EX-9200

Analog I/O series

Data Acquisition Modules
User's Manual

Web site: www.topsccc.com.tw

Trademark:
The names used in this manual for identification only maybe registered trademarks of their respective company
Rev

Table of Contents

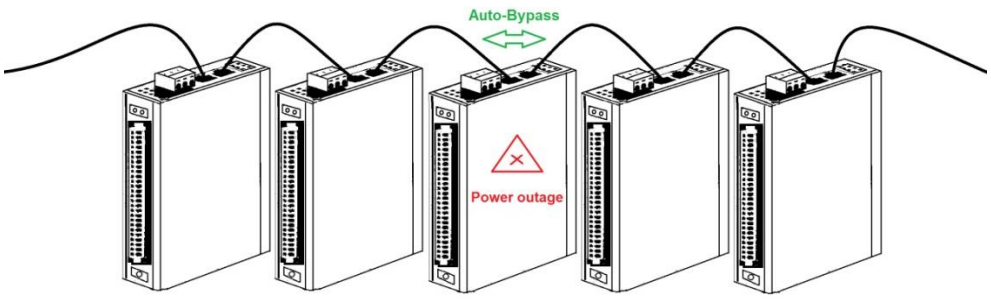
Chapter 1 Product Overview

1.1 Introduction

EX-9200 is based on the popular Ethernet networking standards used today in most business environments. Users can easily add EX-9200 I/O modules to existing Ethernet networks or use EX-9200 modules in new Ethernet-enabled Manufacturing networks. EX-9200 module features a 10/100 Mbps Ethernet switching chip to allow *Daisy Chain connections* in an Ethernet network, making it easier to deploy, and supports industrial popular Modbus/TCP protocol over TCP/IP for data connection. EX-9200 also supports UDP protocol over Ethernet networking. With UDP/IP, EX-9200 I/O modules can actively send I/O data stream to 8 Ethernet nodes. Through Ethernet networking HMI/SCADA system and controller can access or gather real-time data from EX-9200 Ethernet enabled DA&C modules. And, these real-time data can be integrated with business system to create valuable, competitive business information immediately.

1.2 Daisy Chain connection with auto-bypass protection

EX-9200 module has built in two-port Ethernet switches to allow daisy chain connections in an Ethernet network, making it easier to deploy, and helping improve scalability. The two Ethernet ports are fully compliant with IEEE 802.3u 10/100Mbpst through standard RJ-45 connectors Although daisy chain topology brings attractive benefits for users, it still comes with the risk that once any device in the daisy-chain network suffers power outage, it will cause the disconnection of all devices downstream, to prevent this critical issue from happening, ExpertDAQ especially refined the hardware design of EX-9200 so that it(Auto-bypass protection) can rapidly recover the network connection in about 1 seconds, Therefore, the damage will be greatly minimized.



Daisy-Chain Ethernet Cabling

Note: ***Auto-Bypass Protection*** feature guarantees the Ethernet communication. It will automatically active to continue the network traffic when the EX-9200 modules loses its power after 2 second.

1.3 Mixed I/O in One Module to fit all applications

EX-9200 mixed I/O module design concept provides the most cost-effective I/O usage for application system. The most common used I/O type for single function unit are collected in ONE module. This design concept not only save I/O usage and spare modules cost but also speed up I/O relative operations. For small DA&C system or standalone control unit in a middle or large scale, EX-9200 mixed I/O design can easily fit application needs by one or two modules only. With additional embedded control modules, EX-9200 can easily create a localized, less complex, and more distributed I/O architecture.

1.4 Industrial standard Modbus/TCP protocol supported for open connectivity

EX-9200 modules support the popular industrial standard, Modbus/TCP protocol, to connect with Ethernet Controller or HMI/SCADA software built with Modbus/TCP driver. ExpertDAQ also provides OPC server for Modbus/TCP to integrate EX-9200 I/O real-time data value with OPC client enabled software. Users don't need to take care of special driver's development.

1.5 Features

1.5.1 Power-on & Safe value

- Power-on value:

Power-on value is used to set the module default output value when the module is turned-on or watch dog timeout reset. This function is especially importance in some application where the specific initial output states are required User can set power on value by sending Set power-on/safe value command

- Safe value:

Safe value are used to set the module outputs into the specific values when Host watchdog timeout If The host watchdog timer is enabled by sending Set host watchdog timeout value, the host should send Host OK command periodically within Timeout value to refresh the timer, otherwise the module will be forced to safety state.

1.6 Common technical specification of EX-9200

- Ethernet: 10 BASE-T IEEE 802.3 100 BASE-TX IEEE 802.3u
- Wiring: UTP, category 5 or greater
- Bus Connection: Two-port RJ45 modular jack(Auto-bypass protection)
- Comm. Protocol: Modbus/TCP on TCP/IP and UDP
- Data Transfer Rate: Up to 100 Mbps
- Unregulated 10 to 30VDC
- Protection: Over-voltage and power reversal
- Status Indicator: Power, CPU, Communication (Link, Collide, 10/100 Mbps, Tx, Rx)
- Mounting: DIN rail or wall
- Wiring: I/O cable 14 to #28 AWG wire for terminal block.
- Operating Temperature: - 10 to 70º C (14 to 158º F)
- Storage Temperature: - 25 to 85º C (-13 to 185º F)
- Humidity: 5 to 95%, non-condensing
- Atmosphere: No corrosive gases

NOTE:
Equipment will operate below 30% humidity. However, static electricity problems occur much more frequently at lower humidity levels. Make sure you take adequate precautions when you touch the equipment. Consider using ground straps, anti-static floor coverings, etc. if you use the equipment in low humidity environments.

1.7 Software Support

Based on the Modbus/TCP standard, the EX-9200 firmware is a built-in Modbus/TCP server. Therefore, ExpertDAQ provides the necessary DLL drivers, and Windows Utility for users for client data for the EX-9200. Users can configure this DA&C system via Windows Utility; integrate with HMI software package via Modbus/TCP driver or Modbus/TCP OPC Server. Even more, you can use the DLL driver and ActiveX to develop your own applications.

1.8 Package Information

The package of EX-9200 series module will contain the following items. Please check and feel free to contact us if any part missing or damaged after purchasing EX-9200 product.

- EX-9200 module (assembled with DIN Rail)
- Product CD
- Panel mounting bracket
- Start-up manual

1.9 Product Warranty (1 years)

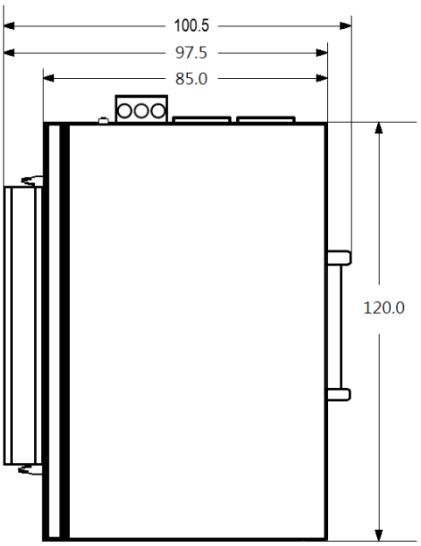
ExpertDAQ warrants to you, the original purchaser, that each of its products will be free from defects in materials and workmanship for one year from the date of purchase. This warranty does not apply to any products which have been repaired or altered by persons other than repair personnel authorized by ExpertDAQ, or which have been subject to misuse, abuse, accident or improper installation. ExpertDAQ assumes no liability under the terms of this warranty as a consequence of such events.

Because of ExpertDAQ's high quality-control standards and rigorous testing, most of our customers never need to use our repair service. If an ExpertDAQ product is defective, it will be repaired or replaced at no charge during the warranty period. For out-of-warranty repairs, you will be billed according to the cost of replacement materials, service time and freight. Please consult your dealer for more details.

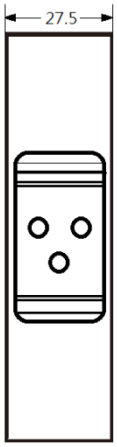
1.10 Dimensions

The following diagrams show the dimensions of the EX-9200 I/O module in millimeters.

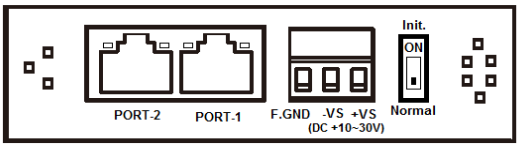
- **EX-9200 series**



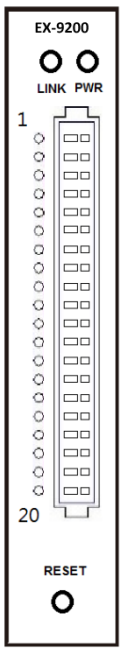
Side View



Rear View



Top View

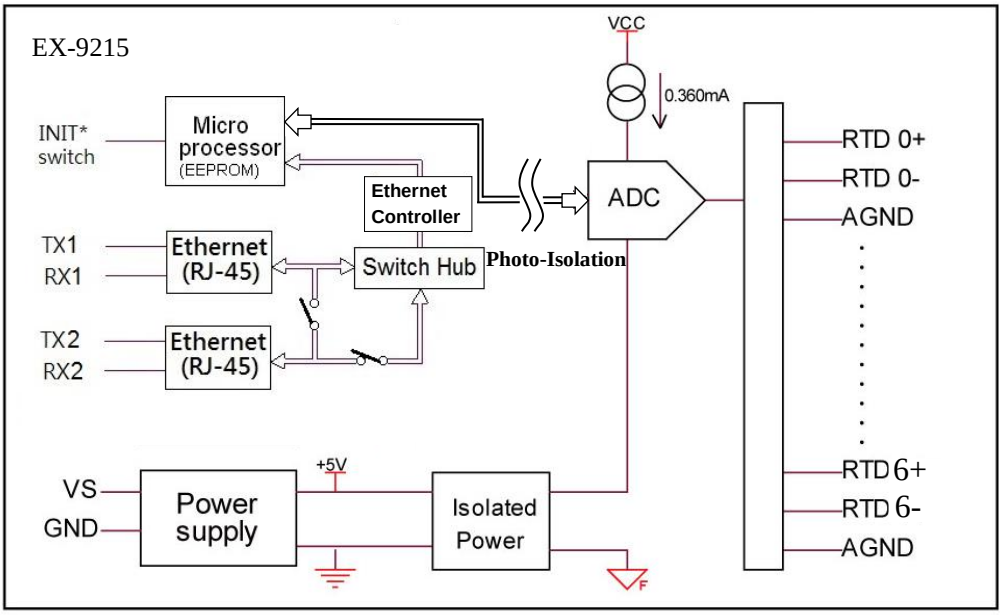


Front View

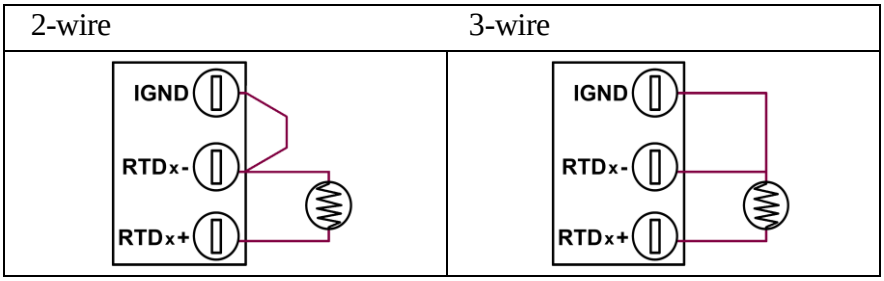
Chapter 2 Block diagram of DIO modules

2.1 EX-9215

2.1.1 Block diagram

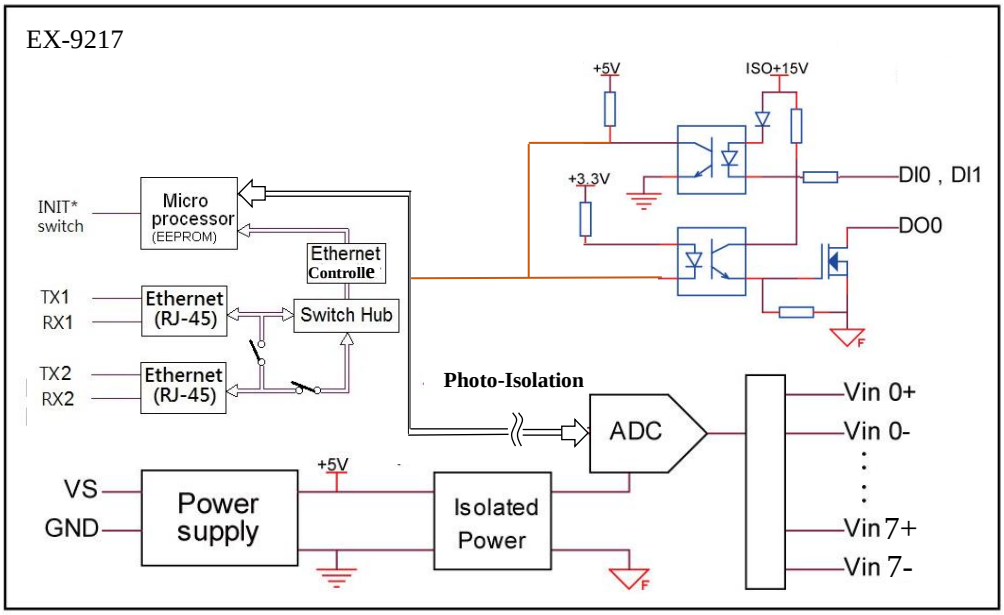


2.1.2 Wire connection

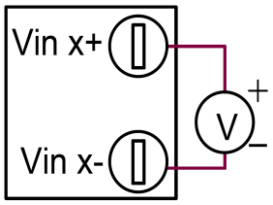
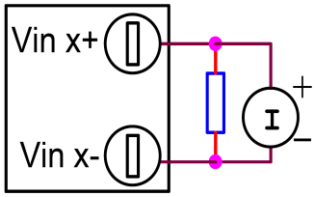


2.2 EX-9217

2.2.1 Block diagram

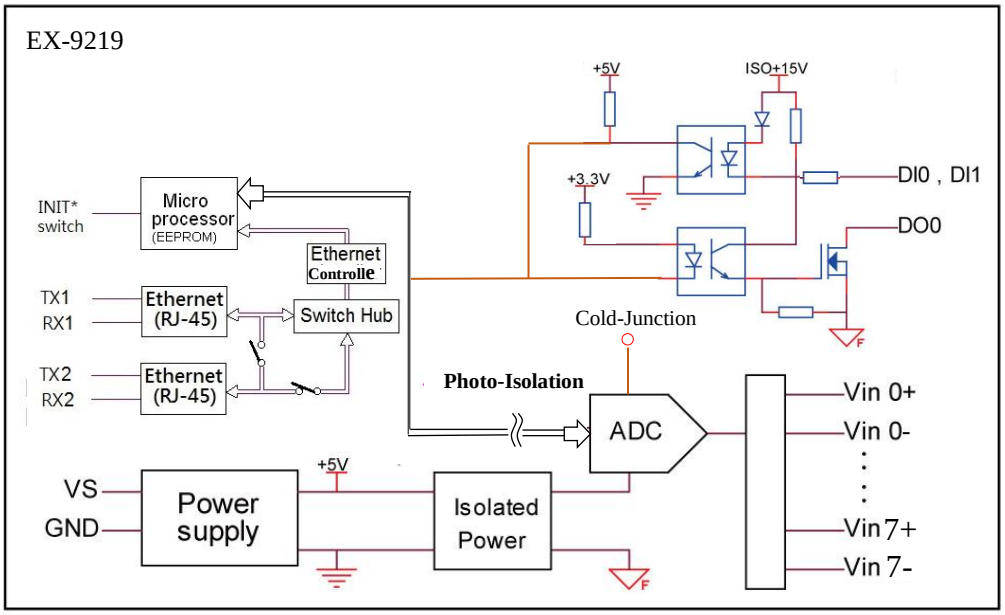


2.2.2 Wire connection

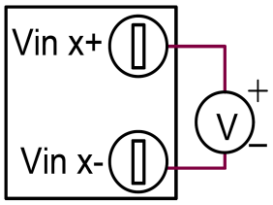
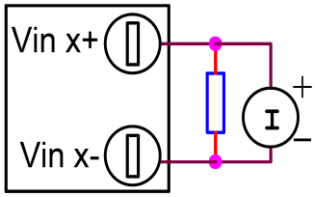
Voltage	Current
	

2.3 EX-9219

2.3.1 Block diagram



2.3.2 Wire connection

Voltage	Current
	

Chapter 3 System Requirements

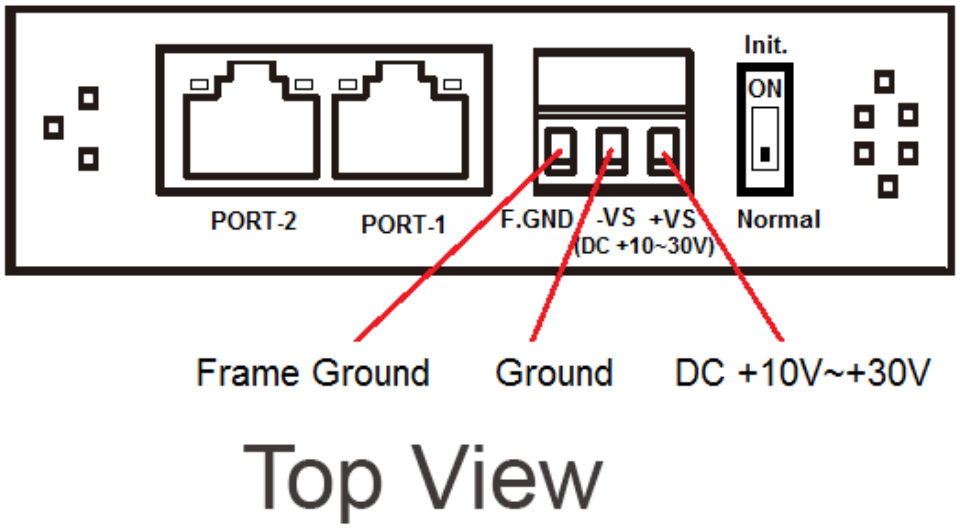
- IBM PC compatible computer with 486 CPU (Pentium is recommended)
- Microsoft 95/98/2000/NT 4.0 (SP3 or SP4)/XP/Win 7,8,10 or higher versions
- At least 32 MB RAM
- 20 MB of hard disk space available
- VGA color monitor
- 2x or higher speed CD-ROM
- Mouse or other pointing devices
- 10 or 100 Mbps Ethernet Card
- 10 or 100 Mbps Ethernet Hub (at least 2 ports)
- Two Ethernet Cable with RJ-45 connector
- Power supply for EX-9200 (+10 to +30 V unregulated)

3.1 Wiring and Connections

This section provides basic information on wiring the power supply, I/O units, and network connection.

3.2 Power supply wiring

Although the EX-9200/TCP systems are designed for a standard industrial unregulated 24 V DC power supply, they accept any power unit that supplies within the range of +10 to +30 VDC. The power supply ripple must be limited to 200 mV peak-to-peak, and the immediate ripple voltage should be maintained between +10 and +30 VDC. Screw terminals +Vs and GND are for power supply wiring.

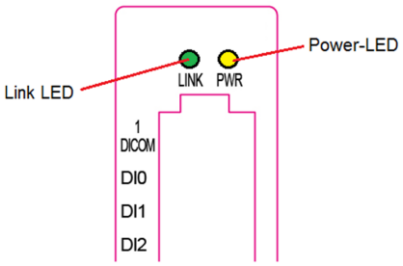


Top View

Note: The wires used should be sized at least 2 mm.

3.3 Status LED indicator for EX-9200 I/O modules

There are Tow flash types of the Status LED indicator on the front panel of EX-9200 series.



No.	Color	LED Status	Definition
1	Yellow (Power-LED)	On	Power-LED, Always ON.
2	Green (LINK-LED)	On	(LINK) This LED is normal on whenever the EX-9200 module's Ethernet wiring is connected
3	Green (LINK-LED)	Blinking	(COM) Blinks whenever EX-9200 module is transmitting or receiving data(I/O command) via Ethernet.

3.4 Ethernet LED indicator of EX-9200

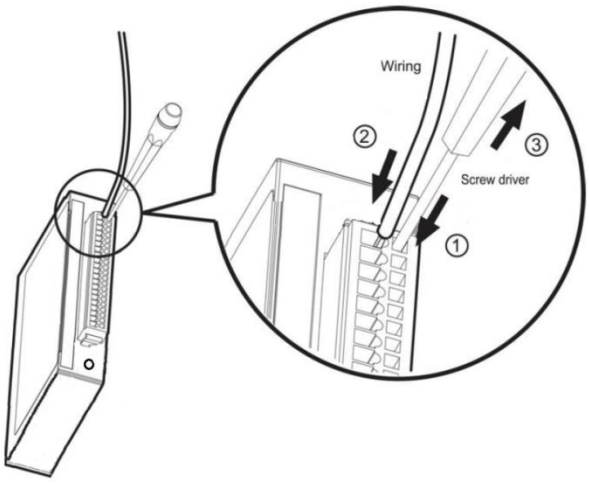
There are two ports of RJ-45 ethernet connector. Each RJ-45 port LEDs built with two indicators to represent the EX-9200 ethernet status as explained below:

- Yellow indicator (Speed): This LED is always ON.
- Green indicator(Link): This LED is normal on whenever the EX-9200 module's Ethernet wiring is connected.

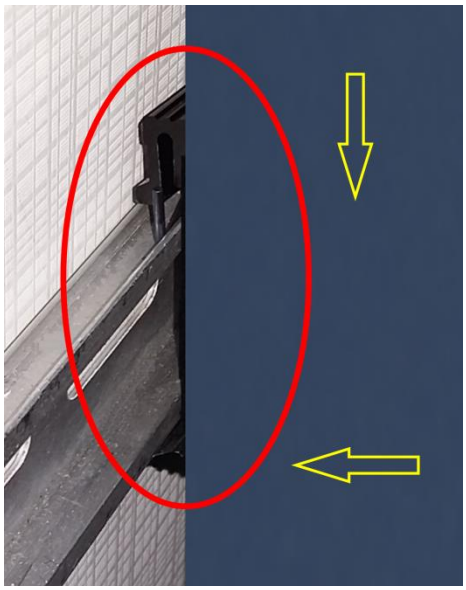
3.5 I/O modules wiring

The system uses a plug-in screw terminal block for the interface between I/O modules and field devices. The following information must be considered when connecting electrical devices to I/O modules. The terminal block accepts wires from # 14 AWG ~ 28 AWG. Always use a continuous length of wire. Do not combine wires to make them longer.

1. Insert the screw driver into the left hole of the terminal.
2. Insert the wiring into the left hole of the terminal.

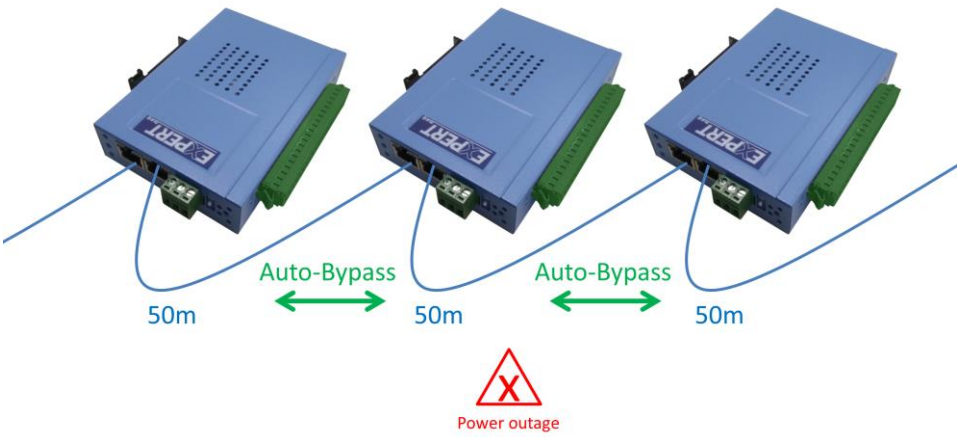


3.6 Rear side installation



3.7 Daisy Chain Limitations

In general, an ethernet cable distance of each 100BASE-TX network segment is can be run 100 meters (about 328 feet). However, cables can pick up electrical noise on long runs. Based on this limitation, the maximum total connection length in daisy chain wiring should also be 100m as if auto-bypass protection active. For example, the distance from first to second module is 50m, so as second to third. When the power fails on second (middle) module, the auto-bypass will activate to bridge the network connection. The total distance from first to the 3rd will become 100m, that means the total network segment is close to limitation.



3.8 Initializing a Module

All EX modules in an Ethernet network must have a *unique IP* address. Therefore, to configure the brand-new EX before using is necessary.

3.8.1 Factory default settings:

- IP Address : 10.0.0.1
- Subnet Mask: 255.255.255.0
- Gateway: 10.0.0.1
- DHCP: Disabled
- Web Server: Disabled
- Module ID: 01
- Password: 00000000

3.8.2 INIT* State settings:

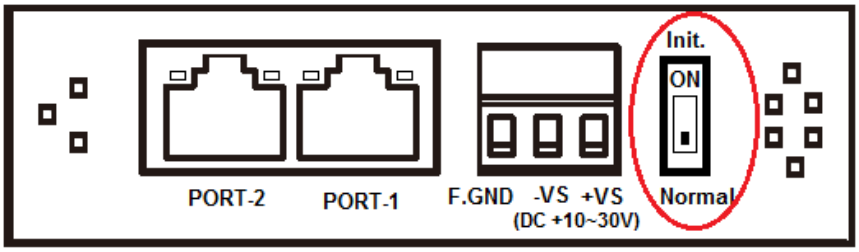
All EX-9200 I/O modules have a special slide-switch as INIT-SWITCH (Ref. Appendix). The I/O modules must be set at "***INIT" State*** when you want to change the default settings, such as the *IP address,* Subnet Mask, Gateway, Password etc. If the "INIT" switch set to "INIT-ON" mode when power ON, Under this state the default configuration is set as following :

- IP Address : 10.0.0.1
- Subnet Mask: 255.255.255.0
- Gateway: 10.0.0.1
- Password: 00000000

Press the "**Default**" button on the EX-9200 utility(Network), the module will be set to factory default state as following :

- IP Address : 10.0.0.1
- Subnet Mask: 255.255.255.0
- Gateway: 10.0.0.1
- DHCP: Disabled
- Web Server: Disabled
- Module ID: 01
- Password: 00000000
- I/O: factory default Mode

Note: Each module must has a unique ID number to be identified when the DHCP enabled, because you would not know the module IP address when DHCP enabled, but if with the different ID number. You can call provided function call(TCP_GetIPFromID() in TCPDAQ.dll) to get correct IP address for each ID number



INIT. Switch

Top View

Chapter 4 Specification

4.1 EX-9215 Specificatiom

The EX-9215 is a 16-bit, 12-channel RTD input module that provides programmable input ranges on all channels. It accepts Various RTD inputs (Type PT100, PT1000, Balco 500, NI604, NI1000) and provides data to the host computer.

- Analog Input

Effective Resolution: 16-bit

Channels：12

Input Type：PT100, PT1000, Balco 500, NI RTD

Input Range：

PT100 Type: -50 ~ 150°C/0 ~ 100°C, 0 ~ 200°C, 0 ~ 400°C, -200 ~ 200°C

PT1000 Type: -40 ~ 160°C

Balco 500 Type: -30 ~ 120°C

Ni604 Type: -80 ~ 100°C

Ni1000 Type: -0 ~ 100°C

- Sampling Rate：10 samples/sec
- Input Impedance：10 MΩ
- Accuracy：±0.15% or better
- Zero Drift：±20 μV/ °C
- Span Drift：25 ppm/ °C
- Built-in Watchdog Timer
- Power Requirements：USB powered (400mA max.) or external unregulated +10 ~ +30 VDC
- Power Consumption：1.5 W/Typical, 2W/max

4.2 EX-9217 Specificatiom

The EX-9217 is a 16-bit, 16-channel Analog input module that provides programmable input ranges on all channels.

- Analog Input

  Effective Resolution：16-bit

  Channels：16

  Input Type：Voltage, Current

  Input Range：±10V, ±5V, ±2.5V, ±1V, ±500mV, ±150mV, 0~20mA, 4~20mA

  Sampling Rate：10 samples/sec.

  Input Impedance：10 MΩ

  Accuracy：±0.15% or better

  Zero Drift：±20 μV/ °C

  Span Drift：25 ppm/ °C

- Digital Input

  Input Channel：2 channels

  Input Type：Voltage (logic 0 for 0<Vin < 3Vdc , logic 1 for 5V<Vin < 24Vdc) or Switch On/Off

  Isolation Voltage：2000 V

- Digital Output

  Output Channel：1 channel

  Output Type：Open Collect to 30Vdc/3A(max)

  Isolation Voltage：2000 V

- Built-in Watchdog Timer

- Power Requirements：USB powered (400mA max.) or external unregulated +10 ~ +30 VDC
- Power Consumption：1.5 W/Typical, 2W/max

4.3 EX-9219 Specificatiom

ranges on all channels. It accepts Various Thermocouple inputs (Type J, K, T, E, R, S, B) and provides data to the host computer in engineering units (ºC). In order to satisfy various temperature requirements in one module, each analog channel is allowed to configure an individual range for several applications.

- Analog Input
  - Effective Resolution：16-bit
  - Channels：16
  - Input Type：J, K, T, E, R, S, B
  - Input Range：

J Type：0 ~ 760 °C

K Type：-100 ~ 1370 °C

T Type：-100 ~ 400 °C

E Type：0 ~ 1000 °C

R Type：500 ~ 1750 °C

S Type：500 ~ 1750 °C

B Type：500 ~ 1800 °C

+/-2.5V,+/-1.0V,+/-500mV,+/-150mV,0~20mA,4~20mA

  - Sampling Rate：10 samples/sec, 20 samples/sec, 50 samples/sec.
  - Input Impedance：10 MΩ
  - Accuracy：±0.15% or better
  - Zero Drift：±20 µV/ °C
  - Span Drift：±25 ppm/ °C
- Digital Input
  - Input Channel：2 channels
  - Input Type：Voltage (logic 0 for 0<Vin < 3Vdc , logic 1 for 5V<Vin < 24Vdc) or Switch On/Off
  - Isolation Voltage：2000 VDC
- Digital Output
  - Output Channel：1 channels
  - Output Type：Open Collect to 30Vdc/3A(max)
  - Isolation Voltage：2000 VDC
- Built-in Watchdog Timer
- Power Requirements：USB powered (400mA max.) or external unregulated +10 ~ +30 VDC
- Power Consumption：1.5 W/Typical, 2W/max

Chapter 5 Appendix

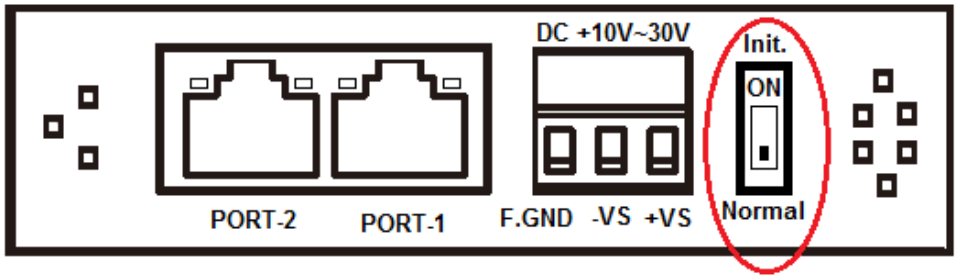
5.1 INIT* switch operation

The EX-9200 "INIT*mode" has two purposes, one for reading module current configuration, and another for configuring the module **IP Address, Subnet Mask, and Gateway**.

♦ Reading module current configuration

Each EX module has a built-in EEPROM which is used to store the configuration information such as address ID, type, DIO mode etc.. If the user unfurtunally forget the configuration of the module. User may use a special mode called "INIT* mode" to resolve the problem When the module is set to "INIT* mode", the default settings are IP Address, Subnet Mask, and Default Gateway (10.0.0.1, 255.255.255.0 and 10.0.0.1)

♦ Originally, the INIT mode is accessed by connecting the INIT* terminal to the GND terminal. New EX-9200 modules have the INIT switch located on the rear side of the module to allow easier access to the INIT mode. For these modules, INIT mode is accessed by sliding the INIT switch to the Init position as shown below.



INIT. Switch

Top View

♦ The following steps show you how to enable INIT* mode and read the current configuration:

1. Power off the module.
2. Sliding the INIT switch to the "Init" position.
3. Power on the module.
4. Start up the Windows Utility, it will search all EX-9200 I/O modules on the host PC' to read the current configuration stored in the EEPROM and set new **IP Address, Subnet Mask, and Default Gateway**,
5. Power off the module again
6. Sliding the INIT switch to the "Normal" position.

♦ Factory default settings:

1. IP Address : 10.0.0.1
2. Subnet Mask : 255.255.255.0
3. Gateway : 10.0.0.1
4. DHCP : Disabled
5. Web Server : Disabled
6. Module ID : 00
7. Password : 00000000

5.2 Module Status

Power-On Reset will let all output go to Power-On Value. The module may accept the host's command to change the output value. Host Watchdog Timeout will let all digital output go to Safe Value if the host watchdog timeout flag is set, and the output command will be ignored. The module's LED will go to flash and user must reset the module status via command to restore normal operation.

5.3 Dual Watchdog Operation

Dual Watchdog = Module Watchdog + Host Watchdog

The Module Watchdog is a hardware reset circuit to monitor the module's operating status. While working in harsh or noisy environment, the module may be down by the external signal. The circuit may let the module to work continues and never halt. The Host Watchdog is a software function to monitor the host's operating status. Its purpose is to prevent the network/communication from problem or host halt. While the timeout occurred, the module will turn the all output into safe state to prevent from unexpected problem of controlled target. The E-9200 module with Dual Watchdog may let the control system more reliable and stable.

5.4 Reset Status

The reset status of a module is set when the module is powered-on or when the module is reset by the module watchdog. It is cleared after the responding of the first $AA5 command. This can be used to check whether the module had been reset. When the $AA5 command responds that the reset status is cleared, that means the module has not been reset since the last $AA5 command was sent. When the $AA5 command responds that the reset status is set and it is not the first time $AA5 command is sent, it means the module has been reset and the digital output value had been changed to the power-on value.

5.5 Input counter and Input latch

Input counter:

Each input channel has internal counter used to software count the state change (*falling edge*) of input signal (max. 300Hz). The counting value can be read and cleared by sending "*Read digital input counter command*" or " *Clear digital input counter command*".

Input latch:

Each input channel has internal latch which is used to latch the pulse signal from the input. This latched state can be read by sending "*Read latched digital input* " command and cleared by sending "*Clear latched digital input*" command. For example, if the digital input is connected to a key switch. The key switch is a pulse signal. The user may lose the strike information by sending command $AA6. The digital input latch can latch the pulse and ready be read by sending "*Read latched digital input* " command. If the latched state=1 means that there is a key strike occurred.

5.6 Power-on & Safe value

Power-on value:

Power-on value is used to set the module default output value when the module is turned-on or watch dog timeout reset. This function is especially importance in some application where the specific initial output states are required User can set power on value by sending *Set power-on/safe value* command

Safe value:

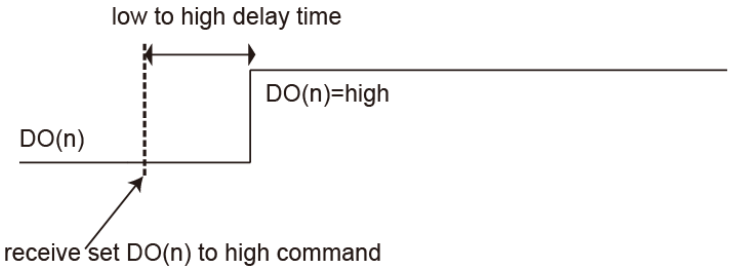
Safe value are used to set the module outputs into the specific values when Host watchdog timeout If The host watchdog timer is enabled by sending *Set host watchdog timeout value*, the host should send *Host OK* command periodically within Timeout value to refresh the timer, otherwise the module will be forced to safety state.

5.7 High/Low delay output mode

EX-9200 series modules supports **high-to-low and low-to-high delay** output function

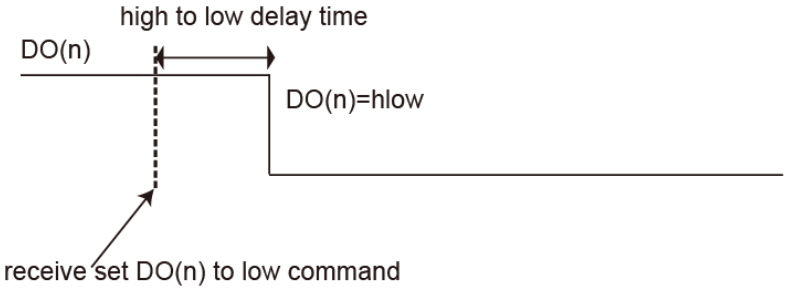
5.7.1 Low to High Delay output

When you choose *Low to High* delay mode, it is almost the same as choosing the DO direct output mode. The only difference is that there will be certain time delay when the output value changes from logic low to logic high. You can define the delay time by entering its value into the delay time text box in the setting area. After you complete the setting, click the" Apply" button. Then you can control the digital output value by the DO button and see its current value by the DO status LED display at the top of the module Display area.



5.7.2 High to Low Delay output

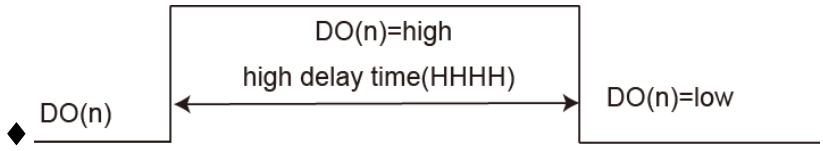
When you choose *High to Low* delay mode, it is almost the same as choosing the DO direct output mode. The only difference is that there will be certain time delay when the output value changes from logic high to logic low. You can define the delay time by entering its value into the Delay time text box in the Setting area. After you complete the setting, click the Apply button. Then you can control the digital output value by the DO button and see its current value by the DO status LED display at the top of the module Display area.



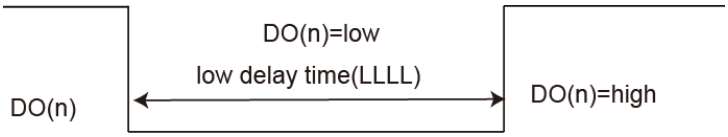
5.8 DO Auto-Off Time Mode

This function is used to force the specific DO channel to work as a monostable operation. After a certain period of time, the DO returns to the stable state until another triggering command is applied.

♦ **Low to High DO(n) pulse output**



♦ **High to Low DO(n) pulse output**



♦ **ASCII command:**

1. $AACONNDD ;Set DO(n) to DO Auto-Off Time Mode
2. $AA9DNNHHHHLLLL ;Set high /low delay width of DO(n)
3. #AA1NDD ; Write DO channel to active

Example: Set the channel-0 of DO to active and the channel-0 of D/O will be auto-off(inactive) after 3 sec.

1. $01CO0006 - Se to DO(0) Auto-Off Time Mode for DO "Low->High->Low.
   (output low is active (ON), outputt high/open is inactive)
2. $019D0017700001 - Set DO High delay time(HHHH=3000ms) and Low delay time(LLLL= 0.5ms).
   (For " Low->High->Low Auto-Off Time mode" LLLL always "0001")
3. #011001 - Write DO channel(0) to active
   wait DO auto-off time (3sec)…….....,
   ...
4. the DO channel(0) auto-off from active to inactive.

Example: Set the channel-1 of DO to inactive and the channel-1 of D/O will be auto-off(active) after 3 sec.

1. $01CO0107 - Se to DO(1) Auto-Off Time Mode for DO "High->Low->High.
   (output low is active (ON), outputt high/open is inactive)
2. $019D0100011770 - Set DO High delay time(HHHH=0.5ms) and Low delay time(LLLL= 3000ms).
   (For " High->Low->High Auto-Off Time mode" HHHH always "0001")
3. #011100 - Write DO channel(1) to inactive
   wait DO auto-off time (3sec)…….....,
   ...
4. the DO channel(1) auto-off from inactive to active.

♦ **Modbus rtu command:**

1. X+1453 ~ X+1484 ;Set DO(0~31) to low to high delay mode(=2) or high to low delay mode(=3)
2. X+1711 ~ X+1774 ;Set high/low delay time of the specific DO(n)
3. X+1775 ~ X+1806 ;Start *auto-off time Mode* operation

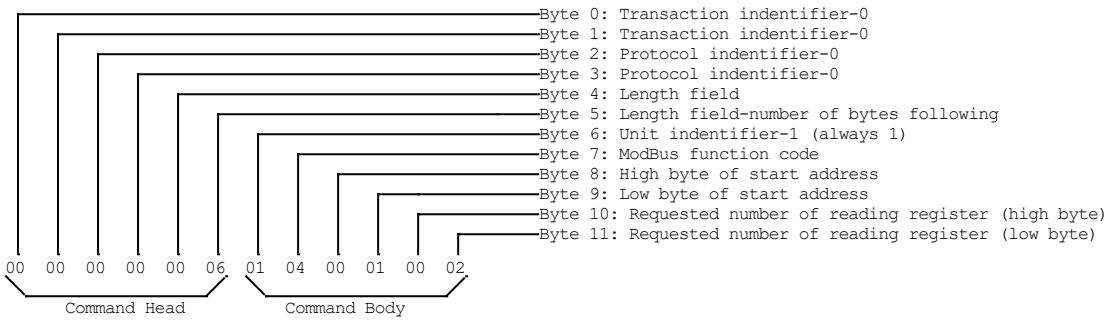
Chapter 6 Modbus Command Structure

EX-9200 system accepts a command/response form with the host computer. When systems are not transmitting they are in listen mode. The host issues a command to a system with a specified address and waits a certain amount of time for the system to respond. If no response arrives, a time-out aborts the sequence and returns control to the host. This chapter explains the structure of the commands with Modbus/TCP protocol, and guides to use these command sets to implement user's programs.

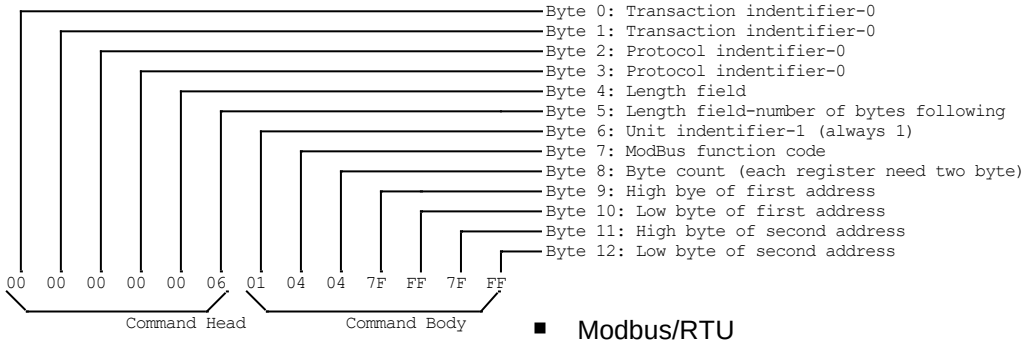
6.1 Command Structure

- Modbus/TCP

It is important to understand the encapsulation of a Modbus request or response carried on the Modbus/TCP network. A complete command is consisted of ***command head*** **and** ***command body*****.** The command head is prefixed by six bytes and responded to pack Modbus format; the command body defines target device and requested action. Following example will help you to realize this structure quickly.
Example：
If you want to read the first two values of Ex-9219 (address：40001~40002) with Modbus/TCP protocol, the request command should be：



And the response should be：



- Modbus/RTU

A Modbus request or response carried on the Modbus/RTU network. A complete command is consisted of ***command body only***. If you want to read the values of Ex-9219 with Modbus/RTU protocol, the request command is the same as Modbus/TCP, but ***without Command Head and first byte of Command body should be filled with module address***

6.2 Modbus Function Code Introductions

Code (Hex)	Name	Usage
01	Read Coil Status	Read Discrete DI/DO Bit
02	Read Input Status	Read Discrete DI/DO Bit
03	Read Holding Registers	Read 16-bit register.
04	Read Input Registers	
05	Write Single Coil	Write data to force coil ON/OFF
06	Write Single Register	Write data in 16-bit integer format
0F	Force Multiple Coils	Write multiple data to force coil ON/OFF
10	Preset Multiple Registers	Write multiple data in 16-bit integer format

Chapter 7 Modbus Address Mapping

7.1 Modbus Mapping Of EX-9215

7.1.1 Register Address (Unit：16 bits)

This register address mapping support Modbus function 03(0x03), 04(0x04), 06(0x06), and 16(0x10)

Where：N=30000 for Function 04(0x04)
N=40000 for Function 03(0x03), Function 06(0x06), and Function 16(0x10)

Address(dec)	Channel	Item	Attribute
N+0290~N+0290	0~15	Analog input burnout status	
N+0291~N+0290	0~15	Analog input high alarm status	R
N+0292~N+0292	0~15	Analog input low alarm status	R
N+0294~N+0309	0~15	Analog input normal value	R (see sec. 11.2)
N+0310	Average	Analog input average value	R
N+0312~N+0327	0~15	Analog input maximum value	R (see sec. 11.2)
N+0312~N+0327	0~15	Analog input maximum value	R (see sec. 11.2)
N+0330~N+0345	0~15	Analog input minimum value	R
N+0348~N+0363	0~15	Analog Input type (0x0007~0x000E)	R/W
N+0364	Average	Average type (0x0007~0x000E)	R/W

7.1.2 Bit Address (Unit：1 bit)

This discrete address mapping support Modbus function 010x01), 02(0x02), 05(0x05), and 15(0x0F)

Where：N=00000 for Function 00(0x00), and 15(0x0F)
N=10000 for Function 01(0x01), Function 05(0x05)

Address(dec)	Channel	Item	Attribute
X+0256~N+0271	0~15	Enable/disable analog channel	R/W
N+0272~N+0287	0~15	Analog input high alarm status	R
N+0288~N+0303	0~15	Analog input low alarm status	R
N+0304~N+0319	0~15	Enable/disable analog channel in average	R/W
N+0320~X+0335	0~15	Reset analog input maximum value	R/W
N+0336~N+0351	0~15	Reset analog input minimum value	R/W
N+0352~N+0367	0~15	Clear analog input high alarm status	R/W
N+0368~N+0383	0~15	Clear analog input low alarm status	R/W
N+0384~N+0399	0~15	Read AD burnout status	R
N+0404		Enable/disable burnout detection (0=disable,1=enable)	R/W
N+0407		Enable/disable DHCP (0=disable,1=enable)	R/W
N+0408		Enable/disable Web Server (0=disnable,1=enable)	R/W
N+0409		Enable/disable CRC/Checksum (0=disable,1=enable)	R/W

Note：

In Modbus PDU each data is addressed is numbered from 1 to n

In the Modbus data model each element within a data block is numbered from 1 to n

7.2 Modbus Mapping Of EX-9217

7.2.1 Register Address (Unit：16 bits)

This register address mapping support Modbus function 03(0x03), 04(0x04), 06(0x06), and 16(0x10)

Where：N=30000 for Function 04(0x04)
N=40000 for Function 03(0x03), Function 06(0x06), and Function 16(0x10)

Address(dec)	Channel	Item	Attribute
N+0000~N+0000	0~1	Digital input data (0x0000~0x0003)	R
N+0002~N+0002	0	Digital input latch status (0x0000~0x0003)	R/W
N+0004~N+0007	0~1	Digital input counter value (2 words/channel)	R (see sec.11.1)
N+0068~N+0068	0~15	Digital output status DO0~DO15 (0x0000~0x0001)	R/W
N+0069~N+0069	16~31	Digital output status DO16~DO31 (0x0000~0x0001)	R/W
N+0080~N+0081	0~1	Digital input mode	R/W
N+0112~N+0113	0~1	Digital input debounce time interval (0~0xffff)	R/W
X+0176~N+0176	0	Digital output pulse low width (0000~0xFFFF in 0.5msec)	R/W
N+0208~N+0208	0	Digital output pulse high width (0000~0xFFFF in 0.5msec)	R/W
N+0240~N+0240	0	Digital output pulse counts	R/W
N+0272~N+0272	0~15	Digital power-on value DO0~DO15 (0x0000~0xFFFF)	R/W
N+0273~N+0273	16~31	Digital power-on value DO16~DO31 (0x0000~0xFFFF)	R/W
N+0290~N+0290	0~15	Analog input burnout status	
N+0291~N+0290	0~15	Analog input high alarm status	R
N+0292~N+0292	0~15	Analog input low alarm status	R
N+0293~N+0293	Cold junction	Cold junction temperature(in 0.1C)	R
N+0294~N+0309	0~15	Analog input normal value	R (see sec. 11.2)
N+0310	Average	Analog input average value	R
N+0312~N+0327	0~15	Analog input maximum value	R (see sec.11.2)
N+0312~N+0327	0~15	Analog input maximum value	R (see sec. 11.2)
N+0330~N+0345	0~15	Analog input minimum value	R
N+0348~N+0363	0~15	Analog Input type (0x0007~0x000E)	R/W
N+0364	Average	Average type (0x0007~0x000E)	R/W

7.2.2 Bit Address (Unit: 1 bit)

This discrete address mapping support Modbus function 010x01), 02(0x02), 05(0x05), and 15(0x0F)

Where：N=00000 for Function 00(0x00), and 15(0x0F)
N=10000 for Function 01(0x01), Function 05(0x05)

Address(dec)	Channel	Item	Attribute
N+0000~N+0001	0~1	DI status (0X0000~0X0003)	R
N+0032~N+0033	0~1	DI latch status (0X0000~0X0003)	R
N+0064~N+0064	0	DO status	R/W
N+0096~N+0097	0~1	Clear DI latch status	R/W
N+0128~N+0129	0~1	Clear DI counter value	R/W
N+0160~N+0161	0~1	Enable/disable DI latch interrupt/Event) 0=disable, no generate interrupt or event 1=enable, generate interrupt or event (for USB/Ethernet connections only)	R/W
N+0224~N+0224	0~1	Start/Stop DO pulse output 0=disable DO pulse output 1=enable DO pulse out until Digital output pulse counts reaches zero (see * in Register address table)	R/W
X+0256~N+0271	0~15	Enable/disable analog channel	R/W
N+0272~N+0287	0~15	Analog input high alarm status	R
N+0288~N+0303	0~15	Analog input low alarm status	R
N+0304~N+0319	0~15	Enable/disable analog channel in average	R/W
N+0320~X+0335	0~15	Reset analog input maximum value	R/W
N+0336~N+0351	0~15	Reset analog input minimum value	R/W
N+0352~N+0367	0~15	Clear analog input high alarm status	R/W
N+0368~N+0383	0~15	Clear analog input low alarm status	R/W
N+0400		Save current DO as power on value	R/W
N+0405		Set DI active state (0=Open active,1=low active)	R/W
N+0406		Set DO active state (0=low active,1=open active)	R/W
N+0407		Enable/disable DHCP (0=disable,1=enable)	R/W
N+0408		Enable/disable Web Server (0=disnable,1=enable)	R/W
N+0409		Enable/disable CRC/Checksum (0=disable,1=enable)	R/W

Note：

In Modbus PDU each data is addressed is numbered from 1 to n

In the Modbus data model each element within a data block is numbered from 1 to n

7.3 Modbus Mapping Of EX-9219

7.3.1 Register Address (Unit: 16 bits)

This register address mapping support Modbus function 03(0x03), 04(0x04), 06(0x06), and 16(0x10)

Where：N=30000 for Function 04(0x04)
N=40000 for Function 03(0x03), Function 06(0x06), and Function 16(0x10)

Address(dec)	Channel	Item	Attribute
N+0000~N+0000	0~1	Digital input data (0x0000~0x0003)	R
N+0002~N+0002	0	Digital input latch status (0x0000~0x0003)	R/W
N+0004~N+0007	0~1	Digital input counter value (2 words/channel)	R (see sec.11.1)
N+0068~N+0068	0~15	Digital output status DO0~DO15 (0x0000~0x0001)	R/W
N+0069~N+0069	16~31	Digital output status DO16~DO31 (0x0000~0x0001)	R/W
N+0080~N+0081	0~1	Digital input mode	R/W
N+0112~N+0113	0~1	Digital input debounce time interval (0~0xffff)	R/W
X+0176~N+0176	0	Digital output pulse low width (0000~0xFFFF in 0.5msec)	R/W
N+0208~N+0208	0	Digital output pulse high width (0000~0xFFFF in 0.5msec)	R/W
N+0240~N+0240	0	Digital output pulse counts	R/W
N+0272~N+0272	0~15	Digital power-on value DO0~DO15 (0x0000~0xFFFF)	R/W
N+0273~N+0273	16~31	Digital power-on value DO16~DO31 (0x0000~0xFFFF)	R/W
N+0290~N+0290	0~15	Analog input burnout status	
N+0291~N+0290	0~15	Analog input high alarm status	R
N+0292~N+0292	0~15	Analog input low alarm status	R
N+0293~N+0293	Cold junction	Cold junction temperature(in 0.1C)	R
N+0294~N+0309	0~15	Analog input normal value	R (see sec. 11.2)
N+0310	Average	Analog input average value	R
N+0312~N+0327	0~15	Analog input maximum value	R (see sec. 11.2)
N+0312~N+0327	0~15	Analog input maximum value	R (see sec. 11.2)
N+0330~N+0345	0~15	Analog input minimum value	R
N+0348~N+0363	0~15	Analog Input type (0x0007~0x000E)	R/W
N+0364	Average	Average type (0x0007~0x000E)	R/W
N+366~N+381	0~15	AD channel cold junction offset (in 0.01C)	R/W

7.3.2 Bit Address (Unit: 1 bit)

This discrete address mapping support Modbus function 010x01), 02(0x02), 05(0x05), and 15(0x0F)

Where：N=00000 for Function 00(0x00), and 15(0x0F)
N=10000 for Function 01(0x01), Function 05(0x05)

Address(dec)	Channel	Item	Attribute
N+0000~N+0001	0~1	DI status (0X0000~0X0003)	R
N+0032~N+0033	0~1	DI latch status (0X0000~0X0003)	R
N+0064~N+0064	0	DO status	R/W
N+0096~N+0097	0~1	Clear DI latch status	R/W
N+0128~N+0129	0~1	Clear DI counter value	R/W
N+0160~N+0161	0~1	Enable/disable DI latch interrupt/Event) 0=disable, no generate interrupt or event 1=enable, generate interrupt or event (for USB/Ethernet connections only)	R/W
N+0224~N+0224	0~1	Start/Stop DO pulse output 0=disable DO pulse output 1=enable DO pulse out until Digital output pulse counts reaches zero (see * in Register address table)	R/W
X+0256~N+0271	0~15	Enable/disable analog channel	R/W
N+0272~N+0287	0~15	Analog input high alarm status	R
N+0288~N+0303	0~15	Analog input low alarm status	R
N+0304~N+0319	0~15	Enable/disable analog channel in average	R/W
N+0320~X+0335	0~15	Reset analog input maximum value	R/W
N+0336~N+0351	0~15	Reset analog input minimum value	R/W
N+0352~N+0367	0~15	Clear analog input high alarm status	R/W
N+0368~N+0383	0~15	Clear analog input low alarm status	R/W
N+0384~N+0399	0~15	Read AD burnout status	R
N+0400		Save current DO as power on value	R/W
N+0404		Enable/disable burnout detection (0=disable,1=enable)	R/W
N+0405		Set DI active state (0=Open active,1=low active)	R/W
N+0406		Set DO active state (0=low active,1=open active)	R/W
N+0407		Enable/disable DHCP (0=disable,1=enable)	R/W
N+0408		Enable/disable Web Server (0=disnable,1=enable)	R/W
N+0409		Enable/disable CRC/Checksum (0=disable,1=enable)	R/W

Note：

In Modbus PDU each data is addressed is numbered from 1 to n

In the Modbus data model each element within a data block is numbered from 1 to n

Chapter 8 Modbus Data Conversion

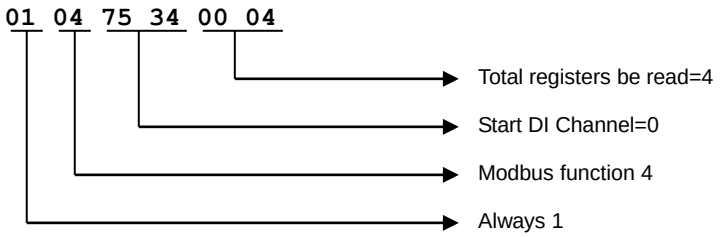
This chapter shows you how to convert Modbus register data to actual analog and digital value

8.1 How To Calculate DI Counter Value

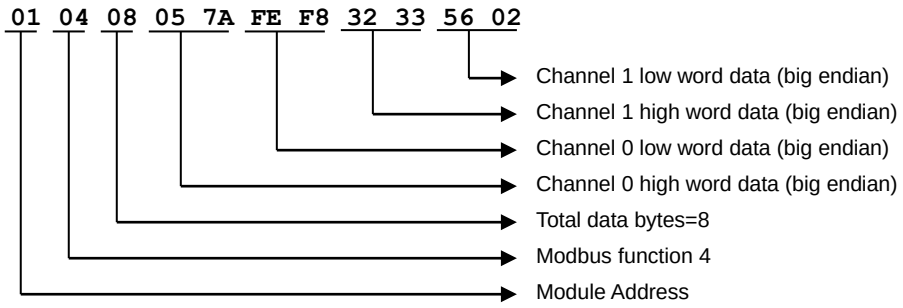
Formula：
Actual DI Channel counts = (register value (high word) <<16) + register value (low word)
Example 1 written with C：

1、 Assume the type of DI Channel 0 and channel 1 function as counter/frequency mode

2、 Send Request command as：(Note：0x7534=30004 start address of counter value of DI channel 0)



3、 Receive Response from module as：




```
Char Resp_data [];//Modbus response data received from Module
//where     Resp_data[6]=01    ; module address
//          Resp_data[7]=04    ; Modbus function 4
//          Resp_data[8]=08    ; total data bytes
//          Resp_data[9],[10]= 0x05,0x7A ;high data of channel 0
//          Resp_data[11],[12]= 0xFE,0xF8 ;low data of channel 0
//          Resp_data[13],[14]= 0x32,0x33 ;high data of channel 1
//          Resp_data[15],[16]= 0x56,0x02 ;low data of channel 1
long Chan0_Counts, Chan1_Counts;
Chan0_Counts =((long)Resp_data[9]<<24) | ((long)Resp_data[10]<<16) |
((long)Resp_data[11]<<8) | Resp_data[12];
Chan1_Counts =((long)Resp_data[13]<<24) | ((long)Resp_data[14]<<16) |
((long)Resp_data[15]<<8) | Resp_data[16];
printf ("\n\rChan 0 Counts=%d", Chan0_Counts);
printf ("\n\rChan 1 Counts=%d", Chan1_Counts);
```

1. Result：

Chan 0 Counts =91946744
Chan 1 Counts =892225154

8.2 How To Convert Modbus Data To AI Voltage/Temperature

8.2.1 Engineering Data Format Table

Type	Input Type	Min.	Max.	Formula	Unit
07	-10V ~ +10V	-10000	+10000	Volt=(MODBUS data) /1000	V
08	-5V ~ + 5V	-5000	+5000	Volt=(MODBUS data) /1000	
09	-2.5V ~ +2.5V	-2500	+2500	Volt=(MODBUS data) /1000	
0A	-1V ~ +1V	-10000	+10000	Volt=(MODBUS data) /10000	
0B	-500 mV ~ +500 mV	-5000	+5000	Volt=(MODBUS data) /10	mV
0C	-150m V ~ +150mV	-15000	+15000	Volt=(MODBUS data) /100	
0D	0 mA ~ +20 mA	00000	+20000	Current=(MODBUS data) /1000	mA
0E	4-20mA	4000	+20000	Current=(MODBUS data) /1000	
0F	Type J T/C (-100°C to 760°C)	-1000	7600	Temperature=(MODBUS data) /10	°C
10	Type K T/C (-100°C to 1370°C)	-1000	13700		
11	Type T T/C (-100°C to 400°C)	-1000	4000		
12	Type E T/C (-100°C to 1000°C)	-1000	10000		
13	Type R T/C (-50°C to 1750°C)	-500	17500		
14	Type S T/C (-50°C to 1750°C)	-500	17500		
15	Type B T/C (00°C to 1800°C)	0	18000		
20	IEC Pt100 (-50C~ 150C)	-500	1500		
21	IEC Pt100 (0C ~ 100C)	0	1000		
22	IEC Pt100 (0C ~ 200C)	0	2000		
23	IEC Pt100 (0C ~ 400C)	0	4000		
24	IEC Pt100 (-200C ~ 200C)	-2000	2000		
25	JIS Pt100 (-50C ~ 150C)	-500	1500		
26	JIS Pt100 (0C ~ 100C)	0	1000		
27	JIS Pt100 (0C~ 200C)	0	2000		
28	JIS Pt100 (0C ~ 400C)	0	4000		
29	JIS Pt100 (-200C ~ 200C)	-2000	2000		
2A	PT1000 (-40C ~ 160C)	-400	1600		
B	BALCO500 (-30C ~ 120C)	-300	1200		
C	Ni604 (-80C ~ 100C)	-800	1000		
D	Ni604 (0C~ 100C)	0	1000		

Example：Assume type of channel 2 is +/-10V and MODBUS data=0x2030(Hex)=8240(Dec)
The voltage of channel 2 is 8240/1000=8.24V
Example：Assume type of channel 1 is +/-500mV and MODBUS data=0xEF1B(Hex)=-4325(Dec)
The voltage of channel 2 is -9235/10=923.5mV
Example：Assume type of channel 1 is 0~20mA and MODBUS data=0x3B84(Hex)=15236(Dec)
The current of channel 2 is 15236/1000=15.236mA
Example：Assume type of channel 2 is Type K T/C (-100°C to 1370°C) and Modbus data=0x2030(Hex)
=8240(Dec)
The temperature of channel 2 is 8240/10=824.0 °C
Example：Assume type of channel 2 is IEC Pt100 (0C ~ 200C) and Modbus data=0x05DC(Hex)=1500(Dec)
The temperature of channel 2 is (1500/10=150 °C

8.2.2 Hex 2's Complement Data Format Table

Type	Input Type	Min	Max.	Formula	Unit
07	-10V ~ +10V	8000	7FFF	Volt=(MODBUS data *10)/32767	V
08	-5V ~ + 5V	8000	7FFF	Volt=(MODBUS data *5)/32767	
09	-2.5V ~ +2.5V	8000	7FFF	Volt=(MODBUS data *2.5)/32767	
0A	-1V ~ +1V	8000	7FFF	Volt=(MODBUS data *1)/32767	
0B	-500 mV ~ +500 mV	8000	7FFF	Volt=(MODBUS data *500)/32767	mV
0C	-150m V ~ +150mV	8000	7FFF	Volt=(MODBUS data *150)/32767	
0D	0 mA ~ +20 mA	8000	7FFF	Current=(MODBUS data *20)/32767	mA
0E	4-20mA	E667	7FFF	Current=(MODBUS data *20)/32767	
0F	Type J T/C (-100°C to 760°C)	EF28	7FFF	Temperature=(MODBUS data) /10	°C
10	Type K T/C (-100°C to 1370°C)	F6A8	7FFF	Temp.=(Modbus data*1370) /32767	
11	Type T T/C (-100°C to 400°C)	E000	7FFF	Temp.=(Modbus data*400) /32767	
12	Type E T/C (-100°C to 1000°C)	F333	7FFF	Temp.=(Modbus data*1000) /32767	
13	Type R T/C (-50°C to 1750°C)	FC57	7FFF	Temp.=(Modbus data*1750) /32767	
14	Type S T/C (-50°C to 1750°C)	FC57	7FFF	Temp.=(Modbus data*1750) /32767	
15	Type B T/C (00°C to 1800°C)	0	7FFF	Temp.=(Modbus data*1800) /32767	
20	IEC Pt100 (-50C~ 150C)	D555	7FFF	Temp.=(Modbus data*150) /32767	
21	IEC Pt100 (0C ~ 100C)	0	7FFF	Temp.=(Modbus data*100) /32767	
22	IEC Pt100 (0C ~ 200C)	0	7FFF	Temp.=(Modbus data*200) /32767	
23	IEC Pt100 (0C ~ 400C)	0	7FFF	Temp.=(Modbus data*400) /32767	
24	IEC Pt100 (-200C ~ 200C)	8000	7FFF	Temp.=(Modbus data*200) /32767	
25	JIS Pt100 (-50C ~ 150C)	D555	7FFF	Temp.=(Modbus data*150) /32767	
26	JIS Pt100 (0C ~ 100C)	0	7FFF	Temp.=(Modbus data*100) /32767	
27	JIS Pt100 (0C~ 200C)	0	7FFF	Temp.=(Modbus data*200) /32767	
28	JIS Pt100 (0C ~ 400C)	0	7FFF	Temp.=(Modbus data*400) /32767	
29	JIS Pt100 (-200C ~ 200C)	8000	7FFF	Temp.=(Modbus data*200) /32767	
2A	PT1000 (-40C ~ 160C)	E000	7FFF	Temp.=(Modbus data*160) /32767	
2B	BALCO500 (-30C ~ 120C)	E000	7FFF	Temp.=(Modbus data*120) /32767	
2C	Ni604 (-80C ~ 100C)	9999	7FFF	Temp.=(Modbus data*100) /32767	
2D	Ni604 (0C~ 100C)	0	1000	Temp.=(Modbus data*100) /32767	

Example：Assume type of channel 2 is +/-10V and MODBUS data=0x2030(Hex)=8240(Dec)
The voltage of channel 2 is (8240*10)/32767=2.514V
Example：Assume type of channel 1 is +/-500mV and MODBUS data=0xEF1B(Hex)=-4325(Dec)
The voltage of channel 2 is (-9235*500)/32767=-64.622mV
Example：Assume type of channel 1 is 0~20mA and MODBUS data=0x3B84(Hex)=15236(Dec)
The current of channel 2 is (15236*20)/32767=9.299mA
Example：Assume type of channel 2 is Type K T/C (-100°C to 1370°C) and Modbus data=0x2030(Hex)
=8240(Dec)
The temperature of channel 2 is (8240*1370)/32767=344.51 °C
Example：Assume type of channel 1 is IEC Pt100 (-200C ~ 200C) and Modbus
data=0xC001(Hex)=-16383(Dec)
The TEMPERATURE of channel 2 is (-16383*200)/32767=-99.996 °C

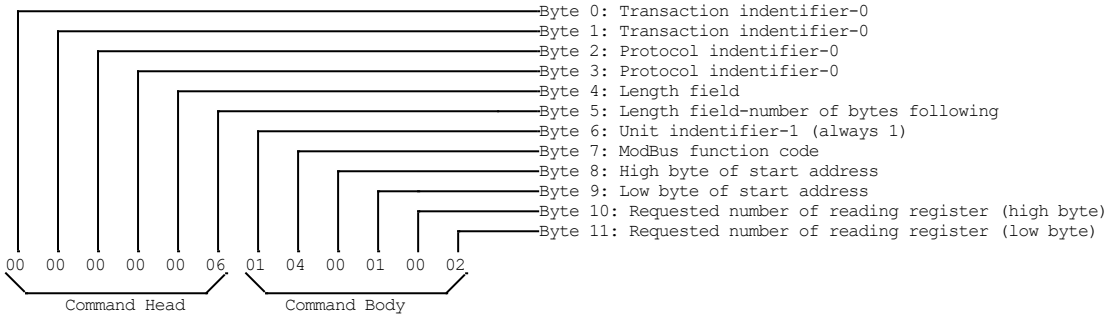
Chapter 9 Modbus Command Structure

EX-9200 system accepts a command/response form with the host computer. When systems are not transmitting they are in listen mode. The host issues a command to a system with a specified address and waits a certain amount of time for the system to respond. If no response arrives, a time-out aborts the sequence and returns control to the host. This chapter explains the structure of the commands with Modbus/TCP protocol, and guides to use these command sets to implement user's programs.

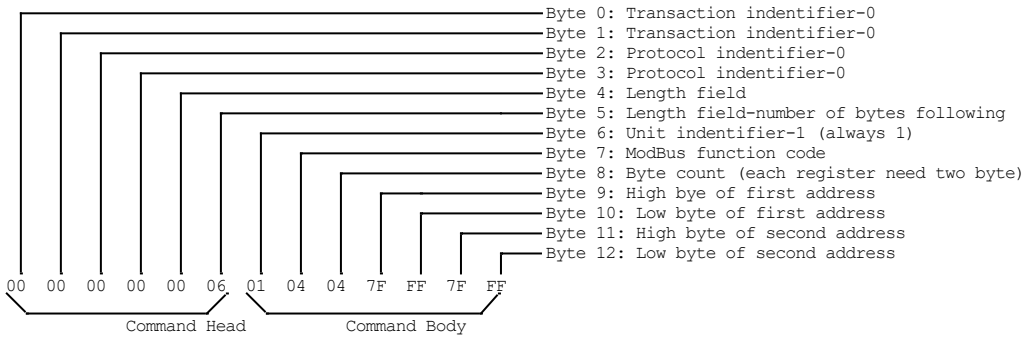
9.1 Command Structure

- Modbus/TCP

It is important to understand the encapsulation of a Modbus request or response carried on the Modbus/TCP network. A complete command is consisted of ***command head*** **and** ***command body*****.** The command head is prefixed by six bytes and responded to pack Modbus format; the command body defines target device and requested action. Following example will help you to realize this structure quickly.
Example：
If you want to read the first two values of Ex-9219 (address：40001~40002) with Modbus/TCP protocol, the request command should be：



And the response should be：



- Modbus/RTU

A Modbus request or response carried on the Modbus/RTU network. A complete command is consisted of ***command body only***. If you want to read the values of Ex-9219 with Modbus/RTU protocol, the request command is the same as Modbus/TCP, but ***without Command Head and first byte of Command body should be filled with module address***

9.2 Modbus Function Code Introductions

Code (Hex)	Name	Usage
01	Read Coil Status	Read Discrete DI/DO Bit
02	Read Input Status	Read Discrete DI/DO Bit
03	Read Holding Registers	Read 16-bit register.
04	Read Input Registers	
05	Write Single Coil	Write data to force coil ON/OFF
06	Write Single Register	Write data in 16-bit integer format
0F	Force Multiple Coils	Write multiple data to force coil ON/OFF
10	Preset Multiple Registers	Write multiple data in 16-bit integer format

Chapter 10 Modbus Address Mapping

10.1 Modbus Mapping Of EX-9215

10.1.1 Register Address (Unit：16 bits)

This register address mapping support Modbus function 03(0x03), 04(0x04), 06(0x06), and 16(0x10)

Where：N=30000 for Function 04(0x04)
N=40000 for Function 03(0x03), Function 06(0x06), and Function 16(0x10)

Address(dec)	Channel	Item	Attribute
N+0290~N+0290	0~15	Analog input burnout status	
N+0291~N+0290	0~15	Analog input high alarm status	R
N+0292~N+0292	0~15	Analog input low alarm status	R
N+0294~N+0309	0~15	Analog input normal value	R (see sec. 11.2)
N+0310	Average	Analog input average value	R
N+0312~N+0327	0~15	Analog input maximum value	R (see sec. 11.2)
N+0312~N+0327	0~15	Analog input maximum value	R (see sec. 11.2)
N+0330~N+0345	0~15	Analog input minimum value	R
N+0348~N+0363	0~15	Analog Input type (0x0007~0x000E)	R/W
N+0364	Average	Average type (0x0007~0x000E)	R/W

10.1.2 Bit Address (Unit：1 bit)

This discrete address mapping support Modbus function 010x01), 02(0x02), 05(0x05), and 15(0x0F)

Where：N=00000 for Function 00(0x00), and 15(0x0F)
N=10000 for Function 01(0x01), Function 05(0x05)

Address(dec)	Channel	Item	Attribute
X+0256~N+0271	0~15	Enable/disable analog channel	R/W
N+0272~N+0287	0~15	Analog input high alarm status	R
N+0288~N+0303	0~15	Analog input low alarm status	R
N+0304~N+0319	0~15	Enable/disable analog channel in average	R/W
N+0320~X+0335	0~15	Reset analog input maximum value	R/W
N+0336~N+0351	0~15	Reset analog input minimum value	R/W
N+0352~N+0367	0~15	Clear analog input high alarm status	R/W
N+0368~N+0383	0~15	Clear analog input low alarm status	R/W
N+0384~N+0399	0~15	Read AD burnout status	R
N+0404		Enable/disable burnout detection (0=disable,1=enable)	R/W
N+0407		Enable/disable DHCP (0=disable,1=enable)	R/W
N+0408		Enable/disable Web Server (0=disnable,1=enable)	R/W
N+0409		Enable/disable CRC/Checksum (0=disable,1=enable)	R/W

Note：

In Modbus PDU each data is addressed is numbered from 1 to n

In the Modbus data model each element within a data block is numbered from 1 to n

10.2 Modbus Mapping Of EX-9217

10.2.1 Register Address (Unit：16 bits)

This register address mapping support Modbus function 03(0x03), 04(0x04), 06(0x06), and 16(0x10)

Where：N=30000 for Function 04(0x04)
N=40000 for Function 03(0x03), Function 06(0x06), and Function 16(0x10)

Address(dec)	Channel	Item	Attribute
N+0000~N+0000	0~1	Digital input data (0x0000~0x0003)	R
N+0002~N+0002	0	Digital input latch status (0x0000~0x0003)	R/W
N+0004~N+0007	0~1	Digital input counter value (2 words/channel)	R (see sec.11.1)
N+0068~N+0068	0~15	Digital output status DO0~DO15 (0x0000~0x0001)	R/W
N+0069~N+0069	16~31	Digital output status DO16~DO31 (0x0000~0x0001)	R/W
N+0080~N+0081	0~1	Digital input mode	R/W
N+0112~N+0113	0~1	Digital input debounce time interval (0~0xffff)	R/W
X+0176~N+0176	0	Digital output pulse low width (0000~0xFFFF in 0.5msec)	R/W
N+0208~N+0208	0	Digital output pulse high width (0000~0xFFFF in 0.5msec)	R/W
N+0240~N+0240	0	Digital output pulse counts	R/W
N+0272~N+0272	0~15	Digital power-on value DO0~DO15 (0x0000~0xFFFF)	R/W
N+0273~N+0273	16~31	Digital power-on value DO16~DO31 (0x0000~0xFFFF)	R/W
N+0290~N+0290	0~15	Analog input burnout status	
N+0291~N+0290	0~15	Analog input high alarm status	R
N+0292~N+0292	0~15	Analog input low alarm status	R
N+0293~N+0293	Cold junction	Cold junction temperature(in 0.1C)	R
N+0294~N+0309	0~15	Analog input normal value	R (see sec. 11.2)
N+0310	Average	Analog input average value	R
N+0312~N+0327	0~15	Analog input maximum value	R (see sec.11.2)
N+0312~N+0327	0~15	Analog input maximum value	R (see sec. 11.2)
N+0330~N+0345	0~15	Analog input minimum value	R
N+0348~N+0363	0~15	Analog Input type (0x0007~0x000E)	R/W
N+0364	Average	Average type (0x0007~0x000E)	R/W

10.2.2 Bit Address (Unit: 1 bit)

This discrete address mapping support Modbus function 010x01), 02(0x02), 05(0x05), and 15(0x0F)

Where：N=00000 for Function 00(0x00), and 15(0x0F)
N=10000 for Function 01(0x01), Function 05(0x05)

Address(dec)	Channel	Item	Attribute
N+0000~N+0001	0~1	DI status (0X0000~0X0003)	R
N+0032~N+0033	0~1	DI latch status (0X0000~0X0003)	R
N+0064~N+0064	0	DO status	R/W
N+0096~N+0097	0~1	Clear DI latch status	R/W
N+0128~N+0129	0~1	Clear DI counter value	R/W
N+0160~N+0161	0~1	Enable/disable DI latch interrupt/Event) 0=disable, no generate interrupt or event 1=enable, generate interrupt or event (for USB/Ethernet connections only)	R/W
N+0224~N+0224	0~1	Start/Stop DO pulse output 0=disable DO pulse output 1=enable DO pulse out until Digital output pulse counts reaches zero (see * in Register address table)	R/W
X+0256~N+0271	0~15	Enable/disable analog channel	R/W
N+0272~N+0287	0~15	Analog input high alarm status	R
N+0288~N+0303	0~15	Analog input low alarm status	R
N+0304~N+0319	0~15	Enable/disable analog channel in average	R/W
N+0320~X+0335	0~15	Reset analog input maximum value	R/W
N+0336~N+0351	0~15	Reset analog input minimum value	R/W
N+0352~N+0367	0~15	Clear analog input high alarm status	R/W
N+0368~N+0383	0~15	Clear analog input low alarm status	R/W
N+0400		Save current DO as power on value	R/W
N+0405		Set DI active state (0=Open active,1=low active)	R/W
N+0406		Set DO active state (0=low active,1=open active)	R/W
N+0407		Enable/disable DHCP (0=disable,1=enable)	R/W
N+0408		Enable/disable Web Server (0=disnable,1=enable)	R/W
N+0409		Enable/disable CRC/Checksum (0=disable,1=enable)	R/W

Note：

In Modbus PDU each data is addressed is numbered from 1 to n

In the Modbus data model each element within a data block is numbered from 1 to n

10.3Modbus Mapping Of EX-9219

10.3.1 Register Address (Unit: 16 bits)

This register address mapping support Modbus function 03(0x03), 04(0x04), 06(0x06), and 16(0x10)

Where：N=30000 for Function 04(0x04)
N=40000 for Function 03(0x03), Function 06(0x06), and Function 16(0x10)

Address(dec)	Channel	Item	Attribute
N+0000~N+0000	0~1	Digital input data (0x0000~0x0003)	R
N+0002~N+0002	0	Digital input latch status (0x0000~0x0003)	R/W
N+0004~N+0007	0~1	Digital input counter value (2 words/channel)	R (see sec.11.1)
N+0068~N+0068	0~15	Digital output status DO0~DO15 (0x0000~0x0001)	R/W
N+0069~N+0069	16~31	Digital output status DO16~DO31 (0x0000~0x0001)	R/W
N+0080~N+0081	0~1	Digital input mode	R/W
N+0112~N+0113	0~1	Digital input debounce time interval (0~0xffff)	R/W
X+0176~N+0176	0	Digital output pulse low width (0000~0xFFFF in 0.5msec)	R/W
N+0208~N+0208	0	Digital output pulse high width (0000~0xFFFF in 0.5msec)	R/W
N+0240~N+0240	0	Digital output pulse counts	R/W
N+0272~N+0272	0~15	Digital power-on value DO0~DO15 (0x0000~0xFFFF)	R/W
N+0273~N+0273	16~31	Digital power-on value DO16~DO31 (0x0000~0xFFFF)	R/W
N+0290~N+0290	0~15	Analog input burnout status	
N+0291~N+0290	0~15	Analog input high alarm status	R
N+0292~N+0292	0~15	Analog input low alarm status	R
N+0293~N+0293	Cold junction	Cold junction temperature(in 0.1C)	R
N+0294~N+0309	0~15	Analog input normal value	R (see sec. 11.2)
N+0310	Average	Analog input average value	R
N+0312~N+0327	0~15	Analog input maximum value	R (see sec. 11.2)
N+0312~N+0327	0~15	Analog input maximum value	R (see sec. 11.2)
N+0330~N+0345	0~15	Analog input minimum value	R
N+0348~N+0363	0~15	Analog Input type (0x0007~0x000E)	R/W
N+0364	Average	Average type (0x0007~0x000E)	R/W
N+366~N+381	0~15	AD channel cold junction offset (in 0.01C)	R/W

10.3.2 Bit Address (Unit: 1 bit)

This discrete address mapping support Modbus function 010x01), 02(0x02), 05(0x05), and 15(0x0F)

Where：N=00000 for Function 00(0x00), and 15(0x0F)
N=10000 for Function 01(0x01), Function 05(0x05)

Address(dec)	Channel	Item	Attribute
N+0000~N+0001	0~1	DI status (0X0000~0X0003)	R
N+0032~N+0033	0~1	DI latch status (0X0000~0X0003)	R
N+0064~N+0064	0	DO status	R/W
N+0096~N+0097	0~1	Clear DI latch status	R/W
N+0128~N+0129	0~1	Clear DI counter value	R/W
N+0160~N+0161	0~1	Enable/disable DI latch interrupt/Event) 0=disable, no generate interrupt or event 1=enable, generate interrupt or event (for USB/Ethernet connections only)	R/W
N+0224~N+0224	0~1	Start/Stop DO pulse output 0=disable DO pulse output 1=enable DO pulse out until Digital output pulse counts reaches zero (see * in Register address table)	R/W
X+0256~N+0271	0~15	Enable/disable analog channel	R/W
N+0272~N+0287	0~15	Analog input high alarm status	R
N+0288~N+0303	0~15	Analog input low alarm status	R
N+0304~N+0319	0~15	Enable/disable analog channel in average	R/W
N+0320~X+0335	0~15	Reset analog input maximum value	R/W
N+0336~N+0351	0~15	Reset analog input minimum value	R/W
N+0352~N+0367	0~15	Clear analog input high alarm status	R/W
N+0368~N+0383	0~15	Clear analog input low alarm status	R/W
N+0384~N+0399	0~15	Read AD burnout status	R
N+0400		Save current DO as power on value	R/W
N+0404		Enable/disable burnout detection (0=disable,1=enable)	R/W
N+0405		Set DI active state (0=Open active,1=low active)	R/W
N+0406		Set DO active state (0=low active,1=open active)	R/W
N+0407		Enable/disable DHCP (0=disable,1=enable)	R/W
N+0408		Enable/disable Web Server (0=disnable,1=enable)	R/W
N+0409		Enable/disable CRC/Checksum (0=disable,1=enable)	R/W

Note：

In Modbus PDU each data is addressed is numbered from 1 to n

In the Modbus data model each element within a data block is numbered from 1 to n

Chapter 11 Modbus Data Conversion

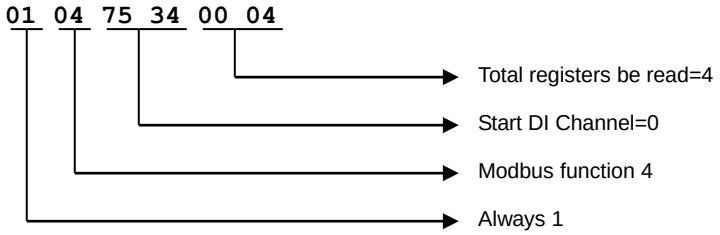
This chapter shows you how to convert Modbus register data to actual analog and digital value

11.1 How To Calculate DI Counter Value

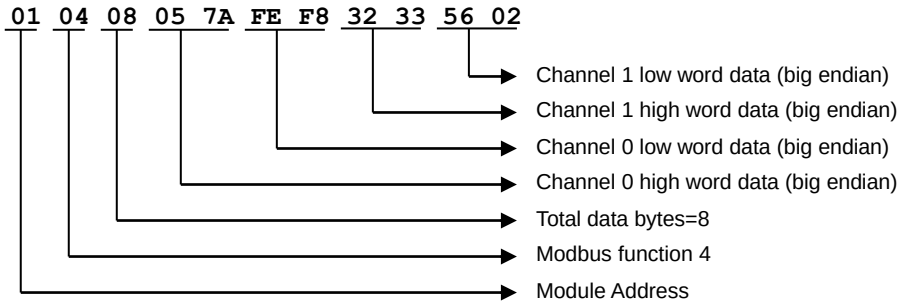
Formula：
Actual DI Channel counts = (register value (high word) <<16) + register value (low word)
Example 1 written with C：

4、 Assume the type of DI Channel 0 and channel 1 function as counter/frequency mode

5、 Send Request command as：(Note：0x7534=30004 start address of counter value of DI channel 0)



6、 Receive Response from module as：




```
Char Resp_data [];//Modbus response data received from Module
//where     Resp_data[6]=01    ; module address
//          Resp_data[7]=04    ; Modbus function 4
//          Resp_data[8]=08    ; total data bytes
//          Resp_data[9],[10]= 0x05,0x7A ;high data of channel 0
//          Resp_data[11],[12]= 0xFE,0xF8 ;low data of channel 0
//          Resp_data[13],[14]= 0x32,0x33 ;high data of channel 1
//          Resp_data[15],[16]= 0x56,0x02 ;low data of channel 1
long Chan0_Counts, Chan1_Counts;
Chan0_Counts =((long)Resp_data[9]<<24) | ((long)Resp_data[10]<<16) |
((long)Resp_data[11]<<8) | Resp_data[12];
Chan1_Counts =((long)Resp_data[13]<<24) | ((long)Resp_data[14]<<16) |
((long)Resp_data[15]<<8) | Resp_data[16];
printf ("\n\rChan 0 Counts=%d", Chan0_Counts);
printf ("\n\rChan 1 Counts=%d", Chan1_Counts);
```

2. Result：

Chan 0 Counts =91946744
Chan 1 Counts =892225154

11.2 How To Convert Modbus Data To AI Voltage/Temperature

11.2.1 Engineering Data Format Table

Type	Input Type	Min.	Max.	Formula	Unit
07	-10V ~ +10V	-10000	+10000	Volt=(MODBUS data) /1000	V
08	-5V ~ + 5V	-5000	+5000	Volt=(MODBUS data) /1000	
09	-2.5V ~ +2.5V	-2500	+2500	Volt=(MODBUS data) /1000	
0A	-1V ~ +1V	-10000	+10000	Volt=(MODBUS data) /10000	
0B	-500 mV ~ +500 mV	-5000	+5000	Volt=(MODBUS data) /10	mV
0C	-150m V ~ +150mV	-15000	+15000	Volt=(MODBUS data) /100	
0D	0 mA ~ +20 mA	00000	+20000	Current=(MODBUS data) /1000	mA
0E	4-20mA	4000	+20000	Current=(MODBUS data) /1000	
0F	Type J T/C (-100°C to 760°C)	-1000	7600	Temperature=(MODBUS data) /10	°C
10	Type K T/C (-100°C to 1370°C)	-1000	13700		
11	Type T T/C (-100°C to 400°C)	-1000	4000		
12	Type E T/C (-100°C to 1000°C)	-1000	10000		
13	Type R T/C (-50°C to 1750°C)	-500	17500		
14	Type S T/C (-50°C to 1750°C)	-500	17500		
15	Type B T/C (00°C to 1800°C)	0	18000		
20	IEC Pt100 (-50C~ 150C)	-500	1500		
21	IEC Pt100 (0C ~ 100C)	0	1000		
22	IEC Pt100 (0C ~ 200C)	0	2000		
23	IEC Pt100 (0C ~ 400C)	0	4000		
24	IEC Pt100 (-200C ~ 200C)	-2000	2000		
25	JIS Pt100 (-50C ~ 150C)	-500	1500		
26	JIS Pt100 (0C ~ 100C)	0	1000		
27	JIS Pt100 (0C~ 200C)	0	2000		
28	JIS Pt100 (0C ~ 400C)	0	4000		
29	JIS Pt100 (-200C ~ 200C)	-2000	2000		
2A	PT1000 (-40C ~ 160C)	-400	1600		
B	BALCO500 (-30C ~ 120C)	-300	1200		
C	Ni604 (-80C ~ 100C)	-800	1000		
D	Ni604 (0C~ 100C)	0	1000		

Example：Assume type of channel 2 is +/-10V and MODBUS data=0x2030(Hex)=8240(Dec)
The voltage of channel 2 is 8240/1000=8.24V
Example：Assume type of channel 1 is +/-500mV and MODBUS data=0xEF1B(Hex)=-4325(Dec)
The voltage of channel 2 is -9235/10=923.5mV
Example：Assume type of channel 1 is 0~20mA and MODBUS data=0x3B84(Hex)=15236(Dec)
The current of channel 2 is 15236/1000=15.236mA
Example：Assume type of channel 2 is Type K T/C (-100°C to 1370°C) and Modbus data=0x2030(Hex)
=8240(Dec)
The temperature of channel 2 is 8240/10=824.0 °C
Example：Assume type of channel 2 is IEC Pt100 (0C ~ 200C) and Modbus data=0x05DC(Hex)=1500(Dec)
The temperature of channel 2 is (1500/10=150 °C

11.2.2 Hex 2's Complement Data Format Table

Type	Input Type	Min	Max.	Formula	Unit
07	-10V ~ +10V	8000	7FFF	Volt=(MODBUS data *10)/32767	V
08	-5V ~ + 5V	8000	7FFF	Volt=(MODBUS data *5)/32767	
09	-2.5V ~ +2.5V	8000	7FFF	Volt=(MODBUS data *2.5)/32767	
0A	-1V ~ +1V	8000	7FFF	Volt=(MODBUS data *1)/32767	
0B	-500 mV ~ +500 mV	8000	7FFF	Volt=(MODBUS data *500)/32767	mV
0C	-150m V ~ +150mV	8000	7FFF	Volt=(MODBUS data *150)/32767	
0D	0 mA ~ +20 mA	8000	7FFF	Current=(MODBUS data *20)/32767	mA
0E	4-20mA	E667	7FFF	Current=(MODBUS data *20)/32767	
0F	Type J T/C (-100°C to 760°C)	EF28	7FFF	Temperature=(MODBUS data) /10	°C
10	Type K T/C (-100°C to 1370°C)	F6A8	7FFF	Temp.=(Modbus data*1370) /32767	
11	Type T T/C (-100°C to 400°C)	E000	7FFF	Temp.=(Modbus data*400) /32767	
12	Type E T/C (-100°C to 1000°C)	F333	7FFF	Temp.=(Modbus data*1000) /32767	
13	Type R T/C (-50°C to 1750°C)	FC57	7FFF	Temp.=(Modbus data*1750) /32767	
14	Type S T/C (-50°C to 1750°C)	FC57	7FFF	Temp.=(Modbus data*1750) /32767	
15	Type B T/C (00°C to 1800°C)	0	7FFF	Temp.=(Modbus data*1800) /32767	
20	IEC Pt100 (-50C~ 150C)	D555	7FFF	Temp.=(Modbus data*150) /32767	
21	IEC Pt100 (0C ~ 100C)	0	7FFF	Temp.=(Modbus data*100) /32767	
22	IEC Pt100 (0C ~ 200C)	0	7FFF	Temp.=(Modbus data*200) /32767	
23	IEC Pt100 (0C ~ 400C)	0	7FFF	Temp.=(Modbus data*400) /32767	
24	IEC Pt100 (-200C ~ 200C)	8000	7FFF	Temp.=(Modbus data*200) /32767	
25	JIS Pt100 (-50C ~ 150C)	D555	7FFF	Temp.=(Modbus data*150) /32767	
26	JIS Pt100 (0C ~ 100C)	0	7FFF	Temp.=(Modbus data*100) /32767	
27	JIS Pt100 (0C~ 200C)	0	7FFF	Temp.=(Modbus data*200) /32767	
28	JIS Pt100 (0C ~ 400C)	0	7FFF	Temp.=(Modbus data*400) /32767	
29	JIS Pt100 (-200C ~ 200C)	8000	7FFF	Temp.=(Modbus data*200) /32767	
2A	PT1000 (-40C ~ 160C)	E000	7FFF	Temp.=(Modbus data*160) /32767	
2B	BALCO500 (-30C ~ 120C)	E000	7FFF	Temp.=(Modbus data*120) /32767	
2C	Ni604 (-80C ~ 100C)	9999	7FFF	Temp.=(Modbus data*100) /32767	
2D	Ni604 (0C~ 100C)	0	1000	Temp.=(Modbus data*100) /32767	

Example：Assume type of channel 2 is +/-10V and MODBUS data=0x2030(Hex)=8240(Dec)
The voltage of channel 2 is (8240*10)/32767=2.514V
Example：Assume type of channel 1 is +/-500mV and MODBUS data=0xEF1B(Hex)=-4325(Dec)
The voltage of channel 2 is (-9235*500)/32767=-64.622mV
Example：Assume type of channel 1 is 0~20mA and MODBUS data=0x3B84(Hex)=15236(Dec)
The current of channel 2 is (15236*20)/32767=9.299mA
Example：Assume type of channel 2 is Type K T/C (-100°C to 1370°C) and Modbus data=0x2030(Hex)
=8240(Dec)
The temperature of channel 2 is (8240*1370)/32767=344.51 °C
Example：Assume type of channel 1 is IEC Pt100 (-200C ~ 200C) and Modbus
data=0xC001(Hex)=-16383(Dec)
The TEMPERATURE of channel 2 is (-16383*200)/32767=-99.996 °C

Chapter 12 Analog And Digital /I/O Channel Type

12.1 DI Channel Types

Code	Type,	Models
00	DI transparent	9217,9219
01	Counter	9217,9219
02	Low to high latch	9217,9219
03	High to low latch	9217,9219
04	Frequency	9217,9219

12.2 AI Channel Types

Code	Type and Range	Models
0x07	+/-10V	9217
0x08	+/-5V	9217
0x09	+/-2.5V	9217,9219
0x0A	+/-1V	9217,9219
0x0B	+/-500mV	9217,9219
0x0C	+/-150mV	9217,9219
0x0D	0-20mA (125 ohms)	9217,9219
0x0E	4-20mA (125 ohms)	9217,9219
0x0F	T/C J type (*-100C~760C*)	9219
0x10	T/C K type (*-100C~1370C*)	9219
0x11	T/C T type (*-100C~400C*)	9219
0x12	T/C E type (*-100C~1000C*)	9219
0x13	T/C R type (*-500C~1750C*)	9219
0x14	T/C S type (*-500C~1750C*)	9219
0x15	T/C B type (*0C~1800C*)	9219
0x20	RTD IEC Pt100 (-50C ~ 150C)	9215
0x21	RTD IEC Pt100 (0C ~ 100C)	9215
0x22	RTD IEC Pt100 (0C ~ 200C)	9215
0x23	RTD IEC Pt100 (0C ~ 400C)	9215
0x24	RTD IEC Pt100 (-200C ~ 200C)	9215
0x25	RTD JIS Pt100 (-50C ~ 150C)	9215
0x26	RTD JIS Pt100 (0C ~ 100C)	9215
0x27	RTD JIS Pt100 (0C ~ 200C)	9215
0x28	RTD JIS Pt100 (0C ~ 400C)	9215
0x29	RTD JIS Pt100 (-200C ~ 200C)	9215
0x2A	RTD Pt1000 (-40C ~ 160C)	9215
0x2B	RTD BALCO500 (-30C ~ 120C)	9215
0x2C	RTD Ni (-80C ~ 100C)	9215
0x2D	RTD Ni (0C ~ 100C)	9215

Chapter 13 TCP/IP Port Assignments

The following table shows you the TCP/IP ports used for EX-9200 series

Functions	Protocol	Port
Modbus/TCP protocol	TCP	502
ASCII command /Modbus RTU protocol	UDP	1025
Broadcast protocol	UDP	5048
Stream data	UDP	5148
Alarm Event data	UDP	5168
Httpd (web server)	TCP	80

Chapter 14 ASCII Commands

14.1 Common Commands

$AACRC	Read CRC Status	14.4.1
$AACRCv	Enabled/Disable CRC	14.4.2
$AA5	Reads The Reset Status	14.4.3
$AAF	Reads The Firmware Version	14.4.4
$AAGATE	Read Gateway Address	14.4.5
$AAGATEnnnnnnnn	Set Gateway Address	14.4.6
$AAIP	Read IP Address	14.4.7
$AAIPnnnnnnnn	Set IP Address	14.4.8
$AAM	Read Module Name	14.4.9
~AAO(name)	Set model name	14.4.10
$AAMASK	Read Mask Address	14.4.11
$AAMASKnnnnnnnn	Set Mask Address	14.4.12
$AAP	Read Communication Protocol	14.4.13
$AAP?	Set Communication Protocol	14.4.14
$AASW	Read Web Server Status	14.4.15
$AASWv	Enabled/Disable Web Server	14.4.16
$AADHCPv	Enabled/Disable DHCP	14.4.18
$AADHCP	Read DHCP Status	14.4.17
^AAMAC	Read MAC Address	14.4.19
~AA6v	Set LED control	14.4.20
~AA6ddddddddd	Write Data to LED board	14.4.21
~AA6ddddddddnnnn	Force LED to flash for specified times	14.4.22
~AA3eTTTTdddd	Write Communication Timeout settings (*)	14.4.23
~AA31	Read Communication Timeout Settings (*)	14.4.24

(*) 1. for all module with fimrware version 5.5 or later

14.2 Digital Commands

@AA	Reads the Digital I/O Status	14.4.57
@AAnn	Sets the Digital Output Channels(DO0~DO7)	14.4.58
@AAnnnn	Sets the Digital Output Channels(DO0~DO15)	14.4.59
@AAnnnnnn	Sets the Digital Output Channels(DO0~DO23)	14.4.60
#AA0Ann	Sets the Digital 1's byte(DO0~DO7) Output	14.4.61
#AA0Bnn	Sets the Digital 2's byte(DO8~DO15) Output	14.4.62
#AA0Cnn	Sets the Digital 3's byte(DO16~DO23) Output	14.4.63
#AAnn	Reads the Digital Input Counter	14.4.64
$AACn	Clears the Digital Input Counter	14.4.65
$AACnn	Clears the Digital Input Counter	14.4.66
$AALS	Reads the Latched DI Status	14.4.67
$AAC	Clears the Latched DI Status	14.4.68
$AA9nn	Read Single Do Pulse High/Low Width	14.4.69
$AA9nnhhhhllll	Set Single Do Pulse High/Low Width	14.4.70
$AAAnn	Read Single Do High/Low Delay Width	14.4.71
$AAAnnhhhhllll	Set Single Do High/Low Delay Width	14.4.72
$AABnn	Read Single Do Pulse Counts	14.4.73
#AA2nncccc	Write Single Do Pulse Counts	14.4.74
#AA3nns	Start/Stop Do Pulse Counts	14.4.75
~AA4v	Reads the Power On/Safe Value	14.4.77
~AA5v	Sets current DO value as Power On/Safe Value	14.4.78
~AA5vnnnnnn	Sets specified value as Power On/Safe Value	14.4.79
~AAD	Read DI/O active state	14.4.80
~AADvv	Set DI/O active state	14.4.81
~AASDBv	Set digital input debounce mode (**)	14.4.82
~AARDB	Read digital input debounce mode (**)	14.4.83

(**) 1. for the modules have digital input channels and with firmware version 5.5 or later

14.3 Analog Commands

#AA	Reads the Analog Inputs of All	14.4.25
#AAn	Reads the single Analog Input	14.4.26
#AAMH	Read Maximum Value Of All Channels	14.4.27
#AAMHn	Read Maximum Value Of Specified Channel	14.4.28
$AAMH	Clear All Maximum Value	14.4.29
$AAMHn	Clear Maximum Value Of Specified Channel	14.4.30
#AAML	Read Minimum Value Of All Channels	14.4.31
#AAMLn	Read Minimum Value Of Specified Channel	14.4.32
$AAML	Clear All Minimum Value	14.4.33
$AAMLn	Clear Minimum Value Of Specified Channel	14.4.34
#AAAV	Read Average Value	14.4.35
$AAE	Read Channel Average Enable/Disable Status	14.4.36
$AAEnnnn	Disable/Enable Channel in Average	14.4.37
#AAAL	Read AD high/low Alarm Status	14.4.38
$AAAHnnnn	Clear A/D High Alarm	14.4.39
$AAALnnnn	Clear A/D Low Alarm	14.4.40
$AAB	Reads Channel burnout Status	14.4.41
%AAB	Read channel burnout enable/disable status	14.4.42
%AABn	enable/disable channel burnout	14.4.43
$AA3	Reads the CJC Temperature	14.4.44
~AAC	Reads the CJC Enable/disable	14.4.45
~AACn	Enables/Disables the CJC	14.4.46
$AA9Snnnn	Sets the all channel CJC Offset	14.4.47
$AA9c	Read single channel CJC Offset	14.4.48
$AA9cSnnnn	Set single channel CJC Offset	14.4.49
$AAR	Read A/D Filter Value	14.4.50
$AARf	Set A/D Filter Value	14.4.51
$AA6	Reads the Channel Enable/Disable Status	14.4.52
$AA5vvvv	Enables/Disables A/D Channel	14.4.53
$AA8Ci	Reads the Single A/D Channel Range	14.4.54
$AA7CiRrr	Sets the Single Channel Range	14.4.55
$AAS1	Reloads the Default Calibration	14.4.56

14.4 Command Description

14.4.1 $AACRC Read CRC Status

Description The command will return the ASCII command check sum and modbus RTU CRC status.

Syntax $AACRC(cr)

is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of an analog input module.
CRC the ASCII protocol check sum and modbus RTU CRC status command
(cr) is the terminating character, carriage return (0Dh).

Response !AAv(cr) if the command is valid or ?AA (cr) if the command is invalid
There is no response if the module detects a syntax error or communication error.

AA (range 00-FF) represents the 2-character hexadecimal address of the analog input module.
! is a delimiter character.
v is the check sum/CRC status v=1 enable, v=0 disable.
(cr) is the terminating character, carriage return (0Dh).

14.4.2 $AACRCv Set CRC

Description The command will set the ASCII command check sum and modbus RTU CRC status.

Syntax $AACRCv(cr)

is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of an analog input module.
CRC the ASCII protocol check sum and modbus RTU CRC status command
v is the check sum/CRC status v=1 enable ,v=0 disable.
(cr) is the terminating character, carriage return (0Dh).

Response !AA(cr) if the command is valid or ?AA (cr) if the command is invalid
There is no response if the module detects a syntax error or communication error.

AA (range 00-FF) represents the 2-character hexadecimal address of the analog input module.
! is a delimiter character.
(cr) is the terminating character, carriage return (0Dh).

14.4.3 $AA5 Read The Reset status

Description The command will read the module reset status.

Syntax $AA5 (cr)

is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of an analog input module.
5 reads the module reset status command
(cr) is the terminating character, carriage return (0Dh).

Response !AAv(cr) if the command is valid or ?AA (cr) if the command is invalid
There is no response if the module detects a syntax error or communication error.

! is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of the analog input module.
V reset status, v=0 no reset, v=1 reset at last once
(cr) is the terminating character, carriage return (0Dh).
Note: Reset status will be set to 0 after reading

14.4.4 $AAF Read The Firmware Version

Description The command will read the module firmware version.

Syntax $AAF(cr)

is a delimiter character.

AA (range 00-FF) represents the 2-character hexadecimal address of an analog input module.

F read the module firmware version command

(cr) is the terminating character, carriage return (0Dh).

Response !AAnnnn(cr) if the command is valid or ?AA (cr) if the command is invalid

There is no response if the module detects a syntax error or communication error.

! is a delimiter character.

AA (range 00-FF) represents the 2-character hexadecimal address of the analog input module.

nnnn Firmware veion string (such as "M5.50")

(cr) is the terminating character, carriage return (0Dh).

14.4.5 $AAGATE Read Gateway Address

Description The command will read the module Gateway address.

Syntax $AAGATE(cr)

is a delimiter character.

AA (range 00-FF) represents the 2-character hexadecimal address of an analog input module.

GATE read the module Gateway address command

(cr) is the terminating character, carriage return (0Dh).

Response !AAnnnnnnnn(cr) if the command is valid or ?AA (cr) if the command is invalid

There is no response if the module detects a syntax error or communication error.

! is a delimiter character.

AA (range 00-FF) represents the 2-character hexadecimal address of the analog input module.

nnnnnnnn gateway address(8 digits) (such as C8A20001)

(cr) is the terminating character, carriage return (0Dh).

14.4.6 $AAGATEnnnnnnnn Set Gateway Address

Description The command will set the module Gateway address.

Syntax $AAGATEnnnnnnnn (cr)

is a delimiter character.

AA (range 00-FF) represents the 2-character hexadecimal address of an analog input module.

GATE set the module Gateway address command

nnnnnnnn gateway address(8 digits) (such as C0A80001 =192.168.0.1)

(cr) is the terminating character, carriage return (0Dh).

Response !AA (cr) if the command is valid or ?AA (cr) if the command is invalid

There is no response if the module detects a syntax error or communication error.

! is a delimiter character.

AA (range 00-FF) represents the 2-character hexadecimal address of the analog input module.

(cr) is the terminating character, carriage return (0Dh).

14.4.7 $AAIP Read IP Address

Description The command will set the module IP address.

Syntax $AAGATEnnnnnnnn (cr)

is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of an analog input module.
IP read the module IP address command
(cr) is the terminating character, carriage return (0Dh).

Response !AAnnnnnnnn (cr) if the command is valid or ?AA (cr) if the command is invalid
There is no response if the module detects a syntax error or communication error.
! is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of the analog input module.
nnnnnnnn IP address(8 digits) (such as C0A8000A =192.168.0.10)
(cr) is the terminating character, carriage return (0Dh).

14.4.8 $AAIPnnnnnnnn Set IP Address

Description The command will set the module IP address.

Syntax $AAGATEnnnnnnnn (cr)

is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of an analog input module.
IP set the module IP address command
Nnnnnnnn IP address(8 digits) (such as C0A80001 =192.168.0.1)
(cr) is the terminating character, carriage return (0Dh).

Response !AA (cr) if the command is valid or ?AA (cr) if the command is invalid
There is no response if the module detects a syntax error or communication error.
! is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of the analog input module.
(cr) is the terminating character, carriage return (0Dh).

14.4.9 $AAM Reads The Module Name

Description The command will read the module name.

Syntax $AAM (cr)

is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of an analog input module.
M read the module name command
(cr) is the terminating character, carriage return (0Dh).

Response !AA(name)(cr) if the command is valid or ?AA (cr) if the command is invalid
There is no response if the module detects a syntax error or communication error.
! is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of the analog input module.
name module name string
(cr) is the terminating character, carriage return (0Dh).

14.4.10 ~AAO(name) Set The Module Name

Description The command will set module name

Syntax ~AA6(name)(cr)

#	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of an analog input module.
O	set name command
name	1~8 digits
(cr)	is the terminating character, carriage return (0Dh).

Response !AA(cr) if the command is valid or ?AA (cr) if the command is invalid

There is no response if the module detects a syntax error or communication error.

!	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of the analog input module.
(cr)	is the terminating character, carriage return (0Dh).

Example Command：~01Oabcdef(cr)

Response：!01 (cr)

The module name is "abcdef"

14.4.11 $AAMASK Read Mask Address

Description The command will read the module sub mask.

Syntax $AAMASK(cr)

#	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of an analog input module.
MASK	read the module sub mask command
(cr)	is the terminating character, carriage return (0Dh).

Response !AAnnnnnnnn(cr) if the command is valid or ?AA (cr) if the command is invalid

There is no response if the module detects a syntax error or communication error.

!	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of the analog input module.
nnnnnnnn	sub mask (8 digits) (such as FFFFFF00)
(cr)	is the terminating character, carriage return (0Dh).

14.4.12 $AAMASKnnnnnnnn Set Mask Address

Description The command will set the module sub mask.

Syntax $AAMASKnnnnnnnn(cr)

#	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of an analog input module.
MASK	set the module sub mask command
nnnnnnnn	sub mask(8 digits) (such as FFFFFF00=255.255.255.0)
(cr)	is the terminating character, carriage return (0Dh).

Response !AA (cr) if the command is valid or ?AA (cr) if the command is invalid

There is no response if the module detects a syntax error or communication error.

!	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of the analog input module.
(cr)	is the terminating character, carriage return (0Dh).

14.4.13 $AAPRead The Communication Protocol

Description The command will read communication protocol .

Syntax $AAP(cr)

#	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of an analog input module.
P	read communication protocol command
(cr)	is the terminating character, carriage return (0Dh).

Response !AAn(cr) if the command is valid or ?AA (cr) if the command is invalid

There is no response if the module detects a syntax error or communication error.

!	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of the analog input module.
n	n=0 ASCII command protocol, n=1 ModBus RTU protocol
(cr)	is the terminating character, carriage return (0Dh).

14.4.14 $AAPv Set The Communication Protocol

Description The command will set communication protocol .

Syntax $AAPn(cr)

#	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of an analog input module.
P	read communication protocol command
n	n=0 ASCII command protocol, n=1 ModBus RTU protocol
(cr)	is the terminating character, carriage return (0Dh).

Response !AA(cr) if the command is valid or ?AA (cr) if the command is invalid

There is no response if the module detects a syntax error or communication error.

!	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of the analog input module.
(cr)	is the terminating character, carriage return (0Dh).

14.4.15 $AASW Read Web Server Status

Description The command will read web server status.

Syntax $AASW(cr)

#	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of an analog input module.
WEB	read we server enable/disable status
(cr)	is the terminating character, carriage return (0Dh).

Response !AAn(cr) if the command is valid or ?AA (cr) if the command is invalid

There is no response if the module detects a syntax error or communication error.

!	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of the analog input module.
n	Web server status, n=0 disable, n=1 enable
(cr)	is the terminating character, carriage return (0Dh).

14.4.16 $AASWv Enabled/Disable Web Server

Description The command will enable/disable web server.

Syntax $AASWn(cr)

#	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of an analog input module.
WEB	enable/disable web server command
n	Web server status, n=0 disable, n=1 enable
(cr)	is the terminating character, carriage return (0Dh).

Response !AA(cr) if the command is valid or ?AA (cr) if the command is invalid

There is no response if the module detects a syntax error or communication error.

!	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of the analog input module.
(cr)	is the terminating character, carriage return (0Dh).

14.4.17 $AADHCP Read DHCP Status

Description The command will read DHCP status.

Syntax $AADHCP(cr)

#	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of an analog input module.
DHCP	read DHCP enable/disable status
(cr)	is the terminating character, carriage return (0Dh).

Response !AAn(cr) if the command is valid or ?AA (cr) if the command is invalid

There is no response if the module detects a syntax error or communication error.

!	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of the analog input module.
n	DHCP status=0 disable, n=1 enable
(cr)	is the terminating character, carriage return (0Dh).

14.4.18 $AADHCPv Enabled/Disable DHCP

Description The command will enable/disable DHCP.

Syntax $AASWn(cr)

#	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of an analog input module.
WEB	enable/disable web server command
n	enable/disable DHCP ,n=0 disable, n=1 enable
(cr)	is the terminating character, carriage return (0Dh).

Response !AA(cr) if the command is valid or ?AA (cr) if the command is invalid

There is no response if the module detects a syntax error or communication error.

!	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of the analog input module.
(cr)	is the terminating character, carriage return (0Dh).

14.4.19 ^AAMAC Read MAC Address

Description The command will read MAC Address.

Syntax ^AAMAC(cr)

^	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of an analog input module.
MAC	read Read MAC Address
(cr)	is the terminating character, carriage return (0Dh).

Response !AAnnnnnnnnnnnn(cr) if the command is valid or ?AA (cr) if the command is invalid

There is no response if the module detects a syntax error or communication error.

!	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of the analog input module.
nnnnnnnnnnnn	MAC Address
(cr)	is the terminating character, carriage return (0Dh).

14.4.20 ~AA6v Set Module Led Control

Description The command will Set Module Led Control.

Syntax ~AA6v(cr)

#	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of an analog input module.
6	set module Led control command
v	LED control mode v="H" controlled by host, v="M" controlled by module
n	enable/disable DHCP ,n=0 disable, n=1 enable
(cr)	is the terminating character, carriage return (0Dh).

Response !AA(cr) if the command is valid or ?AA (cr) if the command is invalid

There is no response if the module detects a syntax error or communication error.

!	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of the analog input module.
(cr)	is the terminating character, carriage return (0Dh).

14.4.21 ~AA6ddddddddd Write Data To Led Board

Description The command will Write Data To Led Board.

Syntax ~AA6ddddddddd(cr)

#	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of an analog input module.
6	set module Led control command
dddddddd	6 digits (32-bits) LED data
n	enable/disable DHCP ,n=0 disable, n=1 enable
(cr)	is the terminating character, carriage return (0Dh).

Response !AA(cr) if the command is valid or ?AA (cr) if the command is invalid

There is no response if the module detects a syntax error or communication error.

!	is a delimiter character.
AA	(range 00-FF) represents the 2-character hexadecimal address of the analog input module.
(cr)	is the terminating character, carriage return (0Dh).

14.4.22 ~AA6ddddddddnnnn Force Led To Flash

Description The command will force Led to flash for the specified times

Syntax ~AA6ddddddddnnnn(cr)

is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of an analog input module.
6 set module Led control command
dddddddd 6 digits (32-LED channels, bit n=1 enable LED channel n flash
nnnn flash times (0000~FFFF)
(cr) is the terminating character, carriage return (0Dh).

Response !AA(cr) if the command is valid or ?AA (cr) if the command is invalid
There is no response if the module detects a syntax error or communication error.

! is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of the analog input module.
(cr) is the terminating character, carriage return (0Dh).

14.4.23 ~AA3ettttdddd Write Communication Timeout settings

Description Enable/disable Communication timeout watchdog, set timeout period and power-on delay time

Syntax ~AA3ettttdddd(cr)

~ Is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of module.
3 is the set communication timeout command
e Enable/disable communication timeout watchdog, e=0 disable=1 enable
tttt (range 0000~FFFF) Communication timeout period (msec)
dddd (range 0000~FFFF) Power-on delay time (msec) to start Communication timeout watchdog
(cr) Is the terminating character, carriage return (0Dh).

Response !AA(cr) If the command is valid or ?AA (cr) if the command is invalid
There is no response if the module detects a syntax error or communication error.

! Is a delimiter character.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
(cr) Is the terminating character, carriage return (0Dh)

Example Command：~013100FF01FF(cr)
Response：!01(cr)
The command set communication timeout period=00FF msec and power-on delay time=1FF msec, and enable communication timeout watchdog.
The DO channels will be set to DO safe value when compunction timeout occurred

Example Command：~013000FF01FF(cr)
Response：!01(cr)
The command to disable communication timeout watchdog.

14.4.24 ~AA3 Read Communication Timeout settings

Description Read Communication timeout watchdog status, communication timeout period, and power-on delay time

Syntax ~AA3etttdddd(cr)

~ Is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of module.
3 is the set communication timeout command
(cr) Is the terminating character, carriage return (0Dh).

Response !AAetttdddd(cr) If the command is valid or ?AA (cr) if the command is invalid
There is no response if the module detects a syntax error or communication error.

! Is a delimiter character.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
(cr) Is the terminating character, carriage return (0Dh)
e communication timeout watchdog status, e=0 disable. e=1 enable
tttt (range 0000~FFFF) Communication timeout period (msec)
dddd (range 0000~FFFF) Power-on delay time (msec) to start Communication timeout watchdog
(cr) Is the terminating character, carriage return (0Dh).

Example Command：~013(cr)
Response：!01100FF01FF(cr)
The communication timeout watchdog is enabled, timeout period=00FF (msec),power-on delay time=01FF (msec)

14.4.25 #AA Read The Analog Inputs Of All

Description The command will return the input value from a specified (AA) module in the currently configured data format.

Syntax #AA(cr)

Is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of an analog input module.
(cr) Is the terminating character, carriage return (0Dh).

Response >(data)(cr) If the command is valid or ?AA (cr) if the command is invalid
There is no response if the module detects a syntax error or communication error.

> Is a delimiter character.
(data) Is the input value in the configured data format of the module
(cr) Is the terminating character, carriage return (0Dh)

Example Command：#21(cr)
Response：
>+7.2111+7.2567+7.3125+7.1000+7.4712+7.2555+7.1234+7.5678+7.2111+7.2567+7.3125+7.1000+7.4712+7.2555+7.1234+7.5678 +3.5678 (cr)
The command response the analog input module at address 21h for its input values of all channels. The analog input module responds with channels from 0 to 15 with +7.2111 volts, +7.2567 volts,+7.3125 volts, +7.1000 volts, +7.4712 volts, +7.2555 volts, +7.1234 volts ,+7.5678 volts,+7.2111 volts, +7.2567 volts,+7.3125 volts, +7.1000 volts, +7.4712 volts, +7.2555 volts, +7.1234 volts and +7.5678 volts. The average value is +3.5678 volts

Example Command：#01(cr)
Response：
>FF5DE4323212AE33233455663E000FF03FF5DE4323212AE33233455663E000FF03FE02 (cr)
The analog input module at address01 has an input value of FF5DE4323212AE33233455663E000FF03FF5DE4323212AE33233455663E000FF03FE02. (The configured data format of the analog input module is two's complement)Where FE02 is the average value

14.4.26 #AAn Read The Single Analog Input

Description The command will return the input value from one of the all channels of a specified (AA) module in the currently configured data format.

Syntax #AAN(cr)
Is a delimiter character.
AA (range 00-3F) represents the 2-character hexadecimal address of the analog input module.
N Identifies the channel you want to read. The value can range from 0 to F
(cr) Is the terminating character, carriage return (0Dh).

Response >(data)(cr) If the command is valid or ?AA (cr) if the command is invalid
There is no response if the module detects a syntax error or communication error.
> Is a delimiter character.
(data) Is the input value of the channel number N. Data consists of a + or - sign followed by five decimal digits with a fixed decimal point.
(cr) Is the terminating character, carriage return (0Dh).

Example Command：#120(cr)
Response：>+1.4567(cr)
The command requests the analog input module at address 12h to return the input value of channel 0. The analog input module responds that the input value of channel 0 is equal to +1.4567 volts.

14.4.27 #AAMH Read Maximum Value Of All Channels

Description The command will return the Read Maximum Value of All Channels from a specified (AA) module in the currently configured data format.

Syntax #AAMH(cr)
is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of an analog input module.
MH Read all channel maximum value command
(cr) is the terminating character, carriage return (0Dh).

Response >AA(data)(cr) if the command is valid or ?AA (cr) if the command is invalid
There is no response if the module detects a syntax error or communication error.
AA (range 00-FF) represents the 2-character hexadecimal address of the analog input module.
> is a delimiter character.
(data) is the Maximum value of all channels in the configured data format of the module.
(cr) is the terminating character, carriage return (0Dh).

Example Command：#21MH(cr)
Response：!21+7.2111+7.2567+7.3125+7.1000+7.4712+7.2555+7.1234+7.5678+7.2111 +7.2567+7.3125+7.1000+7.4712+7.2555+7.1234+7.5678 (cr)
The command response the analog input module at address 21h for its maximum values of all channels. The analog input module responds its maximum values with channels from 0 to 15 with +7.2111 volts, +7.2567 volts, +7.3125 volts, +7.1000 volts, +7.4712 volts, +7.2555 volts, +7.1234 volts, +7.5678 volts,+7.2111 volts,+7.2567 volts,+7.3125 volts, +7.1000 volts, +7.4712 volts, +7.2555 volts, +7.1234 volts and +7.5678 volts

Example Command：#01MH(cr)
Response：!01FF5DE4323212AE3323345663E000FF03 F5DE4323212AE33233455663E000FF03 (cr)
The analog input module at address01 has maximum values of FF5DE4323212AE3323345663E000FF03FF5DE4323212AE3323345663E000FF03.
(The configured data format of the analog input module is two's complement)

14.4.28 #AAMHn Read Maximum Value Of Specified Channel

Description The command will return the Read Maximum value of the specified channel from a specified (AA) Module in the currently configured data format.

Syntax #AAMHn(cr)
is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of an analog input module.
MH Read single channel maximum value command
N channel number
(cr) is the terminating character, carriage return (0Dh).

Response >AA(data)(cr) if the command is valid or ?AA (cr) if the command is invalid
There is no response if the module detects a syntax error or communication error.
AA (range 00-FF) represents the 2-character hexadecimal address of the analog input module.
> is a delimiter character.
(data) is the Maximum value of the specified channels.
(cr) is the terminating character, carriage return (0Dh).

Example Command：#21MH2(cr)
Response：!21+7.2111(cr)
The command response the analog input module at address 21h for its maximum values of the channels 2 with +7.2111 volts

Example Command：#01MH2(cr)
Response：!01FF5D (cr)
The command response the analog input module at address 21h for its maximum values of the channels 2 with FF5D (The configured data format of the analog input module is two's complement)

14.4.29 $AAMH Clear All Maximum Value

Description Clear maximum Value Of all channels

Syntax $AAMH(cr)
$ is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of module.
MH is the clear maximum value of specified channel command.
(cr) is the terminating character, carriage return (0Dh).

Response !AA(cr) if the command is valid or ?AA (cr) if the command is invalid.
There is no response if the module detects a syntax error or communication error.
! Delimiter character indicates a valid command was received.
? Delimiter character indicates the command was invalid.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
(cr) is the terminating character, carriage return (0Dh).

Example Command：$01MH(cr)
Response：!01(cr)
Clear Maximum Value Of all Channels

14.4.30 $AAMHn Clear Maximum Value Of Specified Channel

Description Clear maximum Value Of specified channel

Syntax $AAMHn(cr)

$ is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of module.
MH is the clear maximum value of specified channel command.
n channel 0~15
(cr) is the terminating character, carriage return (0Dh).

Response !AA(cr) if the command is valid or ?AA (cr) if the command is invalid.
There is no response if the module detects a syntax error or communication error.

! Delimiter character indicates a valid command was received.
? Delimiter character indicates the command was invalid.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
(cr) is the terminating character, carriage return (0Dh).

Example Command：$01MHE(cr)
Response：!01(cr)
Clear Maximum Value Of Channel 14 (0x0E)

14.4.31 #AAML Read Minimum Value Of All Channels

Description The command will return the Read Minimum Value of All Channels from a specified (AA) Module in the currently configured data format.

Syntax #AAML(cr)

is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of an analog input module.
ML Read all channel minimum value command
(cr) is the terminating character, carriage return (0Dh).

Response >AA(data)(cr) if the command is valid or ?AA (cr) if the command is invalid
There is no response if the module detects a syntax error or communication error.

AA (range 00-FF) represents the 2-character hexadecimal address of the analog input module.
> is a delimiter character.
(data) is the minimum value of all channels in the configured data format of the module.
(cr) is the terminating character, carriage return (0Dh).

Example Command：#21ML(cr)
Response：
!21+7.2111+7.2567+7.3125+7.1000+7.4712+7.2555+7.1234+7.5678+7.2111+7.2567+7.3125+7.1000+7.4712+7.2555+7.1234+7.5678 (cr)
The command response the analog input module at address 21h for its minimum values of all channels. The analog input module responds its minimum values with channels from 0 to 15 with +7.2111 volts, +7.2567 volts, +7.3125 volts, +7.1000 volts, +7.4712 volts, +7.2555 volts, +7.1234 volts, +7.5678 volts,+7.2111 volts,+7.2567 volts,+7.3125 volts, +7.1000 volts, +7.4712 volts, +7.2555 volts, +7.1234 volts and +7.5678 volts

Example Command：#01ML(cr)
Response：
!01FF5DE4323212AE332334 5663E000FF03F5DE4323212AE332334 5663E000FF03(cr)
The analog input module at address01 has minimum values of FF5DE4323212AE332334 5663E000FF03FF5DE4323212AE332334 5663E000FF03.
(The configured data format of the analog input module is two's complement)

14.4.32 #AAMLn Read Minimum Value Of Specified Channel

Description The command will return the read minimum value of the specified channel from a specified (AA) Module in the currently configured data format.

Syntax #AAMLn(cr)
is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of an analog input module.
ML read single channel maximum value command
N channel number
(cr) is the terminating character, carriage return (0Dh).

Response >AA(data)(cr) if the command is valid or ?AA (cr) if the command is invalid
There is no response if the module detects a syntax error or communication error.
AA (range 00-FF) represents the 2-character hexadecimal address of the analog input module.
> is a delimiter character.
(data) is the minimum value of the specified channels.
(cr) is the terminating character, carriage return (0Dh).

Example Command：#21ML2(cr)
Response：!21+7.2111(cr)
The command response the analog input module at address 21h for its minimum values of the channels 2 with +7.2111 volts

Example Command：#01ML2(cr)
Response：!01FF5D (cr)
The command response the analog input module at address 21h for its minimum values of the channels 2 with FF5D (The configured data format of the analog input module is two's complement)

14.4.33 $AAML Clear All Minimum Value

Description Clear minimum Value Of all channels

Syntax $AAML(cr)
$ is a delimiter character.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
ML is the clear minimum value of all channels command.
(cr) is the terminating character, carriage return (0Dh).

Response !AA(cr) if the command is valid or ?AA (cr) if the command is invalid.
There is no response if the module detects a syntax error or communication error.
! Delimiter character indicates a valid command was received.
? Delimiter character indicates the command was invalid.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
(cr) is the terminating character, carriage return (0Dh).

Example Command：$01ML(cr)
Response：!01(cr)
Clear Minimum Value Of all Channels

14.4.34 $AAMLn Clear Minimum Value Of Specified Channel

Description Clear Minimum Value Of specified Channel

Syntax $AAMLn(cr)

$ is a delimiter character.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
ML is the clear minimum value of all channels command.
n channel 0~15
(cr) is the terminating character, carriage return (0Dh).

Response !AA(cr) if the command is valid or ?AA (cr) if the command is invalid.
There is no response if the module detects a syntax error or communication error.

! Delimiter character indicates a valid command was received.
? Delimiter character indicates the command was invalid.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
(cr) is the terminating character, carriage return (0Dh).

Example Command：$01MLE(cr)
Response：!01(cr)
Clear Minimum Value of Channel 14 (0x0E)

14.4.35 #AAAV Read Average Value

Description The command will return the average value from the channels which is in average mode

Syntax #AAV(cr)

is a delimiter character.
AA (range 00-3F) represents the 2-character hexadecimal address of the analog input module.
V identifies Read Ad average value command
(cr) is the terminating character, carriage return (0Dh).

Response >(data)(cr) if the command is valid or ?AA (cr) if the command is invalid
There is no response if the module detects a syntax error or communication error.

! is a delimiter character.
The command requests the analog input module at address 12h to return the input value of channel 0. The analog input module responds that the input value of channel 0 is equal to +1.4567 volts.

Example Command：#12V(cr)
Response：>+1.4567(cr)
The command requests the analog input module at address 12h to return the average value of the module. The analog input module responds that the average value of module is equal to +1.4567 volts.

14.4.36 $AAE Read Channel Average Enable/Disable Status

Description Read A/D channel in average status

Syntax $AAE (cr)

$ is a delimiter character.
AA (range 003F) represents the 2-character hexadecimal address of module.
E is Read A/D channel in average status command.
(cr) is the terminating character, carriage return (ODh).

Response !AAnnnn(cr) if the command is valid or ?
AA (cr) if the command is invalid.
There is no response if the module detects a syntax error or communication error.
! is a delimiter character indicating a valid command was received.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
nnnn are four hexadecimal values. The values are interpreted by the module as four binary words(4-bit). The first word represents channel 12~15, and the second word represents channel 8~11…etc.
bit x=1 channel x is in average, bit x=0 channel isn't in average
(cr) is the terminating character, carriage return

Examples Command：$01E(cr)
Response：!011020 (cr)
Channel 5 and channel 15 are in average only

14.4.37 $AAEnnnn Disable/Enable Channel in Average

Description Enables/disables channels to be in average mode.

Syntax $AAEvvvv(cr)

$ is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of module.
E is the Enable/disable channels to be in average command.
vvvv are four hexadecimal values. The values are interpreted by the module as four binary words (4-bit). The first word represents the status of channel 12~15, and the second word represents the status of channel 8~11…etc.
Value 0 means enable channel to be in average, value 1 means disable channel to be not in average.
(cr) is the terminating character, carriage return (0Dh).

Response !AA(cr) if the command is valid or ?
AA (cr) if the command is invalid.
There is no response if the module detects a syntax error or communication error.
! Delimiter character indicates a valid command was received.
? Delimiter character indicates the command was invalid.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
(cr) is the terminating character, carriage return (0Dh).

Example Command：$01E0103(cr)
Response：!01(cr)
Hexadecimal 0103 equals binary 0000 0001 0000 0011, which enables channel 0, 1 and 8 to be in average mode only

14.4.38 #AAAL Read AD High/Low Alarm Status

Description The command will return the alarm status from all channels of a specified (AA) module.

Syntax #AAAL(cr)

is a delimiter character.

AA (range 00-FF) represents the 2-character hexadecimal address of the analog input module.

AL identifies Read Ad high/low Alarm Status command

(cr) is the terminating character, carriage return (0Dh).

Response !AA(hhhh)(llll)(cr) if the command is valid or ? AA (cr) if the command is invalid

There is no response if the module detects a syntax error or communication error.

! Is a delimiter character.

Hhhhare four hexadecimal values. The values are interpreted by the module as four binary words (4-bit). The first word represents the status of channel 12~15, and the second word represents the status of channel 8~11…etc.
bit x=0 means the channel x is high alarm, bit x= 1 means the channel x isn't high alarm.

(cr) Is the terminating character, carriage return (0Dh).

llll Are four hexadecimal values. The values are interpreted by the module as four binary words (4-bit). The first word represents the status of channel 12~15, and the second word represents the status of channel 8~11…etc.
bit x=0 means the channel x is low alarm, bit x= 1 means the channel x isn't low alarm.

Example Command：#01AL(cr)

Response：!0100010010(cr)

The command requests the analog input module at address 12h to return thethe alarm status of all channels.

The analog input module responds that the alarm value of all channel s is equal to 00010010 (hex)

The high alarm status=0001 means the channel 0 is high alarm

The low alarm status=0010 means the channel 4 is low alarm

14.4.39 $AAAHnnnn Clear A/D High Alarm

Description Clear A/D High Alarm status (over range status).

Syntax **$AAAHnnnn (cr)**

$ is a delimiter character.

AA (range 003F) represents the 2-character hexadecimal address of module.

H is the clear high alarm command.

nnnn are four hexadecimal values. The values are interpreted by the module as four binary words (4-bit). The first word represents channel 12~15, and the second word represents channel 8~11…etc.
bit x=1 clear high alarm status of channel x, bit x=0 no clear

(cr) is the terminating character, carriage return (ODh).

Response **!AA(cr)** if the command is valid or **?AA (cr)** if the command is invalid.

There is no response if the module detects a syntax error or communication error.

! Is a delimiter character indicating a valid command was received.

AA (range 00-3F) represents the 2-character hexadecimal address of module.

(cr) is the terminating character, carriage return

Examples Command：**$01H0101(cr)**

Response：**!01 (cr)**

Clear high alarm status of channel 0 and 4

14.4.40 $AAALnnnn Clear A/D Low Alarm

Description Clear A/D high alarm status (under range status).

Syntax $AAALnnnn (cr)

$ is a delimiter character.

AA (range 003F) represents the 2-character hexadecimal address of module.

L is the clear high alarm command.

nnnn are four hexadecimal values. The values are interpreted by the module as four binary words (4-bit). The first word represents channel 12~15, and the second word represents channel 8~11…etc.

bit x=1 clear low alarm status of channel x, bit x=0 no clear

(cr) is the terminating character, carriage return (ODh).

Response !AA(cr) if the command is valid or ?AA (cr) if the command is invalid.

There is no response if the module detects a syntax error or communication error.

! Is a delimiter character indicating a valid command was received.

AA (range 00-3F) represents the 2-character hexadecimal address of module.

(cr) is the terminating character, carriage return

Examples Command：$01L0101(cr)

Response：!01 (cr)

Clear low alarm status of channel 0 and 4

14.4.41 $AAB Read Channel Burnout Status

Description Read channel burn out status

Syntax $AAB(cr)

$ is a delimiter character.

AA (range 00-3F) represents the 2-character hexadecimal address of module.

B is the Channel Diagnose command.

(cr) is the terminating character, carriage return (0Dh).

Response !AAnnnn(cr) if the command is valid

?AA(cr) if an invalid command was issued.

There is no response if the module detects a syntax error or communication error.

! Delimiter character indicates a valid command was received.

? Delimiter character indicates the command was invalid.

AA (range 00-3F) represents the 2-character hexadecimal address of the module.

Nnnn (range 0000-FFFF) is a hexadecimal number that equals the 16-bit parameter, representing the status of analog input channels. Bit value 0 means normal status; and bit value 1 means channel open wiring.

(cr) is the terminating character, carriage return (0Dh)

Examples Command：$01B(cr)

Response：!010101(cr)

Channel 0, 8 are open wiring and channel 1~7 and 9~15 are all normal.

14.4.42 %AAB Read Channel Burnout Enable/Disable Status

Description Read channel burnout detection enables/disables status of a specified input module.

Syntax %AAB(cr)

% is a delimiter character.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
B is the Enable/disable burnout command.
(cr) is the terminating character, carriage return (0Dh).

Response !AAbb(cr) if the command is valid or ?AA (cr) if the command is invalid.
There is no response if the module detects a syntax error or communication error.

! Delimiter character indicates a valid command was received.
? Delimiter character indicates the command was invalid.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
B burnout enable/disable status, 0: disable, 1: enable
(cr) is the terminating character, carriage return (0Dh).

Example Command：%01B(cr) read burnout detection enable/disable status of specified module
Response：!001(cr) burnout detection is enabled

Example Command：%01B(cr) read burnout detection enable/disable status of specified module
Response：!001(cr) burnout detection is enabled

14.4.43 %AABn Enable/Disable Burnout Detection

Description Enables/disables channel burnout detection of a specified input module.

Syntax %AABn(cr)

% is a delimiter character.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
B is the Enable/disable burnout command.
n represents enable or disable burnout, value 1 means enable burnout detection, value 0 means disable burnout detection.
(cr) is the terminating character, carriage return (0Dh).

Response !AA(cr) if the command is valid or ?AA (cr) if the command is invalid.
There is no response if the module detects a syntax error or communication error.

! Delimiter character indicates a valid command was received.
? Delimiter character indicates the command was invalid.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
(cr) is the terminating character, carriage return (0Dh).

Example Command：%00B1(cr) this command enable burnout detection of specified module
Response：!00(cr) this command disable burnout detection of specified module

Example Command：%00B0(cr)
Response：!00(cr)

14.4.44 $AA3 Read The CJC Temperature

Description Read cold junction temperature.

Syntax $AA3(cr)

$ is a delimiter character.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
3 is the Read cold junction temperature command.
(cr) is the terminating character, carriage return (0Dh).

Response >DATA(cr) if the command is valid or ?AA (cr) if the command is invalid.

There is no response if the module detects a syntax error or communication error.

> Delimiter character indicates a valid command was received.
? Delimiter character indicates the command was invalid.
DATA CJC temperature in degrees Celsius, consisting of of a sign byte, '+' or '-' and followed by 5 decimal digits with a fixed decimal point in tenth of a degree
(cr) is the terminating character, carriage return (0Dh).

Example Command：$043(cr)

Response：>+0030.2(cr)

The command asks the analog input module at address 04h to send its cold junction temperature data. The module responds with +0030.2C.

14.4.45 ~AAC Read The CJC Enable/Disable

Description read cold junction compensation enable/disable status.

Syntax ~AAC(cr)

~ is a delimiter character.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
C read CJC enable/disable status command.
(cr) is the terminating character, carriage return (ODh).

Response !AAn if the command is valid or ?AA (cr) if the command is invalid.

There is no response if the module detects a syntax error or communication error.

! is a delimiter character indicating a valid command was received.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
N n=1 if CJC enabled, n=0 if CJC disabled
(cr) is the terminating character, carriage return

Examples Command：~01C(cr)

Response：!011(cr)

CJC for all channels is enabled

14.4.46 ~AACn Enable/Disable The CJC

Description enable/disable cold junction compensation.

Syntax ~AACn(cr)

~ s a delimiter character.
AA range 00-3F) represents the 2-character hexadecimal address of module.
C S the enable/disable CJC command.
N =0 disable CJC, n=1 enable CJC
(cr) s the terminating character, carriage return (ODh).

Response !AA if the command is valid or ?AA (cr) if the command is invalid.
There is no response if the module detects a syntax error or communication error.
! s a delimiter character indicating a valid command was received.
AA range 00-FF) represents the 2-character hexadecimal address of module.
(cr) s the terminating character, carriage return

Examples Command：~01C1(cr)
Response：!01(cr)
Noble cold junction compensation

14.4.47 $AA9snnnn Set The All Channel CJC Offset

Description Set all channels to have the same cold junction offset.

Syntax $AA9snnnn(cr)

$ Is a delimiter character.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
9 Is the set cold junction offset command.
S sign of cold junction offset
nnnn cold junction offset (Hex) in 0.01C unit
(cr) Is the terminating character, carriage return (ODh).

Response !AA If the command is valid or ?AA (cr) if the command is invalid.
There is no response if the module detects a syntax error or communication error.
! Is a delimiter character indicating a valid command was received.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
(cr) Is the terminating character, carriage return

Examples Command：$019+0010(cr)
Response：!01(cr)
Set all channels to have the same cold junction offset to +0010(Hex)*0.01=+0.16C.

14.4.48 $AA9c Read Single Channel CJC Offset

Description read cold junction offset of specified channel

Syntax $AA9c(cr)

$ is a delimiter character.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
9 is the set cold junction offset command.
C channel number
(cr) is the terminating character, carriage return (ODh).

Response !AAsnnnn if the command is valid or ?AA (cr) if the command is invalid.
There is no response if the module detects a syntax error or communication error.
! Is a delimiter character indicating a valid command was received.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
S sign of cold junction offset
nnnn cold junction offset in 0.01C unit
(cr) is the terminating character, carriage return

Examples Command：$0192(cr)
Response：!01+0010(cr)
The cold junction offset of channel 2 is +0010(Hex)*0.01=+0.16C.
Response：!AA if the command is valid or ?AA (cr) if the command is invalid.
There is no response if the module detects a syntax error or communication error.
! is a delimiter character indicating a valid command was received.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
(cr) is the terminating character, carriage return

Examples Command：$019+0010(cr)
Response：!01(cr)
Set all channels to have cold junction offset to +0010(Hex)*0.01=+0.16C.

14.4.49 $AA9cSnnnn Set Single Channel CJC Offset

Description set channel cold junction offset individually

Syntax $AA9csnnnn(cr)

$ is a delimiter character.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
9 is the set cold junction offset command.
c is the channel number (0~F)
s sign of cold junction offset
nnnn cold junction offset (Hex) in 0.01C unit
(cr) is the terminating character, carriage return (ODh).

Response !AA if the command is valid or ?AA (cr) if the command is invalid.
There is no response if the module detects a syntax error or communication error.
! Is a delimiter character indicating a valid command was received.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
(cr) is the terminating character, carriage return

Examples Command：$0193+0010(cr)
Response：!01(cr)
Set cold junction offset to +0010(Hex)*0.01=+0.16C to channel 3

14.4.50 $AAR Read AD Filter Value

Description Read A/D cutoff frequency

Syntax $AAR(cr)

$ is a delimiter character.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
R read A/D cutoff frequency command.
(cr) is the terminating character, carriage return (ODh).

Response !AAn if the command is valid or ?AA (cr) if the command is invalid.
There is no response if the module detects a syntax error or communication error.

! is a delimiter character indicating a valid command was received.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
n 0: 50Hz, 1:60Hz, 2:100Hz, 3:120Hz
(cr) is the terminating character, carriage return

Examples Command：$01R(cr)
Response：!011(cr)
A/D cutoff frequency is 60Hz

14.4.51 $AARf Set AD Filter Value

Description Set A/D cutoff frequency

Syntax $AAR(cr)

$ is a delimiter character.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
R read A/D cutoff frequency command.
f 0: 50Hz, 1:60Hz, 2:100Hz, 3:120Hz
(cr) is the terminating character, carriage return (ODh).

Response !AAn if the command is valid or ?AA (cr) if the command is invalid.
There is no response if the module detects a syntax error or communication error.

! is a delimiter character indicating a valid command was received.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
(cr) is the terminating character, carriage return

Examples Command：$01R0(cr)
Response：!011(cr)
Set A/D cutoff frequency to 560Hz

14.4.52 $AA6 Read the Channel Enable/Disable Status

Description Read the status of digital input/output channels

Syntax $AA6(cr)

$ is a delimiter character.
AA represents the 2-character hexadecimal module address
6 is the Digital Data In command.
(cr) is the terminating character, carriage return (0Dh)

Response !AA00(data1)(data2)(cr) if the command is valid.

?AA(cr) if an invalid operation was entered.
! Delimiter indicating a valid command was received.
? Delimiter indicating the command was invalid.
AA represents the 2-character hexadecimal module address of an EX-9200 module.
(data1) An 8-characters hexadecimal value representing the values of the digital input channels.
(data2) An 8-characters hexadecimal value representing the values of the digital output channels.
(cr) is the terminating character, carriage return (0Dh)

Example Read the values of all DI/DO channels

Command：$016(cr)

Response：!010000000F000000FD(cr)

The 4~ 11 characters (0000000F) indicate DI 0~3 channels are active, and DI 04~31 channels are inactive

The 12~ 19 characters (000000FD) indicate DO 0,2,3,4,5,6,7 channels are active, and 1, 4, 8~31 channels are inactive

14.4.53 $AA5vvvv Enable/Disable A/D Channels

Description Enables/disables multiplexing simultaneously for separate channels of a specified input module.

Syntax $AA5vvvv(cr)

$ is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of module.
5 is the Enable/disable Channels command.
vvvv are four hexadecimal values. The values are interpreted by the module as four binary words (4-bit). The first word represents the status of channel 4~7, and the second word represents the status of channel 0~3…etc. Value 0 means the channel is disabled, value 1 means the channel is enabled.
(cr) is the terminating character, carriage return (0Dh).

Response !AA(cr) if the command is valid or ?AA (cr) if the command is invalid.

There is no response if the module detects a syntax error or communication error.

! Delimiter character indicates a valid command was received.
? Delimiter character indicates the command was invalid.
AA (range 00-FF) represents the 2-character hexadecimal address of module.
(cr) is the terminating character, carriage return (0Dh).

Example Command：$0058100(cr)

Response：!00(cr)

Hexadecimal 8100 equals binary 1000 0001 0000 0000, which enables channel 8, 15 and disables channels 0,1,2,3,4, 5, 6,7, and 9,10,11,12,13, and 14..

14.4.54 $AA8Ci Read the Single A/D Channel Range

Description The command read individual channel type.
Syntax $AA8Ci (cr)
$ is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of module.
8C is the read channel type command.
i channel number
(cr) is the terminating character, carriage return (ODh).
Response !AACiRrr(cr) if the command is valid or ?AA (cr) if the command is invalid.
There is no response if the module detects a syntax error or communication error.
! Is a delimiter character indicating a valid command was received.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
i channel number(0~F)
rr type of channel i
(cr) is the terminating character, carriage return
Examples Command：$018C3(cr)
Response：!01C3R08(cr)
The type code of channel 3 is 08 (+/-10V).

14.4.55 $AA7CiRrr Set the Single Channel Range

Description The command set channel type individually.
Syntax $AA7CiRrr(cr)
$ is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of module.
7C is the Set channel type command.
i channel number
rr channel type code
(cr) is the terminating character, carriage return (ODh).
Response !AA if the command is valid or ?AA (cr) if the command is invalid.
There is no response if the module detects a syntax error or communication error.
! Is a delimiter character indicating a valid command was received.
AA (range 00-FF) represents the 2-character hexadecimal address of module.
(cr) is the terminating character, carriage return
Examples Command：017C3R08(cr)
Response：!01(cr)
Set type code 08 (+/-10V) to channel 3.

14.4.56 $AAS1 Reload the Default configuration

Description Reloads the Default configuration
Syntax $AAS1(cr)
$ is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of module.
Is the Reloads the Default configuration command.
(cr) is the terminating character, carriage return (0Dh).
Response !AA(cr) if the command is valid or ?AA (cr) if the command is invalid.
There is no response if the module detects a syntax error or communication error.
! Delimiter character indicates a valid command was received.
? Delimiter character indicates the command was invalid.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
(cr) is the terminating character, carriage return (0Dh).
Example Command：$01S1(cr)
Response：!01(cr)
Reloads the Default configuration

14.4.57 @AA Read the Digital I/O Status

Description read the status of its digital input and digital output channels

Syntax AA(cr)

@ is a delimiter character.

AA (range 00-3F) represents the 2-character hexadecimal Modbus network address (Always 01)

(cr) is the terminating character, carriage return (0Dh)

Response >(DI data)(DO data)(cr) if the command is valid.

?AA(cr) if an invalid operation was entered.

There is no response if the module detects a syntax error or communication error or if the address does not exist.

> Delimiter indicating a valid command was received.

? Delimiter indicating the command was invalid.

AA (range 00-3F) represents the 2-character hexadecimal Modbus network address of an EX-9000 module.

(DI data) 4-character hexadecimal value representing the values of the digital input module.

(DO data) 4-character hexadecimal value representing the values of the digital output module.

(cr) is the terminating character, carriage return (0Dh)

Example Command：@01(cr)

Response：>0000102000210001(cr)

00001020=the status of digital input channels. DI channels 4, 15 are active, and other channels are inactive

00210001=the status of digital output channels. DO channels 0, 16, 18 are active, and other channels are inactive

14.4.58 @AAnn Set the Digital Output Channels

Description Sets the Digital Output Channels

Command：@AAnn(cr)

Syntax @ Command leading code

AA Module address ID (00-3F)

nn Output value to channel 0~7

(cr) is the terminating character, carriage return (0Dh)

Response !AA(cr) Valid command

?AA(cr) Invalid command

(cr) is the terminating character, carriage return (0Dh)

Example Command：@0523(cr)

Response：!05(cr)

! Valid command

05 Module ID

23 Set 23(Hex) to Digital output channels (channel 0, 1, 5 are active, other channels are inactive)

14.4.59 @AAnnnn Set the Digital Output Channels

Description Sets the value to Digital Output Channels 0~15
Command：@AAnnnn(cr)

Syntax
@ Command leading code
AA Module address ID (00-3F)
nnnn Output value to channel 0~15
(cr) is the terminating character, carriage return (0Dh)

Response
!AA(cr) Valid command
?AA(cr) Invalid command
(cr) is the terminating character, carriage return (0Dh)

Example
Command：@050023(cr)
Response：!05(cr)
! Valid command
05 Module ID
0023 Set 23(Hex) to Digital output channels (channel 0, 1, 5 are active, and channel 2,3,5,6,7,8,9,10,11,12,13,14,15 are inactive)

14.4.60 @AAnnnnnn Set The Digital Output Channels

Description Sets the Digital Output Channels (0~23)
Command：@AAnnnnnn(cr)

Syntax
@ Command leading code
AA Module address ID (00-3F)
nnnnnn Output value to channel 0~23
(cr) is the terminating character, carriage return (0Dh)

Response
!AA(cr) Valid command
?AA(cr) Invalid command
(cr) is the terminating character, carriage return (0Dh)

Example
Command：@05010323(cr)
Response：!05(cr)
! Valid command
05 Module ID
010323 Set 23(Hex)to Digital output channels (channel 0,1,5,8,9,16 are active, other channels are inactive)

14.4.61 #AA0Ann Set The Digital 1's Byte(DO0~DO7) Output

Description Sets the value to Digital Output Channels 0~7
Command：#AA0Ann(cr)

Syntax
Command leading code
AA Module address ID (00-3F)
0A is the Sets the Digital Output lowest byte (DO0~DO7) command
nn Output value to channel 0~7
(cr) is the terminating character, carriage return (0Dh)

Response
!AA(cr) Valid command
?AA(cr) Invalid command
(cr) is the terminating character, carriage return (0Dh)

Example
Command：#050A23(cr)
Response：!05(cr)
! Valid command
05 Module ID
23 Set 23(Hex) to Digital output channels (channel 0, 1, 5 are active, and channel 2, 3, 4, 6, 7 are inactive)

14.4.62 #AA0Bnn Set The Digital 2's Byte(DO8~DO15) Output

Description Sets the value to Digital Output Channels 8~15
Command：#AA0Bnn(cr)

Syntax
Command leading code
AA Module address ID (00-3F)
0B is the Sets the Digital Output lowest byte(DO8~DO15) command
nn Output value to channel 8~15
(cr) is the terminating character, carriage return (0Dh)

Response
!AA(cr) Valid command
?AA(cr) Invalid command
(cr) is the terminating character, carriage return (0Dh)

Example
Command：#050B23(cr)
Response：!05(cr)
! Valid command
05 Module ID
23 Set 23(Hex) to Digital output channels (channel 8,9,13 are active, and channel 10, 11,12,14,15 are inactive)

14.4.63 #AA0Cnn Set the Digital 3's byte(DO16~DO23) Output

Description Sets the value to Digital Output Channels 16~23
Command：#AA0Cnn(cr)

Syntax
Command leading code
AA Module address ID (00-3F)
0C is the Sets the Digital Output lowest byte(DO16~DO23) command
nn Output value to channel 16~23
(cr) is the terminating character, carriage return (0Dh)

Response
!AA(cr) Valid command
?AA(cr) Invalid command
(cr) is the terminating character, carriage return (0Dh)

Example
Command：#050C23(cr)
Response：!05(cr)
! Valid command
05 Module ID
23 Set 23(Hex) to Digital output channels (channel 16, 17, 21 are active, and channel 18, 19,21,22,23 are inactive)

14.4.64 #AAnn Read Digital Input Counter

Description Read DI latch status.

Syntax #AAnn(cr)
is a delimiter character.
AA represents the 2-character hexadecimal Modbus address (Always 01)
nn represents DI channel number .
(cr) is the terminating character, carriage return (0Dh)

Response !AAnnnnnnnn(cr) if the command is valid.
?AA(cr) if an invalid operation was entered.
! Delimiter indicating a valid command was received.
? Delimiter indicating the command was invalid.
AA represents the 2-character hexadecimal module address of an EX-9200 module.
nnnnnnnn represents 4-bytes counter value
(cr) is the terminating character, carriage return (0Dh)

Example
Read DI latch status
Command：#0102(cr)
Response：!0100000003
Latch status= 00000003, DI #2 counter value=3

14.4.65 $AACn Clear Digital Input Counter

Description Clear Counter of all DI Channel
Syntax $AACn(cr)
$ is a delimiter character.
AA represents the 2-character hexadecimal module address
C is Clear DI counter command.
n is DI channel number
(cr) is the terminating character, carriage return (0Dh).
Response !AA(cr) if the command is valid.
?AA(cr) if an invalid operation was entered.
! Delimiter indicating a valid command was received.
? Delimiter indicating the command was invalid.
AA represents the 2-character hexadecimal address of an EX-9200 module.
(cr) is the terminating character, carriage return (0Dh).
Example Clear DI #2 counters value
Command：$01C2(cr)
Response：!01(cr)

14.4.66 $AACnn Clear Digital Input Counter

Description Clear Counter of all DI Channel
Syntax $AACn(cr)
$ is a delimiter character.
AA represents the 2-character hexadecimal module address
C is Clear DI counter command.
nn is DI channel number
(cr) is the terminating character, carriage return (0Dh).
Response !AA(cr) if the command is valid.
?AA(cr) if an invalid operation was entered.
! Delimiter indicating a valid command was received.
? Delimiter indicating the command was invalid.
AA represents the 2-character hexadecimal address of an EX-9200 module.
(cr) is the terminating character, carriage return (0Dh).
Example Clear DI #2 counters value
Command：$01C02(cr)
Response：!01(cr)

14.4.67 $AALS Read The Latched DI Status

Description Read DI latch status.
Syntax $AALS (cr)
$ is a delimiter character.
AA represents the 2-character hexadecimal Modbus address (Always 01)
LS represents read DI latch status command.
(cr) is the terminating character, carriage return (0Dh)
Response !AAnnnnnnnn(cr) if the command is valid.
?AA(cr) if an invalid operation was entered.
! Delimiter indicating a valid command was received.
? Delimiter indicating the command was invalid.
AA represents the 2-character hexadecimal module address of an EX-9200 module.
(cr) is the terminating character, carriage return (0Dh)
Example Read DI latch status
Command：$01LS (cr)
Response：!0100000003
Latch status= 00000003, DI #0 latched, DI #1 latched, and DI #2 ~ DI #11 no latched

14.4.68 $AAC Clear the latched DI status

Description Clear all digital input counter of specified DI channel.
Syntax $AAC(cr)
$ is a delimiter character.
AA represents the 2-character hexadecimal module address
CL is clear latch command.
(cr) is the terminating character, carriage return (0Dh).
Response !AA(cr) if the command is valid.
?AA(cr) if an invalid operation was entered.
! Delimiter indicating a valid command was received.
? Delimiter indicating the command was invalid.
AA represents the 2-character hexadecimal address of an EX-9200 module.
(cr) is the terminating character, carriage return (0Dh).
Example Clear all latch status
Command：$01C(cr)
Response：!01(cr)

14.4.69 $AA9nn Read Single Do Pulse High/Low Width

Description Read Do Pulse High/Low Width of specified DO channel
Command $AA9nn(cr)
Syntax $ Command leading code
AA Module address ID (00-3F)
9 is the Read Do Pulse High/Low Width command
nn Digital Output channel number (0~17)
(cr) is the terminating character, carriage return (0Dh)
Response !AAhhhhvvvv(cr) Valid command
hhhh=high pulse width in 0.5msec, vvvv=low pulse width in 0.5msec
?AA(cr) Invalid command
(cr) is the terminating character, carriage return (0Dh)
Example Command：$05902(cr)
Response：!0502000100(cr)
! Valid command
05 Module ID
0200 high pulse width=0200(hex) =512(dec)*0.5msec=256 msec
0100 low pulse width=0100(hex) =256(dec)*05msec=128 msec

14.4.70 $AA9nnhhhhllll Set Single Do Pulse High/Low Width

Description set Do Pulse High/Low Width of specified DO channel
Command $AA9nnhhhhvvvv(cr)
Syntax $ Command leading code
AA Module address ID (00-3F)
9 is the Read Do Pulse High/Low Width command
nn Digital Output channel number (0~17)
hhhh high pulse width in 0.5msec
vvvv low pulse width in 0.5msec
(cr) is the terminating character, carriage return (0Dh)
Response !AAcr) Valid command
?AA(cr) Invalid command
(cr) is the terminating character, carriage return (0Dh)
Example Command：$05902000100(cr)
Response：!05(cr)
! Valid command
05 Module ID
0200 high pulse width=0200(hex)=512(dec)*0.5msec=256 msec
0100 low pulse width=0100(hex)=256(dec)*05msec=128 msec

14.4.71 $AAAnn Read Single Do High/Low Delay Width

Description Read Do High/Low output Delay time of specified DO channel
Command $AAAnn(cr)
Syntax
$ Command leading code
AA Module address ID (00-3F)
A is the Read Do High/Low Delay time command
nn Digital Output channel number (0~17)
(cr) is the terminating character, carriage return (0Dh)
Response
!AAhhhhvvvv(cr) Valid command
hhhh high delay time in 0.5msec
vvvv low delay time in 0.5msec
?AA(cr) Invalid command
(cr) is the terminating character, carriage return (0Dh)
Example
Command：$05A02(cr)
Response：!0502000100(cr)
! Valid command
05 Module ID
0200 high output delay time=0200(hex)=512(dec)*0.5msec=256 msec
0100 low low output delay time=0100(hex)=256(dec)*05msec=128 msec

14.4.72 $AAAnnhhhhllll Set Single Do High/Low Delay Width

Description set Do High/Low output Delay time of specified DO channel
Command $AAAnnhhhhvvvv(cr)
Syntax
$ Command leading code
AA Module address ID (00-3F)
A is the Read Do Pulse High/Low Width command
nn Digital Output channel number (0~17)
hhhh high output delay time in 0.5msec
vvvv low output delay time in 0.5msec
(cr) is the terminating character, carriage return (0Dh)
Response
!AAcr) Valid command
?AA(cr) Invalid command
(cr) is the terminating character, carriage return (0Dh)
Example
Command：$05A02000100(cr)
Response：!05(cr)
! Valid command
05 Module ID
0200 high output delay time=0200(hex)=512(dec)*0.5msec=256 msec
0100 low output delay time=0100(hex)=256(dec)*05msec=128 msec

14.4.73 $AABnn Read Single Do Pulse Counts

Description Read Pulse Counts of single DO channel

Syntax $AABnn(cr)

$ is a delimiter character.
AA represents the 2-character hexadecimal module address
B is read pulse counts command.
nn represents DO channel number
(cr) is the terminating character, carriage return (0Dh).

Response !AAcccc(cr) if the command is valid.

?AA(cr) if an invalid operation was entered.
! Delimiter indicating a valid command was received.
? Delimiter indicating the command was invalid.
AA represents the 2-character hexadecimal address of an EX-9200 module.
cccc represents DO pulse counts
(cr) is the terminating character, carriage return (0Dh).

Example Read pulse counts of DO channel 03

Command：$01B03(cr)
Response：!010020(cr)
The pulse counts of DO channel 03 is 0020(hex) =32(dec)

14.4.74 #AA2nncccc Write Single Do Pulse Counts

Description Set Pulse Counts of Single DO channel

Syntax #AA2nncccc(cr)

is a delimiter character.
AA represents the 2-character hexadecimal module address
2 is set DO pulse counts command.
nn is DO channel number
cccc represents DO pulse output counts
(cr) is the terminating character, carriage return (0Dh).

Response !AA(cr) if the command is valid.

?AA(cr) if an invalid operation was entered.
! Delimiter indicating a valid command was received.
? Delimiter indicating the command was invalid.
AA represents the 2-character hexadecimal address of an EX-9200 module.
(cr) is the terminating character, carriage return (0Dh).

Example Set DO pulse output counts=32 of DO channels 5

Command：#012050020(cr)
Response：!01(cr)

14.4.75 #AA3nns Start/Stop DO Pulse Counts

Description Start/stop DO pulse output of single DO channel

Syntax #AA3nns(cr)

is a delimiter character.
AA represents the 2-character hexadecimal module address
3 is Start/stop DO pulse command.
nn is DO channel number
s s=0 represents stop pulse output, s=1 represents start pulse output
(cr) is the terminating character, carriage return (0Dh).

Response !AA(cr) if the command is valid.

?AA(cr) if an invalid operation was entered.
! Delimiter indicating a valid command was received.
? Delimiter indicating the command was invalid.
AA represents the 2-character hexadecimal address of an EX-9200 module.
(cr) is the terminating character, carriage return (0Dh).

Example Start pulse output of DO channels 5

Command：#013051(cr)
Response：!01(cr)

14.4.76 #AA3nnnnnnnn Start/Stop multiple DO Pulse Counts

Description Start/stop DO pulse output of multiple DO channel
Syntax #AA3nnnnnnnn(cr)
is a delimiter character.
AA represents the 2-character hexadecimal module address
3 is Start/stop DO pulse command.
nnnnnnnnnn is DO channel number start/stop bit
bit m=0 stop channel m pulse output, m=1 start channel m pulse output
(cr) is the terminating character, carriage return (0Dh).
Response !AA(cr) if the command is valid.
?AA(cr) if an invalid operation was entered.
! Delimiter indicating a valid command was received.
? Delimiter indicating the command was invalid.
AA represents the 2-character hexadecimal address of an EX-9200 module.
(cr) is the terminating character, carriage return (0Dh).
Example Start pulse output of DO channels 0,1,2,3,5,15 (0000802F)
Command：#0130000802F(cr)
Response：!01(cr)

14.4.77 ~AA4v Read The Power On/Safe Value

Description Read power-on/safe value
Syntax ~AA4v(cr)
~ Command leading code
AA Module address ID (00 to FF)
4 Command to set DO power-on/safe value
v v="P" set power-on value, v="S" set safe value
(cr) Carriage return
Response !AA(cr) if the valid command
? AA(cr) Invalid command
! Delimiter for valid command
? Delimiter for invalid command
(cr) Carriage return
Example Read power-on/safe value (ID=05)
Command：~054P(cr)
Response：!05500000014(cr) //power-on value=00000014 (DO channel 2, 4 on)

14.4.78 ~AA5v Set Current Do Value As Power On/Safe Value

Description Set current DO value as power-on/safe value
Syntax ~AA5v(cr)
~ Command leading code
AA Module address ID (00 to FF)
5 Command to set DO power-on/safe value
v v="P" set power-on value, v="S" set safe value
(cr) Carriage return
Response !AA(cr) if the valid command
? AA(cr) Invalid command
! Delimiter for valid command
? Delimiter for invalid command
(cr) Carriage return
Example Set current DO value as power-on/safe value (ID=04)
Command：~045P(cr)
Response：!04(cr)

14.4.79 ~AA5vnnnnnn Set Specified Value As Power On/Safe Value

Description Set DO power-on/safe value

Syntax ~AA5vnnnnnn(cr)

~ Command leading code
AA Module address ID (00 to FF)
5 Command to set DO power-on/safe value
v v="P" set power-on value, v="S" set safe value
nnnnnn represents DO power-on/safe value
(cr) Carriage return

Response !AA(cr) if the valid command
? AA(cr) Invalid command
! Delimiter for valid command
? Delimiter for invalid command
(cr) Carriage return

Example Set DO power-on value to 0xFF00FF(ID=04)
Command：~045PFF00FE(cr)
Response：!04(cr)

Example Set DO safe value to 0xFF00FF(ID=04)
Command：~045SFF00FE(cr)
Response：!04(cr)

14.4.80 ~AAD Read DI/O Active State

Description Read input/output active status.

Syntax ~AAD(cr)

~ Command leading code
AA Module address ID (00 to FF)
D Command for reading digital input active status
(cr) Carriage return

Response !AAmn(cr) if the valid command
? AA(cr) if the invalid command
! Delimiter for valid command
? Delimiter for invalid command
m Input active status, m=0 input low voltage/open active, m=1 Input high voltage active (See *)
n Output active status, n=1 output short/close active, n=0 open active (See **)
(cr) Carriage return

Example Read output active status of EX5060 (ID=05)
Command：~05D(cr)
Response：!0501(cr)
01 All input channels are low active and all output channels are short/close active

Note (*) : m is only available for the module which has digital input channels
(**) : n is only available for the module which has digital output channels

14.4.81 ~AADvn Set DI/O Active State

Description Set input/output active status.
Command ~AADvn[CHK](cr)
Syntax
~ Command leading code
AA Module address ID (00 to FF)
D Command for setting digital input active status
v Input active status, v=0 input low voltage/open active, v=1 Input high voltage active (See *)
n Output active status, n=1 output short/close active, n=0 open active (See **)
(cr) is the terminating character, carriage return (0Dh)
Response
!AA (cr) Valid command
?AA(cr) Invalid command
(cr) is the terminating character, carriage return (0Dh)
Example
Command：~05D01(cr)
Response：!05(cr)
05 Module ID
0 Set all inputput channels to high volt/open active
1 Set all output channels to high/open output active
Note:
(*) ：v is only available for the module which has digital input channels
(**) ：n is only available for the module which has digital output channels

14.4.82 ~AASDBv Set DI debounce mode

Description The command set digital input channel debounce mode (pre-read or post-read mode)
Syntax ~AASDBv(cr)
~ Is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of module.
SDB is the Set DI debounce mode command
v v=0 pre-read mode ,v=1 post-read mode
(cr) Is the terminating character, carriage return (0Dh).
Response !AA(cr) If the command is valid or ?AA (cr) if the command is invalid
There is no response if the module detects a syntax error or communication error.
! Is a delimiter character.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
(cr) Is the terminating character, carriage return (0Dh)
Example
Command：~01SDB1(cr)
Response：!01(cr)
The command Set DI debounce mode to post read mode

14.4.83 ~AARDB Readet DI debounce mode

Description The command readt digital input channel debounce mode (pre-read or post-read mode)
Syntax ~AARDB(cr)
~ Is a delimiter character.
AA (range 00-FF) represents the 2-character hexadecimal address of module.
RDB is the Set DI debounce mode command
(cr) Is the terminating character, carriage return (0Dh).
Response !AAv(cr) If the command is valid or ?AA (cr) if the command is invalid
There is no response if the module detects a syntax error or communication error.
! Is a delimiter character.
AA (range 00-3F) represents the 2-character hexadecimal address of module.
v v=0 pre-read mode ,v=1 post-read mode
(cr) Is the terminating character, carriage return (0Dh)
Example
Command：~01RDB(cr)
Response：!011(cr)
The command DI debounce mode is post read mode

Chapter 15 E5KDAQ.DLL API

15.1 Common Functions

Function Name	Description	Sec.
E5K_SearchModules	Search all EX9200 modules	15.4
E5K_OpenModuleUSB	Open module with ID address from USB interface	15.5
E5K_OpenModuleIP	Open module with the module IP address from Ethernet	15.6
E5K_OpenModuleIPEx	Open module with the module IP address from Ethernet and specified HOST IP address (For multiple NIC cards)	15.7
E5K_OpenModuleCOM	Open module with ID address from COM port	15.8
E5K_CloseModules	Close all modules	15.9
E5K_GetDLLVersion	Get E5KDAQ.DLL version	15.10
E5K_VerifyPassWord	Verifies Pass word (Ethernet only)	15.11
E5K_ChangePassWord	Change password (Ethernet connection only)	15.12
E5K_GetLastErrorCode	Get last DLL error code	15.13
E5K_SetRXTimeOutOption	Set receive/send Timeout	15.14
E5K_StartAlarmEventIP	Start alarm event from the default Host IP (Ethernet only)	15.15
E5K_StartAlarmEventIPEx	Start Taregt alarm event from the specified Host IP (For multiple NIC cards)	15.16
E5K_StopAlarmEventIP	Stop alarm event from IP (Ethernet only)	15.17
E5K_StartAlarmEventUSB	Start alarm event from USB (USB only)	15.18
E5K_StopAlarmEventUSB	Stop alarm event from USB (USB only)	15.19
E5K_ReadAlarmEventData	Read alarm event data	15.20
E5K_StartStreamEvent	Start stream event from from the default Host IP (Ethernet only)	15.21
E5K_StartStreamEventEx	Start Taregt stream event from the specified Host IP (For multiple NIC cards)	15.22
E5K_StopStreamEvent	Stop stream event from IP (Ethernet only)	15.23
E5K_ReadStreamEventData	Read stream data from IP (Ethernet only)	15.24
E5K_ReadModuleConfig	Read module configuration	15.25
E5K_SetModuleConfig	Set module configuration	15.26
E5K_WriteModbusDiscrete	Write Modbus discrete(coil/input) data to the specified module	15.27
E5K_WriteModbusRegister	Write Modbus register(holding/input) data to the specified module	15.28
E5K_ReadModbusRegister	Read Modbus (holding/input)register data from the specified module	15.29
E5K_ReadModbusDiscrete	Read Modbus (coil/input)discrete data from the specified module	15.30
E5K_SendASCRequestAndWaitResponse	Send ASCII command to and wait response from the specified module	15.31
E5K_RecvASCII	Receive ASCII data from the specified module	15.32
E5K_SendASCII	Send ASCII command to specified module	15.33
E5K_SendHEXRequestAndWaitResponse	Send binary data to and wait response from the specified module	15.34
E5K_SendHEX	Send HEX command to specified module	15.35
E5K_RecvHEX	Receive Hex data from the specified module	15.36
E5K_CalculateCRC16	Calculate CRC16	15.37
E5K_SetLEDControl	Set LED control mode	15.38
E5K_WriteDataToLED	Write data to LED board	15.39
E5K_FlashLED	Force on-board LED to flash	15.40
E5K_IsValidIPAddress	Check IP address is valid or not	15.41
E5K_IsIPInLocalSubnet	Check Target IP in the default Host IP subnet	15.42
E5K_IsIPInLocalSubnetEx	Check Target IP in the specified Host IP subnet (For multiple NIC cards)	15.43
E5K_GetLocalIP	Get Host IP address	15.44
E5K_TCPConnect	Build a TCP connection from default Host IP address	15.45
E5K_TCPConnectEx	Build a TCP connection from the specified Host IP (For multiple NIC cards)	15.46
E5K_TCPSendData	Send data to TCP connection	15.47
E5K_TCPRecvData	Receive data from TCP connection	15.48
E5K_TCPPing	Ping the specified IP address	15.49
E5K_TCPDisconnect	Disconnect specified TCP connection	15.50
E5K_TCPAllDisconnect	Disconnect all TCP connections	15.51
E5K_UDPConnect	Build an UDP connection from default Host IP address	15.52
E5K_UDPConnectEx	Build an UDCP connection from the specified Host IP (For multiple NIC cards)	15.53
E5K_UDPSendData	Send binary data to UDP connection	15.54
E5K_UDPRecvData	Receive binary data from UDP connection	15.55
E5K_UDPSendASCStr	Send an ASCII string to UDP connection	15.56
E5K_UDPRecvASCStr	Receive an ASCII string from UDP connection	15.57
E5K_UDPDisconnect	Disconnect specified UDP connection	15.58
E5K_UDPAllDisconnect	Disconnect all UDP connections	15.59

15.2 Analog Functions

Function name	Description	Sec.
E5K_ReadAIChannelType	Read type of the specified AI channel	15.51
E5K_SetAIChannelType	Set type of the specified AI channel	15.61
E5K_SetSingleChannelColdJunctionOffset	Set CJC offset of the specified AI channel	15.62
E5K_ReadSingleChannelColdJunctionOffset	Read CJC offset of a specified AI channel	15.63
E5K_ReadMultiChannelColdJunctionOffset	Read CJC offset of the multiple AI channels	15.64
E5K_SetMultiChannelColdJunctionOffset	Set CJC offset of multiple AI channels	15.65
E5K_ReadColdJunctionTemperature	Read cold junction temperature	15.66
E5K_ReadColdJunctionStatus	Read cold junction enable/disable status	15.67
E5K_SetColdJunction	Enable/disable CJC	15.68
E5K_ReadAIChannelConfig	Read configuration of the specified AI channel	15.69
E5K_SetAIChannelConfig	Set configuration of the specified AI channel	15.70
E5K_ReadAIBurnOutStatus	Read burnout status of analog channels	15.71
E5K_ReadAIAlarmStatus	Read AI alarm event status	15.72
E5K_SetAIBurnOut	Enable/disable burnout detection	15.73
E5K_ReadAIBurnOut	Read AI burnout detection enable/disable status	15.74
E5K_SetAIModuleFilter	Set AI filter frequency	15.75
E5K_ReadAIModuleFilter	Read AI filter frequency	15.76
E5K_SetAIChannelEnable	Enable/disable AI channel	15.77
E5K_ReadAIChannelEnable	Read enable/disable status of AI channels	15.78
E5K_ReadAINormalMultiChannel	Read AI normal value of multiple AI channels	15.79
E5K_ReadAIMaximunMultiChannel	Read AI maximum value of multiple AI channels	15.80
E5K_ReadAIMinimunMultiChannel	Read AI minimum value of multiple AI channels	15.81
E5K_ResetAIMaximun	Reset Maximum value of the specified AI channels	15.82
E5K_ResetAIMinimun	Reset Minimum value of the specified AI channels	15.83
E5K_ResetAIHighAlarm	Reset high alarm flag of the specified AI channels	15.84
E5K_ResetAILowAlarm	Reset low alarm flag of the specified AI channels	15.85
E5K_ReadAIChannelAverage	Read in average status of AI channels	15.86
E5K_SetAIChannelAverage	Enable AI channels in average	15.87

15.3 DIO Functions

Function Name	Description	Sec.
E5K_SetDIChannelConfig	Set configuration of the specified DI channel	15.88
E5K_ReadDIChannelConfig	Read configuration of the specified DI channel	15.89
E5K_ReadDIStatus	Read DI status	15.90
E5K_ReadDILatch	Read DI latch status	15.91
E5K_ClearAllDILatch	Clear all DI latch status	15.92
E5K_ClearSingleDICounter	Clear counter value of the specified DI channel	15.93
E5K_ReadMultiDICounter	Read multiple DI counter value	15.94
E5K_WriteDO	Write DO channels	15.95
E5K_ReadDOStatus	Read DO status	15.96
E5K_SetDOSingleChannel	Set/reset single DO channel	15.97
E5K_SetDOPulseWidth	Set DO pulse high/low width	15.98
E5K_ReadDOPulseWidth	Read DO pulse high/low width	15.99
E5K_StartDOPulse	Start DO pulse output	15.100
E5K_StopDOPulse	Stop DO pulse output	15.102
E5K_ReadDOPulseCount	Read back DO pulse count value	15.103
E5K_SetDOPowerOnValue	Set DO power on value	15.104
E5K_ReadDOPowerOnValue	Read DO power on value	15.105
E5K_ReadDIOActiveLevel	Read DI/DO active flag setting	15.106
E5K_SetDIOActiveLevel	Set DI/O channel active flag	15.107

15.4 E5K_SearchModules

Description Search all connected EX-9200 modules

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
```
Declare/public Function E5K_SearchModules Lib "E5KDAQ.dll"
      (Pd as E5K_DEVICE_ID_INFO, interface_type as integer) as integer
```
VC++/BC++Builder: (see E5KDAQ.h)
```
Unsigned short E5K_SearchModules (E5K_DEVICE_ID_INFO *pd,unsigned int
                  interface_type );
```

Parameters

Pd	points to a structure E5K_DEVICE_ID_INFO
Interface_type	indicate what connection be used for searching (see E5KDAQ.h)

Return Code Return how many modules be found, It no module existed, if return with 0

15.5 E5K_OpenModuleUSB

Description Open module by its ID address from USB interface

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
```
Declare/public Function E5K_OpenModuleUSB Lib "E5KDAQ.dll"
                  (ByVal id As Integer) As Integer
```
VC++/BC++Builder: (see E5KDAQ.h)
```
Unsigned short      E5K_OpenModuleUSB (unsigned short id);
```

Parameters Id module ID address

Return Code Return the same ID number as parameter id, if open success Return -1 open error

15.6 E5K_OpenModuleIP

Description Open module by IP address feom default Host IP (for Ethernet connection only)

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
```
Declare/public Function    E5K_OpenModuleIP Lib "E5KDAQ.dll" _
                           (ByVal IP As String,Byval
                           Byval ConnecttimeOut as long,
                           Byval RxTotalTimeOut as long,
                           Byval RxTimeoutInterval as long) As Integer
```
VC++/BC++: (see E5KDAQ.h)
```
Unsigned short      E5K_OpenModuleIP (
                    Char IP[],
                    unsigned long ConnectTimeOut,
                    unsigned long RxTotalTimeOut,
                    unsigned long RxTimeoutInterval);
```

Parameters

IP	points to an IP address array (such as "192.168.0.12")
ConnectTimeOut	Connection timeout interval (msec)
RxTotalTimeOut	Receive frame timeout interval (msec)
RxTimeOutInterval	receive character timeout interval (msec)

Return Code Return the ID address of module, if open success

15.7 E5K_OpenModuleIPEx

Description Open module by IP address from specified Host IP (for Ethernet connection only)

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_OpenModuleIPEx Lib "E5KDAQ.dll" _
                          (ByVal IP As String,Byval
                          Byval ConnecttimeOut as long,
                          Byval RxTotalTimeOut as long,
                          Byval RxTimeoutInterval as long,
                          Byval HostIP as string) As Integer
```

VC++/BC++: (see E5KDAQ.h)

```
Unsigned short      E5K_OpenModuleIPEx (
                    Char IP[],
                    unsigned long ConnectTimeOut,
                    unsigned long RxTotalTimeOut,
                    unsigned long RxTimeoutInterval,
                    char HostIP[]);
```

Parameters

IP	points to tatget IP address array (such as "192.168.0.12")
ConnectTimeOut	Connection timeout interval (msec)
RxTotalTimeOut	Receive frame timeout interval (msec)
RxTimeOutInterval	receive character timeout interval (msec)
HostIP	points to specifed Host IP (for multiple NIC cards)

Return Code If open success, return the ID address of module, otherwise -1

15.8 E5K_OpenModuleCOM

Description Open module by its ID address from COM port

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_OpenModuleCOM Lib "E5KDAQ.dll" _
                          ( Byval devid As Integer,
                            Byval comport As Integer,
                            ByVal RxTotalTimeOut As Long,
                            ByVal RxTimeoutInterval As Long,
                            ByVal BaudRate As Long,
                            ByVal ChksumCRC As Byte) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
int         E5K_OpenModuleCOM(
            unsigned int devid,
            unsigned int comport,
            unsigned long RxTotalTimeOut,
            unsigned long RxTimeoutInterval,
            unsigned long Baudrate,
            unsigned char ChksumCRC);
```

Parameters

devId	module ID address
comport	COM port number
RxTotalTimeOut	Receive time out between characters(msec)
RxTimeoutInterval	Receive total timeout (msec)
Baudrate	Baud rate
ChksumCRC	Enable/disable Check sum/ CRC (1=Enabled,0=disable)

Return Code If open success, return the ID address of module, otherwise -1

15.9 E5K_CloseModules

Description Close all modules

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
Declare/public Function E5K_CloseModules Lib "E5KDAQ.dll" () As Integer

VC++: (see E5KDAQ.h)
Unsigned short E5K_CloseModules (void);

Parameters none no parameters

Return Code refers to the *Error code*.

15.10 E5K_GetDLLVersion

Description Get version of E5KDAQ.DLL

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
Declare/public Function E5K_GetDLLVersion Lib "E5KDAQ.dll" _
(ByRef Major As Integer, ByRef Minor As Integer) As Integer

VC++: (see E5KDAQ.h)
Unsigned short E5K_GetDLLVersion (unsigned int *Major, unsigned int *Minor);

Parameters
Major points' version major buffer
Minor point's version minor buffer

Return Code refers to the *Error code*.

15.11 E5K_VerifyPassWord

Description Verify password of the Ethernet connected module. The function should be called after calling *E5K_OpendModuleIP* () function (for Ethernet Connection only)

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
Declare/public Function E5K_VerifyPassWord Lib "E5KDAQ.dll" _
(ByVal id As Integer,
ByVal Password as String,
ByVal length As Integer) As Integer

VC++: (see E5KDAQ.h)
Unsigned short E5K_VerifyPassWord (int id , UNSIGNED CHAR Password[], unsigned int length);

Parameters
Id module ID address
Password points to password string buffer
Length password length

Return Code Refer to the *Error code*.

15.12 E5K_ChangePassWord

Description Change password of the Ethernet connected module. The function is available after calling E5K_VerifyPassWord function (for Ethernet Connection only)

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_ChangePassWord Lib "E5KDAQ.dll" _
                          (ByVal id As Integer, _
                          ByVal OldPassword as String, _
                          ByVal Oldlength as Integer, _
                          ByVal NewPassword as String, _
                          ByVal Newlength as Integer) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short  E5K_ChangePassWord (
                int id,
                unsigned char oldPassWord[],
                unsigned int oldlength,
                unsigned char newPassWord[],
                unsigned int newlength);
```

Parameters

id	module ID address
oldPassWord[]	points to password string buffer
oldlength	old password length
newPassWord[]	points to new password,
newlength	new password length

Return Code

15.13 E5K_GetLastErrorCode

Description Get E5KDAQ.dll last error code

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_GetLastErrorCode Lib "E5KDAQ.dll" () As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_GetLastErrorCode (void);
```

Parameters none — no parameters

Return Code

15.14 E5K_SetRXTimeOutOption

Description Set receive total timeout and interval timeout of COM port and TCP/IP

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function E5K_SetRXTimeOutOption Lib "E5KDAQ.dll" _
(ByVal RxTotalTimeout as Long, ByVal RxChrTimeOutInterval as Long) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short    E5K_SetRXTimeOutOption (long RxTotalTimeout, long
                  RxChrTimeOutInterval);
```

Parameters

RxTotalTimeout	receive total timeout (msec)
Length	receive character interval timeout (msec)

Return Code Refer to the *Error code*.

15.15 E5K_StartAlarmEventIP

Description Start target alarm event from the default Host IP

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_StartAlarmEventIP Lib "E5KDAQ.dll"
                           (ByVal IPaddress As string) As long
```

VC++: (see E5KDAQ.h)

```
long          E5K_StartAlarmEventIP (char * IPaddress);
```

Parameters

IPaddress	Alarm event source IP address

Return Code Event handle or -1 for error (Refer to the *Error code.)*

15.16 E5K_StartAlarmEventIPEx

Description Start target alarm event from the specified Host IP (for multiple NIC cards)

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_StartAlarmEventIPEx Lib "E5KDAQ.dll"
                           (ByVal IPaddress As string, ByVal HostIP As string) As long
```

VC++: (see E5KDAQ.h)

```
long          E5K_StartAlarmEventIPEx (char * IPaddress, char * HostIP);
```

Parameters

IPaddress	Alarm event source IP address
HostIP	Specifies Host IP

Return Code Event handle or -1 for error (Refer to the *Error code.)*

15.17 E5K_StopAlarmEventIP

Description Stop alarm event from IP

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_StopAlarmEventIP Lib "E5KDAQ.dll"
                           (ByVal IPaddress As string) As Integer
```

VC++: (see E5KDAQ.h)

```
Int            E5K_StopAlarmEventIP (char * IPaddress);
```

Parameters IPaddress Alarm event source IP address

Return Code Refer to the *Error code*.

15.18 E5K_StartAlarmEventUSB

Description Start alarm event from USB connection

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_StartAlarmEventUSB Lib "E5KDAQ.dll"
                           (ByVal Id As integer) As long
```

VC++: (see E5KDAQ.h)

```
long       E5K_StartAlarmEventUSB (int Id);
```

Parameters Id Alarm event source Id

Return Code Refer to the *Error code*.

15.19 E5K_StopAlarmEventUSB

Description Stop alarm event from USB connection

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_StopAlarmEventUSB Lib "E5KDAQ.dll"
                           (ByVal Id As integer) As Integer
```

VC++: (see E5KDAQ.h)

```
Int        E5K_StopAlarmEventUSB (int Id);
```

Parameters Id Alarm event source Id

Return Code Refer to the *Error code*.

15.20 E5K_ReadAlarmEventData

Description Read alarm event information from device

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
Declare/public Function E5K_ReadAlarmEventData Lib "E5KDAQ.dll"
(Intinfo as ALARM_EVENT_INF) As Integer

VC++: (see E5KDAQ.h)
Int E5K_ReadAlarmEventData (struct ALARM_EVENT_INF *Intinfo);

Parameters Intinfo Points to an Alarm event structure (ALARM_EVENT_INF)

Return Code Refer to the *Error code*.

15.21 E5K_StartStreamEvent

Description Start target stream event from the default Host IP

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
Declare/public Function E5K_StartStreamEvent Lib "E5KDAQ.dll"
(ByVal IPaddress As string) As long

VC++: (see E5KDAQ.h)
long E5K_StartStreamEventIP (char * IPaddress);

Parameters IPaddress Stream data event source IP address

Return Code Event handle or -1 for error (Refer to the *Error code.)*

15.22 E5K_StartStreamEventEx

Description Start target stream event from the specified Host IP (for multiple NIC cards)

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
Declare/public Function E5K_StartStreamEventEx Lib "E5KDAQ.dll"
(ByVal IPaddress As string, Byval HostIP) As long

VC++: (see E5KDAQ.h)
long E5K_StartStreamEventIPEx (char * IPaddress, char * HostIP);

Parameters IPaddress Stream data event source IP address
HostIP Specifies Host IP (for multiple NIC cards)

Return Code Event handle or -1 for error (Refer to the *Error code.)*

15.23 E5K_StopStreamEvent

Description Stop to receive stream data from IP

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
Declare/public Function E5K_StopStreamEvent Lib "E5KDAQ.dll"
(ByVal IPaddress As string) As Integer

VC++: (see E5KDAQ.h)
Int E5K_StoptStreamEventIP (char * IPaddress);

Parameters IPaddress Stream data event source IP address

Return Code refers to the *Error code.*

15.24 E5K_ReadStreamEventData

Description Read stream data from IP

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
Declare/public Function E5K_ReadStreamEventData Lib "E5KDAQ.dll"
(StreamIntInfo As STREAM_EVENT_INFO) As Integer

VC++: (see E5KDAQ.h)
Int E5K_ReadStreamEventDa (STREAM_EVENT_INFO StreamIntInfo[]);

Parameters StreamIntInfo Points to a stream data buffer
(structure STREAM_EVENT_INFO) (see E5KDAQ.H)

Return Code Refer to the *Error code.*

15.25 E5K_ReadModuleConfig

Description Read module configuration

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
Declare/public Function E5K_ReadModuleConfig Lib "E5KDAQ.dll" _
(ByVal id As Integer, mp as MODULE_CONFIG) As integer

VC++: (see E5KDAQ.h)
Unsigned short E5K_ReadModuleConfig (unsigned int id, MODULE_CONFIG *mp);

Parameters id module ID address
mp points to a structure MODULE_CONFIG (see E5KDAQ.H)

Return Code Refer to the *Error code.*

15.26 E5K_SetModuleConfig

Description Set module configuration

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_SetModuleConfig Lib "E5KDAQ.dll" _
                           (ByVal id As Integer , mp as MODULE_CONFIG) As integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_SetModuleConfig (unsigned int id, MODULE_CONFIG *mp);
```

Parameters

id	module ID address
mp	points to a structure MODULE_CONFIG (see E5KDAQ.H)

Return Code Refer to the *Error code*.

15.27 E5K_WriteModbusDiscrete

Description Write data to Modbus discrete (Modbus coil)

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_WriteModbusDiscrete Lib "E5KDAQ.dll" _
                           (ByVal id As Integer, ByVal startaddr As Integer,  _
                           ByVal counts As Integer, Discrete As Byte) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_WriteModbusDiscrete (
                    int id,
                    unsigned int startaddr,
                    unsigned int counts,
                    unsigned char Discrete[]);
```

Parameters

id	module ID address
Startaddr	start address in Modbus coil (0000~FFFF)
Counts	how many bit be written
Discrete[]	data buffer be written

Discrete[n]=0 or 1 for Modbus coil address startaddr+n (0<=n<counts)

Return Code Refer to the *Error code*.

15.28 E5K_WriteModbusRegister

Description Write data to Modbus Holding registers

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_WriteModbusRegister Lib "E5KDAQ.dll" _
                          (ByVal id As Integer, ByVal startaddr As Integer,  _
                          ByVal counts As Integer, regs As Integer) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_WriteModbusRegister (
                    Int id,
                    unsigned int startaddr,
                    unsigned int counts, Sunsigned int regs[]);
```

Parameters

id	module ID address
Startaddr	start address in Modbus Holding register(0000~FFFF)
Counts	how many register be written
regs[]	data buffer be written

regs[n]=value for Modbus holding register address startaddr+n (0<=n<counts)

Return Code Refer to the *Error code*.

15.29 E5K_ReadModbusRegister

Description Read Modbus Holding or Input registers

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_ReadModbusRegister Lib "E5KDAQ.dll" _
                          (ByVal id As Integer, ByVal startaddr As Integer, _
                          ByVal counts As Integer, regs As Integer) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_ReadModbusRegister (
                    int id,
                    unsigned int startaddr,
                    unsigned int counts,
                    int regs[]);
```

Parameters

id	module ID address
Startaddr	start address in Modbus Holding or Input register (0000~FFFF)
Counts	how many register be read
Regs []	points data buffer

Return Code Refer to the *Error code*.

15.30 E5K_ReadModbusDiscrete

Description Read Modbus coil or discrete input

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_ReadModbusDiscrete Lib "E5KDAQ.dll" _
                          (ByVal id As Integer,   _
                          ByVal startaddr As Integer, _
                          ByVal counts As Integer,   _
                          Discrete As Byte _
                          ) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_ReadModbusDiscrete(
                    int id,
                    unsigned int startaddr,
                    unsigned int counts,
                    unsigned char Discrete[]);
```

Parameters

id	module ID address
Startaddr	start address in Modbus Holding or Input register (0000~FFFF)
Counts	how many bit be read
Discrete []	points data buffer Discrete[n]=0 or 1 of Modbus coil or discrete input address startaddr+n (0<=n<counts)

Return Code

15.31 E5K_SendASCRequestAndWaitResponse

Description Send an ASCII string to and wait for response from module

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_SendASCRequestAndWaitResponse Lib "E5KDAQ.dll" _
                          (ByVal id As Integer, ByVal asccmd As String,   _
                          ByVal response As String,
                          ByVal rxbuffersize As Integer) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_SendASCRequestAndWaitResponse(
                    int id,
                    unsigned char asccmd[],
                    unsigned char response[],
                    unsigned int rxbuffersize);
```

Parameters

id	module ID address
Asccmd	points to ASCII string buffer
Response	points response buffer
Rxbuffersize	size of response buffer

Return Code Refer to the *Error code*.

15.32 E5K_RecvASCII

Description Receive an ASCII string from the module

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_RecvASCII    Lib "E5KDAQ.dll" _
                           (ByVal id As Integer, _
ByVal Rxbuffer as String, _
ByVal bufferSize as integer) as integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_RecvASCII (int id, char Rxbuffer [], unsigned int BufferSize);
```

Parameters

id	module ID address
Asccmd	points to ASCII string buffer
Rxbuffer	points receive buffer
Buffersize	size of revive buffer

Return Code Refer to the *Error code*.

15.33 E5K_SendASCII

Description Send an ASCII string to the module

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_SendASCII    Lib "E5KDAQ.dll" _
                           (ByVal id As Integer, ByVal Txstring as String) as integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_SendASCII (int id, char Txbuffer []);
```

Parameters

id	module ID address
TX string	points ASCII string buffer

Return Code Refer to the *Error code*.

15.34 E5K_SendHEXRequestAndWaitResponse

Description Send binary data to and wait for response from module

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_SendHEXRequestAndWaitResponse Lib "E5KDAQ.dll" _
                          (ByVal id As Integer, _
                          ByRef Txdata As Byte,  _
                          ByVal Rxlen As Integer,  _
                          ByRef Rxdata As Byte, _
                          ByRef Rxlen As Integer, _
                          ByVal RxBufsize As Integer _
                          ) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_SendHEXRequestAndWaitResponse(
                    int id ,
                    unsigned char cTxData[],
                    unsigned int wTxlen,
                    unsigned char cRxdata[],
                    unsigned int *wRxlen,
                    unsigned int buffersize);
```

Parameters

id	module ID address
cTxData []	points to binary data buffer
wTxlen	how many bytes be sent
cRxdata []	points response buffer
*wRxlen	point to buffer to store the number of byte received
Buffersize	size of response buffer

Return Code Refer to the *Error code*.

15.35 E5K_SendHEX

Description Send a Hex data to the module

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function E5K_SendHex Lib "E5KDAQ.dll" _
(ByVal id As Integer, Byref Hexdata as Byte, ByVal Datalen as Integer) as integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_SendHex (int id, char Hexdata[],unsigned int Datalen);
```

Parameters

id	module ID address
Hexdata	points Hex data buffer
Datalen	size of the data buffer

Return Code Refer to the *Error code*.

15.36 E5K_RecvHEX

Description Receive a Hex data from the module

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_RecvHex    Lib "E5KDAQ.dll" _
                          (ByVal id As Integer, _
                          Byref Rxbuffer as String, _
                          ByVal bufferSize as integer) as integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_RecvHex (int id, char Rxbuffer [], unsigned int BufferSize);
```

Parameters

id	module ID address
Asccmd	points to ASCII string buffer
Rxbuffer	points receive buffer
Buffersize	size of revive buffer

Return Code Refer to the *Error code*.

15.37 E5K_CalculateCRC16

Description Calculate CRC16

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_CalculateCRC16 Lib "E5KDAQ.dll" _
 (bData as byte , byval wLen as integer ,byref wCRC as integer) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_CalculateCRC16 (
                    unsigned char bData[],
                    unsigned int wLen,
                    unsigned int *wCRC);
```

Parameters

bData[]	points to binary data buffer
wLen	size of data
wCRC	points to buffer to store CRC16 value

Return Code Refer to the *Error code*.

15.38 E5K_SetLEDControl

Description Set on-board control mode (controlled by Module or by user AP)

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_SetLEDControl Lib "E5KDAQ.dll" _
                          (ByVal id As Integer, ByVal ControlOption as Integer) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short            E5K_SetLEDControl (int id, char Control Option);
```

Parameters

id	module ID address
ControlOption	On-board LED control mode. 0: controlled by module, 1: controlled by user AP

Return Code Refer to the *Error code*.

15.39 E5K_WriteDataToLED

Description Write data to on-board LED

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
```
Declare/public Function    E5K_WriteDataToLED Lib "E5KDAQ.dll" _
                           (ByVal id As Integer, ByVal LedData) As Integer
```
VC++: (see E5KDAQ.h)
```
Unsigned short             E5K_WriteDataToLED (int id, unsigned long LedData);
```

Parameters
id module ID address
LedData Data to be written to on-board LED. bit #n=0: LED #n off, bit n=1: LED #n on

Return Code Refer to the *Error code*.

15.40 E5K_FlashLED

Description Force on-board LED to flash

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
```
Declare/public Function E5K_FlashLED Lib "E5KDAQ.dll" _
(ByVal id As Integer, ByVal LedMask As Integer, FlashCount As Long) As Integer
```
VC++: (see E5KDAQ.h)
```
Unsigned short E5K_FlashLED (int id, unsigned long LedMask,unsigned int
                    FlashCounts);
```

Parameters
id module ID address
LedMask LED channel mask. bit #n=0: No Flash LED #n, bit n=1: Flash LED #n
FlashCounts Flash counts

Return Code Refer to the *Error code*.

15.41 E5K_IsValidIPAddress

Description Check the validity of the specified IP address

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
```
Declare/public Function    E5K_IsValidIPAddress Lib "E5KDAQ.dll" _
                           (ByVal zIP As String) As Integer
```
VC++: (see E5KDAQ.h)
```
Unsigned short             E5K_IsValidIPAddress (char *zIP);
```

Parameters zIP IP address string (such as 192.168.0.21)

Return Code return 0 if IP is in the same subnet , otherwise not in the same subnet (Refer to the *Error code.)*

15.42 E5K_ IsIPInLocalSubnet

Description Is the specified IP address in the default Host IP subnet

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_ IsIPInLocalSubnet    Lib "E5KDAQ.dll" _
                           (ByVal zIP As String) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short             E5K_IsValidIPAddress (char *zIP);
```

Parameters zIP IP address string (such as 192.168.0.21)

Return Code return 0 if IP is in the same subnet , otherwise not in the same subnet (Refer to the *Error code.)*

15.43 E5K_ IsIPInLocalSubnetEx

Description Is the specified IP address in the specified Host IP subnet (for multiple NIC cards)

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_ IsIPInLocalSubnetEx    Lib "E5KDAQ.dll" _
                           (ByVal zIP As String, ByVal HostIP As String) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short         E5K_IsValidIPAddress (char *zIP,char *HostIP);
```

Parameters

zIP	IP address string (such as 192.168.0.21)
HostIP	Specifies Host IP

Return Code return 0 if IP is in the same subnet , otherwise not in the same subnet (Refer to the *Error code.)*

15.44 E5K_GetLocalIP

Description Get host IP address

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_GetLocalIP Lib "E5KDAQ.dll" _
                           (ip0 as byte,ip1 as byte,ip2 as byte,ip3 as byte) as integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short         E5K_GetLocalIP (char *ip0, char *ip1, char *ip2, char *ip3);
```

Parameters

Ip0	first IP address byte for an EX-9200 that to be connected
Ip1	second IP address byte for an EX-9200 that to be connected
Ip2	third IP address byte for an EX-9200 that to be connected
Ip3	forth IP address byte for an EX-9200 that to be connected

Return Code Refer to the *Error code.*

15.45 E5K_TCPConnect

Description Establish a TCP connection by default Hos IP

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function     E5K_TCPConnect Lib "E5KDAQ.dll" _
(ByVal zIP As String, ByVal port As Integer, ByVal iConnectionTimeout As Integer,
ByVal iSendTimeout As Integer, ByVal iReceiveTimeout As Integer) As long
```

VC++/BC++Builder: (see E5KDAQ.h)

```
SOCKET              E5K_TCPConnect (
                    char szIP [], u_short port, int iConnectionTimeout,
                    int iSendTimeout, int iReceiveTimeout);
```

Parameters

szIP	Target IP address
port	connection port
iConnectionTimeout	connection timeout value(msec)
iSendTimeout	send timeout value(msec)
iReceiveTimeout	receive timeout value(msec)

Return Code return socket handle or 0 for error (call E5K_GetLastError() to read error code)

15.46 E5K_TCPConnectEx

Description Establish a TCP connection by the specified Host IP

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function     E5K_TCPConnectEx Lib "E5KDAQ.dll" _
(ByVal zIP As String, ByVal port As Integer, ByVal iConnecTimeout As Integer, ByVal
iTxTimeout As Integer, ByVal iRxTimeout As Integer,Byval HostIP as string) As long
```

VC++/BC++Builder: (see E5KDAQ.h)

```
SOCKET              E5K_TCPConnect Ex(
                    char szIP [], u_short port, int iConnectionTimeout,
                    int iSendTimeout, int iReceiveTimeout,char HostIP[]);
```

Parameters

szIP	Target IP address
port	connection port
iConnectTimeout	connection timeout value(msec)
iTxTimeout	send timeout value(msec)
iRxTimeout	receive timeout value(msec)
Hostip[]	points to Host IP address buffer (such as "192.168.0.10")

Return Code return socket handle or 0 for error (call E5K_GetLastError() to read error code)

15.47 E5K_TCPSendData

Description Send data to TCP connection

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function     E5K_TCPSendData Lib "E5KDAQ.dll" _
(ByVal sock As Long, ByRef pdata As Byte, ByVal datalen As Integer) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short      E5K_TCPSendData (SOCKET sock, SBYTE *pdata, u_short datalen);
```

Parameters

sock	TCP socket handle
pdata	Points to data buffer
datalen	bytes of data

Return Code Refer to the *Error code*.

15.48 E5K_TCPRecvData

Description Receive data from TCP connection

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function     E5K_TCPRecvData Lib "E5KDAQ.dll" _
(ByVal sock As Long, ByRef pdata As Byte, ByVal psize As Integer) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short      E5K_TCPRecvData (SOCKET sock, char *pdata, unsigned short psize);
```

Parameters

sock	TCP socket handle
pdata	Points to data buffer
psize	size of data buffer

Return Code Bytes of data received

15.49 E5K_TCPPing

Description Ping Specified IP address

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function     E5K_TCPPing Lib "E5KDAQ.dll" _
                            (ByVal zIP As String, ByVal timeout As Integer) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short      E5K_TCPPing (char zIP [], int timeout);
```

Parameters

zIP	IP address string (such as 192.168.0.123)
Timeout	ping timeout (msec)

Return Code Refer to the *Error code*.

15.50 E5K_TCPDisconnect

Description Release a TCP connection

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_TCPDisconnect Lib "E5KDAQ.dll" _
                           (ByVal sock As Long) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short       E5K_TCPDisconnect (SOCKET sock);
```

Parameters sock TCP connection handle

Return Code Refer to the *Error code*.

15.51 E5K_TCPAllDisconnect

Description Release all TCP connections

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_TCPAllDisconnect Lib "E5KDAQ.dll" _
                           () As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short       E5K_TCPAllDisconnect ();
```

Parameters none

Return Code Refer to the *Error code*.

15.52 E5K_UDPConnect

Description Establish a UDP connection by default Hos IP

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_UDPConnect Lib "E5KDAQ.dll" _
(ByVal zIP As String, ByVal port As Integer, ByVal iConnectionTimeout As Integer,
ByVal iSendTimeout As Integer, ByVal iReceiveTimeout As Integer, _
Byref insubnet as integer ) As long
```

VC++/BC++Builder: (see E5KDAQ.h)

```
SOCKET               E5K_UDPConnect (
                     char szIP [], u_short port, int iConnectionTimeout,
                     int iSendTimeout, int iReceiveTimeout,BOOL *insubnet);
```

Parameters

szIP	Target IP address
port	connection port
iConnectionTimeout	connection timeout value(msec)
iSendTimeout	send timeout value(msec)
iReceiveTimeout	receive timeout value(msec)
isubnet	return 1,if szIP is in the Host IP subnet ,otherwise not in Host subnet

Return Code return socket handle or 0 for error (call E5K_GetLastError() to read error code)

15.53 E5K_UDPConnectEx

Description Establish a UDP connection by the specified Host IP

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function     E5K_UDPConnectEx Lib "E5KDAQ.dll" _
(ByVal zIP As String, ByVal port As Integer, ByVal iConnectTimeout As Integer,
ByVal iTxTimeout As Integer, ByVal iRxTimeout As Integer, _
Byref as insubnet as integer,Byval HostIP as string) As long
```

VC++/BC++Builder: (see E5KDAQ.h)

```
SOCKET          E5K_UDPConnectEx (
                char szIP [], u_short port, int iConnectionTimeout,
                int iSendTimeout, int iReceiveTimeout,char HostIP[],
                BOOL *insubnet);
```

Parameters

szIP	Target IP address
port	connection port
iConnectTimeout	connection timeout value(msec)
iTxTimeout	send timeout value(msec)
iRxTimeout	receive timeout value(msec)
isubnet	return 1,if szIP is in the Host IP subnet ,otherwise not in Host subnet
Hostip[]	points to Host IP address buffer (such as "192.168.0.10")

Return Code return socket handle or 0 for error (call E5K_GetLastError() to read error code)

15.54 E5K_UDPSendData

Description Send data to UDP connection

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function     E5K_UDPSendData Lib "E5KDAQ.dll" _
(ByVal sock As Long, ByRef pdata As Byte, ByVal datalen As Integer) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short      E5K_UDPSendData (SOCKET sock, SBYTE *pdata, u_short datalen);
```

Parameters

sock	UDP socket handle
pdata	Points to data buffer
datalen	bytes of data

Return Code Refer to the *Error code*.

15.55 E5K_UDPRecvData

Description Receive data from UDP connection

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function     E5K_UDPRecvData Lib "E5KDAQ.dll" _
            (ByVal sock As Long, ByRef pdata As Byte, ByVal psize As Integer) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short      E5K_UDPRecvData (SOCKET sock, char *pdata, unsigned short psize);
```

Parameters

sock	UDP socket handle
pdata	Points to data buffer
psize	size of data buffer

Return Code Bytes of data received

15.56 E5K_UDPSendASCStr

Description Send ASCII string to UDP connection

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
```
Declare/public Function     E5K_UDPSendASCStr Lib "E5KDAQ.dll" _
            (ByVal sock As Long, ByVal strData As String) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)
```
Unsigned short      E5K_UDPSendASCStr (SOCKET sock, SBYTE *StrData);
```

Parameters

sock	UDP socket handle
StrData	Points to string buffer

Return Code Refer to the *Error code*.

15.57 E5K_UDPRecvASCStr

Description Receive string data from UDP connection

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
```
Declare/public Function     E5K_UDPRecvASCStr Lib "E5KDAQ.dll" _
            (ByVal sock As Long, ByVal strdata As String, ByVal bufsize As Integer) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)
```
Unsigned short      E5K_UDPRecvASCStr(SOCKET sock, char * strdata, unsigned short
                    bufsize);
```

Parameters

sock	UDP socket handle
strdata	Points to string data buffer
bufsize	size of string data buffer

Return Code Bytes of string data received

15.58 E5K_UDPDisconnect

Description Release a UDP connection

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
```
Declare/public Function     E5K_UDPDisconnect Lib "E5KDAQ.dll" _
                            (ByVal sock As Long) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)
```
Unsigned short      E5K_UDPDisconnect (SOCKET sock);
```

Parameters

sock	UDP connection handle

Return Code Refer to the *Error code*.

15.59 E5K_UDPAllDisconnect

Description Release all UDP connections

Syntax Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
Declare/public Function E5K_UDPAllDisconnect Lib "E5KDAQ.dll" () As Integer

VC++/BC++Builder: (see E5KDAQ.h)
Unsigned short E5K_UDPAllDisconnect ();

Parameters none

Return Code Refer to the *Error code*.

15.60 E5K_ReadAIChannelType

Description Read analog channel type

Syntax Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
Declare/public Function E5K_ReadAIChannelType Lib "E5KDAQ.dll" _
(ByVal id As Integer, ByVal AIChannel as Integer, ByRef AIType as Integer) As Integer

VC++/BC++Builder: (see E5KDAQ.h)
Unsigned short E5K_ReadAIChannelType (int id, unsigned int AIChannel, unsigned int *AIType);

Parameters

Id	target module ID
AIChannel	channel number
AIType	buffer pointer to store the Channel Type (See sec. 12.2)

Return Code Refer to the *Error code*.

15.61 E5K_SetAIChannelType

Description Set analog channel type

Syntax Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
Declare/public Function E5K_SetAIChannelType Lib "E5KDAQ.dll" _
(ByVal id As Integer, ByVal AIChannel as Integer, ByVal AIType as Integer) As Integer

VC++/BC++Builder: (see E5KDAQ.h)
Unsigned short E5K_SetAIChannelType (int id, unsigned int AIChannel, unsigned int AItype);

Parameters

Id	target module ID
AIChannel	channel number
AIType	buffer pointer to store the Channel Type (See sec.12.2)

Return Code Refer to the *Error code*.

15.62 E5K_SetSingleChannelColdJunctionOffset

Description Set cold junction offset of single analog channel

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function     E5K_SetSingleChannelColdJunctionOffset Lib "E5KDAQ.dll" _
 (ByVal id As Integer, ByVal chno As Integer, ByVal CjOffset as Double) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short      E5K_SetSingleChannelColdJunctionOffset (int id, unsigned int
                    chno,double CjOffset);
```

Parameters

Id	the target module id
Chno	channel number
CjOffset	channel CJC offset value (such as 0.231)

Return Code Refer to the *Error code*.

15.63 E5K_ReadSingleChannelColdJunctionOffset

Description Read cold junction offset of single analog channel

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function     E5K_ReadSingleChannelColdJunctionOffset Lib "E5KDAQ.dll" _
(ByVal id As Integer, ByVal chno As Integer, ByRef CjOffset as Double) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short      E5K_ReadSingleChannelColdJunctionOffset (
                    int id,
                    unsigned int chno,
                    double *CJoffset);
```

Parameters

Id	the target module id
Chno	channel number
CjOffset	buffer pointer to store channel CJC offset value

Return Code Refer to the *Error code*.

15.64 E5K_ReadMultiChannelColdJunctionOffset

Description Read cold junction offset of multiple analog channels

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function     E5K_ReadMultiChannelColdJunctionOffset Lib "E5KDAQ.dll" _
(ByVal id as Integer, ByVal startch As Integer, Byval counts as integer, Byref
 CjOffset as Double) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short      E5K_ReadMultiChannelColdJunctionOffset (
                    unsigned int id,
                    unsigned int startch,
                    unsigned int counts,
                    double * CjOffset);
```

Parameters

Id	the target module id
Startch	start channel number
Counts	channels to be read
CjOffset	points to an array to store CJC offset value

Return Code Refer to the *Error code*.

15.65 E5K_SetMultiChannelColdJunctionOffset

Description Set cold junction offset of multiple analog channels

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_SetMultiChannelColdJunctionOffset Lib "E5KDAQ.dll" _
( ByVal id As Integer, _
ByVal startch As Integer, _
Byval counts as integer, _
Byref CjOffset As Double _
) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short      E5K_SetMultiChannelColdJunctionOffset (
                    unsigned int id,
                    unsigned int startch,
                    unsigned int counts,
                    double * CjOffset);
```

Parameters

Id	the target module id
Startch	start channel number
Counts	channels to be set
CjOffset	points to an array where store channel CJC offset values

Return Code Refer to the *Error code*.

15.66 E5K_ReadColdJunctionTemperature

Description Read cold junction temperature

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_ReadColdJunctionTemperature Lib "E5KDAQ.dll" _
                           (ByVal id As Integer, ByRef CjTemp as Double) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short E5K_ReadColdJunctionTemperature (int id, double *CJtemp);
```

Parameters

Id	the target module id
CJtemp	CJC temperature (such as 23.67)

Return Code Refer to the *Error code*.

15.67 E5K_ReadColdJunctionStatus

Description Read CJC enable/disable status

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_ReadColdJunctionStatus Lib "E5KDAQ.dll" _
                          (ByVal id As Integer, ByRef Cjs as Byte) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short E5K_ReadColdJunctionStatus (int id, SBYTE *Cjs);
```

Parameters

Id	the target module id
Cjs	CJC enable/disable status (0: disabled, 1: enabled)

Return Code Refer to the *Error code*.

15.68 E5K_SetColdJunction

Description Enable/disable CJC

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_SetColdJunction Lib "E5KDAQ.dll" _
                          (ByVal id As Integer, ByVal Cjs as Byte) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short E5K_SetColdJunction (int id, SBYTE Cjs);
```

Parameters

Id	the target module id
Cjs	CJC enable/disable option (0: disable, 1: enable)

Return Code Refer to the *Error code*.

15.69 E5K_ReadAIChannelConfig

Description Read analog channel configuration

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_ReadAIChannelConfig Lib "E5KDAQ.dll" _
                          (ByVal id As Integer, ByVal chno As Integer, ByRef mConfig As
                          AI_CHANNEL_CONFIG) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short      E5K_ReadAIChannelConfig
                    (Int id, unsigned int chno, AI_CHANNEL_CONFIG * mConfig);
```

Parameters

Id	the target module id
chno	channel number
mConfig	points to a structure to store the channel configuration (see E5KDAQ.H about structure CHANNEL_CONFIG)

Return Code Refer to the *Error code*.

15.70 E5K_SetAIChannelConfig

Description Set analog channel configuration

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_SetAIChannelConfig Lib "E5KDAQ.dll" _
(ByVal id As Integer, ByVal chno As Integer, mConfig As AI_CHANNEL_CONFIG) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short             E5K_SetAIChannelConfig (int id, unsigned int
                           ch,AI_CHANNEL_CONFIG * mConfig);
```

Parameters

Id	the target module id
chno	channel number
mConfig	points to a structure where stores the channel configuration (see E5KDAQ.H about structure CHANNEL_CONFIG)

Return Code Refer to the *Error code*.

15.71 E5K_ReadAIBurnOutStatus

Description Read analog burnout status

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_ReadAIBurnOutStatus Lib "E5KDAQ.dll" _
                           (ByVal id As Integer, ByRef status As Long) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short E5K_ReadAIBurnOutStatus (int id, unsigned int *status);
```

Parameters

Id	the target module id
Status	points to a buffer to store the channel burnout status (Bit #n=0: channel #n is normal, bit #n=1 channel #n is burnout)

Return Code Refer to the *Error code*.

15.72 E5K_ReadAIAlarmStatus

Description Read high/low alarm status of analog channels

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_ReadAIAlarmStatus Lib "E5KDAQ.dll" _
(ByVal id As Integer, ByRef hialarm As Long, ByRef loalarm As Long) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short E5K_ReadAIAlarmStatus (int id, unsigned int *hialarm, unsigned int *loalarm);
```

Parameters

Id	the target module id
Hialarm	points to a buffer to store the channel high alarm status (bit #n=0: channel #n is normal, bit #n=1 channel #n is high alarm)
Loalarm	points to a buffer to store the channel low alarm status (bit #n=0: channel #n is normal, bit #n=1 channel #n is low alarm)

Return Code Refer to the *Error code*.

15.73 E5K_SetAIBurnOut

Description Enable/disable AI burnout detection

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_SetAIBurnOut Lib "E5KDAQ.dll" _
                           (ByVal id As Integer, ByVal option As Byte) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short      E5K_SetAIBurnOut (int id, SBYTE option);
```

Parameters

Id	the target module id
Option	=0: disable burnout detection, =1: enable burnout detection

Return Code Refer to the *Error code*.

15.74 E5K_ReadAIBurnOut

Description Read burnout detection enable/disable status

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_ReadAIBurnOut Lib "E5KDAQ.dll" _
                           (ByVal id As Integer, ByRef option As Byte) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short      E5K_ReadAIBurnOut (int id, SBYTE *option);
```

Parameters

Id	the target module id
Option	points to a buffer to store the burnout detection enable/disable status =0: burnout detection disabled, =1: burnout detection enabled

Return Code Refer to the *Error code*.

15.75 E5K_SetAIModuleFilter

Description Set A/D filter frequency

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_SetAIModuleFilter Lib "E5KDAQ.dll" _
                           (ByVal id As Integer, ByVal Hz as Integer) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short      E5K_SetAIModuleFilter (int id, unsigned int Hz);
```

Parameters

Id	the target module id
Hz	=50: 50Hz, =60: 60Hz, =100 100Hz, =120 120Hz

Return Code Refer to the *Error code*.

15.76 E5K_ReadAIModuleFilter

Description Read A/D filter frequency

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
Declare/public Function E5K_ReadAIModuleFilter Lib "E5KDAQ.dll" _
(ByVal id As Integer, ByRef Hz as Integer) As Integer

VC++/BC++Builder: (see E5KDAQ.h)
Unsigned short E5K_ReadAIModuleFilter (int id, unsigned int *Hz);

Parameters
Id the target module id
Hz points to a buffer to store filter frequency
=50: 50Hz, =60: 60Hz, =100 100Hz, =120 120Hz

Return Code Refer to the *Error code*.

15.77 E5K_SetAIChannelEnable

Description Enable or disable analog channels

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
Declare/public Function E5K_SetAIChannelEnable Lib "E5KDAQ.dll" _
(ByVal id As Integer, ByVal AIEnable as Long) As Integer

VC++/BC++Builder: (see E5KDAQ.h)
Unsigned short E5K_SetAIChannelEnable (int id, unsigned int AIEnable);

Parameters
Id the target module id
AIEnable Enable/disable settings
bit #n=0: disable channel #n, bit #n=1: enable channel #n

Return Code Refer to the *Error code*.

15.78 E5K_ReadAIChannelEnable

Description Read enable/disable status of analog channels

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
Declare/public Function E5K_ReadAIChannelEnable Lib "E5KDAQ.dll" _
(ByVal id As Integer, ByRef AIEnable as Long) As Integer

VC++/BC++Builder: (see E5KDAQ.h)
Unsigned short E5K_ReadAIChannelEnable (int id, unsigned int * AIEnable);

Parameters
Id the target module id
AIEnable points to a buffer to store channel Enable/disable settings
bit #n=0: channel #n is disabled, bit #n=1: channel #n is enabled

Return Code Refer to the *Error code*.

15.79 E5K_ReadAINormalMultiChannel

Description Read normal value of the multiple analog channels

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_ReadAINormalMultiChannel Lib "E5KDAQ.dll" _
                          (ByVal id As Integer, ByVal startch As Integer, ByVal counts as
                           Integer, ByRef AItemp As Double) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short      E5K_ReadAINormalMultiChannel (
                    int id,
                    unsigned int startch,
                    unsigned int counts,
                    double *AItemp);
```

Parameters

Id	the target module id
Startch	start channel number
Counts	channels
AItemp	points to a array to store AI normal values

AItemp[0]=normal value of channel #startch
AItemp[1]=normal value of channel #startch+1
AItemp[2]=normal value of channel #startch+2
…..etc

Return Code Refer to the *Error code*.

15.80 E5K_ReadAIMaximumMultiChannel

Description Read maximum value of the multiple analog channels

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_ReadAIMaximumMultiChannel Lib "E5KDAQ.dll" _
                          (ByVal id As Integer, ByVal startch As Integer, ByVal counts as
                          Integer, ByRef AItemp As Double) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short      E5K_ReadAIMaximumMultiChannel (
                    int id,
                    unsigned int startch,
                    unsigned int count,
                    double *AItemp);
```

Parameters

Id	the target module id
Startch	start channel number
Counts	channels
AItemp	points to a array to store AI maximum values

AItemp[0]= maximum value of channel #startch
AItemp[1]= maximum value of channel #startch+1
AItemp[2]= maximum value of channel #startch+2
…..etc

Return Code Refer to the *Error code*.

15.81 E5K_ReadAIMinumumMultiChannel

Description Read minimum value of the multiple analog channels

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function     E5K_ReadAIMinimunMultiChannel Lib "E5KDAQ.dll" _
(ByVal id As Integer, ByVal startch As Integer, ByVal counts as Integer,
ByRef AItemp As Double) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short      E5K_ReadAIMinimunMultiChannel (
                    int id,
                    unsigned int startch,
                    unsigned int count,
                    double *AItemp);
```

Parameters

Id	the target module id
Startch	start channel number
Counts	channels
AItemp	points to a array to store AI minimum values

AItemp[0]= minimum value of channel #startch
AItemp[1]= minimum value of channel #startch+1
AItemp[2]= minimum value of channel #startch+2
…..etc

Return Code Refer to the *Error code*.

15.82 E5K_ResetAIMaximum

Description Reset analog channel maximum value

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function     E5K_ResetAIMaximum Lib "E5KDAQ.dll" _
                            (ByVal id As Integer, ByVal restopt As Long) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short E5K_ResetAIMaximum (int id, unsigned int resetopt);
```

Parameters

Id	the target module ID
Restopt	rest mask option

bit #n=0: no reset maximum value of channel #n
bit #n=1: reset maximum value of channel #n

Return Code Refer to the *Error code*.

15.83 E5K_ResetAIMinimum

Description Reset analog channel minimum value

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
```
Declare/public Function    E5K_ResetAIMinimum Lib "E5KDAQ.dll" _
                           (ByVal id As Integer, ByVal resetopts As Long) As Integer
```
VC++/BC++Builder: (see E5KDAQ.h)
```
Unsigned short E5K_ResetAIMinimum (int id, unsigned int Restopt);
```

Parameters
Id the target module ID
Restopt rest mask option
bit #n=0: no reset minimum value of channel #n
bit #n=1: reset minimum value of channel #n

Return Code Refer to the *Error code*.

15.84 E5K_ResetAIHighAlarm

Description Reset analog high alarm flag

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
```
Declare/public Function    E5K_ResetAIHighAlarm Lib "E5KDAQ.dll" _
                           (ByVal id As Integer, ByVal restopt As Long) As Integer
```
VC++/BC++Builder: (see E5KDAQ.h)
```
Unsigned short E5K_ResetAIHighAlarm (int id, unsigned int restopt);
```

Parameters
Id the target module ID
Restopt rest mask option
bit #n=0: no reset high alarm flag of channel #n
bit #n=1: reset high alarm flag of channel #n

Return Code Refer to the *Error code*.

15.85 E5K_ResetAILowAlarm

Description Reset analog low alarm flag

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
```
Declare/public Function    E5K_ResetAILowAlarm Lib "E5KDAQ.dll" _
                           (ByVal id As Integer, ByVal restopt As Long) As Integer
```
VC++/BC++Builder: (see E5KDAQ.h)
```
Unsigned short E5K_ResetAILowAlarm (int id, unsigned int restopt);
```

Parameters
Id the target module ID
Restopt rest mask option
bit #n=0: no reset low alarm flag of channel #n
bit #n=1: reset low alarm flag of channel #n

Return Code Refer to the *Error code*.

15.86 E5K_ReadAIChannelAverage

Description Read analog channel in average status

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_ReadAIChannelAverage Lib "E5KDAQ.dll" _
                           (ByVal id As Integer, ByRef inavg As Long) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short E5K_ReadAIChannelAverage (int id, unsigned int * inavg);
```

Parameters

Id	the target module id
Inavg	points to a buffer to store the in average status of channels bit #n=0: channel #n is not in average bit #n=1: channel #n is in average

Return Code Refer to the *Error code*.

15.87 E5K_SetAIChannelAverage

Description Set analog channel in average

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_SetAIChannelAverage Lib "E5KDAQ.dll" _
                           (ByVal id As Integer, ByVal inavg As Long) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short E5K_SetAIChannelAverage    (int    id,unsigned int inavg );
```

Parameters

Id the target module id
inavg in average status of channels
bit #n=0: set channel #n to be not in average
bit #n=1: set channel #n to be in average

Return Code Refer to the *Error code*.

15.88 E5K_SetDIChannelConfig

Description Set DI channel configuration

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_SetDIChannelConfig Lib "E5KDAQ.dll" _
                           (ByVal id As Integer, ByVal chn As Integer, config as
                           DI_CHANNEL_CONFIG) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short      E5K_SetDIChannelConfig (int id, unsigned int
                    chn,DI_CHANNEL_CONFIG * config);
```

Parameters

Id	the target module id
Chn	DI channel number
Config	points to a structure buffer where stores the DI configuration parameters (see E5KDAQ.H)

Return Code Refer to the *Error code*.

15.89 E5K_ReadDIChannelConfig

Description Read DI channel configuration

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function     E5K_ReadDIChannelConfig Lib "E5KDAQ.dll" _
(ByVal id As Integer, ByVal chn As Integer, config e As DI_CHANNEL_CONFIG) As Integer
```

VC++/BC++Builder: (see E5KDAQ.h)

```
Unsigned short      E5K_ReadDIChannelConfig (
                    int id ,
                    unsigned int chn,
                    DI_CHANNEL_CONFIG * confige);
```

Parameters

Id	the target module id
Chn	DI channel number
Config	points to a structure buffer to store the DI configuration parameters (see E5KDAQ.H)

Return Code Refer to the *Error code*.

15.90 E5K_ReadDIStatus

Description Read digital input status

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function     E5K_ReadDIStatus Lib "E5KDAQ.dll" _
                            (ByVal id As Integer, ByRef Didata as Long) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_ReadDIStatus (int id, unsigned long *Didata);
```

Parameters

id	module ID address
Didata	points to a 32-bit buffer to store DI status

Return Code Refer to the *Error code*.

15.91 E5K_ReadDILatch

Description Read digital input latch status

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function     E5K_ReadDILatch Lib "E5KDAQ.dll" _
                            (ByVal id As Integer, ByRef Dilatch as Long) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_ReadDILatch (int id, unsigned long *Dilatch);
```

Parameters

id	module ID address
Dilatch	points to a 32-bit buffer to store DI latch status

Return Code Refer to the *Error code*.

15.92 E5K_ClearAllDILatch

Description Clear all digital input latch status

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_ClearAllDILatch Lib "E5KDAQ.dll" _
                          (ByVal id As Integer) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_ClearAllDILatch (int id);
```

Parameters

id	module ID address

Return Code Refer to the *Error code*.

15.93 E5K_ClearSingleDICounter

Description Clear counter of single digital input channel

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_ClearSingleDICounter Lib "E5KDAQ.dll" _
                          (ByVal id As Integer, ByVal chan As Integer) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_ClearSingleDICounter (int id, unsigned int chan);
```

Parameters

id	module ID address
chan	channel no.

Return Code Refer to the *Error code*.

15.94 E5K_ReadMultiDICounter

Description Clear counter of single digital input channel

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_ReadMultiDICounter Lib "E5KDAQ.dll" _
                          (ByVal id As Integer, ByVal startchn As Integer,
                          ByVal counts As Integer, counterval As Long) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_ReadMultiDICounter (
                    int id,
                    unsigned int startchn,
                    unsigned int counts,
                    unsigned long counterval []);
```

Parameters

id	module ID address
Startchn	channel no.
Counts	how many channels
Counterval []	points to buffer to store counter value

Return Code Refer to the *Error code*.

15.95 E5K_WriteDO

Description Write data to DO channels

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_WriteDO Lib "E5KDAQ.dll"
                           (ByVal id As Integer, ByVal dodata As Long) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_WriteDO (int id, unsigned long dodata);
```

Parameters

id	module ID address
Dodata	32-bit DO data, bit-n of dodata represents DO channel n bit-n=0 inactive, bit-n=1 active

Return Code Refer to the *Error code*.

15.96 E5K_ReadDOStatus

Description Read DO status

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_ReadDOStatus Lib "E5KDAQ.dll" _
                           (ByVal id As Integer, ByRef doval As Long) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_ReadDOStatus (int id, unsigned long *doval);
```

Parameters

id	module ID address
doval	points to a 32-bit data buffer to store DO status, bit-n of doval represents DO channel n bit-n=0 inactive, bit-n=1 active

Return Code Refer to the *Error code*.

15.97 E5K_SetDOSingleChannel

Description Set single DO channel

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function    E5K_SetDOSingleChannel Lib "E5KDAQ.dll" _
(ByVal id As Integer, ByVal chno As Integer, ByVal status As Byte) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short E5K_SetDOSingleChannel (int id, unsigned int chano, unsigned char status);
```

Parameters

id	module ID address
Chano	DO channel number (0~31)
Status	status=0 deactivate DO channel, status=1 activate DO channel

Return Code Refer to the *Error code*.

15.98 E5K_SetDOPulseWidth

Description Set pulse high/low width of specified DO channel

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function     E5K_SetDOPulseWidth Lib "E5KDAQ.dll" _
(ByVal id As Integer, ByVal dochn As Integer, ByVal highwidth as Integer, _
ByVal lowwidth As Integer) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short          E5K_SetDOPulseWidth (   int id,  unsigned int dochn,
                                                unsigned int highInterval,
                                                unsigned int lowInterval);
```

Parameters

id	module ID address
Dochn	DO channel number (0~31)
Highwidth	DO pulse high level width in 0.5msec unit
Lowwidth	DO pulse low level width in 0.5msec unit

Return Code Refer to the *Error code*.

15.99 E5K_ReadDOPulseWidth

Description Read pulse high/low width of specified DO channel

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function     E5K_ReadDOPulseWidth Lib "E5KDAQ.dll" _
                            ByVal id As Integer, ByVal dochn As Integer,
                            ByRef highwidth As Long,
                            ByRef Lowwidth As Long) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_ReadDOPulseWidth (
                    int id,
                    unsigned int dochn,
                    unsigned int *highwidth,
                    unsigned int *Lowwidth);
```

Parameters

id	module ID address
Dochn	DO channel number (0~31)
Highwidth	points to 16-bit buffer to store pulse high width in 0.5msec unit
Lowwidth	points to 16-bit buffer to store pulse low width in 0.5msec unit

Return Code Refer to the *Error code*.

15.100 E5K_StartDOPulse

Description Start DO pulse output

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
Declare/public Function E5K_StartDOPulse Lib "E5KDAQ.dll" _
(ByVal id As Integer, ByVal dochn As Integer, ByVal pulses As integer) As Integer

VC++: (see E5KDAQ.h)
Unsigned short E5K_StartDOPulse (int id, unsigned int Dochn, unsigned int pulses);

Parameters

id	module ID address
Dochn	DO channel number (0~31)
Pulses	how many pulses

Return Code Refer to the *Error code*.

15.101 E5K_StartMultipleDOPulse

Description Start/Stop DO pulse output of multiple DO channels

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
Declare/public Function E5K_StartMultipleDOPulse Lib "E5KDAQ.dll" _
(ByVal id As Integer, ByVal dochn As Integer, ByVal DOchbit as long,ByRef pulses As integer) As Integer

VC++: (see E5KDAQ.h)
Unsigned short E5K_StartMultipleDOPulse (int id, unsigned unsigned long Dochbit, unsigned int pulses[]);

Parameters

id	module ID address
Dochbit	DO channel start/stop pulse output bit (0~31) bit n=1 start channel n ,bit n=0 stop channel n
Pulses[]	points to pulses array

Return Code Refer to the *Error code*.

15.102 E5K_StopDOPulse

Description Stop DO pulse output of single channel

Syntax Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
Declare/public Function E5K_StopDOPulse Lib "E5KDAQ.dll" _
(ByVal id As Integer, ByVal dochn As Integer) As Integer

VC++: (see E5KDAQ.h)
Unsigned short E5K_StopDOPulse (int id, unsigned int dochn);

Parameters

id	module ID address
Dochn	DO channel number (0~31)

Return Code Refer to the *Error code*.

15.103 E5K_ReadDOPulseCount

Description Read pulse count value.
The pulse count value will start decreasing after calling E5K_StartDOPulse () or E5K_StartMultipleDOPulse ()

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
```
Declare/public Function    E5K_ReadDOPulseCount Lib "E5KDAQ.dll" _
(ByVal id As Integer, ByVal dochn As Integer, ByRef counts As Integer) As Integer
```
VC++: (see E5KDAQ.h)
```
Unsigned short             E5K_ReadDOPulseCount
                           (Int id, unsigned int dochn, unsigned int *counts);
```

Parameters

id	module ID address
dochn	DO channel number (0~31)
counts	point to 16-bit buffer to store pulse counter value

Return Code Refer to the *Error code*.

15.104 E5K_SetDOPowerOnValue

Description Set DO power-on value

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
```
Declare/public Function    E5K_SetDOPowerOnValue Lib "E5KDAQ.dll" _
                           (ByVal id As Integer, ByVal poweronvalue As Long) As Integer
```
VC++: (see E5KDAQ.h)
```
Unsigned short             E5K_SetDOPowerOnValue (int id, unsigned long poweronvalue);
```

Parameters

id	module ID address
Poweronvalue	32-bit DO power-on value

Return Code Refer to the *Error code*.

15.105 E5K_ReadDOPowerOnValue

Description Read DO power-on value

Syntax
Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)
```
Declare/public Function    E5K_ReadDOPowerOnValue Lib "E5KDAQ.dll" _
                           (ByVal id As Integer, ByRef Dopoweron as Long) As Integer
```
VC++: (see E5KDAQ.h)
```
Unsigned short             E5K_ReadDOPowerOnValue
                           (Int id, unsigned long *PowerOnValue);
```

Parameters

id	module ID address
Poweronvalue	points to a 32-bit buffer to store DO power-on value

Return Code Refer to the *Error code*.

15.106 E5K_ReadDIOActiveLevel

Description Read DI/DO active level options

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_ReadDIOActiveLevel Lib "E5KDAQ.dll" _
                          (ByVal id As Integer, ByRef DIActiveoption as Byte, _
                          ByRef DOActiveoption as Byte) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short  E5K_ReadDIOActiveLevel (
                int id,
                unsigned char *DIActiveoption,
                unsigned char *DOActiveoption);
```

Parameters

id	module ID address
DIActiveoption	points to 8-it buffer to store DI active status option 0 or 1 (see Table 2)
DOActiveoption	points to 8-it buffer to store DO active status option 0 or 1 (Table 1)

Return Code Refer to the *Error code*.

15.107 E5K_SetDIOActiveLevel

Description Set DI/DO active level options

Syntax

Visual Basic/VB.Net: (see E5KDAQ.bas/E5KDAQ.vb)

```
Declare/public Function   E5K_SetDIOActiveLevel Lib "E5KDAQ.dll" _
                          (ByVal id As Integer, ByVal DIActiveoption as Byte, _
                          ByVal DOActiveoption as Byte) As Integer
```

VC++: (see E5KDAQ.h)

```
Unsigned short      E5K_SetDIOActiveLevel (
                    int id,
                    unsigned char DIActiveoption,
                    unsigned char DOActiveoption);
```

Parameters

id	module ID address
id	module ID address
DIActiveoption	DI active state option 0 or 1 (see Table 2)
DOActiveoption	DO active states option 0 or 1 (see Table 2)

Return Code Refer to the *Error code*.

Module Name	DI Active Option	Description
EX-5060	0	Active state ,if input open or high voltage Inactive state ,if input short or low voltage
	1	Active state ,if input short or low voltage Inactive state ,if input open or high voltage

Table 2

Module Name	DO Active Option	Description
EX-5060	0	Active state ,if relay close Inactive state ,if relay open
	1	Active state ,if relay open Inactive state ,if relay close

Table 3

Chapter 16 E5KDAQ.DLL Error Code

Error Code	Description
00	No Error
01	Device not supported
02	Device is not existed
03	Device driver not activated
04	Device driver open fail
05	Device time out
06	Device response Error
07	Invalid driver version
08	Invalid ID Number
09	Device ID overlapped
10	Invalid interface type
11	Invalid Pass Word or password not be verified
12	Invalid ASCII Command
13	Interrupt Already enabled
14	No Interrupt Data
15	Arguments Out Of Range
16	Invalid Port Number
17	Invalid DO Data
18	Invalid Digital Channel Number
19	Invalid Timer Value
20	Invalid Timer Mode
21	Invalid Counter Number
22	Invalid Counter Value
23	Invalid Counter Mode
24	Invalid A/D Filter Type
25	Invalid A/D Mode
26	Invalid A/D channel number
27	Invalid A/D Gain
28	Invalid A/D Range
29	Invalid A/D count Value
30	Invalid A/D Scan Rate
31	A/D FIFO Half Not Ready
32	Invalid D/A channel number
33	Invalid D/A Value
34	Invalid Debounce Mode
35	Invalid Debounce Time
36	Invalid Modbus Function
37	Invalid Modbus Start Address
38	Modbus Address Out Of Range
39	Modbus Range over 32 Channel
40	Winsck2 Not Opened
41	Windows winsock2 start up error
92	Invalid IP address
43	Can Not Create TCP Socket
44	Can Not Create UDP Socket
45	Can Not Set TCP/IP Timeout
46	Can Not Send Package To Destination
47	No Package Received Until Timeout
48	Unable To Read Stream Data
49	No Connection To Remote IP Address
50	Alarm Event Buffer Empty
51	Stream Event Buffer Empty
52	Unable To Allocate Memory
53	Can Not Ping Remote IP Address

54	ASCII Check Sum error
55	Mosbus CRC error
56	IP not in then subnet
57	COMM port already open
58	No enough buffer size to receive data
59	Invalid parameters
60	USB data over 1024 bytes
61	Invalid Host IP
62	Invalid Stream IP
63	Invalid alarm IP
59	Error Code Out of Range

Chapter 17 Event/Stream Interrupt structure

17.1 Event Interrupt Structure

```
Typedef struct EVENT_INTERRUPT_INFO
{
    unsigned int    szID;          //the ID address which cause the alarm interrupt
    unsigned int    wIntType;      //0= DI interrupt,1= AD_INT_TYPE
    unsigned int    wChno;         //Event channel number
    unsigned int    wStatus;       //=0 for low to high interrupt for DI or high alarm for AI channel
                                   //=1 for high to low interrupt for DI or low alarm for AI channel
    double fAddata;                //AD data if AD alarm occured
} DEVICE_INTERRUPT_INFO;
```

When event occurred, E5KDAQ.DLL will transfer argument with structure **EVENT_INTERRUPT_INFO** to callback function

17.2 Stream Interrupt Structure

```
Typedef struct STREAM_INTERRUPT_INFO
{
Unsigned int        wszID;          //the ID address which cause the alarm change
Unsigned long   dwDi;               //digital input status
Unsigned long   dwDiLatch;          //digital input latch status
Unsigned long   dwDiCount[32];      //digital input counter value
Unsigned long   dwDo;               //digital output status
double              fAiNorValue[17];    //analog input normal value
double              fAiMaxValue[16];    //analog input maximum value
double          fAiMinValue[16];    //analog input minimum value
unsigned int    wAiHighAlarmstatus; //analog input high alarm status
unsigned int    wAiLowAlarmstatus;  //analog input low alarm status
unsigned int    wAiBurnOut ;        //analog input burn-out status(9219,EX9215 only)
double              fCJCTemperature;    //cold junction temperature in 0.1C unit (9219 only)
double              fAoValue[16];       //analog output value
} STREAM_INTERRUPT_INFO;
```

When received active-stream data, E5KDAQ.DLL will transfer argument with structure **STREAM_INTERRUPT_INFO** to callback function

Chapter 18 E5KDAQ ActiveX Control

18.1 Properties Of E5KDSAQ ActiveX Control

Name	Type	Description	Model(s)
AIChannelIndex	Short	Specifies the analog input channel to perform other AI properties read/write operation.	9215,,9217,9219
AINormalValue	Double	Normal voltage of specifies the analog channel	9215,9217,9219
AIMaximumValue	Double	Maximal voltage of specifies the analog channel	9215,9217,9219
AIMinimumValue	Double	Minimal voltage of specifies the analog channel	9215,9217,59019
AILowAlarmStatus	Long	Return the low alarm status of specifies the analog channel (1=Alarm occurred/ 0=No alarm)	9215,9217,9219
AIHighAlarmStatus	Long	Return the high alarm status of specifies the analog channel (1=Alarm occurred/ 0=No alarm)	9215,9217,9219
AIBurnOutStatus	Long	Return the Burnout status of specifies the analog channel (1=open/ 0=normal)	9215,9219
AIChannelEnable	Long	Enable/disable AI channels	9215,9217,9219
AIChannels	Long	Return the total AI channels of model	
AOChannelIndex	Short	Specifies the analog output channel to perform other properties read/write operation.	Reserved
AOChannels	Long	Return the total AO channels of model	
AOValue	Double	Set the analog output voltage	All models
AIColdJunction	Double	Return the cold junction temperature	reserved
AlarmEventADValue	Double	Return the alarm AD value	9215,92179219
AlarmEventChannel	Short	Return the alarm channel number	9215,9217,9219
AlarmEventID	Short	Return the ID address of alarm model	9215,9217,9219
AlarmEventIP	String	Return the IP address of alarm model	All models
AlarmEventStatus	Short	Return 1 if AD low alarm or DI high to low return 0 if AD high alarm or DI low to high	All models
AlarmEventType	Short	Return 1 if AD type alarm event occurred return 0 if DI type alarm event occurred	All models
CheckSumCRC	Boolean	Enable/disable CheckSum/CRC	All models
DIChannelIndex	Short	Specifies the digital input channel to perform other DI properties read/write operation.	9217,9219
DIChannels		Return the total DI channels of model	9217,9219
DIounterValue	Long	Return the counting value for the specified DI channel which functions in "Count/Frequency mode"	9217,9219
DILatchStatus	Long	Return the latch status for the specified DI channel which functions in "Lo-Hi/Hi-Lo latch mode" (1=Latched/ 0=No latched)	9217,9219
DIStartCount	Boolean	Start/stop counting for the specified DI channel which functions in "Count/Frequency mode" (True=Start/ 0=Stop)	9217,9219
DIStatus	Long	Return the status for the specified DI channel which functions in "DI mode" (1=Active/ 0=Inactive)	9217,9219
DOChannelIndex	Short	Specifies the digital output channel to perform other DO properties read/write operation.	9217,9219
DOChannels	Short	Return the total DO channels of model	9217,9219

Name	Type	Description	Model(s)
DOOulseCounts	Long	Set the output count value for the specified DO channel which functions in "Pulse output mode"	9217,9219
DOStatus	Long	Return/set the status for the specified DO channel which functions in "D/O mode" (1=Active/ 0=Inactive)	9217,9219
COMBaudRate	Long	Return/set COM port baud rate	All models
CommunicationType	Short	Return/set communication interface	All models
StreamEventID	Short	Return ID address of module which generate stream data	All models
StreamIP	String	Return IP address of module which generate stream data	All models
Version	String	Return the version of ActiveX control (E5KDAQ.OCX)	All models
LastError	Short	Return the Error code of operation	All models
LastErrorDescription	String	Return the error description	All models
MoudleID	Short	Return the module ID number	All models
ModuleIP	String	Set the remote module IP address	All models
ModuleName	String	Return the module name	All models
ConnectionTimeOut	Long	Return or set the TCP/IP Timeout (ms)	All models
ReceiveTimeOut	Long	Return or set the TCP/IP or COM receive Timeout (ms)	All models
SendTimeOut	Long	Return or set the TCP/IP Send Timeout (ms)	All models
UpdatePeriod	Long	Return/set data update time period(ms)	All models

18.2 Methods of E5KDAQ ActiveX Control

Name	Arguments	Return	Description
Open	None	None	Open E5kDAQ.OCX to start operation (Must be called before accessing properties at run time)
Close	None	None	Close E5KDAQ.OCX (Must be called before terminating the APP)
ReadAlarmEventData	None	Boolean	Return the status of alarm data TRUE=alarm data ready in queue, FLASE=no alarm data
ReadStreamData	None	Boolean	Return the status of stream data TRUE=stream data ready in queue, FLASE=no stream data
SendASCII	string		Send ASCII command
RecvASCII	None		Receive ASCII command
SendHEX	short Buffer[] short length	None	Send Hex data in buffer[]
RecvHEX	short Buffer[] short buffer size	Integer	Receive hex data and store into buffer[] return the data length
StartAlarmEvent	None	Long	Start alarm interrupt return 0 if error occurred, or handle of interrupt
StartStreamEvent	None	Long	Start stream interrupt return 0 if error occurred, or handle of interrupt

18.3 Events of E5KDAQ ActiveX control

Name	Arguments	Return	Description
OnError	short ErrCode(out) string Errmsg(out)	None	be called when error occurred

Chapter 19 EX 9200 Utility Overview

The EX9200 Utility software offers a graphical interface that helps you configure the EX-9200 modules. It is also very convenient to test and monitor your remote DAQ system. The following guidelines will give you some brief instructions on how to use this Utility.

- Main Menu
- Network Setting
- Adding Remote Station
- Security setting
- I/O Module Configuration
- Alarm Setting
- I/O Module Calibration
- Security Setting
- Terminal emulation
- Data/Event Stream

19.1 Main Menu

Double Click the icon of EX-9200 Utility shortcut, the Operation screen will pop up as follow.

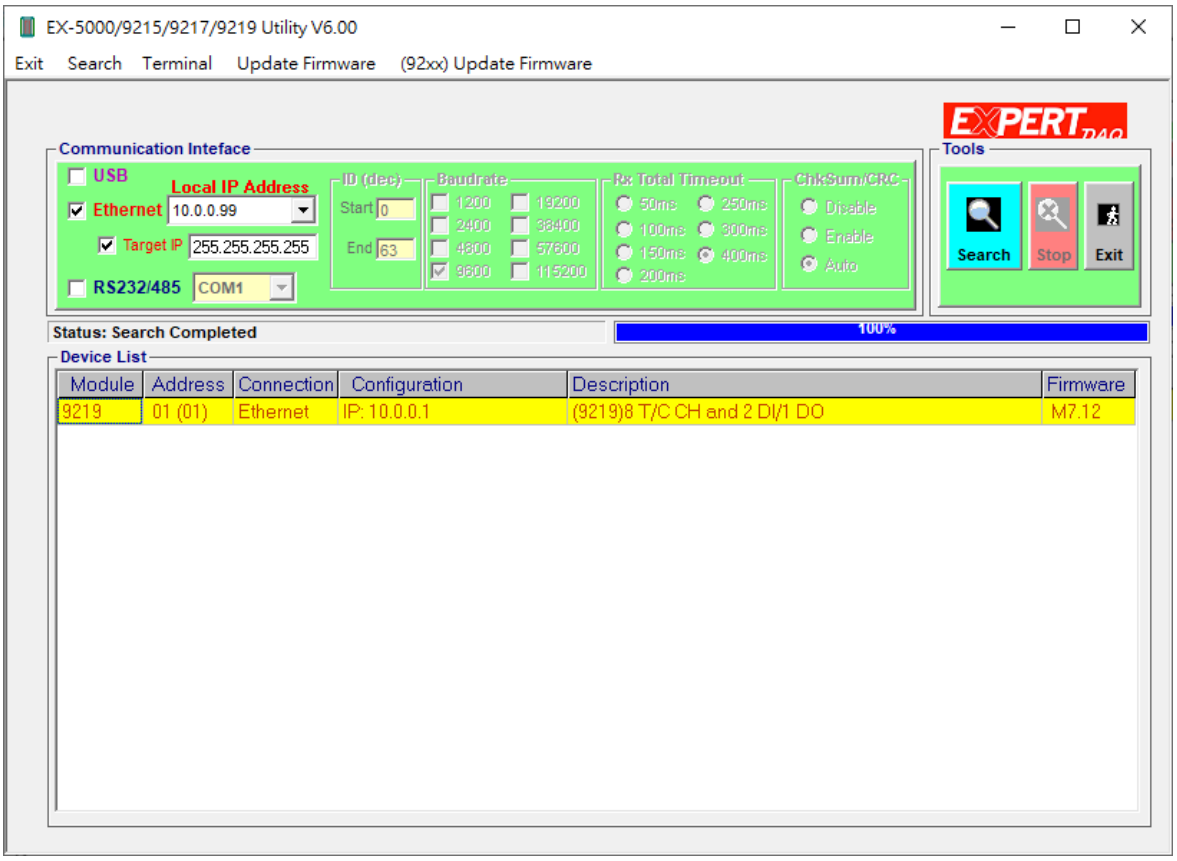


Figure - 1

The top of the operation screen consists of a function menu and a tool bar for user's commonly operating functions.

19.2 Communication Interface Settings

There are three interfaces that EX9200 modules can be connected to

- **Ethernet** option : Enable/disable Ethernet connection
- **RS232/485** option : Enable/disable Serial port connection

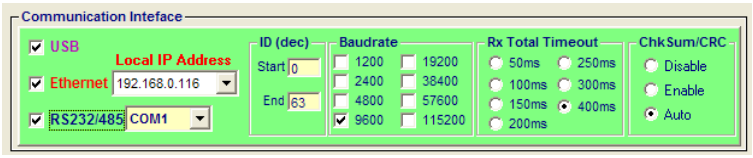


Figure - 2

- **Local IP Address** :EX9200 supports multiple Ethernet cards (multiple host IP addresses). The EX9200 module(s) can connect to one of the Ethernet card's network.

Note:

1. Each Ethernet card should have different IP subnet
2. If the Ethernet cards are in the same subnet, the combo box-structure display area will only appeal with the highest priority IP addresses
3. For example, Assume there are three Ethernet cards installed in Host with the IP addresses **192.168.0.123**(highest priority IP in 192.168.0.xxx subnet), **192.168,1,191** and **192.168.0.34**. The Combobox-structure display area will only appeal with **192.168.0.123** and **192.168.1.191**

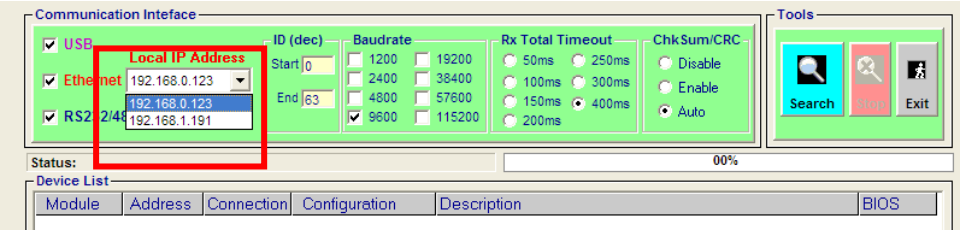


Figure - 3

- **ID Range** :Set start and end ID number range for searching (**RS232/85 only**)
- **Baud rate** :Select COM baud rate(**RS232/85 only**)
- **RX Total Timeout** :Select Receive timeout(**RS232/85 and Ethernet only**)
- **ChkSum/CRC** :Enable/disable/auto Check sum or CRC check (**RS232/485 only**)

19.3 Tool Bar

There are three push buttons(**Search, Stop, Exit**) in the tool bar.

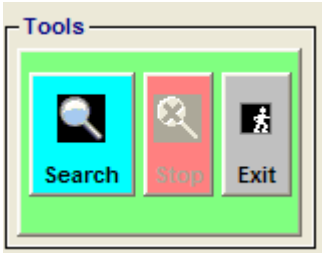


Figure - 4

- **Search** : Click this button to start searching all EX-9200 I/O module(s) on the specified interface(s) **(Ethernet, COM**) automatically. Then the Grid-structure display area will appeal with the searched units and the relative configuration.

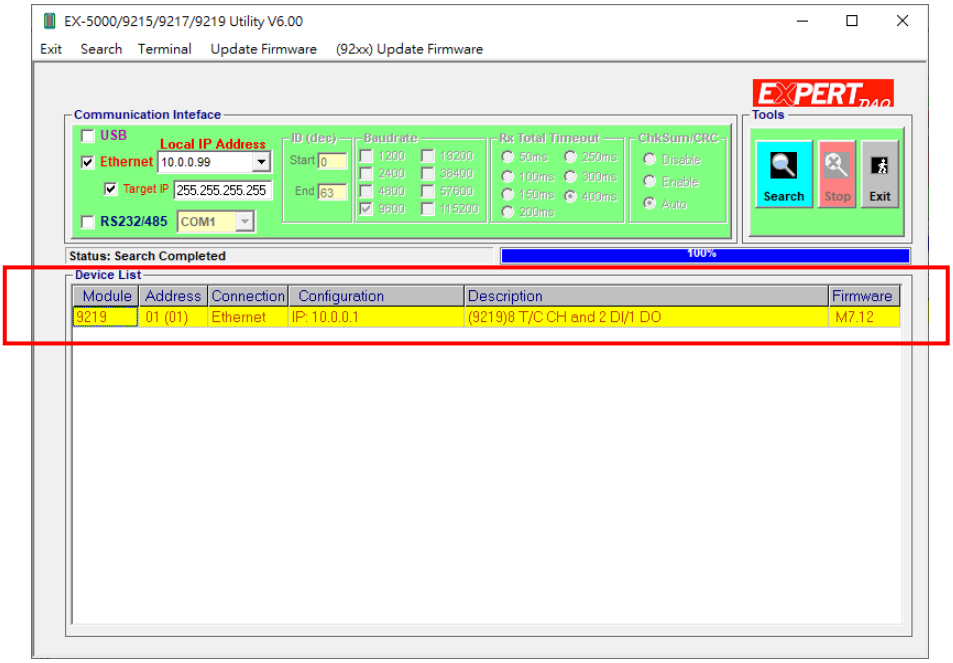
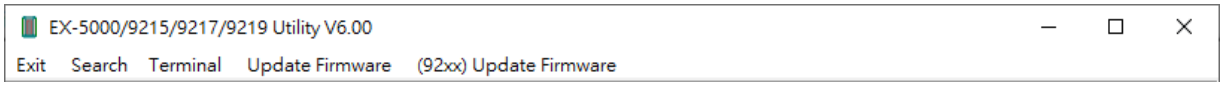


Figure - 5

- **Stop** : Click this button to stop searching
- **Exit** : Click this button to Exit this Utility program.

19.4 Menu Bar



- **Exit** :

  Exit this Utility program.

- **Search** :

  Click this to start searching all EX-9200 I/O module(s) on the specified interface(s) **(Ethernet, COM)** automatically. Then the Grid-structure display area will appeal with the searched units and the relative configuration. (See Figure - 5)

- **Terminal** :

  Call up the operation screen of Terminal emulation to do the request / response command execution.

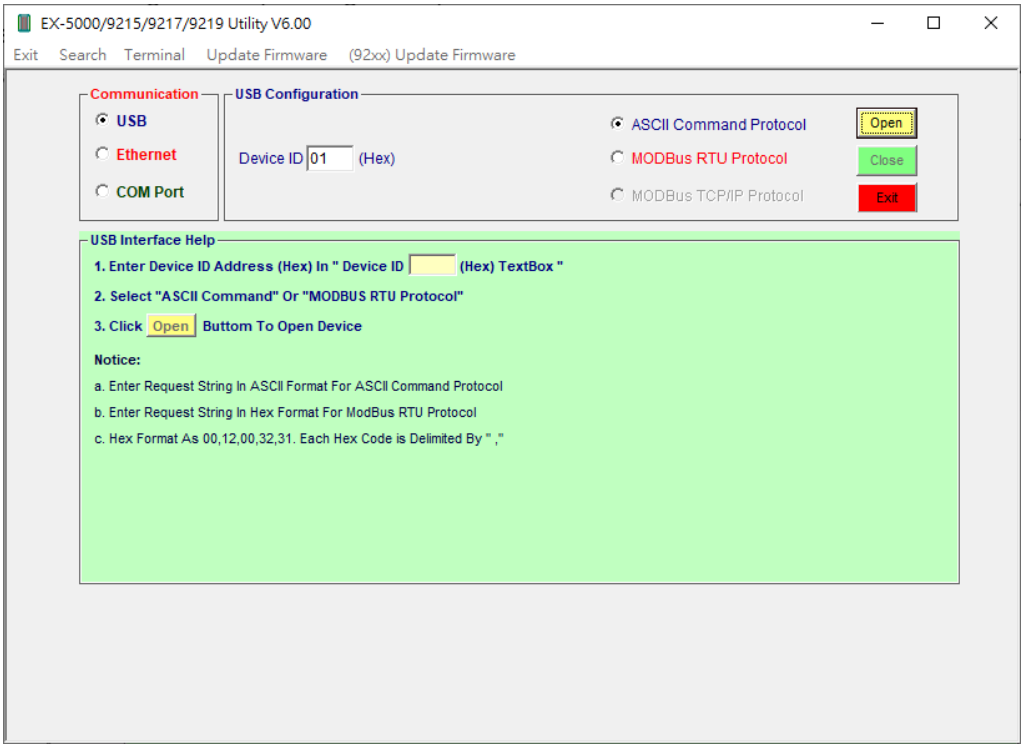


Figure - 6

- **Communication**: Select communication interface to do the request / response command execution
- **Ethernet Configuration**: Connect to target by module ID number or by IP address
- **ASCII command protocol**: do request / response command with ASCII protocol
- **Modbus RTU protocol**: do request / response command with Modbus RTU protocol
- **Modbus TCP/IP protocol:** do request / response command with Modbus /TCP protocol
- **Open button**: Connects to target
- **Close button**: Disconnects to target
- **Exit button**: Exit terminal tool

- **Firmware Update**: Update Firmware through **Ethernet** connection.

  The EX-9200 utility provides on-board firmware update tool that can help you to update firmware through Ethernet interface quickly.

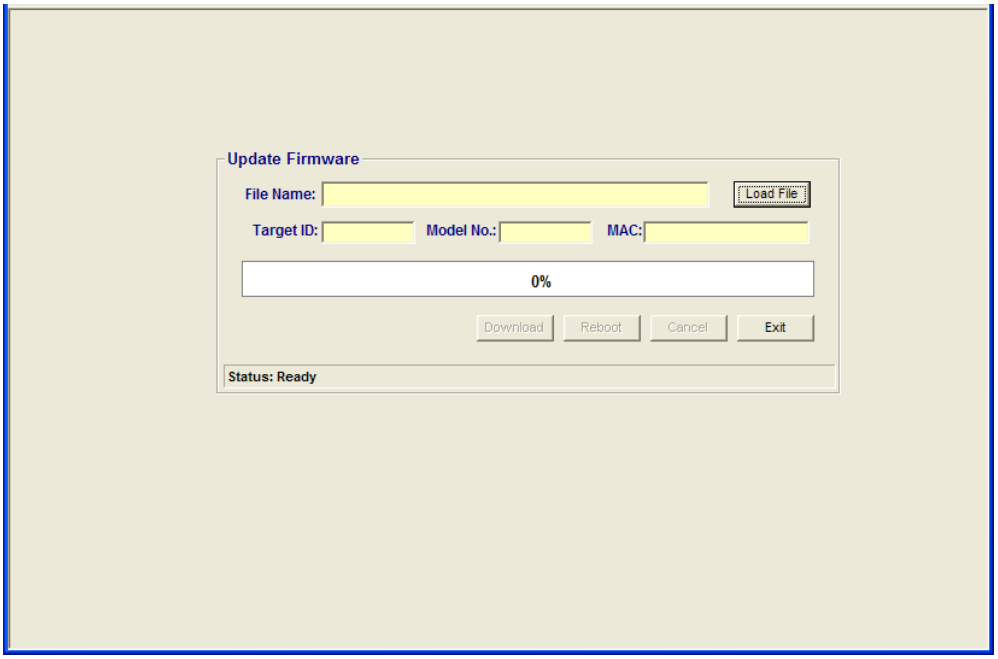


Figure - 7

  - **Load File button**:Load firmware file
  - **DownLoad button**: Search target and start download firmware to target
  - **Cancel button**: Stop download firmware process
  - **Exit button**: Exit this function

- **Help**: EX9200 User's manual

19.5 EX9200 module configuration

Since Utility software detects the EX-9200 on the specified interface, user can begin to setup each unit. Double click any one I/O module listed on the grid-structure display area and entry the correct password (for Ethernet Interface only). The module basic configuration table is listed as shown in for setting

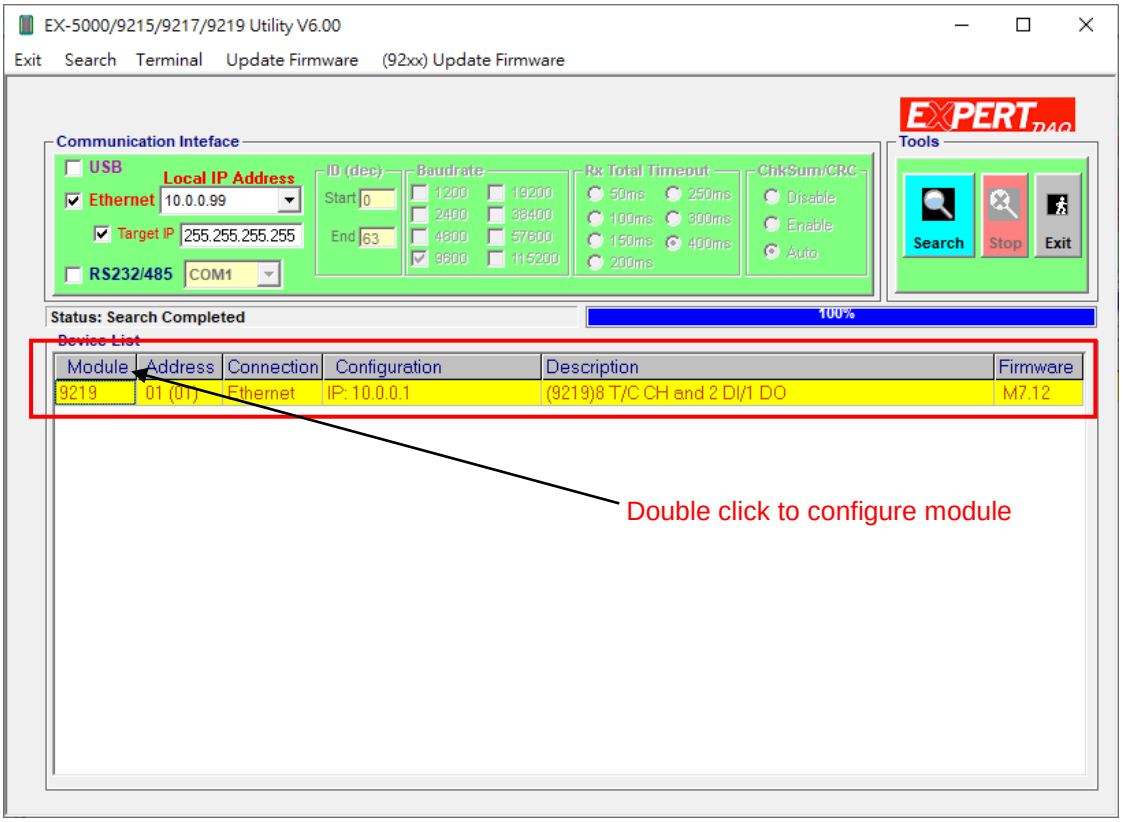


Figure - 8

19.6 EX-9219 Configuration

19.6.1 Module settings tab

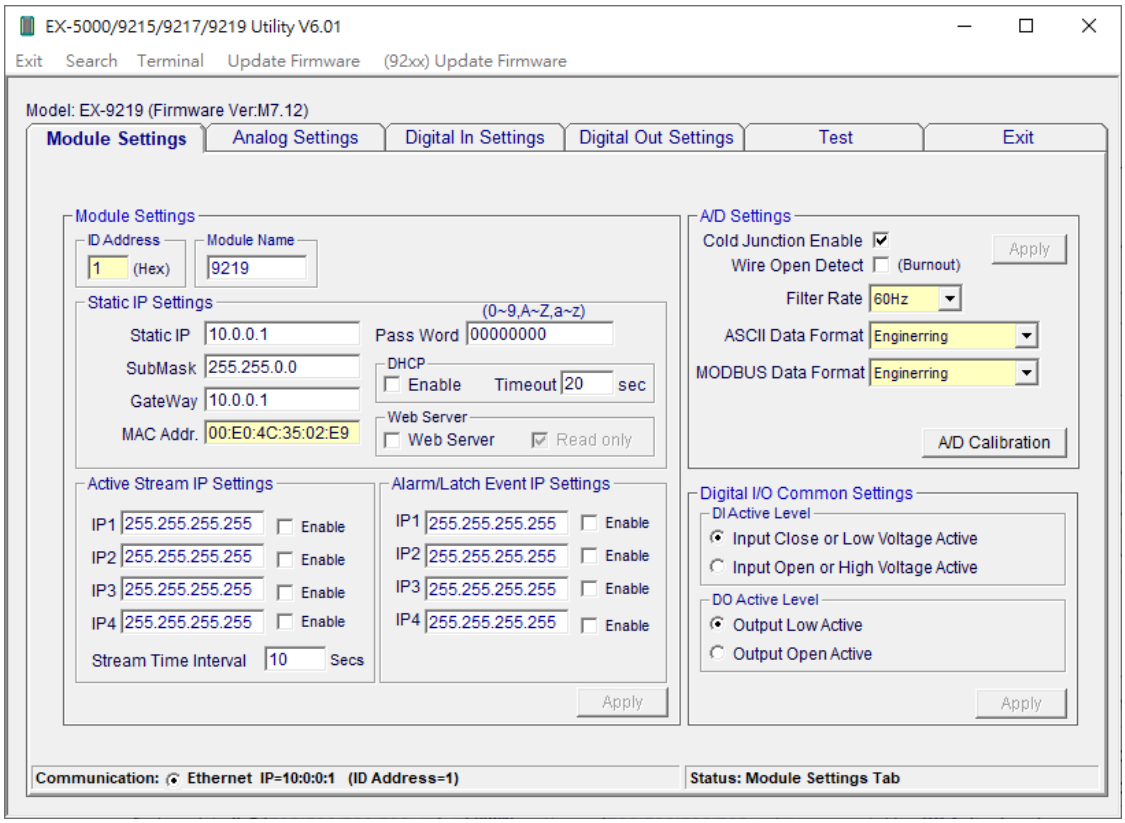


Figure - 9

- **MAC Address**:

  The Ethernet address and needs no further configuration.

- **IP Address**, **Subnet Mask**, and **Default Gateway**: (default 10.0.0.1, 255.0.0.0 and 0.0.0.0)

  The IP address identifies your EX-9200 devices on the global network. Each EX-9200 has same default IP address **10.0.0.1**. Therefore, *please do not initial many EX-9200 at the same time to avoid the Ethernet collision*. If you want to configure the EX-9200 in the host PC's dominating network, only the IP address and Subnet Mask will need to set (The host PC and EX Ethernet I/O must belong to same subnet Mask).
  If you want to configure the EX-9200 via Internet or other network domination, you have to ask your network administrator to obtain a specific IP and Gateway addresses, and then configure each EX-9200 with the individual setting.

- **DHCP:** (default Enabled)

  Allow you to get IP address from the DHCP servo without setting IP address by manual.

- **Web Server:** (default Enabled)

  Allow you monitor and control I/O status on EX-9000 modules remotely through web browser.

- **Module ID**: (default 01)

  Unique ID number of module can be set by DIP switch on the back side of module (See Help)

- **Password:** (default 00000000)

  Allow you to change the password of the module (needed for Ethernet connection only)

- **Stream/Event IP**:

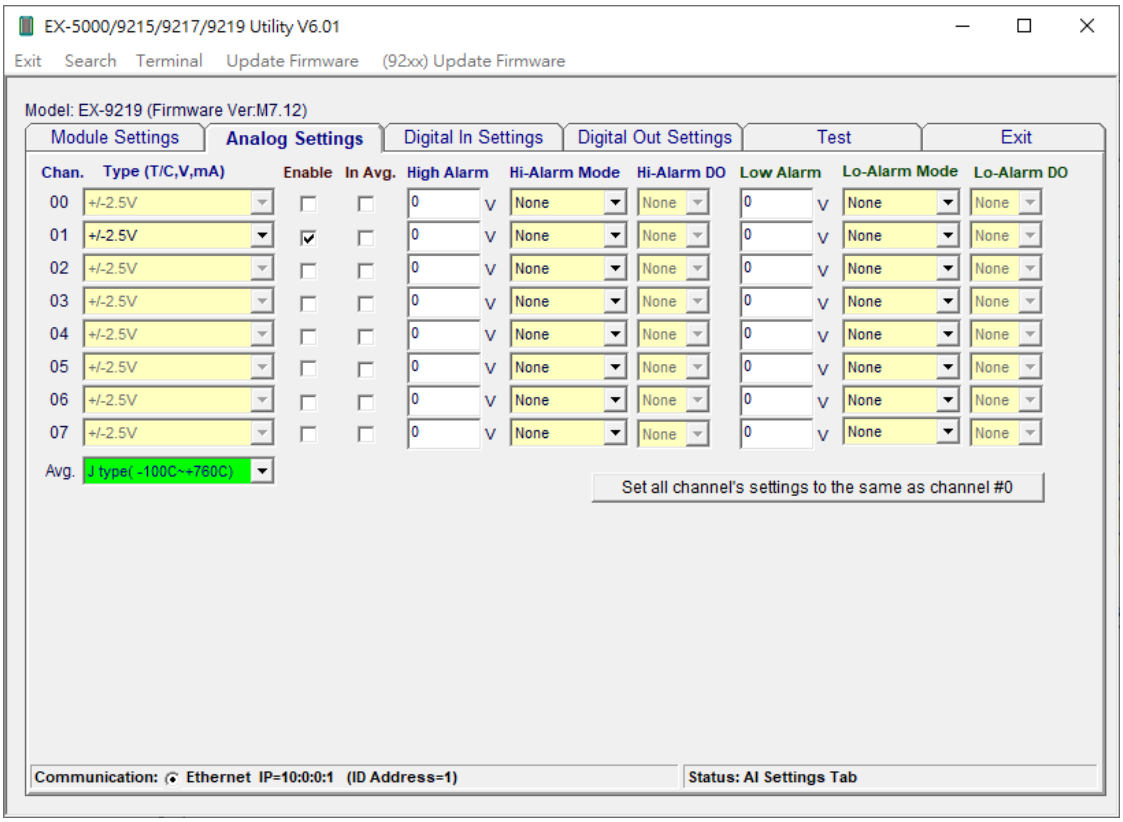
  Set Stream /Event data Destination IP

- **Stream/Event Enable Setting**: (default all disabled)

Enable/disable Stream and Event functions

- **Stream time interval**: (default 10 sec)
  Set time interval for sending stream data
- **DI active state settings**:
  Set DI active state (High input active or low input active)
- **DO active state settings**:
  Set DO output active state (High/open output active or low output active)
- **Cold junction Enable:**
  Enable/disable Cold junction compensation
- **Wire open detect:**
  Enable/disable input wire open detection
- **Filter rate:**
  Select A/D converter filter frequency
- **ASCII Data Format**
  Select ASCII data format (Engineering format or 2's complement format)
- **Modbus Data Format**
  Select Modbus data format (Engineering format or 2's complement format)
- **A/D calibration button**
  Calibrate EX9219 analog channels(Span and zero calibrations)

19.6.2 Analog settings tab



- **Type**:
  Select analog input channel type
- **Enable button**:
  Enable/disable single channel

- **In Avg. button**:

  Enable/disable channel to be in average

- **High Alarm Textbox**:

  Set AI channel high alarm value

- **Low Alarm Textbox**:

  Set AI channel low alarm value

- **High Alarm mode Combobox**:

  Select AI channel high alarm event mode
  None =Disable alarm
  Momentary =Generate Alarm event only when Channel value is greater than High Alarm value
  Latch =Generate Alarm event when Channel value is greater than High Alarm value and latch alarm event until cleared by user)

- **Low Alarm mode Combobox**:

  Select AI channel low alarm event mode
  None =Disable alarm
  Momentary =Generate Alarm event only when Channel value is less than low Alarm value
  Latch =Generate Alarm event when Channel value is less than low Alarm value and latch alarm event until cleared by user)

- **Set All channels to be the same settings as channel #0 button**:

  Set all AI channels to have the same settings as channel #0

19.6.3 Digital input settings tab

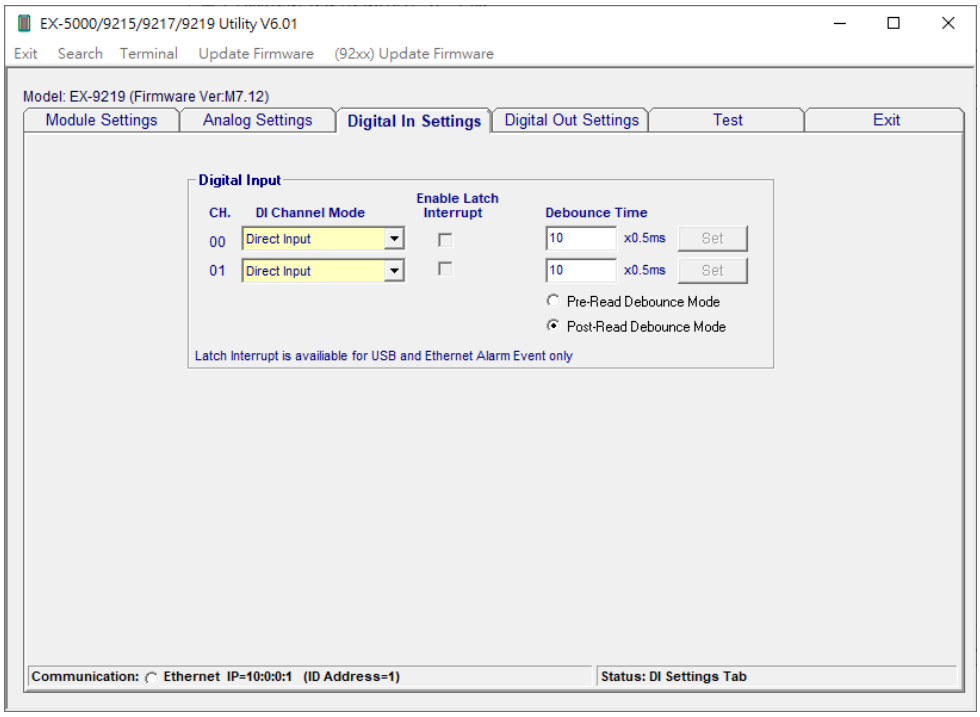


Figure - 10

- **DI channel Input Mode:**

  Direct input ,counter, low to high latch, high to low latch, frequency mode

- **Latch Interrupt:**

  Enable/Disable DI latch interrupt (USB connection only)

- **Debounce time:**

  Set DI input debounce time (0~65535). =0 no debounce

- **Select Pre-read debounce/Post read debounce mode (see** 錯誤! 找不到參照來源。**)**

Pre-read mode

The device read DI state immediately when DI state-changed and then delay 5000 msec to filter all other states changed in this time interval(**A/B/C/D** states are all ignored by device)

Post-read mode

The bounce detection is started when DI state-changed and then read final DI state after 5000 msec delay reached

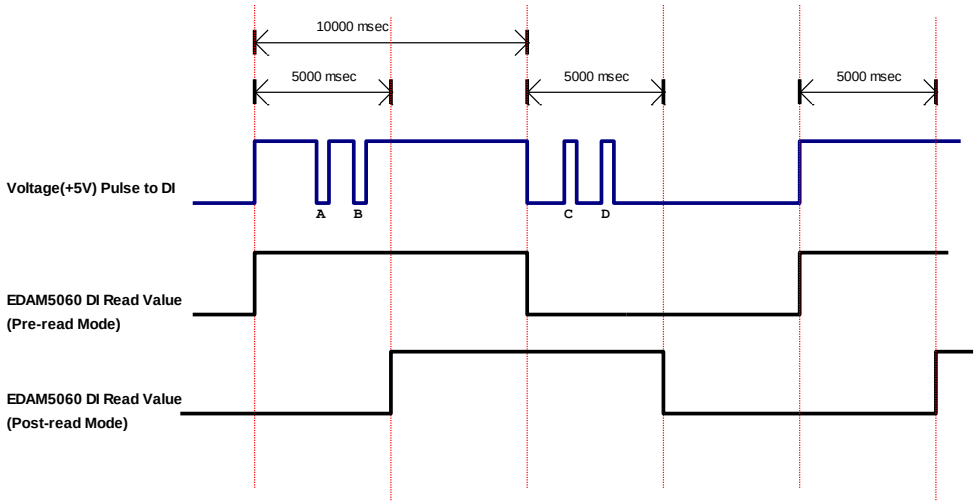


Figure - 11

- **Set all channels settings as channel 0**

Configure all channel settings as channel 0 settings

19.6.4 Digital output settings tab

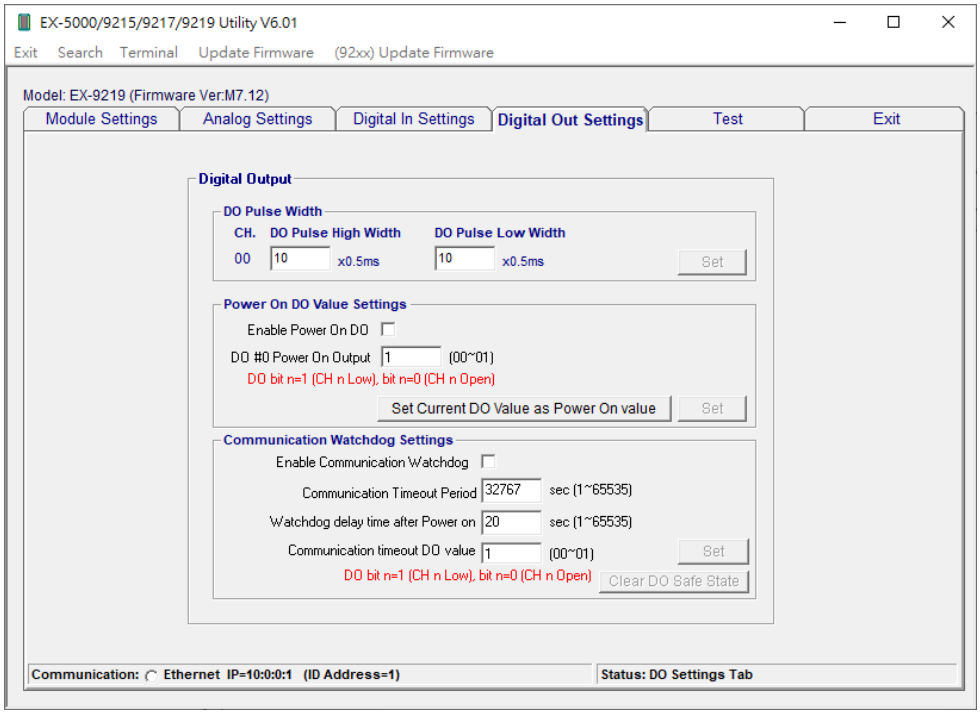


Figure - 12

- **DO pulse width**:

Set high level width and low level width for DO pulse output operation

- **Communication watchdog**:
  - **Enable/disable communication watchdog**

    If there are no data received from host When communication timeout is reached, The DO will be set to specified safe value.
  - **Communication timeout period**

    Set Communication timeout value
  - **Delay time after power–on**

    The delay time to start communication watchdog function after module reboot
  - **Communication timeout DO safe value**

    DO value when communication timeout is reached
- **Enable power-on DO**:

  DO default value after module reboot
- **Power-on DO value**:

  Set Power-on DO value

19.6.5 Test tab

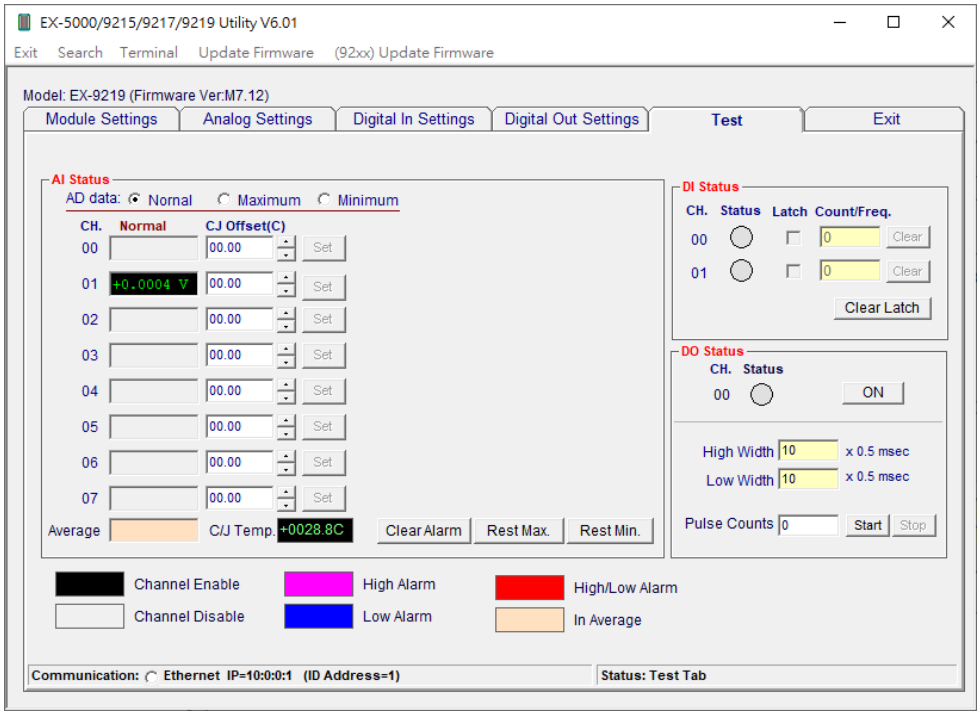


Figure - 13

- **Clear button**

  Clear single DI channel counter to zero
- **ON/OFF button**

  Toggle single DO channel output value
- **Counts TextBox**

  Enter the DO pulse counts
- **Start button**:

  Start to Generate DO pulse output
- **Stop button**:

  Stop DO pulse output

- **Clear Alarm button:**
  Set all analog channel alarm events
- **Resset Max. button:**
  Reset all ananlog Maximun value
- **Reset Min. button:**
  Reset all ananlog Minimun value

19.7 EX-9217 Configuration

19.7.1 Module settings tab

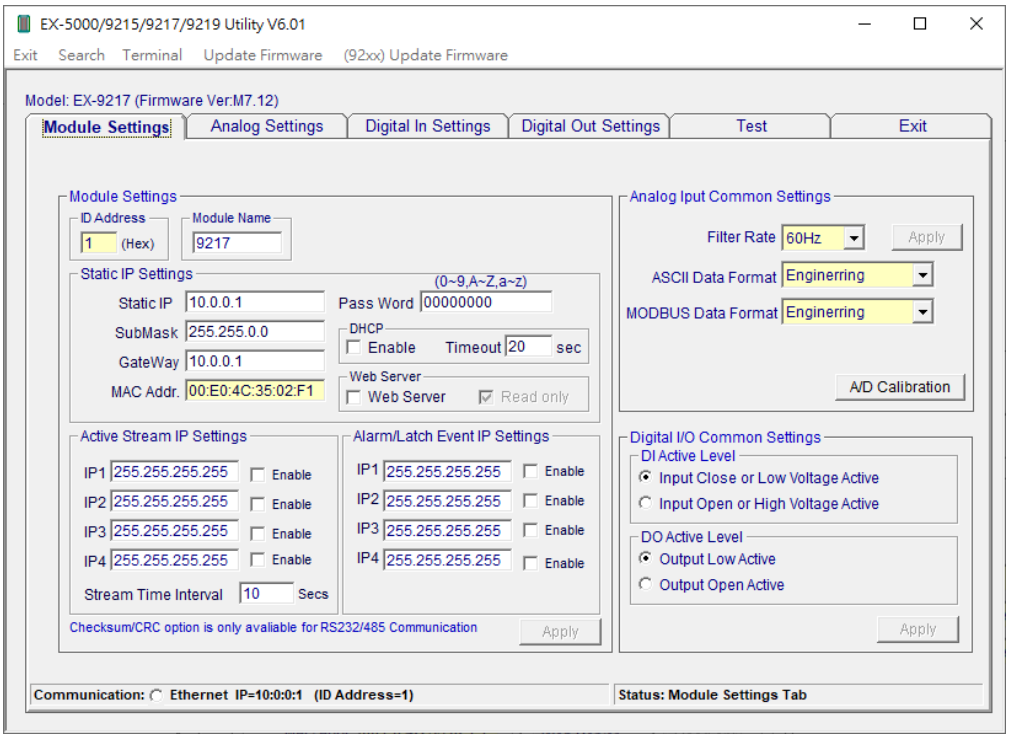


Figure - 14

- **MAC Address**:

  The Ethernet address and needs no further configuration.

- **IP Address**, **Subnet Mask**, and **Default Gateway**: (default 10.0.0.1, 255.0.0.0 and 0.0.0.0)

  The IP address identifies your EX-9200 devices on the global network. Each EX-9200 has same default IP address **10.0.0.1**. Therefore, *please do not initial many EX-9200 at the same time to avoid the Ethernet collision*. If you want to configure the EX-9200 in the host PC's dominating network, only the IP address and Subnet Mask will need to set (The host PC and EX Ethernet I/O must belong to same subnet Mask).
  If you want to configure the EX-9200 via Internet or other network domination, you have to ask your network administrator to obtain a specific IP and Gateway addresses, and then configure each EX-9200 with the individual setting.

- **DHCP:** (default Enabled)

  Allow you to get IP address from the DHCP servo without setting IP address by manual.

- **Web Server:** (default Enabled)

  Allow you monitor and control I/O status on EX-9000 modules remotely through web browser.

- **Module ID**: (default 01)

  Unique ID number of module can be set by DIP switch on the back side of module (See Help)

- **Password:** (default 00000000)

  Allow you to change the password of the module (needed for Ethernet connection only)

- **Stream/Event IP**:

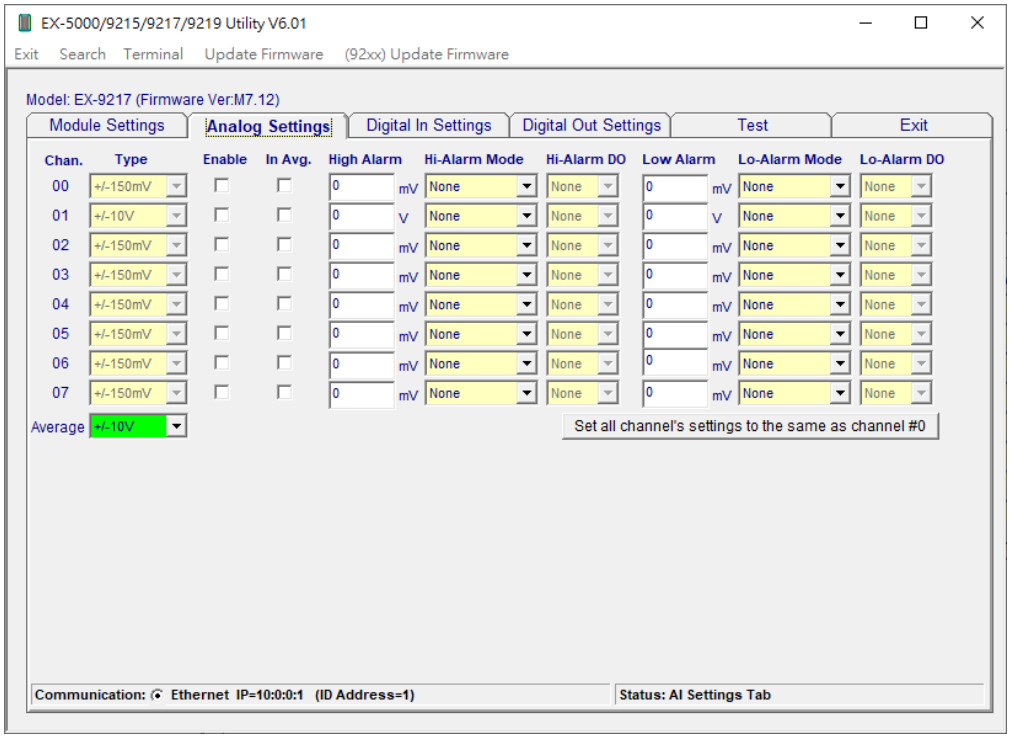
  Set Stream /Event data Destination IP

- **Stream/Event Enable Setting**: (default all disabled)

  Enable/disable Stream and Event functions

- **Stream time interval**: (default 10 sec)
  Set time interval for sending stream data
- **DI active state settings**:
  Set DI active state (High input active or low input active)
- **DO active state settings**:
  Set DO output active state (High/open output active or low output active)
- **Filter rate:**
  Select A/D converter filter frequency
- **ASCII Data Format**
  Select ASCII data format (Engineering format or 2's complement format)
- **Modbus Data Format**
  Select Modbus data format (Engineering format or 2's complement format)
- **A/D calibration button**
  Calibrate EX9217 analog channels(Span and zero calibrations)

19.7.2 Analog settings tab



- **Type**:
  Select analog input channel type
- **Enable button**:
  Enable/disable single channel
- **In Avg. button**:
  Enable/disable channel to be in average
- High Alarm Textbox:
  Set AI channel high alarm value
- Low Alarm Textbox:
  Set AI channel low alarm value

- High Alarm mode Combobox:

  Select AI channel high alarm event mode
  None =Disable alarm
  Momentary =Generate Alarm event only when Channel value is greater than High Alarm value
  Latch =Generate Alarm event when Channel value is greater than High Alarm value and latch alarm event until cleared by user)

- Low Alarm mode Combobox:

  Select AI channel low alarm event mode
  None =Disable alarm
  Momentary =Generate Alarm event only when Channel value is less than low Alarm value
  Latch =Generate Alarm event when Channel value is less than low Alarm value and latch alarm event until cleared by user)

- High Alarm DO Combobox:

  Select DO output channel when high alarm event occured
  None =No DO output
  DO 0 =DO 0 output

- Low Alarm DO Combobox:

  Select DO output channel when low alarm event occured
  None =No DO output
  DO 0 =DO 0 output

- **Set All channels to be the same settings as channel #0 button**:

  Set all AI channels to have the same settings as channel #0

19.7.3 Digital input settings tab

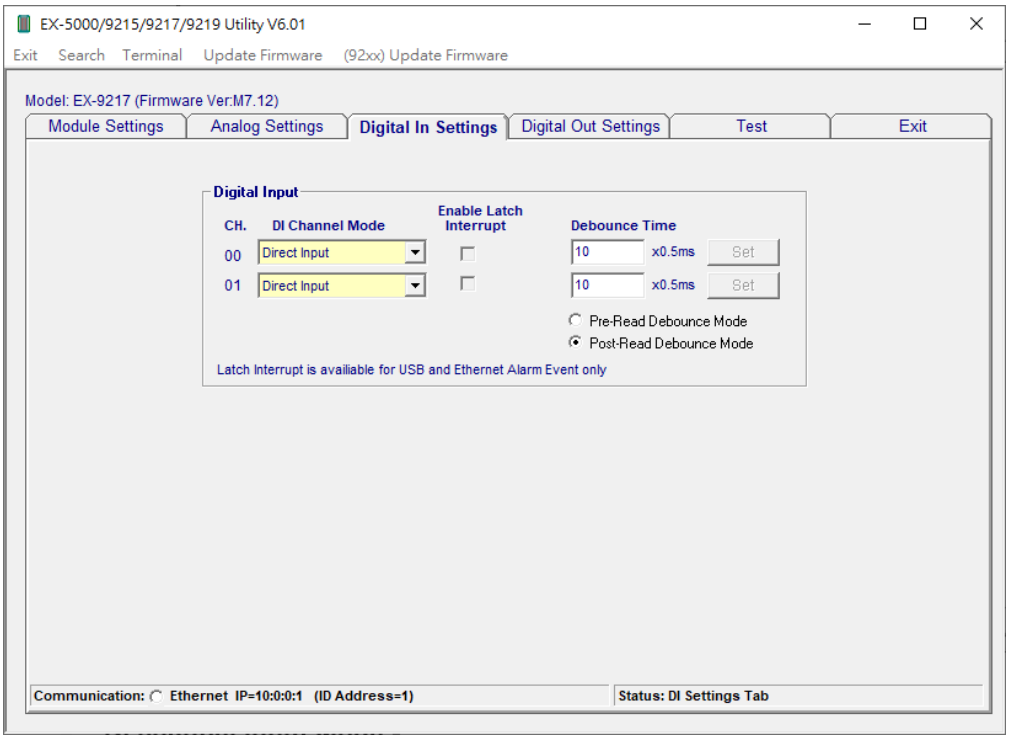


Figure - 15

- **DI channel Input Mode:**

  Direct input ,counter, low to high latch, high to low latch, frequency mode

- **Latch Interrupt:**

  Enable/Disable DI latch interrupt (USB connection only)

- **Debounce time:**

  Set DI input debounce time (0~65535). =0 no debounce

- **Select Pre-read debounce/Post read debounce mode (see 錯誤! 找不到參照來源。)**

  Pre-read mode

  The device read DI state immediately when DI state-changed and then delay 5000 msec to filter all other states changed in this time interval(**A/B/C/D** states are all ignored by device)

  Post-read mode

  The bounce detection is started when DI state-changed and then read final DI state after 5000 msec delay reached

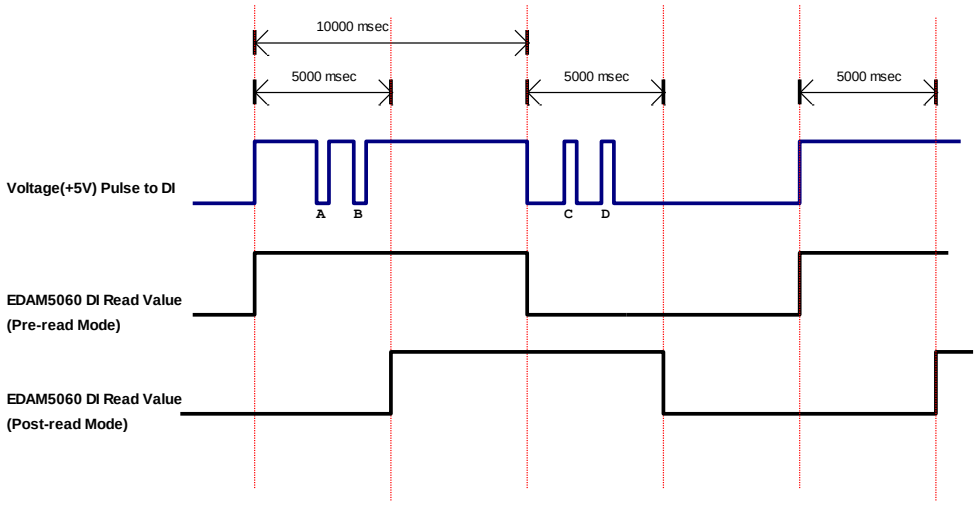


Figure - 16

- **Set all channels settings as channel 0**

  Configure all channel settings as channel 0 settings

19.7.4 Digital output settings tab

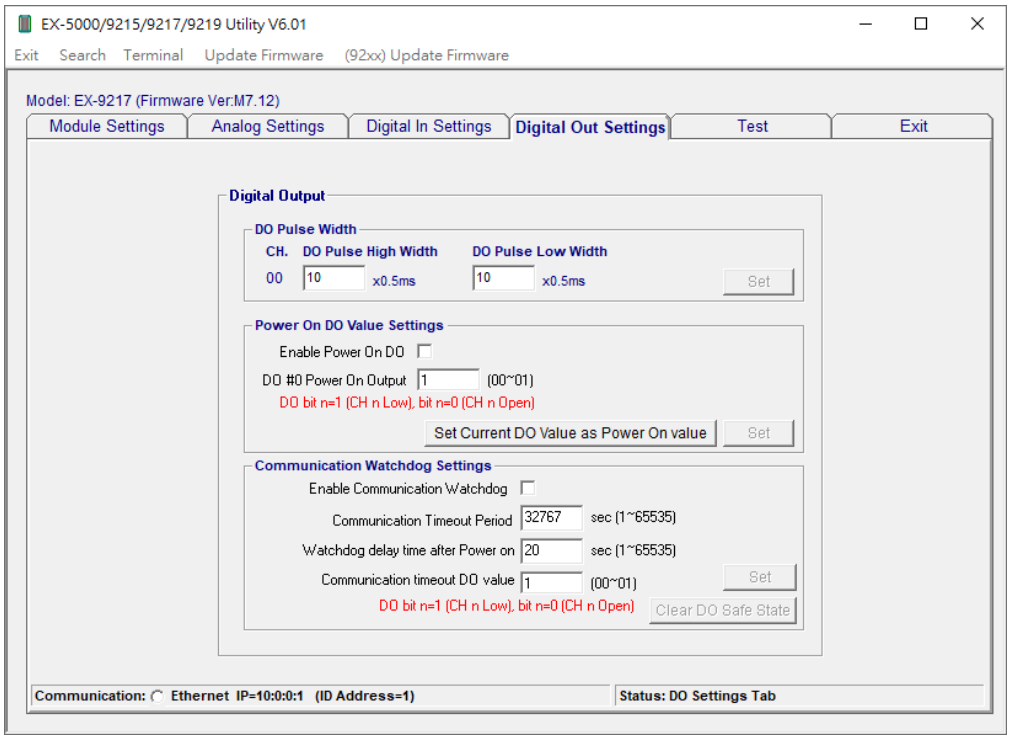


Figure - 17

- **DO pulse width**:

  Set high level width and low level width for DO pulse output operation

- **Communication watchdog**:
  - **Enable/disable communication watchdog**

    If there are no data received from host When communication timeout is reached, The DO will be set to specified safe value.
  - **Communication timeout period**

    Set Communication timeout value
  - **Delay time after power–on**

    The delay time to start communication watchdog function after module reboot
  - **Communication timeout DO safe value**

    DO value when communication timeout is reached
- **Enable power-on DO**:

  DO default value after module reboot
- **Power-on DO value**:

  Set Power-on DO value

19.7.5 Test tab

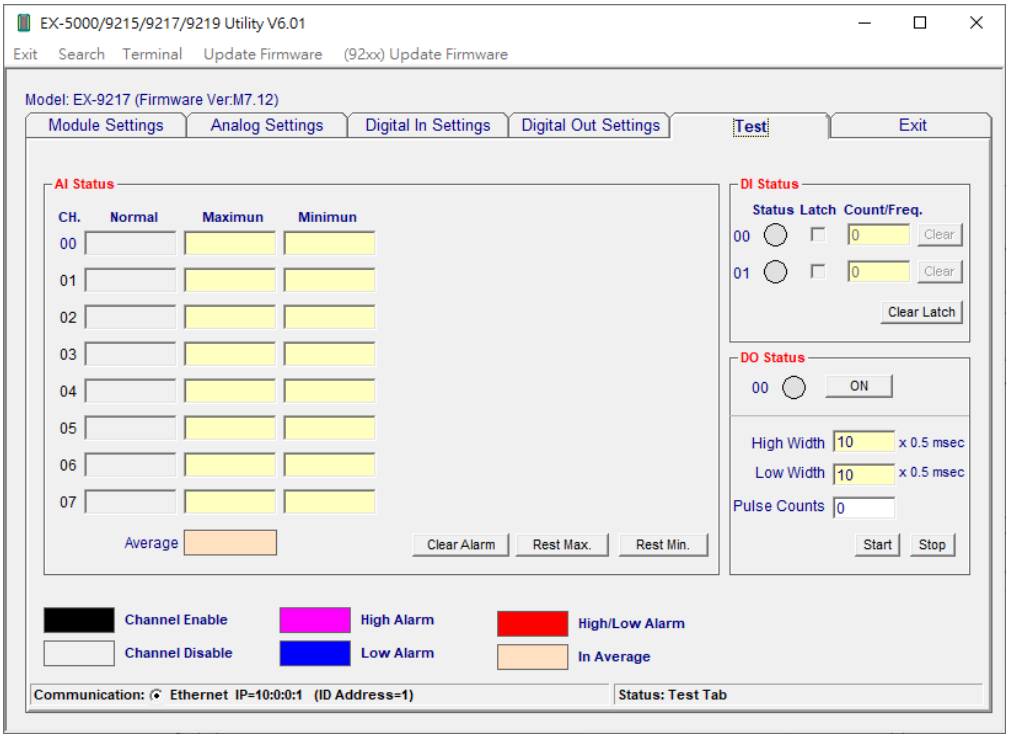


Figure - 18

- **Clear button**

  Clear single DI channel counter to zero
- **ON/OFF button**

  Toggle single DO channel output value
- **Counts TextBox**

  Enter the DO pulse counts
- **Start button**:

  Start to Generate DO pulse output

- **Stop button**:
  Stop DO pulse output
- **Clear Alarm button:**
  Set all analog channel alarm events
- **Resset Max. button:**
  Reset all ananlog Maximun value
- **Reset Min. button:**
  Reset all ananlog Minimun value

19.8 EX-9215 Configuration

19.8.1 Module settings tab

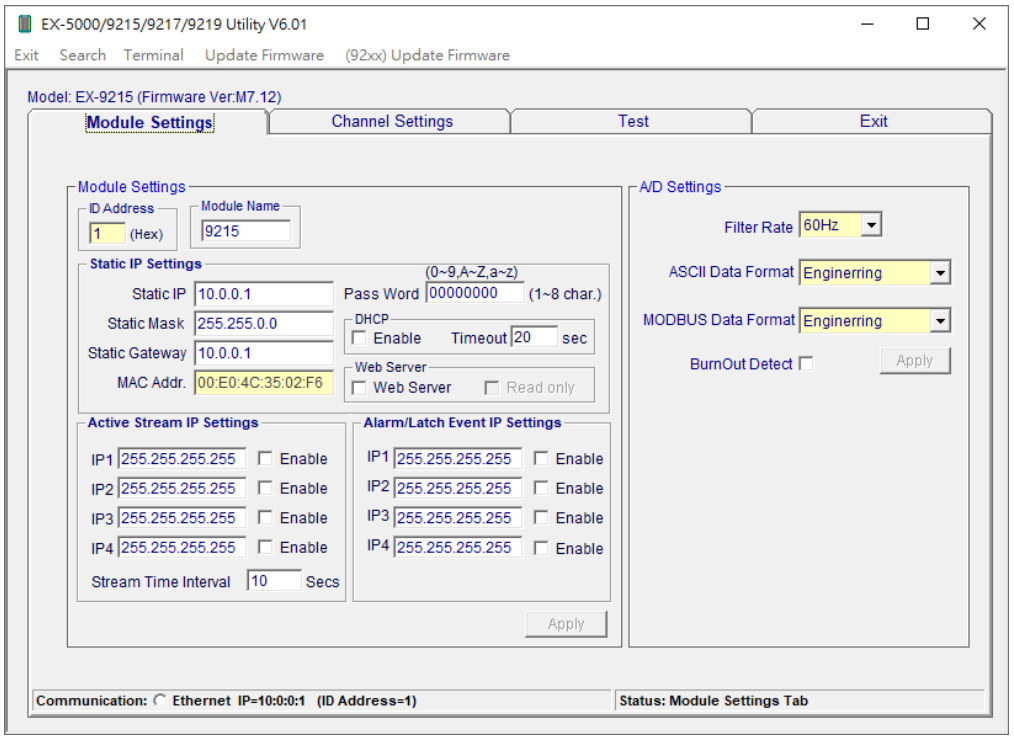


Figure - 19

- **MAC Address**:

  The Ethernet address and needs no further configuration.

- **IP Address**, **Subnet Mask**, and **Default Gateway**: (default 10.0.0.1, 255.0.0.0 and 0.0.0.0)

  The IP address identifies your EX-9200 devices on the global network. Each EX-9200 has same default IP address **10.0.0.1**. Therefore, *please do not initial many EX-9200 at the same time to avoid the Ethernet collision*. If you want to configure the EX-9200 in the host PC's dominating network, only the IP address and Subnet Mask will need to set (The host PC and EX Ethernet I/O must belong to same subnet Mask).
  If you want to configure the EX-9200 via Internet or other network domination, you have to ask your network administrator to obtain a specific IP and Gateway addresses, and then configure each EX-9200 with the individual setting.

- **DHCP:** (default Enabled)

  Allow you to get IP address from the DHCP servo without setting IP address by manual.

- **Web Server:** (default Enabled)

  Allow you monitor and control I/O status on EX-9000 modules remotely through web browser.

- **Module ID**: (default 01)

  Unique ID number of module can be set by DIP switch on the back side of module (See Help)

- **Password:** (default 00000000)

  Allow you to change the password of the module (needed for Ethernet connection only)

- **Stream/Event IP**:

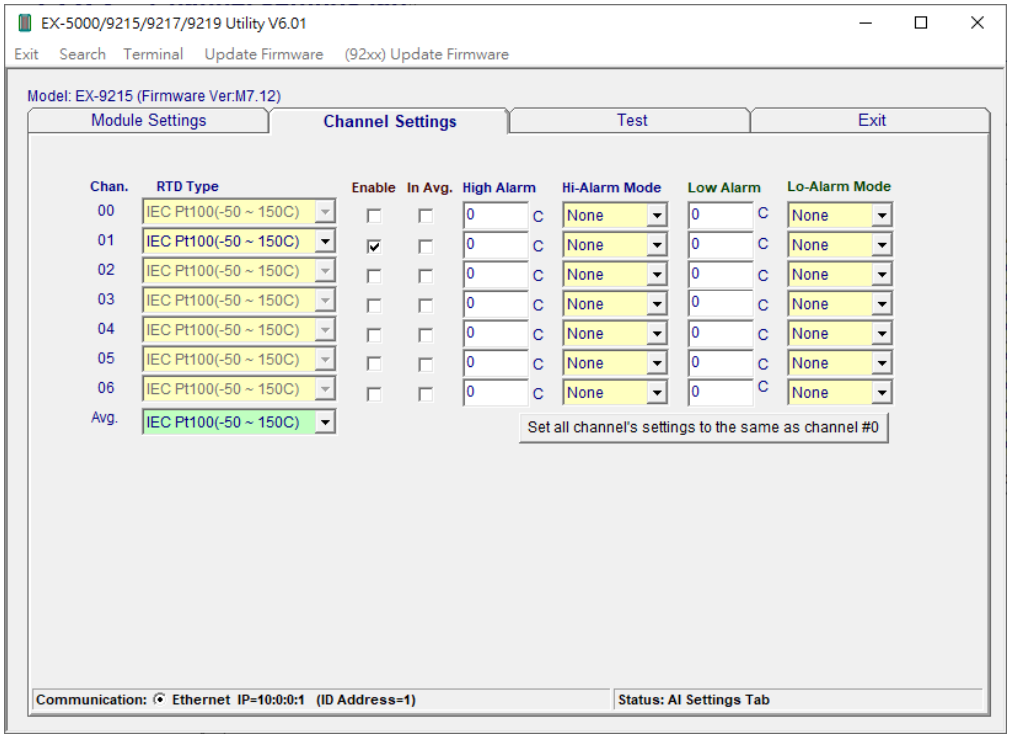
  Set Stream /Event data Destination IP

- **Stream/Event Enable Setting**: (default all disabled)

  Enable/disable Stream and Event functions

- **Stream time interval**: (default 10 sec)
  Set time interval for sending stream data
- **DI active state settings**:
  Set DI active state (High input active or low input active)
- **DO active state settings**:
  Set DO output active state (High/open output active or low output active)
- **Burn out detect:**
  Enable/disable input wire open detection
- **Filter rate:**
  Select A/D converter filter frequency
- **ASCII Data Format**
  Select ASCII data format (Engineering format or 2's complement format)
- **Modbus Data Format**
  Select Modbus data format (Engineering format or 2's complement format)
- **A/D calibration button**
  Calibrate EX9215 analog channels(Span and zero calibrations)

19.8.2 Channel settings tab



- **RTD Type**:
  Select analog input channel type
- **Enable button**:
  Enable/disable single channel
- **In Avg. button**:
  Enable/disable channel to be in average
- High Alarm Textbox:
  Set AI channel high alarm value

- Low Alarm Textbox:

  Set AI channel low alarm value

- High Alarm mode Combobox:

  Select AI channel high alarm event mode
  None =Disable alarm
  Momentary =Generate Alarm event only when Channel value is greater than High Alarm value
  Latch =Generate Alarm event when Channel value is greater than High Alarm value and latch alarm event until cleared by user)

- Low Alarm mode Combobox:

  Select AI channel high alarm event mode
  None =Disable alarm
  Momentary =Generate Alarm event only when Channel value is less than low Alarm value
  Latch =Generate Alarm event when Channel value is less than low Alarm value and latch alarm event until cleared by user)

19.8.3 Test tab

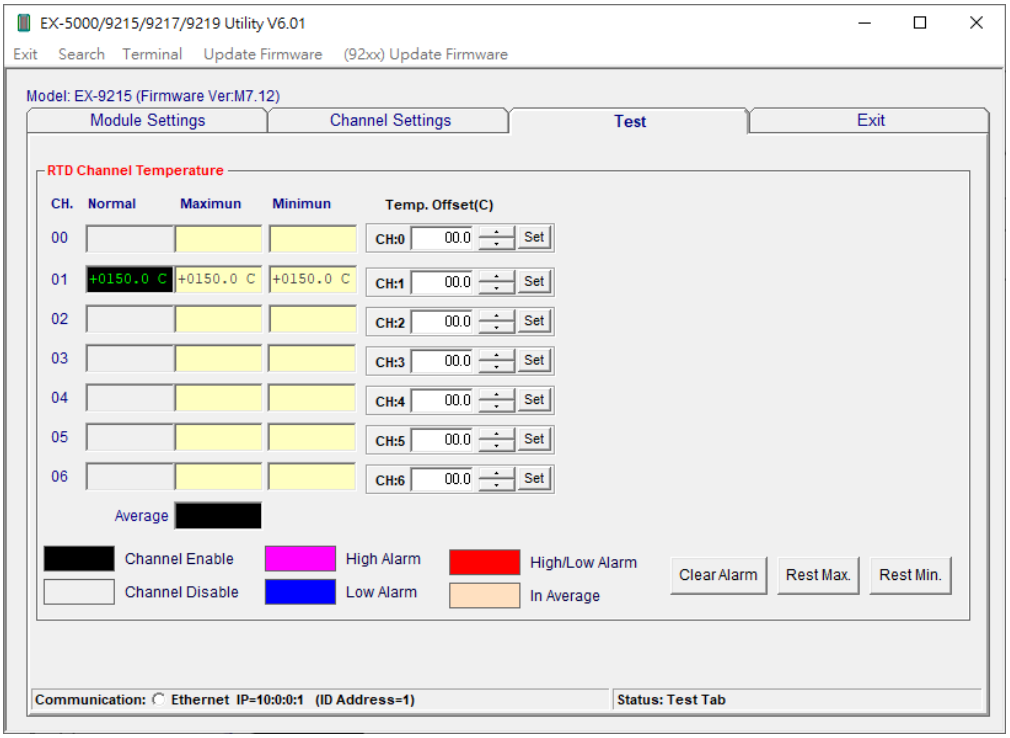


Figure - 20

- **Clear Alarm button:**

  Set all analog channel alarm events

- **Resset Max. button:**

  Reset all ananlog Maximun value

- **Reset Min. button:**

  Reset all ananlog Minimun value

Chapter 20 Reload Default Settings

All EX-5xxx modules provide a way to reload the default settings (see **錯誤! 找不到參照來源。**) as shown in Figure -1

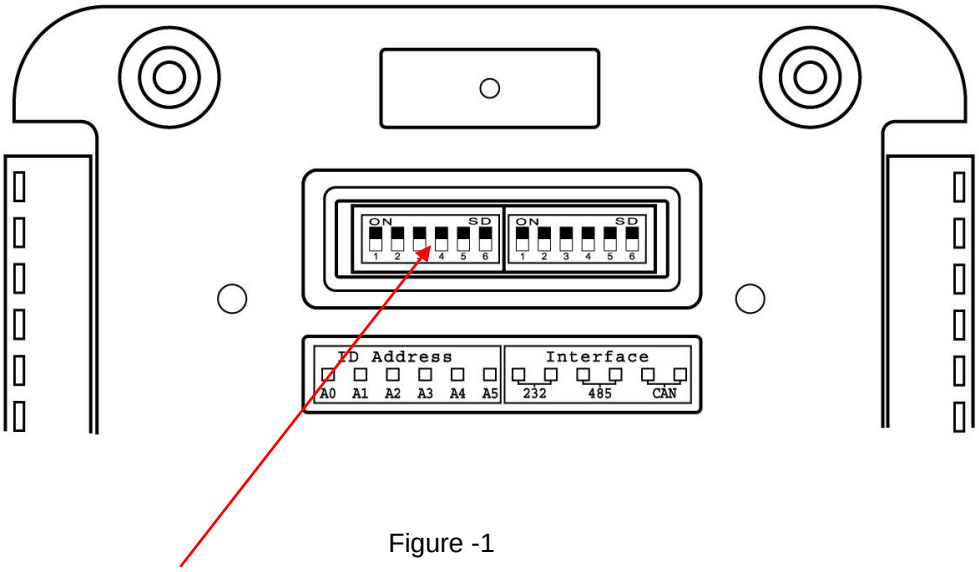


Figure -1

1、 Set all Pins of **ID address DIP switch** to off position (ID=00)

2、 Power off/on to re-boot the module and wait for a few seconds until USB LED or Ethernet LED turn-on

3、 Set the pins of **ID address DIP switch** to the desired position (ID=xx) (ID=0 is reserved for setting default only)

4、 Power off/on again to use default settings

Chapter 21 Zero/Span Calibration

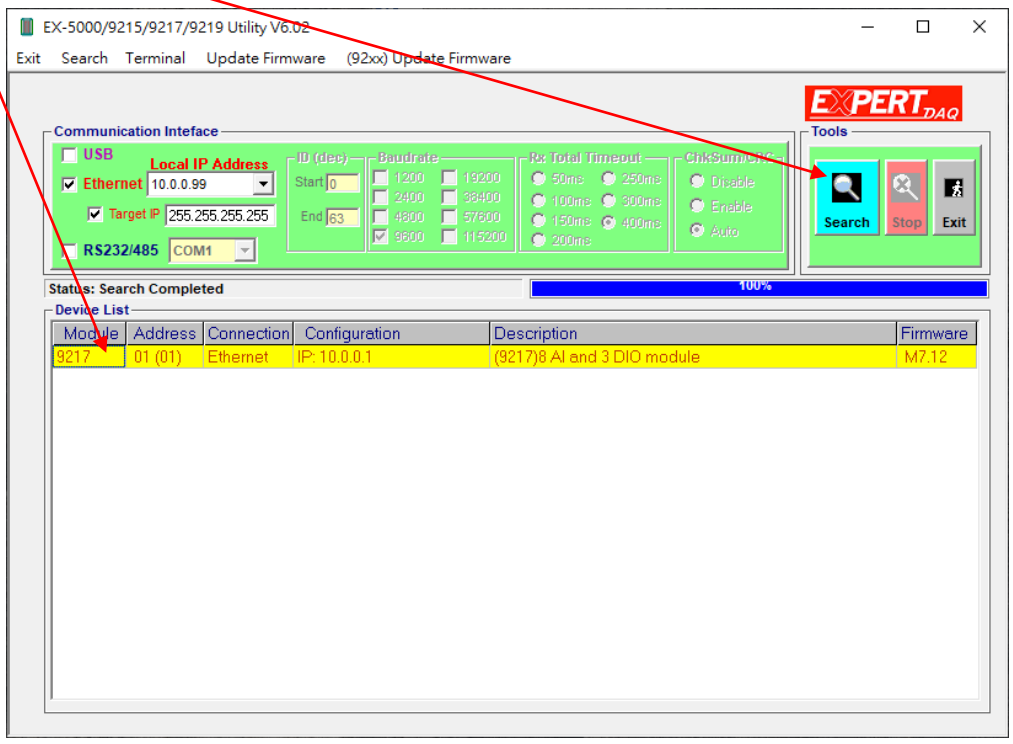
21.1 EX-9217 Calibration

1、 Connect EX9217 to Ethernet hub

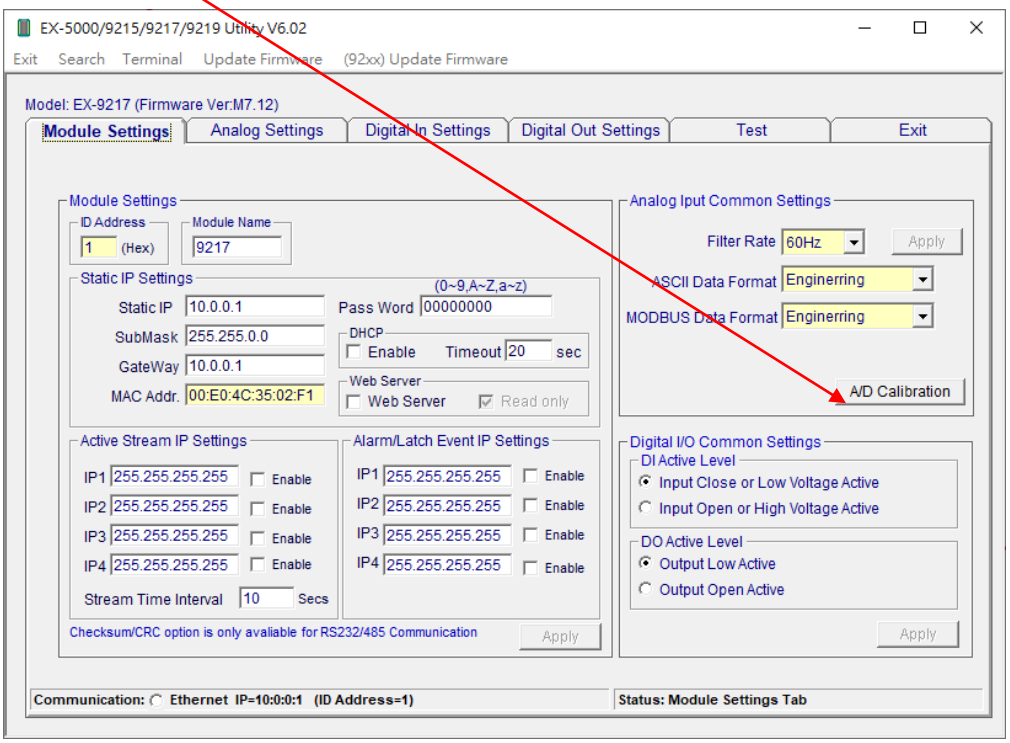
2、 Execute E9200Utility.

3、 Click "**Search**" button to search modules

4、 Double click EX-9217 listed in device list window



5、 Click "**A/D Calibration**" button



6、 Apply **0V** to channel **#0** (**AI0+, AI0-**) as shown in Figure -2

7、 Apply proper voltage(depend on the type been calibrated) to channel **#1** (**AI1+, AI1-**) as shown in Figure -2

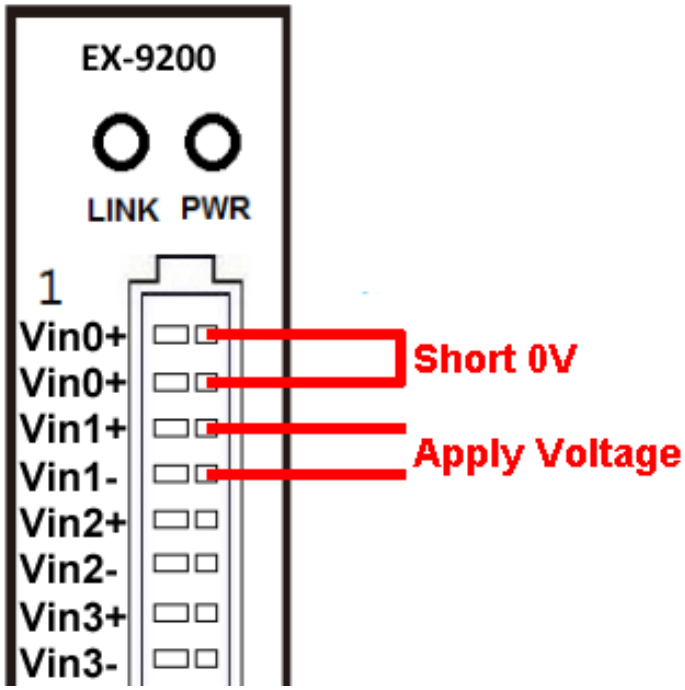
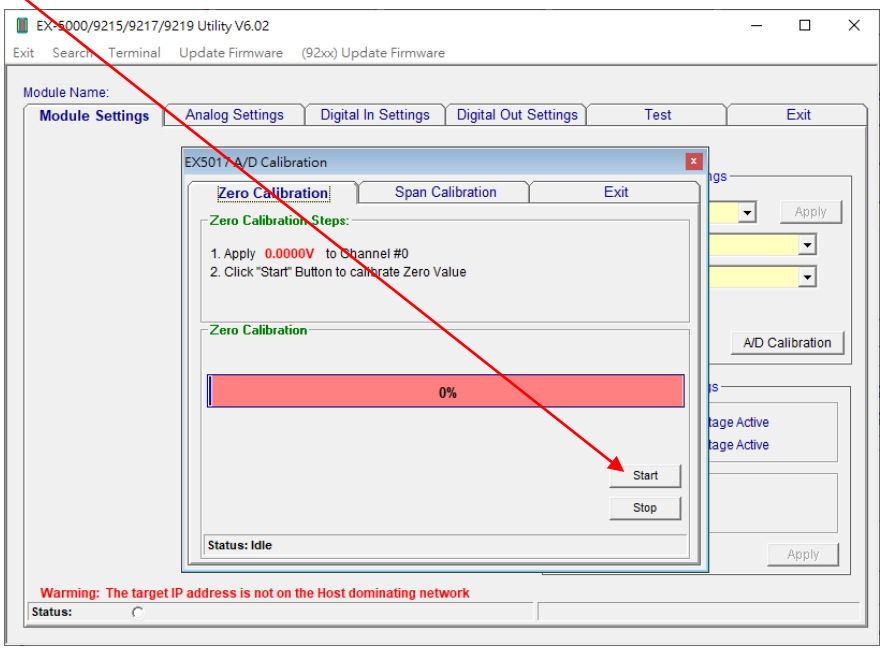


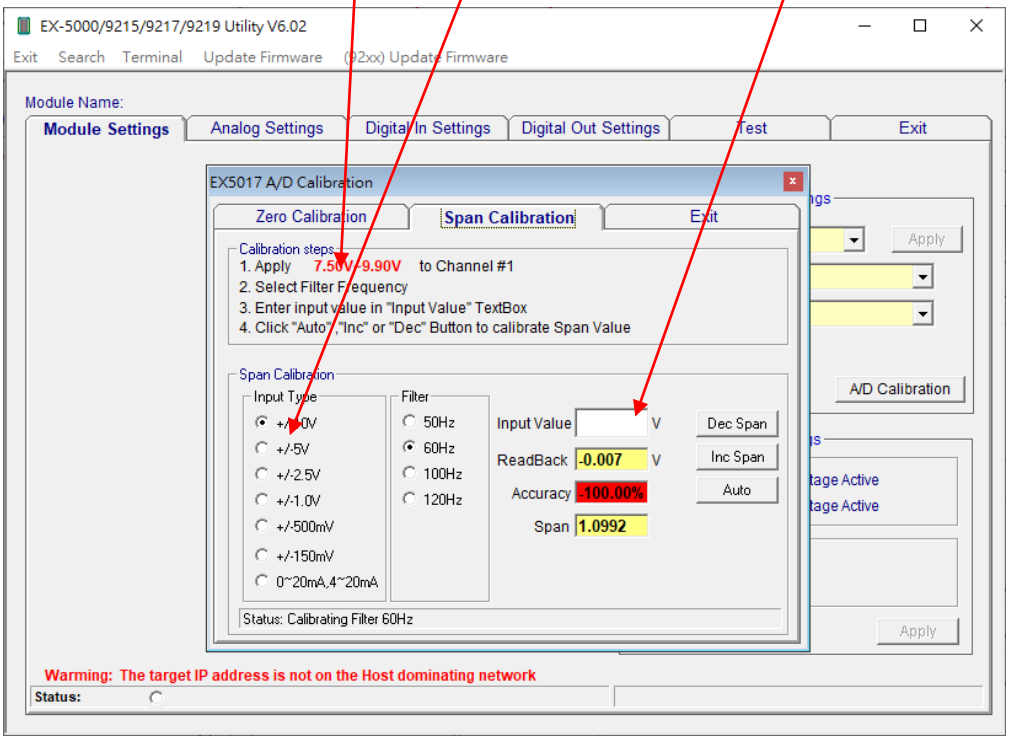
Figure -2

8、 Click "**Start**" button to calibrate Zero offset first



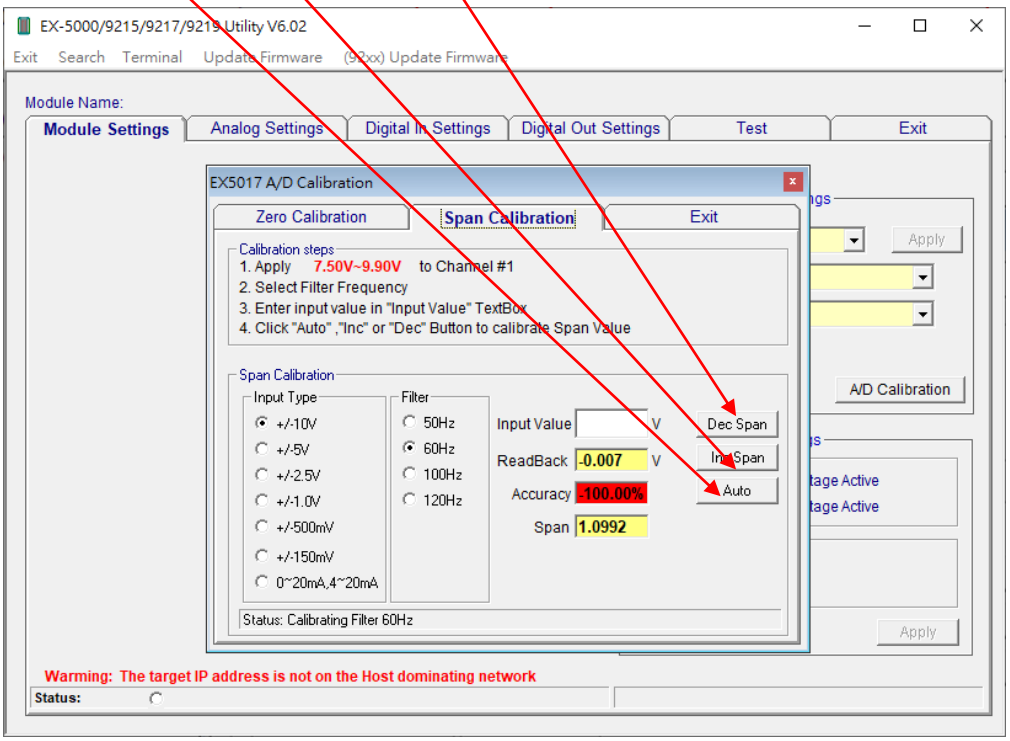
9、 Click "**Span Calibration**" tab and Select **"Input Type**"

10、 Enter the value of voltage(see input range) applied to channel #1 (**AI1+, AI1-**) in the "**Input Value**" textbox



11、 Click "**Auto**" button to start auto-calibrating
Auto-calibration will calibrate Span value of all Filter rate (50Hz, 60Hz, 100Hz, 120Hz)

12、 You can also click "**Inc Span**" or "**Dec Span**" button to fine adjust the Span value



13、 Cilck "**Exit**" button to exit calibration window

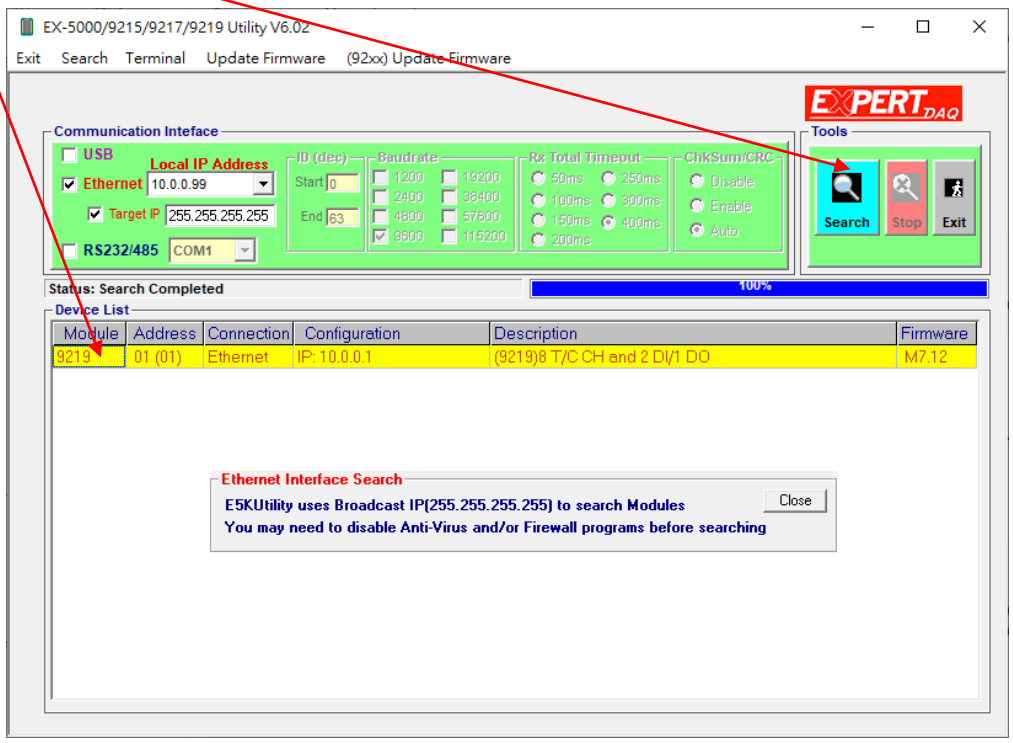
21.2 EX-9219 Calibration

1、 Connect EX9219 to Ethernet hub

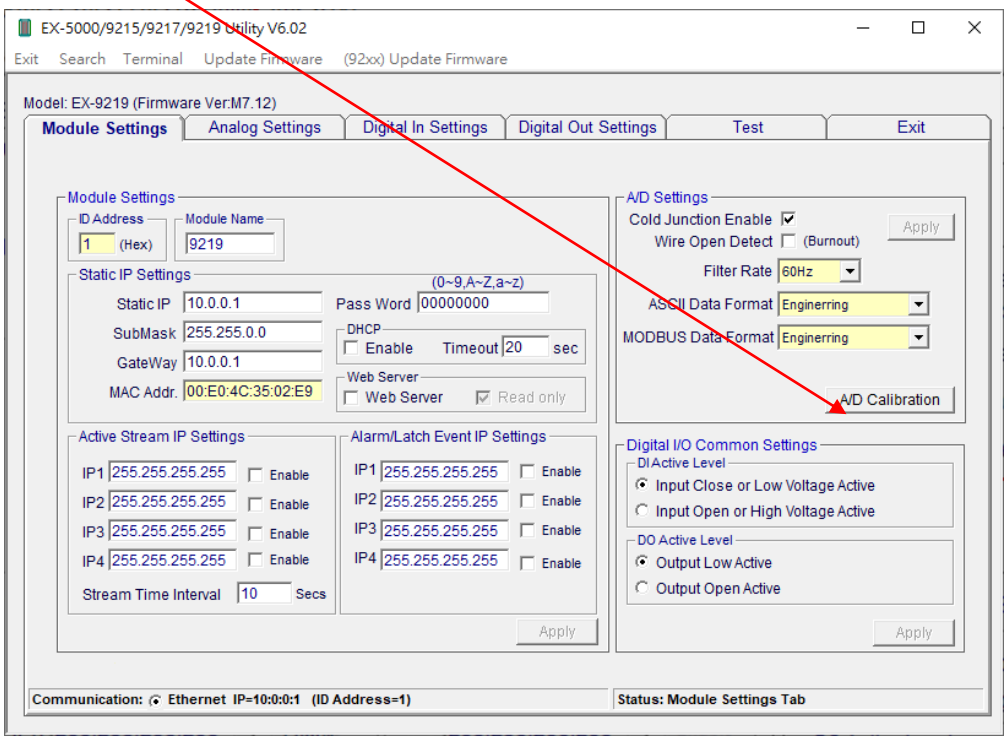
2、 Execute E9200Utility.

3、 Click "**Start**" button to search modules

4、 Double click EX-9219 listed in device list window



5、 Click "**A/D Calibration**" button



6、 Apply 0Vr to channel **#0** (**AI0+, AI0-**) as shown in Figure -3

7、 Apply proper voltage(depend on the type been calibrated) to channel **#1** (**AI1+, AI1-**) as shown in Figure -3

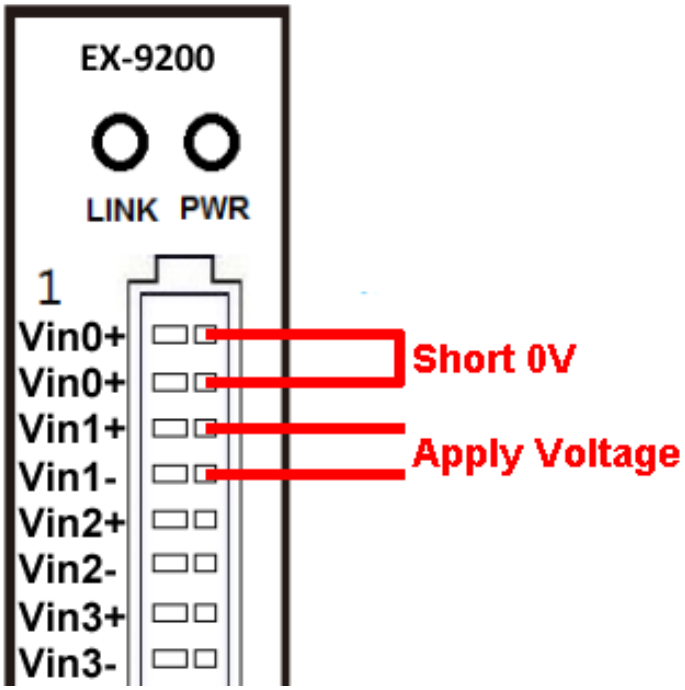
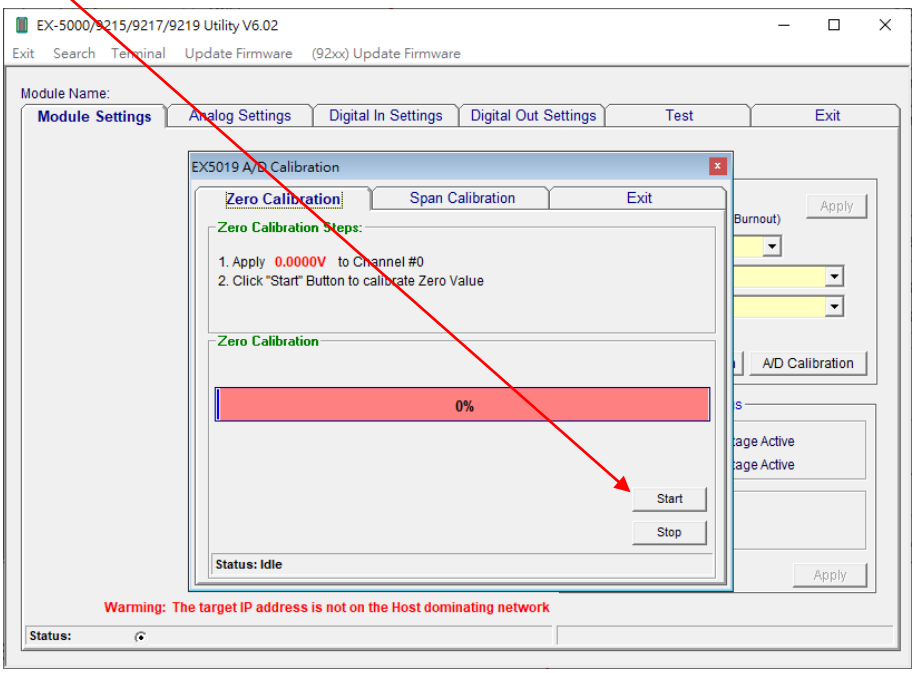


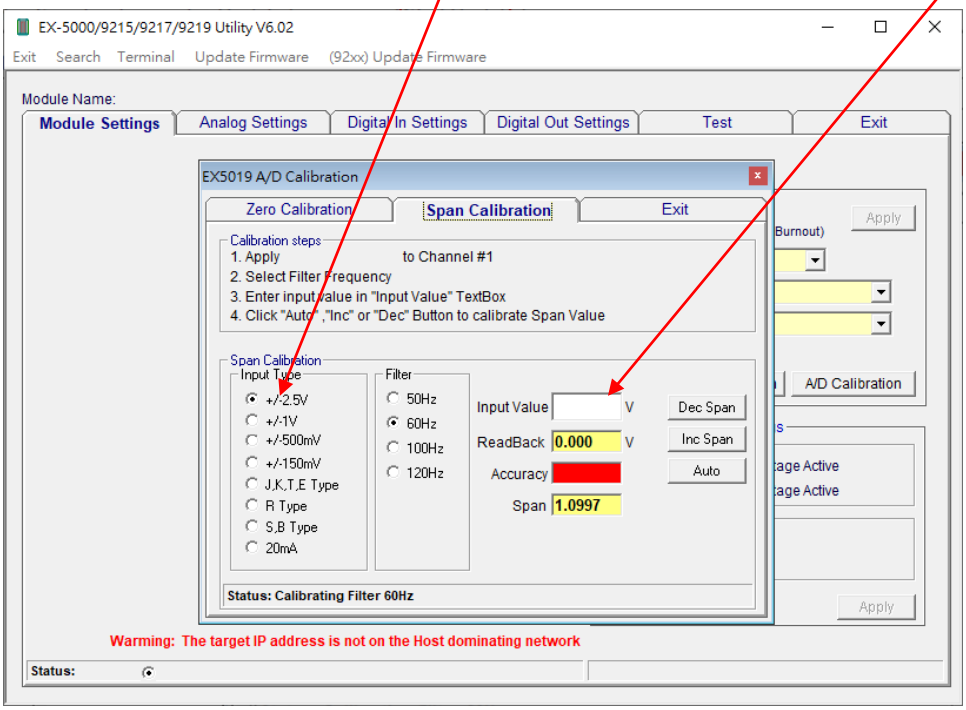
Figure -3

8、 Click "**Start**" button to calibrate Zero offset first



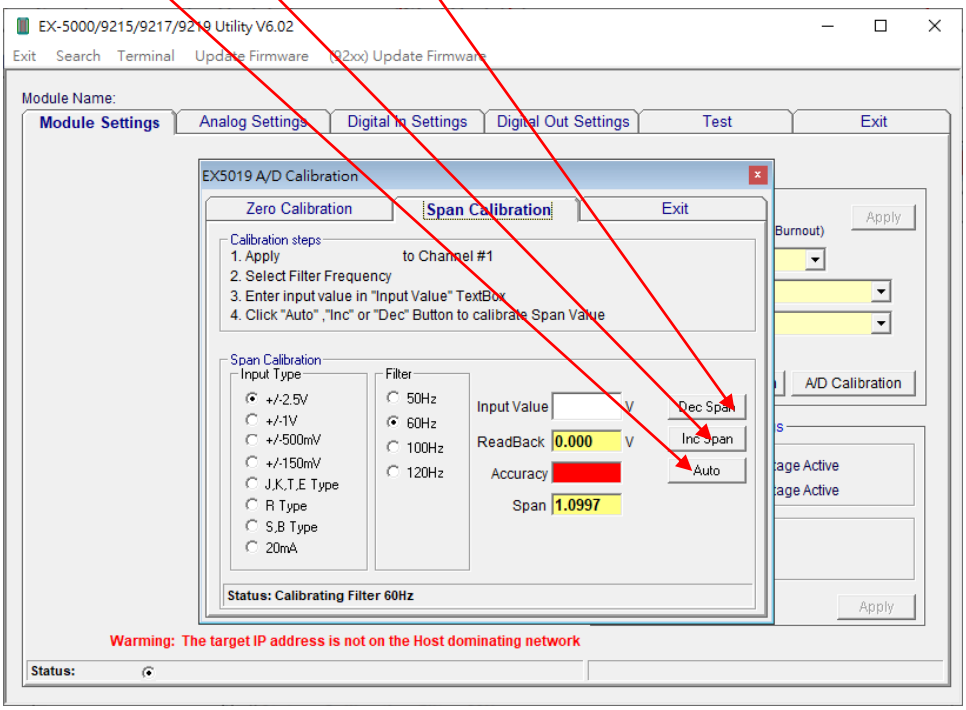
9、 Click "Span Calibration" tab and Select "Input Type"

10、 Enter the value of voltage(see input range) applied to channel #1 (CH1+, CH1-) in the "Input Value" textbox



11、 Click "**Auto**" button to start auto-calibrating
Auto-calibration will calibrate Span value of all Filter rate (50Hz, 60Hz, 100Hz, 120Hz)

12、 You can also click "**Inc Span**" or "**Dec Span**" button to fine adjust the Span value



13、 Cilck "**Exit**" button to exit calibration window