I-8028U/I-9028U
I/O Module User Manual

V 2.0.0 July 2018

Written by Edward Ku

Edited by Anna Huang

Warranty

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

Warning

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

Copyright

Trademarks

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

Contact Us

If you have any problems, please feel free to contact us.
You can count on us for a quick response.
Email: service@icpdas.com

Table of Contents

1. Introduction

The I-9024U/I-9028U is a 4-channel/8-channel Isolated analog output module that can be used on a 9000 series PAC, and allows a programmable output range on all analog output channels (0 to 5 V, ±5 V, 0 to 10 V, ±10 V, +4 to +20 mA or 0 to +20 mA). Each Analog Output channel can be configured for an individual range, providing an RF immunity level matching that defined by the IEC 61000-4-3 standard. The I-9024U/I-9028U module features channel to-channel isolation as well as 4 kV ESD protection and 1000 VDC intra-module isolation.

And the I-8024U/ I-8028U is also a 4-channel/8-channel Isolated analog output module that can be used on a 8000 series PAC.

Features

Model	I-8024U	I-8028U	I-9024U	I-9028U
Channel	4	8	4	8
Voltage or Current Output	Jumper	Jumper	Software	Software
Range	0 ~ +5 VDC, ±5 VDC, 0 ~ +10 VDC, ±10 VDC, 0 ~ +20 mA, +4 ~ +20 mA			
Resolution	16-bit			
Power-on Value	Yes			
Safe Value	Yes			
System LED Indicator	1 LED as Power Indicator			
Dimension (L x W x H)	129 mm × 31 mm × 114 mm		144 mm x 30.3 mm x 134 mm	

Applicable Platform table

The following table shows which platform the module applies to.

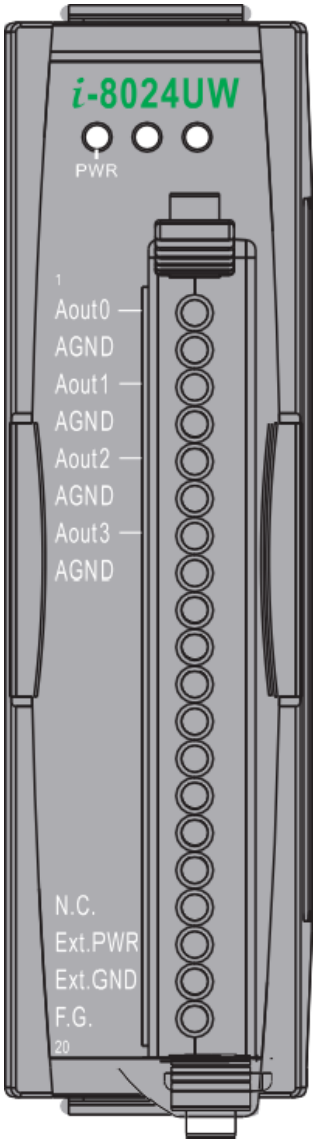
Platform	OS	Module
XPAC	XP-8000(WES)	I-8024UWI-8028UW
	XP-8000-Atom (WES)	I-8024UWI-8028UW
	XP-8000-WES7 (WES7)	I-8024UWI-8028UW
	XP-8000-CE6 (WinCE 6.0)	I-8024UWI-8028UW
	XP-8000-Atom-CE6 (WinCE 6.0)	I-8024UWI-8028UW
	XP-9000-WES7(WES7)	I-9024U/I-9028U
WinPAC	WP-8000 (CE 5.0/7.0)	I-8024UWI-8028UW
	WP-9000-CE7 (CE 7.0)	I-9024U/I-9028U
LinPAC	LinPAC-8000(Linux kernel 3.2/4.4)	I-8024UWI-8028UW
	LinPAC-9000(Linux kernel 3.2/4.4)	I-9024U/I-9028U
IPAC	iPAC-8000 (MiniOS7)	I-8024UWI-8028UW
	I-8000 (MiniOS7)	I-8024UWI-8028UW

1.1. Specifications

Model	I-8024U	I-8028U	I-9024U	I-9028U
Analog Output				
Channels	4	8	4	8
Current Output Wiring	Source			
Range	0 ~ +5 VDC, ±5 VDC, 0 ~ +10 VDC, ±10 VDC, 0 ~ +20 mA, +4 ~ +20 mA			
Resolution	16-bit			
Accuracy	±0.02% of FSR			
Zero Drift	±0.2 μV/°C			
Span Drift	±25 ppm/°C			
Short Circuit Protection	Yes			
Power-on Value	Yes			
Safe Value	Yes			
External Power Requirements				
Reserve Polarity Protection	Yes			
Powered from Terminal Block	Yes, 15 ~ 30 VDC			
Isolation	3000 VDC			
LED Indicators				
System LED Indicator	1 LED as Power Indicator			
Isolation				
Intra-module Isolation, Field-to-Logic	3000 VDC			
EMS Protection				
ESD (IEC 61000-4-2)	±4 kV Contact for each Terminal			
	±8 kV Air for Random Point			
Power				
Power Consumption	2 W Max.			
Mechanical				
Dimension (L x W x H)	129 mm × 31 mm × 114 mm		144 mm x 30.3 mm x 134 mm	
Environment				
Operating Temperature	-25 ~ +75°C			
Storage Temperature	-40 ~ +85°C			
Humidity	10 ~ 90% RH, non-condensing			

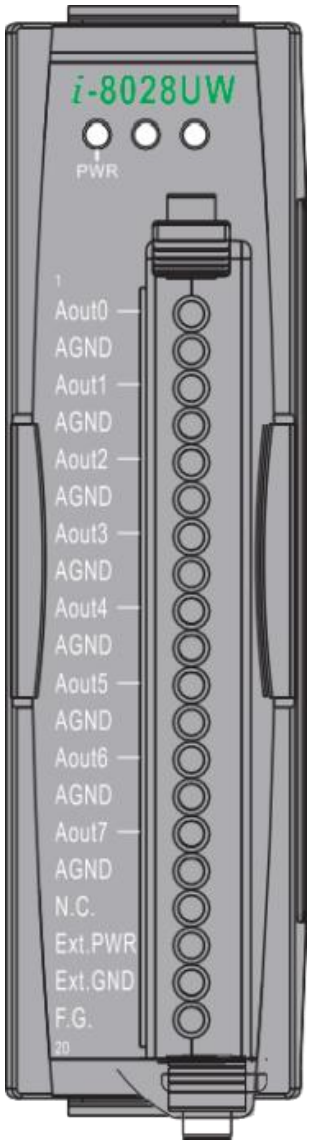
1.2. Pin Assignments

I-8024U



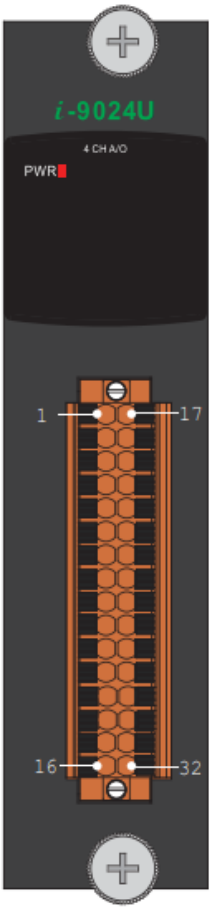
Terminal No.	Pin Assignment
01	Aout0
02	AGND
03	Aout1
04	AGND
05	Aout2
06	AGND
07	Aout3
08	AGND
09	N/A
10	N/A
11	N/A
12	N/A
13	N/A
14	N/A
15	N/A
16	N/A
17	N.C.
18	Ext.PWR
19	Ext.GND
20	F.G.

I-8028U



Terminal No.	Pin Assignment
01	Aout0
02	AGND
03	Aout1
04	AGND
05	Aout2
06	AGND
07	Aout3
08	AGND
09	Aout4
10	AGND
11	Aout5
12	AGND
13	Aout6
14	AGND
15	Aout7
16	AGND
17	N.C.
18	Ext.PWR
19	Ext.GND
20	F.G.

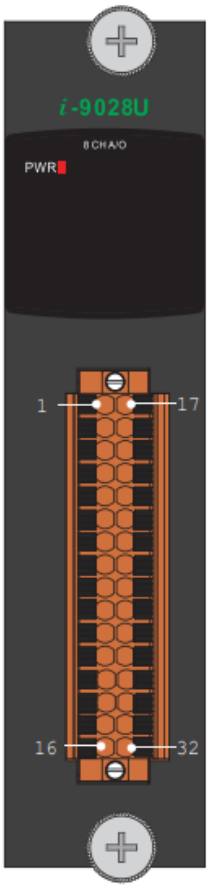
I-9024U



Pin Assignment	Terminal No.		Pin Assignment
Vout0	01	17	Vout1
Iout0	02	18	Iout1
AGND	03	19	AGND
Vout2	04	20	Vout3
Iout2	05	21	Iout3
AGND	06	22	AGND
-	07	23	-
-	08	24	-
-	09	25	-
-	10	26	-
-	11	27	-
-	12	28	-
EXT.PWR	13	29	EXT.PWR
EXT.PWR	14	30	EXT.PWR
EXT.GND	15	31	EXT.GND
EXT.GND	16	32	EXT.GND

32-pin Connector

I-9028U



Pin Assignment	Terminal No.		Pin Assignment
Vout0	01	17	Vout1
Iout0	02	18	Iout1
AGND	03	19	AGND
Vout2	04	20	Vout3
Iout2	05	21	Iout3
AGND	06	22	AGND
Vout4	07	23	Vout5
Iout4	08	24	Iout5
AGND	09	25	AGND
Vout6	10	26	Vout7
Iout6	11	27	Iout7
AGND	12	28	AGND
EXT.PWR	13	29	EXT.PWR
EXT.PWR	14	30	EXT.PWR
EXT.GND	15	31	EXT.GND
EXT.GND	16	32	EXT.GND

32-pin Connector

1.3. Jumper Setting

I-8024U

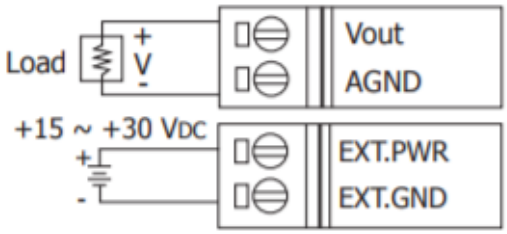
Voltage output

Current output

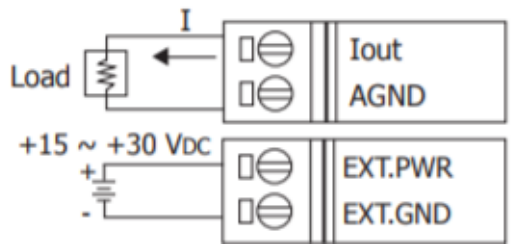
I-8028U

1.4. Wire Connections

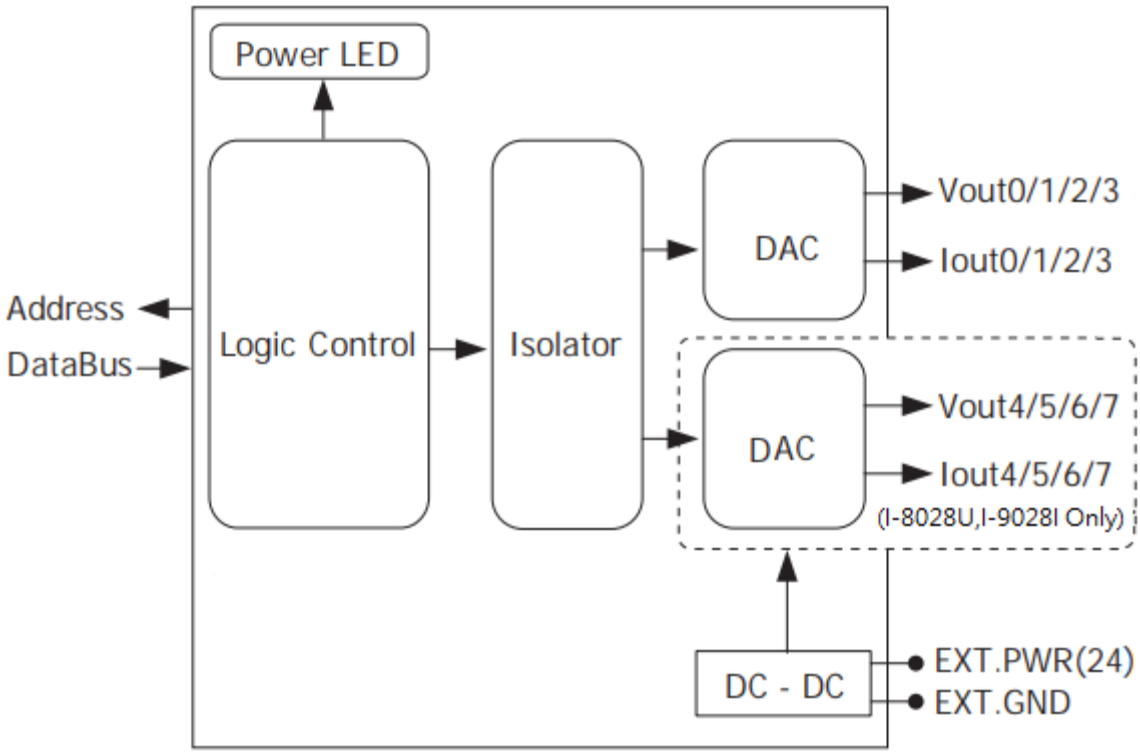
Voltage Output Wiring



Current Output Wiring



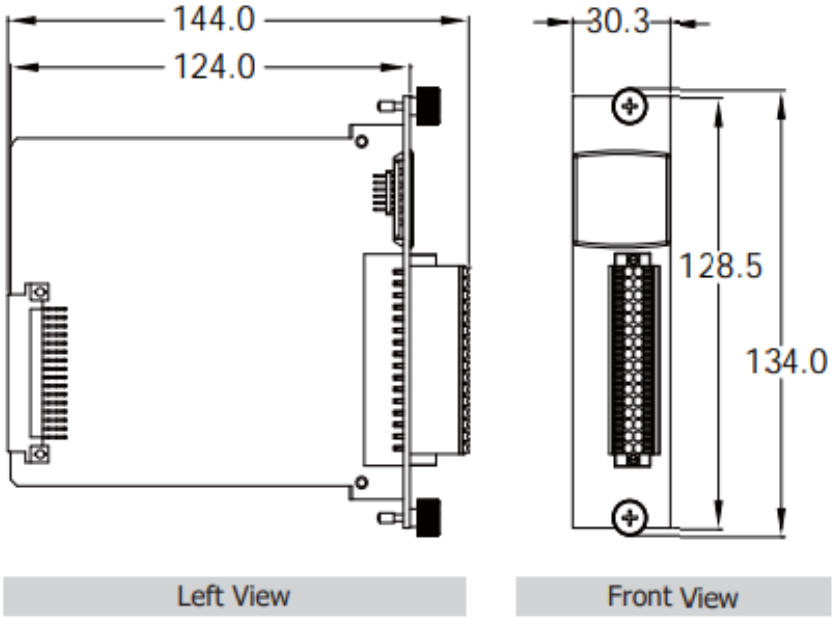
1.5. Block Diagram



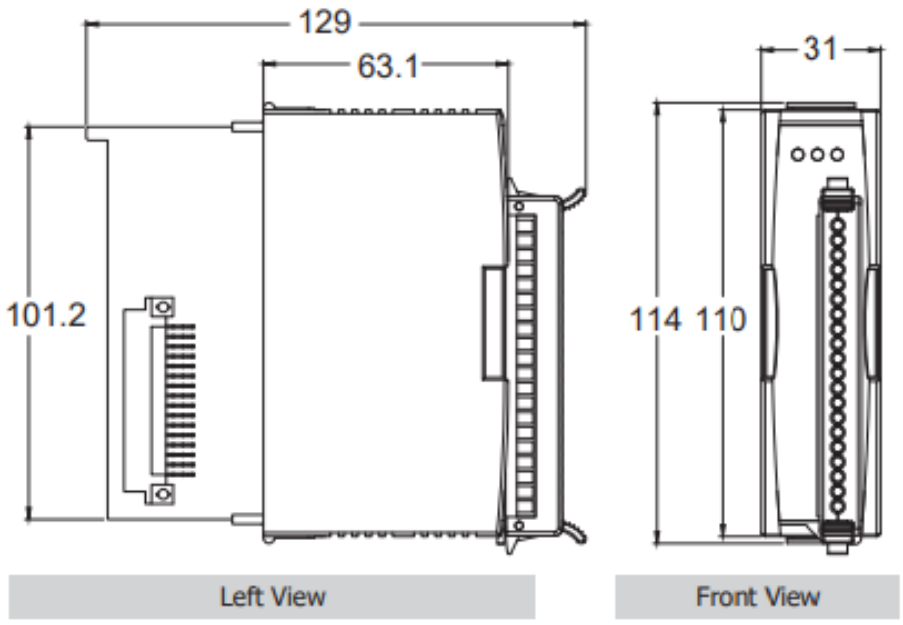
1.6. Dimensions

All dimensions are in millimeters.

I-9024U /I-9028U with Spring clamp terminal connector

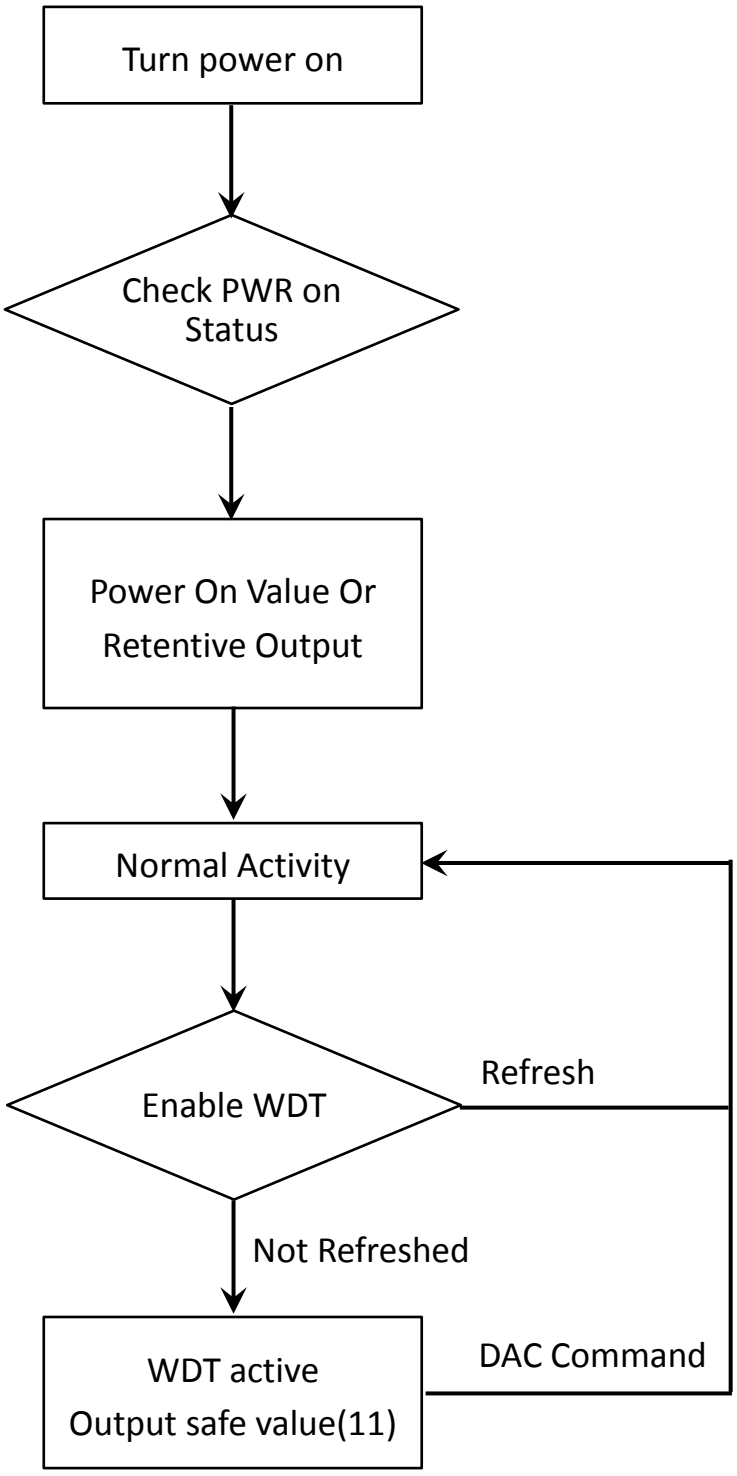


I-8024U /I-8028U with Spring clamp terminal connector



2. Quick Start

The following is an illustration of the operating procedure for the I-8024U/I-8028U/I-9024U/I-9028U module:



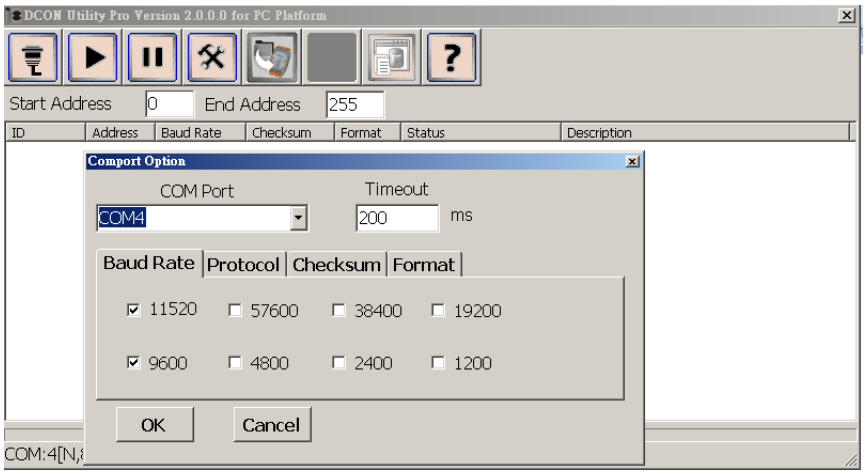
2.1. Getting start with Dcon_Utility

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.

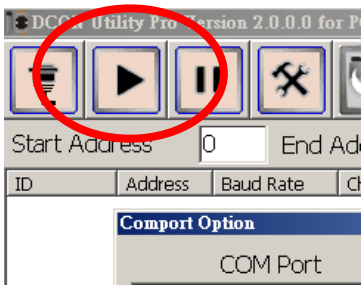
The DCON Utility Pro can be downloaded from the ICP DAS website at:
http://www.icpdas.com/root/product/solutions/software/utilities/dcon_utiltiy_pro.html

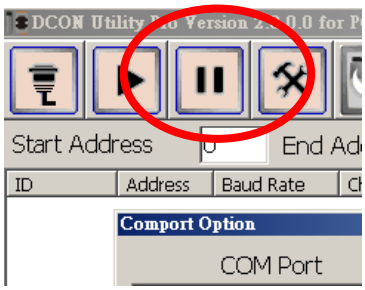
The following operations take I-9028U as a demonstration

1. Launch the DCON Utility and select the appropriate COM Port settings to connect to the I-9028U module

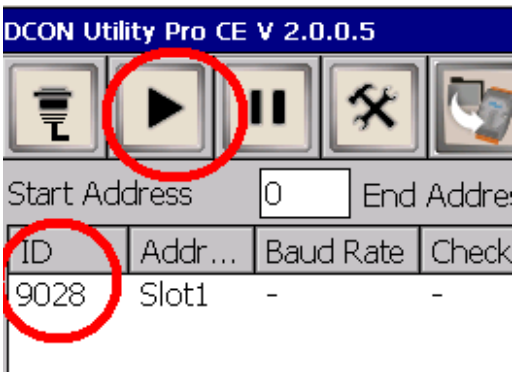


2. Click the “Search” button to start searching for I/O devices connected to the network, and then Click the “Stop Search” button to stop the search.

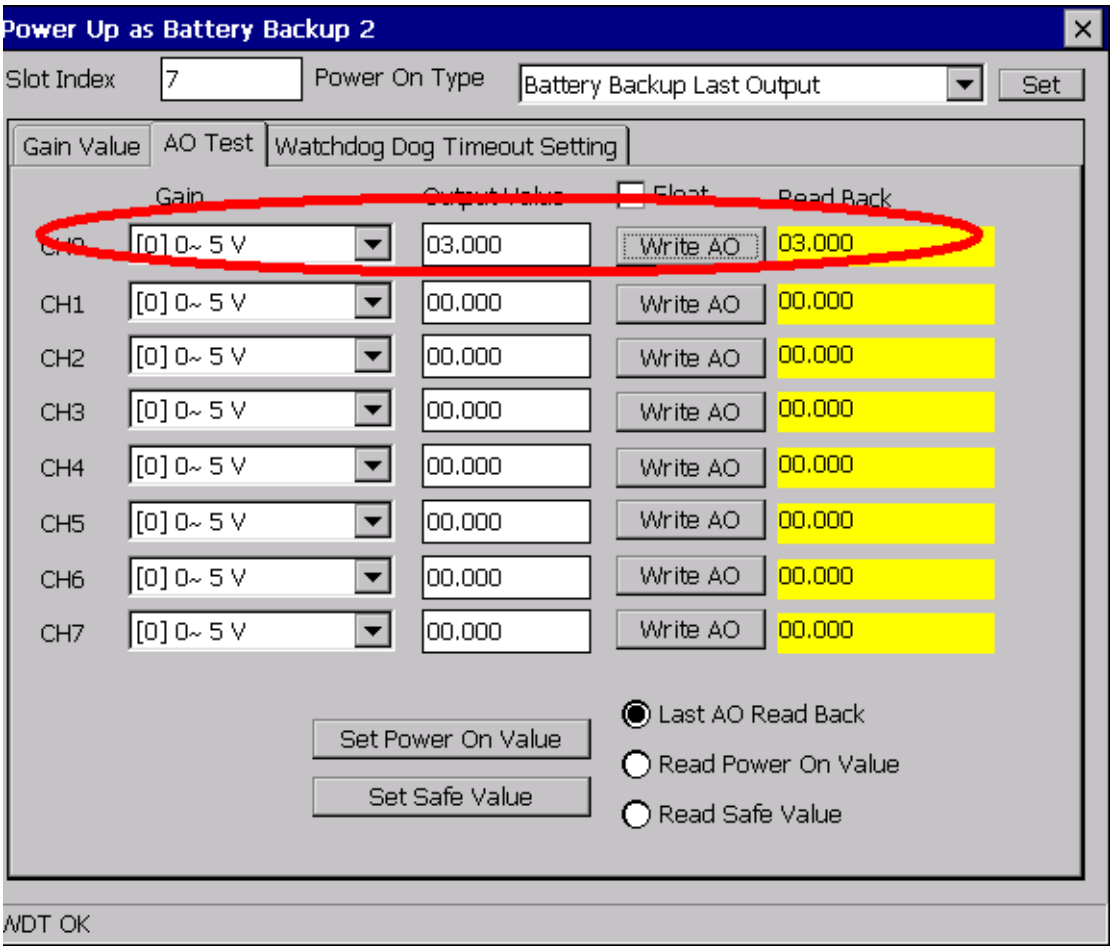




3. If any I/O devices are located, they will be displayed in the device list window. Double-click the name of the module to open the configuration dialog box



4. Use the DCON Utility Pro to configure the parameters for AO, Power-on Mode, WDT and others to quickly test all the I-9028U functions, as illustrated in the example below



2.2. Getting start with API

ICP DAS provides API, library and demo program, including the source code.

The pac_i8028U_init can initialize the module and confirm the hardware ID , and it should be called before using any functions.

The pac_i8028U_WriteAOHex function can be used to directly transmit the Analog Output.

For example:

```
...
pac_i8028U_Init (slot);
...
pac_i8028U_WriteAOHex(slot,ch,gain,hValue);
...
```

The module includes both a Power-on Mode and a WDT that can be used to solve a range of conditions that can make the system become stable.

The two types of Power-on Mode can be used to prevent unknown errors that might cause the Analog Output from the module to inflict damage on other devices Power-on . An introduction to the Power-on Mode is provided in Chapter 3.

The module also contains an embedded software Watchdog (WDT) that can be used to prevent problems resulting from network or communication errors or host malfunctions. A description of the Watchdog usage and functionality is given in Chapter 4.

3. Power-on Mode

If the module is reset for any reason, the Analog Output will be activated in Power-on Mode so as to avoid any unknown errors that might cause the Analog Output from the module to inflict damage on other devices. This ensures that the output from the I-8024U/I-8028U/I-9024U/I-9028U module can be anticipated and guarantees that the output will not damage other devices when the system fail or be reset for any reason.

Two types of Power-on Mode are provided on the I-8024U/I-8028U/I-9024U/I-9028U module.

1. Power-on Value Mode
2. Retentive Mode

Power On mode does not support hot plug functionality. When hot plug on PAC Power On mode will not be affected, it only apply on the others cases caused by PAC power reset.

Configuring Power-on Mode

Power-on Mode included Power-on Value Mode and Retentive Mode. Below is a description of the conditions in each mode will apply when an abnormality be encountered and causes the Power-on Mode to be activated.

Mode	Output
Power-on Value Mode	Ser Power On as Power on Value and reset, then the module will output as configured power on value.
Retentive Mode	Ser Power On as Retentive and reset, then the module will output as late output.

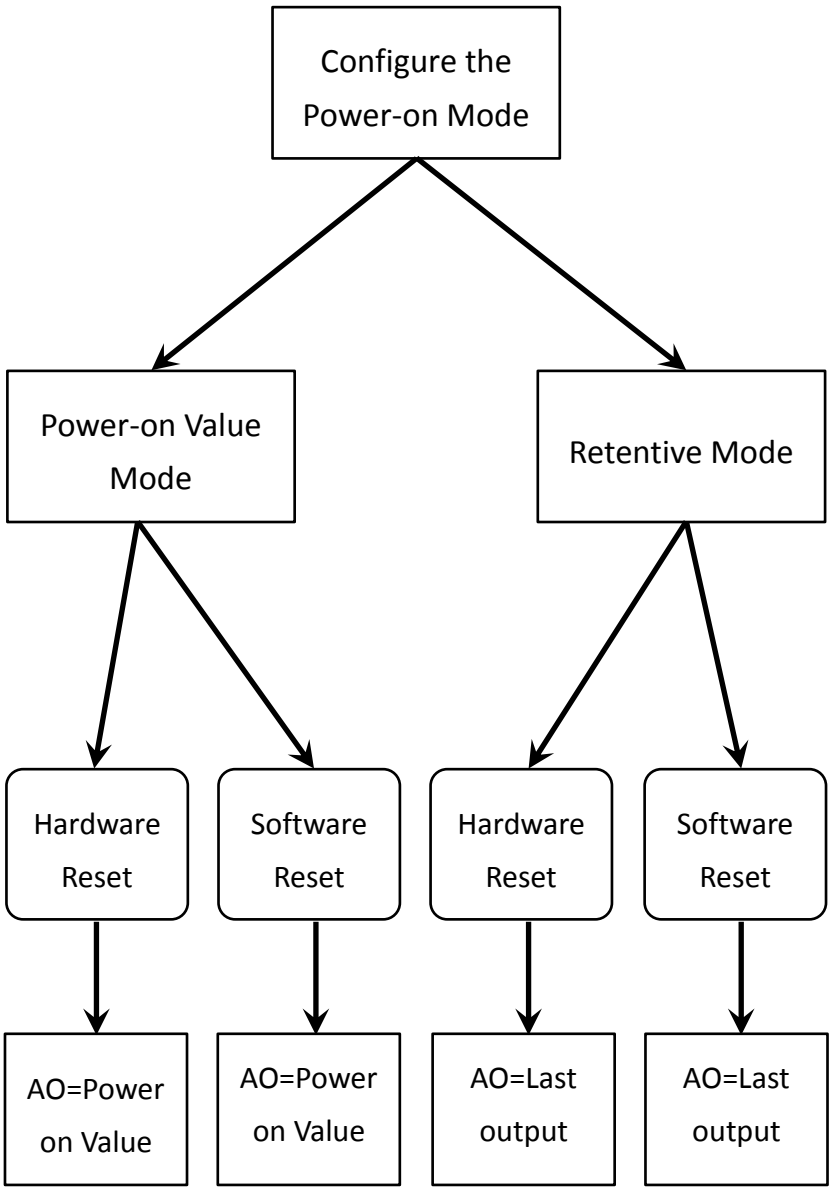
If either the PAC or the module encounters an event that causes the hardware reset or the software reset, the Output will be set to those configured for the specific mode.

The following is an overview of the Analog Output value that will be set for the I-9028U module in each of the Power-on Mode conditions:

Mode	Hardware Reset	Software Reset
Power-on Value Mode	AO = Configured Power-on value	AO = Configured Power-on value
Retentive Mode	AO = Previous AO value	AO = Previous AO value

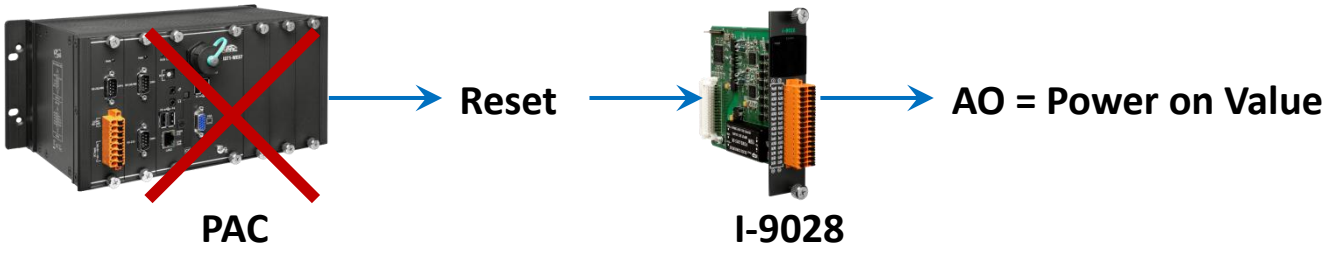
Note: Previous AO value means that after the module reset, the AO value will be retained. For example, if the Power-on Mode is set as Retentive Mode and a hardware reset happened, then the AO value will be the same as the value that existed prior to the hardware reset.

The following is a flowchart that illustrates the outcome of each Power-on Mode:

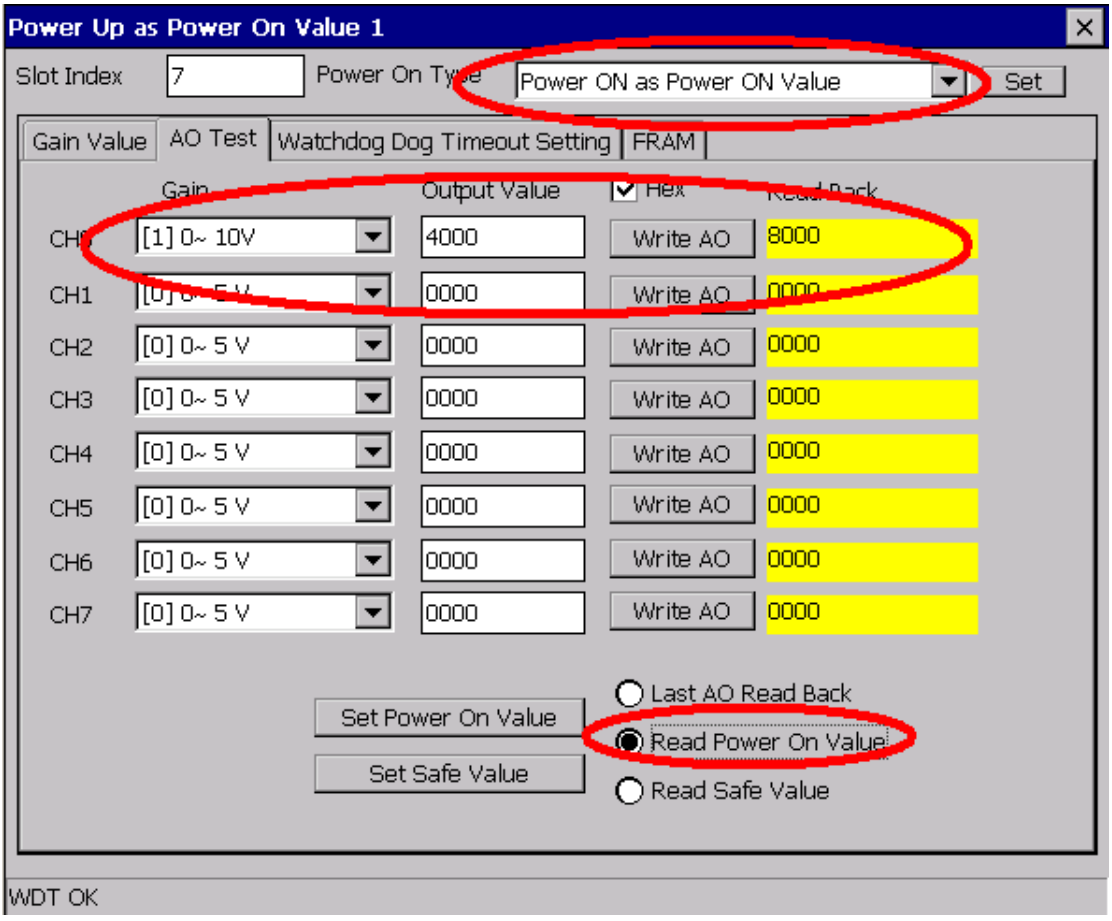


3.1. Power-on Value Mode

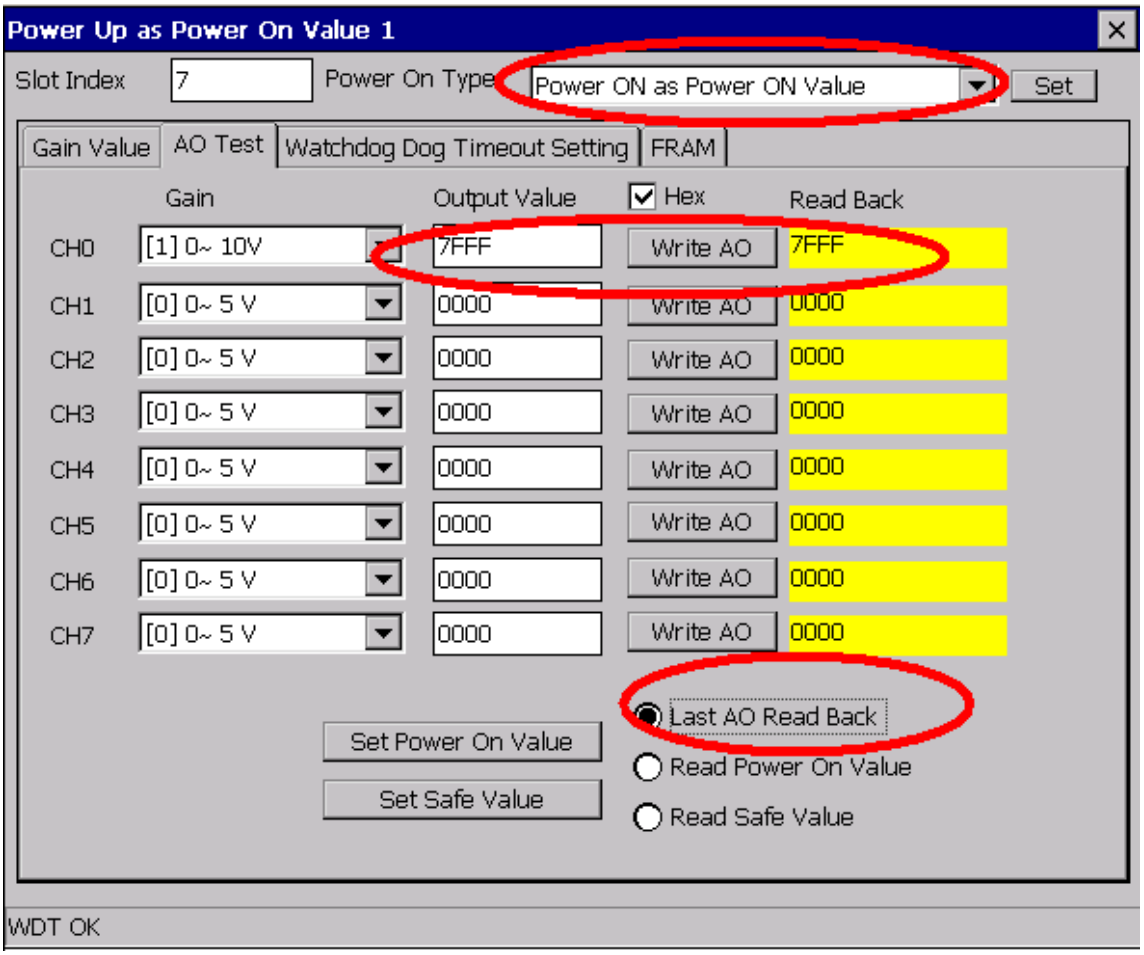
Power-on Value Mode is used to set the output value to the preconfigured Value after an unknown condition has caused the module reset.



For example, set the Power-on Value for channel 0 on the I-9028U module to Gain 1 (0~10 V) and the Output value to 0x8000 in Power-on Value Mode, but the output value is 0x4000.

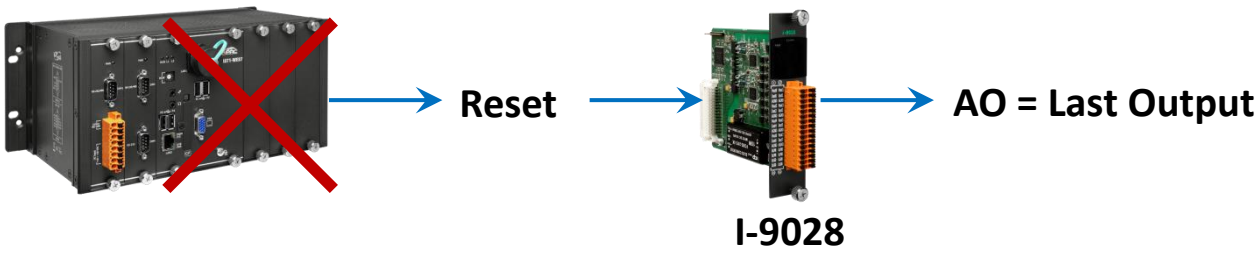


After the the 9000-series PAC is reset, the output value for channel 0 will be Gain 1 (0~10 V) and the output value will be 0x7FFF as below.

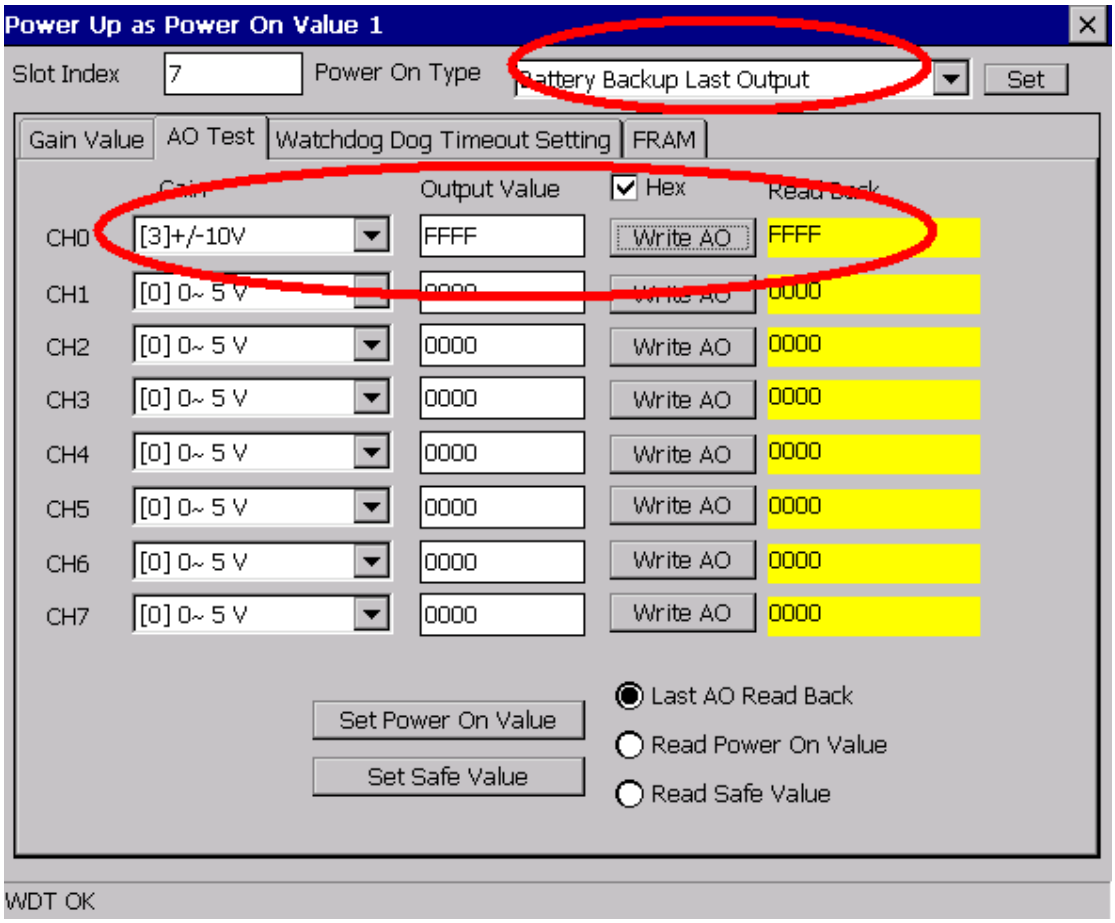


3.2. Retentive Mode

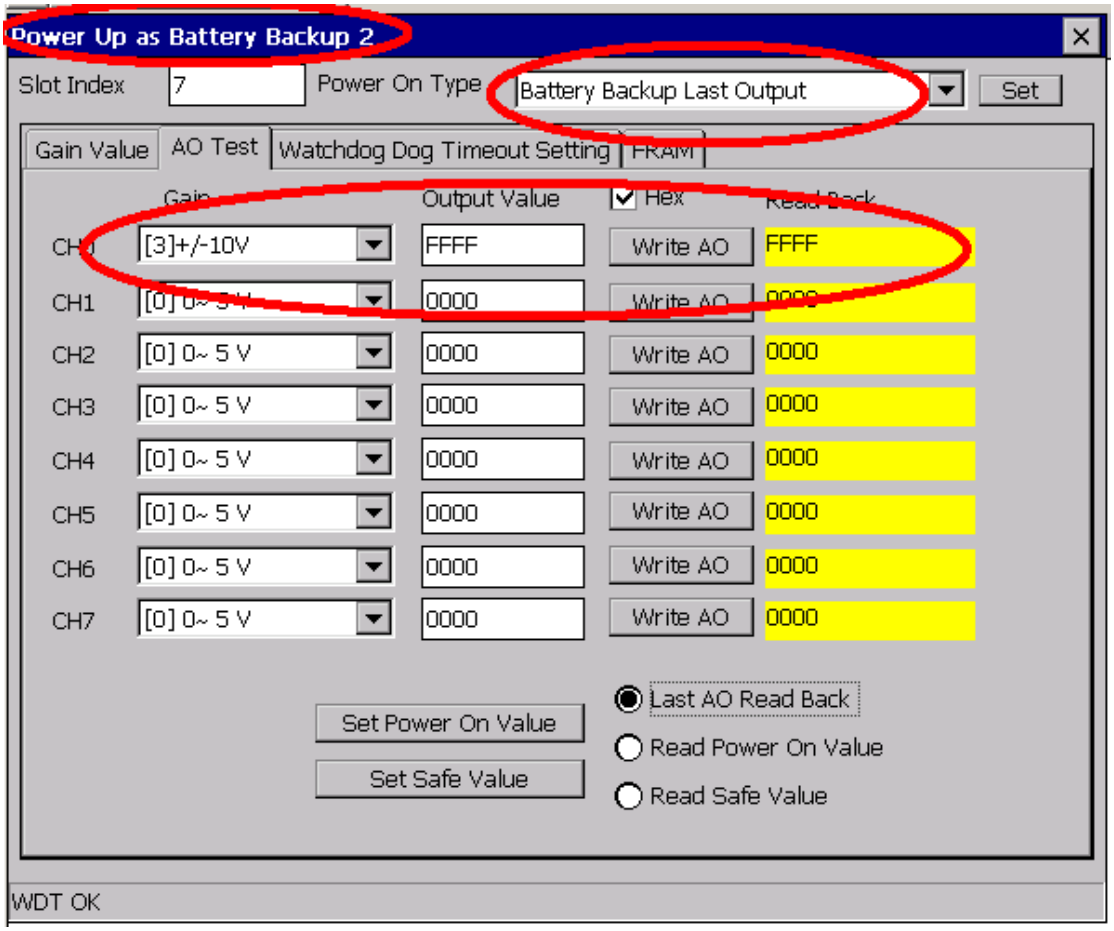
Retentive Mode, also known as Virtual Battery Backup Mode, is used to ensure that the previous AO value is retained if an unknown condition has caused the I-9028U module to be reset.



For example, set channel 0 on the I-9028U module to Gain 3 (+/-10 V) and the Output value to 0xFFFF in Retentive Mode.

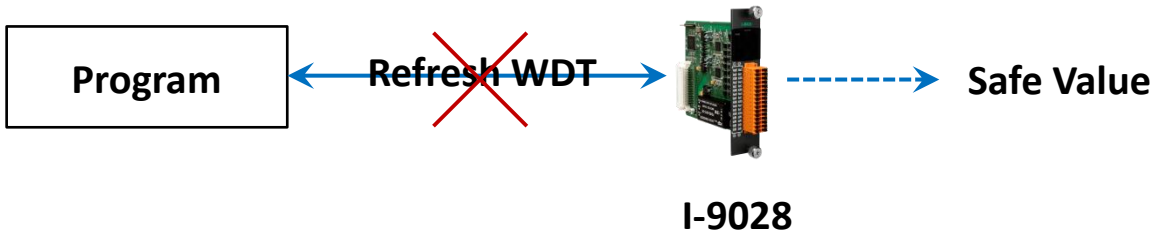


After the power to the 9000-series PAC is reset, the AO value for channel 0 will be Gain 3 (+/-10 V) and the Output value will be set to 0xFFFF as below.

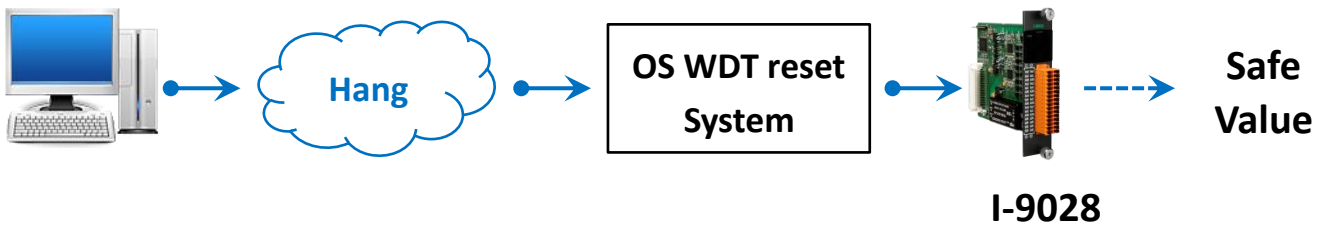


4. Watchdog

The Watchdog Timer (WDT) on the I-9028U module is a software function that monitors the operating status of the module. Its purpose is to prevent problems due to network or communication errors, or 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 happened, 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.



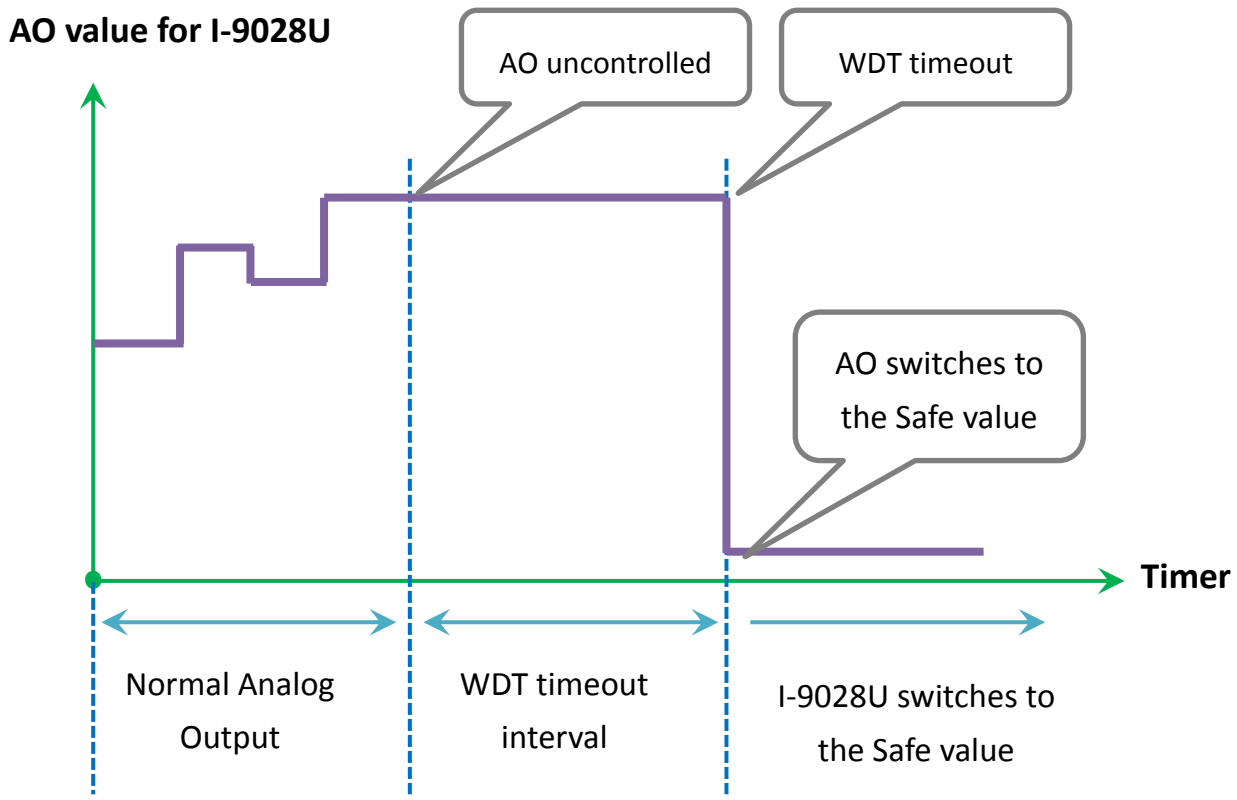
A common application is provided as an example below. If the system encounters an even that causes it to become unresponsive for any reason, the I-9028U WDT and the WDT provided by the operating system for the 9000-series PAC can be used in combination, thereby preventing the system from hang becoming unresponsive, which may cause the Analog Output to be uncontrolled and result in damage to the device. The WDT on the 9000-series PAC will reset the PAC to solve the unresponsiveness problem, and then the Safe Value will be set for the I-9028U module to prevent the AO from becoming uncontrolled.



To set the WDT on the 9000-series PAC to enabled, use the pac_EnableWatchDog(intwdt, DWORD value) function and set the value attribute to 0.

For more detailed information related to the pac_EnableWatchDog function, refer to the PAC Standard API Manual, which can be found at:
ftp://ftp.icpdas.com/pub/cd/winpac_am335x/wp-5231/document/sdk_document/pac_standard_api_manual_1.2.0.pdf

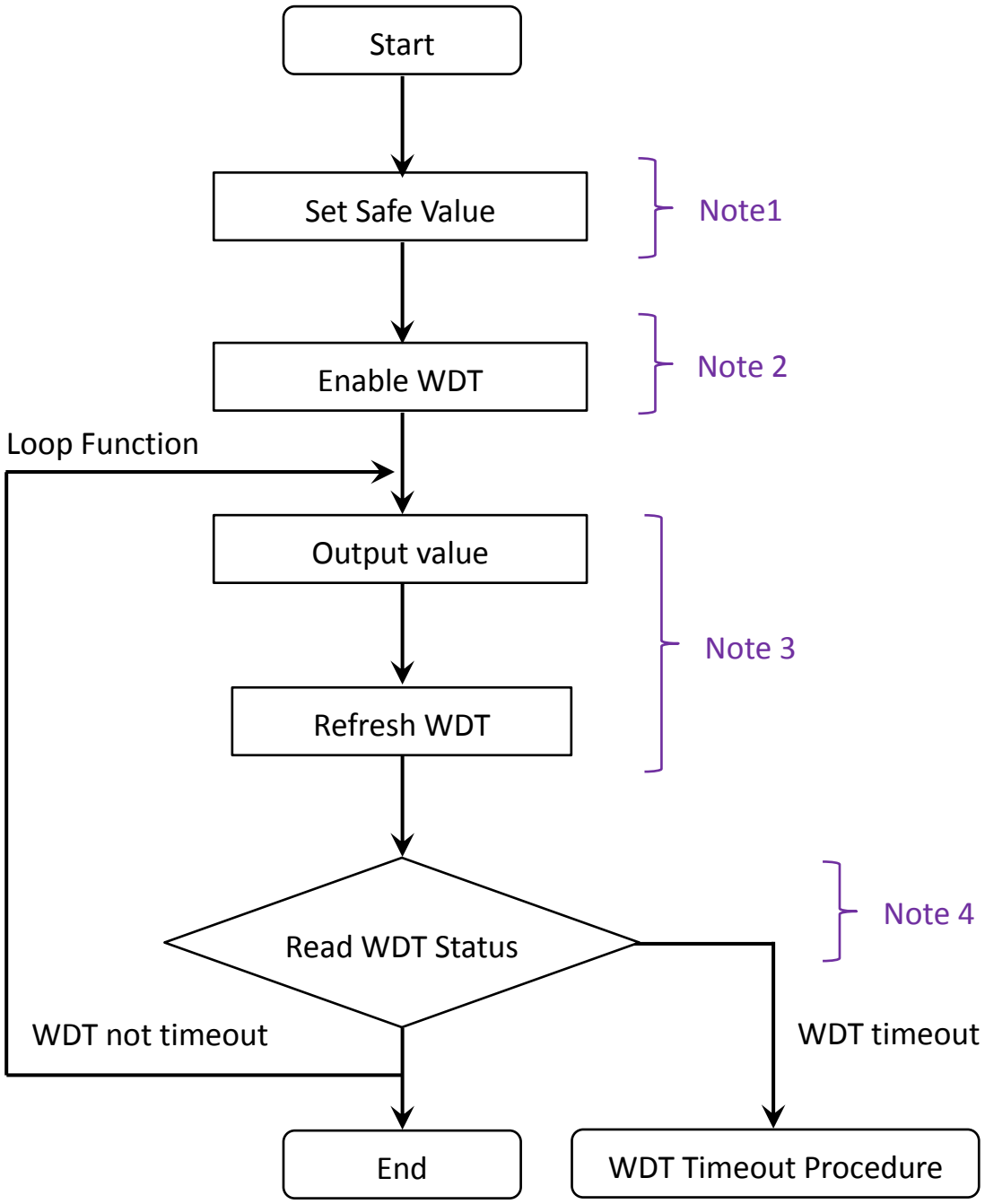
The Analog Output value for the module will be as below. When the WDT timeout is triggered, user can consider necessary to set suit WDT timeout interval and safe value to prevent the AO from becoming uncontrolled.

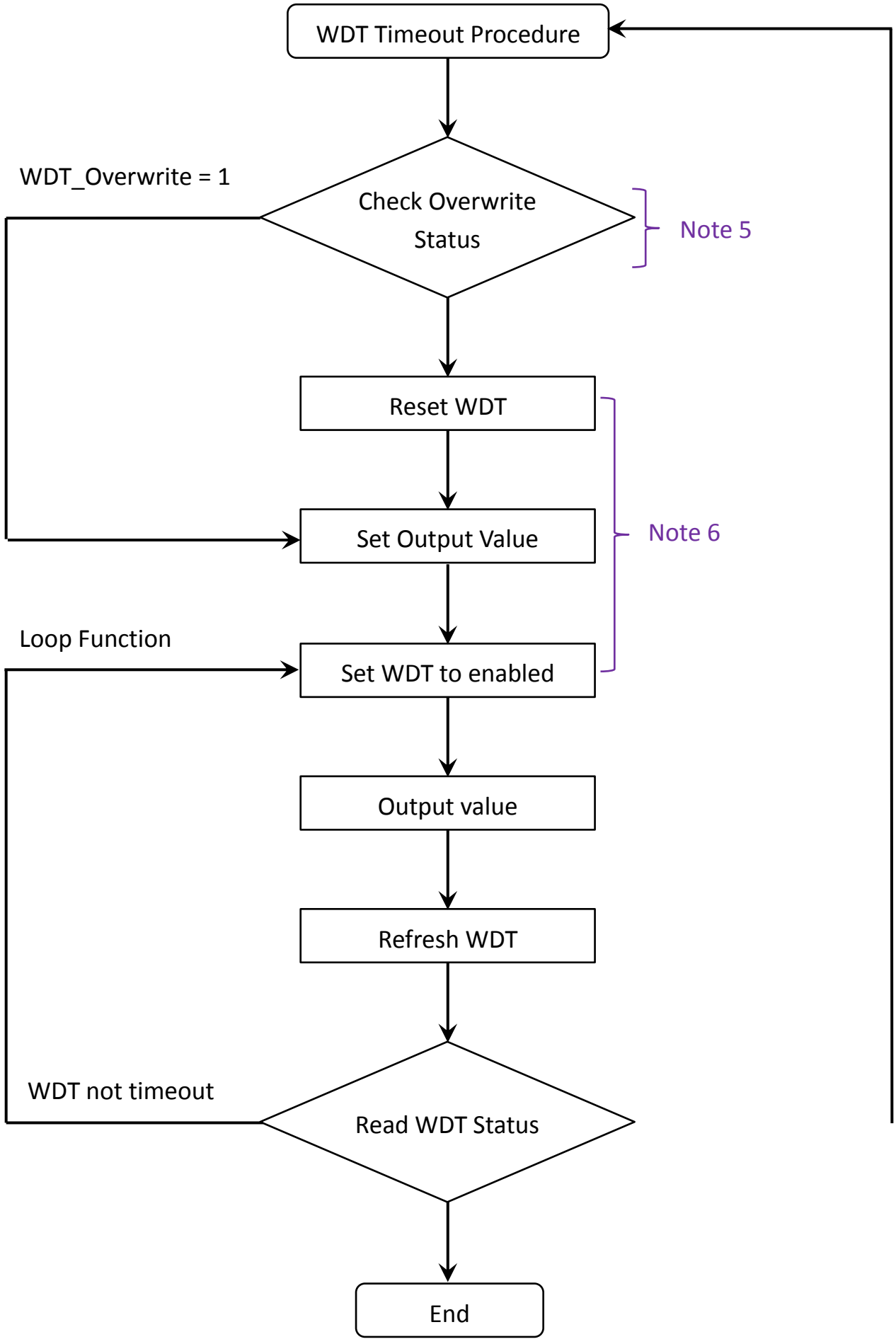


Watchdog operations include basic management operations, such as turning on and refreshing the module. The following sections provide a description of how to programmatically operate the watchdog using the watchdog functions.

Watchdog procedure

The following is an overview of the Watchdog process. More details instructions relating to the notes indicated in the diagrams can be found below. Note that a number of API functions will be used as part of these descriptions. For a more details explanation of these functions, refer to the relevant API reference in Chapter 6.



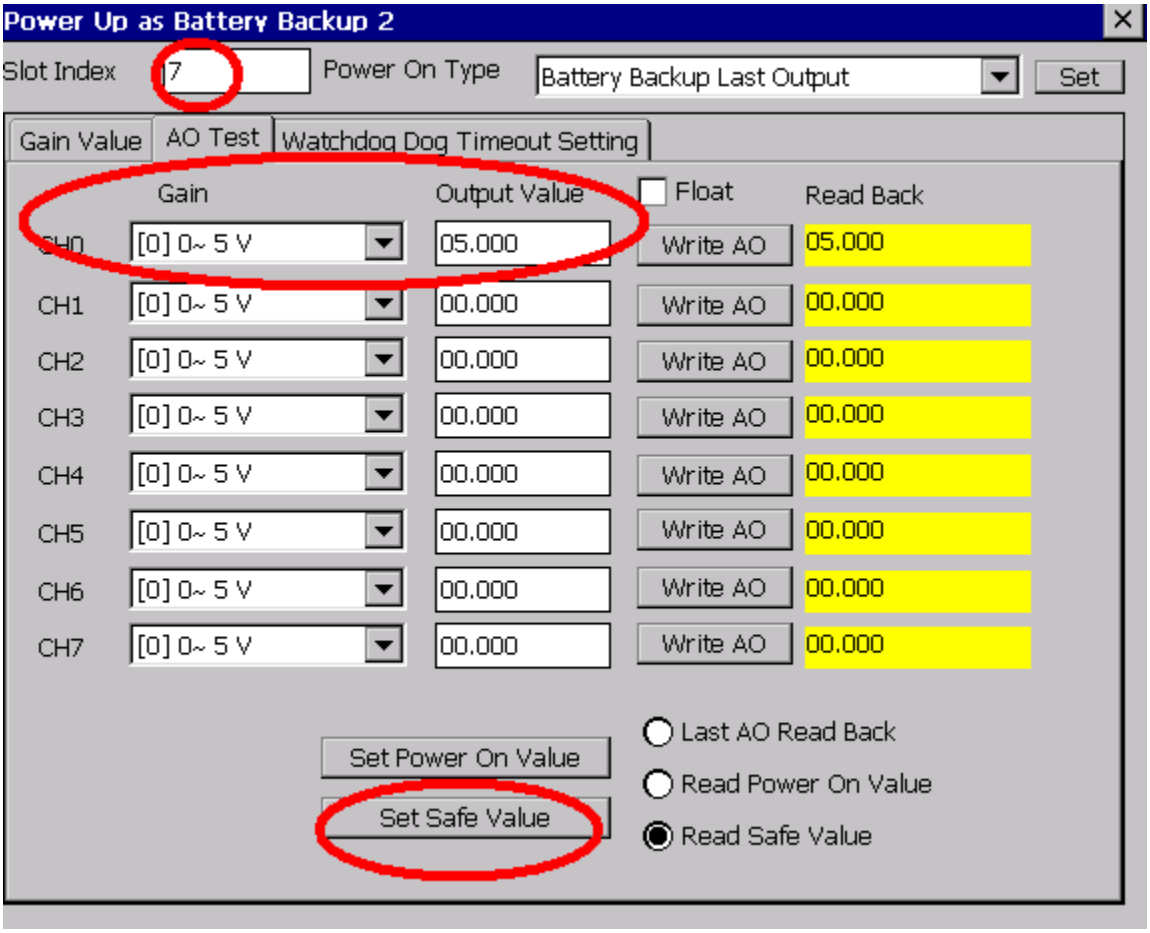

WDT Timeout Procedure
WDT_Overwrite = 1
Check Overwrite Status
Note 5
Reset WDT
Set Output Value
Note 6
Loop Function
Set WDT to enabled
Output value
Refresh WDT
WDT not timeout
Read WDT Status
End

If a Watchdog timeout occurs and the WDT_Overwrite parameter = 0, all the Analog Output values on the I-9028U will be switched to Safe Values, and any operations where an output value needs to be written will not be accepted until the Watchdog is reset. In contrast, if a timeout occurs and the WDT_Overwrite parameter = 1, any output value that is written will be accepted and will reset the Watchdog.

For more details of how to use the Watchdog function, follow the processes described below.

Note 1: Configuring a Safe Value.

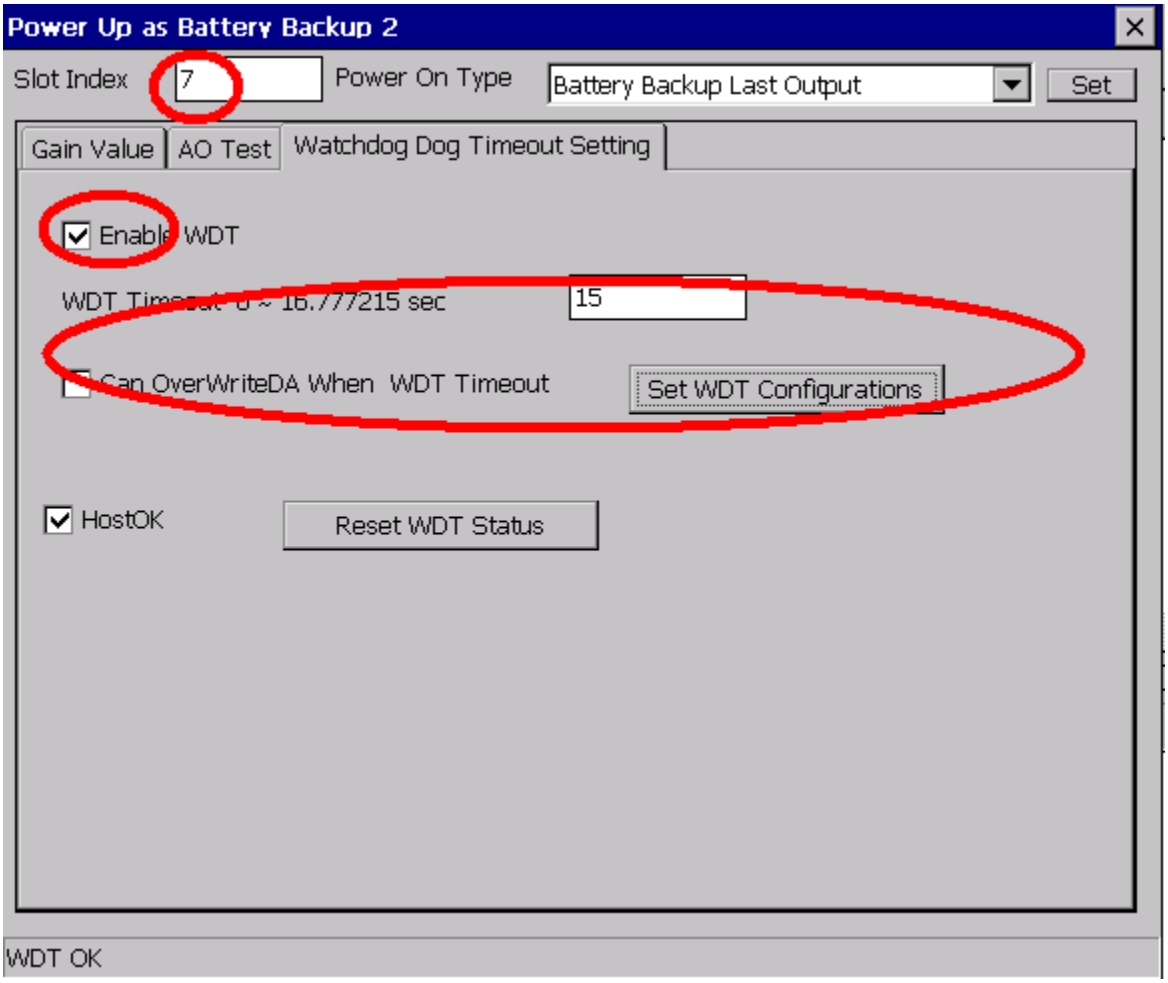
To set the Safe Value, Use the pac_i8028U_WriteSafe_AO (7,0,0,5.0) function to write to channel 0 on slot 0 and set it to Gain 0 (0~5 V) and set the Safe Value for the I-9028U module to 5.0 V.



Note 2: Configuring the Watchdog timeout value and enabling the Watchdog:

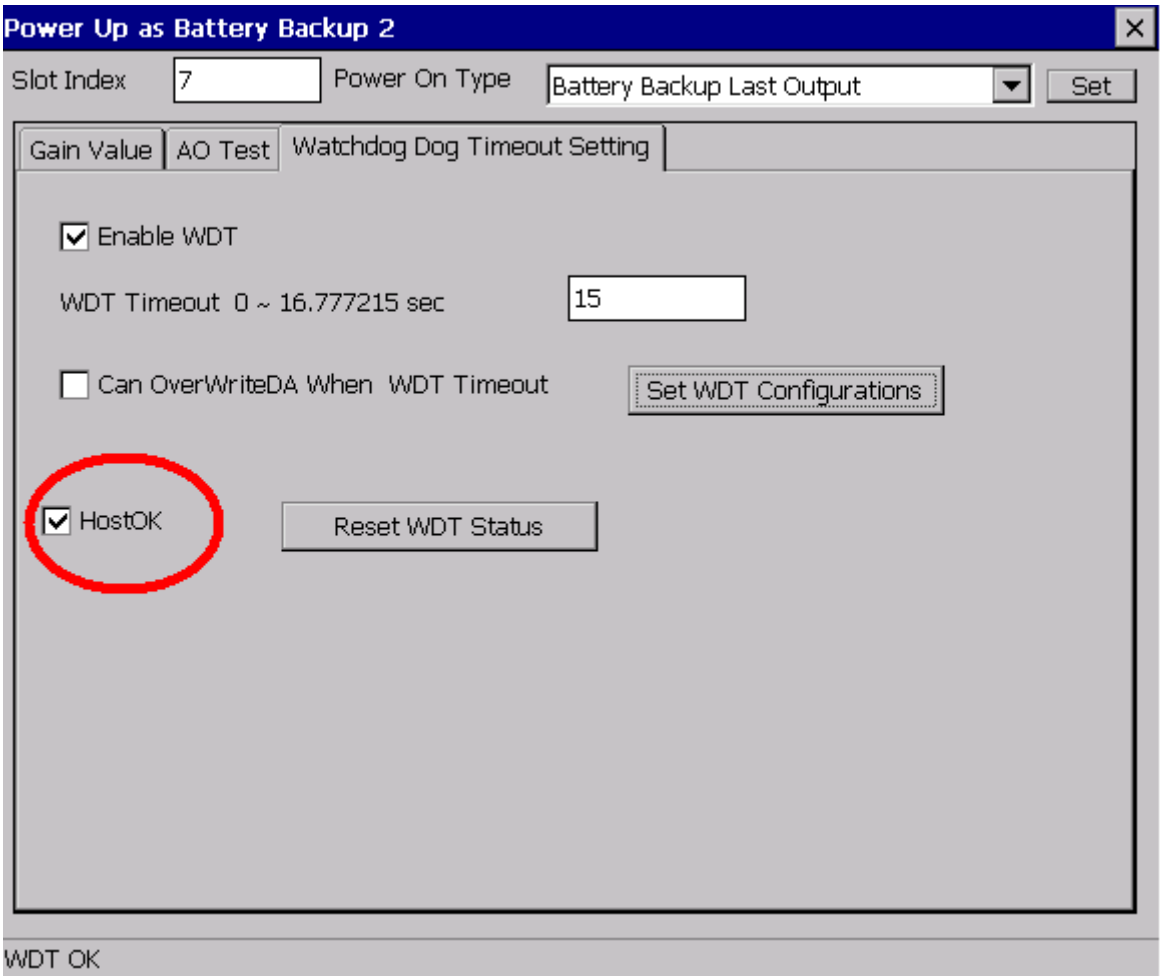
Use the pac_i8028U_SetModuleWDTConfig (7,1,10.0,1) function to enable the WDT, set the WDT timeout value to 10.0 sec, and set the WDT_Overwrite parameter to 0. You then need to enable the WDT function and start the WDT timer.

This is achieved by using the pac_i8028U_GetModuleWDTConfig function, which reads the current configuration being used by the WDT.



Note 3: Refreshing the Watchdog timer.

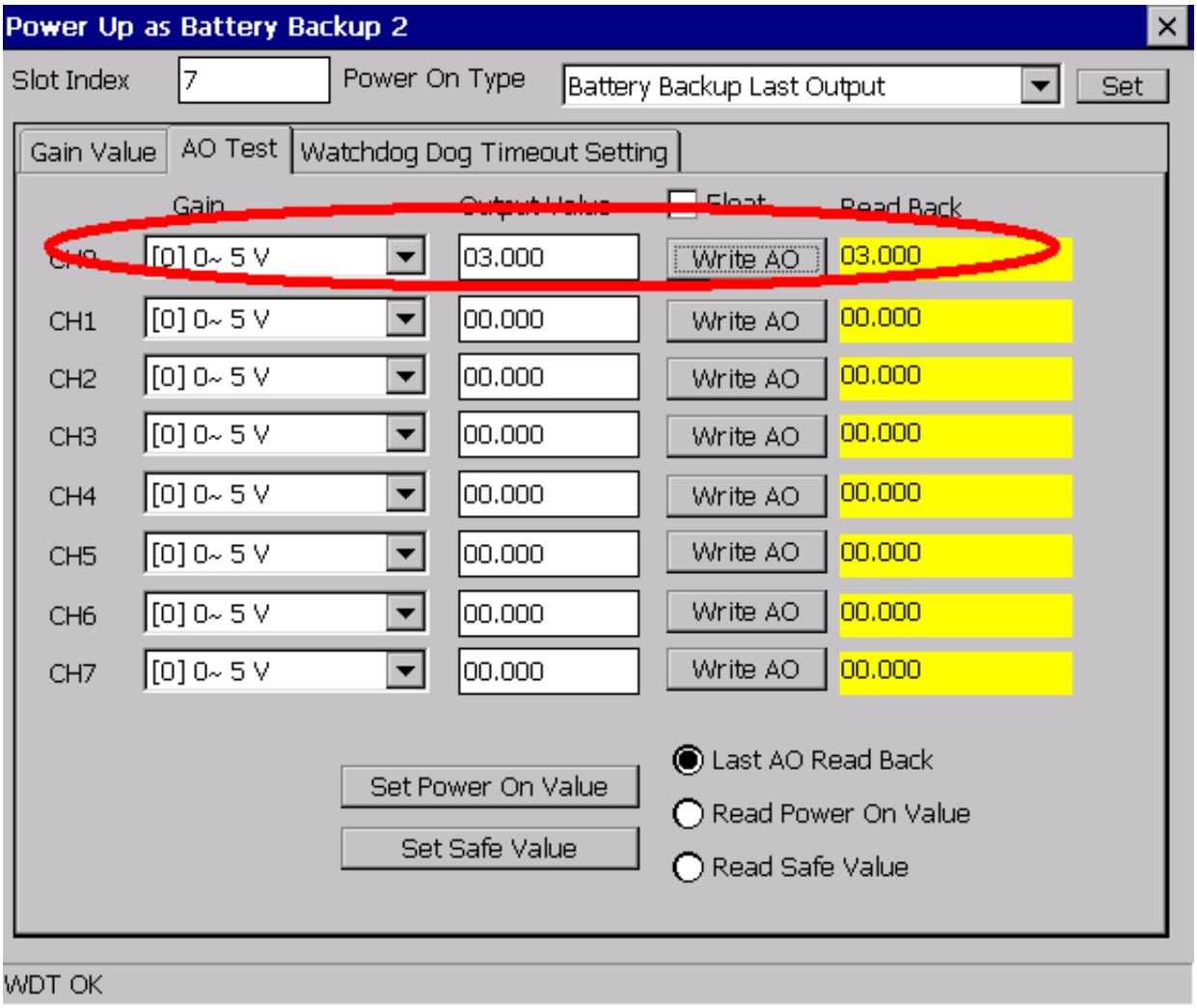
You can refresh the WDT timer using the pac_i8028U_RefreshModuleWDT (7) function.



If the HostOK checkbox is checked, the status of the module will be regularly polled to and the pac_i8028U_RefreshModuleWDT function will be executed to refresh the WDT, ensuring that a WDT timeout will not occur.

Because the WDT timeout value has been set to 10.0 seconds, as described in Note 2, the pac_i8028U_RefreshModuleWDT (0) function must be received by the I-9028U module within 10.0 seconds or a WDT timeout will be triggered and the AO will be set to the Safe Value. Normally, this function can be set to refresh the WDT timer at an interval of at about 10/2 = 5 seconds

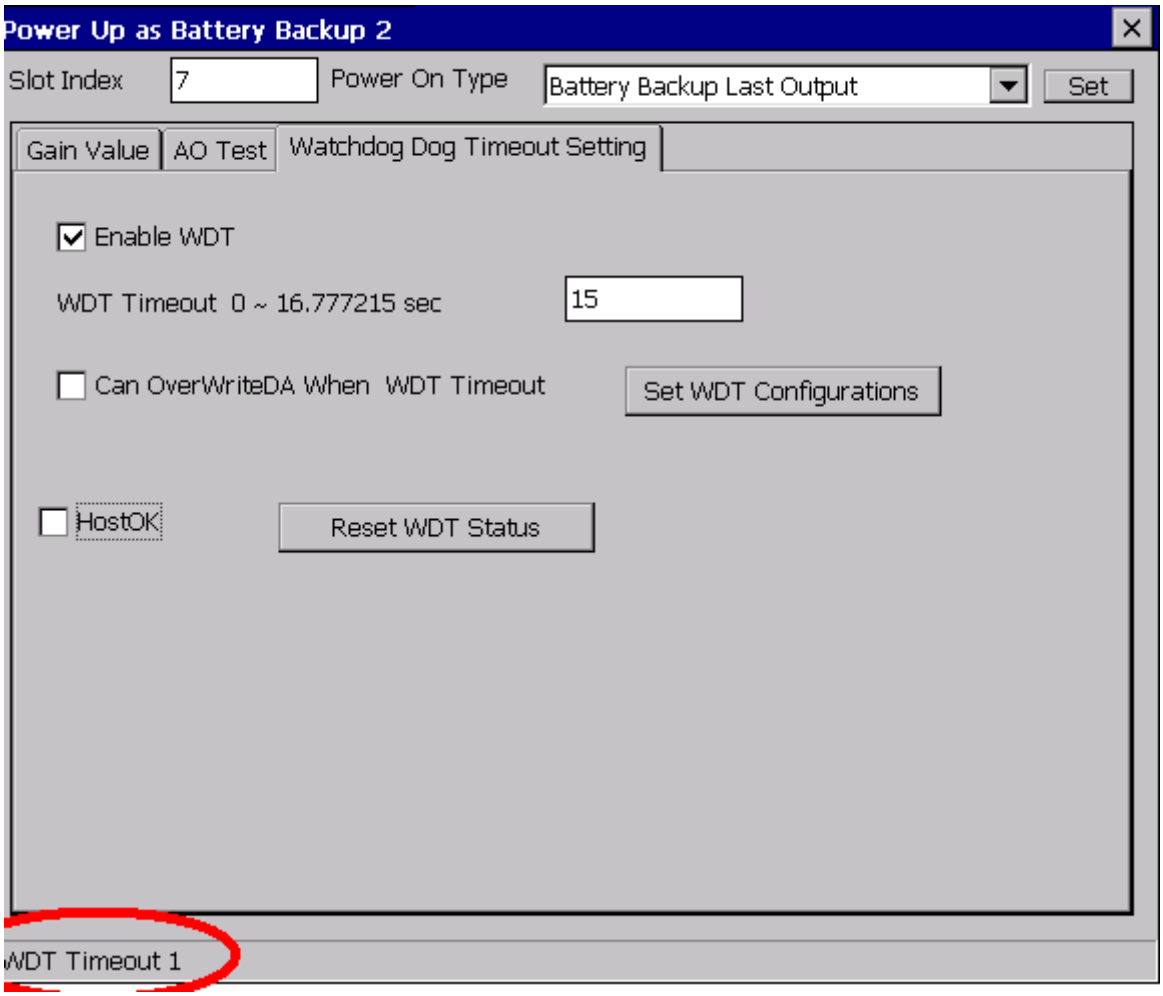
Before a WDT timeout occurs, we can use the pac_i8028U_WriteAO (0,0,0,5.0) function to write to channel 0 on slot 0 and set it to Gain 0 (0~5 V) and set the AO value for the I-9028U module to 5.0 V



Note 4: Checking whether a Watchdog timeout has occurred

If a WDT timeout occurs, the power LED on the I-9028U module will flash and any output command will fail if the WDT_Overwrite parameter =0.
When the I-9028U module switches to the Safe Value, any output command will return a response indicating that the command has failed. This is the most frequently encountered output error related to the I-9028U module.

The best approach to determining whether a WDT timeout has occurred is by polling the WDT status. To do this, use the pac_i8028U_GetModuleWDTStatus (7) function to read the status of the WDT on the module. If a timeout has not occurred, the return value will be 0. If a WDT timeout has occurred, the return value will be 1



Note 5: Checking the communication status if a Watchdog timeout has occurred

If an application has triggered a Watchdog timeout, it means that there may have been a problem with the communication between the host application and one or more of the remote I/O modules.

In this situation, check the communication line is secure at the terminal board.

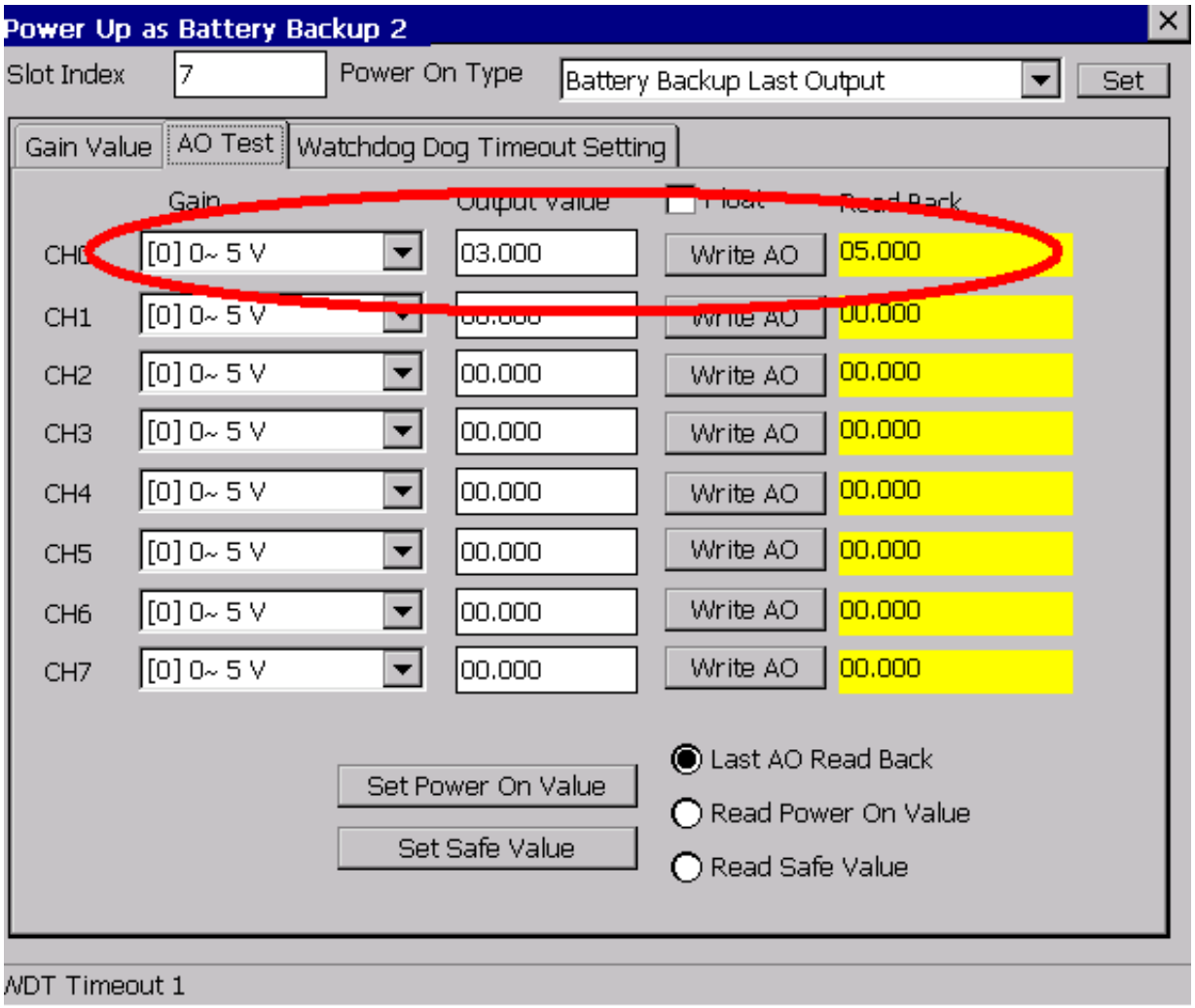
Ensure that the communication problems have been resolved continuing to the next step. If a communication problem still exists within the system, communication will remain unstable and the Safe Value will be triggered often.

Note 6: Writing the AO value when a WDT timeout has occurred and the AO is continually set to the Safe Value

If the WDT_Overwrite parameter =0, use the pac_i8028U_ResetModuleWDT function to reset the WDT and clear the timeout. However, note that this will also disable the WDT.

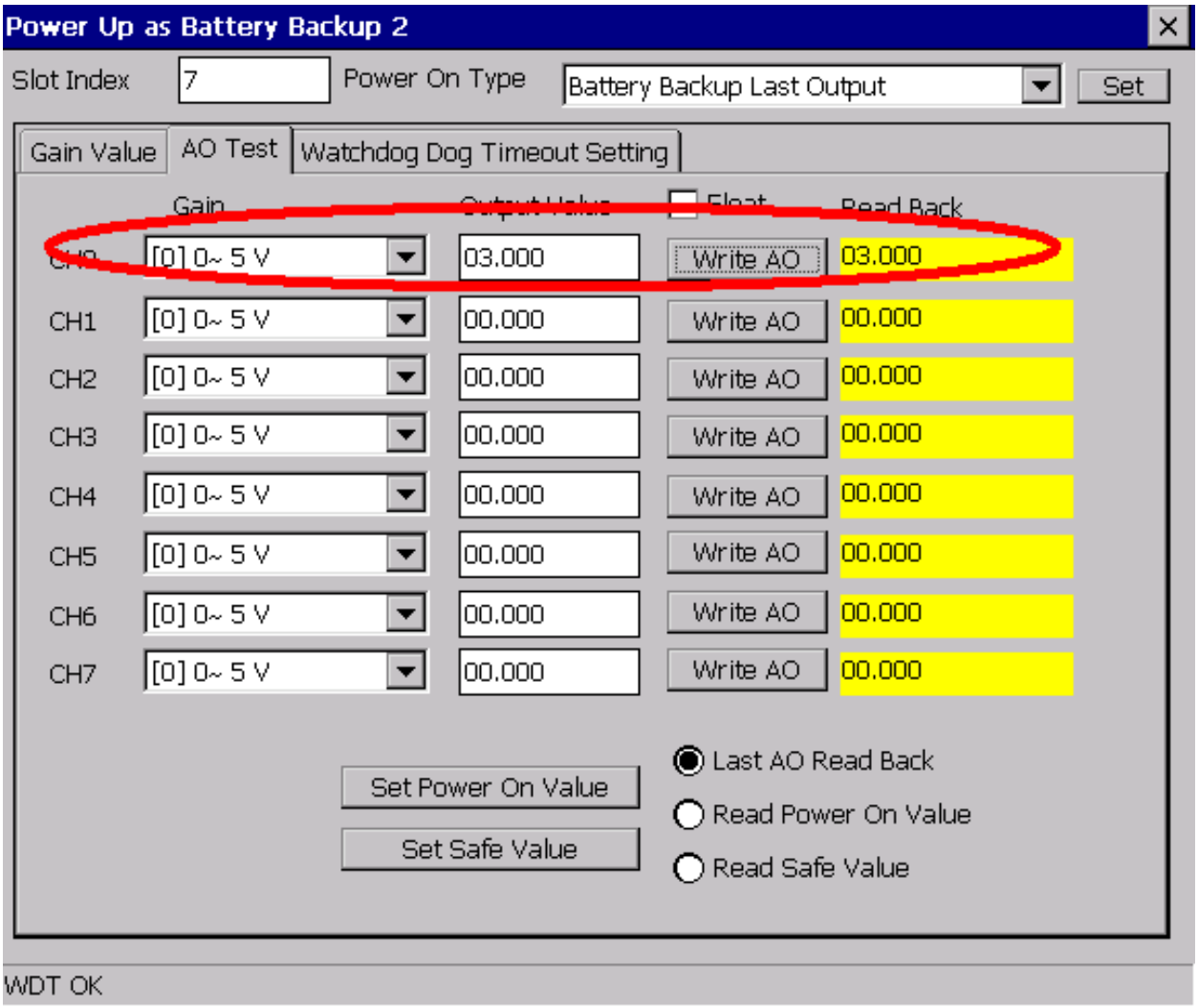
Once the WDT timeout has been cleared, the output module can be controlled again.

If the WDT_Overwrite parameter = 1, use the pac_i8028U_ResetModuleWDT function to write an AO value and the WDT will be disabled.



The WDT timeout and the AO switch to the Safe Value of 5 V.

Writing to the AO will disable the WDT, meaning the AO can be written.



The WDT can then be re-enabled, as described in Note 2.

5. 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-9028U module. 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	Location
I-8024U/I-8028U For I-8000	
Library	CD:\8000\NAPDOS\8000\841x881x\demo\Lib ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/8000/841x881x/demo/lib/
Demo	CD:\8000\NAPDOS\8000\841x881x\demo\IO_in_Slot ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/8000/841x881x/demo/io_in_slot/
I-8024U/I-8028U For IPAC-8000	
Library	CD:\8000\NAPDOS\iPAC8000\Demo\Basic\iP-84x1_iP-88x1\Lib ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/ipac8000/demo/basic/ip-84x1_ip-88x1/lib/
Demo	CD:\8000\NAPDOS\iPAC8000\Demo\Basic\iP-84x1_iP-88x1\IO_in_Slot ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/ipac8000/demo/basic/ip-84x1_ip-88x1/io_in_slot/
I-8024U/I-8028U For WinPAC_AM335x	
Library	CD:\WinPAC_AM335x\VP-x231\SDK\IO_Modules ftp://ftp.icpdas.com/pub/cd/winpac_am335x/vp-x231/sdk/io_modules/
Demo	VC2008 Demo: CD:\ WinPAC_AM335x\VP-x231\demo\PAC\Vc2008\IO\Local ftp://ftp.icpdas.com/pub/cd/winpac_am335x/vp-x231/demo/pac/vc2008/io/local/ C# Demo: CD:\WinPAC_AM335x\VP-x231\demo\PAC\C#\IO\Local ftp://ftp.icpdas.com/pub/cd/winpac_am335x/vp-x231/demo/pac/c%23/io/local/

I-8024U/I-8028U For WinPAC	
Library	CD:\WinPAC\napdos\wp-8x4x_ce50\SDK\IO_Modules ftp://ftp.icpdas.com/pub/cd/winpac/napdos/wp-8x4x_ce50/sdk/io_modules/
Demo	VC2005 Demo: CD:\WinPAC\napdos\wp-8x4x_ce50\Demo\WinPAC\VC2005\IO\Local ftp://ftp.icpdas.com/pub/cd/winpac/napdos/wp-8x4x_ce50/demo/winpac/vc2005/io/local/ C# Demo: CD:\WinPAC\napdos\wp-8x4x_ce50\Demo\WinPAC\C#\IO\Local ftp://ftp.icpdas.com/pub/cd/winpac/napdos/wp-8x4x_ce50/demo/winpac/c%23/io/local/
I-8024U/I-8028U For XP-8000	
Library	CD: \XPAC\XP-8000\SDK\IO ftp://ftp.icpdas.com/pub/cd/xp-8000/sdk/io/
Demo	VC2005 Demo: CD:\XPAC\XP-8000\Demo\pacsdk\vc\IO\Local ftp://ftp.icpdas.com/pub/cd/xp-8000/demo/pacsdk/vc/io/local/ C# Demo: CD:\XPAC\XP-8000\Demo\pacsdk\csharp.net\IO\Local\windows_forms ftp://ftp.icpdas.com/pub/cd/xp-8000/demo/pacsdk/csharp.net/io/local/windows_forms/
I-8024U/I-8028U For XP-8000-CE6	
Library	CD: \XPAC\XP-8000-CE6\SDK\Special_IO ftp://ftp.icpdas.com/pub/cd/xp-8000-ce6/sdk/special_io/
Demo	VC2005 Demo: CD:\XPAC\XP-8000-CE6\demo\XPAC\VC2005\IO\Local ftp://ftp.icpdas.com/pub/cd/xp-8000-ce6/demo/xpac/vc2005/io/local/ C# Demo: CD:\XPAC\XP-8000-CE6\demo\XPAC\C#\IO\Local ftp://ftp.icpdas.com/pub/cd/xp-8000-ce6/demo/xpac/c%23/io/local/

Platform	Location
I-9024U/I-9028U For WP-9000	
Library	CD:\WinPAC_AM335x\wp-9000\SDK\IO_Modules ftp://ftp.icpdas.com/pub/cd/winpac_am335x/wp-9000/sdk/io_modules/
Demo	VC2008 Demo: CD:\WinPAC_AM335x\wp-9000\demo\PAC\Vc2008\IO\Local ftp://ftp.icpdas.com/pub/cd/winpac_am335x/wp-9000/demo/pac/vc2008/io/local/ C# Demo: CD:\WinPAC_AM335x\wp-9000\demo\PAC\C#\IO\Local ftp://ftp.icpdas.com/pub/cd/winpac_am335x/wp-9000/demo/pac/c%23/io/local/
For IPPC-WES7	
Library	CD:\ippc-wes7\sdk\IO ftp://ftp.icpdas.com/pub/cd/ippc-wes7/sdk/io/
I-8024U, I-8028U Demo	VC Demo: CD:\ippc-wes7\demo\pacsdk\vc\io\local\io-8k ftp://ftp.icpdas.com/pub/cd/ippc-wes7/demo/pacsdk/vc/io/local/io-8k/ C# Demo: CD:\ippc-wes7\demo\pacsdk\csharp.net\io\local\io-8k ftp://ftp.icpdas.com/pub/cd/ippc-wes7/demo/pacsdk/csharp.net/io/local/io-8k/
I-9024U, I-9028U Demo	VC Demo: CD:\ippc-wes7\demo\pacsdk\vc\io\local\io-9k ftp://ftp.icpdas.com/pub/cd/ippc-wes7/demo/pacsdk/vc/io/local/io-9k/ C# Demo: CD:\ippc-wes7\demo\pacsdk\csharp.net\io\local\io-9k ftp://ftp.icpdas.com/pub/cd/ippc-wes7/demo/pacsdk/csharp.net/io/local/io-9k/

6. API References

ICPDAS supplies a range of C/C++ API functions for the I-9028U module. When developing a custom program, refer to either the 9028W.h header file, 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 pac_i9028.cs file.

More details of where to find the relevant libraries and files, refer to Chapter 5 Location of the Demo and Library Programs.

API naming table

The following table describes the platforms and in which the product series included and the different part of function name.

Platform	Product included	API prefix characters
Windows CE5 Windows CE6	WP-8000 series XP-8000-CE6 series XP-8000-Atom-CE6 series	“pac_i8028U_ “+ function name
Windows CE7	WP-8000 series WP-9000-CE7 series	“pac_i8028U_ “+ function name
WES	XP-8000 series XP-8000-Atom series	“pac_i8028U_ “+ function name
WES7	XP-8000-WES7 series XP-9000 series	“pac_i8028U_ “+ function name
MiniOS7	I-8000 series iPAC-8000 series	“i8028U_” + function name
Linux	LinPAC-8000 series LinPAC-9000 series	“i8028U_” + function name

The following is an overview of the functions provided in the 9028.lib for use with the 9000 PAC platform. Detailed information related to individual functions can be found in the following sections.

Function	Description
pac_i8028U_Init	Used to initialize the driver and confirm the hardware ID.
pac_i8028U_GetFirmwareVersion	Used to retrieve the version number for the FPGA firmware of the I-9028 module and is used for troubleshooting purposes.
pac_i8028U_GetLibVersion	Used to retrieve the version number for the 9028.lib library file currently installed on the I-9028 module and is used for troubleshooting purposes.
pac_i8028U_GetLibDate	Used to retrieve the release date of the 9028.lib library file currently installed on the I-9028 module and is used for troubleshooting purposes..
pac_i8028U_ReadAO_GainOffset	Used to retrieve the current gain and offset values for a specified input type and channel.
pac_i8028U_WriteAOHex	Used to write the Analog Output value for a specified channel in hexadecimal format.
pac_i8028U_WriteAO	Used to write the Analog Output value for a specified channel in decimal format.
pac_i8028U_ReadAOHex	Used to read the Analog Output value for a specified channel in hexadecimal format.
pac_i8028U_ReadAO	Used to read the Analog Output value for a specified channel
pac_i8028U_SetPowerOnEnStatus	Used to set the Mode for the Power -on Value.
pac_i8028U_GetPowerOnEnStatus	Used to read the current Mode being used for the Power-on Value.
pac_i8028U_WritePowerOnHex_AO	Used to write the Power -on Value for a specified channel in hexadecimal format.
pac_i8028U_WritePowerOn_AO	Used to write the Power -on Value for a specified channel in decimal format.
pac_i8028U_ReadPowerOnHex_AO	Used to read the current Power -on Value for a specified channel in hexadecimal format.
pac_i8028U_ReadPowerOn_AO	Used to read the current Power -on Value for a specified channel in decimal format.
pac_i8028U_WriteSafeHex_AO	Used to write the Safe Value for a specified channel in hexadecimal format.

pac_i8028U_WriteSafe_AO	Used to write the Safe Value for a specified channel in decimal format.
pac_i8028U_ReadSafeHex_AO	Used to read the current Safe Value for a specified channel in hexadecimal format.
pac_i8028U_ReadSafe_AO	Used to read the current Safe Value for a specified channel in decimal format.
pac_i8028U_SetModuleWDTConfig	Used to configure the attributes for the Watchdog parameter on the I-9028 module.
pac_i8028U_GetModuleWDTConfig	Used to read the current attributes of the Watchdog parameter on the I-9028 module.
pac_i8028U_GetModuleWDTStatus	Used to retrieve the current status of the Watchdog on the I-9028 module.
pac_i8028U_ResetModuleWDT	Used to reset the status of the Watchdog on the I-9028 module.
pac_i8028U_RefreshModuleWDT	Used to refresh the status of the Watchdog on the I-9028 module.

6.1. i8028U_Init

This function is used to initialize the driver and confirm the hardware ID.

Syntax

For MiniOS7

```
short i8028U_Init(
    int slot
);
```

For Windows (CE and WES)

```
short pac_i8028U_Init(
    int slot
);
```

Parameters

slot:

specifies the number of slot (0 - 7).

Return Value

0 = the module in the slot is an I-8024U/I-8028U/I-9024U/I-9028U.

-1 = there is no module in this slot.

For other return values, see the Error Codes listed in Appendix A.

Note

Before executing any functions on the I-8024U/I-8028U/I-9024U/I-9028U, the pac_i8028U_Init function needs to be called once for I-8024U/I-8028U/I-9024U/I-9028U. If there are two or more I-8024U/I-8028U/I-9024U/I-9028U modules, you need call the pac_i8028U_Init function for each I-8024U/I-8028U/I-9024U/I-9028U module individually by passing the slot number that the I-8024U/I-8028U/I-9024U/I-9028U module is plugged into.

Example

[C/C++]

```
Int slot, err;
err = i8028U_Init(slot);
```

[C#]

```
int slot;
pac_i8028UNet.pac8028U.Init(slot);
```

6.2. i8028U_GetFirmwareVersion

This function is used to retrieve the version number of the FPGA firmware for a specified module. Note that this function is only used for troubleshooting or recording purposes.

Syntax

For MiniOS7

```
short i8028U_GetFirmwareVersion(
    int slot,
    short* ver
);
```

For Windows (CE and WES)

```
short pac_i8028U_GetFirmwareVersion(
    int slot,
    short* ver
);
```

Parameters

slot:

specifies the slot number (0 - 7).

**ver:*

[output] The version number of the primary FPGA firmware for the module..

Return Value

please refer the Error Code.

Example

[C/C++]

```
short ver = 0, slot = 0;

i8028U_GetFirmwareVersion (slot,&ver);
```

[C#]

```
Int16 slot = 1;;
Int16 firmware = 0
pac_i8028UNet.pac8028U.FirmwareVersion(slot, ref firmware);
```

6.3. i8028U_GetLibVersion

This function is used to retrieve the version number of the 9028.lib library file currently installed on the I-9028 module. Note that this function is only used for troubleshooting or recording purposes.

Syntax

For MiniOS7

```
short i8028U_GetLibVersion(void);
```

For Windows (CE and WES)

```
short pac_i8028U_GetLibVersion(void);
```

Parameters

None

Return Value

The version number of the library file currently being used on the module.

Example

[C/C++]

```
short ver;
ver= pac_i8028U_GetLibVersion();
```

[C#]

```
Int16 ver;
ver = pac_i8028UNet.pac8028U.LibVersion();
```

6.4. i8028U_GetLibDate

This function is used to retrieve the release date of the 9028.lib library file currently installed on the I-9028 module. Note that this function is only used for troubleshooting or recording purposes.

Syntax

For MiniOS7

```
void i8028U_GetLibDate(
    char libDate[]
);
```

For Windows (CE and WES)

```
void pac_i8028U_GetLibDate(
    char libDate[]
);
```

Parameters

**libDate:*

[output] the release date of the pac_i8028U.lib

Return Value

None

Example

[C/C++]

```
Char libDate [32];
i8028U_GetLibDate(libDate);
```

[C#]

```
string date;
date = pac_i8028UNet.pac8028U.LibDate();
```

6.5. i8028U_ReadAO_GainOffset

This function is used to retrieve the gain and offset values for a specified input type and channel on the I-9028U module.

Syntax

For MiniOS7

```
void i8028U_ReadAO_GainOffset(
    int slot,
    int ch,
    short gain,
    unsigned short *gVal,
    short *oVal
);
```

For Windows (CE and WES)

```
void pac_i8028U_ReadAO_GainOffset(
    int slot,
    int ch,
    short gain,
    unsigned short *gVal,
    short *oVal
);
```

Parameters

slot:

specifies the slot number (0 - 7).

ch:

specifies the channel number (0 - 7).

gain:

specifies the type code for the gain (0 - 5).

0: 0 ~ 5 V

1: 0 ~ 10 V

2: +/-5 V

3: +/-10 V

4: +/-20 mA

5: 0 ~ 20mA

**gVal:*

[output] the gain value for the input range.

**oVal:*

[output] the offset value for the input range.

Return Value

None

Example

[C/C++]

```
unsigned short gVal = 0;
short oVal = 0;
int ch,gain;
i8028U_ReadAO_GainOffset(slot, ch, gain, &gVal, &oVal);
```

[C#]

```
Int slot,ch;
short gain;
UInt16 gVal = 0;
Int16 oVal = 0;
pac_i8028UNet.pac8028U.ReadGainOffset(slot, ch, gain, ref gVal, ref oVal);
```

6.6. i8028U_WriteAOHex

This function is used to write the Analog Output value for a specified channel in hexadecimal format.

Syntax

For MiniOS7

```
short i8028U_WriteAOHex(
    int slot,
    int ch,
    short gain,
    short aoHex
);
```

For Windows (CE and WES)

```
short pac_i8028U_WriteAOHex(
    int slot,
    int ch,
    short gain,
    short aoHex
);
```

Parameters

slot:

specifies the slot number (0 - 7).

ch:

specifies the channel number (0 - 7).

gain:

specifies the input type (0 - 4), where:

0: 0 ~ 5 V

1: 0 ~ 10 V

2: +/-5 V

3: +/-10 V

4: +/-20 mA

5: 0 ~ 20 mA

aoHex:

specifies the data in hexadecimal format.

Return Value

Please refer the Error Code.

Example

[C/C++]

```
int slot, ch;
short gain,hVal;
i8028U_WriteAOHex(slot, ch, gain, hVal);
```

[C#]

```
Int slot;
Int16 ch;
short gain;
Int16 da;
pac_i8028UNet.pac8028U.WriteAOHex(slot, ch, gain, da);
```

6.7. i8028U_WriteAO

This function is used to write the Analog Output value for a specified channel in decimal format.

Syntax

For MiniOS7

```
short i8028U_WriteAO(
    int slot,
    int ch,
    short gain,
    float aoData
);
```

For Windows (CE and WES)

```
short pac_i8028U_WriteAO(
    int slot,
    int ch,
    short gain,
    float aoData
);
```

Parameters

slot:

specifies the slot number (0 - 7).

ch:

specifies the channel number (0 – 7).

gain:

specifies the input type (0 - 5), where:

0: 0 ~ 5 V

1: 0 ~ 10 V

2: +/-5 V

3: +/-10 V

4: +/-20 mA

5: 0 ~ 20mA

aoData:

the data in decimal format.

Return Value

0 = No Error

For other return values, see the Error Codes listed in Appendix A.

Example

[C/C++]

```
int slot, ch, val;
short gain;
i8028U_WriteAO(slot, ch, gain,val);
```

[C#]

```
Int slot,ch,val;
short gain = 0;
pac_i8028UNet.pac8028U.WriteAO(slot, ch, gain, val);
```

6.8. i8028U_ReadAOHex

This function is used to read the current Analog Output value from a specified channel in hexadecimal format.

Syntax

For MiniOS7

```
void i8028U_ReadAOHex(
    int slot,
    int ch,
    short *gain,
    short *aoHex
);
```

For Windows (CE and WES)

```
void pac_i8028U_ReadAOHex(
    int slot,
    intch,
    short *gain,
    short *aoHex
);
```

Parameters

slot:

specifies the slot number (0 - 7).

ch:

specifies the channel number (0 – 7).

**gain:*

[Output] specifies the input type (0 - 5), where:

0: 0 ~ 5 V

1: 0 ~ 10 V

2: +/-5V

3: +/-10 V

4: +/-20 mA

5: 0 ~ 20 mA

**aoHex:*

[Output] the data in hexadecimal format.

Return Value

None

Example

[C/C++]

```
int slot, ch ,gain;
shor thVal;
i8028U_ReadAOHex(slot, ch, &gain, &hVal);
```

[C#]

```
Int slot;
Int16 ch;
int gain = 0;
short hVal = 0;
pac_i8028UNet.pac8028U.ReadAOHex(slot, ch, ref gain, ref hVal);
```

6.9. i8028U_ReadAO

This function is used to read the Analog Output value from a specified channel in decimal format.

Syntax

For MiniOS7

```
void i8028U_ReadAO(
    int slot,
    intch,
    short *gain,
    float *ao
);
```

For Windows (CE and WES)

```
void pac_i8028U_ReadAO(
    int slot,
    intch,
    short *gain,
    float *ao
);
```

Parameters

slot:

specifies the slot number (0 - 7).

ch:

specifies the channel number (0 – 7).

**gain:*

[Output] specifies the input type (0 - 5), where:

0: 0 ~ 5 V

1: 0 ~ 10 V

2: +/-5 V

3: +/-10 V

4: +/-20 mA

5: 0 ~ 20 mA

**ao:*

[Output] the data in decimal format.

Return Value

None

Example

[C/C++]

```
int slot, ch, gain;
float fVal = 0.0;
i8028U_ReadAO(slot, ch, &gain, &fVal);
```

[C#]

```
int slot,ch,gain;
float fVal = 0;
pac_i8028UNet.pac8028U.ReadAO(slot, ch, ref gain, ref fVal);
```

6.10. i8028U_SetPowerOnEnStatus

This function is used to set the Mode for the Power-on Value.

Syntax

For MiniOS7

```
short i8028U_SetPowerOnEnStatus(
    int slot,
    short status
);
```

For Windows (CE and WES)

```
short pac_i8028U_SetPowerOnEnStatus(
    int slot,
    short status
);
```

Parameters

slot:

specifies the slot number (0 - 7).

status:

specifies the type for the Power-on Mode (1 - 2), where:

1: Power-on Value Mode

2: Retentive Mode

Return Value

0 = No Error

For other return values, see the Error Codes listed in Appendix A.

Example

[C/C++]

```
int slot;
int type;
i8028U_SetPowerOnEnStatus(slot, type);
```

[C#]

```
Int slot;
Int type;
pac_i8028UNet.pac8028U.SetPowerOnEnableStatus(slot, type);
```

6.11. i8028U_GetPowerOnEnStatus

This function is used to read the current Mode being used for the Power-on Value.

Syntax

For MiniOS7

```
short i8028U_GetPowerOnEnStatus(
    int slot,
    short* status
);
```

For Windows (CE and WES)

```
short pac_i8028U_GetPowerOnEnStatus(
    int slot,
    short* status
);
```

Parameters

slot:

specifies the slot number (0 - 7).

**status:*

[output] specifies the type for the Power-on Mode (1 - 2), where:

1: Power-on Value Mode

2: Retentive Mode

Return Value

returns the type for Power-on Mode (1 - 2) where:

1: Power-on Value Mode

2: Retentive Mode

Example

[C/C++]

```
int slot;
short status;
i8028U_GetPowerOnEnStatus(slot,&status)
```

[C#]

```
Int slot;
short status;
pac_i8028UNet.pac8028U.GetPowerOnEnableStatus(slot, ref status);
```

6.12. i8028U_WritePowerOnHex_AO

This function is used to write the Power-on Value for a specified channel in hexadecimal format.

Syntax

For MiniOS7

```
short i8028U_WritePowerOnHex_AO(
    int slot,
    int ch,
    short gain,
    short aoHex
);
```

For Windows (CE and WES)

```
short pac_i8028U_WritePowerOnHex_AO(
    int slot,
    int ch,
    short gain,
    short aoHex
);
```

Parameters

slot:

specifies the slot number (0 - 7).

ch:

specifies the channel number (0 – 7)

gain:

specifies the input type (0 - 4), where:

0: 0 ~ 5 V

1: 0 ~ 10 V

2: +/-5 V

3: +/-10 V

4: +/-20 mA

5: 0 ~ 20 mA

aoHex:

the data in hexadecimal format.

Return Value

0 = No Error

For other return values, see the Error Codes listed inAppendix A.

Example

[C/C++]

```
int slot, ch;
short gain,hVal;
i8028U_WritePowerOnHex_AO(slot, ch, gain, hVal);
```

[C#]

```
Int slot, ch;
short gain, hVal;

pac_i8028UNet.pac8028U.WritePowerOnHex_AO(slot, ch, gain, hVal);
```

6.13. i8028U_WritePowerOn_AO

This function is used to write the Power-on Value for a specified channel in decimal format.

Syntax

For MiniOS7

```
short i8028U_WritePowerOn_AO(
    int slot,
    int ch ,
    short gain,
    float aoData
);
```

For Windows (CE and WES)

```
short pac_i8028U_WritePowerOn_AO(
    int slot,
    int ch ,
    short gain,
    float aoData
);
```

Parameters

slot:

specifies the slot number (0 - 7).

ch:

specifies the channel number (0 – 7).

gain:

specifies the input type (0 - 5), where:

0: 0~ 5 V

1: 0~ 10V

2: +/- 5V

3: +/-10V

4: +/-20 mA

5: 0 ~ 20mA

aoData:

the data in decimal format.

Return Value

0 = No Error,

For other return values, see the Error Codes listed in Appendix A.

Example

[C/C++]

```
int slot, ch;
short gain ;
float fVal;
i8028U_WritePowerOn_AO(slot, ch , gain, fVal);
```

[C#]

```
int slot ,ch;
short gain;
float fVal
pac_i8028UNet.pac8028U.WritePowerOn_AO(slot, ch, gain, fVal);
```

6.14. i8028U_ReadPowerOnHex_AO

This function is used to read the Power-on Value from a specified channel in hexadecimal format.

Syntax

For MiniOS7

```
void i8028U_ReadPowerOnHex_AO(
    int slot,
    int ch,
    short *gain,
    short *aoHex
);
```

For Windows (CE and WES)

```
void pac_i8028U_ReadPowerOnHex_AO(
    int slot,
    intch,
    short *gain,
    short *aoHex
);
```

Parameters

slot:

specifies the slot number (0 - 7).

ch:

specifies the channel number (0 – 7).

**gain:*

[Output] specifies the input type (0 - 5), where:

0: 0 ~ 5 V

1: 0 ~ 10 V

2: +/-5V

3: +/-10 V

4: +/-20 mA

5: 0 ~ 20 mA

**aoHex:*

[Output] the data in hexadecimal format.

Return Value

None

Example

[C/C++]

```
int slot, ch;
short gain ,hVal;
i8028U_ReadPowerOnHex_AO(slot, ch, &gain, &hVal);
```

[C#]

```
Int slot ,ch;
short gain ,hAO;
pac_i8028UNet.pac8028U.ReadPowerOnHex_AO(slot, ch, ref gain, ref hAO);
```

6.15. i8028U_ReadPowerOn_AO

This function is used to read the Power-on Value from a specified channel in decimal format.

Syntax

For MiniOS7

```
void i8028U_ReadPowerOn_AO(
    int slot,
    int ch,
    short *gain,
    float *aoData
);
```

For Windows (CE and WES)

```
void pac_i8028U_ReadPowerOn_AO(
    int slot,
    int ch,
    short *gain,
    float *aoData
);
```

Parameters

slot:

specifies the slot number (0 - 7).

ch:

specifies the channel number (0 – 7).

**gain:*

[Output] specifies the input type (0 - 5), where:

0: 0 ~ 5 V

1: 0 ~ 10 V

2: +/-5V

3: +/-10 V

4: +/-20 mA

5: 0 ~ 20 mA

**aoData:*

[Output] the data in decimal format.

Return Value

None

Example

[C/C++]

```
int slot, ch;
short gain;
float fVal;
i8028U_ReadPowerOn_AO(slot, ch, &gain, &fVal);
```

[C#]

```
Int slot ,ch;
short gain = 0;
float fVal = 0;
pac_i8028UNet.pac8028U.ReadPowerOn_AO(slot, ch, ref gain, ref fVal);
```

6.16. i8028U_WriteSafeHex_AO

This function is used to write the Safe Value for a specified channel in hexadecimal format.

Syntax

For MiniOS7

```
short i8028U_WriteSafeHex_AO(
    int slot,
    int ch,
    short gain,
    short aoHex
);
```

For Windows (CE and WES)

```
short pac_i8028U_WriteSafeHex_AO(
    int slot,
    int ch,
    short gain,
    short aoHex
);
```

Parameters

slot:

specifies the slot number (0 - 7).

ch:

specifies the channel number (0 – 7).

gain:

specifies the input type (0 - 4), where:

0: 0 ~ 5 V

1: 0 ~ 10 V

2: +/-5 V

3: +/-10 V

4: +/-20 mA

5: 0 ~ 20 mA

aoHex:

the data in hexadecimal format.

Return Value

0 = No Error

For other return values, see the Error Codes listed in Appendix A.

Example

[C/C++]

```
int slot, ch;
short gain ,hVal;
i8028U_WriteSafeHex_AO(slot, ch, gain, hVal);
```

[C#]

```
Int slot ,ch;
short gain ,hVal;

pac_i8028UNet.pac8028U.WriteSafeHex_AO(slot, ch, gain, hVal);
```

6.17. i8028U_WriteSafe_AO

This function is used to write the Safe Value for a specified channel in decimal format.

Syntax

For MiniOS7

```
short i8028U_WriteSafe_AO(
    int slot,
    int ch,
    short gain,
    float aoData
);
```

For Windows (CE and WES)

```
short pac_i8028U_WriteSafe_AO(
    int slot,
    int ch,
    short gain,
    float aoData
);
```

Parameters

slot:

specifies the slot number (0 - 7).

ch:

specifies the channel number (0 – 7).

gain:

specifies the input type (0 - 5), where:

0: 0 ~ 5 V

1: 0 ~ 10 V

2: +/-5V

3: +/-10 V

4: +/-20 mA,

5: 0 ~ 20 mA

aoData:

the data in decimal format.

Return Value

0 = No Error

For other return values, see the Error Codes listed in Appendix A.

Example

[C/C++]

```
int slot, ch;
short gain;
float fVal;
i8028U_WriteSafe_AO(slot, ch, gain, fVal);
```

[C#]

```
Int slot ,ch;
int gain;
float fVal;
pac_i8028UNet.pac8028U.WriteSafe_AO(slot, ch, gain, fVal);
```

6.18. i8028U_ReadSafeHex_AO

This function is used to read the Safe Value from a specified channel in hexadecimal format.

Syntax

For MiniOS7

```
void i8028U_ReadSafeHex_AO(
    int slot,
    int ch,
    short *gain,
    short *aoHex
);
```

For Windows (CE and WES)

```
void pac_i8028U_ReadSafeHex_AO(
    int slot,
    int ch,
    short *gain,
    short *aoHex
);
```

Parameters

slot:

specifies the slot number (0 - 7).

ch:

specifies the channel number (0 – 7).

**gain:*

[Output] specifies the input type (0 - 5), where:

0: 0 ~ 5 V

1: 0 ~ 10 V

2: +/-5 V

3: +/-10 V

4: +/-20 mA

5: 0 ~ 20 mA

**aoHex:*

[Output] the data in hexadecimal format.

Return Value

None

Example

[C/C++]

```
int slot, ch;
short gain ,hVal;
i8028U_ReadSafeHex_AO(slot, ch, &gain, &hVal);
```

[C#]

```
Int slot ,ch;
short gain ,hVal;
pac_i8028UNet.pac8028U.ReadSafeHex_AO(slot, ch, ref gain, ref hVal);
```

6.19. i8028U_ReadSafe_AO

This function is used to read the Safe Value from a specified channel in decimal format.

Syntax

For MiniOS7

```
void i8028U_ReadSafe_AO(
    int slot,
    int ch,
    short *gain,
    float *aoData
);
```

For Windows (CE and WES)

```
void pac_i8028U_ReadSafe_AO(
    int slot,
    int ch,
    short *gain,
    float *aoData
);
```

Parameters

slot:

specifies the slot number (0 - 7).

ch:

specifies the channel number (0 – 7).

**gain:*

[Output] specifies the input type (0 - 5), where:

0: 0 ~ 5 V

1: 0 ~ 10 V

2: +/-5 V

3: +/-10 V

4: +/-20 mA

5: 0 ~ 20 mA

**aoData:*

[Output] the data in decimal format.

Return Value

None

Example

[C/C++]

```
int slot, ch;
short gain;
float fVal;
i8028U_ReadSafe_AO(slot, ch, &gain, &fVal);
```

[C#]

```
Int slot ,ch;
int gain;
float fVal;
pac_i8028UNet.pac8028U.ReadSafe_AO(slot,ch, ref gain, ref fVal);
```

6.20. i8028U_SetModuleWDTConfig

This function is used to configure the attributes for the Watchdog parameter on the I-9028 module.

Syntax

For MiniOS7

```
short i8028U_SetModuleWDTConfig(
    int slot,
    short enStatus,
    unsigned long wdtTimeout,
    int ifWDT_Overwrite
);
```

For Windows (CE and WES)

```
shprt pac_i8028U_SetModuleWDTConfig(
    int slot,
    short enStatus,
    unsigned long wdtTimeout,
    int ifWDT_Overwrite
);
```

Parameters

slot:

specifies the slot number (0 - 7).

enStatus:

sets the status of the Watchdog (0 or 1), where:

0: disables the Watchdog

1: enables the Watchdog

wdtTimeout:

sets duration of the Watchdog timeout in hexadecimal format (0~0xff), which is equal to 0 ~ 25.5 seconds

ifWDT_Overwrite:

determines whether or not an AO value can be written if a WDT timeout is currently active (0 or 1), where:

0: an AO value cannot be written if a WDT timeout is currently active

1: an AO can be written even if a WDT timeout is currently active

Return Value

0 = No Error

For other return values, see the Error Codes listed in Appendix A.

Example

[C/C++]

```
int slot;
short enStatus;
unsigned long wdtTimeout;
int ifWDT_Overwrite;
i8028U_SetModuleWDTConfig(slot, enStatus, wdtTimeout, ifWDT_Overwrite);
```

[C#]

```
Int slot;
short enStatus;
UInt32 timeout;
Int ifWDT;
pac_i8028UNet.pac8028U.SetModuleWDTConfig(slot, enStatus, timeout, ifWDT);
```

6.21. i8028U_GetModuleWDTConfig

This function is used to read the current attributes of the Watchdog parameter on the I-9028 module.

Syntax

For MiniOS7

```
short i8028U_GetModuleWDTConfig(
    int slot,
    short *enStatus,
    unsigned long *wdtTimeout,
    int*ifWDT_Overwrite
);
```

For Windows (CE and WES)

```
short pac_i8028U_GetModuleWDTConfig(
    int slot,
    short *enStatus,
    unsigned long *wdtTimeout,
    int*ifWDT_Overwrite
);
```

Parameters

slot:

specifies the slot number (0 - 7).

**enStatus:*

[Output] the current status of the Watchdog (0 or 1), where:

0: Watchdog disabled

1: Watchdog enabled

**wdtTimeout:*

[Output] the current duration that is set for the Watchdog timeout in hexadecimal format (0~0xff), which is equal to 0 ~ 25.5 seconds

**ifWDT_Overwrite:*

[Output] specifies whether or not an AO value can be written if a WDT timeout is currently active (0 or 1), where:

0: an AO value cannot be written if a WDT timeout is currently active

1: an AO can be written even if a WDT timeout is currently active

Return Value

0 = No Error

For other return values, see the Error Codes listed in Appendix A.

Example

[C/C++]

```
int slot;
short enStatus;
unsigned long wdtTimeout
int ifWDT_Overwrite;
i8028U_GetModuleWDTConfig(slot, &enStatus, &wdtTimeout, &ifWDT_Overwrite);
```

[C#]

```
Int slot;
short enStatus;
UInt32 timeout;
Int ifWDT;
pac_i8028UNet.pac8028U.GetModuleWDTConfig(slot, ref enStatus, ref timeout, ref ifWDT);
```

6.22. i8028U_GetModuleWDTStatus

This function is used to retrieve the current status of the Watchdog on the I-9028 module.

Syntax

For MiniOS7

```
short i8028U_GetModuleWDTStatus(
    int slot
);
```

For Windows (CE and WES)

```
short pac_i8028U_GetModuleWDTStatus(
    int slot
);
```

Parameters

slot:

specifies the slot number (0 - 7).

Return Value

Indicates the status of the Watchdog timeout (0 - 1), where:

0: a Watchdog timeout is not currently active

1: a Watchdog timeout is currently active

Example

[C/C++]

```
int slot ,ret;
ret = i8028U_GetModuleWDTStatus(slot);
```

[C#]

```
Intslot ,ret;
ret = pac_i8028UNet.pac8028U.IsWDTTimeout(slot);
```

6.23. i8028U_ResetModuleWDT

This function is used to reset the status of the Watchdog on the I-9028 module.

Syntax

For MiniOS7

```
void i8028U_ResetModuleWDT(
    int slot
);
```

For Windows (CE and WES)

```
void pac_i8028U_ResetModuleWDT(
    int slot
);
```

Parameters

slot:

specifies the slot number (0 - 7).

Return Value

0 = No Error

For other return values, see the Error Codes listed in Appendix A.

Example

[C/C++]

```
int slot;
i8028U_ResetModuleWDT(slot);
```

[C#]

```
Int slot;
pac_i8028UNet.pac8028U.ResetWDT(slot);
```

6.24. i8028U_RefreshModuleWDT

This function is used to refresh the status of the Watchdog on the I-9028 module.

Syntax

For MiniOS7

```
void i8028U_RefreshModuleWDT(
    int slot
);
```

For Windows (CE and WES)

```
void pac_i8028U_RefreshModuleWDT(
    int slot
);
```

Parameters

slot:

specifies the slot number (0 - 7).

Return Value

0 = No Error

For other return values, see the Error Codes listed in Appendix A.

Example

[C/C++]

```
int slot;
i8028U_RefreshModuleWDT(slot);
```

[C#]

```
Int slot;
pac_i8028UNet.pac8028U.RefreshWDT(slot);
```

Appendix A. Error Code

Error Code	Definition	Description
0	No Error	This error code indicates that there are no errors.
-1	ID Error	This error code indicates that the ID of the module inserted into the specified slot is not for an I-9028U module.
-2	FRAM_ERROR	??
-3	MODULE_STATUS_ERROR	??
-7	WDTTIMEOUT	??

Appendix B. Revision History

This chapter provides revision history information to this document.

The table below shows the revision history.

Revision	Date	Description
1.0.1	January 2018	Initial issue
2.0.0	July 2018	• Modify library , demo path • Added WP-9000 , ippc-wes7 library , demo path • Modify API