I-9012

I/O Module User Manual

V1.0.1 July 2021

Written by Edward Ku/Cindy Huang

Edited by Anna Huang

Warranty

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

Warning

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

Copyright

Trademarks

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

Contact Us

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

Table of Contents

Preface

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

- Chapter 1, “Introduction” – This chapter provides information related to the hardware, such as the specifications, jumper settings and wiring.
- Chapter 2, “Quick Start” – This chapter provides information on how to get started, an overview of the location of the demo programs.
- Chapter 3, “API introduction” – This chapter describes the functions provided in the I-9012 library together with an explanation of the differences in the naming rules used for the Windows platforms.
- Chapter 4, “Calibration” – This chapter describes the calibration process for I-9012 module on Windows platforms.

1. Introduction

I-9012 is a high performance Analog Input module with 16 bits resolution and sampling rate up to 200 kS/s per channel. The I-9012 provide 8 channels and input range could be programmable to ± 5 V or ± 10 V. Trigger Output and Trigger Input Pin cloud make multitude of I-9012 modules Simultaneously Sampled at same time. And the module also provides 4 kV ESD protection and 2500 Vrms intra-module isolation.

Features

- 16-bit AD Converter
- Simultaneously Sampled
- 200 kHz sample rate at single channel
- 2500 Vrms intra-module isolation
- 8 k byte FIFO
- 4 kV ESD Protection
- 2500 Vrms Intra-module Isolation
- Wide Operating Temperature Range: -25 to +75 °C

Applicable Platform Table

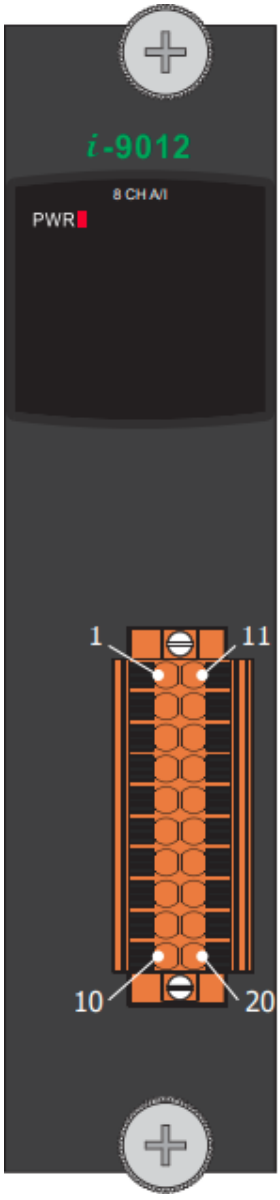
Platform	OS	Module
XPAC	XP-9000-WES7(WES7)	I-9012
WinPAC	WP-9000-CE7 (CE 7.0)	I-9012
LinPAC	LP-9000/LX-9000(Linux kernel 3.2/4.4)	I-9012

1.1. Specifications

Analog Input		
Channels		8-ch single-ended
Input Range		±5 V, ±10 V
Resolution		16-bit
Sample Rate		Single Channel Pacer Internal Mode: 200 kS/s Single Channel Pacer External Mode: 200 kS/s Single Channel Polling Mode: 40 kS/s
FIFO		8 k byte
AD Trigger Mode	Polling	Software trig
	Pacer	Internal Clock
		External Clock
Trig Output		
Frequency Range		16 Hz ~ 200 KHz
Compatibility		5V / TTL
Output Type		Pulse
Pulse Width		100 ns / 10 MHz
Trig Input		
Frequency Range		1 Hz ~ 200KHz
Trig Edge		Falling Edge
High Logic		> 5V
Low Logic		< 0.8V
LED Display		
System LED Indictors		1 LED as Power Indicator
Isolation		
Intra-module Isolation, Field to logic		2500 Vrms

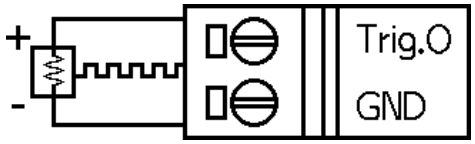
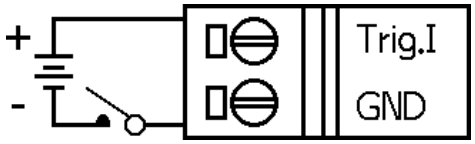
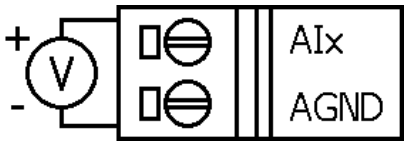
EMS Protection	
ESD(IEC 61000-4-2)	±4 kV Contact for Each Terminal
	±8 kV Air for Random Point
Power	
Power Consumption	1 W Max
Mechanical	
Dimensions (L × W × H)	144 mm × 31 mm × 134 mm
Environment	
Operating Temperature Storage	-25 ~ +75 °C
Temperature	-40 ~ +85 °C
Humidity	10 ~ 90 % RH , Non-condensing

1.2. Pin Assignments

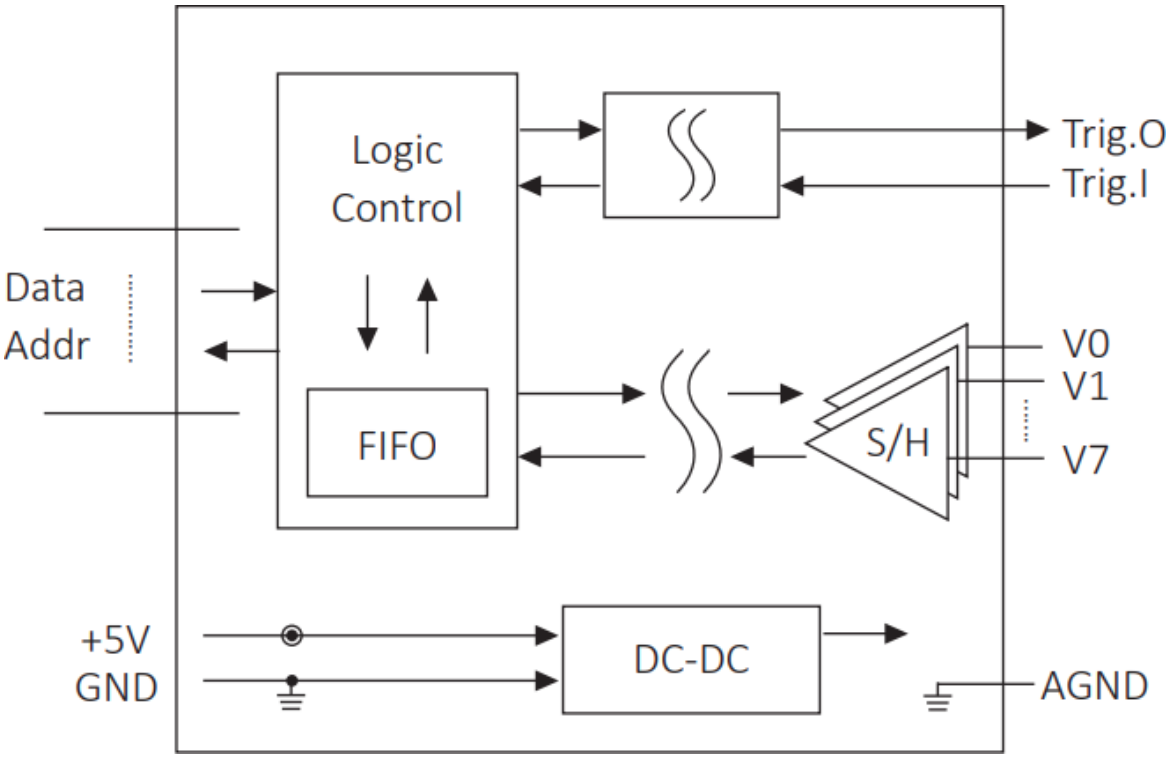


Pin Assignment	Terminal No.		Pin Assignment
Trig. O	01	11	GND
Trig. I	02	12	GND
AI0	03	13	AGND
AI1	04	14	AGND
AI2	05	15	AGND
AI3	06	16	AGND
AI4	07	17	AGND
AI5	08	18	AGND
AI6	09	19	AGND
AI7	10	20	AGND

1.3. Wire Connections

Mode	
Trigger Output	
Trigger Input	
Analog Input	

1.4. Block Diagram



2. Quick Start

This section shows the functions of I-9012 with demos on Windows platforms.

To download the demos, please visit ICP DAS website:

https://www.icpdas.com/en/download/show.php?num=2905

The following table lists the name of demos for each function:

Language / Function	C++	C#
Read Analog Input	pac_i8012W_Basic_Info	pac_i8012W_Basic_Info
Magic Scan Block Mode	pac_i8012W_MagicScan_Block	pac_i8012W_MagicScan_Block
Magic Scan NonBlock Mode	pac_i8012W_MagicScan_NonBlock	pac_i8012W_MagicScan_NonBlock
Magic Scan ISR Mode	pac_i8012W_MagicScan_ISR	NA
Multi Modules Scan	pac_i8012W_Multi_Modules_Scan	pac_i8012W_Multi_Modules_Scan

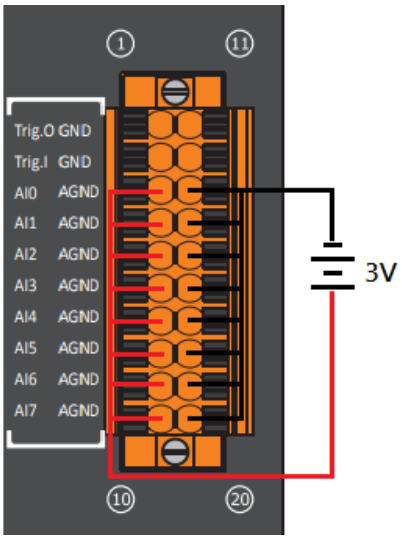
2.1. Windows-based Controllers

2.1.1. Read Analog Input

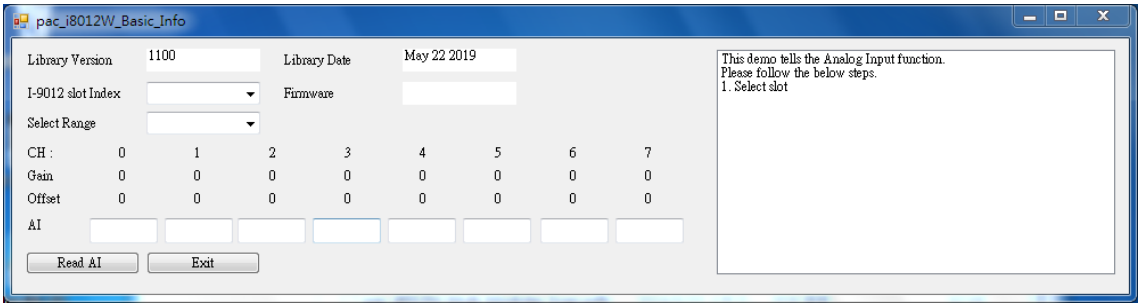
This function is used to read analog signal.

If there is an unused channel, please connect it to AGND to avoid getting floating data.

In this case, all the channels are connected to a 3V-Signal, as shown below:

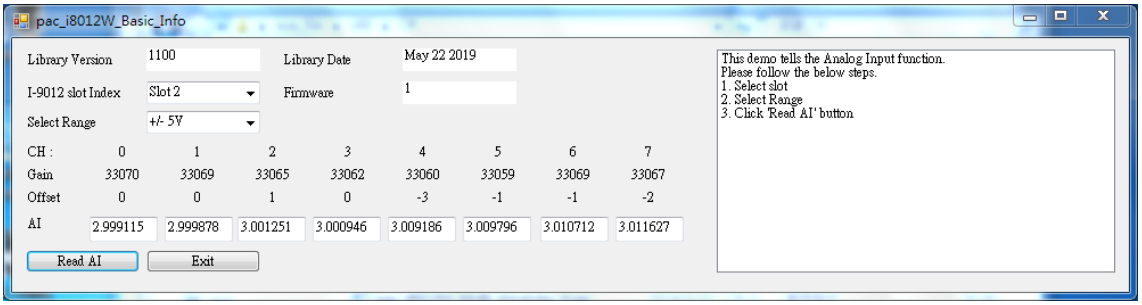


After wiring is done, execute the pac_i8012W_Basic_Info.exe demo.



Select the slot index where I-9012 is and select the range, then click “Read AI” button.

*The slot index starts from 1 at XP-9000 series host and starts from 0 at WP-9000 series host.



*The APIs provide float format and HEX format of data for read.

2.1.2. Magic Scan

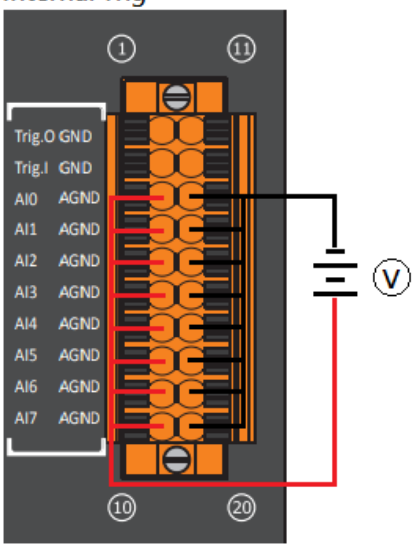
This function is used to sample data at high speed and can be triggered by an internal clock or an external clock, the external clock can be Trig.O pin or other device such as a waveform generator,

And read data in block mode, non-block mode or ISR mode.

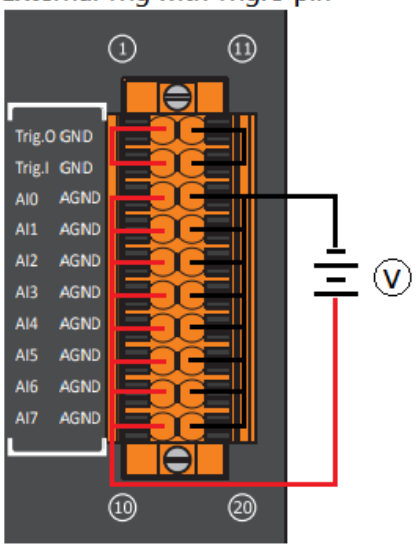
Due to different application, the ways to wiring are also different,

Following pictures are different wiring methods.

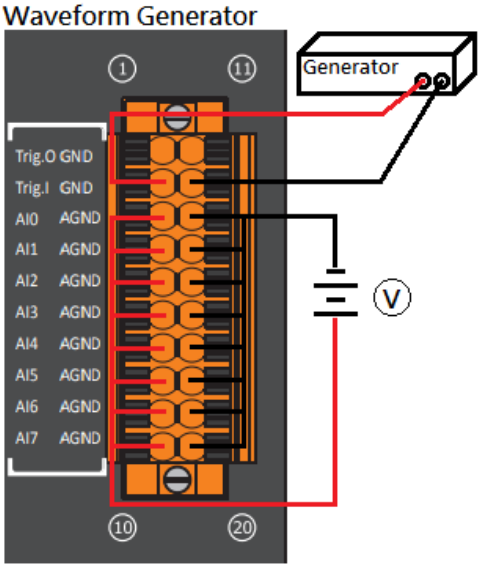
Internal Trig



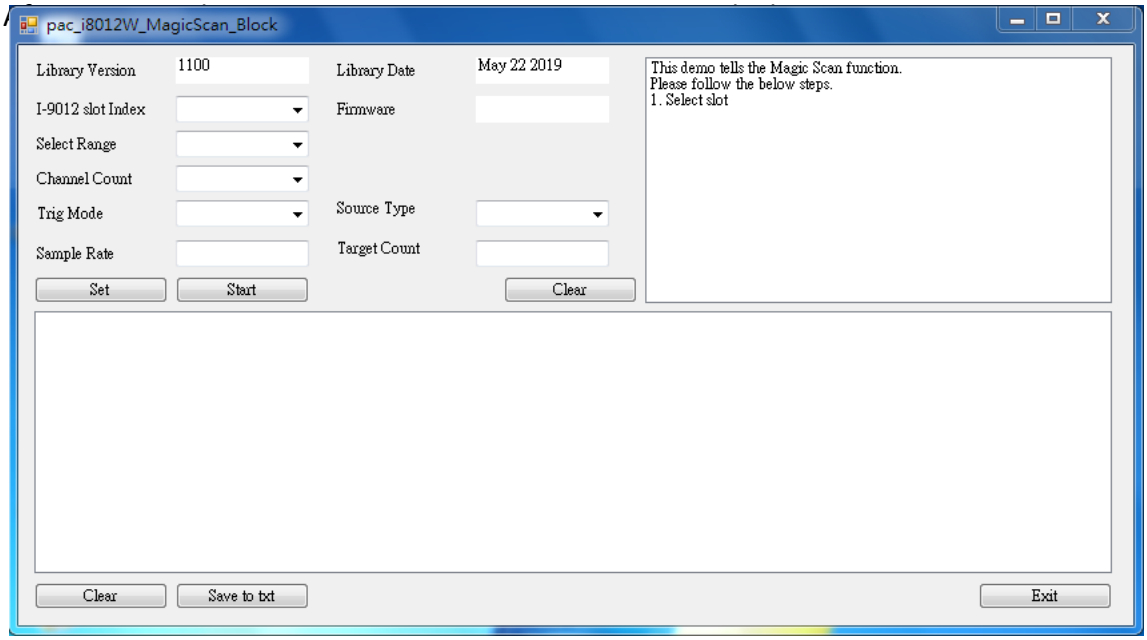
External Trig with Trig.O pin



External Trig with
Waveform Generator



In this case, all the channels are connected to a 3V-Signal and triggered with internal clock, then read data in block mode.



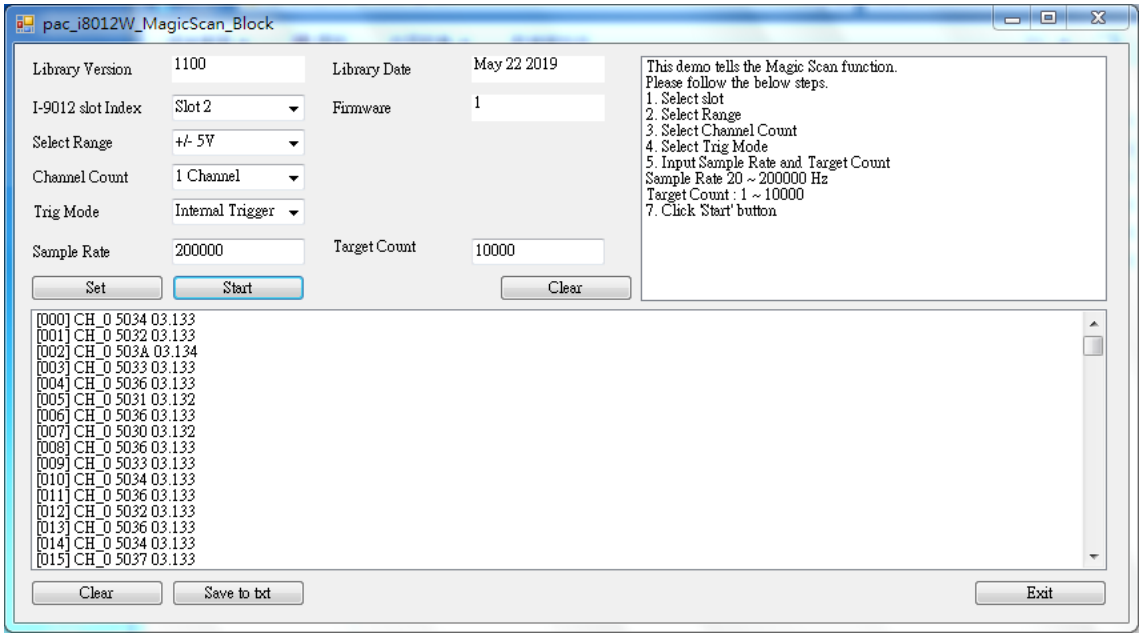
Select slot index, range, channel count, trig mode and source type.

Input sample rate and target count.

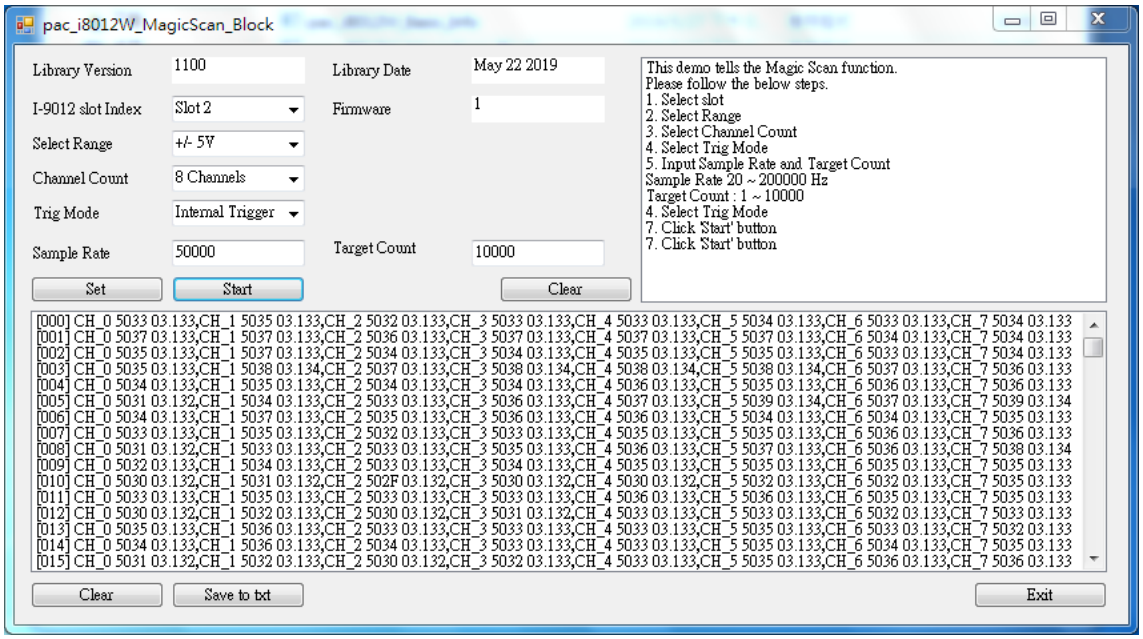
* The sample rate is affected by the channel count and the program process.

Then click “Set” button and “Start” button.

The picture below is the result of Magic Scan at 200K Hz in 1 channel.



The picture below is the result of Magic Scan at 50K Hz in 8 channels.

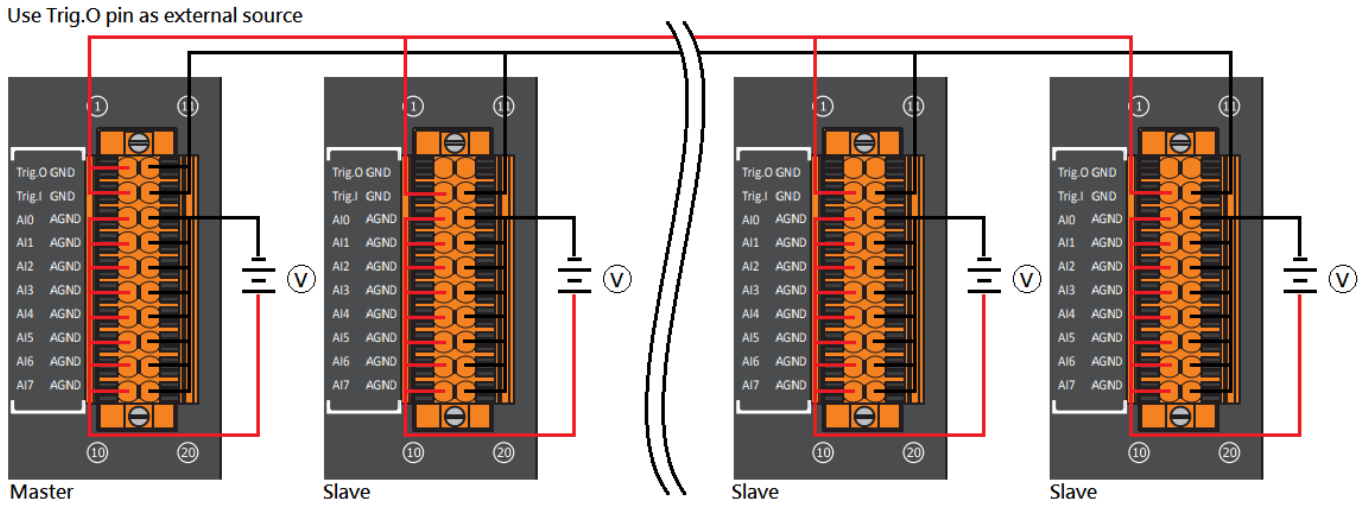


2.1.3. Multi Modules Scan

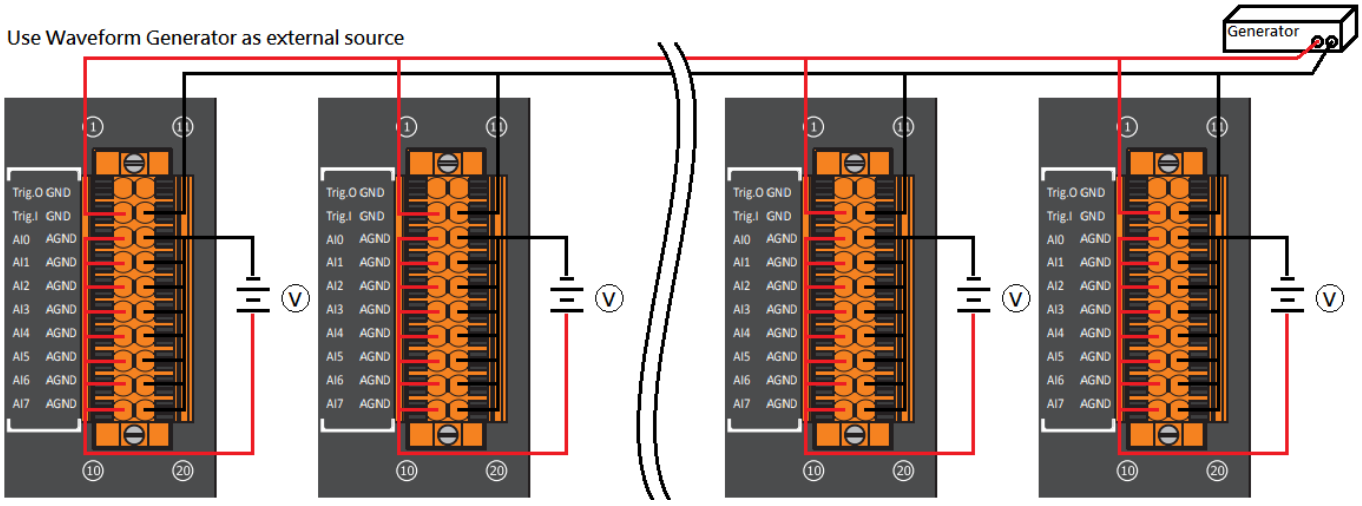
This function is used to make multiple I-9012 to sample data at same time and can only be triggered by an external clock, the external clock can be Trig.O pin or other device such as a waveform generator, and reading data in non-block mode.

Due to different application, the ways to wiring are also different,

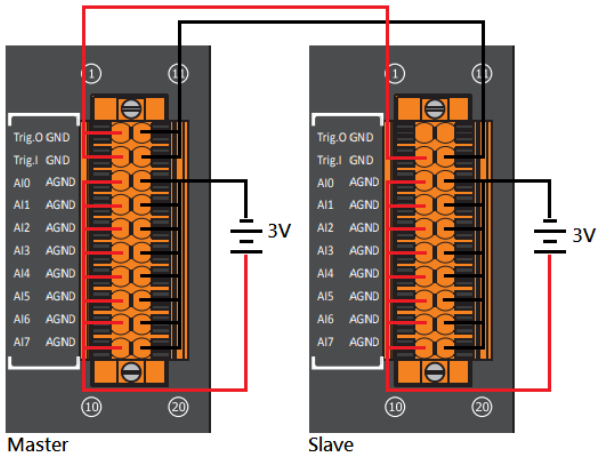
Following pictures are different wiring methods.



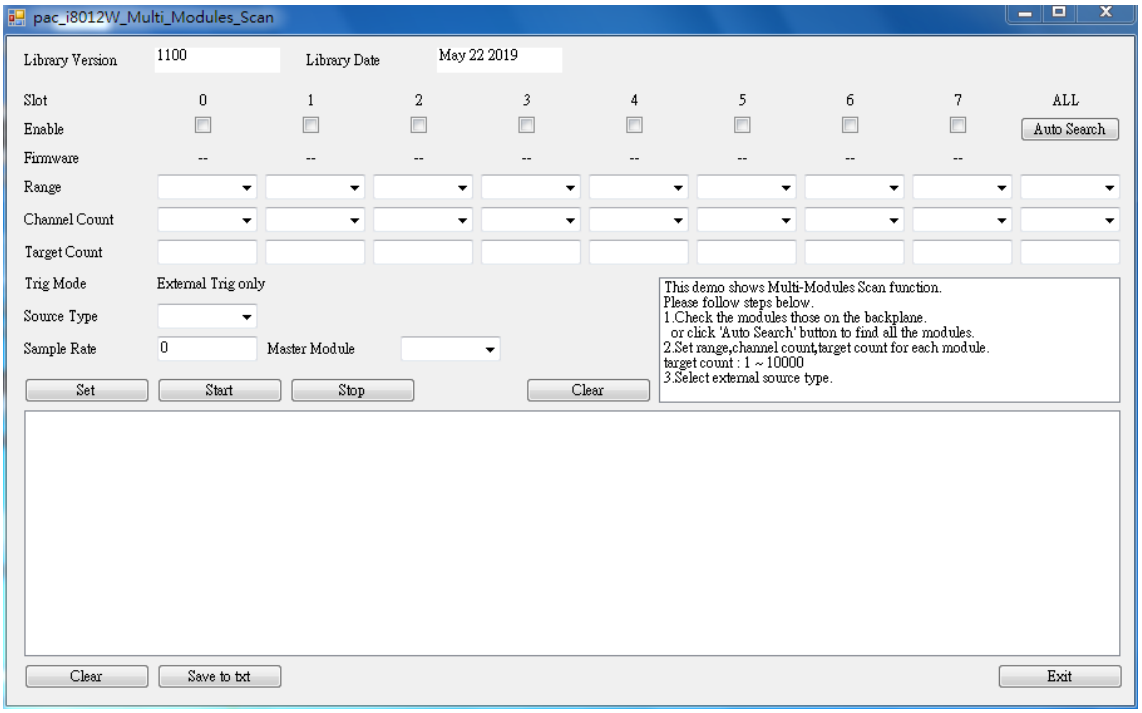
*If using Trig.O pin as an external source, it need set one of I-9012 as master card.



There are two modules of I-9012 in this case, all the channels of both I-9012 are connected to a 3V-Signal and triggered with Trig.O pin, then read data in non-block mode, as shown below:

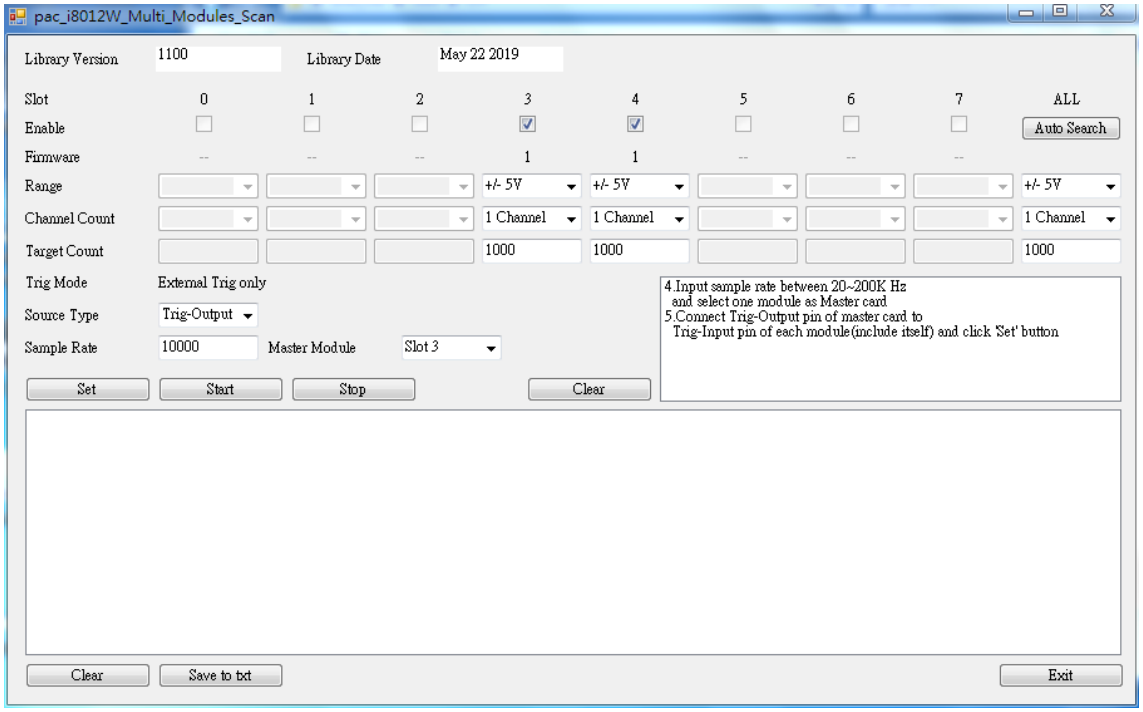


After wiring is done, execute pac_i8012W_Multi_Modules_Scan.exe.



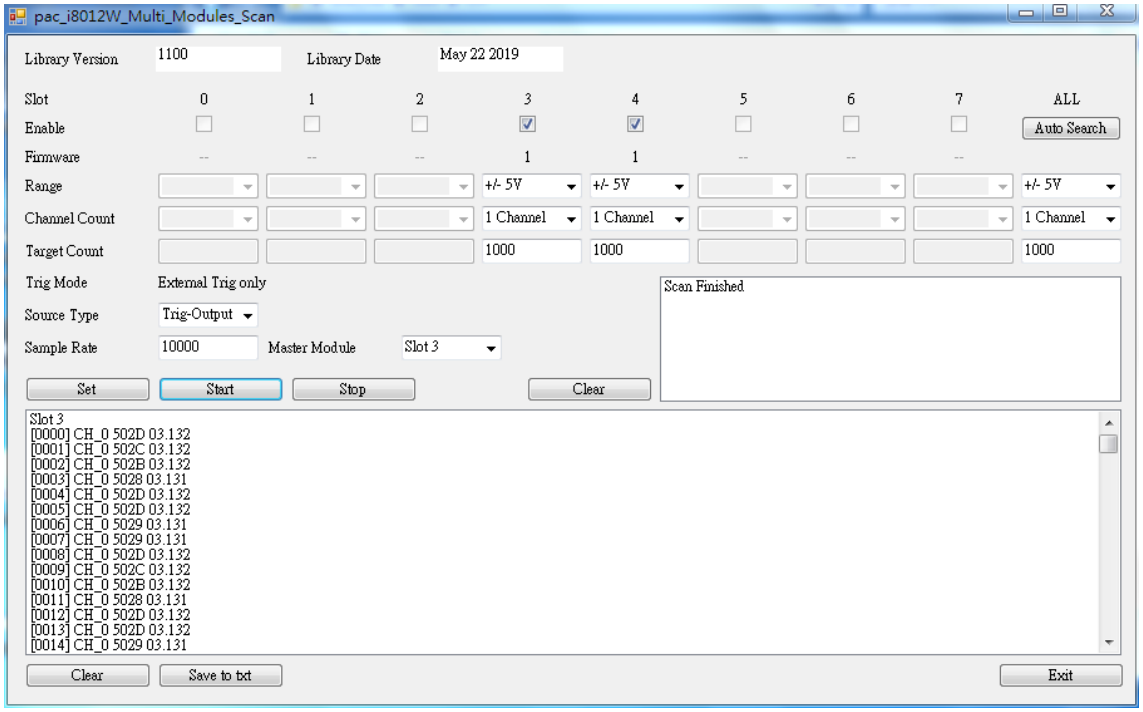
Check the slot index where I-9012 plugged into, select the range and channel count, Input target count, select source type, input sample rate and choose the master card.

* The sample rate is affected by the channel count and the program process.



Then click “Set” button and “Start” button.

The picture below is the result of Multi Modules Scan with two modules at 10K Hz in both 1 channel.

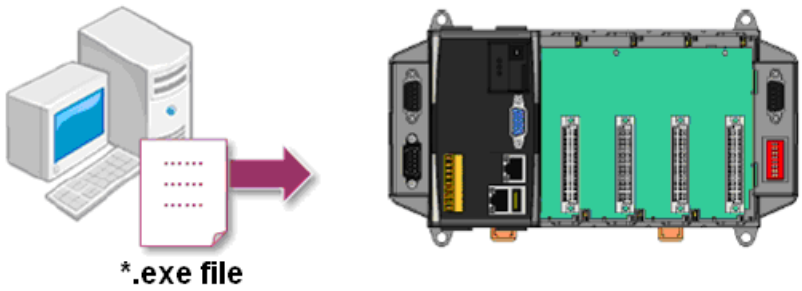


2.2. Linux-based Controllers

The following provides the name of the demos for each function:

Function \ Language	C
Read Analog Input	i9012_Basic_Info
Magic Scan Block	i9012_MagicScan_Block
Magic Scan NonBlock	i9012_MagicScan_NonBlock
Magic Scan ISR	i9012_MagicScan_ISR
Multiple Module Scan	i9012_Multi_Modules_Scan

ICP DAS provides a range of demo programs for different platforms that can be used to verify the functions of the I-9012. This section provides details of the functions for the I-9012. Demos are provided for C language, the source code contained in these programs can also be reused in your own custom programs if needed.



We need to check the following steps before running the program.

1. First, user need to download LinPAC SDK, which is includes GNU toolchain, Libraries, header, examples files, etc.

2. Check the power cable, Ethernet cable, VGA monitor, the communication cable between controller and PC has been connected well, and then check the I-9012 has been plugged in the controller.

3. Next, check the communication between controller and PC is fine, and download the demo program files to the controller.

4. The following is a list of the locations where both the demo programs and associated libraries (libi8k.a) can be found on either the ICP DAS web site, and user can find the related files in the below website:

PRRODUCT	CPU	DOWNLOAD LINK
LP-9000	Am335x	https://www.icpdas.com/en/download/show.php?num=915
LX-9000	X86/E38xx	https://www.icpdas.com/en/download/show.php?num=904

3. API References

ICPDAS supplies a range of C/C++ API functions for I-9012 module.

When developing a custom program, refer to either the pac_i8012W.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.

API Naming Table

The following table shows the API names on different platforms and the beginning of API.

Platform	Product included	API prefix characters	
		C / C++	C#
CE7.0	I-9012	“pac_i8012W_ “ + function name	“pac8012WNet.pac8012W” + function name
WES	I-9012		
Linux	I-9012	“i9012_” + function name	Null

The following is an overview of the functions provided in the pac_i8012W.lib for use with the Windows (CE and WES) platform.

API for I-9012

Function for Windows	Description
pac_i8012W_Init	This function is used to check the hardware ID of I-9012 and initial.
pac_i8012W_GetLibVersion	This function is used to get the library version of libraries.
pac_i8012W_GetLibDate	This function is used to get the library built date of libraries.
pac_i8012W_GetFirmwareVersion	This function is used to get the FPGA version of I-9012.
pac_i8012W_Read_AI	This function is used to read analog input on specific channel in float format.
pac_i8012W_Read_AIHex	This function is used to read analog input on specific channel in HEX format.
pac_i8012W_ConfigMagicScan	This function is used to configure the Magic Scan function of I-9012.
pac_i8012W_StartMagicScan	This function is used to start Magic Scan function.
pac_i8012W_StopMagicScan	This function is used to stop Magic Scan function.
pac_i8012W_Read_FIFO_Block	This function is used to read FIFO in block mode.
pac_i8012W_Read_FIFO_NonBlock	This function is used to read FIFO in non-block mode.
pac_i8012W_InstallMagicScanISR	This function is used to install ISR function for Magic Scan function.
pac_i8012W_UnInstallMagicScanISR	This function is used to uninstall ISR function for Magic Scan function.
pac_i8012W_ClearINT	This function is used to clear the signal of interrupt.
pac_i8012W_ReadFIFO_ISR	This function is used to read FIFO in ISR function.
pac_i8012W_ConfigTrigOut	This function is used to set the frequency of the Trig.O output.
pac_i8012W_Enable_TrigOut	This function is used to start output on Trig.O pin.
pac_i8012W_Disable_TrigOut	This function is used to stop output on Trig.O pin

Function for Windows	Description
pac_i8012W_ReadGainOffset	This function is used to read gain and offset for specific channel.
pac_i8012W_CalibrationHEX	This function is used to calibrate data in HEX format.
pac_i8012W_CalibrationFloat	This function is used to calibrate data in float format.

3.1. pac_i8012W_Init

This function is used to check the hardware ID of I-9012 and initial.

Syntax

For Windows (CE and WES)

```
short pac_i8012W_Init (int slot);
```

For Linux

```
short i9012_Init (int slot);
```

Parameter

slot:

Specific slot number (0 - 7), except range of slot is number 1 ~ 8 for LinPAC.

Return Values

0 = the module in the specific slot is I-9012.

-1 = there is no I-9012 on the slot.

Note

Before using any APIs for I-9012, this API must be called once, If there are two or more modules in the host, this API needs to be called for each I-9012 with the corresponding slot number.

Example

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    int slot=-1,i=0;
    for(i=0;i<8;i++){       // Search every slot
        if(pac_i8012W_Init(i)==0){     // Find first I-9012
            slot=i;
            break;
        }
    }
    return 0;
}
```

[C#]

```
static void Main(string[] args){
    int slot=-1,i = 0;
    for (i = 0; i < 8; i++){        // Search every slot
        if (pac8012WNet.pac8012W.Init(i) == 0){      // Find first I-9012
            slot=i;
            break;
        }
    }
}
```

[C] (LinPAC)

```
int main(){

    int slot=1, ret=0;

    ret=Open_Slot(slot);

    if (ret > 0) {
        printf("Open Slot%d failed, return value=%d \n", slot, ret);
        return (-1);
    }

    if(i9012_Init(slot)==0){        // Find the first I-9012

        break;

    }

    Close_Slot(slot);

    return 0;

}
```

3.2. pac_i8012W_GetLibVersion

This function is used to get the library version of pac_i8012W.lib / pac_i8012W.dll.

Syntax

For Windows (CE and WES)

```
short pac_i8012W_GetLibVersion(void);
```

For Linux

```
short i9012_GetLibVersion(void);
```

Parameter

slot:

Specific slot number (0 - 7), except range of slot is number 1 ~ 8 for LinPAC.

Return Values

The version number of the libraries.

EX : 0x1100 means version 1.1.0.0

Example

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    short ver=0;
    ver=pac_i8012W_GetLibVersion();
    return 0;
}
```

[C#]

```
static void Main(string[] args){
    short LibVer = 0;
    LibVer = pac8012WNet.pac8012W.LibVersion();
}
```

[C] (LinPAC)

```
int main(){

    int slot=1, ret=0, ver;

    ret=Open_Slot(slot);

    if (ret > 0) {
            printf("Open Slot%d failed, return value=%d \n", slot, ret);
            return (-1);
    }

    ver=i9012_GetLibVersion();

    Close_Slot(slot);

    return 0;

}
```

3.3. pac_i8012W_GetLibDate

This function is used to get the library built date of pac_i8012W.lib /pac_i8012W.dll.

Syntax

For Windows (CE and WES)

```
void pac_i8012W_GetLibDate(char libDate[]);
```

Parameter

libDate[]:

The built date of the libraries.

Return Values

None

Example

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    char libDate[20];
    pac_i8012W_GetLibDate(libDate);
    return 0;
}
```

[C#]

```
static void Main(string[] args){
        string LibDate = "";
        LibDate = pac8012WNet.pac8012W.LibDate();
}
```

3.4. pac_i8012W_GetFirmwareVersion

This function is used to get the FPGA version of I-9012.

Syntax

For Windows (CE and WES)

```
short pac_i8012W_GetFirmwareVersion(int slot);
```

For Linux

```
short i9012_GetFirmwareVersion(int slot);
```

Parameter

slot:

Specific slot number (0 - 7), except range of slot is number 1 ~ 8 for LinPAC.

Return Values

The version number of FPGA.

Example

[C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    int slot=1;
    short ver=0;
    pac_i8012W_Init(slot);       // initialize
    ver=pac_i8012W_GetFirmwareVersion(slot);
    return 0;
}
```

[C#]

```
static void Main(string[] args){
        int slot = 1;
        short FarmVer = 0;
        pac8012WNet.pac8012W.Init(slot);          // initialize
        FarmVer = pac8012WNet.pac8012W.FirmwareVersion(slot);
}
```

[C] (LinPAC)

```
int main(){
    int slot=1, ret=0, ver;
    ret=Open_Slot(slot);
    if (ret > 0) {
            printf("Open Slot%d failed, return value=%d \n", slot, ret);
            return (-1);
    }
    i9012_Init(slot);          // Find the first I-9012
    ver=i9012_GetFirmwareVersion(slot);
    Close_Slot(slot);
    return 0;
    }
```

3.5. pac_i8012W_Read_AI

This function is used to read analog input on specific channel in float format.

Syntax

For Windows (CE and WES)

```
int pac_i8012W_Read_AI(int slot,int ch,int gain,float* fval);
```

For Linux

```
int i9012_Read_AI(int slot,int ch,int gain,float* fval);
```

Parameter

slot:

Specific slot number (0 - 7), except range of slot is number 1 ~ 8 for LinPAC.

ch:

Specific channel (0 - 7).

gain:

Specific input range (0 - 1)

0: +/- 5V

1: +/- 10V

**fval:*

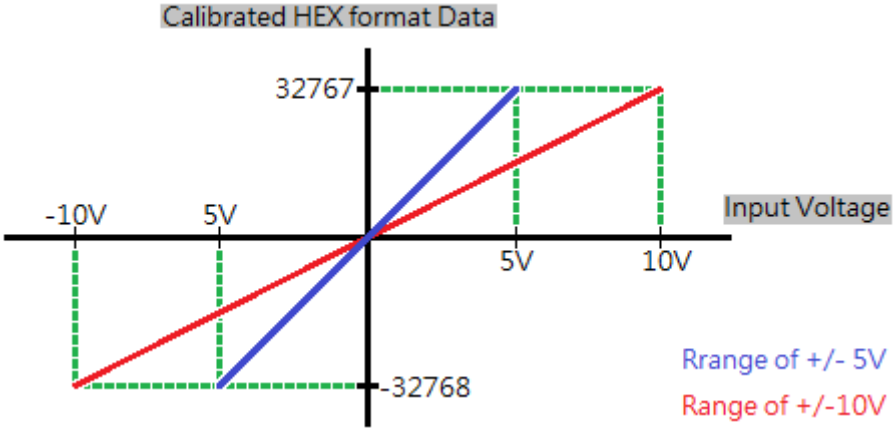
[Output] Calibrated float format data.

Return Values

Please refer to the Error Code.

Note

The following picture shows the scale of voltage and data.



The following formulas describe how to calculate the hexadecimal data into floating data.

```
If(hexadecimal data >=0){
        floating data = (hexadecimal data / 32767) * range(5 or 10)
}
Else{
        floating data = (hexadecimal data / 32768) * range(5 or 10)
}
```

Example

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    int slot=1,ch=0,gain=0;
    float fval=0;
    pac_i8012W_Init(slot);   // initialize
    pac_i8012W_Read_AI(slot,ch,gain,&fval);
    return 0;
}
```

[C#]

```
static void Main(string[] args){
        int slot = 1, ch = 0, gain = 0;
        float fval = 0;
        pac8012WNet.pac8012W.Init(slot);   // initialize
        pac8012WNet.pac8012W.ReadAI(slot, ch, gain, ref fval);
}
```

[C] (LinPAC)

```
int main(){

    int slot=1, ret=0, ch = 0, gain = 0;

    ret=Open_Slot(slot);

    if (ret > 0) {
            printf("Open Slot%d failed, return value=%d \n", slot, ret);
            return (-1);
    }

    i9012_Init(slot);                    // initialize

    i9012_Read_AI(slot,ch,gain,&fval);

    Close_Slot(slot);

    return 0;

}
```

3.6. pac_i8012W_Read_AIHex

This function is used to read analog input on specific channel in Hexadecimal format.

Syntax

For Windows (CE and WES)

```
int pac_i8012W_Read_AIHex(int slot,int ch,int gain,short* hval);
```

For Linux

```
int i9012_Read_AIHex(int slot,int ch,int gain,short* hval);
```

Parameter

slot:

Specific slot number (0 - 7), except range of slot is number 1 ~ 8 for LinPAC.

ch:

Specific channel (0 - 7).

gain:

Specific input range (0 - 1)

0: +/- 5V

1: +/- 10V

**hval:*

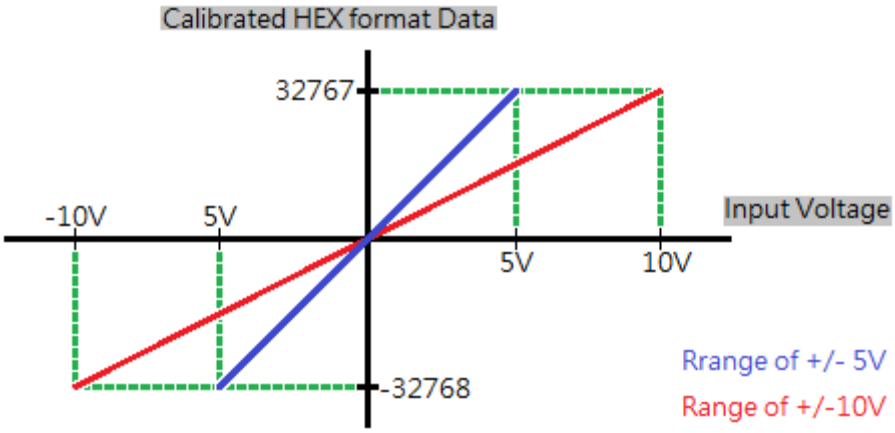
[Output] Calibrated HEX format data.

Return Values

Please refer to the Error Code.

Note

The following picture shows the scale of voltage and data.



Example

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    int slot=1,ch=0,gain=0;
    short hval=0;
    pac_i8012W_Init(slot);   // initialize
    pac_i8012W_Read_AIHex(slot,ch,gain,&hval);
    return 0;
}
```

[C#]

```
static void Main(string[] args){
        int slot = 1, ch = 0, gain = 0;
        short hval = 0;
        pac8012WNet.pac8012W.Init(slot);   // initialize
        pac8012WNet.pac8012W.ReadAIHex(slot, ch, gain, ref hval);
}
```

[C] (LinPAC)

```
int main(){

    int slot=1, ret=0, ch = 0, gain = 0;

    ret=Open_Slot(slot);

    if (ret > 0) {
            printf("Open Slot%d failed, return value=%d \n", slot, ret);
            return (-1);
    }

    i9012_Init(slot);                // initialize

    i9012_Read_AIHex(slot,ch,gain,&hval);

    Close_Slot(slot);

    return 0;

}
```

3.7. pac_i8012W_ConfigMagicScan

This function is used to configure the Magic Scan function of I-9012.

Syntax

For Windows (CE and WES)

```
int pac_i8012W_ConfigMagicScan(int slot, int gain, int chcnt, int type, unsigned long samplerate, unsigned long* realrate);
```

For Linux

```
int i9012_ConfigMagicScan(int slot, int gain, int chcnt, int type, unsigned long samplerate, unsigned long* realrate);
```

Parameter

slot:

Specific slot number (0 - 7), except range of slot is number 1 ~ 8 for LinPAC.

gain:

Specific input range (0 - 1)

0: +/- 5V

1: +/- 10V

chcnt:

Number of channels used for scan. (1 - 8)

EX: chcnt set as 1, Magic Scan will scan channel 0.

chcnt set as 2, Magic Scan will scan channel 0 and channel 1.

type:

Trig mode for Magic Scan (0 - 1)

0: Internal trig.

1: External trig.

samplerate:

The sampling rate for Magic Scan at internal trig (20 - 200000)

** realrate:*

[Output]The real sampling rate.

The sampling rate is made by dividing the basic clock with 16 bit resolution.

Return Values

Please refer to Error Code.

Example

Please refer to Magic Scan demos.

3.8. pac_i8012W_StartMagicScan

This function is used to start Magic Scan function.

Syntax

For Windows (CE and WES)

```
int pac_i8012W_StartMagicScan(int slot);
```

For Linux

```
int i9012_StartMagicScan(int slot);
```

Parameter

slot:

Specific slot number (0 - 7), except range of slot is number 1 ~ 8 for LinPAC.

Return Values

Please refer to Error Code.

Example

Please refer to Magic Scan demos.

3.9. pac_i8012W_StopMagicScan

This function is used to stop Magic Scan function.

Syntax

For Windows (CE and WES)

```
int pac_i8012W_StopMagicScan(int slot);
```

For Linux

```
int i9012_StopMagicScan(int slot);
```

Parameter

slot:

Specific slot number (0 - 7), except range of slot is number 1 ~ 8 for LinPAC.

Return Values

Please refer to Error Code.

Example

Please refer to Magic Scan demos.

3.10. pac_i8012W_Read_FIFO_Block

This function is used to read data from FIFO in block mode.

Syntax

For Windows (CE and WES)

```
int pac_i8012W_Read_FIFO_Block(int slot, short FIFOData[], unsigned long cnt, unsigned long* readFIFOcnt);
```

For Linux

```
int i9012_Read_FIFO_Block(int slot, short FIFOData[], unsigned long cnt, unsigned long* readFIFOcnt);
```

Parameter

slot:

Specific slot number (0 - 7), except range of slot is number 1 ~ 8 for LinPAC.

FIFOData[]:

Specific short format array for store the data that reading from FIFO.

cnt:

Amount of data required.

**readFIFOcnt:*

[Output] Total read counts of data in this process

Return Values

Please refer to Error Code.

Example

Please refer to Magic Scan Block mode demo.

3.11. pac_i8012W_Read_FIFO_NonBlock

This function is used to read data from FIFO in non-block mode.

Syntax

For Windows (CE and WES)

```
int pac_i8012W_Read_FIFO_NonBlock(int slot, short FIFOData[], unsigned long cnt, short* readFIFOcnt);
```

For Linux

```
int i9012_Read_FIFO_NonBlock(int slot ,short FIFOData[], unsigned long cnt, short* readFIFOcnt);
```

Parameter

slot:

Specific slot number (0 - 7), except range of slot is number 1 ~ 8 for LinPAC.

FIFOData[]:

Specific short format array for store the data that reading from FIFO.

cnt:

Amount of data required.

**readFIFOcnt:*

[Output] Total read counts of data in this process

Return Values

Please refer to Error Code.

Example

Please refer to Magic Scan Non-Block mode demo.

3.12. pac_i8012W_InstallMagicScanISR

This function is used to install ISR function for Magic Scan function.

Syntax

For Windows (WES)

```
short pac_i8012W_InstallMagicScanISR(int slot,void (__stdcall *isr)(int slot),short leveltype);
```

For Windows (CE)

```
short pac_i8012W_InstallMagicScanISR(int slot,void (*isr)(int),short leveltype);
```

For Linux

```
short i9012_InstallMagicScanISR(int slot,void (*isr)(int),short leveltype);
```

Parameter

slot:

Specific slot number (0 - 7), except range of slot is number 1 ~ 8 for LinPAC.

*void (__stdcall *isr)(int slot) / void (*isr)(int):*

The function pointer passed for the ISR.

leveltype:

0: 10

1: 2048

The interrupt trigger condition (0 - 1) based on the amount of data in the FIFO buffer.

When the amount of data in the FIFO buffer is greater than the value defined by the leveltype parameter, the interrupt will be triggered and the ISR will be executed to handle the interrupt event.

Return Values

Please refer to Error Code.

Example

Please refer to Magic Scan ISR mode demo.

3.13. pac_i8012W_UnInstallMagicScanISR

This function is used to uninstall ISR function for Magic Scan function.

Syntax

For Windows (CE and WES)

```
short pac_i8012W_UnInstallMagicScanISR(int slot);
```

For Linux

```
short i9012_UnInstallMagicScanISR(int slot);
```

Parameter

slot:

Specific slot number (0 - 7), except range of slot is number 1 ~ 8 for LinPAC.

Return Values

Please refer to Error Code.

Example

Please refer to Magic Scan ISR mode demo.

3.14. pac_i8012W_ClearINT

This function is used to clear the signal of interrupt.

Syntax

For Windows (CE and WES)

```
short pac_i8012W_ClearINT(int slot);
```

For Linux

```
short i9012_ClearINT(int slot);
```

Parameter

slot:

Specific slot number (0 - 7), except range of slot is number 1 ~ 8 for LinPAC.

Return Values

Please refer to Error Code.

Example

Please refer to Magic Scan ISR mode demo.

3.15. pac_i8012W_ReadFIFO_ISR

This function is used to read data from FIFO in ISR mode.

Syntax

For Windows (CE and WES)

```
short pac_i8012W_ReadFIFO_ISR(int slot, short FIFOData[],short* readFIFOcnt);
```

For Linux

```
short i9012_ReadFIFO_ISR(int slot, short FIFOData[],short* readFIFOcnt);
```

Parameter

slot:

Specific slot number (0 - 7), except range of slot is number 1 ~ 8 for LinPAC.

FIFOData[]:

Specific short format array for store the data that reading from FIFO.

**readFIFOcnt:*

[Output] Total read counts of data in this process

Return Values

Please refer to Error Code.

Example

Please refer to Magic Scan ISR mode demo.

3.16. pac_i8012W_ConfigTrigOut

This function is used to set the frequency of the Trig.O output.

Syntax

For Windows (CE and WES)

```
int pac_i8012W_ConfigTrigOut(int slot,unsigned long freq,unsigned long* realfreq);
```

For Linux

```
int i9012_ConfigTrigOut(int slot,unsigned long freq,unsigned long* realfreq);
```

Parameter

slot:

Specific slot number (0 - 7), except range of slot is number 1 ~ 8 for LinPAC.

freq:

The frequency for Yrig.O output (20 - 200000)

** realfreq:*

[Output]The real frequency.

The frequency is made by dividing the basic clock with 16 bit resolution.

Return Values

Please refer to Error Code.

Example

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    int slot=1;
    unsigned long freq=20000,realfreq=0;
    pac_i8012W_Init(slot);
    pac_i8012W_ConfigTrigOut(slot,freq,&realfreq);
    return 0;
}
```

[C#]

```
static void Main(string[] args){
        int slot = 1;
        uint freq = 20000, realfreq=0;
        pac8012WNet.pac8012W.Init(slot);
        pac8012WNet.pac8012W.ConfigTrigOut(slot, freq, ref realfreq);
}
```

[C] (LinPAC)

```
int main(){
    int slot=1, ret=0;
    unsigned long freq=20000,realfreq=0;
    ret=Open_Slot(slot);
    if (ret > 0) {
            printf("Open Slot%d failed, return value=%d \n", slot, ret);
            return (-1);
    }
    i9012_Init(slot);          // Find the first I-9012
    i9012_ ConfigTrigOut(slot, freq, &realfreq);
    Close_Slot(slot);
    return 0;
}
```

3.17. pac_i8012W_Enable_TrigOut

This function is used to start output of Trig.O pin.

Syntax

For Windows (CE and WES)

```
short pac_i8012W_Enable_TrigOut(int slot);
```

For Linux

```
short i9012_Enable_TrigOut(int slot);
```

Parameter

slot:

Specific slot number (0 - 7), except range of slot is number 1 ~ 8 for LinPAC.

Return Values

Please refer to Error Code.

Example

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    int slot=1;
    unsigned long freq=20000,realfreq=0;
    pac_i8012W_Init(slot);
    pac_i8012W_ConfigTrigOut(slot,freq,&realfreq);
    pac_i8012W_Enable_TrigOut(slot);
    return 0;
}
```

[C#]

```
static void Main(string[] args){
        int slot = 1;
        uint freq = 20000, realfreq=0;
        pac8012WNet.pac8012W.Init(slot);
        pac8012WNet.pac8012W.ConfigTrigOut(slot, freq, ref realfreq);
        pac8012WNet.pac8012W.EnableTrigOut(slot);
}
```

[C] (LinPAC)

```
int main(){
    int slot=1, ret=0;
    unsigned long freq=20000,realfreq=0;
    ret=Open_Slot(slot);
    if (ret > 0) {
            printf("Open Slot%d failed, return value=%d \n", slot, ret);
            return (-1);
    }
    i9012_Init(slot);          // Find the first I-9012
    i9012_ ConfigTrigOut(slot, freq, &realfreq);
    i9012_Enable_TrigOut(slot);
    Close_Slot(slot);
    return 0;
}
```

3.18. pac_i8012W_Disable_TrigOut

This function is used to stop output of Trig.O pin.

Syntax

For Windows (CE and WES)

```
short pac_i8012W_Disable_TrigOut(int slot);
```

For Linux

```
short i9012_Disable_TrigOut(int slot);
```

Parameter

slot:

Specific slot number (0 - 7), except range of slot is number 1 ~ 8 for LinPAC.

Return Values

Please refer to Error Code.

Example

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    int slot=1;
    unsigned long freq=20000,realfreq=0;
    pac_i8012W_Init(slot);
    pac_i8012W_ConfigTrigOut(slot,freq,&realfreq);
    pac_i8012W_Disable_TrigOut (slot);
    return 0;
}
```

[C#]

```
static void Main(string[] args){
        int slot = 1;
        uint freq = 20000, realfreq=0;
        pac8012WNet.pac8012W.Init(slot);
        pac8012WNet.pac8012W.ConfigTrigOut(slot, freq, ref realfreq);
        pac8012WNet.pac8012W.DisableTrigOut(slot);
}
```

[C] (LinPAC)

```
int main(){
    int slot=1, ret=0;
    unsigned long freq=20000,realfreq=0;
    ret=Open_Slot(slot);
    if (ret > 0) {
            printf("Open Slot%d failed, return value=%d \n", slot, ret);
            return (-1);
    }
    i9012_Init(slot);          // Find the first I-9012
    i9012_ ConfigTrigOut(slot, freq, &realfreq);
    i9012_Disable_TrigOut (slot);
    Close_Slot(slot);
    return 0;
}
```

3.19. pac_i8012W_ReadGainOffset

This function is used to read calibrated gain value and offset value.

Syntax

For Windows (CE and WES)

```
short pac_i8012W_ReadGainOffset(int slot,int ch,int gain,unsigned short* GainValue, short* offsetValue);
```

For Linux

```
short i9012_ReadGainOffset(int slot,int ch,int gain,unsigned short* GainValue, short* offsetValue);
```

Parameter

slot:

Specific slot number (0 - 7), except range of slot is number 1 ~ 8 for LinPAC.

ch:

Specific channel (0 - 7).

gain:

Specific input range (0 - 1)

0: +/- 5V

1: +/- 10V

**GainValue:*

[Output] The gain value for the input range of specific channel.

**offsetValue:*

[Output] The offset value for the input range of specific channel.

Return Values

Please refer to Error Code.

Example

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    int slot=1,gain=0,ch=0;
    unsigned short GainValue=0;
    short offsetValue=0;
    pac_i8012W_Init(slot);
    for(gain=0;gain<2;gain++)
          for(ch=0;ch<8;ch++)
                pac_i8012W_ReadGainOffset(slot,ch,gain,&GainValue,&offsetValue);
    return 0;
}
```

[C#]

```
static void Main(string[] args){
       int slot = 1, gain = 0, ch = 0;
       ushort gainval = 0;
       short offset = 0;
       pac8012WNet.pac8012W.Init(slot);
       for (gain = 0; gain < 2; gain++)
             for (ch = 0; ch < 8;ch++ )
                   pac8012WNet.pac8012W.ReadGainOffset(slot, ch, gain, ref gainval, ref
offsetval);
}
```

[C] (LinPAC)

```
int main(){
    int slot=1, ret=0, gain=0, ch=0;
    unsigned short GainValue=0;
    short offsetValue=0;
    ret=Open_Slot(slot);
    if (ret > 0) {
        printf("Open Slot%d failed, return value=%d \n", slot, ret);
        return (-1);
    }
    i9012_Init(slot);
    for(gain=0; gain<2; gain++)
        for(ch=0; ch<8; ch++)
            i9012_ReadGainOffset(slot,ch,gain,&GainValue,&offsetValue);
    Close_Slot(slot);
    return 0;
}
```

3.20. pac_i8012W_CalibrationHEX

This function is used to calibrate data that read from FIFO.

Syntax

For Windows (CE and WES)

```
int pac_i8012W_CalibrationHEX(int slot,int ch,int gain,short raw,short* hval);
```

For Linux

```
int i9012_CalibrationHEX(int slot, int ch, int gain, short raw, short* hval);
```

Parameter

slot:

Specific slot number (0 - 7), except range of slot is number 1 ~ 8 for LinPAC.

ch:

Specific channel (0 - 7).

gain:

Specific input range (0 - 1)

0: +/- 5V

1: +/- 10V

raw:

The data is read from the FIFO and is not calibrated.

**hval:*

[Output] The calibrated hexadecimal format data.

Return Values

Please refer to Error Code.

Example

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    int slot=1,ch=0,gain=0,i=0;
    short raw[10],hval[10];
    pac_i8012W_Init(slot);
    ...
    //raw[10] store the data that read from FIFO.
    ...
    for(i=0;i<10;i++)
        pac_i8012W_CalibrationHEX(slot,ch,gain,raw[i],&hval[i]);
    return 0;
}
```

[C#]

```
static void Main(string[] args){
        int slot = 1, ch = 0, gain = 0,i = 0;
        short[] raw = new short[10];
        short[] hval = new short[10];
        pac8012WNet.pac8012W.Init(slot);
        ...
        //raw[10] store the data that read from FIFO.
        ...
        for (i=0;i<10;i++)
            pac8012WNet.pac8012W.CalibrationHEX(slot,ch,gain,raw[i],ref hval[i]);
}
```

[C] (LinPAC)

```
int main(){
    int slot=1, ret=0, gain=0, ch=0, i=0;
    short[] raw = new short[10];    short[] hval = new short[10];
    ret=Open_Slot(slot);
    if (ret > 0) {
          printf("Open Slot%d failed, return value=%d \n", slot, ret);
          return (-1); }
    i9012_Init(slot);
    //raw[10] store the data that read from FIFO.
    for (i=0; i<10; i++)    i9012_ReadGainOffset(slot, ch, gain, &GainValue, &offsetValue);
    Close_Slot(slot);
    return 0;
}
```

3.21. pac_i8012W_CalibrationFloat

This function is used to calibrate data that read from FIFO.

Syntax

For Windows (CE and WES)

```
int pac_i8012W_CalibrationFloat(int slot,int ch,int gain,short raw,float* fval);
```

For Linux

```
int i9012_CalibrationFloat(int slot,int ch,int gain,short raw,float* fval);
```

Parameter

slot:

Specific slot number (0 - 7), except range of slot is number 1 ~ 8 for LinPAC.

ch:

Specific channel (0 - 7).

gain:

Specific input range (0 - 1)

0: +/- 5V

1: +/- 10V

raw:

The data is read from the FIFO and is not calibrated.

**fval:*

[Output] The calibrated float format data.

Return Values

Please refer to Error Code.

Example

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    int slot=1,ch=0,gain=0,i=0;
    short raw[10]
    float fval[10];
    pac_i8012W_Init(slot);
    ...
    //raw[10] store the data that read from FIFO.
    ...
    for(i=0;i<10;i++)
          pac_i8012W_CalibrationFloat(slot,ch,gain,raw[i],&fval[i]);
    return 0;
}
```

[C#]

```
static void Main(string[] args){
       int slot = 1, ch = 0, gain = 0,i = 0;
       short[] raw = new short[10];
       float[] fval = new float[10];
       pac8012WNet.pac8012W.Init(slot);
       ...
       //raw[10] store the data that read from FIFO.
       ...
       for (i=0;i<10;i++)
             pac8012WNet.pac8012W.CalibrationFloat (slot,ch,gain,raw[i],ref fval[i]);
}
```

[C] (LinPAC)

```
int main(){
    int slot=1, ret=0, gain=0, ch=0, i=0;
    short raw[10];          float fval[10];
    ret=Open_Slot(slot);
    if (ret > 0) {
          printf("Open Slot%d failed, return value=%d \n", slot, ret);
          return (-1);    }
    i9012_Init(slot);
    //raw[10] store the data that read from FIFO.
    for (i=0; i<10; i++)    i9012_ CalibrationFloat(slot, ch, gain, raw[i], &fval[i]);
    Close_Slot(slot);
    return 0;    }
```

4. Calibration

Each module calibrated and finished test before shipment, so usually it is unnecessary to calibrate the module again, unless the input impedance is changed or the accuracy is lost.

In order to calibrate the module, the following preparations are required:

- A single stable calibration source, such as a 3 1/2 digit power supply (or better) or a battery output.
- A single 4 1/2 digit voltage meter (15-bit resolution or better)
- A Calibration Program. Please visit ICP DAS website and download demo programs, the calibration program will be inside.

Tips & Warnings

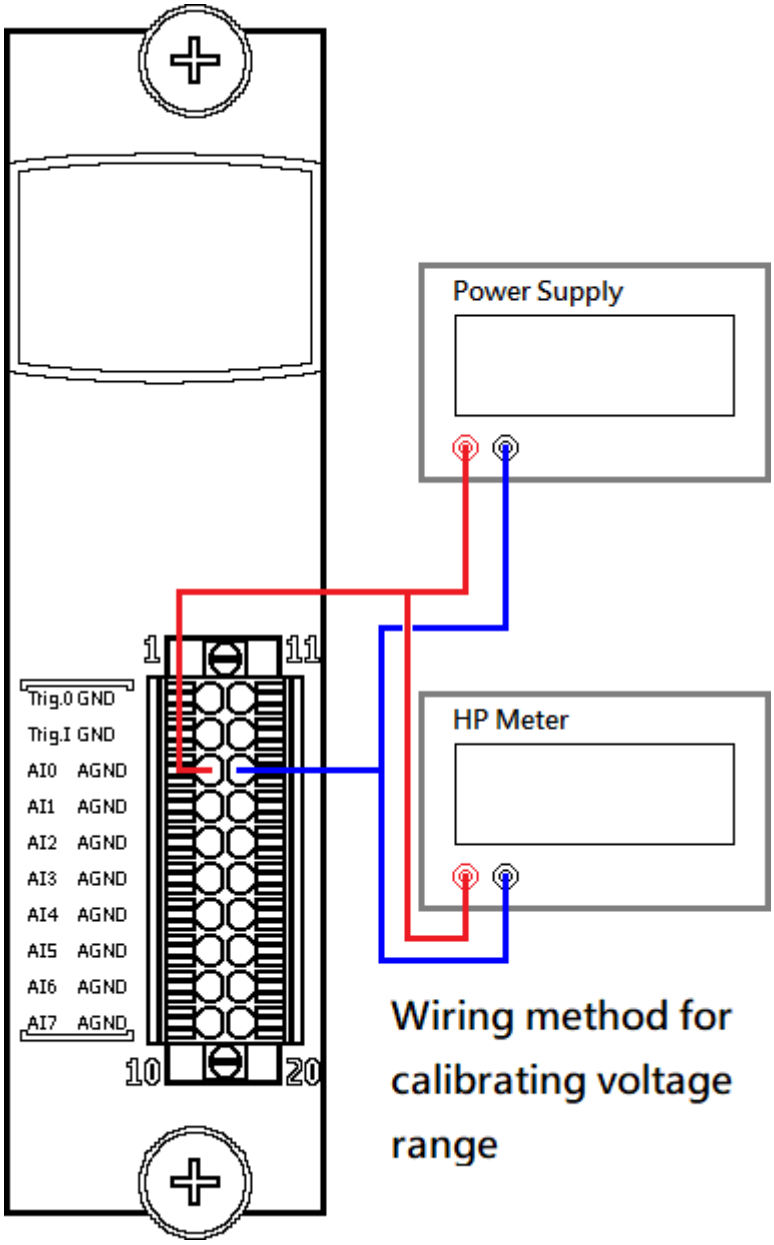
1. An unstable calibration source will cause calibration errors and will affect the accuracy of the data acquisition.

2. I-9012 needs to calibrate every channel one by one with both ranges of +/- 5V and +/- 10V.

4.1. Calibrate the I-9012 on WinCE and WES units

Step 1. Wiring method

Connect the channel to be calibrated, source and voltage meter together, like the following figure:



Then, plug module into the controller.

Step 2. Download and execute calibration program

The calibration program can be downloaded in ICP DAS website.

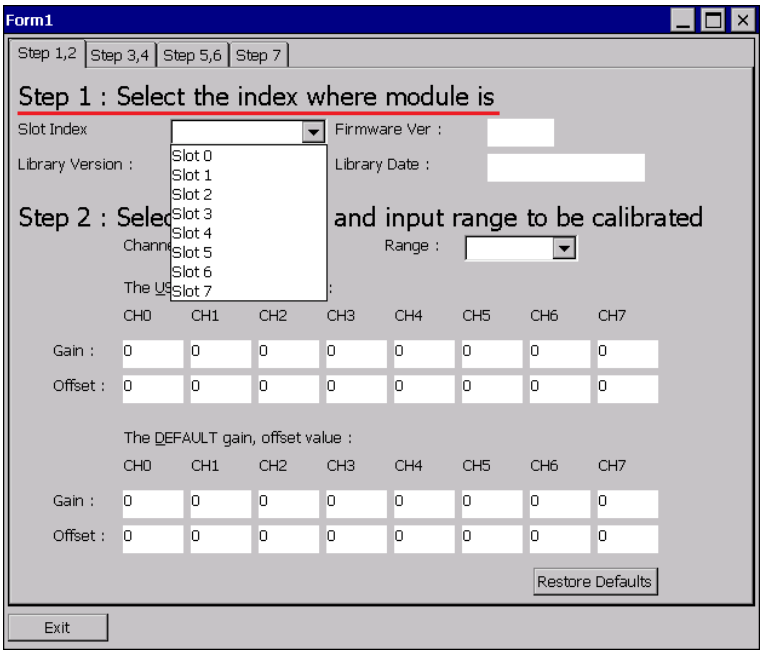
Please refer to the following link:

https://www.icpdas.com/en/download/show.php?num=2905

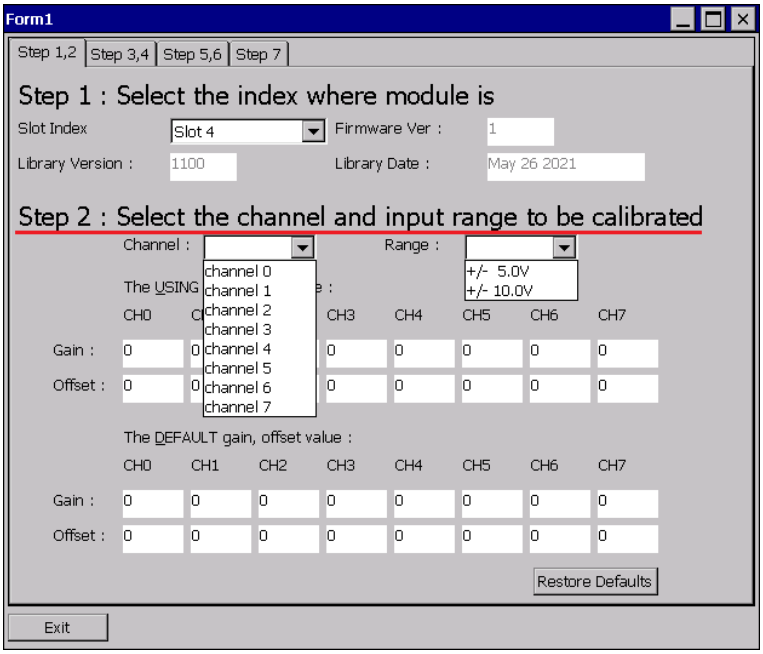
Step 3. Calibrate

After execute the program, please follow the steps one by one.

Select the index where the module is.

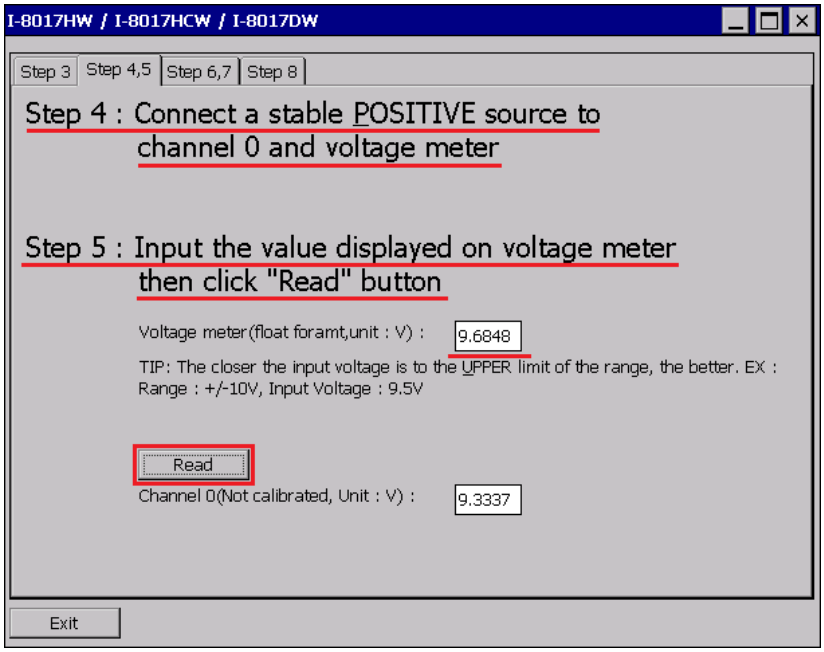


Select the channel and range to be calibrated.



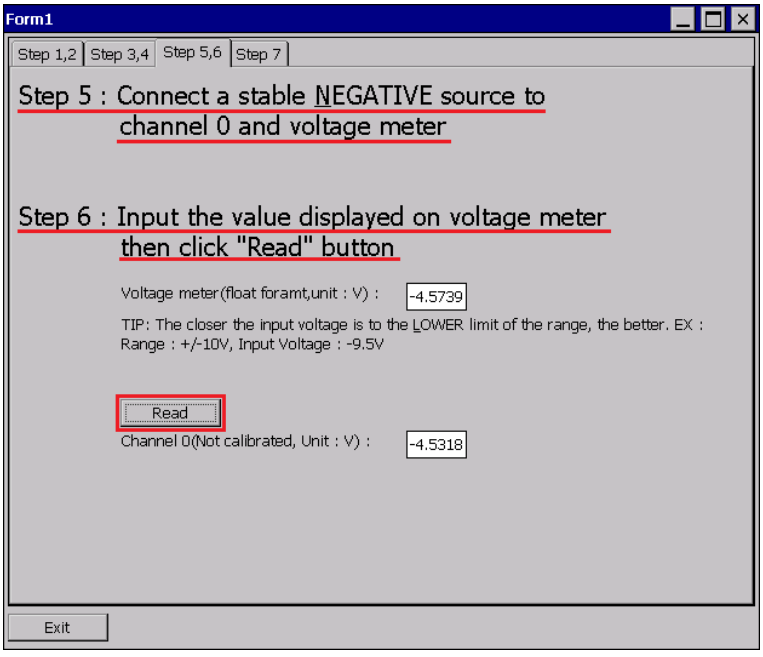
Change to “page Step 3,4”,

Output stable positive source the channel and type the value displayed on the voltage meter, then Click “Read” button.



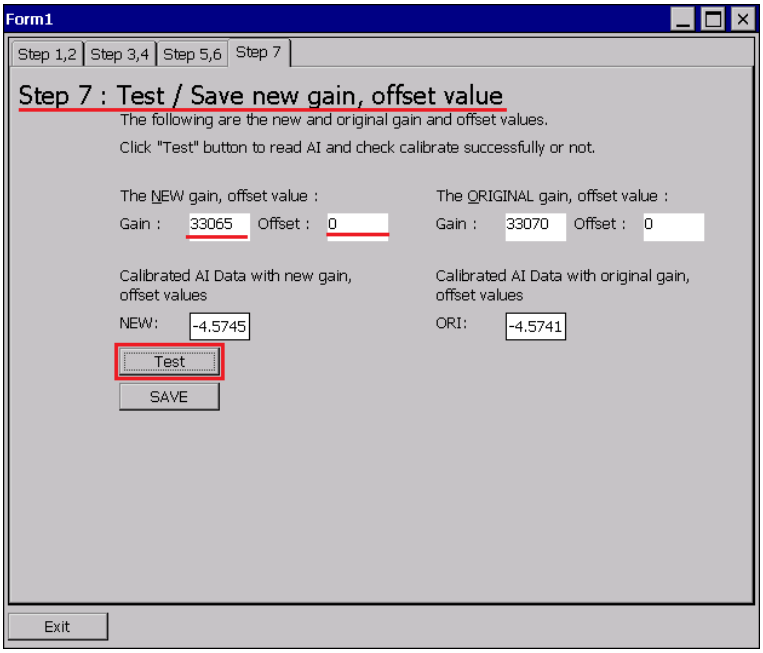
Change to “page Step 5,6”,

Output stable negative source to the channel and type the value displayed on the voltage meter, then Click “Read” button.

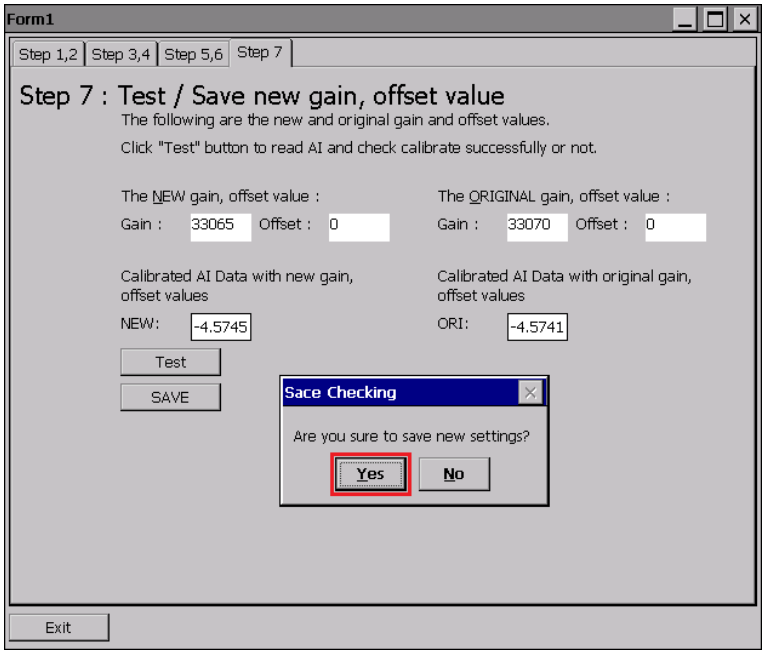


Change to “page Step 7”,

Click “Test” to read calibrated AI data with new and original gain and offset values , and check whether the new gain and offset values are correct or not.



Click “SAVE” to save new gain and offset values.



4.2. Restore I-9012 to defaults on WinCE and WES units

Step 1. Download and execute calibration program

The calibration program can be downloaded in ICP DAS website.

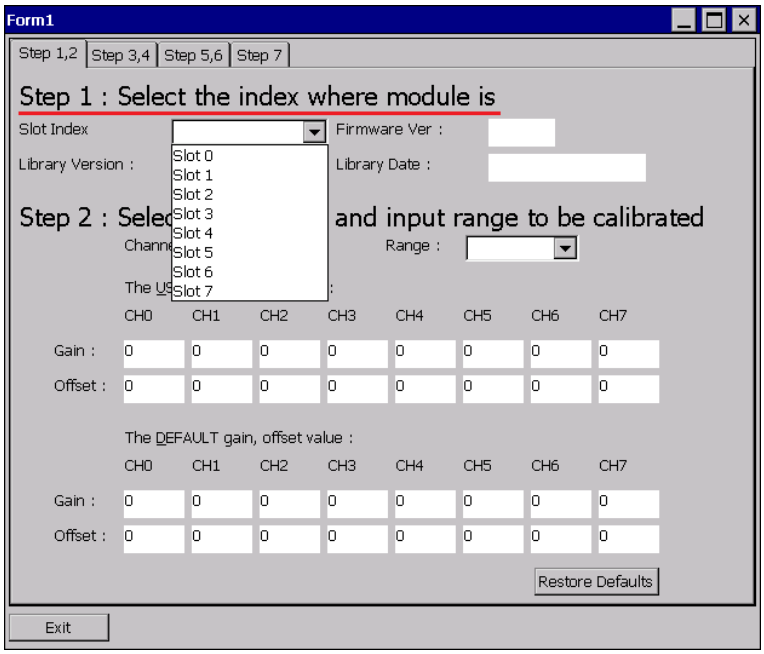
Please refer to the following link:

https://www.icpdas.com/en/download/show.php?num=2905

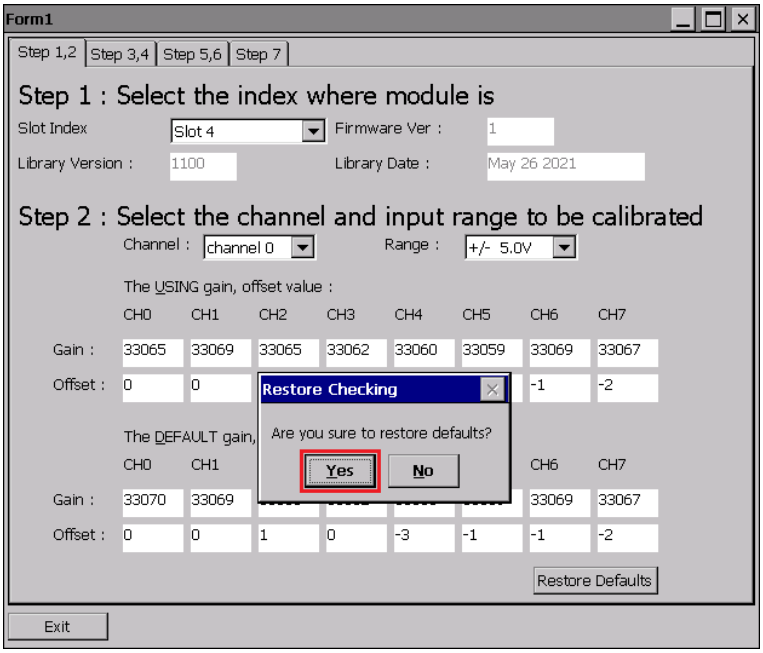
Step 2. Restore defaults

After execute the program, please follow the steps one by one.

Select the index where the module is.



Click “Restore Defaults” button.



Appendix A. Error Code

Error Code	Definition	Description
0	NoError	This indicates that there have been no errors
-1	ID_ERROR	There was a problem with the module ID
-2	FIFO_FULL	The FIFO is full, the data is wrong.
-3	FIFO_EMPTY	The FIFO is empty, no data in FIFO.

Appendix B. Read AI Function Performance

Test with library version 0x1101.

Test Items \ Platform			CE 7.0	WES	IOT
I-9012	HEX format	One channel	93 ~ 96	29	30
		Eight channels	93 ~ 96	29	30
	Float format	One channel	92 ~ 95	29	30
		Eight channels	92 ~ 95	29	30

Unit : K Hz.

Appendix C. Revision History

This chapter provides revision history information to this document.

The table below shows the revision history.

Revision	Date	Description
1.0.0	May 2019	Initial issue
1.0.1	May 2021	• Added calibrate steps. • Added optioning module on Linux platform in Quick start. • Modify the download path of libraries and demos. • Modify performance information for all platforms.