ADAM-6000 Series

**Ethernet-based
Data Acquisition and
Control Modules**

User Manual

Acknowledgements

Intel and Pentium are trademarks of Intel Corporation.

Microsoft Windows and MS-DOS are registered trademarks of

Microsoft Corp.

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

Printed in Taiwan June 2012 5th Edition

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.

Technical Support and Assistance

Step 1. Visit the Advantech web site at **www.advantech.com/support** where you can find the latest information about the product.

Step 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, software, etc.)
- A complete description of the problem
- The exact wording of any error messages

Chapter 1 Understanding Your System 2
1.1 Introduction 2
Figure 1.1:ADAM-6000 System Architecture 2
1.2 Major Features 3
1.2.1 Ethernet-enabled DA&C I/O Modules 3
1.2.2 Intelligent I/O Modules 3
1.2.3 Mixed I/O to Fit All Applications 3
1.2.4 Remote Monitoring & Diagnosis 4
1.2.5 Industrial Standard Modbus/TCP Protocol 4
1.2.6 Customized Web Page 4
1.2.7 Modbus/TCP Software Support 4
1.3 Specifications 5
1.4 Dimensions 6
Figure 1.2:ADAM-6000 Module Dimension 6
1.5 LED Status 6
Figure 1.3:LED Indicators 6
Chapter 2 Selecting Your Hardware 10
2.1 Selecting an I/O Module 10
Table 2.1:I/O Selection Guidelines 11
2.2 Selecting a Link Terminal & Cable 11
Figure 2.1:Ethernet Terminal and Cable Connection 12
Table 2.2:Ethernet RJ-45 port Pin Assignment 12
2.3 Selecting an Operator Interface 13
Chapter 3 Hardware Installation Guide 16
3.1 Determining the Proper Environment 16
3.1.1 Package Contents 16
3.1.2 System Requirements 16
3.2 Mounting 17
3.2.1 Panel Mounting 17
Figure 3.1:Panel Mounting Dimensions 17
Figure 3.2:Fix Module on theBracket 18
3.2.2 DIN-rail mounting 18
Figure 3.3: Fix Module on the DIN-rail Adapter 19
Figure 3.4:Secure Module to a DIN-rail 20
3.3 Wiring & Connections 20
3.3.1 Power Supply Wiring 20
Figure 3.5:ADAM-6000 Module Power Wiring 21
3.3.2 I/O Module Wiring 21
Chapter 4 I/O Module Introduction 24
4.1 Analog Input Modules 24
4.1.1 ADAM-6015 24
Figure 4.1:ADAM-6015 RTD Input Wiring 26
4.1.2 ADAM-6017 27
Figure 4.2:ADAM-6017 Analog Input Wiring 28
Figure 4.3:ADAM-6017 Analog Input Type Setting 29

Figure 4.4:ADAM-6017 Digital Output Wiring 29
4.1.3 ADAM-6018 30
Figure 4.5:ADAM-6018 8-ch Thermocouple Input 30
Figure 4.6:ADAM-6018 Thermocouple Input Wiring 32
Figure 4.7:ADAM-6018 Digital Output Wiring 33
4.1.4 ADAM-6024 34
Figure 4.8:ADAM-6024 Jumper Settings 36
Figure 4.9: 36
Figure 4.10:ADAM-6024 AI/O Wiring 36
Figure 4.11:ADAM-6024 DI Wiring 37
Figure 4.12:ADAM-6024 DO Wiring 37
4.2 Digital I/O Modules 38
4.2.1 ADAM-6050 38
Figure 4.13:ADAM-6050 Digital Input Wiring 39
Figure 4.14:ADAM-6050 Digital Output Wiring 40
4.2.2 ADAM-6051 41
Figure 4.15:ADAM-6051 Digital Input Wiring 42
Figure 4.16:ADAM-6051 Counter (Frequency) Input 43
Figure 4.17:ADAM-6051 DO Wiring 43
4.2.3 ADAM-6052 44
Figure 4.18:ADAM-6052 DI (Dry Contact) Wiring ... 45
Figure 4.19:ADAM-6052 DI (Wet Contact) Wiring ... 46
Figure 4.20:ADAM-6052 Digital Output Wiring 46
4.2.4 ADAM-6060 47
Figure 4.21:ADAM-6060 Digital Input Wiring 49
Figure 4.22:ADAM-6060 Relay Output Wiring 49
4.2.5 ADAM-6066 50
Figure 4.23:ADAM-6066 Digital Input Wiring 52
Figure 4.24:ADAM-6066 Relay Output Wiring 52

Chapter 5 System Configuration Guide 54

5.1 System Hardware Configuration 54
5.1.1 System Requirements 54
5.1.2 Communication Interface 54
5.2 Install ADAM.NET Utility Software 54
5.3 ADAM.NET Utility Overview 55
5.3.1 ADAM.NET Utility Operation Window 55
Figure 5.1:ADAM.NET Utility Operation Window 55
Figure 5.2:ADAM.NET Utility Toolbar 58
5.3.2 Search ADAM-6000 Modules 59
Figure 5.3:Access Control Setting 65
5.3.3 I/O Module Configuration 66
Figure 5.4:Channel & GCL Configuration 66
Figure 5.5:Channels Range Configuration Area 67
Figure 5.6:Integration Time Configuration Area 68
Figure 5.7:Analog Input Trend Log 69

Figure 5.8:Analog Input Average Setting 70
Figure 5.9:Analog Input Alarm Mode Configuration .. 71
Figure 5.10:ADAM-6024 Input Tab 73
Figure 5.11:ADAM-6024 Output Tab 74
Figure 5.12:ADAM-6050 Channel Setting 76
Figure 5.13:Fail Safe Value Configuration 77
Figure 5.14:Individual Channel Configuration: DI 78
Figure 5.15:Individual Channel Configuration: DO 80
Figure 5.16:Low to High Delay Output Mode 82
Figure 5.17:Low to High Delay Output Mode 82
5.3.4 Peer-to-Peer Function 83
Figure 5.18:Basic mode for Peer-to-Peer 84
Figure 5.19:Advanced mode for Peer-to-Peer 84
Figure 5.20:Peer-to-Peer Configuration Tab 86
Figure 5.21:Peer-to-Peer Basic Mode Configuration .. 87
Figure 5.22:Building the Mapping Relationship 89
Figure 5.23:P-to-P Advanced Mode Configuration 90
Figure 5.24:Copy One Setting to Other Channels 92
5.4 ADAM-6000 Web Server 93
5.5 Java Applet Customization..... 93
5.5.1 Introduction 93
Figure 5.25:Structure of the ADAM6060.jar file 97
Figure 5.26:Firmware Upgrade 98
5.6 Source Code of Java Applet Example..... 99
Chapter 6 Planning Your Application Program 108
6.1 Introduction..... 108
6.2 ADAM .NET Class Library 108
Figure 6.1:Modifying ADAM-6050 .NET 109
Figure 6.2:Launching ADAM .NET Class Library ... 110
6.3 ADAM-6000 Commands..... 111
6.3.1 Command Structure 111
6.3.2 Modbus Function Code Introductions 112
6.4 ASCII Commands for ADAM-6000 Modules..... 118
6.4.1 Syntax of ASCII 118
6.4.2 System Command Set 119
6.4.3 Analog Input Command Set 124
6.4.4 Analog Input Alarm Command Set Set 138
6.4.5 Universal I/O Command Set 148
6.4.6 Digital Input/Output Command Set 158
Chapter 7 Graphic Condition Logic(GCL)..... 164
7.1 Overview 164
7.2 GCL Configuration Environment..... 165
Figure 7.1:GCL Configuration Environment 165
Figure 7.2:Four Stages for One Logic Rule 167
7.3 Configure Four Stages of One Logic Rule..... 170
7.3.1 Input Condition Stage 170

Figure 7.3:Input Condition Stage Configuration 170
Figure 7.4:Engineer Unit and Current Value 172
Figure 7.5:Scaling Function of Analog Input Mode .. 173
7.3.2 Logic Stage 176
Figure 7.6:Logic Stage Configuration 176
7.3.3 Execution Stage 178
Figure 7.7:Execution Stage Configuration 178
Figure 7.8:Send to Next Rule Function 179
Figure 7.9:The Next Logic Rule 180
7.3.4 Output Stage 180
Figure 7.10:Output Stage Configuration 181
Figure 7.11:Remote Message Output 186
7.4 Internal Flag for Logic Cascade and Feedback..... 188
7.4.1 Logic Cascade 188
Figure 7.12:Architecture of Local Logic Cascade 189
Figure 7.13:Configuration of Logic Rule 1 190
Figure 7.14:Configuration of Logic Rule 2 190
Figure 7.15:Configuration of Logic Rule 3 191
Figure 7.16: Distributed Logic Cascade 192
Figure 7.17:Configuration of Logic Rule 1 192
Figure 7.18:Configuration of Logic Rule 2 193
Figure 7.19:Configuration of Logic Rule 3 193
7.4.2 Feedback 194
Figure 7.20:Building Logic Feedback 194
7.5 Download Logic and Online Monitoring..... 194
Figure 7.21:Online Monitoring Function 195
Figure 7.22:GCL Execution Sequence 196
7.6 Typical Applications with GCL..... 197
Figure 7.23:Ladder Diagram for On/Off Control 198
Figure 7.24:GCL Logic for On/Off Control 198
Figure 7.25:Time Chart for Sequence Control 199
Figure 7.26:GCL Logic for Sequence Control 200
Figure 7.27:Time Chart for 12 DI to 1 DO 201
Figure 7.28:GCL Logic for 12 DI to 1 DO 202
Figure 7.29:Time Chart for Flicker Application 202
Figure 7.30:GCL Logic for Flicker 203
Figure 7.31:Time Chart for Rising Edge 203
Figure 7.32:Ladder Diagram for Rising Edge 204
Figure 7.33:GCL Logic for Rising Edge 205
Figure 7.34:Time Chart for Falling Edge 205
Figure 7.35:Ladder Diagram for Falling Edge 206
Figure 7.36:GCL Logic for Falling Edge 207
Figure 7.37:Time Chart for Sequence Control 207
Figure 7.38:GCL Logic for Sequence Control 208
Figure 7.39:GCL Logic for Event Trigger 209
Figure 7.40:Event Trigger Configuration 209

Appendix A Design Worksheets 212
Table A.1:I/O Data Base 212
Table A.2:Summary Required Modules 213
Table A.3:Table for Programming 214
Appendix B Data Formats and I/O Range 216
B.1 ADAM-6000 Commands Data Formats 216
B.1.1 Command Structure 216
Figure B.1:Request Comment Structure 217
Figure B.2:Response Comment Structure 217
B.1.2 Modbus Function Code Introductions 218
Table B.1:Response Comment Structure 218
B.2 ADAM-6000 I/O Modbus Mapping Table 224
B.2.1 ADAM-6015 224
B.2.2 ADAM-6017 226
B.2.3 ADAM-6018 228
B.2.4 ADAM-6024 230
B.2.5 ADAM-6052 16-ch Digital I/O Module 231
B.2.6 ADAM-6060/6060W/6066 233
Appendix C Grounding Reference 238
C.1 Field Grounding and Shielding Application 238
C.2 Grounding 239
C.2.1 The 'Earth' for Reference 239
Figure C.1:Think of the Earth as a Ground. 239
C.2.2 The 'Frame Ground' and 'Grounding Bar' 240
Figure C.2:Grounding Bar 240
Figure C.3:Normal and Common Mode. 240
C.2.3 Normal Mode and Common Mode 241
Figure C.4:Normal and Common Mode. 241
C.2.4 Wire impedance 242
Figure C.5:High Voltage Transmission 242
Figure C.6:Wire Impedance 243
C.2.5 Single Point Grounding 243
Figure C.7:Single Point Grounding (1) 243
Figure C.8:Single point grounding (2) 244
C.3 Shielding 244
C.3.1 Cable Shield 244
Figure C.9:Single isolated cable 244
Figure C.10:Double isolated cable 245
C.3.2 System Shielding 246
Figure C.11:System Shielding 246
Figure C.12:The characteristic of the cable 247
Figure C.13:System Shielding (1) 247
Figure C.14:System Shielding (2) 248
C.4 Noise Reduction Techniques 248
Figure C.15:Noise Reduction Techniques 249
C.5 Check Point List 249

CHAPTER 1

Understanding Your System

Sections include:

- Introduction
- Major Features
- Specifications
- Dimensions
- LED Status

Chapter 1 Understanding Your System

1.1 Introduction

ADAM-6000 Ethernet-based data acquisition and control modules provide I/O, data acquisitions, and networking in one module to build a cost-effective, distributed monitoring and control solution for a wide variety of applications. Through standard Ethernet networking, ADAM-6000 retrieves I/O values from sensors, and can publish them as a real-time I/O values to networking nodes via LAN, Intranet, or Internet. With Ethernet-enabled technology, ADAM-6000 series modules build up a cost-effective DA&C system for Building Automation, Environmental Monitoring, Facility Management and eManufacturing applications. Please refer to Figure 1-1 for a brief overview of the ADAM-6000 system architecture.

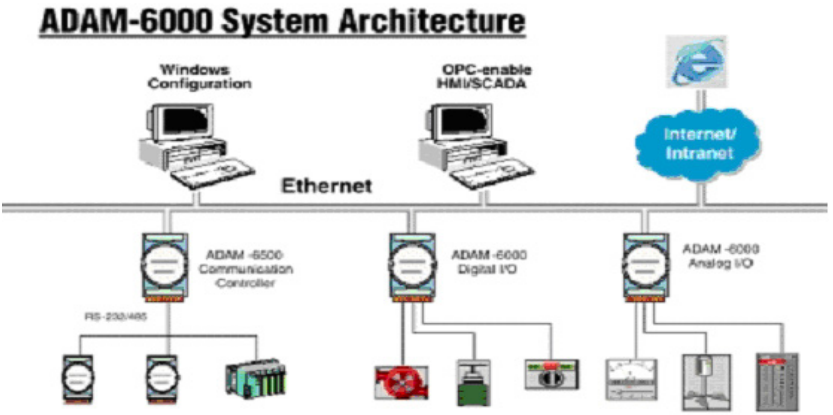


Figure 1.1: ADAM-6000 System Architecture

1.2 Major Features

1.2.1 Ethernet-enabled DA&C I/O Modules

ADAM-6000 is based on popular Ethernet networking standards used in most business environments. Users can easily add ADAM-6000 I/O modules to existing Ethernet networks, or use ADAM-6000 modules in new Ethernet-enabled eManufacturing networks. ADAM-6000 modules feature a 10/100 Mbps Ethernet chip and support industrial popular Modbus/TCP protocols over TCP/IP for data connection. ADAM-6000 also supports UDP protocol over Ethernet networking. With UDP/IP, ADAM-6000 I/O modules can actively send I/O data stream to 8 Ethernet nodes. Through Ethernet networking, HMI/SCADA systems, and controllers, users can access or gather real-time data from ADAM-6000 Ethernet enabled DA&C modules. This data can then be integrated with business systems to compile valuable business information.

1.2.2 Intelligent I/O Modules

Upgraded from traditional I/O modules, the ADAM-6000 series have prebuilt intelligent mathematic functions to empower system capacity. The Digital Input modules provide Counter, Totalizer functions; the Digital Output modules provide pulse output, delay output functions; the Analog Input modules provide the Max./Min./Average data calculation; the Analog Output modules provide the PID loop control function.

1.2.3 Mixed I/O to Fit All Applications

ADAM-6000 series mixed I/O design provides the most cost-effective I/O usage for application systems. The most common used I/O type for single function units are collected in one module. This design concept not only saves I/O usage and saves costs, but also speeds up I/O relative operations. For small DA&C system or standalone control units from mid to large scales, ADAM-6000's mixed I/O design can easily fit application needs with one or two modules only. With additional embedded control modules, ADAM-6000 can easily create a localized, less complex, and more distributed I/O architecture.

1.2.4 Remote Monitoring & Diagnosis

Each ADAM-6000 module features a pre-built I/O module web page to display real-time I/O data values, alarms, and module status thru LAN or Internet. Through any Internet browser, users can monitor real-time I/O data values and alarms at local or remote sites. Then, the web-enabled monitoring system is completed immediately without any programming.

1.2.5 Industrial Standard Modbus/TCP Protocol

ADAM-6000 modules support the popular industrial standard, Modbus/TCP protocol, to connect with Ethernet Controller or HMI/SCADA software built with Modbus/TCP driver. Advantech also provides OPC server for Modbus/TCP to integrate ADAM-6000 I/O real-time data value with OPC client enabled software, freeing users from driver development.

1.2.6 Customized Web Page

Since ADAM-6000 modules build in a default web page, users can monitor and control the I/O status in anywhere through Internet Explorer Browser. Moreover, ADAM-6000 modules can download user-defined web pages for individual applications. Advantech has provided sample programs of JAVA Script for users reference to design their own operator interface, then download it into the specific ADAM-6000 modules via Windows Utility.

1.2.7 Modbus/TCP Software Support

The ADAM-6000 firmware is a built-in Modbus/TCP server. Therefore, Advantech provides the necessary OPC Server, ADAM .NET Class Library and Windows ADAM .NET Utility for users. Users can configure this DA&C system via Windows Utility; integrate with HMI software package via Modbus/TCP driver or Modbus/TCP OPC Server. Even more, you can use the DLL driver and ActiveX to develop your own applications.

1.3 Specifications

Ethernet:	10/100 Base-T
Wiring:	UTP, category 5 or greater
Bus Connection:	RJ45 modular jack
Comm. Protocol:	Modbus/TCP on TCP/IP and UDP
Data Transfer Rate:	Up to 100 Mbps
	Unregulated 10 to 30 VDC
Status Indicator:	Power, CPU, Communication
	(Link, Collide, 10/100 Mbps, Tx, Rx)
Case:	ABS + PC with captive mounting hardware
Screw Terminal Block:	Accepts 0.5 mm 2 to 2.5 mm 2 , 1 - #12 or
	2 - #14 to #22 AWG

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 Dimensions

The following diagram show the dimensions of the l/O modules. (mm)

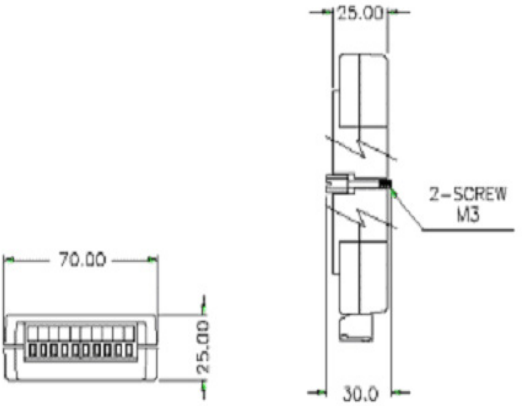


Figure 1.2: ADAM-6000 Module Dimension

1.5 LED Status

There are two LEDs on the ADAM-6000 I/O Series front panel. Each LED has two indicators to represent system status, as explained below:

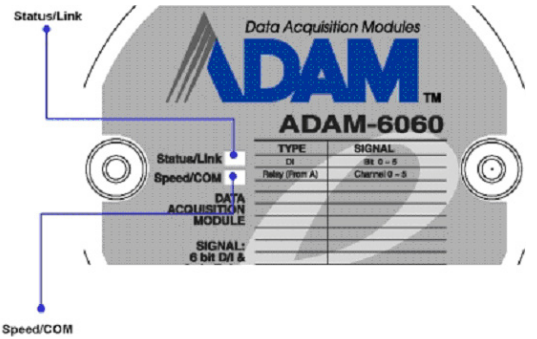


Figure 1.3: LED Indicators

(1) Status: Red indicator. Blinks when ADAM-6000 module is running.

(2) Link: Green indicator. On whenever the Ethernet is connected.

(3) Speed: Red indicator. On when Ethernet speed is higher than 100 Mbps.

(4) COM: Green indicator. Blinks whenever the ADAM-6000 module is transmitting or receiving data via Ethernet.

(5)There is a Locate function in the utility. If you locate the module, the Status (Red) & Link (Green) indicators will stay on.

CHAPTER 2

Selecting Your Hardware

Sections include:

- Selecting an I/O Module
- Selecting a Link Terminal & Cable
- Selecting an Operator Interface

Chapter 2 Selecting Your Hardware

2.1 Selecting an I/O Module

To organize an ADAM-6000 remote data acquisition & control system, you need to select I/O modules to interface the host PC 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 modules to handle I/O points in individual areas of an entire field site?
- How many modules are required for distributed I/O point arrangement?
- How many hubs are required for the connection of these devices?
- 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 for I/O module selection guidelines

.

Table 2.1: I/O Selection Guidelines		
Choose this type of I/O module:	For these types of field devices or operations (examples):	Explanation:
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 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
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 host device
Analog output module	Analog valves, actuators, chart recorders, electric motor drives, analog meters	Interpret host device's output to analog signals (generally through transducers) for field devices.

2.2 Selecting a Link Terminal & Cable

Use the RJ-45 connector to connect the Ethernet port of the ADAM-6000 to the Hub. The cable for connection should be Category 3 (for 10Mbps data rate) or Category 5 (for 100Mbps data rate) UTP/STP cable, which is compliant with EIA/TIA 586 specifications. Maximum length between the Hub and any ADAM-6000 modules is up to 100 meters (appr. 300 ft).

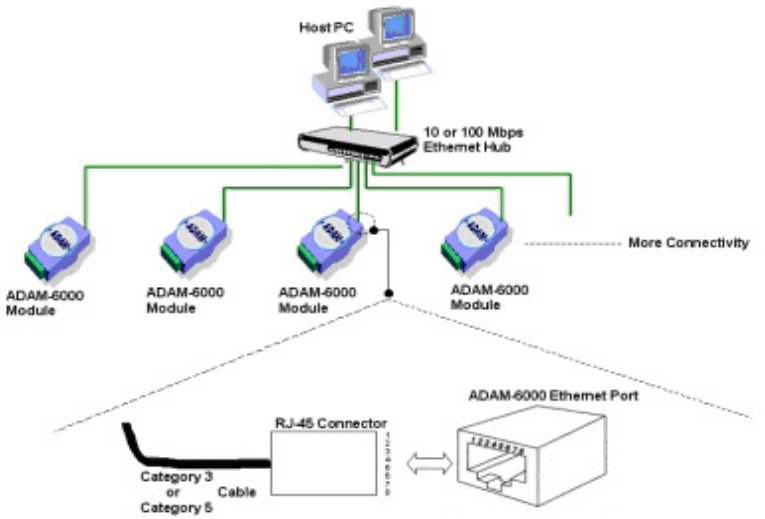


Figure 2.1: Ethernet Terminal and Cable Connection

Table 2.2: Ethernet RJ-45 port Pin Assignment		
PIN NUMBER	SIGNAL	FUNCTION
1	RD+	Receive (+)
2	RD-	Receive (-)
3	TD+	Transmit (+)
4	(Not Used)	-
5	(Not Used)	-
6	TD-	Transmit (-)
7	(Not Used)	-
8	(Not Used)	-

2.3 Selecting an Operator Interface

To complete your Data Acquisition and Control system, selecting the operator interface is necessary. Adopting the Modbus/TCP Protocol, ADAM-6000 I/O modules exhibit high ability in system integration for various applications.

If you want to read the real-time status of ADAM-6000 modules through the web page from anywhere without any engineering effort, there are many Internet browser software:

- Internet Explorer, Netscape, and other browser with JAVA Machine

If you want to develop your own web pages in the ADAM-6000 modules, the JAVA Script will be the quick and easy programming tool to design a specific operator interface.

- J2EE Development Kit

If you want to integrate ADAM-6000 I/O with HMI (Human Machine Interface) software in a SCADA (Supervisory Control and Data Acquisition) system, there are a lot of HMI software packages, which support Modbus/TCP driver.

- Advantech PM Designer
- Wonderware InTouch
- Any other software that supports the Modbus/TCP protocol

Moreover, Advantech also provides OPC Server, the most easy-to-use data exchange tool in worldwide. Any HMI software designed with OPC Client would be able to access ADAM-6000 I/O modules.

- Modbus/TCP OPC Server

If you want to develop your own applications, the ADAM.NET Class Library will be the best tool to build up users operator interface.

With these ready-to-go application software packages, tasks such as remote data acquisition, process control, historical trending and data analysis require only a few keystrokes.

CHAPTER 3

Hardware Installation Guide

Sections include:

- Determining the Proper Environment
- Mounting
- Wiring & Connections

Chapter 3 Hardware Installation Guide

3.1 Determining the Proper Environment

Prior to installing ADAM-6000 modules, please check the following.

3.1.1 Package Contents

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

- ADAM-6000 module with one bracket and DIN-rail adapter
- ADAM-6000 module User Manual
- ADAM-6000 CD

3.1.2 System Requirements

Host Computer

- Microsoft Windows CE/XP/7
- At least 32 MB RAM
- 20 MB of hard disk space available
- VGA color monitor
- 2x or higher speed CD-ROM
- Mouse or other pointing devices
- 10/100 Mbps or higher Ethernet Card

Ethernet Hub (at least 2 ports)

Two Ethernet Cables with RJ-45 connector

Power supply for ADAM-6000 (+10 to +30 V Unregulated)

3.2 Mounting

ADAM-6000 modules are designed as compact units and are allowed to be installed in the field site under the following methods.

3.2.1 Panel Mounting

Each ADAM-6000 Module is packed with a plastic panel mounting bracket. Users can refer the dimensions of the bracket to configure an optimal placement in a panel or cabinet. Fix the bracket first, then, fix the ADAM-6000 module on the bracket.

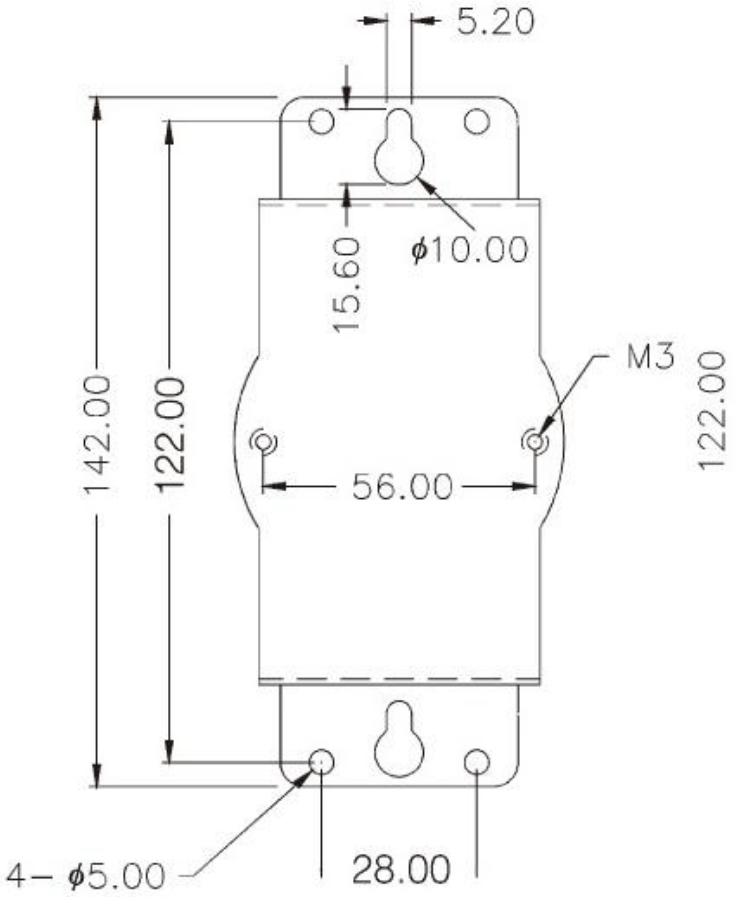


Figure 3.1: Panel Mounting Dimensions

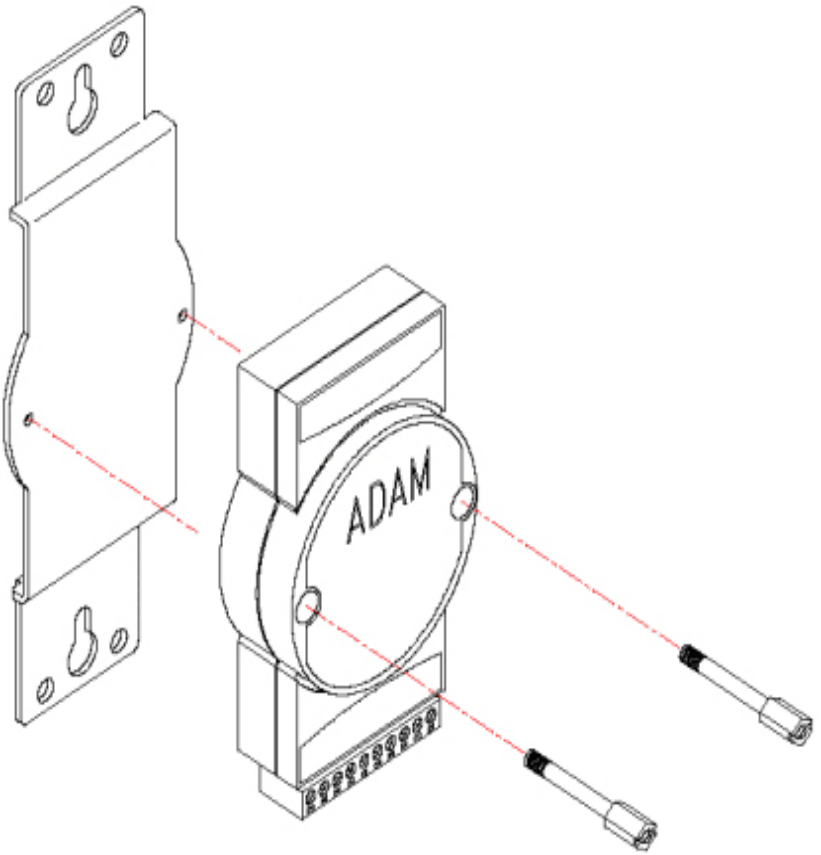


Figure 3.2: Fix Module on theBracket

3.2.2 DIN-rail mounting

The ADAM-6000 module can also be secured to the cabinet by using mounting rails. Fix the ADAM-6000 module with the DIN-rail adapter as Figure 3-3. Then secure it on the DIN-rail as Figure 3-4. If you mount the module on a rail, you should also consider using end brackets at each end of the rail. The end brackets help keep the modules from sliding horizontally along the rail.

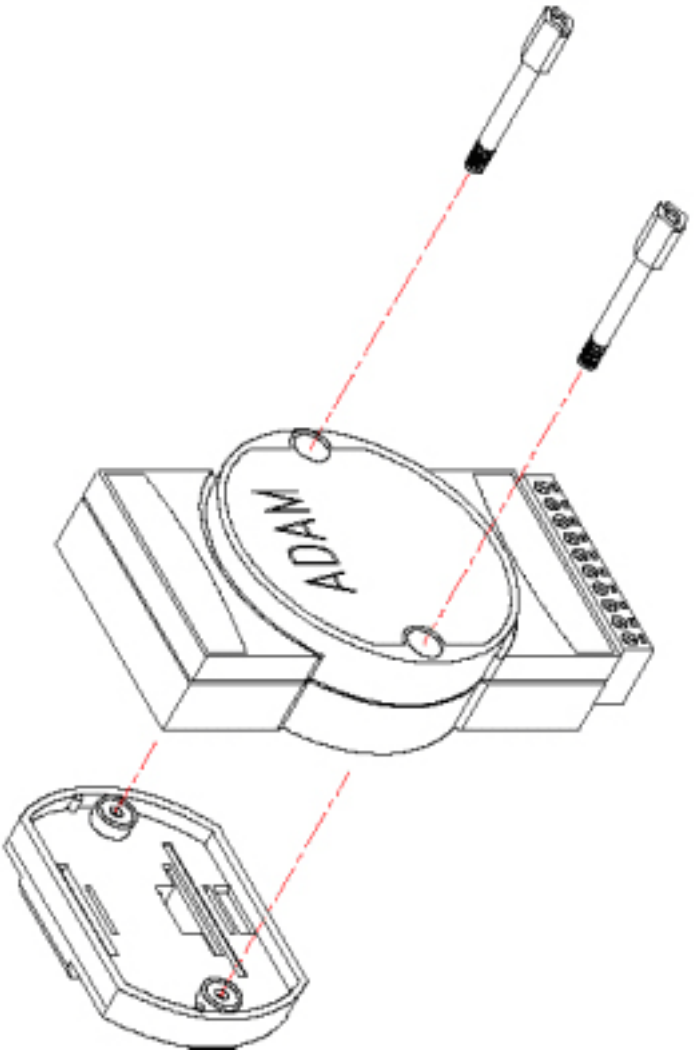


Figure 3.3: Fix Module on the DIN-rail Adapter

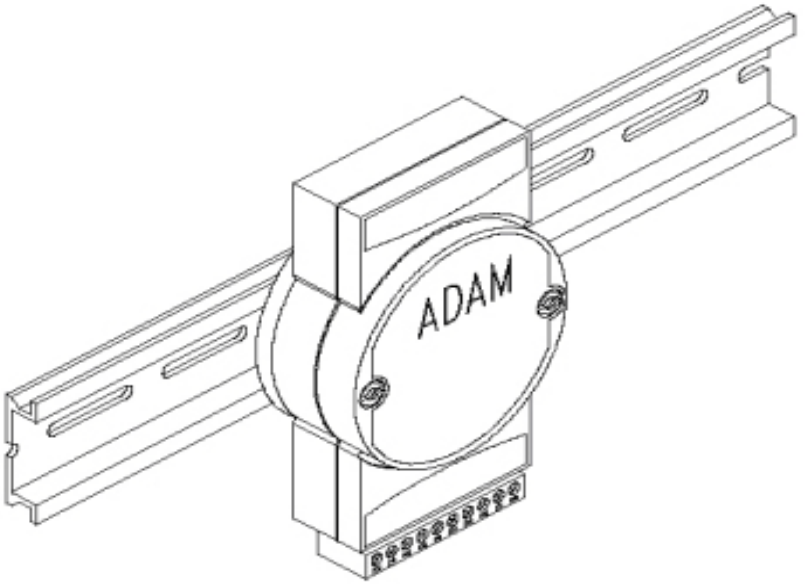


Figure 3.4: Secure Module to a DIN-rail

3.3 Wiring & Connections

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

3.3.1 Power Supply Wiring

Although the ADAM-6000/TCP systems are designed for a standard industrial unregulated 24 VDC 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 at least 2 mm.

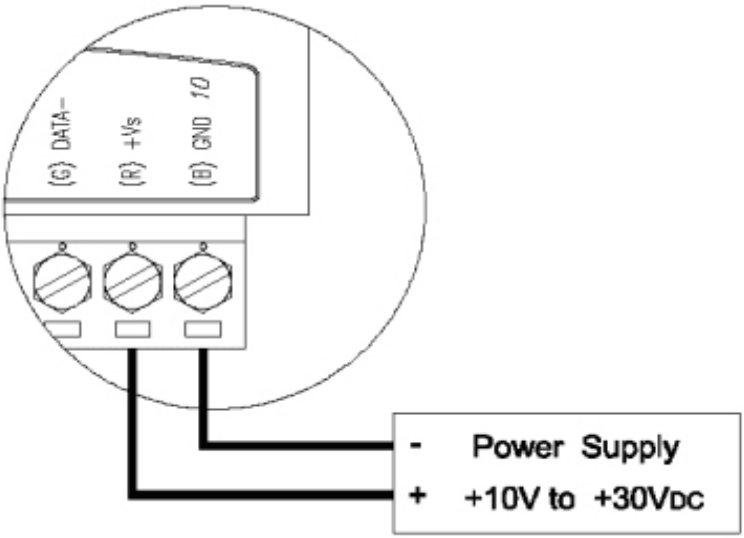


Figure 3.5: ADAM-6000 Module Power Wiring

We advise that the following standard colors (indicated on the modules) be used for power lines:

+Vs (R) Red

GND (B) Black

3.3.2 I/O Module 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.
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.
7. Avoid creating sharp bends in the wires.

CHAPTER 4

I/O Module Introduction

Sections include:

- Analog Input Modules
- Digital I/O Modules
- 16-ch Digital I/O w/ Counter

Chapter 4 I/O Module Introduction

4.1 Analog Input Modules

Analog input modules use an A/D converter to convert sensor voltage, current, thermocouple or RTD signals into digital data. The digital data is then translated into engineering units. When prompted by the host computer, the data is sent through a standard 10/100 Base-T Ethernet or IEEE 802.11b WLAN. Users can read the current status via pre-built webpage or HMI software supported Modbus/TCP protocol. 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.

4.1.1 ADAM-6015

7-ch Isolated RTD Input Module

The ADAM-6015 is a 16-bit, 7-channel RTD input module that provides programmable input ranges on all channels. It accepts various RTD inputs (PT100, PT1000, Balco 500 & Ni) and provides data to the host computer in engineering units (°C). In order to satisfy various temperature requirements in one module, each analog channel is allowed to configure an individual range for several applications.

ADAM-6015 Specifications

- Communication: 10/100 Base-T Ethernet
- Supports Protocols: Modbus/TCP, TCP/IP, UDP, HTTP, ICMP, ARP
- Supports Peer-to-Peer and GCL (Refer to Section 5.3.4 and Chapter 7)

Analog Input:

- Channels: 7 (differential)
- Input Impedance: > 10 Ω
- Input Connections: 2 or 3 wire
- Input Type: Pt, Balco and Ni RTD
- RTD Types and Temperature Range:
- Pt 100: -50 ~ 150°C
  0 ~ 100°C
  0 ~ 200°C
  0 ~ 400°C
  -200 ~ 200°C

IEC RTD 100 ohms (α = 0.0385)

JIS RTD 100 ohms (α = 0.0392)

- Pt 1000: -40 ~ 160°C
- Balco 500: -30 ~ 120°C
- Ni 518: -80 ~ 100°C
  0 ~ 100°C
- Accuracy: ± 0.1% or better
- Span Drift: ± 25 ppm/°C
- Zero Drift: ± 6 μV/°C
- Resolution: 16-bit
- Sampling Rate: 10 sample/second
- CMR @ 50/60 Hz: 90 dB
- NMR @ 50/60 Hz: 60 dB
- Wire Burn-out Detection
- Over Voltage Protection: ±35 VDC
- Built-in TVS/ESD Protection

General:

- Built-in Watchdog Timer
- Isolation Protection: 2000 VDC
- Power Input: Unregulated 10 ~ 30 VDC
- Power Consumption: 2 W @ 24 VDC
- Power Reversal Protection
- Operating Humidity: 20 ~ 95% RH (non-condensing)
- Storage Humidity: 0 ~ 95% RH (non-condensing)
- Operating Temperature: -10 ~ 70°C
- Storage Temperature: -20 ~ 80°C

Application Wiring

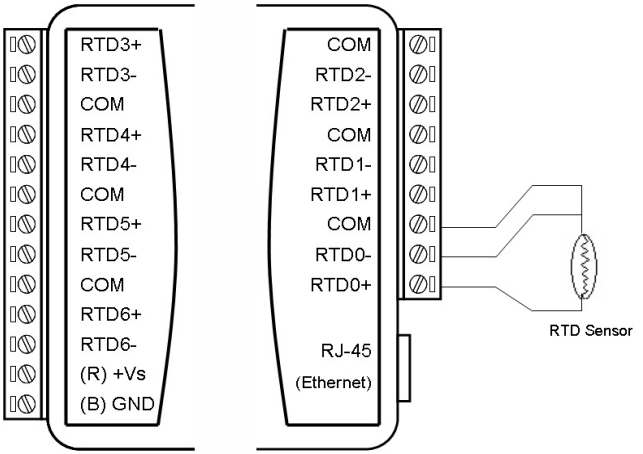


Figure 4.1: ADAM-6015 RTD Input Wiring

Assigning Addresses for ADAM-6015 Modules

Based on the Modbus/TCP standard, the addresses of the I/O channels in ADAM-6000 modules you place in the system are defined by a simple rule. Please refer to Appendix B.2.1 to map the I/O address.

4.1.2 ADAM-6017

8-ch Analog Input with 2-ch Digital Output Module

The ADAM-6017 is a 16-bit, 8-channel analog differential input module that provides programmable input ranges on all channels. It accepts milli-voltage inputs (±150mV, ±500mV), voltage inputs (±1V, ±5V and ±10V) and current input (0~20 mA, 4~20 mA) and provides data to the host computer in engineering units (mV, V or mA). In order to satisfy all plant needs in one module, ADAM-6017 has designed with 8 analog inputs and 2 digital outputs. Each analog channel is allowed to configure an individual range for variety of applications.

ADAM-6017 Specifications

- Communication: 10/100 Base-T Ethernet
- Supports Protocol: Modbus/TCP, TCP/IP, UDP, HTTP, ICMP and ARP
- Supports Peer-to-Peer and GCL (Refer to Section 5.3.4 and Chapter 7)

Analog Input:

- Channels: 8 (differential)
- Input Impedance: > 10MΩ (voltage), 120Ω (current)
- lnput Type: mV, V, mA
- lnput Range: ±150mV, ±500mV, ±1 V, ±5V, ±10V, 0-20 mA, 4-20 mA
- Accuracy: ±0.1% or Better
- Span Drift: ±25 ppm/° C
- Zero Drift: ±6 μV/° C
- Resolution: 16-bit
- Sampling Rate: 10 sample/second
- CMR @ 50/60 Hz: 90 dB
- NMR @ 50/60 Hz: 60 dB
- Over Voltage Protection: ±35 VDC
- Built-in TVS/ESD Protection

Digital Output:

- Channels: 2
- Sink type: Open Collector to 30 V, 100 mA (maximum load)
- Power Dissipation: 300 mW for each module

General:

- Built-in Watchdog Timer
- Isolation Protection: 2000 VDC
- Power Input: Unregulated 10 ~ 30 VDC
- Power Consumption: 2 W @ 24 VDC
- Power Reversal Protection
- Operating Humidity: 20 ~ 95% RH (non-condensing)
- Storage Humidity: 0 ~ 95% RH (non-condensing)
- Operating Temperature: -10 ~ 70°C
- Storage Temperature: -20 ~ 80°C

Application Wiring

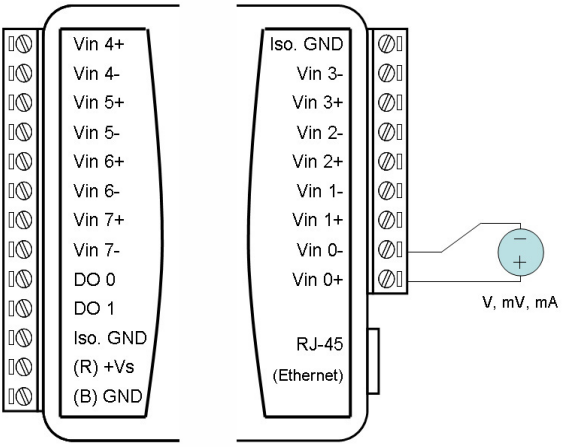


Figure 4.2: ADAM-6017 Analog Input Wiring

ADAM-6017 is built with a 120 resistor in each channel, users do not have to add any resistors in addition for current input measurement. Just adjust the jumper setting to choose the specific input type you need. Refer to Figure 4.3, each analog input channel has built-in a jumper on the PCB for users to set as a voltage mode or current mode.

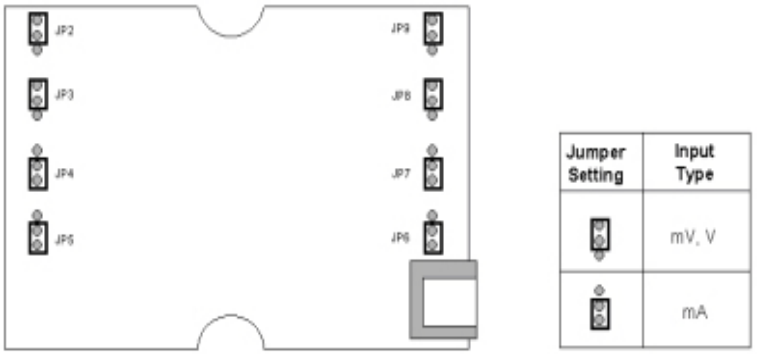


Figure 4.3: ADAM-6017 Analog Input Type Setting

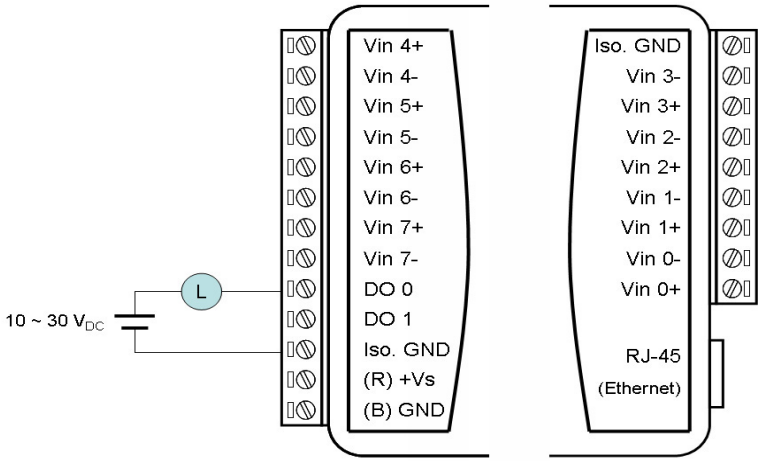


Figure 4.4: ADAM-6017 Digital Output Wiring

Assigning Addresses in ADAM-6017 Modules

Basing on Modbus/TCP standard, the addresses of the I/O channels in ADAM-6000 modules you place in the system are defined by a simple rule. Please refer to Appendix B.2.2 to map the I/O address.

4.1.3 ADAM-6018

Isolated Thermocouple Input with 8-ch Digital Output Module

The ADAM-6018 is a 16-bit, 8-channel thermocouple input module that provides programmable input ranges on all channels. It accepts various Thermocouple inputs (Type J, K, T, E, R, S, B) and allows each analog channel to configure an individual range for several applications. In order to satisfy all plant needs in one module, ADAM-6018 has designed with 8 thermocouple input and 8 digital output channels.

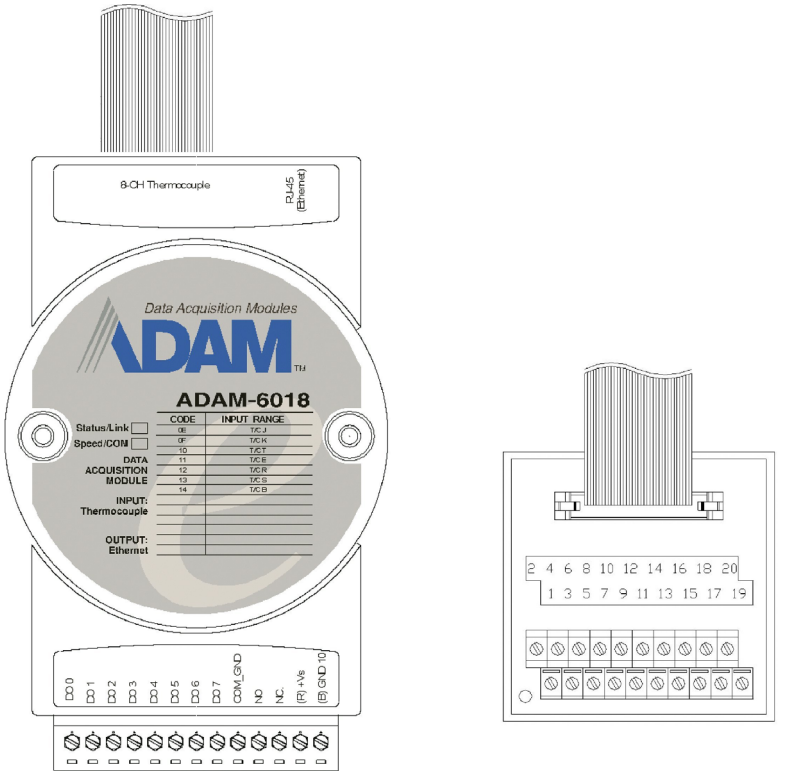


Figure 4.5: ADAM-6018 8-ch Thermocouple Input

ADAM-6018 Specifications

- Communication: 10/100 Base-T Ethernet
- Supports Protocol: Modbus/TCP, TCP/IP, UDP, HTTP, ICMP and ARP
- Supports Peer-to-Peer and GCL (Refer to Section 5.3.4 and Chapter 7)

Analog Input

- Channels: 8 (differential)
- Input Impedance: > 10 MΩ
- lnput Type: Thermocouple
- Thermocouple Type Range:
  - J Type: 0 ~ 760°C
  - K Type: 0 ~ 1370°C
  - T Type: -100 ~ 400°C
  - E Type: 0 ~ 1000°C
  - R Type: 500 ~ 1750°C
  - S Type: 500 ~ 1750°C
  - B Type: 500 ~ 1800°C
- Accuracy: ±0.1% or Better
- Span Drift: ±25 ppm/°C
- Zero Drift: ±6 μV/°C
- Resolution: 16-bit
- Sampling Rate: 10 sample/second
- CMR @ 50/60 Hz: 90 dB
- NMR @ 50/60 Hz: 60 dB
- Over Voltage Protection ±35 VDC
- Built-in TVS/ESD Protection
- Wire Burn-out Detection

Digital Output

- Channels: 8
- Sink type: Open Collector to 30 V, 100 mA (maximum load)
- Power Dissipation: 300 mW for each module

General:

- Built-in Watchdog Timer
- Isolation Protection: 2000 VDC
- Power Input: Unregulated 10 ~ 30 VDC
- Power Consumption: 2 W @ 24 VDC
- Power Reversal Protection
- Operating Humidity: 20 ~ 95% RH (non-condensing)
- Storage Humidity: 0 ~ 95% RH (non-condensing)
- Operating Temperature: -10 ~ 70°C
- Storage Temperature: -20 ~ 80°C

Application Wiring

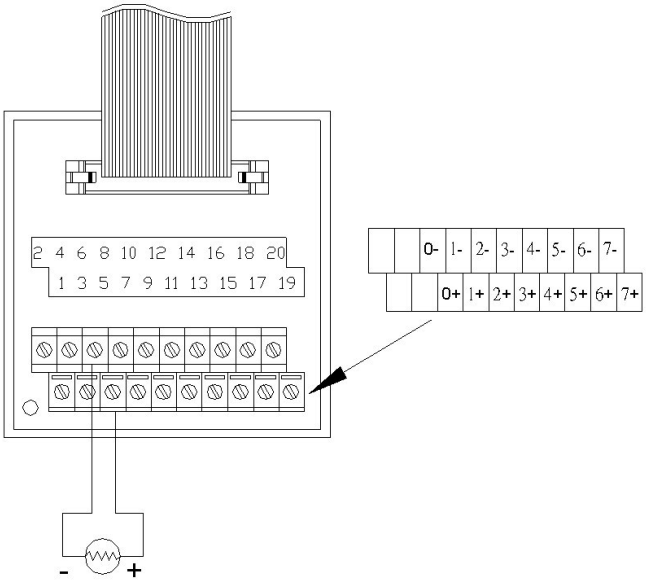


Figure 4.6: ADAM-6018 Thermocouple Input Wiring

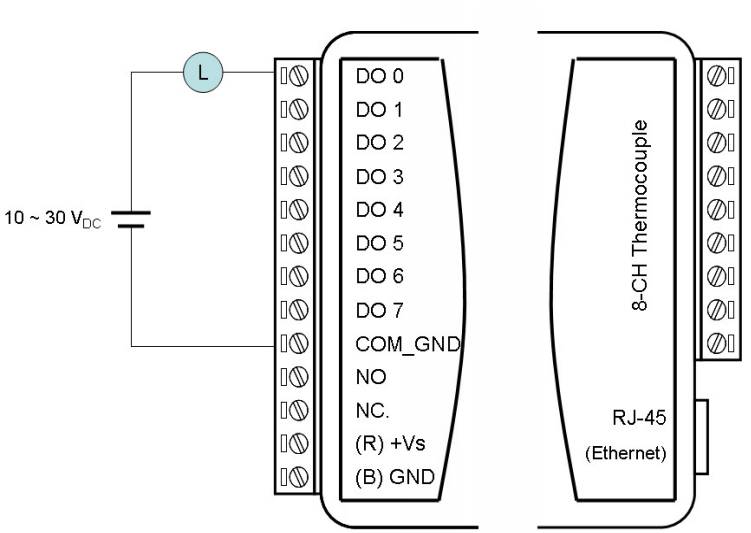


Figure 4.7: ADAM-6018 Digital Output Wiring

Assigning Addresses for ADAM-6018 Modules

Based on the Modbus/TCP standard, the addresses of the I/O channels in ADAM-6000 modules you place in the system are defined by a simple rule. Please refer to Appendix B.2.3 to map the I/O address.

4.1.4 ADAM-6024

12-ch Isolated Universal Input/Output Module

The ADAM-6024 is a 12-channel Universal Input/Output module. There are 6 analog input, 2 analog output, 2 digital input and 2 digital output channels. The analog input channels is 16-bit, universal signal accepted design. It provides programmable input ranges on all channels. It accepts various analog inputs +/-10V, 0~20mA and 4~20mA. The analog output channel is 12 bit with 0~10V, 0~20mA and 4~20mA acceptable input type. Each analog channel is allowed to configure an individual range for several applications.

Specifications

- Communication: 10/100 Base-T Ethernet
- Supports Protocol: Modbus/TCP, TCP/IP, UDP, HTTP, ICMP and ARP
- Receives data from other modules with Peer-to-Peer and GCL function only and generates analog output signals (Refer to Section 5.3.4 and Chapter 7 for more detail about Peer-to-Peer and GCL)

Analog Input

- Channels: 6 (differential)
- Range: ±10 VDC, 0~20 mA, 4~20 mA
- Input Impedance: >10 MΩ
- Accuracy: ±0.1% of FSR
- Resolution: 16-bit
- CMR @ 50/60 Hz: 90 dB
- NMR @ 50/60 Hz: 60 dB
- Span Drift: ±25 ppm/° C
- Zero Drift: ±6 μV/° C
- Isolation Protection: 2000 VDC

Analog Output

- Channels: 2
- Range: 0 ~ 10 VDC, 0~20 mA, 4~20 mA
- Accuracy: ±0.1% of FSR
- Resolution: 12-bit
- Current Load Resistor: 0 ~ 500Ω
- Isolation Protection: 2000 VDC
- Drift: ±50 ppm/°C

Digital Input

- Channels: 2
- Dry Contact: Logic level 0: close to GND
  Logic level 1: open
- Wet Contact: Logic level 0: 0 ~ 3 VDC
  Logic level 1: 10 ~ 30 VDC

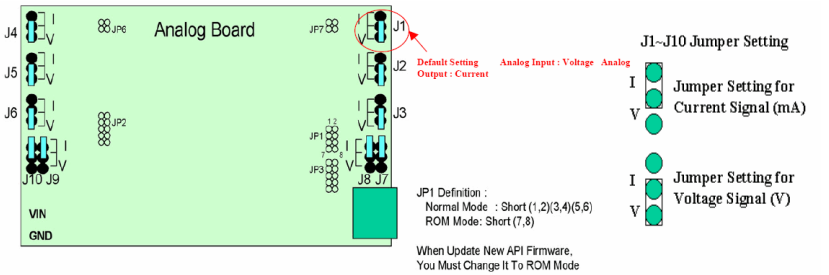
Digital Output

- Channels: 2
- Sink type: Open collector to 30 V, 100 mA (maximum)
- Power Dissipation: 300 mW for each module

General:

- Built-in Watchdog Timer
- Isolation Protection: 2000VDC
- Power Input: Unregulated 10 ~ 30 VDC
- Power Consumption: 4 W @ 24 VDC
- Power Reversal Protection
- Operating Humidity: 20 ~ 95% RH (non-condensing)
- Storage Humidity: 0 ~ 95% RH (non-condensing)
- Operating Temperature: -10 ~ 50°C
- Storage Temperature: -20 ~ 80°C

Jumper Settings



Channel	Jumper	Current	Voltage
AI0	J1	I	V
AI1	J2	I	V
AI2	J3	I	V
AI3	J4	I	V
AI4	J5	I	V
AI5	J6	I	V
AO0	J7	I	V
	J8	I	V
AO1	J9	I	V
	J10	I	V

Figure 4.8: ADAM-6024 Jumper Settings

Figure 4.9:

Application Wiring

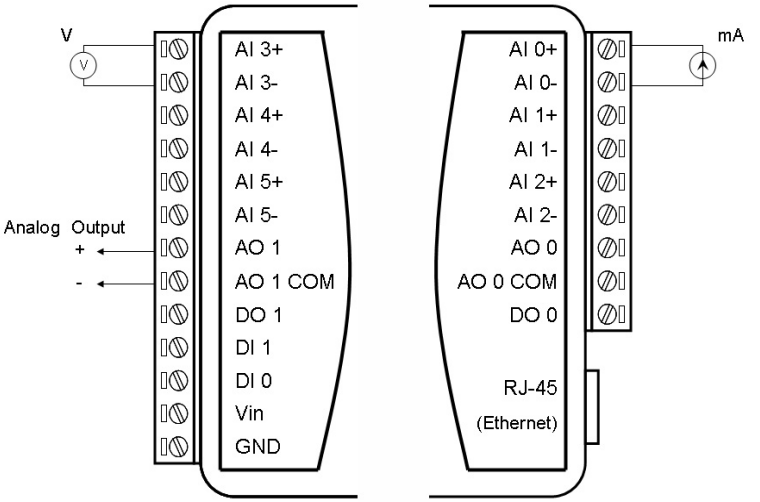


Figure 4.10: ADAM-6024 AI/O Wiring

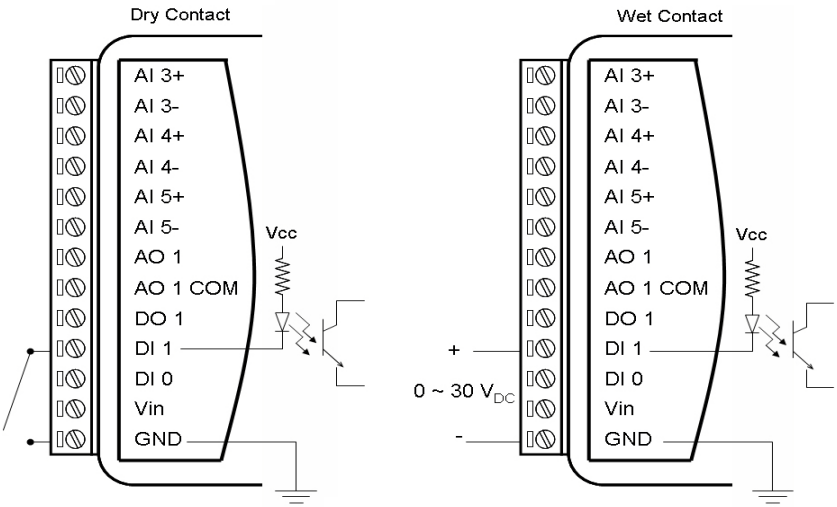


Figure 4.11: ADAM-6024 DI Wiring

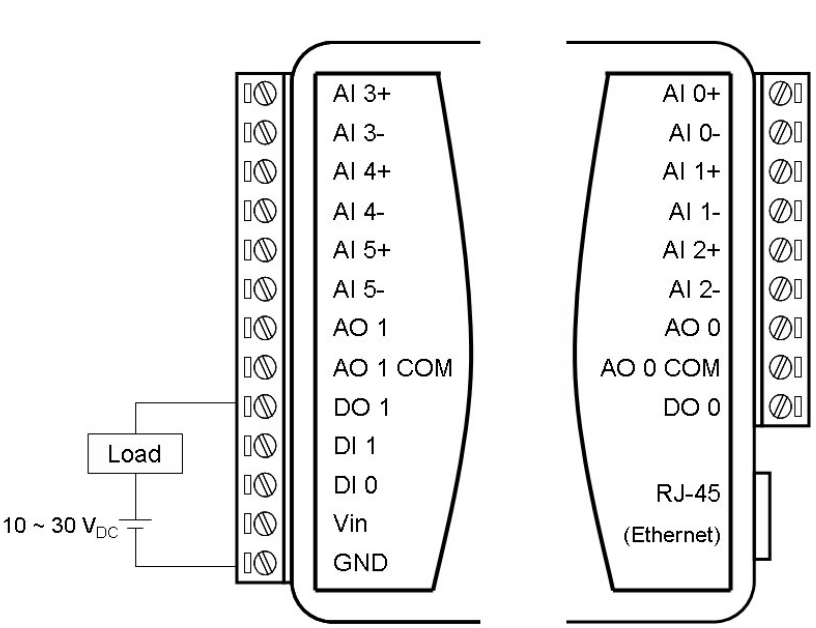


Figure 4.12: ADAM-6024 DO Wiring

Assigning Addresses for ADAM-6024 Modules

Based on the Modbus/TCP standard, the addresses of the I/O channels in ADAM-6000 modules you place in the system are defined by a simple rule. Please refer to Appendix B.2.4 to map the I/O address.

4.2 Digital I/O Modules

4.2.1 ADAM-6050

18-ch Isolated Digital I/O Module

The ADAM-6050 is a high-density I/O module built-in a 10/100 based-T interface for seamless Ethernet connectivity. It provides 12 digital input and 6 digital output channels with 2000 VDC isolation protection. All of the digital input channels support input latch function for important signal handling. Meanwhile, these DI channels allow to be used as 3 KHz counter and frequency input channels. Opposite to the intelligent DI functions, the digital output channels also support pulse output function.

ADAM-6050 Specifications

- Communication: 10/100 Base-T Ethernet
- Supports Protocol: Modbus/TCP, TCP/IP, UDP, HTTP, ICMP and ARP
- Supports Peer-to-Peer and GCL (Refer to Section 5.3.4 and Chapter 7)

Digital Input

- Channels: 12
- Dry Contact:
  - Logic level 0: Close to Ground
  - Logic level 1: Open
- Wet Contact:
  - Logic level 0: 0 ~ 3 VDC
  - Logic level 1: 10 ~ 30 VDC
- Maximum filter frequency: 8 kHz
- Supports 3 kHz counter input (32-bit + 1-bit)
- Frequency input range: 0.2 Hz~ 3 kHz
- Supports inverted DI status

Digital Output

- Channels: 6
- Sink type: Open Collector to 30 V, 100 mA (maximum load)
- Supports 5 kHz pulse output
- Supports high-to-low and low-to-high delay output

General:

- Built-in Watchdog Timer
- Isolation Protection: 2000 VDC
- Power Input: Unregulated 10 ~ 30 VDC
- Power Consumption: 2 W (max) @ 24 VDC
- Power Reversal Protection
- Operating Humidity: 20 ~ 95% RH (non-condensing)
- Storage Humidity: 0 ~ 95% RH (non-condensing)
- Operating Temperature: -10 ~ 70°C
- Storage Temperature: -20 ~ 80°C

Application Wiring

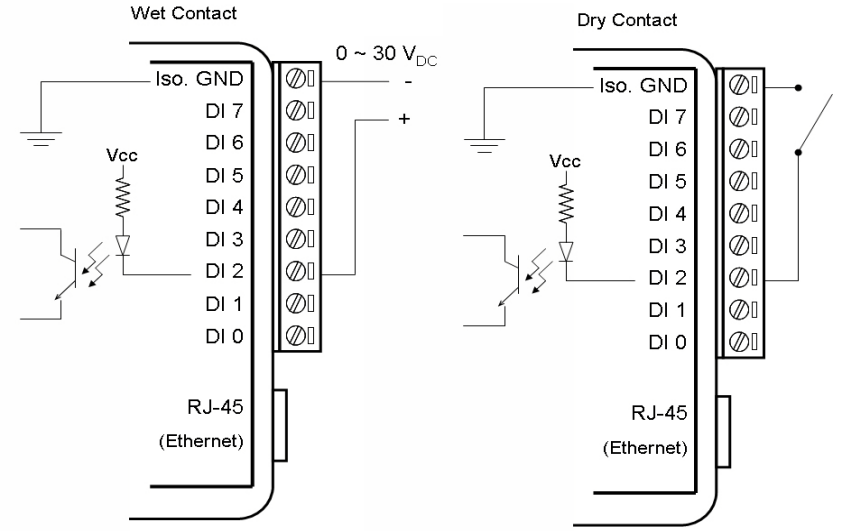


Figure 4.13: ADAM-6050 Digital Input Wiring

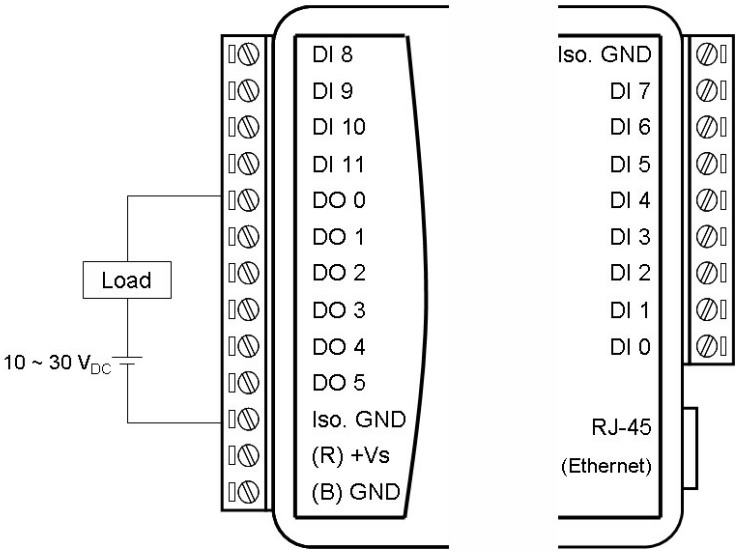


Figure 4.14: ADAM-6050 Digital Output Wiring

Assigning Addresses in ADAM-6050 Modules

Basing on Modbus/TCP standard, the addresses of the I/O channels in ADAM-6000 modules you place in the system are defined by a simple rule. Please refer to Appendix B.2.5 to map the I/O address. All digital input channels in ADAM-6050 are allowed to use as 32-bit counters (Each counter is consisted of two addresses, Low word and High word). Users could configure the specific DI channels to be counters via Windows Utility. (Refer to Section 5.3)

4.2.2 ADAM-6051

14-ch Isolated Digital Input/Output with 2-ch Counter Module

The ADAM-6051 is a high-density I/O module built-in a 10/100 based-T interface for seamless Ethernet connectivity. It provides 12 digital input, 2 digital output, and 2 counter channels with 2000 VDC isolation protection. All of the digital input channels support input latch function for important signal handling. Meanwhile, these DI channels allow to be used as 3 kHz counter and frequency input channels. Opposite to the intelligent DI functions, the digital output channels also support pulse output function.

ADAM-6051 Specifications

- Communication: 10/100 Base-T Ethernet
- Supports Protocol: Modbus/TCP, TCP/IP, UDP, HTTP, ICMP and ARP
- Supports Peer-to-Peer and GCL (Refer to Section 5.3.4 and Chapter 7)

Digital Input

- Channels: 12
- Dry Contact:

  Logic level 0: Close to Ground

  Logic level 1: Open
- Wet Contact:

  Logic level 0: 0 ~ 3 VDC

  Logic level 1: 10 ~ 30 VDC
- Supports 3 kHz counter input (32-bit + 1-bit overflow)
- Supports 3 kHz frequency input
- Supports inverted DI status

Counter Input

- Channels: 2 (32-bit + 1-bit overflow)
- Maximum count: 4,294,967,295
- Frequency range: 0.2 ~ 4500 Hz (frequency mode)

  0 ~ 4500 Hz (counter mode)

Digital Output

- Channels: 2
- Sink type: Open Collector to 30 V, 100 mA (maximum load)
- Support 5 kHz pulse output
- Support high-to-low and low-to-high delay output

General:

- Built-in Watchdog Timer
- Isolation Protection: 2000 VDC
- Power Input: Unregulated 10 ~ 30 VDC
- Power Consumption: 2 W @ 24 VDC
- Power Reversal Protection
- Operating Humidity: 20 ~ 95% RH (non-condensing)
- Storage Humidity: 0 ~ 95% RH (non-condensing)
- Operating Temperature: -10 ~ 70°C
- Storage Temperature: -20 ~ 80°C

Application Wiring

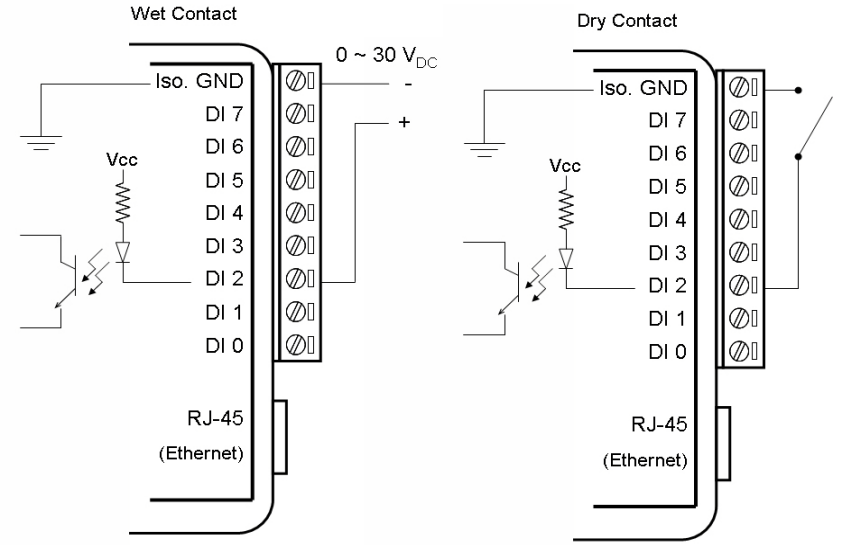


Figure 4.15: ADAM-6051 Digital Input Wiring

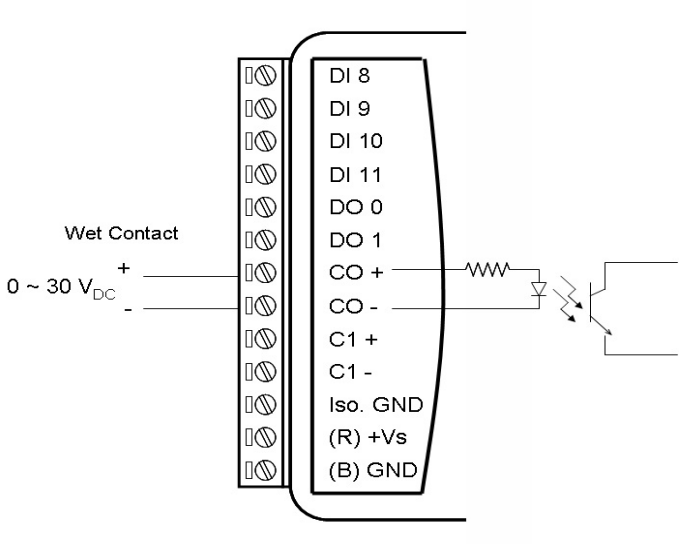


Figure 4.16: ADAM-6051 Counter (Frequency) Input

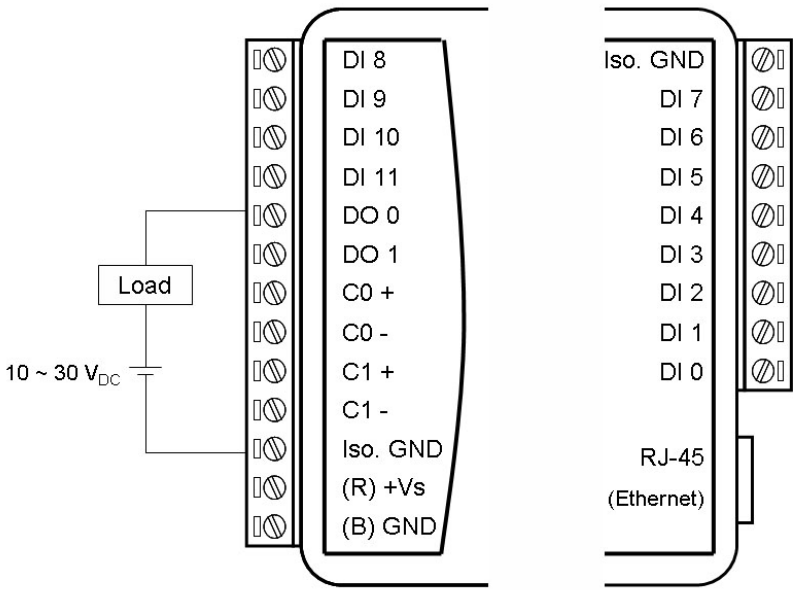


Figure 4.17: ADAM-6051 DO Wiring

Assigning Addresses in ADAM-6051 Modules

Based on Modbus/TCP standard, addresses of the I/O channels in ADAM-6000 modules are defined by a simple rule. Please refer to Appendix B.2.6 to map the I/O address. All digital input channels in ADAM-6051 are allowed to use as 32-bit counters (Each counter is two addresses, Low and High). Users could configure the specific DI channels to be counters via Windows Utility (Refer to Section 5.3).

4.2.3 ADAM-6052

16-ch Source Type Isolated Digital Input/Output Module

The ADAM-6052 is a high-density digital I/O module built-in a 10/100 based-T interface for seamless Ethernet connectivity. It provides 8 digital input, 8 digital output channels. All of the digital input channels support input latch function for important signal handling. The digital output channels support the source type output. Meanwhile, these DI channels allow to be used as 3 kHz counter and frequency input channels. Opposite to the intelligent DI functions, the digital output channels also support pulse output function.

ADAM-6052 Specifications

- Communication: 10/100 Base-T Ethernet
- Supports Protocol: Modbus/TCP, TCP/IP, UDP, HTTP, ICMP and ARP
- Supports Peer-to-Peer and GCL (Refer to Section 5.3.4 and Chapter 7)

Digital Input

- Channels: 8
- Dry Contact:

  Logic level 0: Close to Ground

  Logic level 1: Open
- Wet Contact:

  Logic level 0: 0 ~ 3 VDC

  Logic level 1: 10 ~ 30 VDC
- Supports 3 kHz counter input (32-bit + 1-bit overflow)
- Supports 3 kHz frequency input
- Supports inverted DI status

Digital Output

- Channels: 8
- Source Type: 10 ~ 35 VDC, 1 A (per channel)
- Supports 5 kHz pulse output
- Supports high-to-low and low-to-high delay output

General:

- Built-in Watchdog Timer
- Isolation Protection: 2000 VDC
- Power Input: Unregulated 10 ~ 30 VDC
- Power Consumption: 2 W @ 24 VDC
- Power Reversal Protection
- Operating Humidity: 20 ~ 95% RH (non-condensing)
- Storage Humidity: 0 ~ 95% RH (non-condensing)
- Operating Temperature: -10 ~ 70°C
- Storage Temperature: -20 ~ 80°C

Application Wiring

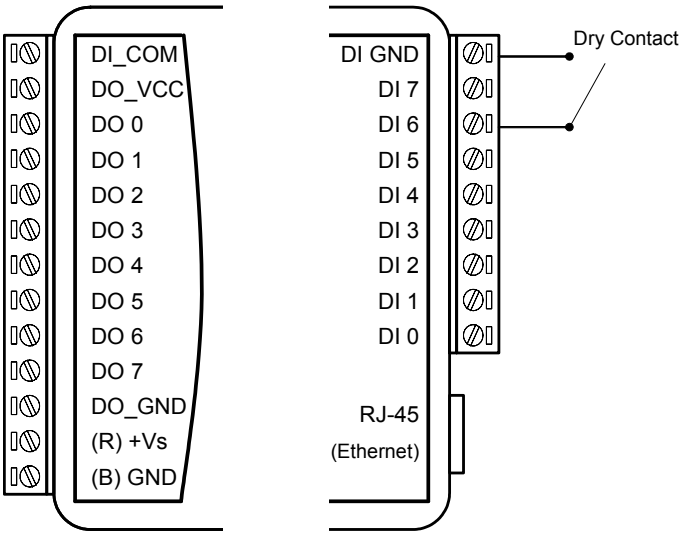


Figure 4.18: ADAM-6052 DI (Dry Contact) Wiring

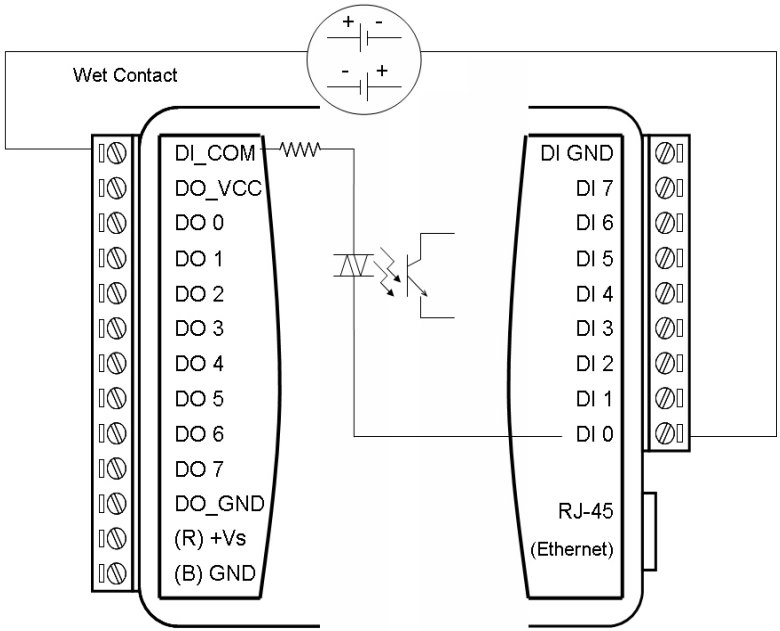


Figure 4.19: ADAM-6052 DI (Wet Contact) Wiring

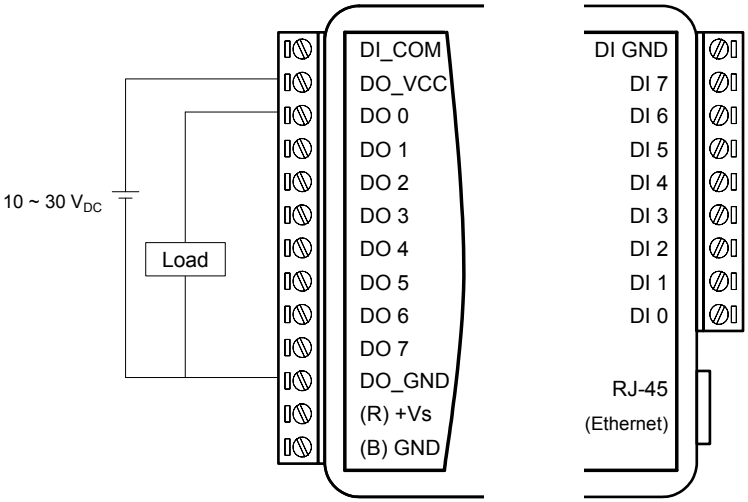


Figure 4.20: ADAM-6052 Digital Output Wiring

Assigning Addresses for ADAM-6052 Modules

Based on Modbus/TCP, the addresses of the I/O channels in ADAM-6000 modules are defined by a simple rule. Please refer to Appendix B.2.7 to map the I/O address. All digital input channels in ADAM-6052 are allowed to use as 32-bit counters (Each counter is consisted of two addresses, Low word and High word). Users could configure the specific DI channels to be counters via Windows Utility. (Refer to Section 5.3)

4.2.4 ADAM-6060

6-ch Digital Input and 6-ch Relay Module

ADAM-6060 is a high-density I/O module with a 10/100 base-T interface for seamless Ethernet connectivity. Bonding with an Ethernet port and webpage, ADAM-6060 offers 6 relay (form A) output and 6 digital input channels. It supports contact as AC 120V@0.5A, and DC 30V@1A. DI channels support input latch for signal handling, and can be used as 3 KHz counter and frequency input channels. Opposite to the intelligent DI functions, the DO channels also support pulse output.

ADAM-6060 Specifications

- Communication: 10/100 Base-T Ethernet
- Supports Protocols: Modbus/TCP, TCP/IP, UDP, HTTP, ICMP, ARP
- Supports Peer-to-Peer and GCL (Refer to Section 5.3.4 and Chapter 7)

Digital Input

- Channels: 6
- Dry Contact:
  - Logic level 0: Close to Ground
  - Logic level 1: Open
- Wet Contact:
  - Logic level 0: 0 ~ 3 VDC
  - Logic level 1: 10 ~ 30 VDC
- Maximum filter frequency: 6 kHz
- Support 3 kHz counter input (32-bit + 1-bit)
- Frequency input range: 0.2 Hz~ 3 kHz
- Support inverted DI status

Relay Output

- Channels: 6 (Form A)
- Contact rating (Resistive):AC: 120 V @ 0.5 A
  DC: 30 V @ 1 A
- Breakdown voltage: 500 VAC (50/60 Hz)
- Relay on time: 7 millisecond
- Relay off time: 3 millisecond
- Total switching time: 10 milliseconds
- Insulation Resistance: 1 GΩ minimum at 500 VDC
- Maximum Switching Rate: 20 operations/minute (at rated load)
- Electrical Endurance

At 12 V / 10 mA	Typical 5 x 10^7 operations
At 6 V / 100 mA	Typical 1 x 10^7 operations
At 60 V / 500 mA	Typical 5 x 10^5 operations
At 30 V / 1000 mA	Typical 1 x 10^6 operations
At 30 V / 2000 mA	Typical 2 x 10^5 operations

- Mechanical endurance Typical 10^8 operations
- Supports pulse output (maximum 3 Hz)

General:

- Built-in Watchdog Timer
- Group configuration
- Isolation Protection: 2000 VDC
- Power Input: Unregulated 10 ~ 30 VDC
- Power Consumption: 3 W (max) @ 24 VDC
- Power Reversal Protection
- Operating Humidity: 20 ~ 95% RH (non-condensing)
- Storage Humidity: 0 ~ 95% RH (non-condensing)
- Operating Temperature: -10 ~ 70°C
- Storage Temperature: -20 ~ 80° C

Application Wiring

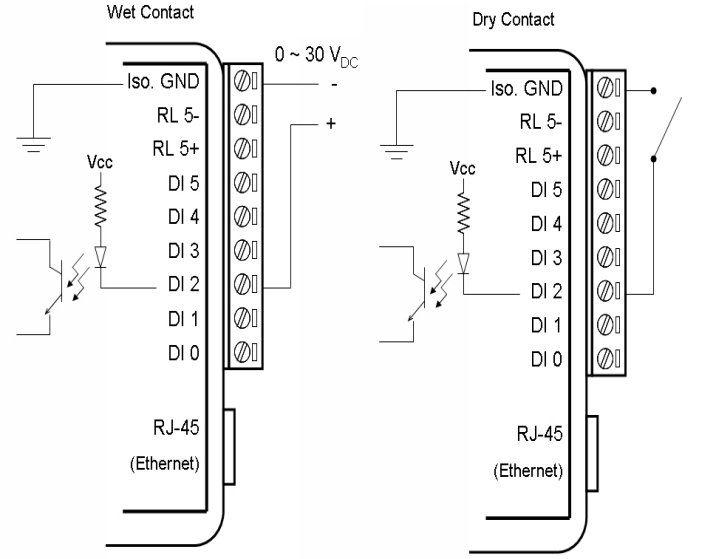


Figure 4.21: ADAM-6060 Digital Input Wiring

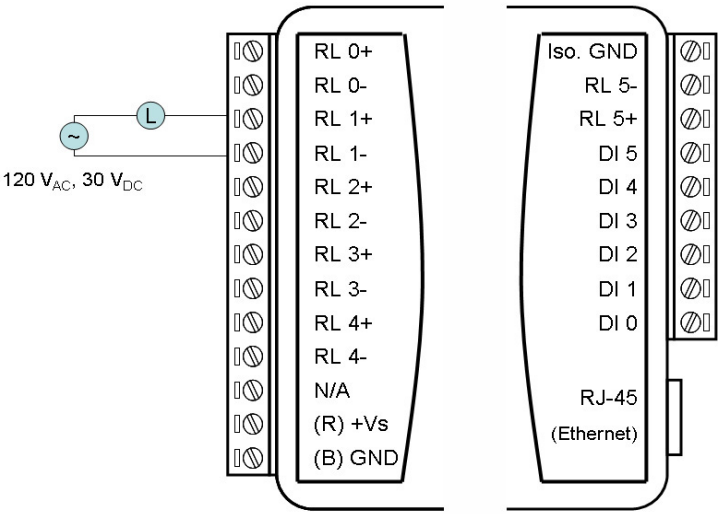


Figure 4.22: ADAM-6060 Relay Output Wiring

Assigning Addresses in ADAM-6060 Modules

Basied on Modbus/TCP standard, the addresses of the I/O channels in ADAM-6000 modules are defined by a simple rule. Refer to Appendix B.2.8 to map the I/O address. All digital input channels in ADAM-6060 are allowed to use as 32-bit counters (Each counter is consisted of two addresses, Low word and High word). Users could configure the specific DI channels to be counters via Windows Utility. (Refer to Section 5.3)

4.2.5 ADAM-6066

6-ch Digital Input and 6-ch Power Relay Module

ADAM-6066 is a high-density I/O module with a 10/100 base-T interface for seamless Ethernet connectivity. ADAM-6066 offers 6 high voltage power relay (form A) output and 6 digital input channels. It supports contact rating as AC 250V@5A, and DC 30V@3A. All of the digital input channels support input latch function for signal handling. Meanwhile, these DI channels can be used as 3 KHz counter and frequency input channels. Opposite to the intelligent DI functions, the digital output channels also support pulse output function.

ADAM-6066 Specifications:

- Communication: 10/100 Base-T Ethernet
- Supports Protocol: Modbus/TCP, TCP/IP, UDP, HTTP, ICMP and ARP
- Supports Peer-to-Peer and GCL (Refer to Section 5.3.4 and Chapter 7)

Digital Input

- Channels: 6
- Dry Contact:

  Logic level 0: Close to Ground

  Logic level 1: Open
- Wet Contact:

  Logic level 0: 0 ~ 3 VDC

  Logic level 1: 10 ~ 30 VDC
- Supports 3 kHz counter input (32-bit + 1-bit)
- Supports 3 kHz frequency input
- Supports inverted DI status

Relay Output

- Channels: 6 (Form A)
- Contact rating (Resistive): AC: 250 V @ 5 A
  DC: 30 V @ 3 A
- Breakdown voltage: 500 VAC (50/60 Hz)
- Relay on time: 7 millisecond
- Relay off time: 3 millisecond
- Total switching time: 10 milliseconds
- Insulation Resistance: 1 GΩ minimum at 500 VDC
- Maximum Switching Rate: 20 operations/minute (at rated load)
- Electrical Endurance

  At 30 VDC / 3 A Typical 1 x 10^5 operations
  (Operating frequency 20 operations/minute)

  At 250 VAC / 3 A Typical 1 x 10^5 operations
  (Operating frequency 20 operations/minute)
- Mechanical endurance Typical 2 x 10^7 operations
  (Under no load at operating frequency of 180 operations/minute)
- Supports pulse output (maximum 3 Hz)

General:

- Built-in Watchdog Timer
- Isolation Protection: 2000 VDC
- Power Input: Unregulated 10 ~ 30 VDC
- Power Consumption: 2.5 W @ 24 VDC
- Power Reversal Protection
- Operating Humidity: 20 ~ 95% RH (non-condensing)
- Storage Humidity: 0 ~ 95% RH (non-condensing)
- Operating Temperature: -10 ~ 70°C
- Storage Temperature: -20 ~ 80°C

Application Wiring

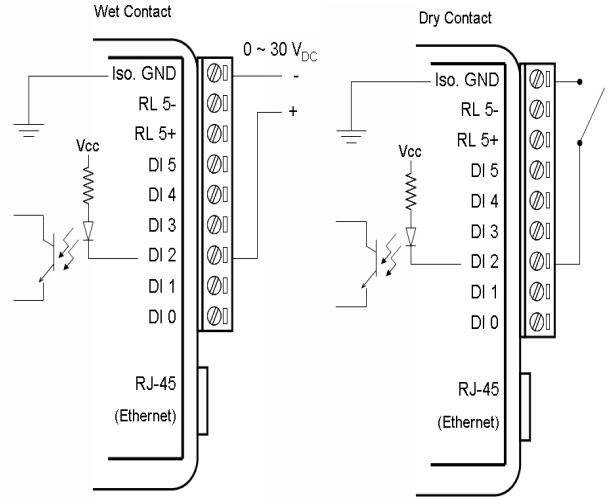


Figure 4.23: ADAM-6066 Digital Input Wiring

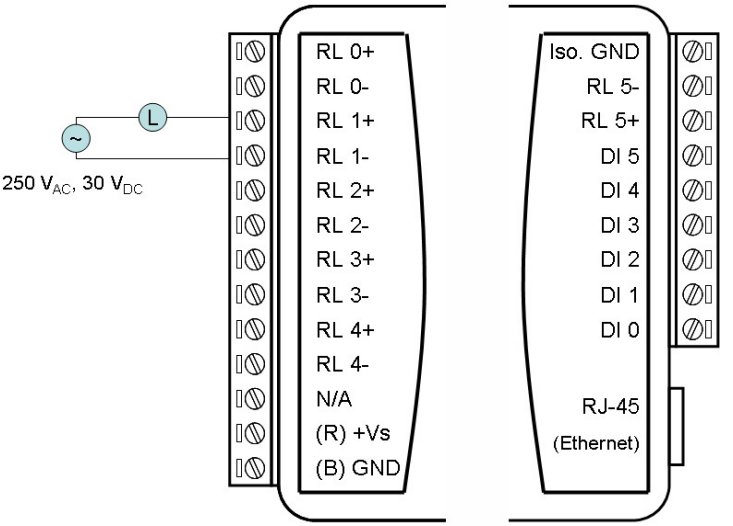


Figure 4.24: ADAM-6066 Relay Output Wiring

CHAPTER 5

System Configuration Guide

Sections include:

- System Hardware Configuration
- Install ADAM/APAX.NET Utility Software
- ADAM.NET Utility Overview
- Java Applet Customization
- Appendix A

Chapter 5 System Configuration Guide

5.1 System Hardware Configuration

As we mentioned in Chapter 3-1, you will need following items to complete your system hardware configuration.

5.1.1 System Requirements

Host Computer

- Microsoft Windows CE/XP/7
- At least 32 MB RAM
- 20 MB of hard disk space available
- VGA color monitor
- 2x or higher speed CD-ROM
- Mouse or other pointing devices
- 10/100 Mbps or higher Ethernet Card

5.1.2 Communication Interface

- 10/100 Mbps Ethernet hub (at least 2 ports) and two Ethernet cables with RJ-45 connector
- Crossover Ethernet cable with RJ-45 connector

5.2 Install ADAM.NET Utility Software

Advantech provides a free download of ADAM.NET Utility software for ADAM-6000 modules operation and configuration. You can find the Utility installation file in the CD with your ADAM module, or link to the web site: http://www.advantech.com and click into the Download Area under Service & Support site to get the latest version of the ADAM-6000 Series ADAM.NET Utility. Once you download and setup the Utility software, there will be a shortcut of the Utility program on the desktop.

Note: Before installing ADAM.NET Utility, you need to install .NET Framework 2.0 or later.

5.3 ADAM.NET Utility Overview

The ADAM.NET Utility software offers a graphical interface that helps you configure the ADAM-6000 modules. It is also very convenient to test and monitor your remote data acquisition and control system. The following guidelines will give you some brief instructions on how to use this Utility.

5.3.1 ADAM.NET Utility Operation Window

After you have successfully installed ADAM.NET Utility, there will be one shortcut icon on the desktop. Double click the shortcut icon that you should be able to see the operation window as Figure 5.1.

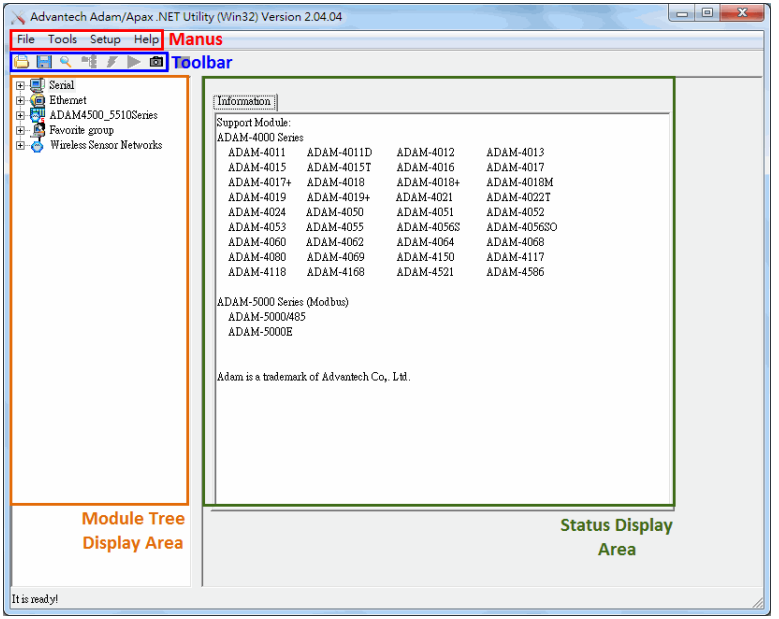


Figure 5.1: ADAM.NET Utility Operation Window

The operation window consists of four areas --- the **Menus**, the **Toolbar**, the **Module Tree Display** Area and the **Status Display** Area.

Menus

The menus at the top of the operation window contain:

File Menu:

1. Open Favorite Group - You can configure your favorite group and save the configuration into one file. Using this option, you can load your configuration file for favorite group.
2. Save Favorite Group - You can configure your favorite group and save the configuration into one file. Using this option, you can save your favorite group into one configuration file.
3. Auto-Initial Group - If you want to have the same favorite group configuration when you exit ADAM.NET utility and launch it again, you need to check this option.
4. Exit - Exit ADAM.NET Utility.

Tools Menu:

1. Search - Search all the ADAM-6000 and ADAM-5000/TCP modules you connected. The operation process will be described in Section 5.3.2.
2. Add Devices to Group - You can add ADAM-6000 modules to your favorite group by this option. You need to select the device you want to add in the **Module Tree Display** area (it will be described below) first, and then select this option to add.
3. Group Configuration - You can update Firmware, Configuration File, and HTML files of a single module/multiple modules using this option. The configuration file includes settings of Device Information, General Information, P2P & Streaming, GCL and Modbus Address XML file. The configuration file can be exported in the **Firmware** tab as a .cfg file.
4. Terminal for Command Testing - ADAM-6000 modules support ASCII command and Modbus/TCP as communication protocol. You can launch the terminal to communicate with ADAM-6000 module by these two protocol directly. (Refer to Section 6.3 and 6.4 for more information about ASCII and Modbus/TCP command.)

5. DiagAnywhere Searcher - There are multiple Advantech products installed with DiagAnywhere server, which gives user remote control ability through Ethernet. When you choose this option, all devices with DiagAnywhere server in the Ethernet you connected with will be listed.

6. Print Screen - You can save current ADAM.NET Utility screen into an image file by this option.

7. Monitor Stream/ Adam5000 Event Data - ADAM-6000 modules support Data Stream function. You can define the Host (such as a PC) by IP. Then ADAM-6000 modules will periodically send its I/O status to the Host. The IP and period to transfer data is configured in the **Stream** tab of **Status Display** area. The configuration tab will be introduced in Section 5.3.2.

Note: When you enable GCL function, Data Stream function will automatically be disabled until you disable GLC function.

8. Monitor Peer-to-Peer (Event Trigger) - ADAM-6000 modules with Peer-to-Peer function can play as Event Trigger function. Refer to Section 5.3.4 for more information. You can choose this option to receive message from ADAM-6000 module which is enabled Peer-to-Peer (Event Trigger) function.

9. Monitor GCL IO Data Message - ADAM-6000 modules with GCL function can play as a standalone controller. Users can define logic rules and run the rules on ADAM-6000 module. User can define the logic rule to send out message, depending on the logic condition, to the Host defined by IP. Refer to Chapter 7 for more information about GCL. You can choose this option to receive I/O data message from ADAM-6000 module which is enabled GCL function.

Setup Menu:

1. Favorite Group - You can configure your favorite group including add one new device, modify or delete one current device, sort current devices and diagnose connection to one device.
2. Refresh COM and LAN node - ADAM.NET utility will refresh the serial and LAN network connection situation.

3. Add COM Port Tree Nodes - This option is used to add serial COM ports in ADAM.NET Utility. You won't need to use this option for ADAM-6000 modules.
4. Show TreeView - Check this option to display the **Module Tree Display** area.

Help Menu:

1. Check Up-to-Date on the Web - Choose this option, it will automatically connect to Advantech download website.You can download the latest utility there.
2. About Adam.NET Utility - Choose this option, you can see version of ADAM.NET Utility installed on your computer.

Toolbar

There are 7 graphical icons on the toolbar for 7 common used options of Menus. Figure 5.2 below shows definition for each graphical icon.

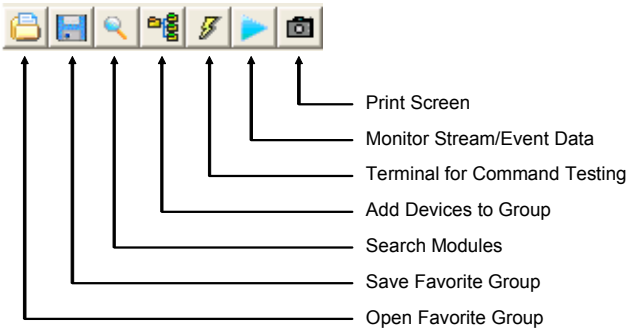


Figure 5.2: ADAM.NET Utility Toolbar

Module Tree Display Area

ADAM.NET Utility is one complete software tool that all ADAM remote I/O module and controller can be configured and operated in this utility. The **Module Tree Display** is on the left part of the utility operation window. There are four categories in the **Module Tree Display** Area:

- **ADAM4000_5000**

All serial I/O Modules (ADAM-4000 and ADAM-5000 RS-485 serial modules) connected to the host PC will be listed in this category.

- **ADAM5000TCP_6000**

All Ethernet I/O Modules (ADAM-6000 and ADAM-5000 TCP modules) connected to the host PC will be listed in this category.

- **ADAM-4500_5510Series**

This is a DOS interface utility for remote controllers such as ADAM-4500 and ADAM-5510 series.

- **Favorite Group**

You can define which devices listed in the three categories above into your personal favorite group. This will make you easier to find your interested modules. Right click on the ADAM device item under the Favorite Group item and you can select **New >> Group** to create a new group. After you create your own group, right click on your group and select **New >> Adam device** to add ADAM devices into your group. You can also select **Diagnose connection** to check the communication.

*Note: Remember to choose the correct module in the **Module Type** combo box when you add a new ADAM devices.*

Status Display Area

Status Display area, on the right part of utility operation window, is the main screen for operation. When you select different items in **Module Tree Display, Status Display** will change dependently. You can do all configurations and test in this area.

5.3.2 Search ADAM-6000 Modules

After you have confirmed the hardware wiring between host PC and your ADAM-6000 module, you can find that module in ADAM.NET Utility. Launch ADAM.NET Utility. Select the **ADAM5000TCP_6000** item on the **Module Tree Display** area. Click the **Search Modules** button on the Toolbar. ADAM.NET Utility will then search all ADAM-6000 modules on the Ethernet network. If your ADAM-6000 modules is used the first

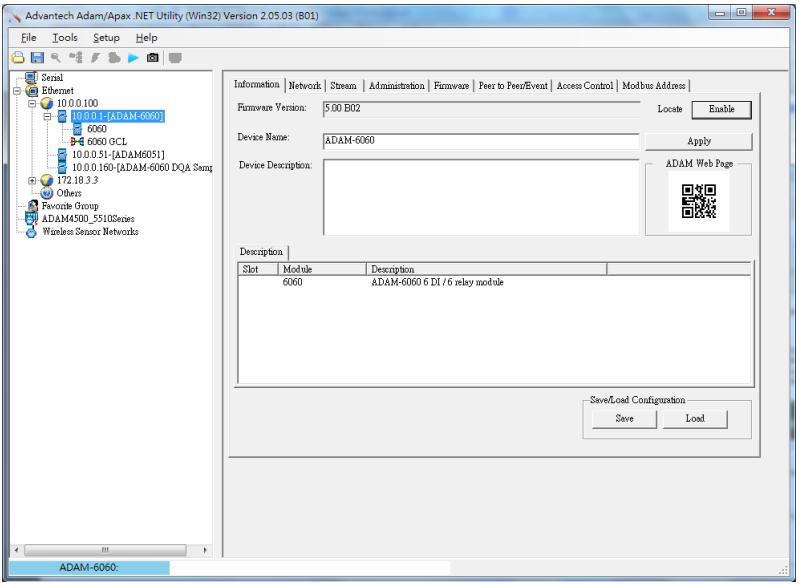
time, its IP will be 10.0.0.1 by default. So you will find it showing on the **Others** item under **ADAM5000TCP_6000.**

Note: If network Firewall is enabled on your computer, you may not be able to connect with your ADAM-6000 module. You need to add ADAM.NET Utility into lists of Program and Service of the Exception for Windows Firewall in Windows Control Panel.

You need to change IP of the ADAM-6000 modules the same subnet with the host PC. Type the correct **IP address**, **Subnet address**, and **Default gateway** on the **Status Display** area. After complete setting, click the **Apply Change** button. A dialog box appears asking you to type the password. The default password of ADAM-6000 module is 00000000.

You can change the password later. After you type the correct password, the ADAM-6000 module is now under IP of your host PC.

When you select the IP address of the ADAM-6000 modules you want use in **Module Tree Display** area, there will be 8 tabs appearing in the **Status Display** area for you to set up general configuration of that module. Refer to figure below. Once you have changed any configuration, remember to click related **Apply** or **Apply Change** button.


Advantech Adam/Apax .NET Utility (Win32) Version 2.05.03 (B01)
File
Tools
Setup
Help
Serial
Ethernet
10.0.0.100
10.0.0.1-[ADAM-6060]
6060
6060 GCL
10.0.0.51-[ADAM6051]
10.0.0.160-[ADAM-6060 DQA Samp
172.18.3.3
Others
Favorite Group
ADAM4500_5510Series
Wireless Sensor Networks
Information
Network
Stream
Administration
Firmware
Peer to Peer/Event
Access Control
Modbus Address
Firmware Version:
5.00 B02
Locate
Enable
Device Name:
ADAM-6060
Apply
Device Description:
ADAM Web Page
Description
Slot
Module
Description
6060
ADAM-6060 6 DI / 6 relay module
Save/Load Configuration
Save
Load
ADAM-6060:

Below is detailed information for the 8 tabs in the **Status Display** area:

Information

You can see the **Firmware Version** on the selected ADAM-6000 module in this tab. You also can change the **Device Name** and **Device Description**. When you have several ADAM-6000 modules in the same network, it is helpful to identify your interested ADAM-6000 modules using specific device name and device description. You can also enable or disable **Locate** function to distinguish the selected module from others in the Ethernet network, by checking Status/Link LED indicator. A **2D barcode** is generated to present the URL of the selected module's web server. Single module's configuration can be saved/loaded in the **Save/Load Configuration** area.The .xml configuration file contains settings of Network, Stream/Event data, Access Control, and IO configuration.

Network

If necessary, you still can change **IP Address**, **Subnet Address**, and **Default Gateway** of selected ADAM-6000 module in this tab. The **Host Idle** (Timeout) text box is used for TCP connection timeout. The maximum number of TCP connections of one ADAM-6000 module is 8. Any application using TCP to communicate with the ADAM-6000 module will occupy at least one connection. If there is no communication for one connection after a specific timeout interval, ADAM-6000 module will close that connection and release it to others. Without this feature, when the number of TCP connection exceeds the maximum number (namely 8), no more other application can connect to the ADAM-6000 modules.

Besides setting a fixed ID, user can choose DHCP in **IP Mode** setting if the module is connected to a network with DHCP server.

Note: When you use web browser to open the web page on Adam-6000, the JVM (Java Virtual Machine) will use several TCP connections mentioned above to download .jar file. Those connections will be released after the .jar file is downloaded completely.

Stream

ADAM-6000 modules can actively send its data to Hosts periodically. It is called *Data Stream*. In this tab, you can define IP address of the Hosts receiving the data transferred by ADAM-6000 modules, as well as the period how often ADAM-6000 modules will send data to the Hosts.

Note: Set the period by **Data Streaming** tab at right. **ADAM-5000/TCP Event Trigger** tab is for ADAM-5000.

Note: When you enable GCL function, Data Stream function will automatically be disabled until you disable GLC function.

Administration

In this tab, you can set up password for selected ADAM-6000 module. You need to type current password in the **Old password** text box, and the new password in the **New password** and **Verify password** text box. There will be many configuration and operation action asking user to type password, so this can help to ensure safety. You can reset and restart the module in the tab.

N**ote:** The default password is 00000000

Firmware

Advantech will continuously release new version of firmware to add or improve functionality of ADAM-6000 modules. You can connect to the Advantech website (http://www.advantech.com) to download the latest version of firmware. There should be three files with different file extension: .bin, .html and .jar. The file with .bin extension is the firmware itself. And the two files with .html and .jar extension are for the Web Server on the ADAM-6000 module. In this tab, you can upgrade the downloaded firmware to your ADAM-6000 module. Click the **Browse** button to load the three firmware files from your computer. Then click **Download** button to download the firmware to the ADAM-6000 module

The configuration of the ADAM-6000 module can be exported as a .cfg file. This file can be used as the source for **Group Configuration** in the **Tools Menu**. The configuration file includes settings of Device Information, General Information, P2P & Streaming, GCL and Modbus Address XML file..

*Note: When you update a new firmware to your module, some of the configurations for the 8 tabs in the **Status Display** area may be changed. We suggest you to confirm the configurations again.*

Note: ADAM-6024 doesn't support webpage upgrade (.html and .jar file).

Peer-to-Peer /Event

You can enable and configure Peer-to-Peer (Event) function in this tab. For more detail about Peer-to-Peer (Event) function, refer to section 5.3.4.

Access Control

You can decide which computers or devices have the ability to control this ADAM-6000 module in this tab. Select the **IP Address** or **MAC Address** radio button to decide the identified method, and then click the **Apply** button. In the **Security IP/MAC Setting** area, you can direct type the IP or MAC address of the authorized computers or devices. Remember to click the **Enable/Disable** check box, meaning that IP or MAC address is selected. Take Figure 5.3 as example, only the computer (or device) with IP Address 172.18.3.52 or 172.18.3.116 can have the authority to control this ADAM-6000 module. If there is no check box selected, it means there is no security limitation that any computer or device can control the ADAM-6000 modules. After completing typing all IP or MAC address, click **Apply** or **Apply all** button.

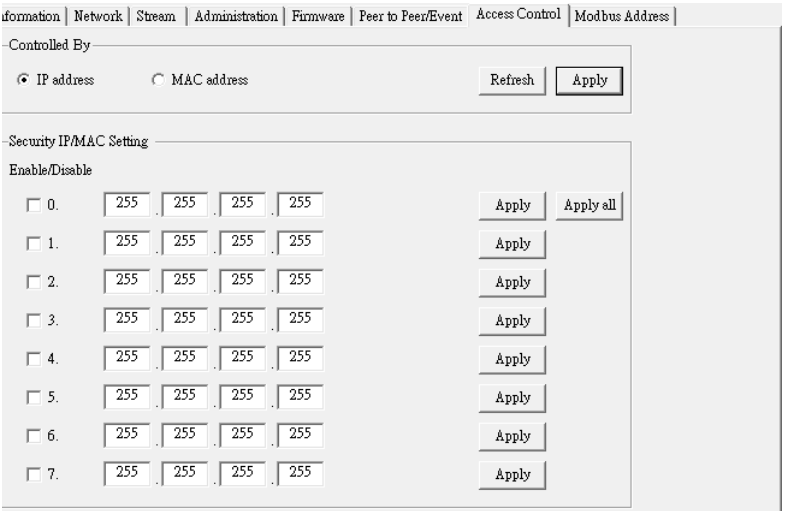


Figure 5.3: Access Control Setting

Modbus Address

You can get and set Modbus address information in this tab. By changing the address in the Base column and click **Apply** button, the Modbus address for each item can be set.

5.3.3 I/O Module Configuration

After you have completed all general configuration of ADAM-6000 module described in previous section, then you need to configure setting for input and output channel such as channel range, calibration and alarm. At the same time, you can see input channel value and set value of output channel in the **Status Display** area of utility. In the **Module Tree Display** area, click the item showing IP of the ADAM-6000 modules you want to use. There will be two items appearing below the IP: **All Channel Configuration** and **GCL Configuration** item. Refer to Figure 5.4 below. (The related feature of **GCL** item will be described in Chapter 7)

Click the cross icon besides the **All Channel Configuration** item, one dialog window will appear to ask you typing password. After you enter the correct password, **Individual Channel Configuration** items will appear below the **All Channel Configuration** item.

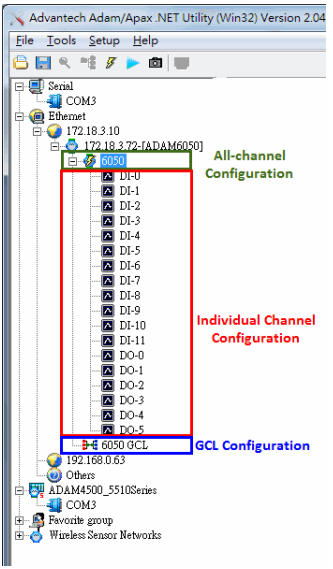


Figure 5.4: Channel & GCL Configuration

If you click the **All Channel Configuration** item, you can read analog input value or configure setting for all channels on the **Status Display** area. If you click the **Individual Channel Configuration** item, you can read AI values or configure setting for the specific channel you choose. Below, we will describe the **All Channel Configuration** and **Individual Channel Configuration** in more detail for ADAM-6000 I/O modules.

- Analog Input Module (ADAM-6015, ADAM-6017 and ADAM-6018)

All Channel Configuration

For these ADAM-6000 modules, when you click the **All Channel Configuration** item in the **Module Tree Display** area, there will be four parts on the **Status Display** area. In the top left-hand corner is the **Channels Range Configuration** area. You can set different range for each channel. In the **Channels Range Configuration** area, select the channel number in the **Channel index** combo box, and then select the range in the **Input range** combo box. After selecting appropriate range, click the **Apply** button. Refer to Figure 5.5 below.

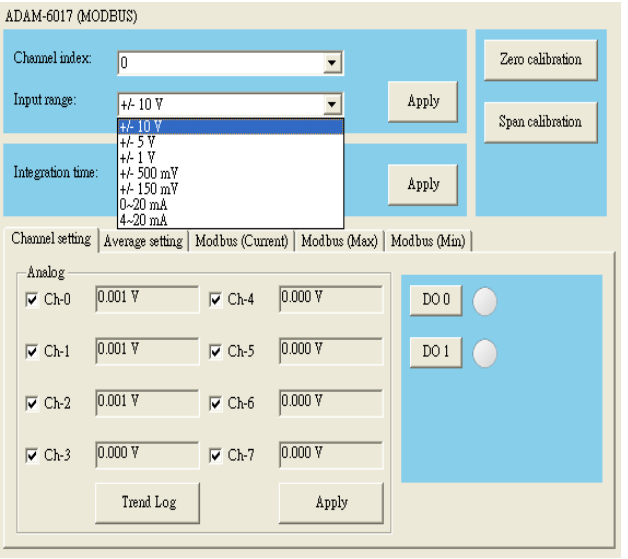


Figure 5.5: Channels Range Configuration Area

In order to remove the noise from the power supply, these analog input modules feature built-in filter. Two filters with different frequencies are provided to remove noise generated from different power supplies. The **Integration Time Configuration** area is under the **Channels Range Configuration** area. Refer to Figure 5.6 below. In the **Integration Time Configuration** area, you can select suitable filter in the **Integration time** combo box. After selecting appropriate filter, click the **Apply** button.

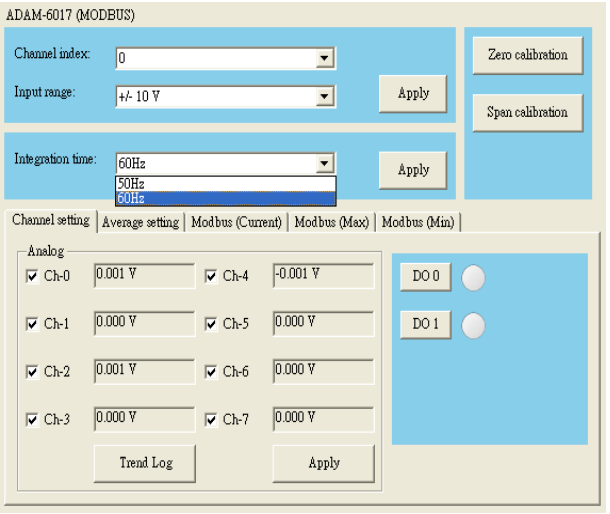


Figure 5.6: Integration Time Configuration Area

In the top right-hand corner of the **Status Display** area is the **Calibration** area. You can choose the **Zero Calibration** button to do zero calibration. After you click the button, a pop-up dialog window will remind you to connect a signal with minimum value of full scale range (for example, 0 Volt) to the calibrated channel. After you complete the hardware wiring, click the **Apply** button to start the calibration action. Similarly, you can choose the **Span Calibration** button to do span calibration. For span calibration, you need to connect a signal with maximum value of full scale range (for example, 10 Volt) to the calibrated channel. It is the same that when you complete the wiring, click the **Apply** button to start the calibration action.

At the bottom of the **Status Display** area, you can see five tabs to see analog input value of all channels:

1. **Channel Setting**

You can see the current value of analog input on this tab. (For ADAM-6017 and ADAM-6018 modules, the value of digital input channel is also displayed on this tab.) Simply choose the check box of the channels you want to monitor and click the **Apply** button.

Besides, you can see the graphical historical trend by clicking the **Trend Log** button. Refer to Figure 5.7 below. Simply choose the check box of the channels you want to log in the **Channel Setting** area at right side, and then click the **Apply** button. After that, click the **Start** button and the data log will start. You can see the real-time historical trend. If you click the **Stop** button, then you can click the **Save to file** to save the trend data into your computer.

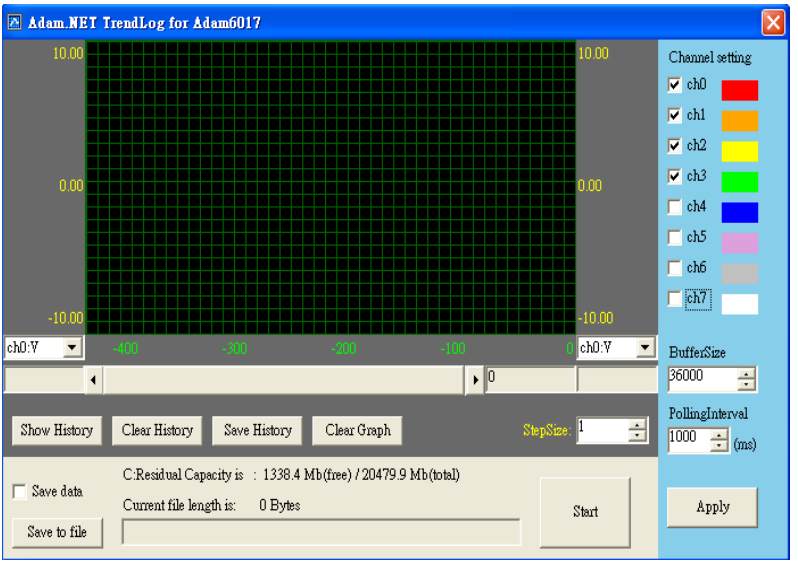


Figure 5.7: Analog Input Trend Log

With the wire burn-out detection function of ADAM-6015 and ADAM-6018, if there is no sensor connected to the input channel of ADAM-6015 or ADAM-6018 module, you can see “Burn out” characters showing in the text box of related channel.

2. **Average Setting**

ADAM-6015, ADAM-6017 and ADAM-6018 modules feature averaging calculation function by its built-in processor. You can simply click the check boxes representing the channels in the **Averaging channel setting** area to decide which channels are used for averaging. For example, by Figure 5.8 below, five channels (channel 0, 1, 2, 3, and 4) are used for averaging. So you can see the average value of those five channels displayed by the **Average** text box.

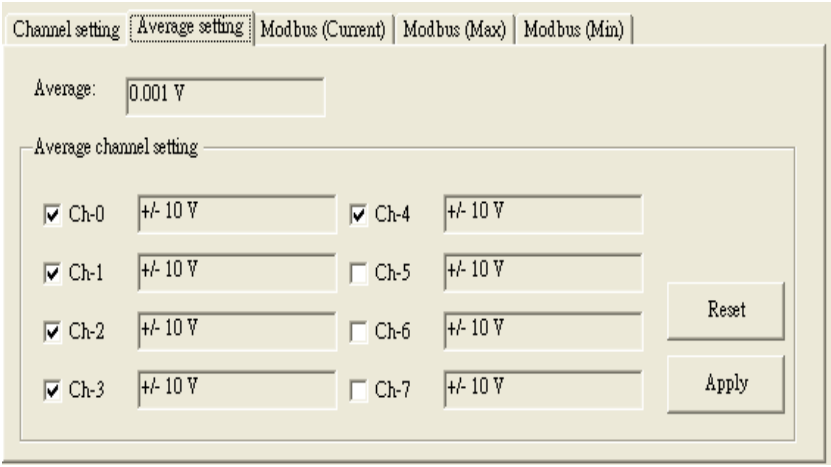


Figure 5.8: Analog Input Average Setting

3. **Modbus (Current)**

You can see current analog input value in decimal, hexadecimal, and engineer unit for all related Modbus address.

4. **Modbus (Max)**

ADAM-6015, ADAM-6017 and ADAM-6018 modules feature historical maximum value recording. You can see historical maximum analog input value in decimal, hexadecimal, and engineer unit for all related Modbus address. To re-initialize the recording, click the buttons representing the channels you want to reset.

5. **Modbus (Min)**

ADAM-6015, ADAM-6017 and ADAM-6018 modules feature historical minimum value recording. You can see historical minimum analog input value in decimal, hexadecimal, and engineer unit for all related Modbus address. If you want to re-initialize the recording, click the buttons representing the channels you want to reset.

Individual Channel Configuration

You can see analog input value and configure setting for each channel. Simply click one of the **Individual Channel Configuration** items for the interested channel. (The average channel you set in the **Averaging setting** will also be displayed here) At the upper part of the **Status Display** area, you can see the current analog input value and defined range of that channel by the **Input value** and **Input range** text box. Refer to Figure 5.9 below.

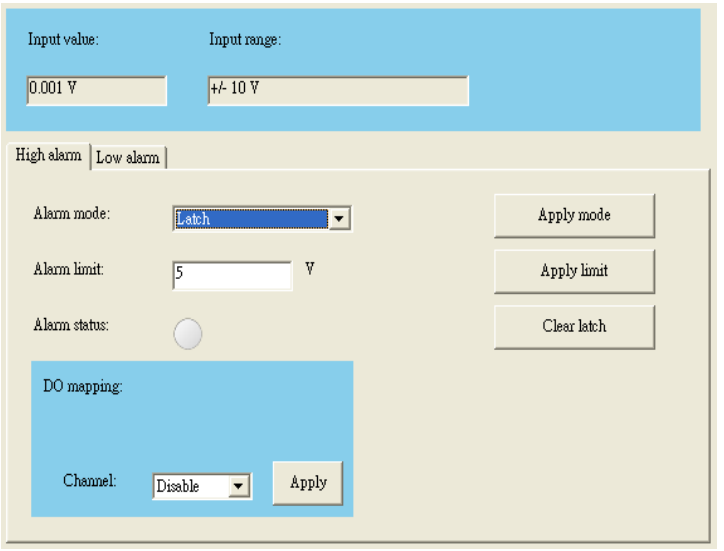


Figure 5.9: Analog Input Alarm Mode Configuration

ADAM-6015, 6017 and 6018 modules all feature built-in alarm function. At the lower part of the **Status Display** area, there are two tabs to configure the high alarm and low alarm for the selected channel: **High alarm** and **Low alarm**. When the analog input value is higher than the high alarm value, or lower than the low alarm value, the *alarm condition occurs*. Then the alarm status will be activated to logic high. For ADAM-6015 module, ADAM.NET Utility can detect the alarm status and show it by the **Alarm status** LED display. For ADAM-6017 and ADAM-6018 module, when the alarm condition occurs, the **Alarm status** LED display will be lit. Besides, the specified digital output channel will generate logic high value if you build the mapping relationship between alarm and DO channel in the **DO mapping** area. You can set the DO channel by **Channel** combo box in the **DO mapping** area. After choosing the interested channel, click the **Apply** button.

There are three alarm modes. You can select the alarm mode by the **Alarm mode** combo box for the low alarm and high alarm respectively.

1. Disable: Alarm is disabled. So even when the alarm condition occurs, nothing will happen.
2. Latch: Once the alarm condition occurs, the alarm status will be activated to logic high level and will keep the value until the alarm is clear manually. Before the value is clear, the **Alarm status** LED will continuously be lit. For ADAM-6017 and ADAM-6018 module, the specific output channel (chosen in the **DO mapping** area) will continuously generate logic high value. You can clear the alarm by click the **Clear latch** button.
3. Momentary: The alarm status will dynamically change depends on if the alarm condition occurs. If the alarm condition occurs, the alarm status will be logic high. If the alarm condition disappears, the alarm status will be logic low. So not only the **Alarm status** LED in the utility but also the specific digital output channel value will change depend on the alarm condition.

After you choose the alarm mode for high alarm or low alarm, click the **Apply mode** button. Then you can define the high alarm value or low alarm value by entering the value in **Alarm limit** text box. After you enter the alarm value, click the **Apply limit** button. Once you have configured the alarm mode and alarm value, you can leverage ADAM-6000 analog input alarm function.

- Universal Input and Output Module (ADAM-6024)

All Channel Configuration

ADAM-6024 module features analog input, analog output, digital input and digital output. Click the **All Channel Configuration** item. In the **Status Display** area, there will be two tabs: **Input** and **Output**. On the **Input** tab, there are still four parts on the **Status Display** area, which is the same as ADAM-6015, ADAM-6017 and ADAM-6018 module. All the configurations in the **Channels Range Configuration**, **Integration Time Configuration** and **Calibration** areas are just the same as the configuration of ADAM-6015, ADAM-6017 and ADAM-6018 module. Refer to Figure 5.10 below.

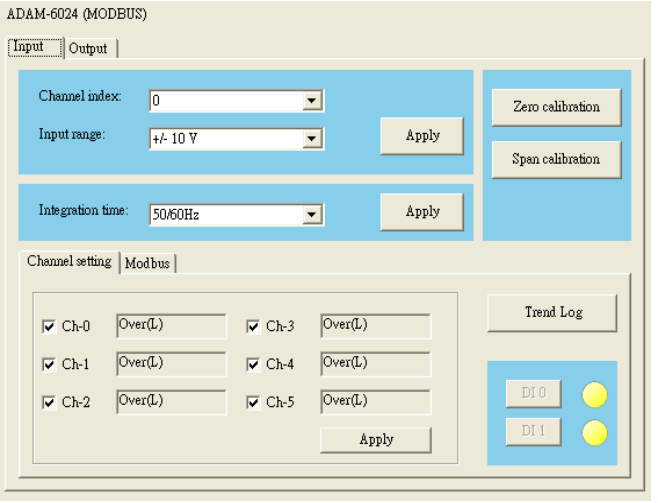


Figure 5.10: ADAM-6024 Input Tab

However, at the bottom of the **Status Display** area, there are only two tabs to see analog input value of all channels: (This is because ADAM-6024 doesn't feature averaging, maximum and minimum calculation function)

1. **Channel Setting**

You can see the current value of analog input on this tab. Choose the check box of the analog input channels you want to monitor and click the **Apply** button. If the analog input value is out of the input range, you will see "Over(L)" in the analog input value text box. At the right side, you can see current digital input value by **DI 0** and **DI 1** LED display.

You also can see the graphical historical trend of analog input channel by clicking the **Trend Log** button. All the operations for trend logging is the same as ADAM-6015, ADAM-6017 and ADAM-6018 module.

2. **Modbus**

You can see current analog input value in decimal and hexadecimal for all related Modbus address.

On **Output** tab, you can write value to analog and digital output channel, as well as configure all related setting. There are also two tabs on the Output tab: **Channel setting** and **Modbus**. Refer to Figure 5.11 below.

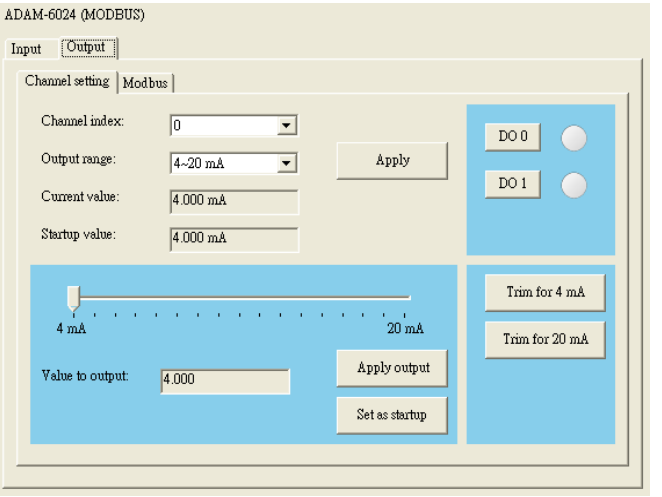


Figure 5.11: ADAM-6024 Output Tab

1. Channel Setting

Similar to analog input configuration, you can set different range for each analog output channel. Select the channel number in the **Channel index** combo box, and then select the range in the **Output range** combo box. After selecting appropriate range, click the **Apply** button.

At the bottom left-hand corner of the **Output** tab, you can define the analog output value by using the horizontal pointer slide or the **Value to Output** text box. After you have chosen the appropriate analog output value, click the **Apply output** button.

Besides, you can set the startup value of the specific analog output channel. (The analog output channel will generate the startup value output when it is power-on. In other words, start-up value can be considered as a power-on value.) Use the horizontal slider or the **Value to output** text box to define the value and click **Apply** button. Then click the **Set as startup** button to save that value as startup value.

At the bottom right-hand corner of the **Output** tab, it is **Calibration** area. There are two buttons used to calibrate the maximum and minimum value of full range. The label on the buttons will change depending on the output range. Take figure 5.11 as example, labels on the two buttons are **Trim for 4 mA** (calibrate for minimum value of full range) and **Trim for 20 mA** (calibrate for maximum value of full range).

After you click one of the buttons, one dialog window will appear. Use another instrument to measure the output value. Then use the four buttons (**-10**, **-1**, **+1**, **+10**) on the dialog window to correct the analog output value. For example, if you click the **Trim for 4 mA** button, the specific analog output channel should generate 4 mA. However, the instrument reads 3.88 mA. So you need to use **+1** and **+10** button to adjust the output value, until the output value is truly 4 mA.

In the top right-hand corner of the **Output** tab, you can control the digital output value by the **DO 0** and **DO 1** button. Their value will be display by the LED near the button.

2. Modbus

You can see current output value in decimal and hexadecimal for all related Modbus address.

- Digital Input and Output Modules

(ADAM-6050, ADAM-6051, ADAM-6052, ADAM-6060, ADAM-6066)

All Channel Configuration

When you click the **All Channel Configuration** item in the **Module Tree Display** area, there will be two tabs: **Channel Setting** and **Modbus**. Take ADAM-6050 as example. Refer to Figure 5.12 below:

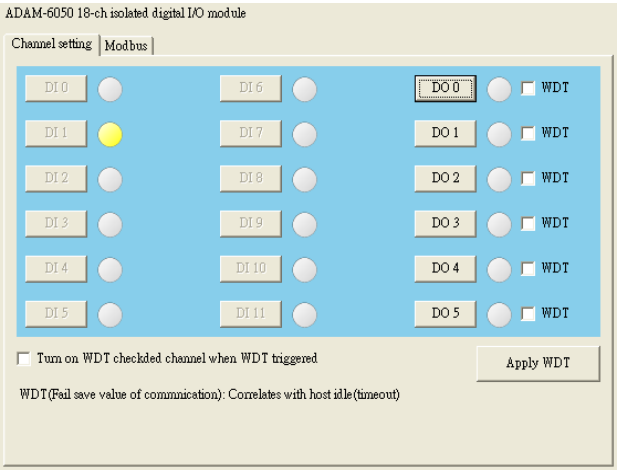


Figure 5.12: ADAM-6050 Channel Setting

1. Channel Setting

You can see value of all digital input channels by related LED display in this tab. Besides, you also can control values of all digital output channels by related button. The LED next to the button will display current value of that digital output channel.

When the communication between host PC and ADAM-6000 digital modules is broken, the digital output channel can generate a predefined value (this value is called *fail safe value*). You can enable or disable this function by click the **Turn on WDT checked channel when WDT triggered** check box. You can define the fail safe value by the **WDT** check box next to the DO status LED. After configure related setting, click the **Apply WDT** button. Take Figure 5.13 below as example. The fail safe value function is enabled. If the communication between ADAM-6000 digital module and host PC is broken, channels 0 and 2 will automatically generate logic high value, while channels 1, 3, 4, 5 will automatically generate logic low value.

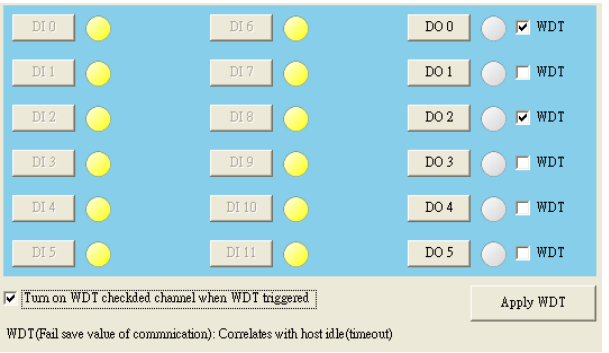


Figure 5.13: Fail Safe Value Configuration

2. Modbus

You can see current digital input or digital output values for all related Modbus address.

Individual Channel Configuration

You can see digital input value and configure setting for each digital input channel. It is the same that you can control the digital output value and configure setting for each digital output channel. Simply click the channel interested in the **Individual Channel Configuration** items.

If you choose a digital input channel, the **Status Display** area should look similar to Figure 5.14 below.

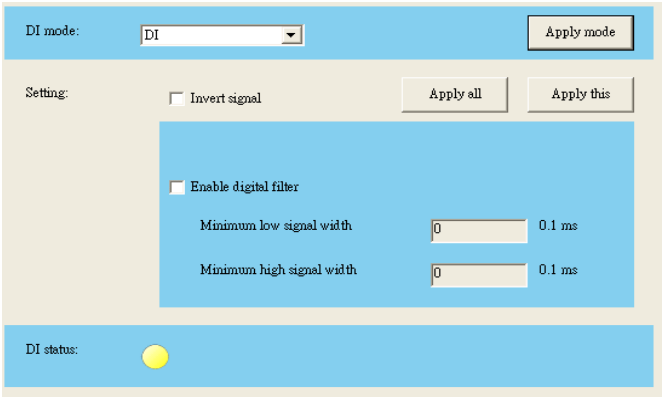


Figure 5.14: Individual Channel Configuration: DI

You can choose different mode for that digital input channel by choosing the **DI mode** combo box at top of **Status Display** area. (You should choose the appropriate mode depending on the hardware specification.) After you have chosen the mode, click the **Apply mode** button. There are a total of five possible DI modes you can choose:

1. DI

Figure 5.14 is the image when you choose DI mode. At the bottom of the **Status Display** area, you can see the digital input value by **DI status** LED display. If the digital module you are using supports Invert DI Status function, there will be **Invert signal** check box in the **Setting** area. You can click the check box to enable or disable that function. Remember to click the **Apply all** button for all channels or **Apply this** button for this specific channel to complete the configuration. When you enable the Invert DI Status function, the ADAM-6000 digital module will automatically inverse the digital input value. For example, if the real external signal value is logic level low, then the **DI status** LED display will be lit.

All ADAM-6000 digital modules support digital filter, so you can enable or disable the filter by click the **Enable digital filter** check box. If you enable the filter, you can define the minimum acceptable signal width by the **Minimum low signal width** and **Minimum high signal width** text box. (Unit: ms) The high frequency noise will be removed by this filter. Remember to click the **Apply all** button for all channels or **Apply this** button for this specific channel to complete the configuration.

2. Counter

When you choose Counter mode, one counter will count the pulse number of the digital signal from the selected channel, and then record the count number in the register. The image of the **Status Display** area looks similar as that of DI mode. At the bottom of the **Status Display** area, current count value of the selected channel is displayed by the **Counter value** text box. You can start or stop the counter to count by clicking the **Star**/**Stop** button next to the **Counter value** text box. You also can reset the counter (the value in the register will be initialized to zero) by clicking the **Clear** button.

Like the DI mode, you can enable/disable the Invert DI Status function and digital filter in the **Setting** area. The operation is the same. There is one extra setting that you can define if the counter should keep the last value when ADAM-6000 digital module powers off. If you enable this function, when the digital module powers off, the last value of counter will be kept in the register. As the module powers on, the counter will continuously count from that value. Without this function, when the module powers off, the counter will reset and the count value in the register will be zero. You can enable or disable this function by clicking the **Keep last value when power off** check box. Remember to click the **Apply all** button for all channels or **Apply this** button for this specific channel to complete the configuration.

3. Low to High Latch

When you choose Low to High Latch mode, once the digital input channel detects logic level changes from low to high, the logic status will be keep as logic high. The logic status will remain the logic high, until you clear latch manually. Then the logic status will back to logic low. The logic status can be seen by the **Latch status** LED display at the bottom of the **Status Display** area. You can clear latch by clicking the **Clear latch** button. It is the same as DI mode that you can enable or disable the Invert DI Status function in the **Setting** area. Remember to click the **Apply all** button for all channels or **Apply this** button for this specific channel to complete the configuration.

4. High to Low Latch

When you choose High to Low Latch mode, once the digital input channel detects logic level changes from high to low, the logic status will be keep as logic low. The logic status will remain the logic low, until you clear latch manually. Then the logic status will back to logic high. The logic status can be seen by the **Latch status** LED display at the bottom of the **Status Display** area. You can clear latch by clicking the **Clear latch** button. It is the same as DI mode that you can enable or disable the Invert DI Status function in the **Setting** area. Remember to click the **Apply all** button for all channels or **Apply this** button for this specific channel to complete the configuration.

5. Frequency

When you choose Frequency mode, ADAM-6000 digital module will calculate the frequency value of the digital input signal from the selected channel. The frequency value will be displayed by the **Frequency value** text box at the bottom of the **Status Display** area.

If you choose a digital output channel in the **Individual Channel Configuration** items, the **Status Display** area should look similar to Figure 5.15 below.

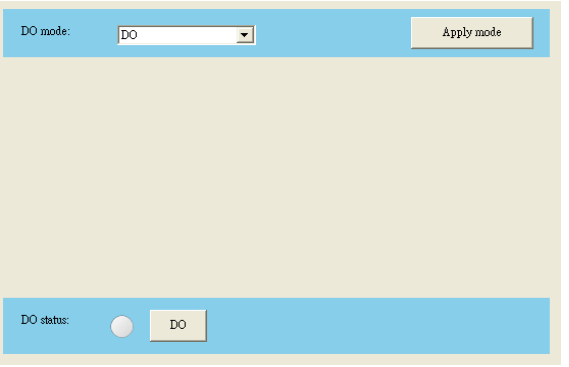


Figure 5.15: Individual Channel Configuration: DO

You can choose different mode for that digital output channel by choosing the **DO mode** combo box at top of **Status Display** area. (You should choose the appropriate mode depending on the hardware specification.) After you have chosen the mode, click the **Apply mode** button. There are totally four possible DO modes you can choose:

1. DO

Figure 5.15 is the image when you choose DO mode. You can control the digital output value of the selected channel by the **DO** button. The current digital output value will be shown by the **DO status** LED display.

2. Pulse Output

The pulse output is the same as PWR. After you choose the Pulse output mode, the selected digital output channel can generate continuous pulse train or finite pulses. You can define the pulse width by entering into the **Low signal width** and **High signal width** text box in the **Setting** area. (Unit: 0.1 ms) The frequency and duty cycle of the pulse output signal will be calculated automatically and displayed by the **Output frequency** and **Duty cycle** text box. After you complete the setting, click the **Apply change** button. Then you can choose to generate continuous pulse train or finite pulses by selecting the **Continuous** (for pulse train) or the **Fixed total** (for finite pulses) radio button. The text box at the right hand of the **Fixed total** button is used to define how many pulses you want to generate. After select the pulse output mode, click the **Start** or **Stop** button to generate or stop the pulse output.

3. Low to High Delay

When you choose Low to High delay mode, it is almost the same as choosing the DO mode. The only difference is that there will be certain time delay when the output value changes from logic low to logic high. Refer to Figure 5.16 below for its process. You can define the delay time by entering its value into the **Delay time** text box in the **Setting** area. After you complete the setting, click the **Apply** button. Then you can control the digital output value by the **DO** button and see its current value by the **DO status** LED display at the bottom of the Status Display area.

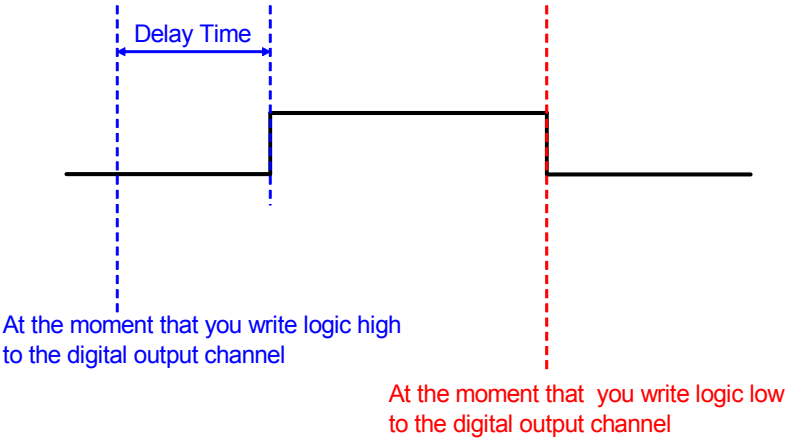


Figure 5.16: Low to High Delay Output Mode

4. High to Low Delay

When you choose High to Low delay mode, it is almost the same as choosing the DO mode. The only difference is that there will be certain time delay when the output value changes from logic high to logic low. Refer to Figure 5.17 below for its process. You can define the delay time by entering its value into the **Delay time** text box in the **Setting** area. After you complete the setting, click the **Apply** button. Then you can control the digital output value by the **DO** button and see its current value by the **DO status** LED display at the bottom of the Status Display area.

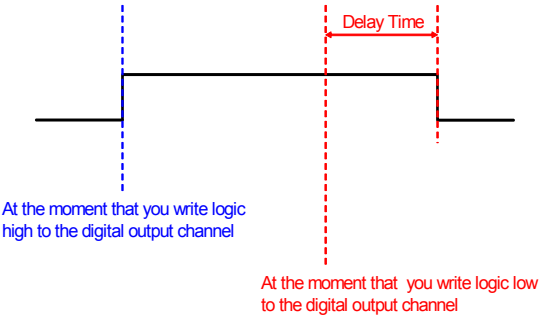


Figure 5.17: Low to High Delay Output Mode

5.3.4 Peer-to-Peer Function

• What is Peer-to-Peer?

When you want to send a signal from one module to another module, Peer-to-Peer is a perfect solution. With Peer-to-Peer function enabled, ADAM-6000 modules can actively update its input value to other devices such as PC or another ADAM-6000 module. One typical application is using a pair of ADAM-6000 modules. The value of input channel on one module will be automatically updated to output channel on another module. The data will be transferred automatically as long as the connection between the two ADAM-6000 modules is already built. No controller is needed to take care of the communication. ADAM-6000 modules feature two types of Peer-to-Peer function:

Note: Please use Ethernet Switch between a pair of Peer-to-Peer modules. Do not use an Ethernet hub. This can prevent data packet collision.

Note: ADAM-6000 modules support 2 features: Peer-to-Peer (Event) and GCL (GCL will be introduced in Chapter 7). You cannot enable these two features at the same time. So if you has enabled GCL function before, and want to use Peer-to-Peer (Event) function now, you need to disable GCL function first. (See Section 7.2 for how to disable GCL)

Note: To utilize Peer-to-Peer function, you need to upgrade firmware version of your ADAM-6000 module to 3.x or later.

1. Basic Mode:

For basic mode, there will be only ***one target device*** to receive the data transferred from one ADAM-6000 module (Module A). Usually the target device is another ADAM-6000 module (Module B). The input channels of Module A will be mapping to the output channels of module B. Meanwhile, value of all the input channels of module A will automatically update to output channels of module B. Of course, you can define mask to disconnect relationship between some input and output channels.

2. Advanced Mode:

For advanced mode, there will be ***multiple target devices*** to receive the data transferred from one ADAM-6000 module (Module A). For example, there can be several ADAM-6000 modules receiving data from the Module A. You can define different target devices (by different IP address) to each channel of module A. For example, you can define the input channel 1 of Module A is mapping to the output channel 3 of Module B, while input channel 2 of Module A is mapping to the output channel 4 of Module C. So value of input channel 1 and 2 on Module A will automatically update to channel 3 on Module B and channel 4 on Module C, respectively. Refer to the figures 5.18 and 5.19 below, and you will more clearly understand the difference between these two modes.

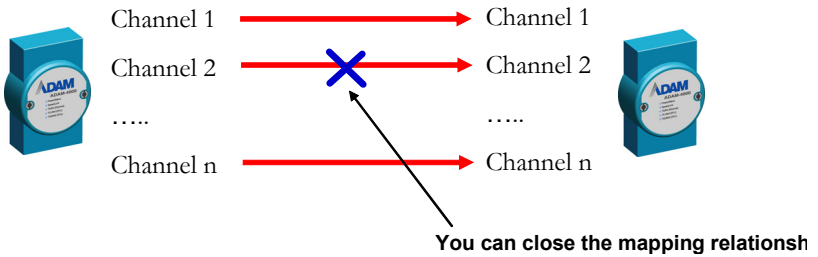


Figure 5.18: Basic mode for Peer-to-Peer

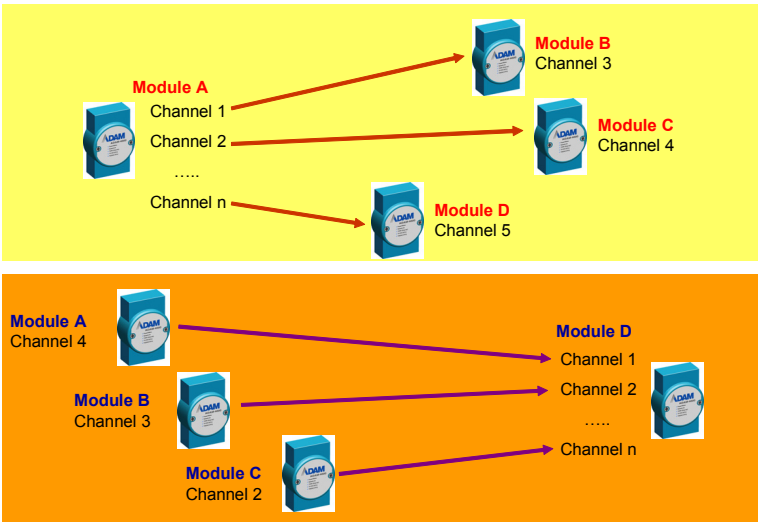


Figure 5.19: Advanced mode for Peer-to-Peer

As for when the data will be updated from one ADAM-6000 module to its target devices, there are also two options to choose:

1. Period Time Function:
The value of the input channel will be updated to the target devices with the defined period.

2. Period Time Function + C.O.S (Change of Status) Function:
The value of the input channel will still be updated to the target devices with the defined period. Moreover, when C.O.S happens (the change of the analog input value is greater than specific deviation or digital input status changes), the value of the input channel will also update to the target devices immediately.

• How to use Peer-to-Peer function to implement Event Trigger

In many applications, the data will only be sent to a host computer when specific event happens. Typical event is that the digital or analog signal changes. To implement this kind of application, ADAM-6000 modules enabled with Peer-to-Peer function is a perfect solution.

The target device of Peer-to-Peer can be a computer, simply by entering the IP of that computer into the **Destination** text box of **Peer-to-Peer/ Event** configuration tab in ADAM.NET Utility. (The detail information about configuration is described below.) Choose Basic mode and Period Time function + C.O.S. function as communication method.

There should be one program running on the host computer to receive the data, and we provide an example C program (VC++ 6.0) in the CD with ADAM-6000 module. Although the ADAM-6000 modules will send data to the host computer periodically (the reason is for communication security, see Note below), you can still distinguish the message is sent from Period Time function or C.O.S. function. The message contains which channels has changed. So if you find there is no change for all channels in the message information, then you can realize that there is no event happening.

*Note: There will be uncertainty for network communication. Sometimes there might be packet lost when event occurs. This is the reason we **Period Time function** + **C.O.S. function** (no **C.O.S. function** only). When event occurs, even if the packet is lost, the data will be sent again when the next period reaches. This can help to make the system more reliable.*

- How to configure Peer-to-Peer functions

As we have mentioned in section 5.3.2, when you select the IP address of the ADAM-6000 modules you want use in **Module Tree Display** area, there will be 8 tabs appearing in the **Status Display** area for you to set up general configurations of that module. You can configure all Peer-to-Peer function setting in the **Peer-to-Peer/Event** tab. Refer to Figure 5-20 below to see the image when you choose the **Peer-to-Peer/Event** tab.

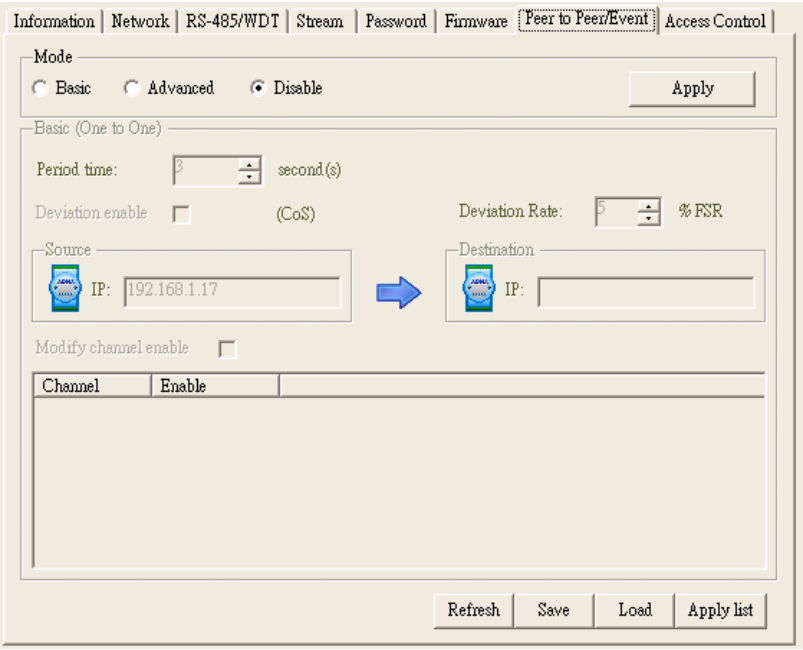


Figure 5.20: Peer-to-Peer Configuration Tab

The Peer-to-Peer function is disabled by default. You can enable it and choose the basic mode or advanced mode by click the **Basic** or **Advanced** radio button in the **Mode** area. After you choose the mode, click the **Apply** button. ADAM-6000 modules features Peer-to-Peer and GCL function in the same hardware. (The GCL feature will be introduced in Chapter 7) However, only one of them can be enabled at one time. If you have enabled GCL function before and now choose to enable Peer-to-Peer function, there will be one warning message asking you to disable GCL function first. (Refer to section 7.2 for how to disable GCL). After GCL function is disable, you can select Basic or Advanced mode for Peer-to-Peer function.

- Basic Mode Configuration

When you choose the basic mode, the Status Display should look like the Figure 5.21 below. You can define the target device by entering its IP address into the **Destination** text box in the **Basic (One to One)** area.

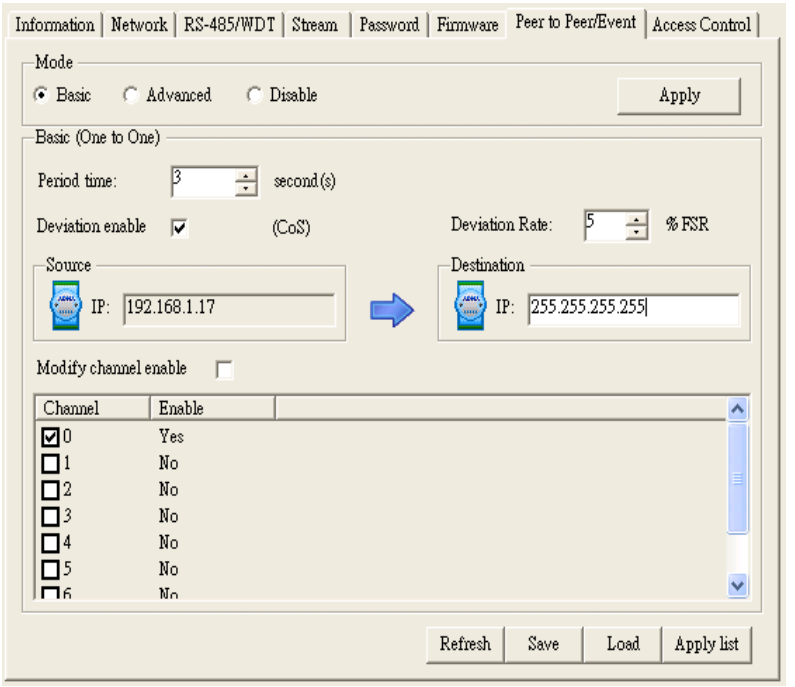


Figure 5.21: Peer-to-Peer Basic Mode Configuration

Note: FSR represents "Full Scale Range"

We have mentioned that there are two methods to transfer data from the ADAM-6000 (source) to the target device (destination): **Period Time function** or **Period Time function** + **C.O.S. function**. You can choose these two methods by click the **Deviation Enable** check box (for AI modules) or **Enable Change of State** check box (for digital modules).

If this check box is not checked, the transfer method is **Period Time function**. The period to transfer data from source to destination is defined by the **Period time** numeric control in the **Basic (One to One)** area.

If the check box is checked, the transfer method becomes **Period Time function + C.O.S. function**. You can define the deviation for analog input by the **Deviation Rate** numeric control (value is percentage unit and represent the change value divided by the total range).

For **Period Time function + C.O.S function**, the data will be transferred from source to destination periodically. Besides, when the analog input value change is greater than what the **Deviation Rate** defined or the digital input channel value changes, the data will also update from source to destination automatically.

By default, all input channels of the source module will all be mapping to all output channels of the destination module. However, you can manually define which input channels are mapping to output channels, by clicking the **Modify channel enable** check box. When this check box is checked, you can select which input channels will be mapping to the corresponding output channels by click related **Channel** check box. Refer to Figure 5.22 below. In this example, only the value of input channels 0, 1, 2, 3 of the source module will update to the output channels 0, 1, 2, 3 of the destination module. After you have selected the channel, click the **Apply list** button to download this configuration to the source module. You can save current mapping relation into a configuration file in your computer by clicking the **Save** button. You also can load previous mapping configuration file by clicking the **Load** button. If you click the **Refresh** button, the current mapping configuration on the source module will be displayed in the Channel-Enable table. (The area enclosed by the red square in Figure 5.22)

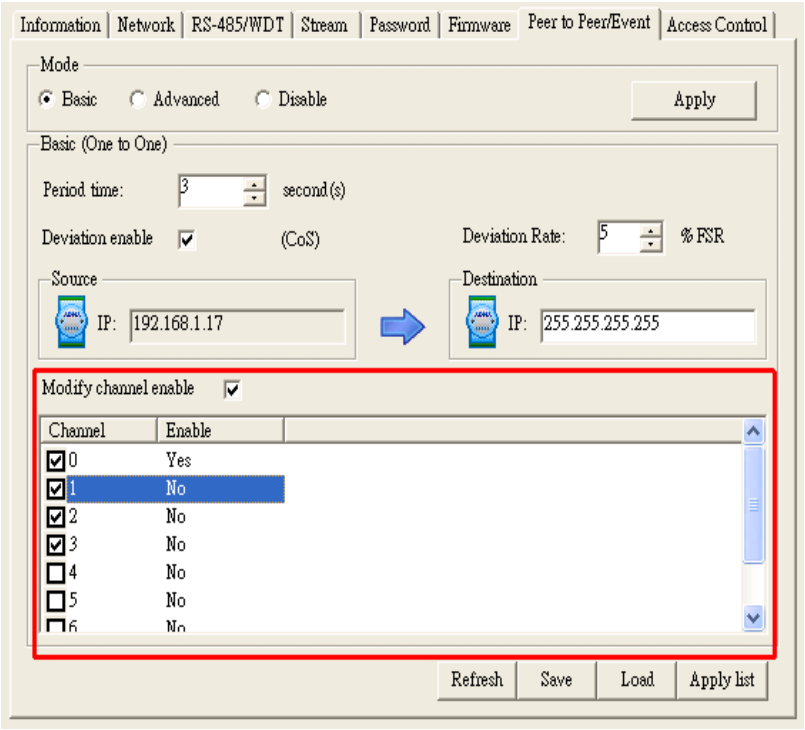


Figure 5.22: Building the Mapping Relationship

- Advanced Mode Configuration

When you choose the advanced mode, the **Status Display** area should look like the Figure 5.23 below. With advanced mode, each channel on the source ADAM-6000 module can be mapping to channel on different target devices. You can configure the mapping relation using the two block areas **Source** and **Destination** in the **Advanced (One to Multi)** area.

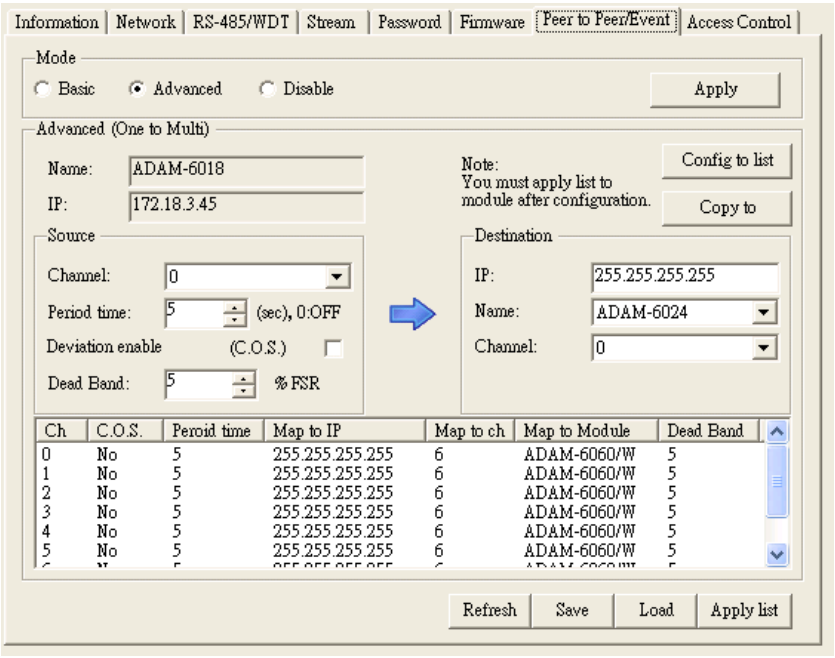


Figure 5.23: P-to-P Advanced Mode Configuration

Below are the steps to define the mapping relationship:

1. Select the input channel by the **Channel** combo box in **Source**.
2. Use **Period time** numeric control, **Deviation enable (C.O.S)** check box (for analog modules) or **Change of state (C.O.S)** check box (for digital modules), and **Deviation Rate** numeric control in the **Source** area to define when to transfer the data for that channel. The configuration operation is similar to Basic mode.
3. Define the target device for that channel by entering its IP address into the **IP** text box in the **Destination** area. Choose the correct module name from the **Name** combo box, and choose the output channel to receive the data by the **Channel** combo box.
4. After you have completed all these configurations for this channel, click the **Config to list** button. You can see your configuration for that channel is displayed by the mapping table below the **Source** and **Destination** area.

*Note: The mapping setting is only restored in memory of your computer, and it will download to the target ADAM-6000 module after you click the **Apply list** button below the mapping table. It is not suggested to download the mapping configuration immediately if you only complete setting for one channel.*

5. Repeat the step 1 to step 4 for another input channel. Continuously repeat the configuration until you have configured all the input channels which you want to create the mapping relationship. Click the **Apply list** button to download all mapping configuration to the target module. You can save all configurations in the mapping table into a file by clicking the **Save** button. Or you can load previous configuration from a file by clicking the **Load** button. If you click the **Refresh** button, the real configuration on the source module will be uploaded to your computer and you can see it in the mapping table.

Note: We suggest you to download all channels mapping configuration together at one time, instead of downloading one-channel setting many times. The reason is that this can save the times to use the flash memory on target module and help to extend the flash memory life.

In order to save setting time for the mapping configuration, you can copy one channel setting to other channels, and then only change what needs to change. You can do this by clicking the **Copy to** button, then a dialog window will pop-up. Refer to the Figure 5.24 below for the image of that dialog window.

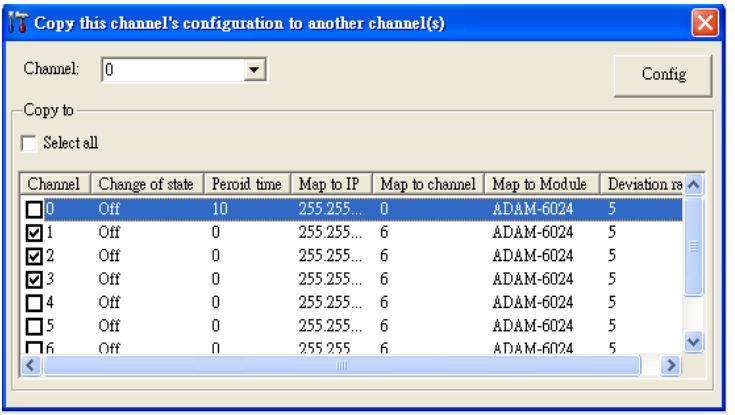


Figure 5.24: Copy One Setting to Other Channels

Choose the channel which provides the setting for other channels by the **Channel** combo box at the top of the dialog window. Then select channels which you want to copy setting to by clicking the **Channel** check box in the **Copy to** area. Using Figure 5.24 as an example, setting of channel 0 will copy to channel 1, 2, and 3. If you want to copy the setting to all channels, click the **Select all** check box. After selecting the channels, click the **Config** button. Then you will find the setting of the channels you selected has been copied in the mapping table. After that, you can individually select the channel needed to modify and change the parameters. Therefore, you don't need to do all the configurations and you can concentrate on setting on the parameters needed to be modified.

- **Peer-to-Peer Data Transfer Performance**

1. Wired LAN Module

Condition: transfer data from one channel of an ADAM-6050 module to one channel of another ADAM-6050 module, via one Ethernet switch.

Data Transfer Time: < 1.2 millisecond

5.4 ADAM-6000 Web Server

ADAM-6000 I/O modules all features built-in web server. Remote computer or devices can monitor and control I/O status on ADAM-6000 modules remotely through web browser. There is default built-in web page on ADAM-6000 modules. You can modify the web page using Java Applet. Refer to Section 5.5 for more detail.

To use your computer to browse the web page on ADAM-6000 module, you need to install Java Virtual Machine first. Then you can simply type the IP address to connect to your ADAM-6000 module in web browser. There will be one dialog window asking you to ask the password. After you have typed the correct password, you can start to monitor or control I/O on ADAM-6000 modules.

Note: For ADAM-6024 module, when you want to browse the web page, you need to enter user name (root) and password (00000000).

5.5 Java Applet Customization

5.5.1 Introduction

In this section, we will tell you the way to create an applet web page to monitor the status of ADAM-6060 through the Web browser. To write an input processing applet, you need to know how to define a class with multiple methods. To understand how an applet processes input data, you must learn what events are and how events are handled in Java programs. We don't intend to teach you how to write the applet because it is beyond the scope of our discussion here. Instead, we will provide you with a small-but-useful example as well as the relevant class, methods and suggested template. We refer the interested user who is intended to know more details to the following web site

http://java.sun.com/docs/books/tutorial/

To write an applet that is capable of processing ADAM-6060 input data in a very short time, we provide you with a class which includes all necessary methods. The kernel functions/methods to communicate with our product and display the current, updated status has been fine-tuned for any signal it can process. Four major methods are developed for the purpose, listed below: .

Employing these four methods, you can customize your applet and focus solely on the user interface you intend to create and the number of channels you want to monitor.

• boolean ForceCoil(int CoilAddr, boolean IsTrunOn)

This method is used for digital output of module channels. The parameter CoilAddr is integer data type and the coil address of the channel. IsTrueOn is the parameter used to indicate ON or OFF. If the method is successful, it will return true.

• boolean ReadCoil(int StartingAddr, int NoOfPoint, byte ModBusRTU[]) This method is used for digital input of module channels. The parameter StartingAddr is the starting address of desired channel. NoOfPoint is to indicate how many desired channels to be monitored. Both of the parameters are of integer data type. The third parameter, ModBusRTU is an array with data type of byte, which is used to carry digital inputs of the desired channels. The default size is 128.

• boolean ReadRegister(int StartingAddr, int NoOfPoint, byte ModBusRTU[]) This method is used for analog input of module channels. The parameter StartingAddr is the starting address of desired channel. NoOfPoint is to indicate how many desired channels to be monitored. Both of the parameters are of integer data type. The third parameter, ModBusRTU is an array with data type of byte, which is used to carry analog inputs of the desired channels. The default size is 128.

An Example

To process ADAM-6060 input and display the result/status on an applet, we will use objects from the standard java class library and the class we develop. Specifically, we provide Modbus class to handle the communication with ADAM-6000 I/O modules. Now we're going to teach you step by step how to customize your Web page.

Java Applet Programming

To create your own Web page, you have to follow some rules. There are two parts in this section. We start from the HTML file. Please refer below for the default HTML source code.

```
<HTML>
<HEAD>
<TITLE>
ADAM-6000 Ethernet-Enabled DA&C Modules
</TITLE>
</HEAD>
<BODY>
<APPLET
        CODEBASE = "."
         CODE     = "Adam6060.class"
        ARCHIVE = "Adam6060.jar"
         NAME     = "Adam6060 Relay Module"
         WIDTH    = 500

        HEIGHT   = 400
         HSPACE   = 0
         VSPACE   = 0
        ALIGN    = middle
>
<PARAM NAME = "HostIP" VALUE = "010.000.000.000">
</APPLET>
</BODY>
</HTML>
```

Firstly, the HTML file must be named index.html. The name of parameter in <APPLET...> cannot change. The lines CODE = "Adam6060.class" and ARCHIVE = "Adam6060.jar" indicate where the class and jar files (your Java Applet program) are for ADAM-6060 module. WIDTH and HEIGHT are parameters to set the visible screen size of your Java Applet Web page.

The HTML is a good template for you to create your own embedded Web page; however, the parameter names and most of their values cannot be modified, or it will not work. You can only change the value of WIDTH and HEIGHT parameters, e.g. WIDTH = 640 and HEIGHT = 480. However, you must change the value of CODE and ARCHIVE when you try to write it for another module, say ADAM-6017, and thus you should use Adam6017.class and Adam6017.jar instead of Adam6060.class and Adam6060.jar.

Some Instructions when Writing a Java Applet

To enable your java applet to communicate with ADAM-6000 I/O modules, you have to include the following code in the very beginning of your program:

```
import Adam.ModBus.*;
```

In constructor it is suggested to add the following fragment in your exception handler:

```
    Try {
        HostIP  = getParameter("HostIP");
        Adam6060Connection = new ModBus(HostIP);
        if (HostIP == "")
        labAdamStatusForDIO.setText("Get Host IP is null !!");
    else
        labAdamStatusForDIO.setText("Get Host IP :" +
        Adam6060Connection.GetHostIP() + " Ver 1.00");
    ||||||
    ||||||
}
```

The fragment is used to obtain the host IP value and check if it is null. To acquire the necessary parameter information from the index.html, you need to add the fragment below.

```
    public String[][] getParameterInfo() {
        String[][] pinfo =
            {
            {"HostIP", "String", ""},
            };
    return pinfo;
}
```

As for mouse/keyboard events and graphical user interface, they are beyond the scope of our discussion here and we will leave them to users.

After you finish your program and compile, it should generate a couple of classes, e.g. ADAM6060.class, ADAM6060$1.class, ADAM6060$2, and myFramPanel.class in our example. Then, follow the standard way to combine the generated classes with ModBus.class which must be placed in the directory path œAdam/ModBus/ into a jar file. In this case, the name for the file should be ADAM6060.jar. The figure below shows the structure to make the jar file.

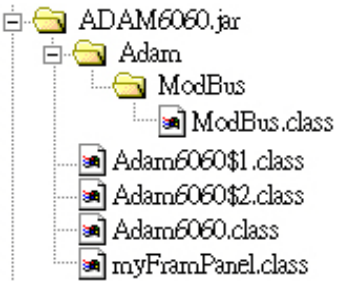


Figure 5.25: Structure of the ADAM6060.jar file

Start your ADAM utility, and open the Firmware tab in the Status Display area as shown below (Refer to Section 5.3.2). Then, tell the utility where the path is for the JAR and HTML files. In this case, they are ADAM-

6060.jar and index.html. Push button, and a confirmation window pops up. After you confirm, it will start processing.

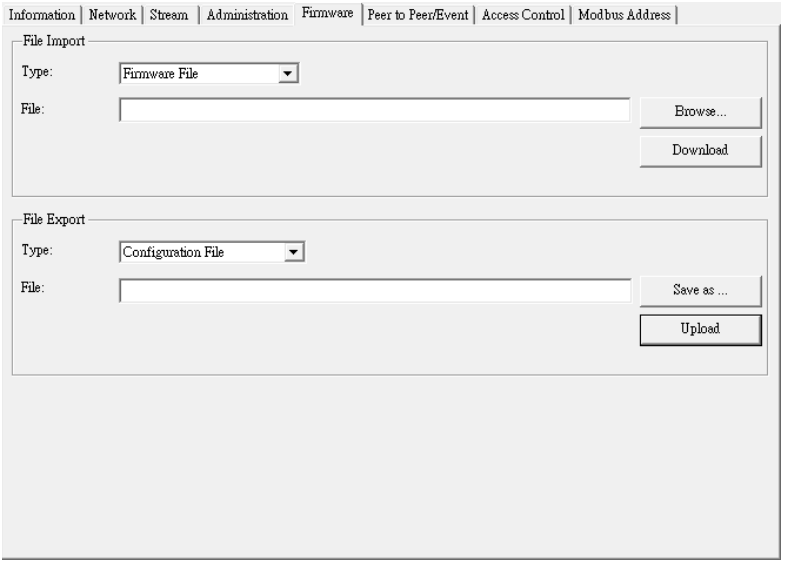


Figure 5.26: Firmware Upgrade

5.6 Source Code of Java Applet Example

```
import Adam.ModBus.*;
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import java.io.*;
import java.lang.*;

public class Adam6060 extends Applet {
    boolean isStandalone = false;
    String var0;
    Thread AdamPoilThread;
    String HostIP;
    long ErrCnt = 0;
    boolean IsAdamRuning = false;
    ModBus Adam6060Connection;

    Label Label1 = new Label();

    myFramPanel palStatus = new myFramPanel (2);
    myFramPanel pal1 = new myFramPanel (3);
    myFramPanel pal2 = new myFramPanel (3);
    myFramPanel palAdamStatus = new myFramPanel (1);

    Label labStartAddress = new Label("Start Address:");
    TextField txtStartAddress = new TextField("1");
    Label labCount = new Label("No. of coils to read(Max 128):");
    TextField txtCount = new TextField("1");
    Button btAdam6060 = new Button("Read Coils");
    TextArea txtMsg = new TextArea("", 1, 10, 1);
    Label labAdamStatusForDIO = new Label("Status : ");
```

```
        /**Get a parameter value*/
        public String getParameter(String key, String def) {
          return isStandalone ? System.getProperty(key, def) :
          (getParameter(key) != null ? getParameter(key) : def);
        }

        /**Constructor*/
        public Adam6060() {
        }
        /**Applet Initialization*/
        public void init() {
          try {
                HostIP= getParameter("HostIP");
                Adam6060Connection = new ModBus(HostIP);
    //create ADAM-6060 module object
        if (HostIP == "")//check the Host IP
                labAdamStatusForDIO.setText("Get Host IP is null !!");
        else
                labAdamStatusForDIO.setText("GetHostIP:"+
    Adam6060Connection.GetHostIP() + " Ver 1.00");

        jbInit();
    }
    catch(Exception e) {
        e.printStackTrace();
      }
    }

    /**Component initialization and displayed screen*/
    private void jbInit() throws Exception {
        this.setLayout(null);
```

```
    palStatus.setBackground(Color.lightGray);
    palAdamStatus.setBackground(Color.lightGray);
    palStatus.setBounds(new Rectangle(42, 50, 409, 15 *2 + 0 * 2 + 77 +
152 + 33 ));
    pal1.setBounds(new Rectangle(12, 15 , 385, 77));
    pal2.setBounds(new Rectangle(12, 15 + 77 + 0 , 385, 152));
   palAdamStatus.setBounds(new Rectangle(12, 15 + 77+0*2+ 152, 385,
33));
    palStatus.setLayout(null); pal1.setLayout(null);
    pal1.add(labStartAddress, null); pal1.add(txtStartAddress, null);
    pal1.add(labCount, null); pal1.add(txtCount, null);
    pal1.add(btAdam6060, null);
labStartAddress.setBounds(new Rectangle(20, 15, 85, 20));
txtStartAddress.setBounds(new Rectangle(205, 15, 60, 20));
labCount.setBounds(new Rectangle(20, 40, 180, 20));
txtCount.setBounds(new Rectangle(205, 40, 60, 20));
btAdam6060.setBounds(new Rectangle(275, 40, 80, 22));


btAdam6060.addMouseListener(new java.awt.event.MouseAdapter() {
public void mousePressed(MouseEvent e) {//mouse event handling
        int i, j;
        long    lAddress, lCount;
        byte ModBusRTU[] = new byte[128];


        if
(Adam6060Connection.ReadCoil((int)Long.parseLong(txtStartAd-
dress.getText()), (int)Long.parseLong(txtCount.getText()), Mod-
BusRTU))
{
        lAddress = Long.parseLong(txtStartAddress.getText());
        for( i = 0; i < Long.parseLong(txtCount.getText()); i++)
        {
```

```
                txtMsg.append ("Address:" + String.valueOf(lAddress) +
"       ->      " + String.valueOf((int)ModBusRTU[i]) + "\n");
                lAddress++;
        }
}
else
{
        try
        {
        Adam6060Connection = new ModBus(HostIP);
}
        catch(Exception eNet) { eNet.printStackTrace(); }
    palAdamStatus.setLayout(null);
    pal2.setLayout(null);
    pal2.add(txtMsg, null);
    txtMsg.setBounds(new Rectangle(15, 15, 355, 120));

    Label1.setFont(new java.awt.Font("DialogInput", 3, 26));
    Label1.setForeground(Color.blue);
    Label1.setText("ADAM-6060 DI/O Module");
    Label1.setBounds(new Rectangle(83, 17, 326, 29));
    this.add(Label1, null);
    this.add(palStatus, null);
    palStatus.add(pal1, null);
    palStatus.add(pal2, null);

    palStatus.add(palAdamStatus, null);

    labAdamStatusForDIO.setBounds(new Rectangle(10, 8, 350, 12));
    palAdamStatus.add(labAdamStatusForDIO, null);
    }
```

```
/**Applet Information Acquisition*/
public String getAppletInfo() {
return "Applet Information";
}
/**Get parameter info*/ public String[][] getParameterInfo() { String[][]
pinfo =
{
{"HostIP", "String", ""},
};
return pinfo; }

 /**Main method: for the purpose of laying out the screen in local PC*/
 public static void main(String[] args) { Adam6060 applet = new
Adam6060(); applet.isStandalone = true;
Frame frame;
frame = new Frame() {
protected void processWindowEvent(WindowEvent e) {
super.processWindowEvent(e); if (e.getID() == Window-
Event.WINDOW_CLOSING) { System.exit(0);
}
} public synchronized void setTitle(String title) { super.setTitle(title);
enableEvents(AWTEvent.WINDOW_EVENT_MASK);
}
}; frame.setTitle("Applet Frame"); frame.add(applet, BorderLay-
out.CENTER); applet.init();
applet.start();
frame.setSize(500,620);
Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
frame.setLocation((d.width - frame.getSize().width) / 2, (d.height
- frame.getSize().height) / 2);
frame.setVisible(true);
}
```

```
}

/**Displayed Screen*/
class myFramPanel extends Panel
{
int panelType; Label labMassage = new Label("");

public myFramPanel() {
//super();
}

public myFramPanel(int myType) {
//super();
panelType = myType;
}

public myFramPanel(int myType, String Msg, int msgTextLength) {
//super();
panelType = myType; if (Msg != "") { labMassage.setText(Msg); this.set-
Layout(null);
labMassage.setBounds(new Rectangle(20, 3, msgTextLength,
15));

}
}

this.add(labMassage);
 public void paint(Graphics g) { Dimension size = getSize();

if (panelType == 1) {
int off;
```

```
off = 4;
g.setColor(Color.white);
g.drawRect(0, 0, size.width - 1, size.height - 1);


 1);

g.setColor(Color.darkGray);
g.drawLine(size.width - 1, 0, size.width - 1, size.height -

g.drawLine(0, size.height - 1, size.width - 1, size.height
- 1);g.setColor(Color.black);
g.setColor(Color.black);
g.drawRect(off, off, size.width-2- off*2, size.height
- 2 - off * 2);
}
else if (panelType == 2){
g.setColor(Color.white);
g.drawRect(0, 0, size.width - 1, size.height - 1);

 4);

- 4);

g.drawLine(size.width - 4, 2, size.width - 4, size.height -

g.drawLine(2, size.height - 4, size.width - 4, size.height
 g.setColor(Color.darkGray); g.drawLine(2, 2, size.width - 4, 2); g.draw-
Line(2, 2, 2, size.height - 4);

 1);
```

```
g.drawLine(size.width - 1, 0, size.width - 1, size.height -

g.drawLine(0, size.height - 1, size.width - 1, size.height
- 1);g.setColor(Color.black);
}
else if (panelType == 3) {
int off;
off = 4;
g.setColor(Color.white);
g.drawRect(0, 0, size.width - 1, size.height - 1);
 1);

- 1);

g.setColor(Color.darkGray);
g.drawLine(size.width - 1, 0, size.width - 1, size.height -

g.drawLine(0, size.height - 1, size.width - 1, size.height

g.setColor(Color.black);
g.drawRect(off, off + 5, size.width-2- off*2, size.height - 2 - off * 2 -5 );
                }
                else {
                        g.setColor(Color.darkGray);
                        g.drawRect(0, 0, size.width - 1, size.height - 1);
                }
        }
};
```

CHAPTER 6

Planning Your Application Program

Sections include:

- Introduction
- ADAM.NET Class Library

Chapter 6 Planning Your Application Program

6.1 Introduction

After completing the system configuration, you can begin to plan the application program. This chapter introduces two programming tools for users to execute system data acquisition and control. The DLL drivers and command sets provide a friendly interface between your applications and ADAM-6000 I/O modules.

6.2 ADAM .NET Class Library

Advantech ADAM .NET Class library enables you quickly and easily develop your application programs written in the Microsoft Visual Studio .NET 2003. The ADAM .NET Class library includes all necessary functions to utilize ADAM-6000 modules.

Before installing ADAM .NET Class library, you have to install the ***Microsoft Visual Studio .NET 2003*** as well as the ***Microsoft Visual Studio .NET 2003 documentation***. For WinCE programmers, you also need to install the ***ActiveSync*** and ***Windows CE .NET Utilities v1.1 for Visual Studio .NET 2003*** from Microsoft website. Then, you can install the ADAM .NET Class library by the setup file included in ADAM CD. You can also free download the installation file at the Advantech website: http://www.advantech.com. Run the installation setup file Adam.NET Class Library.exe, that it will install ADAM .NET Class library for Win32 and WinCE platform.

After you complete the installation of the ADAM .NET Class library, the Win32 Class library will be installed into the following path: **Program Files\Advantech\ Adam.NET Class Library\VS2003**, and the WinCE Class library will be installed into the following path: **Program Files\Microsoft Visual Studio .NET 2003\ CompactFrameworkSDK\v1.0.5000\Windows CE**.

In order to help you to be familiar with developing your application program in a short time, there are many built-in example programs in the path: **Program Files\Advantech\ Adam.NET Class Library\VS2003\Samples** for all ADAM-6000 modules. Simply opening these example programs, you can quickly get the idea how to write a program code to control ADAM-6000 modules. Moreover, you can directly compile and execute the program after modifying the correct IP address and module name, to start a data acquisition and control application in a shortest time.

Take ADAM-6050 module as example, you can simply launch the example program project **ADAM 60xxxDIO.sln** written in Microsoft Visual Basic .NET located in **Program Files\Advantech\Adam.NET Class Library\VS2003\Samples\Win32\ VB\Adam60xxDIO**. After you launch it, open the source code and modify the IP address and module name to meet the real situation. Refer to Figure 6.1 below.

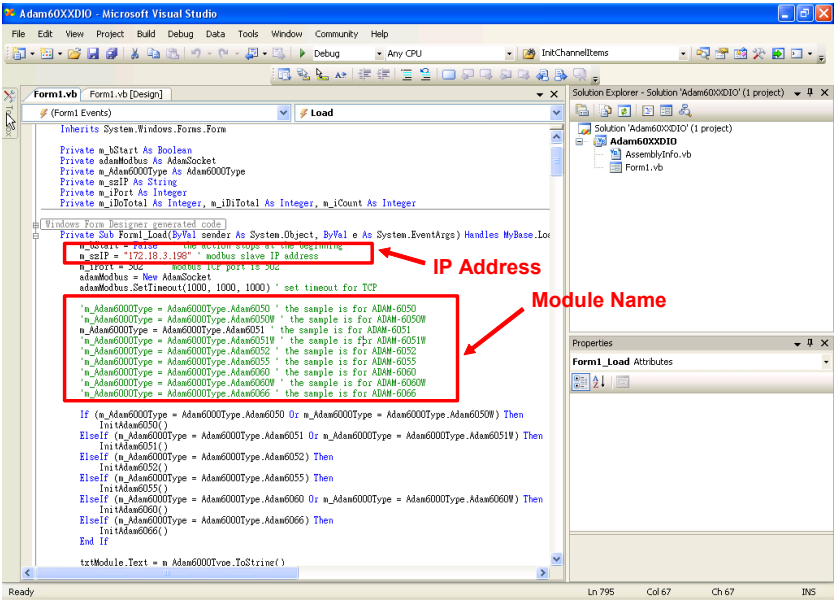


Figure 6.1: Modifying ADAM-6050 .NET

After you complete the code modification, you can directly compile the program. Then you can execute the program to start the application.

There are plenty of functions in ADAM. NET Class library, and help documentation is provided to let you understand more about these functions. You can launch the help documentation by clicking the Start button on the taskbar. Refer to the Figure 6.2 below.

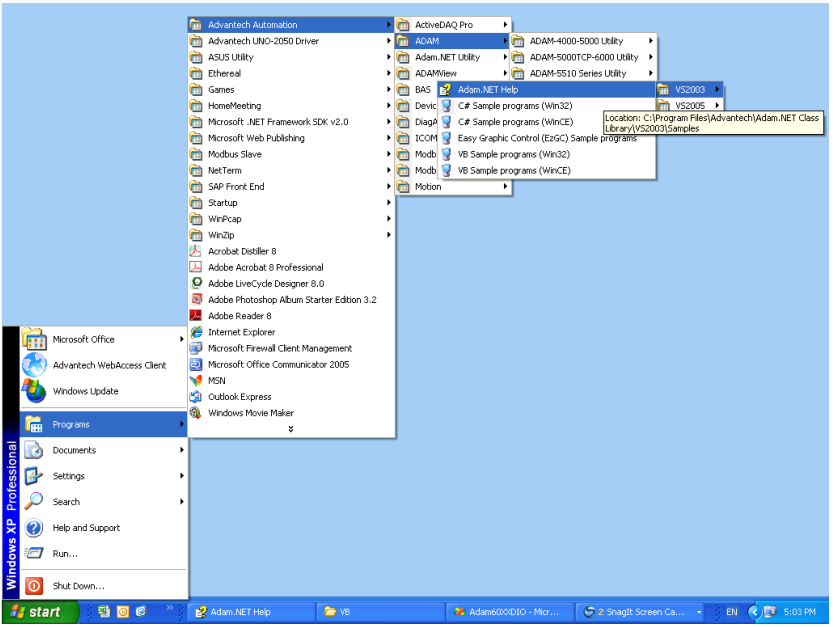


Figure 6.2: Launching ADAM .NET Class Library

6.3 ADAM-6000 Commands

ADAM-6000 and ADAM-5000/TCP system accept a command/response form with the host computer. When systems are not transmitting they are in listen mode. The host issues a command to a system with a specified address and waits a certain amount of time for the system to respond. If no response arrives, a time-out aborts the sequence and returns control to the host. This chapter explains the structure of the commands with Modbus/TCP protocol, and you can refer to the example codes provided by .NET class library to see how to write or read Modbus/TCP address of ADAM-6000 modules. The examples are in the folder listed below:

Windows win32 operating system (such as windows 2000/XP/Vista):

Program Files\Advantech\Adam.NET Class Library\VS2003\samples\Win32\VB\ModbusTCP

Program Files\Advantech\Adam.NET Class Library\VS2003\samples\Win32\Vc#\ModbusTCP

Windows CE.NET operating system:

Program Files\Advantech\Adam.NET Class Library\VS2003\samples\WinCE\VB\ModbusTCP

Program Files\Advantech\Adam.NET Class Library\VS2003\samples\WinCE\Vc#\ModbusTCP

Note: Please refer to Appendix B.2 for the Modbus/TCP address of ADAM-6000 modules.

6.3.1 Command Structure

It is important to understand the encapsulation of a Modbus request or response carried on the Modbus/TCP network. A complete command is consisted of command head and command body. The command head is prefixed by six bytes and responded to pack Modbus format; the command body defines target device and requested action. Following example will help you to realize this structure quickly.

6.3.2 Modbus Function Code Introductions

To fullfill the programming requirement, there is a series of function code standard for users reference.

Code (Hex)	Name	Usage
01	Read Coil Status	Read Discrete Output Bit
02	Read Input Status	Read Discrete Input Bit
03	Read Holding Registers	
04	Read Input Registers	Read 16-bit register. Used to read integer or floating point process data.
05	Force Single Coil	Write data to force coil On/Off
06	Preset Single Register	Write data in 16-bit format
08	Loopback Diagnosis	Diagnostic testing of the communication port
15	Force Multiple Coils	Write multiple data to force coil On/Off
16	Preset Multiple Registers	Write data in 16-bit format

Function Code 01

The function code 01 is used to read the discrete outputs ON/OFF status of ADAM-6000 modules in a binary data format.

Request message format for function code 01:

Command Body					
Station Address	Function Code	Start Address High Byte	Start Address Low Byte	Requested Number of Coil High Byte	Requested Number of Coil Low Byte

Example: Read coil number 1 to 8 (address number 00017 to 00024) from ADAM-6000 Modules

01 01 00 17 00 08

Response message format for function code 01:

Command Body				
Station Address	Function Code	Byte Count	Data	Data

Example: Coils number 2 and 7 are on, all others are off.

01 01 01 42

In the response the status of coils 1 to 8 is shown as the byte value 42 hex, equal to 0100 0010 binary.

Function Code 02

The function code 02 is used to read the discrete inputs ON/OFF status of ADAM-6000 in a binary data format.

Request message format for function code 02:

Command Body					
Station Address	Function Code	Start Address High Byte	Start Address Low Byte	Requested Number of Input High Byte	Requested Number of Input Low Byte

Example: Read coil number 1 to 8 (address number 00001 to 00008) from ADAM-6000 modules

01 02 00 01 00 08

Response message format for function code 02:

Command Body				
Station Address	Function Code	Byte Count	Data	Data

Example: input number 2 and 3 are on, all others are off.

01 02 01 60

In the response the status of input 1 to 8 is shown as the byte value 60 hex, equal to 0110 0000 binary.

Function Code 03/04

The function code 03 or 04 is used to read the binary contents of input registers

Request message format for function code 03 or 04:

Command Body					
Station Address	Function Code	Start Address High Byte	Start Address Low Byte	Requested Number of Register High Byte	Requested Number of Register Low Byte

Example: Read Analog inputs #1 and #2 in addresses 40001 to 40002 as floating point value from ADAM-6017 module

01 04 00 01 00 02

Response message format for function code 03 or 04:

Command Body				
Station Address	Function Code	Byte Count	Data	Data

Example: Analog input #1 and #2 as floating point values where AI#1=100.0 and AI#2=55.32

01 04 08 42 C8 00 00 47 AE 42 5D

Function Code 05

Force a single coil to either ON or OFF. The requested ON/OFF state is specified by a constant in the query data field. A value of FF 00 hex requests it to be ON. A value of 00 00 hex requests it to be OFF. And a value of FF FF hex requests it to release the force.

Request message format for function code 05:

Command Body					
Station Address	Function Code	Coil Address High Byte	Coil Address Low Byte	Force Data High Byte	Force Data Low Byte

Example: Force coil 3 (address 00003) ON in ADAM-6000 module

01 05 00 03 FF 00

Response message format for function code 05:

The normal response is an echo of the query, returned after the coil state has been forced.

Command Body					
Station Address	Function Code	Coil Address High Byte	Coil Address Low Byte	Force Data High Byte	Force Data Low Byte

Function Code 06

Presets integer value into a single register. Request message format for function code 06:

Command Body					
Station Address	Function Code	Register Address High Byte	Register Address Low Byte	Preset Data High Byte	Preset Data Low Byte

Example: Preset register 40002 to 00 04 hex in ADAM-6000 module

01 06 00 02 00 04

Response message format for function code 06:

The normal response is an echo of the query, returned after the coil state has been preset.

Command Body					
Station Address	Function Code	Register Address High Byte	Register Address Low Byte	Preset Data High Byte	Preset Data Low Byte

Function Code 08

Echoes received query message. Message can be any length up to half the length of the data buffer minus 8 bytes.

Request message format for function code 08:

Command Body		
Station Address	Function Code	Any data, length limited to approximately half the length of the data buffer

Example: 01 08 00 02 00 04

Response message format for function code 08:

Command Body		
Station Address	Function Code	Data bytes recieved

Example: 01 08 00 02 00 04

Function Code 15 (0F hex)

Forces each coil in a sequence of coils to either ON or OFF. Request message format for function code 15:

Command Body								
Station Address	Function Code	Start Address High Byte	Start Address Low Byte	Requested Number of Coil High Byte	Requested Number of Coil Low Byte	Byte Count	Force Data High Byte	Force Data Low Byte

Example: Request to force a series of 10 coils starting at address 00017 (11 hex) in ADAM-6000 module.

01 0F 00 11 00 0A 02 CD 01 The query data contents are two bytes: CD 01 hex, equal to 1100 1101 0000 0001 binary. The binary bits are mapped to the addresses in the following way.

Bit: 1 1 0 0 1 1 0 1 0 0 0 0 0 0 0 1

Address (000XX): 24 23 22 21 20 19 18 17-- -- - - 26 25

Response message format for function code 08: The normal responses return the station address, function code, start address, and requested number of coil forced.

Command Body					
Station Address	Function Code	Start Address High Byte	Start Address Low Byte	Requested Number of Coil High Byte	Requested Number of Coil Low Byte

Example: 01 0F 00 11 00 0A

Function Code 16 (10 hex)

Preset values into a sequence of holding registers. Request message format for function code 16:

Command Body							
Station Address	Function Code	Start Address High Byte	Start Address Low Byte	Requested Number of Register High Byte	Requested Number of Register Low Byte	Byte Count	Data

Example: Preset constant #1 (address 40009) to 100.0 in ADAM-6000 module.

01 10 00 09 00 02 04 42 C8 00 00

Response message format for function code 08: The normal responses return the station address, function code, start address, and requested number of registers preset.

Command Body					
Station Address	Function Code	Start Address High Byte	Start Address Low Byte	Requested Number of Register High Byte	Requested Number of Register Low Byte

Example: 01 10 00 09 00 02

6.4 ASCII Commands for ADAM-6000 Modules

For users do not familiar to Modbus protocol, Advantech offers a function library as a protocol translator, integrating ASCII command into Modbus/TCP structure. Therefore, users familiar to ASCII command can access ADAM-6000 easily. Before explaining the structure of ASCII command packed with Modbus/TCP format. Lets see how to use an ASCII command and how many are available for your program.

6.4.1 Syntax of ASCII

Command Syntax: [delimiter character][address][channel][command][data][checksum][carriage return] Every command begins with a delimiter character. There are two valid characters:

$ and # The delimiter character is followed by a two-character address (hex-decimal) that specifies the target system. The two characters following the address specified the module and channel.

Depending on the command, an optional data segment may follow the command string. An optional two-character checksum may also be appended to the command string. Every command is terminated with a carriage return (cr).

Note: All commands should be issued in UPPERCASE characters only!

The command set is divided into the following five categories:

- System Command Set
- Analog Input Command Set
- Analog Input Alarm Command Set
- Universal I/O Command Set
- Digital I/O Command Set

Every command set category starts with a command summary of the particular type of module, followed by datasheets that give detailed information about individual commands. Although commands in different subsections sometime share the same format, the effect they have on a certain module can be completely different than that of another. Therefore, the full command sets for each type of modules are listed along with a description of the effect the command has on the given module.

6.4.2 System Command Set

Command Syntax	Command Name	Description
$aaM	Read Module Name	Return the module name from a specified module
$aaF	Read Firmware Version	Return the firmware version from a specified module
#aaVd-bbbbddddddd dd	Write GCL Internal Flags	Write value(s) to GCL internal flag(s) on a specific ADAM-6000 module
$aaVd	Read GCL Internal Flags	Read all GCL internal flags' values from a specific ADAM-6000 module

Note: *Command **$aaM** and **$aaF** support all ADAM-6000 I/O modules.*

*Command **#aaVdbbbbddddddd** supports ADAM-6050, 6051, 6052, 6060, 6066.*

*Command **#aaVd** supports ADAM-6050, 6051, 6052, 6060, 6066.*

$aaM

Name Read Module Name

Description Returns the module name from a specified module.

Syntax $aaM(cr)

$ is a delimiter character.

aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module you want to interrogate. (Always 01)

M is the Module Name command.

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

Response !aa60bb(cr) if the command is valid.
?aa(cr) if an invalid operation was entered.
There is no response if the module detects a syntax error, communication error or if the address does not exist.
! delimiter indicating a valid command was received.
? delimiter indicating the command was in-valid.
aa (range 00-FF) represents the 2-character hexadecimal slave address of an ADAM-6000 module.
bb (range 00-FF) represents the 2-character model number of an ADAM-6000 module.
(cr) is the terminating character, carriage return (0Dh).

Example command: $01M(cr)
response: !016050(cr)
The command requests the system at address 01h to send its module name. The system at address 01h responds with module name 6050 indicating that there is an ADAM-6050 at address 01h.

$aaF

Name Read Firmware Version

Description Returns the firmware version from a specified module.

Syntax $aaF(cr)
$ is a delimiter character.
aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module you want to interrogate. (Always 01)
F is the Firmware Version command.
(cr) is the terminating character, carriage return (0Dh).

Response !aa(version)(cr) if the command is valid.
?aa(cr) if an invalid operation was entered.
There is no response if the module detects a syntax error, communication error or if the address does not exist.
! delimiter indicating a valid command was received.
? delimiter indicating the command was invalid.
aa (range 00-FF) represents the 2-character hexadecimal slave address of an ADAM-6000 module.
(version) represents the firmware version of the module.
(cr) is the terminating character, carriage return (0Dh).

Example command: $01F(cr)
response: !01 1.01(cr)
The command requests the system at address 01h to send its firmware version.
The system responds with firmware version 1.01.

#aaVdbbbbddddddd

Name Write Value(s) to GCL Internal Flags (Auxiliary Flags)

Description This command sets a single or all GCL internal flag(s) on the specific ADAM-6000 module. Refer to section 7.3.1 and 7.3.4 for definition of GCL internal flag.

Syntax #aaVdbbbbddddddd(cr)
is a delimiter character.
aa (range 00-FF) represents the 2-character hexadecimal slave network address of the ADAM-6000 module. (Always 01)
Vd is the GCL Internal Flag command.
bbbb is used to indicate which GCL internal flag(s) to set.
Writing to all GCL internal flags: 0000
Writing to a single GCL internal flag: First character is 1, and 2 ~ 4 characters indicate the GCL internal flag number which can range from 0h to Fh.
ddddddd is the hexadecimal representation of the GCL internal

flag value(s).
Each character represents 4 GCL internal flags' values.
(cr) is the terminating character, carriage return (0Dh)

Response >aa(cr) if the command was valid.
?aa(cr) if an invalid command has been issued.
There is no response if the module detects a syntax error or communication error or if the address does not exist.
> delimiter indicating a valid command was received.
? delimiter indicating the command was invalid.
aa (range 00-FF) represents the 2-character hexadecimal slave address of a module that is responding.
(cr) is the terminating character, carriage return (0Dh)

Example command: #01Vd000000000000(cr)
response: >01(cr)
This command sets all GCL internal flags values.
Values of All GCL internal flags (Flag 0 ~ 15) are logic low.

command: #01Vd00000000FFFF(cr)
response: >01(cr)
This command sets all GCL internal flags values.
Values of All GCL internal flags (Flag 0 ~ 15) are logic high.

command: #01Vd100300000001(cr)
response: >01(cr)
This command sets one specific GCL internal flag value.
GCL internal flag (Flag 3) is logic high.

command: #01Vd100E00000000(cr)
response: >01(cr)
This command sets one specific GCL internal flag value.
GCL internal flag (Flag 14) is logic low.

$aaVd

Name Read GCL Internal Flags' (Auxiliary Flags) Values

Description This command reads all GCL internal flags' values from the specific ADAM-6000 module. Refer to section 7.3.1 and 7.3.4 for definition of GCL internal flag.

Syntax $aaVd(cr)
$ is a delimiter character.
aa (range 00-FF) represents the 2-character hexadecimal slave network address of the ADAM-6000 module. (Always 01)
Vd is the GCL Internal Flag command.
(cr) is the terminating character, carriage return (0Dh)

Response >aadddddddd(cr) if the command was valid.
?aa(cr) if an invalid command has been issued.
There is no response if the module detects a syntax error or communication error or if the address does not exist.
> delimiter indicating a valid command was received.
? delimiter indicating the command was invalid.
aa (range 00-FF) represents the 2-character hexadecimal slave address of a module that is responding.
dddddddd is the hexadecimal representation of the GCL internal flag value(s).
Each character represents 4 GCL internal flags' values.
(cr) is the terminating character, carriage return (0Dh)

Example command: $01Vd (cr)
response: >010000002B(cr)
This command reads all GCL internal flags values.
The characters "B" mean "1011" and character "2" means "0010".
Therefore, GCL internal flag 0, 1, 3, 5 are logic high while other GCL internal flags are all logic low.

6.4.3 Analog Input Command Set
(ADAM-6015, 6017, 6018)

Command Syntax	Command Name	Description
#aan	Read Analog Input from Channel N	Return the input value from the specified analog input channel
#aa	Read Analog Input from all channels	Return the input values from all analog input channels
$aa0	Span Calibration	Calibrate the analog input module to correct the gain error
$aa1	Offset Calibration	Calibrate the analog input module to correct the offset error
$aa6	Read Channel Enable/Disable Status	Asks a specified module to return the Enable/Disable status of all analog input channels
$aa5mm	Set Channel Enable/Disable Status	Set Enable/Disable status for analog input channels
#aaMH	Read all Max. Data	Read the maximum data from all analog input channels
#aaMHn	Read single Max. Data	Read the maximum date from a specified analog input channel
#aaML	Read all Min. Data	Read the minimum data from all analog input channels
#aaMLn	Read single Min. Data	Read the minimum data from a specified analog input channel
#aaDnd	Set Digital Output	Sets the status for the specified digital output channels
$aaBnn	Read Analog Input Range Code	Return the input range code from the specific analog input channel

#aan

Name	Read Analog Input from Channel N
Description	Returns the input data from a specified analog input channel in a specified module.
Syntax	#aan(cr) # is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module you want to interrogate. (Always 01) n (range 0-8) represents the specific channel you want to read the input data. (cr) is the terminating character, carriage return (0Dh).
Response	>(data)(cr) if the command is valid. ?aa(cr) if an invalid operation was entered. There is no response if the module detects a syntax error or communication error or if the address does not exist. > delimiter indicating a valid command was received. ? delimiter indicating the command was invalid. (cr) is the terminating character, carriage return (0Dh).
Example	command: #012(cr) response: >+10.000 Channel 2 of the ADAM-6000 analog module at address 01h responds with an input value +10.000.

#aa

Name	Read Analog Input from All Channels
Description	Returns the input data from all analog input channels in a specified module.
Syntax	#aa(cr) # is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module you want to interrogate. (Always 01) (cr) is the terminating character, carriage return (0Dh).
Response	>(data)(data)(data)(data)(data)(data)(data)(data)(data)(cr) if the command is valid. ?aa(cr) if an invalid operation was entered. There is no response if the module detects a syntax error or communication error or if the address does not exist. > delimiter indicating a valid command was received. ? delimiter indicating the command was invalid. (cr) is the terminating character, carriage return (0Dh).
Note:	The latest data returned is the Average value of the preset channels in this module.
Example	command: #01(cr) response: >+10.000+10.000+10.000+10.000+10.000 +10.000+10.000+10.000+10.000

$aa0

Name	Span Calibration
Description	Calibrates a specified module to correct for gain errors
Syntax	$aa0(cr) $ is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module which is to be calibrated. (Always 01) 0 represents the span calibration command. (cr) is the terminating character, carriage return (0Dh)
Response	!aa(cr) if the command is valid. ?aa(cr) if an invalid operation was entered. There is no response if the module detects a syntax error or communication error or if the address does not exist. ! delimiter indicating a valid command was received. ? delimiter indicating the command was invalid. aa (range 00-FF) represents the 2-character hexadecimal slave address of an ADAM-6000 module. (cr) is the terminating character, carriage return (0Dh)
Note:	In order to successfully calibrate an analog input module's input range, a proper calibration input signal should be connected to the analog input module before and during the calibration process.

$aa1

Name	Zero Calibration
Description	Calibrates a specified module to correct for offset errors
Syntax	$aa1(cr) $ is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal Modbus address of the ADAM-6000 module which is to be calibrated. (Always 01) 1 represents the zero calibration command. (cr) is the terminating character, carriage return (0Dh)
Response	!aa(cr) if the command is valid. ?aa(cr) if an invalid operation was entered. There is no response if the module detects a syntax error or communication error or if the address does not exist. ! delimiter indicating a valid command was received. ? delimiter indicating the command was invalid. aa (range 00-FF) represents the 2-character hexadecimal slave address of an ADAM-6000 module. (cr) is the terminating character, carriage return (0Dh)
Note:	In order to successfully calibrate an analog input module's input range, a proper calibration input signal should be connected to the analog input module before and during the calibration process.

$aa6

Name	Read Channel Enable/Disable Status
Description	Asks a specified module to return the Enable/Disable status of all analog input channels
Syntax	$aa6(cr) $ is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module you want to interrogate. (Always 01) 6 is the read channels status command. (cr) is the terminating character, carriage return (0Dh)
Response	!aamm(cr) if the command is valid. ?aa(cr) if an invalid operation was entered. There is no response if the module detects a syntax error or communication error or if the address does not exist. ! delimiter indicating a valid command was received. ? delimiter indicating the command was invalid. aa (range 00-FF) represents the 2-character hexadecimal slave address of an ADAM-6000 module. mm are two hexadecimal values. Each value is interpreted as 4 bits. The first 4-bit value represents the status of channels 7-4, the second 4 bits represents the status of channels 3-0. A value of 0 means the channel is disabled, while a value of 1 means the channel is enabled. (cr) is the terminating character, carriage return (0Dh)
Example	command: $016(cr) response: !01FF(cr) The command asks the specific module at address 01h to send Enable/Disable status of all analog input channels. The analog input module responds that all its channels are enabled (FF equals 1111 and 1111).

$aa5mm

Name	Set Channel Enable/Disable Status
Description	Set Enable/Disable status for all analog input channels
Syntax	$aa5mm(cr) $ is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module. (Always 01) 5 identifies the enable/disable channels command. mm (range 00-FF) are two hexadecimal characters. Each character is interpreted as 4 bits. The first 4-bit value represents the status of channels 7-4; the second 4-bit value represents the status of channels 3-0. A value of 0 means the channel is disabled, while a value of 1 means the channel is enabled. (cr) is the terminating character, carriage return (0Dh)
Response	!aa(cr) if the command is valid. ?aa(cr) if an invalid operation was entered. There is no response if the module detects a syntax error or communication error or if the address does not exist. ! delimiter indicating a valid command was received. ? delimiter indicating the command was invalid. aa (range 00-FF) represents the 2-character hexadecimal slave address of an ADAM-6000 module. (cr) is the terminating character, carriage return (0Dh)
Example	command: $01581(cr) response: !01(cr) The command enables/disables channels of the analog input module at address 01h. Hexadecimal 8 equals binary 1000, which enables channel 7 and disables channels 4, 5 and 6. Hexadecimal 1 equals binary 0001, which enables channel 0 and disables channels 1, 2 and 3.

#aaMH

Name	Read Maximum Value
Description	Read the maximum values from all analog input channels in a specified analog module
Syntax	#aaMH(cr) # is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module to be read. (Always 01) MH represents the read maximum value command. (cr) is the terminating character, carriage return (0Dh)
Response	>(data)(data)(data)(data)(data)(data)(data)(data)(data)(cr) if the command is valid. ?aa(cr) if an invalid operation was entered. There is no response if the module detects a syntax error or communication error or if the address does not exist. > delimiter indicating a valid command was received. ? delimiter indicating the command was invalid. aa (range 00-FF) represents the 2-character hexadecimal slave address of an ADAM-6000 module. (cr) is the terminating character, carriage return (0Dh)
Example	command: #01MH(cr)
Response	+10.000+10.000+10.000+10.000+10.000 +10.000 +10.000+10.000+10.000 The command asks the specific module at address 01h to send historic maximum value from analog input channels.
Note:	The latest data returned is the Average value of the preset channels in this module.

#aaMHn

Name	Read Maximum Value from channel N
Description	Read the maximum value from a specific channel in a specified module
Syntax	#aaMHn(cr) # is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module to be read. (Always 01) MH represents the read maximum value command. n (range 0-8) represents the specific channel you want to read the input data. (cr) is the terminating character, carriage return (0Dh)
Response	>(data)(cr) if the command is valid. ?aa(cr) if an invalid operation was entered. There is no response if the module detects a syntax error or communication error or if the address does not exist. > delimiter indicating a valid command was received. ? delimiter indicating the command was invalid. aa (range 00-FF) represents the 2-character hexadecimal slave address of an ADAM-6000 module. (cr) is the terminating character, carriage return (0Dh)
Example	command: #01MH2(cr) response: >+10.000 The command asks the specific module at address 01h to send historic maximum value from analog input channel 2.

#aaML

Name	Read Minimum Value
Description	Read the minimum values from all analog input channels in a specified module
Syntax	#aaML(cr) # is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module to be read. (Always 01) ML represents the read minimum value command. (cr) is the terminating character, carriage return (0Dh)
Response	>(data)(data)(data)(data)(data)(data)(data)(data)(data)(cr) if the command is valid. ?aa(cr) if an invalid operation was entered. There is no response if the module detects a syntax error or communication error or if the address does not exist. > delimiter indicating a valid command was received. ? delimiter indicating the command was invalid. aa (range 00-FF) represents the 2-character hexadecimal slave address of an ADAM-6000 module. (cr) is the terminating character, carriage return (0Dh)
Example	command: #01ML(cr) response:>+10.000+10.000+10.000+10.000 +10.000+10.000+10.000+10.000+10.000 The command asks the specific module at address 01h to send historic minimum value from all AI channels.
Note:	The latest data returned is the Average value of the preset channels in this module.

#aaMLn

Name	Read Minimum Value from channel N
Description	Read the minimum value from a specific analog input channel in a specified module
Syntax	#aaMLn(cr) # is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module to be read. (Always 01) ML represents the read minimum value command. n (range 0-8) represents the specific channel you want to read the input data. (cr) is the terminating character, carriage return (0Dh)
Response	>(data)(cr) if the command is valid. ?aa(cr) if an invalid operation was entered. There is no response if the module detects a syntax error or communication error or if the address does not exist. > delimiter indicating a valid command was received. ? delimiter indicating the command was invalid. aa (range 00-FF) represents the 2-character hexadecimal slave address of an ADAM-6000 module. (cr) is the terminating character, carriage return (0Dh)
Example	command: #01ML3(cr) response: >+10.000 The command asks the specific module at address 01h to send historic minimum value from analog input channel 3.

#aaDnd

Name	Set Digital Output
Description	Set the digital output status in ADAM-6000 analog input module.
Syntax	#aaDnd(cr) # is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module to be read. (Always 01) D represents the digital output setting command. n (range 0-1) represents the specific channel you want to set the output status. d (range 0-1) represents the status you want to set to the specific channel (cr) is the terminating character, carriage return (0Dh)
Response	!aa(cr) if the command is valid. ?aa(cr) if an invalid operation was entered. There is no response if the module detects a syntax error or communication error or if the address does not exist. ! delimiter indicating a valid command was received. ? delimiter indicating the command was invalid. aa (range 00-FF) represents the 2-character hexadecimal slave address of an ADAM-6000 module. (cr) is the terminating character, carriage return (0Dh)
Example	command: #01D01(cr) response: !01 The command set digital channel 0 "ON" status for the specific module at address 01h.

$aaBnn

Name	Read Analog Input Range Code from Channel N
Description	Returns the range code from a specified analog input channel in a specified module.
Syntax	$aaBnn(cr) $ is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module you want to interrogate. (Always 01) B is the Analog Input Range Code command. nn (range 00-07) represents the specific channel you want to read the range code. (cr) is the terminating character, carriage return (0Dh).
Response	!aa(data)(code) if the command is valid. ?aa(cr) if an invalid operation was entered. There is no response if the module detects a syntax error or communication error or if the address does not exist. ! delimiter indicating a valid command was received. ? delimiter indicating the command was invalid. (cr) is the terminating character, carriage return (0Dh). (code) is the range code read. Refer to the tables below to see the meaning of range code for different modules.

ADAM-6015 Analog Input Channel Range Code		
Range Code (Hex)	Range Code (Decimal)	Range Description
20	32	Pt 100 (α=0.00385) -50~150° C
21	33	Pt 100 (α=0.00385) 0~100° C
22	34	Pt 100 (α=0.00385) 0~200° C
23	35	Pt 100 (α=0.00385) 0~400° C
24	36	Pt 100 (α=0.00385) -200~200° C
25	37	Pt 100 (α=0.00392) -50~150° C
26	38	Pt 100 (α=0.00392) 0~100° C
27	39	Pt 100 (α=0.00392) 0~200° C

28	40	Pt 100 (α=0.00392) 0~400° C
29	41	Pt 100 (α=0.00392) -200~200° C
2A	42	Pt 1000 -40~160° C
2B	43	Balco 500 -30~120° C
2C	44	Ni 518 -80~100° C
2D	45	Ni 518 0~100° C

ADAM-6017 Analog Input Channel Range Code		
Range Code (Hex)	Range Code (Decimal)	Range Description
08	8	± 10 V
09	9	± 5 V
0A	10	± 1 V
0B	11	± 500 mV
0C	12	± 150 mV
0D	13	0 ~ 20 mV
07	7	4 ~ 20 mA

ADAM-6018 Analog Input Channel Range Code		
Range Code (Hex)	Range Code (Decimal)	Range Description
0E	14	Thermocouple J (0~760 ℃)
0F	15	Thermocouple K (0~1370 ℃)
10	16	Thermocouple T (-100~400 ℃)
11	17	Thermocouple E (0~1000 ℃)
12	18	Thermocouple R (500~1750 ℃)
13	19	Thermocouple S (500~1750 ℃)
14	20	Thermocouple B (500~1800 ℃)

Example command: $01B07(cr)
response: !0112

Since the ADAM-6018 is used, we can know the range code of channel 7 is 12, meaning **"Thermocouple R (500~1750° C)"**.

6.4.4 Analog Input Alarm Command Set Set (ADAM-6015, 6017, 6018)

Command Syntax	Command Name	Description
$aaCjAhs	Set Alarm Mode	Set the High/Low alarm in either Momentary or Latching mode
$aaCjAh	Read Alarm Mode	Returns the alarm mode for the specified channels
$aaCjAhEs	Enable/Disable Alarm	Enables/Disables the high/low alarm of the specified channels
$aaCjCh	Clear Latch Alarm	Resets a latched alarm
$aaCjAhCCn	Set Alarm Connection	Connects the High/Low alarm of a specified input channel to interlock with a specified output channel
$aaCjRhC	Read Alarm Connection	Returns the alarm configuration of a specified input channel
$aaCjAhU	Set Alarm Limit	Sets the High/Low alarm limit value to a specified channel
$aaCjRhU	Read Alarm Limit	Returns the High/Low alarm limit value of the specified channel
$aaCjS	Read Alarm Status	Reads whether an alarm occurred in the specified channel

$aaCjAhs

Name Set Alarm Mode

Description Sets the High/Low alarm of the specified input channel in the addressed ADAM-6000 module to either Latching or Momentary mode.

Syntax $aaCjAhs(cr)

$ is a delimiter character.

aa (range 00-FF) represents the 2-character hexadecimal slave network address of an ADAM-6000 module.

(Always 01)

Cj identifies the desired channel j (j : 0 to 7).

A is the Set Alarm Mode command.

h indicates alarm types (H = High alarm, L = Low alarm)

s indicates alarm modes (M = Momentary mode,

L = Latching mode)

(cr) represents terminating character, carriage return (0Dh)

Response !aa(cr) if the command was valid

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

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

! delimiter indicating a valid command was received.

aa represents the 2-character hexadecimal slave address of the corresponding ADAM-6000 module.

(cr) represents terminating character, carriage return (0Dh)

Example command: $01C1AHL(cr)

response: !01(cr)

Channel 1 of the ADAM-6000 module at address 01h is instructed to set its High alarm in Latching mode.

The module confirms that the command has been received.

$aaCjAh

Name	Read Alarm Mode
Description	Returns the alarm mode for the specified channel in the specified ADAM-6000 module.
Syntax	$aaCjAh(cr) $ is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of an ADAM-6000 module. (Always 01) Cj identifies the desired channel j (j : 0 to 7). A is the Read Alarm Mode command. h indicates the alarm types (H = High alarm, L = Low alarm) (cr) represents terminating character, carriage return (0Dh)
Response	!aas(cr) if the command was valid ?aa(cr) if an invalid operation was entered. There is no response if the system detects a syntax error or communication error or if the address does not exist. ! delimiter indicating a valid command was received. aa represents the 2-character hexadecimal slave address of the corresponding ADAM-6000 module. s indicates alarm modes (M = Momentary mode, L = Latching mode) (cr) represents terminating character, carriage return (0Dh)
Example	command: $01C1AL(cr) response: !01M(cr) Channel 1 of the ADAM-6000 module at address 01h is instructed to return its Low alarm mode. The system responds that it is in Momentary mode.

$aaCjAhEs

Name	Enable/Disable Alarm
Description	Enables/Disables the High/Low alarm of the specified input channel in the addressed ADAM-6000 module
Syntax	$aaCjAhEs(cr) $ is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of an ADAM-6000 module. (Always 01) Cj identifies the desired channel j (j : 0 to 7). AhEs is the Set Alarm Mode command. h indicates alarm type (H = High alarm, L = Low alarm) s indicates alarm enable/disable (E = Enable, D = Disable) (cr) represents terminating character, carriage return (0Dh)
Response	!aa(cr) if the command was valid ?aa(cr) if an invalid operation was entered. There is no response if the system detects a syntax error or communication error or if the address does not exist. ! delimiter indicating a valid command was received. aa represents the 2-character hexadecimal slave address of the corresponding ADAM-6000 module. (cr) represents terminating character, carriage return (0Dh)
Example	command: $01C1ALEE(cr) response: !01(cr) Channel 1 of the ADAM-6000 module at address 01h is instructed to enable its Low alarm function. The module confirms that its Low alarm function has been enabled.
Note:	An analog input module requires a maximum of 2 seconds after it receives an Enable/Disable Alarm command to let the setting take effect.

$aaCjCh

Name	Clear Latch Alarm
Description	Sets the High/Low alarm to OFF (no alarm) for the specified input channel in the addressed ADAM-6000 module
Syntax	$aaCjCh(cr) $ is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave network address of an ADAM-6000 module. (Always 01) Cj identifies the desired channel j (j : 0 to 7). Ch is the Clear Latch Alarm command. h indicates alarm type (H = High alarm, L = Low alarm) (cr) represents terminating character, carriage return (0Dh)
Response	!aa(cr) if the command was valid ?aa(cr) if an invalid operation was entered. There is no response if the system detects a syntax error or communication error or if the address does not exist. ! delimiter indicating a valid command was received. aa represents the 2-character hexadecimal slave address of the corresponding ADAM-6000 module. (cr) represents terminating character, carriage return (0Dh)
Example	command: $01C1CL(cr) response: !01(cr) Channel 1 of the ADAM-6000 module at address 01h is instructed to set its Low alarm state to OFF. The system confirms it has done so accordingly.

$aaCjAhCCn

Name	Set Alarm Connection
Description	Connects the High/Low alarm of the specified input channel to interlock the specified digital output in the addressed ADAM-6000 module
Syntax	$aaCjAhCCn(cr) $ is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of an ADAM-6000/TCP module. (Always 01) Cj identifies the desired analog input channel j (j : 0 to 7). AhC is the Set Alarm Connection command. h indicates alarm type (H = High alarm, L = Low alarm) Cn identifies the desired digital output channel n (n : 0 to 1). To disconnect the digital output, n should be set as ˜*™. (cr) represents terminating character, carriage return (0Dh)
Response	!aa(cr) if the command was valid ?aa(cr) if an invalid operation was entered. There is no response if the system detects a syntax error or communication error or if the address does not exist. ! delimiter indicating a valid command was received. aa represents the 2-character hexadecimal slave address of the corresponding ADAM-6000 module. (cr) represents terminating character, carriage return (0Dh)
Example	command: $01C1ALCC0(cr) response: !01(cr) Channel 1 of the ADAM-6000 module at address 01h is instructed to connect its Low alarm to the digital output of channel 0 in the specific module. The system confirms it has done so accordingly.

$aaCjRhC

Name	Read Alarm Connection
Description	Returns the High/Low alarm limit output connection of a specified input channel in the addressed module
Syntax	$aaCjRhC(cr) $ is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of an ADAM-6000 module. (Always 01) Cj identifies the desired analog input channel j (j : 0 to 7). RhC is the Read Alarm Connection command. h indicates alarm type (H = High alarm, L = Low alarm) (cr) represents terminating character, carriage return (0Dh)
Response	!aaCn(cr) if the command was valid ?aa(cr) if an invalid operation was entered. There is no response if the system detects a syntax error or communication error or if the address does not exist. ! delimiter indicating a valid command was received. aa represents the 2-character hexadecimal slave address of the corresponding ADAM-6000 module. Cn identifies the desired digital output channel n (n : 0 to 1) whether interlock with the alarm of the specific analog input channel. If the values of n are "*", the analog input has no connection with a digital output point. (cr) represents terminating character, carriage return (0Dh)
Example	command: $01C1RLC(cr) response: !01C0(cr) Channel 1 of the ADAM-6000 module at address 01h is instructed to read its Low alarm output connection. The system responds that the Low alarm output connects to the digital output at channel 0 in the specific module.

$aaCjAhU

Name	Set Alarm Limit
Description	Sets the High/Low alarm limit value for the specified input channel of a specified ADAM-6000 module.
Syntax	$aaCjAhU(data)(cr) $ is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of an ADAM-6000 module. (Always 01) Cj identifies the desired analog input channel j (j : 0 to 7). AhU is the Set Alarm Limit command. h indicates alarm type (H = High alarm, L = Low alarm) (data) represents the desired alarm limit setting. The format is always in engineering units. (cr) represents terminating character, carriage return (0Dh)
Response	!aa(cr) if the command was valid ?aa(cr) if an invalid operation was entered. There is no response if the system detects a syntax error or communication error or if the address does not exist. ! delimiter indicating a valid command was received. aa represents the 2-character hexadecimal slave address of the corresponding ADAM-6000 module. (cr) represents terminating character, carriage return (0Dh)
Example	command: $01C1AHU+080.00(cr) response: !01(cr) The high alarm limit of the channel 1 in the specific module at address 01h is been set +80. The system confirms the command has been received.
Note:	An analog input module requires a maximum of 2 seconds after it receives a Set Alarm Limit command to let the settings take effect.

$aaCjRhU

Name	Read Alarm Limit
Description	Returns the High/Low alarm limit value for the specified input channel in the addressed ADAM-6000 module
Syntax	$aaCjRhU(cr) $ is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of an ADAM-6000 module. (Always 01) Cj identifies the desired analog input channel j (j : 0 to 7). RhU is the Read Alarm Limit command. h indicates alarm type (H = High alarm, L = Low alarm) (cr) represents terminating character, carriage return (0Dh)
Response	!aa(data)(cr) if the command was valid ?aa(cr) if an invalid operation was entered. There is no response if the system detects a syntax error or communication error or if the address does not exist. ! delimiter indicating a valid command was received. aa represents the 2-character hexadecimal slave address of the corresponding ADAM-6000 module. (data) represents the desired alarm limit setting. The format is always in engineering units. (cr) represents terminating character, carriage return (0Dh)
Example	command: $01C1RHU(cr) response: !01+2.0500(cr) Channel 1 of the ADAM-6000 module at address 01h is configured to accept 5V input. The command instructs the system to return the High alarm limit value for that channel. The system responds that the High alarm limit value in the desired channel is 2.0500 V.

$aaCjS

Name	Read Alarm Status
Description	Reads whether an alarm occurred to the specified input channel in the specified ADAM-6000 module
Syntax	$aaCjS(cr) $ is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of an ADAM-6000 module. (Always 01) Cj identifies the desired analog input channel j (j : 0 to 7). S is the Read Alarm Status command. (cr) represents terminating character, carriage return (0Dh)
Response	!aahl(cr) if the command was valid ?aa(cr) if an invalid operation was entered. There is no response if the system detects a syntax error or communication error or if the address does not exist. ! delimiter indicating a valid command was received. aa represents the 2-character hexadecimal slave address of the corresponding ADAM-6000 module. h represents the status of High alarm. “1” means the High alarm occurred, ‘0” means it did not occur. l represents the status of Low alarm. ˜1” means the Low alarm occurred, ˜0” means it did not occur. (cr) represents terminating character, carriage return (0Dh)
Example	command: $01C1S(cr) response: !0101(cr) The command asks the module at address 01h to return its alarm status for channel 1. The system responds that a High alarm has not occurred, but the Low alarm has occurred.

6.4.5 Universal I/O Command Set
(ADAM-6024)

Command Syntax	Command Name	Description
$aa5mm	Set AI Channels Enable/Disable Status	Set Enable/Disable status to all analog input channels
$aa6	Read AI Channels Enable/Disable Status	Return the Enable/Disable status of all analog input channels
#aa	Read AI Values from all channels	Return the input values from all analog input channels
#aacc	Read AI Value from a specified channel	Return the input value from a specified analog input channel
$aaDcc	Read AO Startup Value	Read startup output value from a specific analog output channel
$aaDcchhh	Set AO Startup Value	Set startup output value to a specific analog output channel
#aaccdd.ddd	Write AO Values	Write value to the specific analog output channel
#aa7	Read DI Values from all channels	Return the input values from all digital input channels
#aaccdd	Write DO Values	Write value to a single specific digial output channel or to all digital output channels
$aaBnn	Read AI Channel Range Code	Return the channel range code form specific analog input channel
$aaCnn	Read AO Channel Range Code	Return the channel range code form specific analog output channel

$aa5mm

Name Set AI Channels Enable/Disable Status

Description Set Enable/Disable status for all analog input channels of the specified module

Syntax $aa5mm(cr)
$ is a delimiter character.
aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module. (Always 01)
5 identifies the enable/disable channels command.
mm (range 00-FF) are two hexadecimal characters.
Each character is interpreted as 4 bits.
The first 4-bit value represents the status of channels 5-4; the second 4-bit value represents the status of channels

3-0. A value of 0 means the channel is disabled, while a value of 1 means the channel is enabled.
(cr) is the terminating character, carriage return (0Dh)

Response
!aa(cr) if the command is valid.
?aa(cr) if an invalid operation was entered.
There is no response if the module detects a syntax error or communication error or if the address does not exist.
! delimiter indicating a valid command was received.
? delimiter indicating the command was invalid.
aa (range 00-FF) represents the 2-character hexadecimal slave address of an ADAM-6000 module.
(cr) is the terminating character, carriage return (0Dh)

Example
command: $01521(cr)
response: !01(cr)
The command enables/disables channels of the analog input module at address 01h. Hexadecimal 2 equals binary 0010, which enables channel 5 and disables channels 4. Hexadecimal 1 equals binary 0001, which enables channel 0 and disables channels 1, 2 and 3.

$aa6

Name
Read AI Channels Enable/Disable Status

Description
Asks a specified module to return the Enable/Disable status of all analog input channels

Syntax
$aa6(cr)
$ is a delimiter character.
aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module you want to interrogate. (Always 01)
6 is the read channels status command.
(cr) is the terminating character, carriage return (0Dh)

Response
!aamm(cr) if the command is valid.
?aa(cr) if an invalid operation was entered.
There is no response if the module detects a syntax error or communication error or if the address does not exist.
! delimiter indicating a valid command was received.

? delimiter indicating the command was invalid.
aa (range 00-FF) represents the 2-character hexadecimal slave address of an ADAM-6000 module.
mm are two hexadecimal values.
Each value is interpreted as 4 bits.
The first 4-bit value represents the status of channels 5-4, the second 4 bits represents the status of channels 3-0.
A value of 0 means the channel is disabled, while a value of 1 means the channel is enabled.
(cr) is the terminating character, carriage return (0Dh)

Example
command: $016(cr)
response: !013F(cr)
The command asks the specific module at address 01h to send the Enable/Disable status of all analog input channels.
Channels 0~5 are all enabled. (3F equals 0011 and 1111).

#aa

Name
Read AI Values from All Channels

Description
Returns the input data from all analog input channels in a specific module.

Syntax
#aa(cr)
is a delimiter character.
aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module you want to interrogate. (Always 01)
(cr) is the terminating character, carriage return (0Dh).

Response
>(data)(data)(data)(data)(data)(data)(cr) if command valid.
?aa(cr) if an invalid operation was entered.
There is no response if the module detects a syntax error or communication error or if the address does not exist.
> delimiter indicating a valid command was received.
? delimiter indicating the command was invalid.
(cr) is the terminating character, carriage return (0Dh).

Example
command: #01(cr)
response:>+10.000+10.000+10.000+10.000+10.000+10.00
0

#aacc

Name	Read AI Value from One Channel
Description	Returns the input data from a specified analog input channel in a specified module.
Syntax	#aacc(cr) # is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module you want to interrogate. (Always 01) cc (range 00-05) represents the specific channel you want to read the input data. (cr) is the terminating character, carriage return (0Dh).
Response	>(data)(cr) if the command is valid. ?aa(cr) if an invalid operation was entered. There is no response if the module detects a syntax error or communication error or if the address does not exist. > delimiter indicating a valid command was received. ? delimiter indicating the command was invalid. (cr) is the terminating character, carriage return (0Dh).
Example	command: #0103(cr) response: >+10.000 Analog input channel 3 of the ADAM-6024 module at address 01h responds with an input value +10.000.

$aaDcc

Name	Read AO Startup Value from One Channel
Description	Returns the startup value from a specified analog output channel in a specified module.
Syntax	$aaDcc(cr) $ is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module you want to interrogate. (Always 01) D represents the analog output channel startup command. cc (range 00-01) represents the specific channel you want to read the startup value. (cr) is the terminating character, carriage return (0Dh).

Response
!aahhh(cr) if the command is valid.
?aa(cr) if an invalid operation was entered.
There is no response if the module detects a syntax error or communication error or if the address does not exist.
! delimiter indicating a valid command was received.
? delimiter indicating the command was invalid.
aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module you want to interrogate. (Always 01)
hhh (range 000-FFF) represents the 3-character hexadecimal startup value of the specific AO channel.
(cr) is the terminating character, carriage return (0Dh).

Example
command: $01D01(cr)
response: !01FFF(cr)
Startup value for analog output channel 1 of the ADAM-6024 module at address 01h responds with a value +10.000. (The AO range of channel 1 is 0~10V)

$aaDcchhh

Name
Set AO Startup Value to One Channel

Description
Set the startup value to a specified analog output channel in a specified module.

Syntax
$aaDcchhh(cr)
$ is a delimiter character.
aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module you want to interrogate. (Always 01)
D represents the analog output channel startup command.
cc (range 00-01) represents the specific channel you want to set the startup value.
hhh (range 000-FFF) represents the 3-character hexadecimal startup value of the specific AO channel.
(cr) is the terminating character, carriage return (0Dh).

Response
!aa(cr) if the command is valid.
?aa(cr) if an invalid operation was entered.
There is no response if the module detects a syntax error or

communication error or if the address does not exist.
! delimiter indicating a valid command was received.
? delimiter indicating the command was invalid.
aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module you want to interrogate. (Always 01)
(cr) is the terminating character, carriage return (0Dh).

Example
command: $01D01FFF(cr)
response: !01(cr)
Startup value for analog output channel 1 of the ADAM-6024 module at address 01h is set with a value +10.000. (The AO range of channel 1 is 0~10V)

#aaccdd.ddd

Name
Write AO Value to One Channel

Description
Write output value to a specified analog output channel in a specified module.

Syntax
#aaccdd.ddd(cr)
is a delimiter character.
aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module you want to interrogate. (Always 01)
cc (range 00-01) represents the specific channel you want to write output value.
dd.ddd (in engineering unit) represents the analog output value of the specific analog output channel.
(cr) is the terminating character, carriage return (0Dh).

Response
>(cr) if the command is valid.
?aa(cr) if an invalid operation was entered.
There is no response if the module detects a syntax error or communication error or if the address does not exist.
> delimiter indicating a valid command was received.
? delimiter indicating the command was invalid.
aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module you want to interrogate. (Always 01)
(cr) is the terminating character, carriage return (0Dh).

Example	command: #010105.555(cr) response: >(cr) Value for analog output channel 1 of the ADAM-6024 module at address 01h is set with a value +05.555.

$aa7

Name	Read DI Channel Status
Description	This command requests that the specified module return the status of its digital input channels.
Syntax	$aa7(cr) $ is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave network address of the ADAM-6000 module. (Always 01) 7 is the Digital Data In command. (cr) is the terminating character, carriage return (0Dh)
Response	!aa(data)(cr) if the command is valid. ?aa(cr) if an invalid operation was entered. There is no response if the module detects a syntax error or communication error or if the address does not exist. ! delimiter indicating a valid command was received. ? delimiter indicating the command was invalid. aa (range 00-FF) represents the 2-character hexadecimal slave network address of an ADAM-6000 module. (data) a 2-character hexadecimal value representing the values of the digital input module. (cr) is the terminating character, carriage return (0Dh)
Example	command: $017(cr) response: !0101 (cr) The command asks the specific module at address 01h to return the values of all DI channels. DI channel 0 is ON and channel 1 is OFF since the return value is 1 (01b).

#aaccdd

Name	Write DO Value to a Single Channel or All Channels
Description	This command sets a single or all digital output channels to the specific module.
Syntax	#aaccdd(cr)

is a delimiter character.
aa (range 00-FF) represents the 2-character hexadecimal slave network address of the ADAM-6000 module. (Always 01)
cc is used to indicate which channel(s) you want to set.
Writing to all channels (byte):
Both characters should be equal to zero.
Writing to a single channel (bit):
First character is 1.
Second character indicates channel number (00-01).
dd is the hexadecimal representation of the digital output value(s).
Writing to all channels (byte)
The digital equivalent of these hexadecimal characters represent the channels' values.
Writing to a single channel (bit)
First character is always 0. The value of the second character is either 0 or 1. (The DO value)

Response

>(cr) if the command was valid.
?aa(cr) if an invalid command has been issued.
There is no response if the module detects a syntax error or communication error or if the address does not exist.
> delimiter indicating a valid command was received.
? delimiter indicating the command was invalid.
aa (range 00-FF) represents the 2-character hexadecimal slave network address of a module that is responding.
(cr) is the terminating character, carriage return (0Dh)

Example

command: #011101(cr)
response: >(cr)
An output bit with value 1 is sent to digital output channel 1 of a module at address 01h.
DO channel 1 of the specific module is set to ON.
command: #010002(cr)
response: >(cr) An output byte with value 02 (10b) is sent to the module at address 01h.
DO channels 1 is set to ON, and channel 0 is set to OFF.

$aaBnn

Name	Read Analog Input Range Code from Channel N
Description	Returns the range code from a specified analog input channel in a specified module.
Syntax	$aaBnn(cr) $ is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module you want to interrogate. (Always 01) B is the Analog Input Range Code command. nn (range 00-07) represents the specific channel you want to read the range code. (cr) is the terminating character, carriage return (0Dh).
Response	!aa(data)(code) if the command is valid. ?aa(cr) if an invalid operation was entered. There is no response if the module detects a syntax error or communication error or if the address does not exist. ! delimiter indicating a valid command was received. ? delimiter indicating the command was invalid. (cr) is the terminating character, carriage return (0Dh). (code) is the range code read. Refer to the tables below to see the meaning of range code for different modules.

ADAM-6024 Analog Input Channel Range Code		
Range Code (Hex)	Range Code (Decimal)	Range Description
07	7	4 ~ 20 mA
08	9	± 10 V
0D	13	0 ~ 20 mA

Example command: $01B01(cr)

response: !010D

We can know the range code of channel 1 is 0D, meaning **"0~24 mA"**.

$aaCnn

Name	Read Analog Output Range Code from Channel N
Description	Returns the range code from a specified analog output channel in a specified module.
Syntax	$aaCnn(cr) $ is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module you want to interrogate. (Always 01) C is the Analog Output Range Code command. nn (range 00-07) represents the specific channel you want to read the range code. (cr) is the terminating character, carriage return (0Dh).
Response	!aa(data)(code) if the command is valid. ?aa(cr) if an invalid operation was entered. There is no response if the module detects a syntax error or communication error or if the address does not exist. ! delimiter indicating a valid command was received. ? delimiter indicating the command was invalid. (cr) is the terminating character, carriage return (0Dh). (code) is the range code read. Refer to the tables below to see the meaning of range code for different modules.

ADAM-6024 Analog Output Channel Range Code		
Range Code (Hex)	Range Code (Decimal)	Range Description
00	0	0 ~ 20 mA
01	1	4 ~ 20 mA
02	2	0 ~ 10 V

Example command: $01C01(cr)

response: !0102

We can know the range code of channel 1 is 02, meaning **"0~10 V"**.

6.4.6 Digital Input/Output Command Set
(ADAM-6050, 6051, 6052, 6060, 6066)

Command Syntax	Command Name	Description
$aa6	Read Channels	Asks a specified input module to return the status of all channels
#aabb	Write Digital Output	Writes specified values to either a single channel or all channels simultaneously
$aaJCFFFFssmm	Read DI Channel Counter Value	Returns the counter value from specified DI channels in a specified module

$aa6

Name Read Channel Status

Description This command requests that the specified ADAM-6000 module return the status of its digital input channels

Syntax $aa6(cr)
$ is a delimiter character.
aa (range 00-FF) represents the 2-character hexadecimal slave network address of the ADAM-6000 module. (Always 01)
6 is the Digital Data In command.
(cr) is the terminating character, carriage return (0Dh)

Response !aa00(data)(data)(data)(data)(cr) if the command is valid.
?aa(cr) if an invalid operation was entered.
There is no response if the module detects a syntax error or communication error or if the address does not exist.
! delimiter indicating a valid command was received.

? delimiter indicating the command was invalid.

aa (range 00-FF) represents the 2-character hexadecimal slave network address of an ADAM-6000 module.
(data) a 2-character hexadecimal value representing the values of the digital input module.

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

Example command: $016(cr)

response: !01000FFD(cr) The command asks the specific module at address 01h to return the values of all channels.

The first 2-character portion of the response (exclude the "!" character) indicates the address of the ADAM-6000 module. The second 2-character portion of the response is reserved, and will always be 00 currently.

The 5~8 characters of the response indicate 15~12, 11~8, 7~4 and 3~0 channels value. In this example, channels 15~12 are all OFF since the 5 characters of the response is "0" (0000b). Channels 11~8 and 7~4 are all ON, since the 6 and 7 characters of the response are both "F" (1111b). Channel 0, 2, 3 are ON and channel 1 is OFF, since the 8 characters of the response is "D" (1101b).

#aabb(data)

Name Write Digital Output

Description This command sets a single or all digital output channels to the specific ADAM-6000 module.

Syntax #aabb(data)(cr)

is a delimiter character.

aa (range 00-FF) represents the 2-character hexadecimal slave network address of the ADAM-6000 module.
(Always 01)

bb is used to indicate which channel(s) you want to set.

Writing to all channels (write a byte): both characters should be equal to zero (BB=00).

Writing to a single channel (write a bit): first character is 1, second character indicates channel number which can range

from 0h to Fh.

(data) is the hexadecimal representation of the DO value(s).

When writing to a single channel (bit)

The first character is always 0. The value of the second character is either 0 or 1.

When writing to all channels (byte)

2 or 4-characters are significant. The digital equivalent of these hexadecimal characters represent the channels' values.

Response >(cr) if the command was valid.

?aa(cr) if an invalid command has been issued.

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

> delimiter indicating a valid command was received.

? delimiter indicating the command was invalid.

aa (range 00-FF) represents the 2-character hexadecimal slave network address of a module that is responding.

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

Example command: #011201(cr)

response: >(cr)

An output bit with value 1 is sent to channel 2 of a digital output module at address 01h.

Channel 2 of the digital output module is set to ON.

command: #010012(cr)

response: >(cr)

An output byte with value 12h (00010010) is sent to the digital output module at address 01h. Channels 1and 4 will be set to ON, and all other channels will be set to OFF.

$aaJCFFFFssmm

Name	Read DI Channel Counter Value
Description	Returns the counter value from specified DI channels in a specified module.
Syntax	$aaJCFFFFssmm(cr) $ is a delimiter character. aa (range 00-FF) represents the 2-character hexadecimal slave address of the ADAM-6000 module you want to interrogate. (Always 01) JCFFFF is the Digital Input Channel Counter Value command. ss (range 00-07) represents the specific start channel you want to read the counter value. mm (range 00-07) represents the total channel numbers you want to read the counter value. (cr) is the terminating character, carriage return (0Dh).
Response	>aa(data) if the command is valid. ?aa(cr) if an invalid operation was entered. There is no response if the module detects a syntax error or communication error or if the address does not exist. > delimiter indicating a valid command was received. ? delimiter indicating the command was invalid. (data) is the counter value read. (cr) is the terminating character, carriage return (0Dh).
Example	Command: $01JCFFFF0001(cr) Response: >010000000A(cr) The command requests the module at address 01h to return count value from channel 0. (the first read channel is 0, represent by “00”, and only one channel is read) That module return the count value 0000000A(h) from channel 0.
Example	Command: $01JCFFFF0C02(cr) Response: >010000000A00000001(cr) The command requests the module at address 01h to return count value from channel 12 and 13. (The first read channel is 12, represent by “0C”, and two channels are read)

That module return the count value 0000000A(h) from channel 12 and 00000001(h) from channel 13.

CHAPTER 7

Graphic Condition Logic (GCL)

Sections include:

- Overview
- GCL Configuration Environment
- Four Stages of One Logic Rule
- Logic Cascade and Feedback
- Logic and Online Monitoring
- Typical Application with GCL

Chapter 7 Graphic Condition Logic(GCL)

7.1 Overview

In a traditional control and data acquisition system, there must be one controller to manage the system. Remote I/O modules like the ADAM-6000 modules, only acquire data from sensors, or generate signal to control other devices or equipment. There must be a computer (or a controller, such as PLC) responsible to get the data from the input modules, manipulate the data, execute logic operation and process depending on the input data, and generate output data to the output modules based on the logic decision.

The computer (or controller) and remote I/O modules form a complete control system within the same network. The complexity of logic operation and process depend on the application, and it is implemented by the program written on the computer (or controller). There are plenty of software applications to write programs. Examples are C language, Microsoft Visual Studio for computer, and Ladder language for PLC controller.

In many applications, the logic operation and process is not very difficult that it seems not so necessary to implement a computer or controller which are too powerful than needed. Now, ADAM-6000 modules feature logic operation and process ability by the new design --- ***Graphic Logic Condition (GCL)***. This feature makes the ADAM-6000 modules become a smart I/O module that it can play as a standalone control system.

People can define the logic operation and process rules in the ADAM.NET Utility and download the rules to the ADAM-6000 modules. Then ADAM-6000 modules will execute the logic rules to process different action depending on the input conditions. **With GCL enabled, a computer (or a controller) can be removed from the control system since the ADAM-6000 modules can play as controller by themselves.**

The configuration environment for GCL in ADAM.NET Utility is completely graphical, making it very easy and intuitive to complete the logic rule configuration. After completing the logic rule configuration and download, engineers can see the real-time execution situation and input value in ADAM.NET Utility on line. We will introduce these features in more detail by following content.

Note: To utilize GCL function, you need to upgrade firmware version of your ADAM-6000 module to 4.x or later.

7.2 GCL Configuration Environment

As we have mentioned in Section 5.3.3, when you click the item list representing the ADAM-6000 module in the Module Tree Display area, there will be two item lists appearing below: **All Channel Configuration** and **GCL Configuration** list item. You can configure all GCL related setting by clicking the **GCL Configuration** item list.

For the two features Peer-to-Peer and GCL, only one of them can be enable at one time. If you enable Peer-to-Peer function before, when you click the **GCL Configuration** and launch the GCL configuration environment, you will find that it is disabled by default. Once you click the **Program GCL** button (refer to table below) to enable the GCL feature, the Peer-to-Peer function will be disabled. Below is the image (Figure 7.1) of the Status Display area after you have clicked the **GCL Configuration** item list.

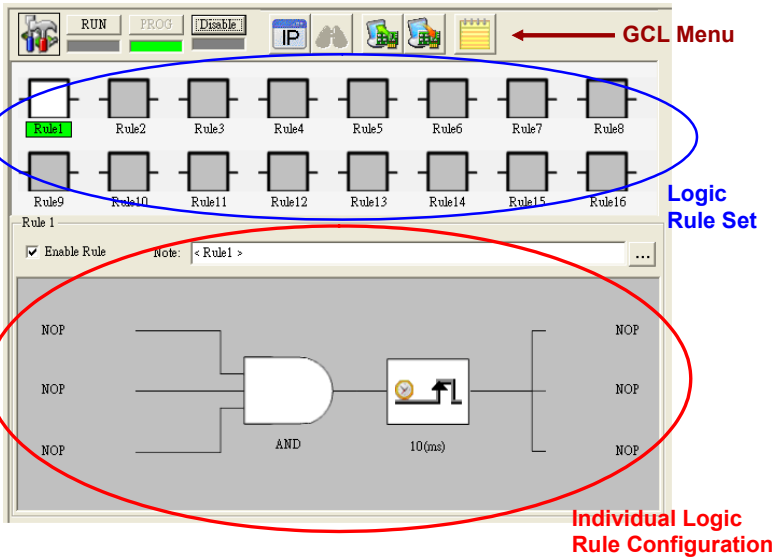


Figure 7.1: GCL Configuration Environment

At the top of the Status Display area is the **GCL Menu** area. Refer to the table below to see function for each graphical icon on the menu:

Icon	Function	Description
	Current Status	This icon shows current GCL status. The status represented in the Icon cell is the **Disable**, **Programming** and **Running** mode (From top to button) Note: You cannot enable Peer-to-Peer/Data Stream function and GCL function at the same time. So if you want to enable GCL, Peer-to-Peer and Data Stream function will be disabled automatically.
	Run GCL	Select the **Running** mode. If this mode is chosen, the LED below the button is lit.
	Program GCL	Select the **Programming** mode. If this mode is chosen, the LED below the button is lit.
	Disable GCL	Select the **Disable** mode. If this mode is chosen, the LED below the button is lit.
	IP Table Configuration	Click this button to configure IP table. IP table can used to define the destination for output. Only available in the **Programming** mode.
	Monitoring	Enable Online Monitoring. Only available in the **Running** mode.
	Upload Project	Upload GCL configurations from ADAM-6000 module to computer. Only available in the **Programming** mode.
	Download Project	Download current GCL configurations to the ADAM-6000 module. Only available in the **Programming** mode.
	Project Content	Click this button to show current GCL configurations. You can also save current configurations into a file, or load previous configuration from a specific file.

Below the **GCL Menu** area is the **Logic Rule Set** area. There are 16 logic rules available on one ADAM-6000 module, so you can see 16 logic rule icons here. Simply click the logic rule icon to configure that rule. For example, if you want to configure rule 12, just click the logic rule icon with text "Rule 12" below. The text background color of the selected logic rule icon will become green.

At the bottom of the Status Display area is the **Individual Logic Rule Configuration** area. After you have selected the rule you want to configure in the **Logic Rule Set** area, click the **Enable Rule** check box to enable that logic rule. The color of that logic rule icon will become white after you enable it. You can write some description for that logic rule by clicking the button next to the **Note** text box. There are four stages for one logic rule: **Input Condition**, **Logic**, **Execution** and **Output**, which are all displayed by graphical icon in the **Individual Logic Rule Configuration** area. Refer to Figure 7.2 below. You can simply click the graphical icon to configure each stage and one related configuration window will pop-up.

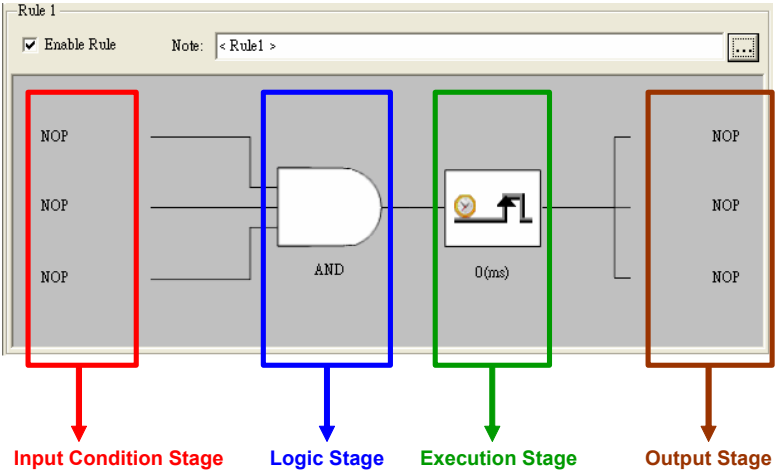


Figure 7.2: Four Stages for One Logic Rule

Input Condition Stage	Option	Description	Condition	Section
	NoOperation	No Opeartion	N/A	7.3.1
	AI	Local AI channel value	>, =, <	
	DI	Local DI channel value	True, False	
	DI_Counter	Local counter input channel value	>, =, <	
	DI_Frequency	Local frequency input channel value	>, =, <	
	Timer	Local internal Timer value	>, =, <	
	AuxFlag	Local internal Flag value	True, False	
	DO	Local DO channel value	True, False	
	Coutner	Local internal counter value	>, =, <	

Logic Stage	Option	Description	Section
	AND	AND operation	7.3.2
	OR	OR operation	
	NAND	NAND operation	
	NOR	NOR operation	
Execution Stage	Option	Description	Section
	Execution_ Period	Define the execution time for this logic rule	7.3.3
	SendTo NextRule	Combine output of this logic rule to next logic rule to form Logic Cascade	

	Option	Description	Section
Output Stage	NoOperation	No Opeartion	7.3.4
	AO	Local or remote AO channel value	
	DO	Local or remote DO channel value	
	DI_Counter	Local or remote counter input channel setting	
	DO_Pulse	Local or remote pulse output channel setting	
	Timer	Local internal Timer setting	
	AuxFlag	Local or remote internal Flag value	
	RemoteMessage	Remote message	
	Counter	Local internal counter setting	

7.3 Configure Four Stages of One Logic Rule

7.3.1 Input Condition Stage

The Input Condition stage is a logic condition decision for the input data. The decision result will be logic ***True*** or ***False***, sending to the Logic stage for logic operation. Take analog input mode as example, you can define the condition as if the analog input value is greater than a specific value (the limit). So when the input value becomes greater than the limit, the input stage will transfer ***True*** to the Logic stage. Otherwise, it will transfer ***False*** to the Logic stage.

When you click the Input Condition stage icon, you should see a dialog window similar to Figure 7.3 below. You can choose the input mode by the **Mode** combo box. The default mode is **NoOperation**, meaning there is no input condition. You can choose local analog input channel **(AI)**, local digital input channel **(DI)**, local counter input channel **(DI_Counter)**, local frequency input channel **(DI_Frequency)**, internal timer (**Timer**), internal flag (**AuxFlag**) , local digital output channel **(DO)** and internal counter **(Counter)** as the input mode. After you choose the appropriate input mode and complete all related setting, click the **OK** button. That Input Condition stage icon will change its pattern to present the current condition. We will describe each mode in more detail below.

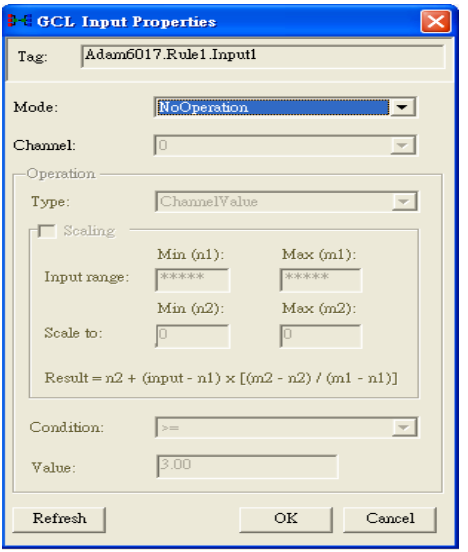


Figure 7.3: Input Condition Stage Configuration

Local Analog Input Channel (AI)

Below are the steps to configure analog input condition:

1. After you choose **AI** as input mode, select the channel by the **Channel** combo box.
2. In the **Operation** area, you can define the input condition operation. Select the analog input type by the **Type** combo box. There are two input types you can choose for analog input: If you select **ChannelValue**, the current value of the selected analog input channel is used as input for condition. If you select **Deviation**, the deviation (Dividing difference between present sample value and previous sample value by the total range value) of the selected analog input channel is used as input for condition.
3. Select the appropriate condition for that input channel by the **Condition** combo box and the **Value** text box. Refer to the examples in the table below to see how to define analog input condition:

Channel	Type	Condition	Value	Description
0	Channel-Value	>=	5	If the value of analog channel 0 is more than or equal to 5, the condition result is logic **True**. Otherwise, the condition result is logic **False**.
2	Channel-Value	=	3.2	If the value of analog channel 2 equals to 3.2, the condition result is logic **True**. Otherwise, the condition result is logic **False**.
3	Channel-Value	<=	1.7	If the value of analog channel 3 is less than or equal to 1.7, the condition result is logic **True**. Otherwise, the condition result is logic **False**.
5	Deviation	N/A	20	If the deviation of analog channel 5 is greater than 20%, the condition result is logic **True**. Otherwise, the condition result is logic **False**.

The analog input will read voltage (or current) from the channel we specified. Usually, the voltage (or current) value can represent the real-world

physical unit value (we call it *engineer unit value*) and there is linear relationship between the voltage (or current) value and the engineer unit value. For example, the current and the engineer unit value have linear relationship as shown below:

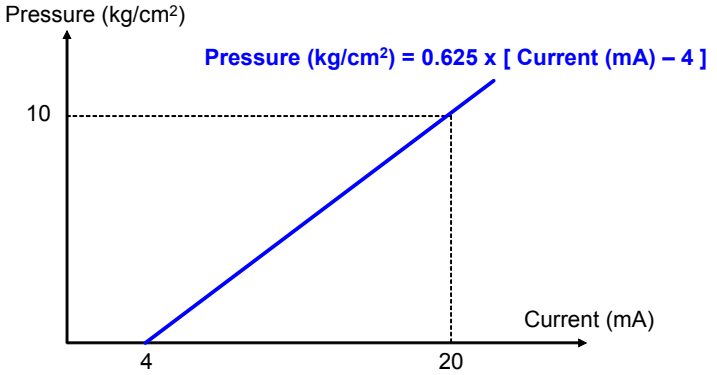


Figure 7.4: Engineer Unit and Current Value

ADAM-6000 analog input module features ***Scaling*** function to convert the voltage (or current) value to the engineer unit value. For example, that's say the condition is if the pressure value is more than or equal to 2.5 kg/cm^2. Without scaling function, you need to convert the pressure value (2.5 kg/cm^2) to the current value (8 mA). Then you enter the current value 8 mA in the **Value** text box of the **Operation** area to define the condition.

Actually, you don't need to transfer the pressure value to current value by yourself. You can enable the scaling function by clicking the **Scaling** check box in the **Operation** area. Then enter the minimum and maximum value for the engineer unit in the **Min** and **Max** text box of the **Scale to** item, to build relationship between the voltage (or current) value and the engineer unit value. For example here, you should enter 0 and 10 as the minimum and maximum pressure value. Since ADAM-6000 module can automatically transfer the pressure value to the current value, you just need to enter the pressure value, 2.5 kg/cm^2, into the **Value** text box to define the condition. The configuration window should looks like Figure 7.5 below. This functionality can help you to configure the condition more intuitively.

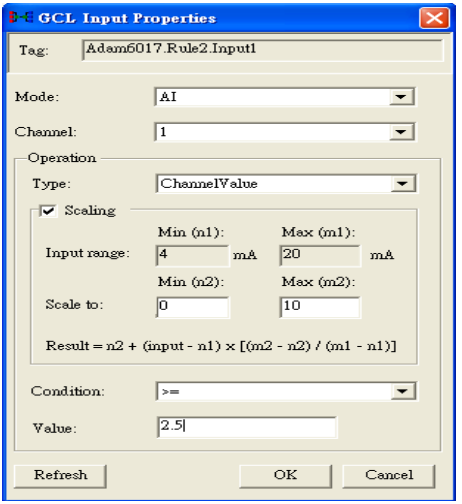


Figure 7.5: Scaling Function of Analog Input Mode

Local Digital Input Channel (DI)

After you choose **DI** as input mode, select the channel by the **Channel** combo box. The value of the selected digital input channel will directly be the input of condition. If the value of the selected digital input channel is logic high, the condition result is logic **True**. If the value is logic low, the condition result is logic **False**.

Local Counter Input Channel (DI_Counter)

After you choose **DI_Counter** as input mode, select the channel by the **Channel** combo box. The count value of the selected counter input channel will directly be the input of condition. Like the Analog Input condition, select the appropriate condition for that input channel by the **Condition** combo box and the **Value** text box. The condition will compare the value read from the counter input channel with the value set by the **Value** text box. If condition is satisfied, the condition result is logic **True**. Otherwise, the condition result is logic **False.**

Local Frequncy Input Channel (DI_Frequency)

After you choose **DI_Frequency** as input mode, select the channel by the **Channel** combo box. The frequency value of the frequency input channel will directly be the input of condition. Like the Counter Input condition, select the appropriate condition for that input channel by the **Condition** combo box and the **Value** text box. The condition will compare the frequency value read from the frequency input channel and the value set by the **Value** text box. If condition is satisfied, the condition result is logic **True**. Otherwise, the condition result is logic **False**.

Internal Timer (Timer)

There are 16 local timers on one ADAM-6000 module. After the timer is started, its value represents how long the time has passed. Here, you can read the timer value and use it as input condition. After you choose **Timer** as input mode, select appropriate timer by the **Index** combo box. (From timer 0 to timer 15) Then you can define the condition by the **Condition** combo box and the **Value** text box (unit: 0.01 second).

Only when the condition is met, the condition result is logic **High**. For example, if you choose >= in the **Condition** combo box and type 500 in the **Value** text box, it means the condition result will remain logic **Low** until the timer value is greater 5 second (meaning 5 second is passed). After 5 seconds is passed, the condition result will become logic **High**.

Internal Flag (AuxFlag)

There are 16 internal flags on one ADAM-6000 module. The data type of internal flag is digital, meaning its value is either logic **True** or logic **False**. You can read the internal flag value and use it as input condition. After you choose **AuxFlag** as input mode, select appropriate internal flag by the **Index** combo box. (From flag 0 to flag 15) Then you can define the condition by the **Condition** combo box.

If you choose **True** in the **Condition** combo box, it means only when the internal flag value equals to logic **True**, the condition result is logic **True**. If you choose **False** in the **Condition** combo box, only when the internal flag value equals to logic **False**, the condition result is logic **True**.

You can use internal flag to implement logic cascade or logic feedback. Refer to Section 7.4 for more detail about how to achieve this.

Note: You can use other program applications to read or write internal flags through ASCII command or Modbus/TCP address. Refer to section 6.4.2 and Appendix B.2.

Local Digital Output Channel (DO)

After you choose **DO** as input mode, select the channel by the **Channel** combo box. The value of the selected digital output channel will directly be the input of condition. If you choose **True** in the **Condition** combo box, it means only when the value of the selected DO channel equals to logic **True**, the condition result is logic **True**. If you choose **False** in the **Condition** combo box, only when the value of the selected DO channel equals to logic **False**, the condition result is logic **True**.

Internal Counter (Counter)

There are 8 internal counters on ADAM-6000 module, you can read its value as input condition. After you choose **Counter** as input mode, select the counter index by the **Channel** combo box. (From counter 0 to counter 7). The count value of the selected internal counter will directly be the input of condition. Like the Frequency Input condition, select the appropriate condition for that counter by the **Condition** combo box and the **Value** text box. The condition will compare the value read from the internal counter and the value set by the **Value** text box. If condition is satisfied, the condition result is logic **True**. Otherwise, the condition result is logic **False**.

7.3.2 Logic Stage

When you click the Logic stage icon, you should see a dialog window similar to Figure 7.6 below.

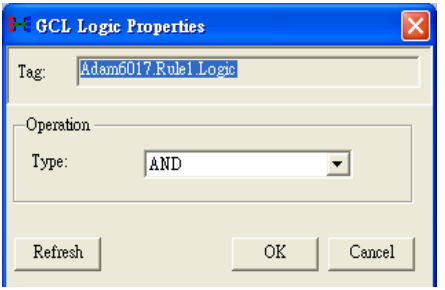


Figure 7.6: Logic Stage Configuration

For each logic rule, there will be at most three input conditions passing logic **True** or **False** values to the Logic stage here. You can choose four logic operations by the **Type** combo box: **AND**, **OR**, **NAND**, **NOR**. The logic operation will process the input logic values, and generate a logic result value to the next Execution stage. After you have selected the appropriate logic operation, click the **OK** button. The Logic stage icon will change its pattern to present the current logic operation.

In order to present how the four logic operations work, we use the truth table to describe. Here we take two input conditions as example. The letter "T" means logic **True**, while the letter "F" means logic **False**.

AND

Input Condition 1	Input Condition 2	Logic value to the Execution Stage
F	F	F
F	T	F
T	F	F
T	T	T

OR

Input Condition 1	Input Condition 2	Logic value to the Execution Stage
F	F	F
F	T	T
T	F	T
T	T	T

NAND (not AND)

Input Condition 1	Input Condition 2	Logic value to the Execution Stage
F	F	T
F	T	T
T	F	T
T	T	F

NOR (not OR)

Input Condition 1	Input Condition 2	Logic value to the Execution Stage
F	F	T
F	T	F
T	F	F
T	T	F

7.3.3 Execution Stage

When you click the Execution stage icon, you should see a dialog window similar to Figure 7.7 below. There are two possible execution setting you can choose by the **Type** combo box in the **Operation** area: Execution Period (**Execution_Period**) and Send to Next Rule (**SendToNextRule**). After you choose the appropriate execution setting, click the **OK** button. The Execution stage icon will change its pattern to present current execution setting condition. We will describe each type in more detail below.

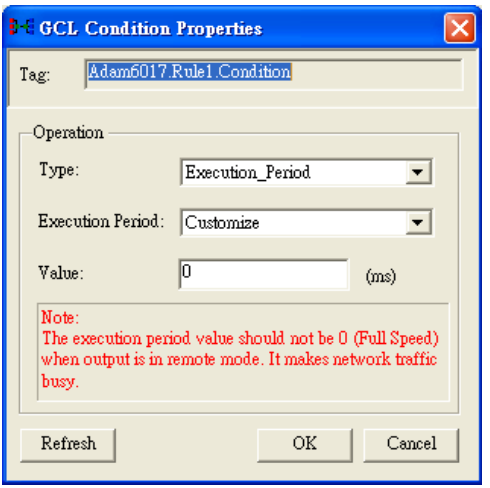


Figure 7.7: Execution Stage Configuration

Execution Period (Execution_Period)

As we mentioned before, the Logic stage will transfer logic result value, logic **True** or logic **False**, to the Execution stage here. The Execution stage will pass this value to the Output stage after a specific period. Below are the steps to configure this period:

1. Select **Execution_Period** in the **Type** combo box.
2. Choose the appropriate period by the **Execution Period** combo box. You can select some pre-defined period from 1 ms to 60000 ms. You can also select **Customize** to define the period by yourself, then enter the period value by the **Value** text box (unit is ms).
3. Click the **OK** button to complete the configuration.

*Note: If you choose **Full speed** in the **Execution Period** combo box, the execution speed will be as fast as possible. There might be network communication traffic problem when the output is on another module, since the execution speed is too fast that too many network packets are transferred on the Ethernet.*

Note: When you want to use ADAM.NET Utility to configure one ADAM-6000 module which is already running its GCL rules, remember to stop the GCL logic rules first.

Send to Next Rule (SendToNextRule)

You can combine different logic rules into one single rule, which can help building more complex logic architecture. There are two methods to combine different logic rules: one way is using ***Send to Next Rule function*** here, another way is using ***Internal Flag***.

When you use ***Send to Next Rule function***, you can set output of one logic rule being input of the next logic rule. Please note it can only combine two logic rules which are next to each other on the same module. If you want to combine different logic rules which are not next to each other, or even on different modules, you need to use internal flag for logic rule cascade. We will introduce this feature in more detail in section 7.4.

After you select **SendToNextRule** in the **Type** combo box, one of the output icons will become the next rule. Refer to Figure 7.8 below.

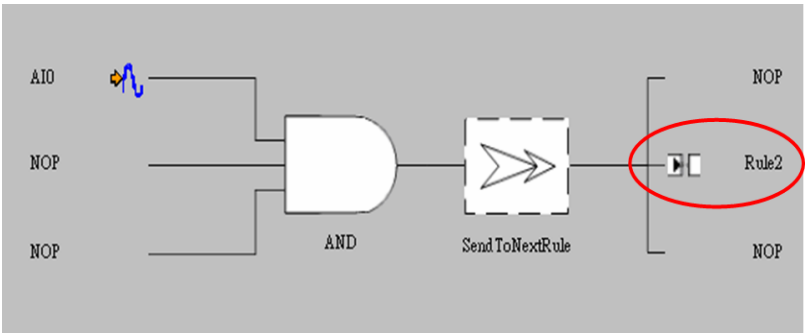


Figure 7.8: Send to Next Rule Function

If you click the next logic rule icon, you will find one of the input condition become previous logic rule. Refer to Figure 7.9. Therefore, the logic result value from the previous logic rule (in this example, logic rule 1) will be one of logic input value of current logic rule (in this example, logic rule 2). This makes the two neighbor logic rules combined together. We call it *Logic Cascade*. Using this method for Logic Cascade, only the two neighbor logic rules can be combined together. If you want to combine two logic rules that are not next to each other, you need to use internal flag. Please refer to Section 7.4.1.

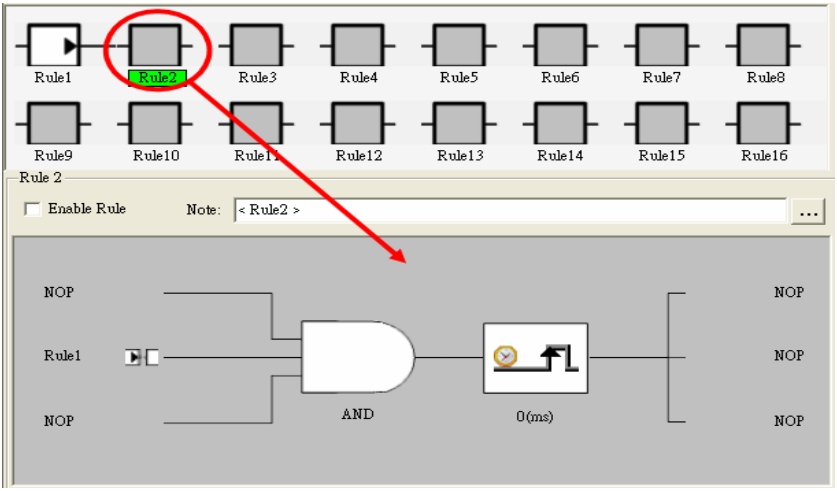


Figure 7.9: The Next Logic Rule

7.3.4 Output Stage

When you click the Output stage icon, you should see a dialog window similar to Figure 7.10 below. There are three outputs for one logic rule. The logic result value from the Execution stage will be passed to the three outputs. And the three outputs will have different action depend on the logic result value.

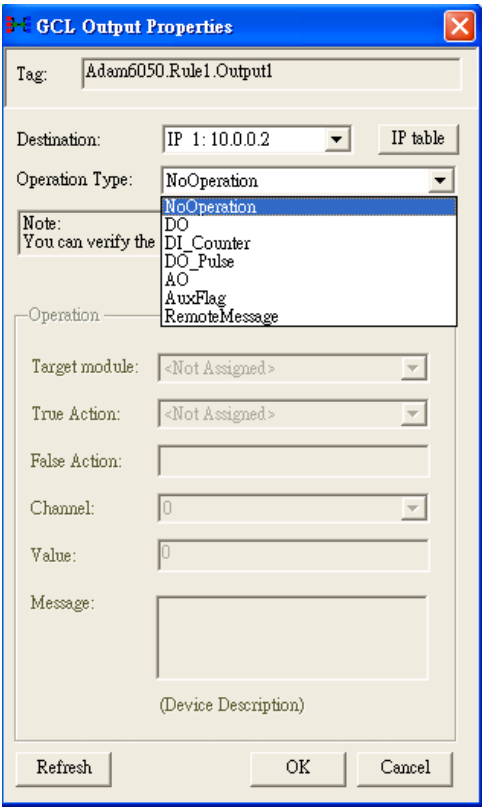


Figure 7.10: Output Stage Configuration

You need to decide the target device for the output by the **Destination** combo box. You can choose **Local**, meaning the output is on the same module, or another remote module by its IP address. Select the appropriate IP address listed in the combo box. The IP addresses are defined in the IP table and you can click the **IP Table** button to configure. (The action of clicking the **IP Table** button here is just the same as click the **IP Table Configuration** button in the **GCL Menu** area.)

Note: When your output destination is not Local, meaning there will be communication between the specific ADAM-6000 module to its target device, remember to use Ethernet switch to connect the ADAM-6000 module with its target device. Do not use an Ethernet hub. This can prevent data packet collision.

After you decide the target device, then you can choose the output action by the **Operation Type** combo box. The default setting is **NoOperation**, meaning there is no output action. You can choose analog output (**AO**), digital output (**DO**), counter channel setting **(DI_counter)**, pulse output (**DO_Pulse**), local timer (**Timer**), local or remote internal flag (**Aux-Flag**), remote message output **(RemoteMessage)** and local internal counter setting **(Counter)** as the output action.

After you have chosen the target device and output action, you can click the **Verify** button to check if the target device exists and supports GCL feature to execute the output action. (If you choose **NoOperation** as output action, it will not check.) After you choose the appropriate output action and complete all related setting, click the **OK** button. That Output stage icon will change its pattern to present the current condition. We will describe each output action in more detail below.

Analog Output (AO)

Below are the steps to configure analog output:

1. Choose correct model name by the **Target module** combo box in the **Operation** area. If the **Destination** is **Local**, you don't need to choose the model name.
2. Choose the appropriate output range by the **TargetRange** combo.
3. Define which channel is responsible to generate output signal on the target device by the **Channel** combo box.
4. Define what value is generated by the **Value** text box. (The unit of the value depends on the range in the **TargetRange** combo box.)
5. Click the **OK** button to complete the configuration.

*Note: You can see the action description by the **Action** text box. When the logic result value passed from Execution stage is logic **True**, the selected analog output channel will generate the new value you defined. When the logic result value passed from Execution stage is logic **False**, the selected analog output channel will not change its output value.*

Digital Output (DO)

Below are the steps to configure digital output:

1. Choose correct model name by the **Target module** combo box in the **Operation** area. If the **Destination** is **Local**, you don't need to choose the model name.
2. Define to generate **True** or **False** digital output signal for the true action (When the logic result value passed from Execution stage is logic **True**) by the **True Action** combo box.
3. The false action (When the logic result value passed from Execution stage is logic **False**) is displayed from the **False Action** text box and will automatically be set according to the true action. The false action will be opposite to the true action. For example, when you choose **False** in the **True Action** combo box, the **False Action** text box will automatically display **True**.
4. Define which channel is responsible to generate output signal on the target device by the **Channel** combo box.
5. Click the **OK** button to complete the configuration.

Counter Channel Setting (DI_Counter)

Below are the steps to configure counter channel setting:

1. Choose correct model name by the **Target module** combo box in the **Operation** area. If the **Destination** is **Local**, you don't need to choose the model name.
2. Define what action (start the counter by **Start**, or stop the counter by **Stop**, or reset the counter by **Reset**) will be taken for the true action (When the logic result value passed from Execution stage is logic **True**) by the **True Action** combo box.
3. The false action (When the logic result value passed from Execution stage is logic **False**) is displayed by the **False Action** text box and will automatically be set according to the true action. Define which counter channel is responsible to take the defined action by the **Channel** combo box.
4. Click the **OK** button to complete the configuration.

Pulse Output (DO_Pulse)

Below are the steps to configure pulse output:

1. Choose correct model name by the **Target module** combo box in the **Operation** area.
2. Define what action (continuous generate pulse train by **Continue**, stop pulse generation by **Stop**, or only generate finite number of pulse by **Num of pulse**) will be taken for the true action (When the logic result value passed from Execution stage is logic **True**) from the **True Action** combo box.
3. The false action (When the logic result value passed from Execution stage is logic **False**) is displayed by the **False Action** text box and will always be **Keep current status**, meaning there is no action change for the selected digital output channel.
4. Define which digital output channel is responsible to take the defined action (start or stop pulse generation) by the **Channel** combo box.
5. If you choose **Num of pulse** in the **True Action** combo box, type the number of pulse you want to generate in the **Value** text box.
6. Click the **OK** button to complete the configuration.

Local Timer (Timer)

There are 16 local timers on ADAM-6000 module. Here, you can define the timer action depending on the logic result value from the Execution stage. After you have chosen **Timer** in the **Operation type** combo box, select the interested timer by the **Index** combo box in the **Operation** area. (From timer 0 to timer 15) Then you can define the timer action by the **Type** combo box in the **Operation** area.

When you choose **ON-Delay**, the timer will start to count the time passed when the logic result value passed from the Execution stage is logic **True**, while the timer will stop counting and reset its value to zero when the logic result value is logic **False**. If you choose **OFF-Delay**, the true action and false action will be opposite: the timer will start to count the time passed when the logic result value is logic **False**, while it will stop counting and rest its value to zero when the logic result value is logic **True**. The action description is displayed by the **True/False act** text box. After you have completed the setting, click the **OK** button.

Local or Remote Internal Flag (AuxFlag)

You can assign the logic result value from the Execution stage, to local or remote internal flag. Select the appropriate internal flag by the **Index** combo box. Define what value you want to assign to the internal flag for the true action (When the logic result value passed from Execution stage is logic **True**) by the **True Action** combo box. The false action (When the logic result value is logic **False**) is displayed by the **False Action** text box and will automatically be opposite to the value in the **True Action**. After you have completed the setting, click the **OK** button.

Remote Message Output (RemoteMessage)

We can send the **Device Description** (you can edit it by the **Message** text box in the **Operation** area) as message to the target device. When you have several logic rules which all will send message, we need to know the message received by the target device is sent by which logic rule. So, you can give the message a index, defined by the **Value** text box in the **Operation** area. Click **OK** to complete the configuration. So when logic result value passed from the Execution stage is **True**, the message will be sent to the target device. If the logic result is **False**, the message won't be sent out.

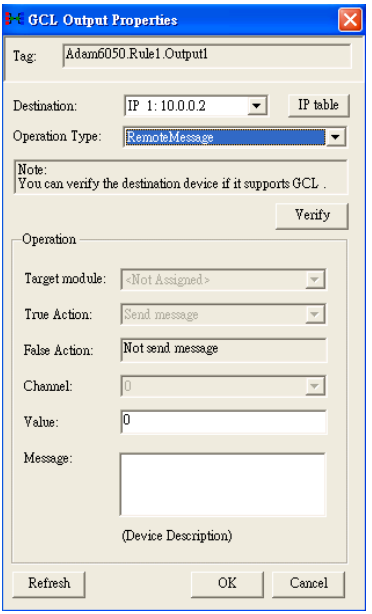


Figure 7.11: Remote Message Output

Note: The total message sent out will include Device Description, the logic rule number which sends this message, the message index, module IP, module name and all I/O status.

Local Internal Counter Setting (Counter)

Below are the steps to configure internal counter setting:

1. Define what action (increase one count to internal counter value by **Positive edge trigger (F->T**), or reset internal counter by **Reset**) will be taken for the true action (When the logic result value passed from Execution stage is logic **True)** by the **True Action** combo box.

*Note: When you choose **Positive edge trigger (F->T)** as action, the counter will only add one count for the first time that the logic result value from Execution Stage is logic high. After the first time, the counter value will not change even the logic result value from Execution Stage is still logic high. So it is the reason why it is called positive edge trigger.*

2. The false action (When the logic result value passed from Execution stage is logic False) is displayed by the **False Action** text box and will automatically be set according to the true action. The false action will be opposite to the true action. Refer to the table below to see the relationship between true action and false action.
3. Define which counter channel is responsible to take the defined action by the **Channel** combo box.
4. Click the **OK** button to complete the configuration.

Below is the table showing the true action and false action for different output action:

Output Action	True action (the logic result value from the Execution stage is logic **True**)	False action (the logic result value from the Execution stage is logic **False**)
AO	Change the analog output value	Keep current status
DO	Output True value	Output False value
	Output False value	Output True value
Counter Channel (DI_Counter)	Start counter counting	Stop counter counting
	Stop counter counting	Start counter counting
	Reset counter	Do nothing
Pulse Output	Generate continuous pulse train	Keep current status
	Generate finite pulses	
	Stop pulse generation	
Timer	Start counting time	Stop counting time and reset timer value to zero
	Stop counting time and reset timer value to zero	Start counting time

Internal Flag	Assign True value to flag	Assign False value to flag
	Assign False value to flag	Assign True value to flag
Remote Message	Send message to target device	Do nothing
Internal Counter Setting (Counter)	Increase 1 count to counter	Do nothing
	Reset counter	

7.4 Internal Flag for Logic Cascade and Feedback

7.4.1 Logic Cascade

Using internal flag as interface, you can combine different logic together to form a new single logic rule which can play more complex logic architecture. You can combine logic rules on the same module, or even on different modules. Please refer to example below to understand how the internal flag works.

Local Logic Cascade

Here, we take one simple example to describe the logic cascade. We use two analog input channels (channel 0 and channel 1) of ADAM-6017 to measure signal from sensors. As long as either of the two input channel read voltage between 3 ~ 5 Volt, digital channel 0 should generate logic high value. Otherwise, the digital channel 0 should generate logic low value. The logic architecture should look like the Figure 7.12 below.

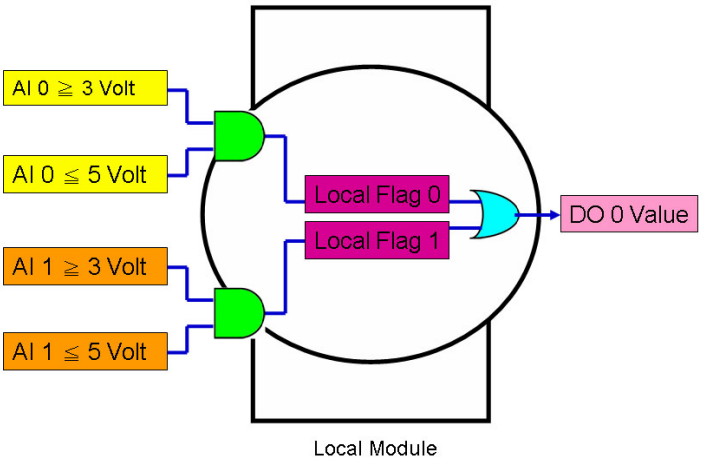


Figure 7.12: Architecture of Local Logic Cascade

In order to implement this logic architecture, we need to use three logic rule and two internal flag to achieve this. Refer to Figure 7.13 ~ 7.15 below for how to configure the three logic rules.

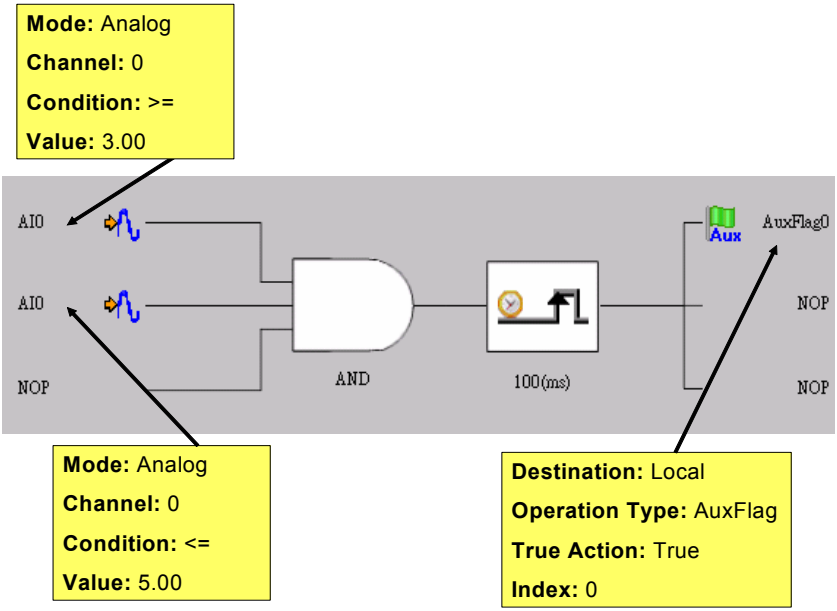


Figure 7.13: Configuration of Logic Rule 1

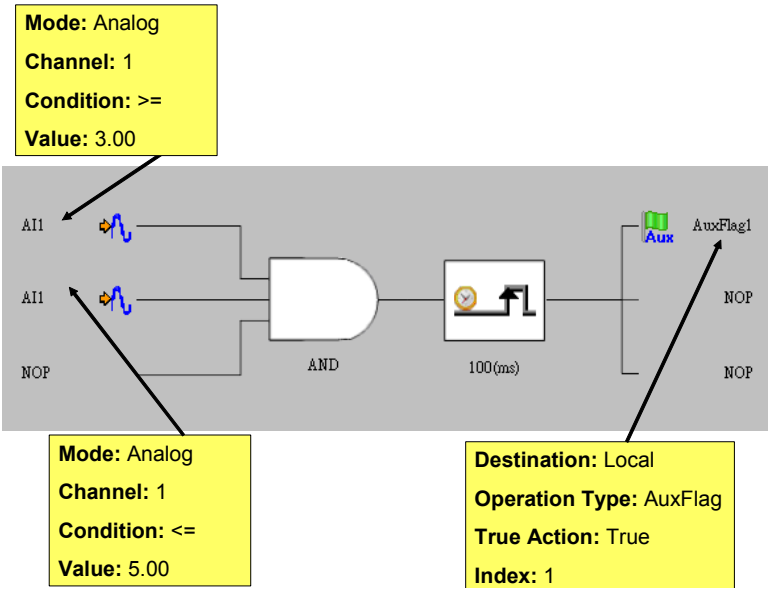


Figure 7.14: Configuration of Logic Rule 2

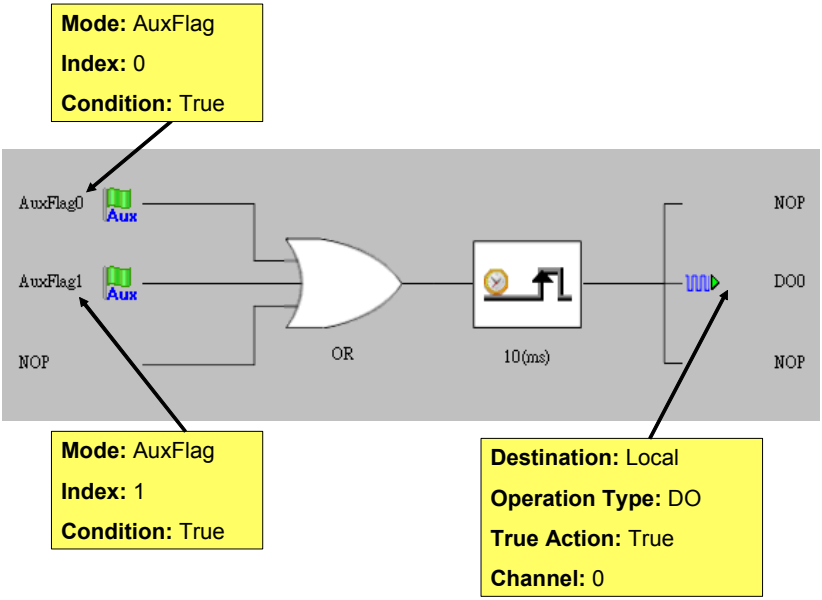


Figure 7.15: Configuration of Logic Rule 3

We use the logic rule 1 to check if AI channel 0 value of the ADAM-6017 is within 3 ~ 5 Volt. Logic rule 2 is used to check if AI channel 1 value is within 3 ~ 5 Volt. The comparison result of logic rule 1 and 2 is assigned to internal flag 0 and 1. The logic rule 3 read the value of these two internal flags and use the OR logic operation to define the output of digital output channel 0. You can find that we have built the logic architecture as shown by Figure 7.12 by the internal flag.

Distributed Logic Cascade

Logic Cascade function is not limited on one single module. Since you can define the internal flag on another module, the logic cascade structure can be across different modules. Take the previous application as example, now you can define the logic rule 1, 2, 3 are running on module A, B and C. Then the logic structure becomes across three ADAM-6000 modules, and we call it Distributed Logic Cascade. Refer to Figure 7.16 for the logic architecture. And the configuration for the three logic rules can be shown by the Figures 7.17 ~ 7.19 below.

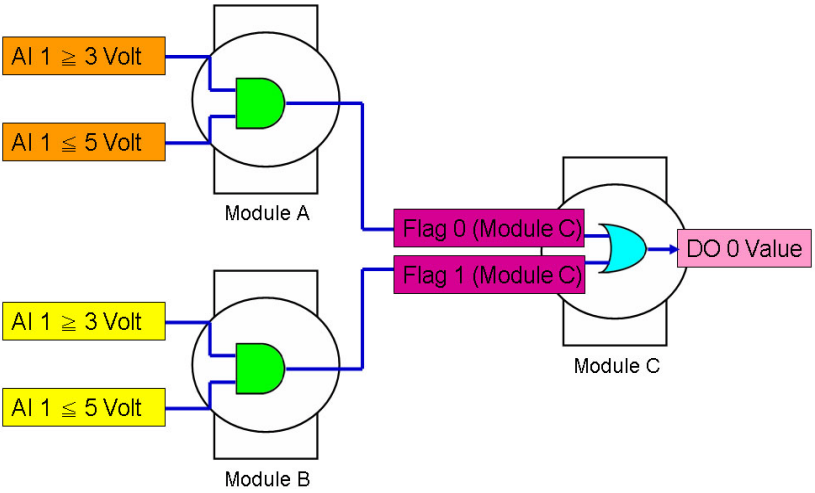


Figure 7.16: Distributed Logic Cascade

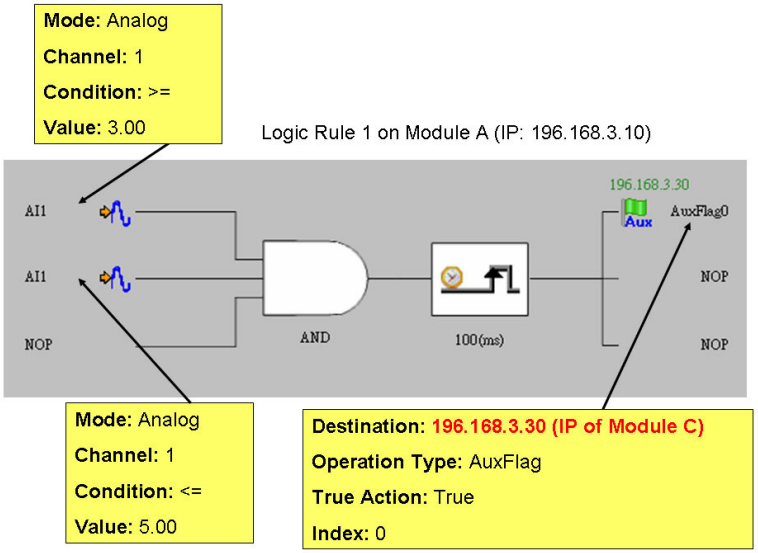


Figure 7.17: Configuration of Logic Rule 1

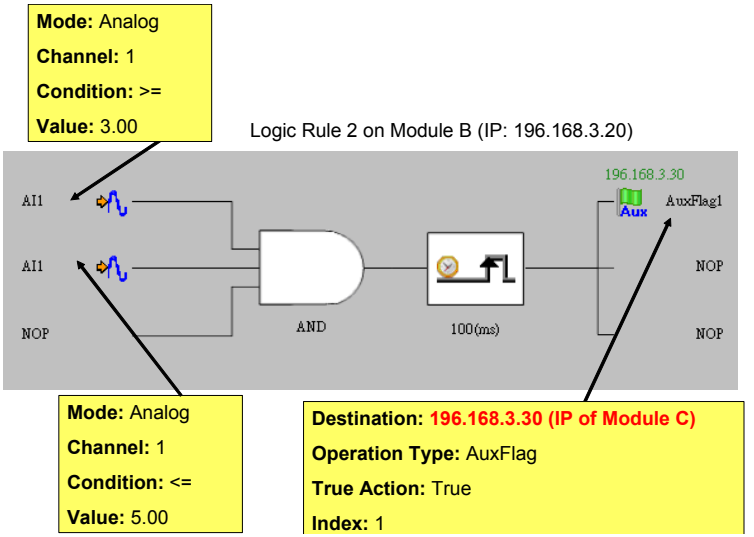


Figure 7.18: Configuration of Logic Rule 2

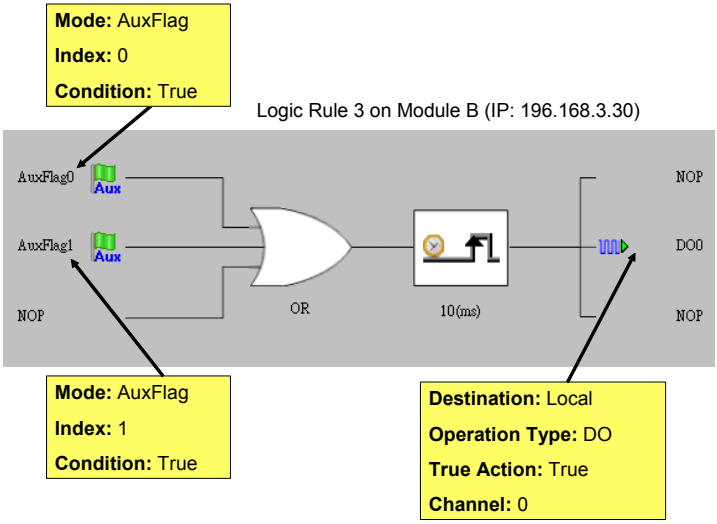


Figure 7.19: Configuration of Logic Rule 3

Using Local or Distributed Logic Cascade architecture, there will be no limitation for input numbers of logic rules. And you can build any logic architecture to meet your application requirement.

7.4.2 Feedback

When you choose the same internal flag for the input condition and output of one single logic rule, the logic rule has logic feedback ability. Refer to Figure 7.20 below. In this example, one input condition and one output are dedicated to the same internal flag 0 (AuxFlag 0). So the output value in current execution will become the input of the next execution. This gives this logic rule feedback ability.

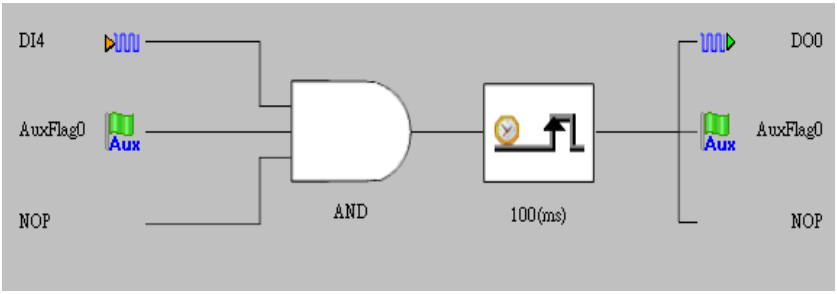


Figure 7.20: Building Logic Feedback

7.5 Download Logic and Online Monitoring

After you have completed all the configurations for GCL logic rule, click the **Download Project** button in the **GCL Menu** area. All the configuration will be downloaded to the target device. Then you can click the **Run GCL** button in the **GCL Menu** area to execute the project on the target module. You can see the **Current Status** icon become the **Running** mode.

ADAM-6000 Module features special Online Monitoring function. In the Running mode, you can click the **Monitoring** button in the **GCL Menu** area to enable this function. Then you can see the execution situation on the **Individual Logic Rule Configuration** area. The yellow dot appears when the execution flow proceed to that stage. Besides, you can see the current input value besides the **Input Condition** icon. Refer to Figure 7.21 for the Online Monitoring function. In this example, you can see the input condition DI 1 and DI 3 has been satisfied, so the yellow dot appears next to the two Input Condition icons. And you can see the current input value showing at the top of the three Input Stage icons.

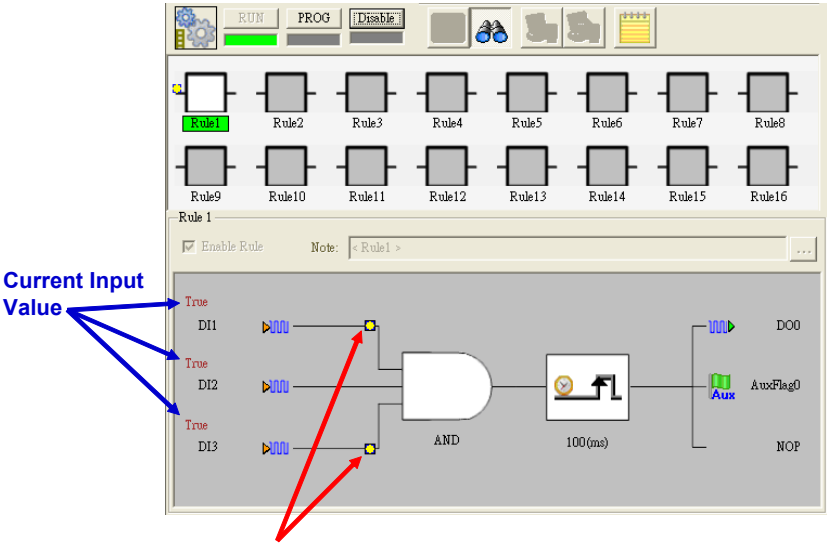


Figure 7.21: Online Monitoring Function

*Note: When you use Internal Flags **(AuxFlag)** as the inputs of GCL logic rules, you can dynamically change the flag values in the online monitoring window of ADAM.NET Utility. Simply double click the input icons represented the internal flag, and you can see the flag values change from **True** to **False**, or from **False** to **True**.*

GCL Rule Execution Sequence

There are 16 logic rules on one ADAM-6000 module. Refer to the figure 7.22 below to see the execution flow for one cycle. You can see there are 3 groups for one cycle: **Input Condition** + **Logic**, **Execution**, and **Output**. All the **Input Condition** + **Logic** stages of rules which are enabled will execute sequentially first. Then, all the **Execution** stages of rules which are enabled will be executed in sequence. At the end, all **Output** stages of rules which are enabled will be executed in order.

For some advanced applications, you may combine different rules together by Logic Cascade architecture (described in section 7.4.1). For example, the output of rule 1 is connected to the input of rule 2, by assigning to the same internal flag. Based on the execution flow mentioned above, the **Input Condition** + **Logic**, **Execution** and **Output** stages of rule 1 will be executed sequentially. Therefore, the output of rule 1 will be updated at the last stage (**Output** stage) in the first cycle, and the input of rule 2 can detect the change of output of rule 1 in the next cycle.

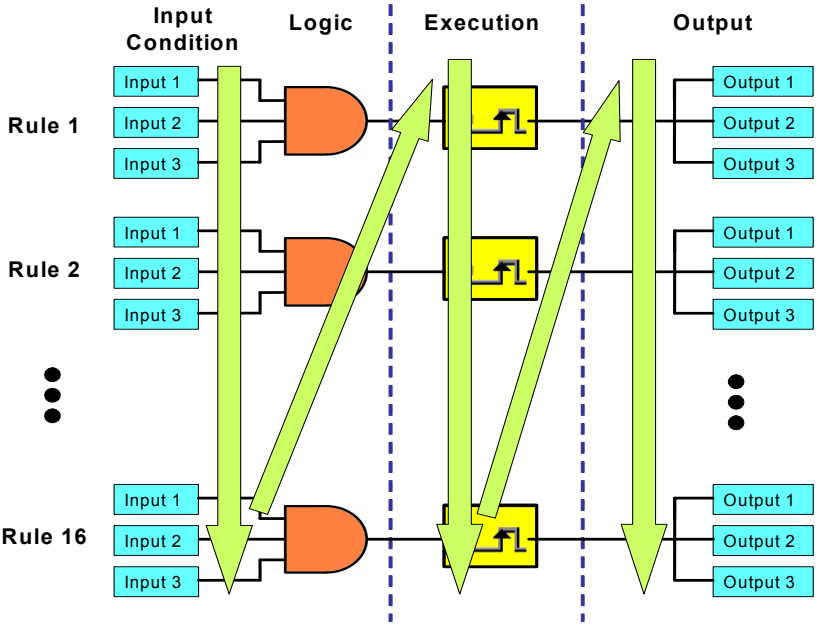


Figure 7.22: GCL Execution Sequence

GCL Execution and Data Transfer Performance

1. Local Output

Condition: Running 1 logic rule on one ADAM-6050 module

Processing time: < 1 milliseconds

(Processing time includes hardware input delay time, 1 logic rule execution time and hardware output delay time)

If multiple logic rules are used, the processing time can be estimated by equation below:

- logic rule number n ≤ 16

Approximate Processing Time (for one cycle) = 600 + n*370 (μs)

2. Remote Output

Condition: Running 1 logic rule on one ADAM-6050, output is on another ADAM-6050 module through one Ethernet Switch

Processing + Communication time: < 3 milliseconds

7.6 Typical Applications with GCL

In order to shorten GCL configuration time, Advantech has provided several example project files for some typical applications. You can find these example project files on the CD with the ADAM module. Simply load these example project files by clicking the **Project Content** button of **GCL Menu** bar (Refer to the Table 7.1 of Section 7.2). You can modify an example project based on your application requirements. Then you can download the modified project to your module and execute it. We will introduce each example project file in more detail below:

1. Empty Project

When you want to clear all configurations for GCL, it is simple to load this example project. Then you don't need to clear all the configurations manually.

2. On/Off Control (Two buttons to control On and Off Seperately)

In some automation applications, two digital inputs (DI 0 and DI 1) are used to control one digital output status (DO 0). The DO status will become logic high when DI 0 is logic high, and the DO status will return to logic low when DI 1 is logic high. For example, motor operation is controlled by two buttons. When the first button is pressed, the motor is started. If the second button is pressed, the motor will be stopped immediately. PLCs are typically used for this kind of industrial automation application, and the ladder diagram will look like Figure 7.23 below:

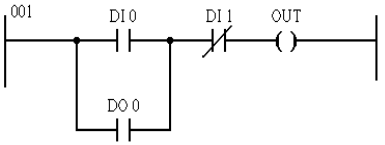


Figure 7.23: Ladder Diagram for On/Off Control

Now, we can use GCL logic to achieve the same control operation. Two logic rules are used. The complete logic architecture is shown by Figure 7.24 below:

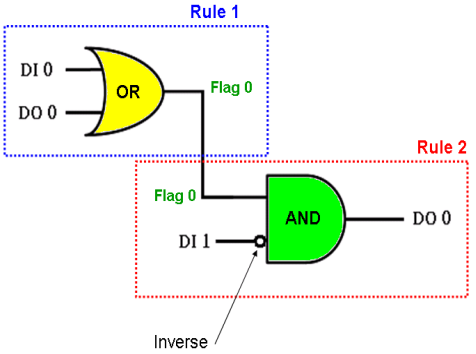


Figure 7.24: GCL Logic for On/Off Control

After you load the example project file, you can find that it uses rule 1 and rule 2. One output of rule 1 and one input of rule 2 are assigned to the same internal flag: Flag 0. This can combine 2 or more logic rules together, that we call it ***Logic Cascade*** (Please refer to Section 7.4.1). We have configured the condition for DI 1 input as **False**, so the DI input value is inverted before entering the AND operator of rule 2. The GCL logic architecture is similar to the PLC ladder diagram.

3. Sequential Control (Turn On in Sequence and Remain On)

In this kind of application, several digital outputs will be activated in sequence and latch their values. In the example project we provide, DO 0 ~ DO 5 will sequentially be controlled to change their status. The time chart for this application can be shown by Figure 7.25 below:

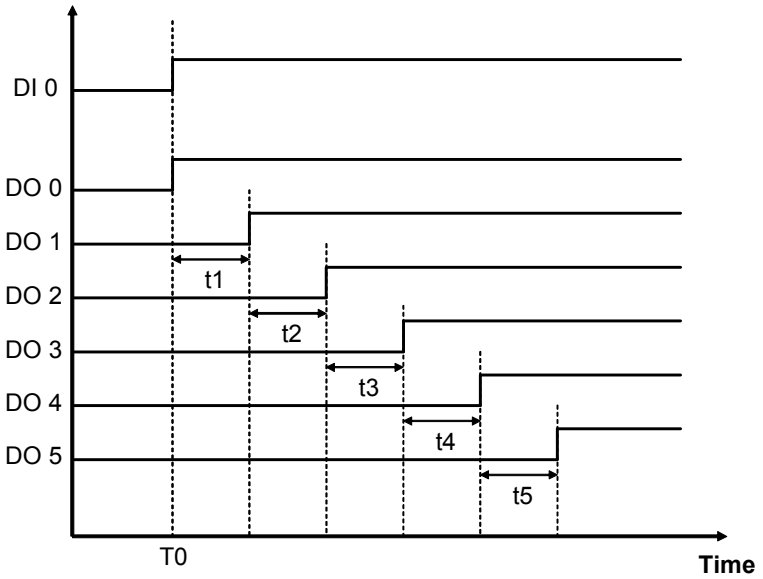


Figure 7.25: Time Chart for Sequence Control

(Turns On in Sequence and Remains On)

In the example project, DI 0 is used as a trigger to start the sequential control action. Therefore, when DI 0 becomes logic high (at the moment T0), DO 0 will become logic high immediately. Then, DO1 ~ DO5 will sequentially be activated to logic high after a specific time interval. You can decide the time interval t1 ~ t5 (They can be different values). In this example project, t1 ~ t5 are all 5 seconds.

We can use 6 logic rules and 1 internal timer for this GCL application. In the first logic rule, DI 0 is used to trigger Timer 0 and DO 0. Since the timer has been triggered, it will start counting time and DO 1 ~ DO 5 will be activated after a specific amount of time has elapsed.. The GCL architecture can be shown by Figure 7.26 below:

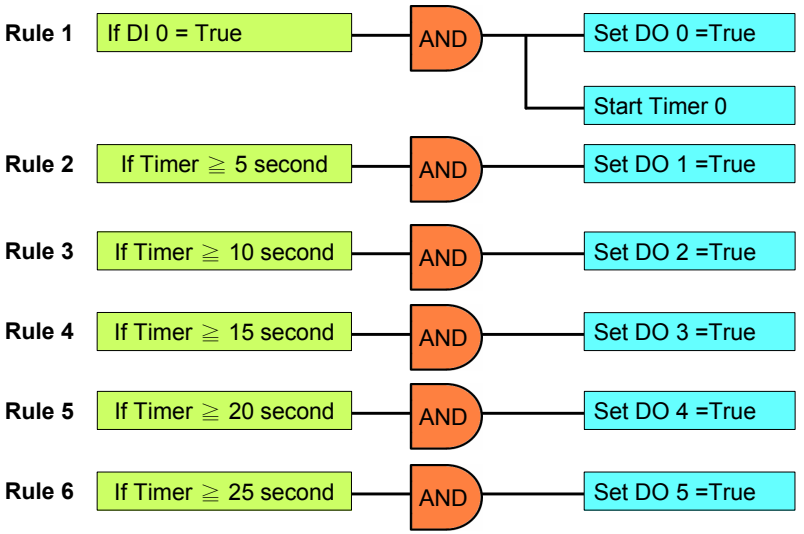


Figure 7.26: GCL Logic for Sequence Control

(Turns On in Sequence and Remains On)

4. **Multiple DI to control one DO (12 DI to 1 DO)**

In many applications, only when multiple digital inputs are logic high (related conditions are all satisfied), digital output status will become logic high. In the example project we provide, only when DI 0 ~ DI 11 are all logic high at the same time, the DO 0 will become logic high. The time chart of this application can be shown by Figure 7.27 below. The green band area indicates the moment that all 12 DI are logic high, thus DO 0 will be logic high. For other time interval, there is at least one DI channel whose value is not logic high, so DO 0 value is logic low.

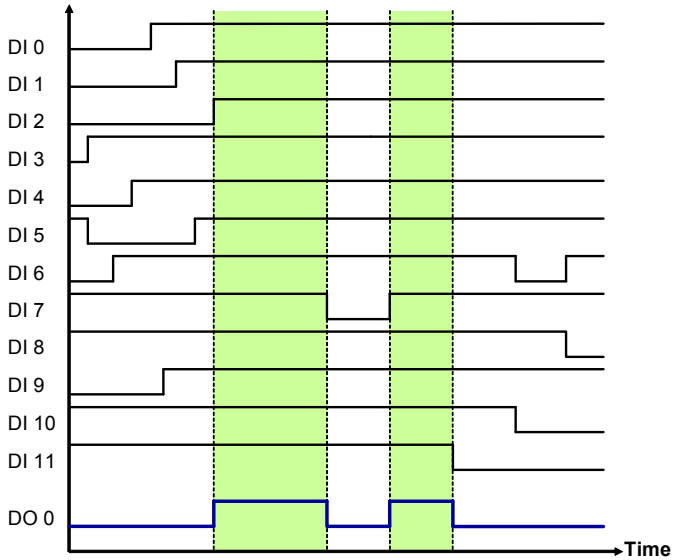


Figure 7.27: Time Chart for 12 DI to 1 DO

You can simply implement one AND logic operator to achieve this control system. However, since one logic rule only has three inputs, we need to use ***Logic Cascade*** function to have 12 inputs. There are two ways to achieve ***Logic Cascade***:

- Select **SendtoNextRule** in Execution Stage of one logic rule. It will combine this logic rule to the next logic rule. (Refer to Section 7.3.3)
- Assign output of one logic rule and input of another logic rule to the same internal flag, combining the two logic rules together. (Refer to Section 7.4.1)

With the first method, the two logic rules must be next to each other. For example, rule 1 and rule 2 can be combined together. But you cannot combine rule 1 and rule 3. There is no such limitation if you use the second method to combine two different logic rules. Using the second method, you can even combine two rules on different modules together. Here, this example project adapts the first method for ***Logic Cascade***. So the GCL logic architecture can be shown by Figure 7.28 below:

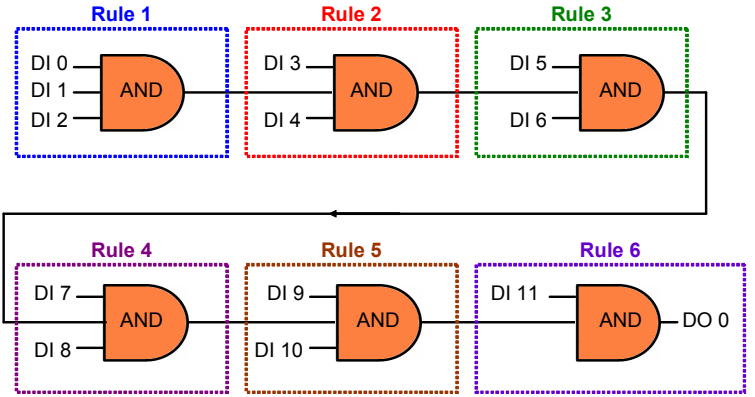


Figure 7.28: GCL Logic for 12 DI to 1 DO

5. Flicker

Flicker is commonly used in automation control application. Typical example is we want to make the alarm flashing, controlled by digital output. This application requires a continuous pulse train by a digital output channel and we can decide the period of the pulse train. The time chart for flicker application can be shown by Figure 7.29 below:

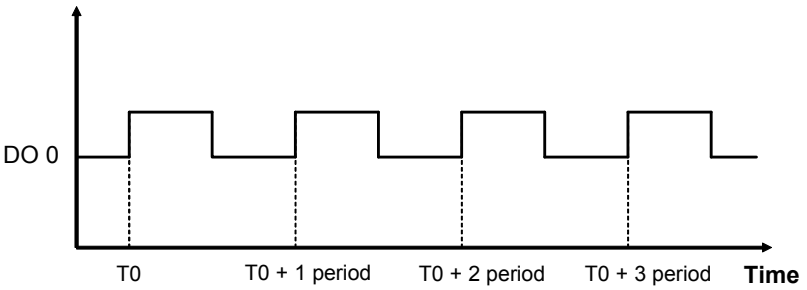


Figure 7.29: Time Chart for Flicker Application

We need to use 1 Internal Flag (Flag 0) and 2 logic rules for the Flicker application described above. In logic rule 1, the value of Flag 0 is inverted (By choosing **NAND** in the Logic stage) periodically. (Here, it is 0.5 second) The period is defined by the **Execution_Period** in the **Execution** Stage. (Refer to the Section 7.3.3) The status of DO 0 is controlled by Flag 0 in logic rule 2, so DO 0 will change every 0.5 second. The GCL logic rule architecture is shown by Figure 7.30 below:

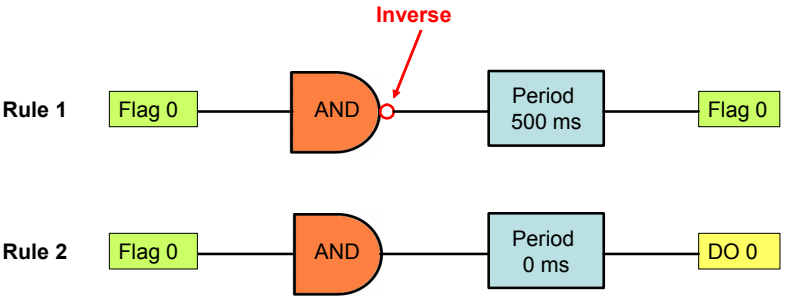


Figure 7.30: GCL Logic for Flicker

6. Rising Edge

For Rising Edge application, the DO status will be activated to logic high, when DI value is changed from logic low to logic high (it is so-called rising edge). But the DO value won't continuously remain logic high. Instead, after a specific time interval (in the example, it is 1 second), the DO value will return to logic low. Refer below for its time chart:

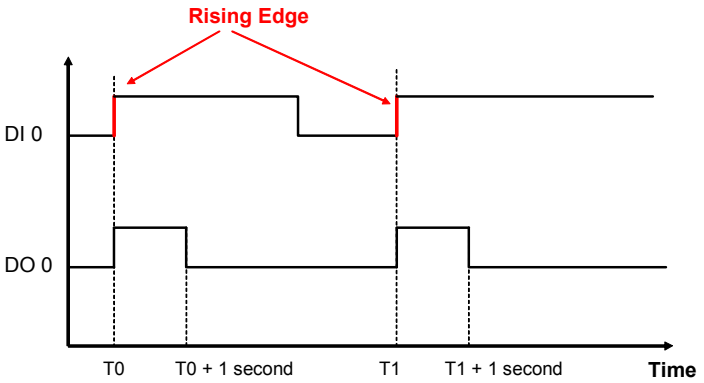


Figure 7.31: Time Chart for Rising Edge

You can see that DO 0 will only be triggered when rising edge of DI 0 occurs. In the example project we provide, the DO status will remain logic high for 1 second. Then it will back to logic low. When PLC is used for this kind of application, the ladder diagram will look similar to Figure 7.32 below:

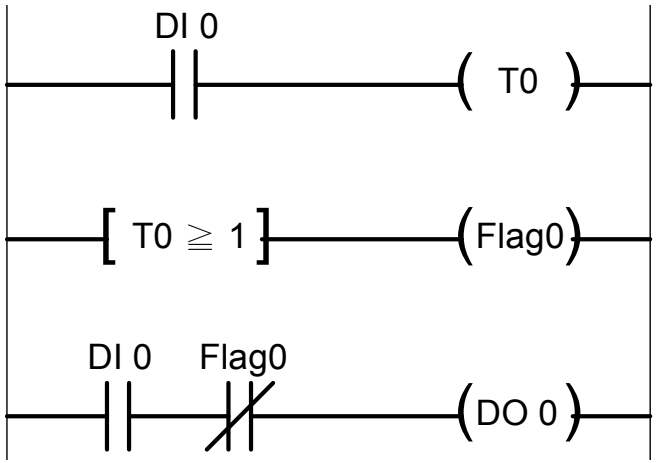


Figure 7.32: Ladder Diagram for Rising Edge

When you use GCL to achieve rising edge application, 3 logic rules, 1 Internal Timer (Timer 0) and 1 Internal Flag (Flag 0) are needed. Refer to Figure 7.33 below for GCL logic architecture. With logic rule 3, DO 0 value is controlled by DI 0 and Flag 0. Flag 0 is initially set as **False**.

When rising edge occurs (DI value changes from logic low to logic high), DO will be activated (logic rule 3 are satisfied), and Timer 0 starts to count time (logic rule 1 are satisfied). After Timer 0 counts up to the specific time interval (1 second), Flag 0 will become logic **True** by logic rule 2, making DO 0 value logic low (logic rule 3 are not satisfied). The GCL architecture is similar to the ladder diagram.

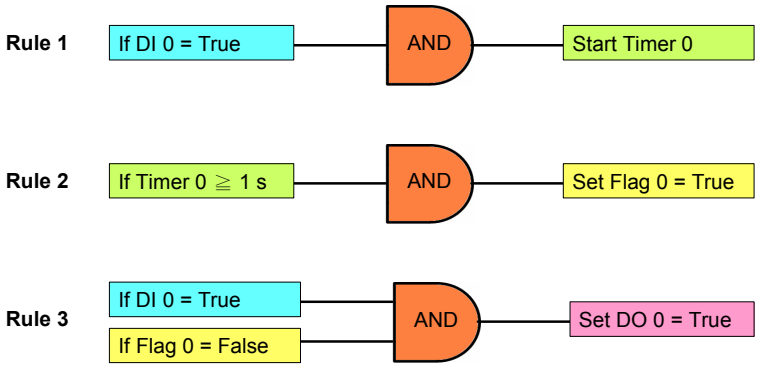


Figure 7.33: GCL Logic for Rising Edge

7. Falling Edge

For Falling Edge application, the DO value will be activated to logic high, when DI value is changing from logic high to logic low (it is so-called falling edge). But the DO value won't continuously remain logic high. Instead, after a specific period (in the example project, it is 1 second), the DO value will back to logic low. Refer to Figure 7.34 below for its time chart:

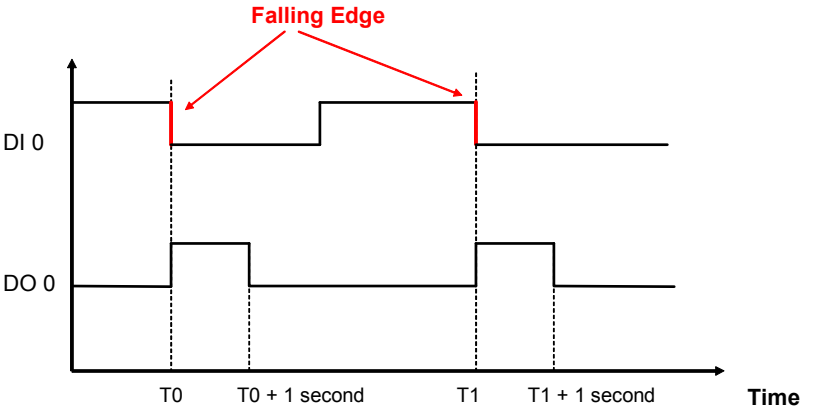


Figure 7.34: Time Chart for Falling Edge

You can see the DO 0 will only be triggered when falling edge of DI 0 occurs. In the example project we provide, the DO status will remain logic high for 1 second. Then it will back to logic low. When PLC is used for this kind of application, the ladder diagram will look similar to Figure 7.35 below:

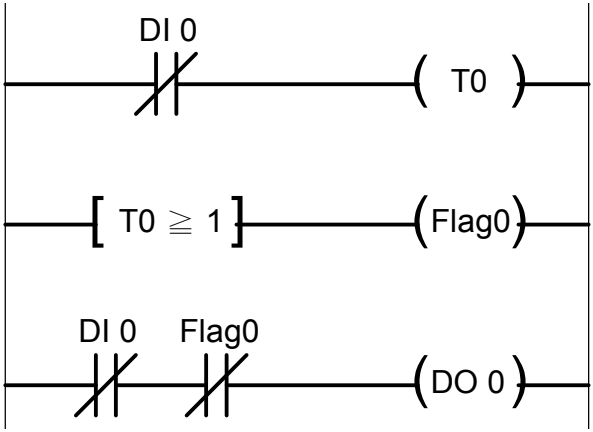


Figure 7.35: Ladder Diagram for Falling Edge

When you use GCL to achieve falling edge application, 3 logic rules, 1 Internal Timer (Timer 0) and 1 Internal Flag (Flag 0) are needed. Please refer to Figure 7.36 below for GCL logic architecture. With logic rule 3, DO 0 value is controlled by DI 0 and Flag 0. Flag 0 value is logic **False** at beginning.

When falling edge occurs (DI value changes from logic high to logic low), DO will be activated (logic rule 3 are satisfied), and Timer 0 starts to count time (logic rule 1 are satisfied). After Timer 0 counts up to the specific time interval (1 second), Flag 0 will become logic **True** by logic rule 2, making DO 0 value logic low (logic rule 3 are not satisfied). The GCL architecture is similar to the ladder diagram.

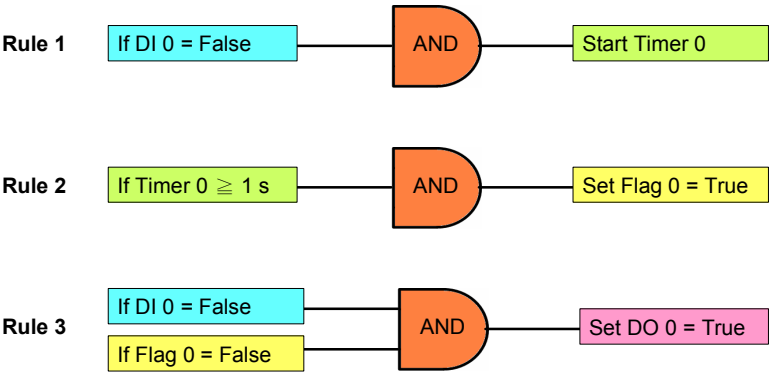


Figure 7.36: GCL Logic for Falling Edge

8. Sequential Control (Turn On and Off in Sequence Continuously)
This type of automation application is similar to the 3rd application we have introduced. They are both sequential control applications. For example 3, DO channel will keep its value after it is turned on. In example 8 here, after DO channel is turned on, it will be turned off after a specific time. You can see the time chart for this application by Figure 7.37 below. There is one time base needed to control the digital output sequential action. In this example, the period of the time base to turn off one DO and turn on the next-door DO is 1 second.

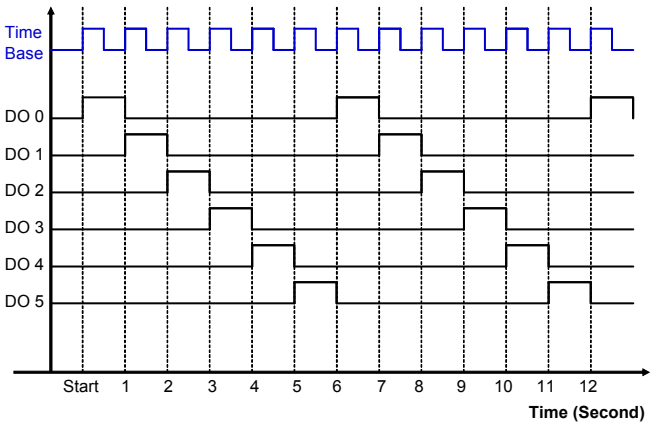


Figure 7.37: Time Chart for Sequence Control
(Turn On and Off in Sequence Continuously)

In order to implement this kind of application, 9 logic rules, 1 Internal Counter (Counter 0) and 1 Internal Flag (Flag 0) are used. In the example project we provide, logic rule 1 and 8 are used to create the time base. By logic rule 8, Flag 0 value will change every 0.5 second. In logic rule 1, once the Flag 0 value is logic high, the Counter 0 will increase 1 unit. So every 1 second, Counter 0 will increase 1 unit, making Counter 0 the time base.

Logic rules 9 ~ 14 are used to control DO 0 ~ 5. Which logic rule should be executed is based on Counter 0 value. Since Counter 0 value will continuously add 1 unit every 1 second, logic rules 9 ~ 14 will be executed in sequence every 1 second. Therefore, DO 0 ~ DO 5 will be activated sequentially in 1 second. When logic rule 15 is executed, Counter 0 will reset and its value will back to zero. So it makes the logic rules execution become a continuous loop. Refer to Figure 7.38 below for its GCL architecture.

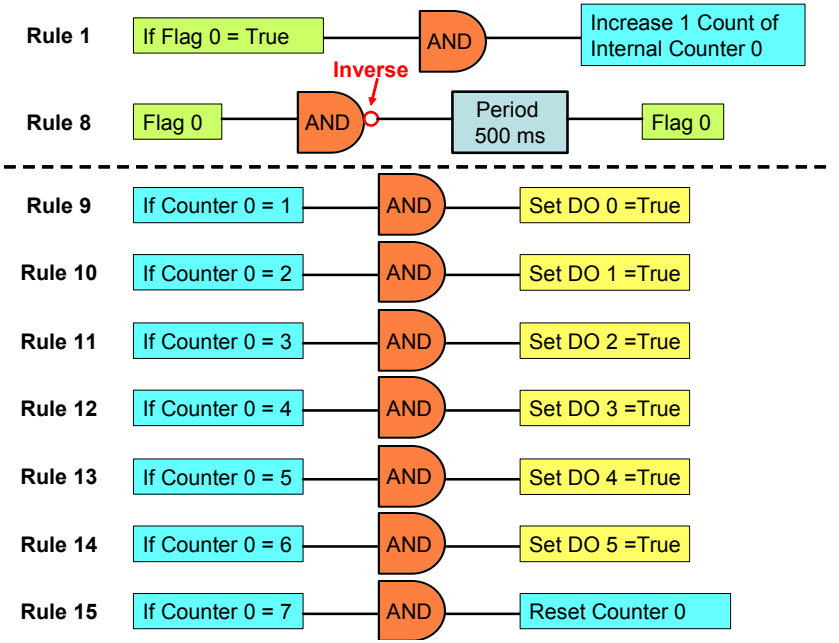


Figure 7.38: GCL Logic for Sequence Control
(Turn On and Off in Sequence Continuously)

9. DI Event Trigger (Only Occurs Once)

We can simply use GCL to perform Event trigger. For this kind of application, a DI channel is used to trigger some action. So, the input condition of GCL logic rule will be if the DI value is logic True, and output of the rule can be some desired action, such as sending remote message. When the DI value becomes logic True, the input condition is satisfied. The GCL logic rule will send message continuously until the DI value backs to logic False. However, it is not what we plan. (We don't want the message is sent out continuously. Instead, we want the message will only be sent once at the first moment that the condition is satisfied.)

This kind of application can be simply achieved by using one counter input channel. You can refer to figure 7.39 below. Select local counter input channel (DI_Counter) in the Input condition for one logic rule. There are two output used for the same logic rule, one is reset the counter input channel, and another is the desired action you want. Then, when the counter input channel detect DI signal, the condition is satisfied and the desired action will be done. At the same time, the GCL rule reset the counter input channel. So the desired action will only be executed once.

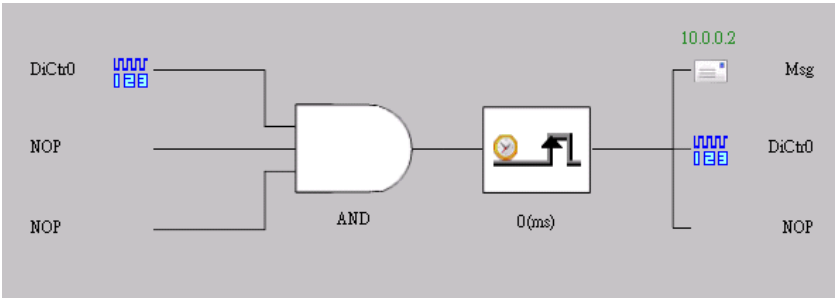


Figure 7.39: GCL Logic for Event Trigger (Only Occurs Once)

The true image of configuration of GCL logic rule in ADAM.NET Utility can be shown by Figure 7.40 below. (Here, the desired action is to send remote message.)

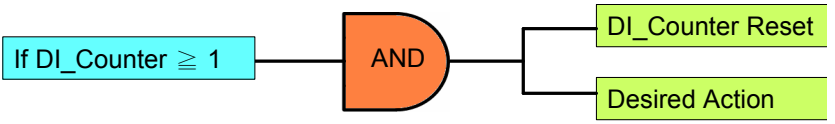


Figure 7.40: Event Trigger Configuration (Only Occurs Once)

APPENDIX A

Design Worksheets

Appendix A Design Worksheets

An organized system configuration will lead to efficient performance and reduce engineer effort. This Appendix provides the necessary worksheet, helping users to configure their DA&C system in order.

Follow these working steps to build up your system relational document:

1. Asking questions and getting answers for your control strategy.

- What will be monitored and controlled?
  (List the equipment)
- What will be monitored and controlled separately?
  (Divide the function area)
- What will be monitored and controlled by ADAM-6000 I/O?
  (List the target equipment in different function areas)

2. Identify the I/O types & fullfill Table A-1 to establish the I/O database.

Table A.1: I/O Data Base

Function Area	Equipment Input or Output	I/O Module Type	I/O Module Product No.	Voltage of Range	Current of Range	Special Requirements

3. Mapping the I/O data base into ADAM-6000 I/O modules.

1) In column A, note the TCP IP addresses for individual function areas.

2) In column B, list the I/O module's product number.

3) In column C, enter the max # of I/O points available per module.

4) In column D, total the number of the I/O point you need.

5) In column E, calculate the total # of these modules that you will need.

6) In column F, enter the # of spare modules needed for future expansion.

7) In column G, enter the total # (Required + Spare) of these modules that you need for these systems.

<A> <B> <C> <D> <E> <F> <G>

Table A.2: Summary Required Modules

ADAM-6000 IP Address	I/O Module Product No. I/O Points per Module	Total I/O Points Required	I/O Module Required	Spare I/O Modules	Total I/O Modules

4. Implement the Modbus address in to the I/O table.

ADAM-6000 IP Address

Table A.3: Table for Programming		
I/O Type	Channel Number I/O Address Tag Name	Equipment & Description

These several worksheets are very useful to hardware wiring and software integration, please make copies to establish your own system configuration documentation.

APPENDIX B

Data Formats & I/O Range

Appendix B Data Formats and I/O Range

B.1 ADAM-6000 Commands Data Formats

ADAM-6000 and ADAM-5000/TCP system accept a command/response form with the host computer. When systems are not transmitting they are in listen mode. The host issues a command to a system with a specified address and waits a certain amount of time for the system to respond. If no response arrives, a time-out aborts the sequence and returns control to the host. This chapter explains the structure of the commands with Modbus/TCP protocol.

B.1.1 Command Structure

It is important to understand the encapsulation of a Modbus request or response carried on the Modbus/TCP network. A complete command is consisted of command head and command body. The command head is prefixed by six bytes and responded to pack Modbus format; the command body defines target device and requested action. Following example will help you to realize this structure quickly.

Example:

If you want to read the first two values of ADAM-6017 (address: 40001~40002), the request command should be:

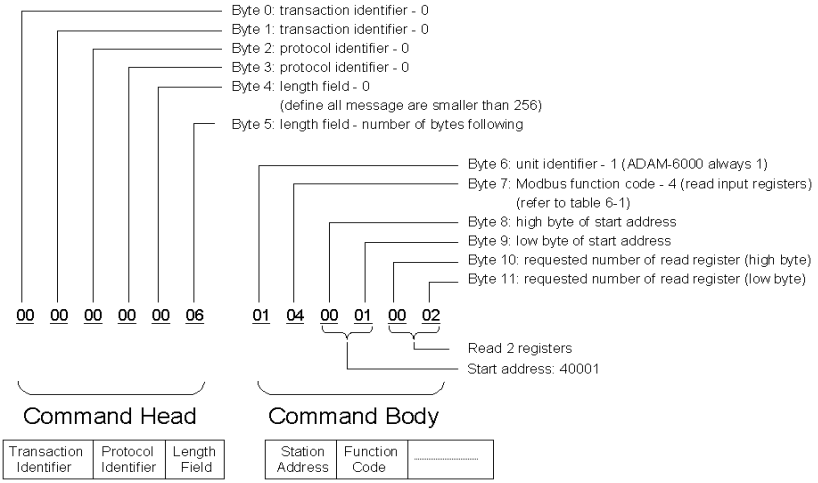


Figure B.1: Request Comment Structure

And the response should be:

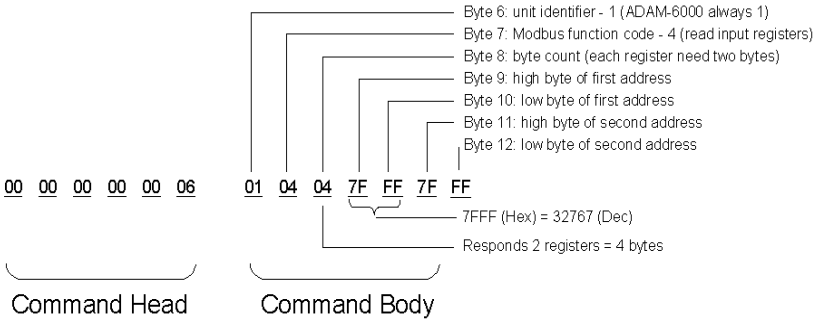


Figure B.2: Response Comment Structure

B.1.2 Modbus Function Code Introductions

To full-fill the programming requirement, there is a series of function code standard for user's reference…

Code (Hex)	Name	Usage
01	Read Coil Status	Read Discrete Output Bit
02	Read Input Status	Read Discrete Input Bit
03	Read Holding Registers	Read 16-bit register. Used to read integer or floating point process data.
04	Read Input Registers	
05	Force Single Coil	Write data to force coil ON/OFF
06	Preset Single Register	Write data in 16-bit integer format
08	Loopback Diagnosis	Diagnostic testing of the communication port
0F	Force Multiple Coils	Write multiple data to force coil ON/OFF
10	Preset Multiple Registers	Write multiple data in 16-bit integer format

Function Code 01

The function code 01 is used to read the discrete output's ON/OFF status of ADAM-6000 modules in a binary data format.

Request message format for function code 01:

Command Body					
Station Address	Function Code	Start Address High Byte	Start Address Low Byte	Requested Number of Coil High Byte	Requested Number of Coil Low Byte

Example: Read coil number 1 to 8 (address number 00017 to 00024) from ADAM-6000 Modules

01 01 00 17 00 08

Response message format for function code 01:

Command Body					
Station Address	Function Code	Byte Count	Data	Data	…

Example: Coils number 2 and 7 are on, all others are off.

01 01 01 42

In the response the status of coils 1 to 8 is shown as the byte value 42 hex, equal to 0100 0010 binary.

Function Code 02

The function code 02 is used to read the discrete input's ON/OFF status of ADAM-6000 in a binary data format.

Request message format for function code 02:

Command Body					
Station Address	Function Code	Start Address High Byte	Start Address Low Byte	Requested Number of Input High Byte	Requested Number of Input Low Byte

Example: Read coil number 1 to 8 (address number 00001 to 00008) from ADAM-6000 modules

01 02 00 01 00 08

Response message format for function code 02:

Command Body					
Station Address	Function Code	Byte Count	Data	Data	…

Example: input number 2 and 3 are on, all others are off.

01 02 01 60

In the response the status of input 1 to 8 is shown as the byte value 60 hex, equal to 0110 0000 binary.

Function Code 03/04

The function code 03 or 04 is used to read the binary contents of input registers

Request message format for function code 03 or 04:

Command Body					
Station Address	Function Code	Start Address High Byte	Start Address Low Byte	Requested Number of Register High Byte	Requested Number of Register Low Byte

Example: Read Analog inputs #1 and #2 in addresses 40001 to 40002 as floating point value from ADAM-6017 module

01 04 00 01 00 02

Response message format for function code 03 or 04:

Command Body					
Station Address	Function Code	Byte Count	Data	Data	…

Example: Analog input #1 and #2 as floating point values where AI#1=100.0 and AI#2=55.32

01 04 08 42 C8 00 00 47 AE 42 5D

Function Code 05

Force a single coil to either ON or OFF. The requested ON/OFF state is specified by a constant in the query data field. A value of FF 00 hex requests it to be ON. A value of 00 00 hex requests it to be OFF. And a value of FF FF hex requests it to release the force.

Request message format for function code 05:

Command Body					
Station Address	Function Code	Coil Address High Byte	Coil Address Low Byte	Force Data High Byte	Force Data Low Byte

Example: Force coil 3 (address 00003) ON in ADAM-6000 module

01 05 00 03 FF 00

Response message format for function code 05:

The normal response is an echo of the query, returned after the coil state has been forced.

Command Body					
Station Address	Function Code	Coil Address High Byte	Coil Address Low Byte	Force Data High Byte	Force Data Low Byte

Function Code 06

Presets integer value into a single register.

Request message format for function code 06:

Command Body					
Station Address	Function Code	Register Address High Byte	Register Address Low Byte	Preset Data High Byte	Preset Data Low Byte

Example: Preset register 40002 to 00 04 hex in ADAM-6000 module

01 06 00 02 00 04

Response message format for function code 06:

The normal response is an echo of the query, returned after the coil state has been preset.

Command Body					
Station Address	Function Code	Register Address High Byte	Register Address Low Byte	Preset Data High Byte	Preset Data Low Byte

Function Code 08

Echoes received query message. Message can be any length up to half the length of the data buffer minus 8 bytes.

Request message format for function code 08:

Command Body		
Station Address	Function Code	Any data, length limited to approximately half the length of the data buffer

Example: 01 08 00 02 00 04

Response message format for function code 08:

Command Body		
Station Address	Function Code	Data bytes received

Example: 01 08 00 02 00 04

Function Code 15 (0F hex)

Forces each coil in a sequence of coils to either ON or OFF.

Request message format for function code 15:

Command Body								
Station Address	Function Code	Start Address High Byte	Start Address Low Byte	Requested Number of Coil High Byte	Requested Number of Coil Low Byte	Byte Count	Force Data High Byte	Force Data Low Byte

Example: Request to force a series of 10 coils starting at address 00017 (11 hex) in ADAM-6000 module.

01 0F 00 11 00 0A 02 CD 01

The query data contents are two bytes: CD 01 hex, equal to 1100 1101 0000 0001 binary. The binary bits are mapped to the addresses in the following way.

Bit: 1 1 0 0 1 1 0 1 0 0 0 0 0 0 0 1

Address (000XX): 24 23 22 21 20 19 18 17 - - - - - - 26 25

Response message format for function code 08:

The normal responses return the station address, function code, start address, and requested number of coil forced.

Command Body					
Station Address	Function Code	Start Address High Byte	Start Address Low Byte	Requested Number of Coil High Byte	Requested Number of Coil Low Byte

Example: 01 0F 00 11 00 0A

Function Code 16 (10 hex)

Preset values into a sequence of holding registers.

Request message format for function code 16:

Command Body							
Station Address	Function Code	Start Address High Byte	Start Address Low Byte	Requested Number of Register High Byte	Requested Number of Register Low Byte	Byte Count	Data

Example: Preset constant #1 (address 40009) to 100.0 in ADAM-6000 module.

01 10 00 09 00 02 04 42 C8 00 00

Response message format for function code 08:

The normal responses return the station address, function code, start address, and requested number of registers preset.

Command Body					
Station Address	Function Code	Start Address High Byte	Start Address Low Byte	Requested Number of Register High Byte	Requested Number of Register Low Byte

Example: 01 10 00 09 00 02

B.2 ADAM-6000 I/O Modbus Mapping Table

B.2.1 ADAM-6015

7-ch RTD Input Module

Address 0X	Channel	Description	Attribute	Address 4X	Channel	Descrip-tion	Attribute
00101	0	Reset Historical Max. Value	R/W	40001	0	AI Value	Read
00102	1	Reset Historical Max. Value	R/W	40002	1	AI Value	Read
00103	2	Reset Historical Max. Value	R/W	40003	2	AI Value	Read
00104	3	Reset Historical Max. Value	R/W	40004	3	AI Value	Read
00105	4	Reset Historical Max. Value	R/W	40005	4	AI Value	Read
00106	5	Reset Historical Max. Value	R/W	40006	5	AI Value	Read
00107	6	Reset Historical Max. Value	R/W	40007	6	AI Value	Read
00108	-	Reserved	-	40008	-	Reserved	-
00109	Average Ch 0~6	Reset Historical Max. Value	R/W	40009	Average Ch 0 ~ 6	AI Value	Read
00111	0	Reset Historical Min. Value	R/W	40011	0	Historical Max. AI Value	Read
00112	1	Reset Historical Min. Value	R/W	40012	1	Historical Max. AI Value	Read
00113	2	Reset Historical Min. Value	R/W	40013	2	Historical Max. AI Value	Read
00114	3	Reset Historical Min. Value	R/W	40014	3	Historical Max. AI Value	Read
00115	4	Reset Historical Min. Value	R/W	40015	4	Historical Max. AI Value	Read
00116	5	Reset Historical Min. Value	R/W	40016	5	Historical Max. AI Value	Read
00117	6	Reset Historical Min. Value	R/W	40017	6	Historical Max. AI Value	Read
00118	-	Reserved	-	40018	-	Reserved	-
00119	Average Ch 0 ~ 6	Reset Historical Min. Value	R/W	40019	Average Ch 0 ~ 6	Historical Max. AI Value	Read
00121	0	Burnout Flag[1]	Read	40021	0	Historical Min. AI Value	Read
00122	1	Burnout Flag[1]	Read	40022	1	Historical Min. AI Value	Read

Address 0X	Channel	Description	Attribute	Address 4X	Channel	Descrip-tion	Attribute
00123	2	Burnout Flag[1]	Read	40023	2	Historical Min. AI Value	Read
00124	3	Burnout Flag[1]	Read	40024	3	Historical Min. AI Value	Read
00125	4	Burnout Flag[1]	Read	40025	4	Historical Min. AI Value	Read
00126	5	Burnout Flag[1]	Read	40026	5	Historical Min. AI Value	Read
00127	6	Burnout Flag[1]	Read	40027	6	Historical Min. AI Value	Read
				40028	-	Reserved	-
				40029	Average Ch 0 ~ 6	Historical Min. AI Value	Read
00131	0	High Alarm Flag[2]	Read	40305	0 ~ 15	GCL Inter-nal Flag Value	R/W
00132	1	High Alarm Flag[2]	Read				
00133	2	High Alarm Flag[2]	Read				
00134	3	High Alarm Flag[2]	Read				
00135	4	High Alarm Flag[2]	Read				
00136	5	High Alarm Flag[2]	Read				
00137	6	High Alarm Flag[2]	Read				
00138	-	Reserved	-				
00139	Average Ch 0 ~ 6	High Alarm Flag[2]	Read				
00141	0	Low Alarm Flag[3]	Read				
00142	1	Low Alarm Flag[3]	Read				
00143	2	Low Alarm Flag[3]	Read				
00144	3	Low Alarm Flag[3]	Read				
00145	4	Low Alarm Flag[3]	Read				
00146	5	Low Alarm Flag[3]	Read				
00147	6	Low Alarm Flag[3]	Read				
00148	-	Reserved	-				
00149	Average Ch 0 ~ 6	Low Alarm Flag[3]	Read				

Remarks:

1. When channel cannot detect RTD signal, this bit register will be 1.
2. User can configure High alarm value in the ADAM.NET utility. When AI value is higher than High alarm value, this bit will be 1.
3. User can configure the Low alarm value in the ADAM.NET utility. When AI value is lower than the Low alarm value, this bit will be 1.

B.2.2 ADAM-6017

8-ch Analog Input Module

Address 0X	Channel	Description	Attribute	Address 4X	Channel	Description	Attribute
00017	0	DO Value	R/W				
00018	1	DO Value	R/W				
00101	0	Reset Historical Max. Value	R/W	40001	0	AI Value	Read
00102	1	Reset Historical Max. Value	R/W	40002	1	AI Value	Read
00103	2	Reset Historical Max. Value	R/W	40003	2	AI Value	Read
00104	3	Reset Historical Max. Value	R/W	40004	3	AI Value	Read
00105	4	Reset Historical Max. Value	R/W	40005	4	AI Value	Read
00106	5	Reset Historical Max. Value	R/W	40006	5	AI Value	Read
00107	6	Reset Historical Max. Value	R/W	40007	6	AI Value	Read
00108	7	Reset Historical Max. Value	R/W	40008	7	AI Value	Read
00109	Average Ch 0 ~ 7	Reset Historical Max. Value	R/W	40009	Average Ch 0 ~ 7	AI Value	Read
00111	0	Reset Historical Min. Value	R/W	40011	0	Historical Max. AI Value	Read
00112	1	Reset Historical Min. Value	R/W	40012	1	Historical Max. AI Value	Read
00113	2	Reset Historical Min. Value	R/W	40013	2	Historical Max. AI Value	Read
00114	3	Reset Historical Min. Value	R/W	40014	3	Historical Max. AI Value	Read
00115	4	Reset Historical Min. Value	R/W	40015	4	Historical Max. AI Value	Read
00116	5	Reset Historical Min. Value	R/W	40016	5	Historical Max. AI Value	Read
00117	6	Reset Historical Min. Value	R/W	40017	6	Historical Max. AI Value	Read

Address 0X	Channel	Description	Attribute	Address 4X	Channel	Description	Attribute
00118	7	Reset Historical Min. Value	R/W	40018	7	Historical Max. AI Value	Read
00119	Average Ch 0 ~ 7	Reset Historical Min. Value	R/W	40019	Average Ch 0 ~ 7	Historical Max. AI Value	Read
00131	0	High Alarm Flag[1]	Read	40021	0	Historical Min. AI Value	Read
00132	1	High Alarm Flag[1]	Read	40022	1	Historical Min. AI Value	Read
00133	2	High Alarm Flag[1]	Read	40023	2	Historical Min. AI Value	Read
00134	3	High Alarm Flag[1]	Read	40024	3	Historical Min. AI Value	Read
00135	4	High Alarm Flag[1]	Read	40025	4	Historical Min. AI Value	Read
00136	5	High Alarm Flag[1]	Read	40026	5	Historical Min. AI Value	Read
00137	6	High Alarm Flag[1]	Read	40027	6	Historical Min. AI Value	Read
00138	7	High Alarm Flag[1]	Read	40028	7	Historical Min. AI Value	Read
00139	Average Ch 0 ~ 7	High Alarm Flag[1]	Read	40029	Average Ch 0 ~ 7	Historical Min. AI Value	Read
00141	0	Low Alarm Flag[2]	Read	40305	0 ~ 15	GCL Internal Flag Value	R/W
00142	1	Low Alarm Flag[2]	Read				
00143	2	Low Alarm Flag[2]	Read				
00144	3	Low Alarm Flag[2]	Read				
00145	4	Low Alarm Flag[2]	Read				
00146	5	Low Alarm Flag[2]	Read				
00147	6	Low Alarm Flag[2]	Read				
00148	7	Low Alarm Flag[2]	Read				
00149	Average Ch 0 ~ 7	Low Alarm Flag[2]	Read				

Remarks:

1. User can configure the High alarm value in the ADAM.NET utility. When AI value is higher than the High alarm, this bit will be 1.
2. Users can configure the Low alarm value in the ADAM.NET utility. When AI value is lower than the Low alarm, this bit will be 1.

B.2.3 ADAM-6018

8-ch Thermocouple Input Module

Address 0X	Channel	Description	Attribute	Address 4X	Channel	Description	Attribute
00017	0	DO Value	R/W				
00018	1	DO Value	R/W				
00019	2	DO Value	R/W				
00020	3	DO Value	R/W				
00021	4	DO Value	R/W				
00022	5	DO Value	R/W				
00023	6	DO Value	R/W				
00024	7	DO Value	R/W				
00101	0	Reset Historical Max. Value	R/W	40001	0	AI Value	Read
00102	1	Reset Historical Max. Value	R/W	40002	1	AI Value	Read
00103	2	Reset Historical Max. Value	R/W	40003	2	AI Value	Read
00104	3	Reset Historical Max. Value	R/W	40004	3	AI Value	Read
00105	4	Reset Historical Max. Value	R/W	40005	4	AI Value	Read
00106	5	Reset Historical Max. Value	R/W	40006	5	AI Value	Read
00107	6	Reset Historical Max. Value	R/W	40007	6	AI Value	Read
00108	7	Reset Historical Max. Value	R/W	40008	7	AI Value	Read
00109	Average Ch 0 ~ 7	Reset Historical Max. Value	R/W	40009	Average Ch 0 ~ 7	AI Value	Read
00111	0	Reset Historical Min. Value	R/W	40011	0	Historical Max. AI Value	Read
00112	1	Reset Historical Min. Value	R/W	40012	1	Historical Max. AI Value	Read
00113	2	Reset Historical Min. Value	R/W	40013	2	Historical Max. AI Value	Read
00114	3	Reset Historical Min. Value	R/W	40014	3	Historical Max. AI Value	Read
00115	4	Reset Historical Min. Value	R/W	40015	4	Historical Max. AI Value	Read
00116	5	Reset Historical Min. Value	R/W	40016	5	Historical Max. AI Value	Read

Address 0X	Channel	Description	Attribute	Address 4X	Channel	Description	Attribute
00117	6	Reset Historical Min. Value	R/W	40017	6	Historical Max. AI Value	Read
00118	7	Reset Historical Min. Value	R/W	40018	7	Historical Max. AI Value	Read
00119	Average Ch 0 ~ 7	Reset Historical Min. Value	R/W	40019	Average Ch 0 ~ 7	Historical Max. AI Value	Read
00121	0	Burnout Flag[1]	Read	40021	0	Historical Min. AI Value	Read
00122	1	Burnout Flag[1]	Read	40022	1	Historical Min. AI Value	Read
00123	2	Burnout Flag[1]	Read	40023	2	Historical Min. AI Value	Read
00124	3	Burnout Flag[1]	Read	40024	3	Historical Min. AI Value	Read
00125	4	Burnout Flag[1]	Read	40025	4	Historical Min. AI Value	Read
00126	5	Burnout Flag[1]	Read	40026	5	Historical Min. AI Value	Read
00127	6	Burnout Flag[1]	Read	40027	6	Historical Min. AI Value	Read
00128	7	Burnout Flag[1]	Read	40028	7	Historical Min. AI Value	Read
				40029	Average Ch 0 ~ 7	Historical Min. AI Value	Read
00131	0	High Alarm Flag[2]	Read	40305	0~15	GCL Internal Flag Value	R/W
00132	1	High Alarm Flag[2]	Read				
00133	2	High Alarm Flag[2]	Read				
00134	3	High Alarm Flag[2]	Read				
00135	4	High Alarm Flag[2]	Read				
00136	5	High Alarm Flag[2]	Read				
00137	6	High Alarm Flag[2]	Read				
00138	7	High Alarm Flag[2]	Read				
00139	Average Ch 0 ~ 7	High Alarm Flag[2]	Read				
00141	0	Low Alarm Flag[3]	Read				
00142	1	Low Alarm Flag[3]	Read				
00143	2	Low Alarm Flag[3]	Read				
00144	3	Low Alarm Flag[3]	Read				
00145	4	Low Alarm Flag[3]	Read				
00146	5	Low Alarm Flag[3]	Read				
00147	6	Low Alarm Flag[3]	Read				
00148	7	Low Alarm Flag[3]	Read				
00149	Average Ch 0 ~ 7	Low Alarm Flag[3]	Read				

Remarks:

1. When the specific channel cannot detect Thermocouple signal, this bit register will be 1.
2. User can configure the High alarm value in the ADAM.NET utility. When AI value is higher than High alarm value, this bit will be 1.
3. User can configure the Low alarm value in the ADAM.NET utility. When AI value is lower than the Low alarm value, this bit will be 1.

B.2.4 ADAM-6024

12-Ch Universal I/O Module

Address 0X	Ch	Description	Attribute	Address 4X	Ch	Description	Attribute
00001	0	DI Value	Read	40001	0	AI Value	Read
00002	1	DI Value	Read	40002	1	AI Value	Read
				40003	2	AI Value	Read
00017	0	DO Value	R/W	40004	3	AI Value	Read
00018	1	DO Value	R/W	40005	4	AI Value	Read
				40006	5	AI Value	Read
				40011	0	AO Value	R/W
				40012	1	AO Value	R/W
				40021	0	AI Status[1]	Read
				40022	1	AI Status[1]	Read
				40023	2	AI Status[1]	Read
				40024	3	AI Status[1]	Read
				40025	4	AI Status[1]	Read
				40026	5	AI Status[1]	Read

Remarks:

1. AI Status: Bit Value 0: normal

Bit Value 1: over high

Bit Value 2: over low

Bit Value 0: invalid calibration

B.2.5 ADAM-6052 16-ch Digital I/O Module

Address 0X	Ch	Description	Attribute	Address 4X	Ch	Description	Attribute
00001	0	DI Value	Read	40001~40002	0	Counter/Frequency Value[1]	Read
00002	1		Read	40003~40004	1		Read
00003	2		Read	40005~40006	2		Read
00004	3		Read	40007~40008	3		Read
00005	4		Read	40009~40010	4		Read
00006	5		Read	40011~40012	5		Read
00007	6		Read	40013~40014	6		Read
00008	7		Read	40015~40016	7		Read
00017	0	DO Value	R/W	40017~40018	0	Pulse Output Low Level Width[2]	R/W
00018	1		R/W	40019~40020	1		R/W
00019	2		R/W	40021~40022	2		R/W
00020	3		R/W	40023~40024	3		R/W
00021	4		R/W	40025~40026	4		R/W
00022	5		R/W	40027~40028	5		R/W
00023	6		R/W	40029~40030	6		R/W
00024	7		R/W	40031~40032	7		R/W
00033	0	Counter Start(1)/ Stop(0)	R/W	40033~40034	0	Pulse Output High Level Width[2]	R/W
00034		Clear Counter(1)	Write	40035~40036	1		R/W
00035		Clear Overflow[3]	R/W	40037~40038	2		R/W
00036		DI Latch Status[4]	R/W	40039~40040	3		R/W
00037	1	Counter Start(1)/ Stop(0)	R/W	40041~40042	4		R/W
00038		Clear Counter(1)	Write	40043~40044	5		R/W
00039		Clear Overflow[3]	R/W	40045~40046	6		R/W
00040		DI Latch Status[4]	R/W	40047~40048	7		R/W
00041	2	Counter Start(1)/ Stop(0)	R/W				
00042		Clear Counter(1)	Write	40049~40050	0	Set Absolute Pulse[5]	R/W
00043		Clear Overflow[3]	R/W	40051~40052	1		R/W
00044		DI Latch Status[4]	R/W	40053~40054	2		R/W
00045	3	Counter Start(1)/ Stop(0)	R/W	40055~40056	3		R/W
00046		Clear Counter(1)	Write	40057~40058	4		R/W

Address 0X	Ch	Description	Attribute	Address 4X	Ch	Description	Attribute
00047	3	Clear Overflow[3]	R/W	40059~40060	5	Set Absolute Pulse[5]	R/W
00048		DI Latch Status[4]	R/W	40061~40062	6		R/W
00049	4	Counter Start(1)/ Stop(0)	R/W	40063~40064	7		R/W
00050		Clear Coun-ter(1)	Write				
00051		Clear Overflow[3]	R/W	40065~40066	0	Set Incremental Pulse[6]	R/W
00052		DI Latch Status[4]	R/W	40067~40068	1		R/W
00053	5	Counter Start(1)/ Stop(0)	R/W	40069~40070	2		R/W
00054		Clear Coun-ter(1)	Write	40071~40072	3		R/W
00055		Clear Overflow[3]	R/W	40073~40074	4		R/W
00056		DI Latch Status[4]	R/W	40075~40076	5		R/W
00057	6	Counter Start(1)/ Stop(0)	R/W	40077~40078	6		R/W
00058		Clear Coun-ter(1)	Write	40079~40080	7		R/W
00059		Clear Overflow[3]	R/W				
00060		DI Latch Status[4]	R/W	40301	All	DI Value	Read
00061	7	Counter Start(1)/ Stop(0)	R/W	40303	All	DO Value	R/W
00062		Clear Coun-ter(1)	Write	40305	0~15	GCL Internal Flag Value	R/W
00063		Clear Overflow[3]	R/W				
00064		DI Latch Status[4]	R/W				

Remarks:

1. How to retrieve the counter/frequency value:
   Counter (decimal) = (value of 40002) x 65535 + (value of 40001)

   Frequency (decimal) = (value of 40001)/10 Hz
2. Time Unit: 0.1 ms
3. If the count number is overflow, this bit will be 1. Once this bit is read, the value will return to 0.

4. When DI channel is configured as “High to low latch” or “Low to high latch”, this bit will be 1 if the latch condition occurs. After that, value of this bit will keep 1 until user writes 0 to this bit (clear the latch status).
5. Decide how many pulses will be generated. When user writes 0 to this bit, it will continuously generate pulse.
6. During the pulse generation, user can use this bit to generate more pulses. For example, “Absolute pulse” is set as 100. During its generation, user can set “Incremental pulse” as 10. After the 100 pulses are generated, the extra 10 pulses will continue to be generated.

B.2.6 ADAM-6060/6060W/6066

12-ch DI & Relay Module

Address 0X	Ch	Description	Attribute	Address 4X	Ch	Description	Attribute
00001	0	DI Value	Read	40001~40002	0	Counter/Frequency Value[1]	Read
00002	1		Read	40003~40004	1		Read
00003	2		Read	40005~40006	2		Read
00004	3		Read	40007~40008	3		Read
00005	4		Read	40009~40010	4		Read
00006	5		Read	40011~40012	5		Read
00017	0	DO Value	R/W	40013~40014	0	Pulse Output Low Level Width[2]	R/W
00018	1		R/W	40015~40016	1		R/W
00019	2		R/W	40017~40018	2		R/W
00020	3		R/W	40019~40020	3		R/W
00021	4		R/W	40021~40022	4		R/W
00022	5		R/W	40023~40024	5		R/W
00033	0	Counter Start(1)/ Stop(0)	R/W	40025~40026	0	Pulse Output High Level Width[2]	R/W
00034		Clear Counter(1)	Write	40027~40028	1		R/W
00035		Clear Overflow[3]	R/W	40029~40030	2		R/W
00036		DI Latch Status[4]	R/W	40031~40032	3		R/W
00037	1	Counter Start(1)/ Stop(0)	R/W	40033~40034	4		R/W
00038		Clear Counter(1)	Write	40035~40036	5		R/W
00039		Clear Overflow[3]	R/W				
00040		DI Latch Status[4]	R/W	40037~40038	0	Set Absolute Pulse[5]	R/W

Address 0X	Ch	Description	Attribute	Address 4X	Ch	Description	Attribute
00041	2	Counter Start(1)/ Stop(0)	R/W	40039~40040	1	Set Absolute Pulse[5]	R/W
00042		Clear Coun-ter(1)	Write	40041~40042	2		R/W
00043		Clear Overflow[3]	R/W	40043~40044	3		R/W
00044		DI Latch Status[4]	R/W	40045~40046	4		R/W
00045	3	Counter Start(1)/ Stop(0)	R/W	40047~40048	5		R/W
00046		Clear Coun-ter(1)	Write				
00047		Clear Overflow[3]	R/W	40049~40050	0	Set Incremental Pulse[6]	R/W
00048		DI Latch Status[4]	R/W	40051~40052	1		R/W
00049	4	Counter Start(1)/ Stop(0)	R/W	40053~40054	2		R/W
00050		Clear Coun-ter(1)	Write	40055~40056	3		R/W
00051		Clear Overflow[3]	R/W	40057~40058	4		R/W
00052		DI Latch Status[4]	R/W	40059~40060	5		R/W
00053	5	Counter Start(1)/ Stop(0)	R/W				
00054		Clear Coun-ter(1)	Write	40301	All	DI Value	Read
00055		Clear Overflow[3]	R/W	40303	All	DO Value	R/W
00056		DI Latch Status[4]	R/W	40305	0~15	GCL Internal Flag Value[7]	R/W

Remarks:

1. How to retrieve the counter/frequency value:
   Counter (decimal) = (value of 40002) x 65535 + (value of 40001)

   Frequency (decimal) = (value of 40001)/10 Hz
2. Time Unit: 0.1 ms
3. If the count number is overflow, this bit will be 1. Once this bit is read, the value will return to 0.
4. When DI channel is configured as “High to low latch” or “Low to high latch”, this bit will be 1 if the latch condition occurs. After

that, value of this bit will keep 1 until user writes 0 to this bit (clear the latch status).

5. Decide how many pulses will be generated. When user writes 0 to this bit, it will continuously generate pulse.
6. During the pulse generation, user can use this bit to generate more pulses. For example, “Absolute pulse” is set as 100. During its generation, user can set “Incremental pulse” as 10. After the 100 pulses are generated, the extra 10 pulses will continue to be generated.
7. Only for ADAM-6060 and ADAM-6066

APPENDIX C

Grounding Reference

Appendix C Grounding Reference

C.1 Field Grounding and Shielding Application

Overview

Unfortunately, it's impossible to finish a system integration task at one time. We always meet some trouble in the field. A communication network or system isn't stable, induced noise or equipment is damaged or there are storms. However, the most usual issue is just simply improper wiring, ie, grounding and shielding. You know the 80/20 rule in our life: we spend 20% time for 80% work, but 80% time for the last 20% of the work. So is it with system integration: we pay 20% for Wire / Cable and 0% for Equipment. However, 80% of reliability depends on Grounding and Shielding. In other words, we need to invest more in that 20% and work on these two issues to make a highly reliable system. This application note brings you some concepts about field grounding and shielding. These topics will be illustrated in the following pages.

Grounding

1.1 The 'Earth' for reference

1.2 The 'Frame Ground' and 'Grounding Bar'

1.3 Normal Mode and Common Mode

1.4 Wire impedance

1.5 Single Point Grounding

2. Shielding

2.1 Cable Shield

2.2 System Shielding

3. Noise Reduction Techniques

4. Check Point List

C.2 Grounding

C.2.1 The 'Earth' for Reference

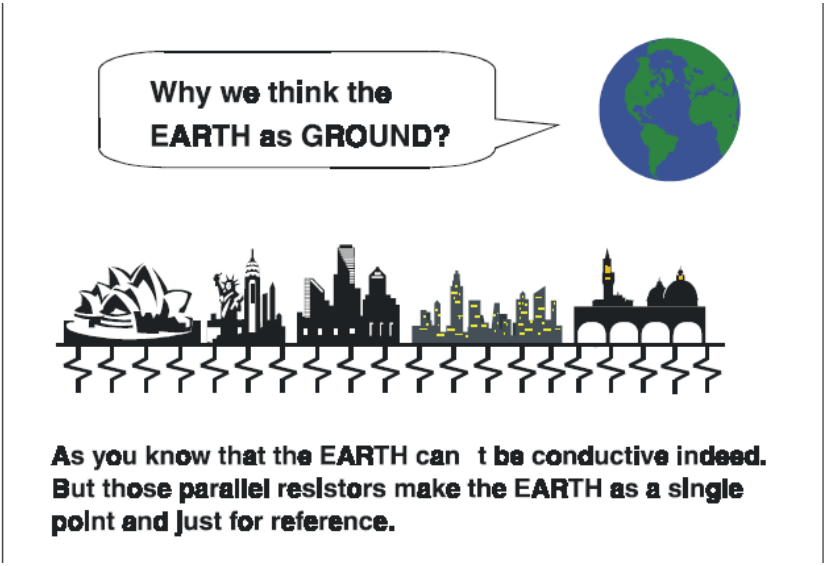


Figure C.1: Think of the Earth as a Ground.

As you know, the EARTH cannot be conductive. However, all build- ings lie on, or in, the EARTH. Steel, concrete and associated cables (such as lighting arresters) and power system were connected to EARTH. Think of them as resistors. All of those infinite parallel resistors make the EARTH as a single reference point.

C.2.2 The 'Frame Ground' and 'Grounding Bar'

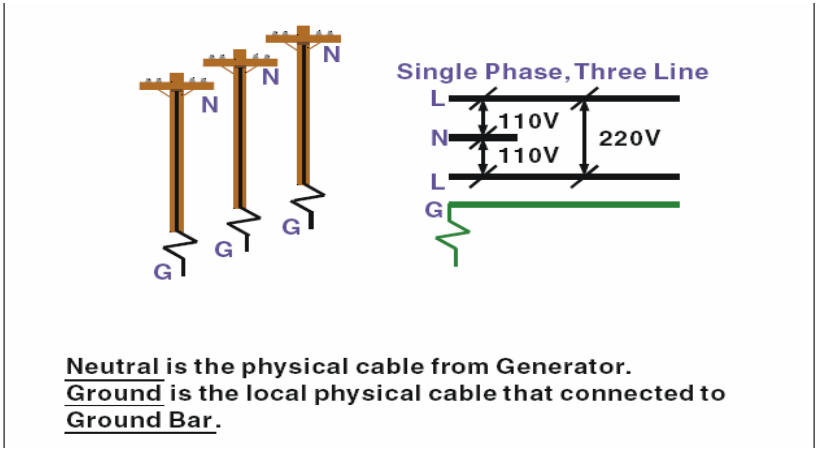


Figure C.2: Grounding Bar

Grounding is one of the most important issues for our system. Just like Frame Ground of the computer, this signal offers a reference point of the electronic circuit inside the computer. If we want to communicate with this computer, both Signal Ground and Frame Ground should be connected to make a reference point of each other's electronic circuit. Generally speaking, it is necessary to install an individual grounding bar for each system, such as computer networks, power systems, telecommunication networks, etc. Those individual grounding bars not only provide the individual reference point, but also make the earth a our ground!

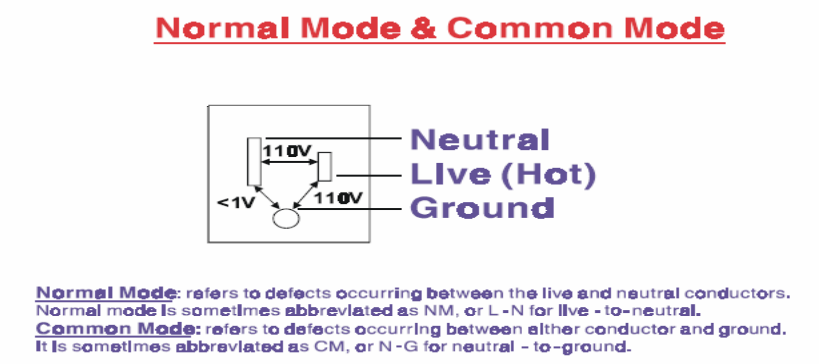


Figure C.3: Normal and Common Mode.

C.2.3 Normal Mode and Common Mode

Have you ever tried to measure the voltage between a live circuit and a concrete floor? How about the voltage between neutral and a concrete floor? You will get nonsense values. 'Hot' and 'Neutral' are just relational signals: you will get 110VAC or 220VAC by measuring these signals. Normal mode and common mode just show you that the Frame Ground is the most important reference signal for all the systems and equipments.

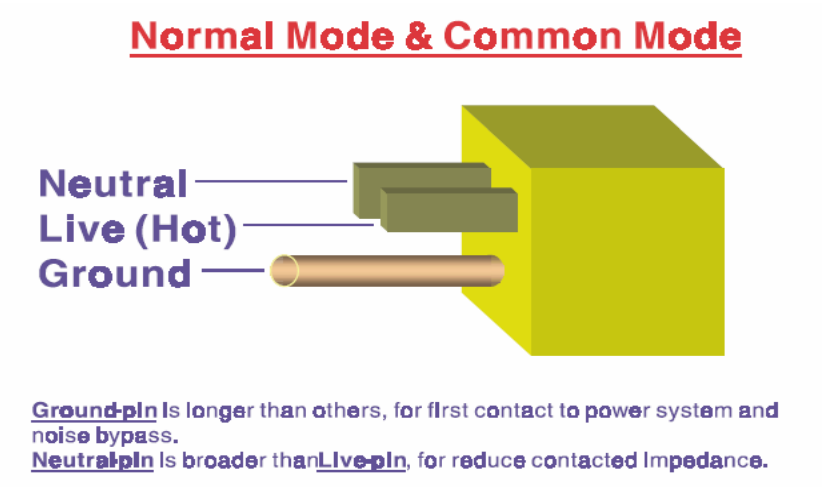


Figure C.4: Normal and Common Mode.

- Ground-pin is longer than others, for first contact to power system and noise bypass.

- Neutral-pin is broader than Live-pin, for reducing contact impedance.

C.2.4 Wire impedance

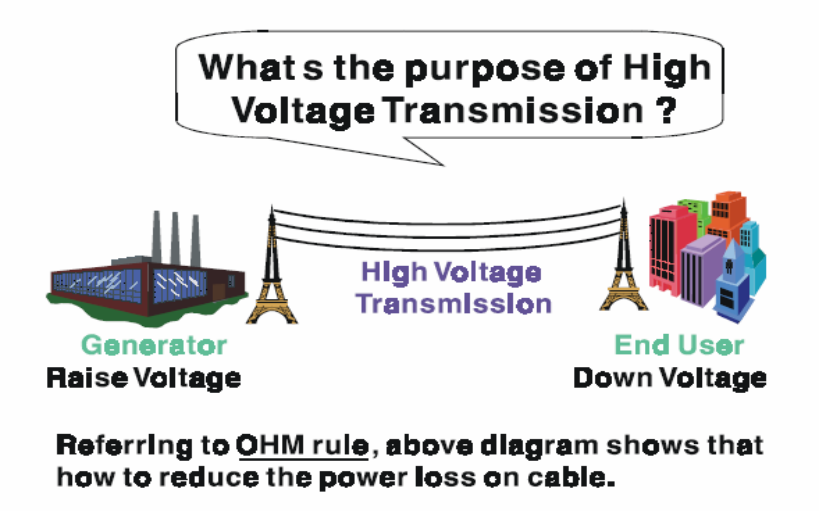


Figure C.5: High Voltage Transmission

- What's the purpose of high voltage transmission? We have all seen high voltage transmission towers. The power plant raises the voltage while generating the power, then a local power station steps down the voltage. What is the purpose of high voltage transmission wires ? According to the energy formula, P = V * I, the current is reduced when the voltage is raised. As you know, each cable has impedance because of the metal it is made of. Referring to Ohm's Law, (V = I * R) this decreased current means lower power losses in the wire. So, high voltage lines are for reducing the cost of moving electrical power from one place to another.

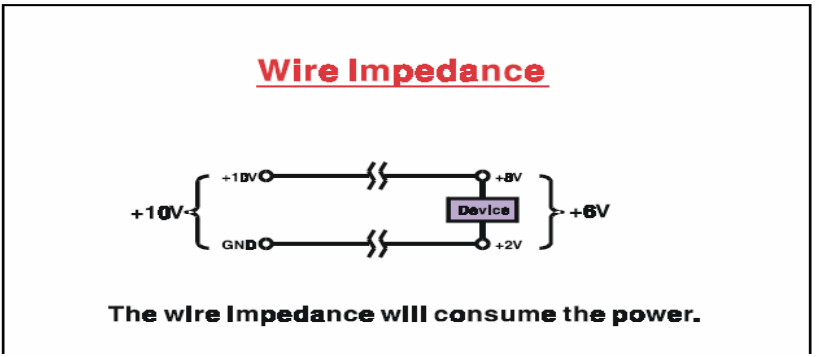


Figure C.6: Wire Impedance

C.2.5 Single Point Grounding

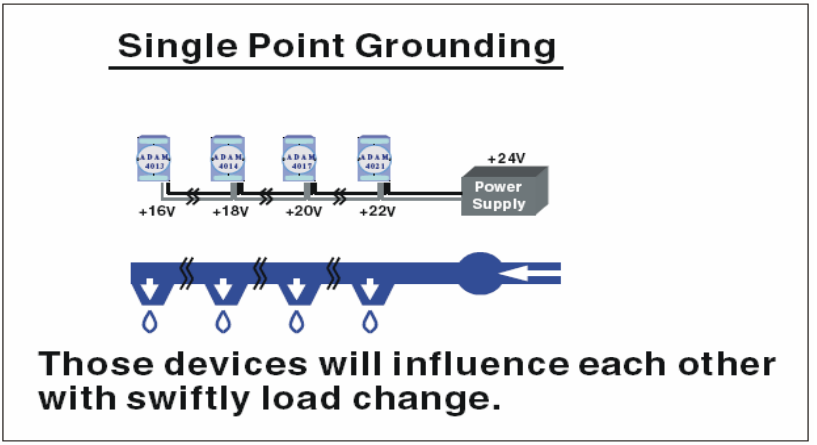


Figure C.7: Single Point Grounding (1)

- What's Single Point Grounding? Maybe you have had an unpleasant experience while taking a hot shower in Winter. Someone turns on a hot water faucet somewhere else. You will be impressed with the cold water! The bottom diagram above shows an example of how devices will influence each other with swift load change. For example, normally we turn on all the four hydrants for testing. When you close the hydrant 3 and hydrant 4, the other two hydrants will get more flow. In other words, the hydrant cannot keep a constant flow rate.

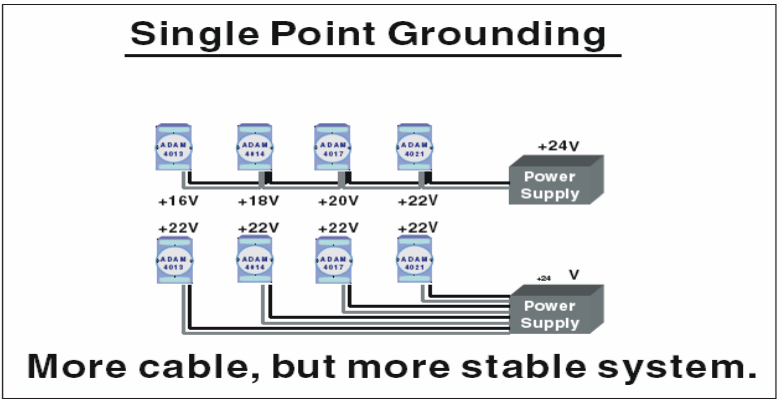


Figure C.8: Single point grounding (2)

The above diagram shows you that a single point grounding system will be a more stable system. If you use thin cable for powering these devices, the end device will actually get lower power. The thin cable will consume the energy.

C.3 Shielding

C.3.1 Cable Shield

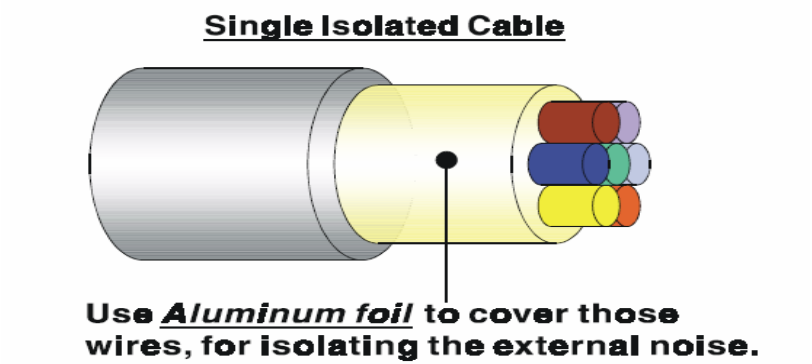


Figure C.9: Single isolated cable

- Single isolated cable The diagram shows the structure of an isolated cable. You see the isolated layer which is spiraled Aluminum foil to cover the wires. This spiraled structure makes a layer for shielding the cables from external noise.

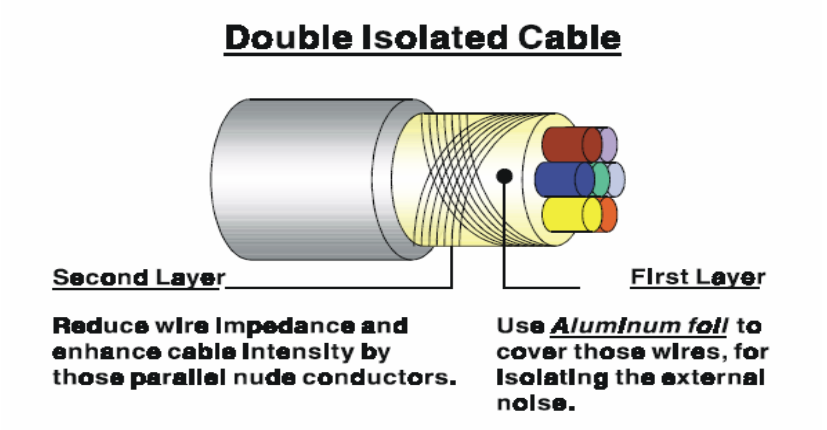


Figure C.10: Double isolated cable

- Double isolated cable Figure 10 is an example of a double isolated cable. The first isolating layer of spiraled aluminum foil covers the conductors. The second isolation layer is several bare conductors that spiral and cross over the first shield layer. This spiraled structure makes an isolated layer for reducing external noise. Additionally, follow these tips just for your reference.

- The shield of a cable cannot be used for signal ground. The shield is designed for carrying noise, so the environment noise will couple and interfere with your system when you use the shield as signal ground.
- The higher the density of the shield - the better, especially for communication network.
- Use double isolated cable for communication network / AI / AO.
- Both sides of shields should be connected to their frame while inside the device. (for EMI consideration)
- Don’t strip off too long of plastic cover for soldering.

C.3.2 System Shielding

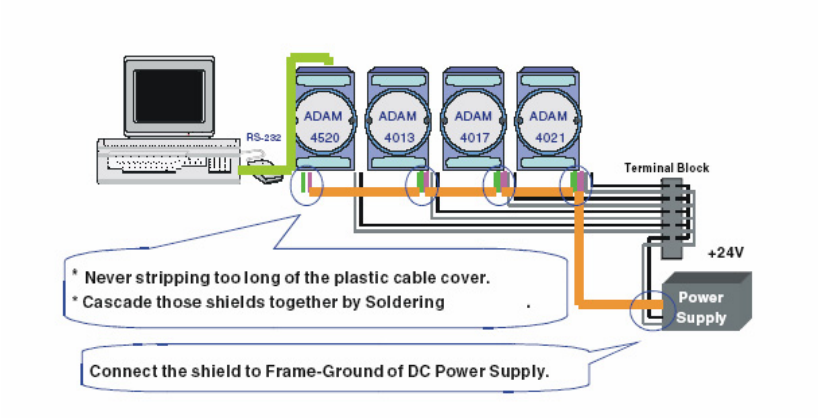


Figure C.11: System Shielding

- Never stripping too much of the plastic cable cover. This is improper and can destroy the characteristics of the Shielded-Twisted-Pair cable. Besides, the bare wire shield easily conducts the noise.
- Cascade these shields together by soldering. Please refer to following page for further detailed explanation.
- Connect the shield to Frame Ground of DC power supply to force the conducted noise to flow to the frame ground of the DC power supply. (The 'frame ground' of the DC power supply should be connected to the system ground)

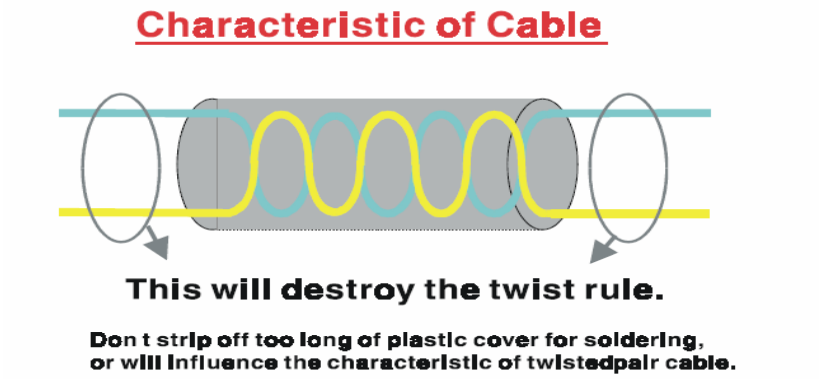


Figure C.12: The characteristic of the cable

• The characteristic of the cable Don't strip off too much insulation for soldering. This could change the effectiveness of the Shielded-Twisted-Pair cable and open a path to introduce unwanted noise.

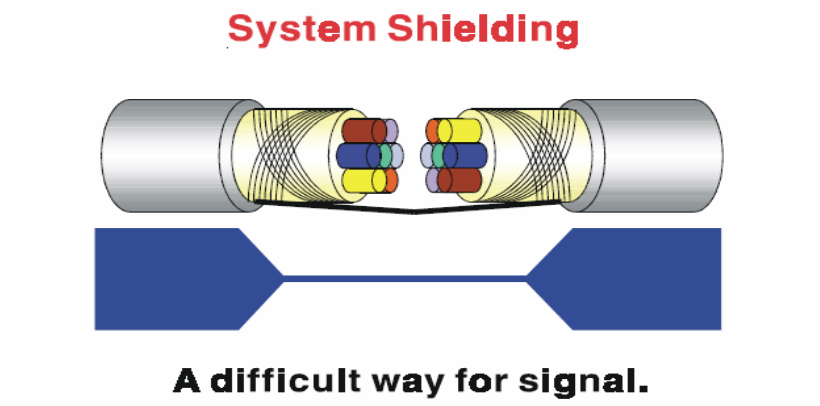


Figure C.13: System Shielding (1)

• Shield connection (1)

If you break into a cable, you might get in a hurry to achieve your goal. As in all electronic circuits, a signal will use the path of least resis- tance. If we make a poor connection between these two cables we will make a poor path for the signal. The noise will try to find another path for easier flow.

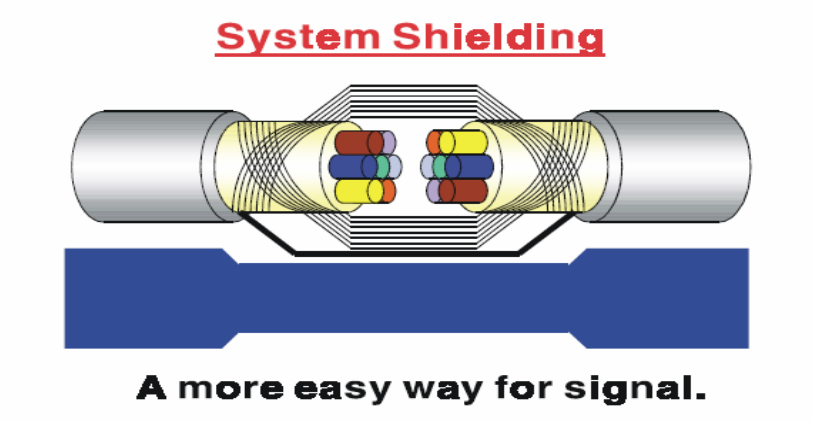


Figure C.14: System Shielding (2)

- Shield connection (2)

The previous diagram shows you that the fill soldering just makes an easier way for the signal.

C.4 Noise Reduction Techniques

- Isolate noise sources in shielded enclosures.
- Place sensitive equipment in shielded enclosure and away from computer equipment.
- Use separate grounds between noise sources and signals.
- Keep ground/signal leads as short as possible.
- Use Twisted and Shielded signal leads.
- Ground shields on one end ONLY while the reference grounds are not the same.
- Check for stability in communication lines.
- Add another Grounding Bar if necessary.
- The diameter of power cable must be over 2.0 mm_2.
- Independent grounding is needed for A/I, A/O, and communication network while using a jumper box.

- Use noise reduction filters if necessary. (TVS, etc)
- You can also refer to FIPS 94 Standard. FIPS 94 recommends that the computer system should be placed closer to its power source to eliminate load-induced common mode noise.

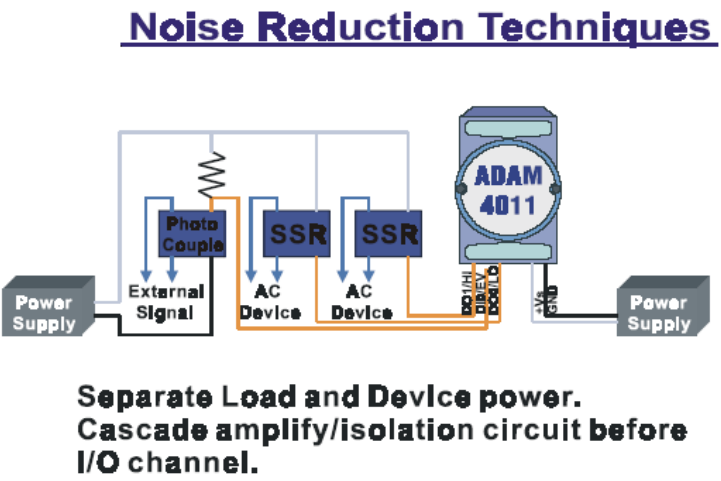


Figure C.15: Noise Reduction Techniques

C.5 Check Point List

- Follow the single point grounding rule?
- Normal mode and common mode voltage?
- Separate the DC and AC ground?
- Reject the noise factor?
- The shield is connected correctly?
- Wire size is correct?
- Soldered connections are good?
- The terminal screw are tight?