WinCE-Based XPAC/WinPAC

標準 API 使用者手冊

版本 1.3.2, 2024 9 月

產品技術服務與使用資訊

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

Written by Sean

Edited by Anna Huang

保固說明

泓格科技股份有限公司 (ICP DAS) 所生產的產品，均保證原始購買者對於有瑕疵之材 料，於交貨日起保有為期一年的保固。

免責聲明

泓格科技股份有限公司對於因為應用本產品所造成的損害並不負任何法律上的責任。本公司保留有任何時間未經通知即可變更與修改本文件內容之權利。本文所含 資訊如有變更，恕不予另行通知。本公司盡可能地提供正確與可靠的資訊，但不保 證此資訊的使用或其他團體在違反專利或權利下使用。此處包涵的技術或編輯錯誤、遺漏，概不負其法律責任。

版權所有

商標識別

本文件提到的所有公司商標、商標名稱及產品名稱分別屬於該商標或名稱的擁有者 所有於識別的名稱只可登記其各自公司的商標。

聯絡我們

如果您有任何問題，請隨時與我們聯繫。
我們將盡速為您服務

Email: service@icpdas.com

目錄

1. 關於手冊

本參考手冊提供 PAC API 的安裝及各個函數庫的使用和參考說明。

支援的 PAC 產品型號

本參考手冊支援的 PAC 產品型號。

WinPAC 系列(AM335x 平台系列)

WP-9000 系列 (金屬外殼)	
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 系列	
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 系列	
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 系列(AM335x 平台系列)

ViewPAC 系列	
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 系列 (x86 平台系列)

XP-8x4x-CE 系列	
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 系列	
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 系列	
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 系列(PXA270 平台系列)

WP-8000 系列	
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 系列	
WP-5141	WinCE 5.0 Based palm-sized WinPAC
WP-5141-OD	WinCE 5.0 Based palm-sized WinPAC with Audio
ViewPAC 系列	
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

相關信息

您可以從 CD 上獲得的 PAC 額外的信息或由泓格科技網站下載最新的版本。

WinPAC 系列(AM335x 平台系列)

WP-9000 系列

CD:\WinPAC_AM335x\wp-9000

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

WP-8000 系列

CD:\WinPAC_AM335x\wp-8x2x

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

WP-5000 series:

CD:\WinPAC_AM335x\Wp-5231

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

ViewPAC series:

CD:\WinPAC_AM335x\VP-x231

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

CD:\WinPAC_AM335x\VP-x201

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

XPAC 系列 (x86 平台系列)

XP-8x4x-CE 系列

CD:\XP-8000-CE6\Document\

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

XP-8x4x-Atom-CE6 系列

CD:\XPAC-ATOM-CE6\Document\

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

XP-8x3x-CE6 系列

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

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

WinPAC 系列(PXA270 平台系列)

WP-8000 系列

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

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

WP-5000 系列

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

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

ViewPAC 系列

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

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

如何聯絡我們?

有關任何泓格產品的支援服務，請用以下方式聯繫泓格科技客戶支援:

- 訪問 ICP DAS technical support 網站
  https://www.icpdas.com/tw/faq/index.php?model=WP-8421-CE7
- 從我們的網站提交問題管理記錄(PMR)
  http://www.icpdas.com/sevices/contact_customerservice.htm
- 發送電子郵件至 service@icpdas.com

修訂

歷史表下面顯示的修訂歷史。

修訂	日期	說明
1.0.1	2012/08	英文版首次發行
1.1.0	2013/01	附錄 F 增加 PAC slot 和 COM port 定義及相關訊息
1.1.1	2013/12	修改底板定時器 API 函數支持。 修改錯誤處理 API 函數支持。 添加 pac_GetModuleLastOutputSource/pac_GetModuleWDTStatus 功能。
1.2.0	2014/06	增加 WinCE7.0 Based WinPAC 支援 增加 2 個 2 API 函數(pac_GetBackLight/pac_SetBackLight)
1.3.0	2017/09	中文版首次發行 (版本號碼與英文版相同)
1.3.1	2021/04	修正範例程式碼錯誤內容
1.3.2	2024/09	增加 pac_ReadDICNTAll /pac_ClearDICNTAll 函數

2. 入門

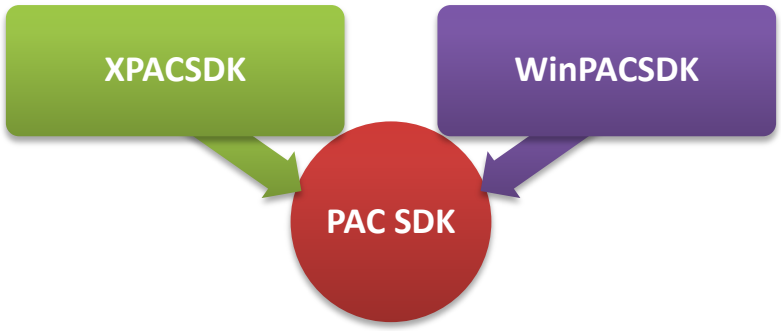
本章提供一個導覽方式介紹首次使用 PACSDK 時，需要了解的基本程序有關程式下載，安裝和配置等步驟。

2.1. PACSDK 介紹

PACSDK 是一個 XPAC，WinPAC 與 ViewPAC 系列平台上使用的應用程序軟件開發工具包，內含頭文件，函數庫，說明文件及相關工具等。

PACSDK 已經取代 XPACSDK 和 WinPACSDK

泓格科技已經發布了新的 SDK(PACSDK)，其中合併並取代了舊的 XPACSDK 和 WinPACSDK。



XPAC/WinPAC 上的 SDK 函數庫統一名稱為 PACSDK，新 PACSDK.dll 提供兩個平台，一個是專為 WinPAC 上系列 (ARM 平台)和另一個用於 XPAC 系列 (x86 支持平台)。XPAC 系列 XPAC 系列平台上，PACSDK.dll (x86)用於鏈接到 C 語言開發的程序以取代以往的 SDK (XPACSDK_CE.dll)。而 WinPAC 系列台上 PACSDK.dll (ARM)用於鏈接到 C 開發的程序以取代以往的 SDK (WinPACSDK.dll)。PACNET.dll 用於.NET CF 程序(C#, VB)，可用於 XPAC 和 WinPAC 系列平台上，以取代之前的.NET SDK (XPacNet.dll 和 WinPacNet.dll) 兩個函數庫。

新/舊 SDK 文件比較

項目	WinPACSDK 函數庫	XPACSDK(CE6)函數庫	PACSDK 函數庫
發展的頭文件	WinPacSDK.h	XPacSDK_CE.h	PACSDK.h PACSDK_PWM.h
開發庫文件	WinPacSDK.lib	XPacSDK_CE.lib	PACSDK.lib PACSDK_PWM.lib
目標設備原生的 DLL 文件	WinPacSDK.dll	XPacSDK_CE.dll	PACSDK PACSDK_PWM.dll
目標設備 NET CF 的 DLL 文件	WinPacNet.dll	XpacNet.dll	PACNET.dll

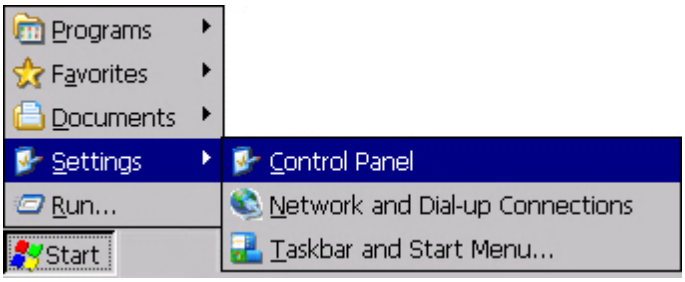
PACSDK 的優點包括

- 易於移植 WinPAC 上的程式至 XPAC 系列
- 易於移植 XPAC 程序至 WP 系列

一套 PACSDK 的 API 幾乎與舊的 SDK (WinPACSDK.dll 和 XPACSDK_CE.dll) 相同，但也有一些修改和更新。請參考 附錄 C. 了解更多的細節。

2.2. 安裝 PACSDK

XPAC/WinPAC 上平台的 SDK 安裝包，它支持 PACSDK 程式庫，可讓用戶開發 XPAC 和 WinPAC 系列的應用程序。



安裝新版的安裝包

1. 獲取 WinPAC 上/ XPAC 平台 SDK 的最新版本安裝包

XPAC (x86) platform SDK:

最新版本的安裝包置於 FTP 網站，如下:

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/

檔名: xpacsdk_ce_n.n.n_vsxxxx.msi

n.n.n :平台 SDK 版本號碼

xxxx: 2005 表示此檔案適用於 VS2005, 2008 表示此檔案適用於 VS2008

提示 & 警告

提供 PACSDK 程式庫的 SDK 安裝包版本號碼必需大於或等於 1.4.0,檔案如 PacSDK_CE_1.4.0_VS2008.msi 或 PacSDK_CE_1.4.0_VS2005.msi

WinPAC (WinCE5) platform SDK:

最新版本的安裝包置於 FTP 網站，如下:

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

檔名: pac270_sdk_yyyymmdd.msi

yyyymmdd : SDK 釋出的日期

提示 & 警告

提供 PACSDK 程式庫的 SDK 安裝包釋出日期晚於 2012/10/15，檔案如 PAC270_SDK_20121015.msi

WinPAC (WinCE7) platform SDK:

最新版本的安裝包置於 FTP 網站，如下:

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

檔名: AM335x_WINCE7_SDK_yyyymmdd.msi

yyyymmdd : SDK 釋出的日期

2. 於 PC 上執行安裝包 (*.msi)，依安裝提示進行安裝直到整個步驟完成為止。

WinPAC(WinCE5.0)平台的 SDK 安裝包安裝到 PC 後，所有 PACSDK 程式庫檔案會被複製到電腦默認的位置

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

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

XPAC 平台的 SDK 安裝包安裝到 PC 後，所有 PACSDK 程式庫檔案會被複製到電腦默認的位置

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

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

WinPAC(WinCE7.0)平台的 SDK 安裝包安裝到 PC 後，所有 PACSDK 程式庫檔案會被複製到電腦默認的位置

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

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

更新 XPACSDK / WinPACSDK 程式庫為 PACSDK 程式庫

請參考文件 w6-10_How_to_update_to_PACSDK_library_from_WinPacSDK_library_en.pdf 及 w6-10_How_to_update_to_PACSDK_library_from_WinPacSDK_library_tc.pdf，
這兩個文件會說明如何將使用者程式使用的 WinPACSDK 程式庫更新為 PACSDK 程式庫。
文件置放位置為

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

請參考文件 X6-10_How_to_update_to_PACSDK_library_from_XPacSDK_library_en.pdf 及 X6-10_How_to_update_to_PACSDK_library_from_XPacSDK_library_tc.pdf，這兩個文件會說明如何將使用者程式使用的 XPACSDK 程式庫更新為 PACSDK 程式庫。
文件置放位置為

http://ftp.icpdas.com/pub/cd/xp-8000-ce6/document/faq/sdk/

或

http://ftp.icpdas.com/pub/cd/xpac-atom-ce6/document/faq/sdk/

2.3. 開發環境設定

XPAC 系列模組的作業系統是 Windows CE 6.0，WinPAC 系列模組的操作系統是 Windows CE5.0。XPAC 和 WinPAC 系列的開發工具有點不同。XPAC 的開發工具為 Visual Studio 2005/2008。而 WinPAC 的開發工具除了支援 Visual Studio 2005/2008 外，還支援嵌入式的 Visual C + +(EVC)。Windows embedded compact 7.0 OS 的 WinPAC 系列，開發工具就只支援 Visual studio 2008。

2.3.1. Visual Studio C/C++/MFC (XPAC 系列)

應用平台

XPAC 系列

必要的頭文件及程式庫文件

以下列出的程式庫檔，頭文件檔或 DLL 檔，需於開發 PAC 應用程式時，包含至開發專案內。

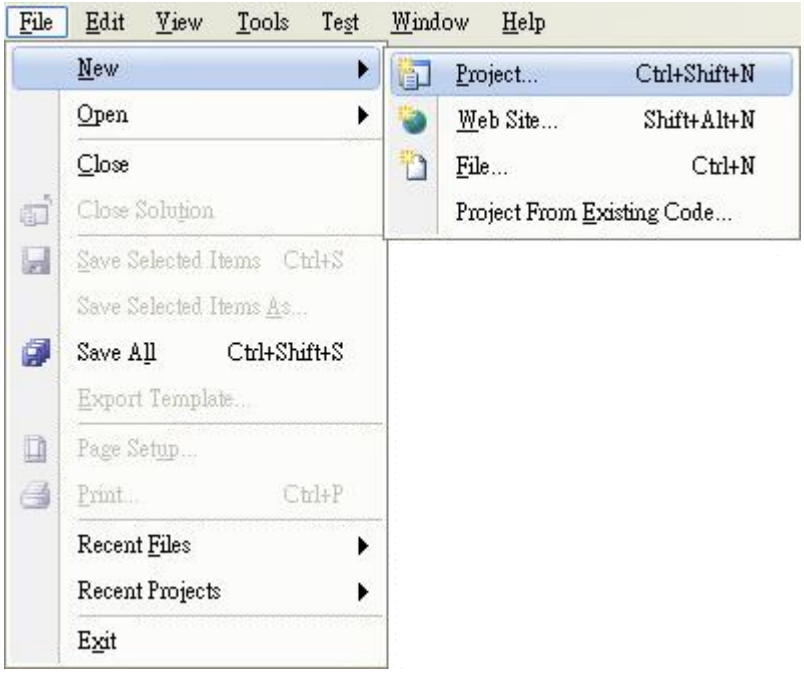
- PACSDK.h
- PACSDK.lib

如果 I-7K/I-87K PWM 模組使用於 PAC 系列設備上，你需要包括下面的文件檔

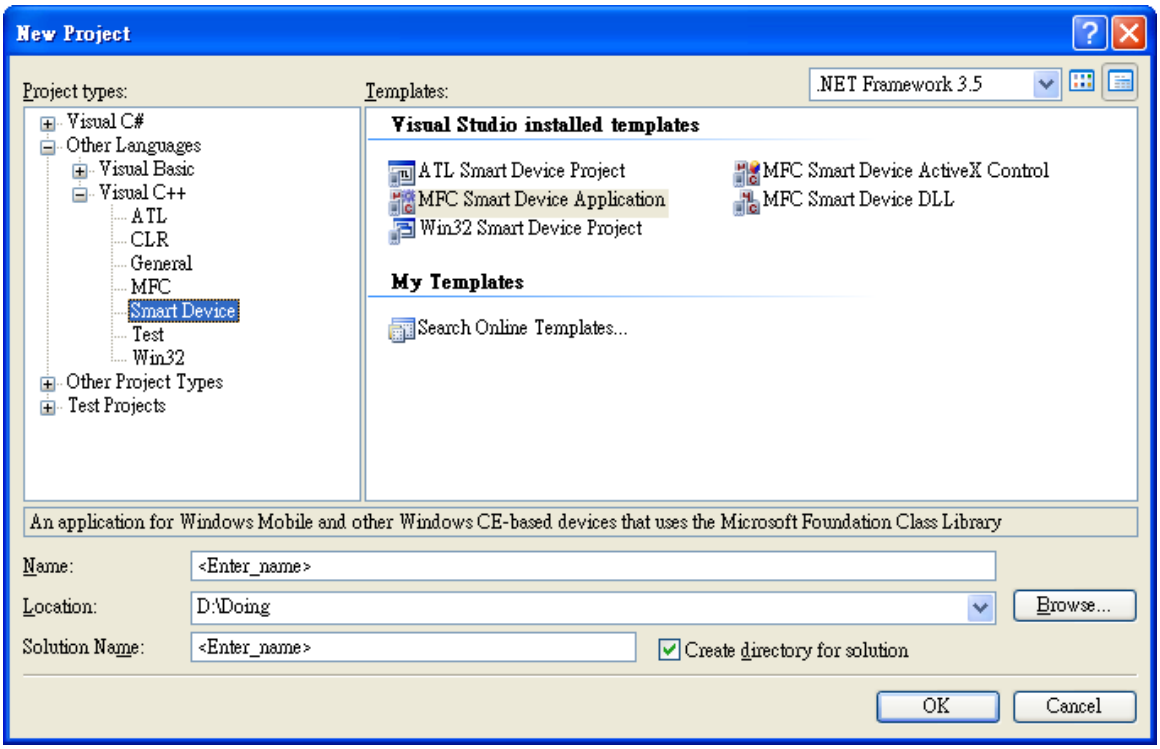
- PACSDK_PWM.h
- PACSDK_PWM.lib

如何於 Visual Studio 2005/2008 使用新的 SDK 開發程式

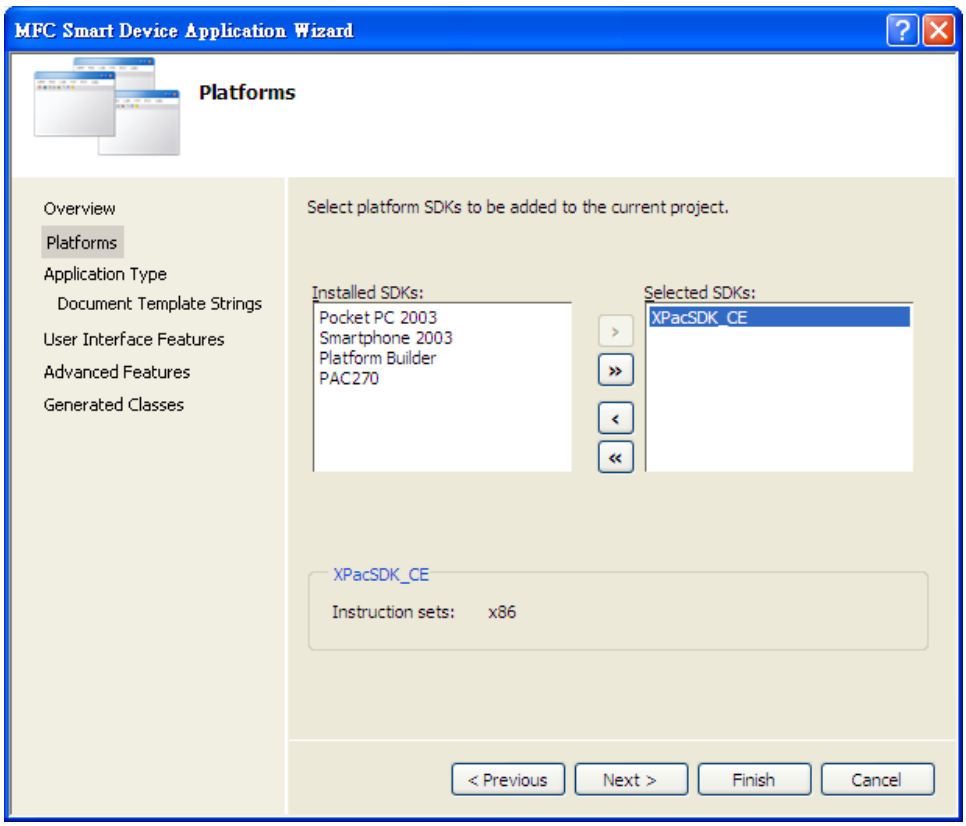
步驟 1：通過使用 Visual Studio 2005/2008 創建一個新的專案



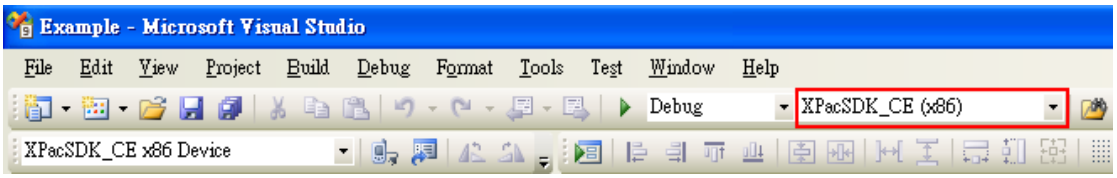
步驟 2：選擇智能設備(Smart Device)



步驟 3：選擇 XPacSDK_CE 平台並添加到當前項目中



步驟 4：在配置工具欄，選擇 XPacSDK_CE(x86)

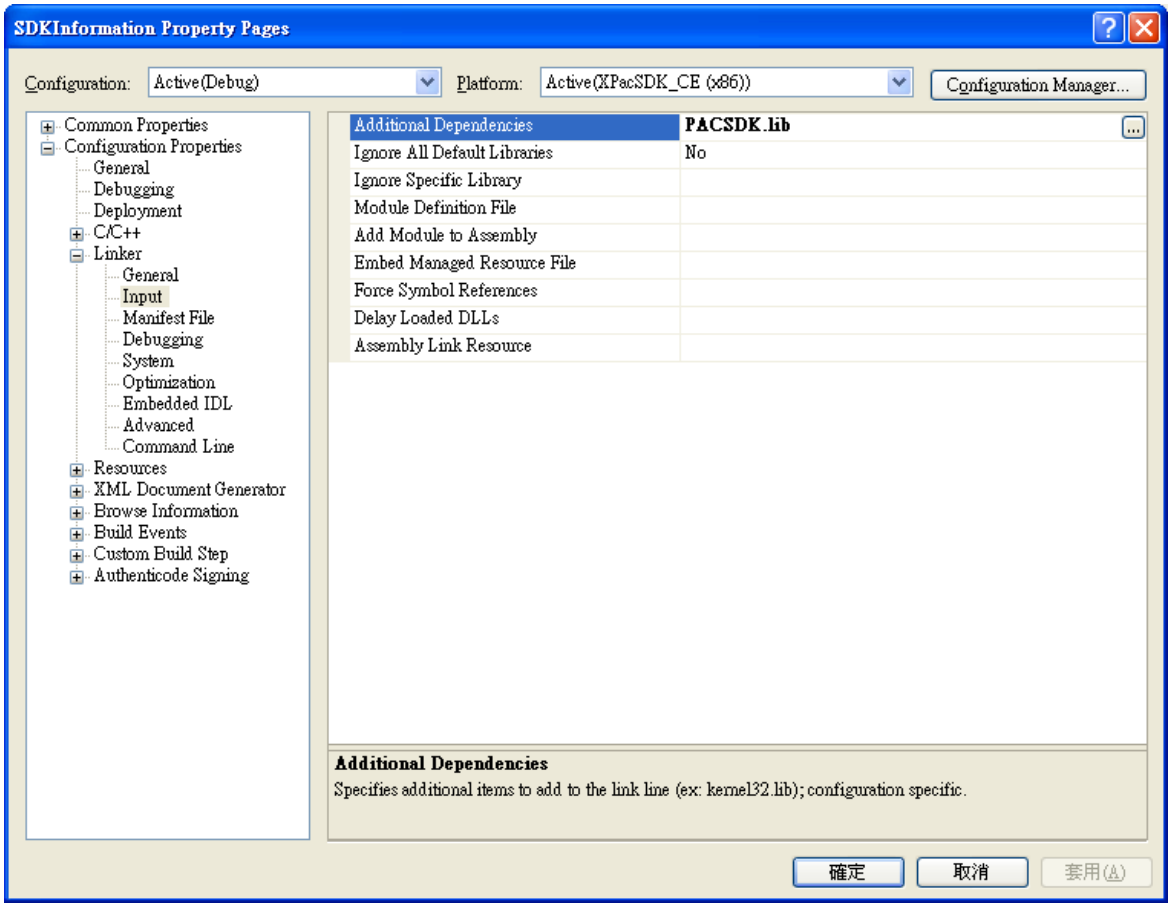


步驟 5：程式碼內包含 PACSDK.h

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

步驟 6：包含 PACSDK 程式庫

在右窗格中，將 PACSDK.lib 加入 Additional Dependencies 項目中。



2.3.2. eVC C/C++/MFC (PXA270 系列)

應用平台

- WP-8x3x/WP-8x4x 系列
- WP-5000 系列(WP-5231 除外)
- VP-23Wx/VP-25Wx/VP-413x 系列

必要的頭文件及程式庫文件

以下列出的程式庫檔，頭文件檔或 DLL 檔，需於開發 PAC 應用程式時，包含至開發專案內。

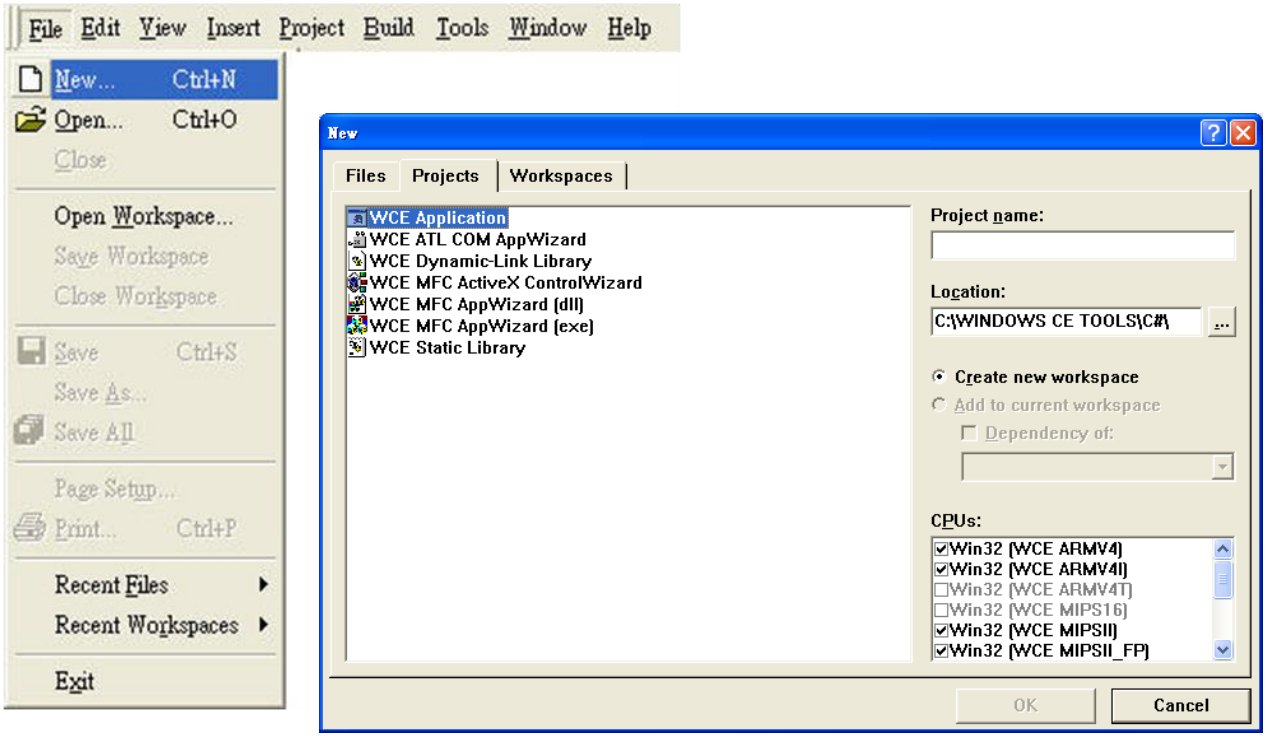
- PACSDK.h
- PACSDK.lib

如果 I-7K/I-87K PWM 模組使用於 PAC 系列設備上，你需要包括下面的文件檔

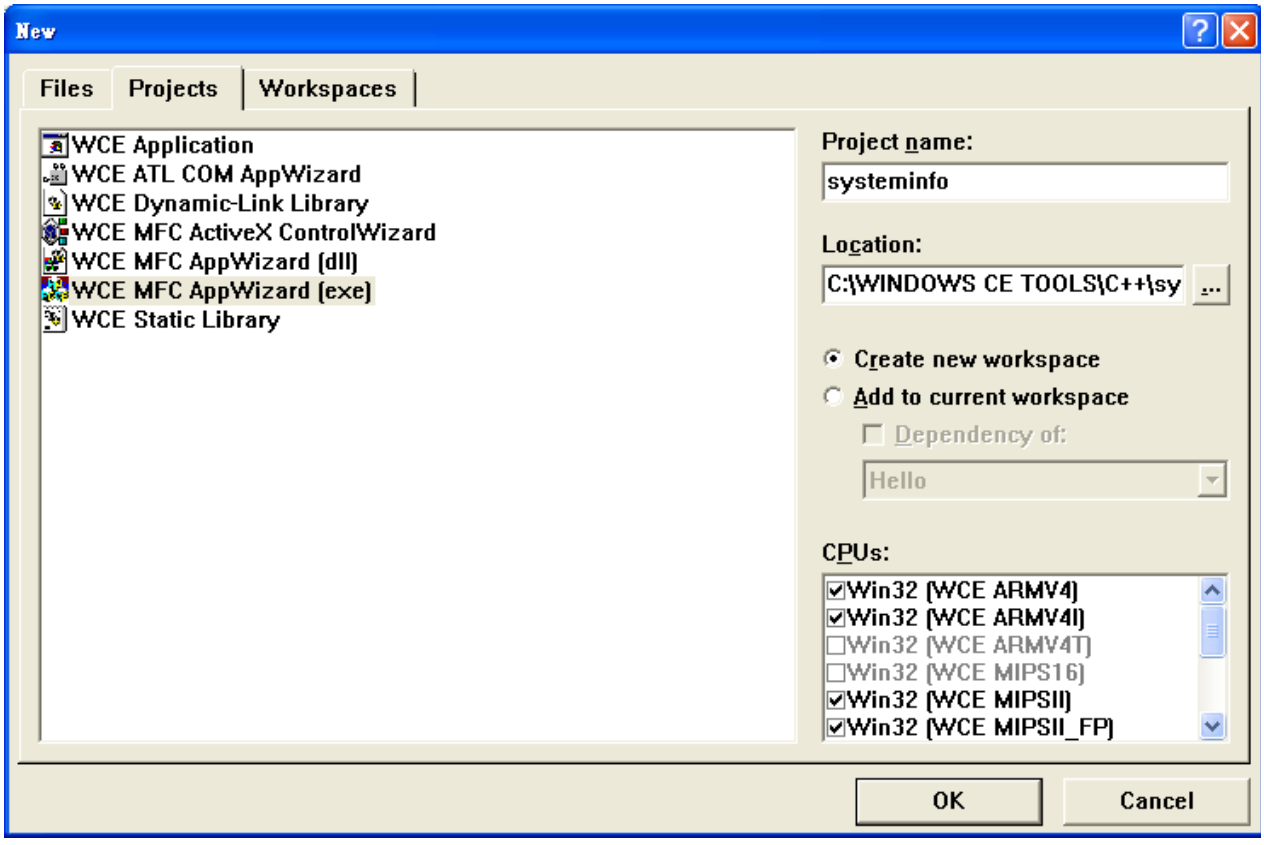
- PACSDK_PWM.h
- PACSDK_PWM.lib

如何於 Microsoft Embedded Visual C++(eVC)使用新的 SDK 開發程式

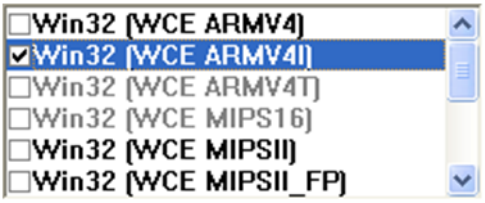
步驟 1：通過用 eVC 創建一個新專案



步驟 2：選擇 WCE MFC AppWizard(exe)



步驟 3：選擇 Win32 [WCE ARMV4I] 平台添加到當前項目中



步驟 4: 在 configuration toolbar, 選擇 “Win32 [WCE ARMV4] Release” 項目。

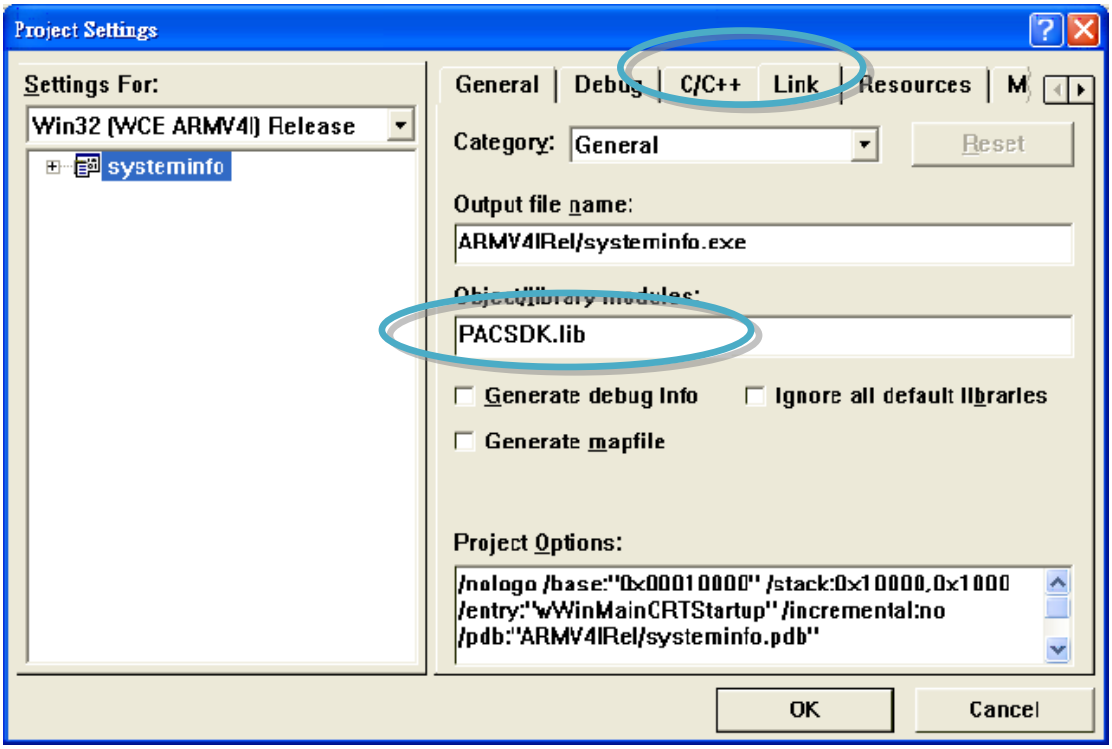
步驟 5: 程式碼內包含 PACSDK.h

#include “PACSDK.h”

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

步驟 6: 包含 PACSDK 程式庫

在右窗格中，Link Tab，將 PACSDK.lib 加入 Object/library modules:項目中



2.3.3. Visual Studio C/C++/MFC (PXA270 系列)

應用平台

- **WinPAC 系列**
- ViewPAC 系列
- XPAX 系列

必要的頭文件及程式庫文件

以下列出的程式庫檔，頭文件檔或 DLL 檔，需於開發 PAC 應用程式時，包含至開發專案內。

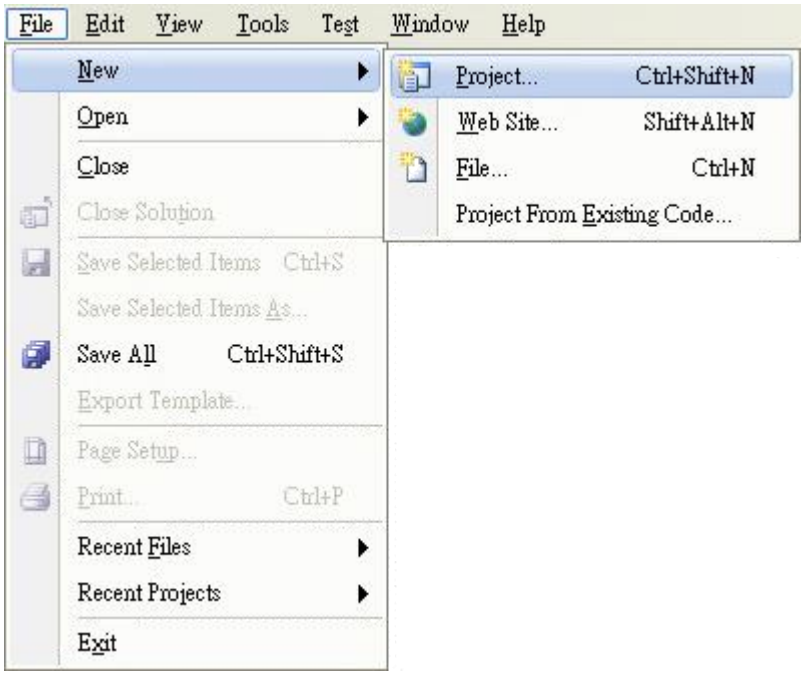
- PACSDK.h
- PACSDK.lib

如果 I-7K/I-87K PWM 模組使用於 PAC 系列設備上，你需要包括下面的文件檔

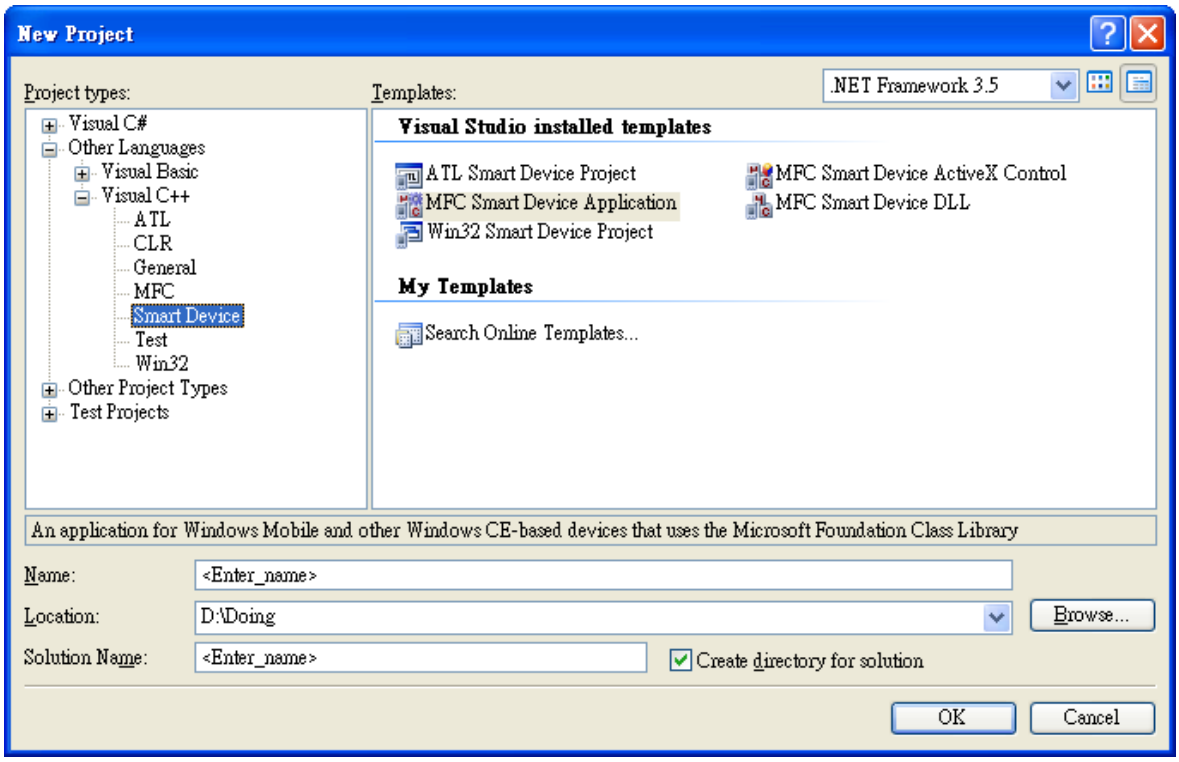
- PACSDK_PWM.h
- PACSDK_PWM.lib

如何於 Visual Studio 2005/2008 使用新的 SDK 開發程式

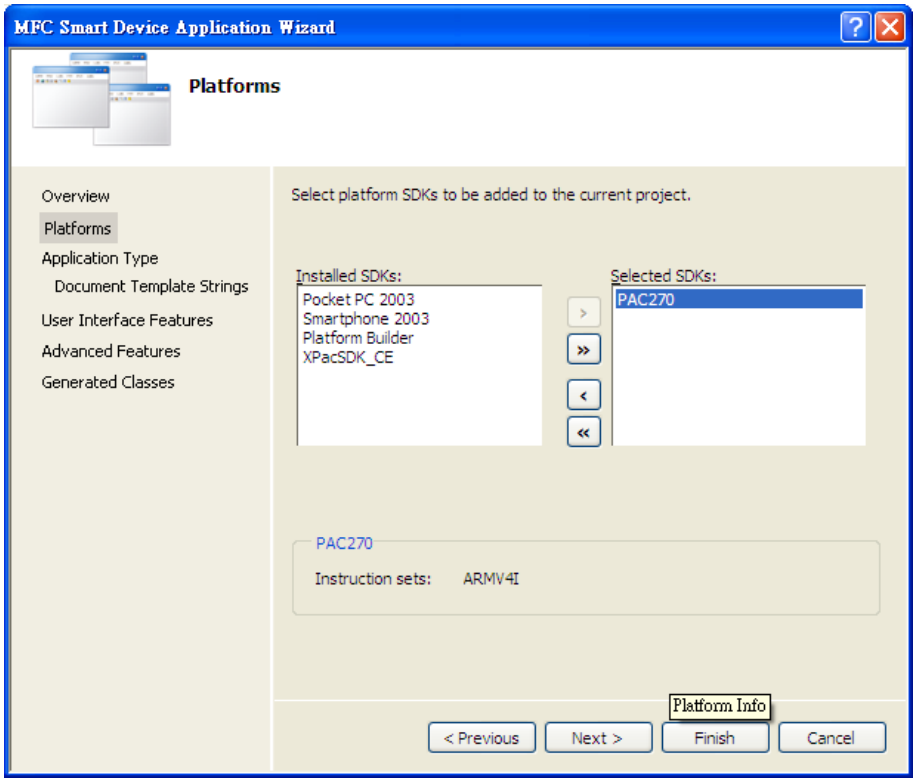
步驟 1：通過使用 Visual Studio 2005/2008 創建一個新的專案



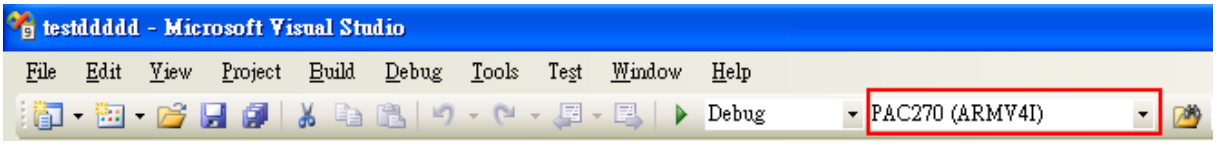
步驟 2：選擇智能設備(Smart Device)



步驟 3：選擇 PAC270 平台並添加到當前項目中



步驟 4：在配置工具欄，選擇 PAC270(ARMV4I)



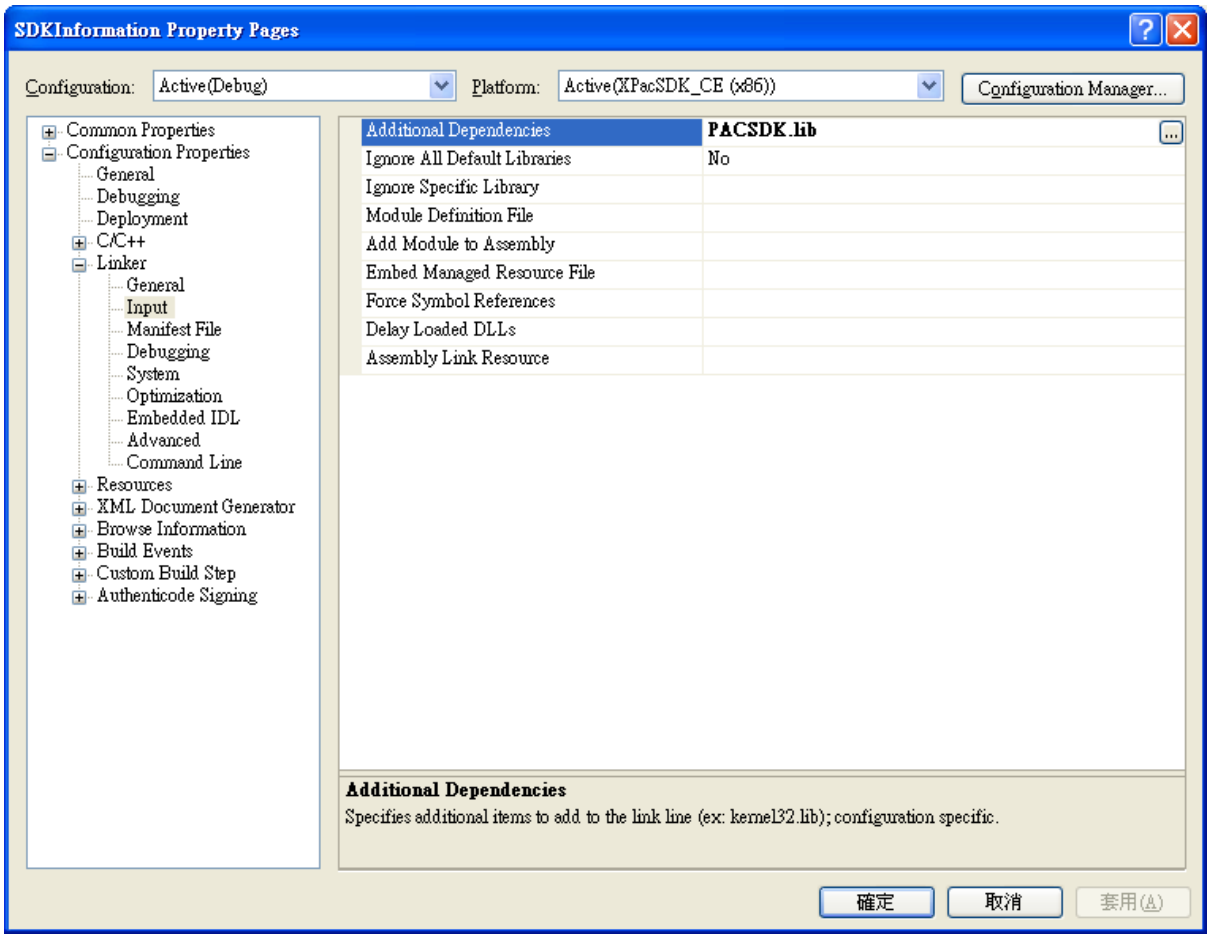
步驟 5：程式碼內包含 PACSDK.h

#include “PACSDK.h”

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

步驟 6：包含 PACSDK 程式庫

在右窗格中，將 PACSDK.lib 加入 Additional Dependencies 項目中。



2.3.4. Visual Studio C/C++/MFC (AM335x 系列)

應用平台

- **WinPAC 系列**
- ViewPAC 系列
- XPAX 系列

必要的頭文件及程式庫文件

以下列出的程式庫檔，頭文件檔或 DLL 檔，需於開發 PAC 應用程式時，包含至開發專案內。

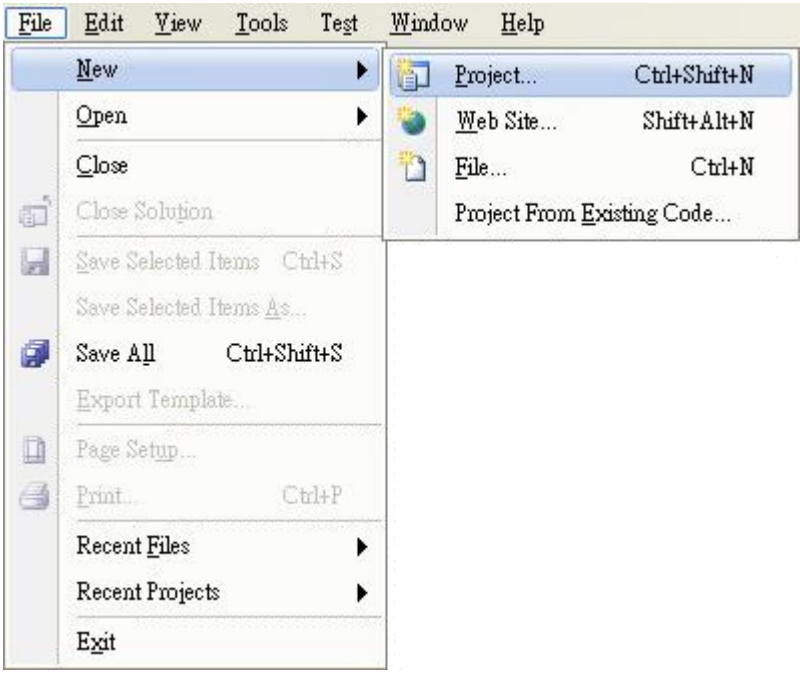
- PACSDK.h
- PACSDK.lib

如果 I-7K/I-87K PWM 模組使用於 PAC 系列設備上，你需要包括下面的文件檔

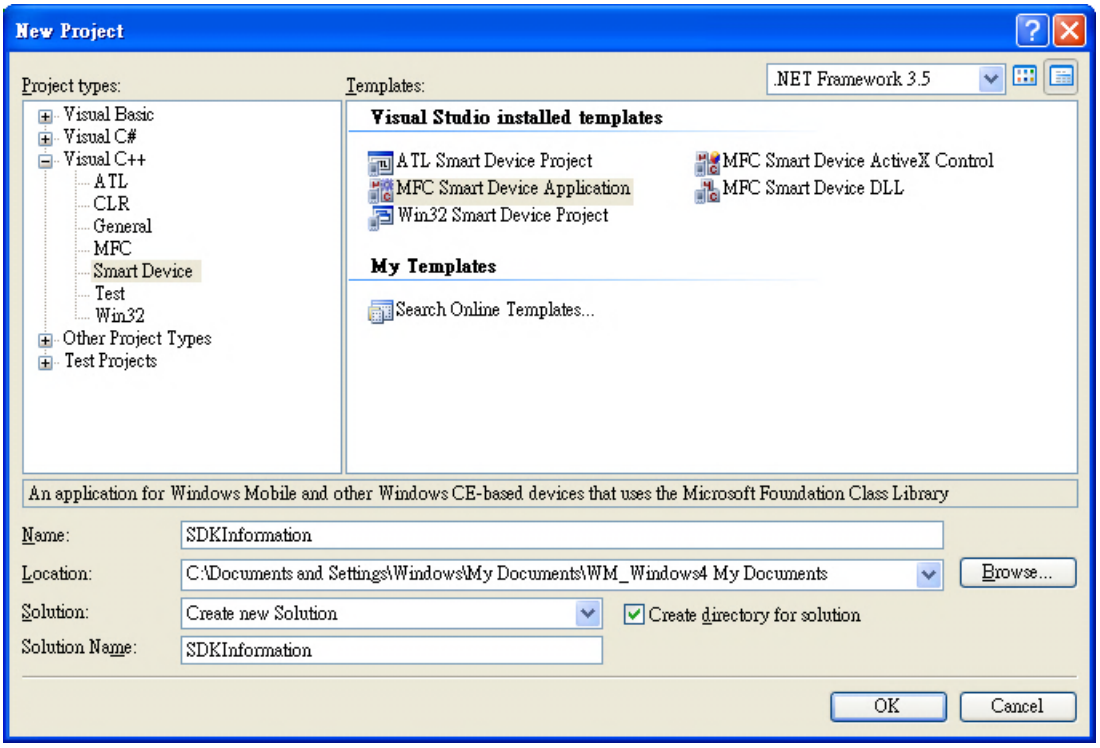
- PACSDK_PWM.h
- PACSDK_PWM.lib

如何於 Visual Studio 2008 使用新的 SDK 開發程式

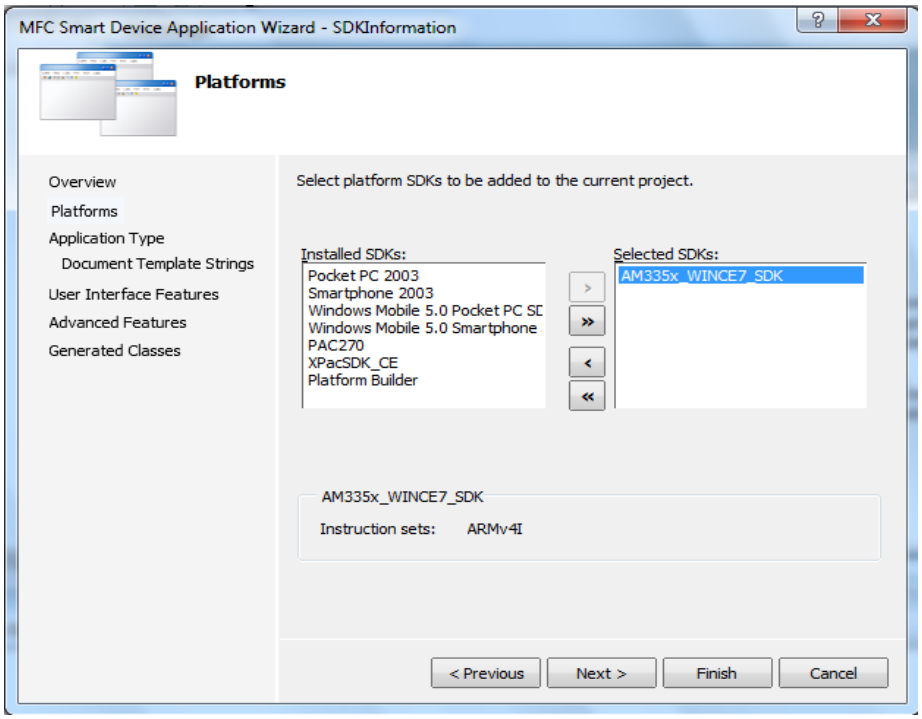
步驟 1：通過使用 Visual Studio 2008 創建一個新的專案



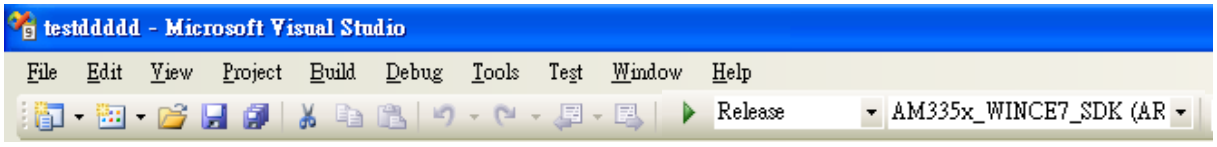
步驟 2：選擇智能設備(Smart Device)



步驟 3：選擇 AM335x_WINCE7_SDK 平台並添加到當前項目中



步驟 4：在配置工具欄，選擇 AM335x_WinCE7_SDK



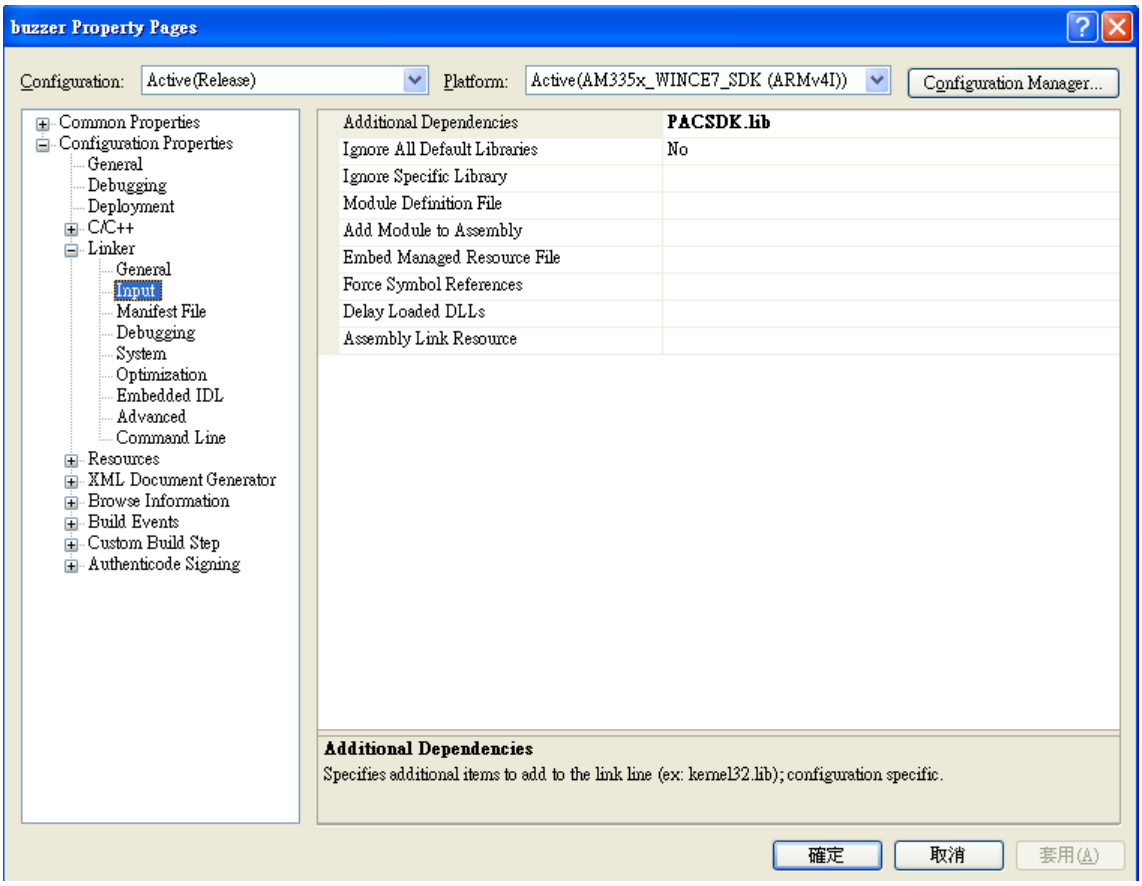
步驟 5：程式碼內包含 PACSDK.h

#include “PACSDK.h”

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

步驟 6：包含 PACSDK 程式庫

在右窗格中，將 PACSDK.lib 加入 Additional Dependencies 項目中。



2.3.5. C# (XPAC/WinPAC 系列)

應用平台

- **WinPAC 系列**
- ViewPAC 系列
- XPAX 系列

必要的程式庫文件

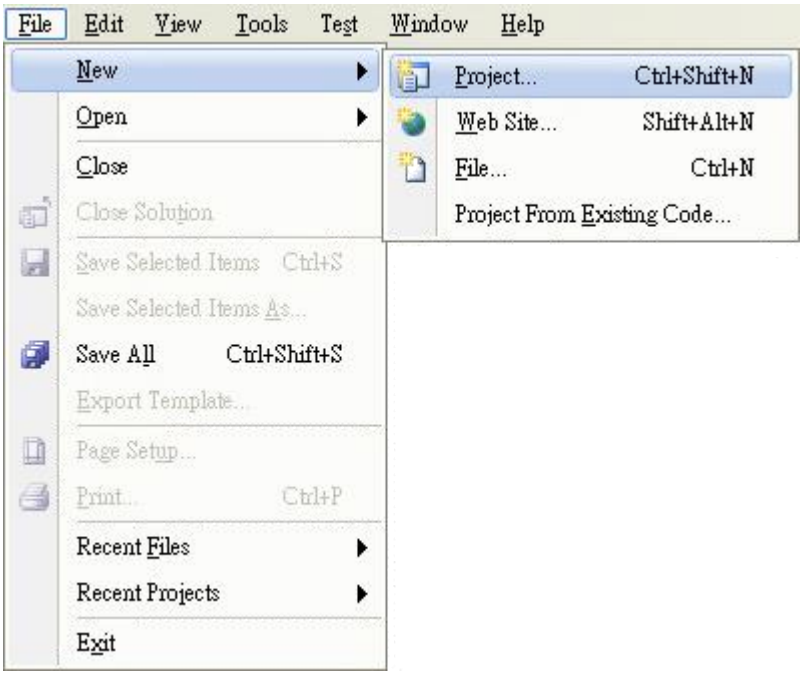
以下列出的 DLL 檔，需於開發 XPAC 應用程式時，包含至開發專案內或當作插件(plug-in)

- PACSDK.dll
- PACSDK_PWM.dll (如果於裝置上使用 I-7K/I-87K PWM 模組)

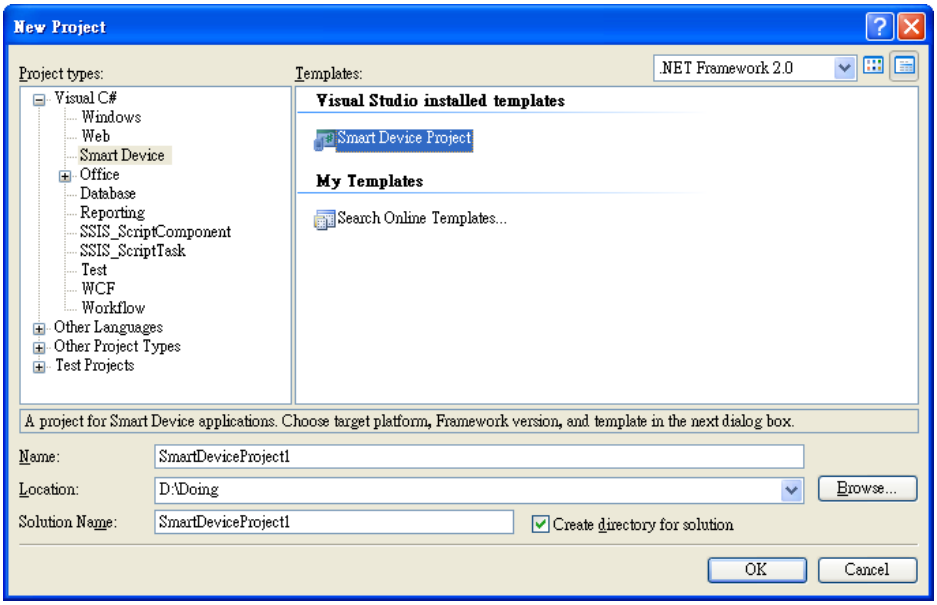
如何於 Visual Studio 2005/2008 使用新的 SDK 開發程式

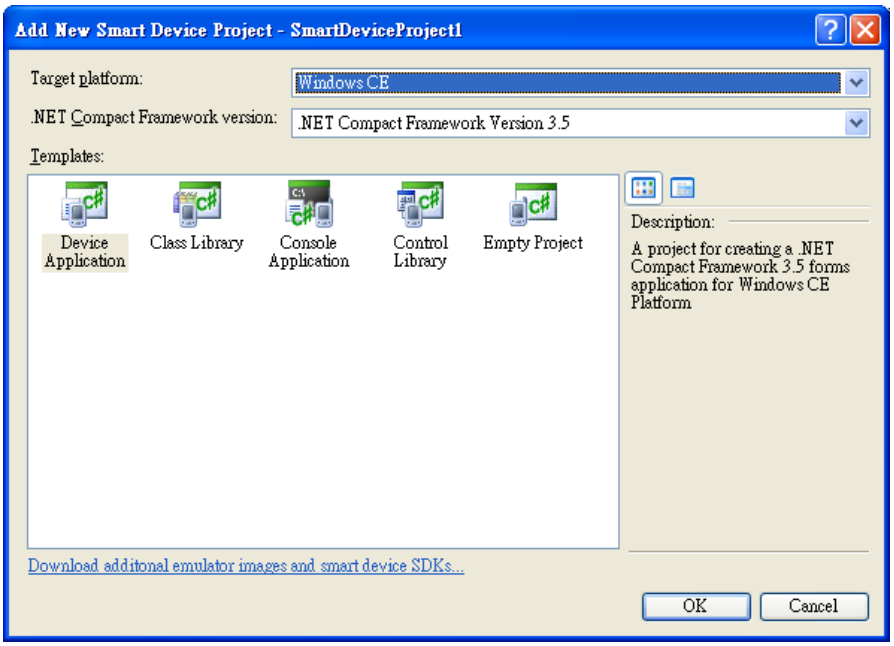
1.使用 Dll Import:

步驟 1：通過使用 Visual Studio 2005/2008 創建一個新的 C#專案



步驟 2: 選擇智能設備(Smart Device)





步驟 3: 因為使用 “DllImport”，需於程式先加入 using System.Runtime.InteropServices，然後才能使用 “DllImport”

以使用 PACSDK API，pac_WriteDO 函數為例

[此函數定義在 PACSDK.h 頭文件內]

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

[如何於你的.NET 專案使用]

- 將以下這一行加入你的程式開頭:

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

- 加入以下函數宣告程式碼，如下:

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

- 於你的.NET 程式內使用 pac_WriteDO 函數

[程式碼片斷]

```
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);
      }
   }
}
```

2. 使用 PACNET.dll

PACNET.dll 是一個.NET Compact Framework 的程式庫，PACNET.dll 可用於 C＃的程式，也可用於 VB.net 程試。

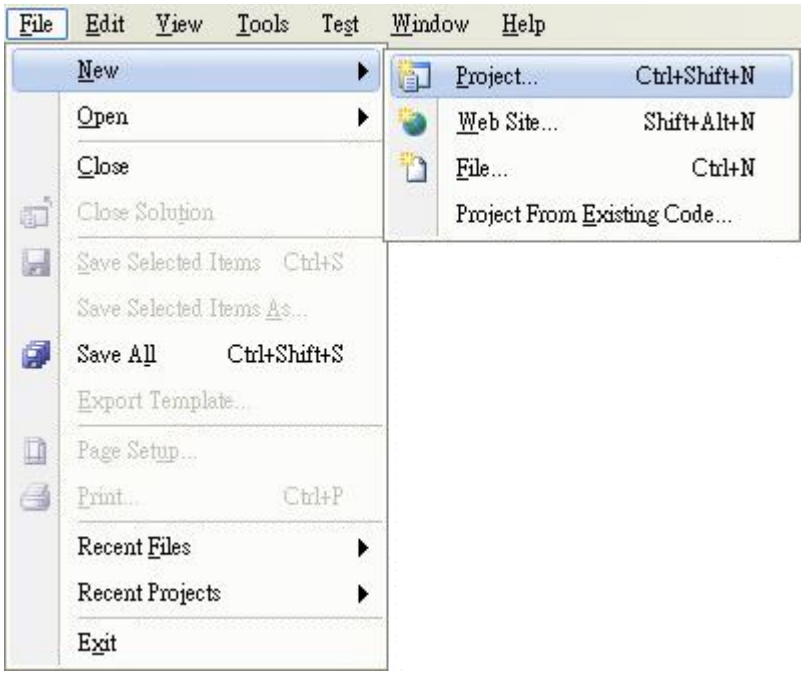
。PACNET.dll(PACNET.dll 需於執行檔置於同一個目錄下)

這個庫的最新版本位於：

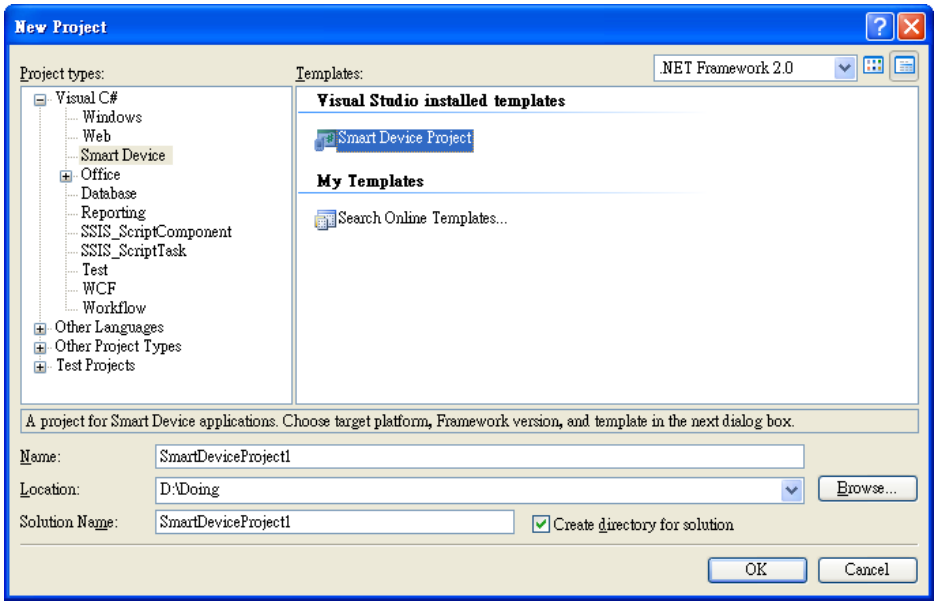
CD root\XP-8000-CE6\SDK\XPacNET (in the companion CD)

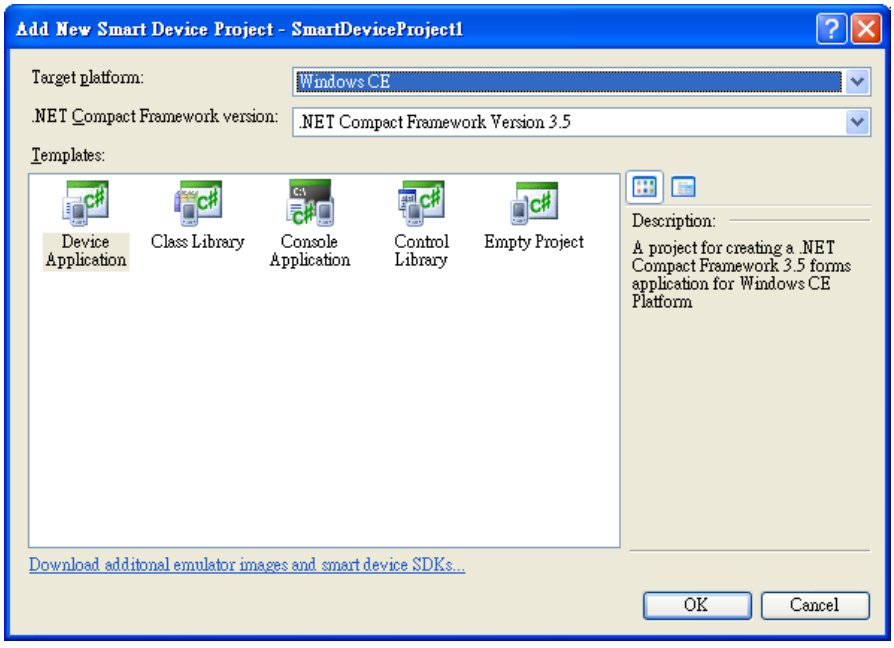
ftp://ftp.icpdas.com/pub/cd/xp-8000-ce6/sdk/xpacnet

步驟 1：通過使用 Visual Studio 2005/2008 創建一個新的 C#專案

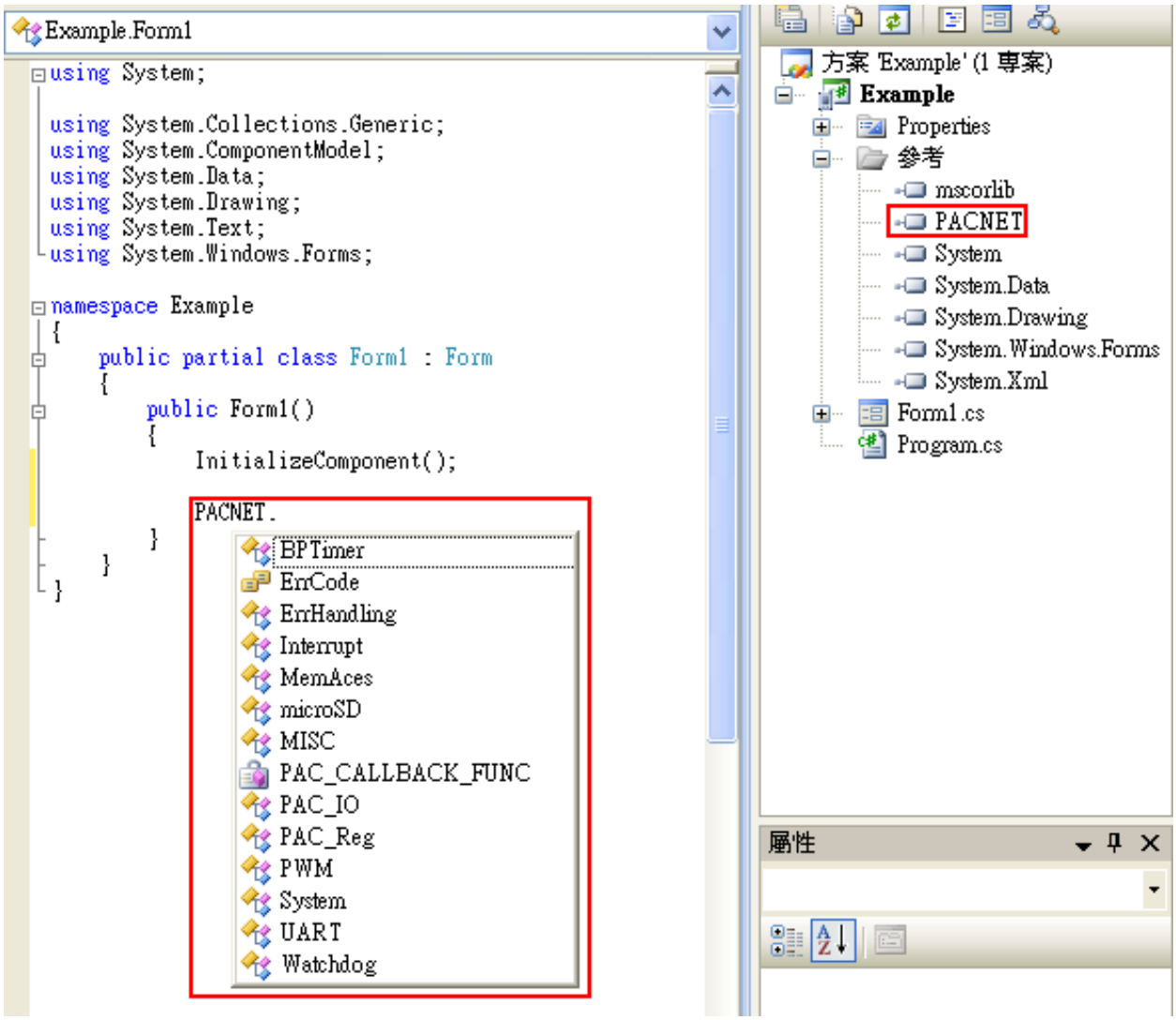


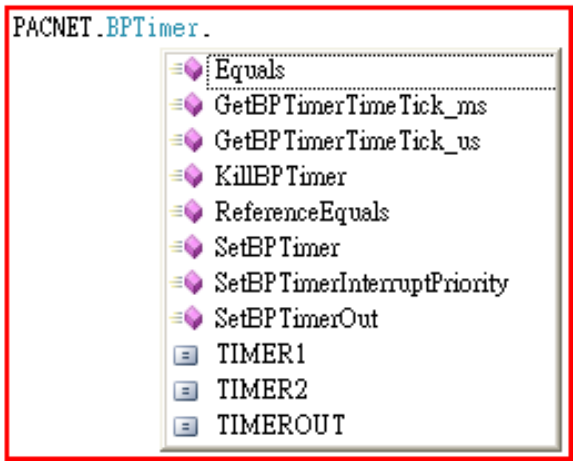
步驟 2：選擇智能設備(Smart Device)





步驟 3: 將 PACNET.dll 加入專案的參考內，即可 PACNET.dll





2.3.6. VB.net(XPAC/WinPAC 系列)

應用平台

- **WinPAC 系列**
- ViewPAC 系列
- XPAX 系列

必要的程式庫文件

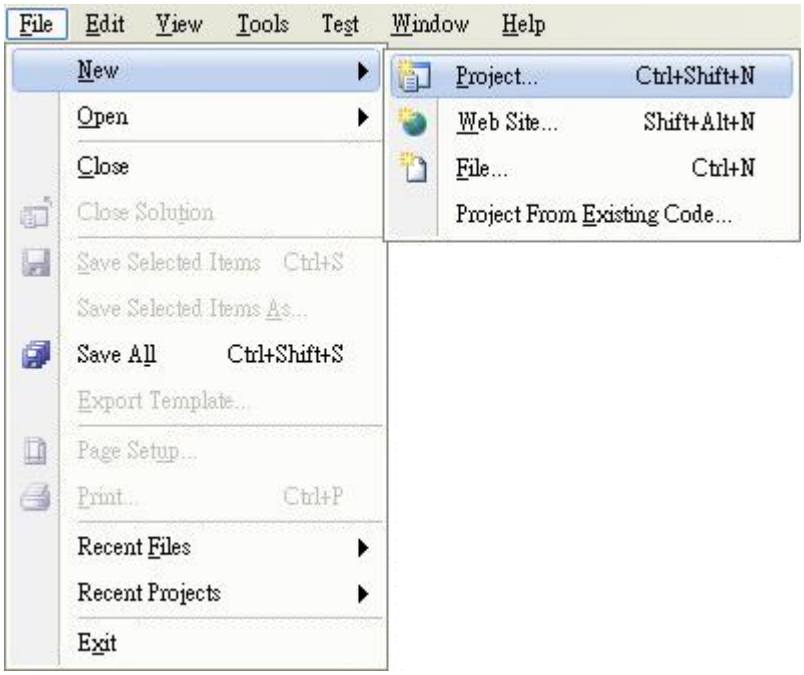
以下列出的 DLL 檔，需於開發 XPAC 應用程式時，包含至開發專案內或當作插件(plug-in)

- PACSDK.dll
- PACSDK_PWM.dll (如果於裝置上使用 I-7K/I-87K PWM 模組)

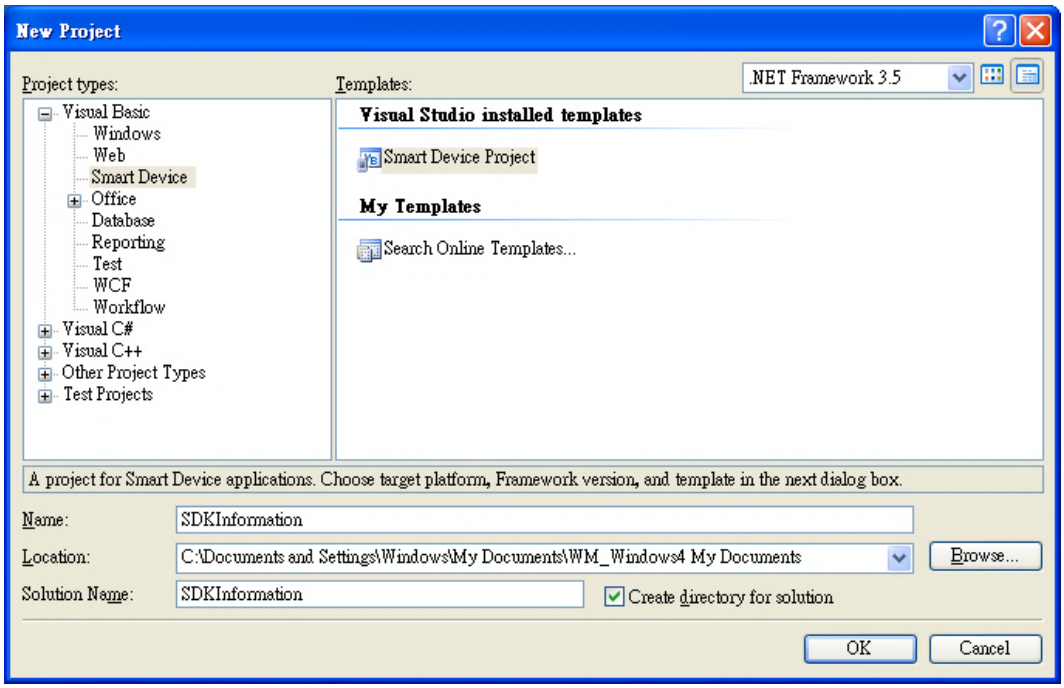
如何於 Visual Studio 2005/2008 使用新的 SDK 開發程式

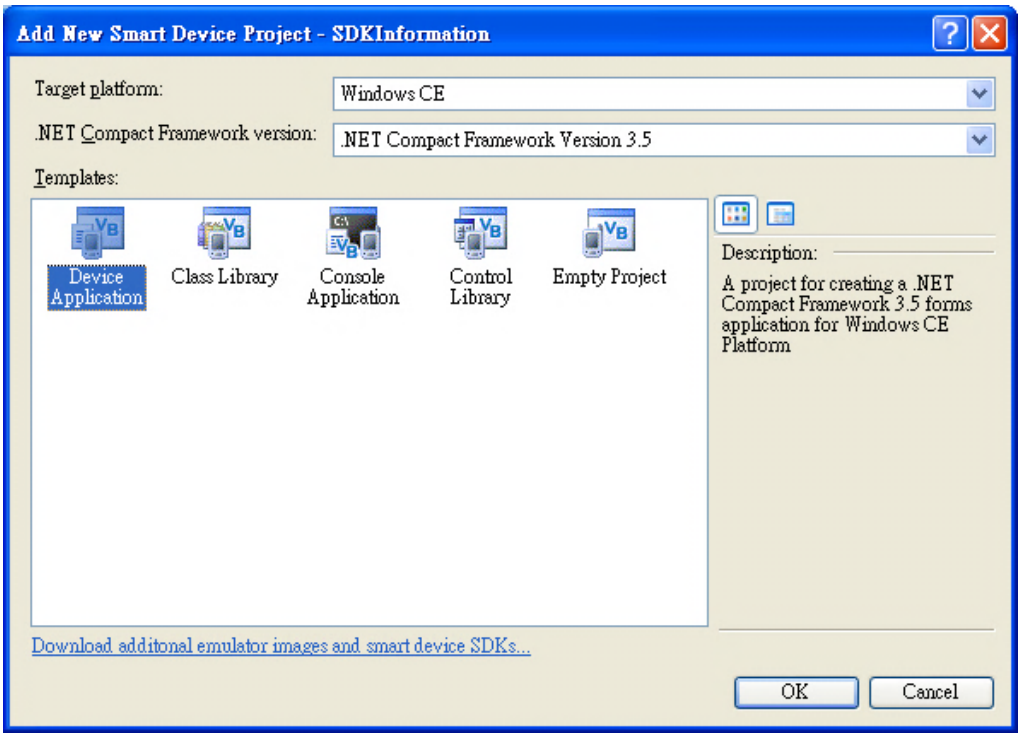
1. 使用 Dll Import:

步驟 1：通過使用 Visual Studio 2005/2008 創建一個新的 VB.net 專案



步驟 2: 選擇智能設備(Smart Device)





步驟 3：使用 “DllImport”，,需於程式先加入 using System.Runtime.InteropServices，然後才能使用 “DllImport”

以使用 PACSDK API，pac_WriteDO 函數為例

[此函數定義在 PACSDK.h 頭文件內]

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

[如何於你的.NET 專案使用]

- 將以下這一行加入你的程式開頭:

  Imports System.Runtime.InteropServices

- 加入以下函數宣告程式碼，如下:

  <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

- .NET 程式內使用 pac_WriteDO 函數

[程式碼片斷]

```
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
```

2. 使用 PACNET.dll

PACNET.dll 是一個.NET Compact Framework 的程式庫，PACNET.dll 可用於 C＃的程式，也可用於 VB.net 程式。

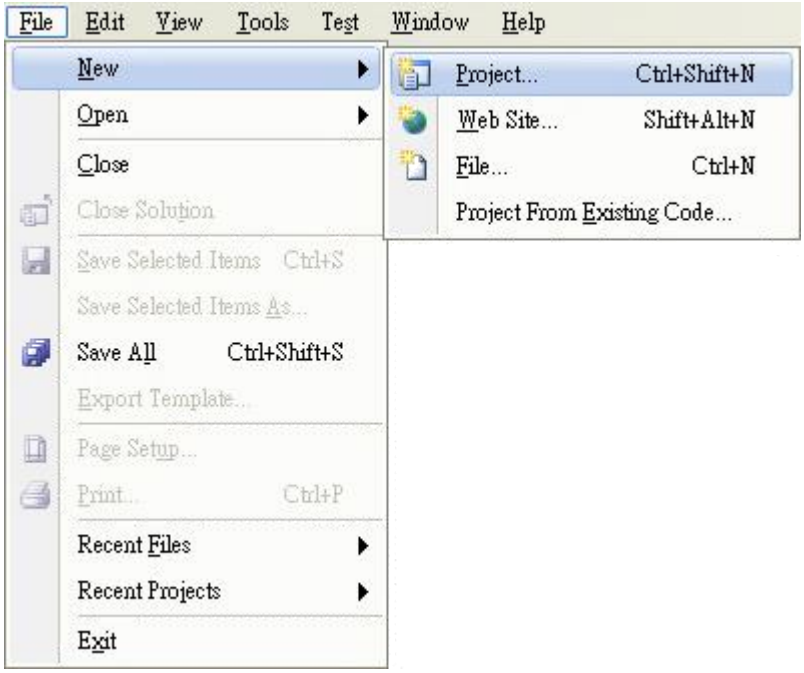
。PACNET.dll(PACNET.dll 需於執行檔置於同一個目錄下)

這個庫的最新版本位於：

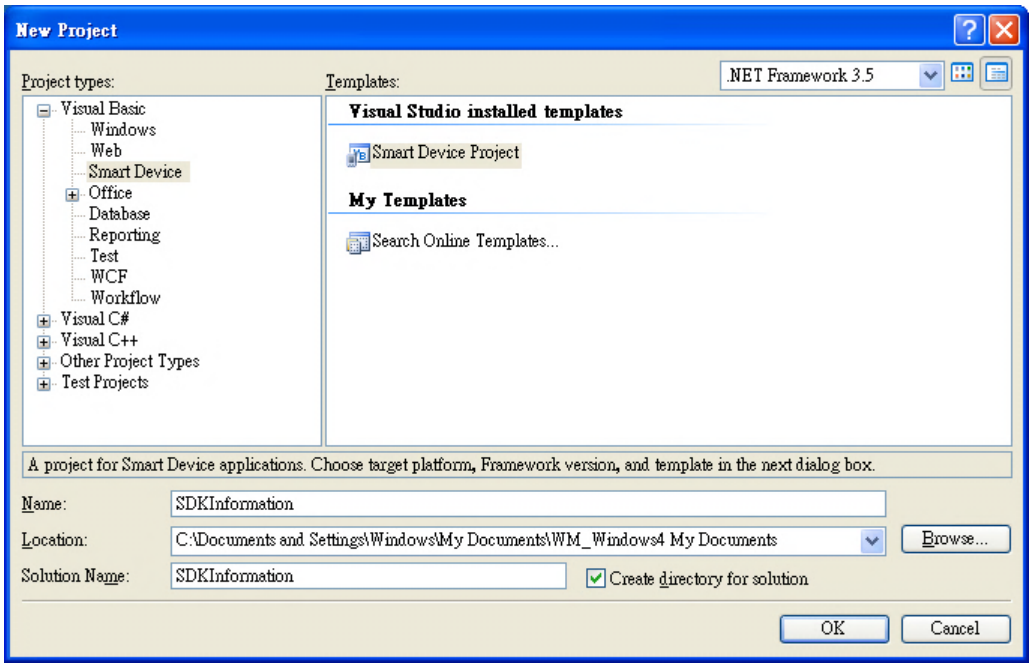
CD root\XP-8000-CE6\SDK\XPacNET (在隨附的 CD 中)

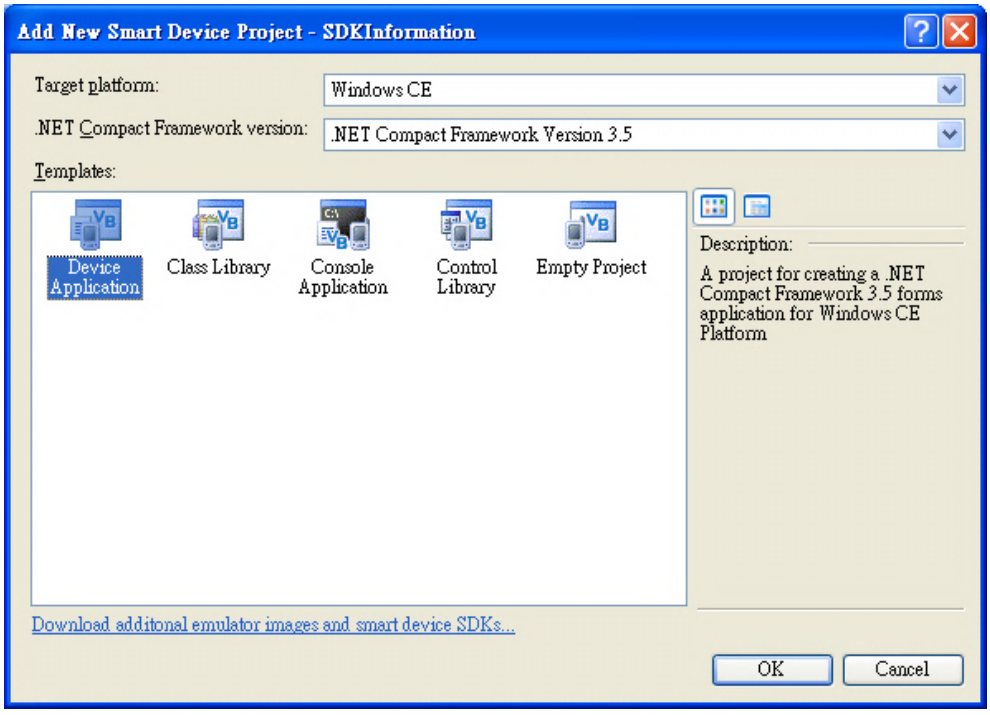
ftp://ftp.icpdas.com/pub/cd/xp-8000-ce6/sdk/xpacnet

步驟 1: 通過使用 Visual Studio 2005/2008 創建一個新的 VB.net 專案

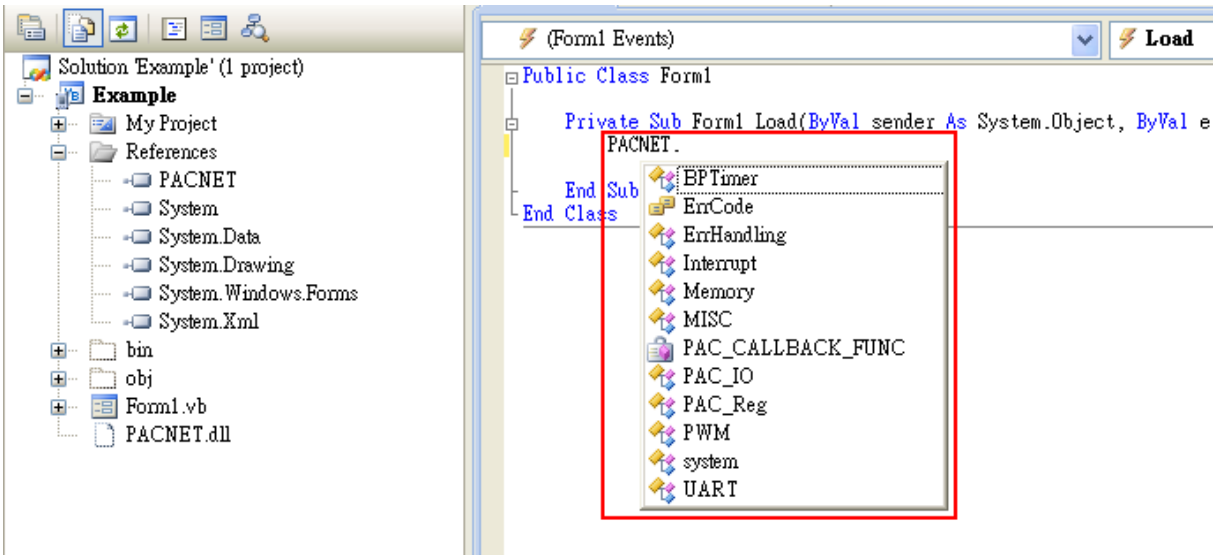


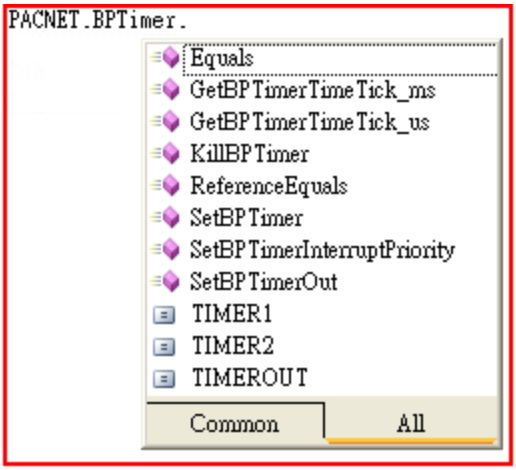
步驟 2: 選擇智能設備(Smart Device)





步驟 3: 將 PACNET.dll 加入專案的參考內，即可 PACNET.dll





以下列出每個平台的參考文件位址，每個文件有詳細說明，教導使用者如何開發程式。

XP-8x4x-CE6

詳細說明，請參考 xp-8000-ce6 user manual 第四章

ftp://ftp.icpdas.com/pub/cd/xp-8000-ce6/document/user_manual/

XP-8x4x-Atom-CE6

詳細說明，請參考 xp-8000-Atom-ce6 user manual 第四章

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

XP-8x3x-CE6

詳細說明，請參考 xp-8000-Atom-ce6 user manual 第四章

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

WP-8000 (CE5.0)

詳細說明，請參考 wp-8000 user manual 第四章

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

WP-5000(CE5.0)

詳細說明，請參考 wp-5000 user manual 第四章

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

VP-2xWx(CE5.0)

詳細說明，請參考 ViewPAC user manual 第四章

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

WP-9000 (CE7.0)

詳細說明，請參考 wp-9000 user manual 第四章

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

WP-8000 (CE7.0)

詳細說明，請參考 wp-8000 user manual 第四章

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

WP-5000(CE7.0)

詳細說明，請參考 wp-5000 user manual 第四章

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

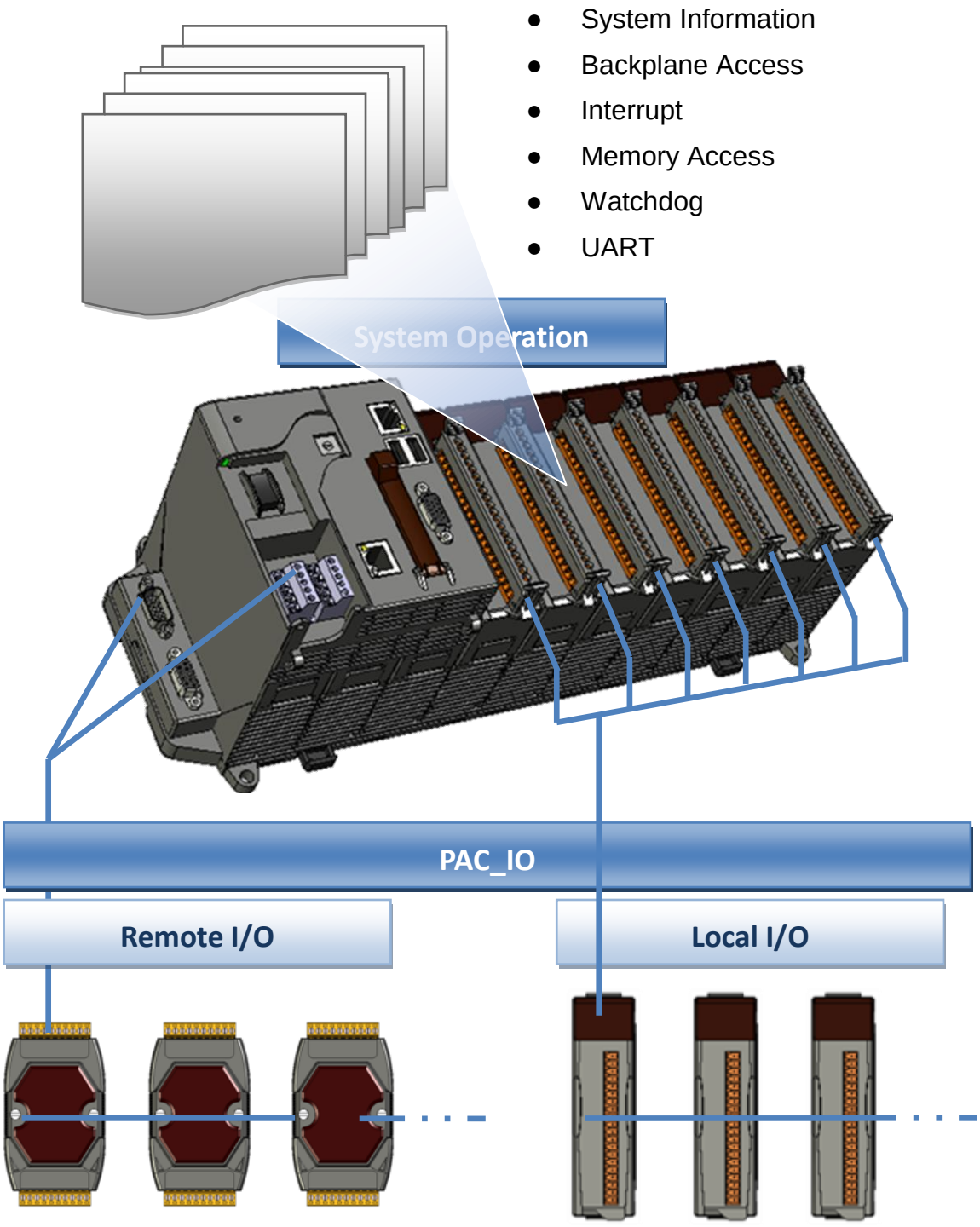
ViewPAC(CE7.0)

詳細說明，請參考 ViewPAC user manual 第四章

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

3. PACSDK API 函數說明

PACSDK 程式庫的 API 函數及其分類如下



3.1. System Information API (系統相關 API)

系統信息功能和消息描述或更改系統配置，設置和屬性。

支援的 PAC

以下列出 System information API 函數所支援的 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

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_GetBatteryLevel	Y	-	Y	-	Y	Y	Y	-	Y
pac_RegistryHotPlug (Beta 測試)	-	-	-	-	-	-	-	-	-
pac_UnregistryHotPlug (Beta 測試)	-	-	-	-	-	-	-	-	-
pac_SetBackLight	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetBackLight	Y	Y	Y	Y	Y	Y	Y	Y	Y

支援的 API 函數

PACSDK 函數	PACNET 函數	說明
pac_GetModuleName	Sys.GetModuleName	檢索插入 XPAC/WinPAC 系列設備，指定 I/O 模組的名稱。
pac_GetRotaryID	Sys.GetRotaryID	取得旋轉開關的位置數值
pac_GetSerialNumber	Sys.GetSerialNumber	取得 XPAC / WinPAC 上硬件的序列號碼
pac_GetSDKVersion	Sys.GetSDKVersion	取得當前 PACSDK.dll 的版本號碼
pac_ChangeSlot	Sys.ChangeSlot	切換 Slot 位置至，要操作的 87K 模組
pac_CheckSDKVersion	Sys.CheckSDKVersion	用於當前使用的 PACSDK.dll 的版本號碼與指定版本號碼比較
pac_ModuleExists	Sys.ModuleExists	確定 I/O 模組存在與否，可適用於本地或遠端模組。
pac_GetOSVersion	Sys.GetOSVersion	取得當前作業系統(OS)的版本號碼
pac_GetMacAddress	Sys.GetMacAddress	取得裝置上的 MAC address 值
pac_ReBoot	Sys.ReBoot	重新啟動系統
pac_GetCPUVersion	Sys.GetCPUVersion	取得 CPU 版本
pac_EnableLED	Sys.EnableLED	設置 LED 的狀態
pac_EnableLEDs	Sys.EnableLEDs	設置指定 LED 的狀態
pac_BackwardCompatible()	Sys.BackwardCompatible	決定了操作系統是否在向後兼容模式或無法正常工作。 確定作業系統可否向下相容的運作
pac_GetEbootVersion	Sys.GetEbootVersion	取得 Eboot 的版本 (WinPAC 適用)
pac_GetComMapping	Sys.GetComMapping	取得 COM port 的索引映射
pac_GetModuleType	Sys.GetModuleType	透過此函數檢索 WinPAC/XPAC 槽位上的 I/O 模組類型

PACSDK 函數	PACNET 函數	說明
pac_GetPacNetVersion	Sys.GetPacNetVersion	獲取當前平台所使用的 PACNET.dll 版本號。
pac_BuzzerBeep	Sys.BuzzerBeep	蜂鳴器聲音輸出。
pac_GetBuzzerFreqDuty	Sys.GetBuzzerFreqDuty	取得蜂鳴器頻率值，與聲音的持續時間。
pac_SetBuzzerFreqDuty	Sys.SetBuzzerFreqDuty	設定頻率值和蜂鳴聲音的持續時間。
pac_StopBuzzer	Sys.StopBuzzer	將停止蜂鳴器。
pac_GetDIPSwitch	Sys.GetDIPSwitch	取得 DIP 開關的數值
pac_GetSlotCount	Sys.GetSlotCount	取得所有 I/O slot 的總數量
pac_EnableRetrigger	Sys.EnableRetrigger	設定 slot 中斷再觸發器狀態。
pac_GetBackplaneID	Sys.GetBackplaneID	取得底板 ID
pac_GetBatteryLevel	Sys.GetBatteryLevel	取得底板電池電量狀態
pac_RegistryHotPlug(Beta 測試)	Sys.RegistryHotPlug (Beta 測試)	開啟熱插拔後註冊註冊表。
pac_UnregistryHotPlug(Beta 測試)	Sys.UnregistryHotPlug (Beta 測試)	關閉熱插拔後，刪除註冊表鍵值
pac_SetBackLight	Sys.SetBackLight	設定 ViewPAC 系列產品的螢幕背光值
pac_GetBackLight	Sys.GetBackLight	取得 ViewPAC 系列產品的螢幕背光值

3.1.1. pac_GetModuleName

此函數檢索插入 XPAC/WinPAC 系列設備，指定 slot 上 I/O 模組的名稱。

語法

C++

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

參數

Slot

[in] 輸入在 XPAC/WinPAC 系列上，被插入模組的 slot 號碼。

strName

[out] 接收指定 I/O 模組名稱的暫存區指標。

回傳值

假如 8K I/O 模組沒有定義，回傳值會是一些其他值。

範例

[C]

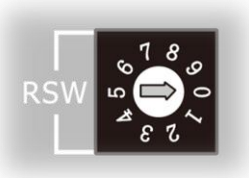
```
byte slot = 1;
char strName[10] ;
pac_GetModuleName(slot, strName);
```

[C#]

```
byte slot = 1;
string strName;
int ModuleType = 0;
ModuleType = PACNET.Sys.GetModuleName(slot, ref strName);
//因為是參考指標,你必須使用關鍵字, ref.
```

```
//此函數有一個重載函數如下
//回傳值為模組字串名稱.
byte slot = 1;
string strName;
strName = PACNET.Sys.GetModuleName(slot);
```

3.1.2. pac_GetRotaryID



透過此函數獲取當前 RSW 的數值。

語法

C++

```
int pac_GetRotaryID();
```

參數

此函數沒有參數。

回傳值

讀取成功，此函數將會回傳當前 RSW 的數值。

若讀取失敗，函數將會回傳一的無效值，欲知更多錯誤細節內容請呼叫 pac_GetLastError 函數。

範例

[C]

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

[C#]

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

3.1.3. pac_GetSerialNumber

透過此函獲取當前 XPAC/WinPAC 平台上的 SerialNumber。

語法

C++

```
voidpac_GetSerialNumber(
    LPSTR SerialNumber
);
```

參數

SerialNumber

[out] XPAC/WinPAC 平台上的序列號。

回傳值

此函數沒有回傳值。

範例

[C]

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

[C#]

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

備註

如果檢索值是 null，表示函數執行失敗或設備是不是有效的產品。

3.1.4. pac_GetSDKVersion

此函數可檢索當前平台所使用的 PACSDK.dll 版本號。

語法

C++

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

參數

sdk_version

[out] 當前平台所使用的 PACSDK.dll 版本號。

回傳值

此函數沒有回傳值。

範例

[C]

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

[C#]

```
string PacSDK;
string PacNET;
PacSDK = PACNET.Sys.GetPacSDKVersion(); //獲取 PacSDK 版本序號
PacNET = PACNET.Sys.GetPacNetVersion(); //獲取 PacNet 版本序號
```

3.1.5. pac_ChangeSlot

當在程式中欲操作不同槽位上的 87K 模組時，需先使用此函數來進行槽位變更後，方可繼續對該模組進行操作。

語法

C++

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

參數

slotNo

[in] 輸入欲切換的 87K 模組槽位編號。

回傳值

此函數沒有回傳值。

範例

[C]

```
BYTE slot;
HANDLE hPort;
BOOL RET;
char buf[Length] ;
hPort = uart_Open("");
pac_ChangeSlot(slot);
//切換到 87K 模組對應的槽位數
RET = uart_SendCmd(hPort, "$ 00M", buf);
//$ 00M：詢問設備名稱
uart_Close(hPort);
```

[C#]

```
BYTE slot;
IntPtr hPort;
bool ret;
string BUF;
hPort = PACNET.UART.Open(PACNET.MISC.AnsiString(""));
PACNET.Sys.ChangeSlot(slot);
//切換到 87 模組對應的槽位數
RET = PACNET.UART.SendCmd(hPort, PACNET.MISC.AnsiString("$ 00M"), BUF);
PACNET.UART.Close(hPort);
```

備註

為了能讓 PAC 上的系統知道當前要使用 Uart API 來操控哪個槽位上的模組(87K 系列)前，需先要以此 pac_ChangeSlot 函數進行槽位的轉換。

倘若槽位上的模組為 8K 系列，則可忽略此函數不用。

3.1.6. pac_CheckSDKVersion

此功能用於比較當前的 PACSDK.dll 的版本號與指定版本號狀況。

此功能不支持所有版本的 XPACSDK 和 WinPACSDK。

語法

C++

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

參數

Version

[in] 輸入欲比較的 PACSDK 的版本號。

如果版本號為 1.0.0.1 或以上，輸入的參數須變更為 0x01000001

回傳值

True: 當前使用的 PACSDK 版本比輸入的版本號更新。

False: 當前使用的 PACSDK 版本比輸入的版本號更舊。

範例

[C]

```
BOOL bVersion;
bVersion = pac_CheckSDKVersion(0x01000001);
//如果你的應用程序應該使用比 1.0.0.1 版本更新
If(!bVersion)
{
   MessageBox( “使用中的 PACSDK.dll 版本太舊” );
   //顯示一些警告，並關閉應用程序
}
```

[C#]

```
BOOL bVersion;
bVersion = PACNET.Sys.CheckSDKVersion(0x01000001);
//如果你的應用程序應該使用比 1.0.0.1 版本更新
If(!bVersion)
{
   MessageBox.show(“使用中的 PACSDK.dll 版本太舊”);
   //顯示一些警告，並關閉應用程序
}
```

3.1.7. pac_ModuleExists

此函數判斷某槽位上是否存在 I/O 模組，或判斷是否存在遠端模組。

語法

C++

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

參數

hPort

[in] 當透過 COM 埠詢問遠端模組時，需傳入用 uart_Open()的 hPort。

如果要詢問本機 slot 上的 I/O 模組，此欄位僅需填入 0。

slot

[in] 待入欲確認的 slot 編號。

如果是遠端模組，此欄位填入該 I/O 模組的 ID 值。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//檢查 slot5 是否有模組
BOOL bExist;
bExist = pac_ModuleExists(0, 5); //檢查 slot5 是否有 8K/9K 模組
HANDLE hPort= uart_Open("");
bExist = pac_ModuleExists(hPort, 5); //檢查 slot5 是否有 87K/97K 模組
uart_Close(hPort);
if(bExist)
{
   MessageBox("該模組存在!");
}
else
{
   MessageBox("該模組不存在!");
}
```

[C#]

```
//檢查 slot5 是否有模組
BOOL bExist;
bExist = PACNET.Sys.ModuleExists(0, 5); //檢查 slot5 是否有 8K/9K 模組
IntPtr hPort= PACNET.UART.Open(PACNET.MISC.AnsiString(“”));
bExist = PACNET.Sys.ModuleExists(hPort , 5); //檢查 slot5 是否有 87K/97K 模組
PACNET.UART.Close(hPort);
if(bExist)
{
   MessageBox.show(“該模組存在!”);
}
else
{
   MessageBox.show(“該模組不存在!”);
}
```

3.1.8. pac_GetOSVersion

獲取當前平台裝置的 OS 版本資訊。

語法

C++

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

參數

OS_VERSION

[out] 回傳該 OS 的版本編號。

回傳值

此函數沒有回傳值。

範例

[C]

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

[C#]

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

3.1.9. pac_GetMacAddress

獲取此裝置上網路埠的 MAC address。

語法

C++

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

參數

LAN

[in] 指定要查詢的 LAN 編號。

MacAddr

[out] 回傳該 LAN 編號所對應的 MAC address。

回傳值

此函數沒有回傳值。

範例

[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

透過此函數可對 PAC 進行不斷電暖開機。

語法

C++

```
void pac_ReBoot();
```

參數

此函數沒有參數。

回傳值

此函數沒有回傳值。

範例

[C]

```
pac_ReBoot();
```

[C#]

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

3.1.11. pac_GetCPUVersion

透過此函數可獲取當前 CPU 版本號。

語法

C++

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

參數

cpu_version

[out] CPU 的版本號。

回傳值

此函數沒有回傳值。

範例

[C]

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

[C#]

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

3.1.12. pac_EnableLED

透過此函數，可控制 LED 燈號開啟或關閉。

語法

C++

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

參數

bFlag

設定 LED 燈的模式。

True: 打開 LED 燈

False: 關閉 LED 燈

回傳值

此函數沒有回傳值。

範例

[C]

```
pac_EnableLED(true);
```

[C#]

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

3.1.13. pac_EnableLEDs

透過此函數，可以設置指定的 LED 燈號狀態。

語法

C++

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

參數

pin

指定的 LED 編號。

0: L1 LED 燈

1: L2 LED 燈

bFlag

指定的 LED 開關模式。

true: 指定的 LED 燈號開啟。

false: 指定的 LED 燈號關閉。

回傳值

此函數沒有回傳值。

範例

[C]

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

[C#]

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

3.1.14. pac_BackwardCompatible()

此功能決定了操作系統是否在向後兼容模式或無法正常工作。

語法

C++

```
Bool pac_BackwardCompatible();
```

參數

此函數沒有參數。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

備註

該功能僅適用於 WP-8000 系列所，它是用於後端兼容舊的 PAC 控制器的 WinCon 系列。

範例

[C]

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

[C#]

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

3.1.15. pac_GetEbootVersion

此函數檢索當前 EBOOT 的版本。

語法

C++

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

參數

eboot_version

[out] EBOOT 的版本。

回傳值

此函數沒有回傳值。

範例

[C]

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

[C#]

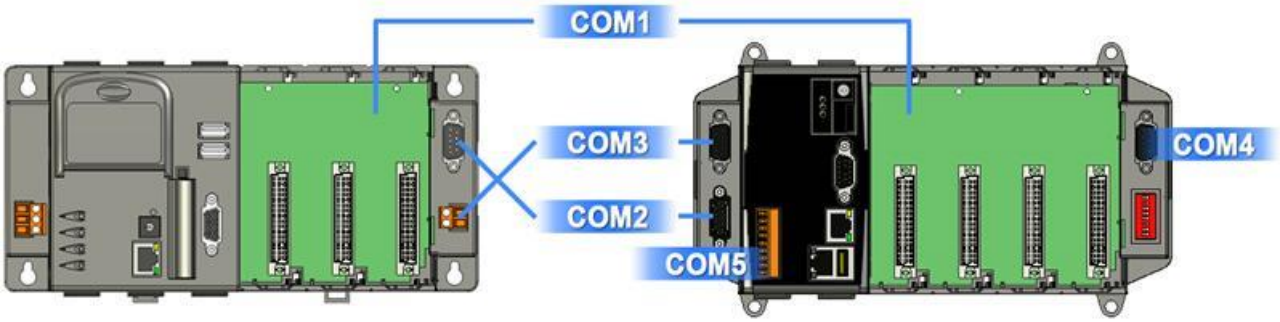
```
string Eboot;
EBOOT = PACNET.Sys.GetEbootVersion();
```

3.1.16. pac_GetComMapping

此函數檢索 COM port 的索引映射。

[普通模式]

[向後兼容模式]



語法

C++

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

參數

NDX

[in] 向後兼容與否之間指定哪個 slotCOM port 映射。

回傳值

返回 COM port 的索引下的向後兼容，如果向下兼容運行。否則，返回在正常模式下 COM port 的索引。

備註

該功能僅適用於 WP-8000 系列，它是用於後端兼容舊的 PAC 控制器的 WinCon 系列。

範例

[eVC]

```
int currentCOM; //當前 COM port
int normalCOM; //正常模式//COM port 索引
currentCOM = pac_GetComMapping(normalCOM);
//如果該設備正常模式下運行，則回傳值，currentCOM 等於 normalCOM
//例如：normalCOM = 0;呼叫這個 API 之後，currentCOM = 0
//否則，如果該設備是運行向下兼容模式，那麼 回傳值，currentCOM
//是向後兼容的映射索引
//例如：normalCOM = 0; 呼叫這個 API 之後，currentCOM = 1
```

[C#]

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

3.1.17. pac_GetModuleType

透過此函數檢索 WinPAC/XPAC 槽位上的 I/O 模組類型。

語法

C++

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

參數

slot

[in] 輸入欲查詢的 slot 編號。

回傳值

對應於 WinPAC-8000/ViewPAC 系列的回傳值

各回傳值代表的模組類型如下表。

值	描述
0	無模組存在
0x80	一般的 I-8000W 模組
0x81	I-8000RW 模組(R 代表：提供 PowerOnValue 及 SafeValue)
0xE3	32 DI 的 I-8000W 模組
0xE0	32 DO 的 I-8000W 模組
0xE2	16 DI 和 16 DO 的 I-8000W 模組
0xC3	16 DI 的 I-8000W 模組
0xC0	16 DO 的 I-8000W 模組
0xC2	8 DI 和 8 DO 的 I-8000W 模組
0x40	沒有定義的模組

對於 WinPAC 上-5000 系列

下表顯示了所定義的值

值	描述
0	無 XW 板存在
0x80	一般的 XW 板
0xE3	32 DI XW 板
0xE0	32 DO XW 板
0xE2	16 DI 和 16 DO XW 板
0xC3	16 DI XW 板
0xC0	16 DO XW 板
0xC2	8 DI 和 8 DO XW 板
0x40	無定義 XW 板

範例

[C]

```
int iDIO_Slot = 1;
int Type= 0;
Type = pac_GetModuleType(iDIO_Slot);
if(Type == 0xE2 | | Type == 0xC2){
   //該模組是 DIO 模組
   ...
}
```

[C#]

```
byte slot = 1;
int ModuleType = 0;
ModuleType = PACNET.Sys.GetModuleType(slot);
if(ModuleType == 0xE2 | | ModuleType == 0xC2){
   //該模組是 DIO 模組
   ...
}
```

3.1.18. pac_GetPacNetVersion

透過此函數獲取，當前平台所使用的 PACNET.dll 版本號。

語法

C++

```
string pac_GetPacNetVersion();
```

參數

此函數沒有參數。

回傳值

回傳字串為當前 PACNET.dll 的版本號。

範例

[C#]

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

3.1.19. pac_BuzzerBeep

該函數設定蜂鳴器發出聲響的次數與持續時間，可用來生成的簡單的蜂鳴器聲音輸出

語法

C++

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

參數

count

[in] 指定鳴叫次數。

milliseconds

[in] 指定的間隔時間，以毫秒為單位。

回傳值

此函數沒有回傳值。

範例

[C]

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

[C#]

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

3.1.20. pac_GetBuzzerFreqDuty

此函數取得蜂鳴器頻率值，與聲音的持續時間。

語法

C++

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

參數

freq

[out] 的聲音的頻率。

duty

[out] 聲音的持續時間。

回傳值

此函數沒有回傳值。

範例

[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

此功能設定頻率值和蜂鳴聲音的持續時間。

語法

C++

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

參數

freq

[out] 的聲音的頻率。

duty

[out] 聲音的持續時間。

回傳值

此函數沒有回傳值。

範例

[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

此功能將停止蜂鳴器。

語法

C++

```
void pac_StopBuzzer();
```

參數

此函數沒有參數。

回傳值

此函數沒有回傳值。

範例

[C]

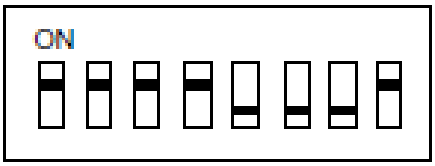
```
pac_StopBuzzer();
```

[C#]

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

3.1.23. pac_GetDIPSwitch

此函數取得 DIP 開關的數值。



語法

C++

```
int pac_GetDIPSwitch();
```

參數

此函數沒有參數。

回傳值

DIP 開關的回傳值。

範例

[C]

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

[C#]

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

3.1.24. pac_GetSlotCount

此功能將檢索 I/O slot 的總數。

語法

C++

```
int pac_GetSlotCount();
```

參數

此函數沒有參數。

回傳值

回傳是 I/O slot 的數目。

範例

[C]

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

[C#]

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

3.1.25. pac_EnableRetrigger

透過此函數來給定，slot 中斷訊號重新觸發器的狀態。

語法

C++

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

參數

iValues

[in] 指定重新觸發值，0～255，單位= 10 微秒。(0 表示禁用重新觸發功能)。

回傳值

此函數沒有回傳值。

範例

此函數沒有範例。

備註

當以下情況發生時適合使用重新觸發機制。

如果中斷被發送，但卻不被系統服務，重新觸發功能將再次發送一個中斷，此操作會一直持續直到該中斷被服務。

3.1.26. pac_GetBackplaneID

此函數檢索底板的 ID。

語法

C++

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

參數

backplane_version

[out] 回傳底板的 ID。

回傳值

此函數沒有回傳值。

範例

[C]

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

[C#]

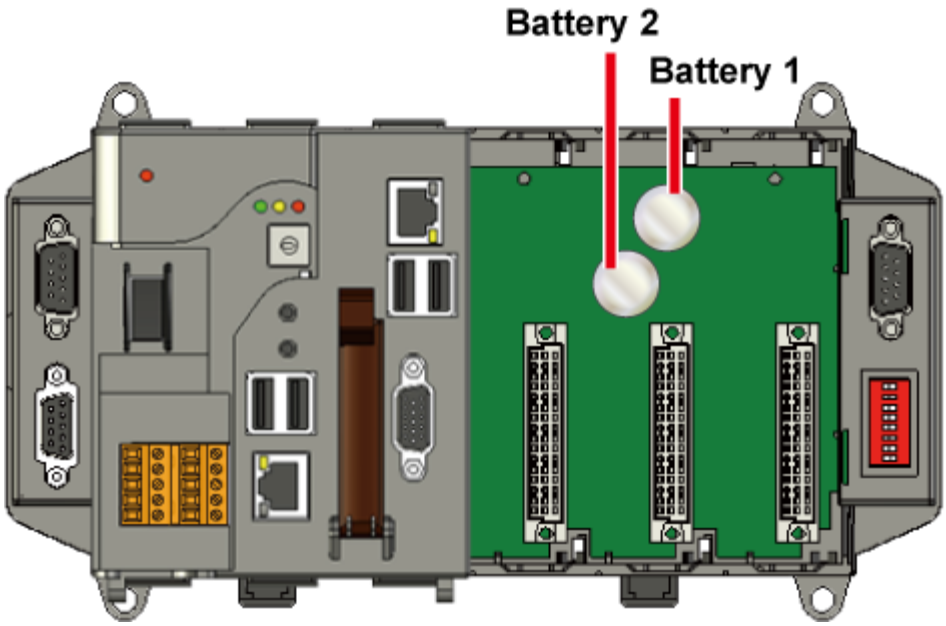
```
string Backplane= PACNET.Backplane.GetBackplaneID();
```

3.1.27. pac_GetBatteryLevel

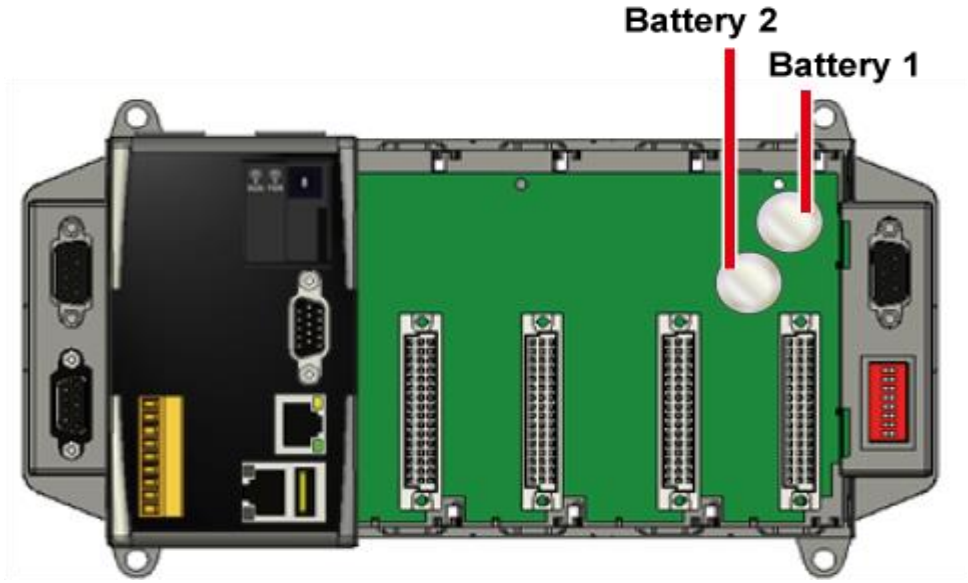
此函數檢索底板的電池電量狀態。

此功能支持以下系列機型。

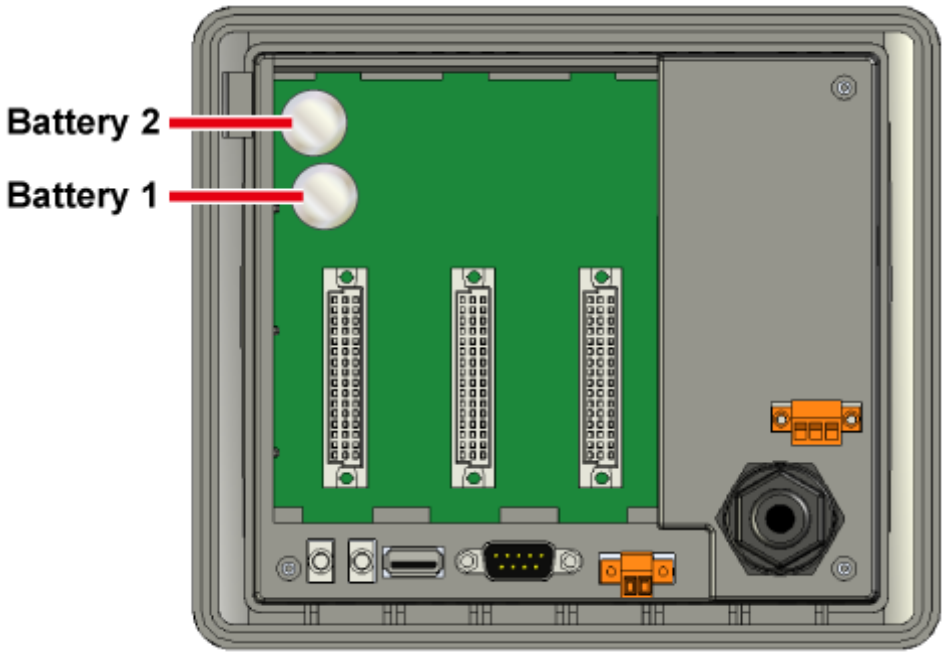
XPAC



WinPAC



ViewPAC



語法

C++

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

參數

nBattery

[in] 指定電池的索引值。

1 表示第一顆電池。

2 表示第二顆電池。

3 指 RTC 電池。(對於 XPAC_Atom 系列)

回傳值

1 表示高電壓。

0 表示低電壓。(僅適用於 XPAC 系列)

2 表示低電壓。(僅適用於 WinPAC 上系列)

範例

[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 測試)

功能寄存器開啟熱插拔註冊表。

語法

C++

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

參數

hWnd

[in] 指定的句柄標識。

回傳值

此函數沒有回傳值。

範例

此函數沒有範例。

3.1.29. pac_UnregistryHotPlug(Beta 測試)

關閉熱插拔註冊表鍵值。

語法

C++

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

參數

hWnd

[in] 指定的句柄。

回傳值

此函數沒有回傳值。

範例

此函數沒有範例。

3.1.30. pac_SetBackLight

設定 ViewPAC 系列產品的螢幕背光值。

語法

C++

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

參數

亮度範圍值: 0 ~ 100。

回傳值

此函數沒有回傳值。

範例

[C]

```
pac_SetBackLight(level);
```

[C#]

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

3.1.31. pac_GetBackLight

讀取 ViewPAC 系列產品的螢幕背光值。

語法

C++

```
DWORD pac_GetBackLight( );
```

參數

此函數沒有參數。

回傳值

背光亮度值。

範例

[C]

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

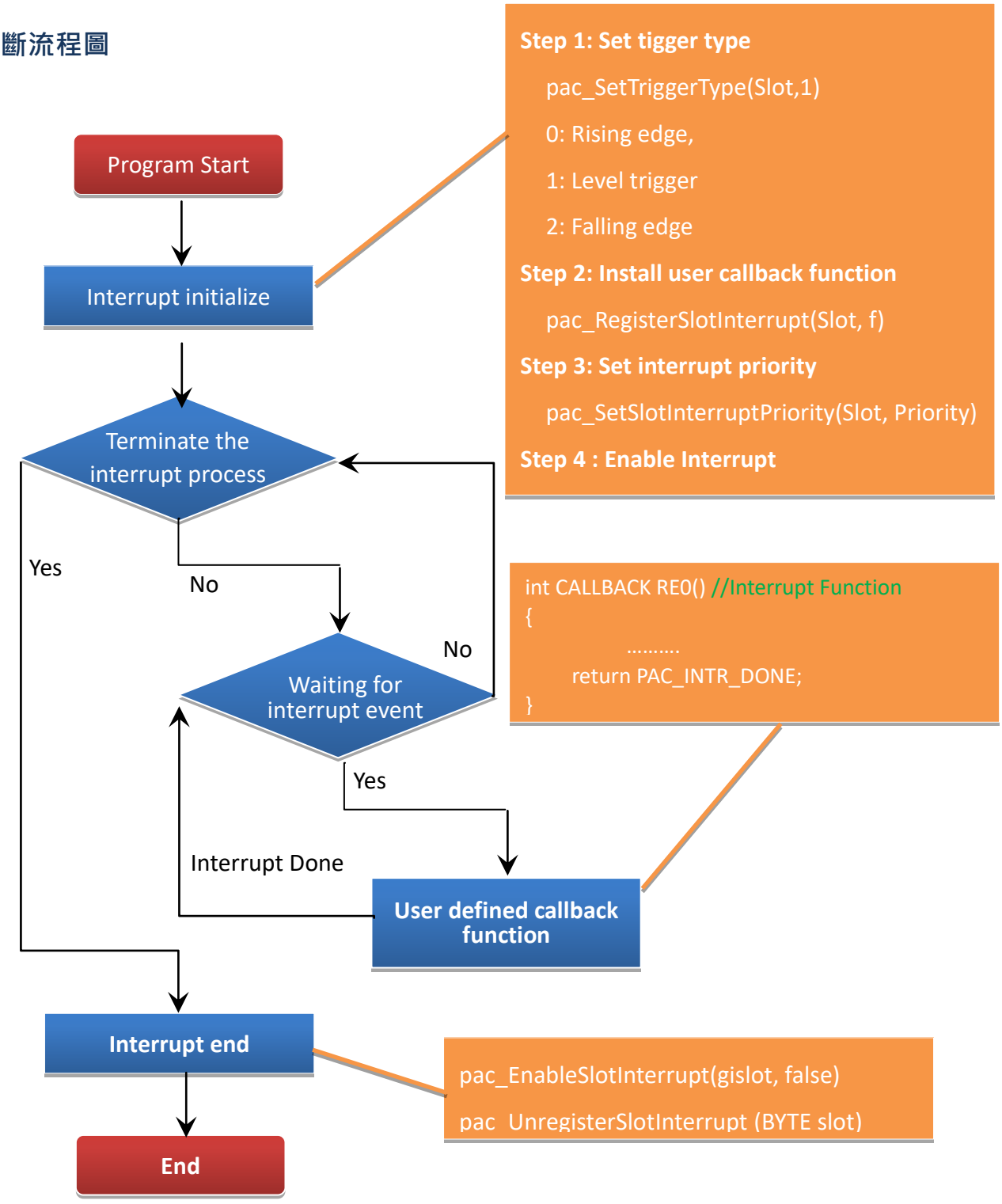
[C#]

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

3.2. Interrupt API (系統相關 API)

中斷函數提供 Slot 中斷操作，可用於計數、計時、檢測外部事件，以及使用串行接口發送和接收數據

中斷流程圖



支援的 PAC

以下列出 Interrupt API 函數所支援的 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

中斷函數

以下 API 函數用於支援檢索或設定插槽中斷。

PACSDK 函數	PACNET 函數	說明
pac_RegisterSlotInterrupt	Interrupt.RegisterSlotInterrupt	用來註冊指定 slot 中斷對應的中斷服務程式
pac_UnregisterSlotInterrupt	Interrupt.UnregisterSlotInterrupt	用於註銷指定的 slot 中斷服務程式及關閉 slot 中斷功能
pac_EnableSlotInterrupt	Interrupt.EnableSlotInterrupt	開啟指定的 slot 中斷功能
pac_SetSlotInterruptPriority	Interrupt.SetSlotInterruptPriority	設置指定的 slot 中斷線程(thread)的優先等級
pac_InterruptInitialize	Interrupt.InterruptInitialize	初始化一個 slot 中斷功能
pac_GetSlotInterruptEvent	Interrupt.GetSlotInterruptEvent	檢索 pac_InterruptInitialize 註冊的 slot 中斷事件句柄
pac_SetSlotInterruptEvent	Interrupt.SetSlotInterruptEvent	設定 slot 對應的中斷事件句柄
pac_SetTriggerType	Interrupt.SetTriggerType	設定各 slot 的中斷觸發類型
pac_GetSlotInterruptID	Interrupt.GetSlotInterruptID	檢索 slot 中斷的 ID
pac_InterruptDone	Interrupt.InterruptDone	此函數送信號至內核通知中斷處理已經完成

3.2.1. pac_RegisterSlotInterrupt

此函數用來註冊指定 slot 中斷對應的中斷服務程式。

語法

C++

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

參數

slot

[in] 槽位的索引值。

f

CALLBACK 函數。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
Int slot= 3; //slot3
int CALLBACK slot_callback_proc()
{
   return true;
   //如果 return true，SDK 將自動執行 pac_InterruptDone
   //否則，如果需要，用戶自行做 pac_InterruptDone。
   //如果中斷類型是電平觸發，不管 return true 或 FLASE，
   //無需添加 pac_InterruptDone，它會正常工作。
}
void CIntrDlg :: OnButton1()
{
   pac_RegisterSlotInterrupt(slot，slot_callback_proc);
   pac_EnableSlotInterrupt(slot，TRUE); //啟用中斷 slot
}

void CIntrDlg :: OnButton2()
{
   pac_EnableSlotInterrupt(slot，FALSE); //禁止中斷 slot
   pac_UnregisterSlotInterrupt(slot); //註銷 slot 中斷
}
```

備註(用於 XPAC 系列)

預設中斷觸發類型為電平觸發(level trigger)。

對於 XP-8000 系列，僅支持水平觸發類型(level trigger)。

3.2.2. pac_UnregisterSlotInterrupt

此函數用於註銷指定的 slot 中斷服務程式及關閉 slot 中斷功能。

語法

C++

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

參數

slot

[in] 槽位的索引值。

回傳值

如果函數調用成功，則回傳值是 TRUE。

如果函數失敗，則回傳值是 FALSE。若想獲得更多的錯誤信息，調用 pac_GetLastError。

範例

[C]

```
Int slot= 3; //slot3
int CALLBACK slot_callback_proc()
{
   //做一些事情
   pac_InterruptDone(slot);
   return false;
   //如果 return true，SDK 將自動執行 pac_InterruptDone
   //否則，如果需要，用戶應該做 pac_InterruptDone
   //如果中斷類型是電，不管 return true 或 FLASE，
   //無需添加 pac_InterruptDone，它會正常工作。
}
void CIntrDlg :: OnButton1()
{
   pac_RegisterSlotInterrupt(slot，slot_callback_proc);
   pac_EnableSlotInterrupt(slot，true); //啟用中斷 slot
}
void CIntrDlg :: OnButton2()
{
   pac_EnableSlotInterrupt(slot，false); //禁止中斷 slot
   pac_UnregisterSlotInterrupt(slot); //註銷 slot 中斷
}
```

3.2.3. pac_EnableSlotInterrupt

這個函數開啟指定的 slot 中斷功能。

語法

C++

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

參數

slot

[in] 槽位的索引值。

bEnable

[in] 指定 slot 中斷開啟與否。

回傳值

此函數沒有回傳值。

範例

[C]

```
int slot= 3; //slot3
int CALLBACK slot_callback_proc()
{
   //做一些事情
   pac_InterruptDone(slot);
   return true;
   //如果 return true，SDK 將自動執行 pac_InterruptDone
   //否則，如果需要，用戶應該做 pac_InterruptDone
   //如果中斷類型是電平觸發，不管 return true 或 FLASE，
   //無需添加 pac_InterruptDone，它會正常工作。
}
void CIntrDlg :: OnButton1()
{
   pac_RegisterSlotInterrupt(slot，slot_callback_proc);
   pac_EnableSlotInterrupt(slot，true); //啟用中斷 slot
}

void CIntrDlg :: OnButton2()
{
   pac_EnableSlotInterrupt(slot，false); //禁止中斷 slot
   pac_UnregisterSlotInterrupt(slot); //註銷 slot 中斷
}
```

備註

默認觸發類型為電平觸發(level trigger)。

對於 XP-8000 系列，僅支持水平觸發類型。

3.2.4. pac_SetSlotInterruptPriority

此功能設置指定的 slot 中斷線程(thread)的優先等級。

語法

C++

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

參數

slot

[in] 槽位的索引值。

nPriority

[in] 線程優先等級。

此值的範圍從 0 到 255，其中 0 為最高優先級。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

此函數沒有範例。

3.2.5. pac_InterruptInitialize

該函數初始化一個 slot 中斷功能。

語法

C++

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

參數

slot

[in] 槽位的索引值。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

此函數沒有範例。

備註

預設觸發類型為電平觸發(level trigger)。

對於 XP-8000 系列，僅支持水平觸發類型。

如果你想獲得已註冊的事件句柄，請調用 pac_GetSlotInterruptEvent API。

3.2.6. pac_GetSlotInterruptEvent

此函數檢索 pac_InterruptInitialize 註冊的 slot 中斷事件句柄。

語法

C++

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

參數

slot

[in] 槽位的索引值。

回傳值

如果函數成功，這個程序處理該事件的對象。

如果函數失敗，則回傳值是 NULL。若想獲得更多的錯誤信息，調用 pac_GetLastError。

範例

此函數沒有範例。

3.2.7. pac_SetSlotInterruptEvent

此功能設定 slot 對應的中斷事件句柄。

語法

C++

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

參數

slot

[in] 槽位的索引值。

hEvent

[in] slot 對應的中斷事件句柄。

回傳值

此函數沒有回傳值。

範例

此函數沒有範例。

3.2.8. pac_SetTriggerType

這個函數設定各 slot 的中斷觸發類型

語法

C++

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

參數

Slot

[in] 槽位的索引值。

ITYPE

[in] 指定中斷觸發類型。

0：上升沿觸發(rising edge trigger) (預設值)

1：電平觸發 (level trigger)

2：下降沿觸發 (falling edge trigger)

回傳值

此函數沒有回傳值。

範例

此函數沒有範例。

備註

對於 XP-8000 系列，僅支持水平觸發類型。

3.2.9. pac_GetSlotInterruptID

此函數檢索 slot 中斷的 ID。

語法

C++

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

參數

slot

[in] 槽位的索引值。

回傳值

如果函數調用成功，則回傳值是槽中斷的 ID。

如果函數失敗，則回傳值是 FALSE。若想獲得更多的錯誤信息，調用 pac_GetLastError。

範例

此函數沒有範例。

3.2.10. pac_InterruptDone

此函數送信號至內核通知中斷處理已經完成。

語法

C++

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

參數

slot

[in] 要被清除中斷觸發的槽位索引值。

回傳值

此函數沒有回傳值。

範例

[C]

```
HANDLE hIntr;
BOOL bExit = FALSE;
BYTE slot= 0;
DWORD INTP_Thread(PVOID PCONTEXT){
while(bExit)
{
   WaitForSingleObject(hIntr，INFINITE);
   //等待事件觸發
   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 (記憶體存取 API)

記憶體存取函數提供了，可用於讀取/寫，EEPROM/SRAM，或安裝/卸載 MicroSD 的管理功能

EEPROM 與 SRAM 的使用情境與應用:

可用來記錄某些資料，裝置斷電後也可保持資料。

支援的 PAC

以下列出 Memory Access API 函數所支援的 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_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 系列產品只支援 type 1 (EEPROM),沒有支援 type 0 (SRAM)。

記憶體存取函數

以下函數用於檢索或設定內存

PACSDK 函數	PACNET 函數	說明
pac_GetMemorySize	Memory.GetMemorySize	取得指定類型的記憶體大小。
pac_ReadMemory	Memory.ReadMemory	讀取指定類型的記憶體數據。
pac_WriteMemory	Memory.WriteMemory	寫入指定類型的記憶體數據。
pac_EnableEEPROM	Memory.EnableEEPROM	設置 EEPROM 的狀態為可讀寫。
pac_SDExists	Memory.SDExists	檢查的 micro SD 記憶卡，是否存在。
pac_SDMount	Memory.SDMount	掛載 micro SD 記憶卡
pac_SDOnside	Memory.SDOnside	檢查的 Micro SD
pac_SDUnmount	Memory.SDUnmount	卸載 micro SD 記憶卡，如果 Micro SD 已掛載。

3.3.1. pac_GetMemorySize

此函數取得指定類型的記憶體大小

語法

C++

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

參數

mem_type

指定當前的記憶體類型。

PAC_MEM_SRAM 0(WP-5000 系列不支持的 SRAM)

PAC_MEM_EEPROM 1

回傳值

回傳值指定類型的記憶體大小。

範例

[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

此函數從指定的記憶體類型中，讀取數據。

語法

C++

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

參數

address

指定從哪裡讀取的記憶體地址。

EEPROM

0～0x1FFF 的(8KB)為用戶區

0x2000～0x3FFF(8KB)被保留為系統

SRAM

允許被使用者使用的 SRAM 記憶體範圍為 0～0x6FFFF(448KB)，另 64KB SRAM 保留給系統使用。

lpBuffer

接收到記憶體中的數據。

dwLength

要讀取的長度。

mem_type

指定當前的記憶體類型。

PAC_MEM_SRAM 0 (WP-5000 系列不支持的 SRAM)

PAC_MEM_EEPROM 1

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。pac_GetLastError 在 PACERROR.h 定義為，一個非零的錯誤代碼指示故障。為了得到錯誤的一般描述，請使用 pac_GetErrorMessage。信息資源是可選的，因此，如果你調用 pac_GetErrorMessage 它可能會失敗。

範例

[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);
```

備註

如果使用者舊的程序將數據寫入到 EEPROM 0x2000~0x3FFF 之間的位址，或是使用 SDK v2.0.1.0 或更早的版本寫入數據至 SRAM 的最後一段位址。如果該程序改使用新版的 PACSDK.dll 或 PACNET.dll 時，可能無法將數據寫入 EEPROM 或 SRAM 內。

有兩種方法來解決這個問題：

1. 修改程序，以使數據改寫入至的 EEPROM 0～0x1FFF 的地址或 SRAM 0～0x6FFFF 地址。
2. 從 ICPDAS 取得舊版的 SDK。

3.3.3. pac_WriteMemory

此功能將數據，存儲在指定類型的記憶體中。

語法

C++

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

參數

address

指定從哪個記憶體地址寫入資料。

EEPROM

0～0x1FFF(8KB)為用戶區

0x2000～0x3FFF(8KB)被保留為系統區

SRAM

允許被使用者使用的 SRAM 記憶體範圍為 0～0x6FFFF(448KB)，另 64KB SRAM 保留給系統使用。

lpBuffer

一個指標，指向包含被寫入到記憶體中的數據的緩衝區。

dwLength

要寫入的資料長度。

mem_type

指定當前的記憶體類型。

PAC_MEM_SRAM 0 (WP-5000 系列不支持的 SRAM)

PAC_MEM_EEPROM 1

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。pac_GetLastError 在 PACERROR.h 定義為，一個非零的錯誤代碼指示故障。為了得到錯誤的一般描述，請使用 pac_GetErrorMessage。信息資源是可選的，因此，如果你調用 pac_GetErrorMessage 它可能會失敗。

範例

[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);
```

備註

如果使用者舊的程序將數據寫入到 EEPROM 0x2000~0x3FFF 之間的位址，或是使用 SDK v2.0.1.0 或更早的版本寫入數據至 SRAM 的最後一段位址。如果該程序改使用新版的 PACSDK.dll 或 PACNET.dll 時，可能無法將數據寫入 EEPROM 或 SRAM 內。

有兩種方法來解決這個問題：

1. 修改程序，以使數據改寫入至的 EEPROM 0～0x1FFF 的地址或 SRAM 0～0x6FFFF 地址。
2. 從 ICPDAS 取得舊版的 SDK。

3.3.4. pac_EnableEEPROM

此功能設置 EEPROM，關閉防寫入保護狀態。

語法

C++

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

參數

bEnable

指定了 EEPROM 的模式。

True: 關閉 EEPROM 防寫入保護狀態

False: 開啟 EEPROM 防寫入保護狀態

回傳值

此函數沒有回傳值。

範例

[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);
```

備註

在對 EEPROM 做寫入資料前，需要解除 EEPROM 的保護。

寫入資料到 EEPROM 完成後，需要開啟 EEPROM 保護。

3.3.5. pac_SDExists

這個函數會檢查 micro SD 卡，是否存在。

語法

C++

```
bool pac_SDExists();
```

參數

此函數沒有參數。

回傳值

如果 micro SD 卡存在，則回傳 TRUE。

如果 micro SD 卡不存在，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

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

[C#]

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

3.3.6. pac_SDMount

這個函數會掛載 Micro_SD 卡。

(在 Micro_SD 卡以插入裝置中，但是沒被掛載的情況下使用)

語法

C++

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

參數

szPartitionName

磁碟分區的名稱。例如 Part00。

回傳值

如果函數掛載 Micro_SD 成功，則回傳 TRUE。

如果函數掛載 Micro_SD 失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

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

[C#]

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

3.3.7. pac_SDOnside

此功能檢查 Micro SD 是否在 On-side。

語法

C++

```
bool pac_SDOnside();
```

參數

此函數沒有參數。

回傳值

如果 Micro SD On-side，則回傳 TRUE。

如果 Micro SD 無 On-side，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

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

[C#]

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

3.3.8. pac_SDUnmount

此功能卸載的 Micro SD，如果 Micro SD 已被掛載。

語法

C++

```
bool pac_SDUnmount();
```

參數

此函數沒有參數。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

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

[C#]

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

3.4. Watchdog API (看門狗 API)

一個看門狗定時器是用來檢測和恢復 PAC 中故障的電子計時器。在正常操作時，電腦 PAC 重新復位看門狗定時器，以防止它超時。

如果由於硬件故障或程序錯誤，程序會無法重新復位看門狗，定時器將超時，並產生超時信號。此超時信號會重置系統。

看門狗操作包括基本的管理操作，如啟動，復位等。

WDT 的使用情境與應用:

看門狗定時器 (WDT) 的典型應用:
防止微處理器閉鎖，是 WDT 的一個典型應用，通常軟體有一個 "主循環" 程式，用其調用副程式以實現不同的任務。每次程式循環對 WDT 進行一次復位，如果任何原因造成程式循環操作失敗，看門狗定時器則發生超時，對系統進行復位。需要注意的是，WDT 不能檢測瞬間系統故障，按照定義，只有在 WDT 計數器達到預定的 時間間隔時才會復位處理器。正是這一原因，需要選擇一個最短超時週期，以便在系統失控前由 WDT 產生復位，使系統恢復正常工作。

支援的 PAC

以下列出 Watchdog API 函數所支援的 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

看門狗函數

以下函數用於檢索或設定看門狗。

PACSDK 函數	PACNET 函數	說明
pac_EnableWatchDog	Sys.WDT.EnableWatchDog	看門狗定時器啟動。
pac_DisableWatchDog	Sys.WDT.DisableWatchDog	停止看門狗定時器。
pac_RefreshWatchDog	Sys.WDT.RefreshWatchDog	看門狗定時器復位。
pac_GetWatchDogState	Sys.WDT.GetWatchDogState	查看看門狗定時器狀態為啟動還是停止。
pac_GetWatchDogTime	Sys.WDT.GetWatchDogTime	查看看門狗超時時間大小。
pac_SetWatchDogTime	Sys.WDT.SetWatchDogTime	看門狗定時器啟動與設定超時時間,或是於看門狗啟動狀態中，重新設定超時時間大小。

3.4.1. pac_EnableWatchDog

此函數啟動看門狗定時器，並設定超時時間。

語法

C++

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

參數

WDT

[in] 指定看門狗的類型：

PAC_WDT_HW 0

PAC_WDT_OS 1

value

[in] 指定看門狗超時時間。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。在 PACERROR.h 定義一個非零的錯誤代碼指示故障。為了得到錯誤的一般描述，請調用 pac_GetErrorMessage。信息資源是可選的，因此，如果你調用 pac_GetErrorMessage，它可能會失敗 (**沒有對應的錯誤描述**)。

範例

[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);
```

備註

PAC_WDT_OS 看門狗:

PAC_WDT_OS 觸發時，由 CPU 看門狗模組發出重啟訊號

此看門狗的第二個參數的單位是秒，最大值是 1321 秒(約 22 分鐘)。

(適用於 XPAC 系列與 WinPAC 系列)

PAC_WDT_HW 看門狗:

PAC_WDT_HW 觸發時，由獨立看門狗模組發出重啟訊號

適用於 XPAC 系列

此第二個參數是一個值，它是在 1～63 單位。

一個單位是 0.5 秒左右。1 表示最短的超時，63 是最長的，大約需要 30 秒。

適用於 WinPAC 系列

此第二個參數是一個值，該值介於 1～31 單位。

一個單位大約是 200 毫秒。1 表示最短的超時時間，31 是最長的，它需要大約 6.2 秒。

註: 看門狗超時時間不能為零。兩種看門狗，都具有電子計時器，監視模組與硬體復位電路。

3.4.2. pac_DisableWatchDog

此函數將停止看門狗定時器。

語法

C++

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

參數

WDT

[in] 指定看門狗類型：

PAC_WDT_HW 0

PAC_WDT_OS 1

回傳值

此函數沒有回傳值。

範例

[C]

```
pac_DisableWatchDog(PAC_WDT_OS);
```

[C#]

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

3.4.3. pac_RefreshWatchDog

此函數復位該類型看門狗的計時器。

注意:

使用時，必須放在軟體的主循環程式碼中，以免發生當機時，無法觸發看門狗系統復位操作

語法

C++

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

參數

WDT

[in] 指定看門狗類型：

PAC_WDT_HW 0

PAC_WDT_OS 1

回傳值

此函數沒有回傳值。

範例

[C]

```
pac_RefreshWatchDog(PAC_WDT_OS);
```

[C#]

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

3.4.4. pac_GetWatchDogState

此函數檢索看門狗狀態啟動還是停止。

語法

C++

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

參數

WDT

[in] 指定看門狗類型：

PAC_WDT_HW 0

PAC_WDT_OS 1

回傳值

如果看門狗狀態啟動中，則回傳 TRUE。

如果看門狗狀態停止中，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[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

此函數取得看門狗超時時間。

語法

C++

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

參數

WDT

[in] 指定看門狗類型：

PAC_WDT_HW 0

PAC_WDT_OS 1

回傳值

回傳 pac_EnableWatchDog 或 pac_SetWatchDogTime 看門狗超時時間。

回傳 OS 看門狗的單位為秒。

回傳 XPAC(WinPAC)硬件看門狗的單位為 0~63(0~31)個單位。

(每單位時間大小，請參考前面說明)。

範例

[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

此功能啟動看門狗定時器與設定超時時間,或於看門狗啟動狀態中，重新設定超時時間大小。

語法

C++

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

參數

WDT

[in] 指定看門狗類型：

PAC_WDT_HW

PAC_WDT_OS

value

[in] 指定看門狗超時時間。

回傳值

此函數沒有回傳值。

範例

[C]

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

[C#]

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

備註

PAC_WDT_OS 看門狗:

PAC_WDT_OS 觸發時，由 CPU 看門狗模組發出重啟訊號

此看門狗的第二個參數的單位是秒，最大值是 1321 秒(約 22 分鐘)。

(適用於 XPAC 系列與 WinPAC 系列)

PAC_WDT_HW 看門狗:

PAC_WDT_HW 觸發時，由獨立看門狗模組發出重啟訊號

適用於 XPAC 系列

此第二個參數是一個值，它是在 1～63 單位。

一個單位是 0.5 秒左右。1 表示最短的超時，63 是最長的，大約需要 30 秒。

適用於 WinPAC 系列

此第二個參數是一個值，該值介於 1～31 單位。

一個單位大約是 200 毫秒。1 表示最短的超時時間，31 是最長的，它需要大約 6.2 秒。

註: 看門狗超時時間不能為零。兩種看門狗，都具有電子計時器，監視模組與硬體復位電路。

3.5. Registry API (註冊表 API)

註冊表操作，包括基本的管理操作，如讀取和寫入到註冊表中。以下主題介紹了如何創建，刪除或編程方式使用註冊表函數修改註冊表。

支援的 PAC

以下列出 Registry API 函數所支援的 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

註冊表函數

以下函數用於檢索或設定註冊表。

PACSDK 函數	PACNET 函數	說明
pac_RegCountKey	PAC_Reg.RegCountKey	檢索指定註冊表鍵中有多少個子鍵。
pac_RegCountValue	PAC_Reg.RegCountValue	檢索指定註冊表鍵中有多少個值項。
pac_RegCreateKey	PAC_Reg.RegCreateKey	建立指定註冊表鍵。
pac_RegDeleteKey	PAC_Reg.RegDeleteKey	刪除指定註冊表鍵。
pac_RegDeleteValue	PAC_Reg.RegDeleteValue	刪除指定註冊表鍵上的值項。
pac_RegGetDWORD	PAC_Reg.RegGetDWORD	擷取指定註冊表鍵上的 DWORD 值項。
pac_RegGetKeyByIndex	PAC_Reg.RegGetKeyByIndex	檢索註冊表鍵中指定索引的子鍵名稱。
pac_RegGetKeyInfo	PAC_Reg.RegGetKeyInfo	檢索註冊表鍵中指定索引的子鍵類型。
pac_RegGetString	PAC_Reg.RegGetString	擷取指定註冊表鍵上的字串值項。
pac_RegGetValueByIndex	PAC_Reg.RegGetValueByIndex	檢索註冊表鍵中指定索引的值項。
pac_RegKeyExist	PAC_Reg.RegKeyExist	判斷指定註冊表鍵存在與否。
pac_RegSave	PAC_Reg.RegSave	下令儲存註冊表所有設定值。
pac_RegSetString	PAC_Reg.RegSetString	寫入指定註冊表鍵上的字串值項。
pac_RegSetDWORD	PAC_Reg.RegSetDWORD	寫入指定註冊表鍵上的 DWORD 值項。

3.5.1. pac_RegCountKey

此函數將檢索指定註冊表鍵中有多少個子鍵。

語法

C++

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

參數

KEYNAME

[in] 指定註冊表鍵的絕對路徑。

回傳值

返回包含由指定註冊表鍵的子鍵的數目。

範例

[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

此函數將檢索指定註冊表鍵中有多少個值項。

語法

C++

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

參數

KEYNAME

[in] 指定註冊表鍵的絕對路徑。

回傳值

返回與該鍵上的值項數目。

範例

[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

這個函數建立指定註冊表鍵。

語法

C++

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

參數

KEYNAME

[in] 要創建註冊表鍵的絕對路徑。

回傳值

如果建立成功，則回傳 TRUE。

如果建立失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[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

此功能刪除指定的註冊表鍵。

語法

C++

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

參數

KEYNAME

[in] 要刪除註冊表鍵的絕對路徑。

回傳值

如果刪除成功，則回傳 TRUE。

如果刪除失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

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

[C#]

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

備註

如果刪除不存在的鍵，會回傳 false。如果函數調用成功，函數將刪除指定的鍵，包括其所有子項和值。目前應用程序開啟正在使用的鍵,不能調用 RegDeleteKey 刪除此鍵。

3.5.5. pac_RegDeleteValue

此功能刪除指定註冊表鍵上的值項。

語法

C++

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

參數

KEYNAME

[in] 要刪除註冊表值的路徑。

回傳值

如果刪除成功，則回傳 TRUE。

如果刪除失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

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

[C#]

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

備註

該功能只能用於葉鍵(leaf Key)。 函數 pac_RegDeleteKey 可以在葉鍵之外的任何註冊表中使用。 如果你要刪除一個值項，你應該調用 pac_RegDeleteValue。

3.5.6. pac_RegGetDWORD

此函數擷取指定註冊表鍵上的 DWORD 值項。

語法

C++

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

參數

KEYNAME

[in] 註冊表的絕對路徑。

回傳值

返回的標示鍵的 DWORD 值。

範例

[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

此函數檢索註冊表鍵中指定索引的子鍵名稱。

語法

C++

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

參數

KEYNAME

[in] 註冊表鍵的絕對路徑。

dwIndex

[in] 註冊表子鍵的具體指標。

lpName

[out] 指定一個緩衝區，以獲取鍵名。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[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

此函數檢索註冊表鍵中指定索引的子鍵類型。

語法

C++

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

參數

KEYNAME

[in] 註冊表鍵的絕對路徑。

回傳值

我們定義了四種類型的有關回傳值：

PKT_NONE 0

PKT_KEY 1

PKT_STRING 2

PKT_DWORD 3

範例

[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

此函數擷取指定的註冊表鍵上的字串值項。。

語法

C++

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

參數

KEYNAME

[in] 註冊表鍵的絕對路徑。

lpData

[out] 指向接收該字串值項的數據緩衝區的指標。

dwLength

[in] 指定 lpData 的字串長度。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[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

此函數檢索註冊表鍵中指定索引的值項。

語法

C++

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

參數

KEYNAME

[in] 註冊表鍵的絕對路徑。

dwIndex

[in] 值項具體的索引值。

lpName

[out] 指向接收該值項的數據緩衝區的指標。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[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

此功能判斷指定的註冊表鍵存在與否。

語法

C++

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

參數

KEYNAME

[in] 要檢查是否存在的註冊表鍵絕對路徑。

回傳值

如果存在，則回傳 TRUE。

如果不存在，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[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

該函數，下令儲存註冊表所有設定值。

語法

C++

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

參數

KEYNAME

[in] :

HKEY_CLASSES_ROOT

HKEY_CURRENT_USER

HKEY_LOCAL_MACHINE

HKEY_USERS

回傳值

如果儲存成功，則回傳 TRUE。

如果儲存失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

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

[C#]

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

3.5.13. pac_RegSetString

寫入指定註冊表鍵上的字串值項。

語法

C++

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

參數

KEYNAME

[in] 要寫入字串值項的註冊表鍵絕對路徑。

assignStr

[in] 要寫入的字串值。

dwLength

[in] 字串長度。

回傳值

如果寫入成功，則回傳 TRUE。

如果寫入失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[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

寫入指定的註冊表鍵上的 DWORD 值項。。

語法

C++

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

參數

KEYNAME

[in] 要寫入 DWORD 值項的註冊表鍵絕對路徑。

assignStr

[in] 要寫入的數值。

回傳值

如果寫入成功，則回傳 TRUE。

如果寫入失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[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);
```

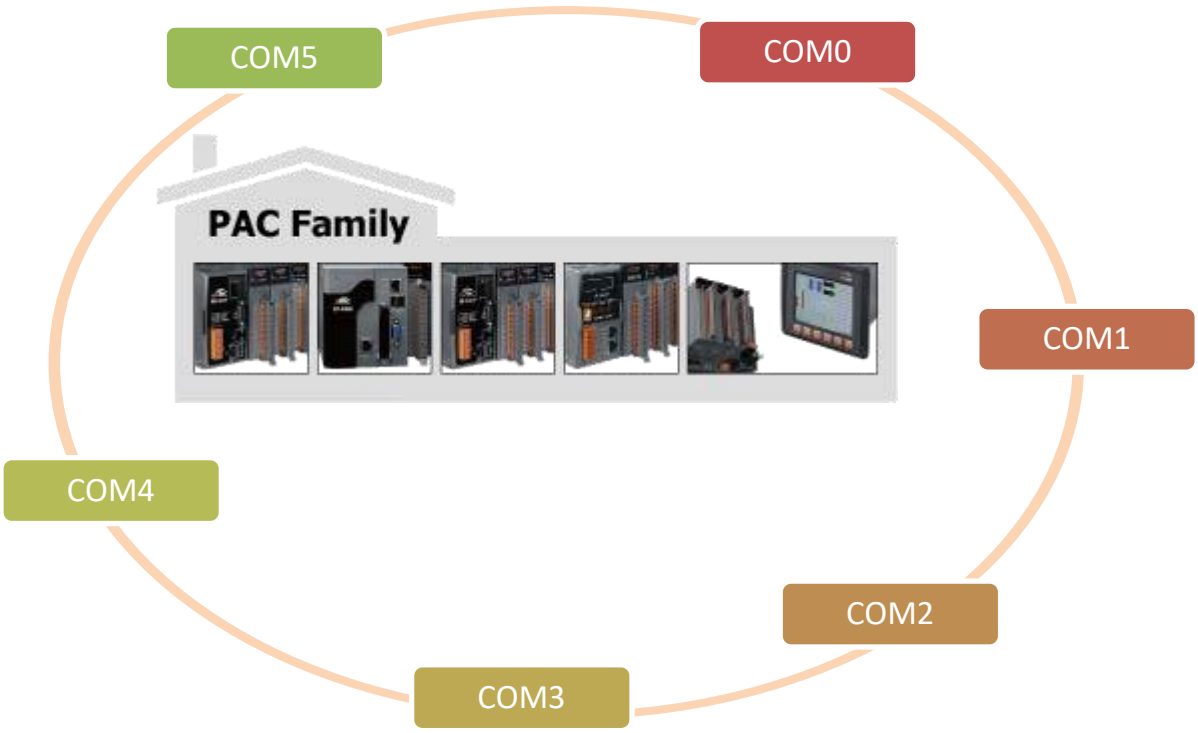
3.6. UART API (COM Port 通訊 API)

UART 操作包括基本的管理操作，如打開，發送，接收和關閉函數。以下主題介紹了如何使用 UART 功能操作，與 UART 編程。

我們提供幾個 COM port 功能(uart_Send / uart_Recv ...)與泓格模組(I-87K 系列，I-811xW/I-814xW 系列，I-7000 系列)進行通信。
所有的功能都是基於 C++中(的 CreateFile / CloseHandle 函數/ WriteFile 的/的 ReadFile/GetCommModemStatus .....)標準 COM port API 函數。
使用本節所述的這些功能與 I-87K 模組溝通。

每個 PAC 中預定義的 COM port

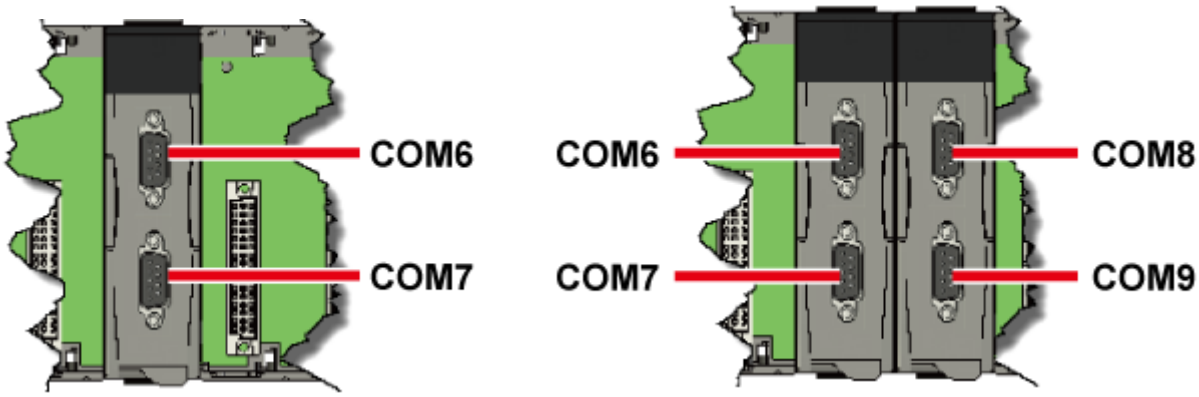
詳細的 PAC COM port 定義請參照附錄 F



每個 PAC 中的擴展 COM port

有些 PAC 有 I/O 插槽，可用來擴充通訊埠。

WP-8000 / ViewPAC (WinCE)

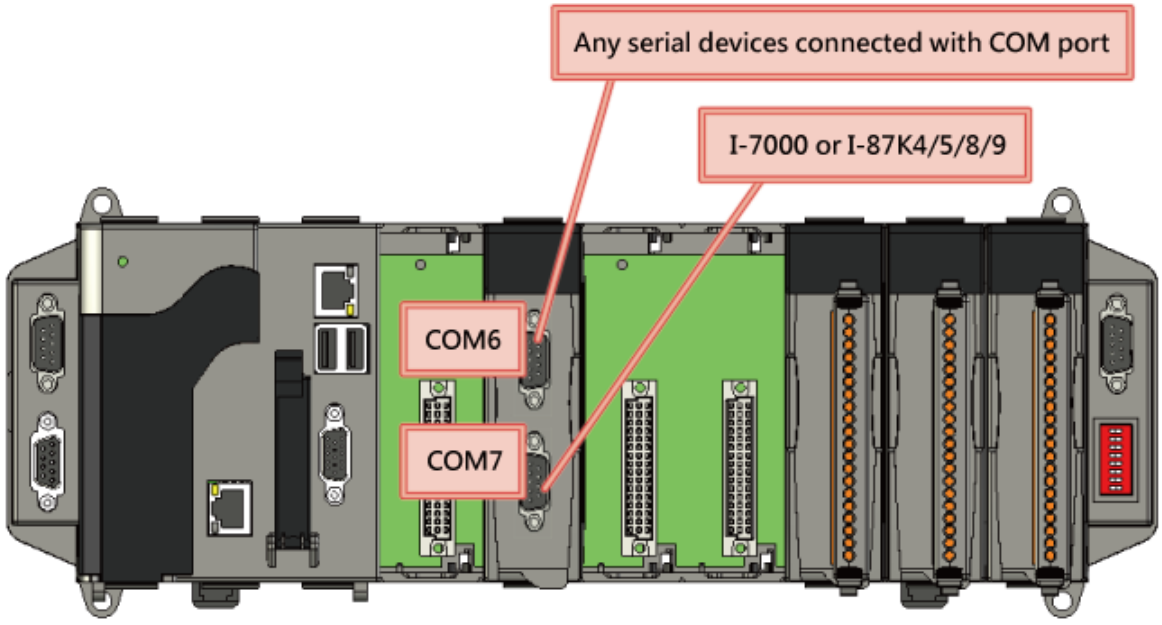


當高配置 I-87K 插入插槽時，請先呼叫函數 pac_ChangeSlot 變更至特定插槽，然後再進行其他操作。請參考示範「87k_Basic」。 關於 I-87K 指令（DCON 協定），請參考 http://ftp.icpdas.com/pub/cd/8000cd/napdos/dcon/io_module/87k_high_profile_modules.htm

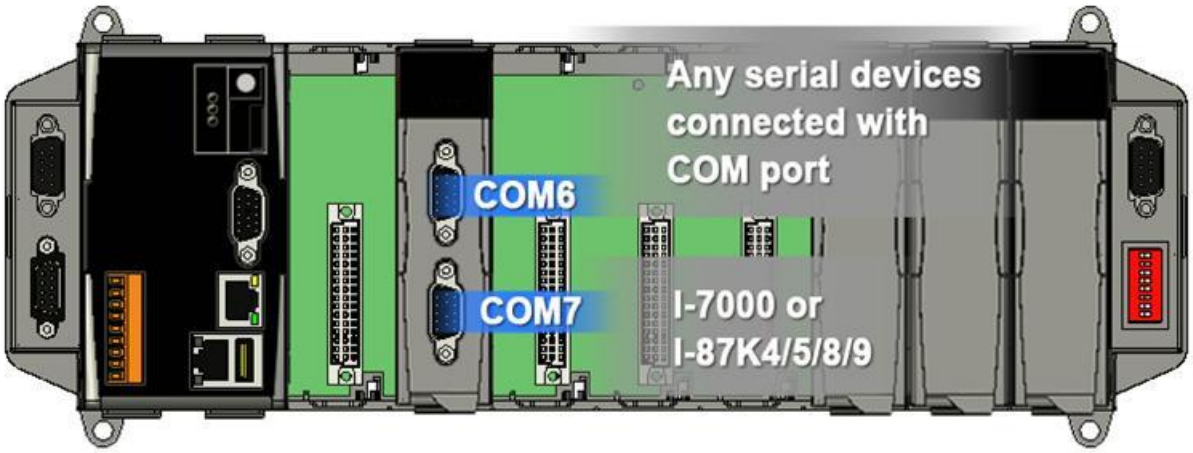
雖然使用者可以使用 UART API 來設定和讀取 I-87K 系列模組的值，但我們提供了更方便的 API 來執行此操作。請參閱第 7 節 PAC_IO API

使用本節的這些涵式，透過 I-811xW/I-814xW 系列模組與外部設備進行通信

XPAC



WP-8000 / ViewPAC (WinCE)



WP-5000

WP-5000 系列沒有用於插入 I-8K 和 I-87K 系列的插槽，但此部分的 UART API 也可用於 WP-5000 系列的 COM1/COM2/COM3/COM4。 除 COM1/COM2/COM3/COM4 外，所有功能均可透過 XW5xxx/XV5xxx(com0)擴充板與外部設備通訊。

要了解更多信息，請參閱用戶手冊 - 第 5 章 API 和演示參考。

支援的 PAC

以下列出 UART API 函數所支援的 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_Open	Y	Y	Y	Y	Y	Y	Y	Y	Y
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 函數

以下函數用於檢索或設定 UART。

PACSDK 函數	PACNET 函數	說明
uart_Open	UART.Open	打開 COM port，並指定波特率，奇偶校驗位，數據位和停止位。
uart_Close	UART.Close	關閉已打開的 COM port。
uart_SendExt	UART.SendExt	通過已打開的 COM port，發送字形式數據。
uart_Send	UART.Send	通過已打開的 COM port 發送數據。
uart_RecvExt	UART.RecvExt	接收字串+ 0x0D。 分配一個[0x0D] 字符，來終止字串。
uart_Recv	UART.Recv	通過已打開的 COM port 接收數據。
uart_SendCmdExt	UART.SendCmdExt	通過已打開的 COM port 發送命令。
uart_SetTimeOut	UART.SetTimeOut	設置超時。
uart_EnableCheckSum	UART.EnableCheckSum	啟動校驗。
uart_SetTerminator	UART.SetTerminator	設置終結符字符。
uart_BinSend	UART.BinSend	在考慮命令長度的情況下，發送帶或不帶空字的命令字串。
uart_BinRecv	UART.BinRecv	在考慮接收命令長度的情況下，接收帶或不帶空字符的回應字串資料。
uart_BinSendCmd	UART.BinSendCmd	發送二進制命令並接收固定長度的二進制數據。
uart_GetLineStatus	UART.GetLineStatus	讀取 COM port 的指定腳位狀態。
uart_GetDataSize	UART.GetDataSize	接收尚未由 uart Recv 操作讀取的 byte 數長度，或檢索為寫入操作傳輸的剩餘用戶數據。
uart_SetLineStatus	UART.SetLineStatus	設置 modem 線路的狀態。

3.6.1. uart_Open

此功能打開 COM port，並指定波特率，奇偶校驗位，數據位和停止位。

語法

C++

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

參數

ConnectionString

[in] 指定的 COM port，波特率，奇偶校驗位，數據位和停止位。

默認設置為 COM,1115200N,8,1。

ConnectionString 中的格式如下：

“com_portBAUD_RATE,parity_bits,data_bits,stop_bits”

警告：有每個參數之間沒有空格。

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

回傳值

已打開的 COM port 的句柄。非零表示成功。

如果函數失敗，則回傳值是 INVALID_HANDLE_VALUE。

(C/C+ +/MFC 程式 INVALID_HANDLE_VALUE 為 0xffffffff。.NET Framework 程式中，INVALID_HANDLE_VALUE 為-1)

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

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

[C#]

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

備註

如果 COM port 已經被其它程序開啟, uart_Open 功能無法此開啟的 COM port。

[使用 I-811xW/I-814xW 系列模組]

COM port 的名字是 COM6/COM7/MSA1/MSB1 ....。 MSAx / MSBx 是較早的舊用法，新用法為 COMx。

例如：

uart_Open(“COM6: ,9600,N,8,1”);

uart_Open(“MSA1: ,9600,N,8,1”);

有關如何設置 I-811xW/I-814xW 系列模組，請參考下面的手冊

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(2)_english.pdf

[使用 I-87K 系列模組]

僅使用 COM0 與 I-87K 系列模組進行通信。詳情請參閱 Sec.8 UART 的 API。

3.6.2. uart_Close

此功能，將關閉已打開的 COM port。

語法

C++

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

參數

hPort

[in] 已打開 COM port 的句柄。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[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);
```

備註

該函數指定的 COM port 被關閉之後，不無法再使用它，除非 COM port 被重新開啟。

3.6.3. uart_SendExt

此功能通過已打開的 COM port 發送一個字串數據。

當校驗通過 uart_EnableCheckSum 功能啟用時，兩個 byte 的校驗和自動添加到字串，字符 [0x0D] 被添加到字串末尾終止字串(BUF)。

此函數取代 uart_Send 功能。

語法

C++

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

參數

hPort

[in] 已打開 COM port 的句柄。

BUF

[in] 一個指向接收數據的陣列的指標。

out_Len

[in] 收數據陣列的長度，以 byte 為單位。

回傳值

如果函數成功，返回值為非零(TRUE)。

如果函數失敗，或正在完成同步，則返回值為零(FALSE)。

範例

[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);
```

備註

一個字串為“BUF”，在字串中不能包括空格字符。否則，該字串會被空格字符裁斷。例如：“$ 01M 02 03”用戶定義的字串。送出實際字串會變為“$ 01M” +0x0D。

這個函數會調用 PurgeComm()來清除 COM port 輸出緩衝區。該函數送出資料時，函數會自動加一個終止字符 0x0D 至資料最後面(請參閱 uart_SetTerminator 函數)。例如：(Check sum Disable 時)。在“BUF”五個 byte (ABCD +0x0)。此功能將發送五個 byte (ABCD +0x0D)。

3.6.4. uart_Send

此功能通過已打開的 COM port 發送數據。

該函數將發送一個字串。如果校驗通過 uart_EnableCheckSum 函數, 該功能會自動將兩個校驗 byte 加入字串。然後發送字串的末尾進一步添加[0x0D] 來表示該字串(BUF)的終止。

語法

C++

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

參數

hPort

[in] 已打開的 COM port 的句柄。

BUF

[in] 數據接受陣列的指標。

回傳值

如果函數成功, 返回值為非零(TRUE)。

如果函數失敗, 返回值為零(false)。

範例

[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);
```

備註

一個字串為 “BUF”，在字串中不能包括空格字符。否則，該字串會被空格字符裁斷。例如：“$ 01M 02 03” 用戶定義的字串。送出實際字串會變為 “$ 01M” +0x0D。

這個函數會調用 PurgeComm()來清除 COM port 輸出緩衝區。該函數送出資料時，函數會自動加一個終止字符 0x0D 至資料最後面(請參閱 uart_SetTerminator 函數)。例如：(Check sum Disable 時)。在 “BUF” 五個 byte (ABCD +0x0)。此功能將發送五個 byte (ABCD +0x0D)。

3.6.5. uart_RecvExt

此功能通過已打開的 COM port 獲取數據。這個函數接收一個字串+0x0D。[0x0D] 字符最後會自動被函數移除。

如果校驗通過 uart_EnableCheckSum 函數設定後, 該函數會自動檢查兩個校驗 byte 的字串。

此函數替換 uart_Recv。uart_Recv 可能會導致緩衝區溢出的一些情況。

語法

C++

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

參數

hPort

[in] 已打開的 COM port 的句柄。

BUF

[in] 一個指向接收數據的陣列的指標。

out_Len

[in] 接收數據的陣列長度, 以 byte 為單位

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

如果函數沒有收到一個字符 0x0D, 其他數據仍然存儲 “BUF”，但回傳值是 FALSE。調用 pac_GetLastError 函數, 將得到一個錯誤代碼(PAC_ERR_UART_READ_TIMEOUT)

如果這個函數來接收實際的數據大小比的 buf 緩衝區的長度較大, 它會返回 FALSE。調用 pac_GetLastError 功能將得到一個錯誤代碼 (PAC_ERR_UART_INTERNAL_BUFFER_OVERFLOW)

範例

[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);
```

備註

接收的該終止字符是 0x0D。(請參閱 uart_SetTerminator 函數。)

例如：

(Check sum is disabled)。當這個函數接收五個 byte 資料 (ABCD +0x0D)。在接收的變數“BUF”將取得五個 byte (ABCD +0x0), 函數會移除 0x0D。

當這個函數接收四個 byte (ABCD),沒有 0x0D。在“BUF”將是四個 byte (ABCD)。但 reurn 值是 False。

3.6.6. uart_Recv

此功能通過已打開的 COM port 獲取數據。

這個函數會收到一個字串+0x0D。等待一個字符[0x0D] 來表示一個字串的結束。然後如果校驗通過 uart_EnableCheckSum 函數後, 該函數會自動檢查兩個校驗 byte 的字串。

此功能將提供一個字串, 沒有最後一個 byte [0x0D] 。

語法

C++

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

參數

hPort

[in] 已打開的 COM port 的句柄。

BUF

[in] 一個指向接收數據的陣列的指標。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

如果這個函數不接受 character0x0D, 其他數據仍然存儲 “BUF”，但回傳值是 0。調用 pac_GetLastError 功能將得到一個錯誤代碼(pac_ERR_uart_READ_TIMEOUT)。

範例

[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);
```

備註

接收的該終止字符是 0x0D。(請參閱 uart_SetTerminator 函數。)

例如：

(Check sum is disabled)。當這個函數接收五個 byte 資料 (ABCD +0x0D)。在接收的變數“BUF”將取得五個 byte (ABCD +0x0), 函數會移除 0x0D。

當這個函數接收四個 byte (ABCD),沒有 0x0D。在“BUF”將是四個 byte (ABCD)。但 reurn 值是 False。

3.6.7. uart_SendCmdExt

此功能通過已打開的 COM port 發送命令。

這個功能是 uart_SendExt 和 uart_RecvExt 的組合。

用於發送一個命令的操作是一樣的 uart_SendExt。

用於接收響應的動作是一樣的 uart_RecvExt。

此函數替換 uart_SendCmd。該 uart_SendCmd 可能會導致緩衝區溢位的情況。

語法

C++

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

參數

hPort

[in] 已打開的 COM port 的句柄。

CMD

[in] 一個命令。

out_Len

[in] 命令字元長度。

szResult

[out] 一個指向接收數據陣列的指標。

in_Len

[in] 收數據陣列字元長度。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
HANDLE hOpen;
char buf[Length] ;
hOpen = uart_Open("COM1,9600,N,8,1");
ret = uart_SendCmdExt(hOpen,"$00M", 4, buf, Length); //$00M:  詢問裝置名稱
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);
```

備註

此函數調用 PurgeComm()來清除串行 COM port 的輸入和輸出緩衝器。

請參閱 uart_SendExt / uart_RecvExt 的更多的細節說明。

3.6.8. uart_SetTimeOut

這個函數設置超時計時器。

語法

C++

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

參數

hPort

[in] 已打開的 COM port 的句柄。

msec

[in] 超時時間, 單位為毫秒。

ctoType

[in] 指定如下超時定時器類型：

CTO_TIMEOUT_ALL 0

CTO_READ_RETRY_TIMEOUT 1

CTO_READ_TOTAL_TIMEOUT 2

CTO_WRITE_TOTAL_TIMEOUT 3

回傳值

此函數不返回值。

範例

[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);
```

備註

CTO_READ_TOTAL_TIMEOUT：

用於計算讀取操作的總超時周期的常數，以毫秒為單位。CTO_READ_TOTAL_TIMEOUT 的值為零，表示此功能不用於讀取操作。

CTO_WRITE_TOTAL_TIMEOUT：

用於計算寫入操作的總超時周期的常數，以毫秒為單位。CTO_WRITE_TOTAL_TIMEOUT 的值為零，表示此功能不用於寫入操作。

CTO_READ_RETRY_超時：

用於計算讀取操作的超時時間的常數，以系統 tick 計數。

CTO_TIMEOUT_ALL：

用於計算寫入和讀取操作的總超時周期的常數，以毫秒為單位。CTO_TIMEOUT_ALL 的值為零，表示此功能不用於寫入和讀取操作。

3.6.9. uart_EnableCheckSum

此功能開啟的檢查和關閉。

添加兩個校驗 byte，用來產生校驗和數據的結束。

語法

C++

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

參數

hPort

[in] 已打開的 COM port 的句柄。

bEnable

[in] 決定開啟或關閉校驗。

默認情況下是關閉。

回傳值

此函數沒有回傳值。

範例

[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

此函數設定終止字元符號。

語法

C++

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

參數

hPort

[in] 已打開的 COM port 的句柄。

szTerm

[in] 終止字元符號。

默認情況下為 CR(0x0D)。

回傳值

此函數沒有回傳值。

範例

[C]

```
HANDLE hPort;
char result[32] ;
hPort = uart_Open(""); //打開 COM0，格式為：115200, N, 8,1
uart_SetTerminator(hPort, "\r");
pac_ChangeSlot(0); //I-87K 模組在 slot0
uart_SendCmd(hPort, "$00M", result); //$00M：詢問裝置名稱
uart_Close(hPort);
```

[C#]

```
byte[] result = new byte[32] ;
IntPtr hPort = PACNET.UART.Open(""); //打開 COM0，格式為：115200,N,8,1
PACNET.UART.SetTerminator(hPort, PACNET.MISC.AnsiString("\r"));
PACNET. Sys.ChangeSlot(0); //I-87K 模組在 slot0
PACNET.UART.SendCmd(hPort, PACNET.MISC.AnsiString("$00M"), result); //$00M：詢問裝置名稱
string str = PACNET.MISC.WideString(result);
PACNET.UART.Close(hPort);
```

備註

此功能涉及到 uart_Send / uart_Recv / uart_SendCmd。

3.6.11. uart_BinSend

發送的命令字串是固定長度，它是由參數“in_Len”控制。這個函數和 uart_Send 之間的區別在於, uart_BinSend 發送的字串達到設定的長度條件即終止發送，而不是以字符“CR”(進位返回)條件來終止發送。因此，這個函數發送的字串無論是否包含空字符(0x0)，都會被送出，直到命令字串達到設定的長度為止。

此外，由於這個功能沒有任何錯誤檢查機制（校驗, CRC, LRC ...等)。如果需要的通信檢查系統，用戶必須於程式碼內添加錯誤檢查機制。

語法

C++

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

參數

hPort

[in] 已打開的 COM port 的句柄。

BUF

[in] 一個指標，它指向數據接收陣列。

in_Len

[in] 數據接收陣列的長度。

回傳值

如果函數調用成功，則回傳值是 1。

如果函數失敗，則回傳值是 0。

範例

[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);
```

備註

注意，這個函數通常用於與其他設備進行通信，但不適合泓格的 DCON(I-7000/8000/87K)系列模組。

這個函數會調用 PurgeComm()來清除串行 COM port 輸出緩衝區。

3.6.12. uart_BinRecv

這個函數應用於收到的固定長度的資料。接收資料的長度由參數 “in_Len” 控制。這個函數和 uart_Recv 之間的區別在於, uart_BinRecv 接收的字串達到設定的長度時即終止接收, 而不是以字符 “CR” (進位返回)來終止接收。

由於這個功能沒有任何錯誤檢查機制 (校驗, CRC, LRC ...等)。如果需要的通信檢查系統, 用戶必須於程式碼內添加錯誤檢查機制。

語法

C++

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

參數

hPort

[in] 已打開的 COM port 的句柄。

BUF

[out] 一個指標, 它指向接收數據陣列。

in_Len

[in] 結果字串的長度。

回傳值

如果函數調用成功, 則回傳值是 1。

如果函數失敗, 則回傳值是 0。

範例

[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);
```

備註

注意，這個函數通常用於與其他設備進行通信，但不適合泓格的 DCON(I-7000/8000/87K) 系列模組。

3.6.13. uart_BinSendCmd

該函數發送的二進制命令和接收二進制的固定長度數據。

這個功能是 uart_BinSend 和 uart_BinRecv 的組合。

用於發送一個命令的操作是一樣的 uart_BinSend。

用於接收回應的動作是一樣的 uart_BinRecv。

語法

C++

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

參數

hPort

[in] 句柄打開的 COM。

ByteCmd

[in] 一個命令。

in_Len

[in] 命令字串的長度。

ByteResult

[out] 一個指標，它指向接收數據陣列。

out_Len

[in] 結果字串的長度。

回傳值

如果函數調用成功，則回傳值是 1。

如果函數失敗，則回傳值是 0。

範例

[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);
```

備註

這個函數會調用 PurgeComm()來清除串行 COM port 輸出和輸入緩衝器。

3.6.14. uart_GetLineStatus

此函數取得 COM port 的指定腳位狀態。

語法

C++

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

參數

hPort

[in] 已打開的 COM port 的句柄。

Pin

[in] 指定 COM port 的腳位。

此參數可以以設定如下：

CTS 0

DSR 1

RI 2

CD 3

回傳值

TRUE 表示該引腳的狀態為 ON, 0 表示 OFF。

範例

[C]

```
HANDLE hPort = uart_Open(“COM5,115200,N,8,1”);
BOOL ret = uart_GetLineStatus(hPort, DSR); //指定腳位，例如: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” ); //指定腳位，例如: DSR
bool ret = PACNET.UART.GetLineStatus(hPort, 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

此函數取得輸入/輸出緩衝區中，尚未”uart_Recv 讀取”/ ”發送”的字節數。

語法

C++

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

參數

hPort

[in] 已打開的 COM port 的句柄。

DATA_TYPE

[in] 指定進或出緩衝區。

參考下值：

＃定義 IN_DATA 0

＃定義 OUT_DATA 1

回傳值

輸入/輸出緩衝區的 byte 數, 但尚未讀/寫。

3.6.16. uart_SetLineStatus

此功能設定 COM port 的指定腳位的狀態。

語法

C++

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

參數

hPort

[in] 已打開的 COM port 的句柄。

pin

[in] 指定 COM port 的腳位。

可以設定下值：

＃定義 DTR 1

＃定義 RTS 2

＃定義 DTR RTS + 3

mode

[in] 0：禁用，設置引腳信號為 OFF

1：啟用，設置引腳信號為 ON

回傳值

如果函數調用成功，則回傳值不為零。

如果函數失敗，則回傳值是零。

為了得到一個錯誤代碼，調用 pac_GetLastError。在 PACERROR.h 定義一個非零的錯誤代碼指示故障。如果錯誤代碼 pac_GetLastError 得到的是 PAC_ERR_UART_GET_COMM_STATUS_ERROR，調用 GetLastError 函數獲取最後一個錯誤 OCDE 的 Windows API 函數。

範例

[C]

```
HANDLE hPort = uart_Open(“COM5, 9600, N, 8,1”); //XPAC 上的 COM5 DTR pin
//HANDLE hPort = uart_Open(“COM4, 9600, N, 8,1”); //WinPAC 上的 COM4 DTR pin
uart_SetLineStatus(hPort, 1, 1); //設置 DTR 為 ON
uart_Close(hPort);
```

[C#]

```
IntPtr 的 hPort = PACNET.UART.Open(“COM5, 9600, N, 8,1”); //XPAC 上的 COM5 DTR pin
//IntPtr 的 hPort = PACNET.UART.Open(“COM4, 9600, N, 8,1”); //XPAC 上的 COM4 DTR pin
PACNET.UART。SetLineStatus(hPort, 1,1); //設置 DTR 為 ON
PACNET.UART.Close(hPort);
```

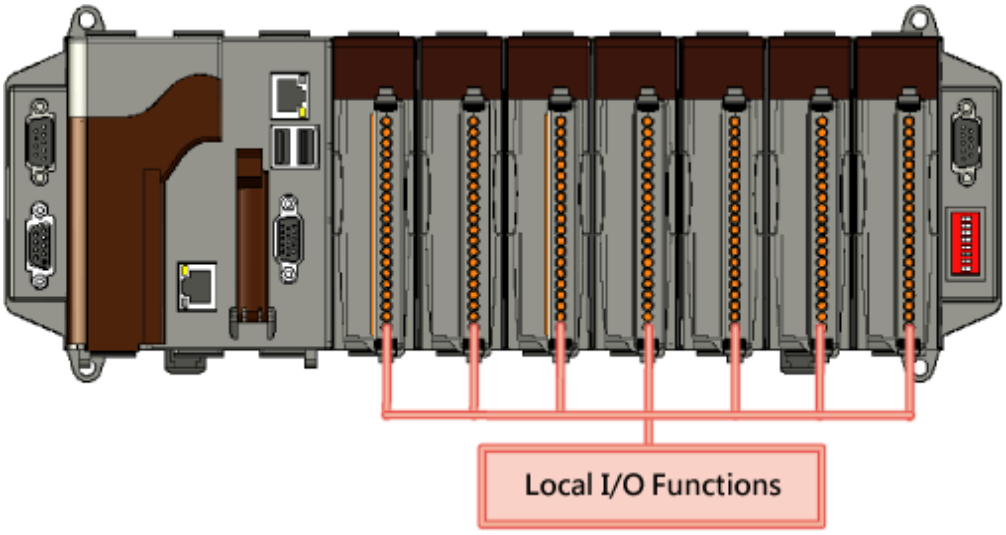
3.7. PAC_IO API (I/O 模組控制 API)

PAC_IO API 支持操作 I/O 模組，它可以分為以下幾部分：

Local (I/O in slot)

本地模式，控制 slot 上的 I/O 模組，Slot 範圍為從 0 到 7，如下

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

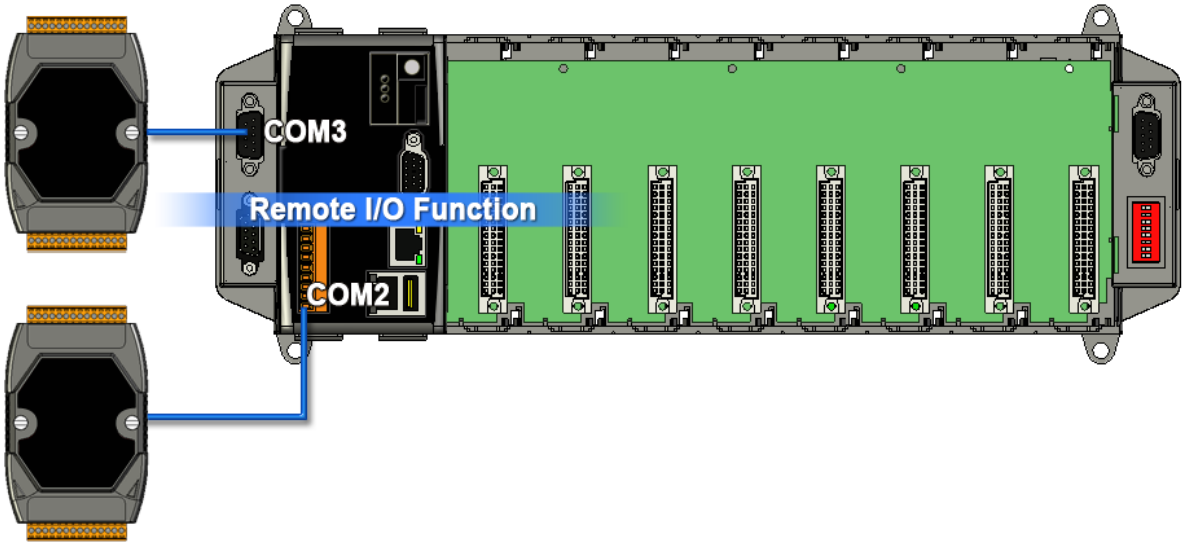


Remote

遠端模式，控制時需要遠端模組的位置(範圍是從 0 到 255)，PAC_REMOTE_IO。如下：

```
//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);
```



在 PACSDK.dll，修改處理發送 DCON 命令，而不確定的模組名稱，新 PAC_IO API 函數可以支持訪問 I-87K/I-8K，I-7000，I- 8000，TM 系列，等 DCON 模組。

有關 XPAC/WinPAC 上擴充模組的更多信息，請參閱:

I-8K/I-87K 系列

http://www.icpdas.com/products/PAC/i-8000/8000_IO_modules.htm

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

I-7000 系列 xxx

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

I-8K 單位 xx

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

TM 系列

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

API 函數的多功能 DCON 模組

PAC_IO 的 API 提供了 2 種功能。

第一種，包括 pac_WriteDO， pac_ReadDIO， pac_ReadDI， pac_ReadDO， pac_ReadDIO_DIBit，pac_ReadDIO_DOBit，pac_ReadDIBit，Pac_ReadDOBit，pac_ReadDICNT 和 pac_ClearDICNT，pac_ReadDICNTAll 和 pac_ClearDICNTAll 等函數是給只具有 DI 或 DO 通道之 DCON 模組使用。

另一種，包括 pac_WriteDO_MF， pac_ReadDIO_MF 和 pac_ReadDI_MF 等函數，用於訪問多功能 DCON 模組，(多功能模組是指 AI/AO 或 Counter 模組具有 DI 或 DO 通道)，如 I-87005W/I-87016W/I-87082W/I-7016/I-7088 等模組。

兩種的函數(pac_WriteDO 和 pac_WriteDO_MF)的指示放置，因為定義在相同的 參數 和 回傳值，函數是一樣的。

用於詢問 DIO DCON 模組的函數(pac_WriteDO， pac_ReadDIO 等)，不能被用來詢問多功能 DCON 模組。如果使用該函數詢問多功能 DCON 模組，該函數將 return 0x14003 其意思為 “Uart response error “，反之亦然。

支援的 PAC

以下列出 System information API 函數所支援的 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

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_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

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_ReadModuleSafeValueDO/pac_ReadModuleSafeValueDO_MF	Y	-	Y	-	Y	Y	Y	-	Y
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

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_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
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 函數

以下函數用於檢索或設定 I/O 模組。

PACSDK 函數	PACNET 函數	說明
pac_GetBit	PAC_IO.GetBit	檢索參數值的特定位元值。常利用於 DI/DO 模組讀回狀態值時，分拆指定通道的狀態。
pac_WriteDO/pac_WriteDO_MF	PAC_IO.WriteDO/PAC_IO.WriteDO_MF	寫入 DO 值，至 DO 模組。
pac_WriteDOBit	PAC_IO.WriteDOBit	寫入 DO 值的一個位元，至 DO 模組。 只有對應於 DO 值的該位元 DO 通道被改變。
pac_ReadDO/pac_ReadDO_MF	PAC_IO.ReadDO/PAC_IO.ReadDO_MF	讀取 DO 模組的 DO 輸出值。
pac_ReadDI/pac_ReadDI_MF	PAC_IO.ReadDI/PAC_IO.ReadDI_MF	讀取 DI 模組的 DI 值。
pac_ReadDIO/pac_ReadDIO_MF	PAC_IO.ReadDIO/PAC_IO.ReadDIO_MF	讀取 DIO 模組的 DI 和 DO 值。
pac_ReadDILatch	PAC_IO.ReadDILatch	讀取 DI 模組的 DI 鎖存值。
pac_ClearDILatch	PAC_IO.ClearDILatch	清除 DI 模組的鎖存值。
pac_ReadDIOLatch	PAC_IO.ReadDIOLatch	讀取 DIO 模組的 DI 的鎖存值和 DO 的通道輸出值
pac_ClearDIOLatch	PAC_IO.ClearDIOLatch	清除 DIO 模組的 DI 的鎖存值和 DO 的通道輸出值
pac_ReadDICNT/pac_ReadDICNT_MF	PAC_IO.ReadDICNT/PAC_IO.ReadDICNT_MF	讀取的 DI 模組的 DI 通道的計數器值。
pac_ClearDICNT/pac_ClearDICNT_MF	PAC_IO.ClearDICNT/PAC_IO.ClearDICNT_MF	清除 DI 模組的 DI 通道的計數器值。
pac_ReadDICNTAll	PAC_IO.ReadDICNTAll	讀取 DI 模組所有的 DI 通道的計數器值。
pac_ClearDICNTAll	PAC_IO.ClearDICNTAll	清除 DI 模組所有的 DI 通道的計數器值。
pac_WriteAO/pac_WriteAO_MF	PAC_IO.WriteAO/PAC_IO.WriteAO_MF	寫入 AO 值至 AO 模組。

PACSDK 函數	PACNET 函數	說明
pac_ReadAO	PAC_IO.ReadAO	讀取 AO 模組的 AO 值
pac_ReadAI	PAC_IO.ReadAI	讀取 AI 模組的 AI 值。
pac_ReadAIHex	PAC_IO.ReadAIHex	讀取 AI 模組的 16 進制 AI 值。
pac_ReadAIAllExt	PAC_IO.ReadAIAllExt	讀取 AI 模組所有通道的 AI 值
pac_ReadAIAll	PAC_IO.ReadAIAll	讀取 AI 模組所有通道的 AI 值。
pac_ReadAIAllHexExt	PAC_IO.ReadAIAllHexExt	讀取 AI 模組 16 進制所有通道的 AI 值
pac_ReadAIAllHex	PAC_IO.ReadAIAllHex	讀取 AI 模組 16 進制的所有通道的 AI 值。
pac_ReadCNT	PAC_IO.ReadCNT	讀取計數器/頻率模組的計數器值。
pac_ClearCNT	PAC_IO.ClearCNT	清除計數器/頻率模組的計數器值。
pac_ReadCNTOverflow	PAC_IO.ReadCNTOverflow	清除計數器/頻率模組的計數器溢出值。
pac_WriteModuleSafeValueDO pac_WriteModuleSafeValueDO_MF	PAC_IO.WriteModuleSafeValueDO PAC_IO.WriteModuleSafeValueDO_MF	寫入 DO 安全輸出值到 DO 模組。
pac_ReadModuleSafeValueDO pac_ReadModuleSafeValueDO_MF	PAC_IO.ReadModuleSafeValueDO PAC_IO.ReadModuleSafeValueDO_MF	讀取 DO 模組的安全輸出值。
pac_WriteModulePowerOnValueDO pac_WriteModulePowerOnValueDO_MF	PAC_IO.WriteModulePowerOnValueDO PAC_IO.WriteModulePowerOnValueDO_MF	寫入 DO 模組的開機 DO 預設輸出的設定值，到 DO 模組。
pac_ReadModulePowerOnValueDO pac_ReadModulePowerOnValueDO_MF	PAC_IO.ReadModulePowerOnValueDO PAC_IO.ReadModulePowerOnValueDO_MF	讀取的 DO 模組開機 DO 預設輸出的設定值。
pac_WriteModuleSafeValueAO	PAC_IO.WriteModuleSafeValueAO	寫入 AO 模組的 AO 安全輸出值。

PACSDK 函數	PACNET 函數	說明
pac_ReadModuleSafeValueAO	PAC_IO.ReadModuleSafeValueAO	讀取 AO 模組的 AO 安全輸出值。
pac_WriteModulePowerOnValueAO	PAC_IO.WriteModulePowerOnValueAO	寫入 AO 模組的 AO 啟動預設輸出的設定值。
pac_ReadModulePowerOnValueAO	PAC_IO.ReadModulePowerOnValueAO	讀取 AO 模組的 AO 啟動預設輸出的設定值。
pac_GetModuleLastOutputSource	PAC_IO.GetModuleLastOutputSource	讀取模組的最後一個輸出源。
pac_GetModuleWDTStatus	PAC_IO.GetModuleWDTStatus	讀取模組看門狗狀態。
pac_GetModuleWDTConfig	PAC_IO.GetModuleWDTConfig	讀取模組看門狗設定值。
pac_SetModuleWDTConfig	PAC_IO.SetModuleWDTConfig	設置模組看門狗設定。
pac_ResetModuleWDT	PAC_IO.ResetModuleWDT	重置模組的看門狗超時後的輸出狀態，讓模組能重新進入給使用者控制輸出的狀態。
pac_RefreshModuleWDT	PAC_IO.RefreshModuleWDT	重置模組的看門狗的超時計數
pac_InitModuleWDTInterrupt	PAC_IO.InitModuleWDTInterrupt	初始化插槽模組的看門狗發出中斷訊號的功能。
pac_GetModuleWDTInterruptStatus	PAC_IO.GetModuleWDTInterruptStatus	讀取插槽模組看門狗中斷功能狀態。
pac_SetModuleWDTInterruptStatus	PAC_IO.SetModuleWDTInterruptStatus	設定插槽模組的看門狗中斷功能狀態。

3.7.1. pac_GetBit

該函數檢索參數值的特定位元值，常利用於 DI/DO 模組讀回狀態值時，分拆指定通道的狀態。

語法

C++

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

參數

v

16 進制的 I/O 結果值。

NDX

特定的檢索位元。

回傳值

具體的指標值。

範例

[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);
```

備註

該函數所使用即為運算式 v＆(1 << index)。

3.7.2. pac_WriteDO/pac_WriteDO_MF

這個函數寫入 DO 值，至 DO 模組。

語法

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
);
```

參數

hPort

[in] 詢問 87K 模組時，需帶入 uart_Open 串口句柄

詢問 8K 模組時，帶入 0。

iSlot

[in] 接收命令的槽位

如果是遠端的 I/O 模組，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iDO_TotalCh

[in] DO 模組的 DO 通道的總通道數。

iDO_Value

[in] 一個 8 位十六進制值，第 0 位映射 DO0, 31 位對應於 DO31 等。當該位為 1 時，表示該輸出通道為 on，而 0 表示該輸出通道 off。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

備註

該的 pac_WriteDO 和 pac_WriteDO_MF 函數其參數和回傳值定義是相同的。不同的是，pac_WriteDO 是適用於詢問純 DIO DCON 模組和 pac_WriteDO_MF 是適用於詢問多功能 DCON 模組。

範例

[C] pac_WriteDO

實施例 1：

```
HANDLE hPort; //遠端模組
hPort = uart_Open( "COM2, 9600, N, 8, 1" );
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //通道 2 輸出
BOOL RET = pac_WriteDO(hPort, PAC_REMOTE_IO(1), TOTAL_CHANNEL, do_value);
uart_Close(hPort);
```

實施例 2：

```
//本地 87K 模組
HANDLE hPort;
hPort = uart_Open("");
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //通道 2 輸出
BOOL RET = pac_WriteDO(hPort, 1, TOTAL_CHANNEL, do_value);
uart_Close(hPort);
```

實施例 3：

```
//本地 8K 模組
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //通道 2 輸出
BOOL RET = pac_WriteDO(0, 1, TOTAL_CHANNEL, do_value);
```

[C] pac_WriteDO_MF

實施例 1：

```
//遠端模組
HANDLE hPort;
hPort = uart_Open( “COM2, 9600, N, 8, 1” );
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //通道 2 輸出
BOOL RET = pac_WriteDO_MF(hPort, PAC_REMOTE_IO(1), TOTAL_CHANNEL, do_value);
uart_Close(hPort);
```

實施例 2：

```
//本地 87K 模組
HANDLE hPort;
hPort = uart_Open(“”);
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //通道 2 輸出
BOOL RET = pac_WriteDO_MF(hPort, 1, TOTAL_CHANNEL, do_value);
uart_Close(hPort);
```

[C#] pac_WriteDO

實施例 1：

```
//遠端模組
IntPtr hPort;
hPort = PACNET.UART.Open( “COM1, 9600, N, 8, 1” );
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //通道 2 輸出
Bool RET = PACNET.PAC_IO.WriteDO(hPort, PACNET.PAC_IO.REMOTE_IO(1),
TOTAL_CHANNEL, do_value);
PACNET.UART.Close(hPort);
```

實施例 2：

```
//本地 87K 模組
IntPtr hPort; hPort = PACNET.UART.Open(“”);
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //通道 2 輸出
boolRET = PACNET.PAC_IO.WriteDO(hPort, 1, TOTAL_CHANNEL, do_value);
PACNET.UART.Close(hPort);
```

實施例 3：

```
//本地 8K 模組
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //通道 2 輸出
bool 型 RET = PACNET.PAC_IO.WriteDO(0, 1, TOTAL_CHANNEL, do_value);
```

[C#] pac_WriteDO_MF

實施例 1：

```
//遠端模組
IntPtr hPort; hPort = PACNET.UART.Open( "COM1, 9600, N, 8, 1" );
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //通道 2 輸出
boolRET = PACNET.PAC_IO.WriteDO_MF(hPort, PACNET.PAC_IO.REMOTE_IO(1),
TOTAL_CHANNEL, do_value);
PACNET.UART.Close(hPort);
```

實施例 2：

```
//本地 87K 模組
IntPtr hPort; hPort = PACNET.UART.Open("");
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //通道 2 輸出
boolRET = PACNET.PAC_IO.WriteDO_MF(hPort, 1, TOTAL_CHANNEL, do_value);
PACNET.UART.Close(hPort);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果為遠程模組，第二個參數需要使用，PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。

3.7.3. pac_WriteDOBit

這個函數寫入 DO 的值的一個位元至 DO 模組,只有對應於 DO 值的該位元 DO 通道被改變。

語法

C++

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

參數

hPort

[in] 87K 模組, 帶入 uart_Open 串口句柄

8K 模組, 帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的, 請使用 PAC_REMOTE_IO(0 ... 255)。

iChannel

[in]] 要變化 DO 通道。

iDO_TotalCh

[in] 在 DO 模組的 DO 通道的總數。

iBitValue

[in] 1 打開的 DO 通道; 0 關閉。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

實施例 1：

```
HANDLE hPort; //本地 87K 模組
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);
```

實施例 2：

```
BYTE iSlot = 1; //本地 8K 模組
int iChannel = 2;
int iDO_TotalCh = 8;
int iBitValue = 1;
BOOL RET = pac_WriteDOBit(0, iSlot, iChannel, miDO_TotalCh, iBitValue);
```

[C#]

```
IntPtrhPort; //本地 87K 模組
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);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的, 第二個參數的範圍是從 0 到 7。如果為遠程模組, 第二個參數需要使用, PAC_REMOTE_IO(0 ... 255), 其範圍是從 0 到 255。

3.7.4. pac_ReadDO/pac_ReadDO_MF

該函數讀取 DO 模組的 DO 輸出值。

語法

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
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

8K 模組，帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iDO_TotalCh

[in] 在 DO 模組的 DO 通道的總數。

lDO_Value

[in] 從 DO 模組讀取的 DO 輸出值的指標。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C] pac_ReadDO

實施例 1：

```
//本地 87K 模組
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);
```

實施例 2：

```
//本地 8K 模組
byte slot= 1;
int TOTAL_CHANNEL = 8;
DWORD do_value;
BOOL RET = pac_ReadDO(0, slot, TOTAL_CHANNEL, &do_value);
```

[C] pac_ReadDO_MF

實施例 1：

```
//本地 87K 模組
HANDLE hPort;
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

```
//本地 87K 模組
IntPtr hPort;
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

```
//本地 87K 模組
IntPtr hPort;
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);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的, 第二個參數的範圍是從 0 到 7。如果為遠程模組, 第二個參數需要使用, PAC_REMOTE_IO(0 ... 255), 其範圍是從 0 到 255。

3.7.5. pac_ReadDI/pac_ReadDI_MF

該函數讀取 DI 模組的 DI 值。

語法

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
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

8K 模組，帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iDI_TotalCh

[in] DI 模組的總通道數。

lDI_Value

[out] DI 值讀回的指標。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C] pac_ReadDI

實施例 1：

```
//本地 87K 模組
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);
```

實施例 2：

```
//本地 8K 模組
BYTE iSlot = 2;
int iDI_TotalCh = 8;
DWORD lDI_Value;
BOOL IRET = pac_ReadDI(0, iSlot, iDI_TotalCh,  &lDI_Value);
```

[C] pac_ReadDI_MF

```
//本地 87K 模組
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

```
//本地 87K 模組
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

```
//本地 87K 模組
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);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的, 第二個參數的範圍是從 0 到 7。如果是遠端模組, 第二個參數需要使用 PAC_REMOTE_IO(0 ... 255), 其範圍是從 0 到 255。

3.7.6. pac_ReadDIO/pac_ReadDIO_MF

該函數讀取 DIO 模組的 DI 和 DO 值。

語法

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
);
```

參數

hPort

[in] 87K 模組, 帶入 uart_Open 串口句柄

8K 模組, 帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的, 請使用 PAC_REMOTE_IO(0 ... 255)函數。

iDI_TotalCh

[in] 的 DIO 模組的 DI 通道的總數。

iDO_TotalCh

[in]] 的 DIO 模組的 DO 通道的總數。

lDI_Value

[out] DI 讀回值的指標。

lDO_Value

[out] DO 讀回值的指標。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C] pac_ReadDIO

實施例 1：

```
//本地 87K 模組
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);
```

實施例 2：

```
//本地 8K 模組
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

```
//本地 87K 模組
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_MF(hPort, iSlot, iDI_TotalCh, iDO_TotalCh, &lDI_Value,
&lDO_Value);
uart_Close(hPort);
```

[C#] pac_ReadDIO

```
//本地 87K 模組
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

```
//本地 87K 模組
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);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。

3.7.7. pac_ReadDILatch

該函數讀取 DI 模組的 DI 鎖存值 (Latch)。

語法

C++

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

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

8K 模組，帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iDI_TotalCh

[in]] DI 模組的 DI 通道的總數。

iLatchType

[in] 指定要讀鎖存器的值後面的鎖存型。

1 =鎖定高狀態

0 =鎖定低狀態

lDI_Latch_Value

[out] 指標變數儲由 DI 模組讀回的鎖存值。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

實施例 1：

```
HANDLE hPort; //本地 87K 模組
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);
```

實施例 2：

```
BYTE iSlot = 1; //本地 8K 模組
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; //本地 87K 模組
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);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的, 第二個參數的範圍是從 0 到 7。如果是遠端模組, 第二個參數需要使用 PAC_REMOTE_IO(0 ... 255), 其範圍是從 0 到 255。

3.7.8. pac_ClearDILatch

該函數清除 DI 模組的鎖存值(Latch)。

語法

C++

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

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

8K 模組，帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

實施例 1：

```
//本地 87K 模組
HANDLE hPort;
hPort = uart_Open(“”);
BYTE iSlot = 1;
BOOL IRET = pac_ClearDILatch(hPort, iSlot);
uart_Close(hPort);
```

實施例 2：

```
//本地 8K 模組
BYTE iSlot = 1;
BOOL IRET = pac_ClearDILatch(0, iSlot);
```

[C#]

```
//本地 87K 模組
IntPtr hPort;
hPort = PACNET.UART.Open(“”);
byte iSlot = 1;
boolIRET = PACNET.PAC_IO.ClearDILatch(hPort, iSlot);
PACNET.UART.Close(hPort);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。

3.7.9. pac_ReadDIOLatch

該函數讀取 DIO 模組的 DI 及 DO 通道鎖存值。

語法

C++

```
BOOL pac_ReadDIOLatch(
    HANDLE hPort,
    int slot,
    int iDI_TotalCh,
    int iDO_TotalCh,
    int iLatchType,
    DWORD * lDI_Latch_Value,
    DWORD * lDO_Latch_Value
);
```

參數

hPort

[in] 87K 模組, 帶入 uart_Open 串口句柄

8K 模組, 帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的, 請使用 PAC_REMOTE_IO(0 ... 255)函數。

iDI_TotalCh

[in]] DIO 模組的 DI 通道的總數。

iDO_TotalCh

[in] DIO 模組的 DO 通道的總數量。

iLatchType

[in] 讀回鎖存值的類型。

1 =鎖定高狀態

0 =鎖定低狀態

lDI_Latch_Value

[out] 指標讀回 DI 鎖存值。

lDO_Latch_Value

[out] 指標讀回 DO 鎖存值。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

實施例 1：

```
//本地 87K 模組
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);
```

實施例 2：

```
//本地 8K 模組
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(0, iSlot, iDI_TotalCh, iDO_TotalCh, iLatchType, &
lDI_Latch_Value, &lDO_Latch_Value);
```

[C#]

```
//本地 87K 模組
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;
boolIRET = PACNET.PAC_IO.ReadDIOLatch(hPort, iSlot, iDI_TotalCh, iDO_TotalCh, iLatchType,
ref lDI_Latch_Value, ref lDO_Latch_Value);
PACNET.UART.Close(hPort);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。

3.7.10. pac_ClearDIOLatch

此功能會清除 DIO 模組的 DI 和 DO 通道鎖存值。

語法

C++

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

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

8K 模組，帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

實施例 1：

```
//本地 87K 模組
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
BOOL IRET = pac_ClearDIOLatch(hPort, iSlot);
uart_Close(hPort);
```

實施例 2：

```
//本地 8K 模組
BYTE iSlot = 1;
BOOL IRET = pac_ClearDIOLatch(0, iSlot);
```

[C#]

```
//本地 87K 模組
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
boolIRET = PACNET.PAC_IO.ClearDIOLatch(hPort, iSlot);
PACNET.UART.Close(hPort);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。

3.7.11. pac_ReadDICNT/pac_ReadDICNT_MF

該函數讀取的 DI 模組的 DI 通道計數器值。

語法

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
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

8K 模組，帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iChannel

[in] 該計數器的值所屬的通道。

iDI_TotalCh

[in]] DI 模組的 DI 通道總數。

lCounter_Value

[out] 指標變數讀回計數器值。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C] pac_ReadDICNT

實施例 1：

```
//本地 87K 模組
HANDLE hPort;
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);
```

實施例 2：

```
//本地 8K 模組
BYTE iSlot = 1;
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

```
//本地 87K 模組
HANDLE hPort;
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

```
//本地 87K 模組
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

```
//本地 87K 模組
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);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。

3.7.12. pac_ClearDICNT/pac_ClearDICNT_MF

這個函數將清除 DI 模組的 DI 通道計數器值。

語法

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
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

8K 模組，帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iChannel

[in] 該計數器的值所屬的通道。

iDI_TotalCh

[in]] DI 模組的 DI 通道總數。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C] pac_ClearDICNT

實施例 1：

```
//本地 87K 模組
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);
```

實施例 2：

```
//本地 8K 模組
BYTE iSlot = 1;
int iChannel = 2;
int iDI_TotalCh = 8;
BOOL IRET = pac_ClearDICNT(0, iSlot, iChannel, iDI_TotalCh);
```

[C] pac_ClearDICNT_MF

```
//本地 87K 模組
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

```
//本地 87K 模組
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

```
//本地 87K 模組
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);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的, 第二個參數的範圍是從 0 到 7。如果是遠端模組, 第二個參數需要使用 PAC_REMOTE_IO(0 ... 255), 其範圍是從 0 到 255。

3.7.13. pac_ReadDICNTAll

該函數讀取 DI 模組所有的 DI 通道計數器值。

語法

C++ - pac_ReadDICNTAll

```
BOOL pac_ReadDICNTAll(
    HANDLE hPort,
    int slot,
    int iDI_TotalCh,
    DWORD    lCounter_Value[]
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

8K 模組，帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iDI_TotalCh

[in]] DI 模組的 DI 通道總數。

lCounter_Value[]

[out] 指標變數讀回計數器值陣列。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C] pac_ReadDICNTAll

實施例 1：

```
//本地 87K 模組
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

```
//本地 87K 模組
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);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。

3.7.14. pac_ClearDICNTAll

這個函數將清除 DI 模組所有的 DI 通道計數器值。

語法

C++ -pac_ClearDICNTAll

```
BOOL pac_ClearDICNT(
    HANDLE hPort,
    int slot,
    int iDI_TotalCh
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

8K 模組，帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iDI_TotalCh

[in]] DI 模組的 DI 通道總數。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C] pac_ClearDICNTAll

實施例 1：

```
//本地 87K 模組
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

```
//本地 87K 模組
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iDI_TotalCh = 8;
boolIRET = PACNET.PAC_IO.ClearDICNTAll(hPort, iSlot, iDI_TotalCh);
PACNET.UART.Close(hPort);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。

3.7.15. pac_WriteAO/pac_WriteAO_MF

該函數寫入 AO 值至 AO 模組。

語法

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
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

8K 模組，帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iChannel

[in] 一個指定的 AO 值的通道。

iAO_TotalCh

[in]] AO 模組的 AO 通道總數。

float fValue

[in]] AO 值寫入 AO 模組。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C] pac_WriteAO

實施例 1：

```
//本地 87K 模組
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);
```

實施例 2：

```
//本地 8K 模組
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

```
//本地 87K 模組
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);
```

備註

1. 該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。
2. 以下為 pac_WriteAO / pac_WriteAO _MF 功能和可用的模組比照表：

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

該函數讀取 AO 模組的 AO 值。

語法

C++

```
BOOL pac_ReadAO(
    HANDLE hPort,
    int slot,
    int iChannel,
    int iAO_TotalCh,
    float* fValue
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

8K 模組，帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iChannel

[in] 指定讀取的 AO 值通道。

iAO_TotalCh

[in]] AO 模組的 AO 通道總數。

float fValue

[in]] 指標是從 AO 模組讀回的 AO 值。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

實施例 1：

```
HANDLE hPort; //本地 87K 模組
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);
```

實施例 2：

```
BYTE iSlot = 1; //本地 8K 模組
int iChannel = 2;
int iAO_TotalCh = 8;
float fValue;
BOOL IRET = pac_ReadAO(0, iSlot, iChannel, iAO_TotalCh,  &fValue);
```

[C#]

```
IntPtr hPort; //本地 87K 模組
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);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的, 第二個參數的範圍是從 0 到 7。如果是遠端模組, 第二個參數需要使用 PAC_REMOTE_IO(0 ... 255), 其範圍是從 0 到 255。

3.7.17. pac_ReadAI

該函數讀取 AI 模組的 AI 值。

語法

C++

```
BOOL pac_ReadAI(
    HANDLE hPort,
    int slot,
    int iChannel,
    int iAI_TotalCh,
    float* fValue
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄 。

8K 模組，帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iChannel

[in] 指定讀取的 AI 值通道。

iAI_TotalCh

[in] AI 模組的 AI 通道總數

fValue

[in] 指標是從 AI 模組讀回的 AI 值。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

實施例 1：

```
HANDLE hPort; //本地 87K 模組
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);
```

實施例 2：

```
BYTE iSlot = 1; //本地 8K 模組
int iChannel = 2;
int iAI_TotalCh = 8;
float fValue;
BOOL IRET = pac_ReadAI(0, iSlot, iChannel, iAI_TotalCh, &fValue);
```

[C#]

```
IntPtr hPort; //本地 87K 模組
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);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的, 第二個參數的範圍是從 0 到 7。如果是遠端模組, 第二個參數需要使用 PAC_REMOTE_IO(0 ... 255), 其範圍是從 0 到 255。

3.7.18. pac_ReadAIHex

該函數讀取 AI 模組的 16 進制 AI 值。

語法

C++

```
BOOL pac_ReadAIHex(
    HANDLE hPort,
    int slot,
    int iChannel,
    int iAI_TotalCh,
    int * iValue
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄　　在本地。

8K 模組，帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iChannel

[in] 指定讀取的 AI 值通道。

iAI_TotalCh

[in]] AI 模組的 AI 通道總數

iValue

[in]] 指標指向從 AI 模組讀回的 AI 值。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

實施例 1：

```
HANDLE hPort; //本地 87K 模組
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);
```

實施例 2：

```
BYTE iSlot = 1; //本地 8K 模組
int iChannel = 2;
int iAI_TotalCh = 8;
INT iValue;
BOOL IRET = pac_ReadAIHex(0, iSlot, iChannel, iAI_TotalCh, &iValue);
```

[C#]

```
IntPtr hPort; //本地 87K 模組
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);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的, 第二個參數的範圍是從 0 到 7。如果是遠端模組, 第二個參數需要使用 PAC_REMOTE_IO(0 ... 255), 其範圍是從 0 到 255。

3.7.19. pac_ReadAIAllExt

該函數讀取 AI 模組的所有通道 AI 值。

此函數替換 pac_ReadAIAll。

語法

C++

```
BOOL pac_ReadAIAllExt(
    HANDLE hPort,
    int slot,
    float fValue[] ,
    DWORD Buff_Len,
    DWORD *Channel
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

8K 模組，帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

fValue[]

[out] 陣列包含讀取 AI 模組的 AI 值。

Buff_Len

[in] 用於指定的 fValueS 緩衝區的大小。

channel

[out] 用於讀取指定 AI 模組的總可用通道量的指標。如果回傳值是 TRUE，此通道量才有效。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

實施例 1：

```
//本地 87K 模組
HANDLE hPort;
int ichannelnumber = 0;
hPort = uart_Open(“”);
BYTE iSlot = 1;
float fValue[8] ;
BOOL IRET = pac_ReadAIAllExt(hPort, iSlot, fValue, 8,  &ichannelnumber);
uart_Close(hPort);
```

實施例 2：

```
//本地 8K 模組
BYTE iSlot = 1;
int ichannelnumber = 0;
float fValue[8] ;
BOOL IRET = pac_ ReadAIAllExt(0, iSlot, fValue, 8,  &ichannelnumber);
```

[C#]

```
//本地 87K 模組
IntPtr hPort;
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);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的, 第二個參數的範圍是從 0 到 7。如果是遠端模組, 第二個參數需要使用 PAC_REMOTE_IO(0 ... 255), 其範圍是從 0 到 255。

3.7.20. pac_ReadAIAll

該函數讀取 AI 模組的所有通道的所有 AI 值。

該函數可能會導致緩衝區溢出的一些情況。

語法

C++

```
BOOL pac_ReadAIAll(
    HANDLE hPort,
    int slot,
    float fValue[]
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

8K 模組，帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

fValue[]

[out] 陣列包含由 AI 模組讀回的 AI 值。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

實施例 1：

```
//本地 87K 模組
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
float fValue[8] ;
BOOL IRET = pac_ReadAIAll(hPort, iSlot, fValue);
uart_Close(hPort);
```

實施例 2：

```
//本地 8K 模組
BYTE iSlot = 1;
float fValue[8] ;
BOOL IRET = pac_ReadAIAll(0, iSlot, fValue);
```

[C#]

```
//本地 87K 模組
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
float fValue[8] ;
boolIRET = PACNET.PAC_IO.ReadAIAll(hPort, iSlot, fValue);
PACNET.UART.Close(hPort);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。

3.7.21. pac_ReadAIAllHexExt

該函數讀取 AI 模組的 16 進制所有通道 AI 值。

此函數替換 pac_ReadAIAllHex。

語法

C++

```
BOOL pac_ReadAIAllHexExt(
    HANDLE hPort,
    int slot,
    INT iValue [] ,
    DWORD Buff_Len,
    DWORD* Channel
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

8K 模組，帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iValue []

[out] 陣列包含由 AI 模組讀回的 AI 值。

Buff_Len

[in]，用於指定 iValue 緩衝區的大小。

channel

[out] 用於讀取指定 AI 模組的總可用通道量的指標。如果回傳值是 TRUE，此通道量是有效的

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

實施例 1：

```
//本地 87K 模組
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);
```

實施例 2：

```
//本地 8K 模組
BYTE iSlot = 1;
int ichannelnumber = 0;
int iValue [8] ;
BOOL IRET = pac_ReadAIAllHexExt(0, iSlot, iValue, 8,  &ichannelnumber);
```

[C#]

```
//本地 87K 模組
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);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的, 第二個參數的範圍是從 0 到 7。如果是遠端模組, 第二個參數需要使用 PAC_REMOTE_IO(0 ... 255), 其範圍是從 0 到 255。

3.7.22. pac_ReadAIAllHex

該函數讀取 AI 模組的 16 進制的所有通道 AI 值。

該函數可能會導致緩衝區溢出的一些情況。

語法

C++

```
BOOL pac_ReadAIAllHex(
    HANDLE hPort,
    int slot,
    int iValue []
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

8K 模組，帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iValue []

[out] 陣列包含讀取 AI 模組的 AI 值。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

實施例 1：

```
//本地 87K 模組
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iValue [8] ;
BOOL IRET = pac_ReadAIAllHex(hPort, iSlot, iValue);
uart_Close(hPort);
```

實施例 2：

```
//本地 8K 模組
BYTE iSlot = 1;
int iValue [8] ;
BOOL IRET = pac_ReadAIAllHex(0, iSlot, iValue);
```

[C#]

```
//本地 87K 模組
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iValue [8] ;
boolIRET = PACNET.PAC_IO.ReadAIAllHex(hPort, iSlot, iValue);
PACNET.UART.Close(hPort);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。

3.7.23. pac_ReadCNT

該函數讀取計數器/頻率模組的計數器值。

語法

C++

```
BOOL pac_ReadCNT(
    HANDLE hPort,
    int slot,
    int iChannel,
    DWORD * lCounter_Value
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

8K 模組，帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iChannel

[in] 從指定的計數器/頻率模組的通道，讀取一個計數器的值。

lCounter_Value

[out] 用於讀出從計數器/頻率模組的計數器值的指標。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

實施例 1：

```
//本地 87K 模組
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);
```

實施例 2：

```
//本地 8K 模組
BYTE iSlot = 1;
int iChannel = 0;
DWORD lCounter_Value;
BOOL IRET = pac_ReadCNT(0, iSlot, iChannel,  &lCounter_Value);
```

[C#]

```
//本地 87K 模組
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);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的, 第二個參數的範圍是從 0 到 7。如果是遠端模組, 第二個參數需要使用 PAC_REMOTE_IO(0 ... 255), 其範圍是從 0 到 255。

3.7.24. pac_ClearCNT

此功能清除計數器/頻率模組的計數器值。

語法

C++

```
BOOL pac_ClearCNT(
    HANDLE hPort,
    int slot,
    int iChannel
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

8K 模組，帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iChannel

[in] 清除模組計數器值的通道 ，從計數器/頻率後面 。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

實施例 1：

```
//本地 87K 模組
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
int iChannel = 0;
BOOL IRET = pac_ClearCNT(hPort, iSlot, iChannel);
uart_Close(hPort);
```

實施例 2：

```
//本地 8K 模組
BYTE iSlot = 1;
int iChannel = 0;
BOOL IRET = pac_ClearCNT(0, iSlot, iChannel);
```

[C#]

```
//本地 87K 模組
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iChannel = 0;
boolIRET = PACNET.PAC_IO.ClearCNT(hPort, iSlot, iChannel);
PACNET.UART.Close(hPort);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。

3.7.25. pac_ReadCNTOverflow

此功能清除計數器/頻率模組的計數器溢出值。

語法

C++

```
BOOL pac_ReadCNTOverflow(
    HANDLE hPort,
    int slot,
    int iChannel,
    int * iOverflow
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

8K 模組，帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iChannel

[in] 計數器/頻率模組從後面的通道，讀取計數器溢出值。

iOverflow

[out] 讀取計數器/頻率模組的計數器的溢出值指標。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[eVC/VC/VS]

實施例 1：

```
//本地 87K 模組
HANDLE hPort;
hPort = uart_Open(“”);
BYTE iSlot = 1;
int iChannel = 0;
int iOverflow;
BOOL IRET = pac_ReadCNTOverflow(hPort, iSlot, iChannel, &iOverflow);
uart_Close(hPort);
```

實施例 2：

```
//本地 8K 模組
BYTE iSlot = 1;
int iChannel = 0;
int iOverflow;
BOOL IRET = pac_ReadCNTOverflow(0, iSlot, iChannel, &iOverflow);
```

[C#]

```
//本地 87K 模組
IntPtr hPort;
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);
```

備註

該功能可用支援本地或遠端模組。當模組是本地的, 第二個參數的範圍是從 0 到 7。如果是遠端模組, 第二個參數需要使用 PAC_REMOTE_IO(0 ... 255), 其範圍是從 0 到 255。

3.7.26. pac_WriteModuleSafeValueDO pac_WriteModuleSafeValueDO_MF

這個函數寫入 DO 安全輸出值到 DO 模組。

語法

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
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

8K 模組，帶入 0

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iDO_TotalCh

[in] DO 模組的 DO 通道的總數。

iValue

[in] 一個 8 位十六進制值，第 0 位映射 DO0, 31 位對應於 DO31 等。當該位為 1 時，表示該數字通道輸出，而 0 表示該數字通道沒有輸出。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C] pac_WriteModuleSafeValueDO

實施例 1：

```
//如果遠端模組
HANDLE hPort;
hPort = uart_Open( "COM2, 9600, N, 8, 1" );
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //通道 2 輸出
BOOL RET = pac_WriteModuleSafeValueDO(hPort, PAC_REMOTE_IO(1), TOTAL_CHANNEL,
do_value); uart_Close(hPort);
```

實施例 2：

```
//本地 87K 模組
HANDLE hPort;
hPort = uart_Open("");
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //通道 2 輸出
BOOL RET = pac_WriteModuleSafeValueDO(hPort, 1, TOTAL_CHANNEL, do_value);
uart_Close(hPort);
```

[C] pac_WriteModuleSafeValueDO_MF

實施例 1：

```
//如果遠端模組
HANDLE hPort;
hPort = uart_Open( “COM2, 9600, N, 8, 1” );
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //通道 2 輸出
BOOL RET = pac_WriteModuleSafeValueDO_MF(hPort, PAC_REMOTE_IO(1),
TOTAL_CHANNEL, do_value); uart_Close(hPort);
```

實施例 2：

```
//本地 87K 模組
HANDLE hPort;
hPort = uart_Open(“”);
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //通道 2 輸出
BOOL RET = pac_WriteModuleSafeValueDO_MF(hPort, 1, TOTAL_CHANNEL, do_value);
uart_Close(hPort);
```

[C#] pac_WriteModuleSafeValueDO

實施例 1：

```
//如果遠端模組
IntPtr hPort; hPort = PACNET.UART.Open( “COM1, 9600, N, 8, 1” );
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //通道 2 輸出
boolRET = PACNET.PAC_IO.pac_WriteModuleSafeValueDO(hPort,
PACNET.PAC_IO.REMOTE_IO(1), TOTAL_CHANNEL, do_value);
PACNET.UART.Close(hPort);
```

實施例 2：

```
//本地 87K 模組
IntPtr hPort; hPort = PACNET.UART.Open("");
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //通道 2 輸出
boolRET = PACNET.PAC_IO.pac_WriteModuleSafeValueDO(hPort, 1, TOTAL_CHANNEL,
do_value);
PACNET.UART.Close(hPort);
```

[C#] pac_WriteModuleSafeValueDO_MF

實施例 1：

```
//如果遠端模組
IntPtr hPort; hPort = PACNET.UART.Open( "COM1, 9600, N, 8, 1" );
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //通道 2 輸出
boolRET = PACNET.PAC_IO.pac_WriteModuleSafeValueDO_MF(hPort,
PACNET.PAC_IO.REMOTE_IO(1), TOTAL_CHANNEL, do_value);
PACNET.UART.Close(hPort);
```

實施例 2：

```
//本地 87K 模組
IntPtr hPort; hPort = PACNET.UART.Open("");
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //通道 2 輸出
boolRET = PACNET.PAC_IO.pac_WriteModuleSafeValueDO_MF(hPort, 1, TOTAL_CHANNEL,
do_value);
PACNET.UART.Close(hPort);
```

備註

1. 該功能可用支援本地或遠端模組。當模組是本地的, 第二個參數的範圍是從 0 到 7。如果是遠端模組, 第二個參數需要使用 PAC_REMOTE_IO(0 ... 255), 其範圍是從 0 到 255。
2. 具備啟動預設輸出值或安全輸出值函數 I-7K/I-87K 系列模組, 可以支持這個 API 函數。I-8K 系列模組僅 I-8041RW 提供此功能, I-9K 系列 DO 模組都支援此功能。

3.7.27. pac_ReadModuleSafeValueDO pac_ReadModuleSafeValueDO_MF

該函數讀取 DO 模組的安全輸出值。

語法

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
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iChannel

[in] 在 DO 模組的 DO 通道的總數。

lValue

[in]] 從 DO 模組讀取 DO 安全輸出值的指標。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C] pac_ReadModuleSafeValueDO

```
//本地 87K 模組
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

```
//本地 87K 模組
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

```
//本地 87K 模組
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

```
//本地 87K 模組
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);
```

備註

1. 該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。
2.具備啟動預設輸出值或安全輸出值函數 I-7K/I-87K 系列模組，可以支持這個 API 函數。I-8K 系列模組僅 I-8041RW 提供此功能， I-9K 系列 DO 模組都支援此功能。

3.7.28. pac_WriteModulePowerOnValueDO pac_WriteModulePowerOnValueDO_MF

寫入 DO 模組的開機 DO 預設輸出設定值，到 DO 模組。

語法

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
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iDO_TotalCh

[in] 在 DO 模組的 DO 通道的總數。

iValue

[in] 一個 8 位十六進制值，第 0 位映射 DO0, 31 位對應於 DO31 等。當該位為 1 時，表示該數字輸出通道輸出，而 0 表示該數字輸出通沒輸出。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C] pac_WriteModulePowerOnValueDO

實施例 1：

```
HANDLE hPort; //如果遠端模組
hPort = uart_Open( “COM1, 9600, N, 8, 1” );
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //通道 2 輸出
BOOL RET = pac_WriteModulePowerOnValueDO(hPort, PAC_REMOTE_IO(1),
TOTAL_CHANNEL, do_value); uart_Close(hPort);
```

實施例 2：

```
HANDLE hPort; //本地 87K 模組
hPort = uart_Open(“”);
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //通道 2 輸出
BOOL RET = pac_WriteModulePowerOnValueDO(hPort, 1, TOTAL_CHANNEL, do_value);
uart_Close(hPort);
```

[C] pac_WriteModulePowerOnValueDO_MF

實施例 1：

```
HANDLE hPort; //如果遠端模組
hPort = uart_Open( “COM1, 9600, N, 8, 1” );
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //通道 2 輸出
BOOL RET = pac_WriteModulePowerOnValueDO_MF(hPort, PAC_REMOTE_IO(1),
TOTAL_CHANNEL, do_value); uart_Close(hPort);
```

實施例 2：

```
HANDLE hPort; //本地 87K 模組
hPort = uart_Open(“”);
int TOTAL_CHANNEL = 8;
DWORD do_value = 4; //通道 2 輸出
BOOL RET = pac_WriteModulePowerOnValueDO(hPort, 1, TOTAL_CHANNEL, do_value);
uart_Close(hPort);
```

[C#] pac_WriteModulePowerOnValueDO

實施例 1：

```
IntPtr 的 hPort = PACNET.UART.Open( “COM1, 9600, N, 8, 1” ); //如果遠端模組
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //通道 2 輸出
boolRET = PACNET.PAC_IO.pac_WriteModulePowerOnValueDO(hPort,
PACNET.PAC_IO.REMOTE_IO(1), TOTAL_CHANNEL, do_value);
PACNET.UART.Close(hPort);
```

實施例 2：

```
IntPtr 的 hPort = PACNET.UART.Open( "" ); //本地 87K 模組
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //通道 2 輸出
boolRET = PACNET.PAC_IO.pac_WriteModulePowerOnValueDO(hPort, 1, TOTAL_CHANNEL,
do_value);
PACNET.UART.Close(hPort);
```

[C#] pac_WriteModulePowerOnValueDO_MF

實施例 1：

```
//如果遠端模組
IntPtr 的 hPort = PACNET.UART.Open( "COM1, 9600, N, 8, 1" );
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //通道 2 輸出
boolRET = PACNET.PAC_IO.pac_WriteModulePowerOnValueDO_MF(hPort,
PACNET.PAC_IO.REMOTE_IO(1), TOTAL_CHANNEL, do_value);
PACNET.UART.Close(hPort);
```

實施例 2：

```
//本地 87K 模組
IntPtr 的 hPort = PACNET.UART.Open( "" );
int TOTAL_CHANNEL = 8;
UINT do_value = 4; //通道 2 輸出
boolRET = PACNET.PAC_IO.pac_WriteModulePowerOnValueDO_MF(hPort, 1,
TOTAL_CHANNEL, do_value);
PACNET.UART.Close(hPort);
```

備註

1. 該功能可用支援本地或遠端模組。當模組是本地的, 第二個參數的範圍是從 0 到 7。如果是遠端模組, 第二個參數需要使用 PAC_REMOTE_IO(0 ... 255), 其範圍是從 0 到 255。
2. 具備啟動預設輸出值或安全輸出值函數 I-7K/I-87K 系列模組可以支持這個 API 函數。I-8K 系列模組僅 I-8041RW 提供此功能, I-9K 系列 DO 模組都支援此功能。

3.7.29. pac_ReadModulePowerOnValueDO pac_ReadModulePowerOnValueDO_MF

該函數讀取 DO 模組的開機 DO 預設輸出設定值。

語法

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
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iChannel

[in] 在 DO 模組的 DO 通道的總數。

lValue

[in] DO 預設輸出設定值的指標,從 DO 模組讀取。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C] pac_ReadModulePowerOnValueDO

```
//本地 87K 模組
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#]

```
//本地 87K 模組
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

```
//本地 87K 模組
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

```
//本地 87K 模組
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);
```

備註

1. 該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。
2. 具備啟動預設輸出值或安全輸出值函數 I-7K/I-87K 系列模組可以支持這個 API 函數。I-8K 系列模組僅 I-8041RW 提供此功能， I-9K 系列 DO 模組都支援此功能。

3.7.30. pac_WriteModuleSafeValueAO

該函數寫入 AO 模組的 AO 安全輸出值。

語法

C++

```
BOOL pac_WriteModuleSafeValueAO(
    HANDLE hPort,
    int slot,
    int iChannel,
    int iAO_TotalCh,
    float fValue
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iChannel

[in] 要設定 AO 值的通道。

iAO_TotalCh

[in]] AO 模組的 AO 通道總數。

float fValue

[in]] 寫入 AO 模組的 AO 值。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
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#]

```
//本地 87K 模組
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);
```

備註

1. 該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。
2. 具備啟動預設輸出值或安全輸出值函數 I-7K/I-87K 系列模組可以支持這個 API 函數。I-8K 系列模組僅 I-8041RW 提供此功能， I-9K 系列 DO 模組都支援此功能。

3.7.31. pac_ReadModuleSafeValueAO

該函數讀取 AO 模組的 AO 安全輸出值。

語法

C++

```
BOOL pac_ReadModuleSafeValueAO(
    HANDLE hPort,
    int slot,
    int iChannel,
    int iAO_TotalCh,
    float* fValue
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iChannel

[in] 要讀取值得 AO 通道。

iAO_TotalCh

[in]] AO 模組的 AO 通道總數。

float fValue

[in]] 從 AO 模組讀回 AO 安全輸出值的指標。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
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#]

```
//本地 87K 模組
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);
```

備註

1. 該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。
2. 具備啟動預設輸出值或安全輸出值函數 I-7K/I-87K 系列模組可以支持這個 API 函數。I-8K 系列模組僅 I-8041RW 提供此功能， I-9K 系列 DO 模組都支援此功能。

3.7.32. pac_WriteModulePowerOnValueAO

該函數寫入 AO 模組的 AO 啟動預設輸出設定值。

語法

C++

```
BOOL pac_WriteModulePowerOnValueAO(
    HANDLE hPort,
    int slot,
    int iChannel,
    int iAO_TotalCh,
    float fValue
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iChannel

[in] 要設定的 AO 值的通道。

iAO_TotalCh

[in]] AO 模組的 AO 通道總數。

float fValue

[in]] 寫入 AO 模組的 AO 值。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
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#]

```
//本地 87K 模組
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);
```

備註

1. 該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。
2. 具備啟動預設輸出值或安全輸出值函數 I-7K/I-87K 系列模組可以支持這個 API 函數。I-8K 系列模組僅 I-8041RW 提供此功能， I-9K 系列 DO 模組都支援此功能。

3.7.33. pac_ReadModulePowerOnValueAO

此功能在讀取 AO 模組的 AO 啟動預設輸出設定值。

語法

C++

```
BOOL pac_ReadModulePowerOnValueAO(
    HANDLE hPort,
    int slot,
    int iChannel,
    int iAO_TotalCh,
    float* fValue
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

iSlot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

iChannel

[in] 要讀取的 AO 通道。

iAO_TotalCh

[in]] AO 模組的 AO 通道總數。

float fValue

[in] 讀取 AO 啟動預設輸出值的指標。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
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#]

```
//本地 87K 模組
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);
```

備註

1. 該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。
2. 具備啟動預設輸出值或安全輸出值函數 I-7K/I-87K 系列模組可以支持這個 API 函數。I-8K 系列產品沒有提供的功能模組。

3.7.34. pac_GetModuleLastOutputSource

該函數讀取模組的最後一個輸出。

語法

C++

```
short pac_GetModuleLastOutputSource(
    HANDLE hPort,
    int slot
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

回傳值

對於 I-8K(I-8kRw)

0：無動作

1：開機值

2：安全輸出值

3：一般 DO 命令

對於 i-87k/i-7k 模組

0：其他(開機值或一般 DO 命令)

2：安全輸出值

範例

[C]

```
//本地 87K 模組
HANDLE hPort;
int iSlot = 0;
int lastOutput = 0;
hPort = uart_Open("");
lastOutput = pac_GetModuleLastOutputSource(hPort, iSlot);
uart_Close(hPort);
```

[C#]

```
//本地 87K 模組
IntPtr hPort;
int iSlot = 0;
hPort = PACNET.UART.Open("");
int lastOutput = PACNET.PAC_IO.GetModuleLastOutputSource(hPort, iSlot);
PACNET.UART.Close(hPort);
```

備註

1. 該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。
2. 具備啟動預設輸出值或安全輸出值函數 I-7K/I-87K 系列模組可以支持這個 API 函數。I-8K 系列模組僅 I-8041RW 提供此功能， I-9K 系列 DO 模組都支援此功能。

3.7.35. pac_GetModuleWDTStatus

該函數讀取模組看門狗狀態。

語法

C++

```
bool pac_GetModuleWDTStatus(
    HANDLE hPort,
    int slot
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

回傳值

如果看們狗超時事件被觸發，則回傳 TRUE。

如果看們狗超時事件尚未被觸發，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
HANDLE hPort;
int iSlot = 0;
boolBSTATUS = 0;
hPort = uart_Open("");
BSTATUS = pac_GetModuleWDTStatus(hPort, iSlot);
uart_Close(hPort);
```

[C#]

```
//本地 87K 模組
IntPtr hPort;
int iSlot = 0;
hPort = PACNET.UART.Open("");
bool BSTATUS = PACNET.PAC_IO.GetModuleWDTStatus(hPort, iSlot);
PACNET.UART.Close(hPort);
```

備註

1. 該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。
2. 具備啟動預設輸出值或安全輸出值函數 I-7K/I-87K 系列模組可以支持這個 API 函數。I-8K 系列模組僅 I-8041RW 提供此功能， I-9K 系列 DO 模組都支援此功能。

3.7.36. pac_GetModuleWDTConfig

該函數讀取模組看門狗設定值。

語法

C++

```
bool pac_GetModuleWDTConfig(
    HANDLE hPort,
    int slot,
    short* enStatus,
    unsigned long* wdtTimeout,
    int * ifWDT_Overwrite
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

enStatus

[out] 1：看門狗激活。

0：看門狗禁用

wdtTimeout

[out] 回傳值，每一個單位是 100 毫秒。

ifWDT_Overwrite(僅適用於 I-8K)

[out] 1：看門狗輸出值可覆寫

0：看門狗輸出值不可覆寫

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
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#]

```
//本地 87K 模組
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);
```

備註

1.該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。

2.具備啟動預設輸出值或安全輸出值函數 I-7K/I-87K 系列模組可以支持這個 API 函數。I-8K 系列模組僅 I-8041RW 提供此功能， I-9K 系列 DO 模組都支援此功能。

3.7.37. pac_SetModuleWDTConfig

設置模組的看門狗功能。

語法

C++

```
bool pac_SetModuleWDTStatus(
    HANDLE hPort,
    int slot,
    short enStatus,
    unsigned longwdtTimeout,
    int ifWDT_Overwrite
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

enStatus

[in] 1：看門狗激活。

0：看門狗禁用

longwdtTimeout

[in]] 看門狗超時時間，每個單位是 100 毫秒。

ifWDT_Overwrite(僅適用於 I-8K)

[in] 1：看門狗輸出值可覆寫

0：看門狗輸出值不可覆寫

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
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#]

```
//本地 87K 模組
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);
```

備註

1.該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。

2.具備啟動預設輸出值或安全輸出值函數 I-7K/I-87K 系列模組可以支持這個 API 函數。I-8K 系列模組僅 I-8041RW 提供此功能， I-9K 系列 DO 模組都支援此功能。

3.7.38. pac_ResetModuleWDT

此功能重置模組的看門狗超時後的輸出狀態，讓模組能重新進入給使用者控制輸出的狀態。

語法

C++

```
bool pac_ResetModuleWDT(
    HANDLE hPort,
    int slot
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
HANDLE hPort;
int iSlot = 0;
hPort = uart_Open("");
pac_ResetModuleWDT(hPort, iSlot);
uart_Close(hPort);
```

[C#]

```
//本地 87K 模組
IntPtr hPort;
int iSlot = 0;
hPort = PACNET.UART.Open("");
PACNET.PAC_IO.ResetModuleWDT(hPort, iSlot);
PACNET.UART.Close(hPort);
```

備註

1.該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。

2.具備啟動預設輸出值或安全輸出值函數 I-7K/I-87K 系列模組可以支持這個 API 函數。I-8K 系列模組僅 I-8041RW 提供此功能， I-9K 系列 DO 模組都支援此功能。

3.7.39. pac_RefreshModuleWDT

這個函數將重置模組的看門狗超時計數。

語法

C++

```
bool pac_RefreshModuleWDT(
    HANDLE hPort,
    int slot
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
HANDLE hPort;
int iSlot = 0;
hPort = uart_Open("");
pac_RefreshModuleWDT(hPort, iSlot);
uart_Close(hPort);
```

[C#]

```
//本地 87K 模組
IntPtr hPort;
int iSlot = 0;
hPort = PACNET.UART.Open("");
PACNET.PAC_IO.RefreshModuleWDT(hPort, iSlot);
PACNET.UART.Close(hPort);
```

備註

1.該功能可用支援本地或遠端模組。當模組是本地的，第二個參數的範圍是從 0 到 7。如果是遠端模組，第二個參數需要使用 PAC_REMOTE_IO(0 ... 255)，其範圍是從 0 到 255。

2.具備啟動預設輸出值或安全輸出值函數 I-7K/I-87K 系列模組可以支持這個 API 函數。I-8K 系列模組僅 I-8041RW 提供此功能， I-9K 系列 DO 模組都支援此功能。

3.7.40. pac_InitModuleWDTInterrupt

這個函數初始化插槽模組的看門狗發出中斷訊號的功能。

語法

C++

```
bool pac_RefreshModuleWDT(
    int slot,
    PAC_CALLBACK_FUNC f
);
```

參數

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

f

[in] 一個 CALLBACK 函數。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
int CALLBACK slot_callback_proc()
{
//做一些事情
   return true;
}
int iSlot = 0;
pac_InitModuleWDTInterrupt(iSlot, slot_callback_proc);
```

[C#]

```
PACNET.CALLBACK_FUNC slot_callback_proc; //全局
int slot_callback_proc()
{
//做一些事情
   return 0;
}
int iSlot = 0;
PACNET.PAC_IO.InitModuleWDTInterrupt(iSlot, slot_callback_proc);
```

備註

提供啟動預設輸出或安全輸出值函數的 I-8K 系列模組可以支持這個 API 函數。I-8K 系列模組僅 I-8041RW 提供此功能， I-9K 系列 DO 模組都支援此功能。

3.7.41. pac_GetModuleWDTInterruptStatus

該函數讀取插槽模組看門狗中斷功能狀態。

語法

C++

```
short pac_GetModuleWDTInterruptStatus(
    int slot
);
```

參數

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

回傳值

中斷狀態。

範例

[C]

```
int iSlot = 0;
short sStatus = 0;
sStatus = pac_GetModuleWDTInterruptStatus(iSlot);
```

[C#]

```
int iSlot = 0;
short sStatus = 0;
sStatus = PACNET.PAC_IO.GetModuleWDTInterruptStatus(iSlot);
```

備註

提供啟動預設輸出或安全輸出值函數的 I-8K 系列模組，可以支援這個 API 函數。

I-8K 系列模組僅 I-8041RW 提供此功能， I-9K 系列 DO 模組都支援此功能。

3.7.42. pac_SetModuleWDTInterruptStatus

此功能設定插槽模組的看門狗中斷功能為開啟或關閉。

語法

C++

```
bool pac_SetModuleWDTInterruptStatus(
    int slot,
    short enStatus
);
```

參數

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

enStatus

[in] 中斷狀態。

1: 開啟

0: 關閉

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
int iSlot = 0;
short sStatus = 0;
pac_SetModuleWDTInterruptStatus(iSlot, sStatus);
```

[C#]

```
int iSlot = 0;
short sStatus = 0;
PACNET.PAC_IO.SetModuleWDTInterruptStatus(iSlot, sStatus);
```

備註

提供啟動預設輸出或安全輸出值函數的 I-8K 系列模組可以支援這個 API 函數. I-8K 系列模組僅 I-8041RW 提供此功能, I-9K 系列 DO 模組都支援此功能。

3.8. PWM API (脈衝寬度調變模組控制)

PWM API 只支援 7K/I-87K PWM 模組的操作。

在使用 PWM API 函數前，請參考前面的章節，PAC_IO，了解對本地插槽的定義，以及如何使用遠端 I/O 模組的更多細節。

在開發使用於 WinPAC / XPAC 系列設備上的 I-7K / I-87K PWM 模塊的 C / C ++程序時，需要鏈結 PACSDK.lib 與 PACSDK_PWM.lib 到用戶的開發專案中。
此外，放在 WinPAC 上/ XPAC 系列設備中的可執行程式，必須與 PACSDK.dll 和 PACSDK_PWM.dll 放在一起，才能正確的執行。

在開發.net CF 程式時，該項目僅需引用 PACNET.dll。
在 WinPAC 系列設備中構建的可執程式，僅需要 PACNET.dll 和 PACSDK.dll 檔案，即可執行。

如需了解有關，XPAC/WinPAC 系列兼容的 I-7K / I-87K PWM 模組的更多信息，請參考

I-87K 系列

https://www.icpdas.com/tw/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 模組(如 I-87088W 模組)

I-7000 系列

https://www.icpdas.com/tw/product/guide+Remote__I_O__Module__and__Unit+RS-485__I_O__Modules+I-7000#464

(如 I-7088 模組)

PWM API 套裝函數，並不適用於 I-8K PWM 模組。

支援的 PAC

以下列出 PWM API 函數所支援的 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 函數

以下函數用於檢索或設定 PWM。

PACSDK 函數	PACNET 函數	說明
pac_SetPWMDuty	PWM.SetPWMDuty	設定指定通道的佔空比值。
pac_GetPWMDuty	PWM.GetPWMDuty	讀取指定通道的佔空比值。
pac_SetPWMFrequency	PWM.SetPWMFrequency	設置一指定通道的頻率值。
pac_GetPWMFrequency	PWM.GetPWMFrequency	讀取指定通道的頻率值。
pac_SetPWMMode	PWM.SetPWMMode	設置連續模式對一個指定的通道。
pac_GetPWMMode	PWM.GetPWMMode	讀取連續模式對一個指定的通道。
pac_SetPWMDITriggerConfig	PWM.SetPWMDITriggerConfig	設置硬件觸發指定通道。
pac_GetPWMDITriggerConfig	PWM.GetPWMDITriggerConfig	讀取硬件觸發指定通道。
pac_SetPWMStart	PWM.SetPWMStart	設置 PWM 輸出 port 的狀態。
pac_SetPWMSynChannel	PWM.SetPWMSynChannel	設置一指定通道的 PWM 同步狀態。
pac_GetPWMSynChannel	PWM.GetPWMSynChannel	讀取指定通道的 PWM 同步狀態。
pac_SyncPWMStart	PWM.SyncPWMStart	啟動 PWM 同步。
pac_SavePWMConfig	PWM.SavePWMConfig	保存 PWM 配置。
pac_GetPWMDIOStatus	PWM.GetPWMDIOStatus	讀取 PWM 輸出 port 和 DI port 的狀態。
pac_SetPWMPulseCount	PWM.SetPWMPulseCount	設置一指定通道的 PWM 升壓值。
pac_GetPWMPulseCount	PWM.GetPWMPulseCount	讀取指定通道的 PWM 升壓值。

3.8.1. pac_SetPWMDuty

這個函數用於設置指定 PWM 通道的 duty cycle。

語法

C++

```
bool pac_SetPWMDuty(
    HANDLE PORT,
    int slot,
    short chIndex,
    float duty
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

chIndex

[in] 指定的通道。

duty

[in] 設定給 PWM 模組通道的 Duty cycle 值。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
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

該函數讀取指定通道的佔空比值。

語法

C++

```
bool pac_SetPWMDuty(
    HANDLE PORT,
    int slot,
    short chIndex,
    float*duty
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

chIndex

[in] 指定的通道。

duty

[out] 讀取到 PWM 模組指定通道的 Duty cycle 值。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
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

此功能設置為一個指定通道的頻率值。

語法

C++

```
bool pac_SetPWMFrequency(
    HANDLE PORT,
    int slot,
    short chIndex,
    unsigned long freq
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

chIndex

[in] 指定的通道。

freq

[in] 設定至 PWM 模組指定通道的頻率值。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
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

這個函數讀取一個指定通道的頻率值。

語法

C++

```
bool pac_GetPWMFrequency(
    HANDLE PORT,
    int slot,
    short chIndex,
    unsigned long*freq
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

chIndex

[in] 指定的通道。

freq

[in] 讀取到 PWM 模組指定通道的頻率值。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
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#]

```
//本地 87K 模組
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

此功能設置指定的通道為連續模式輸出

語法

C++

```
bool pac_SetPWMMode(
    HANDLE PORT,
    int slot,
    short chIndex,
    long mode
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

chIndex

[in] 指定的通道。

mode

[in]] 0：禁用 PWM 連續模式

1：啟用 PWM 連續模式

(如果 PWM 連續模式被激活，用於 PWM 的步進值將被自動設置為 1)

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
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#]

```
//本地 87K 模組
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

該函數讀取指定通道的連續模式狀態。

語法

C++

```
bool pac_GetPWMMode(
    HANDLE PORT,
    int slot,
    short chIndex,
    long *mode
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

chIndex

[in] 指定的通道。

mode

[out] 0：禁用 PWM 連續模式

1：使用 PWM 連續模式

(如果 PWM 連續模式被激活，用於 PWM 的步進值將被自動設置為 1)

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
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

此功能設置外部硬體觸發指定通道。

語法

C++

```
bool pac_SetPWMDITriggerConfig(
    HANDLE PORT,
    int slot,
    short chIndex,
    short config
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

chIndex

[in] 指定的通道。

config

[in]] 0：禁用硬件觸發

1：啟用觸發啟動

2：啟用觸發停止

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
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#]

```
//本地 87K 模組
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

該函數讀取指定通道的外部硬體觸發狀態。

語法

C++

```
bool pac_GetPWMDITriggerConfig(
    HANDLE PORT,
    int slot,
    short chIndex,
    short* config
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

chIndex

[in] 指定的通道。

config

[out] 0：禁用硬體觸發

1：啟用觸發啟動

2：啟用觸發停止

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
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#]

```
//本地 87K 模組
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

此功能設置 PWM 輸出 port 的狀態。

語法

C++

```
bool pac_SetPWMStart(
    HANDLE PORT,
    int slot,
    short enStatus
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

enStatus

[in] Bit 0 對應於 PWM 通道 0, Bit 1 對應於 PWM 通道 1....等等。

當該 Bit 為 0，則表示該 PWM 輸出 port 是關閉的，為 1 表示該 PWM 輸出 port 為開啟的

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
short Status= 0x01;
BOOL IRET = pac_SetPWMStart(hPort, iSlot, Status);
uart_Close(hPort);
```

[C#]

```
//本地 87K 模組
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

此功能設置指定通道的 PWM 同步狀態。

語法

C++

```
bool pac_SetPWMSynChannel(
    HANDLE PORT,
    int slot,
    short chIndex,
    short enStatus
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

chIndex

[in] 指定的通道。

enStatus

[in]] 0：禁用 PWM 同步

1：啟用 PWM 同步

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
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#]

```
//本地 87K 模組
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

這個函數讀取一個指定通道的 PWM 同步狀態。

語法

C++

```
bool pac_GetPWMSynChannel(
    HANDLE PORT,
    int slot,
    short chIndex,
    short* enStatus
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

chIndex

[in] 指定的通道。

enStatus

[out] 0：禁用 PWM 同步

1：啟用 PWM 同步

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
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#]

```
//本地 87K 模組
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

此功能啟動 PWM 同步。

語法

C++

```
bool pac_SyncPWMStart(
    HANDLE PORT,
    int slot,
    short enStatus
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

enStatus

[in]] 0：停止 PWM 同步

1：啟動 PWM 同步

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
short Status= 0;
BOOL IRET = pac_SyncPWMStart(hPort, iSlot, Status);
uart_Close(hPort);
```

[C#]

```
//本地 87K 模組
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

該功能可以儲存 PWM 配置。

語法

C++

```
bool pac_SavePWMConfig(
    HANDLE PORT,
    int slot,
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot = 1;
BOOL IRET = pac_SavePWMConfig(hPort, iSlot);
uart_Close(hPort);
```

[C#]

```
//本地 87K 模組
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot = 1;
boolIRET = PACNET.PWM.SavePWMConfig(hPort, iSlot);
PACNET.UART.Close(hPort);
```

3.8.14. pac_GetPWMDIOStatus

該函數讀取 PWM 輸出 port 與 DI/DO port 的狀態。

語法

C++

```
bool pac_GetPWMDIOStatus(
    HANDLE PORT,
    int slot,
    unsigned char pwmBitArr [] ,
    unsigned char diBitArr []
);
```

參數

hPort

[in] 87K 模組, 帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的, 請使用 PAC_REMOTE_IO(0 ... 255)函數。

pwmBitArr

[out] , 其中陣列[0] 對應於 PWM 通道 0 和陣列[1] 對應於 PWM 通道 1 等等。當陣列的值是 0 時, 它表示該 PWM 是關閉, 1 表示在 PWM 處於作動狀態。

diBitArr

[out] 其中陣列[0] 對應的 DI/DO 通道 0 和陣列[1] 對應的 DI 通道 1, 等等。當該位為 0, 則表示該 DI/DO 為關閉 , 1 表示該 DI/DO 是作動。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
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#]

```
//本地 87K 模組
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

此功能設置指定通道的 PWM 步進值(step)。

語法

C++

```
bool pac_SetPWMPulseCount(
    HANDLE PORT,
    int slot,
    short chIndex,
    LONG CNT
);
```

參數

hPort

[in] 87K 模組, 帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的, 請使用 PAC_REMOTE_IO(0 ... 255)函數。

chIndex

[in] 指定的通道。

CNT

[in] PWM 的步進值(step) (0x0001 至 0xFFFF)

(當設置大於 1 步, PWM 的連續模式將被自動設置為禁用)

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
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#]

```
//本地 87K 模組
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

該函數讀取指定通道的 PWM 步進值(step)。

語法

C++

```
bool pac_GetPWMPulseCount(
    HANDLE PORT,
    int slot,
    short chIndex,
    long* CNT
);
```

參數

hPort

[in] 87K 模組，帶入 uart_Open 串口句柄

slot

[in] 接收命令的槽位

如果 I/O 模組是遠端的，請使用 PAC_REMOTE_IO(0 ... 255)函數。

chIndex

[in] 指定的通道。

CNT

[out] PWM 的步進值(Step) (0x0001 至 0xFFFF)

(當設置為大於 1 步時, PWM 的連續模式將被自動設置為禁用)

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
//本地 87K 模組
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#]

```
//本地 87K 模組
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(底板計時器 API)

底板定時器 API 支持硬體定時器，包括計時器超時/計時器 1/計時器 2。

WinCE 平台上開發程式使用的計時器

一般 VC 程式的計時器或.NET CF C#/VB 程式的計時器，由於這兩種計時器無法調整其執行 thread 的優先度，其精度會受系統的排程影響而有毫秒(millisecond)以上的誤差，所以無法實現即時性的要求。

BPtimer 是 ICPDAS PAC 提供的硬體計時器，它提供高解析度的計時器，單位為微秒(microsecond)，可以調整計時器執行緒的優先度以符合即時要求。

底板 API 函數:

(1).pac_SetBPTimerOut

(2).pac_SetBPTimer

(3).pac_KillBPTimer

(4).pac_SetBPTimerInterruptPriority

[使用限制]

1、 BPtimer 提供兩種模式(同一個時間，只能使用一種計時器):

計時器 1: 單位為 1 microsecond

計時器 2: 單位為 10 microsecond。

2、 BPtimer 與底板的 slot 及部份 COM PORT 共用中斷。

同時使用時，中斷會互相影響，而影響其功能。

不建議 BPTimeR 同時間與其它中斷的模組或 COM port 使用。

支援的 PAC

以下列出 Backplane Timber API 函數所支援的 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

底板計時器功能

以下函數用於檢索或設定底板計時器。

PACSDK 函數	PACNET 函數	說明
pac_GetBPTimerTimeTick_ms	BPTimer.GetBPTimerTimeTick_ms	這個函數用於取得從系統啟動至函讀取時經過的時間(毫秒)，但不包括該系統休眠期間
pac_GetBPTimerTimeTick_us	BPTimer.GetBPTimerTimeTick_us	這個函數用於取得從系統啟動至函讀取時經過的時間(微秒)，但不包括該系統休眠期間
pac_SetBPTimer	BPTimer.SetBPTimer	創建一個具可設定超時值的硬體定時器
pac_SetBPTimerOut	BPTimer.SetBPTimerOut	創建一個具可設定高/低波超時值的硬體定時器
pac_SetBPTimerInterruptPriority	BPTimer.SetBPTimerInterruptPriority	設置底板定時器的線程的優先度
pac_KillBPTimer	BPTimer.KillBPTimer	註銷先前由 pac_SetBPTimer 調用設置的定時器

3.9.1. pac_GetBPTimerTimeTick_ms

這個函數用於取得從系統啟動至函讀取時經過的時間(毫秒)，但不包括該系統休眠期間。

語法

C++

```
DWORD pac_GetBPTimerTimeTick_ms(void );
```

參數

此函數沒有參數。

回傳值

取得毫秒數表示成功。

範例

此函數沒有範例。

3.9.2. pac_GetBPTimerTimeTick_us

這個函數用於取得從系統啟動至函讀取時經過的時間(微秒)，但不包括該系統休眠期間。

語法

C++

```
DWORD pac_GetBPTimerTimeTick_us(void );
```

參數

此函數沒有參數。

回傳值

取得微秒數表示成功。

範例

此函數沒有範例。

3.9.3. pac_SetBPTimer

這個函數創建一個具可設定超時值的硬體定時器。

指定一個時間值，當超時發生時，底板會傳送一個中斷信號給系統，然後以訊息事件傳遞給應用程序定義的 CALLBACK 函數。

語法

C++

```
Bool pac_SetBPTimer(int type,unsigned int uElapse, pac_TIMEROUT_CALLBACK_FUNC f);
```

參數

type

[in] 指定計時器的類型。

1(定時器 1)：1 微秒計時器

2(定時器 2)：10 微秒計時器

其他：不適用

uElapse

[in] 指定所用的時間。

定時器 1：一個 TIMEROUT 設定範圍值為 0～65535 的整數，單位:微秒。

定時器 2：一個 TIMEROUT 設定範圍值為 0～65535，單位:10 微秒。

f

CALLBACK 函數。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數失敗，則回傳 0。

範例

[C]

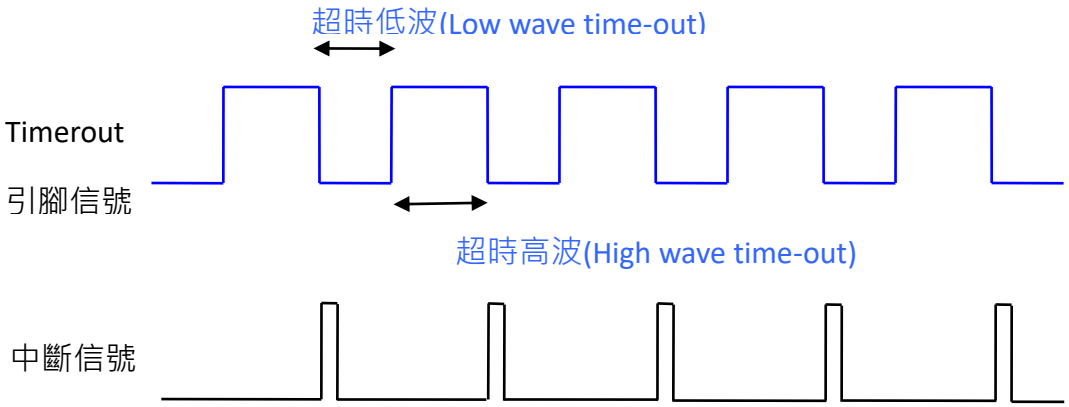
```
int CALLBACK TIMER() //中斷功能
{
   / *將用戶控件添加程式碼在這裡* /
   return 0; //中斷完成
}
//設置定時器以 200 微秒的時間間隔
pac_SetBPTimer(1, 200, TIMER);
```

3.9.4. pac_SetBPTimerOut

這個函數創建一個具可指定高/低波超時值的硬體定時器。

高/低波的超時值被指定，每次超時高波和低波發生時，底板會傳送一個中斷信號給系統和然後以訊息事件傳遞給應用程序定義的 CALLBACK 函數。

在每個 slot 的 TIMEROUT 引腳將被觸發而 TIMEROUT 信號會同時輸出至各個 slot。該 timeoutput 引腳可被用來獲取每個 slot 的同步數據。



語法

C++

```
bool pac_SetBPTimerOut(unsigned int uHighElapse,unsigned int uLOwElapse,
pac_TIMEROUT_CALLBACK_FUNC f);
```

參數

uHighElapse

[in] 指定用於高波 TIMEROUT 信號時間值，範圍從 0～65535 的整數，微秒為單位。

uLOwElapse

[in] 指定用低波 TIMEROUT 信號時間值，範圍從 0～65535 的整數，微秒為單位。

f

CALLBACK 函數。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數失敗，則回傳 0。

範例

[C]

```
int CALLBACK TIMER() //中斷功能
{
  / *將用戶控件添加程式碼在這裡* /
  return 0; //中斷完成
}
//設置定時器以 200 微秒的時間間隔
pac_SetBPTimerOut(200, 300, TIMER);
```

3.9.5. pac_SetBPTimerInterruptPriority

此功能設置底板定時器的線程的優先度。

語法

C++

```
bool pac_SetBPTimerInterruptPriority(
    int type,
    int nPriority
);
```

參數

type

[in] 指定底板計時器。

0：TIMEROUT

1：定時器 1

2：定時器 2

nPriority

[in] 線程的優先度

此值的範圍從 0 到 255，其中 0 為最高優先級。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
int CALLBACK TIMER() //中斷功能
{
   / *將用戶控件添加程式碼在這裡* /
   return 0; //中斷完成
}
pac_SetBPTimer(1, 200, TIMER); //設置定時器以 200 微秒的時間間隔
pac_SetBPTimerInterruptPriority(1, 100); //設置定時器 1 的優先級為 100
```

3.9.6. pac_KillBPTimer

此功能註銷先前由 pac_SetBPTimer 調用設置的定時器。

語法

C++

```
void pac_KillBPTimer(int type);
```

參數

type

[in] 指定計時器。

0(TIMEROUT)

1(定時器 1)：1 微秒計時器

2(定時器 2)：10 微秒計時器

回傳值

此函數沒有回傳值。

範例

[C]

```
//註銷定時器 1
pac_KillBPTimer(1);
```

3.10. Error Handling API (錯誤處理 API)

錯誤處理功能，使您能夠接收，並為您的應用程序顯示錯誤信息。

支援的 PAC

以下列出 Error Handling API 函數所支援的 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_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

錯誤處理函數

以下函數用於錯誤處理。

PACSDK 函數	PACNET 函數	說明
pac_GetLastError	ErrHandling.GetLastError	返回最後一個錯誤的代碼，方便使用者判斷，開發程式出錯時，是發生何種錯誤。
pac_SetLastError	ErrHandling.SetLastError	設置最後錯誤代碼，方便使用者判斷，開發式發生錯誤時，是因何種問題發生錯誤。
pac_GetErrorMessage	ErrHandling.GetErrorMessage	返回錯誤訊息字串。
pac_ClearLastError	ErrHandling.ClearLastError	清除最後的錯誤代碼。

3.10.1. pac_GetLastError

該函數返回最後一個錯誤的代碼，方便使用者判斷，開發程式出錯時，是發生何種錯誤。

語法

C++

```
DWORD pac_GetLastError();
```

參數

此函數沒有參數。

回傳值

返回最後一個錯誤的代碼。

範例

此函數沒有範例。

備註

當函數的返回值指示，這樣的調用，將返回有用數據時，應立即調用 pac_GetLastError 函數。這是因為，某些函數會調用 pac_SetLastError（0），擦除由最近失敗的函數設置的錯誤代碼。

相關範例，請參閱本章中的 pac_GetErrorMerrage。

要獲取 XPAC 錯誤代碼的錯誤字符串，請使用 pac_GetErrorMessage 函數。有關錯誤代碼的完整列表，請參閱附錄 A.系統錯誤代碼。

下表列出了每個功能參考的系統錯誤代碼範圍。

錯誤類型	解釋	範圍
PAC_ERR_SUCCESS	沒有錯誤，成功	0x00000
PAC_ERR_UNKNOWN	發生一個沒有定義的錯誤	0x00001
基本的	定義常見的錯誤條件	0x10000～
記憶體	關於記憶體訪問相關的錯誤	0×11000～
看門狗	關於看門狗相關的錯誤	0x12000～
中斷	關於系統中斷相關的錯誤	0x13000～
UART	關於 UART 協議的錯誤	0x14000～
IO	關於 I/O 模組錯誤	0x15000～
用戶	關於用戶的錯誤	0x20000～

3.10.2. pac_SetLastError

設置最後錯誤代碼。開發程式如果發生錯誤時，方便使用者判斷是何種問題引起的錯誤。

語法

C++

```
void pac_SetLastError(
    DWORD errno
);
```

參數

errno

[in] 指定的最後一個錯誤代碼。

回傳值

此函數沒有回傳值。

範例

此函數沒有範例。

備註

應用程序可以有選擇地檢索該函數設置使用 pac_GetLastError 函數的值。

錯誤代碼被定義為 DWORD 值。如果要定義一個錯誤代碼，請確保您的錯誤代碼不會與 PACDK 定義的錯誤代碼衝突。

我們建議您的錯誤代碼應大於地址 0x20000。

有關錯誤代碼定義的更多信息，請參閱 pac_GetLastError 本文件中。

3.10.3. pac_GetErrorMessage

輸入錯誤碼至此函數，返回錯誤訊息字串。

語法

C++

```
void pac_GetErrorMessage(
    DWORD dwMessageID,
    LPTSTR lpBuffer
);
```

參數

dwMessageID

[in] 錯誤碼。

lpBuffer

[out] 一個指向接收錯誤消息的緩衝區的指標。

回傳值

此函數沒有回傳值。

範例

[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;
            }
        }
   }
}
```

備註

該 pac_GetErrorMessage 函數可以用來獲得通過 pac_GetLastError 返回的 XPAC 錯誤代碼錯誤信息字串，如下面的示例所示。

```
TCHAR Buffer[32] ;
pac_GetErrorMessage(pac_GetLastError(), Buffer);
MessageBox( NULL, Buffer, L"Error", MB_OK | MB_ICONINFORMATION );
```

3.10.4. pac_ClearLastError

清除最後的錯誤代碼。

語法

C++

```
void pac_ClearLastError();
```

參數

此函數沒有參數。

回傳值

此函數沒有回傳值。

範例

此函數沒有範例。

備註

該 pac_ClearLastError 函數清除最後一個錯誤。

3.11. Misc API(雜項 API)

支援的 PAC

以下列出 Misc API 函數所支援的 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

雜項函數

以下函數用於檢索或設定記憶體。

PACSDK 函數	PACNET 函數	說明
byte AnsiString	MISC.AnsiString	將 unicode 字串，轉換為 ANSI byte 陣列。
WideString	MISC.WideString	將 ANSIbyte 陣列，轉換為 Unicode 字串。
pac_AnsiToWideString	N/A	將 ANSI 字串，轉換為 Unicode 字串。
pac_WideToAnsiString pac_WideStringToAnsi	N/A	將 Unicode 字串，轉換成 ANSI 字串。
pac_DoEvent/pac_DoEvents	N/A	處理所有事件。
pac_GetCurrentDirectory	MISC.GetCurrentDirectory	獲取，當前應用程序的所在目錄。
pac_GetCurrentDirectoryW	N/A	獲取，當前應用程序的所在目錄，目錄名稱字串為 TCHAR 類型。

3.11.1. byte AnsiString

這個函數將 unicode 字串，轉換為 ANSI byte 陣列。

語法

C#

```
byte [] AnsiString(
    string str
);
```

參數

str

[in] 指向 Unicode 字串進行轉換。

回傳值

返回 ANSI byte 陣列。

範例

[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);
```

範例

NET 中，如果我們要轉換一個 Unicode 字串為 ANSI 或反之，我們應該通過轉換成 byte 陣列。

3.11.2. WideString

這個函數轉換成一個 ANSI byte 陣列轉換為 Unicode 字串。

語法

C#

```
String WideString(
    byte [] CharStr
);
```

參數

CharStr

[in] 要轉換的 ANSI byte 陣列指標。

回傳值

返回 Unicode 字串。

範例

[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);
```

範例

.NET 中，如果我們要轉換一個 Unicode 字串為 ANSI，或反之，我們應該通過轉換成 byte 陣列.

3.11.3. pac_AnsiToWideString

該函數將 ANSI 字串，轉換為 Unicode 字串。

語法

C++

```
void pac_AnsiToWideString(
    LPCSTR ASTR,
    LPTSTR WSTR
);
```

參數

ASTR

[in] ANSI 字串指標。

WSTR

[in] 一個指標，接收轉換後的 Unicode 字串緩衝區的位置。

回傳值

此函數沒有回傳值。

範例

[C]

```
char ansiString[128] = "This is an ansi string";
TCHAR uniString[128] ;
pac_AnsiToWideString(ansiString, uniString);
MessageBox(NULL, uniString, NULL, MB_OK); //"This is an ansi string"'將正確顯示在訊息框
```

備註

字串緩衝區的最大大小為 2 Kbyte .

3.11.4. pac_WideToAnsiString/pac_WideStringToAnsi

該函數將 Unicode 字串，轉換成 ANSI 字串。

語法

C++

```
void pac_WideToAnsiString(
    LPCTSTR WSTR,
    LPSTR ASTR
);
```

參數

WSTR

[in] 進行轉換的 Unicode 字串指標。

ASTR

[in] 一個指標，指向接收轉換 ANSI 字串緩衝區的位置。

回傳值

此函數沒有回傳值。

範例

[C]

```
TCHAR uniString[128] = TEXT("This is a unicode string");
char ansiString[128] ;
pac_WideStringToAnsi(uniString, ansiString);
printf("%s", ansiString);    //"This is a unicode string"將正確顯示在提示命令字元頁面上
```

備註

字串緩衝區的最大為 2byte。

3.11.5. pac_DoEvent/pac_DoEvents

當您運行 Windows 窗體時，它將創建一個新窗體，然後等待事件處理。

每次表單處理事件時，它會處理與該事件關聯的所有程式碼。 所有事件在隊列中等待。當您的程式碼處理事件時，您的應用程序不回應或有所反應。 如果您在程式碼中調用 pac_DoEvents，則應用程序可以處理其他事件。

語法

C++

```
void pac_DoEvents();
```

參數

此函數沒有參數。

回傳值

此函數沒有回傳值。

範例

[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

此函數獲取，當前應用程序的所在目錄。

語法

C++

```
bool pac_GetCurrentDirectory(
    char* cBuffer,
    DWORD nSize
);
```

參數

cBuffer

[out] 接收當前目錄名稱字串緩衝區的指標。

nSize

[in] 用於指定緩衝區的字串長度。

回傳值

如果函數調用成功，則回傳值是 TRUE。

如果函數失敗，則回傳值是 FALSE。

若想獲得更多的錯誤信息，調用 Windows API 的 GetLastError 來獲最後一個錯誤碼。

範例

[C]

```
char buf[1024] ;
pac_GetCurrentDirectory(buf, 1024);
```

[C#]

```
string str;
str = PACNET.MISC. GetCurrentDirectory();
```

3.11.7. pac_GetCurrentDirectoryW

此函數獲取當前應用程序的所在目錄，目錄名稱字串為 TCHAR 類型。

語法

C++

```
bool pac_GetCurrentDirectoryW(
    LPTSTR cBuffer,
    DWORD nSize
);
```

參數

LPTSTR

[out] 接收當前目錄字串的緩衝區指標。

nSize

[in] 指定緩衝區的大小。

回傳值

如果函數調用成功，則回傳 TRUE。

如果函數調用失敗，則回傳 FALSE。若想獲得更多的錯誤訊息，請調用 pac_GetLastError()。

範例

[C]

```
TCHAR buf[1024] ;
pac_GetCurrentDirectory(buf, 1024);
```

附錄

A. System Error Codes (錯誤代碼)

這下表提供了系統錯誤代碼的列表。有些時候，很多功能未能在 pac_GetLastError 功能開啟。要檢索您的應用程序的錯誤，使用 pac_GetErrorMessage 功能。

錯誤代碼	錯誤信息
0x00001	未知錯誤
0x10001	slot 登記錯誤
0x10002	未註冊 slot 錯誤
0x10003	未知模組
0x10004	模組不存在
0x10005	無效的 COM port 號碼
0x10006	不支持的功能
0x10007	模組不存在
0x10008	未註冊 slot 錯誤
0x11001	EEPROM 的訪問無效的地址
0x11002	SRAM 的存取無效的地址
0x11003	SRAM 的存取無效的類型
0x11004	NVRAM 存取無效的地址
0x11005	EEPROM 寫入保護
0x11006	EEPROM 寫入失敗
0x11007	EEPROM 讀失敗
0x12001	該輸入值無效
0x12002	該 WDT 不存在
0x12003	該 WDT 初始化錯誤
0x13001	創建中斷事件失敗
0x14001	UART 校驗和錯誤
0x14002	UART 讀取超時
0x14003	UART 響應誤差
0x14004	在輸入範圍 UART

錯誤代碼	錯誤信息
0x14005	UART 超過輸入範圍
0x14006	UART 打開存檔
0x14007	UART 通訊獲得調製解調器狀態錯誤
0x14008	UART 得到錯誤的線路狀態
0x14009	UART 內部緩衝區溢出
0x15001	I/O 卡不支持此 API 函數
0x15002	API 函數不支持此 I/O 卡
0x15003	slot 的價值超過其範圍
0x15004	通道的輸出值，超過其範圍
0x15005	增益值超出其範圍
0x15006	沒有支援的中斷模組
0x15007	I/O 值超出範圍
0x15008	I/O 通道超出範圍
0x1500A	DIO 通道不能覆蓋
0x1500B	AI/O 通道不能覆蓋
0x16001	底板計時器註冊
0x16002	底板計時器沒註冊

B. API 比較

下表列出了每個 API 函數，對應的產品是否有支援，其中 Y 表示支持，X 則表明不支持

系統訊息函數

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
pac_GetCPUVersion	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_EnableLED	Y	Y	Y	Y	-	-	Y	Y	Y

系統訊息函數

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_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

系統訊息函數

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_RegistryHotPlug(Beta 測試)	-	-	-	-	-	-	-	-	-
pac_UnregistryHotPlug(Beta 測試)	-	-	-	-	-	-	-	-	-
pac_SetBackLight	Y	Y	Y	Y	Y	Y	Y	Y	Y
pac_GetBackLight	Y	Y	Y	Y	Y	Y	Y	Y	Y

中斷功能函數

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

記憶體存取函數

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 系列產品只支援 type 1 (EEPROM),沒有支援 type 0 (SRAM)。

看門狗函數

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

註冊表函數

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_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 port 函數

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 函數

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 函數

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_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_IO 函數

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_ReadModuleSafeValueDO/pac_ReadModuleSafeValueDO_MF	Y	-	Y	-	Y	Y	Y	-	Y
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_IO 函數

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_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
pac_InitModuleWDTInterrupt	Y	-	Y	-	Y	Y	Y	-	Y
pac_GetModuleWDTInterruptStatus	Y	-	Y	-	Y	Y	Y	-	Y
pac_SetModuleWDTInterruptStatus	Y	-	Y	-	Y	Y	Y	-	Y

PWM 函數

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

底板計時器函數

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

雜項:字串轉換、目前執行程式所在資料夾路徑 函數

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 是 XPACSDK 和 WinPACSDK 的下一個版本。
它的基礎建立在 XPACSDK 和 WinPACSDK 函數褲，並提供以下內容：

C.1. PACSDK.dll 修改和更新

新的 PACSDK.dll 提供對兩個平台的支持，一個是為 WinPAC 系列（ARM 平台）設計的，另一個是 XPAC 系列（x86 平台）。
但是，下面列出，在新的 PACSDK 中包含了一些修改和更新。
（注意：與舊的 WinPAC / XPAC SDK 相比，需要對舊的 WinPAC / XPAC 程序進行這些修改和更新，以便它能夠與新的 SDK 配合使用）

1. pac_EnableLED

原來的 pac_EnableLED（bool bFlag）函數只能用於之前的 SDK 中的 WinPAC 系列，原來的 pac_EnableLED（int pin，bool bFlag）功能只能用於之前的 SDK 中的 XPAC 系列。
因此，由於參數衝突，此 API 函數無法集成到 PACSDK.dll

因此，PACSDK.dll 中的函數已更改為 pac_EnableLED（bool Flag）已保留，並添加了一個新的 API 函數，pac_Enable LED（int pin，bool bFlag）。

2. 添加註冊表 API，用於 XPAC 系列

下面列出的 API 函數套件在之前的 SDK，XPACSDK_CE.dl 中未提供，並且在 XPAC 系列的新 PACSDK.dll 中支援。
(舊的 SDK，WinPAC 系列的 WinPACSDK.dll 和新的 SDK，WinPAC 系列的 PACSDK.dll 提供了以下所有功能的支持)

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. 添加 I/O 的看門狗定時器，啟動預設輸出值/看門狗安全輸出值的 API 於純 DIO 模組

新 PACSDK.dll 提供 I/O 看門狗定時器，啟動預設輸出值和安全輸出值功能的純 DIO DCON 模組的支持。

舊的 SDK，WinPacSDK.dll 和 XPacSDK_CE.dll 不支持這些功能(詳見註 1)。

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

備註

1. 在每個 API 函數用於其設置有啟動預設輸出值或安全輸出值函數的 DCON 模組。
2. 具備啟動預設輸出值或安全輸出值函數 I-7K/I-87K 系列模組可以支持上述的 API 函數。

I-8K 系列模組僅 I-8041RW 提供此功能, I-9K 系列 DO 模組都支援此功能。

4 新增多功能模組的 I/O 連接的 API 函數

新 PACSDK.dll 提供 I/O 的連接功能支持(包括寫/讀 DIO，AIO，讀取 DI 計數器和 I/O 看門狗定時器，Power-ON 和 Safe value 函數為多功能 DCON 模組(請參閱附註 2 關於多功能模組的定義)不支持舊的 SDK(WinPAC.dll 和 XPACSDK_CE.dll)。

pac_WriteAO_MF(註 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

備註

1. pac_ReadDOBit，pac_ReadDICNT 和 pac_ClearDICNT 函數，這是在支援於舊 SDK，用於存取純 DIO DCON 模組的 DIO 通道，純 DIO DCON 模組定義為模組只有 DI，DO 或 DIO 通道。
2. 以上加_MF 的 API 函數是用於存取多功能 DCON 模組的 DIO 通道。多功能 DCON 模組定義為 AIO 或計數器模組具有 DIO 通道(如 I-87005W/I-87016W/I-87082W/I-7016/I-7088 等。)
3. 該功能如上所述(pac_WriteDO / pac_ReadDIO 等)不能被用來訪問多功能 DCON 模組。只有 pac_xxx_MF 這些 API 函數才允許存取多功能 DCON 模組。
4. PAC_IO API 函數發送 DCON 命令，無需先判斷模組名稱。PAC_IO API 函數可以存取到該 I-87K/I-8K ，I-7000 系列模組，I-8000 系列模組單元，tM 系列模組，和其他的 OEM / ODM DCON 模組。
5. pac_WriteAO / pac_WriteAO _MF 功能和可用的模組比較表如下：

自 2012 年 11 月 1 日

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. 新增雜項 API 於 PACSDK

PACSDK.dll 提供以下兩項雜項的 API 函數。

pac_GetCurrentDirectory

pac_GetCurrentDirectoryW

6. 添加 XPAC 系列的記憶體保留區

為了保留一些存儲器 EEPROM 和 SRAM 的部分給系統使用，該 pac_ReadMemory 和 pac_WriteMemory 功能的保留部分必須改變。

保留的部分與 WinPAC 上的 SDK 相同。保留的部分記憶體定義為

EEPROM

0 ~ 0x1FFF 的(8 KB)為用戶

0x2000 ~ 0x3FFF(8 KB)保留給系統

SRAM

允許被使用者使用的 SRAM 記憶體範圍為 0 ~ 0x6FFFF(448KB)，另 64KB SRAM 保留給系統使用。

舊的 XPAC SDK(XPacSDK_CE.dll)，EEPROM 的所有存儲器空間(0～0x3FFF，16KB)和 SRAM 的所有存儲器空間(0～0x80000，512KB) 是全部可讓使用者使用。

7. 在 C 程序中使用新的 SDK（PACSDK）

要在 C 的程序中使用新的 PACSDK，需要在程序中更改一些程式碼。

由 PACSDK.h 取代先前的頭文件

```
#include “WinPacSDK.h”
```

改變成

```
#include “PACSDK.h”
```

WinPacSDK.h 同時用於 WinPAC 或 ViewPAC 系列，必須被替換為 PACSDK.h

```
#include “XPacSDK_CE.h”
```

改變成

```
#include “PACSDK.h”
```

XPacSDK_CE.h 用於 XPAC 系列，並且必須被替換為 PACSDK.h

取代 lib 函數褲文件為 PACSDK.lib

WinpacSDK.lib //WinPAC 上或系列的 ViewPAC

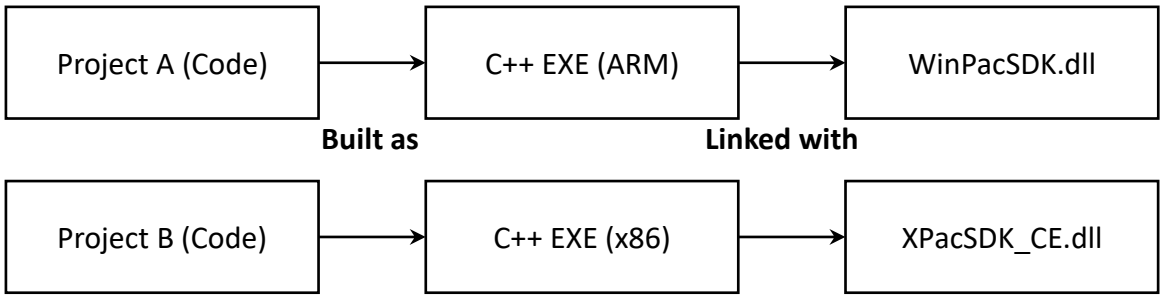
XPacSDK_CE.lib //XPAC 系列

改變成

PACSDK.lib

WinPacSDK.lib 用於 WinPAC 或 ViewPAC 系列，XPacSDK_CE.lib 用於 XPAC 系列，皆由 PACSDK.lib 取代

在原有流程的 C 程序所調用舊的 SDK 的說明如下



即使應用於 WinPAC 上系列模組的 Project A，和應用於 XPAC 系列模組的 Project B，兩個功能相同。使用舊的 SDK，源程式碼部分也不會完全相同，因為使用不同的頭文件，並且在前面的 SDK 中定義的幾個函數名和錯誤代碼也是不同的。

因此，項目 A 和項目 B 被視為單獨的程序，不能共享源程式碼

以上的結果是

Project A 被建構為基於 ARM 的可執行程序，它必須與 WinPacSDK.dll 運行。

Project B 被構建為基於 x86 的可執行程序，它必須與 XPacSDK_CE.dll 運行。

另一個 C 專案，現在引用新的 SDK(PACSDK.dll)的流程圖如下所示：

使用新 SDK 的好處是

應用於 WinPAC 系列模組的程序和應用於 XPAC 系列模組的其他程序在功能上是相同的，因為在程式庫上使用的頭文件和 API 函數和錯誤代碼是完全相同的，源程式碼可以共享兩個程序。

具有共享源程式碼的項目可以構建為兩個不同的平台可執行程序，在構建項目時選擇開發環境中的不同平台設置。

上述的結果是構建為基於 ARM 項目的可執行程序，該程序使用基於 ARM 的 PACSDK.dll 運行，並且還構建為基於 x86 的可執行程序，該程序使用基於 x86 的 PACSDK.dll 進行運行。

C.2. PACNET SDK 修改和更新

.NET Compact Framework 環境允許在不同平台上使用多種高級語言（C#，VB），而不需要為特定的架構重寫。

新的 PACNET.dll 替換了以前的.NETCF SDK(WinPacNet.dll 及 XPacNet.dll)，這意味著鏈接到 WinPAC 設備上的 PACNET.dll 的 NET CF 程序可以遷移到 XPAC 設備，而無需重寫程式碼或重建項目，反之亦然。

1. API 函數分類

WinPacNet.dll 或 XPacNet.dll 的所有 API 函數都放在 WinPacNet.WinPAC.xxx/XPacNET.XPac.xxx 類中，但 PACNET.dll 的 API 函數分為 PACNET.sys、PACNET.Memory、PACNET.Interrupt 等

應用於 API 用戶手冊中定義的 PACNET.dll 的 API 函數的分類如下。

在 API 手冊中分類	類名在 PACNET.dll
2.1. 系統信息的 API	Sys
2.1. 蜂鳴器 API	Sys.Buzzer
2.2. 中斷 API	Interrupt
2.3. 內存訪問 API	Memory
2.4. 看門狗的 API	Sys.WDT
2.5. 註冊表 API	PAC_Reg
2.6. UART 的 API	UART
2.7. PAC_IO API	PAC_IO
2.8. PWM API	PWM
2.9. 底板定時器 API	BPTimer
2.10. 錯誤處理的 API	ErrHandling
2.11. 雜項的 API	MISC

2. API 函數修改

LED 控制 API 函數(pac_EnableLED)

參閱 PACSDK.dll 修改和更多的細節更新 “pac_EnableLED”。

在 PACNET 的 LED 函數為:

XPacNet.dll: XPacNet.XPac.pac_EnableLED (pin,bFlag)

PACNET.dll: PACNET.Sys.EnableLEDs (pin,bFlag)

加為 XPAC 系列註冊表 API

請參閱 “添加為 XPAC 系列註冊表 API” 的 PACSDK.dll 修改和更多的細節更新。

註冊表 API 函數組被放置在 PACNET.PAC_Reg 類別。

添加 I/O 的看門狗定時器，啟動預設輸出值/看門狗安全輸出值的 API，於純 DIO 模組

請參閱 “添加 I/O 的看門狗定時器，啟動預設輸出值/看門狗安全輸出值的 API，於純 DIO 模組” 的參考 PACSDK.dll 修改和更新，更多的細節。

在 I/O WDT 的套件，函數被放置在 PACNET.PAC_IO 類別。

添加 I/O 看門狗定時器，啟動預設輸出值/看門狗安全輸出值的 API，於多功能模組

請參閱 “添加 I/O 的看門狗定時器，啟動預設輸出值/看門狗安全輸出值的 API，於純 DIO 模組” 的參考 PACSDK.dll 修改和更新，更多的細節。

在 I/O WDT 的套件，啟動預設輸出值/看門狗安全輸出值的 API 函數的多功能模組也放在 PACNET.PAC_IO 類別。

新增雜項 API 函數的 PACSDK

請參閱 “新增雜項的 API 函數 PACSDK” 的 PACSDK.dll 修改和更多的細節更新的參考。

雜項套件 API 函數放在 PACNET.MISC 類別中。

3. 列舉的錯誤代碼

添加一個函數來列舉 PACSDK 的所有錯誤代碼。

程式碼片段如下 (該程式碼適用於每個 C#/VB 演示文件)

```
uint ec = PACNET.ErrHandling.GetLastError();
MessageBox.Show(((PACNET.ErrCode)ec).ToString() + "\nError Code: 0x" + ec.ToString("X"));
```

該示例程式碼是用來顯示錯誤代碼號和其產生數字的定義。

如果用戶程序發生最後一個錯誤代碼 0x10001。

將顯示帶有 “PAC_ERR_UNKNOWN 錯誤代碼：0x10001” 標題的消息框。

4. 在 C#或 VB.net 程序中使用新的 SDK（PACNET）

要使用新的 PACNET 在 C#或 VB.net 程序，專案中，有些程式碼需要改變。

C#程序

修改 XPAC 系列設備的程式碼，“using XPACNET” → “using PACNET”。

```
Using XPacNet;
```

改變成

```
Using PACNET;
```

修改 WinPAC 系列設備的程式碼，“using WinPacNet” → “using PACNET”。

```
Using WinPacNet;
```

改變成

```
Using PACNET;
```

VB.net 程序

修改 XPAC 系列設備的程式碼，“Imports XPacNET” → “Imports PACNET”。

```
Imports XpacNet
```

改變成

```
Imports PACNET
```

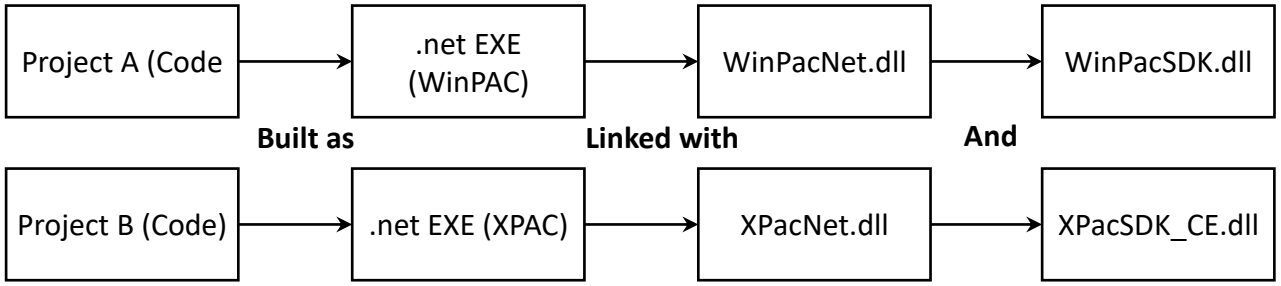
修改 WinPAC 系列設備的程式碼，“Imports WinPacNet” → “Imports PACNET”。

```
Imports WinPacNet
```

改變成

```
Imports PACNET
```

使用前.NETCF 程式庫(WinPacNet.dll 或 XPacNet.dll)，該流程圖如下：

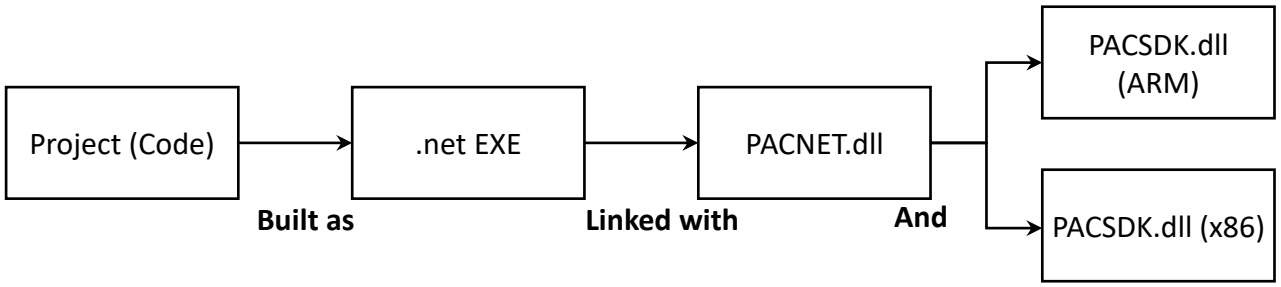


適用於 WinPAC 上系列模組的 A 專案和適用於 XPAC 系列模組的 B 專案在功能上是相同的。但源程式碼 不完全一樣，使用不同的.NET CF 的程式庫和一些函數名和錯誤代碼是不同的。因此，A 專案和 B 專案被視為單獨的程序。

WinPAC 系列上的 A 專案被構建為必須與 WinPacNet.dll 和 WinPacSDK.dll 搭配才能運行的可執行程序。

XPAC 系列的 B 專案被構建為必須與 XpacNet.dll 和 XPacSDK_CE.dll 搭陪才能運行的可執行程序。

新的 NETCF 庫(PACNET.dll)和流程圖變為：



使用新 PACNET.dll 的好處是

應用於 WinPAC 系列模組的程序和 XPAC 系列模組的其他程序功能相同，因為使用相同的.NET CF 庫，庫中的 API 函數和錯誤代碼完全相同，所以可以為兩個程序共享源代碼。
一個共享源代碼可以構建為可執行程序，並鏈接相同的.NET CF 庫（PACNET.dll）。
唯一的變化是鏈接不同的平臺本機 SDK。（PACSDK.dll（ARM）在 WinPAC 系列上使用，PACSDK.dll（x86）用於 XPAC 系列）。

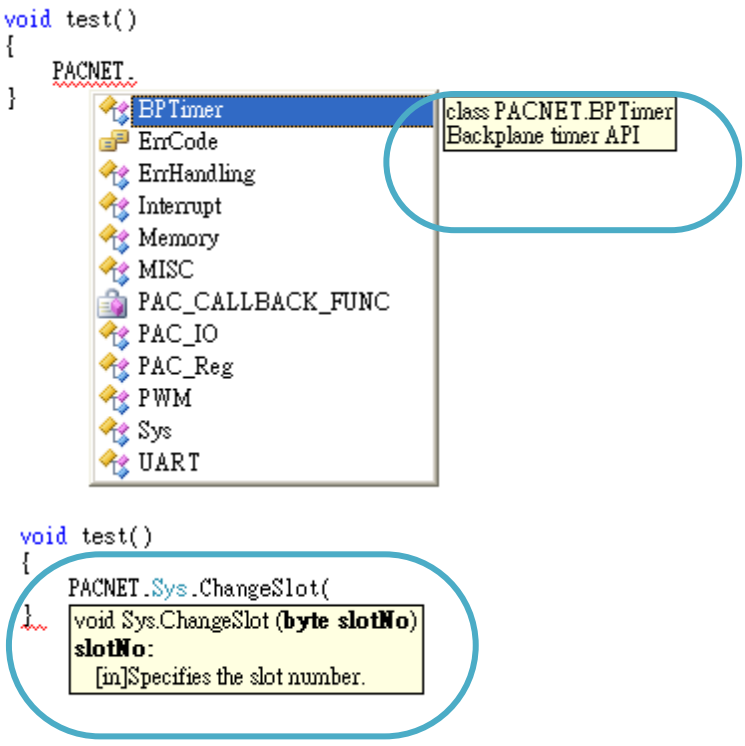
注意

PACNET.dll 是在.Net CF V2.0 環境開發，可以在所有 XPAC 和 WinPAC 系列設備上使用。

5. 顯示 PACNET.dll 類別的工具提示

當在 VS2005/VS2008 IDE 中開發程序時，輸入對系統類或命名空間或 roll over 類別的引用，工具提示彈出在您的滑鼠游標上，不僅給出了方法的參數和變量，還提供了這些方法的一些描述，類和命名空間。

關於工具提示的說明與 PAC API 手冊中相同。（參照下圖）



注意

該 PACNET.dll 必須和 PACNET.xml 放置在同一個文件夾，工具提示將在 Visual Studio IDE 中顯示。

C.3. 錯誤代碼修改和更新

1. WinPAC 系列

修改

WinPacSDK.H 中定義錯誤代碼，PAC_ERR_EEP_ACCESS_RESTRICTION 和 PAC_ERR_SRAM_INVALID_TYPE，在 PACSDK.h 被修改為 PAC_ERR_EEP_INVALID_ADDRESS 和 PAC_ERR_MEMORY_INVALID_TYPE。

錯誤代碼(PAC_ERR_MEMORY_BASE + 1)

PAC_ERR_EEP_ACCESS_RESTRICTION

改變成

PAC_ERR_EEP_INVALID_ADDRESS

錯誤代碼(PAC_ERR_MEMORY_BASE + 3)

PAC_ERR_SRAM_INVALID_TYPE

改變成

PAC_ERR_MEMORY_INVALID_TYPE

增加

```
//基本
PAC_ERR_MODULE_UNEXISTS                    (PAC_ERR_BASE + 7)
PAC_ERR_INVALID_SLOT_NUMBER                (PAC_ERR_BASE + 8)
//中斷
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. XPAC 系列

修改

定義 XPacSDK_CE.h 中錯誤代碼 PAC_ERR_INTR_CRATE_EVENT_FAILURE 拼寫錯誤，它在 PACSDK.h 為糾正 PAC_ERR_INTR_CREATE_EVENT_FAILURE

//中斷程序

錯誤代碼(PAC_ERR_INTR_BASE + 1)

PAC_ERR_INTR_CRATE_EVENT_FAILURE

改變成

PAC_ERR_INTR_CR Ë ATE_EVENT_FAILURE

//基本

PAC_ERR_MODULE_UNEXISTS

原來的錯誤碼：PAC_ERR_BASE + 4

改變成

PAC_ERR_BASE + 7

增加

//基本

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)

//內存訪問

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)

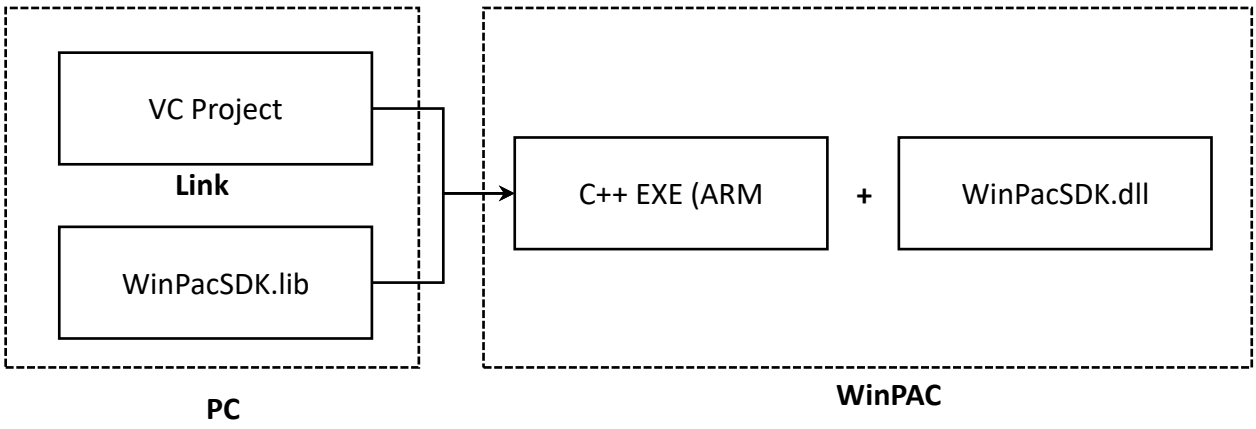
PAC_ERR_IO_AO_CANNOT_OVERWRITE (PAC_ERR_IO_BASE 11)

D. 使用多功能 DCON 模組

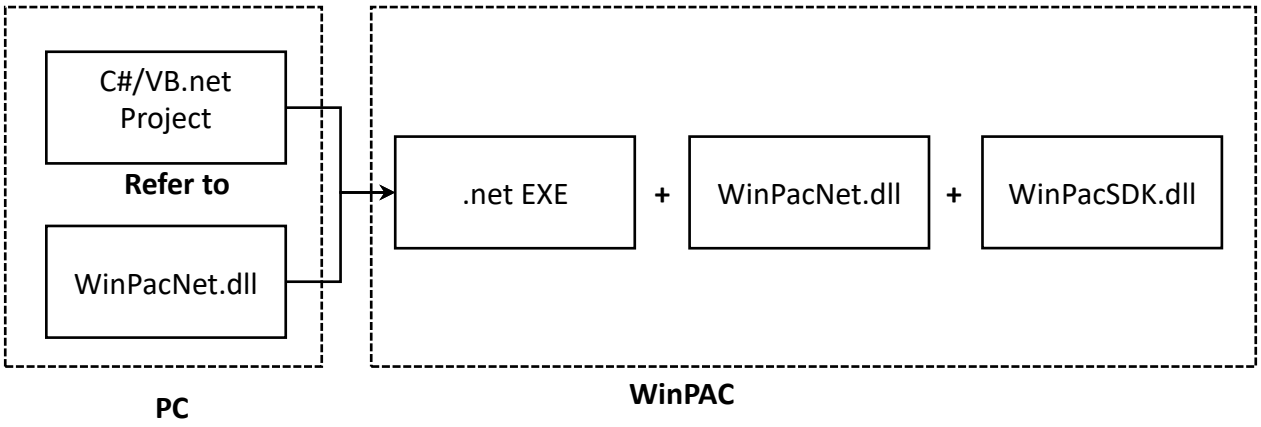
D.1. 在 WinPAC 上的設備

1. 在用戶已經使用 WinPAC 上一系列的設備和它們的方案，是基於舊的 SDK(WinPacSDK.dll/WinPacNet.dll)與舊的 DCON 模組(注 2)，在 WinPAC 上的設備和不使用多功能 DCON 模組(注 1)。

   用戶的程序可以繼續使用舊程式庫，而無需進行修改。
   (舊的 SDK 將繼續維護(修正錯誤)和定期更新，但不會增加新的功能)
   使用舊的 SDK 為下面的流程圖：
   在 VC++項目需要在構建時鏈接 WinPacSDK.lib，而構建在 WinPAC 系列設備中的可執行文件必須與 WinPacSDK.dll 一起使用。



   在構建 C#/VB.net 項目時，需要引用 WinPacNet.dll，WinPAC 系列設備中構建的可執行文件，必須與 WinPacNet.dll 和 WinPacSDK.dll 配合使用。

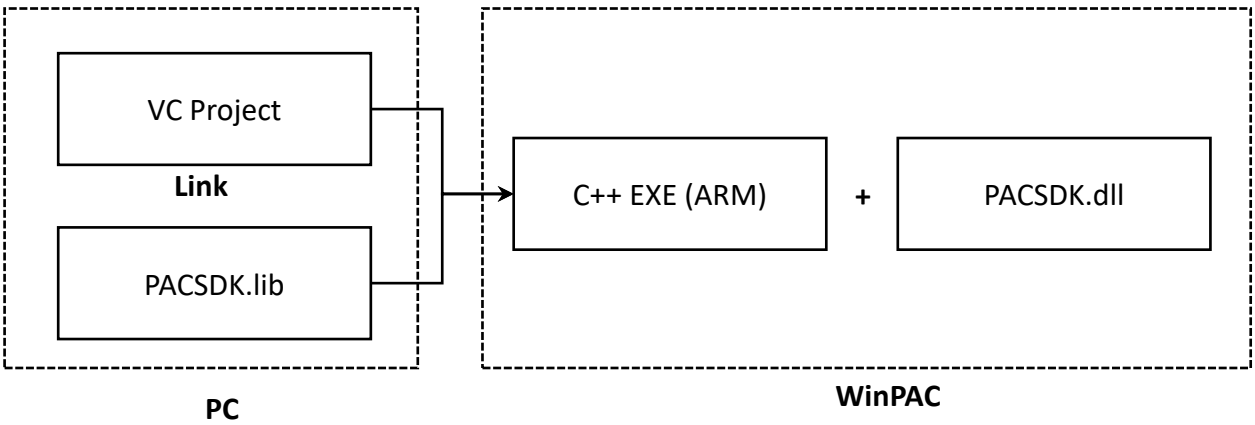


2. 用戶使用的是 WinPAC 系列設備，它們的程序是基於使用舊的 SDK (WinPacSDK.dll)運作於 WinPAC 設備上舊 DCON 模組和多功能 DCON 模組。

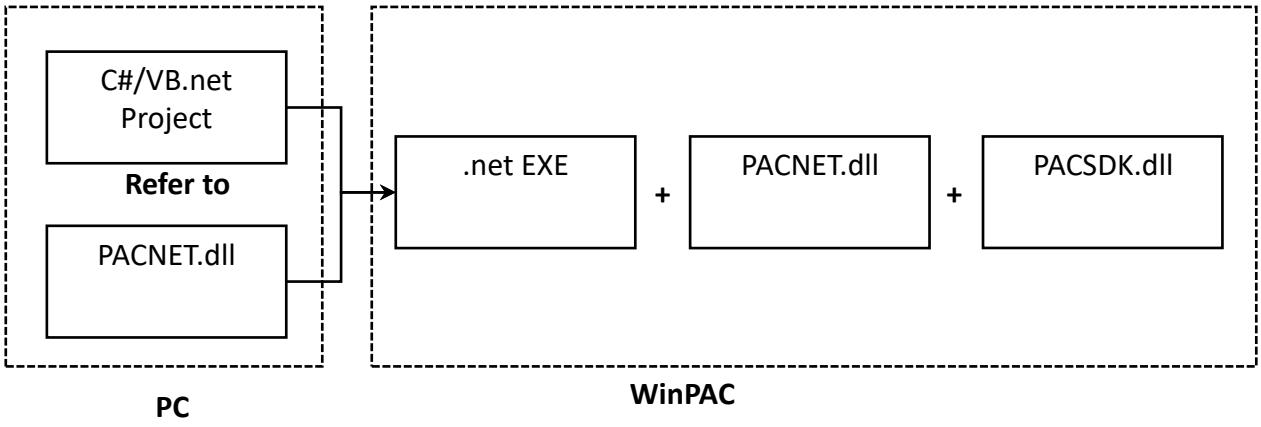
   新的 PACSDK.dll 提供了允許訪問多功能模組的 pac_xxx_MF API 函數，因此必須更新代碼(使用新的 PACSDK.dll)，才能在程序中使用。
   (更多細節請參閱 W6-10_How_to_update_to_PACSDK_library_from_ WinPacSDK_library_ EN .pdf)

3. 用戶從未使用過 WinPAC 系列設備。他們的程序將使用新 SDK，基於舊 DCON 模組或多功能模組。我們的 API 手冊提供 PACSDK.dll 的說明，並且附帶 CD / FTP 中的範例程序與新的 PACSDK.dll 相鏈接，

   所以在開發基於新的 PACSDK.dll 的新程序時，用戶應該參考範例程序，並遵循 API 指令，而不是那些舊的使用 WinPACSDK 的範例程式。
   使用新的 SDK 時，參考下面的流程圖：
   在建立 VC 項目時，需要連接 PACSDK.lib，並且 WinPAC 系列設備中構建的可執行文件必須與 PACSDK.dll 一起使用。



C#/VB.net 項目在構建時需要參考 PACNET.dll，並且放置在 WinPAC 系列設備中的構建的可執行檔，必須與 PACNET.dll 和 PACSDK.dll 配合使用。



備註

多功能 DCON 模塊定義為主要作為 AIO 或計數器但配有 DIO 通道的模組。(如 I-87005W/ I-87016W/I-87082W/I-7016 /I-7088 等）

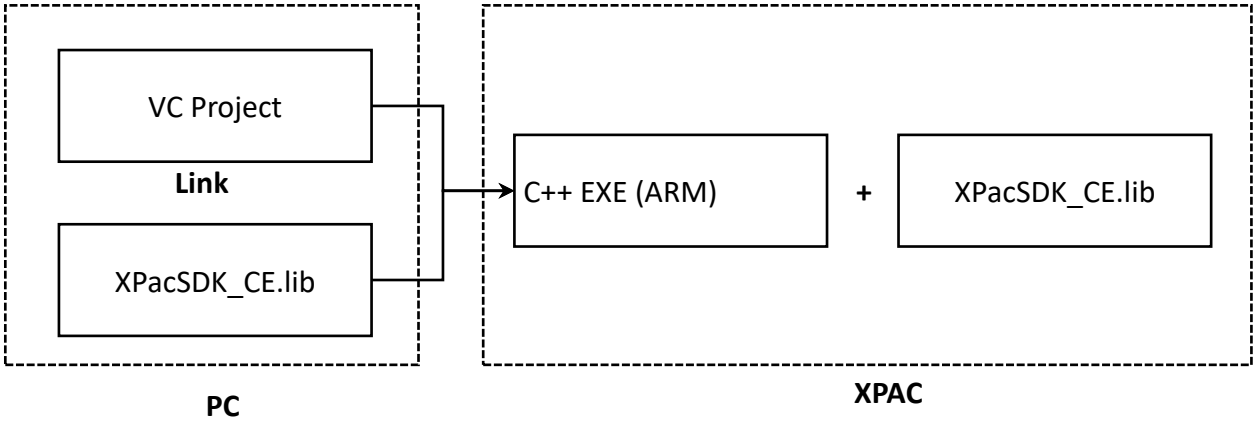
舊 DCON 模塊定義：非多功能 DCON 模組定義為舊 DCON 模組。

D.2. 在 XPAC 上的設備

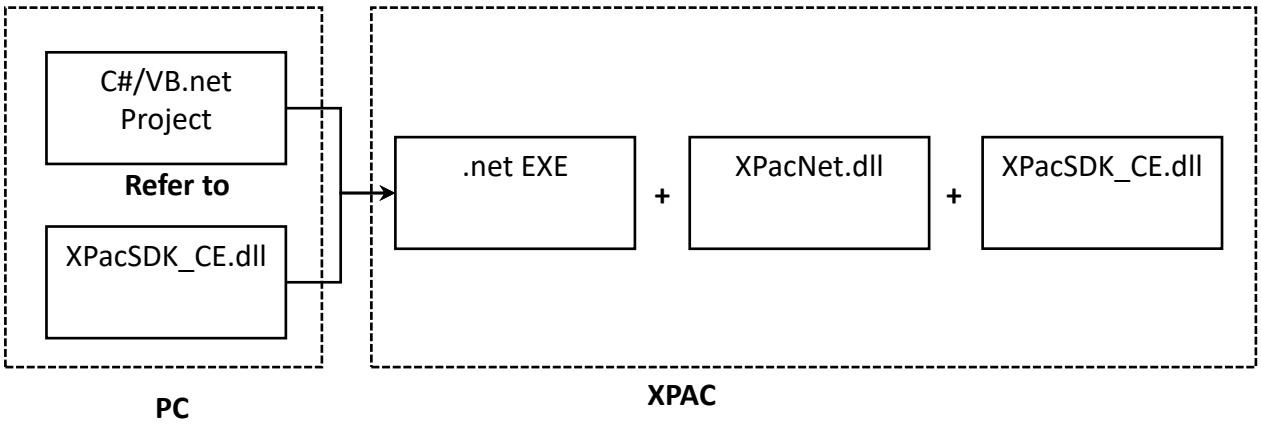
1. 用戶使用的是 XPAC 系列設備，它的程序是基於使用舊的 SDK (XPacSDK_CE.dll/XPacNet.dll) 運作於 XPAC 設備上舊 DCON 模組(注 2)和多功能 DCON 模組(注 1)。

   新的 PACSDK.dll 提供了允許訪問多功能模組的 pac_xxx_MF API 函數，因此必須更新程式碼 (使用新的 PACSDK.dll)，才能在程序中使用。
   用戶的程序可以繼續使用舊程式庫，而無需進行修改。
   (舊的 SDK 將繼續維護(修錯誤)和更新，但會不會增加新的功能)

   使用舊的 SDK 時，參考下面的流程圖：
   VC 項目需要在構建時鏈接 XPacSDK_CE.lib，並且 XPAC 系列設備中構建的可執行文件必須與 XPacSDK_CE.dll 一起使用。

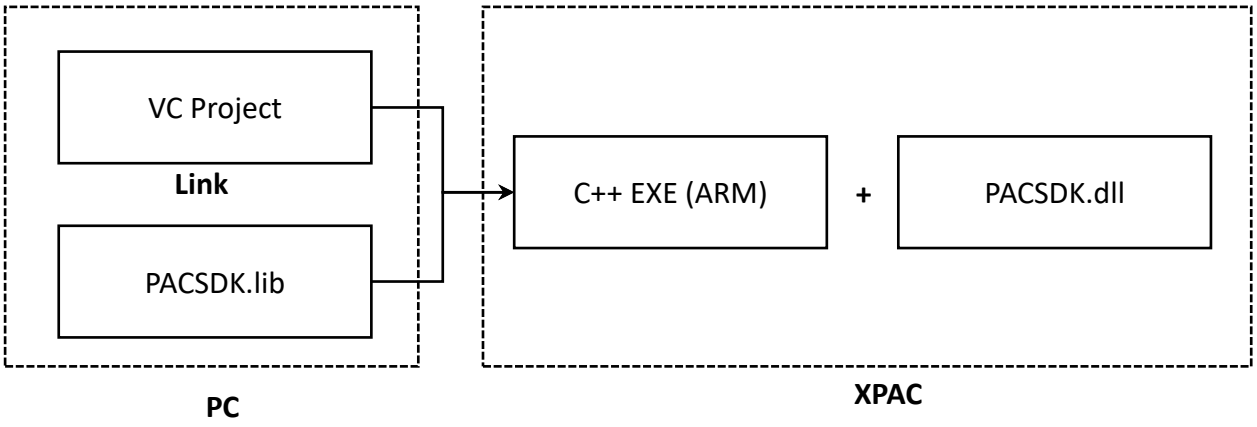


   C#/VB.net 項目在構建時需要參考 XPacNet.dll，並且 XPAC 系列設備中構建的可執行文件，必須與 XPacNet.dll 和 XPacSDK_CE.dll 一起使用。

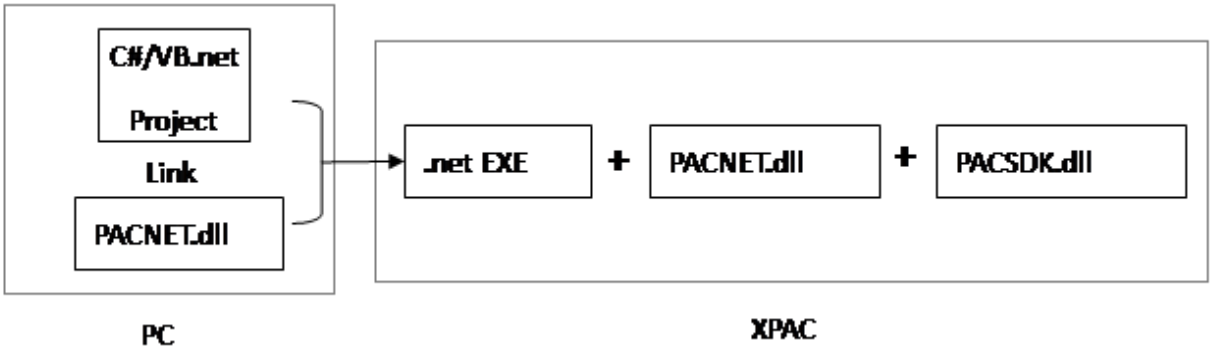


使用新的 SDK 時，參考下面的流程圖：

VC 建立需要鏈接 PACSDK.lib 的項目，XPAC 系列設備中構建的可執行文件必須與 PACSDK.dll 一起使用。



C#/VB.net 項目在構建時需要參考 PACNET.dll，並且 XPAC 系列設備中構建的可執行文件，必須與 PACNET.dll 和 PACSDK.dll 一起使用。



E. 如何升級 WinPACSDK.dll/XPACSDK.dll

與從 WinPacSDK 庫更新 PACSDK 庫的問題和解決方案

請參閱

W6-10_How_to_update_to_PACSDK_library_from_WinPacSDK_library_ tc.pdf

位於

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_tc.pdf

與從 XPacSDK 庫和解決方案更新到 PACSDK 庫問題

請參閱

5233-10_How_to_update_to_PACSDK_library_from_XPacSDK_library_tc.pdf

位於

http://ftp.icpdas.com/pub/cd/xp-8000-ce6/document/faq/sdk/x6-10_how_to_update_to_pacsdk_library_from_xpacsdk_library_tc.pdf

F. COM port 及 slot 定義之比較表

每個 PAC 控制器有它對應的通信 port 和 slot 數量。因此，應用在程序呼叫 API 函數時，需輸入對正確對應的 slot 數量和 COM port 號碼。定義如下表

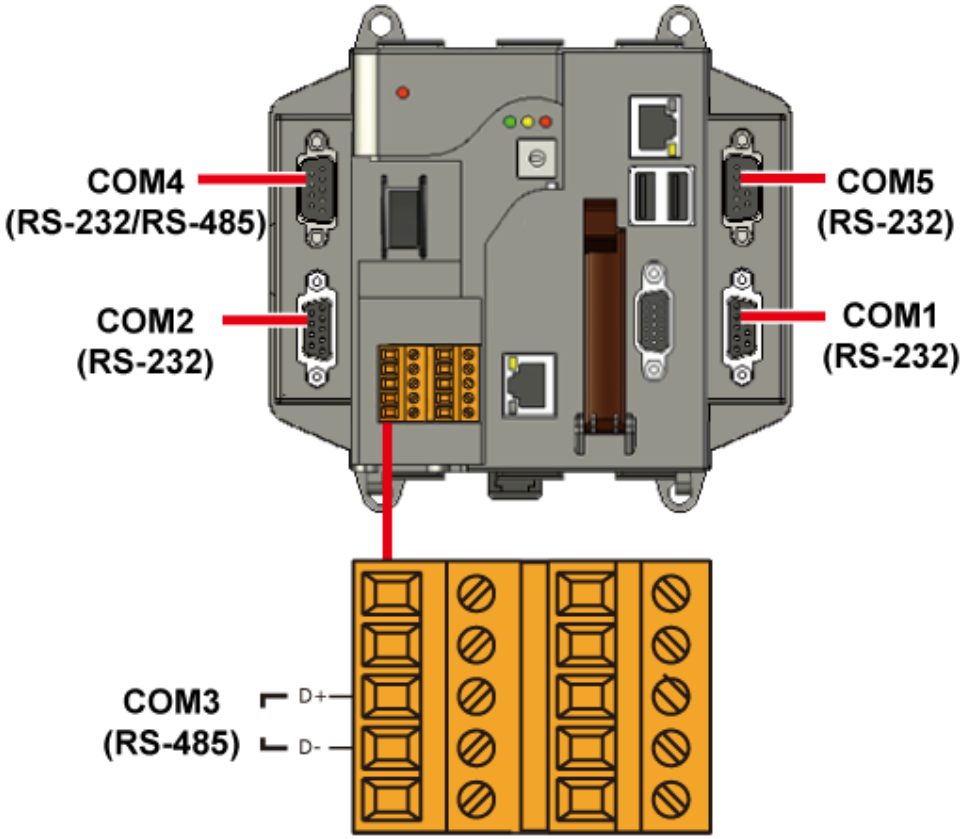
	Slot number (Slot 定義)	COM0	COM1	COM2	COM3	COM4	COM5
XP-8131-CE6	1 (1)	-	底板	RS-232	RS-485	RS-232/RS-485	RS-232
XP-8331-CE6	3 (1 ~ 3)						
XP-8731-CE6	7 (1 ~ 7)						
XP-8041-CE6	0	-	RS-232	RS-232	RS-485	RS-232/RS-485	RS-232
XP-8341-CE6	3 (1 ~ 3)		底板				
XP-8741-CE6	7 (1 ~ 7)						
XP-8141-Atom-CE6	1 (1)	-	底板	RS-232	RS-485	RS-232/RS-485	RS-232
XP-8341-Atom-CE6	3 (1 ~ 3)						
XP-8741-Atom-CE6	7 (1 ~ 7)						
WP-9221-CE7	2(0 ~ 1)	底板	RS-232/RS-485	RS-485	RS-232/RS-485	RS-232	-
WP-9421-CE7	4 (0 ~ 3)						
WP-9821-CE7	8 (0 ~ 7)						
WP-8121-CE7	1 (0)	底板	RS-232	RS-485	-	-	-
WP-8421-CE7	4 (0 ~ 3)				RS-232/RS-485	RS-232	

WP-8821-CE7	8 (0 ~ 7)						
	Slot number (Slot 定義)	**COM0**	**COM1**	**COM2**	**COM3**	**COM4**	**COM5**
WP-8131	1 (0)	底板	RS-232	RS-485	-	-	-
WP-8431	4 (0 ~ 3)				RS-232/RS-485	RS-232	
WP-8831	8 (0 ~ 7)						
WP-8141	1 (0)	底板	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)	底板	-	RS-485	RS-232	-	-
VP-25W1							
VP-4131							
VP-1231-CE7	3 (0 ~ 2)	底板	-	RS-485	RS-232	-	-

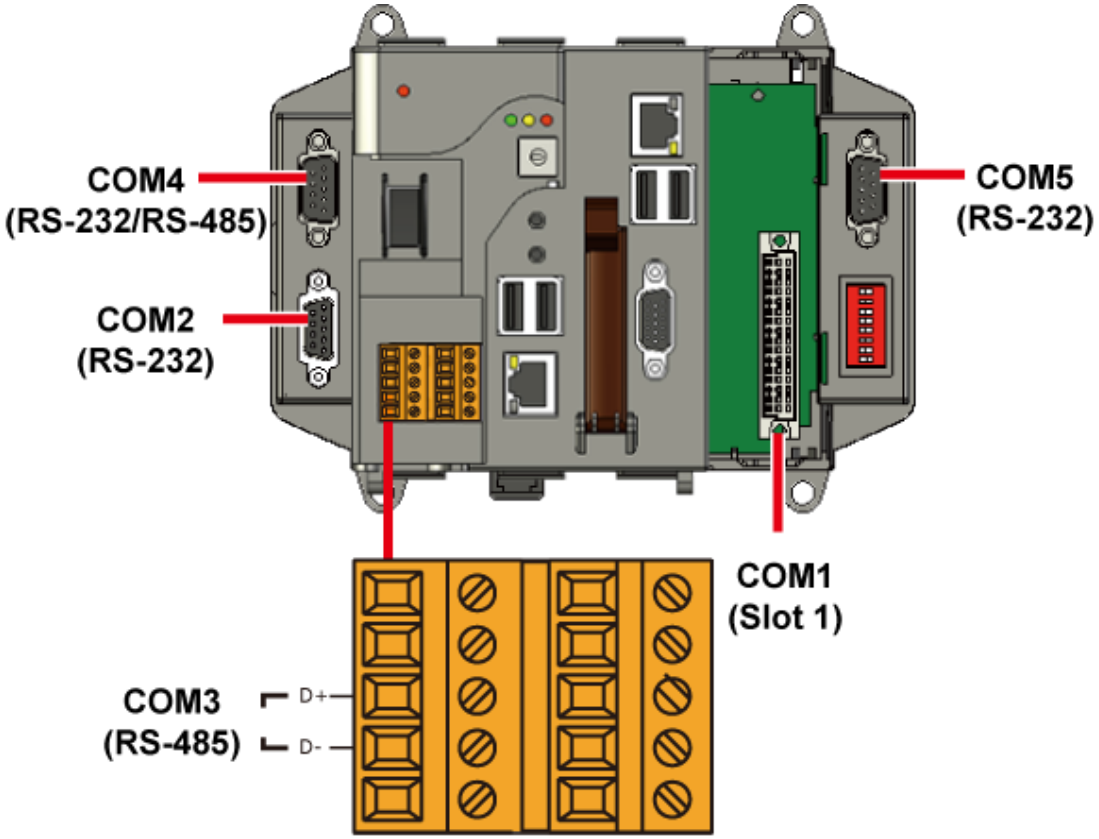
VP-4231-CE7							
VP-6231-CE7							

底板：這是僅用於訪問 I-87K 模組的 RS232 接口。

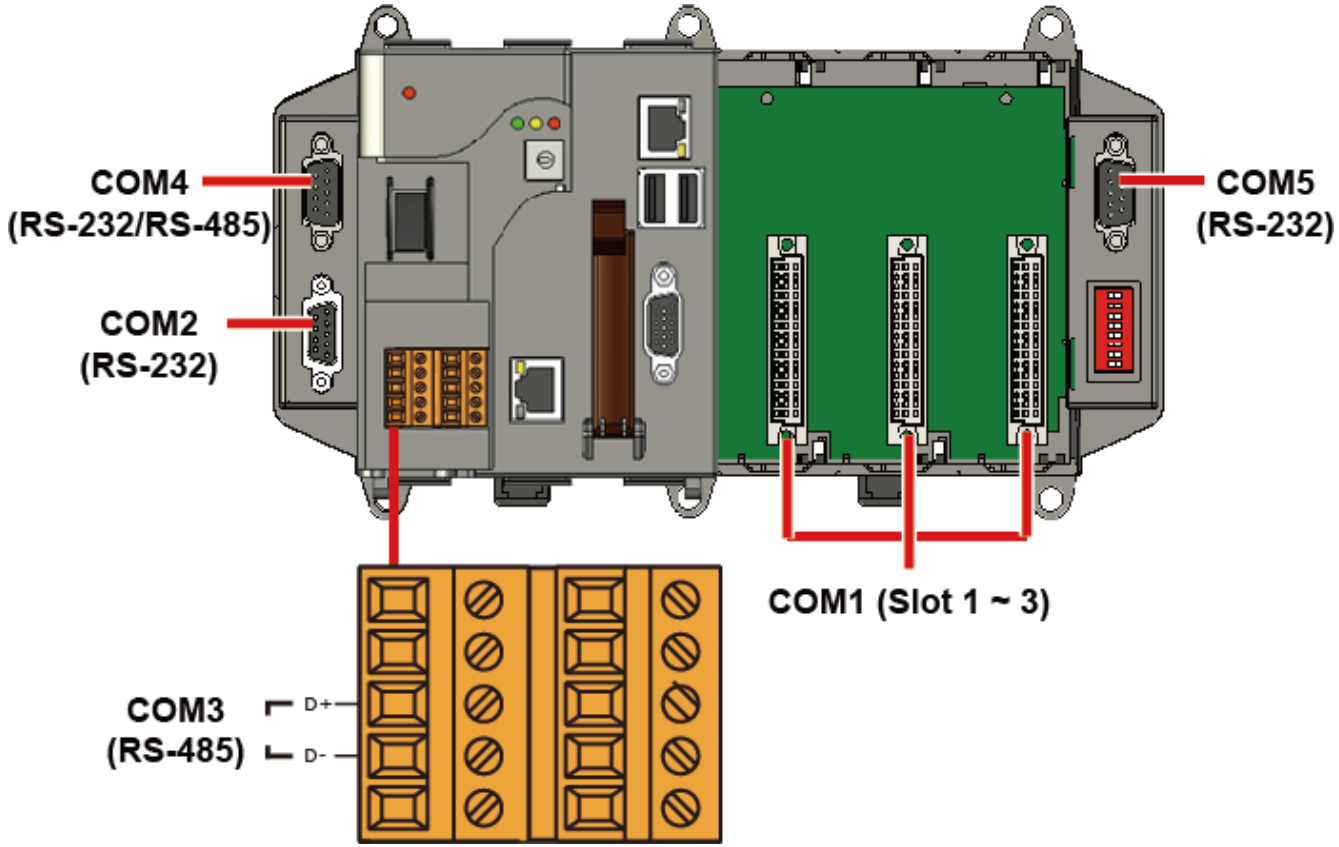
F.1. XP-8041-CE6



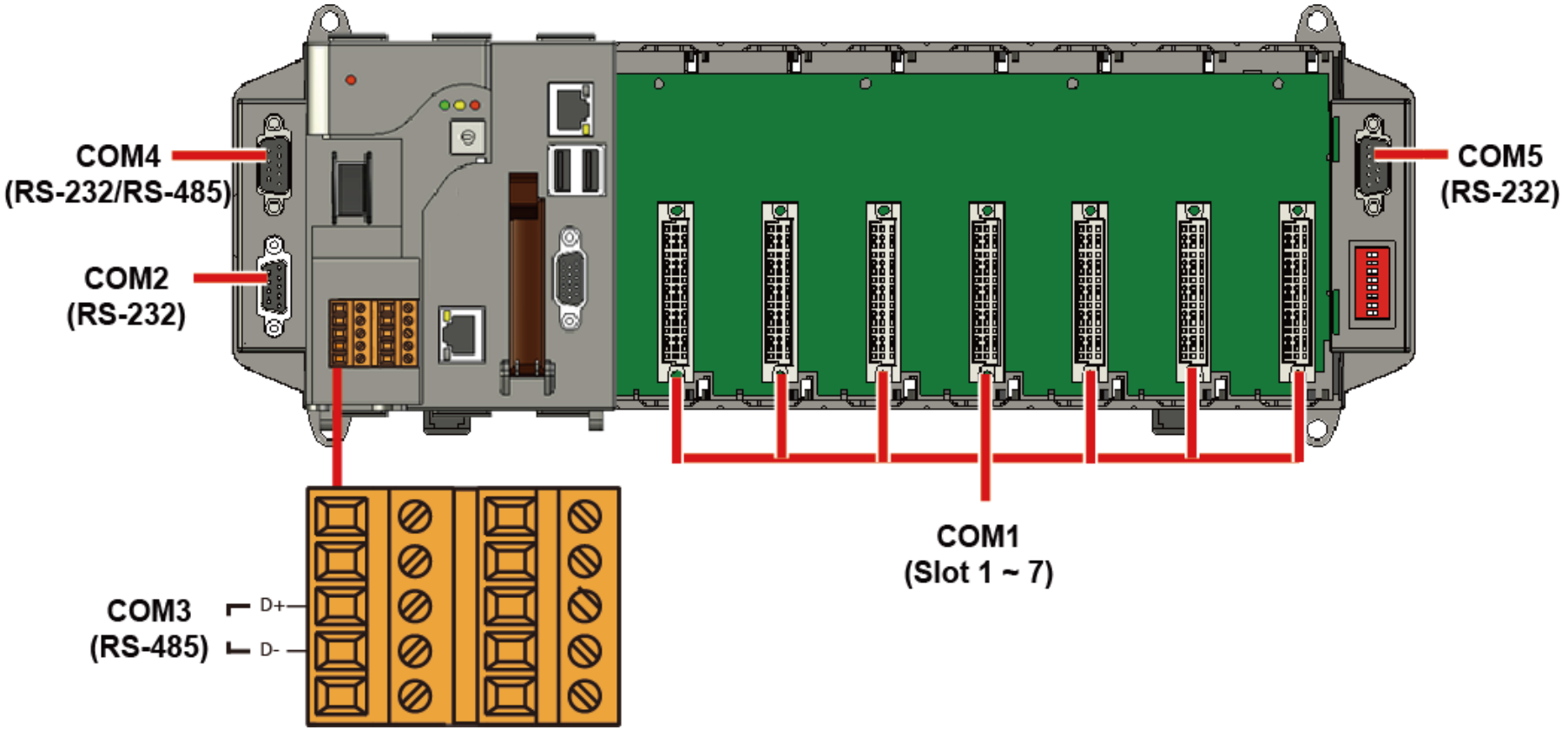
F.2. XP-8131-CE6/XP-8141-Atom-CE6



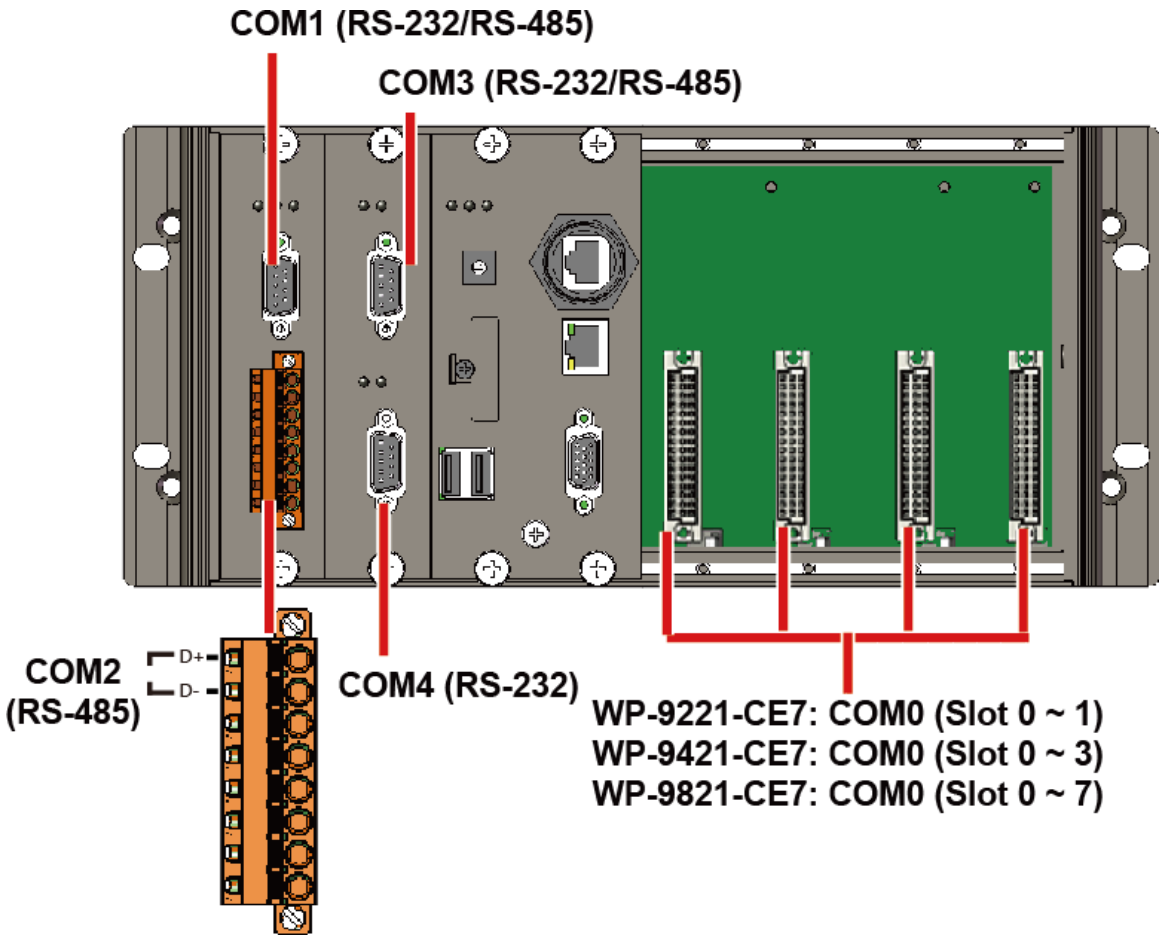
F.3. XP-8331-CE6/XP-8341-CE6/XP-8341-Atom-CE6



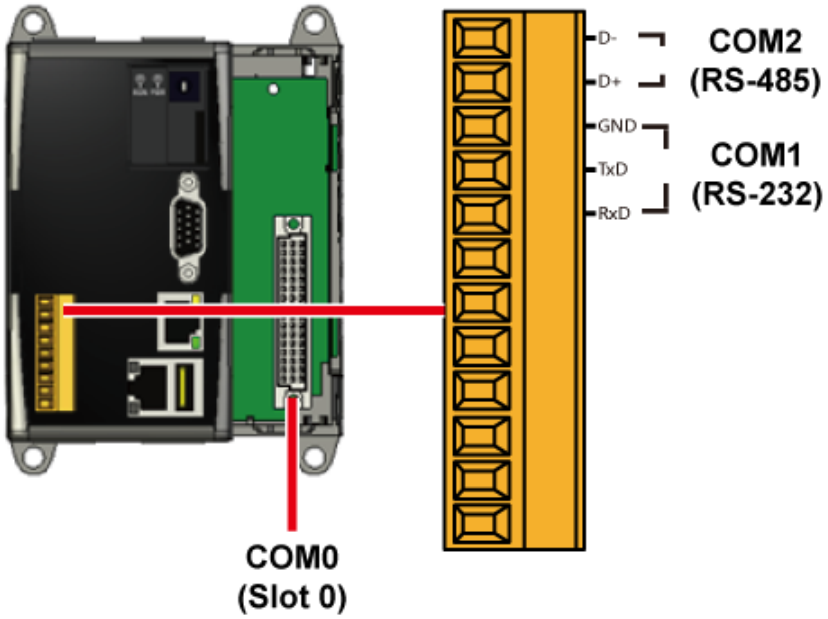
F.4. XP-8731-CE6/XP-8741-CE6/XP-8741-Atom-CE6



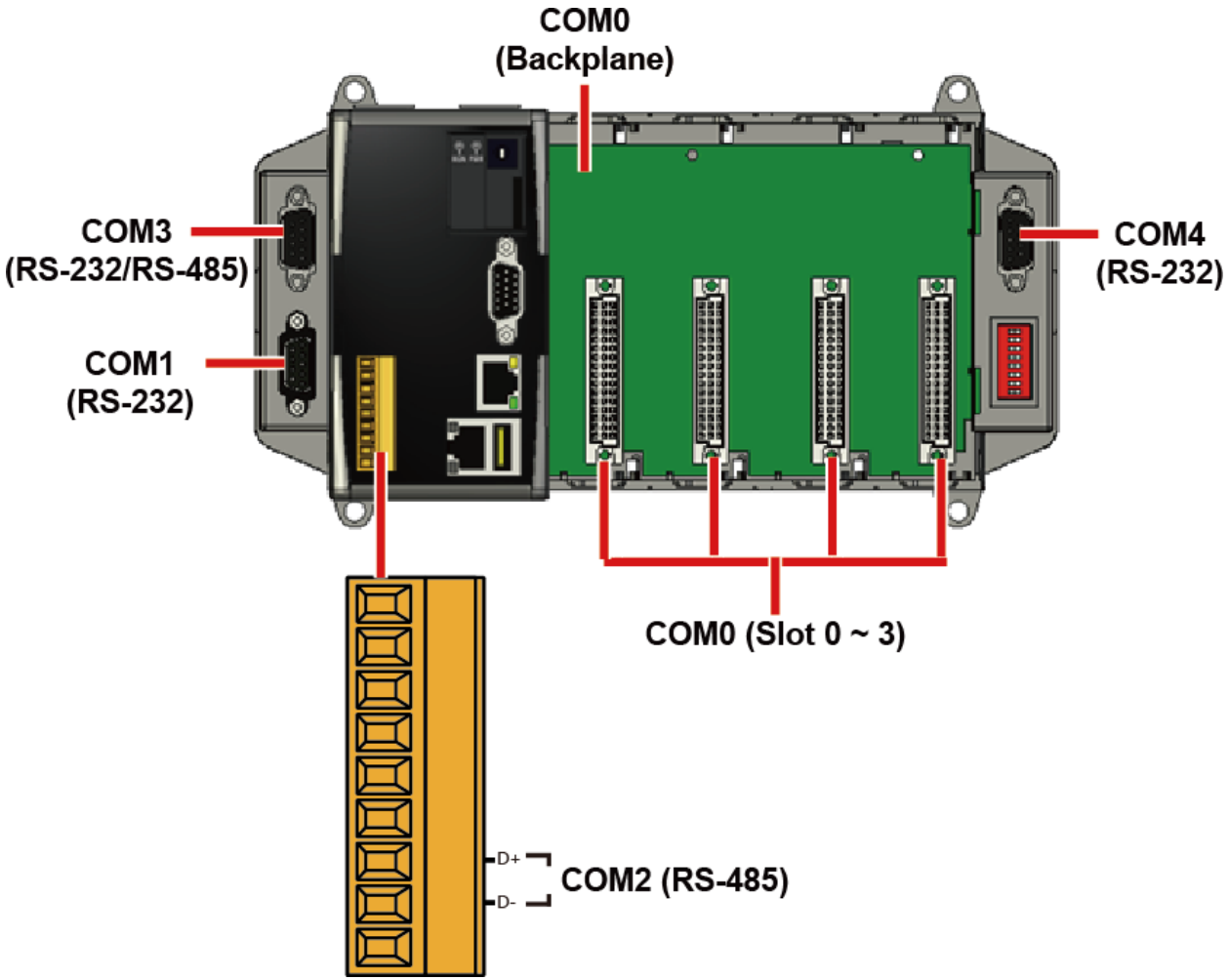
F.5. WP-9221-CE7/WP-9421-CE7/WP-9821-CE7



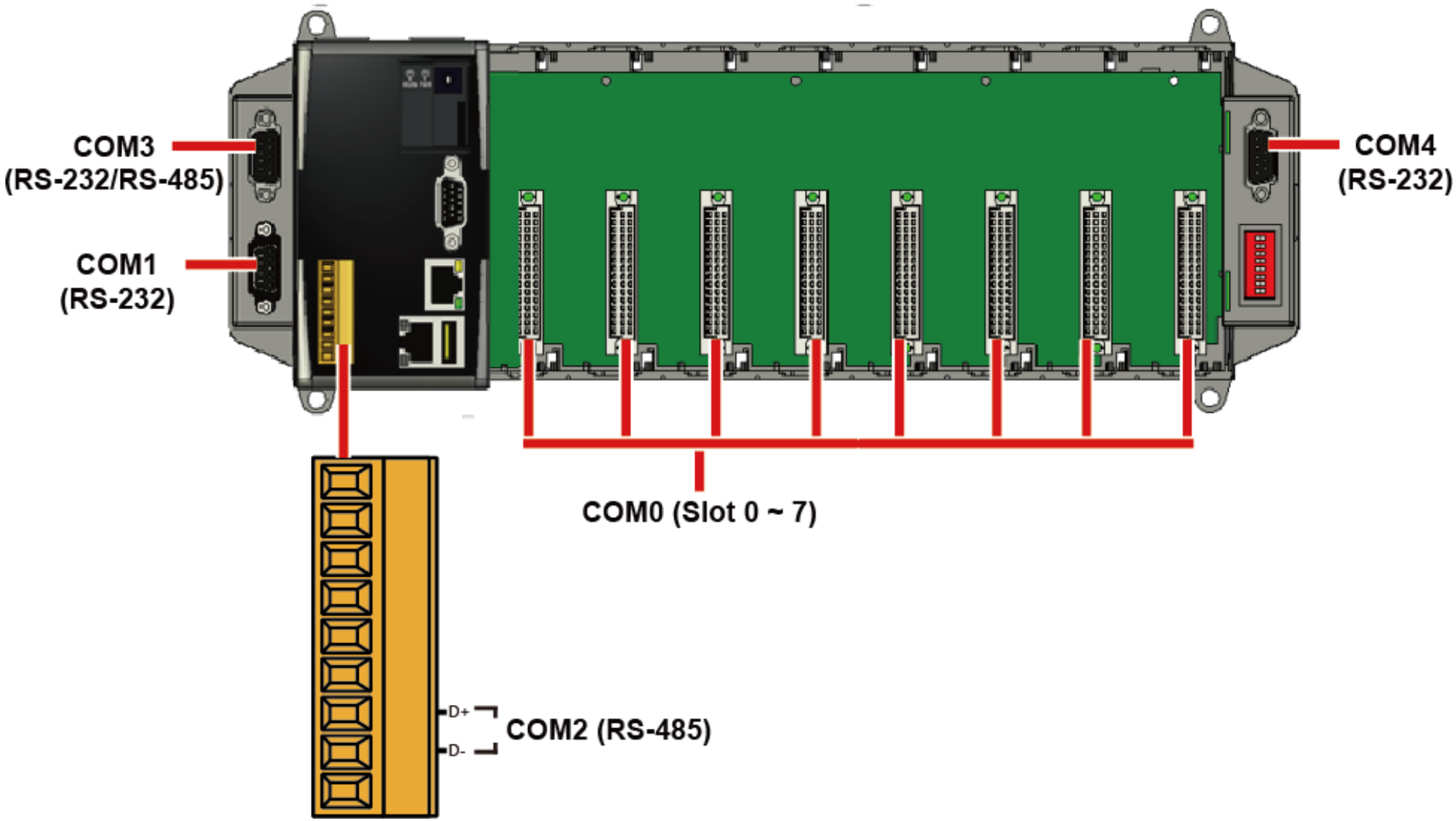
F.6. WP-8121-CE7/WP-8131/WP-8141



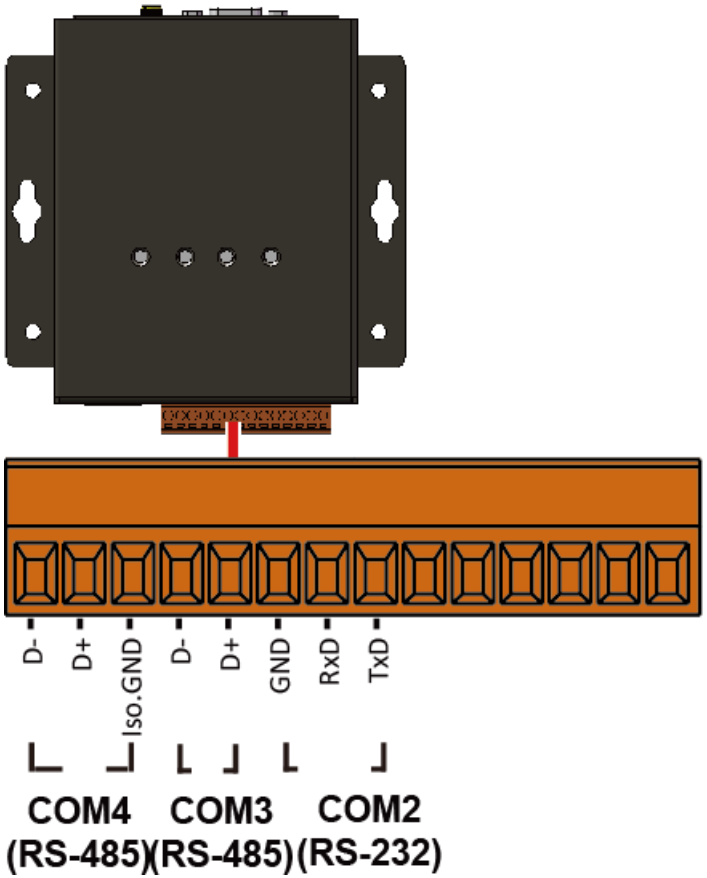
F.7. WP-8421-CE7/WP-8431/WP-8441



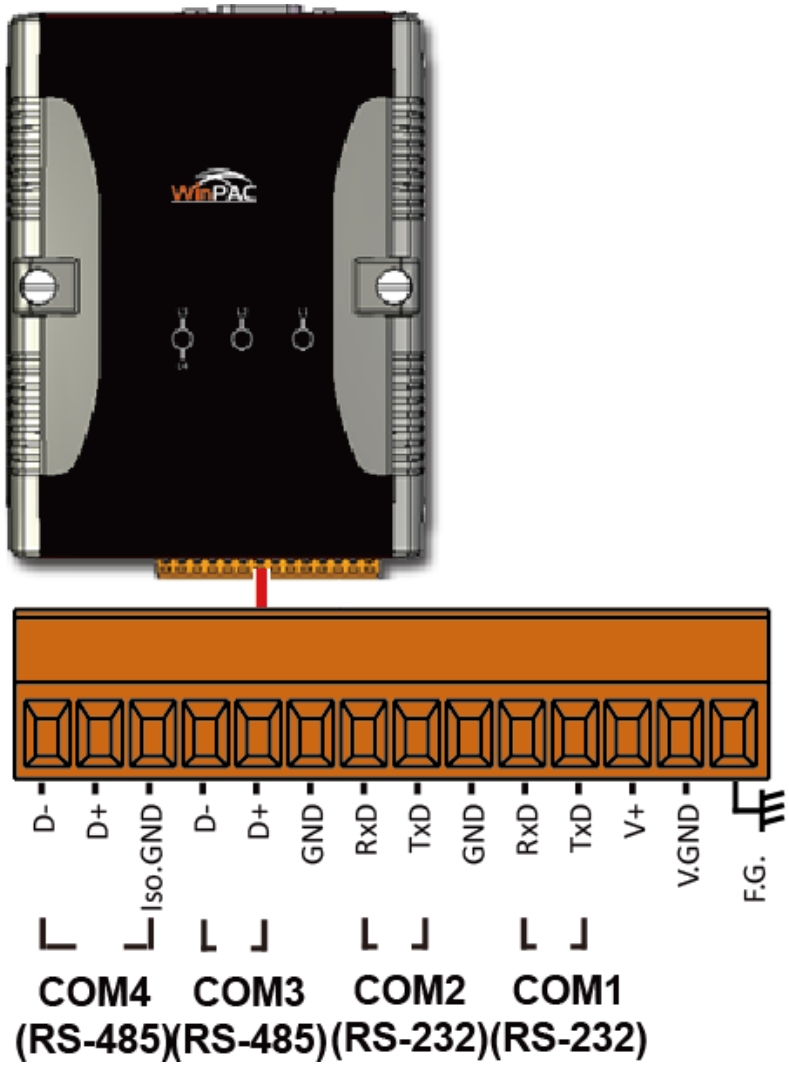
F.8. WP-8821-CE7/WP-8831/WP-8841



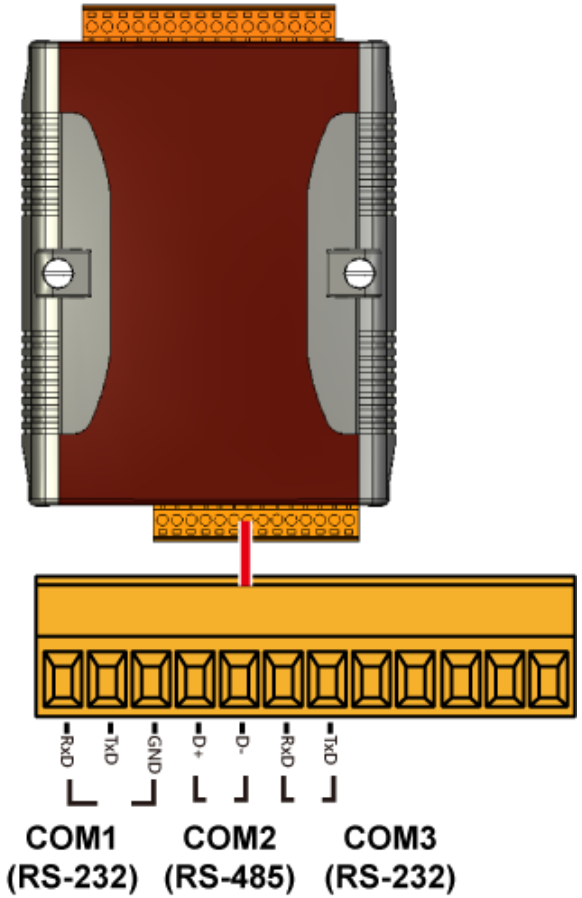
F.9. WP-5231M-CE7/WP-5231PM-3GWA-CE7/WP-5231PM-4GE-CE7/WP-5231PM-4GC-CE7



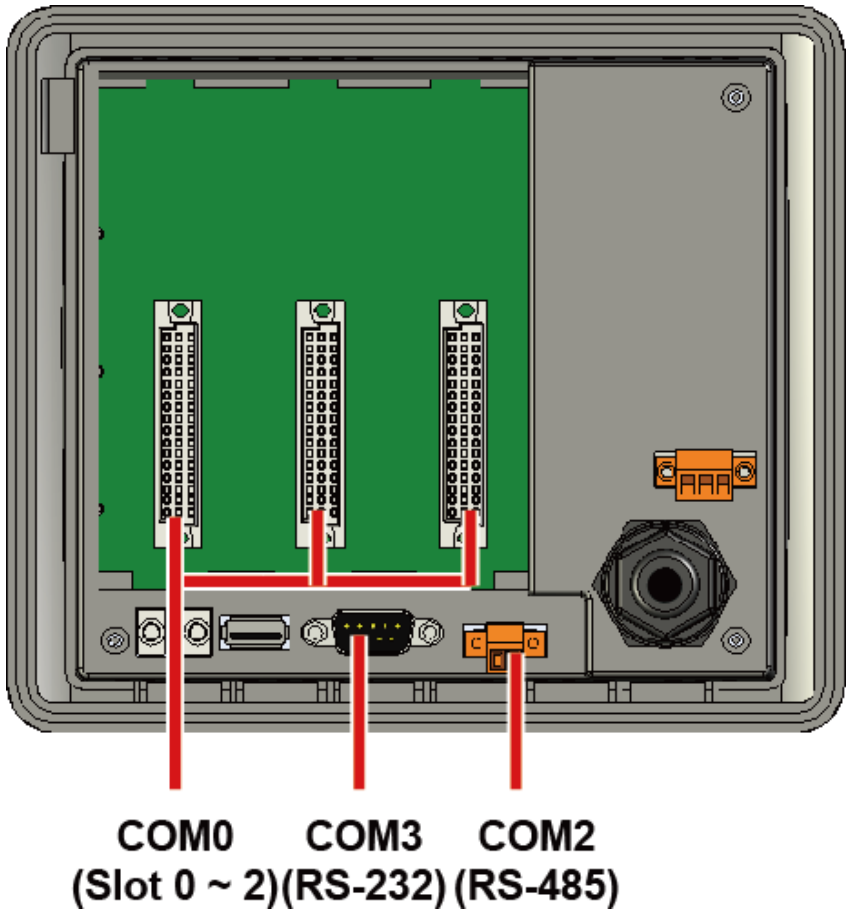
F.10. WP-5231-CE7



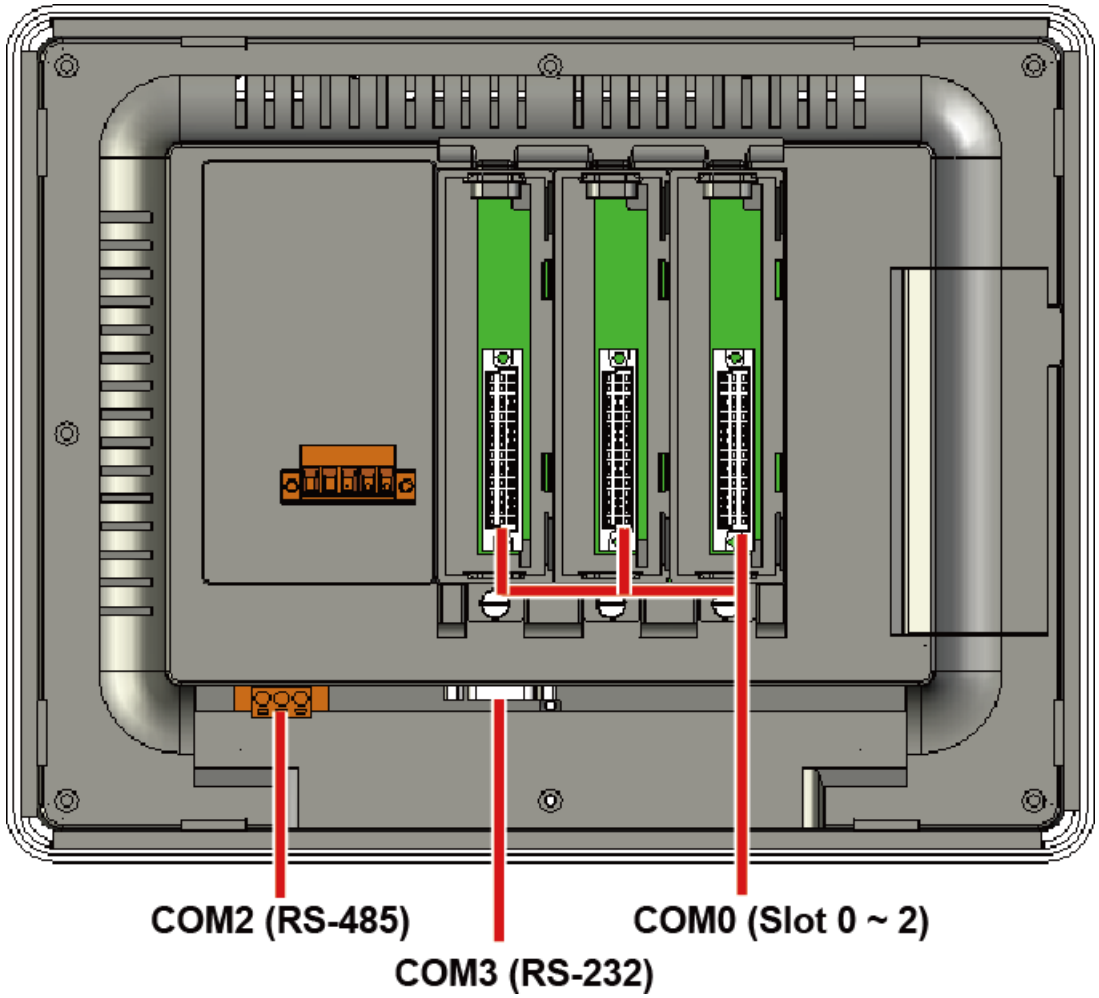
F.11. WP-5141/WP-5141-OD/WP-5151/WP-5151OD



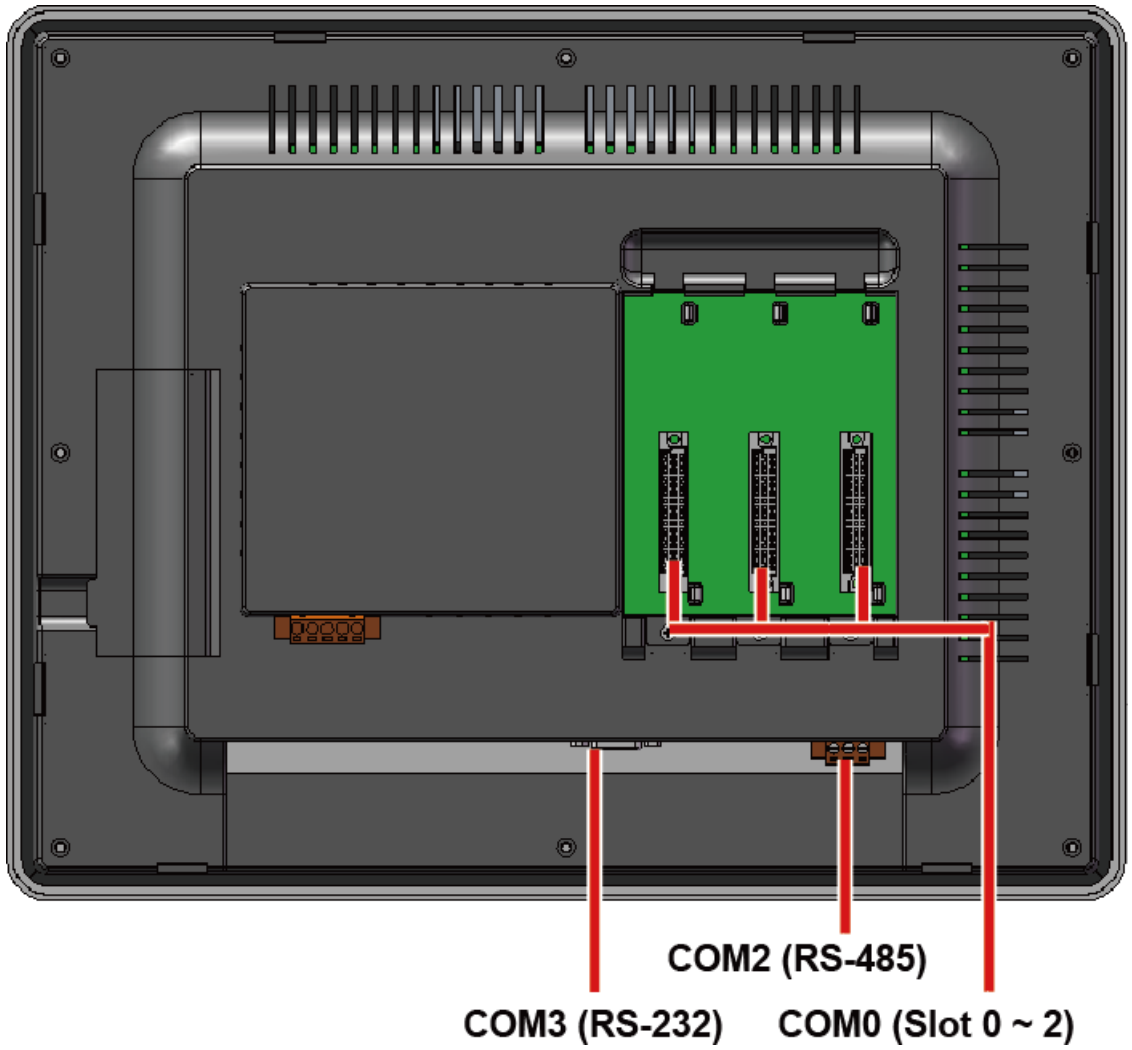
F.12. VP-1231-CE7



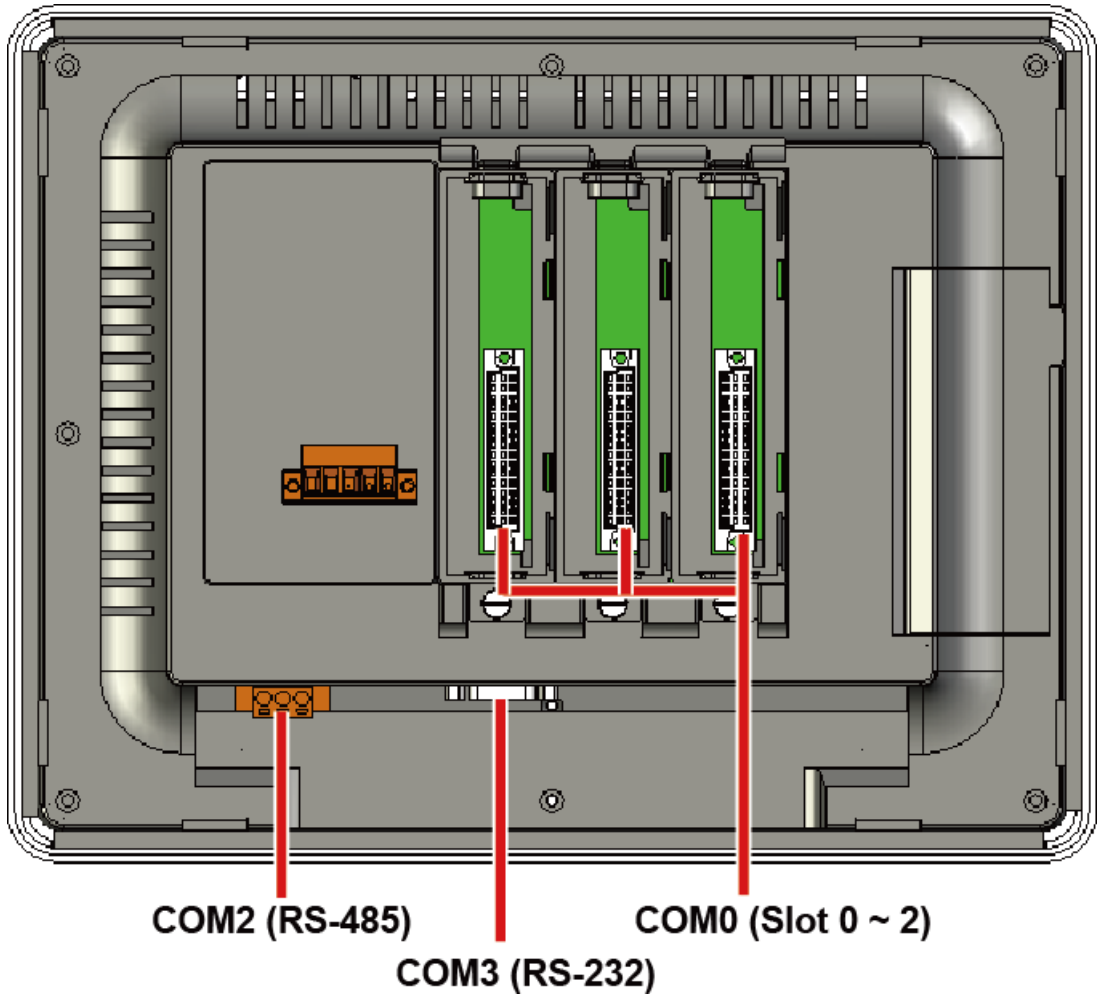
F.13. VP-4231-CE7



F.14. VP-6231-CE7



F.15. VP-4131



F.16. VP-23W1/VP-25W1