I-9024U/I-9028U
I/O Module User Manual

V 1.0.2April 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-9028U is an 8-channel source type 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-9028U module features channel to-channel isolation as well as 4 kV ESD protection and 1000 VDC intra-module isolation.

1.1. Features

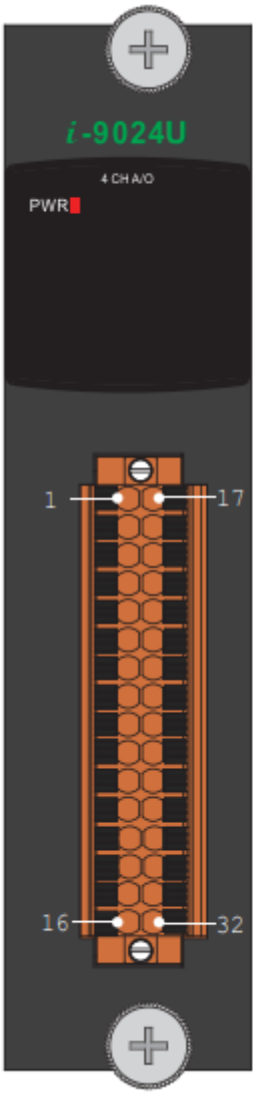
- 8-channel Voltage or Current Output
- Jumper-selectable Voltage or Current Output
- Individual Channel Configuration
- 2500 VDC Intra-module Isolation
- Open Wire Detection for Current Output
- RF Immunity
- Short Circuit Protection
- Able to set Power-on and Safe Values
- 4 kV ESD Protection
- Wide Operating Temperature Range: -25 to +75°C

1.2. Specifications

Analog Output	
Channels	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)	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

1.3. Pin Assignments

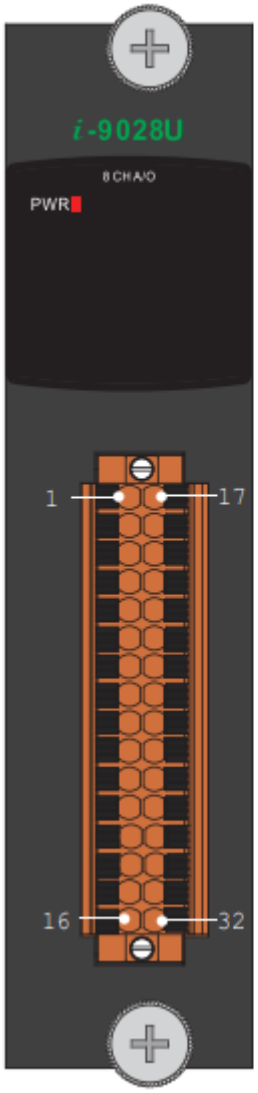
I-9024U



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

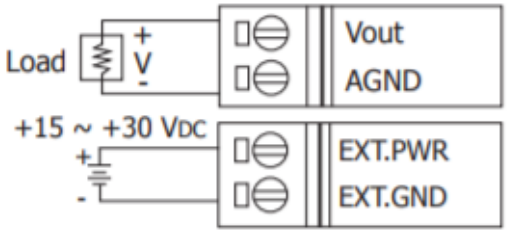


Pin Assignment	Terminal No.		Pin Assignment
Vout0	01	17	Vout1
Iout0	02	18	Iout1
AGND	03	19	AGND
Vout2	04	20	Vout3
Iout2	05	21	Iout3
AGND	06	22	AGND
Vout4	07	23	Vout5
Iout4	08	24	Iout5
AGND	09	25	AGND
Vout6	10	26	Vout7
Iout6	11	27	Iout7
AGND	12	28	AGND
EXT.PWR	13	29	EXT.PWR
EXT.PWR	14	30	EXT.PWR
EXT.GND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

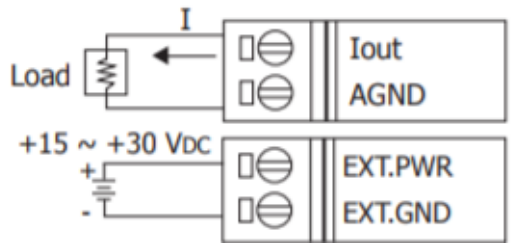
32-pin Connector

1.4. Wire Connections

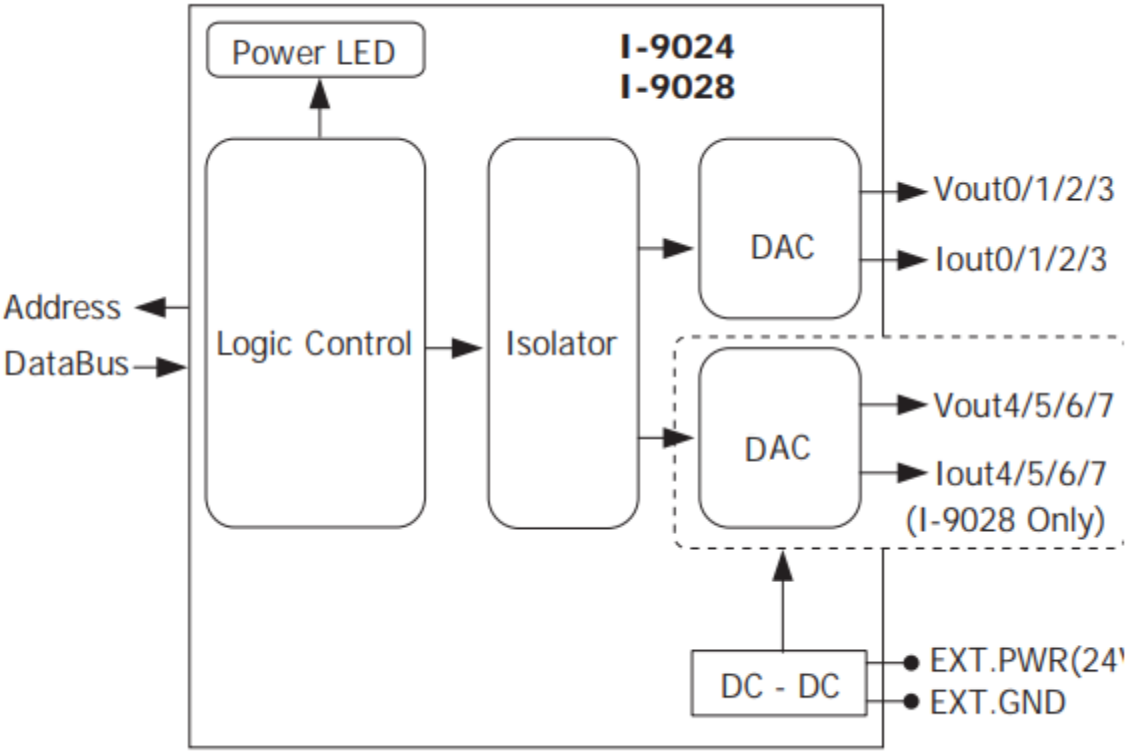
Voltage Output Wiring



Current Output Wiring



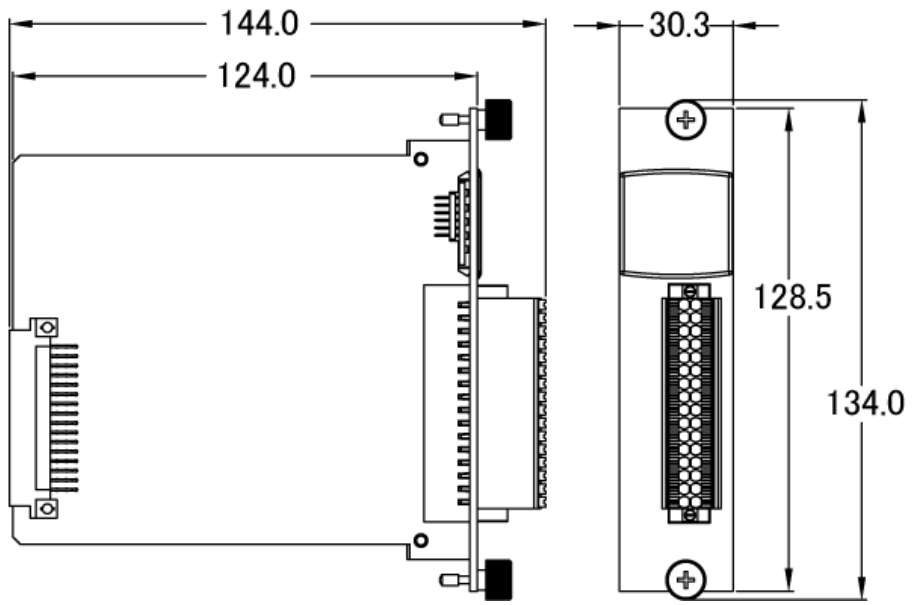
1.5. Block Diagram



1.6. Dimensions

All dimensions are in millimeters.

I-9028 with Spring clamp terminal connector



Left Side View

Top View

1.7. 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
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/
Demo io-8k	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/
Demo io-9k	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/

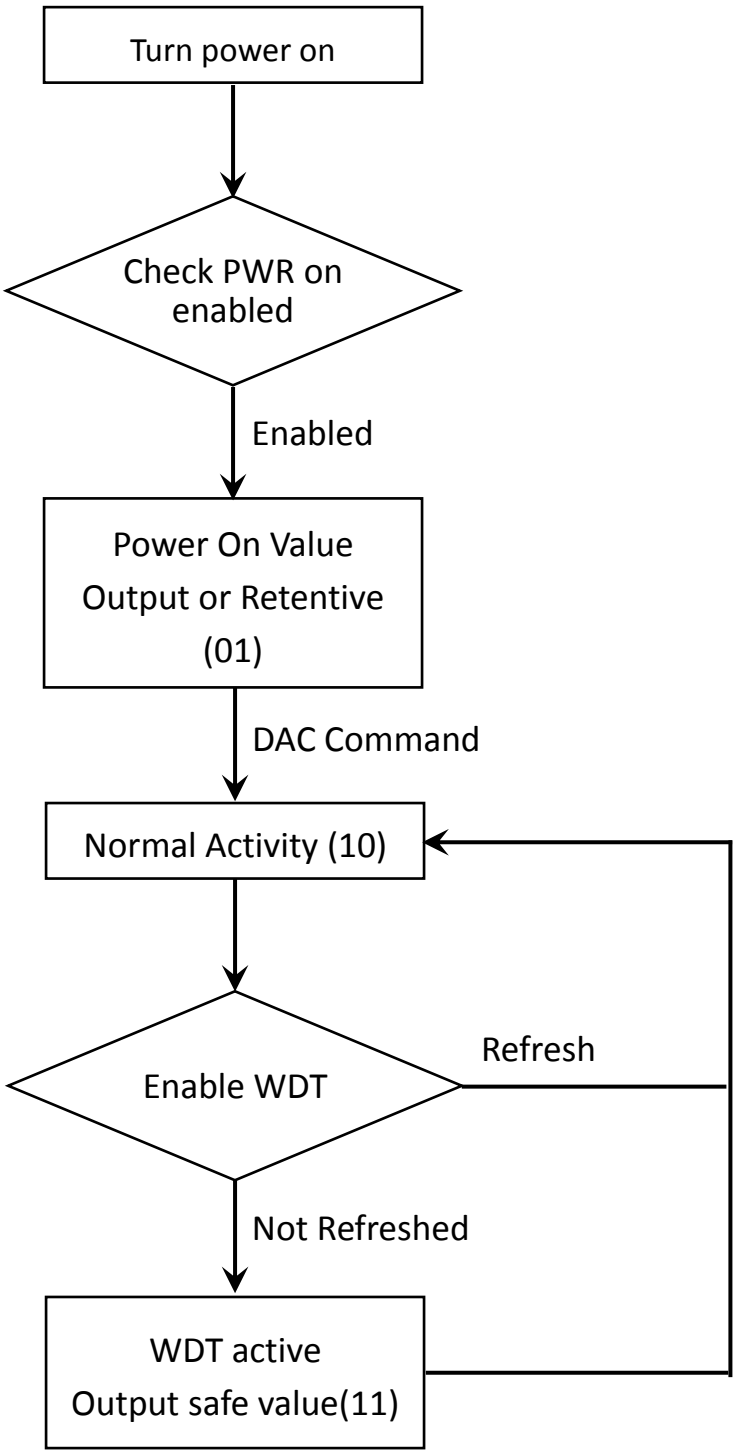
C# Demo:

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/

2. Quick Start

The I-9028U is an 8-channel Analog Output module that can output both voltage and current. The following is an illustration of the operating procedure for the I-9028U module:



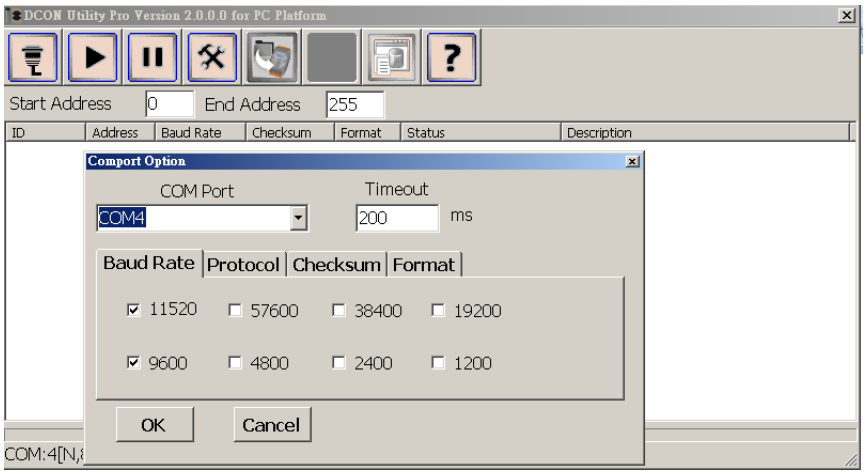
2.1. I-9028U Analog Output using the DCON Utility

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.

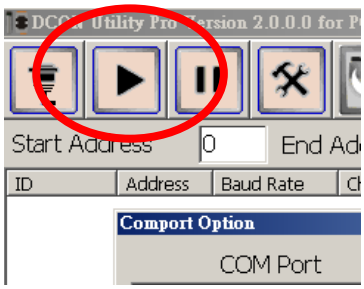
The DCON Utility Pro can be downloaded from the ICP DAS website at:
ftp://ftp.icpdas.com/pub/cd/ippc-wes7/tools/dcon_utility_pro/

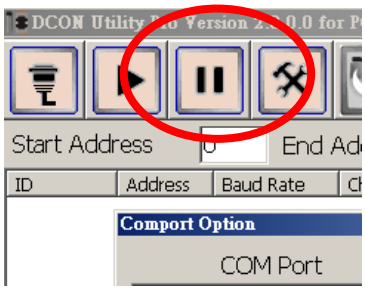
Follow the procedure described below to test the I-9028U

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

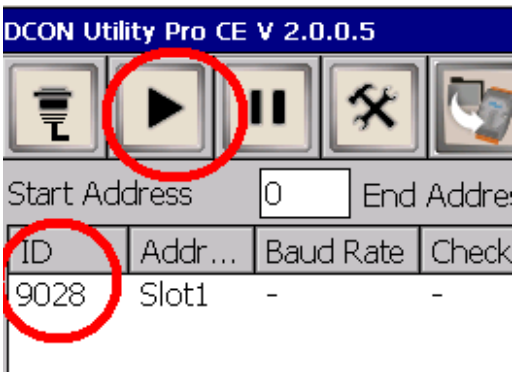


2. Click the “Search” button to start searching for I/O devices connected to the network, and then Click the “Stop Search” button to stop the search.

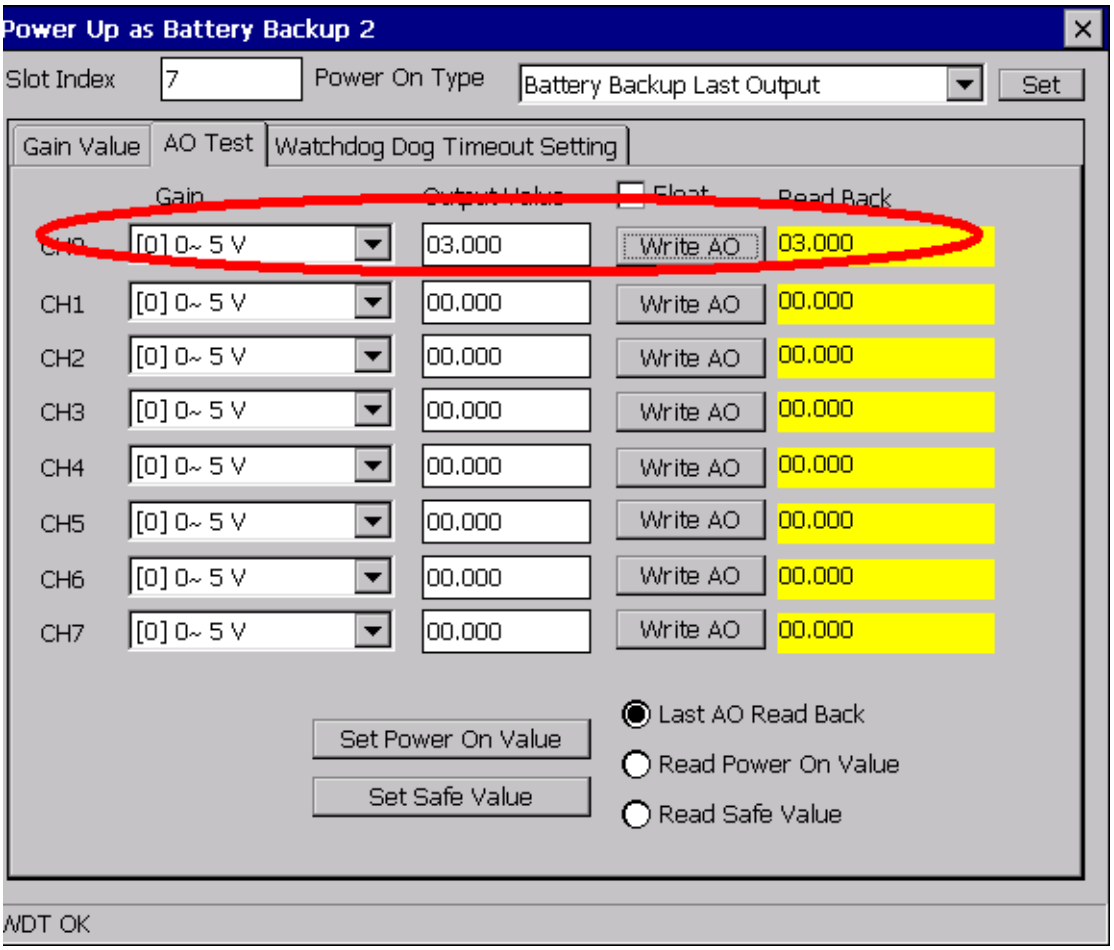




3. If any I/O devices are located, they will be displayed in the device list window. Double-click the name of the module to open the configuration dialog box



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. Power-on I-9028U Analog Output using the API

ICP DAS provides APIs, libraries and demo programs, including the source code, that allow integration of the I-9028U into Windows platforms.

The pac_i8028U_WriteAOHex function can be used to directly transmit the Analog Output to your device after calling pac_i8028U_Init function. For more detailed information regarding the use of these functions, refer to Chapter 5.

For example:

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

The I-9028U module includes both a Power-on Mode and a WDT that can be userd to solve a range of conditions that can make the system become stable. The three types of Power-on Mode provided by the I-9028U can be used to prevent unknown errors that might cause the Analog Output from the I-9028U module to inflict damage on other devicesPower-on . An introduction to the Power-on Mode is provided in Chapter 3.

The I-9028U 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.

Note that the both the Power-on Mode and the WDT will affect the AO value for the I-9028U module. An overview of the AO values that may be encountered under both normal or abnormal conditions when either Power-on Mode or the WDT are active are outlined in Section 2.3.

3. Power-on Mode

Three types of Power-on Mode are provided on the I-9028U module. These are:

1. Power-on Value Mode
2. Retentive Mode

If the I-9028U 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 that might cause the Analog Output from the I-9028U module to inflict damage on other devices. This ensures that the output from the I-9028U 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-9028U 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.

3.1. Configuring Power-on Mode

Two types of Power-on Mode are provided on the I-9028U module, 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:

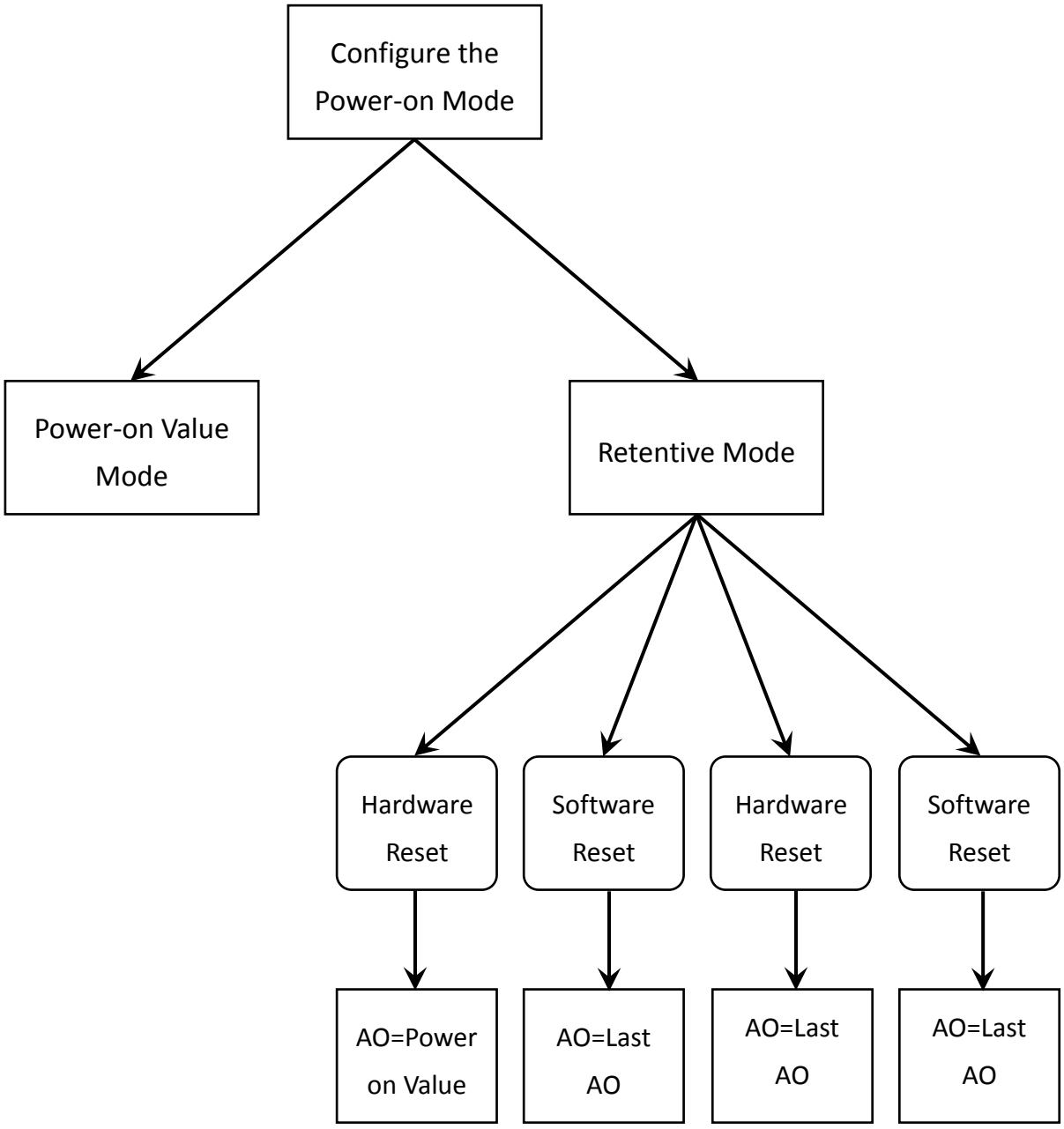
Mode	Output
Power-on Value Mode	The Power-on Value will be used as the AO value when the system is reset
Retentive Mode	The previous AO value will be retained

If either the 9000-series PAC or the I-9028U module encounters an event that causes the hardware or the software to be reset, the values for the Power-on Mode for the Analog 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 = Previous AO value
Retentive Mode	AO = Previous AO value	AO = Previous AO

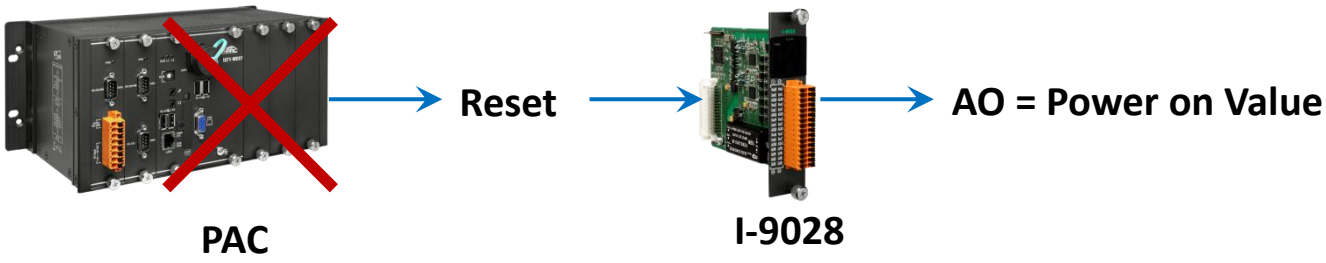
The following is a flowchart that illustrates the outcome of each Power-on Mode:



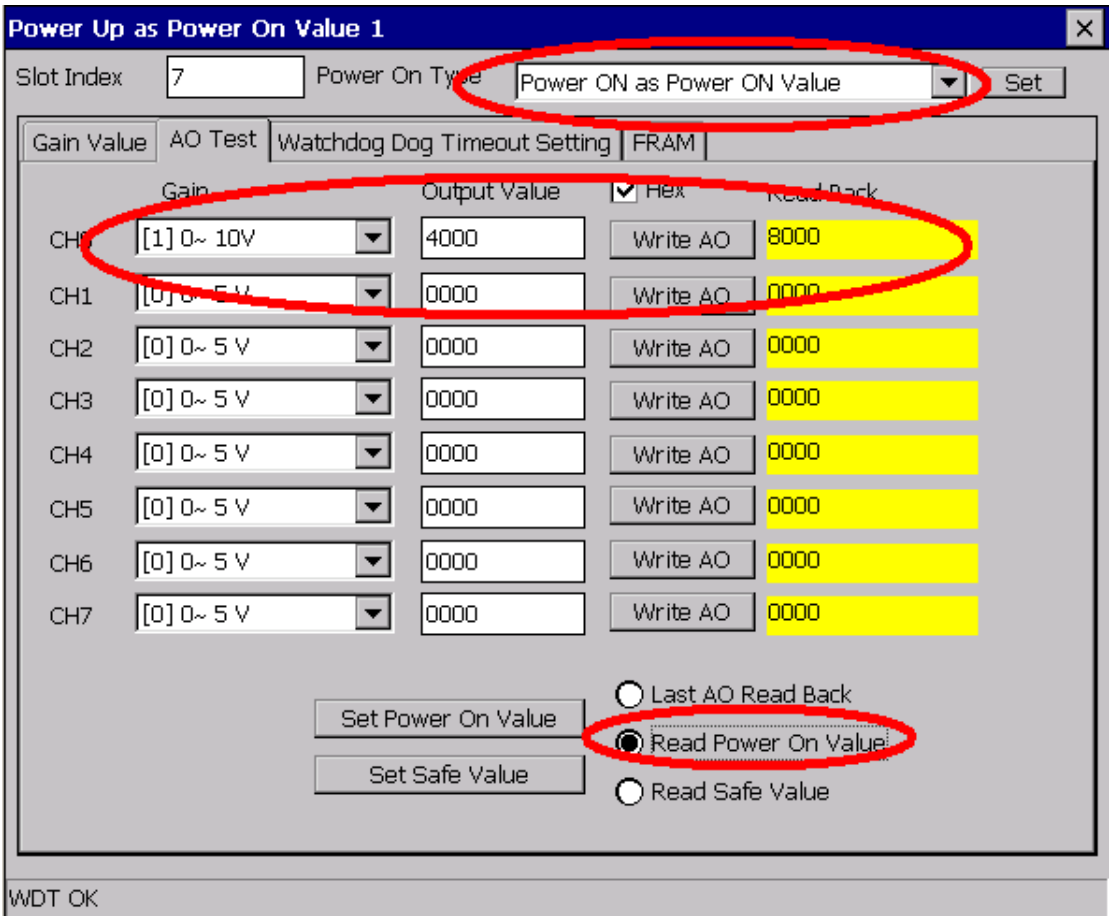
Note: the term Previous AO value means that the AO value that was active prior to the reset will be retained. For example, if the Power-on Mode is set as Retentive Mode and a hardware reset occurs, then the AO value will be the same as the value that existed prior to the hardware reset.

3.1.1. Power-on Value Mode

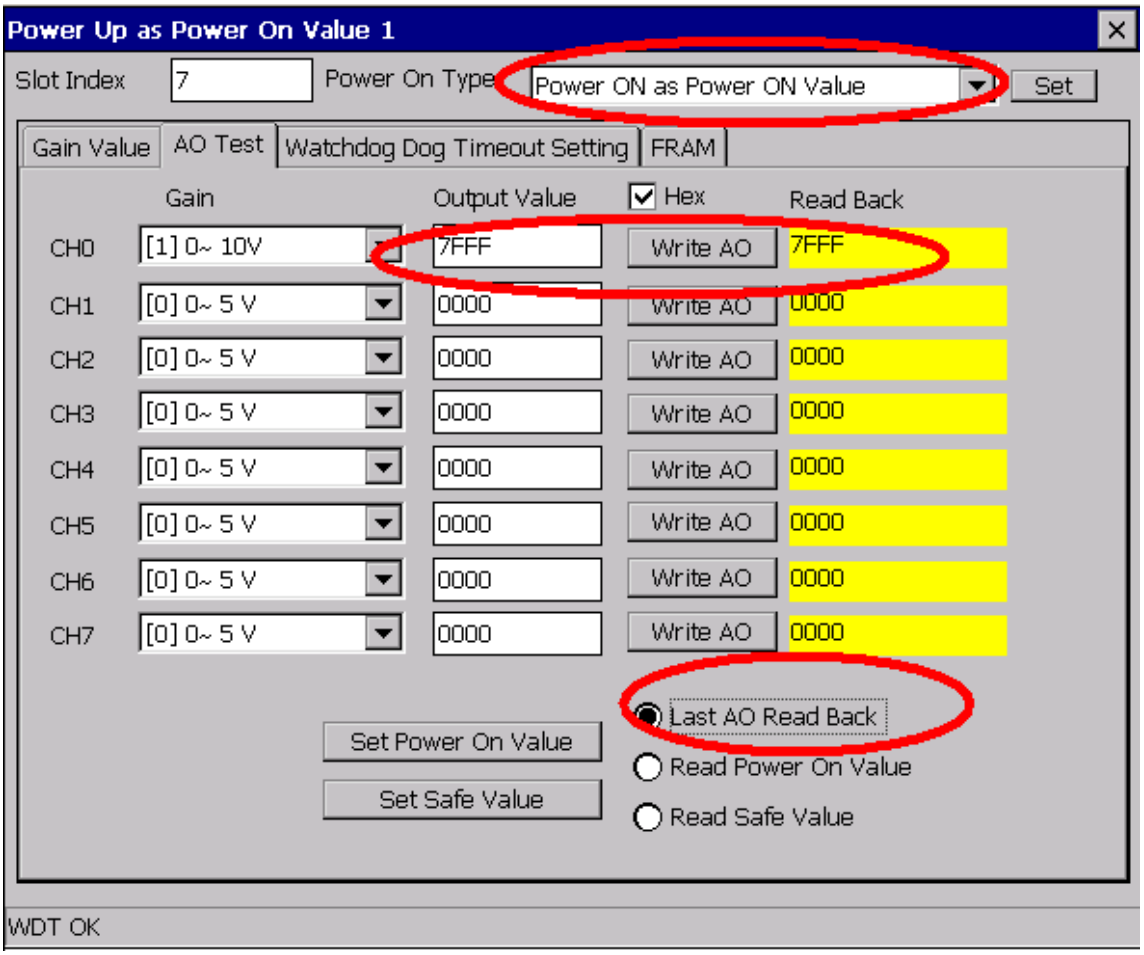
Power-on Value Mode is used to set the AO value to the preconfigured Power-on Value after an unknown condition has caused the I-9028U module to be reset.



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.

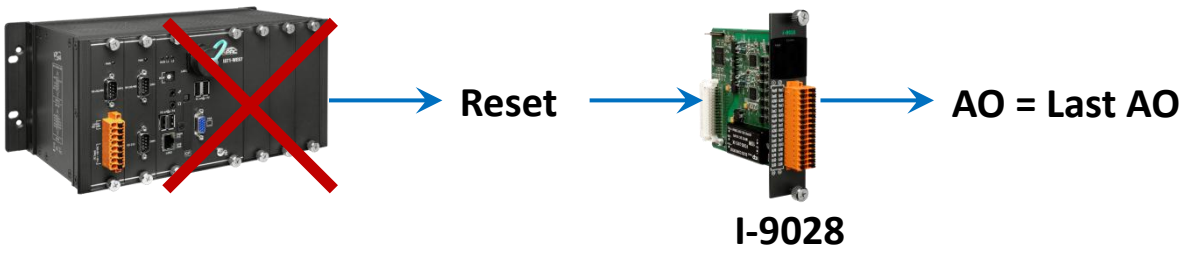


After the power to the 9000-series PAC is reset, the output value for channel 0 will be Gain 1 (0~10 V) and the output value will be set to 0x7FFF, as illustrated in the figure below.

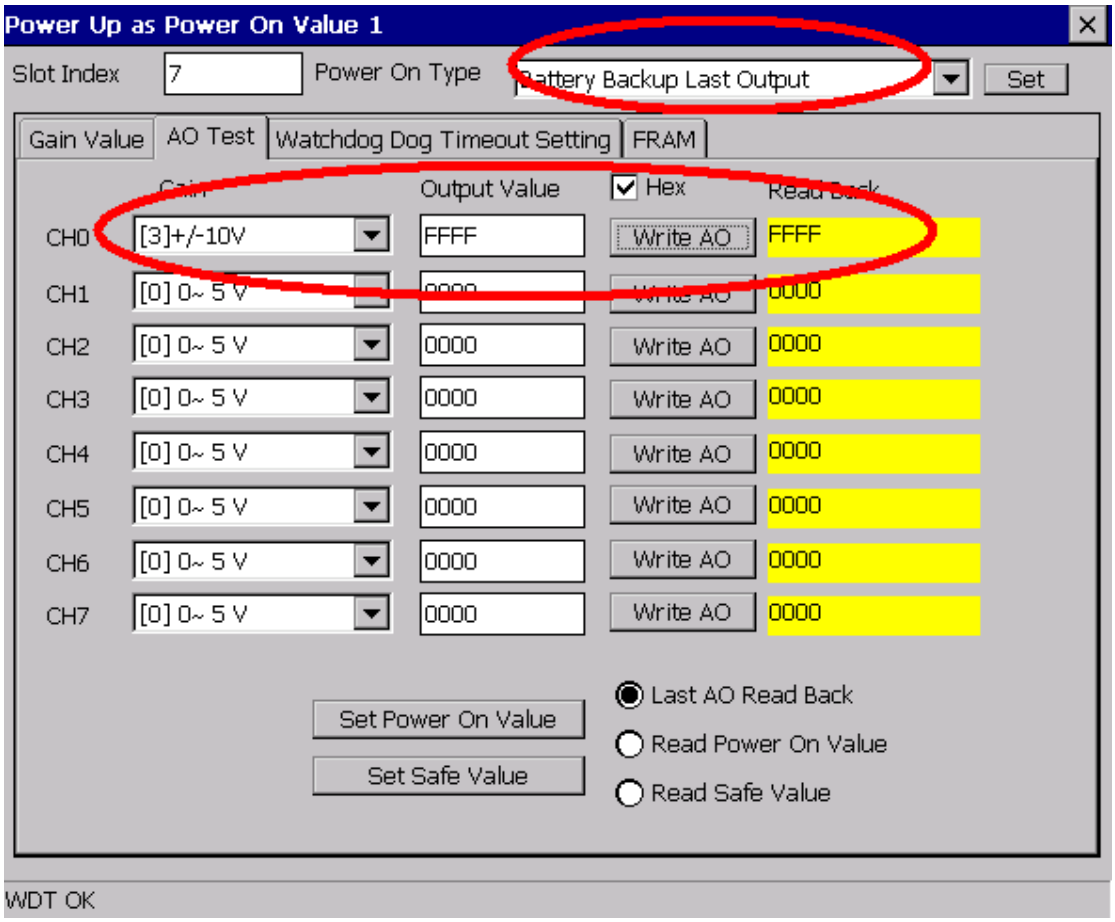


3.1.2. Retentive Mode

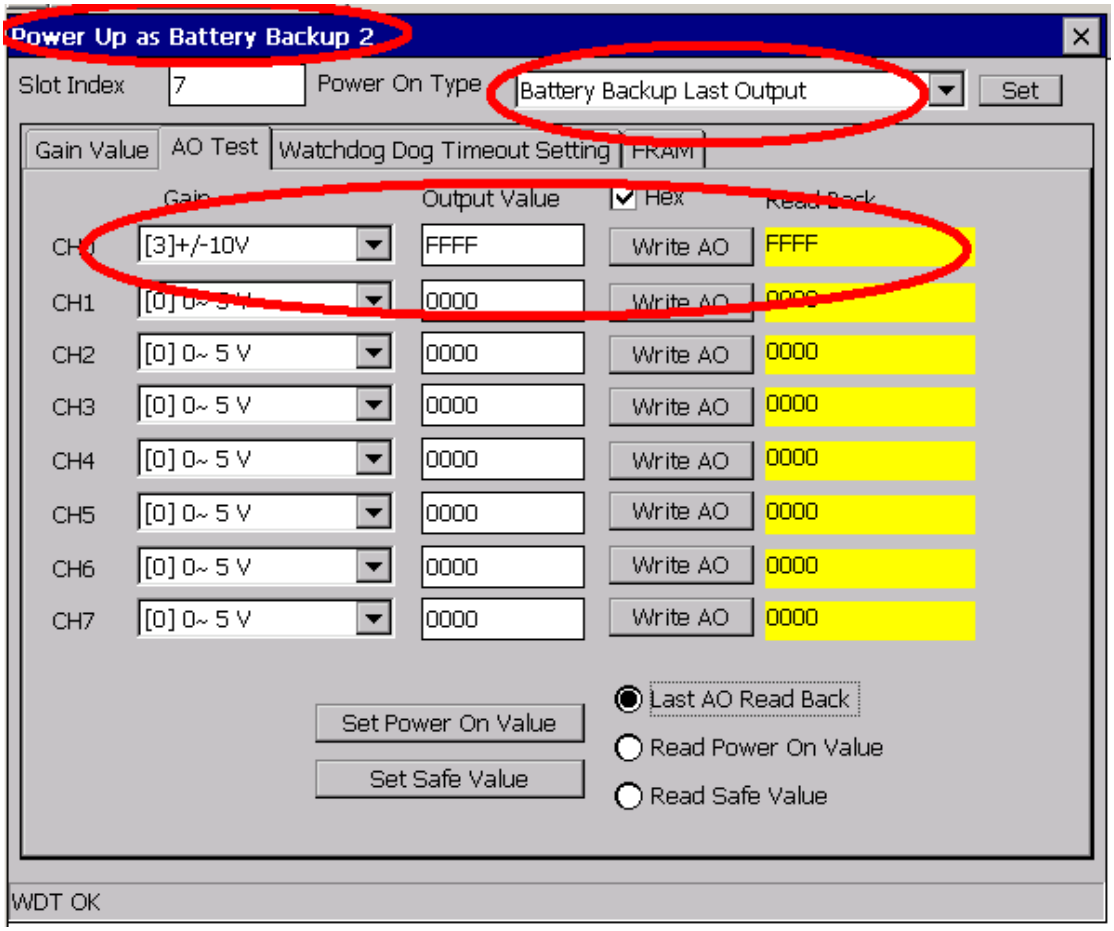
Retentive Mode, also known as Virtual Battery Backup Mode, is used to ensure that the previous AO value is retained if an unknown condition has caused the I-9028U module to be reset.



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

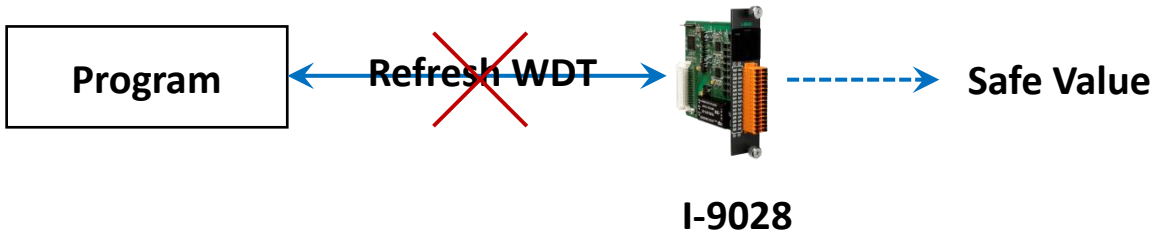


After the power to the 9000-series PAC is reset, the AO value for channel 0 will be Gain 3 (+/-10 V) and the Output value will be set to 0xFFFF, as Illustrated in the figure below.

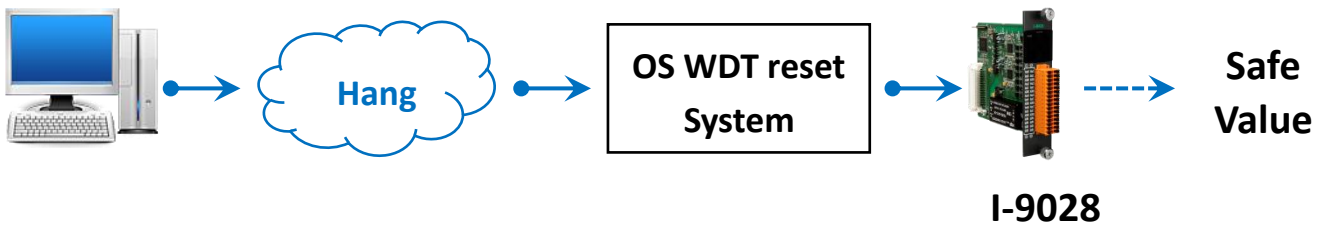


4. Watchdog

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 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.



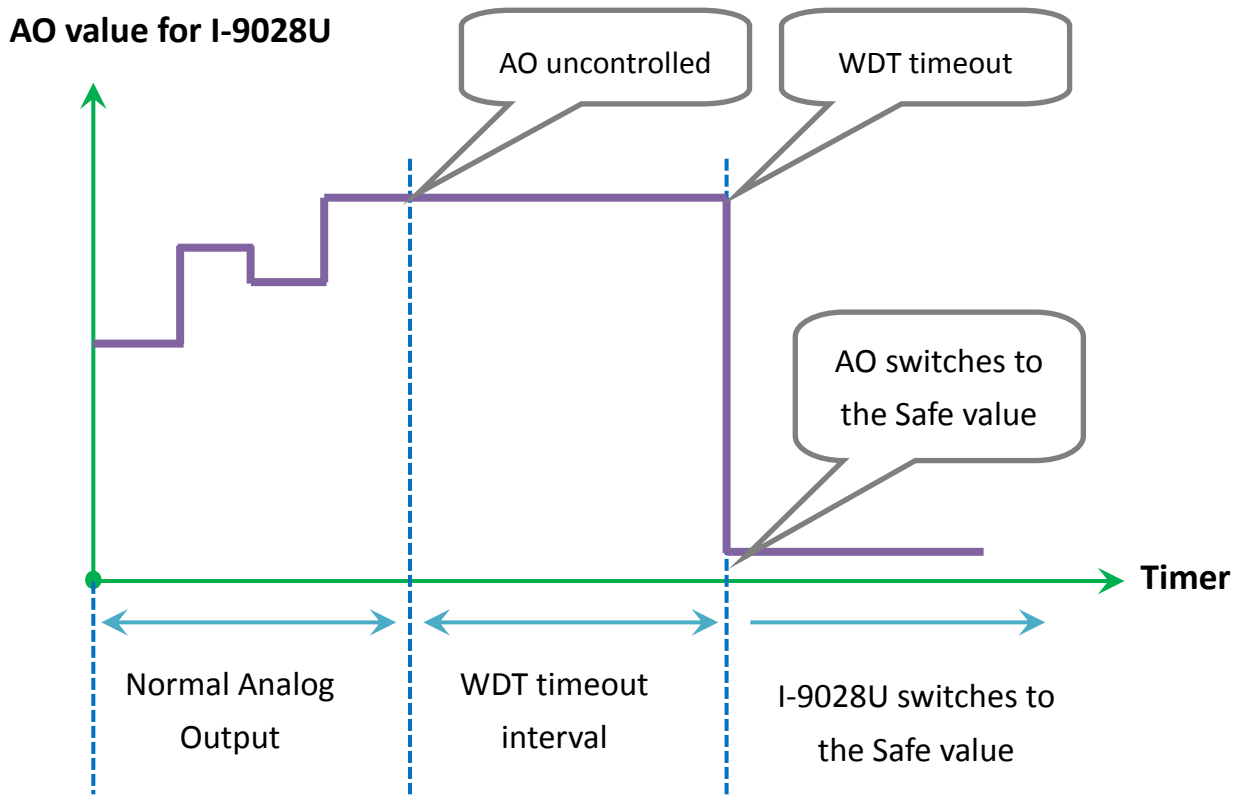
A common application is provided as an example below. If the system encounters an even that causes it to become unresponsive for any reason, the I-9028U WDT and the WDT provided by the operating system for the 9000-series PAC can be used in combination, thereby preventing the system from hang becoming 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

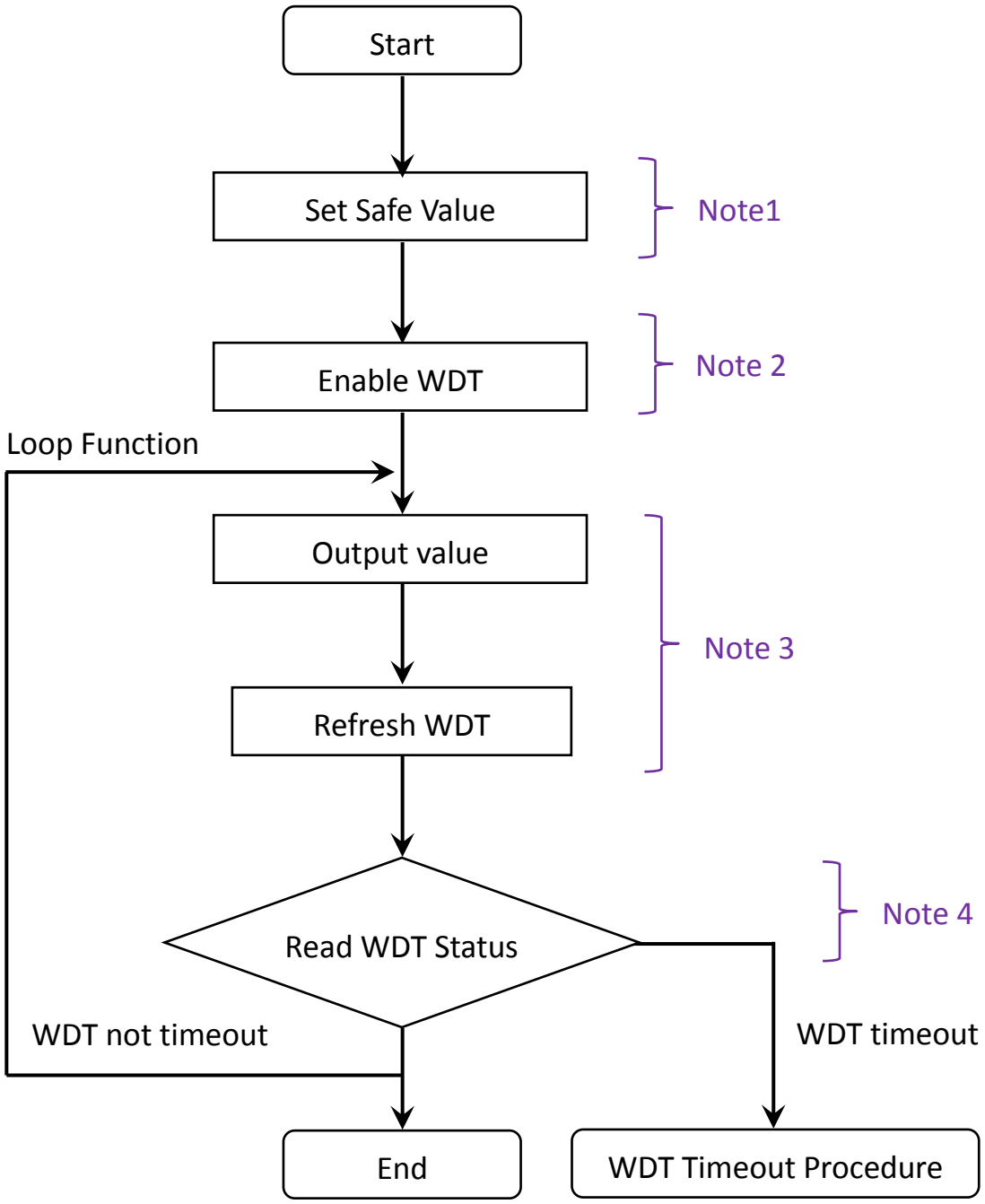
The Analog Output value for the I-9028U module will be as below. When the WDT timeout is triggered, user can consider necessary to set suit WDT timeout interval and safe value to prevent the AO from becoming uncontrolled.

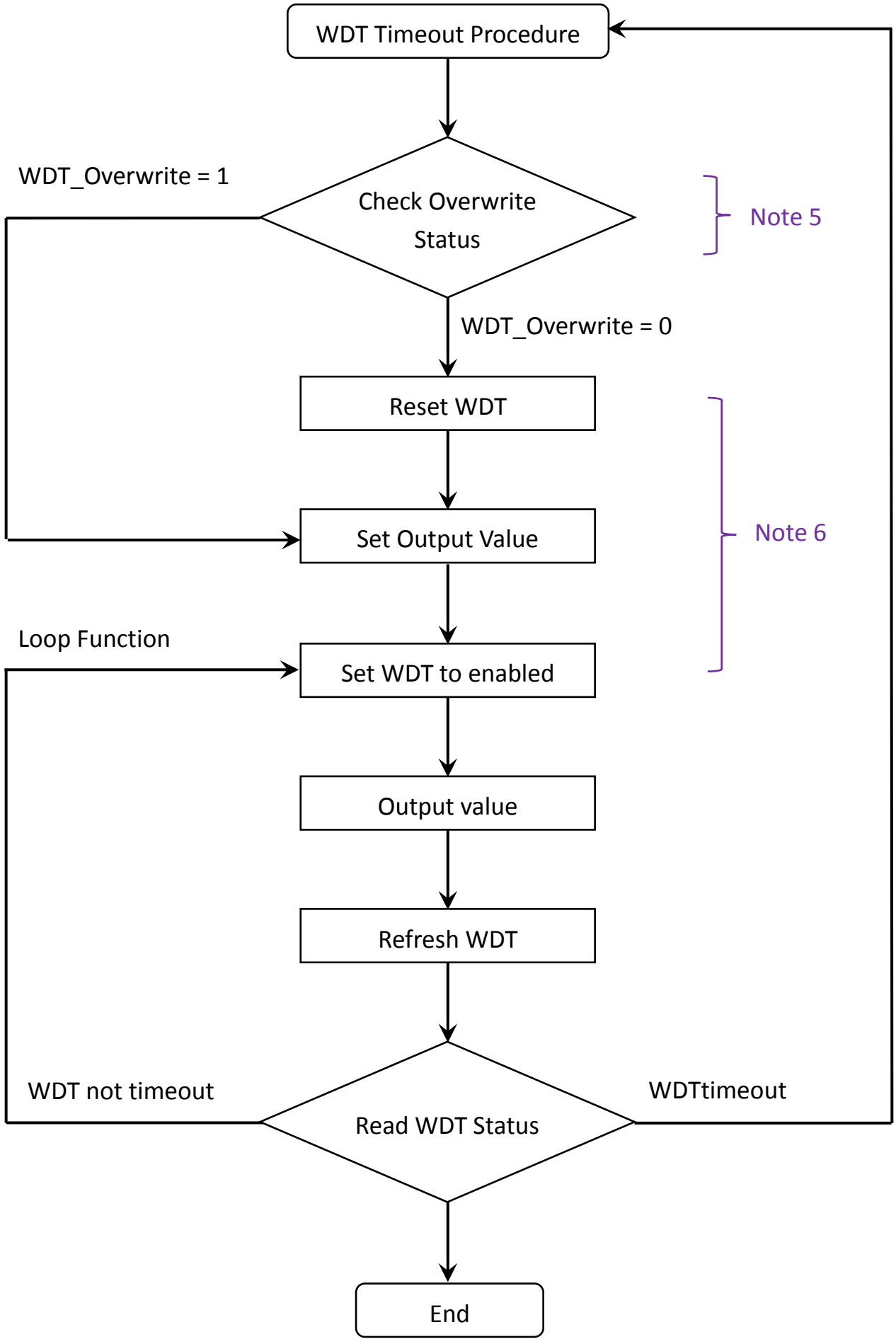


Watchdog operations include basic management operations, such as turning on and refreshing the module. The following sections provide a description of how to programmatically operate the watchdog using the watchdog functions.

Watchdog procedure

The following is an overview of the Watchdog process. More detailed 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 detailed explanation of these functions, refer to the relevant API reference in Chapter 5.



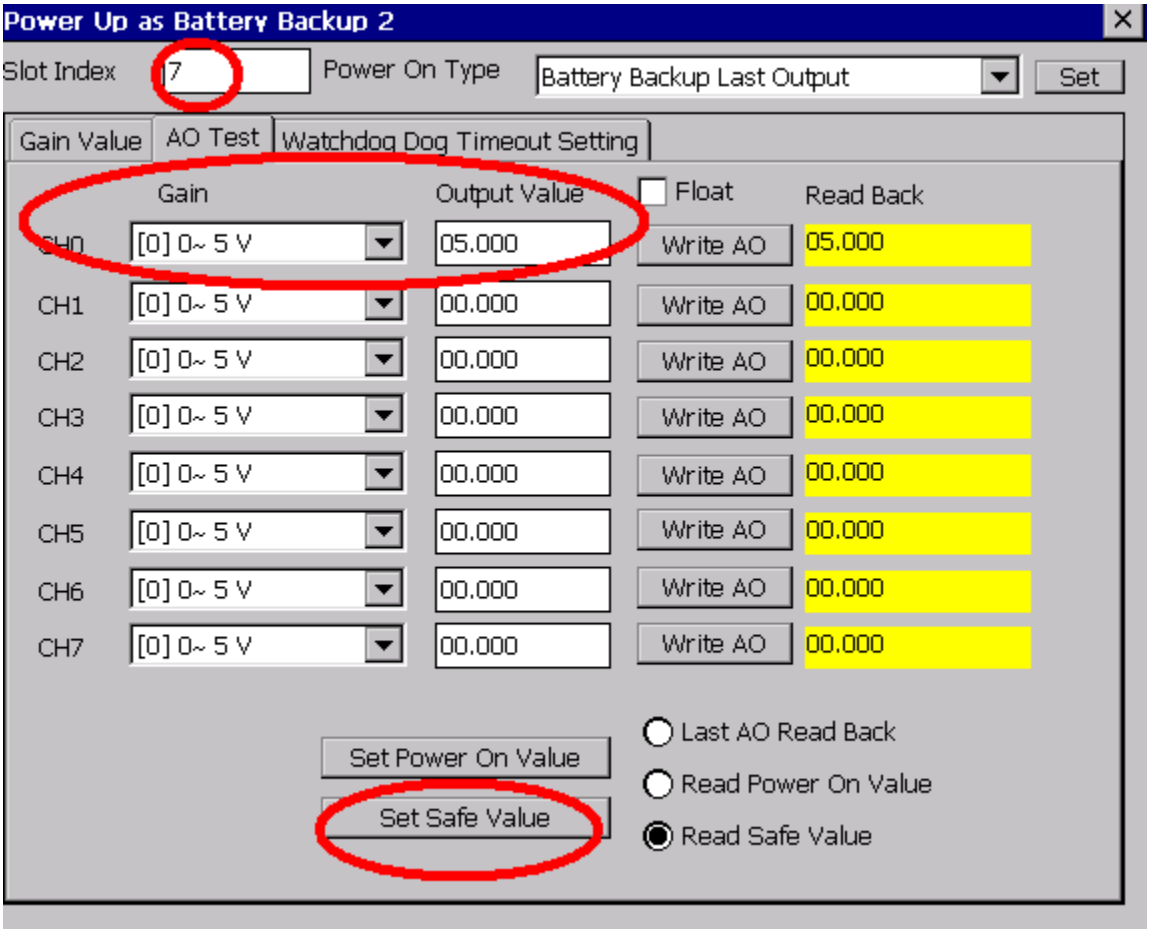

WDT Timeout Procedure
WDT_Overwrite = 1
Check Overwrite Status
Note 5
WDT_Overwrite = 0
Reset WDT
Set Output Value
Note 6
Loop Function
Set WDT to enabled
Output value
Refresh WDT
WDT not timeout
Read WDT Status
WDTtimeout
End

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.

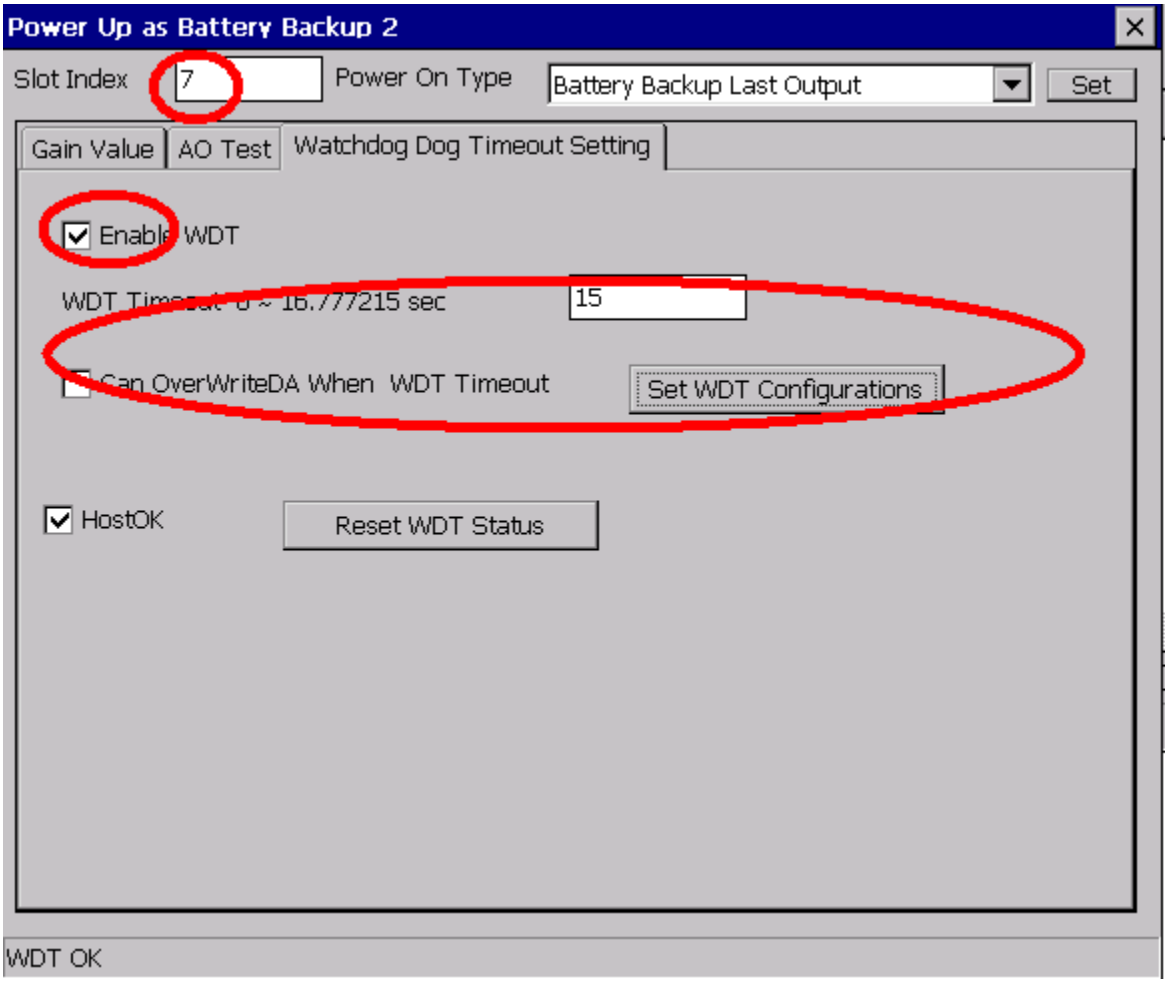
To set the Safe Value, Use the pac_i8028U_WriteSafe_AO (7,0,0,5.0) function to write to channel 0 on slot 0 and set it to Gain 0 (0~5 V) and set the Safe Value for the I-9028U module to 5.0 V.



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

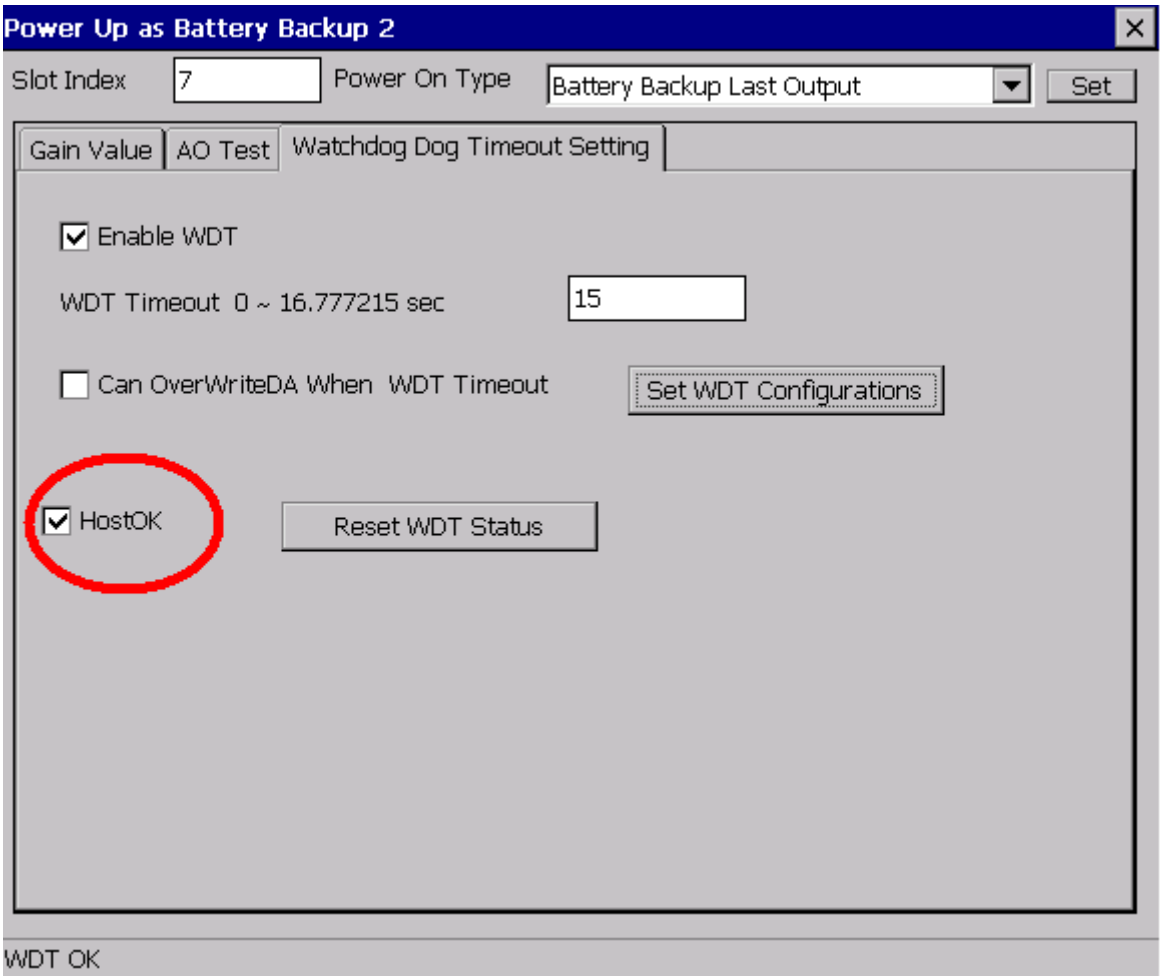
Use the pac_i8028U_SetModuleWDTConfig (7,1,10.0,1) function to enable the WDT, set the WDT timeout value to 10.0 sec, and set the WDT_Overwrite parameter to 0. You then need to enable the WDT function and start the WDT timer.

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.

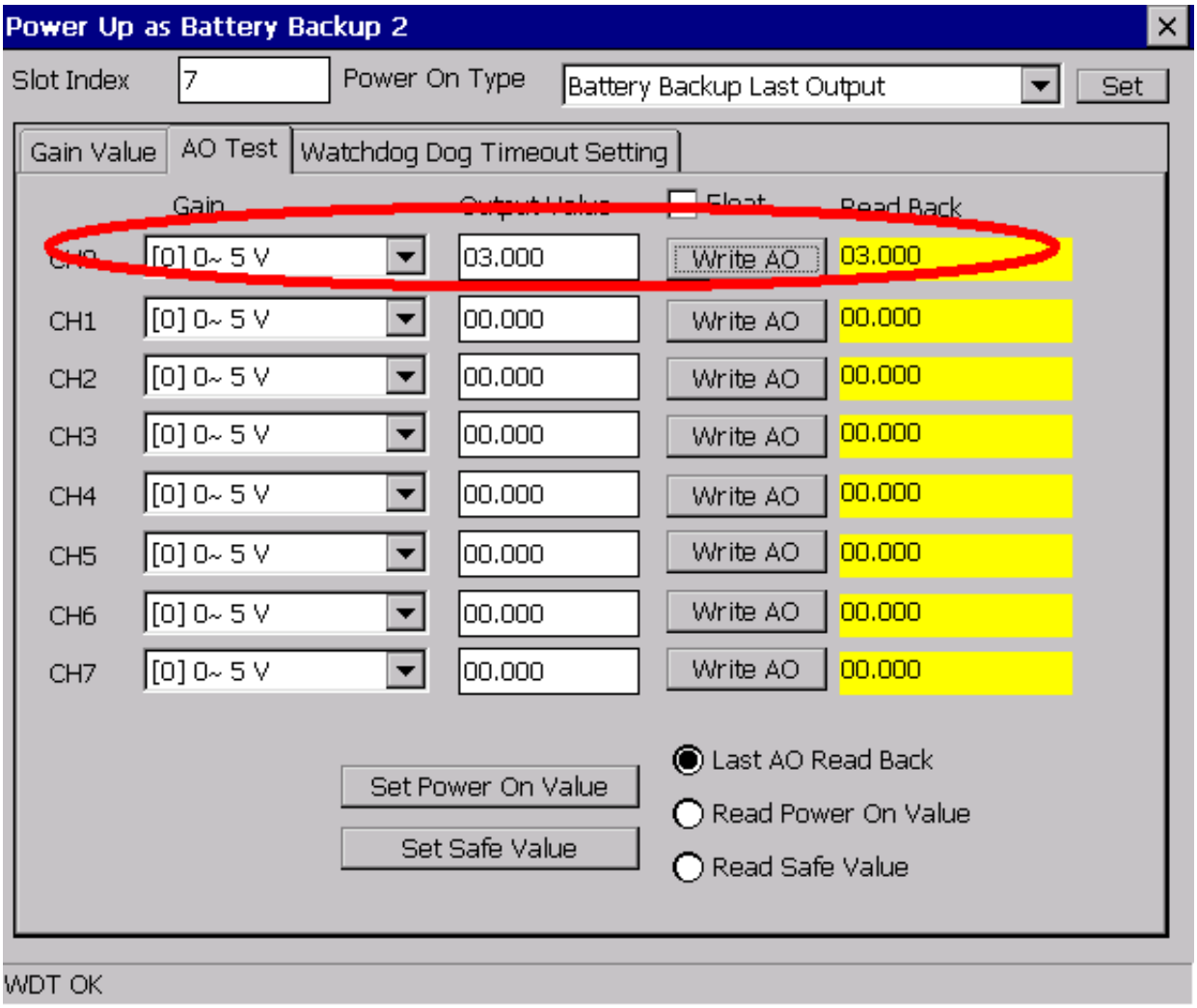
You can refresh the WDT timer using the pac_i8028U_RefreshModuleWDT (7) function.



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

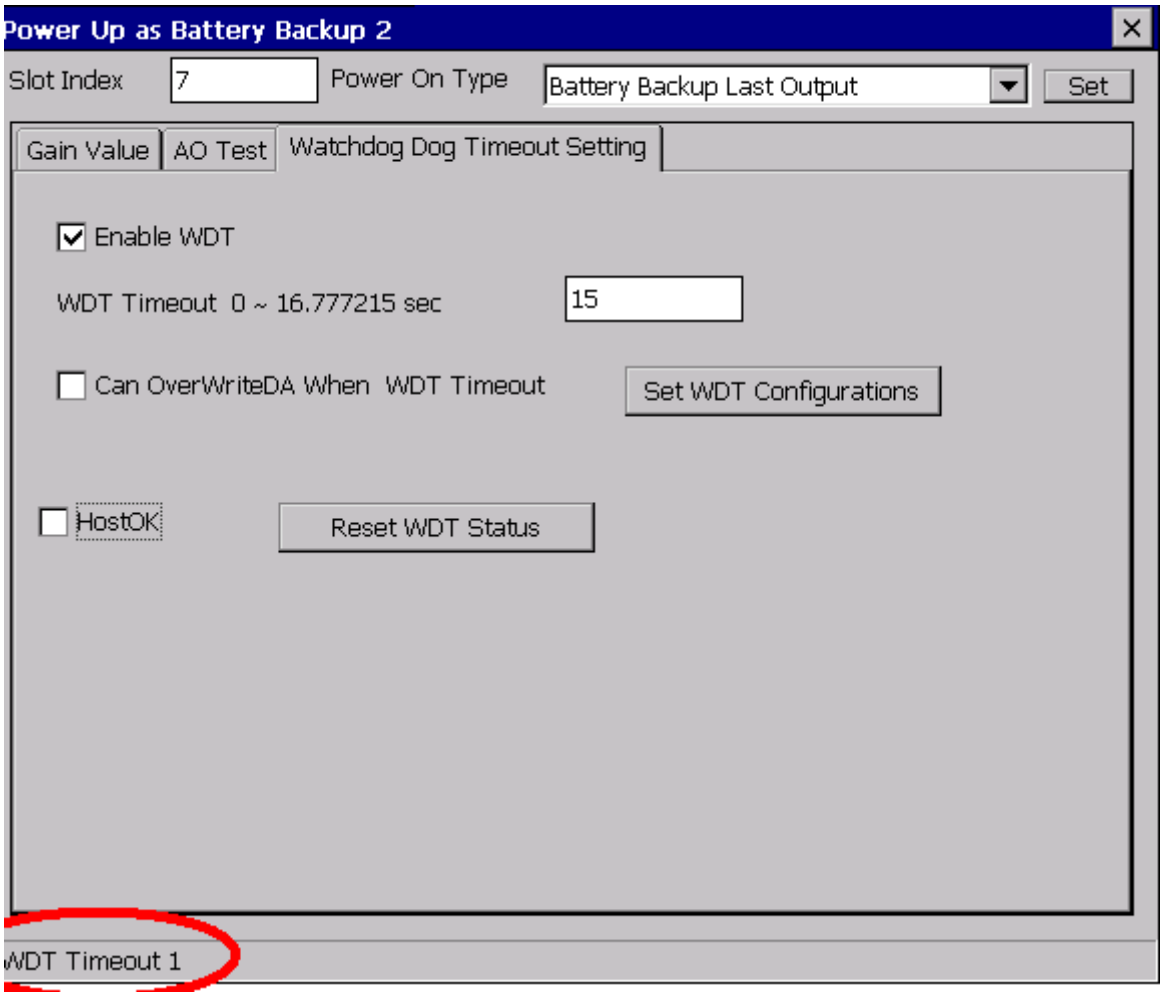
Before a WDT timeout occurs, we can use the pac_i8028U_WriteAO (0,0,0,5.0) function to write to channel 0 on slot 0 and set it to Gain 0 (0~5 V) and set the AO value for the I-9028U module to 5.0 V



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 that the communication line is secure at the terminal board.

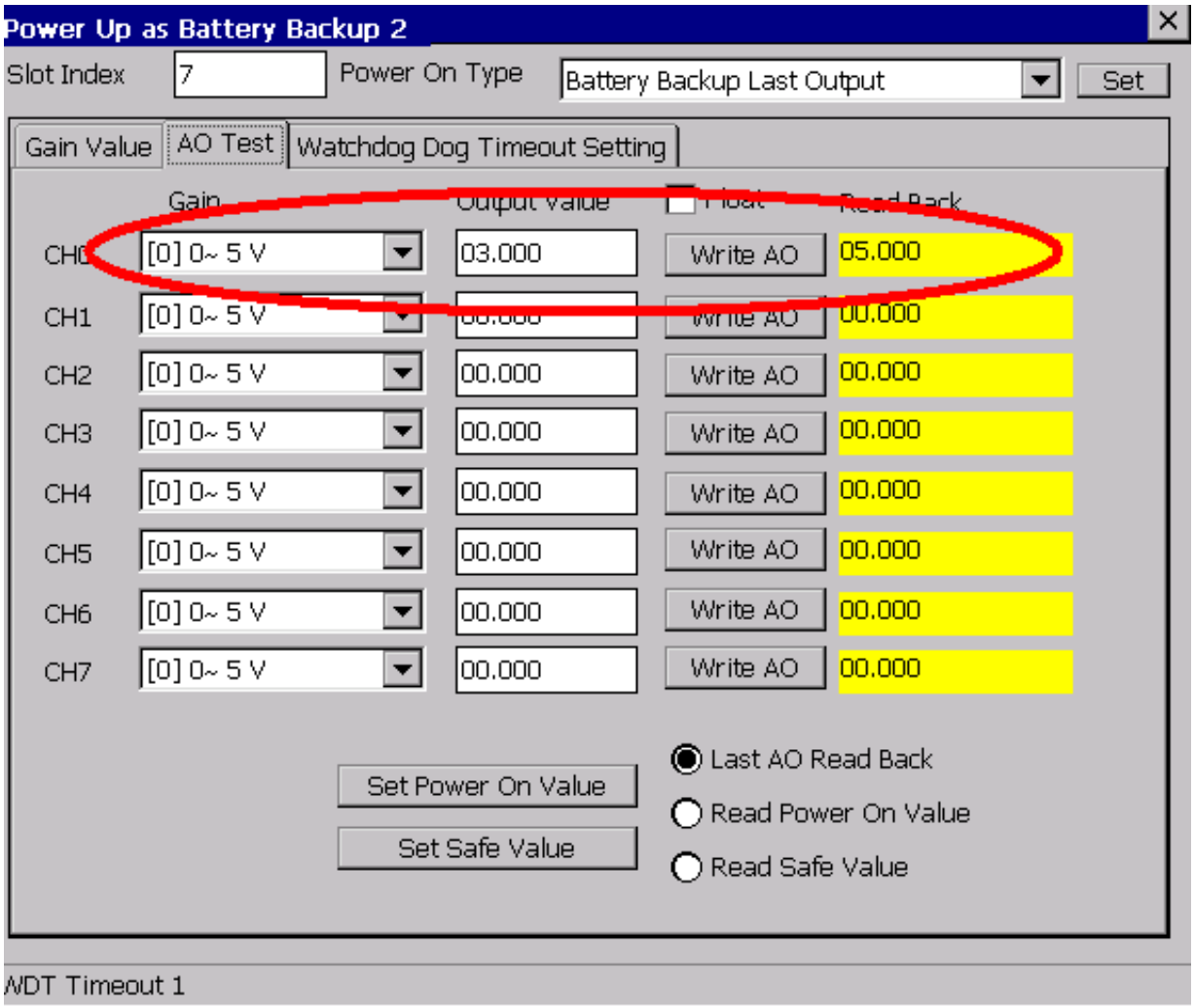
Ensure that the communication problems have been resolved continuing to the next step. If a communication problem still exists within the system, communication will remain unstable and the Safe Value will be triggered often.

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.

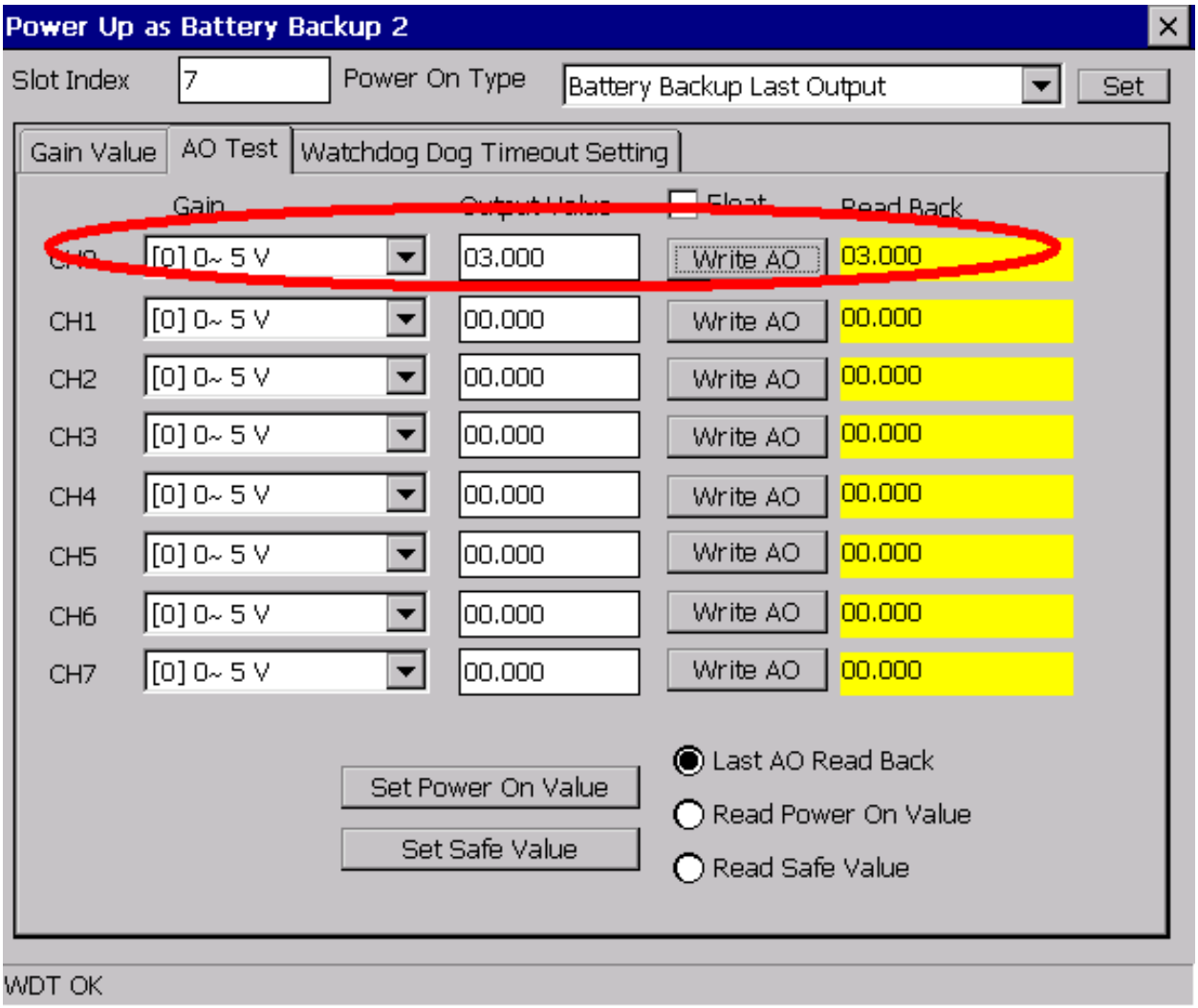
Once the WDT timeout has been cleared, the output module can be controlled again.

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. 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 1.7 Location of the Demo and Library Programs.

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.

5.1. pac_i8028U_Init

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

Syntax

C++

```
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-9028U.

-1 = there is no I-9028U module in this slot.

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

Note

Before executing any functions on the I-9028U module, the pac_i8028U_Init function needs to be called once for each I-9028U module connected to the network. If there are two or more I-9028U modules, you need call the pac_i8028U_Init function for each I-9028U module individually by specifying the slot number where the I-9028U module is inserted.

Example

[C]

```
intslot, err;
err = pac_i8028U_Init(slot);
```

[C#]

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

5.2. pac_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

C++

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

Parameters

slot:

specifies the slot number (0 - 7).

Return Value

ver:

The version number of the FPGA firmware for the I-9028U module.

Example

[C/C++]

```
shortver = 0, slot = 0;

pac_i8028U_GetFirmwareVersion (slot,&ver);
```

[C#]

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

5.3. pac_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

C++

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

Parameters

None

Return Value

The version number of the library file currently being used on the I-9028 module.

Example

[C++]

```
shortver;
ver= pac_i8028U_GetLibVersion();
```

[C#]

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

5.4. pac_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

C++

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

Parameters

**libDate:*

the release date of the pac_i8028U.lib

Return Value

None

Example

[C++]

```
charlibDate [32];
pac_i8028U_GetLibDate(libDate);
```

[C#]

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

5.5. pac_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

C++

```
pac_i8028U_ReadAO_GainOffset(
    int slot,
    intch,
    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).

**gVal:*

the gain value for the input range.

**oVal:*

the offset value for the input range.

Return Value

None

Example

[C++]

```
unsigned short gVal = 0;
shortoVal = 0;
intch,gain;
pac_i8028U_ReadAO_GainOffset(slot, ch, gain, &gVal, &oVal);
```

[C#]

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

5.6. pac_i8028U_WriteAOHex

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

Syntax

C++

```
short pac_i8028U_WriteAOHex(
    int slot,
    intch,
    short gain,
    shortaoHex
);
```

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

0 = No Error

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

Example

[C++]

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

[C#]

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

5.7. pac_i8028U_WriteAO

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

Syntax

C++

```
short pac_i8028U_WriteAO(
    int slot,
    intch,
    short gain,
    floataoData
);
```

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++]

```
int slot, ch, val;
short gain;
pac_i8028U_WriteAO(slot, ch, gain,val);
```

[C#]

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

5.8. pac_i8028U_ReadAOHex

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

Syntax

C++

```
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++]

```
int slot, ch ,gain;
shorthVal;
pac_i8028U_ReadAOHex(slot, ch, &gain, &hVal);
```

[C#]

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

5.9. pac_i8028U_ReadAO

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

Syntax

C++

```
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++]

```
int slot, ch, gain;
floatfVal = 0.0;
pac_i8028U_ReadAO(slot, ch, &gain, &fVal);
```

[C#]

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

5.10. pac_i8028U_SetPowerOnEnStatus

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

Syntax

C++

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

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++]

```
int slot;
int type;
pac_i8028U_SetPowerOnEnStatus(slot, type);
```

[C#]

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

5.11. pac_i8028U_GetPowerOnEnStatus

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

Syntax

C++

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

Parameters

slot:

specifies the slot number (0 - 7).

Return Value

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

1: Power-on Value Mode

2: Retentive Mode

Example

[C++]

```
int slot;
shortstatus;
pac_i8028U_GetPowerOnEnStatus(slot,&status)
```

[C#]

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

5.12. pac_i8028U_WritePowerOnHex_AO

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

Syntax

C++

```
short pac_i8028U_WritePowerOnHex_AO(
    int slot,
    intch,
    short gain,
    shortaoHex
);
```

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++]

```
int slot, ch;
shortgain,hVal;
pac_i8028U_WritePowerOnHex_AO(slot, ch, gain, hVal);
```

[C#]

```
Intslot, ch;
short gain, hVal;

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

5.13. pac_i8028U_WritePowerOn_AO

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

Syntax

C++

```
short pac_i8028U_WritePowerOn_AO(
    int slot,
    intch ,
    short gain,
    floataoData
);
```

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++]

```
int slot, ch;
short gain ;
floatfVal;
pac_i8028U_WritePowerOn_AO(slot, ch , gain, fVal);
```

[C#]

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

5.14. pac_i8028U_ReadPowerOnHex_AO

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

Syntax

C++

```
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++]

```
int slot, ch;
short gain ,hVal;
pac_i8028U_ReadPowerOnHex_AO(slot, ch, &gain, &hVal);
```

[C#]

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

5.15. pac_i8028U_ReadPowerOn_AO

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

Syntax

C++

```
void pac_i8028U_ReadPowerOn_AO(
    int slot,
    intch,
    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++]

```
int slot, ch;
shortgain;
floatfVal;
pac_i8028U_ReadPowerOn_AO(slot, ch, &gain, &fVal);
```

[C#]

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

5.16. pac_i8028U_WriteSafeHex_AO

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

Syntax

C++

```
short pac_i8028U_WriteSafeHex_AO(int slot, intch ,short gain, short aoHex);
    int slot,
    intch,
    short gain,
    shortaoHex
);
```

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++]

```
int slot, ch;
shortgain ,hVal;
pac_i8028U_WriteSafeHex_AO(slot, ch, gain, hVal);
```

[C#]

```
Intslot ,ch;
short gain ,hVal;

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

5.17. pac_i8028U_WriteSafe_AO

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

Syntax

C++

```
short pac_i8028U_WriteSafe_AO(
    int slot,
    intch,
    short gain,
    floataoData
);
```

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++]

```
int slot, ch;
shortgain;
floatfVal;
pac_i8028U_WriteSafe_AO(slot, ch, gain, fVal);
```

[C#]

```
Intslot ,ch;
int gain;
floatfVal;
pac_i8028UNet.pac8028U.WriteSafe_AO(slot, ch, gain, fVal);
```

5.18. pac_i8028U_ReadSafeHex_AO

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

Syntax

C++

```
void pac_i8028U_ReadSafeHex_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: +/-5 V

3: +/-10 V

4: +/-20 mA

5: 0 ~ 20 mA

**aoHex:*

[Output] the data in hexadecimal format.

Return Value

None

Example

[C++]

```
int slot, ch;
short gain ,hVal;
pac_i8028U_ReadSafeHex_AO(slot, ch, &gain, &hVal);
```

[C#]

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

5.19. pac_i8028U_ReadSafe_AO

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

Syntax

C++

```
void pac_i8028U_ReadSafe_AO(
    int slot,
    intch,
    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++]

```
int slot, ch;
short gain;
floatfVal;
pac_i8028U_ReadSafe_AO(slot, ch, &gain, &fVal);
```

[C#]

```
Intslot ,ch;
int gain;
floatfVal;
pac_i8028UNet.pac8028U.ReadSafe_AO(slot,ch, ref gain, ref fVal);
```

5.20. pac_i8028U_SetModuleWDTConfig

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

Syntax

C++

```
pac_i8028U_SetModuleWDTConfig(
    int slot,
    shortenStatus,
    unsigned long wdtTimeout,
    intifWDT_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++]

```
int slot;
shortenStatus;
unsigned long wdtTimeout;
intifWDT_Overwrite;
pac_i8028U_SetModuleWDTConfig(slot, enStatus, wdtTimeout, ifWDT_Overwrite);
```

[C#]

```
Intslot;
shortenStatus;
UInt32 timeout;
intifWDT;
pac_i8028UNet.pac8028U.SetModuleWDTConfig(slot, enStatus, timeout, ifWDT);
```

5.21. pac_i8028U_GetModuleWDTConfig

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

Syntax

C++

```
shortpac_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++]

```
int slot;
shortenStatus;
unsigned long wdtTimeout
intifWDT_Overwrite;
pac_i8028U_GetModuleWDTConfig(slot, &enStatus, &wdtTimeout, &ifWDT_Overwrite);
```

[C#]

```
Intslot;
shortenStatus;
UInt32 timeout;
intifWDT;
pac_i8028UNet.pac8028U.GetModuleWDTConfig(slot, ref enStatus, ref timeout, ref ifWDT);
```

5.22. pac_i8028U_GetModuleWDTStatus

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

Syntax

C++

```
shortpac_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++]

```
int slot ,ret;
ret = pac_i8028U_GetModuleWDTStatus(slot);
```

[C#]

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

5.23. pac_i8028U_ResetModuleWDT

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

Syntax

C++

```
voidpac_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++]

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

[C#]

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

5.24. pac_i8028U_RefreshModuleWDT

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

Syntax

C++

```
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++]

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

[C#]

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

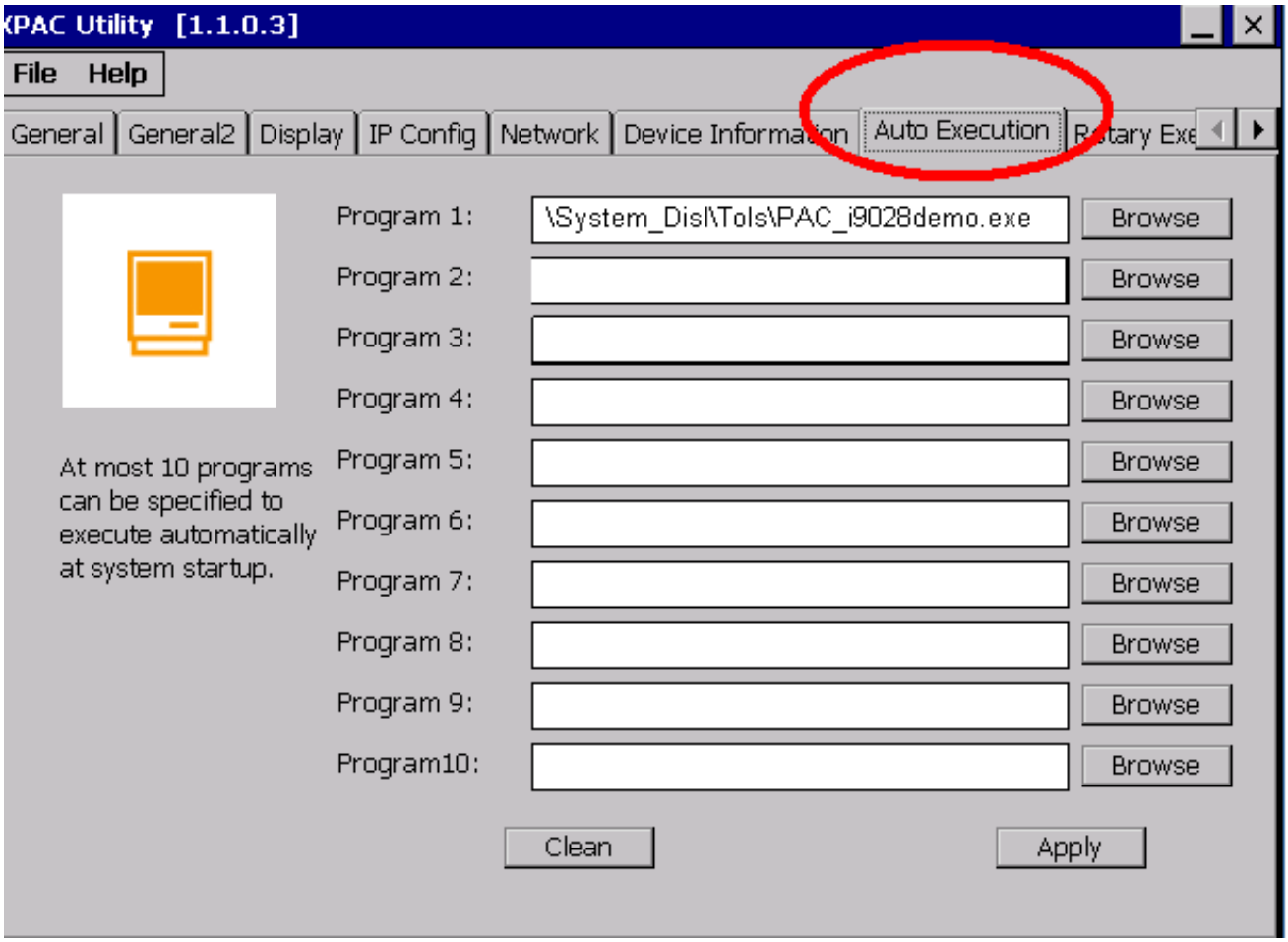
A.Error Codes

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.

B. Using Auto Execution to re-execute a failed program

The Watchdog (WDT) is a software function that monitors the operating status of the host, and, when enabled, is able to prevent problems from occurring.

The XPAC Utility is a tool provided by ICPDAS that can be used to configure various parameters associate with a PAC device. ICPDAS recommends that the Auto Execution function found in the XPAC utility be used to automatically re-execute a custom program if the PAC is reset for any reason.



By using a combination of the Auto execution function in the XPAC Utility and the Power-on Mode function on the I-9028U module, the stability of an application's performance can be improved.

More detailed information related to the XPAC Utility can be found at:
http://ftp.icpdas.com/pub/cd/xp-8000-ce6/document/user_manual/

C. 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
1.0.2	April 2018	• Modify library , demo path • Added WP-9000 , ippc-wes7 library , demo path • Modify API