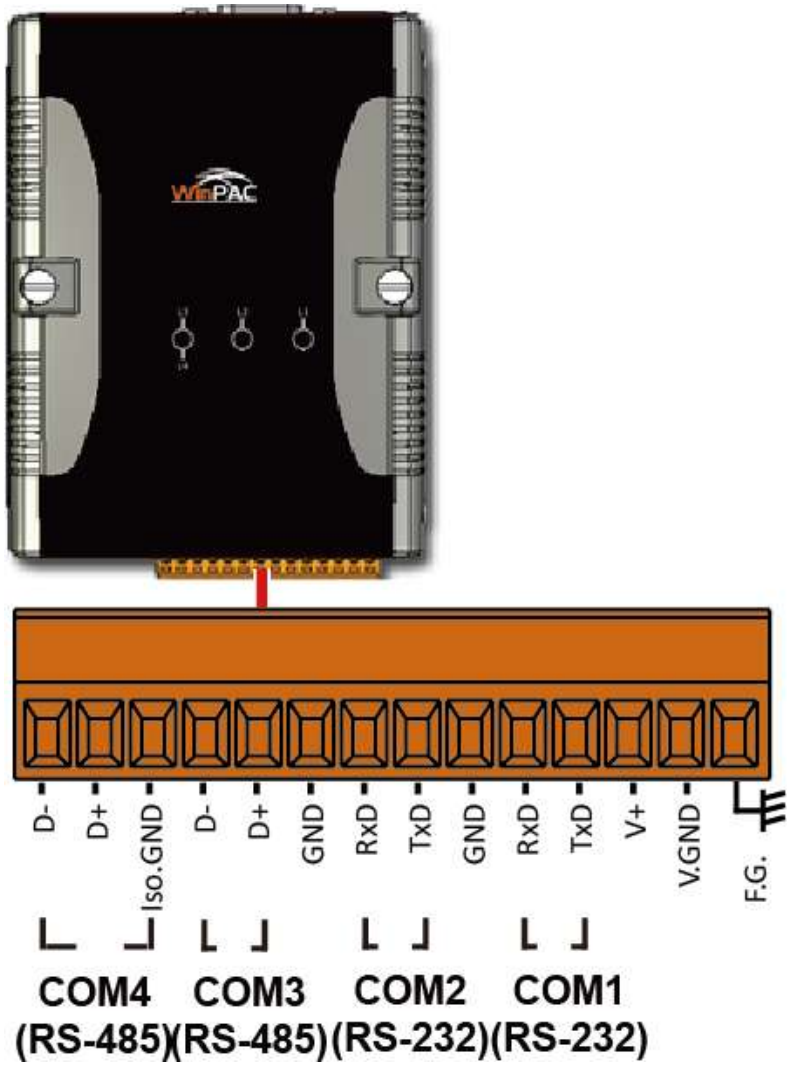
F.8. WP-8821-CE7/WP-8831/WP-8841



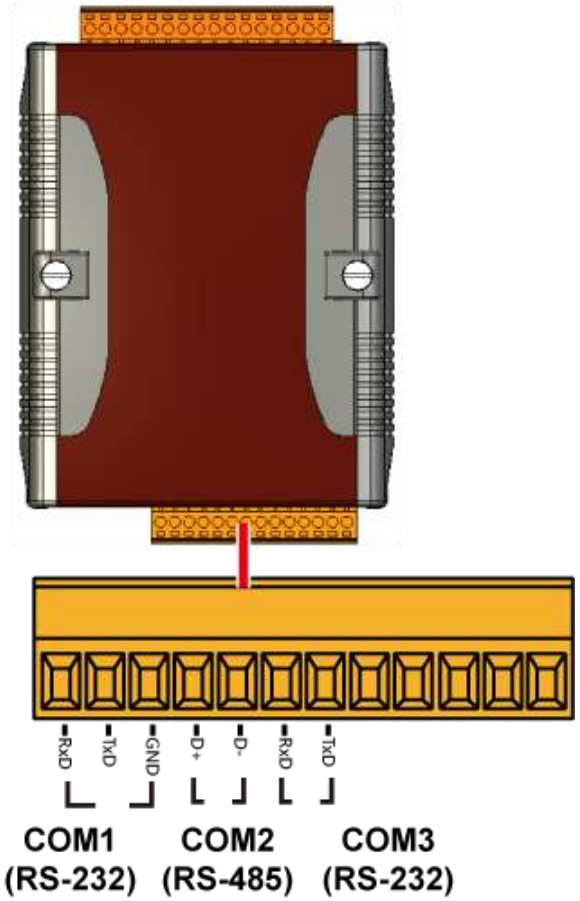
F.9. WP-5231M-CE7/WP-5231PM-3GWA-CE7/WP-5231PM-4GE-CE7/WP-5231PM-4GC-CE7



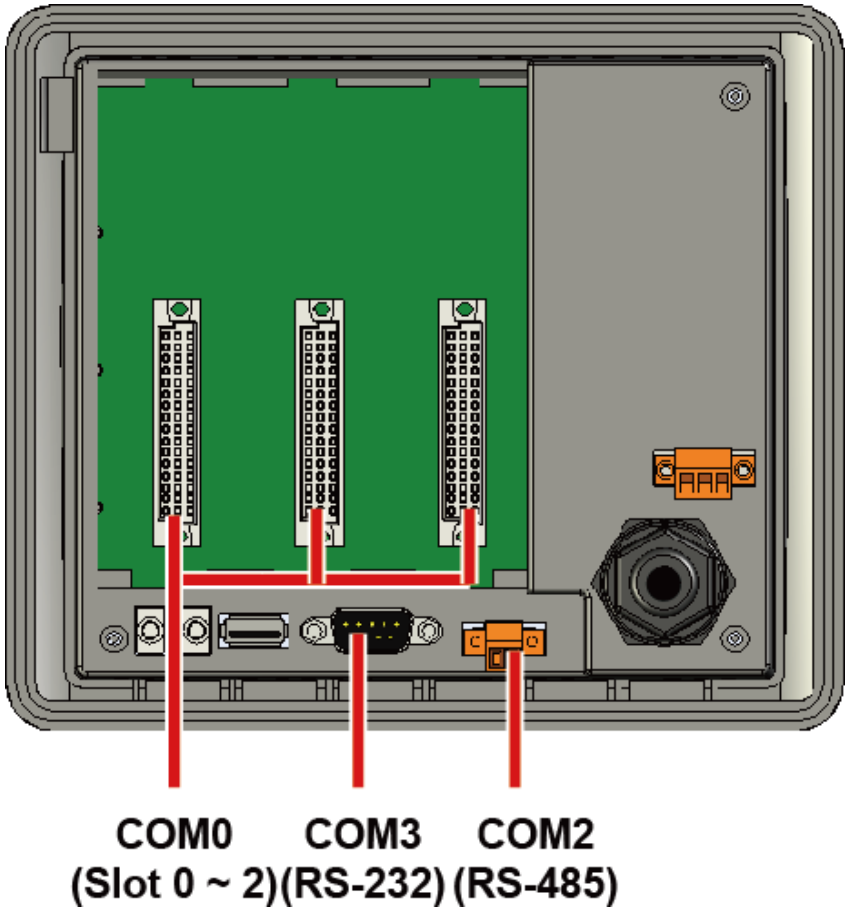
F.10. WP-5231-CE7



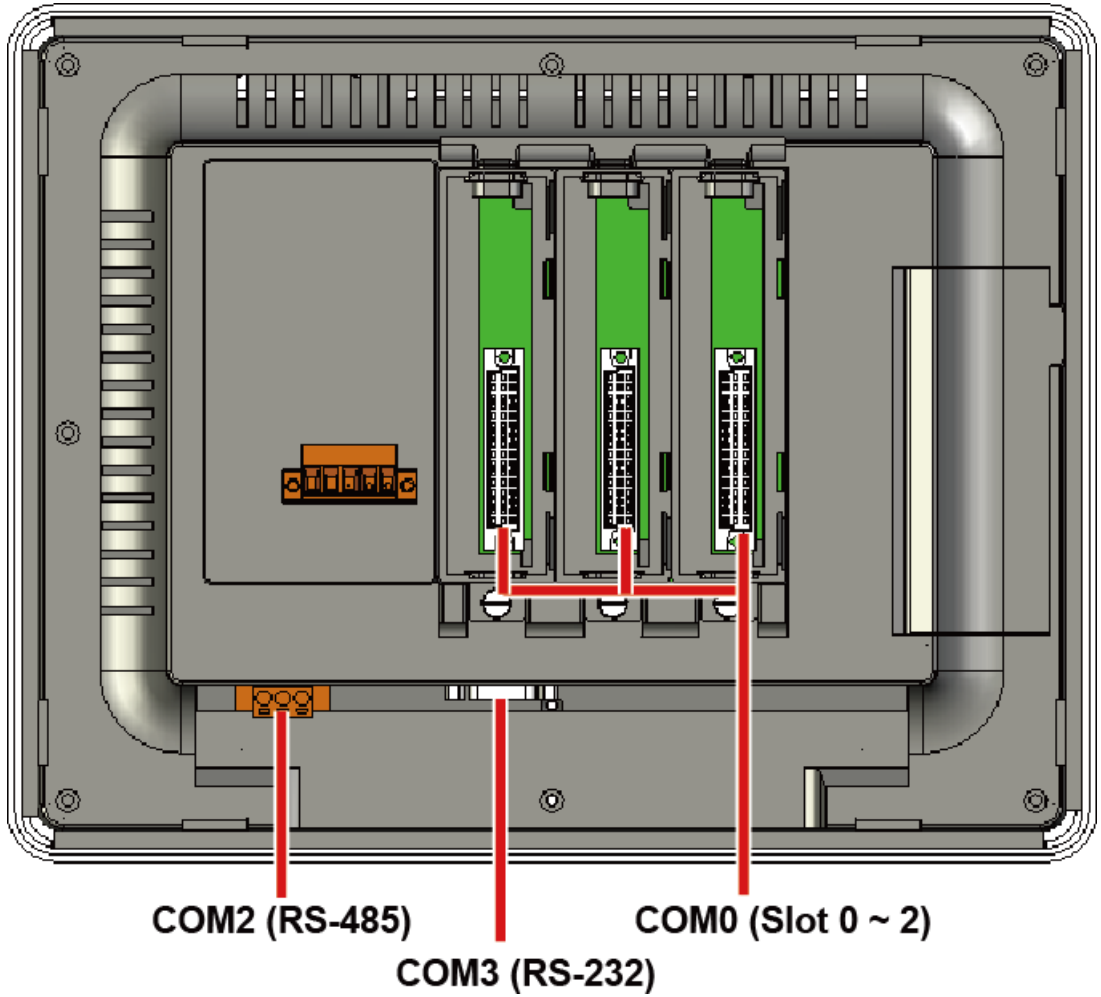
F.11. WP-5141/WP-5141-OD/WP-5151/WP-5151OD



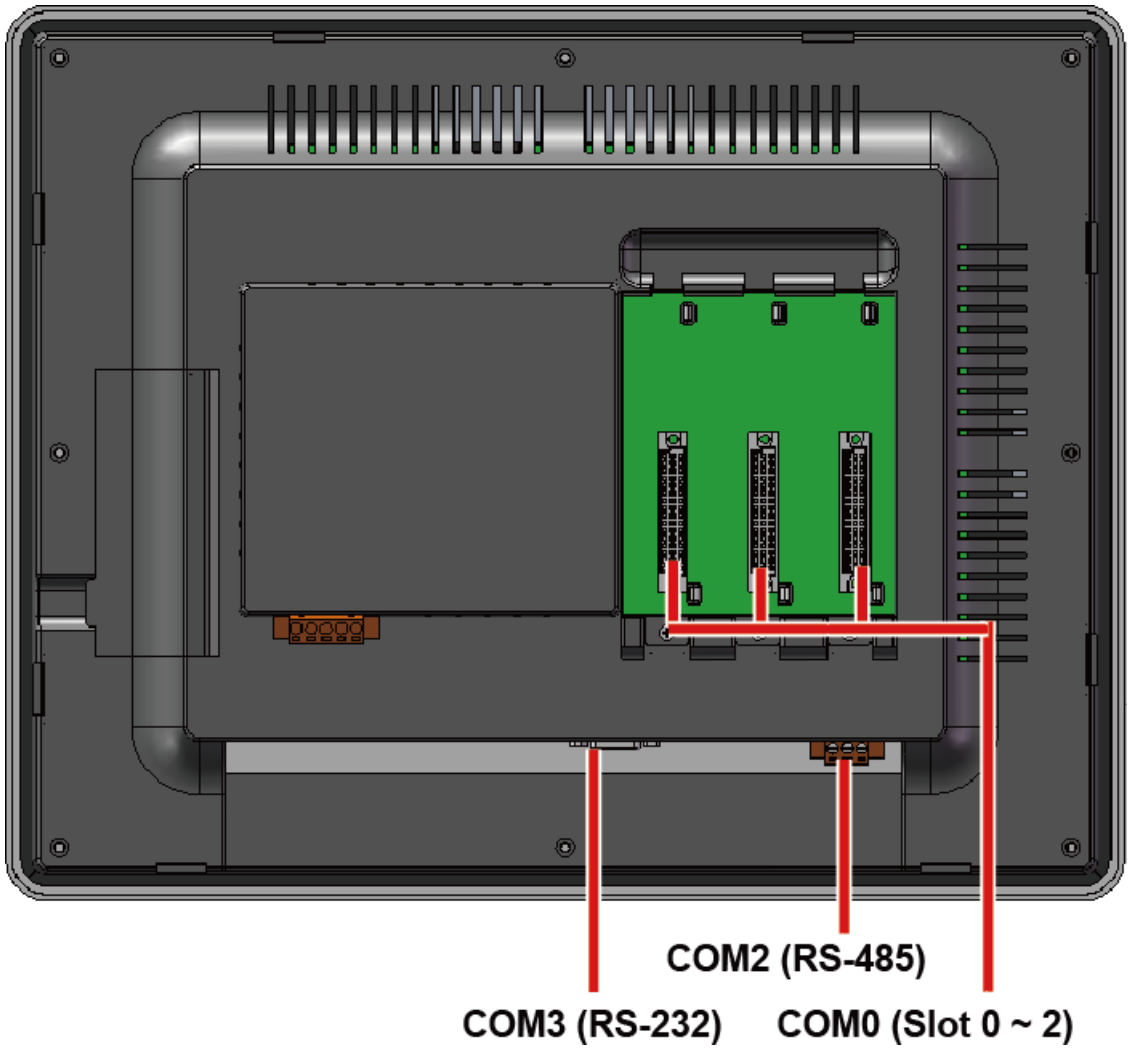
F.12. VP-1231-CE7



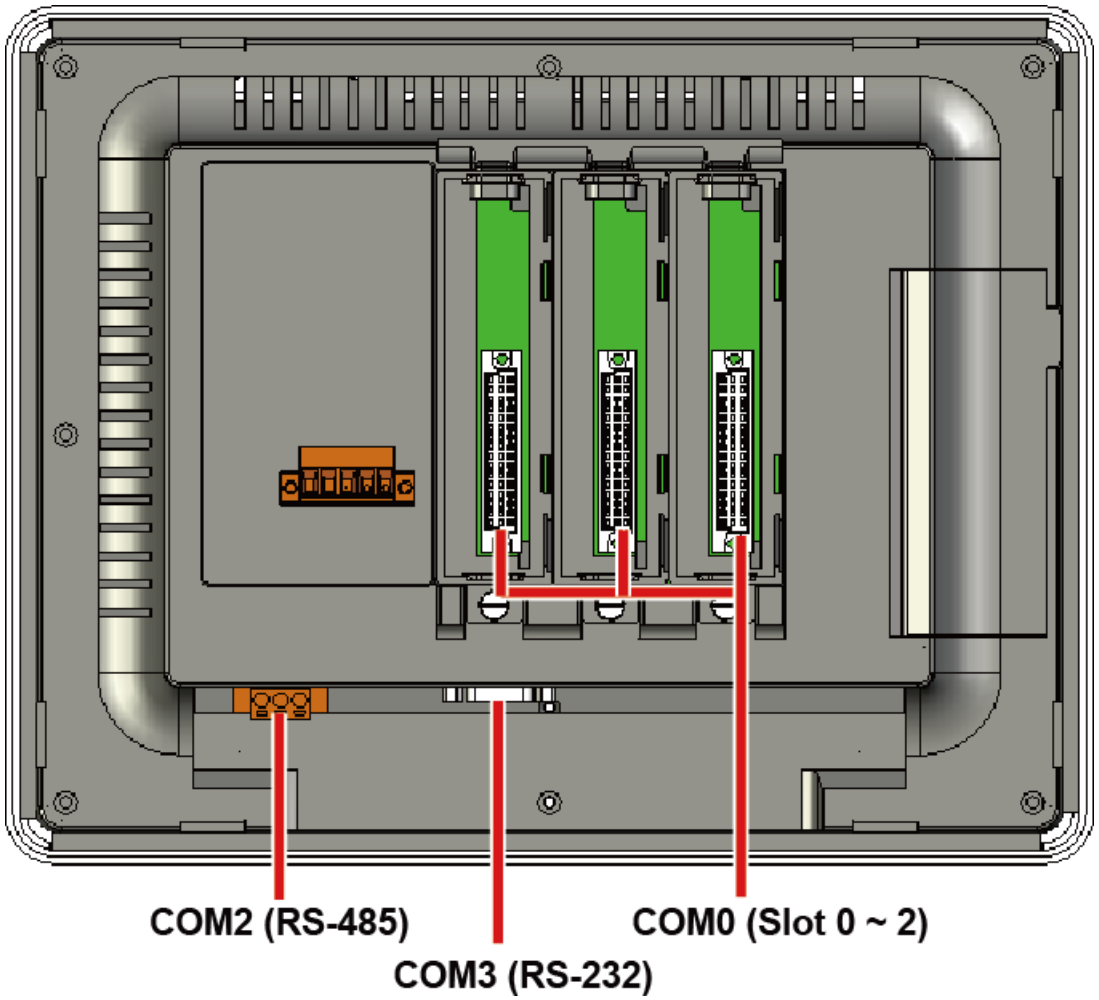
F.13. VP-4231-CE7



F.14. VP-6231-CE7



F.15. VP-4131



F.16. VP-23W1/VP-25W1