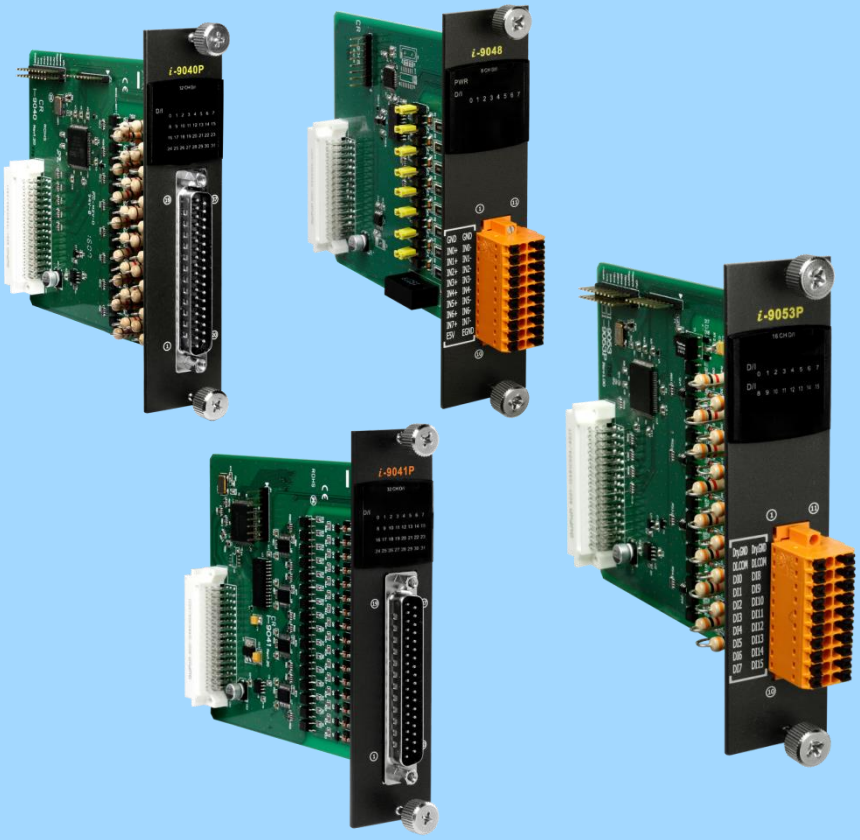
I-9K DIO Module
Common User Manual

V 1.0.2 January 2018



Written by Sean Hsu

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-9K DIO modules are based on a parallel interface with high communication speed and I-9K DIO 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-9K DO modules provide programmable Power-on value / safe value /Retentive functions and All of I-9K DI modules provide DI Low Pass Filter function.

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

- Chapter 1, “Introduction” – This chapter provides information related to the hardware, such as the specifications, the jumper settings details and wiring information.
- Chapter 2, “I-9K DO module features” – This chapter introduces the features of I-9K DO module.
- Chapter 3, “I-9K DI module features” –– This chapter introduces the features of I-9K DI module.
- Chapter 4, “API References” – This chapter describes the functions provided in the I-9K DIO library together with an explanation of the differences in the naming rules used for the different Windows platforms.

1. Introduction

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

I/O module features comparison

Model	I-9K Series	I-8K series
Communication interface	Parallel bus	Parallel bus
Protocol	-	-
Communication speed	Fast	Fast
DI with latched function	-	-
DI with counter input	-	-
Power on value	Y	-
Safe value	Y	-
Programmable slew-rate for AO module	-	-

Now I-9000 DIO series modules include:

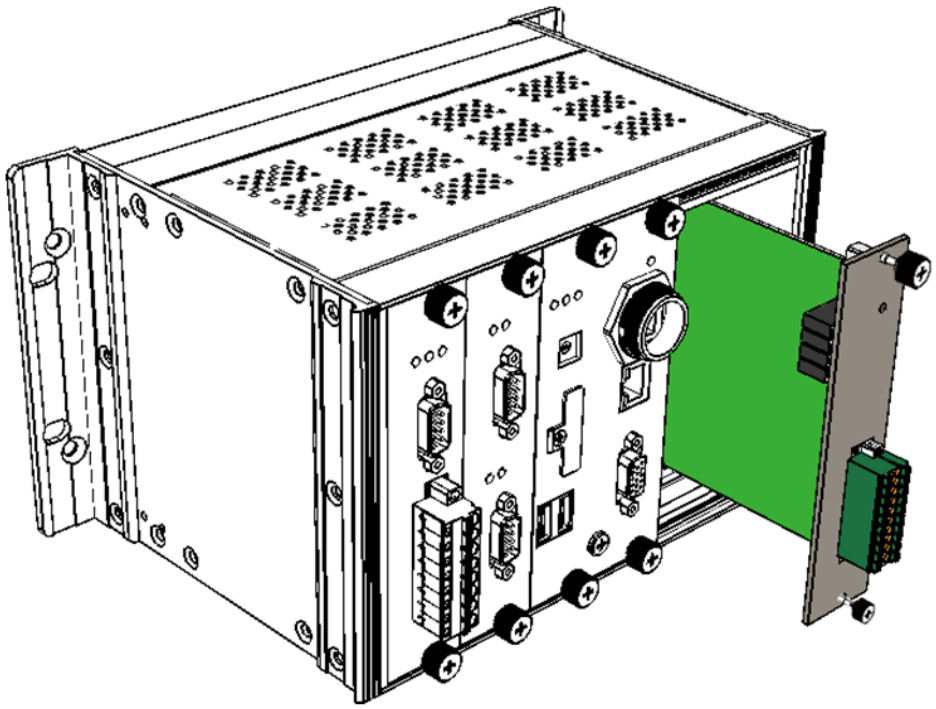
- I-9041P : 32-channel Isolated Digital Output Module
- I-9040P : 32-channel Isolated Digital Input with Low Pass Filter Module
- I-9057P : 16-channel Isolated Digital Output Module
- I-9053P : 16-channel Isolated Digital Input with Low Pass Filter Module
- I-9064 : 8-channe Power Relay Output Module
- I-9048 : 8-channel Digital Input with Interrupt 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, jumper settings details and wiring information.

Those I-9K 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

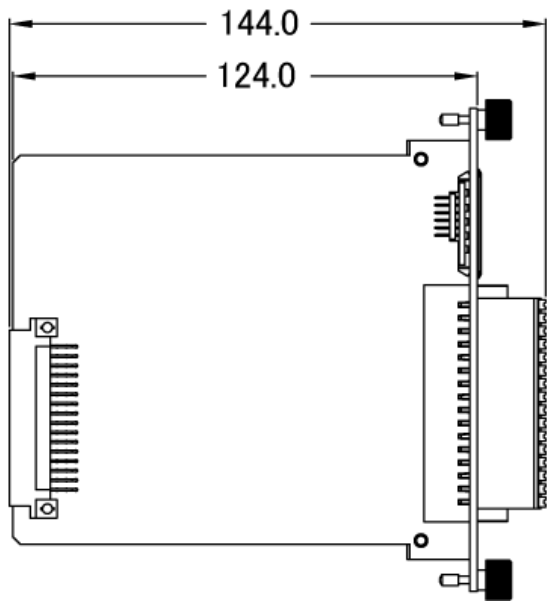
The 9000 PAC series above has expansion slots that enable the addition of optional I/O modules for expanding the capability of the main.



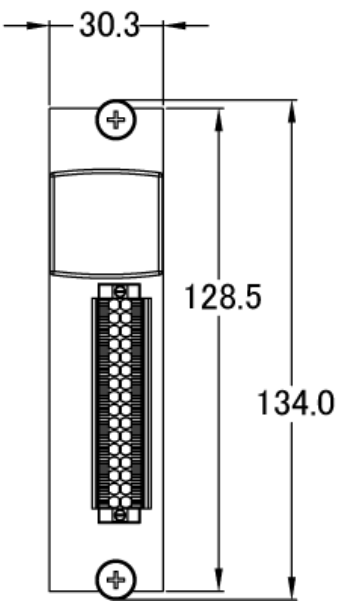
1.1. I/O Module Dimensions

All dimensions are in millimeters.

I-9K module with Spring clamp terminal connector

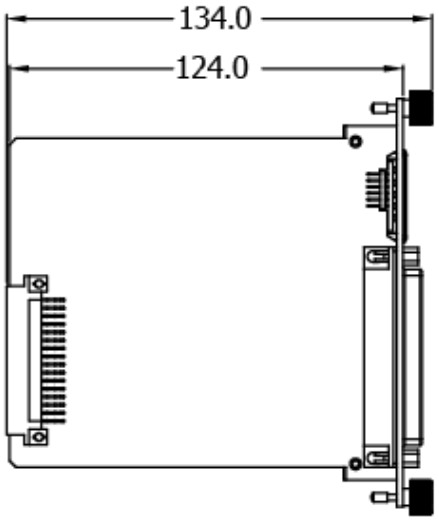


Left Side View

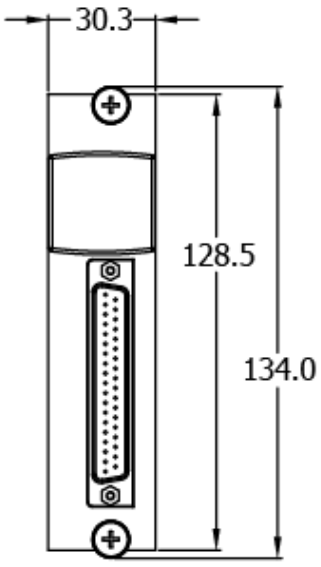


Top View

I-9K module with D-Sub connector



Left Side View

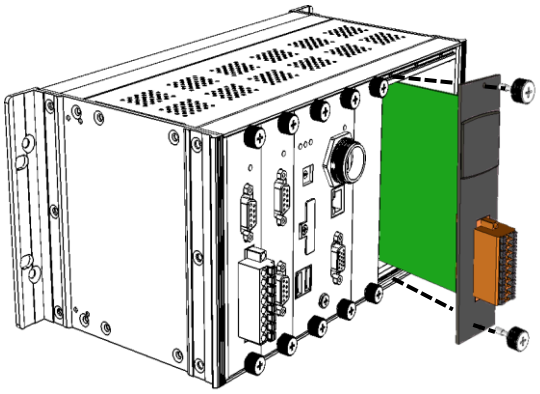


Top View

1.2. Inserting the I/O Modules

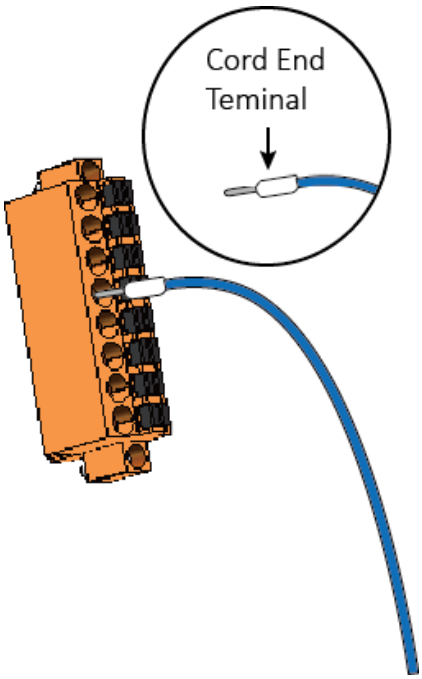
Follow the procedure described below to insert the I/O module.

1. Insert the I/O module



2. Wiring connection

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

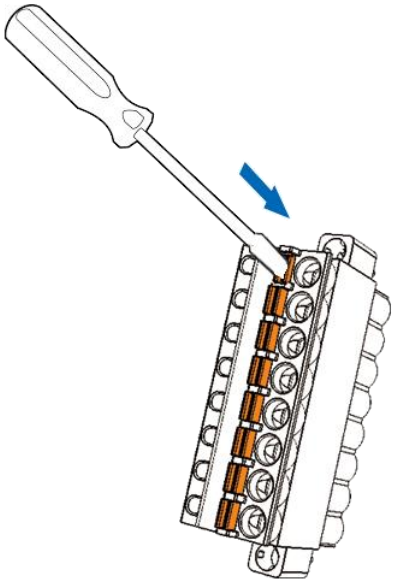


Note:

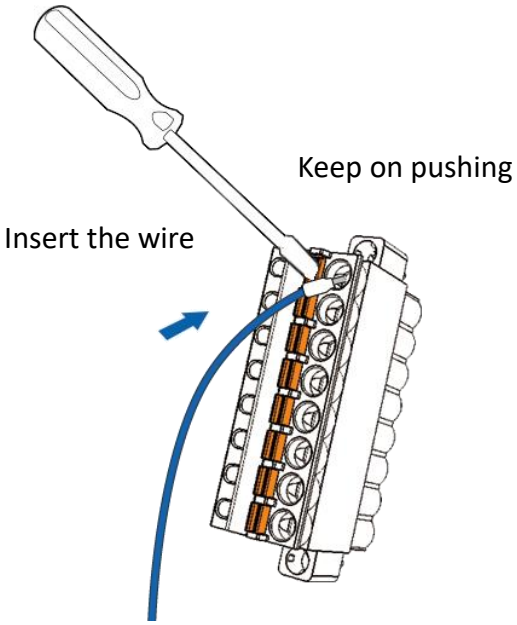
Except I-9040/I-9041 modules, the other I-9K modules support spring clamp terminal connector. The spring clamp terminal connector for the I-9K I/O module connector offers the advantages (anti-vibration, stable clamping and installation easier) relative to screw terminals.

A tip on how to connect the wiring to the connector

1. Use screwdriver to push the orange clip in.

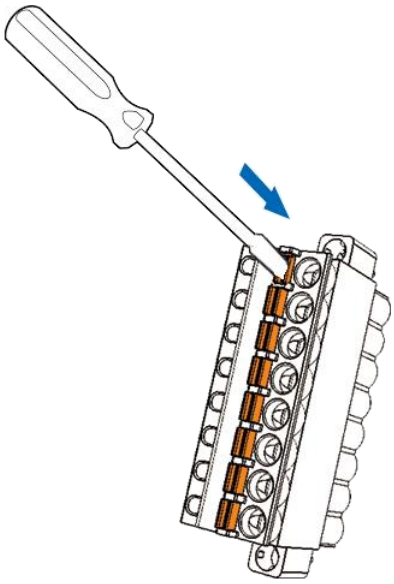


2. Insert the wiring into the terminal block

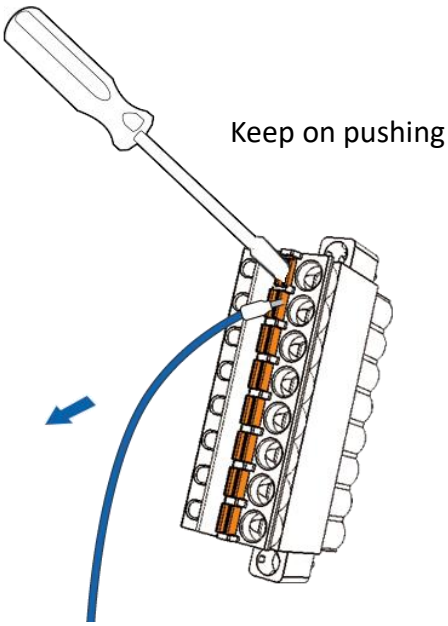


A tip on how to remove the wiring from the connector

1. Use screwdriver to push the orange clip in.



2. Remove the wiring from the terminal block



1.3. 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-9K DIO 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.

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

Platform	CD Location
WP-9x2x-CE7	CD:\SDK\IO\ CD:\Demo\pacsdk\
XP-9x7x-WES7	CD:\SDK\IO\ CD:\Demo\pacsdk\

1.4. I-9K DIO 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:

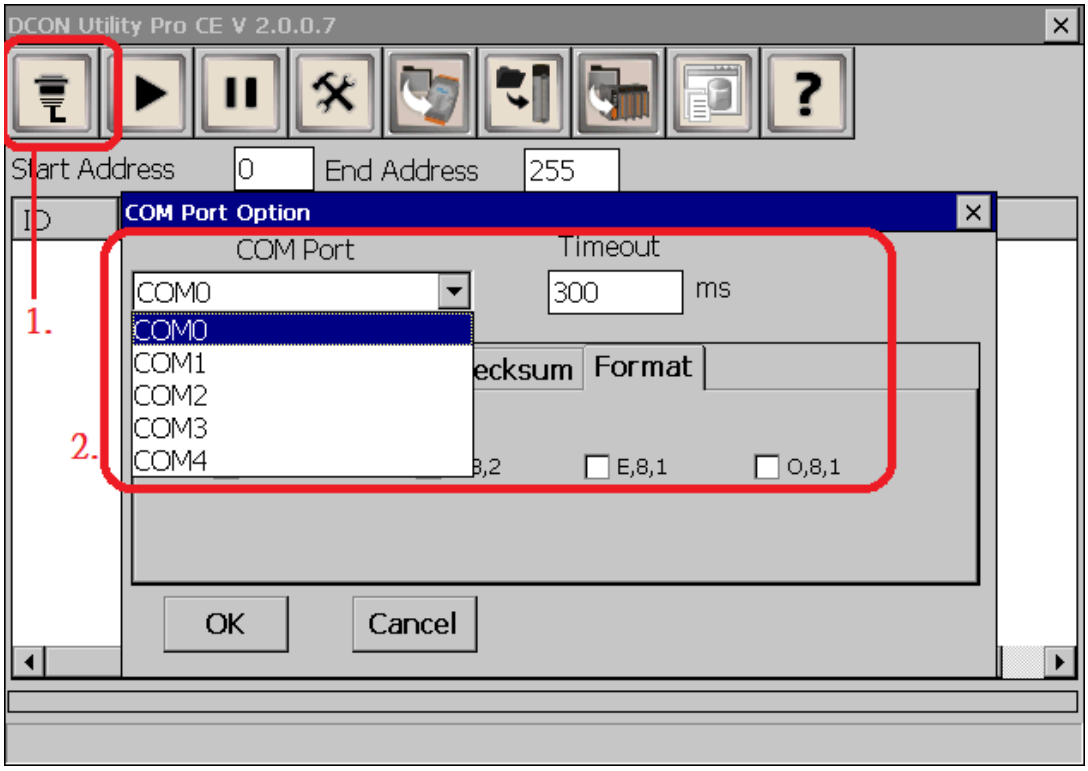
Platform	Location	
WP-9000-CE7	CD	CD:\WinPAC_AM335x\Wp-5231\System_Disk\Tools\DCON_Utility_Pro
	FTP	http://ftp.icpdas.com.tw/pub/cd/winpac_am335x/wp-5231/system_disk/tools/dcon_utility_pro
XP-9000(WES)	CD	CD:\ XPAC\XPAC-Atom\tools\DCON_Utility_pro
	FTP	http://ftp.icpdas.com.tw/pub/cd/xpac-atom/tools/dcon_utility_pro/

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

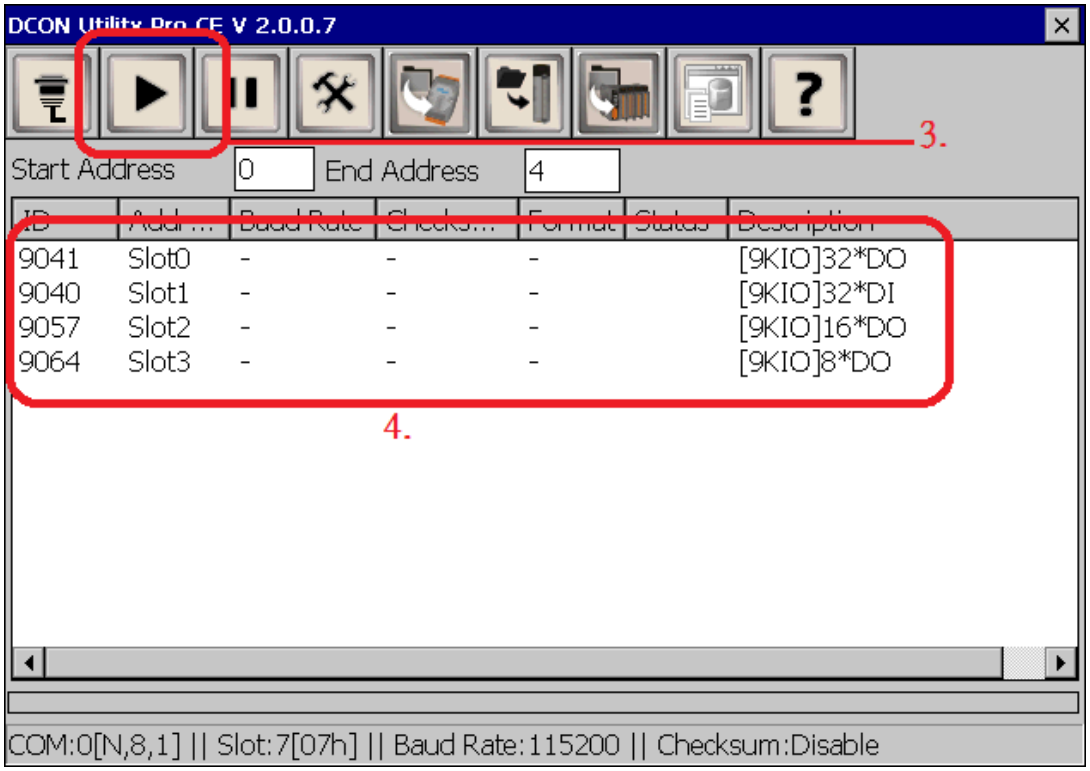
Follow the procedure described below to test the I-9K DIO module.

1. Click the button to open "COM Port Option"

2. Select COM Port and format



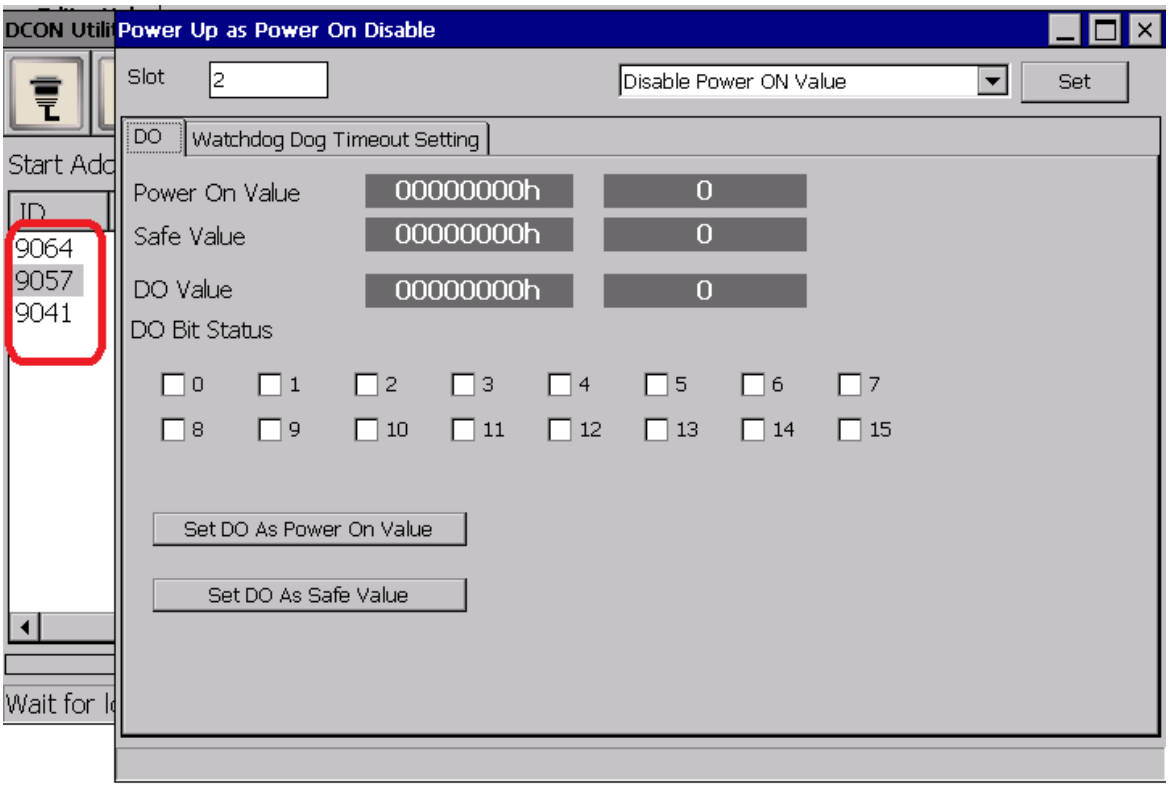
3. Click button to start scan Module



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

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

6. Use the DCON Utility Pro to configure the parameters for DIO, Power-on Mode, WDT and others to quickly test all the I-9K DIO functions, as illustrated in the example below



1.5. I-9K DIO module using the API

ICP DAS provides APIs, libraries and demo programs, including the source code, that allow integration of the I-9K DIO into Windows platforms. For more detailed information regarding the use of these functions, refer to Chapter 4.

The I-9K DO module includes Power-on Mode and Watchdog functions. The three types of Power-on Mode provided by the I-9K DO can be used to prevent unknown errors that might cause the Digital Output from the I-9K DO module to inflict damage on other devices. An introduction to the Power-on Mode is provided in Chapter 2.1.

The I-9K DO module also contains the embedded software Watchdog (WDT) that can be used to prevent problems resulting from host malfunctions. A description of the Watchdog usage and functionality is given in Chapter 2.2

Considering the PAC upgrade and software migration, the number and name for each PACSDK.dll and PACNET.dll function for I-9K module plugged on 9000 PAC series and I-8K 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.

2. I-9K DO module features

The I-9K 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-9K 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-9K DO module integrates over-current, over-voltage and short-circuit functionality.
Compared to the I-8K DO module, I-9K DO module has a programmable power-on value and safe value functions, please refer to Chapter 2.1 more for details.

2.1. Power-on Mode

I-9K DO Module has 3 types of Power-on mode:

Three types of Power-on Mode are provided on the I-9K DO module. Disable Power-on Mode, Power-on Value Mode and Retentive Mode. Below is a description of the conditions in which each mode will apply should an abnormality be encountered that causes the Power-on Mode to be activated:

Mode	Output
Power-on Mode Disabled	The DO value will be set to zero
Power-on Value Mode	The Power-on value will be used as the DO value when the system is reset
Retentive Mode	The previous DO value will be retained

I-9K DO module is reset for any reason, the Digital Output will be activated in Power-on Mode so as to avoid any unknown errors that might cause the Digital Output from the I-9K DO module to inflict damage on other devices. This ensures that the output from the I-9K DO module can be anticipated and guarantees that the output will not damage other devices should the system fail or be reset for any reason.

Note that the I-9K DO module does not support hot plug functionality. When it hot plug on 9000 PAC and 9000 PAC is normal not reset, Power-on Mode will not be affected, it only apply on the others cases caused by PAC power reset and Power-on Mode will be affected

The following is an overview of the DO value that will be set for the I-9K DO module in each of the Power-on Mode conditions:

Mode	Hardware Reset	Software Reset
Power-on Mode Disabled	DO=0	DO= Previous DO value
Power-on Value Mode	DO= Configured Power-on value	DO= Previous DO value
Retentive Mode	DO= Previous DO value	DO= Previous DO value

2.1.1. Disable Power-on Mode

Disable Power-on Mode is used to set DO value to zero after an unknown condition has caused the I-9K DO module to be reset.

Mode	Output*
Disable Power-on Mode	Module will enter zero
Power-on Value Mode	Module will enter Power-on Value
Retentive Mode	Module will keep last value DO value

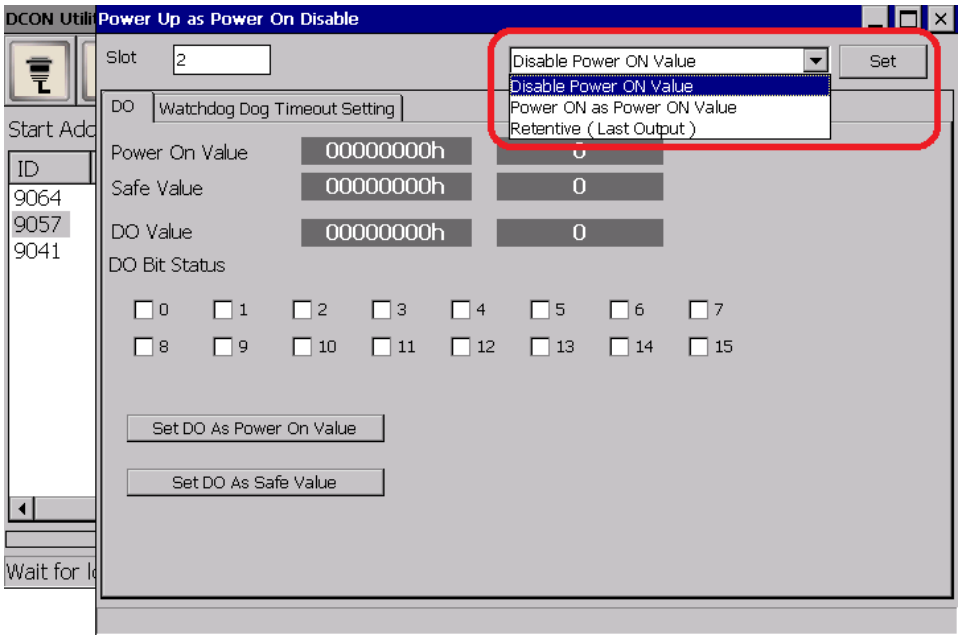
Note:

Above is a description of the conditions in which each mode will apply should an abnormality be encountered that causes the Power-on Mode to be activated:

And I-9K default setting is “Disable Mode”, which can be changed to other modes and set the corresponding output value in DCON Utility or PACSDK API.

This section will show you how to use this pattern:

1. Execute DCON Utility can see default “Disable Power-on Value” setting



2. Then you can reboot the system, the module will output nothing when the Power-on

2.1.2. Power-on Value Mode

Power-on Value Mode is used to set the DO value to the preconfigured Power-on Value after an unknown condition has caused the I-9K DO module to be reset.

Mode	Output*
Disable Power-on Mode	Module will enter zero
Power-on Value Mode	Module will enter Power-on Value
Retentive Mode	Module will keep last value DO value

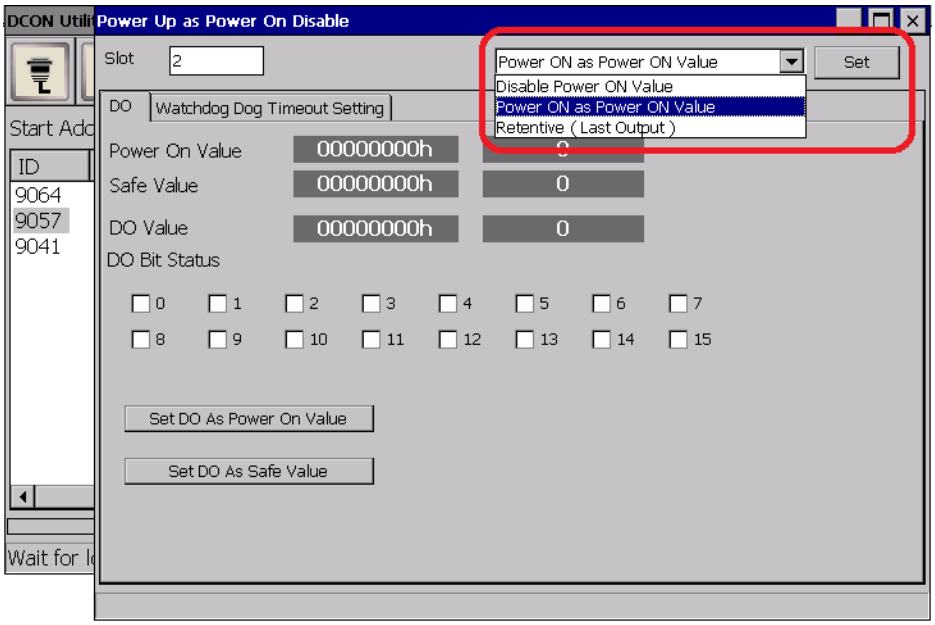
Note:

Above is a description of the conditions in which each mode will apply should an abnormality be encountered that causes the Power-on Mode to be activated:

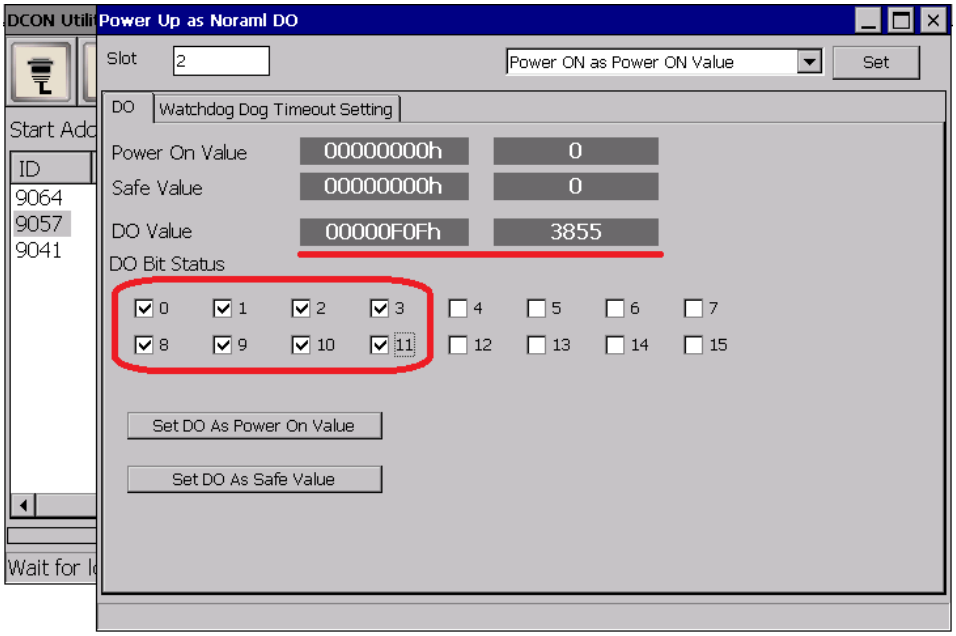
I-9K Power-on Mode which can be changed to other modes or and set the corresponding output value in DCON Utility or PACSDK API.

This section will show you how to use this pattern:

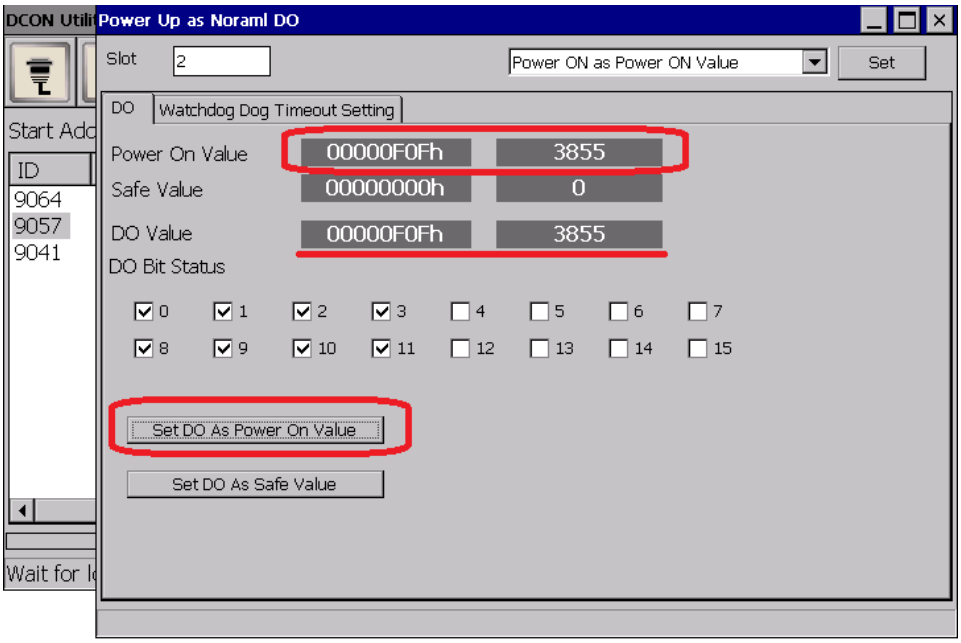
1. Execute DCON Utility and go to module form, then select "Power-on as Power-on Value" mode, after that click "Set" button to set



2. Setting the PowerOnValue, then click “Set Power-on Value” button to set value



3. Select “Read Power-on Value” radio button, then the Power-on Value will show



4. Now you can reboot the system, the module will output the Power-on value when the Power on

2.1.3. Retentive Mode

Retentive Mode, is used to ensure that the previous DO value is retained if an unknown condition has caused the I-9K DO module to be reset

Mode	Output*
Disable Power-on Mode	Module will enter zero
Power-on Value Mode	Module will enter Power-on Value
Retentive Mode	Module will keep last value DO value

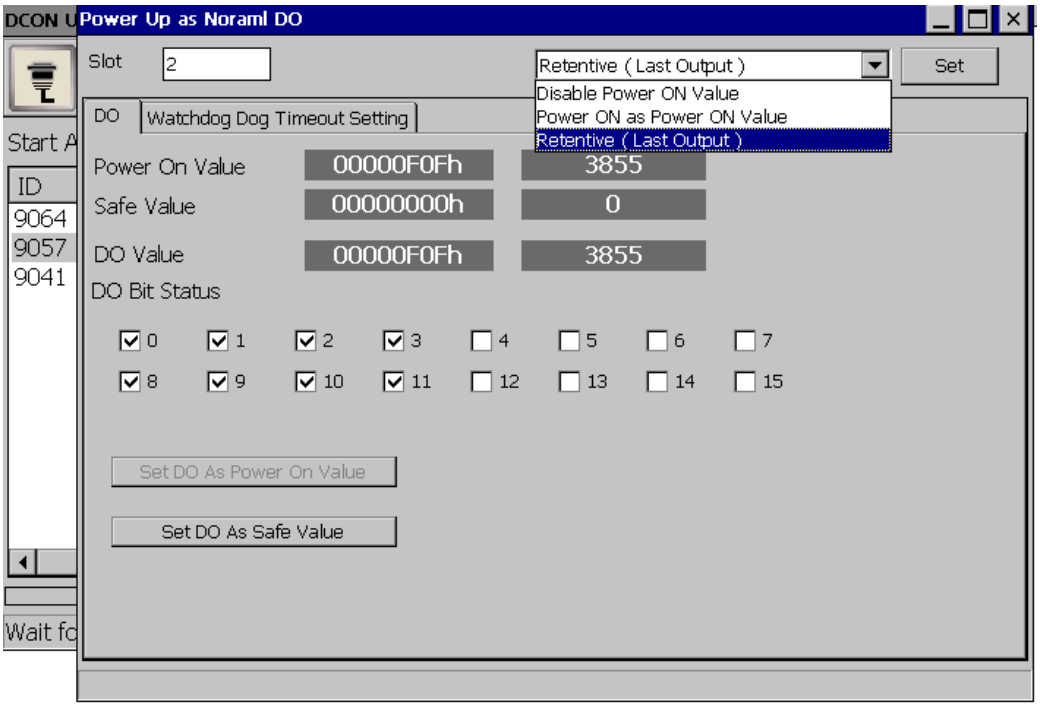
Note:

Above is a description of the conditions in which each mode will apply should an abnormality be encountered that causes the Power-on Mode to be activated:

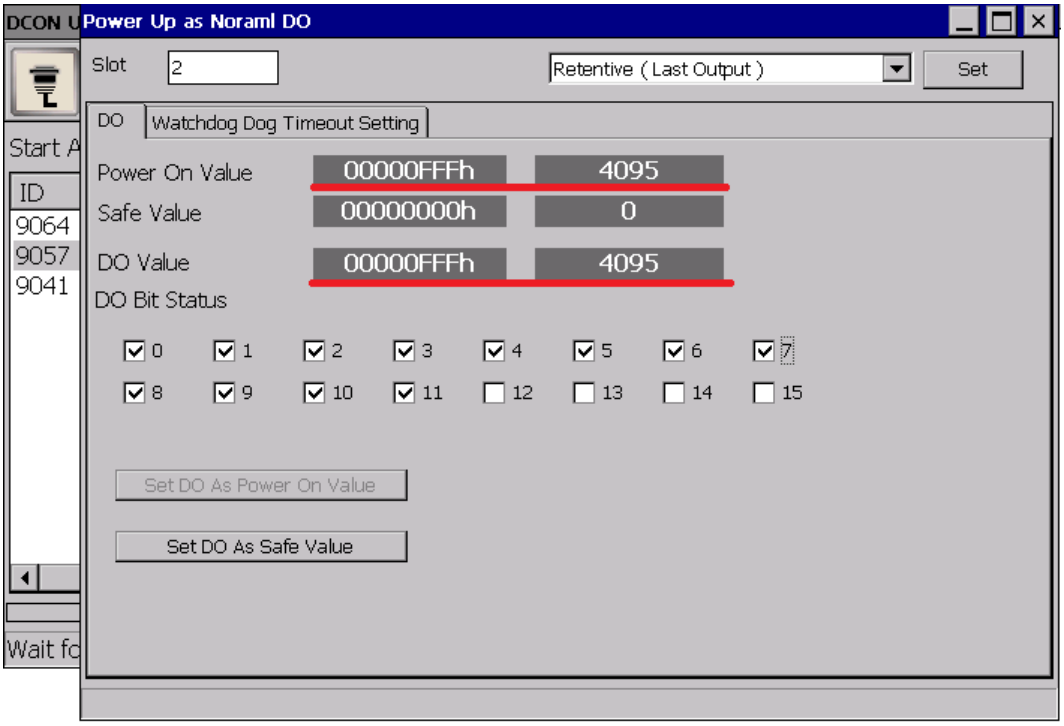
I-9K Power-on Mode which can be changed to other modes or and set the corresponding output value in DCON Utility or PACSDK API.

This section will show you how to use this pattern:

1. Execute DCON Utility and go to module form, then select “Battery Backup Last Output” mode, after that click “Set” button to set



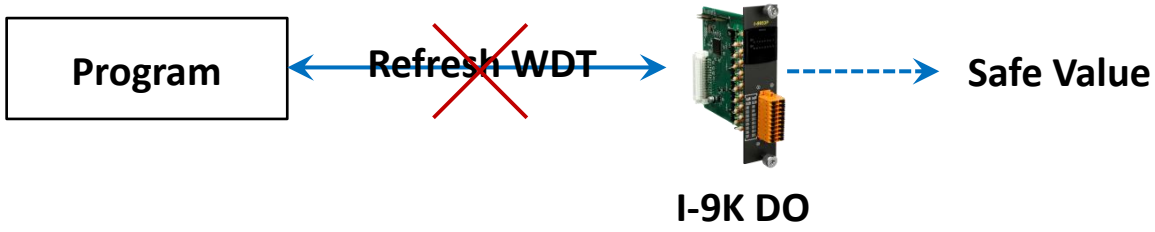
2. Now module will remember last output value



3. When the system reboot, the module will out the value that it remember the last output

2.2. Watchdog & Safe Value

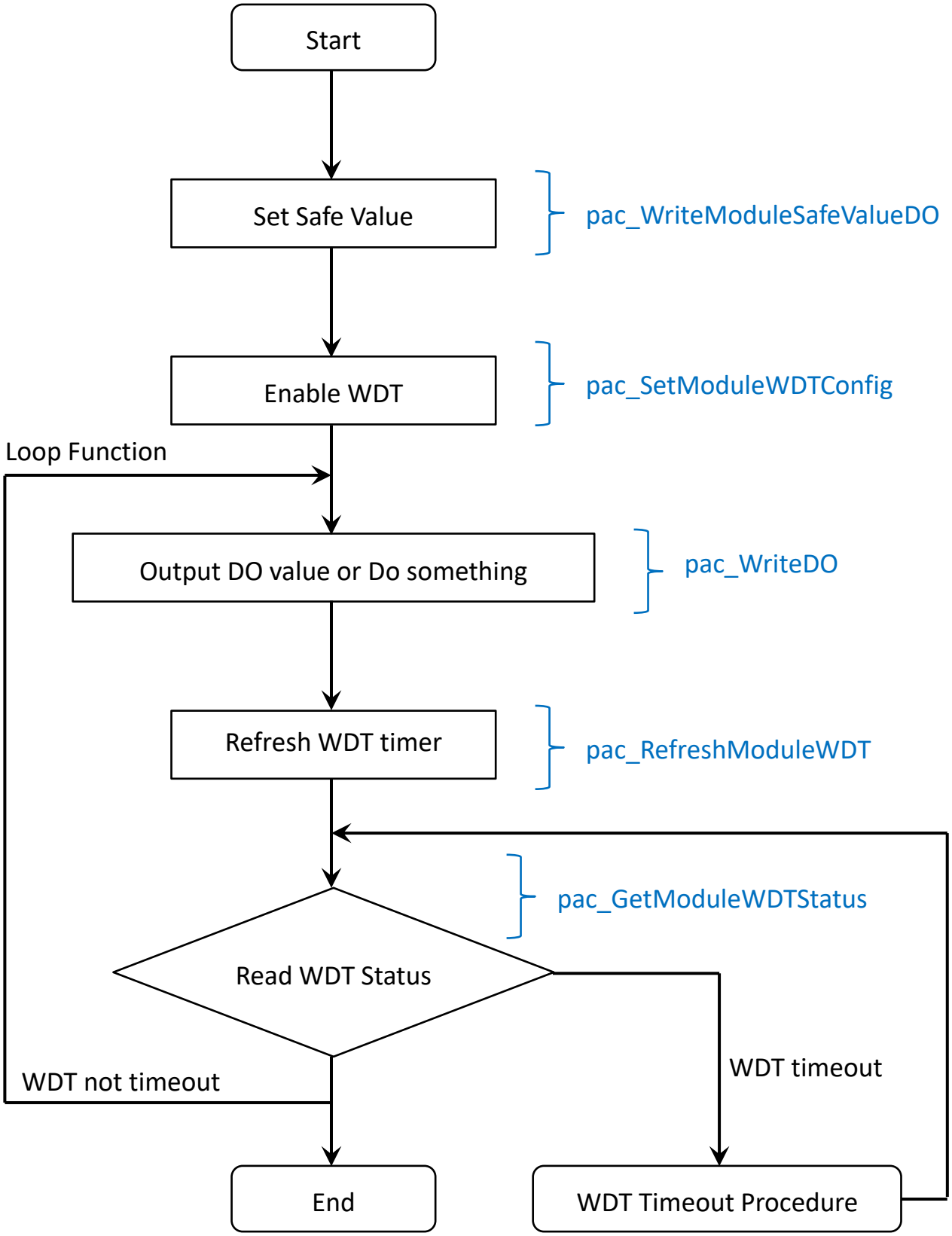
I-9K DO module equips a Hardware Watchdog (WDT) that monitors the operating status of the module. Its purpose is to prevent problems due to host malfunctions, such as situations where the device is affected by noise, or the program is not stable, etc. When the “refresh WDT” function fails and a Watchdog 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.

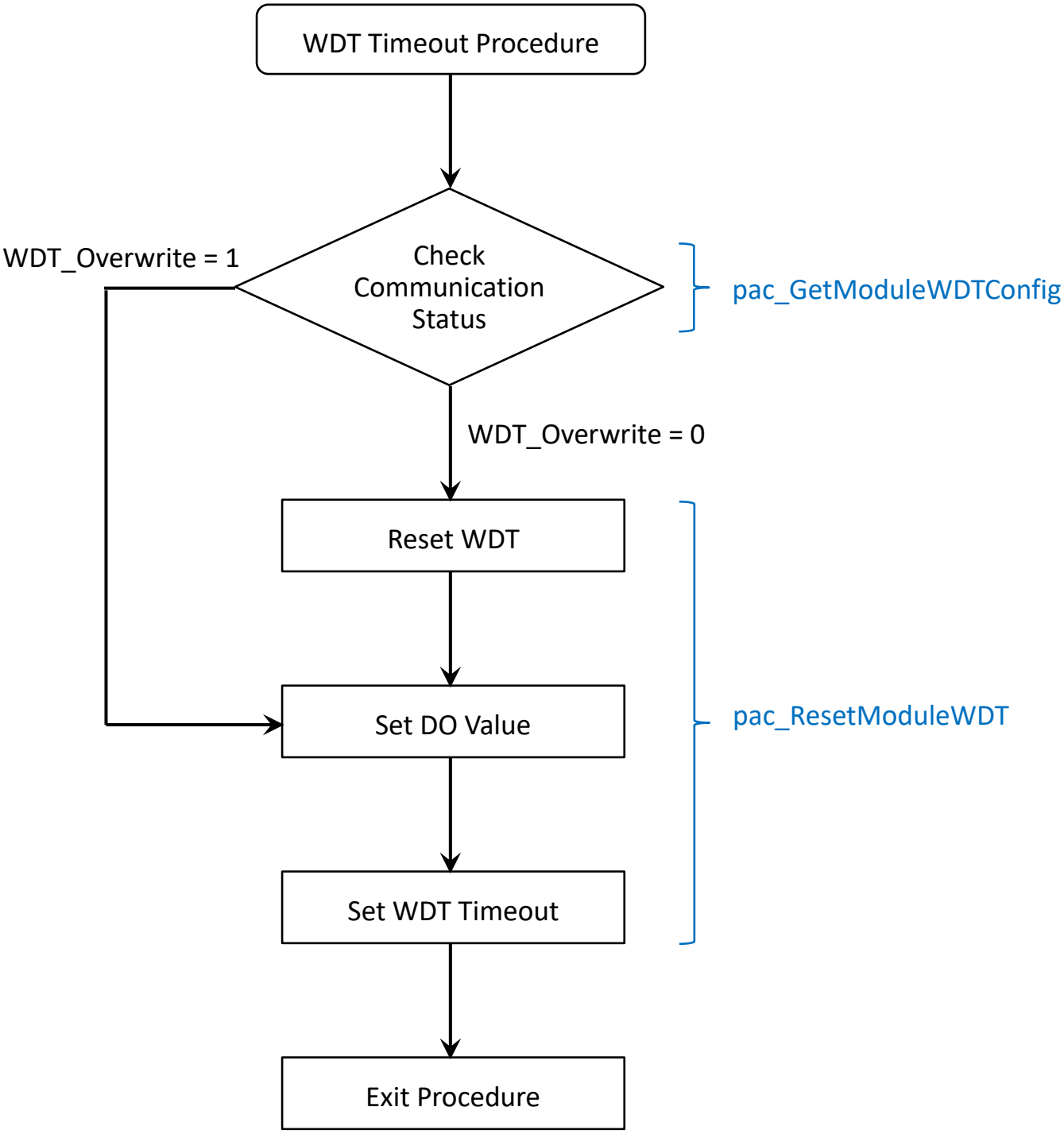


When Watchdog timeout occurs and WDT_Overwrite = 0, I-9K digital output values will all enter Safe Values, the operation for writing output value will not accept until reset Watchdog. If timeout occurs and WDT_Overwrite = 1, writing output value will be accept and Watchdog will be reset by writing output value.

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.

2.2.1. I-9K DO Watchdog procedure

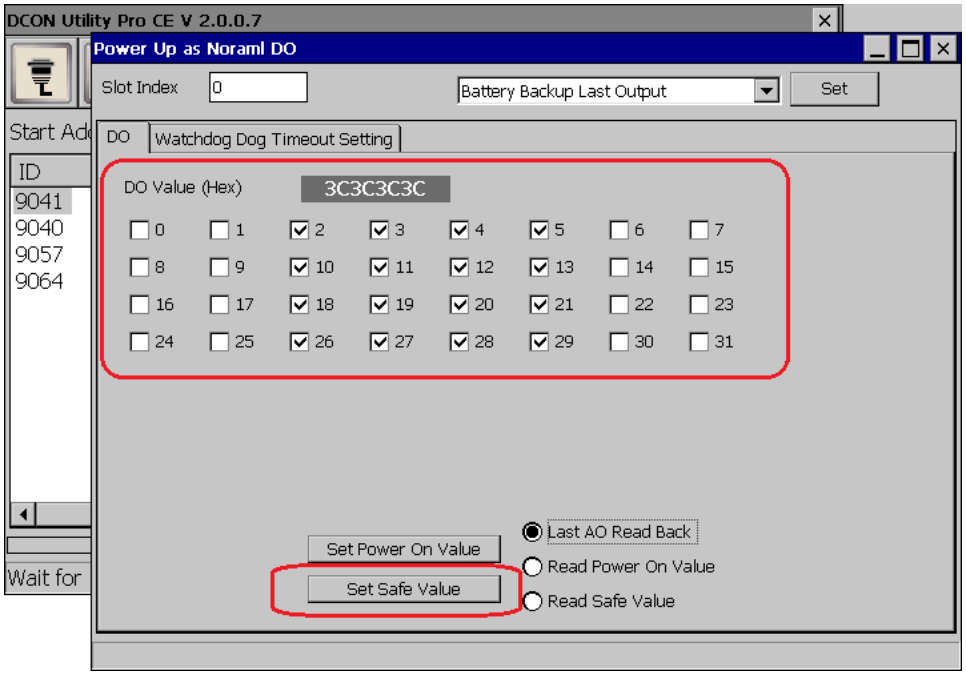



WDT Timeout Procedure
Check Communication Status
WDT_Overwrite = 1
WDT_Overwrite = 0
pac_GetModuleWDTConfig
Reset WDT
Set DO Value
Set WDT Timeout
pac_ResetModuleWDT
Exit Procedure

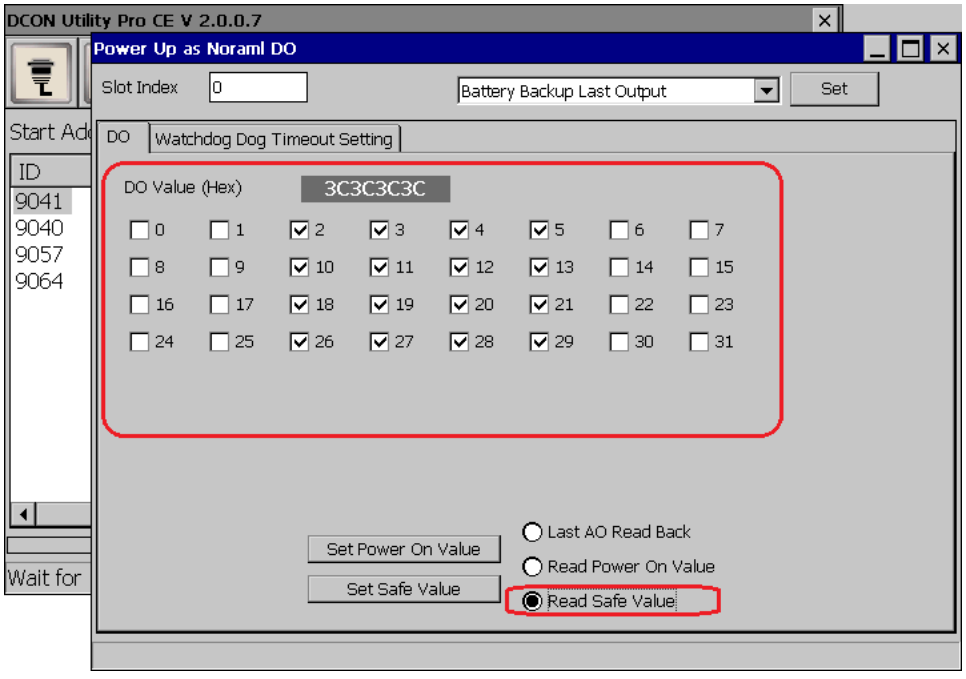
2.2.2. Operate Watchdog & Safe Value

Here show the easy way to use watchdog and how the safe value work by using DCON utility Pro.

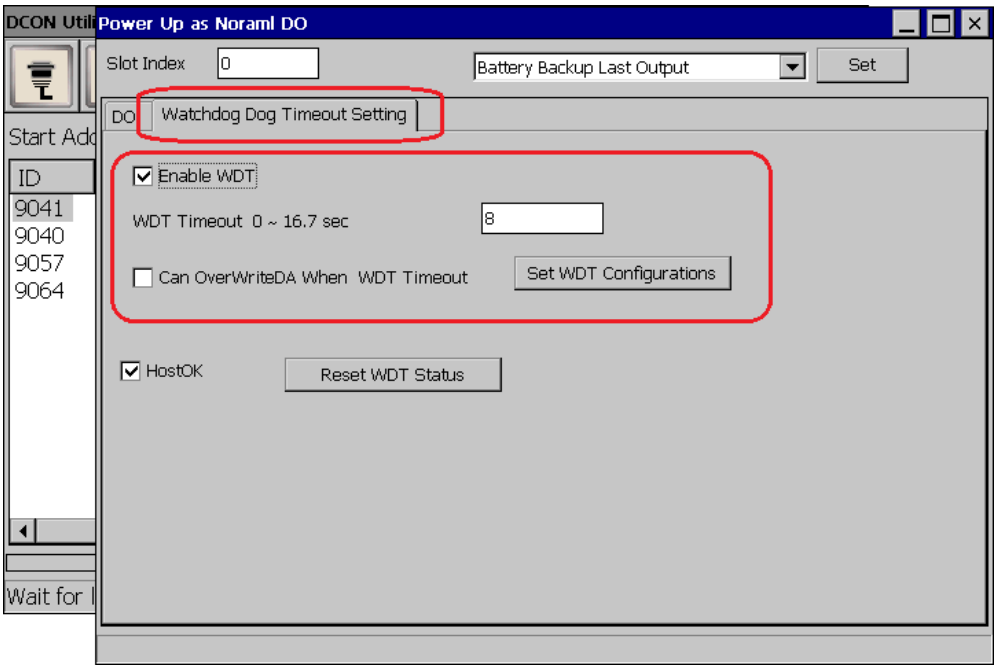
1. Setting the output value then click “Set Safe Value”



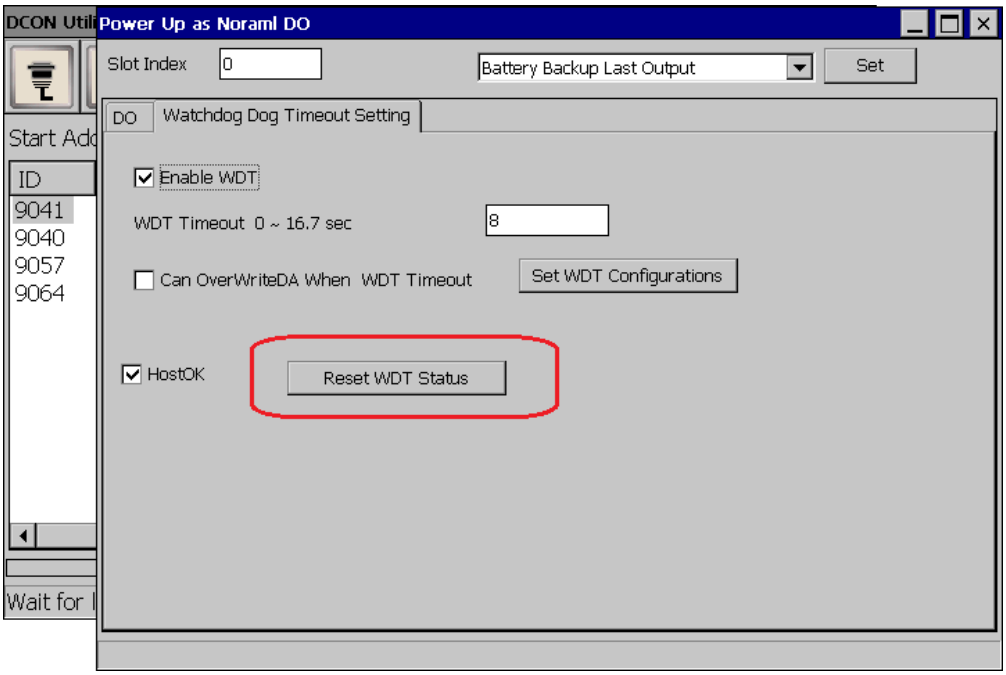
2. Click “Read Safe Value” to check value is right



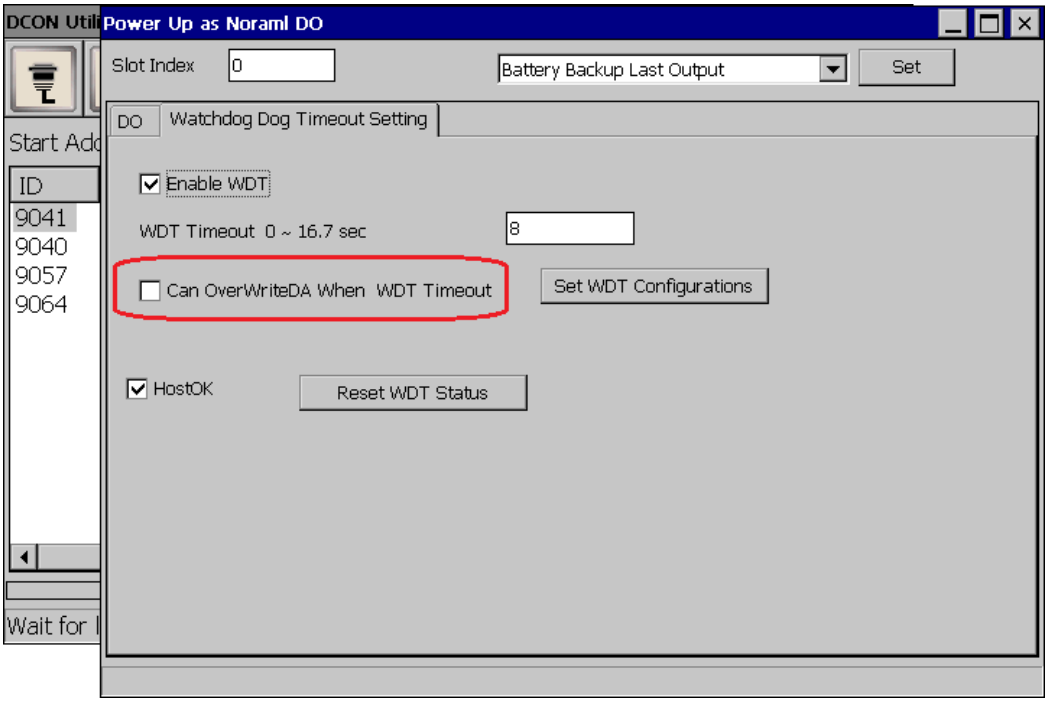
3. Choice “Watchdog Timeout Setting” page, this page can setting enable/disable watchdog, Watchdog timeout second, overwrite when timeout



4. After setting, click ”Set WDT Configurations” to save setting and the Watchdog will start counting until timeout or reset. If WDT timeout, system will reboot without power off, then the module will out save value. But if click “Reset WDT Status” before timeout, then the watchdog will recalculate



5. If "Can OverwriteDA When WDT Timeout" didn't set to WDT configuration, then the module can't output any value when the WDT start counting



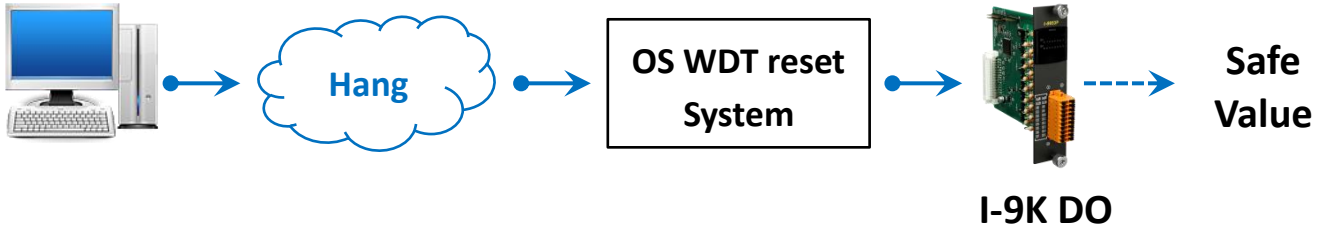
2.3. I-9K DO module usage scenarios

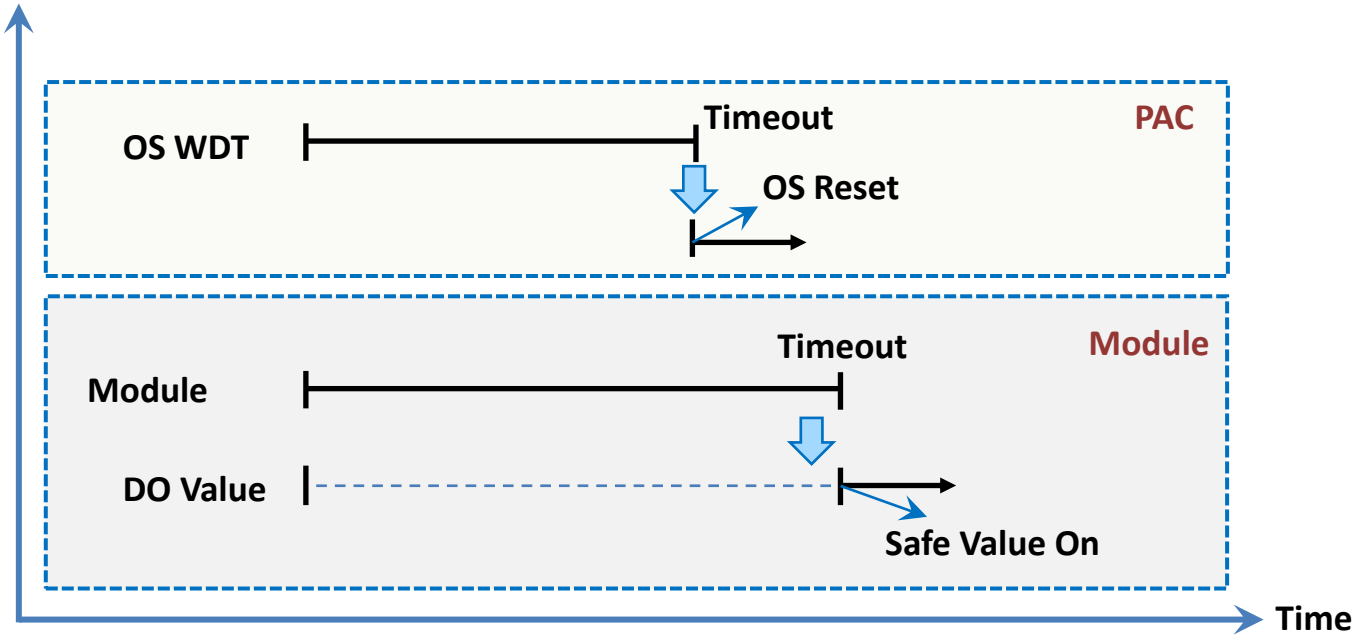
I-9K DO module must be plugged into the 9000 PAC series and the module can function properly. Two type of Watchdog also supported for 9000 PAC series and they are a built-in hardware circuit to monitor the operation of the system and will reset the system if a failure occurs in the hardware or the software.

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

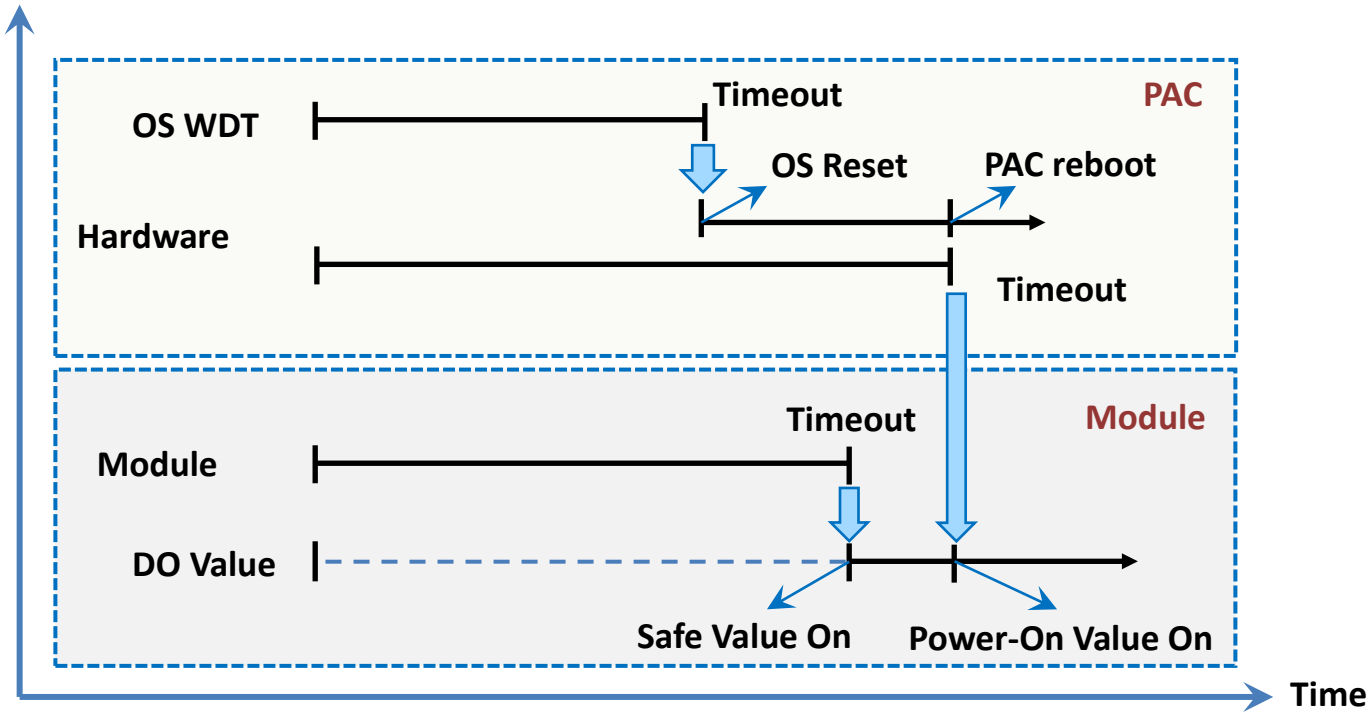
If the PAC system encounters an event that causes it to become unresponsive for any reason, the I-9K DO WDT and the OS WDT provided by 9000 PAC series can be used in combination, thereby preventing the system from hang becoming unresponsive, which may cause the Digital Output to be uncontrolled and result 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-9K DO module to prevent the DO 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-9K DO 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

3. I-9K DI module features

The I-9K 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-9K DI module includes LED indicators are provided for monitoring DI channel status, together with 4 kV ESD protection and 3750 VDC intra-module isolation.

Compared to the I-8K DI module, I-9K DI module has the digital input low pass filter function.

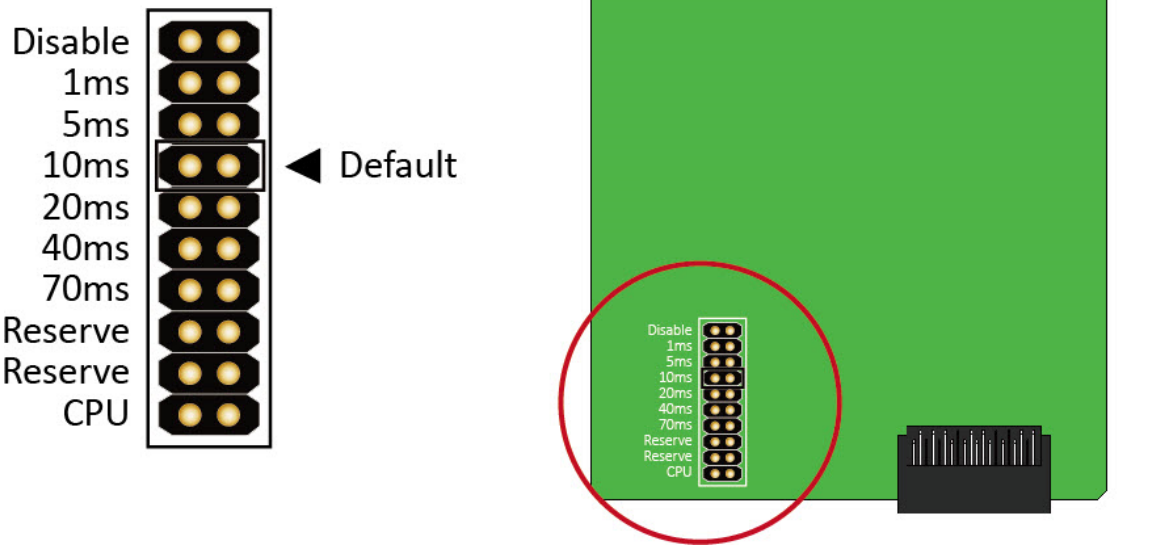
Digital Input Low Pass Filter specification

Digital Input		
Response time	OFF to ON	1ms/5ms/10ms/20ms/40ms/70ms (by Jumper select) or CPU parameter setting
		Default Setting is 10ms
	ON to OFF	1ms/5ms/10ms/20ms/40ms/70ms (by Jumper select) or CPU parameter setting
		Default Setting is 10ms

Low Pass Filter Selection

1. Setting by Hardware (1ms/5ms/10ms/20ms/40ms/70ms by Jumper select)
2. Setting by Software (Set Jumper position to CPU)

The default low pass filter setting of I-9K DI module is 10ms, which can be changed to other modes and set the corresponding output value in DCON Utility or PACSDK API.



4. API References

ICPDAS supplies a range of C/C++ API functions for the I-9K 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 1.3. Location of the Demo and Library Programs.

4.1. Function List

The common API functions of 9K DIO Module list as below table. Detailed information related to individual functions can be found in the following sections.

Systematics	Function	Description
DIO Read/Write	pac_WriteDO	This function writes the DO values to DO modules.
	pac_ReadDO	This function reads the DO value of the DO module.
	pac_ReadDI	This function reads the DI value of the DI module.
	pac_ReadDIO	This function reads the DI and the DO values of the DIO module.
PowerOnValue	pac_GetModulePowerOnEnStatus	This function gets the Power-on Value / Retentive / Disable mode from the DO modules.
	pac_SetModulePowerOnEnStatus	This function set the Power-on Value / Retentive / Disable mode to the DO modules.
	pac_WriteModulePowerOnValueDO	This function writes the DO Power-on values to DO modules.
	pac_ReadModulePowerOnValueDO	This function reads the Power-on value of the DO modules.
WDT	Pac_GetModuleLastOutputSource	Reads the last output source of a module.
	pac_GetModuleWDTStatus pac_GetModuleWDTStatusEX	Reads the module watchdog status of a module.

	pac_ReadModuleSafeValueDO	This function reads the safe value of the DO modules.
	pac_WriteModuleSafeValueDO	This function writes the DO safe values to DO modules.
	pac_GetModuleWDTConfig	Reads the module watchdog status of a module.
	pac_SetModuleWDTConfig	This function enables/disables the module watchdog and sets the module watchdog timeout value of a module.
	pac_ResetModuleWDT	Used to reset the status of watchdog on the DO module.
	pac_RefreshModuleWDT	Used to refresh the status of watchdog on the DO module.
	pac_InitModuleWDTInterrupt	Initializes and enables interrupt of a module watchdog.
	pac_GetModuleWDTInterruptStatus	Reads interrupt status of a module watchdog.
	pac_SetModuleWDTInterruptStatus	Enables/disables interrupt of a module watchdog.

4.2. 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 9k remote

IntPtr hPort = PACNET.UART.Open("");
int iDO_TotalCh = 8;
DWORD iDO_Value = 4; // turn on the channel two
BOOL ret = pac_WriteDO(hPort, 0, 1, iDO_TotalCh, iDO_Value);
PACNET.UART.Close(hPort);
```

[C#]

```
// If the module is 9k local

IntPtr hPort = PACNET.UART.Open("");
int iDO_TotalCh = 8;
uint iDO_Value = 4; // turn on the channel two
bool ret = PACNET.IO.WriteDO(hPort, 0, 1, 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.

4.3. 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 9k remote

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 9k 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.

4.4. 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 9kremote

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 9k 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.

4.5. 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.

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]

Example 1:

```
// 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);
```

Example 2:

```
// If the module is 9k local
BYTE iSlot=1;
int iDI_TotalCh=8;
int iDO_TotalCh=8;
DWORD lDI_Value;
DWORD lDO_Value;
BOOL iRet = pac_ReadDIO(0, iSlot,iDI_TotalCh, iDO_TotalCh, &lDI_Value, &lDO_Value);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
hPort = PACNET.UART.Open("");
byte iSlot=1;
int iDI_TotalCh=8;
int iDO_TotalCh=8;
uint lDI_Value;
uint lDO_Value;
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.

4.6. pac_GetModulePowerOnEnStatus

This function is used to get the current Power-on mode from the DO modules.

[Only used for I-9K DO/I-8041RW module]

Syntax

C++

```
BOOL pac_GetModulePowerOnEnStatus (
    int iSlot,
    short *enStatus
);
```

Parameters

iSlot

[in] The slot in which module is to receive the command. Specifies the slot number (0 - 7)

enStatus

[out] Get the module Power-on mode.

0: Disable Power-on value and Retentive Mode.

1: Enable Power-on value Mode.

2: Enable Retentive Mode.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Example

[C]

```
int iSlot = 0;
short get_mode = 0;
pac_GetModulePowerOnEnStatus(iSlot, &get_mode);
```

[C#]

```
Int iSlot = 0;
Short get_mode = 0;
pac_GetModulePowerOnEnStatus(iSlot, ref get_mode);
```

4.7. pac_SetModulePowerOnEnStatus

This function is used to set the Power-on mode to the DO modules.

[Only used for I-9K DO/I-8041RW module]

Syntax

C++

```
BOOL pac_SetModulePowerOnEnStatus (
    int iSlot,
    short enStatus
);
```

Parameters

iSlot

[in] The slot in which module is to receive the command. specifies the slot number (0 - 7) .

enStatus

[in] Set the module Power-on mode.

0: Disable Power-on value and Retentive Mode.

1: Enable Power-on value Mode.

2: Enable Retentive Mode..

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Example

[C]

```
int iSlot = 0;
short get_mode = 0;
pac_SetModulePowerOnEnStatus(iSlot, get_mode);
```

[C#]

```
Int iSlot = 0;
Short Set_mode = 0;
pac_SetModulePowerOnEnStatus(iSlot, Set_mode);
```

4.8. pac_WriteModulePowerOnValueDO

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

Syntax

C++

```
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.

4.9. pac_ReadModulePowerOnValueDO

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

Syntax

C++

```
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.

4.10. pac_WriteModuleSafeValueDO

This function writes the DO safe values to DO modules.

Syntax

C++

```
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.

4.11. 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 87k local
HANDLE hPort;
int iSlot =0;
int lastOutput=0;
hPort = uart_Open("");
lastOutput = pac_GetModuleLastOutputSource(hPort , iSlot);
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
int iSlot =0;
hPort = PACNET.UART.Open("");
int lastOutput= PACNET.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.

4.12. pac_GetModuleWDTStatus

This function reads the status of watchdog on the module.

Syntax

C++

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

Parameters

hPort

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

slot

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

If the 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 87k 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 87k 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.

4.13. 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 87k 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 87k 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 87k 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.

4.14. pac_SetModuleWDTConfig

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

Syntax

C++

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

Parameters

hPort

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

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.

4.15. 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 87k 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.

4.16. pac_ResetModuleWDT

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

Syntax

C++

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

Parameters

hPort

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

slot

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

If the 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 87k local
HANDLE hPort;
int iSlot =0;
hPort = uart_Open("");
pac_ResetModuleWDT(hPort, iSlot);
uart_Close(hPort);
```

[C#]

```
// If the module is 87k local
IntPtr hPort;
int iSlot =0;
hPort = PACNET.UART.Open("");
PACNET.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.

4.17. pac_RefreshModuleWDT

This function refresh 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 87k 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.

4.18. pac_InitModuleWDTInterrupt

This function initializes and enables interrupt of a module watchdog.

Syntax

C++

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

Parameters

slot

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

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

ft

[in] A call back function..

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

```
int CALLBACK slot_callback_proc()
{
// do something
   return true;
}

int iSlot =0;
pac_InitModuleWDTInterrupt (iSlot, slot_callback_proc);
```

[C#]

```
PACNET.CALLBACK_FUNC slot_callback_proc; //global
int slot_callback_proc()
{
// do something
   return 0;
}

int iSlot =0;
PACNET.IO.InitModuleWDTInterrupt (iSlot, slot_callback_proc);
```

4.19. pac_GetModuleWDTInterruptStatus

This function reads interrupt status of a module watchdog.

Syntax

C++

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

Parameters

slot

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

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

Return Value

Interrupt status.

Examples

[C]

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

[C#]

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

4.20. pac_SetModuleWDTInterruptStatus

This function enables/disables interrupt of a module watchdog.

Syntax

C++

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

Parameters

slot

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

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

enStatus

[in] Interrupt status.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Examples

[C]

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

[C#]

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

Revision History

This chapter provides revision history information to this document.

The table below shows the revision history.

Revision	Date	Description
1.0.0	January 2018	Initial issue
1.0.2	January 2018	Add function list table. Add pac_ReadDIO function.