I-8017/I-9017 Series

I/O Module User Manual

V3.0.3 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

Table of Contents **3**

Preface **6**

1. Introduction **7**

1.1. Specifications 10

1.1.1. I-8017HW/I-8017HCW/I-8017DW 10

1.1.2. I-9017/I-9017-15/I-9017C-15 12

1.2. Pin Assignments 14

1.2.1. I-8017HW 14

1.2.2. I-8017HCW 15

1.2.3. I-8017DW 16

1.2.4. I-9017 17

1.2.5. I-9017-15 18

1.2.6. I-9017C-15 19

1.3. Jumper Settings 20

1.3.1. Single-Ended/Differential Jumper Selection 20

1.3.2. Input Impedance Jumper Selection 21

1.3.3. Voltage/Current Measurement Jumper Selection 22

1.4. Wire Connections 23

1.4.1. I-8017HW/I-9017-15 23

1.4.2. I-8017DW/I-8017HCW/I-9017 24

1.4.3. I-9017C-15 24

1.5. Block Diagram 25

1.5.1. I-8017HW/I-8017DW/I-8017HCW 25

1.5.2. I-9017/I-9017-15 25

1.5.3. I-9017C-15 26

2. Quick Start 27

2.1. MiniOS7-based Controllers 27

2.1.1. Basic Function 27

2.2. Windows-based Controllers 30

2.2.1. Basic Function 30

2.3. Linux-based Controllers 32

3. API introduction 33

3.1. i8017H_Init / pac_i8017HW_Init 36

3.2. i8017H_GetFirmwareVersion / pac_i8017HW_GetFirmwareVersion 38

3.3. i8017H_GetLibVersion / pac_i8017HW_GetLibVersion 40

3.4. i8017H_GetLibDate / pac_i8017HW_GetLibDate 42

3.5. i8017H_GetSingleEndJumper / pac_i8017HW_GetSingleEndJumper 43

3.6. i8017H_ReadAI / pac_i8017HW_ReadAI 45

3.7. i8017H_ReadAI_AVG / pac_i8017HW_ReadAI_AVG 49

3.8. i8017H_ReadAIHex / pac_i8017HW_ReadAIHex 52

3.9. i8017H_ReadAIHex_AVG / pac_i8017HW_ReadAIHex_AVG 57

3.10. i8017H_ReadGainOffset_Info / pac_i8017HW_ReadGainOffset_Info 61

3.11. i8017H_Read_mA_GainOffset / pac_i8017HW_Read_mA_GainOffset 64

3.12. i8017H_Select_SingleEnd / pac_i8017HW_Select_SingleEnd 67

3.13. i8017H_Get_D_Sub_Status / pac_i8017HW_Get_D_Sub_Status 69

4. Calibration 71

4.1. Calibrate I-8017HW series modules on iPAC-8000 72

4.2. Restore I-8017HW series modules to defaults on iPAC-8000 76

4.3. Calibrate the I-8017HW series modules on WinCE and WES units 78

4.4. Restore I-8017HW series modules to defaults on WinCE and WES units 82

5. Troubleshooting 84

5.1. Verifying Analog Input functionality on a WinCE or WES PAC device 85

5.2. Service Request Requirements 89

5.3. What to do when the data read from the module seems unstable 90

Appendix A. Error Code 91

Appendix B. Read AI Function Performance 92

Appendix B. Revision History 93

Preface

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

- Chapter 1, "Introduction" – This chapter provides information related to the hardware, such as the specifications, jumper settings and wiring.
- Chapter 2, "Quick Start" – This chapter provides information on how to get started, an overview of the location of the demo programs.
- Chapter 3, "API introduction" –This chapter describes the functions provided in the I-8017HW library together with an explanation of the differences in the naming rules used for the MiniOS7 and Windows platforms.
- Chapter 4, "Calibration" – This chapter describes the calibration process for I-8017HW series modules on MiniOS7 and Windows platforms.
- Chapter 5, "Troubleshooting" – This chapter provides some troubleshooting solutions should you encounter any problems while operating the I-8017HW.

1. Introduction

I-8017W/I-8017HCW/I-8017DW/I-9017/I-9017-15/I-9017C-15 (Hereinafter referred to as I-8017HW series modules) are high performance analog input modules, up to 16-channel single-ended or 8-channel differential inputs. It features 14-bit resolution, 100Ks/s sampling rates.

I-8017HW series modules can be used to measure voltage and current sources, except for I-9017C-15.

Applications

- High speed data acquisition systems
- Process monitoring and control
- Vibration analysis
- Digital pattern generator from the digital I/O port

The following table shows the differences between I-8017HW series modules.

Items		I-8017HW	I-8017HCW	I-8017DW
Channels		16-ch Single-ended/ 8-ch Differential		
Range	Voltage	±10 V, ±5 V, ±2.5 V, ±1.25 V		
	Current	+/- 20mA (Requires External 125 Ω Resistor)	+/-20 mA (Requires external 125 Ω resistor in single-ended wire method, or jumper selectable in differential method)	
Dimensions (W x L x H, unit: mm)		30 x 115 x 102	30 x 114 x 85	

The following table shows the differences between I-9017 series modules.

Items		I-9017	I-9017-15	I-9017C-15
Channels		16-ch Single-ended 8-ch Differential	30-ch Single-ended 15-ch Differential	15-ch Differential
Range	Voltage	±10 V, ±5 V, ±2.5 V, ±1.25 V		n/a
	Current	+/- 20mA (Requires external 125 Ω resistor in single-ended wire method, or jumper selectable in differential method)	+/- 20mA (Requires External 125 Ω Resistor)	+/- 20mA

Applicable Platform table

The following table shows which platform the module applies to.

Platform	OS	Module
XPAC	XP-8000(WES)	I-8017HW/I-8017DW/I-8017HCW
	XP-8000-Atom (WES)	I-8017HW/I-8017DW/I-8017HCW
	XP-8000-WES7 (WES7)	I-8017HW/I-8017DW/I-8017HCW
	XP-8000-CE6 (WinCE 6.0)	I-8017HW/I-8017DW/I-8017HCW
	XP-8000-Atom-CE6 (WinCE 6.0)	I-8017HW/I-8017DW/I-8017HCW
	XP-9000-WES7(WES7)	I-9017/I-9017-15/I-9017C-15
WPAC	WP-8000 (CE 5.0/7.0)	I-8017HW/I-8017DW/I-8017HCW
	WP-9000-CE7 (CE 7.0)	I-9017/I-9017-15/I-9017C-15
LinPAC	LinPAC-8000 (Linux kernel 2.6~4.4)	I-8017HW/I-8017DW/I-8017HCW
	LP-9000/LX-9000 (Linux kernel 3.2/4.4)	I-9017/I-9017-15/I-9017C-15
IPAC	iPAC-8000 (MiniOS7)	I-8017HW/I-8017DW/I-8017HCW
	I-8000 (MiniOS7)	I-8017HW/I-8017DW/I-8017HCW

I-8017DW module is equipped with a D-sub connection, meaning that it can be connected using a 37-pin D-sub Connector, as shown in the image below:

For more detailed information regarding 37-pin D-sub Connectors refer to the models indicated in the table below:

Model	Description
DN-37-A	I/O Connector Block with DIN-Rail Mounting and 37-pin D-sub Connector (Pitch: 5.08 mm)
DN-37-381-A	I/O Connector Block with DIN-Rail Mounting and 37-pin D-sub Connector (Pitch: 3.81 mm)
CA-3705A	Male-Female D-sub Cable 0.5 m
CA-3710A	Male-Female D-sub Cable 1 m
CA-3715A	Male-Female D-sub Cable 1.5 m

1.1. Specifications

1.1.1. I-8017HW/I-8017HCW/I-8017DW

Model	I-8017HW	I-8017HCW	I-8017DW
Analog Input			
Channels	8-ch Differential/16-Single-ended		
Voltage Input Range	±1.25, ±2.5, ±5 V, ±10 V		
Current Input Range	±20 mA (Requires External 125 Ω Resistor)	±20 mA (Jumper Select)	
Resolution	14-bit		
Sample Rate	Single Channel Polling Mode :90K S/s Single Channel Interrupt Mode: 50K S/s 8 channel Scan Mode : 16 K S/s		
Accuracy	0.1% of FSR		
Zero Drift	± 0.1 uV/°C		
Span Drift	± 10 ppm/°C		
Input Impedance	20 K, 200 K, 20 M (Jumper Select)		
Input Bandwidth	100 KHz		
LED Indicators			
System LED Indicator	1 LED as Power Indicator		
I/O LED Indicator	16 LEDs as User defined Indicators		
EMS Protection			
ESD Protection	±4 kV Contact for each Terminal		

Model	I-8017HW	I-8017HCW	I-8017DW
Isolation			
Intra-module Isolation, Field-to-Logic	2500 Vrms		
Power			
Power Consumption	2 W Max.		
Mechanical			
Dimension (L x W x H, unit: mm)	30 x 115 x 102		30 x 114 x 85
Environment			
Operating Temperature	-25 °C ~ +75°C		
Storage Temperature	-30 °C ~ +80°C		
Humidity	10% ~ 90% RH, non-condensing		

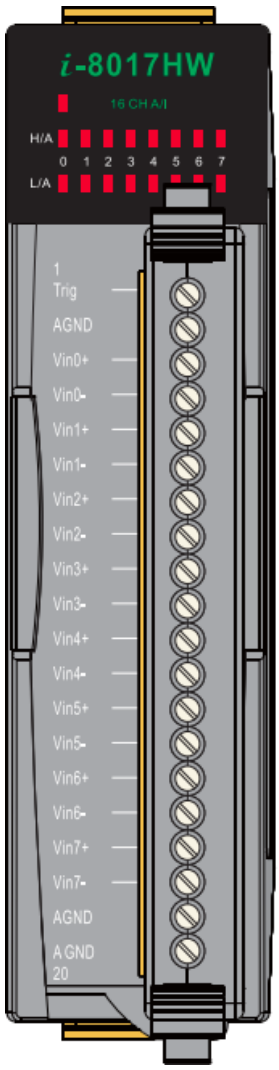
1.1.2. I-9017/I-9017-15/I-9017C-15

Model	I-9017	I-9017-15	I-9017C-15
Analog Input			
Channels	8 Differential/ 16 Single-ended	15 Differential/ 30 Single-ended	15 Differential
Voltage Input Range	±1.25, ±2.5, ±5 V, ±10 V		-
Current Input Range	±20 mA (Requires OptionalExternal 125 Ω Resistor)		±20 mA
Resolution	14-bit		
Sample Rate	Single Channel Polling Mode :900K S/s Single Channel Interrupt Mode: 50K S/s 15-channel Scan Mode : 16 K S/s		
Accuracy	0.1% of FSR		
Input Bandwidth	100 KHz		
Zero Drift	± 0.1 uV/°C		
Span Drift	± 10 ppm/°C		
Input Impedance	20 K, 200 K, 20 M (Jumper Select)		125 ohm
LED Indicators			
System LED Indicator	1 LED as Power Indicator		
I/O LED Indicator	--		
EMS Protection			
ESD (IEC 61000-4-2)	±4 kV Contact for each Terminal ±8 kV Air for Random Point		

Model	I-9017	I-9017-15	I-9017C-15
Isolation			
Intra-module Isolation, Field-to-Logic	2500 Vrms		
Power			
Power Consumption	2 W Max.		
Mechanical			
Dimension (L x W x H)	144 mm x 30.3 mm x 134 mm		
Environment			
Operating Temperature	-25 °C ~ +75°C		
Storage Temperature	-40°C ~ +85°C		
Humidity	10 % ~ 90% RH, non-condensing		

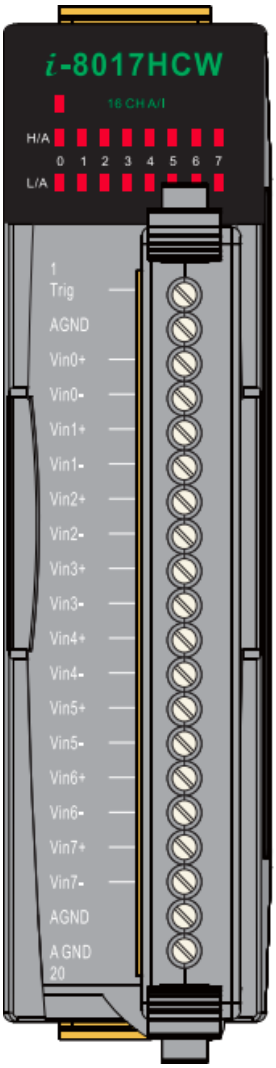
1.2. Pin Assignments

1.2.1. I-8017HW



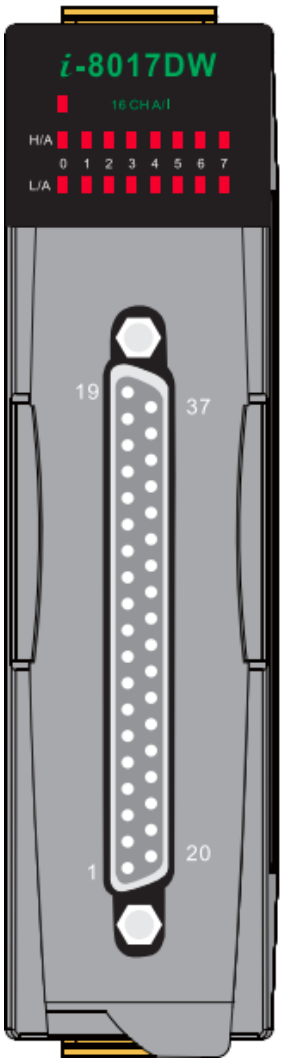
Terminal No.	Pin Assignment	
	Differentia	Single-ended
01	Trig	Trig
02	AGND	AGND
03	Vin0+	Vin0
04	Vin0-	Vin8
05	Vin1+	Vin1
06	Vin1-	Vin9
07	Vin2+	Vin2
08	Vin2-	Vin10
09	Vin3+	Vin3
10	Vin3-	Vin11
11	Vin4+	Vin4
12	Vin4-	Vin12
13	Vin5+	Vin5
14	Vin5-	Vin13
15	Vin6+	Vin6
16	Vin6-	Vin14
17	Vin7+	Vin7
18	Vin7-	Vin15
19	AGND	AGND
20	AGND	AGND

1.2.2. I-8017HCW



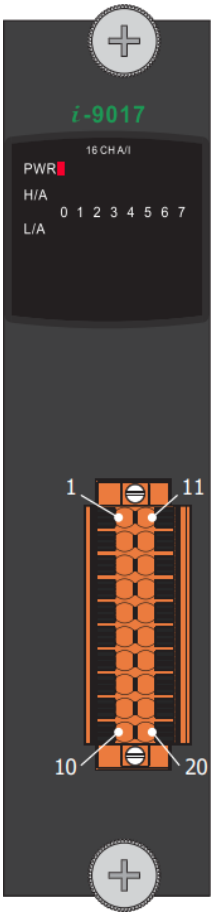
Terminal No.	Pin Assignment	
	Differentia	Single-ended
01	Trig	Trig
02	AGND	AGND
03	Vin0+	Vin0
04	Vin0-	Vin8
05	Vin1+	Vin1
06	Vin1-	Vin9
07	Vin2+	Vin2
08	Vin2-	Vin10
09	Vin3+	Vin3
10	Vin3-	Vin11
11	Vin4+	Vin4
12	Vin4-	Vin12
13	Vin5+	Vin5
14	Vin5-	Vin13
15	Vin6+	Vin6
16	Vin6-	Vin14
17	Vin7+	Vin7
18	Vin7-	Vin15
19	AGND	AGND
20	AGND	AGND

1.2.3. I-8017DW



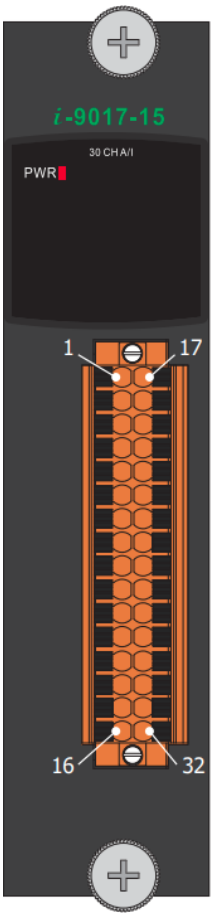
Pin Assignment		Terminal No.		Pin Assignment	
Differentia	Single-ended			Differentia	Single-ended
AGND	AGND	19	37	BK Sensor	BK Sensor
Trig	Trig	18	36	-	-
AI7-	AI15	17	35	-	-
AI7+	AI17	16	34	-	-
AI6-	AI14	15	33	-	-
AI6+	AI6	14	32	-	-
AI5-	AI13	13	31	-	-
AI5+	AI5	12	30	-	-
AI4-	AI12	11	29	-	-
AI4+	AI4	10	28	-	-
AI3-	AI11	09	27	-	-
AI3+	AI3	08	26	-	-
AI2-	AI10	07	25	-	-
AI2+	AI2	06	24	-	-
AI1-	AI9	05	23	-	-
AI1+	AI1	04	22	-	-
AI0-	AI8	03	21	AGND	AGND
AI0+	AI0	02	20	AGND	AGND
BK Sensor	BK Sensor	01			

1.2.4. I-9017



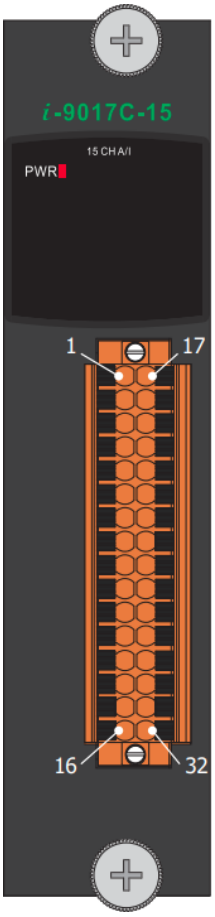
Pin Assignment		Terminal No.		Pin Assignment	
Differential	Single-ended			Differential	Single-ended
Trig+	Trig+	1	11	Trig-	Trig-
V0+(I0+)	Vin0	2	12	V0-(I0-)	Vin8
V1+(I1+)	Vin1	3	13	V1-(I1-)	Vin9
V2+(I2+)	Vin2	4	14	V2-(I2-)	Vin10
V3+(I3+)	Vin3	5	15	V3-(I3-)	Vin11
V4+(I4+)	Vin4	6	16	V4-(I4-)	Vin12
V5+(I5+)	Vin5	7	17	V5-(I5-)	Vin13
V6+(I6+)	Vin6	8	18	V6-(I6-)	Vin14
V7+(I7+)	Vin7	9	19	V7-(I7-)	Vin15
AGND	AGND	10	20	AGND	AGND

1.2.5. I-9017-15



Pin Assignment		Terminal No.		Pin Assignment	
V0+	Vin0	01	17	V0 -	Vin15
V1+	Vin1	02	18	V1 -	Vin16
V2+	Vin2	03	19	V2 -	Vin17
V3+	Vin3	04	20	V3 -	Vin18
V4+	Vin4	05	21	V4 -	Vin19
V5+	Vin5	06	22	V5 -	Vin20
V6+	Vin6	07	23	V6 -	Vin21
V7+	Vin7	08	24	V7 -	Vin22
V8+	Vin8	09	25	V8-	Vin23
V9+	Vin9	10	26	V9 -	Vin24
V10+	Vin10	11	27	V10 -	Vin25
V11+	Vin11	12	28	V11 -	Vin26
V12+	Vin12	13	29	V12 -	Vin27
V13+	Vin13	14	30	V13 -	Vin28
V14+	Vin14	15	31	V14 -	Vin29
AGND	AGND	16	32	AGND	AGND

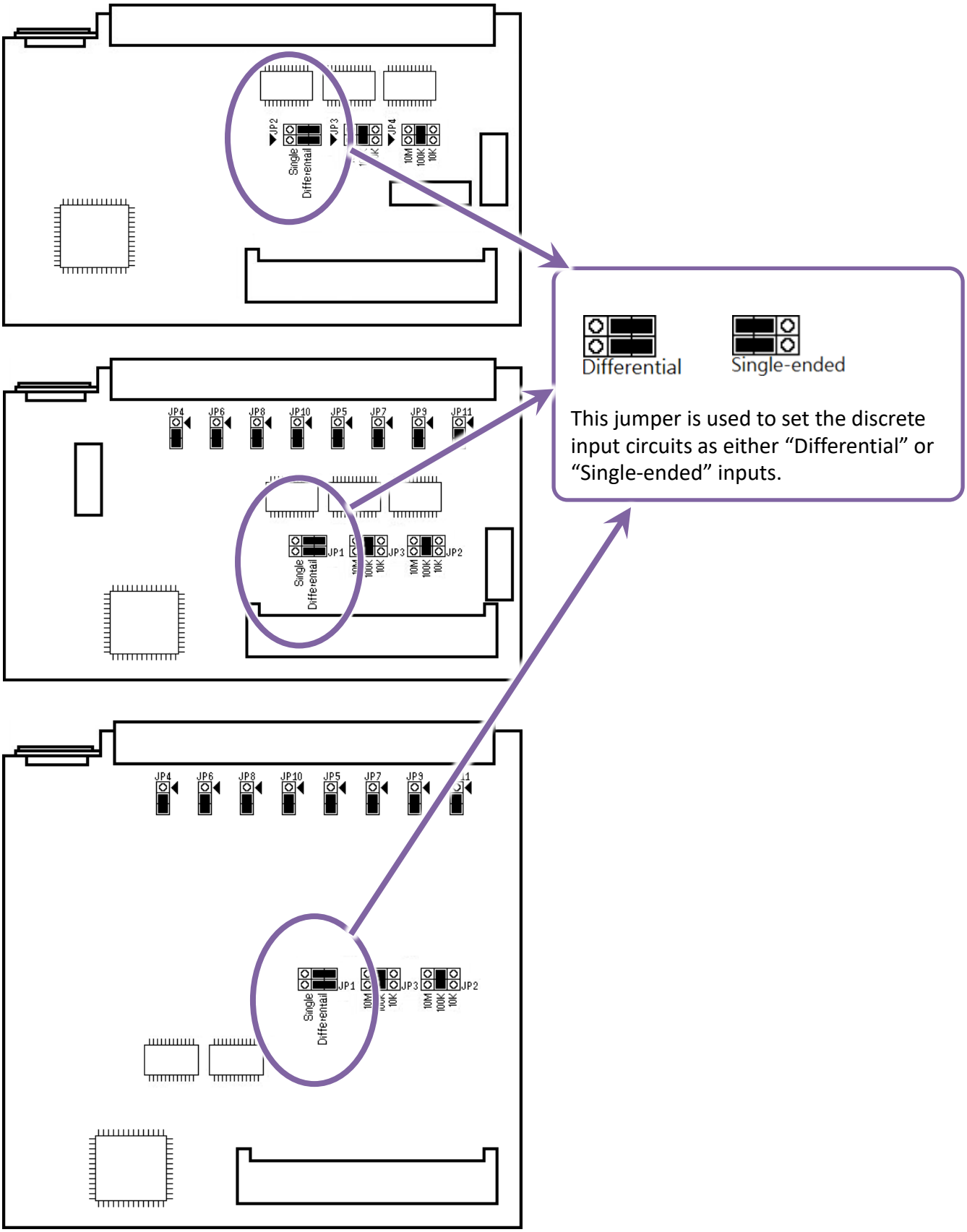
1.2.6. I-9017C-15



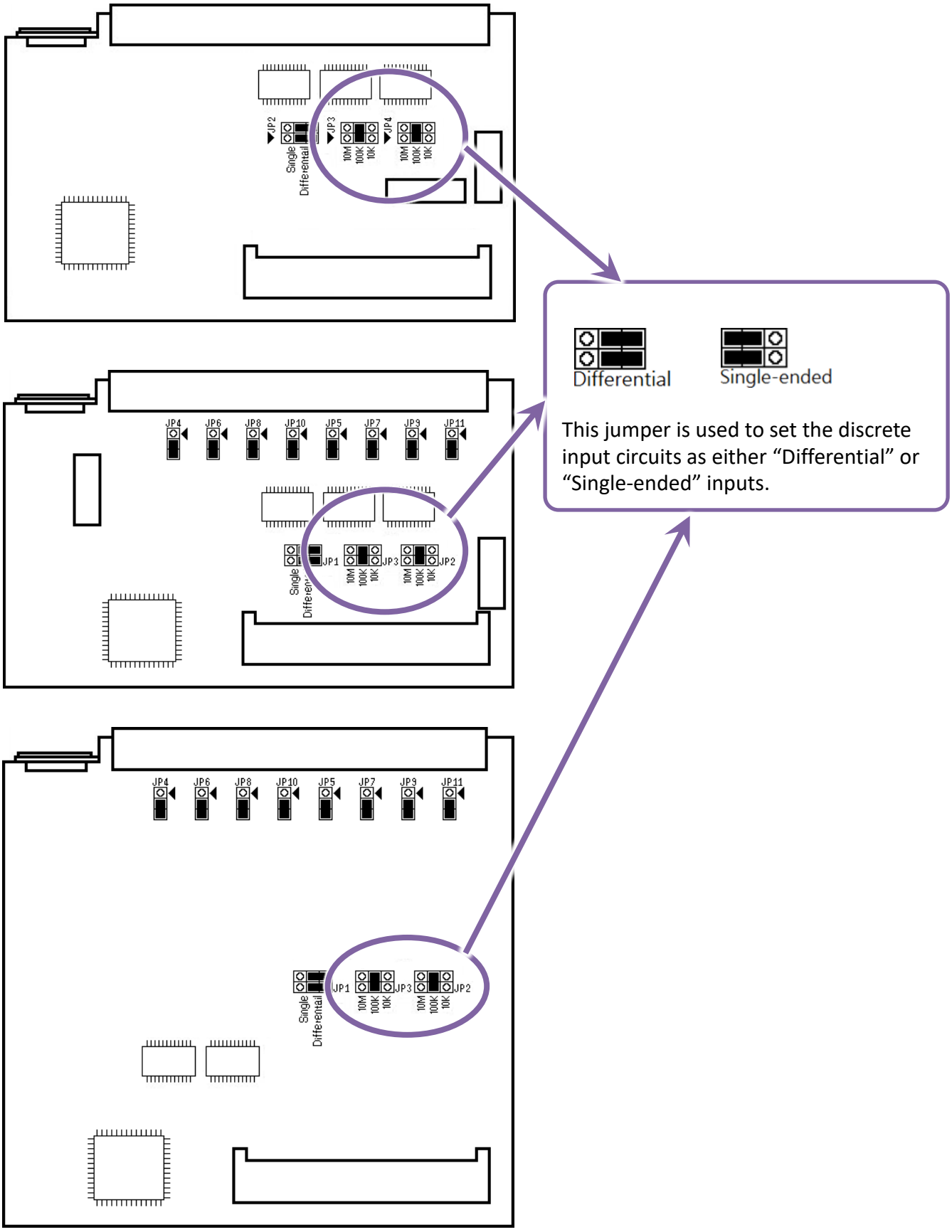
Pin Assignment	Terminal No.		Pin Assignment
I0+	01	17	I0 -
I1+	02	18	I1 -
I2+	03	19	I2 -
I3+	04	20	I3 -
I4+	05	21	I4 -
I5+	06	22	I5 -
I6+	07	23	I6 -
I7+	08	24	I7 -
I8+	09	25	I8-
I9+	10	26	I9 -
I10+	11	27	I10 -
I11+	12	28	I11 -
I12+	13	29	I12 -
I13+	14	30	I13 -
I14+	15	31	I14 -
AGND	16	32	AGND

1.3. Jumper Settings

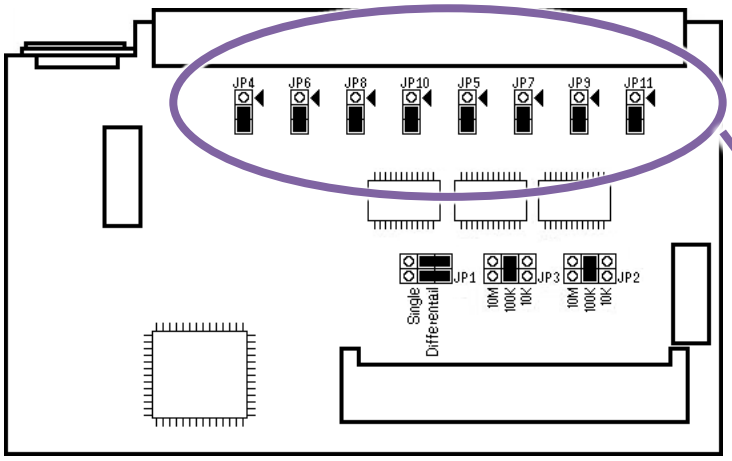
1.3.1. Single-Ended/Differential Jumper Selection

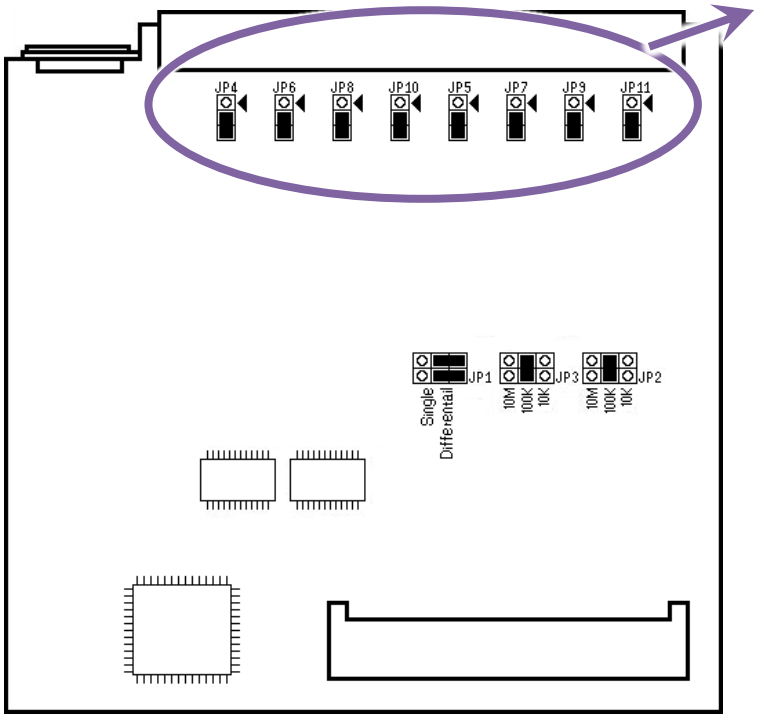


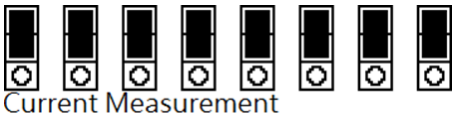
1.3.2. Input Impedance Jumper Selection

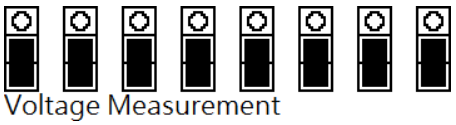


1.3.3. Voltage/Current Measurement Jumper Selection









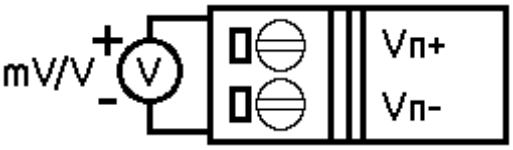
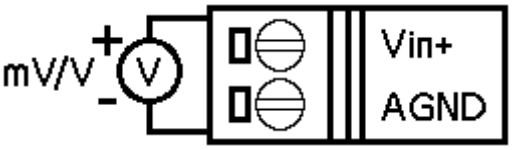
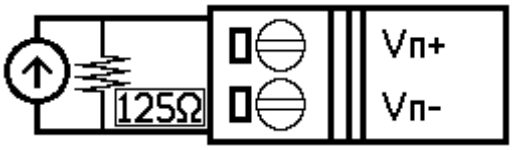
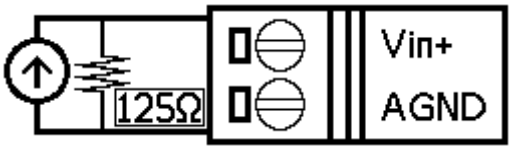
By default, I-8017HCW module is configured for current measurement.

I-8017DW is configured for voltage measurement.

I-9017 is configured for voltage measurement.

1.4. Wire Connections

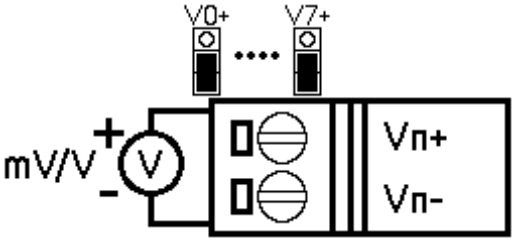
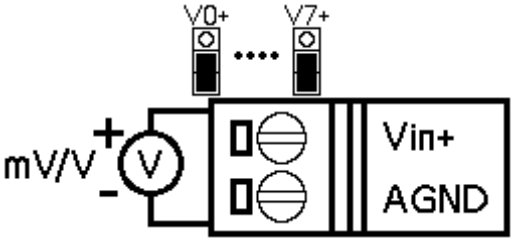
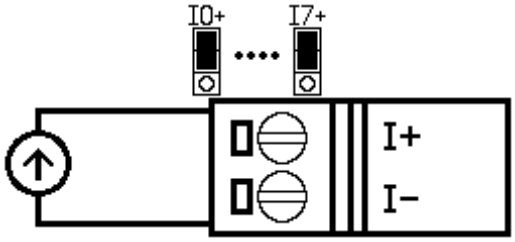
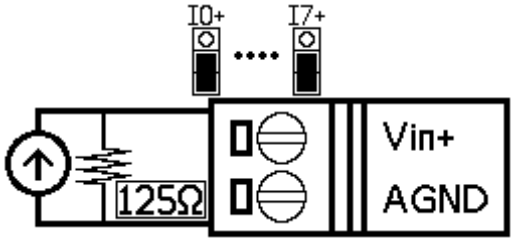
1.4.1. I-8017HW/I-9017-15

Input Type	Differential	Single-ended
Jumper Position		
Voltage		
Current		

Tips & Warnings

When connecting to a current source, an optional external 125 Ω resistor is required.

1.4.2. I-8017DW/I-8017HCW/I-9017

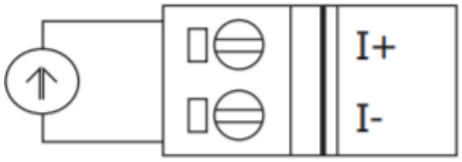
Input Type	Differential	Single-ended
Jumper Position		
Voltage		
Current		

Tips & Warnings

Differential Input Type: Current Input Wiring need to jumper at current input.

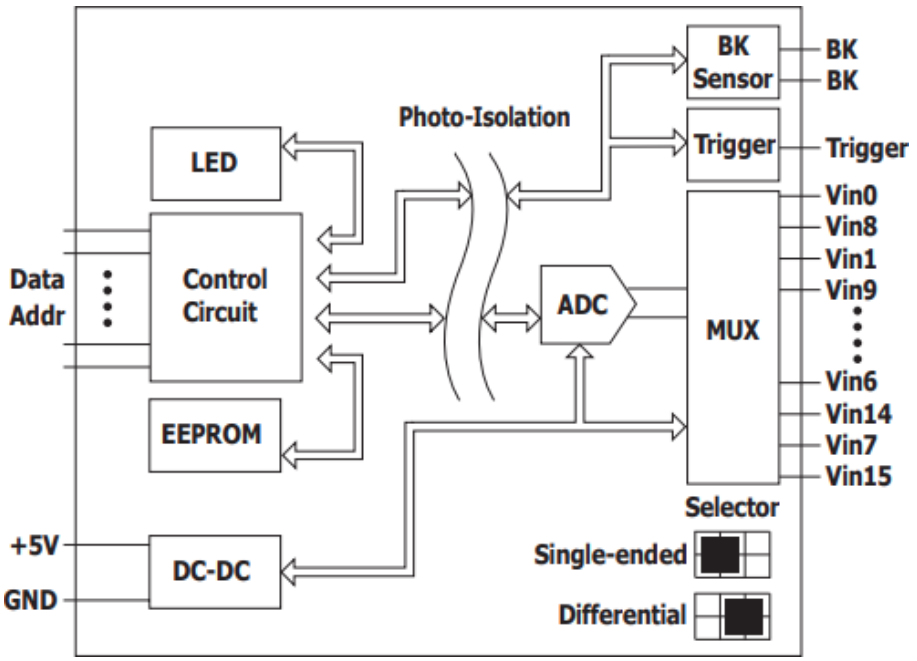
Single-ended Input Type: Current Input Wiring need to jumper at voltage input, an options external 125 Ω resistor is required.

1.4.3. I-9017C-15

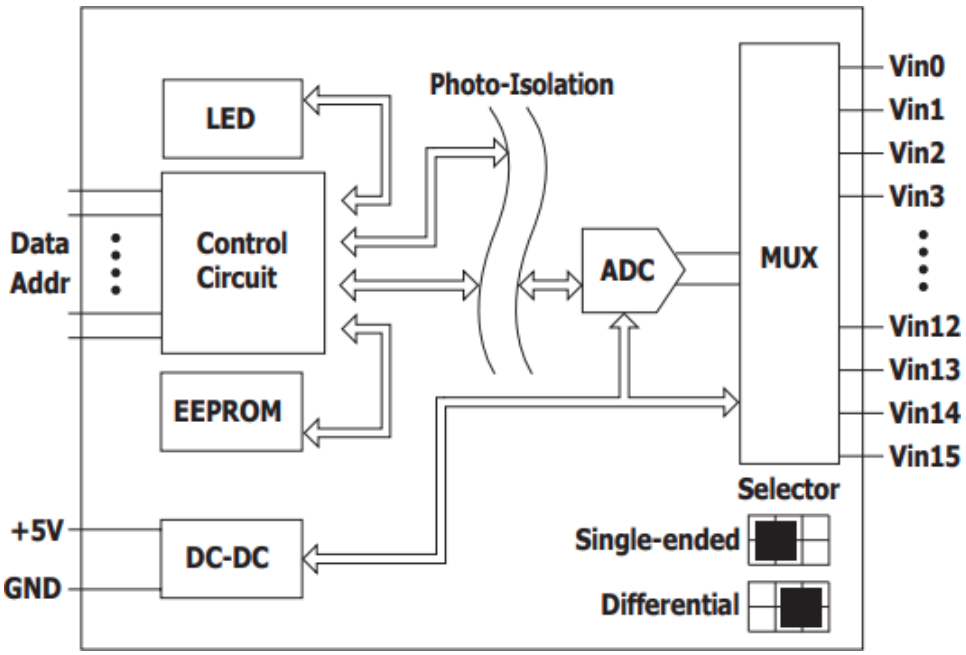
	Current Input Wiring
Differential	

1.5. Block Diagram

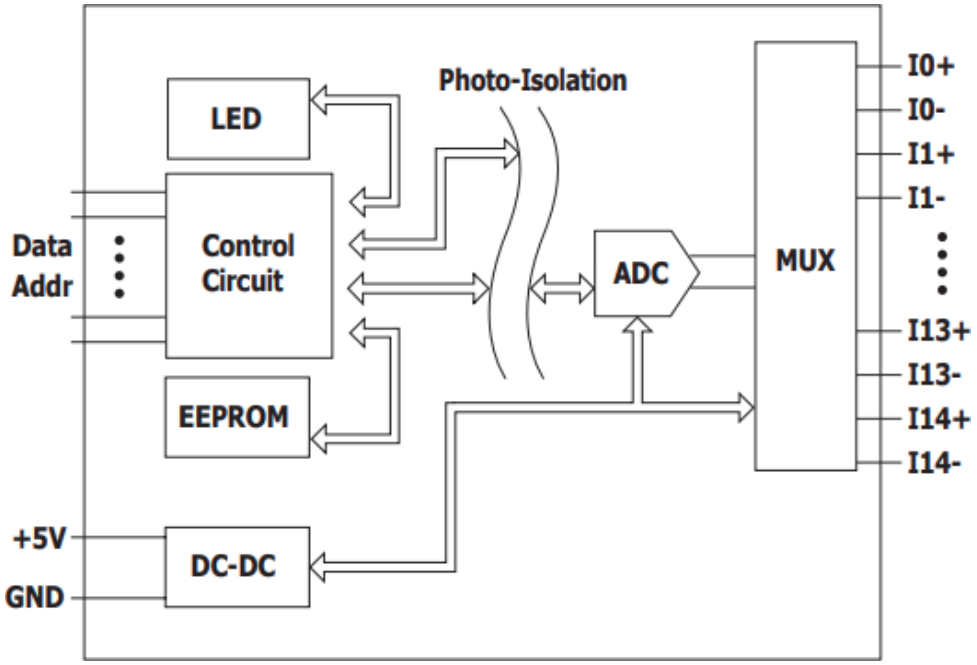
1.5.1. I-8017HW/I-8017DW/I-8017HCW



1.5.2. I-9017/I-9017-15



1.5.3. I-9017C-15



2. Quick Start

This chapter will be accompanied by demos to explain how to implement functions such as read AI and calibration process

Demos can be downloaded in the following link:

https://www.icpdas.com/en/download/show.php?num=2323

2.1. MiniOS7-based Controllers

2.1.1. Basic Function

Basic function can be used to retrieve configuration information and verify the AI reading function.

Basic 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.

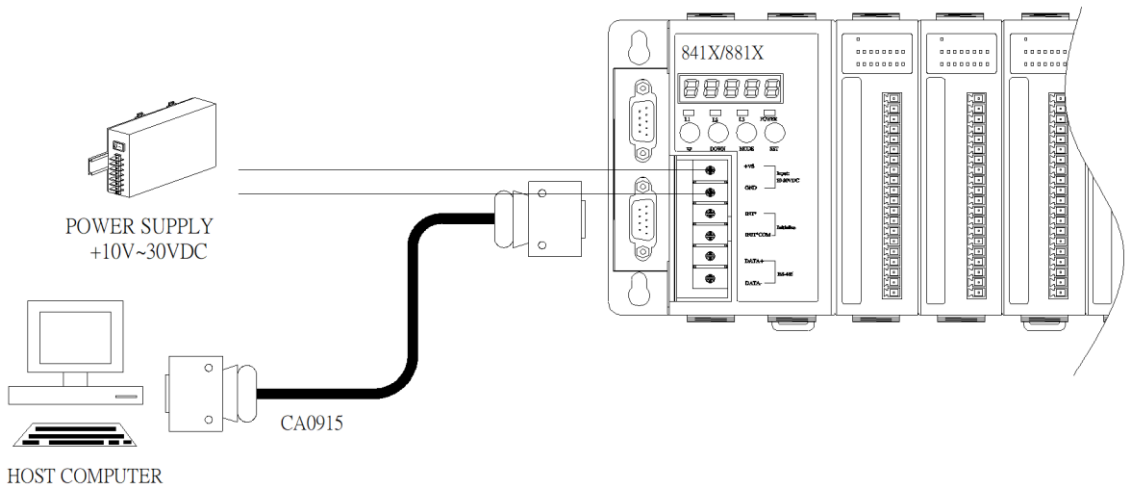
The following steps take Base_Info.exe as example and display the information of I-8017HW.

Step 1: Please refer to the "Wire Connections", connect a stable signal source (such as a battery) to I-8017HW

Tips & Warnings

1. Unused channels should be connected to GND to avoid floating.
2. A battery output should provide a stable enough signal.
3. A 125 Ohm resistor is required when measuring current input.

Step 2: Plug I-8017HW in the MiniOS7 controller, connect the power supply to the unit and connect the unit to the Host PC by RS-232 cable

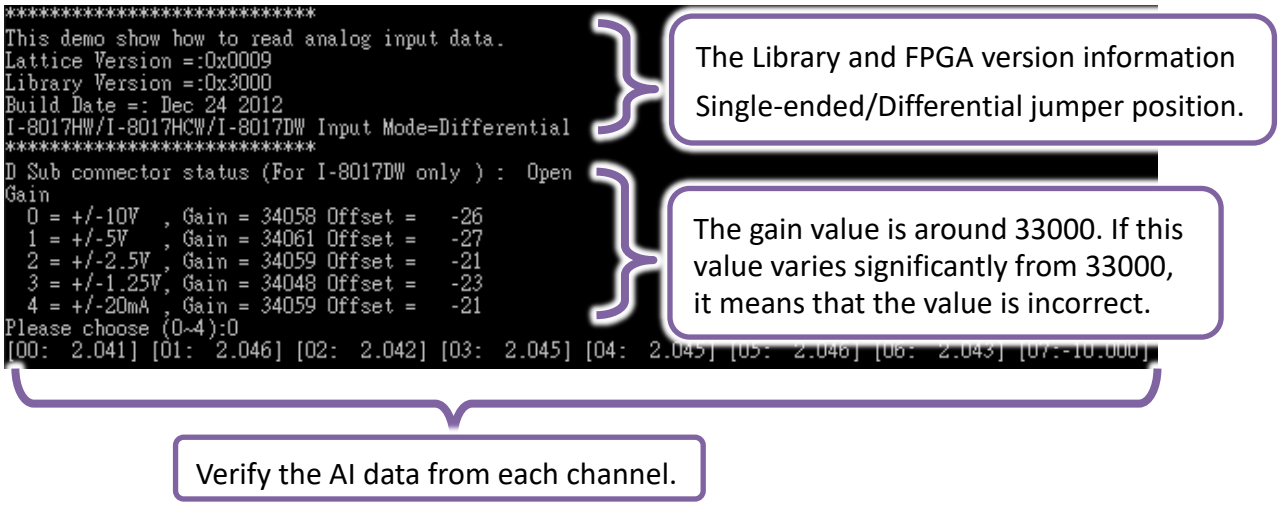


Step 3: Run 7188xw.exe on the host PC and open the COM Port which is connecting to the MiniOS7 Controller

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



2.2. Windows-based Controllers

2.2.1. Basic Function

Basic function can be used to retrieve configuration information and verify the AI reading function.

Basic 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.

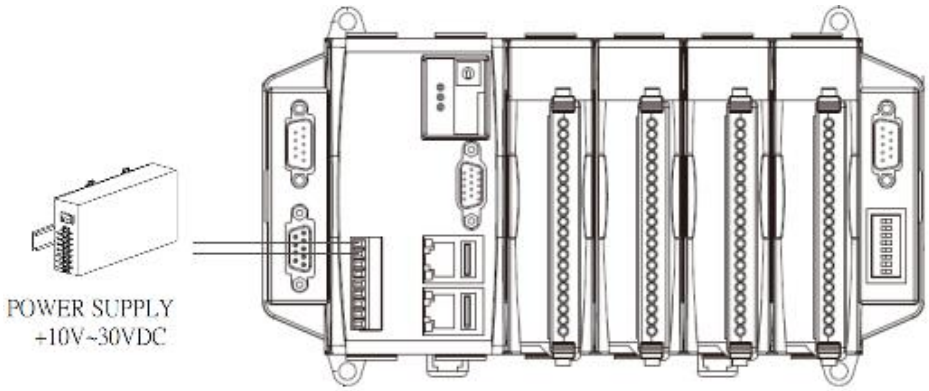
The following steps take pac_i8017HW_Basic_Info.exe as example and display the information of I-8017HW.

Step 1: Please refer to the “Wire Connections”, connect a stable signal source (such as a battery) to I-8017HW

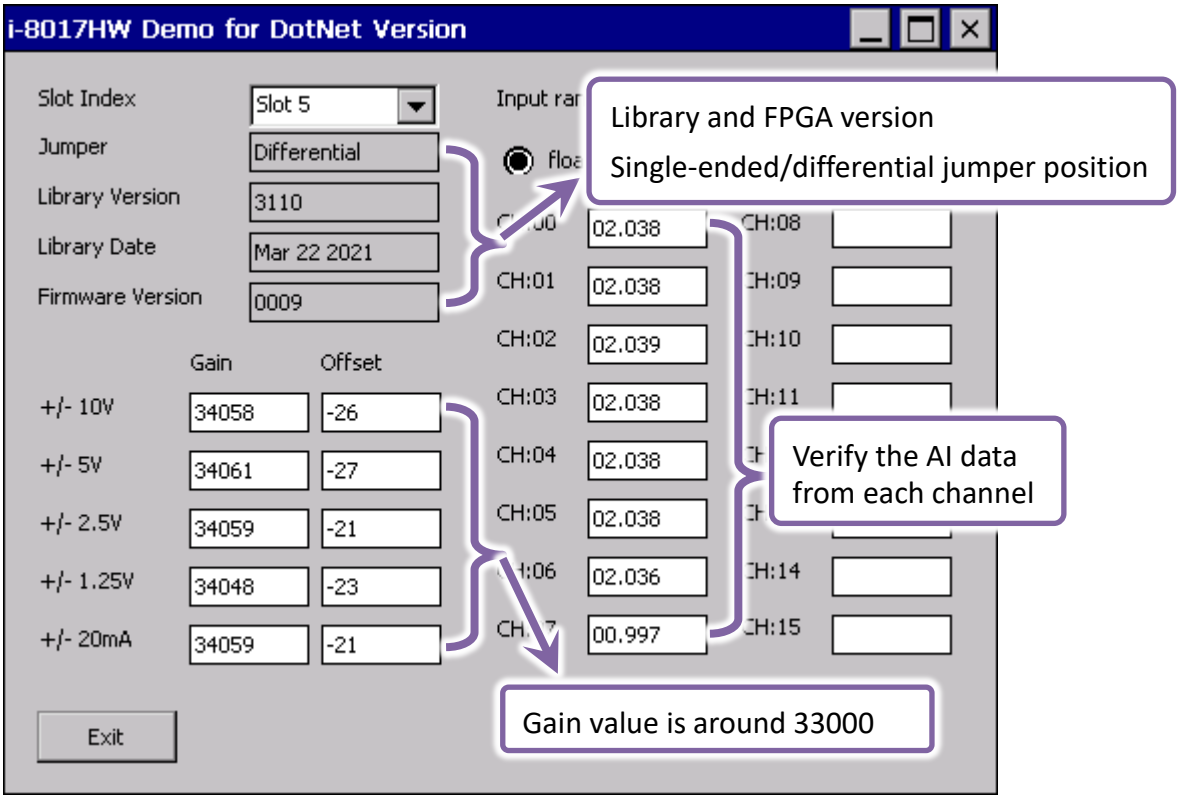
Tips & Warnings

1. Unused channels should be connected to GND to avoid floating.
2. A battery output should provide a stable enough signal.
3. A 125 Ohm resistor is required when measuring current input.

Step 2: Plug I-8017HW in the Windows-based controller and connect the power supply to the unit.



Step 3: Run pac i8017HW Basic Info.exe on the controller and verify that basic information and AI data read from each channel are correct, as shown in the figure below:

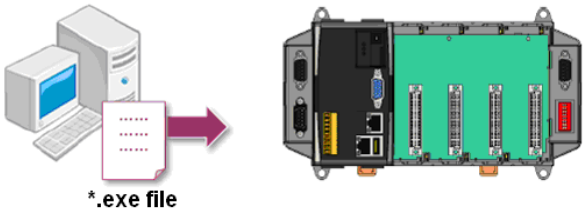


2.3. Linux-based Controllers

Basic function can be used to retrieve configuration information and verify the AI reading function.

Basic 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.



Please follow the steps below to learn how to use I-8017HW series modules.

Step 1: Users need to download LinPAC SDK, which is includes GNU tool chain, Libraries, header, examples files, etc.

Step 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-8017W/9017 has been plugged in the controller.

Step 3: Refer to the Jumper Settings section. Ensure that the Differential/Single-ended selection jumper is in the Differential position. Connect a stable signal source to the module (e.g., a battery output) using the differential wiring method.

Step 4: Next, check the communication between controller and PC is fine or not, and download the demo program files to the controller.

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. API introduction

ICPDAS supplies a range of C/C++ API functions for I-8017HW series modules.

When developing a program, refer to either the 8017HW.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=2323

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-8017HW / I-8017HCW /I-8017DW</td><td rowspan="4">“pac_i8017W_ “
+ function name</td><td rowspan="4">“pac8017HWNet.
pac8017HW”
+ function name</td></tr>
<tr><td>Windows CE6</td><td>I-8017HW / I-8017HCW /I-8017DW</td></tr>
<tr><td>Windows CE7</td><td>I-8017HW / I-8017HCW /I-8017DW
I-9017 / I-9017-15 / I-9017C-15</td></tr>
<tr><td>WES</td><td>I-8017HW / I-8017HCW /I-8017DW
I-9017 / I-9017-15 / I-9017C-15</td></tr>
<tr><td>MiniOS7</td><td>I-8017HW / I-8017HCW /I-8017DW</td><td>“i8017W_”
+ function name</td><td rowspan="3">Null</td></tr>
<tr><td rowspan="2">Linux</td><td>I-8017HW / I-8017HCW /I-8017DW</td><td>“i8017W_”
+ function name</td></tr>
<tr><td>I-9017 / I-9017-15 /I-9017C-15</td><td>“I9017_”
+ function name</td></tr>
</table>

The following is an overview of the functions provided in the 8017HW.lib and pac_i8017HW.lib.

API for I-8017HW series and I-9017 series

API	Description
i8017H_Init pac_i8017HW_Init	Used to initialize the module
i8017H_GetFirmwareVersion pac_i8017HW_GetFirmwareVersion	Used to read the firmware (FPGA) version information
i8017H_GetLibVersion pac_i8017HW_GetLibVersion	Used to read the version and build information for the currently installed Library
i8017H_GetLibDate pac_i8017HW_GetLibDate	Used to read the build date information for the currently installed Library
i8017H_GetSingleEndJumper pac_i8017HW_GetSingleEndJumper	Used to read the status of the input jumper (Differential or Single-ended mode)
i8017H_ReadAI pac_i8017HW_ReadAI	Used to read the Analog Input value from a specific channel in float format
i8017H_ReadAI_AVG pac_i8017HW_ReadAI_AVG	Used to read the average Analog input value from a specific channel in float format
i8017H_ReadAIHex pac_i8017HW_ReadAIHex	Used to read the Analog Input value from a specific channel in 16-bit hexadecimal format
i8017H_ReadAIHex_AVG pac_i8017HW_ReadAIHex_AVG	Used to read the average Analog input value from a specific channel in hexadecimal format
i8017H_ReadGainOffset_Info pac_i8017HW_ReadGainOffset_Info	Used to read the calibrated voltage Gain and Offset values

API for I-9017 and I-9017C-15:

API	Description
i8017H_Read_mA_GainOffset pac_i8017HW_Read_mA_GainOffset	Used to read the calibrated current Gain and Offset values

API for I-9017-15

API	Description
i8017H_Select_SingleEnd pac_i8017HW_Select_SingleEnd	Used to set the Single-ended/ differential mode of I-9017-15.

API for I-9017-15

API	Description
i8017H_Select_SingleEnd pac_i8017HW_Select_SingleEnd	Used to set the Single-ended/ differential mode of I-9017-15.

API for I-8017DW

API	Description
i8017H_Get_D_Sub_Status pac_i8017HW_Get_D_Sub_Status	Used to get connector status between D sub and 8017DW.

3.1. i8017H_Init / pac_i8017HW_Init

This function is used to initialize the module and must be called at least once before using any other function.

Syntax

For MiniOS7

```
short i8017HW_Init(int slot);
```

For Windows (CE and WES)

```
short pac_i8017HW_Init(int slot);
```

For Linux

```
short I8017_Init(int slot);                    // for LinPAC-8000
short I9017_Init(int slot);                    // for LinPAC-9000, LX-9000
```

Parameters

slot:

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

Return Values

Refer to Appendix A: “Error Code” for more details.

Examples

[C/C++]

```
int slot;
i8017HW_Init(slot);
```

[C#]

```
int slot;
pac8017HW.Init(slot);
```

[C] (For LinPAC)

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

    I8017_Init(slot);
    Close_Slot(slot);
    return 0;
}
```

3.2. i8017H_GetFirmwareVersion / pac_i8017HW_GetFirmwareVersion

This function is used to read the firmware (FPGA) version information for the module.

Syntax

For MiniOS7

```
short i8017HW_GetFirmwareVersion(int slot, short* firmware);
```

For Windows (CE and WES)

```
short pac_i8017HW_GetFirmwareVersion(int slot, short* firmware);
```

For Linux

```
short I8017_GetFirmwareVersion(int slot, short* firmware);        // for LinPAC-8000
short I9017_GetFirmwareVersion(int slot, short* firmware);        // for LP-9000, LX-9000
```

Parameters

slot

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

**firmware*

[Output]The firmware version information for the I-8017 module.

Return Values

Refer to Appendix A: “Error Code” for more details.

Examples

[C/C++]

```
int slot;
short firmware;
i8017HW_GetFirmwareVersion(slot, &firmware);
```

[C#]

```
int slot;
Int16 firmware = 0;
pac8017HWNet.pac8017HW.FirmwareVersion(slot, ref firmware);
```

[C] (For LinPAC)

```
int main(){
    int slot, ret;
    short firmware;
    ret=Open_Slot(slot);
    if (ret > 0) {
            printf("Open Slot%d failed, return value=%d \n", slot, ret);
            return (-1);
    }
    I8017_Init(slot);
    I8017_GetFirmwareVersion(slot, &firmware);
    Close_Slot(slot);
    return 0;
}
```

3.3. i8017H_GetLibVersion / pac_i8017HW_GetLibVersion

This function is used to read the version and build information for the Library.

Syntax

For MiniOS7

```
short i8017HW_GetLibVersion(void);
```

For Windows (CE and WES)

```
short pac_i8017HW_GetLibVersion(void);
```

For Linux

```
short I8017_GetLibVersion(void);          // for LinPAC-8000
short I9017_GetLibVersion(void);          // for LinPAC-9000, LX-9000
```

Parameters

None

Return Values

The version number.

Others: Refer to Appendix A: “Error Code Definitions” for more details.

Examples

[C/C++]

```
short version;
version = i8017HW_GetLibVersion();
```

[C#]

```
Int16 version;
version = pac8017HWNet.pac8017HW.LibVersion();
```

[C] (For LinPAC)

```
int main(){
    int slot, ret;
    short version;
    ret=Open_Slot(slot);
    if (ret > 0) {
            printf("Open Slot%d failed, return value=%d \n", slot, ret);
            return (-1);
    }
    I8017_Init(slot);
    version = i8017HW_GetLibVersion();
    Close_Slot(slot);
    return 0;
}
```

3.4. i8017H_GetLibDate / pac_i8017HW_GetLibDate

This function is used to read the build date information for the Library.

Syntax

For MiniOS7

```
void i8017HW_GetLibDate(char libDate[]);
```

For Windows (CE and WES)

```
void pac_i8017HW_GetLibDate(char libDate[]);
```

Parameters

libDate[]

A string indicating the build date of the Library.

Return Values

Refer to Appendix A: “Error Code” for more details.

Examples

[C/C++]

```
char date;
i8017HW_GetLibDate(date);
```

[C#]

```
string date;
date = pac8017HWNet.pac8017HW.LibDate();
```

3.5. i8017H_GetSingleEndJumper / pac_i8017HW_GetSingleEndJumper

This function is used to read whether the jumper is set to either Differential or Single-ended mode.

Syntax

For MiniOS7

```
short pac_i8017HW_GetSingleEndJumper(int iSlot, short* selectJumper);
```

For Windows (CE and WES)

```
short pac_i8017HW_GetSingleEndJumper(int iSlot, short* selectJumper);
```

For Linux

```
short I8017_GetSingleEndJumper(int iSlot, short* selectJumper);   // for LinPAC-8000
short I9017_GetSingleEndJumper(int iSlot, short* selectJumper);   // for LP-9000, LX-9000
```

Parameters

iSlot

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

** selectJumper*

[Output] The status of module.

0: Differential Mode

1: Single-ended Mode

Return Values

Refer to Appendix A: “Error Code” for more details.

Examples

[C/C++]

```
int slot, jumper;
i8017HW_GetSingleEndJumper(slot, &jumper);
```

[C#]

```
int slot,jumper;
pac8017HWNet.pac8017HW.SingleEndJumper(slot, ref jumper);
```

[C] (For LinPAC)

```
int main(){
    int slot, jumper, ret;
    ret=Open_Slot(slot);
    if (ret > 0) {
            printf("Open Slot%d failed, return value=%d \n", slot, ret);
            return (-1);
    }
    I8017_Init(slot);
    I8017_GetSingleEndJumper(slot, & jumper);
    Close_Slot(slot);
    return 0;
}
```

Note

The old version LinPAC SDK will show I8017_GetSingleEndJumper(slot) function in demo, if you want to use new I8017_GetSingleEndJumper(slot, & jumper) function, it is necessary to install the latest SDK and recompile your examples.

3.6. i8017H_ReadAI / pac_i8017HW_ReadAI

This function is used to read the Analog Input value in float format from a specific channel of the module.

Syntax

For MiniOS7

```
short i8017HW_ReadAI(int iSlot, int iChannel, int iGain, float* fValue);
```

For Windows (CE and WES)

```
short pac_i8017HW_ReadAI(int iSlot, int iChannel, int iGain, float* fValue);
```

For Linux

```
short I8017_ReadAI(int iSlot, int iChannel, int iGain, float* fValue); // for LinPAC-8000
short I9017_ReadAI(int iSlot, int iChannel, int iGain, float* fValue); // for LP-9000, LX-9000
```

Parameters

iSlot

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

iChannel

Specifies channel number.

For I-8017HW / I-8017HCW / I-8017DW / I-9017 => 0 ~ 8 in differential method; 0 ~ 15 in single-ended method.

For I-9017-15 => 0 ~ 14 in differential method; 0 ~ 29 in single-ended method.

For I-9017C-15 => always 0 ~ 14.

iGain

Specifies the input range: 0: +/- 10.0V 1: +/- 5.0V 2: +/- 2.5V 3: +/- 1.25V 4: +/- 20mA

* *fValue*

[Output] the analog input value in float format.

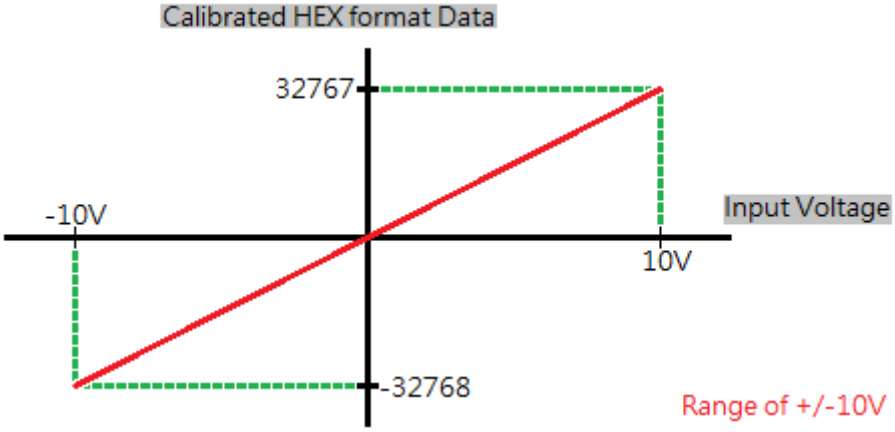
Return Values

Others: Refer to Appendix A: “Error Code” for more details.

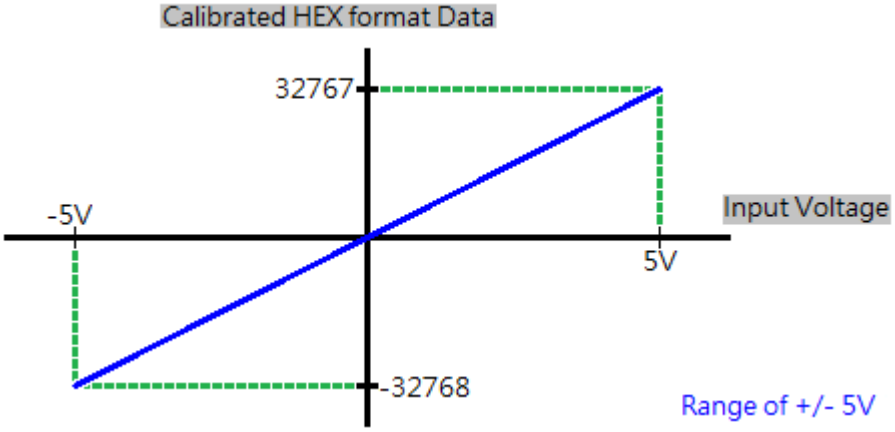
Note

I-8017HW series modules equipped with a 14-bit AD chip.

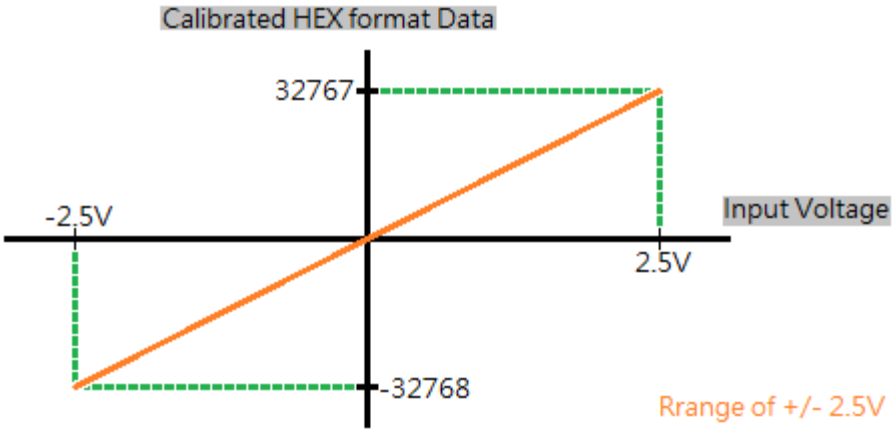
This function will calibrate the data that read from the chip and convert it into 16-bit data. The following pictures show the scale of voltage and data and how to calculate the hexadecimal data into floating data.



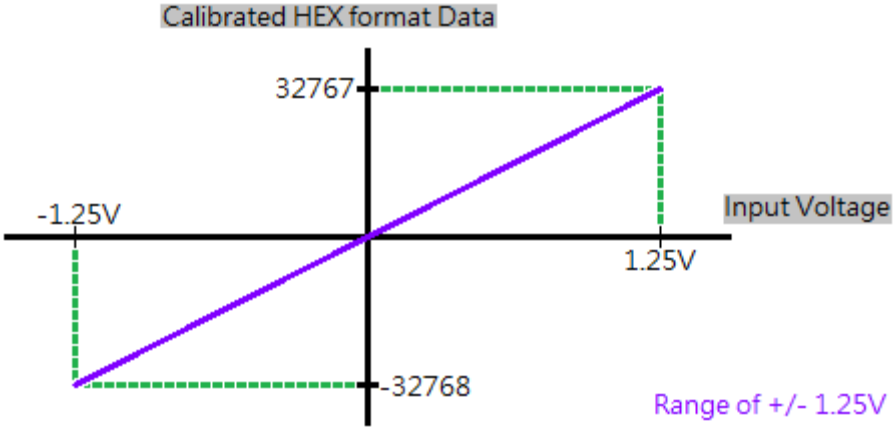
If(hexadecimal data >=0)	**floating data = (hexadecimal data / 32767) * 10**
Else	**floating data = (hexadecimal data / 32768) * 10**



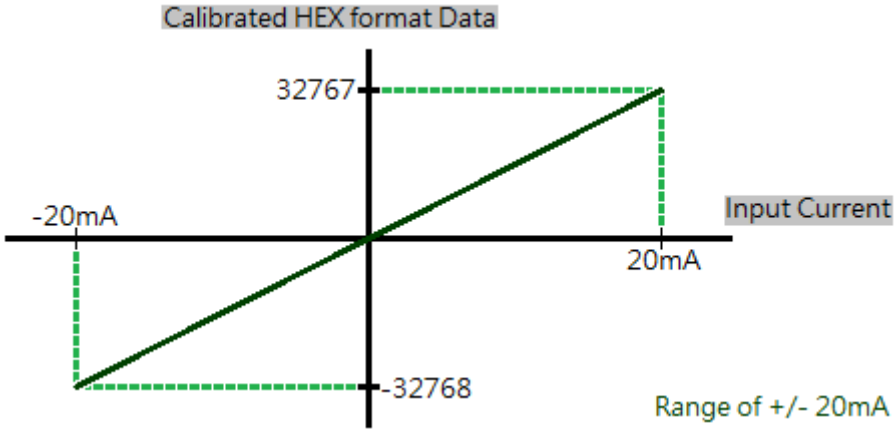
If(hexadecimal data >=0)	**floating data = (hexadecimal data / 32767) * 5**
Else	**floating data = (hexadecimal data / 32768) * 5**



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**

Examples

[C++]

```
int slot, ch, gain;
float fValue;
pac_i8017HW_ReadAI(slot, ch, gain,& fValue);
```

[C#]

```
int slot, ch, gain;
float fValue;
pac8017HWNet.pac8017HW.ReadAI(slot, ch, gain, ref fValue);
```

[C] (For LinPAC)

```
int main(){
    int slot, ch, gain, ret;
    float fValue;
    ret=Open_Slot(slot);
    if (ret > 0) {
            printf("Open Slot%d failed, return value=%d \n", slot, ret);
            return (-1);
    }
    I8017_Init(slot);
    I8017_ReadAI(slot, ch, gain, &fValue);
    Close_Slot(slot);
    return 0;
}
```

3.7. i8017H_ReadAI_AVG / pac_i8017HW_ReadAI_AVG

This function is used to read the average Analog Input value in float format from the module.

Syntax

For MiniOS7

```
short i8017HW_ReadAI_AVG(int slot,int iChannel,int iGain,unsigned short averageCnt,
    float* fValue);
```

For Windows (CE and WES)

```
short pac_i8017HW_ReadAI_AVG(int slot,int iChannel,int iGain,unsigned short averageCnt,
    float* fValue);
```

For Linux

```
short I8017_ReadAI_AVG(int slot, int iChannel, int iGain, unsigned short averageCnt, float*
fValue);                                                    // for LinPAC-8000
short I9017_ReadAI_AVG(int slot, int iChannel, int iGain, unsigned short averageCnt, float*
fValue);                                                    // for LinPAC-9000, LX-9000
```

Parameters

Slot

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

iChannel

Specifies the channel number

For I-8017HW / I-8017HCW / I-8017DW / I-9017 => 0 ~ 8 in differential method; 0 ~ 15 in single-ended method.

For I-9017-15 => 0 ~ 14 in differential method; 0 ~ 29 in single-ended method.

For I-9017C-15 => always 0 ~ 14.

iGain

Specifies the input range: 0: +/- 10.0V 1: +/- 5.0V 2: +/- 2.5V 3: +/- 1.25V 4: +/- 20mA

averageCnt

the average count for each sampling routine.(1 ~ 65535)

** fValue*

[Output] the analog input value in float format.

Return Values

Others: Refer to Appendix A: “Error Code” for more details.

Note

The parameter “fValue” is an arithmetic mean value.

This function will read 14-bit AD data many times, depend on the parameter “averageCnt”, and add all the values together.

Then, calibrate the average and convert it into 16-bit data

Examples

[C++]

```
int slot, ch, gain;
unsigned short cnt;
float fValue;
pac_i8017HW_ReadAI_AVG(slot, ch, gain, cnt,& fValue);
```

[C#]

```
int slot, ch, gain;
Uint cnt;
float fValue;
pac8017HWNet.pac8017HW.ReadAI_AVG(slot, ch, gain, cnt, ref fValue);
```

[C] (For LinPAC)

```
int main(){
    int slot, ch, gain;
    unsigned short cnt;
    float fValue;
    Open_Slot(slot);

    I8017_Init(slot);
    I8017_ReadAI_AVG(slot, ch, gain, cnt,& fValue);
    Close_Slot(slot);
    return 0;
}
```

3.8. i8017H_ReadAIHex / pac_i8017HW_ReadAIHex

This function is used to read the Analog Input value in 16-bit hexadecimal format.

Syntax

For MiniOS7

```
short i8017HW_ReadAIHex(int iSlot,int iChannel,int iGain,short* iValue);
```

For Windows (CE and WES)

```
short pac_i8017HW_ReadAIHex(int iSlot,int iChannel,int iGain,short* iValue);
```

[C] (For LinPAC)

```
short I8017_ReadAIHex(int iSlot,int iChannel,int iGain,short* iValue); // for LinPAC-8000
short I9017_ReadAIHex(int iSlot,int iChannel,int iGain,short* iValue); // for LP-9000, LX-9000
```

Parameters

iSlot

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

iChannel

Specifies the channel number

For I-8017HW / I-8017HCW / I-8017DW / I-9017 => 0 ~ 8 in differential method; 0 ~ 15 in single-ended method.

For I-9017-15 => 0 ~ 14 in differential method; 0 ~ 29 in single-ended method.

For I-9017C-15 => always 0 ~ 14.

iGain

Specifies the input range

0: +/- 10.0V

1: +/- 5.0V

2: +/- 2.5V

3: +/- 1.25V

4: +/- 20mA

** iValue*

[Output] the analog input value in hexadecimal format.

Return Values

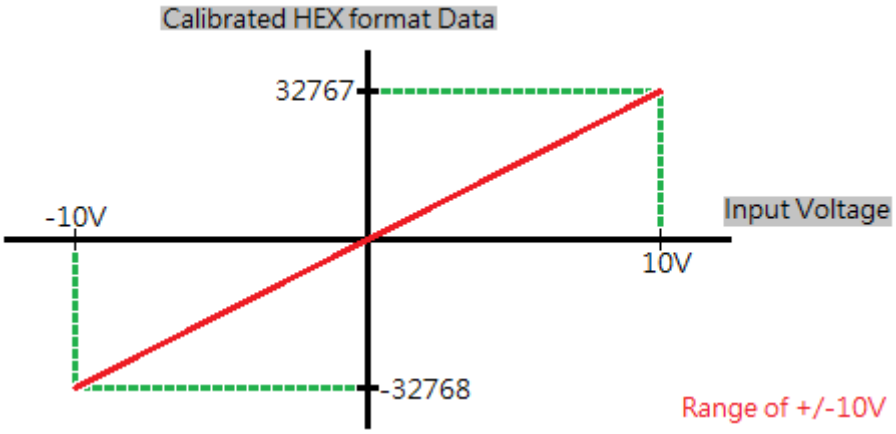
Others: Refer to Appendix A: “Error Code” for more details.

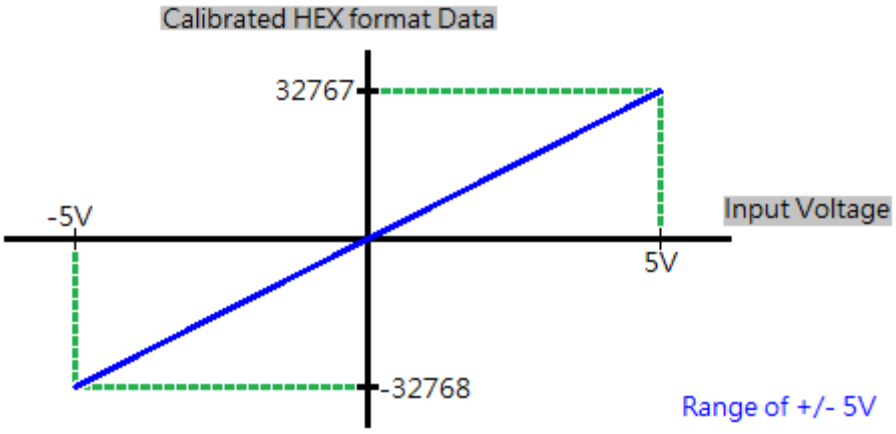
Note

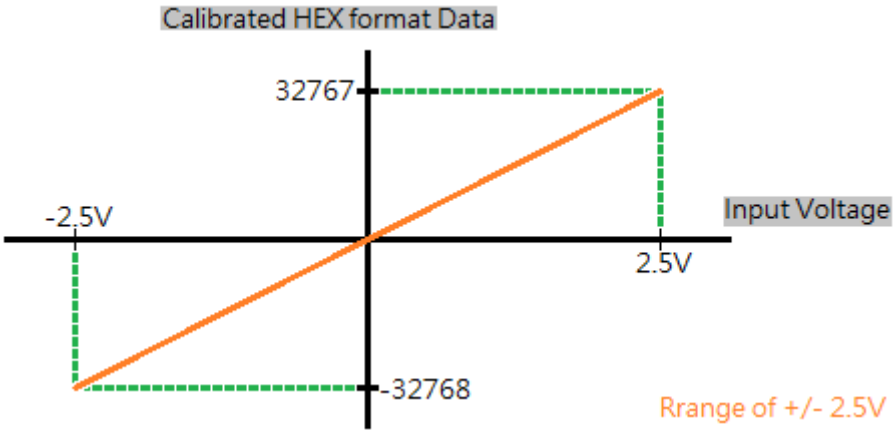
I-8017HW series modules use a 14-bit AD chip.

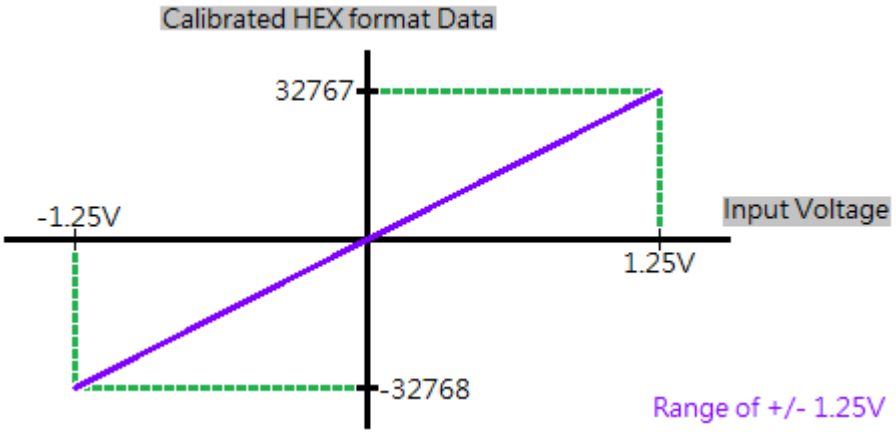
This function will calibrated the data that read from the chip and convert it into 16-bit data.

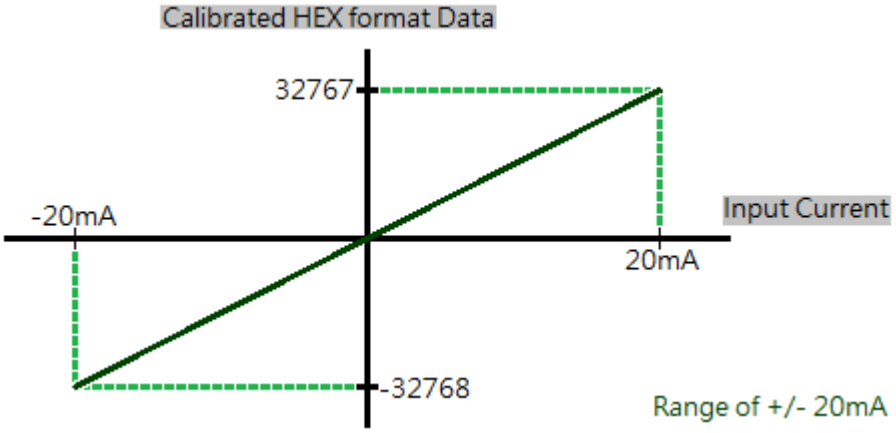
The following pictures show the scale of voltage and data.








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


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

Examples

[C++]

```
int slot, ch, gain;
short hval;
pac_i8017HW_ReadAIHex(slot, ch, gain,& hval);
```

[C#]

```
int slot, ch, gain;
int hval;
pac8017HWNet.pac8017HW.ReadAIHex(slot, ch, gain, ref hval);
```

[C] (For LinPAC)

```
int slot, ch, gain;
short hval;
Open_Slot(slot);        I8017_Init(slot);
I8017_ReadAIHex(slot, ch, gain, &hval);
```

3.9. i8017H_ReadAIHex_AVG / pac_i8017HW_ReadAIHex_AVG

This function is used to read the average Analog Input value in 16-bit hexadecimal format.

Syntax

For MiniOS7

```
short i8017HW_ReadAIHex_AVG(int slot,int iChannel,int iGain,unsigned short averageCnt,
    short* iValue);
```

For Windows (CE and WES)

```
short pac_i8017HW_ReadAIHex_AVG(int slot,int iChannel,int iGain,unsigned short
averageCnt,
    short* iValue);
```

For Linux

```
short I8017_ReadAIHex_AVG(int slot,int iChannel,int iGain,unsigned short averageCnt,short*
iValue);
short I9017_ReadAIHex_AVG(int slot,int iChannel,int iGain,unsigned short averageCnt,short*
iValue);
```

Parameters

slot

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

iChannel

Specifies the channel number

For I-8017HW / I-8017HCW / I-8017DW / I-9017 => 0 ~ 8 in differential method; 0 ~ 15 in single-ended method.

For I-9017-15 => 0 ~ 14 in differential method; 0 ~ 29 in single-ended method.

For I-9017C-15 => always 0 ~ 14.

iGain

Specifies the input range: 0: +/- 10.0V 1: +/- 5.0V 2: +/- 2.5V 3: +/- 1.25V 4: +/- 20mA

averageCnt

the average count for each sampling routine.

** iValue*

[Output] the analog input value in hexadecimal format.

Return Values

Others: Refer to Appendix A: “Error Code” for more details.

Note

The parameter “iValue” is an arithmetic mean value.

This function will read 14-bit AD data many times, depend on the parameter “averageCnt”, and add all the values together.

Then, calibrate the average and convert it into 16-bit data

Examples

[C++]

```
int slot, ch, gain;
unsigned short cnt;
short hval;
pac_i8017HW_ReadAIHex_AVG(slot, ch, gain, cnt,& hval);
```

[C#]

```
int slot, ch, gain;
uint cnt;
int hval;
pac8017HWNet.pac8017HW.ReadAIHex_AVG(slot, ch, gain, cnt, ref hval);
```

[C] (For LinPAC)

```
int main(){
    int slot, ch, gain, ret;
    unsigned short cnt;
    short hval;
    ret=Open_Slot(slot);
    if (ret > 0) {
        printf("Open Slot%d failed, return value=%d \n", slot, ret);
        return (-1);
    }
    I8017_Init(slot);
    I8017_ReadAIHex_AVG(slot, ch, gain, cnt, & hval);
    Close_Slot(slot);
    return 0;
}
```

3.10. i8017H_ReadGainOffset_Info / pac_i8017HW_ReadGainOffset_Info

This function is used to read the calibrated Gain and Offset values for the I-8017 module inserted in a specific slot

Syntax

For MiniOS7

```
short i8017HW_ReadGainOffset_Info(
    int iSlot, int iGain, unsigned short* iGainValue, short* iOffsetValue
);
```

For Windows (CE and WES)

```
short pac_i8017HW_ReadGainOffset_Info(
    int iSlot, int iGain, unsigned short* iGainValue, short* iOffsetValue
);
```

For Linux

```
short I8017_ReadGainOffset_Info(
    int iSlot, int iGain, unsigned short* iGainValue, short* iOffsetValue);
    //for LinPAC-8000
short I9017_ReadGainOffset_Info(
    int iSlot, int iGain, unsigned short* iGainValue, short* iOffsetValue);
    //for LinPAC-9000, LX-9000
```

Parameters

iSlot

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

iGain

Specifies the input range 0: +/- 10.0V 1: +/- 5.0V 2: +/- 2.5V
3: +/- 1.25V 4: +/- 20mA

**iGainValue*

[Output]Specifies the calibrated Gain value

**iOffsetValue*

[Output]Specifies the calibrated Offset value

Return Values

Refer to Appendix A: “Error Code” for more details.

Examples

[C++]

```
Int slot,Gain;
unsigned short GainValue;
short OffsetValue;
short pac_i8017HW_ReadGainOffset_Info(slot, Gain,& GainValue,& OffsetValue);
```

[C#]

```
Int slot,Gain;
unsigned short GainValue;
short OffsetValue;
pac8017HWNet.pac8017HW.GainOffset_Info(slot, Gain, ref GainValue, ref OffsetValue);
```

[C] (For LinPAC)

```
int main(){
    int slot, Gain, ret;
    unsigned short GainValue;
    short OffsetValue;
    ret=Open_Slot(slot);
    if (ret > 0) {
            printf("Open Slot%d failed, return value=%d \n", slot, ret);
            return (-1);
    }
    I8017_Init(slot);
    I8017_ReadGainOffset_Info(slot, Gain, &GainValue, &OffsetValue);
    Close_Slot(slot);
    return 0;
}
```

3.11. i8017H_Read_mA_GainOffset / pac_i8017HW_Read_mA_GainOffset

This function is used to read the calibrated Gain and Offset values for the I-9017/I-9017C-15 module inserted in a specific slot.

Syntax

For MiniOS7

```
short i8017H_Read_mA_GainOffset(
    int slot, short ch, unsigned short* GainValue, short* offsetValue
);
```

For Windows (CE and WES)

```
short pac_i8017H_Read_mA_GainOffset(
    int slot, short ch, unsigned short* GainValue, short* offsetValue
);
```

For Linux

```
short I8017_Read_mA_GainOffset(
        int slot, short ch, unsigned short* GainValue, short* offsetValue);
        // for LinPAC-8000
short I9017_Read_mA_GainOffset(
        int slot, short ch, unsigned short* GainValue, short* offsetValue);
        //for LP-9000, LX-9000
```

Parameters

slot

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

ch

Specifies the channel

Valid range :

I-8017HCW/I-9017 = 0 to 7

I-9017C-15 = 0 to 14

** GainValue*

Specifies the calibrated Gain value

** offsetValue*

Specifies the calibrated Offset value

Return Values

Refer to Appendix A: “Error Code” for more details.

Examples

[C/C++]

```
int slot;
short ch;
unsigned short GainValue;
short OffsetValue;
i8017H_Read_mA_GainOffset(slot, ch,& GainValue, & OffsetValue);
```

[C#]

```
Int slot;
Int16 ch;
UInt16 GainValue;
Int16 OffsetValue;
pac8017HWNet.pac8017HW. Ch_mAGainOffset (slot,ch,ref GainValue,ref OffsetValue);
```

[C] (For LinPAC)

```
int main(){
    int slot, ret;
    short ch;
    unsigned short GainValue;
    short OffsetValue;
    ret=Open_Slot(slot);
    if (ret > 0) {
            printf("Open Slot%d failed, return value=%d \n", slot, ret);
            return (-1);
    }
    I8017_Init(int slot);
    I8017_Read_mA_GainOffset(slot, ch, &GainValue, &OffsetValue);
    Close_Slot(slot);
    return 0;
}
```

3.12. i8017H_Select_SingleEnd / pac_i8017HW_Select_SingleEnd

This function is used to set the Single-ended/ differential mode of I-9017-15.

Syntax

For MiniOS7

```
short i8017HW_Select_SingleEnd(int slot, short selection);
```

For Windows (CE and WES)

```
short pac_i8017HW_Select_SingleEnd(int slot, short selection);
```

For Linux

```
short I9017_Select_SingleEnd(int slot, short selection);          // for LinPAC-9000, LX-9000
```

Parameters

slot

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

selection

The status of module.

0: Differential Mode

1: Single-ended Mode

Return Values

Refer to Appendix A: “Error Code” for more details.

Examples

[C/C++]

```
int slot;
short status;
pac_i8017H_Select_SingleEnd(slot, status);
```

[C#]

```
Int slot;
Int16 selection;
pac8017HWNet.pac8017HW. Select_SingleEnd_Differential (slot, selection);
```

[C] (For LinPAC)

```
int main(){
    int slot, ret;
    short status;
    ret=Open_Slot(slot);
    if (ret > 0) {
            printf("Open Slot%d failed, return value=%d \n", slot, ret);
            return (-1);
    }
    I9017_Init(slot);
    I9017_Select_SingleEnd(slot, status);
    Close_Slot(slot);
    return 0;
}
```

3.13. i8017H_Get_D_Sub_Status / pac_i8017HW_Get_D_Sub_Status

This function is used to get connector status between D sub and 8017DW.

Syntax

For MiniOS7

```
short i8017HW_Get_D_Sub_Status(int iSlot, short* D_Sub_Status);
```

For Windows (CE and WES)

```
short pac_i8017HW_Get_D_Sub_Status(int iSlot, short* D_Sub_Status);
```

For Linux

```
short I8017_Get_D_Sub_Status(int iSlot, short* D_Sub_Status);          // for LinPAC-8000
```

Parameters

slot

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

**D_Sub_Status*

The status of D_Sub.

1 : Open

0 : Close

Return Values

Refer to Appendix A: “Error Code” for more details.

Note

function for I-8017DW module only, in the others 8017 series module, the value of D Sub Status will always be 1.

Examples

[C/C++]

```
int slot;
short status;
pac_i8017HW_Get_D_Sub_Status(slot, status);
```

[C#]

```
int slot;
Int16 selection;
pac8017HWNet.pac8017HW.D_Sub_Status(slot, selection);
```

[C] (For LinPAC)

```
int main(){
    int slot, ret;
    short status;
    ret=Open_Slot(slot);
    if (ret > 0) {
            printf("Open Slot%d failed, return value=%d \n", slot, ret);
            return (-1);
    }
    I8017_Init(slot);
    I8017_Get_D_Sub_Status(slot, status);
    Close_Slot(slot);
    return 0;
}
```

4. Calibration

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

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

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

Tips & Warnings

1. An unstable calibration source will cause calibration errors and will affect the accuracy of the data acquisition.
2. I-8017HW / I-8017HCW / I-8017DW / I-9017-15 only uses channel 0 to calibrate every type of range.
3. The gain and offset value of the range of +/- 20 mA for I-8017HW / I-8017HCW / I-8017DW/I-9017-15 are the same as the range of +/- 2.5V.

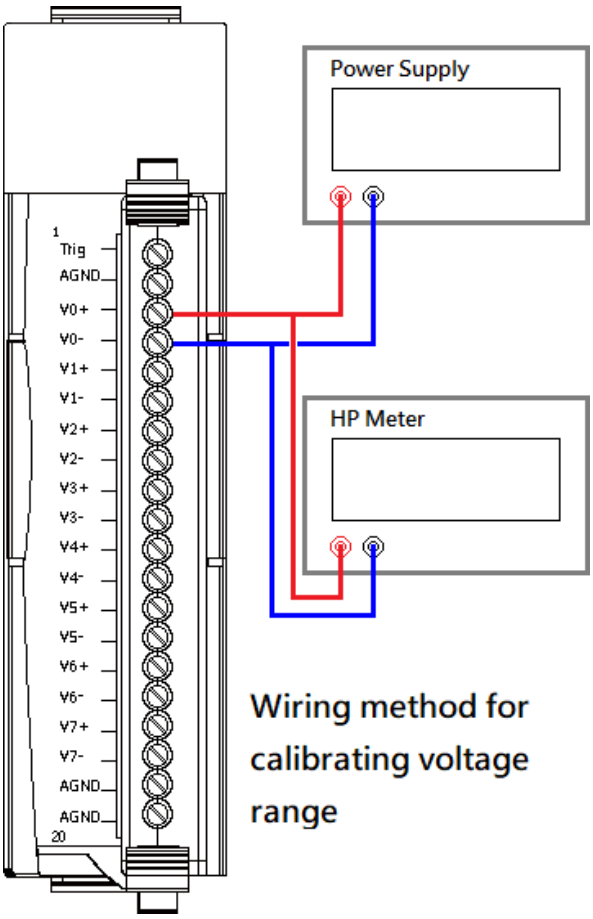
   If users wish to calibrate +/- 20 mA, calibrate +/- 2.5V will be fine.
4. I-9017 only uses channel 0 to calibrate every voltage range.
5. I-9017 / I-9017C-15 needs to calibrate every channel one by one within the range of +/- 20mA.

4.1. Calibrate I-8017HW series modules on iPAC-8000

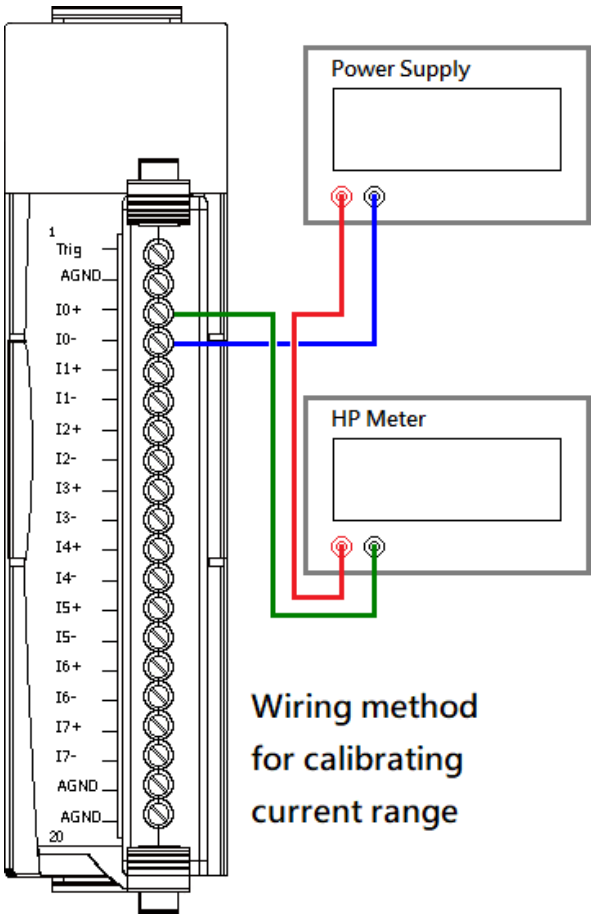
Step 1: Wiring method

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:



Wiring method for calibrating voltage range



Wiring method for calibrating current range

Then, plug module into the controller.

Step 2: Download, upload and execute calibration program

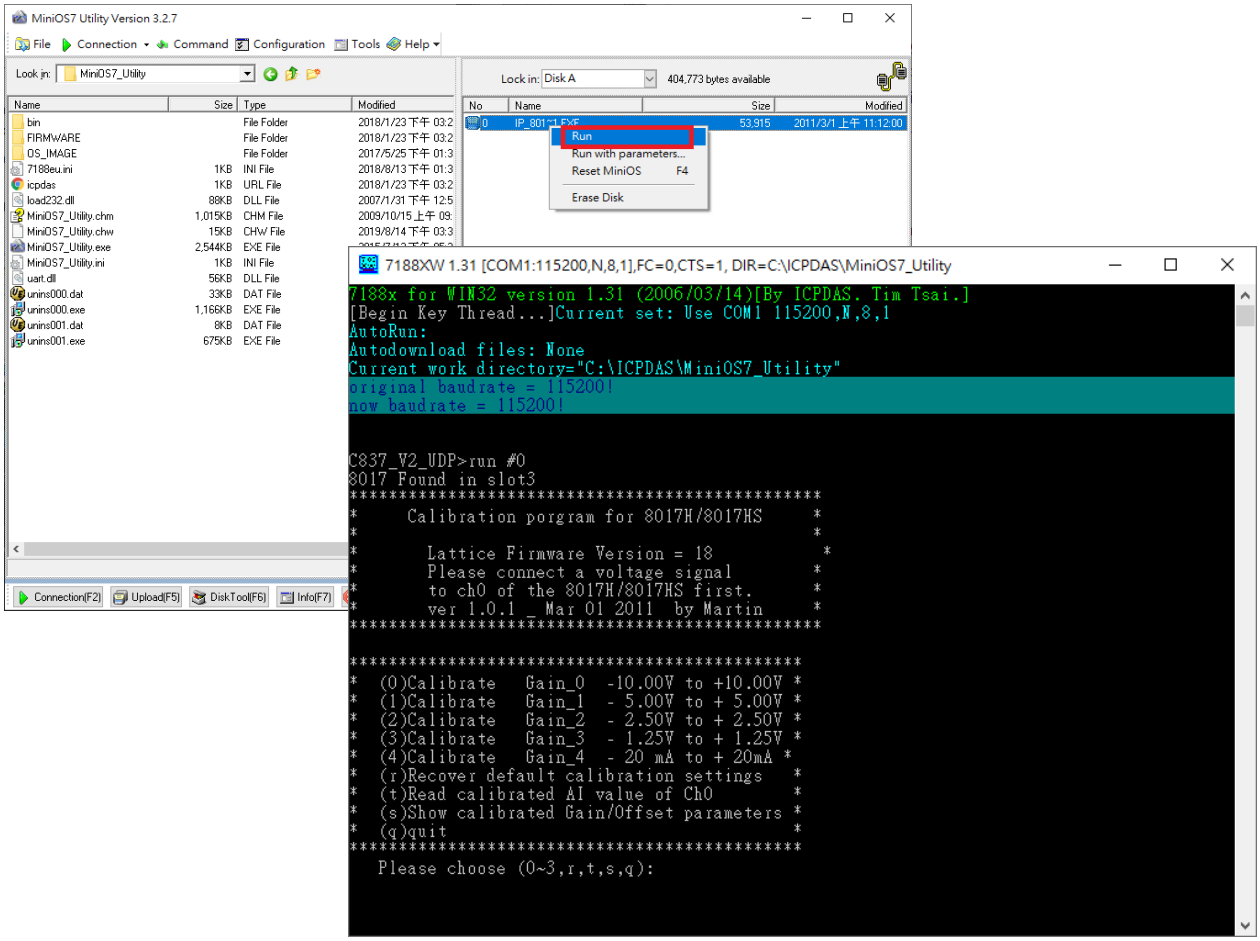
The calibration program can be downloaded in ICP DAS website.

Please refer to the following link:

https://www.icpdas.com/en/download/show.php?num=2323

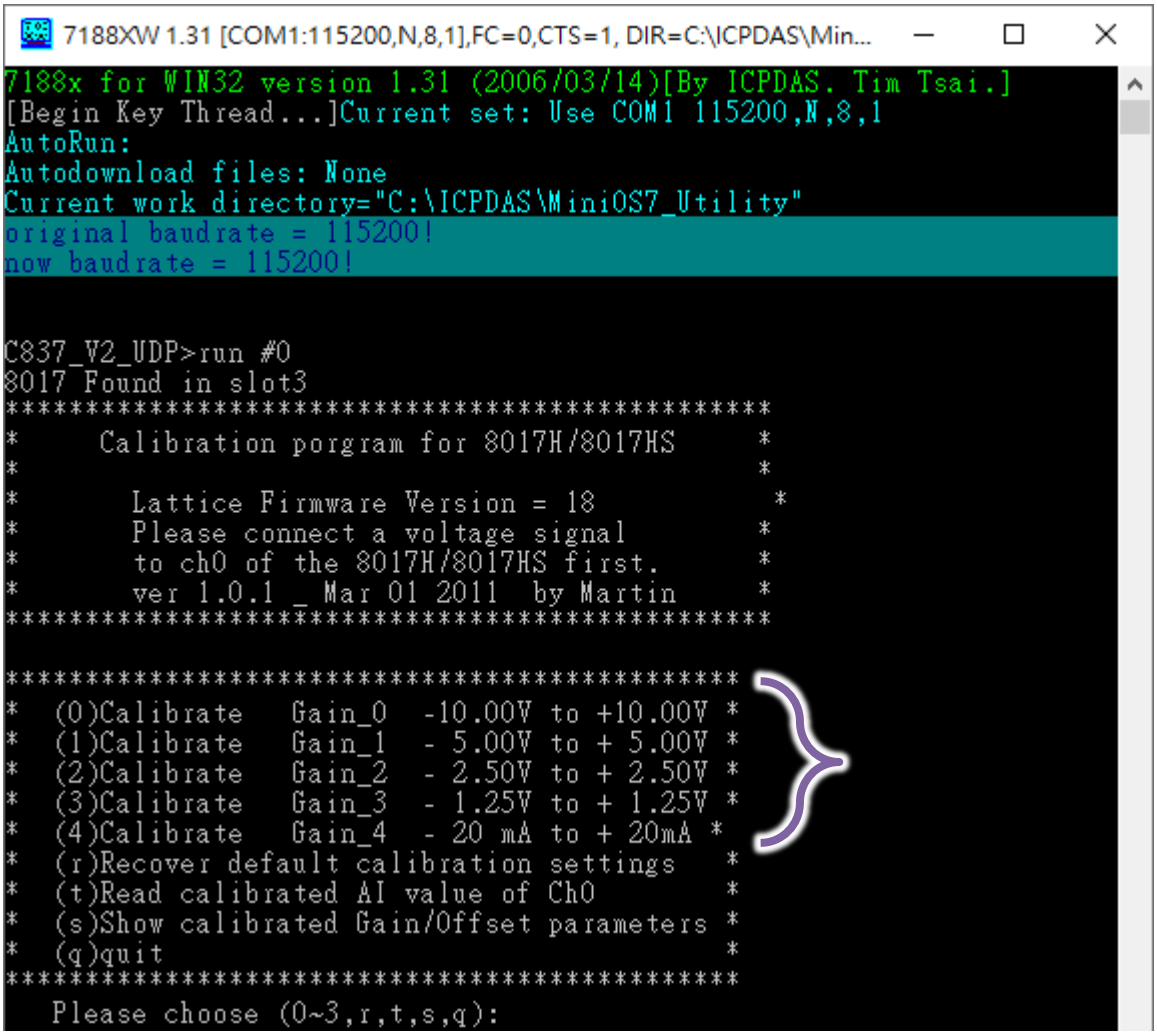
In order to upload programs, please refer to the "**2.2.2. Installing the MiniOS7" and "2.5.2. Uploading and Executing iPAC-8000 programs**" of the IP-8000 user manual to download MiniOS7 Utility which can help users to upload programs and learn how to operate.

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



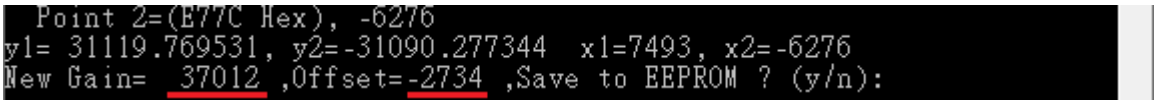
Step 3: Calibrate

After execute the program, select the range that needs to be calibrated and press the number corresponding to the range.



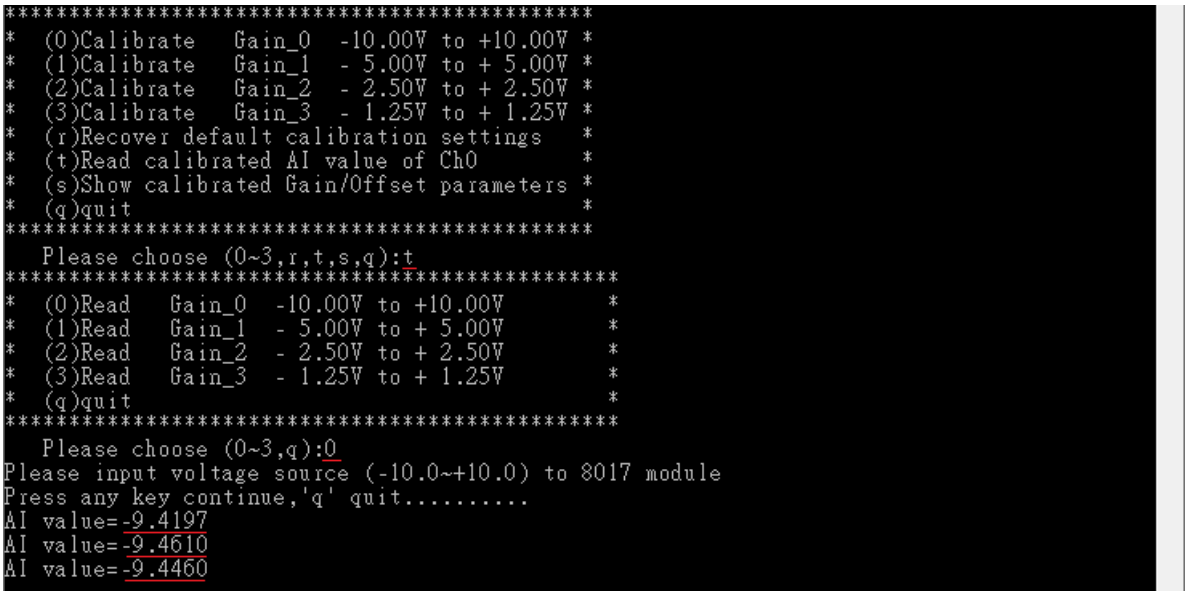
Output stable positive source to channel 0, and type the value displayed on the voltage meter.

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



Press ‘y’ to save new gain and offset values

After finish the calibration, press ‘t’ to test calibrated AI data with new gain and offset values, and check whether the AI value is correct or not



4.2. Restore I-8017HW series modules to defaults on iPAC-8000

Step 1: Download, upload and execute calibration program

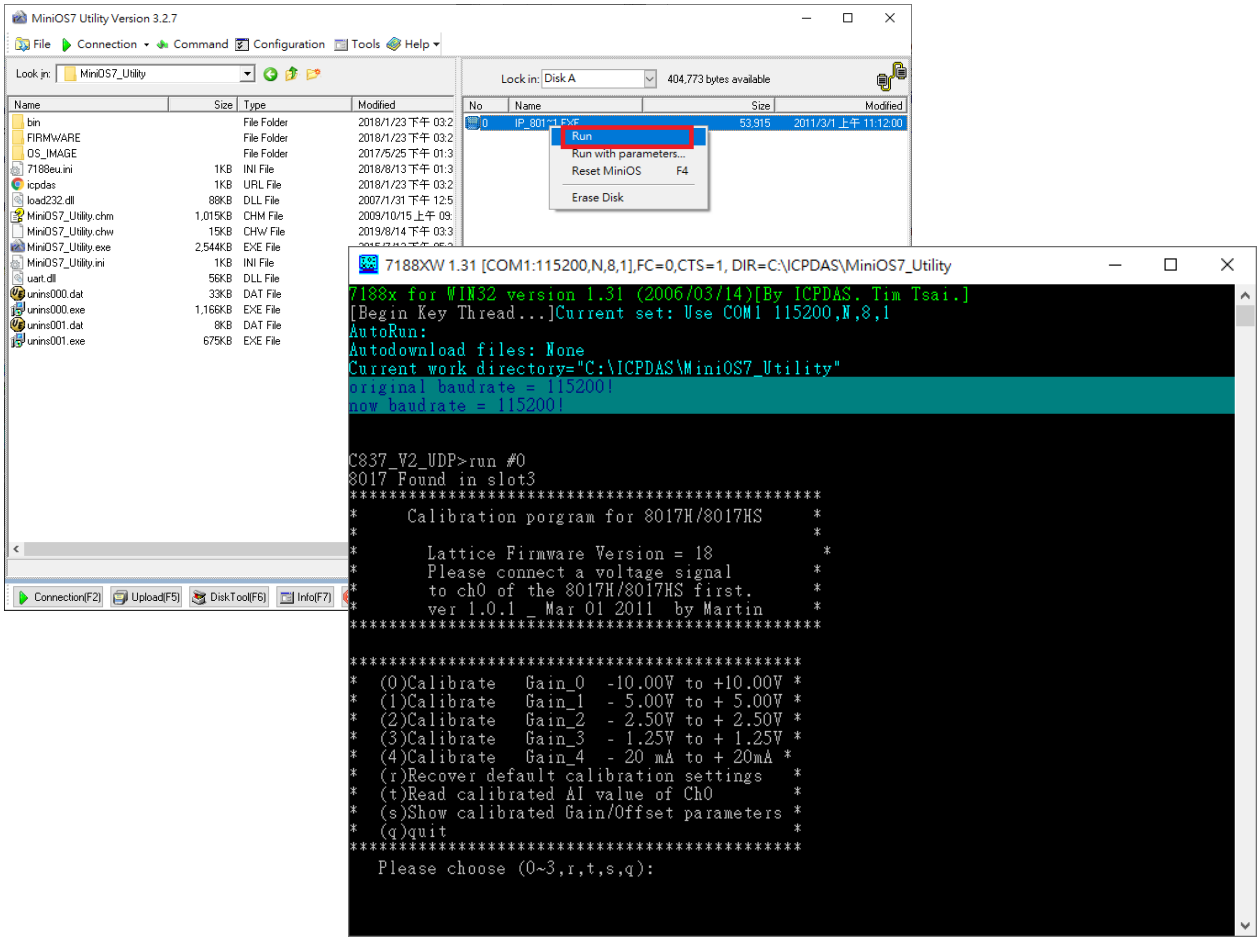
The calibration program can be downloaded in ICP DAS website.

Please refer to the following link:

https://www.icpdas.com/en/download/show.php?num=2323

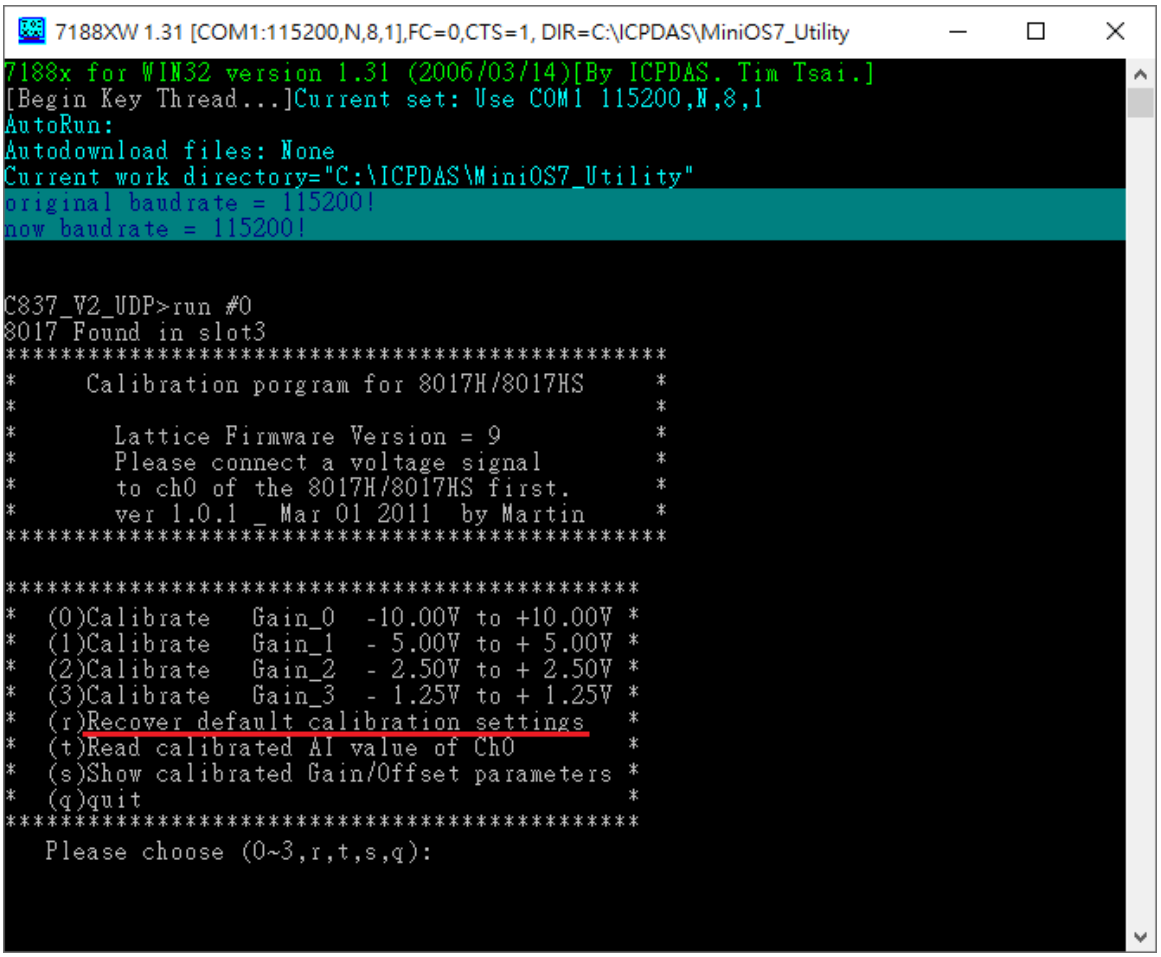
In order to upload programs, please refer to the "**2.2.2. Installing the MiniOS7" and "2.5.2. Uploading and Executing iPAC-8000 programs**" of the IP-8000 user manual to download MiniOS7 Utility which can help users to upload programs and learn how to operate.

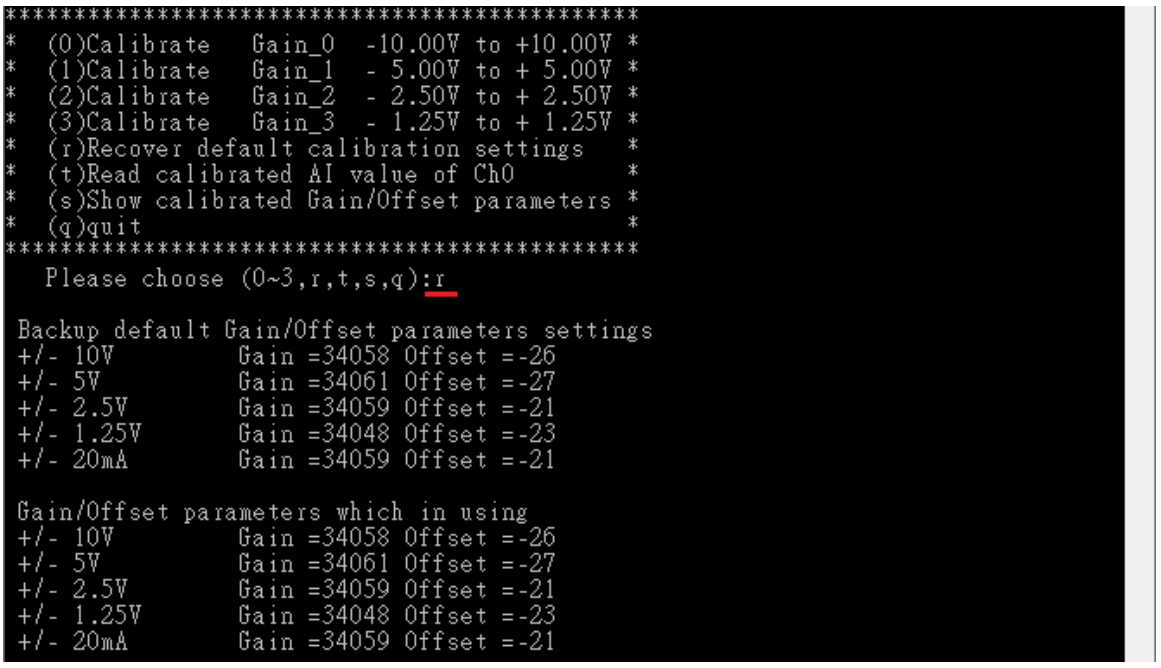
After uploading the calibration program, right click on it and click "Run" to execute it.



Step 2: Restore defaults

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

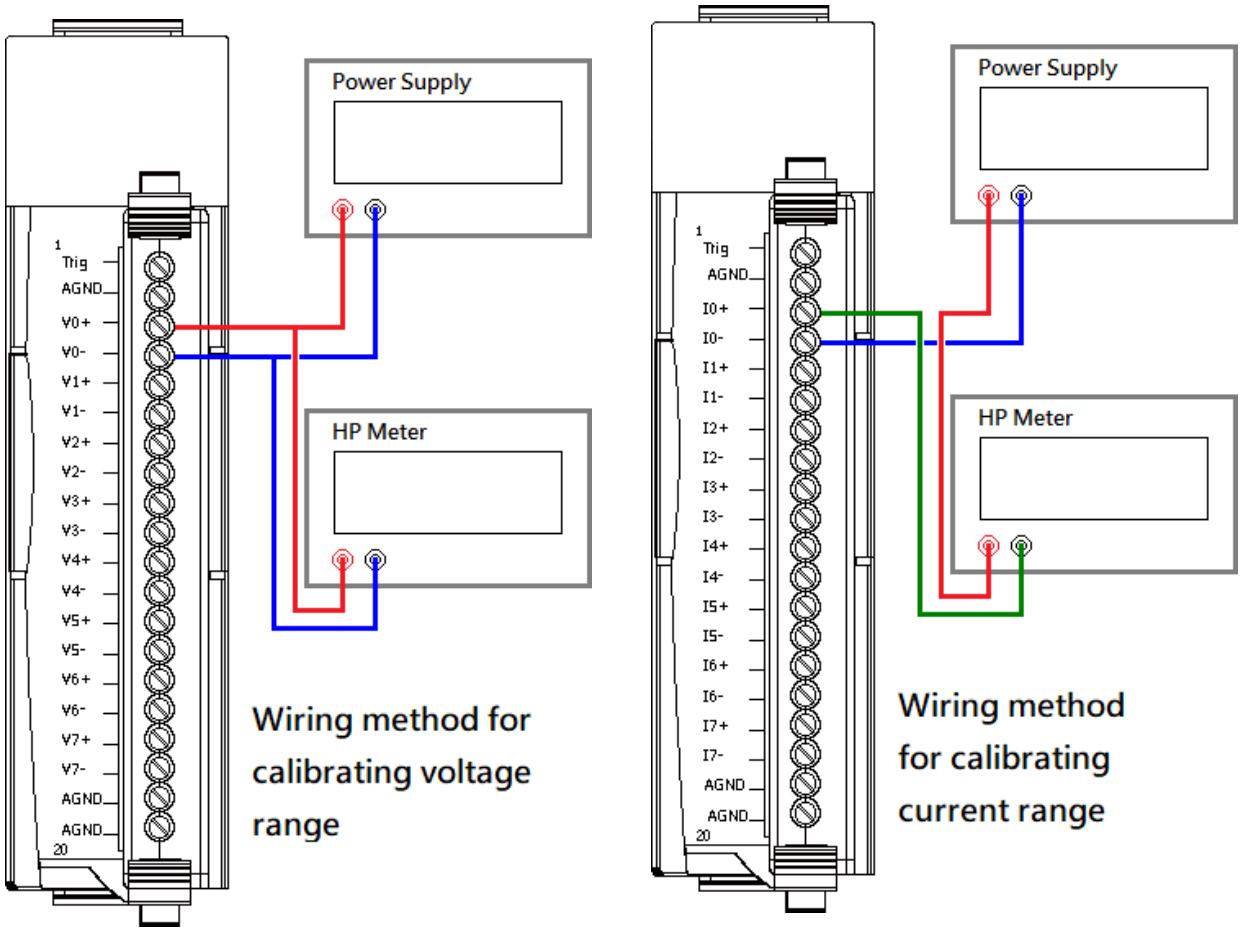




4.3. Calibrate the I-8017HW series modules on WinCE and WES units

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 module into the controller.

Step 2: Download and execute calibration program

The calibration program can be downloaded in ICP DAS website.

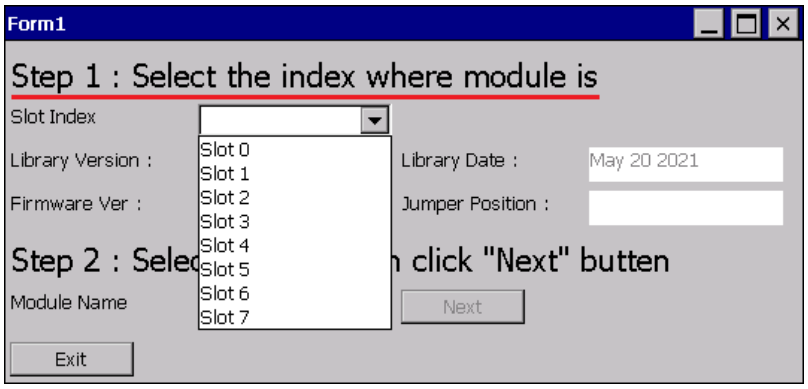
Please refer to the following link:

https://www.icpdas.com/en/download/show.php?num=2323

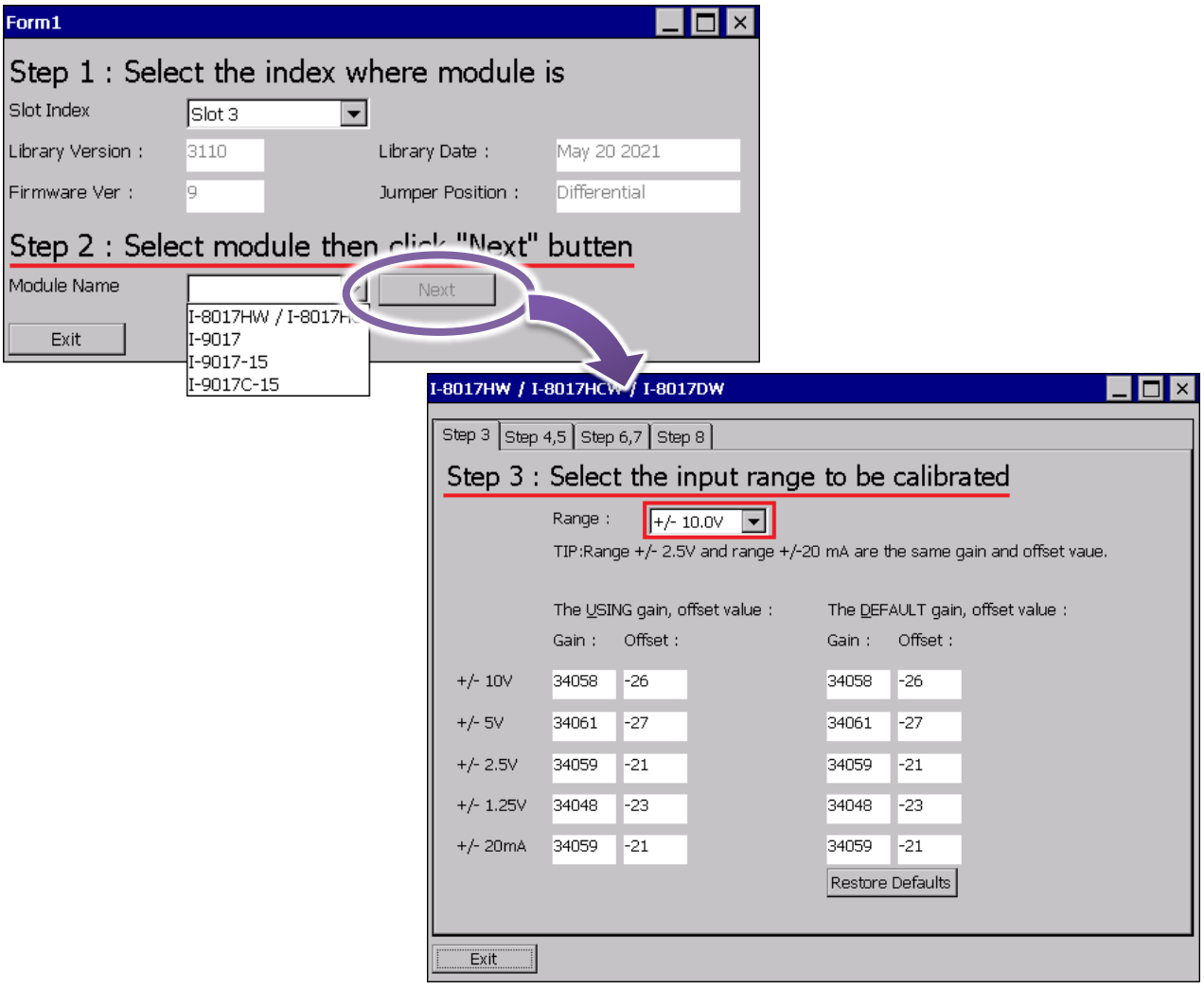
Step 3: Calibrate

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

Select the index where the module is.

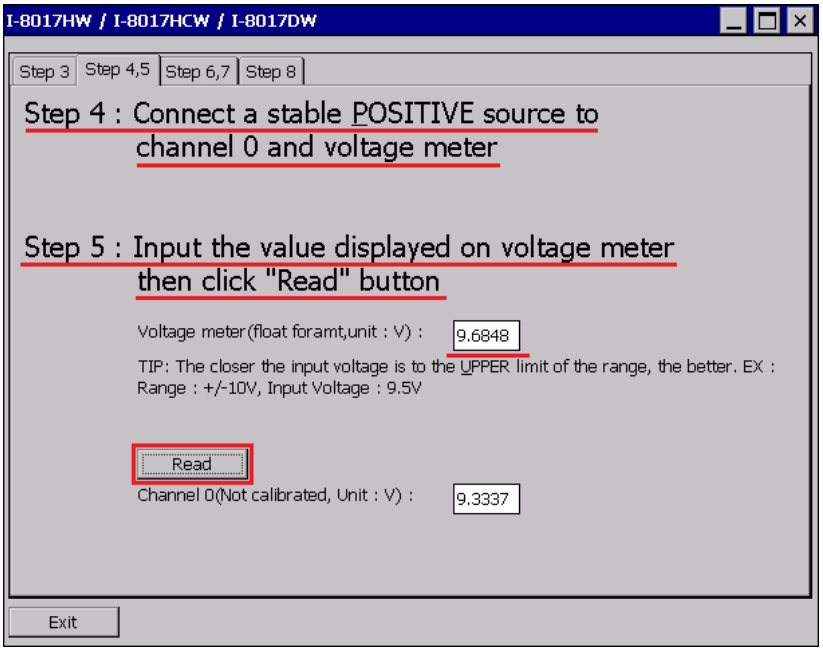


Select the name of module and click “Next” button.



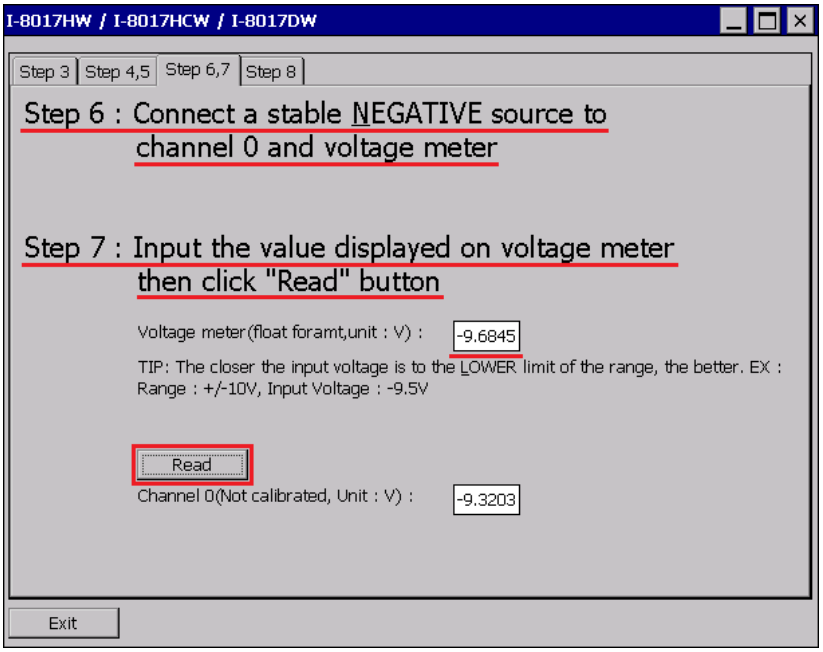
Change to “page Step 4,5”,

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



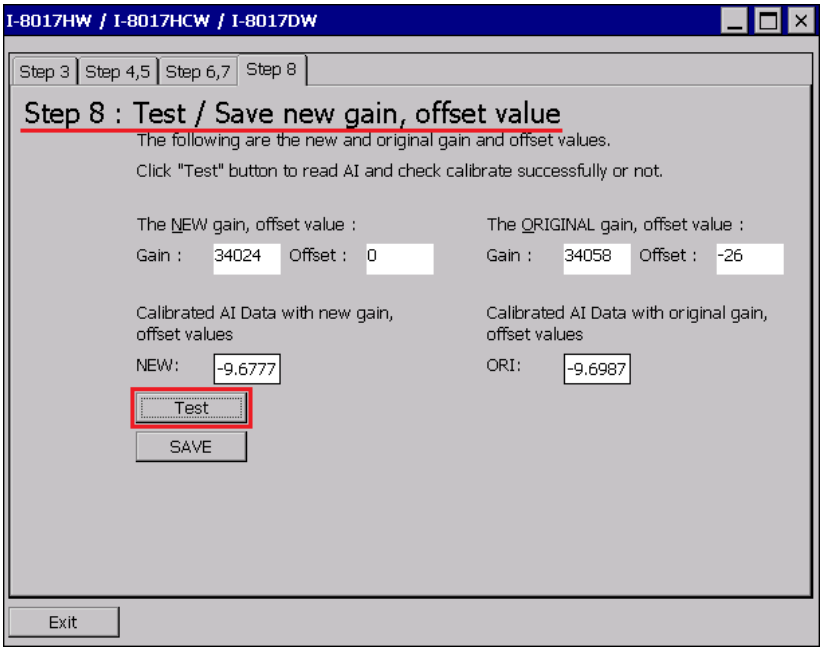
Change to “page Step 6,7”,

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

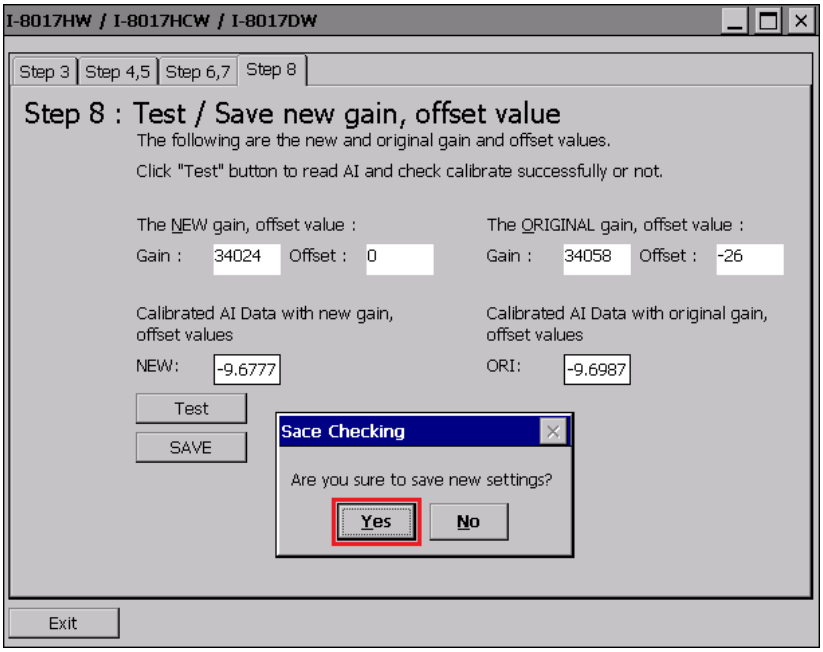


Change to “page Step 8”,

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.



4.4. Restore I-8017HW series modules to defaults on WinCE and WES units

Step 1: Download and execute calibration program

The calibration program can be downloaded in ICP DAS website.

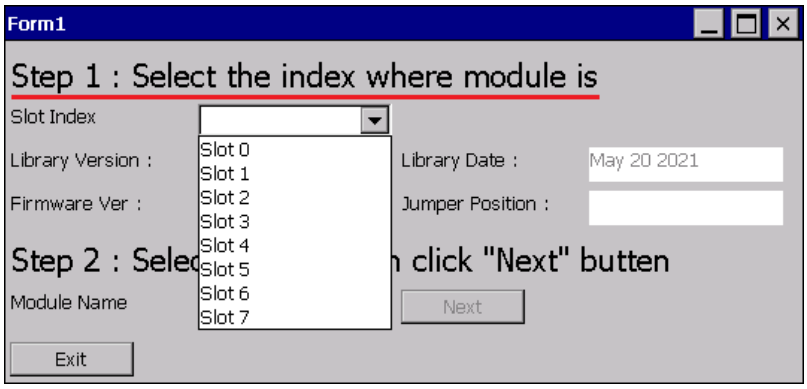
Please refer to the following link:

https://www.icpdas.com/en/download/show.php?num=2323

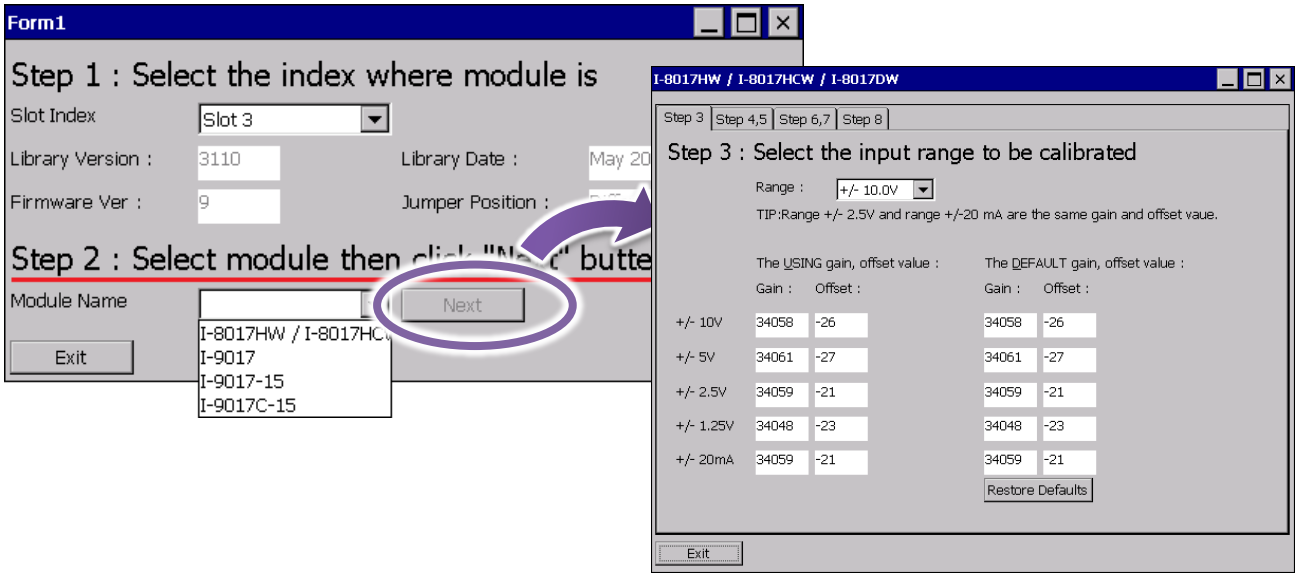
Step 2: Restore defaults

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

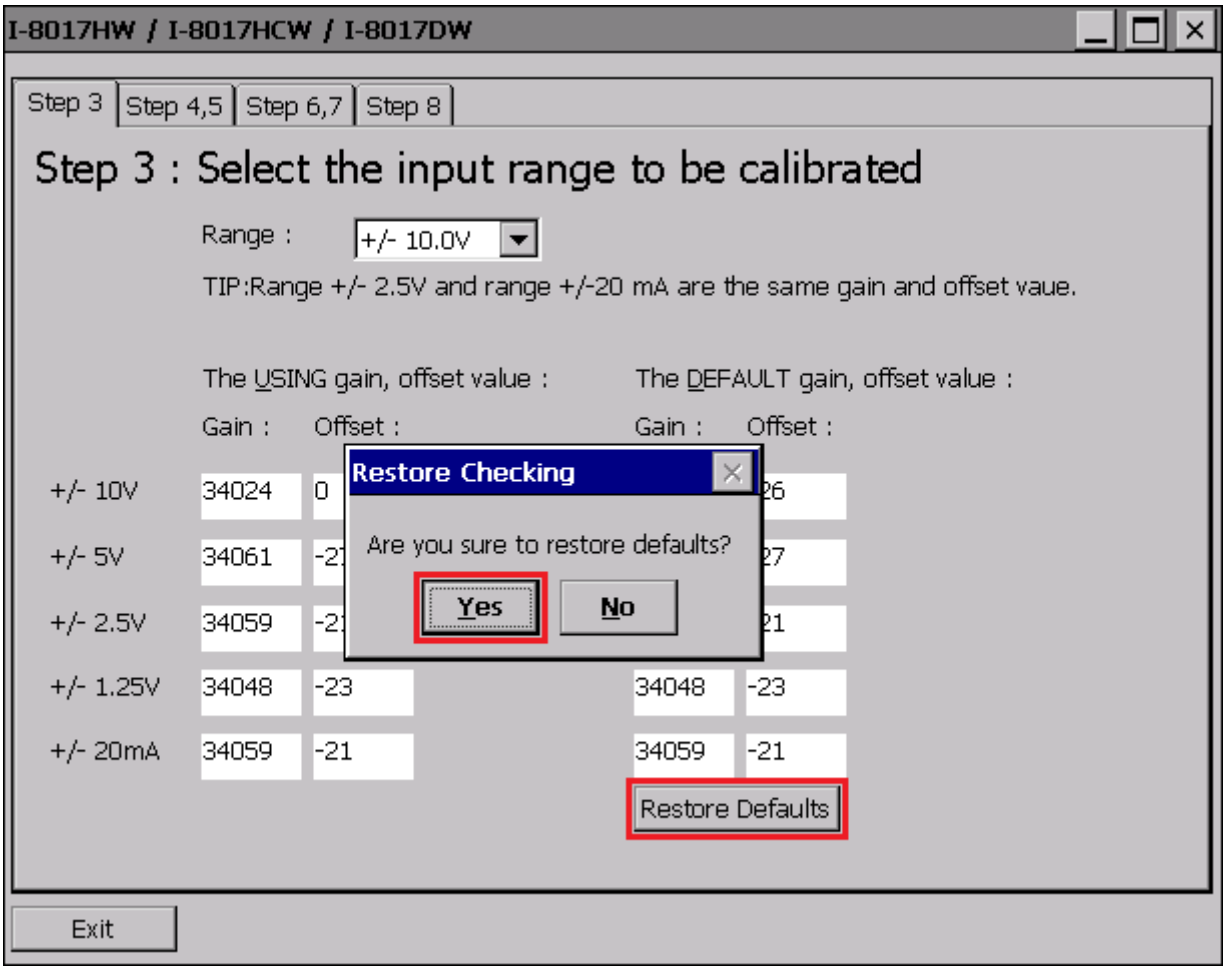
Select the index where the module is.



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



Click “Restore Defaults” button.



5. Troubleshooting

This chapter discusses how to solve some common problems you may encounter.

This chapter contains:

- How to verify the AI function on a WinCE or WES PAC Service/Request Requirements
- What to do when the data read from the module seems unstable

5.1. Verifying Analog Input functionality on a WinCE or WES PAC device

If the data read from the module is inconsistent with the input signal, and you would like to confirm the input function, the pac_i8017W_Utility.exe tool may be helpful. The utility can only be used with modules designed for controllers using the WinCE and WES platforms and is located in the I-8017W C# demo program folder for the controller. (See the Location of the Demo Programs section for more details)

Step 1: Connect a stable signal to the module.

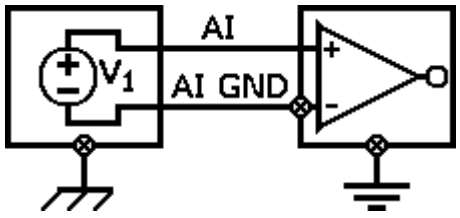
a. Connect your input signal according to whether differential or single-ended Jumper settings are used. (See the Jumper Settings section for more details)

b. The input range can be from -10 V to +10 V.

c. Insert the module into a slot in a Windows platform controller and then power on the controller.

Tips & Warnings

1. A battery output should provide a stable enough signal.
2. A 125 Ω resistor is required when measuring current input.
3. If the result is not as stable as the input signal when measuring the voltage using the differential input type, it is recommended that an additional wire is connected between the Vn- and the AGND (analog ground) pins to enhance the accuracy. Note that this method has no benefit in enhancing accuracy when measuring current input.



Step 2: Launch the pac_i8017W_Utility.exe

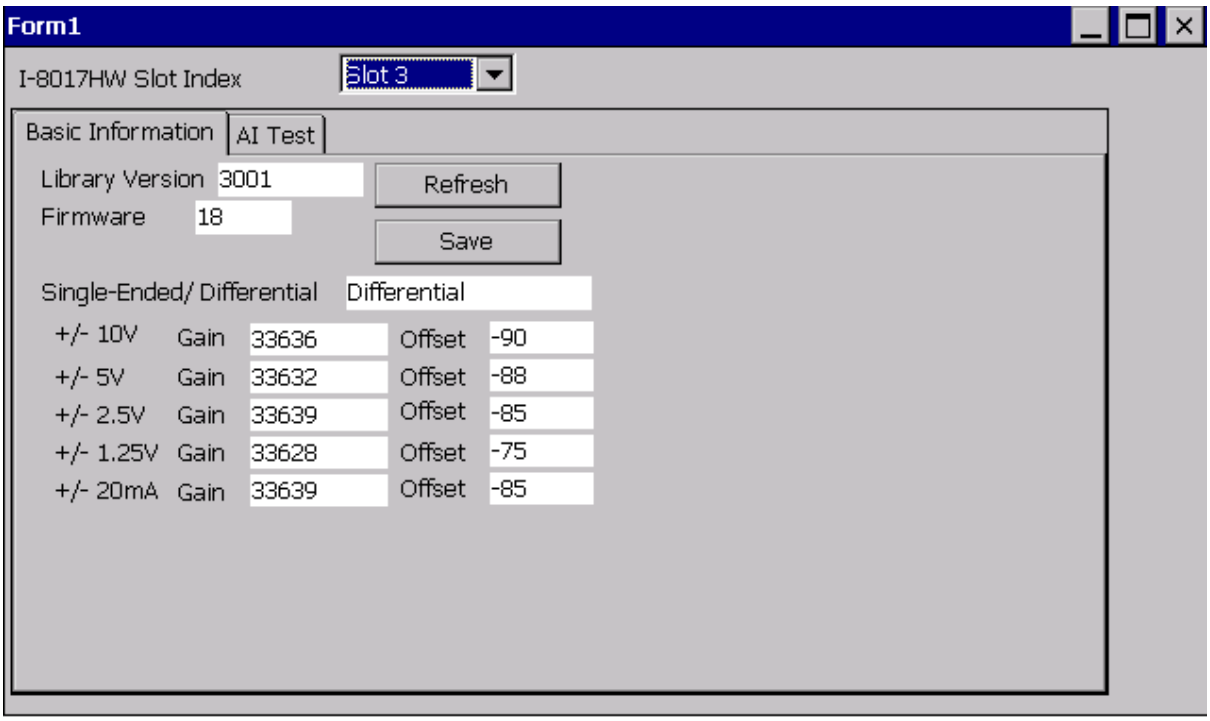
Step 3: Read the information from the module

a. Select the slot that the module is connected to from the slot index drop-down list.

b. Click the Basic Information tab.

The Basic Information page includes:

- The version information for the FPGA firmware
- The current position of the Differential/Single-ended jumper
- The Gain and Offset values for each input type



Click the Save button to save all the information to the Slot1_8017W_Info.txt file. This information is useful for troubleshooting when requesting service.

Verifying the Gain and Offset Values

In a normal situation, the Gain value should be around 33000 (33000 to 34000). If the value is greatly different from 33000, it means that the value is incorrect. To correct this situation, try the following:

a. Press Refresh to retrieve the Gain values again and confirm whether or not they are correct.

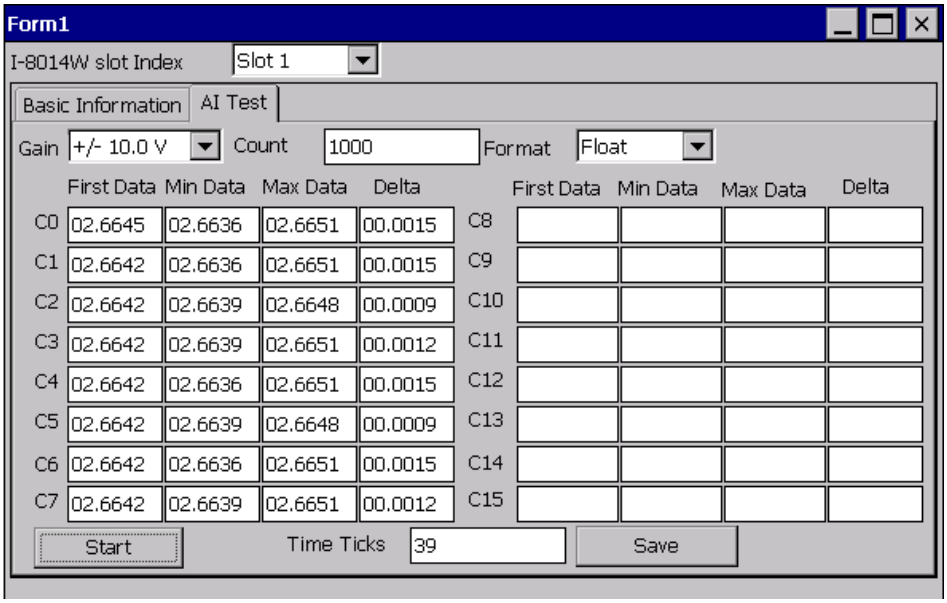
b. Relocate the module to a different slot, and then repeat Steps 2 and 3 to confirm whether or not the Gain values are correct.

Test the input function.

a. Click the AI test tab, and then select the required input range from the Gain drop-down list.

b. Enter the required sample count, and choose the data format from the Format drop-down list.

c. Click the Start button.



After the sampling process is completed, the data will be displayed in the respective columns for each channel.

d. If necessary, click the Save button to save the data and the sampling time to the SampleData_Hex_mm_dd_hh_mim_sec.csv file.

5.2. Service Request Requirements

If you are using a stable signal source to output a signal to the module, such as a battery, and are receiving incorrect or unstable data, prepare the following three items and e-mail them to service@icpdas.com

- An image of the physical wiring
- The file saved from the Basic Information tab (See step 3 in Section 6.1 above)
- The file saved from the AI Test tab (See step 4 in Section 6.1 above)

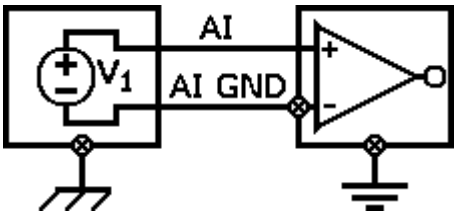
5.3. What to do when the data read from the module seems unstable

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:

- A noise-corrupted signal source
- Instability in the signal source
- A floating signal source that is not referenced to a system ground(earth or building ground)

Because of the nature of the high-speed data acquisition function on the module, 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.

It is recommended that the V- pin is connected to the AGND (system ground) pin when measuring differential signals, as shown in the figure below.



Appendix A. Error Code

Code	Definition	Description
0	NoError	This indicates that there have been no errors
-1	ID_ERROR	There is a problem with the module ID
-2	SLOT_ERROR	There is a Slot index error
-3	CHANNEL_ERROR	There is a Channel index error (0 - 15)
-4	GAIN_ERROR	There is a Gain error (0 - 4)
-6	NOT_SUPPORT_ERROR	The function is not support the Firmware
-7	NOT_Calibration	The module is not calibrated

Appendix B. Read AI Function Performance

Test with library version 0x3110.

Test Items \ Platform			MiniOS7	CE 5.0	CE 6.0	CE 7.0	WES	IOT
8017H	HEX format	One channel	29	94 ~ 95	71 ~ 82	112 ~ 115	77	81
		Eight channels	19	38 ~ 39	35 ~ 39	36 ~ 37	36	37
	Float format	One channel	7	89 ~ 93	70 ~ 82	111 ~ 113	77	82
		Eight channels	4	37 ~ 38	35 ~ 39	36 ~ 37	36	36
8017HC	HEX format	One channel	28	95 ~ 96	72 ~ 82	113 ~ 116	77	82
		Eight channels	18	38 ~ 39	35 ~ 39	36 ~ 37	36	37
	Float format	One channel	7	90 ~ 91	71 ~ 82	111 ~ 114	77	82
		Eight channels	6	37 ~ 38	35 ~ 39	36 ~ 37	35	37

Unit : K Hz.

Notes

1. There is no need to switch the MUX when using a single channel as it provides the best performance. However, when using multiple channels the MUX needs to be switched and you should be aware that the performance will be affected by switching the MUX.
2. The MiniOS7 system is not designed for mathematical operations, so it is more suitable for non-continuous data sampling in high speed applications.
3. Large amounts of non-continuous data samples can be saved on the other memory devices, for example MicroSD cards or NAND flash memory.
4. A Backplane Timer Interrupt can be used for the CE5 and CE6 platforms when performing continuous data sampling.
5. The Timer on the WES platform can be affected by Ethernet communication or when using a mouse. If greater accuracy is required for the sample frequency (less than 50 ms), it is recommended that either the CE5 or the CE6 platform is used.

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	January 2018	Initial issue
2.0.0	January 2018	• Added content for the I-8017DW and I-8017HCW modules • Added calibration instructions for modules based on the Windows platform • Added performance information for all platforms
3.0.0	March 2018	• Added content for the I-9017, I-9017-15,I-9017C-15 modules • Modify library , demo path • Added WP-9000 , ippc-wes7 library , demo path • Modify API functions
3.0.2	August 2019	Modify calibrate steps
3.0.3	May 2021	• Modify Quick start • Modify the download path of libraries and demos • Modify performance information for all platforms • Added optioning modules on Linux platform