I-8015W/I-8015W-12

I/O Module User Manual

V1.0.1 September 2025

Written by Edward Ku/Cindy Huang

Edited by Anna Huang

Warranty

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

Warning

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

Copyright

Trademarks

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

Contact Us

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

Table of Contents

Preface

The **I-8015W/I-8015W-12 User Manual** provides detailed information required to install, configure, and operate the I-8015W series RTD input modules. This document is intended for system integrators, developers, and technical engineers who need to implement data acquisition or industrial automation systems using ICP DAS products.

This manual is organized into several chapters for ease of reference:

- **Chapter 1 – Product Overview**
  Introduces the module, including its specifications, pin assignments, wiring methods, RTD type settings, block diagram, and mechanical dimensions.
- **Chapter 2 – Installation & Getting Started**
  Provides step-by-step guidance on hardware installation, software setup, and initial verification on various platforms such as WinCE, Windows, and Linux.
- **Chapter 3 – API Reference**
  Describes the API functions available for program development, including syntax, parameters, return values, and sample code.
- **Chapter 4 – Modbus Protocol Reference**
  Details the Modbus RTU communication protocol supported by the I-8015W, including address mapping and RTD type codes.
- **Chapter 5 – Demo Programs**
  Offers ready-to-use examples that demonstrate how to acquire temperature values, set offset values, and read firmware versions.
- **Chapter 6 – Troubleshooting**
  Lists common issues and solutions, as well as guidelines for verifying communication and system configuration.
- **Appendices**
  Provide additional technical information such as performance benchmarks, error code definitions, and revision history.

By following this manual, users will be able to quickly integrate the I-8015W series modules into their systems and maximize the reliability of temperature measurement applications.

1. Product Overview

This chapter introduces the I-8015W/I-8015W-12 RTD input modules, covering product features, specifications, wiring, RTD type settings, and module dimensions. It provides essential hardware information for system design, installation, and integration.

1.1. Introduction

This section outlines the purpose and functionality of the I-8015W/I-8015W-12 I/O module, highlighting their role in industrial automation and data acquisition systems.

The I-8015W/I-8015W-12 are 8/12-channel RTD input modules designed for precise temperature measurement. They support multiple RTD sensor types (Pt100, Ni120, Cu50, Cu100) and feature:

- High-speed sampling (85 Hz per channel)
- Automatic compensation for 3-wire RTD connections
- 3000 VDC isolation protection
- ±4 kV ESD protection
- RoHS compliance

Typical applications include:

- Building and factory automation
- Machine monitoring
- Remote diagnosis and maintenance
- Testing equipment

Applications

- Building Automation
- Machine Automation
- Remote Diagnosis
- Factory Automation
- Remote Maintenance
- Testing Equipment

1.2. Specifications

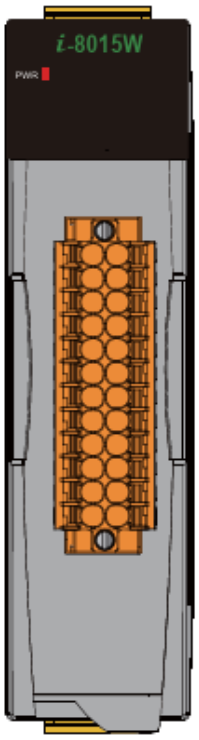
This section lists the system specifications, I/O specifications, and environmental conditions of the I-8015W/I-8015W-12.

Model	I-8015W	I-8015W-12
COM Port		
Ports	RS-232	
Data Format	N, 8, 1	
Baud Rate	921600 bps	
Protocol	Modbus RTU	
CPU Module		
Dual Watchdog Timer	Module (1.6 Seconds), Communication (Programmable)	
LED Indicators		
System LED Indicator	1 LED as Power Indicator	
I/O LED Indicator	12	24
Isolation		
Intra-module Isolation, Field-to-Logic	3000 VDC	
EMS Protection		
ESD (IEC 61000-4-2)	±4 kV Contact for Each port, ±8 kV Air for Random Point	
ESD (IEC 61000-4-2)	±4 kV Contact for Each port, ±8 kV Air for Random Point	
Power		
Consumption	0.39 W Max.	0.4 W Max.
Mechanical		
Dimensions (W x L × H)	31 mm × 114 mm × 126 mm	

Model		I-8015W	I-8015W-12
Analog Input			
Channels		8	12
Sensor Type		Pt100, Ni120, Cu50, Cu100	
Resolution		16-bit	
Accuracy Rate	Fast Mode	±0.1 % of FSR (25 °C)	
	Normal Mode	±0.05 % of FSR (25 °C)	
Sampling Rate	Fast Mode	85 Hz (per channel)	
	Normal Mode	1.5 Hz (per channel)	
Input Impedance		> 1 MΩ	
Individual Channel Configuration		Yes	
3-wire RTD Lead Resistance		Yes	
Elimination		400 Ω	
Resistance Measurement		Yes	
Environment			
Operating Temperature Storage		-25 ~ +75 °C	
Temperature		-40 ~ +85 °C	
Humidity		10 ~ 95 % RH , Non-condensing	

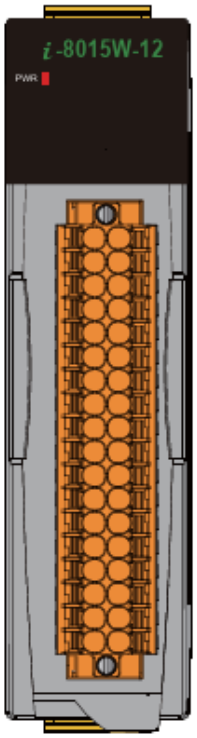
1.3. Pin Assignments

This section illustrates the pin configuration of the modules. Understanding pin assignments ensures proper wiring and avoids hardware damage.



I-8015W

Pin Assignment	Terminal No.		Pin Assignment
A1	01	13	A0
B1	02	14	B0
/B1	03	15	/B0
A3	04	16	A2
B3	05	17	B2
/B3	06	18	/B2
A5	07	19	A4
B5	08	20	B4
/B5	09	21	/B4
A7	10	22	A6
B7	11	23	B6
/B7	12	24	/B6



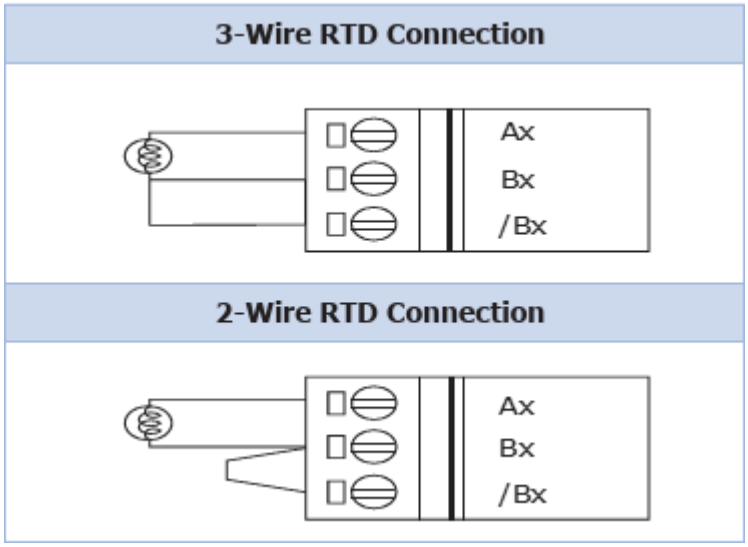
I-8015W-12

Pin Assignment	Terminal No.		Pin Assignment
A1	01	19	A0
B1	02	20	B0
/B1	03	21	/B0
A3	04	22	A2
B3	05	23	B2
/B3	06	24	/B2
A5	07	25	A4
B5	08	26	B4
/B5	09	27	/B4
A7	10	28	A6
B7	11	29	B6
/B7	12	30	/B6
A9	13	31	A8
B9	14	32	B8
/B9	15	33	/B8
A11	16	34	A10
B11	17	35	B10
/B11	18	36	/B10

1.4. Wire Connections

This section explains the wiring methods for connecting RTD sensors to the module. Proper wiring ensures accurate temperature measurement and minimizes noise interference.

- **2-wire RTD**: Simple but less accurate due to lead resistance.
- **3-wire RTD**: Provides automatic lead resistance compensation.



1.5. RTD Type Setting (TT)

This section describes how to configure the RTD type code for each channel. Correct RTD type selection ensures accurate measurement.

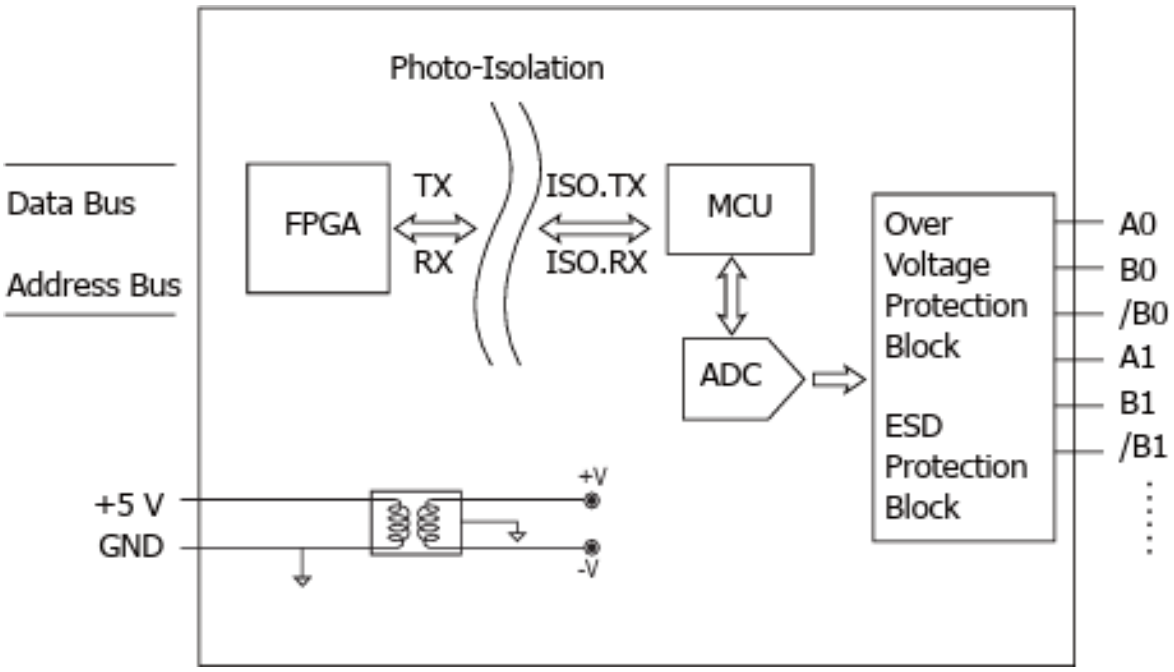
Type Code	RTD Type	Temperature
0 x 20	Pt 100, α = 0.00385	-100 ~ + 100 °C
0 x 21	Pt 100, α = 0.00385	0 ~ + 100 °C
0 x 22	Pt 100, α = 0.00385	0 ~ + 200 °C
0 x 23	Pt 100, α = 0.00385	0 ~ + 600 °C
0 x 24	Pt 100, α = 0.003916	-100 ~ + 100 °C
0 x 25	Pt 100, α = 0.003916	0 ~ + 100 °C
0 x 26	Pt 100, α = 0.003916	0 ~ + 200 °C
0 x 27	Pt 100, α = 0.003916	0 ~ + 600 °C
0 x 28	Ni 120	-80 ~ + 100 °C
0 x 29	Ni 120	0 ~ + 100 °C
0 x 2B	Cu 100, α = 0.00421	-20 ~ + 150 °C
0 x 2C	Cu 100, α = 0.00427	0 ~ + 200 °C
0 x 2E	Pt 100, α = 0.00385	-200 ~ + 200 °C
0 x 2F	Pt 100, α = 0.003916	-200 ~ + 200 °C
0 x 80	Pt 100, α = 0.00385	-200 ~ + 600 °C
0 x 81	Pt 100, α = 0.003916	-200 ~ + 600 °C
0 x 82	Cu 50	-50 ~ + 150 °C
0 x 83	Ni 100	-60 ~ 180 °C
0 x 84	Ni 120	-80 ~ 150 °C
0 x 85	Cu 100, α = 0.00428	0 ~ 150 °C
0 x 86	Pt 100, α = 0.00385	-100 ~ 300 °C
0 x 87	Pt 100, α = 0.003916	-100 ~ 300 °C

Tips & Warnings

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

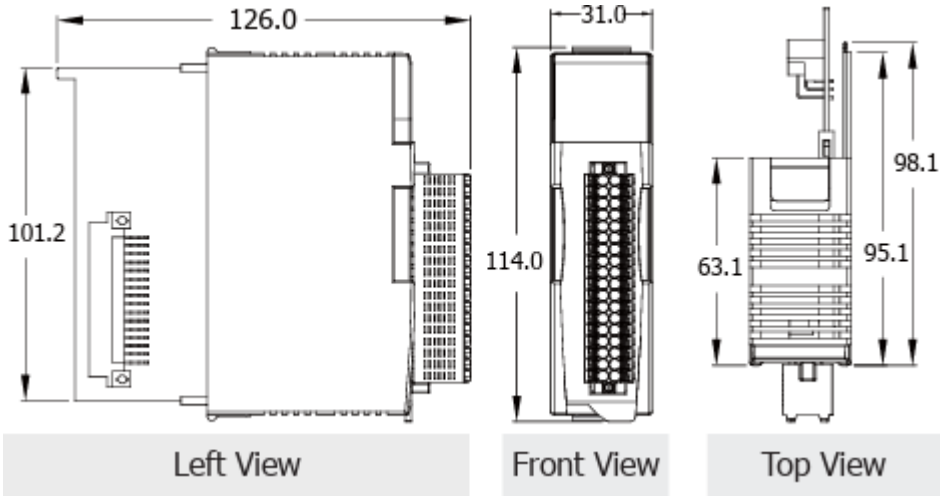
1.6. Block Diagram

This section provides the internal block diagram of the module, illustrating the signal flow and isolation protection design.



1.7. Dimensions

This section shows the physical dimensions of the module, which are essential for installation and system layout planning.



1.8. Supported PACs/ViewPACs

This section lists the PAC (Programmable Automation Controllers) series that support the I-8015W/I-8015W-12 I/O modules.

Platform	PAC/ViewPAC series
Windows-Based Controllers	XP-8000-WES7
	WES iPPC with I/O Slot(s)
WinCE-Based Controllers	WP-8000
	XP-8000-CE6
	WinCE ViewPAC with I/O Slot(s)
Linux-Based Controllers	LP-8000
	LX-8000
MiniOS7-Based Controllers	iP-8000
	I-8000
	MiniOS7 ViewPAC with I/O Slot(s)

2. Getting Started

This chapter provides the steps required to install and set up the I-8015W/I-8015W-12 I/O module, including hardware installation, software configuration, and initial testing. It also introduces software utilities available on different platforms (WinCE, Windows, Linux). By following the procedures in this chapter, users can complete basic setup and verify that the modules function correctly.

Package Checklist

After receiving the I-8015W/I-8015W-12 I/O module, please verify that all components are included and undamaged. If any item is missing or defective, contact your supplier or distributor immediately for assistance.

I-8015W / I-8015W-12

Pin Terminal

I-8015W：32 PCS

I-8015W-12：36 PCS

Quick Start Guide

2.1. Hardware Installation

This section explains how to install the I-8015W/I-8015W-12 I/O module into PACs/ViewPACs. Users should follow the recommended procedures in order to avoid damage to the module or the host controller.

Step 1: Power off the PAC/ViewPAC before installation.

Step 2: Insert the I/O module into the corresponding I/O slot of the PAC/ViewPAC.

Step 3: Secure the module firmly to ensure proper electrical contact.

Step 4: Reconnect power and verify that the PAC/ViewPAC detects the module after booting.

2.2. Software Installation and Configuration

This section introduces the software utilities and configuration procedures required for using the I-8015W/I-8015W-12 I/O modules on different platforms. Depending on the target operating system (WinCE, Windows, Linux), different utilities and SDKs are available for system configuration, module management, and data acquisition testing.

2.2.1. WinCE-based Controllers

The I-8015W/I-8015W-12 I/O modules support multiple utilities on WinCE platforms for configuration and management. These utilities include PAC Utility, PACSDK.dll, and DCON Utility, which must meet the minimum version requirements listed below.

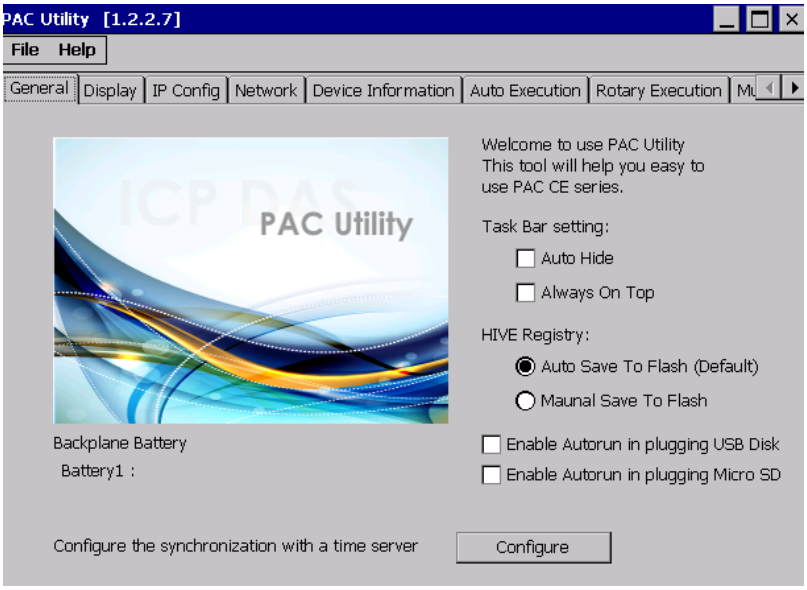
Models	OS	PAC_utility	PACSDK.dll	DCON_utility.exe
WP-8x2x-CE7	OS V1124 and later	V1.2.2.7 and latter	V4.5.0.4 and later	V4.3.0.3 and latter
WP-9x2x-CE7	OS V1125 and later			
VP-x23x-CE7	OS V1024 and later			

Tips & Warnings

- If the utility version is older than the listed requirement, update to the latest version before configuration.
- For detailed instructions, refer to the corresponding PAC user manuals.

2.2.1.1. PAC Utility

PAC Utility is a bundled software application for WinCE-based PACs, enabling system management and configuration. It provides functions such as COM port assignment and basic module status checks.



For detailed instructions on PAC Utility, please refer to the corresponding PAC user manuals:

- **WP-8000 Series**

https://www.icpdas.com/en/download/show.php?num=462

- **XP-8000-CE6 Series**

https://www.icpdas.com/en/download/show.php?num=463

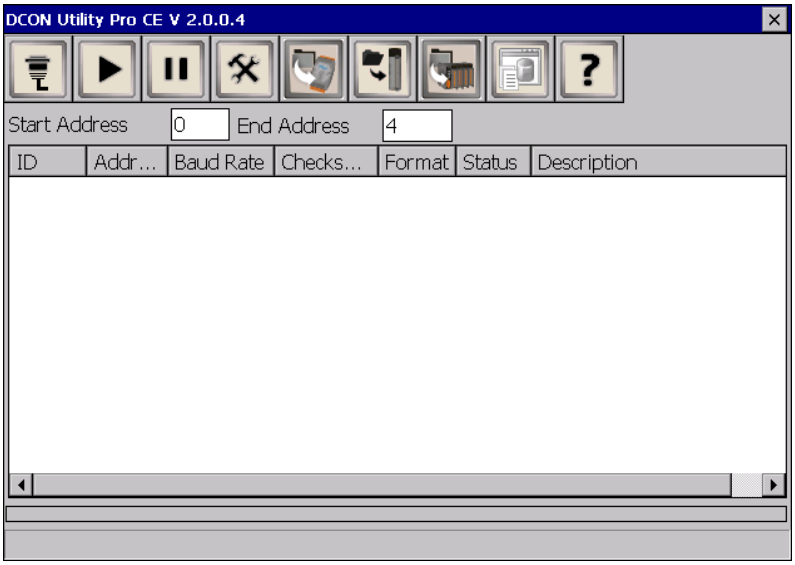
- **WinCE ViewPAC Series**

https://www.icpdas.com/en/download/show.php?num=758

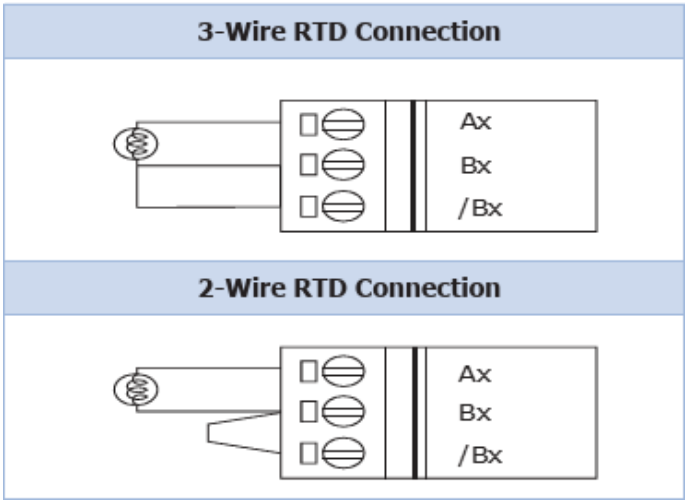
2.2.1.2. DCON Utility

DCON Utility Pro allows users to configure and test I/O modules via RS-232/RS-485 or Ethernet interfaces. It is widely used for verifying module data and checking AI readings.

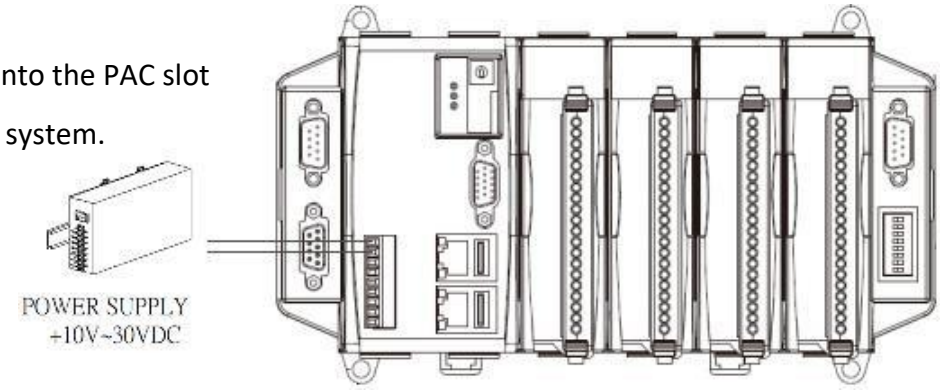
For more details, please refer to the official ICP DAS documentation:
https://www.icpdas.com/en/product/guide+Software+Utility_Driver+DCON_Utility_Pro



Step 1. Connect a stable signal source (e.g., battery)



Step 2. Insert the module into the PAC slot and power on the system.



Insert the I-8015W into the PAC slot and power on the controller. After rebooting, the OS will automatically mount the module to a COMx port.

- Default COM port numbers are mapped according to slot positions.

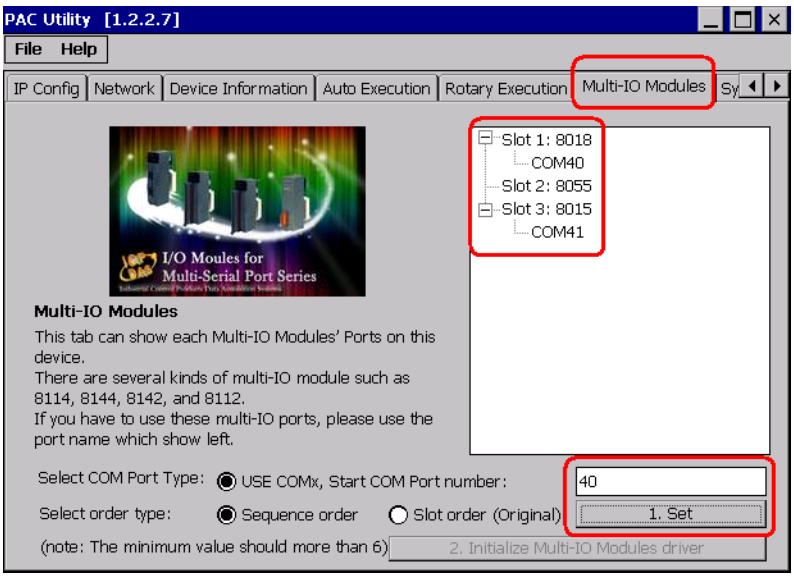
Slot No.	Port No.
Slot0	COM30:
Slot1	COM31:
.	.
.	.
.	.
Slot7	COM37:

- If the default COM port number is not suitable, it can be changed using **PAC Utility**.

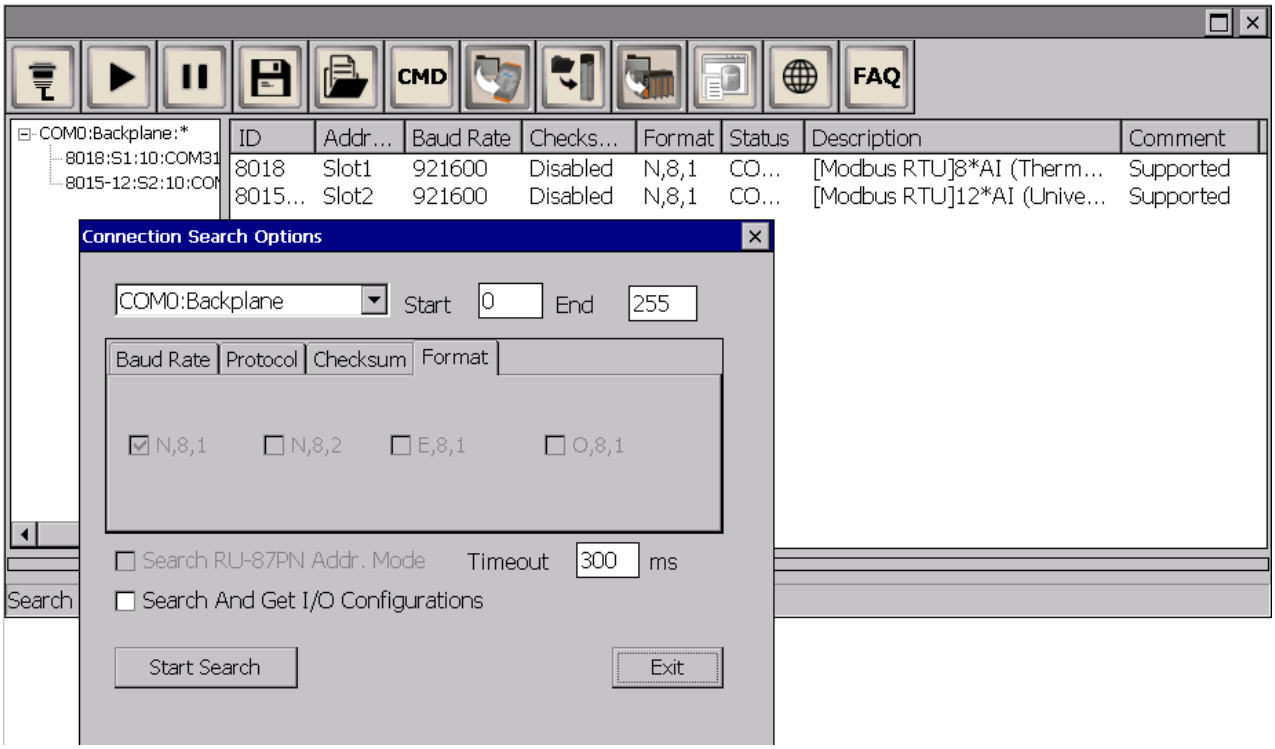
Example :

I-8018W → COM40;

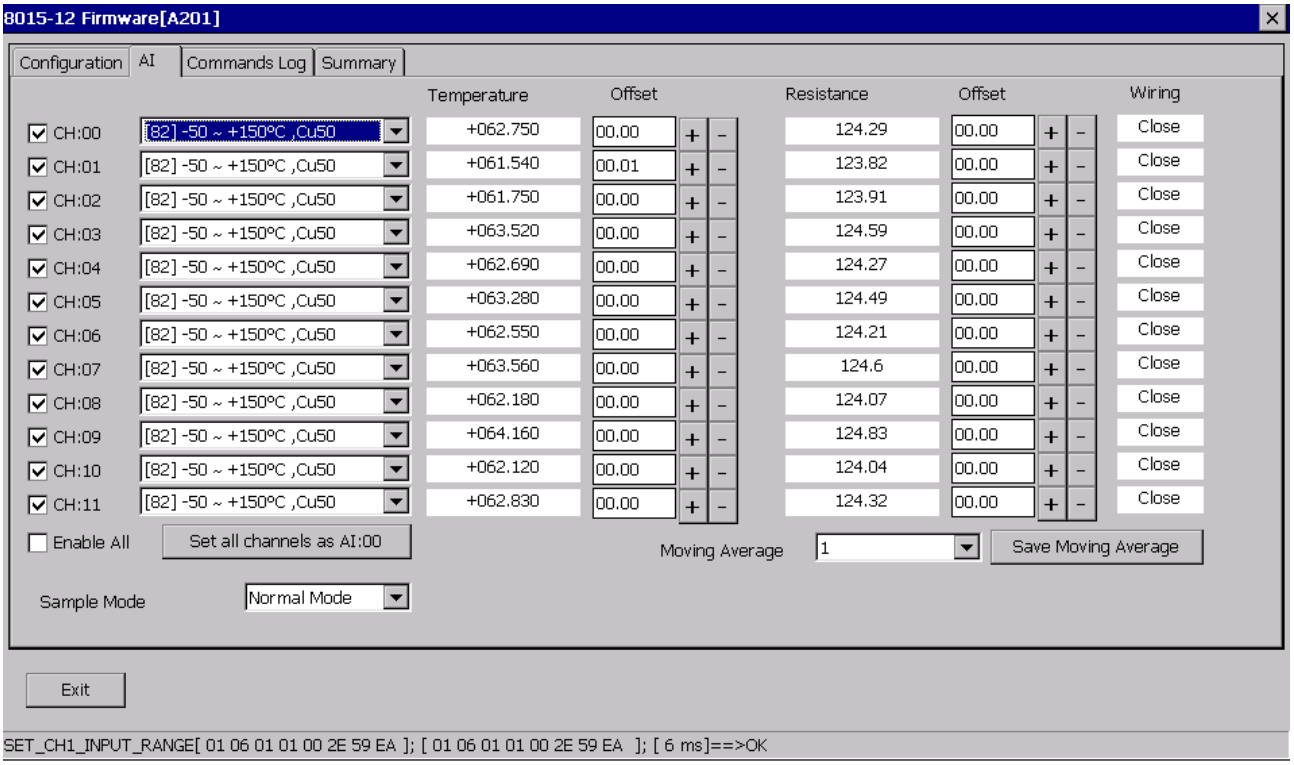
I-8015W → COM41



Step 3. Run **DCON Utility Pro**, check COM port mapping, and verify AI readings.



If the setup is correct, AI channel data should be displayed accurately in the utility window.



2.2.2. Windows-based Controllers

On Windows-based PAC controllers, the I-8015W/I-8015W-12 I/O modules can be configured using both PAC Utility and DCON Utility Pro. These tools help assign COM ports, configure parameters, and monitor AI channel values.

2.2.2.1. PAC Utility

PAC Utility is a bundled software application for Windows-based PACs, enabling system management and configuration. It provides functions such as COM port assignment and basic module status checks.

For detailed instructions on PAC Utility, please refer to the corresponding PAC user manuals:

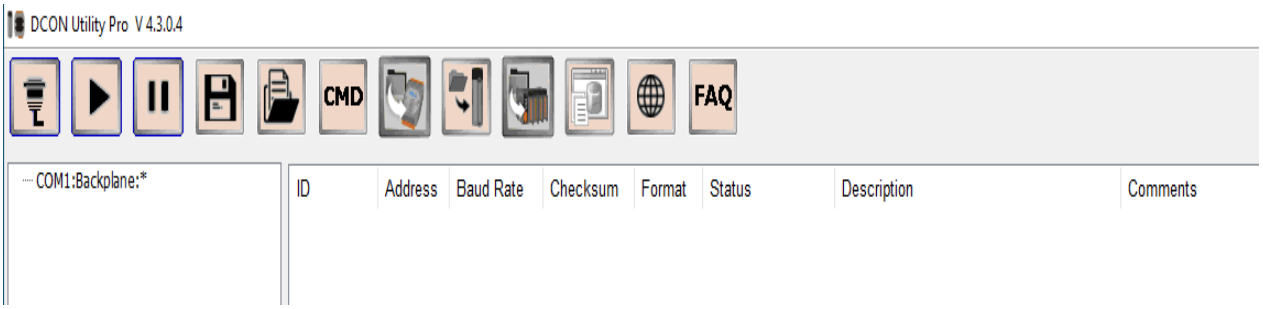
- **XP-8000-WES7**

https://www.icpdas.com/en/download/show.php?num=464

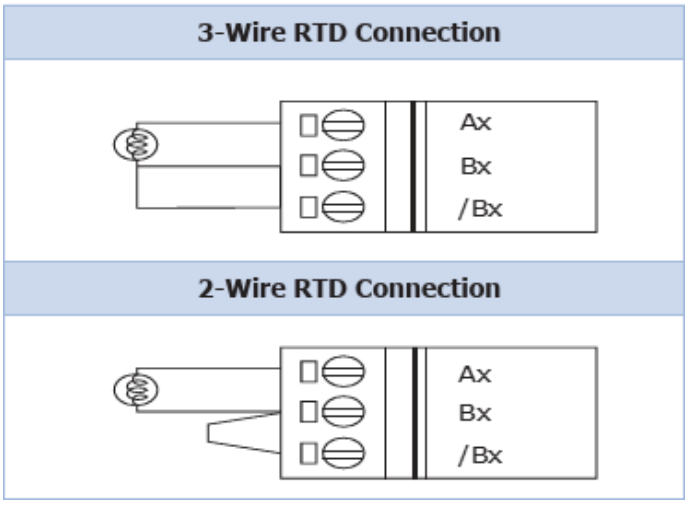
2.2.2.2. DCON Utility Pro

DCON Utility Pro allows users to configure and test I/O modules via RS-232/RS-485 or Ethernet interfaces. It is widely used for verifying module data and checking AI readings.

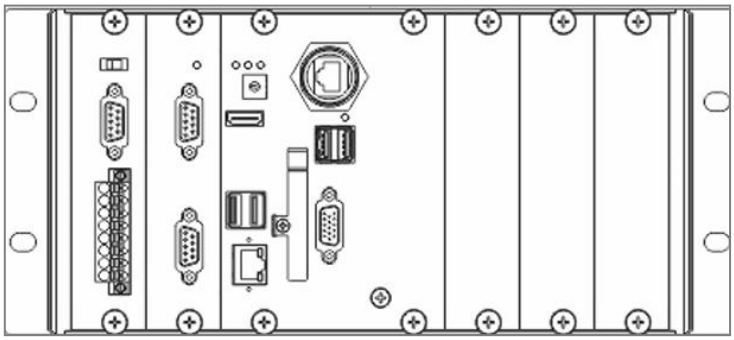
For more details, please refer to the official ICP DAS documentation:
https://www.icpdas.com/en/product/guide+Software+Utility_Driver+DCON_Utility_Pro



Step 1. Connect a stable signal source (e.g., battery)



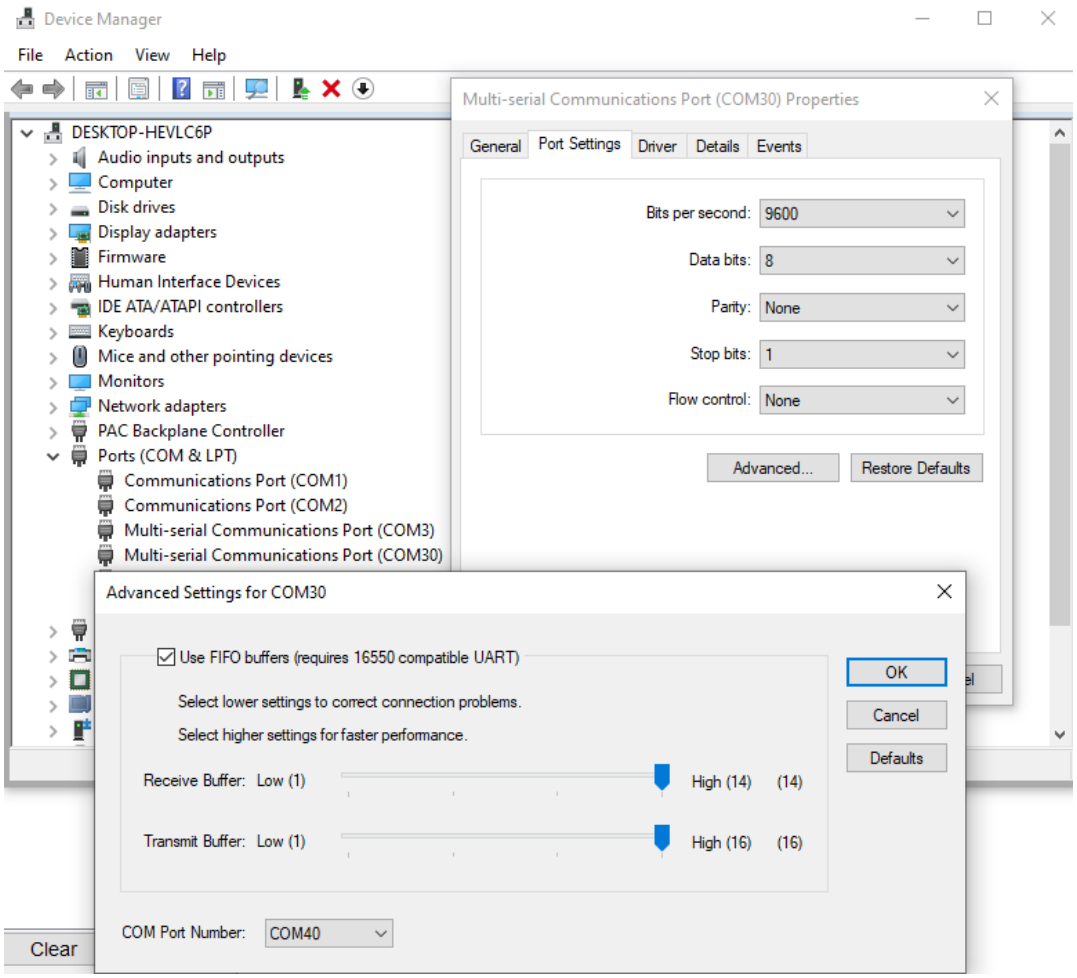
Step 2. Insert the module into the PAC slot and power on the system.



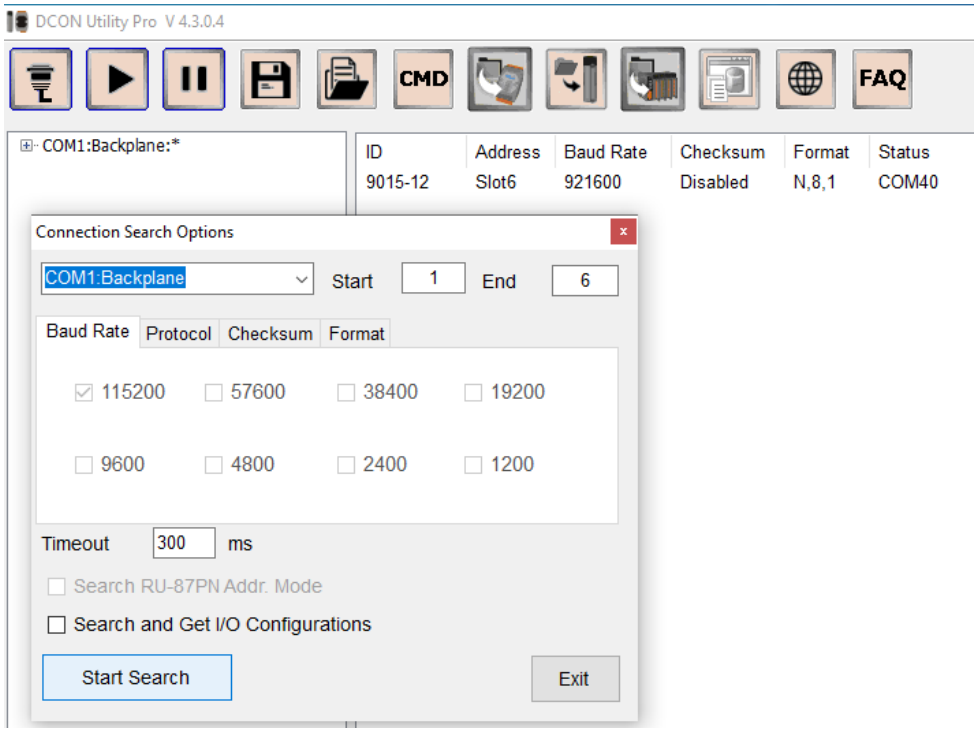
Insert the I-8015W into the PAC slot and power on the controller. After rebooting, the OS will automatically mount the module to a COMx port.

- Default COM port numbers are mapped according to slot positions.
- If the default COM port number is not suitable, it can be changed using **Windows Device Manager**.

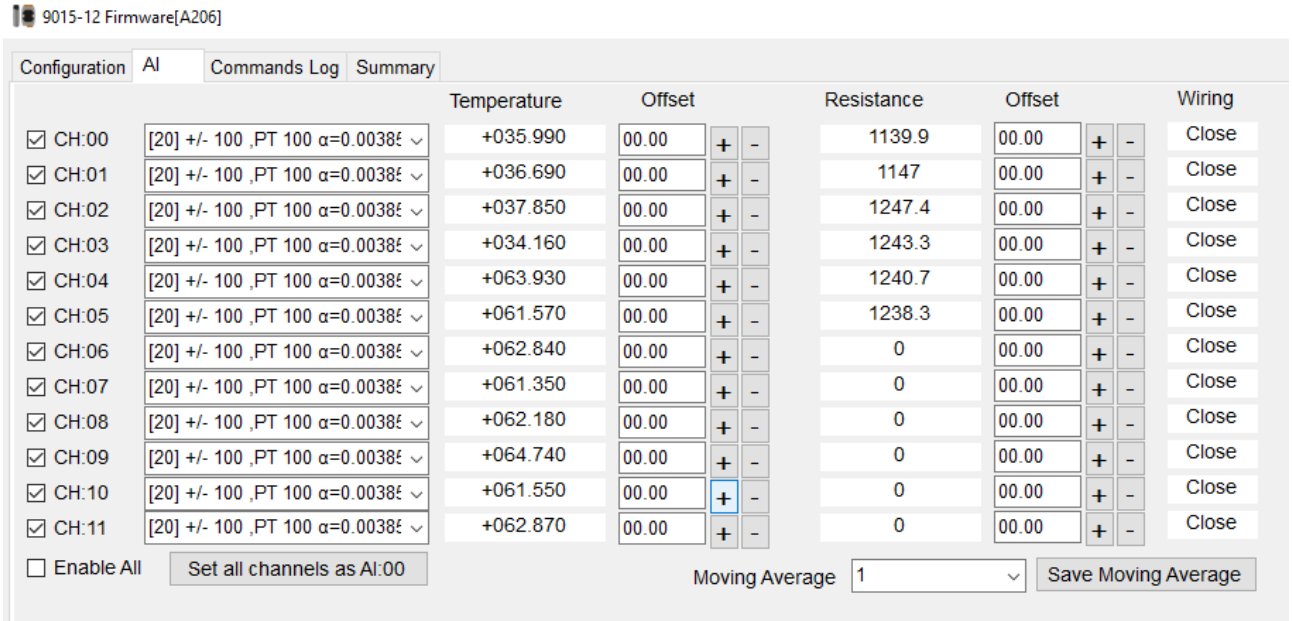
Slot No.	Port No.
Slot0	COM30:
Slot1	COM31:
.	.
.	.
.	.
Slot7	COM37:



Step 3. Run **DCON Utility Pro**, check COM port mapping, and verify AI readings.



If the setup is correct, AI channel data should be displayed accurately in the utility window.



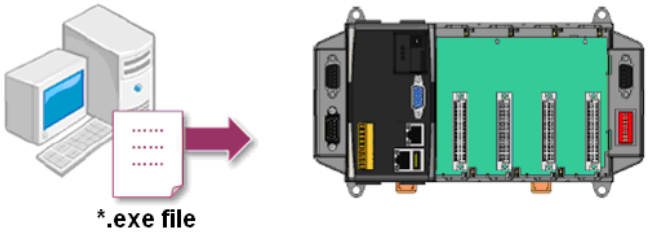
2.2.3. Linux-based Controllers

This section explains how to set up and test the I-8015W/I-8015W-12 I/O modules on Linux-based PACs. Users need to install the LinPAC SDK and compile demo programs.

Basic Functions

The LinPAC SDK provides diagnostic functions that allow users to verify whether the module is correctly configured and operating.

- 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



Step 1. Download and Install the LinPACSDK

Step 2. Verify the Hardware Connections

Check power cable, Ethernet cable, VGA monitor, and communication cables between controller and PC, and verify that the I-9012 has been properly plugged into the controller.

Step 3. Transfer demo programs to the controller and execute them to verify AI readings.

Step 4. Reference SDK and Download Links

ICP DAS provides SDKs and demo programs for different PAC product lines. The following table lists supported Linux-based PACs and their SDK download links.

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 and Demo References

This chapter introduces both the Application Programming Interfaces (APIs) and demo programs provided for the I-8015W/I-8015W-12 I/O modules. Developers can use the APIs to configure and control the modules programmatically, while the demo programs provide practical examples to verify module functions.

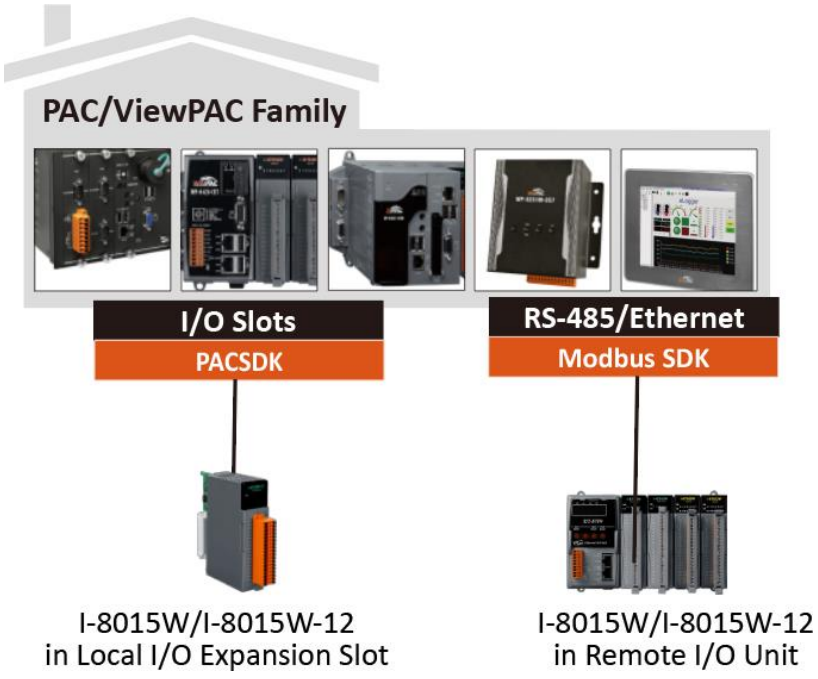
The APIs in this manual are primarily designed for **MiniOS7-based, WinCE-based** and **Windows-based** PAC development environments.

SDK Selection

Before developing applications for the I-8015W/I-8015W-12 I/O modules, developers must install the correct SDK according to the system architecture and hardware connection method. The SDK provides the necessary headers, libraries, and utilities for compilation and integration.

Please download the PAC/ViewPAC SDK from ICP DAS official site, according to your PAC/ViewPAC: https://www.icpdas.com/en/download/show.php?num=3143

- **PACSDK** → Used when the I-8015W/I-8015W-12 is installed in a Local I/O Expansion Slot of PAC/ViewPAC.
- **Modbus SDK** → Used when the I-8015W/I-8015W-12 is connected as a Remote I/O Unit through RS-485 or Ethernet.



3.1. Modbus References

This section provides details about the Modbus RTU protocol supported by the I-8015W/I-8015W-12 I/O modules, including communication parameters, supported function codes, address mapping, and RTD type code definitions.

Protocol Parameters

- Baud Rate: 921600 bps
- Data Bits: 8
- Parity: None
- Stop Bits: 1
- Function Supported: 0x04 (Read Input Registers)

Error Response

If the required function is not supported, the module will respond with the following information:

00	Address	1 byte	1 ~ 247
01	Function code	1 byte	Function code| 0x80
02	Exception code	1 byte	01

If an error occurs in the CRC, the module will not send a response

3.1.1. Modbus Address Mapping

The following table lists the Modbus register mapping of the I-8015W/I-8015W-12 I/O modules, showing how channel data, configuration values, and status information can be accessed.

Address	Description	Attribute
40001 -40008	Temperature of channel 0 to 7	**R**
40257 -40264	RTD Type code of channel 0 to 7	**R/W**
40289 -40296	Channel temperature offset of channel 0 to 7 in 0.01C	**R/W**
40385 -40392	Channel resistance offset of channel 0 to 7 in 0.01 ohms	**R/W**
40481	Firmware version (low word)	**R**
40482	Firmware version (high word)	**R**
40483	Module name (low word) 0x1500, 0x150C for I-8015W-12	**R**
40484	Module name (high word) 0x0080	**R**
40490	Disable/enable channels, bit 0 for channel 0, bit 1 for channel 1, etc. 0 to disable and 1 to enable Sampling time = 2ms + number of enabled channels * 0.85ms	**R/W**
40498	Number of moving averaging, 1 to 128, default 1	**R/W**
40501	Number of moving averaging without written to EEPROM, 1 to 128, default 1	**R/W**
40865 -40872	Resistance of channel 0 to 7 in 0.01 ohm for Pt100 range and 0.1 ohm for Pt1000 range	**R**
00129 -00136	Open wire status of channel 0 to 7, 1 for open wire	**R**

3.1.2. Modbus Function Code 0x04 (Read Input Channels)

Function code 0x04 is used to read contiguous analog input channels. This allows acquisition of temperature data from the RTD inputs.

Request

00	Address	1 Byte	1
01	Function code	1 Byte	0x04
02 ~ 03	Starting channel	2 Bytes	0 to 7 for reading temperature inputs
04 ~ 05	Number of input channels (N)	2 Bytes	1 to 8 for reading temperature inputs.
06 ~ 07	Check sum	2 Bytes	Check sum of data 00 to 05

Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0 x 04
02	Byte count	1 Byte	2 x N
03 ~	Number of input channels	2 x N Bytes	2 bytes for each channel
03 + 2N ~ 04 + 2N	Check sum	2 Bytes	check sum of data 00 to (02+2N)

Error Response

00	Address	1 Byte	1 to 247
01	Function code	1 Byte	0 x 84
02	Exception Code	1 Byte	02: starting channel out of range 03: (starting channel + number of input channels) out of range, incorrect number of bytes received
03 ~ 04	Check sum	2 x N Bytes	check sum of data 00 to 02

Note: for the two-byte data, the data are arranged as high byte first then the low byte.

3.1.3. RTD Type Code Table

This table defines the RTD type codes supported by the I-8015W/I-8015W-12. Each type code corresponds to a specific RTD sensor and measurement range.

Type Code	RTD Type	Min.	Max.
0x20	Pt 100, α = 0.00385, -100 ~ 100°C	-10000	10000
0x21	Pt 100, α = 0.00385, 0 ~ 100°C	0	10000
0x22	Pt 100, α = 0.00385, 0 ~ 200°C	0	20000
0x23	Pt 100, α = 0.00385, 0 ~ 600°C	0	60000
0x24	Pt 100, α = 0.003916, -100 ~ 100°C	-10000	10000
0x25	Pt 100, α = 0.003916, 0 ~ 100°C	0	10000
0x26	Pt 100, α = 0.003916, 0 ~ 200°C	0	20000
0x27	Pt 100, α = 0.003916, 0 ~ 600°C	0	60000
0x28	Ni 120, -80 ~ 100°C	-8000	10000
0x29	Ni 120, 0 ~ 100°C	0	10000
0x2A	Pt 1000, α = 0.00385, -200 ~ 600°C	-2000	6000
0x2B	Cu 100, α = 0.00421, -20 ~ 150°C	-2000	15000
0x2C	Cu 100, α = 0.00427, 0 ~ 200°C	0	20000
0x2D	Cu 1000, α = 0.00421, -20 ~ 150°C	-2000	15000
0x2E	Pt 100, α = 0.00385, -200 ~ 200°C	-20000	20000
0x2F	Pt 100, α = 0.003916, -200 ~ 200°C	-20000	20000

The under range value is –32768 and the over range value is +32767.

3.2. API References

This chapter introduces the Application Programming Interfaces (APIs) for the I-8015W/I-8015W-12 I/O modules. Developers can use these APIs to initialize modules, read temperature values, configure RTD type codes, and apply moving average filters.

Tips & Warnings

The main SDK for PAC development must be installed and deployed according to the instructions provided in each PAC's user manual. The APIs described in this chapter are specifically provided for the I-8015W/I-8015W-12 I/O modules.

API Naming Table

API functions use different naming prefixes depending on the platform. This ensures consistent usage within each environment while allowing cross-platform compatibility. The following table summarizes the prefixes required for different environments.

Platform	Required Header & Library	API Prefix
MiniOS7-Based PACs	i8015W.h、8015W.LIB	i8015W_*
WinCE-Based PACs	pac_i8015W.h、pac_i8015W.lib、pac_i8015W.dll、pac_i8015WNet.dll (for C# only)	pac_i8015W_*
Windows-Based PACs	Required Header & Library	pac_i8015W_*

The API prefix listed in the table (e.g., i8015W_*, pac_i8015W_*) indicates the common naming rule for each platform. The asterisk * represents the actual function name.

Example: Get Library Version

- MiniOS7 → i8015W_GetLibVersion();
- WinCE/WES→ pac_i8015W_GetLibVersion();

API Function Overview

The following table lists the main API functions supported by the I-8015W/I-8015W-12 I/O modules, along with a brief description of their purpose. Detailed information related to individual functions can be found in the following sections.

Function	Description
i8015W_GetLibVersion/ pac_i8015W_GetLibVersion	This function is used to read the version information for the currently installed library
i8015W_GetLibDate/ pac_i8015W_GetLibDate	This function is used to read the build date information for the currently installed library.
i8015W_GetFirmwareVersion/ pac_i8015W_GetFirmwareVer	This function is used to read the firmware version information for the module in the specified slot.
i8015W_Init/ pac_i8015W_Init	This function is used to initialize the i8015W. Note that this function MUST be used before using any other function in the library,
i8015W_ReadAIHex/ pac_i8015W_ReadAIHex	This function reads the 2's Temperature value of the i8015W module.
i8015W_ReadAIHexAll/ pac_i8015W_ReadAIHexAll	This function reads all the Temperature values of all channels in 2's complement-mode of the i8015W module.
i8015W_ReadAI/ pac_i8015W_ReadAI	This function reads the Temperature value of the i8015W module.
i8015W_ReadAIAll/ pac_i8015W_ReadAIAll	This function reads all the Temperature values of all channels in the i8015W module.
i8015W_SetTypeCode/ pac_i8015W_SetTypeCode	This function sets the Type code of Temperature in the i8015W module.
i8015W_GetTypeCode/ pac_i8015W_GetTypeCode	This function gets the Type code of Temperature in the i8015W module.
i8015W_SetMovingAverage/ pac_i8015W_SetMovingAverage	This function sets the Moving Average of Temperature value in the i8015W module.
i8015W_GetMovingAverage/ pac_i8015W_GetMovingAverage	This function gets the Moving Average of Temperature value in the i8015W module.

3.2.1. i8015W_GetLibVersion

This function is used to read the version information for the library installed on the I-8015w module.

Syntax

For MiniOS7

```
short i8015W_GetLibVersion();
```

For Windows (CE and WES)

```
Short pac_i8015W_GetLibVersion();
```

Parameter

This function does not require any parameters

Return Values

The version information for the Library file

Refer to Appendix A: “Error Code Definitions” for details of other returned values.

Example

[C++] MiniOS7

```
int version;
version = i8015W_GetLibVersion( );
```

[C++] Windows

```
int version;
version = pac_i8015W_GetLibVersion( );
```

[C#]

```
string version;
version = pac8015W.pac_i8015W_GetLibVersion( );
```

3.2.2. i8015W_GetLibDate

This function is used to read the build date for the Library currently installed on the I-8015w module.

Syntax

For MiniOS7

```
void i8015W_GetLibDate(char* LibDate);
```

For Windows (CE and WES)

```
void pac_i8015W_GetLibDate(char* LibDate);
```

Parameter

**LibDate: [out]*

The build date for the I-8015W library.

Return Values

This function does not return a value

Refer to Appendix A: “Error Code Definitions” for details of other returned values.

Example

[C/C++] MiniOS7

```
char libdate[32];
i8015W_GetLibDate(libdate);
```

[C/C++] Windows

```
char libdate[32];
pac_i8015W_GetLibDate(libdate);
```

[C#]

```
string libdate;
libdate = pac8015W.pac_i8015W_GetLibDate();
```

3.2.3. i8015W_GetFirmwareVersion

This function is used to read the firmware (LPGA) version information for the I-8015w module inserted in the specified slot.

Syntax

For MiniOS7

```
short i8015W_GetFirmwareVersion(int slot, short* firmwareVersion);
```

For Windows (CE and WES)

```
short pac_i8015W_GetFirmwareVer(int slot, short* firmwareVersion);
```

Parameter

slot: [in]

Specifies the slot where the I-8084W module is inserted. Valid range = 0 to 7.

**firmwareVersion: [out]*

The firmware version information for the I-8084W LPGA code.

Return Values

This function does not return a value

Refer to Appendix A: “Error Code Definitions” for details of other returned values.

Example

[C/C++] MiniOS7

```
int slot = 1;
short firmwareVersion;
i8015W_GetFirmwareVersion(slot, &firmwareVersion);
```

[C/C++] Windows

```
int slot = 1;
short firmwareVersion;
pac_i8015W_GetFirmwareVer(slot, &firmwareVersion);
```

[C#]

```
int slot = 1;
string firmwareVersion;
firmwareVersion = pac8015W.pac_i8015W_GetFirmwareVer(slot);
```

3.2.4. i8015W_Init

This function is used to initialize the the 8015W module.

Syntax

For MiniOS7

```
short i8015W_Init(int slot);
```

For Windows (CE and WES)

```
short pac_i8015W_Init(int slot);
```

Parameter

slot: [in]

Specifies the slot where the I-8015W module is inserted. Valid range = 0 to 7

Return Values

0: The module inserted in the slot is an I-8015W.

1: The module inserted in the slot is an I-8015W-12.

-1: There is no I-8015W module inserted in this slot.

Refer to Appendix A: “Error Code Definitions” for details of other returned values.

Note: Before executing any functions on the I-8015W module, the Init function needs to be called once for each I-8015W. If there are two or more I-8015W modules inserted in the controller, this function needs to be called individually for each I-8015W module by specifying the slot number where the I-8015W is inserted

Example

[C++] MiniOS7

```
int slot = 1;
i8015W_Init(slot);
```

[C++] Windows

```
int slot = 1;
pac_i8015W_Init(slot);
```

[C#]

```
int slot = 1;
pac8015W.pac_i8015W_Init (slot);
```

3.2.5. i8015W_ReadAIHex

This function is used to read temperature value on specific channel in Hexadecimal format.

Syntax

For MiniOS7

```
short i8015W_ReadAIHex(int slot,short ch,short* hval);
```

For Windows (CE and WES)

```
short pac_i8015W_ReadAIHex(int slot,short ch,short* hval);
```

Parameter

slot:

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

ch:

Specific channel (0 - 7).

**hval:*

[Output] HEX format data.

Return Values

Please refer to the Error Code.

Example

[C/C++] MiniOS7

```
int slot = 1;
short hval=0,ch=0;
i8015W_Init(slot);       // initialize
i8015W_ReadAIHex(slot,ch,&hval);
```

[C/C++] Windows

```
int slot=1;
short hval=0,ch=0;
pac_i8015W_Init(slot);       // initialize
pac_i8015W_ReadAIHex(slot,ch,&hval);
```

[C#]

```
int slot = 1, ch = 0;
short hval = 0;
pac8015WNet.pac8015W.pac_i8015W_Init(slot);     // initialize
pac8015WNet.pac8015W.pac_i8015W_ReadAIHex(slot, ch, ref hval);
```

3.2.6. i8015W_ReadAIHexAll

This function is used to read temperature value on all channels in Hexadecimal format.

Syntax

For MiniOS7

```
short i8015W_ReadAIHexAll(int slot,short hval[]);
```

For Windows (CE and WES)

```
short pac_i8015W_ReadAIHexAll (int slot, short hVal[]);
```

Parameter

slot:

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

** hVal[]:*

[Output] the hexadecimal data array.

Return Values

0 = No Error.

Example

[C/C++] MiniOS7

```
int slot = 1;
short hVal[16];
i8015W_Init(slot);       // initialize
i8015W_ReadAIHexAll (slot,hval);
```

[C/C++] Windows

```
int slot=0;
short hVal[16];
pac_i8015W_Init(slot);
pac_i8014W_ReadAIHexAll (slot,hVal);
```

[C#]

```
int slot=0;
Int16[] hval = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
pac8015WNet.pac8015W.pac_i8015W_Init(slot);     // initialize
pac8015WNet.pac8015W.pac_i8015W_ReadAIHexAll(slot, hval);
```

3.2.7. i8015W_ReadAI

This function is used to read temperature value on specific channel in float format.

Syntax

For MiniOS7

```
short i8015W_ReadAIHex(int slot,short ch, float* fval);
```

For Windows (CE and WES)

```
short pac_i8015W_ReadAIHex(int slot,short ch, float* fval);
```

Parameter

slot:

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

ch:

Specific channel (0 - 7).

**fval:*

[Output] float format data.

Return Values

Please refer to the Error Code.

Example

[C/C++] MiniOS7

```
int slot = 1,ch=0;
float fval=0.0;
i8015W_Init(slot);       // initialize
i8015W_ReadAIHex(slot,ch,&fval);
```

[C/C++] Windows

```
int slot=1,ch=0;
float fval=0.0;
pac_i8015W_Init(slot);       // initialize
pac_i8015W_ReadAIHex(slot,ch,&fval);
```

[C#]

```
int slot = 1, ch = 0;
short hval = 0;
pac8015WNet.pac8015W.pac_i8015W_Init(slot);     // initialize
pac8015WNet.pac8015W.pac_i8015W_ReadAIHex(slot, ch, ref hval);
```

3.2.8. i8015W_ReadAIAll

This function is used to read temperature value on all channels in float format.

Syntax

For MiniOS7

```
short i8015W_ReadAIAll(int slot, float fVal[]);
```

For Windows

```
short pac_i8015W_ReadAIAll (int slot, float fVal[]);
```

Parameter

slot:

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

** fVal[]:*

[Output] the float format data array.

Return Values

0 = No Error.

Example

[C/C++] MiniOS7

```
int slot=0;
float fVal[16];
i8015W_ReadAIAll(slot, fVal);
```

[C/C++] Windows

```
int slot=0;
float fVal[16];
pac_i8015W_ReadAIAll(slot, fVal);
```

[C#]

```
int slot=0;
float[] fval = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
pac8015WNet.pac8015W.pac_i8015W_ReadAIAll(slot, fval);
```

3.2.9. i8015W_SetTypeCode

This function code is used to set the RTD type code.

For MiniOS7

```
short i8015W_SetTypeCode(int slot, short ch, short typecode);
```

For Windows

```
short pac_i8015W_SetTypeCode(int slot, short ch, short typecode);
```

Parameter

slot:

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

ch:

Specific channel (0 - 7).

TypeCode:

RTD Type code value, see Section “3.5 RTD Type Code Table” for details.

Return Values

0 = No Error.

Example

[C/C++]

```
int slot=0,ch=0,typecode=0x20;
i8015W_SetTypeCode(slot, ch, typecode);
```

[C/C++]

```
int slot=0,ch=0,typecode=0x20;
pac_i8015W_SetTypeCode(slot, ch, typecode);
```

[C#]

```
int slot=0,ch=0,typecode=0x20;
pac8015WNet.pac8015W.pac_i8015W_SetTypeCode(slot, ch, typecode);
```

3.2.10. i8015W_GetTypeCode

This function code is used to get the RTD type code.

For MiniOS7

```
short i8015W_GetTypeCode(int slot, short ch, short* typecode);
```

For Windows

```
short pac_i8015W_SetTypeCode(int slot, short ch, short* typecode);
```

Parameter

slot:

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

ch:

Specific channel (0 - 7).

**TypeCode:*

[Output] RTD Type code value, see Section “3.5 RTD Type Code Table” for details.

Return Values

0 = No Error.

Example

[C/C++]

```
int slot=0,ch=0,typecode=0x20;
i8015W_GetTypeCode(slot, ch, typecode);
```

[C/C++]

```
int slot=0,ch=0,typecode=0x20;
pac_i8015W_GetTypeCode(slot, ch, typecode);
```

[C#]

```
int slot=0,ch=0,typecode=0;
pac8015WNet.pac8015W.pac_i8015W_GetTypeCode(slot, ch,ref typecode);
```

3.2.11. i8015W_SetMovingAverage

This function is used to set the Number of moving average for temperature value.

Syntax

For MiniOS7

```
short i8015W_SetMovingAverage (int slot,short count);
```

For Windows

```
short pac_i8015W_SetMovingAverage(int slot,short count);
```

Parameter

slot:

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

count:

1 ==> Do not use averaging

n ==> Average of n consecutive sample value, n:1~128

Return Values

0 = No Error.

Example

[C/C++] MiniOS7

```
int slot=0, count=1;
i8015W_SetMovingAverage(slot, count);
```

[C/C++] Windows

```
int slot=0, count=1;
pac_i8015W_SetMovingAverage(slot, count);
```

[C#]

```
int slot=0, count=1;
pac8015WNet.pac8015W.pac_i8015W_SetMovingAverage(slot, count);
```

3.2.12. i8015W_GetMovingAverage

This function is used to get the Number of moving average for temperature value.

Syntax

For MiniOS7

```
short i8015W_SetMovingAverage (int slot,short* count);
```

For Windows

```
short pac_i8015W_SetMovingAverage(int slot,short* count);
```

Parameter

slot:

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

**count:*

[Output] 1 ==> Do not use averaging

n ==> Average of n consecutive sample value, n:1~128

Return Values

0 = No Error.

Example

[C/C++] MiniOS7

```
int slot=0, count=1;
i8015W_GetMovingAverage(slot, &count);
```

[C/C++] Windows

```
int slot=0, count=1;
pac_i8015W_GetMovingAverage(slot, &count);
```

[C#]

```
int slot=0, count=1;
pac8015WNet.pac8015W.pac_i8015W_GetMovingAverage(slot, ref count);
```

3.3. Demo Programs

This chapter provides demo programs that demonstrate how to use the I-8015W/I-8015W-12 I/O modules in different environments. Before running the demos, developers must ensure that the correct header and library files are included in their projects. The following tables summarize the required files and available demo programs for each PAC platform.

Demo Program Requirements by Platform

The required files and supported demo programs vary depending on the PAC environment. The following table provides a quick reference for MiniOS7, WinCE, and Windows-based PACs.

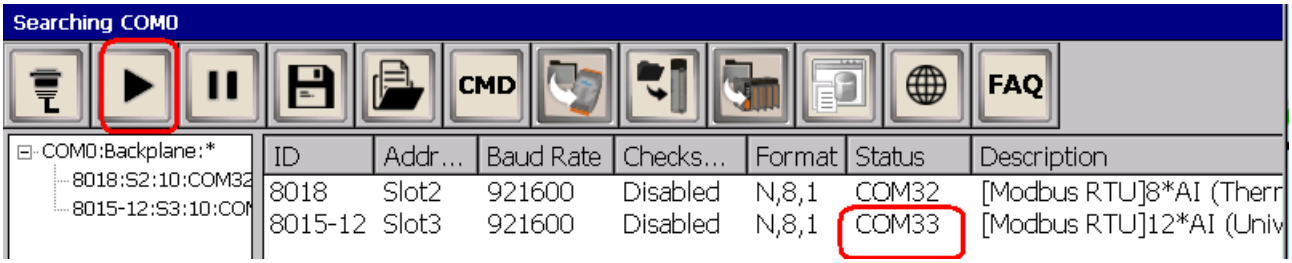
Platform	Required Header & Library	Available Demo Programs
MiniOS7-Based PACs	i8015W.h、8015W.LIB	Demo: I-8015W_MiniOS7
WinCE-Based PACs	pac_i8015W.h、pac_i8015W.lib、pac_i8015W.dll、pac_i8015WNet.dll	C++ Demo: i8015W_Basic_Info_Cplusplus C# Demo: i8015W_Basic_Info_Csharp C# nModbus Demo: i8015S_nModbusRTU_master_demo
Windows-Based PACs		

3.3.1. Read Temperature Value

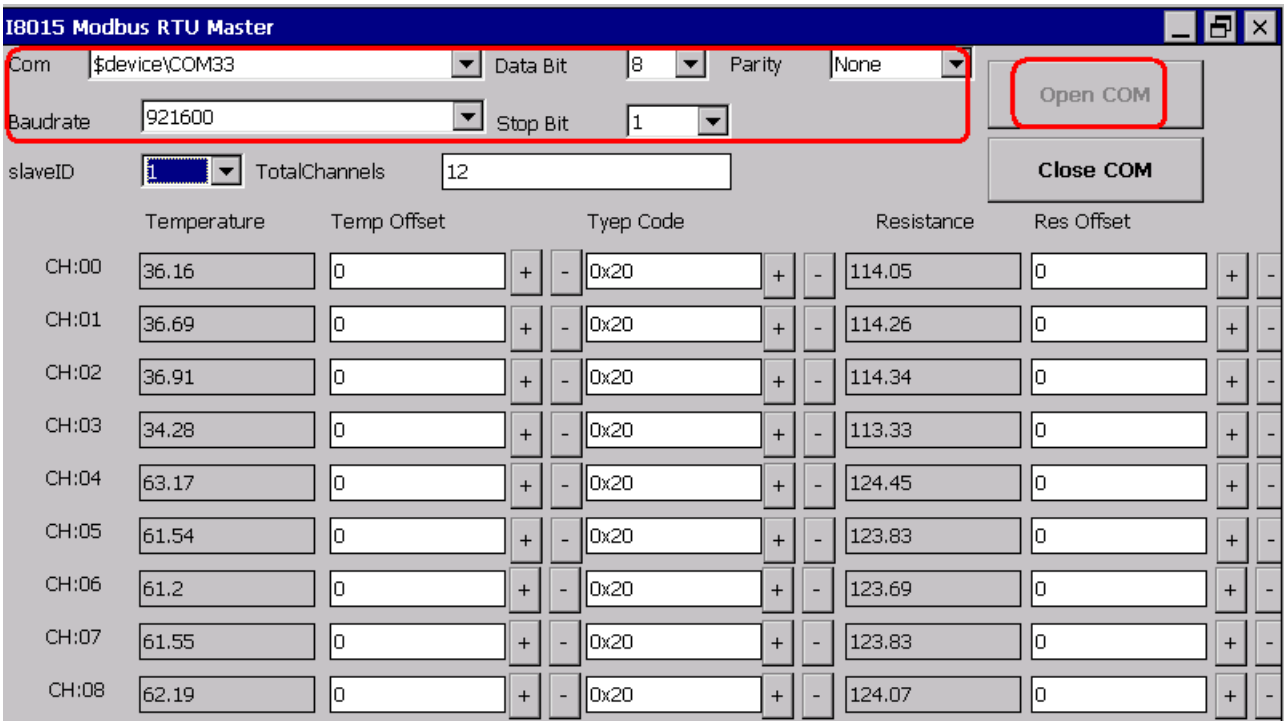
This demo describes how to retrieve temperature values from the I-8015W/I-8015W-12 I/O module using both Modbus communication and API calls. It verifies whether the module can correctly acquire and report RTD measurement data.

nModbus Demo (WinCE-based/Windows-Based)

Step 1. Use **DCON Utility** to confirm the COM port assigned to the I-8015W.



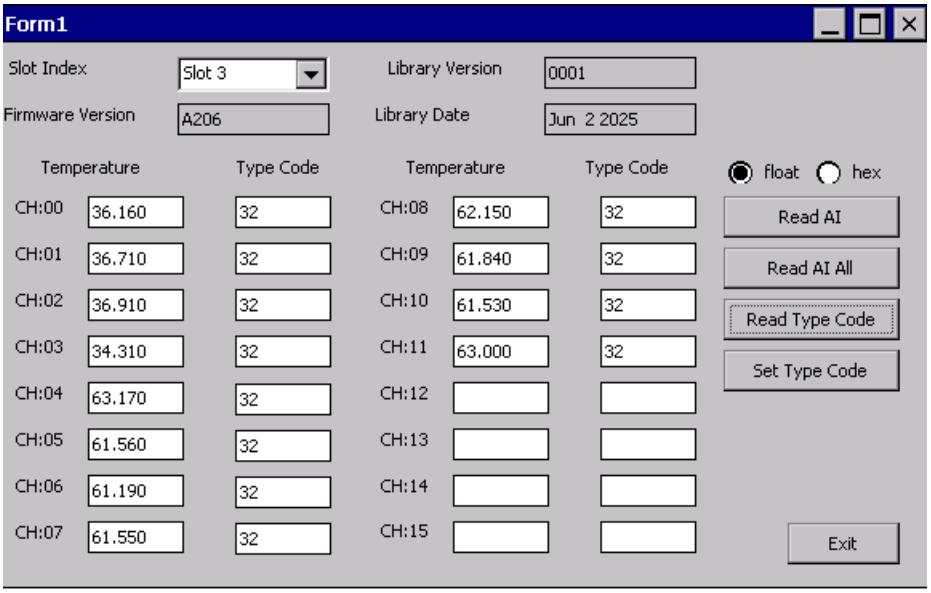
Step 2. Execute the **nModbus demo**, specifying COM settings (e.g., COM33,921600,n,8,1) to read AI values.



I-8015w API Demo (WinCE-based/Windows-Based)

Step 1. Use **DCON Utility** to confirm the COM port assigned to the I-8015W.

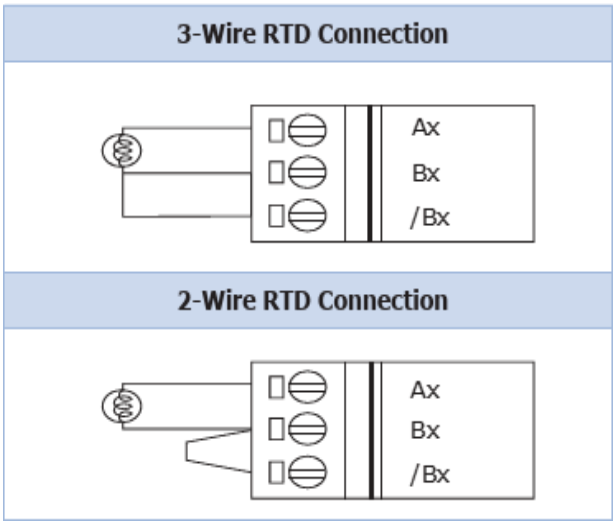
Step 2. Execute the **8015W_Basic_Info_Csharp demo**, specifying slot index settings to read AI values.



3.3.2. Set the offset Value

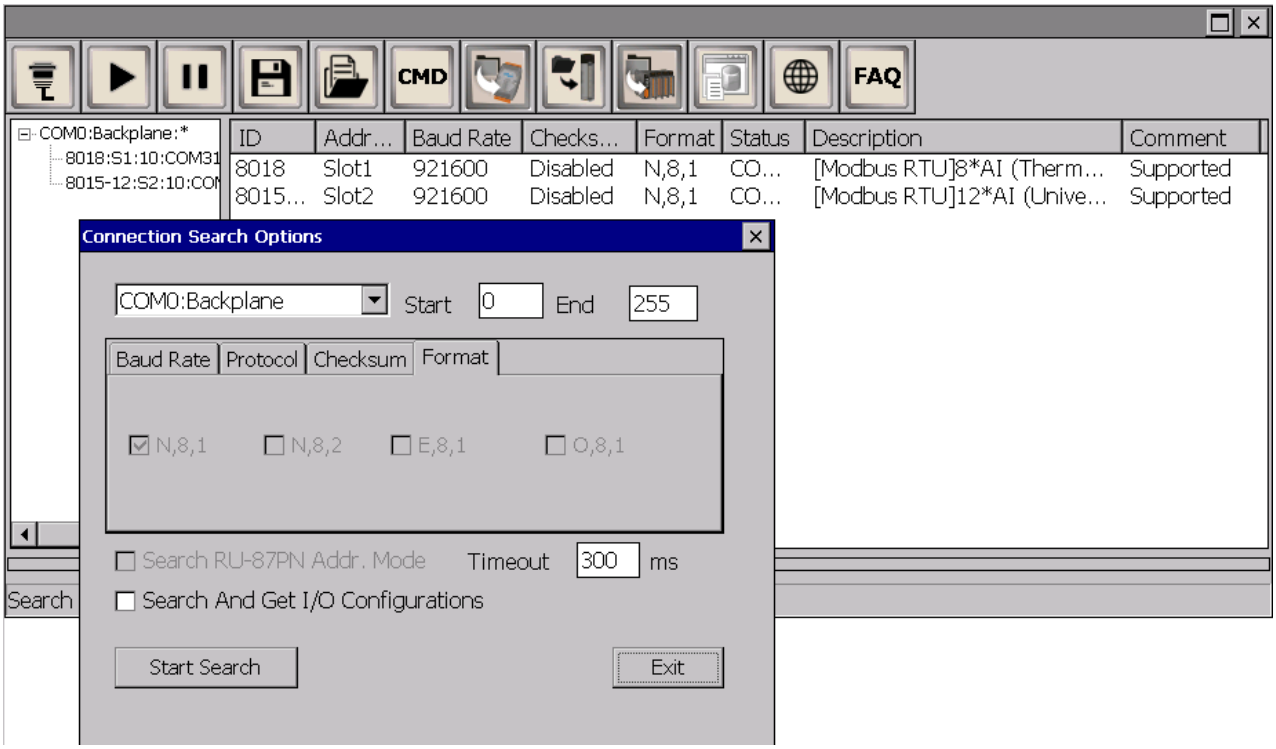
This demo describes how to apply offset values to temperature and resistance channels. Offset adjustment helps calibrate the readings so that they align with real-world reference values.

Step 1. Connect a stable input signal (e.g., 2-wire or 3-wire RTD) to the module. Insert the module into a slot and power on the controller.

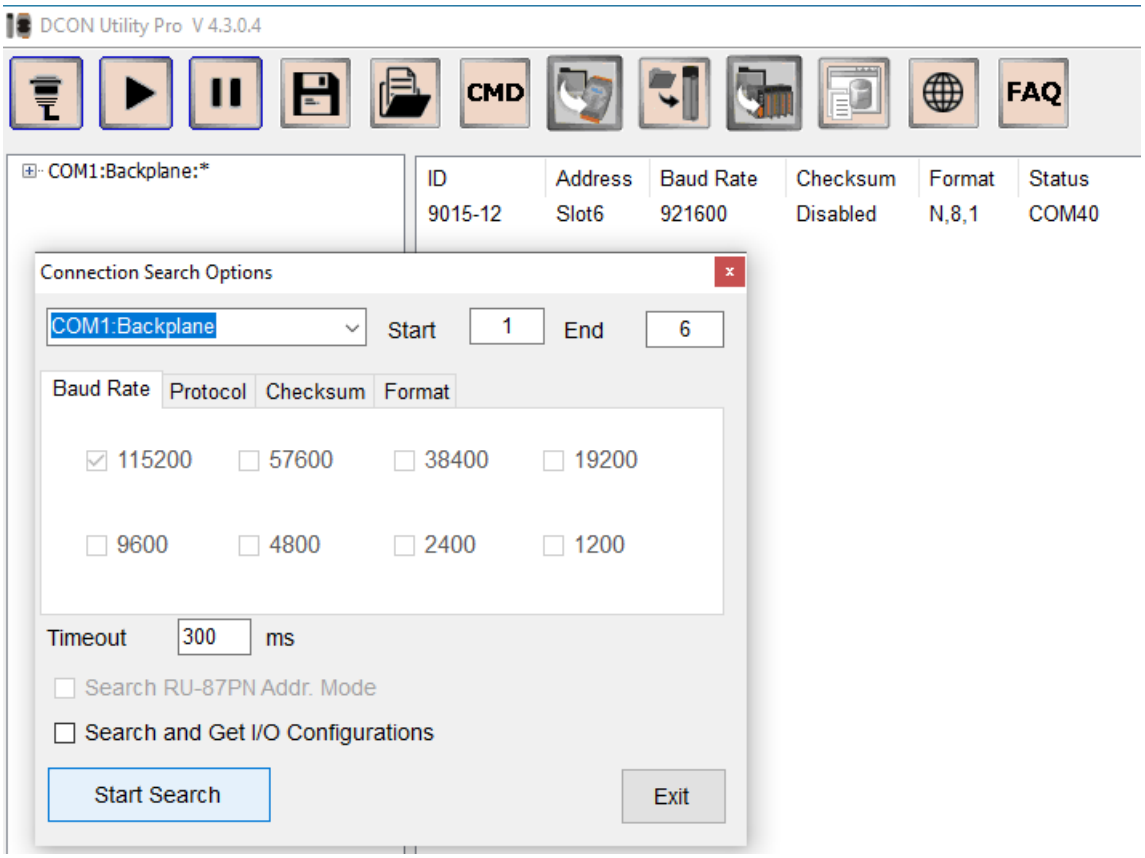


Step 2. Launch **DCON Utility** and select the “8015” module.

For WinCE-based



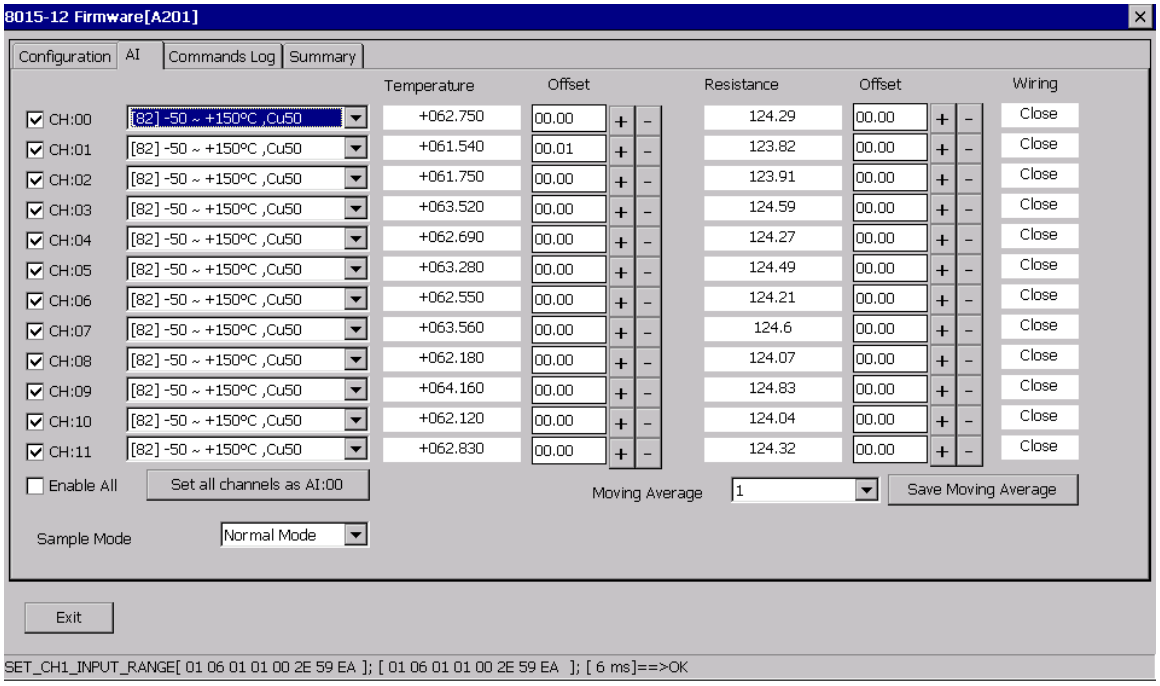
For Windows-based



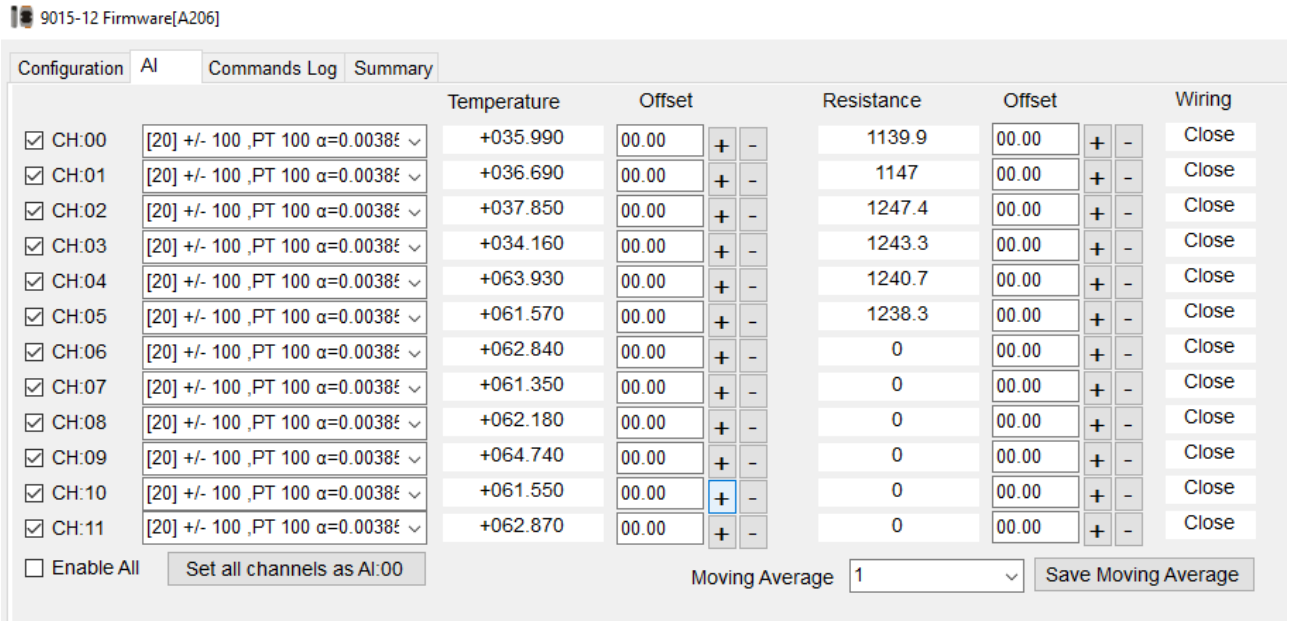
Step 3. Set Temperature offset value and resistance offset in the AI tab.

- **Temperature offset**: Set in AI tab or via Modbus registers 40289–40296 (unit: 0.01 °C).
- **Resistance offset**: Set in AI tab or via Modbus registers 40385–40392 (unit: 0.01 Ω).

For WinCE-based



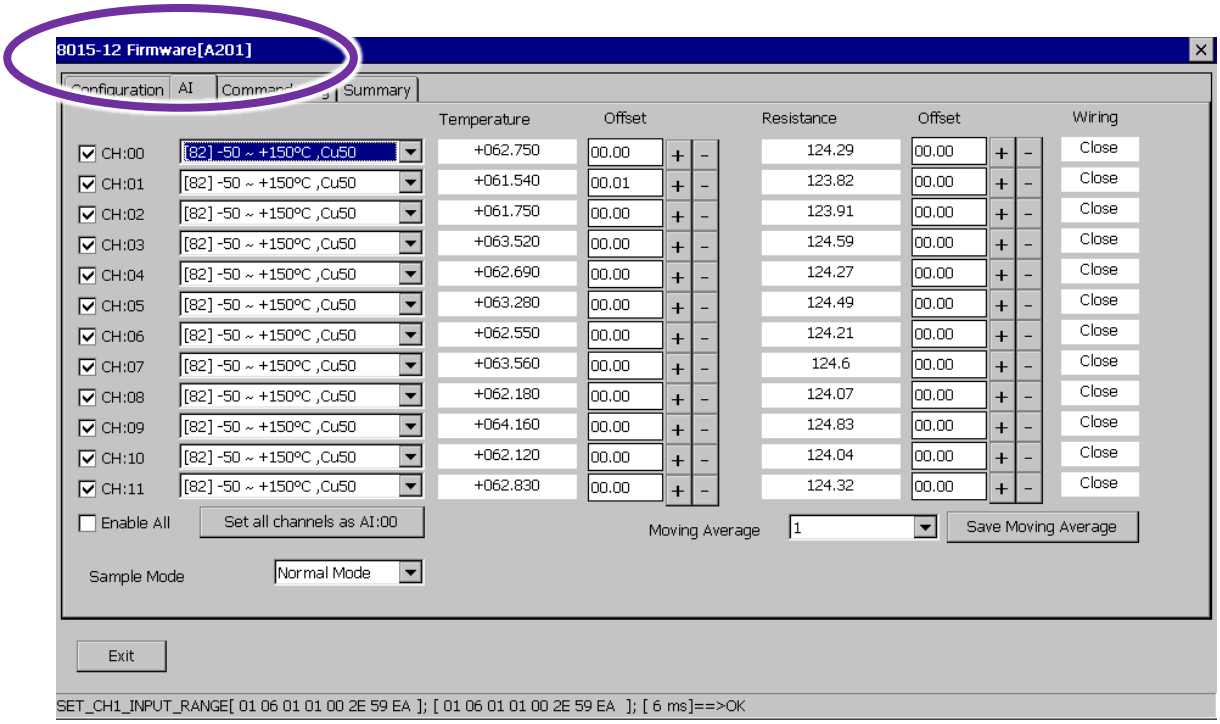
For Windows-based



3.3.3. Get the Firmware Version

This demo describes how to check the firmware version of the I-8015W/I-8015W-12 I/O module. Firmware version information is useful for system validation, compatibility checks, and troubleshooting.

Launch **DCON Utility** and select the I-8015W module to view firmware version in the control form.



Alternatively, use Modbus registers to read version information:

- **Firmware version (low word)**: Modbus registers 40481.
- **Firmware version (high word)**: Modbus registers 40482.

4. Troubleshooting

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

If you have any difficulty using I-8015w series modules, please check the relevant information here. If you cannot solve your problem or have good suggestions and comments about ICP DAS products, please visit the ICP DAS website to contact us. We will serve you wholeheartedly with the most powerful technical force.

Email: service@icpdas.com

Website: http://www.icpdas.com

Communication related

Under permitted conditions, you can also use RS-232 to communicate with the module.

After using SendToCOM.exe to open the COM port mounted by I-8015w, you can send the DCON command $01M to test whether the communication is normal.

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

Appendix C. Revision History

This chapter provides revision history information to this document.

The table below shows the revision history.

Revision	Date	Description
1.0.1	September 2025	Initial issue