I-97K I/O Module
Common User Manual

Version 1.0.3 April 2023

Written by Sean

Edited by Anna Huang

Warranty

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

Warning

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

Copyright

Trademarks

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

Contact Us

If you have any problems, please feel free to contact us.

You can count on us for a quick response.

Email: service@icpdas.com

Table of Contents

1. Introduction

The I-97K series module are based on the serial interface that can be used to expand the I/O capability function of the 9000 series PAC.

Supported PACs

This manual is intended to support the development of I-97K series I/O modules for the following PACs.

Platform	CPU	Slot
WP-9x2x-CE7	AM335x (ARM)	2,4,8
XP-9x7x-WES7	E3827/E3845 (X86)	1,3,7

Compared with I-9K I/O Modules

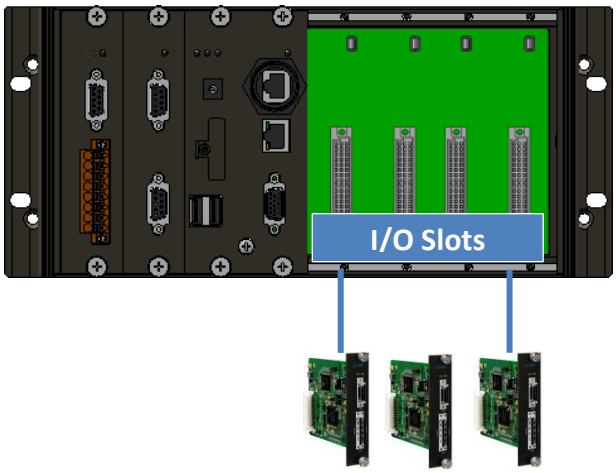
The 9000 series PAC has the I/O expansion slot that supports both I-9K and I-97K series modules. Below is a comparison table of the features between the I-9K and I-97K series modules

Model	I-9K Series	I-97K Series
Communication interface	Parallel bus	Serial bus
Protocol	-	DCON
Communication speed	Fast	Slow
DI with latched function	-	Y
DI with counter input	-	Y (100 Hz)
Power on value	Y	Y
Safe value	Y	Y
Programmable slew-rate for AO module	-	Y

I-97K series modules consist of the following modules:

- I-97015: 8-Channel RTD Input Module
- I-97017Z: 10/20 Channel Analog Input with High Voltage Protection
- I-97018/S: 8-Channel Thermocouple Input Module
- I-97019/S: 8-Channel Thermocouple Input Module
- I-97024U: 4-Channel Isolated Voltage/Current Output Module
- I-97028U: 8-Channel Isolated Voltage/Current Output Module

For more information about each module, please visit the I-9K/I-97K selection guide and click on the module name you want to learn more about at the following link:
https://www.icpdas.com/tw/product/guide+Remote_I_O_Module_and_Unit+PAC_I_O_Modules+I-9K_I-97K_Series#1662



I-97K series I/O modules

1.1. Features

The basic features of I-97K I/O module are given as following:

1.1.1. Watchdog & Safe Value

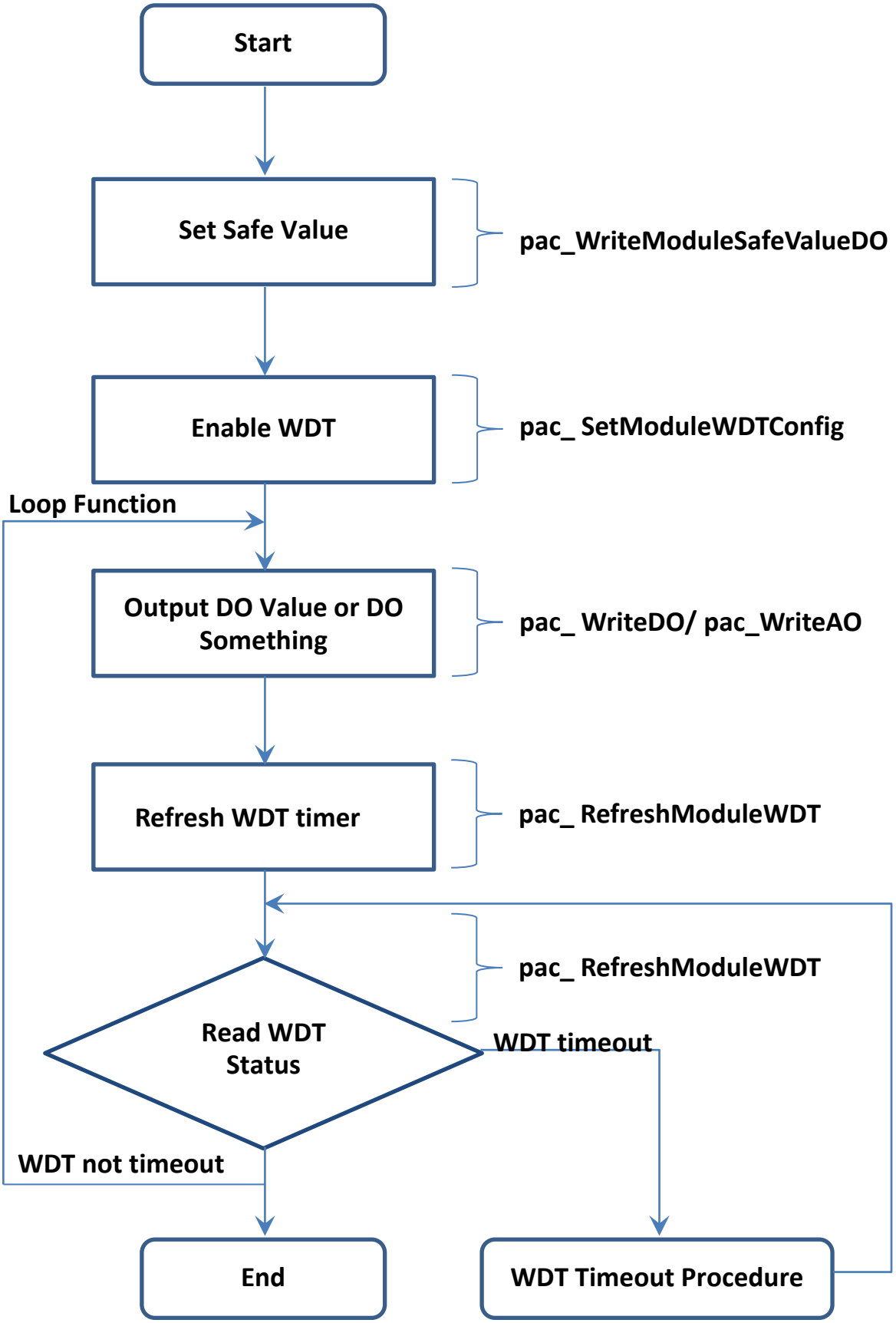
I-97K DO/AO module equip a Hardware Watchdog (WDT) that monitors the operating status of the module. Its purpose is to prevent problems due to host malfunctions, such as situations where the device is affected by noise, or the program is not stable, etc. When the “refresh WDT” function fails and a Watchdog timeout occurs, all output values on the module will be set to the Safe Value state in order to prevent the controlled target from performing any erroneous operations.

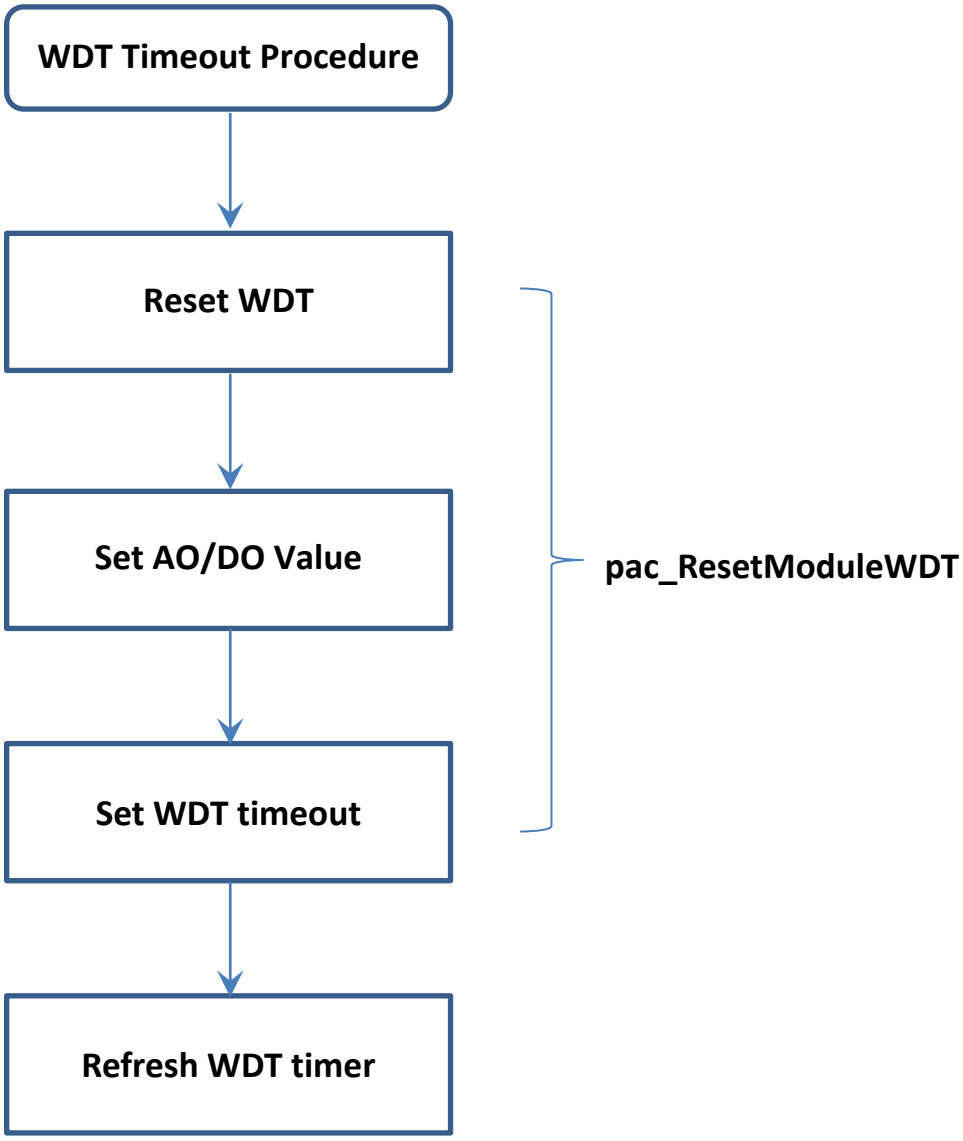
When Watchdog timeout occurs, I-97K output values will all enter Safe Values, the operation for writing output value will not accept until reset Watchdog (Call pac_ResetModuleWDT).

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

When module is reset, the Watchdog status will be disabled, user need to set it again.

I-97K DO/AO Watchdog procedure





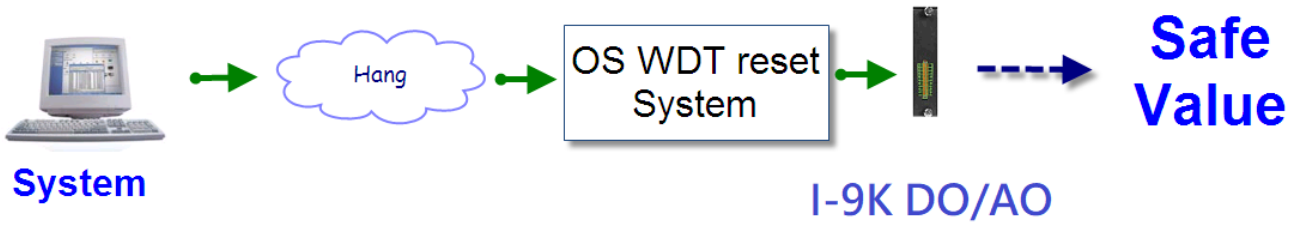
1.1.2. DO/AO module usage scenarios

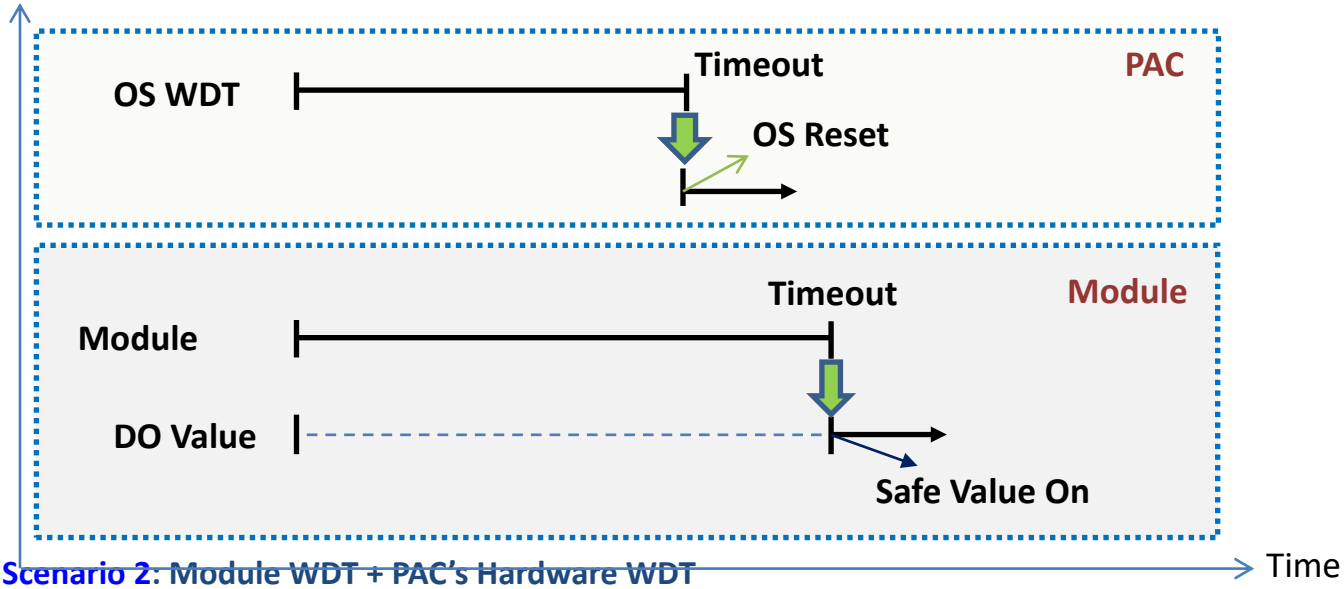
The 9000 PAC series supports two watchdogs for monitoring system operation. They are both built-in hardware circuits that will reset the system if there is a hardware or software failure.

Operation	I/O module status
Hardware WDT Reset on 9000 PAC	I/O will enter Power-on Mode
OS WDT Reset on 9000 PAC	I/O will enter Safe Value
Power Reset on 9000 PAC	I/O will enter Power-on Mode

Scenario 1: Module WDT + PAC's OS WDT

If the PAC system encounters an event that causes it to become unresponsive for any reason, the I-97K DO/AO WDT and the OS WDT provided by 9000 PAC series can be used in combination, thereby preventing the system from hang becoming unresponsive, which may cause the Digital Output to be uncontrolled and result in damage to the device. The OS WDT on the 9000 PAC series will reset the PAC to solve the unresponsiveness problem, and then the Safe Value will be set for the I-97K DO/AO module to prevent the DO/AO from becoming uncontrolled.

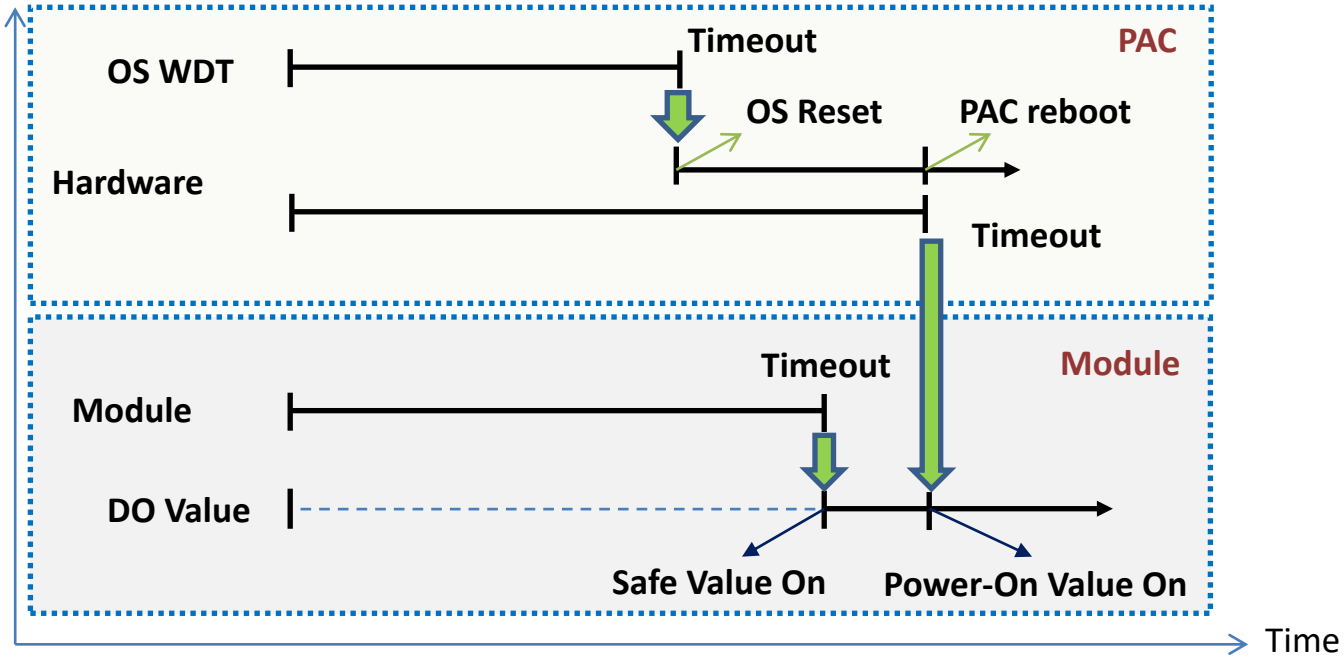




Scenario 2: Module WDT + PAC's Hardware WDT

PAC is equipped with dual watchdog. Hardware WDT is another electronic timer which also built-in hardware circuit to monitor the operation of the system and will reset the system if a failure occurs in the hardware or the software. The difference between OS WDT and Hardware WDT is that the backplane of PAC will be reset while Hardware WDT timeout occurs. All module plugged on the slot of the backplane will be also reset and the power-on value will be output for I-97K DO/AO module.

In practical scenarios, the Hardware WDT timeout value is set longer than OS WDT to prevent the OS WDT still fails to start. The Hardware WDT acts as PAC's second line of defense.



User can use pac_EnableWatchDog (int wdt, DWORD value) function of PACSDK library to set Hardware WDT or software WDT enabled/disabled on 9000 PAC series.

Wdt =0; for OS WDT (**PAC_WDT_OS**) enable.

Wdt =1; for Hardware WDT **(PAC_WDT_HW**) enable.

PAC_WDT_HW and **PAC_WDT_OS** are different definitions of the names, both watchdogs with electronic timer, monitor module and hardware reset circuit

1.1.3. AO Module

The I-97K AO module offers the various analog output channels (4/8/…etcs), each of which features photo-couple isolation, supports sink-type /source-type output using an open collector or relay output. . The basic features of I-97K AO module are given as following:

- 4 kV ESD protection
- 3000 VDC isolated analog output.
- Programmable PowerOn Value of analog output.
- Programmable slew rate.
- Software calibration.

The module's output have 3 different conditions :

1. **Safe Value**. If the host watchdog timeout is set, the output is set to Safe Value. While the module receive the output command, #AA(Data) or #AAN(Data) or call pac_WriteAO () function and will not change the output to the output command value. The host watchdog timeout status is set and stored while the host watchdog timeout interval expired, and only can be cleared by DCON command ~AA1 or call pac_ResetModuleWDT() function. If user want to change the output, need to clear the host watchdog timeout status first, and send output command to change the output to desired values.
2. **PowerOn Value**. Only the module reset, and the host watchdog timeout status is clear, the module's output is set to predefined PowerOn Value.
3. **Output Command Value**. If the host watchdog timeout status is clear, the user send command, #AA(Data) or #AAN(Data) or call pac_WriteAO () function to change the output value of module.

Slew Rate Control

Slew rate control is to adjust the output slope. Most analog output change is instantaneous. In many applications this characteristic is undesirable and a gradual controlled output slew rate is more appropriate.

The I-97K AO module allows programmable slew rate control. While the output command is sent to the module to change the analog value, the output will automatically slope to the new value at the specified slew rate. The I-97K AO module update the analog output value and the output is smoothly stepped until the final output value is reached.

The Slew rate is modified by sending DCON command or using DCON utility.

Current Readback

The I-97K AO module have the analog-to-digit converter to monitor the current output signal. The current readback may find the fault of improper wiring or loads while the readback value is far from the output value.

1.1.4. AI Module

The I-97K AI module offers the various Analog Input channels (8/10/20…etcs), each of which features photo-couple isolation, supports sink-type /source-type output using an open collector or relay output.

The I-97K AI module includes LED indicators that can be used to monitor the status of the Analog Input channels. 4 kV ESD protection and 3750 Vrms intra-module isolation is provided as standard.

The I-97K AI module integrates overcurrent, overvoltage and short-circuit functions.

For more information about each I-97k AI module, please visit the I-9K/I-97K selection guide and click on the module name you want to learn more about at the following link:

https://www.icpdas.com/tw/product/guide+Remote_I_O_Module_and_Unit+PAC_I_O_Modules+I-9K_I-97K_Series#1662

1.1.5. DO Module

The I-97K DO module offers the various digital output channels (8/16/32…etcs), each of which features photo-couple isolation, supports sink-type /source-type output using an open collector or relay output.

The I-97K DO module includes LED indicators that can be used to monitor the status of the Digital Output channels. 4 kV ESD protection and 3750 Vrms intra-module isolation is provided as standard.

The I-97K DO module integrates overcurrent, overvoltage and short-circuit functions.

The digital outputs of I-97K DO module can be set by two other conditions.

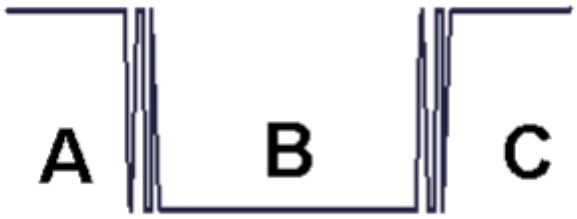
1. **Safe Value**. When the host watchdog is enabled and a host watchdog timeout expired, the “safe value” is loaded into the digital output ports. The set digital output commands have no effect on the digital output ports until the host watchdog timeout status is cleared. The host watchdog timeout status is set and stored. The status is not changed even after power-on reset. It can be cleared by the reset host watchdog timeout status command ~AA1 or call ***pac_ResetModuleWDT()*** function. See Chapter 2. Watchdog & Safe Value for more details.
2. **Power On Value**. When the module is powered on and the host watchdog timeout status is cleared, the “power-on value” is loaded into the digital output ports. If the host watchdog timeout status is not cleared on power-on, then the safe value is loaded into the digital output ports. Both the safe value and power-on value are set by the ~AA5V or call ***pac_WriteModulePowerOnValueDO()*** function.

1.1.6. DI Module

The I-97k DI module offers the various digital Input channels (8/16/32…etcs), each channel features photo-couple isolation and can be eithers sink-type /source-type input, selectable by wiring.

The I-97K DI module includes LED indicators are provided for monitoring DI channel status, together with 4 kV ESD protection and 3750 VDC intra-module isolation.

The I-97K DI modules provide commands to read the latched high digital input and latched low digital input status. Following is an Example to show the usefulness of the latched digital input. When we want to read the key stroke of a key switch which is connected to the digital input channel of a module, the input signal of the key stroke is a pulse signal as shown in the following figure.

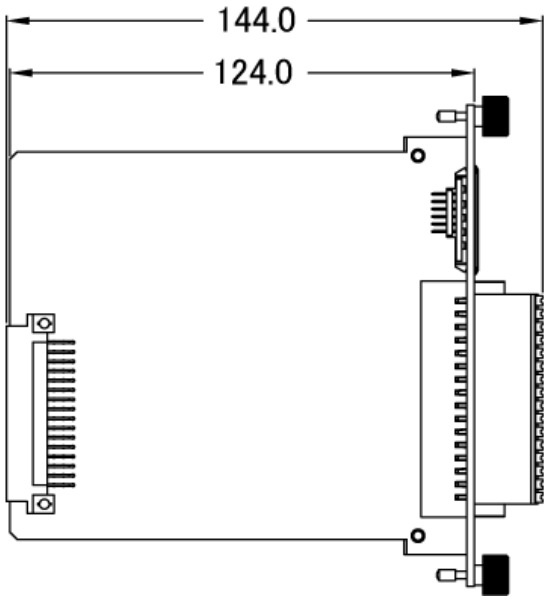


If we just use the read digital input status command to read the signal and we cannot send the command during the B period due to some reasons, then we will lose the key stroke information. However, with the read latched digital input function using ***pac_ReadDILatch()***, we can still get the key stroke information even we are not able to send command in B period. For details of the read latched digital input function. Refer to Chapter 3. API Ferences

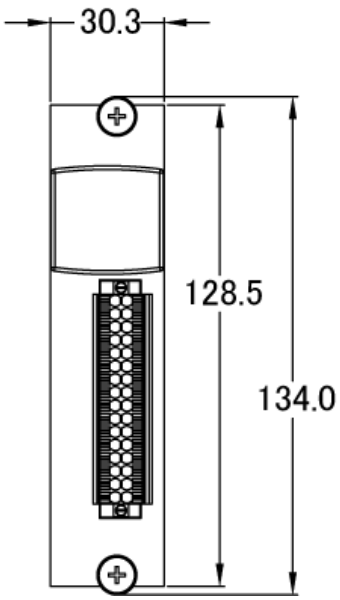
1.2. Dimensions

All dimensions are in millimeters.

I-97K module with Spring clamp terminal connector

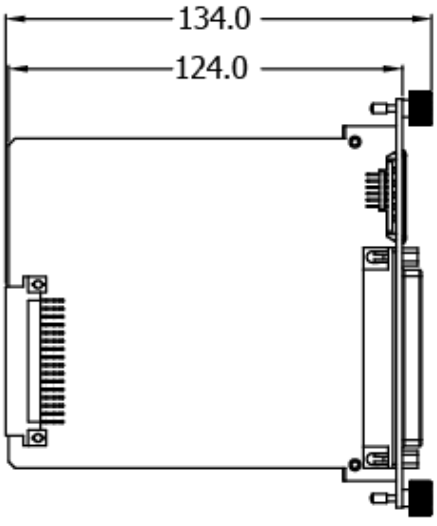


Left Side View

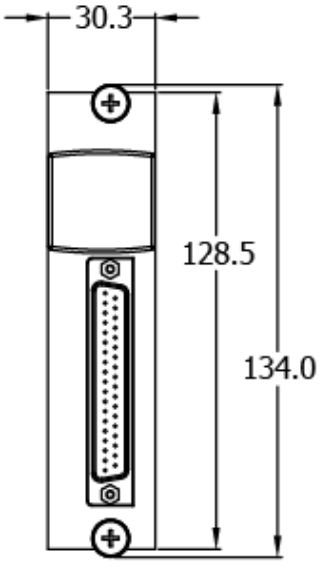


Top View

I-97K module with D-Sub connector



Left Side View



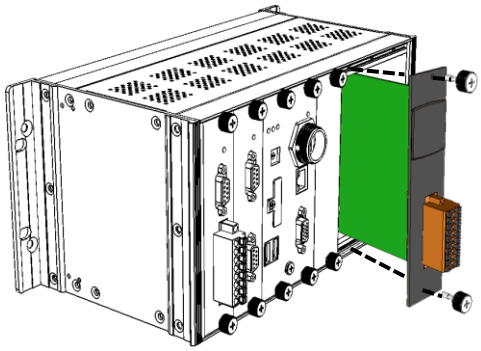
Top View

1.3. Installation

1.3.1. Inserting the I-97K I/O Modules

The I-97k series I/O module can be inserted in the expansion slot of the supported PAC.

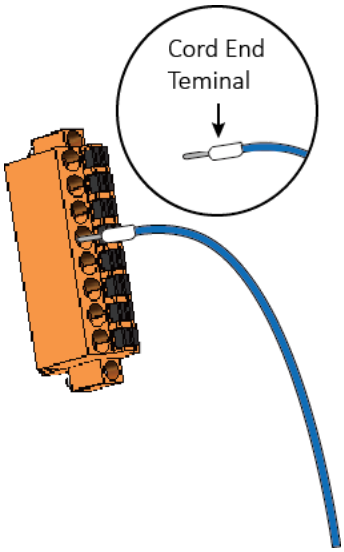
Here are steps on how to install and wiring the I-97k series I/O modules.



Step 1. Insert the I/O module

Step 2. Wiring connection

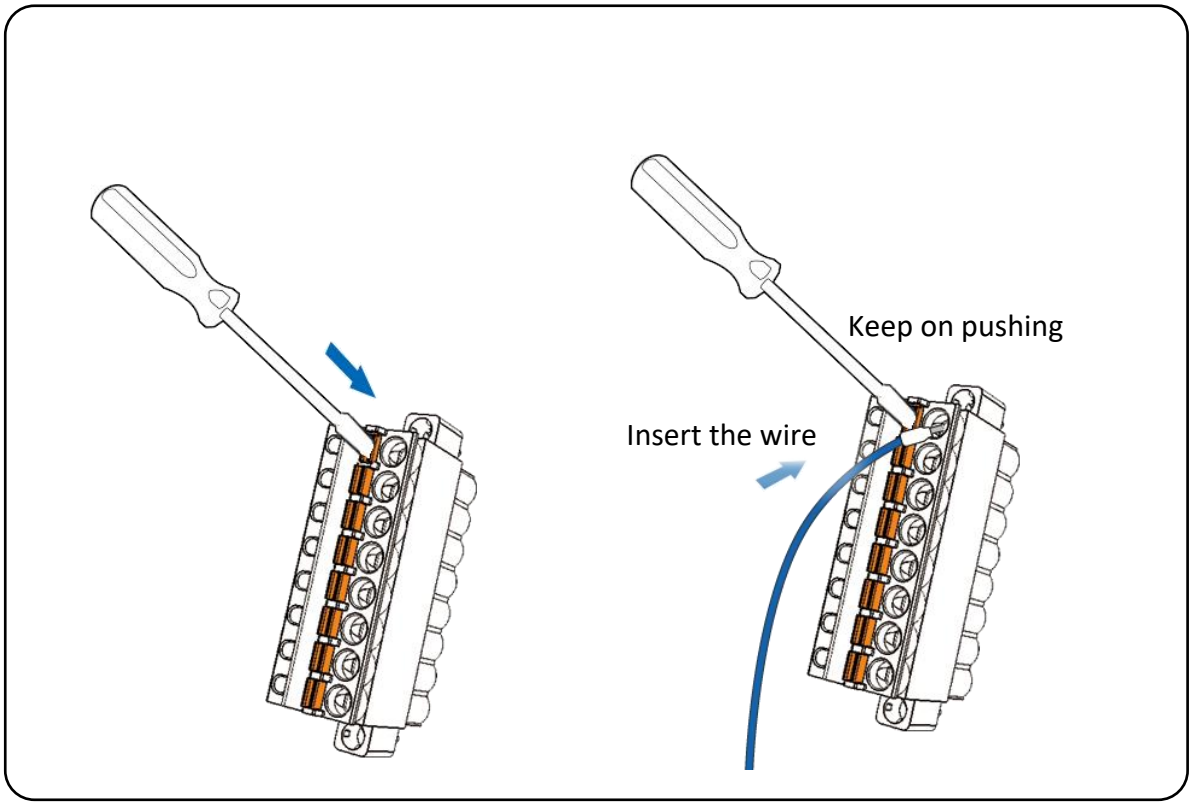
The metal part of the cord end terminal on the wire can be direct wired to the terminal.



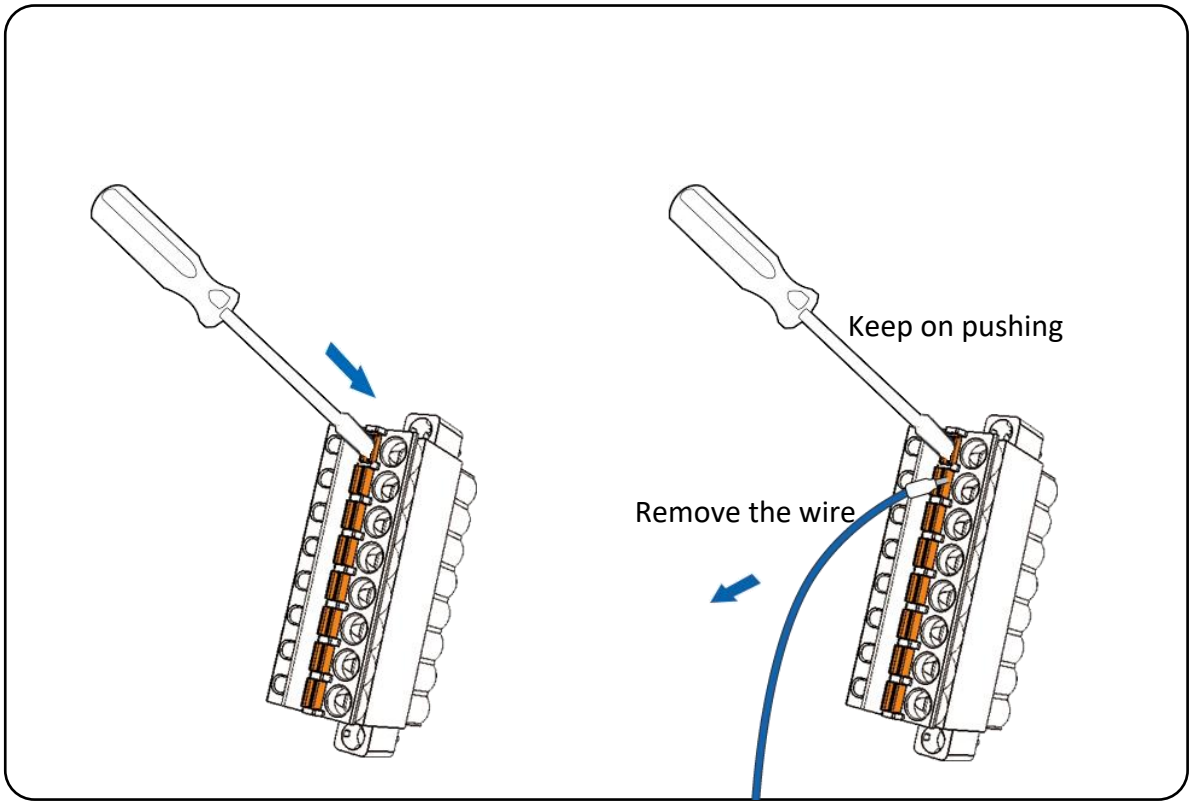
Tips & Warnings

Except I-97018/I-97019 modules, the other I-97K I/O modules support spring clamp terminal connector. The spring clamp terminal connector for the I-97K I/O module connector offers the advantages (anti-vibration, stable clamping and installation easier) relative to screw terminals.

A tip on how to connect the wiring to the connector



A tip on how to remove the wiring from the connector



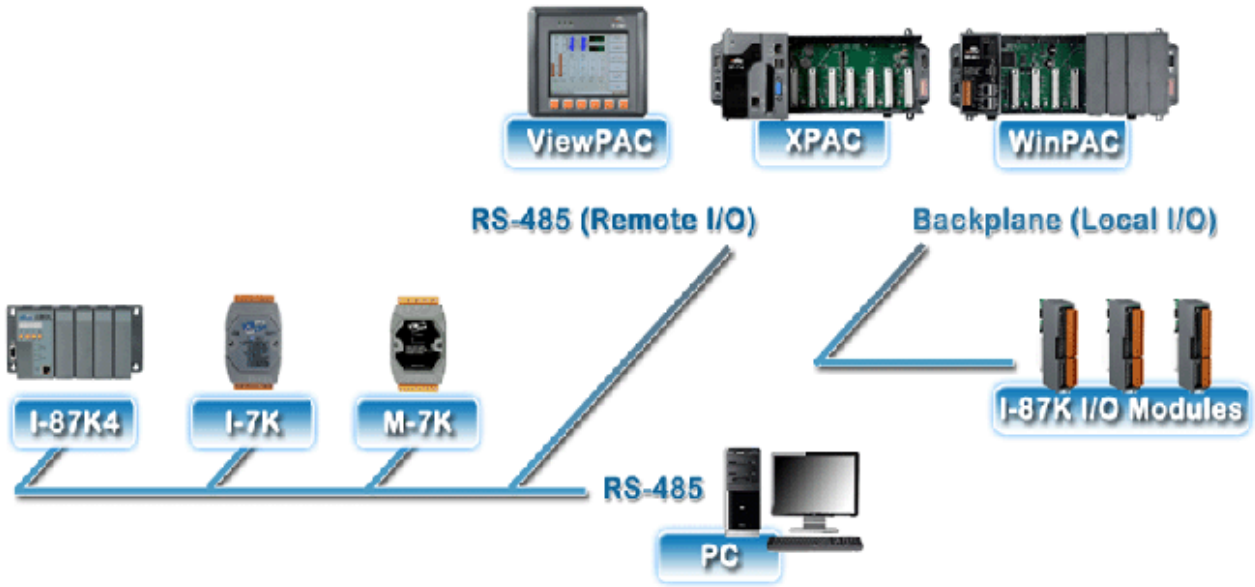
1.3.2. Installing the DCON Utility Pro

The I-97K series I/O module can be configured using the DCON Utility Pro.

The DCON Utility Pro is a toolkit that provides users with an intuitive and easy-to-use interface for searching, configuring and testing the I/O modules via the serial port (RS-232/485) or Ethernet port (using virtual com port) on Windows PC or ICP DAS embedded PAC.

The latest version of the DCON Utility Pro can be obtained from:

https://www.icpdas.com/en/product/guide+Software+Utility_Driver+DCON_Utility_Pro



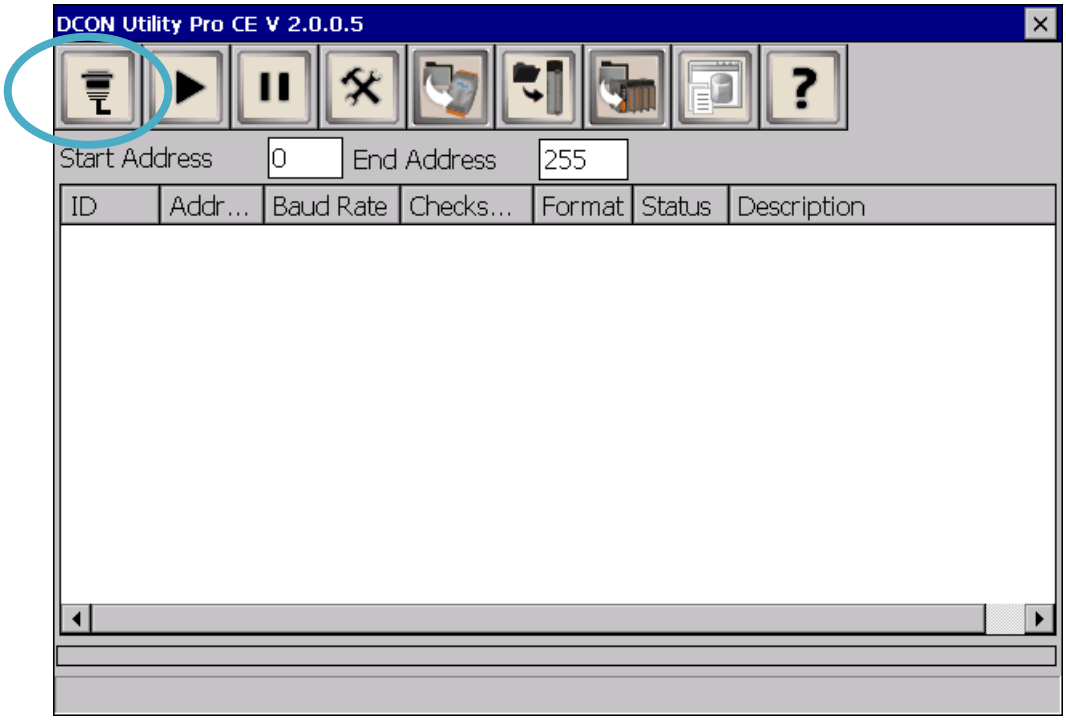
1.4. Configuration

The I-97K series I/O module can be configured using the I/O configuration dialog of the DCON Utility Pro.

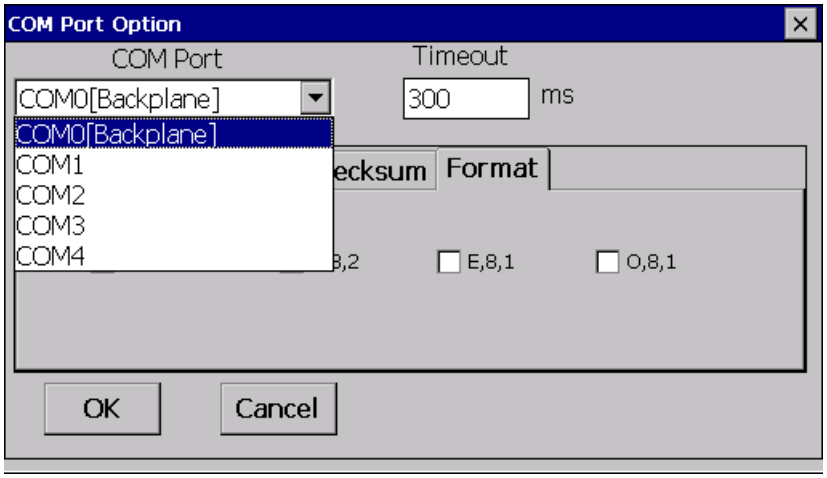
Here are steps on how to configure the I-97k series I/O module.

Step 1. Start the DCON Utility Pro

Step 2. Click the  **button**



Step 3. Configure the communication settings

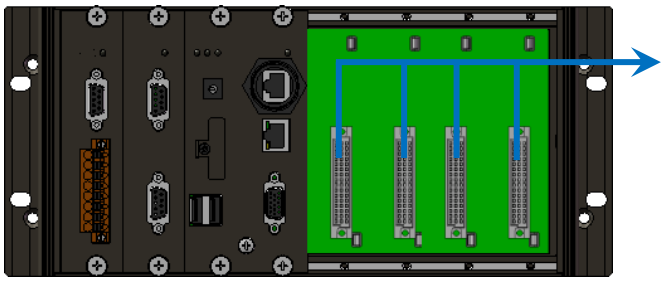


Tips & Warnings

The I-97K series I/O modules are automatically recognized via the backplane of the supported PAC. The backplane on the system that has the assigned COM Port number.

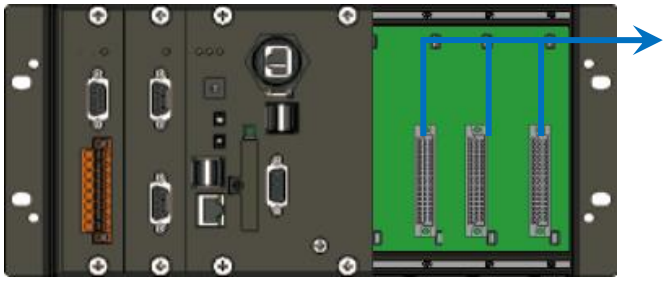
The assigned COM port number for the backplane of the supported PAC are as shown below.

WP-9x2x-CE7



COM0 [Backplane]

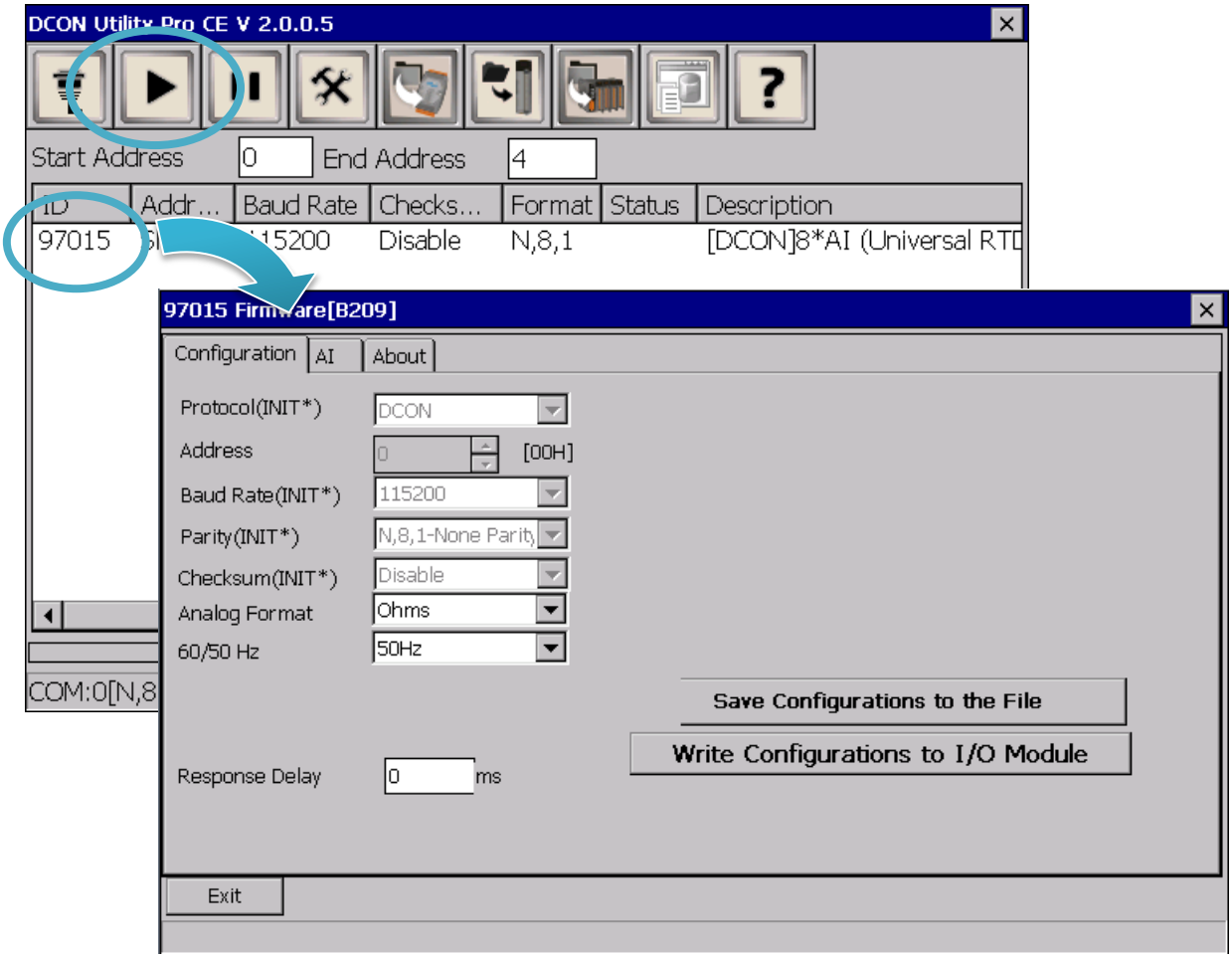
XP-9x7x-WES7



COM1 [Backplane]

Step 4. **Click the**  **button**

Step 5. **Click the module name to configure the I/O module**



2. SDK and Demo Resources

PACSDK comprises a set of API functions and demo resources that you can use to access and develop I-97K series I/O modules. With the PACSDK you can create an application that communicates with your I-97K series I/O modules. Therefore, before programming with I-97K series I/O module, the PACSDK should already be installed on your system.

For more information on how to install and update the PACSDK, please refer to the user manual of the purchased PAC.

PACSDK Installation Guide for 9000 series PACs

- **WP-9x2x-CE7 Series User Manual:**
  https://www.icpdas.com/en/download/show.php?num=44
- **XP-9x7x-WES7 Series User Manual:**
  https://www.icpdas.com/en/download/show.php?num=41

PACSDK updated for Programming Languages

PACSDK can be used with VC++, C# and VB.Net on Windows platforms. The table below shows the programming languages and their corresponding library.

	SDKs	Programming Languages
Native SDK	PACSDK.dll	VC++
.NET CF SDK	PACNET.dll	VC#, VB.NET

PACSDK updated and Demo Usage for Supported PACs

The latest version of the PACSDK and demo resources for the supported PACs can be obtained from:

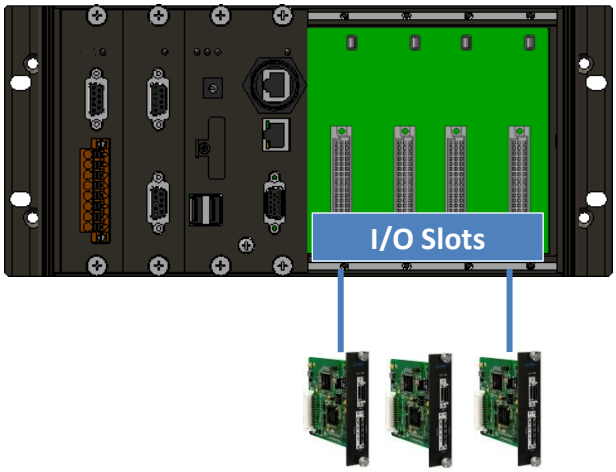
- **WP-9x2x-CE7 SDK:**
  https://www.icpdas.com/en/download/show.php?num=2348
- **XP-9x7x-WES7 SDK:**
  https://www.icpdas.com/en/download/show.php?num=2540

3. API References

ICPDAS supplies a range of C/C++ API functions for the I-97K DIO module. When developing a custom program, refer to PACSDK.h/PACSDK.lib/PACSDK.dll, or the API functions described in the following sections for more detailed information.

ICPDAS also supplies a range of C# function that can be used to develop custom .NET programs. These functions are ported from the relevant C/C++ functions. For more information related to the .NET functions, refer to the PACNET.DLL file.

More details of where to find the relevant libraries and files, and refer to Chapter 1.5 and Chapter 1.6.



I-97K series I/O modules

The table below lists the API functions that can be used to access the I-97K I/O modules.

PACSDK	PACNET	Description
pac_WriteDO	pac_IO.WriteDO	To write the DO values to DO modules.
pac_ReadDO	pac_ IO.ReadDO	To read the DO value of the DO module.
pac_ReadDI	pac_ IO.ReadDI	To read the DI value of the DI module.
pac_ReadDIO	pac_ IO.ReadDIO	To read the DI and the DO values of the DIO module.
pac_ReadDILatch	pac_ IO.ReadDILatch	To read the DI latch value of the DI module.
pac_ClearDILatch	pac_ IO.ClearDILatch	This function clears the latch value of the DI module.
pac_ReadDIOLatch	pac_ IO.ReadDIOLatch	To read the latch values of the DI and DO channels of the DIO module.
pac_ClearDIOLatch	pac_ IO.ClearDIOLatch	To clear the latch values of DI and DO channels of the DIO module.

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

Syntax

C++

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

pac_ClearDICNT	pac_ IO.ClearDICNT	To clear the counter value of the DI channel of the DI module.
pac_WriteAO	pac_ IO.WriteAO	To write the AO value to the AO modules.
pac_ReadAO	pac_ IO.ReadAO	To read the AO value of the AO module.
pac_ReadAI	pac_ IO.ReadAI	To read the engineering-mode AI value of the AI module.
pac_ReadAIHex	pac_ IO.ReadAIHex	To read the 2's complement-mode AI value of the AI module.
pac_ReadAIAllExt	pac_ IO.ReadAIAllExt	To read all the AI values of all channels in engineering-mode of the AI module.
pac_ReadAIAll	pac_ IO.ReadAIAll	To read all the AI values of all channels in engineering-mode of the AI module.
pac_ReadAIAllHexExt	pac_ IO.ReadAIAllHexExt	To read all the AI values of all channels in 2's complement-mode of the AI module.
pac_ReadAIAllHex	pac_ IO.ReadAIAllHex	To read all the AI values of all channels in 2's complement-mode of the AI module. The function maybe causes the buffer overflow in some situation.

PACSDK	PACNET	Description
pac_WriteModulePowerOnValueDO	pac_ IO.WriteModulePowerOnValueDO	To write the DO Power-on values to DO modules.
pac_ReadModulePowerOnValueDO	pac_ IO.ReadModulePowerOnValueDO	To read the Power-on value of the DO modules.
pac_WriteModuleSafeValueDO	pac_ IO.WriteModuleSafeValueDO	To write the DO safe values to DO modules.
pac_WriteModuleSafeValueAO	pac_ IO.WriteModuleSafeValueAO	To write the AO safe value to the AO modules.
pac_ReadModuleSafeValueAO	pac_ IO.ReadModuleSafeValueAO	To read the AO safe value of the AO module.
pac_WriteModulePowerOnValueAO	pac_ IO.WriteModulePowerOnValueAO	To write the AO power on value to the AO modules.
pac_ReadModulePowerOnValueAO	pac_ IO.ReadModulePowerOnValueAO	To read the AO power on value of the AO module.
pac_GetModuleLastOutputSource	pac_ IO.GetModuleLastOutputSource	To read the last output source of a module.
pac_GetModuleWDTStatus	pac_ IO.GetModuleWDTStatus	To read the status of watchdog on the module.
pac_GetModuleWDTConfig	pac_ IO.GetModuleWDTConfig	To read the status of watchdog on a module.

pac_SetModuleWDTConfig	pac_ IO.SetModuleWDTConfig	To enable/disable the host watchdog and sets the host watchdog timeout value of a module.
pac_ReadModuleSafeValueDO	pac_ IO.ReadModuleSafeValueDO	To read the safe value of the DO modules.
pac_ResetModuleWDT	pac_ IO.ResetModuleWDT	To reset the host watchdog timeout status of a module.
pac_RefreshModuleWDT	pac_ IO.RefreshModuleWDT	To refresh the host watchdog of a module.

3.1. pac_WriteDO

This function writes the DO values to DO modules.

Syntax

C++

```
BOOL pac_WriteDO (
    HANDLE hPort,
    int iSlot,
    int iDO_TotalCh,
    DWORD iDO_Value
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iDO_TotalCh

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

iDO_Value

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

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Example

[C]

```
// If the module is 97K local

IntPtr hPort = PACNET.UART.Open("");
int iDO_TotalCh = 8;
int iSlot = 1;
DWORD iDO_Value = 4; // turn on the channel two
BOOL ret = pac_WriteDO(hPort, iSlot, iDO_TotalCh, iDO_Value);
PACNET.UART.Close(hPort);
```

[C#]

```
// If the module is 97K local

IntPtr hPort = PACNET.UART.Open("");
int iDO_TotalCh = 8;
int iSlot = 1;
uint iDO_Value = 4; // turn on the channel two
bool ret = PACNET.IO.WriteDO(hPort, iSlot, iDO_TotalCh, iDO_Value);
PACNET.UART.Close(hPort);
```

Remarks

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

3.2. pac_ReadDO

This function reads the DO value of the DO module.

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iDO_TotalCh

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

lDO_Value

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

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Example

[C]

```
// If the module is 97K local

IntPtr hPort = PACNET.UART.Open("");
BYTE slot = 1;
int iTotal_channel = 8;
DWORD iDo_value;
BOOL ret = pac_ReadDO(hPort, slot , iTotal_channel , &iDo_value );
uart_Close(hPort);
```

[C#]

```
// If the module is 97K local

IntPtr hPort = PACNET.UART.Open("");
byte slot = 1;
int iTotal_channel = 8;
uint iDo_value;
bool ret = PACNET.IO.ReadDO(hPort, slot , iTotal_channel , ref iDo_value );
PACNET.UART.Close(hPort);
```

Remarks

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

3.3. pac_ReadDI

This function reads the DI value of the DI module.

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iDI_TotalCh

[in] The total channels of the DI module.

lDI_Value

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

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Example

[C]

```
// If the module is 97K local

IntPtr hPort = PACNET.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); uart_Close(hPort);
```

[C#]

```
// If the module is 97K local

IntPtr hPort = PACNET.UART.Open("");
byte iSlot = 2;
int iDI_TotalCh = 8;
uint lDI_Value;
bool iRet = PACNET.IO.ReadDI(hPort, iSlot, iDI_TotalCh, ref lDI_Value);
PACNET.UART.Close(hPort);
```

Remarks

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

3.4. pac_ReadDIO

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

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot. 0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iDI_TotalCh

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

iDO_TotalCh

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

lDI_Value

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

lDO_Value

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

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 97K local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot=1;
int iDI_TotalCh=8;
int iDO_TotalCh=8;
DWORD lDI_Value;
DWORD lDO_Value;
BOOL iRet = pac_ReadDIO(hPort, iSlot,iDI_TotalCh, iDO_TotalCh, &lDI_Value, &lDO_Value);
uart_Close(hPort);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot=1;
int iDI_TotalCh=8;
int iDO_TotalCh=8;
uint lDI_Value;
uint lDO_Value;
bool iRet = PACNET.IO.ReadDIO(hPort, iSlot,iDI_TotalCh, iDO_TotalCh, ref lDI_Value, ref
lDO_Value);
PACNET.UART.Close(hPort);
```

Remarks

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

3.5. pac_ReadDILatch

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

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iDI_TotalCh

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

iLatchType

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

1 = latched high status

0 = latched low status

lDI_Latch_Value

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

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 97K local
HANDLE hPort;
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);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot=1;
int iDI_TotalCh=8;
int iLatchType=0;
uint lDI_Latch_Value;
bool iRet = PACNET.IO.ReadDILatch(hPort, iSlot, iDI_TotalCh, iLatchType, ref lDI_Latch_Value);
PACNET.UART.Close(hPort);
```

Remarks

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

3.6. pac_ClearDILatch

This function clears the latch value of the DI module.

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 97K local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot=1;
BOOL iRet = pac_ClearDILatch(hPort, iSlot);
uart_Close(hPort);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot=1;
bool iRet = PACNET.IO.ClearDILatch(hPort, iSlot);
PACNET.UART.Close(hPort);
```

Remarks

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

3.7. pac_ClearDIOLatch

This function clears the latch values of DI and DO channels of the DIO module.

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 97K local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot=1;
BOOL iRet = pac_ClearDIOLatch(hPort, iSlot);
uart_Close(hPort);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot=1;
bool iRet = PACNET.IO.ClearDIOLatch(hPort, iSlot);
PACNET.UART.Close(hPort);
```

Remarks

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

3.8. pac_ReadDIOLatch

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

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iDI_TotalCh

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

iDO_TotalCh

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

iLatchType

[in] The type of the latch value read back.

1 = latched high status

0 = latched low status

lDI_Latch_Value

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

lDO_Latch_Value

[out] The pointer to the DO latch value read back.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 97K local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot=1;
int iDI_TotalCh=8;
int iDO_TotalCh=8;
int iLatchType=0;
DWORD lDI_Latch_Value;
DWORD lDO_Latch_Value;
BYTE cDI_Latch_BitValue;
BYTE cDO_Latch_BitValue;
BOOL iRet = pac_ReadDIOLatch(hPort, iSlot,iDI_TotalCh,iDO_TotalCh,iLatchType,
&lDI_Latch_Value,&lDO_Latch_Value);
uart_Close(hPort);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot=1;
int iDI_TotalCh=8;
int iDO_TotalCh=8;
int iLatchType=0;
uint lDI_Latch_Value;
uint lDO_Latch_Value;
byte cDI_Latch_BitValue;
byte cDO_Latch_BitValue;
bool iRet =PACNET.IO.ReadDIOLatch(hPort,iSlot,iDI_TotalCh,iDO_TotalCh,iLatchType, ref
lDI_Latch_Value, ref lDO_Latch_Value);
PACNET.UART.Close(hPort);
```

Remarks

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

3.9. pac_ReadDICNT

This function reads the counts of the DI channels of the DI module.

Syntax

C++

```
BOOL pac_ReadDICNT(
    HANDLE hPort,
    int iSlot,
    int iChannel,
    int iDI_TotalCh,
    DWORD *lCounter_Value
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] The channel that the counter value belongs.

iDI_TotalCh

[in] Total number of the DI channels of the DI module.

lCounter_Value

[out] The pointer to the counter value.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 97K local
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);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot=1;
int iChannel =2;
int iDI_TotalCh=8;
uint lCounter_Value;
bool iRet = PACNET.IO.ReadDICNT(hPort, iSlot,iChannel,iDI_TotalCh, ref lCounter_Value);
PACNET.UART.Close(hPort);
```

Remarks

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

3.10. pac_ClearDICNT

This function clears the counter value of the DI channel of the DI module.

Syntax

C++

```
BOOL pac_ClearDICNT(
    HANDLE hPort,
    int iSlot,
    int iChannel,
    int iDI_TotalCh
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] The channel that the counter value belongs.

iDI_TotalCh

[in] Total number of the DI channels of the DI module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 97K local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot=1;
int iChannel=2;
int iDI_TotalCh=8;
BOOL iRet = pac_ClearDICNT(hPort, iSlot,iChannel,iDI_TotalCh);
uart_Close(hPort);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot=1;
int iChannel=2;
int iDI_TotalCh=8;
bool iRet = PACNET.IO.ClearDICNT(hPort, iSlot,iChannel,iDI_TotalCh);
PACNET.UART.Close(hPort);
```

Remarks

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

3.11. pac_WriteAO

This function writes the AO value to the AO modules.

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] The channel that is written thee AO value to.

iAO_TotalCh

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

float fValue

[in] The AO value to write to the AO module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 97K local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot=1;
int iChannel=2;
int iAO_TotalCh=8;
float fValue=5;
BOOL iRet = pac_WriteAO(hPort, iSlot, iChannel, iAO_TotalCh, fValue);
uart_Close(hPort);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot=1;
int iChannel=2;
int iAO_TotalCh=8;
float fValue=5;
bool iRet = PACNET.IO.WriteAO(hPort, iSlot,iChannel,iAO_TotalCh,fValue);
PACNET.UART.Close(hPort);
```

Remarks

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

3.12. pac_ReadAO

This function reads the AO value of the AO module.

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] Read the AO value from the channel.

iAO_TotalCh

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

float fValue

[in] The pointer to the AO value that is read back from the AO module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 97K local
HANDLE hPort;
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);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot=1;
int iChannel=2;
int iAO_TotalCh=8;
float fValue;
bool iRet = PACNET.IO.ReadAO(hPort, iSlot,iChannel,iAO_TotalCh,ref fValue);
PACNET.UART.Close(hPort);
```

Remarks

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

3.13. pac_ReadAI

This function reads the engineering-mode AI value of the AI module.

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] Read the AI value from the channel.

iAI_TotalCh

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

fValue

[in] The pointer to the AI value that is read back from the AI module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 97K local
HANDLE hPort;
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);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot=1;
int iChannel=2;
int iAI_TotalCh=8;
float fValue;
bool iRet = PACNET.IO.ReadAI(hPort, iSlot,iChannel,iAI_TotalCh, ref fValue);
PACNET.UART.Close(hPort);
```

Remarks

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

3.14. pac_ReadAIHex

This function reads the 2's complement-mode AI value of the AI module.

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] Read the AI value from the channel.

iAI_TotalCh

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

iValue

[in] The pointer to the AI value that is read back from the AI module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 97K local
HANDLE hPort;
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);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot=1;
int iChannel=2;
int iAI_TotalCh=8;
int iValue;
bool iRet = PACNET.IO.ReadAIHex(hPort, iSlot,iChannel,iAI_TotalCh, ref iValue);
PACNET.UART.Close(hPort);
```

Remarks

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

3.15. pac_ReadAIAllExt

This function reads all the AI values of all channels in engineering-mode of the AI module.

This function replaces pac_ReadAIAll.

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

fValue[]

[out] The array contains the AI values that read back from the AI module.

Buff_Len

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

Channel

[out] The pointer to a variable that specifies the total available channel numberer of AI module. This channel number is only valid if the return value is TRUE.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 97K local
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);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
Int channelnumber=0;
hPort = PACNET.UART.Open("");
byte iSlot=1;
float fValue[8];
bool iRet = PACNET.IO.ReadAIAllExt(hPort, iSlot, fValue, 8, ref channelnumber);
PACNET.UART.Close(hPort);
```

Remarks

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

3.16. pac_ReadAIAll

This function reads all the AI values of all channels in engineering-mode of the AI module.

The function maybe causes the buffer overflow in some situation.

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

fValue[]

[out] The array contains the AI values that read back from the AI module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 97K local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot=1;
float fValue[8];
BOOL iRet = pac_ReadAIAll(hPort, iSlot, fValue);
uart_Close(hPort);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot=1;
float fValue[8];
bool iRet = PACNET.IO.ReadAIAll(hPort, iSlot, fValue);
PACNET.UART.Close(hPort);
```

Remarks

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

3.17. pac_ReadAIAllHexExt

This function reads all the AI values of all channels in 2's complement-mode of the AI module.

This function replaces pac_ReadAIAllHex.

Syntax

C++

```
BOOL pac_ReadAIAllHex(
    HANDLE hPort,
    int iSlot,
    int iValue[],
    DWORD Buff_Len,
    DWORD *Channel
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iValue[]

[out] The array contains the AI values that read back from the AI module.

Buff_Len

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

Channel

[out] The pointer to a variable that specifies the total available channel numberer of AI module. This channel number is only valid if the return value is TRUE.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 97K local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot=1;
int iValue[8];
int ichannelnumber=0;
BOOL iRet = pac_ReadAIAllHexExt(hPort, iSlot, iValue, 8, &ichannelnumber);
uart_Close(hPort);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot=1;
int ichannelnumber=0;
int iValue[8];
bool iRet = PACNET.IO.ReadAIAllHex(hPort, iSlot, iValue, 8, ref ichannelnumber);
PACNET.UART.Close(hPort);
```

Remarks

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

3.18. pac_ReadAIAllHex

This function reads all the AI values of all channels in 2's complement-mode of the AI module.

The function maybe causes the buffer overflow in some situation.

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iValue[]

[out] The array contains the AI values that read back from the AI module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 97K local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot=1;
int iValue[8];
BOOL iRet = pac_ReadAIAllHex(hPort, iSlot, iValue);
uart_Close(hPort);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot=1;
int iValue[8];
bool iRet = PACNET.IO.ReadAIAllHex(hPort, iSlot, iValue);
PACNET.UART.Close(hPort);
```

Remarks

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

3.19. pac_WriteModulePowerOnValueDO

This function writes the DO Power-on values to DO modules.

Syntax

C++

```
bool pac_WriteModulePowerOnValueDO(
    HANDLE hPort,
    int iSlot,
    int iDO_TotalCh,
    unsigned long lValue
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iDO_TotalCh

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

iValue

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

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Example

[C]

```
HANDLE hPort = uart_Open("");
Int iSlot = 0;
Int iDO_TotalCh=32;
Int iValue = 0xffffffff;
PACNET.PAC_IO.WriteModulePowerOnValueDO(hPort, iSlot,
        iDO_TotalCh, iValue);
uart_Close(hPort);
```

[C#]

```
IntPtr hPort = PACNET.UART.Open("");
Int iSlot = 0;
int iDO_TotalCh = 32;
uint iValue = 4; // turn on the channel two
bool ret = PACNET.IO. pac_WriteModulePowerOnValueDO(hPort, iSlot ,
        iDO_TotalCh , iValue );
PACNET.UART.Close(hPort);
```

Remarks

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

3.20. pac_ReadModulePowerOnValueDO

This function reads the Power-on value of the DO modules.

Syntax

C++

```
BOOL pac_ReadModulePowerOnValueDO (
    HANDLE hPort,
    int iSlot,
    int iDO_TotalCh,
    unsigned long *lValue
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

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

lValue

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

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Example

[C]

```
HANDLE hPort;
hPort = uart_Open("");
BYTE slot = 1;
int total_channel = 32;
DWORD do_value;
BOOL ret = pac_ReadModulePowerOnValueDO(hPort, slot , total_channel , do_value );
uart_Close(hPort);
```

[C#]

```
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte slot = 1;
int total_channel = 32;
uint do_value;
bool ret = PACNET.IO. pac_ReadModulePowerOnValueDO (hPort, slot , total_channel , ref
do_value );
PACNET.UART.Close(hPort);
```

Remarks

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

3.21. pac_WriteModuleSafeValueDO

This function writes the DO safe values to DO modules.

Syntax

C++

```
BOOL pac_WriteModuleSafeValueDO(
    HANDLE hPort,
    int iSlot,
    int iDO_TotalCh,
    DWORD lValue
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iDO_TotalCh

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

iValue

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

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Example

[C]

```
HANDLE hPort = uart_Open("");
int iSlot = 1;
int total_channel = 32;
DWORD do_value = 4; // turn on the channel two
BOOL ret = pac_WriteModuleSafeValueDO(hPort, iSlot , total_channel , do_value );
uart_Close(hPort);
```

[C#]

```
IntPtr hPort; hPort = PACNET.UART.Open("");
int iSlot = 1;
int total_channel = 32;
uint do_value = 4; // turn on the channel two
bool ret = PACNET.IO. pac_WriteModuleSafeValueDO(hPort, iSlot , total_channel , do_value );
PACNET.UART.Close(hPort);
```

Remarks

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

3.22. pac_WriteModuleSafeValueAO

This function writes the AO safe value to the AO modules.

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] The channel that is written thee AO value to.

iAO_TotalCh

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

float fValue

[in] The AO value to write to the AO module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 97K local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot=1;
int iChannel=2;
int iAO_TotalCh=8;
float fValue=5;
BOOL iRet = pac_WriteModuleSafeValueAO(hPort, iSlot,iChannel,iAO_TotalCh,fValue);
uart_Close(hPort);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot=1;
int iChannel=2;
int iAO_TotalCh=8;
float fValue=5;
bool iRet = PACNET.IO.WriteModuleSafeValueAO(hPort, iSlot,iChannel,iAO_TotalCh,fValue);
PACNET.UART.Close(hPort);
```

Remarks

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

3.23. pac_ReadModuleSafeValueAO

This function reads the AO safe value of the AO module.

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] Read the AO value from the channel.

iAO_TotalCh

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

float fValue

[in] The pointer to the AO safe value that is read back from the AO module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 97K local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot=1;
int iChannel=2;
int iAO_TotalCh=8;
float fValue;
BOOL iRet = pac_ReadModuleSafeValueAO(hPort, iSlot,iChannel,iAO_TotalCh, &fValue);
uart_Close(hPort);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot=1;
int iChannel=2;
int iAO_TotalCh=8;
float fValue;
bool iRet = PACNET.IO.ReadModuleSafeValueAO(hPort, iSlot,iChannel,iAO_TotalCh,ref fValue);
PACNET.UART.Close(hPort);
```

Remarks

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

3.24. pac_WriteModulePowerOnValueAO

This function writes the AO power on value to the AO modules.

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] The channel that is written thee AO value to.

iAO_TotalCh

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

float fValue

[in] The AO value to write to the AO module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 97K local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot=1;
int iChannel=2;
int iAO_TotalCh=8;
float fValue=5;
BOOL iRet = pac_WriteModulePowerOnValueAO(hPort, iSlot,iChannel,iAO_TotalCh,fValue);
uart_Close(hPort);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot=1;
int iChannel=2;
int iAO_TotalCh=8;
float fValue=5;
bool iRet = PACNET.IO.WriteModulePowerOnValueAO(hPort,
iSlot,iChannel,iAO_TotalCh,fValue);
PACNET.UART.Close(hPort);
```

Remarks

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

3.25. pac_ReadModulePowerOnValueAO

This function reads the AO power on value of the AO module.

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

[in] Read the AO value from the channel.

iAO_TotalCh

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

float fValue

[in] The pointer to the AO power on value that is read back from the AO module.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 97K local
HANDLE hPort;
hPort = uart_Open("");
BYTE iSlot=1;
int iChannel=2;
int iAO_TotalCh=8;
float fValue;
BOOL iRet = pac_ReadModulePowerOnValueAO(hPort, iSlot,iChannel,iAO_TotalCh, &fValue);
uart_Close(hPort);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot=1;
int iChannel=2;
int iAO_TotalCh=8;
float fValue;
bool iRet = PACNET.IO.ReadModulePowerOnValueAO(hPort, iSlot,iChannel,iAO_TotalCh,ref
fValue);
PACNET.UART.Close(hPort);
```

Remarks

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

3.26. pac_GetModuleLastOutputSource

This function reads the last output source of a module.

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

Return Value

0: No action

1: by Power On Value

2: by Safe Value

3: by regular DO command

Examples

[C]

```
// If the module is 97K local
HANDLE hPort;
int iSlot =0;
int lastOutput=0;
hPort = uart_Open("");
lastOutput = pac_GetModuleLastOutputSource(hPort , iSlot);
uart_Close(hPort);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
int iSlot =0;
hPort = PACNET.UART.Open("");
int lastOutput= PACNET.IO.GetModuleLastOutputSource(hPort , iSlot);
PACNET.UART.Close(hPort);
```

Remarks

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

3.27. pac_GetModuleWDTStatus

This function reads the status of watchdog on the module.

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

Return Value

If the return value is TRUE, it means watchdog timeout occurred

If the return value is FALSE, it means watchdog timeout doesn't occur.

Examples

[C]

```
// If the module is 97K local
HANDLE hPort;
int iSlot =0;
bool bStatus=0;
hPort = uart_Open("");
bStatus = pac_GetModuleWDTStatus (hPort , iSlot);
uart_Close(hPort);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
int iSlot =0;
hPort = PACNET.UART.Open("");
bool bStatus= PACNET.IO. pac_GetModuleWDTStatus(hPort , iSlot);
PACNET.UART.Close(hPort);
```

Remarks

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

3.28. pac_GetModuleWDTConfig

This function reads the status of watchdog on a module.

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

enStatus

[out] 1: the host watchdog is enabled

0: the host watchdog is disabled

wdtTimeout

[out] The unit of return value is 100ms.

ifWDT_Overwrite

[out] 1: the host watchdog does overwrite

0: the host watchdog does not overwrite

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 97K local
HANDLE hPort;
int iSlot =0;
short sStatus=0;
unsigned long ulWDTtime=0;
int iOverwrite= 0;
hPort = uart_Open("");
pac_GetModuleWDTConfig (hPort, iSlot, &sStatus, &ulWDTtime, &iOverwrite);
uart_Close(hPort);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
int iSlot =0;
short sStatus=0;
unsigned long ulWDTtime=0;
int iOverwrite= 0;
hPort = PACNET.UART.Open("");
PACNET.IO. GetModuleWDTConfig (hPort , iSlot, ref sStatus, ref ulWDTtime, ref iOverwrite);
PACNET.UART.Close(hPort);
```

Remarks

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

3.29. pac_SetModuleWDTConfig

This function enables/disables the host watchdog and sets the host watchdog timeout value of a module.

Syntax

C++

```
bool pac_SetModuleWDTConfig(
    HANDLE hPort,
    int iSlot,
    short enStatus,
    unsigned long wdtTimeout,
    int ifWDT_Overwrite
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

enStatus

[in] 1: the host watchdog is enabled

0: the host watchdog is disabled

wdtTimeout

[in] The unit of return value is 100ms.

ifWDT_Overwrite

[in] 1: the host watchdog does overwrite

0: the host watchdog does not overwrite

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Example

[C]

```
HANDLE hPort = uart_Open("");
int iSlot =0;
short sStatus=0;
unsigned long ulWDTtime=0;
int iOverwrite= 0;
pac_SetModuleWDTConfig (hPort, iSlot, sStatus, ulWDTtime, iOverwrite);
uart_Close(hPort);
```

[C#]

```
IntPtr hPort = PACNET.UART.Open("");
int iSlot =0;
short sStatus=0;
unsigned long ulWDTtime=0;
int iOverwrite= 0;
PACNET.IO.SetModuleWDTConfig (hPort , iSlot, sStatus, ulWDTtime, iOverwrite);
PACNET.UART.Close(hPort);
```

Remarks

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

3.30. pac_ReadModuleSafeValueDO

This function reads the safe value of the DO modules.

Syntax

C++

```
BOOL pac_ReadModuleSafeValueDO (
    HANDLE hPort,
    int iSlot,
    int iDO_TotalCh,
    unsigned long *lValue
);
```

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro, PAC_REMOTE_IO(0...255).

iChannel

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

lValue

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

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Example

[C]

```
HANDLE hPort = uart_Open("");
BYTE iSlot = 1;
int iTotal_channel = 32;
DWORD iDo_value;
BOOL ret = pac_ReadModuleSafeValueDO(hPort, iSlot , iTotal_channel , &iDo_value );
uart_Close(hPort);
```

[C#]

```
IntPtr hPort = PACNET.UART.Open("");
byte iSlot = 1;
int iTotal_channel = 32;
uint iDo_value;
bool ret = PACNET.IO. pac_ReadModuleSafeValueDO (hPort, iSlot , iTotal_channel , ref
iDo_value );
PACNET.UART.Close(hPort);
```

Remarks

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

3.31. pac_ResetModuleWDT

This function resets the host watchdog timeout status of a module.

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97k modules plugged in local slot.

0, if the module is 9k modules plugged in local slot.

iSlot

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

If the IO module is remote, please use the macro,

PAC_REMOTE_IO(0...255).

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
// If the module is 97K local
HANDLE hPort;
int iSlot =0;
hPort = uart_Open("");
pac_ResetModuleWDT(hPort, iSlot);
uart_Close(hPort);
```

[C#]

```
// If the module is 97K local
IntPtr hPort;
int iSlot =0;
hPort = PACNET.UART.Open("");
PACNET.IO.ResetModuleWDT(hPort , iSlot);
PACNET.UART.Close(hPort);
```

Remarks

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

3.32. pac_RefreshModuleWDT

This function refreshes the host watchdog of a module.

Syntax

C++

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

Parameters

hPort

[in] The serial port HANDLE opened by uart_Open(), if the module is 97K modules in local.

iSlot

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

If the IO module is remote, please use the macro,

PAC_REMOTE_IO(0...255).

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Example

[C]

```
HANDLE hPort = uart_Open("");
int iSlot =0;
pac_RefreshModuleWDT(hPort, iSlot);
uart_Close(hPort);
```

[C#]

```
IntPtr hPort = PACNET.UART.Open("");
int iSlot =0;
PACNET.IO.RefreshModuleWDT(hPort , iSlot);
PACNET.UART.Close(hPort);
```

Remarks

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

Appendix.

A. API Functions for Using with DCON Protocol Commands

The 9000 series PAC can uses DCON protocol to communicate with I-97K series I/O modules via the RS-232 interface of the backplane. DCON protocol is a very simple and easy used ASCII based protocol. It is suit query-response I/O applications.

The 9000 series PAC uses the DCON commands to communicate with I-97K series I/O modules that can be implemented by calling the UART API functions included from PACSDK.

For Example:

```
HANDLE hPort = uart_Open("");

uart_SendCmdExt(hPort,"$00M",30,result,30); //Use "$AAM" to read module name

uart_Close(hPort);
```

The table below provides links to the corresponding I-87K series modules and their related DCON instruction sets for the I-97K series modules.

I-97K and the corresponding I-87K		Default Settings	Command Sets
I-97015	I-87015W	Default Settings	Command Set
I-97017Z	I-87017ZW	Default Settings	Command Set
I-97018	I-87018PW	Default Settings	Command Set
I-97019	I-87019RW	Default Settings	Command Set
I-97024U	I-87024UW	Default Settings	Command Set
I-97028U	I-87028UW	Default Settings	Command Set