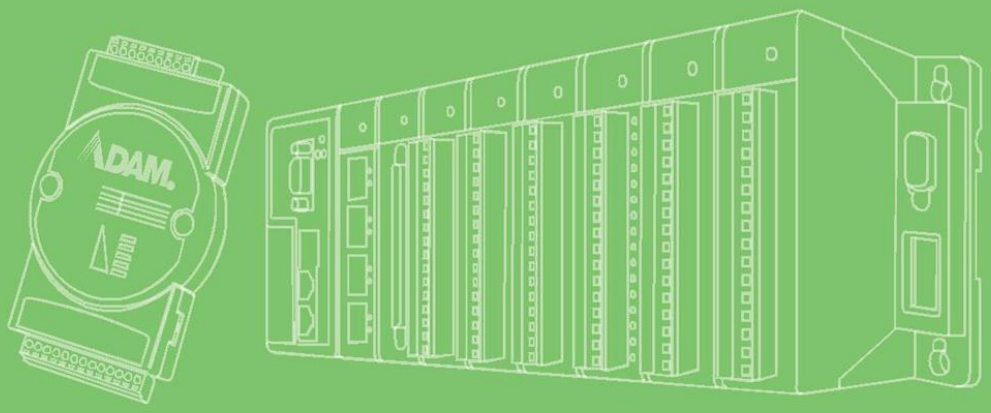


User Manual

Advantech CODESYS for ADAM-5560

CONTENTS

Chapter 1

1. Introduction

1.1. About This Manual

This document describes the use of the CODESYS programming environment and the Wince runtime system for the Advantech ADAM-5560 series products.

Advantech provides add-on package for CoDeSys which allows developers and end users to connected I/O modules; perform configurations, and simple testing of the I/O.

This manual supplies information about how to apply CoDeSys to control Advantech ADAM-5560, including software installation, writing a new program in CoDeSys to testing ADAM I/O modules.

1.2. Organization of This Manual

This user manual is divided into the following sections:

Introduction

This section gives the user a basic idea of this manual.

Installations

This section provides instructions on how to install CoDeSys and Advantech Add-on Package

Create and run a project

This section gives the new user a walk-through in creating a simple program.

Advantech I/O Modules

This section introduces the detail configuration and mapping variables of Advantech ADAM I/O modules

Diagnosis and Troubleshooting

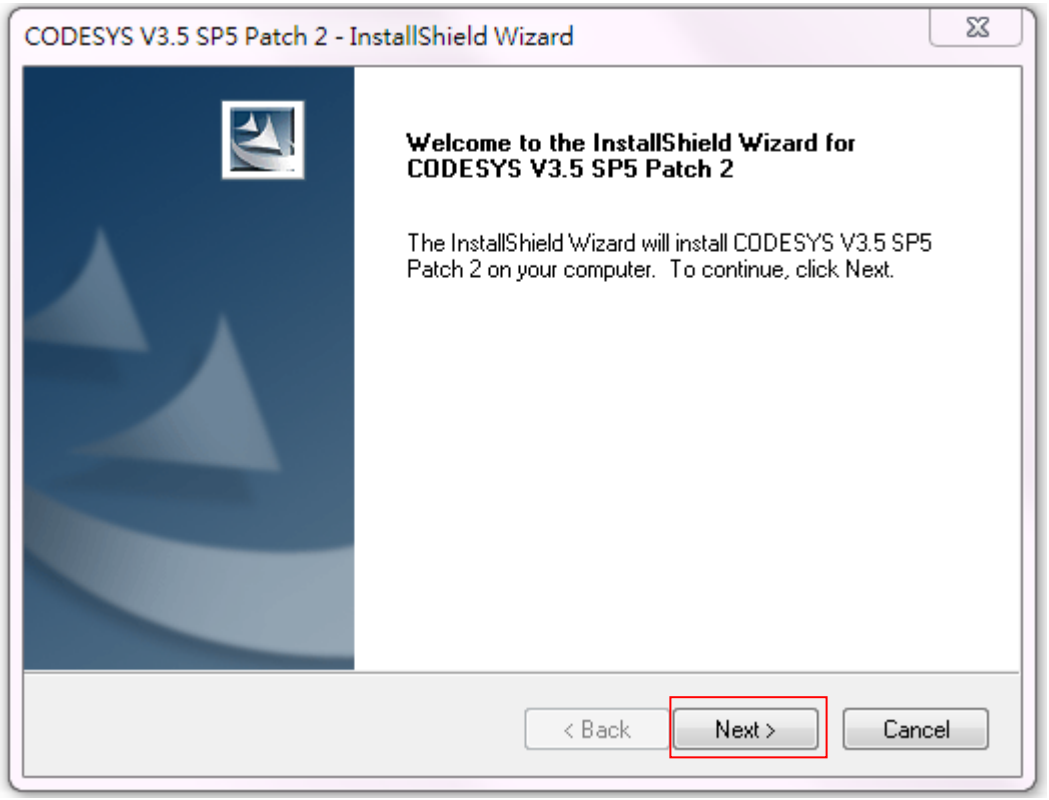
This section provides instructions on how to troubleshooting and diagnose operation mistakes or module errors.

Chapter 2

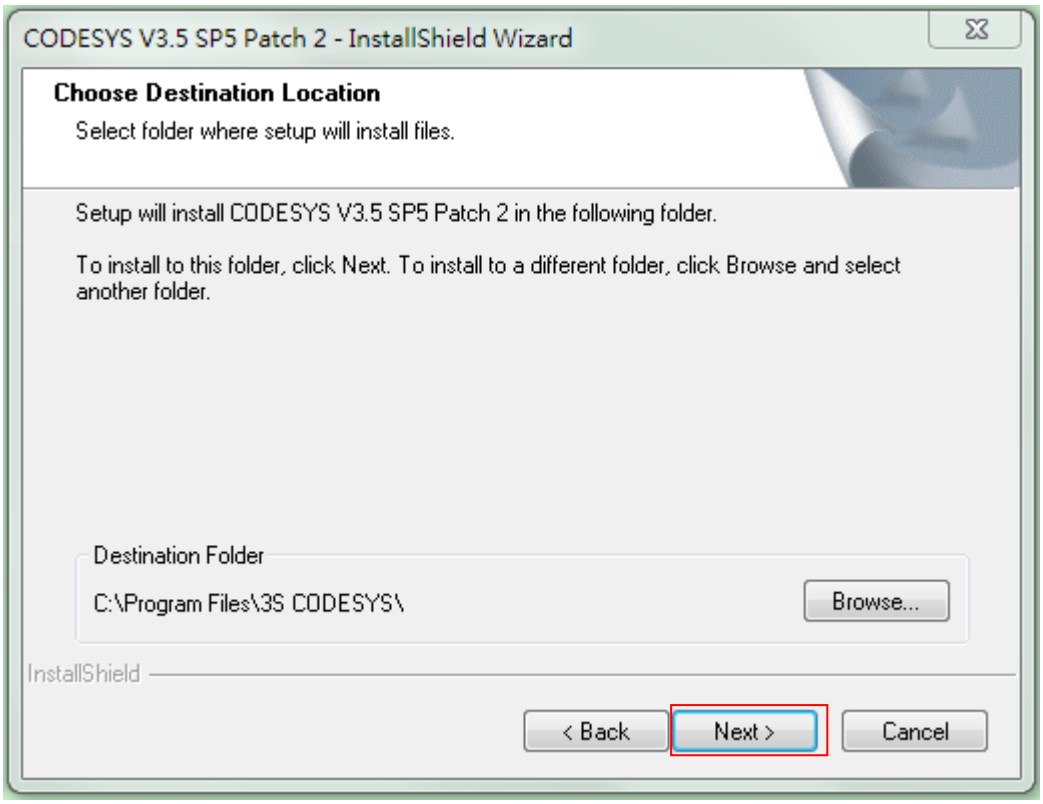
2. Installations

2.1. CODESYS Installation

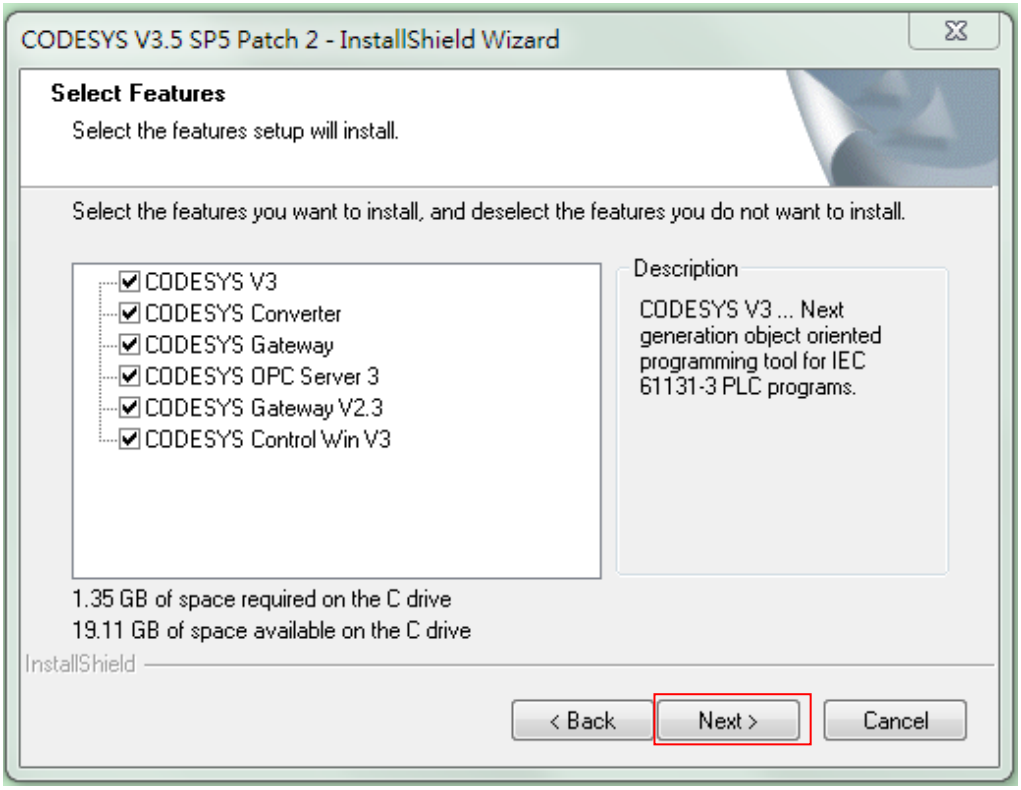
Step1: Double click and execute the "**Setup_CODESYSV<Version>.exe"** to start the installation assistant and then click Next to continue.



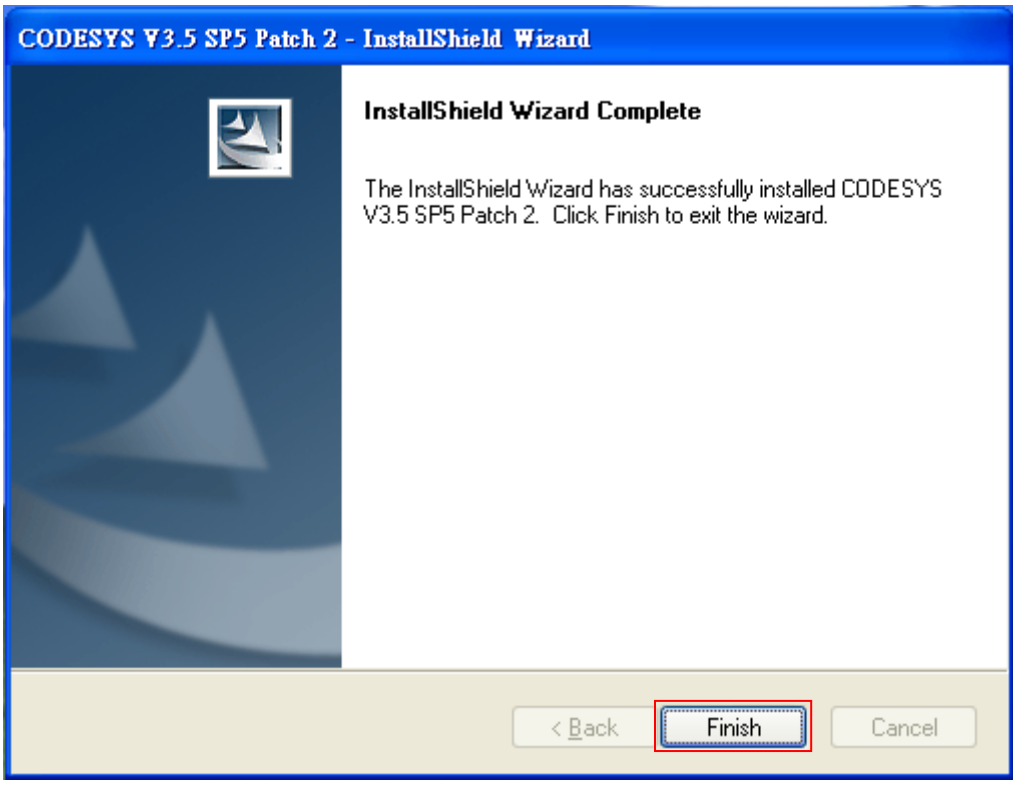
Step 2: You will then be prompted for the installation location. By default, CoDeSys will install to C:\Program Files\3S CODESYS, but you can specify the location or folder name of your choice. Click Next to proceed.



Step 3: Select all features and then click Next to proceed.



Step 4: Complete to install CoDeSys and you'll see CoDeSys icon  which is available on the desktop. Click "Finish" to close the installation wizard.



2.2. Add-on Package Installation

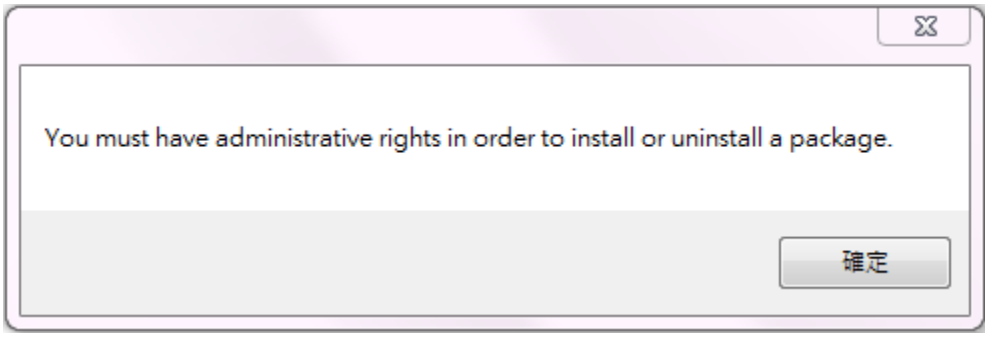
Now that you have CoDeSys on your system, you'll want to do a few steps to install add-on package on your environment.

2.2.1. First-time Installation

Step 1: Install the latest version of the add-on package by double-clicking the executable "**Advantech ADAM CODESYS ADD ON.package"**.

Note!

If you get an error stating "you must have administrative rights in order to install or uninstall a package" on Window 7, you should turn UAC off.

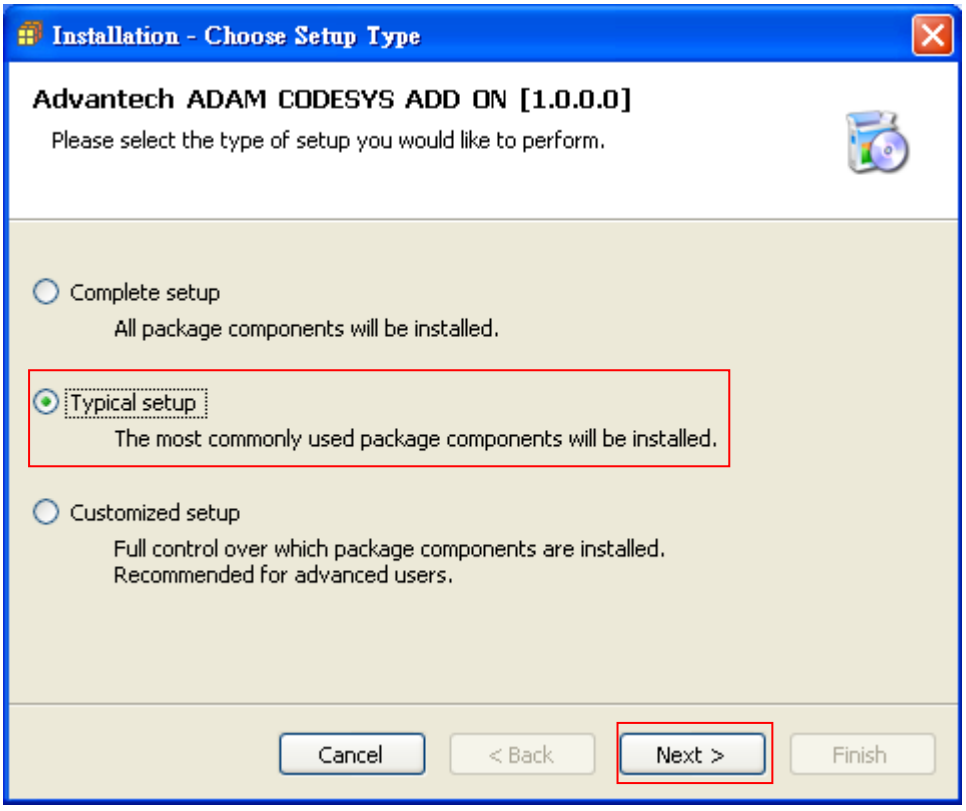


(a) Open User Account Control Settings by clicking the **Start** button, and then clicking **Control Panel**. In the search box, type **uac**, and then click **Change User Account Control settings**.

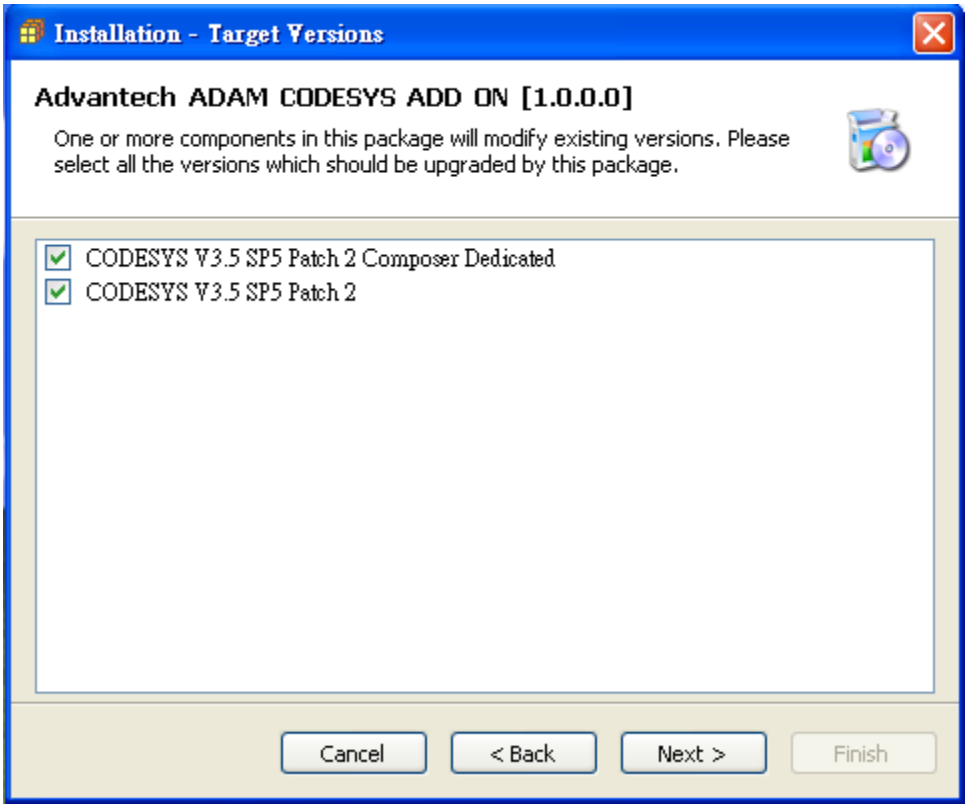
(b) Move the slider to the Never notify position, and then click **OK**. If you're prompted for an administrator password or confirmation, type the password or provide confirmation. You will need to restart your computer for UAC to be turned off.

For more information about notification options, please refer to Microsoft website.

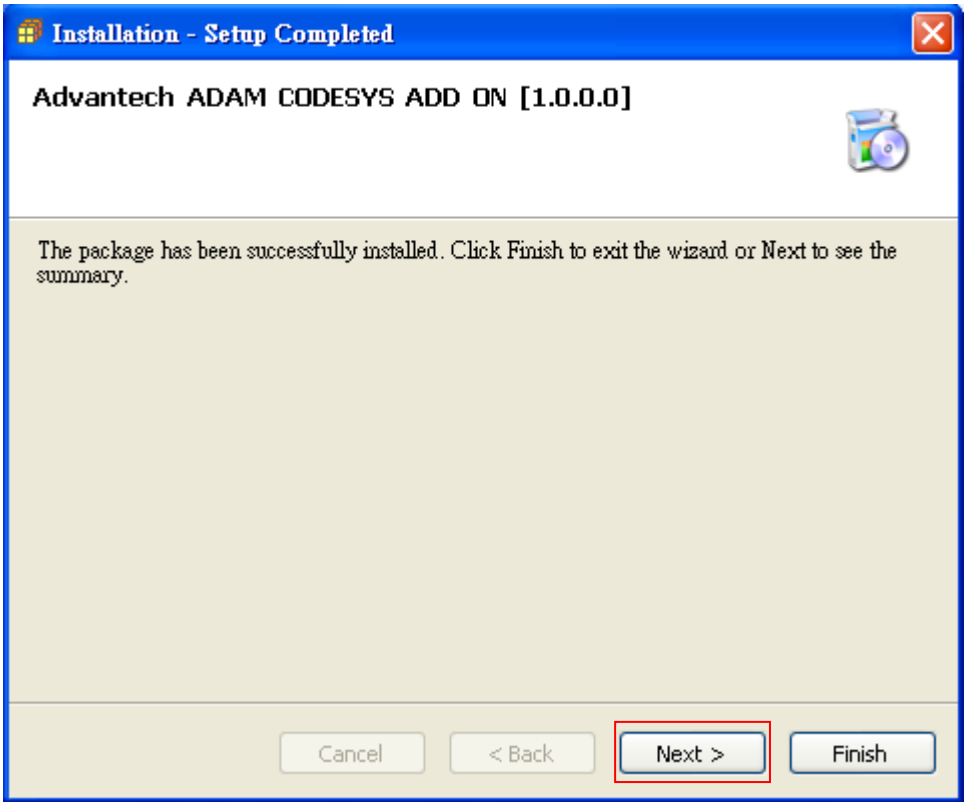
Step 2: When you are asked to choose setup type, choose the "**Typical setup**" and then click Next.



Step 3: Select all versions and then click next.



Step 4: After the files have installed, you will see the completion screen. Click "Finish" to close the installation wizard.

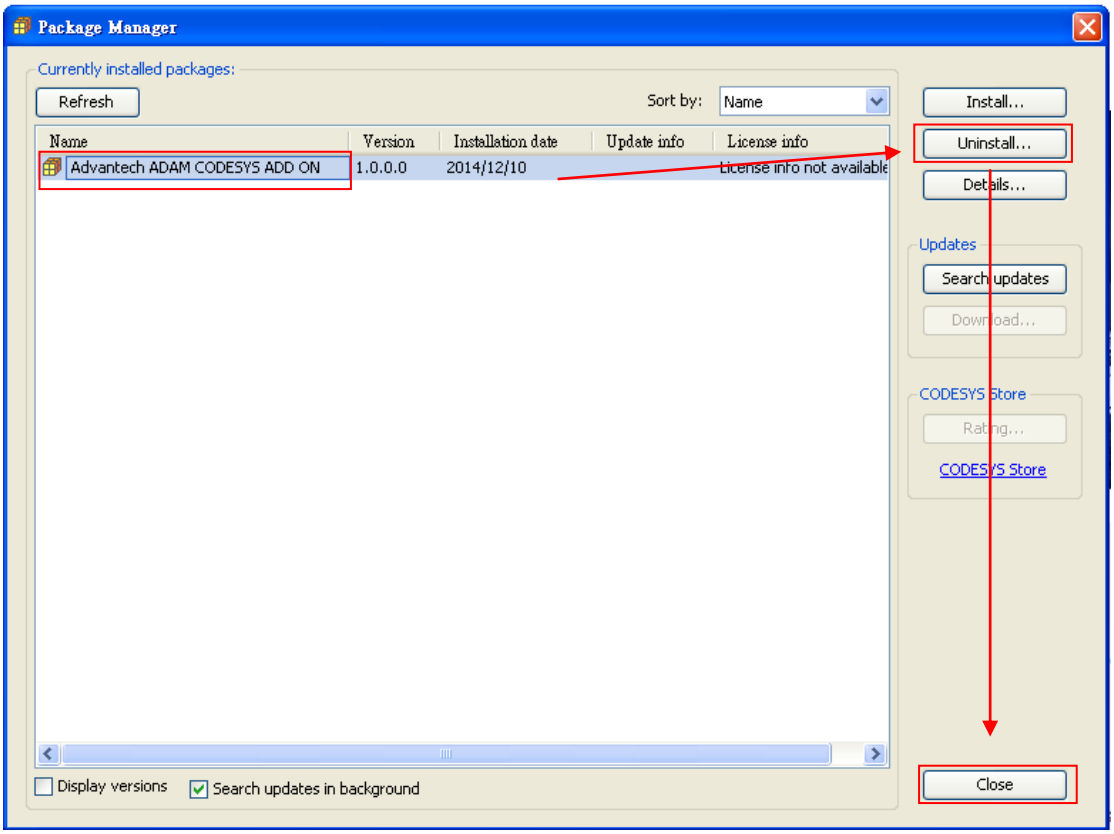


2.2.2. Updating the Package

It's highly recommended that you uninstall the previous version package before updating and installing new add-on package.

Start CoDeSys and perform command **Package Manager** from the menu (**Tools -> Package Manager**). Select the package you want to uninstall and then click “Uninstall”. Click "Close" to close the package manager.

After uninstalling the old package successfully, please refer to First-time Installation.


Package Manager
Currently installed packages:
Refresh
Sort by:
Name
Name
Version
Installation date
Update info
License info
Advantech ADAM CODESYS ADD ON
1.0.0.0
2014/12/10
License info not available
Install...
Uninstall...
Details...
Updates
Search updates
Download...
CODESYS Store
Rating...
CODESYS Store
Display versions
Search updates in background
Close

Chapter 3

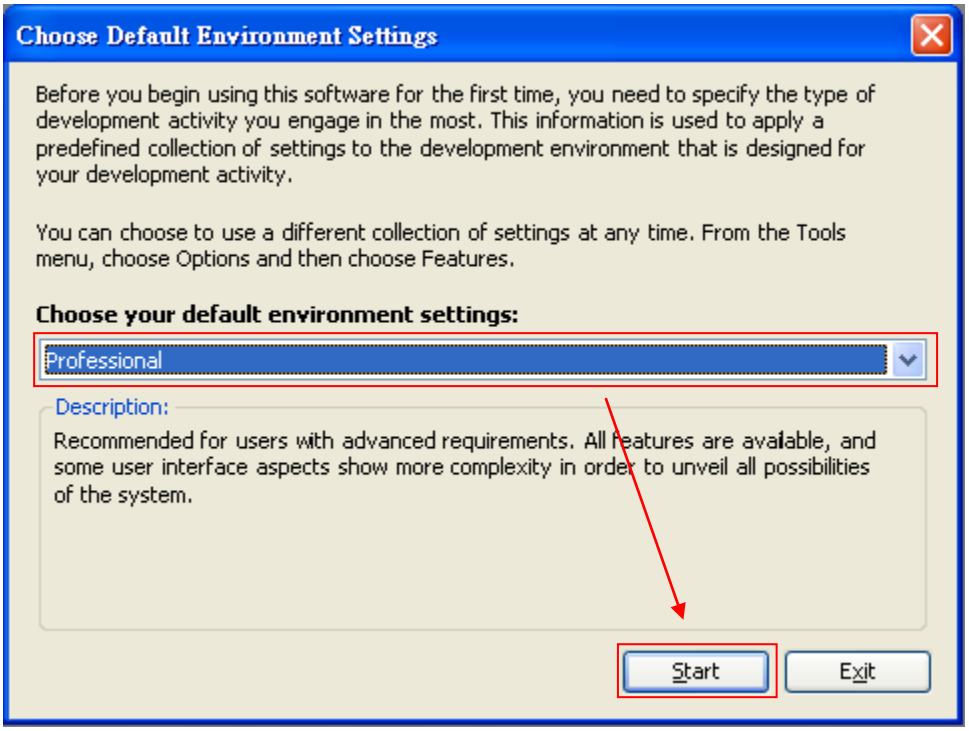
3. Create and run a project

3.1. Start CoDeSys

Start CoDeSys by double-clicking the CoDeSys icon  which is available on the desktop. Alternatively, you can start the CoDeSys programming system with

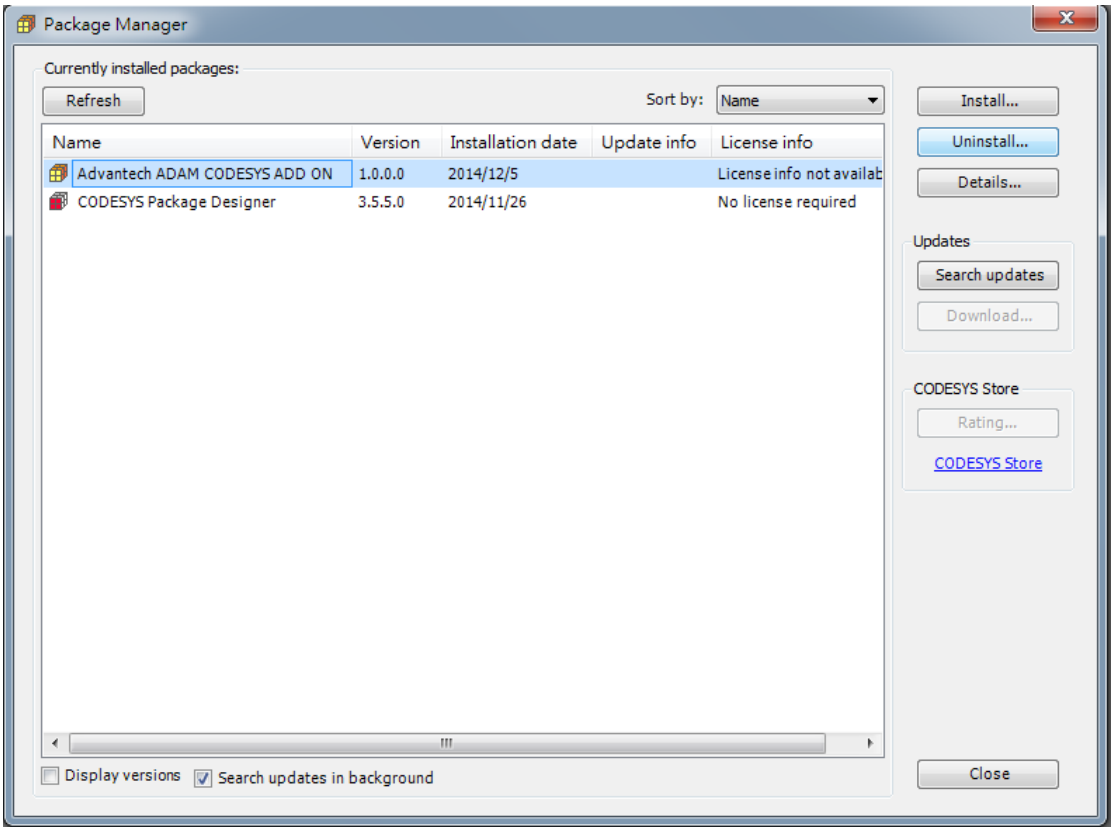
Start -> Programs -> 3S Software -> CoDeSys -> CODESYS V<version>

When you start the programming system the first time after first installation on the system, you will be asked to choose the default collection of settings and features. Choose the "**Professional**" and then click Start to proceed.



Before creating a project, make sure that Advantech ADAM add-on package is installed successfully. Choose **Package Manger** from the **Tools** menu:

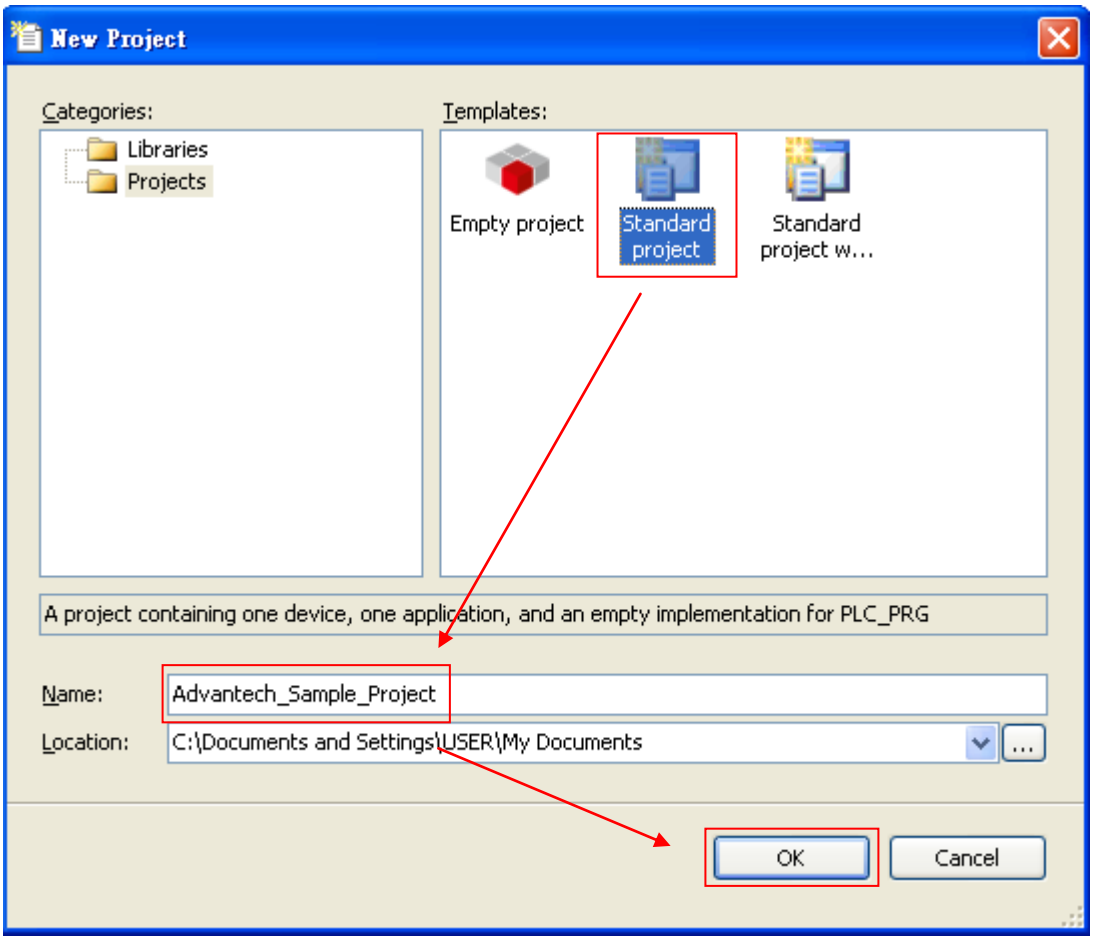
If the Advantech ADAM add-on package didn't show in manager, please refer to Chapter 2 and update your package.



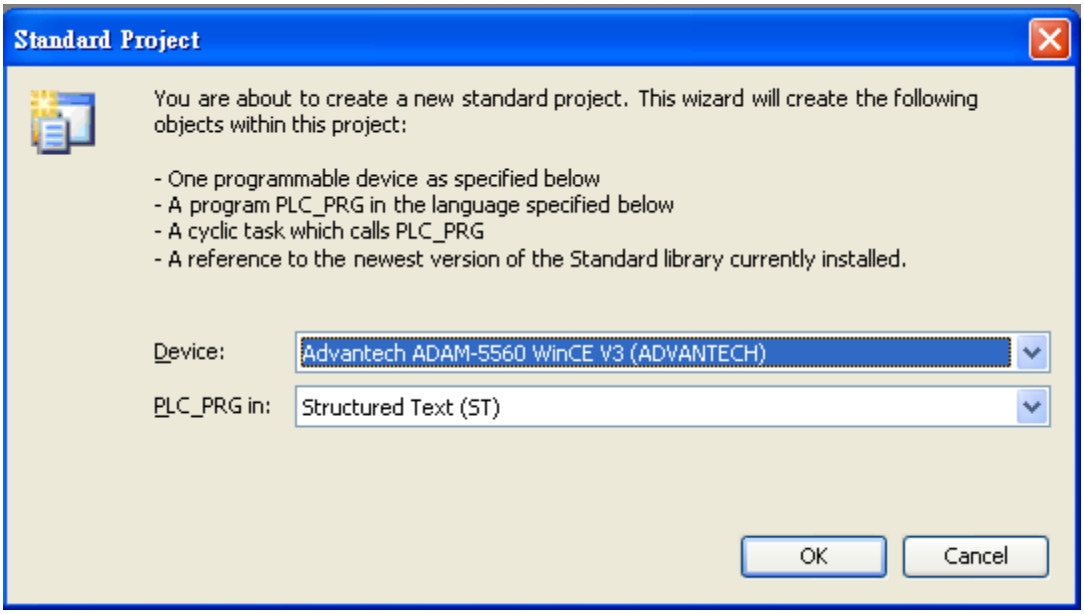
3.2. Create a Project

Step 1: To create a new project, choose command **New project** from the **File** menu:

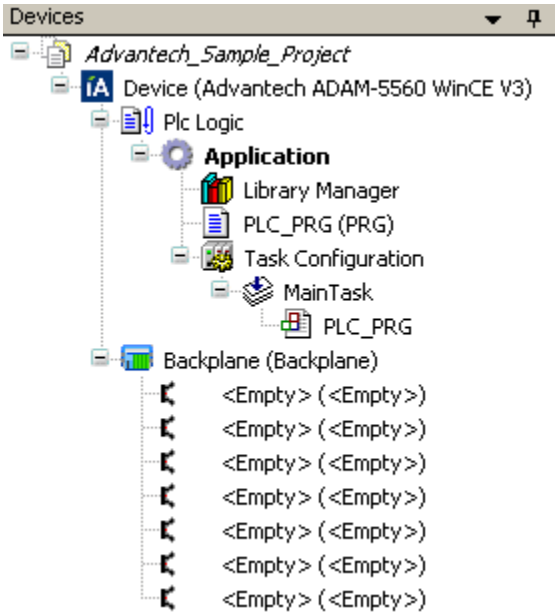
In the New Project dialog select **Standard project** in the 'Templates' field and enter a Name and a Location path for the project file. Press OK to confirm.



Step 2: You will then be prompted for choosing devices. Choose device **Advantech ADAM-5560 WinCE V3 (ADVANTECH)** and programming language **Structured Text (ST)** (depend on developer) for PLC_PRG. Press OK to open the new project.



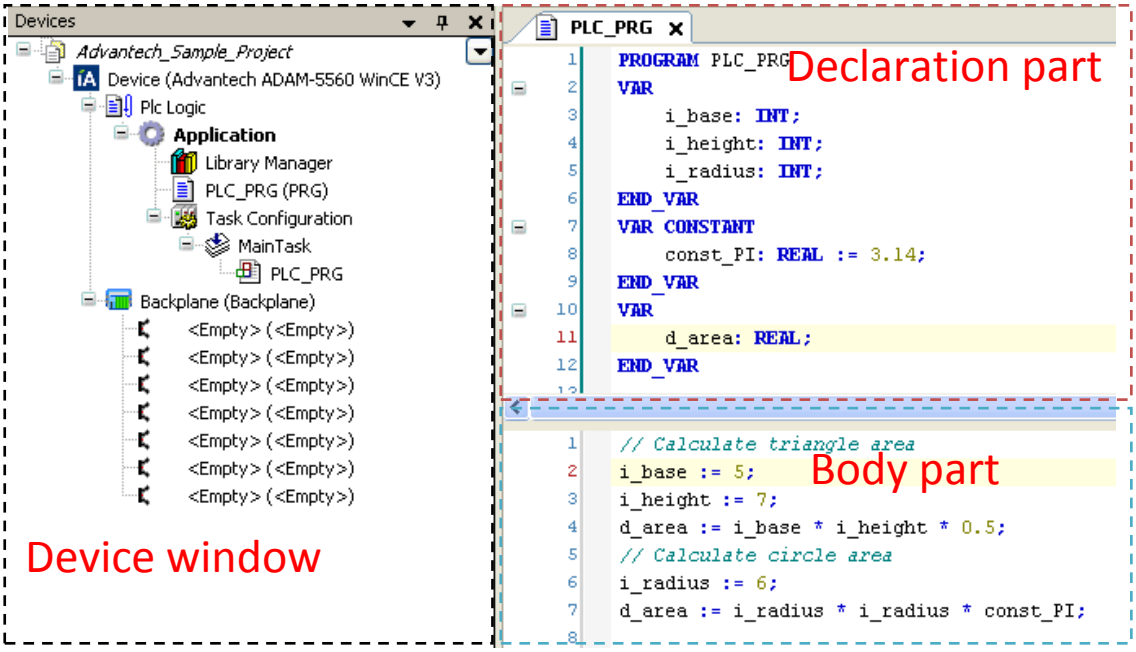
Step 3: The project name now will appear in the title bar of the CoDeSys user interface and the Devices window.



3.3. Write a Program

In the **Devices** window, double-chick **PLC_PRG(PRG)** and language editor window will open. The editor consists of a declaration part (upper) and a body part (lower), separated by a

screen divider. The declaration part shows line numbers at the left border and the embracing keywords "VAR" and "END_VAR" for the variables declaration.



In the declaration part of the editor put the cursor behind VAR and press the Return-key. A new empty line will be displayed where you enter the declaration of variables.

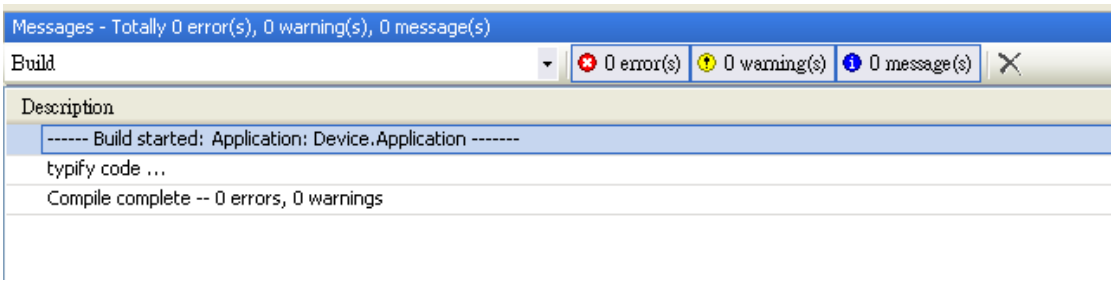
Here, we write a simple program to calculate the area of triangle and circle, so declare **i_base, i_height, i_radius** which are of type INTEGER, **d_area** of type REAL:

```
PROGRAM PLC_PRG
VAR
    i_base: INT;
    i_height: INT;
    i_radius: INT;
END_VAR
VAR CONSTANT
    const_PI: REAL := 3.14;
END_VAR
VAR
    d_area: REAL;
END_VAR
```

In the body part of the PLC_PRG editor put the cursor in line 1 and enter the following lines:

```
// Calculate triangle area
i_base := 5;
i_height := 7;
d_area := i_base * i_height * 0.5;
// Calculate circle area
i_radius := 6;
d_area := i_radius * i_radius * const_PI;
```

We need to check the program for syntactic errors and perform command **Build**  from the menu (**Build -> Build**) or press **<F11>**:



Note!

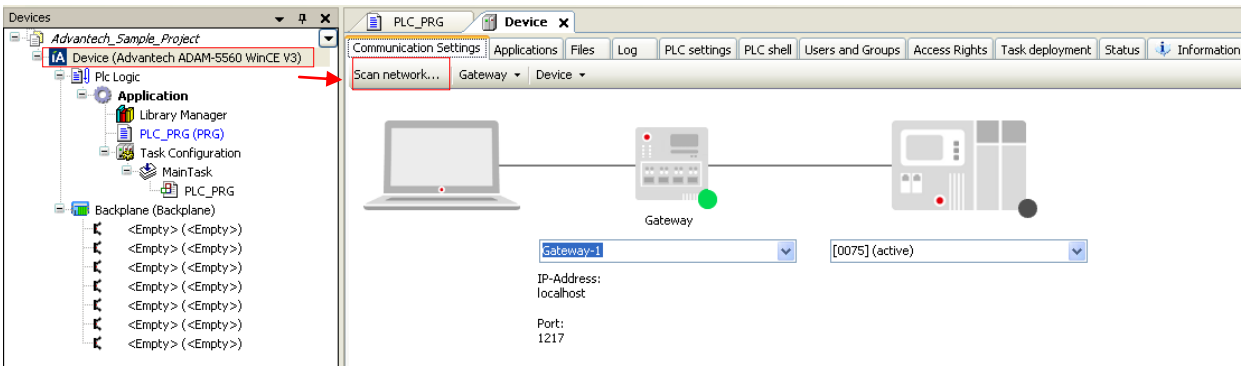
No code will be generated in this case. Error messages will be displayed in the Messages window which is placed at the lower part of the user interface per default.

3.4. Connect to the Target Device

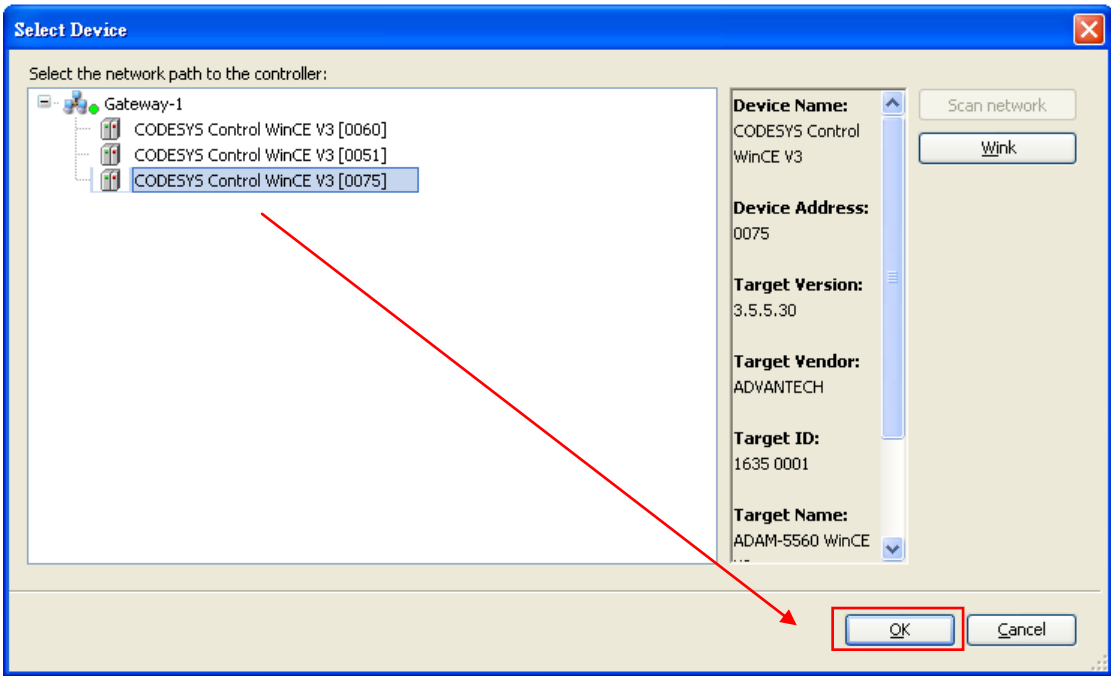
In this section, we want to discuss how to connect to ADAM-5560.

We need to set the active application by using Device editor. It displays an icon of programming device, the current gateway and the target device with their connection status.

The **Device editor** opens by double clicking the device name in the device tree.



Click the **Scan network** button to search for available devices in your local network. You will then be prompted for the device selection. Choose your target device and click OK to proceed.



In Device editor, it will show the connection status. Please check that the colored status points are all in green.

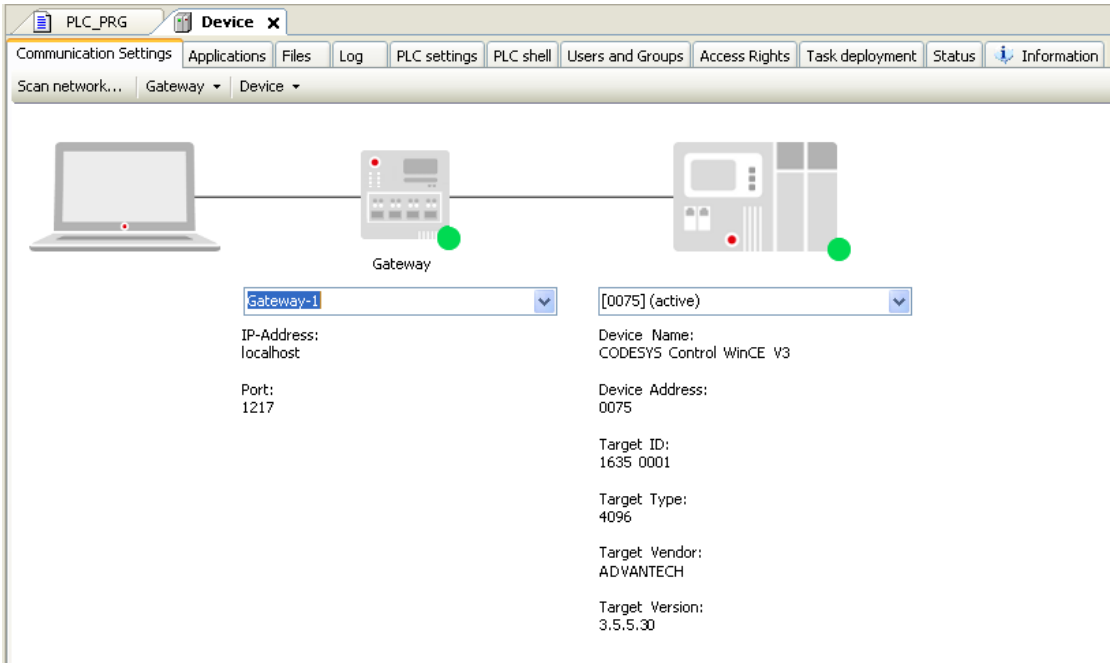
Note!

Meaning of the colored status point on the gateway and the device:

Red: Connection cannot be established

Green: Connection established

Black: Connection not defined



3.5. Run the Application

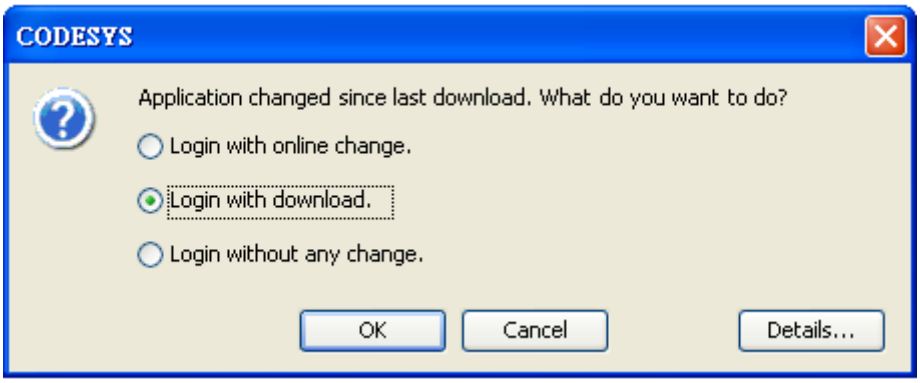
We can download the application by performing command **Login** from the menu (**Online -> Login**) or press **<Alt+F8>**. You will then be prompted for choosing login options. Here, we choose “Login with download” for the first time and click OK to proceed.

Note!

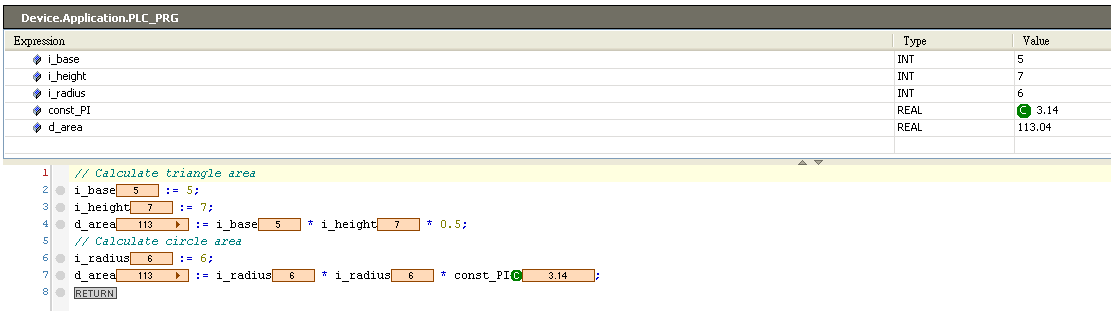
Meaning of login options:

“Login with online change”: Only the modified objects will be loaded.

“Login with download”: The complete application will be loaded and initialized completely. “Login without any change”: The latest modifications will not be loaded.



We run the program by performing command **Start** from the menu **(Debug -> Start)** or press **<F5>**. The online view of PLC_PRG will be opened: In the upper part a table shows the watch variables in application. In the lower part you see the code lines as entered in offline mode, supplemented by the little inline monitoring windows behind each variable, showing the actual value.



Stop the program by performing command **Stop** from the menu (**Debug -> Stop**) or press **<Shift+F8>**.

If you want to change into the offline mode and disconnect the programming system from the target device, perform command **Logout** from the menu (**Online -> Logout**) or press **<Ctrl+F8>**.

Chapter 4

4. Advantech I/O Modules

4.1. Insert I/O Modules into CODESYS

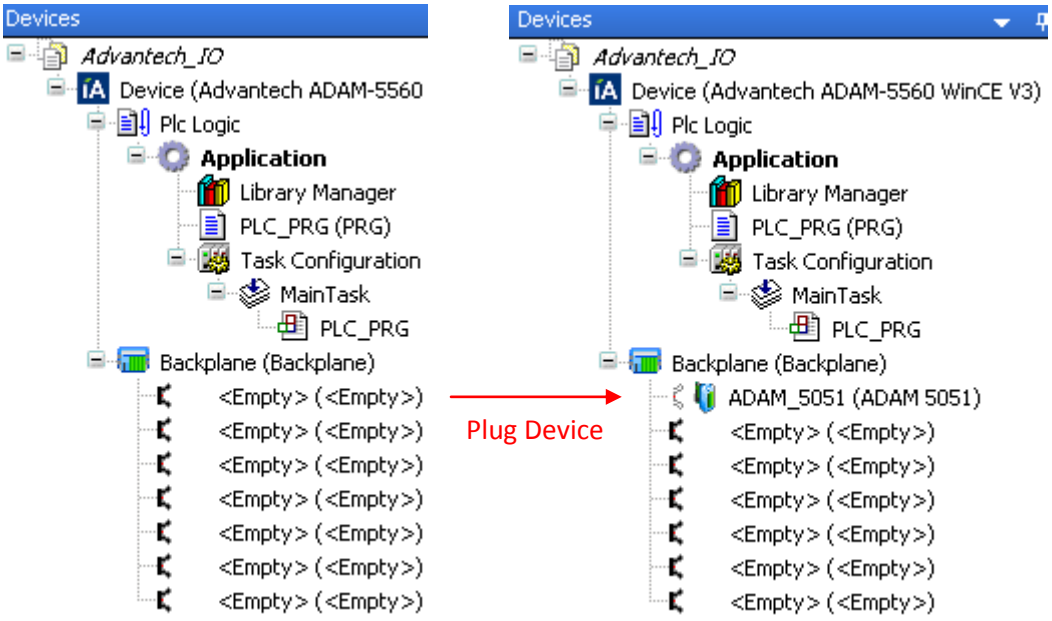
We can add and configure Advantech I/O modules as objects in the device tree.

Note!

ADAM-5560 does NOT support hot plugging for I/O modules.

An empty slot is identified by icon and entry <Empty> (<Empty>). An already occupied slot shows icon and the device name.

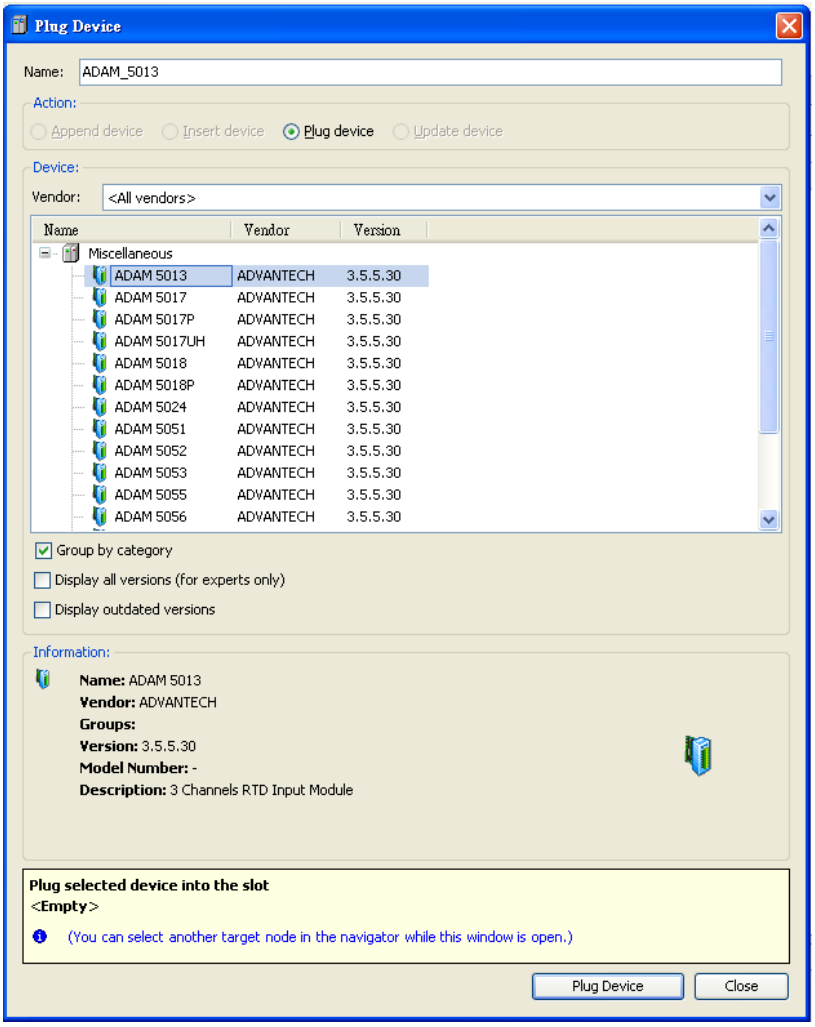
Choose one of available slot and click **Plug Device** in context menu.



It will open the **Plug Device dialog**, where you can choose one of available devices for the current slot. The existing entry will be replaced by the new one in an occupied slot.
Click **Plug Device** to proceed and then press **Close** to close the device dialog.

Note!

You can remove the existing device by click **Delete** in context menu.



4.2. Map Variables to I/O Modules

In this section, we want to discuss how to map variable of program to Advantech I/O modules. For more details on creating a new program please refer to Chapter 3.

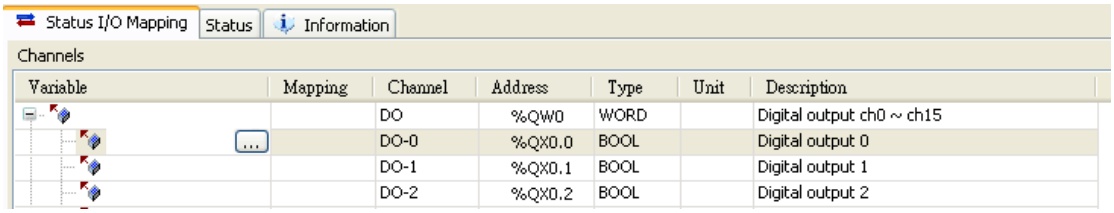
Here, we declare *bValue* in declaration part and set true in body part.

```
PROGRAM PLC_PRG
VAR
    bValue: BOOL;
END_VAR
```

```
bValue:= true;
```

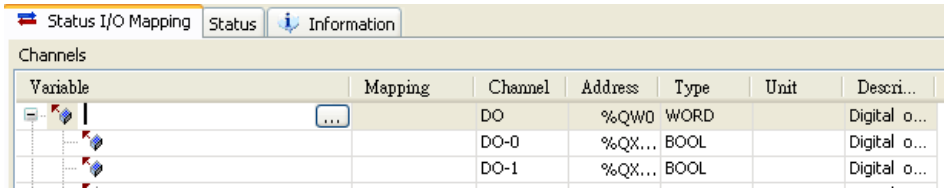
Open **Module Editor** by double clicking the device name in the device tree. Double-click on the variable column and choose mapping variable by clicking the button [...]. In this example,

we try to map the variable (*bValue*) to channel 0, so we double-click on the first row of variable column.

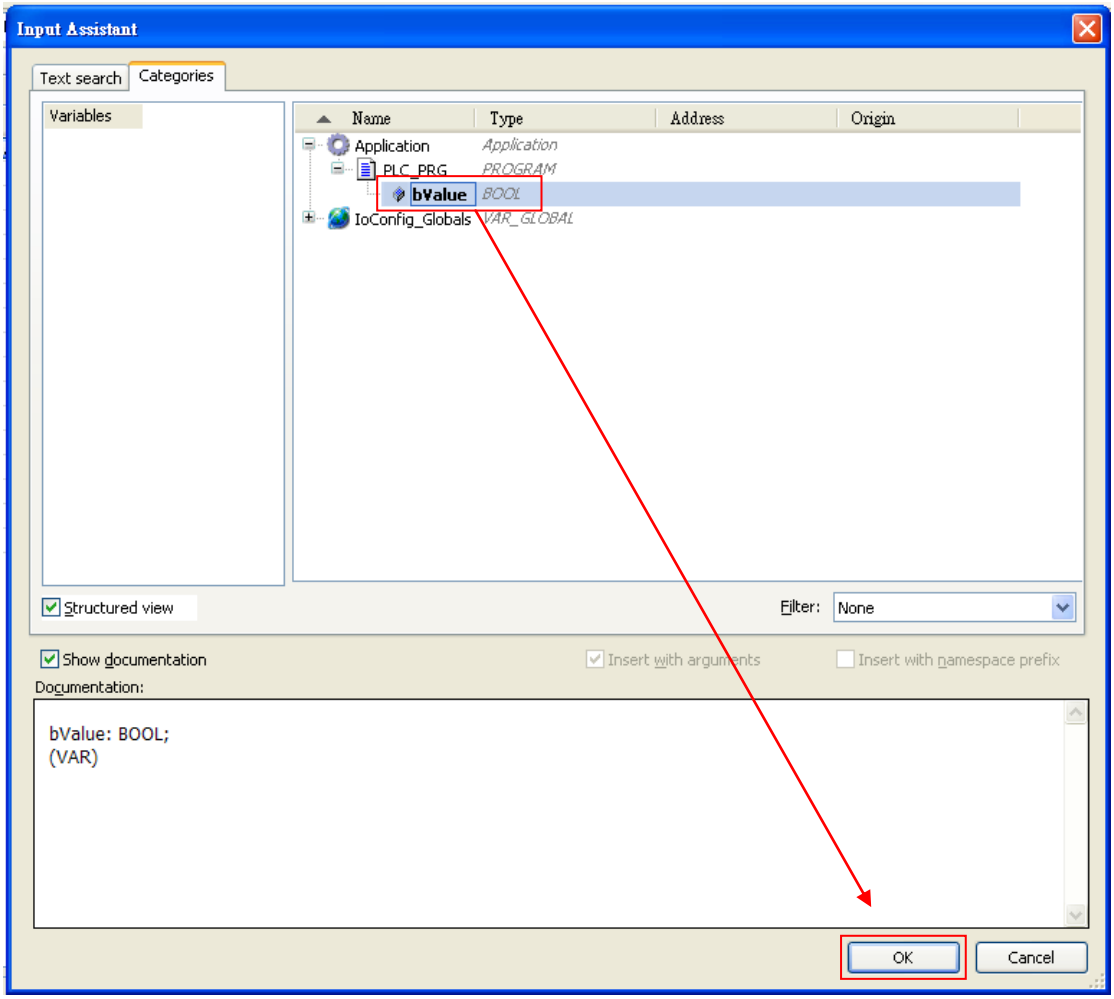


Note!

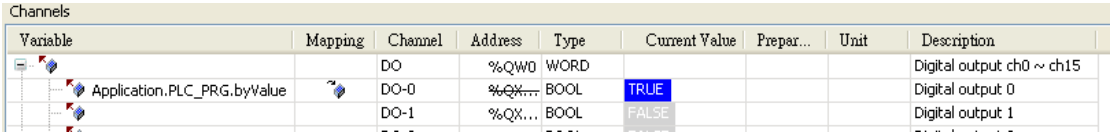
If you want to control all channels, declare a WORD variable and map it.



It will open the **Input Assistant Dialog**, where you can choose one of available devices for the current slot. The existing entry will be replaced by the new one in an occupied slot. Click **Plug Device** to proceed.

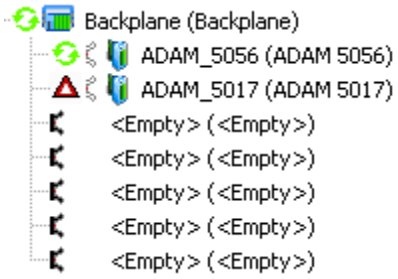


Now, we can download the application by performing command **Login** and then performing command **Start**. The channel-0 of I/O module will be lit up.



Note!

If the Advantech modules are correctly configured, it will show a green circle icon next to the device name in the device tree. If it shows a red triangle , see Chapter 6 for troubleshooting.



4.3. Support List

Advantech provides 21 types of ADAM-5000 I/O modules for various applications so far. Following table is the I/O modules support list. In the following section, we will introduce I/O modules according to their types.

Module	Name	Specification	Reference
Analog Input	ADAM-5013	3-ch RTD Input	Isolated
	ADAM-5017	8-ch AI	Isolated
	ADAM-5017P	8-ch AI w/Independent Input Range	Isolated
	ADAM-5017UH	8-ch Ultra High Speed AI (200KHz)	Isolated
	ADAM-5018	7-ch TC Input	Isolated
	ADAM-5018P	7-ch TC Input w/Independent Input Range	Isolated
Analog Output	ADAM-5024	4-ch AO	Isolated
Digital Input	ADAM-5051	16-ch DI	Non-isolated
	ADAM-5051D	16-ch DI w/LED	Non-isolated
	ADAM-5051S	16-ch Isolated DI w/LED	Isolated
	ADAM-5052	8-ch Isolated DI w/LED	Isolated
	ADAM-5053	32-ch Isolated DI	Isolated
Digital Output	ADAM-5056	16-ch DO	Non-isolated
	ADAM-5056D	16-ch DO w/LED	Non-isolated
	ADAM-5056S	16-ch Isolated DO w/LED	Isolated
	ADAM-5056SO	16-ch Source Type Isolated DO w/LED	Isolated
	ADAM-5057S	32-ch Isolated DO	Isolated
Digital I/O	ADAM-5055S	16-ch Isolated DI/O w/LED	Isolated
Relay Output	ADAM-5060	6-ch Relay Output	Isolated
	ADAM-5069	8-ch Power Relay Output w/LED	Isolated
Counter/	ADAM-5081	4-ch/8-ch High Speed Counter/Frequency	Isolated

4.4. Digital Input Modules

In this section, we are going to introduce digital input modules.

The **Module editor** opens by double clicking the device name in the device tree. It consists of three tab pages, that is, **Status I/O Mapping, Status and Information.**

Status I/O Mapping: Show the I/O mapping status between variable to module channel. It consists of seven columns.

Mapping: The mapping status of each variable.

Note!

There are two categories of variables: **Channel values** and **Error ID**.

Channel values: The data type of each channel is in single bit. If the value is “true”, it means that the channel is on; “false” for off. All channel values can represent as one word.

For detailed variable mapping information, see Chapter 4.2.

Error ID: This variable holds the status of I/O module and its data type is in Word (16 Bits). Get module error ID by mapping the last variable in table. For detailed error ID information, see Chapter 5.3.

Address: The starting physical address of the variables for this I/O group. The board shown below has 16 digital inputs. This will require either 16 Boolean addresses or 1 WORD (2 BYTE) addresses.

Note!

Meaning of address expression:

% = Directly Mapped variable

I = Physical Input

W = Word (16 bits)

X = Single bit

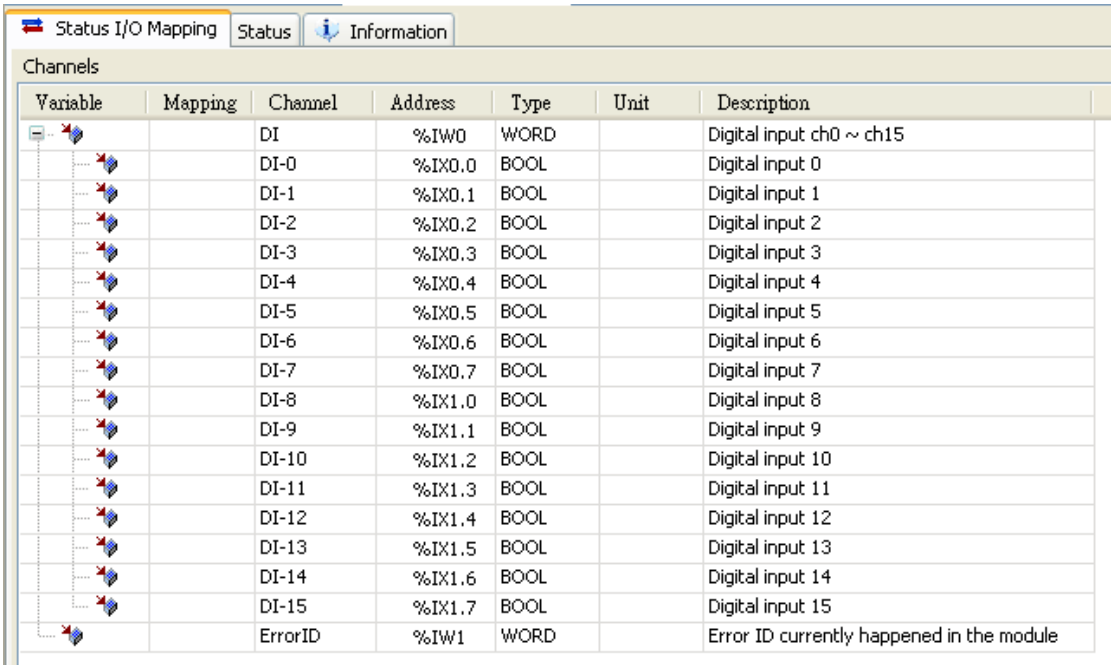
$(N1). $(N2) = The starting address. The first number means the starting byte; the second number means the starting bit.

Type: The data type of each variable.

Description: The description of each variable.

Status: The reserved page.

Information: Provide the brief information to current module.



Variable	Mapping	Channel	Address	Type	Unit	Description
		DI	%IW0	WORD		Digital input ch0 ~ ch15
		DI-0	%IX0.0	BOOL		Digital input 0
		DI-1	%IX0.1	BOOL		Digital input 1
		DI-2	%IX0.2	BOOL		Digital input 2
		DI-3	%IX0.3	BOOL		Digital input 3
		DI-4	%IX0.4	BOOL		Digital input 4
		DI-5	%IX0.5	BOOL		Digital input 5
		DI-6	%IX0.6	BOOL		Digital input 6
		DI-7	%IX0.7	BOOL		Digital input 7
		DI-8	%IX1.0	BOOL		Digital input 8
		DI-9	%IX1.1	BOOL		Digital input 9
		DI-10	%IX1.2	BOOL		Digital input 10
		DI-11	%IX1.3	BOOL		Digital input 11
		DI-12	%IX1.4	BOOL		Digital input 12
		DI-13	%IX1.5	BOOL		Digital input 13
		DI-14	%IX1.6	BOOL		Digital input 14
		DI-15	%IX1.7	BOOL		Digital input 15
		ErrorID	%IW1	WORD		Error ID currently happened in the module

4.5. Digital Output Modules

In this section, we are going to introduce digital output modules.

The **Module editor** opens by double clicking the device name in the device tree. It consists of three tab pages, that is, **Status I/O Mapping, Status and Information.**

Status I/O Mapping: Show the I/O mapping status between variable to module channel. It consists of seven columns.

Mapping: The mapping status of each variable.

Note!

There are two categories of variables: **Channel values** and **Error ID**.

Channel values: The data type of each channel is in single bit. Set the value to "true" for switching on the channel; "false" for switching off. All channel values can represent as one word.

For detailed variable mapping information, see Chapter 4.2.

Error ID: This variable holds the status of I/O module and its data type is in Word (16 Bits). Get module error ID by mapping the last variable in table. For detailed error ID information, see Chapter 5.3.

Address: The starting physical address of the variables for this I/O group. The board shown below has 16 digital inputs. This will require either 16 Boolean addresses or 1 WORD (2 Byte) addresses.

Note!

Meaning of address expression:

% = Directly Mapped variable

Q = Physical Output

W = Word (16 bits)

X = Single bit

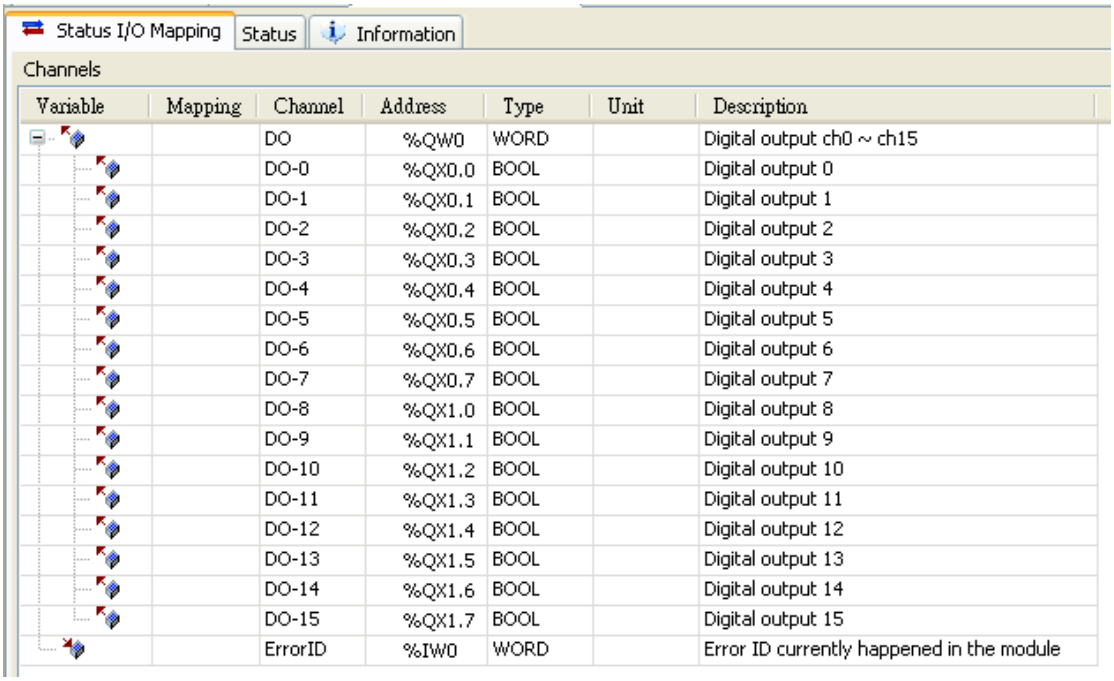
$(N1). $(N2) = The starting address. The first number means the starting byte; the second number means the starting bit.

Type: The data type of each variable.

Description: The description of each variable.

Status: The reserved page.

Information: Provide the brief information to current module.



Variable	Mapping	Channel	Address	Type	Unit	Description
		DO	%QW0	WORD		Digital output ch0 ~ ch15
		DO-0	%QX0.0	BOOL		Digital output 0
		DO-1	%QX0.1	BOOL		Digital output 1
		DO-2	%QX0.2	BOOL		Digital output 2
		DO-3	%QX0.3	BOOL		Digital output 3
		DO-4	%QX0.4	BOOL		Digital output 4
		DO-5	%QX0.5	BOOL		Digital output 5
		DO-6	%QX0.6	BOOL		Digital output 6
		DO-7	%QX0.7	BOOL		Digital output 7
		DO-8	%QX1.0	BOOL		Digital output 8
		DO-9	%QX1.1	BOOL		Digital output 9
		DO-10	%QX1.2	BOOL		Digital output 10
		DO-11	%QX1.3	BOOL		Digital output 11
		DO-12	%QX1.4	BOOL		Digital output 12
		DO-13	%QX1.5	BOOL		Digital output 13
		DO-14	%QX1.6	BOOL		Digital output 14
		DO-15	%QX1.7	BOOL		Digital output 15
		ErrorID	%IW0	WORD		Error ID currently happened in the module

4.6. Analog Input Modules

In this section, we are going to introduce analog input modules.

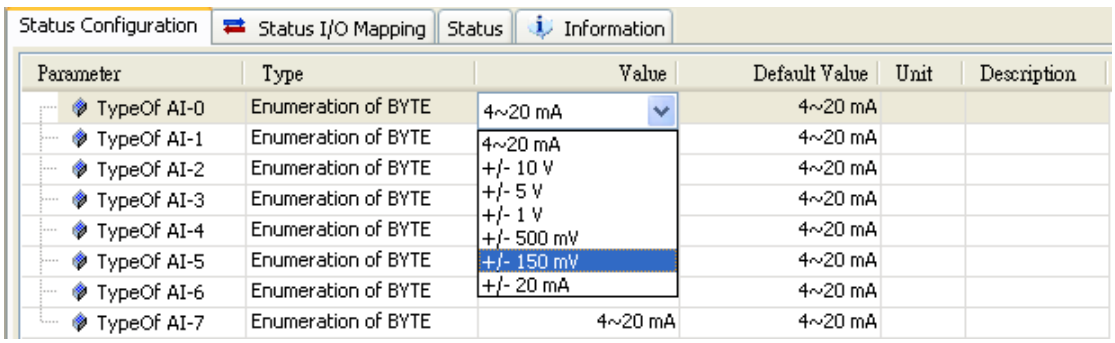
The **Module editor** opens by double clicking the device name in the device tree. It consists of four tab pages, that is, **Status Configuration**, **Status I/O Mapping, Status and Information.**

Status Configuration: Provide the channel status page for setting channel ranges.

Double-click on the value column of the particular channel.

Note!

For ADAM-5013, ADAM-5017 and ADAM-5018, all channels are restricted to the same channel range and base on the first channel (ch-0).



Status I/O Mapping: Show the I/O mapping status between variable to module channel.

Mapping: The mapping status of each variable.

Note!

There are two categories of variables: **Channel values** and **Error ID**.

Channel values: The data type of each channel is in REAL.

For detailed variable mapping information, see Chapter 4.2.

Error ID: This variable holds the status of I/O module and its data type is in Word (16 Bits). Get module error ID by mapping the last variable in table. For detailed error ID information, see Chapter 5.3.

Address: The starting physical address of the variables for this I/O group. The board shown below has 8 analog inputs. This will require 8 DWORD addresses.

Note!

Meaning of address expression:

% = Directly Mapped variable

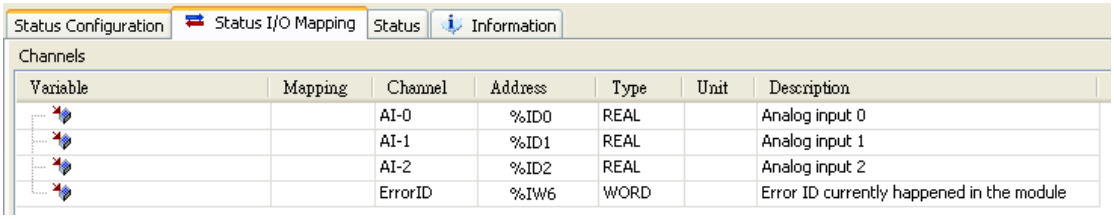
I = Physical Input

D = Double word (32 Bits)

$(N) = The starting address.

Type: The data type of each variable.

Description: The description of each variable.



Status: The reserved page.

Information: Provide the brief information for current module.

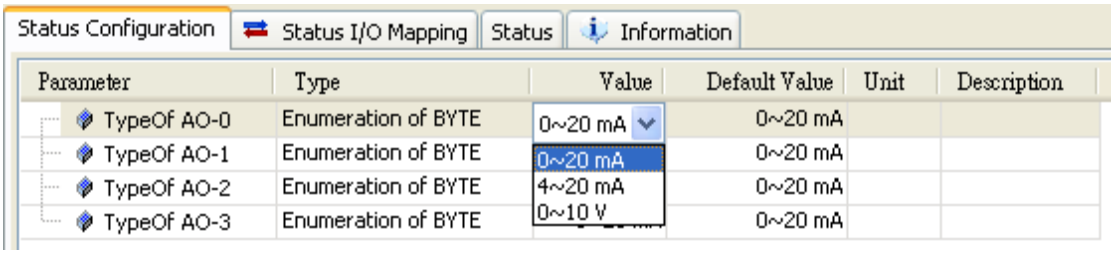
4.7. Analog Output Modules

In this section, we are going to introduce analog output modules.

The **Module editor** opens by double clicking the device name in the device tree. It consists of four tab pages, that is, **Status Configuration**, **Status I/O Mapping, Status and Information.**

Status Configuration: Provide the channel status page for setting channel ranges.

Double-click on the value column of the particular channel.



Status I/O Mapping: Show the I/O mapping status between local variable to module channel.

Mapping: The mapping status of each variable.

Note!

There are two categories of variables: **Channel values** and **Error ID**.

Channel values: The data type of each channel is in REAL.

For detailed variable mapping information, see chapter 4.2.

Error ID: This variable holds the status of I/O module and its data type is in Word (16 Bits). Get module error ID by mapping the last variable in table. For detailed error ID information, see chapter 5.3.

Address: The starting physical address of the variables for this I/O group. The board shown below has 4 analog outputs. This will require 4 DWORD addresses.

Note!

Meaning of address expression:

% = Directly Mapped variable

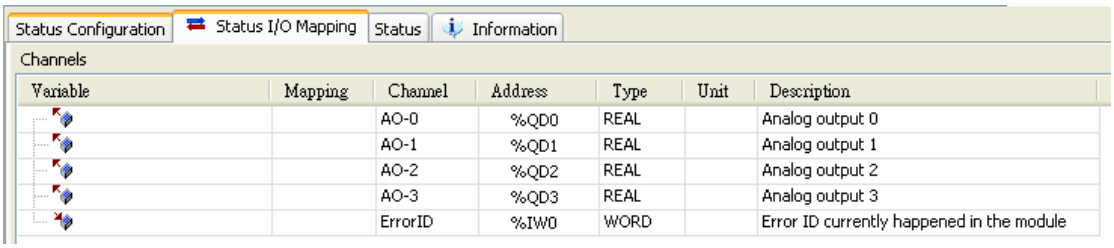
Q = Physical Output

D = Double word (32 Bits)

$(N) = The starting address.

Type: The data type of each variable.

Description: The description of each variable.



Variable	Mapping	Channel	Address	Type	Unit	Description
		AO-0	%QD0	REAL		Analog output 0
		AO-1	%QD1	REAL		Analog output 1
		AO-2	%QD2	REAL		Analog output 2
		AO-3	%QD3	REAL		Analog output 3
		ErrorID	%IW0	WORD		Error ID currently happened in the module

Status: The reserved page.

Information: Provide the brief information for current module.

4.8. Relay Output Modules

In this section, we are going to introduce relay output modules.

The **Module editor** opens by double clicking the device name in the device tree. It consists of three tab pages, that is, **Status I/O Mapping, Status and Information.**

Status I/O Mapping: Show the I/O mapping status between local variable to module channel. It consists of seven columns.

Mapping: The mapping status of each variable.

Note!

There are two categories of variables: **Channel values** and **Error ID**.

Channel values: The data type of each channel is in single bit. Set the value to "true" for switching on the channel; "false" for switching off. All channel values can represent as one word.

For detailed variable mapping information, see Chapter 4.2.

Error ID: This variable holds the status of I/O module and its data type is in Word (16 Bits). Get module error ID by mapping the last variable in table. For detailed error ID information, see Chapter 5.3.

Address: The starting physical address of the variables for this I/O group. The board shown below has 6 relay outputs. This will require either 6 Boolean addresses or 1 Byte address.

Note!

Meaning of address expression:

% = Directly Mapped variable

Q = Physical Output

W = Word (16 bits)

X = Single bit

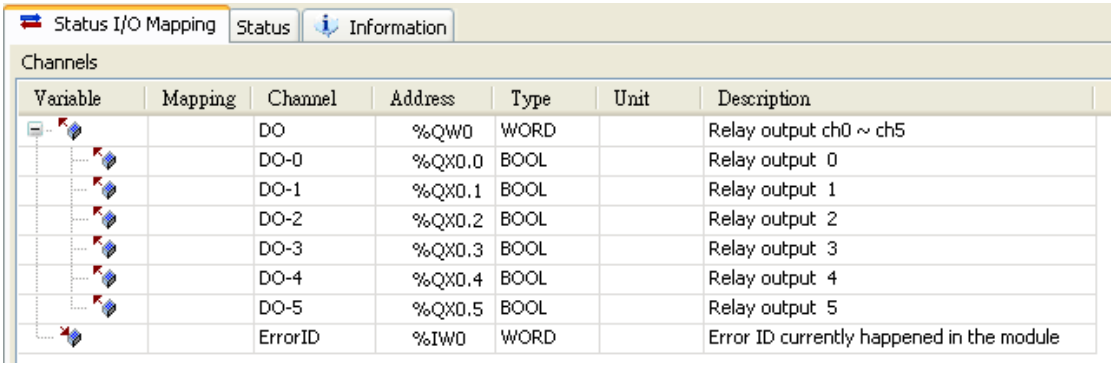
$(N1). $(N2) = The starting address. The first number means the starting byte; the second number means the starting bit.

Type: The data type of each variable.

Description: The description of each variable.

Status: The reserved page.

Information: Provide the brief information to current module.



Variable	Mapping	Channel	Address	Type	Unit	Description
		DO	%QW0	WORD		Relay output ch0 ~ ch5
		DO-0	%QX0.0	BOOL		Relay output 0
		DO-1	%QX0.1	BOOL		Relay output 1
		DO-2	%QX0.2	BOOL		Relay output 2
		DO-3	%QX0.3	BOOL		Relay output 3
		DO-4	%QX0.4	BOOL		Relay output 4
		DO-5	%QX0.5	BOOL		Relay output 5
		ErrorID	%IW0	WORD		Error ID currently happened in the module

4.9. Counter/Frequency Modules

In this section, we are going to introduce counter/frequency modules.

The **Module editor** opens by double clicking the device name in the device tree. It consists of four tab pages, that is, **Status Configuration**, **Status I/O Mapping, Status and Information.**

Status Configuration: Provide the channel status page for setting channel ranges. Double-click on the value column of the particular channel. The module contains counter channels and DO/Alarm channels as shown image below.

For counter channels, we offer five types of counting mode (Bi-direction, Up/Down, A/B Phase) for different application purposes. The counter will count up or down according to your applications. This counting function helps us obtain the most accurate data.

Before starting counting, we have to set **Startup Value**, i.e. Initial value**,** and the default is 0. The data type of startup value is in DWORD.

Parameter	Type	Value	Default Value	Unit	Description
CTR 0					Channel config of CTR 0
Mode	Enumeration of BYTE	Bi-directory	Bi-directory		
Startup Value	DWORD(16#0..16#FFFFFFFF)		0		
CTR 2					Channel config of CTR 2
Mode	Enumeration of BYTE		Bi-directory		
Startup Value	DWORD(16#0..16#FFFFFFFF)		0		
CTR 4					Channel config of CTR 4
Mode	Enumeration of BYTE	Bi-directory	Bi-directory		
Startup Value	DWORD(16#0..16#FFFFFFFF)	0	0		
CTR 6					Channel config of CTR 6
Mode	Enumeration of BYTE	Bi-directory	Bi-directory		
Startup Value	DWORD(16#0..16#FFFFFFFF)	0	0		

Bi-directory
Up/Down
A/B-1X
A/B-2X
A/B-4X

We can also set **Digital Filter** (in us) for high or low level minimum signal width to reduce noise spike.

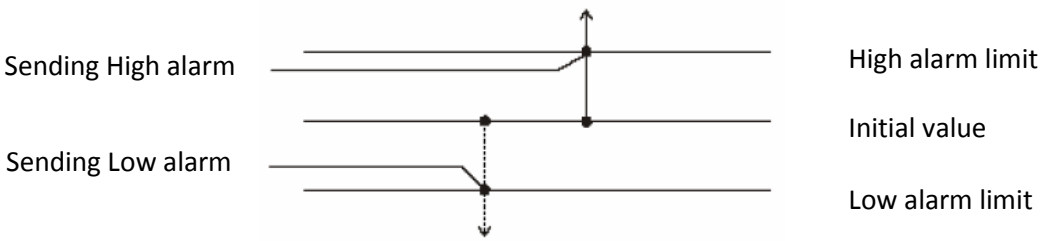
Digital Filter	UINT(1..65000)	1	1	us	Digital filter

For DO/Alarm channels, you can change the setting by double-clicking and selecting an item from the pull-down menus:

Mode: Select "DO" or "Local Alarm Latch"

Alarm Type: Select "High" or "Low", i.e. high alarm or low alarm.

Alarm Limit: Set alarm limit value.



Alarm Mapping: Select counter channel number.

ALM-1					Alarm output...
Mode	Enumeration of BYTE	DO	DO		
Alarm Type	Enumeration of BYTE	Low	Low		
Alarm Mapping	Enumeration of BYTE	Channel 0	Channel 0		
Alarm Limit	DWORD	0	0		
ALM-2					Alarm output...
Mode	Enumeration of BYTE	DO	DO		
Alarm Type	Enumeration of BYTE	Low	Low		
Alarm Mapping	Enumeration of BYTE	Channel 0	Channel 0		
Alarm Limit	DWORD	0	0		
ALM-3					Alarm output...
Mode	Enumeration of BYTE	DO	DO		
Alarm Type	Enumeration of BYTE	Low	Low		
Alarm Mapping	Enumeration of BYTE	Channel 0	Channel 0		
Alarm Limit	DWORD	0	0		

Status I/O Mapping: Show the I/O mapping status between local variable to module channel.

Mapping: The mapping status of each variable. For detailed variable mapping information, see chapter 4.2.

Note!

There are three categories of variables: **Status variables**, **Setting variables** and **Error ID**.

Status variables: These variables are read-only and hold the module status.

Name	Data type	Information
CTR Counting	Single Bit	“true” for starting counting; “false” for stop.
CTR Overflow	Single Bit	“true” for overflow

ALM Status	Single Bit	“true” for reach alarm latch.

Setting variables: These variables are read-write. You can change the setting by changing the column value.

Name	**Data type**	**Information**
DO value	Single Bit	Set the value to “true” for switching on the channel; “false” for switching off.
CTR Enable	Single Bit	Set “true” for starting to count; “false” for stopping to count.
CTR Reset	Single Bit	Set “true” for set current counting number to startup counting number.
CTR Clear Overflow	Single Bit	Set “true” for clear overflow.
ALM Clear Alarm	Single Bit	Set “true” for clear the alarm latch.

For detailed variable mapping information, see chapter 4.2.

Error ID: This variable holds the status of I/O module and its data type is in Word (16 Bits). Get module error ID by mapping the last variable in table. For detailed error ID information, see chapter 5.3.

Address: The starting physical address of the variables for this I/O group. The board shown below has 4 counter inputs and 4 digital outputs. This will require 4 DWORD addresses for counter channel values and 4 Boolean addresses for digital outputs.

Note!

Meaning of address expression:

% = Directly Mapped variable

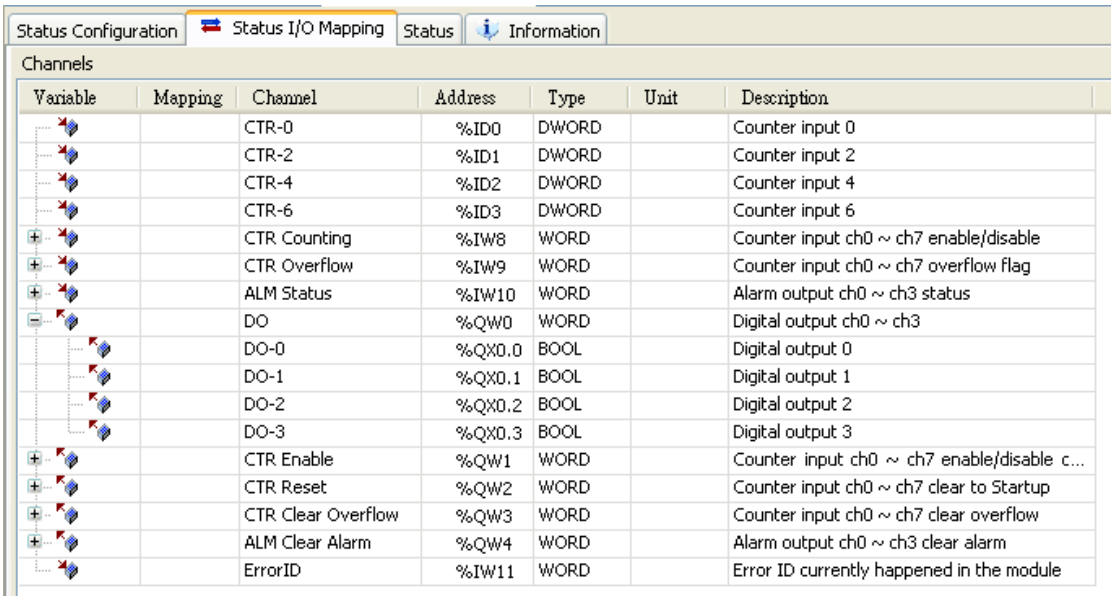
I = Physical Input **Q** = Physical Output

X = Single bit **D** = Double word (32 Bits)

$(N) = The starting address.

Type: The data type of each variable.

Description: The description of each variable.



Variable	Mapping	Channel	Address	Type	Unit	Description
		CTR-0	%ID0	DWORD		Counter input 0
		CTR-2	%ID1	DWORD		Counter input 2
		CTR-4	%ID2	DWORD		Counter input 4
		CTR-6	%ID3	DWORD		Counter input 6
⊞		CTR Counting	%IW8	WORD		Counter input ch0 ~ ch7 enable/disable
⊞		CTR Overflow	%IW9	WORD		Counter input ch0 ~ ch7 overflow flag
⊞		ALM Status	%IW10	WORD		Alarm output ch0 ~ ch3 status
⊟		DO	%QW0	WORD		Digital output ch0 ~ ch3
		DO-0	%QX0.0	BOOL		Digital output 0
		DO-1	%QX0.1	BOOL		Digital output 1
		DO-2	%QX0.2	BOOL		Digital output 2
		DO-3	%QX0.3	BOOL		Digital output 3
⊞		CTR Enable	%QW1	WORD		Counter input ch0 ~ ch7 enable/disable c...
⊞		CTR Reset	%QW2	WORD		Counter input ch0 ~ ch7 clear to Startup
⊞		CTR Clear Overflow	%QW3	WORD		Counter input ch0 ~ ch7 clear overflow
⊞		ALM Clear Alarm	%QW4	WORD		Alarm output ch0 ~ ch3 clear alarm
		ErrorID	%IW11	WORD		Error ID currently happened in the module

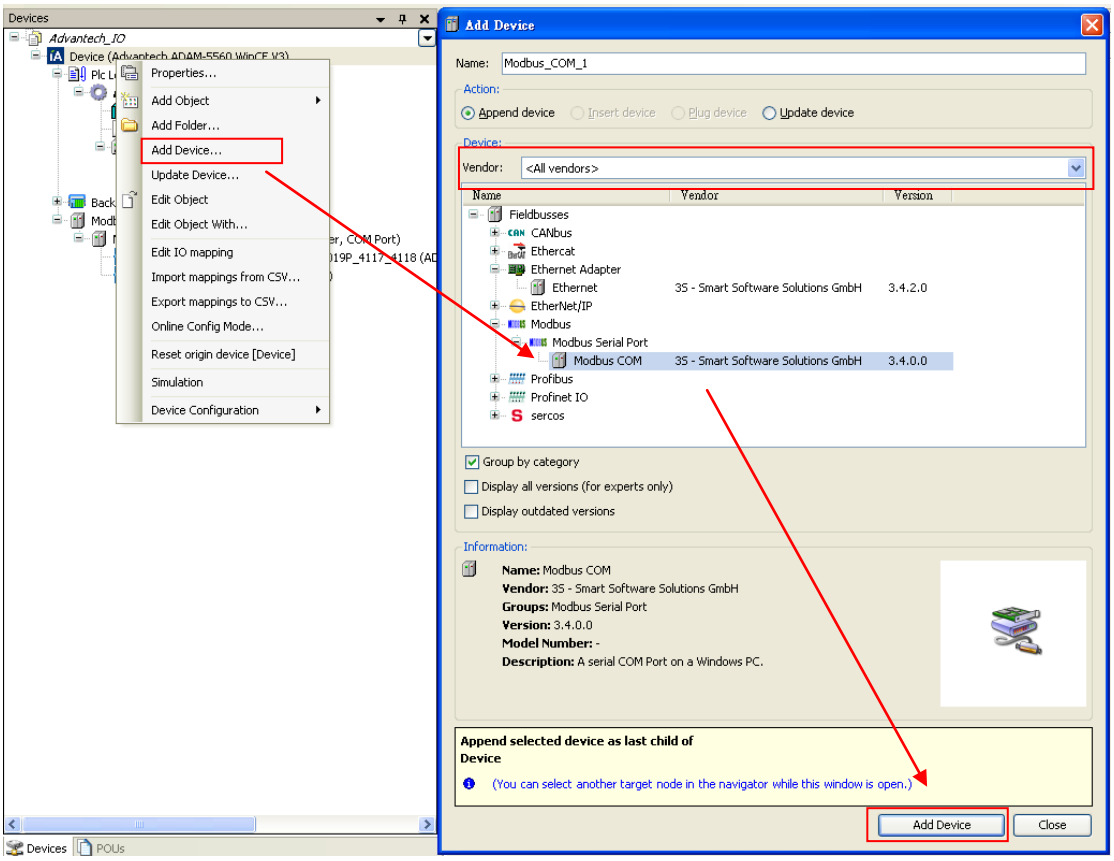
Chapter 5

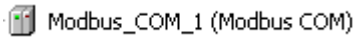
5. Advantech Fieldbus Modules

Advantech ADAM series distributed data acquisition and control systems are the ideal tools for creating multi-drop Fieldbus networks. The module design allows users to create application-specific configurations with ease. System communicates with their controlling host using either serial COM port or Modbus communication protocols.

5.1. Modbus

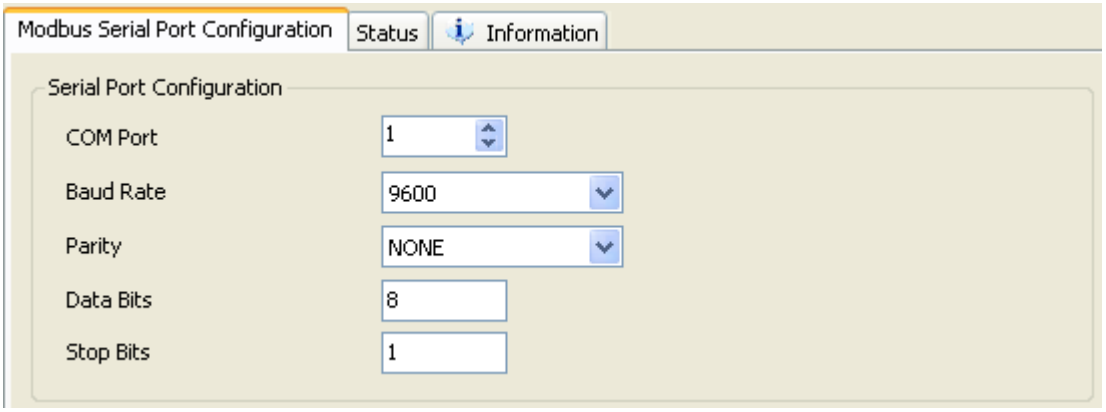
In Device Window, Right-click on **Device** and click **Add Device.** You will then be prompted for the Add Device dialog. Type the device name in Name container. Choose **Modbus COM** in the **Modbus** option and click **Add Device** to proceed and then press Close to close the device dialog.



Now, you'll see Modbus COM 

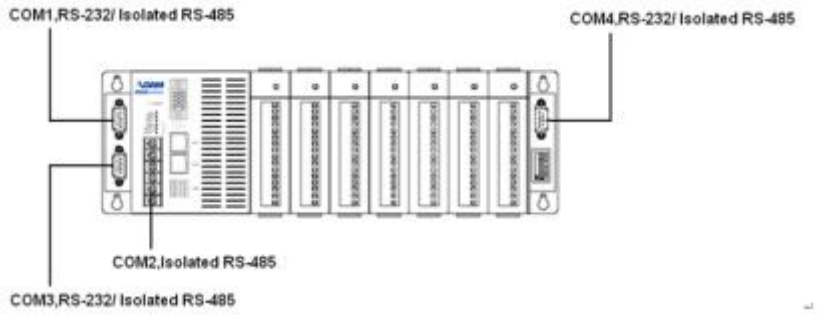
in the device tree.

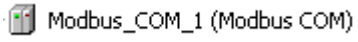
Double-click the COM port icon to set configuration.



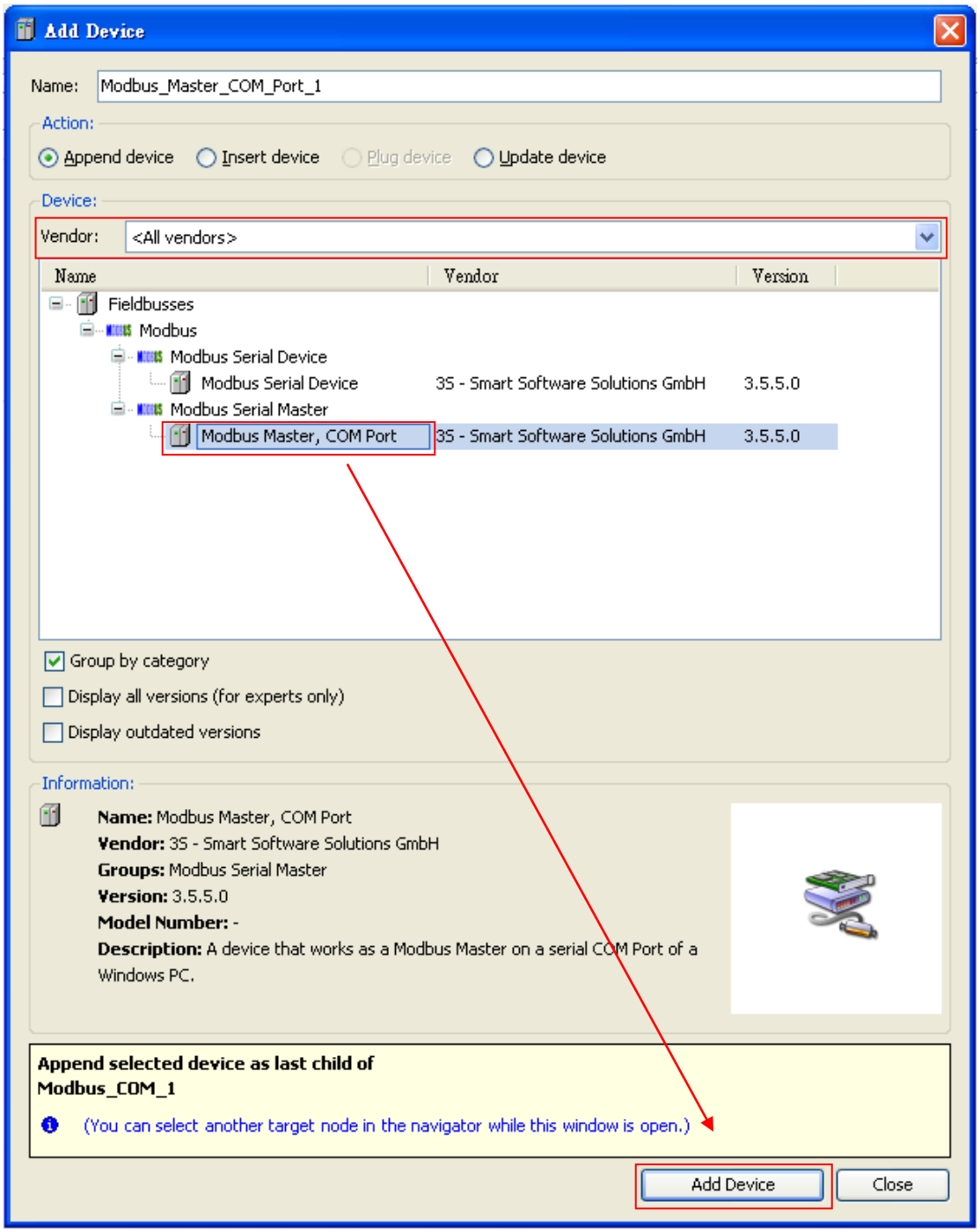
Note!

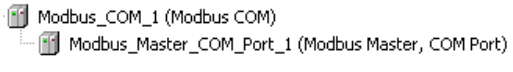
The location of COM ports is shown below. Follow the figure to set COM Port index.



Right click on Modbus COM 

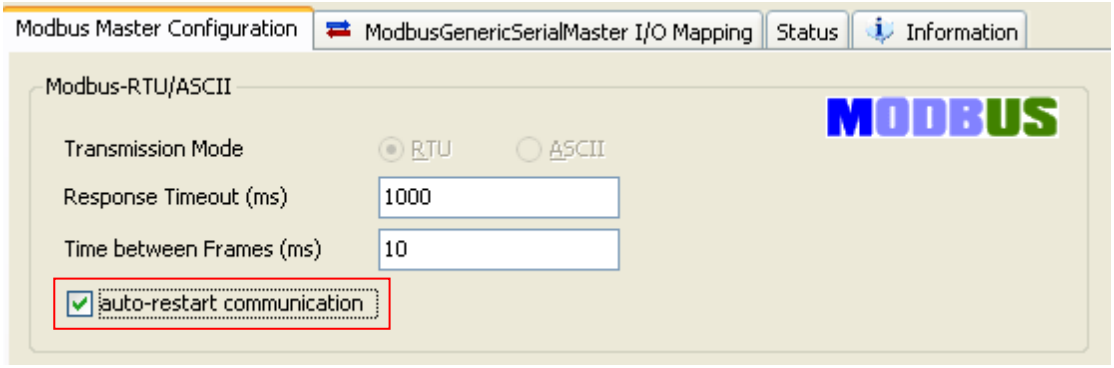
in the device tree and click **Add Device.** Type the device name in Name container. Choose **Modbus Master, COM Port** in the **Modbus** option. Click **Add Device** to proceed and then press **Close** to close the device dialog.



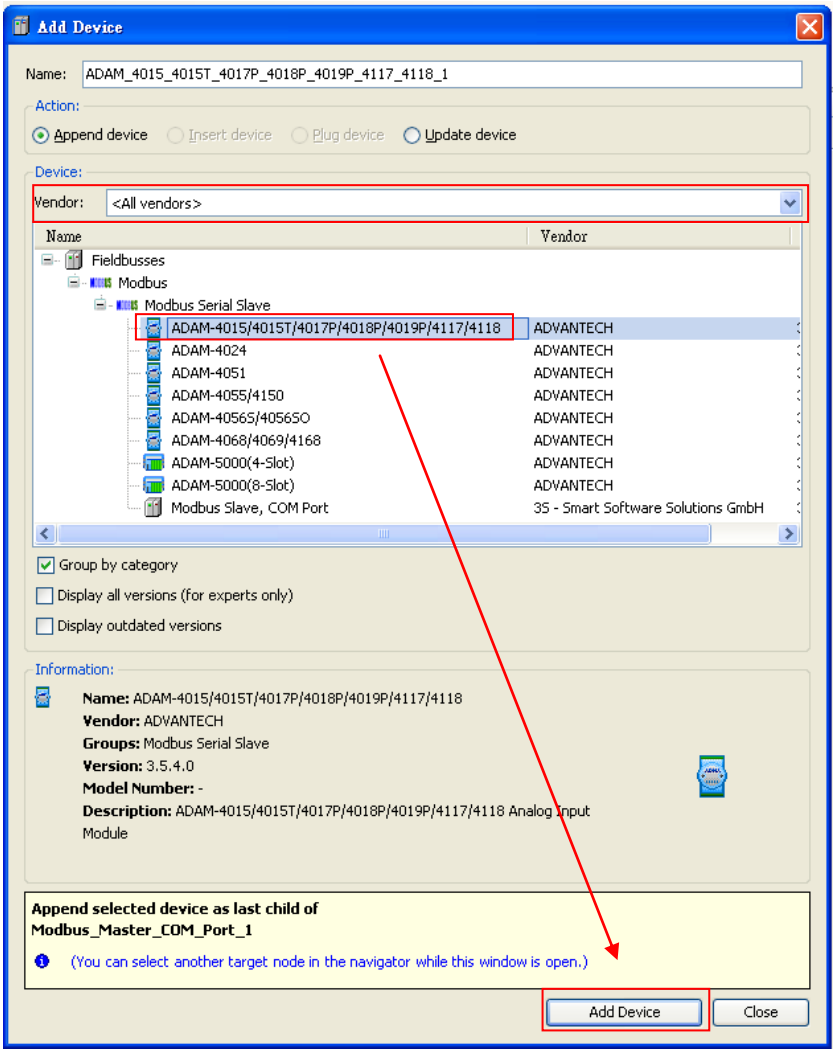
Now, you'll see Modbus master 

in the device tree. Double-click COM port to set master configuration. For further convenience,

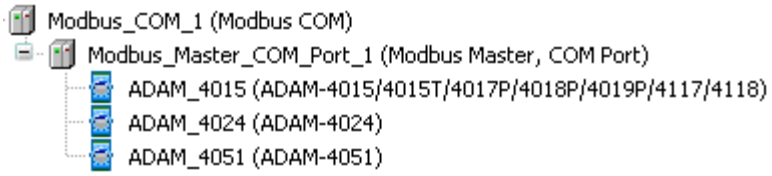
recommend you to check **auto-restart communication**.



Right click on Modbus master in the device tree and click **Add Device** in context menu. It will open the **Add Device dialog**, where you choose one available device for the current connection. Type the device name in Name container. Click **Add Device** to proceed and then press **Close** to close the device dialog.



Now, you will see Advantech ADAM modules on device list.



5.1.1. Modbus RTU Client

Open the **Modbus Device editor** by double-clicking on the device icon in the device tree. The editor is subdivided in the following tabs and the details will be described below:

Modbus Slave Configuration:

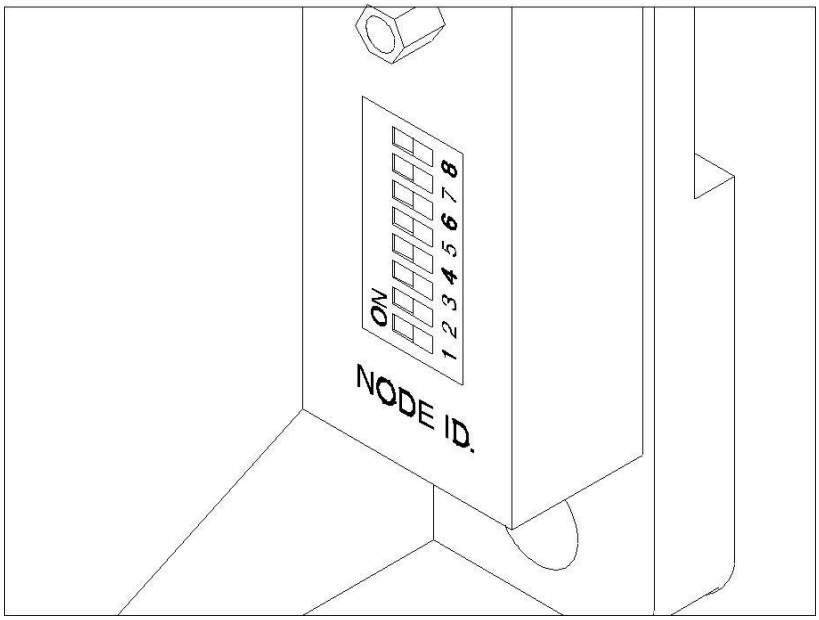
The following parameters deal with Modbus slave settings:

Slave Address: This number may range between 1 and 247; it serves to identify the node within the network.

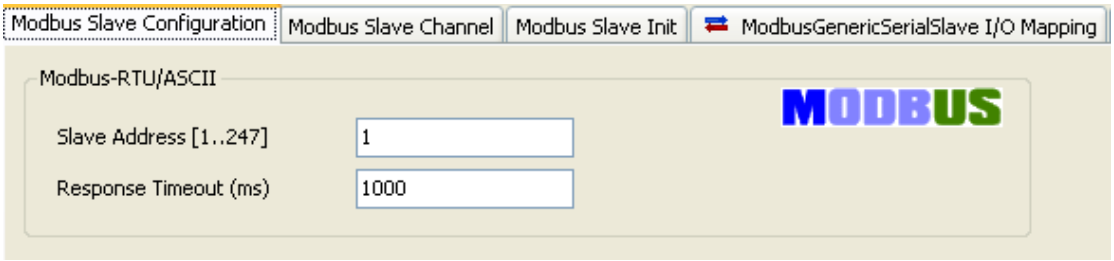
Note!

For ADAM-4000 series, move hardware switch to Initial mode and use **Advantech Adam/Apax Utility** to set slave address.

For ADAM-5000 series, use the 8-pin DIP switch to set slave address. Valid settings range from 0 to 127 where ON in any of the 8 DIP switch positions equates to a binary 1, and OFF equates to a binary 0. For initial setting, set address as 0 and baud rate setting will be fixed to 9600 bps. It is recommended to setting the range from 1 to 127.



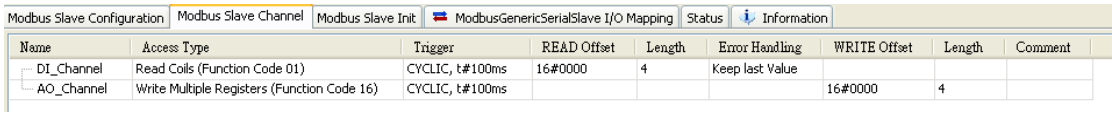
Response Timeout: The time span may be adapted individually to the slave and will override the Response Timeout set for the related master.



Modbus Slave Channel

This page is used to map variables from a slave's Modbus register into CODESYS.

You can revise the default value by double-clicking the table. The following parameters deal with slave channel settings:



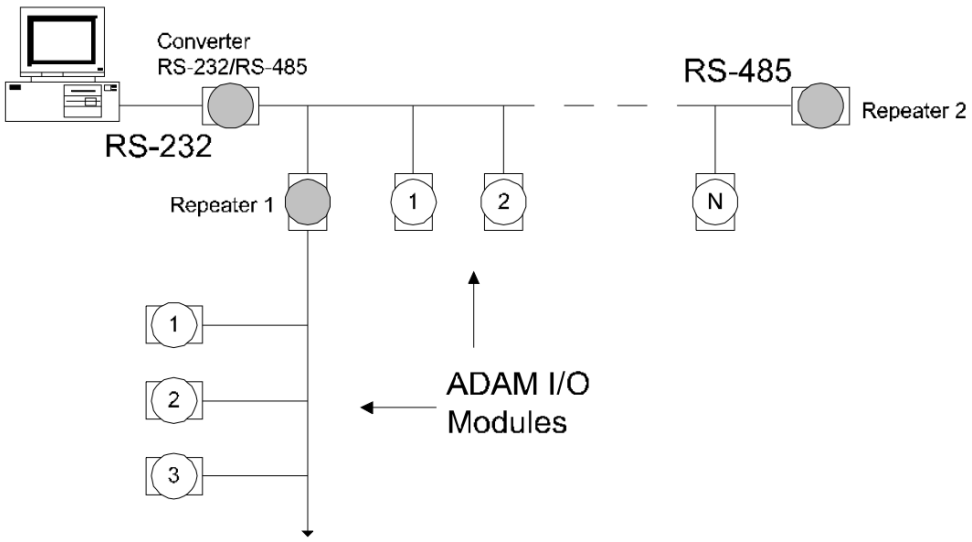
Column Name	Description
Name	The channel name
Access Type	Read Coil Status (Function Code 01) Read Holding Register (Function Code 03)

	Force Multiple Coils (Function Code 15) Preset Multiple Registers (Function Code 16)
Trigger	CYCLIC: Trigger main task periodically. RISING_EDGE: Trigger main task while the boolean trigger variable is in a rising edge.
Cycle Time	If set Trigger as CYCLIC, it represents poll interval (in milliseconds)
READ Offset	Register number for reading (Range 0-65535)
WRITE Offset	Register number for writing (Range 0-65535)
Length	Number of registers (=Words) to be read or written.

We have already introduced **Modbus Slave Configuration** and **Modbus Slave Channel.**
In the following section, we will introduce **Modbus Slave I/O Mapping** for ADAM-4000 Series and ADAM-5000 Series according to their Modbus function code.

5.1.1.1. ADAM-4000 Series

The ADAM-4000 series is a set of intelligent sensor-to-computer interface modules containing built-in microprocessor. They are remotely controlled through a simple set of commands issued in ASCII format and transmitted in RS-485 protocol.
The figure below shows the brief overview of the ADAM-4000 system architecture.



Advantech provides 15 types of ADAM-4000 modules for various applications so far. Following table is ADAM-4000 series support list.

Name	Specification
ADAM-4015	6-ch RTD Input Module
ADAM-4015T	6-ch Thermocouple Input Module
ADAM-4017+	8-ch Analog Input Module
ADAM-4018+	8-ch Analog Input Module
ADAM-4019+	8-ch Analog Input Module
ADAM-4117	8-ch Analog Input Module
ADAM-4118	8-ch Thermocouple Input Module
ADAM-4024	4-ch Analog Output Module
ADAM-4051	16-ch Digital Input Module
ADAM-4055	8-ch Digital Input and 8-ch Digital Output Module
ADAM-4150	8-ch Digital Input and 8-ch Digital Output Module
ADAM-4056S (SO)	12-ch Digital Output Module
ADAM-4068	8-ch Relay Output Module
ADAM-4069	8-ch Relay Output Module
ADAM-4168	8-ch Relay Output Module

5.1.1.1.1. Read Coil Status

The Modbus Function 01 is used to read coil status, or the ON/OFF status of digital input (DI) modules. Open the **Modbus Device editor** by double-clicking on the device icon in the

device tree. In **ModbusGenericSerialSlave I/O Mapping**, it shows the I/O mapping status between variable to module channel. It consists of seven columns.

Mapping: The mapping status of each variable.

The data type of each channel is in single bit. If the value is “true”, it means that the channel is on; “false” for off. For detailed variable mapping information, see chapter 4.2.

Address: The starting physical address of the variables for this I/O group. The board shown below has 8 digital inputs. This will require either 8 Boolean addresses or 2 Byte addresses.

Note!

Meaning of address expression:

% = Directly Mapped variable

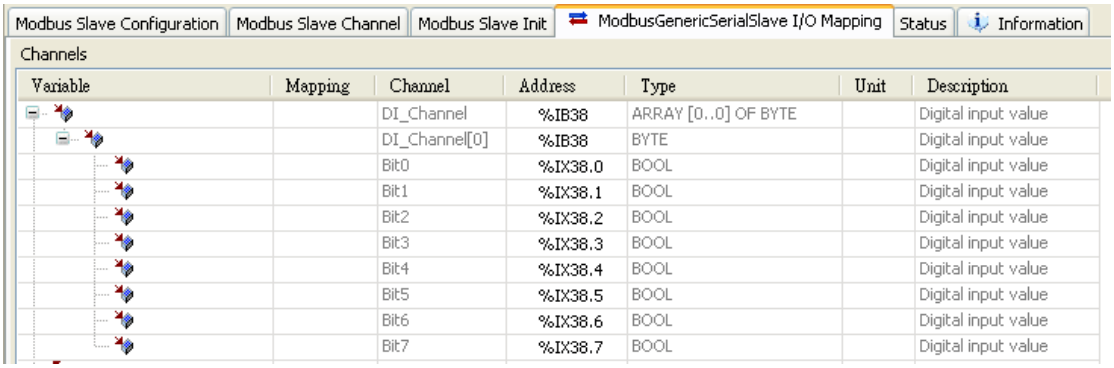
I = Physical Input

X = Single bit

$(N1). $(N2) = The starting address. The first number means the starting byte; the second number means the starting bit.

Type: The data type of each variable.

Description: The description of each variable.



5.1.1.1.2. Read Holding Registers

The Modbus Function 03 is used to read holding registers, or the quantity of registers to read from the analog input (AI) modules. Open the **Modbus Device editor** by double-clicking on the device icon in the device tree. In **ModbusGenericSerialSlave I/O Mapping**, it shows the I/O mapping status between variable to module channel. It consists of seven columns.

Mapping: The mapping status of each variable.

The data type of each channel is in WORD.
For detailed variable mapping information, see chapter 4.2.

Address: The starting physical address of the variables for this I/O group. The board shown below has 8 analog inputs. This will require 8 WORD addresses.

Note!

Meaning of address expression:

% = Directly Mapped variable

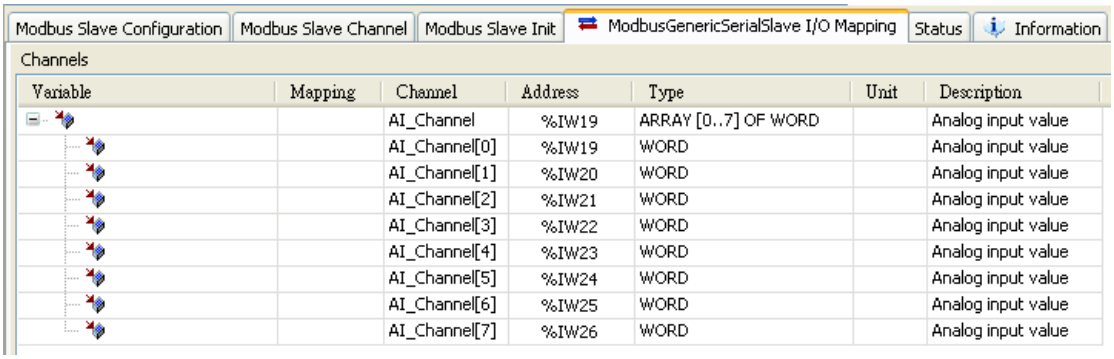
I = Physical Input

W = Single word (16 Bits)

$(N) = The starting address.

Type: The data type of each variable.

Description: The description of each variable.



Variable	Mapping	Channel	Address	Type	Unit	Description
		AI_Channel	%IW19	ARRAY [0..7] OF WORD		Analog input value
		AI_Channel[0]	%IW19	WORD		Analog input value
		AI_Channel[1]	%IW20	WORD		Analog input value
		AI_Channel[2]	%IW21	WORD		Analog input value
		AI_Channel[3]	%IW22	WORD		Analog input value
		AI_Channel[4]	%IW23	WORD		Analog input value
		AI_Channel[5]	%IW24	WORD		Analog input value
		AI_Channel[6]	%IW25	WORD		Analog input value
		AI_Channel[7]	%IW26	WORD		Analog input value

5.1.1.1.3. Force Multiple Coils

The Modbus Function 15 is used to force multiple coils or ON/OFF state to digital output (DO) modules. Open the **Modbus Device editor** by double-clicking on the device icon in the device tree. In **ModbusGenericSerialSlave I/O Mapping**, it shows the I/O mapping status between variable to module channel. It consists of seven columns.

Mapping: The mapping status of each variable.

The data type of each channel is in single bit. Set the value to “true” for switching on the channel; “false” for switching off. For detailed variable mapping information, see chapter 4.2.

Address: The starting physical address of the variables for this I/O group. The board shown below has 8 digital inputs. This will require either 8 Boolean addresses or 2 Byte addresses.

Note!

Meaning of address expression:

% = Directly Mapped variable

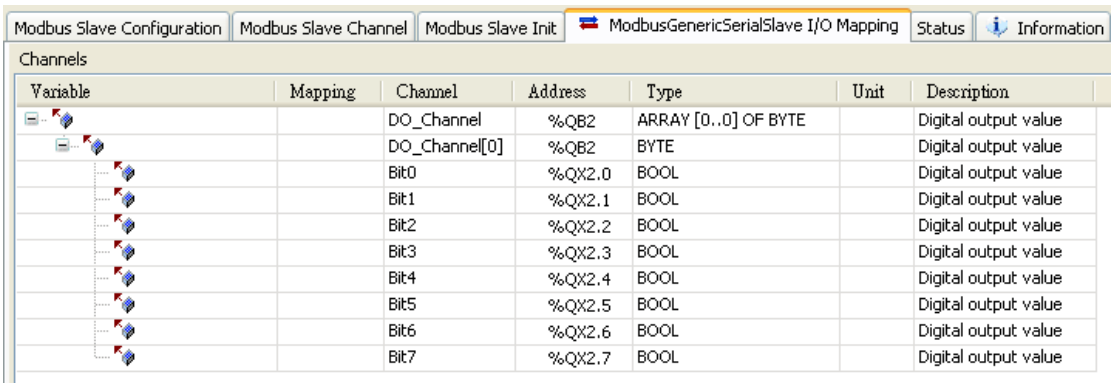
Q = Physical Output

X = Single bit

$(N1). $(N2) = The starting address. The first number means the starting byte; the second number means the starting bit.

Type: The data type of each variable.

Description: The description of each variable.



Variable	Mapping	Channel	Address	Type	Unit	Description
		DO_Channel	%QB2	ARRAY [0..0] OF BYTE		Digital output value
		DO_Channel[0]	%QB2	BYTE		Digital output value
		Bit0	%QX2.0	BOOL		Digital output value
		Bit1	%QX2.1	BOOL		Digital output value
		Bit2	%QX2.2	BOOL		Digital output value
		Bit3	%QX2.3	BOOL		Digital output value
		Bit4	%QX2.4	BOOL		Digital output value
		Bit5	%QX2.5	BOOL		Digital output value
		Bit6	%QX2.6	BOOL		Digital output value
		Bit7	%QX2.7	BOOL		Digital output value

5.1.1.1.4. Preset Multiple Registers

The Modbus Function 16 is used to preset multiple register or write the contents to analog output (AO) modules. Open the **Modbus Device editor** by double-clicking on the device icon in the device tree. In **ModbusGenericSerialSlave I/O Mapping**, it shows the I/O mapping status between variable to module channel. It consists of seven columns.

Mapping: The mapping status of each variable.

The data type of each channel is in BOOL.
For detailed variable mapping information, see chapter 4.2.

Address: The starting physical address of the variables for this I/O group. The board shown below has 16 digital inputs. This will require either 16 Boolean addresses or 2 Byte addresses.

Note!

Meaning of address expression:

% = Directly Mapped variable

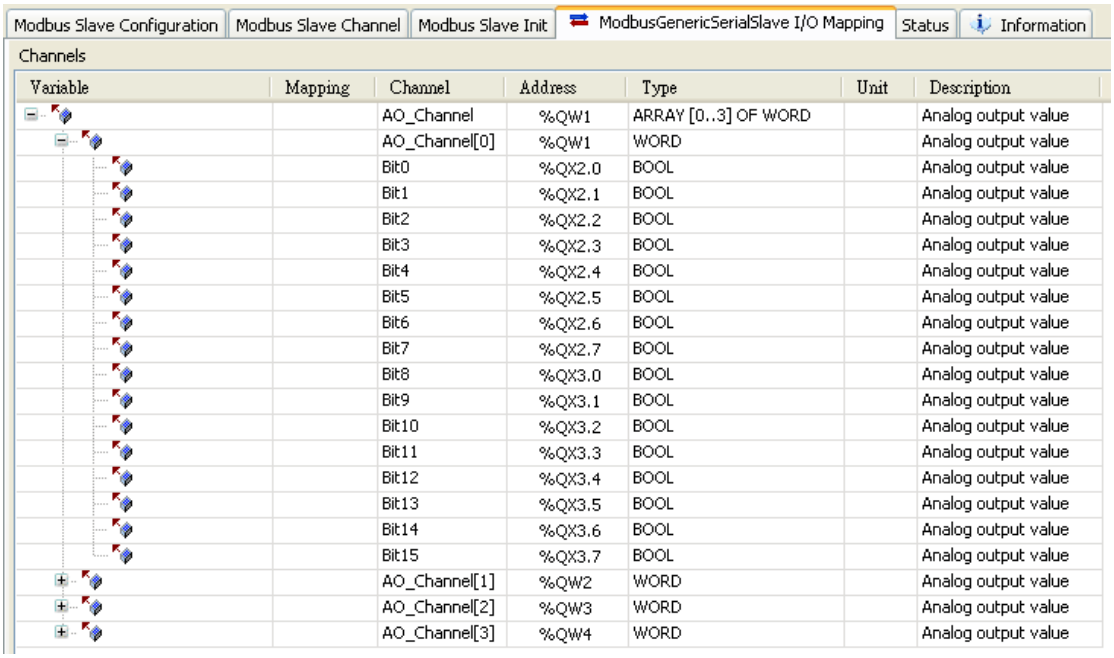
Q = Physical Output

X = Single bit

$(N1). $(N2) = The starting address. The first number means the starting byte; the second number means the starting bit.

Type: The data type of each variable.

Description: The description of each variable.

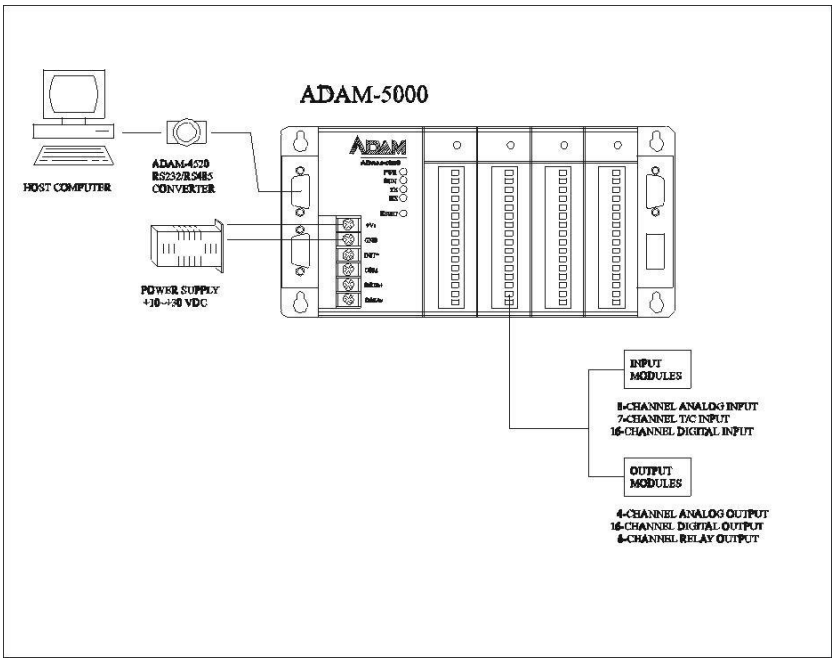


Variable	Mapping	Channel	Address	Type	Unit	Description
		AO_Channel	%QW1	ARRAY [0..3] OF WORD		Analog output value
		AO_Channel[0]	%QW1	WORD		Analog output value
		Bit0	%QX2.0	BOOL		Analog output value
		Bit1	%QX2.1	BOOL		Analog output value
		Bit2	%QX2.2	BOOL		Analog output value
		Bit3	%QX2.3	BOOL		Analog output value
		Bit4	%QX2.4	BOOL		Analog output value
		Bit5	%QX2.5	BOOL		Analog output value
		Bit6	%QX2.6	BOOL		Analog output value
		Bit7	%QX2.7	BOOL		Analog output value
		Bit8	%QX3.0	BOOL		Analog output value
		Bit9	%QX3.1	BOOL		Analog output value
		Bit10	%QX3.2	BOOL		Analog output value
		Bit11	%QX3.3	BOOL		Analog output value
		Bit12	%QX3.4	BOOL		Analog output value
		Bit13	%QX3.5	BOOL		Analog output value
		Bit14	%QX3.6	BOOL		Analog output value
		Bit15	%QX3.7	BOOL		Analog output value
		AO_Channel[1]	%QW2	WORD		Analog output value
		AO_Channel[2]	%QW3	WORD		Analog output value
		AO_Channel[3]	%QW4	WORD		Analog output value

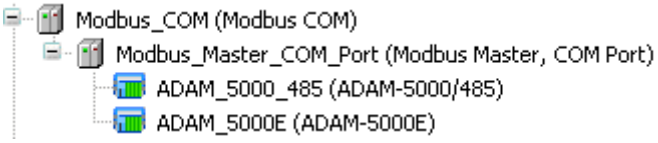
5.1.1.2. ADAM-5000 Series

The ADAM-5000 series is a complete product line that provides a wide variety of features in a data acquisition and control application. It includes 4 I/O-slots **ADAM-5000/485** and 8 I/O-slots **ADAM-5000E**. They are remotely controlled by the host computer through a set of commands and transmitted in a RS-485 / RS-232 network. Please refer to Chapter 4.3 and check your ADAM I/O modules are supported by CoDeSys.

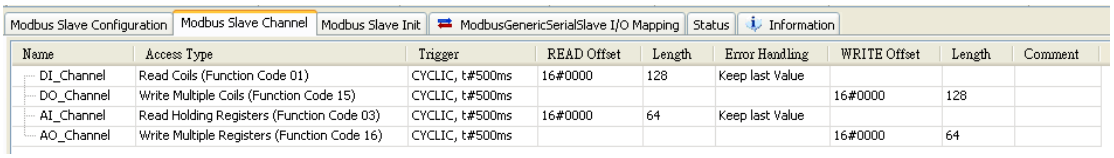
The following diagram shows the system configurations possible with the ADAM-5000.



Please refer to Chapter 5.1 and add **ADAM-5000/485 (4-slot)** or **ADAM-5000E (8-slot)** to device list.



Open the **Modbus Device editor** by double-clicking on the device icon in the device tree. The editor is subdivided in the 6 tabs. For more detailed information about **Modbus Slave Configuration** and **Modbus Slave Channel**, please refer to Chapter 5.1.1.



For ADAM-5000 series, Modbus slave address has been automatically assigned. CoDeSys create default channel setting for different type module. The Modbus address mapping tables are shown below.

For Digital Input / Output Module:

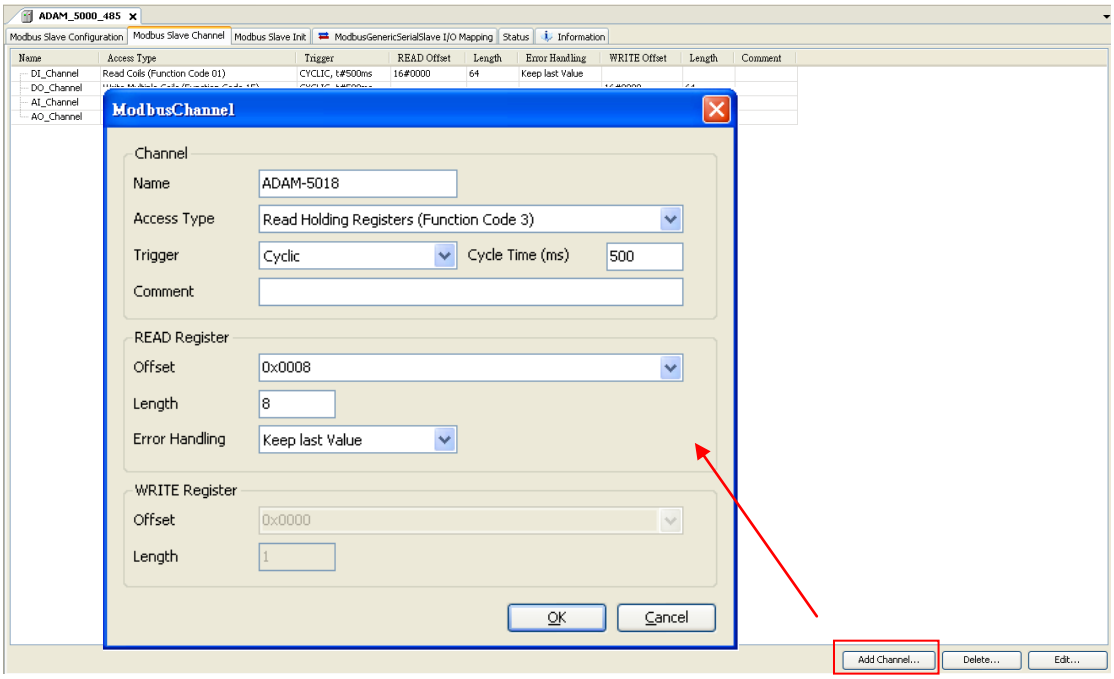
	Slot 0	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7
Bit 0	00001	00017	00033	00049	00065	00081	00097	00113
Bit 1	00002	00018	00034	00050	00066	00082	00098	00114
Bit 2	00003	00019	00035	00051	00067	00083	00099	00115
Bit 3	00004	00020	00036	00052	00068	00084	00100	00116
Bit 4	00005	00021	00037	00053	00069	00085	00101	00117
Bit 5	00006	00022	00038	00054	00070	00086	00102	00118
Bit 6	00007	00023	00039	00055	00071	00087	00103	00119
Bit 7	00008	00024	00040	00056	00072	00088	00104	00120
Bit 8	00009	00025	00041	00057	00073	00089	00105	00121
Bit 9	00010	00026	00042	00058	00074	00090	00106	00122
Bit 10	00011	00027	00043	00059	00075	00091	00107	00123
Bit 11	00012	00028	00044	00060	00076	00092	00108	00124
Bit 12	00013	00029	00045	00061	00077	00093	00109	00125
Bit 13	00014	00030	00046	00062	00078	00094	00110	00126
Bit 14	00015	00031	00047	00063	00079	00095	00111	00127
Bit 15	00016	00032	00048	00064	00080	00096	00112	00128

For Analog Input / Output Module:

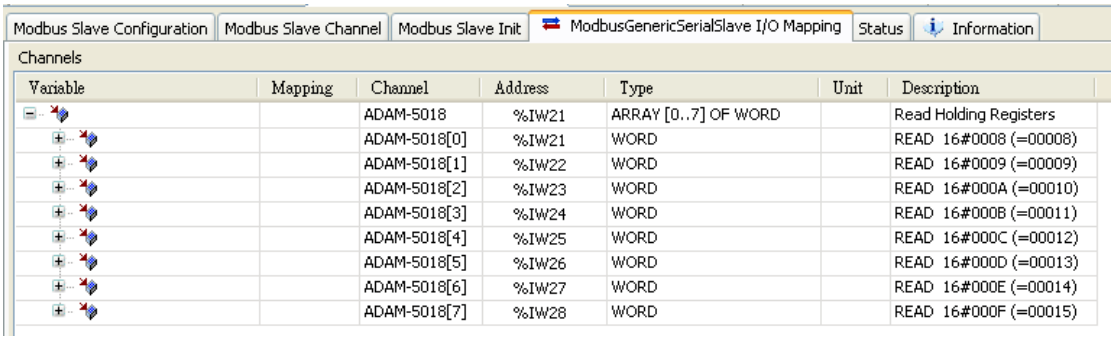
	Slot 0	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7
Ch 0	40001	40009	40017	40025	40033	40041	40049	40057
Ch 1	40002	40010	40018	40026	40034	40042	40050	40058
Ch 2	40003	40011	40019	40027	40035	40043	40051	40059
Ch 3	40004	40012	40020	40028	40036	40044	40052	40060
Ch 4	40005	40013	40021	40029	40037	40045	40053	40061
Ch 5	40006	40014	40022	40030	40038	40046	40054	40062

Ch 6	40007	40015	40023	40031	40039	40047	40055	40063
Ch 7	40008	40016	40024	40032	40040	40048	40056	40064

You can use default channel setting or add a new channel to a Modbus slave. For example, you insert ADAM-5018 (Analog Input module) in the second slot (slot 1) of ADAM-5000E. According to the Modbus address mapping table, start address as 40009 and length as 8. Now, go to **Modbus Slave Channel** page and click **Add Channel.** Select **Access type** as **Read Holding Register** and fill in **Offset** and **Length.** This dialog may be closed by clicking **OK** for creating the channel.



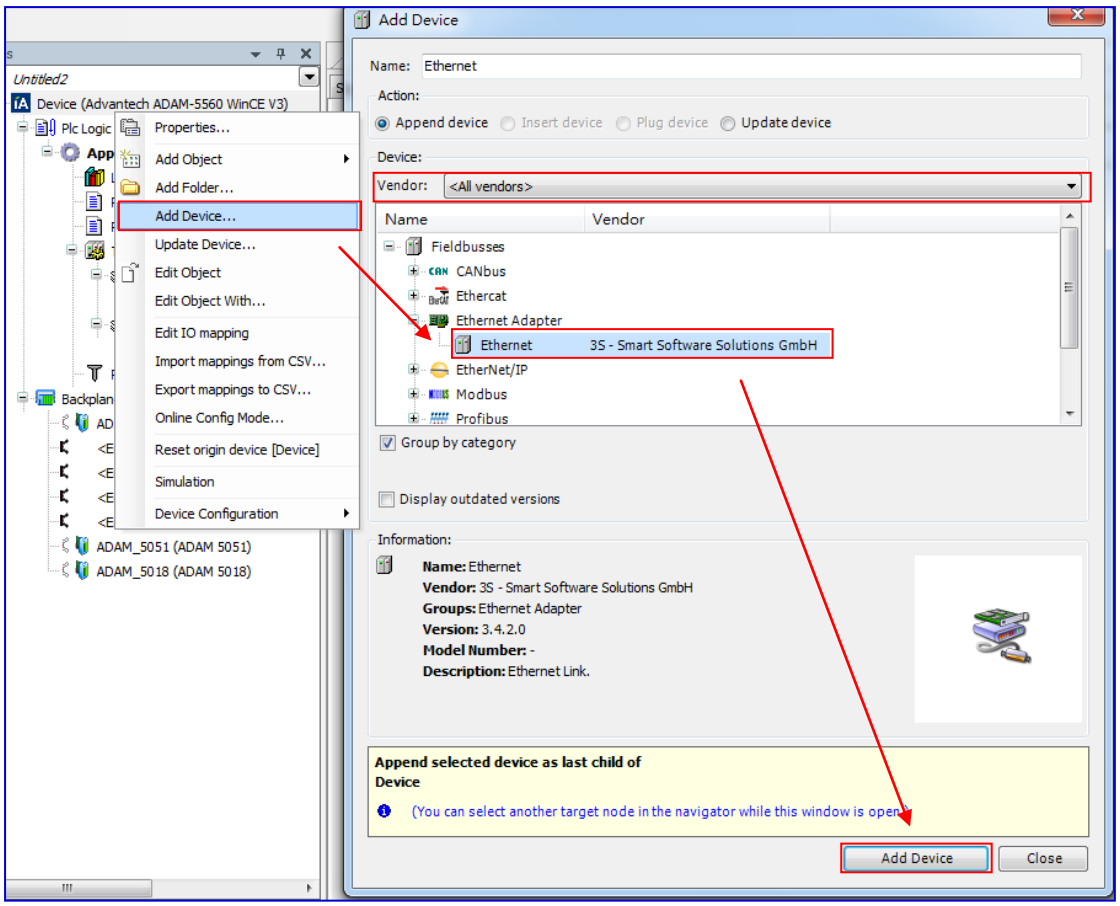
Correspondent to the set channels, the slave's process data can then be monitored under **ModbusGenericSerialSlave I/O Mapping** page.

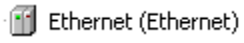


For more detailed information about how to control ADAM I/O modules, please refer to Chapter 4.4 to Chapter 4.9.

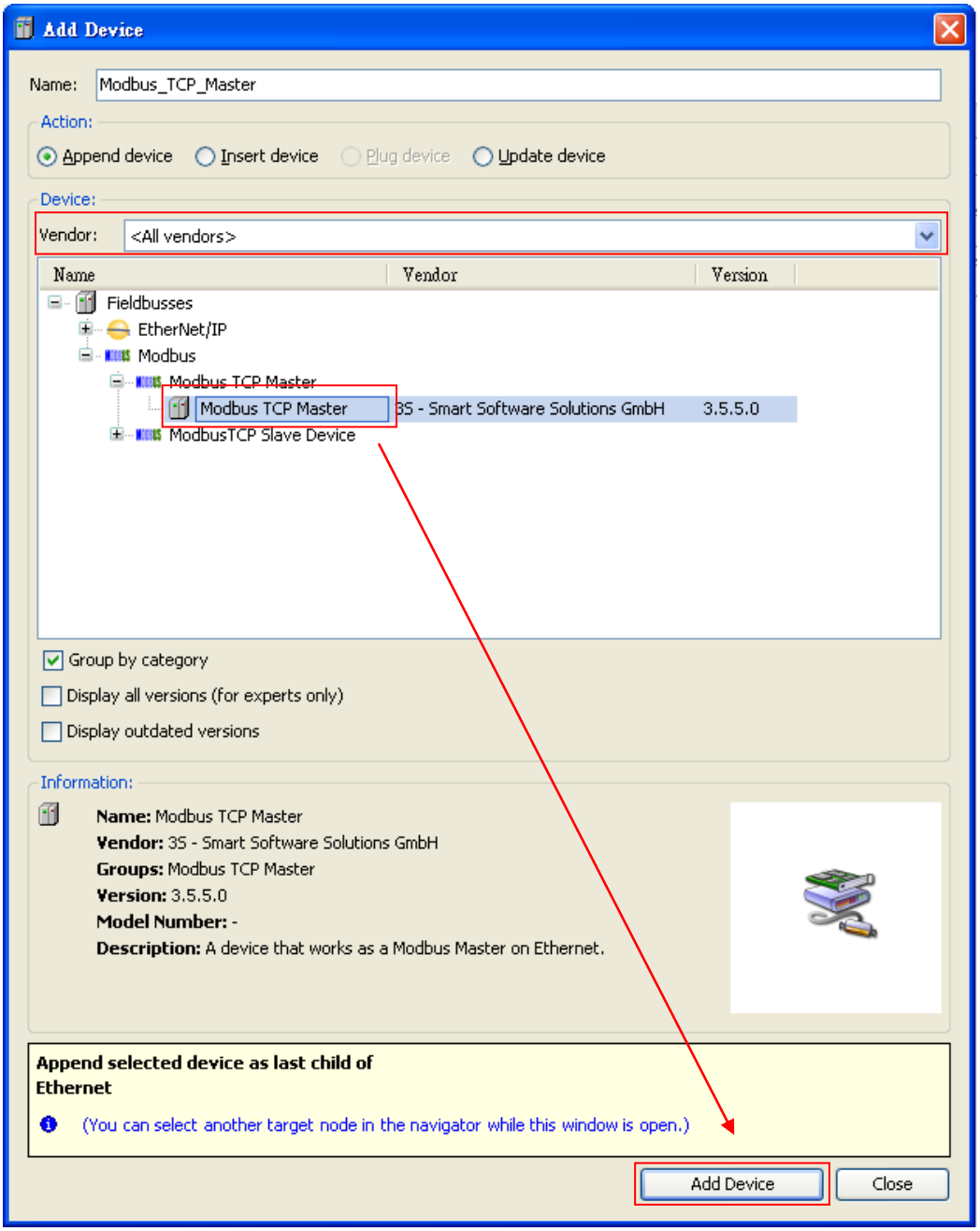
5.1.2. Modbus TCP Client

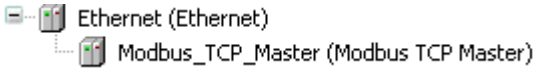
In Device Window, Right-click on **Device** and click **Add Device.** You will then be prompted for the Add Device dialog. Choose **Ethernet** in **Ethernet Adapter** and click **Add Device** to close the dialog.



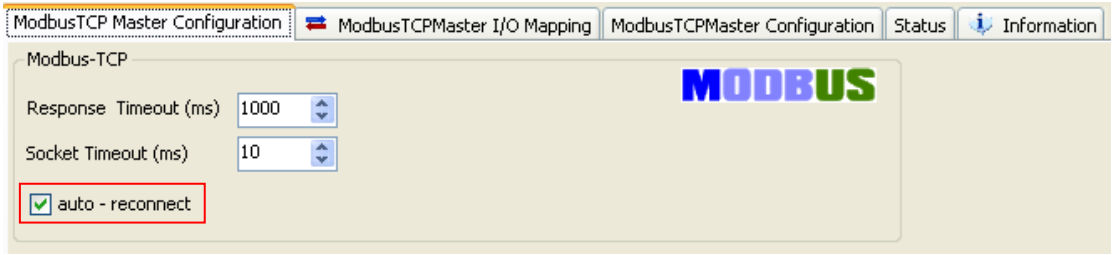
Right click on Ethernet adapter 

in the device tree and click **Add Device.** Type the device name in Name container. Choose **Modbus TCP Master** in the **Modbus** option. Click **Add Device** to proceed and then press **Close** to close the device dialog.

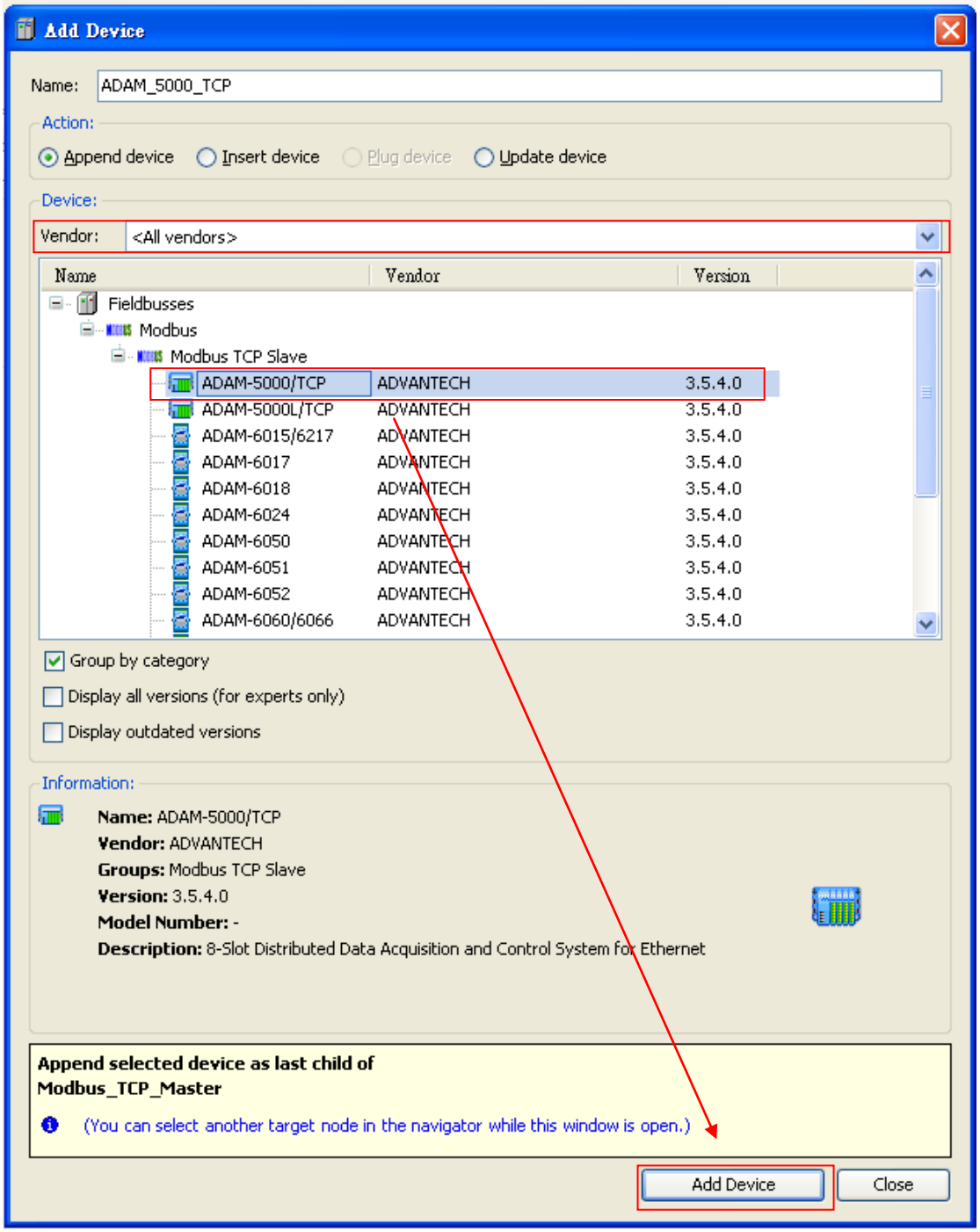


Now, you'll see Modbus TCP master 

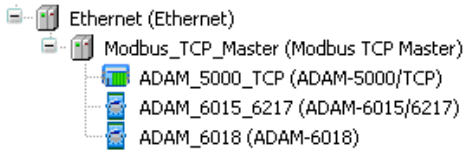
in the device tree. Double-click Modbus TCP master to set master configuration. For further convenience, recommend you to check **auto-restart communication**.



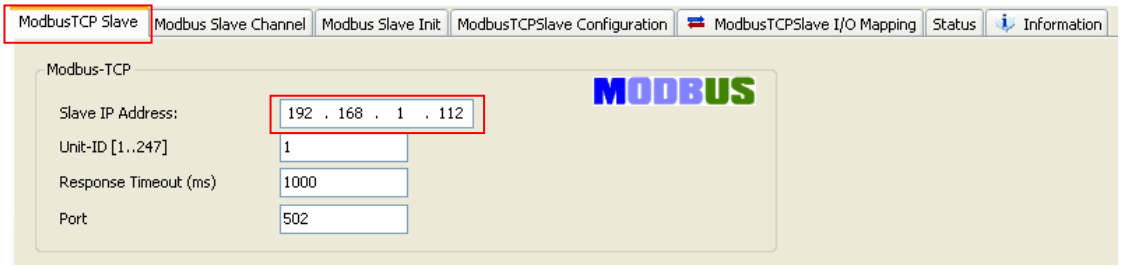
Right click on Modbus TCP master in the device tree and click **Add Device** in context menu. It will open the **Add Device dialog**, where you choose one available device for the current connection. Type the device name in Name container. Click **Add Device** to proceed and then press **Close** to close the device dialog.



Now, you will see Advantech ADAM modules on device list.

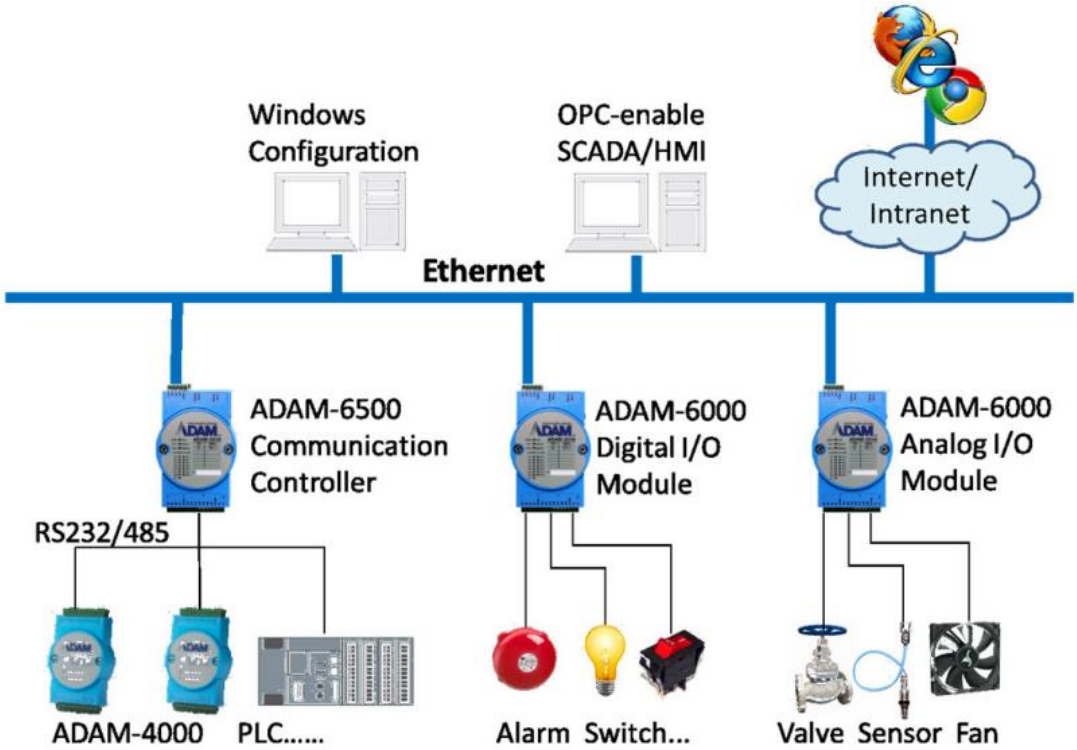


Now, set the IP address at ADAM module. Double-click on the device icon in the device tree and open the respective editors. Enter the module IP address in the register-tab "Modbus TCP Slave" (in this example: address 192.168.1.112) and keep port as 502.



5.1.2.1. ADAM-6000 Series

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.
The figure below shows the brief overview of the ADAM-6000 system architecture.



Advantech provides 16 types of ADAM-6000 modules for various applications so far. Following table is ADAM-6000 series support list.

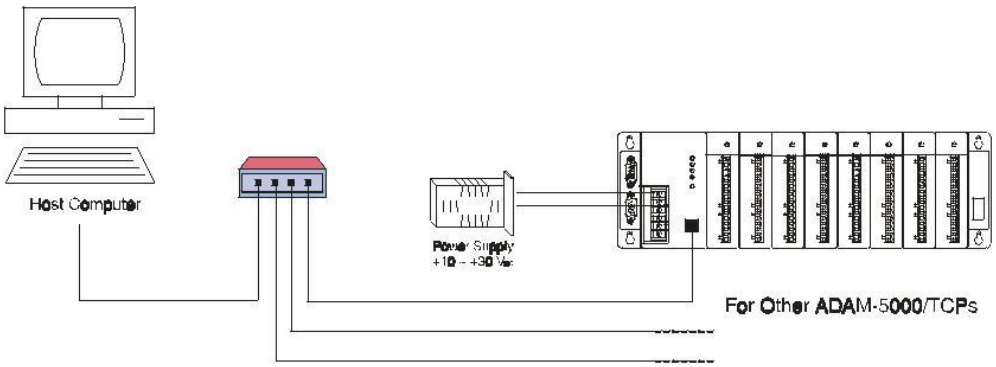
Name	Specification
ADAM-6015	7-ch RTD Input Module
ADAM-6217	8-ch Analog Input Module
ADAM-6017	8-ch Analog Input Module with 2-ch DO
ADAM-6018	8-ch Thermocouple Input Module with 8-ch DO
ADAM-6050	18-ch Digital I/O Module
ADAM-6051	14-ch Digital I/O Module with 2-ch Counter
ADAM-6052	16-ch Digital I/O Module
ADAM-6266	4-ch Relay Output Module with 4-ch DI
ADAM-6250	15-ch Digital I/O Module
ADAM-6060	6-ch Digital Input and 6-ch Relay Module
ADAM-6066	6-ch Digital Input and 6-ch Power Relay Module
ADAM-6024	12-ch Universal Input/Output Module
ADAM-6224	4-ch Analog Output Module
ADAM-6251	16-ch Digital Input Module
ADAM-6256	16-ch Digital Output Module
ADAM-6260	6-ch Relay Output Module

For more detailed information about how to control ADAM-6000 modules, please refer to Chapter 5.1.1.1.1 to Chapter 5.1.1.1.4.

5.1.2.2. ADAM-5000 Series

ADAM-5000/TCP Series works as a Modbus data server. It allows PCs or tasks to access its current data simultaneously from LAN, Intranet, or Internet. The ADAM-5000/TCP Series uses a convenient backplane system common to the ADAM-5000 series. Advantech's complete line of ADAM-5000 modules integrates with the ADAM-5000/TCP Series to support your applications.

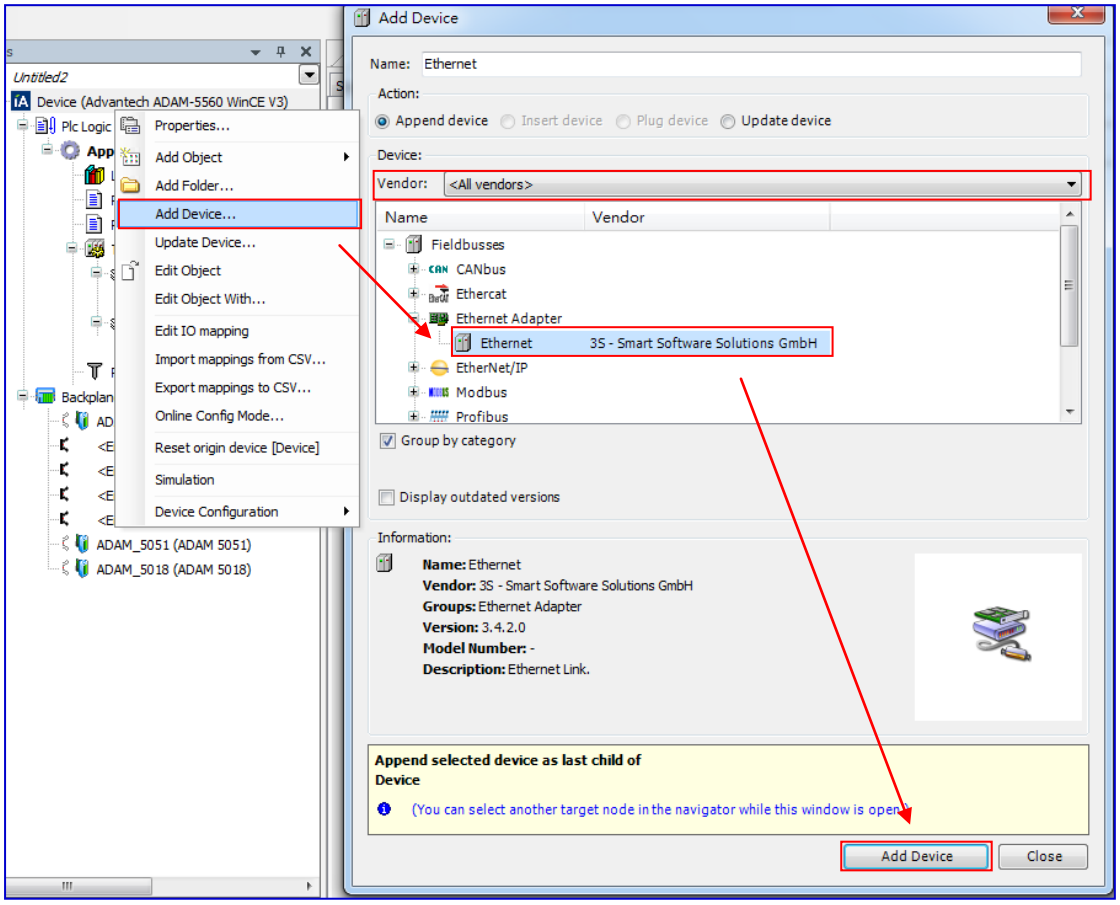
The figure below shows the brief overview of the ADAM-5000/TCP system architecture.

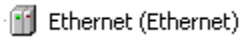


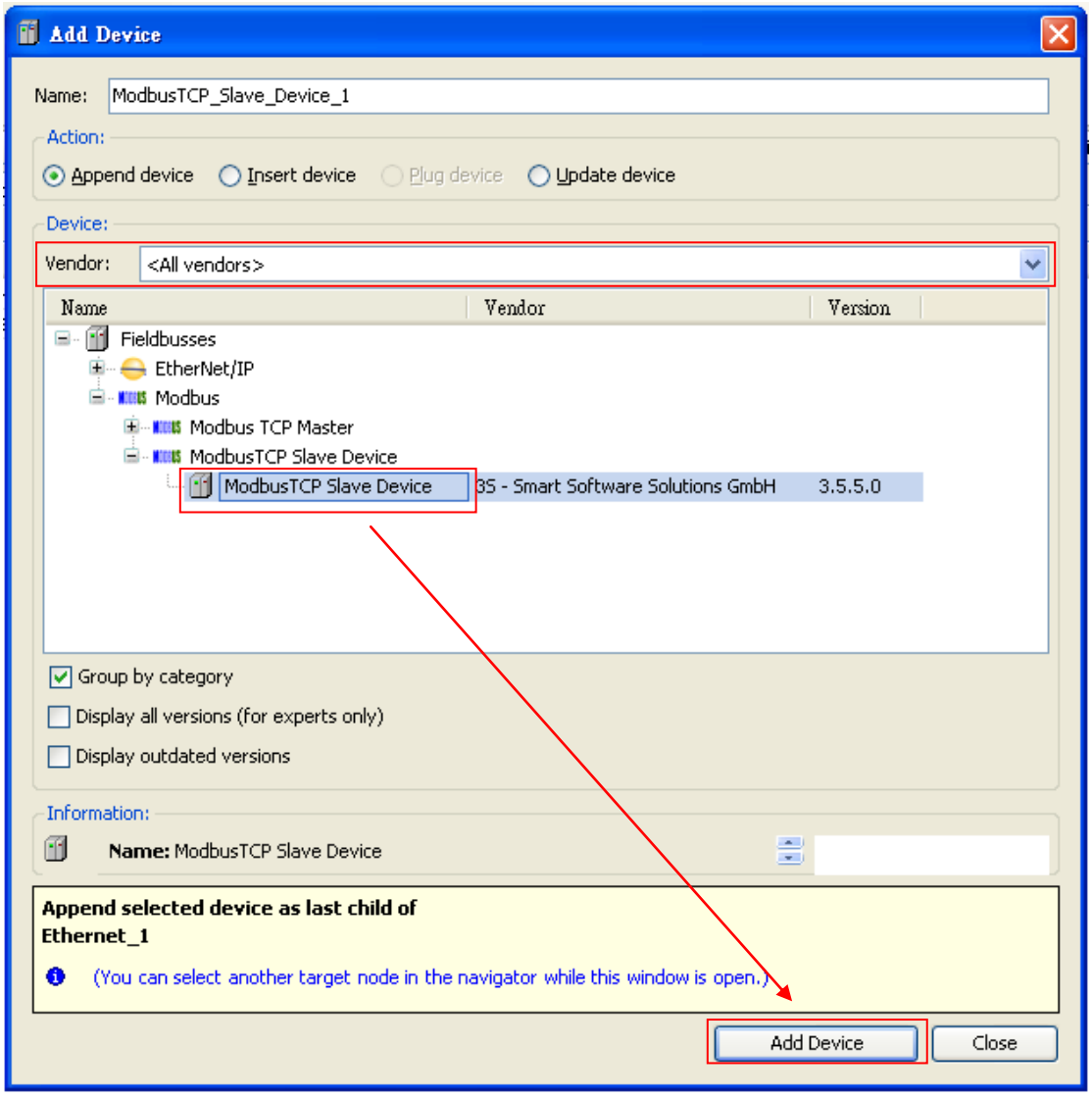
For more detailed information about how to access ADAM-5000 current data via Modbus protocol, please refer to Chapter 5.1.1.2.

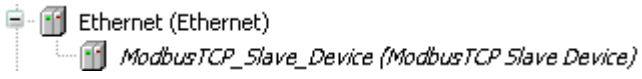
5.1.3. Modbus TCP Server

In Device Window, Right-click on **Device** and click **Add Device.** You will then be prompted for the Add Device dialog. Choose **Ethernet** in **Ethernet Adapter** and click **Add Device** to close the dialog.



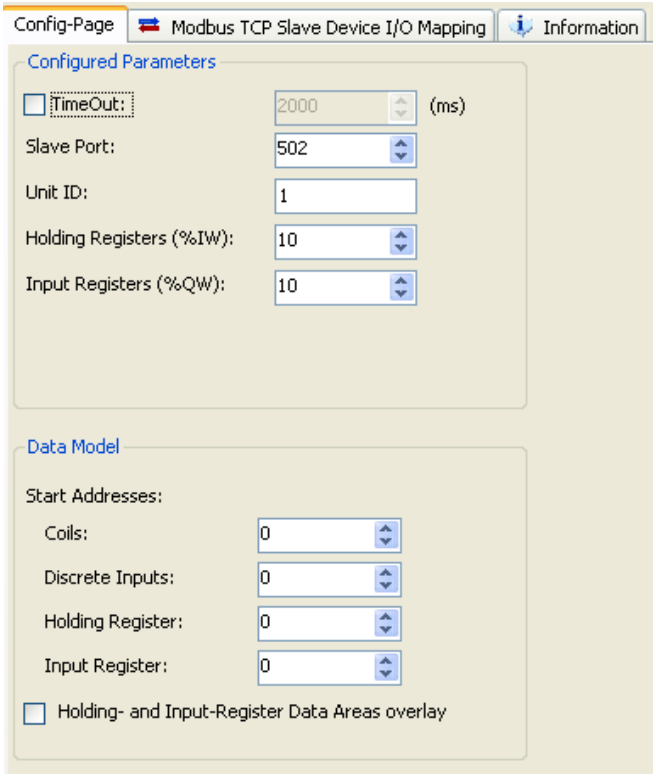
Right click on Ethernet adapter  in the device tree and click **Add Device.** Type the device name in Name container. Choose **Modbus TCP Slave** in the **Modbus** option. Click **Add Device** to proceed and then press **Close** to close the device dialog.



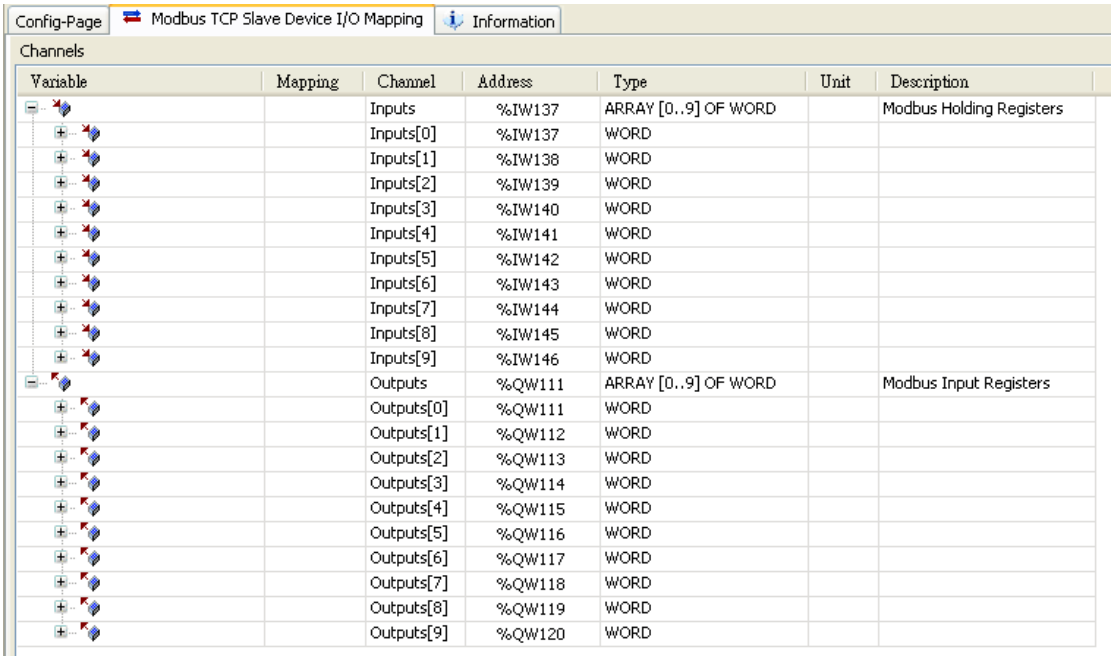
Now, you'll see Modbus TCP slave 

in the device tree. Double-click Modbus TCP slave and go to **Config-Page** to set configuration.

Timeout	Activation of the timing supervision function. The timeout interval is given in milliseconds. Values are adjustable in steps of 500 ms. If there is no write command received within this time, the outputs will be set to 0.**To keep the output value, we recommend you to uncheck Timeout.**
Slave Port	Port number of the slave. Keep slave port as 502.
Unit-ID	(Optional) Unit ID of the slave.
Holding Register	Number of holding register: Possible values: 1-4096. Maximum number can be limited in the device description.
Input Register	Number of input register: Possible values: 1-4096. Maximum number can be limited in the device description.



Correspondent to the number of holding register and input register, the slave channel can then be mapped under **Modbus TCP Slave Device I/O Mapping** page. For more detailed information about how to map variable, please refer to Ch4.2.



Variable	Mapping	Channel	Address	Type	Unit	Description
		Inputs	%IW137	ARRAY [0..9] OF WORD		Modbus Holding Registers
		Inputs[0]	%IW137	WORD		
		Inputs[1]	%IW138	WORD		
		Inputs[2]	%IW139	WORD		
		Inputs[3]	%IW140	WORD		
		Inputs[4]	%IW141	WORD		
		Inputs[5]	%IW142	WORD		
		Inputs[6]	%IW143	WORD		
		Inputs[7]	%IW144	WORD		
		Inputs[8]	%IW145	WORD		
		Inputs[9]	%IW146	WORD		
		Outputs	%QW111	ARRAY [0..9] OF WORD		Modbus Input Registers
		Outputs[0]	%QW111	WORD		
		Outputs[1]	%QW112	WORD		
		Outputs[2]	%QW113	WORD		
		Outputs[3]	%QW114	WORD		
		Outputs[4]	%QW115	WORD		
		Outputs[5]	%QW116	WORD		
		Outputs[6]	%QW117	WORD		
		Outputs[7]	%QW118	WORD		
		Outputs[8]	%QW119	WORD		
		Outputs[9]	%QW120	WORD		

Chapter 6

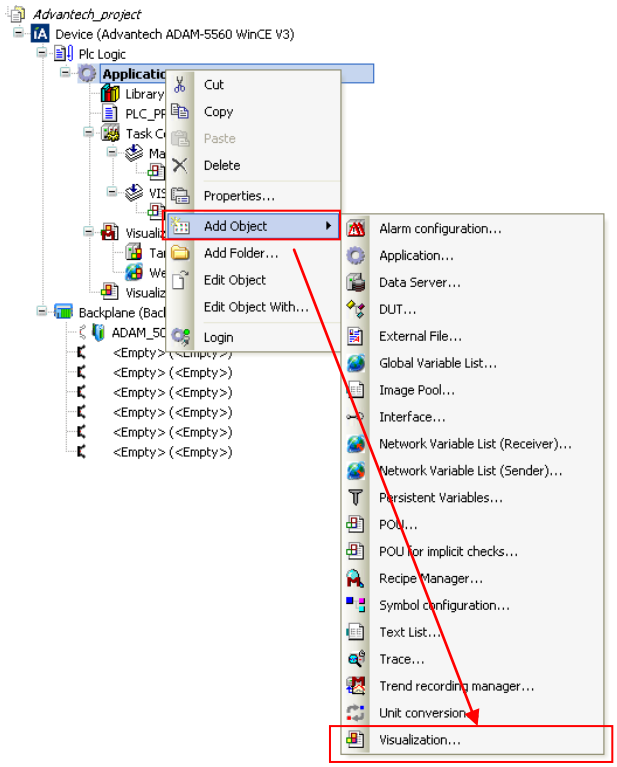
6. Examples

6.1. Visualization

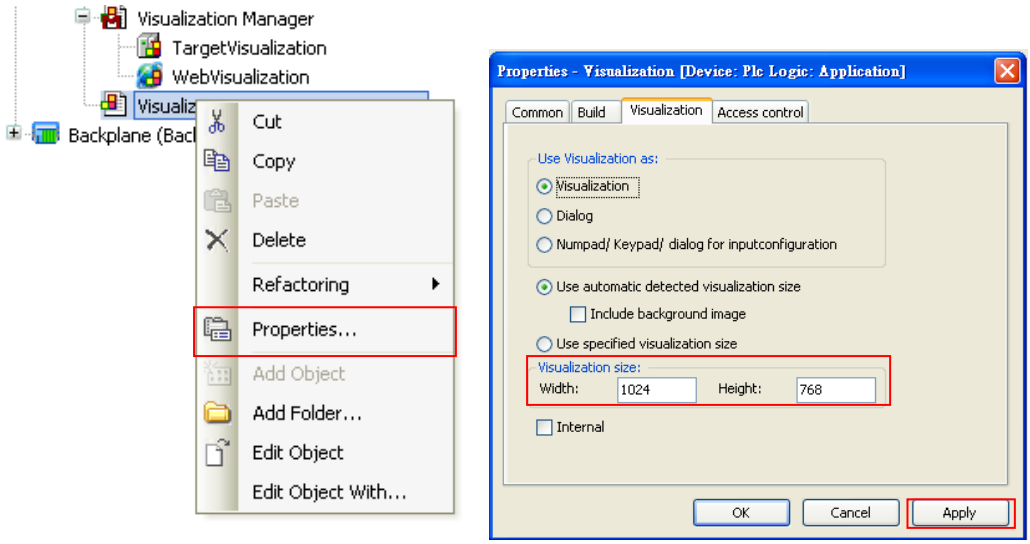
The CoDeSys visualization is a graphical representation of the project variables which allows inputs to the program in online mode via mouse and keypad. The CoDeSys visualization editor, which is part of the programming system, provides graphic elements which can be arranged as desired and can be connected with project variables. The following example project shows how to write a scrolling LED program in visualizations.

6.1.1. Create a new Visualization

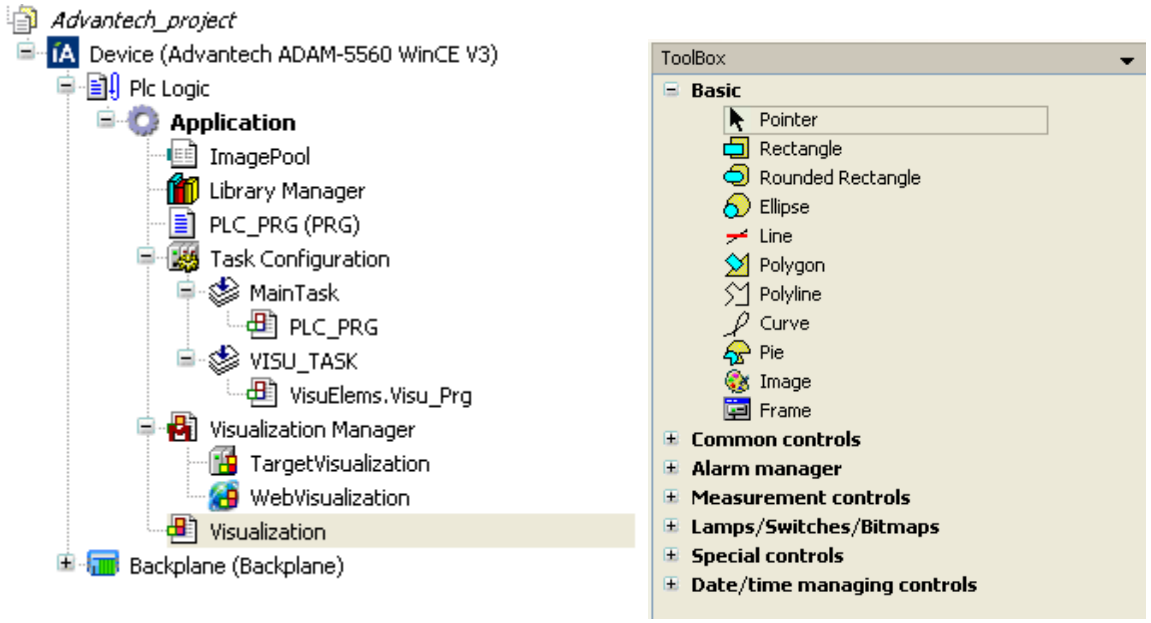
Step 1: To create a new visualization, right-click on **Application** and use command **Visualization** from the menu. You will then be prompted for **New visualization dialog.** Enter the name of the new visualization.



Step 2: Resize the width and height of the visualization object (number of pixels) by right-click on the visualization object.

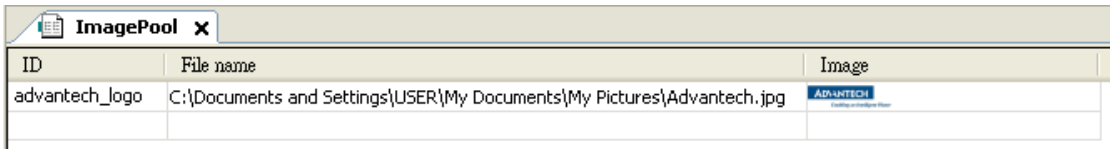


Step 3: Now, double-click on the visualization object and you'll see **ToolBox** on the right side. You can insert various geometric forms, as well as bitmaps, metafiles, buttons into your visualization by pushing down selected element into editor window.

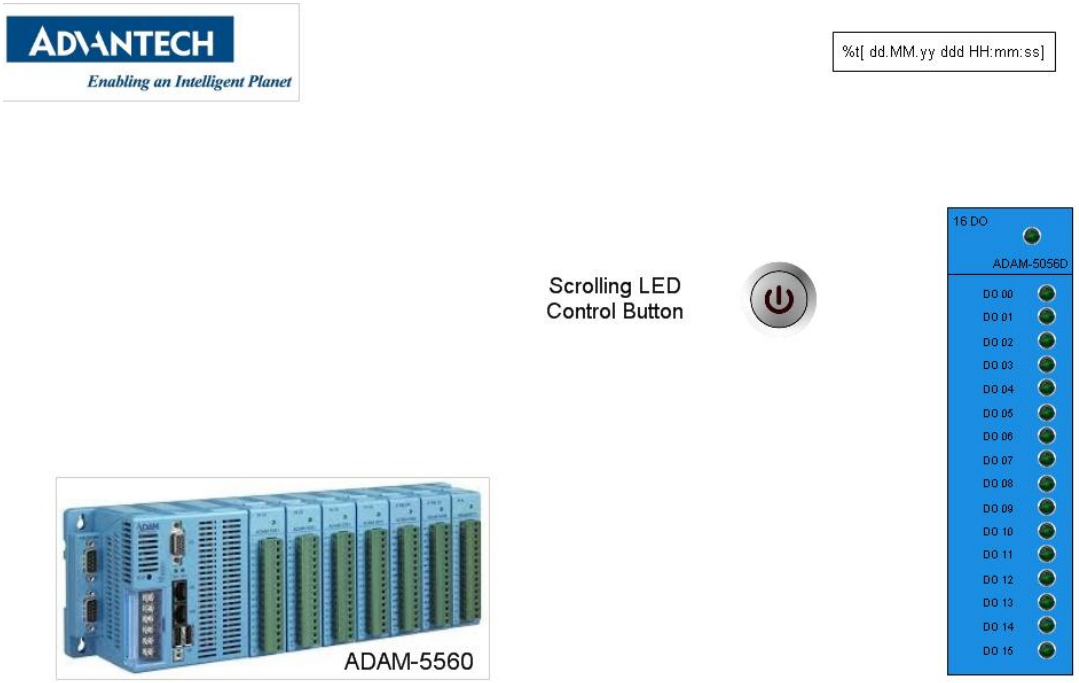


6.1.2. Visualize the Scrolling LED

Step 1: In this case, we need to insert Advantech images into the visualization, so we create an image pool. Again, right-click on **Application** and use command **Image Pool** from the menu. Enter string ID and the path of the image file



In order to visualize the scrolling LED, we insert lamps, rectangle, button and images.



Step 2: Create a PLC program in PLC_PRG(PRG). For more detailed information about how to write a program, please refer to Ch3.3.

PLC_PRG

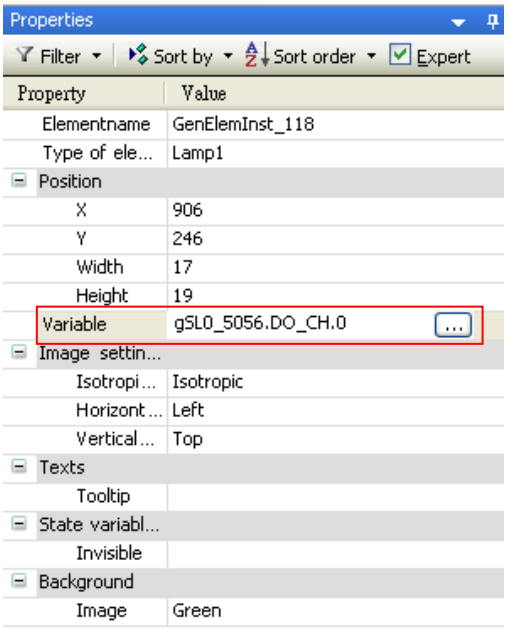
```
PROGRAM PLC_PRG
VAR
    n: WORD := 1;
    vLED: WORD := 1;
    switch: BOOL;
    SL0_5056_STATUS: BOOL;
END_VAR
```

```

IF switch THEN
    IF n <= 16 THEN
        vLED:=vLED * WORD#2;
        n:=n+1;
    ELSE
        n:=1;
        vLED:=WORD#1;
    END_IF;
END_IF;
IF gSL0_5056.Err <> 0 THEN
    SL0_5056_STATUS := FALSE;
ELSE
    SL0_5056_STATUS := TRUE;
END_IF

gSL0_5056.DO_CH := vLED;
```

Step 3: Map variables to the lamp objects. For the first lamp, map to first bit of DO channel variable (**gSL0_5056.DO_CH**). For more detailed information about how to map variable, please refer to Ch4.2.



Step 4: You have to connect to target device and running the program. Click on the button and the result was shown below.

6.2. Remanent Variables

Remanent variables retain their value throughout the usual program run time. They are declared as "Retain Variables" or "Persistent Variables". For keeping variables values even after the controller had been terminated or after the application has been reloaded, CODESYS offers different types:

6.2.1. Retain Variables

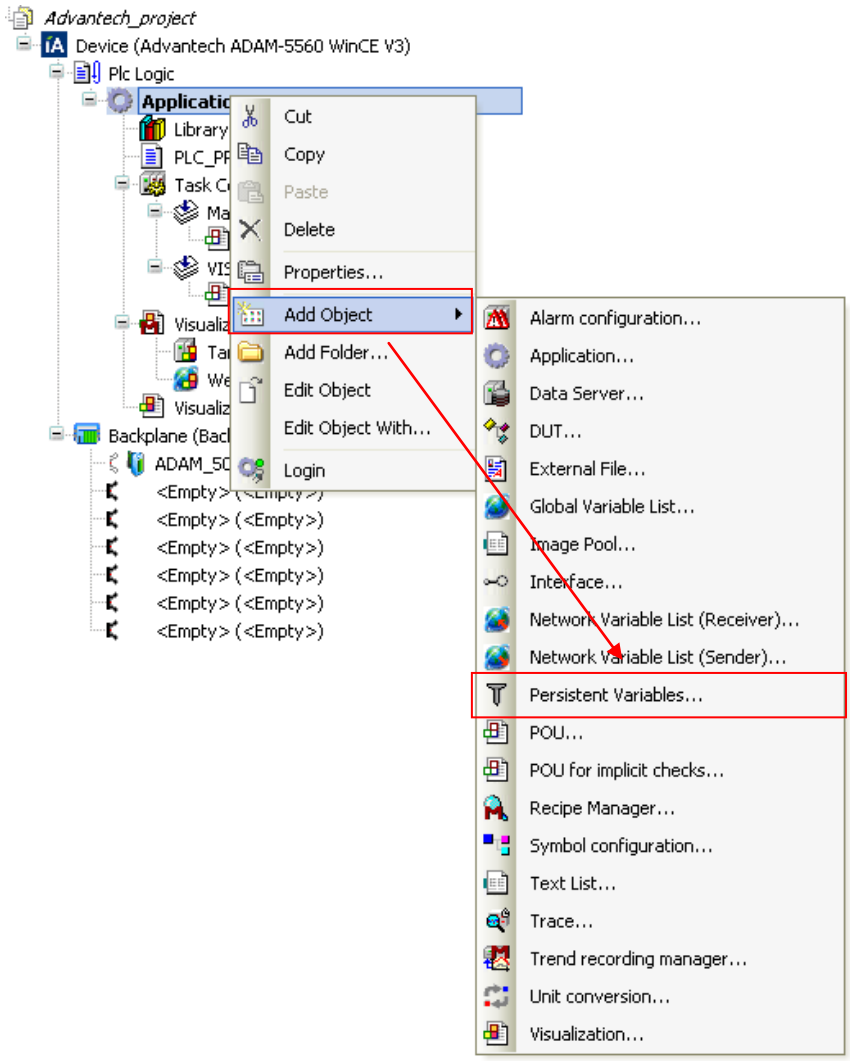
As we discuss how to declare variable in Ch3.3, we can declare retain variables by adding the keyword RETAIN" in the declaration part, behind the keyword for the base variable's type.

Here, we declare 1 retain variable “var3_retain” with initial value 1000 and 2 normal variables.

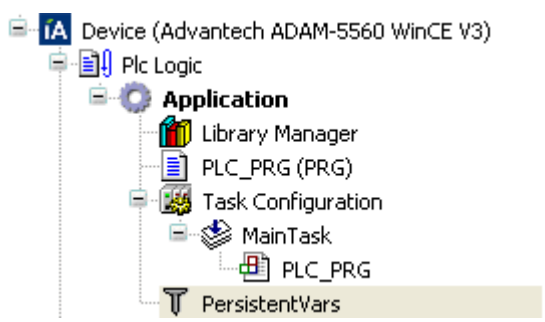
```
PLC_PRG
1  PROGRAM PLC_PRG
2  VAR
3      var1: int := 0;
4      var2_init: int := 1000;
5  END_VAR
6  VAR RETAIN
7      var3_retain: int := 1000;
8  END_VAR
```

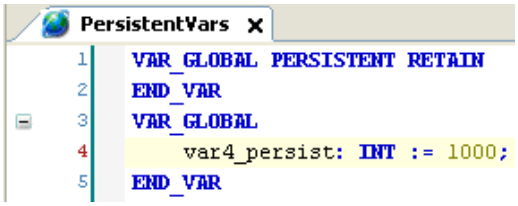
6.2.2. Persistent Variables

To create a new persistent variable, right-click on **Application** and use command **Persistent Variables** from the menu.



Double-click the object in the device tree. Here, we declare a global variable “var4_persist” with initial value 1000.





6.2.3. Variable Behavior

We want to investigate and compare the variable behavior between retain and persistent, so we keep adding their value in the body part of the PLC_PRG editor.

```
1  var1 := var1 +1;
2  var2_init := var2_init +1;
3  var3_retain := var3_retain +1;
4
5  var4_persist := var4_persist +1;
```

The result was shown below after we run our program on the target device.

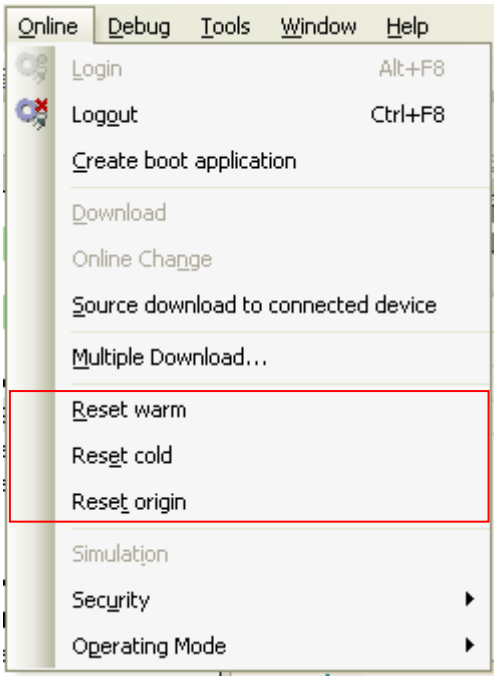
```
1  var1 37 := var1 37 +1;
2  var2_init 1037 := var2_init 1037 +1;
3  var3_retain 1037 := var3_retain 1037 +1;
4
5  var4_persist 1037 := var4_persist 1037 +1;RETURN
```

There three **Online Commands** for controlling the application program on a real or on the simulation target system after having logged in, including **Reset warm**, **Reset cold** and **Reset origin**.

The command **Reset warm** will reset (with exception of the retain and persistent variables) all variables of the currently active application to their initialization values. The situation is that which occurs in the event of a power outage or by turning the controller off, then on (warm restart) while the program is running.

The command **Reset cold** will reset (with exception of the persistent variables) all variables of the currently active application to their initialization values. The situation is that which occurs at the start of a program which has been downloaded just before to the target device.

The command **Reset origin** resets all variables of the currently active application, including the retain and persistent variables to their initialization values and erases the application on the target device.



The following is the overview on the behavior of remanent variables:

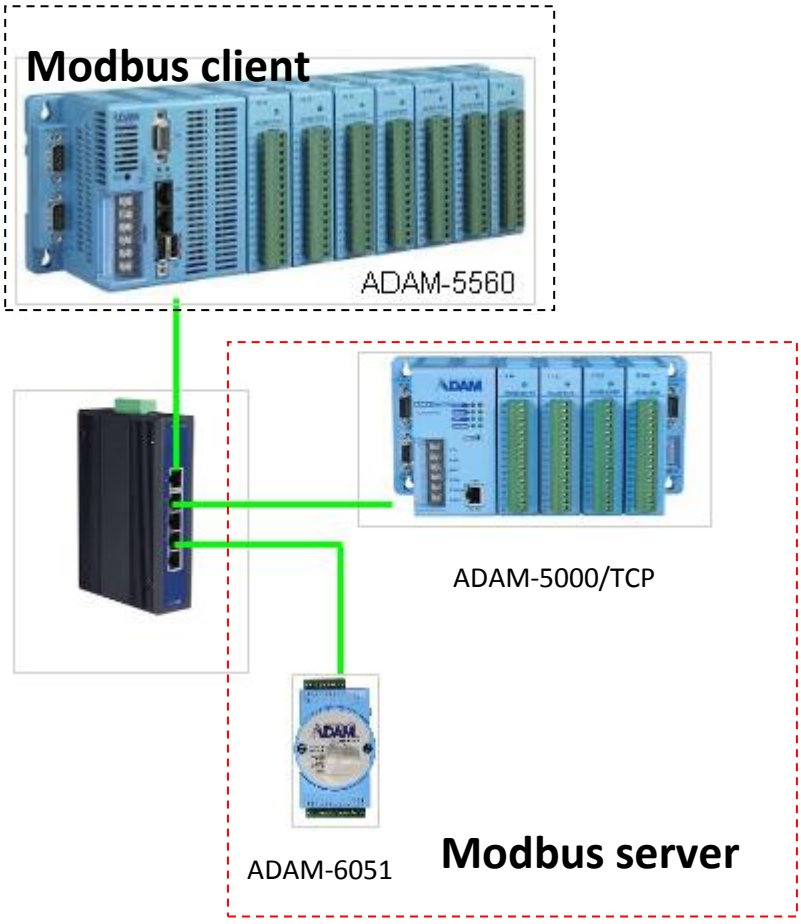
o = Value is maintained - = Value is initialized

After online command	**VAR**	**VAR RETAIN**	**VAR PERSISTENT RETAIN**
Reset warm <application>	-	o	o
Reset cold <application>	-	-	o
Reset origin <application>	-	-	-
Download <application>	-	-	o
Online Change <application>	o	o	o
Reboot the target device	-	o	o

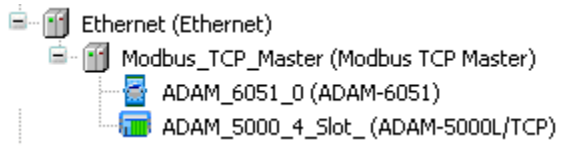
6.3. Modbus TCP Client

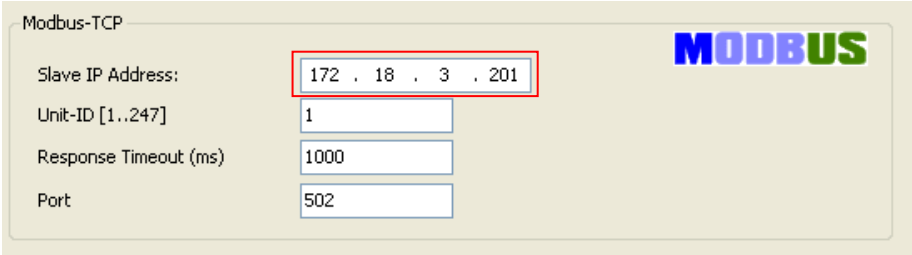
We use the following Advantech devices to demonstrate Modbus TCP client.

- ADAM-5560
- ADAM-5000/TCP with:
  – Slot 0: ADAM-5051(16-ch Digital Input)
  – Slot 1: ADAM-5056(16-ch Digital Output)
  – Slot 2: ADAM-5017(8-ch Analog Input)
  – Slot 3: ADAM-5024(4-ch Analog Output)
- ADAM-6051(14-ch Digital I/O with 2-ch Counter)



Step 1: Refer to Ch5.2 and add Modbus TCP master to the project. Remember to enter the device's IP address.

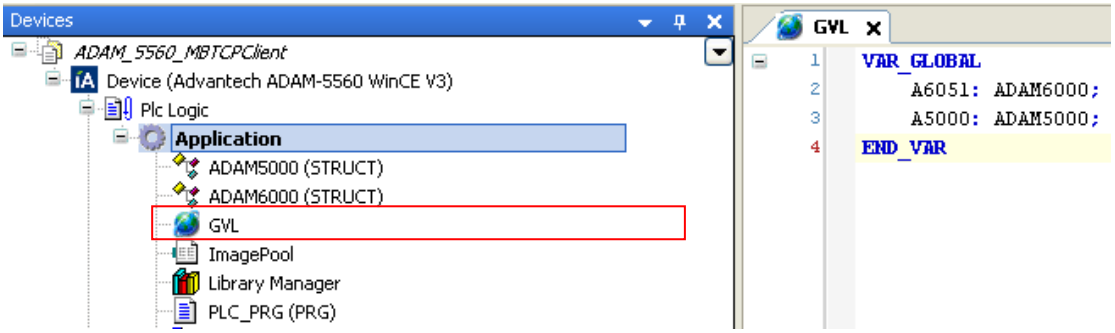




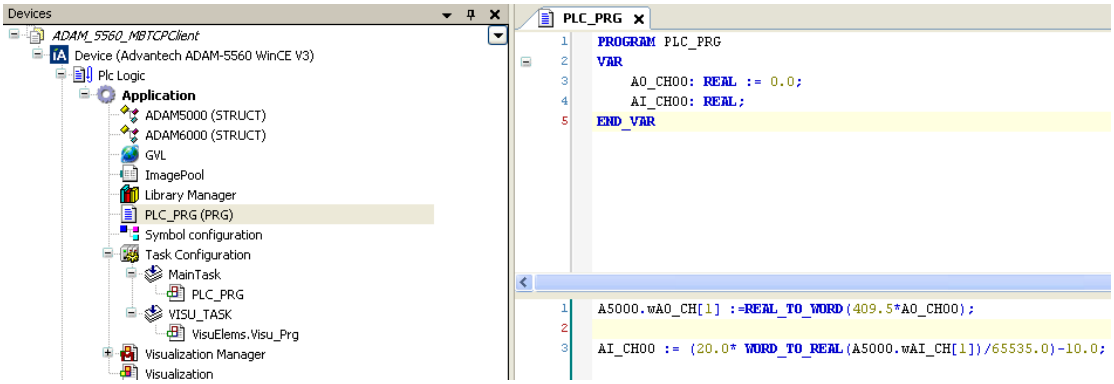
Step 2: Start to write our program. Define new structure to store ADAM-5000 and ADAM-6000 channel data and declare them as global variables.

```
TYPE ADAM5000 :
STRUCT
    byDI_CH : ARRAY [1..8] OF BYTE;
    byDO_CH : ARRAY [1..8] OF BYTE;
    wAI_CH:   ARRAY [1..32] OF WORD;
    wAO_CH:   ARRAY [1..32] OF WORD;
END_STRUCT
END_TYPE
```

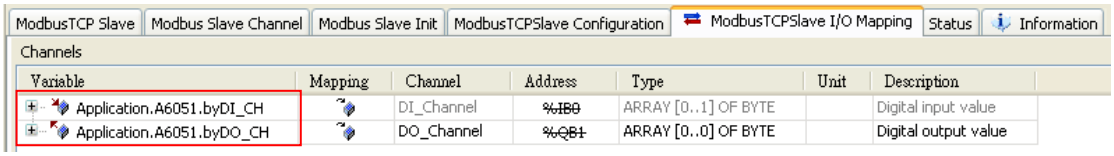
```
TYPE ADAM6000 :
STRUCT
    byDI_CH : ARRAY [1..2] OF BYTE;
    byDO_CH : ARRAY [1..2] OF BYTE;
    byAI_CH : ARRAY [1..8] OF WORD;
    byAO_CH : ARRAY [1..4] OF WORD;
END_STRUCT
END_TYPE
```

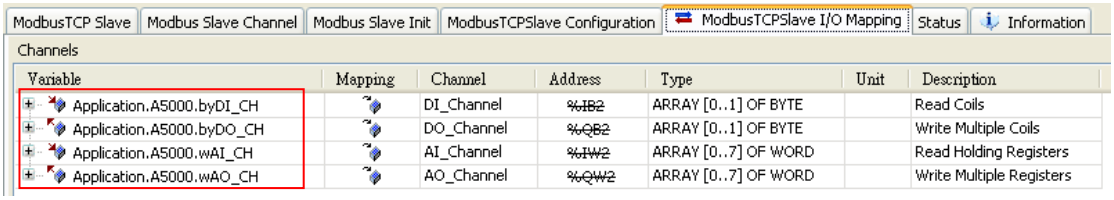


We write the program in PLC_PRG.



Step 3: Map Modbus data to the variables that we declared in previous step.





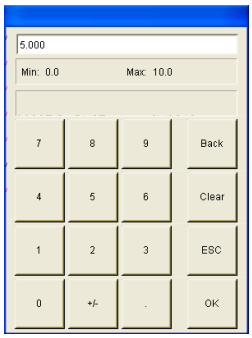
Step 4: Map all variables to the textfield objects in visualization.

Object			Mapping variable
SL0_ADAM-5051DI_CH00	1X0001	%d	Text variables Text variable: A5000.byDI_CH[1].0 Tooltip variable:
SL1_ADAM-5056DO_CH00	0X0017	%d	Text variables Text variable: A5000.byDO_CH[1].0 Tooltip variable: Inputconfiguration OnDialogClosed: Configure... OnMouseClick: Configure... OnMouseDown: Configure... Toggle a V...: A5000.byDO_CH[1].0
SL2_ADAM-5017AI_CH00	3X0017	%1.3fV	Text variables Text variable: PLC_PRG.AI_CH00 Tooltip variable:
SL3_ADAM-5024AO_CH00	4X0025	%1.3fV	Text variables Text variable: PLC_PRG.AO_CH00 Tooltip variable: Inputconfiguration OnDialogClosed: Configure... OnMouseClick: Configure... OnMouseDown: Configure... Write a Var...: Variable : , InputTyp...
ADAM-6051DI_CH00~07	1X0001~8	%d	Text variables Text variable: A6051.byDI_CH[1] Tooltip variable:
ADAM-6051DO_CH00~01	0X0017~18	%d	Text variables Text variable: A6051.byDO_CH[1] Tooltip variable: Inputconfiguration OnDialogClosed: Configure... OnMouseClick: Configure... OnMouseDown: Configure... Write a Var...: Variable : , InputType

Step 5: Compile our project and connect to the target device. The result was shown below.

For DO/AO channel, you can set its value by clicking on the textfield object and enter value in the pop-up dialog.

SL0_ADAM-5051DI_CH00	1X0001	0
SL0_ADAM-5051DI_CH01	1X0002	0
SL0_ADAM-5051DI_CH02	1X0003	0
SL0_ADAM-5051DI_CH03	1X0004	0
SL1_ADAM-5056DO_CH00	0X0017	0
SL1_ADAM-5056DO_CH01	0X0018	1
SL1_ADAM-5056DO_CH02	0X0019	1
SL1_ADAM-5056DO_CH03	0X0020	0
SL2_ADAM-5017AI_CH00	3X0017	0.001 V
SL3_ADAM-5024AO_CH00	4X0025	5.000 V



ADAM-6051DI_CH00~07	1X0001~8	255
ADAM-6051DI_CH08~11	1X0009~12	15
ADAM-6051DO_CH00~01	0X0017~18	0

6.4. Modbus TCP Server

We use the following Advantech devices to demonstrate Modbus TCP server.

- ADAM-5560

  –Slot 0: ADAM-5056(16-ch Digital Output)

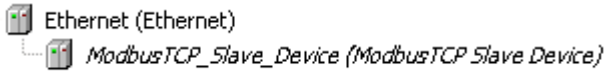
  – Slot 1: ADAM-5051(16-ch Digital Input)

  – Slot 2: ADAM-5017UH (8-ch Analog Input)

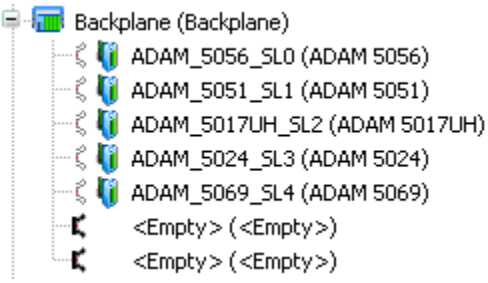
  – Slot 3: ADAM-5024(4-ch Analog Output)

  – Slot 4: ADAM-5069(8-ch Power Relay Output)

Step 1: Add the Modbus TCP slaves to the project and rename them if necessary.



Step 2: Add the Advantech I/O modules to the project.

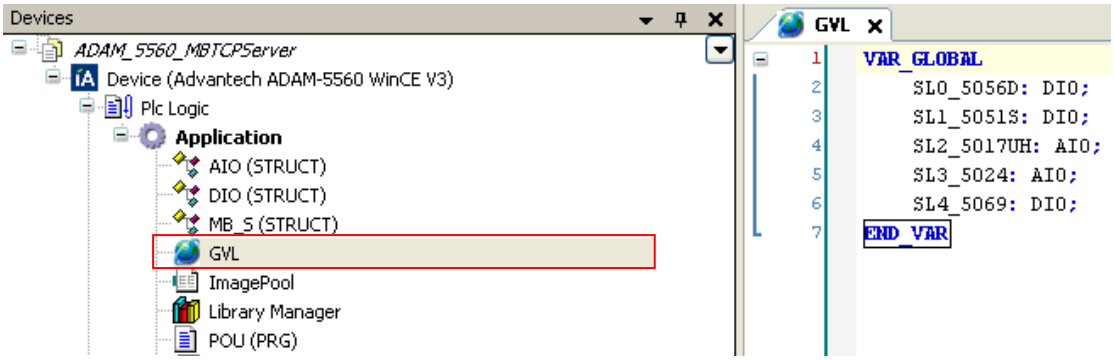


Step 3: Start to write our program. Define new structure to store DI/DO, AI/AO, Modbus channel data and declare them as global variables.

```
TYPE DIO :
STRUCT
    DI_CH:WORD;
    DO_CH:WORD;
    Err:WORD;
END_STRUCT
END_TYPE
```

```
TYPE AIO :
STRUCT
    CH00:REAL;
    CH01:REAL;
    CH02:REAL;
    CH03:REAL;
    CH04:REAL;
    CH05:REAL;
    CH06:REAL;
    CH07:REAL;
    Err:WORD;
END_STRUCT
END_TYPE
```

```
TYPE MB_S :
STRUCT
    MB3:ARRAY [1..100] OF WORD;
    MB4:ARRAY [1..100] OF WORD;
END_STRUCT
END_TYPE
```



We write the program in PLC_PRG.

POU

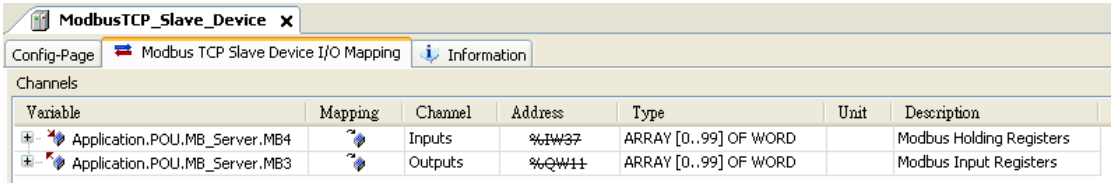
```
1  PROGRAM POU
2  VAR
3      MB_Server: MB_S;
4      IN1: WORD;
5      OUT1: WORD;
6      test: DWORD;
7      pREAL: POINTER TO REAL;
8  END_VAR
```

```
SL0_5056D.DO_CH:= MB_Server.MB4[1];//5056 DO CH0~15
pREAL:= ADR(MB_Server.MB4[3]);
SL3_5024.CH00 := pREAL^;//5024 AO CH0
pREAL:= ADR(MB_Server.MB4[5]);
SL3_5024.CH01 := pREAL^;//5024 AO CH1
pREAL:= ADR(MB_Server.MB4[7]);
SL3_5024.CH02 := pREAL^;//5024 AO CH2
pREAL:= ADR(MB_Server.MB4[9]);
SL3_5024.CH03 := pREAL^;//5024 AO CH3
SL4_5069.DO_CH:= MB_Server.MB4[11];//5069 CH0~7

MB_Server.MB3[1]:= SL0_5056D.Err ;//5056 Status
MB_Server.MB3[2]:= SL1_5051S.Err ;//5051 Status
MB_Server.MB3[3]:= SL2_5017UH.Err;//5017UH Status
MB_Server.MB3[4]:= SL3_5024.Err;//5024 Status
MB_Server.MB3[5]:= SL4_5069.Err;//5069 Status

MB_Server.MB3[11]:= SL1_5051S.DI_CH;//5051 CH0~15
MB_Server.MB3[13]:= %IW18;//5017UH CH0
MB_Server.MB3[14]:= %IW19;
MB_Server.MB3[15]:= %IW20;//5017UH CH1
MB_Server.MB3[16]:= %IW21;
MB_Server.MB3[17]:= %IW22;//5017UH CH2
MB_Server.MB3[18]:= %IW23;
MB_Server.MB3[19]:= %IW24;//5017UH CH3
MB_Server.MB3[20]:= %IW25;
MB_Server.MB3[21]:= %IW26;//5017UH CH4
MB_Server.MB3[22]:= %IW27;
MB_Server.MB3[23]:= %IW28;//5017UH CH5
MB_Server.MB3[24]:= %IW29;
MB_Server.MB3[25]:= %IW30;//5017UH CH6
MB_Server.MB3[25]:= %IW31;
MB_Server.MB3[27]:= %IW32;//5017UH CH7
MB_Server.MB3[28]:= %IW33;
```

Step 4: Set Modbus TCP server configuration and map Modbus data to the variables that we declared in previous step.



Step 5: Map all variables to the textfield objects in visualization.

Object			Mapping variable
ADAM-5056 CH0~CH15	4X0001	%d	Text variables Text variable: POU.MB_Server.MB4[1] Tooltip variable:
ADAM-5024 CH0	4X0003	%1.3f mA	Text variables Text variable: SL3_5024.CH00 Tooltip variable:
ADAM-5069 CH0~CH7	4X0011	%d	Text variables Text variable: POU.MB_Server.MB4[11] Tooltip variable:
ADAM-5051 CH0~CH15	3X0011	%d	Text variables Text variable: POU.MB_Server.MB3[11] Tooltip variable:
ADAM-5017UH CH0	3X0013	%1.3f mA	Text variables Text variable: SL2_5017UH.CH00 Tooltip variable:

Step 6: Compile our project and connect to the target device. The result was shown below.

ADVANTECH
Enabling an Intelligent Planet

26.02.15 Thu 01:03:07

Module Name	Address	Status
ADAM-5056	3X0001	0
ADAM-5051	3X0002	0
ADAM-5017UH	3X0003	0
ADAM-5024	3X0004	0
ADAM-5069	3X0005	0

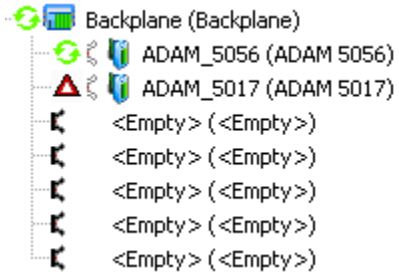
Module Channel	Address	Value
ADAM-5056 CH0~CH15	4X0001	0
ADAM-5024 CH0	4X0003	0.000 mA
ADAM-5024 CH1	4X0005	0.000 mA
ADAM-5024 CH2	4X0007	0.000 mA
ADAM-5024 CH3	4X0009	0.000 mA
ADAM-5069 CH0~CH7	4X0011	0
ADAM-5051 CH0~CH15	3X0011	0
ADAM-5017UH CH0	3X0013	4.000 mA
ADAM-5017UH CH1	3X0015	4.000 mA
ADAM-5017UH CH2	3X0017	4.000 mA
ADAM-5017UH CH3	3X0019	4.000 mA
ADAM-5017UH CH4	3X0021	4.000 mA
ADAM-5017UH CH5	3X0023	4.000 mA
ADAM-5017UH CH6	3X0025	4.000 mA
ADAM-5017UH CH7	3X0027	4.000 mA

Chapter 7

7. Diagnosis and Troubleshooting

7.1. Error Notification

In chapter 4, we introduce how to write a program to control Advantech I/O modules. If the Advantech modules are correctly configured, it will show a green circle icon next to the device name in the device tree after performing command **Login** and **Start**. If it shows a red triangle , it means that I/O module encountered several errors while running.



7.2. Log Information

We can get log information from **Advantech CoDeSys** or **target machine**, i.e. ADAM-5560.

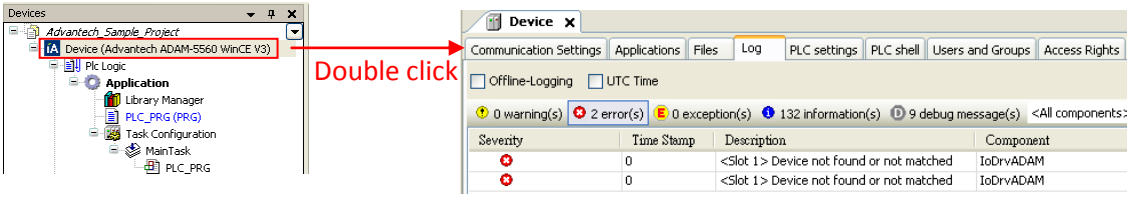
In Advantech CoDeSys development environment, double click the device name in the device tree to open **Device editor**. Select the **Log** dialog and it will display the log of the Advantech I/O module. A log entry line contains the following information:

Severity: There are four categories: warnings, errors, exceptions, information. The display of the entries of each category can be switched on or off by using the corresponding button from the bar above the listing. Each button always contains the current number of loggings in the respective category.

Time Stamp: Date and Time.

Description: Description of the event, for example “Device not found or not matched."

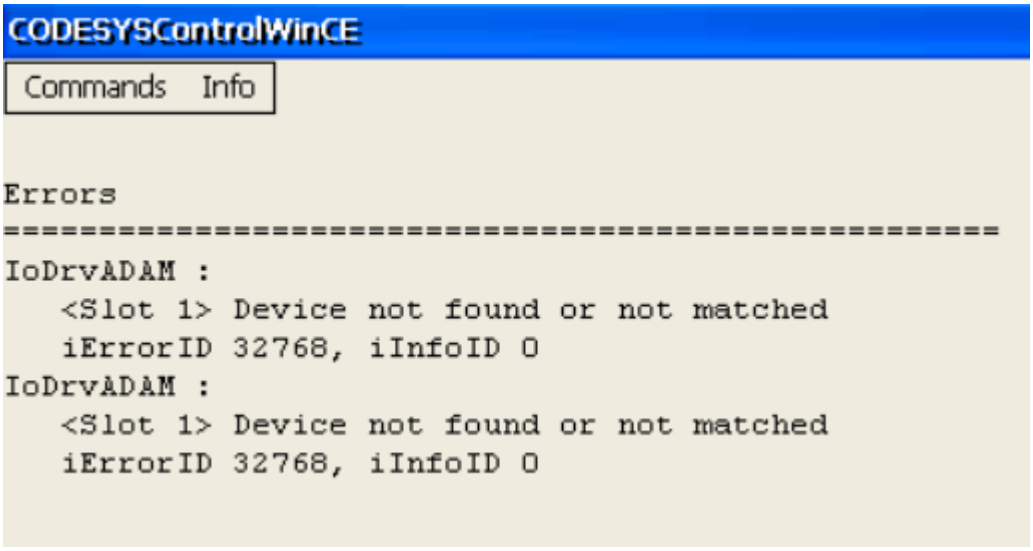
Component: ID and name of the component



On target machine, we can also get error ID from CoDeSys WinCE runtime.

In the ADAM-5560 environment, open runtime by double-clicking the runtime icon which is available on the lower-right corner of the desktop.

Get error ID by performing command Errors from the menu (**Info -> Errors**).



7.3. Error ID

Following table is the error ID for I/O modules.

Error ID	Description
0x8000	The module didn't exist or match the setting module. Make sure that the setting module matches for the device that is being plugged and check your module is plugged in target device appropriately.

0x8001	The system failed to open the module. Please close all programs and reboot. If the system cannot returns to normal condition or the error occurred, please contact Advantech for technical support.
0x8002	The system was unable to complete configuration. Please power-off the system and plug the module again. If the error occurred, please replace a new module and contact Advantech for technical support.
0x8003	The system failed to read value from the module. Please power-off the system and plug the module again. If the error occurred, please replace a new module and contact Advantech for technical support.
0x8004	The system failed to write value to the module. Please power-off the system and plug the module again. If the error occurred, please replace a new module and contact Advantech for technical support.
0x8005	For counter module, the system failed to start/stop counter. Please power-off the system and plug the module again. If the error occurred, please replace a new module and contact Advantech for technical support.
0x8006	For counter module, the system failed to clear counting value. Please power-off the system and plug the module again. If the error occurred, please replace a new module and contact Advantech for technical support.
0x8007	For counter module, the system failed to clear overflow flag. Please power-off the system and plug the module again. If the error occurred, please replace a new module and contact Advantech for technical support.
0x8008	For counter module, the system failed to clear alarm flag. Please power-off the system and plug the module again. If the error occurred,

	please replace a new module and contact Advantech for technical support.