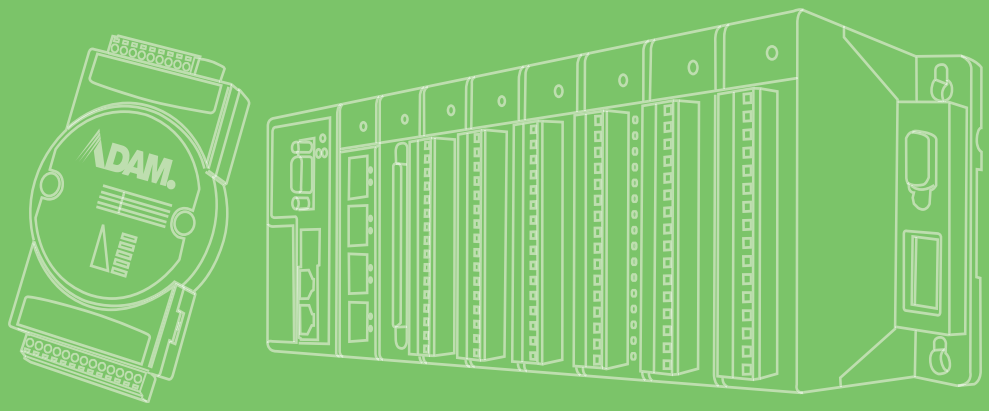


User Manual

ADAM-5000/ECAT

4-slot Distributed High Speed I/O System for EtherCAT

ADVANTECH

Enabling an Intelligent Planet

Copyright

Acknowledgements

IBM and PC are trademarks of International Business Machines Corporation.

All other product names or trademarks are properties of their respective owners.

EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

Product Warranty (2 years)

Advantech warrants to you, the original purchaser, that each of its products will be free from defects in materials and workmanship for two years 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 Advantech, or which have been subject to misuse, abuse, accident or improper installation. Advantech assumes no liability under the terms of this warranty as a consequence of such events.

Because of Advantech's high quality-control standards and rigorous testing, most of our customers never need to use our repair service. If an Advantech 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.

If you think you have a defective product, follow these steps:

1. Collect all the information about the problem encountered. (For example, CPU speed, Advantech products used, other hardware and software used, etc.) Note anything abnormal and list any onscreen messages you get when the problem occurs.
2. Call your dealer and describe the problem. Please have your manual, product, and any helpful information readily available.
3. If your product is diagnosed as defective, obtain an RMA (return merchandize authorization) number from your dealer. This allows us to process your return more quickly.
4. Carefully pack the defective product, a fully-completed Repair and Replacement Order Card and a photocopy proof of purchase date (such as your sales receipt) in a shippable container. A product returned without proof of the purchase date is not eligible for warranty service.
5. Write the RMA number visibly on the outside of the package and ship it prepaid to your dealer.

Part No. 2003M50000 Edition 1

Printed in Taiwan January 2015

Declaration of Conformity

CE

This product has passed the CE test for environmental specifications when shielded cables are used for external wiring. We recommend the use of shielded cables. This kind of cable is available from Advantech. Please contact your local supplier for ordering information.

FCC Class A

Note: This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

FM

This equipment has passed the FM certification. According to the National Fire Protection Association, work sites are classified into different classes, divisions and groups, based on hazard considerations. This equipment is compliant with the specifications of Class I, Division 2, Groups A, B, C and D indoor hazards.

Technical Support and Assistance

1. Visit the Advantech web site at www.advantech.com/support where you can find the latest information about the product.
2. Contact your distributor, sales representative, or Advantech's customer service center for technical support if you need additional assistance. Please have the following information ready before you call:
   - Product name and serial number
   - Description of your peripheral attachments
   - Description of your software (OS, version, application software, etc.)
   - A complete description of the problem
   - The exact wording of any error messages

Safety Precaution - Static Electricity

Follow these simple precautions to protect yourself from harm and the products from damage.

- To avoid electrical shock, always disconnect the power from your PC chassis before you work on it. Don't touch any components on the CPU card or other cards while the PC is on.

Disconnect power before making any configuration changes. The sudden rush of power as you connect a jumper or install a card may damage sensitive electronic components.

Safety Instructions

1. Read these safety instructions carefully.
2. Keep this User Manual for later reference.
3. Disconnect this equipment from any AC outlet before cleaning. Use a damp cloth. Do not use liquid or spray detergents for cleaning.
4. For plug-in equipment, the power outlet socket must be located near the equipment and must be easily accessible.
5. Keep this equipment away from humidity.
6. Put this equipment on a reliable surface during installation. Dropping it or letting it fall may cause damage.
7. The openings on the enclosure are for air convection. Protect the equipment from overheating. DO NOT COVER THE OPENINGS.
8. Make sure the voltage of the power source is correct before connecting the equipment to the power outlet.
9. Position the power cord so that people cannot step on it. Do not place anything over the power cord.
10. All cautions and warnings on the equipment should be noted.
11. If the equipment is not used for a long time, disconnect it from the power source to avoid damage by transient overvoltage.
12. Never pour any liquid into an opening. This may cause fire or electrical shock.
13. Never open the equipment. For safety reasons, the equipment should be opened only by qualified service personnel.
14. If one of the following situations arises, get the equipment checked by service personnel:
15. The power cord or plug is damaged.
16. Liquid has penetrated into the equipment.
17. The equipment has been exposed to moisture.
18. The equipment does not work well, or you cannot get it to work according to the user's manual.
19. The equipment has been dropped and damaged.
20. The equipment has obvious signs of breakage.
21. DO NOT LEAVE THIS EQUIPMENT IN AN ENVIRONMENT WHERE THE STORAGE TEMPERATURE MAY GO BELOW -20° C (-4° F) OR ABOVE 60° C (140° F). THIS COULD DAMAGE THE EQUIPMENT. THE EQUIPMENT SHOULD BE IN A CONTROLLED ENVIRONMENT.
22. CAUTION: DANGER OF EXPLOSION IF BATTERY IS INCORRECTLY REPLACED. REPLACE ONLY WITH THE SAME OR EQUIVALENT TYPE RECOMMENDED BY THE MANUFACTURER, DISCARD USED BATTERIES ACCORDING TO THE MANUFACTURER'S INSTRUCTIONS.
23. The sound pressure level at the operator's position according to IEC 704-1:1982 is no more than 70 dB (A).

DISCLAIMER: This set of instructions is given according to IEC 704-1. Advantech disclaims all responsibility for the accuracy of any statements contained herein.

Chapter 1

Understanding Your System

1.1 Introduction

EtherCAT (Ethernet Control Automation Technology) is a high-performance, Ethernet-based fieldbus industrial network system. The protocol is standardized in IEC 61158 and apply to automation applications that need faster and more efficient communications. Short data update times with precise synchronization make EtherCAT suitable for real-time requirements in automation technology.

1.1.1 EtherCAT Features

Functional Principle

In EtherCAT network, master sends an Ethernet frames pass through all of the slave nodes. The Standard Ethernet packet or frame is no longer received, interpreted, and copied as process data at every node. Instead, slave devices read the data addressed to them and input data are also inserted in the same time while the telegram passes through the device, processing data "on the fly". Typically the entire network can be addressed with just one frame.

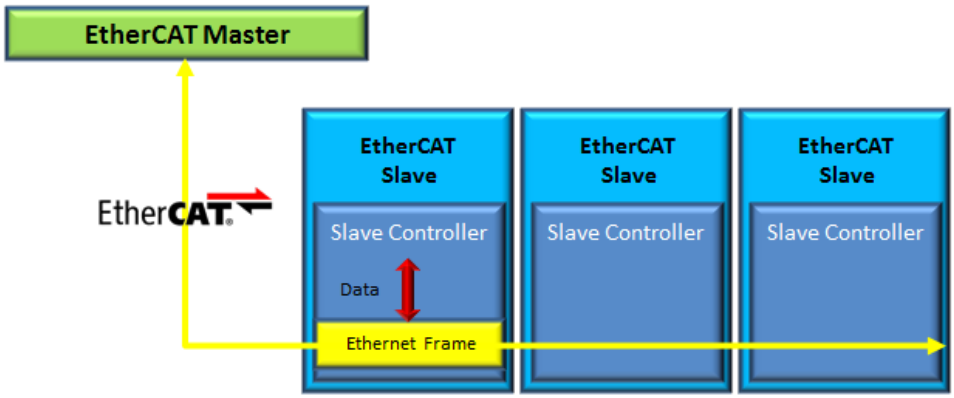


Figure 1.1 EtherCAT Function Principle

Protocol

Data exchange are cyclically updated between EtherCAT master and slaves. Data in EtherCAT frames are transported directly within the standard IEEE 802.3 Ethernet frame using Ethertype 0x88a4 and are processed by the EtherCAT Slave Controller on the fly.

Each EtherCAT datagram is a command that consists of a header, data and a working counter.. The datagram header indicates what type of access the master device would like to execute:

- Read, write, read-write
- Access to a specified slave device through direct addressing
- Access to multiple slave devices through logical addressing

Logical addressing is used for the cyclical exchange of process data. The header and data are used to specify the operation that the slave must perform, and the working counter is updated by the slave to let the master to know that a slave has processed the command.

Every EtherCAT datagram ends with a 16 Bit Working Counter (WKC). The Working Counter counts the number of devices that were successfully addressed by this EtherCAT datagram.

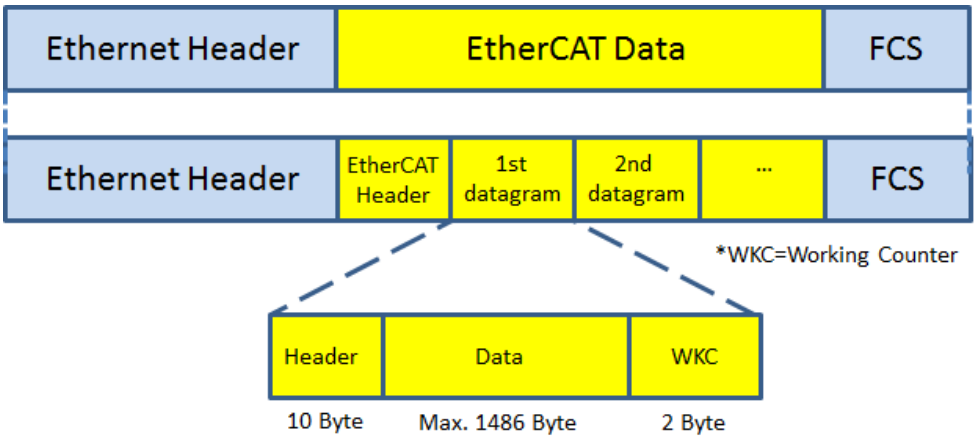


Figure 1.2 EtherCAT Protocol

EtherCAT datagrams are processed before receiving the complete frame.
In case data is invalid, the **frame check sum (FCS)** is not valid and the slave will not set data valid for the local application.

Topology

EtherCAT supports a variety of network topologies, including line, tree, ring and star. The line and tree topologies are more conducive to fieldbus applications because they require fewer connections and utilize a much simpler and more flexible cabling schema that switches and hubs are not necessary for lines or trees topology.

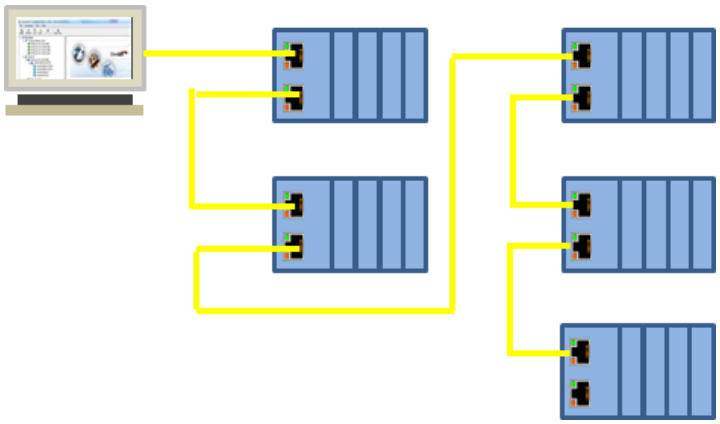


Figure 1.3 EtherCAT Topology

Inexpensive industrial Ethernet cable can be used between two nodes up to 100m apart in 100BASE-TX mode. EtherCAT makes a pure bus or line topology with hundreds of nodes possible without the limitations. Up to 65,535 devices can be connected to EtherCAT, so network expansion is almost unlimited.

EtherCAT protocol supports individual nodes to be connected and disconnected during operation. If one of slaves in network is removed, the rest of the network can continue to operate normally. Additionally, EtherCAT also enables other communication features such as cable redundancy or even master redundancy with Hot Standby.

Synchronization

Distributed Clocks (DC) mechanism is used to provide highly precise time synchronization between slaves in an EtherCAT network, which is equivalent to the IEEE 1588 Precision Time Protocol standard. By using distributed clocks, EtherCAT is able to synchronize the time in all local bus devices within a very narrow tolerance range. All EtherCAT slaves are provided with an internal clock which named as System Time ($t_{Local\ Time}$). One EtherCAT Slave, usually the first slave, will be used as a Reference Clock and distributes its Clock cyclically.

Possible misalignments between the reference clock and the clocks of the other slaves are usually due to the following reason: when a slave is switched on, the internal free-running register that holds the current time is reset to zero. Unfortunately, this action does not take place exactly at the same time in all the different slaves, and this result in an initial offset (t_{offset}) among clocks that has to be compensated.

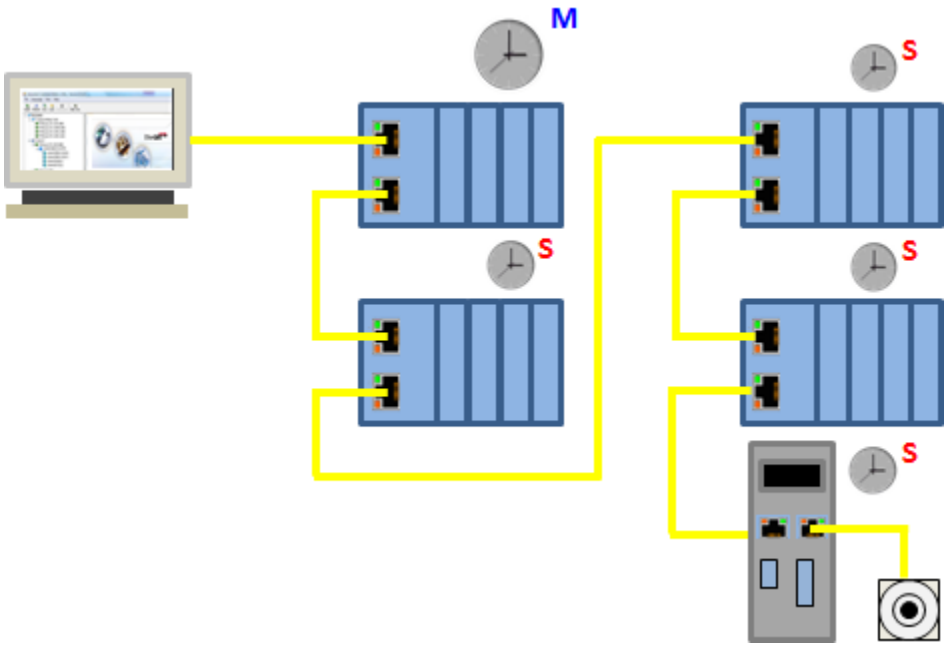


Figure 1.4 EtherCAT Distributed Clock

Typically, master sends a broadcast to all other slaves in the system. Once receiving the message, slaves will latch the value of their internal clock. There are two latch values, one is receiving and the other is returning back. Thus, the master can read all latched values and calculate the delay for each slave ($t_{\text{Propagation Delay}}$). Delays will be stored into offset register. In the following, master will send a message periodically to all other slaves in EtherCAT network to make the first slave the reference clock and forcing all other slaves to set their internal clock by the calculated offset.

$$\Delta t = \left(t_{Local\ Time} + t_{Offset} - t_{Propagation\ Delay}\right) - t_{Received\ System\ Time}$$

Because synchronization between slaves in DC mode is done by internal clocks in hardware, EtherCAT guarantee the time jitter is less than 1us.

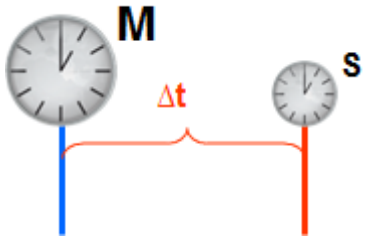


Figure 1.5 EtherCAT Distributed Clock Jitter

Diagnosis with exact localization

EtherCAT is an ultra -fast I/O system. To reach the best high-speed communication, high communication accuracy is demanded. EtherCAT comprises a wide range of system-inherent diagnostic features which help detect and locate system errors precisely. Apart from broken wire detection and localization, the protocol, physical layer and topology of the EtherCAT system enable individual quality monitoring of each individual transmission segment.

As mentioned, every EtherCAT datagram ends with a 16 bit **Working Counter (WKC)** to count the number of devices that were successfully addressed by this EtherCAT datagram. Master can check the data exchange situation by WKC in the same cycle and the error frame can be detected by analyzing the nodes' error counters. The slave application will be executed only as the frame is received correctly.
The automatic evaluation of the associated error counters enables precise localization of critical network sections.

Bit errors during transmission are detected reliably by the analysis of the **CRC (Cyclic Redundancy Check)** check sum. CRC is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data.
In addition to the error detection and localization protocol, transmission physics and topology of the EtherCAT system allow an individual quality monitoring of every single transmission path. There is a very effective monitoring mechanism in EtherCAT.

1.2 Major Features

ADAM-5000/ECAT is an EtherCAT slave that is created by mounting a group of ADAM-E5000 IO series modules on it. There are 4 slots on ADAM-5000/ECAT so that IO modules can be flexibly combined to achieve the optimum EtherCAT slave for the application with simple wiring and space-saving.

The features of the EtherCAT Slave are described below.

1.2.1 High performance 32-bit ARM RISC Processor

EtherCAT functionality is integrated into the 32-bit ARM Cortex-A8 processors. These devices integrate an ARM processing core with a slew of other peripherals and interfaces that make them attractive devices for building industrial automation equipment.

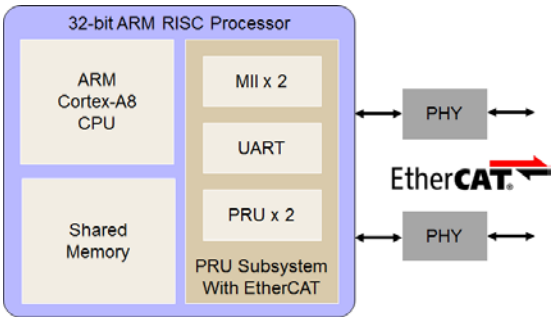


Figure 1.6 EtherCAT Slave Controller

This processor integrates the **programmable real-time unit (PRU)** subsystem, which supports very low-level interaction with the MII interfaces. This processor support multiple operating frequency ranges from 300 MHz for simple applications up to 1 GHz for complex applications that require high performance and is configured with one PRU coprocessor (two real-time cores). The PRU processes EtherCAT telegrams on-the-fly, parses them, decodes the address and executes EtherCAT commands.

In EtherCAT Layer 2, the PRU real-time cores share the tasks of datagram processing, distributed clocking, address mapping, error detection and handling and host interface.

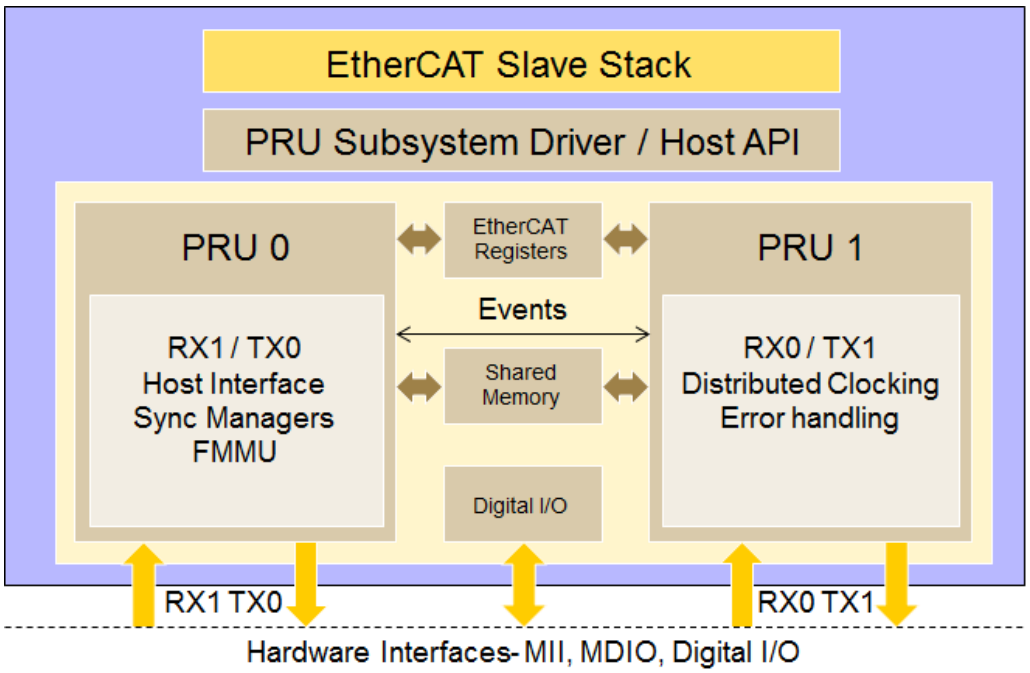


Figure 1.7 EtherCAT Firmware Architecture

PRUs also emulate EtherCAT register space in the internal shared memory. With their deterministic real-time processing capability, the PRUs handle EtherCAT datagrams with consistent and predictable processing latency.

1.2.2 Multi-IO modules are just a single EtherCAT node on the network

The **Modular Device Profile** (MDP) defines a modeling of structures within in a device. Mainly the object dictionary structure and corresponding behavior of the entries is defined by the MDP. The intention is to provide an easy way for master and configuration tools to handle the devices. A modular structure can be used for all kind of devices that supports physical or logical modules.

There are three types:

- Gateways from Fieldbus to EtherCAT
- Extendable bus coupler with an internal backbone
- Multi Axis servo drives (physical modules) with each axis having independent operation modes (logic modules)

ADAM-5000/ECAT as an extendable bus main unit with an internal backplane supports the MDP and provides a basic structure for masters and configuration tools to handle slaves with complex (modular) structure easily.

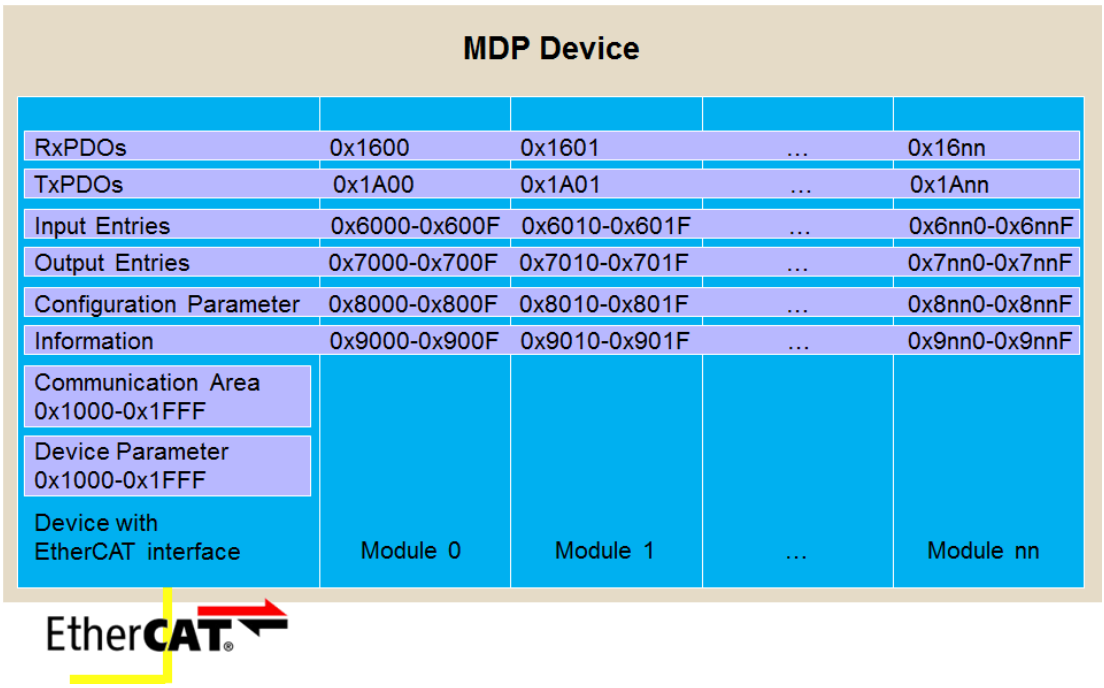


Figure 1.8 Model of a Modular Device

Mainly the object dictionary structure and corresponding behavior of the entries is defined by the MDP.

1.2.3 Supports EtherCAT Free-Run and Distributed Clock (DC) mode

EtherCAT telegrams are sent by the EtherCAT master to all connected EtherCAT slaves. All EtherCAT slaves perform continually synchronous/ asynchronous sequences of calculations and/or data copying actions.

There are two modes for EtherCAT to do data transferring. One is **Free-Run mode** and the other is **DC mode**.

Free-Run mode

In "Free Run" mode the local cycle is triggered through a local timer interrupt and the cycle time can be modified by the master (optional) in order to change the timer interrupt. Local cycle operates independent of the communication cycle and the slave refreshes the I/O data asynchronous to the master.

DC mode

In this mode, the slave refreshes the I/O data in synchronization with the communication cycle of the master. Distributed Clocks are used for very precise timing

requirements and for timing signals that can be generated independent of the communication cycle.

In this mode, a mechanism called a **Distributed Clock (DC)** is used to synchronize EtherCAT communications. The clock is shared to synchronize the master and the slaves. Interruptions (Sync0) are generated in the slaves at precise intervals based on this clock. Each slave executes I/O processing at this precise time. This means that all EtherCAT devices can share the same EtherCAT system time (global time base) for synchronization.

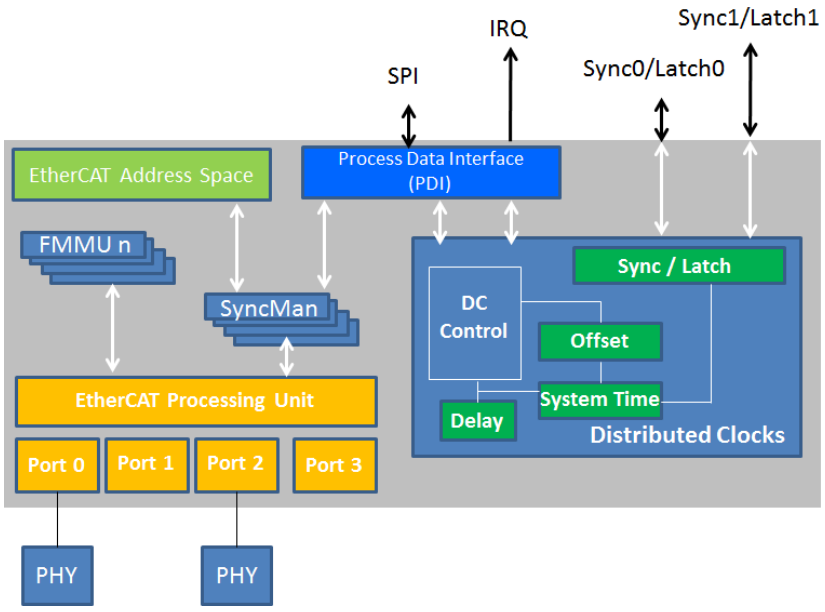


Figure 1.9 Interfaces of the distributed clock unit

Note!

If EtherCAT master accesses ADAM-5000/ECAT with analog input / output modules , there is ADC/DAC limitation in the existing module hardware design and can't support read/write data through local bus in specified short communication cycle time. Master will not acquire the right information before data completely transferring. For the low-response module like A I/O, ADAM-5000/ECAT can dispatch write command or response prior-time of data.

1.2.4 Supports EtherCAT SyncManager mode

EtherCAT provides a mechanism to synchronize slave memory access. Because the memory of a slave can be used for exchanging data between the master and the local application without any restrictions.

Since both the EtherCAT network (Master) and the PDI (Slaves) access the DPRAM in the ESC, the DPRAM access needs to ensure data consistency. The SyncManager

enables consistent and secure data exchanges between the EtherCAT master and the slaves, generating interrupts to inform both sides of changes to protect data in the DPRAM from being accessed simultaneously. If the slave uses FMMUs, the SyncManagers for the corresponding data blocks are located between the DPRAM and the FMMU.

EtherCAT SyncManagers can operate in two modes:

Mailbox mode

The mailbox mode implements a handshake mechanism for data exchange. EtherCAT master and slave only get access to the buffer after the other one has finished its access. When the sender writes the buffer, the buffer is locked for writing until the receiver has read it out. The mailbox mode is typically used for parameter settings and no real-time exchange of data.

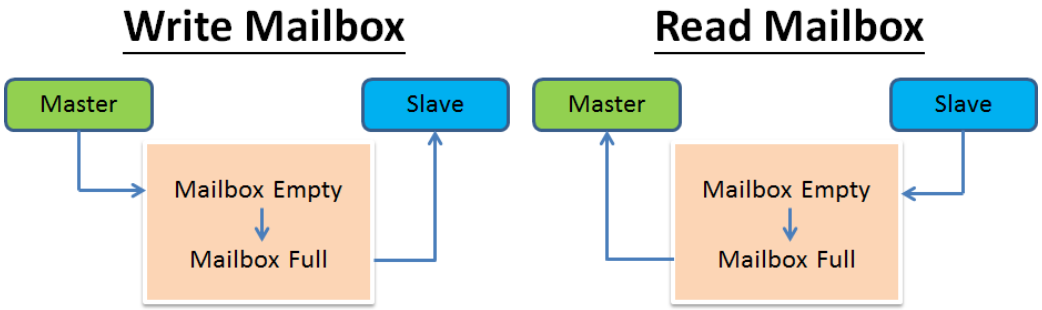


Figure 1.10 Mailbox Mode

Buffered Mode

The buffered mode is typically used for cyclic data exchange because the buffered mode allows access to the communication buffer at any time for both sides, EtherCAT master and slaves.

Note!

SyncManagers running in buffered mode need three times the process data size allocated in the DPRAM.

Note!

If the buffer is written faster than it is read out by the receiver, old data is dropped. Thus, the receiver always gets the latest consistent buffer content which was written by the sender.

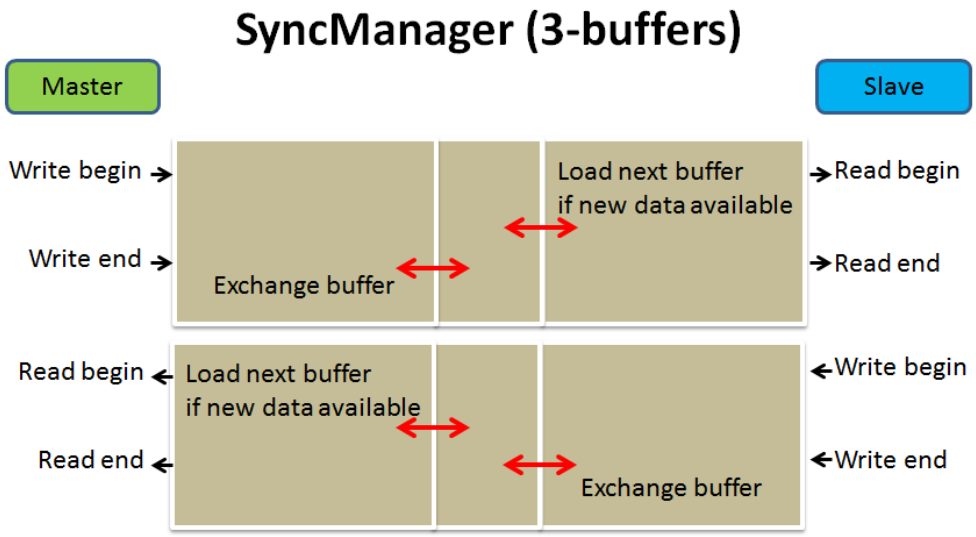


Figure 1.11 SyncManager 3-Buffer-Mode

1.2.5 Completed set of I/O modules for total solutions

The ADAM-5000/ECAT Series uses a convenient backplane system. Advantech's complete line of ADAM-5000 EtherCAT series modules integrates with the ADAM-5000/ECAT to support your applications. Full ranges of digital module supports 10 to 30 VDC input and outputs. A set of analog modules provide 12-bit resolution and programmable input and output (including bipolar) signal ranges. For details, refer to **Appendix IO Modules.**

Type		New	Description
Main Unit		**ADAM-5000/ECAT**	EtherCAT IO Coupler
Digital	**Input**	**ADAM-E5051S/53S**	16/32-ch Isolated Digital Input Module
	Output	**ADAM-E5056S/SO**	16-ch Sink/Source Type Isolated Digital Output Module
	Relay	**ADAM-E5057S /69**	32/8-ch Relay Type Isolated Digital Output Module
	Counter	**ADAM-E5082**	2-ch High Speed Counter/Frequency Module
Analog	**Input**	**ADAM-E5017UH**	8-ch Ultra High Speed Analog Input Module
	Output	**ADAM-E5024H**	4-ch High Speed Analog Output Module

Table 1.1 ADAM-5000/ECAT I/O modules support list

ADAM-5000/ECAT is designed with a high I/O capacity and supports all types of digital and analog I/O modules. Providing four slots for any mixed modules, this slave IO system handles up to 128 I/O points. ADAM-5000/ECAT supports

ADAM-5000/ECAT not only has a higher I/O capacity, but it also has a smarter

diagnostic ability. There are six indicators on the front case of the CPU module. Users can read the system status clearly, including power, CPU, Ethernet link, Communication active, communication rate, and more.

1.2.6 Software Support

Based on the EtherCAT standard, Advantech provides the necessary DLL drivers and Windows Utility to configure ADAM-5000/ECAT including IO modules which are integrated on it in Window XP/7/8. Advantech also provides Common Motion IO API for users to develop their own applications under “Common Motion Architecture” to unify user interfaces of all Advantech motion devices.

Even more, users also can do parameter settings via 3-rd party EtherCAT master, such as TwinCAT or Acontis, by ADAM-5000/ECAT ESI file to connect in customer’ existing EtherCAT network.

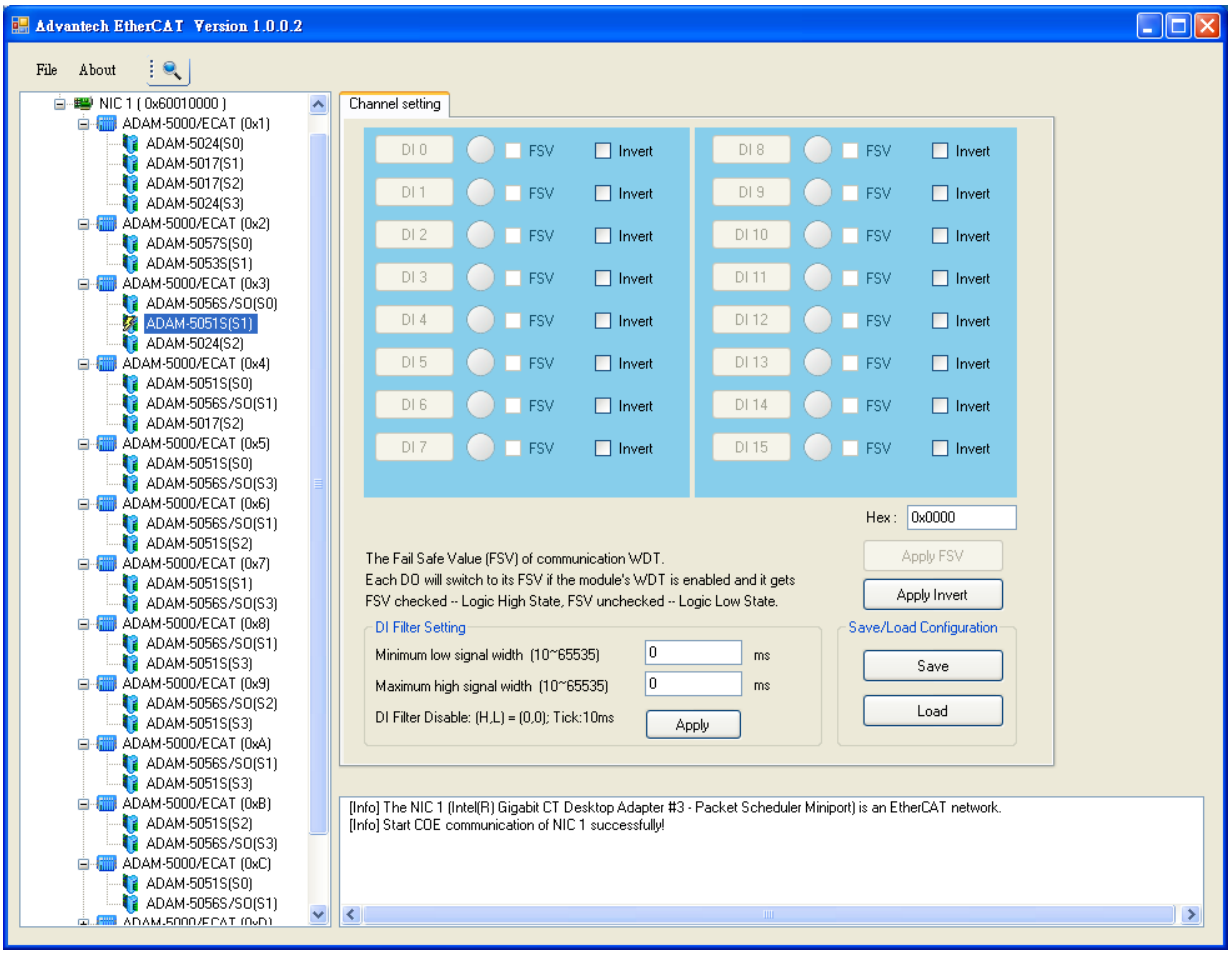


Figure 1.12 EtherCAT Windows Utility

1.3 Technical Specification

1.3.1 System Specification

Item	Description
Number of IO modules	4
Communications protocol	EtherCAT
Physical layer	100BASE-TX (IEEE 802.3)
Baud rate	100Mbps
Memory	Flash ROM: 64M SPI RAM: 4G DDR3
Send/receive PDO data sizes (Max. number of bytes fieldbus)	Input: 1,024 bytes max. Output: 1,024 bytes max.
Mailbox data size	Input: 128 bytes Output: 128 bytes
Mailbox	Emergency messages, SDO requests, and SDO information
Node address setting range	Three rotate switches for up to 4,096 slave IDs (x1, x10, x100)
Communications cycle in DC Mode	250 min.
Input voltage	10~30V_{DC}
Power consumption	2.5 W @ 24 VDC (not including I/O modules)

Note!

This depends on the specifications of the EtherCAT master and Unit configuration.

1.3.2 General Specification

Item	Description
Connectors	1 x Screw-terminal for RS-485 (communication) 1 x DB9-M for RS-232 (internal use) 1 x Screw-terminal for power input 2 x RJ-45 for LAN
Dimensions (W x H x D)	231 x 110 x 75 mm
Mounting	DIN-rail, wall
Atmosphere	No corrosive gases
Operating Temperature	- 10 ~ 70°C (14 ~ 158°F)
Storage Temperature	- 25 ~ 85°C (-13 ~ 185°F)

Operating Humidity	5 ~ 95%, non-condensing
Protection	I/O Module Isolation : 3,000 V_{DC} LAN Communication: 1,500 V_{DC} Over-voltage and power reversal
Certification	CE, FCC class A

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.

1.4 ADAM-5000/ECAT Main Unit

1.4.1 Dimensions

The following diagrams show the dimensions of the system unit and an I/O unit. All dimensions are in millimeters.

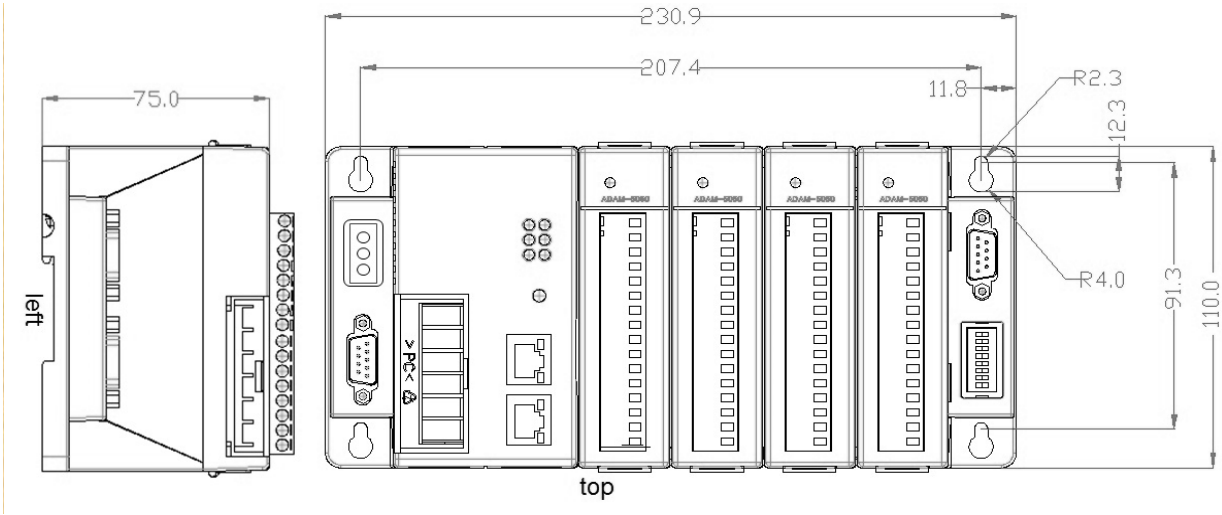


Figure 1.13 ADAM-5000/ECAT system and I/O module dimensions

There are six LEDs on the ADAM-5000/ECAT front panel. The LEDs indicate ADAM-5000/ECAT's system status, as explained below:

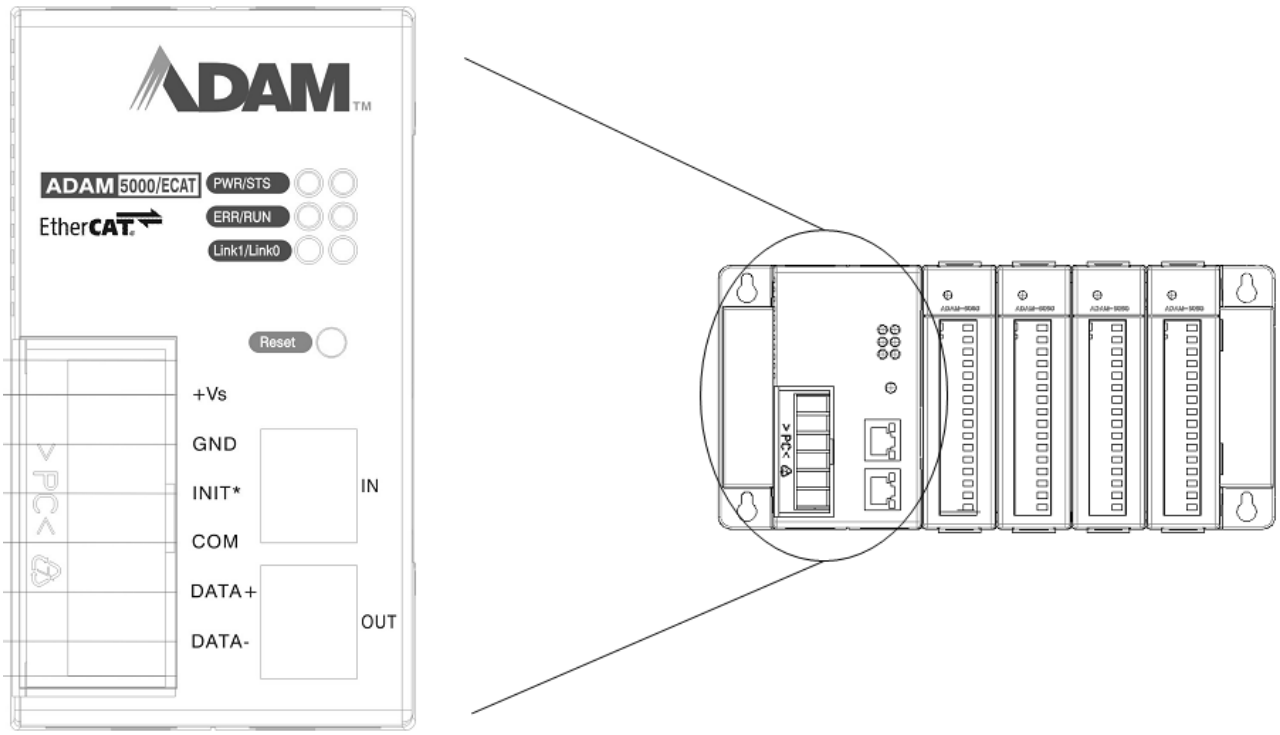


Figure 1.14 LED indicators in ADAM-5000/ECAT system

1.4.2 LED Indicator

The LED indicator status follows EtherCAT specification

1.4.2.1 LED States

Power LED (Green) on front panel

Indicator states	System State	Description
Off	Power off	The system is not on power on
Off	System in Reset	The system is in reset stage or the reset button is pushed
On	Power on	The system is on power on

EtherCAT Link/Activity LED (Green)

Indicator states	Link	Activity	Condition
On	Yes	No	Port open/connected
Flickering	Yes	Yes	Port open/connected
Off	No	No applicable	Port closed/disconnected

EtherCAT Run LED (Green)

Indicator states	Slave state	Condition
Off	INITIALISATION	The device is in state INIT
Blinking	PRE-OPERATIONAL	The device is in state PRE-OPERATIONAL
Single Flash	SAFE-OPERATIONAL	The device is in state SAFE-OPERATIONAL
On	OPERATIONAL	The device is in state OPERATIONAL
Flickering	BOOTSTRAP	The device is booting and has not yet entered the INIT state, or: The device is in state BOOTSTRAP. Firmware download operation in progress

EtherCAT Err LED (Red)

Indicator states	Error Name	Description	Example
Off	No error	The device is in working condition	
Blinking	Invalid Configuration	General configuration error	State change commanded by master is impossible due to register or object setting, or invalid hardware configuration (pin sharing violation detected by ESC)
Single Flash	Local error	Slave device application has changed the EtherCAT state autonomously, due to local error. Error Indicator bit is set to 1 in AL Status register.	Device changes its EtherCAT state from OpMode to SafeOpError due to a synchronization error.
Double Flash	Process Data Watchdog Timeout/ EtherCAT Watchdog Timeout	An application watchdog timeout has occurred.	Sync Manger Watchdog timeout.

System/STS LED (white)

Indicator states	Indicator color	Description
Off	White	The device is not power on.
Blinking	Green	The device is in working condition
Blinking	Orange	The device is in abnormal condition. For instance:

		IO module doesn't work normal. IO module mismatch with the configure setting. For details, refer to "Emergency Messages Codes" section.

1.4.2.2 LED Timing Diagram

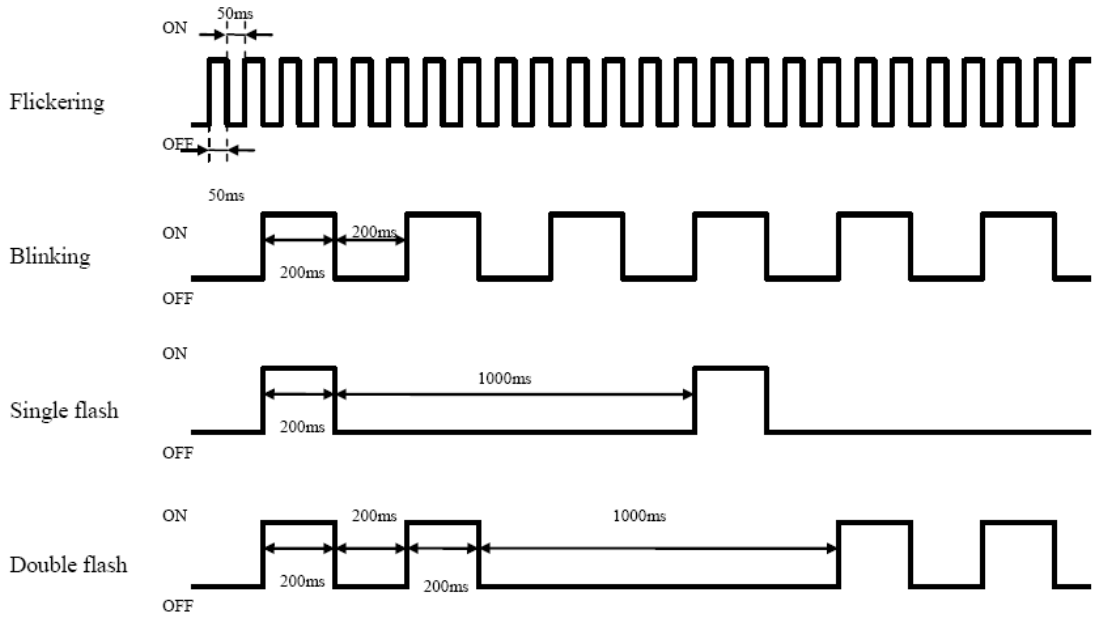
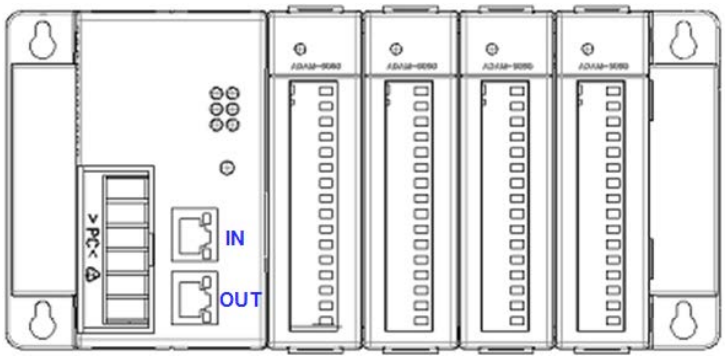


Figure 1.15 LED indicators flash rates

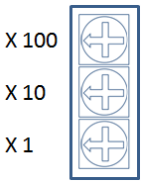
1.4.3 LAN Connector

Two RJ45 connectors can be set as EtherCAT port. 
One connector is EtherCAT network " IN", the other one is EtherCAT network "OUT"



1.4.4 Rotate Switch

ADAM-5000/ECAT uses three hexadecimal rotate switches to represent the slave ID. (Range:000~FFF)



For example, if user arranges the rotate switches following the sequence: 0, 4, F, the slave ID will be set: 0 x 256 + 4 x 16 + F x 1= 80.

Note!
Slave ID=000 is not allowed, the default slave ID is 001.

Note!
The other three connectors are for internal use. (RS-232 x 2, DIP Switch)

1.4.5 Basic Function Diagram

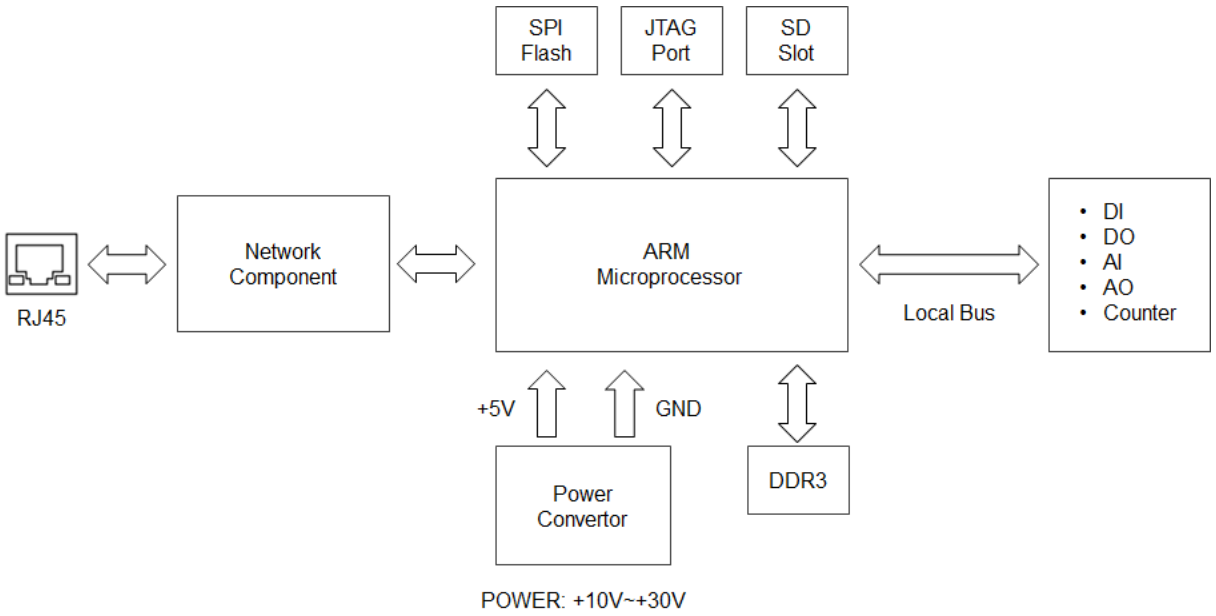


Figure 1.16 Function block diagram

Chapter 2

Selecting Your Hardware Components

2.1 Selecting I/O Module

To organize an ADAM-5000/ECAT I/O system, you need to select I/O modules to interface the main unit with field devices or processes that you have previously determined. There are several things should be considered when you select the I/O modules. What type of I/O signal is applied in your system? How much I/O is required to your system? How will you place the main unit for concentrate the I/O points of an entire process.

How many ADAM-5000/ECAT main units are required for distributed I/O points arrangement.

What is the required voltage range for each I/O module? What isolation environment is required for each I/O module? What are the noise and distance limitations for each I/O module? Refer to table 2-1 I/O as module selection guidelines

Table 2.1: I/O Selection Guidelines

Type	Application	Description
Discrete input module and block I/O module	Selector switches, pushbuttons, photoelectric eyes, limit switches, circuit breakers, proximity switches, level switches, motor starter contacts, relay contacts, thumbwheel switches	Input modules sense ON/OFF or OPENED/CLOSED signals. Discrete signals can be either AC or DC.
Discrete output module and block I/O module	Alarms, control relays, fans, lights, horns, valves, motor starters, solenoids	Output module signals interface with ON/OFF or OPENED/CLOSED devices. Discrete signals can be either AC or DC.
Analog input module	Thermocouple signals, RTD signals, temperature transducers, pressure transducers, load cell transducers, humidity transducers, flow transducers, potentiometers.	Convert continuous analog signals into input values for ADAM-5000/ECAT
Analog output module	Analog valves, actuators, chart recorders, electric motor drives, analog meters	Interpret ADAM-5000/ECAT output to analog signals (generally through transducers) for field devices.

Advantech provides more than 10 types of ADAM-5000 I/O modules for various applications so far. The table 2-2 will help you to select the ADAM-5000 I/O modules quickly and easily.

Table 2.2: I/O Modules Selection Guide			
Type		**Model**	**Description**
Digital	**Input**	**ADAM-E5051S**	16-ch Isolated Digital Input Module
		ADAM-E5053S	32-ch Isolated Digital Input Module
	Output	**ADAM-E5056S**	16-ch Sink Type Isolated Digital Output Module
		ADAM-E5057S	32-ch Sink Type Isolated Digital Output Module
	Relay	**ADAM-E5069**	8-ch Power Relay Output Module
	Counter	**ADAM-E5082**	2-ch High Speed Counter/Frequency Mod-ule
Analog	**Input**	**ADAM-E5017**	8-ch Analog Input Module
		ADAM-E5017UH	8-ch Ultra High Speed Analog Input Mod-ule
	Output	**ADAM-5024H**	4-ch Analog Output Module

2.2 Selecting Power Supply

ADAM-5000/ECAT system works under unregulated power source between +10 and +30 VDC. When you arrange different I/O modules on ADAM-5000/ECAT's back plant, it may require comparable power supply. Use the following steps as guidelines for selecting a power supply for your ADAM-5000/ECAT system.

- Refer to table 2.3 to check the power consumption of ADAM-5000/ECAT main unit and each I/O module.

2.2.1 Power Consumption of ADAM-5000/ECAT and IO Modules

Table 2.3: Power Consumption of ADAM-5000/ECAT and IO Modules		
Main Unit	**Description**	**Power Consumption**
ADAM-5000/ECAT	4-slot Distributed High Speed I/O System for EtherCAT	2.5 W
DI/O module	**Description**	**Power Consumption**
ADAM-E5051S	16-ch Isolated Digital Input w/ LED Module	0.53 W
ADAM-E5053S	32-ch Isolated Digital Input Module.	1 W
ADAM-E5056S	16-ch Sink Type Isolated Digital Output w/ LED Module	0.8 W
ADAM-E5057S	32-ch Isolated Digital Output Module.	1 W
ADAM-E5069	8-ch Power Relay Output w/ LED Module	2.2 W
AI/O module	**Description**	**Power Consumption**
ADAM-E5017/17UH	8-ch High speed Analog Input Module	2.2 W
ADAM-E5024	4-ch Analog Output Module	2.9 W

- Calculate the Summary of the whole system's power consumption.
  For example, there are following items in your system.
  ADAM-5000/ECAT * 3 & ADAM-E5053S* 3 & ADAM-E5017 * 2 & ADAM-E5056S * 4 & ADAM-E5057S * 1 & ADAM-E5069 * 2
  The power consumption is:
  2.5W * 3 + 1W * 3 + 2.2W * 2 + 0.8W * 4 + 1W * 1 + 2.2W * 2 = 23.5W

Select a suitable power supply from Table2.4 or other comparable power resource for system operation.

2.2.2 Power Supply Specification Table

Table 2.4: Table 2.4: Power Supply Specification Table

Specification		PWR-242	PWR-243	PWR-244
Input	Input Voltage	90~264 VAC	85~132 VAC 170~264 VAC	100~240 VAC
	Input Frequency	47~63 Hz		
	Input Current	1.2 A max.	1.4 A max.	25 A/110 VAC 50 A/220 VAC (Inrush current)
	Short Protection	Yes		
Output	Output Voltage	+24 VDC		
	Output Current	2.1 A	3 A	4.2 A
	Overload Protection	Yes		
General	Dimension	181 mm x 113 mm x60 mm (L x W x H)		
	Operating Temperature	0~50º C (32~122º F)		
	DIN-rail Mountable	Yes	No	No

2.3 Selecting Operator Interface

2.3.1 Common Motion Utility

To complete the ADAM-5000/ECAT EtherCAT Slave I/O System connection with the EtherCAT master, producing the EtherCAT Network Information file (ENI) is necessary. The ENI is created based on EtherCAT Slave Information files (ESI) which are provided by ADAM-5000/ECAT.

There are two ways to producing ENI file:

(1) Advantech Common Motion Utility

(2) Other EtherCAT master software, such as TwinCAT or Acontis

Chapter 3

Hardware Installation Guide

3.1 Determining the proper environment

Before you start to install the ADAM-5000/ECAT system, there are something needed to check.

3.1.1 Check the content of shipping box

Unpack the shipping boxes and make sure that the contents include:

- ADAM-5000/ECAT main unit with two blank slot covers

3.1.2 System Requirement

- Host computer
  - Computer with Atom or Core i CPU
  - Windows XP/7 (32 or 64 bit) or higher versions
  - At least 32 MB RAM
  - 20 MB of hard disk space available
  - VGA color monitor
  - Mouse or other pointing devices
- 100 Mbps Ethernet Port (at least 2 ports)
- Two Ethernet Cable with RJ-45 connector
- Power supply

3.1.3 I/O modules

At least one I/O module is needed to use the system. Prepare the required I/O modules as the interface for a variety of field singles.

3.2 Installing your main unit and module

When inserting modules into the system, align the PC board of the module with the grooves on the top and bottom of the system. Push the module straight into the system until it is firmly seated in the back plane connector (see figure 3-1). Once the module is inserted into the system, push in the retaining clips located at the top and bottom of the module to firmly secure the module to the system (see figure 3-2).

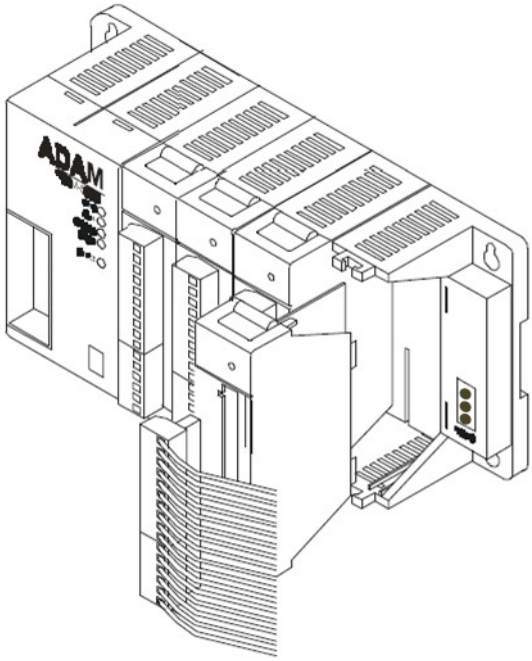


Figure 3.1 Module alignment and installation

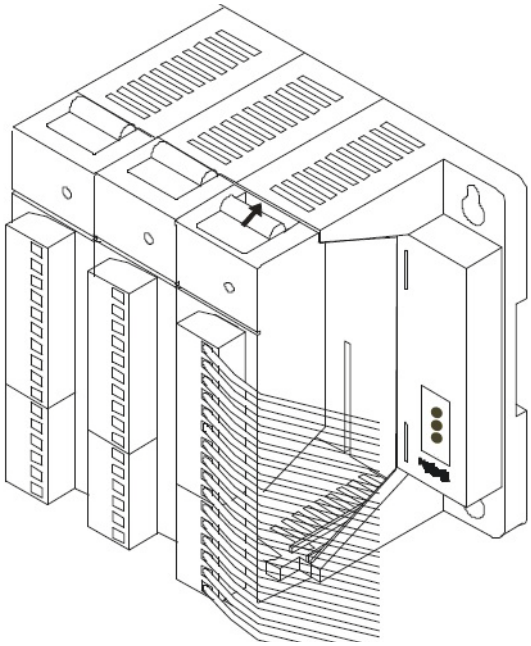


Figure 3.2 Secure the module to the system

3.3 Mounting

The ADAM-5000/ECAT system can be installed on a panel or on a DIN rail.

3.3.1 Panel mounting

Mount the system on the panel horizontally to provide proper ventilation. You cannot mount the system vertically, upside down or on a flat horizontal surface. A standard #7 tatting screw (4 mm diameter) should be used.

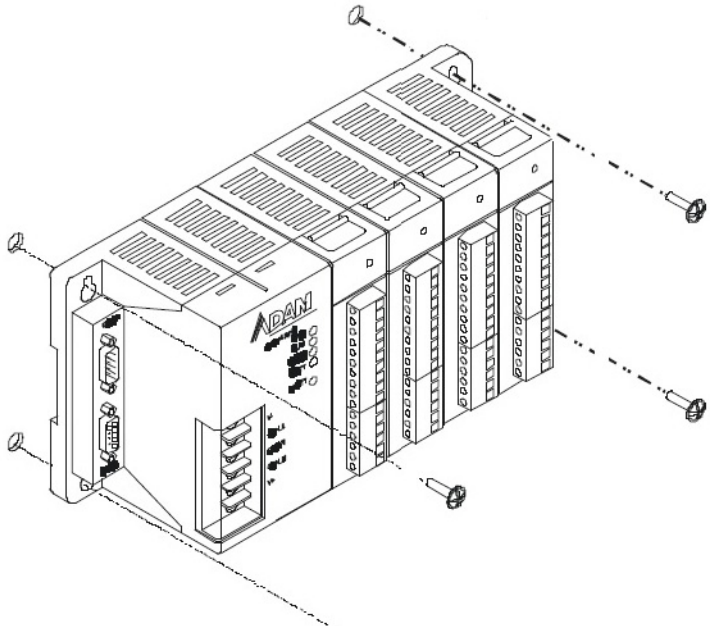


Figure 3.3 ADAM-5000/ECAT panel mounting screw placement

3.3.2 DIN rail mounting

The system can also be secured to the cabinet by using mounting rails (see figure 3-4). If you mount the system on a rail, you should also consider using end brackets at each end of the rail. The end brackets help keep the system from sliding horizontally along the rail. This minimizes the possibility of accidentally pulling the wiring loose. If you examine the bottom of the system, you will notice two small retaining clips. To secure the system to a DIN rail, place the system on to the rail and gently push up on the retaining clips (see figure 3-5). The clips lock the system on the rail. To remove the system, pull down on the retaining clips, lift up on the base slightly, and pull it away from the rail.

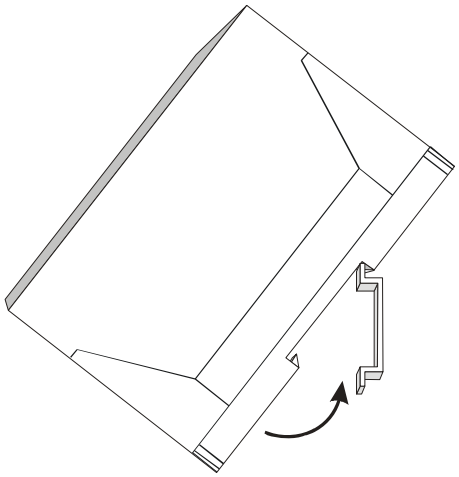


Figure 3.4 ADAM-5000/ECAT DIN rail mounting

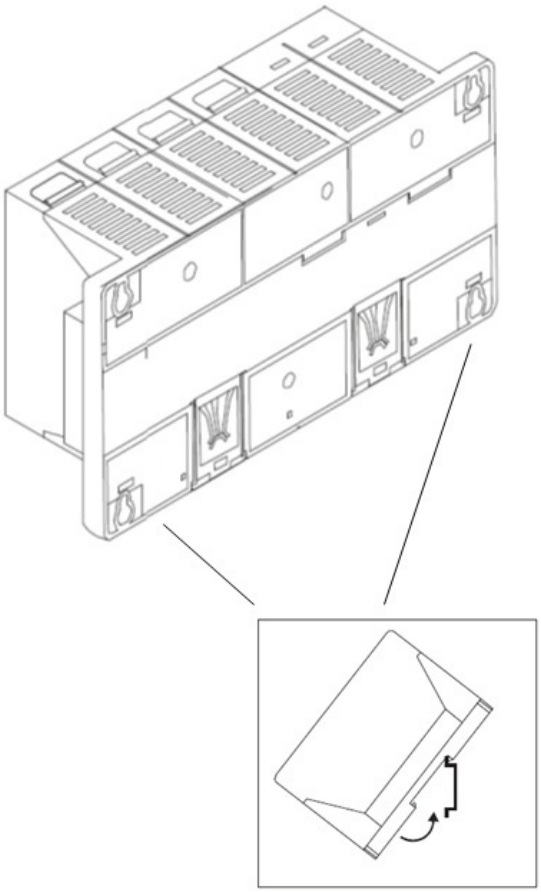


Figure 3.5 Secure ADAM-5000/ECAT System to a DIN rail

3.4 Wiring and Connections

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

3.4.1 Power supply wiring

Although the ADAM-5000/ECAT 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.

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

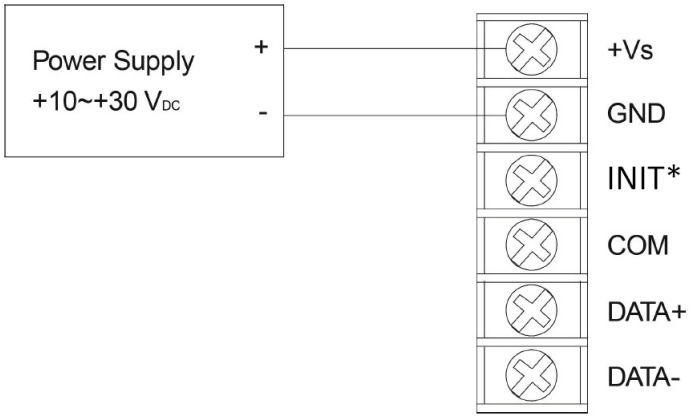


Figure 3.6 ADAM-5000/ECAT power wiring

3.4.2 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.

1. The terminal block accepts wires from 0.5 mm to 2.5 mm.
2. Always use a continuous length of wire. Do not combine wires to make them longer.
3. Use the shortest possible wire length.
4. Use wire trays for routing where possible.
5. Avoid running wires near high-energy wiring.
6. Avoid running input wiring in close proximity to output wiring where possible.
7. Avoid creating sharp bends in the wires.

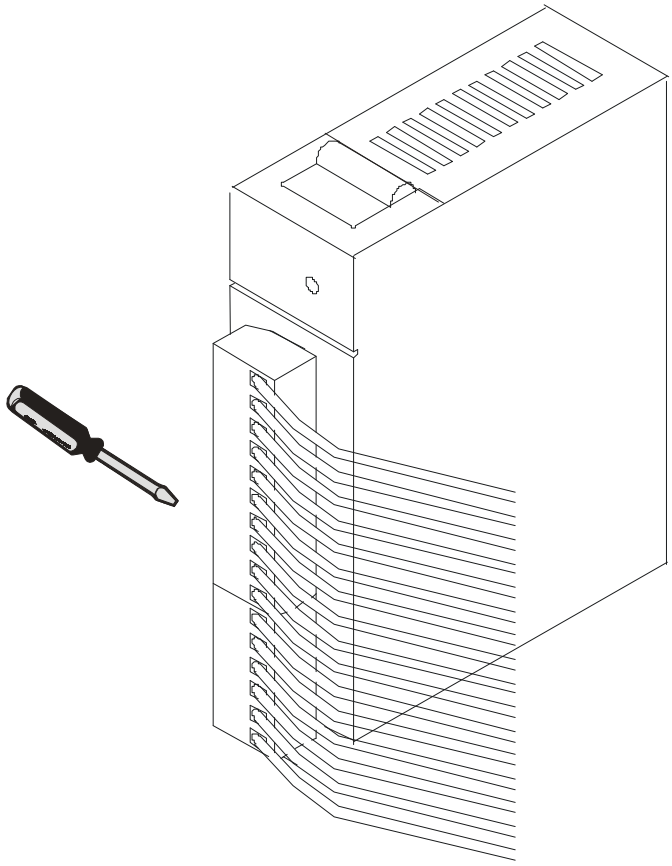


Figure 3.7 ADAM-5000 I/O Module Terminal Block wiring

Chapter 4

Object Dictionary

4.1 Object Dictionary

In industrial automation, use of device profiles is a very common method to describe the functionality and parameters of the devices. EtherCAT allows the use of multiple protocols for communications. For acyclic data exchange, EtherCAT provides mailbox communication protocols (CoE, SoE, EoE, FoE, AoE). Cyclic data is exchanged in Process Data Objects (PDOs) with fixed or configurable PDO sizes.

ADAM-5000/ECAT Slave Terminal uses the CAN application protocol over EtherCAT (CoE) as the device profile. This is the most commonly used EtherCAT communication protocol for acyclic data access. It is also provides mechanisms to configure PDOs for cyclic data exchange.
ADAM-5000/ECAT also uses File Access over EtherCAT (FoE) to update firmware.
In EtherCAT, several protocols are available for the application layer to implement the required specification of the product development. Data is exchanged cyclically or acyclically and data sizes can be fixed or configurable.

There are two methods for reading and writing data to objects: **Process Data Object (PDO)** and **Service Data Object (SDO)**. PDOs are used for cyclic, time critical data exchange, while SDOs are used for non-time-critical communication and configuration. PDOs are the primary method for communication during run-time but they must be configured prior to being used. PDO transmission occurs only while the EtherCAT state machine is in the OP mode (and limited transactions during safe operational state). SDO messages may be used any time after initialization.

4.1.1 Structure of Object Dictionary

Several device profiles can be applied for EtherCAT devices by using CoE. ADAM-5000/ECAT Slave Terminal uses **Modular Device Profile (MDP)** which a standardized structure for the **Object Dictionary (OD)** provided by CoE is defined. It is a helpful configuration mechanism for EtherCAT slave structure like ADAM-5000/ECAT, usable for a large number of devices from very simple one to complex sub-structured. The modular device profile can be applied for different device types. Depending on the device type the object dictionary can be static (defined in ROM) or dynamic (defined in RAM). The dynamic object dictionary can be configured by the EtherCAT master during PRE-OP by writing the expected module configuration or can be generated automatically after power-on.

The structure of the object dictionary is divided into several function specific areas.

Index	Object Dictionary Area	Description / Value
EtherCAT Communication Area		
0x1000-0x1FFF	Communication Area	Standard communication area
Object Area of the Modules		
0x2000 - 0x5FFF	Manufacturer Specific Area	
0x6000 - 0x6FFF	Input Area	Objects that can be mapped to TxPDOs
0x7000 - 0x7FFF	Output Area	Objects that can be mapped to RxPDOs
0x9000 - 0x9FFF	Information Area	Scanned information from the modules
0xA000 - 0xAFFF	Reserved Area	Diagnostic, status, statistic or other information
0xB000 - 0xBFFF	Reserved Area	Service objects
0xC000 - 0xEFFF	Reserved Area	
Object Area of the Device		
0xF000 - 0xFFFF	Device Area	Parameters belonging to the device

Table 4.1 Object Dictionary Structure

Each module can occupy several objects in the function specific areas. The standard defines 16 objects per module in a specific area, but this number may be adapted to the device requirements. In the following it will be assumed that there are 16 objects per module defined and up to 255 modules available.

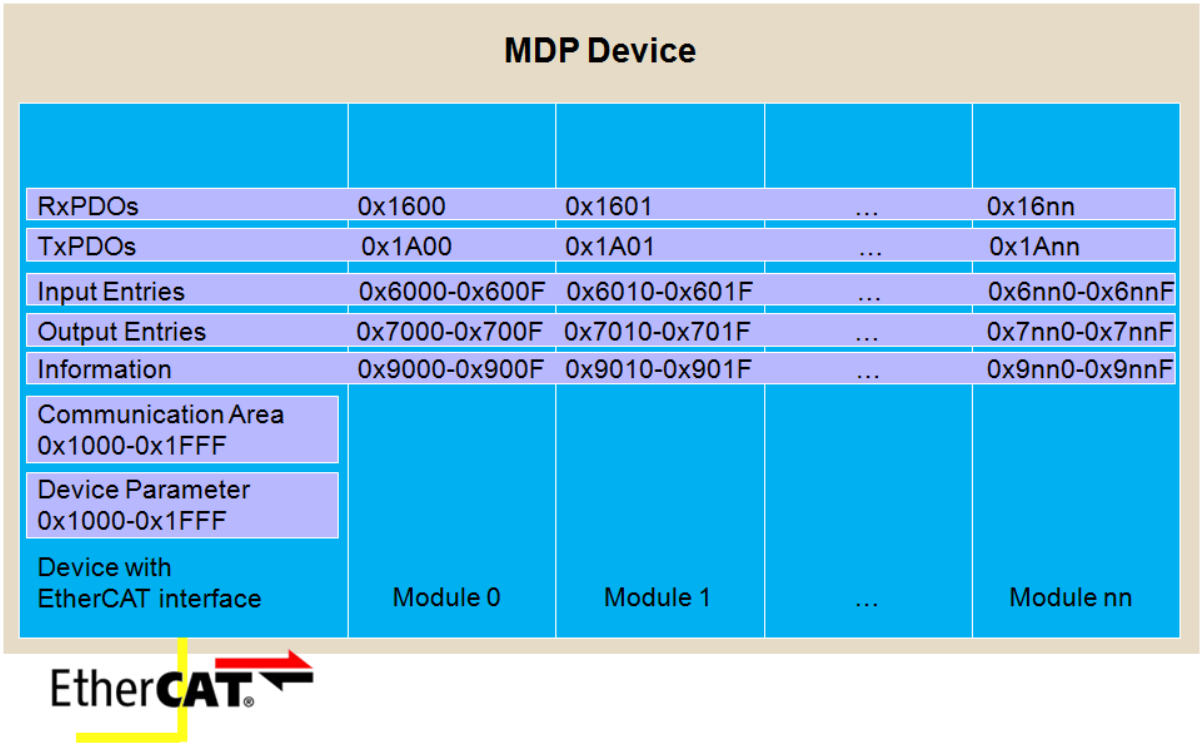


Figure 4.1 Model of a Modular Device

4.1.1.1 Process Data Objects (PDOs)

In EtherCAT network, process data is divided into segments with a maximum of 8 bytes. These segments are known as **process data objects (PDOs)**, which are used to cyclically transfer data. PDOs are used in CoE for broadcasting high-priority control- and status information. Receive PDOs (RxPDOs) and transmit PDOs (TxPDOs) are distinguished: an input/output module sends its input data with TxPDOs and receives its output data in the RxPDOs. EtherCAT Slaves support PDO mapping for I/O control.

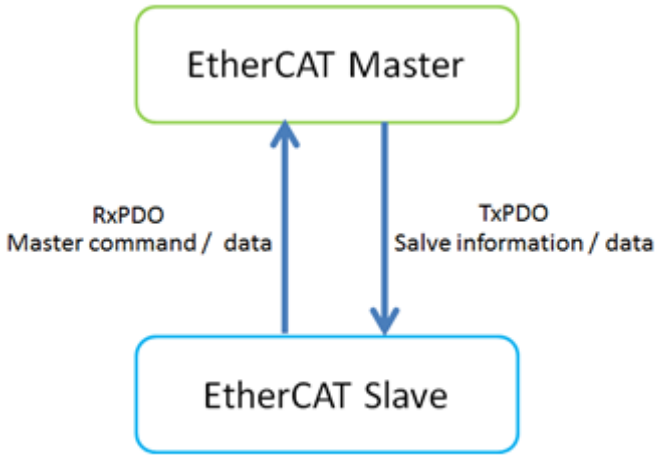


Figure 4.2 PDOs mechanism

4.1.1.2 Service Data Objects (SDOs)

EtherCAT Slave Terminals support SDO communications.

The EtherCAT master can read and write data from and to entries in the object dictionary with SDO communications to make parameter settings and monitor status.

4.1.2 Object Dictionary of ADAM-E5000 I/O modules

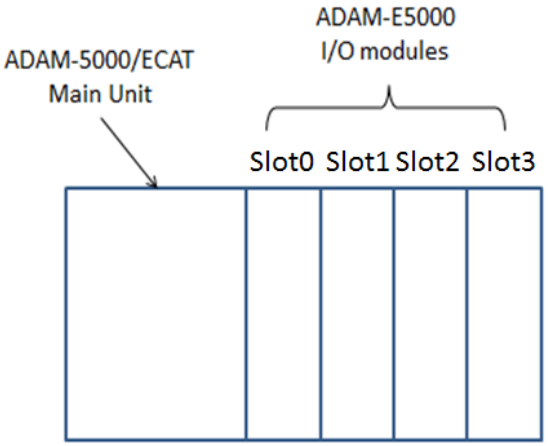
This section describes how objects are assigned in an EtherCAT Slave Terminal with the following format.

Index (hex)	Sub-Index (hex)	Name	Range	Data Type	Access	SDO	PDO	State
(Index)	(SubIndex)	(Object Name)	(Data range)	(Type)	(Access)	--	--	(State)

Table 4.2 Format of Objects

Each item has the following meaning.

Item	Description
Index	This is the index of the object that is expressed as a four-digit hexadecimal number.
Sub-Index	This is the subindex of the object that is expressed as a two-digit hexadecimal number.
Name	This is the name of the object.
Range	For a read-only (RO) object, this is the range of the data that you can read. For a read/write (RW) object, this is the setting range of the data.
Data Type	UINT, USINT
Access	This data tells if the object is read-only, write-only or read/write. RO: Read only ; WO: Write only ; RW: Read and Write
SDO	-
PDO	Tx or Rx
State	This objects can be operate when the EtherCAT state machine of this device is in the specific state. Pre-Op : Pre-Operational ; Op : Operational state



4.1.2.1 Assigning Input /Output Modules

Get Current Value by PDO

The x of index is a number which represents the slot number the ADAM-E5017 plugs in. For 4-slot ADAM-5000/ECAT, the first (leftmost) slot number is 00h and the last (rightmost) slot number is 03h. For example, if you need to get the value of ADAM-E5017 AI module which plugs in the second slot of ADAM-5000/ECAT, the x will be set to 1.

ADAM-E5017 (8-ch Analog Input Module)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
60x0h	01h – 08h	AI 1 – AI 8	0-65535	UINT	RO	--	Tx	Op

ADAM-E5017UH (8-ch Ultra High Speed Analog Input Module)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
60x0h	01h – 08h	AI 1 – AI 8	0-4095	UINT	RO	--	Tx	Op

ADAM-E5024 (4-ch Analog Output Module)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
70x0h	01h – 04h	AO 1 – AO 4	0-4095	UINT	WO	--	Rx	Op

ADAM-E5051S (16-ch Isolated Digital Input Module with LED)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
60x0h	01h – 02h	DI 1 – DI 2	0-255	USINT	RO	--	Tx	Op

ADAM-E5053S (32-ch Isolated Digital Input Module)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
60x0h	01h – 04h	DI 1 – DI 4	0-255	USINT	RO	--	Tx	Op

ADAM-E5056S (16-ch Isolated Digital Output Module with LED)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
70x0h	01h – 02h	DO 1 – DO 2	0-255	USINT	WO	--	Rx	Op

ADAM-E5057S (16-ch Isolated Digital Output Module with LED)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
70x0h	01h – 04h	DO 1 – DO 4	0-255	USINT	WO	--	Rx	Op

ADAM-E5069 (8-ch Relay Output Module)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
70x0h	01h	DO1	0-255	USINT	WO	--	Rx	Op

Note! More detail information of **Object Dictionary** can be referred in Chapter 5.

Setting by SDO

Select the slot of module which you want to diagnosis or change the setting.

For 4-slot ADAM-5000/ECAT, the first (leftmost) slot number is 00h and the last (rightmost) slot number is 03h.

For example, if you need to modify the setting of ADAM-E5017 AI module which plug in the second slot of ADAM-5000/ECAT, the Module No (Index: 217Fh:01h) will be set to 01h.

Get Current Value by SDO

ADAM-E5017 (8-ch Analog Input Module)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
217Fh	01h	Module No	0-3	USINT	RW	V	--	Pre-Op Op
2180h- 2187h	01h	Get Current Value	0-65535	USINT	RO	V	--	Pre-Op Op
2180h- 2187h	02h	Channel Status	--	USINT	RO	V	--	Pre-Op
2180h	03h	Channel Masking Status	0-255	USINT	RW	V	--	Pre-Op
2180h	04h	Input Range	--	USINT	RW	V	--	Pre-Op
2180h	05h	Integration Time	--	USINT	RW	V	--	Pre-Op
2180h	06h – 07h	Calibration	--	USINT	RW	V	--	Pre-Op

ADAM-E5017UH (8-ch Ultra High Speed Analog Input Module)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
217Fh	01h	Module No	0-7	USINT	RW	V	--	Pre-Op Op
2180h-2187h	01h	Get Current Value	0-4095	USINT	RO	V	--	Pre-Op Op
2180h-2187h	02h	Channel Status	--	USINT	RO	V	--	Pre-Op
2180h	03h	Channel Masking Status	0-255	USINT	RW	V	--	Pre-Op
2180h	04h	Input Range	--	USINT	RW	V	--	Pre-Op
2180h	05h	Integration Time	--	USINT	RW	V	--	Pre-Op
2180h	06h – 07h	Calibration	--	USINT	RW	V	--	Pre-Op

ADAM-E5024 (4-ch Analog Output Module)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
217Fh	01h	Module No	0-3	USINT	RW	V	--	Pre-Op Op
2180h-2183h	01h	Get Current Value	0-4095	USINT	RO	V	--	Pre-Op Op
2180h-2183h	02h	Output Range	30-32	USINT	RW	V	--	Pre-Op
2180h-2183h	03h – 05h	Calibration	--	USINT	RW	V	--	Pre-Op

ADAM-E5051S (16-ch Isolated Digital Input Module with LED)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
217Fh	01h	Module No	0-3	USINT	RW	V	--	Pre-Op
								Op
2180h-2181h	01h	Get Current Value	0-255	USINT	RO	V	--	Pre-Op Op
2180h-2181h	02h	Invert	0-255	USINT	RW	V	--	Pre-Op Op
2180h	03h	Digital Filter Scale (Low)	0-65535	USINT	RW	V	--	Pre-Op Op
2180h	04h	Digital Filter Scale (High)	0-65535	USINT	RW	V	--	Pre-Op Op

ADAM-E5053S (32-ch Isolated Digital Input Module)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
217Fh	01h	Module No	0-3	USINT	RW	V	--	Pre-Op Op
2180h-2183h	01h	Get Current Value	0-255	USINT	RO	V	--	Pre-Op Op
2180h-2183h	02h	Invert	0-255	USINT	RW	V	--	Pre-Op Op
2180h	03h	Digital Filter Scale (Low)	--	USINT	RW	V	--	Pre-Op Op
2180h	04h	Digital Filter Scale (High)	--	USINT	RW	V	--	Pre-Op Op

ADAM-E5056S/SO (16-ch Isolated Digital Output Module with LED)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
217Fh	01h	Module No	0-3	USINT	RW	V	--	Pre-Op Op
2180h-2181h	01h	Get Current Value	0-255	USINT	RO	V	--	Pre-Op Op
2180h-2181h	02h	FSV	0-255	USINT	RW	V	--	Pre-Op Op

ADAM-E5057S (16-ch Isolated Digital Output Module with LED)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
217Fh	01h	Module No	0-3	USINT	RW	V	--	Pre-Op Op
2180h-2183h	01h	Get Current Value	0-255	USINT	RO	V	--	Pre-Op Op
2180h-2183h	02h	FSV	0-255	USINT	RW	V	--	Pre-Op Op

ADAM-E5069 (8-ch Relay Output Module)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
217Fh	01h	Module No	0-3	USINT	RW	V	--	Pre-Op Op
2180h	01h	Get Current Value	0-255	USINT	RO	V	--	Pre-Op Op
2180h	02h	FSV	0-255	USINT	RW	V	--	Pre-Op Op

Note! More detail information of **Object Dictionary** can be referred in Chapter 5.

Chapter 5

Utility

5.1 Introduction

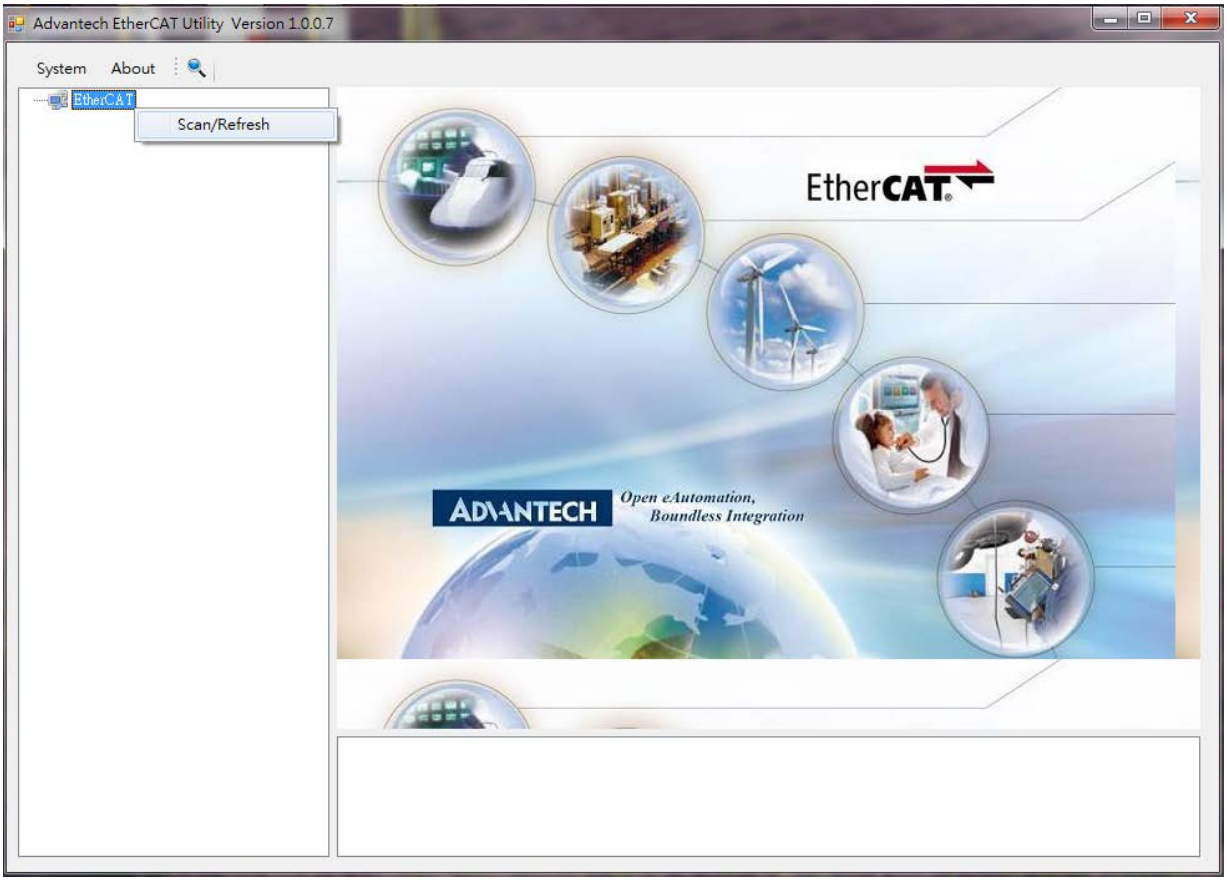
This chapter introduces how to access ADAM-5000/ECAT through Advantech EtherCAT Utility. It provides windows-based application development environment for user to configure and verify ADAM-5000/ECAT function. Easy-to-use and quick parameter setting feature help user shorten system installation and evaluation time.
In this chapter, detail introduction about accessing ADAM-5000/ECAT via TwinCAT also be provided.

5.1.1 Advantech EtherCAT Utility

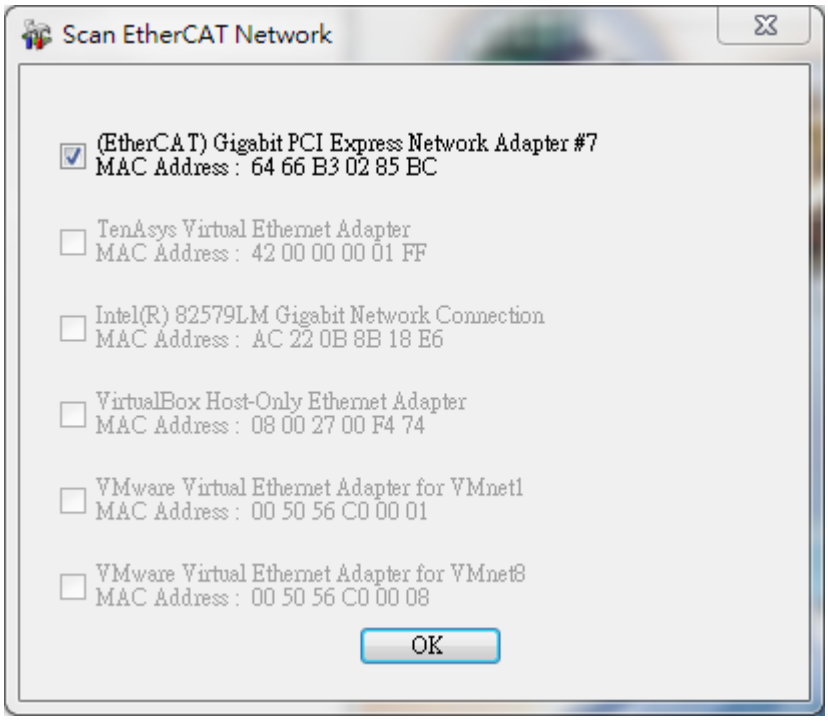
There are three files and one folder in the utility root folder. The main executive program named Advantech_EtherCAT_Utility.exe, two dll files called aecapi.dll and EtherCATClassLibrary.dll which are the kernel of this utility. The program WinPcap should be installed, which dominates the EtherCAT network packets captured from and transmitted to ADAM-5000CAT in Windows environments. Ethercat_slave_files folder contains the ESI (EtherCAT Slave Information) files. From the beginning user open the utility, the program might scan this folder and construct an information map to set the slave information.

5.1.1.1 Main Form

Right click the tree node [**EtherCAT**] or click the [**System**] button in the menu bar and click the button [**Scan/Refresh**], or just click the  in the toolbar. It will search the network interface cards in the computer and search EtherCAT slaves connected to the NIC (network interface card) individually. It might take times for searching each network interface cards.

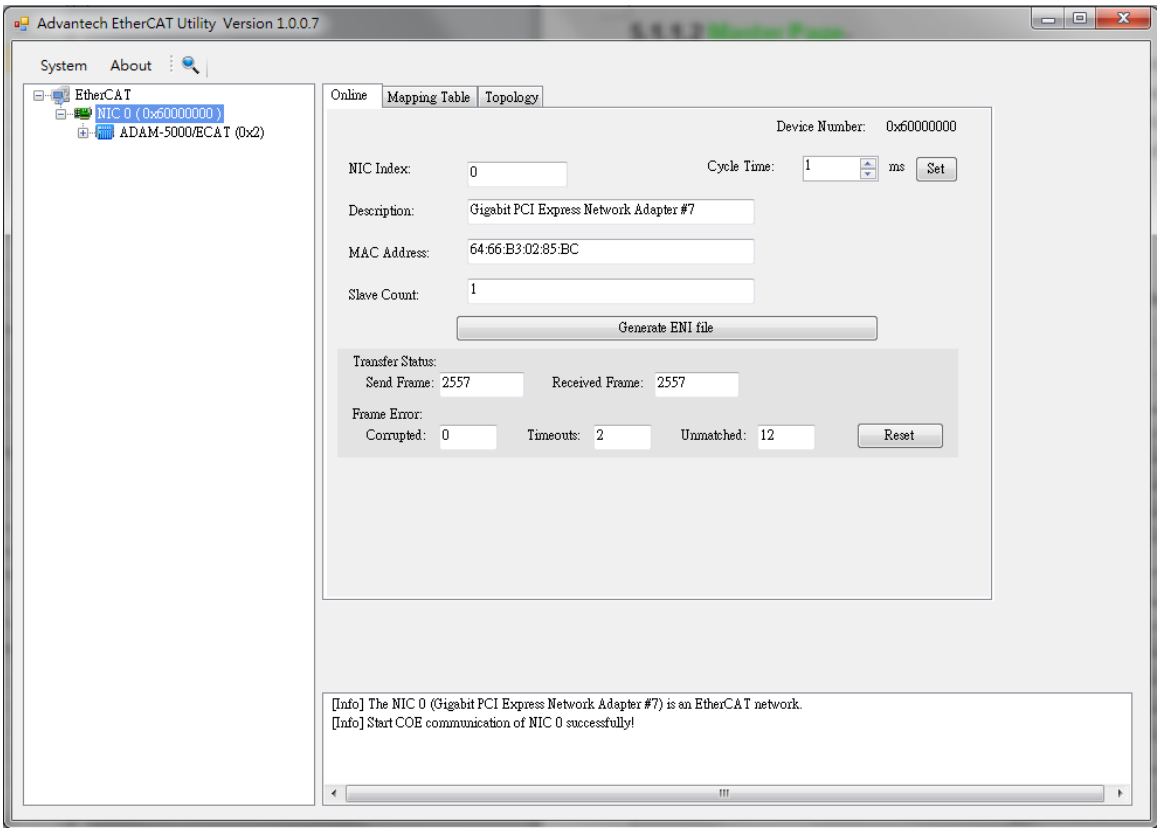


When the program is finished searching, the result will show and let user choose the NIC which user wants to use. If it is an EtherCAT network, the NIC will be labeled as “(EtherCAT)”.



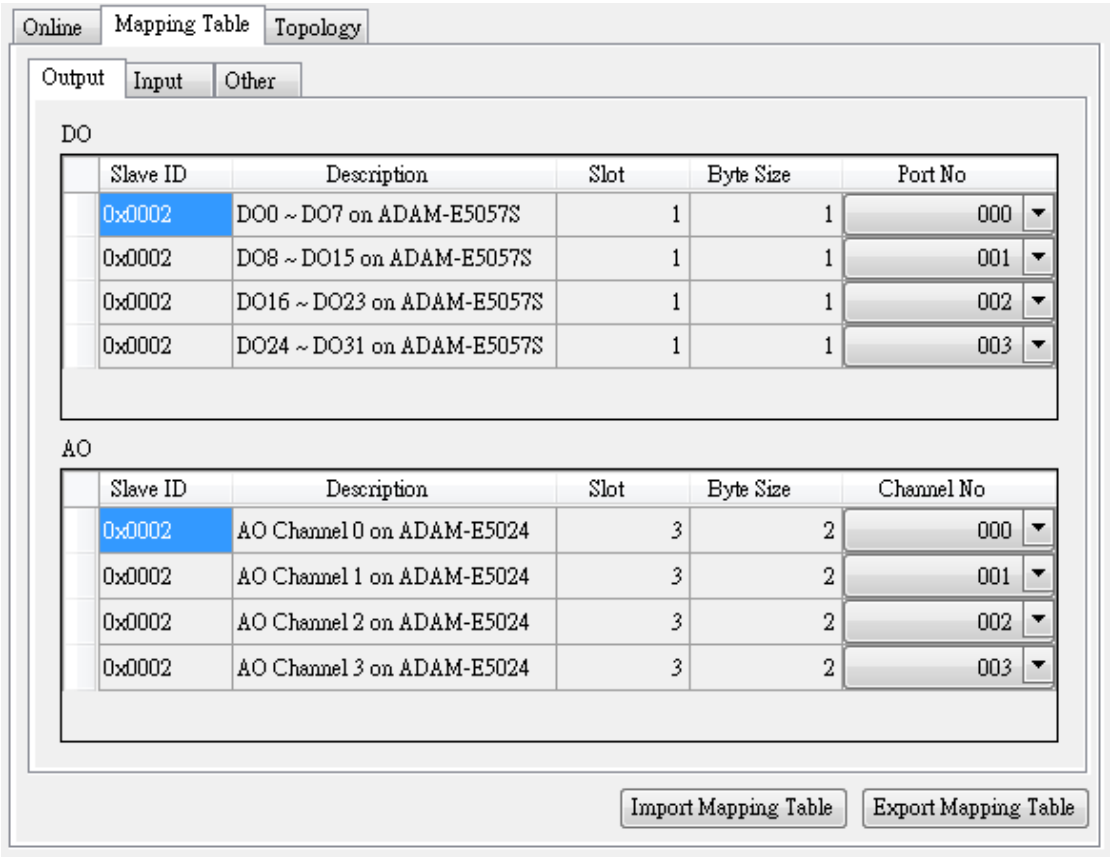
5.1.1.2 Master Page

The slaves connected to the NIC will show in the child node in the NIC node. The NIC information shows in the right panel. The slave will switch to Op mode automatically. User can change the cycle time on the upper right corner of this panel. The minimum value of cycle time is 1ms. For Common Motion API, the device number of this network interface card is also shown both on the upper right corner of this panel and the name of NIC node on the left.



5.1.1.2.1 Mapping Table

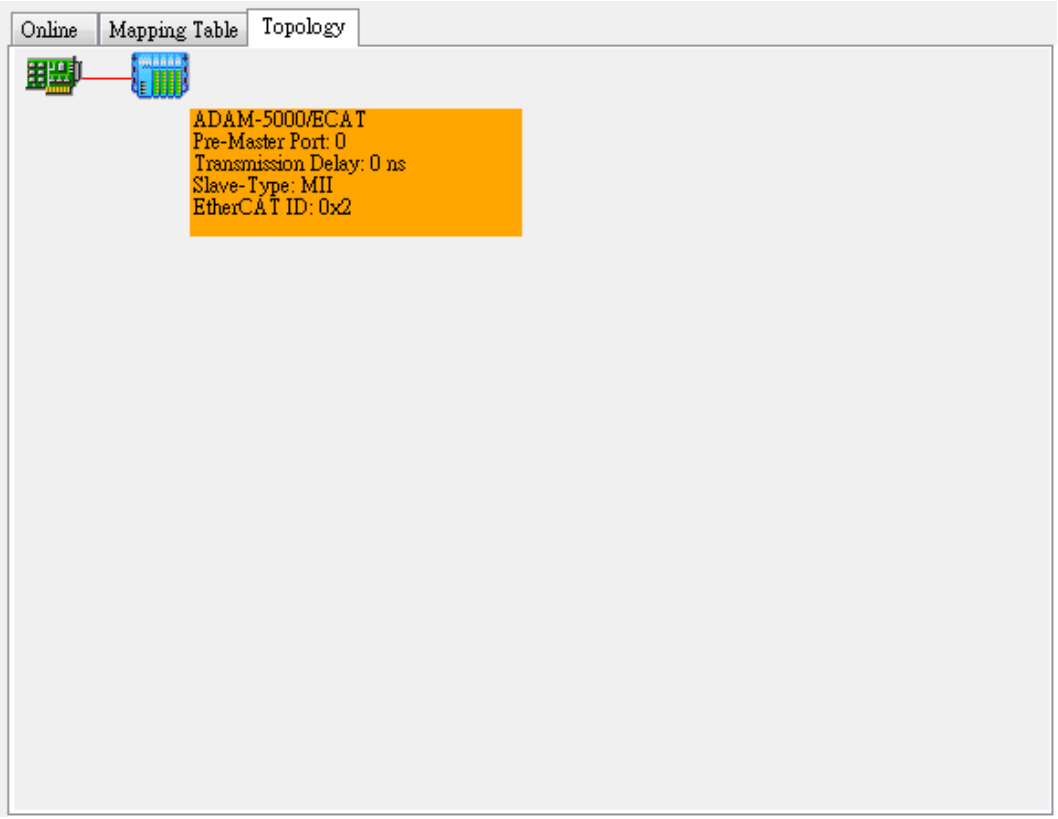
This page shows the physical and logical index mapping table of the slave modules.



For Common Motion API, user read / writes value from / to modules of slave need to specify the port / channel number. In this page, user can specify the logical index of these module ports and channels and export the setting. Then after import this setting to Common Motion API, user can operate these ports and channels by the logical index.

5.1.1.2.2 Topology

If users want to know topology information, please change to [**Topology**] table. Then you will see the topology status connected to this NIC.



It shows topology of master and slaves. The icon of master and slaves can show information when users move the cursor on it, and the page will automatically jump to the slave page which slave icon you click it. Although the redundancy is supported on the ADAM-5000/ECAT that EtherCAT can work properly if the network cable be plugged in both IN or OUT port, the topology may display abnormal due to incorrect wiring.

5.1.1.2.3 ENI File

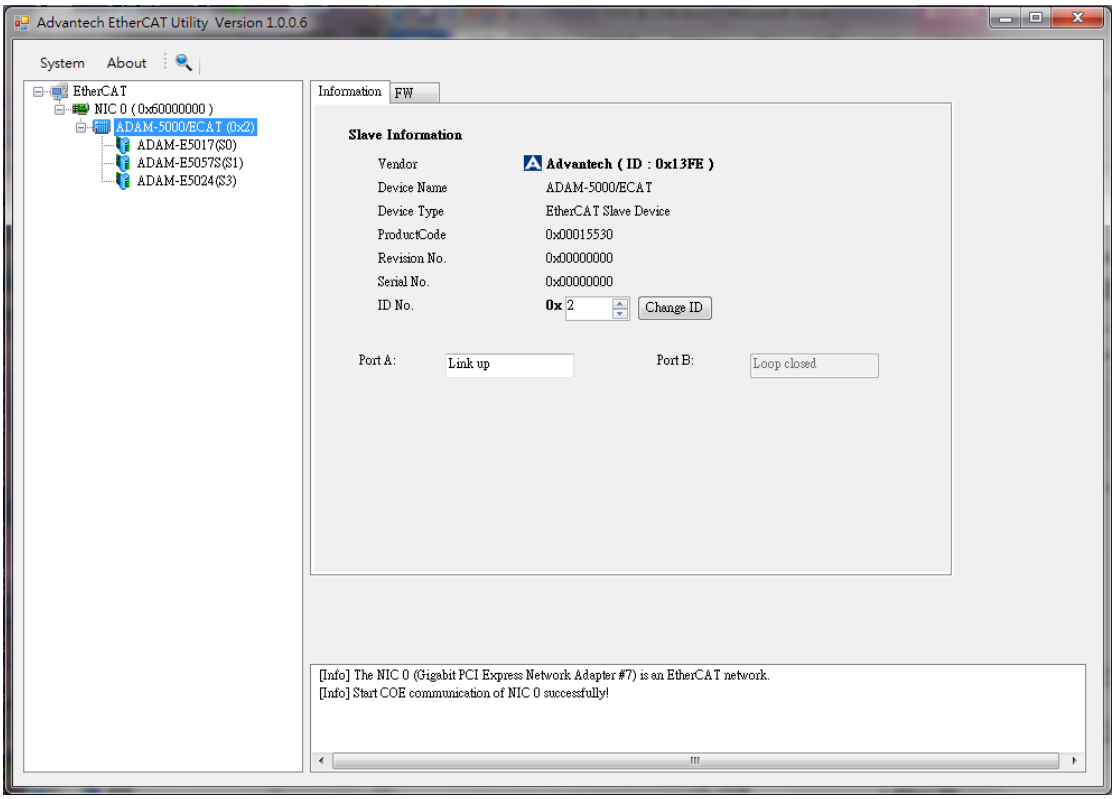
The EtherCAT Network Information (ENI) contains the necessary settings to configure an EtherCAT network. The XML-based file contains general information about the master and the configurations of every slave device connected to the master. To produce ENI file for Common Motion API, press the [**Generate ENI file**] button and choose the path you want to save.

5.1.1.3 Slave Page

Once user chooses the slave node in the left tree node, the right panel will show the slave information. It contains two tables, ***Information*** and ***FW***.

5.1.1.3.1 Information

The information tab shows the slave information such as vendor name, vendor ID, device name, device type, product code, revision number, serial number and slave alias ID which defined in the EEPROM and corresponding ESI file. If vendor logo described in detail in the ESI file, the logo might be shown in front of the vendor name.



ID No.

The ID No. (alias ID) is defined in the EEPROM and can be modify manually. For Common Motion API, the lowest value of alias ID is 0x0001 and the highest is 0xFFFF. After changing this value, press [**Change ID**] button for modifying the data in EEPROM. If the slave is ADAM-5000/ECAT, the last 3 digits of alias ID is assigned by 3 locate switch in the left-hand side of slave. When the ADAM-5000/ECAT power on, the last 3 digits value of alias in EEPROM will be covered by the setting of locate switch, but the digit in thousands will keep until user changes the ID manually.

Note! *Due to EhterCAT Master limitation, the [**Change ID**] function will fail in the beginning. It will work correctly after a while (the time varies directly as the number of slaves)*

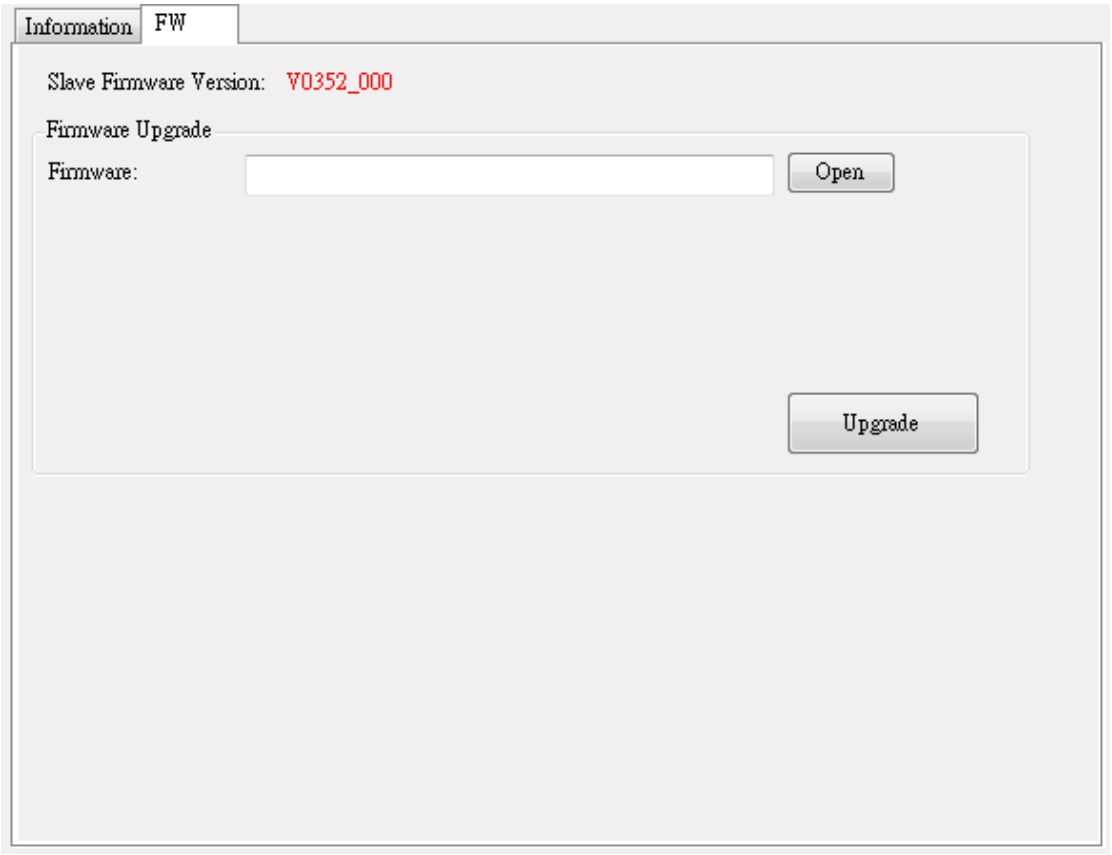
Data Link Layer Status

The status show in the middle of this tab indicates the DLL status (data link layer status) of the individual ports of the EtherCAT slave. The DLL status can have two different states:

Status	Description
Link up	The port is linked up and transmits the EtherCAT datagram.
Loop Closed	The port is closed.

5.1.1.3.2 FW

In this tab, user can know firmware version of slave and can upgrade the firmware.



Click [**Open**] to select lastest firmware file you have acquired. Clicking [**Upgrade**] will activate the downloading procedure to hardware and progress bar will show the task process.

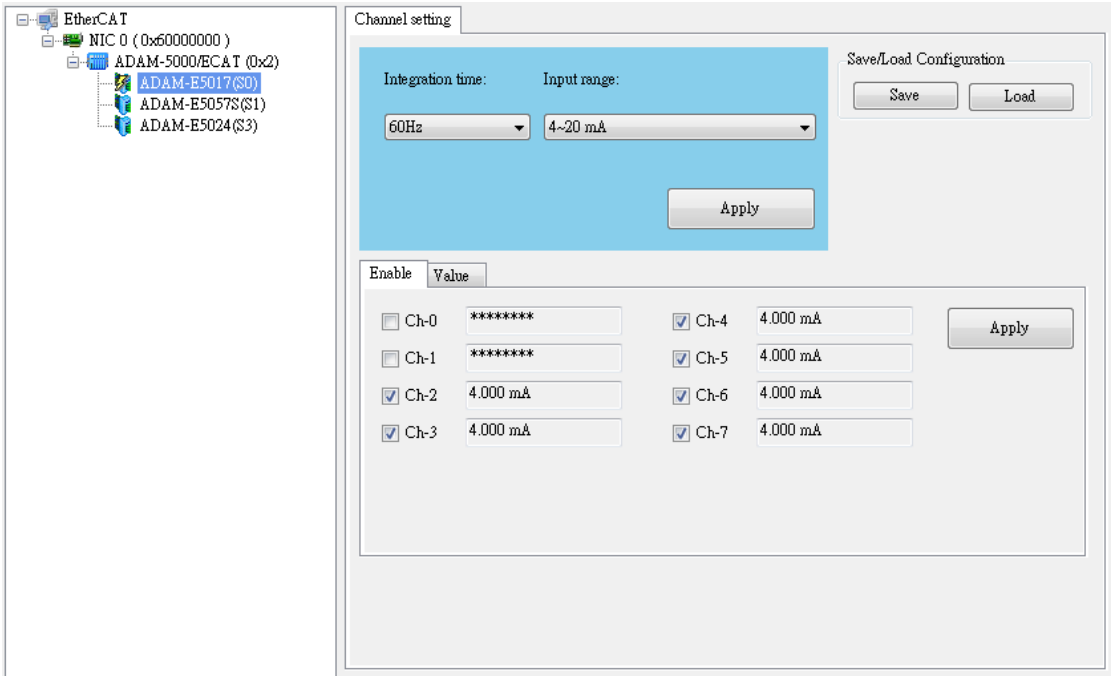
5.1.1.4 Module Page

If the EtherCAT slave is ADAM-5000/ECAT, the modules plug in ADAM-5000/ECAT will show in the slave tree node.

5.1.1.4.1 Analog Input Modules

- ***ADAM-E5017***

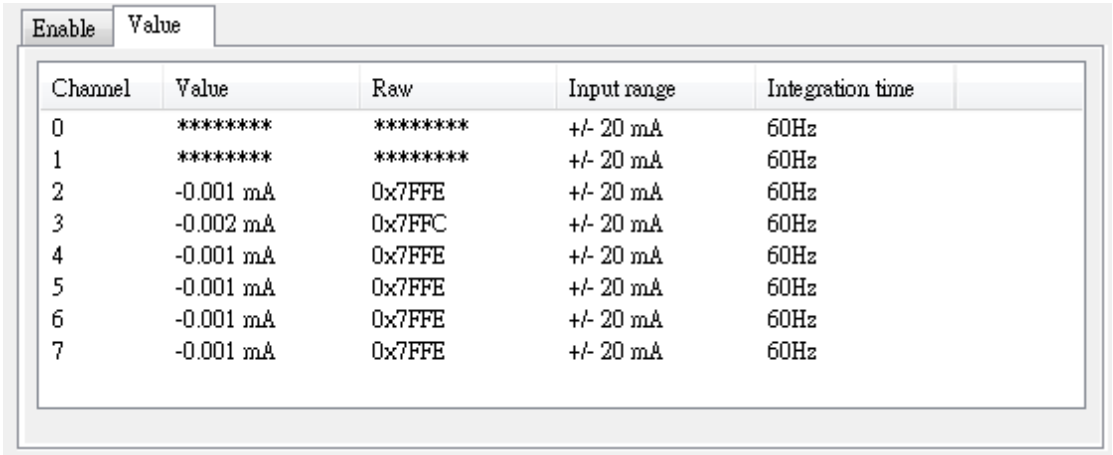
This page support ADAM-E5017 8-ch Analog Input Module.



Input Information

There are 8 channels in ADAM-E5017 and the current value in engineering units are shown in the [**Enable]** block in this page

Switching to the [**Value]** block, the current analog input value both in engineering units and raw data will show in this block. The other setting such as input range and integration time of each channel will also be shown.



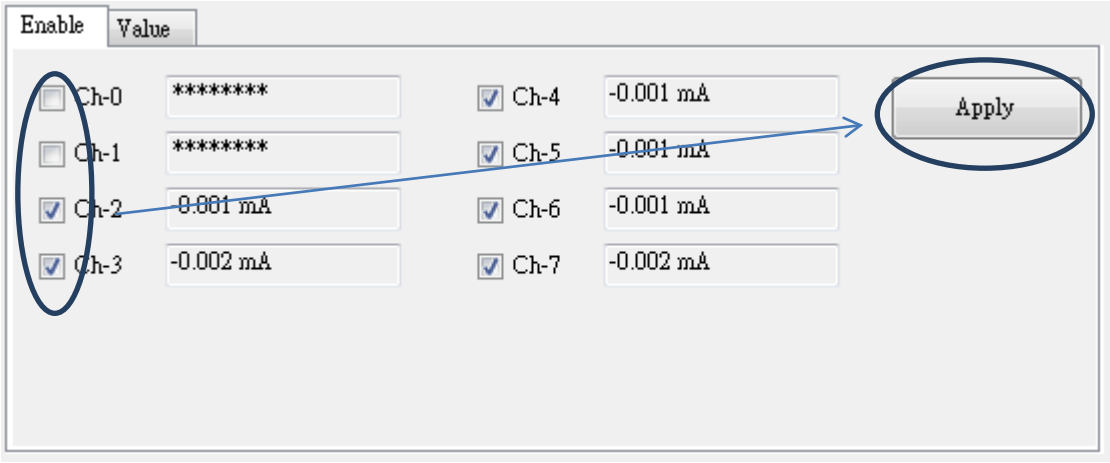
To get the data in engineering units, the conversion formula are shown below

Input Range	Raw data units (R) to Engineering value (E)	Engineering value (E) to Raw data units (R)
+/- 10 V	E = R * 20 / 65535 -10 (V)	R = (E + 10) * 65535 / 20
+/- 5 V	E = R * 10 / 65535 - 5 (V)	R = (E + 5) * 65535 / 10
+/- 1 V	E = R * 2 / 65535 - 1 (V)	R = (E + 1) * 65535 / 2
+/- 500 mV	E = R * 1000 / 65535 - 500 (mV)	R = (E + 500) * 65535 / 1000
+/- 150 mV	E = R * 300 / 65535 - 150 (mV)	R = (E + 150) * 65535 / 300
+/- 20 mA	E = R * 40 / 65535 - 20 (mA)	R = (E + 20) * 65535 / 40
4~20 mA	E = R * 16 / 65535 + 4 (mA)	R = (E - 4) * 65535 / 16

Channel Enable Status

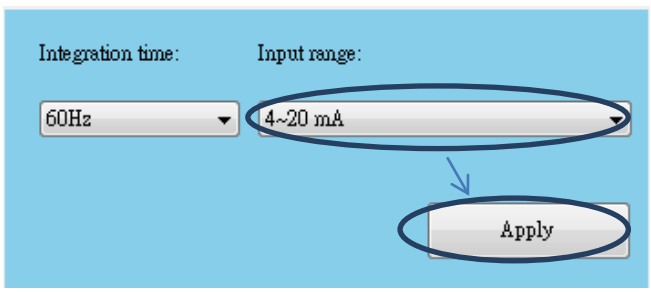
In the [**Enable]** block in this page, user can disable the analog input channel by uncheck the check box in front of each channel then click [**Apply**] to apply the setting.

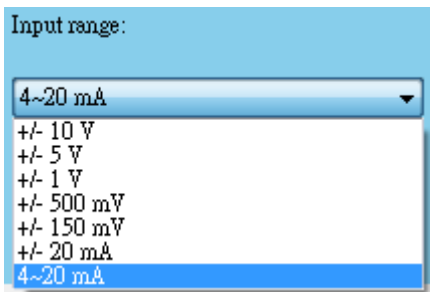
The module sequential process input data of each channel from channel 0 to channel 7 and then back to channel 0 again and again. If we just enable one channel of this module, the update rate will be 8 times compared to the situation which all 8 channels are enabled. Therefore, user can disable unused channels to speed up the update rate.



Input Range

All of channels of ADAM-E5017 module use the same setting of input range. There are 7 type of input can be selected. Choosing the properly range of input and click [**Apply**] to apply the setting.

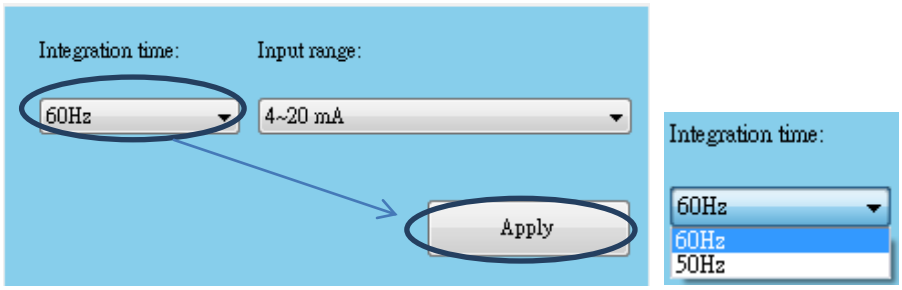




Integration Time

The integration time is designed for filtering the noise which the frequency of the input signal is 50Hz or 60 Hz.

All of channels of ADAM-E5017 module use the same setting of integration time. There are 2 type of integration time can be selected. Choosing the properly range of input and click [**Apply**] to apply the setting.

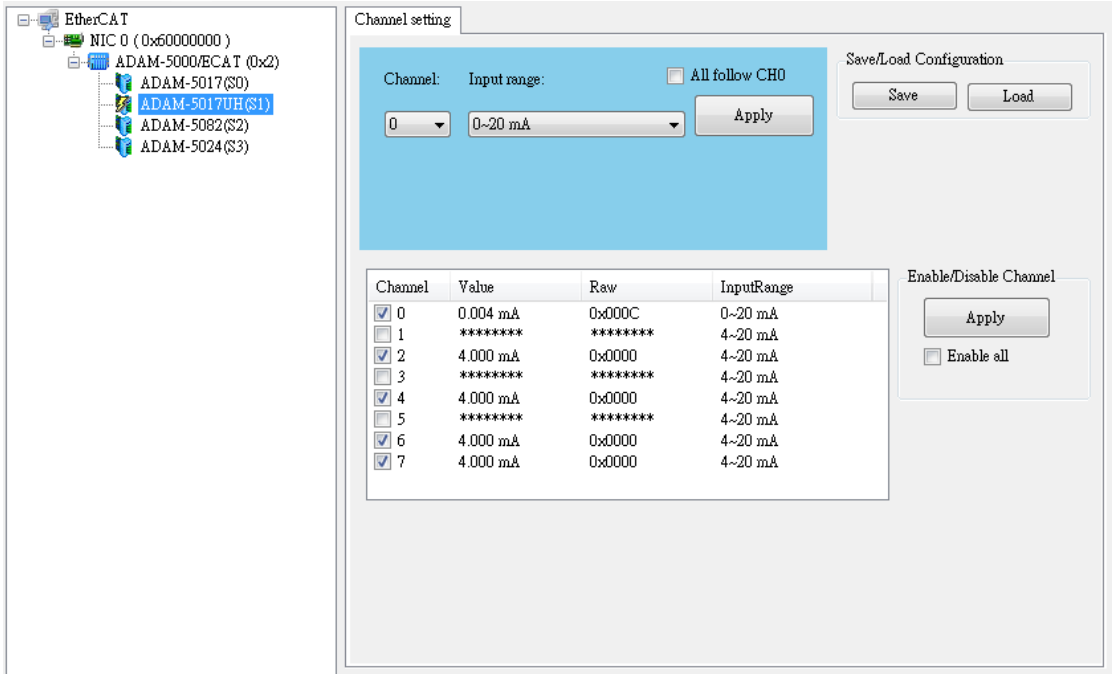


Save and Load Configuration

In this block, you can save the all of setting of AI module by xml file and load the configuration file from previous setting to simplify the configuration procedure.

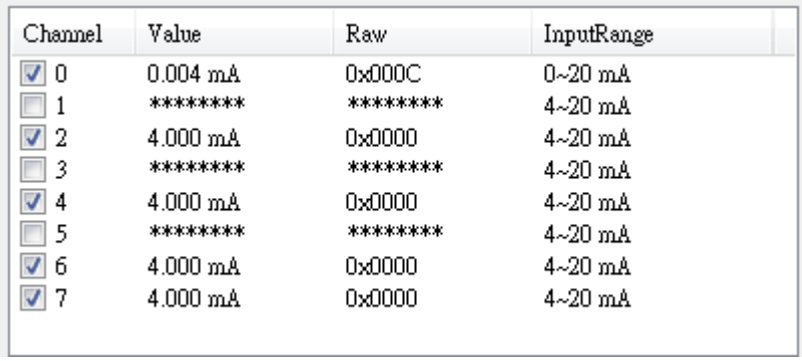
- ***ADAM-E5017UH***

This page support ADAM-E5017UH 8-ch Ultra High Speed Analog Input Module.



Input Information

The information block shows the current analog input value both in engineering units and raw data. The other setting such as input range of each channel will also be shown.



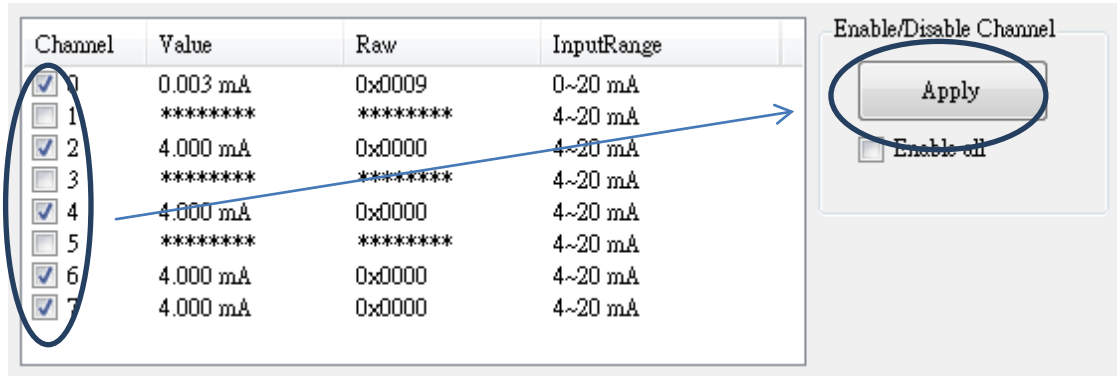
Channel	Value	Raw	InputRange
☑ 0	0.004 mA	0x000C	0~20 mA
☐ 1	********	********	4~20 mA
☑ 2	4.000 mA	0x0000	4~20 mA
☐ 3	********	********	4~20 mA
☑ 4	4.000 mA	0x0000	4~20 mA
☐ 5	********	********	4~20 mA
☑ 6	4.000 mA	0x0000	4~20 mA
☑ 7	4.000 mA	0x0000	4~20 mA

To get the data in engineering units, the conversion formula are shown below

Input Range	Raw data units (R) to Engineering value (E)	Engineering value (E) to Raw data units (R)
4~20 mA	E = R * 16 / 4095 + 4 (mA)	R = (E - 4) * 4095 / 16
+/- 10 V	E = R * 20 / 4095 -10 (V)	R = (E + 10) * 4095 / 20
0~500 mV	E = R * 500 / 4095 (mV)	R = E * 4095 / 500
0~20 mA	E = R * 20 / 4095 (mA)	R = E * 4095 / 20
0~10 V	E = R * 10 / 4095 (V)	R = E * 4095 / 10

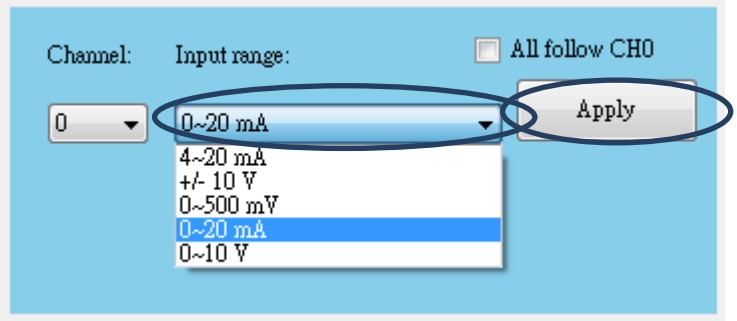
Channel Enable Status

In the information block, user can disable the analog input channel by uncheck the check box in front of each channel then click [**Apply**] to apply the setting.



Input Range

Different from ADAM-E5017, ADAM-E5017UH module offers user to set input range of each channel separately. There are 5 type of input can be selected. Choosing the properly range of input and click [**Apply**] to apply the setting.

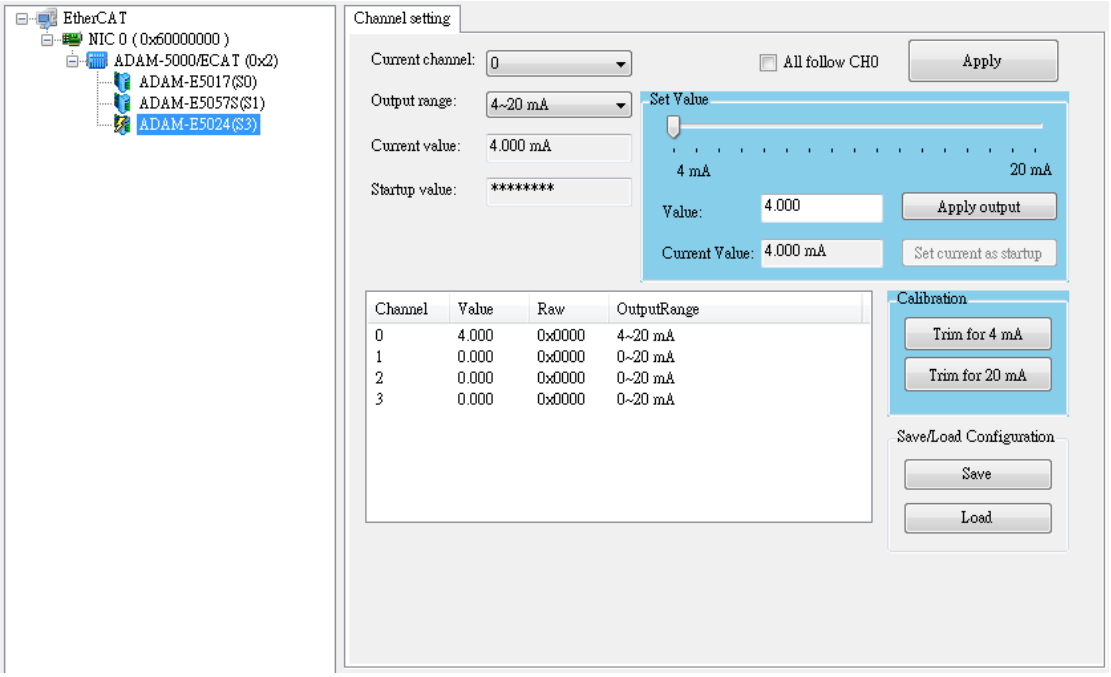


Save and Load Configuration

In this block, you can save the all of setting of AI module by xml file and load the configuration file from previous setting to simplify the configuration procedure.

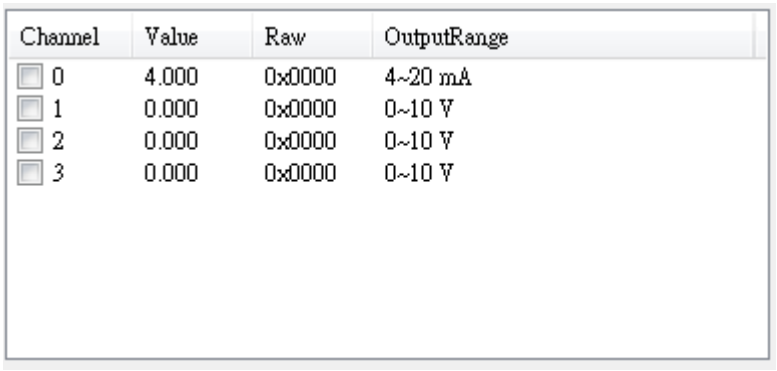
5.1.1.4.2 Analog Output Modules

This page support ADAM-E5024 4-ch Analog Output Module



Output Information

The information block shows the current analog output value both in engineering units and raw data. The other setting such as output range of each channel will also be shown.

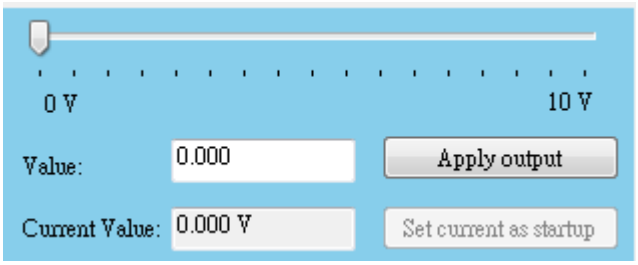


To get the data in engineering units, the conversion formula are shown below

Output Range	Raw data units (R) to Engineering value (E)	Engineering value (E) to Raw data units (R)
0~20 mA	E = R * 20 / 4095 (mA)	R = E * 4095 / 20
4~20 mA	E = R * 16 / 4095 + 4 (mA)	R = (E – 4) * 4095 / 16
0~10 V	E = R *10 / 4095 (V)	R = E * 4095 / 10

Set Value

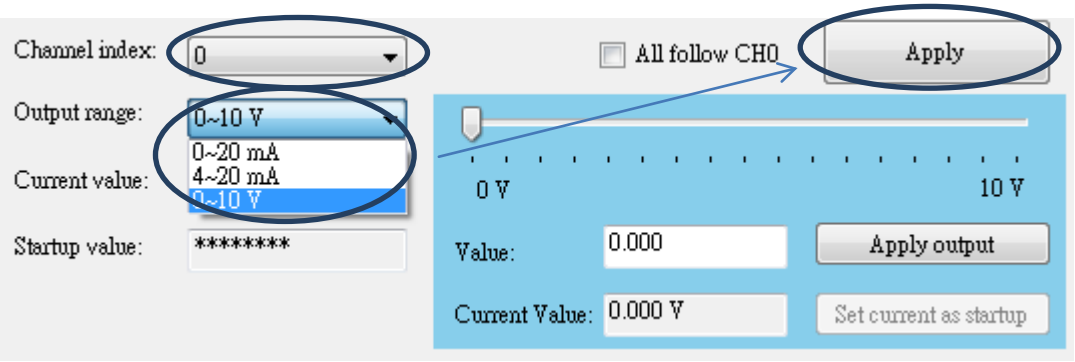
There are 4 channels in ADAM-E5024 and the current value in engineering units can be set in this page. Before you apply the output of current value, be sure the type of output range is correct.



Output Range

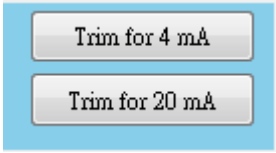
The output range can be set to different type for different channel of ADAM-E5024 module. There are 3 type of output can be selected. Selecting the channel you want to modify and choosing the properly range of output and click [**Apply**] to apply the setting.

If you check the [**All follow CH0]**, all of the channels in this module will follow the changes of setting of channel 0.



Calibration

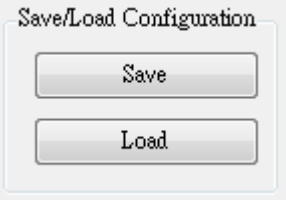
To trim for 4 mA or 20mA, click the button in the Calibration block. The output range of channel must be set to 0~20mA or 4~20mA, the corresponding trim button will appear.



User must use accurate instrument to calibrate the module, otherwise the channel value will be not accurate.

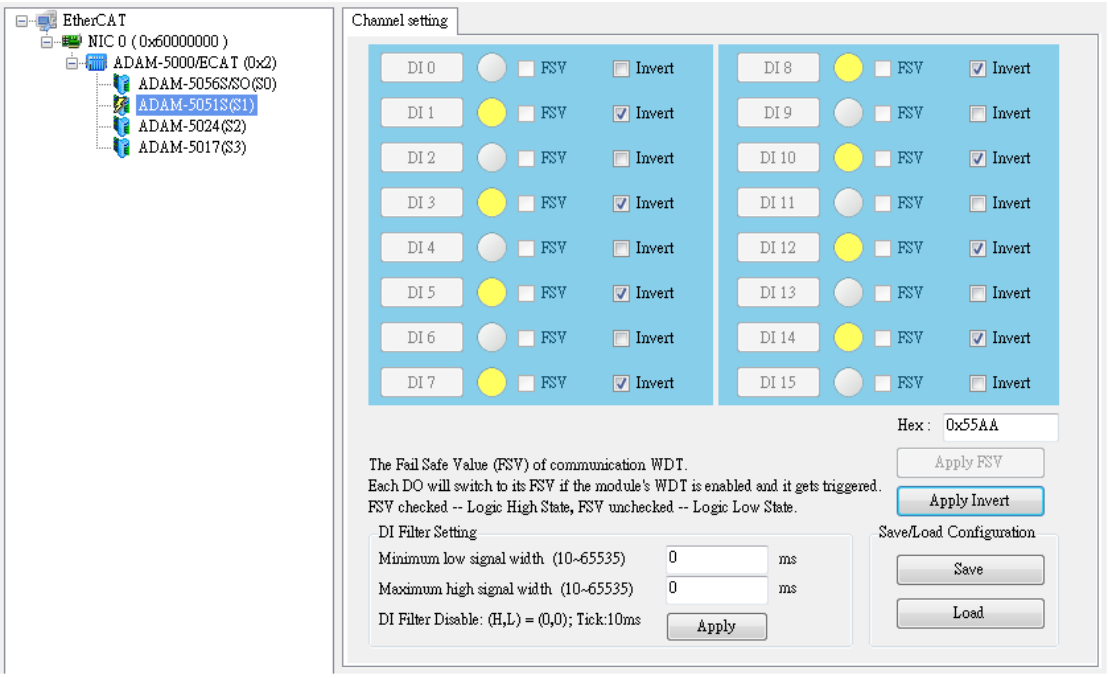
Save and Load Configuration

In this block, you can save the all of setting of AO module by xml file and load the configuration file from previous setting to simplify the configuration procedure.



5.1.1.4.3 Digital Input Modules

This page support ADAM-E5051S 16-ch Isolated Digital Input Module with LED and ADAM-E5053S 32-ch Isolated Digital Input Module.



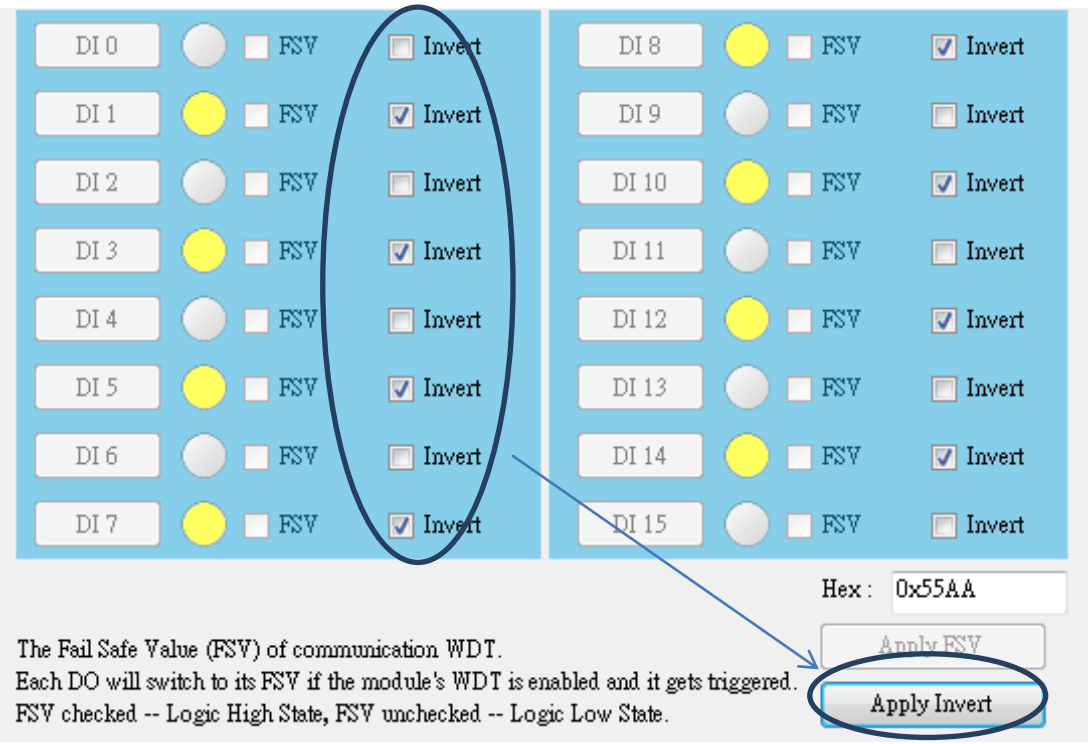
Input Information

There are little differences between ADAM-E5051S and ADAM-E5053S in the information block due to the channel number.

 indicates the DI is in effect (ON) and its value is 1;  indicates the DI is not in effect (OFF) and its value is 0.

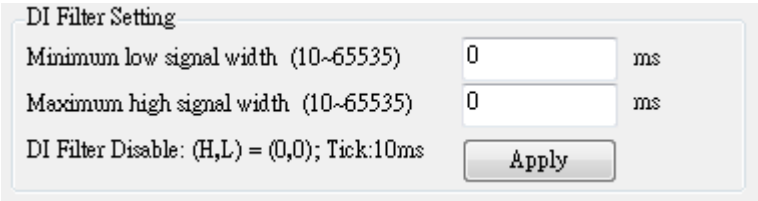
Invert

If you need to invert the signal level of channels, checking the **Invert** check box on each channel and clicking [**Apply Invert**] to apply the setting.



Digital Filter Scale (Minimum Low Signal Width)

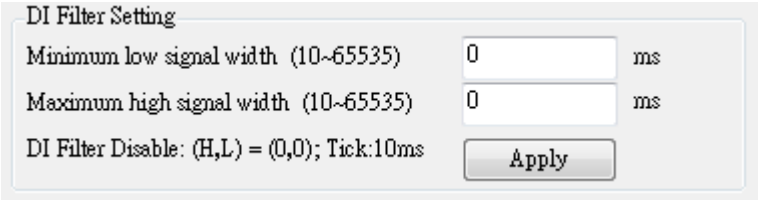
The digital input filter is a feature that eliminates noise from input signals. Once the digital input filter is set, it stores data during each sampling and then compares the data of the input terminal state. If all the input signals reach the LOW level within the filtering time, use the state as the input terminal value. If not, use the previous value.



The range of the digital filter time is 10 to 65535 in minimum second. To disable the filter, the value of this index must be set to 0.

Digital Filter Scale (Maximum High Signal Width)

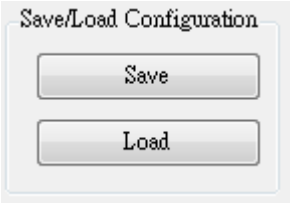
The digital input filter is a feature that eliminates noise from input signals. Once the digital input filter is set, it stores data during each sampling and then compares the data of the input terminal state. If all the input signals reach the HIGH level within the filtering time, use the state as the input terminal value. If not, use the previous value.



The range of the digital filter time is 10 to 65535 in minimum second. To disable the filter, the value of this index must be set to 0.

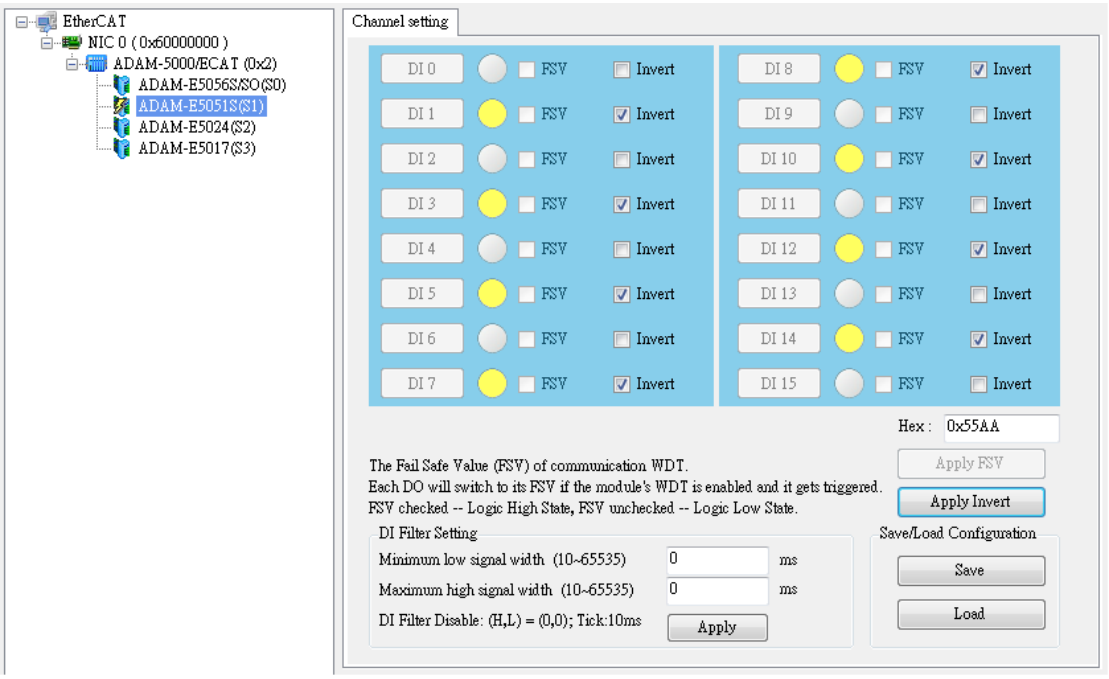
Save and Load Configuration

In this block, you can save the all of setting of DI module by xml file and load the configuration file from previous setting to simplify the configuration procedure.



5.1.1.4.4 Digital / Relay Output Modules

This page support ADAM-E5056S/SO 16-ch Isolated Digital Output Module with LED, ADAM-E5057S 32-ch Isolated Digital Output Module and ADAM-E5069 Relay Output Module.



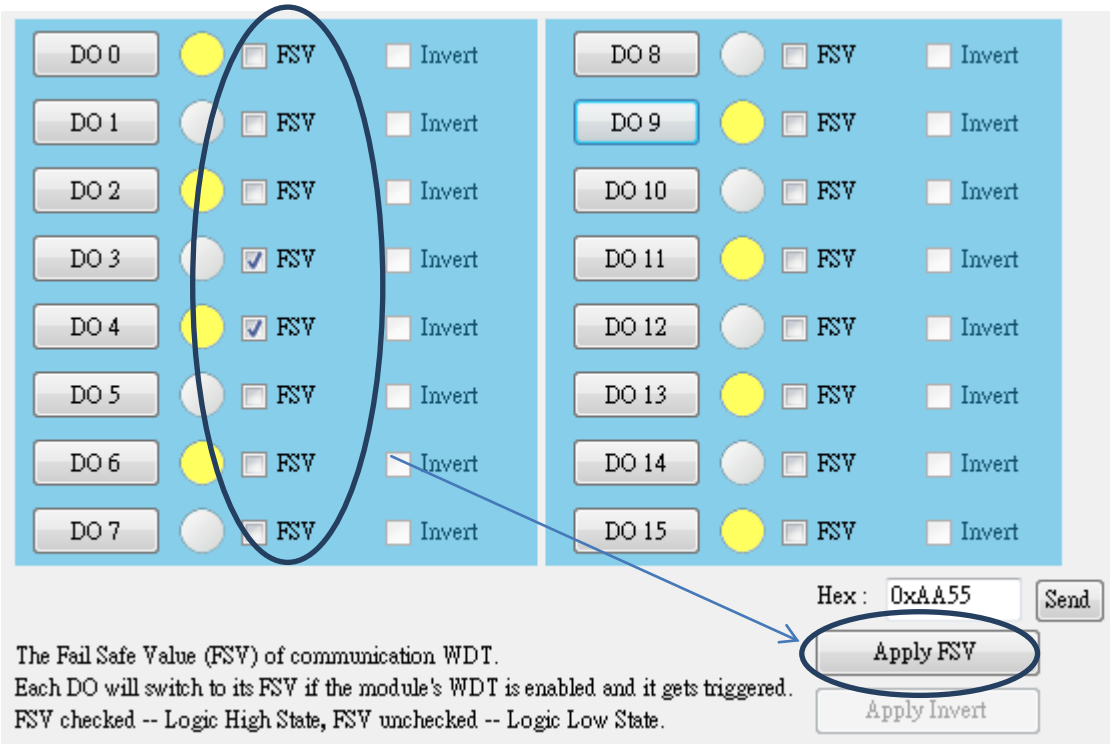
Output Information

There are little differences between each module in the information block due to the numbers of channel.

 indicates the DO is in effect (ON) and its value is 1;  indicates the DO is not in effect (OFF) and its value is 0.

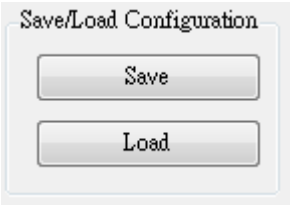
FSV (The Fail Safe Value of communication WDT)

If there are some problem to ADAM-5000/ECAT such as network disconnect, each DO will switch to its fail safe value if the module's WDT is enabled. If you need to enable the FSV, checks the [**FSV]** check box on each channel and clicks [**Apply FSV**] to apply the setting.



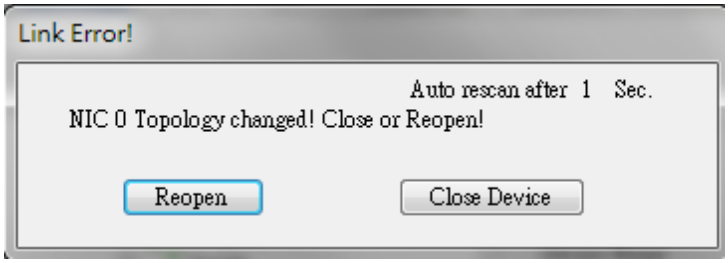
Save and Load Configuration

In this block, you can save the all of setting of DO module by xml file and load the configuration file from previous setting to simplify the configuration procedure.



5.1.1.4.5 Link Status Change

If the link status change such as adding or deleting a slave in online mode will trigger a “topology changed” event. And if the cable link NIC and slave are broken down, this will trigger a “link down” event. Then users will see the below dialog box:



Clicking “Reopen” button will reload bus and “Close Device” will close the NIC. If the link status is “topology changed” or “link up”, it will rescan this NIC automatically after three seconds.

5.1.2 TwinCAT®

The TwinCAT software is developed by Beckhoff for operating EtherCAT device. By importing the ESI (EtherCAT Slave Information) files to the folder of TwinCAT, the ADAM-5000/ECAT can be run properly.

5.1.2.1 ESI files

For each EtherCAT Slave a device description, the so called EtherCAT Slave Information (ESI) has to be delivered. This is done in form of an XML file (eXtensible Markup Language). It describes EtherCAT specific as well as application specific features of the slave. Those ESI files must be imported into TwinCAT IO EtherCAT installation folder before TwinCAT started.

The default path of installation folder is

C:\TwinCAT\Io\EtherCAT

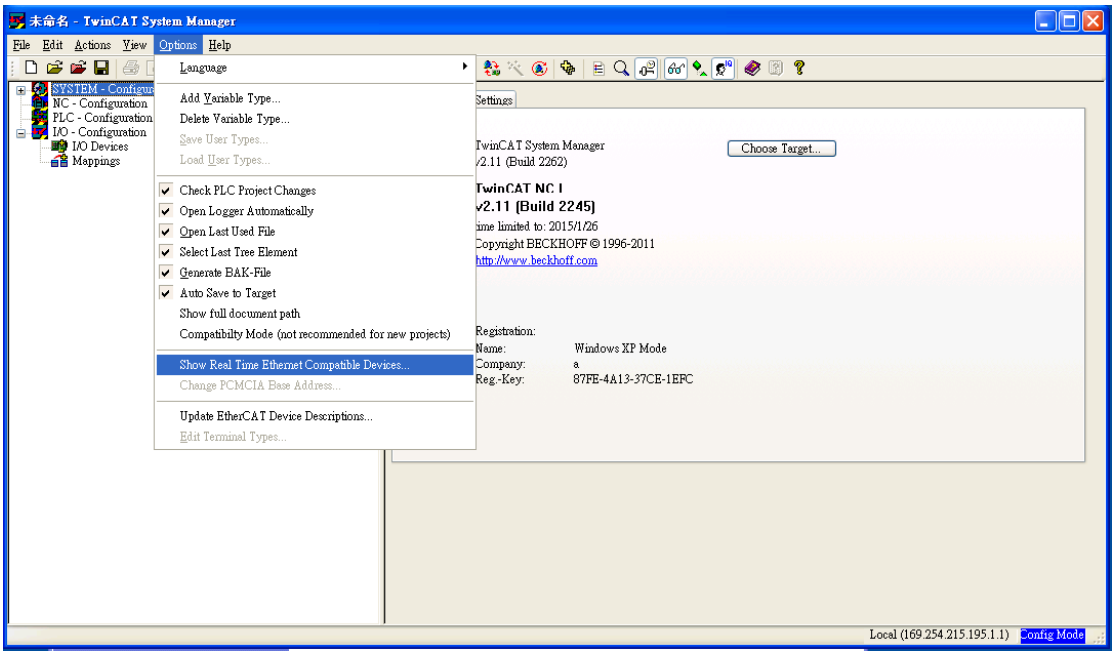
After importing the ESI file, reopen the TwinCAT and wait for the utility rebuild the EtherCAT device description cache.

5.1.2.2 Main Form

To start TwinCAT System Manager Software, right-click on the TwinCAT icon in the system tray and choose System Manager.

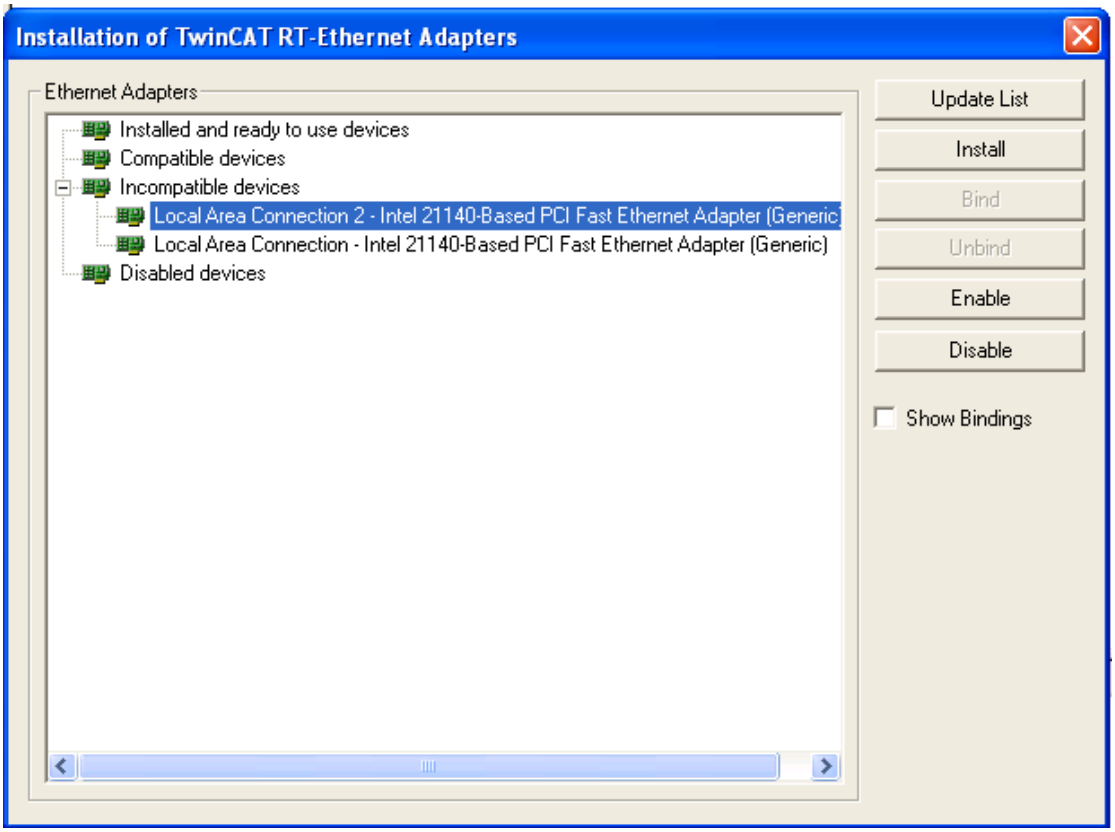
5.1.2.2.1 Install EtherCAT Driver

The first execution of TwinCAT, the network interface card driver for EtherCAT must be installed. Choose the [**Options**] menu -> [**Show Real Time Ethernet Compatible Drivers…**]



All of the network interface cards are listed in the [**Incompatible devices**] tree node. Select the device which wants to be used for EtherCAT by TwinCAT and press the [**Install**] button in the right hand side of this form. After the driver has been installed, the devices will be shown in the [**Installed and ready to use devices**] node.

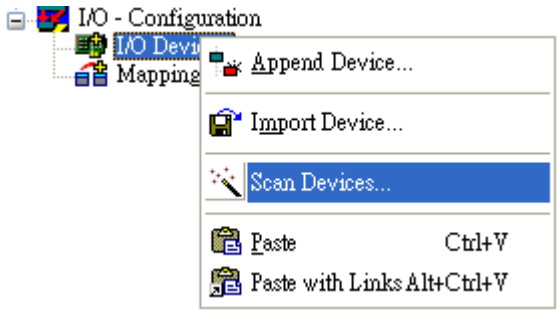
Even though the driver has been installed for TwinCAT used, it is no different from normal network interface card and can still connect to Ethernet.


Installation of TwinCAT RT-Ethernet Adapters
Ethernet Adapters
Installed and ready to use devices
Compatible devices
Incompatible devices
Local Area Connection 2 - Intel 21140-Based PCI Fast Ethernet Adapter (Generic)
Local Area Connection - Intel 21140-Based PCI Fast Ethernet Adapter (Generic)
Disabled devices
Update List
Install
Bind
Unbind
Enable
Disable
Show Bindings


Hardware Installation
The software you are installing for this hardware:
TwinCAT RT-Ethernet Intermediate Miniport Driver
has not passed Windows Logo testing to verify its compatibility with Windows XP. (Tell me why this testing is important.)
Continuing your installation of this software may impair or destabilize the correct operation of your system either immediately or in the future. Microsoft strongly recommends that you stop this installation now and contact the hardware vendor for software that has passed Windows Logo testing.
Continue Anyway
STOP Installation

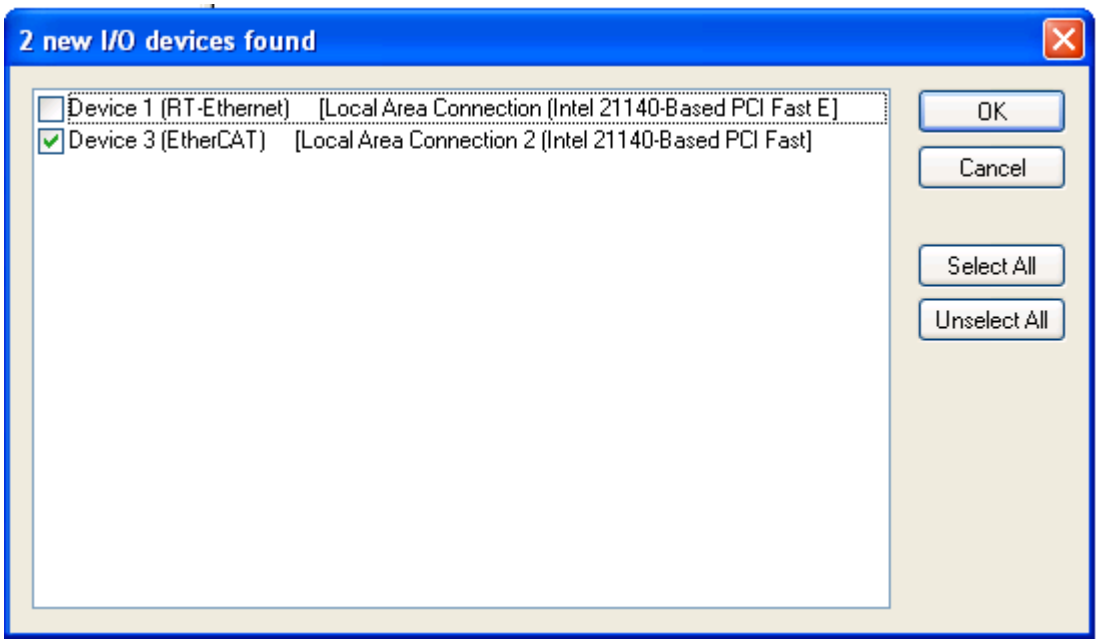
5.1.2.2.2 Scan for ADAM-5000/ECAT devices

Select [**I/O – Configuration**] -> [**I/O Devices**] menu in the left hand side of main form, click right mouse button and select [**Scan Devices…**] for scan the EtherCAT devices. Before scan the devices, ensure the mode of TwinCAT is in the [**Config Mode**]. The current mode of TwinCAT System Manager is displayed in the lower right corner of utility and user can switch the mode by click the button  to  in the toolbar.



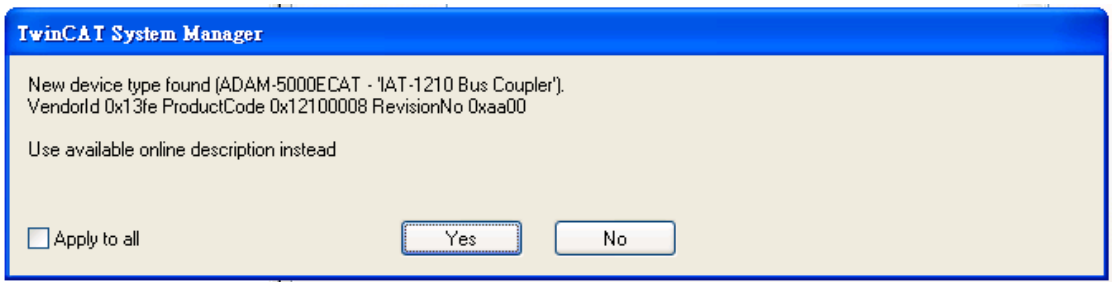
Click [OK] when the HINT window appears.

The list of network interface cards will be shown in the window. When there is/are slave(s) connected to the network interface card and had been found by TwinCAT, the check box of these devices will be checked.



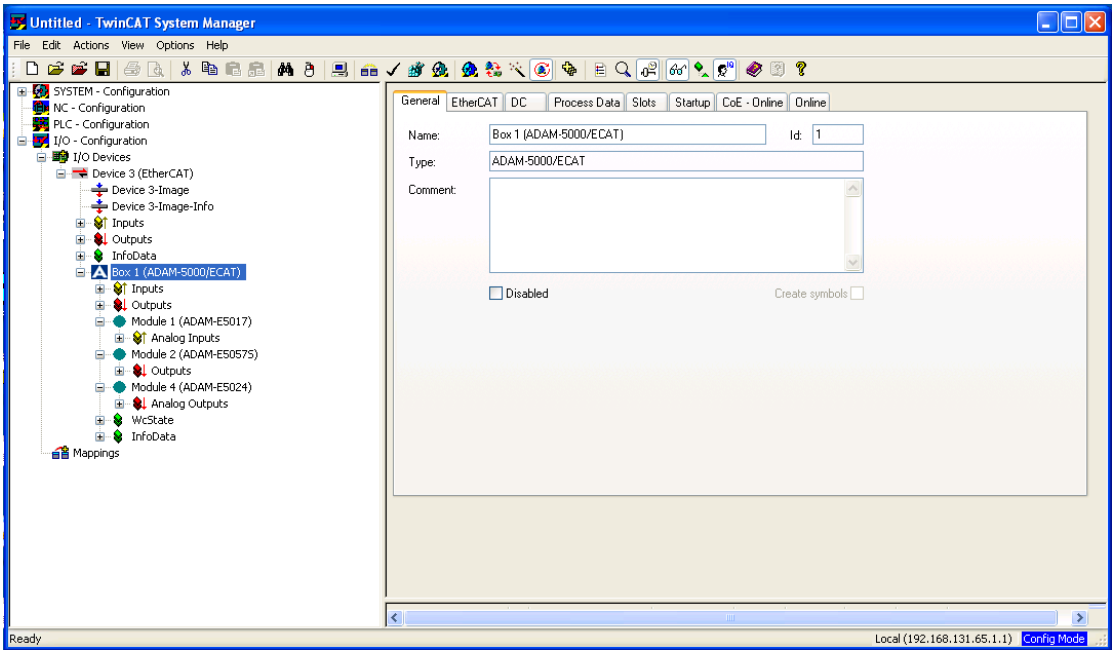
Click [Yes] to scan for EtherCAT slave device(s).

If the ESI files do not import into TwinCAT IO EtherCAT installation folder, the warning windows will show. Because of the complete EEPROM information, the ADAM-5000/ECAT can be operated correctly by online description. But some EtherCAT slave devices may not follow this procedure to operate; therefore we still recommend user import the ESI files.



Click [Yes] to activate the slave(s).

After all of works are done, the ADAM-5000/ECAT will be shown as a slave device of the EtherCAT master and the “RUN” LED in ADAM-5000/ECAT will turn to solid green, than user can start to operate the device.

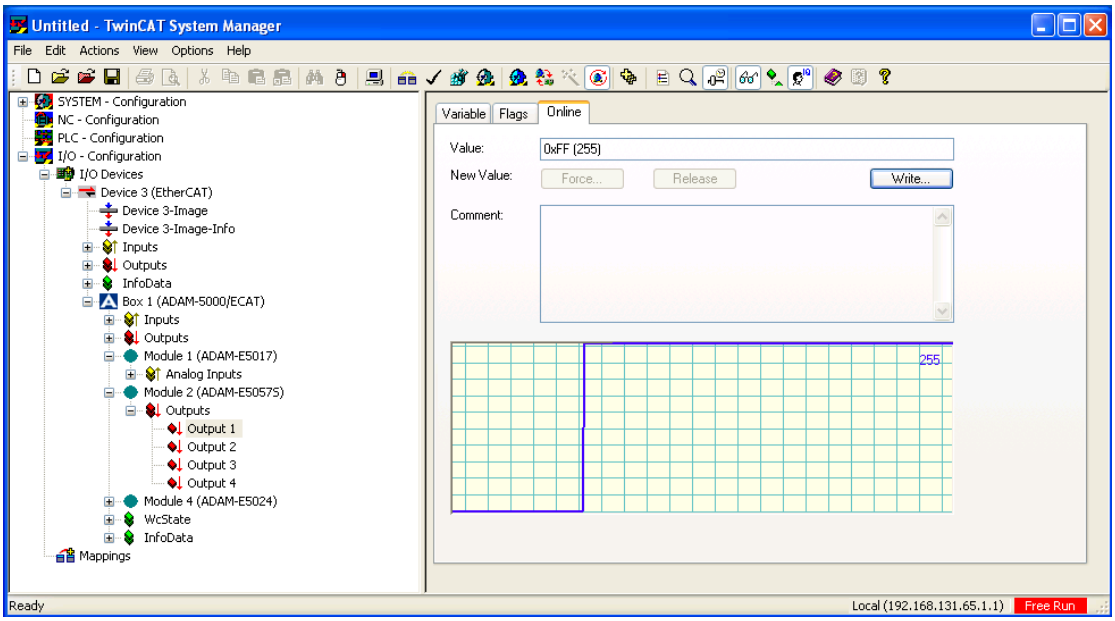


5.1.2.2.3 PDO upload & download

The input and output data of the EtherCAT slave are represented as CANopen process data objects (PDO). In the left-hand window of the TwinCAT System Manager, click on the branch you wish to configure.

For example, the ADAM-E5057S DO module has been plug in the slot 1 of ADAM-5000/ECAT. There are 32 digital output channels, each port composed of 8 channels. To set the output value of first port (DO0 ~ DO7), click on the branch "***Output 1***" in the left-hand side of window. In the ***Online*** tab, click ***Write*** button to set the value.

In this case, the value of first port is set to 0xFF, which means the DO0 to DO7 signals are set to HIGH level.

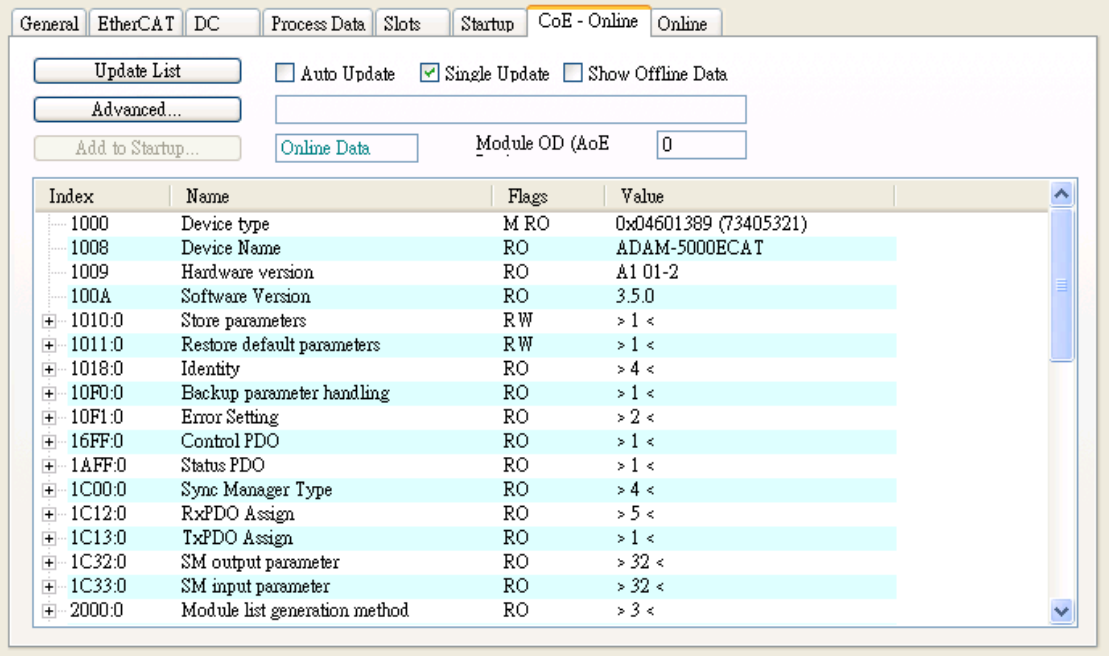


In the branch of input module (Analog Input & Digital Input), the window will show the input signals of each

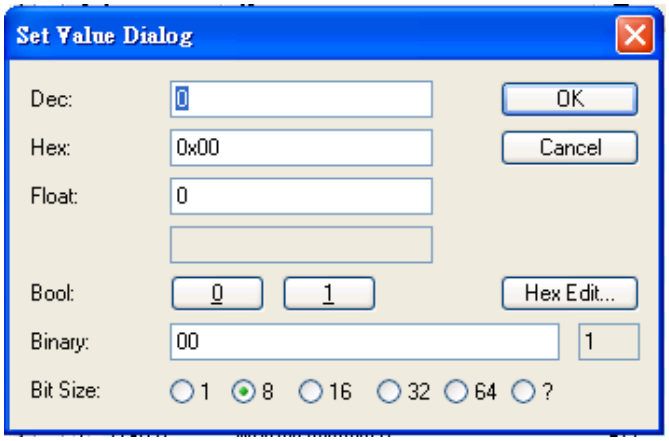
channel ports and user can diagnosis the input signal via the trend plot in this page.

5.1.2.2.4 CoE – Online Tab (SDO upload & download)

For ADAM-5000/ECAT, the EtherCAT slave supports the CANopen over EtherCAT (CoE) protocol. The ***CoE - Online*** tab in TwinCAT is displayed and this dialog lists the content of the object list of the slave (SDO upload).



To modify the content of an object which the access authority flag is RW, double click the column of object and modify the value in the ***Set Value Dialog***.



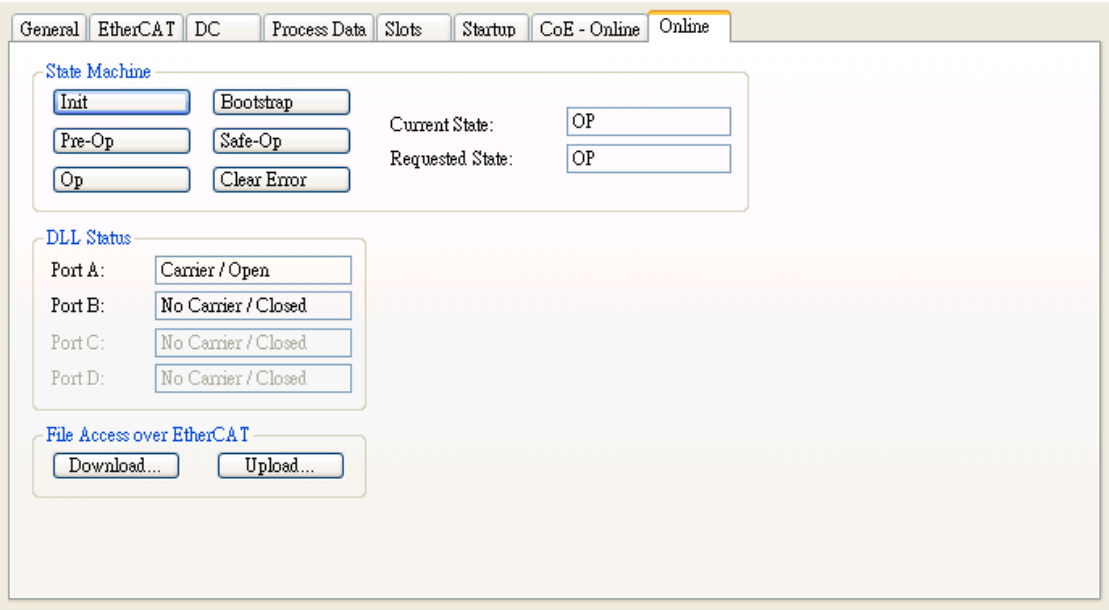
Object list display

Column		Description
Index		Index and sub-index of the object
Name		Name of the object
Flags	RW	The object can be read, and data can be written to the object (read/write)
	RO	The object can be read, but no data can be written to the object (read only)
	P	An additional P identifies the object as a process data object.

Value	Value of the object

5.1.2.2.5 Online Tab

In this Tab, user can know and switch the EtherCAT state machine; data link layer status of slave and upgrade firmware of ADAM-5000/ECAT by File Access over EtherCAT (FoE). The detail information of this tab from Beckhoff is shown below.



State Machine

State	Description
Init	This button attempts to set the EtherCAT device to the Init state.
Bootstrap	This button attempts to set the EtherCAT device to the Bootstrap state.
Pre-Op	This button attempts to set the EtherCAT device to the pre-operational state.
Safe-Op	This button attempts to set the EtherCAT device to the safe-operational state.
Op	This button attempts to set the EtherCAT device to the operational state.
Clear Error	This button attempts to delete the fault display. If an EtherCAT slave fails during change of state it sets an error flag. Example: An EtherCAT slave is in PREOP state (pre-operational). The master now requests the SAFEOP state (safe-operational). If the slave fails during change of state it sets the error flag. The current state is now displayed as ERR PREOP. When the Clear Error button is pressed the error flag is cleared, and the current state is displayed as PREOP again.
Current State	Indicates the current state of the EtherCAT device.
Requested State	Indicates the state requested for the EtherCAT device.

DLL Status

Indicates the DLL status (data link layer status) of the individual ports of the EtherCAT slave. The DLL status can have four different states:

Status	Description
No Carrier / Open	No carrier signal is available at the port, but the port is open.
No Carrier / Closed	No carrier signal is available at the port, and the port is closed.
Carrier / Open	A carrier signal is available at the port, and the port is open.
Carrier / Closed	A carrier signal is available at the port, but the port is closed.

File Access over EtherCAT

Download	With this button a file can be written to the EtherCAT device.
Upload	With this button a file can be read from the EtherCAT device.

5.1.2.3 Analog Input /Output Modules

ADAM-E5017 (8-ch Analog Input Module)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
217Fh	01h	Module No	0-7	USINT	RW	V	--	Pre-Op Op
2180h-2187h	01h	Get Current Value	0-65535	USINT	RO	V	--	Pre-Op Op
2180h-2187h	02h	Channel Status	--	USINT	RO	V	--	Pre-Op
2180h	03h	Channel Masking Status	0-255	USINT	RW	V	--	Pre-Op
2180h	04h	Input Range	--	USINT	RW	V	--	Pre-Op
2180h	05h	Integration Time	--	USINT	RW	V	--	Pre-Op
2180h	06h – 07h	Calibration	--	USINT	RW	V	--	Pre-Op

ADAM-E5024 (4-ch Analog Output Module)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
217Fh	01h	Module No	0-7	USINT	RW	V	--	Pre-Op Op
2180h-2183h	01h	Get Current Value	0-4095	USINT	RO	V	--	Pre-Op Op
2180h-2183h	02h	Output Range	30-32	USINT	RW	V	--	Pre-Op
2180h-2183h	03h – 05h	Calibration	--	USINT	RW	V	--	Pre-Op

5.1.2.3.1 ADAM-E5017 8-ch Analog Input Module

The ADAM-E5017 is a 12-bit, 8-channel analog differential input module that provides programmable input ranges on all channels. It accepts millivolt inputs (±150mV, ±500mV), voltage inputs (±1V, ±5V and ±10V) and current input (±20 mA). The module provides data to the host computer in engineering units (mV, V or mA).

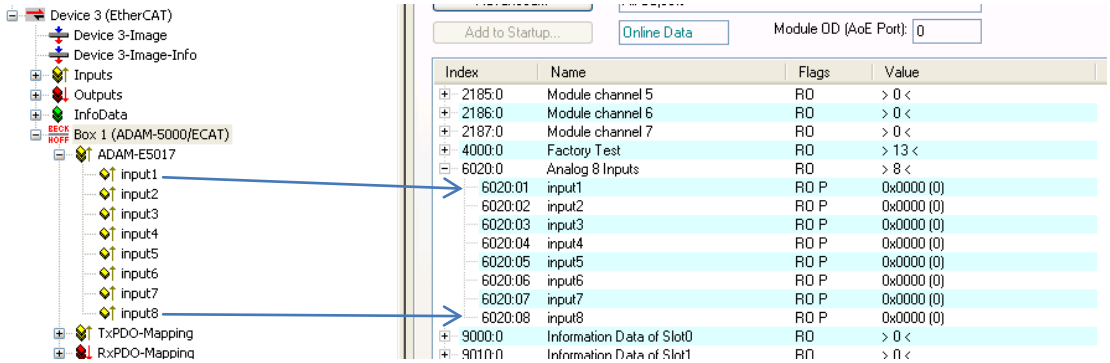
Get Current Value by PDO

There are 8 channels in ADAM-E5017. Specify the channel which need to diagnosis in the left-hand branch of TwinCAT System Manager.

This index (sub-index) shows the raw data of each channel of ADAM-E5017 module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
60x0h	01h – 08h	AI 1 – AI 8	0-65535	UINT	RO	--	Tx	Op

The x of index is a number which represent the slot number the ADAM-E5017 plugs in. For example, if you need to get the value of ADAM-E5017 AI module which plugs in the ***second*** slot of ADAM-5000/ECAT, the x will be set to 1.



To get the data in engineering units, the conversion formula are shown below

Output Range	Raw data units (R) to Engineering value (E)	Engineering value (E) to Raw data units (R)
+/- 10 V	E = R * 20 / 65535 -10 (V)	R = (E + 10) * 65535 / 20
+/- 5 V	E = R * 10 / 65535 - 5 (V)	R = (E + 5) * 65535 / 10
+/- 1 V	E = R * 2 / 65535 - 1 (V)	R = (E + 1) * 65535 / 2
+/- 500 mV	E = R * 1000 / 65535 - 500 (mV)	R = (E + 500) * 65535 / 1000
+/- 150 mV	E = R * 300 / 65535 - 150 (mV)	R = (E + 150) * 65535 / 300
+/- 20 mA	E = R * 40 / 65535 - 20 (mA)	R = (E + 20) * 65535 / 40
4~20 mA	E = R * 16 / 65535 + 4 (mA)	R = (E - 4) * 65535 / 16

Setting by SDO

Select Slot of Module

Select the slot of module which you want to diagnosis or change the setting. For 4-slot ADAM-5000/ECAT, the first (leftmost) slot number is 00h and the last (rightmost) slot number is 03h.

For example, if you need to modify the setting of ADAM-E5017 AI module which plug in the second slot of ADAM-5000/ECAT, the Module No (Index: 217Fh:01h) will be set to 01h.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
217Fh	01h	Module No	0-7	USINT	RW	V	--	Pre-Op Op

Get Current Value by SDO (NOT Recommend)

The current value can be reached by PDO and it will also show in following object index. There are 8 channels in ADAM-E5017, index 2180h and its sub-index are used for the setting of first channel, index 2181h and its sub-index are used for the setting of second channel and so on.

This index (sub-index) shows the raw data of each channel of ADAM-E5017 module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h-2187h	01h	SubIndex 001	0-65535	USINT	RO	V	--	Pre-Op Op

To get the data in engineering units, the conversion formula are shown below

Input Range	Raw data units (R) to Engineering value (E)	Engineering value (E) to Raw data units (R)
+/- 10 V	E = R * 20 / 65535 -10 (V)	R = (E + 10) * 65535 / 20
+/- 5 V	E = R * 10 / 65535 - 5 (V)	R = (E + 5) * 65535 / 10
+/- 1 V	E = R * 2 / 65535 - 1 (V)	R = (E + 1) * 65535 / 2
+/- 500 mV	E = R * 1000 / 65535 - 500 (mV)	R = (E + 500) * 65535 / 1000
+/- 150 mV	E = R * 300 / 65535 - 150 (mV)	R = (E + 150) * 65535 / 300
+/- 20 mA	E = R * 40 / 65535 - 20 (mA)	R = (E + 20) * 65535 / 40
4~20 mA	E = R * 16 / 65535 + 4 (mA)	R = (E - 4) * 65535 / 16

Channel Status

This index (sub-index) shows the channel status of each channel of ADAM-E5017 module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h-2187h	02h	SubIndex 002	--	USINT	RO	V	--	Pre-Op

To modify this value of input range, the EtherCAT state machine of slave must be switched to Pre-Op mode. After changing this setting, be sure to switch the state machine from Pre-Op back to Op mode.

The meaning of this value is

Channel Status	Value
Normal	00h
Current Over Range	02h
Current Under Range	04h
Channel Burn-Out	08h

Channel Enable/Disable Status

This index (sub-index) shows the enable/disable status of each channel of ADAM-E5017 module. A bit of this value corresponds to a channel, the MSB (Most Significant Bit) corresponds to the channel 7 and the LSB (Least Significant Bit) corresponds to the channel 0. To enable a channel, the bit value is set to be 1 and, on the contrary, 0 for disable.

For example, if you need to disable the channel 0-2 of ADAM-E5017 AI module, the value of this index must be set to F8h (11111000).

Index	Sub-Index	Object Name	Range	Data Type	Access	SDO	PDO	State
2180h	03h	SubIndex 003	0-255	USINT	RW	V	--	Pre-Op

To modify this value of enable/disable status, the EtherCAT state machine of slave must be switched to Pre-Op mode. After changing this setting, be sure to switch the state machine from Pre-Op back to Op mode.

Input Range

This index (sub-index) shows the input range of all channels of ADAM-E5017 module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h	04h	SubIndex 004	--	USINT	RW	V	--	Pre-Op

To modify this value of input range, the EtherCAT state machine of slave must be switched to Pre-Op mode. After changing this setting, be sure to switch the state machine from Pre-Op back to Op mode.

The meaning of this value is

Input Range	Value
+/- 10 V	08h
+/- 5 V	09h
+/- 1 V	0Ah
+/- 500 mV	0Bh
+/- 150 mV	0Ch
+/- 20 mA	0Dh
4~20 mA	07h

Integration Time

This index (sub-index) shows the integration time of channels of ADAM-E5017 module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h	05h	SubIndex 005	--	USINT	RW	V	--	Pre-Op

To modify this value of integration time, the EtherCAT state machine of slave must be switched to Pre-Op mode. After changing this setting, be sure to switch the state machine from Pre-Op back to Op mode.

The meaning of this value is

Input Range	Value
60 Hz	00h
50 Hz	80h

Calibration

These indexes (sub-indexes) are used to calibrate channels of ADAM-E5017 module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h	06h – 07h	SubIndex 006 – SubIndex 007	--	USINT	RW	V	--	Pre-Op

This function is available only for Advantech EtherCAT Utility.

5.1.2.3.2 ADAM-E5017UH 8-ch Ultra High Speed Analog Input Module

The ADAM-E5017UH is a 12-bit plus sign bit, 8-channel analog differential input module that provides programmable input ranges on each channel. It accepts voltage inputs (±10 V and 0-10 V) and current inputs (0-20 mA and 4-20 mA). The module provides data to the host microprocessor in engineering units (mV, V or mA) or two's complement format. Its sampling rate depends on the data format received: up to 200k Hz (total)

Get Current Value by PDO

There are 8 channels in ADAM-E5017UH. Specify the channel which need to diagnosis in the left-hand branch of TwinCAT System Manager.

This index (sub-index) shows the raw data of each channel of ADAM-E5017UH module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
60x0h	01h – 08h	AI 1 – AI 8	0-4095	UINT	RO	--	Tx	Op

The x of index is a number which represent the slot number the ADAM-E5017UH plugs in. For example, if you need to get the value of ADAM-E5017UH AI module which plugs in the ***second*** slot of ADAM-5000/ECAT, the x will be set to 1.

To get the data in engineering units, the conversion formula are shown below

Input Range	Raw data units (R) to Engineering value (E)	Engineering value (E) to Raw data units (R)
4~20 mA	E = R * 16 / 4095 + 4 (mA)	R = (E - 4) * 4095 / 16
+/- 10 V	E = R * 20 / 4095 -10 (V)	R = (E + 10) * 4095 / 20
0~500 mV	E = R * 500 / 4095 (mV)	R = E * 4095 / 500
0~20 mA	E = R * 20 / 4095 (mA)	R = E * 4095 / 20
0~10 V	E = R * 10 / 4095 (V)	R = E * 4095 / 10

Setting by SDO

Select Slot of Module

Select the slot of module which you want to diagnosis or change the setting. For 4-slot ADAM-5000/ECAT, the first (leftmost) slot number is 00h and the last (rightmost) slot number is 03h.

For example, if you need to modify the setting of ADAM-E5017UH AI module which plug in the second slot of ADAM-5000/ECAT, the Module No (Index: 217Fh:01h) will be set to 01h.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
217Fh	01h	Module No	0-7	USINT	RW	V	--	Pre-Op Op

Get Current Value by SDO (NOT Recommend)

The current value can be reached by PDO and it will also show in following object index. There are 8 channels in ADAM-E5017UH, index 2180h and its sub-index are used for the setting of first channel, index 2181h and its sub-index are used for the setting of second channel and so on.

This index (sub-index) shows the raw data of each channel of ADAM-E5017 module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h- 2187h	01h	SubIndex 001	0-4095	USINT	RO	V	--	Pre-Op Op

To get the data in engineering units, the conversion formula are shown below

Input Range	Raw data units (R) to Engineering value (E)	Engineering value (E) to Raw data units (R)
4~20 mA	E = R * 16 / 4095 + 4 (mA)	R = (E - 4) * 4095 / 16
+/- 10 V	E = R * 20 / 4095 -10 (V)	R = (E + 10) * 4095 / 20
0~500 mV	E = R * 500 / 4095 (mV)	R = E * 4095 / 500
0~20 mA	E = R * 20 / 4095 (mA)	R = E * 4095 / 20
0~10 V	E = R * 10 / 4095 (V)	R = E * 4095 / 10

Channel Status

This index (sub-index) shows the channel status of each channel of ADAM-E5017UH module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h- 2187h	02h	SubIndex 002	--	USINT	RO	V	--	Pre-Op

To modify this value of input range, the EtherCAT state machine of slave must be switched to Pre-Op mode. After changing this setting, be sure to switch the state machine from Pre-Op back to Op mode.

The meaning of this value is

Channel Status	Value
Normal	00h
Current Over Range	02h
Current Under Range	04h
Channel Burn-Out	08h

Channel Enable/Disable Status

This index (sub-index) shows the enable/disable status of each channel of ADAM-E5017UH module. A bit of this value corresponds to a channel, the MSB (Most Significant Bit) corresponds to the channel 7 and the LSB (Least Significant Bit) corresponds to the channel 0. To enable a channel, the bit value is set to be 1 and, on the contrary, 0 for disable.

For example, if you need to disable the channel 0-2 of ADAM-E5017UH AI module, the value of this index must be set to F8h (11111000).

Index	Sub-Index	Object Name	Range	Data Type	Access	SDO	PDO	State
2180h	03h	SubIndex 003	0-255	USINT	RW	V	--	Pre-Op

To modify this value of enable/disable status, the EtherCAT state machine of slave must be switched to Pre-Op mode. After changing this setting, be sure to switch the state machine from Pre-Op back to Op mode.

Input Range

This index (sub-index) shows the input range of each channel of ADAM-E5017UH module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h-2187h	04h	SubIndex 004	--	USINT	RW	V	--	Pre-Op

To modify this value of input range, the EtherCAT state machine of slave must be switched to Pre-Op mode. After changing this setting, be sure to switch the state machine from Pre-Op back to Op mode.

The meaning of this value is

Input Range	Value
4~20 mA	07h
+/- 10 V	08h
0~500 mV	43h
0~20 mA	46h
0~10 V	48h

Calibration

These indexes (sub-indexes) are used to calibrate channels of ADAM-E5017UH module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h	06h – 07h	SubIndex 006 – SubIndex 007	--	USINT	RW	V	--	Pre-Op

This function is available only for Advantech EtherCAT Utility.

5.1.2.3.3 ADAM-E5024 4-ch Analog Output Module

The ADAM-E5024 is a 4-channel analog output module. It receives its digital input from the host computer. The format of the data is engineering units. It then uses the D/A converter controlled by the system module to convert the digital data into output signals.

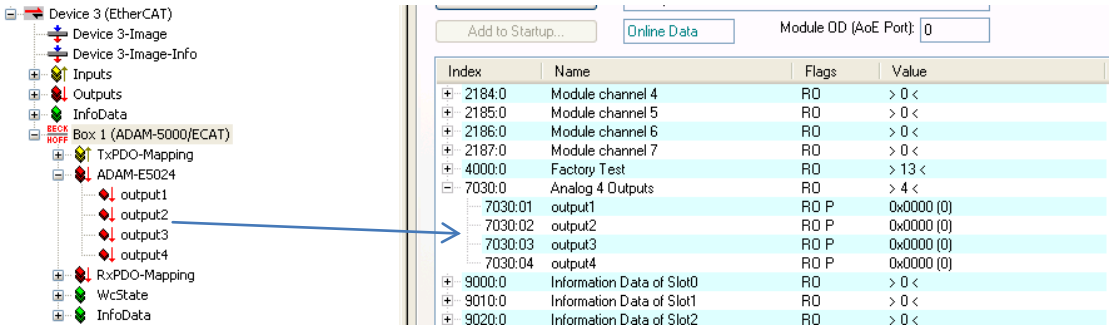
Set Current Value by PDO

There are 4 channels in ADAM-E5024. Specify the channel which need to diagnosis in the left-hand branch of TwinCAT System Manager.

This index (sub-index) shows the raw data of each port of ADAM-E5024 module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
70x0h	01h – 04h	AO 1 – AO 4	0-4095	UINT	WO	--	Rx	Op

The x of index is a number which represent the slot number the ADAM-E5056S/SO plugs in. For example, if you need to get the value of ADAM-E5056S/SO DO module which plugs in the ***second*** slot of ADAM-5000/ECAT, the x will be set to 1.



To get the data in engineering units, the conversion formula are shown below

Output Range	Raw data units (R) to Engineering value (E)	Engineering value (E) to Raw data units (R)
0~20 mA	E = R * 20 / 4095 (mA)	R = E * 4095 / 20
4~20 mA	E = R * 16 / 4095 + 4 (mA)	R = (E – 4) * 4095 / 16
0~10 V	E = R *10 / 4095 (V)	R = E * 4095 / 10

Setting by SDO

Select Slot of Module

Select the slot of module which you want to diagnosis or change the setting. For 4-slot ADAM-5000/ECAT, the first (leftmost) slot number is 00h and the last (rightmost) slot number is 03h.

For example, if you need to modify the setting of ADAM-E5024 AO module which plug in the second slot of ADAM-5000/ECAT, the Module No (Index: 217Fh:01h) will be set to 01h.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
217Fh	01h	Module No	0-7	USINT	RW	V	--	Pre-Op Op

Get Current Value Setting by SDO

There are 4 channels in ADAM-E5024, index 2180h and its sub-index are used for the setting of first channel, index 2181h and its sub-index are used for the setting of second channel and so on.

This index (sub-index) shows the raw data of each channel of ADAM-E5024 module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h- 2183h	01h	SubIndex 001	0-4095	USINT	RO	V	--	Pre-Op Op

To get the data in engineering units, the conversion formula are shown below

Output Range	Raw data units (R) to Engineering value (E)	Engineering value (E) to Raw data units (R)
0~20 mA	E = R * 20 / 4095 (mA)	R = E * 4095 / 20
4~20 mA	E = R * 16 / 4095 + 4 (mA)	R = (E – 4) * 4095 / 16
0~10 V	E = R *10 / 4095 (V)	R = E * 4095 / 10

Output Range

This index (sub-index) shows the output range of each channel of ADAM-E5024 module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h- 2183h	02h	SubIndex 002	30-32	USINT	RW	V	--	Pre-Op

To modify this value of output range, the EtherCAT state machine of slave must be switched to Pre-Op mode. After changing this setting, be sure to switch the state machine from Pre-Op back to Op mode.

The meaning of this value is

Output Range	Value
0~20 mA	30h
4~20 mA	31h
0~10 V	32h

Calibration

These indexes (sub-indexes) are used to calibrate each channel of ADAM-E5024 module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h-2183h	03h – 05h	SubIndex 003 – SubIndex 005	--	USINT	RW	V	--	Pre-Op

This function is available only for Advantech EtherCAT Utility.

5.1.2.4 Digital Input / Output Modules

ADAM-E5051S (16-ch Isolated Digital Input Module with LED)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
217Fh	01h	Module No	0-7	USINT	RW	V	--	Pre-Op Op
2180h- 2181h	01h	Get Current Value	0-255	USINT	RO	V	--	Pre-Op Op
2180h- 2181h	02h	Invert	0-255	USINT	RW	V	--	Pre-Op Op
2180h	03h	Digital Filter Scale (Low)	0-65535	USINT	RW	V	--	Pre-Op Op
2180h	04h	Digital Filter Scale (High)	0-65535	USINT	RW	V	--	Pre-Op Op

ADAM-E5053S (32-ch Isolated Digital Input Module)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
217Fh	01h	Module No	0-7	USINT	RW	V	--	Pre-Op Op
2180h- 2183h	01h	Get Current Value	0-255	USINT	RO	V	--	Pre-Op Op
2180h- 2183h	02h	Invert	0-255	USINT	RW	V	--	Pre-Op Op
2180h	03h	Digital Filter Scale (Low)	--	USINT	RW	V	--	Pre-Op Op
2180h	04h	Digital Filter Scale (High)	--	USINT	RW	V	--	Pre-Op Op

ADAM-E5056S/SO (16-ch Isolated Digital Output Module with LED)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
217Fh	01h	Module No	0-7	USINT	RW	V	--	Pre-Op Op
2180h- 2181h	01h	Get Current Value	0-255	USINT	RO	V	--	Pre-Op Op
2180h- 2181h	02h	FSV	0-255	USINT	RW	V	--	Pre-Op Op

ADAM-E5057S (16-ch Isolated Digital Output Module with LED)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
217Fh	01h	Module No	0-7	USINT	RW	V	--	Pre-Op Op
2180h-2183h	01h	Get Current Value	0-255	USINT	RO	V	--	Pre-Op Op
2180h-2183h	02h	FSV	0-255	USINT	RW	V	--	Pre-Op Op

ADAM-E5069 (8-ch Relay Output Module)								
Index	**Sub-Index**	**Name**	**Range**	**Data Type**	**Access**	**SDO**	**PDO**	**State**
217Fh	01h	Module No	0-7	USINT	RW	V	--	Pre-Op Op
2180h	01h	Get Current Value	0-255	USINT	RO	V	--	Pre-Op Op
2180h	02h	FSV	0-255	USINT	RW	V	--	Pre-Op Op

5.1.2.4.1 ADAM-E5051S 16-ch Isolated Digital Input Module with LED

The ADAM-E5051S provides 16 isolated digital input channels for critical environments need individual channel isolating protection.

Get Current Value by PDO

There are 2 ports in ADAM-E5051S and each port consists of 8 channels. Specify the port which need to diagnosis in the left-hand branch of TwinCAT System Manager.

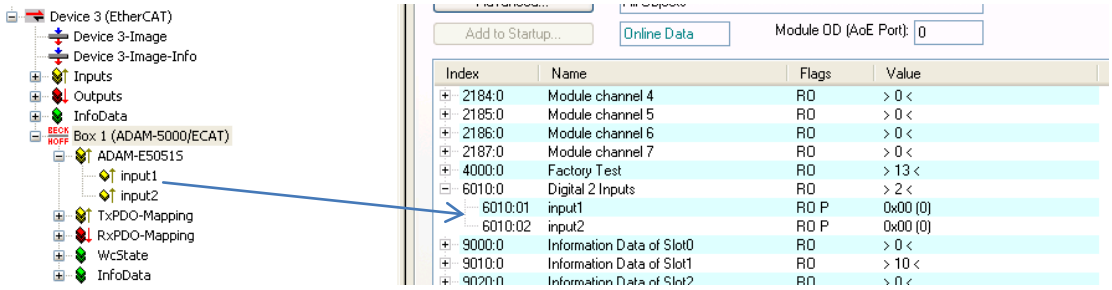
This index (sub-index) shows the raw data of each port of ADAM-E5051S module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
60x0h	01h – 02h	Input 1 – Input 2	0-255	USINT	RO	--	Tx	Op

The x of index is a number which represent the slot number the ADAM-E5051S plugs in. For example, if you need to get the value of ADAM-E5051S DI module which plugs in the ***second*** slot of ADAM-5000/ECAT, the x will be set to 1.

A bit of this value corresponds to a channel, the MSB (Most Significant Bit) of the ***first*** port (Index: 60x0h:01h) corresponds to the channel 7 (DI 7) and the LSB (Least Significant Bit) corresponds to the channel 0 (DI 0). The MSB (Most Significant Bit) of the ***second*** port (Index: 60x0h:02h) corresponds to the channel 15 (DI 15) and the LSB (Least Significant Bit) corresponds to the channel 8 (DI 8).

The value of bit 1 represents the input signal is in HIGH level, and 0 means the signal is in LOW level.



Setting by SDO

Select Slot of Module

Select the slot of module which you want to diagnosis or change the setting. For 4-slot ADAM-5000/ECAT, the first (leftmost) slot number is 00h and the last (rightmost) slot number is 03h.

For example, if you need to modify the setting of ADAM-E5051S DI module which plugs in the ***second*** slot of ADAM-5000/ECAT, the Module No (Index: 217Fh:01h) will be set to 01h.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
217Fh	01h	Module No	0-7	USINT	RW	V	--	Pre-Op Op

Get Current Value by SDO (NOT Recommend)

The current value can be reached by PDO and it will also show in following object index. There are 2 ports in ADAM-E5051S and each port consists of 8 channels. Index 2180h and its sub-index are used for the setting of first port; index 2181h and its sub-index are used for the setting of second port.

This index (sub-index) shows the raw data of each port of ADAM-E5051S module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h- 2181h	01h	SubIndex 001	0-255	USINT	RO	V	--	Pre-Op Op

A bit of this value corresponds to a channel, the MSB (Most Significant Bit) of the ***first*** port (Index: 2180h:01h) corresponds to the channel 7 (DI 7) and the LSB (Least Significant Bit) corresponds to the channel 0 (DI 0). The MSB (Most Significant Bit) of the ***second*** port (Index: 2181h:01h) corresponds to the channel 15 (DI 15) and the LSB (Least Significant Bit) corresponds to the channel 8 (DI 8).

The value of bit 1 represents the input signal is in HIGH level, and 0 means the signal is in LOW level.

Invert

This index (sub-index) shows the invert status of each port of ADAM-E5051S module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h- 2181h	02h	SubIndex 002	0-255	USINT	RW	V	--	Pre-Op Op

A bit of this value corresponds to a channel, the MSB (Most Significant Bit) corresponds to the channel 7 and the LSB (Least Significant Bit) corresponds to the channel 0. To invert a digital signal of an input channel, the bit value is set to be 1 and, on the contrary, 0 for hold the input signal.
For example, if you need to invert the signal of channel 8-11 of ADAM-E5051S DI module, the value of second port (Index: 2181h:02h) must be set to 0Fh (00001111).

Digital Filter Scale (Minimum Low Signal Width)

This index (sub-index) shows the minimum low signal width of ADAM-E5051S module. The digital input filter is a feature that eliminates noise from input signals. Once the digital input filter is set, it stores data during each sampling and then compares the data of the input terminal state. If all the input signals reach the LOW level within the filtering time, use the state as the input terminal value. If not, use the previous value.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h	03h	SubIndex 003	0-65535	USINT	RW	V	--	Pre-Op Op

The range of the digital filter time is 10 to 65535 in minimum second. To disable the filter, the value of this index must be set to 0.

Digital Filter Scale (Maximum High Signal Width)

This index (sub-index) shows the maximum high signal width of ADAM-E5051S module. The digital input filter is a feature that eliminates noise from input signals. Once the digital input filter is set, it stores data during each sampling and then compares the data of the input terminal state. If all the input signals reach the HIGH level within the filtering time, use the state as the input terminal value. If not, use the previous value.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h	04h	SubIndex 004	0-65535	USINT	RW	V	--	Pre-Op Op

The range of the digital filter time is 10 to 65535 in minimum second. To disable the filter, the value of this index must be set to 0.

5.1.2.4.2 ADAM-E5053S 32-ch Isolated Digital Input Module

The ADAM-E5053S provides 32 isolated digital input channels for critical environments need individual channel isolating protection.

Get Current Value by PDO

There are 4 ports in ADAM-E5053S and each port consists of 8 channels. Specify the port which need to diagnosis in the left-hand branch of TwinCAT System Manager.

This index (sub-index) shows the raw data of each port of ADAM-E5053S module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
60x0h	01h – 04h	Input 1 – Input 4	0-255	USINT	RO	--	Tx	Op

The x of index is a number which represent the slot number the ADAM-E5053S plugs in. For example, if you need to get the value of ADAM-E5053S DI module which plugs in the ***second*** slot of ADAM-5000/ECAT, the x will be set to 1.

A bit of this value corresponds to a channel, the MSB (Most Significant Bit) of the ***first*** port (Index: 60x0h:01h) corresponds to the channel 7 (DI 7) and the LSB (Least Significant Bit) corresponds to the channel 0 (DI 0). The MSB (Most Significant Bit) of the ***last*** port (Index: 60x0h:04h) corresponds to the channel 31 (DI 31) and the LSB (Least Significant Bit) corresponds to the channel 24 (DI 24).

The value of bit 1 represents the input signal is in HIGH level, and 0 means the signal is in LOW level.

Setting by SDO

Select Slot of Module

Select the slot of module which you want to diagnosis or change the setting. For 4-slot ADAM-5000/ECAT, the first (leftmost) slot number is 00h and the last (rightmost) slot number is 03h.

For example, if you need to modify the setting of ADAM-E5053S DI module which plugs in the ***second*** slot of ADAM-5000/ECAT, the Module No (Index: 217Fh:01h) will be set to 01h.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
217Fh	01h	Module No	0-7	USINT	RW	V	--	Pre-Op Op

Get Current Value by SDO (NOT Recommend)

The current value can be reached by PDO and it will also show in following object index. There are 4 ports in ADAM-E5053S and each port consists of 8 channels. Index 2180h and its sub-index are used for the setting of first port; index 2181h and its sub-index are used for the setting of second port and so on.

This index (sub-index) shows the raw data of each port of ADAM-E5051S module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h-2183h	01h	SubIndex 001	0-255	USINT	RO	V	--	Pre-Op Op

A bit of this value corresponds to a channel, the MSB (Most Significant Bit) of the ***first*** port (Index: 2180h:01h) corresponds to the channel 7 (DI 7) and the LSB (Least Significant Bit) corresponds to the channel 0 (DI 0). The MSB (Most Significant Bit) of the ***last*** port (Index: 2183h:01h) corresponds to the channel 31 (DI 31) and the LSB (Least Significant Bit) corresponds to the channel 24 (DI 24).

The value of bit 1 represents the input signal is in HIGH level, and 0 means the signal is in LOW level.

Invert

This index (sub-index) shows the invert status of each port of ADAM-E5053S module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h-2183h	02h	SubIndex 002	0-255	USINT	RW	V	--	Pre-Op Op

A bit of this value corresponds to a channel, the MSB (Most Significant Bit) corresponds to the channel 7 and the LSB (Least Significant Bit) corresponds to the channel 0. To invert a digital signal of an input channel, the bit value is set to be 1 and, on the contrary, 0 for hold the input signal.
For example, if you need to invert the signal of channel 8-11 of ADAM-E5053S DI module, the value of second port (Index: 2181h:02h) must be set to 0Fh (00001111).

Digital Filter Scale (Minimum Low Signal Width)

This index (sub-index) shows the minimum low signal width of ADAM-E5053S module. The digital input filter is a feature that eliminates noise from input signals. Once the digital input filter is set, it stores data during each sampling and then compares the data of the input terminal state. If all the input signals reach the LOW level within the filtering time, use the state as the input terminal value. If not, use the previous value.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h	03h	SubIndex 003	--	USINT	RW	V	--	Pre-Op Op

The range of the digital filter time is 10 to 65535 in minimum second. To disable the filter, the value of this index must be set to 0.

Digital Filter Scale (Maximum High Signal Width)

This index (sub-index) shows the maximum high signal width of ADAM-E5053S module. The digital input filter is a feature that eliminates noise from input signals. Once the digital input filter is set, it stores data during each sampling and then compares the data of the input terminal state. If all the input signals reach the HIGH level within the filtering time, use the state as the input terminal value. If not, use the previous value.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h	04h	SubIndex 004	--	USINT	RW	V	--	Pre-Op Op

The range of the digital filter time is 10 to 65535 in minimum second. To disable the filter, the value of this index must be set to 0.

5.1.2.4.3 ADAM-E5056S/SO 16-ch Isolated Digital Output Module with LED

The ADAM-E5056S/SO provides 16 isolated digital output channels for critical environments need individual channel isolating protection.

Set Current Value by PDO

There are 2 ports in ADAM-E5056S/SO and each port consists of 8 channels. Specify the port which need to diagnosis in the left-hand branch of TwinCAT System Manager.

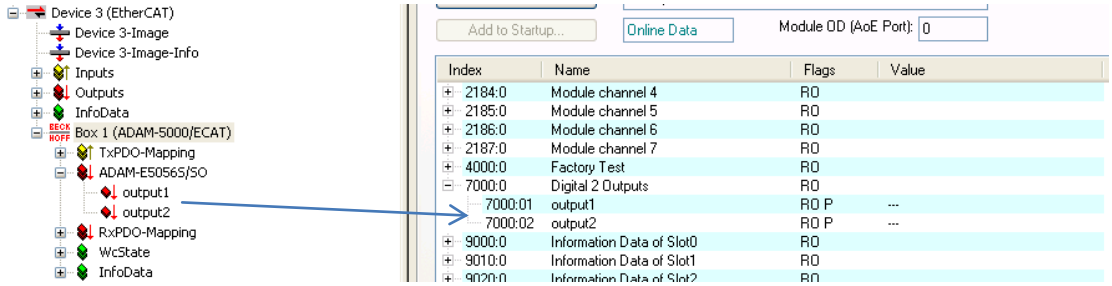
This index (sub-index) shows the raw data of each port of ADAM-E5056S/SO module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
70x0h	01h – 02h	Output 1 – Output 2	0-255	USINT	WO	--	Rx	Op

The x of index is a number which represent the slot number the ADAM-E5056S/SO plugs in. For example, if you need to get the value of ADAM-E5056S/SO DO module which plugs in the ***second*** slot of ADAM-5000/ECAT, the x will be set to 1.

A bit of this value corresponds to a channel, the MSB (Most Significant Bit) of the ***first*** port (Index: 70x0h:01h) corresponds to the channel 7 (DI 7) and the LSB (Least Significant Bit) corresponds to the channel 0 (DI 0). The MSB (Most Significant Bit) of the ***second*** port (Index: 70x0h:02h) corresponds to the channel 15 (DO 15) and the LSB (Least Significant Bit) corresponds to the channel 8 (DO 8).

The value of bit 1 represents the output signal is in HIGH level, and 0 means the signal is in LOW level.



Setting by SDO

Select Slot of Module

Select the slot of module which you want to diagnosis or change the setting. For 4-slot ADAM-5000/ECAT, the first (leftmost) slot number is 00h and the last (rightmost) slot number is 03h.

For example, if you need to modify the setting of ADAM-E5056S/SO DO module which plugs in the ***second*** slot of ADAM-5000/ECAT, the Module No (Index: 217Fh:01h) will be set to 01h.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
217Fh	01h	Module No	0-7	USINT	RW	V	--	Pre-Op Op

Get Current Value Setting by SDO

There are 2 ports in ADAM-E5056S/SO and each port consists of 8 channels. Index 2180h and its sub-index are used for the setting of first port; index 2181h and its sub-index are used for the setting of second port.
This index (sub-index) shows the raw data of each port of ADAM-E5056S/SO module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h-2181h	01h	SubIndex 001	0-255	USINT	RO	V	--	Pre-Op Op

A bit of this value corresponds to a channel, the MSB (Most Significant Bit) of the ***first*** port (Index: 2180h:01h) corresponds to the channel 7 (DO 7) and the LSB (Least Significant Bit) corresponds to the channel 0 (DO 0). The MSB (Most Significant Bit) of the ***second*** port (Index: 2181h:01h) corresponds to the channel 15 (DO 15) and the LSB (Least Significant Bit) corresponds to the channel 8 (DO 8).
The value of bit 1 represents the output signal is in HIGH level, and 0 means the signal is in LOW level.

FSV (The Fail Safe Value of communication WDT)

This index (sub-index) shows the FSV status of each port of ADAM-E5056S/SO module. Each DO will switch to its FSV if the module's WDT is enabled and it gets triggered.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h-2181h	02h	SubIndex 002	0-255	USINT	RW	V	--	Pre-Op Op

A bit of this value corresponds to a channel, the MSB (Most Significant Bit) corresponds to the channel 7 and the LSB (Least Significant Bit) corresponds to the channel 0. To set the output channel to logic high state when fail occurred, the bit value is set to be 1 and, on the contrary, 0 for disable this function.
For example, if you need to set the output channel 8-11 of ADAM-E5056S/SO DO module to logic high state when fail occurred, the value of second port (Index: 2181h:02h) must be set to 0Fh (00001111).

5.1.2.4.4 ADAM-E5057S 32-ch Isolated Digital Output Module

The ADAM-E5057S provides 32 isolated digital output channels for critical environments need individual channel isolating protection.

Set Current Value by PDO

There are 4 ports in ADAM-E5057S and each port consists of 8 channels. Specify the port which need to diagnosis in the left-hand branch of TwinCAT System Manager.

This index (sub-index) shows the raw data of each port of ADAM-E5057S module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
70x0h	01h – 04h	Output 1 – Output 4	0-255	USINT	WO	--	Rx	Op

The x of index is a number which represent the slot number the ADAM-E5057S plugs in. For example, if you need to set the value of ADAM-E5057S DO module which plugs in the ***second*** slot of ADAM-5000/ECAT, the x will be set to 1.

A bit of this value corresponds to a channel, the MSB (Most Significant Bit) of the ***first*** port (Index: 70x0h:01h) corresponds to the channel 7 (DO 7) and the LSB (Least Significant Bit) corresponds to the channel 0 (DO0). The MSB (Most Significant Bit) of the ***last*** port (Index: 70x0h:04h) corresponds to the channel 31 (DO 31) and the LSB (Least Significant Bit) corresponds to the channel 24 (DO 24).

The value of bit 1 represents the output signal is in HIGH level, and 0 means the signal is in LOW level.

Setting by SDO

Select Slot of Module

Select the slot of module which you want to diagnosis or change the setting. For 4-slot ADAM-5000/ECAT, the first (leftmost) slot number is 00h and the last (rightmost) slot number is 03h.

For example, if you need to modify the setting of ADAM-E5057S DO module which plugs in the ***second*** slot of ADAM-5000/ECAT, the Module No (Index: 217Fh:01h) will be set to 01h.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
217Fh	01h	Module No	0-7	USINT	RW	V	--	Pre-Op Op

Get Current Value Setting by SDO

There are 4 ports in ADAM-E5057S and each port consists of 8 channels. Index 2180h and its sub-index are used for the setting of first port; index 2181h and its sub-index are used for the setting of second port and so on.

This index (sub-index) shows the raw data of each port of ADAM-E5057S module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h-2183h	01h	SubIndex 001	0-255	USINT	RO	V	--	Pre-Op Op

A bit of this value corresponds to a channel, the MSB (Most Significant Bit) of the ***first*** port (Index: 2180h:01h) corresponds to the channel 7 (DO 7) and the LSB (Least Significant Bit) corresponds to the channel 0 (DO 0). The MSB (Most Significant Bit) of the ***last*** port (Index: 2183h:01h) corresponds to the channel 31 (DO 31) and the LSB (Least Significant Bit) corresponds to the channel 24 (DO 24).

The value of bit 1 represents the output signal is in HIGH level, and 0 means the signal is in LOW level.

FSV (The Fail Safe Value of communication WDT)

This index (sub-index) shows the FSV status of each port of ADAM-E5057S module. Each DO will switch to its FSV if the module's WDT is enabled and it gets triggered.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h-2183h	02h	SubIndex 002	0-255	USINT	RW	V	--	Pre-Op Op

A bit of this value corresponds to a channel, the MSB (Most Significant Bit) corresponds to the channel 7 and the LSB (Least Significant Bit) corresponds to the channel 0. To set the output channel to logic high state when fail occurred, the bit value is set to be 1 and, on the contrary, 0 for disable this function.
For example, if you need to set the output channel 8-11 of ADAM-E5057S DO module to logic high state when fail occurred, the value of second port (Index: 2181h:02h) must be set to 0Fh (00001111).

5.1.2.5 Relay Output Modules

5.1.2.5.1 ADAM-E5069 Relay Output Module

The ADAM-E5069 relay output module provides 8 relay channels of Form A. Switches can be used to control the relays. Considered to user friendly, the ADAM-E5069 also built with LED indicator for status reading easily. And it also provides a choice to clear or keep output status when reset by adjusting a jumper.

Set Current Value by PDO

There are 1 ports in ADAM-E5069 and consists of 8 channels. Specify the port which need to diagnosis in the left-hand branch of TwinCAT System Manager.
This index (sub-index) shows the raw data of each port of ADAM-E5069 module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
70x0h	01h	Output 1	0-255	USINT	WO	--	Rx	Op

The x of index is a number which represent the slot number the ADAM-E5069 plugs in. For example, if you need to set the value of ADAM-E5069 Relay module which plugs in the ***second*** slot of ADAM-5000/ECAT, the x will be set to 1.
A bit of this value corresponds to a channel, the MSB (Most Significant Bit) of the port (Index: 70x0h:01h) corresponds to the channel 7 (relay output 7) and the LSB (Least Significant Bit) corresponds to the channel 0 (relay output 0).
The value of bit 1 represents the output signal is in HIGH level, and 0 means the signal is in LOW level.

Setting by SDO

Select Slot of Module

Select the slot of module which you want to diagnosis or change the setting. For 4-slot ADAM-5000/ECAT, the first (leftmost) slot number is 00h and the last (rightmost) slot number is 03h.

For example, if you need to modify the setting of ADAM-E5069 Relay module which plugs in the ***second*** slot of ADAM-5000/ECAT, the Module No (Index: 217Fh:01h) will be set to 01h.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
217Fh	01h	Module No	0-7	USINT	RW	V	--	Pre-Op Op

Get Current Value by SDO

There is 1 port in ADAM-E5069 and consists of 8 channels.

This index (sub-index) shows the raw data of ADAM-E5069 module.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h	01h	SubIndex 001	0-255	USINT	RO	V	--	Pre-Op Op

A bit of this value corresponds to a channel, the MSB (Most Significant Bit) of the port (Index: 2180h:01h) corresponds to the channel 7 (relay output 7) and the LSB (Least Significant Bit) corresponds to the channel 0 (relay output 0).

The value of bit 1 represents the output signal is in HIGH level, and 0 means the signal is in LOW level.

FSV (The Fail Safe Value of communication WDT)

This index (sub-index) shows the FSV status of each port of ADAM-E5069 module. Each relay output will switch to its FSV if the module's WDT is enabled and it gets triggered.

Index	Sub-Index	Name	Range	Data Type	Access	SDO	PDO	State
2180h	02h	SubIndex 002	0-255	USINT	RW	V	--	Pre-Op Op

A bit of this value corresponds to a channel, the MSB (Most Significant Bit) corresponds to the channel 7 and the LSB (Least Significant Bit) corresponds to the channel 0. To set the relay output channel to logic high state when fail occurred, the bit value is set to be 1 and, on the contrary, 0 for disable this function.

Chapter 6

Programming Guide

6.1 Introduction

After completing the system configuration, you can begin to plan the application program. This chapter introduces programming tools Common Motion API definitions and how to use them for users to operate ADAM-5000/ECAT high speed system.

According to this Architecture, all of functions and properties have been classified three types: **System, Property** and **IO**.

6.1.1 ADAM-5000/ECAT Support API List

Type	Function	Description
System	Acm_GetAvailableDevs	Get the list of available device numbers and device names
	Acm_DevOpen	Open a specified device to get device handle
	Acm_DevClose	Close a device
	Acm_LoadENI	Load ENI (EtherCAT network information) file
	Acm_LoadMapFile	Load I/O mapping file in the EtherCAT network
	Acm_DevReOpen	Reopen Device
Property	Acm_GetProperty	Get the property value through assigned PropertyID
	Acm_GetU32Property	Get the property value belonging to unsigned 32 bit integer type
	Acm_GetI32Property	Get the property value belonging to signed 32 bit integer type
	Acm_GetF64Property	Get the property value belonging to double type
	Acm_SetProperty	Set the property value through assigned PropertyID.
	Acm_SetU32Property	Set the property value belonging to unsigned 32 bit integer type.
	Acm_SetI32Property	Set the property value belonging to signed 32 bit integer type
	Acm_SetF64Property	Set the property value belonging to double type
	Acm_GetChannelProperty	Get the DI/DO/AI/AO channel property value
	Acm_SetChannelProperty	Set the DI/DO/AI/AO channel property value
	Acm_GetMultiChannelProperty	Get the value continuous channels assigned by start channel ID and channel count
	Acm_SetMultiChannelProperty	Set the value continuous channels assigned by start channel ID and channel count
Communication State	Acm_DevGetComStatus	Get the communication network status
	Acm_DevGetSlaveStates	Get the slave device status

Event	Acm_DevEnableEvent	Enable the event check function
	Acm_DevCheckEvent	Check event
Master	Acm_DevGetSlaveInfo	Get the slave information according to the rotate switch on device
	Acm_DevGetErrorTable	Get the error device ID list when disconnection occurs
DI	Acm_DaqDiGetBit	Get the bit data of specified DI channel
	Acm_DaqDiGetByte	Get the byte data of specified DI Port
	Acm_DaqDiGetBytes	Get the byte data of continuous DI Port
DO	Acm_DaqDoSetBit	Set the bit data of specified DO channel
	Acm_DaqDoSetByte	Set the byte data of specified DO port
	Acm_DaqDoSetBytes	Set the byte data of continuous DO Port
	Acm_DaqDoGetBit	Get the bit data of specified DO channel
	Acm_DaqDoGetByte	Get the byte data of specified DO port
	Acm_DaqDoGetBytes	Get the byte data of continuous DO Port
AI	Acm_DaqAiGetRawData	Get the binary value of an analog input channel
	Acm_DaqAiGetVoltData	Get the voltage value of an analog input channel when voltage inputs
	Acm_DaqAiGetCurrData	Get the current value of an analog input channel when current inputs
AO	Acm_DaqAoSetRawData	Set the binary value of an analog output channel
	Acm_DaqAoSetVoltData	Set the voltage output value of an specified analog output channel within the analog voltage output range
	Acm_DaqAoSetCurrData	Set the current output value of a specified analog output channel within the analog current output range
	Acm_DaqAoGetRawData	Get the binary output value of a specified analog output channel
	Acm_DaqAoGetVoltData	Get the voltage output value of a specified analog output channel within the analog voltage output range
	Acm_DaqAoGetCurrData	Get the current output value of a specified analog output channel within the analog current output range

6.1.2 I/O Mapping

There might be many ADAM-5000/ECAT slaves in EtherCAT network and various DI/DO/AI/AO modules in each slave. Mapping information is important for user to configure and operate these IO modules. The mapping table can be separated into four parts: DI mapping, DO mapping, AI mapping, AO mapping.

- **DI mapping:**

  DI mapping sorts all DI by 8-bit per port in the entire network.

  For example:

  Slave1: 32 DI channels (8 bits per port, total 4 ports)

  Slave2: 16 DI channels (8 bits per port, total 2 ports)

  There are totally 6 ports for 48 DI channels, Port0, Port1, Port2, Port3, Port4 and Port5. User needs to know the port number of specified DI before access it.

  If user wants to access DI7~DI15 in slave2, he must pass port number=3.

  Port number of any DI port can be defined by user.

DI

Slave ID	Description	Slot	Byte Size	Port No
0x0200	DI0 ~ DI7 on ADAM-E5051S	0	1	000
0x0200	DI8 ~ DI15 on ADAM-E5051S	0	1	001
0x0100	DI0 ~ DI7 on ADAM-E5053S	1	1	002
0x0100	DI8 ~ DI15 on ADAM-E5053S	1	1	003
0x0100	DI16 ~ DI23 on ADAM-E5053S	1	1	004

- **DO mapping:**

  DO mapping sorts all DO by 8-bit per port in the entire network.

  For example:

  Slave1: 32 DO channels (8 bits per port, total 4 ports)

  Slave2: 16 DO channels (8 bits per port, total 2 ports)

  There are totally 6 ports for 48 DO channels, Port0, Port1, Port2, Port3, Port4 and Port5. User needs to know the port number of specified DO before access it.

  If user wants to access DO7~DO15 in slave2, he must pass port number=3.

  Port number of any DO port can be defined by user.

Slave ID	Description	Slot	Byte Size	Port No
0x0200	DO0 ~ DO7 on ADAM-E5056S/SO	2	1	000
0x0200	DO8 ~ DO15 on ADAM-E5056...	2	1	001
0x0100	DO0 ~ DO7 on ADAM-E5057S	0	1	002
0x0100	DO8 ~ DO15 on ADAM-E5057S	0	1	003
0x0100	DO16 ~ DO23 on ADAM-E5057S	0	1	004

- **AI mapping：**

  AI mapping sorts all AI channels in the entire network.

  For example:

  Slave1: 8 AI channels

  Slave2: 16 AI channels

  Each AI channel maps to one channelID for total 24 channels

  User needs to know the channelID of specified AI before access it.

  If user wants to access the first AI in slave2, he must pass ChannelID=8.

  ChannelID of any AI channel can be defined by user.

AI

Slave ID	Description	Slot	Byte Size	Channel No
0x0200	AI Channel 5 on ADAM-E5017	3	2	005
0x0200	AI Channel 6 on ADAM-E5017	3	2	006
0x0200	AI Channel 7 on ADAM-E5017	3	2	007
0x0100	AI Channel 0 on ADAM-E5017	2	2	008
0x0100	AI Channel 1 on ADAM-E5017	2	2	009

- **AO mapping：**

  AI mapping sorts all AI channels in the entire network.

  For example:

  Slave1: 4 AO channels

  Slave2: 4 AO channels

  Each AI channel maps to one channelID for total 24 channels

  User needs to know the channelID of specified AI before access it.

  If user wants to access the first AI in slave2, he must pass ChannelID=8.

  ChannelID of any AI channel can be defined by user.

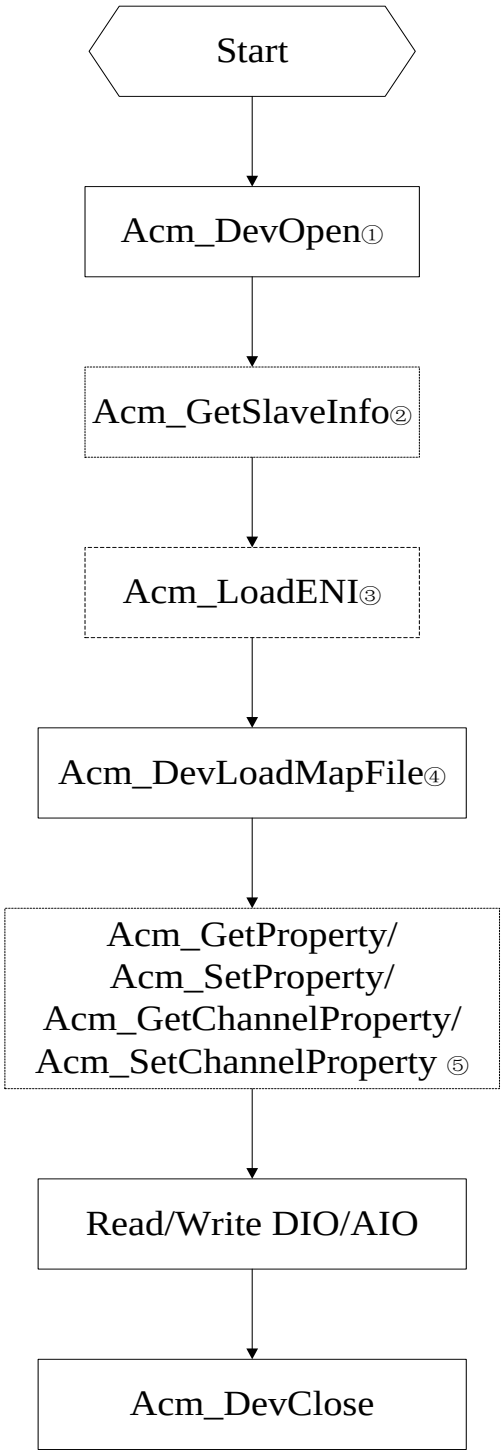
AO

Slave ID	Description	Slot	Byte Size	Channel No
0x0200	AO Channel 2 on ADAM-E5024	1	2	002
0x0200	AO Channel 3 on ADAM-E5024	1	2	003
0x0100	AO Channel 0 on ADAM-E5024	3	2	004
0x0100	AO Channel 1 on ADAM-E5024	3	2	005
0x0100	AO Channel 2 on ADAM-E5024	3	2	006

Note! User can mapping all DI/DO/AI/AO in whole network through Utility

6.2 Flow Charts

6.2.1 Basic Flow



① DeviceNumber need to be passed to **Acm_DevOpen**. DeviceNumber can be acquired by **Acm_GetAvailableDevs.**

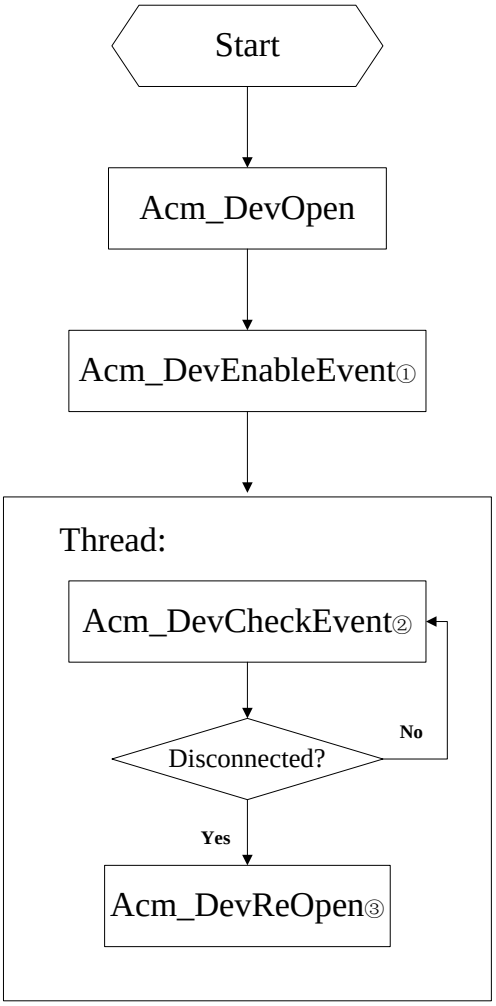
② User can check the slave information by **Acm_GetSlaveInfo**. (Not necessary)

③ User need to load ENI file before operating DI/O and AI/O. ENI file can be produced by Utility.

④ User can load mapping file produced by utility, it's not necessary if user not change the default mapping index of DI/O and AI/O, but user should know the mapping index of DI/O and AI/O by utility.

⑤ Properties of DI/DO/AI/AO can be set by **Acm_GetChannelProperty** / **Acm_SetChannelProperty**

6.2.2 Event



① User need to enable **Acm_DevEnableEvent** first before checking Event.

② Enable thread for Event and call **Acm_DevCheckEvent** cyclically in thread

③ User need to call **Acm_DevReOpen** to reopen device once Event occurs .But user must make sure the network reconnected before calling this API.

6.3 API Function

6.3.1 System

6.3.1.1 Acm_GetAvailableDevs

Format:

U32 Acm_GetAvailableDevs (DEVLIST *DeviceList, U32 MaxEntries, PU32 OutEntries)

Purpose:

Get the list of available device numbers and device names which are defined in Common Motion architecture.

Parameters:

Name	Type	IN/OUT	Description
DeviceList	DEVLIST *	**IN**	**Pointer to returned available device info list.**
MaxEntries	U32	**IN**	**Get the max devices count.**
OutEntries	PU32	**OUT**	**Return the count of available device.**

Return Value:

Error Code.

Comments:

The structure of **DEVLIST** is:

```
typedef struct tagPT_DEVLIST
{
    DWORD DeviceNum;
    CHAR DeviceName[50];
    SHORT NumOfSubDevices;
} DEVLIST, *LPDEVLIST;
```

DeviceNum:

Device Number needed for **Acm_DevOpen**.

DeviceName:

Device name. For example, “EtherCAT(0)”

NumOfSubDevices:

Display the count of connected slave modules

Example:

```
DEVLIST devLst[100];
```

```
U32 mstDevCnt = 0;
U32 slvDevNumber;
U32 deviceNumber;

errCde = Acm_GetAvailableDevs(devLst, 100, &mstDevCnt);
if(errCde == SUCCESS)
{
    mstDevCnt = 1;
    for(U32 i = 0; i<mstDevCnt; i++)
    {
        if(devLst[i].dwDeviceNum>>24==Adv_EtherCAT)
        {
            deviceNumber = devLst[i].dwDeviceNum;
            slvDevNumber = devLst[i].nNumOfSubdevices;
            break;
        }
    }
}
```

6.3.1.2 Acm_DevOpen

Format:

U32 Acm_DevOpen(U32 DeviceNumber, PHAND DeviceHandle)

Purpose:

Open a specified device to get device handle.

Parameters:

Name	Type	IN/OUT	Description
DeviceNumber	**U32**	**IN**	**Device Number. User can gets Device number by Acm_GetAvailableDevs**
DeviceHandle	**PHAND**	**OUT**	Return a pointer to the device handle

Return Value:

Error Code.

Comments:

This function should be called firstly before any operation of the device.

Please be noted that **Acm_DevClose** should be called at the end of the program after device open successfully.

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
```

6.3.1.3 Acm_DevClose

Format:

U32 Acm_DevClose(PHAND DeviceHandle)

Purpose:

Close a device.

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	PHAND	IN	A pointer to the device handle gotten by **Acm_DevOpen**

Return Value:

Error Code.

Comments:

Acm_DevClose should be called at the end of the program after device open successfully.

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
….
errCde = Acm_DevClose(&deviceHandle);
```

6.3.1.4 Acm_LoadENI

Format:

U32 Acm_LoadENI (HAND DeviceHandle, PI8 FilePath)

Purpose:

Load ENI (EtherCAT network information) file.

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	HAND	IN	The device handle gotten by **Acm_DevOpen**
FilePath	**PI8**	**IN**	**Pointer to a string that saves ENI file's path.**

Return Value:

Error Code.

Comments:

Before running ADAM-5000/ECAT, user needs to load ENI file after **Acm_DevOpen** Please refer to 2.1 Basic Flow). ENI file is configured by utility.

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      ……
 }
```

6.3.1.5 Acm_LoadMapFile

Format:

U32 Acm_DevLoadMapFile (HAND DeviceHandle, PI8 FilePath)

Purpose:

Load I/O mapping file in the EtherCAT network.

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	HAND	IN	The device handle gotten by **Acm_DevOpen**
FilePath	**PI8**	**IN**	**Pointer to a string that saves I/O mapping file's path.**

Return Value:

Error Code.

Comments:

Before calling this API function, user needs to call **Acm_DevOpen**, **Acm_LoadENI** first. Please refer to 2.1 Basic Flow). ENI file is configured by utility.

User also need to load I/O mapping file in their own program before controlling DI/O and AI/O. This file is configured by Utility and records all physical and logical mapping relation in whole EtherCAT network. If user just use the default I/O mapping relationship without any modification, use need not to load mapping file by this API, but user should know the I/O mapping relationship by utility.

All DI/O are arranged by “Port” (1 port=8 bits) in sequence. User can re-arrange the port number to map the real DI/O in ADAM-5000/ECAT by themselves.
The same, All AI/O arranged by “Channel” in sequence. User can re-arrange the channel ID to map the real DI/O in ADAM-5000/ECAT.
Please be noted that you should know all mapping relationship between PortNumber or ChannelID and physical DI/DO/AI/AO.

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
   errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
   if(errCde != SUCCESS)
        return ;
   errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
   if(errCde != SUCCESS)
        return ;
   ……
}
```

6.3.1.6 Acm_DevReOpen

Format:

U32 Acm_DevReOpen(HAND DeviceHandle)

Purpose:

Reopen Device

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	HAND	IN	The device handle gotten by **Acm_DevOpen**

Return Value:

Error Code.

Comments:

User needs to call this API to reopen the device in the following situation:

(1) Slave is re-connected after disconnection.

(2) Communication is not in OP mode.

(3) Any of slaves is not in OP mode.

Example:

Plese refer to the example of "Acm_DevCheckEvent"

6.3.2 Property

6.3.2.1 Acm_GetProperty

Format:

U32 Acm_GetProperty(HAND Handle, U32 PropertyID, PVOID Buffer, PU32 BufferLength)

Purpose:

Get the property (feature property, configuration property or parameter property) value through assigned PropertyID.

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
ProperyID	**U32**	**IN**	**Property ID to query.**
Buffer	**PVOID**	**OUT**	**Return Property value.**
BufferLength	**PU32**	**IN/OUT**	**Buffer byte size for the property. This value must the same as the length of inquired property, or error will occur and return the actual size of the property in buffer length**

Return Value:

Error Code.

Comments:

User should pay attention on the data type and BufferLength to get the value of Property according to PropertyID.

If the BufferLength is not the correct size, the return value will be error code " **DataSizeNotCorrect** ".

In this case, Buffer will return the value with the size of the property in BufferLength.

About the detail information of PropertyID, please refer to **Acm_GetU32Property**, **Acm_GetI32Property**, **Acm_GetF64Property** in Property List.

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
U32 maxDICnt, datalen;
```

```
errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
      if(errCde != SUCCESS)
           return ;

 datalen = sizeof(ULONG);
    errCde = Acm_GetProperty(deviceHandle, FT_DaqDiMaxChan, & maxDICnt, &datalen);
    if(errCde != SUCCESS)
    ……
 }
```

6.3.2.2 Acm_GetU32Property

Format:

U32 Acm_GetU32Property (HAND Handle, U32 PropertyID, PU32 Value)

Purpose:

Get the property value belonging to unsigned 32 bit integer type.

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
ProperyID	**U32**	**IN**	**Property ID to query. Please be noted that the type of property value must be unsigned 32 bit integer**
Value	**PU32**	**OUT**	**Return Property value with unsigned 32 bit integer type**

Return Value:

Error Code.

Comments:

Please be noted that **Acm_GetU32Property** is used to get the property value with unsigned 32 bit integer type.

User can get the detail information about the property value type in Property List.

Please refer to **Acm_GetProperty**, **Acm_GetI32Property**,

Acm_GetF64Property in Property List.

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
U32 maxDICnt;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
      if(errCde != SUCCESS)
           return ;
 errCde = Acm_GetU32Property(deviceHandle, FT_DaqDiMaxChan, & maxDICnt);
    if(errCde != SUCCESS)
    ……
 }
```

6.3.2.3 Acm_GetI32Property

Format:

U32 Acm_GetI32Property (HAND Handle, U32 PropertyID, PI32 Value)

Purpose:

Get the property value belonging to signed 32 bit integer type.

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
ProperyID	**U32**	**IN**	**Property ID to query. Please be noted that the type of property value must be signed 32 bit integer**
Value	**PI32**	**OUT**	**Return Property value with signed 32 bit integer type**

Return Value:

Error Code.

Comments:

Please be noted that **Acm_GetI32Property** is used to get the property value with signed 32 bit integer type.

User can get the detail information about the property value type in Property List.

Please refer to **Acm_GetProperty**, **Acm_GetU32Property**, **Acm_GetF64Property** in Property List.

Example:

Please refer to the **Acm_GetU32Property** example.

6.3.2.4 Acm_GetF64Property

Format:

U32 Acm_GetF64Property(HAND Handle, U32 PropertyID, PF64 Value)

Purpose:

Get the property value belonging to double type.

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
ProperyID	**U32**	**IN**	**Property ID to query. Please be noted that the type of property value must be double**
Value	**PF64**	**OUT**	**Return Property value with double type**

Return Value:

Error Code.

Comments:

Please be noted that **Acm_GetF64Property** is used to get the property value with double type.

User can get the detail information about the property value type in Property List.

Please refer to **Acm_GetU32Property**, **Acm_GetI32Property**, in Property List.

Example:

Please refer to the **Acm_GetU32Property** example.

6.3.2.5 Acm_SetProperty

Format:

U32 Acm_SetProperty(HAND Handle, U32 PropertyID, PVOID Buffer, U32 BufferLength)

Purpose:

Set the property (configuration property or parameter property) value through assigned PropertyID.

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
ProperyID	**U32**	**IN**	**Property ID to set.**
Buffer	**PVOID**	**OUT**	**A pointer to assigned property value**
BufferLength	**PU32**	**IN**	**Buffer byte size for the property. This value must the same as the length of inquired property, or error will occur**

Return Value:

Error Code.

Comments:

User should pay attention that not all of properties in Property List can be set to new property value; only the writable properties can be reset property value.

If the BufferLength is not correct, the return value will be error code " DataSizeNotCorrect".

About the detail information of PropertyID, please refer to **Acm_SetU32Property**, **Acm_SetI32Property**, **Acm_SetF64Property** in Property List.

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
U32 cycleTime, datalen;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      cycleTime = 1;
 datalen = sizeof(ULONG);
```

```
    errCde = Acm_SetProperty(deviceHandle, CFG_MasCycleTime, &cycleTime, datalen);
    if(errCde != SUCCESS)
    ……
}
```

6.3.2.6 Acm_SetU32Property

Format:

U32 Acm_SetU32Property (HAND Handle, U32 PropertyID, U32 Value)

Purpose:

Set the property value belonging to unsigned 32 bit integer type.

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
ProperyID	**U32**	**IN**	**Property ID to set. Please be noted that the type of property value must be unsigned 32 bit integer**
Value	**U32**	**IN**	**Property value with unsigned 32 bit integer type**

Return Value:

Error Code.

Comments:

Please be noted that **Acm_SetU32Property** is used to get the property value with unsigned 32 bit integer type.

User can set the detail information about the property value type in Property List.

Please refer to **Acm_SetProperty**, **Acm_SetI32Property**, **Acm_SetF64Property** in Property List.

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
 errCde = Acm_SetU32Property(deviceHandle, CFG_MasCycleTime, 1);
 if(errCde != SUCCESS)
 ……
```

```
}
```

6.3.2.7 Acm_SetI32Property

Format:

U32 Acm_SetI32Property (HAND Handle, U32 ProperyID, I32 Value)

Purpose:

Set the property value belonging to signed 32 bit integer type.

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
ProperyID	**U32**	**IN**	**Property ID to set. Please be noted that the type of property value must be signed 32 bit integer**
Value	**PI32**	**OUT**	**Property value with signed 32 bit integer type**

Return Value:

Error Code.

Comments:

Please be noted that **Acm_SetI32Property** is used to get the property value with signed 32 bit integer type.

User can get the detail information about the property value type in Property List.

Please refer to **Acm_SetProperty**, **Acm_SetU32Property**, **Acm_SetF64Property** in Property List.

Example:

Please refer to the **Acm_SetU32Property** example.

6.3.2.8 Acm_SetF64Property

Format:

U32 Acm_SetF64Property (HAND Handle, U32 ProperyID, F64 Value)

Purpose:

Set the property value belonging to double type.

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
ProperyID	**U32**	**IN**	**Property ID to query. Please be noted that the type of property value must be double**
Value	**PF64**	**OUT**	**Property value with double type**

Return Value:

Error Code.

Comments:

Please be noted that **Acm_SetF64Property** is used to get the property value with double type.

User can get the detail information about the property value type in Property List.

Please refer to **Acm_SetProperty**, **Acm_SetU32Property**, **Acm_GetI32Property** in Property List.

Example:

Please refer to the **Acm_SetU32Property** example.

6.3.2.9 Acm_GetChannelProperty

Format:

U32 Acm_GetChannelProperty (HAND Handle, U32 ChannelID, U32 ProperyID, PF64 Value)

Purpose:

Get the DI/DO/AI/AO channel property value.

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
ChannelID	U32	IN	**DI/DO/AI/AO channel ID**
ProperyID	U32	IN	PropertyID
Value	PF64	OUT	Get property value

Return Value:

Error Code.

Comments:

If user wants to get the DI/DO property value by channel.

This channelID can be get by calculation:

ChannelID= Port number * 8 + Port Index

Take the second DO in Port 3 for an example, the ChannelID= 3*8+2=26.

User should check or set the Port number and ChannelID in whole network through Mapping Info in Utility before using this API to get the property.

User can get the detail information about the property value type in Property List.

Note!

User need call **Acm_LoadENI** and **Acm_DevLoadMapFile** (If user uses the default mapping relationship, this API need not to load first before getting Property value by **Acm_GetChannelProperty**

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
F64 DiInvert;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
      if(errCde != SUCCESS)
           return ;

    errCde = Acm_GetChannelProperty (deviceHandle, 0, CFG_CH_DaqDiInvertEnable, &
DiInvert);
    if(errCde != SUCCESS)
    ……
}
```

6.3.2.10 Acm_SetChannelProperty

Format:

U32 Acm_SetChannelProperty (HAND Handle, U32 ChannelID, U32 ProperyID, F64 Value)

Purpose:

Set the DI/DO/AI/AO channel property value.

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
ChannelID	U32	IN	**DI/DO/AI/AO channel ID**
ProperyID	U32	IN	PropertyID
Value	F64	IN	Set Property value

Return Value:

Error Code.

Comments:

If user wants to get the DI/DO property value by channel.

This channelID can be get by calculation:

ChannelID= Port number * 8 + Port Index

Take the second DO in Port 3 for an example, the ChannelID= 3*8+2=26.

User should check or set the Port number and ChannelID in whole network through Mapping Info in Utility before using this API to get the property.

User can get the detail information about the property value type in Property List.

Note!

User need call **Acm_LoadENI** and **Acm_DevLoadMapFile** (If user uses the default mapping relationship, this API need not to load) first before getting Property value by Acm_SetChannelProperty

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
      if(errCde != SUCCESS)
           return ;
```

```
    //Invert DI.
    errCde = Acm_SetChannelProperty (deviceHandle, 0, CFG_CH_DaqDiInvertEnable, 1.0);
    if(errCde != SUCCESS)
    ……
}
```

6.3.2.11 Acm_GetMultiChannelProperty

Format:

U32 Acm_GetMultiChannelProperty (HAND Handle, U32 ProperyID, U32 StartChID, U32 ChCount, PF64 ValueArray)

Purpose:

Get the value continuous channels assigned by start channel ID and channel count.

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
ProperyID	U32	IN	**PropertyID**
StartChID	U32	**IN**	**Start channelID.**
ChCount	U32	IN	**Channel count**
ValueArray	PF64	OUT	**Value array.**

Return Value:

Error Code.

Comments:

If user wants to get the DI/DO property value by channel.

This channelID can be get by calculation:

ChannelID= Port number * 8 + Port Index

Take the second DO in Port 3 for an example, the ChannelID= 3*8+2=26.

User should check or set the Port number and ChannelID in whole network through Mapping Info in Utility before using this API to get the property.

User can get the detail information about the property value type in Property List.

Note!

1. User need call **Acm_LoadENI** and **Acm_DevLoadMapFile**(If user uses the

default mapping relationship, this API need not to load) first before getting Property value by this API.

2. The **StartChID + ChCount** cannot greater than total channel count.

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
F64 diInverArray[10] ;
int i = 0;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
      if(errCde != SUCCESS)
           return ;
      ……
    errCde = Acm_GetMultiChannelProperty (deviceHandle, CFG_CH_DaqDiInvertEnable, 0, 10,
diInverArray);
    if (errCde != SUCCESS)
    ……
}
```

6.3.2.12 Acm_SetMultiChannelProperty

Format:

U32 Acm_SetMultiChannelProperty (HAND Handle, U32 ProperyID, U32 StartChID, U32 ChCount, PF64 ValueArray)

Purpose:

Set the value continuous channels assigned by start channel ID and channel count.

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
ProperyID	U32	IN	**PropertyID**

StartChID	U32	**IN**	**Start channelID.**
ChCount	U32	IN	**Channel count**
ValueArray	PF64	IN	**Value array**

Return Value:

Error Code.

Comments:

If user wants to set the DI/DO property value by channel.

This channelID can be get by calculation:

ChannelID= Port number * 8 + Port Index

Take the second DO in Port 3 for an example, the ChannelID= 3*8+2=26.

User should check or set the Port number and ChannelID in whole network through Mapping Info in Utility before using this API to get the property.

User can get the detail information about the property value type in Property List.

Note!

1. User need call **Acm_LoadENI** and **Acm_DevLoadMapFile**(If user uses the default mapping relationship, this API need not to load) first before setting Property value by this API.
2. The **StartChID + ChCount** cannot greater than total channel count.

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
F64 diInverArray[10] ;
int i = 0;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
      if(errCde != SUCCESS)
           return ;
      ……
```

```
    for(i = 0; i<10; i++)
    { diInverArray [i] = 1.0;}
    errCde = Acm_SetMultiChannelProperty (deviceHandle, CFG_CH_DaqDiInvertEnable, 0, 10,
diInverArray);
    if (errCde != SUCCESS)
    ……
}
```

6.3.3 Communication State

6.3.3.1 Acm_DevGetComStatus

Format:

U32 Acm_DevGetComStatus (HAND DeviceHandle, U16 RingNo, PU16 pStatus)

Purpose:

Get the communication network status

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
RingNo	U16	IN	**Reserved (It can be an arbitrary value)**
pStatus	PU16	OUT	**Return the communication status** **Bit : Status** **0 : STATE_INIT** **1 : STATE_PREOP** **2 : STATE_SAFEOP** **3 : STATE_OP** **4 : STATE_ACK_ERR**

Return Value:

Error Code.

Comments:

This API is used to check if all the slaves in network are in OP mode. Please try to call **Acm_DevReOpen** to re-open the network if receiving not OP mode value. Usually use this API in Timer to do cyclical check.

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
U16 comStatus = 0;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
    errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
    if(errCde != SUCCESS)
        return ;
    ……
```

```
    errCde = Acm_DevGetComStatus(deviceHandle, 0, &comStatus);
    if(errCde ==SUCCESS&&comStatus!=0x8)
        ……
    ……
}
```

6.3.3.2 Acm_DevGetSlaveStates

Format:

U32 Acm_DevGetSlaveStates (HAND DeviceHandle, U16 RingNo, U16 SlaveIP, PU16 pStatus)

Purpose:

Get the slave device status

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
RingNo	U16	IN	**Reserved (It can be an arbitrary value)**
SlaveIP	U16	IN	**Number of rotate switch on slave** **Range：0x001~0xfff**
pStatus	PU16	OUT	**Return the communication status** **Bit Status** **0 ： STATE_INIT** **1 ： STATE_PREOP** **2 ： STATE_SAFEOP** **3 ： STATE_OP** **4 ： STATE_ACK_ERR**

Return Value:

Error Code.

Comments:

This API is used to check the slave current status in network.

Please try to call **Acm_DevReOpen** to re-open the network if receiving not OP mode value.

State Value	Description
0x11	AEC_SLAVE_STATE_ACK_ERR + AEC_SLAVE_STATE_INIT

0x12	**AEC_SLAVE_STATE_ACK_ERR + AEC_SLAVE_STATE_PREOP**
0x14	**AEC_SLAVE_STATE_ACK_ERR + AEC_SLAVE_STATE_SAFEOP**
0x18	**AEC_SLAVE_STATE_ACK_ERR + AEC_SLAVE_STATE_OP**

Usually use this API in Timer to do cyclical check.

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
U16 slvStatus = 0;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      ……

  errCde = Acm_DevGetSlaveStates(masterHandle,  0, 0x001, &slvState)
 if(errCde ==SUCCESS&& slvState!=0x8)
      ……
   ……
}
```

6.3.4 Event

6.3.4.1 Acm_DevEnableEvent

Format:

U32 Acm_DevEnableEvent (HAND DeviceHandle, U32 MasEnableEvt)

Purpose:

Enable the event check function

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
MasEnableEvt	U32	IN	**Eanble Event type:** **Bit Status** **0 ： Disconnection** **1~31 ： Reserved**

Return Value:

Error Code.

Comments:

This API is used to enable the event check function. User can find the disconnection event occurs by **Acm_DevCheckEvent** when open the disconnection check function. Also, user can get the information about which slaves are disconnected by **Acm_DevGetErrorTable**.

Note!

Check event need to open **threads** to call **Acm_DevCheckEvent.**

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
U32 slvNumber;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
    Acm_DevEnableEvent(deviceHandle, 1);
     ……
    pThreadObject = AfxBeginThread( (AFX_THREADPROC)CEventDlg::CheckEvtThread, this,
    THREAD_PRIORITY_TIME_CRITICAL, 0, 0, NULL );
```

```
    ……
}

UINT CEventDlg::CheckEvtThread(LPVOID ThreadArg)
{
     U32 Result = 0, evtStatus;
     CEventDlg*pThreadInfo = (CEventDlg*)ThreadArg;
     U32 errTable[100], errCnt = 0;
     while (pThreadInfo->m_bInit)
     {
          Result = Acm_DevCheckEvent (pThreadInfo-> deviceHandle, &evtStatus , INFINITE);
          if (Result ==SUCCESS)
          {
               if(evtStatus&0x01)
               {
                 Result  = Acm_DevGetErrorTable(pThreadInfo-> deviceHandle, 0,  errTable,
            &errCnt);
                        If(Result !=0)
                             …
                        Acm_DevReOpen(pThreadInfo-> deviceHandle);
               }
          }
     }
     return 0;
}
```

6.3.4.2 Acm_DevCheckEvent

Format:

U32 Acm_DevCheckEvent (HAND DeviceHandle, PU32 MasCheckEvt, U32 Millisecond)

Purpose:

Check event

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
MasCheckEvt	PU32	OUT	**Check enabled event status**

Millisecond	**U32**	**IN**	**Specify the time out value in millisecond for each checking (INFINITE as usual)**

Return Value:

Error Code.

Comments:

This API is used to check events status only for those event enabled by **Acm_DevEnableEvent**.

User can find the disconnection event occurs by **Acm_DevCheckEvent** when open the disconnection check function.

Also, user can get the information about which slaves are disconnected by **Acm_DevGetErrorTable**.

Note!

Check event need to open **threads** to call **Acm_DevCheckEvent.**

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
U32 slvNumber;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
    Acm_DevEnableEvent(deviceHandle, 1);
     ……
    pThreadObject = AfxBeginThread( (AFX_THREADPROC)CEventDlg::CheckEvtThread, this,
    THREAD_PRIORITY_TIME_CRITICAL, 0, 0, NULL );

    ……
 }

 UINT CEventDlg::CheckEvtThread(LPVOID ThreadArg)
 {
      U32 Result = 0, evtStatus;
      CEventDlg*pThreadInfo = (CEventDlg*)ThreadArg;
      U32 errTable[100], errCnt = 0;
```

```
    while (pThreadInfo->m_bInit)
    {
        Result = Acm_DevCheckEvent (pThreadInfo-> deviceHandle, &evtStatus , INFINITE);
        if (Result ==SUCCESS)
        {
            if(evtStatus&0x01)
            {
              Result  = Acm_DevGetErrorTable(pThreadInfo-> deviceHandle, 0,  errTable,
          &errCnt);
                If(Result !=0)
                    …
                Acm_DevReOpen(pThreadInfo-> deviceHandle);
            }
        }
    }
    return 0;
}
```

6.3.5 Master

6.3.5.1 Acm_DevGetSlaveInfo

Format:

U32 Acm_DevGetSlaveInfo(HAND DeviceHandle, U16 RingNo, U16 SlaveIP, PVOID pInfo)

Purpose:

Get the slave information according to the rotate switch on device

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
RingNo	U16	IN	Reserved **(It can be an arbitrary value)**
SlaveIP	U16	**IN**	**Number of rotate switch on slave** **Range：0x001~0xfff**
pInfo	PVOID	**OUT**	**Returned information from slave. It should be a pointer to the struct type below:** typedef struct _ADV_SLAVE_INFO {

```
    ULONG SlaveID;
    ULONG Position;
    ULONG VendorID;
    ULONG ProductID;
    ULONG RevisionNo;
    ULONG SerialNo;
    char Name[256];
    char FwVersion[8];
}ADV_SLAVE_INFO,*PADV_SLAVE_INFO;
```

Return Value:

Error Code.

Comments:

This API is used to help user to get the slave information through rotate switch.

Name	Description
SlaveID	Number of rotate switch in device. It must the same as SlaveIP
Position	Slave scanning position in network
VendorID	**Device vendor ID**
ProductID	**Device product ID**
RevisionNo	**Device revision number**
SerialNo	**Device serial number**
Name	**Device name**
FwVersion	**Firmware version**

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
U16 slvStatus = 0;
ADV_SLAVE_INFO slvInfo;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
    errCde = Acm_DevGetSlaveInfo (deviceHandle, 0, 0x001, &slvInfo);
```

```
    ......
}
```

6.3.5.2 Acm_DevGetErrorTable

Format:

U32 Acm_DevGetErrorTable(HAND DeviceHandle, U16 RingNo, PU32 ErrorTableArray, PU32 ArrayElements)

Purpose:

Get the error device ID list when disconnection occurs

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
RingNo	U16	IN	Reserved **(It can be an arbitrary value)**
ErrorTableArray	PU32	OUT	**Disconnected devices list. Please be noted that the length of list should be more than the number of disconnected devices**
ArrayElements	PU32	IN/OUT	**IN: Length of ErrorTableArray** **Out: number of disconnected device**

Return Value:

Error Code.

Comments:

Get the disconnected device list by this API. The internal error list will be clear after calling this API successfully. The error list will update when new disconnection occur. The error list will keep record all disconnected devices occurred at different time

Example:

Please refer to Acm_DevCheckEvent example.

6.3.6 Digital Input

6.3.6.1 Acm_DaqDiGetBit

Format:

U32 Acm_DaqDiGetBit (HAND DeviceHandle, U16 DiChannel, PU8 BitData)

Purpose:

Get the bit data of specified DI channel

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
DiChannel	U16	IN	**DI channel ID** **Range: 0~Max DI count -1**
BitData	PU8	**OUT**	**Return DI value**

Return Value:

Error Code.

Comments:

Get the DI value according to the specified Port Number and ChannelID

This channelID can be get by calculation:

ChannelID= Port number * 8 + Port Index

Take the second DO in Port 3 for an example, the ChannelID= 3*8+2=26.

User should check or set the Port number and ChannelID in whole network through Mapping Info in Utility before using this API to get the property.

The DI value will be affected by **CFG_CH_DaqDiInvertEnable**, **CFG_CH_DaqDiLowFilter**, **CFG_CH_DaqDiHighFilter**. Confirm that the property value is correct first. (Please refer to the Property chapter)

Note!

User need call **Acm_LoadENI** and **Acm_DevLoadMapFile**(If user uses the default mapping relationship, this API need not to load) first before calling **Acm_DaqDiGetBit.**

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
F64 DiInvert;
U16 chID_DI;
U8 bitValue = 0;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
```

```
        errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
        if(errCde != SUCCESS)
            return ;
        ……
        //Get the third DI channel of Port 1
        chID_DI = 1 * 8 + 2;
      errCde = Acm_GetChannelProperty (deviceHandle, chID_DI, CFG_CH_DaqDiInvertEnable, &
DiInvert);
      if(errCde !=SUCCESS)
        return;
      if(DiInvert >0)
      {
        DiInvert = 0;
          errCde = Acm_SetChannelProperty (deviceHandle, chID_DI, CFG_CH_DaqDiInvertEnable,
      DiInvert);
      }
      errCde = Acm_DaqDiGetBit(deviceHandle, chID_DI , &bitValue);
      ……
}
```

6.3.6.2 Acm_DaqDiGetByte

Format:

U32 Acm_DaqDiGetByte(HAND DeviceHandle, U16 DiPort, PU8 ByteData)

Purpose:

Get the byte data of specified DI Port

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
DiPort	U16	IN	**Port number of the specified 8 bit DI**
ByteData	PU8	OUT	**Return DI Port value**

Return Value:

Error Code.

Comments:

Get the DI value according to the specified Port Number.

User should check or set the Port number and ChannelID in whole network through

Mapping Info in Utility before using this API to get the property.

The DI value will be affected by **CFG_CH_DaqDiInvertEnable**, **CFG_CH_DaqDiLowFilter**, **CFG_CH_DaqDiHighFilter**. Confirm that the property value is correct first. (Please refer to the Property chapter)

Note!

1. DI port number must be consistent to the mapping info in Utility.
2. User need call **Acm_LoadENI** and **Acm_DevLoadMapFile** (If user uses the default mapping relationship, this API need not to load) first before calling **Acm_DaqDiGetBit.**

Example:

```
U U32 deviceNumber = 0x60000000;
HAND deviceHandle;
U8 byteValue = 0;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
      if(errCde != SUCCESS)
           return ;
      ……
      //Get value of port1
    errCde = Acm_DaqDiGetByte(deviceHandle, 1, & byteValue);
    ……
}
```

6.3.6.3 Acm_DaqDiGetBytes

Format:

U32 Acm_DaqDiGetBytes (HAND DeviceHandle, U16 StartPort, U16 NumPort, PU8 ByteDataArray)

Purpose:

Get the byte data of continuous DI Port

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
StartPort	U16	IN	**The start port number.**
NumPort	U16	IN	**Number of ports**
ByteDataArray	PU8	**OUT**	**Return DI value**

Return Value:

Error Code.

Comments:

Get the continuous DI byte value according to the specified Port Number.

User should check or set the Port number and ChannelID in whole network through Mapping Info in Utility before using this API to get the property.

The DI value will be affected by **CFG_CH_DaqDiInvertEnable**, **CFG_CH_DaqDiLowFilter**, **CFG_CH_DaqDiHighFilter**. Confirm that the property value is correct first. (Please refer to the Property chapter)

Note!

1. (StartPort + NumPort) must less than the total port number.
2. User need call **Acm_LoadENI** and **Acm_DevLoadMapFile** (If user uses the default mapping relationship, this API need not to load) first before calling **Acm_DaqDiGetBit.**

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
U8 byteValueArray[3];

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
      if(errCde != SUCCESS)
```

```
            return ;
        ……
        //Get value of port1, port2, port3;
    errCde = Acm_DaqDiGetBytes (deviceHandle, 1, 3, byteValueArray);
        ……
}
```

6.3.7 Digital Output

6.3.7.1 Acm_DaqDoSetBit

Format:

U32 Acm_DaqDoSetBit(HAND DeviceHandle, U16 DoChannel, U8 BitData)

Purpose:

Set the bit data of specified DO channel

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
DoChannel	U16	IN	**DO channel ID** **Range: 0~Max DO count -1**
BitData	U8	IN	**DO value (0 or 1)**

Return Value:

Error Code.

Comments:

Get the DO value according to the specified Port Number and ChannelID

This channelID can be get by calculation:

ChannelID= Port number * 8 + Port Index

Take the second DO in Port 3 for an example, the ChannelID= 3*8+2=26.

User should check or set the Port number and ChannelID in whole network through Mapping Info in Utility before using this API to get the property.

The DO value will be affected by **CFG_CH_DaqDoFsvEnable**. Confirm that the property value is correct first. (Please refer to the Property chapter)

Note!

User need call **Acm_LoadENI** and **Acm_DevLoadMapFile** (If user uses the default mapping relationship, this API need not to load) first before calling **Acm_DaqDoSetBit.**

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
U16 chID_DO;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
      if(errCde != SUCCESS)
           return ;
      ……
      //Set the third DO channel of Port 1
      chID_DO = 1 * 8 + 2;
    errCde = Acm_DaqDoSetBit (deviceHandle, chID_DO , 1);
    ……
}
```

6.3.7.2 Acm_DaqDoSetByte

Format:

U32 Acm_DaqDoSetByte(HAND DeviceHandle, U16 DoPort, U8 ByteData)

Purpose:

Set the byte data of specified DO port

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
DoPort	U16	IN	**DO port number**
ByteData	U8	IN	**DO value**

Return Value:

Error Code.

Comments:

Get the DO value according to the specified Port Number.

User should check or set the Port number and ChannelID in whole network through Mapping Info in Utility before using this API to get the property.

The DO value will be affected by **CFG_CH_DaqDoFsvEnable**. Confirm that the property value is correct first. (Please refer to the Property chapter)

Note!

User need call **Acm_LoadENI** and **Acm_DevLoadMapFile** (If user uses the default mapping relationship, this API need not to load) first before calling **Acm_DaqDoSetByte.**

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
      if(errCde != SUCCESS)
           return ;
      ……
      //set value of port1
    errCde = Acm_DaqDoSetByte (deviceHandle, 1,  0xff);
    ……
}
```

6.3.7.3 Acm_DaqDoSetBytes

Format:

U32 Acm_DaqDoSetBytes (HAND DeviceHandle, U16 StartPort, U16 NumPort, PU8 ByteDataArray)

Purpose:

Set the byte data of continuous DO Port

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
StartPort	U16	IN	**The start port number.**

NumPort	U16	IN	**Number of ports**
ByteDataArray	PU8	**IN**	**DO value array**

Return Value:

Error Code.

Comments:

Set the continuous DO byte value according to the specified Port Number.

User should check or set the Port number and ChannelID in whole network through Mapping Info in Utility before using this API to get the property.

The DO value will be affected by **CFG_CH_DaqDoFsvEnable**. Confirm that the property value is correct first. (Please refer to the Property chapter)

Note!

1. (StartPort + NumPort) must less than the total port number.
2. User need call **Acm_LoadENI** and **Acm_DevLoadMapFile** (If user uses the default mapping relationship, this API need not to load) first before calling **Acm_DaqDoSetBytes.**

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
U8 byteValueArray[3];

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
      if(errCde != SUCCESS)
           return ;
      ……
      //Set value of port1, port2, port3;
      byteValueArray[0] = 0x0f;
      byteValueArray[1] = 0xf0;
      byteValueArray[2] = 0xff;
    errCde = Acm_DaqDoSetBytes (deviceHandle, 1, 3, byteValueArray);
```

```
    ……
}
```

6.3.7.4 Acm_DaqDoGetBit

Format:

U32 Acm_DaqDoGetBit(HAND DeviceHandle, U16 DoChannel, PU8 BitData)

Purpose:

Get the bit data of specified DO channel

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
DoChannel	U16	IN	**DO channel ID** **Range: 0~Max DO count -1**
BitData	PU8	**OUT**	**Return DO value**

Return Value:

Error Code.

Comments:

Get the DO value according to the specified ChannelID

This channelID can be get by calculation:

ChannelID= Port number * 8 + Port Index

Take the second DO in Port 3 for an example, the ChannelID= 3*8+2=26.

User should check or set the Port number and ChannelID in whole network through Mapping Info in Utility before using this API to get the property.

The DO value will be affected by **CFG_CH_DaqDoFsvEnable**. Confirm that the property value is correct first. (Please refer to the Property chapter)

Note!

User need call **Acm_LoadENI** and **Acm_DevLoadMapFile** (If user uses the default mapping relationship, this API need not to load) first before calling **Acm_DaqDoGetBit.**

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
U16 chID_DO;
U8 bitValue = 0;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
```

```
{
    errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
    if(errCde != SUCCESS)
        return ;
    errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
    if(errCde != SUCCESS)
        return ;
    ……
    //Get the third DO channel of Port 1
    chID_DO = 1 * 8 + 2;
  errCde = Acm_DaqDoGetBit(deviceHandle, chID_DO , &bitValue);
  ……
}
```

6.3.7.5 Acm_DaqDoGetByte

Format:

U32 Acm_DaqDoGetByte (HAND DeviceHandle, U16 DoPort, PU8 ByteData)

Purpose:

Get the byte data of specified DO port

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
DoPort	U16	IN	**DO Port number** **Range: 0~Max DO count -1**
ByteData	PU8	**OUT**	**Return DO value**

Return Value:

Error Code.

Comments:

Get the DO value according to the specified Port number

User should check or set the Port number and ChannelID in whole network through Mapping Info in Utility before using this API to get the property.

The DO value will be affected by **CFG_CH_DaqDoFsvEnable**. Confirm that the property value is correct first. (Please refer to the Property chapter)

Note!

User need call **Acm_LoadENI** and **Acm_DevLoadMapFile** (If user uses the default

mapping relationship, this API need not to load) first before calling **Acm_DaqDoGetByte.**

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
U8 byteValue = 0;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
      if(errCde != SUCCESS)
           return ;
      ……
      //Get value of Port 1
    errCde = Acm_DaqDoGetByte(deviceHandle, 1 , & byteValue);
    ……
}
```

6.3.7.6 Acm_DaqDoGetBytes

Format:

U32 Acm_DaqDoGetBytes(HAND DeviceHandle, U16 StartPort, U16 NumPort, PU8 ByteDataArray)

Purpose:

Get the byte data of continuous DO Port

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
StartPort	U16	IN	**The start port number.**
NumPort	U16	IN	**Number of ports**
ByteDataArray	PU8	**IN**	**DO value array**

Return Value:

Error Code.

Comments:

Get the continuous DO byte value according to the specified Port Number.

User should check or set the Port number and ChannelID in whole network through Mapping Info in Utility before using this API to get the property.

The DO value will be affected by **CFG_CH_DaqDoFsvEnable**. Confirm that the property value is correct first. (Please refer to the Property chapter)

Note!

3. (StartPort + NumPort) must less than the total port number.
4. User need call **Acm_LoadENI** and **Acm_DevLoadMapFile** (If user uses the default mapping relationship, this API need not to load) first before calling **Acm_DaqDoGetBytes.**

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
U8 byteValueArray[3];

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
      if(errCde != SUCCESS)
           return ;
      ……
      //Get value of port1, port2, port3;
    errCde = Acm_DaqDoGetBytes (deviceHandle, 1, 3, byteValueArray);
    ……
}
```

6.3.8 Analog Input

6.3.8.1 Acm_DaqAiGetRawData

Format:

U32 Acm_DaqAiGetRawData(HAND DeviceHandle, U16 AiChannel, PU16 AiData)

Purpose:

Get the binary value of an analog input channel

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
AiChannel	U16	IN	**AI channel ID** **Range: 0~Max AI count -1**
AiData	PU16	OUT	**Pointer to the returned AI binary value**

Return Value:

Error Code.

Comments:

Get the AI binary value according to the specified ChannelID

User should check or set the Port number and ChannelID in whole network through Mapping Info in Utility before using this API to get the property.

The AI value will be affected by **CFG_CH_DaqAiRange**, **CFG_CH_DaqAiMask**, **CFG_CH_DaqAiIntegrationTime**. Confirm that the property value is correct first. (Please refer to the Property chapter)

Note!

User need call **Acm_LoadENI** and **Acm_DevLoadMapFile** (If user uses the default mapping relationship, this API need not to load) first before calling **Acm_DaqAiGetRawData.**

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
USHORT aiData = 0;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
```

```
        if(errCde != SUCCESS)
            return ;
        ……

        errCde = Acm_DaqAiGetRawData(deviceHandle, 0, &aiData);
    ……
}
```

6.3.8.2 Acm_DaqAiGetVoltData

Format:

U32 Acm_DaqAiGetVoltData(HAND DeviceHandle, U16 AiChannel, PF32 AiData)

Purpose:

Get the voltage value of an analog input channel when voltage inputs

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
AiChannel	U16	IN	**AI channel ID** **Range: 0~Max AI count -1**
AiData	PF32	OUT	**Pointer to the returned AI voltage value**

Return Value:

Error Code.

Comments:

Get the AI voltage value according to the specified ChannelID

User should check or set the Port number and ChannelID in whole network through mapping Info in Utility before using this API to get the property.

As the AI channel is set to voltage level by **CFG_CH_DaqAiRange**, user can gets the Input voltage value; Error will occur when calling this API as the AI channel is set to current level.

The AI value will be affected by **CFG_CH_DaqAiRange**, **CFG_CH_DaqAiMask**, **CFG_CH_DaqAiIntegrationTime**. Confirm that the property value is correct first. (Please refer to the Property chapter)

Note!

User need call **Acm_LoadENI** and **Acm_DevLoadMapFile** (If user uses the default mapping relationship, this API need not to load) first before calling

Acm_DaqAiGetVoltData.

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
F32 aiData = 0;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
      if(errCde != SUCCESS)
           return ;
      ……
      errCde = Acm_DaqAiGetVoltData(deviceHandle, 0, &aiData);
    ……
}
```

6.3.8.3 Acm_DaqAiGetCurrData

Format:

U32 Acm_DaqAiGetCurrData(HAND DeviceHandle, U16 AiChannel, PF32 AiData)

Purpose:

Get the current value of an analog input channel when current inputs

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
AiChannel	U16	IN	**AI channel ID** **Range: 0~Max AI count -1**
AiData	PF32	OUT	**Pointer to the returned AI current value**

Return Value:

Error Code.

Comments:

Get the AI current value according to the specified ChannelID

User should check or set the Port number and ChannelID in whole network through mapping Info in Utility before using this API to get the property.

As the AI channel is set to current level by **CFG_CH_DaqAiRange**, user can gets the Input current value; Error will occur when calling this API as the AI channel is set to voltage level.

The AI value will be affected by **CFG_CH_DaqAiRange**, **CFG_CH_DaqAiMask**, **CFG_CH_DaqAiIntegrationTime**. Confirm that the property value is correct first. (Please refer to the Property chapter)

Note!

User need call **Acm_LoadENI** and **Acm_DevLoadMapFile** (If user uses the default mapping relationship, this API need not to load) first before calling **Acm_DaqAiGetCurrData**.

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
F32 aiData = 0;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
      if(errCde != SUCCESS)
           return ;
      ……
      errCde = Acm_DaqAiGetCurrData (deviceHandle, 0, &aiData);
    ……
}
```

6.3.9 Analog Output

6.3.9.1 Acm_DaqAoSetRawData

Format:

U32 Acm_DaqAoSetRawData (HAND DeviceHandle, U16 AoChannel, U16 AoData)

Purpose:

Set the binary value of an analog output channel

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
AoChannel	U16	IN	**AO channel ID** **Range: 0~Max AO count -1**
AoData	U16	IN	**AO binary value**

Return Value:

Error Code.

Comments:

Set the AO binary value according to the specified ChannelID

User should check or set the Port number and ChannelID in whole network through Mapping Info in Utility before using this API to get the property.

The AO value will be affected by **CFG_CH_DaqAoRange**. Confirm that the property value is correct first. (Please refer to the Property chapter)

Note!

User need call **Acm_LoadENI** and **Acm_DevLoadMapFile** (If user uses the default mapping relationship, this API need not to load) first before calling **Acm_DaqAoSetRawData**.

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
      if(errCde != SUCCESS)
           return ;
      ……
```

```
    errCde = Acm_DaqAoSetRawData (deviceHandle, 0, 0);
    ……
}
```

6.3.9.2 Acm_DaqAoSetVoltData

Format:

U32 Acm_DaqAoSetVoltData (HAND DeviceHandle, U16 AoChannel, F32 AoData)

Purpose:

Set the voltage output value of an specified analog output channel within the analog voltage output range.

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
AoChannel	U16	IN	**AO channel ID** **Range: 0~Max AO count -1**
AoData	F32	IN	**AO voltage value**

Return Value:

Error Code.

Comments:

Set the AO voltage value according to the specified ChannelID

User should check or set the Port number and ChannelID in whole network through Mapping Info in Utility before using this API to get the property.

As the AO channel is set to voltage level by **CFG_CH_DaqAoRange**, user can sets the output voltage value; Error will occur when calling this API as the AO channel is set to current level.

The AO value will be affected by **CFG_CH_DaqAoRange**. Confirm that the property value is correct first. (Please refer to the Property chapter)

Note!

User need call **Acm_LoadENI** and **Acm_DevLoadMapFile**(If user uses the default mapping relationship, this API need not to load) first before calling **Acm_DaqAoSetVoltData**.

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
```

```
errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
      if(errCde != SUCCESS)
           return ;
      ……

      errCde = Acm_DaqAoSetVoltData (deviceHandle, 0, 10.0);
    ……
}
```

6.3.9.3 Acm_DaqAoSetCurrData

Format:

U32 Acm_DaqAoSetCurrData (HAND DeviceHandle, U16 AoChannel, F32 AoData)

Purpose:

Set the current output value of a specified analog output channel within the analog current output range.

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
AoChannel	U16	IN	**AO channel ID** **Range: 0~Max AO count -1**
AoData	F32	IN	**AO current value**

Return Value:

Error Code.

Comments:

Set the AO current value according to the specified ChannelID

User should check or set the Port number and ChannelID in whole network through Mapping Info in Utility before using this API to get the property.

As the AO channel is set to current level by **CFG_CH_DaqAoRange**, user can sets the output current value; Error will occur when calling this API as the AO channel is

set to voltage level.

The AO value will be affected by **CFG_CH_DaqAoRange**. Confirm that the property value is correct first. (Please refer to the Property chapter)

Note!

User need call **Acm_LoadENI** and **Acm_DevLoadMapFile** (If user uses the default mapping relationship, this API need not to load) first before calling **Acm_DaqAoSetCurrData**.

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
      if(errCde != SUCCESS)
           return ;
      ……

      errCde = Acm_DaqAoSetCurrData (deviceHandle, 0, 10.0);
    ……
}
```

6.3.9.4 Acm_DaqAoGetRawData

Format:

U32 Acm_DaqAoGetRawData (HAND DeviceHandle, U16 AoChannel, PU16 AoData)

Purpose:

Get the binary output value of a specified analog output channel.

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**

AoChannel	U16	IN	**AO channel ID** **Range: 0~Max AO count -1**
AoData	PU16	OUT	**Pointer to the AO binary value**

Return Value:

Error Code.

Comments:

Get the AO binary value according to the specified ChannelID

User should check or set the Port number and ChannelID in whole network through Mapping Info in Utility before using this API to get the property.

The AO value will be affected by **CFG_CH_DaqAoRange**. Confirm that the property value is correct first. (Please refer to the Property chapter)

Note!

User need call **Acm_LoadENI** and **Acm_DevLoadMapFile** (If user uses the default mapping relationship, this API need not to load) first before calling **Acm_DaqAoGetRawData**.

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
U16 aoData;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
      if(errCde != SUCCESS)
           return ;
      ……

      errCde = Acm_DaqAoGetRawData (deviceHandle, 0, & aoData);
    ……
}
```

6.3.9.5 Acm_DaqAoGetVoltData

Format:

U32 Acm_DaqAoGetVoltData (HAND DeviceHandle, U16 AoChannel, PF32 AoData)

Purpose:

Get the voltage output value of a specified analog output channel within the analog voltage output range.

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
AoChannel	U16	IN	**AO channel ID** **Range: 0~Max AO count -1**
AoData	PF32	OUT	**Pointer to the AO voltage value**

Return Value:

Error Code.

Comments:

Get the AO voltage value according to the specified ChannelID

User should check or set the Port number and ChannelID in whole network through Mapping Info in Utility before using this API to get the property.

As the AO channel is set to voltage level by **CFG_CH_DaqAoRange**, user can sets the output voltage value; Error will occur when calling this API as the AO channel is set to current level.

The AO value will be affected by **CFG_CH_DaqAoRange**. Confirm that the property value is correct first. (Please refer to the Property chapter)

Note!

User need call **Acm_LoadENI** and **Acm_DevLoadMapFile** (If user uses the default mapping relationship, this API need not to load) first before calling **DaqAoGetVoltData**.

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
F32 aoData;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
```

```
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
      if(errCde != SUCCESS)
           return ;
      ……

      errCde = Acm_DaqAoGetVoltData (deviceHandle, 0, & aoData);
    ……
}
```

6.3.9.6 Acm_DaqAoGetCurrData

Format:

U32 Acm_DaqAoGetCurrData (HAND DeviceHandle, U16 AoChannel, PF32 AoData)

Purpose:

Get the current output value of a specified analog output channel within the analog current output range.

Parameters:

Name	Type	IN/OUT	Description
DeviceHandle	**HAND**	**IN**	The device handle gotten by **Acm_DevOpen**
AoChannel	U16	IN	**AO channel ID** **Range: 0~Max AO count -1**
AoData	PF32	OUT	**Pointer to the AO current value**

Return Value:

Error Code.

Comments:

Get the AO current value according to the specified ChannelID

User should check or set the Port number and ChannelID in whole network through Mapping Info in Utility before using this API to get the property.

As the AO channel is set to current level by **CFG_CH_DaqAoRange**, user can sets the output current value; Error will occur when calling this API as the AO channel is set to voltage level.

The AO value will be affected by **CFG_CH_DaqAoRange**. Confirm that the property value is correct first. (Please refer to the Property chapter)

Note!

User need call **Acm_LoadENI** and **Acm_DevLoadMapFile** (If user uses the default mapping relationship, this API need not to load) first before calling **Acm_DaqAoGetCurrData**.

Example:

```
U32 deviceNumber = 0x60000000;
HAND deviceHandle;
F32 aoData;

errCde = Acm_DevOpen(deviceNumber , &deviceHandle);
if(errCde == Success)
{
      errCde = Acm_LoadENI(deviceHandle, "C: \\ENI.xml");
      if(errCde != SUCCESS)
           return ;
      errCde = Acm_DevLoadMapFile (deviceHandle, "C: \\Mapping.xml");
      if(errCde != SUCCESS)
           return ;
      ……

      errCde = Acm_DaqAoGetCurrData (deviceHandle, 0, & aoData);
    ……
 }
```

6.4 Property

6.4.1 Device Property

All properties can be read by

Acm_GetProperty

Acm_GetU32Property

Acm_GetI32Property

Acm_GetF64Property

and be set by

Acm_SetProperty

Acm_SetU32Property

Acm_SetI32Property

Acm_SetF64Property

6.4.1.1 Feature

FT_DaqDiMaxChan

PROPERTY ID	50
PROPERTY TYPE	U32
R/W	R
DESCRIPTION	**Get all DI channel count in the network.** If user uses the default mapping relationship, the **Acm_DevLoadMapFile** need not to load, or else user should download mapfile use **Acm_DevLoadMapFile** before get this property.

FT_DaqDoMaxChan

PROPERTY ID	51
PROPERTY TYPE	U32
R/W	R
DESCRIPTION	**Get all DO channel count in the network.** If user uses the default mapping relationship, the **Acm_DevLoadMapFile** need not to load, or else user should download mapfile use **Acm_DevLoadMapFile** before get this property.

FT_DaqAiMaxSingleChan

PROPERTY ID	54
PROPERTY TYPE	U32
R/W	R
DESCRIPTION	**Get all single-end AI channel count in the network.** If user uses the default mapping relationship, the **Acm_DevLoadMapFile** need not to load, or else user should download mapfile use **Acm_DevLoadMapFile** before get this property.

FT_DaqAiMaxDiffChan

PROPERTY ID	55
PROPERTY TYPE	U32
R/W	R
DESCRIPTION	**Get all differential AI channel count in the network.** If user uses the default mapping relationship, the **Acm_DevLoadMapFile** need not to load, or else user should download mapfile use **Acm_DevLoadMapFile** before get this property.

FT_DaqAoMaxChan

PROPERTY ID	57
PROPERTY TYPE	U32
R/W	R
DESCRIPTION	**Get all AO channel count in the network.** If user uses the default mapping relationship, the **Acm_DevLoadMapFile** need not to load, or else user should download mapfile use **Acm_DevLoadMapFile** before get this property.

6.4.1.2 Configuration

CFG_MasCycleTime

PROPERTY ID	261
PROPERTY TYPE	U32
R/W	R/W
DESCRIPTION	**Set/Get the cycle time for data communication.** **Range:1~100, uint: ms.** **Default: 1.**

6.4.2 DI/DO/AI/AO channel property

Properties value can be gotten through
Acm_GetChannelProperty
Acm_GetMultiChannelProperty

and set properties value by
Acm_SetChannelProperty
Acm_SetMultiChannelProperty.

Channel ID of DI/DO/AI/AO should be passed into property.

All properties in this chapter, user should know the DI/DO port and AI/AO channel mapping information by utility.

If user uses the default mapping relationship, the **Acm_DevLoadMapFile** need not to load, or else user should download mapfile use **Acm_DevLoadMapFile**.

6.4.2.1 Configuration

CFG_CH_DaqDiInvertEnable

PROPERTY ID	1500
PROPERTY TYPE	F64
R/W	R/W
DESCRIPTION	**Read or set DI Invert.** **0： Not Invert** **1： Invert**

CFG_CH_DaqDiLowFilter

PROPERTY ID	1501
PROPERTY TYPE	F64
R/W	R/W
DESCRIPTION	Read or set down limit value of DI filter. Range：0~65535 Note! DI filter down limit value will copy to all DI as one of them has been successfully set for ADAM-E5017.

CFG_CH_DaqDiHighFilter

PROPERTY ID	1502
PROPERTY TYPE	F64
R/W	R/W
DESCRIPTION	Read or set upper limit value of DI filter. Range：0~65535 Note! DI filter down limit value will copy to all DI as one of them has been successfully set for ADAM-E5017.

CFG_CH_DaqDoFsvEnable

PROPERTY ID	1503
PROPERTY TYPE	F64
R/W	R/W
DESCRIPTION	Read or set DO Failure Safe Value 0：As communication error，DO will be in Low Logic state 1：As communication error，DO will be in High Logic state

CFG_CH_DaqAiRange

PROPERTY ID	1505
PROPERTY TYPE	F64
R/W	R/W

DESCRIPTION	Read or Set AI channel input range Value Description 0 +/- 10V 1 +/- 5V (Only ADAM-E5017 Support) 2 +/- 2.5V (Not Support) 3 +/- 1.25V (Not Support) 4 +/- 0.625V (Not Support) 5 +/- 1V (Only ADAM-E5017 Support) 6 +/- 0.5V (Only ADAM-E5017 Support) 7 +/- 0.15V (Only ADAM-E5017 Support) 8 0~10V (Only ADAM-E5017UH Support) 9 0~500mV (Only ADAM-E5017UH Support) 10~ 15 Reserved 16 0 ~ 20 mA (Only ADAM-E5017UH Support) 17 4~ 20 mA 18 +/- 20 mA (Only ADAM-E5017 Support) Note! AI range setting will copy to all channels for ADAM-E5017

CFG_CH_DaqAiEnable

PROPERTY ID	1506
PROPERTY TYPE	F64
R/W	R/W
DESCRIPTION	Enable/Disable this AI 0：Disable this AI 1：Enable this AI

CFG_CH_DaqAiIntegrationTime

PROPERTY ID	1507
PROPERTY TYPE	F64
R/W	R/W
DESCRIPTION	Read or Set AI setting time 0：60HZ 1：50HZ

CFG_CH_DaqAoRange

PROPERTY ID	1504
PROPERTY TYPE	F64
R/W	R/W
DESCRIPTION	Read or Set AO range Value Description 0 +/- 10V (Not Support) 1 +/- 5V (Not Support) 2 +/- 2.5V (Not Support) 3 +/- 1.25V (Not Support) 4 +/- 0.625V (Not Support) 5 0~10V 6~15 Reserved 16 0 ~ 20 mA 17 4~ 20 mA

Error Code

0x80000000	InvalidDevNumber
0x80000001	DevRegDataLost
0x80000002	LoadDllFailed
0x80000003	GetProcAddrFailed
0x80000004	MemAllocateFailed
0x80000005	InvalidHandle
0x80000006	CreateFileFailed
0x80000007	OpenEventFailed
0x80000008	EventTimeOut
0x80000009	InvalidInputParam
0x8000000a	PropertyIDNotSupport
0x8000000b	PropertyIDReadOnly
0x8000008d	NotSetServoComPort
0x8000008e	OpenComPortFailed
0x8000008f	ReadComPortTimeOut
0x80000090	SetComPortStateFailed
0x80000091	SevroTypeNotSupport
0x80000092	ReadComBufFailed
0x80000096	SlaveIOUpdateError
0x80000097	NoSlaveDevFound
0x80000098	MasterDevNotOpen
0x80000099	MasterRingNotOpen
0x800000c8	InvalidDIPort
0x800000c9	InvalidDOPort
0x800000ca	InvalidDOValue
0x800000cb	CreateEventFailed
0x800000cc	CreateThreadFailed
0x800000ef	DataSizeNotCorrect
0x83000000	EC_GetNICNumberFailed
0x83000001	EC_GetNICInfoFailed
0x83000002	EC_OpenMasterDevFailed
0x83000003	EC_GetSlaveFailed
0x83000004	EC_StartOpModeFailed
0x83000005	EC_CloseDeviceFailed
0x83000006	EC_MemAllocateFailed

0x83000007	EC_InvalidNicIndex
0x83000008	EC_OpenDevFailed
0x83000009	EC_ReadFileFailed
0x8300000a	EC_GetNicInfoFailed
0x8300000b	EC_GetSDOFailed
0x8300000c	EC_InvalidParameter
0x8300000d	EC_GetPDOOffsetFailed
0x8300000e	EC_InitialMappingInfoFalied
0x8300000f	EC_InitResourceFailed
0x83000010	EC_SetSDOFailed
0x83000011	EC_InvalidPortType
0x83000012	EC_SetCycleTimeFailed
0x83000013	EC_InvalidAoRange
0x83000014	EC_InvalidAiRange
0x83000015	EC_GetSlaveInfoFaied
0x83000016	EC_GetNetWorkStateFaied
0x83000017	EC_RegisterEventFailed
0x83000018	EC_InvalidIntegrationTime
0x83000019	EC_InvalidAIMask
0x8300001a	EC_InvalidDIFilter
0x8300001b	EC_SetSlaveStateFailed
0x8300001c	EC_ZeroCalibrationFailed
0x8300001d	EC_InvalidMasterHandle
0x8300001e	EC_InvalidENIFile
0x8300001f	EC_InvalidCaliType
0x83000020	EC_SetCaliValueFailed
0x83000021	EC_AOCalibrationFailed
0x83000022	EC_InvalidIOMapping
0x83000023	EC_PortIndexGreaterThanPortNum
0x83000024	EC_ChannelIDGreaterThanChannelNum
0x83000025	EC_InputIndexGreaterThanInputNum
0x83000026	EC_OutputIndexGreaterThanOutputNum
0x83000027	EC_SetMaskFailed
0x83000028	EC_SetAIRangeFailed
0x83000029	EC_SetIntegrationTimeFailed
0x8300002a	EC_PropertyNotSupported
0x8300002b	EC_SlaveIDConflicted

0x8300002c	EC_SpanCalibrationFailed
0x8300002d	EC_InvalidAiValue
0x8300002e	EC_InvalidAoValue
0x8300002f	EC_GetModuleFailed

Appendix A

ADAM-E5000 IO modules

This appendix A will discuss in detail the specifications, functions and application wiring of the ADAM-5000/ECAT I/O module series. To organize an ADAM-5000/ECAT, you need to select I/O modules to interface the main unit with field devices or processes that you have previously determined. Advantech provides the following types of ADAM-E5000 I/O modules for various applications so far. Following table is the I/O modules support list we provided for user's choice.

A.1 Analog Input Modules

Analog input modules use an A/D converter to convert sensor voltage, current signals into digital data. The digital data is then translated into engi-neering units. The analog input modules protect your equipment from ground loops and power surges by providing opto-isolation of the A/D input and transformer based isolation up to 3,000 V_{DC}.

A.1.1 ADAM-E5017 8-ch Analog Input Module

The ADAM-E5017 is a 16-bit, 8-channel analog differential input module that provides programmable input ranges on all channels. It accepts millivolt inputs (±150mV, ±500mV), voltage inputs (±1V, ±5V and ±10V) and current input (±20 mA). The module provides data to the host computer in engineering units (mV, V or mA). This module is an extremely cost-effective solution for industrial measurement and monitoring applications. Its opto-isolated inputs provide 3,000 V_{DC} of isolation between the analog input and the module, protecting the module and peripherals from damage due to high input line voltage. Additionally, the module uses analog multiplexers with active over- voltage protection. The active protection circuitry assures that signal fidelity is maintained even under fault conditions that would destroy other multiplexers. This module can withstand an input voltage surge of 70 Vp-p with ±15 V supplies. The jumpers of ADAM-E5017 are designed for current input. Refer to the diagram below for the locations (JP1). Short the pin-head by the jumpers to set the channel to be current mode.

ADAM-E5017

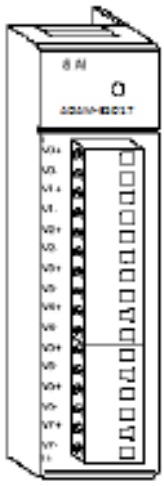


Figure A.1 ADAM-E5017 Module Front View

Application Wiring

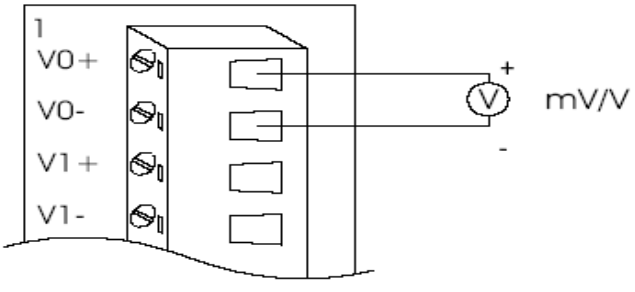


Figure A.2 Millivolt and Volt Input

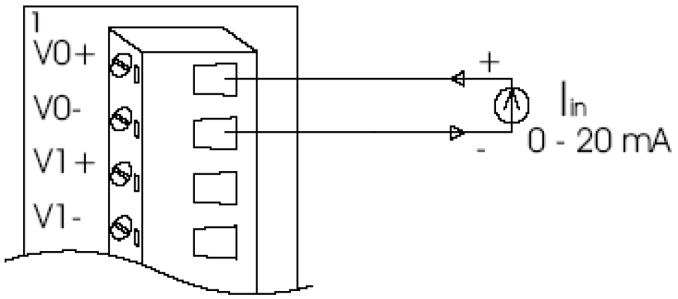


Figure A.3 Process Current Input

Note! *To keep measurement accuracy please short the channels that are not in use.*

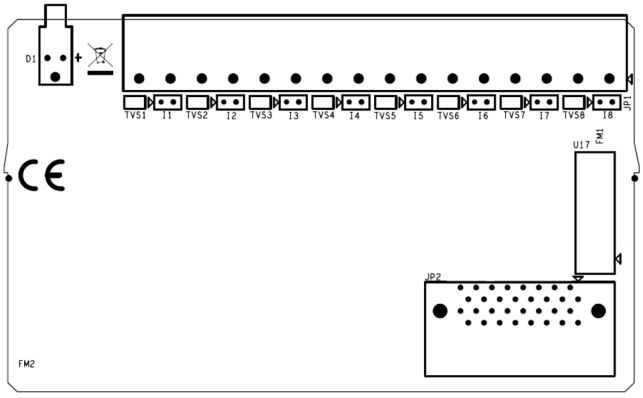


Figure A.4 Locations of Jumpers

Technical Specifications of ADAM-E5017

Table A1.1: Technical Specifications of ADAM-E5017	
Analog Input Channels	Eight differential
Input Type	mV, V, mA
Input Range	± 150 mV, ± 500 mV, ± 1 V, ± 5 V, ± 10 V and ± 20 mA
Isolation Voltage	3000 VDC
Sampling Rate	10 samples/sec (total)
Analog Input Signal Limit	15 V max.
Max. allowable voltage difference between two connectors in a module	15 V max.
Input Impedance	2MΩ (voltage input) 120Ω (current input)
Bandwidth	13.1 Hz @ 50 Hz, 15.72 Hz @ 60 Hz
Accuracy	± 0.1% or better
Zero Drift	± 1.5 μV/°C
Span Drift	± 25 PPM/°C
CMR @ 50/60 Hz	92 dB min.
Power Requirements	+ 10 to + 30 VDC (non-regulated)
Power Consumption	1.2 W

A.1.2 ADAM-E5017UH 8-ch Ultra High Speed Analog Input Module

The ADAM-E5017UH is a 12-bit plus sign bit, 8-channel analog differential input module that pro-vides programmable input ranges on each channel. It accepts voltage inputs (±10 V and 0-10 V) and current inputs (0-20 mA and 4-20 mA). The module provides data to the host microprocessor in engineering units (mV, V or mA) or two's complement format. Its sampling rate depends on the data format received: up to 200k Hz (total). Space is reserved for 125-ohm, 0.1%, 10 ppm resistors (See Figure 9). Each input channel has 3000 V_{DC} of optical isolation between the out-side analog input line and the module, protecting the module and peripherals from high input line voltages. Additionally, the module uses analog multiplexers with active overvoltage protec-tion. The active protection circuitry assures that signal fidelity is maintained even under fault conditions that would destroy other multiplexers. The analog inputs can withstand a constant 70 Vp-p input with ±15V supplies. The jumpers of ADAM-E5017UH are designed for current input.

ADAM-E5017UH

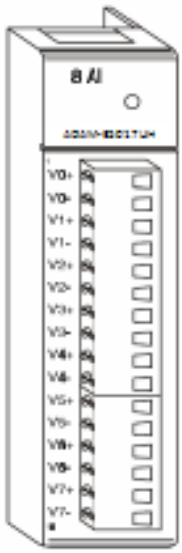


Figure A.5 ADAM-E5017UH Module Front View

Application Wiring

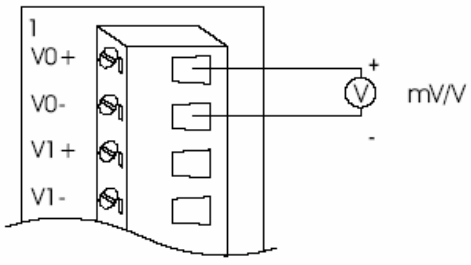


Figure A.6 Millivolt and Volt Input

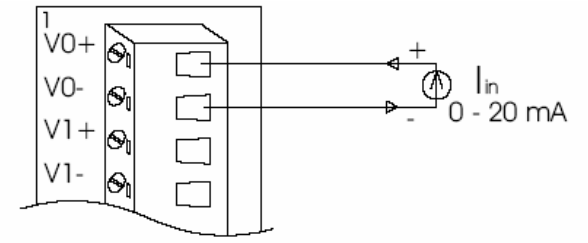


Figure A.7 Process Current Input

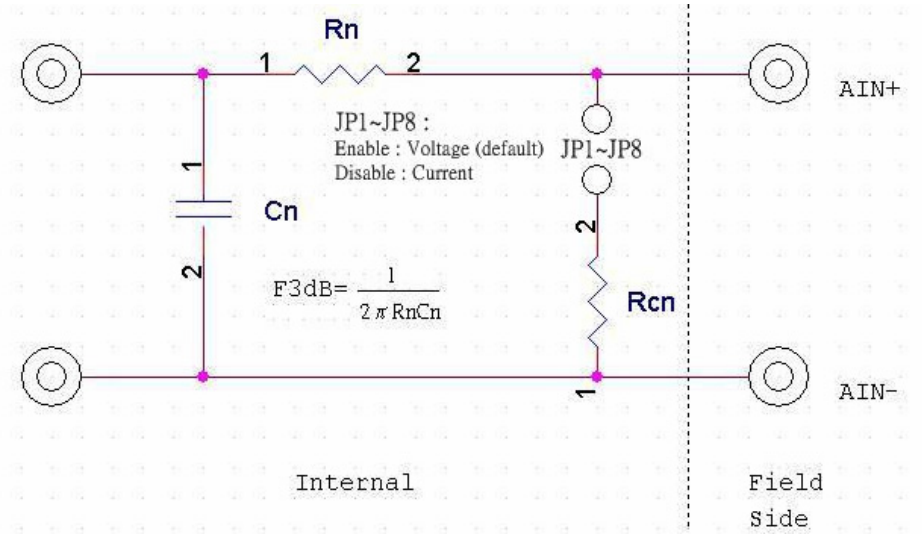


Figure A.8 Locations of RC Filter Jumper Setting

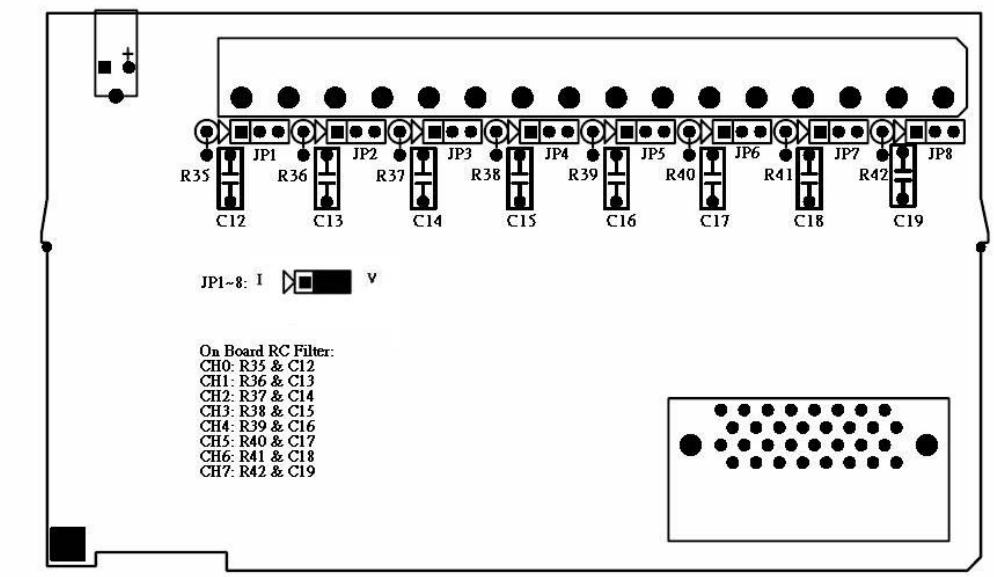


Figure A.9 Locations of RC Filter Jumper Setting

Note! *To maintain measurement accuracy please short channels not in use.*

Table A1.2: Technical Specifications of ADAM-E5017UH	
Analog Input Channels	Eight differential
Resolution	12 bits
Input Type	mV, V, mA
Input Range	+0~10V , ± 10 V , +4~20mA and ± 20 mA
Isolation Voltage	3000 VDC
Sampling Rate	200k samples/sec (single channel) 50k samples/sec (8 channel)
Analog Input Signal Limit	15 V max.
Max. allowable voltage difference between two connectors in a module	15 V max.
Input Impedance	2MΩ (voltage input) 120Ω (current input)
Bandwidth	200kHz
Accuracy	± 0.1% or better
Low or high pass filter	Configured by User
CMR @ 50/60 Hz	92 dB min.
Power Requirements	+ 10 to + 30 VDC (non-regulated)
Power Consumption	1.75 W (typical); 2.2W (max)
Signal Input Bandwidth	200kHz for both voltage and current inputs

Table A1.3: ADAM-E5017UH Input Sgnal Ranges						
	Input Range	Offset Error @ 25°C	Offset Error @ -10 to +70°C	Gain Error @ 25°C	Gain Error @ -10 to +70°C	Display Resolution
Voltage Inputs	0 ~ 10 V	±1 LSB	±2 LSB	±1 LSB	±2 LSB	2.7 mV
	± 10 V	±1 LSB	±2 LSB	±1 LSB	±2 LSB	2.7 mV
Current Inputs	0 ~ 20 mA	±1 LSB	±1 LSB	±1.5 LSB	±2 LSB	5.3 μ
	4 ~ 20 mA	±1 LSB	±1 LSB	±1.5 LSB	±2 LSB	5.3 μ

Appendix A ADAM-E5000 IO Modules

A.2 Analog Output Modules

A.2.1 ADAM-E5024H 4-ch Analog Output Module

The ADAM-E5024H is a 4-channel high speed analog output module. It receives its digital input through the EtherCAT interface of the ADAM-5000/ECAT from the host computer. The format of the data is engineering units. It then uses the D/A converter controlled by the system module to convert the digital data into output signals.

You can specify slew rates and start up currents through the configuration software. The analog output can also be configured as current or voltage through the software utility. The module protects your equipment from ground loops and power surges by providing opto-isolation of the D/A output and transformer based isolation up to 3000 VDC.

Slew Rate

The slew rate is defined as the slope indicated the ascending or descending rate per second of the analog output from the present to the required.

ADAM-E5024H

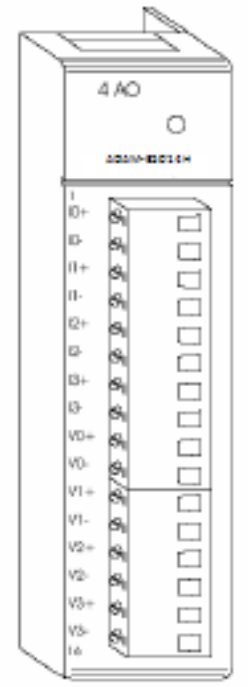


Figure A2.1 ADAM-E5024 Module Frontal View

Application Wiring

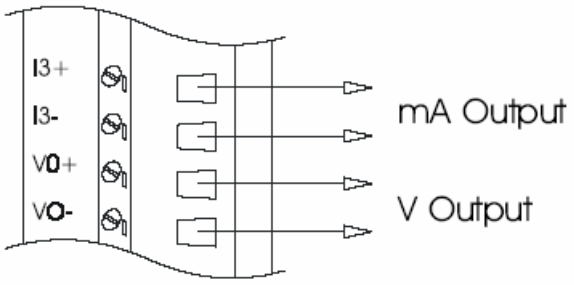


Figure A2.2 Analog Output

Technical Specifications of ADAM-E5024H

Table A2.1: Technical Specifications of ADAM-E5024H	
Analog Output Channels	Four
Output Type	V, mA
Output Range	0-20mA, 4-20mA, 0-10V
Isolation Voltage	3000 Vdc
Output Impedance	0.5 Ohms
Accuracy	±0.1% of FSR for current output ±0.2% of FSR for voltage output
Zero Drift	Voltage output: ±30 μV/ºC Current output: ±0.2 μA/ºC
Resolution	±0.015% of FSR
Span Temperature Coefficient	±25 PPM/ºC
Programmable Output Slope*	0.125-128.0 mA/sec 0.0625-64.0 V/sec
Current Load Resistor	0-500 Ohms (source)
Power Consumption	2.5W (Max.)

A.3 Digital Input/Output Modules

A.3.1 ADAM-E5051S 16-ch Isolated Digital Input Module with LED

The ADAM-E5051S provides 16 isolated digital input channels for critical environments need individual channel isolating protection. Different from other ADAM-E5000 I/O mod-ules, ADAM-E5051S designed with 21 pins plug terminal.

ADAM-E5051S

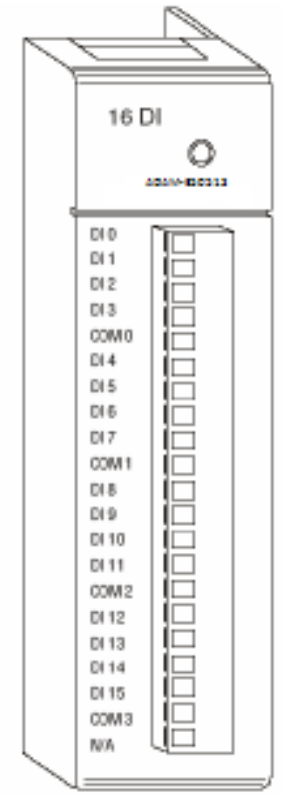


Figure A3.1 ADAM-E5051S Module Front View

Application Wiring

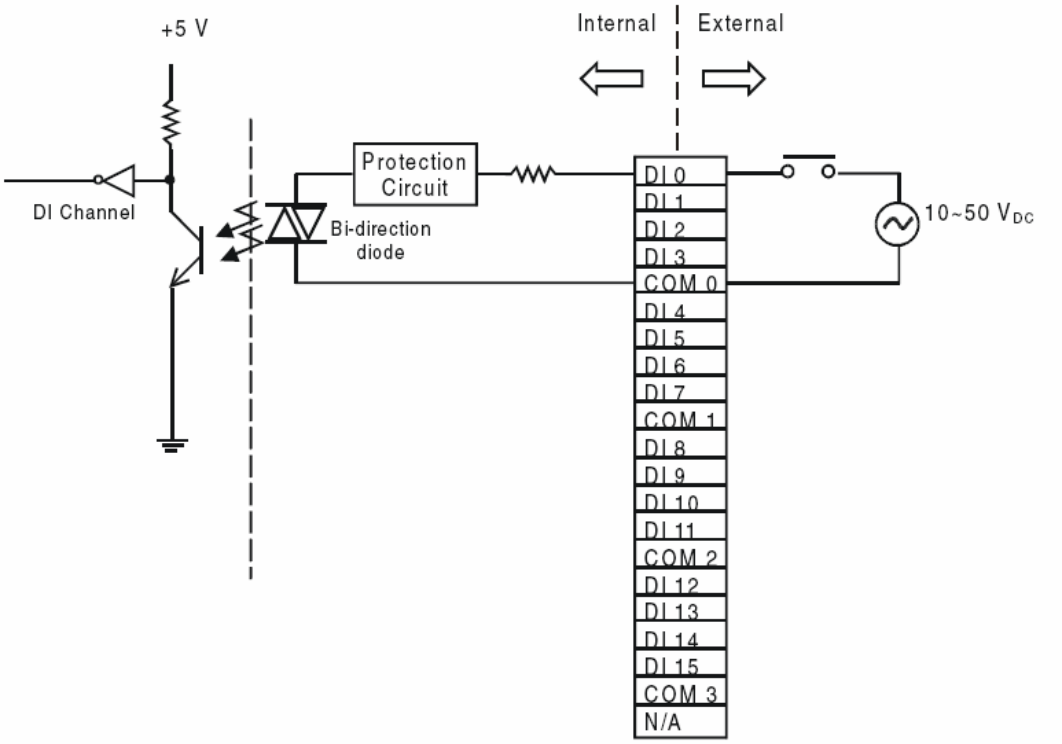


Figure A3.2 ADAM-E5051S Module Wiring Diagram

Table A3.1: Technical Specifications of ADAM-E5051S	
Point	16 (4-channel/group)
Digital Input	Logic Level 0: + 3 V max Logic Level 1: + 10 to 50 V
Optical Isolation	2500 V_{DC}
Opto-isolator response time	25 µs
Over-voltage Protection	70 V_{DC}
Power Consumption	0.8 W
LED Indicator	On when active
I/O Connector Type	21-pin plug-terminal

A.3.2 ADAM-E5053S 32-ch Isolated Digital Input Module

The ADAM-E5053S provides 32 isolated digital input channels for critical environ-ments need individual channel isolating protection. Different from other ADAM-E5000 I/O modules, ADAM-E5053S designed with 40-pin flat cable wiring terminal.

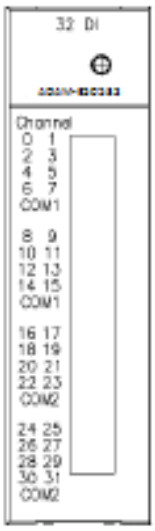


Figure A3.3 ADAM-E5053S Module Front View

Application Wiring

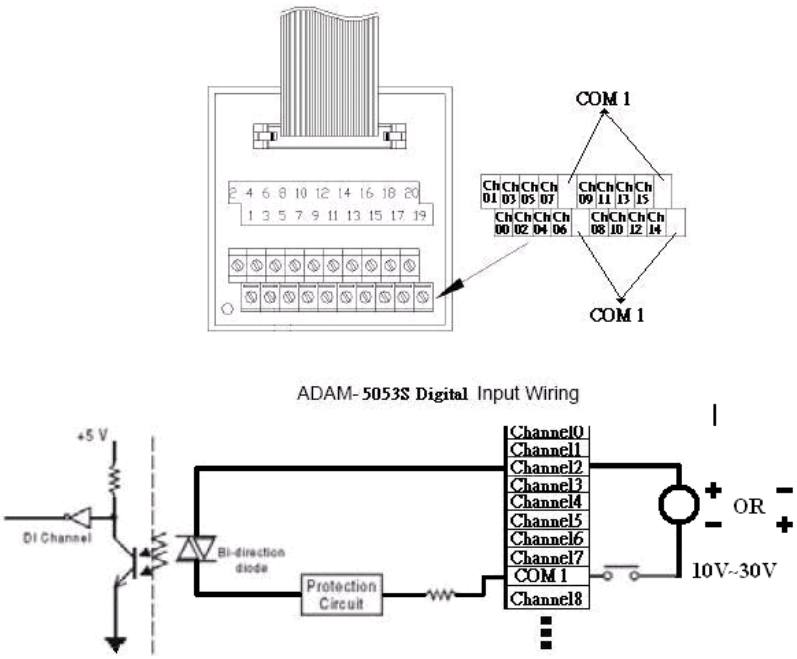


Figure A3.4 ADAM-E5053S Module Wiring Diagram

Table A3.2: ADAM-E5053S Technical Specifications

Point	32
Digital Input	Logic Level 0: + 5 V max Logic Level 1: 19 to 35 V
Optical Isolation	2500 VDC
Opto-isolator response time	25 µs
Over-voltage Protection	35 VDC
Power Consumption	1 W (max.)
I/O Connector Type	40-pin flat cable wiring terminal

A.3.3 ADAM-E5056S 16-ch Isolated Digital Output Module with LED

The ADAM-E5056S provides 16 isolated digital output channels for critical environments need individual channel isolating protection. Different from other ADAM-E5000 I/O mod-ules, ADAM-E5056S designed with 21 pins plug terminal.

ADAM-E5056S

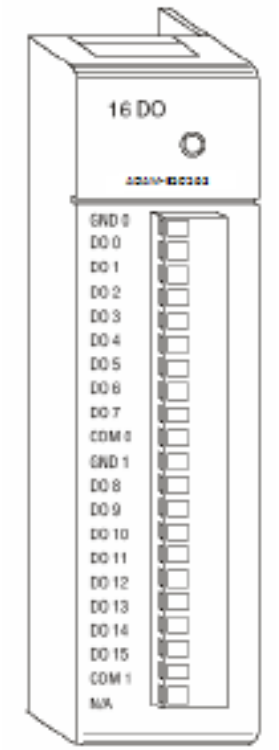


Figure A3.5 ADAM-E5056S Module Front View

Application Wiring

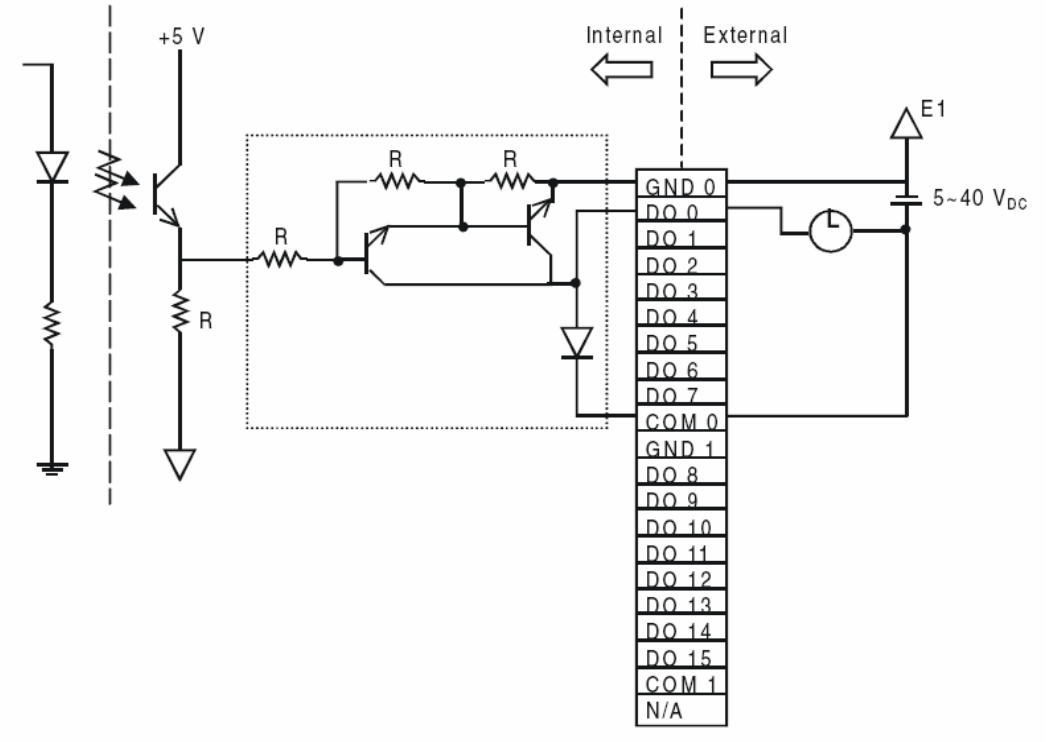


Figure A3.6 ADAM-E5056S Module Wiring Diagram

Table A3.3: Technical Specifications of ADAM-E5056S	
Points	16 (8-channel/group)
Digital Output	Open collector to 40 V 200 mA max load per channel
Optical Isolation	2500 V_{DC}
Opto-isolator response time	25 µs
Supply Voltage	5 ~ 40 V_{DC}
Power consumption	0.6 W
LED Indicator	On when active
I/O Connector Type	21-pin plug-terminal

A.3.4 ADAM-E5056SO 16-ch Isolated Digital Output Module with LED

The ADAM-E5056SO provides 16 channels source type isolated digital output for criti-cal environments need individual channel isolating protection. Addition to the source output wiring, all of the specification and command sets are the same with ADAM- E5056S.

ADAM-E5056SO

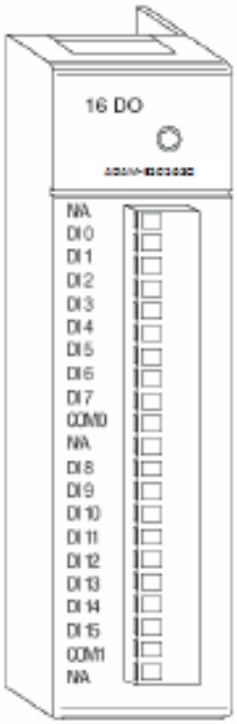


Figure A3.7 ADAM-E5056SO Module Front View

Application Wiring

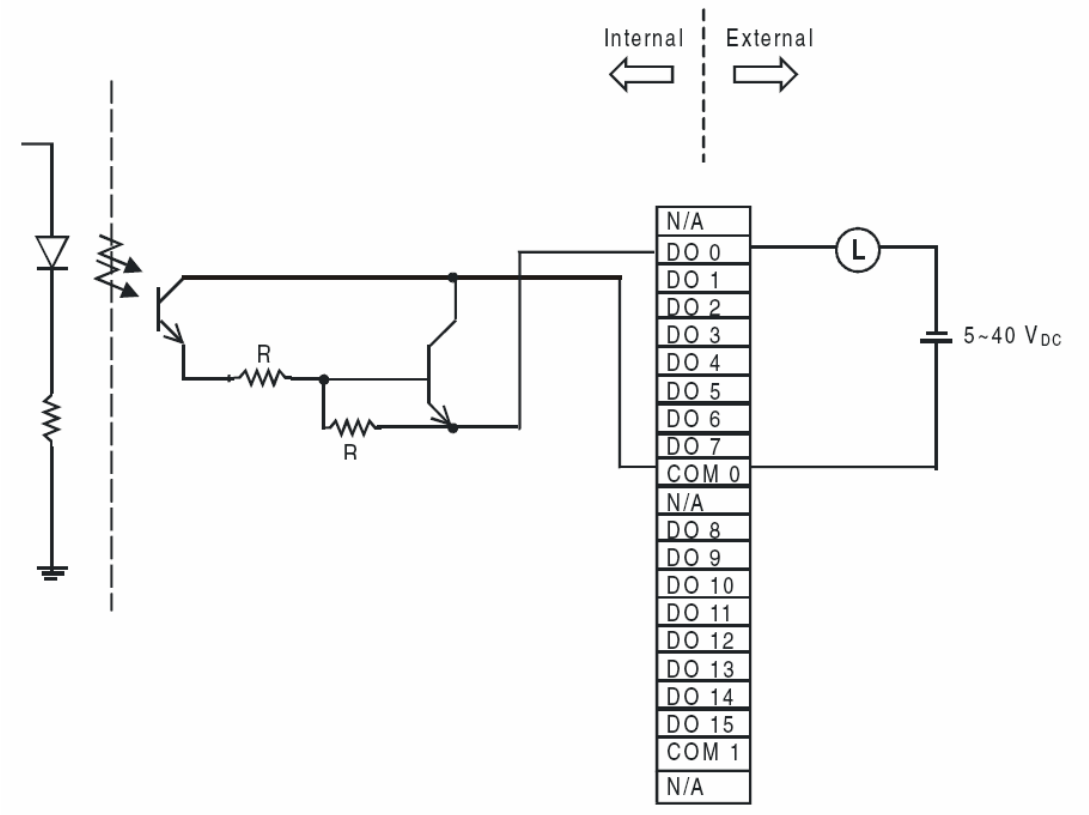


Figure A3.8 ADAM-E5056SO Module Wiring Diagram

Table A3.4: Technical Specifications of ADAM-E5056SO	
Points	16 (8-ch/group)
Digital Output	Open collector to 40 V 200 mA max load per channel
Optical Isolation	2500 VDC
Opto-isolator response time	25 us
Supply Voltage	5 ~ 40 VDC
Power consumption	0.6 W
LED Indicator	On when active
I/O Connector Type	21-pin plug-terminal

A.3.5 ADAM-E5057S 32-ch Isolated Digital Output Module

The ADAM-E5057S provides 32 isolated digital output channels for critical environ-ments need individual channel isolating protection. Different from other ADAM-E5000 I/O modules, ADAM-E5057S designed with 40-pin flat cable wiring terminal.

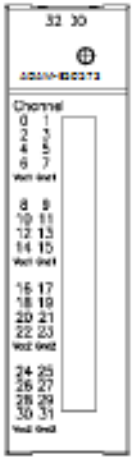


Figure A3.9 ADAM-E5057S Module Front View

Application Wiring

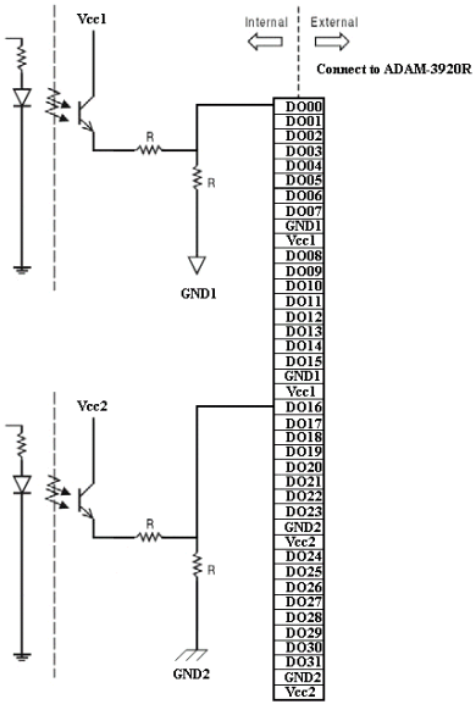


Figure A3.10 ADAM-E5057S Module Wiring Diagram

Table A3.5: ADAM-E5057S Technical Specification	
Points	32
Digital Output	Contact with ADAM-3920R
Optical Isolation	2500 VDC
Contact Rating	10 A 250VAC, 10 A 30VDC
Power Input	+24 VDC
Power consumption	1 W (max.)
Relay Type	SPST (Form A)
I/O Connector Type	40-pin flat cable wiring terminal

A.4 Relay Output Modules

A.4.1 ADAM-E5069 Relay Output Module

The ADAM-E5069 relay output module provides 8 relay channels of Form A. Switches can be used to control the relays. Considered to user friendly, the ADAM-E5069 also built with LED indicator for status reading easily. And it also provides a choice to clear or keep output status when reset by adjusting a jumper.

Specifications

Number of Output Channel: 8 Form A

Contact Rating:	AC:	250V@5A
	DC:	30V@5A
Breakdown Voltage:		750 VAC (50/60 Hz)
Insulation Resistance:		1000M£[@500VDC
LED Indicator:	On:	Active
	Off:	Non-active
Power Consumption:		0.25W (typical) 2.2W (Max)
Isolation Resistance:		4000 VRMS
Relay response Time: On:		5 ms
	Off:	
Clear or Keep Relay Sta		

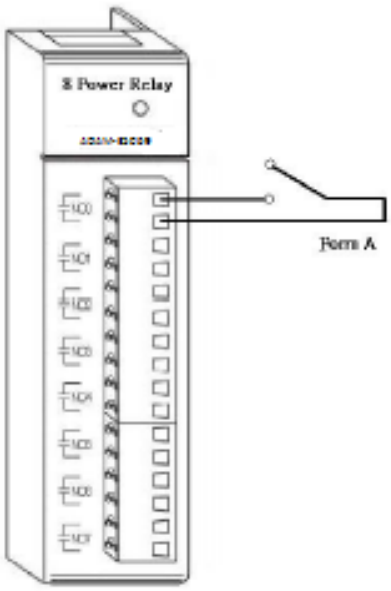


Figure A4.1 ADAM-E5069 Module Front View Wiring

ADVANTECH

Enabling an Intelligent Planet

www.advantech.com

Please verify specifications before quoting. This guide is intended for reference purposes only.

All product specifications are subject to change without notice.