I-8017HW Series
User Manual

Version 2.0.1 July 2013

Service and usage information for

I-8017HW **I-8017DW** **I-8017HCW**

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, not for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

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 via email at service@icpdas.com

We guarantee to provide a response within two working days.

Table of Contents

1. Introduction

I-8017 series includeI-8017HW, I-8017DW and I-8017HCW modules are high-performance Analog Input modules. Up to 16 single-ended or 8 differential input channels can be connected to a single module, selectable via a jumper.

The nodules feature 14-bit resolution, and 100 k S/s sampling rates, and also provide isolation protection of 2500 Vrms.

Applications:

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

For more details related to resolution or to high-performance modules, please refer to selection guide on the web page below and select the appropriate module.

http://www.icpdas.com/products/Remote_IO/i-8ke/selection_analog_i8k.htm

Each of the three modules can be used to measure both voltage and current source. The I-8017DW and I-8017HCW modules include a Jumper that can be used to set the discrete input circuits to add a125 Ω resistor, so it's not necessary to add external resistor for Differential input. For the I-8017HW, however, an external 125 Ω resistor needs to be added in order to measure the current source.

(For more details, see Section 1.2 "Jumper Settings" and Section 1.5 "Wire Connections".

The 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:

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

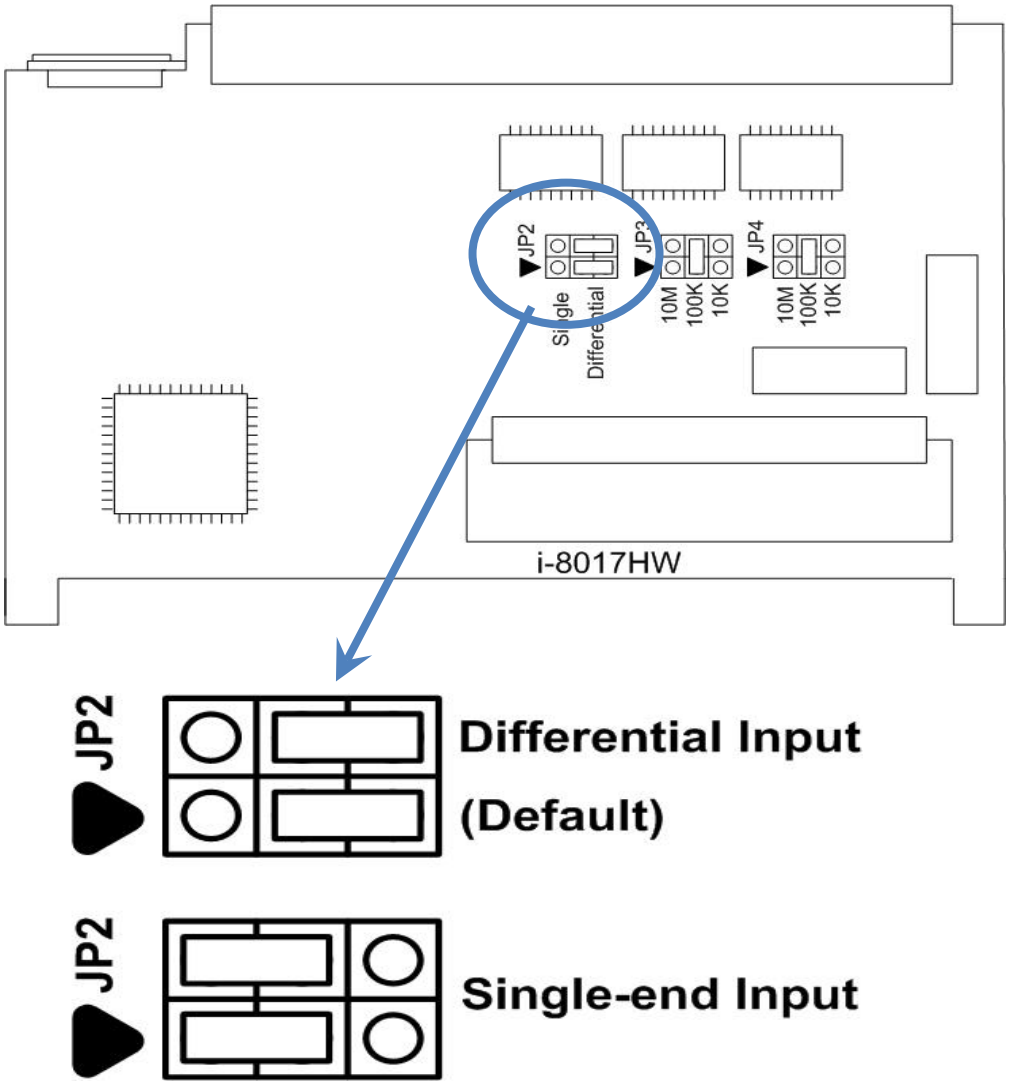
1.1. Specifications

Models	I-8017HW	I-8017HCW	I-8017DW
Analog Input			
Input Channels	16-channel Single-ended/8-channel Differential		
Voltage Input Range	+/- 10V, +/- 5V, +/- 2.5V, +/- 1.25V		
Current Input Range -20 to +20 mA	Requires an Optional External 125 Ω Resistor	Jumper Selectable	
Resolution	14-bit		
Sample Rate	Single Channel Polling Mode :100 k S/s Single Channel Interrupt Mode: 50 k S/s 8 channel Scan Mode : 16 k S/s		
Accuracy	±0.1% of FSR		
Zero Drift	± 0.1uV/°C		
Span Drift	± 10ppm/°C		
ESD Protection	4kV		
Input Impedance	20K, 200K, 20M(Jumper Select)		
Input Bandwidth	100 KHz		
Intra-module Isolation, Field to Logic	2500 VDC		
Connector	20 Pin Terminal Block		D-sub 37 Pin
LED Display			
1 LED as Power Indicator/16 LEDs as Status Indicators			
Power			
Power Consumption	2W Max		
Environment			
Operating Temperature	-25 to +75 °C		
Storage Temperature	-30 to +75 °C		
Humidity	5 ~ 95%, Non-condensing		
Dimensions			
102 mm x 30 mm x 115 mm (L x W x H) Details			

1.2. Jumper Settings

I-8017HW:

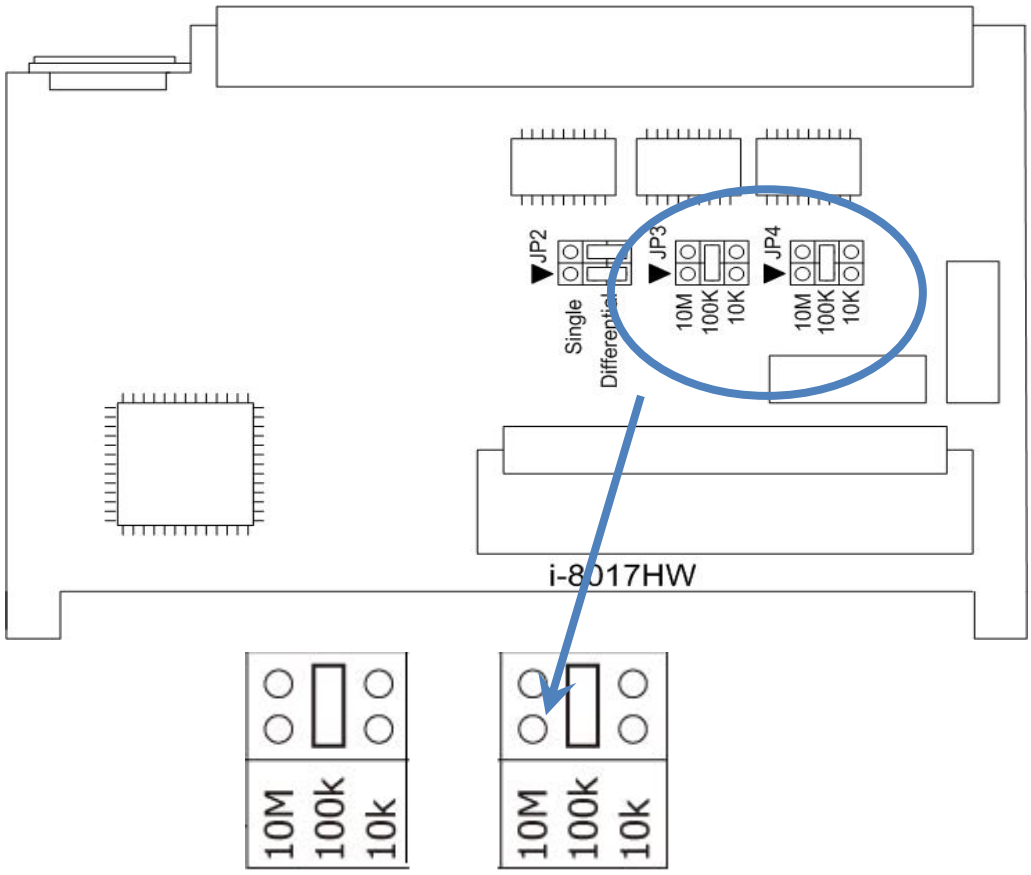
Single-ended and Differential Jumper:



This jumper is used to set the discrete input circuits as either

“Single-ended” or “Differential” inputs.

Adjusting the Input impedance



Select Input Impedance: 200 kΩ (Default)

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

The I-8017 series modules allows three input impedance options including as 20 kΩ,

200kΩ (the default setting) and 20MΩ to meet system requirements.

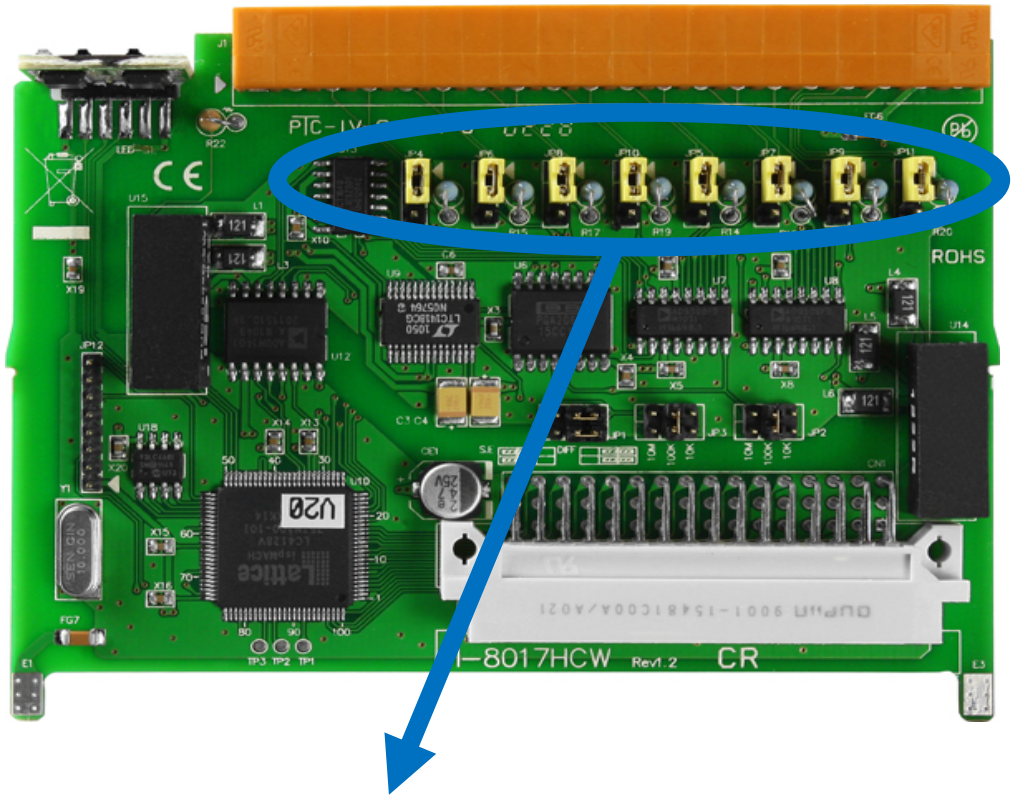
In most cases, 200kΩ is sufficient. Note that each time the input impedance is

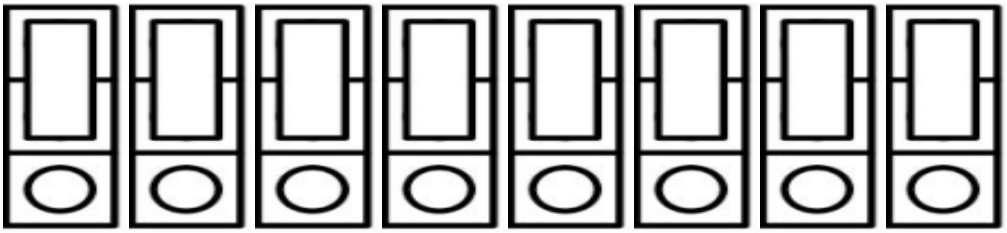
adjusted on a calibrated module, the module must be recalibrated. For more details, refer to the relevant Calibration information, which can be found in Section 5.1 if you are using either an I-8000 or iPAC-8000 (MiniOS7 platform) controller, or in Section 5.2 if you are using a WinCE or WES platform unit.

I-8017DW and I-8017HCW:

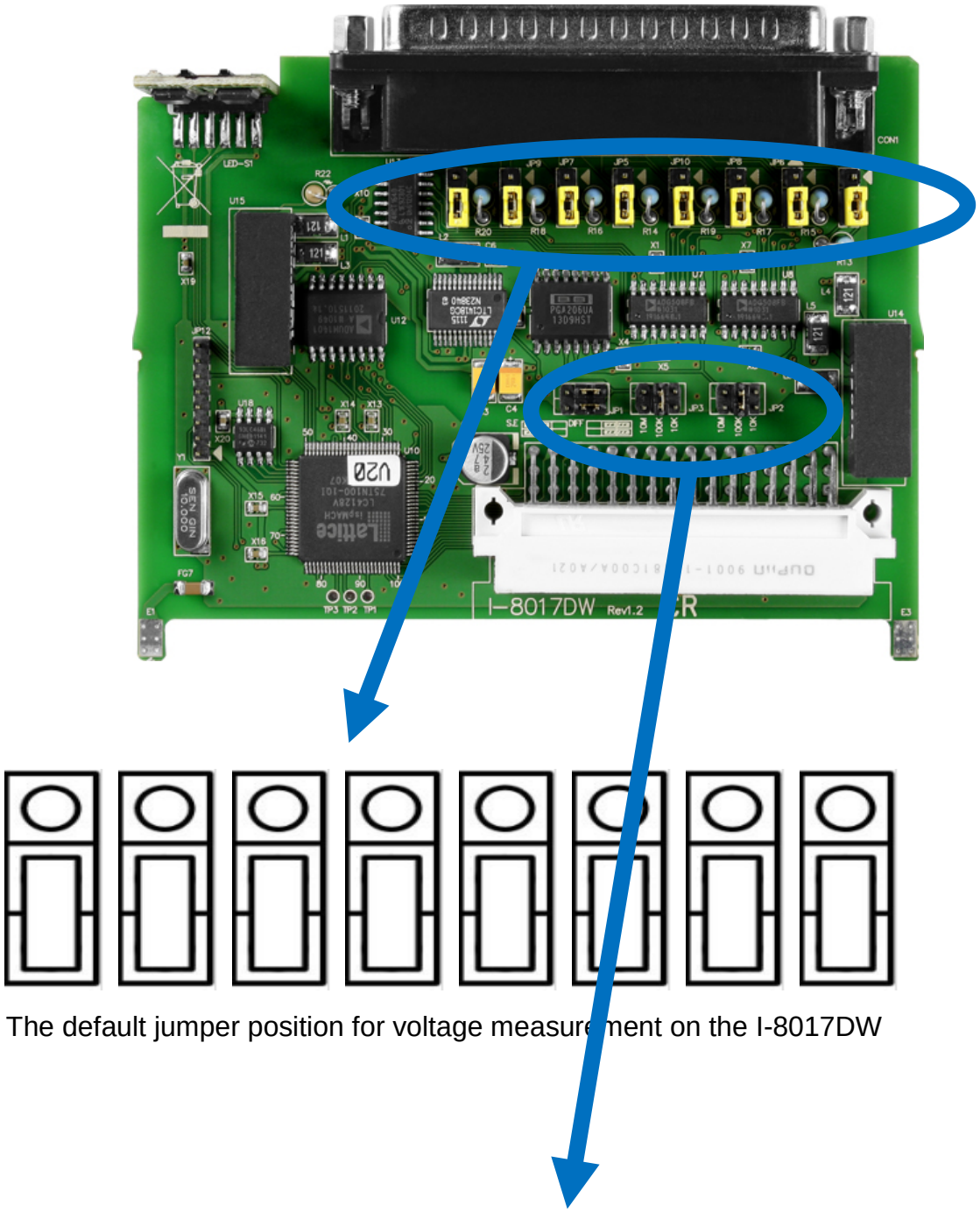
125Ω Resistor Jumper

By default, the I-8017HCW module is configured for current source measurement, and the I-8017DW is configured for voltage measurement, as illustrated below:





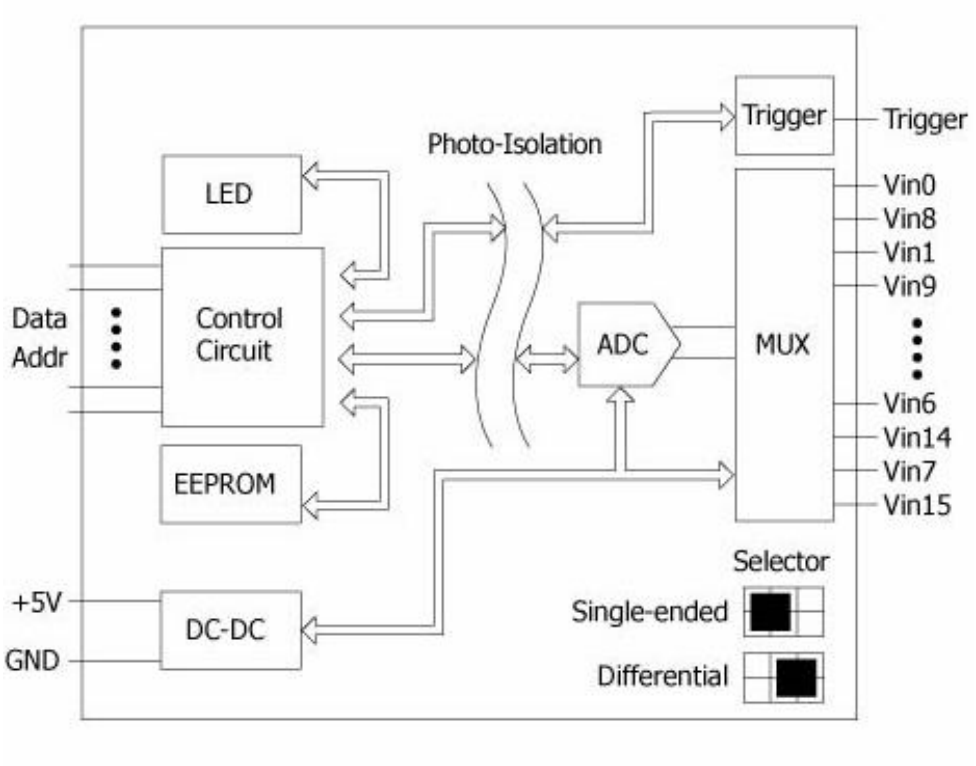
The default jumper position for current measurement on the I-8017HCW



The default jumper position for voltage measurement on the I-8017DW

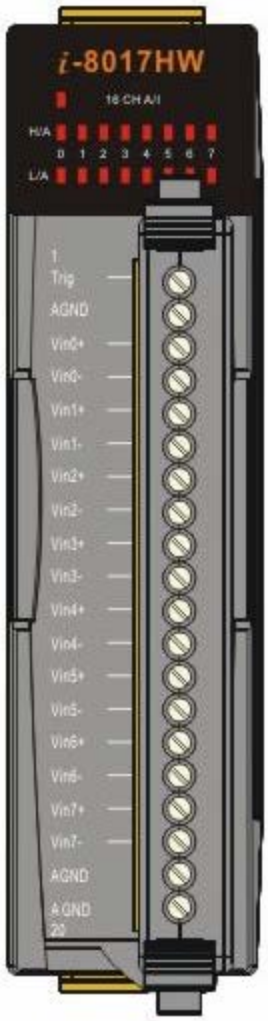
The jumper settings used to adjust both the input impedance and the single-ended and differential input on the I-8017HW are the same as those for the I-8017DW and the I-8017HCW.

1.3. I/O Structure



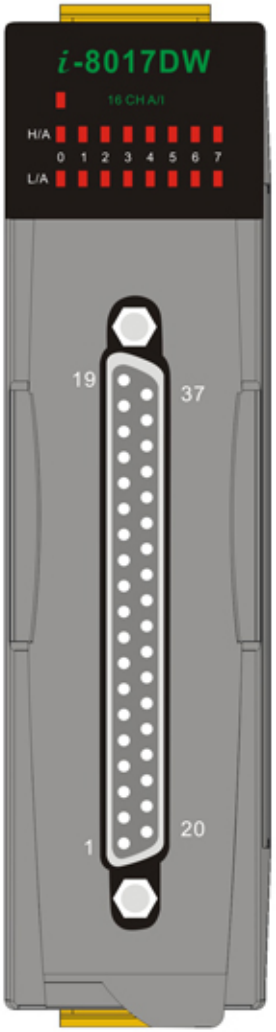
1.4. Pin Assignments

I-8017HW:



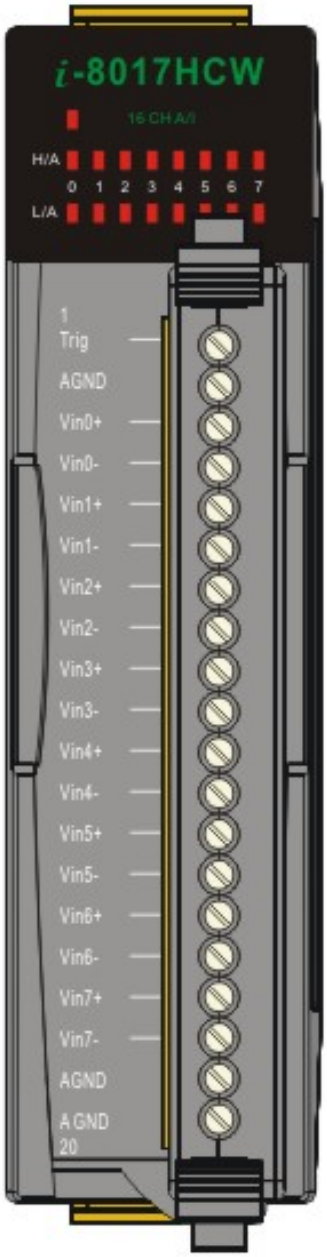
Terminal No.	Pin Assignment Name	
	Differential	Single-ended
01	Trig	Trig
02	AGND	AGND
03	Vin0 +	Vin0
04	Vin0 -	Vin8
05	Vin1 +	Vin1
06	Vin1 -	Vin9
07	Vin2 +	Vin2
08	Vin2 -	Vin10
09	Vin3 +	Vin3
10	Vin3 -	Vin11
11	Vin4 +	Vin4
12	Vin4 -	Vin12
13	Vin5 +	Vin5
14	Vin5 -	Vin13
15	Vin6 +	Vin6
16	Vin6 -	Vin14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

I-8017DW:



Pin Assignment		Terminal No.		Pin Assignment	
Differential	Single-ended			Differential	Single-ended
AGND	AGND	19	37	BK Sensor	BK Sensor
Trig	Trig	18	36	-	-
AI7-	Ai15	17	35	-	-
AI7+	AI7	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			

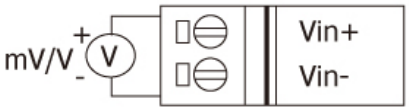
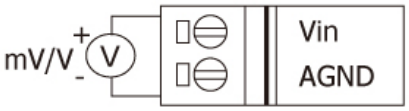
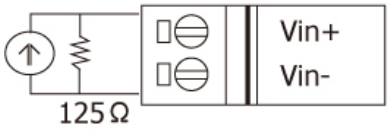
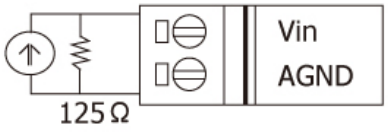
I-8017HCW:



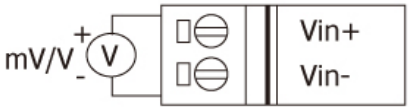
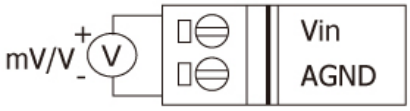
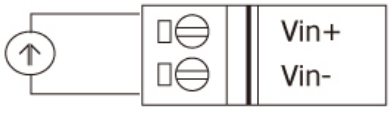
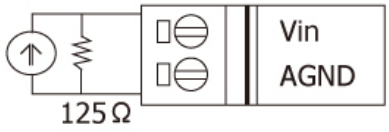
Terminal No.		Pin Assignment	
		Differential	Single-ended
	01	Trig	Trig
	02	AGND	AGND
	03	Vin0+	Vin0
	04	Vin0-	Vin8
	05	Vin1+	Vin1
	06	Vin1-	Vin9
	07	Vin2+	Vin2
	08	Vin2-	Vin10
	09	Vin3+	Vin3
	10	Vin3-	Vin11
	11	Vin4+	Vin4
	12	Vin4-	Vin12
	13	Vin5+	Vin5
	14	Vin5-	Vin13
	15	Vin6+	Vin6
	16	Vin6-	Vin14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

1.5. Wire Connections

I-8017HW:

Input Type	Differential	Single-ended
Voltage Input Wiring		
Current Input Wiring		
Note: When connecting to a current source, an optional external 125Ω resistor is required.		

I-8017DW and I-8017HCW:

Input Type	Differential	Single-ended
Voltage Input Wiring		
Current Input Wiring		
Note : 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 Ohm resistor is required.		

1.6. Location of the Demo Programs

ICP DAS provides a range of demo programs for different platforms that can be used to verify the functions of the module. The source code contained in these programs can also be reused in your own custom programs if needed. The following is a list of the locations where both the demo programs and associated libraries can be found on either the ICP DAS web site or the enclosed CD.

Platform	Location
For I-8000 devices using the MiniOS7 platform. On the Web:	
Library	ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/8000/841x881x/demo/lib/
Demo	ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/8000/841x881x/demo/io_in_slot/
For I-8000 devices using the MiniOS7 platform. On the CD:	
Library	CD:\Napdos\8000\841x881x\demo\Lib
Demo	CD:\Napdos\8000\841x881x\demo\IO_in_Slot
For iPAC-8000 devices using the MiniOS7 platform. On the Web:	
Library	ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/ipac8000/demo/basic/ip-84x1_ip-88x1/lib/
Demo	ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/ipac8000/demo/basic/ip-84x1_ip-88x1/io_in_slot/
For iPAC-8000 devices using the MiniOS7 platform. On the CD:	
Library	CD:\Napdos\iPAC8000\Demo\Basic\iP-84x1_iP-88x1\Lib

Demo	CD:\Napdos\iPAC8000\Demo\Basic\iP-84x1_iP-88x1\IO_in_Slot
For devices using Windows CE5 platform. On the Web:	
Library	ftp://ftp.icpdas.com/pub/cd/winpac/napdos/wp-8x4x_ce50/sdk/io_modules/
Demo	ftp://ftp.icpdas.com/pub/cd/winpac/napdos/wp-8x4x_ce50/demo/winpac/evc/pac_io/local/ (eVC demo) ftp://ftp.icpdas.com/pub/cd/winpac/napdos/wp-8x4x_ce50/demo/winpac/dotnet/c%23.net/pac_io/local/ (C# demo)
Platform	**Location**
For devices using the Windows CE5 platform. On the CD:	
Library	CD:\napdos\wp-8x4x_ce50\sdk\IO_Modules
Demo (eVC & C#)	CD:\napdos\wp-8x4x_ce50\Demo\WinPAC\eVC\PAC_IO\Local CD:\napdos\wp-8x4x_ce50\Demo\WinPAC\DOTNET\C#.NET\PAC_IO\Local
For devices using the Windows CE6 platform. On the Web:	
XP-8000-CE6	ftp://ftp.icpdas.com/pub/cd/xp-8000-ce6/sdk/special_io/ ftp://ftp.icpdas.com/pub/cd/xp-8000-ce6/demo/xpac/vc2008/io/local/ ftp://ftp.icpdas.com/pub/cd/xp-8000-ce6/demo/xpac/c%23/io/local/
XP-8000-Atom-CE6	ftp://ftp.icpdas.com/pub/cd/xpac-atom-ce6/sdk/special_io/ ftp://ftp.icpdas.com/pub/cd/xpac-atom-ce6/demo/xpac/vc2008/io/local/ ftp://ftp.icpdas.com/pub/cd/xpac-atom-ce6/demo/xpac/c%23/io/local/
For devices using the Windows CE6 platform. On the CD:	

XP-8000-CE6	CD:\SDK\Special_IO CD:\Demo\XPAC\VC2008\IO\Local CD:\Demo\XPAC\C#\IO\Local
XP-8000-Atom-CE6	CD:\SDK\Special_IO CD:\Demo\XPAC\VC2008\IO\Local CD:\Demo\XPAC\C#\IO\Local
For devices using the Windows Embedded Standard (WES) platform. On the Web:	
XP-8000	ftp://ftp.icpdas.com/pub/cd/xp-8000/sdk/io/ ftp://ftp.icpdas.com/pub/cd/xp-8000/demo/xpac/
XP-8000-Atom	ftp://ftp.icpdas.com/pub/cd/xpac-atom/sdk/io/ ftp://ftp.icpdas.com/pub/cd/xpac-atom/demo/xpac/
For devices using the Windows Embedded Standard (WES) platform. On the CD	
XP-8000	CD:\SDK\IO CD:\Demo\ XPAC
XP-8000-Atom	CD:\SDK\IO CD:\Demo \XPAC

Note: The library, dll and compiled executable files for each platform are only suitable for that specific platform. For example, the i-8000 library and compiled demo programs will not on an iP-8000 PAC device and will cause an error

2. Quick Start

This section provides a **Getting Started** guide to help users understand the operation process when using each of the modules on either the MiniOS7 or the Windows platforms.

- For MiniOS7-based Controllers, see Section 2.1 (i-8000 and iPAC-8000 modules)
- For Windows-based Controllers, see Section 2.2 (CE5, CE6 and WES modules)

2.1. MiniOS7-based Controllers

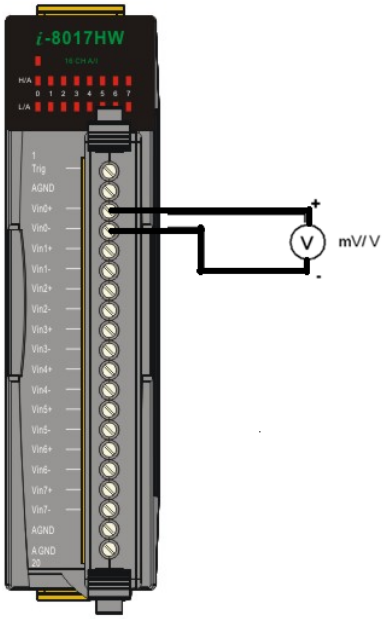
The 8017ai.exe executable file, which is located in the *8017h_ReadAI* folder of the demo programs, can be used to retrieve the basic configuration information related to the module and to verify the AI read functions. The basic configuration information includes:

- The version number and the published date of the library.
- The FPGA version
- The Differential/Single-ended jumper settings
- The Gain and Offset values for each input range
- The data read from each channel

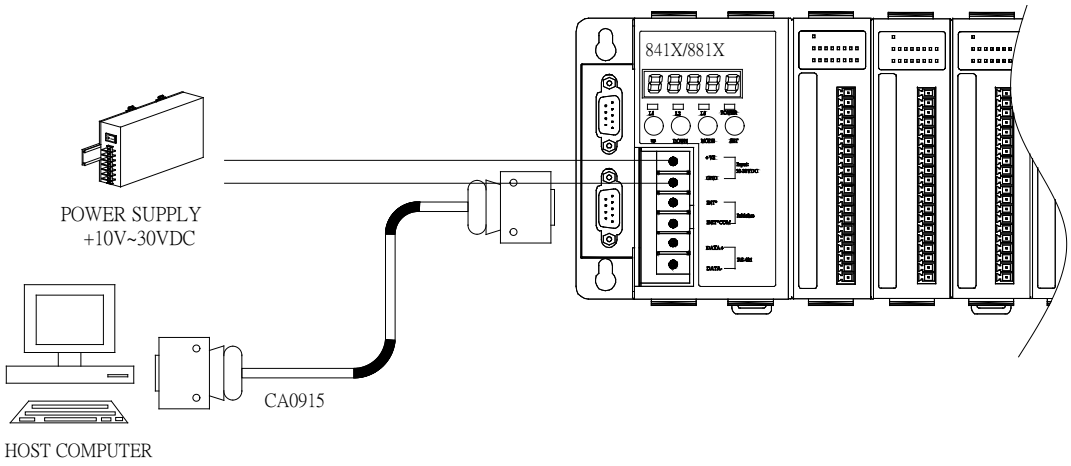
(See the Location of the Demo Programs section for details of where to find the 8017ai.exe file in the demo programs folder)

Step 1. Refer to the Jumper Settings section. Ensure that the Differential/Single-ended selection jumper is in the Differential position.

Step 2. Connect a stable signal source to the module (e.g., a battery output) using the differential wiring method, as illustrated below.



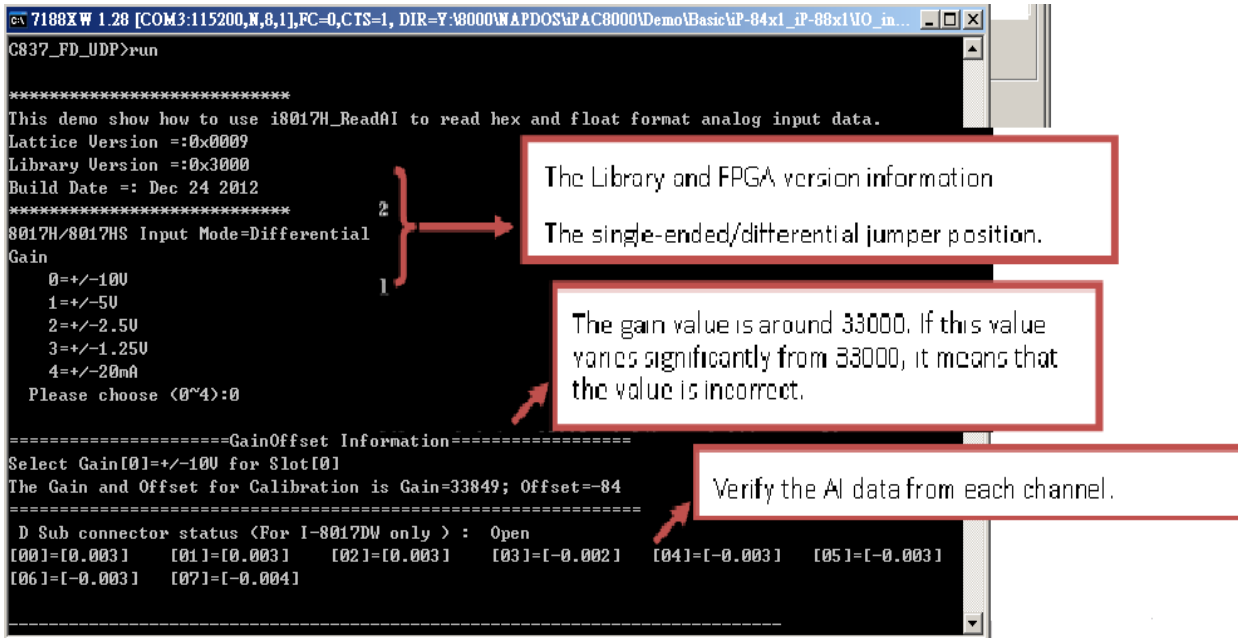
Step3. Connect the power supply to the module, and connect the control unit to the Host PC using an RS-232 cable.



Step4. Launch the 8017ai.exe executable file on the Host PC, and then verify that the basic information and the AI data from each channel is correct, as indicated in the diagram below:

Tips & Warnings

Unused channels should be connected to GND to avoid floating, it may get some noise values.

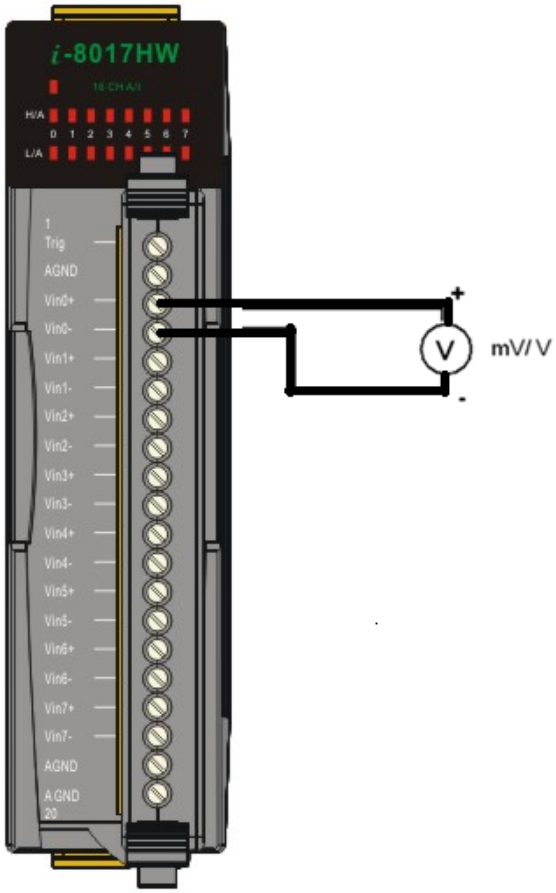


2.2. Windows-based Controllers

The pac_i8017HW_Utility.exe executable file, which is located in the *pac_i8017HW_Utility* folder of the demo programs, can be used to retrieve the basic configuration information related to the module, and to verify the AI read functions. The basic configuration information includes:

- The version number and the published date of the library.
- The FPGA version
- The Differential/Single-ended jumper settings
- The Gain and Offset values for each input range
- The data read from each channel

(See the Location of the Demo Programs section for details of where to find the pac_i8017HW_Utility.exe file in the demo programs folder)



Step 1. Refer to the Jumper Settings section. Ensure that the Differential/Single-ended selection jumper is in the Differential position.

Step 2. Connect a stable signal source to the module (e.g., a battery output) using the differential wiring method.

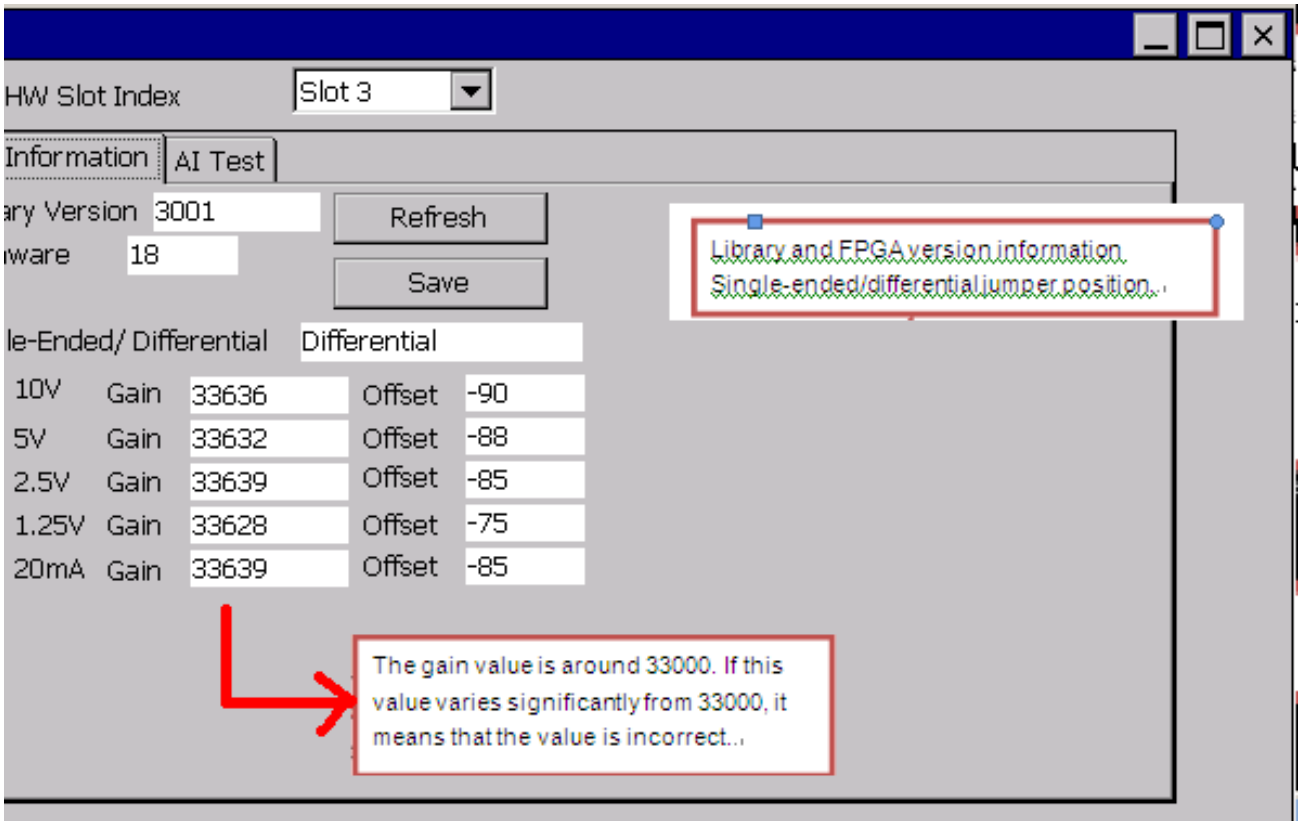
Step 3. Insert the module into a vacant slot in the control unit and power on the PAC controller

Step 4. Launch the pac_i8017HW_Utility.exe executable file on the controller, and then verify that the basic information and the AI data read from each channel is correct, as indicated in the diagram below:

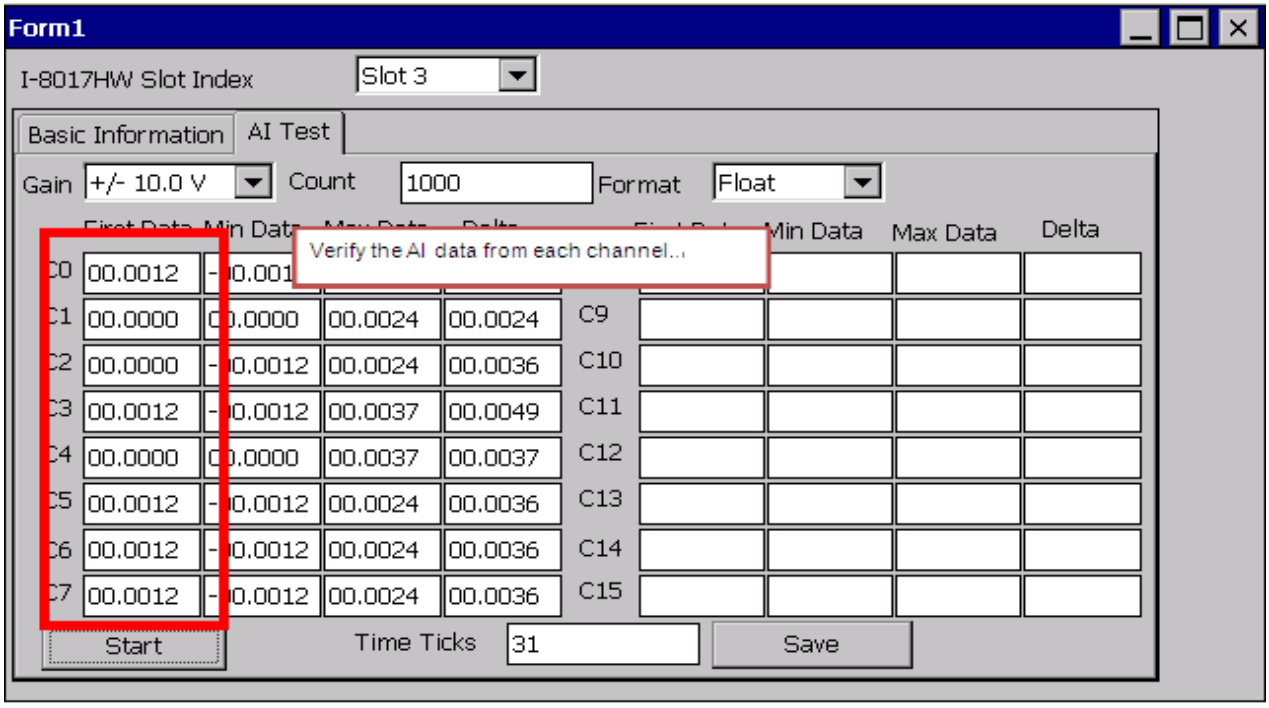
Tips & Warnings

Unused channels should be connected to GND to avoid floating, it may get some noise values..

Read the FPGA version information and the Gain and Offset values for each voltage range



Read the AI information from each channel



3. API for Windows-based Controllers

Function List

Function	Description
pac_i8017HW_GetLibVersion	Used to read the version and build information for the currently installed Library
pac_i8017HW_GetLibDate	Used to read the build date information for the currently installed Library
pac_i8017HW_GetFirmwareVersion	Used to read the firmware (FPGA) version information
pac_i8017HW_Init	Used to initialize the module
pac_i8017HW_SetLED	Used to set the LEDs
pac_i8017HW_GetSingleEndJumper	Used to read the status of the input jumper (Differential or Single-ended mode)
pac_i8017HW_ReadAI	Used to read the Analog Input value from a specific channel in float format
pac_i8017HW_ReadAIHex	Used to read the Analog Input value from a specific channel in 16-bit hexadecimal format

3.1. pac_i8017HW_GetLibVersion

This function is used to read the version and build information for the Library currently installed on the I-8017 module inserted in a specific slot.

Syntax

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

Parameters

This function is not passed any parameters

Return Values

The version number and build information for the Library used by the module.

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

Examples:

[C++]

```
short version = pac_i8017HW_GetLibVersion();
Print("\n Library Version =: %04X",i8017HW_GetLibVersion());
```

[C#]]

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

3.2. pac_i8017HW_GetLibDate

This function is used to read the build date information for the Library currently installed on the I-8017 module inserted in a specific slot

Syntax

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

Parameters

*LibDate:

[out] The release date of I-8017 library

Return Values

None

Examples

[C++]

```
char* builtDate;
pac_i8017HW_GetLibDate(&builtDate);
```

[C#]

```
string builtDate;;
pac_i8017HW_GetLibDate(ref builtDate);
```

3.3. pac_i8017HW_GetFirmwareVersion

This function is used to read the firmware (FPGA) version information for the I-8017 module inserted in a specific slot.

Syntax

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

Parameters

slot:

[in] Specifies the slot where the I-8017 module is inserted. Valid range = 0 to 7

* version:

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

Return Values

0 = No Error.

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

Examples

[C++]

```
int slot;
short ret, firmware;
ret = pac_i8017HW_GetFirmwareVersion(slot, &firmware);
```

[C#]

```
int slot;
Int16 firmware = 0;
Int16 ret = pac_i8017HW_GetFirmwareVersion(slot, ref firmware);
```

3.4. pac_i8017HW_Init

This function is used to initialize the I-8017 module inserted in a specified slot, and must be called at least once before using any other function.

Syntax

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

Parameters

slot:

[in] Specifies slot where the I-8017 module is inserted. Valid range = 0 to 7

Return Values

0 = No Error.

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

Examples

[C++]

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

[C#]

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

3.5. pac_i8017HW_SetLED

This function is used to set the LEDs for the I-8017 module inserted in a specified slot to ON or OFF. The LEDs can also be used to act as an alarm.

Syntax

```
short pac_i8017HW_SetLED(int slot, unsigned short ledValue);
```

Parameters

slot:

[in] Specifies slot where the I-8017 module is inserted. Valid range = 0 to 7

ledValue:

[in] Specifies the LED to be set. Valid range = 0 to 0xFFFF

Return Values

0 = No Error.
Others: Refer to Appendix A: “Error Code Definitions” for more details.

Examples

[C++]

```
int slot;
unsigned short led;
short ret = pac_i8017HW_Init(int slot);
if (ret == 0)
pac_i8017HW_SetLED(slot, led);
```

[C#]

```
int slot;
UInt16 led;
Int16 ret = pac8017HW.Init(slot);
If (ret == 0)
pac8017HW.SetLED(slot, led);
```

3.6. pac_i8017HW_GetSingleEndJumper

This function is used to read whether the input jumper for the I-8017 module inserted in a specified slot is set to either Differential or Single-ended mode.

Syntax

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

Parameters

slot:

[in] Specifies the slot where the I-8017 module is inserted. Valid range = 0 to 7

*selectJumper:

[out] To get Differential Mode/Single-ended Mode in specifies slot
0: Differential Mode
1: Single-ended Mode

Return Values

0 = No Error.
Others: Refer Refer to Appendix A: “Error Code Definitions” for more details.

Examples

[C++]

```
int slot=1;
short Jumper = pac_i8017HW_GetSingleEndJumper(slot);
```

[C#]

```
int slot;
Int16 Jumper = pac8017HW. SingleEndJumper(slot);
```

3.7. pac_i8017HW_ReadAI

This function is used to read the Analog Input value in float format from a specific channel of the I-8017 module inserted in a specified slot according to a given Gain value

Syntax

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

Parameters

iSlot:

[in] Specifies the slot where the I-8017 module is inserted. Valid range = 0 to 7

iChannel:

[in] Specifies the channel number
0 to 7 if using Differential Mode
0 to 15 if using Single-ended Mode

iGain:

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

*fValue:

[out] To get analog input value in float format.

Return Values

0 = No Error.

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

Examples

[C++]

```
int slot=1, ch=0, gain=1;
float data;
pac_i8017HW_Init(slot);
short ret = pac_i8017HW_ReadAI(slot,ch,gain,&data);
```

[C#]

```
int slot=1, ch=0, gain=1;
float data;
pac8017HW.Init (slot);
Int16 ret= pac8017HW. ReadAI(slot, ch, gain,ref data);
```

3.8. pac_i8017HW_ReadAIHex

This function is used to read the Analog Input value in 16-bit hexadecimal format from a specific channel of the I-8017 module inserted in a specified slot according to a given Gain value.

Syntax

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

Parameters

iSlot:

[in] Specifies the slot where the I-8017 module is inserted. Valid range = 0 to 7

iChannel:

[in] Specifies the channel number
0 to 7 if using Differential Mode
0 to 15 if Single-ended Mode

iGain:

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

*iValue:

[out] To get a 16 bits analog input value in hexadecimal format

Return Values

0 = No Error.

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

Examples

[C++]

```
int slot=1, ch=0, gain=1;
short* data;
pac_i8017HW_Init(slot);
short ret = pac_i8017HW_ReadAIHex(slot,ch,gain,&data);
```

[C#]

```
int slot=1, ch=0, gain=1;
float data;
pac8017HW.Init (slot);
Int16 ret=    pac8017HW. ReadAIHex(slot, ch, gain,ref data);

Note: the I-8017HW, I-8017DW and I-8017HCW modules use a 14-bit AD chip, so it is more convenient for the pac_i8017HW_ReadHex function to return 16-bit data when the user needs to scale the hexadecimal data. Consequently, there will be least two bits of invalid data.

The following code can be used to convert 16-bit data to 14-bit.

short Convert16To14(short bit16Data)
{
short bit14Data= bit16Data >> 2;
bit14Data &= 0x3fff;
return bit14Data;
}
```

4. API for MiniOS7-based Controller

Function List

Function	Description
i8017H_GetLibVersion	Used to read the version and build information for the currently installed Library
i8017H_GetLibDate	Used to read the built date information for the currently installed Library
i8017H_GetFirmwareVersion	Used to read the firmware (FPGA) version information
i8017H_Init	Used to initialize the module
i8017H_ReadGainOffset_Info	Used to read the calibrated Gain and Offset values
i8017H_GetSingleEndJumper	Used to read the status of the input jumper (Differential or Single-ended Mode)
i8017H_SetLED	Used to set the LEDs
i8017H_ReadAI	Used to read the Analog Input value from a specific channel in float format
i8017H_ReadAIHex	Used to read the Analog Input value from a specific channel in 16-bit hexadecimal format
i8017H_Set_ChannelGainMode	Used to set the MUX configuration
i8017H_Get_AD_FValue	Used to read the A/D value in float format
i8017H_Get_AD_HValue	Used to read the A/D value in 14-bit hexadecimal format
i8017H_AD_Polling	Used to set the polling mode to sample A/D values from a single channel
i8017H_AD_TimerINT	Used to set the timer interrupt to sample A/D values from a single channel
i8017H_AD_TimerINT_Scan	Used to set the timer interrupt to sample A/D values from all channels

4.1. i8017H_GetLibVersion

This function is used to read the version and build information for the Library currently installed on the I-8017 module inserted in a specific slot

Syntax

```
short i8017H_GetLibVersion(void);
```

Parameters

This function is not passed any parameters.

Return Values

The version number and build information for the Library used by the module.
Example: 0x106; = Rev:1.0.6

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

Example

[C++]

```
short version = i8017H_GetLibVersion();
```

4.2. i8017H_GetLibDate

This function is used to read the build date information for the Library currently installed on the I-8017 module inserted in a specific slot

Syntax

```
void i8017H_GetLibDate(char libDate[]);
```

Parameters

*LibDate:

[out] The release date of I-8017 library

Return Values

0 = No Error.
Others: Refer to Appendix A: “Error Code Definitions” for more details

Examples

[C++]

```
char* builtDate;
i8017H_GetLibDate(&builtDate);
```

4.3. i8017H_GetFirmwareVersion

This function is used to read the firmware (FPGA) version information for the I-8017 module inserted in a specific slot.

Syntax

```
short i8017H_GetFirmwareVersion(int iSlot, short* version);
```

Parameters

iSlot

[in] Specifies the slot where the I-8017 module is inserted. Valid range = 0 to 7

* version:

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

Return Values

0 = No Error.

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

Examples

[C++]

```
int slot;
short ret, firmware;
ret = i8017H_GetFirmwareVersion(slot, &firmware);
```

4.4. i8017H-Init

This function is used to initialize the I-8017 module inserted in a specific slot, and must be called at least once before using any other function.

Syntax

```
void i8017H_Init(int iSlot);
```

Parameters

iSlot

[in] Specifies the slot where the I-8017 module is inserted. Valid range = 0 to 7

Return Values

0 = No Error.
Others: Refer to Appendix A: “Error Code Definitions” for more details.

Examples

[C++]

```
int slot;
short ret = i8017H_Init(slot);
```

4.5. i8017H_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

```
short i8017H_ReadGainOffset_Info
(
  int iSlot,
  int iGain,
  unsigned short* iGainValue,
  short* iOffsetValue
);
```

Parameters

iSlot:

[in] Specifies the slot where the I-8017 module is inserted. Valid range = 0 to 7

iGain:

[in] Specifies the input range

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

* *iGainValue*:

[out] The calibrated Gain value

* *iOffsetValue*:

[out] The calibrated Offset value

Return Values

0 = No Error.

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

Examples

[C++]

```
int slot=1, ch=0, gain=1;
unsigned short cgain;
short coffset;
i8017H_Init(slot);
short ret = ReadGainOffset_Info (slot,ch,gain,& cgain,&coffset);
```

4.6. i8017H_GetSingleEndJumper

This function is used to read whether the input jumper for the I-8017 module inserted in a specified slot is set to either Differential or Single-ended mode.

Syntax

```
short i8017H_GetSingleEndJumper(int iSlot, short* selectJumper);
```

Parameter

iSlot

[in] Specifies slot where the I-8017 module is inserted. Valid range = 0 to 7

**selectJumper*

[out] The position that the jumper is set in
0: Differential Mode
1: Single-ended Mode

Return Values

0 = No Error.
Others: Refer to Appendix A: “Error Code Definitions” for more details

Examples

[C++]

```
int slot=1;
short Jumper = i8017H_GetSingleEndJumper(slot);
```

4.7. i8017H_SetLED

This function is used to set the LEDs for the I-8017 module inserted in a specified slot to ON or OFF. The LEDs can also be used to act as an alarm.

Syntax

```
void i8017H_SetLED(int iSlot, unsigned short iLedValue);
```

Parameters

iSlot

[in] Specifies slot where the I-8017 module is inserted. Valid range = 0 to 7

iLedValue

[in] Specifies the LED ON/ OFF status ranged from 0 to 0xFFFF

Return Values

0 = No Error.
Others: Refer to Appendix A: “Error Code Definitions” for more details.

Examples

[C++]

```
int slot;
unsigned short led;
short ret = i8017H_Init(int slot);
if (ret == 0)
i8017H_SetLED(slot, led);
```

4.8. i8017H_ReadAI

This function is used to read the Analog Input value in float format from a specific channel of the I-8017 module inserted in a specified slot according to a given Gain value.

Syntax

```
short i8017H_ReadAI
(
  int iSlot,
  int iChannel,
  int iGain,
  float* fValue
);
```

Parameters

iSlot:

[in] Specifies the slot where the I-8017 module is inserted. Valid range = 0 to 7

iChannel:

[in] Specifies the channel number
0 to 7 if using Differential Mode
0 to 15 if using Single-ended Mode

iGain:

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

* fValue:

[out] The analog input data in float format

Return Values

0 = No Error.

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

Examples

[C++]

```
int slot=1, ch=0, gain=1;
float data;
i8017H_Init(slot);
short ret = i8017H_ReadAI(slot,ch,gain,&data);
```

4.9. i8017H_ReadAIHex

This function is used to read the Analog Input value in 16-bit hexadecimal format from a specific channel of the I-8017 module inserted in a specified slot according to a given Gain value.

Syntax

```
short i8017H_ReadAIHex
(
  int iSlot,
  int iChannel,
  int iGain,
  short* iValue
);
```

Parameters

iSlot:

[in] Specifies the slot where the I-8017 module is inserted. Valid range = 0 to 7

iChannel:

[in] Specifies the channel number

0 to 7 if using Differential Mode

0 to 15 if using Single-ended Mode

iGain:

[in] Specifies the input range

0: +/- 10.0V

1: +/- 5.0V

2: +/- 2.5V

3: +/- 1.25V

4: +/- 20mA

* iValue:

[out] The analog input data in 16-bit hexadecimal format

Return Values

0 = No Error.

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

Examples

[C++]

```
int slot=1, ch=0, gain=1;
short* data;
i8017H_Init(slot);
short ret = i8017H_ReadAIHex(slot,ch,gain,&data);
```

4.10. i8017H_Set_ChannelGainMode

This function is used to configure the Analog Input channel range for the I-8017 module inserted in a specified slot prior to using the ADC (analog to digital converter).

Tips & Warnings

If the channel, range or mode to be sampled is different to a previous sampling, this function must be called first to change the MUX configuration.

Note that calling this function only needs to be considered when using the following functions:

1. i8017H_Get_AD_FValue

2. i8017H_Get_AD_FValue

This is because other functions that are used to sample multiple data signals already include a function to configure the MUX.

Syntax

```
void i8017H_Set_ChannelGainMode
(
int iSlot,
int iChannel,
int iGain,
int iMode
);
```

Parameters

iSlot

[in] Specifies the slot where the I-8017 module is inserted. Valid range = 0 to 7

iChannel

[in] Specifies the channel number

0 to 7 if using Differential Mode

0 to 15 if using Single-ended Mode

iGain

[in] Specifies the input range

0: +/- 10.0V

1: +/- 5.0V

2: +/- 2.5V

3: +/- 1.25V

4: +/- 20mA

iMode

[in] Specifies the Mode to be used

0 = Polling Mode

1 = Timer Interrupt Mode

(Use 0 for the i8017_Get_AD_Value and i8017_Get_AD_Value_Hex functions)

Return Values

0 = No Error.

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

Examples

[C++]

```
int slot=1, ch=0, gain=1,mode=0;
short* data;
i8017H_Init(slot);
short ret = i8017H_Set_ChannelGainMode (slot,ch,gain,mode);
```

4.11. i8017H_Get_AD_FValue

This function is used to read the A/D value for both voltage and current from the I-8017 module inserted in a specified slot in float format.

Syntax

```
float i8017H_Get_AD_FValue(int iGain);
```

Parameters

iGain

Specifies the input range

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

Return Values

The analog input value in float format.

Examples

[C++]

```
int gain=1;
float data;
i8017H_Init(slot);
data = i8017H_Get_AD_FValue (gain);
```

4.12. i8017H_Get_AD_HValue

This function is used to read the A/D value for both voltage and current from the I-8017 module inserted in a specified slot in hexadecimal format.

The resolution for the I-8017HW, I-8017DW and I-8017HCW modules is 14-bit and uses 14-bit 2's Complement format to store values.

When the A/D value is between 0000 and 1FFF → 0 to + max. value

When the A/D value is between 2000 and 3FFF → - max. value to 0 (a little less than 0)

The algorithm used to convert a 14-bit integer value (iValue) to a float value (fValue) is as follows:

fValue = iValue/8192*span,

fValue >= 0, iValue between 0000 and 1FFF

fValue = (8192-iValue)/8192*span,

fValue < 0, iValue between 2000 and 3FFF

Syntax

```
int i8017H_Get_AD_HValue(void);
```

Parameters

None

Return Values

The analog input value in 14-bits hexadecimal format..

Examples

[C++]

```
int gain=1;
int data;
i8017H_Init(slot);
data = i8017H_Get_AD_HValue (gain);
```

4.13. i8017H_AD_Polling

This function is used to set the polling mode to sample the A/D values from a single channel of the I-8017 module inserted in a specified slot.

Tips & Warnings

The data is an uncalibrated 16-bit value.

ARRAY_HEX_TO_FLOAT or HEX_TO_FLOAT can be used to calibrate the data and convert it to a float value.

Syntax

```
int i8017H_AD_Polling
(
  int iSlot,
  int iChannel,
  int iGain,
  int iDataCount,
  int *iDataPointer
);
```

Parameters

iSlot

[in] Specifies the slot where the I-8017 module is inserted. Valid range = 0 to 7

iChannel

[in] Specifies the channel number

0 to 7 if using Differential Mode

0 to 15 if using Single-ended Mode

iGain

[in] Specifies the input range

0: +/- 10.0V

1: +/- 5.0V

2: +/- 2.5V

3: +/- 1.25V

4: +/- 20mA

iDataCount

[in] Specifies the data length to read in, valid length is ranged from 1 to 8192

* iDataPointer:

[out] The pointer to the data buffer

Return Values

0 = No Error.

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

4.14. i8017H_AD_TimerINT

This function is used to set the timer interrupt to sample the A/D values from a single channel of the I-8017 module inserted in a specified slot.

Tips & Warnings

The data is an uncalibrated 16-bit value.

ARRAY_HEX_TO_FLOAT or HEX_TO_FLOAT can be used to calibrate the data and convert it to a float value.

Syntax

```
int i8017H_AD_TimerINT
(
  int iSlot,
  int iChannel,
  int iGain,
  int iDataCount,
  unsigned int iSampleCLK,
  int *iDataPointer
);
```

Parameters

iSlot

[in] Specifies the slot where the I-8017 module is inserted. Valid range = 0 to 7

iChannel

[in] Specifies the channel number

0 to 7 if using Differential Mode

0 to 15 if using Single-ended Mode

iGain

[in] Specifies the input range

0: +/- 10.0V

1: +/- 5.0V

2: +/- 2.5V

3: +/- 1.25V

4: +/- 20mA

iDataCount

[in] Specifies the data length to read in, valid length is ranged from 1 to 8192

iSampleCLK

[in] Specifies the sampling rate ranged = 200 to 50K (units: Hz)

* iDataPointer:

[out] The pointer to the data buffer

Return Values

0 = No Error.

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

4.15. i8017H_AD_TimerINT_Scan

This function is used to set the timer interrupt to sample the A/D values from all 8 channels of the I-8017 module inserted in a specified slot.

Tips & Warnings

The data is an uncalibrated 16-bit value.

ARRAY_HEX_TO_FLOAT or HEX_TO_FLOAT can be used to calibrate the data and convert it to a float value.

Syntax

```
int i8017H_AD_TimerINT_Scan
(
  int iSlot,
  int iGain,
  int iDataCount,
  unsigned int iSampleCLK,
  int *iDataPointer
);
```

Parameters

iSlot

[in] Specifies the slot where the I-8017 module is inserted. Valid range = 0 to 7

iChannel

[in] Specifies the channel number
0 to 7 if using Differential Mode

0 to 15 if using Single-ended Mode

iGain

[in] Specifies the input range

0: +/- 10.0V

1: +/- 5.0V

2: +/- 2.5V

3: +/- 1.25V

4: +/- 20mA

iDataCount

[in] Specifies the data length to read in, valid length is ranged from 1 to 8192

iSampleCLK

[in] Specifies the sampling rate ranged = 200 to 50K (units: Hz)

* iDataPointer:

[out] The pointer to the data buffer

The data stored in the buffer is arranged as follows:

Differential Input Mode:

ch0 ch1 ch2...ch7 ch0 ch1....ch7 ch0 ch1 ...

Single-ended Input Mode:

ch0 ch1 ch2...ch7 ch0 ch1....ch15 ch0 ch1 ...

Return Values

0 = No Error.

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

5. Calibration

Each I-8017 module is factory calibrated and thoroughly tested and verified before shipment, so it is usually unnecessary to calibrate the module again unless the input impedance is changed on a calibrated module or the accuracy is lost.

All of 8 or 16 channels in an I-8017 module use one calibration parameter in the same input range, but different input range. User needs to calibrate different input range if using different input range for an I-8017 module.

+/-1.25 V and +/- 20mA use one calibration parameter. If user needs to use+/-20 mA input range, it needs to calibrate +/- 1.25V parameter.

For example:

A: User uses channel 0 and channel 1 with +/- 10V only, then only need to calibrate input range +-/ 10V for a I-8017 module.

B: Users uses ch0 with +/- 10V, ch1 with +/- 20 mA,

Then need to calibrate input range +/- 10V and +/-1.25V (+/-20 mA use +/-1.25V calibration parameter)

To calibrate the module, in addition to inserting the module into a controller slot, the following items 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. See the Location of the Calibration Demo Programs section below for details.

Tips & Warnings

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

2. If you wish to perform calibration using ± 20 mA, select ± 2.5 V instead as both types use the same Gain and Offset values.

3. The calibration program only uses channel 0 for input of the calibration source.

Select the appropriate usage platform to locate the relevant calibration demo program for the I-8017HW, I-8017DW and I-8017HCW modules.

Platform	Location
For I-8000 modules. On the Web:	
ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/8000/841x881x/demo/io_in_slot/8017h/calibration/	
For I-8000 modules. On the CD:	
CD:\Napdos\8000\841x881x\demo\IO_in_Slot\8017h\Calibration	
For iPAC-8000 modules. On the Web:	
ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/ipac8000/demo/basic/ip-84x1_ip-88x1/io_in_slot/8017h/calibration/	
For iPAC-8000 modules. On the CD:	
CD:\Napdos\iPAC8000\Demo\Basic\iP-84x1_iP-88x1\IO_in_Slot\8017h\Calibration	
For devices using the Windows CE5 Platform. On the Web:	
ftp://ftp.icpdas.com/pub/cd/winpac/napdos/wp-8x4x_ce50/demo/winpac/c%23/io/local/pac_i8017hw_dotnet/pac_i8017hw_calibration/ (C# demo)	

Platform	Location
For devices using the Windows CE5 Platform. On the CD:	
CD:\napdos\wp-8x4x_ce50\Demo\WinPAC\DOTNET\C#.NET\PAC_IO\Local\pac_i8017HW_Dotnet\pac_i8017HW_calibration	
For devices using the Windows CE6 Platform. On the Web:	
XP-8000-CE6: ftp://ftp.icpdas.com/pub/cd/xp-8000-ce6/demo/xpac/c%23/io/local/pac_i8017hw_dotnet/pac_i8017hw_calibration/ XP-8000-Atom-CE6: ftp://ftp.icpdas.com/pub/cd/xpac-atom-ce6/demo/xpac/c%23/io/local/pac_i8017hw_dotnet/pac_i8017hw_calibration/	
For devices using the Windows CE6 Platform. On the CD:	
XP-8000-CE6 CD:\Demo\XPAC\C#\IO\Local*pac_i8017HW_Dotnet\pac_i8017HW_calibration* XP-8000-Atom-CE6 CD:\Demo\XPAC\C#\IO\Local*pac_i8017HW_Dotnet\pac_i8017HW_calibration*	
For devices using the Windows Embedded Standard (WES) Platform. On the Web:	
XP-8000: ftp://ftp.icpdas.com/pub/cd/xp-8000/demo/xpac/csharp.net/io/local/windows_forms/pac_i8017hw_dotnet/pac_i8017hw_calibration/ XP-8000-Atom: ftp://ftp.icpdas.com/pub/cd/xpac-atom/demo/xpac/csharp.net/io/local/windows_forms/pac_i8017hw_dotnet/pac_i8017hw_calibration/	
For devices using the Windows Embedded Standard (WES) Platform. On the CD:	
XP-8000 CD:\Demo\XPAC\csharp.net\IO\Local\windows_forms\pac_i8017HW_Dotnet\pac_i8017HW_calibration XP-8000-Atom CD:\Demo\XPAC\csharp.net\IO\Local\windows_forms\pac_i8017HW_Dotnet\pac_i8017HW_calibration	

5.1. MiniOS7-based Controller

This section contains:

Calibrating on i-8000 and iPAC-8000 Units

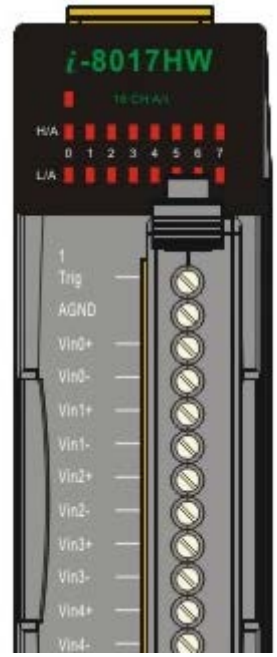
Step 1. Repeat Steps 1 to 3 as described in the Quick Start guide.

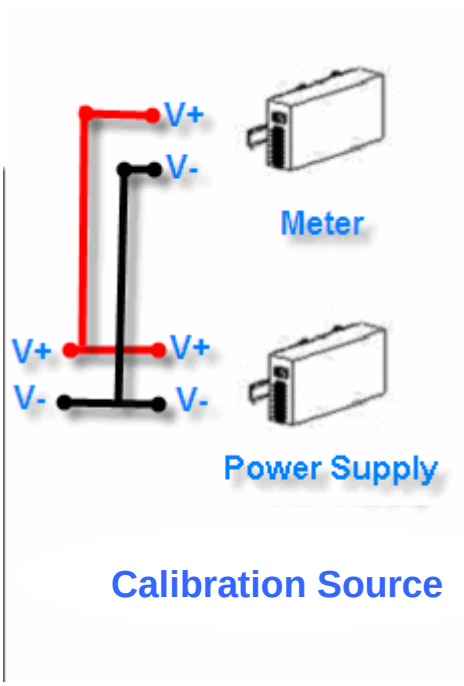
a. Attach the power supply to the control unit and then connect the control unit to the Host PC.

b. Set the Differential/Single-ended input jumper to the Differential position and connect the calibration source to **channel 0** using the differential wiring method.

c. Connect the meter, as illustrated in the following figure.

d. Turn on the control unit.



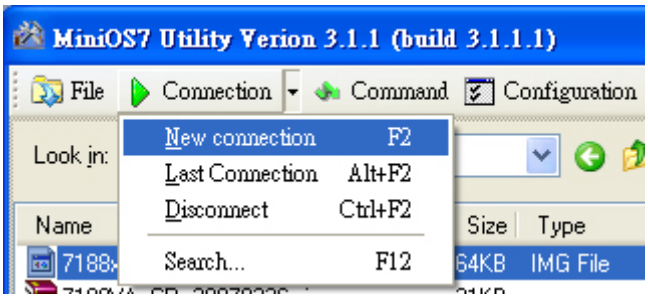


Step 2. Launch the MiniOS7 Utility on the Host PC. Upload the calibration program to the control unit and execute it.

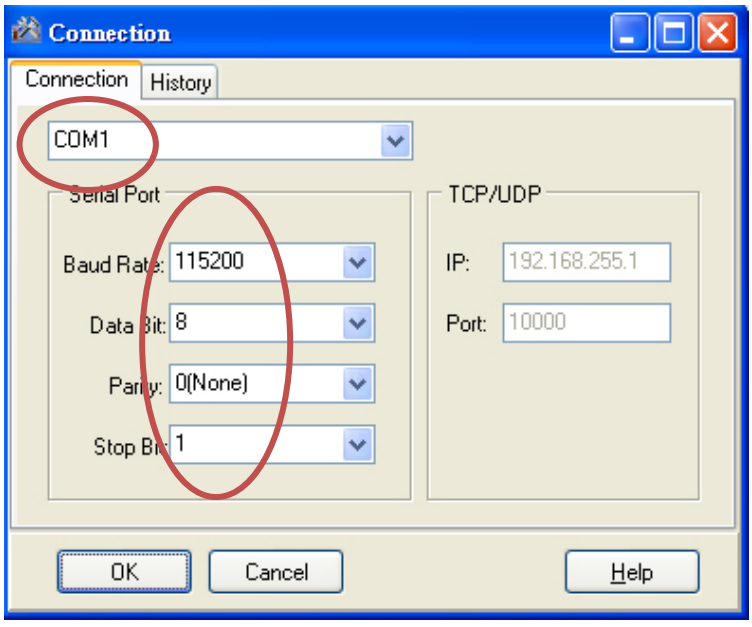
The MiniOS7 Utility can be downloaded from the web site shown below. Select the appropriate calibration program for your controller.

- MiniOS7 Utility: http://www.icpdas.com/download/minios7.htm
- 8017cal.exe: This is the calibration program for I-8000 units, which is located in the same folder as the demo programs. (See the Location of the Demo Programs section)
- iP_8017cal.exe: This is the calibration program for iP-8000 units, which is located in the same folder as the demo programs. (See the Location of the Demo Programs section)

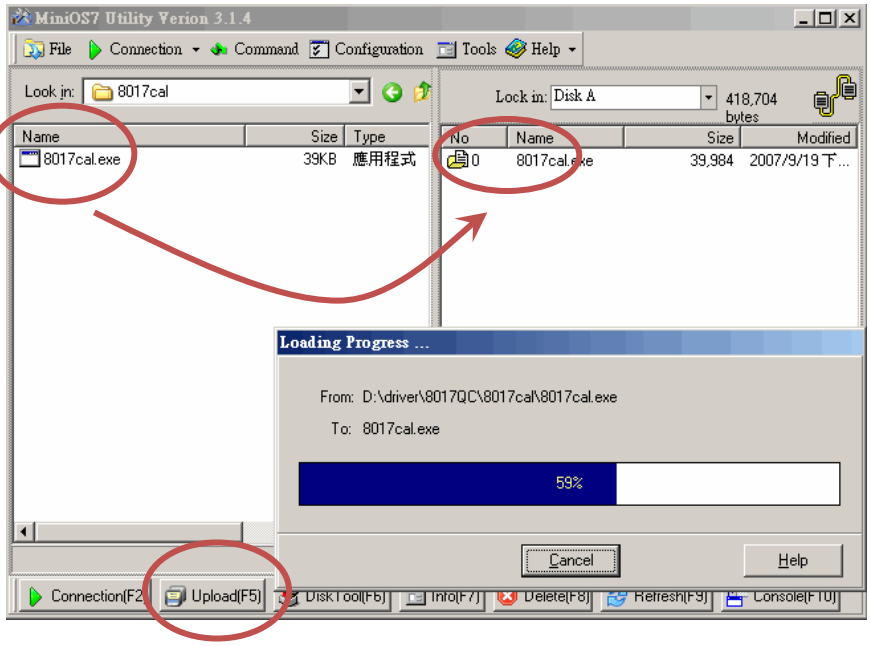
a. Launch the MiniOS7 Utility on the Host PC, and then choose **New Connection** from the **Connection** menu, or press **F2**.



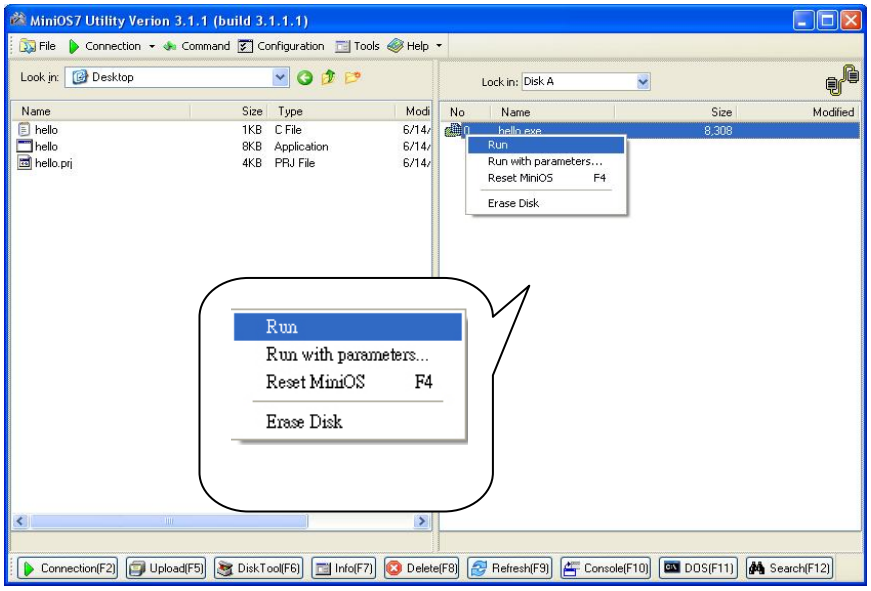
b. From the drop-down list, select the COM Port for the Host PC that is connected to the control unit, configure the communication parameters to match those indicated below, and then click the **OK** button.



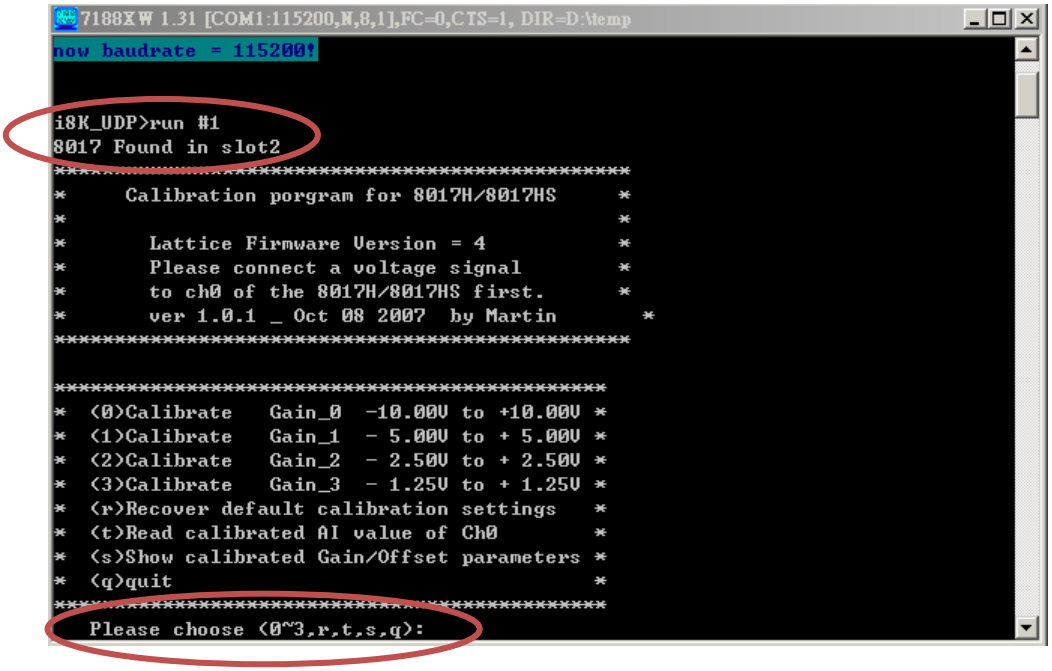
c. Select the name of the calibration program and then click the **Upload** button (or press **F5**) to upload the program to the MiniOS7 PAC unit.



d. Once the file has been uploaded, right-click the name of the updated calibration file and choose Run.

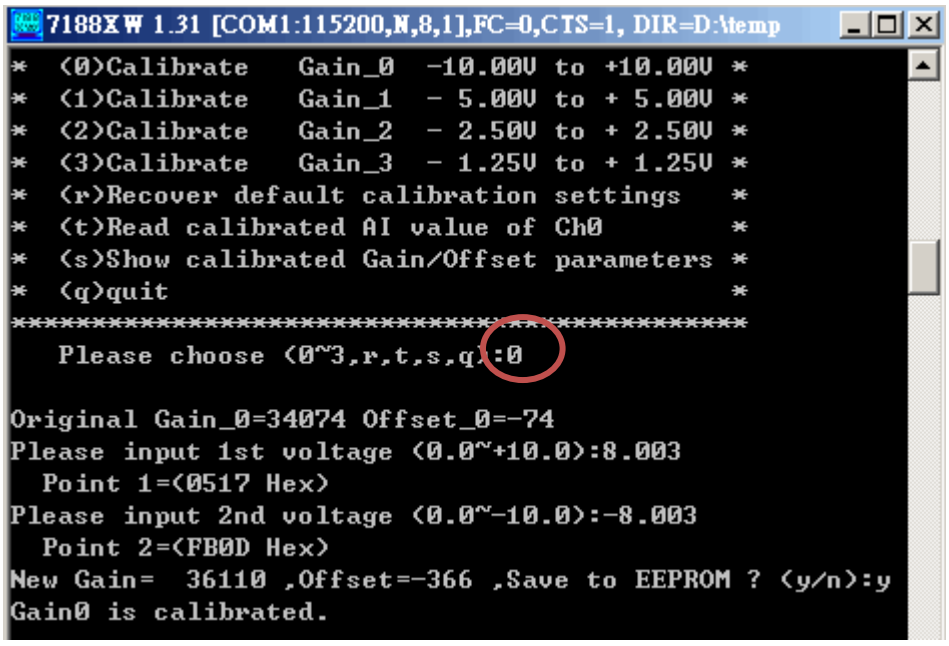


The calibration program will be executed on the control unit and 7188xw.exe will be executed on the Host PC to provide a command line interface.



Step 3. Calibrate the module using the following procedure.

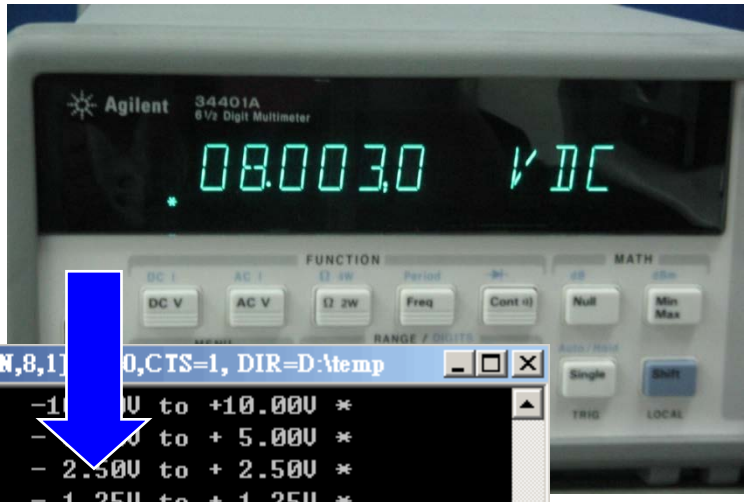
a. Select the required input type by typing an option from 0 to 3, and then press **Enter**.



b. Determine two values (points) within the range of the input type selected for the calibration process. For example, after selecting option 0 (-10 V to +10 V), +8 V and -8 V can be used as the two calibration points.

c. Set the calibration source output to one of the two points (e.g., 8 V in this example)

d. At the "**Input 1st voltage**" prompt on the console, type the value displayed on the meter and then press **Enter**.

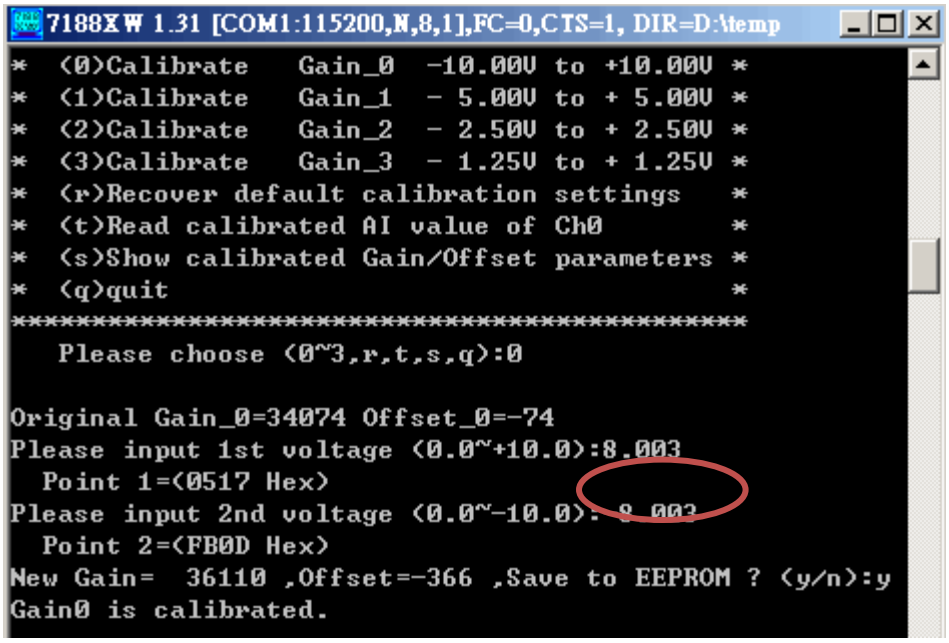



```
7188XW 1.31 [COM1:115200,N,8,1]    0,CTS=1, DIR=D:\temp
*  (0)Calibrate    Gain_0  -1    V to +10.00V *
*  (1)Calibrate    Gain_1  -     V to + 5.00V *
*  (2)Calibrate    Gain_2  - 2.50V to + 2.50V *
*  (3)Calibrate    Gain_3  - 1.25V to + 1.25V *
*  (r)Recover default calibration settings    *
*  (t)Read calibrated AI value of Ch0         *
*  (s)Show calibrated Gain/Offset parameters  *
*  (q)quit                                    *
*********************************************
   Please choose (0~3,r,t,s,q):0

Original Gain_0=34074 Offset 0=-74
Please input 1st voltage (0.0~+10.0):8.003
  Point 1=(0517 Hex)
Please input 2nd voltage (0.0~-10.0):-8.003
  Point 2=(FB0D Hex)
New Gain=  36110 ,Offset=-366 ,Save to EEPROM ? (y/n):y
Gain0 is calibrated.
```

e. Set the calibration source output to the second point (e.g., - 8 V in this example).

f. At the "**Input 2nd voltage**" prompt on the console, type the value displayed on the meter and then press **Enter**



The new Gain and Offset values for this calibration will then be displayed on the console as:

New Gain= 3xxxx, Offset= nnn, Save to EEPROM? (y/n):

g. Type **y** and press **Enter** to accept the values and save the settings to EEPROM.

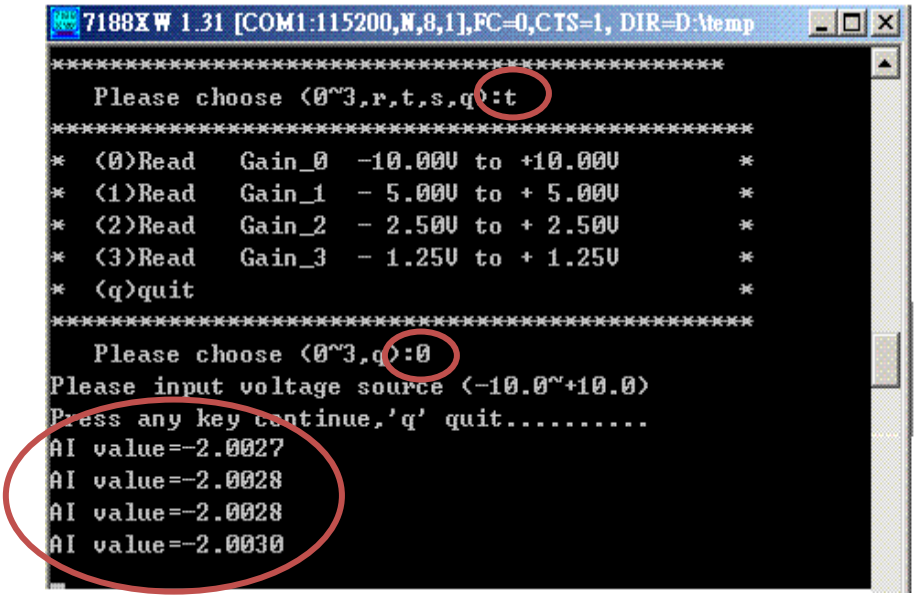
The calibration for the -10 V to +10 V input range is now complete.

Verifying the Calibration

Step 1. Set the calibration source to output a voltage to channel 0 on the module. For example, -2 V.

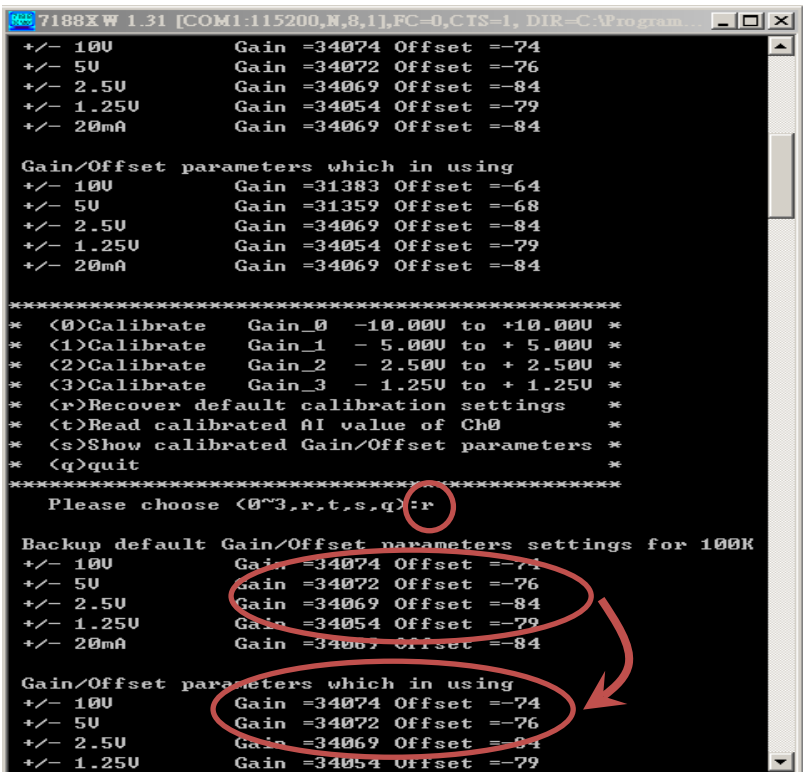
Step 2. In the same calibration program console window, type **t** (Read the calibrated AI value for channel 0), and then select the input type that was just calibrated (e.g., 0, -10 V to +10 V).

Step 3. Confirm that the values displayed for channel 0 are correct.



Restoring the Default Calibration Settings

When using the default input impedance of 200 kΩ, the calibration program provides a **Recover Default Calibration Settings** function **(r)** that can be used to restore the Gain and Offset values to the factory default settings.



5.2. Windows-based Controllers

Each module is factory calibrated and well verified before shipment, so it is usually unnecessary to calibrate the module again, unless the input impedance is changed on a calibrated module, or the accuracy is lost.

To calibrate it, in addition to inserting the module into a controller slot, the following items are required:

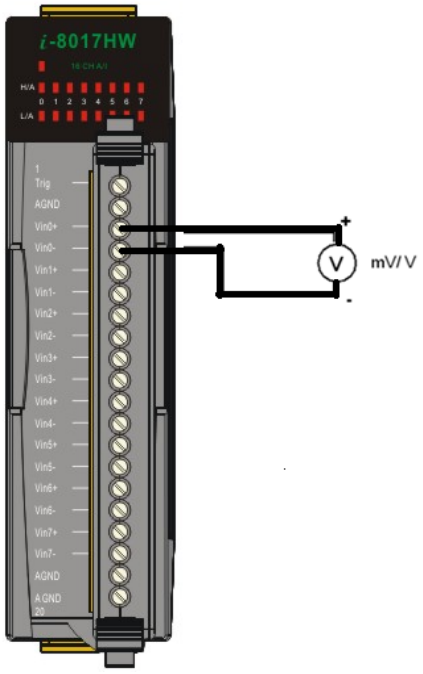
- A single stable calibration source, such as a 3 1/2 digit power supplier (or better), or a battery output.
- A single 4 1/2 digit voltage meter (15-bit resolution or better)
- A Calibration Program. See Location of the Demo Programs section for the contained in the demo programs folder.

Tips & Warnings

1. An unstable calibration source will cause calibration errors and affect the accuracy of the data acquisition.
2. If you wish to perform calibration using ±20 mA, select ±2.5 V instead as both types use the same gain and offset values.
3. The calibration program uses channel 0 to accept the calibration source only.

This section contains:

Calibrating on WinCE and WES PAC Units



Step 1. Refer to the Jumper Settings section. Ensure that the Differential/Single-ended input selection jumper is in the Differential position.

Step 2. Connect the calibration source to channel 0 of the module using the differential wiring method, as illustrated.

Step 3. Insert the module into a vacant slot on the controller and power on the controller.

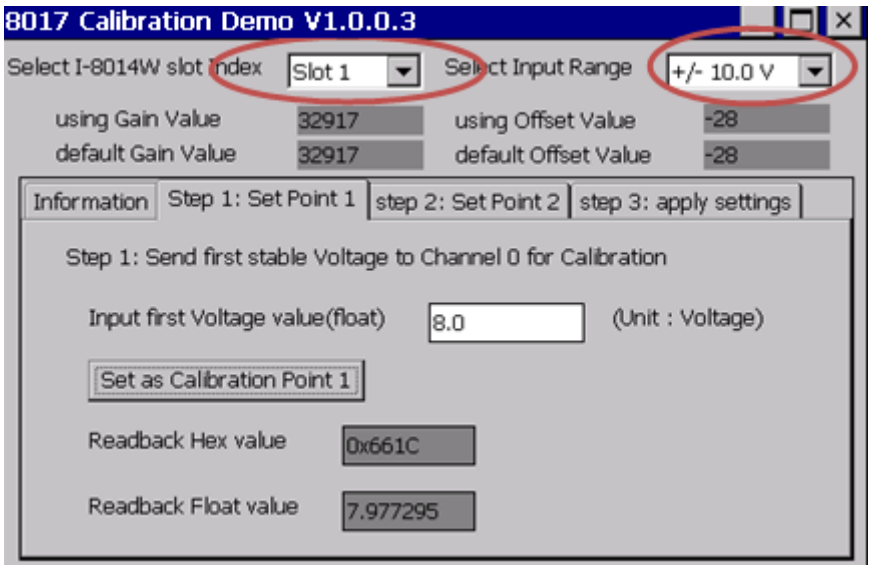
Step 4. Launch the pac_i8017W_Calibration.exe executable file on the controller to display the Calibration dialog box.

(See the Location of the Demo Programs section for details of where to find the c# demo programs for the module)

Tips & Warnings

Only channel 0 can be used to perform calibration.

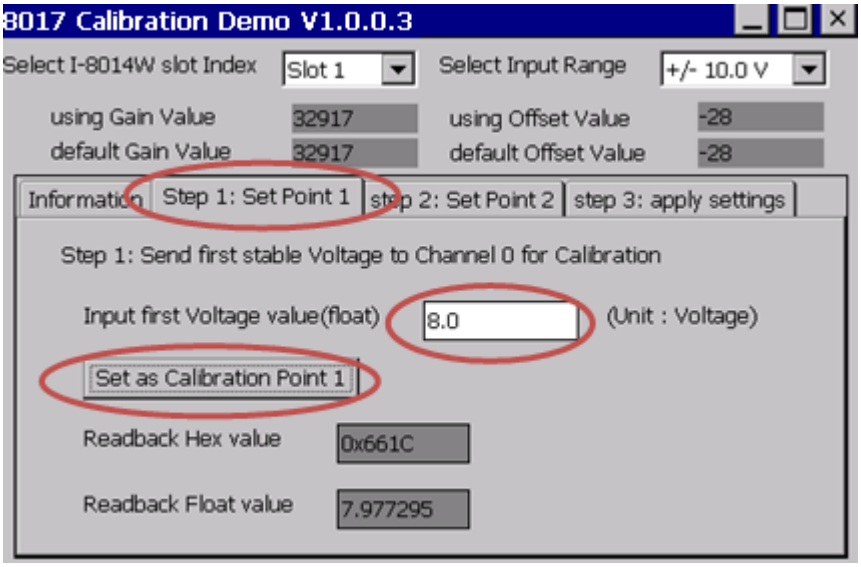
Step 5. In the upper section of the Calibration dialog box, select the I-8014W slot number and the input range from the respective drop-down lists.



Step 6. Determine two values (points) within the range of the input type selected for the calibration process. For example, after selecting -10 V to +10 V as the input range, +8 V and -8 V can be used as the two calibration points:

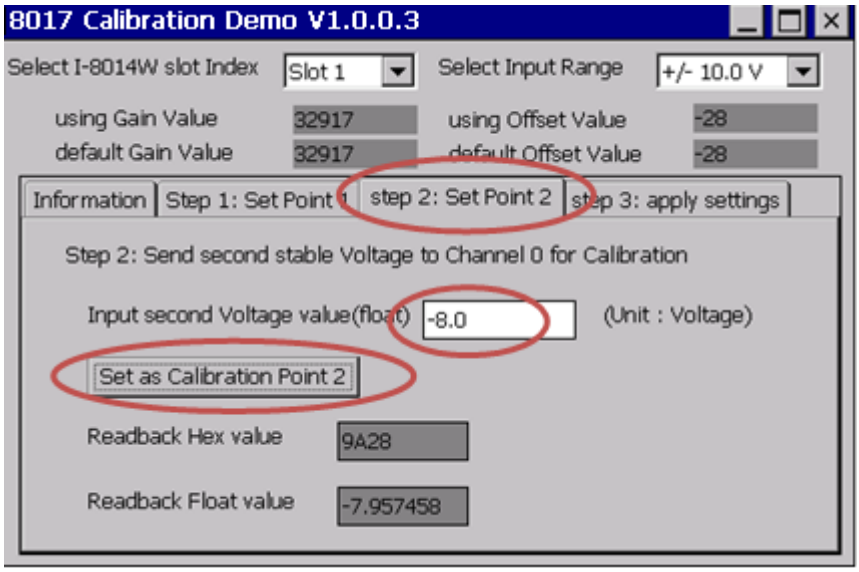
Step 7. Set the calibration source output to one of the two points (e.g., 8 V)

Step 8. Click the **Step 1: Set Point 1** tab and type the value displayed on the meter (e.g., 8.0) in the **Input First Voltage Value** text box, and then click the **Set as Calibration Point 1** button.

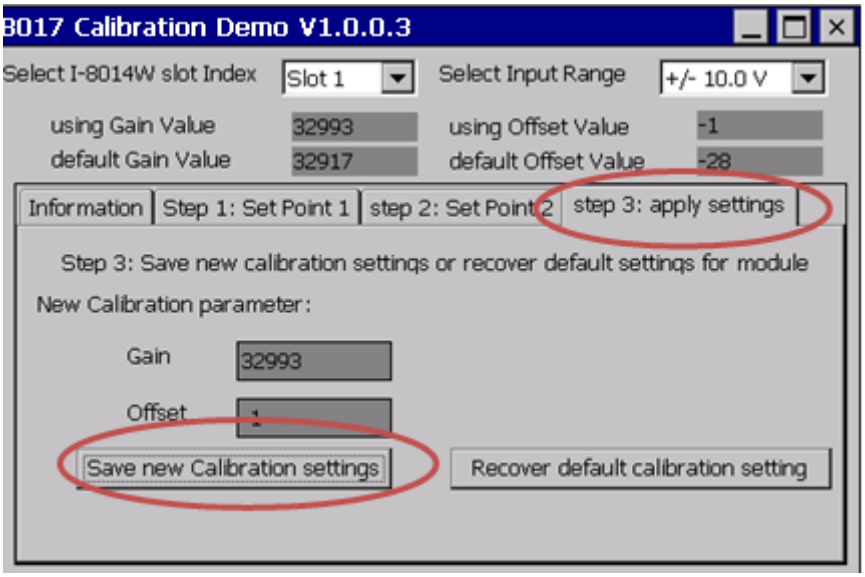


Step 9. Set the calibration source output to the second value (e.g., - 8 V in this example)

Step 10. Click the **Step 2: Set Point 2** tab and type the value displayed on the meter (e.g., - 8.0) in the **Input Second Voltage Value** text box, and then click the **Set as Calibration Point 2** button.



Step 11. Click the **Step 3: Apply Settings** tab, and check that the calibration parameters are correct. Click the **Save New Calibration Settings** button to save the calibration settings.

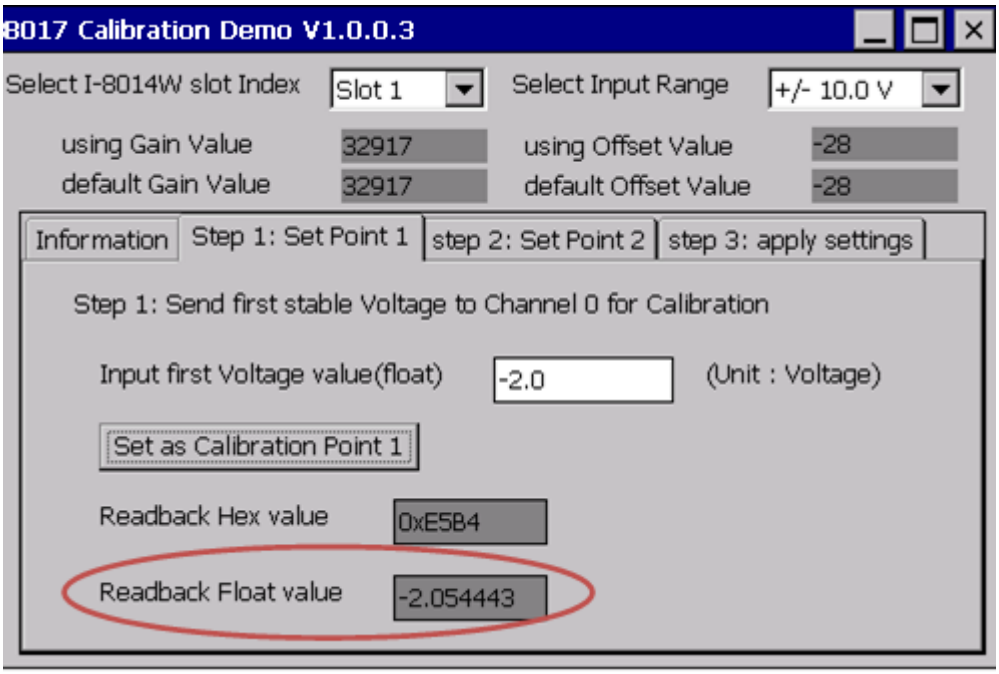


The calibration for the -10 V - +10 V input range is now complete.

Verifying the Calibration

Step 1. Set the calibration source to output a voltage to channel 0 on the I-8014W module. For example, -2 V.

Step 2. In the Calibration dialog box, click the **Step 1: Set Point 1** tab and confirm that the **AI Readback Float value** is as illustrated in the image below:

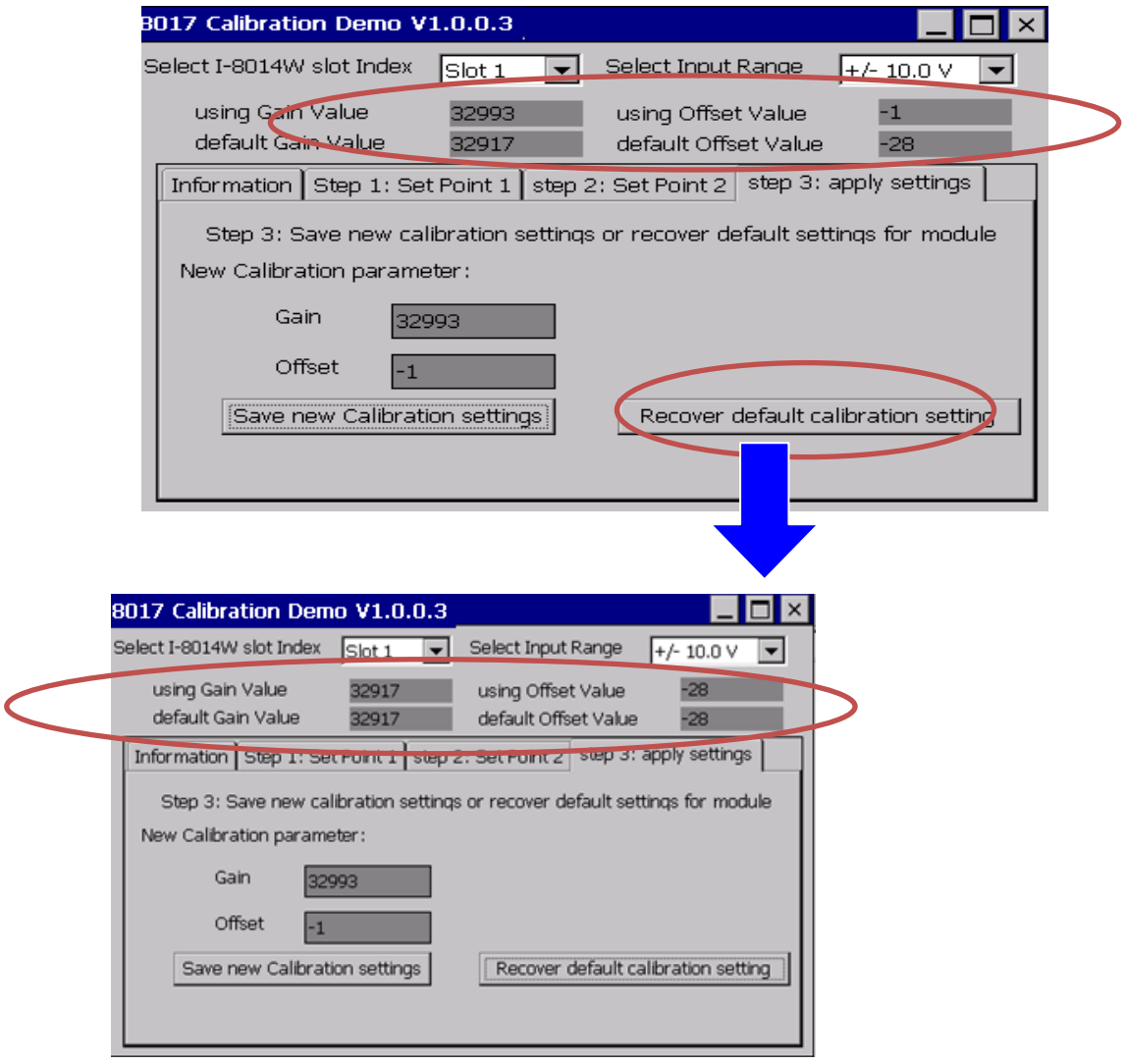


Restoring the Default Calibration Settings

When using the default input impedance of 200 kΩ, the calibration program includes a **Recover Default Calibration Settings** function that can be used to restore the Gain and Offset values to the factory default values:

Click the **Step3: Apply Settings** tab, and then click the **Recover Default Calibration Settings** button. The Gain and Offset settings will be restored to the factory default values and will be displayed in the upper section of the Calibration dialog box.

For an input impedance of 200 kΩ (the default setting), the calibration program provides **Recover default calibration settings** function to restore the Gain and Offset values to factory default:



6. Troubleshooting

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

This chapter contains:

6.1. 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.

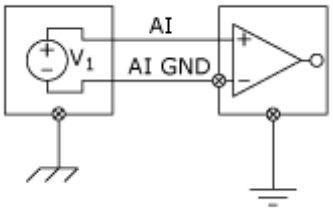
c. Insert the module into a slot in a Windows platform controller and then power on the controller.

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

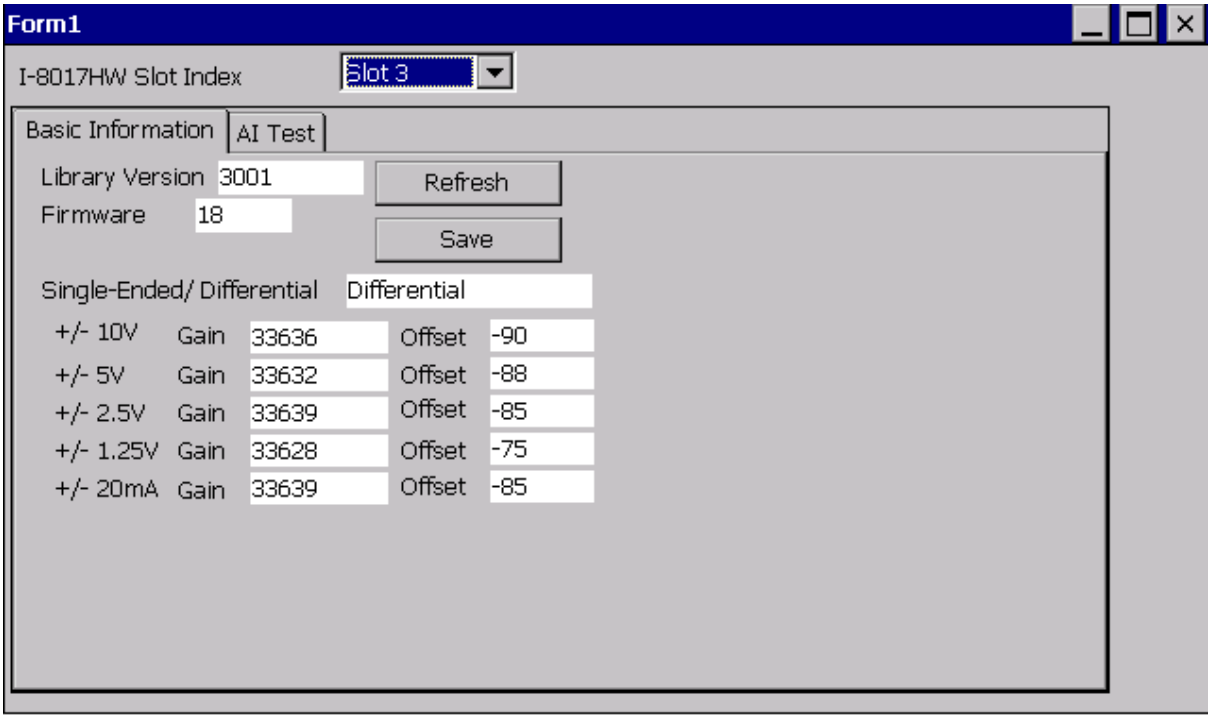
Step 3: Read the information from the module

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.

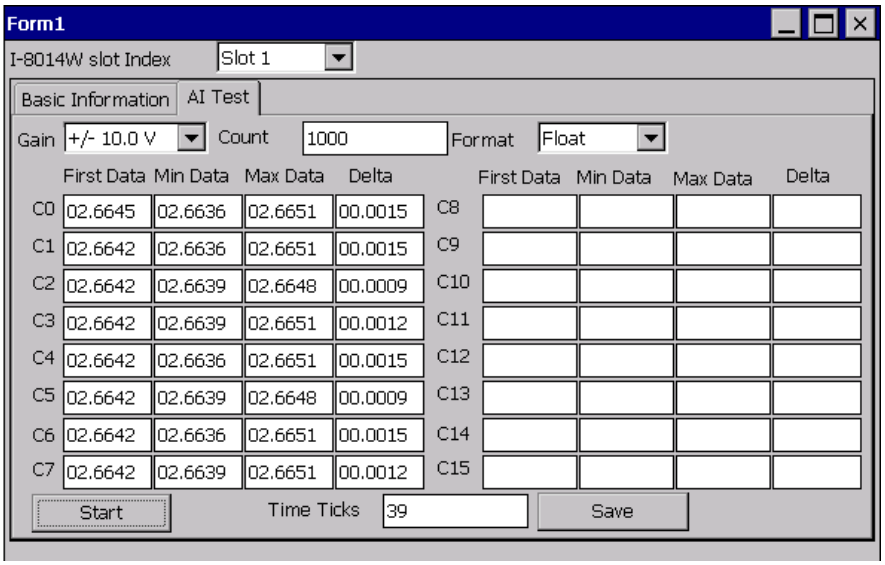
b. Relocate the module to a different slot, and then repeat Steps 2 and 3 to confirm whether or not the Gain values are correct.

Step 4. 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.

6.2. 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)

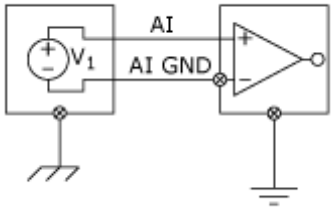
6.3. What to do when the data read from the module seems unstable

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 Definitions

Error code	Definition
0	No Errors
-1	ID_ERROR
-2	SLOT_ERROR
-3	CHANNEL_ERROR
-4	GAIN_ERROR
-5	INT_MODE_ERROR
-6	NOT_SUPPORT_ERROR
-7	INVALID_Calibration
-8	Bad_Calibration

Appendix B. Read AI Function Performance

Using a single channel:

Platform	ReadAI (Polling)	ReadAIHex (Polling)
WES	90~95 KHz	90~95 KHz
CE6	90~95 KHz	90~95 KHz
CE5	90 KHz	90~95 KHz
IP-8000	7.6KHz	36KHz
I-8000	2KHz	12KHz

Using multiple channels:

Platform	ReadAI (Polling)	ReadAIHex (Polling)
WES	35 KHz	35 KHz
CE6	35 KHz	35 KHz
CE5	35 KHz	35 KHz
IP-8000	6.6KHz	22KHz
I-8000	2KHz	9KHz

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 C. Revision Information

V2.0.0

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

V1.0.0

First Release for the I-8017HW module only