I-8014(C)W/I-9014(C)

I/O Module User Manual

V2.0.2 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, “Magic Scan” – This chapter introduces the functions of I-8014W, such as read AI, Magic Scan, the programming procedures, and demo programs.
- Chapter 4, “API introduction” – This chapter describes the functions provided in the I-8014W library together with an explanation of the differences in the naming rules used for the MiniOS7 and Windows platforms.
- Chapter 5, “Troubleshooting” – This chapter provides some troubleshooting solutions should you encounter any problems while operating the I-8014W.

1. Introduction

I-8014W, I-8014CW, I-9014 and I-9014C are high performance analog input module.

I-8014W and I-9014 provided 16 single-ended or 8 differential channels.

I-8014CW and I-9014C provided 8 differential channels.

They feature 16-bit resolution, 250Ks/s sampling rates, and 4K-sample FIFO.

I-8014W/I-8014CW/I-9014/I-9014C (Hereinafter referred to as I-8014W series modules) contain an impressive scan function called Magic Scan, which are able to improve many of the functions and meets the demands of high-end users. The Magic Scan mechanism not only scans the different input channels at vastly different rates, but also at different gains.

Even in a multi-channel scan, the sampling rates can be maintained at 250KS/s.

I-8014W includes a 4K onboard FIFO buffer for A/D conversion. With the Magic Scan function and 4K FIFO, the I-8014W can easily implement high-speed and time-critical data acquisition applications.

The following table shows the differences between I-8014W/I-9014 and I-8014CW/I-9014C:

	I-8014W/I-9014	I-8014CW/I-9014C
Input Range	+/- 10 V, +/- 5 V, +/- 2.5 V, +/- 1.25 V and +/- 20 mA	+/- 20 mA only
Input Type	Differential or Single-Ended Mode	Differential Mode only
Wire Connection for current measurement	Need external 125 ohm resistor for current measurement	Do not need external resistor for current measurement
Calibration Parameter	8 channels AI using 1 calibration parameter	8 channels AI using independent calibration parameter

Applicable Platform table

The following table shows which platform the module applies to.

Platform	OS	Module
XPAC	XP-8000(WES)	I-8014W
	XP-8000-Atom (WES)	I-8014W
	XP-8000-WES7 (WES7)	I-8014W
	XP-8000-CE6 (WinCE 6.0)	I-8014W
	XP-8000-Atom-CE6 (WinCE 6.0)	I-8014W
	XP-9000-WES7(WES7)	I-9014
WPAC	WP-8000 (CE 5.0/7.0)	I-8014W
	WP-9000-CE7 (CE 7.0)	I-9014
LinPAC	LinPAC-8000 (Linux kernel 2.6~4.4)	I-8014W
	LP-9000/LX-9000 (Linux kernel 3.2/4.4)	I-9014
IPAC	iPAC-8000 (MiniOS7)	I-8014W

1.1. Features

I-8014W/I-9014

- 16 single-ended/8 differential inputs (jumper selectable)
- Input Range : +/- 10V, +/- 5V, +/- 2.5V, +/- 1.25V, +/- 20mA(Software selectable)
- 16-bit 250KHz ADC converter
- 4K-samples FIFO buffer
- External trigger mode : post-trigger
- Internal/external trigger start

I-8014CW/I-9014C

- 8 differential inputs
- Input Range : +/- 20mA
- 16-bit 250KHz ADC converter
- 4K-samples FIFO buffer
- External trigger mode : post-trigger
- Internal/external trigger start

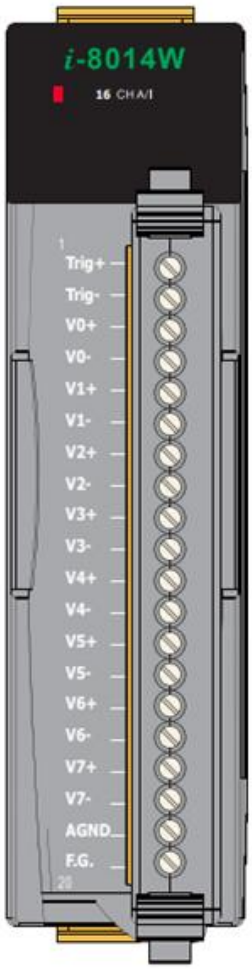
1.2. Specifications

Model	I-8014W/I-9014	I-8014CW/I-9014C
Analog Output		
Channels	8-ch Differential/16-Single-ended	8-ch Differential
Voltage Input Range	±1.25, ±2.5, ±5 V, ±10 V	-
Current Input Range	-20 mA ~ +20 mA(Requires Optional External 125 Ω Resistor)	-20 mA ~ +20 mA
Resolution	16-bit	
Sample Rate	Single Channel Polling Mode :250K S/s	
FIFO	4 k Words	
Accuracy	0.05% of FSR	
Input Mode	Polling, Pacer (Magic Scan)	
Magic Scan Mode	Mode 1: Standard Mode Mode 2: Virtual Sample and Hold	
Overvoltage Protection	-45 V ~ +60 V	
Input Impedance	20 K, 200 K, 20 M (Jumper Select)	125 Ω
LED Indicators		
Power LED Indicator	Yes	
Isolation		
Intra-module Isolation, Field-to-Logic	2500 Vrms	
Power		
Power Consumption	2.5 W Max.	

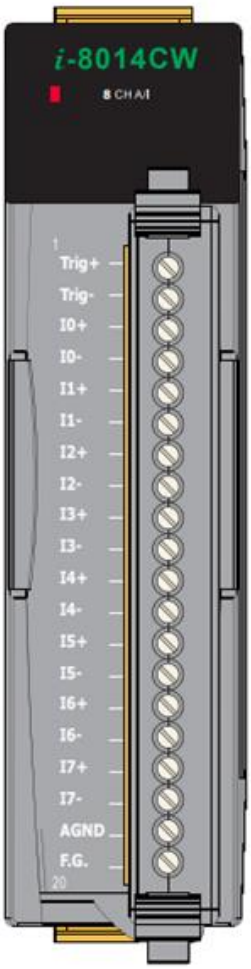
Model	I-8014W/I-9014	I-8014CW/I-9014C
Mechanical		
Dimension (L x W x H)	For I-8014IW: 102 mm x 30 mm x 115 mm For I-9014I: 144 mm x 31 mm x 134 mm	
Environment		
Operating Temperature	-25 °C ~ +75°C	
Storage Temperature	For I-8014IW: -30 °C ~ +85°C For I-9014I: -40°C ~ +85°C	
Humidity	10 % ~ 90% RH, non-condensing	

1.3. Pin Assignments

1.3.1. I-8014W/I-8014CW

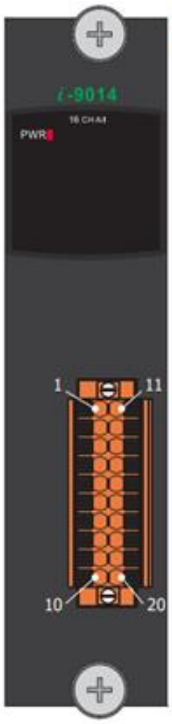


Terminal No.		Pin Assignment	
		Differential	Single-ended
	01	Trig+	Trig+
	02	Trig-	Trig-
	03	V0+	V0
	04	V0-	V8
	05	V1+	V1
	06	V1-	V9
	07	V2+	V2
	08	V2-	V10
	09	V3+	V3
	10	V3-	V11
	11	V4+	V4
	12	V4-	V12
	13	V5+	V5
	14	V5-	V13
	15	V6+	V6
	16	V6-	V14
	17	V7+	V7
	18	V7-	V15
	19	AGND	AGND
	20	F.G.	F.G.



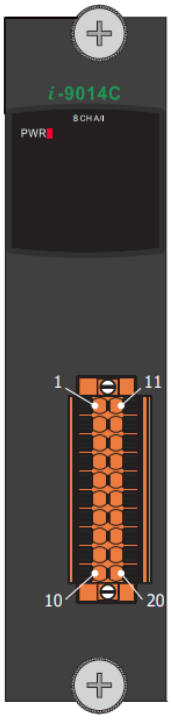
Terminal No.		Pin Assignment
	01	Trig+
	02	Trig-
	03	I0+
	04	I0-
	05	I1+
	06	I1-
	07	I2+
	08	I2-
	09	I3+
	10	I3-
	11	I4+
	12	I4-
	13	I5+
	14	I5-
	15	I6+
	16	I6-
	17	I7+
	18	I7-
	19	AGND
	20	F.G.

1.3.2. I-9014W/I-9014C



I-9014

Pin Assignment		Terminal No.			Pin Assignment	
Differential	Single-ended				Differential	Single-ended
Trig+	Trig+	01		11	Trig-	Trig-
V0+	V0	02		12	V0-	V8
V1+	V1	03		13	V1-	V9
V2+	V2	04		14	V2-	V10
V3+	V3	05		15	V3-	V11
V4+	V4	06		16	V4-	V12
V5+	V5	07		17	V5-	V13
V6+	V6	08		18	V6-	V14
V7+	V7	09		19	V7-	V15
AGND	AGND	10		20	F.G.	AGND

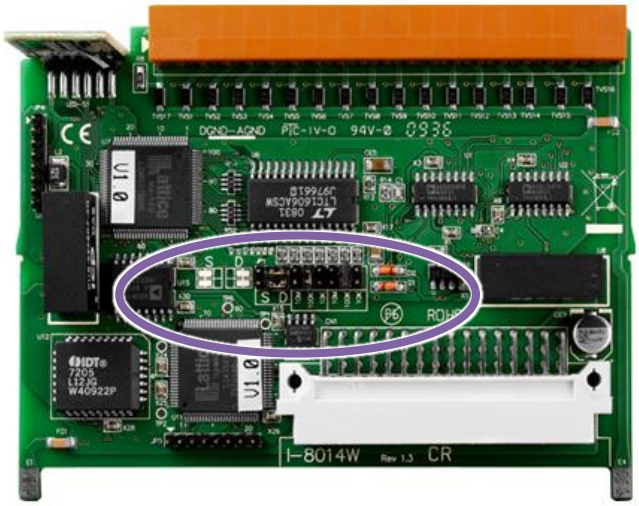


I-9014C

Pin Assignment	Terminal No.			Pin Assignment
Trig+	01		11	Trig-
I0+	02		12	I0-
I1+	03		13	I1-
I2+	04		14	I2-
I3+	05		15	I3-
I4+	06		16	I4-
I5+	07		17	I5-
I6+	08		18	I6-
I7+	09		19	I7-
AGND	10		20	F.G.

1.4. Jumper Settings

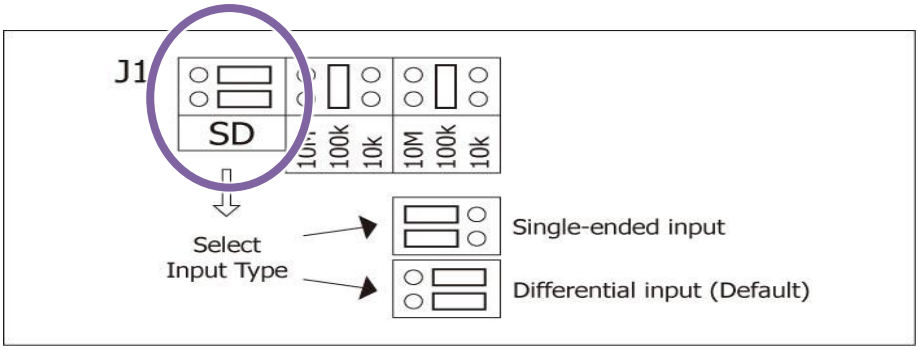
1.4.1. I-8014W/I-8014CW



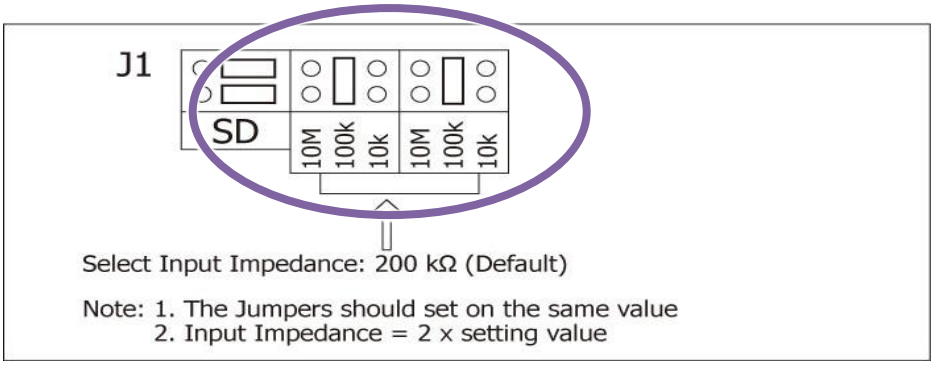
Tips & Warnings

I-8014CW do not have those Jumper, it is only with Differential Mode and Input impedance 20KΩ.

Differential / Single-ended Jumper Selection



Input impedance Jumper Selection

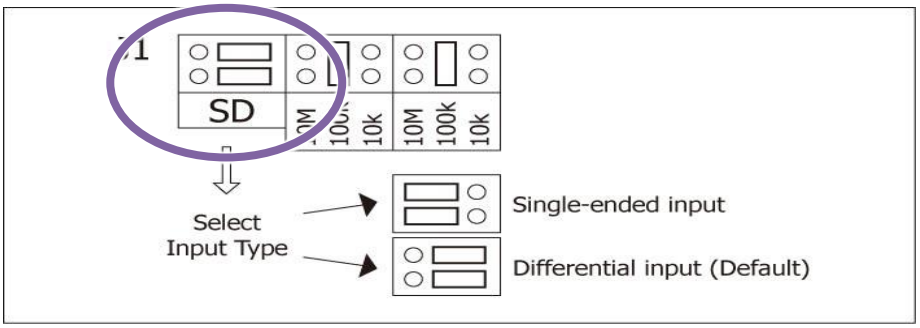


1.4.2. I-9014W/I-9014C

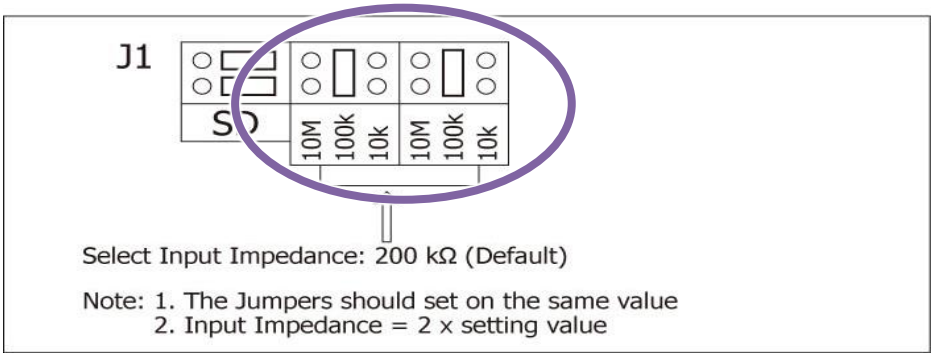
Tips & Warnings

I-9014C do not have those Jumper, it is only with Differential Mode and Input impedance 20 KΩ

Differential / Single-ended Jumper Selection



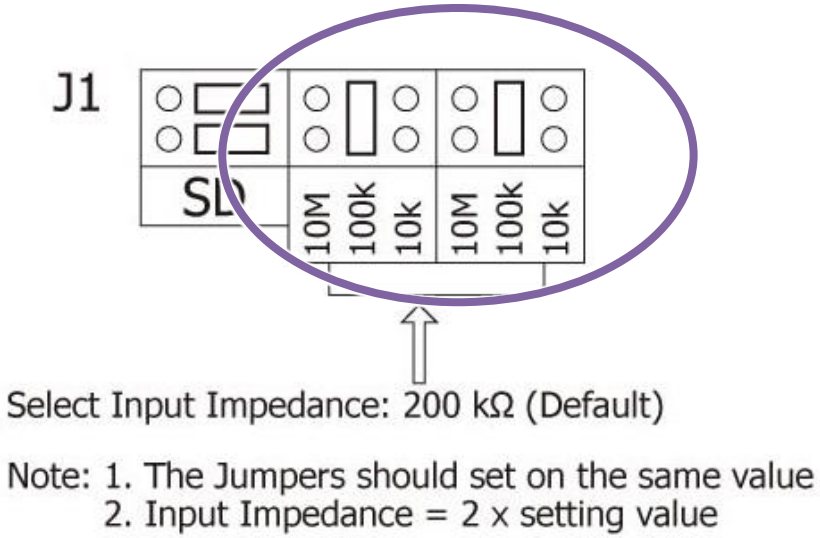
Input impedance Jumper Selection



Adjusting the Input impedance

I-8014W allows three input impedance options, including 20 KΩ, 200 KΩ (default setting) and 20 MΩ to meet system requirements. In most cases, 200KkΩ is sufficient.

Note that each time the input impedance is adjusted on a calibrated module, the module must be recalibrated. Refer to the Calibration section on page 19 if you are using an I-8000 or iPAC-8000 (MiniOS7 platform controller), or refer to page 32 for details of the calibration process if you are using a module based on the WinCE or WES platform.

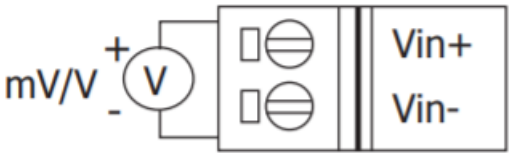
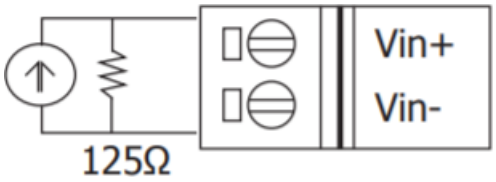
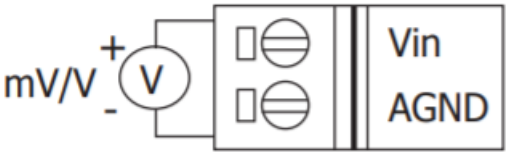
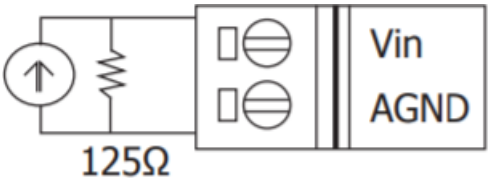


Select Input Impedance: 200 kΩ (Default)

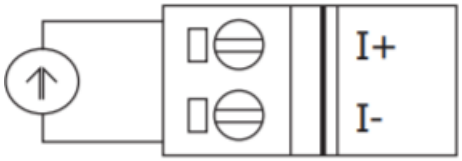
Note: 1. The Jumpers should set on the same value
2. Input Impedance = 2 x setting value

1.5. Wire Connections

1.5.1. I-8014W/I-9014

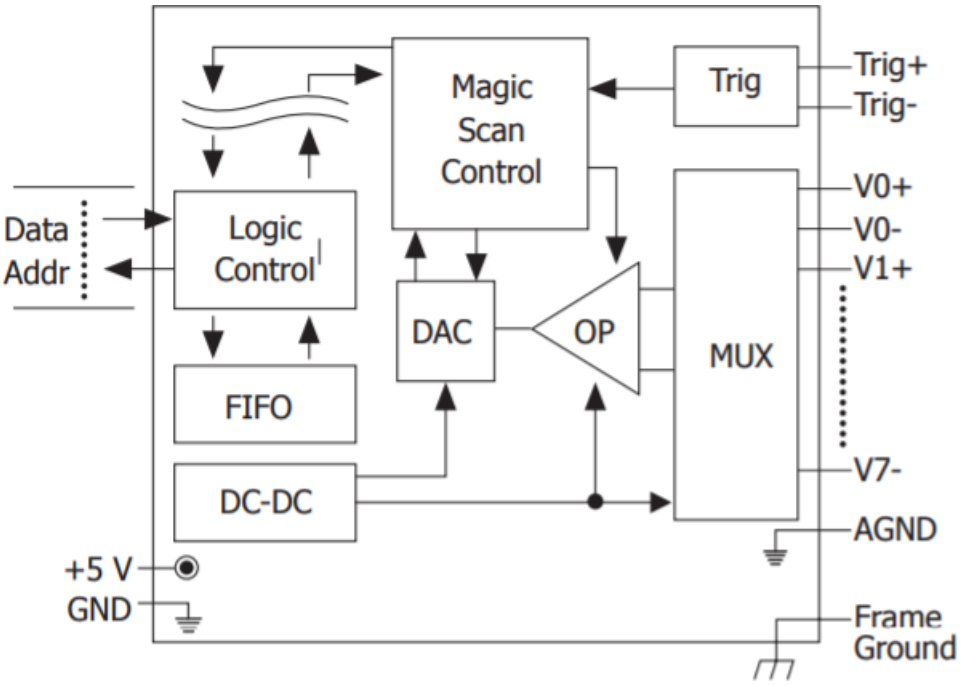
	Voltage Input Wiring	Current Input Wiring
Differential		
Single-ended		

1.5.2. I-8014CW/I-9014C

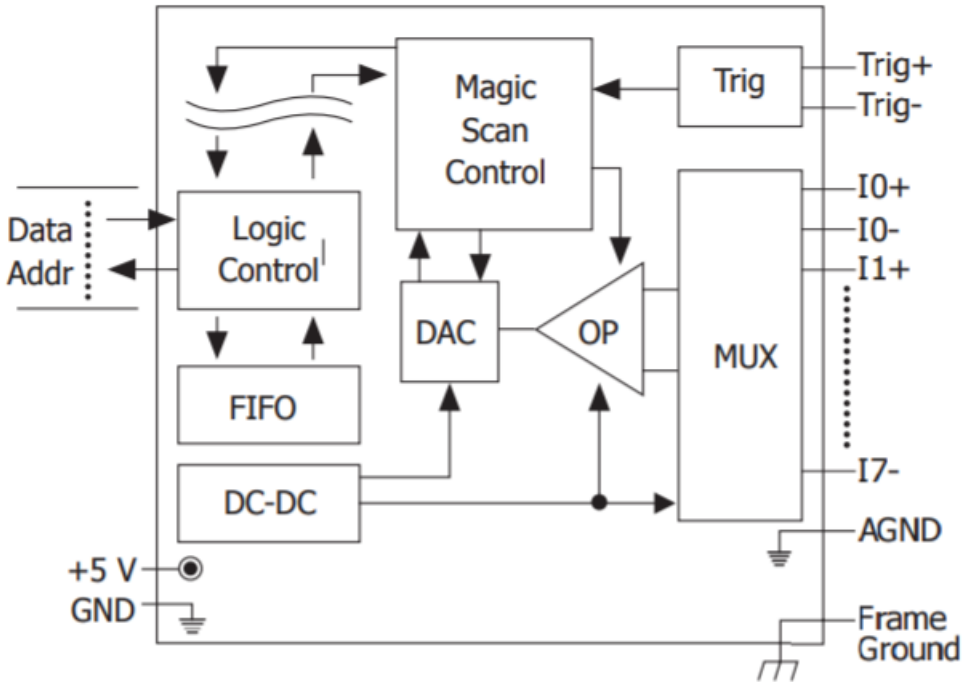
	Current Input Wiring
Differential	

1.6. Block Diagram

1.6.1. I-8014W/I-9014



1.6.2. I-8014CW/I-9014C



2. Quick Start

This chapter will be accompanied by demos to explain how to implement functions such as read AI, Magic Scan, calculate AI average, calibration process.

Demos can be downloaded in the following link:

https://www.icpdas.com/en/download/show.php?num=2897

2.1. MiniOS7-based Controllers

2.1.1. Basic Function

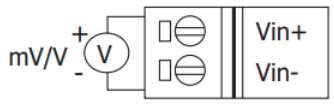
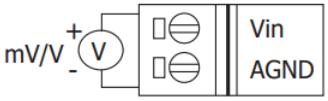
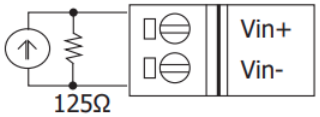
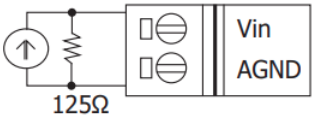
Basic function can be used to retrieve configuration information related to I-8014W and verify the AI reading function.

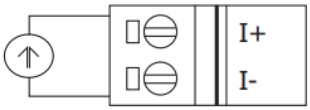
Basic configuration information includes:

- Version number and published date of the library.
- FPGA version.
- Single-ended/Differential jumper settings.
- Gain and offset values for each input range.
- Data reading of each channel.

The following steps take Base_Info.exe as example and display the information of I-8014W.

Step 1. Please refer to the “Wire Connections”, connect a stable signal source (such as a battery) to I-8014W

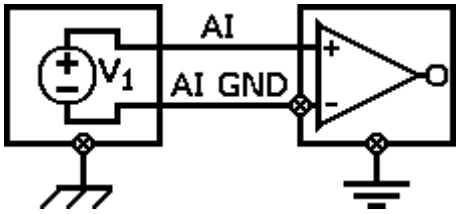
I-8014W			
Input Type	**Differential**	**Input Type**	**Single-ended**
Voltage Input Wiring		Voltage Input Wiring	
Current Input Wiring		Current Input Wiring	

I-8014CW	
Input Type	**Differential**
Current Input Wiring	

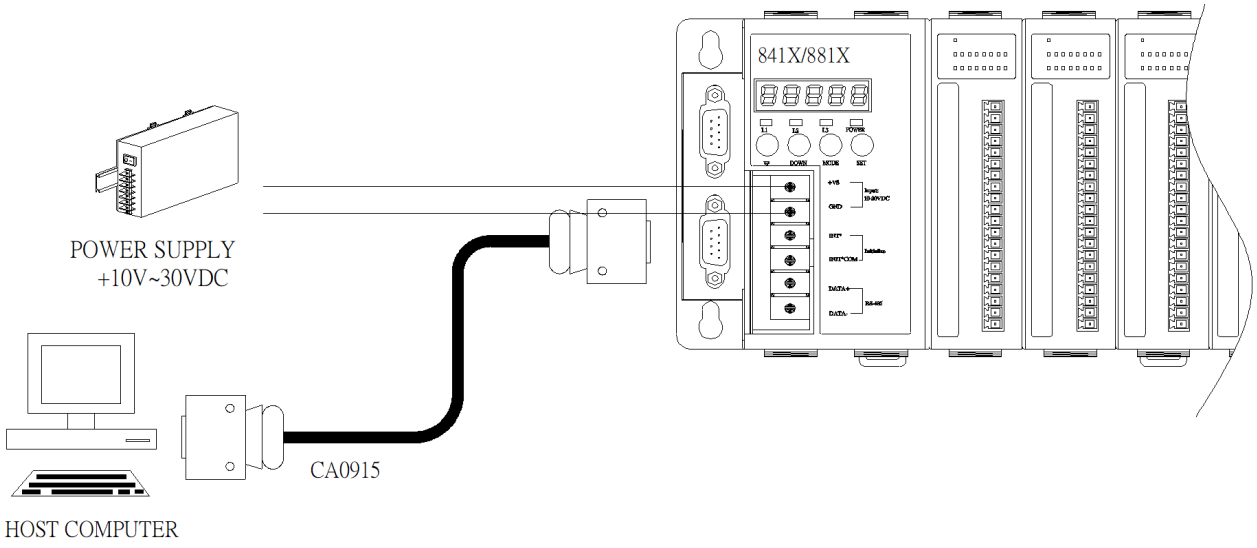
Tips & Warnings

1. Unused channels should be connected to GND to avoid floating.
2. A battery output can provide a stable enough signal for test.
3. A 125 Ohm resistor is required when measuring current input.
4. When measuring the voltage by differential input type, if the result is not as stable as the input signal, it is recommended that an additional is connected between the Vn- and the AGND (analog ground pin) to enhance the accuracy.

   When measuring current input, this method has no benefit in enhancing accuracy.



Step 2: Plug I-8014W in the MiniOS7 controller, connect the power supply to the unit and connect the unit to the Host PC by RS-232 cable

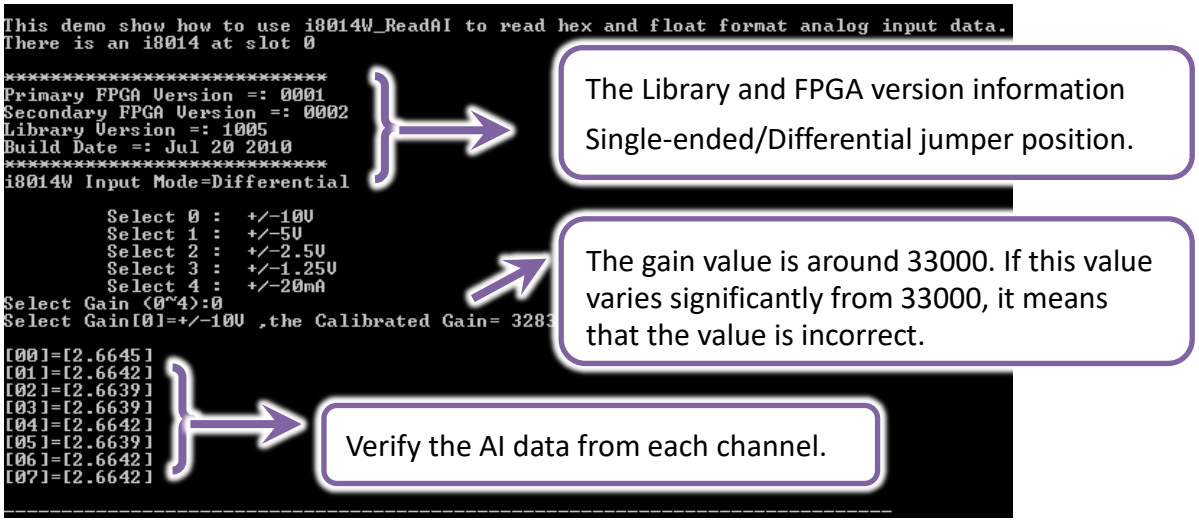


Step 3: Run 7188xw.exe on the host PC and open the COM Port which is connecting to the MiniOS7 Controller

Tips & Warnings

7188xw.exe is an interface for PC, it can help users to communicate with MiniOS7 PAC, please refer to the MiniOS7 PAC user manual for more detail.

Step 4: Run Base_Info.exe on the controller and verify that basic information and AI data from each channel are correct or not, as indicated in the diagram below:



Tips & Warnings

I-8014CW/I-9014C only can select max 8 channels and +/- 20 mA Input Range

2.1.2. Magic Scan Function

The magic scan function is used for high-speed sampling of data.

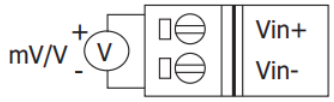
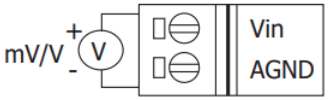
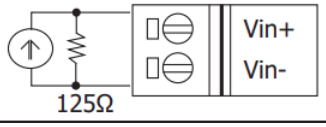
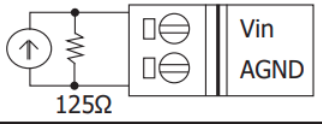
This is the major function of I-8014W series module.

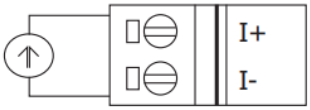
Configuration of Magic Scan function includes:

- Sample Channels
- Input Range
- Sample rate
- Sample count

The following steps take the MAGIC.EXE as example and display the process of data acquisition.

Step 1: Please refer to the "Wire Connections", connect a cyclically source (such as a Sine wave) to I-8014W

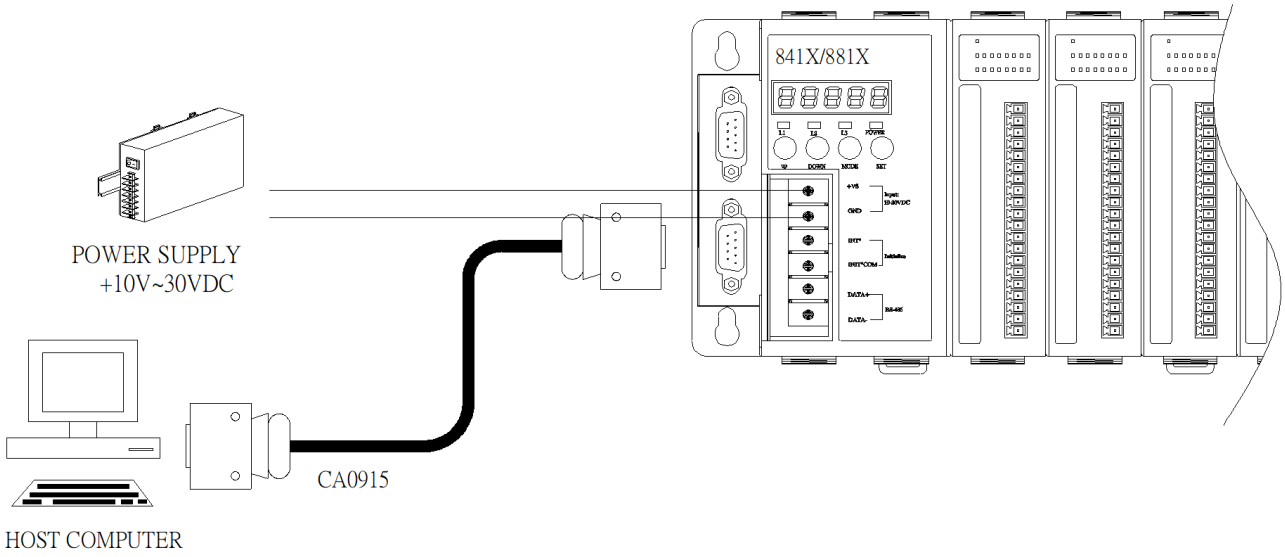
<table>
<tr><th colspan="4">I-8014W</th></tr>
<tr><th>Input Type</th><th>Differential</th><th>Input Type</th><th>Single-ended</th></tr>
<tr><td>Voltage Input Wiring</td><td> 
</td><td>Voltage Input Wiring</td><td> 
</td></tr>
<tr><td>Current Input Wiring</td><td> 
</td><td>Current Input Wiring</td><td> 
</td></tr>
</table>

<table>
<tr><th colspan="2">I-8014CW</th></tr>
<tr><th>Input Type</th><th>Differential</th></tr>
<tr><td>Current Input Wiring</td><td> 
</td></tr>
</table>

Tips & Warnings

Unused channels should be connected to GND to avoid floating.

Step 2: Plug I-8014W in the MiniOS7 controller, connect the power supply to the unit and connect the unit to the Host PC by RS-232 cable

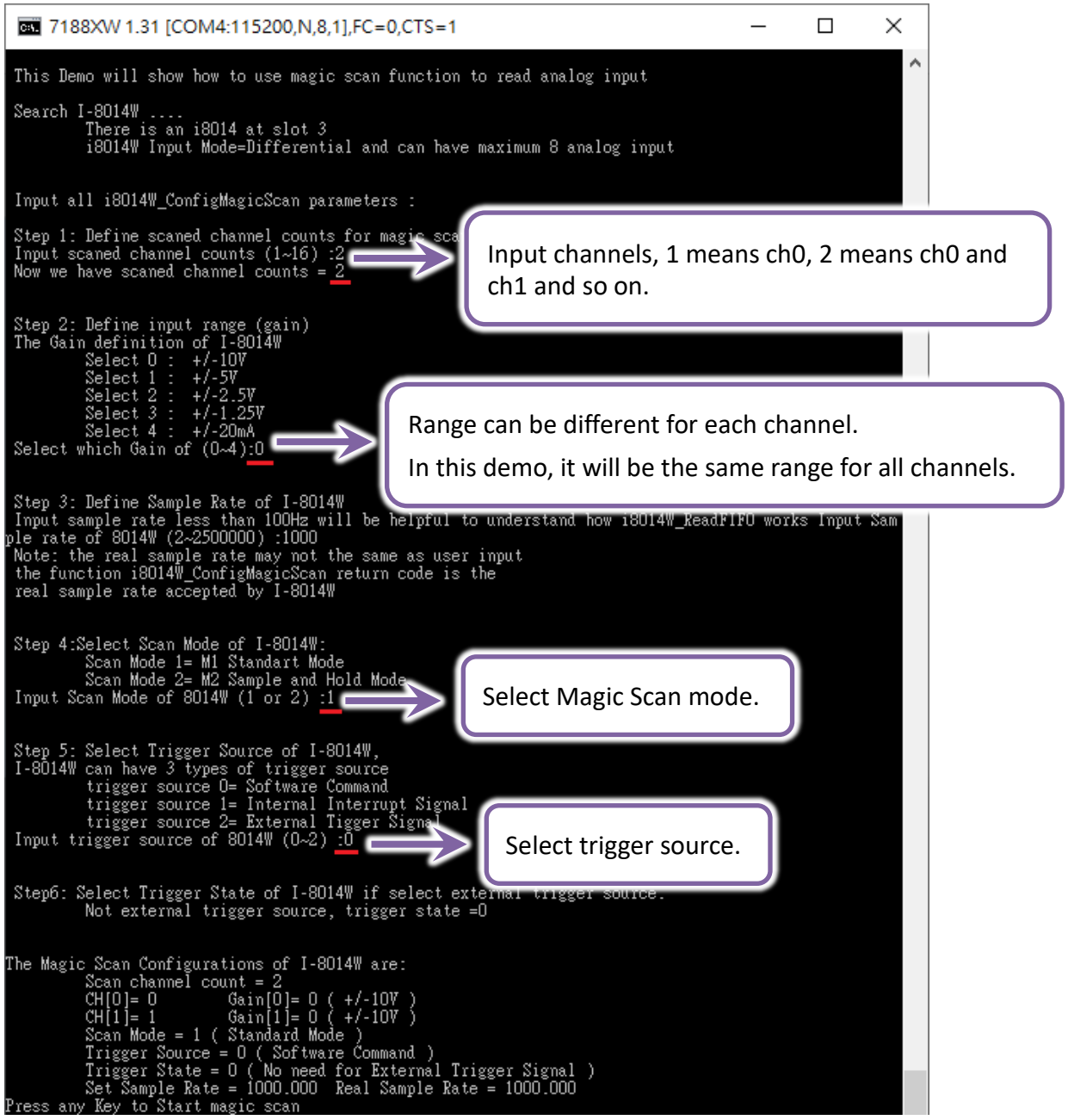


Step 3: Run 7188xw.exe on the host PC and open the COM Port which is connecting to the MiniOS7 Controller

Tips & Warnings

7188xw.exe is a interface for PC, it can help users to communicate with MiniOS7 PAC, please refer to the MiniOS7 PAC user manual for more detail.

Step 4: Run MAGIC.EXE on the controller and input parameters, as indicated in the diagram below:

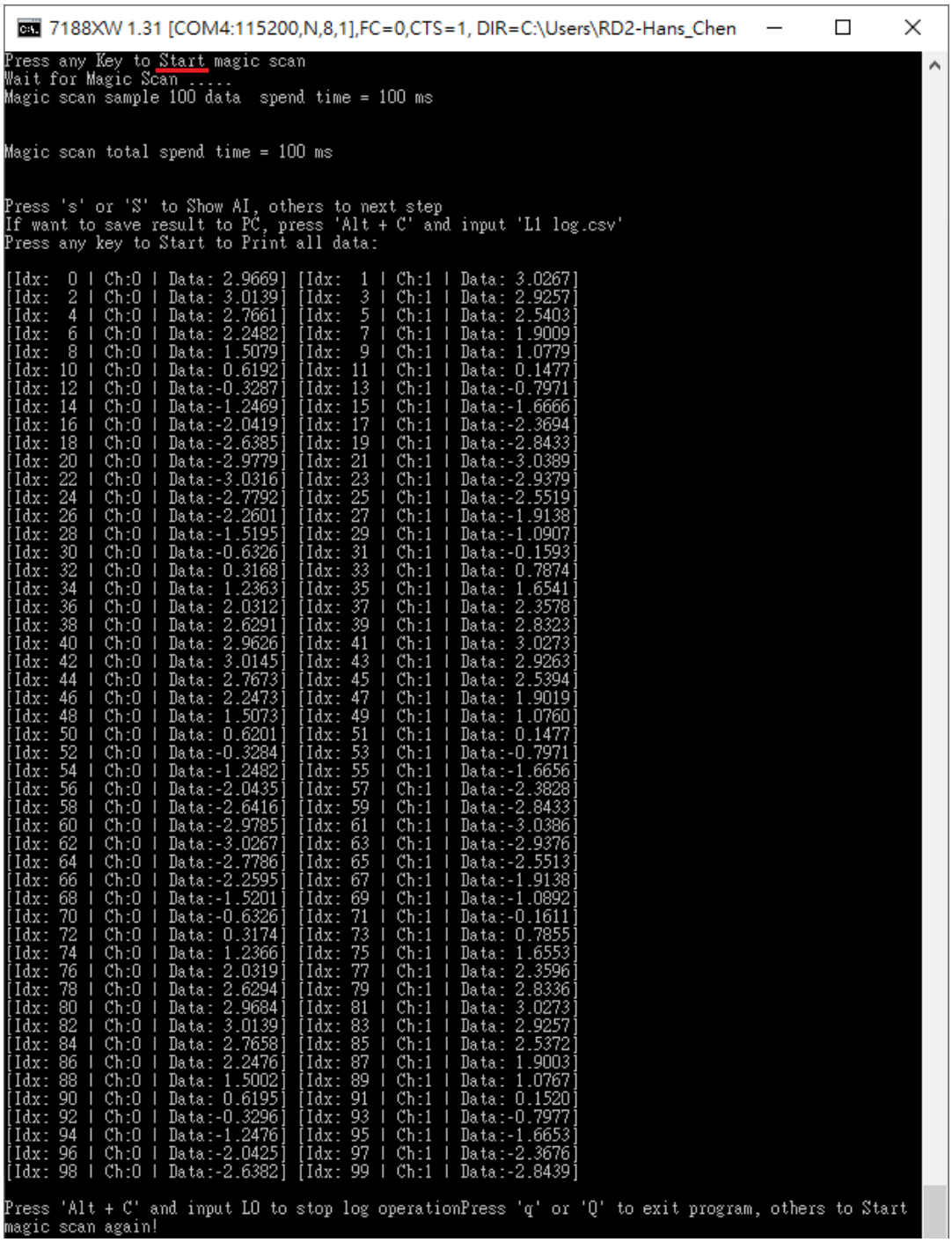


Tips & Warnings

I-8014CW/I-9014C only can select max 8 channels and +/- 20 mA Input Range.

In this demonstration, there is a 50 Hz, 3 Vpp, Sine wave connect with channel 0 and channel 1.

For more detail about the attributes of Magic Scan, please refer to 3. Magic Scan.



2.2. Windows-based Controllers

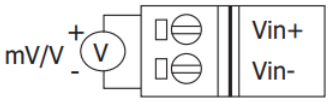
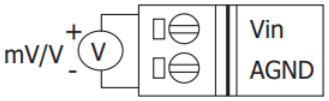
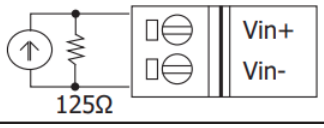
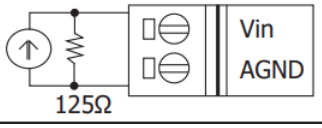
2.2.1. Basic Function

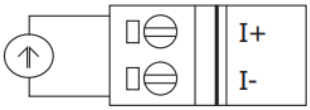
Basic function can be used to retrieve configuration information related to I-8014W and verify the AI reading function. Basic configuration information includes:

- Version number and published date of the library.
- FPGA version
- Single-ended/Differential jumper settings
- Gain and offset values for each input range
- Data reading of each channel

The following steps take pac_i8014W_ReadAI_CSharp.exe as example and display the information of I-8014W.

Step 1. Please refer to the "Wire Connections", connect a stable signal source (such as a battery) to I-8014W

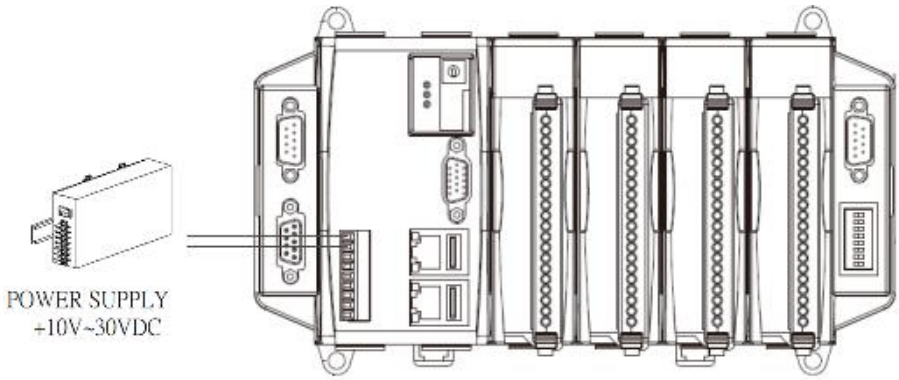
<table>
<tr><th colspan="4">I-8014W</th></tr>
<tr><th>Input Type</th><th>Differential</th><th>Input Type</th><th>Single-ended</th></tr>
<tr><td>Voltage Input Wiring</td><td> 
</td><td>Voltage Input Wiring</td><td> 
</td></tr>
<tr><td>Current Input Wiring</td><td> 
</td><td>Current Input Wiring</td><td> 
</td></tr>
</table>

<table>
<tr><th colspan="2">I-8014CW</th></tr>
<tr><th>Input Type</th><th>Differential</th></tr>
<tr><td>Current Input Wiring</td><td> 
</td></tr>
</table>

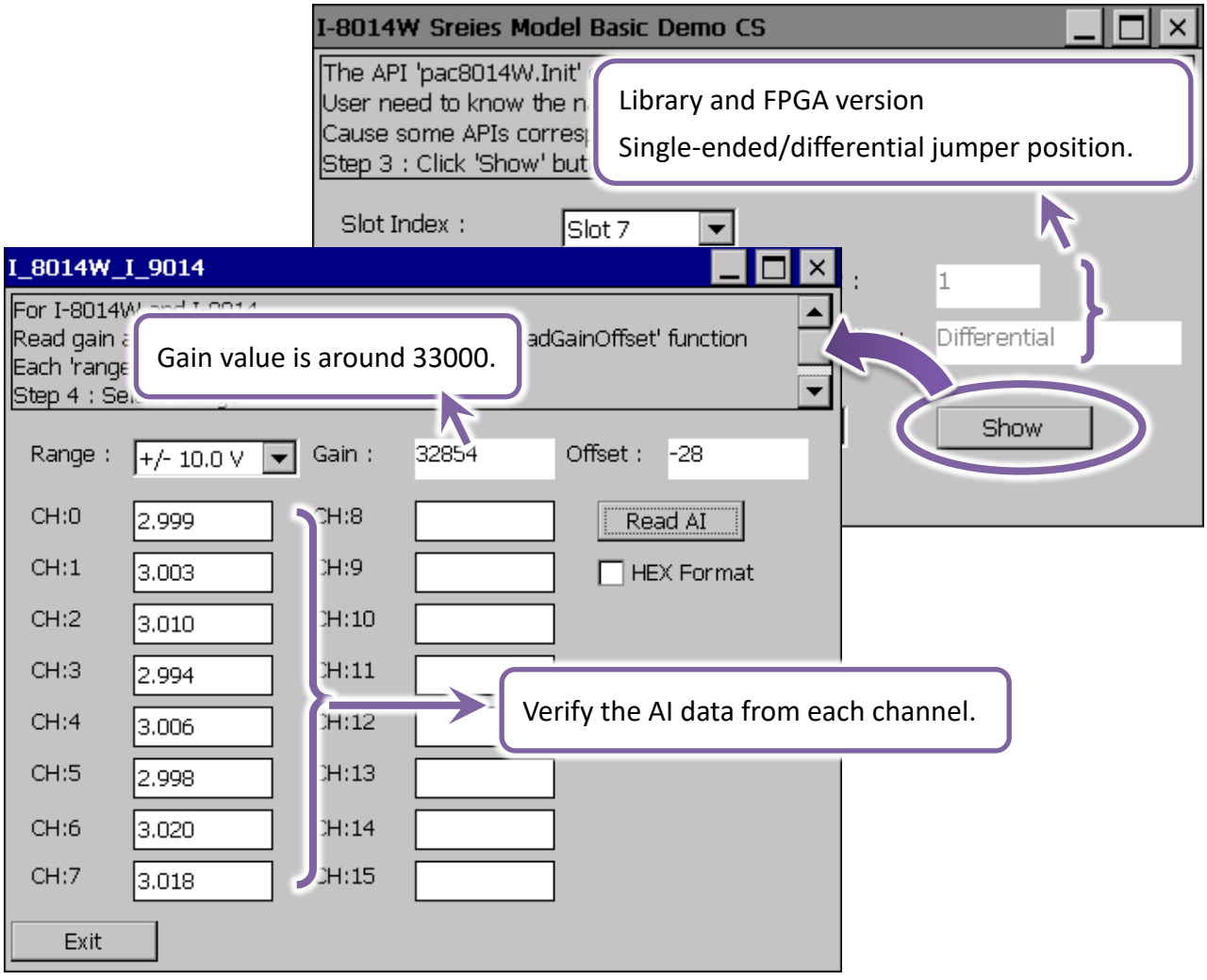
Tips & Warnings

Unused channels should be connected to GND to avoid floating.

Step 2. Plug I-8014W in the Windows-based controller and connect the power supply to the unit



Step 3. Run pac_i8014W_ReadAI_CSharp.exe on the controller and verify that basic information and AI data read from each channel are correct, as shown in the figure below:



Tips & Warnings

I-8014CW/I-9014C only can select max 8 channels and +/- 20 mA Input Range

2.2.2. Magic Scan Function

The magic scan function is used for high-speed sampling of data.

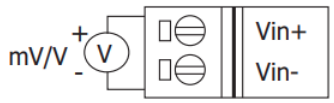
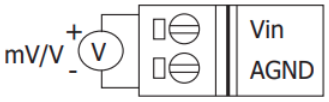
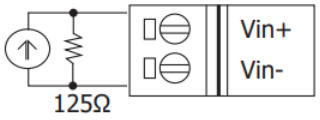
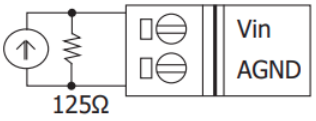
This is the major function of I-8014W series module.

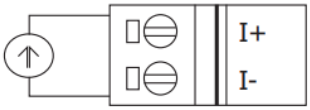
Configuration of Magic Scan function includes:

- Sample Channels (1 for channel 0, 2 for channel 0 and channel 2 and so on)
- Input Range
- Sample rate (Max 200 KHz, depend on scan mode and Sample channels)
- Sample count

The following steps take the pac_i8014W_MagicScan_Block_CSharp.exe as example and display the process of data acquisition.

Step 1: Please refer to the "Wire Connections", connect a cyclically source (such as a Sine wave) to I-8014W

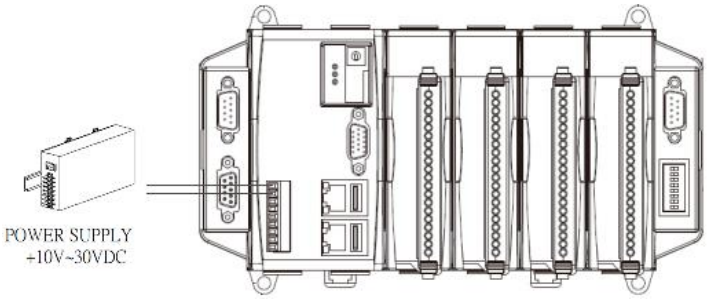
I-8014W			
Input Type	**Differential**	**Input Type**	**Single-ended**
Voltage Input Wiring		Voltage Input Wiring	
Current Input Wiring		Current Input Wiring	

I-8014CW	
Input Type	**Differential**
Current Input Wiring	

Tips & Warnings

Unused channels should be connected to GND to avoid floating.

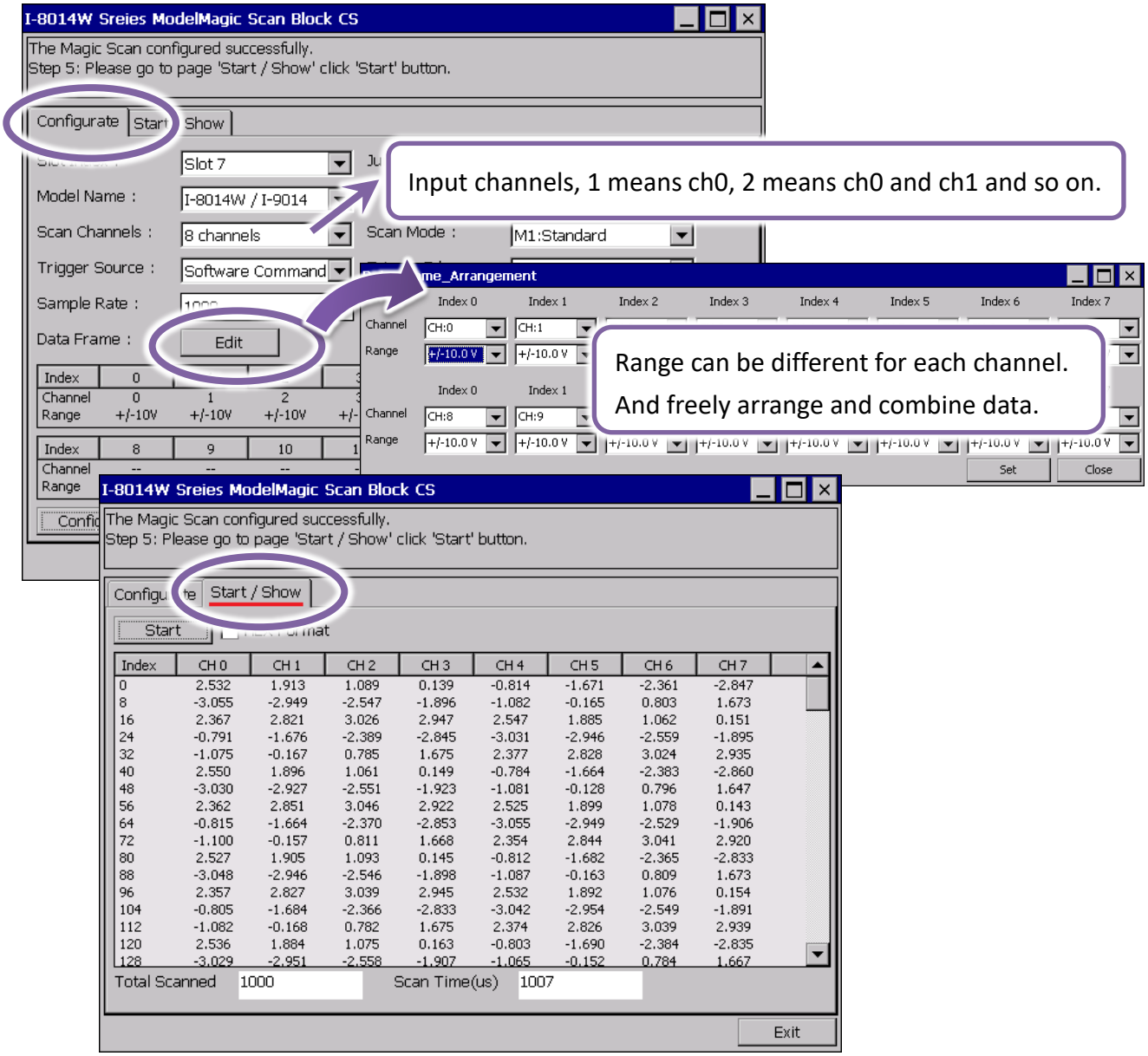
Step 2. Plug I-8014W in the Windows-based controller and connect the power supply to the unit



Step 3. Run pac_i8014W_MagicScan_Block_CSharp.exe on the controller and input parameters, as indicated in the diagram below:

In this demonstration, there is a 50 Hz, 3 Vpp, Sine wave connect with channel 0 to channel 7.

For more detail about the attributes of Magic Scan, please refer to 3. Magic Scan.



2.2.3. Average Function

The Average function is used for calculate data average with Magic Scan function.

After this function is activated, it will continue to calculate in the bottom background, and users can read back the data by Polling.

Tips & Warnings

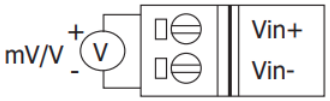
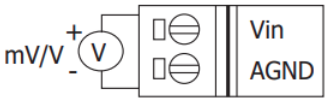
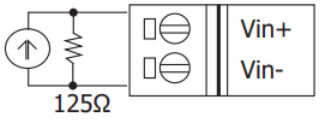
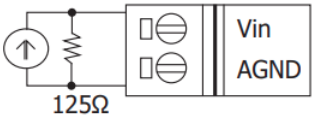
This function can only use on WinCE6.0, WinCE7.0 PACs.

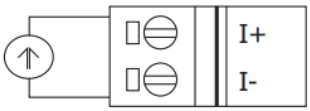
Configuration of Average function includes:

- Sample Channels (always 8 channels in Differential or 16 channels in Single-Ended)
- Input Range (every channels are the same range)
- Sample rate (Max 5 KHz)
- Average level (Calculate the average value of every 256 data at most)

The following steps take the pac_i8014W_ReadAI_Average_AllSlot_CSharp.exe as example and display the process of data acquisition.

Step 1: Please refer to the "Wire Connections", connect a cyclically source (such as a Sine wave) to I-8014W

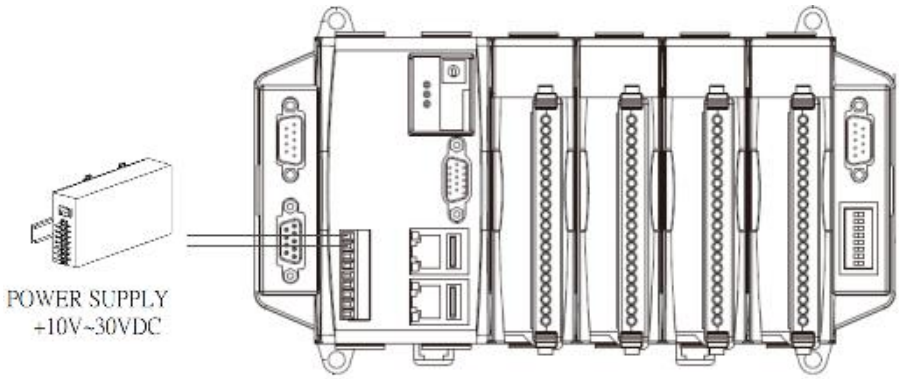
I-8014W			
Input Type	**Differential**	**Input Type**	**Single-ended**
Voltage Input Wiring		Voltage Input Wiring	
Current Input Wiring		Current Input Wiring	

I-8014CW	
Input Type	**Differential**
Current Input Wiring	

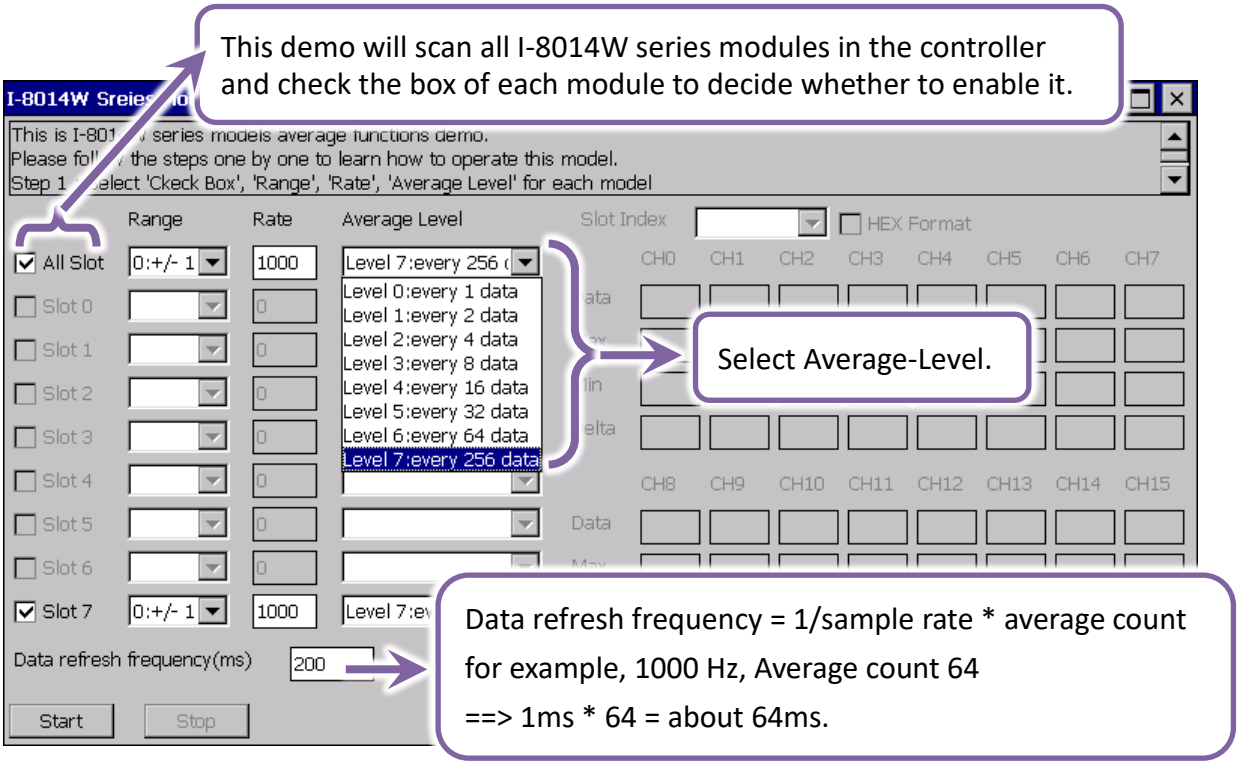
Tips & Warnings

Unused channels should be connected to GND to avoid floating.

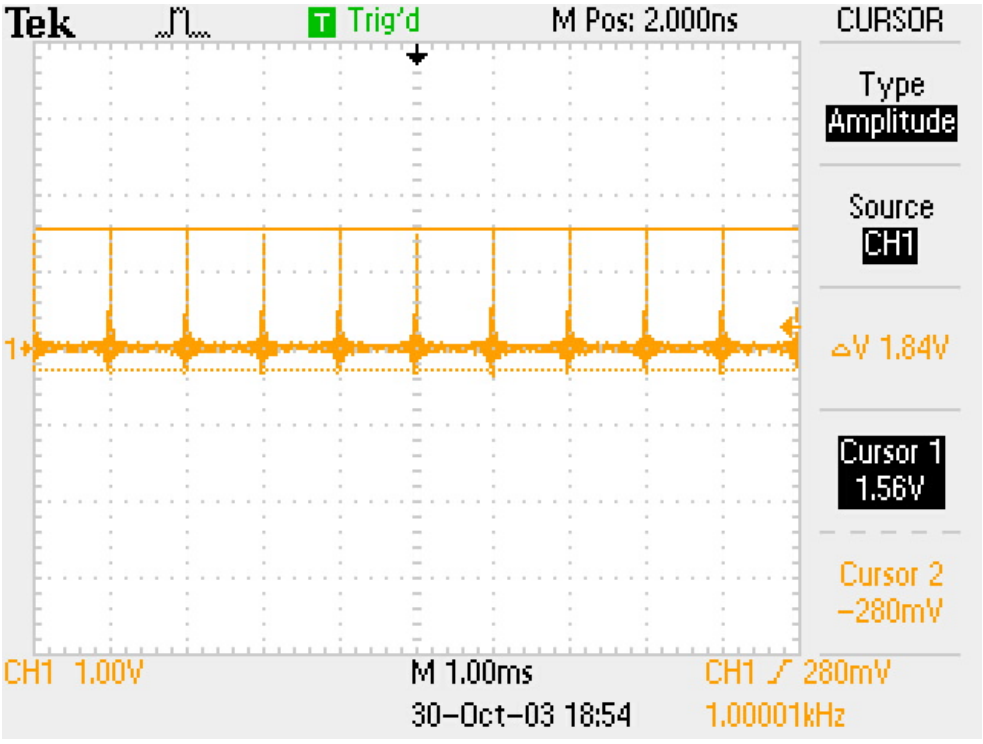
Step 2. Plug I-8014W in the Windows-based controller and connect the power supply to the unit



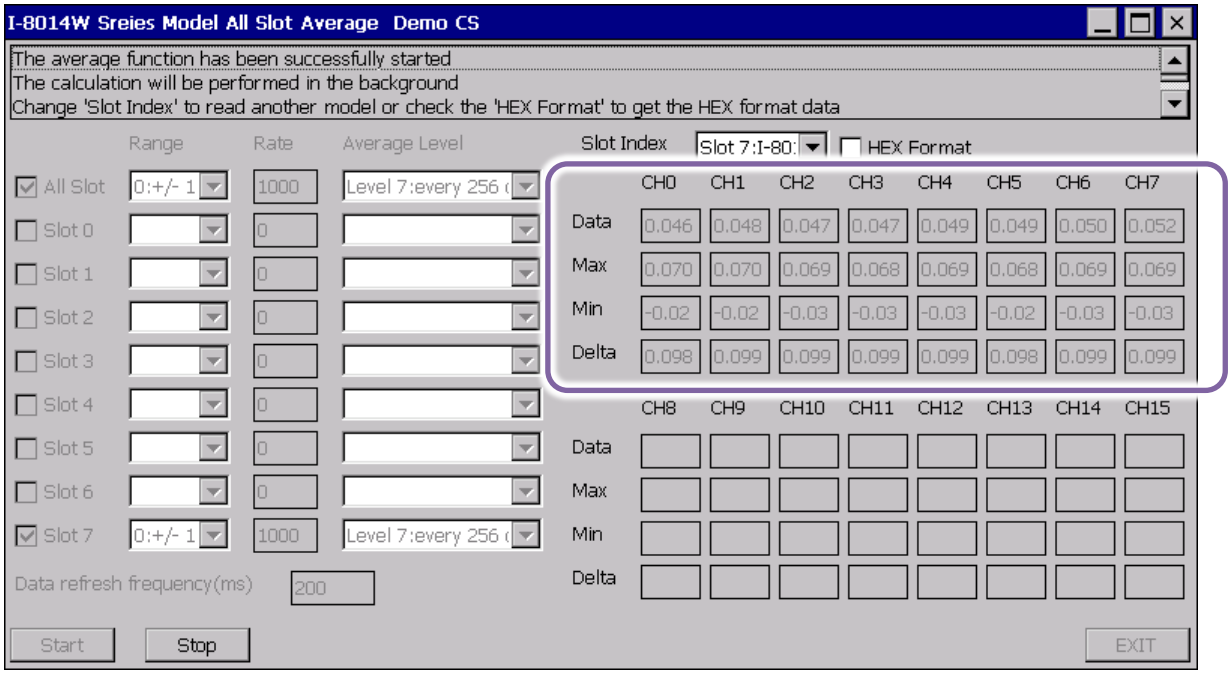
Step 3. Run pac_i8014W_ReadAI_Average_AllSlot_CSharp.exe on the controller and input parameters, as indicated in the diagram below:



In this demo, there is a 50 Hz, 1.5 Vpp, noise connect with channel 0 to channel 7, shown as below picture:

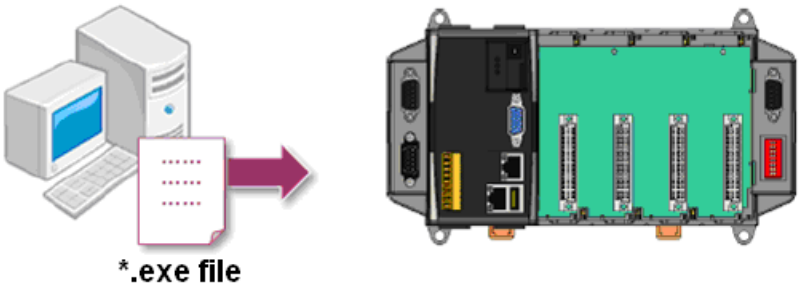


The results are shown in the below picture:



2.3. Linux-based Controllers

ICP DAS provides a range of demo programs for different platforms that can be used to verify the functions of the I-8014W/9014. 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-8014W/9014 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-8x4x	PXA270	https://www.icpdas.com/en/download/show.php?num=982
LP-8x2x/9000	AM335x	https://www.icpdas.com/en/download/show.php?num=915
LX-8000/9000	x86/E38xx	http://www.icpdas.com/en/download/show.php?num=904

3. Magic Scan

This chapter provides details related to Magic Scan, which is a key function included on the I-8014W series modules for multi-channel analog data acquisition at high sampling rates.

The attributes and operation steps of Magic Scan included:

I. Configuration

- Channels
- Data arrangement
- Sample Rate
- Scan Mode
  - Standard
  - Sample and Hold
- Trigger Source
  - Software Trigger
  - Interrupt Trigger
  - External Trigger
    - Rising edge
    - Falling edge

II. Startup

III. Read Raw Data

- Block Mode
- Non-Block Mode
- Interrupt Mode (Need install ISR function)

IV. Stop

V. Calibrate raw data

3.1. Configuration

Before start sampling data, user needs to configure Magic Scan settings, the settings include as follow:

- **Channels:**

  Set the number of channels to sample data.

- **Data arrangement:**

  Data arrangement allows users to arrange the data that store into FIFO.

- **Sample Rate:**

  Set the data sampling frequency, I-8014W uses independent internal hardware clock to trigger AD.

  The actual frequency might be different from the setting.

- **4K FIFO with interrupt to reduce CPU loading:**

  There is a FIFO for interrupt service notification.

  This feature can increase the performance for sampling application.

  The program does not need to sample data all the time.

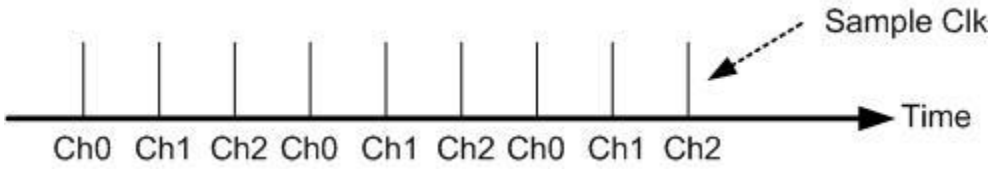
  Only need to wait for the interrupt notification when the data count reaches the limit level

- **Scan Mode:**

There are two types of scan mode, one is a standard and another is virtual sample and hold.

1. **Standard:**

Every clock samples only one channel, such as below:



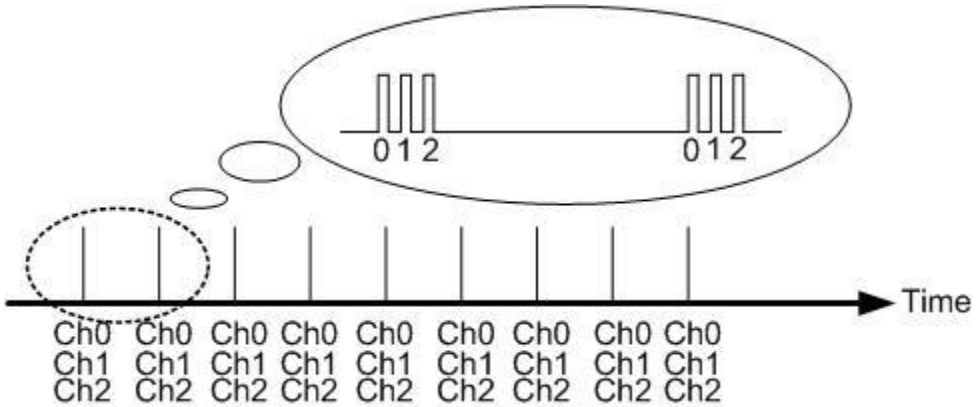
While set mode as Standard,

Sample rate can be 250 KHz with one channel or 125 KHz with 2 channels and so on.

Channels	Max Rate
1	250KHz
2	125KHz
3	83.3KHz
4	62.5KHz

2. **Sample and Hold:**

Every clock samples all channels that have been set.



While set mode as Sample and Hold,

Sample rate can be 125 KHz with one channel or 62.5 KHz with 2 channels and so on.

Channels	Max Rate	Total Rate
1	125KHz	125KHz
2	62.5KHz	125KHz
4	31.25 KHz	125 KHz

- **Trigger Source:**

  There are three types of trigger source, software trigger, interrupt trigger and external trigger.

  This setting decides the way of starting sample data.

  1. **Software Trigger:**

     When set as softest trigger, Magic Scan will be start immediately when the application calls startup API.

     If there are two or more modules, users need to startup Magic Scan for each module.

  2. **Interrupt trigger:**

     When set as interrupt trigger, users need to install an ISR function to read data before calls startup API.

  3. **External trigger:**

     When set as external trigger, users need to output a pulse to modules after call startup API.

     This trigger can be set as either rising edge or falling edge.

     Modules will wait until they receive the external signal from the Trig+ and Trig- pins and then startup Magic Scan.

     This method is very helpful for activate two or more modules in the same time.

3.2. Startup

After configure settings, call the startup API will be fine.

In the sampling process, data will be saved in the FIFO automatically, users need to read back data to avoid FIFO overflow.

Depending on the application and trigger source, users need to read raw data during the sampling or read all raw data at the end of acquisition.

3.3. Read Raw Data

There are three ways to read raw data.

- **Block Mode:**

  After startup Magic Scan, call this API will be fine.

  This mode is easier to develop application.

  But, this mode will block the program until the sampling finished.

- **Non-Block Mode:**

  This mode requires continuous polling to get data from FIFO to avoid overflow.

  It has better processing efficiency of CPU resources.

- **Interrupt Mode:**

  This Mode requires users to install an ISR function before startup Magic Scan, and get data from FIFO in the ISR function.

  Whenever the data in FIFO reaches a certain amount, an interrupt signal will be output to notify the CPU to read data from FIFO.

  This mode requires less CPU loading.

3.4. Stop

After the acquisition is complete, users need to stop Magic Scan, in order to restore the status of module and driver, such as clear FIFO, velar buffer.

Tips & Warnings

Users should read all the data before stop Magic Scan; otherwise the data will be cleared.

3.5. Calibrate Raw Data

After acquire data, those data need to be calibrated to be the actual physical quantity.

Bring the data and few parameter into the calibrate APIs will be fine.

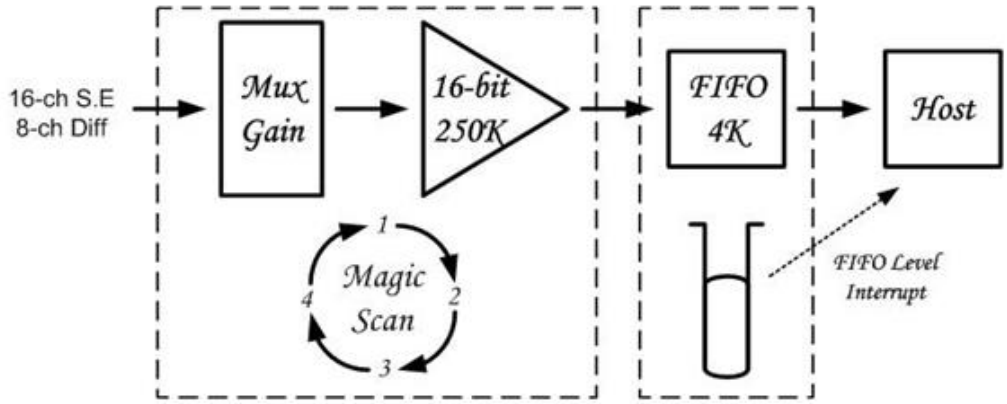
Tips & Warnings

I-8014W equipped with a 4 k-sample FIFO buffer which can store 4096 data samples from Magic Scan to ensure that no data is lost.

The acquisition data is sequentially saved to the FIFO buffer during the scan process. To prevent the FIFO buffer from being filled, the data needs to be read from the FIFO buffer within a specific timeframe.

If the FIFO buffer is filled, data can no longer be saved until a command is executed that clears the FIFO buffer. In contrast, if data is read from the FIFO buffer too frequently, CPU resources will be wasted and performance will be affected.

To achieve the optimum balance, two modes for transferring data from the FIFO are provided, polling mode and interrupt mode.



4. API introduction

ICPDAS supplies a range of C/C++ API functions for the I-8014(C)W/I-9014(C) module.

When developing a program, refer to either the 8014W.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.NET programs, these functions are ported from the relevant C/C++ functions.

Download link: https://www.icpdas.com/en/download/show.php?num=2897

API Naming Table

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

<table>
<tr><th rowspan="2">Platform</th><th rowspan="2">Product included</th><th colspan="2">API prefix characters</th></tr>
<tr><th>C / C++</th><th>C#</th></tr>
<tr><td>Windows CE5</td><td>I-8014W /I-8014CW</td><td rowspan="5">“pac_i8014W_“
+ function name</td><td rowspan="5">“pac8014Wnet.pac8014W.”
+ function name</td></tr>
<tr><td>Windows CE6</td><td>I-8014W /I-8014CW</td></tr>
<tr><td>Windows CE7</td><td>I-8014W /I-8014CW
I-9014 / I-9014C</td></tr>
<tr><td>WES</td><td>I-8014W /I-8014CW</td></tr>
<tr><td>WES7</td><td>I-8014W /I-8014CW
I-9014 / I-9014C</td></tr>
<tr><td>MiniOS7</td><td>I-8014W /I-8014CW</td><td>“i8014W_”
+ function name</td><td rowspan="3">Null</td></tr>
<tr><td rowspan="2">Linux</td><td>I-8014W /I-8014CW</td><td>“I8014W_”
+ function name</td></tr>
<tr><td>I-9014 / I-9014C</td><td>“I9014_”
+ function name</td></tr>
</table>

The following is an overview of the functions provided for I-8014W series modules.

Detailed information related to individual functions can be found in the following sections.

API for I-8014(C)W and I-9014(C)

API	Description
i8014W_Init pac_i8014W_Init	This function is used to initialize the driver and modules, it must be call once for each module at the beginning of program.
i8014W_GetFirmwareVer_L1 pac_i8014W_GetFirmwareVer_L1	This function is used to read the version number of the first FPGA.
i8014W_GetFirmwareVer_L2 pac_i8014W_GetFirmwareVer_L2	This function is used to read the version number of the secondary FPGA.
i8014W_GetLibVersion pac_i8014W_GetLibVersion	This function is used to read the version number of the 8014W.lib.
i8014W_GetLibDate pac_i8014W_GetLibDate	This function is used to read the release date of the 8014W.lib.
i8014W_GetSingleEndJumper pac_i8014W_GetSingleEndJumper	This function is used to read the single-ended/differential jumper position on the module.
i8014W_ReadAI pac_i8014W_ReadAI	This function is used to read the input from a specific channel in float format.
i8014W_ReadAIHex pac_i8014W_ReadAIHex	This function is used to read the input from a specific channel in hexadecimal format.
i8014W_ConfigMagicScan pac_i8014W_ConfigMagicScan	This function is used to configure all the parameters when using Magic Scan, and it should be called before any Magic Scan instructions.
i8014W_StartMagicScan pac_i8014W_StartMagicScan	This function is used to start Magic Scan.

API	Description
i8014W_StopMagicScan pac_i8014W_StopMagicScan	This function is used to stop Magic Scan.
i8014W_ReadFIFO pac_i8014W_ReadFIFO	This function is used to read data from the FIFO buffer after the Magic Scan function has been triggered.
i8014W_ReadFIFO_BlockMode pac_i8014W_ReadFIFO_BlockMode	This function is used to read data from the FIFO buffer in block mode after the magic scan is started.
i8014W_ReadFIFO_InISR pac_i8014W_ReadFIFO_InISR	This function is used to read data from the FIFO buffer after the ISR function has been installed and the magic scan is started.
i8014W_UnLockFIFO pac_i8014W_UnLockFIFO	This function is used to unlock the FIFO buffer when it is locked after being filled.
i8014W_ClearFIFO pac_i8014W_ClearFIFO	This function is used to clear the FIFO buffer after the Unlock FIFO function has been executed.
i8014W_InstallMagicScanISR pac_i8014W_InstallMagicScanISR	This function is used to install the ISR to control interrupt events form the FIFO buffer.
i8014W_UnInstallMagicScanISR pac_i8014W_UnInstallMagicScanISR	This function is used to uninstall the Magic Scan ISR.
i8014W_ClearInt pac_i8014W_ClearInt	This function is used to clear the status of the Magic Scan interrupts.

API for I-8014W and I-9014

API	Description
i8014W_ReadGainOffset pac_i8014W_ReadGainOffset	This function is used to obtain the gain and offset values on each input type for I-8014W/I-9014.
i8014W_CalibrateData pac_i8014W_CalibrateData	This function is used to calibrate the Magic Scan raw data and convert the data to a float format.
i8014W_CalibrateDataHex pac_i8014W_CalibrateDataHex	This function is used to calibrate the Magic Scan raw data and convert the data to a hexadecimal format.

API for I-8014CW and I-9014C

API	Description
i8014W_Read_mA_GainOffset pac_i8014W_Read_mA_GainOffset	This function is used to obtain the gain and offset values on each input type for I-8014CW/I-9014C.
i8014W_Calibrate_CH_mA pac_i8014W_Calibrate_CH_mA	This function is used to calibrate the Magic Scan raw data and convert the data to a float format.
i8014W_Calibrate_CH_mA_Hex pac_i8014W_Calibrate_CH_mA_Hex	This function is used to calibrate the Magic Scan raw data and convert the data to a hexadecimal format.

API for I-8014(C)W and I-9014(C) on only CE6.0, CE7.0 and Linux platform

API	Description
pac_i8014W_Start_Average_INT	This function is used to startup calculate data average with Magic Scan function, the "pac_i8014W_Init" will also be called.
pac_i8014W_Stop_Average_INT	This function is used to stop the Average Function.
pac_i8014W_ReadAIHex_Average_INT	This function is used to read one calibrated HEX format AI data
pac_i8014W_ReadAI_Average_INT	This function is used to read one calibrated float format AI data
pac_i8014W_ReadAllAIHex_Average_INT	This function is used to read multiple calibrated HEX format AI data
pac_i8014W_ReadAllAI_Average_INT	This function is used to read multiple calibrated float format AI data

4.1. i8014W_Init / pac_i8014W_Init

This function is used to initialize the driver and module.

Syntax

For MiniOS7

```
short i8014W_Init(int slot);
```

For Windows (CE and WES)

```
short pac_i8014W_Init(int slot);
```

For Linux

```
short i8014W_Init(int slot);          // for LinPAC-8000
short i9014_Init(int slot);           // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

Return Values

0 = initialized I-8014W series module successfully.

-1 = No I-8014W series module in this slot.

Note

This function must be called once for each I-8014W series module.

If there are two or more modules, user needs call i8014W_Init function for each module individually by passing the slot number where the I-8014W/I-9014 module is plugged into.

Example

[C/C++]

```
int slot=0;
if(i8014W_Init(slot)==0){    // initialized successfully.
}
```

[C#]

```
int slot=0;
if(pac8014Wnet.pac8014W.Init(slot)==0) {    // initialized successfully.
}
```

[C] (for LinPAC)

```
int main(){
    int slotIndex, err, ret;
    ret=Open_Slot(slotIndex);
    if (ret > 0) {
            printf("Open Slot%d failed, return value=%d \n", slotIndex, ret);
            return (-1);
    }
    err=i8014W_Init(slotIndex);
    if(err==0)
    printf("There is an I-8014W module in slot %d\n", slotIndex);
    else
    printf("There is no I-8014W module in slot %d\n", slotIndex);
    Close_Slot();
    return 0;
}
```

4.2. i8014W_GetFirmwareVer_L1 / pac_i8014W_GetFirmwareVer_L1

This function is used to retrieve the version number of the primary FPGA firmware and troubleshooting or recording purposes.

Syntax

For MiniOS7

```
short i8014W_GetFirmwareVer_L1(int slot);
```

For Windows (CE and WES)

```
short pac_i8014W_GetFirmwareVer_L1(int slot);
```

For Linux

```
short i8014W_GetFirmwareVer_L1(int slot);          // for LinPAC-8000
short i9014_GetFirmwareVer_L1(int slot);           // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

Return Values

The version number of the first FPGA.

Example

[C/C++]

```
short ver=0, slot=0;
ver= i8014W_GetFirmwareVer_L1(slot);
```

[C#]

```
Int slot=0, ver=0;
ver =pac8014Wnet.pac8014W.FirmwareVer_L1(slot);
```

[C] (for LinPAC)

```
int main(){
    int ver=0, slot=1;
    Open_Slot(slot);
    i8014W_Init(slot);      // initialize
    ver= i8014W_GetFirmwareVer_L1(slot);
    Close_Slot();
    return 0;
}
```

4.3. i8014W_GetFirmwareVer_L2 / pac_i8014W_GetFirmwareVer_L2

This function is used to retrieve the version number of the secondary FPGA firmware and troubleshooting or recording purposes.

Syntax

For MiniOS7

```
short i8014W_GetFirmwareVer_L2(int slot);
```

For Windows (CE and WES)

```
short pac_i8014W_GetFirmwareVer_L2(int slot);
```

For Linux

```
short i8014W_GetFirmwareVer_L2(int slot);          // for LinPAC-8000
short i9014_GetFirmwareVer_L2(int slot);           // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

Return Values

The version number of the secondary FPGA

Example

[C/C++]

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

ver= i8014W_GetFirmwareVer_L2(slot);
```

[C#]

```
Int slot, ver;

ver =pac8014Wnet.pac8014W.FirmwareVer_L2(slot);
```

[C] (for LinPAC)

```
int main(){
    int ver=0, slot=1;

    Open_Slot(slot);

    i8014W_Init(slot);       // initialize

    ver= i8014W_GetFirmwareVer_L2(slot);
    Close_Slot();
    return 0;
}
```

4.4. i8014W_GetLibVersion / pac_i8014W_GetLibVersion

This function is used to retrieve the version number of library and troubleshooting or recording purposes.

Syntax

For MiniOS7

```
short i8014W_GetLibVersion(void);
```

For Windows (CE and WES)

```
short pac_i8014W_GetLibVersion(void);
```

For Linux

```
short i8014W_GetLibVersion(void);          // for LinPAC-8000
short i9014_GetLibVersion(void);           // for LinPAC-9000, LX-9000
```

Parameter

None

Return Values

The version number of library.

Example

[C/C++]

```
short version;

version = i8014W_GetLibVersion();
```

[C#]

```
Int version;

version = pac8014Wnet.pac8014W.LibVersion();
```

[C] (for LinPAC)

```
int main(){
    int slot=1, version;

    Open_Slot(slot);

    i8014W_Init(slot);          // initialize

    version = i8014W_GetLibVersion();

    Close_Slot();
    return 0;
}
```

4.5. i8014W_GetLibDate / pac_i8014W_GetLibDate

This function is used to retrieve the release date of library and troubleshooting or recording purposes.

Syntax

For MiniOS7

```
void i8014W_GetLibDate(char *LibDate);
```

For Windows (CE and WES)

```
void pac_i8014W_GetLibDate(char libDate[]);
```

Parameter

**libDate:*

[Output] the release date of library.

Return Values

None

Example

[C/C++]

```
char libDate [32];
i8014W_GetLibDate(libDate);
```

[C#]

```
string version;
version = pac8014Wnet.pac8014W.LibDate();
```

4.6. i8014W_GetSingleEndJumper / pac_i8014W_GetSingleEndJumper

This function is used to retrieve the single-ended/differential jumper position of the module.

Syntax

For MiniOS7

```
short i8014W_GetSingleEndJumper(int slot);
```

For Windows (CE and WES)

```
short pac_i8014W_GetSingleEndJumper(int slot);
```

For Linux

```
short i8014W_GetSingleEndJumper(int slot);          // for LinPAC-8000
short i9014_GetSingleEndJumper(int slot);           // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

Return Values

0: The jumper is in the differential position.

1: The jumper is in the single-ended position.

Example

[C/C++]

```
short jumper,slot;
jumper = i8014W_GetSingleEndJumper(slot);
```

[C#]

```
short jumper,slot;
jumper = pac8014Wnet.pac8014W.SingleEndJumper(slot);
```

[C/C++]

```
int main(){
    int slot=1, jumper=0, maxCh=0;
    Open_Slot(slot);
    i8014W_Init(slot);      // initialize
    jumper = i8014W_GetSingleEndJumper(slot);
    Close_Slot();
    return 0;
}
```

4.7. i8014W_ReadAI / pac_i8014W_ReadAI

This function is used to read a floating point input (calibrated) from one specified channel.

Syntax

For MiniOS7

```
short i8014W_ReadAI(int slot, int ch, int gain, float* fVal);
```

For Windows (CE and WES)

```
short pac_i8014W_ReadAI(int slot, short ch, short gain, float* fVal);
```

For Linux

```
short i8014W_ReadAI(int slot, int ch, int gain, float* fVal);      // for LinPAC-8000
short i9014_ReadAI(int slot, int ch, int gain, float* fVal);       // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

Ch:

specifies the channel number, 0 – 7 for differential input, or 0 – 15 for single-ended input.

Gain:

specifies the input type (0 – 4), where:0: +/-10 V, 1: +/-5 V, 2: +/-2.5 V, 3: +/-1.25 V, 4: +/-20 mA

**fVal:*

[Output] the floating-point data.

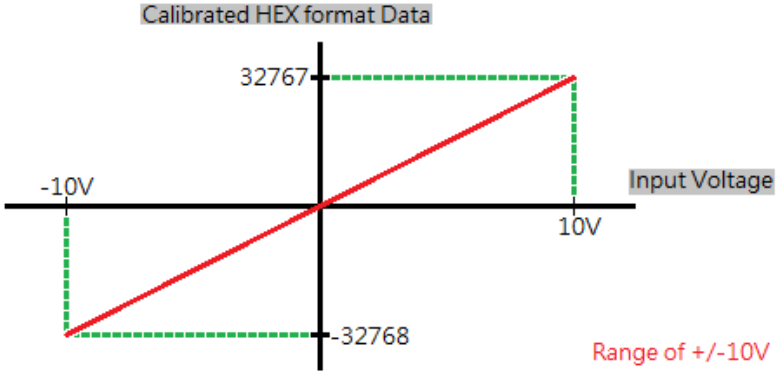
Return Values

0 = No Error

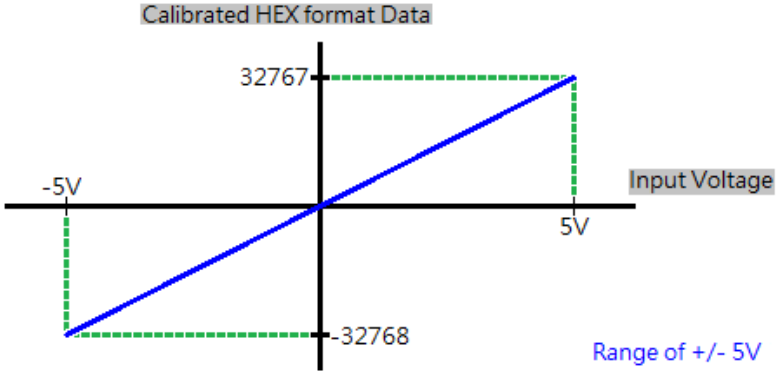
Note

I-8014CW/I-9014C only can select max 8 channels and +/- 20 mA Input Range.

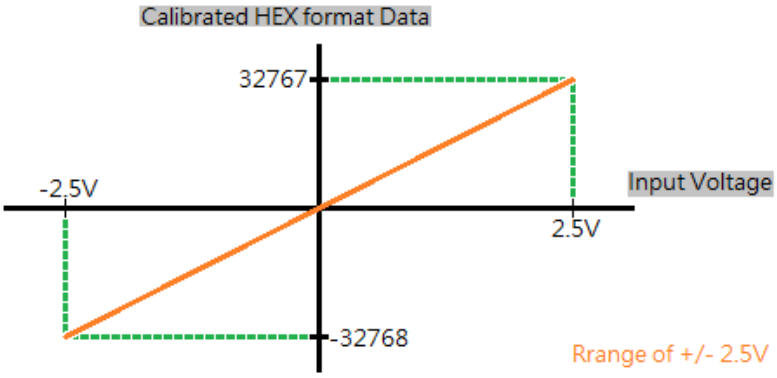
The following pictures show the scale of voltage and data and how to calculate the hexadecimal data into floating data.



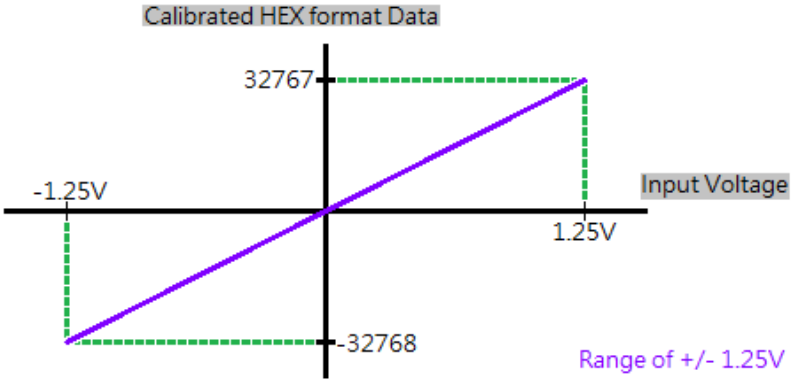
If(hexadecimal data >=0)	**floating data = (hexadecimal data / 32767) * 10**
Else	**floating data = (hexadecimal data / 32768) * 10**



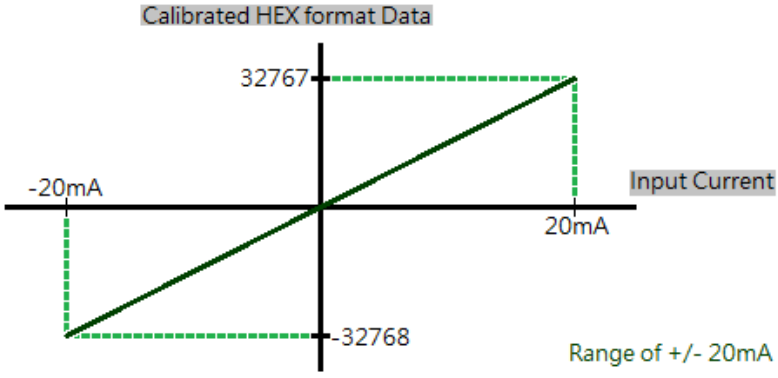
If(hexadecimal data >=0)	**floating data = (hexadecimal data / 32767) * 5**
Else	**floating data = (hexadecimal data / 32768) * 5**



If(hexadecimal data >=0)	floating data = (hexadecimal data / 32767) * 2.5
Else	floating data = (hexadecimal data / 32768) * 2.5



If(hexadecimal data >=0)	floating data = (hexadecimal data / 32767) * 1.25
Else	floating data = (hexadecimal data / 32768) * 1.25



If(hexadecimal data >=0)	floating data = (hexadecimal data / 32767) * 20
Else	floating data = (hexadecimal data / 32768) * 20

Example

[C/C++]

```
int slot, ch, gain;
float fVal=0.0;
i8014W_ReadAI( slot, ch, gain, &fVal);
```

[C#]

```
int slot, ch, gain;
float fVal=0.0;
pac8014Wnet.pac8014W.ReadAI(slot, ch, gain, ref fVal);
```

[C] (for LinPAC)

```
int main(){
int slot=1, ch, gain, ret;
float fVal=0.0;
gain = 0;      // "+/-10V"
ret=Open_Slot(slot);
if (ret > 0) {
        printf("Open Slot%d failed, return value=%d \n", slot, ret);
        return (-1);
}
i8014W_Init(slot);          // initialize
for(ch=0;ch<8;ch++)    {
   i8014W_ReadAI( slot, ch, gain, &fVal);
   printf("\n[%02d]= [ %05.4f ]", ch, fVal);
}
Close_Slot(slot);
return 0;
}
```

4.8. i8014W_ReadAIHex / pac_i8014W_ReadAIHex

This function is used to read a hexadecimal input (calibrated) from one specified channel.

Syntax

For MiniOS7

```
short i8014W_ReadAIHex(int slot, int ch, int gain, short* hVal);
```

For Windows (CE and WES)

```
short pac_i8014W_ReadAIHex(int slot, short ch, short gain, short* hVal);
```

For Linux

```
short i8014W_ReadAIHex(int slot, int ch, int gain, short* hVal);// for LinPAC-8000
short i9014_ReadAIHex(int slot, int ch, int gain, short* hVal);  // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

Ch:

specifies the channel number, 0 – 7 for differential input, or 0 – 15 for single-ended input.

Gain:

specifies the input type (0 – 4), where:0: +/-10 V, 1: +/-5 V, 2: +/-2.5 V, 3: +/-1.25 V, 4: +/-20 mA

**hVal:*

[Output] the hexadecimal data.

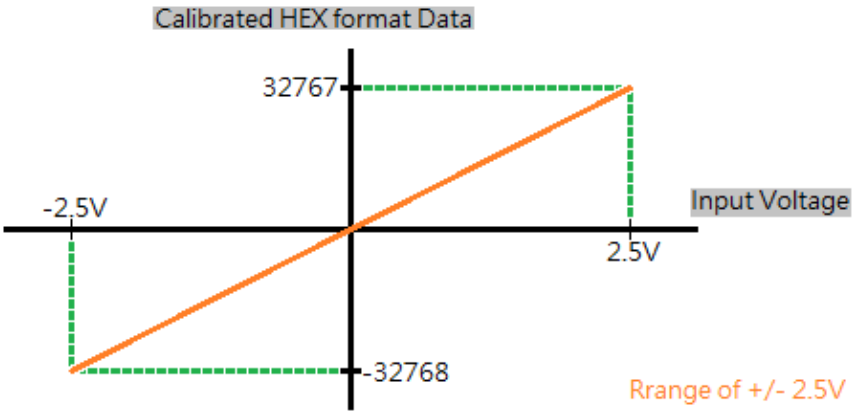
Return Values

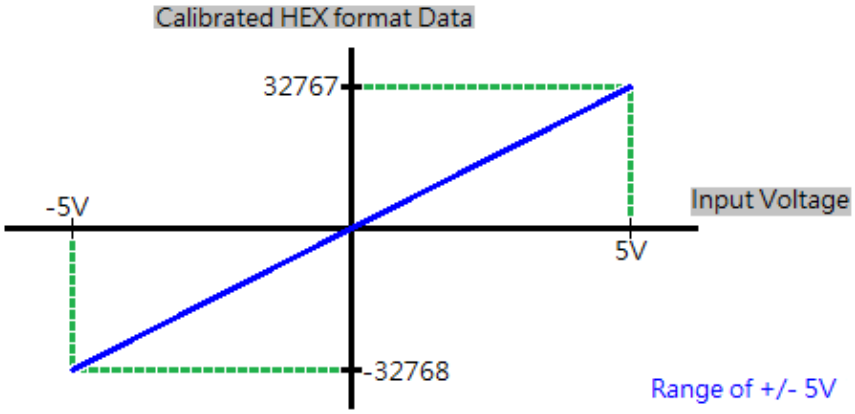
0 = No Error

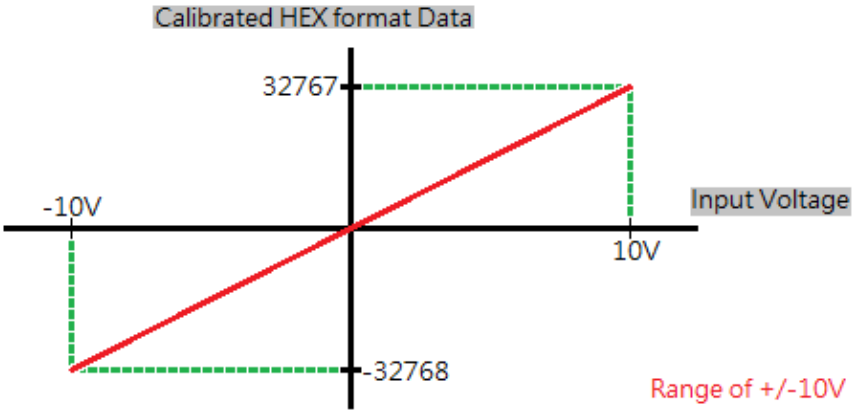
Note

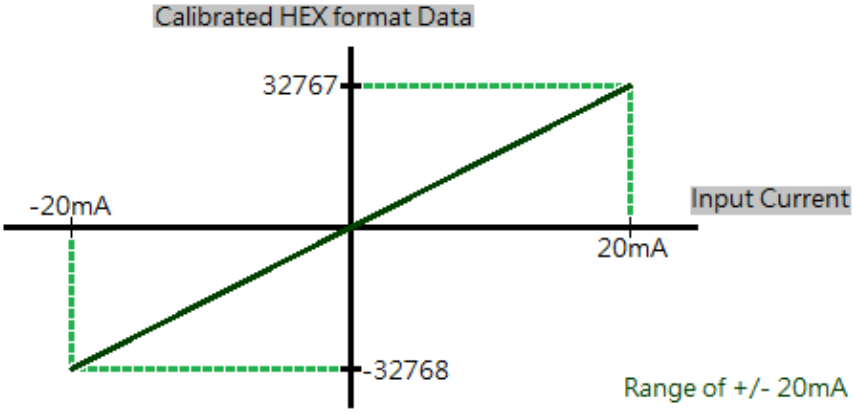
I-8014CW/I-9014C only can select max 8 channels and +/- 20 mA Input Range.

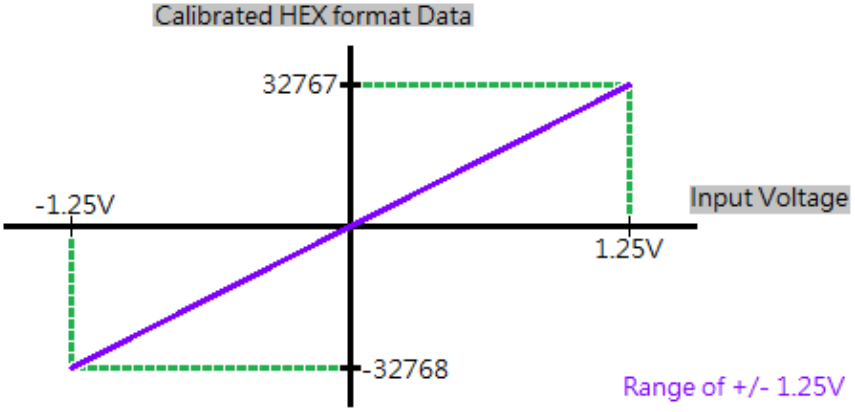
The following pictures show the scale of voltage and data.








Calibrated HEX format Data
32767
-20mA
Input Current
20mA
-32768
Range of +/- 20mA


Calibrated HEX format Data
32767
-1.25V
Input Voltage
1.25V
-32768
Range of +/- 1.25V

Example

[C/C++]

```
int slot, ch, gain;
short hVal=0.0;
i8014W_ReadAIHex(slot, ch, gain, &hVal);
```

[C#]

```
int slot, ch, gain;
int hVal=0.0;
pac8014Wnet.pac8014W.ReadAIHex(slot, ch, gain, ref hVal);
```

[C] (for LinPAC)

```
int slot, ch, gain;
short hVal=0.0;
Open_Slot(slot);
i8014W_Init(slot);          // initialize
i8014W_ReadAIHex(slot, ch, gain, &hVal);
```

4.9. i8014W_ConfigMagicScan / pac_i8014W_ConfigMagicScan

This function is used to configure all the parameters when using Magic Scan, and should be called before executing any Magic Scan instructions.

Syntax

For MiniOS7

```
void i8014W_ConfigMagicScan(
    int slot, int chArr[], int gainArr[], int scanChCount, float sampleRate, int scanMode,
    int triggerSource, int triggerState , float* realSampleRate
);
```

For Windows (CE and WES)

```
void pac_i8014W_ConfigMagicScan(
    int slot, short chArr[], short gainArr[], short scanChCount, float sampleRate, short
scanMode,
    short triggerSource, short triggerState, float* realSampleRate
);
```

For Linux

```
void i8014W_ConfigMagicScan(
    int slot, int chArr[], int gainArr[], int scanChCount, float sampleRate, int scanMode,
    int triggerSource, int triggerState , float* realSampleRate
);                                                          // for LinPAC-8000

void i9014_ConfigMagicScan(
    int slot, int chArr[], int gainArr[], int scanChCount, float sampleRate, int scanMode,
    int triggerSource, int triggerState , float* realSampleRate
);                                                          // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

chArr[]:

Create an array that used to set the channels to be scanned. The channel indices define the scanning order; the maximum number of channels is 16.

gainArr[]:

Create an array that used to set the input type for the corresponding channel with the same index as stored in chArr[], where: **0**: +/-10 V, **1**: +/-5 V, **2**: +/-2.5 V, **3**: +/-1.25 V, **4**: +/-20 mA

scanChCount:

Number of the channels, that have been added to chArr[].

sampleRate:

The total sampling rate, 2 – 250 kHz.

scanMode:

1: Standard mode

2: Virtual Sample and Hold mode

triggerSource:

0: Software trigger

1: Internal hardware trigger

2: External hardware trigger

triggerState:

0: Rising edge trigger. This is only valid when using an external hardware trigger.

1: Falling edge trigger. This is only valid when using an external hardware trigger.

**realSampleRate:*

[Output] the real sampling rate that was used by the I-8014W.

Return Values

None

Note

I-8014CW/I-9014C only can select max 8 channels and +/- 20 mA Input Range

Example

[C/C++]

```
int slot, chArr[16], gainArr[16], scanChCount;
float sampleRate,realsampleRate;
int scanMode, triggerSource, triggerState;
chArr[0]=0;              // element 0 assigned to channel 0
...
chArr[15]=15;            // element 15 assigned to channel 15
gainArr[0]=0;            // element 0 assigned to input type 0
...
gainArr[15]=4;           // element 15 assigned to input range 4
scanChCount=1;           //only sample chArr[0] (channel 0 )
sampleRate=25000.0; //set the sample rate to 25 KHz
scanMode=1;              // use M1 standard mode
triggerSource=1;         // use internal interrupt signal Mode
triggerState=0;
i8014W_ConfigMagicScan(slot,chArr,gainArr,scanChCount, sampleRate,
scanMode,triggerSource,triggerState,& realsampleRate);
```

[C#]

```
int slot, chArr[16], gainArr[16], scanChCount;
float sampleRate,realsampleRate;
int scanMode, triggerSource, triggerState;
chArr[0]=0;                 // element 0 assigned to channel 0
...
chArr[15]=15;               // element 15 assigned to channel 15
gainArr[0]=0;               // element 0 assigned to input type 0
...
gainArr[15]=4;              // element 15 assigned to input range 4
scanChCount=1;              //only sample chArr[0] (channel 0 )
sampleRate=25000.0; //set the sample rate to 25 KHz
scanMode=1;                 // use M1 standard mode
triggerSource=1;            // use internal interrupt signal Mode
triggerState=0;
pac8014Wnet.pac8014W.ConfigMagicScan(slot, chArr , gainArr , scanChCount, sampleRate,
scanMode, triggerSource, triggerState, ref realsampleRate);
```

[C] (for LinPAC)

```
int slot, chArr[16], gainArr[16], scanChCount, scanMode, triggerSource, triggerState;
float sampleRate, realsampleRate;
chArr[0]=0;              // element 0 assigned to channel 0
...
chArr[15]=15;            // element 15 assigned to channel 15
gainArr[0]=0;            // element 0 assigned to input type 0
...
gainArr[15]=4;           // element 15 assigned to input range 4
scanChCount=1;           //only sample chArr[0] (channel 0 )
sampleRate=25000.0; //set the sample rate to 25 KHz
scanMode=1;              // use M1 standard mode
triggerSource=1;         // use internal interrupt signal Mode
triggerState=0;
Open_Slot(slot);
i8014W_Init(slot);

i8014W_ConfigMagicScan(slot, chArr, gainArr, scanChCount, sampleRate, scanMode,
triggerSource, triggerState, & realsampleRate);
```

4.10. i8014W_StartMagicScan / pac_i8014W_StartMagicScan

This function is used to start Magic Scan. Once Magic scan starts, the converted data is immediately save to FIFO. When an external hardware trigger is selected, after this function is executed, module will wait until receive a trigger signal.

Syntax

For MiniOS7

```
short i8014W_StartMagicScan(int slot);
```

For Windows (CE and WES)

```
short pac_i8014W_StartMagicScan(int slot);
```

For Linux

```
short i8014W_StartMagicScan(int slot);          // for LinPAC-8000
short i9014_StartMagicScan(int slot);           // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

Return Values

0 = No Error

Example

[C/C++]

```
int slot;

i8014W_StartMagicScan(slot);
```

[C#]

```
int slot;

pac8014Wnet.pac8014W.StartMagicScan(slot);
```

[C] (for LinPAC)

```
int main(){

    int slot;

    Open_Slot(slot);

    i8014W_Init(slot);
    i8014W_StartMagicScan(slot);
    Close_Slot(slot);

    return 0;

}
```

4.11. i8014W_StopMagicScan / pac_i8014W_StopMagicScan

This function is used to stop Magic Scan.

All operations for saving data to the FIFO buffer are also stopped because no data will be converted.

Syntax

For MiniOS7

```
short i8014W_StopMagicScan(int slot);
```

For Windows (CE and WES)

```
short pac_i8014W_StopMagicScan(int slot);
```

For Linux

```
short i8014W_StopMagicScan(int slot);          // for LinPAC-8000
short i9014_StopMagicScan(int slot);           // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

Return Values

0 = No Error

Example

[C/C++]

```
int slot;

i8014W_StopMagicScan(slot);
```

[C#]

```
int slot;

pac8014Wnet.pac8014W.StopMagicScan(slot);
```

[C/C++]

```
int main(){

    int slot;

    Open_Slot(slot);

    i8014W_Init(slot);
    i8014W_StopMagicScan(slot);
    Close_Slot(slot);

    return 0;

}
```

4.12. i8014W_ReadFIFO / pac_i8014W_ReadFIFO

After starting Magic Scan function, user needs call any one of reading FIFO functions to obtained data from FIFO.

This one is used to read data from FIFO in Non-Block mode, users need to keep calling this function to avoid FIFO overflow until all data are obtained, and then stop Magic Scan.

Syntax

For MiniOS7

```
short i8014W_ReadFIFO(
int slot,
shor thexData[],
shor treadCount,
short* dataCountFromFIFO
);
```

For Windows (CE and WES)

```
short pac_i8014W_ReadFIFO(
int slot,
shor thexData[],
shor treadCount,
short* dataCountFromFIFO
);
```

For Linux

```
short i8014W_ReadFIFO(
int slot,
shor thexData[],
shor treadCount,
short* dataCountFromFIFO
);                                                    // for LinPAC-8000

short i9014_ReadFIFO(
int slot,
shor thexData[],
shor treadCount,
short* dataCountFromFIFO
);                                                    // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

hexData []:

Specifies the starting address of the data array used to store the data read in hexadecimal format.

readCount:

Specifies the amount of data required.

** dataCountFromFIFO:*

[Output] the amount of data read in this process.

Total sample rate	default read count
<1000	8
>1000, <4000	16
>4000, <10000	32
>10000, <12000	64
>12000, <20000	128
>20000, <32000	256
>32000, <40000	512
>40000	2048

Return Values

0 = No Error

Example

[C/C++]

```
int slot, ret;
short buffer[5000] ,hexData[204800], readCnt=0, readCntfromfifo=0;
long totalRead =0, totalSaved, TargetCnt=10000;
pac_i8014W_StartMagicScan(slot);
while(true){
   ret= pac_i8014W_ReadFIFO(slot, buffer, readCnt, &readCntfromfifo);
   if(ret==FIFO_LATCHED){
         pac_i8014W_StopMagicScan(slot);
         break;
   }
   else{
         if(ReadCntFromFIFO>0){
               memcpy(hexData + totalRead, buffer, sizeof(short)*(ReadCntFromFIFO));
               totalRead+=ReadCntFromFIFO;
               if(totalRead >= TargetCnt){
                     pac_i8014W_StopMagicScan(slot);
                     break;
               }
         }
}
}
```

[C#]

```
int slot;
short hexData[8192], totalScaned=0,TargetCnt=1000, readCnt=0;
pac8014Wnet.pac8014W.StartMagicScan(slot);
while(true){
   pac8014Wnet.pac8014W.ReadFIFO (slot,hexData+totalScaned,
TargetCnt-totalScaned,&readCnt);
   if(readCnt>0){
         totalScaned+=readCnt;
         if(totalScaned>= TargetCnt){
               pac8014Wnet.pac8014W.StopMagicScan(slot);
               break;
         }
   }
   if(readCnt==MAX_FIFO || totalScaned>=TargetCnt){
         pac8014Wnet.pac8014W.StopMagicScan(slot);
   }
}
```

[C] (for LinPAC)

```
int slot, ret;
short hexData[8192] , totalScaned=0,TargetCnt=1000, readCnt=0;
Open_Slot(slot);
i8014W_Init(slot);                                    // initialize
i8014W_StartMagicScan(slot);
while(true){
   ret= i8014W_ReadFIFO (slot,hexData+totalScaned, TargetCnt-totalScaned,&readCnt);
   if(readCnt>0){
         totalScaned+=readCnt;
         if(totalScaned>= TargetCnt){
               i8014W_StopMagicScan(slot);
               break;
         }
   }
   if(readCnt==MAX_FIFO || totalScaned>=TargetCnt){
         i8014W_StopMagicScan(slot);
   i8014W_UnLockFIFO(slot);
         i8014W_ClearFIFO(slot);
   }
}
```

4.13. i8014W_ReadFIFO_BlockMode / pac_i8014W_ReadFIFO_BlockMode

After starting Magic Scan function, user needs call any one of reading FIFO functions to obtained data from FIFO.

This one is used to read data from FIFO in Block mode, users need to call this function after starting the Magic Scan function.

The application will remain blocked until the sampling is completed.

Syntax

For MiniOS7

```
short i8014W_ReadFIFO_BlockMode(
int slot, short dataBuf[], long readCount,long * dataCountFromFIFO
);
```

For Windows (CE and WES)

```
short pac_i8014W_ReadFIFO_BlockMode(
int slot, short dataBuf[], long readCount,long *dataCountFromFIFO
);
```

For Linux

```
short i8014W_ReadFIFO_BlockMode(
int slot, short dataBuf[], long readCount,long * dataCountFromFIFO); // for LinPAC-8000
short i9014_ReadFIFO_BlockMode(
int slot, short dataBuf[], long readCount,long * dataCountFromFIFO); // for LP-9000, LX-9000
```

Parameter

slot:

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

dataBuf []:

Specifies the starting address of the data array used to store the data read in hexadecimal format.

readCount:

specifies the amount of data required.

** dataCountFromFIFO:*

[Output] the amount of data read in this process.

Return Values

0 = No Error

Example

[C/C++]

```
int slot,ret;
short buffer [10000] , readCntfromfifo=0;
long TargetCnt=10000;

ret=pac_i8014W_ReadFIFO_BlockMode(slot,buffer[],TargetCnt,&readCntfromfifo);
pac_i8014W_StopMagicScan(slot);
if(ret== FIFO_LATCHED){
   //Error
}
```

[C#]

```
int slot,ret;
short hexData[10000];
long readCnt=0;
short totalScaned=0,TargetCnt=10000;
pac8014Wnet.pac8014W.StartMagicScan(slot);
ret=pac8014Wnet.pac8014W.ReadFIFO_BlockMode(slot, hexData, TargetCnt, ref readCnt);
pac8014Wnet.pac8014W.StopMagicScan(slot);
if(ret== FIFO_LATCHED){
   //Error
}
```

[C] (for LinPAC)

```
int slot, chArr[16], gainArr[16], scanMode=0, triggerSource=-1, triggerState, scanChCount=0;
short ret, rawData[8192];
long readCnt=0, TargetCnt=10000;
float realsampleRate, sampleRate=0;
Open_Slot(slot);
i8014W_Init(slot);                              // initialize
i8014W_ConfigMagicScan(slot, chArr, gainArr, scanChCount, sampleRate, scanMode,
triggerSource, triggerState, &realsampleRate);
i8014W_StartMagicScan(slot);
ret=i8014W_ReadFIFO_BlockMode(slot, rawData, TargetCnt , &readCnt);
i8014W_StopMagicScan(slot);
if(ret== FIFO_LATCHED){
   //Error
}
```

4.14. i8014W_ReadFIFO_InISR / pac_i8014W_ReadFIFO_InISR

After starting Magic Scan function, user needs call any one of reading FIFO functions to obtained data from FIFO.

This one is used to read data from FIFO in interrupt mode, users need to install an ISR function and call this function in it.

Syntax

For MiniOS7

```
short i8014W_ReadFIFO_InISR (
int slot, short hexData [], short triggerLevel, short * dataCountFromFIFO);
```

For Windows (CE and WES)

```
short pac_i8014W_ReadFIFO_InISR (
int slot, short hexData [], short triggerLevel, short * dataCountFromFIFO);
```

For Linux

```
short i8014W_ReadFIFO_InISR (
    int slot, short hexData [], short triggerLevel, short * dataCountFromFIFO
);                                                          // for LinPAC-8000
short i9014_ReadFIFO_InISR (
int slot, short hexData [], short triggerLevel, short * dataCountFromFIFO
);                                                          // for LP-9000, LX-9000
```

Parameter

slot:

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

hexData []:

Specifies the starting address of the data array used to store the data read in hexadecimal format.

triggerLevel:

Specifies the amount of data to read.

triggerLevel	Read Count	triggerLevel	Read Count
0	8	4	128
1	16	5	256
2	32	6	512
3	64	7	2048

** dataCountFromFIFO:*

[Output] the amount of data read in this process.

Return Values

0 = No Error

Example

[C/C++]

```
int slot,ret,TrgLevel;
short buffer [10000] , readCntfromfifo=0;
long TargetCnt=10000 ,totalRead =0;
void main()
{
...
pac_i8014W_InstallMagicScanISR(slot,ISRFUN, TrgLevel);
pac_i8014W_StopMagicScan(slot);
While(1){
        If(totalRead>= TargetCnt){
        …
        }
}
}
void ISRFUN(int slot);
{
   ret=i8014W_ReadFIFO_InISR(slot, buffer, TrgLevel, readCntfromfifo);
totalRead +=readCntfromfifo;
i8014W_ClearInt(slot);
   if(ret== FIFO_LATCHED){
        i8014W_StopMagicScan(slot);
        i8014W_UnLockFIFO(slot);
        i8014W_ClearFIFO(slot);
   }
}
```

[C#]

```
int slot, ret, TrgLevel;

short buffer [10000] , readCntfromfifo=0;
long TargetCnt=10000;

void main()

{

...

pac8014Wnet.pac8014W.InstallMagicScanISR(slot, ISRFUN, TrgLevel);

pac8014Wnet.pac8014W.StartMagicScan(slot);

While(1)     {      …     }

}

void ISRFUN(int slot);

{

   ret=pac8014Wnet.pac8014W.ReadFIFOInISR(slot, buffer [],TrgLevel, ref readCntfromfifo);

   pac8014Wnet.pac8014W.ClearInt(slot);

   if(ret== FIFO_LATCHED)

   {

         pac8014Wnet.pac8014W.StopMagicScan(slot);

         pac8014Wnet.pac8014W.UnLockFIFO(slot);

         pac8014Wnet.pac8014W.ClearFIFO(slot);

   }

}
```

[C] (for LinPAC)

```
int slot, ret, TrgLevel;
short buffer [10000], readCntfromfifo=0;
long TargetCnt=10000, totalRead =0;
void main()
{
...
i8014W_InstallMagicScanISR(slot, ISRFUN, TrgLevel);
i8014W_StopMagicScan(slot);
While(1){
        If(totalRead>= TargetCnt){
            …
        }
}
}
void ISRFUN(int slot);
{
   ret=i8014W_ReadFIFO_InISR(slot, buffer, TrgLevel, readCntfromfifo);
totalRead +=readCntfromfifo;
i8014W_ClearInt(slot);
   if(ret== FIFO_LATCHED){
        i8014W_StopMagicScan(slot);
        i8014W_UnLockFIFO(slot);
        i8014W_ClearFIFO(slot);
   }
}
```

4.15. i8014W_UnLockFIFO / pac_i8014W_UnLockFIFO

This function is used to unlock the FIFO buffer when it is locked after being filled. Ensure that the FIFO buffer is unlocked and cleared before starting the next Magic Scan process.

Syntax

For MiniOS7

```
void i804W_UnLockFIFO (int slot);
```

For Windows (CE and WES)

```
void pac_i8014W_UnLockFIFO(int slot);
```

For Linux

```
void i8014W_UnLockFIFO(int slot);          // for LinPAC-8000
void i9014_UnLockFIFO(int slot);           // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

Return Values

None

Example

[C/C++]

```
int slot;
i8014W_UnLockFIFO (slot);
```

[C#]

```
int slot;
pac8014Wnet.pac8014W.UnLockFIFO(slot);
```

[C] (for LinPAC)

```
void main()
{
    int slot;
    Open_Slot(slot);
    i8014W_Init(slot);       // initialize
    i8014W_UnLockFIFO(slot);
    Close_Slot(slot);
    return 0;
}
```

4.16. i8014W_ClearFIFO / pac_i8014W_ClearFIFO

This function is used to clear the FIFO buffer after the FIFO has been unlocked. Ensure that the FIFO buffer is unlocked and cleared before starting the next Magic Scan process.

Syntax

For MiniOS7

```
void i8014W_ClearFIFO (int slot);
```

For Windows (CE and WES)

```
void pac_i8014W_ClearFIFO(int slot);
```

For Linux

```
void i8014W_ClearFIFO(int slot);          // for LinPAC-8000
void i9014_ClearFIFO(int slot);           // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

Return Values

None

Example

[C/C++]

```
int slot;
i8014W_ClearFIFO (slot);
```

[C#]

```
int slot;
pac8014Wnet.pac8014W.ClearFIFO(slot);
```

[C] (for LinPAC)

```
int slot;
Open_Slot(slot);
i804W_ClearFIFO (slot);
```

4.17. i8014W_InstallMagicScanISR / pac_i8014W_InstallMagicScanISR

This function is used to install the ISR to control interrupt events form the FIFO. When the amount of data in the FIFO buffer is greater than the value defined by the triggerLevel parameter, an interrupt event will occurs and the ISR will be executed to handle the event. In the ISR, use the ReadFIFO to transfer data from the FIFO buffer and then ClearInt to reset the status of the interrupt.

Syntax

For MiniOS7

```
short i8014W_ InstallMagicScanISR(
int slot,
void (*isr)(int slot),
inttriggerLevel
);
```

For Windows (CE and WES)

```
short pac_i8014W_InstallMagicScanISR(
int slot,
void(*isr)(int slot),
shorttriggerLevel
);
```

For Linux

```
short i8014W_ InstallMagicScanISR(
int slot,
void (*isr)(int slot),
   inttriggerLevel
);                                                    // for LinPAC-8000
short i9014_ InstallMagicScanISR(
int slot,
void (*isr)(int slot),
inttriggerLevel
);                                                    // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

**isr (int slot):*

the function pointer passed for the ISR.

triggerLevel:

specifies the interrupt trigger condition (0 – 7) based on the amount of data in the FIFO buffer.
If the value is set to greater than 7, it will be automatically forced to 7.
If the amount of data in the FIFO buffer is greater than the value defined by the triggerLevel parameter, the interrupt will be triggered and the ISR will be executed to handle the interrupt event.
The following is a definition of the triggerLevelvalues table lists the definition of triggerLevel and associated Data Count values:

triggerLevel	Read Count	triggerLevel	Read Count
0	8	4	128
1	16	5	256
2	32	6	512
3	64	7	2048

Return Values

0 = No Error

Example

[C/C++]

```
void main()
{
int slot, TrgLevel;
i8014W_InstallMagicScanISR(slot, ISRFUN, TrgLevel);
...
i8014W_UnInstallMagicScanISR(slot);
}
void ISRFUN(int slot);
{
    ….
}
```

[C#]

```
void main()
{
int slot, TrgLevel;
pac8014Wnet.pac8014W.InstallMagicScanISR(slot, ISRFUN, TrgLevel);
...
i8014W_UnInstallMagicScanISR(slot);
}
void ISRFUN(int slot);
{
….
}
```

[C] (for LinPAC)

```
void main()
{
   int slot, TrgLevel;
   Open_Slot(9);
   Open_Slot(slot);
   i8014W_Init(slot);                    // initialize
   ....
   i8014W_InstallMagicScanISR(slot, ISRFUN, TrgLevel);
   …
   i8014W_UnInstallMagicScanISR(slot);
   Close_Slot(slot);
         Close_Slot(9);
   return 0;
}

void ISRFUN(int slot);
{
   ….
}
```

4.18. i8014W_UnInstallMagicScanISR / pac_i8014W_UnInstallMagicScanISR

This function is used to uninstall the Magic Scan ISR.

Syntax

For MiniOS7

```
short i8014W_UnInstallMagicScanISR(int slot);
```

For Windows (CE and WES)

```
short pac_i8014W_UnInstallMagicScanISR(int slot);
```

For Linux

```
short i8014W_UnInstallMagicScanISR(int slot);          // for LinPAC-8000
short i9014_UnInstallMagicScanISR(int slot);           // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

Return Values

0 = No Error

Example

[C/C++]

```
int slot;

i8014W_UnInstallMagicScanISR (slot);
```

[C#]

```
int slot;

pac8014Wnet.pac8014W.UnInstallMagicScanISR(slot);
```

[C/C++]

```
int slot;
Open_Slot(slot);

i8014W_UnInstallMagicScanISR (slot);
```

4.19. i8014W_ClearInt / pac_i8014W_ClearInt

This function is used to clear the status of the Magic Scan interrupts. When using ISR, this function should be called to clear the status of any interrupts that have been triggered in order to continue processing future interrupt events.

Syntax

For MiniOS7

```
void i8014W_ClearInt (int slot);
```

For Windows (CE and WES)

```
void pac_i8014W_ClearInt(int slot);
```

For Linux

```
void i8014W_ClearInt(int slot);                    // for LinPAC-8000
void i9014_ClearInt(int slot);                     // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

Return Values

None

Example

[C/C++]

```
int slot;
i8014W_ ClearInt (slot);
```

[C#]

```
int slot;
pac8014Wnet.pac8014W.ClearInt(slot);
```

[C] (for LinPAC)

```
int slot;
i8014W_ ClearInt (slot);
```

4.20. i8014W_ReadGainOffset / pac_i8014W_ReadGainOffset

This function is used to obtain the gain and offset values on each input type for I-8014W and I-9014.

Syntax

For MiniOS7

```
void i8014W_ ReadGainOffset(
int slot, int gain, unsigned short* gainValue, short* offsetValue
);
```

For Windows (CE and WES)

```
void pac_i8014W_ReadGainOffset(
int slot, short gain, unsigned short* gainValue, short* offsetValue
);
```

For Linux

```
void i8014W_ReadGainOffset(
int slot, short gain, unsigned short* gainValue, short* offsetValue
);                                                    // for LinPAC-8000
void i9014_ReadGainOffset(
int slot, short gain, unsigned short* gainValue, short* offsetValue
);                                                    // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

Gain:

specifies the input type (0 – 4), where:**0**: +/-10 V, **1**: +/-5 V, **2**: +/-2.5 V, **3**: +/-1.25 V, **4**: +/-20 mA

**gainValue:*

[Output] the gain value for the input range.

**offsetValue:*

[Output] the offset value for the input range.

Return Values

None

Example

[C/C++]

```
unsigned short gVal=0;

short oVal=0, slot, gain;

i8014W_ReadGainOffset(slot, gain, &gVal, &oVal);
```

[C#]

```
int slot,gain,gval;

uint16 oval;

pac8014Wnet.pac8014W.ReadGainOffset(slot, gain, ref gval, ref oval);
```

[C] (for LinPAC)

```
int main(){

    int slot, gain;

    unsigned short gVal=0;

    short oVal=0;
    Open_Slot(slot);
    i8014W_Init(int slot);

    i8014W_ReadGainOffset(slot, gain, &gVal, &oVal);
    printf("\nThe Gain and Offset values for Calibration are: Gain=%u; Offset=%d", ch, gVal,
oVal);
    Close_Slot(slot);
    return 0;

}
```

4.21. i8014W_CalibrateData / pac_i8014W_CalibrateData

This function is used to calibrate the raw data read during the Magic Scan process and to convert the data to a floating point value.

Syntax

For MiniOS7

```
void i8014W_CalibrateData(
int slot, short iGain, short dataFromFIFO, float* calibratedAI
);
```

For Windows (CE and WES)

```
void pac_i8014W_CalibrateData(
int slot, short iGain, short dataFromFIFO, float* calibratedAI
);
```

For Linux

```
void i8014W_CalibrateData(
    int slot, short iGain, short dataFromFIFO, float* calibratedAI
);                                                          // for LinPAC-8000
void i9014_CalibrateData(
int slot, short iGain, short dataFromFIFO, float* calibratedAI
);                                                          // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

iGain:

specifies the input type (0 – 4), where:**0**: +/-10 V, **1**: +/-5 V, **2**: +/-2.5 V, **3**: +/-1.25 V, **4**: +/-20 mA

dataFromFIFO:

the raw data read from the FIFO buffer.

** calibratedAI:*

[Output] the floating point value.

Return Values

None

Example

[C/C++]

```
int slot, I, scanChCount;
short hexData[8192], TargetCnt=1000, gainArr[16];
float calibratedAI=0;
printf("Start printing all the data:\n\n\r");
for(i=0;i<totalScaned;i++);
{
slot = 0;
i8014W_CalibrateData(slot, gainArr[I %scanChCount], hexData[i], &calibratedAI);
printf("Arr[%d]=[%5.4f]\t",i%scanChCount, calibratedAI);
}
```

[C#]

```
int slot, I, scanChCount;
short hexData[8192], TargetCnt=1000, gainArr[16];
float calibratedAI=0;
printf("Start printing all the data:\n\n\r");
for(i=0;i<totalScaned;i++);
{
slot = 0;
pac8014Wnet.pac8014W.CalibrateData (slot, gainArr[I %scanChCount], hexData[i],
                                    ref calibratedAI);
printf("Arr[%d]=[%5.4f]\t", i%scanChCount, calibratedAI);
}
```

[C] (for LinPAC)

```
int slot=1, I, scanChCount;
short hexData[8192], totalScaned =1000, gainArr[16];
float calibratedAI=0;
Open_Slot(slot);
i8014W_Init(slot);
.....
printf("Start printing all the data:\n\n");
for(i=0; i<totalScaned; i++);
{
i8014W_CalibrateData(slot, gainArr[I %scanChCount], hexData[i], &calibratedAI);
printf("Arr[%d]=[%5.4f]\t", i%scanChCount, calibratedAI);
}
```

4.22. i8014W_CalibrateDataHex / pac_i8014W_CalibrateDataHex

This function is used to calibrate the raw data read in Magic Scan process.

Syntax

For MiniOS7

```
void i8014W_CalibrateDataHex(
int slot, short iGain, short dataFromFIFO, short* calibratedAI
);
```

For Windows (CE and WES)

```
void pac_i8014W_ CalibrateDataHex(
int slot, short iGain, short dataFromFIFO, short* calibratedAI
);
```

For Linux

```
void pac_i8014W_ CalibrateDataHex(
int slot, short iGain, short dataFromFIFO, short* calibratedAI
);                                                    // for LinPAC-8000
void pac_i9014_ CalibrateDataHex(
int slot, short iGain, short dataFromFIFO, short* calibratedAI
);                                                    // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

iGain:

specifies the input type (0 – 4), where:**0**: +/-10 V, **1**: +/-5 V, **2**: +/-2.5 V, **3**: +/-1.25 V, **4**: +/-20 mA

dataFromFIFO:

the raw data read from the FIFO buffer.

** calibratedAI :*

[Output] the calibrated hexadecimal value.

Return Values

None

Example

[C/C++]

```
int slot, I, scanChCount;
short hexData[8192], totalScaned =1000, calibratedAI=0, gainArr[16];
long readCnt=0;
printf("Start printing all the data:\n\n\r");
for(i=0;i<totalScaned;i++);
{
slot = 0;
i8014W_CalibrateDataHex (slot, gainArr[I %scanChCount], hexData[i], &calibratedAI);
printf("Arr[%d]=[%5.4f]\t",i%scanChCount, calibratedAI);
}
```

[C#]

```
int slot, I, calibratedAI=0, scanChCount;
short hexData[8192], totalScaned =1000, gainArr[16];
long readCnt=0;
printf("Start printing all the data:\n\n\r");
for(i=0;i<totalScaned;i++);
{
slot = 0;
pac8014Wnet.pac8014W.CalibrateDataHex (slot, gainArr[I %scanChCount],hexData[i],
                                                    ref calibratedAI);
printf("Arr[%d]=[%5.4f]\t",i%scanChCount, calibratedAI);
}
```

[C] (for LinPAC)

```
int slot, I, scanChCount;
short hexData[8192], totalScaned =1000, calibratedAI=0, gainArr[16];
long readCnt=0;
Open_Slot(slot);
i8014W_Init(slot);
printf("Start printing all the data:\n\n");
for(i=0;i<totalScaned;i++);
{
slot = 0;
i8014W_CalibrateDataHex (slot, gainArr[I %scanChCount], hexData[i], &calibratedAI);
printf("Arr[%d]=[%5.4f]\t", i%scanChCount, calibratedAI);
}
```

4.23. i8014W_Read_mA_GainOffset / pac_i8014W_Read_mA_GainOffset

This function is used to obtain the gain and offset values on each input type for I-8014CW/I-9014C.

Syntax

For MiniOS7

```
void i8014W_Read_mA_GainOffset(
int slot, int channel, unsigned short* gainValue, short* offsetValue
);
```

For Windows (CE and WES)

```
void pac_ i8014W_Read_mA_GainOffset(
int slot, int channel,unsigned short* gainValue, short* offsetValue
);
```

For Linux

```
void i8014W_Read_mA_GainOffset(
int slot, int channel, unsigned short* gainValue, short* offsetValue
);                                                    // for LinPAC-8000
void i9014_Read_mA_GainOffset(
int slot, int channel, unsigned short* gainValue, short* offsetValue
);                                                    // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

Channel:

specifies the channel (0 – 7), for +/-20 mA

**gainValue:*

[Output] the gain value for the input range.

**offsetValue:*

[Output] the offset value for the input range.

Return Values

None

Example

[C/C++]

```
unsigned short gVal=0;
short oVal=0,slot,gain;
i8014W_Read_mA_GainOffset (slot,ch,&gVal,&oVal);
```

[C#]

```
int slot,gain,gval;
uint16 oval;
pac8014Wnet.pac8014W.Read_mA_GainOffset(slot, gain, ref gval, ref oval);
```

[C] (for LinPAC)

```
int main(){
    unsigned short gVal=0;
    short oVal=0;
    int slot, gain;
    Open_Slot(slot);
    i8014W_Init(slot);
    i8014W_Read_mA_GainOffset (slot, ch, &gVal, &oVal);
    Close_Slot(slot);
    return 0;
}
```

4.24. i8014W_Calibrate_CH_mA / pac_pac_i8014W_Calibrate_CH_mA

This function is used to calibrate the Magic Scan raw data and convert the data to a float format.

Syntax

For MiniOS7

```
void i8014W_Calibrate_CH_mA(
    int slot, int ch, short dataFromFIFO, float* calibratedAI
);
```

For Windows (CE and WES)

```
void pac_i8014W_Calibrate_CH_mA(
    int slot, int ch, short dataFromFIFO, float* calibratedAI
);
```

For Linux

```
void i8014W_Calibrate_CH_mA(
    int slot, int ch, short dataFromFIFO, float* calibratedAI
);                                                    // for LinPAC-8000
void i9014_Calibrate_CH_mA(
    int slot, int ch, short dataFromFIFO, float* calibratedAI
);                                                    // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

Channel:

specifies the channel (0 – 7), for +/-20 mA

dataFromFIFO:

the raw data read from the FIFO buffer.

** calibratedAI :*

[Output] the floating point value.

Return Values

None

Example

[C/C++]

```
int slot, ch;
short hexData;
float calibratedAI=0;
i8014W_Calibrate_CH_mA (slot, ch ,hexData, &calibratedAI);
```

[C#]

```
int slot, ch;
short hexData;
float calibratedAI=0;
pac8014Wnet.pac8014W.Calibrate_CH_mA(slot, ch, hexData, ref calibratedAI);
```

[C] (for LinPAC)

```
int main(){
   int slot, ch;
   short hexData;
   float calibratedAI=0;
   Open_Slot(slot);
   i8014W_Init(slot);
   i8014W_Calibrate_CH_mA (slot, ch ,hexData, &calibratedAI);
   Close_Slot(slot);
   return 0;
}
```

4.25. i8014W_Calibrate_CH_mA_Hex / pac_i8014W_Calibrate_CH_mA_Hex

This function is used to calibrate the Magic Scan raw data and convert the data to a hexadecimal format.

Syntax

For MiniOS7

```
void i8014W_Calibrate_CH_mA_Hex(
   int slot, int ch, short dataFromFIFO, short* calibratedAI
);
```

For Windows (CE and WES)

```
void pac_i8014W_Calibrate_CH_mA_Hex(
   int slot, int ch, short dataFromFIFO, short* calibratedAI
);
```

For Linux

```
void i8014W_Calibrate_CH_mA_Hex(
   int slot, int ch, short dataFromFIFO, short* calibratedAI
);                                                    // for LinPAC-8000
void i9014_Calibrate_CH_mA_Hex(
   int slot, int ch, short dataFromFIFO, short* calibratedAI
);                                                    // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

Channel:

specifies the channel (0 – 7), for +/-20 mA

dataFromFIFO:

the raw data read from the FIFO buffer.

** calibratedAI :*

[Output] the calibrated hexadecimal value.

Return Values

None

Example

[C/C++]

```
int slot, ch;
short hexData;
float calibratedAI=0;
i8014W_Calibrate_CH_mA_Hex(slot, ch ,hexData, &calibratedAI);
```

[C#]

```
int slot, ch;
short hexData;
float calibratedAI=0;
pac8014Wnet.pac8014W.Calibrate_CH_mA_Hex (slot, ch, hexData, ref calibratedAI);
```

[C] (for LinPAC)

```
int main(){
    int slot, ch;
    short hexData;
    float calibratedAI=0;
    Open_Slot(slot);
    i8014W_Init(slot);
    i8014W_Calibrate_CH_mA_Hex(slot, ch ,hexData, &calibratedAI);
    Close_Slot(slot);
    return 0;
}
```

4.26. pac_i8014W_Start_Average_INT

This function is used to startup calculate data average with Magic Scan.

Syntax

For Windows (CE6.0 and CE7.0)

```
short pac_i8014W_Start_Average_INT(
int slot, short gain, float sample_rate, short average_level
);
```

For Linux

```
short i8014W_Start_Average_INT(
    int slot, short gain, float sample_rate, short average_level
);                                                    // for LinPAC-8000
short i9014_Start_Average_INT(
int slot, short gain, float sample_rate, short average_level
);                                                    // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

gain:

specifies the input type.

For I-8014W & I-9014 => 0 : +/-10 V, 1 : +/-5 V, 2 : +/-2.5 V, 3 : +/-1.25 V, 4 : +/-20 mA

For I-8014CW & I-9014C => 4 : +/-20 mA

sample_rate:

Sampling rate.

For differential : 1 ~ 5000 Hz.

For single-ended : 1 ~ 2500 Hz.

average_level:

Set the amount of data to be averaged (0 ~ 7).

0 : Calculate average for every 1 data

1 : Calculate average for every 2 data

2 : Calculate average for every 4 data

3 : Calculate average for every 8 data

4 : Calculate average for every 16 data

5 : Calculate average for every 32 data

6 : Calculate average for every 64 data

7 : Calculate average for every 256 data

Return Values

0 : No error, find I-8014W / I-9014.

1 : No error, find I-8014CW / I-9014C.

Example

[C/C++]

```
int slot=0, gain=0, average_level=7;
float sample_rate=2000;
pac_i8014W_Start_Average_INT(slot,gain, sample_rate, average_level);
```

[C#]

```
int slot=0, gain=0,average_level=7;
float sample_rate=2000;
pac8014WNet.pac8014W.Start_Average_INT(slot, gain, sample_rate, average_level);
```

[C] (for LinPAC)

```
int main(){
    int slot=1, gain=0, average_level=7;
    float sample_rate=2000;
    Open_Slot(slot);
    i8014W_Init(slot);
    i8014W_Start_Average_INT(slot, gain, sample_rate, average_level);
    Close_Slot(slot);
    return 0;
}
```

4.27. pac_i8014W_Stop_Average_INT

This function is used to stop the Average Function

Syntax

For Windows (CE6.0 and CE7.0)

```
short pac_i8014W_Stop_Average_INT(int slot);
```

For Linux

```
short i8014W_Stop_Average_INT(int slot);          // for LinPAC-8000
short i9014_Stop_Average_INT(int slot);           // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

Return Values

0 = No Error

Example

[C/C++]

```
int slot=0;
pac_i8014W_Stop_Average_INT(slot);
```

[C#]

```
int slot=0;
pac8014WNet.pac8014W.Stop_Average_INT(slot);
```

[C] (for LinPAC)

```
int slot=1, ret;
ret=Open_Slot(slot);
if (ret > 0) {
        printf("Open Slot%d failed, return value=%d \n", slot, ret);
        return (-1);
}
i8014W_Init(slot);
i8014W_Stop_Average_INT(slot);
```

4.28. pac_i8014W_ReadAIHex_Average_INT

This function is used to read one calibrated HEX format AI data.

Syntax

For Windows (CE6.0 and CE7.0)

```
short pac_i8014W_ReadAIHex_Average_INT(
    int slot, short ch, short *hVal
);
```

For Linux

```
short i8014W_ReadAIHex_Average_INT(
    int slot, short ch, short *hVal
);                                                    // for LinPAC-8000
short i9014_ReadAIHex_Average_INT(
    int slot, short ch, short *hVal
);                                                    // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

ch:

specifies the channel (0 ~ 7),

For differential : 0 ~ 7.

For single-ended : 0 ~ 15.

** hVal:*

[Output] the calibrated hexadecimal format value.

Return Values

0 = No Error

Example

[C/C++]

```
int slot=0, ch=0;
short hexData=0;
pac_i8014W_ReadAIHex_Average_INT(slot, ch, & hexData);
```

[C#]

```
int slot=0, ch=0;
short hexData=0;
pac8014WNet.pac8014W.ReadAIHex_Average_INT(slot, ch, ref hexData);
```

[C] (for LinPAC)

```
int main(){
    int slot=1, ch=0, ret;
    short hexData=0;
    ret=Open_Slot(slot);
    if (ret > 0) {
            printf("Open Slot%d failed, return value=%d \n", slot, ret);
            return (-1);
    }
    i8014W_Init(slot);
    i8014W_ReadAIHex_Average_INT(slot, ch, &hexData);
    Close_Slot(slot);
    return 0;
}
```

4.29. pac_i8014W_ReadAI_Average_INT

This function is used to read one calibrated float format AI data.

Syntax

For Windows (CE6.0 and CE7.0)

```
short pac_i8014W_ReadAI_Average_INT(
    int slot, short ch, float *fVal
);
```

For Linux

```
short i8014W_ReadAI_Average_INT(
    int slot, short ch, float *fVal
);                                                    // for LinPAC-8000
short i9014_ReadAI_Average_INT(
    int slot, short ch, float *fVal
);                                                    // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

ch:

specifies the channel (0 ~ 7),

For differential : 0 ~ 7.

For single-ended : 0 ~ 15.

** fVal:*

[Output] the calibrated float format value.

Return Values

0 = No Error

Example

[C/C++]

```
int slot=0, ch=0;
float fData=0;
pac_i8014W_ReadAI_Average_INT(slot, ch ,&fData);
```

[C#]

```
int slot=0, ch=0;
float fData =0;
pac8014WNet.pac8014W.ReadAI_Average_INT (slot, ch,ref fData);
```

[C] (for LinPAC)

```
int main(){
    int slot=1, ch=0, ret;
    float fData=0;
    ret=Open_Slot(slot);
    if (ret > 0) {
            printf("Open Slot%d failed, return value=%d \n", slot, ret);
            return (-1);
    }
    i8014W_Init(slot);              // initialize
    i8014W_ReadAI_Average_INT(slot, ch ,&fData);
    Close_Slot(slot);
    return 0;
}
```

4.30. pac_i8014W_ReadAllAIHex_Average_INT

This function is used to read multiple calibrated HEX format AI data

Syntax

For Windows (CE6.0 and CE7.0)

```
short pac_i8014W_ReadAllAIHex_Average_INT(
    int slot, short hVal[]
);
```

For Linux

```
short i8014W_ReadAllAIHex_Average_INT(
    int slot, short hVal[]
);                                                  // for LinPAC-8000
short i9014_ReadAllAIHex_Average_INT(
    int slot, short hVal[]
);                                                  // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

** hVal[]:*

[Output] the calibrated hexadecimal data array.

Return Values

0 = No Error

Example

[C/C++]

```
int slot=0;
short hVal[16];
pac_i8014W_ReadAllAIHex_Average_INT(slot,hVal);
```

[C#]

```
int slot=0;
Int16[] hval = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
pac8014WNet.pac8014W.ReadAllAIHex_Average_INT(slot, hval);
```

[C] (for LinPAC)

```
int main(){
    int slot=1, ret;
    short hVal[16];
    ret=Open_Slot(slot);
    if (ret > 0) {
            printf("Open Slot%d failed, return value=%d \n", slot, ret);
            return (-1);
    }
    i8014W_Init(slot);                    // initialize
    i8014W_ReadAllAIHex_Average_INT(slot, hVal);
    Close_Slot(slot);
    return 0;
}
```

4.31. pac_i8014W_ReadAllAI_Average_INT

This function is used to read multiple calibrated float format AI data

Syntax

For Windows (CE6.0 and CE7.0)

```
short pac_i8014W_ReadAllAI_Average_INT(
    int slot, float fVal[]
);
```

For Linux

```
short i8014W_ReadAllAI_Average_INT(
    int slot, float fVal[]
);                                                  // for LinPAC-8000
short i9014_ReadAllAI_Average_INT(
    int slot, float fVal[]
);                                                  // for LinPAC-9000, LX-9000
```

Parameter

slot:

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

** fVal[]:*

[Output] the calibrated hexadecimal data array.

Return Values

0 = No Error

Example

[C/C++]

```
int slot=0;
float fVal[16];
pac_i8014W_ReadAllAI_Average_INT(slot, fVal);
```

[C#]

```
int slot=0;
float[] fval = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
pac8014WNet.pac8014W.ReadAllAI_Average_INT(slot, fval);
```

[C] (for LinPAC)

```
int main(){
    int slot=1, ret;
    float fVal[16];
    ret=Open_Slot(slot);
    if (ret > 0) {
            printf("Open Slot%d failed, return value=%d \n", slot, ret);
            return (-1);
    }
    i8014W_Init(slot);                    // initialize
    i8014W_ReadAllAI_Average_INT(slot, fVal);
    Close_Slot(slot);
    return 0;
}
```

5. Troubleshooting

This chapter discusses how to solve problems you may encounter.

- What to do when the data read from I-8014W seems unstable
- How to solve the error of FIFO LATCHED (Error Code : -6).
- How to Calibration / Restore defaults.

If there is a problem that cannot be solved, please send an email to service@icpdas.com and tell us the relevant information, such as connection, host, platform, application, operation method.

The more detailed the better.

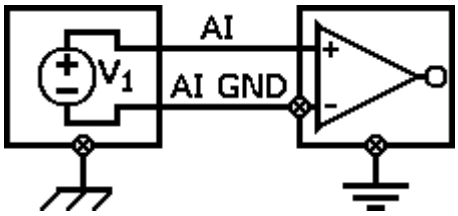
5.1. What to do when the data read from I-8014W seems unstable

If the voltage can be measured correctly when testing using a battery, but not when using the real signal source, the error may be caused by any or all of the following factors:

- The source is disturbed by noise
- Unstable signal source
- Floating signal source not referenced to system ground(earth or building ground)

Because of the high-speed data acquisition function, any noise coupled to a signal or any change in voltage on an unstable source is also captured. In this situation, signal filtering or isolation should be considered in order to enhance the quality of the signal.

In another hand, It is recommended that connect the Vn- pin with the AGND pin when measuring differential signals, as shown in the figure.



5.2. How to solve the error of FIFO LATCHED (Error Code : -6)

After the ***StartMagicScan*** instruction is executed, it will continue scanning the channels and converting data unless the ***StopMagicScan*** command is executed. Consequently, the converted data is continuously saved to the FIFO buffer. If the Magic Scan is not stopped after obtaining the required data, or the data is not read from the FIFO buffer within the required time frame, the FIFO buffer will be filled and then locked. When the FIFO buffer is locked, the FIFO LATCHED error (-6) will occur and any new data will not be able to be saved to the FIFO buffer.

To solve this error, execute the following instructions:

1. Call ***StopMagicScan*** function to Stop Magic.
2. Call ***ReadFIFO*** function to get the remaining data in the FIFO buffer, or ***ClearFIFO*** function to clear it.
3. Call ***UnLockFIFO*** function to unlock the FIFO buffer.
4. Call ***StartMagicScan*** function to reset Magic Scan.

5.3. How to Calibration / Restore defaults

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 stable source for calibration, such as a 3 1/2 digit power supply (or better) or a battery output.
- A 4 1/2 digital 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

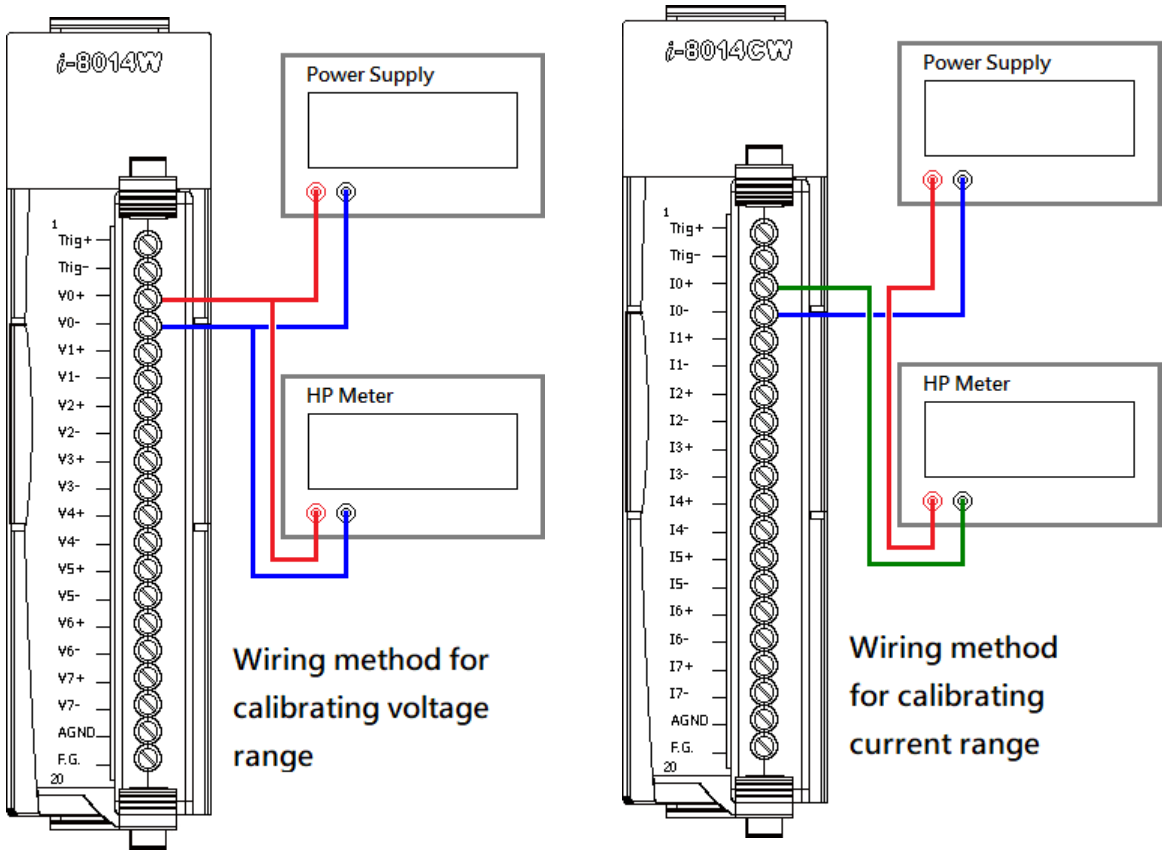
1. An unstable source will cause calibration errors and will affect the accuracy of the data acquisition.
2. I-8014W / I-9014 only uses channel 0 to calibrate every type of range.
3. The gain and offset value of the range of +/- 20 mA for I-8014W / I-9014 are the same as the range of +/- 2.5V.
4. If users wish to calibrate +/- 20 mA, calibrate +/- 2.5V will be fine.
5. I-8014CW / I-9014C needs to calibrate every channel within the range of +/- 20mA.

5.3.1. Calibrate I-8014(C)W on iPAC-8000

Wiring Method

Please refer to the "**2.1.2. Wiring the iPAC-8000**" chapter of the IP-8000 user manual to establish RS-232 connection between the controller and the PC, and connect the power supply to the controller.

Set the Differential/Single-ended jumper to the differential position, connect source and modules in differential mode and connect the voltage or current meter to the wiring, like the following figure:



Then, plug I-8014(C)W into the controller.

Step 1. Download, upload and execute calibration program

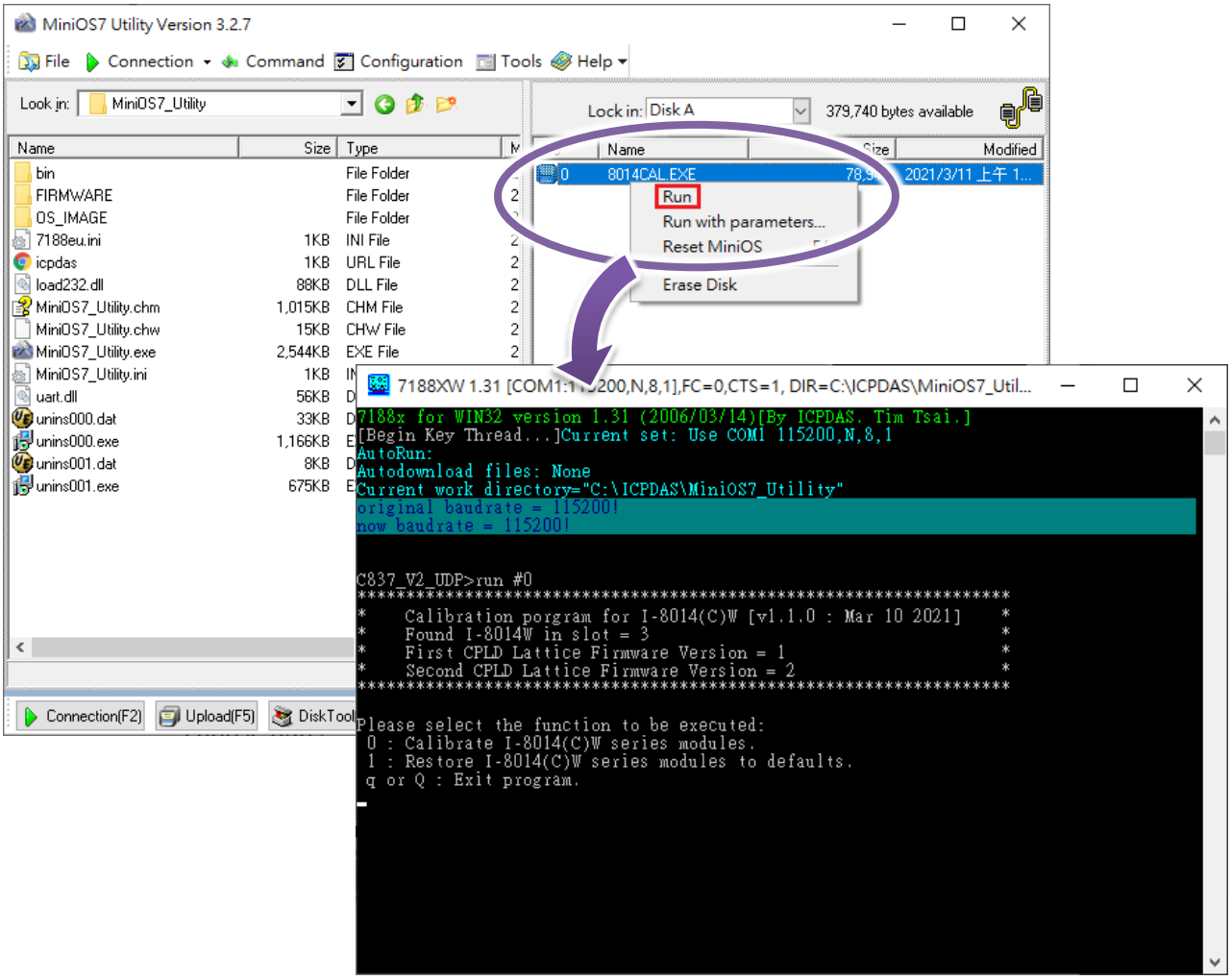
The calibration program can be downloaded in ICP DAS website.

Please refer to the following link:

https://www.icpdas.com/en/download/show.php?num=2897

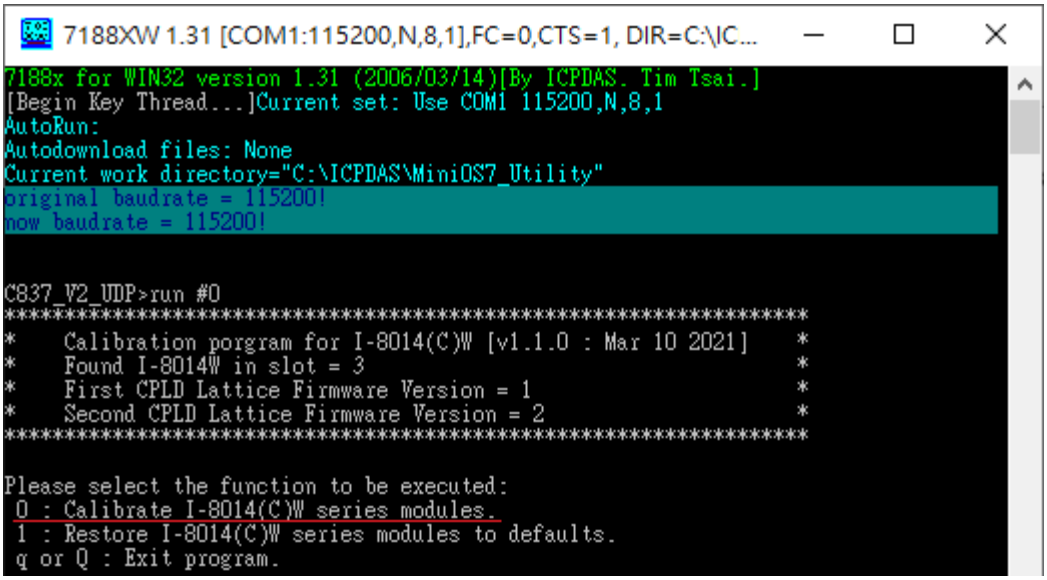
In order to upload programs, please refer to the "2.2.2. Installing the MiniOS7" and “2.5.2. Uploading and Executing iPAC-8000 programs” of the IP-8000 user manual to download MiniOS7 Utility which can help users to upload programs and learn how to operate.

After uploading the calibration program, right click on it and click "Run" to execute it.



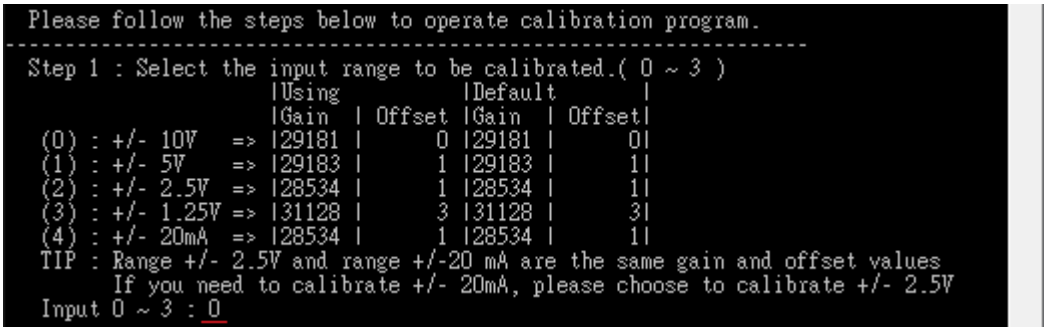
Step 2. Calibrate

After execute the program, press ‘0’ to calibrate modules and follow the steps appeared at the screen.

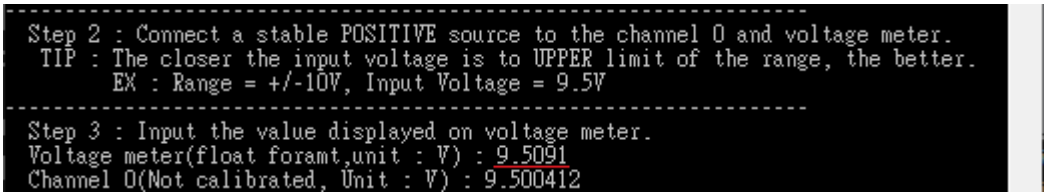


Select the range of I-8014W.

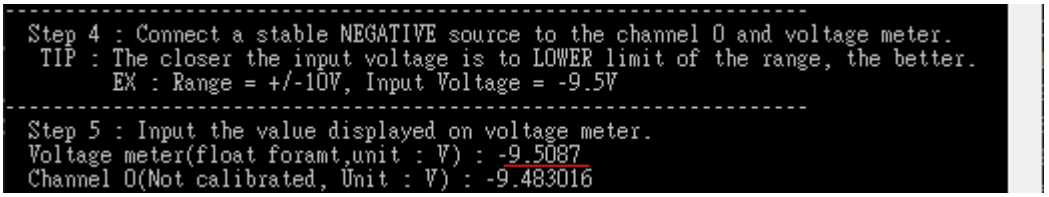
If you want to calibrate I-8014W / I-9014 with ±20 mA, please select ±2.5 V, because both types of range use the same gain and offset values.



Output stable positive source to channel 0, and type the value displayed on the voltage meter.



Output stable negative source to channel 0, and type the value displayed on the voltage meter.



After finish Step 5, the new gain and offset values will be showed at the screen.

Press ‘r’ 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.

Press ‘s’ to save new gain and offset values.

This program can also be used to calibrate I-8014CW / I-9014C, the differences are that users need to output current as source and select channel to be calibrated.

5.3.2. Restore I-8014(C)W to defaults on iPAC-8000

Step 1. Download, upload and execute calibration program

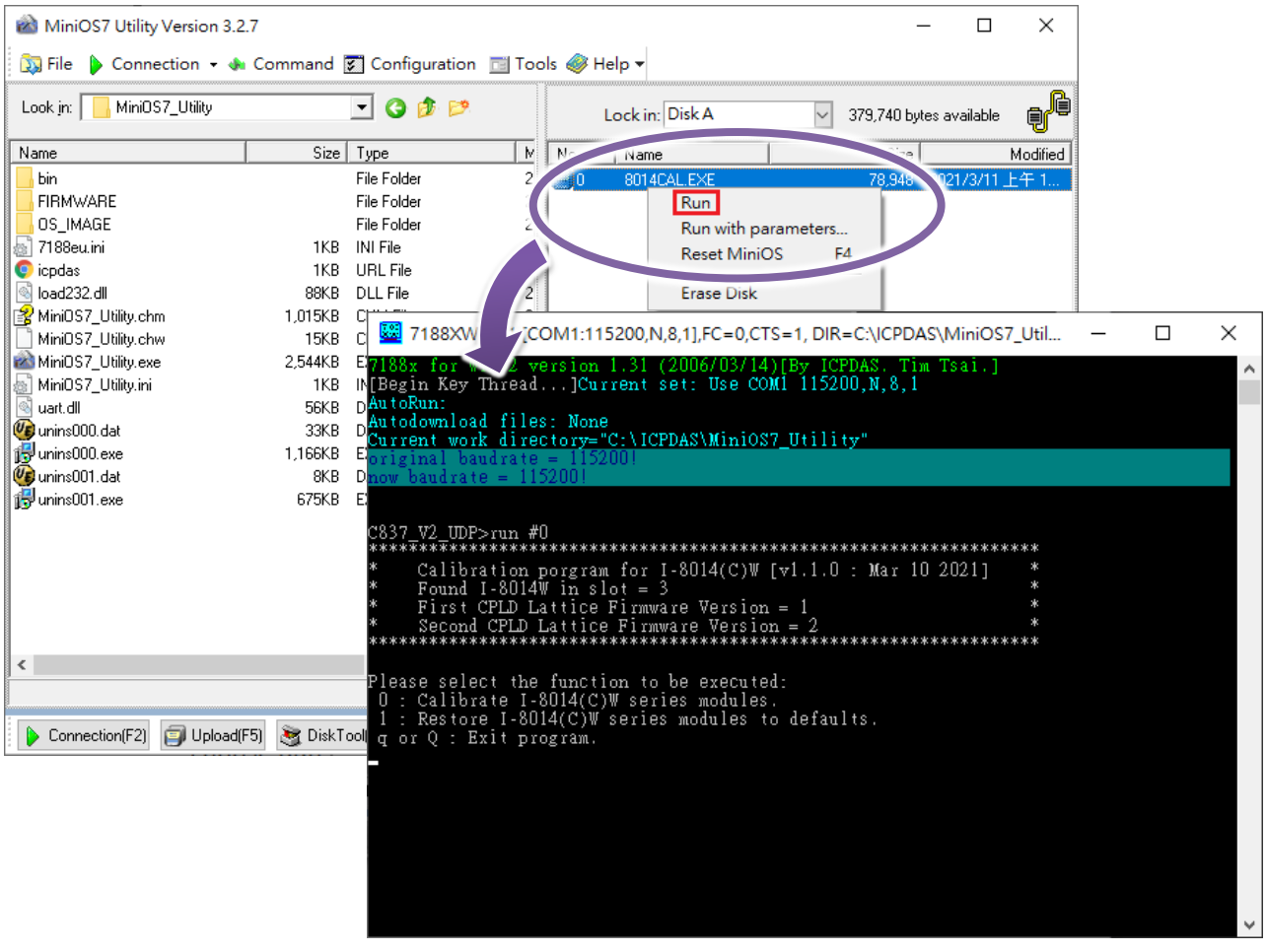
The calibration program can be downloaded in ICP DAS website.

Please refer to the following link:

https://www.icpdas.com/en/download/show.php?num=2897

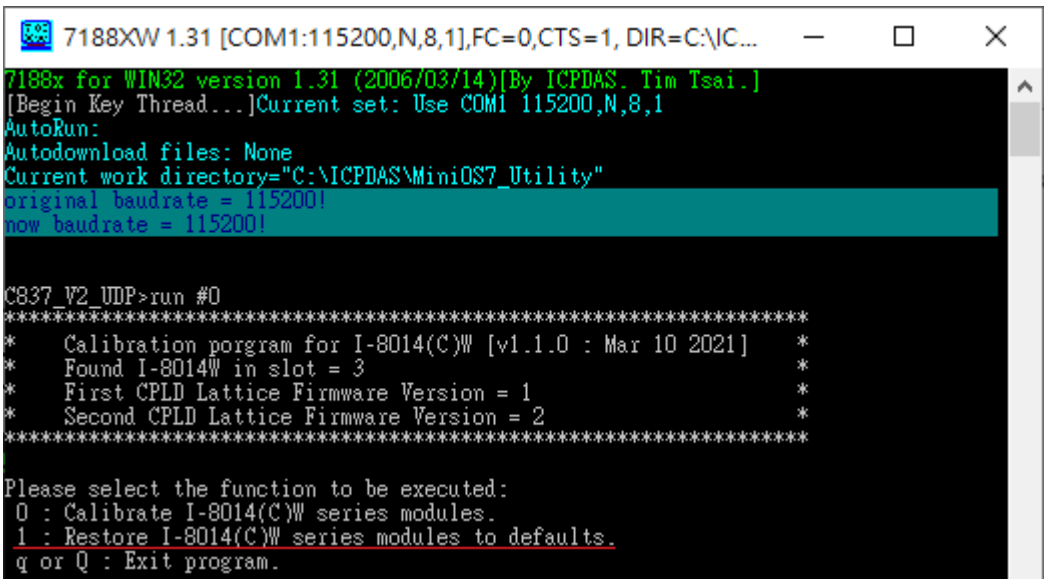
In order to upload programs, please refer to the "2.2.2. Installing the MiniOS7" and “2.5.2. Uploading and Executing iPAC-8000 programs” of the IP-8000 user manual to download MiniOS7 Utility which can help users to upload programs and learn how to operate.

After uploading the calibration program, right click on it and click "Run" to execute it.

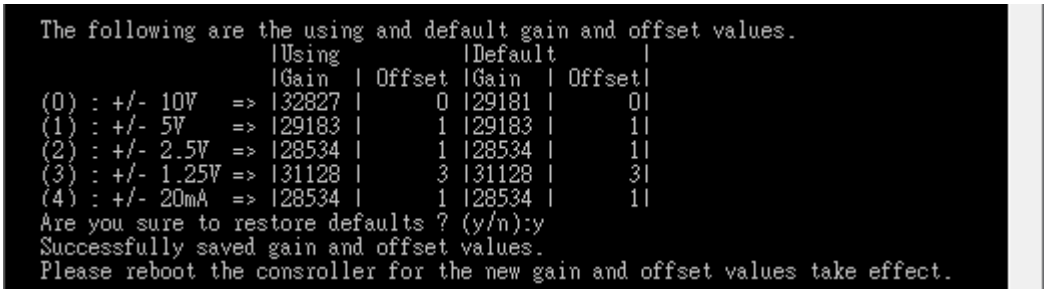


Step 2. Restore defaults

After execute the program, press ‘1’ to restore defaults.



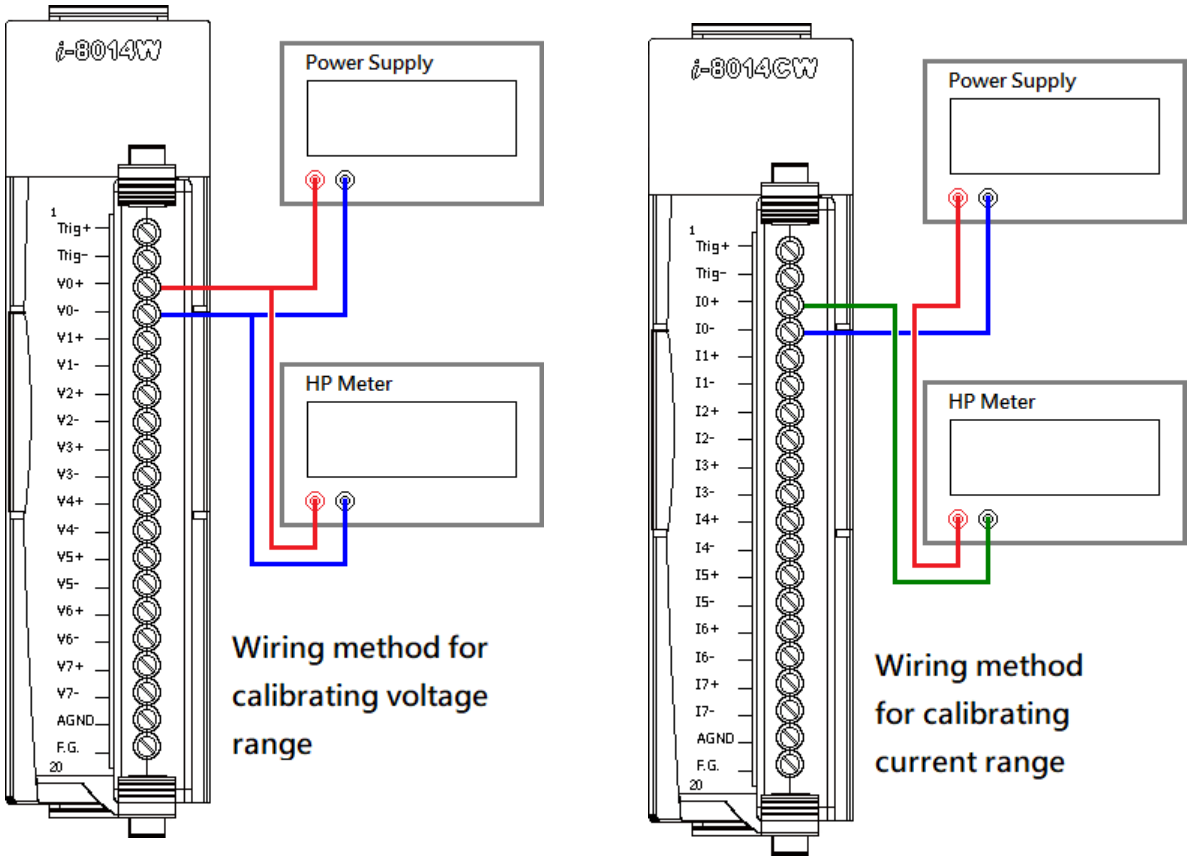
Press ‘y’ to restore.



5.3.3. Calibrate the I-8014(C)W on WinCE and WES units

Step 1. Wiring method

Set the Differential/Single-ended jumper to the differential position, connect source and modules in differential mode and connect the voltage or current meter to the wiring, like the following figure:



Then, plug I-8014(C)W into the controller.

Step 2. Download and execute calibration program

The calibration program can be downloaded in ICP DAS website.

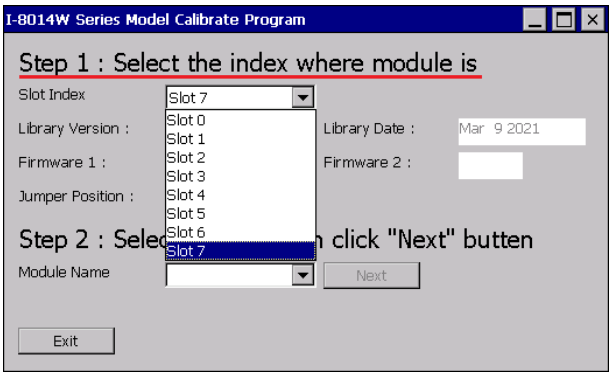
Please refer to the following link:

https://www.icpdas.com/en/download/show.php?num=2897

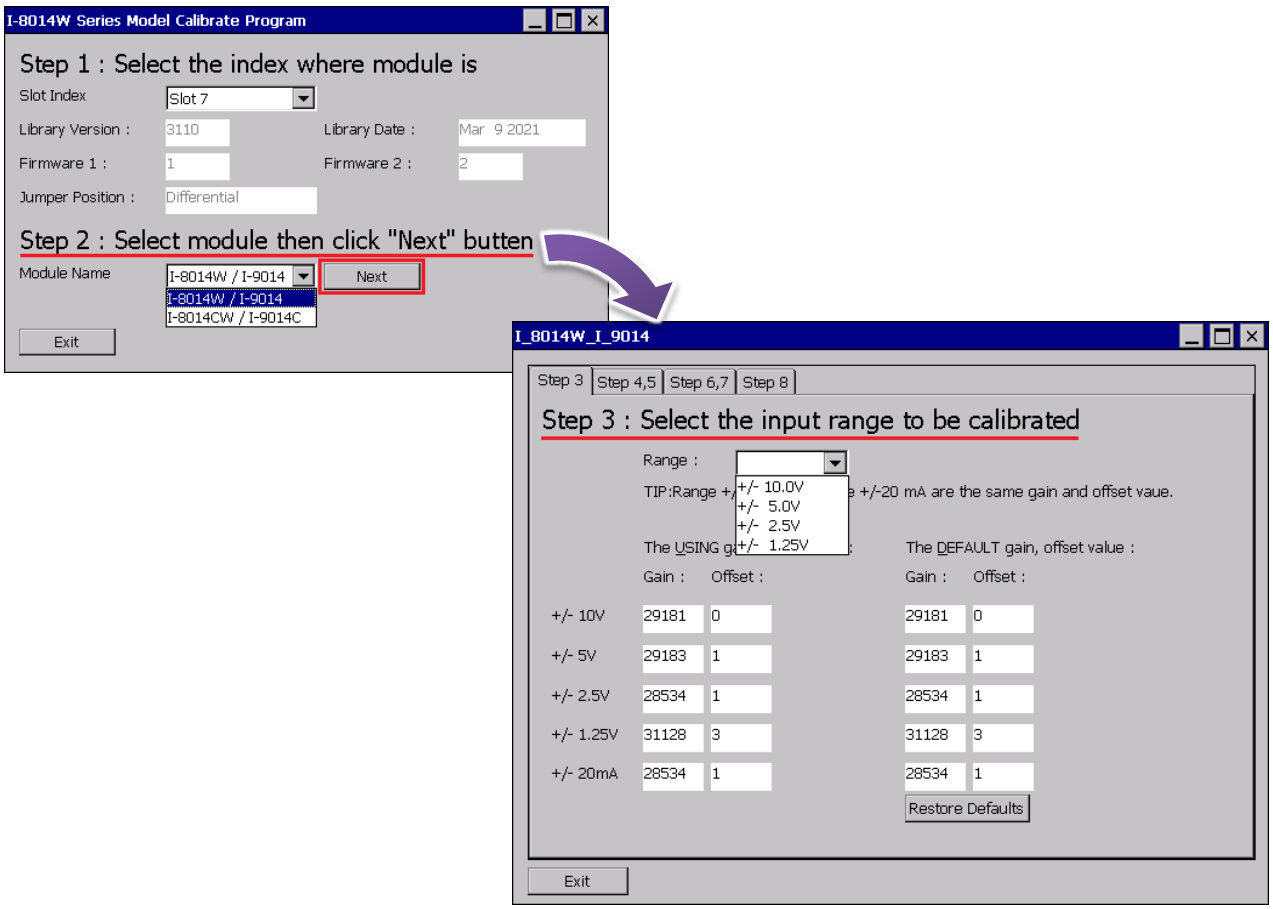
Step 3. Calibrate

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

Select the index where the module is.

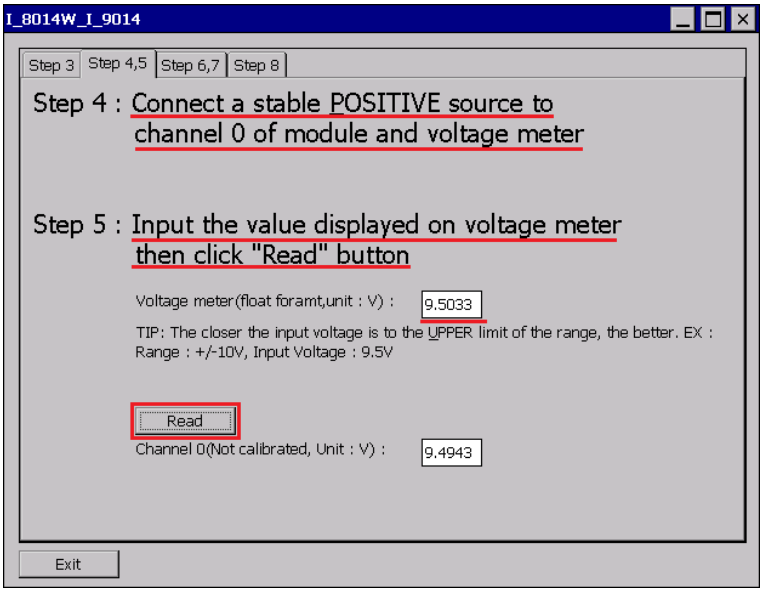


Select the name of module and click “Next” button.



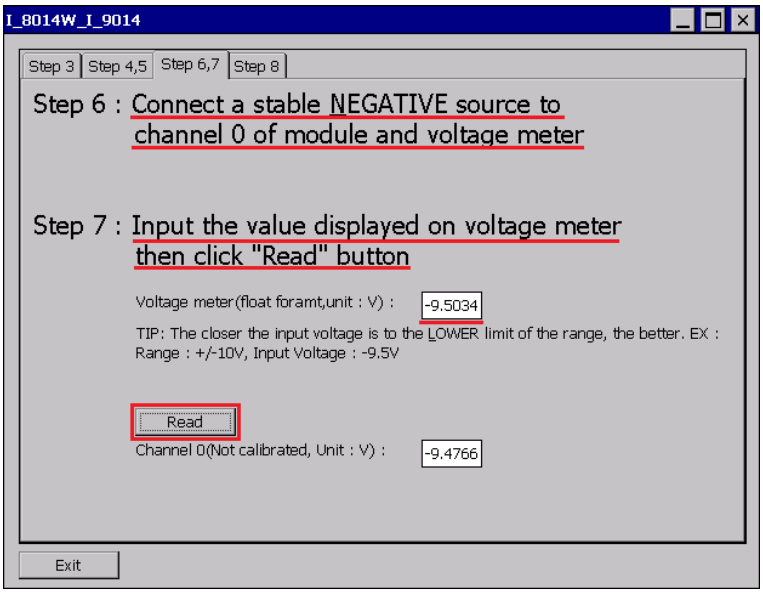
Change to “page Step 4,5”,

Output stable positive source to channel 0 and type the value displayed on the voltage meter, then Click “Read” button.



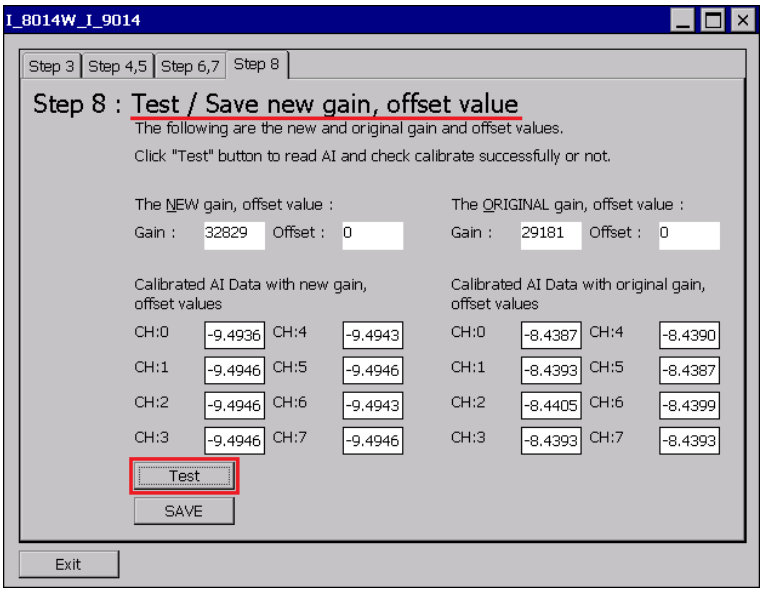
Change to “page Step 6,7”,

Output stable negative source to channel 0 and type the value displayed on the voltage meter, then Click “Read” button.

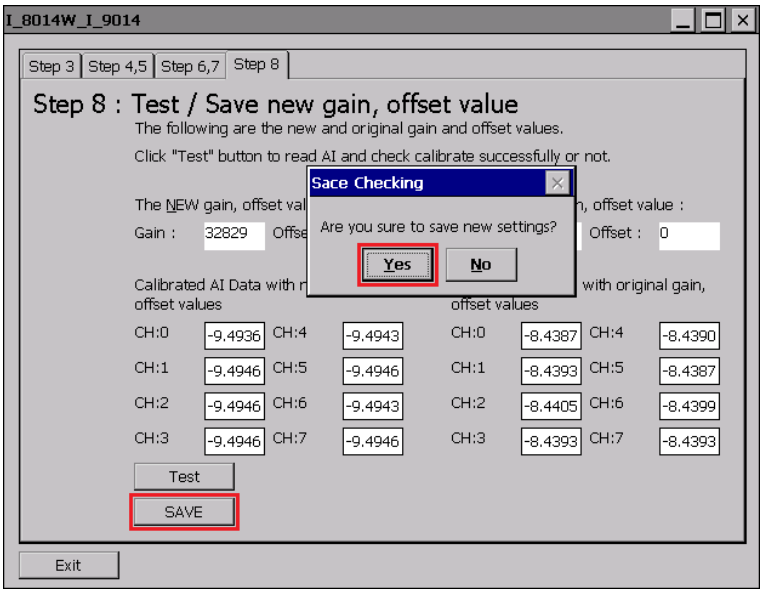


Change to “page Step 8”,

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.



5.3.4. Restore I-8014(C)W to defaults on WinCE and WES units

Step 1. Download and execute calibration program

The calibration program can be downloaded in ICP DAS website.

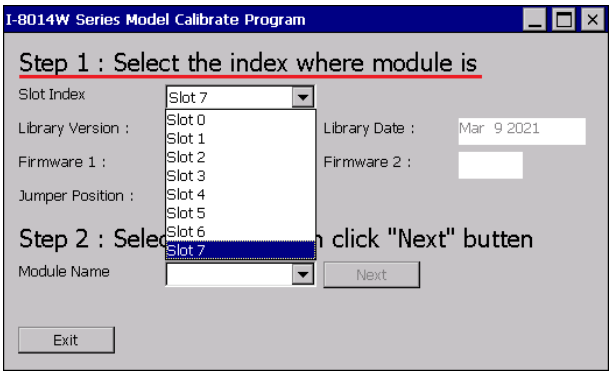
Please refer to the following link:

https://www.icpdas.com/en/download/show.php?num=2897

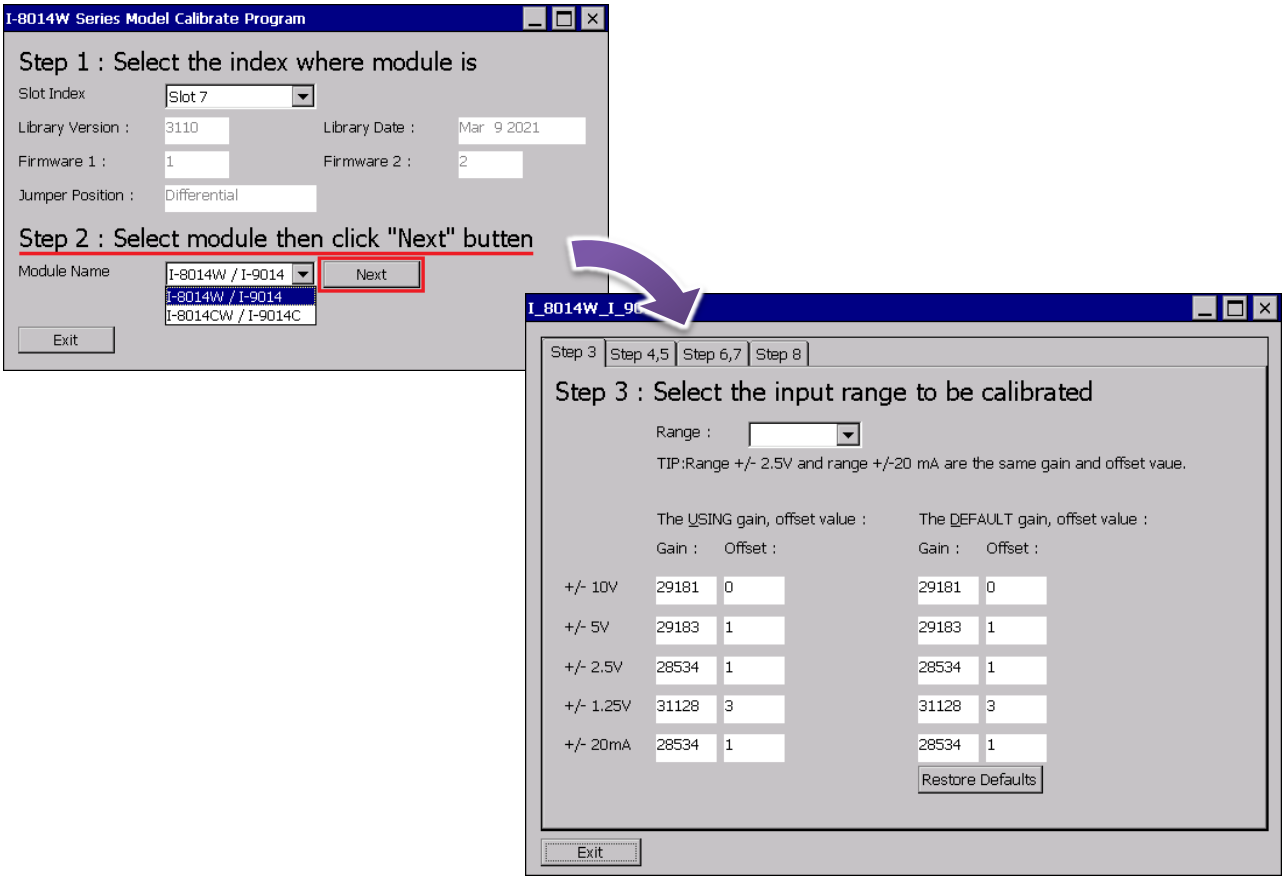
Step 2. Restore defaults

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

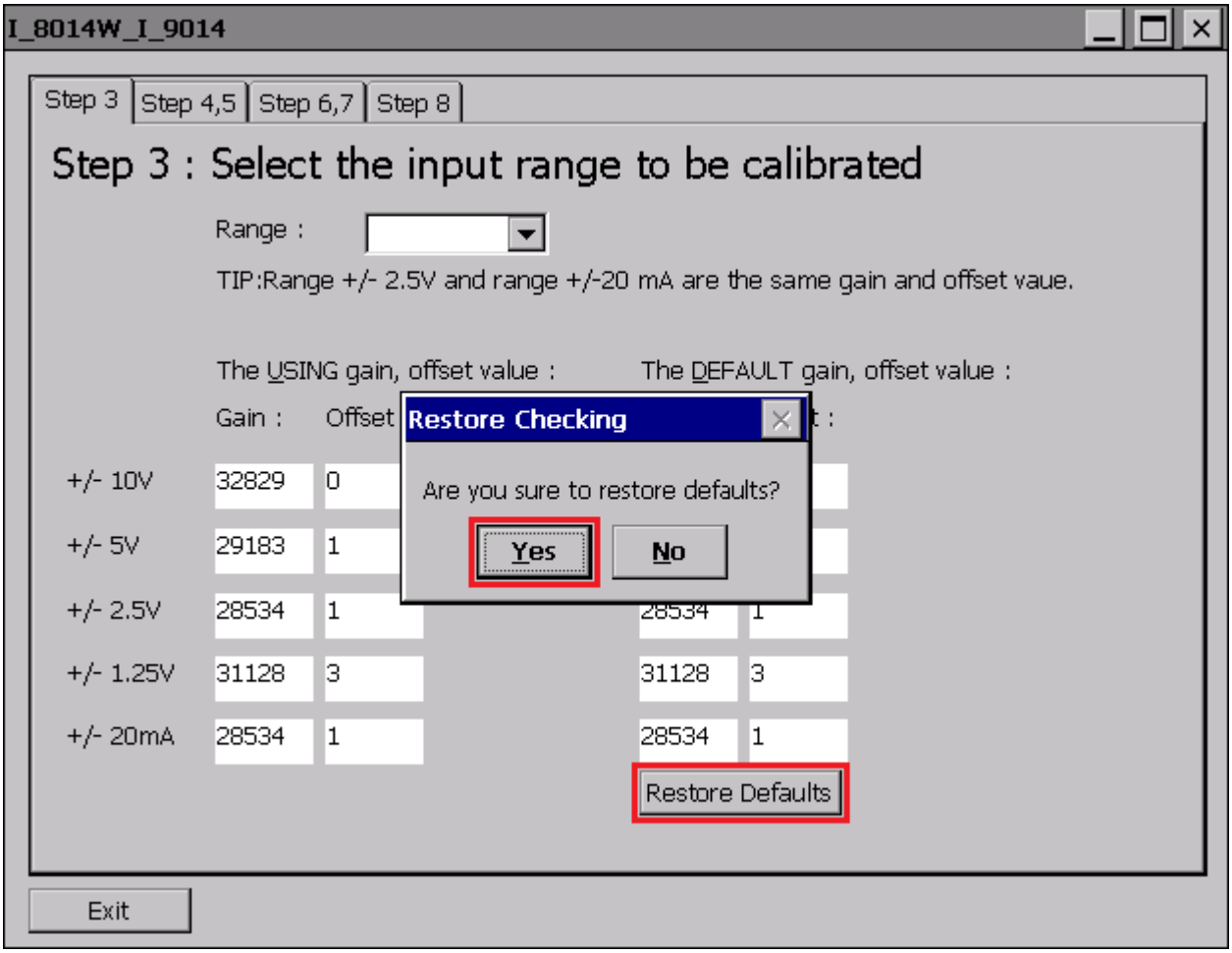
Select the index where the module is.



Select the name of module and click “Next” button.



Click “Restore Defaults” button.



Appendix A. Error Code

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	SLOT_ERROR	Parameter of slot is out of range.
-3	CHANNEL_ERROR	Parameter of channel index is out of range.
-4	GAIN_ERROR	Parameter of gain is out of range.
-5	FIFO_EMPTY	There is no data in the FIFO buffer
-6	FIFO_LATCHED	The FIFO buffer is full and has been latched
-7	FIFO_OVERFLOW	The FIFO buffer is full
-8	TX_NOTREADY	There was an error between primary FPGA and secondary FPGA
-9	FIFO_ISR_ERROR	There was an error when FIFO sent an interrupt signal.
-10	Average_Sample_Rate_DIFF_Error	Parameter of sample rate is out of range in different mode.
-11	Average_Sample_Rate_SING_Error	Parameter of sample rate is out of range in single-end mode.
-12	Average_Level_Error	Parameter of average level is out of range.
-100	REG_SLOT_ISR_ERROR	There was an error when install ISR function.

Appendix B. Revision History

This chapter provides revision history information to this document.

The table below shows the revision history.

Revision	Date	Description
1.0.0	Jan 2018	Initial issue
2.0.0	MAY 2018	• Added content for the I-9014/I-9014C modules • Modify Quick start • Modify Magic Scan • Added API naming table • Added Applicable Platform table • Modify library , demo path • Added API “i8014W_ReadFIFO_BlockMode”,”i8014W_ReadFIFO_InISR”
2.0.2	MAR 2021	• Modify Quick start • Modify Magic Scan • Modify library , demo path • Added APIs of Average function. • Added optioning modules on Linux platform.