I-97K I/O Module
Common User Manual

Version 1.0.2 April 2021

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

Preface

The I-97K I/O modules are based on a serial interface and I-97K I/O module must be plugged into the 9000 PAC series(WP-9000, XP-9000, LX-9000 and LP-9000 **)** and the module can function properly. All of I-97K DO modules provide programmable Power-on value / safe value /Retentive functions and All of I-97K DI modules provide DI Low Pass Filter function.

The information contained in this manual is divided into the following topics:

- Chapter 1, “Introductions” – This chapter provides information related to the hardware, such as the specifications, the jumper settings details and wiring information.
- Chapter 2, “I-97K I/O module features” – This chapter introduces the features of I-97K AIO/DIO module.
- Chapter 3, “API References” – This chapter describes the functions provided in the I-97K library together with an explanation of the differences in the naming rules used for the different Windows platforms.
- Chapter 4, “DCON protocol commands” — This chapter introduces the DCON protocol commands.

Chapter 1. Introductions

1.1. Introductions

I-97K series modules are provided for combining a variety of I/O functions within the 9000 series programmable automation controllers (PAC). The I-97K series module is based on a serial interface. The differences between the I-9K and I-97K series are listed as follows:

I/O module features comparison

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

Now I-97K series modules include:

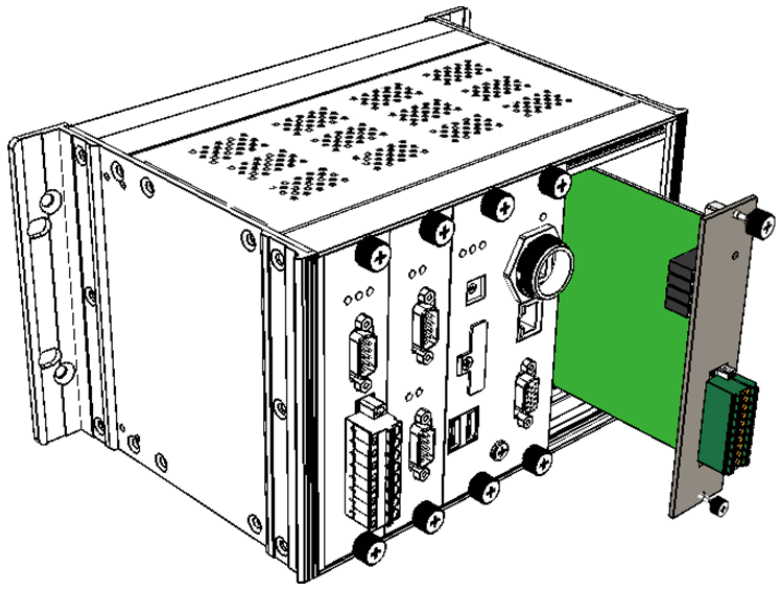
- I-97015 : 8-channel RTD Input Module
- I-97017Z : 10/20-channel Analog Input Module with High Voltage Protection
- I-97018 : 8-channel Thermocouple Input Module
- I-97019 : 8-channel Thermocouple Input Module
- I-97024U : 4-channel Isolated Source Type Voltage or Current Output module
- I-97028U :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

Refer to http://www.icpdas.com/root/product/solutions/remote_io/i-9k_i-97k/i-9k_i-97k_dio.html for more details regarding of the module specification.

Those I-97K I/O Modules must work then plugin any slot with the following PAC:

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

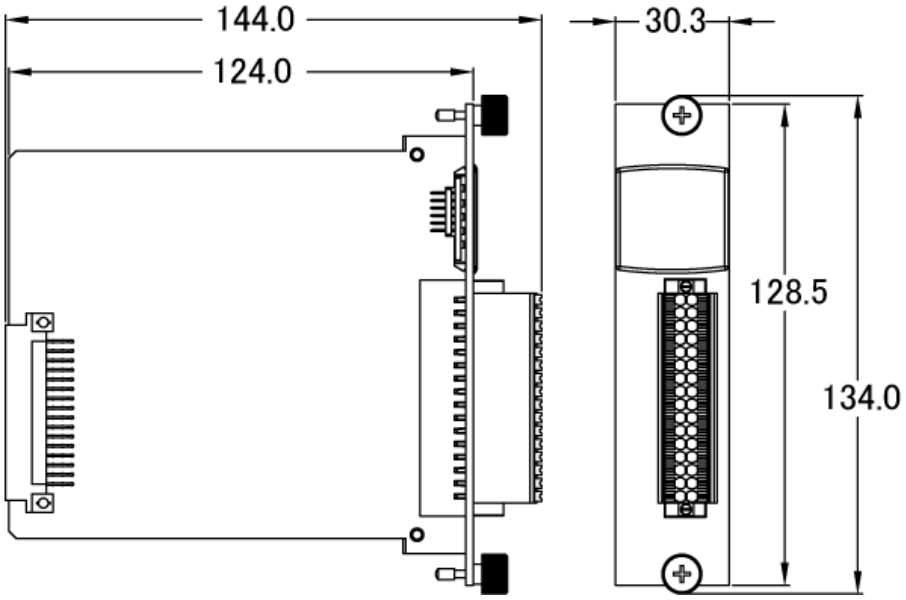
The 9000 PAC series above has expansion slots that enable the addition of optional I/O modules for expanding the capability of the main



1.2. I/O Module Dimensions

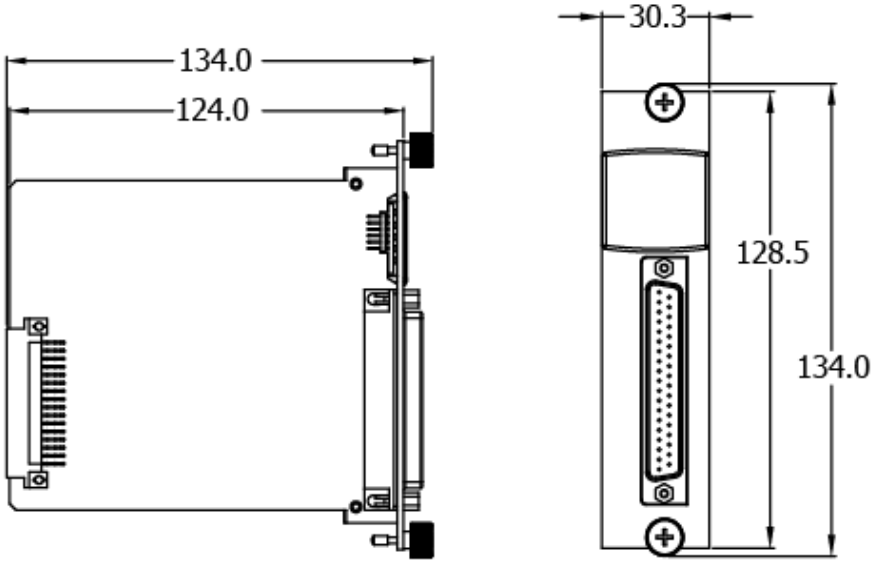
All dimensions are in millimeters.

I-97K module with Spring clamp terminal connector



Left Side View | Top View

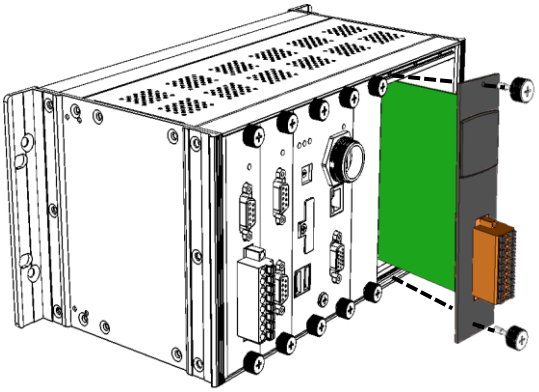
I-97K module with D-Sub connector



Left Side View | Top View

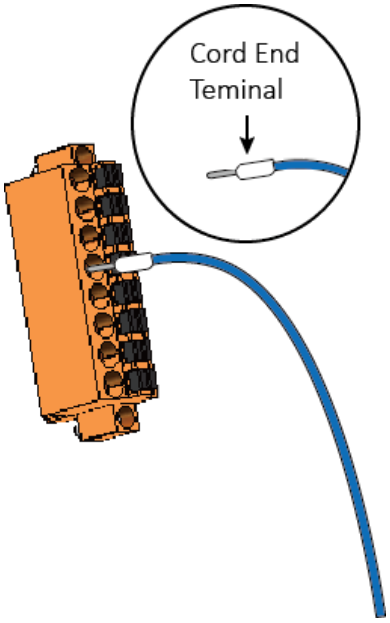
1.3. Inserting the I/O Modules

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.

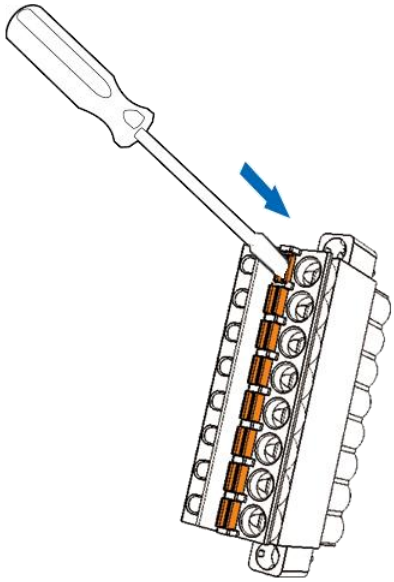


Note:

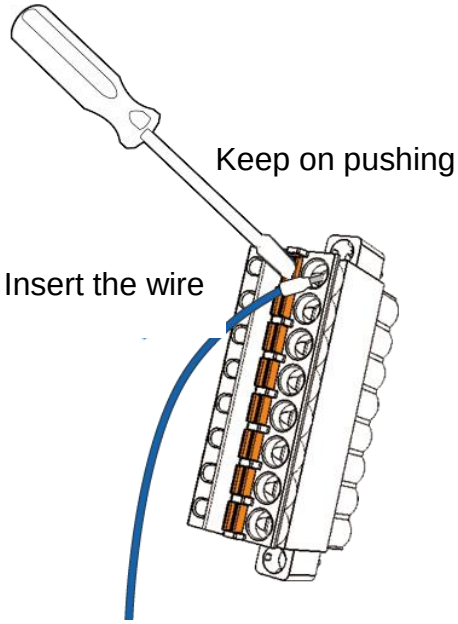
Except I-97018/I-97019 modules, the other I-97K I/O modules support spring clamp terminal connector. The spring clamp terminal connector for the I-97K I/O module connector offers the advantages (anti-vibration, stable clamping and installation easier) relative to screw terminals.

A tip on how to connect the wiring to the connector

1. Use screwdriver to push the orange clip in.

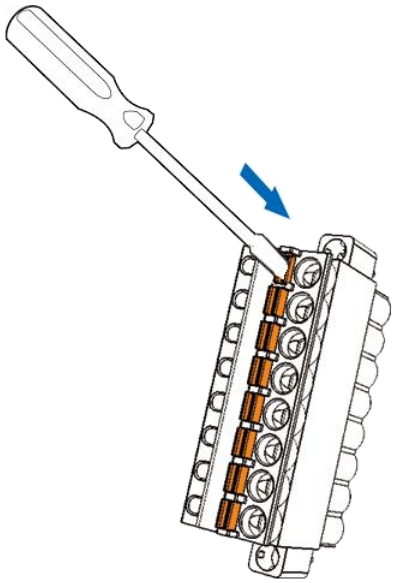


2. Insert the wiring into the terminal block

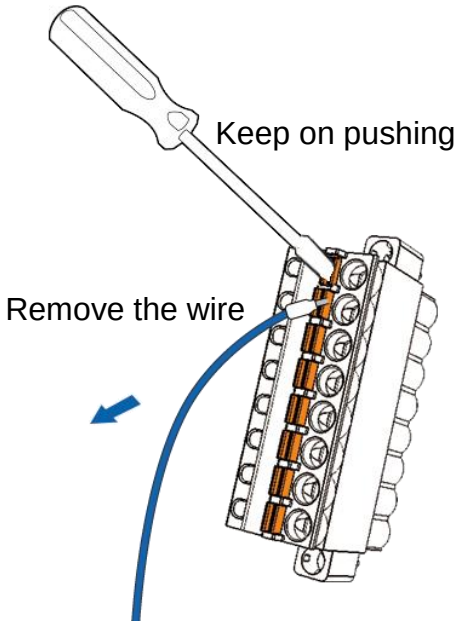


A tip on how to remove the wiring from the connector

1. Use screwdriver to push the orange clip in.



2. Remove the wiring from the terminal block



1.4. I-97K I/O module using the DCON Utility

ICP DAS provides a tool known as the “DCON Utility Pro” which can be used to simplify search, configuration and testing operations for I/O modules, as well as providing the ability to verify the device settings and I/O functions, and can be used on all versions of Windows.

Support DCON and Modbus: DCON Utility Pro can support DCON and Modbus protocol for all ICPDAS and the others modules. It can select multi-options such as BaudRate, Checksum , Format and etc options for search module.

Download the DCON Utility pro from:

Platform		Location
WP-9000-CE7	CD	CD:\WinPAC_AM335x\Wp-5231\System_Disk\Tools\DCON_Utility_Pro
	FTP	http://ftp.icpdas.com.tw/pub/cd/winpac_am335x/wp-5231/system_disk/tools/dcon_utility_pro
XP-9000(WES)	CD	CD:\ XPAC\XPAC-Atom\tools\DCON_Utility_pro
	FTP	http://ftp.icpdas.com.tw/pub/cd/xpac-atom/tools/dcon_utility_pro/

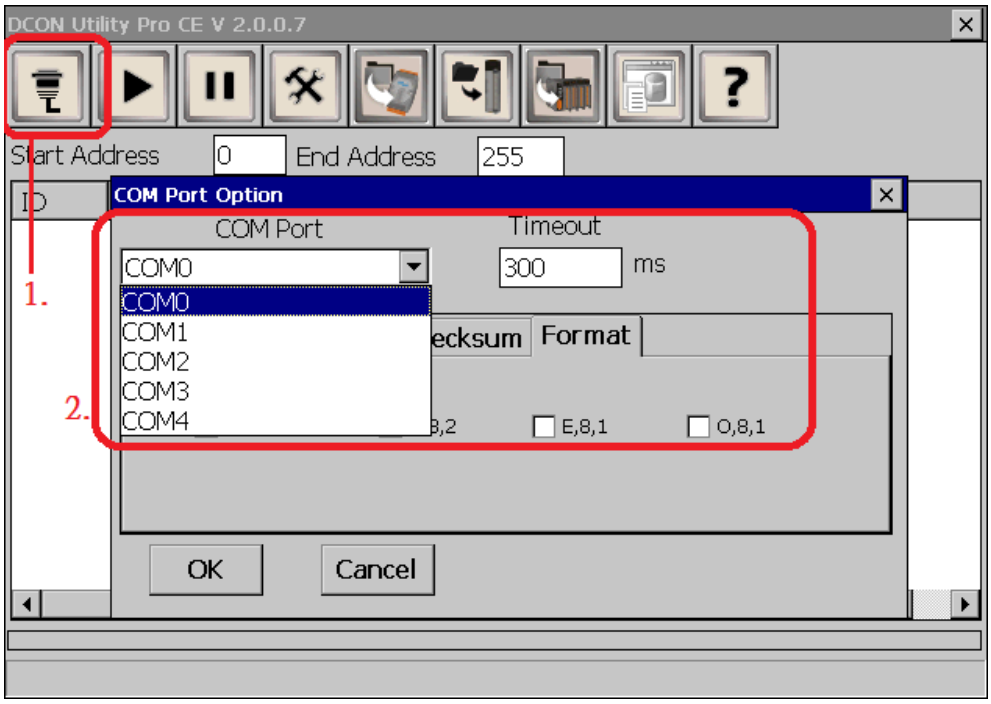
More information about DCON Utility can reference as below link:

http://ftp.icpdas.com/pub/cd/8000cd/napdos/driver/dcon_utility/Manual/DCON_Utility_Pro_usermanual_v1.1_20150508.pdf

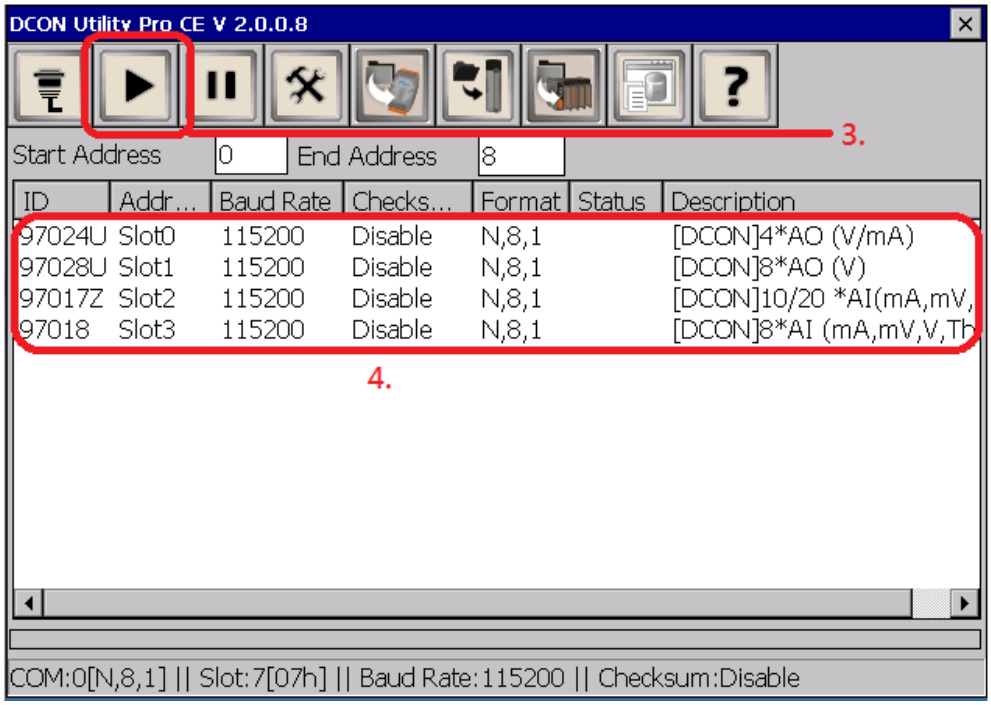
Follow the procedure described below to test the I-97K I/O module.

Step 1 : Click the button to open “COM Port Option”.

Step 2 : Select COM Port and format.



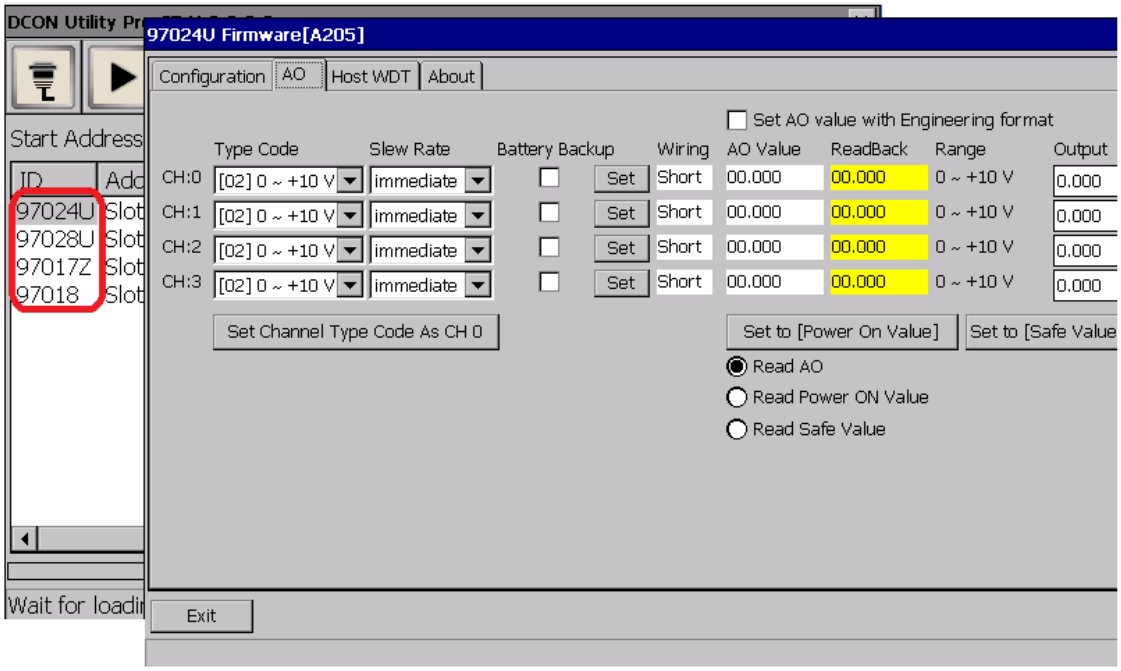
Step 3 : Click button to start scan Module.



Step 4 : If any I/O devices are located, they will be displayed in the device list window.

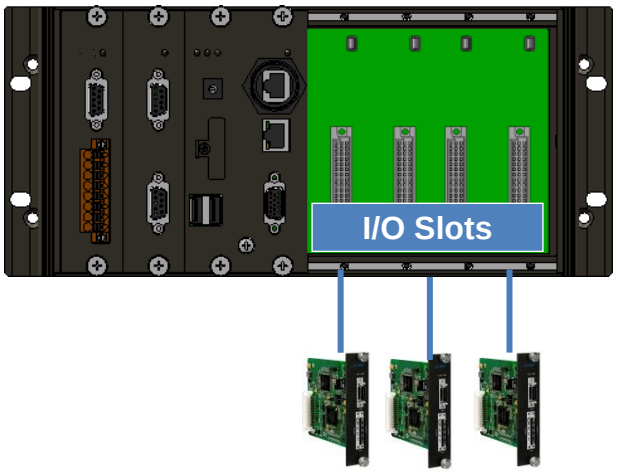
Step 5 : Double-click the name of the module to open the configuration dialog box.

Step 6 : Use the DCON Utility Pro to configure the parameters for AO, Power-on Mode, WDT and others to quickly test all the I-97K I/O functions, as illustrated in the example below



1.5. I-97K I/O module using the SDK API

Selecting an SDK Library for I-97K I/O Module



I-97K series I/O modules

The following table shows the appropriate SDK library to be used for I-97K I/O modules.

SDK	I-97K series
Native SDK	PACSDK.dll
.NET CF SDK	PACNET.dll

ICP DAS provides APIs, libraries and demo programs, including the source code, that allow integration of the I-97K I/O into Windows platforms. For more detailed information regarding the use of these functions, refer to Chapter 3.

Considering the PAC upgrade and software migration, the number and name for each PACSDK.dll and PACNET.dll function for I-97K module plugged on 9000 PAC series and I-87K module plugged on 8000 PAC series are the same. The benefits of the implementing a unified SDK is that the programs for each platform can be easily migrated.

All API functions of the PACSDK.dll and PACNET.dll are used to read/write DI/AI, DO/AO, and DI latched status in the program. The configuration settings for I-97K I/O must be fsmodified by the DCON utility, refer to the Chapter 1.4 “I-97K module using DCON Utility Pro” for more details.

1.6. Location of the Demo and library Programs

ICP DAS provides a range of demo programs for different platforms that can be used to verify the functions of the I-97K I/O modules. The source code contained in these programs can also be reused in your own custom programs if needed. The following is a list of the locations where both the demo programs and associated libraries can be found on either the ICP DAS web site or the enclosed CD.

Library files download location:

Platform	Web Site Location
WP-9x2x-CE7	ftp://ftp.icpdas.com/pub/cd/winpac_am335x/wp-9000/sdk/pacsdk ftp://ftp.icpdas.com/pub/cd/winpac_am335x/wp-9000/sdk/pacnet
XP-9x7x-WES7	ftp://ftp.icpdas.com/pub/cd/ippc-wes7/sdk/pacsdk/

Platform	CD Location
WP-9x2x-CE7	CD:\SDK\PACSDK CD:\SDK\PACNET
XP-9x7x-WES7	CD:\SDK\PACSDK\

Demo download location:

Platform	Web Site Location
WP-9x2x-CE7	ftp://ftp.icpdas.com/pub/cd/winpac_am335x/wp-9000/demo/pac
XP-9x7x-WES7	ftp://ftp.icpdas.com/pub/cd/ippc-wes7/demo/pacsdk/

Platform	CD Location
WP-9x2x-CE7	CD:\Demo\PAC\
XP-9x7x-WES7	CD:\Demo\PACSDK\

1.7. I-97K I/O module using the DCON protocol command

The 9000 PAC backplane with RS232 interface is only used to access the I-97K Module.

Each 9000 PAC has a corresponding backplane communication port. Therefore, the corresponding COM port number is applied in the API function of the program to access I-97K module through the DCON protocol command.

PAC	Backplane (serial interface)
WP-9000 series	COM0
XP-9000 series	COM1

The PACSDK/PACSDK library provides several UART API functions (uart_Send/uart_Recv…) to communicate with I-97K series I/O module.

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

The user can use UART API to send DCON command to I-97K series modules.

For example:

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

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

uart_Close(hPort);
```

About I-97K commands (DCON protocol), please refer to Chapter 4, "DCON protocol commands.

Chapter 2. I-97K I/O module features

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

2.1. I-97K Watchdog & Safe Value

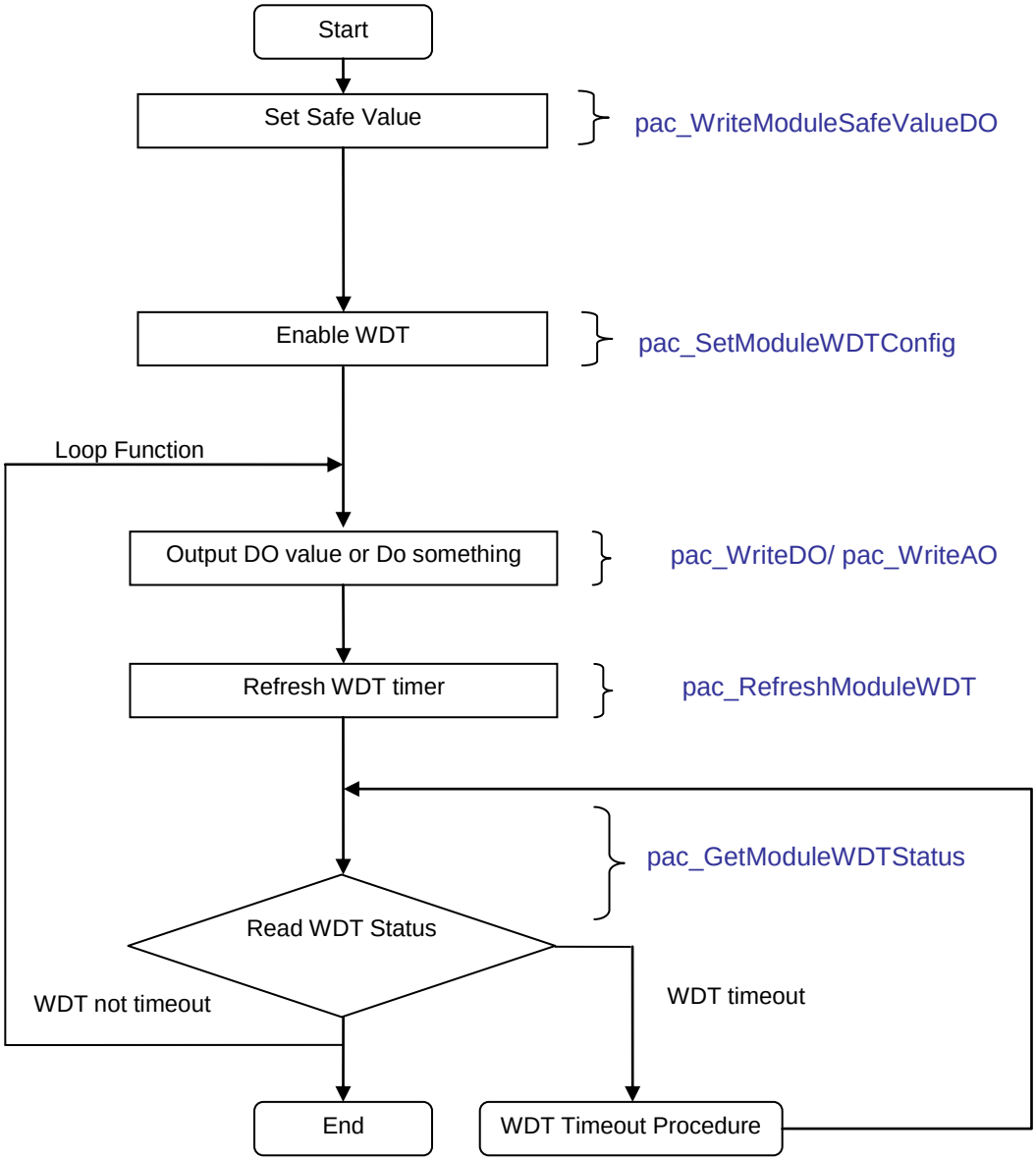
I-97K DO/AO module equip a Hardware Watchdog (WDT) that monitors the operating status of the module. Its purpose is to prevent problems due to host malfunctions, such as situations where the device is affected by noise, or the program is not stable, etc. When the “refresh WDT” function fails and a Watchdog 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.

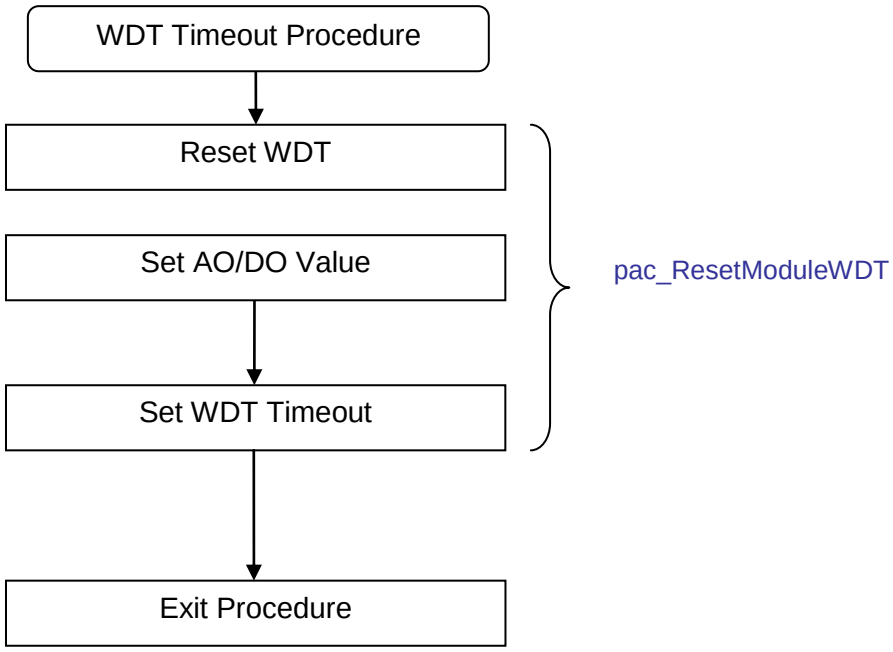
When Watchdog timeout occurs, I-97K output values will all enter Safe Values, the operation for writing output value will not accept until reset Watchdog (Call pac_ResetModuleWDT).

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





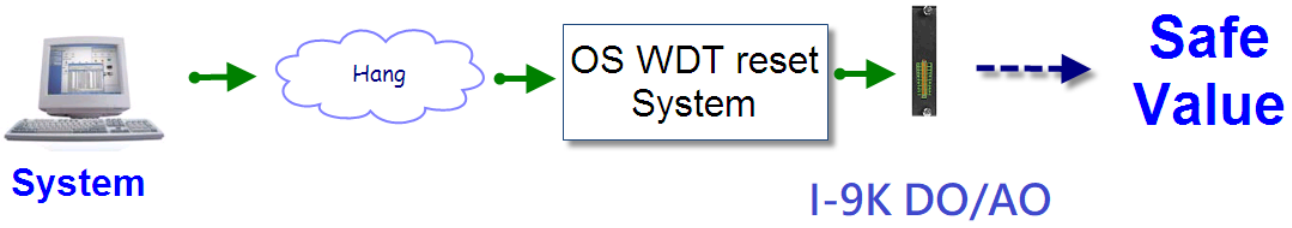
2.2. I-97K DO/AO module usage scenarios

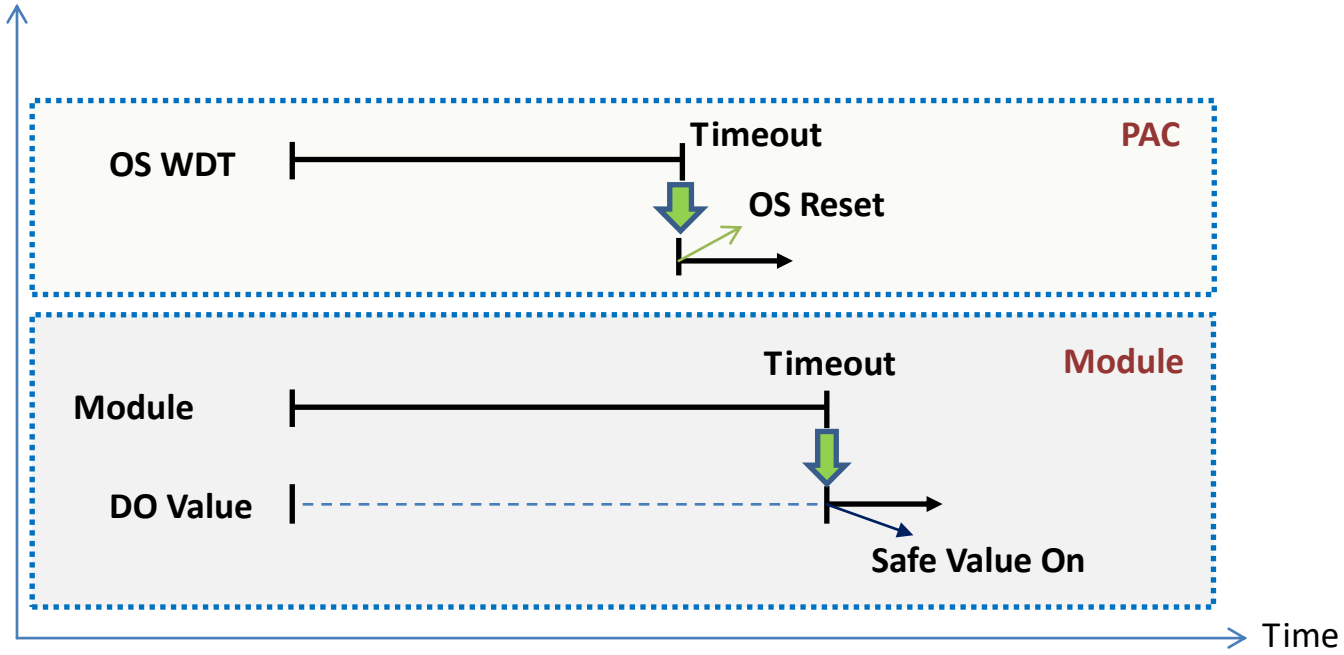
I-9K DO/AO module must be plugged into the 9000 PAC series and the module can function properly. Two type of Watchdog also supported for 9000 PAC series and they are a built-in hardware circuit to monitor the operation of the system and will reset the system if a failure occurs in the hardware or the software.

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.

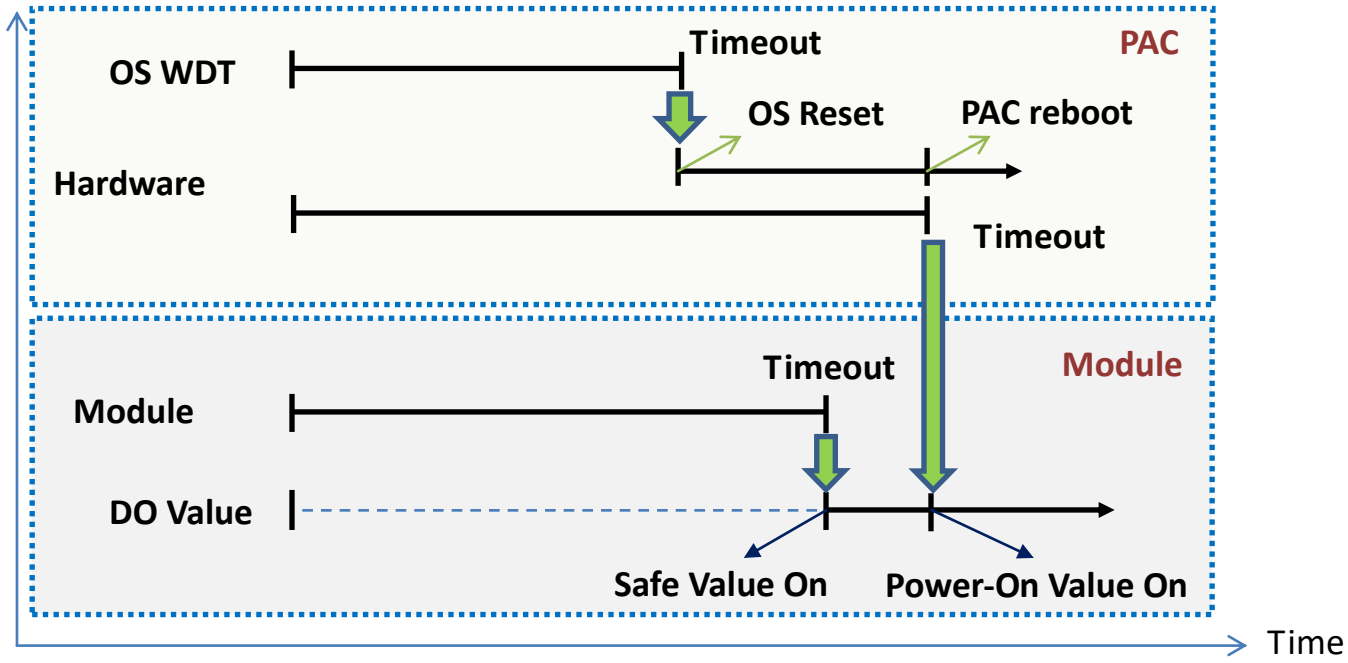




Scenario 2: Module WDT + PAC's Hardware WDT

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

2.3. I-97K AO module features

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.

2.4. I-97K AI module features

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.

Refer to "I-97K Analog Sensor Input Module" table on
http://www.icpdas.com/root/product/solutions/remote_io/i-9k_i-97k/i-9k_i-97k_dio.html
for more details regarding of the I-97K AI module specification.

2.5. I-97K DO module features

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. **PowerOn 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.

2.6. I-97K DI module features

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.

Chapter 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.

3.1. Function List

The common API functions of I-97K I/O Module list as below table. Detailed information related to individual functions can be found in the following sections.

PACSDK Functions	PACNET Functions	Description
pac_WriteDO	PAC_IO.WriteDO	This function writes the DO values to DO modules.
pac_ReadDO	PAC_IO.ReadDO	This function reads the DO value of the DO module.
pac_ReadDI	PAC_IO.ReadDI	This function reads the DI value of the DI module.
pac_ReadDIO	PAC_IO.ReadDIO	This function reads the DI and the DO values of the DIO module.
pac_ReadDILatch	PAC_IO.ReadDILatch	reads the DI latch value of the DI module.
pac_ClearDILatch	PAC_IO.ClearDILatch	clears the latch value of the DI module.
pac_ReadDIOLatch	PAC_IO.ReadDIOLatch	reads the latch values of the DI and DO channels of the DIO module.
pac_ClearDIOLatch	PAC_IO.ClearDIOLatch	clears the latch values of DI and DO channels of the DIO module.
pac_ReadDICNT	PAC_IO.ReadDICNT	reads the counts of the DI channels of the DI module.
pac_ClearDICNT	PAC_IO.ClearDICNT	clears the counter value of the DI channel of the DI module.

pac_WriteAO	PAC_IO.WriteAO	writes the AO value to the AO modules.
pac_ReadAO	PAC_IO.ReadAO	reads the AO value of the AO module
pac_ReadAI	PAC_IO.ReadAI	reads the engineering-mode AI value of the AI module.
pac_ReadAIHex	PAC_IO.ReadAIHex	reads the 2's complement-mode AI value of the AI module.
pac_ReadAIAllExt	PAC_IO.ReadAIAllExt	reads all the AI values of all channels in engineering-mode of the AI module
pac_ReadAIAll	PAC_IO.ReadAIAll	reads all the AI values of all channels in engineering-mode of the AI module.
pac_ReadAIAllHexExt	PAC_IO.ReadAIAllHexExt	reads all the AI values of all channels in 2's complement-mode of the AI module
pac_ReadAIAllHex	PAC_IO.ReadAIAllHex	reads all the AI values of all channels in 2's complement-mode of the AI module.

PACSDK Functions	PACNET Functions	Description
pac_WriteModuleSafeValueDO	PAC_IO.WriteModuleSafeValueDO	This function writes the DO safe values to DO modules
pac_ReadModuleSafeValueDO	PAC_IO.ReadModuleSafeValueDO	This function reads the safe value of the DO modules
pac_WriteModulePowerOnValueDO	PAC_IO.WriteModulePowerOnValueDO	This function writes the DO Power-on values to DO modules
pac_ReadModulePowerOnValueDO	PAC_IO.ReadModulePowerOnValueDOP	This function reads the Power-on value of the DO modules
pac_WriteModuleSafeValueAO	PAC_IO.WriteModuleSafeValueAO	writes the AO safe value to the AO modules.
pac_ReadModuleSafeValueAO	PAC_IO.ReadModuleSafeValueAO	reads the AO safe value of the AO module.

pac_WriteModulePowerOnValueAO	PAC_IO.WriteModulePowerOnValueAO	writes the AO power on value to the AO modules.
pac_ReadModulePowerOnValueAO	PAC_IO.ReadModulePowerOnValueAO	reads the AO power on value of the AO module.
pac_GetModuleLastOutputSource	PAC_IO.GetModuleLastOutputSource	reads the last output source of a module.
pac_GetModuleWDTStatus	PAC_IO.GetModuleWDTStatus	reads the status of watchdog on the module.
pac_GetModuleWDTStatus	PAC_IO.GetModuleWDTStatusEX	reads the status of watchdog on the module.
pac_GetModuleWDTConfig	PAC_IO.GetModuleWDTConfig	reads the status of watchdog on a module.
pac_SetModuleWDTConfig	PAC_IO.SetModuleWDTConfig	This function enables/disables the host watchdog and sets the host watchdog timeout value of a module
pac_ResetModuleWDT	PAC_IO.ResetModuleWDT	resets the host watchdog timeout status of a module.
pac_RefreshModuleWDT	PAC_IO.RefreshModuleWDT	This function refresh the host watchdog of a module

3.2. pac_WriteDO

This function writes the DO values to DO modules.

Prototype

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.

3.3. pac_ReadDO

This function reads the DO value of the DO module.

Prototype

```
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.

3.4. pac_ReadDI

This function reads the DI value of the DI module.

Prototype

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.

3.5. pac_ReadDIO

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

Syntax

C++ for pac_ReadDIO

```
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 in local.

0, if the module is 9k 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).

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.

IDO_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] for pac_ReadDIO

```
// 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#] for pac_ReadDIO

```
// 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.

3.6. 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 in local.

0, if the module is 9K 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).

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.

3.7. 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 in local.

0, if the module is 9K 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.

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.

3.8. 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 in local.

0, if the module is 9K 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).

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_ClearDIOLatch

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

Syntax

C++

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

Parameters

hPort

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

0, if the module is 9K 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.

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.

3.10. pac_ReadDICNT

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

Syntax

C++ for pac_ReadDICNT

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

Parameters

hPort

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

0, if the module is 9K 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).

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] for pac_ReadDICNT

```
// 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] for pac_ReadDICNT_MF

```
// 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#] for pac_ReadDICNT

```
// 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);
```

[C#] for pac_ReadDICNT_MF

```
// 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.

3.11. pac_ClearDICNT

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

Syntax

C++ for pac_ClearDICNT

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

Parameters

hPort

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

0, if the module is 9K 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).

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] for pac_ClearDICNT

```
// 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] for pac_ClearDICNT_MF

```
// 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_MF(hPort, iSlot,iChannel,iDI_TotalCh);
uart_Close(hPort);
```

[C#] for pac_ClearDICNT

```
// 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);
```

[C#] for pac_ClearDICNT_MF

```
// 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_MF(hPort, iSlot,iChannel,iDI_TotalCh);
PACNET.UART.Close(hPort);
```

Remarks

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

3.12. pac_WriteAO

This function writes the AO value to the AO modules.

Syntax

C++ for pac_WriteAO

```
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 in local.

0, if the module is 9K 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).

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] for pac_WriteAO

```
// 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#] pac_WriteAO

```
// 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.

3.13. pac_ReadAO

This function reads the AO value of the AO module.

Syntax

C++

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

Parameters

hPort

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

0, if the module is 9K 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).

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.

3.14. pac_ReadAI

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

Syntax

C++

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

Parameters

hPort

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

0, if the module is 9K 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).

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.

3.15. pac_ReadAIHex

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

Syntax

C++

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

Parameters

hPort

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

0, if the module is 9K 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).

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.

3.16. 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 slot,
    float fValue[],
    DWORD Buff_Len,
    DWORD *Channel
);
```

Parameters

hPort

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

0, if the module is 9K 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).

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.

3.17. pac_ReadAIAll

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

The function maybe causes the buffer overflow in some situation.

Syntax

C++

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

Parameters

hPort

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

0, if the module is 9K 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).

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.

3.18. 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 slot,
    int iValue[],
    DWORD Buff_Len,
    DWORD *Channel
);
```

Parameters

hPort

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

0, if the module is 9K 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).

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.

3.19. pac_ReadAIAllHex

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

The function maybe causes the buffer overflow in some situation.

Syntax

C++

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

Parameters

hPort

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

0, if the module is 9K 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).

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.

3.20. pac_WriteModulePowerOnValueDO

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

Prototype

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

Parameters

hPort

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

3.21. pac_ReadModulePowerOnValueDO

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

Prototype

BOOL pac_ReadModulePowerOnValueDO (

HANDLE hPort, int slot, int iDO_TotalCh, unsigned long *lValue);

Parameters

hPort

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

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

Slot

[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.

3.22. pac_WriteModuleSafeValueDO

This function writes the DO safe values to DO modules.

Prototype

BOOL pac_WriteModuleSafeValueDO(HANDLE hPort, int slot,
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.

3.23. pac_WriteModuleSafeValueAO

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

Syntax

C++

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

Parameters

hPort

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

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.

3.24. pac_ReadModuleSafeValueAO

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

Syntax

C++

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

Parameters

hPort

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

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.

3.25. pac_WriteModulePowerOnValueAO

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

Syntax

C++

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

Parameters

hPort

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

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.

3.26. pac_ReadModulePowerOnValueAO

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

Syntax

C++

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

Parameters

hPort

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

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.

3.27. pac_GetModuleLastOutputSource

This function reads the last output source of a module.

Syntax

C++

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

Parameters

hPort

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

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

.

slot

[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.

3.28. pac_GetModuleWDTStatus

This function reads the status of watchdog on the module.

Syntax

C++

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

Parameters

hPort

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

slot

[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.

3.29. pac_GetModuleWDTConfig

This function reads the status of watchdog on a module.

Syntax

C++

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

Parameters

hPort

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

slot

[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 (only for i-9K)

[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.

3.30. pac_SetModuleWDTConfig

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

Prototype

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 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).

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 (only for i-9K)

[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.

3.31. pac_ReadModuleSafeValueDO

This function reads the safe value of the DO modules.

Prototype

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 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).

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.

3.32. pac_ResetModuleWDT

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

Syntax

C++

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

Parameters

hPort

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

slot

[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.

3.33. pac_RefreshModuleWDT

This function refresh the host watchdog of a module.

Prototype

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.

Chapter 4. DCON protocol commands

So far, the DCON commands for I-87K corresponding to I-97K are listed below.

I-97K module	I-87K module	DCON Command Set link
I-97015	I-87015W	Command Set
I-97017Z	I-87017ZW	Command Set
I-97018	I-87018PW	Command Set
I-97019	I-87019RW	Command Set
I-97024U	I-87024UW	Command Set
I-97028U	I-87028UW	Command Set