RPAC-2658M User Manual

ICP DAS CO., LTD. would like to congratulate your purchase of our Win-GRAF PACs. The ease of the integration of the controller system and the power of the Win-GRAF software program combine to make a powerful, yet inexpensive industrial process control system.

ICP DAS Win-GRAF Linux-based PAC includes:

RPAC:	RPAC-2658M

ICP DAS Win-GRAF WinCE-based PAC includes:

ViewPAC:	VP-**x**208-CE7	[x: 2, 3, 4, 5, 6]
	VP-**x**238-CE7	[x: 1, 4, 6]
WinPAC:	WP-5238-CE7	
	WP-8**x**28-CE7	[x: 1, 4, 8]
	WP-9**x**28-CE7	[x: 2, 4, 8]
XPAC:	XP-8**x**38-CE6	[x: 0, 1, 3, 7]

Legal Liability

ICP DAS assumes no liability for any damage resulting from the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, neither for any infringements of patents nor other rights of third parties resulting from its use.

Trademarks & Copyright

Names are used for identification purposes only and may be registered trademarks of their respective companies. Copyright © 2020 by ICP DAS CO., LTD. All rights are reserved.

The version number and release date of this document are listed at the bottom of each page.
Users can download the latest document on the website:
https://www.icpdas.com/en/download/index.php?nation=US&kind1=&model=&kw=win-graf

Technical Service

- Win-GRAF Web site:
  https://www.icpdas.com/en/product/guide+Software+Development__Tools+Win-GRAF

- Win-GRAF Download Center:
  https://www.icpdas.com/en/download/index.php?nation=US&kind1=&model=&kw=win-graf

- Win-GRAF Demo program:
  Refer to Chapter 12 for more information about Win-GRAF demo programs.

If you have any problems, please feel free to contact us. Email: service@icpdas.com

Table of Contents

RPAC-2658M User Manual 1
Table of Contents 2
Chapter 1 Software Installation & Hardware Setting 7
1.1 Installing the Win-GRAF Workbench 7
1.2 Run the Win-GRAF Workbench 8
1.2.1 Win-GRAF Operating Environment 9
1.2.2 Win-GRAF Library Manager 10
1.3 Setting the Win-GRAF PAC's IP Address 11
1.4 Description of the Hardware 12
Chapter 2 A Simple Win-GRAF Program 13
2.1 Creating a New Win-GRAF Project 13
2.1.1 Creating a Template Project (Demo01) 13
2.1.2 Important Project Settings 15
2.2 Introduction of the Project 17
2.2.1 Demo01 - LD Program 17
2.2.2 Demo01 - Variables 19
2.3 Give it a Try 20
2.3.1 Declaring Variables 20
2.3.2 Creating an LD Program 22
2.3.3 Compiling the Program 27
2.3.4 Download the Program to PAC 28
2.3.5 Testing the Program 31
Chapter 3 Modbus Slave: Allow the SCADA/HMI Software to Access Win-GRAF Variables 33
3.1 To Enable Win-GRAF PAC as a Modbus TCP Slave 33
3.2 To Enable the Win-GRAF PAC as a Modbus RTU Slave 39
Chapter 4 Using "I/O Boards" Function 42
4.1 Ping_ip: Check the Connection of an Ethernet/ Internet Device 43
4.2 i_scale: Scale Conversion 45
4.3 RPAC_2000_LED: Control the L1 and L2 LED on the RPAC-2658M 47
4.4 RPAC_PAC_state: Detect the LAN state of the RPAC-2658M 48
Chapter 5 Modbus Master: connecting to Modbus Slave Devices 49
5.1 Enabling Win-GRAF PAC as a Modbus RTU/ASCII Master 49
5.1.1 Read DI data 52
5.1.2 Write DO Data 55
5.1.3 Read AI Data 57

5.1.4 Write AO Data (16-bit) 60
5.1.5 Write AO Data (32-bit) 62
5.1.6 To Disable/Enable the Modbus RTU/ASCII Master Port 64
5.2 Enabling the Win-GRAF PAC as a Modbus TCP/UDP Master 65
5.2.1 Connecting ET-7000 Series I/O Module 69
5.2.2 Connecting the ET-7060 (6 DI, 6 Relay Output) 72
5.2.3 Connecting the ET-7018Z (10 AI) 74
5.2.4 To Disable/Enable the Modbus TCP/UDP Master Port 75
5.3 Connecting a Modbus TCP Slave Device with two IP addresses 76
5.4 Accessing multiple Modbus RTU Slave via the tGW-700 Gateway 80
5.4.1 Configuring the tGW-700 (Modbus TCP to Modbus RTU/ASCII Gateway) 80
5.4.2 Accessing the LC series Module via the tGW-700 Gateway 83
5.4.3 Test the Project (demo_tgw725.zip) 87
Chapter 6 Retain Variable and Data Storage 88
6.1 Using the RETAIN_xxx Function 89
6.1.1 RETAIN_VAR: Retain a Variable 90
6.1.2 RETAIN_ARY: Retain an Array Variable 91
6.1.3 RETAIN_FLAG_SET/GET/CLR (Set/Get/Clear the Retain Flag) 92
6.2 Save Retain Variables and Data to a File 94
6.3 Save the Retain Data to a FRAM 98
6.3.1 EEP_READ (Read a Value from the FRAM) 99
6.3.2 EEP_WRITE (Write a Value to the FRAM) 99
Chapter 7 Exchange Data between PACs (Data Binding) 100
Chapter 8 Connecting DCON I/O Modules 104
8.1 Setting the "DCON" I/O Board 105
8.2 Using I/O Function Blocks 107
8.2.1 “D_7065” Function Block 108
8.2.2 “D_7018Z” Function Block 109
8.2.3 “D_7083” Function Block 111
8.2.4 “D_87084_FREQ” Function Block 112
8.2.5 “D_87084_CNT4” Function Block 113
8.2.6 “D_87084_CNT8” Function Block 114
8.2.7 “DL_100T485” Function Block 115
8.2.8 “D_GPS721” Function Block 116
8.3 Using the Count Function for I-87082W, I-87084W, I-7083, and I-7080 Modules 118
8.3.1 COUNTER_START 118
8.3.2 COUNTER_STOP 120
8.3.3 COUNTER_GET 121
8.3.4 COUNTER_STATE 122
8.3.5 COUNTER_RESET 123

Chapter 9 On-Line Change 124
9.1 Limitations of "On-Line Change" 124
9.2 Using "On-Line Change" 126
Chapter 10 Data/Type Conversion and Using the PAC Time 129
10.1 AI Data Conversion (i_scale) 129
10.2 AO Data Conversion (i_scale) 130
10.3 Type Conversion Functions (ANY_TO_xxx) 132
10.4 BCD Conversion (BIN _TO_BCD, BCD_TO_BIN) 133
10.5 Pack/Unpack Integer or Boolean 134
10.6 Pack/Unpack BYTE, WORD, DWORD 136
10.7 Unpack Variable to Byte Array or Pack Byte Array into Variable 139
10.8 Get/Set the PAC Time 141
Chapter 11 Commonly Used Tools and Useful Tips 142
11.1 Upgrade Win-GRAF Libraries 142
11.2 Upgrade Win-GRAF Driver 143
11.3 Spy List 145
11.4 Backup/Restore a Win-GRAF Project 147
11.5 Soft Reboot on the PAC 149
11.6 Using ST Syntax in an LD and FBD Program 150
11.7 Apply a Recipe on the PAC 151
11.8 Functions and Function Blocks Supported by Win-GRAF PACs 153
11.9 Upload the Win-GRAF Source Code 155
11.10 Set Up the PAC Password 157
11.11 Using Function or Function Block in the ST Program 159
11.12 Protect Your Win-GRAF Program to Avoid Unauthorized Use 160
Chapter 12 Description of Win-GRAF Demo Projects 164
12.1 The List of Demo Programs 165
12.2 Timer Operations 167
12.2.1 Start, Stop and Reset the Timer 167
12.2.2 Periodic Operations 168
12.2.3 Detect the Steady ON or Steady OFF Signal 170
12.2.4 Keep Outputting ON for Some Time after Triggering 171
12.3 Operations of Serial Port Communication 172
12.3.1 Send a String via a COM Port 173
12.3.2 Request-Reply Communication via a COM Port 174
12.3.3 Wait for a Remote Device to Send Data to a COM Port 176
12.3.4 Reply Data to the Remote Device Periodically via a COM Port 178

12.4 Read/Write Data from/to a File on the PAC 179
12.4.1 Write Data to a File on the PAC 180
12.4.2 Read Data from a File on the PAC 182
12.4.3 Data Logging 185
Chapter 13 Using a C Program to Access Win-GRAF Variables 190
13.1 Publishing the Win-GRAF Variable for C Program 190
13.2 Downloading and Compiling the C Program 192
13.3 Accessing Win-GRAF Variables 195
13.4 Descriptions of Functions in the "Quicker.a" 196
13.4.1 R/W a Boolean Variable 196
13.4.2 R/W an Integer Variable 197
13.4.3 R/W a REAL Variable 199
13.4.4 R/W a String Variable 200
Chapter 14 Redundancy 201
14.1 What Kinds of Data will Automatically Send to the Passive-PAC? 203
14.2 Redundant System (Rotary Switch: 7 & 9) 204
14.2.1 The Architecture of Win-GRAF Near-Field Redundant System 204
14.2.2 Set up the Redundant System 207
14.2.3 Test the Redundant Project (demo_rdn_2) 210
14.3 Description of Win-GRAF Demo Projects 212
14.3.1 [Important] "I/O Board" Settings (i_redundancy and i_redundancy_rs485) 212
14.3.2 Declaring Variables (demo_RDN_2) 216
14.3.3 Introduction of the "demo_RDN_2" Project 217
14.4 The Description for Other Redundant Projects 219
14.4.1 Test the Redundant Project (demo_rdn_3, demo_rdn_1) 219
14.4.2 Introduction of the "demo_RDN_4" Project 223
14.4.3 Test the Redundant Project (demo_rdn_4) 228
Chapter 15 Schedule Control 235
15.1 Install the Schedule-Control Utility and Restore the Win-GRAF Project 235
15.2 Introduction of the “demo_schedule” Project 236
15.2.1 "I/O Boards" Setting (Schedule) 237
15.2.2 Public the Variable Data (Data Binding) 238
15.3 Description of the Schedule-Control Utility Example 239
15.4 Testing the Schedule-Control on the PAC 241
15.5 How do I Use the Schedule-Control Utility 243
15.5.1 Address for Control Variables (BOOL, DINT, and REAL) 243
15.5.2 Configure the Target 244
15.5.3 Configure the Season 246

15.5.4 Configure the Normal Day / Holiday / Special Day 248
15.5.5 Configure the Schedule Time 250
15.5.6 Save and Send the Configuration File 251
15.5.7 Time Synchronization 252

Chapter 16 Win-GRAF SMS Function 253

16.1 "GSM_Open", "Send_SMS" and "Read_SMS" Functions 254
16.2 Introduction of the "Demo_SMS" Project 257
16.3 Test for SMS Messaging 260

Appendix A Data types and Ranges 264
Appendix B Troubleshooting while On-Line the PAC 265
Appendix C Allows the Win-GRAF to Connect via PAC's Serial Port 267
Appendix D Pin Assignment of PAC's Serial Ports 271

Chapter 1 Software Installation & Hardware Setting

1.1 Installing the Win-GRAF Workbench

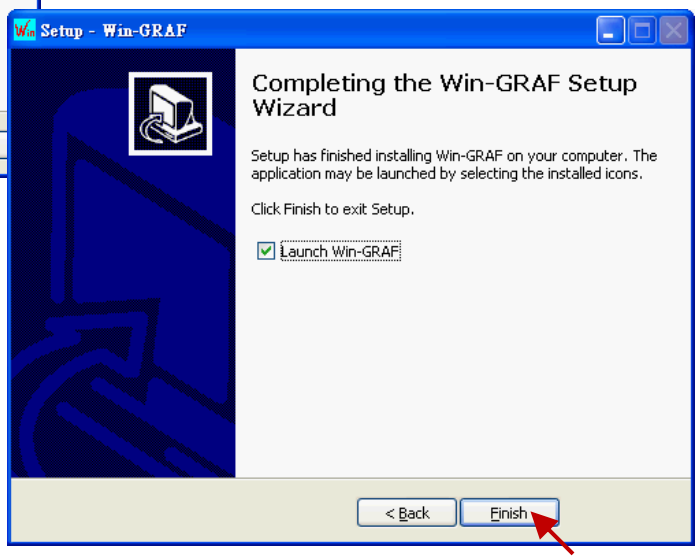
Before installing the Win-GRAF Workbench, check the installation environment on your PC.

System requirements:

- **O.S.**: Windows 7, Windows 8, Windows 10 (32-bit or 64-bit)
- **Microsoft .Net Framework 3.5** (Download it on the Microsoft web site: https://www.microsoft.com/zh-tw/download/details.aspx?id=22)
- **RAM:** 1 GB minimum (Recommended: 2 GB or more)
- **Available hard-disk space:** 200 MB minimum

Installation Steps:

1. Download the Win-GRAF workbench installation from the ICPDAS website. (https://www.icpdas.com/en/download/show.php?num=711, Win-GRAF-setup-ver-1.xx.zip)
2. Double-click the "Win-GRAF-setup-ver-x.xx.exe" setup execution file and follow the execution steps.



1.2 Run the Win-GRAF Workbench

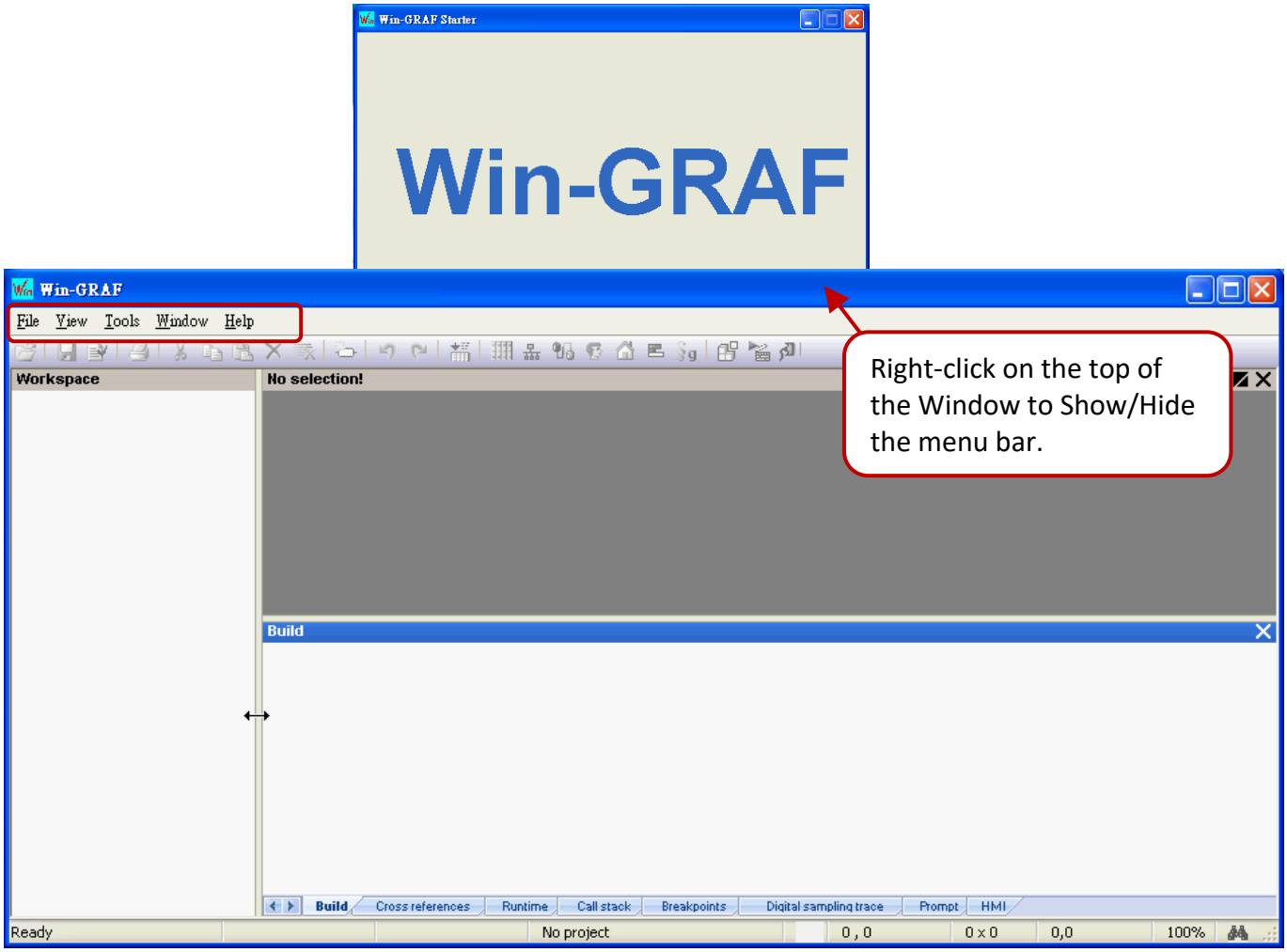
Before running the Win-GRAF Workbench, make sure the USB license Key (Win-GRAF Dongle) is plugged into your PC, otherwise, the workbench will run in demo mode.

Limitations in demo mode:

1. Applications are limited to 40 I/Os.
2. The code generated by the compiler stops after 15 minutes.
3. The simulation stops after 15 minutes.

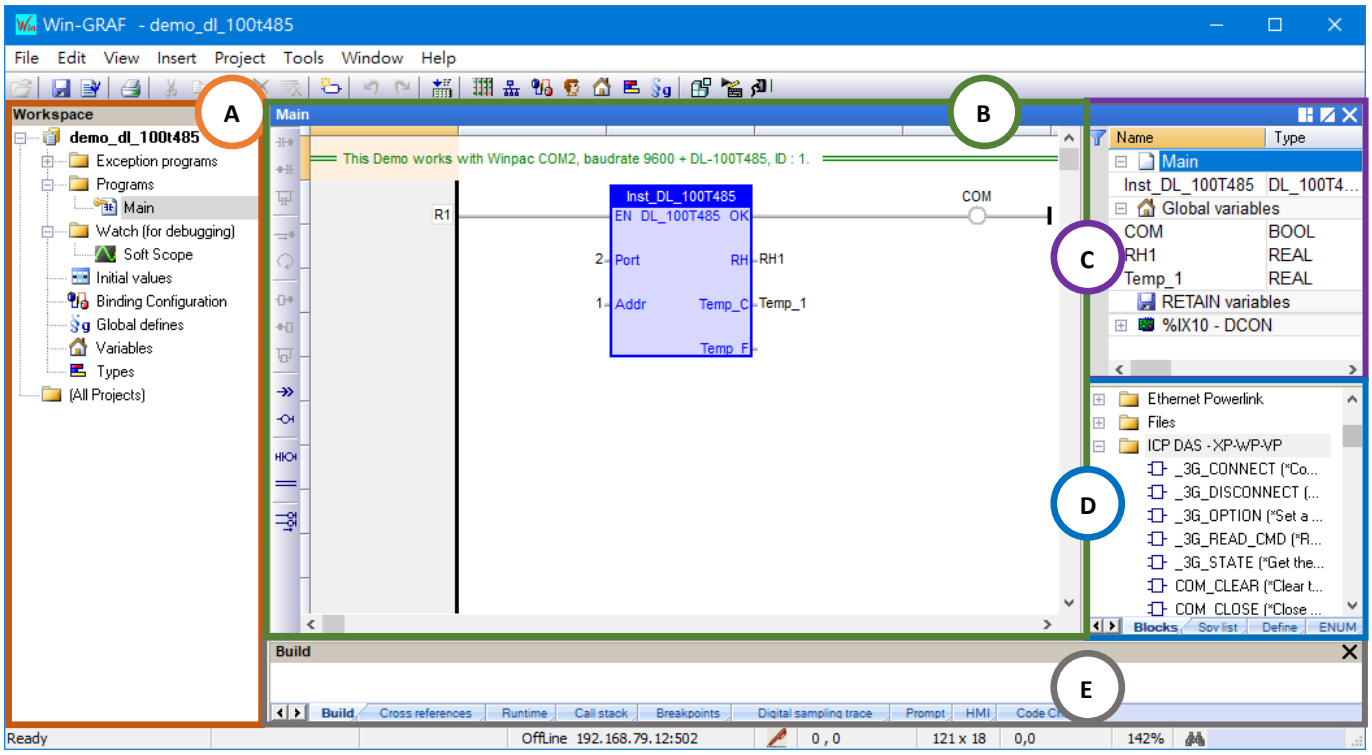
Start the software by clicking the “Win-GRAF” in the Start menu.

Note: If the USB license Key is plugged after starting the software, restart the Win-GRAF Workbench to run the fully licensed version.



1.2.1 Win-GRAF Operating Environment

The main user interface (UI) of the Win-GRAF workbench is shown below.



A. The **Workspace**:
   The Workbench enables you to edit several projects in the same workspace. Each project and relevant information are stored as a folder on the disk. The project list is stored in a file suffixed by ".W5L" that only stores the list of project folders and few configuration data.

B. The **Editor** Area:
   Double-click the item listed in the Workspace to display the editing page. E.g., program and variables.

C. The **Variable Editor** enables the declaration of variables and instances quickly during development.

D. The **Blocks** pane provides some available functions and blocks you can use in programs.
   The **Spy list** pane enables the quick dynamic view of variables during debugging.

E. The **Output** window reports messages and provides most of the diagnostic tools

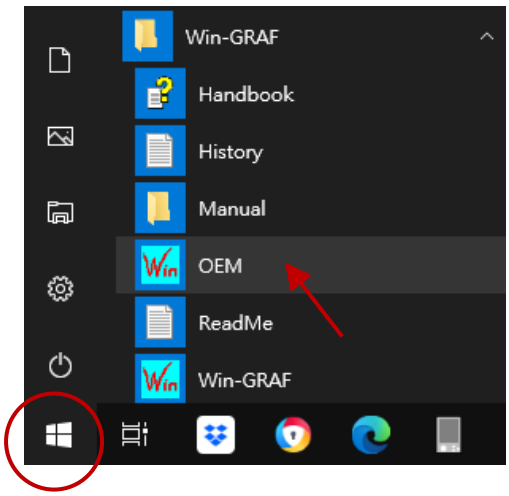
Tips:

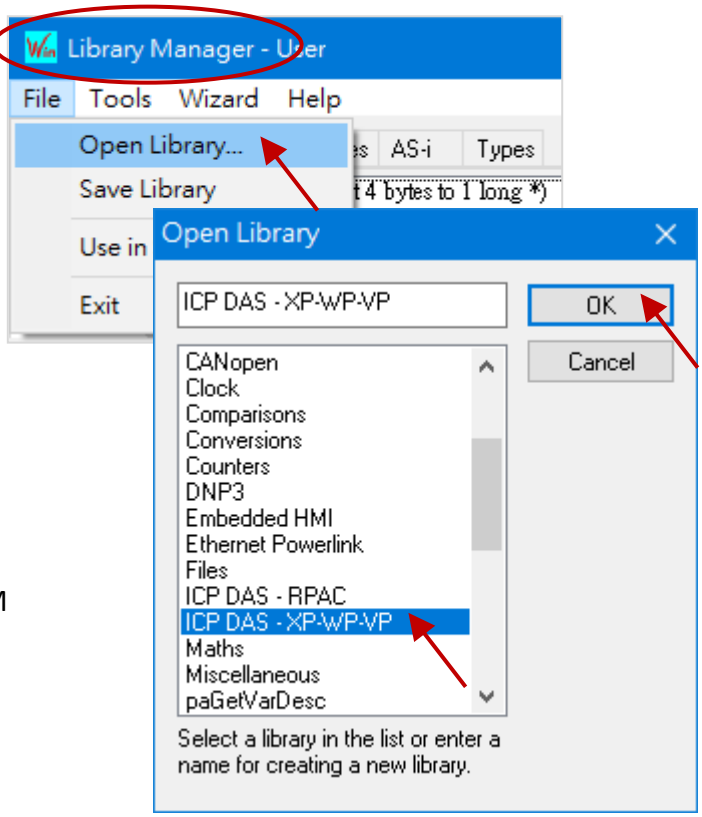
1. Press the "**F1**" key where you want to get help with the function.
2. To resize a window, click and drag the side or corner of the window to change its size.
3. Click the menu command **View** - **Output** or **Infos Tab1** or **Infos Tab2**" if the pane is closed carelessly.
4. Click the menu command **Help** - **Language** to change the language. The setting will be applied after automatically restarting the Win-GRAF workbench.

1.2.2 Win-GRAF Library Manager

To help with Functions and Function Blocks, open the Win-GRAF Library Manager.

1. Click the **Start** menu, and click **OEM** in the **Win-GRAF** folder.
2. Click **File** and **Open Library** in the Library Manager window then select "ICP DAS – XP-WP-VP" or "ICP DAS – RPAC" and click "OK".

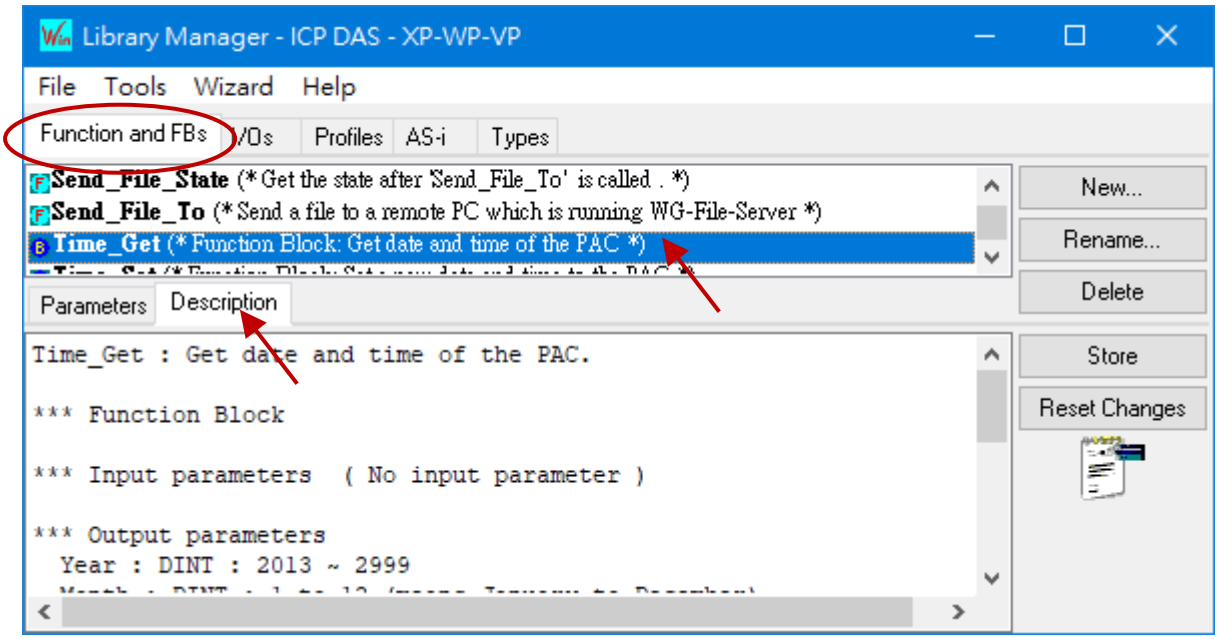




Note:

Close Workbench before opening the OEM if you want to edit the library.

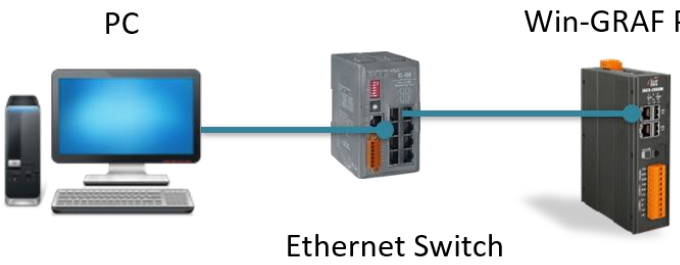
3. Click any entry under the "Function and FBs" tab and click the "Description" to view the description of this Function or Function Block.



1.3 Setting the Win-GRAF PAC's IP Address

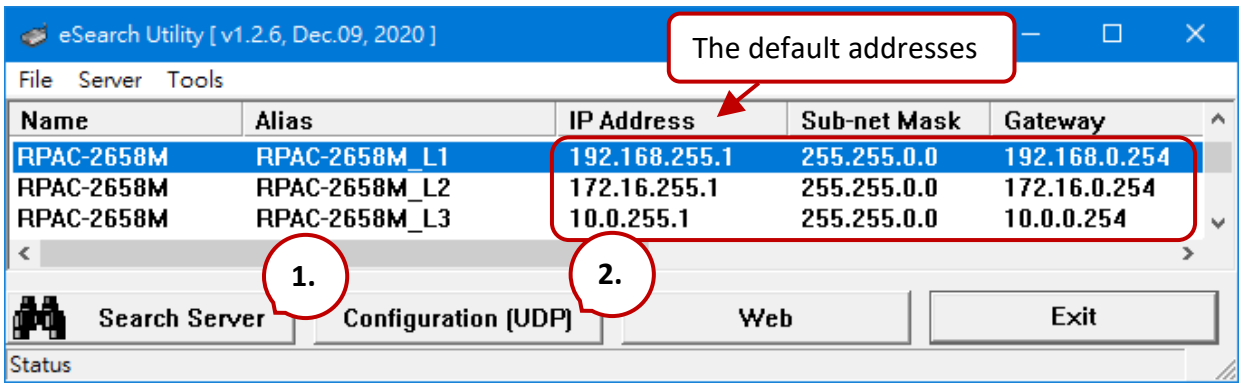
Follow the steps below to configure the IP, Mask, and Gateway address of the RPAC-2658M.

Hardware Wiring Diagram

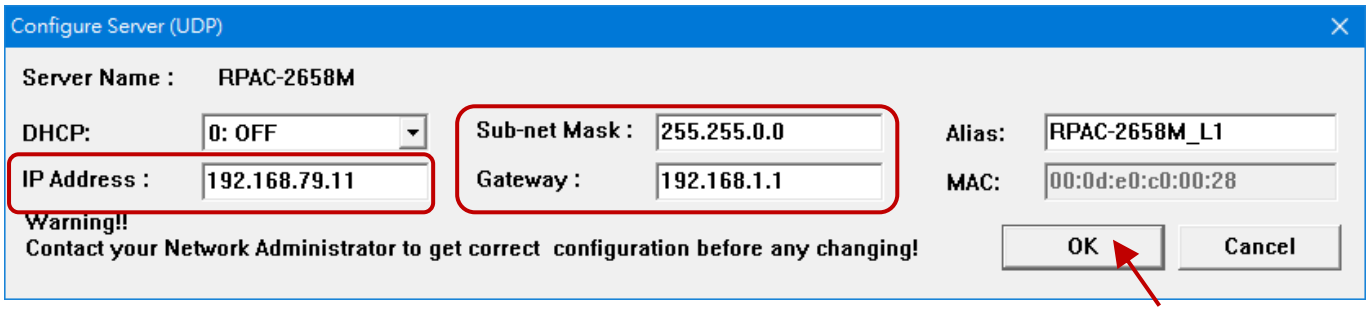


Notice:
Recommended to connect to the PC/ software (e.g., Win-GRAF) via **LAN1** with the same network segment.

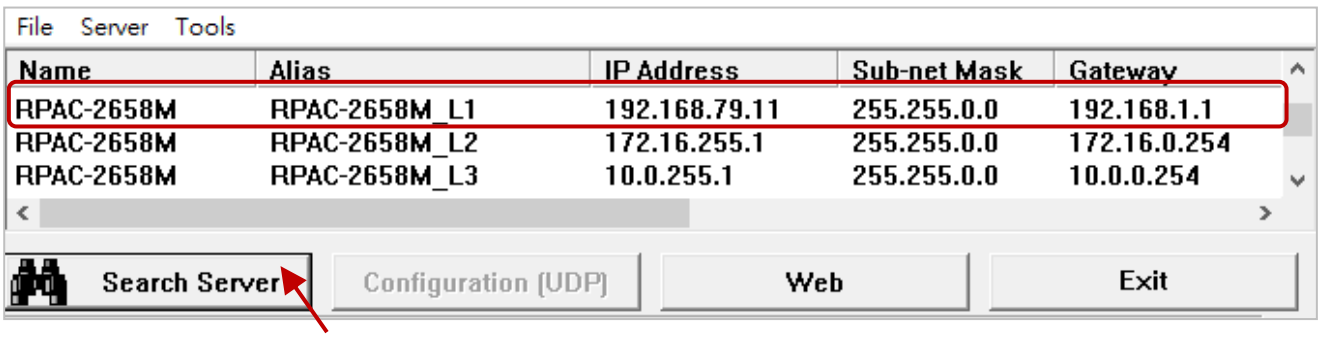
1. Download and install the "eSearch Utility" software on the ICP DAS website.
2. Click the **Search Server** button to search devices on the network. Three IP settings will be displayed for one RPAC.



3. Click the **Configuration** button to set the desired field and click the **OK** button.

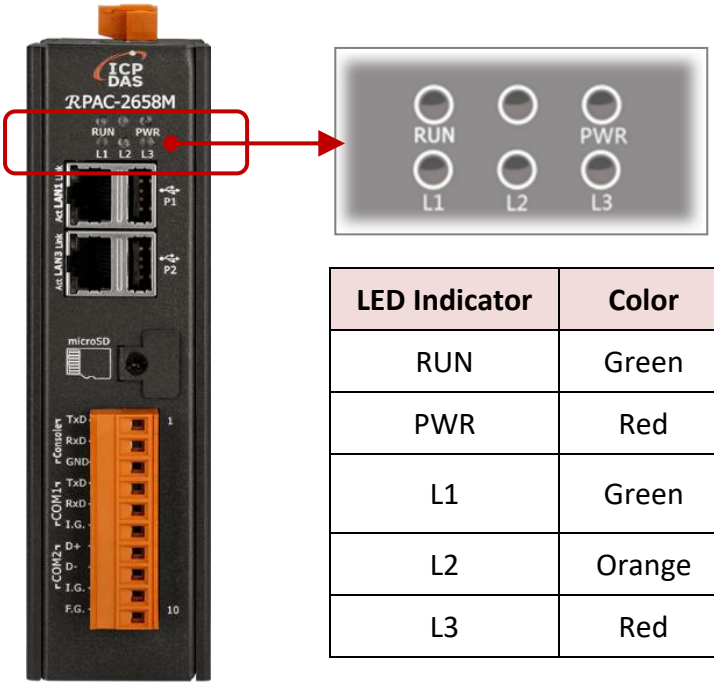


4. Search the device again to refresh the status. Note that LAN1, LAN2, and LAN3 must be set as different network segments.



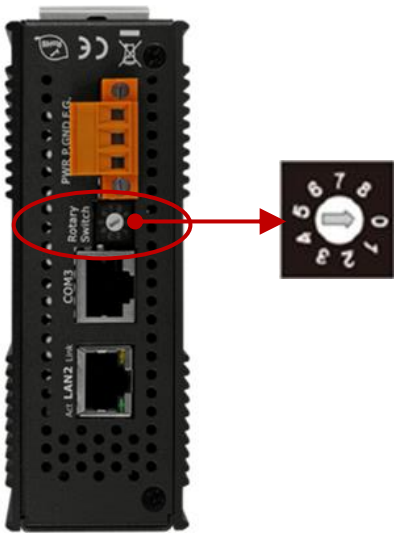
1.4 Description of the Hardware

LED 指示燈



LED Indicator	Color	Description
RUN	Green	Power on and OS is running (Flashing state)
PWR	Red	Power is ON
L1	Green	User programable LED & hardware error indicator
L2	Orange	User programable LED
L3	Red	Redundancy status indicator (Active PAC)

旋轉開關 (Rotary Switch)



Rortary Switch	Description
0	Normal Mode
1	Don’t start user application
2	LAN1: DHCP Mode
3	LAN1: Static IP Mode (IP: 192.168.255.1, Mask: 255.255.0.0)
4	Firmware Update Mode

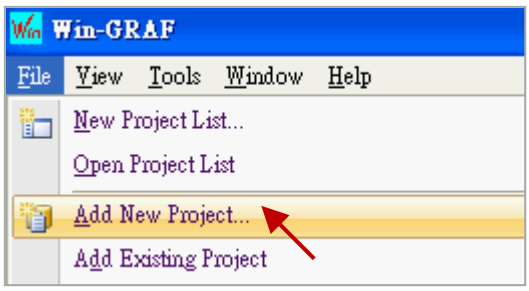
Chapter 2 A Simple Win-GRAF Program

2.1 Creating a New Win-GRAF Project

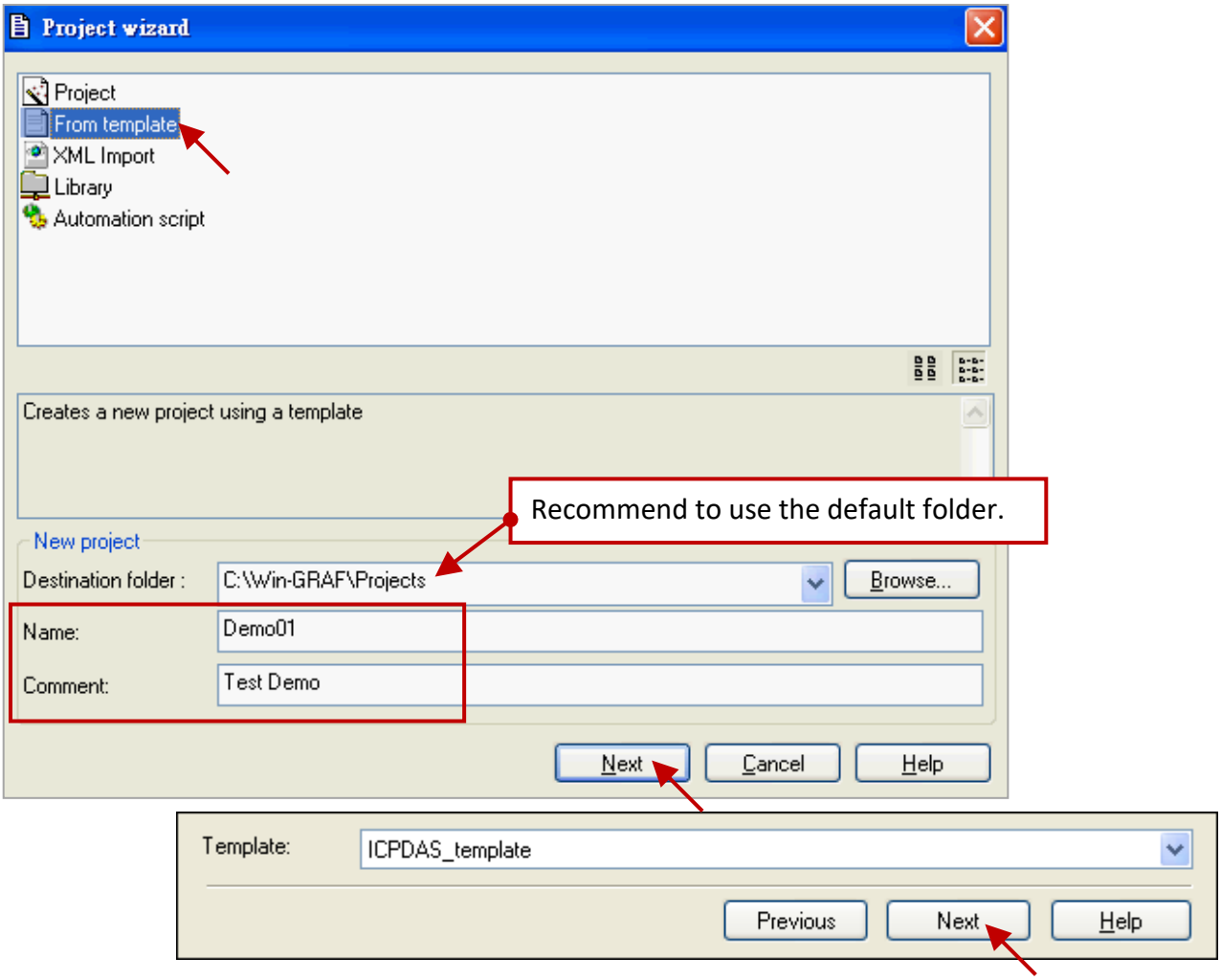
The following sections will introduce you to a simple template project that used to get/set (read/write) the Win-GRAF PAC's system time. Follow the steps below to complete this demo program.

2.1.1 Creating a Template Project (Demo01)

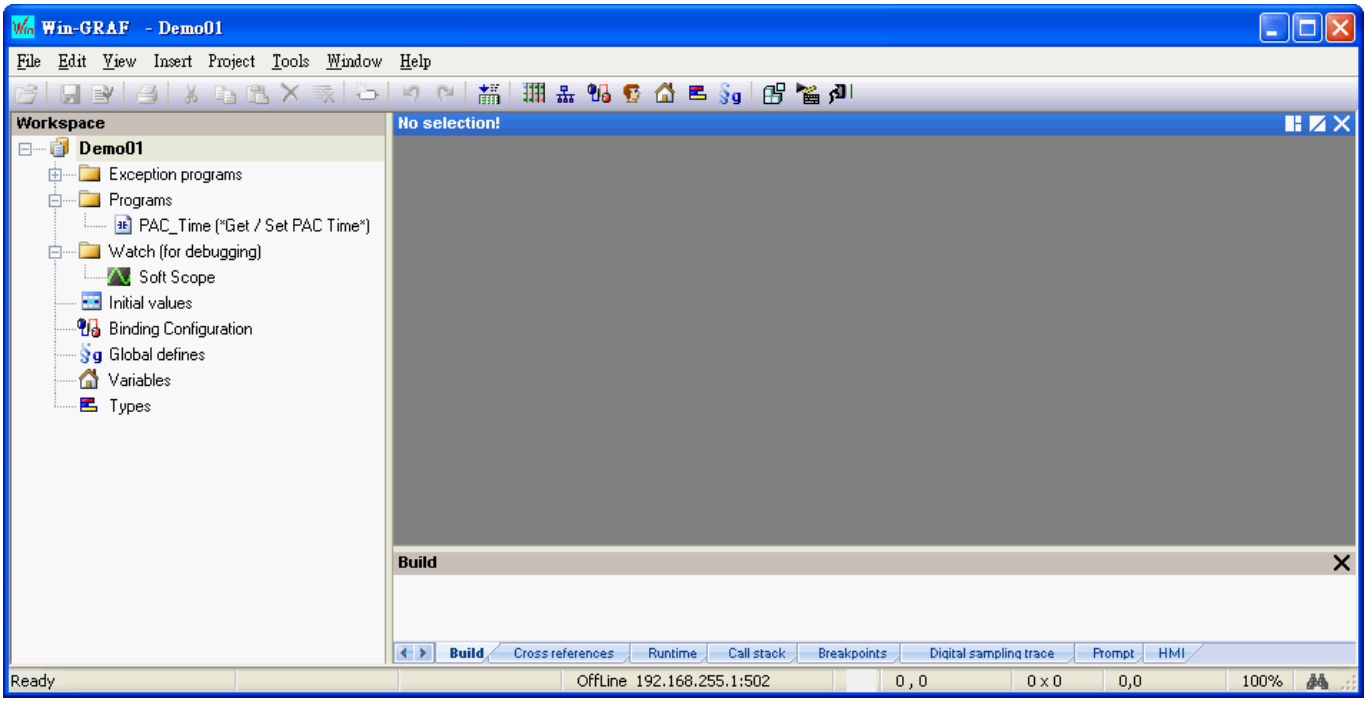
1. Run the Win-GRAF Workbench (refer to Section 1.2), and click the menu command "File - Add New Project...".



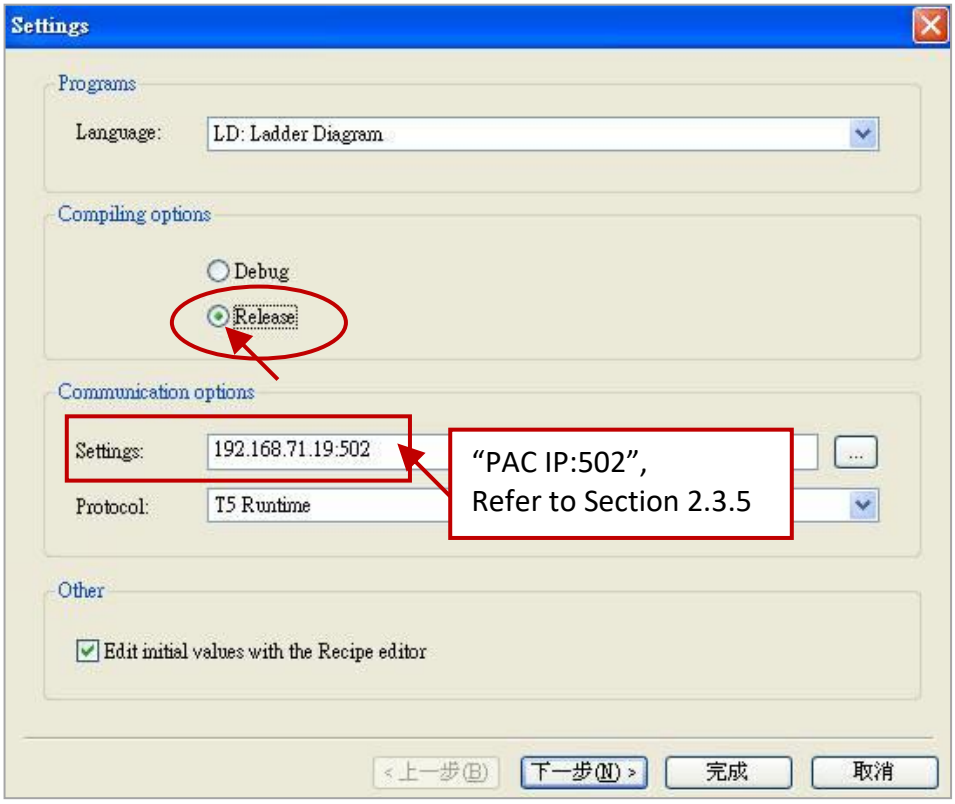
2. Click "From template" to create a project from a template, enter a project name (e.g., "Demo01") in the Name field and add a simple note in the Comment field, then click "Next". By default, it will show an "ICPDAS_template" option provided by Win-GRAF Workbench, just click "Next" to continue.



3. Now, you have created the “Demo01” template project.



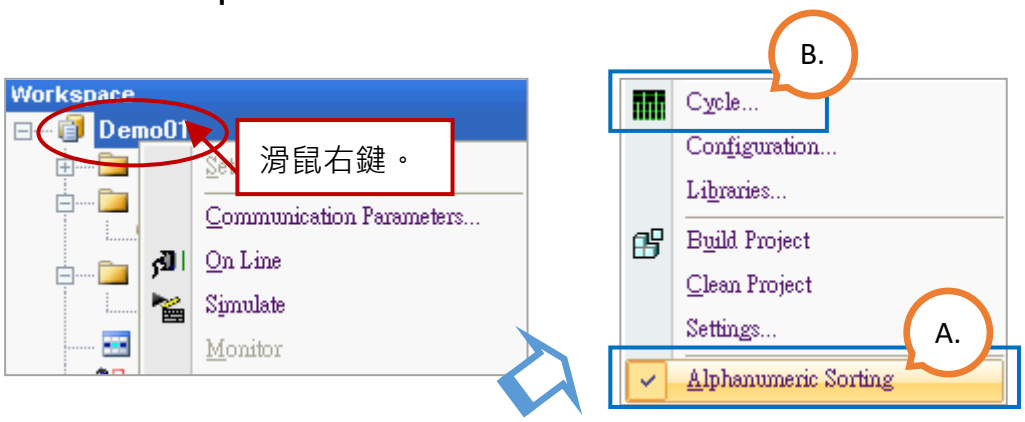
Note: If you change to click "Project" in step2 to create a project, select the "Release" in the “Compiling options" setting and click "OK". The rest of the settings items can be set in the subsequent chapters.



2.1.2 Important Project Settings

Two important settings must be done after creating the project.

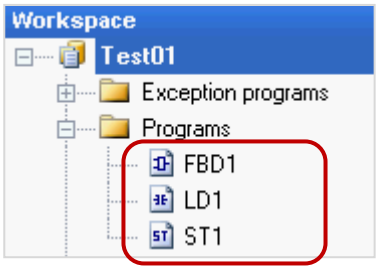
1. Program Execution Sequence



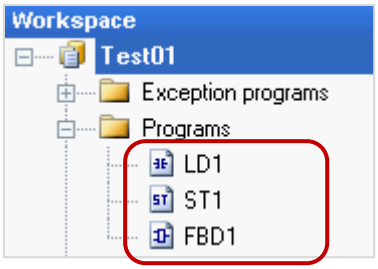
A. Check Program Execution Sequence

Right-click the project name (e.g., "Demo01"), select the "Alphanumeric Sorting" option to display the program name in alphanumeric order; unselect the option to display the name in execution order.

(In alphanumeric order if selected)

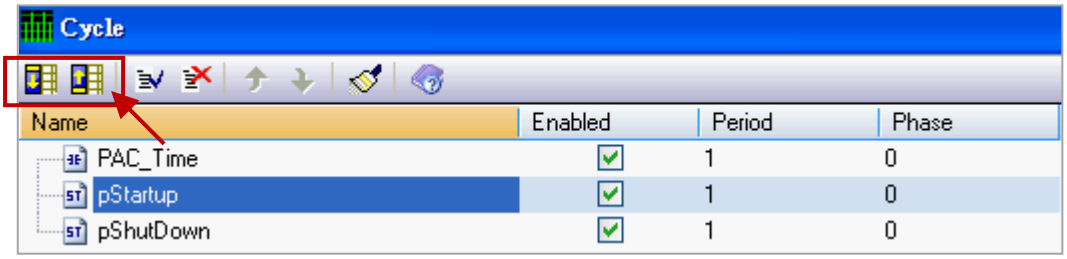


(In execution order if unselected)



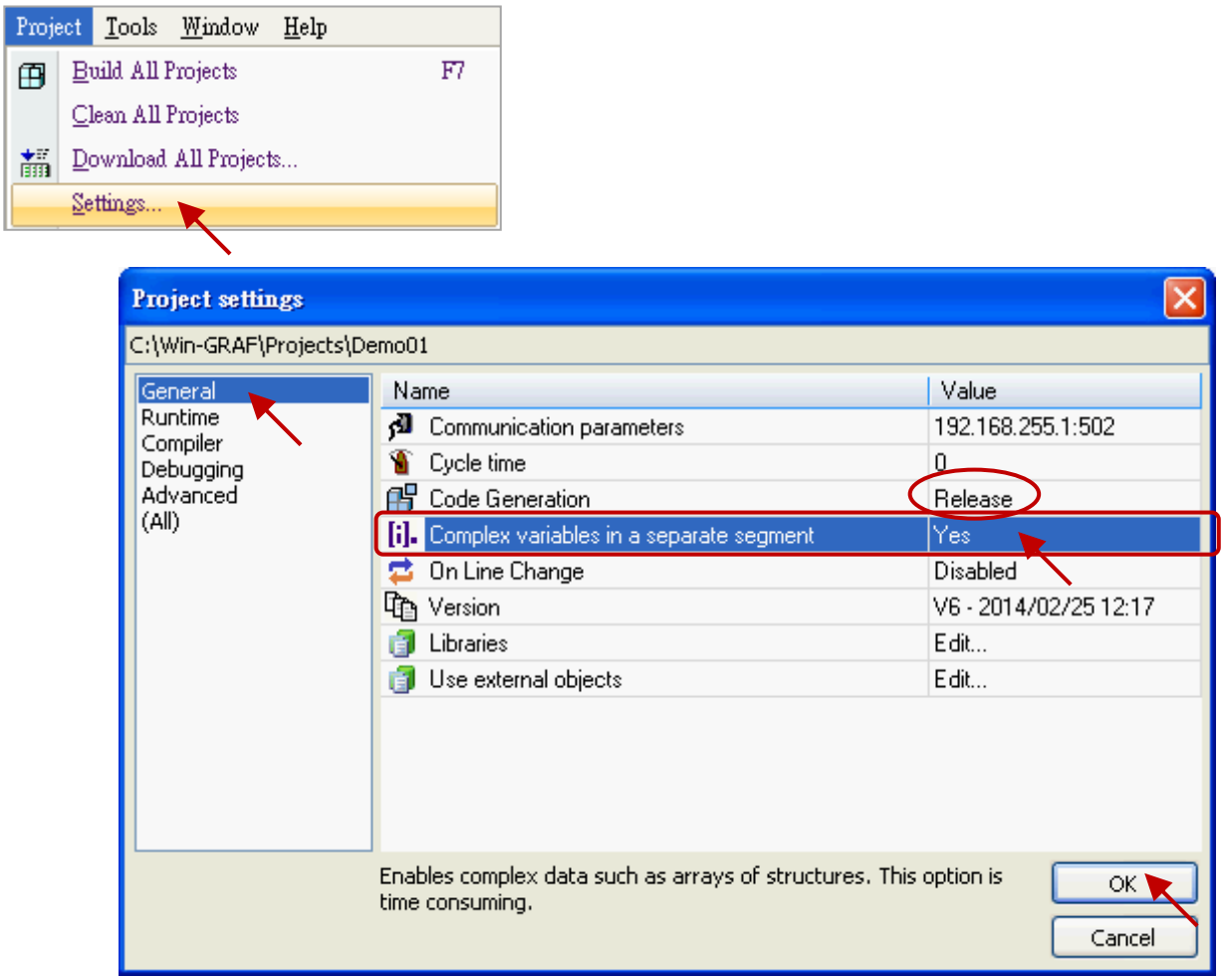
B. Modify Program Execution Sequence

Right-click the project name (e.g., "Demo01") and click "Cycle" to open the settings window, then click the "Move Up" or "Move Down" button to change the order.



2. **Enable the "Complex variables in a separate segment" option**

   If the complex variables such as Array, Structures, or FB Instances will be used in the project, click the menu command “Project - Settings..." to open the Project settings window. Click the "General" option and set the "Complex variables in a separate segment" to "Yes" and then click "OK".



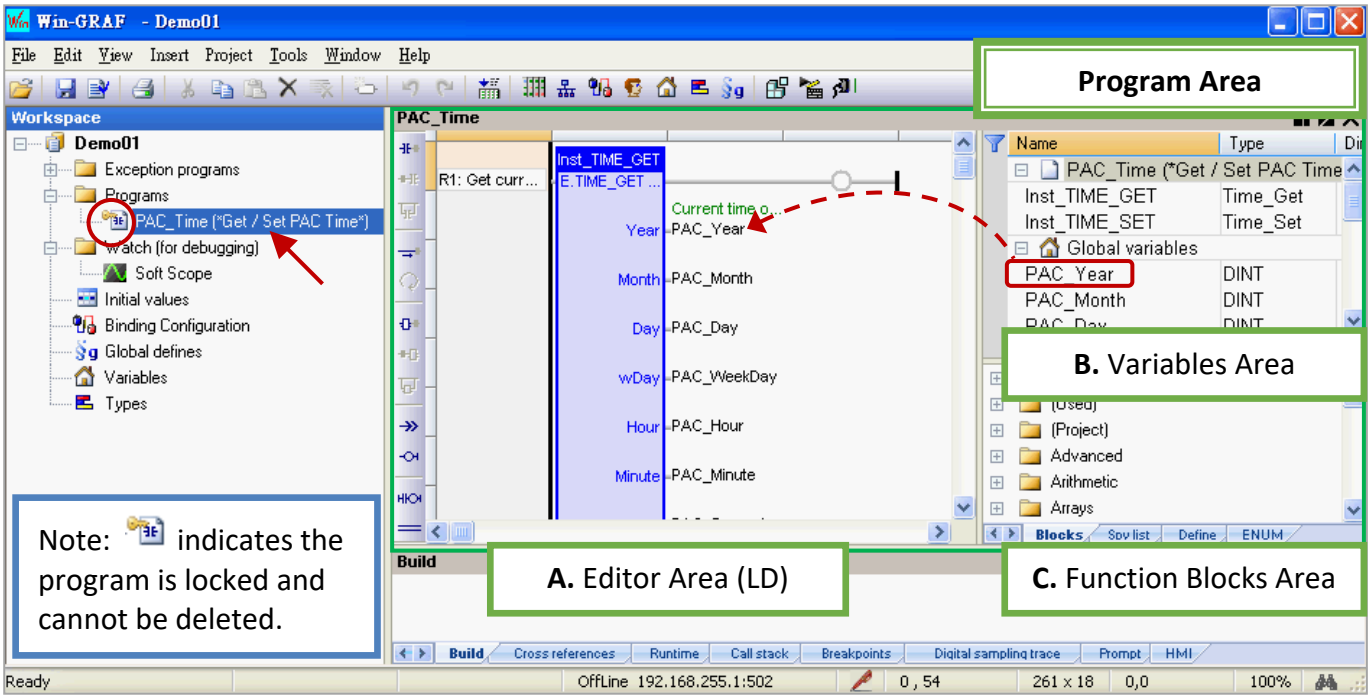
Note:

1. For more details, click the menu command **Help - Topic** and search the keyword "Complex variables in a separate segment".
2. Make sure the **Code Generation** is set to **Release**.

2.2 Introduction of the Project

2.2.1 Demo01 - LD Program

This program is used to read/write the Win-GRAF PAC's system time. In the Workspace, double-click the LD program name (i.e., "PAC_Time") to open all relevant windows. As the screenshot below, the Program Area has three main parts:

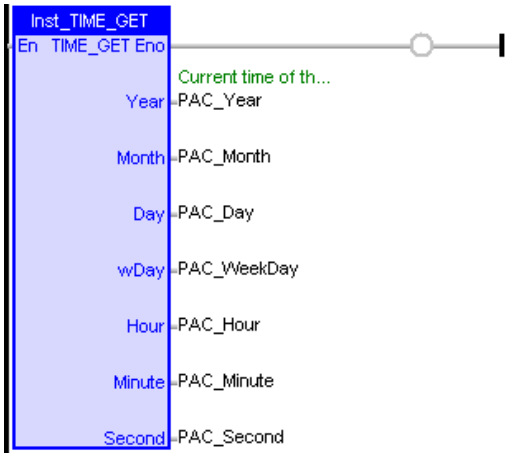


Tips: Click on the Editor Area and press the "+" or "-" key to zoom in or zoom out the content.

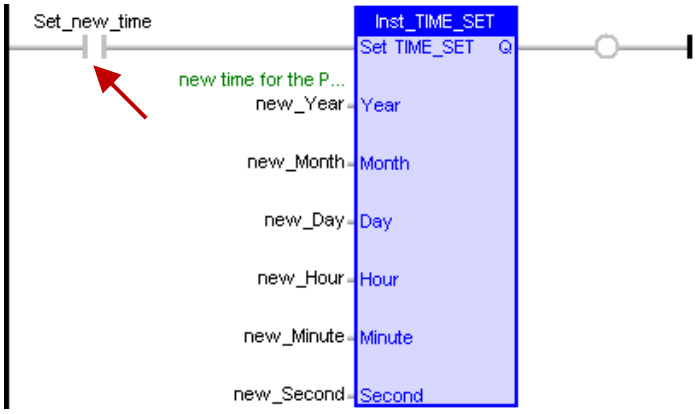
A. Editor Area:

This area allows you to edit the program, click the object button on the left side for programming, and assign variables by drag and drop them from the variable area.

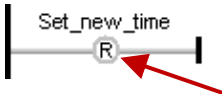
R1: Get current time of the PAC



R2: Set "Set_new_time" to "True" to set the new time



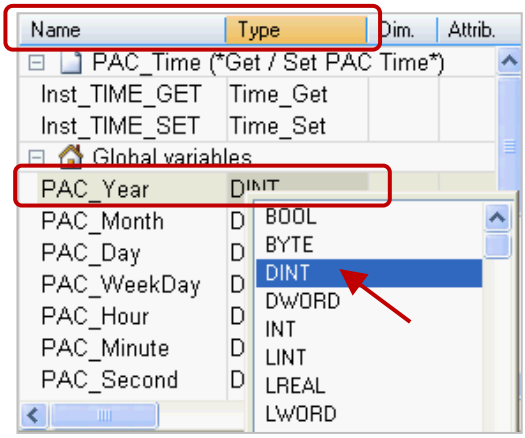
R3: Reset it to "False"



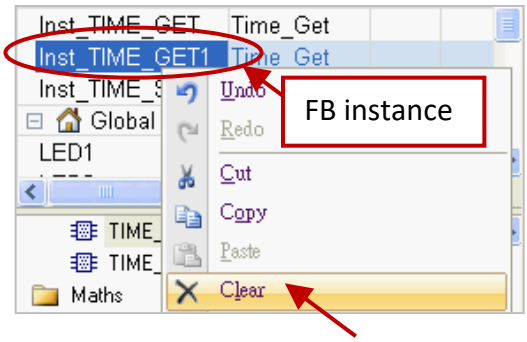
Note: You can press the SPACE bar to change its type (/, S, R), and press the F1 key for more details.

B. Variables Area:

This area shows FB instances and variables that are used in this program. Double-click any “Name” or “Type” entry to modify its name or data type, then click “Enter” to complete the setting.

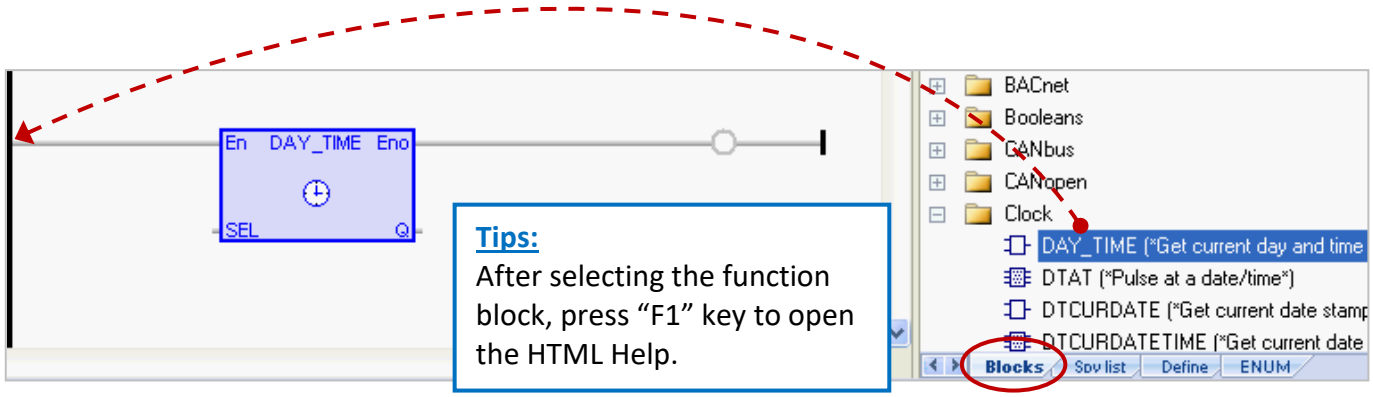


Note: For using function block (FB) in the program correctly, the instance name “Inst_xxx..” will be automatically added when a function block is added. For safety reasons, this FB instance name will not be automatically deleted even if the function block has been removed in the program. If it is necessary, right-click the FB instance name and select “Clear” to manually remove it.



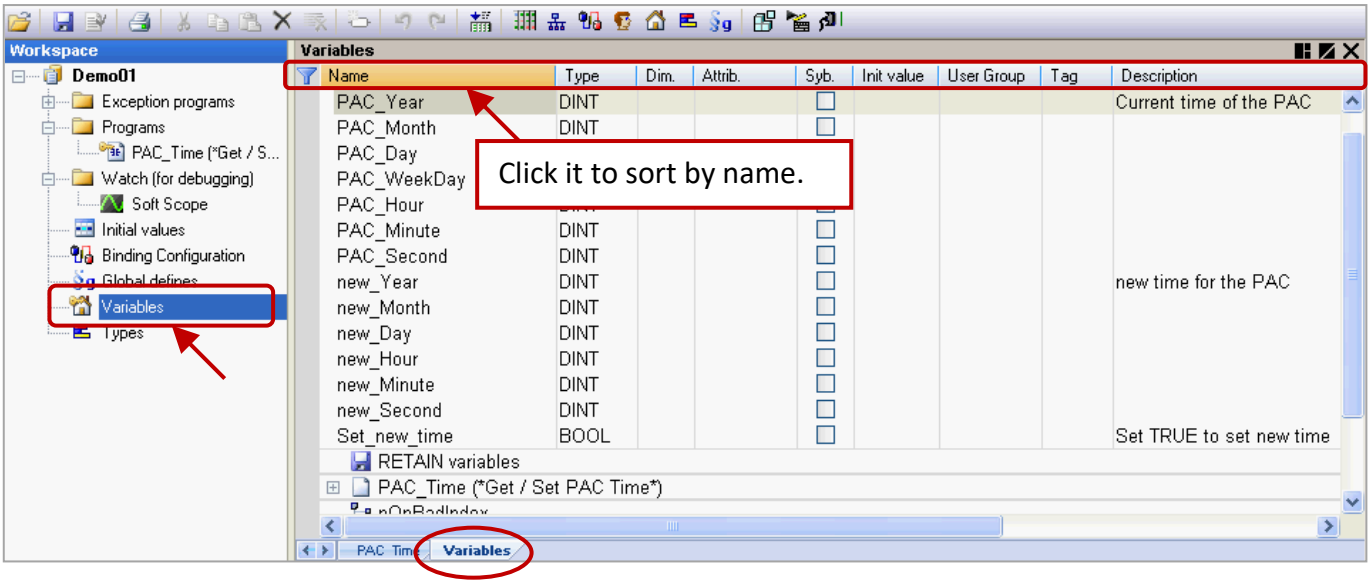
C. Function Blocks Area:

In the “Blocks” tab, drag and drop the specific function blocks to the editor area.

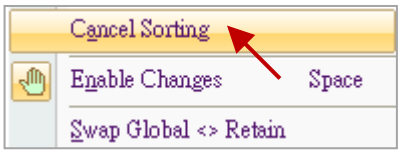


2.2.2 Demo01 - Variables

In the workspace, double-click the “Variables” item to open the Variables window. The following screenshot shows all the needed and defined variables in this “Demo01” project.



Tips: In the Variables window, click any field name (e.g., “Name”) to sort data. Also, right-click anywhere and select the “Cancel Sorting” option to back to the original order.



Field description: (Press the “F1” key to look up the details)
Double-click any entry in a column to set or modify the data.

Name:	A valid variable name starts with a letter (e.g., "A to Z" or "a to z") followed by any number of letters, numbers (e.g., "0 to 9"), or an underscore (i.e., “_”).
Type:	Data type. (Refer to Appendix A for the value range)
Dim.:	To specify the range of an array. (E.g., enter “10”, which means the use of the Counter [**0**] to [**9**]).
Attrib.:	Double-click this field item to set it to “Read Only” which means users can only read this variable but cannot modify it.
Syb.:	The function will be available soon. The selected variable name will be embedded in the application that can be used to access parameter data by call the name in the PAC.
Init value:	To set the initial value of the variable.
User Group:	All the variables can be divided into some groups (e.g., “Group1”, “Group2”) and it is convenient for users to look up or search these variables.
Tag:	To enter a nickname for the variable.
Description:	To enter a simple note for the variable.

2.3 Give it a Try

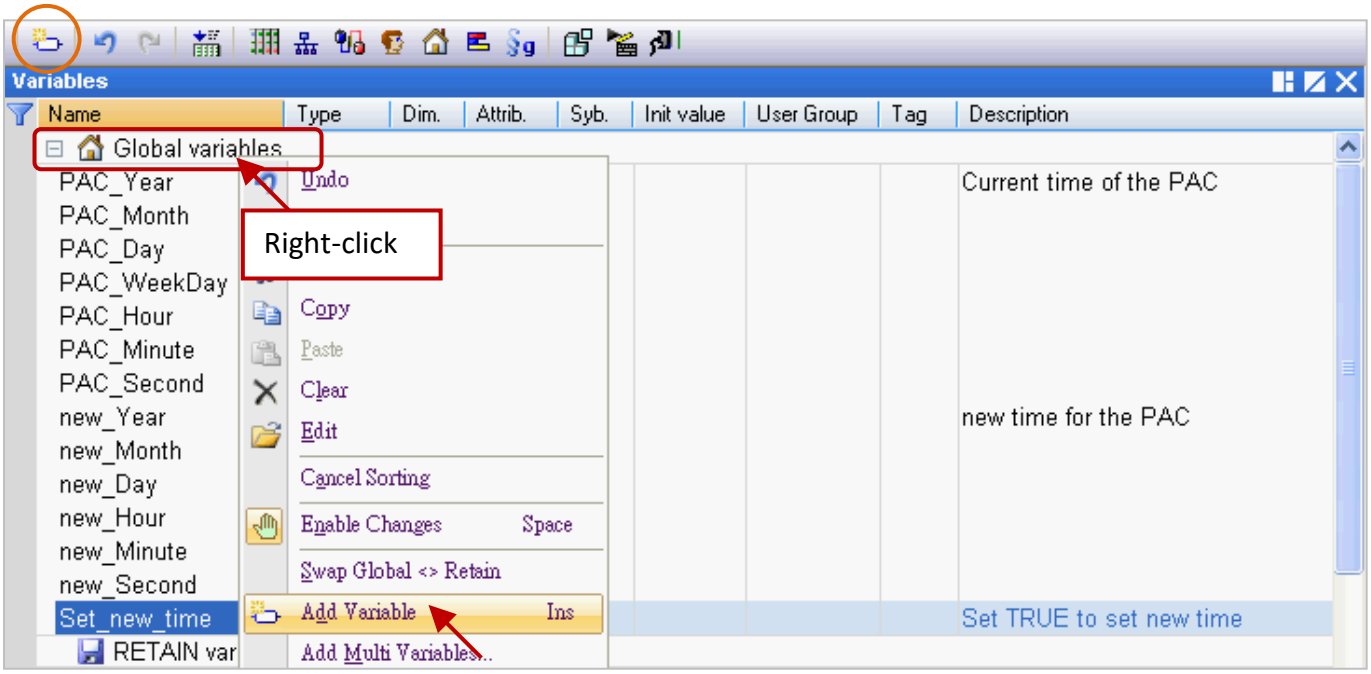
We have described the LD program (Section 2.2.1) and variables (Section 2.2.2) in the "Demo01" project. This chapter describes how to declare variables and add an LD program with the blinking function.

Note: "ULINT" and "LWORD" are not supported for Win-GRAF PAC.

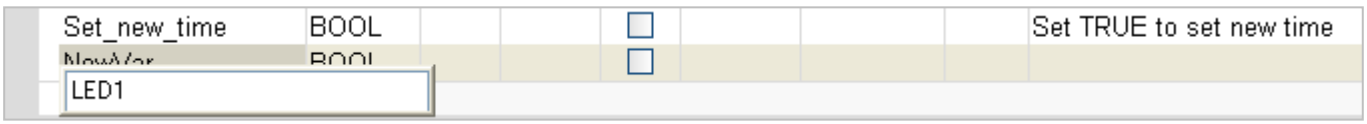
2.3.1 Declaring Variables

First, we will declare two boolean variables (i.e., "LED1" and "LED2") that are used in the program.

1. In the "Variables" window, right-click anywhere in the "Global variables" group and select "Add Variable" to add a variable. (Also click the icon or press the "Ins" key).



2. Double-click the "NewVar" to change the name to "LED1" and click "Enter" to finish the setting. In this case, the data type is "BOOL".



Note: The settings will be done only after clicking the "Enter" key.

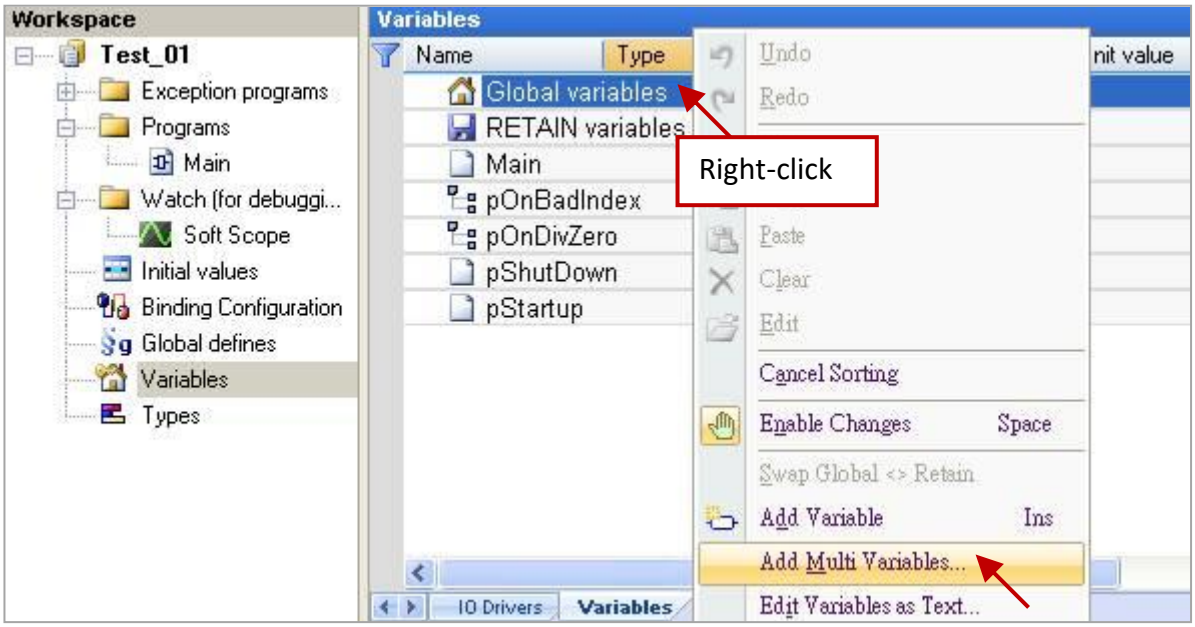
3. Follow the previous steps to add the "LED2" boolean variable.

Tips #1:

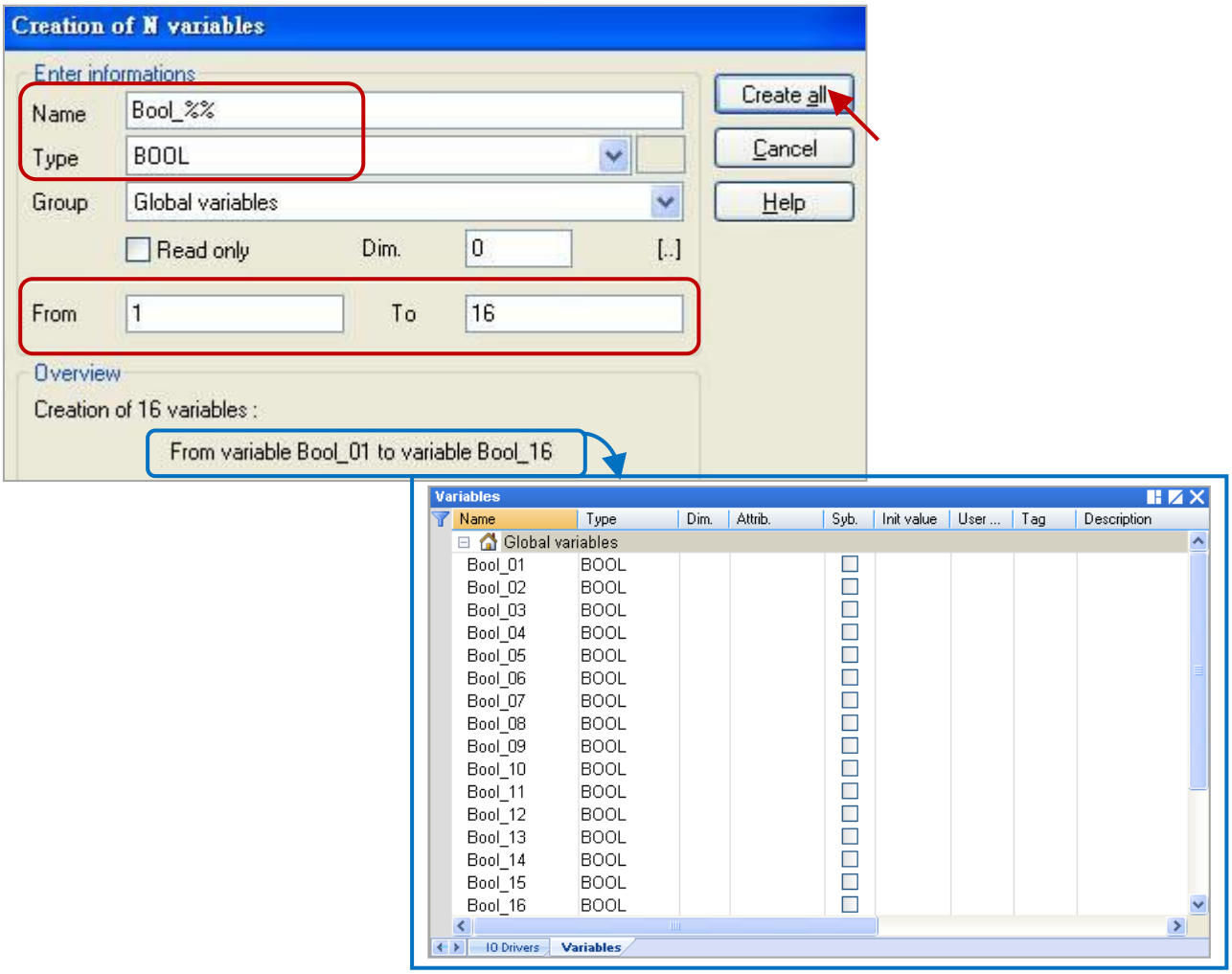
To add variables in sequential order, set the name as "LED" in step2 and then press "Ctrl+C" and "Ctrl+V" many times to create "LED**1**" and "LED**2**"…, then delete the "LED" variable.

Tip #2:

1. To add multiple variables in sequential order, for example, “**Bool_01**” to “**Boo_16**” Boolean variables. First right-click on the “Global variables” group and select the “Add Multi Variables”.



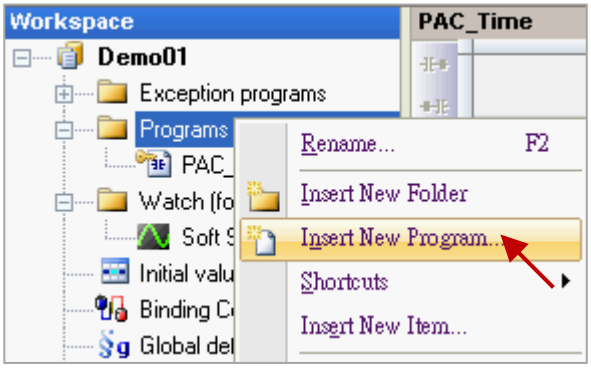
2. Set the Name as “Bool_%%”, the Type as “BOOL”, and the **From** and **To** fields to 1 and 16. Then, click the Create all button to add variables.



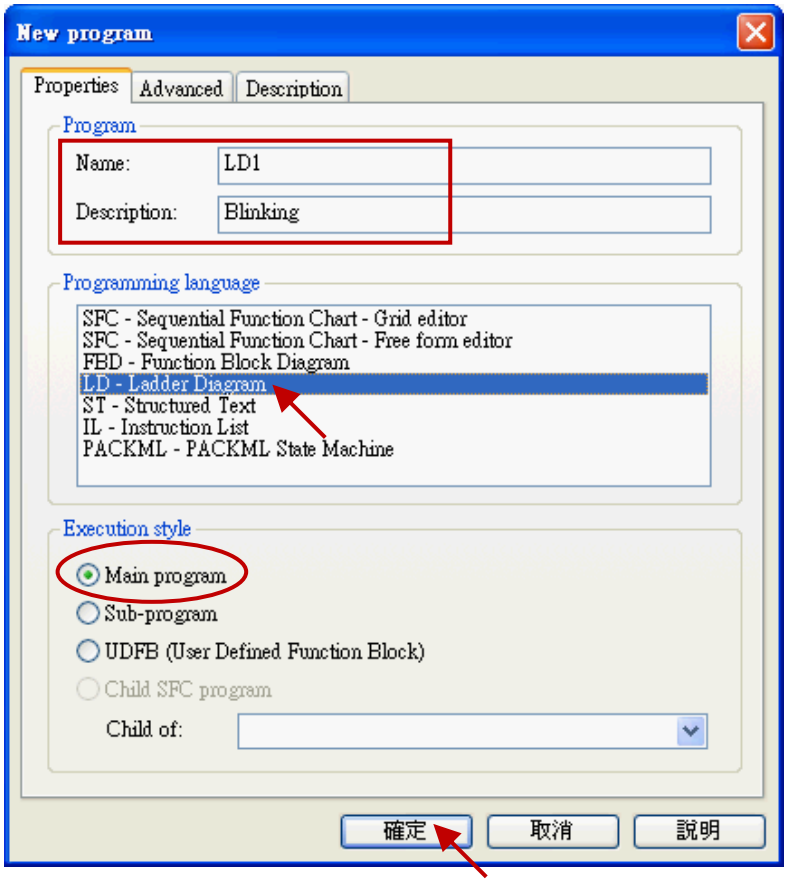
2.3.2 Creating an LD Program

In this section, we will create an LD program with a blinking function in the “Demo01” project. To begin, follow the steps below:

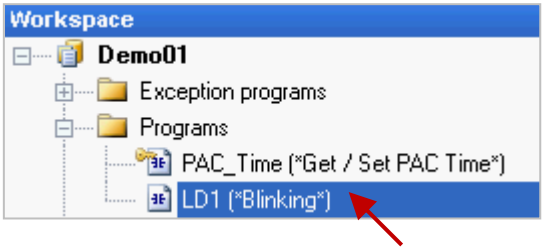
1. In the workspace, right-click the “Programs” folder and select “Insert New Program…”.



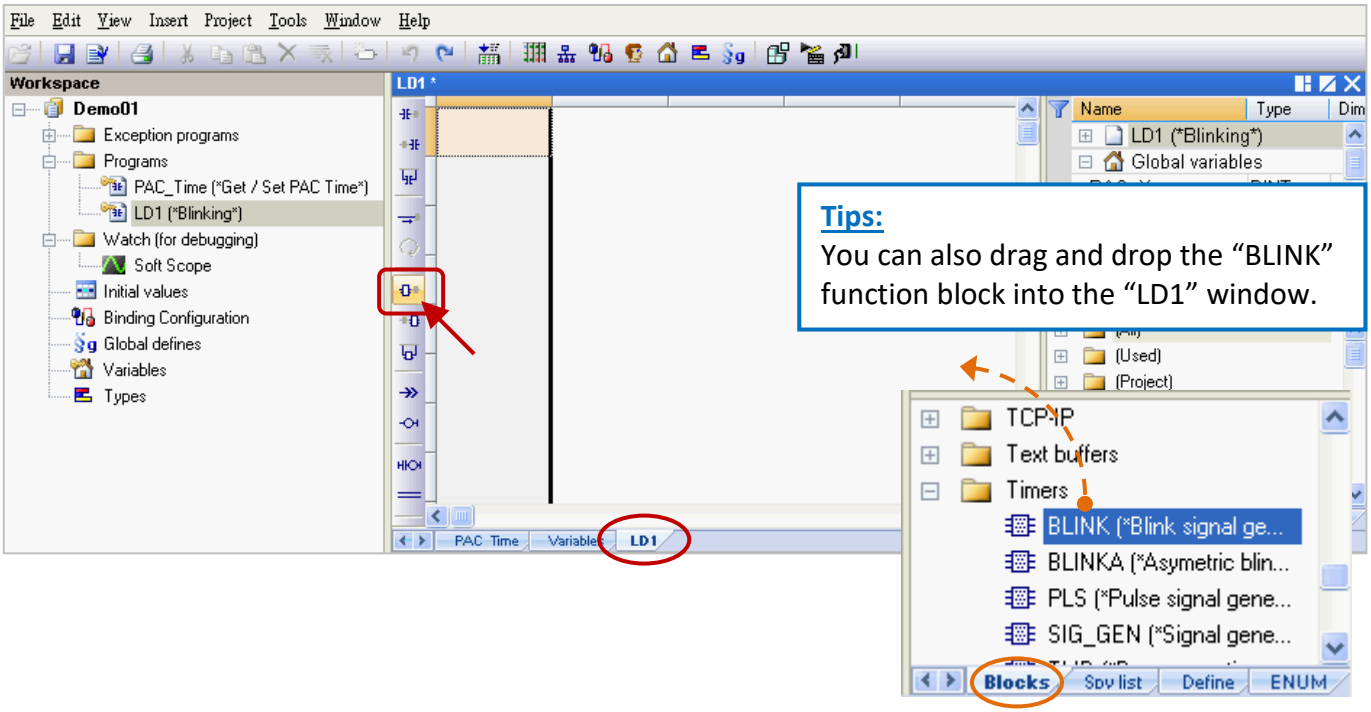
2. Fill in a program name in the “Name” field and enter a simple note in the “Description” field, and then select the “LD – Ladder Diagram” as the programming language and click the “OK” button.



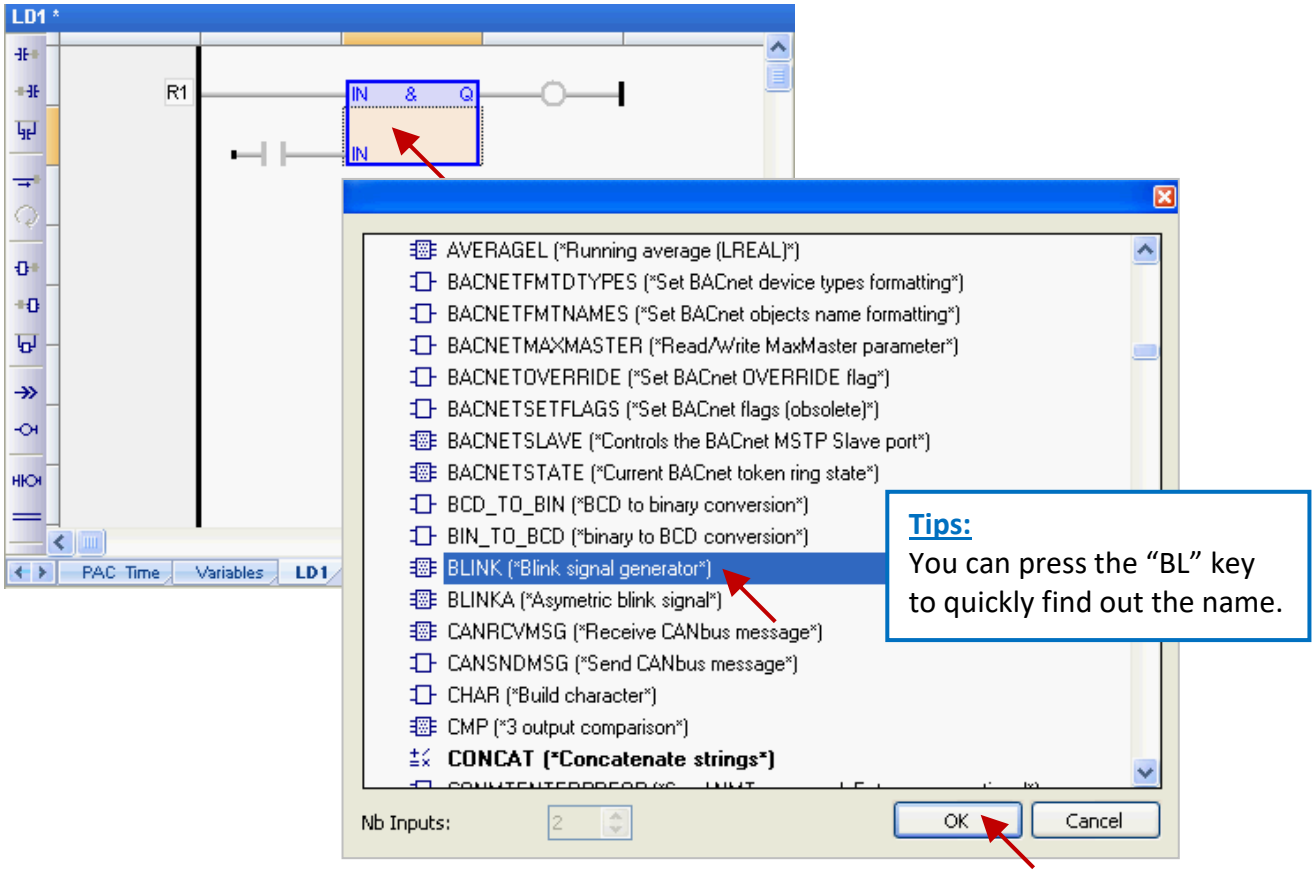
3. Double-click the “LD1” program to open the editor window.



4. Click the “Insert FB..” button on the left of the “LD1” window to add a function block.



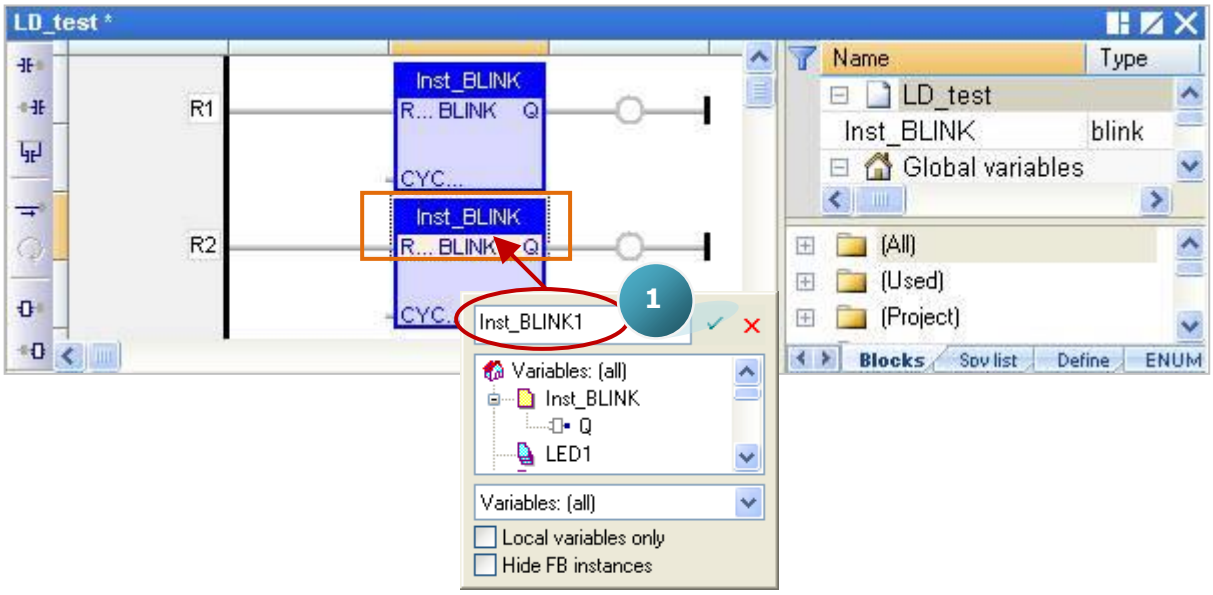
5. Double-click this function block and select the “BLINK”, then click the “OK” button.



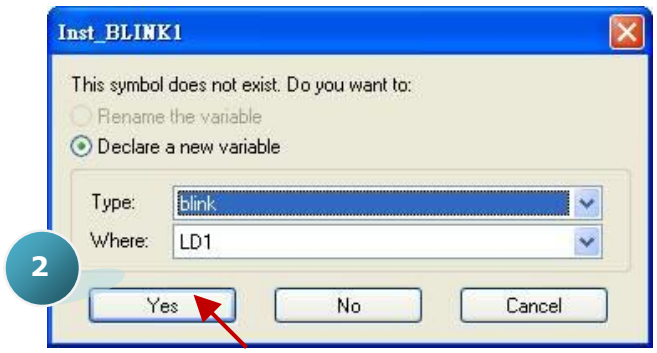
Important Notice:

Users may add a function block by copy & paste an existing one during programming. But, this may cause an exception due to the same function instance name. Therefore, **users must create a new name for the copied function block.**

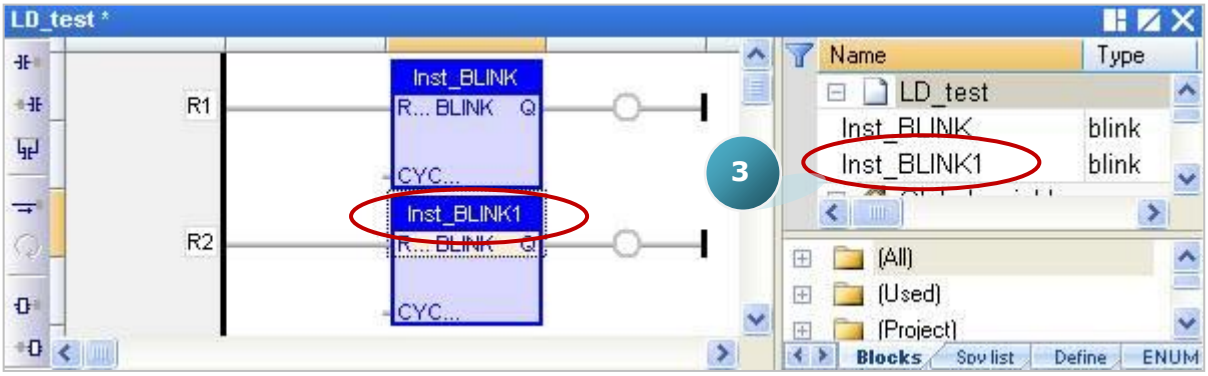
1. Double-click the function block and enter a new name (e.g., "Inst_BLINK**1**"), then click the ✓ button to complete the setting.



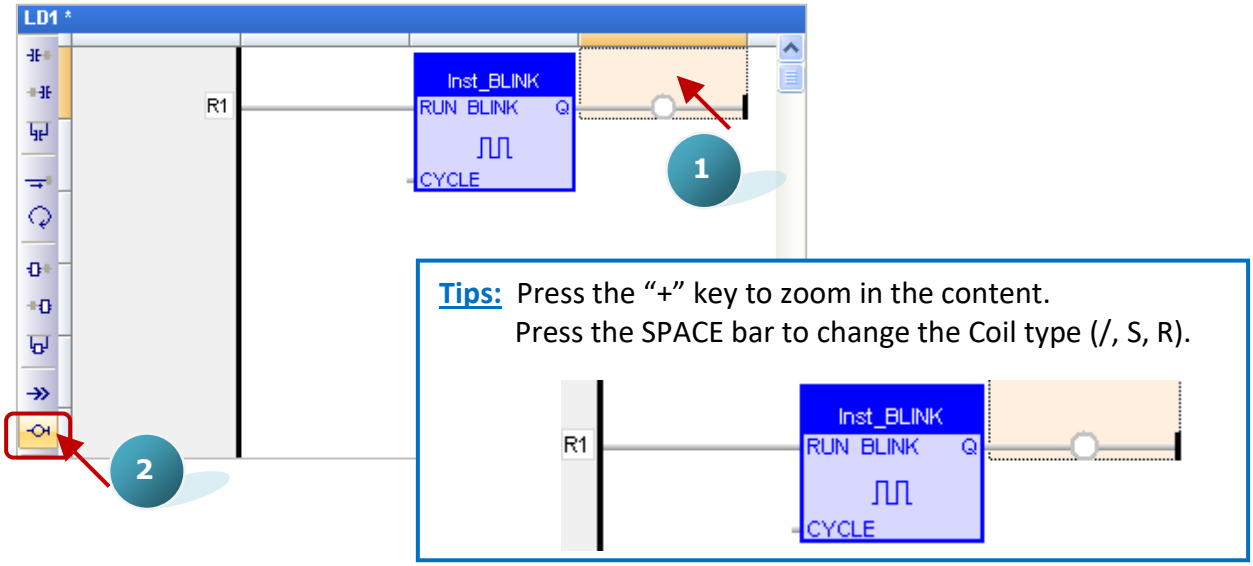
2. In the "Inst_BLINK1" window, click "Yes" to create this function instance name.



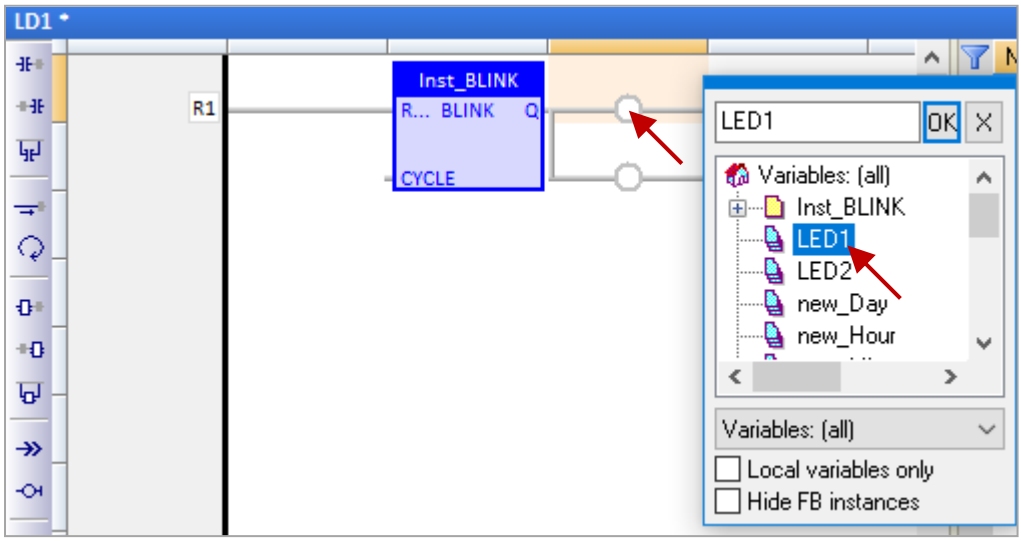
3. Now, there is an "Inst_BLINK**1**" FB instance added in the Variables Area.



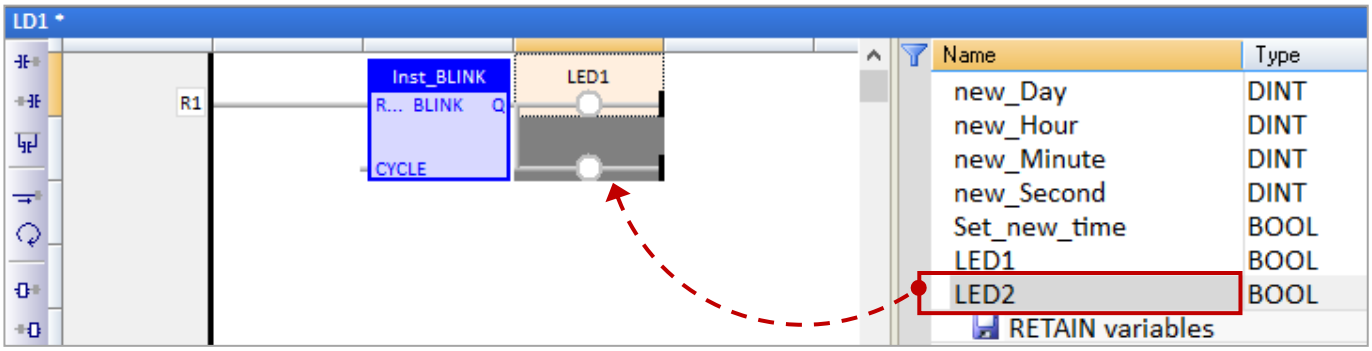
6. Click the “Coil” on the right of the “BLINK” function block, and continuously click the “Insert Coil” button to add two “Coil”.



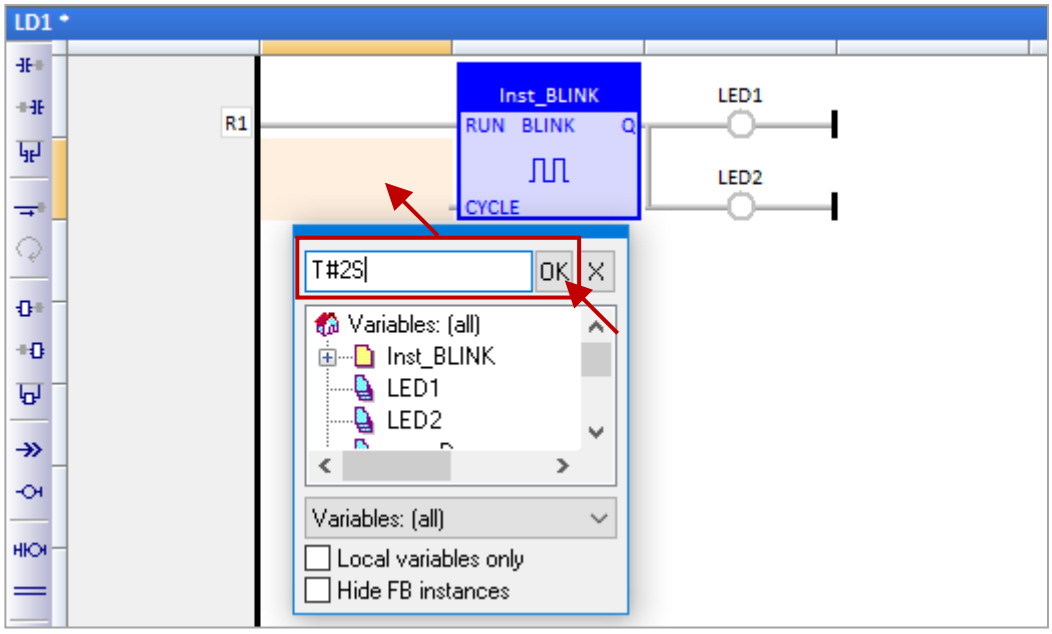
7. Double-click the first “Coil” and double-click “LED1” to assign the variable.



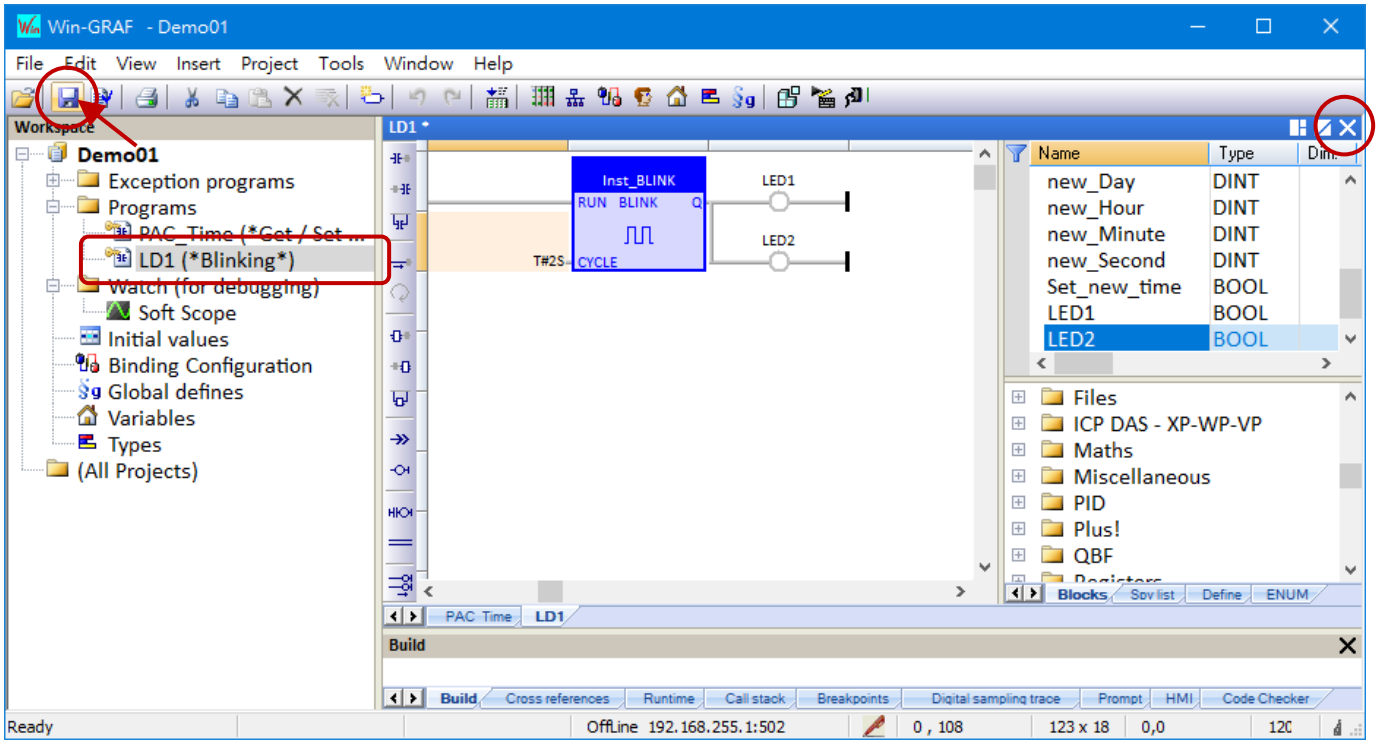
Also, mouse drag-and-drop the “LED2” variable from the variables area.



8. Double-click on the left of the “CYCLE” and enter “T#2S” (to blink every two seconds) and then click OK to finish the setting.



9. Finally, click the “Save” button to save the “LD1” program.

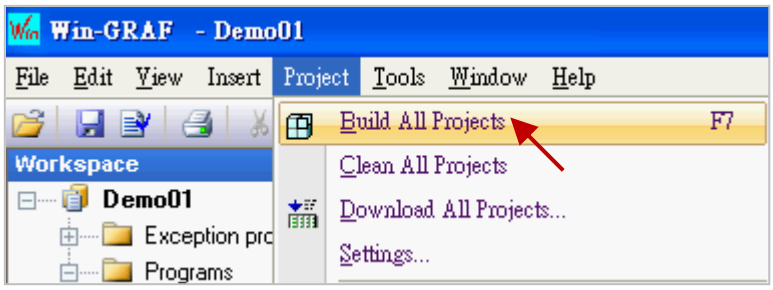


Note: “ ” means this program is opened (locked, cannot be deleted). Click the “X” symbol in the upper-right corner of the window to close this program (un-locked, “ ”).

2.3.3 Compiling the Program

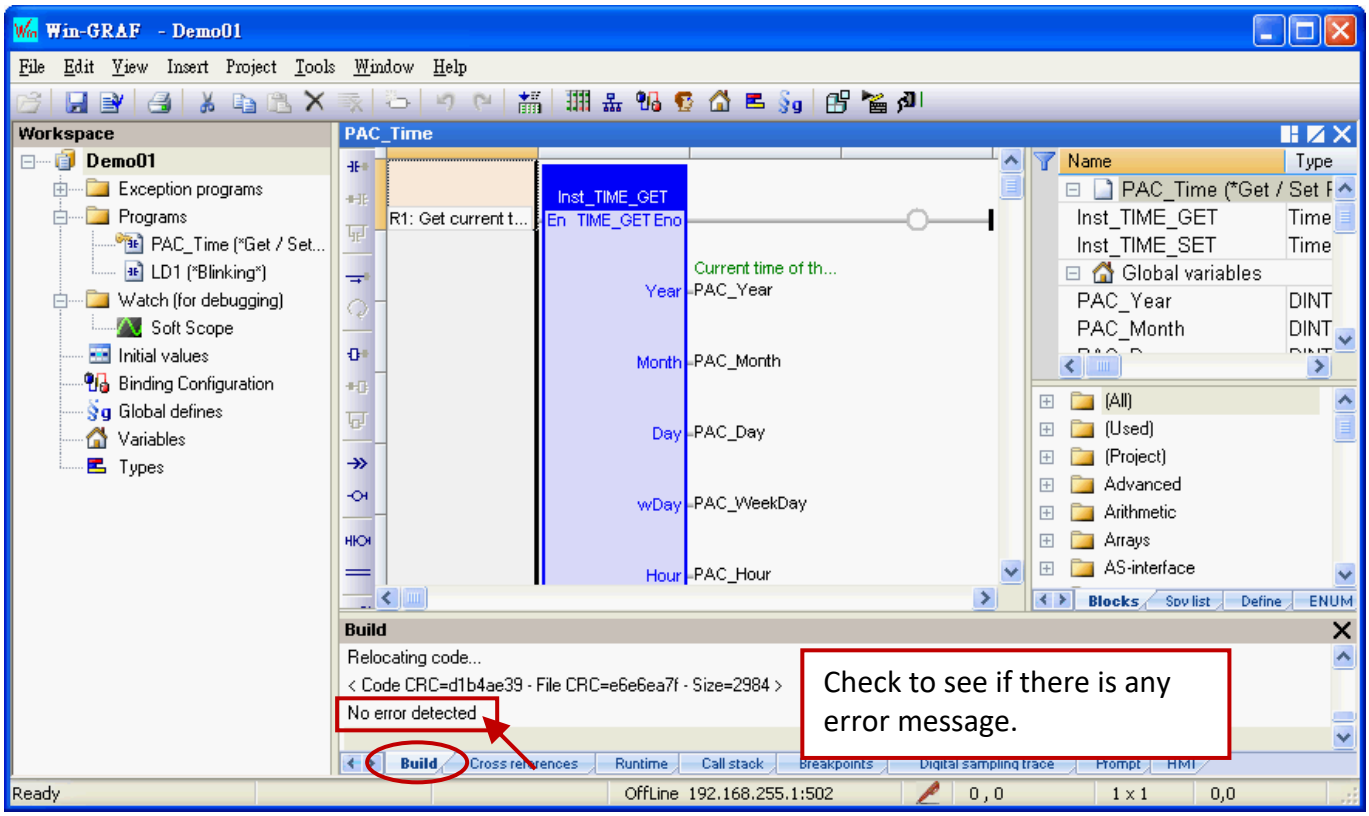
In the previous section, we have added and saved the LD program. For the Win-GRAF project can function properly on the PAC, we need to compile the programs. To begin, follow the steps below:

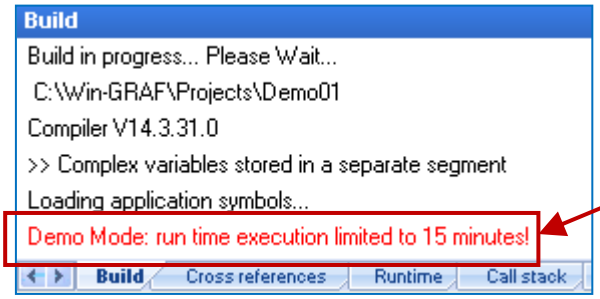
1. Click the menu command "**Project - Build All Projects**" to compile all programs.



2. If "No error detected" appears in the message area which means the compilation was successful.

   Note: **If the program is modified and saved again,** run the "Clean All Projects" command before compiler the program.



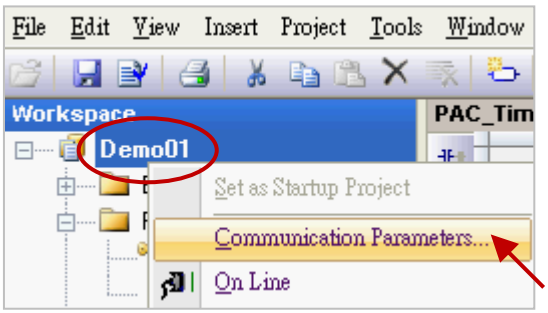


In Demo mode, the message indicates that the project can only run for 15 minutes on the PAC.

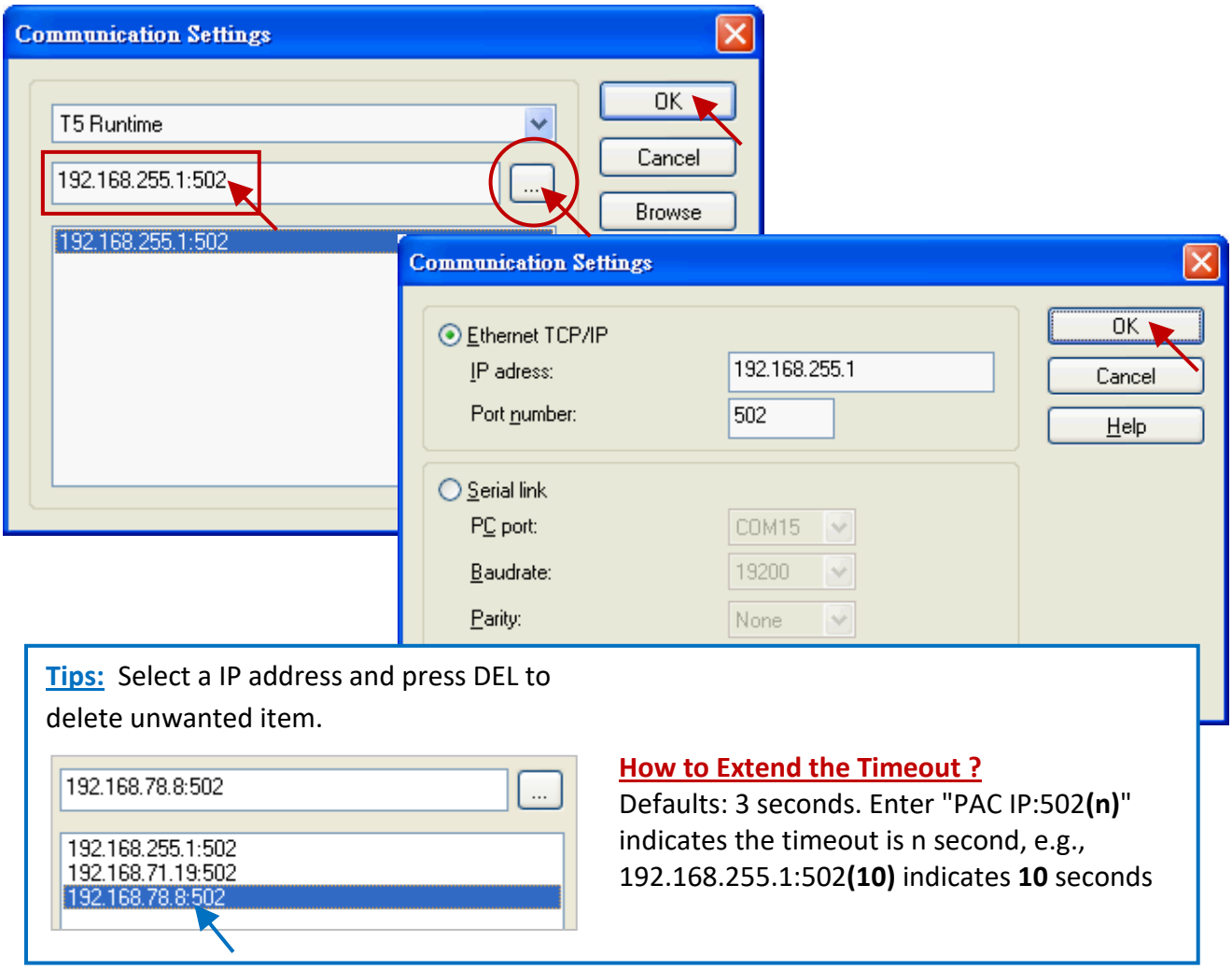
2.3.4 Download the Program to PAC

Before downloading the program, set the Ethernet communication parameters properly. Also, refer to Appendix C uses the serial link.

1. Right-click the project name (i.e., "Demo01") and select “Communication Parameters…” to open the settings window.



2. Enter the IP address with the TCP port number (e.g., the factory setting “192.168.255.1:502”) and click “OK”. Also, click the ... button to add/modify the IP address.



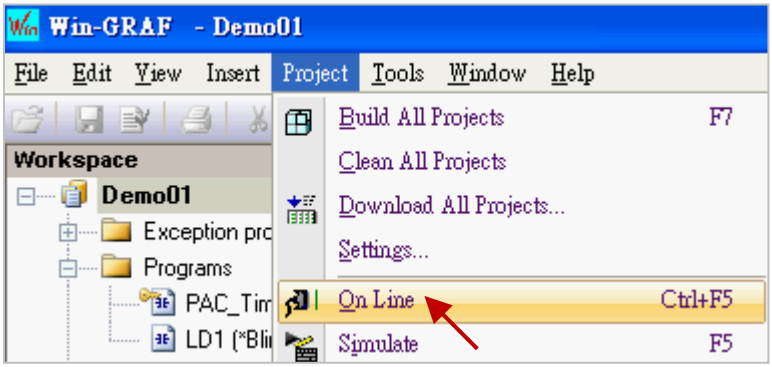
Tips: Select a IP address and press DEL to delete unwanted item.

How to Extend the Timeout ?

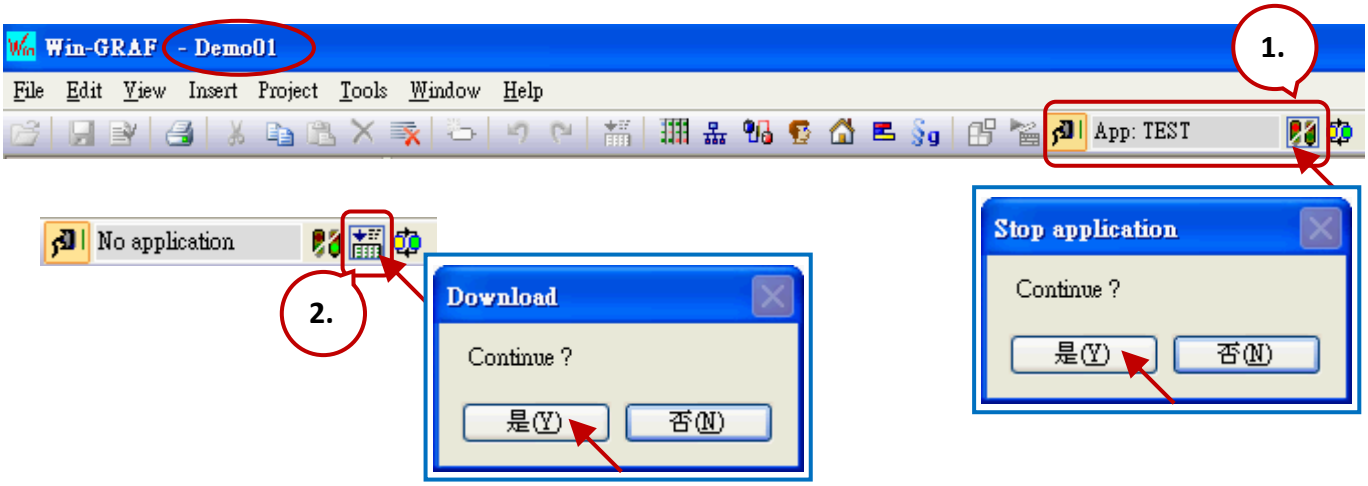
Defaults: 3 seconds. Enter "PAC IP:502**(n)**" indicates the timeout is n second, e.g., 192.168.255.1:502**(10)** indicates **10** seconds

3. **Before establishing a connection, make sure the PAC and the network are working properly.**

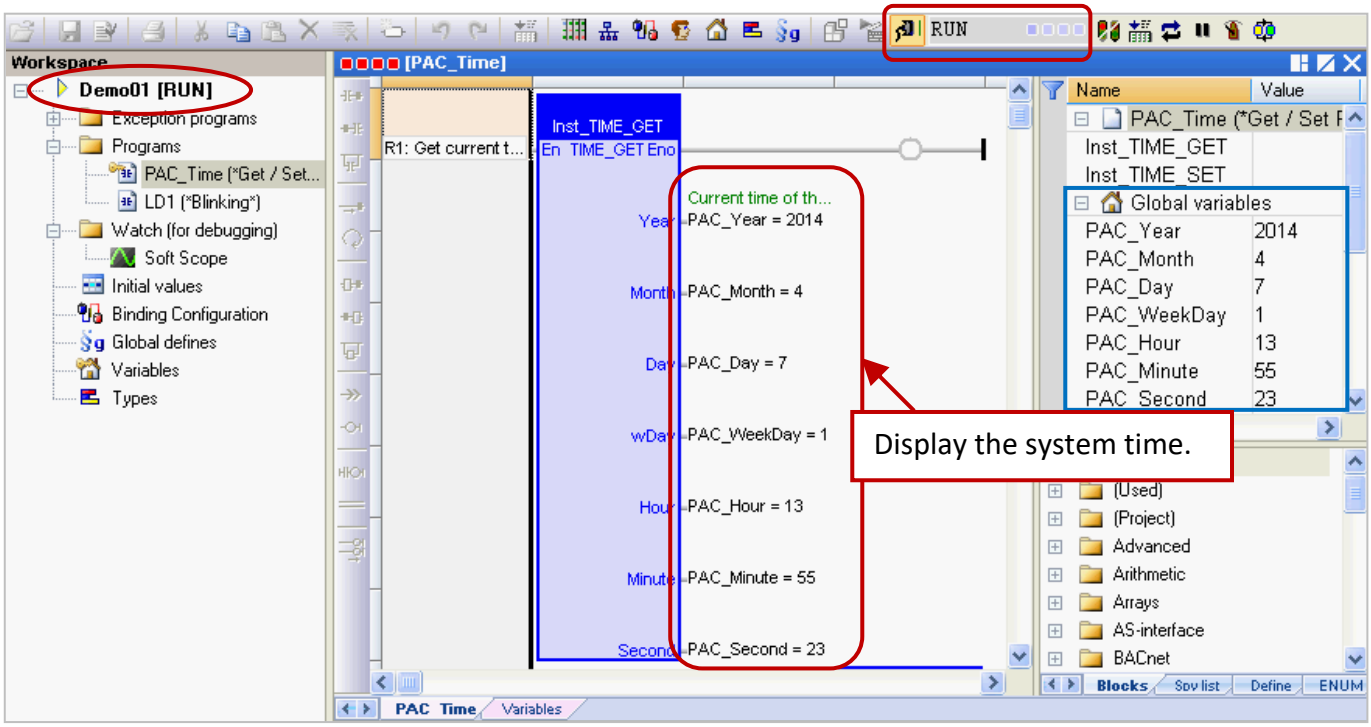
4. Click the menu command “**Project - On Line**” (or) to connect with the PAC.



5. If the displayed project name (TEST) is different from the current one (Demo01), click the “Stop application” button to stop running the old project. Then, click the “Download” button to download the “Demo01” project.



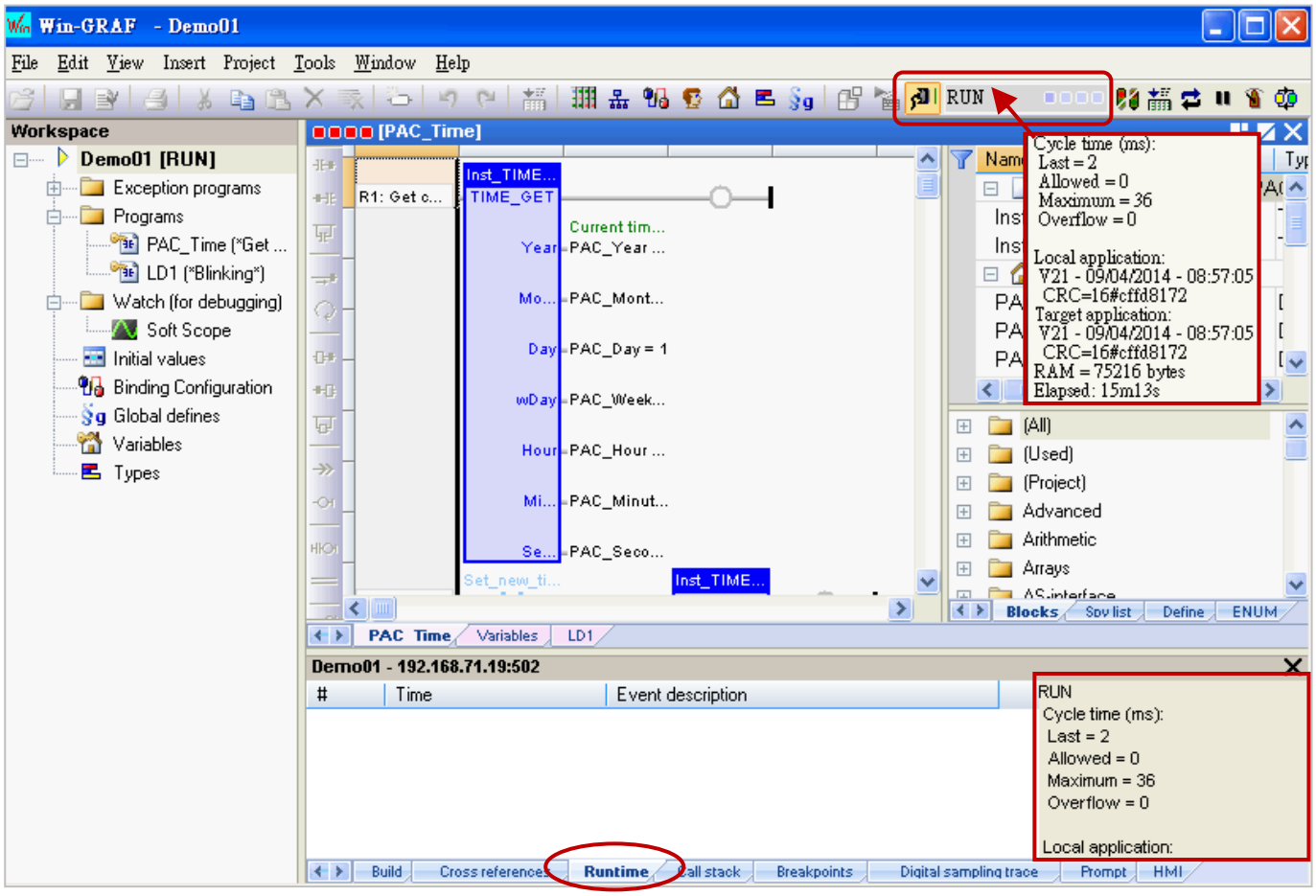
6. If “RUN” appears which means the "Demo 01" project is running on the PAC.
   Note: Refer to Appendix B for troubleshooting if an error message is displayed.



Cycle time

When the Workbench is connected with the PAC, move the cursor over the “RUN” position on the toolbar to view the cycle time of the program on the PAC. You can also view the cycle time in the bottom-right corner of the message area.

Also, check to see if there is an error message in the Runtime window while the program is running.

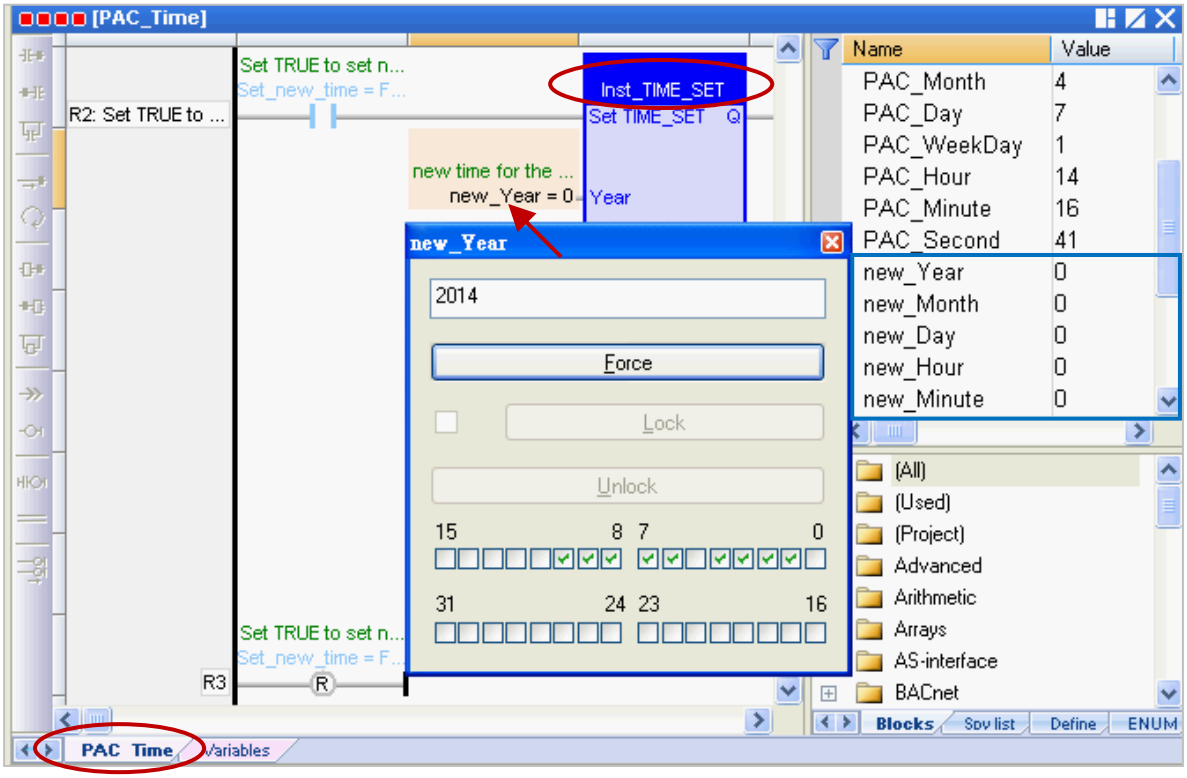


2.3.5 Testing the Program

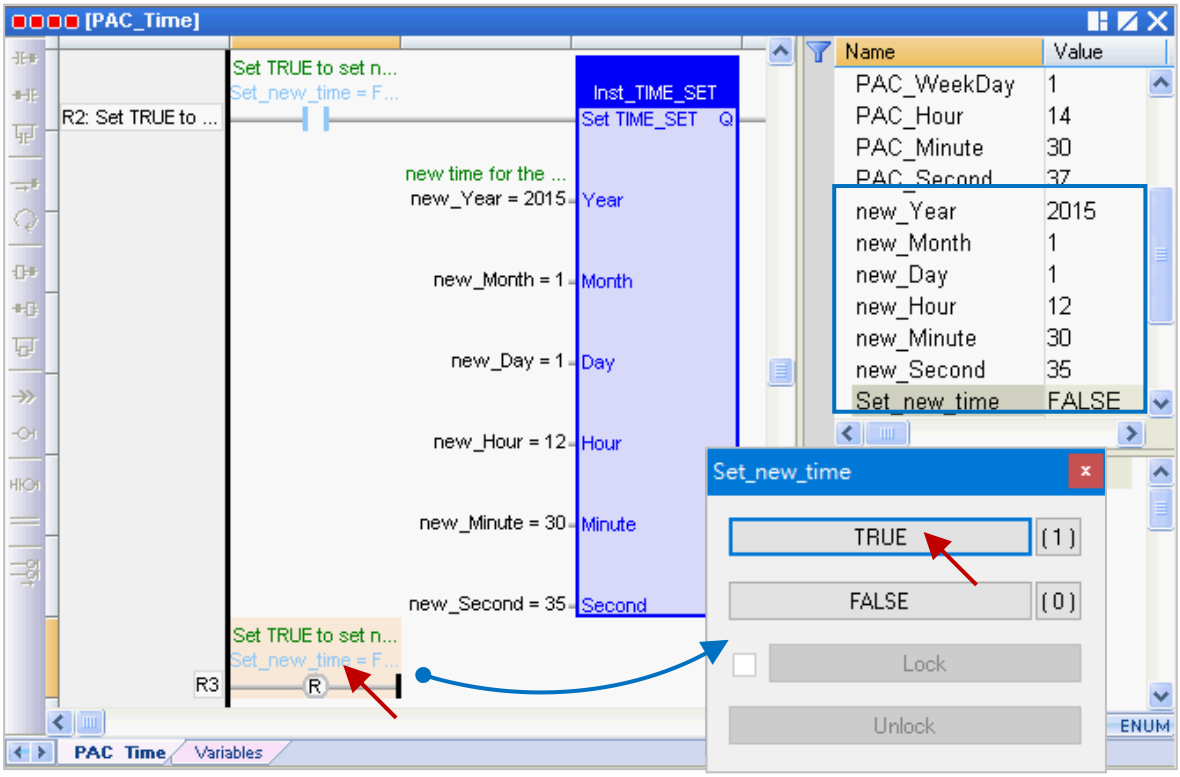
In the previous section, the “Demo01” project is successfully downloaded. The following will describe how to test the program.

The “PAC_Time” Program:

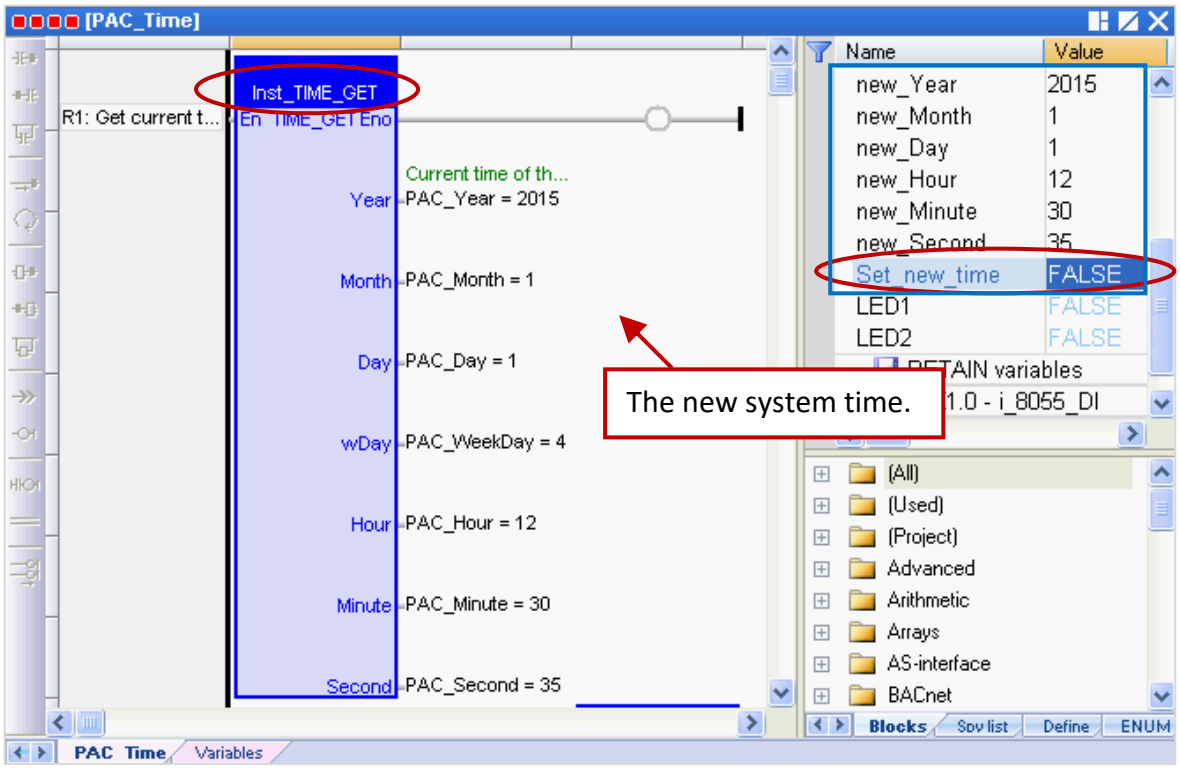
1. Double-click the variable name (e.g., “new_Year”) in the “TIME_**SET**” function block (or the Variables area) to change the PAC’s system time.



2. Set the “Set_new_time” variable to “TRUE” to write the new system time.

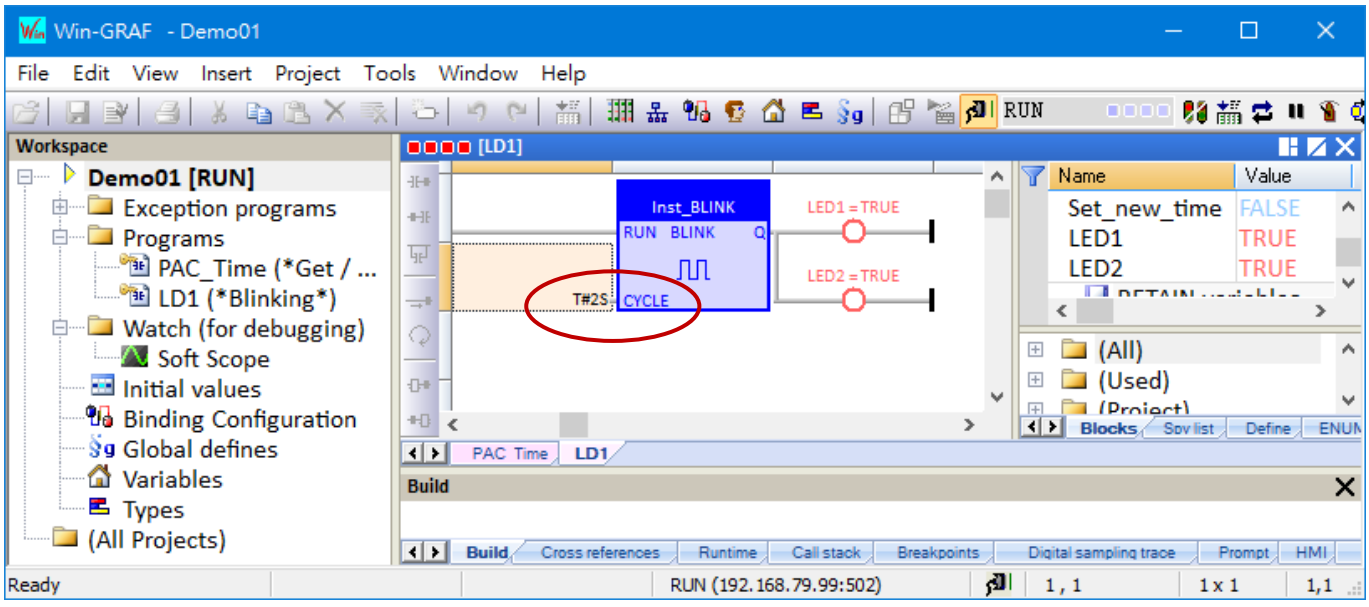


3. Then, the new system time will show on the “TIME_**GET**” function block (or the Variables Area), and the “Set_new_time” variable will be reset to “FALSE” automatically.



The “LD1” program:

4. When the “Demo01” project is running, you can see both the LED1 and LED2 are blinking every two seconds (i.e., "T#2S”) that cannot be changed. If you want to change the time, click the button again to disconnect and assign a “TIME” variable.



Note: **Click the “On Line” button instead of the “Stop Application” button or the project on the PAC will stop running.**

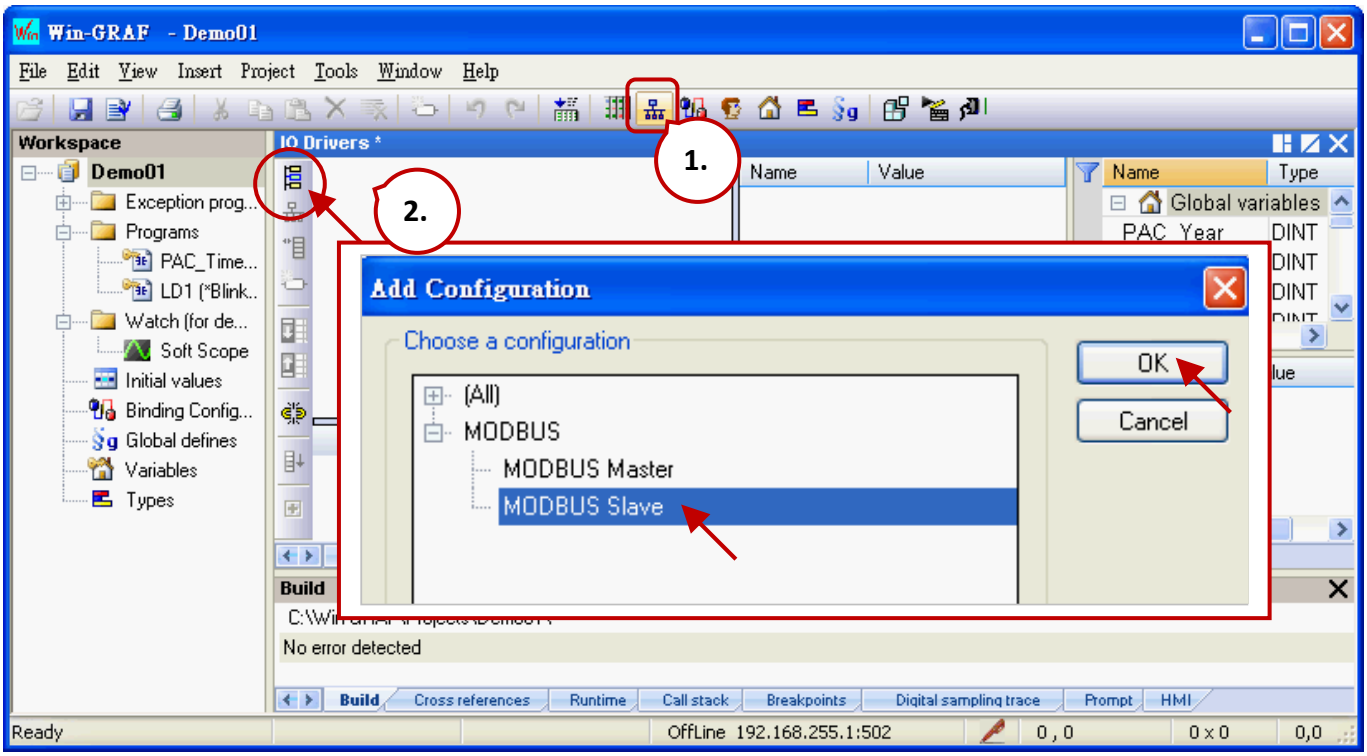
Chapter 3 Modbus Slave: Allow the SCADA/HMI Software to Access Win-GRAF Variables

The chapter describes how to allow the SCADA/HMI software (e.g., "InduSoft") to access Win-GRAF variables data via Modbus **TCP or** Modbus **RTU** protocol. To begin, follow the steps below to set Win-GRAF PAC as Modbus Slave and specify variables.

3.1 To Enable Win-GRAF PAC as a Modbus TCP Slave

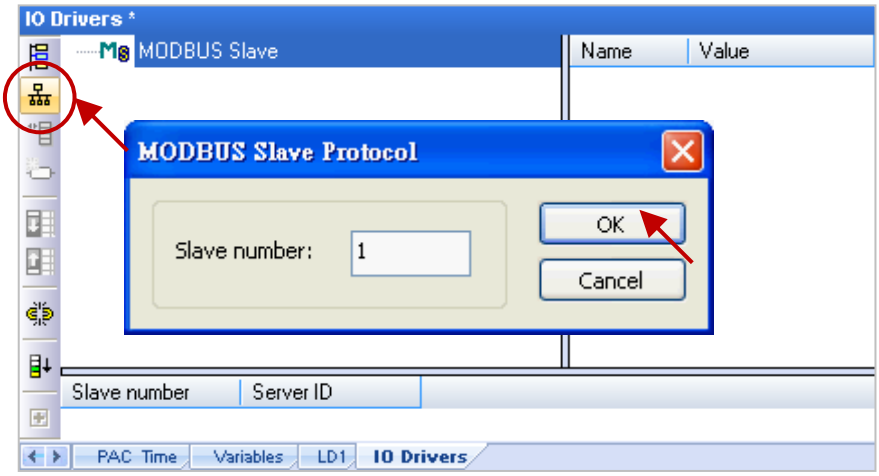
Setup the PAC to act as a Modbus Slave

1. Click the "Open Fieldbus Configuration" button on the toolbar to open the "IO Drivers" window.
2. Click the "Insert Configuration" button on the left side of the "IO Drivers" window and then select the "MODBUS Slave" and click "OK" to enable a Modbus TCP Slave.



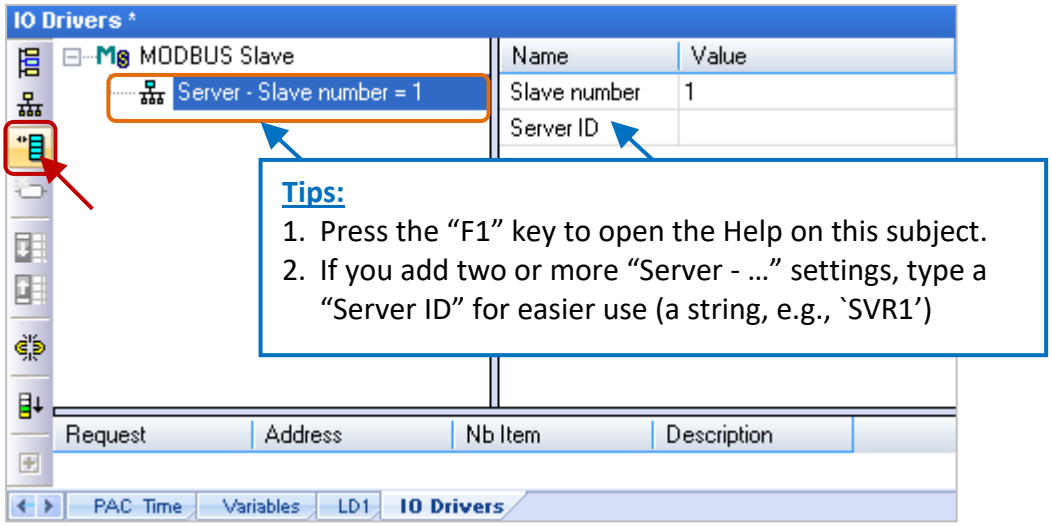
Set the Slave ID (or Server ID)

3. Click the "Insert Master/Port" button to set the "Slave number" (e.g., "1") and click "OK".

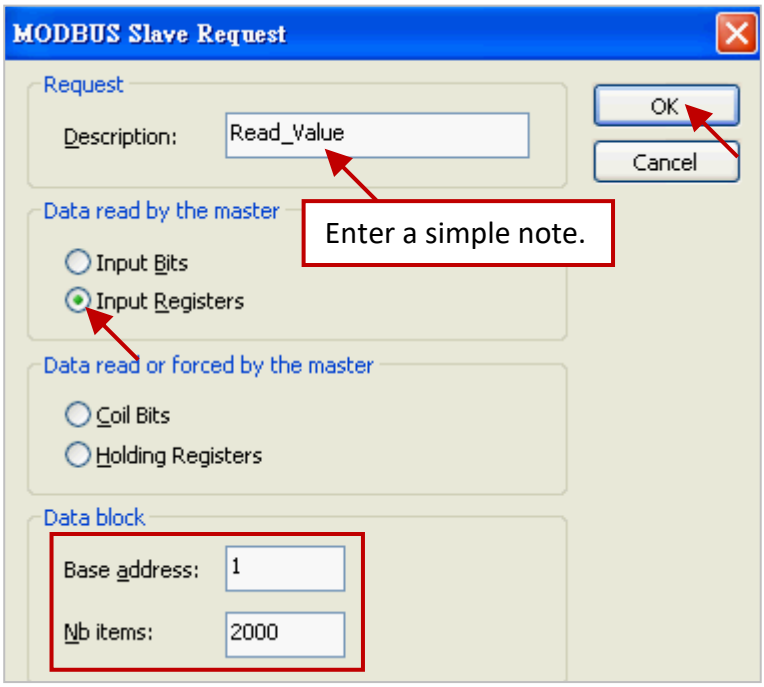


Create data Block

4. Click the “Insert Slave/Data Block” button on the left side to open the “MODBUS Slave Request” window.



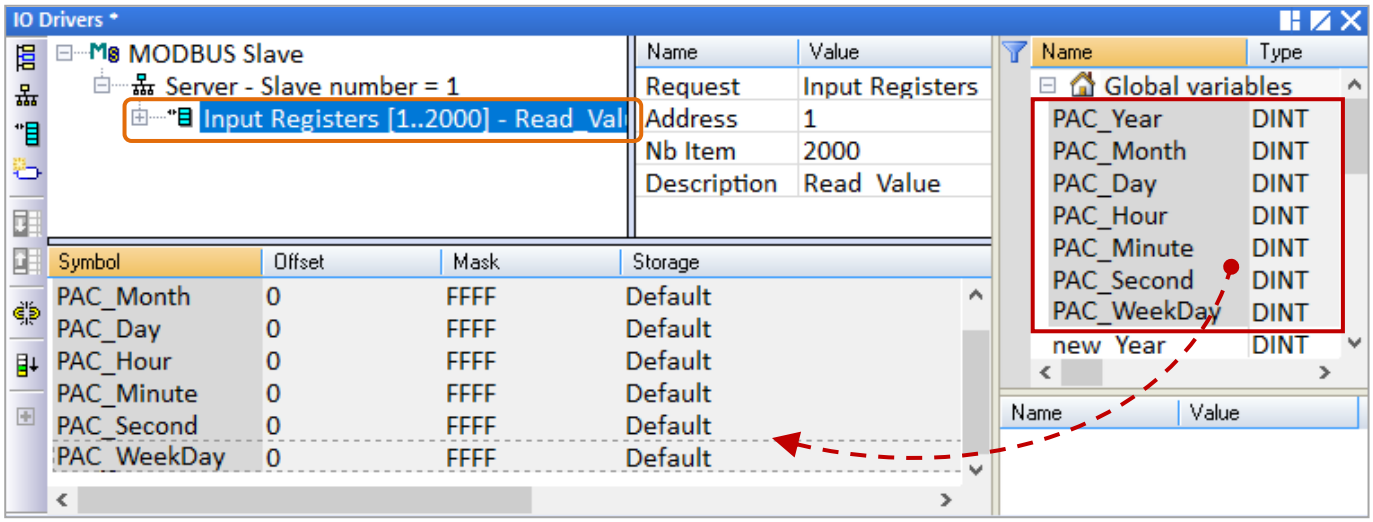
5. Read AI variables:
   Enter a note in the “Description” field and click the "Input Registers” option. It is recommended to set “Base address” to “1” and set “Nb items” to a value greater than “200”.



The “Nb items” refers to how much variables data (bits or words) can be accessed in a “data block”. If the address of the data request from the Modbus Master (i.e., SCADA/HMI) is greater than this value (e.g., “2000”), the Modbus Slave (i.e., Win-GRAF PAC) will not respond.

Specify variables for Modbus Master to access data

6. Drag and drop the variables (e.g., “PAC_xxx”, data type: “DINT”) to the address mapping area.



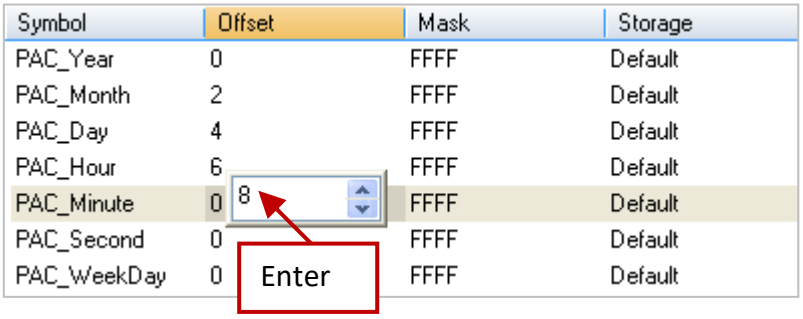
Specify the offset to map with Modbus address

7. Double-click the “Offset” field and fill in a value, then press the “Enter” key to finish the setting.

Note:

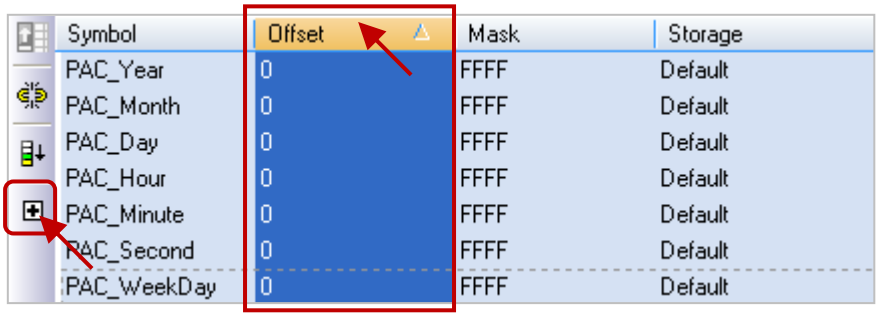
(1) “Offset” is a 0-based address. So, the Modbus address (1-based) is equal to the Offset plus 1.

(2) When using a 32-bit (or above) variable (e.g., “DINT”), it requires two consecutive addresses. The “Offset” values are 0, 2, 4, 6, etc.

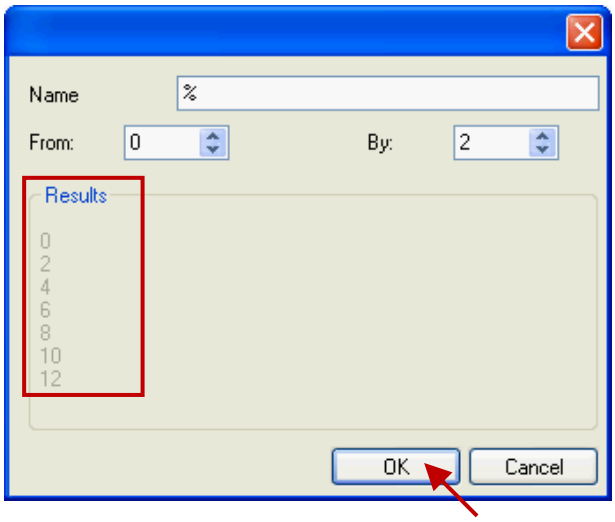


Tips:

Click column header “Offset” to select entire columns and click the “Iterate Property” button on the left side to open the settings window.

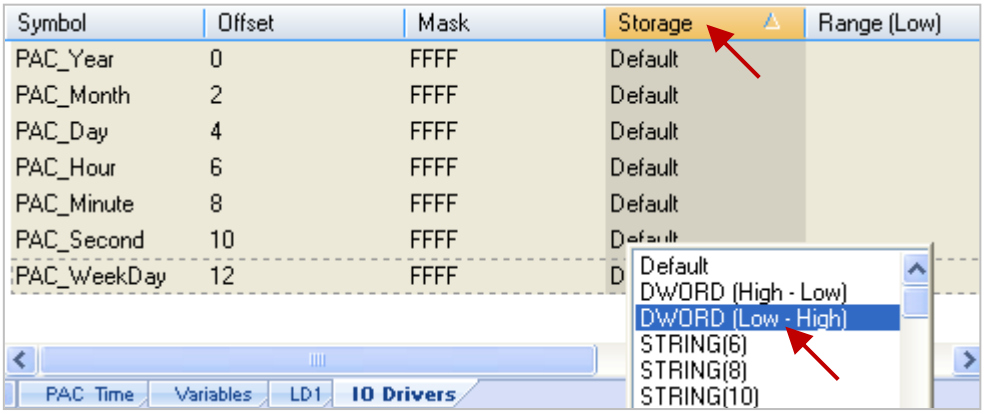


Keep the “Name” setting, enter “0” into “From” field, and enter “2” into “By” field, then click “OK”.

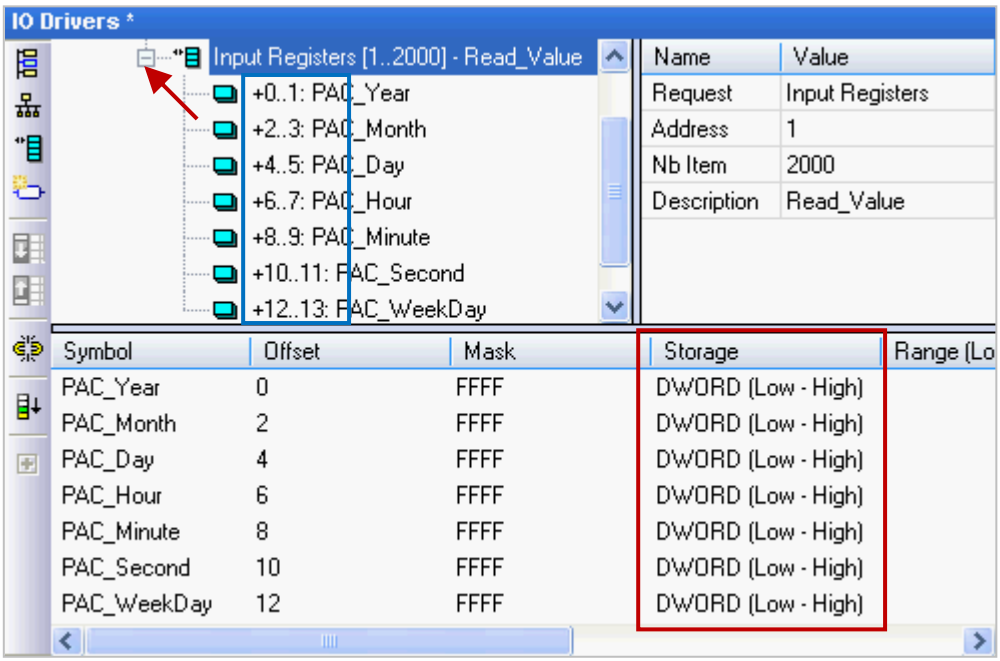


Set Storage for 32-bit variables

8. Click the column header “Storage” to select entire columns and press the “Enter” key to display a drop-down menu. Then, double-click “DWORD (Low – High)” to set all entries.



To expand this “Data Block” and you can see that the data will be stored by using two addresses.

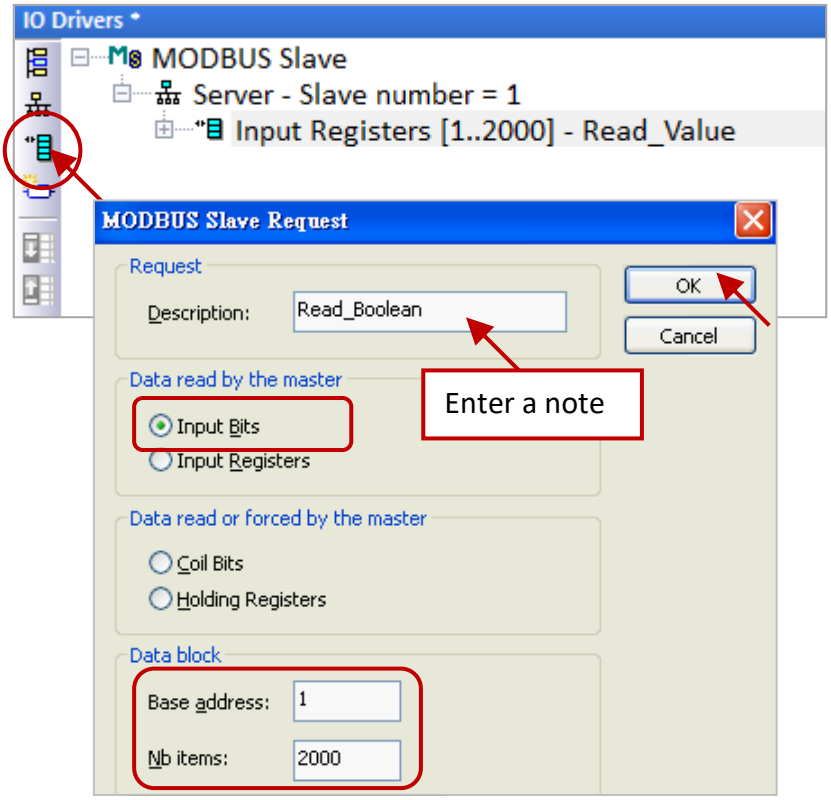


Create the second data block

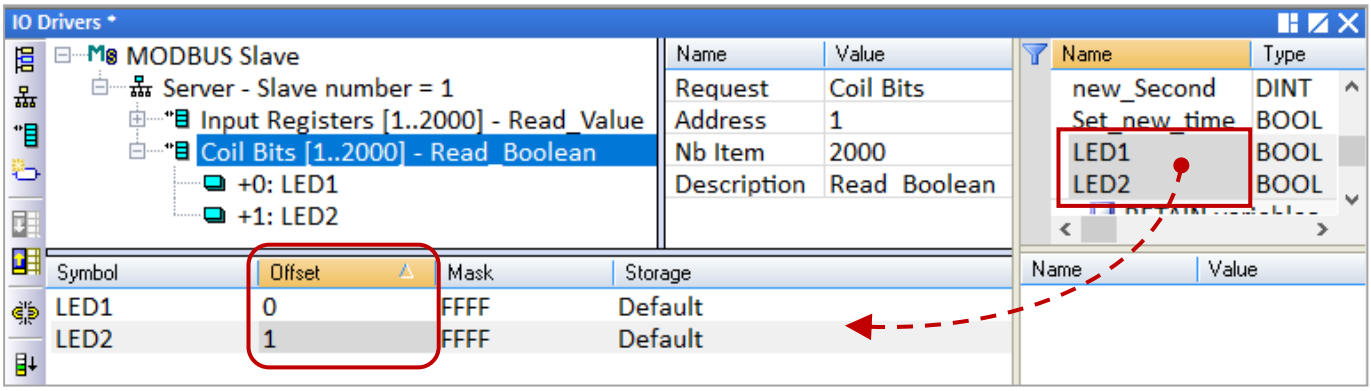
9. To add a “Data Block” for the Modbus Master to read Boolean value. The configure way is similar to steps 4 to 8:

   1) Click the “Insert Slave/Data Block” button on the left side to open the settings window.

   2) In the “MODBUS Slave Request” window, enter a note and select the “Input Bits” option, then set “Base address” to “1” and set “Nb items” to “2000”.



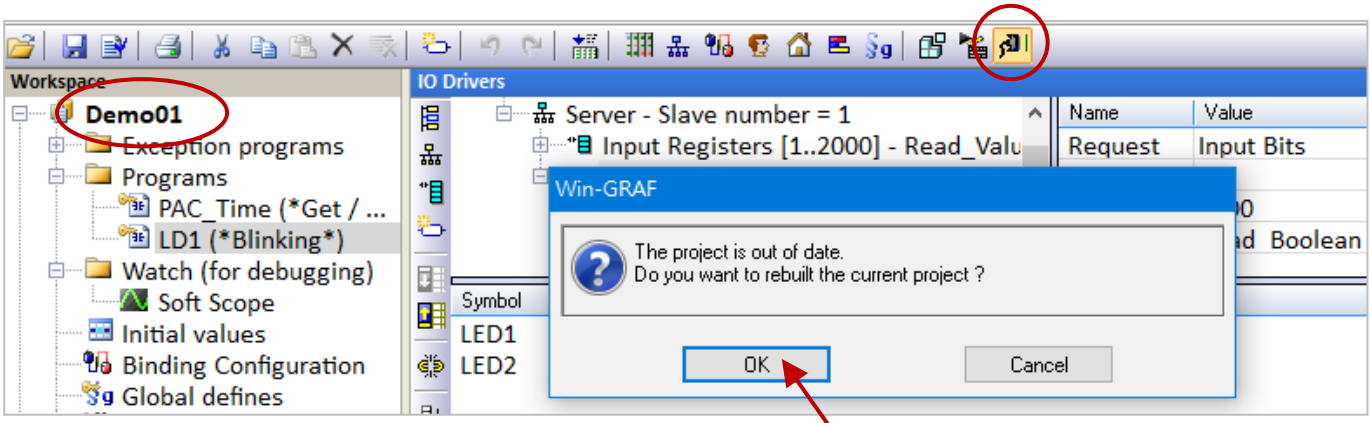
   3) Drag and drop Boolean variables (i.e., “LED1” and “LED2”; data type: BOOL) to the address mapping area and set the “Offset” to “0” and to “1”.



You have completed the settings for the Modbus Slave. Finally, follow the instructions below to download the program to the Win-GRAF PAC.

Download this project

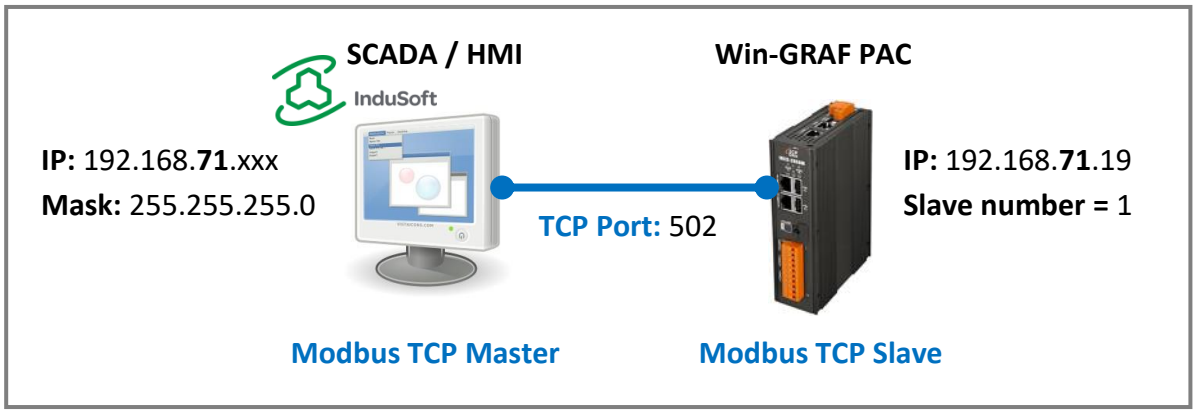
10. Click the "On Line" button ( ) to establish a connection and download this project to the Win-GRAF PAC. If the project has ever been compiled or changed, the dialog box below will be displayed. Click OK to re-built and download the project.



Before downloading, right-click the project name and select "Communication Parameters…" to set the PAC IP (e.g., "192.168.71.19:502")

Note:
"Bad version" indicates the version is changed. Click the "Stop application" button to stop running the project and click the "Download" button to download the new version of the project.

After completing all steps, the SCADA/HMI software can access Win-GRAF variables via the Modbus TCP protocol.

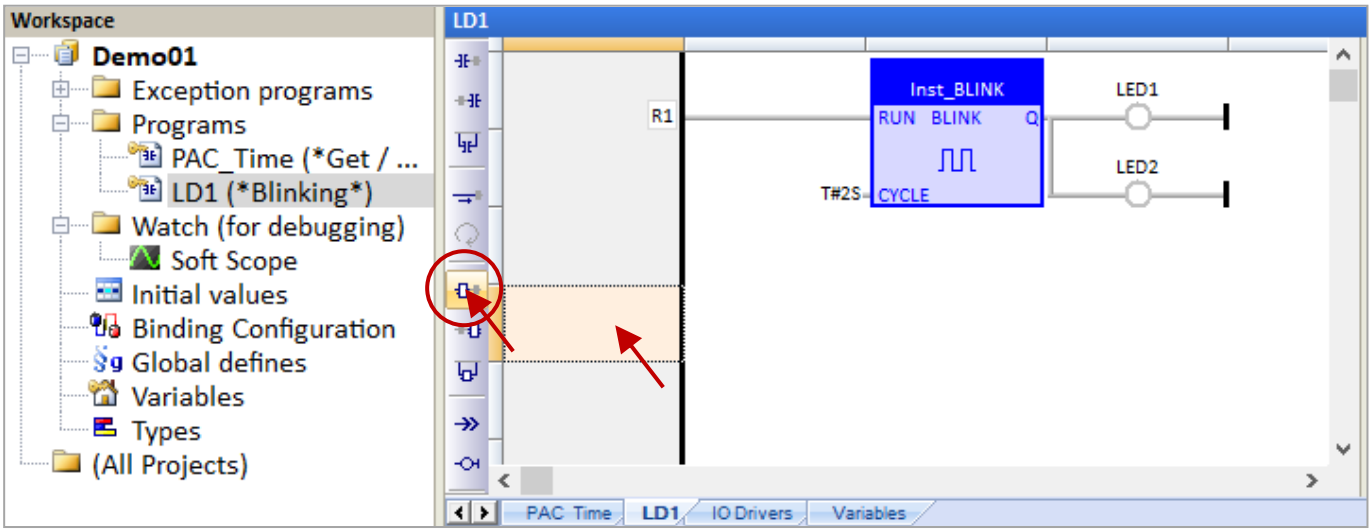


3.2 To Enable the Win-GRAF PAC as a Modbus RTU Slave

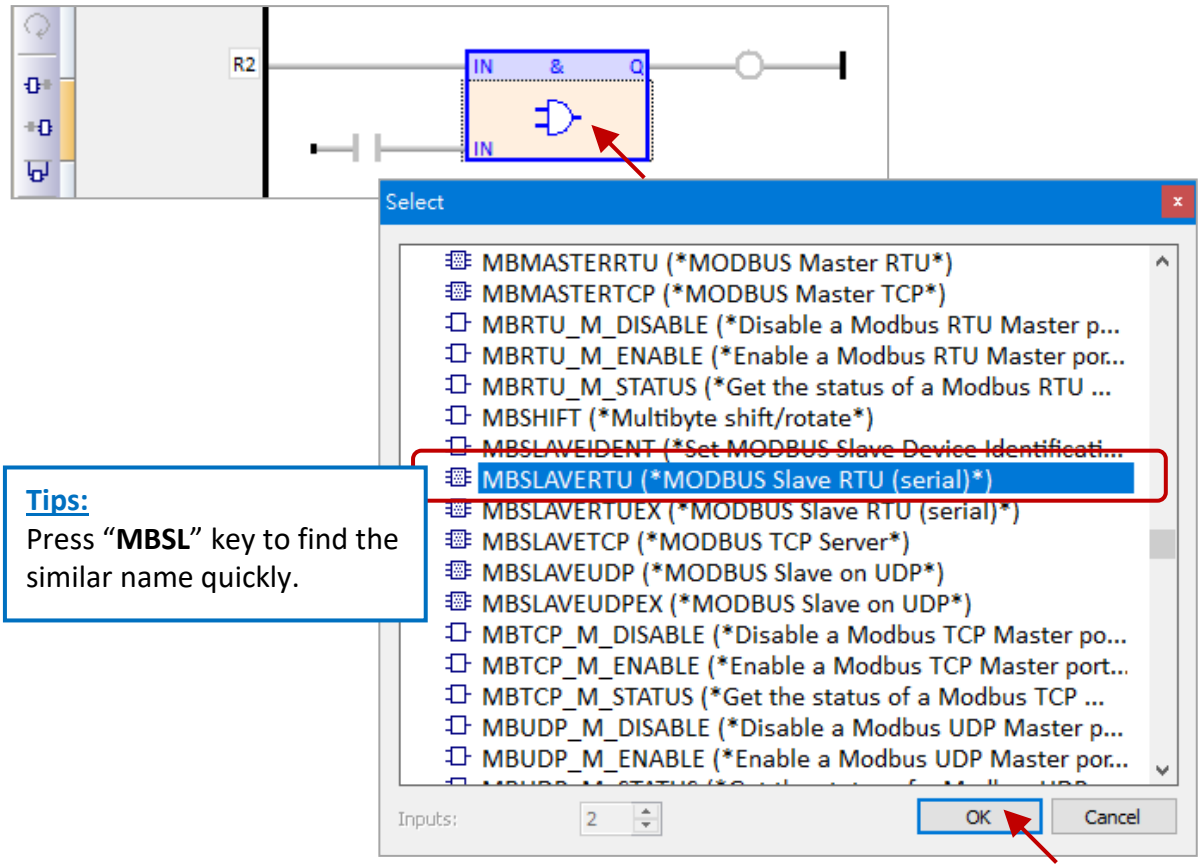
Before you get started, make sure that Win-GRAF PAC is set as a Modbus Slave and variables are set to public, refer to Section 3.1. To access data via Modbus RTU, the “MBSLAVERTU” (or “MBSLAVERTUEX”) function block must be added to the program. To begin, follow the steps below:

Add the “MBSLAVERTU” function block

1. In the “LD1” window, click on the position where you want to add this function block and click the “Insert FB..” button on the left side of the window.

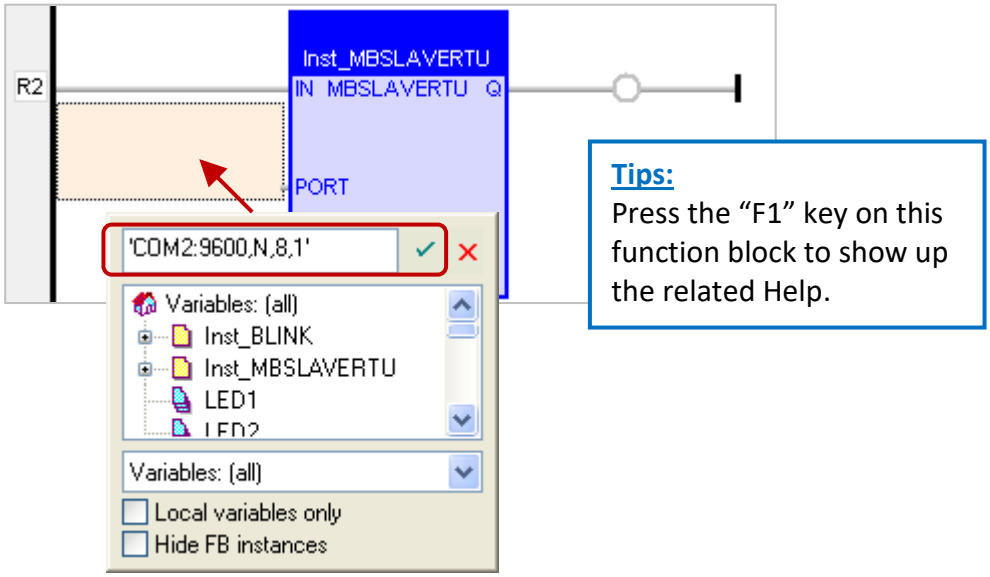


2. Double-click the function block and select the name “MBSLAVERTU”, then click “OK”.

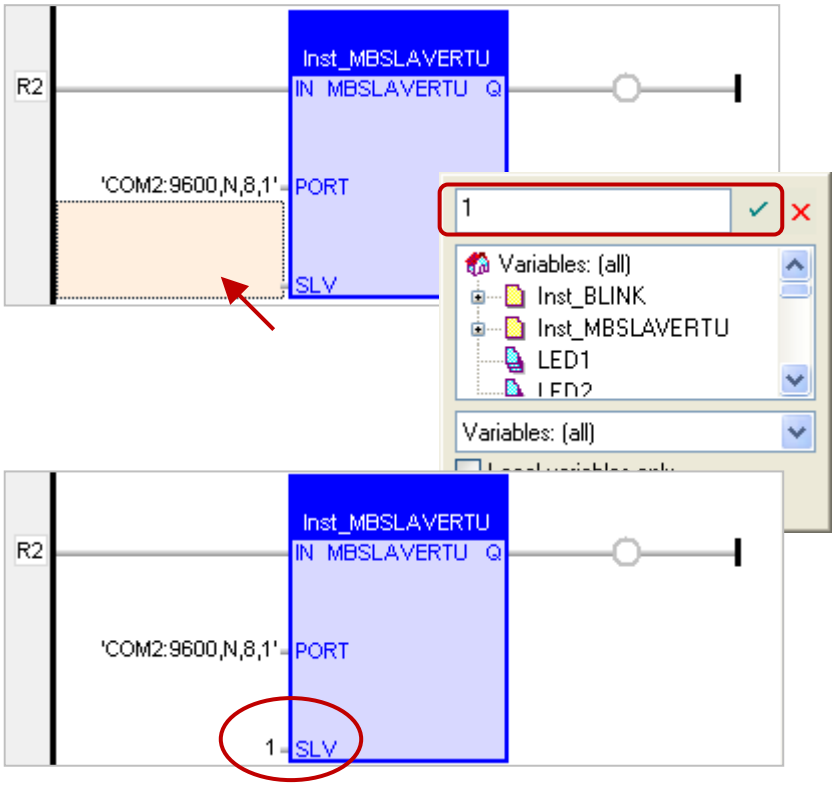


Tips:
Press “**MBSL**” key to find the similar name quickly.

3. In the “MBSLAVERTU” function block, double-click the left side of the “PORT” and enter a string 'COM2:9600,N,8,1' which means using the COM2 of Win-GRAF PAC to communicate with the Modbus Master), then click ✓ to complete the settings.

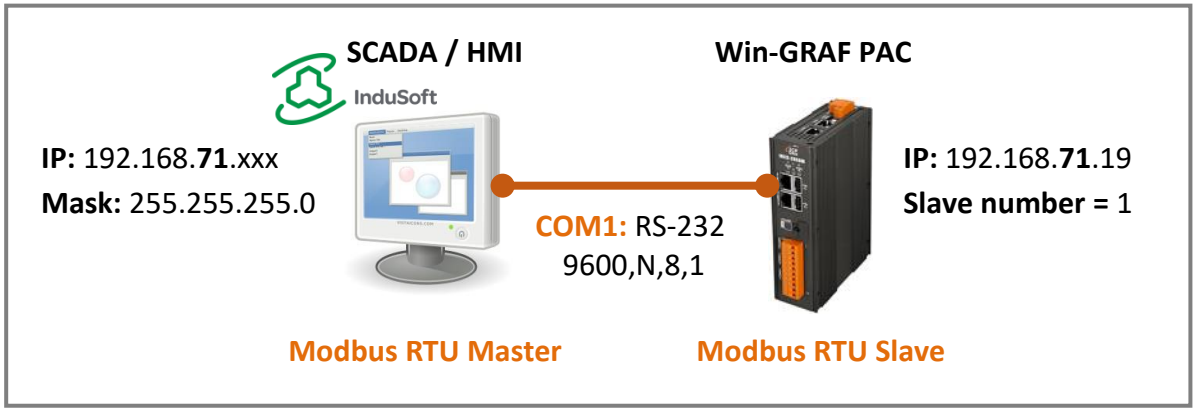


4. Double-click the left side of the “SLV” and enter “1” (the value set in Section 3.1 - Step 3), then click ✓ to finish the setting.



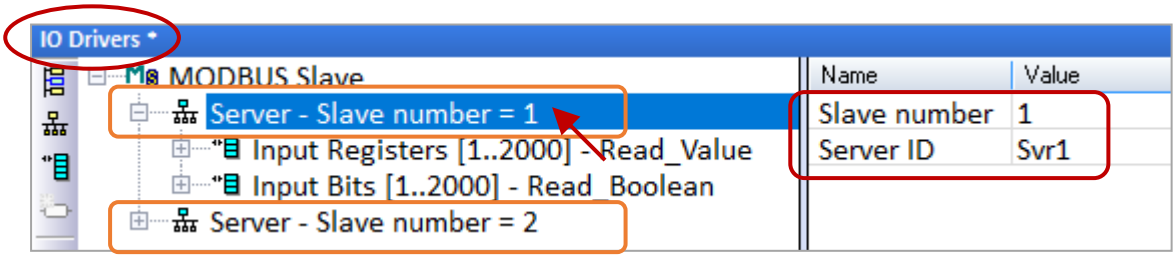
Now, you have completed the setting of the “MBSLAVERTU” function block, and then download the project to the Win-GRAF PAC.

Note: Users can enable multiple Modbus RTU Slave ports on a PAC (recommend not over 16 Ports). The way is to add multiple “MBSLAVERTU” function blocks and set the different “Port” numbers.

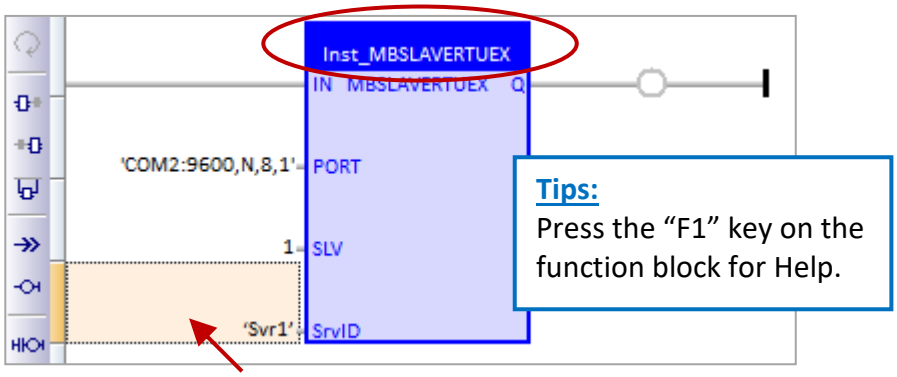


Add the "MBSLAVERTUEX" function block

The "MBSLAVERTU**EX**" function block can be used if there are several Modbus Slave configurations (recommend to set one) in the "IO Drivers" window.



Follow steps 1 to 4 above to add the "MBSLAVERTU**EX**" function block in the "LD1" window. Compare with "MBSLAVERTU", it adds a "SrvID" setting. Double-click on the left side of the "SrvID" and enter the Server ID (e.g., 'Svr1' with String format) that is set from the MODBUS Slave configuration.



Note:

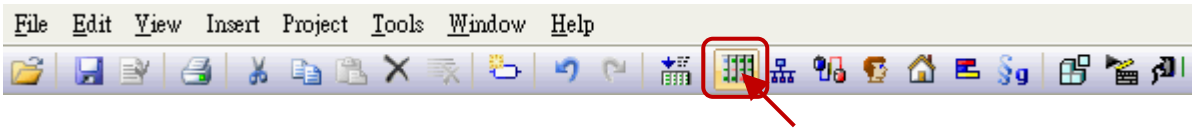
- With MBSlaveRTU**ex**, you can specify the ID of a server from the MODBUS Slave configuration. If you specify an empty string, the first server is used.
- MBSlaveRTU always works with the first server in the configuration.

The "MBSLAVERTU**EX**" function is completed by now, download the project to the Win-GRAF PAC.

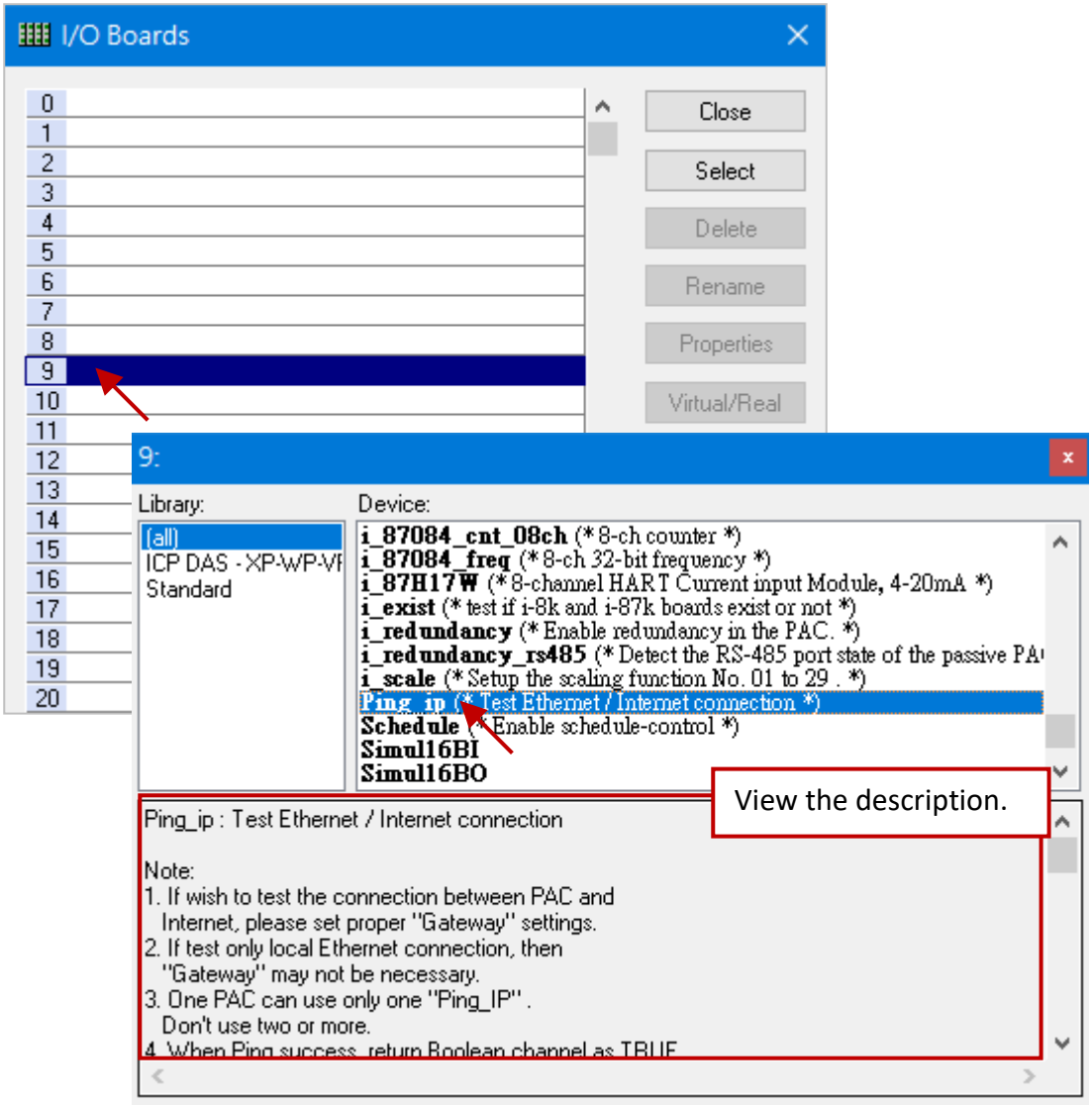
Chapter 4 Using “I/O Boards” Function

Note: **RPAC-2658M does not support Local I/O. For connecting with remote I/O modules, refer to Chapter 8 – Connecting DCON I/O modules.**

1. In the Win-GRAF, click the "Open I/Os" button from the toolbar to open the “I/O Boards” window.



2. Double-click any number to choose the needed function.

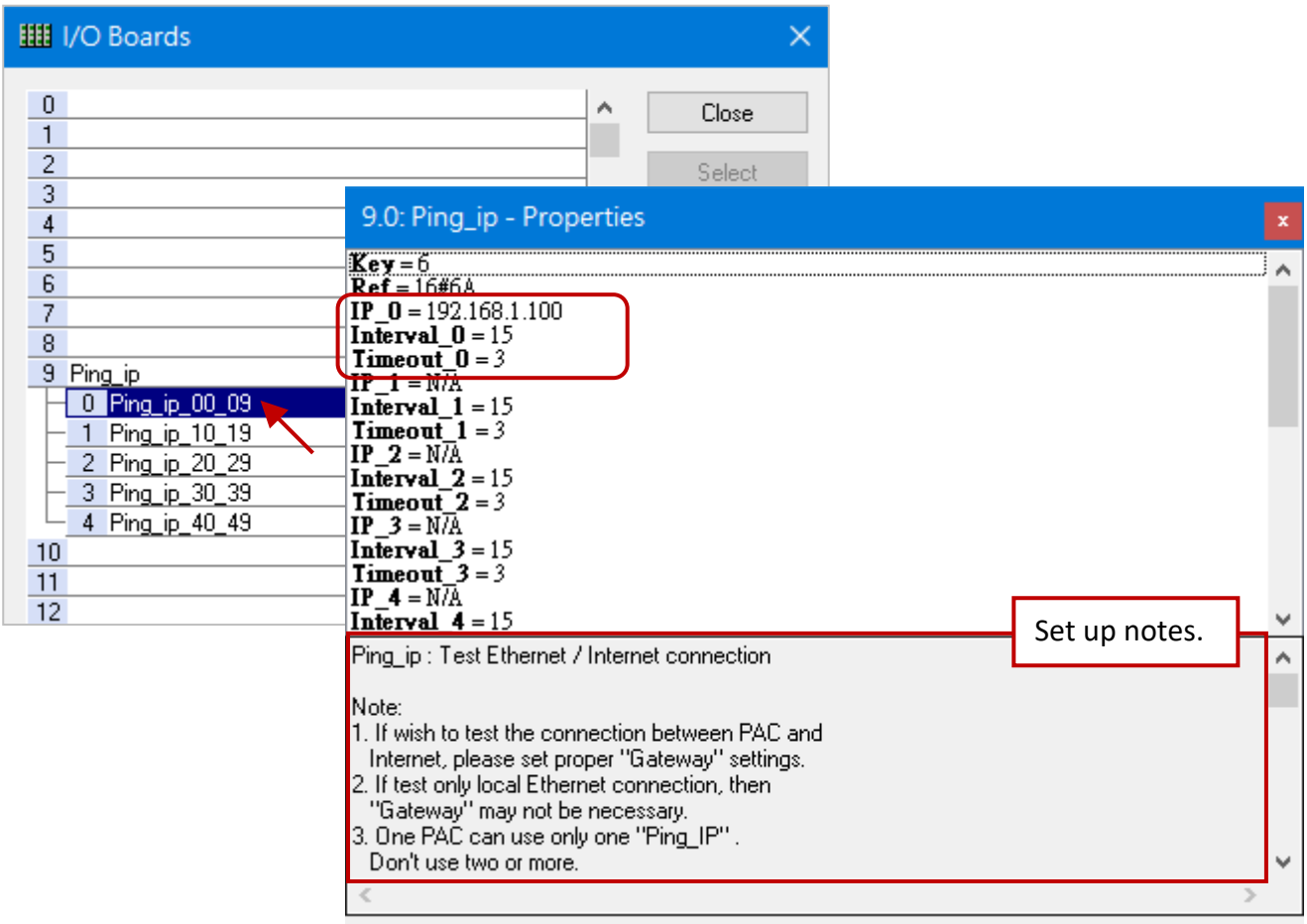


Note: “i_8xxx” or “i_87xxx” is only available for the PAC that supports the local I/O module.

4.1 Ping_ip: Check the Connection of an Ethernet/ Internet Device

The "Ping_ip" function can be used to know if the Ethernet/ Internet connection of the remote device is working. Up to 50 IP settings can be used. Refer to Chapter4 to add this I/O board.

1. Double-click the "0: Ping_ip_00_09" to open the “Properties” window.



Note:

1. To detect the connection between PAC and the Internet, set the "Gateway" properly.
2. When detecting the Ethernet LAN connection, the "Gateway" setting is necessary.
3. Only one "Ping_IP" can be used for each PAC.
4. When the connection (Ping) is successful, a boolean value "TRUE" will be returned.
5. When the connection (Ping) fails, it will try again. If it still fails, a boolean value "FALSE" will be returned.

Parameters:

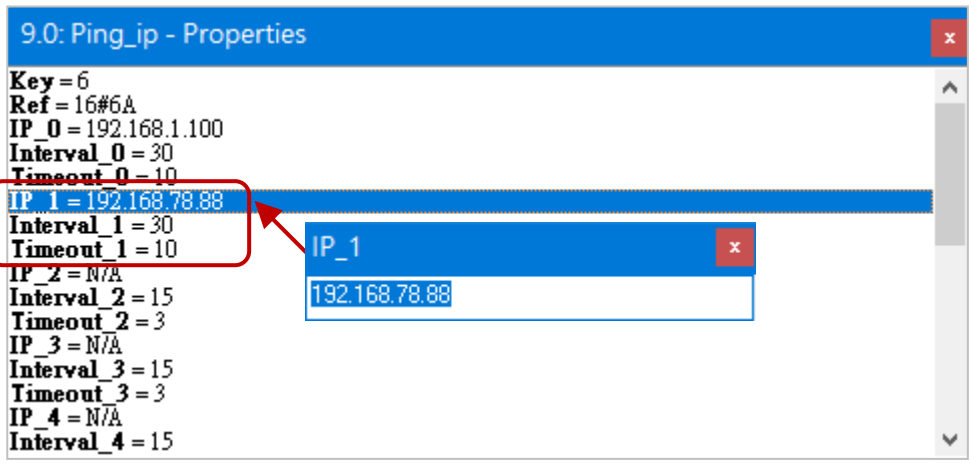
IP_01 to IP_49: (Data type: "STRING")
The IP address of targets. Set as 'N/A' if wish to disable it.
For example, 192.168.1.100 or 52.19.125.242 or N/A.

Interval_01 to Interval_49: (Data type: "DINT")
The unit is second. The interval to send one "ping" command is 15 seconds by default. The valid value used in the PAC is from 6 to 86,400 seconds.

Timeout_01 to Timeout_49: (Data type: "DINT")
The unit is second. The timeout setting of the "ping" command is 3 seconds by default. The valid value used in the PAC is from 2 to 30.

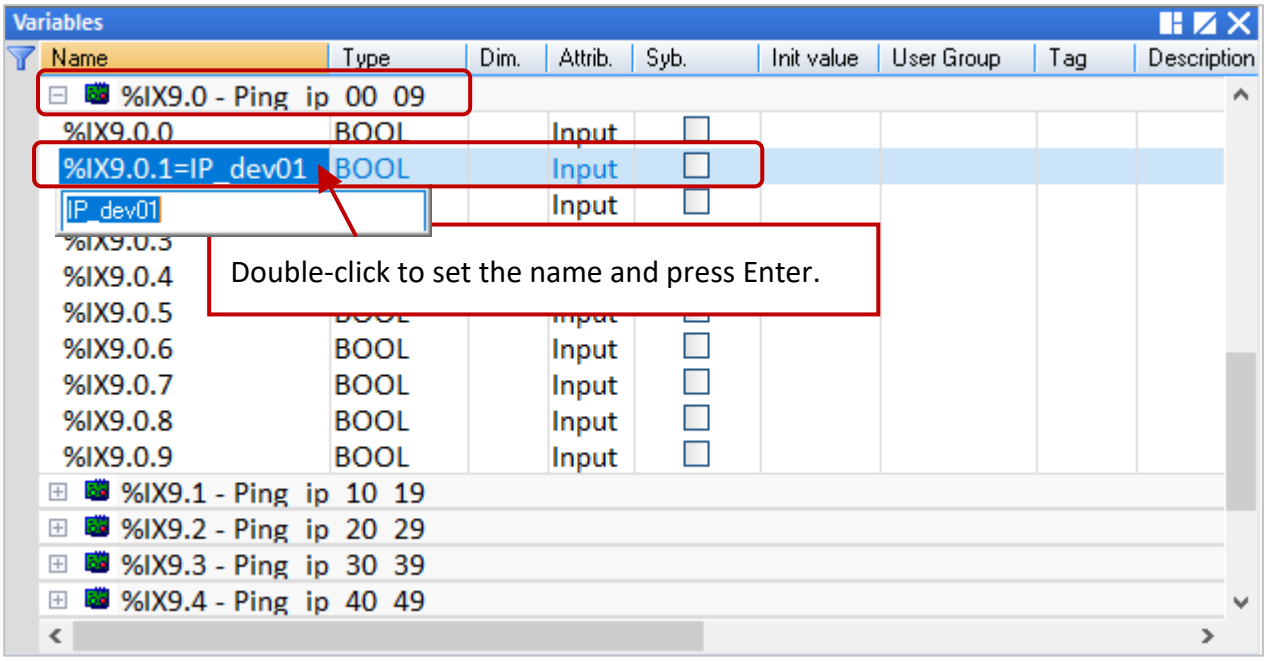
Note: The valid **Interval_x** value taken in the PAC **must be at least triple** the **Timeout_x** value. For example, if "Timeout_0" is set as 10 however "Interval_0" is set as 20, then the "Interval_0" value used in the PAC will be 30.

2. Double-click the item to fill in the value, then press the “Enter” key to complete the setting.



3. After configuring the "Ping_ip" function in the “I/O Boards” window, 50 “BOOL” Input variables will be automatically added to the “Variables” window. While connecting with PAC, the status of the connection for the remote device will be displayed.

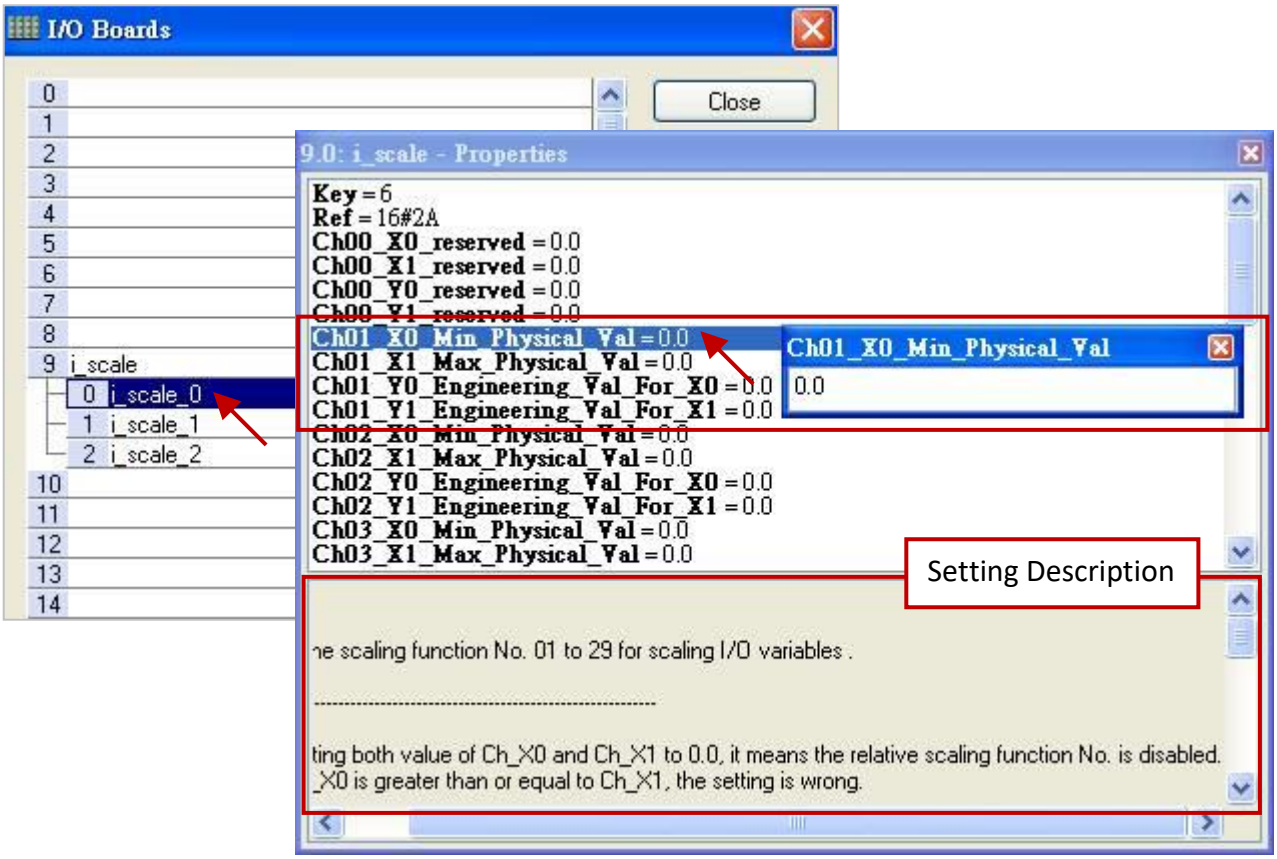
TRUE: The connection is ok.
FALSE: Connection failed or cable problem.



4.2 i_scale: Scale Conversion

The “i_scale” function provides up to 29 scale conversion settings to convert a physical I/O value to an engineering value. Refer to Chapter4 to add this I/O board.

1. Double-click the "i_scale_0" (or "i_scale_1" or "i_scale_2") to open the “Properties” window and then to view the description.



Parameters: (‘**Ch’ stands for Ch01 to Ch29;** "Ch00” is a reserved item)

Ch_X0_Min_Physical_Val:	The minimum value of an AI (or AO) module (X0).
Ch_X1_Max_Physical_Val:	The maximum value of an AI (or AO) module (X1).
Ch_Y0_Engineering_Val_For_X0:	The engineering value after scaling X0.
Ch_Y1_Engineering_Val_For_X1:	The engineering value after scaling X1.

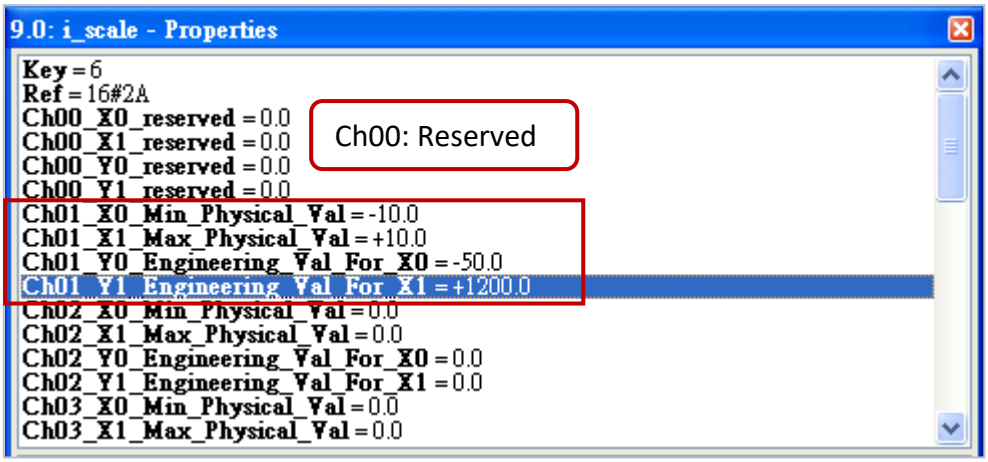
2. Double-click the desired item and enter a value, then press the “Enter” key to set.

Notice:

1. If the **Ch_X0** and **Ch_X1** are set to "0.0", which means the scaling function No. is disabled.
2. If the **Ch_X0** is greater than or equal to **Ch_X1**, which means the setting is wrong.
3. If the **Ch_Y0** is equal to **Ch_Y1**, which means the setting is wrong.

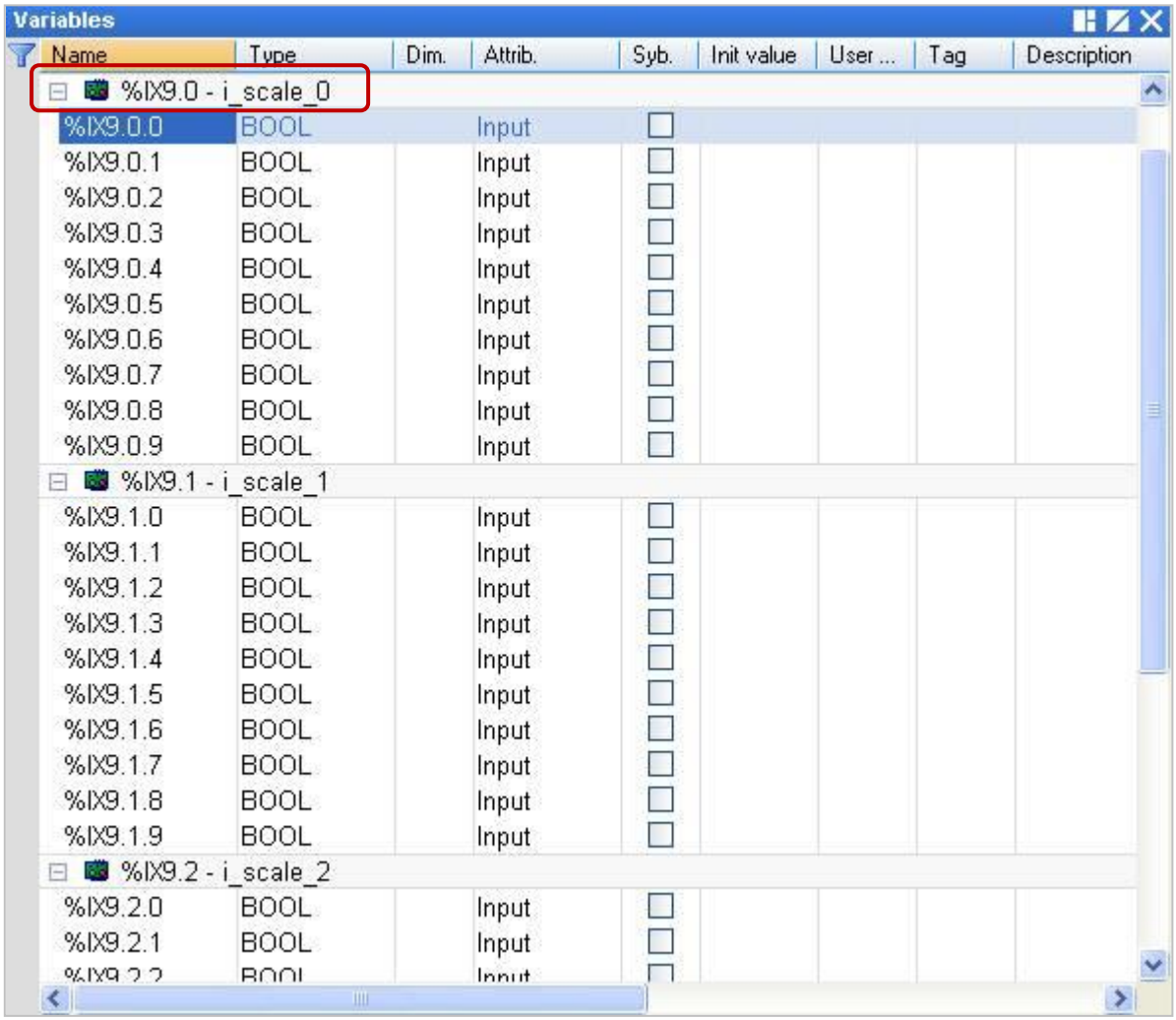
For example, for converting an AI value from 4 to 20 mA to an engineering value from 0 to 10000, set Ch_**X0** as "4.0", Ch_**X1** as "20.0", Ch_**Y0** as "0.0", and Ch_**Y1** as "10000.0".

For example, for converting an AO value from -10 to +10 V to an engineering value from -50 to 1200, set Ch_**X0** as "-10.0", Ch_**X1** as "+10.0", Ch_**Y0** as "-50.0", and Ch_**Y1** as "+1200.0".



3. After configuring the “i_scale” in the “I/O Boards” window, 30 Boolean variables will automatically be added. When connecting to the PAC, the status of the scale conversion will be displayed in the “Variables” window.

   True: scaling function is ok.
   FALSE: scaling function is not enabled or setting error.

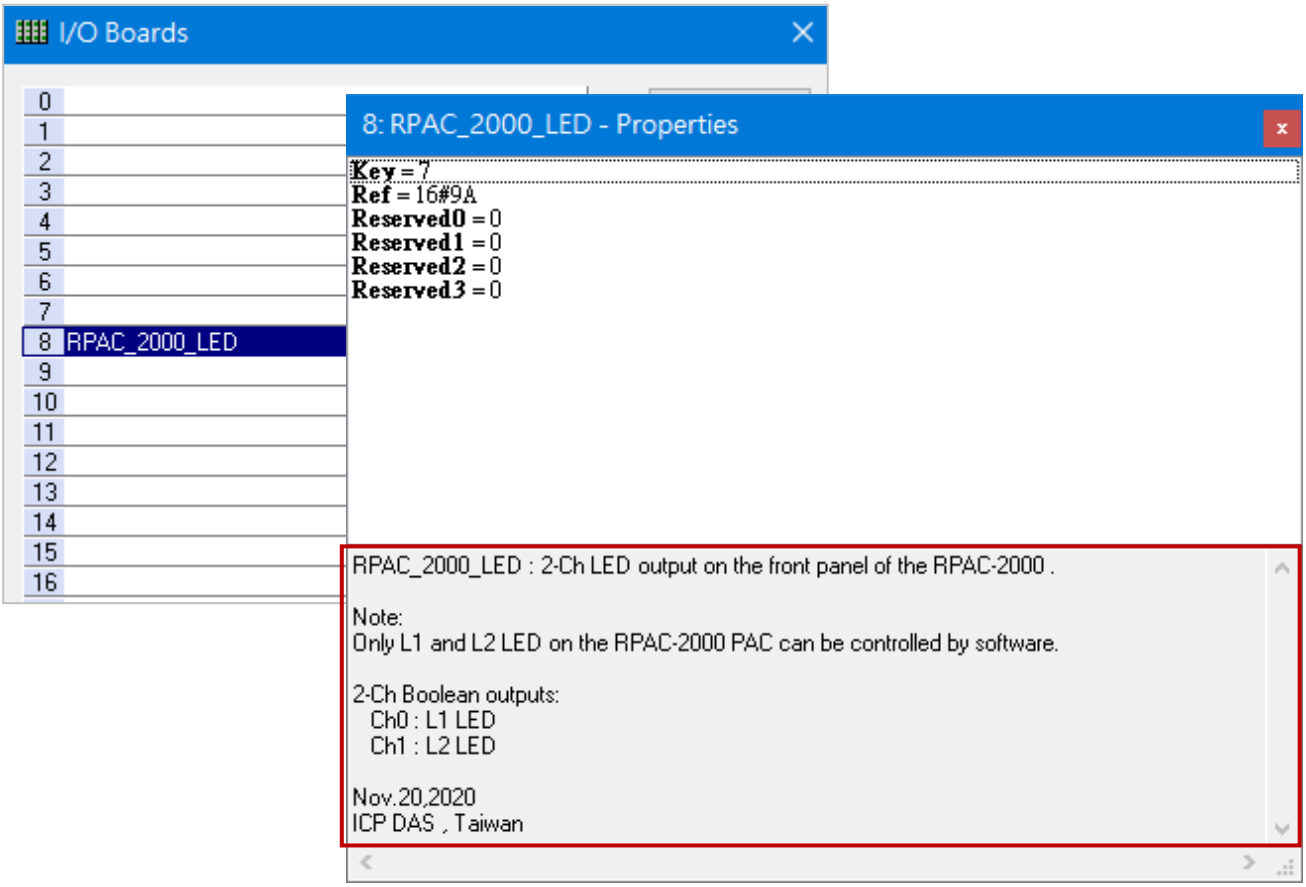


4.3 RPAC_2000_LED: Control the L1 and L2 LED on the RPAC-2658M

The RPAC_2000_LED function can be used to control the L1 and L2 LED on the front panel of the RPAC-2658M.

Note: Only L1 and L2 on the PAC can be controlled by software. Refer to Chapter4 to add this I/O board.

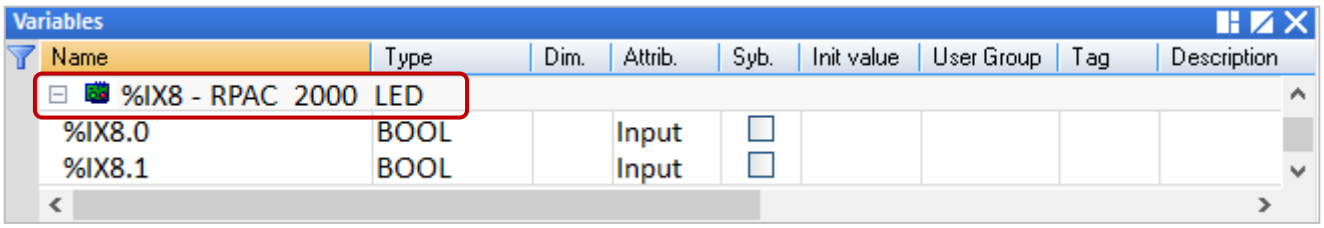
1. Double-click the "RPAC_2000_LED" to open the “Properties” window and then to view the description.



Parameters:

Ch0, Ch1: (Data Type: "BOOL") indicates L1 and L2 LED.

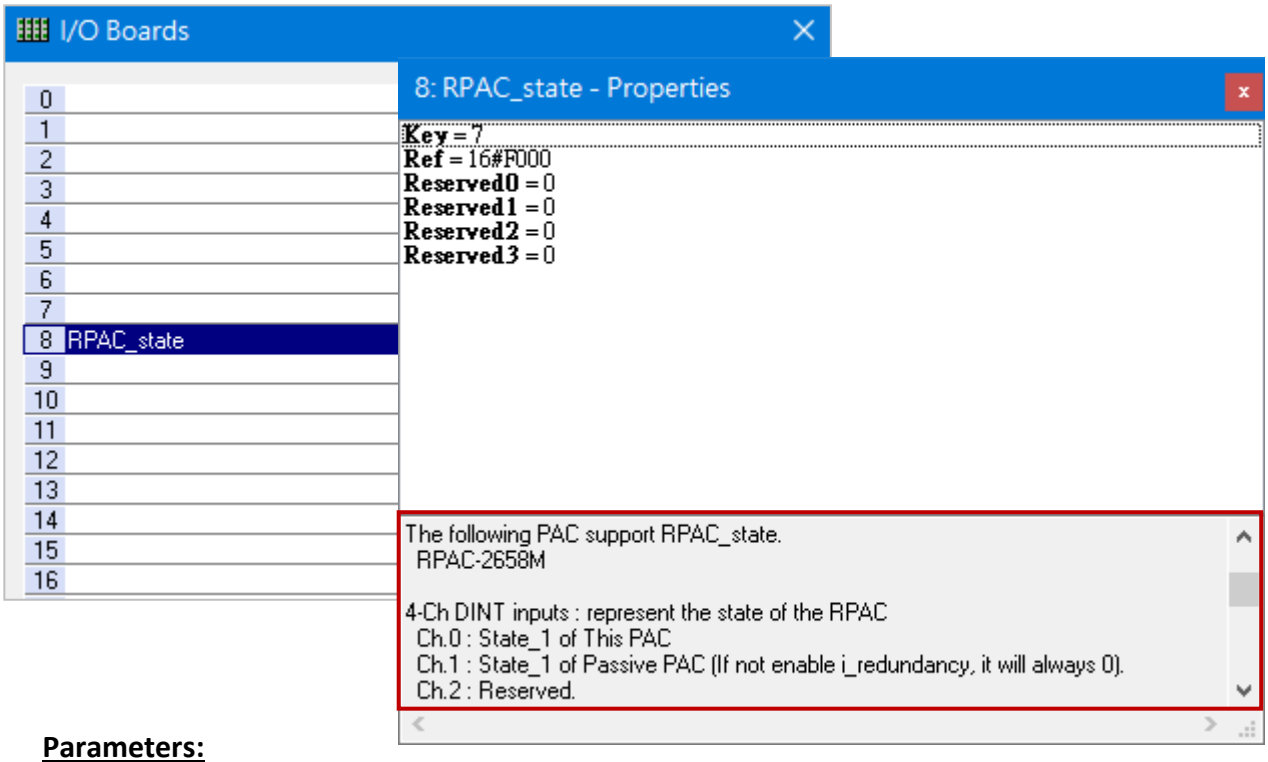
2. After adding the “RPAC_2000_LED” function in the “I/O Boards” window, two Boolean variables will automatically be added. When connecting to the PAC, the status of the LED will be displayed in the “Variables” window.



4.4 RPAC_PAC_state: Detect the LAN state of the RPAC-2658M

The RPAC_PAC_state function can be used to detect the state of the RPAC-2658M. Refer to Chapter4 to add this I/O board.

1. Double-click the "RPAC_PAC_state" function to open the “Properties” window and then to view the description.



Parameters:

4-Ch DINT inputs that represent the state of the RPAC PAC

Ch.0: State_1 of This PAC

Ch.1: State_1 of Passive PAC (If the i_redundancy is disabled, it will always 0).

Ch.2, Ch.3: Reserved.

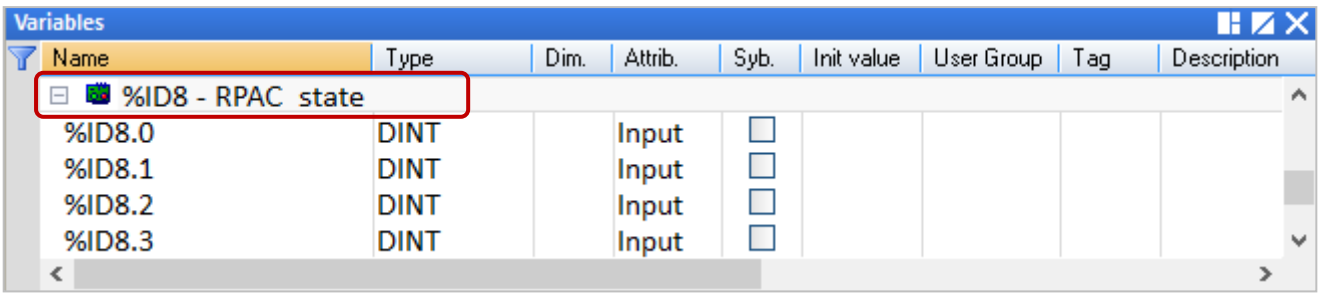
State :

bit0 to bit2: The status of LAN1, LAN2, and LAN3. 0: OK, 1: Failed or Not wired.

bit3: The status of the hardware (e.g., EEPROM, Disk, RAM). 0: OK, 1: Failed.

bit4: The status of I/O slot - 0: OK, 1: Failed. (Only for local I/O slots of PAC)

2. After configuring the “RPAC_PAC_state” function in the “I/O Boards” window, four DINT variables will automatically be added. When connecting to the PAC, the status of the PAC will be displayed in the “Variables” window.

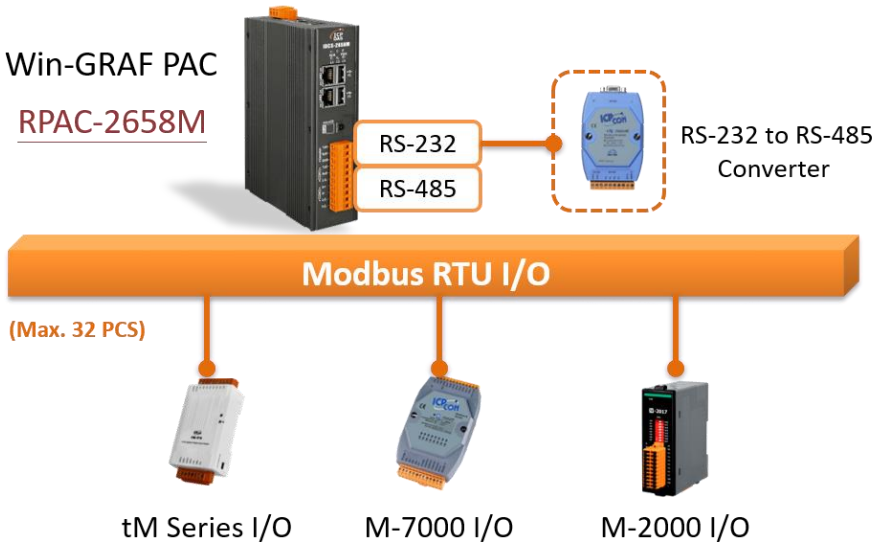


Chapter 5 Modbus Master: connecting to Modbus Slave Devices

This chapter lists the way to set the Win-GRAF PAC to act as a Modbus Master to connect Modbus RTU/ASCII Slave or Modbus TCP/UDP Slave devices.

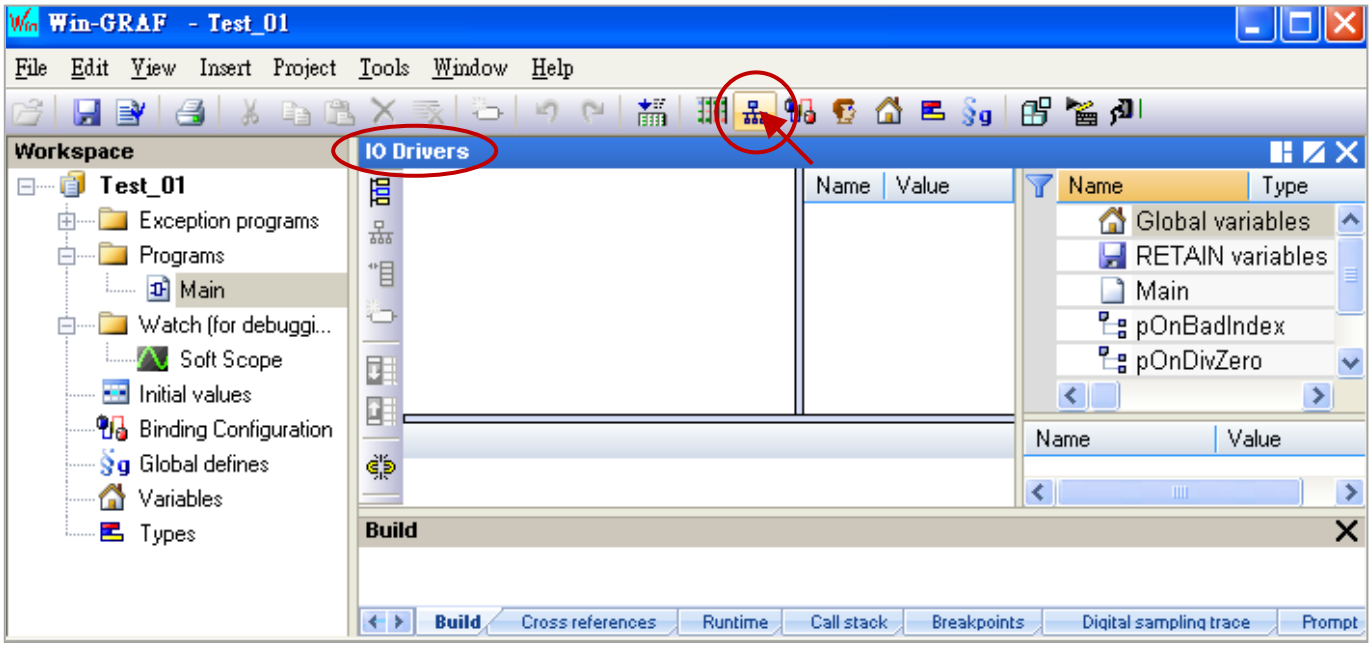
5.1 Enabling Win-GRAF PAC as a Modbus RTU/ASCII Master

Application Diagram:

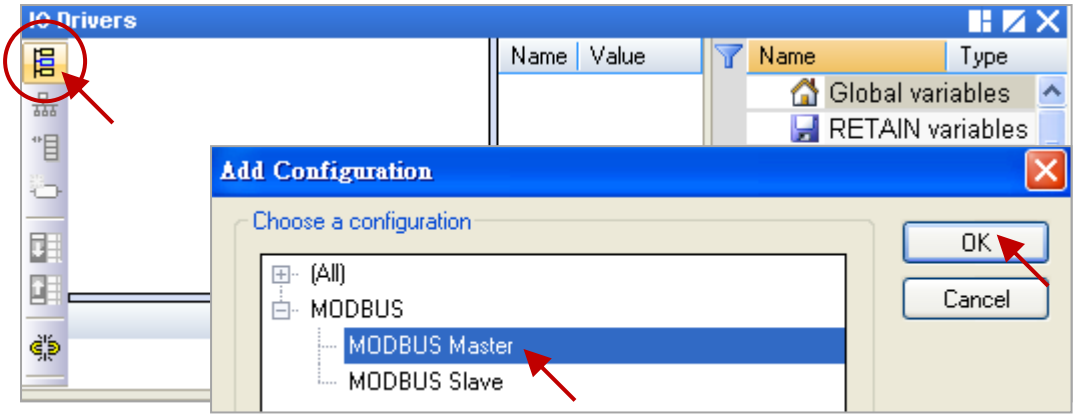


Follow the steps below:

1. Click the “Open Fieldbus Configuration” button on the toolbar to open the “IO Drivers” window.



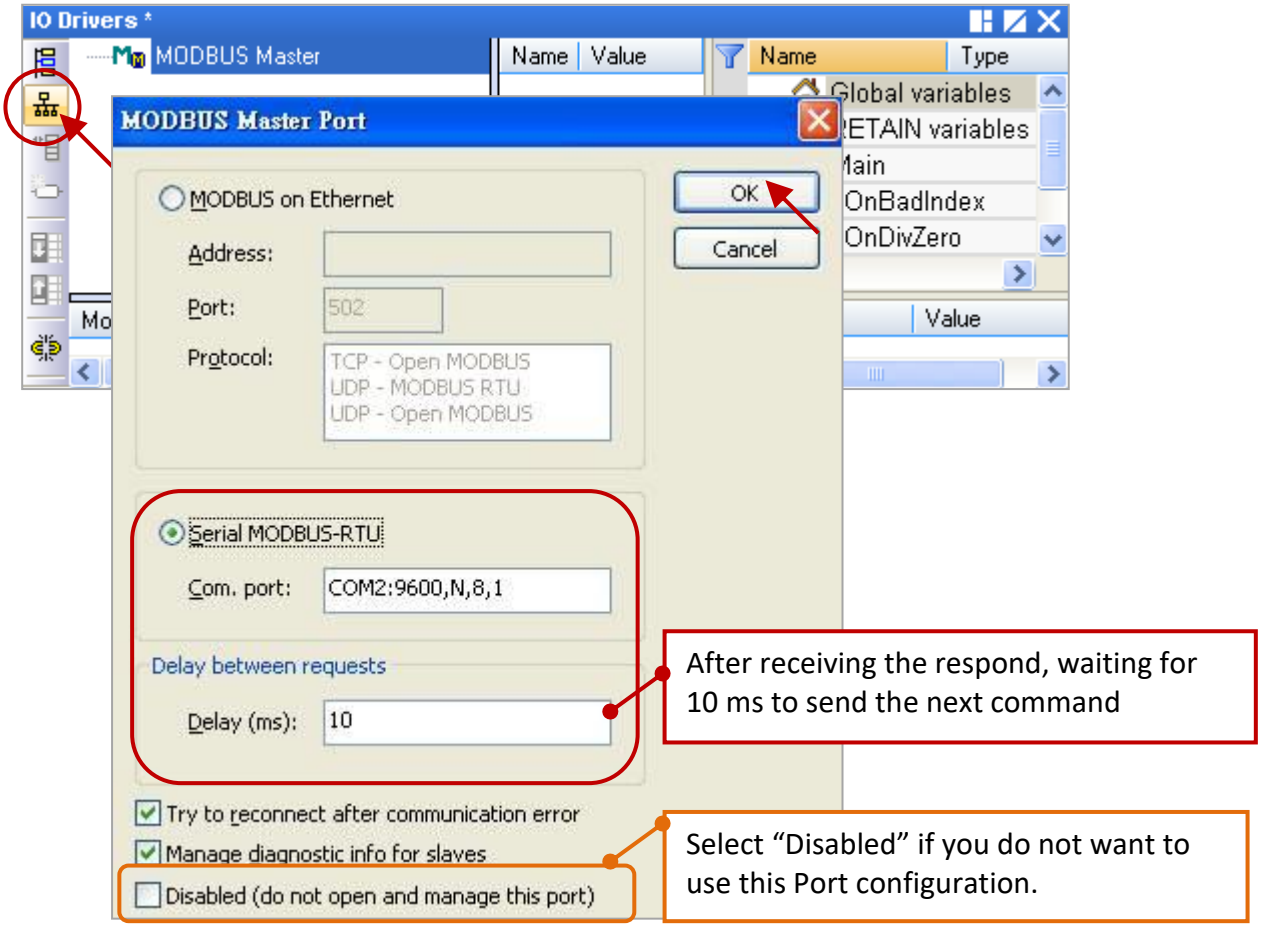
2. **Set the PAC to act as a Modbus Master**. Click the “Insert Configuration” button on the left of the “IO Drivers” window and click the “MODBUS Master”, then click “OK”.



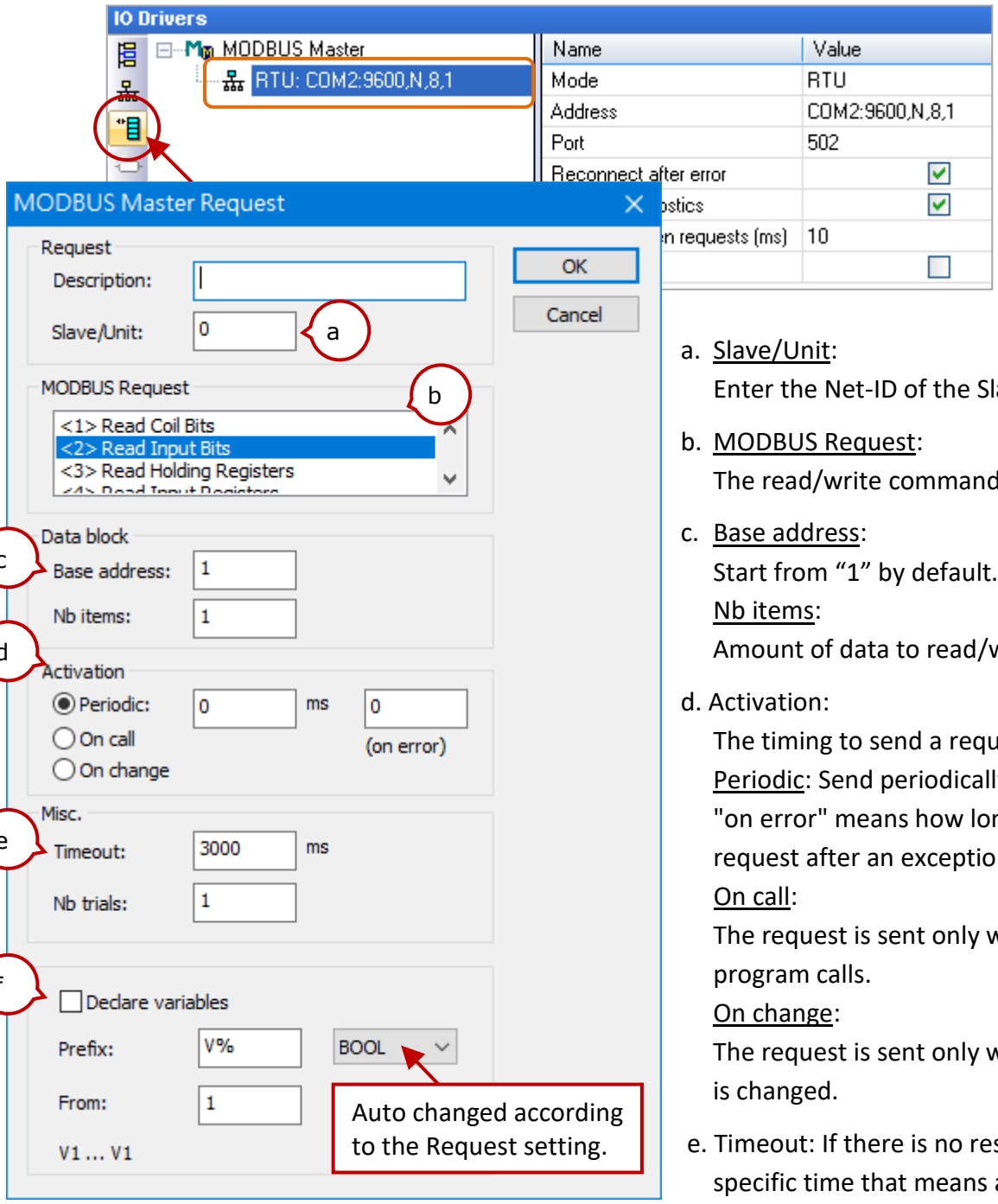
3. **Choose the serial port.** Click the “Insert Master/Port” button on the left side to open the settings window. Select “Serial MODBUS-RTU” and set the COM Port and the Delay time, then click OK.

<u>COM Port</u>:
When using the Modbus **RTU** protocol, enter **COM2:9600,N,8,1.**
When using the Modbus **ASCII** protocol, enter **ASCII:COM2:9600,N,8,1.**

<u>Delay (ms):</u>
Recommended to set as 10 (ms), the value can be 0 to 10000.



4. **Create a data block.** Click the “Insert Slave/Data Block” button to open the settings window.



a. Slave/Unit:
Enter the Net-ID of the Slave device.

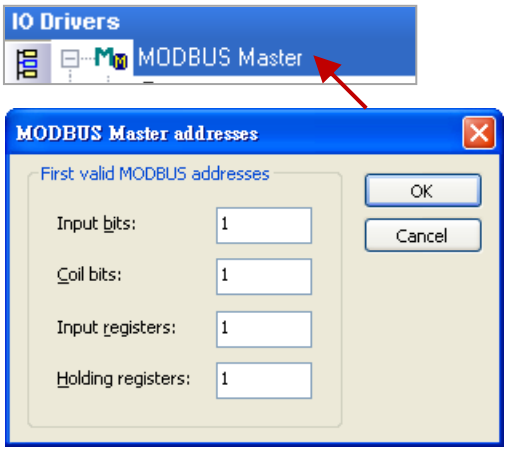
b. MODBUS Request:
The read/write commands.

c. Base address:
Start from “1” by default.
Nb items:
Amount of data to read/write.

d. Activation:
The timing to send a request.
Periodic: Send periodically,
"on error" means how long to send a request after an exception occurs.
On call:
The request is sent only when the program calls.
On change:
The request is sent only when the data is changed.

e. Timeout: If there is no response in specific time that means abnormal. (Modbus RTU/ASCII: 200 to 1000 ms)

f. Declaring single or several variables with consecutive numbers according to the Request and Nb items. If "Nb items" is set to 5, check the box to add V1-V5 BOOL variables.



Note:

- If it is necessary to change the “Base address”, right-click on “MODBUS Master” and click “MODBUS Master Addresses” to modify the value.

- Also, refer to Section 5.1.6 to use the "MBRTU_M_disable" function in the program to disable the Modbus RTU/ASCII Master port.

This table lists five data blocks, and each data block stands for one Modbus Master Request.

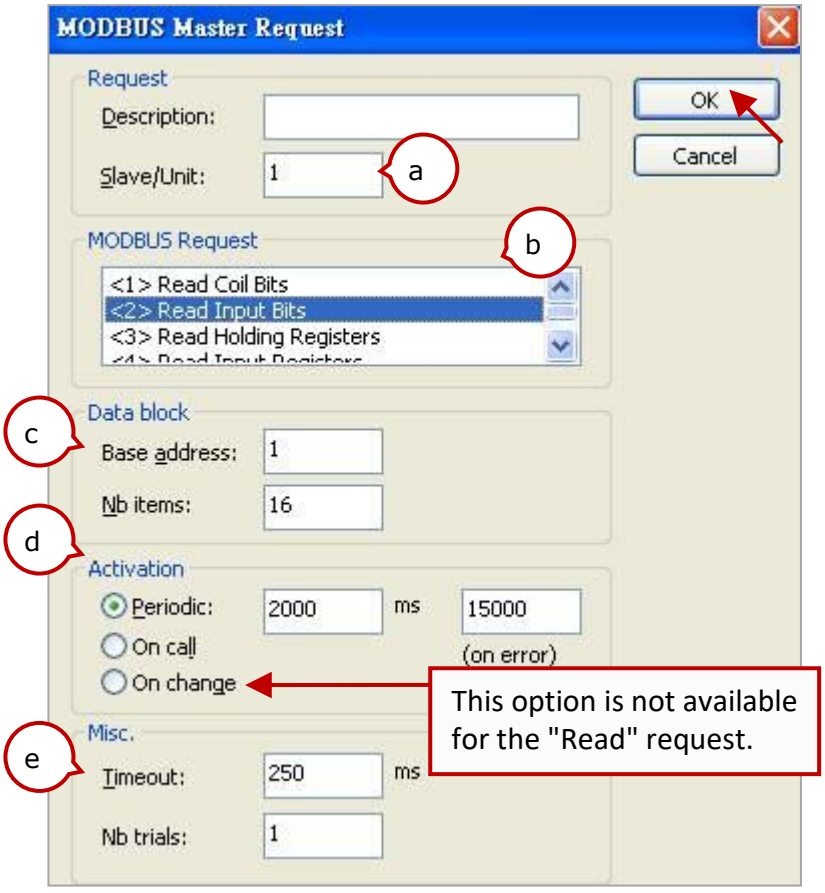
Function Code	Modbus Request	Description
2	Read Input Bits	5.1.1 Read DI data
5	Write single coil bit	5.1.2 Write DO data
4	Read Input Registers	5.1.3 Read AI data
6	Write single holding register	5.1.4 Write AO data (16-bit)
16	Write Holding Registers	5.1.5 Write AO data (32-bit)

5.1.1 Read DI data

Description:

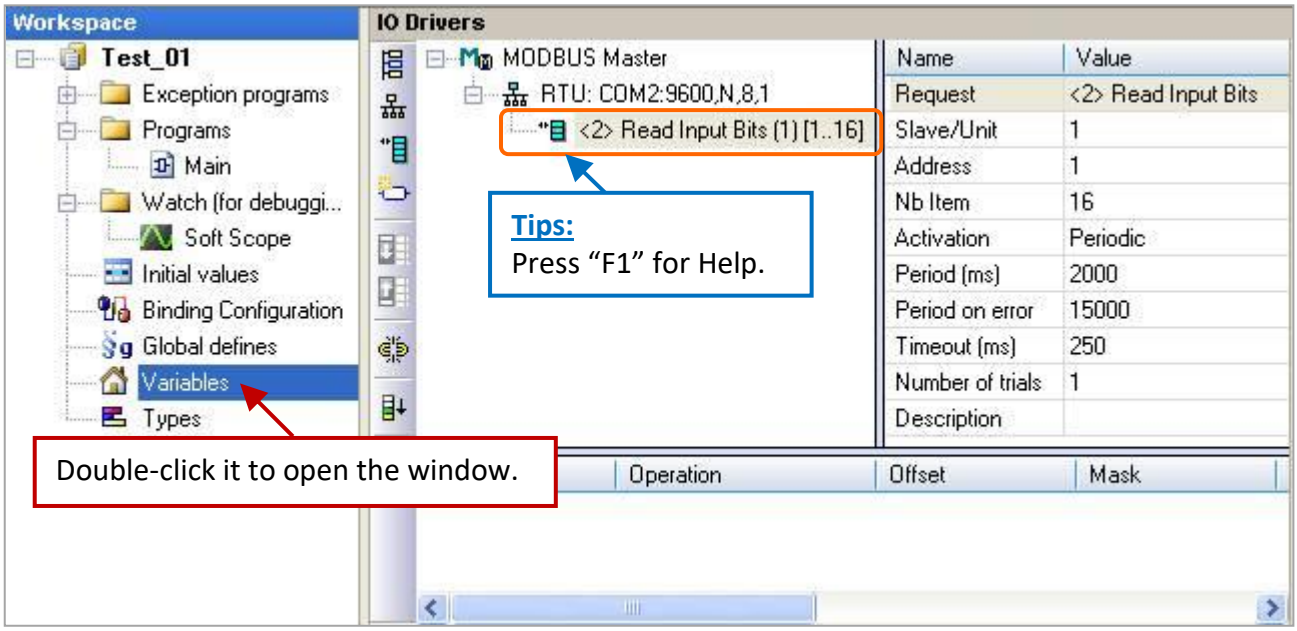
To read 16 DI values from the device (Slave ID = 1) by sending the request every two seconds. It will be sent after 15 seconds if an exception occurs. No response for 250 ms can be regarded as abnormal.

1. Completing the settings in the “MODBUS Master Request” window, and then click “OK".



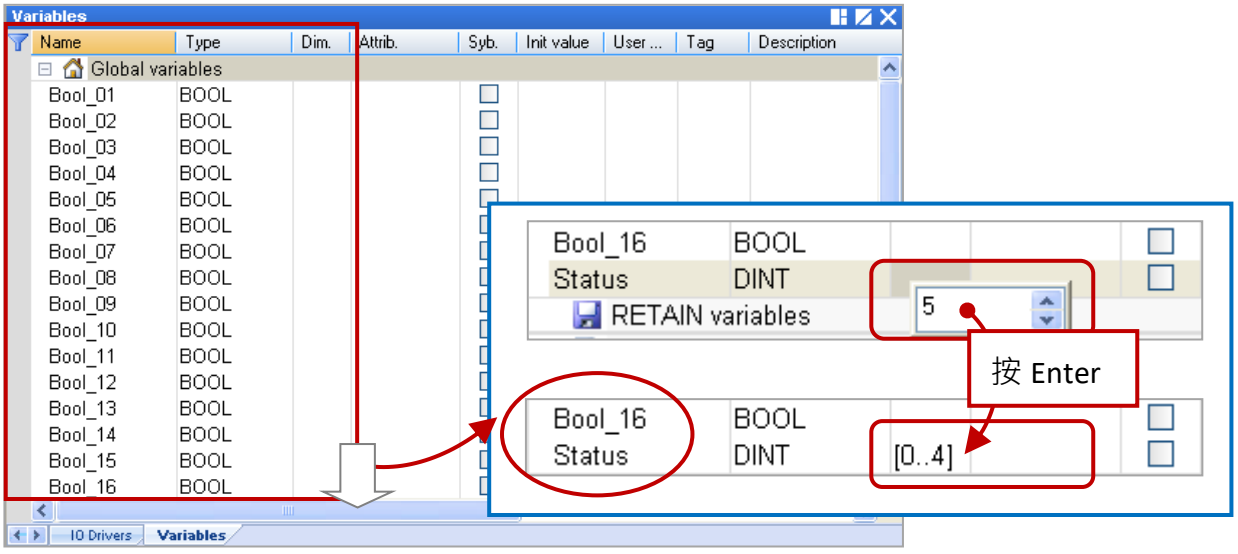
Note: Refer to step4 in Section 5.1 for the description of items.

2. Next, open the “Variables” window to add variables.



“Boo_01 to Boo_16” (BOOL): Add 16 Boolean variables to read data.
“Status" (DINT, Dim. = 5): Add one Array variable to display the access status of data.

After completing the settings, the results are shown below.

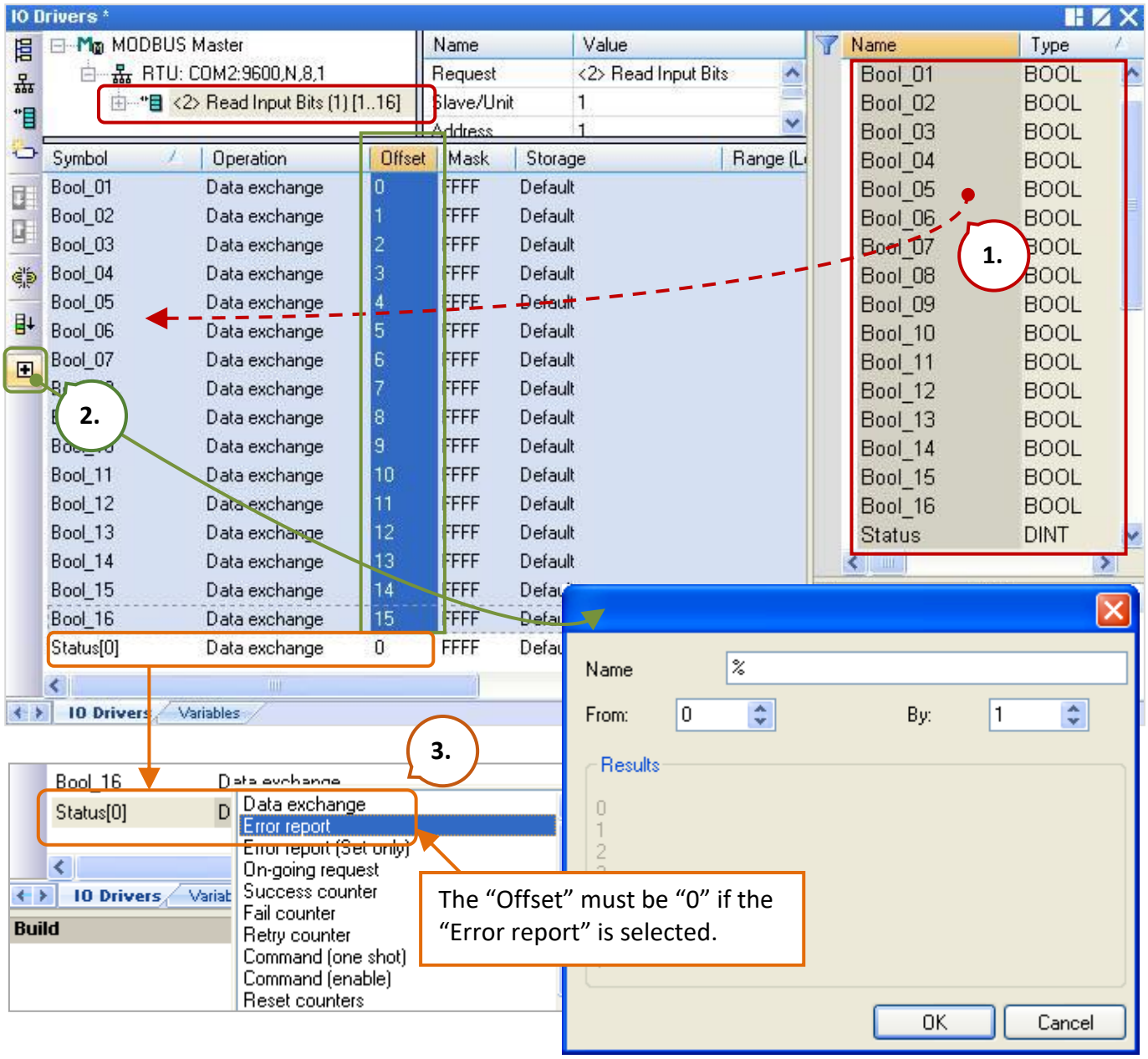


3. **Choose variables.** In the "IO Drivers" window, drag “Bool_01” to “Bool_16” and “Status” variables to the address mapping area of the data block.

Note: The “Status” is an Array variable, including Status[0] to Status[4]. Only **Status[0]** is required in the address mapping area.

4. **Configure Offset values.** Select the “Offset” field from **“Boo_01”** to **“Boo_16”** and click the “Iterate Property” button on the left side to set the “Offset” value (From: “0” ; By: “1”).

5. **Configure Operation status.**
   Set the “Operation” column of “**Status[0]**” to “**Error report**” to return an error code when fails to read data. The value will be reset to “0” when a successful read.



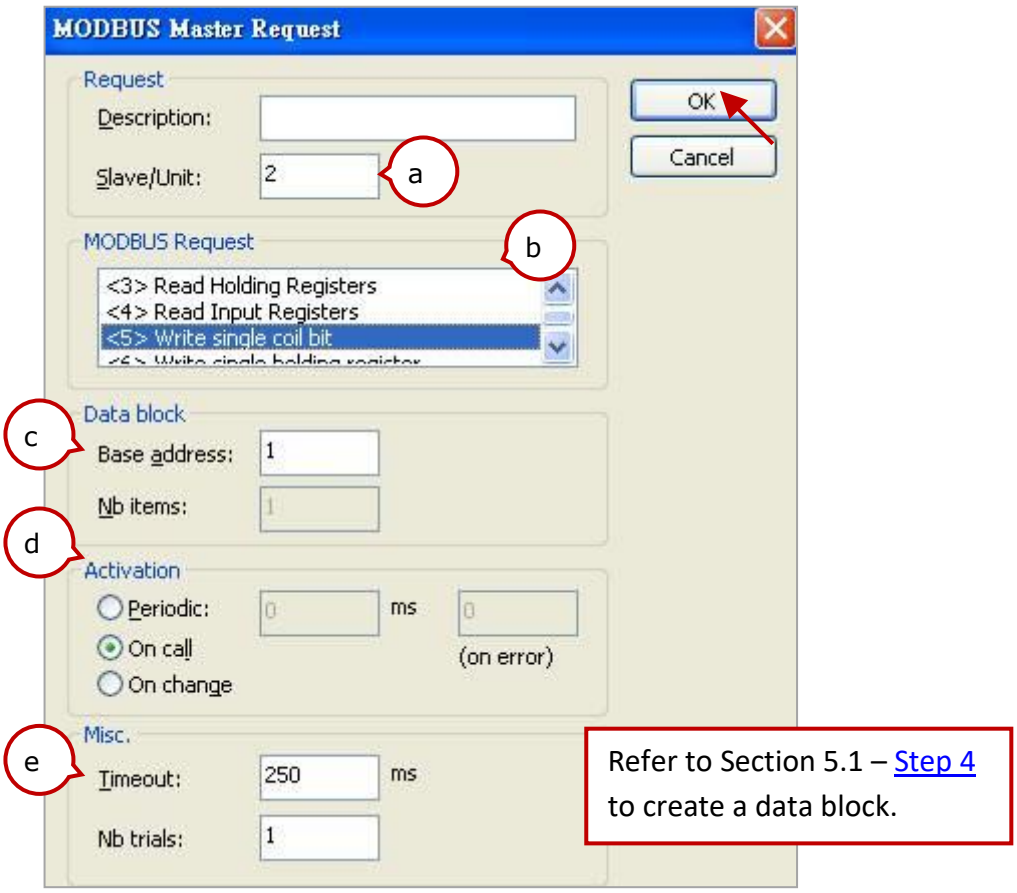
Also press “F1” in the “IO Drivers” window for more details on the Modbus Master Configuration.

Error Code	Description	Error Code	Description
0	The communication is OK.	8	Data Parity Error.
1	MODBUS function not supported.	10	Invalid gateway path.
2	Invalid MODBUS address.	11	Gateway target failed.
3	Invalid MODBUS value.	128	Communication timeout.
4	MODBUS Server failure.	129	Bad CRC16.
6	The server is busy.	130	RS-232 communication error.

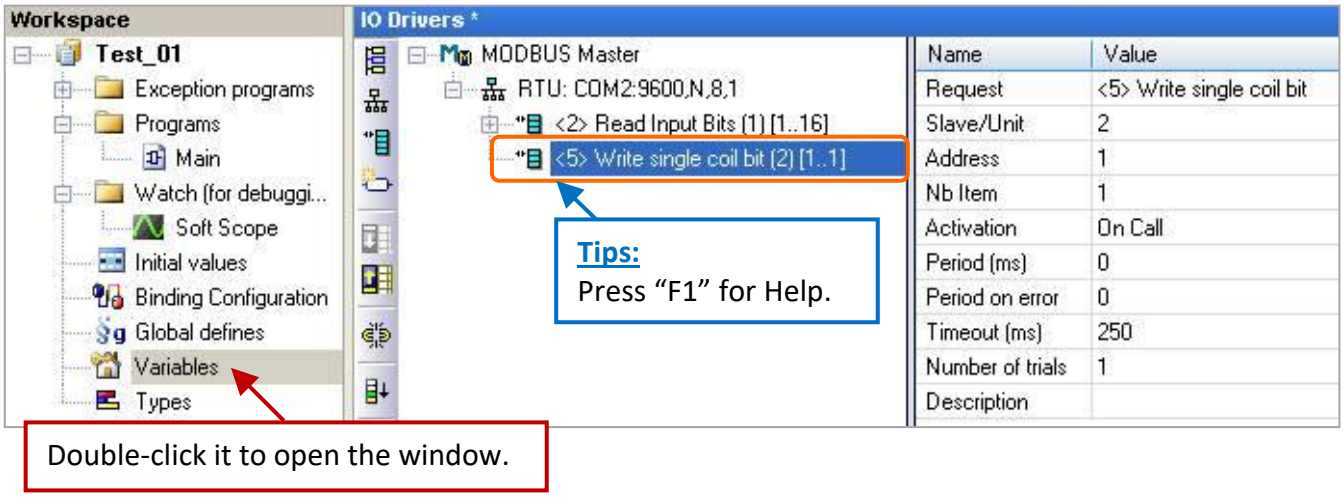
5.1.2 Write DO Data

Description:
To write a single DO data to the device (Slave ID = 2) by sending the request when the program calls. No response for 250 ms can be regarded as abnormal.

1. Completing the settings in the “MODBUS Master Request” window, and then click “OK".

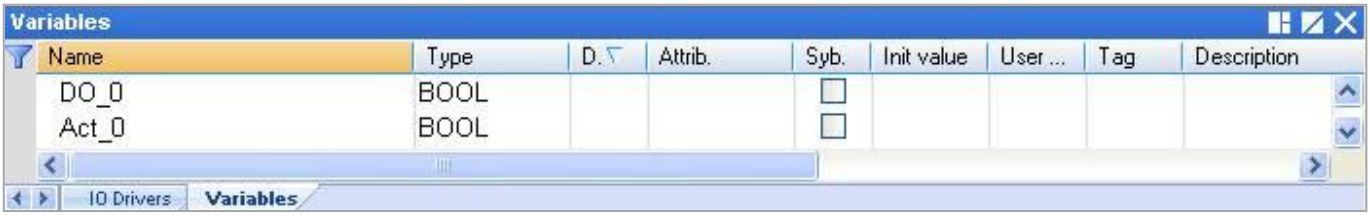


2. Open the “Variables” window and add the needed variables.



DO_0 (BOOL): Add one Boolean variable to write DO data.
Act_0 (BOOL): Add one Boolean variable to active the “On call” procedure.

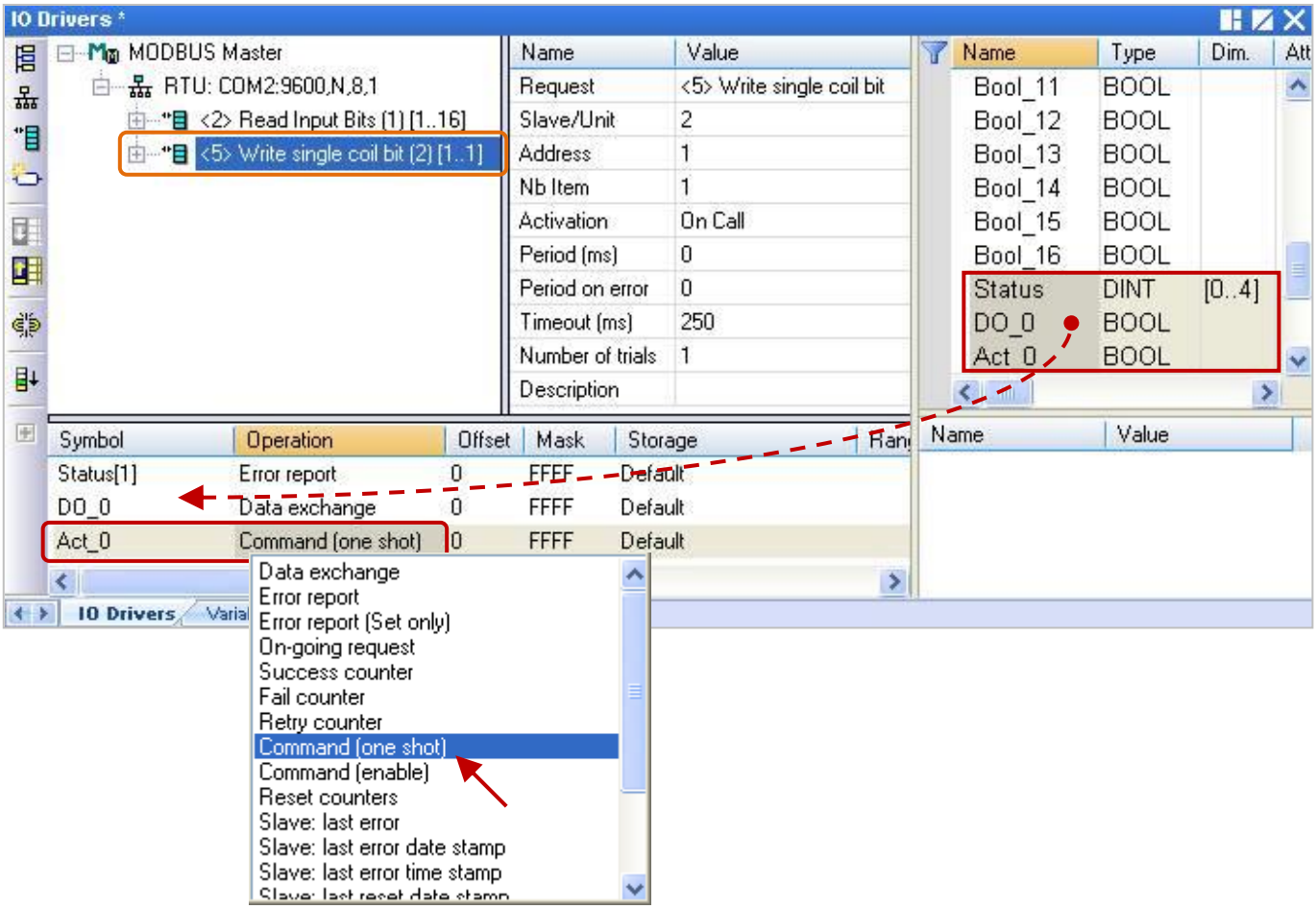
After completing the settings, the results are shown below.



3. **Choose variables.** In the "IO Drivers" window, drag “DO_0”, “Act_0” and “Status” variables to the address mapping area of the data block.

   Note: The “Status” is an Array variable, including Status[0] to Status[4]. Only **Status[1]** is required in the address mapping area.

4. **Configure Operation status.**
   1) Set the Operation of **Status[1]** to “**Error report**” which means to return an error code when fails to read data and be reset to 0 when a successful read.
   2) Set the **Act_0** entry to “**Command (one shot)**” which means to send a request once when “Act_0” is set to **TRUE.** After that, set it automatically to FALSE. "Command (Enable)“ means that the request will be sent continuously when “Act_0” is set to “TRUE”. Set it to FALSE to stop sending a command.

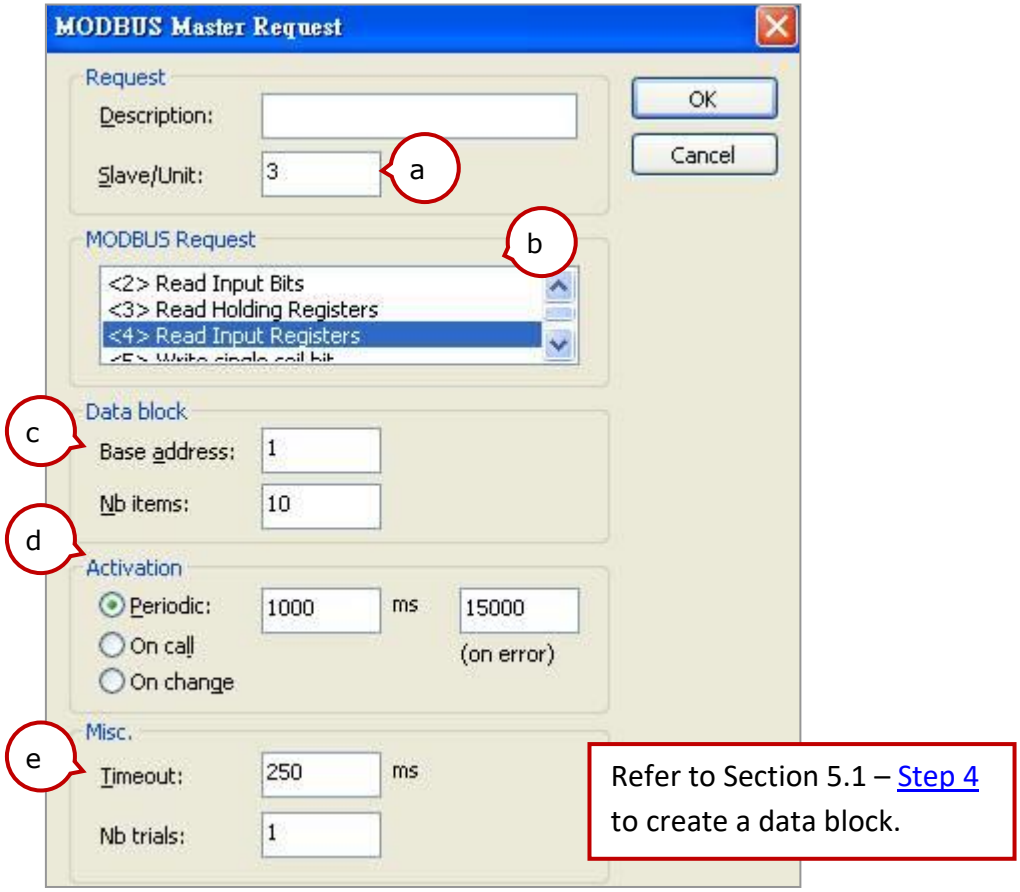


5.1.3 Read AI Data

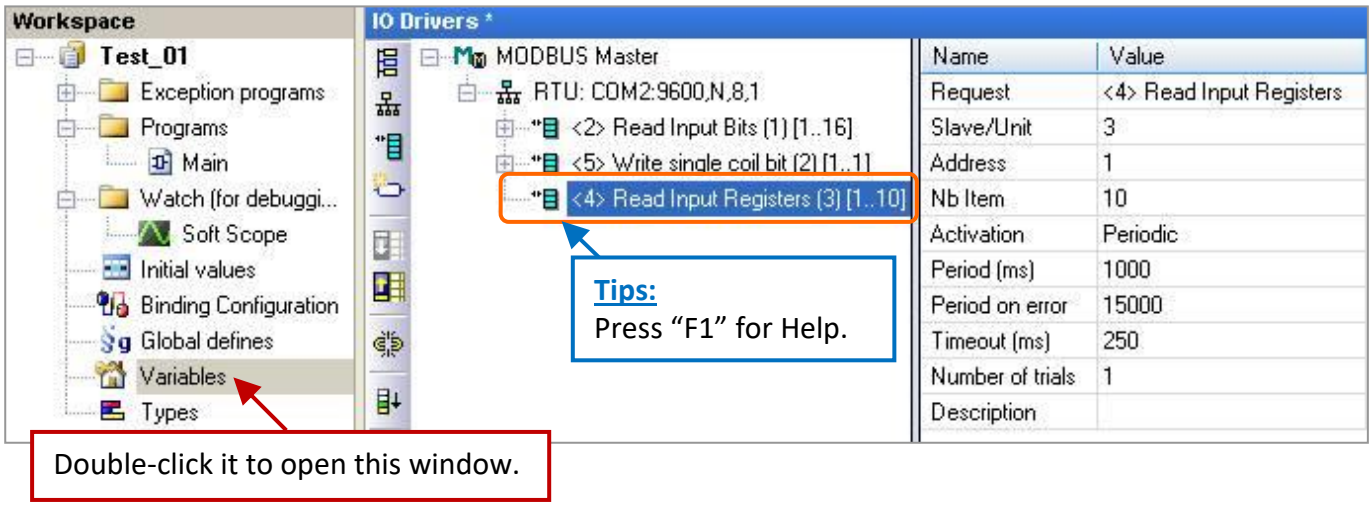
Description:

To read 10 AI data from the device (Slave ID = 3) by sending the request every second. It will be sent after 15 seconds if an exception occurs. No response for 250 ms can be regarded as abnormal.

1. Completing the settings in the “MODBUS Master Request” window, and then click “OK".



2. Open the “Variables” window and add the needed variables.

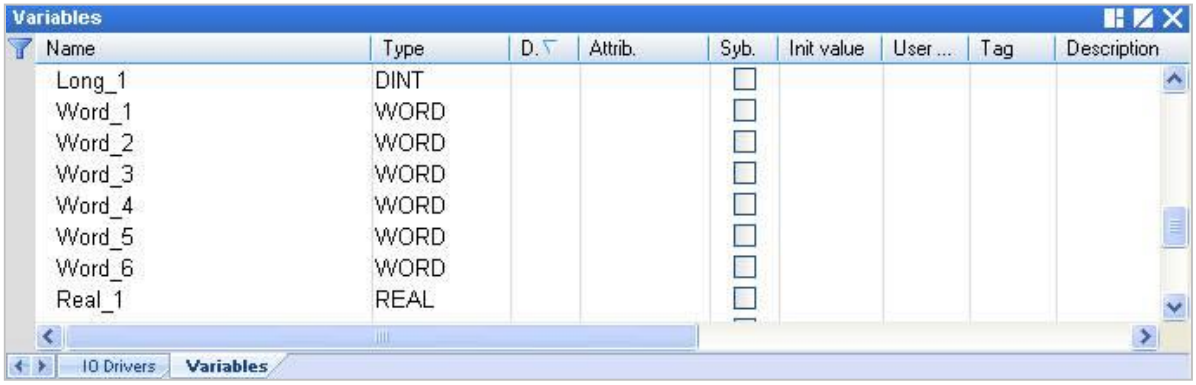


Word_1 to Word_6 (WORD): Add six WORD (16-bit) variables to read AI data.

Long_1 (DINT): Add one DINT (32-bit) variable to read the AI data.

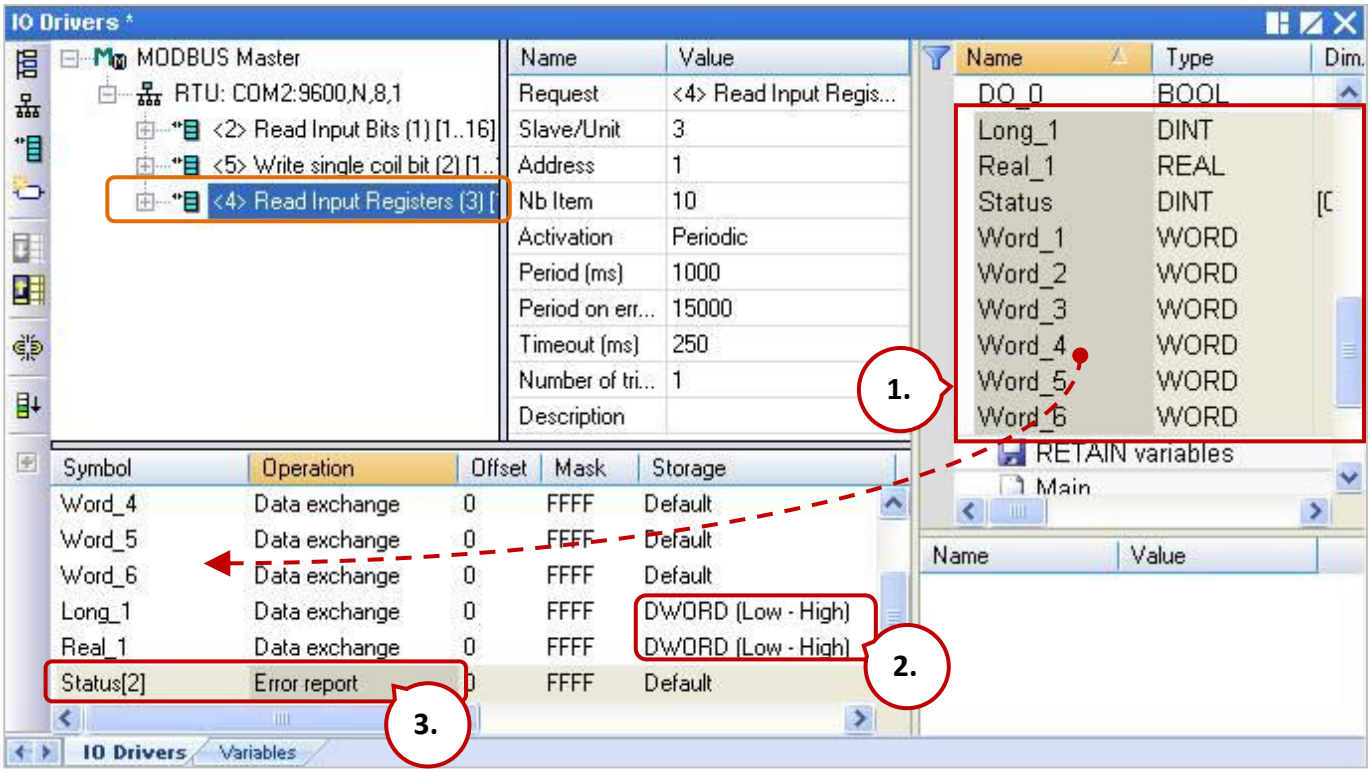
Real_1 (REAL): Add one REAL (32-bit) variable to read the AI data.

After completing the settings, the results are shown below.



3. **Choose variables.**
   In the "IO Drivers" window, drag "Word_1" to "Word_6", “Long_1”, “Real_1”, and “Status” variables to the address mapping area of the data block.

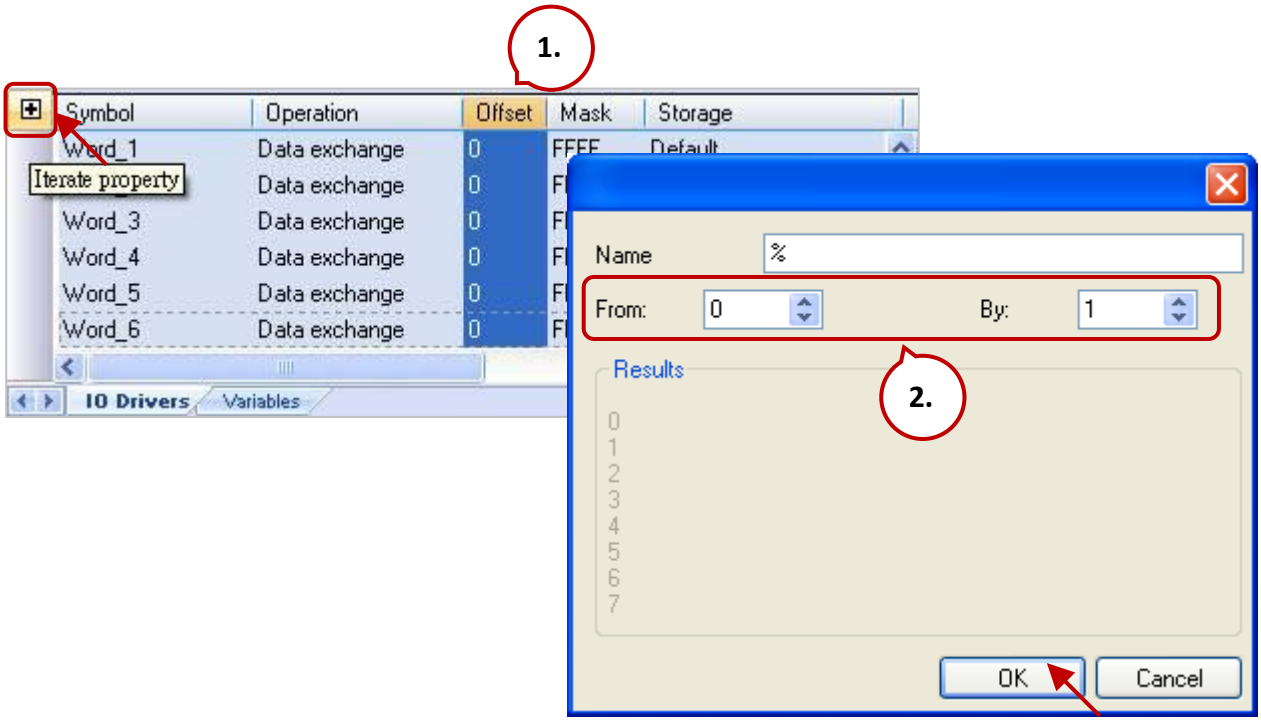
   Note: The “Status” is an Array variable, including Status[0] to Status[4]. Only **Status[2]** is required in the address mapping area.



4. **Configure Storage format.**
   Set the Storage of **Long_1** and **Real_1** (32-bit) to “**DWORD (Low – High)**”, mapping two Word (16-bit).

5. **Configure Operation status.**
   Select the Operation of **Status[2]** to “**Error report**” which means to return an error code when fails to read data and reset to 0 when a successful read.

6. **Configure Offset values.**
   1) Select the “Offset” field from "**Word_1 to Word_6**" and click the “Iterate Property” button on the left side to set the “Offset” value (From: “0” ; By: “1”).
   2) Double-click the Offset field of “**Long_1**” (32-bit) and fill in “6”, then press the “Enter” key.
   3) Double-click the Offset field of “**Real_1**” (32-bit) and fill in “8”, then press the “Enter” key.



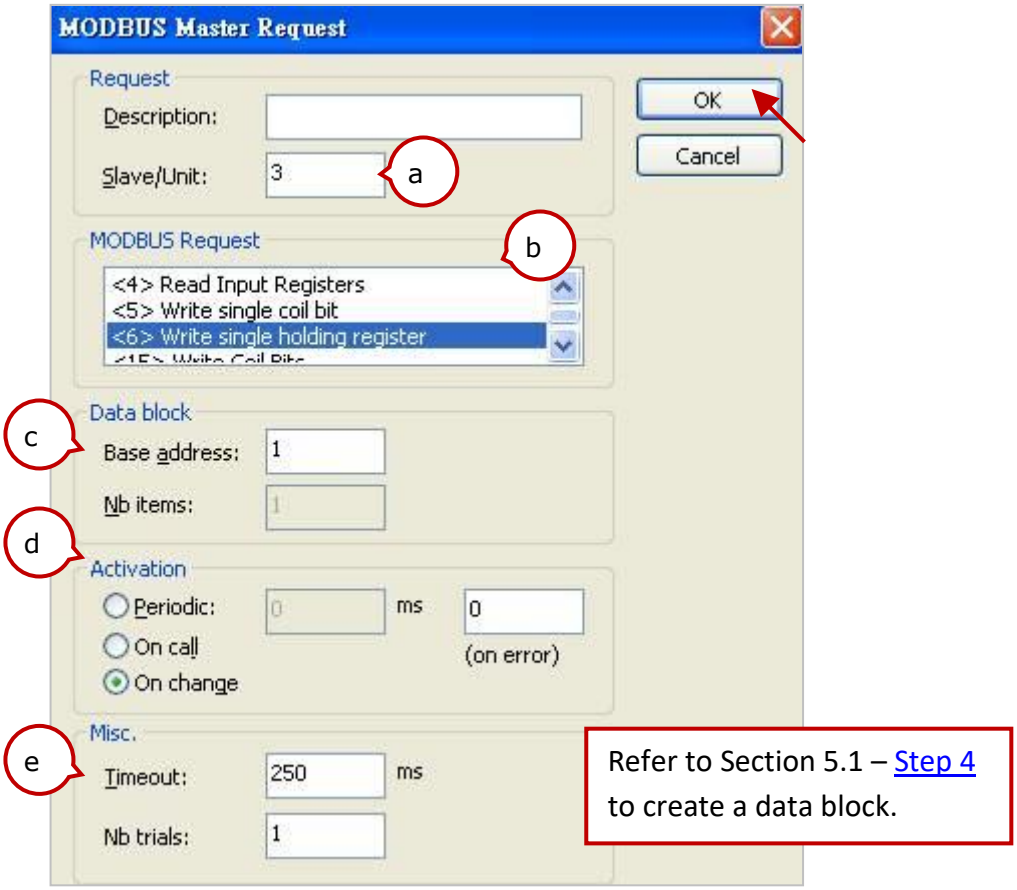
Note: **T**wo Modbus addresses are required for 32-bit data.

5.1.4 Write AO Data (16-bit)

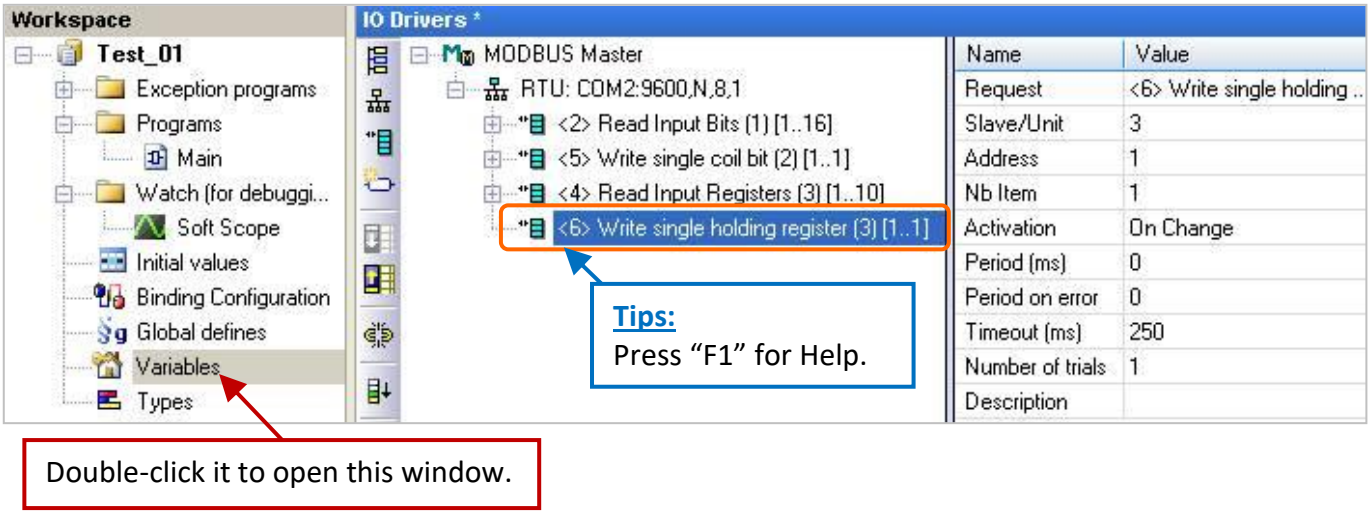
Description:

To write a single AO data to the device (Slave ID = 3) by sending the request when the data is changed. No response for 250 ms can be regarded as abnormal.

1. Completing the settings in the “MODBUS Master Request” window, and then click “OK".

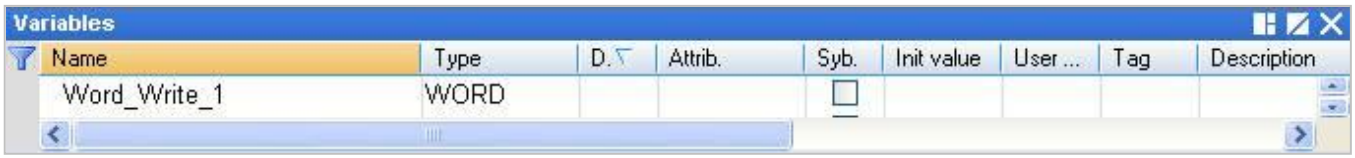


2. Open the “Variables” window and add the needed variables.



Word_Write_1 (WORD): Add one WORD (16-bit) variable to write the AO data.

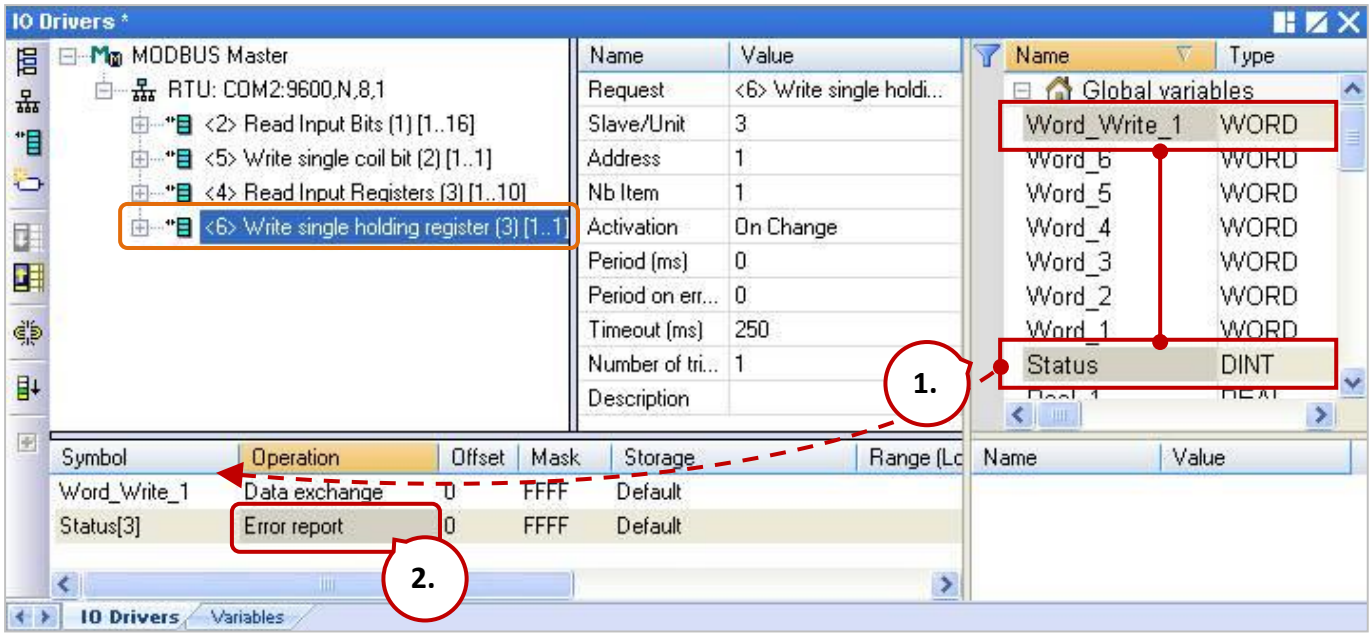
After completing the settings, the results are shown below.



3. **Choose variables.** In the "IO Drivers" window, drag “Word_Write_1” and “Status” variables to the address mapping area of the data block.

   Note: The “Status” is an Array variable, including Status[0] to Status[4]. Only **Status[3]** is required in the address mapping area.

4. **Configure Operation status.**
   Set the **Status[3]** entry to “**Error report**” which means to return an error code when fails to read data and reset to 0 when a successful read.

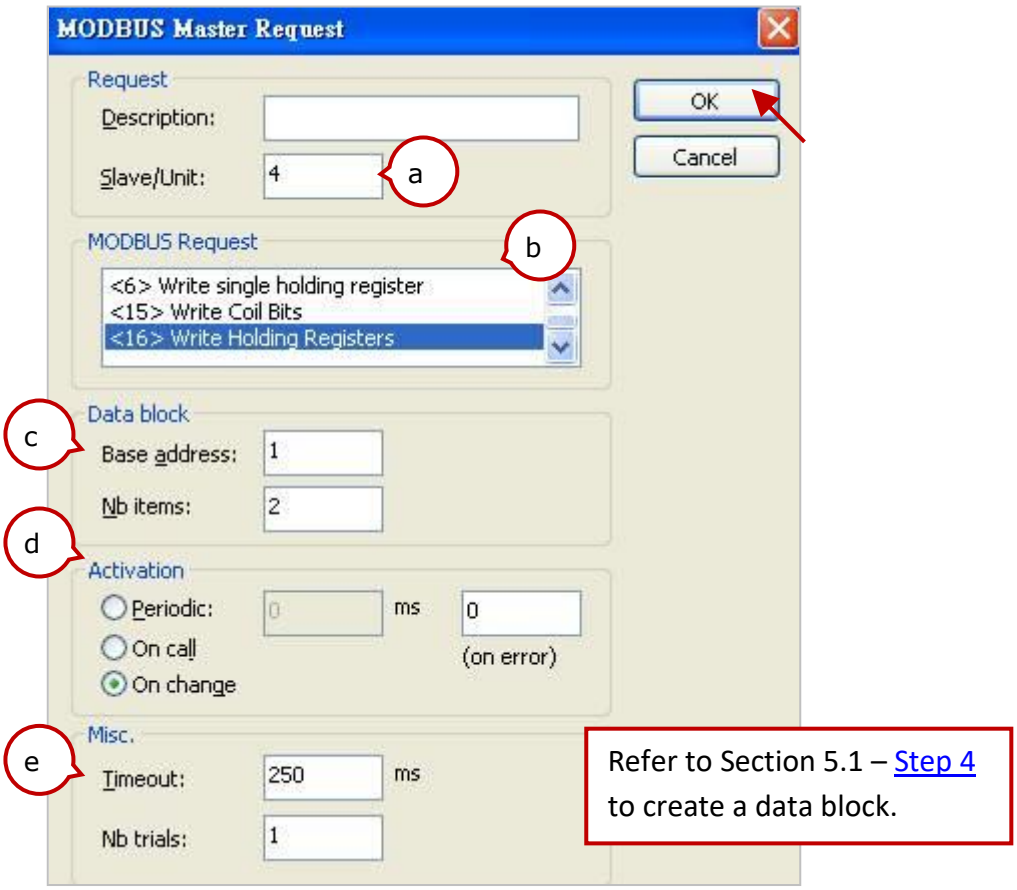


5.1.5 Write AO Data (32-bit)

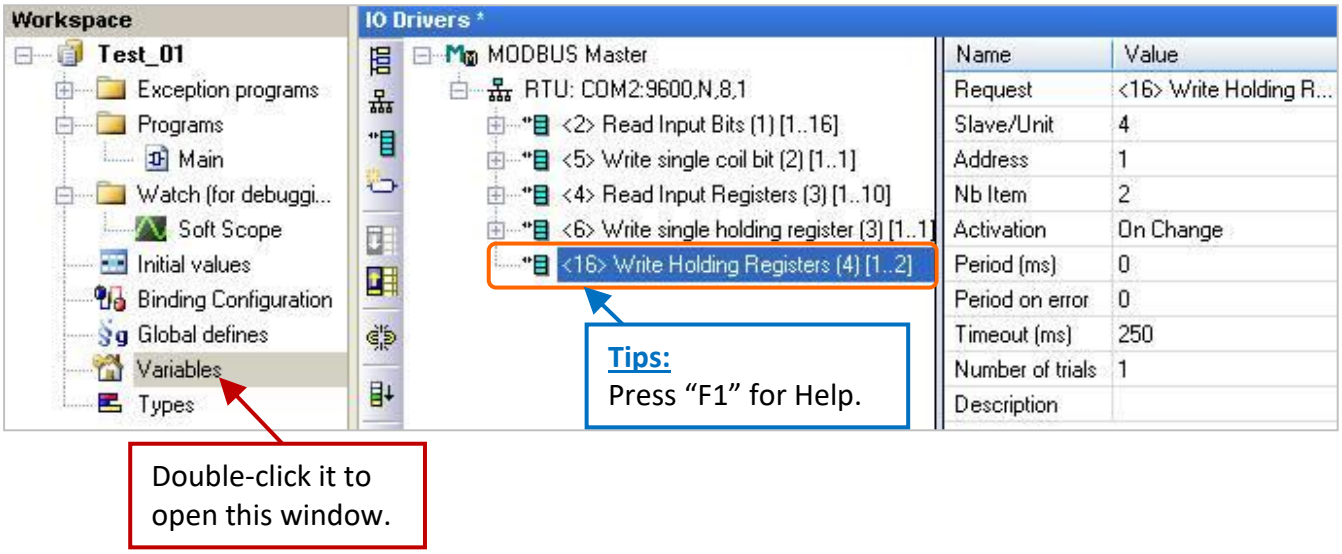
Description:

To write a single AO data to the device (Slave ID = 4) by sending the request when the data is changed. No response for 250 ms can be regarded as abnormal.

1. Completing the settings in the “MODBUS Master Request” window, and then click “OK".

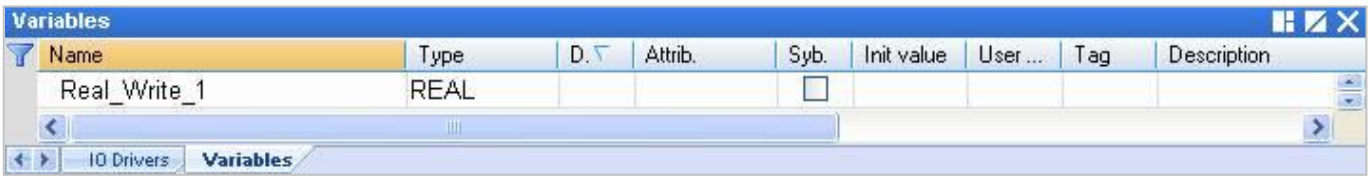


2. Open the “Variables” window and add the needed variables.



Real_Write_1 (REAL): Add one REAL (32-bit) variable to write AO data.

After completing the settings, the results are shown below.

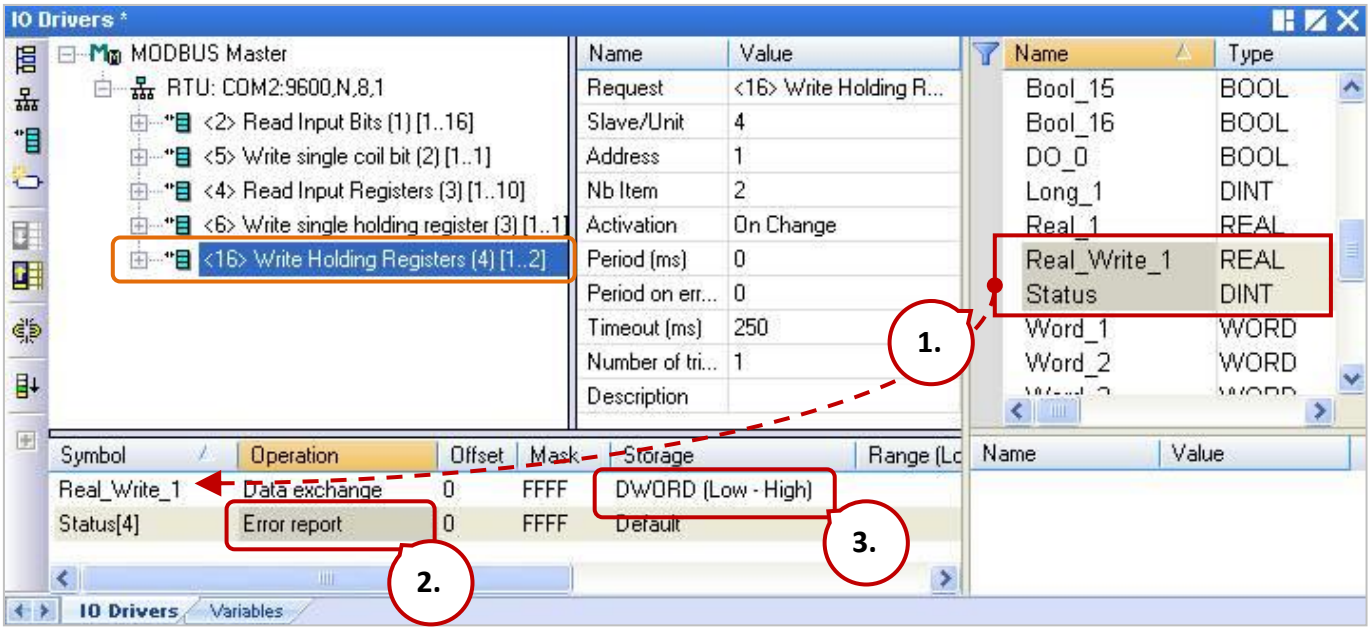


3. **Choose variables.** In the "IO Drivers" window, drag “Word_Write_1” and “Status” variables to the address mapping area of the data block.

   Note: The “Status” is an Array variable, including Status[0] to Status[4]. Only **Status[4]** is required in the address mapping area.

4. **Configure Operation status.**
   Set the Operation of **Status[4]** to “**Error report**” which means to return an error code when fails to read data and reset to 0 when a successful read.

5. **Configure Storage format.**
   Set the Storage of **Real_Write_1** (32-bit) to “**DWORD (Low – High)**”, mapping two Word (16-bit).



5.1.6 To Disable/Enable the Modbus RTU/ASCII Master Port

The Modbus RTU/ASCII Master port that is enabled in the Win-GRAF "Fieldbus Configuration" - "IO Drivers" window will be activated automatically whenever the PAC is powered on.

If the user wants to disable the Modbus **RTU/ASCII** Master port while the program is running, using the **"MBRTU_M_disable"** function as follows.

```
(* Declare To_disable as BOOL *)
If  To_disable  then
  To_disable := FALSE ;
  MBRTU_M_disable(2) ;
End_if;
```

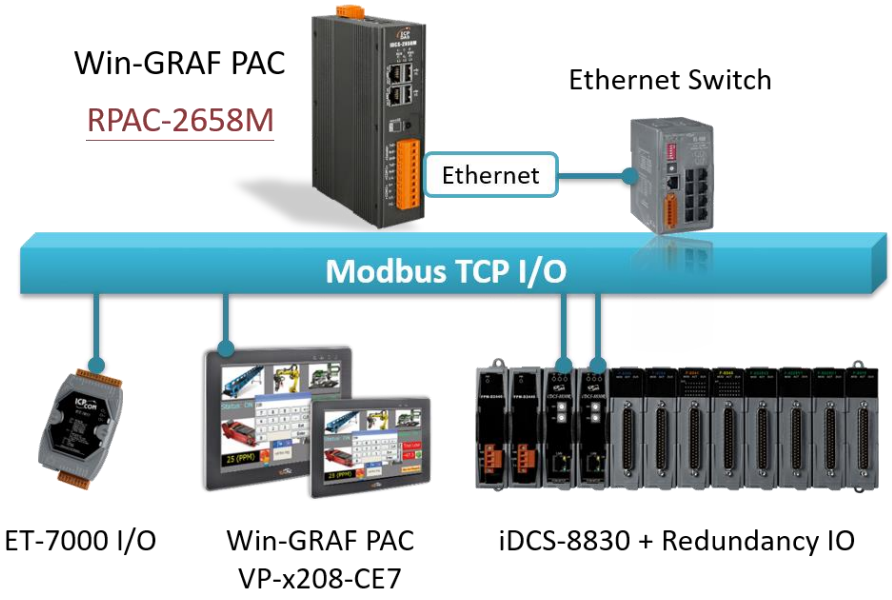
As with the above codes, when "To_disable" is set to "TRUE", the Modbus RTU/ASCII Master port (i.e., COM2) will be disabled. Also, using the **"MBRTU_M_enable"** function to enable it as follows.

```
(* Declare  To_enable as BOOL
    Declare Status_com2 as BOOL *)
If  To_enable  then
  To_enable := FALSE ;
  MBRTU_M_enable(2) ;
End_if;
Status_com2 := MBRTU_M_status(2) ;
```

The **"MBRTU_M_status"** function listed above is used to get the status of the Modbus RTU/ASCII Master port. True: Enabled; False: Disabled.

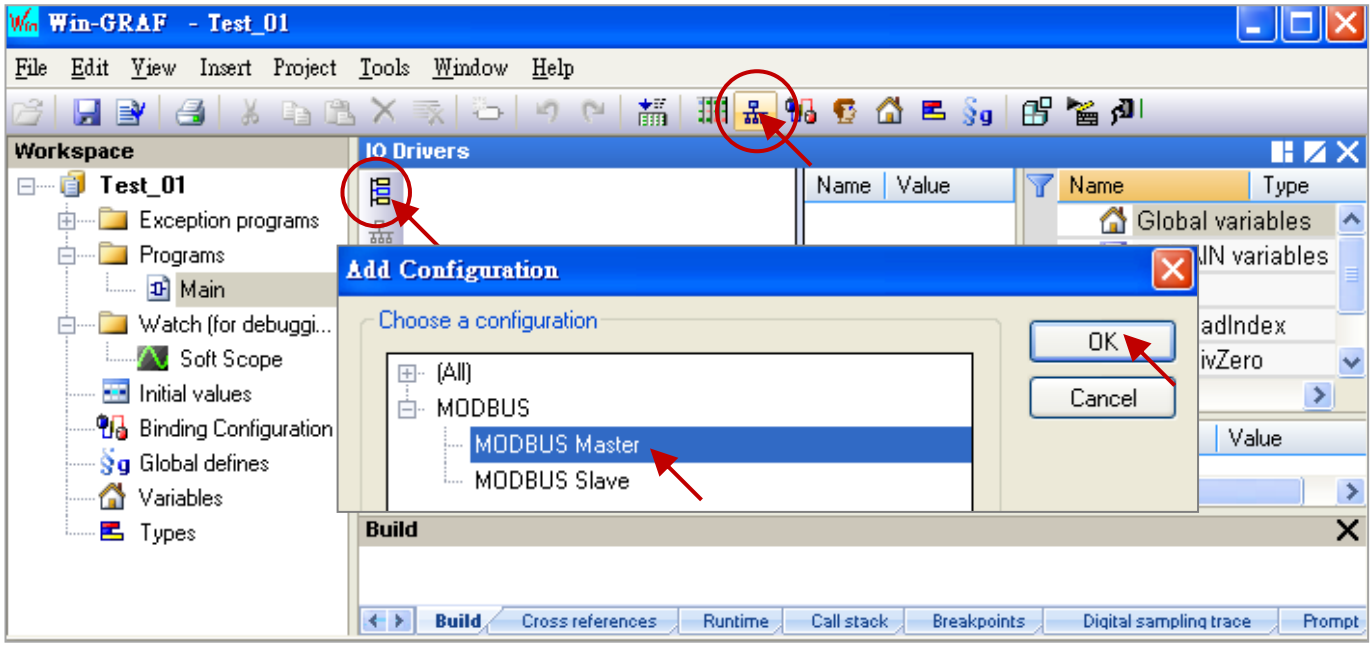
5.2 Enabling the Win-GRAF PAC as a Modbus TCP/UDP Master

Application Diagram:



Follow the steps below:

1. Click the tool icon “Open Fieldbus Configuration” to open the “IO Drivers” window.
2. **Set the PAC to act as a Modbus Master**. Click the “Insert Configuration” button on the left of the “IO Drivers” window and click the “MODBUS Master”, then click “OK”.

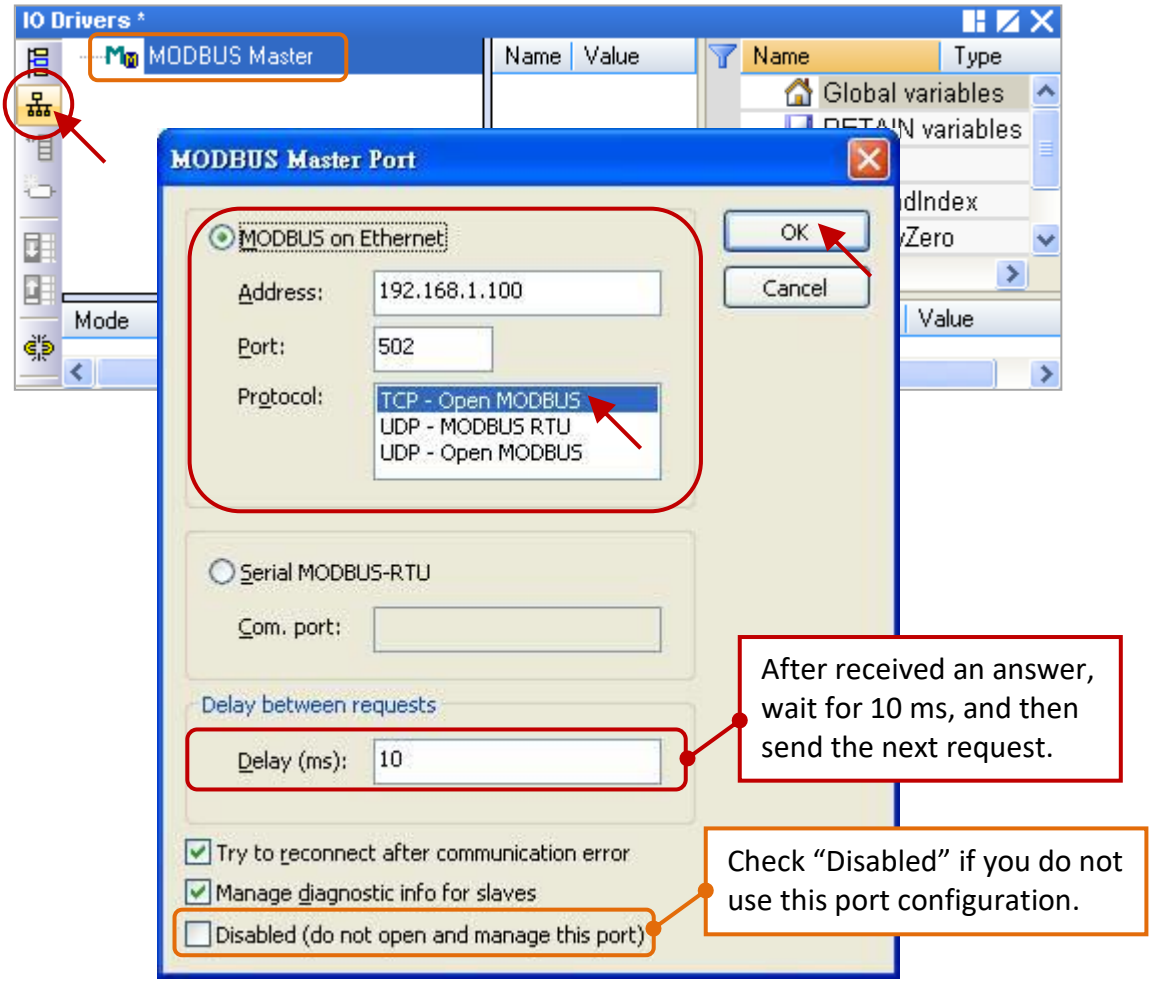


Note:

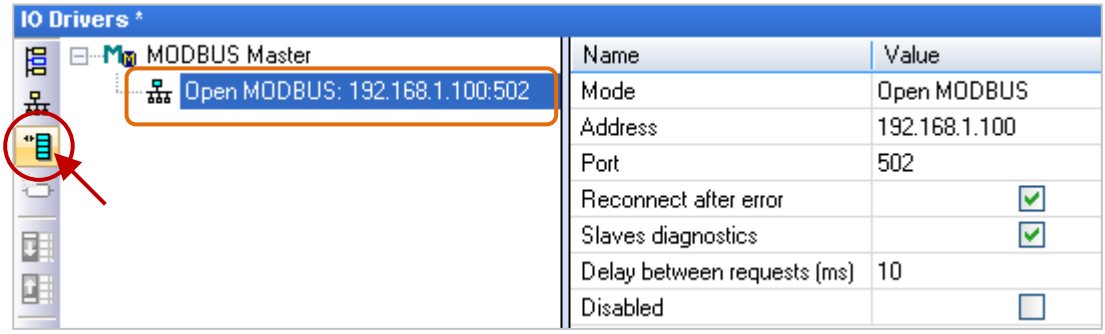
The Modbus Master configuration allows using several Modbus TCP/UDP or Modbus RTU/ASCII ports. If you want to disable the Modbus TCP/UDP port while the program is running, refer to Section 5.2.4.

3. **Choose the Ethernet port.** Click the “Insert Master/Port” button on the left side to open the settings window. Select “MODBUS on Ethernet” and set the Address, Port, Protocol, and the Delay time, then click OK.

Address: Enter the IP Address of the Modbus Slave device (e.g., “192.168.1.100”).
Port: TCP port number of the Modbus Slave device.
Protocol: When using Modbus TCP, select the “TCP – Open MODBUS”.
When using Modbus UDP, select the “UDP – Open MODBUS”.
Delay: Enter the delay time of the request (e.g., 10 ms, can be 0 to 10000).



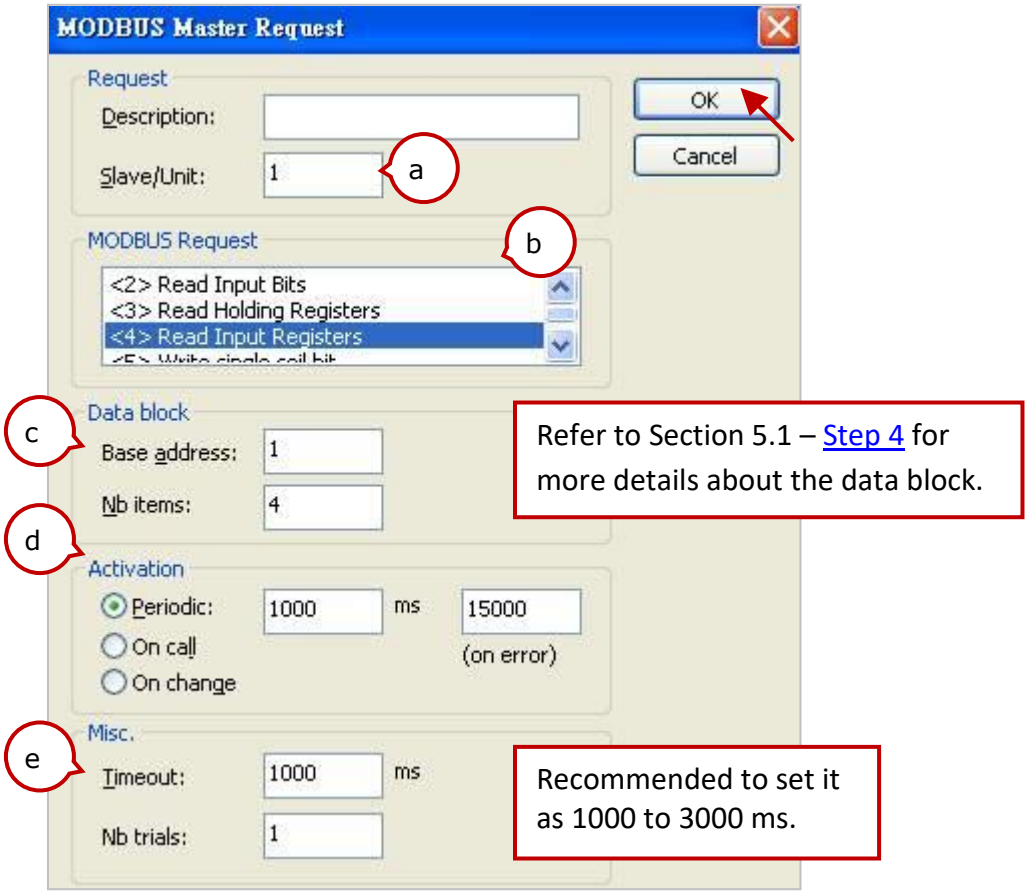
4. **Create a data block.** Click the “Insert Slave/Data Block” button to open the settings window.



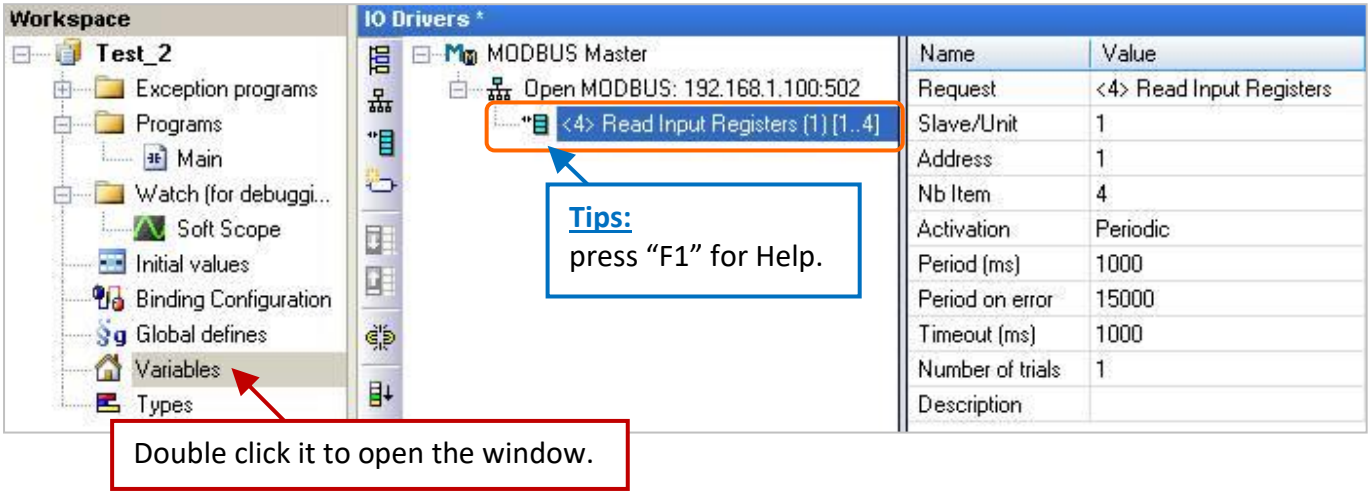
Read AI Data

To read 4 AI data from the device (Slave ID = 1) by sending the request every second. It will be sent after 15 seconds if an exception occurs. No response for 1000 ms can be regarded as abnormal.

1. Completing the settings in the “MODBUS Master Request” window, and then click “OK".

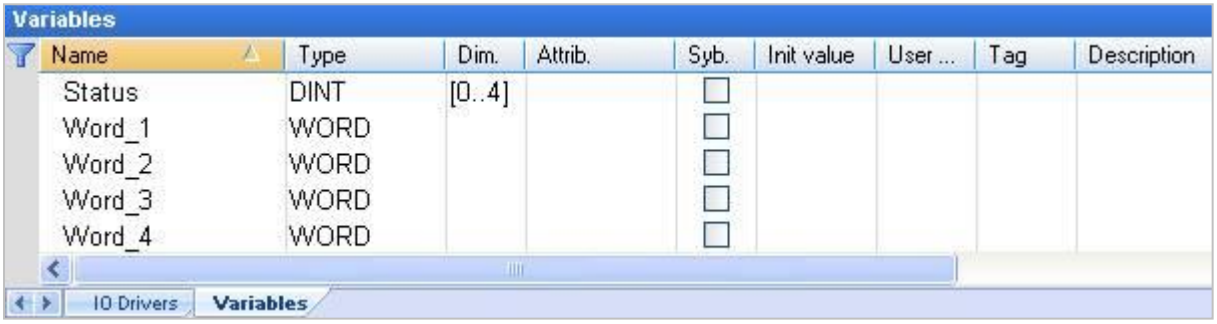


2. Open the “Variables” window and add the needed variables.



“Word_1 to Word_4” (WORD): Add 4 WORD (16-bit) variables to read AI data.
“Status" (DINT, Dim.=5): Add one Array variable to display the status for accessing data.

After completing the settings, the results are shown below.

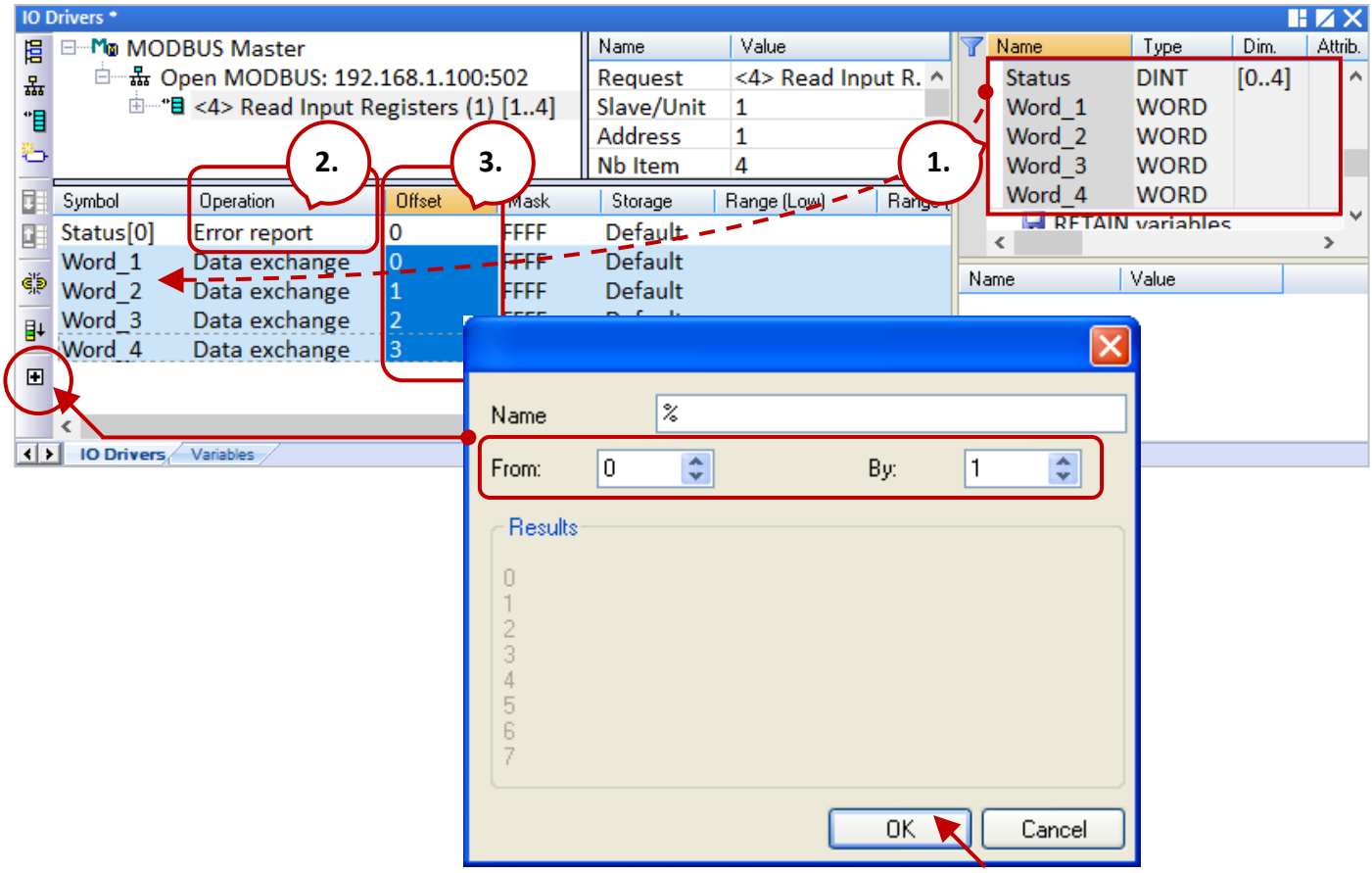


3. **Choose variables.** In the "IO Drivers" window, drag “Word_1 to Word_4” and “Status” variables to the address mapping area of the data block.

   Note: The “Status” is an Array variable, including Status[0] to Status[4]. Only **Status[0]** is required in the address mapping area.

4. **Configure Operation status.**
   Set the Operation of the “Status[0]” to “**Error report**” which means to return an error code when fails to read data and reset to 0 when a successful read.

5. **Configure Offset values.** Select the “Offset” field from **“Word_1”** to **“Word_4”** and click the “Iterate Property” button on the left side to set the “Offset” value (From: “0” ; By: “1”).



Note:

Using a similar way to create the data block to read/write IO data, also refer to Section 5.1.1 to 5.1.5.

5.2.1 Connecting ET-7000 Series I/O Module

The ET-7000 series is a web-based Ethernet I/O module. Win-GRAF PAC can be set as a Modbus TCP Master to connect several ET-7000 modules. It is recommended that one RPAC-2658M can connect up to 100 ET-7000 modules.

For more information about ET-7000 modules, please visit the website:
https://www.icpdas.com/en/product/guide+Remote__I_O__Module__and__Unit+Ethernet__I_O__Modules+ET-7000_ET-7200

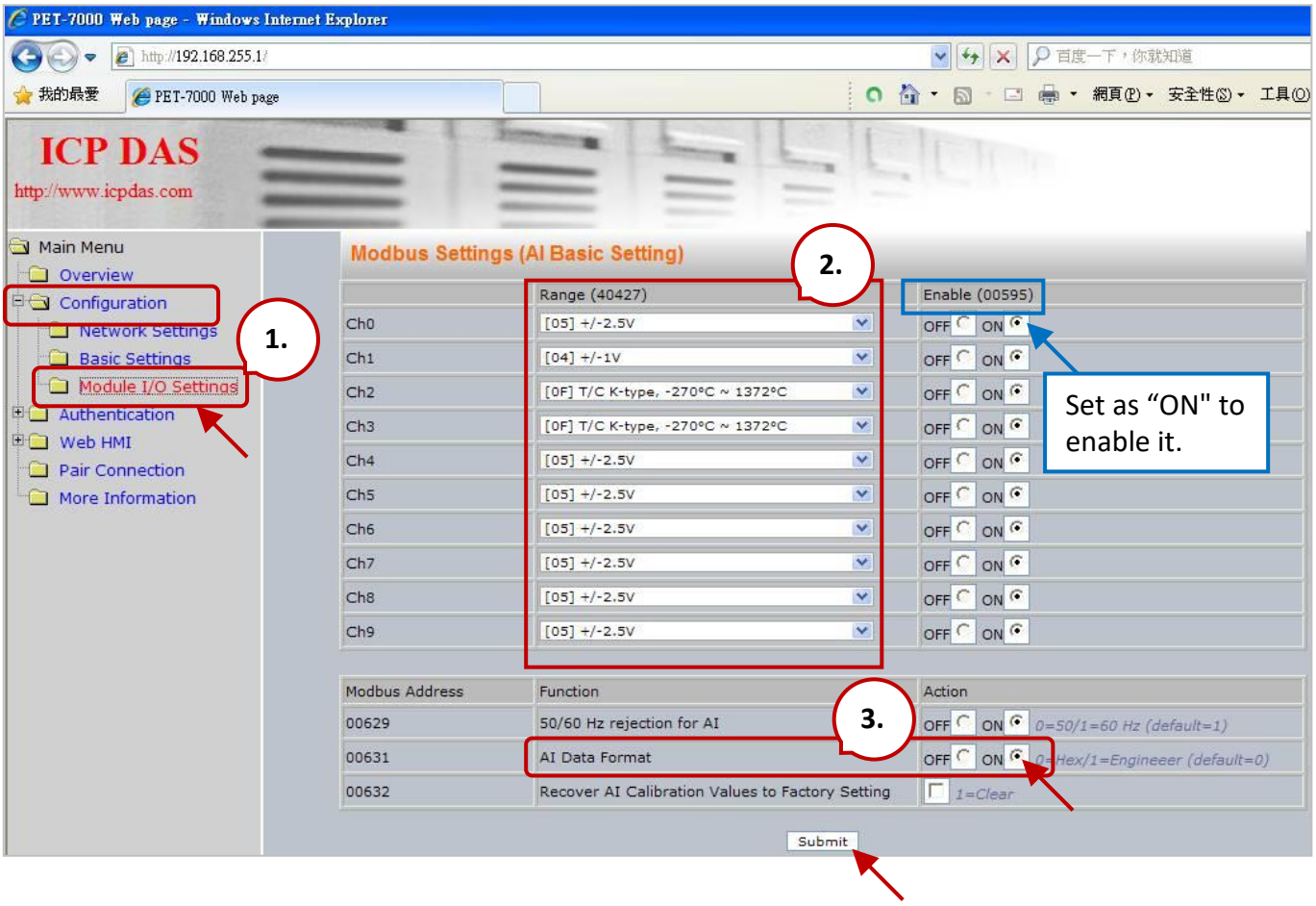
Configure an ET-7000 Module

First, configure the ET-7000 module by using the web browser. The factory settings of ET-7000 are IP address = 192.168.255.1 and Mask = 255.255.0.0. Make sure that both the module and your PC are on the same network segment (e.g., IP address = 192.168.255.100, Mask = 255.255.0.0).

Notice: The Dip Switch on the rear of the ET-7000 must be set as "Normal".

1. Open the browser and enter the IP address of the ET-7000 to connect it.

2. Enter the user name and password to log into ET-7000.



3. Click “Module I/O Settings” under the “Configuration” folder and set the range of channels and then click “Submit”. When using the AO module (e.g., ET-7018Z), set "AI Data Format" to "ON" (Engineering).

   For example,
   The data -10000 to +1000 stands for **-1 to +1 V.**
   The data -25000 to +25000 stands for **-2.5 to +2.5 V.**
   The data 258 stands for **25.8 (℃)**.

Type Code	Range	Data Format	Minimum	Maximum
04	**-1 to +1 V**	**Engineering**	**-10000**	**+10000**
		2’s comp HEX	8000h	7FFFh
05	**-2.5 to +2.5 V**	**Engineering**	**-25000**	**+25000**
		2’s comp HEX	8000h	7FFFh
18	Type M Thermocouple **-200 to 100°C**	**Engineering**	**-20000**	**+10000**
		2’s comp HEX	8000h	4000h

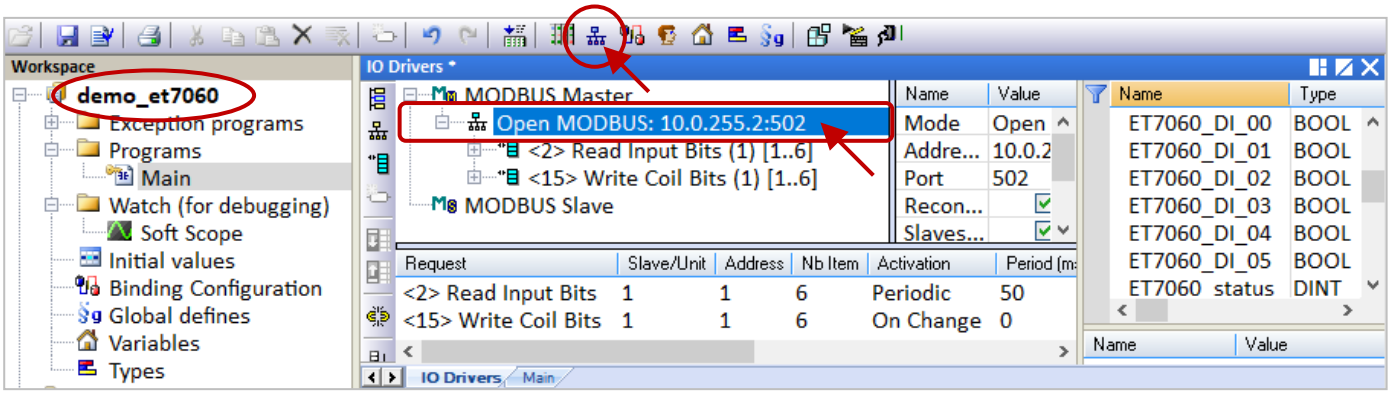
Restore and Modify the Win-GRAF Project:

Win-GRAF demo programs as described below can be downloaded from the website.

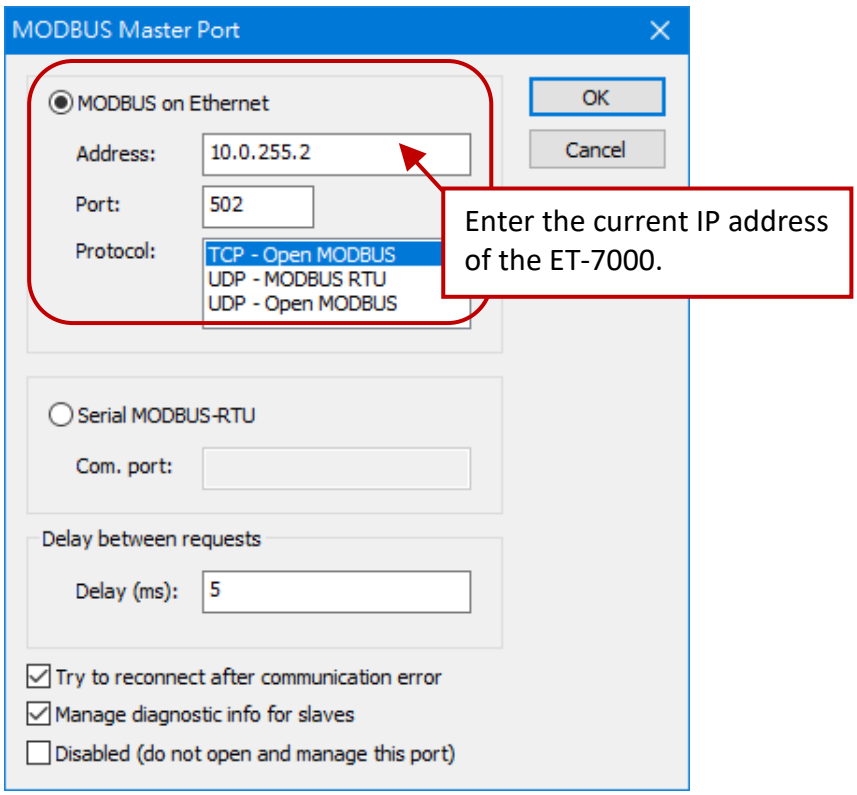
To restore the project to Win-GRAF workbench, click the "File - Add Existing Project - From Zip". (Refer to Section 11.4 if necessary).

Demo Project	File Name	Description
ET-7060	demo_ET7060.zip	Read 6 DIs, write 6 DOs
ET-7018Z	demo_ET7018z.zip	Read 10 AIs

1. Click the “Open Fieldbus Configuration” button to open the “IO Drivers” window.



2. Double-click “Open Modbus: IP:502” to open the “MODBUS Master Port” window. In this case, a Modbus **TCP** Master is enabled to connect to an ET-7000. Enter the IP address of the ET-7000 and the "Port" is fixed "502”, also set the "Protocol" to "TCP - Open Modbus".



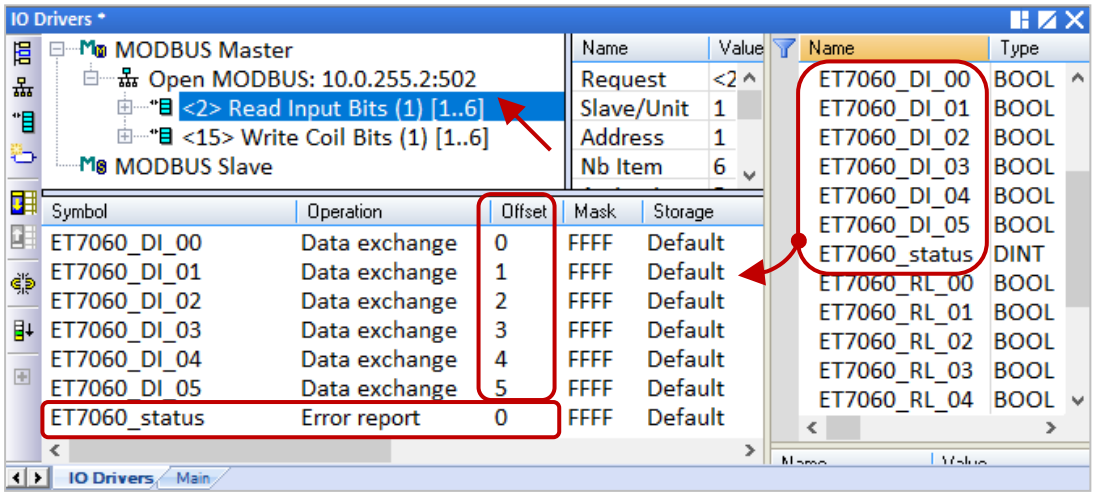
5.2.2 Connecting the ET-7060 (6 DI, 6 Relay Output)

ET-7060 is an Ethernet I/O module with 6 DI and 6 Relay Output channels. The Win-GRAF project describes in this section is "demo_ET7060.zip". Before using the module, make sure you have completed the module and port settings noted in Section 5.2.1.

Description of the Demo:

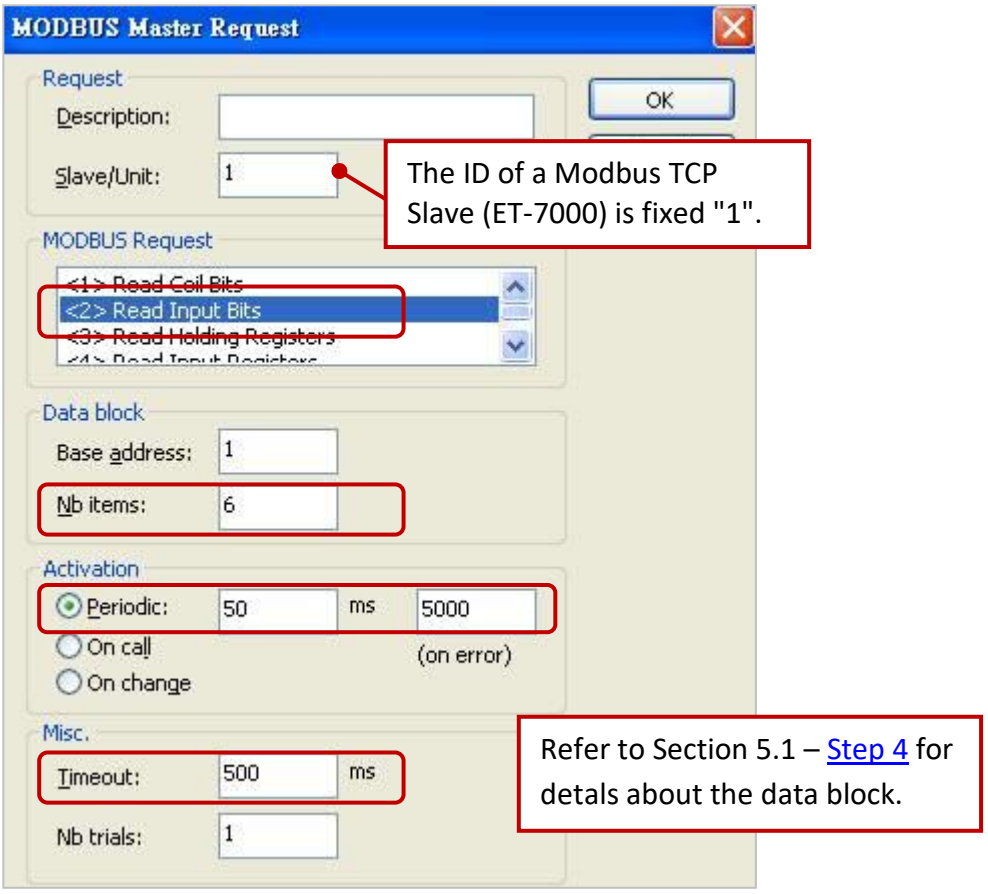
To connect an ET-7060 via LAN3 of the RPAC. There are two data blocks, one is used to read 6 DI data, the other is used to write 6 DO data.

1. Click (or double-click) the first Data Block (i.e., <2> Read Input Bits) to view the settings.

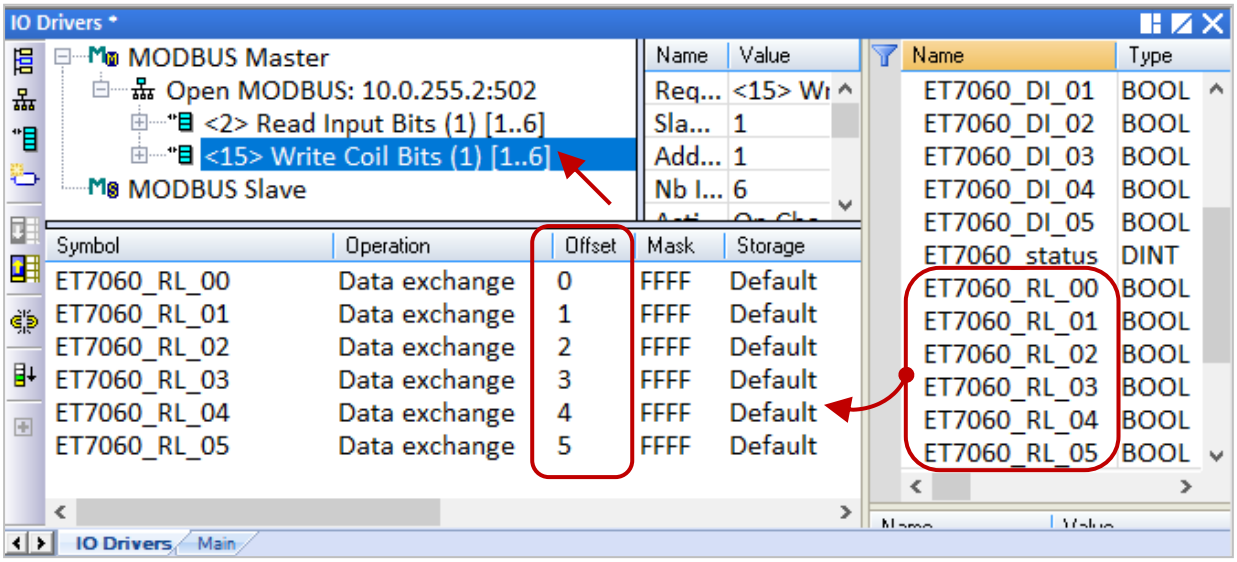


Description:

To read **6** DI data from the device (Slave ID = **1**) by sending the request every **50 ms**. It will be sent after **5** seconds if an exception occurs. No response for **500 ms** can be regarded as abnormal.

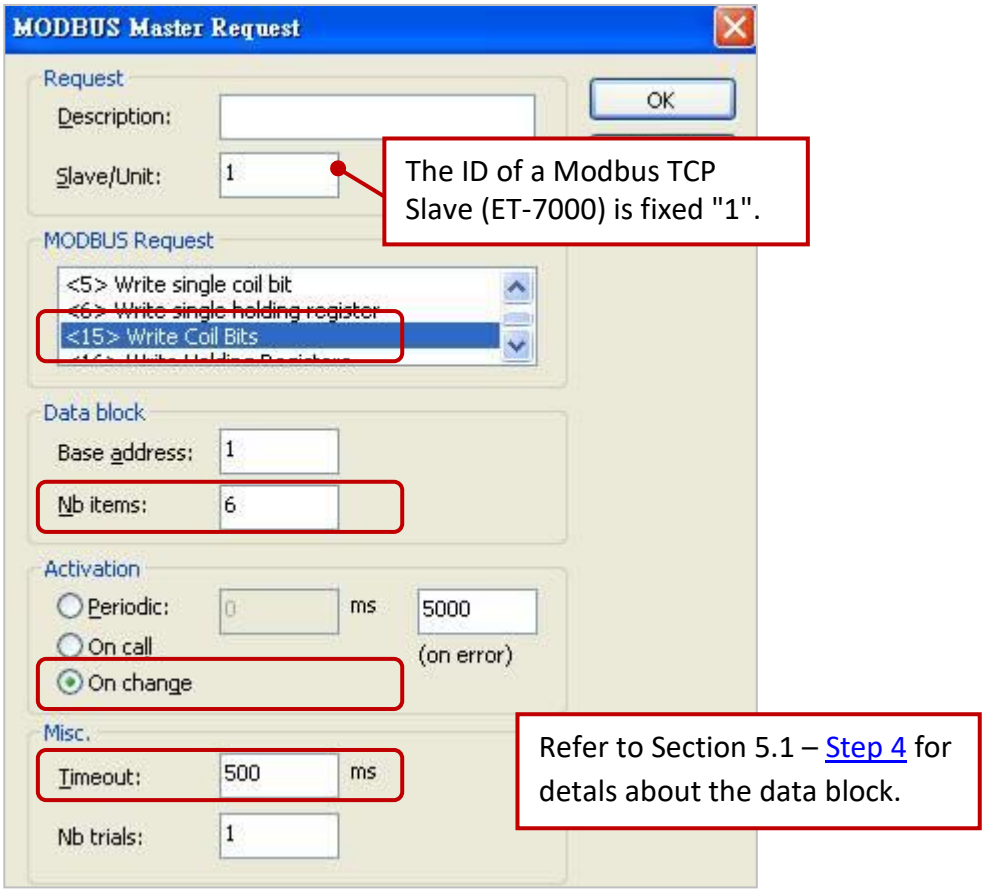


2. Click (or double-click) the second Data Block (<15> Write Coil Bits) to view the settings.



Description:

To write **6** DO data to the device (Slave ID = **1**) by sending the request when the data is changed. No response for **500 ms** can be regarded as abnormal.



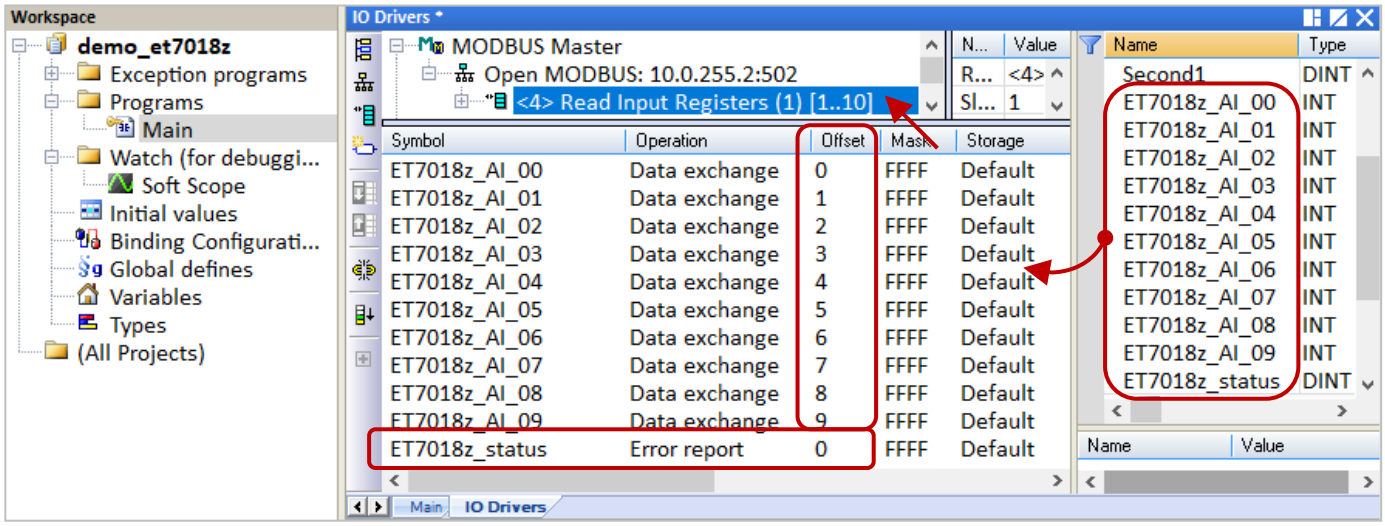
5.2.3 Connecting the ET-7018Z (10 AI)

The ET-7018Z is an Ethernet I/O module with 10 AI channels. The Win-GRAF project describes in this section is "demo_ ET7018z.zip". Before using it, make sure you have completed the module and port settings noted in Section 5.2.1.

Description of the Demo:

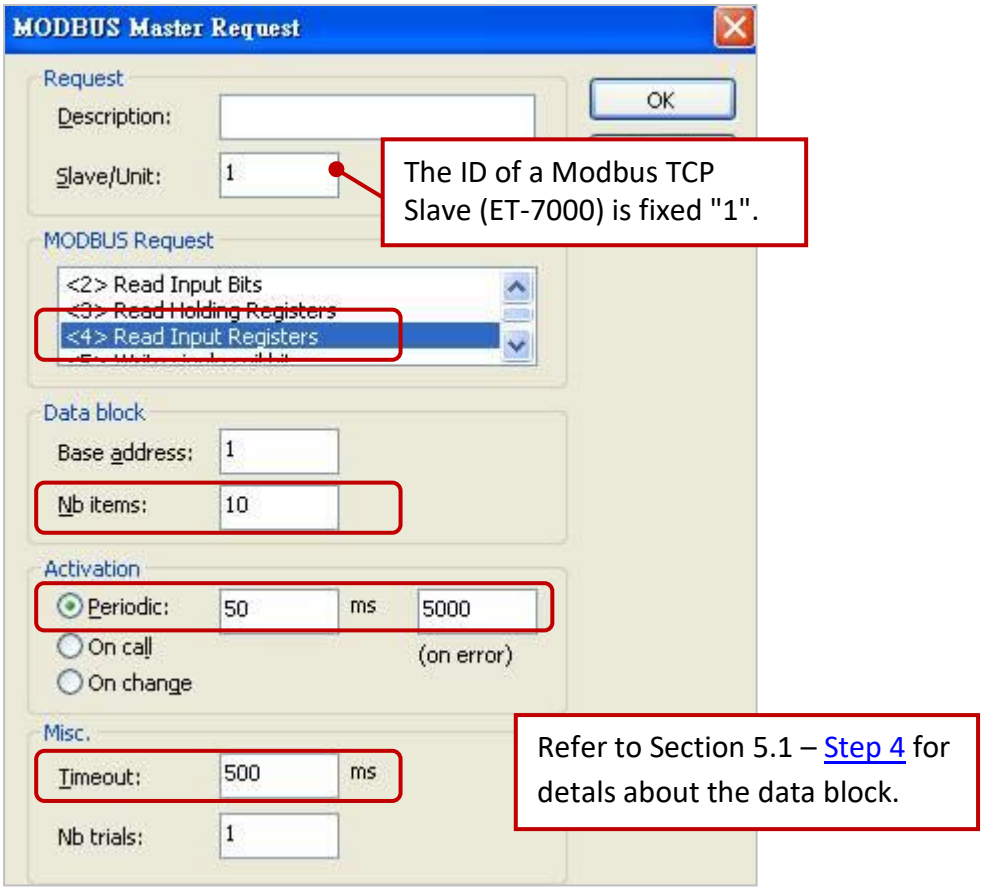
To connect an ET-7018Z via LAN3 of the RPAC. There is one data block used to read 10 AI data.

1. Click (or double-click) the data block (i.e., <4> Read Input Registers) to view the settings.



Description:

To read **6** DI data from the device (Slave ID = **1**) by sending the request every **50 ms**. It will be sent after **5** seconds if an exception occurs. No response for **500 ms** can be regarded as abnormal.



5.2.4 To Disable/Enable the Modbus TCP/UDP Master Port

The Modbus TCP/UDP Master port enabled in the Win-GRAF "Fieldbus Configuration" - "IO Drivers" window will be activated automatically whenever the PAC is powered on.

If you want to disable the Modbus **TCP/UDP** Master port while the program is running, using the **"MBTCP_M_disable"** or **"MBUDP_M_disable"** function as follows.

```
(* Declare To_disable as BOOL *)
If  To_disable  then
  To_disable := FALSE ;
  MBTCP_M_disable ( '10.0.255.2' , 502 ) ;
End_if;
```

When "To_disable" is set to "TRUE", the Modbus TCP Master port that connects to a TCP Slave (IP address = "10.0.255.2" and Port_No = 502) will be disabled. If it is necessary, using the **"MBTCP_M_enable"** (or **"MBUDP_M_enable**) function to enable it as follows.

```
(* Declare  To_enable as BOOL
    Status_tcp as BOOL *)
If  To_enable  then
  To_enable := FALSE ;
  MBTCP_M_enable ( '10.0.255.2' , 502 ) ;
End_if;
Status_tcp := MBTCP_M_status ( '10.0.255.2' , 502 ) ;
```

The **"MBTCP_M_status"** or **"MBUDP_M_status"** function can be used to get the status of the Modbus TCP/UDP Master port. (True: enabled ； False: or disabled)

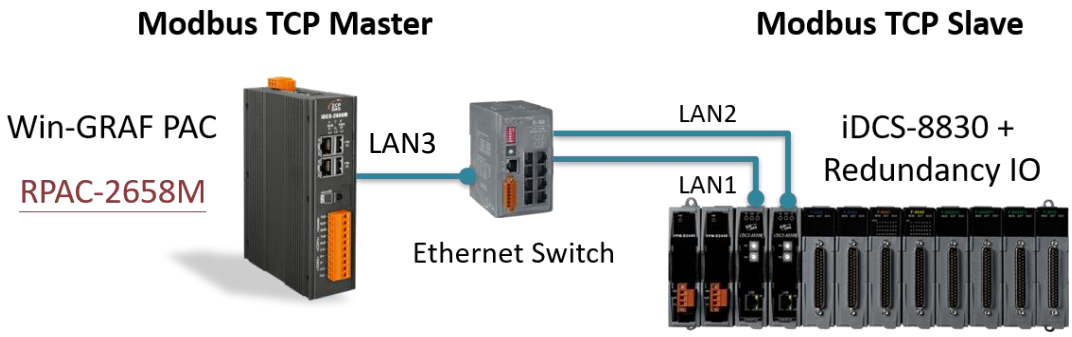
For easy maintenance and error avoidance, declare a STRING variable (e.g., "IP_addr2", the Length is set to "20") and set its initial value (e.g., "10.0.255.2“). As follows:

```
If  To_disable  then
   To_disable := FALSE ;
   MBTCP_M_disable ( IP_addr2 , 502 ) ;
End_if;
Status_tcp2 := MBTCP_M_status ( IP_addr2 , 502 ) ;
```

5.3 Connecting a Modbus TCP Slave Device with two IP addresses

This section describes how to create the redundant "Modbus Master Request", i.e., when one IP address of a Modbus TCP Slave is unavailable, the other IP address still can be used to read/write data.

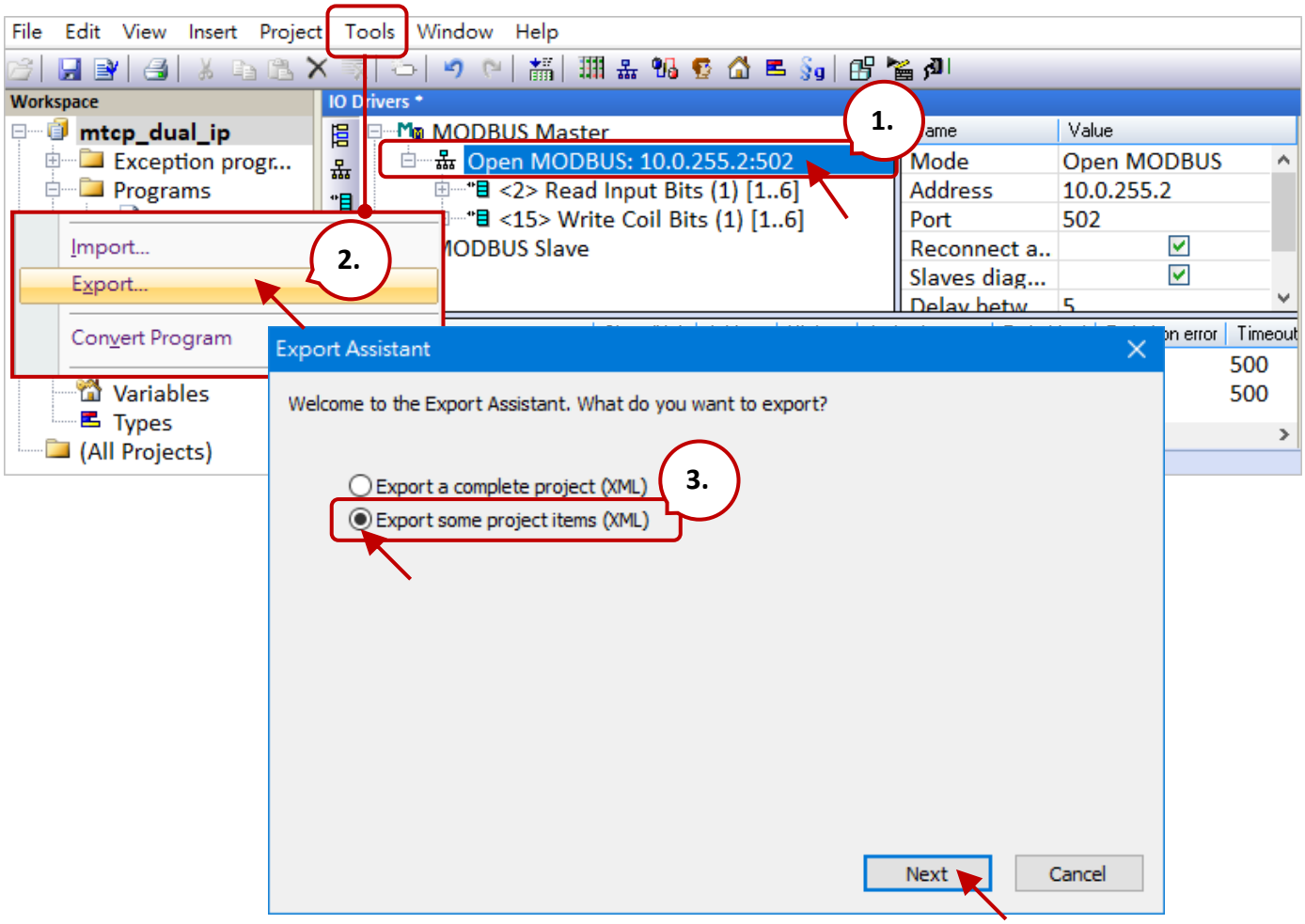
Application Diagram:



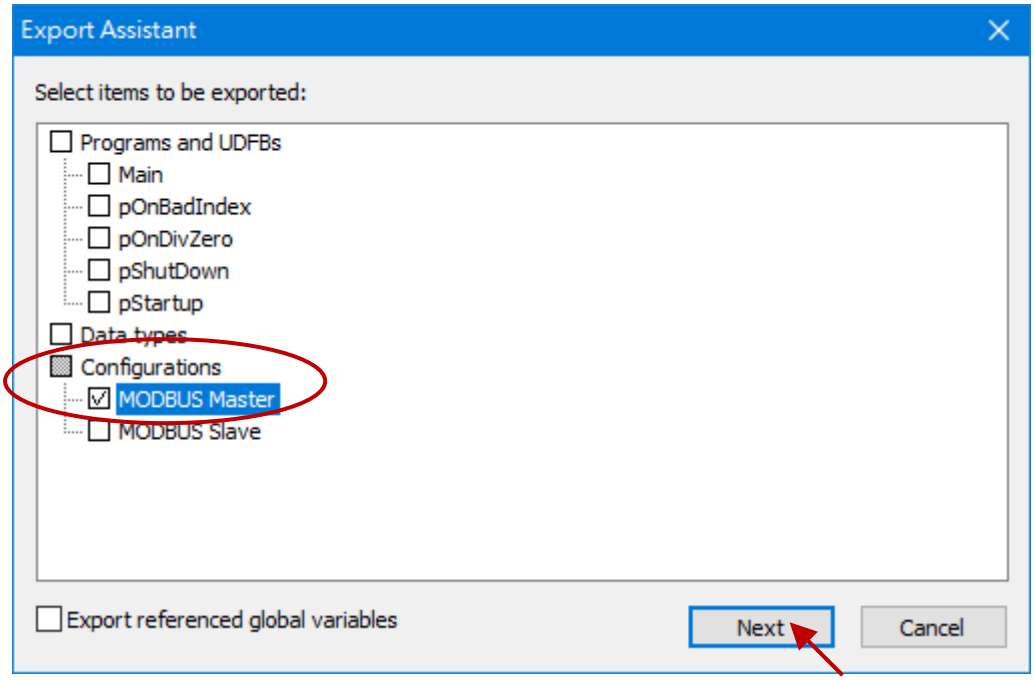
Follow the steps below:

Only one IP address of the Modbus Slave can be specified in the Modbus Master setting. The following describes how to make a copy of the read/write settings and then access data via the second Slave IP address.

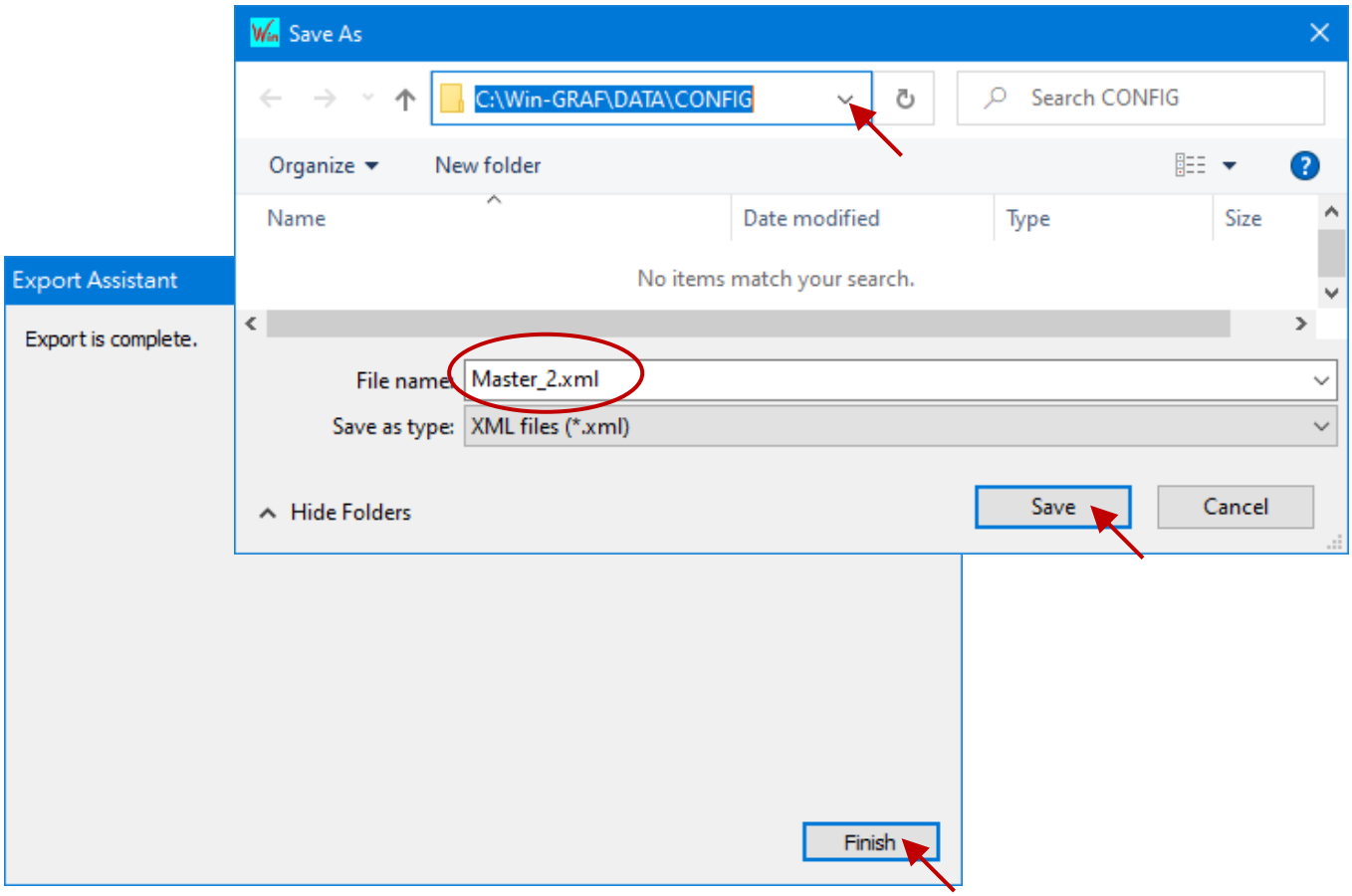
1. Click on "**Open MODBUS:**" in the IO Drivers window and click the menu command **Tools** - **Export**.
2. In the "Export Assistant" window, select "Export some project items (XML)" and click "Next".



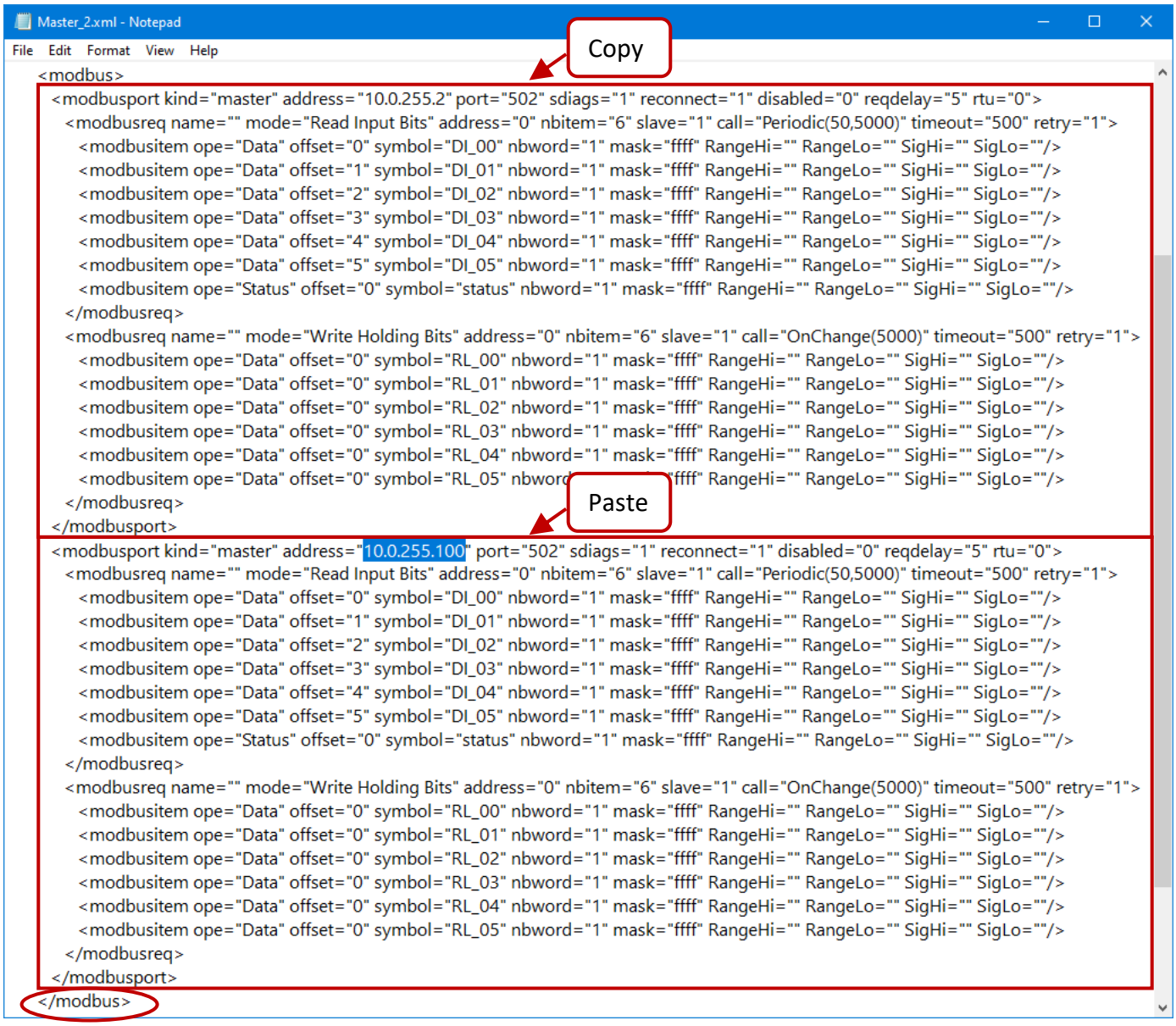
3. Check only the “MODBUS Master” configuration and click the “Next” button.



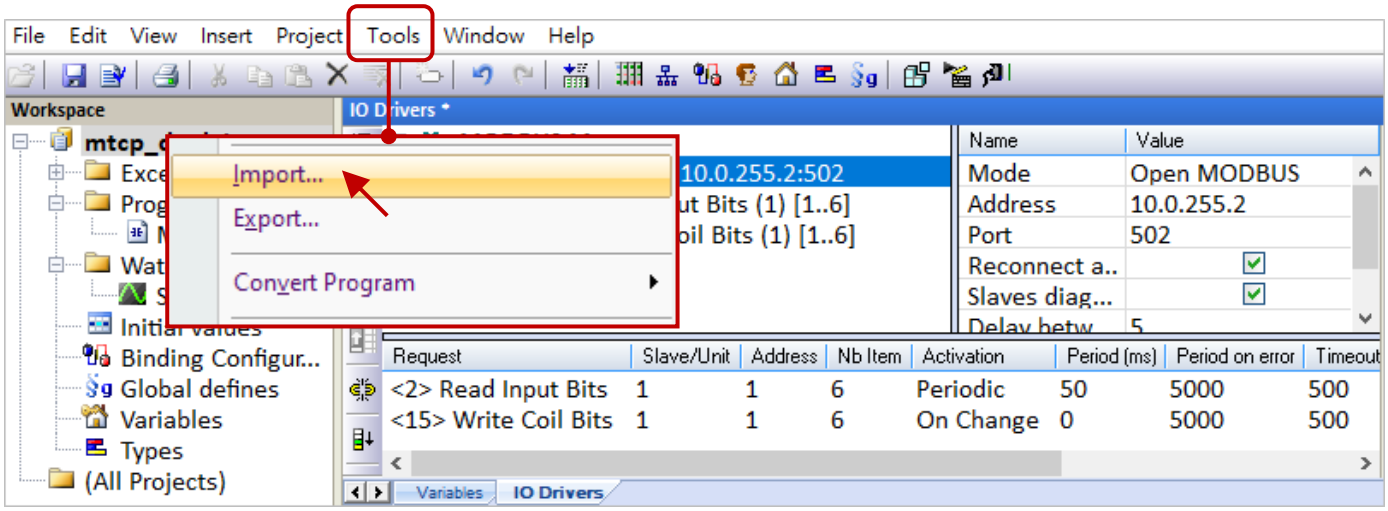
4. Specify the file location (default in C:\Win-GRAF\DATA\CONFIG) and enter a file name (e.g., Master_2.xml), then click the "Save" button. Finally, click "Finish" to export the file.



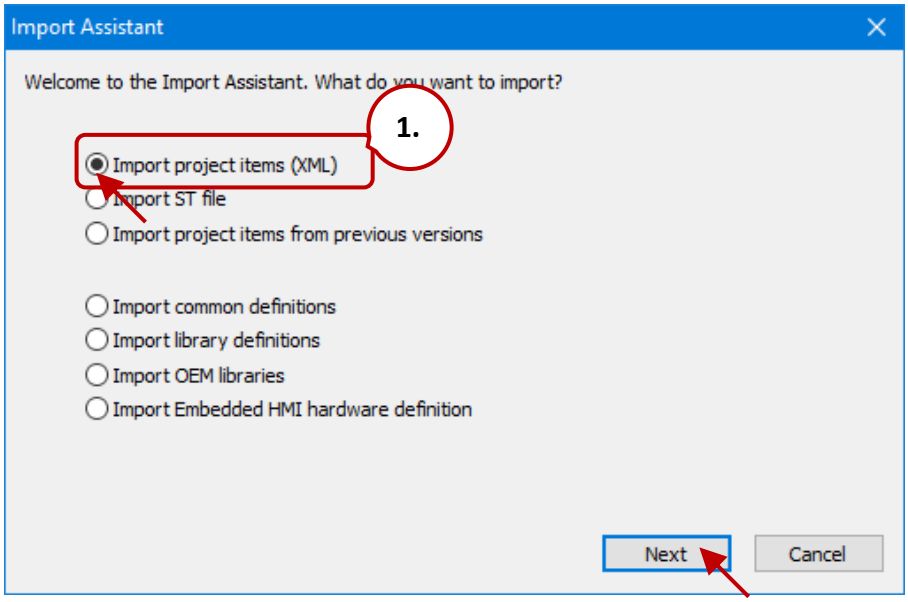
5. Open the ".xml" file in Notepad and copy the code from <modbusport> to </modbusport> tags. Then, paste the code above the </modbus> tag and change the address to the second IP address of the Modbus Slave device (e.g., "10.0.255.2"), then save and close the file.



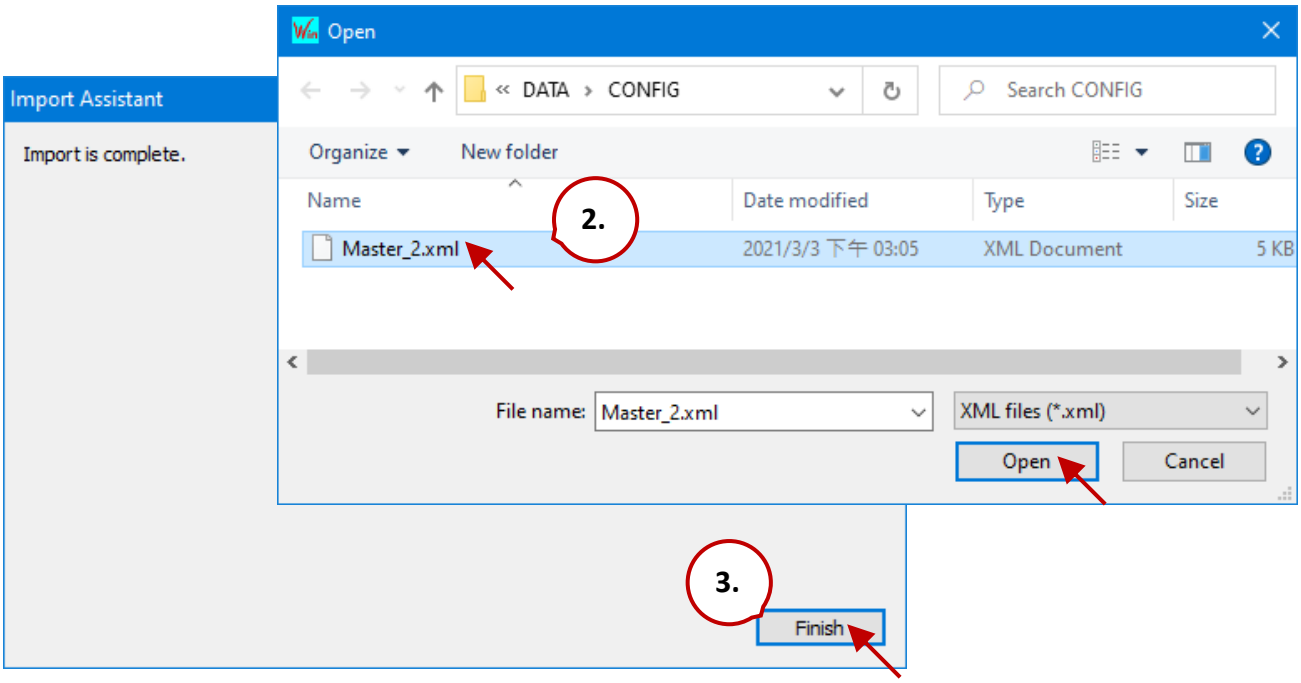
6. Click the menu command **Tools - Import i**n the Win-GRAF Workbench.



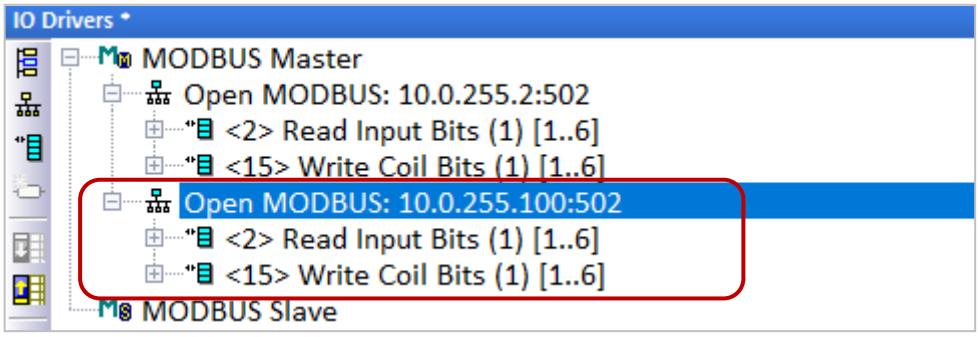
7. Click “Import project items (XML)” and the “Next” button in the Import Assistant window.



8. Select the file you want to import (e.g., “Master_2.xml”) and click the “Open” button, and then click “Finish” to import the file.



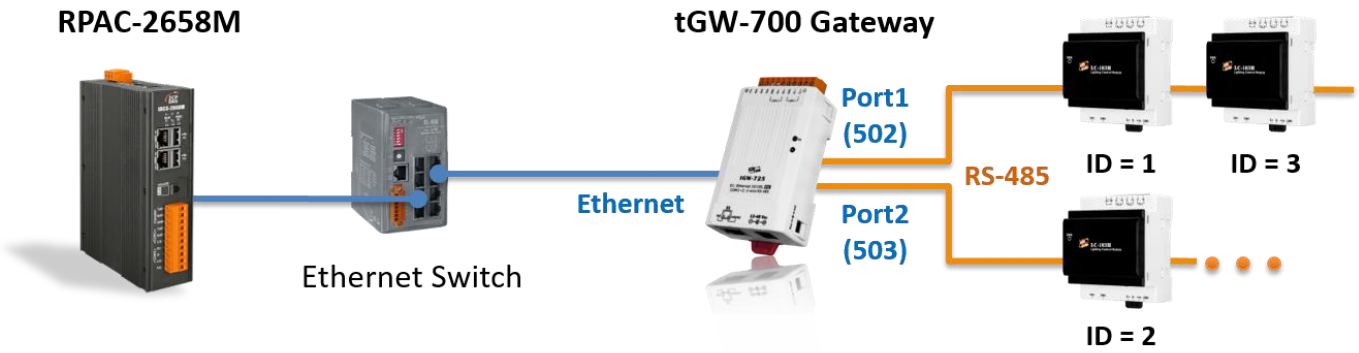
9. By now, it allows accessing data from a Modbus Slave with its second IP address.



5.4 Accessing multiple Modbus RTU Slave via the tGW-700 Gateway

When using Modbus RTU (RS-232/485/422) devices for long-distance transmission, a lower baud rate is set, which allows for stable and longer transmission distance but slower transmission speed. In this case, you can use the tGW-700 series (Modbus TCP to RTU/ASCII gateway) to send data over Ethernet to improve communication efficiency.

This chapter describes how a Win-GRAF PAC communicates with several LC-103 modules via the tGW-700 gateway (see figure below).



The demo program (**demo_tgw725.zip**) can be download on the website:

5.4.1 Configuring the tGW-700 (Modbus TCP to Modbus RTU/ASCII Gateway)

The tGW-700 is a Modbus TCP to RTU/ASCII gateway that enables a Modbus/TCP host to communicate with serial Modbus RTU/ASCII devices through an Ethernet network and eliminates the cable length limitation of legacy serial communication devices.

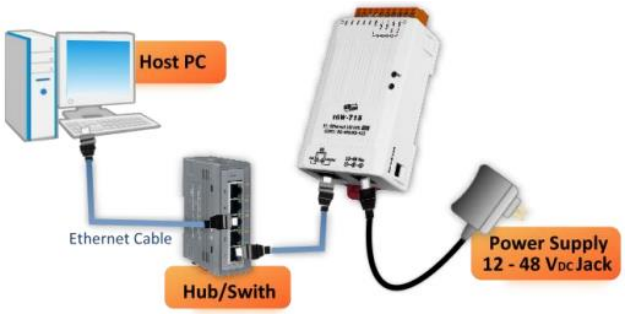
Visit the tGW-700 series webpage for more information about the product, and refer to Chapter 3 & Chapter 4 of the tGW-700 series User Manual for instructions on tGW-700's network setting, testing methods, and web configuration.

Before using the tGW-700, configuring the network and COM Port setting:

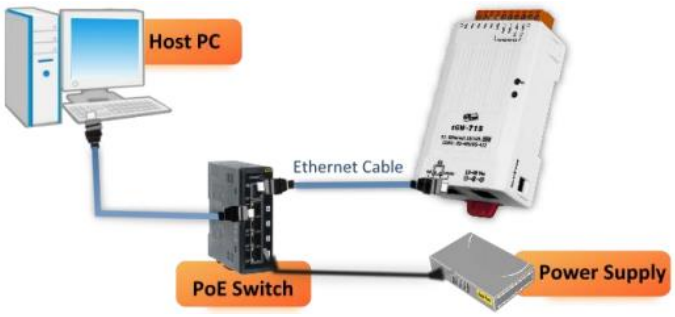
- **Connect the Power Supply and the Host PC**

1. Check that the Init/Run switch is in the “Run” position.

2. Connect both the tGW-700 and the PC on the same sub-network, and then power on the tGW-700.



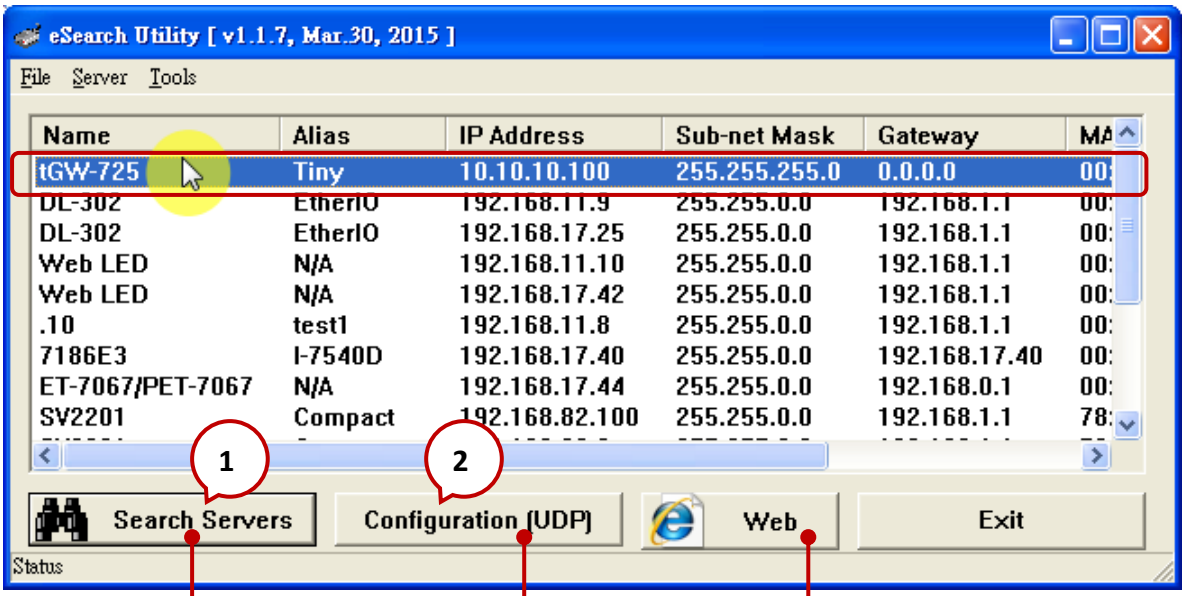
+12 to +48 V_{DC} jack Power Supply (Non-PoE)



PoE Power Supply

- **Install "eSearch Utility" on PC and then Search and Configure the Network Setting for the tGW-700.**

  https://www.icpdas.com/en/product/guide+Software+Utility_Driver+eSearch__Utility

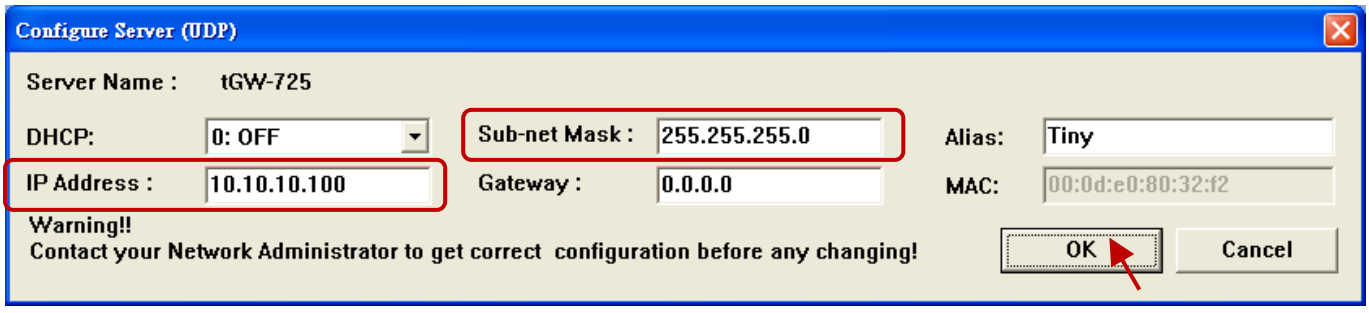


Search the tGW-700.

Set the IP address, Mask, and Gateway.

Open the tGW-700 Web Server.

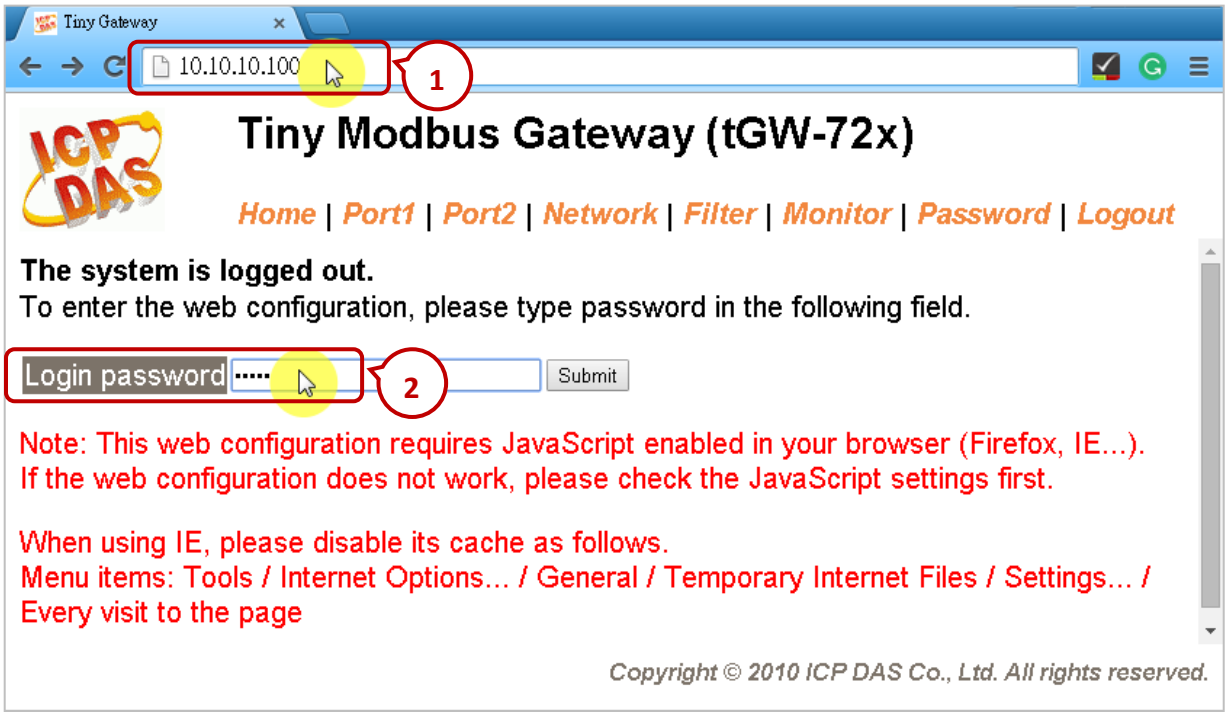
The user can modify the IP Address, Subnet Mask, and Gateway settings. After that, click the "OK" button. New settings will take effect within 2 seconds.



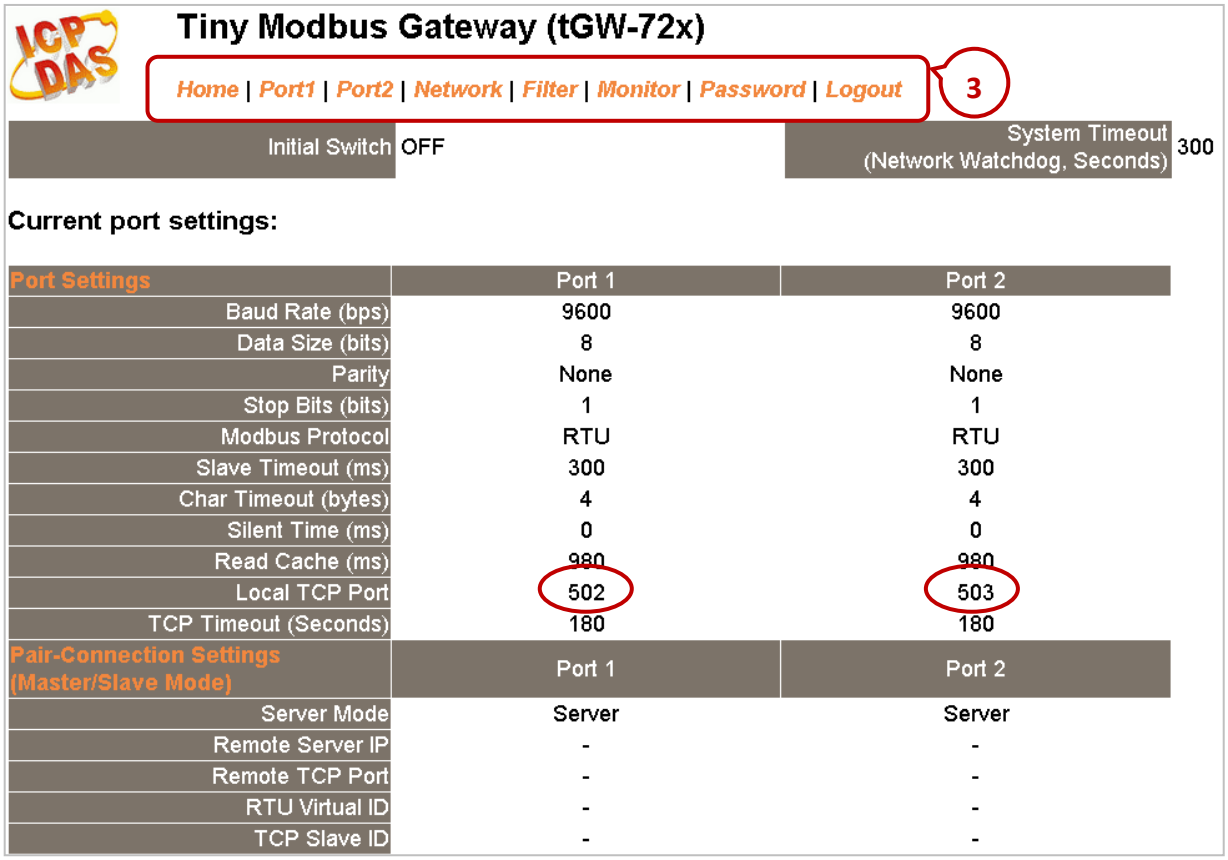
- **Web Configuration**

The following describes the COM Port setting:

1. Enter the IP address of tGW-700 into the address bar. Note that both the tGW-700 and a PC must on the same sub-network.
2. Enter the password (the factory setting is “admin”) and click Submit to log in.



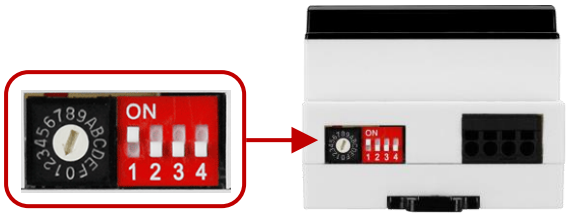
3. After that, the current port setting will be displayed on the Home page. Also, change the settings on the “Port1” or “Port2” page.



5.4.2 Accessing the LC series Module via the tGW-700 Gateway

The following describes how Win-GRAF PAC access data from LC-103 modules via the tGW-725 gateway.

tGW-725 is a tiny Modbus TCP to RTU/ASCII gateway with PoE and 2-port RS-485.
LC-103H is a lighting control module that supports the Modbus RTU protocol and provides 1 channel for digital input and 3 channels for relay output. Before using it, set the ID number of the module, e.g., adjust the rotary switch to “1" at the bottom of the module to set the ID as "1".

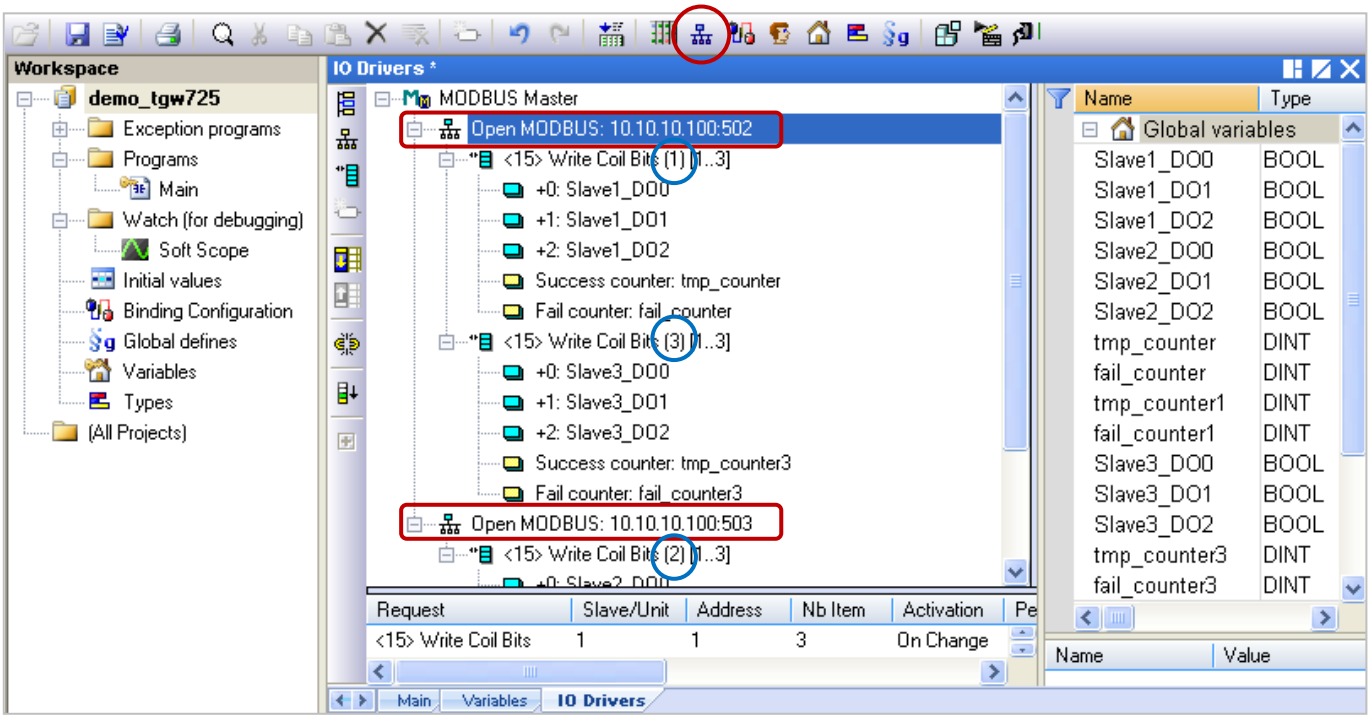


The LC series module user manual:
http://www.icpdas.com/en/download/show.php?num=596&model=LC-103H/

Download the demo program (**demo_tgw725.zip**) on the website. Click the menu commands **File - Add Existing Project - From Zip...** in the Workbench to open the project.

Description of the Demo: (Refer Section 5.2 to set a Modbus TCP Master)

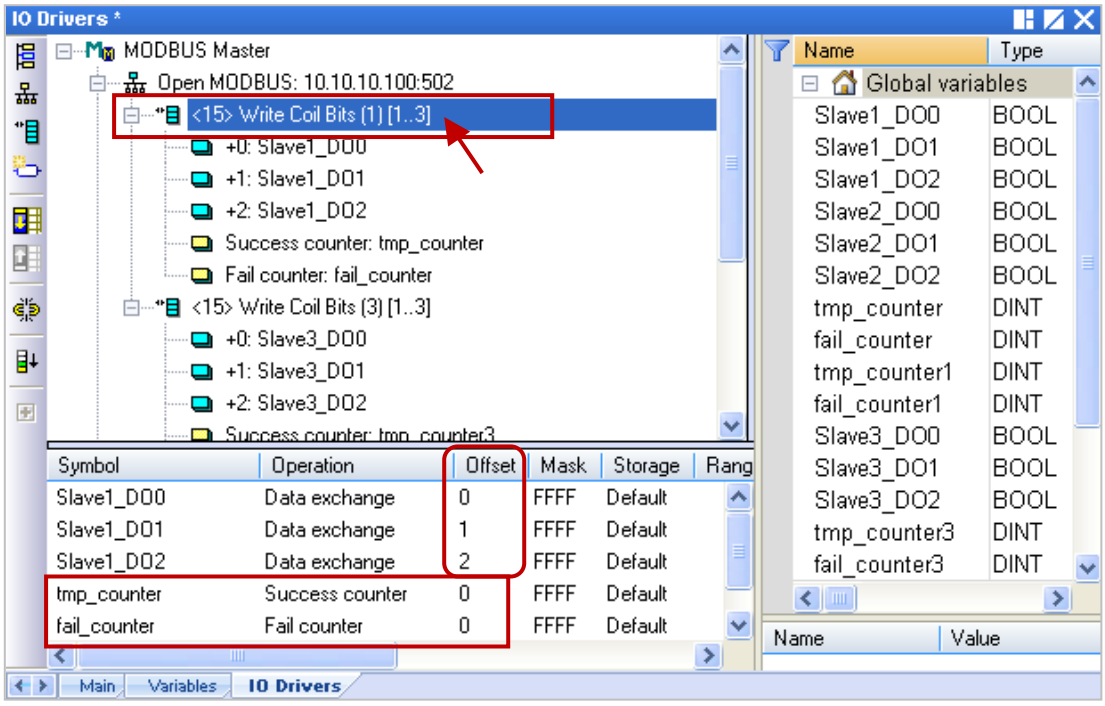
1. Click the “Open Fieldbus Configuration” tool button to open the “IO Drivers” window.



The table lists the information about each piece of equipment:

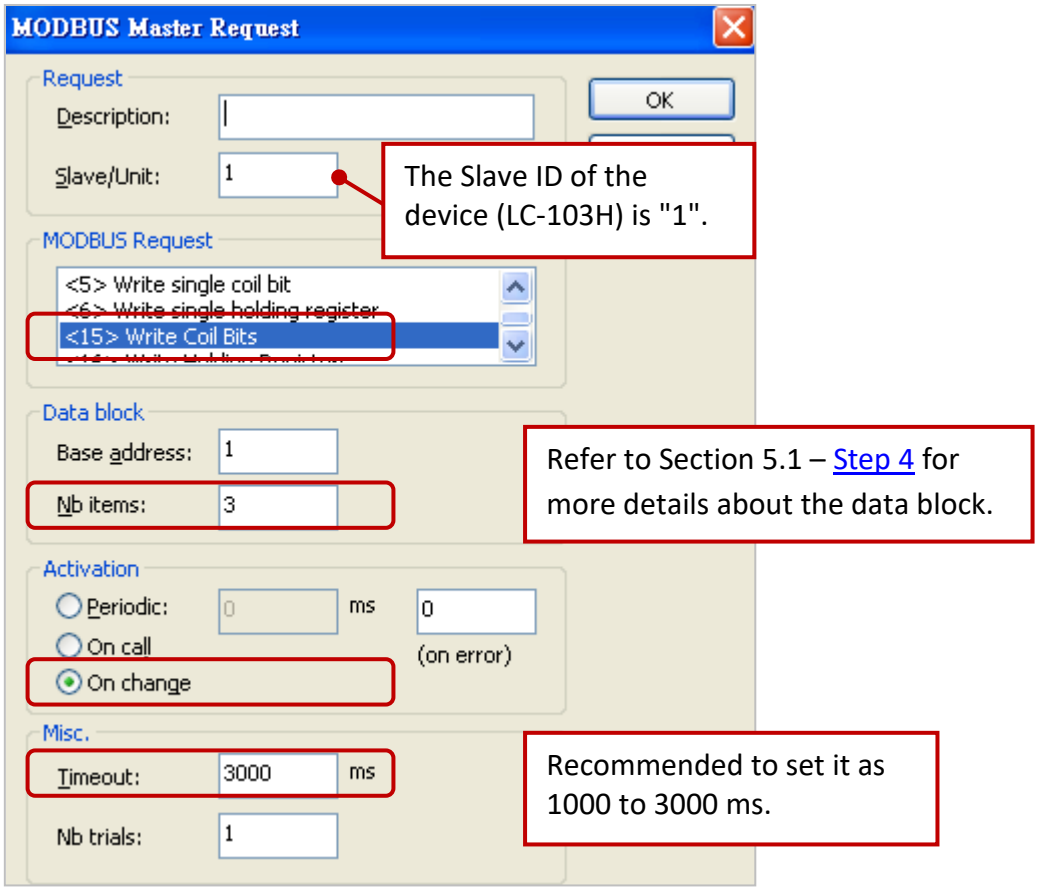
Win-GRAF PAC	tGW-725			LC-103H	
Modbus TCP Master	Modbus TCP to RTU/ASCII Gateway			Modbus RTU Slave	
	IP Address 10.10.10.100	TCP Port = 502	COM1	Two modules	Slave ID = 1 &3
		TCP Port = 503	COM2	One module	Slave ID = 2

2. Click (or double-click) the first data block under port 502 to view the settings.
   The "Operation" is set to "Success counter" (or "Fail counter"), which means the value of the variable will be added 1 if succeed (or failed) to send the command. Also, the “Offset” value of these items must be set as “0”.

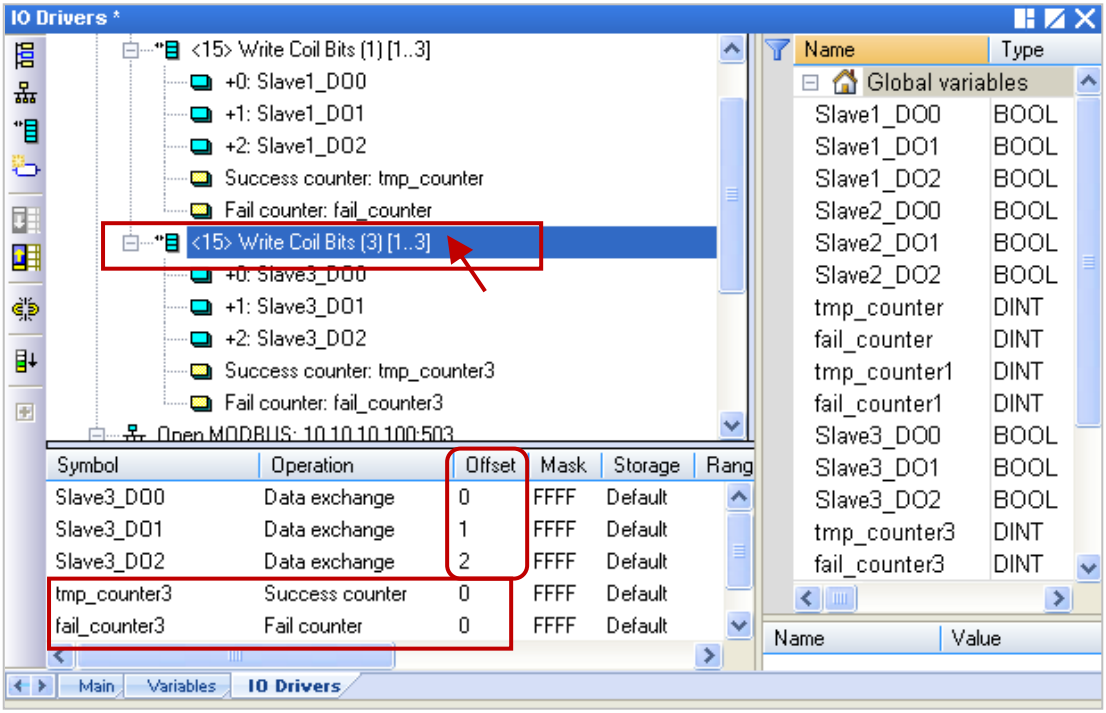


Description:

To write **three** DO data to the device (Slave ID = **1**) by sending the request when the program calls. No response for **3 seconds** can be regarded as abnormal.

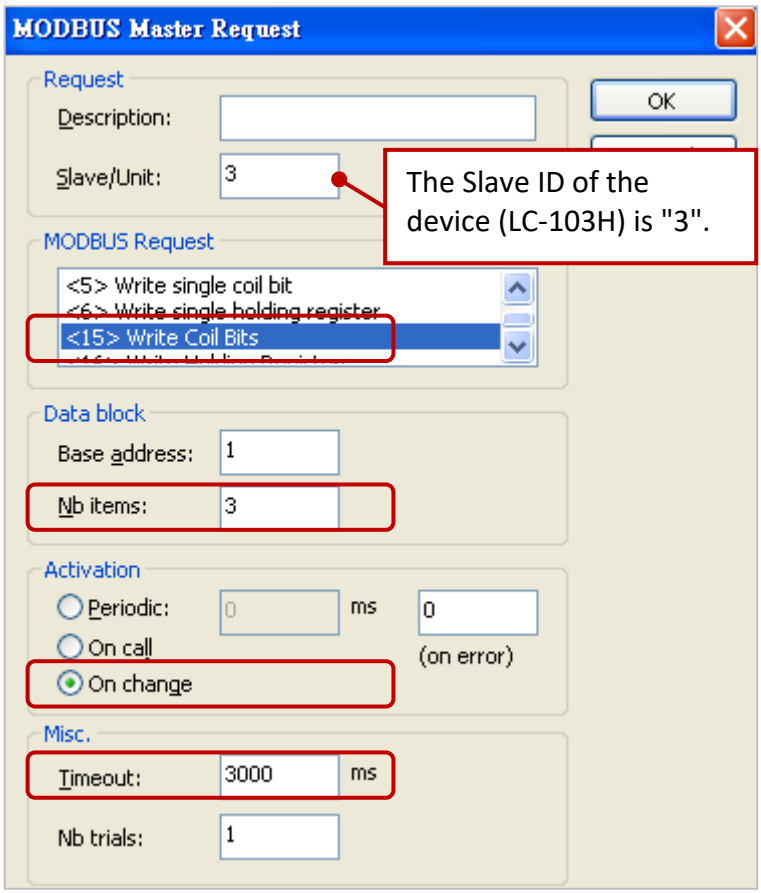


3. Click (or double-click) the second data block under port 502 to view the settings.
   The "Operation" is set to "Success counter" (or "Fail counter"), which means the value of the variable will be added 1 if succeed (or failed) to send the command. Also, the “Offset” value of these items must be set as “0”.

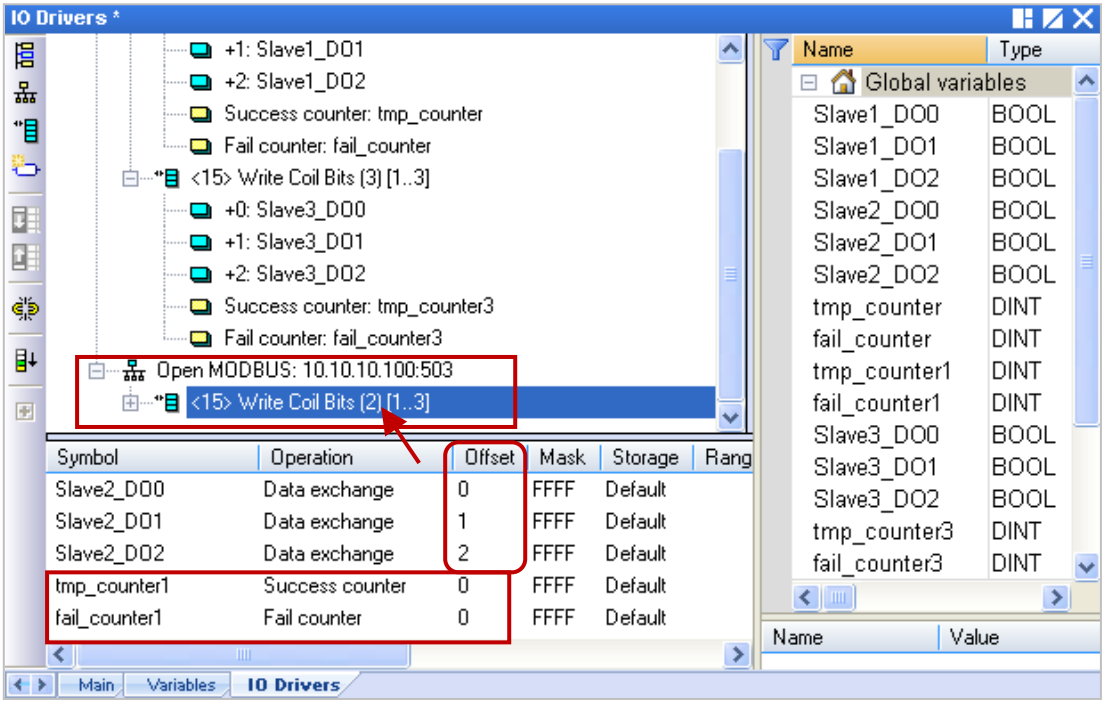


Description:

To write **three** DO data from the device (Slave ID = **3**) by sending the request when the program calls. No response for **3 seconds** can be regarded as abnormal.

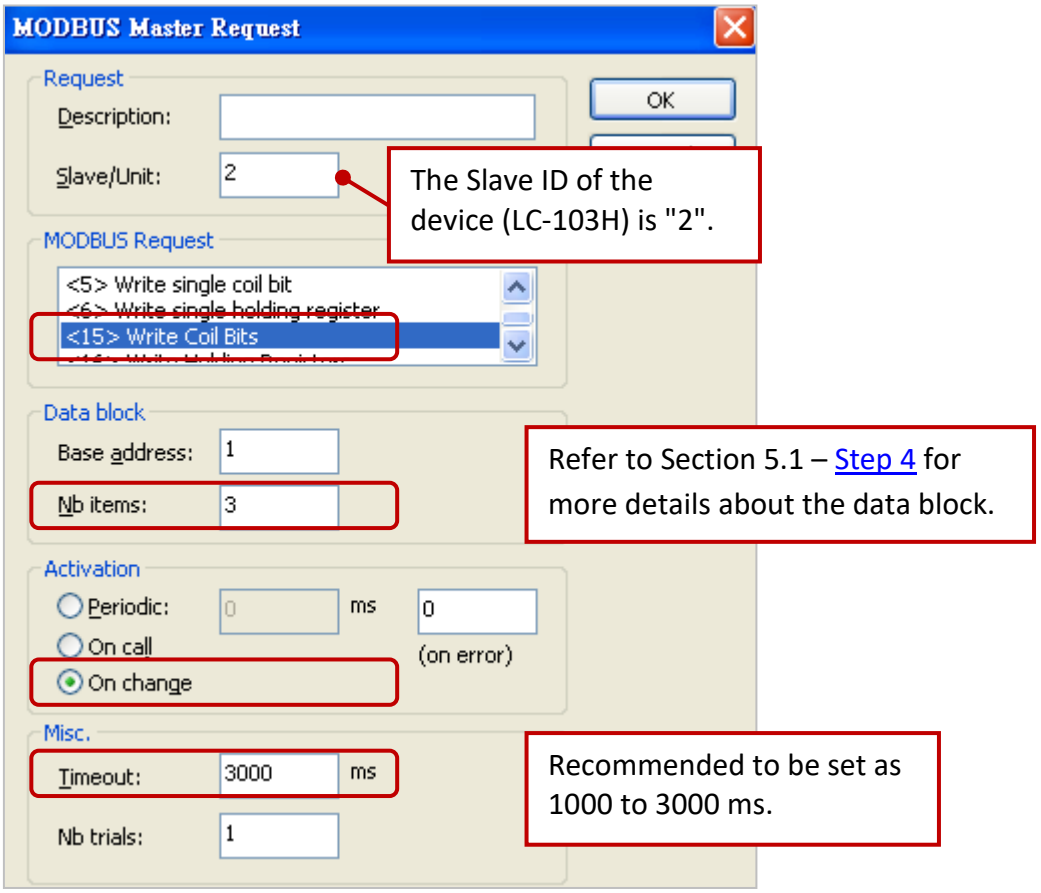


4. Click (or double-click) the first data block under port 503 to view the settings.
   The "Operation" is set to "Success counter" (or "Fail counter"), which means the value of the variable will be added 1 if succeed (or failed) to send the command. Also, the “Offset” value of these items must be set as “0”.



Description:

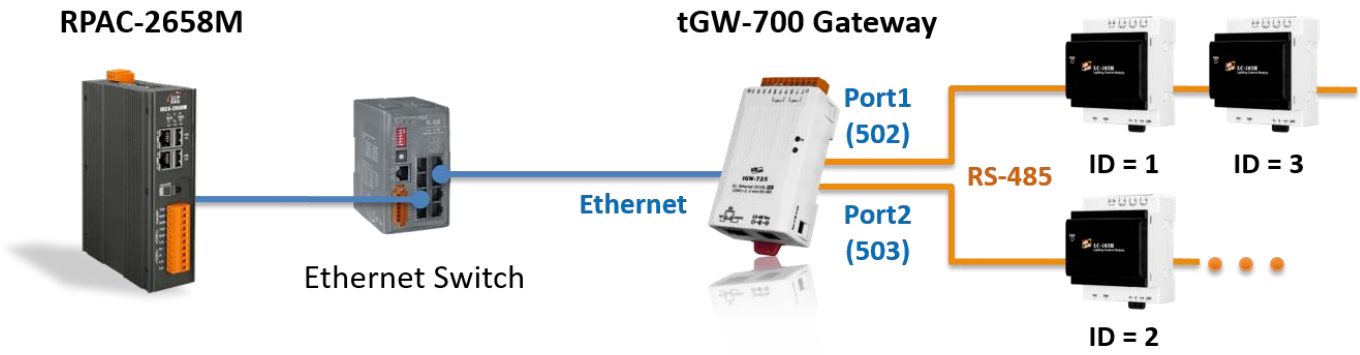
To write **three** DO data to the device (Slave ID = **2**) by sending the request when the program calls. No response for **3 seconds** can be regarded as abnormal.



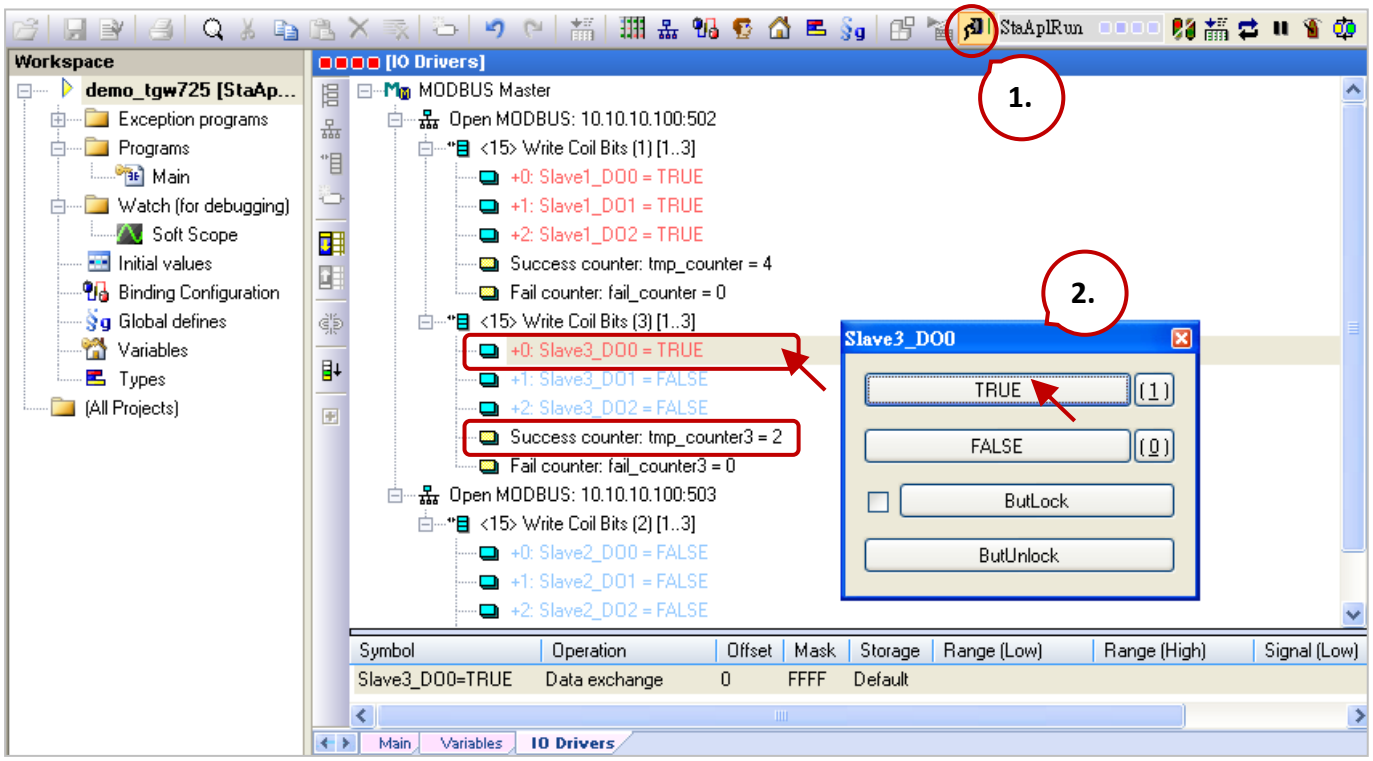
5.4.3 Test the Project (demo_tgw725.zip)

Please download this demo program to your Win-GRAF PAC before testing.

The Hardware Wiring:



Click the "On Line" button to connect with the PAC. In the “IO Drivers” window, double-click any DO variable to set it as "TRUE". If successes to write data, the "tmp_counter" value will be added "1".



Note: While the PAC is turned on, it will send a Modbus request to the Modbus slave device. So, you can see the value of "tmp_counter" is "1" at the beginning, which means succeed to write data.

Chapter 6 Retain Variable and Data Storage

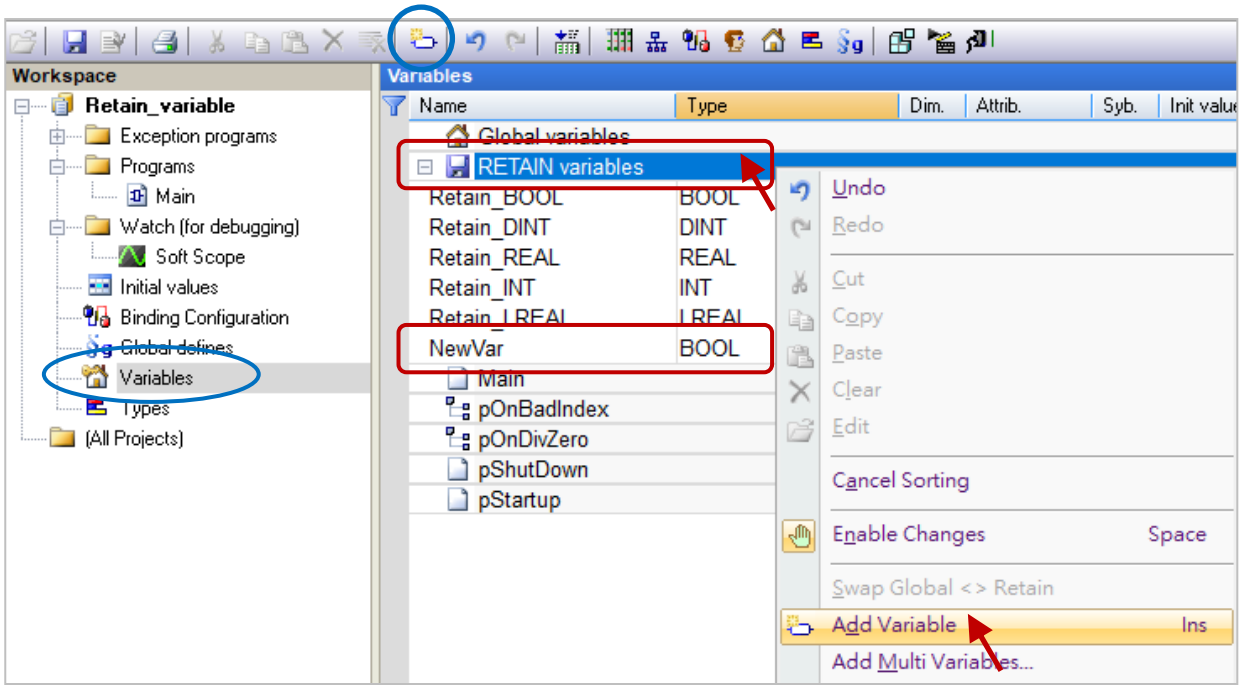
Retain variables are used to protect important data. The system can retain the latest variable data during an unexpected power failure. The user can either use RETAIN variables in the Variables window or use the RETAIN_xxx functions.

Note:

- To avoid internal data conflict, do not use both RETAIN variables and functions in the same project. Otherwise, functions will invalid.
- Using a great amount of RETAIN variables will cause the lower performance of the system due to access data to a disk frequently.

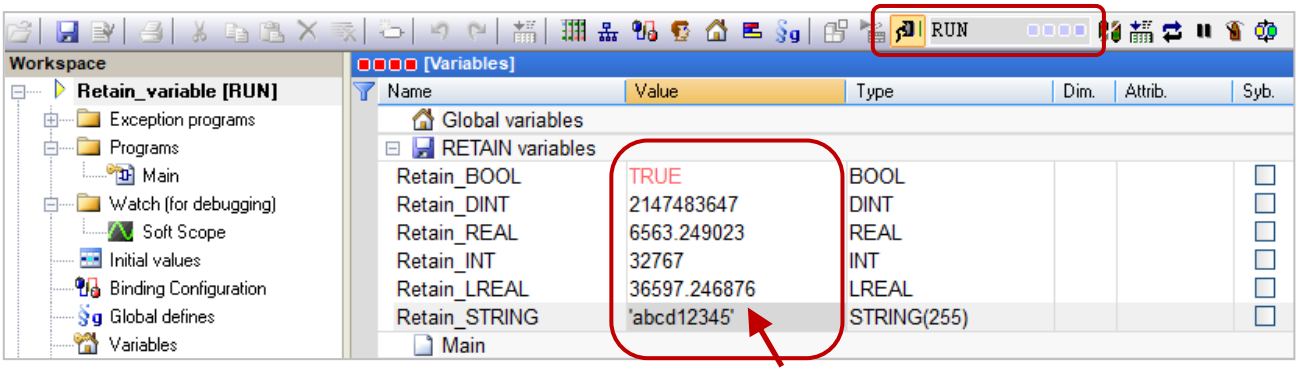
Add RETAIN variables

In the **Variables** window, right-click on **RETAIN variables** and click **Add Variable** to add a RETAIN variable. Next, set the name and type for the variable and press Enter.



Test RETAIN variables

Download the project to Win-GRAF PAC and change the value online. If the setting is correct, the modified value will be displayed after a reboot.



6.1 Using the RETAIN_xxx Function

This chapter lists the way to use the "RETAIN_ARY", "RETAIN_FLAG_CLR", "RETAIN_FLAG_GET", "RETAIN_FLAG_SET"and "RETAIN_VAR" Functions. The Win-GRAF PAC has a built-in storage memory for users to save retain variable data that will not be lost when the PAC is shutdown. The data still exists after the PAC is turned on.

Note:

1. To avoid internal data conflict, do not use both RETAIN variables and functions in the same project. Otherwise, functions will invalid.
2. Function **"Retain_Var()"** or **Retain_Ary()** can only be used in the first PAC Cycle or in the Cycle that performs the On-line Change. When a function calls in another Cycle, it will return "FALSE".
3. The user must specify an initial value for the retain variable. If not, an invalid value will be received after executing the function in the program on the PAC.

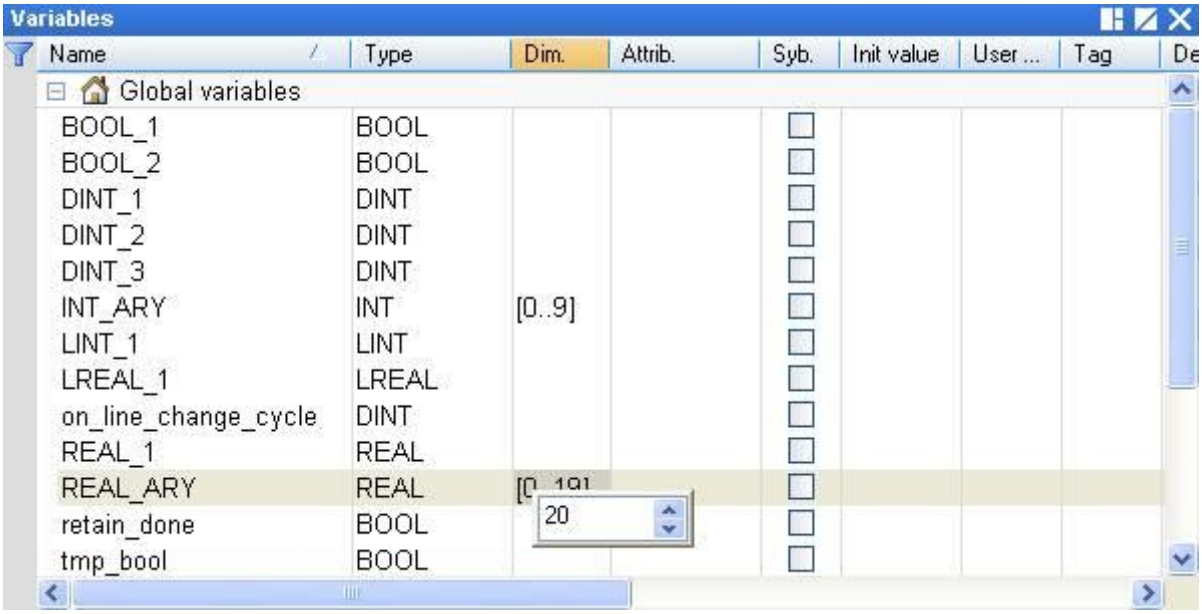
Download the demo program (**demo_retain.zip**) on the website. Click the menu commands **File - Add Existing Project - From Zip...** in the Workbench to open the project.

ST Program:

The demo describes Function Retain_Var() and Retain_Ary().

```
(* "on_line_change_cycle" is declared as DINT, a non-zero value indicates that on-line-change
   is being executed in that Cycle. "retain_done" is declared as BOOL with an initial value
   "FALSE". "tmp_bool" is declared as BOOL.  *)

 on_line_change_cycle  :=  GetSysInfo (_SYSINFO_CHANGE_CYCLE) ;
 if (retain_done = FALSE) or (on_line_change_cycle <> 0) then
      retain_done := TRUE ;  (* just do it one time *)
      tmp_bool := Retain_Var ( DINT_1 , 1) ; (* retain a DINT variable *)
      tmp_bool := Retain_Var ( DINT_2 , 2) ;
      tmp_bool := Retain_Var ( REAL_1 , 3) ; (* retain a REAL variable *)
      tmp_bool := Retain_Var ( BOOL_1 , 4) ; (* retain a BOOL variable *)
      tmp_bool := Retain_Var ( BOOL_2 , 5) ;

     (* retain 10 elements of an INT array variable at retain addr starting at 6. *)
      tmp_bool := Retain_Ary ( INT_ARY ,  6 ,  10) ;
     (* retain 20 elements of a REAL array variable at retain addr starting at 16. *)
      tmp_bool := Retain_Ary ( REAL_ARY , 16 , 20) ;
      tmp_bool := Retain_Var ( DINT_3 , 36) ;
      (* 64-bit variable can use only addr from 10,001 to 12,000 *)
      tmp_bool := Retain_Var ( LINT_1  , 10001) ;  (* retain a LINT variable (64-bit) *)
      tmp_bool := Retain_Var ( LREAL_1 , 10002) ; (* retain a LREAL variable (64-bit) *)
   end_if ;
```

You can check or set variables in the "Variables" window.



6.1.1 RETAIN_VAR: Retain a Variable

Tips:
Press “F1” key to see more details.

Name:
A variable name (DO NOT use Array variable or String).
Variable type can be BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, TIME, LINT, or LREAL.

Addr:
Data Type: DINT. The address number for retaining the variable can be 1 to 12,000.

Q:
Data Type: BOOL. TRUE: Ok; FALSE: Error.

Note:

1. One variable (or one element of the array) can be stored at only one address. Do Not assign the same address to two or more variables, or the variable data will be wrong.
2. When using the 64-bit data type (e.g., LINT or LREAL), set the address to 10001 to 12000.

6.1.2 RETAIN_ARY: Retain an Array Variable

Tips:
Press “F1” key to see more details.

Name[]:
An ARRAY variable name (DO NOT use String). The data type can be BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, TIME, LINT, or LREAL.

Addr:
Data Type: DINT.
The starting address number for retaining the array variable; can be 1 to 12,000.

Num:
Data Type: DINT.
The number of elements in the array variable to be retained.

For example:
If an array has 10 elements, setting the Num as 1 to 10 is correct. But, set it larger than 100 is wrong.
If an array has 5 elements, setting the Num as 1 to 5 is correct. But, set it larger than 5 is wrong.

Q:
Data Type: BOOL.
TRUE: Ok; FALSE: Error.

Note:

1. One variable (or one element of the array) can be stored at only one address. Do Not assign the same address to two or more variables, or the variable data will be wrong.
2. When using the 64-bit data type (e.g., LINT or LREAL), set the address to 10001 to 12000.

6.1.3 RETAIN_FLAG_SET/GET/CLR (Set/Get/Clear the Retain Flag)

How to Use:

"**Retain Flag**" is a flag (TRUE/FALSE) that is used to check if the retained data is a valid value and is stored in the memory. For the program to function properly, the user must specify an initial value for all retain variables. If not, the data read from the memory after turning on the PAC is normally a random value.

If "Retain_Flag_Set()" is set to TRUE which means all retain data have been specified an initial value. "Retain_Flag_Get()" is used to get the state of the flag and "Retain_Flag_Clr()" is used to clear the state of the flag.

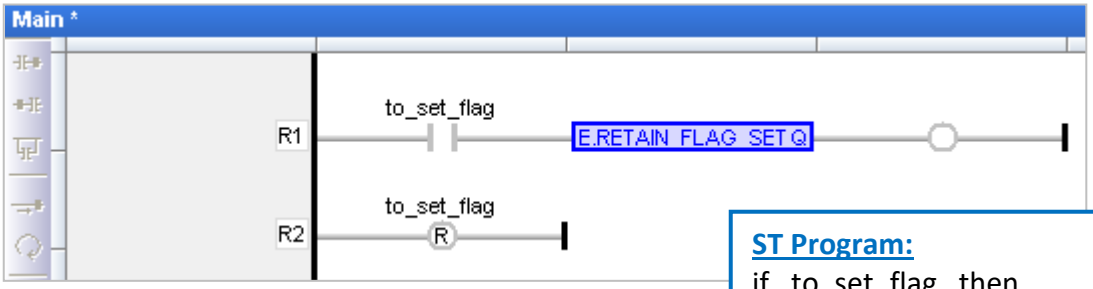
ST Program:

```
(*  "on_line_change_cycle" is declared as DINT, a non-zero value indicates that on-line-change is
    being executed in that Cycle.
   "retain_done" is declared as BOOL with an initial value "FALSE“ .
   "tmp_bool", "retain_flag" and "to_set_flag" are declared as BOOL.
*)

 on_line_change_cycle  :=  GetSysInfo (_SYSINFO_CHANGE_CYCLE) ;

 if (retain_done = FALSE) or (on_line_change_cycle <> 0) then
   retain_done := TRUE ;  (* just do it one time *)
   tmp_bool := Retain_Var( DINT_1 , 1) ;  (* retain a DINT variable *)
   tmp_bool := Retain_Var( DINT_2 , 2) ;
   tmp_bool := Retain_Var( REAL_1 , 3) ; (* retain a REAL variable *)
   tmp_bool := Retain_Var( BOOL_1 , 4) ; (* retain a BOOL variable *)

   (* ... After doing all the Retain Functions ... *)
   retain_flag := Retain_Flag_Get();
   if  (retain_flag = FALSE)  then
     (*If Retain variable does not set up any proper value, you can do some proper operation here. *)
     (* ... *)
   end_if ;

 end_if ;

  (* When all Retain variables are assigned proper values, remember to set the "to_set_flag" to
     "TRUE" for calling "Retain_Flag_Set() once, so that, when next time you use the
     "Retain_Flag_Get()", it can return "TRUE".   *)

 if  (to_set_flag = TRUE)  then
   to_set_flag := FALSE ;
   tmp_bool := Retain_Flag_Set() ;
 end_if ;
```

LD Program:

(Press the “F1” key to view the description of the setting.**)**

RETAIN_FLAG_SET: Set the retain flag.



Q: Data Type: BOOL. Always return TRUE.

ST Program:

```
if  to_set_flag  then
    to_set_flag  :=  FALSE  ;
    TMP_BOOL  :=  Retain_Flag_Set() ;
end_if ;
```

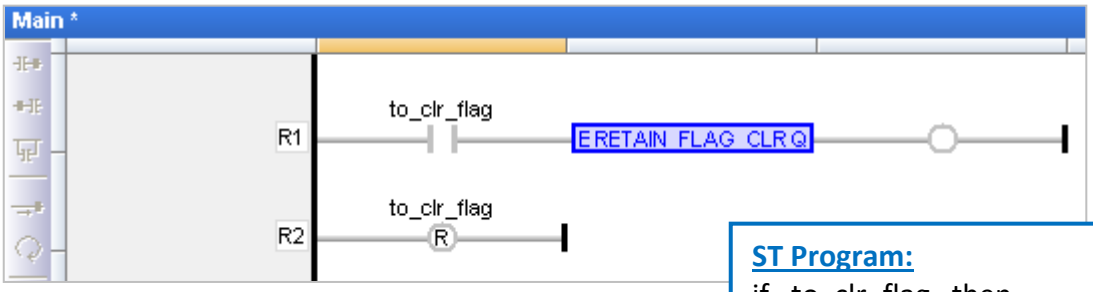
RETAIN_FLAG_GET: Get the state of the retain flag.

Q: Data Type: BOOL.
“TRUE”: the flag is set;
“FALSE”: the flag is not set.

ST Program:

```
retain_flag  :=  Retain_Flag_Get() ;
```

RETAIN_FLAG_CLR: Clear the retain flag.



Q: Data Type: BOOL. Always return TRUE.

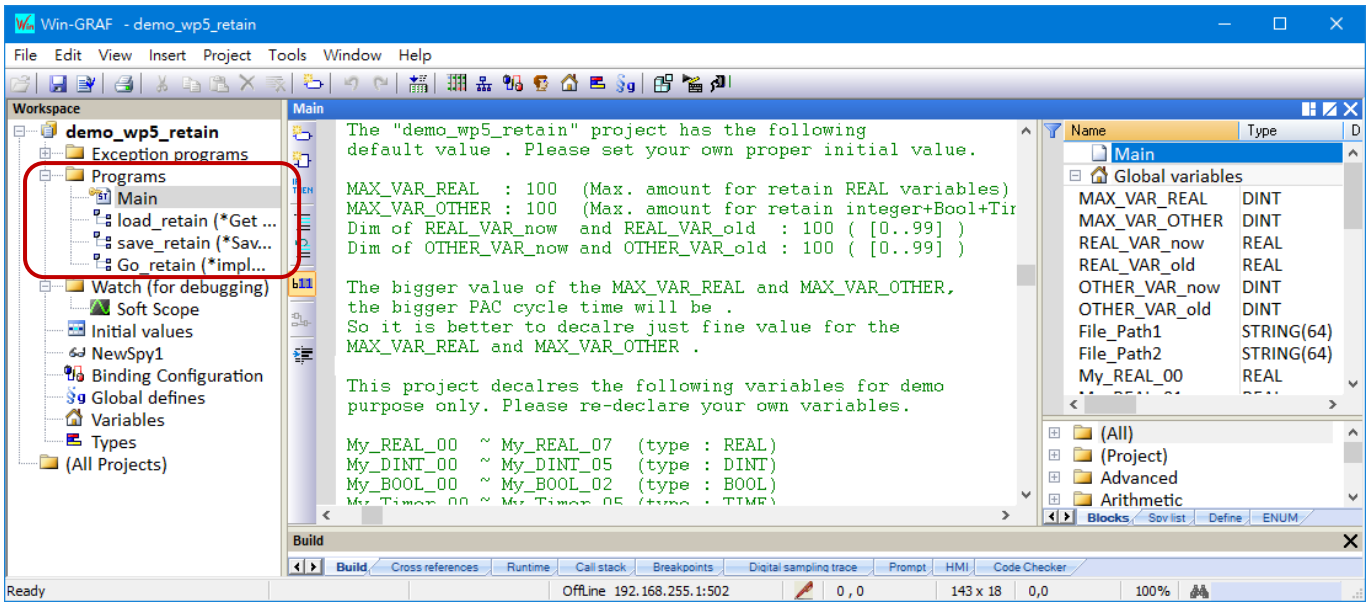
ST Program:

```
if  to_clr_flag  then
    to_clr_flag  :=  FALSE  ;
    TMP_BOOL  :=  Retain_Flag_Clr() ;
end_if ;
```

6.2 Save Retain Variables and Data to a File

Download the demo program (**demo_wp5_retain.zip**) on the website and then run the File menu commands "Add Existing Project" and "From Zip..." in the Win-GRAF Workbench to open the project.

This project includes an ST main program (i.e., Main) and three ST sub-programs (i.e., load_retain, save_retain, and Go_retain).



Limitation :

This project is not applicable for retaining variable data that changes frequently, e.g., the data will be changed every second or every minute. In this case, the retained data is stored in a file under the /System_Disk path. If the data changes frequently, updating the file will consume lots of CPU time, also reduce the performance of the PAC.

The table lists the setting value used in the"demo_wp5_retain". Also, change to a proper value to meet specific application requirements.

Variable Name	Value	Description
MAX_VAR_REAL	Init. = 100	The maximum amount of REAL retains variables can be used.
MAX_VAR_OTHER	Init. = 100	The total maximum amount of Integer, BOOL, and TIMER retain variables can be used.
REAL_VAR_now REAL_VAR_old	Dim. = 100	[0..99], which must be the same as the "MAX_VAR_REAL" value.
OTHER_VAR_now OTHER_VAR_old	Dim. = 100	[0..99], which must be the same as the "MAX_VAR_OTHER" value.

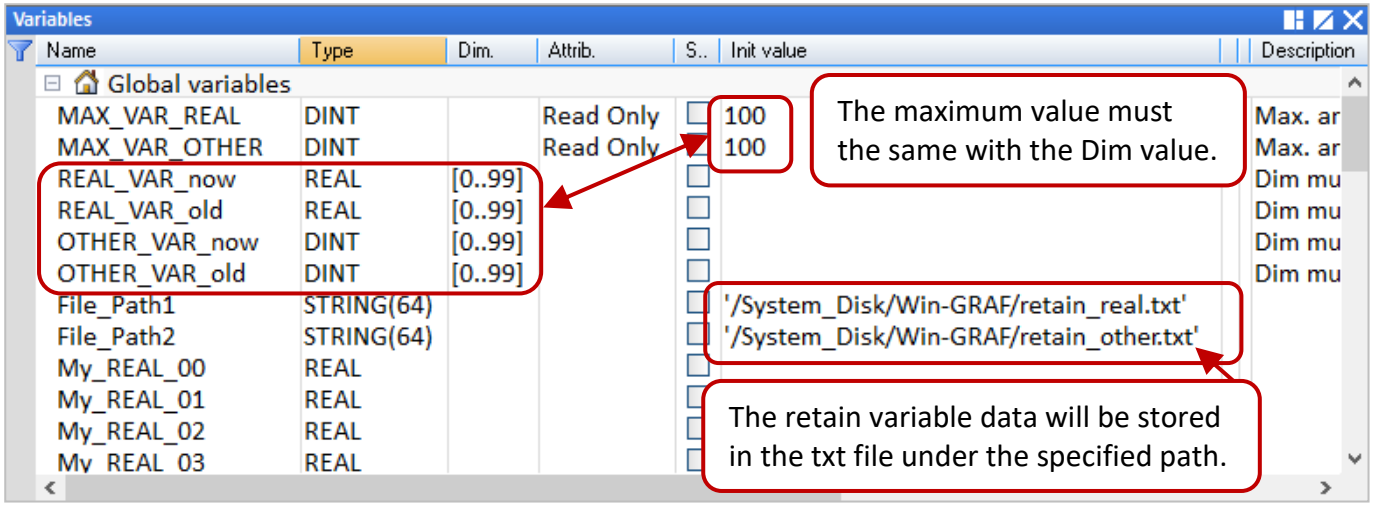
Note:

The bigger the value of the MAX_VAR_REAL and MAX_VAR_OTHER, the larger the PAC cycle time will be. So, specify the proper amount of MAX_VAR_REAL and MAX_VAR_OTHER.

The following variables are declared in this project.

My_REAL_00 ~ My_REAL_07 (Data Type: REAL)
My_DINT_00 ~ My_DINT_05 (Data Type: DINT)
My_BOOL_00 ~ My_BOOL_02 (Data Type: BOOL)
My_Timer_00 ~ My_Timer_05 (Data Type: TIME)

You can see more variables in the "Variables" window.



"Go_retain" sub-program is used to retain data. The following code can be modified by using the customized variables

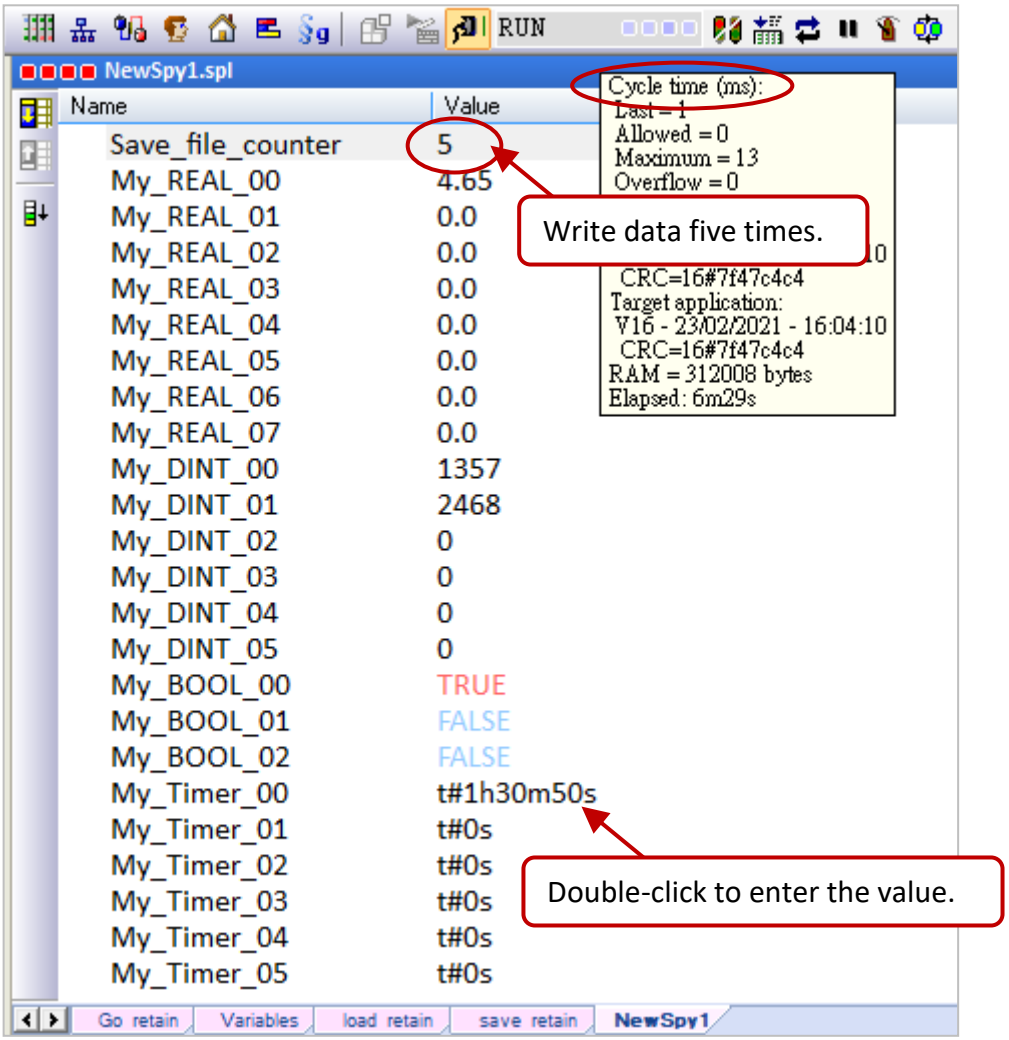
```
(* Add your REAL variables for retain here *)
(* ------------------------------------------------ *)
My_REAL_00 := REAL_VAR_now[0] ;
My_REAL_01 := REAL_VAR_now[1] ;
My_REAL_02 := REAL_VAR_now[2] ;
My_REAL_03 := REAL_VAR_now[3] ;
My_REAL_04 := REAL_VAR_now[4] ;
My_REAL_05 := REAL_VAR_now[5] ;
My_REAL_06 := REAL_VAR_now[6] ;
My_REAL_07 := REAL_VAR_now[7] ;
(* ------------------------------------------------ *)


(* Add your integer, BOOL, Timer variables for retain here *)
(* ................................................ *)
My_DINT_00 := OTHER_VAR_now[0] ;
My_DINT_01 := OTHER_VAR_now[1] ;
My_DINT_02 := OTHER_VAR_now[2] ;
My_DINT_03 := OTHER_VAR_now[3] ;
My_DINT_04 := OTHER_VAR_now[4] ;
My_DINT_05 := OTHER_VAR_now[5] ;
```

```
    My_BOOL_00 := Any_to_BOOL( OTHER_VAR_now[6] ) ;
    My_BOOL_01 := Any_to_BOOL( OTHER_VAR_now[7] ) ;
    My_BOOL_02 := Any_to_BOOL( OTHER_VAR_now[8] ) ;

    My_Timer_00 := Any_to_TIME( OTHER_VAR_now[9] ) ;
    My_Timer_01 := Any_to_TIME( OTHER_VAR_now[10] ) ;
    My_Timer_02 := Any_to_TIME( OTHER_VAR_now[11] ) ;
    My_Timer_03 := Any_to_TIME( OTHER_VAR_now[12] ) ;
    My_Timer_04 := Any_to_TIME( OTHER_VAR_now[13] ) ;
    My_Timer_05 := Any_to_TIME( OTHER_VAR_now[14] ) ;
    (* ..................................................... *)


(* Add your REAL variables for retain here *)
 (* ------------------------------------------------- *)
 REAL_VAR_now[0]  := My_REAL_00 ;
 REAL_VAR_now[1]  := My_REAL_01 ;
 REAL_VAR_now[2]  := My_REAL_02 ;
 REAL_VAR_now[3]  := My_REAL_03 ;
 REAL_VAR_now[4]  := My_REAL_04 ;
 REAL_VAR_now[5]  := My_REAL_05 ;
 REAL_VAR_now[6]  := My_REAL_06 ;
 REAL_VAR_now[7]  := My_REAL_07 ;
 (* ------------------------------------------------- *)


 (* Add your integer, BOOL, Timer variables for retain here *)
 (* ..................................................... *)
 OTHER_VAR_now[0]  := My_DINT_00 ;
 OTHER_VAR_now[1]  := My_DINT_01 ;
 OTHER_VAR_now[2]  := My_DINT_02 ;
 OTHER_VAR_now[3]  := My_DINT_03 ;
 OTHER_VAR_now[4]  := My_DINT_04 ;
 OTHER_VAR_now[5]  := My_DINT_05 ;

 OTHER_VAR_now[6]  := Any_to_DINT( My_BOOL_00 ) ;
 OTHER_VAR_now[7]  := Any_to_DINT( My_BOOL_01 ) ;
 OTHER_VAR_now[8]  := Any_to_DINT( My_BOOL_02 ) ;

 OTHER_VAR_now[9]  := Any_to_DINT( My_Timer_00 ) ;
 OTHER_VAR_now[10] := Any_to_DINT( My_Timer_01 ) ;
 OTHER_VAR_now[11] := Any_to_DINT( My_Timer_02 ) ;
 OTHER_VAR_now[12] := Any_to_DINT( My_Timer_03 ) ;
 OTHER_VAR_now[13] := Any_to_DINT( My_Timer_04 ) ;
 OTHER_VAR_now[14] := Any_to_DINT( My_Timer_05 ) ;
 (* ..................................................... *)
```

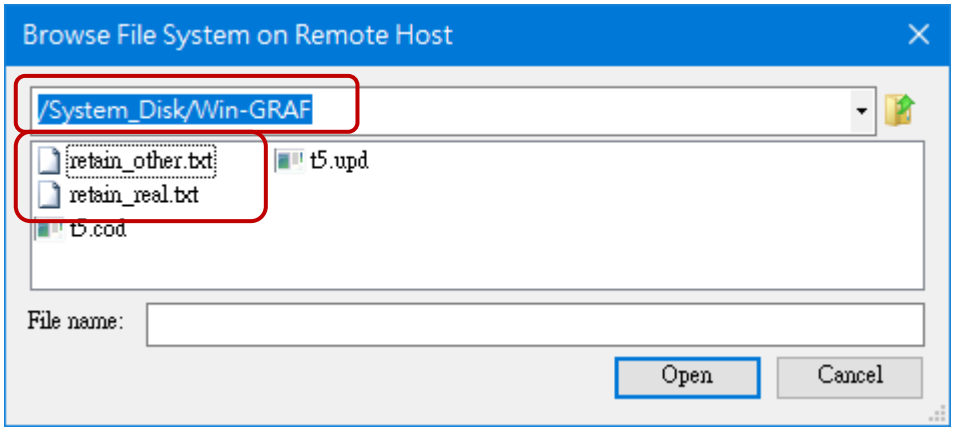
Test Project:

Make sure that the communication IP is correct and download the project to the PAC. In the Spy List window, all values are "0" (or "FALSE"). Change some values, and the .txt file (i.e., retain_real.txt or retain_other.txt) will automatically be created in the specified folder on PAC, and then write data to the file.

Note: "Save_file_counter" displays the cumulative number of times for updating the file. The project does not apply to the needs of updating the file frequently under the "/System_disk/..." path.



To check files in the Win-GRAF PAC by using the SSH tool (refer to Section 13.2)



6.3 Save the Retain Data to a FRAM

RPAC-2658M built-in a FRAM for users to read/write data that will not be lost due to PAC shutdown.

FRAM has the following features:

Advantages: Besides the retain variable, there is another way to save important data.

Disadvantages:

When using the retain variable, the CPU time is much less than 1 ms. It takes much more CPU time to read or write data in a FRAM. **Do not** call the "EEP_Read" or "EEP_Write" function frequently, or it will increase the cycle time of the PAC.

Note:

EEP_Read() and EEP_Write() functions can be used to read/write data from/to a FRAM or EEPROM.

ST Program:

```
(*   Declare "FIRST_CYCLE" as a “BOOL” variable and has an initial value “TRUE”.
     Declare "tmp_bool" as a “BOOL” variable.
     Declare "New_Val" and "Old_Val" as “DINT” variables.  *)

(*   Read the memory once in the first Cycle.   *)
  if  FIRST_CYCLE  then
    FIRST_CYCLE  :=  FALSE ;  (* means it is not the first Cycle any more *)
    tmp_bool :=  EEP_Read ( 1 , New_Val) ;
  end_if ;

  (* Safe programming: write to the memory only when the value is changed.  *)
  if  New_Val <> Old_Val  then
    Old_Val  :=  New_Val ;
    tmp_bool :=  EEP_Write ( 1 , New_Val ) ;
  end_if ;
```

```
(* Hazardous: The memory may be destroyed very soon. *)
(*   Declare "FIRST_CYCLE" as a “BOOL” variable and has an initial value “TRUE”.
     Declare "tmp_bool" as a “BOOL” variable.
     Declare "New_Val" and "Old_Val" as “DINT” variables.  *)

(*   Read the memory once in the first Cycle.   *)
  if  FIRST_CYCLE  then
    FIRST_CYCLE  :=  FALSE ;  (* means it is not the first Cycle any more *)
    tmp_bool  :=  EEP_Read ( 1 , New_Val) ;
  end_if ;

 (* Hazardous programming: Write the "New_Val" value to the memory one time in every Cycle. *)
  tmp_bool  :=  EEP_Write ( 1 , New_Val ) ;
```

6.3.1 EEP_READ (Read a Value from the FRAM)

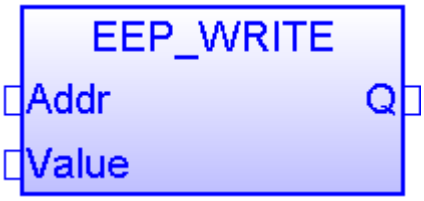
Tip:
Press “F1” key to see the detailed setting descriptions.

Addr: (Data Type: “DINT”)
The address can be 1 to 1200.
When using the 64-bit data type (e.g., LINT or LREAL), set the "Addr" to 1001 to 1200.

@Name :
Specify a variable name to store data read from the FRAM.
DO NOT use string variable. The data type can be BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, TIME, LINT, or LREAL.

Q:
Data Type: BOOL. TRUE: Ok; FALSE: Error.
If the data type of the variable is REAL or LREAL but the read data is not a REAL value or an error occurs, "Q" will return FALSE. The variable value "0.0" indicates the data is not a REAL value.

6.3.2 EEP_WRITE (Write a Value to the FRAM)



Addr: (Data Type: “DINT”)
The address can be 1 to 1200. If the variable type of the "Value" parameter is 64-bit data (e.g., LINT or LREAL), the "Addr" can be 1001 to 1200 only.

Value :
Specify the value to write to the FRAM.
DO NOT use string variable. The value type can be BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, TIME, LINT, or LREAL.

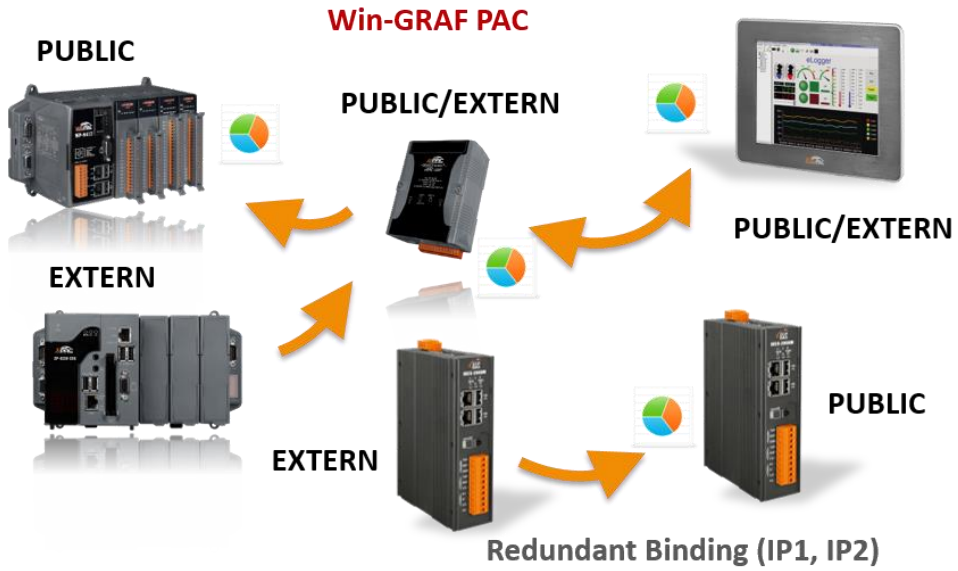
Q:
Data Type: BOOL. TRUE: Ok; FALSE: Error.

Chapter 7 Exchange Data between PACs (Data Binding)

The Binding feature allows multiple Win-GRAF PACs to exchange data to each other in an event-triggered manner that is more efficient than polling. Win-GRAF offers two ways to configure Binding:

- **PUBLIC:** To make the data publicly available for other PACs or the C program in the same PAC.
- **EXTERN:** To get data from the other PAC.

Application Diagram:



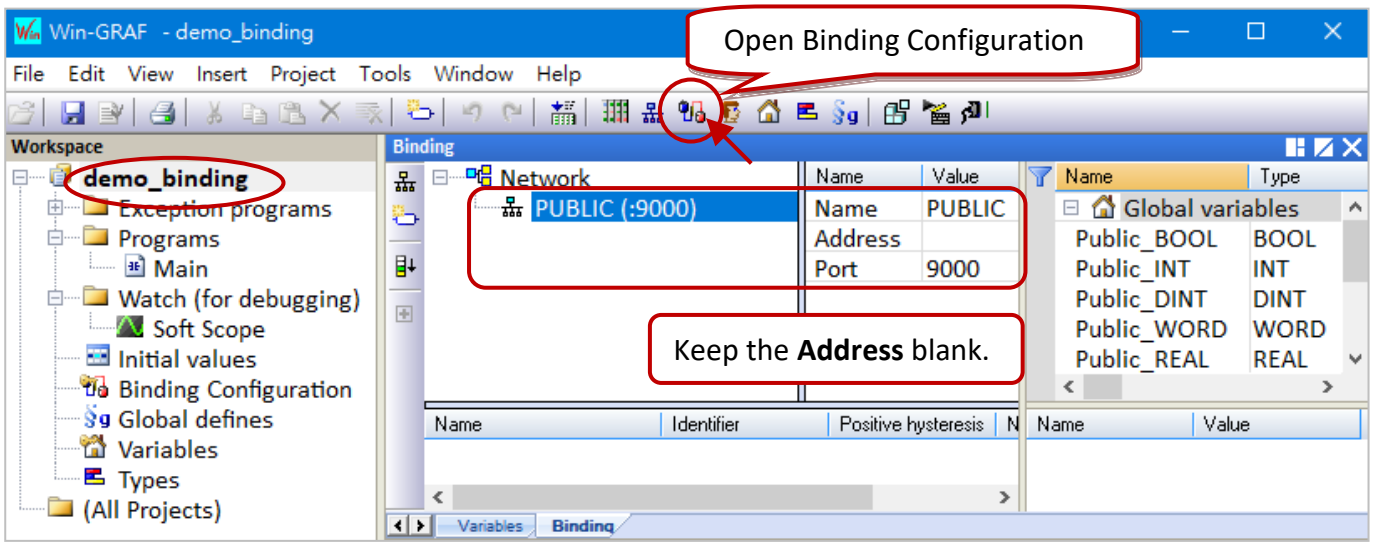
Note:

The maximum number of "Binding" features available for each RPAC-2658M is **32** (EXTRN).

The "PUBLIC" Setting: Make the variable data accessible to other PACs.

1. Click the "Open Binding Configuration" button to open the "Binding" window.

   "PUBLIC (: 9000)" indicates that the variable data is made public via the port number "9000" (the number is fixed, do not change it).

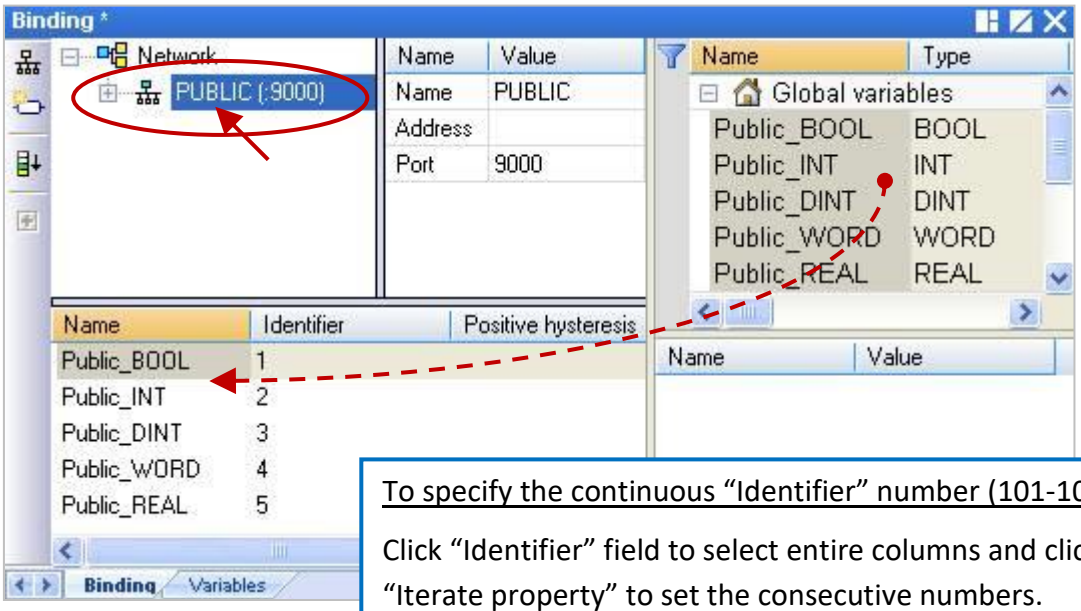


2. Before making the variable data public, adding these variables first. The following table lists all the necessary variables in this example.

Variables Name	Data Type
Public_BOOL	BOOL
Public_INT	INT
Public_DINT	DINT
Public_WORD	WORD
Public_REAL	REAL

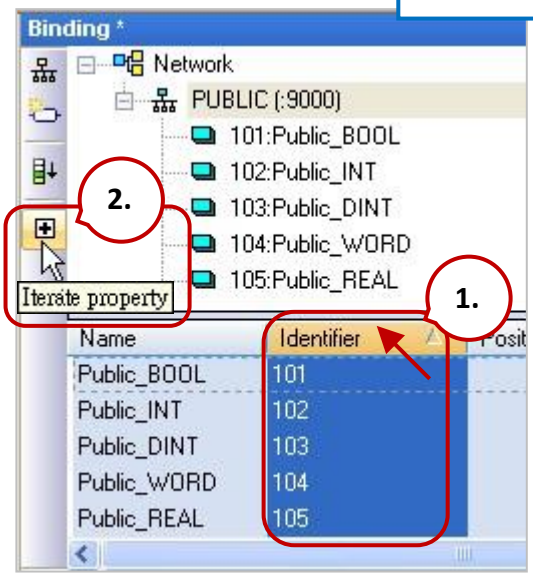
3. Click on the "PUBLIC (: 9000)" and drag-and-drop the needed variables into the ID mapping area, then the ID will automatically be generated. For other PACs, using the same ID to read/write data on this PAC.

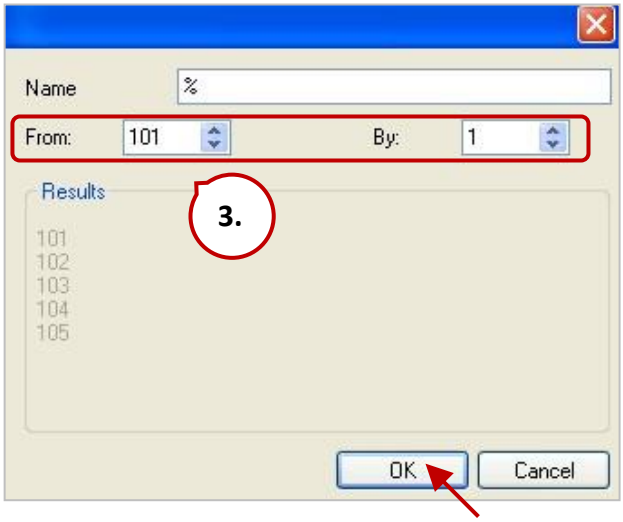
Note: Up to 8192 variables can be used in the Public setting. The number of "Identifier" can only be "1 to 8192".



To specify the continuous "Identifier" number (101-105)

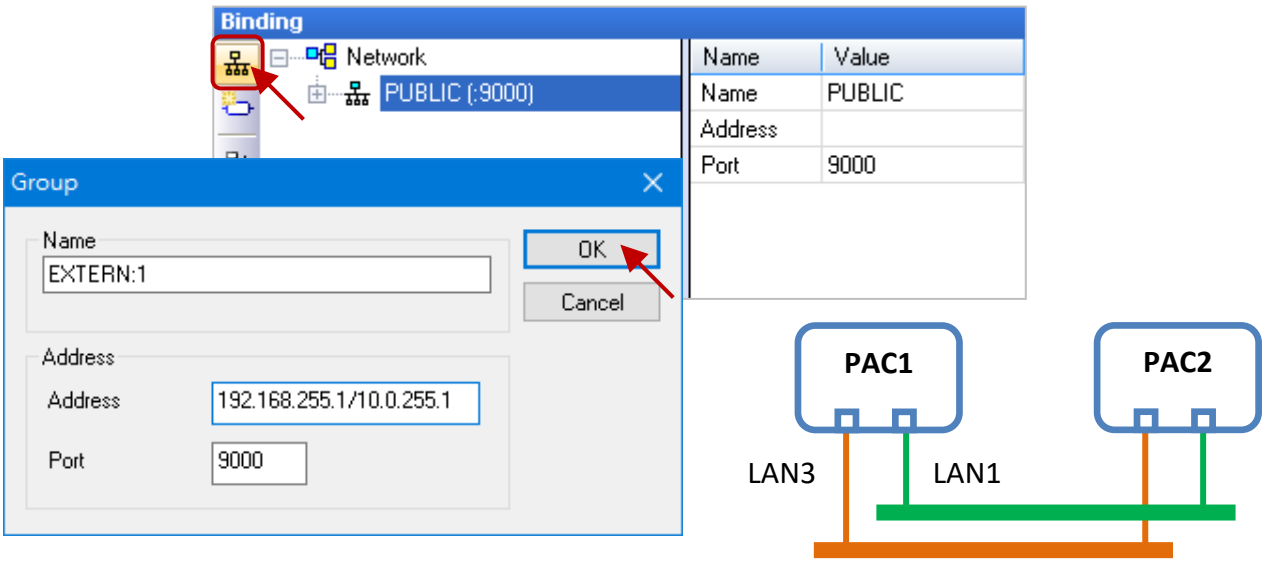
Click "Identifier" field to select entire columns and click "Iterate property" to set the consecutive numbers.





The "EXTERN" Setting: To get data from the other PAC.

4. Click the "Insert Master/Port" button on the left and the "Group" window will appear. Follow the description below to set these fields and click the "OK“ button.



Name: Enter a custom name.

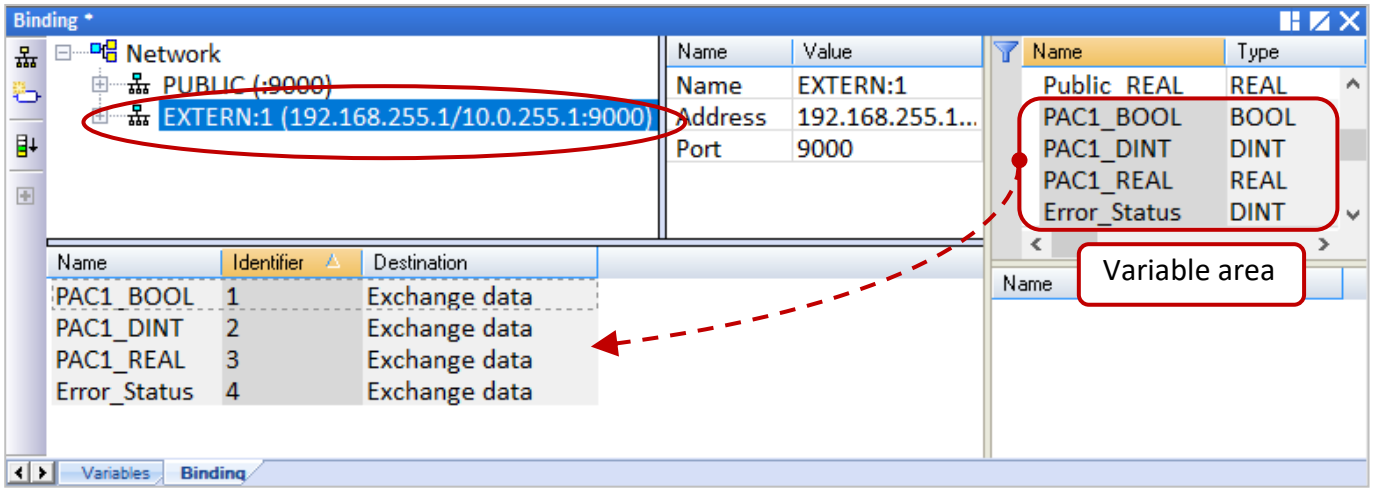
Address: Enter the IP address of the PAC that you want to get data from (e.g., "192.168.255.1"). Two IP Addresses (LAN1: 192.168.x.x, LAN3: 10.0.x.x) can be set if the PAC has two Ethernet ports. When one IP address is nonfunctional, it will try to connect to the second IP address.

Port: The fixed port number "9000", do not change it.

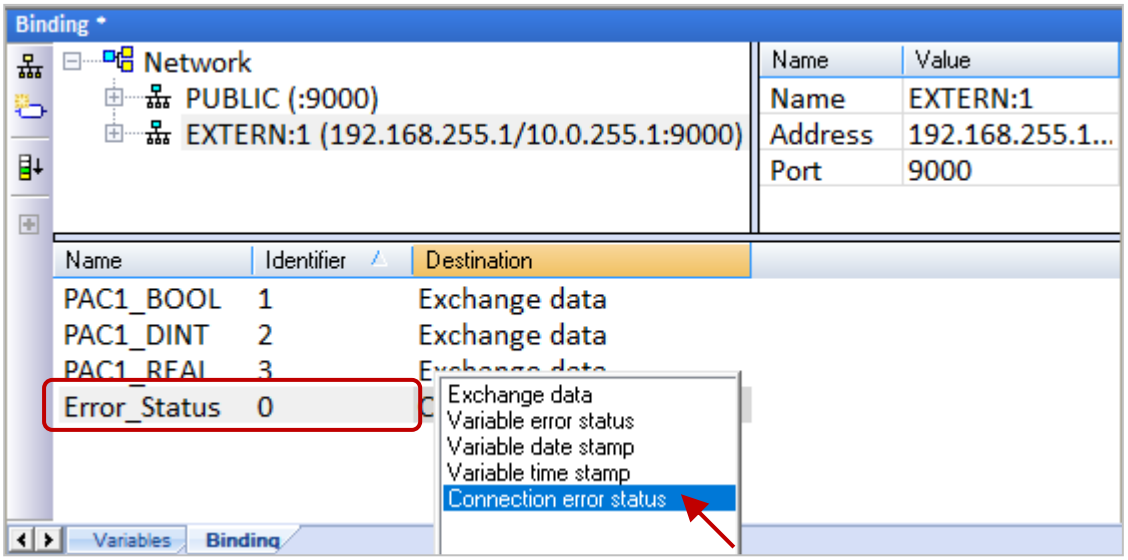
5. In the Variable area, add variables for getting data. The table lists the needed variables in this example.

Variables Name	Data Type
PAC1_BOOL	BOOL
PAC1_DINT	DINT
PAC1_REAL	REAL
Error_Status	DINT

6. Drag-and-drop these variables into the ID mapping area of the "EXTERN:1" and the identifier will automatically be generated. Change to the same ID as the public variable in remote public PAC.



7. The "Error_Status" variable is used to know the communication status of the PAC. Set the ID to "0" and double-click its "Destination" field, then set it as "Connection error status".



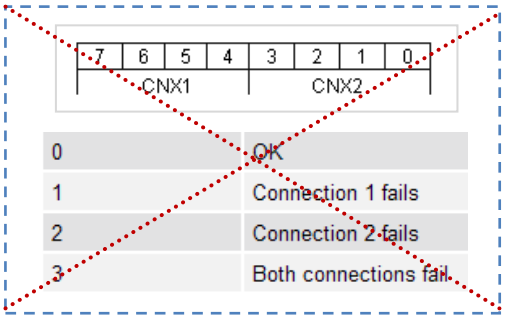
Note:

If two IP addresses are set in the EXTERN group, "Error_Status" will return two communication statuses (8-bit). Bit 0-3 represents the connection status of the first IP and its value is 15 if all bits are 1; Bit 4-7 represents the connection status of the second IP and its value is 240 if all bits are 1. As long as the value is not equal to “0” which means the communication is abnormal.

IP2 Connection Status				**IP1** Connection Status				**Status Description**
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
0				0				Connection OK
0				≠0 (1 ~ 15)				IP1 Connection error
≠ 0 (16 ~ 240)				0				IP2 Connection error
≠ 0				≠ 0				IP1 and IP2 Connection error

Notice:

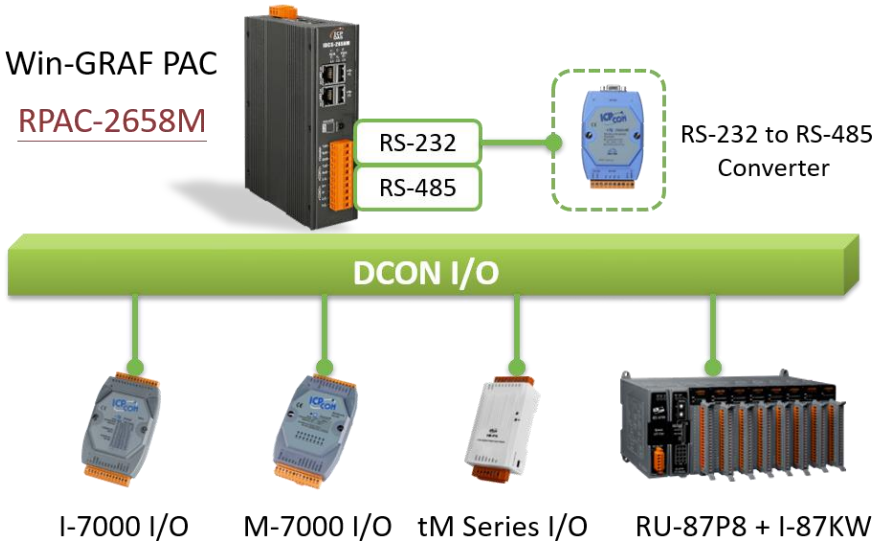
In "HTML Help" (press "F1"), the description of the "**Connection status**" and "**Variable status**" in the topic "Networked applications - Dual binding on redundant ETHERNET" **is wrong**, please ignore it. It does not conform to the usage of Win-GRAF PAC.



Chapter 8 Connecting DCON I/O Modules

Win-GRAF PACs can connect to remote DCON I/O modules via the COM Port (RS-485). Up to 16 DCON ports can be enabled on a PAC, and up to 50 remote DCON modules can be connected to each port (no more than 32 recommended).

For "I-87KW" series I/O modules, an RS-485 I/O expansion unit is required (e.g., RU-87P4/8).



Product Page - Remote I/O Module and Unit

Product Page - DCON Utility Pro

Before using remote DCON I/O modules, configuring each one with "DCON Utility Pro" software.

Parameter	The Setting Value
Protocol	DCON
Address	1 to 255
Baudrate	'9600' is recommended and must the same as the PAC setting
Checksum	'Enabled' is recommended and must the same as the PAC setting
Data format and I/O settings	Set them according to application requirements

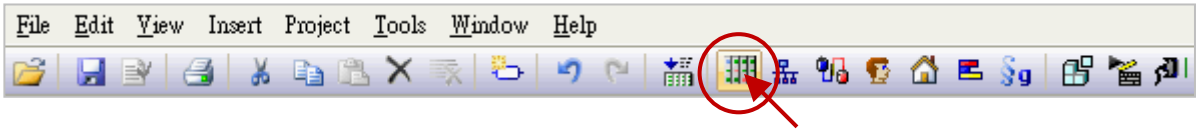
Note:

1. When using Analog Input (AI) modules, set the Data format to **"2's Complement"**.
2. When using the Analog Output (AO) module, set the Data format to **"Engineering".**

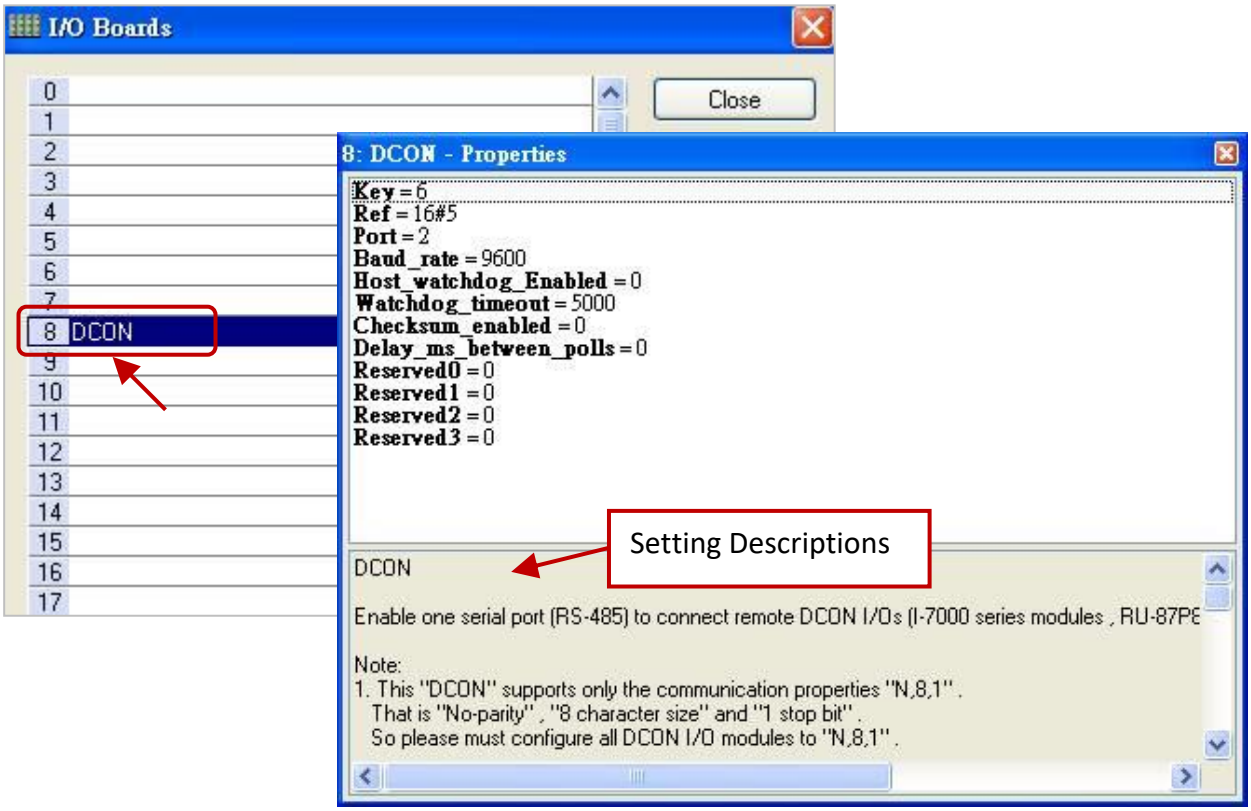
8.1 Setting the "DCON" I/O Board

"DCON" I/O board allows enabling one RS-485 port to connect remote DCON I/O modules. Up to 16 DCON ports can be enabled for one PAC.

1. Click the "Open I/Os" button to open the “I/O Boards” window.



2. Double-click the slot number to add the "DCON" I/O board, and then double-click "DCON" to open the “Properties” window.



Parameters:

Note: When using DCON protocol, set the data format of I/O modules as "N,8,1" which stands for no parity-bit, eight data-bit, and one stop-bit.

Port: COM port number (1 ~ 37, depends on the PAC.)

Baud_rate: Communication baud rate in bps, can be set to 1200, 2400, 4800, 9600, 19200, 38400, 57600, and 115200 (bps). ‘9600’ will be used if setting the other value.

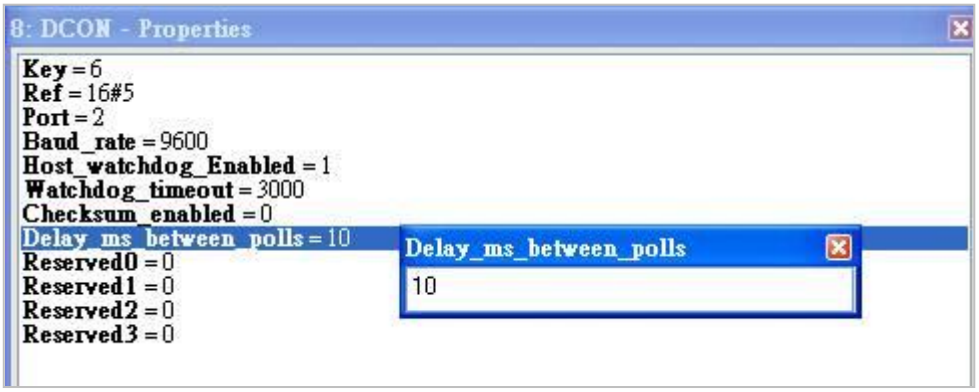
Host_watchdog_Enabled: 1: Host-watchdog is enabled. 0: Disabled.
‘1’ will be used if setting a non-zero value.

Watchdog_timeout: Unit: ms, can be set to 3000 to 25500 (i.e., 3 to 25.5 seconds).
'3000' will be used if the setting is less than 3000;
'25500' will be used if the setting is larger than 25500.
If "Host_watchdog_Enabled" is set to "0", this setting will be ignored.

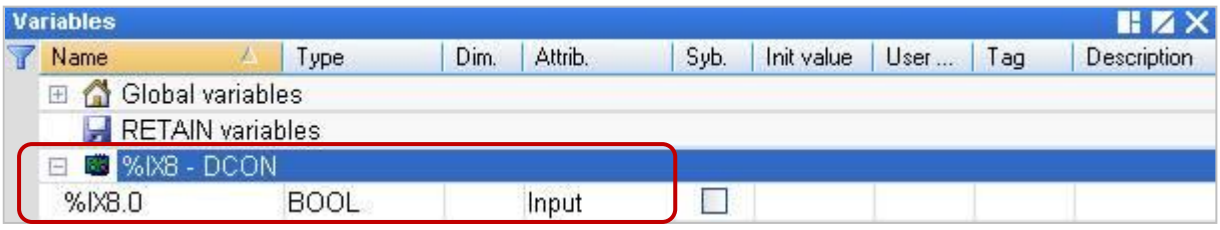
Checksum_enabled: 1: Checksum is enabled. 0: Disabled.
'1' will be used if setting a non-zero value.
(For communication security, it is recommended to enable Checksum)

Delay_ms_between_polls: Unit: ms, the default value is 0. Valid range is 0 to 1000.
'0' will be used if the setting is less than 0;
'1000' will be used if the setting is larger than 1000.
If no wireless module is connected, set to a smaller value (e.g., 0 to 10).
If wireless modules are connected (e.g., the ZigBee I/O), set to a larger value (e.g., 30 to 100 or other values). The larger the value, the slower the Polling efficiency will be.

3. Double-click any item to enter the value.



4. After adding the "DCON" in the "I/O Boards" window, a "BOOL" input variable will be automatically added to the "Variables" window. The variable is used to display the communication status of the COM Port on Win-GRAF PAC. (TRUE: OK; FALSE: error.).



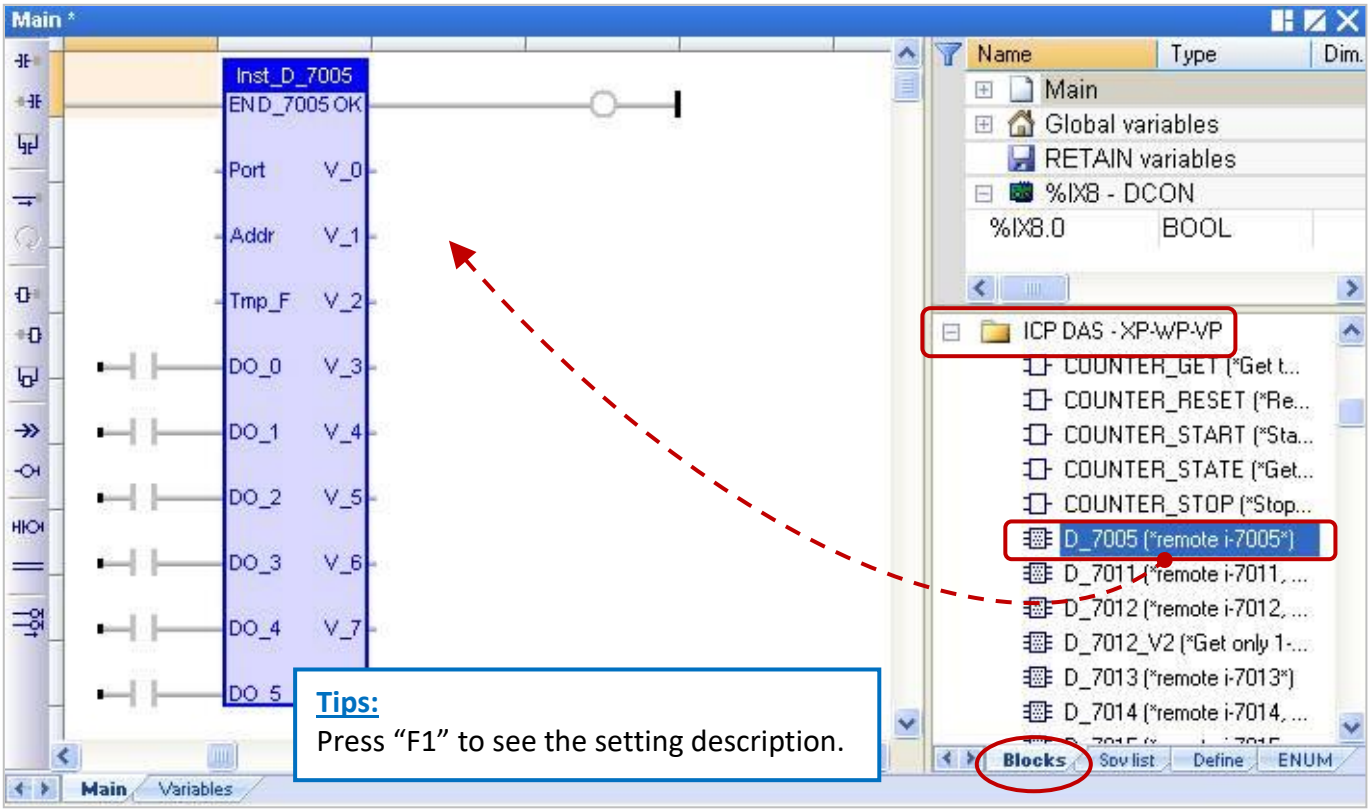
8.2 Using I/O Function Blocks

Win-GRAF supports many remote DCON I/O modules, open the "Library Manager" (Refer to Section 1.2.2) or the Help (press "F1") to find out the description of these function and function blocks.

This section describes the usage of "D_7065", "D_7018Z", "D_7083", "D_87084_freq", "D_87084_cnt4", "D_87084_cnt8", "DL_100T485", and "D_GPS721" function blocks.

Model	Description
I-7065D-G	4-ch Isolated DI and 5-ch **Power Relay** Module with LED Display
I-7065AD-G	4-ch Isolated DI and 5-ch **AC-SSR Output** Module with LED Display
I-7065BD-G	4-ch Isolated DI and 5-ch **DC-SSR Output** Module with LED Display
I-7018Z-G/S	10-ch Thermocouple Input Module
I-7083D-G	3-axis, 32-bit Encoder/Counter Input Module with LED Display
I-7083BD-G	3-axis, 32-bit Encoder/Counter Input Module with Virtual Battery Backup and LED Display
I-87084W-G	4/8-ch Counter/Frequency/Encoder Input Module
DL-100T485	IP66 Remote Temperature and Humidity Data Logger with LCD Display (RS-485)
GPS-721	GPS Receiver Module with RS-485 (Asia Only)

In the **Blocks** pane of the LD Program, there are many functions and function blocks in the "ICP DAS - XP-WP-VP" folder. Drag and drop the desired function to the program editing area to use it.



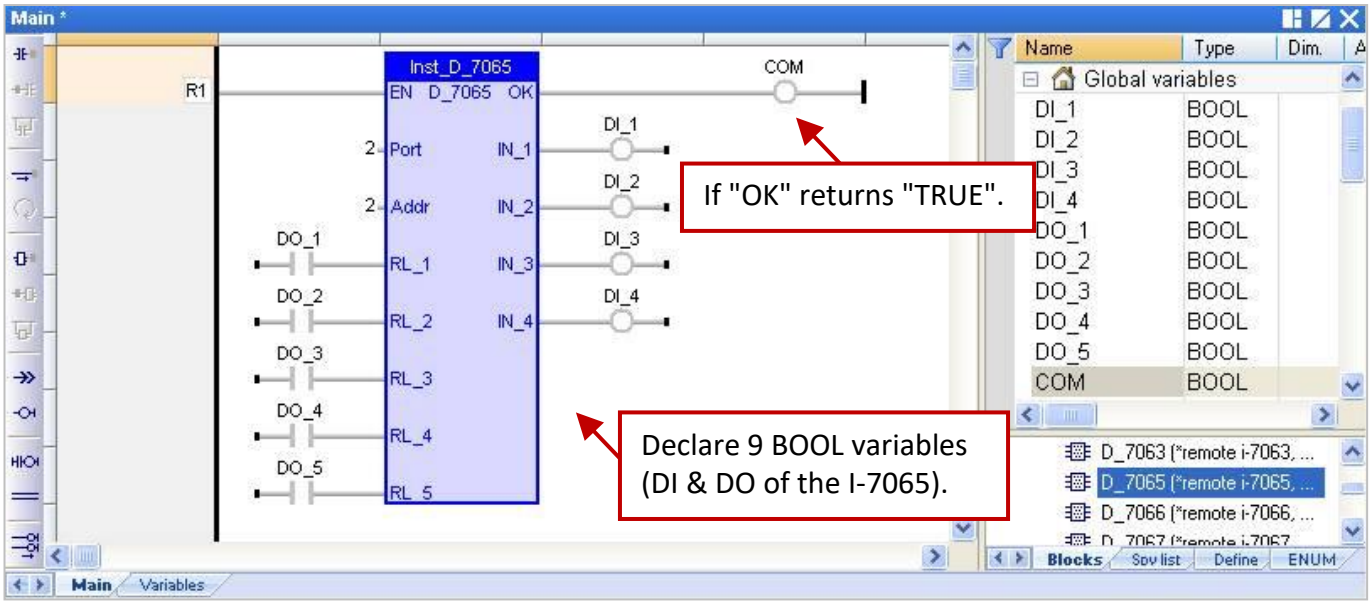
8.2.1 “D_7065” Function Block

“D_7065”: Used to connect a remote I-7065D, I-7065AD, or I-7065BD DCON module.

Note:

1. Make sure that the I/O module has been configured by using DCON Utility Pro (refer to Chapter 8).
2. Make sure that "DCON" has been added in the "I/O boards" window (refer to Section 8.1).
3. Download the demo program (**demo_d_7065.zip**) on the website. Click the menu commands **File - Add Existing Project - From Zip...** in the Workbench to open the project.
4. Only when the communication status of the module is normal (TRUE), the received DI data is correct.

Example: Connect the I-7065 (Addr. = 2) via COM2 on PAC. It has 4 DI and 5 Relay output channels.



Input Parameters	Data Type	Description
EN	BOOL	TRUE: Enabled; FALSE: Disabled.
Port	DINT	COM port number (can be 1 to 37, depends on PAC). Note that specify a constant value, not a variable value.
Addr	DINT	The Net-ID address of the module can be 1 to 255. Note that specify a constant value, not a variable value.
RL_1 to RL_5	BOOL	5-Ch DO values.

Output Parameters	Data Type	Description
OK	BOOL	TRUE: Communication is Ok. FALSE: Communication failed.
IN_1 to IN_4	BOOL	4-Ch DI values.

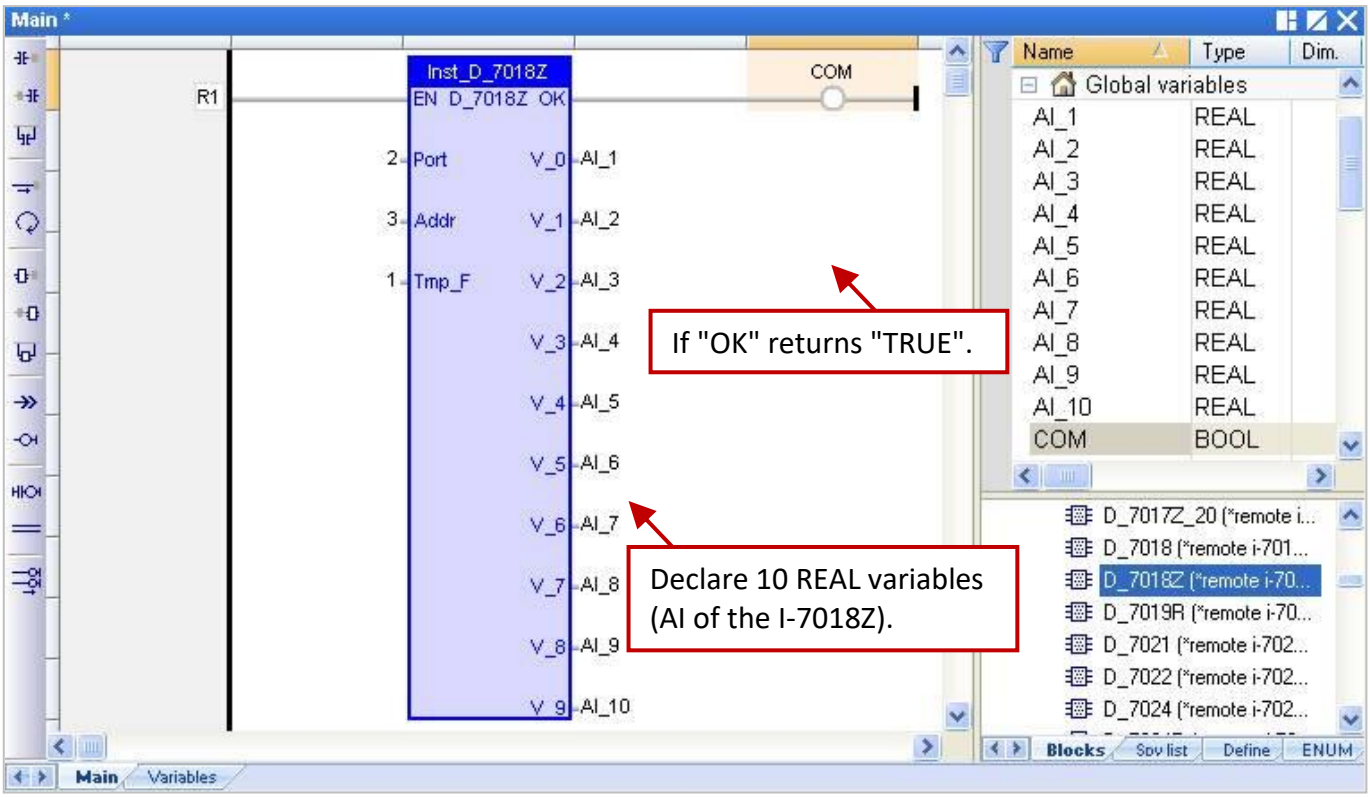
8.2.2 “D_7018Z” Function Block

“D_7018Z”: Used to connect a remote I-7018Z DCON module to measure voltage, current, or temperature.

Note:

1. Make sure that the I/O module has been configured by using DCON Utility Pro (refer to Chapter 8). Set the data format of the AI module to "**2’s complement**", or Win-GRAF cannot read data correctly.
2. Make sure that "DCON" has been added in the "I/O boards" window (refer to Section 8.1).
3. Download the demo program (**demo_d_7018z.zip**) on the website. Click the menu commands **File - Add Existing Project - From Zip...** in the Workbench to open the project.
4. Only when the communication status of the module is normal (TRUE), the received AI data is correct.

Example: Connect the I-7018Z (Addr. = 3) to measure the Celsius temperature via COM2 on PAC.



Input Parameters	Data Type	Description
EN	BOOL	TRUE: Enabled; FALSE: Disabled.
Port	DINT	COM port number (can be 1 to 37, depends on PAC). Note that specify a constant value, not a variable value.
Addr	DINT	The Net-ID address of the module can be 1 to 255. Note that specify a constant value, not a variable value.
Tmp_F	DINT	Temperature Format can be 1 or 2: 1: Celsius. 2: Fahrenheit. 'Celsius' will be used if setting it as another value.

Output Parameters	Data Type	Description
OK	BOOL	TRUE: Communication is Ok. FALSE: Communication failed.
V_0 to V_9	REAL	10-Ch AI value. The type code can be set in DCON Utility Pro. If a Temperature type is set, the unit of data is 'Degree‘. E.g., the value of 25.75 stands for 25.75 degrees. If a Voltage type is set, the unit of data is 'V‘. E.g., the value 0.85421 stands for 0.85421 V or 854.21 mV. If a Current type is set, the unit of data is 'mA‘. E.g., the value of 1.5567 stands for 1.5567 mA.

Open-wire Detection:

If the temperature data is greater than "9000.0", it means that

1. The temperature sensor may be broken-line.
2. The temperature sensor may be broken.
3. The setting on the DCON module does not match it on the temperature sensor.
4. The resistance measured by the sensor is incorrect.

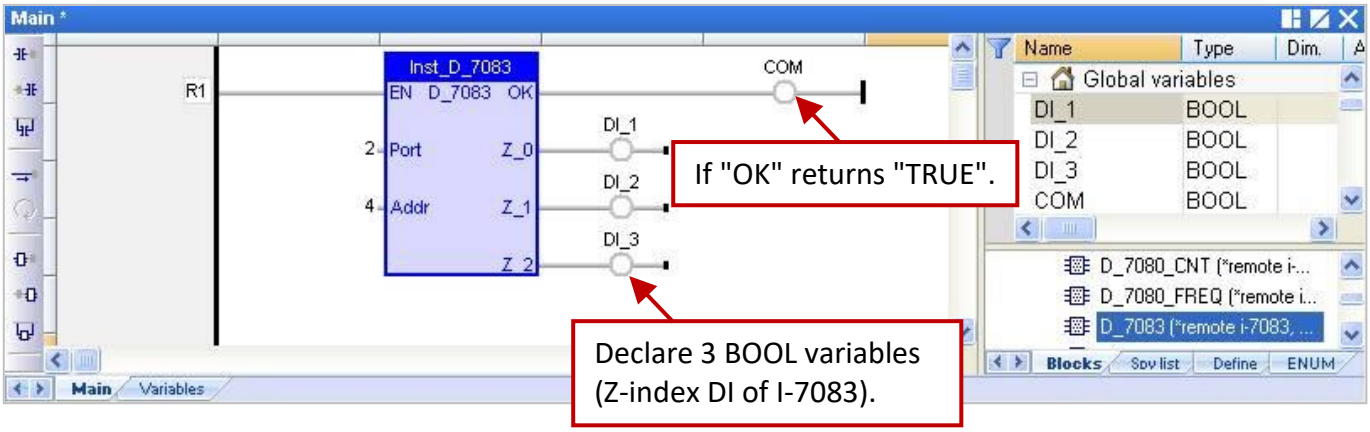
8.2.3 “D_7083” Function Block

“D_7083”: Used to connect a remote I-7083D or I-7083BD DCON module.

Note:

1. Make sure that the I/O module has been configured by using DCON Utility Pro (refer to Chapter 8). Set the data format of the AI module to "**2’s complement**", or Win-GRAF cannot read data correctly.
2. Make sure that "DCON" has been added in the "I/O boards" window (refer to Section 8.1).
3. Download the demo program (**demo_d_7083.zip**) on the website. Click the menu commands **File - Add Existing Project - From Zip...** in the Workbench to open the project.
4. Only when the communication status of the module is normal (TRUE), the received DI data is correct.
5. To get the Encoder value from the I-7083D/BD module, using the **"D_7083"** function block and the **“Counter_xxxx”** function (refer to Section 8.3) to operate the Encoder function.

Example: Connect the I-7083 (Addr. = 4) via COM2 on PAC. It has 3 DI channels



Input Parameters	Data Type	Description
EN	BOOL	TRUE: Enabled; FALSE: Disabled.
Port	DINT	COM port number (can be 1 to 37, depends on PAC). Note that specify a constant value, not a variable value.
Addr	DINT	The Net-ID address of the module can be 1 to 255. Note that specify a constant value, not a variable value.

Output Parameters	Data Type	Description
OK	BOOL	TRUE: Communication is Ok. FALSE: Communication failed.
Z_0 to Z_2	BOOL	3-axis Z-index DI value.

8.2.4 “D_87084_FREQ” Function Block

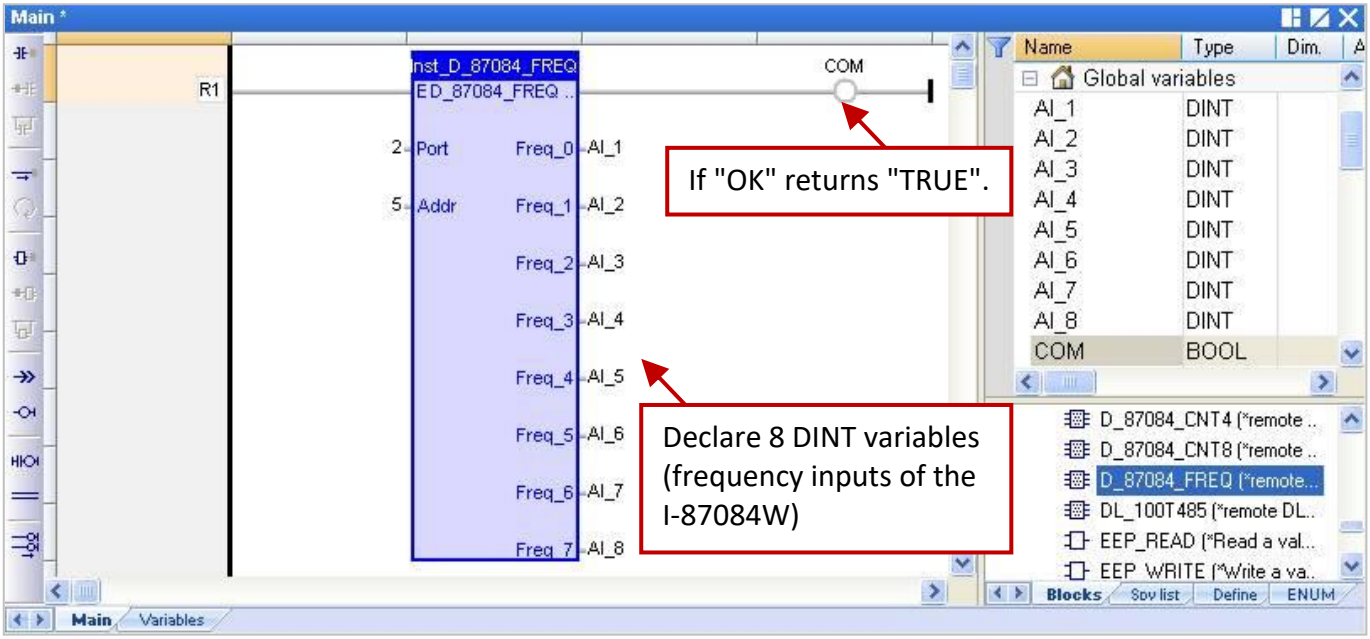
"D_87084_freq": Used to connect a remote I-87084W in an RS-485 I/O expansion unit (e.g., RU-87P4/8 or I-87K4/5/8/9) to measure frequency values.

Note:

1. Make sure that the I/O module has been configured by using DCON Utility Pro (refer to Chapter 8). Set the frequency format of I-87084W to "**Hex format**", or the function will not work.
2. Make sure that "DCON" has been added in the "I/O boards" window (refer to Section 8.1).
3. Download the demo program (**demo_d_87084_fr.zip**) on the website. Click the menu commands **File - Add Existing Project - From Zip...** in the Workbench to open the project.
4. Only when the communication status of the module is normal (TRUE), the received AI data is correct.

Example:

Connect the I-87084W (Addr. = 5) via COM2 on PAC to measure frequency values of eight channels.



Input Parameters	Data Type	Description
EN	BOOL	TRUE: Enabled; FALSE: Disabled.
Port	DINT	COM port number (can be 1 to 37, depends on PAC). Note that specify a constant value, not a variable value.
Addr	DINT	The Net-ID address of the module can be 1 to 255. Note that specify a constant value, not a variable value.

Output Parameters	Data Type	Description
OK	BOOL	TRUE: Communication is Ok. FALSE: Communication failed.
Freq_0 to Freq_7	DINT	8-Ch frequency value, unit is Hz.

8.2.5 “D_87084_CNT4” Function Block

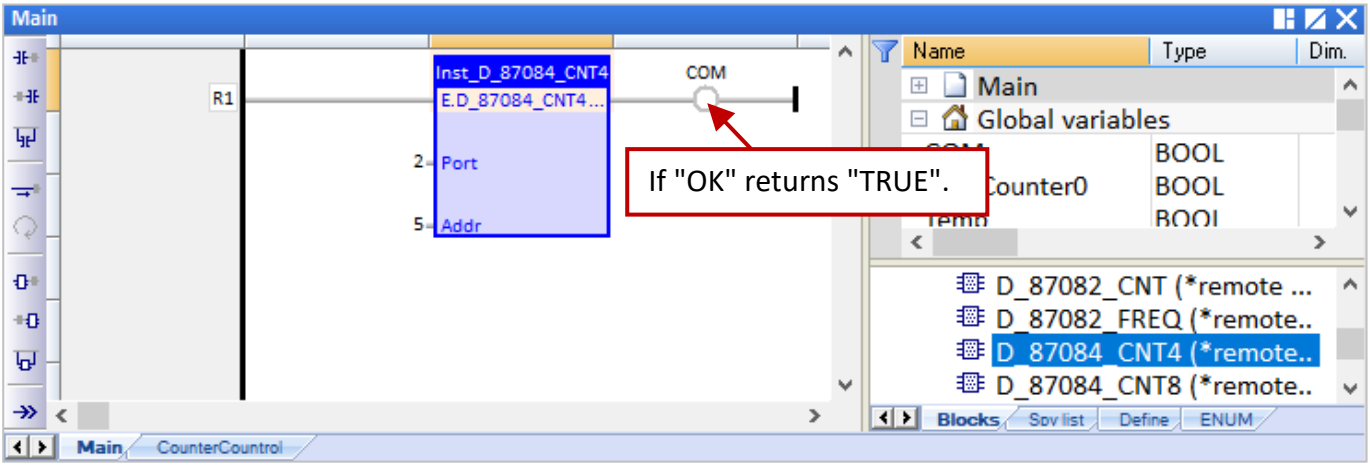
"D_87084_CNT4": Used to connect a remote I-87084W in an RS-485 I/O expansion unit (e.g., RU-87P4/8 or I-87K4/5/8/9) to measure the counter value of four channels.

Note:

1. Make sure that the I/O module has been configured by using DCON Utility Pro (refer to Chapter 8). Set the data format of I-87084W to "**Hex format**", or the function will not work.
2. Make sure that "DCON" has been added in the "I/O boards" window (refer to Section 8.1).
3. Download the demo program (**demo_d_87084_c4.zip**) on the website. Click the menu commands **File - Add Existing Project - From Zip...** in the Workbench to open the project.
4. Only when the communication status of the module is normal (TRUE), the count value is correct.
5. To get the counter value from the I-87084W module, using the **"D_87084_CNT4"** function block and the **“Counter_xxxx”** function (refer to Section 8.3) to operate the Counter function.

Example:

Connect the I-87084W (Addr. = 5) via COM2 on PAC to measure the counter value of four channels.



Input Parameters	Data Type	Description
EN	BOOL	TRUE: Enabled; FALSE: Disabled.
Port	DINT	COM port number (can be 1 to 37, depends on PAC). Note that specify a constant value, not a variable value.
Addr	DINT	The Net-ID address of the module can be 1 to 255. Note that specify a constant value, not a variable value.

Output Parameters	Data Type	Description
OK	BOOL	TRUE: Communication is Ok. FALSE: Communication failed.

8.2.6 “D_87084_CNT8” Function Block

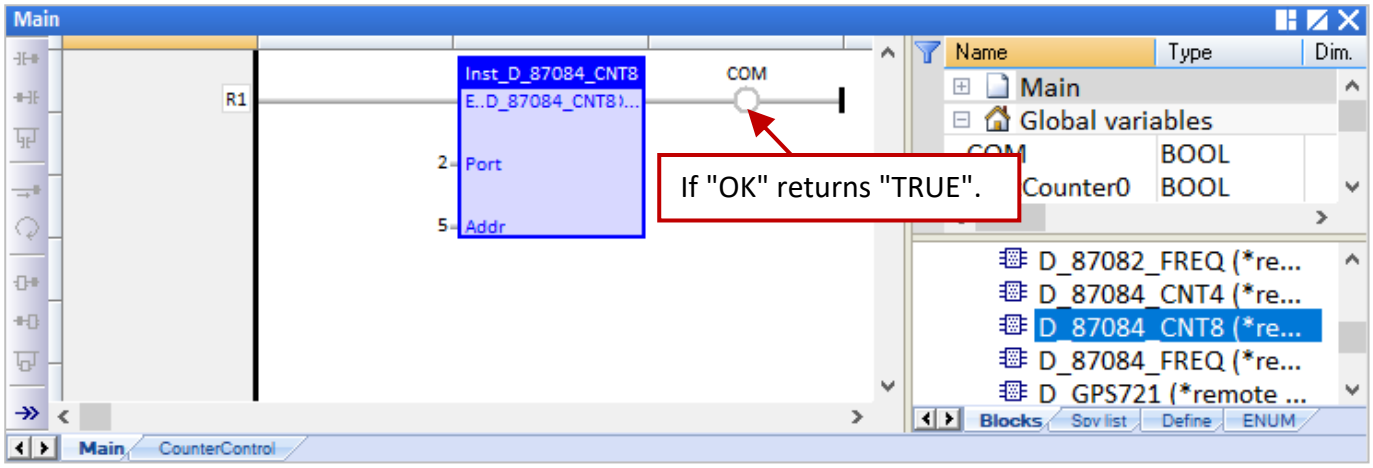
"D_87084_CNT8": Used to connect a remote I-87084W in an RS-485 I/O expansion unit (e.g., RU-87P4/8 or I-87K4/5/8/9) to measure the counter value of eight channels.

Note:

1. Make sure that the I/O module has been configured by using DCON Utility Pro (refer to Chapter 8). Set the data format of I-87084W to "**Hex format**", or the function will not work.
2. Make sure that "DCON" has been added in the "I/O boards" window (refer to Section 8.1).
3. Download the demo program (**demo_d_87084_c8.zip**) on the website. Click the menu commands **File - Add Existing Project - From Zip...** in the Workbench to open the project.
4. Only when the communication status of the module is normal (TRUE), the count value is correct.
5. To get the counter value from the I-87084W module, using the **"D_87084_CNT8"** function block and the **“Counter_xxxx”** function (refer to Section 8.3) to operate the Counter function.

Example:

Connect the I-87084W (Addr. = 5) via COM2 on PAC to measure the counter value of eight channels.



Input Parameters	Data Type	Description
EN	BOOL	TRUE: Enabled; FALSE: Disabled.
Port	DINT	COM port number (can be 1 to 37, depends on PAC). Note that specify a constant value, not a variable value.
Addr	DINT	The Net-ID address of the module can be 1 to 255. Note that specify a constant value, not a variable value.

Output Parameters	Data Type	Description
OK	BOOL	TRUE: Communication is Ok. FALSE: Communication failed.

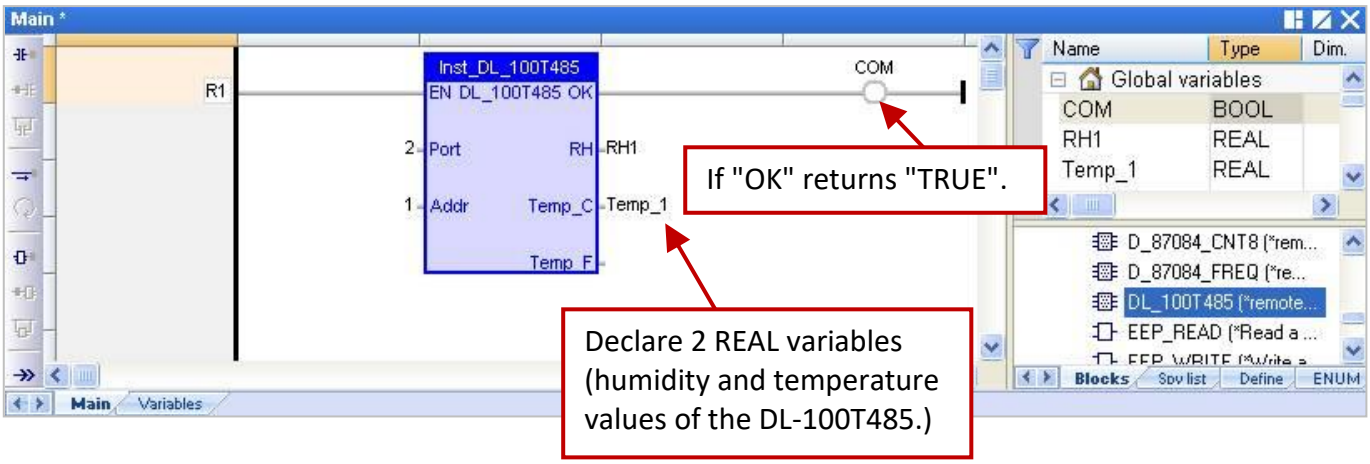
8.2.7 “DL_100T485” Function Block

"DL_100T485": Used to connect a remote DL-100T485 module to get humidity and temperature value.

Note:

1. Make sure that the I/O module has been configured by using DL Utility. DL-100T485’s default value: Address is "1", Baudrate is "9600", and Checksum is "Disable". (Download DL-100T485 user manual)
2. Make sure that "DCON" has been added in the "I/O boards" window (refer to Section 8.1).
3. Download the demo program (**demo_dl_100T485.zip**) on the website. Click the menu commands **File - Add Existing Project - From Zip...** in the Workbench to open the project.
4. Only when the communication status of the module is normal (TRUE), the AI value is correct.

Example:
Connect the DL_100T485 (Addr. = 1) via COM2 on PAC to get humidity and temperature value



Input Parameters	Data Type	Description
EN	BOOL	TRUE: Enabled; FALSE: Disabled.
Port	DINT	COM port number (can be 1 to 37, depends on PAC). Note that specify a constant value, not a variable value.
Addr	DINT	The Net-ID address of the module can be 1 to 255. Note that specify a constant value, not a variable value.

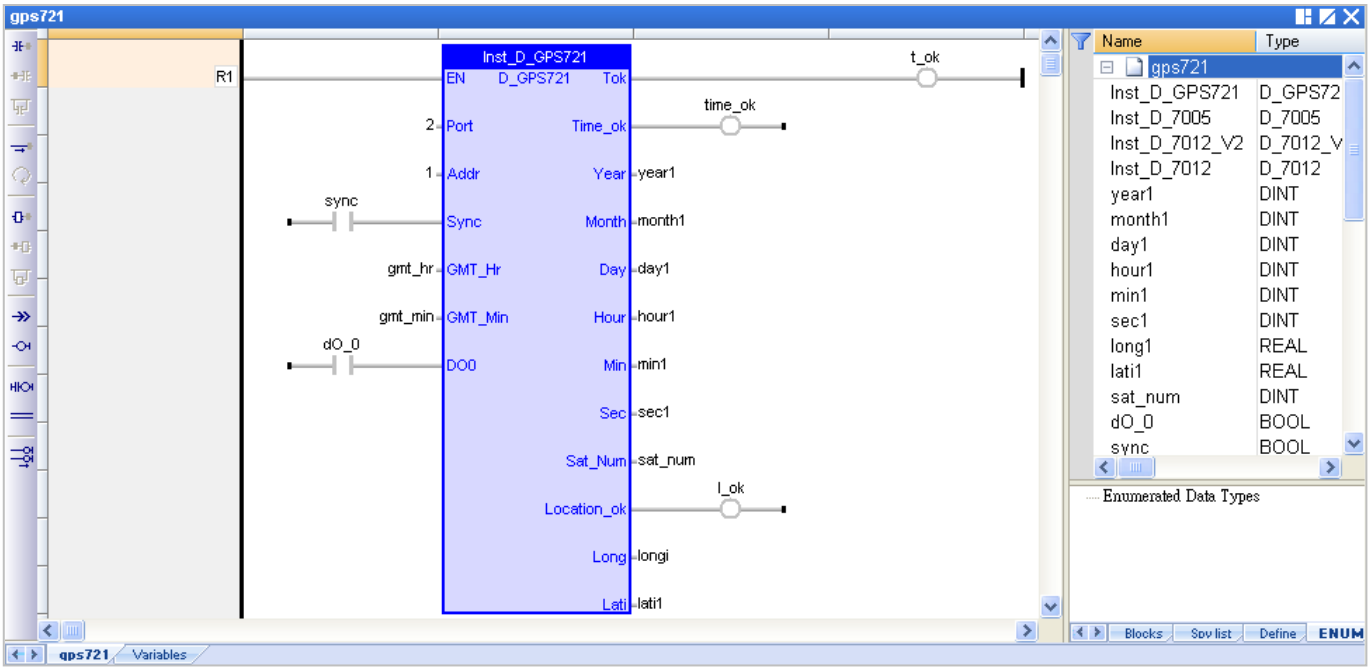
Output Parameters	Data Type	Description
OK	BOOL	TRUE: Communication is Ok. FALSE: Communication failed.
RH	REAL	"Relative humidity"; the unit is 1%. E.g., the value of 45.7 stands for 245.7 %.
Temp_C	REAL	The temperature C in degrees Celsius. E.g., the value of 25.7 stands for 25.7 °C.
Temp_F	REAL	The temperature F in degrees Fahrenheit. E.g., the value of 78.26 stands for 78.26 °F.

8.2.8 “D_GPS721” Function Block

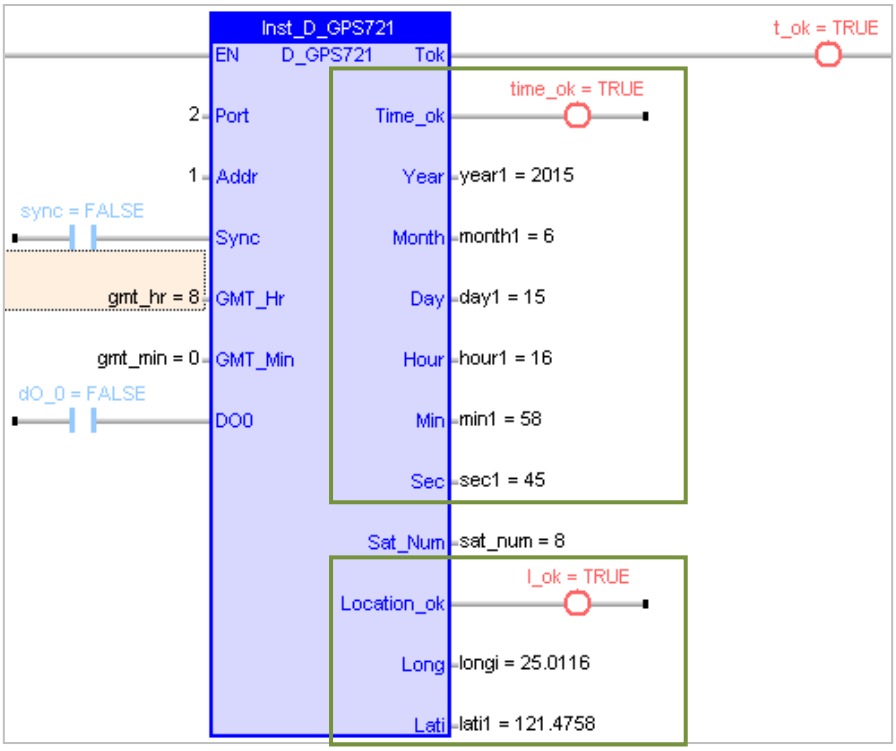
Using the “D_GPS721” function to link one “GPS-721” remote GPS receiver module (includes one DO and one PPS outputs).

Note:

1. Only one GPS-721 module can be used for one PAC.
2. Make sure that the module has been configured by using DCON Utility Pro (refer to Chapter 8). GPS-721’s default value: Address is "1", Baudrate is "9600", and Checksum is "Disable".
3. Make sure that "DCON" has been added in the "I/O boards" window (refer to Section 8.1).
4. Download the demo program (**dmeo_gps721.zip**) on the website. Click the menu commands **File - Add Existing Project - From Zip...** in the Workbench to open the project.



Input Parameters	Data Type	Description
EN	BOOL	TRUE: Enabled; FALSE: Disabled.
Port	DINT	COM port number (can be 1 to 37, depends on PAC). Note that specify a constant value, not a variable value.
Addr	DINT	The Net-ID address of the module can be 1 to 255. Note that specify a constant value, not a variable value.
Sync	BOOL	Time synchronization, TRUE: Enabled; FALSE: Disabled. The PAC time will be automatically corrected if there is a 5-second (or more) time difference between a PAC and a GPS-721.
GMT_Hr & GMT_Min	DINT	Time Zone. for example, Beijing and Taipei are **GMT+8** (**GMT_Hr = 8**, GMT_Min = 0), USA is **GMT-6** (**GMT_Hr = -6**, GMT_Min = 0), and New Delhi is **GMT+5.5** (**GMT_Hr = 5, GMT_Min = 30**).
DO0	BOOL	Digital output channel of the GPS-721 module



Output Parameters	Data Type	Description
Tok	BOOL	TRUE: Communication of GPS-721 is Ok; FALSE: Communication failed, the return values below are invalid.
Time_ok	BOOL	TRUE: The value of Year, Month, Day, Hour, Min, and Sec are valid. FALSE: The value of Year, Month, Day, Hour, Min, and Sec are invalid.
Year、Month **Day、Hour** **Min、Sec**	DINT	Year (2009 to ...), Month (1 to 12), Day (1 to 31), Hour (0 to 23), Minute (0 to 59), and (0 to 59). **Note:** "Time_ok" will be automatically set to "FALSE" from 23:59:00 to 00:00:59 (2 minutes) every day and time synchronization will not be performed.
Sat_Num	DINT	The number of satellites in use. 0: No satellite is found. Or, 1 to 9 satellites are in use.
Location_ok	BOOL	**Note:** Only when "Location_ok" is "TRUE", the value of Long and Lati is valid. TRUE: GPS-721 has got the latitude/longitude of the current position. FALSE: the value of Long and Lati is valid.
Long	REAL	Longitude (Positive: The East, Negative: The West). E.g., the value of 25.0121 stands for 25.0121° E.
Lati	REAL	Latitude (Positive: The North, Negative: The South). E.g., the value of "121.4576" stands for 121.4576° N.

8.3 Using the Count Function for I-87082W, I-87084W, I-7083, and I-7080 Modules

This section lists the way to perform the Counter/Encoder function by using the "COUNTER_START", "COUNTER_STOP", "COUNTER _GET", "COUNTER_STATE", and “COUNTER_RESET” functions in the LD or ST program.

Note:

- Adding the I/O function block before using the "COUNTER_XXX" function, or it will not work.
- The I-87K I/O module can be plugged into an RS-485 I/Oexpansion unit (e.g., RU-87P4/8).

Model	Function Block	Encoder	Counter	Frequency
I-7080	"D_7080"		✓	✓
I-7083	"D_7083"	✓ **(Section 8.2.3)**	✓	
I-87082W	"D_87082"		✓	✓
I-87084W	"D_87084_cnt4" or "D_87084_cnt8"	✓	✓ **(Section 8.2.5)**	✓ **(Section 8.2.4)**

8.3.1 COUNTER_START

Description: Start the counting of a counter or encoder.

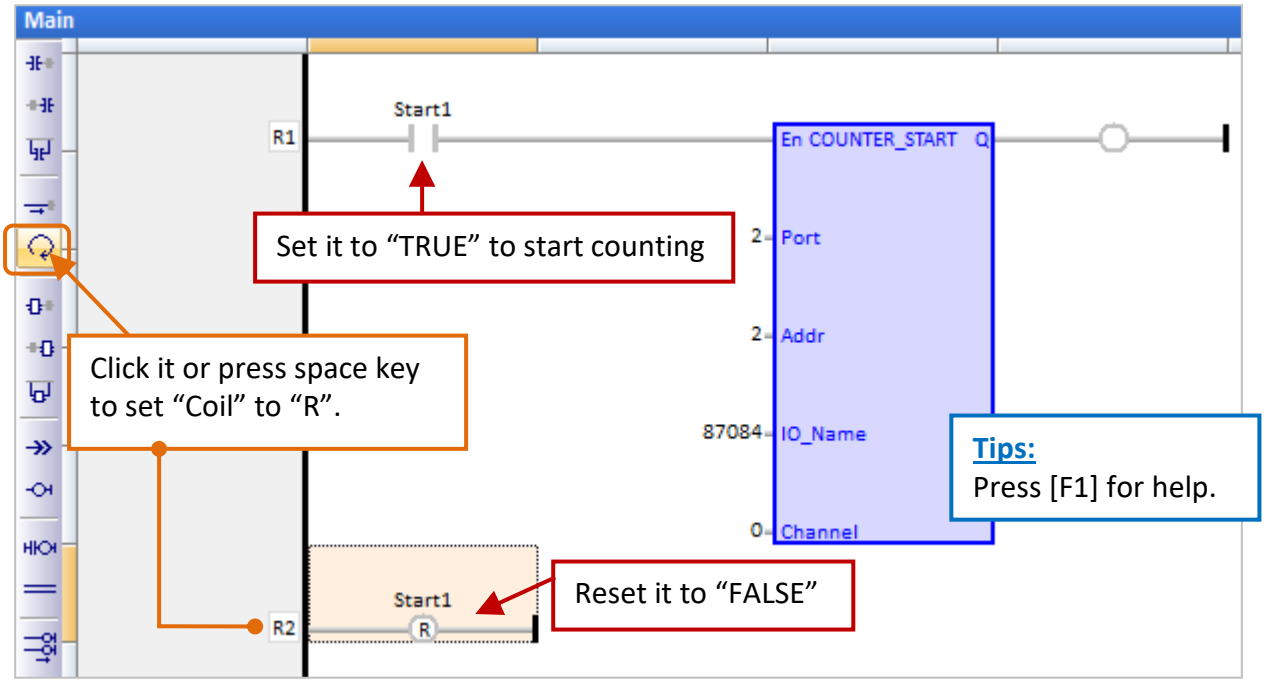
ST program:

```
IF   Start1 = TRUE   THEN
  Start1  :=  FALSE ;
  TMP_BOOL  :=  Counter_Start (Port, Addr, IO_Name, Channel) ;
END_IF ;
```

Note: "Start1" and "TMP_BOOL" Boolean variables can be declared in the Variables area.

LD program:

"Start1" (BOOL): set it to "TRUE" to start counting and then reset the "Start1" to "FALSE".



Input Parameters	Data Type	Description
Port	DINT	COM port number (can be 1 to 37, depends on PAC).
Addr	DINT	The Net-ID address of the module can be 1 to 255.
IO_Name	DINT	The name of relative Counter/Encoder module. it can be set to "87084", "87082", "7083" and "7080".
Channel	DINT	The channel number of the Counter/Encoder module. it can be set to "0", "1", and so on.

Output Parameters	Data Type	Description
Q	BOOL	TRUE: Ok. FALSE: Error.

8.3.2 COUNTER_STOP

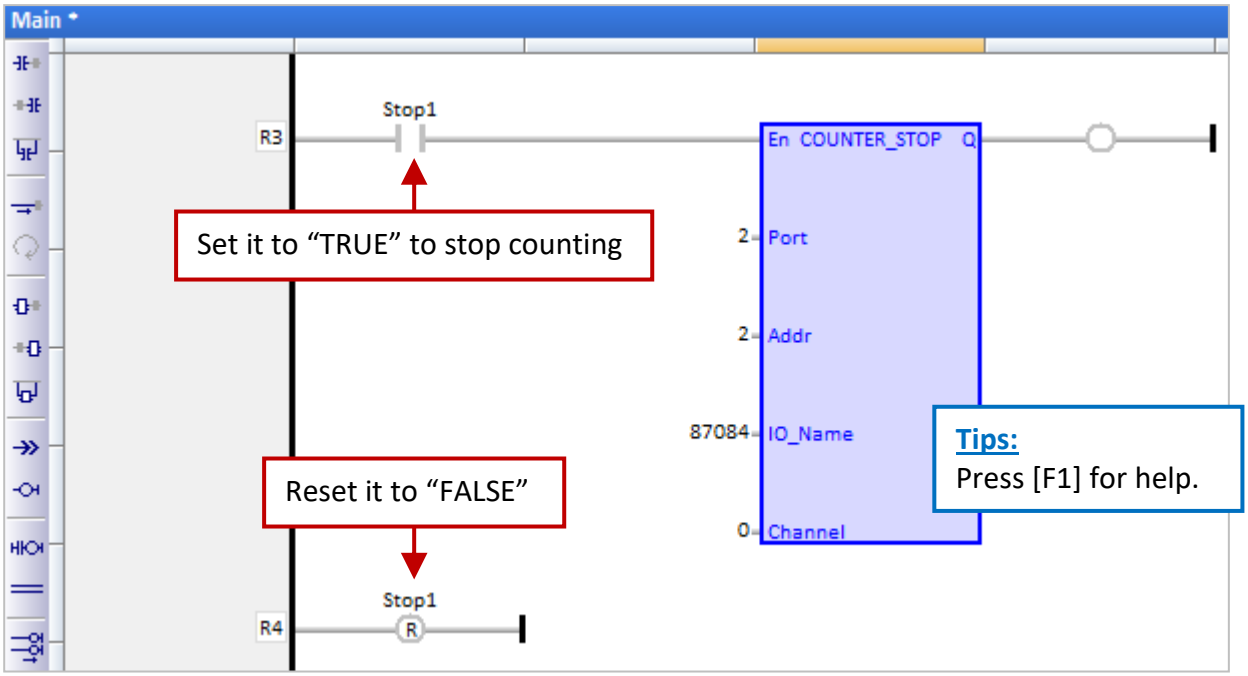
Description: Stop the counting of a counter or encoder.

ST program: "Start1" and "TMP_BOOL" boolean variables can be declared in the Variables area.

```
IF   Stop1 = TRUE   THEN
  Stop1  :=  FALSE ;
  TMP_BOOL  :=  Counter_Stop (Port, Addr, IO_Name, Channel) ;
END_IF ;
```

LD program:

"Stop1" (BOOL): set it to "TRUE" to stop counting and then reset the "Stop1" to "FALSE".



Input Parameters	Data Type	Description
Port	DINT	COM port number (can be 1 to 37, depends on PAC).
Addr	DINT	The Net-ID address of the module can be 1 to 255.
IO_Name	DINT	The name of relative Counter/Encoder module. it can be set to "87084", "87082", "7083" and "7080".
Channel	DINT	The channel number of the Counter/Encoder module. it can be set to "0", "1", and so on.

Output Parameters	Data Type	Description
Q	BOOL	TRUE: Ok. FALSE: Error.

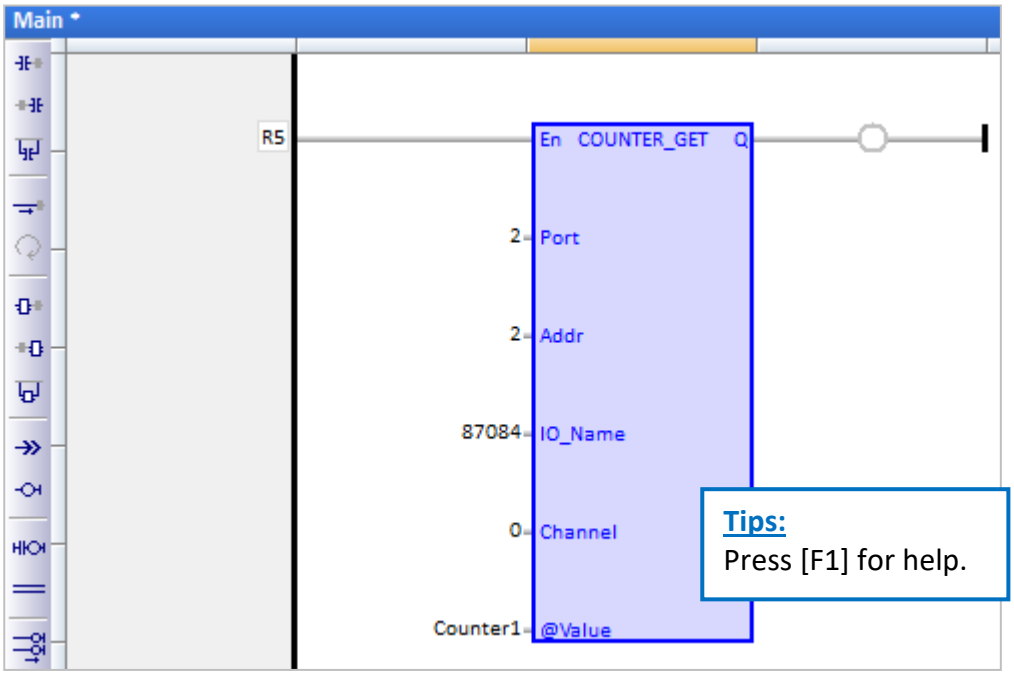
8.3.3 COUNTER_GET

Description: Get a counter or an encoder value.

ST program: "TMP_BOOL" (BOOL) and "Counter1“ (DINT) can be declared in the Variables area.

```
TMP_BOOL := Counter_Get (Port, Addr, IO_Name, Channel, Counter1) ;
```

LD program:



Input Parameters	Data Type	Description
Port	DINT	COM port number (can be 1 to 37, depends on PAC).
Addr	DINT	The Net-ID address of the module can be 1 to 255.
IO_Name	DINT	The name of relative Counter/Encoder module. it can be set to "87084", "87082", "7083" and "7080".
Channel	DINT	The channel number of the Counter/Encoder module. it can be set to "0", "1", and so on.
@Value	“DINT”, “UDINT”, “DWORD”, or “LINT”	Return the value of a counter or an encoder. Refer to Appendix A for the range of data.

Output Parameters	Data Type	Description
Q	BOOL	TRUE: Ok. FALSE: Error.

8.3.4 COUNTER_STATE

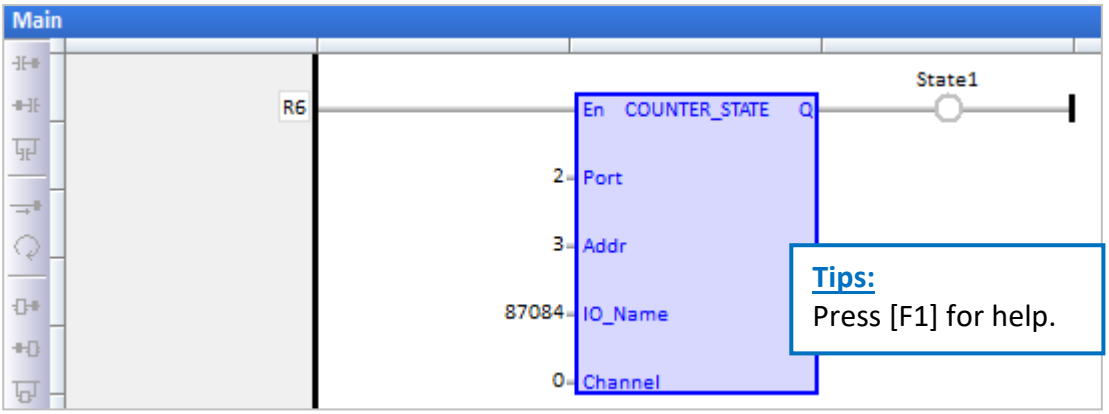
Description: Get the counting state of a counter or an encoder.

ST program: "TMP_BOOL" (BOOL) can be declared in the Variables area.

```
TMP_BOOL := Counter_State (Port, Addr, IO_Name, Channel) ;
```

LD program:

“State1” (BOOL): Display the status of counting.



Input Parameters	Data Type	Description
Port	DINT	COM port number (can be 1 to 37, depends on PAC).
Addr	DINT	The Net-ID address of the module can be 1 to 255.
IO_Name	DINT	The name of the Counter/Encoder module. it can be set to "87084", "87082", "7083" and "7080".
Channel	DINT	The channel number of the Counter/Encoder module. it can be set to "0", "1", and so on.

Output Parameters	Data Type	Description
Q	BOOL	TRUE: Ok. FALSE: Error.

8.3.5 COUNTER_RESET

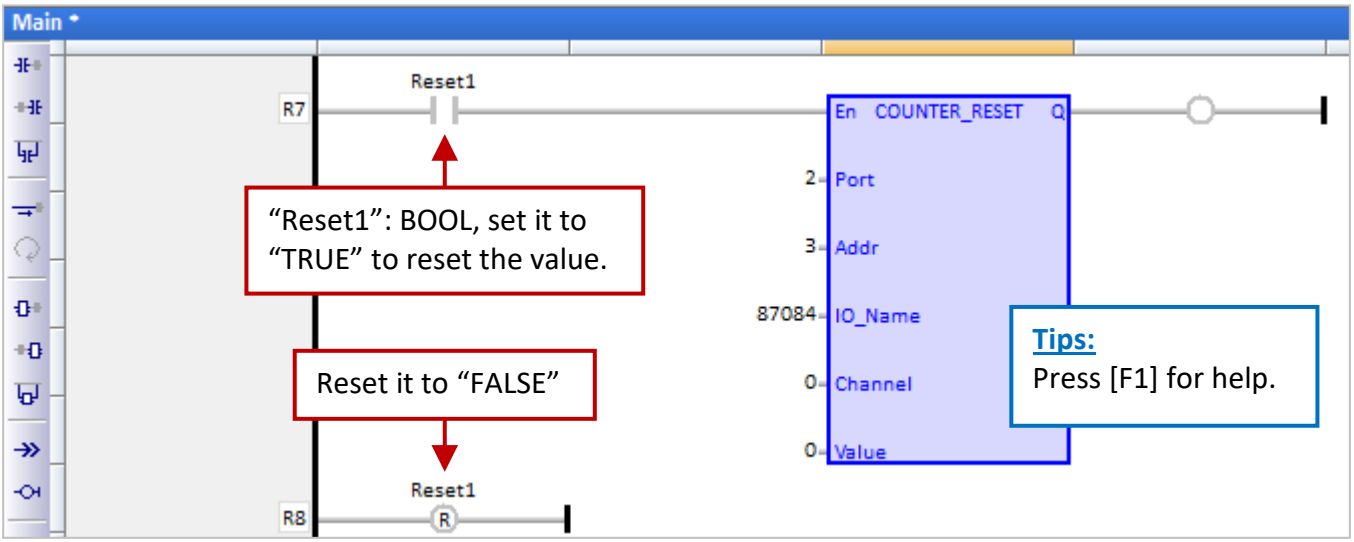
Description: Reset a counter or encoder to a specified value.

ST program: "Reset1“ (DINT) and "TMP_BOOL" (BOOL) can be declared in the Variables area.

```
IF   Reset1 = TRUE   THEN
  Reset1  :=  FALSE ;
  TMP_BOOL  :=  Counter_Reset (Port, Addr, IO_Name, Channel, Value) ;
END_IF ;
```

LD program:

“Reset1” (BOOL): set it to “TRUE” to reset the value and then reset the “Reset1” to “FALSE”.



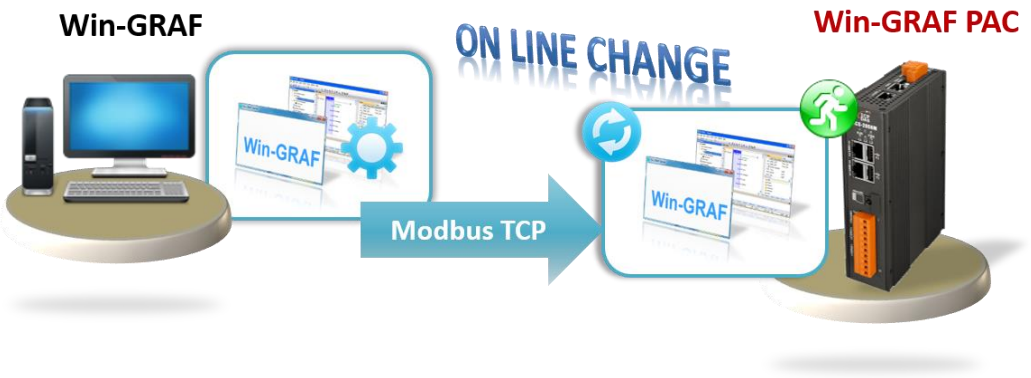
Input Parameters	Data Type	Description
Port	DINT	COM port number (can be 1 to 37, depends on PAC).
Addr	DINT	The Net-ID address of the module can be 1 to 255.
IO_Name	DINT	The name of relative Counter/Encoder module. it can be set to "87084", "87082", "7083" and "7080".
Channel	DINT	The channel number of the Counter/Encoder module. it can be set to "0", "1", and so on.
Value	“DINT”, “UDINT”, “DWORD”, or “LINT”	Specify a new value for the Counter/Encoder.

Output Parameters	Data Type	Description
Q	BOOL	TRUE: Ok. FALSE: Error.

Chapter 9 On-Line Change

The "On-Line Change" function allows Win-GRAF PAC to update the project with a little modification while it is running. The modified project must be the same name as the one running on the PAC.

This feature is mainly used in emergencies, in some cases, the system must be running for a 24-hour a day and cannot stop working for updating the project. Except for that situation, it is not recommended to do the online modification. The best and safest way is to stop running the project and then download the modified one to the PAC.

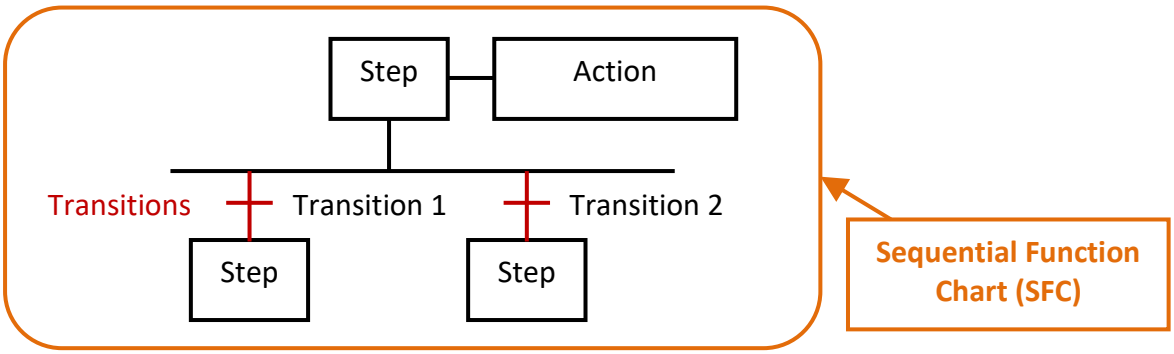


9.1 Limitations of "On-Line Change"

Please notice the important limitations before enabling the "On-Line Change"!

When On-Line Change is enabled, you can perform on the fly the following kinds of changes:

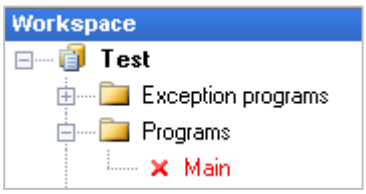
- Change the code of a program.
- Change the condition of an SFC transition or the actions of an SFC step.



- Create, rename, or delete global and local variables.
- Create, rename, or delete global and local function block instances.

The following kinds of changes are not allowed:

- Create, delete, or rename a program.
  (The warning message will be displayed when a program is deleted.)

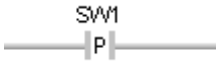
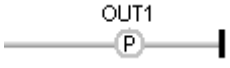
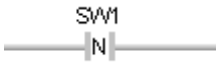
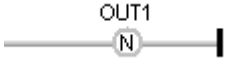


- Change SFC charts.
- Change the local parameters and variables of a UDFB.
- Change the type or dimension (or string length) of a variable or function block instance.
- Change the set of I/O boards.
- Change the definition of RETAIN variables.

Besides, the following programming features that are not safe during a change should not be used:

- Pulse (P or N) contacts and coils (edge detection).

  ✍ Instead, you must use declared instances of R_TRIG and F_TRIG function blocks.

Rising Pulse Detection		
P (False to True)	**Not Safe**	**Safe**
		
		
Decreased Pulse Detection		
N (True to False)		
		

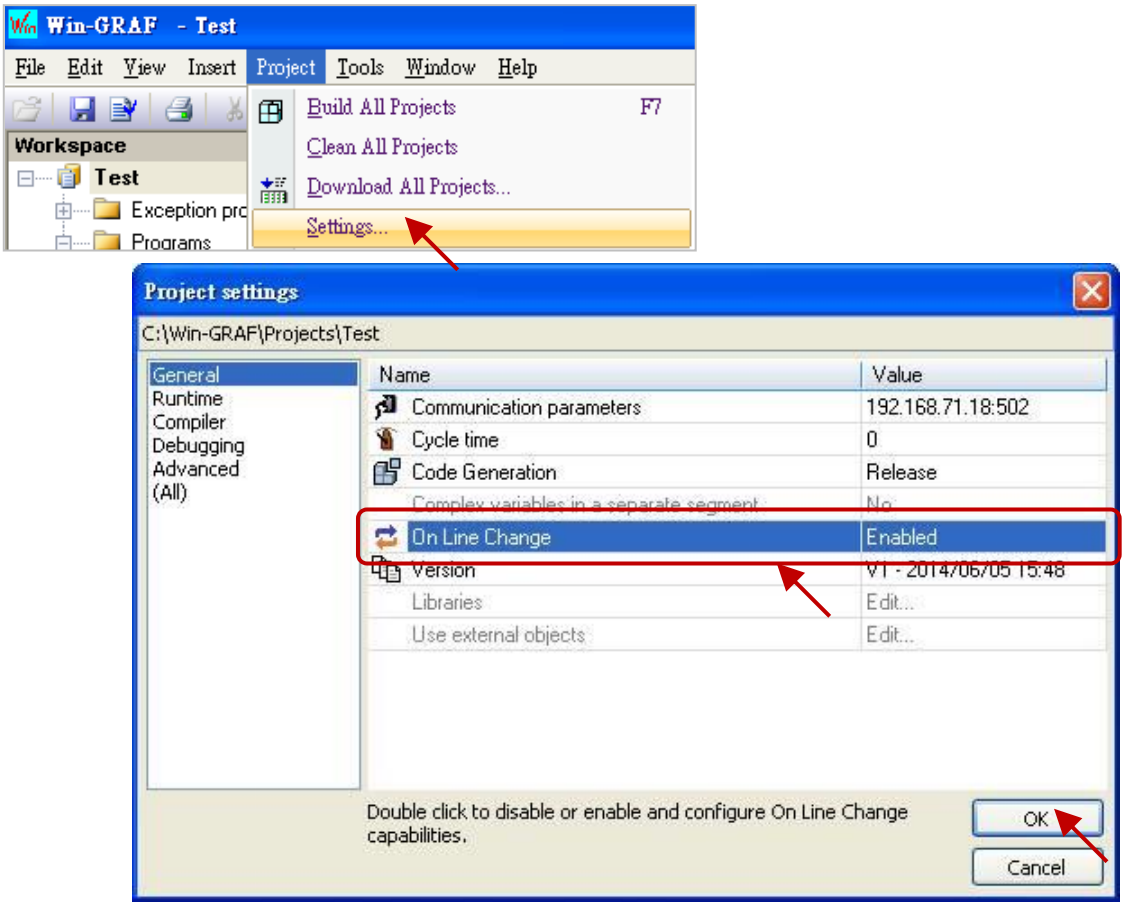
- Loops in FBD with no declared variable linked.

  ✍ You need to explicitly insert a variable in the loop.

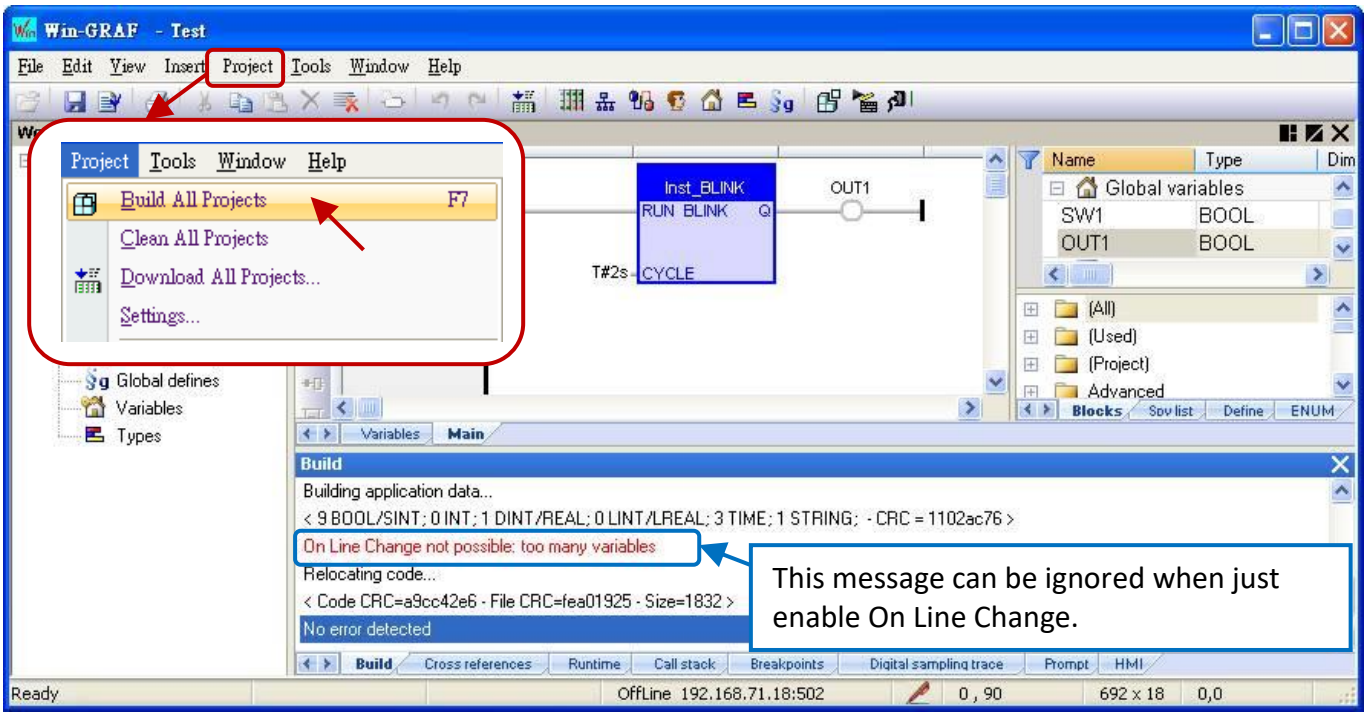
9.2 Using "On-Line Change"

Enable the "On-Line Change" Function:

1. Click the menu command **"Project - Settings..."** and double-click the **On-Line Change** item to set it to **Enabled**, and then click OK.



2. Click the menu command **Project - Build All Projects** to compile programs and update the number of variables displayed in the "On Line Change" window.



Allocate the number of available variables:

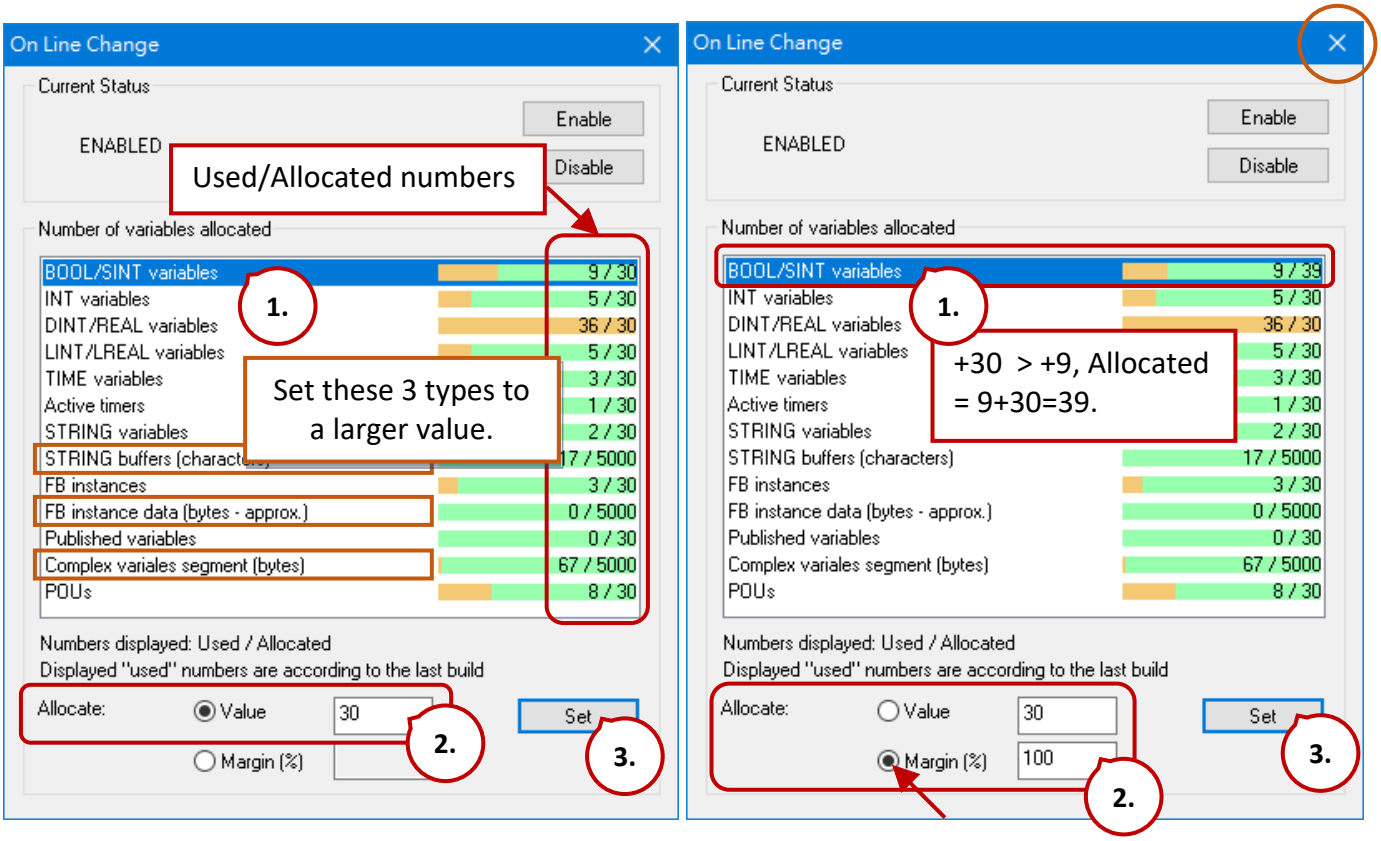
To allow the declaration of new variables or POUs (name of the sub-program or UDFB) after the Online Change function is enabled, you have to define the amount of memory to be allocated in the PAC for each type of data.

3. Follow step 1 to open the "On Line Change" window. As the figure below, click the desired Variable type and set the number of the **Value** or **Margin(%)**, and then click the **Set** button.

Allocate	Description
Value	Total amount (as the figure on the left). Click the BOOL variable and set it to 30. After clicking Set, the Allocated number will be 30.
Margin	Used + Used * n%. In this example, click the BOOL variable and set it to 100%. After clicking Set, the Allocated number will be 9+9*100% = 18.

Notice:

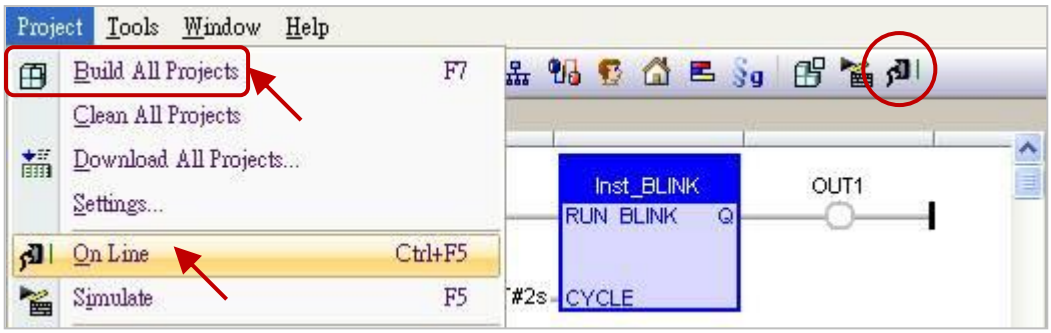
- If both “Value" and “Margin" values are set, the larger number will be added, as the figure on the right. In this case, the **Margins** setting (100%=9) is less than the **Value** setting (30). After clicking Set, the Allocated number will be 9 + 30 = 39.



- There are some "used“ numbers by default even if the project is empty.
- "STRING buffers (characters)", "FB instance data (bytes – approx.)" and "Complex variables segment (bytes)" need to set a larger number (e.g., "5000").

After completing the settings, click the "X" in the upper right corner of the window to close it.

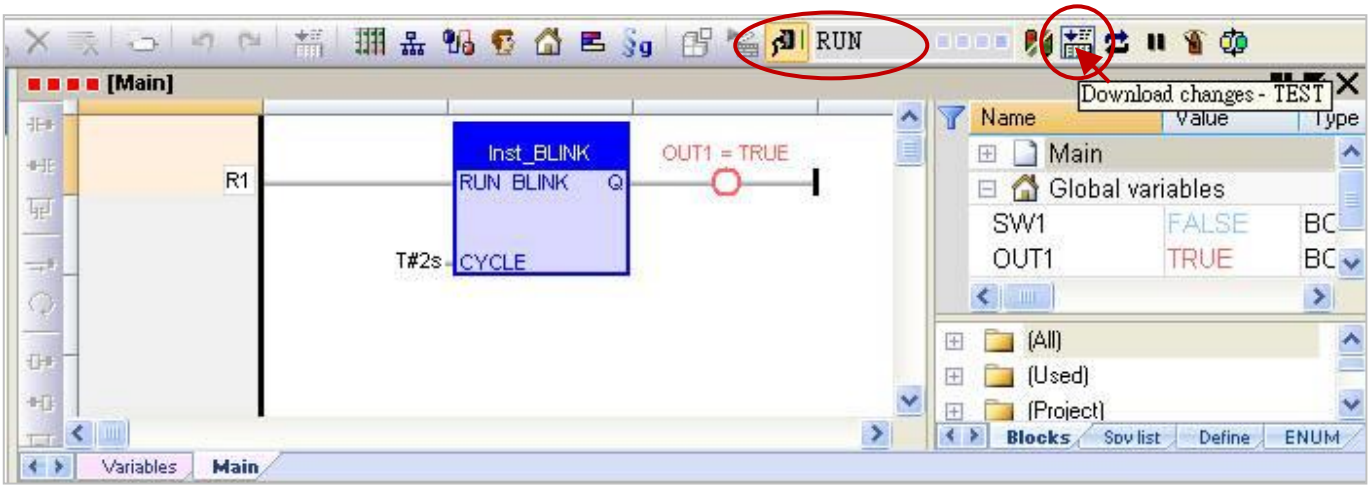
4. Click the menu command **Project** - **Build All Projects** to compile programs again. Then click “On Line” to connect to the PAC.



5. After connecting to the PAC, click the "Download changes" to download the program to the PAC.

Note:

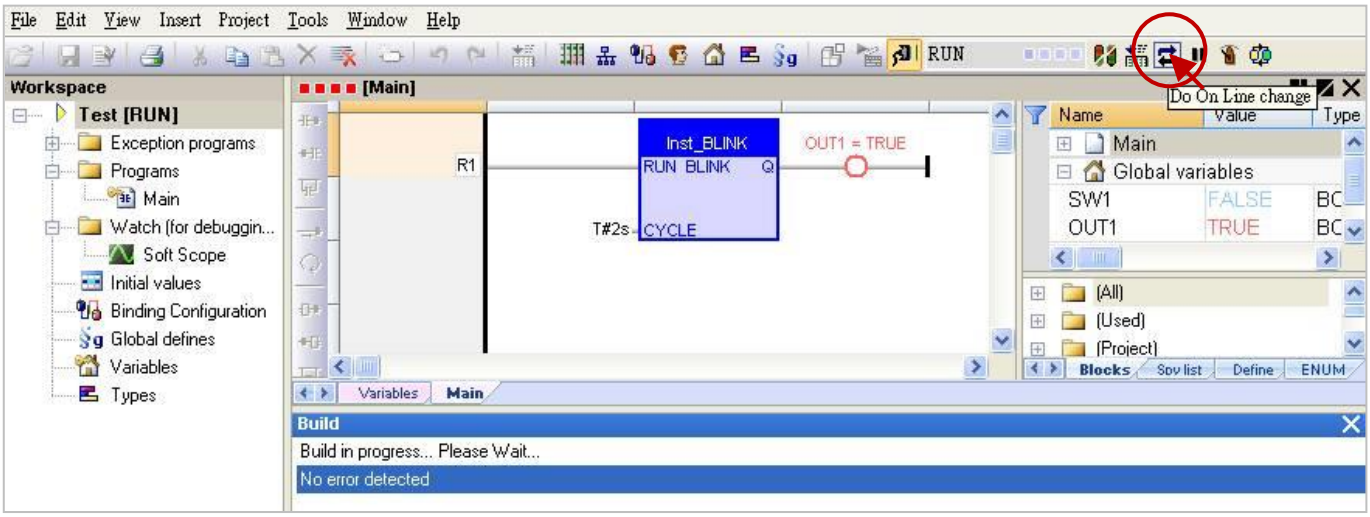
"On-Line Change" only can do the online modification for the same Win-GRAF project. If the project name is different, the user must stop running the project and download the modified one again.



6. Click the "Do On-Line Change" button to execute the program.

Note:

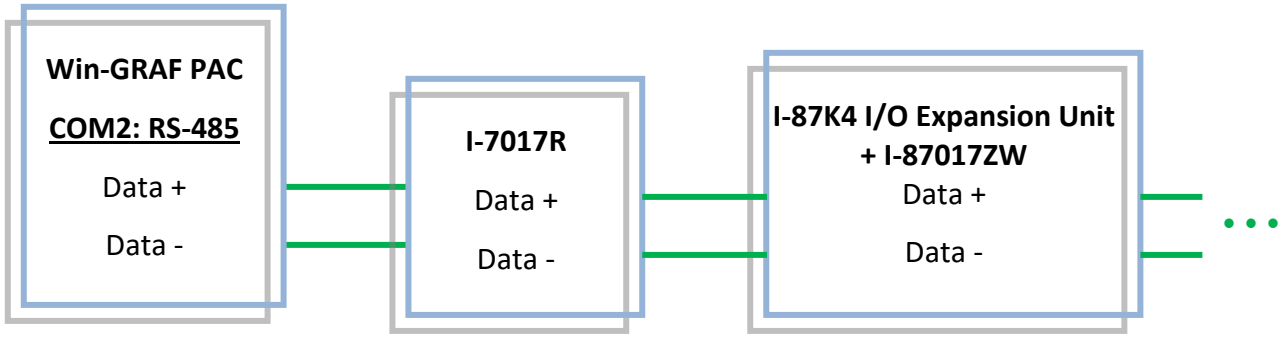
To ensure that the PAC functions properly, there are some restrictions of usage after enabling the "On-Line Change", see Section 9.1. Make sure all programs are correct before running this function.



Chapter 10 Data/Type Conversion and Using the PAC Time

10.1 AI Data Conversion (i_scale)

Refer to the following settings when connecting to the remote analog input modules, for example, connecting the I-87017ZW or I-7017R through the RS-485 port on the PAC.

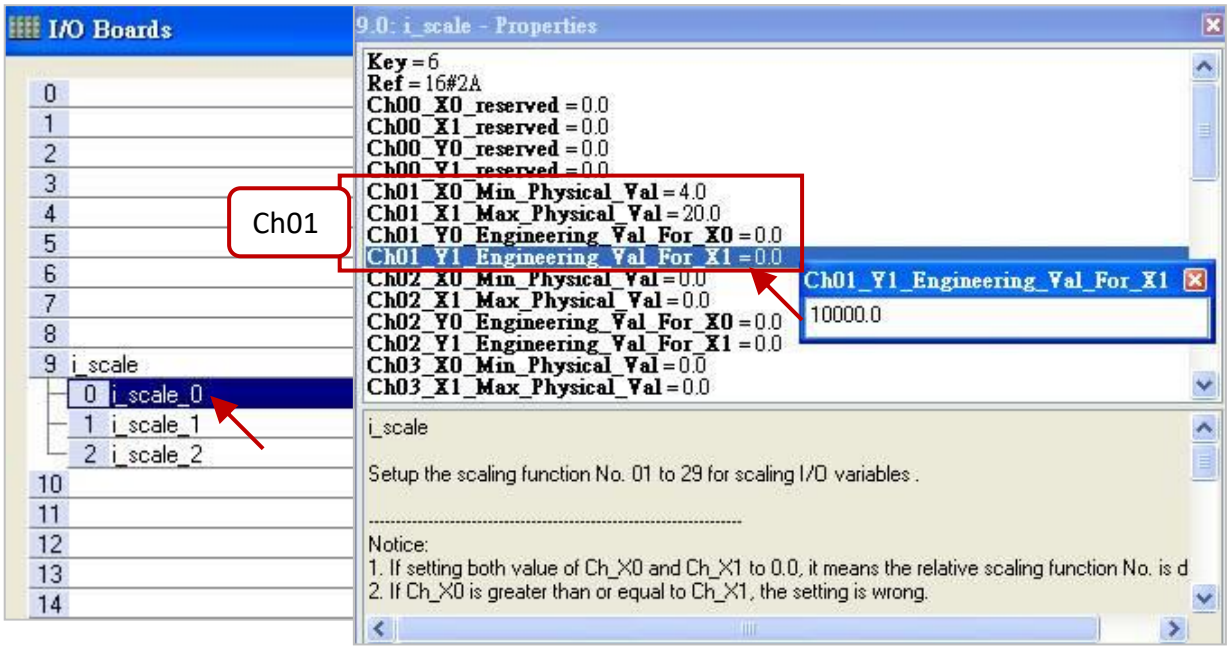


1. First, add "i_scale" in the "I/O Boards" window, and double-click on "i_scale_x" to open the Properties window (refer to Chapter4)
   Note: ONLY one "i_scale" can be used in the Win-GRAF project.

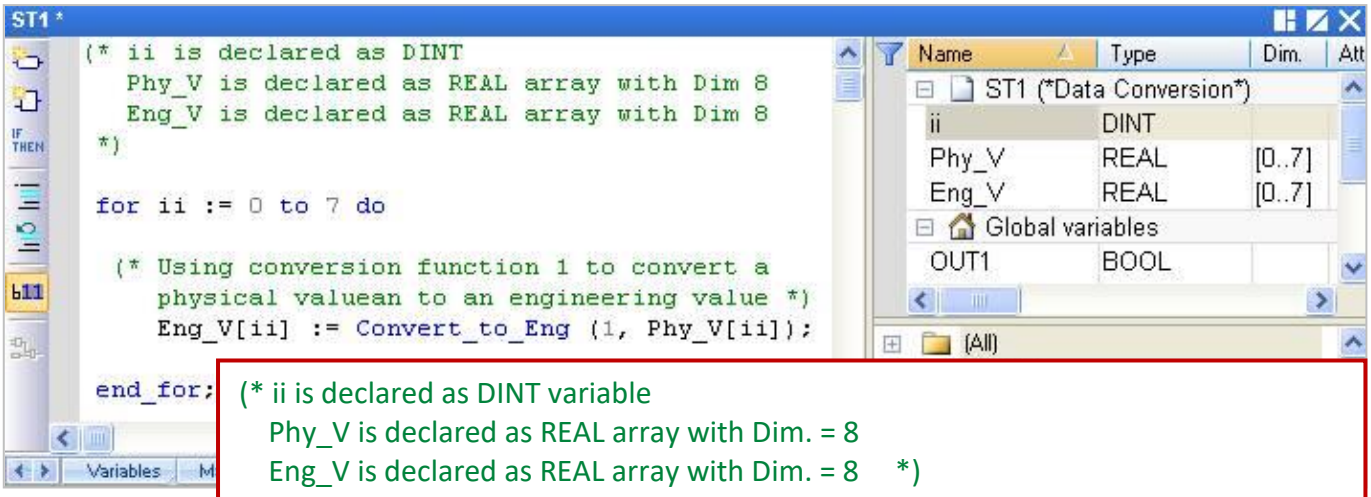
2. Set values for the desired scaling function No.

   For example, converting the current input (4-20 mA) into engineering values (0-10000) by using the function No. 01.

   Ch**01**_X0_Min_Physical_Val: "4.0"
   Ch**01**_X1_Max_Physical_Val: "20.0"
   Ch**01**_Y0_Engineering_Val_For_X0: "0.0"
   Ch**01**_Y1_Engineering_Val_For_X1: "10000.0"



3. Edit an ST Program to convert a physical value (e.g., Phy_V[0] to [7]) to an engineering value (e.g., Eng_V[0] to [7]).

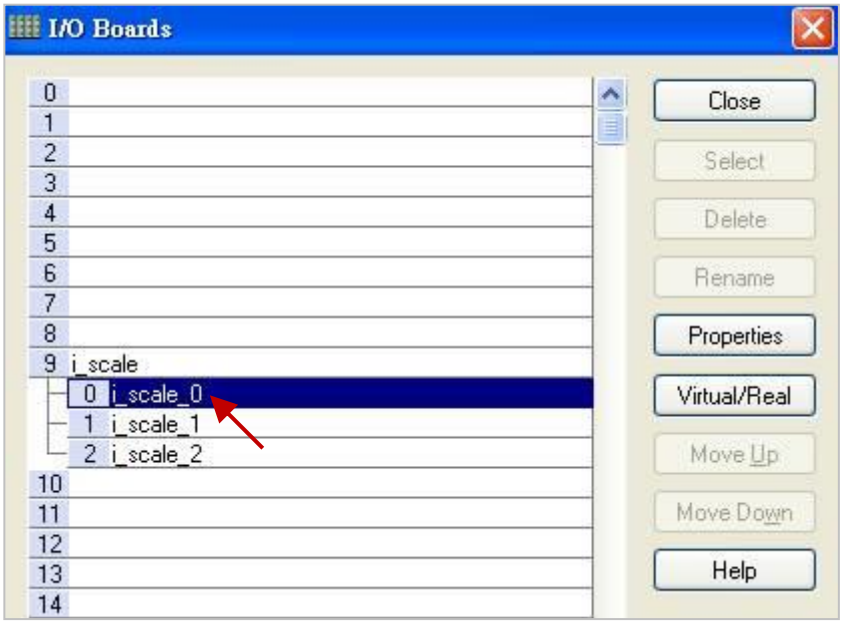



```
(* ii is declared as DINT variable
   Phy_V is declared as REAL array with Dim. = 8
   Eng_V is declared as REAL array with Dim. = 8     *)

for ii := 0 to 7 do

(* Using scaling function No.1 to convert a physical value to an engineering value *)
   Eng_V[ii] := Convert_to_Eng (1, Phy_V[ii]);

end_for;
```

10.2 AO Data Conversion (i_scale)

Refer to the following settings when connecting to the remote analog output modules, for example, connecting the I-7024 through the RS-485 port on the PAC.

1. Add "i_scale" in the "I/O Boards" window, and double-click on "i_scale_x" to open the Properties window (refer to Chapter4)
   Note: ONLY one "i_scale" can be used in the Win-GRAF project.



2. Set values for the desired scaling function No.

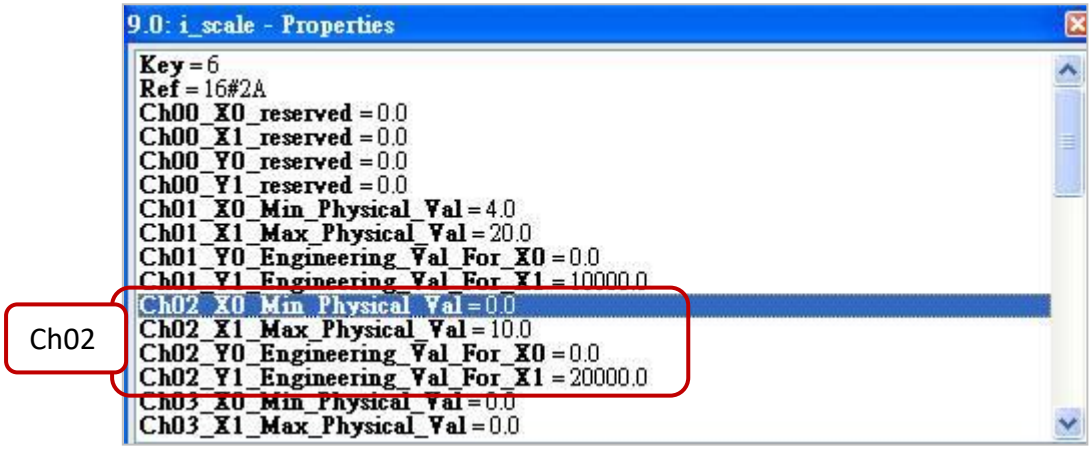
For example, converting the engineering values (0-20000) into the voltage output (0-10V) by using the function No. 02.

Ch**02**_X0_Min_Physical_Val: “0.0”

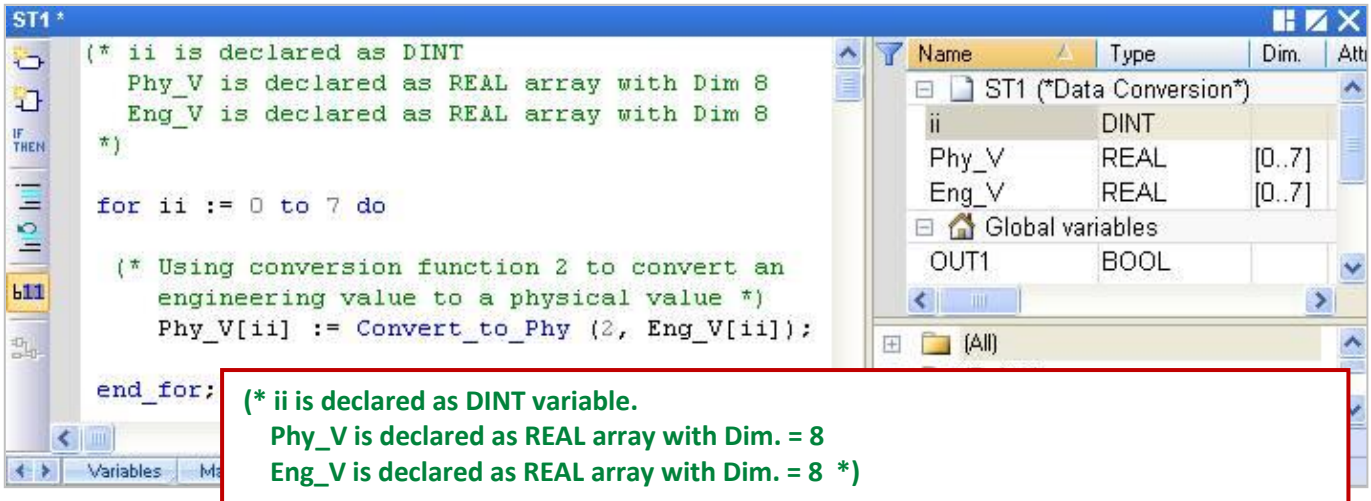
Ch**02**_X1_Max_Physical_Val: “10.0”

Ch**02**_Y0_Engineering_Val_For_X0: “0.0”

Ch**02**_Y1_Engineering_Val_For_X1: “20000.0”



3. Edit an ST Program to convert an engineering value to a physical value (e.g., Eng_V[0] to [7]).




```
(* ii is declared as DINT variable.
   Phy_V is declared as REAL array with Dim. = 8
   Eng_V is declared as REAL array with Dim. = 8  *)

for ii := 0 to 7 do

    (* Using conversion function 2 to convert an engineering value to a physical value *)

    Phy_V[ii] := Convert_to_Phy (2, Eng_V[ii]);

end_for;
```

10.3 Type Conversion Functions (ANY_TO_xxx)

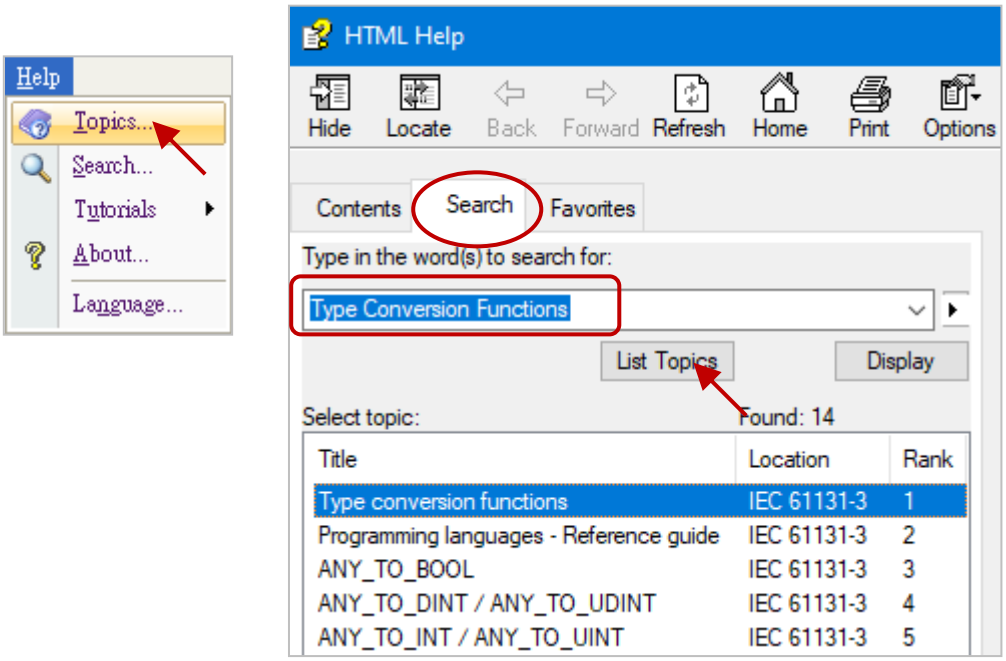
When making comparisons (e.g., > , < , = , <= , >= , <>) and performing calculations (e.g., +, - , * , /) with different types of data or variables in the program, refer to the table below to convert them into the same data type to perform properly.

Type Conversion Functions	Descriptions
ANY_TO_BOOL	Convert to Boolean
ANY_TO_SINT	Convert to Short Integer (8-bit)
ANY_TO_INT	Convert to Integer (16-bit)
ANY_TO_DINT	Convert to Long Integer (32-bit – Default)
ANY_TO_LINT	Convert to Long Integer (64-bit)
ANY_TO_TIME	Convert to Timer
ANY_TO_REAL	Convert to Real
ANY_TO_LREAL	Convert to Double
ANY_TO_STRING	Convert to String
NUM_TO_STRING	Convert Number to String. It allows specifying the number of decimal places to be converted
ATOH	Convert Hexadecimal String to Integer
HTOA	Convert Integer to Hexadecimal String

For example, the ST program is used to convert the type DINT to REAL and then perform the calculation.

```
REAL_Val_1  :=  ANY_TO_REAL (DINT_Val_1) * 3.5 + 4.8 ;
```

Also, open the "HTML Help" from the menu bar and enter the keyword to search for the relevant instructions.



10.4 BCD Conversion (BIN _TO_BCD, BCD_TO_BIN)

The BCD code uses 4 bits to represent the decimal numbers 0 to 9. Suppose a decimal value is "132" which can be converted to a BCD code of "0001 0011 0010" or a binary value of 10000100 (i.e., $2^7 + 2^2 = 128 + 4 = 132$).

Note: The BCD code can only be used to represent numbers from 0 to 9 and can not use the value 1010, 1011, 1100, 1101, 1110, and 1111, which will return "0".

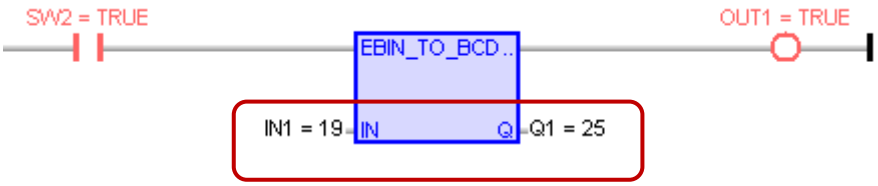
Decimal	BCD				Description
	2^3	2^2	2^1	2^0	
0	0	0	0	0	0
1	0	0	0	1	$2^0 = 1$
2	0	0	1	0	$2^1 = 2$
3	0	0	1	1	$2^1 + 2^0 = 3$
4	0	1	0	0	$2^2 = 4$
5	0	1	0	1	$2^2 + 2^0 = 5$
6	0	1	1	0	$2^2 + 2^1 = 6$
7	0	1	1	1	$2^2 + 2^1 + 2^0 = 7$
8	1	0	0	0	$2^3 = 8$
9	1	0	0	1	$2^3 + 2^0 = 9$

The table lists functions that can be used to do BCD (Binary Coded Decimal) conversion.

Type Conversion Function	Description
BIN _TO_BCD	Convert Binary to BCD value
BCD_TO_BIN	Convert BCD value to Binary

Note: IN1, IN2, Q1, and Q2 are declared as DINT. If the IN is less than or equal to 0, the Q will return 0.

BIN _TO_BCD: $\mathbf{19}_{(10)} = 0001\ 1001_{(BCD)} = 2^4 + 2^3 + 2^0 = 16 + 8 + 1 = \mathbf{25}$



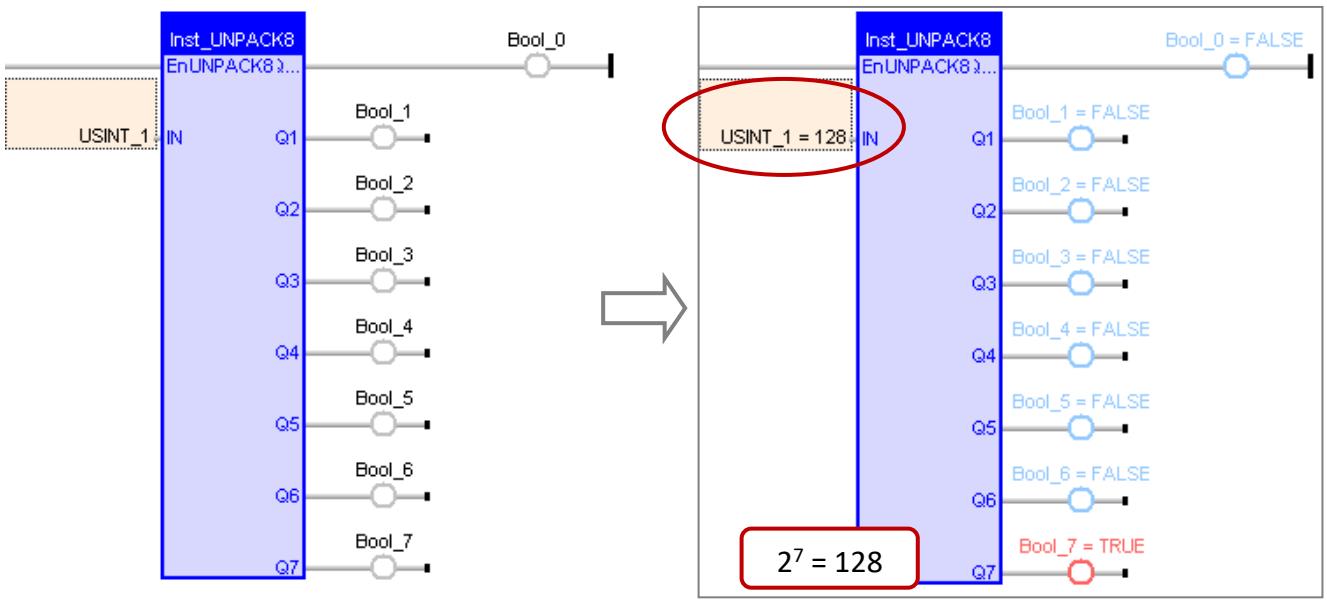
BCD_TO_BIN: $\mathbf{33}_{(10)} = 0010\ 0001_{(2)} = \mathbf{21}_{(BCD)}$

$\mathbf{15}_{(10)} = 0000\ \sout{1111}_{(2)} = \mathbf{0}_{(BCD)}$

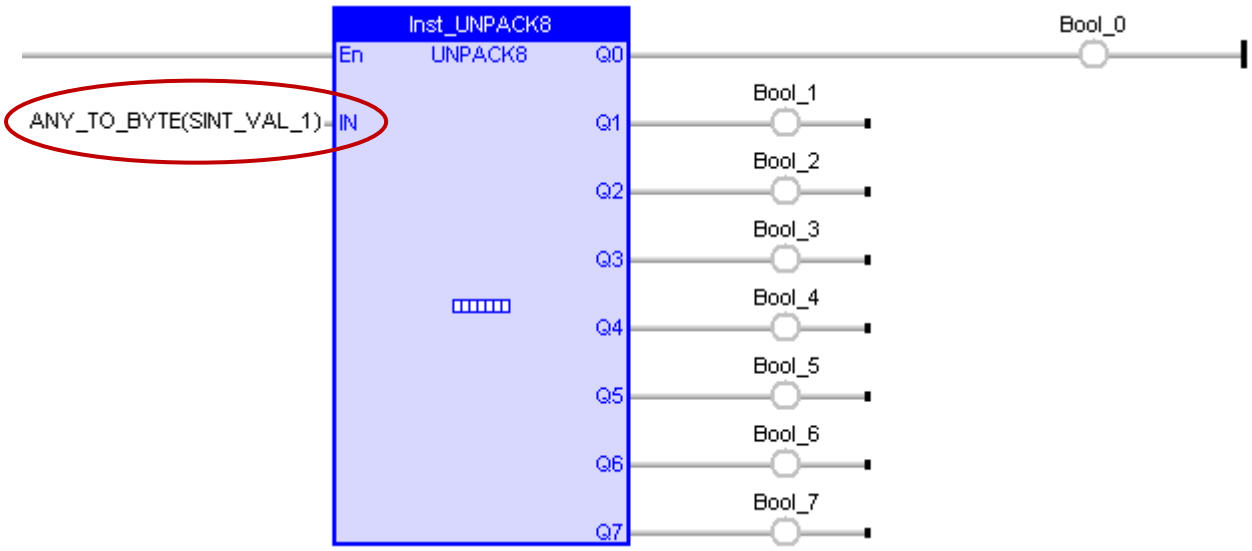
10.5 Pack/Unpack Integer or Boolean

Unpack Integer to Boolean:

The "**UNPACK8**" function can be used to unpack **one BYTE** (or USINT, range: 0 to 255) to **8 Booleans**.



To unpack **one SINT** to **8 Booleans**, first, add the "**ANY_TO_BYTE()**" with the ST syntax to convert the SINT into BYTE type, as follows:



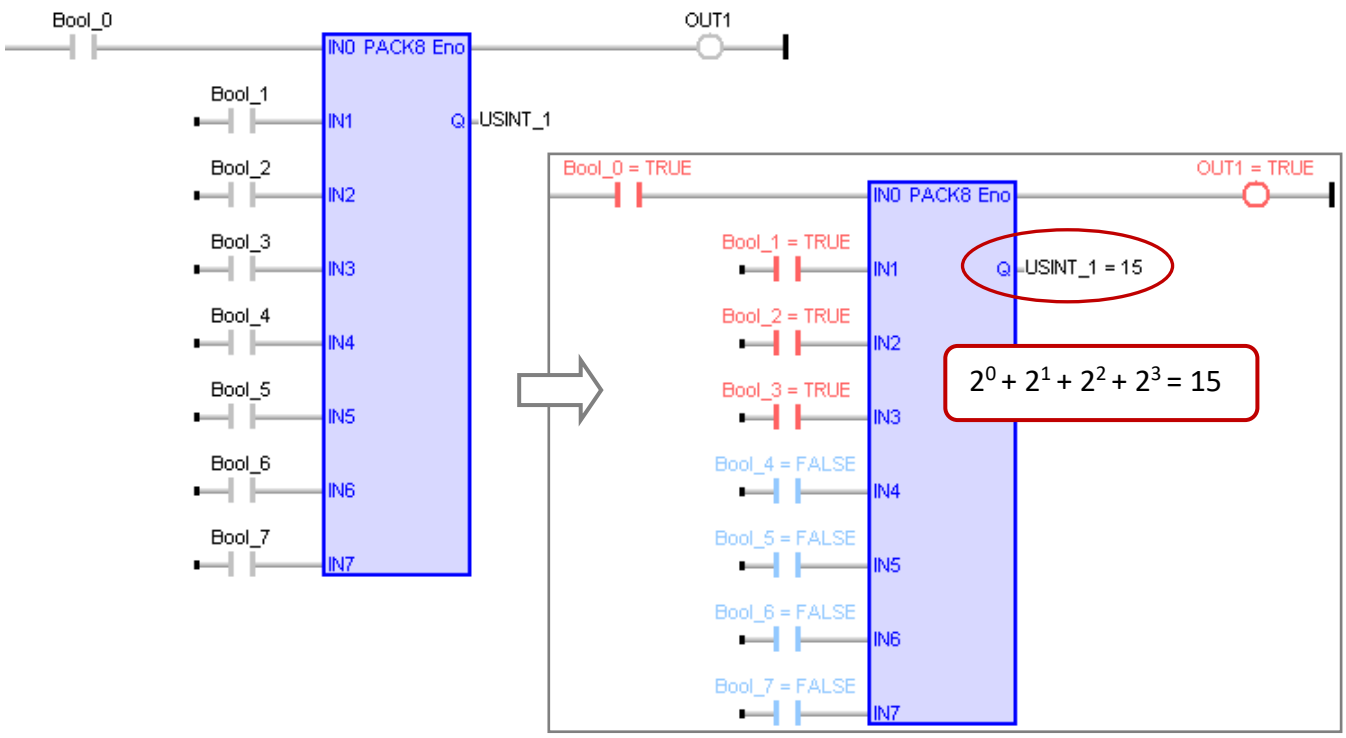
Pack Boolean into Integer:

The "**PACK8**" function can be used to pack **8 Booleans** into **one BYTE**(or USINT, range: 0 to 255).

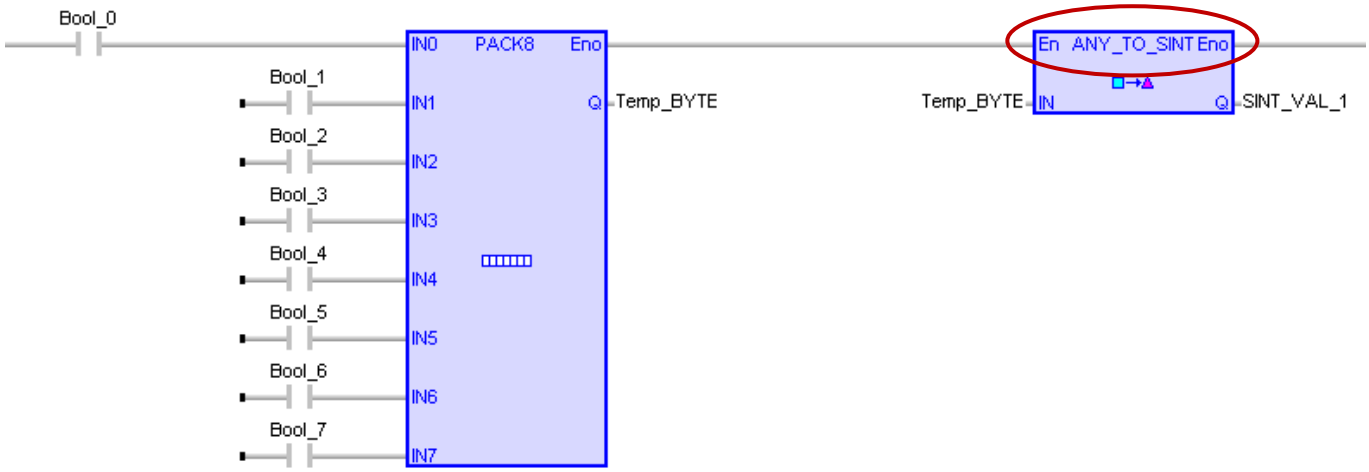
ST Program:

```
USINT_1  :=  PACK8 (Bool_0, I Bool_1, Bool_2, Bool_3, Bool_4, Bool_5, Bool_6, Bool_7);
```

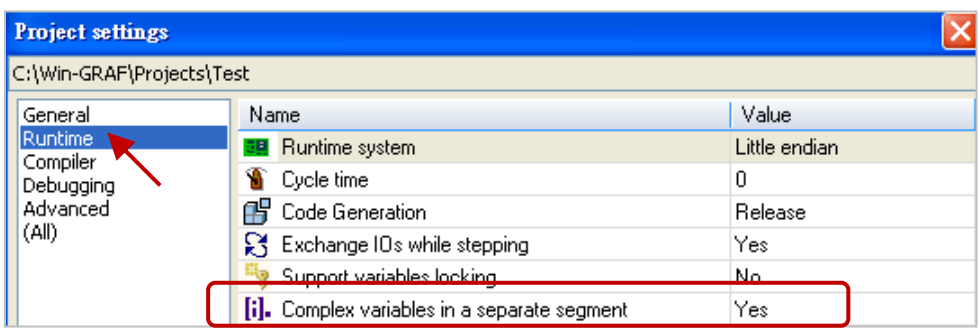
LD Program:



To pack **8 Booleans** into **one SINT**, first specify a "BYTE" variable to the output(Q) for storing the value temporarily and add an "ANY_TO_SINT" function to convert the BYTE to SINT, as follows:



Note: If the compilation fails, click the menu command **Project - Settings** and click "Runtime" to see if the setting of "**Complex variables in a separate segment**" is "**Yes**".



10.6 Pack/Unpack BYTE, WORD, DWORD

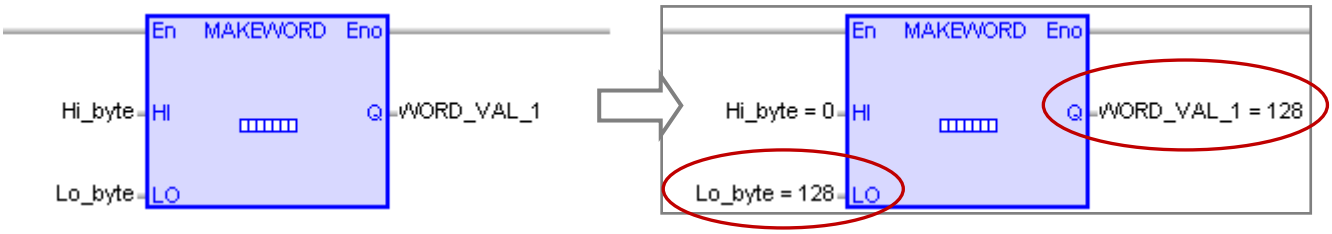
Pack Two 8-bit Data into One 16-bit Data

The "**MAKEWORD**" function can be used to pack **2 Bytes** (or USINT) into **one WORD** (or UINT).

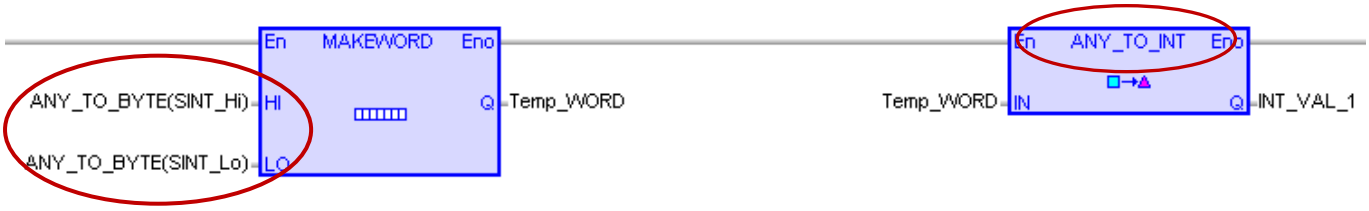
ST Program:

```
WORD_VAL_1  :=  MAKEWORD (Hi_byte, Lo_byte);
```

LD/FBD Program:



To pack **2 SINT** into **one INT**, first, add the "ANY_TO_BYTE()" with the ST syntax to convert SINT into BYTE and add the "ANY_TO_INT" function to convert the packed WORD into INT type.



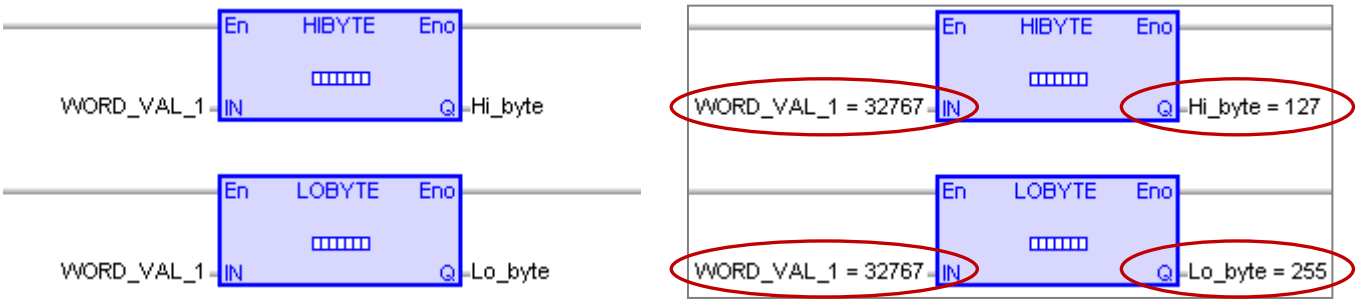
Unpack One 16-bit Data to Two 8-bit Data

"HIBYTE" and "LOBYTE" functions can be used to unpack **one WORD** (or UINT) to **2 BYTEs** (or USINT).

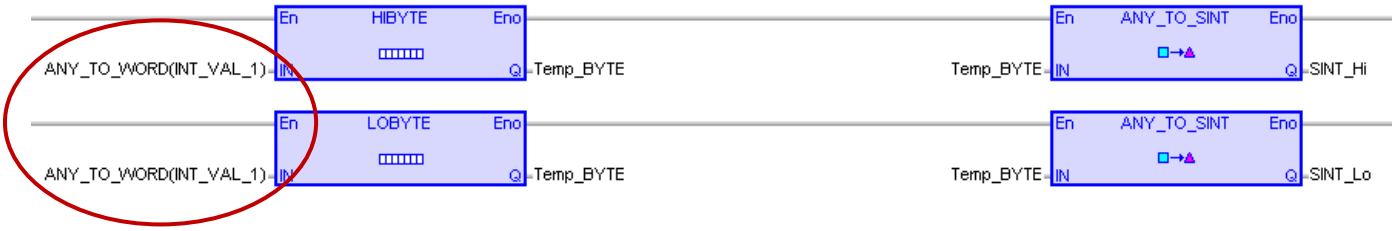
ST Program:

```
Hi_byte  :=  HIBYTE (WORD_VAL_1);
Lo_byte  :=  LOBYTE (WORD_VAL_1);
```

LD/FBD Program:



To unpack **one INT** to **2 SINTs**, first, add the "ANY_TO_WORD()" with the ST syntax to convert INT into WORD and add the "ANY_TO_SINT" function to convert the unpacked BYTE into SINT type.



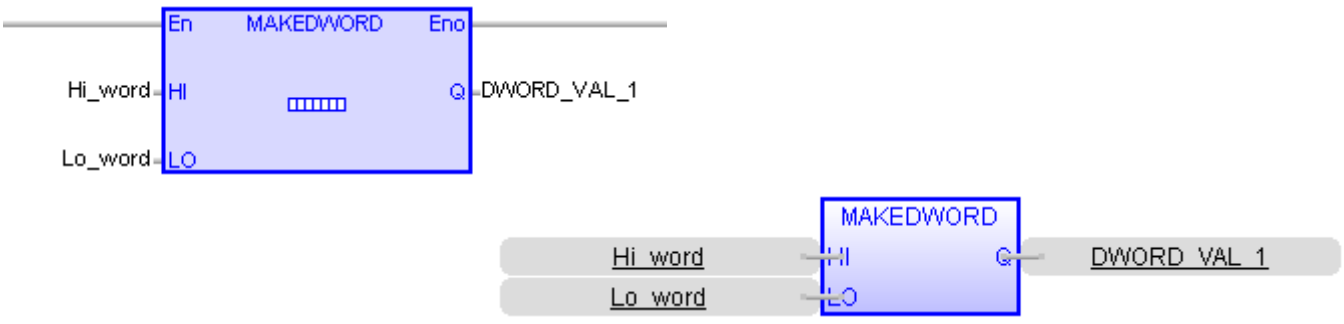
Pack Two 16-bit Data into One 32-bit Data

The "**MAKEDWORD**" function can be used to pack **2 WORDs** (or UINT) into **one DWORD** (or UDINT).

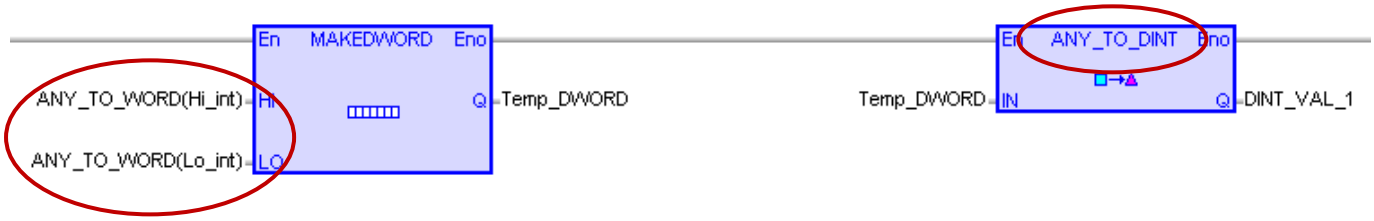
ST Program:

```
DWORD_VAL_1  :=  MAKEDWORD (Hi_word, Lo_word);
```

LD/FBD Program:



To pack **2 INTs** into **one DINT**, first, add the "ANY_TO_WORD()" with the ST syntax to convert INT to WORD and add the "ANY_TO_DINT" function to convert the unpacked DWORD into DINT type.



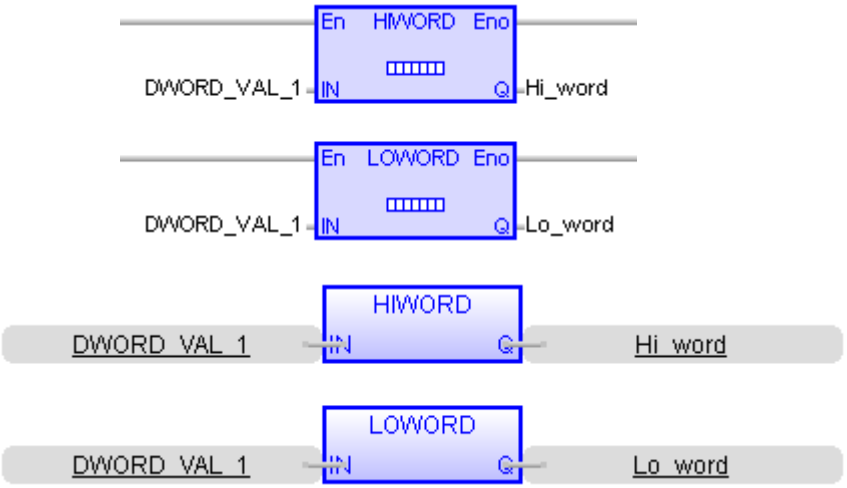
Unpack One 32-bit Data to Two 16-bit Data

"**HIWORD**" and "**LOWORD**" functions can be used to unpack **one DWORD** (or UDINT) to **2 WORDs** (or UINT).

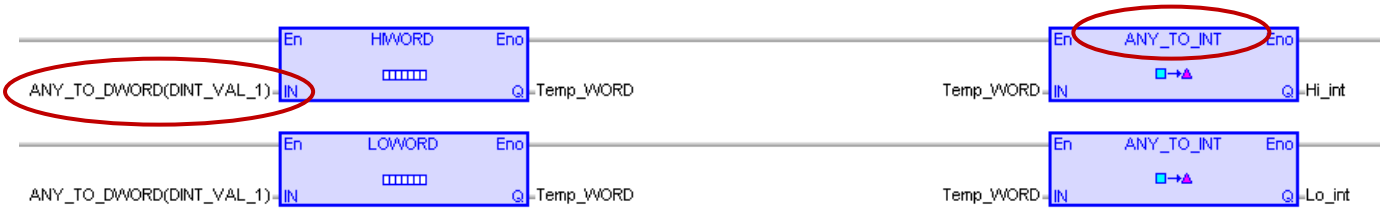
ST Program:

```
Hi_word  :=  HIWORD (DWORD_VAL_1);
Lo_word  :=  LOWORD (DWORD_VAL_1);
```

LD/FBD Program:



To unpack **one DINT** to **2 INTs**, first, add the "ANY_TO_DWORD()" with the ST syntax to convert DINT to DWORD and add the "ANY_TO_INT" function to convert the unpacked WORD into INT type.

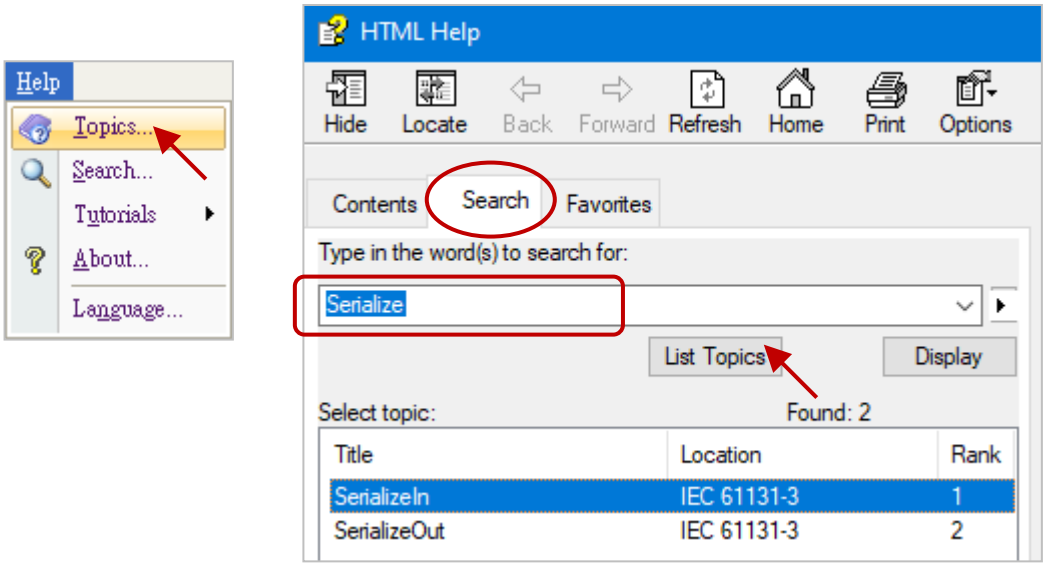


10.7 Unpack Variable to Byte Array or Pack Byte Array into Variable

The "**SerializeOut**" function can unpack a Win-GRAF Variable value to a Byte Array (or USINT Array);
The "**SerializeIn**" function can pack a Byte Array (or USINT Array) into a Win-GRAF Variable value.

Note: 1. The Dim. of Array must be set to at least "8".
2. The STRING variable can not be used in the "**Serialize**" function.

Also, open the "HTML Help" from the menu bar and enter the keyword to search for the relevant instructions.



If the returned value of **SerializeOut()** and **SerializeIn()** is "0", which means the storage location is incorrect or not enough space for the Array data.

```
(* Declare  TMP_DINT as a DINT,
            buf as a BYTE Array, Dim. = 10,
            DINT_Val as a DINT,
            Word_Val as a WORD,
            REAL_Val as a REAL  *)
```

Note:

Data Type	Byte
BOOL, SINT, USINT, BYTE	1
INT, UINT, WORD	2
DINT, UDINT, DWORD, REAL	4
LINT, LREAL	8

Example 1

(* To unpack one DINT_Val to 4 Bytes, and save them separately to the buf[2], buf[3], buf[4], and buf[5] from location 2 of the BYTE Array in Little-Endian order. *)

```
TMP_DINT  :=  SerializeOut (buf, DINT_Val, 2, FALSE) ;
```

Note: The last parameter "FALSE" means that use Little-Endian ordering (i.e., the low byte is stored at the start address).

Example 2

(* To unpack one Word_Val to 2 Bytes, and save them separately to the buf[0] and buf[1] from the location 0 of the BYTE Array in Big-Endian order. *)

```
TMP_DINT := SerializeOut(buf, Word_Val, 0, TRUE) ;
```

Note: The last parameter " TRUE " means that use Big-Endian ordering (i.e., the high byte is stored at the start address).

Example 3

(* To pack the buf[0], buf[1], buf[2], and buf[3] in the Byte Array into one REAL_Val in Little-Endian order *)

```
TMP_DINT := SerializeIn(buf, REAL_Val, 0, FALSE) ;
```

Note: The last parameter "FALSE" means that use Little-Endian ordering (i.e., the low byte is stored at the start address).

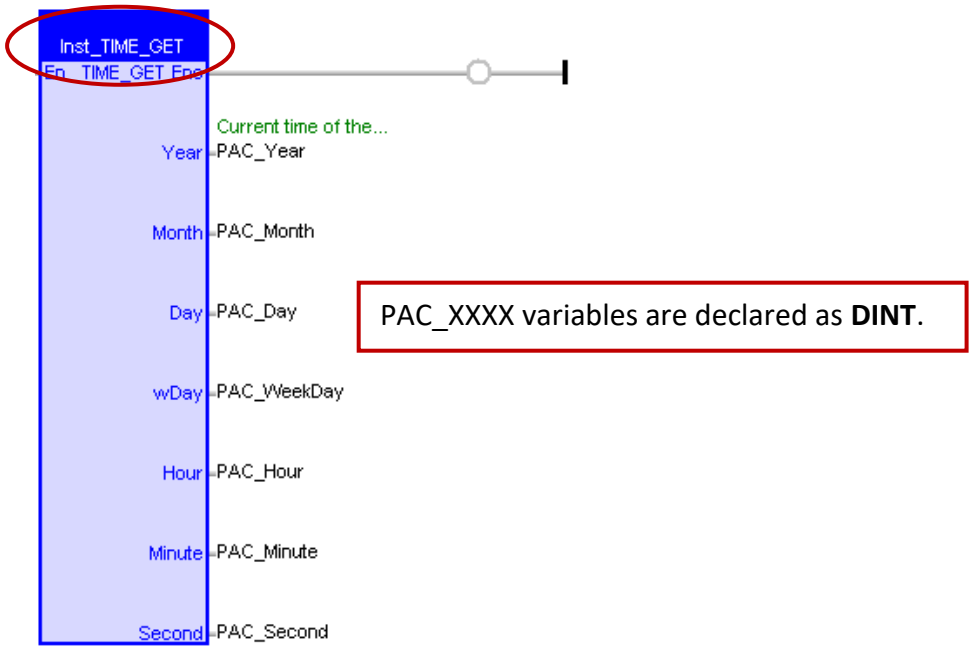
Example 4

(* To map one DINT_Val to one REAL_Val in "Little-Endian" order. *)

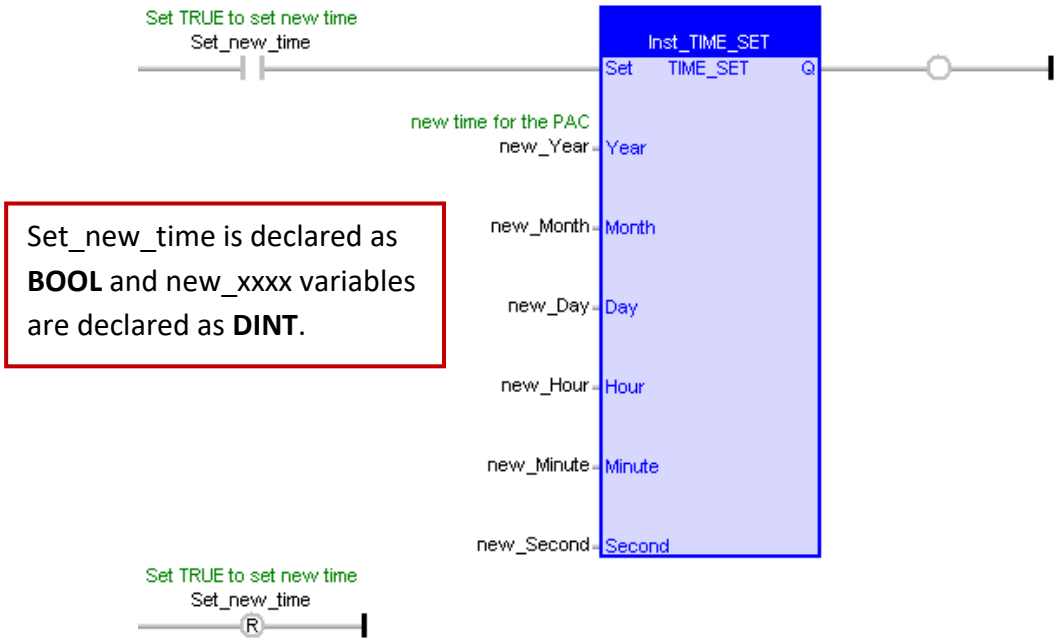
```
TMP_DINT := SerializeOut(buf, DINT_Val, 0, FALSE) ;
TMP_DINT := SerializeIn(buf, REAL_Val, 0, FALSE) ;
```

10.8 Get/Set the PAC Time

The "**TIME_GET**" function block can be used to get the current time of a Win-GRAF PAC.
(Refer to Section 2.2.1)



The "**TIME_SET**" function block can be used to set the current time of a Win-GRAF PAC. First, fill the new time to the variables of "new_Year", "new_Month", "new_Day", "new_WeekDay", "new_Hour", "new_Minute" and "new_Second", then set the "Set_new_time" to "TRUE" one time.



Chapter 11 Commonly Used Tools and Useful Tips

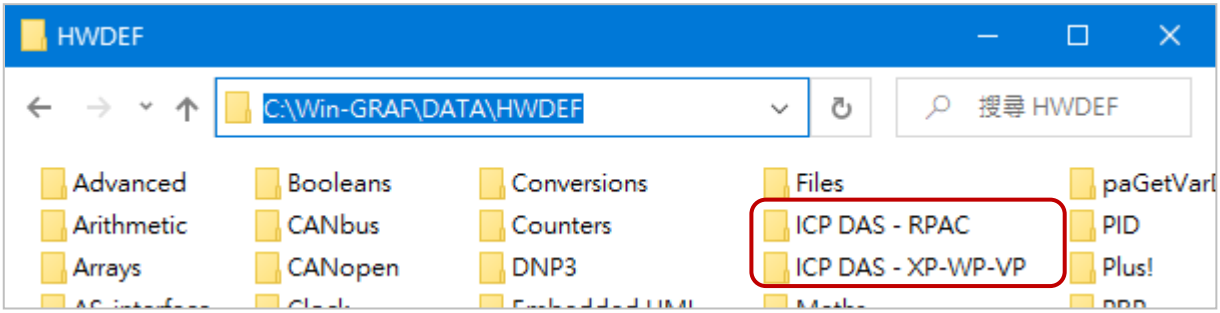
11.1 Upgrade Win-GRAF Libraries

Download the latest version of Win-GRAF libraries (e.g., "win-graf-lib-x.xx.zip") on the website: https://www.icpdas.com/en/download/show.php?num=695

ICP DAS Win-GRAF Libraries, including functions, function blocks, and so on, are stored in the “ICP DAS - XP-WP-VP” and "ICP DAS – RPAC" folders under the “C:\Win-GRAF\DATA\HWDEF” directory.

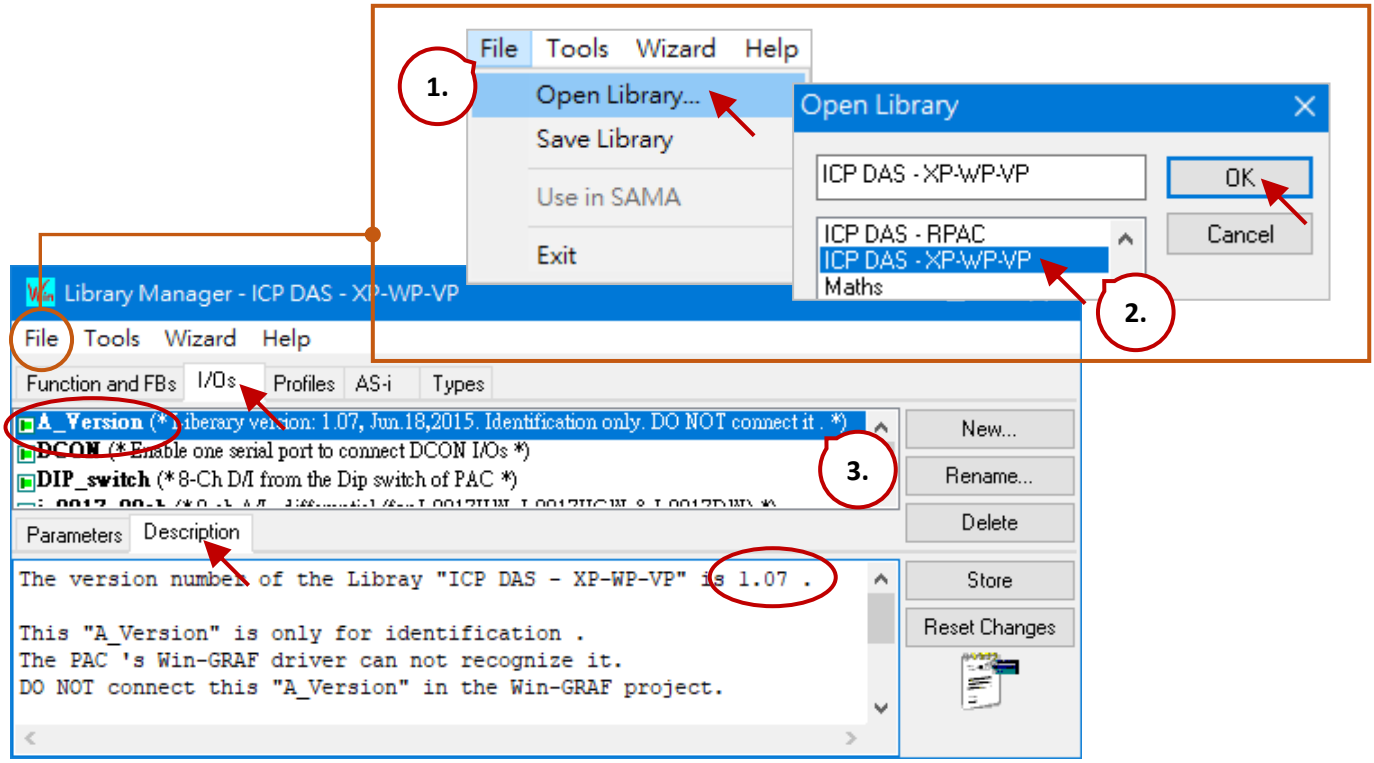
1. Close all Win-GRAF Workbench windows before upgrading. It is recommended to back up the old version of libraries to the other folder (e.g., D:\temp\xxx.zip) before deleting the libraries in the current folder.

2. Copy the new version of “ICP DAS - XP-WP-VP” and "ICP DAS – RPAC" folders to the directory - “**C:\Win-GRAF\DATA\HWDEF**” and execute the Win-GRAF Workbench.



Note:

To know the version number of the Win-GRAF library, open the “ICP DAS - XP-WP-VP” library in the "Library Manager" (refer to Section 1.2.2) and click "A_Version" on the "I/Os" page and then click "Description" to check the version number (e.g., "1.07").



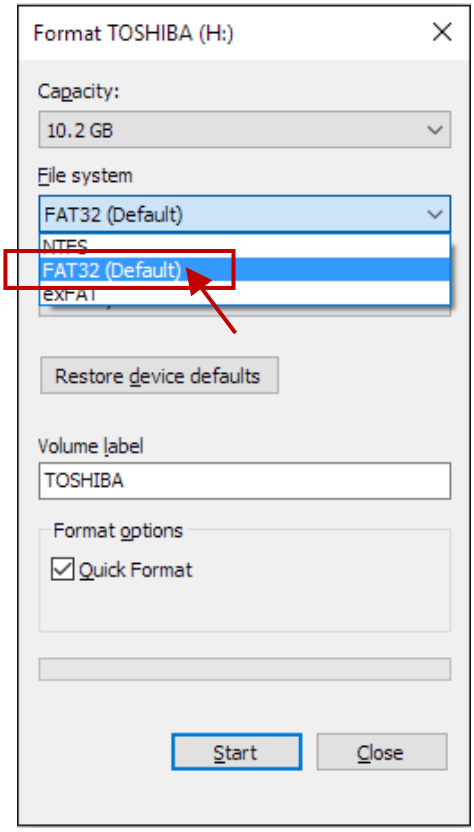
11.2 Upgrade Win-GRAF Driver

The user can download the latest driver on the website and follow the steps below to update the firmware. https://www.icpdas.com/en/download/index.php?model=RPAC-2658M

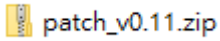
Step 1:

a. Prepare a flash drive.

b. Format your flash drive to **FAT32** format.

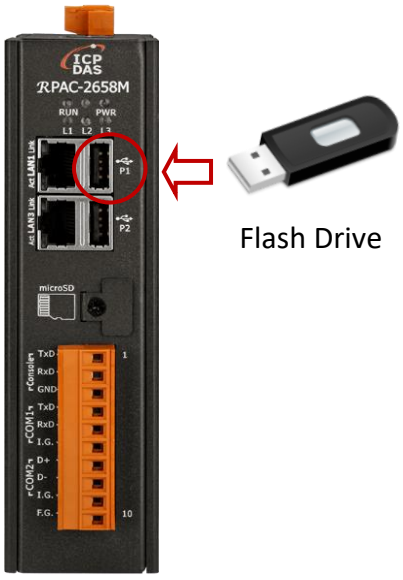


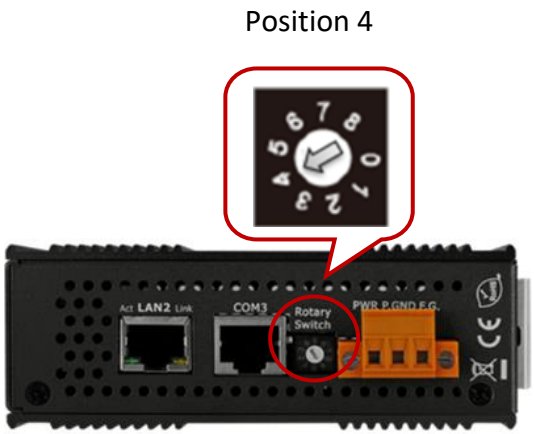
c. Put patch files on your flash drive.
The file name is **patch_vxx.xx.zip.**



Step 2:

Plug your flash drive into the USB port of RPAC-2658M and set the rotary switch to position **4**.





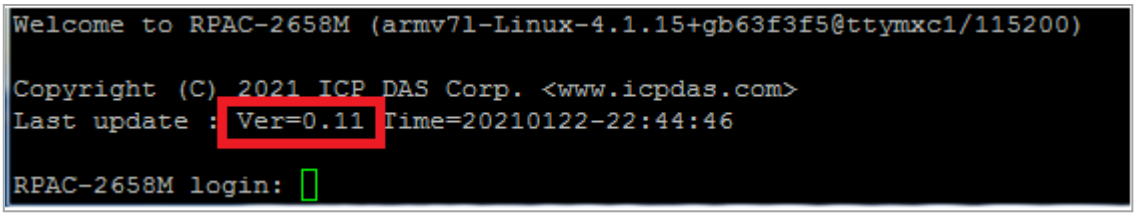
Step 3:

Turn the power on and the firmware update process will be executed automatically. When the firmware update is completed, the **L1, L2, and L3** LED will be **ON**.

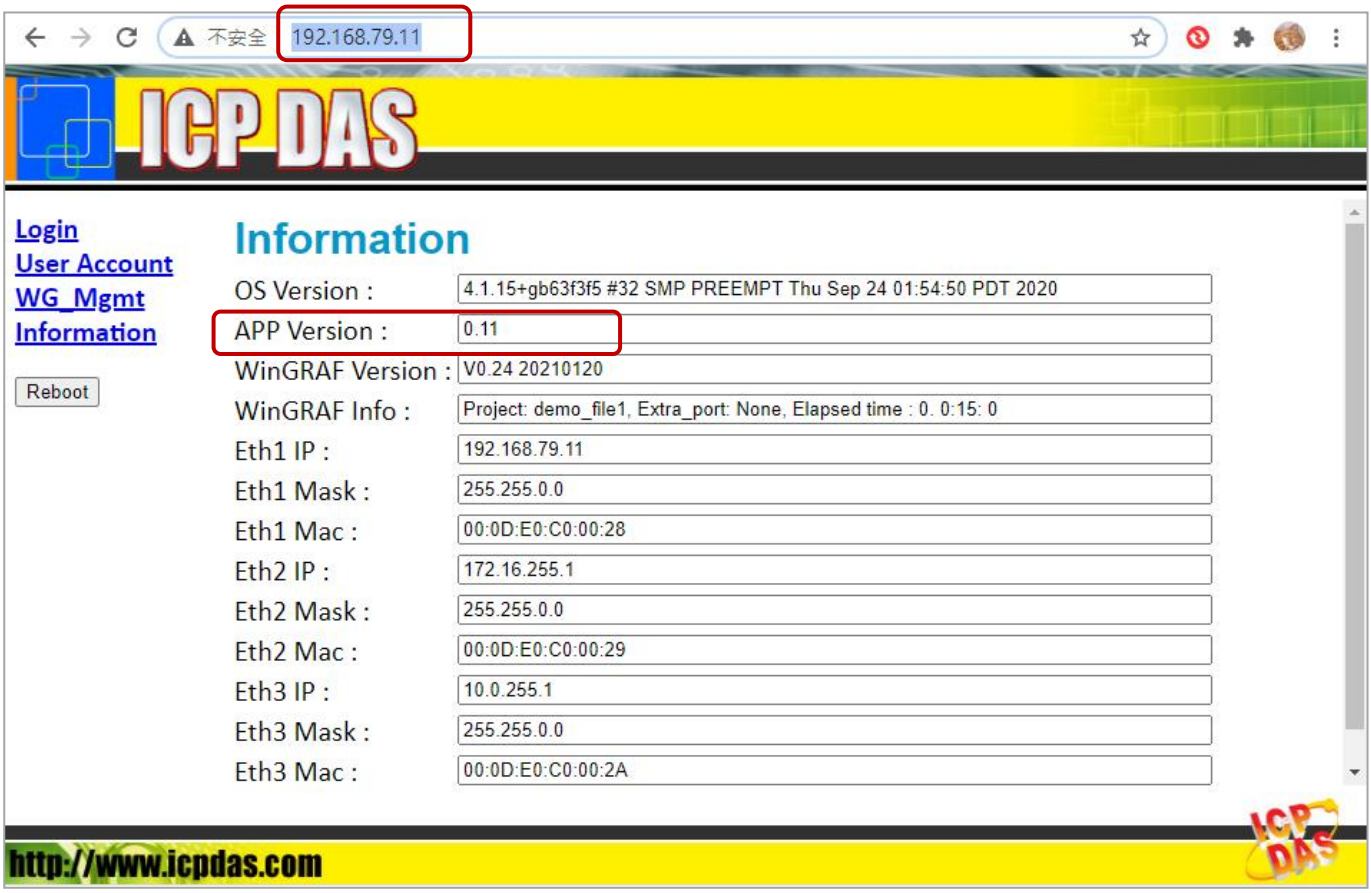
Step 4:
Power off and unplug your flash drive.

Step5:
Set the rotary switch to position **0** and power on again. The new firmware version will be shown on the console port or the web page. It means that the firmware update is successful.

Console port:
Refer to Section 13.2 to connect to the PAC by using the SSH tool.



Web page:
Enter the IP address of the PAC.



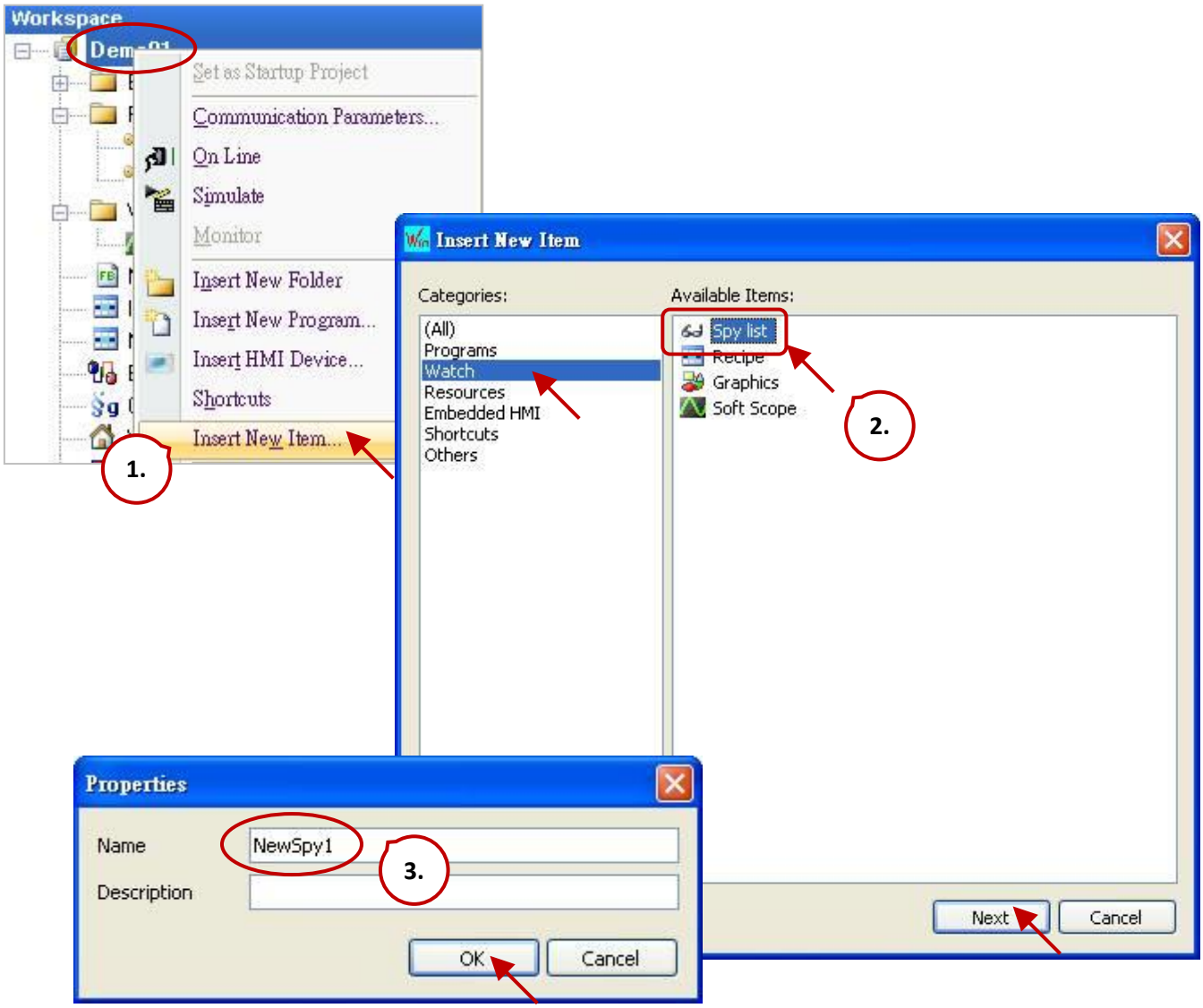
11.3 Spy List

The Spy List is a monitoring function that enables you to watch Win-GRAF variables at runtime. Hundreds of variables may be declared in an application. Workbench allows users to add the specified variable to the Spy List and monitor the current value and status of the variable at runtime.

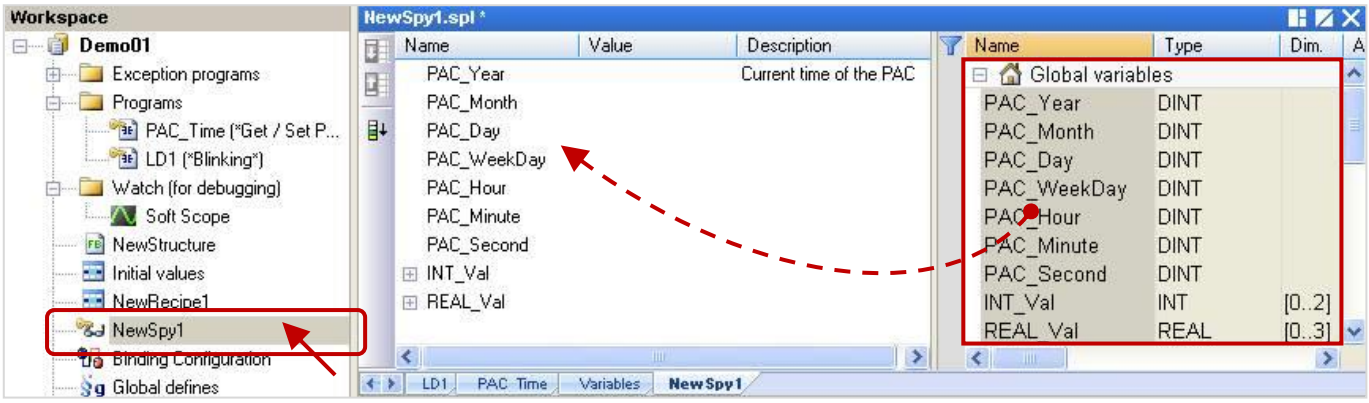
In online mode, you can write and force variable values in a Spy List to actively influence the PLC application behavior.

Steps:

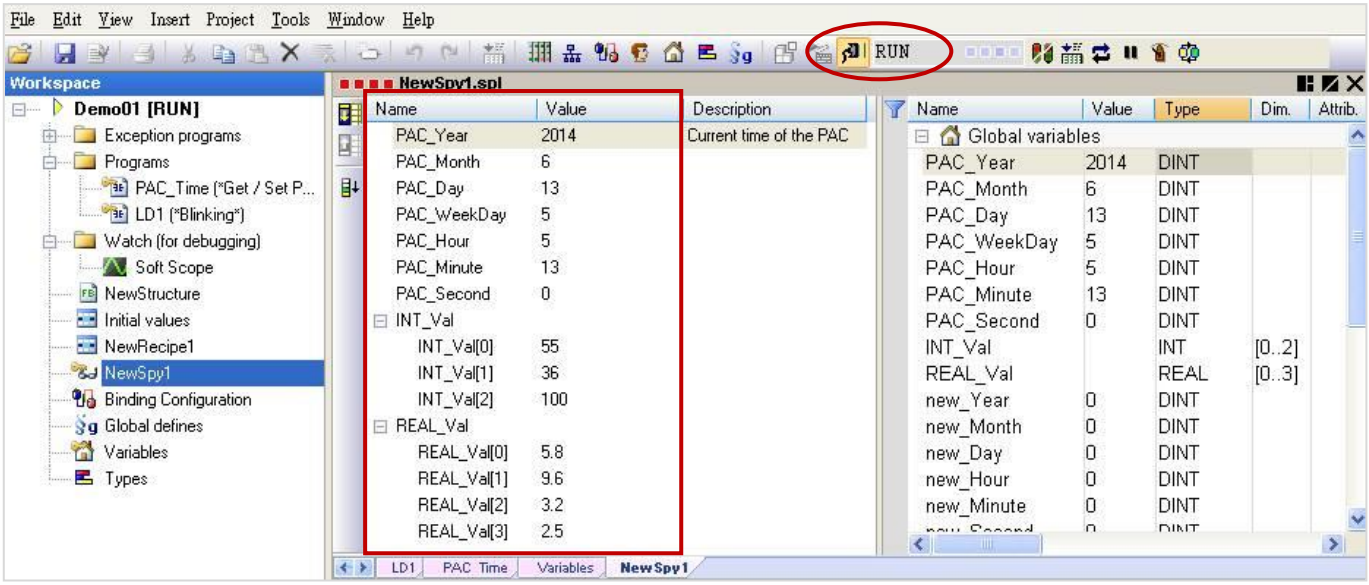
1. Right-click on the project name (e.g., "Demo01") and select **Insert New Item**.
2. Click the **Watch** category and select the **Spy List** item, then click **Next** to the next step.
3. Enter a name (e.g., "NewSpy1") and press **OK**.



4. Double-click "NewSpy1" on the left side to open the window and drag the selected variable you want to observe into this window.



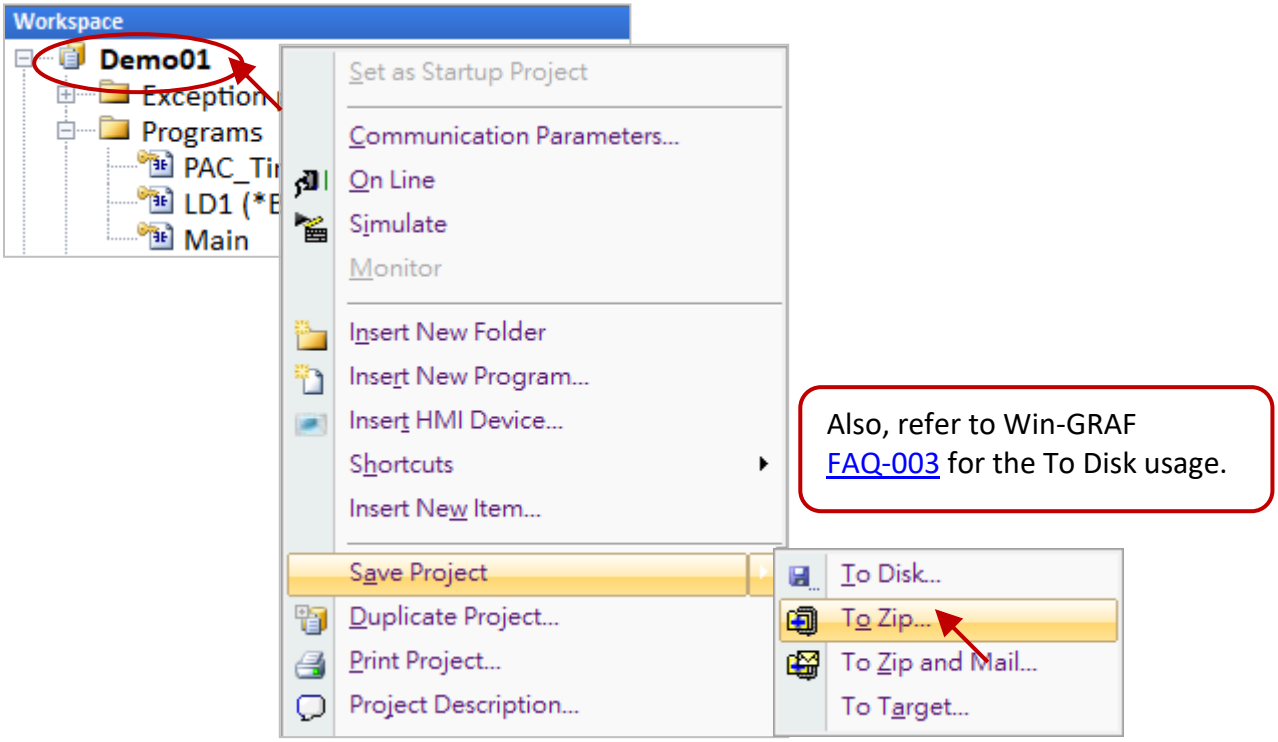
5. If the Win-GRAF workbench is connected to the PAC then the current variable values are shown.



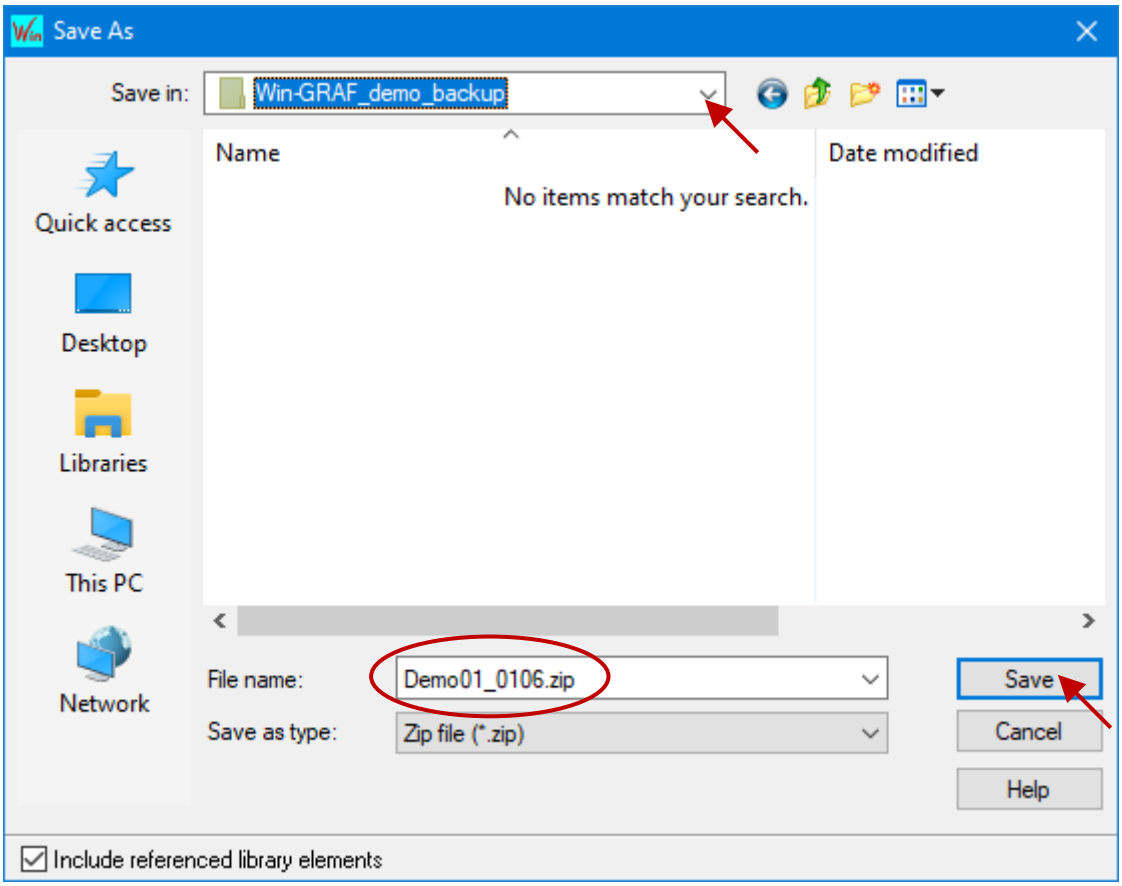
11.4 Backup/Restore a Win-GRAF Project

Back up a Win-GRAF Project:

1. Right-click the project name (e.g., "Demo01") and select "Save Project" and then "To Zip".

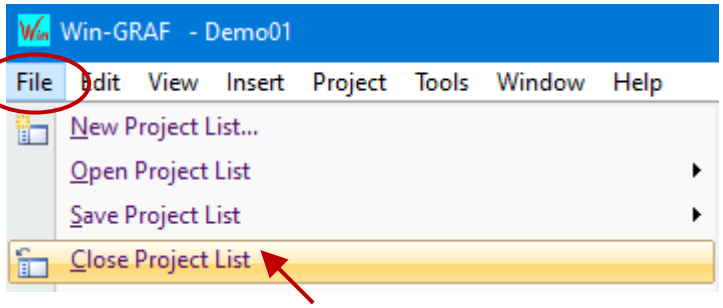


2. Specify the location (e.g., D:\Win-GRAF_demo_backup) to save the file and enter the project name (e.g., "Demo01_0106"), and then click "Save" to complete the backup.

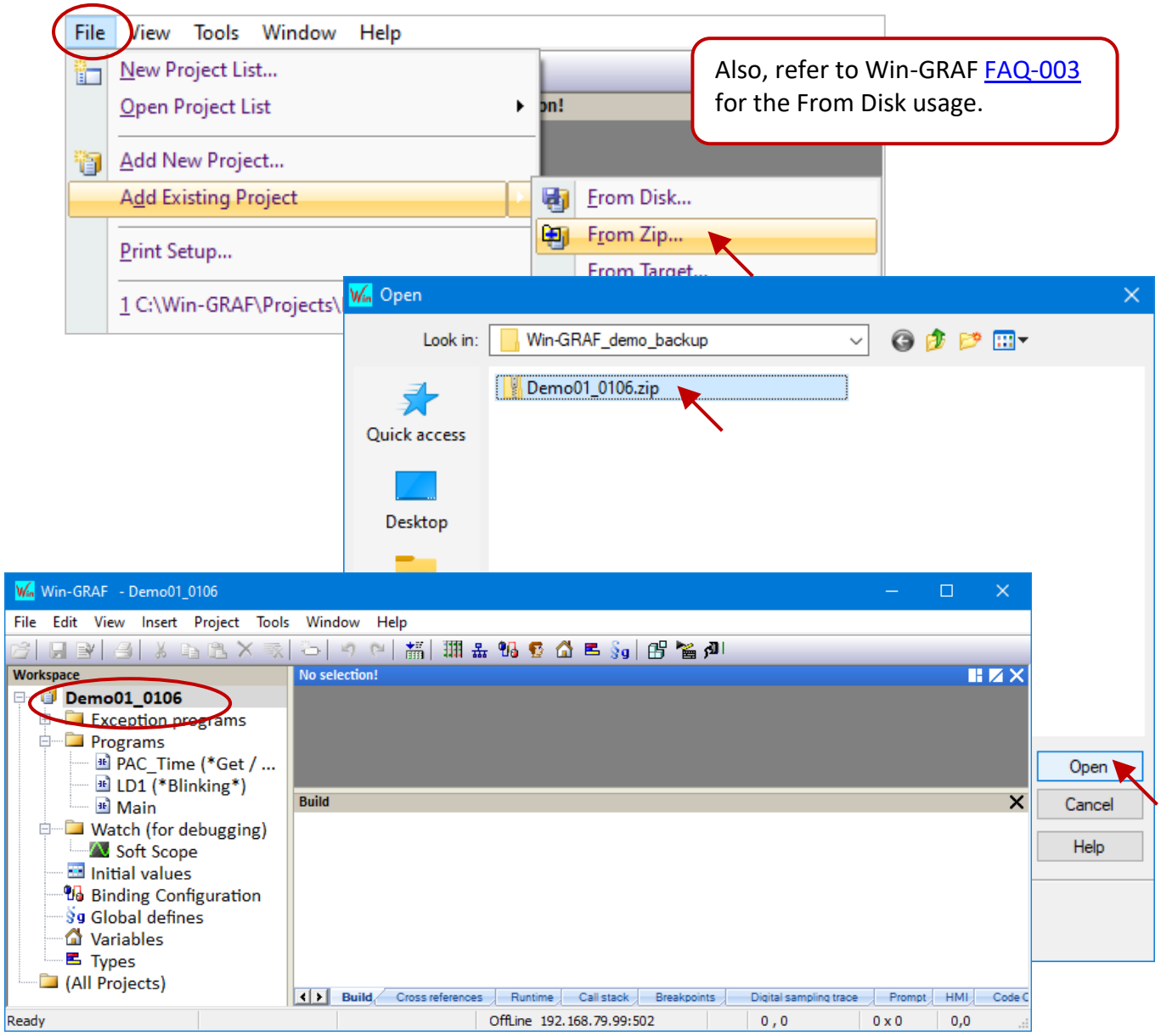


Restore a Win-GRAF Project:

Note: Before restoring, click the menu command **File - Close Project List** to close all projects.



1. Click the menu command "**File** - **Add Existing Project** - **From Zip**", select the project name (e.g., "Demo01_0106"), and click "Open" to complete the restoring.



11.5 Soft Reboot on the PAC

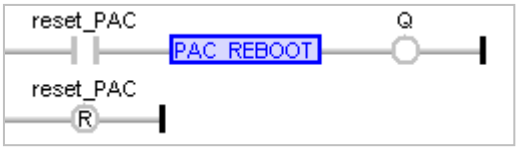
In some cases, the user may want to reboot the PAC by using the software way. Win-GRAF provides the "PAC_Reboot" function for users to restart the PAC automatically.

Note:

DO NOT call this Function in every PAC Cycle, or the PAC will reboot all the time.

Safety Coding:

```
(* "reset_PAC" is declared as BOOL with an initial value "FALSE" and
   "TMP_BOOL" is declared as BOOL *)
(* ONLY when "reset_PAC" is set to "TRUE", the PAC will reboot  *)

    if  reset_PAC  then
      reset_PAC := FALSE ;
      TMP_BOOL  :=  PAC_Reboot( ) ;
    end_if ;
```



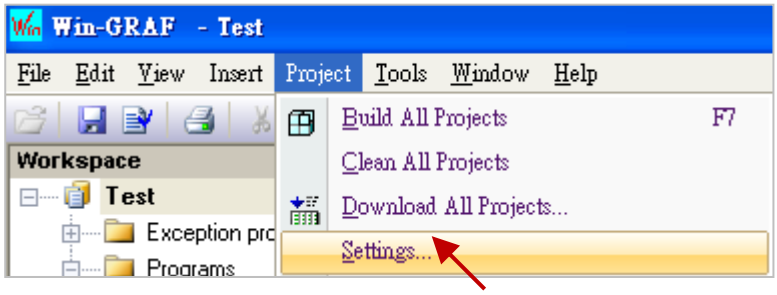
Dangerous Coding:

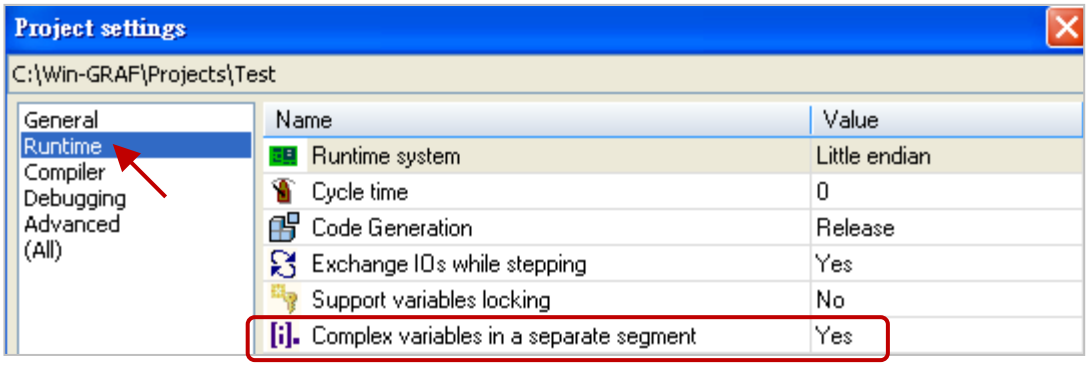
```
(*  "TMP_BOOL" is declared as BOOL *)
(* Dangerous! This code below will cause the PAC to restart repeatedly. *)

TMP_BOOL  :=  PAC_Reboot( ) ;
```

If the PAC keeps rebooting, simply Set the rotary switch of the PAC to "**1**" (i.e., Safe mode) and reboot. After that, download the project again and Set the rotary switch of the PAC to "**0**" (i.e., Normal mode).

11.6 Using ST Syntax in an LD and FBD Program

Win-GRAF Workbench allows users to use a simple ST syntax in the LD or FBD program to facilitate programming. First, click the menu command **Project - Settings** and click **Runtime** and then set the "Complex variables in a separate segment" to "Yes".

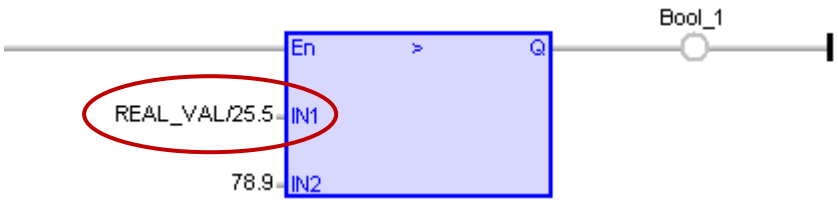




Example:

LD Program:

Using division (REAL_VAL/25.5).



FBD Program:

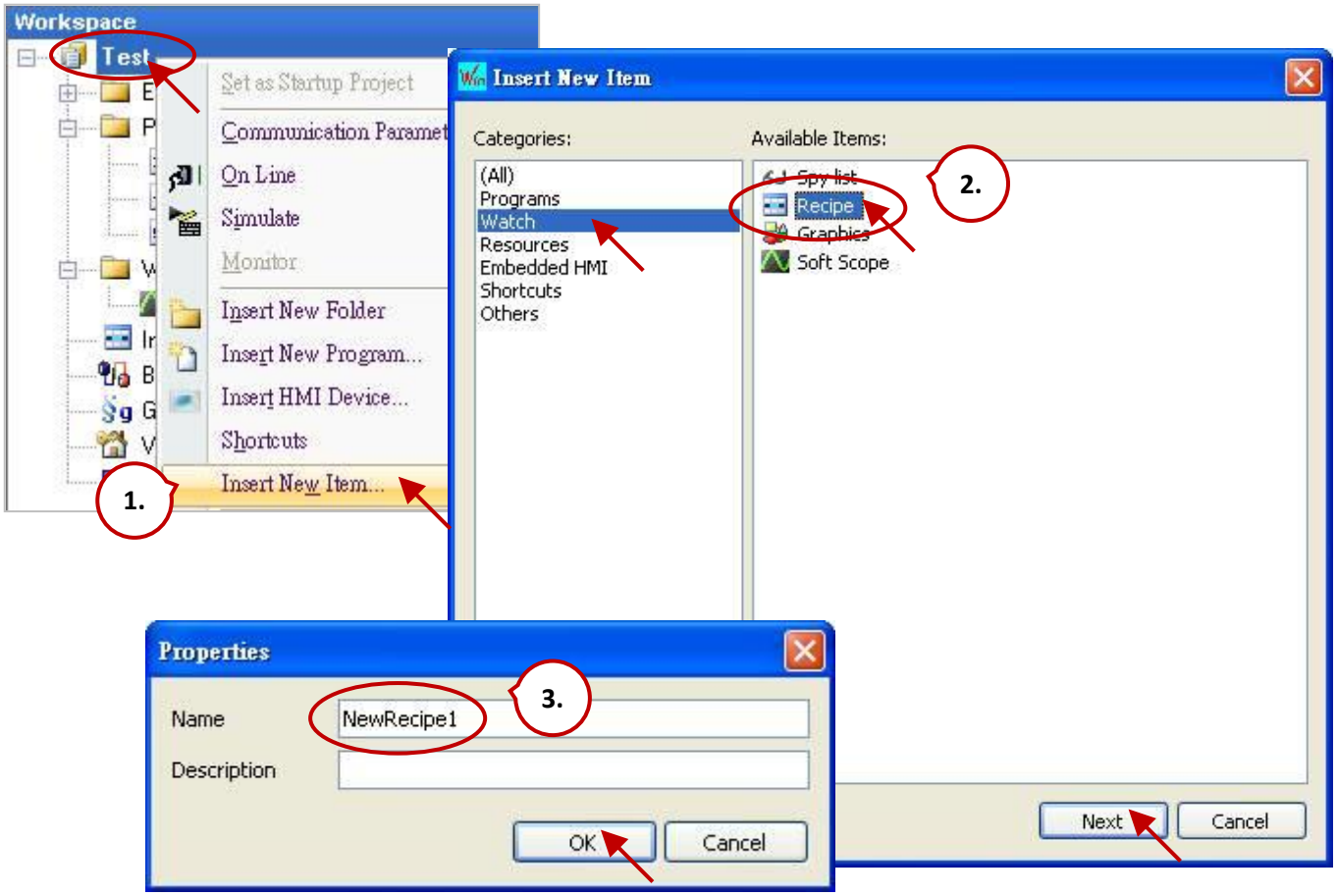
Call a function "ANY_TO_BYTE()" to convert the type from "SINT" to "BYTE".

11.7 Apply a Recipe on the PAC

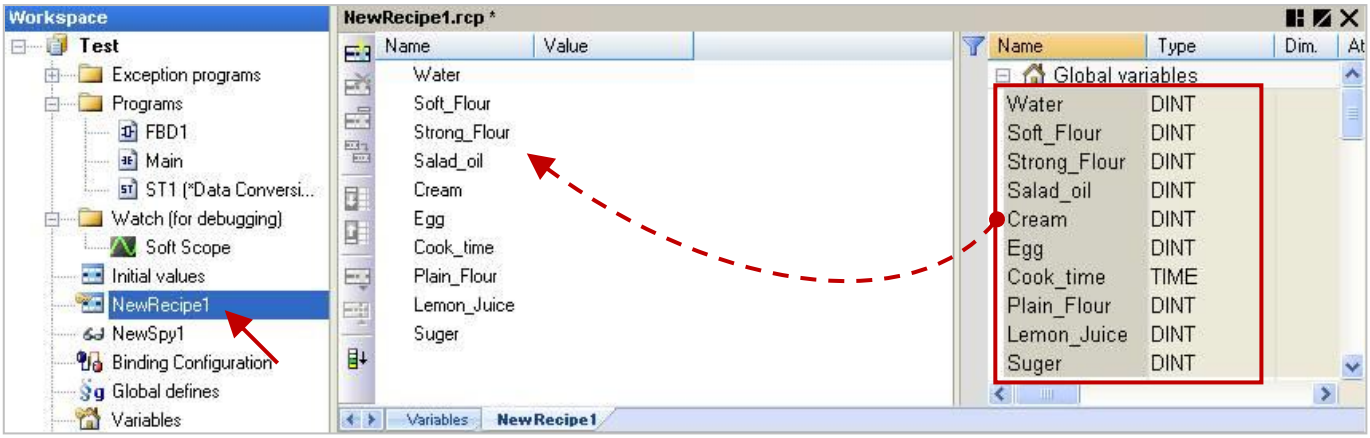
For some applications, it needs to change the configuration of the product in the Recipe to meet the requirement of the different manufacturing processes. The user can assign the required variables and values for each product. Next time when producing the other product, the user can choose the desired product configuration to apply to the PAC.

Follow the steps below:

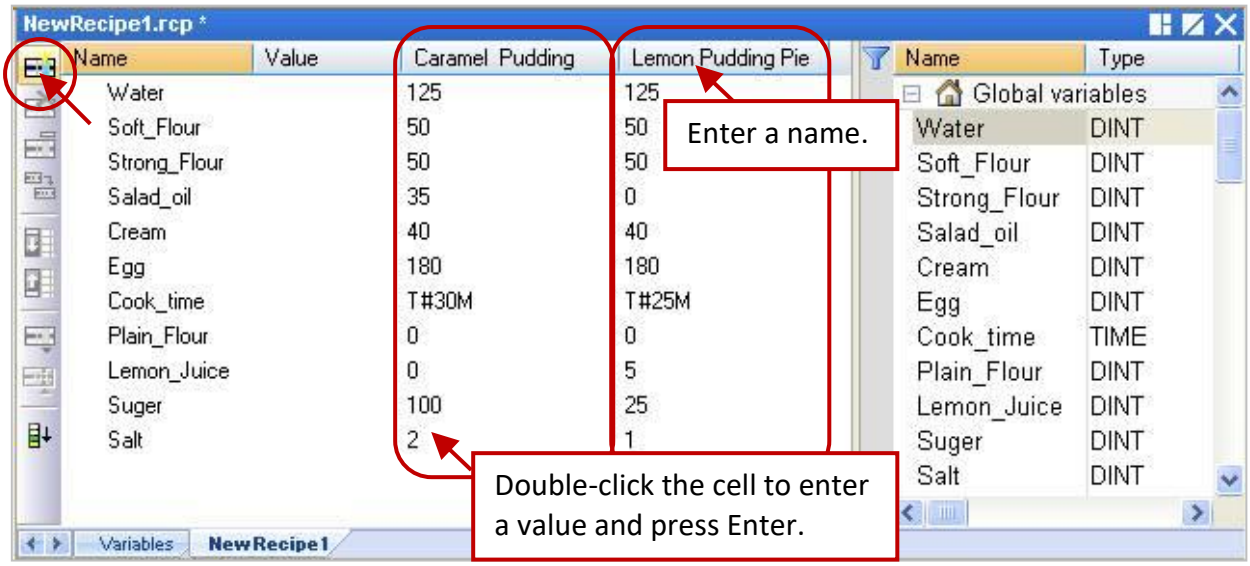
1. **Create a Recipe.** Right-click the project name and select **Insert New Item**. Click the **Watch/ Recipe** item and click **Next**, then enter a name (e.g., "NewRecipe1") and click **OK**.



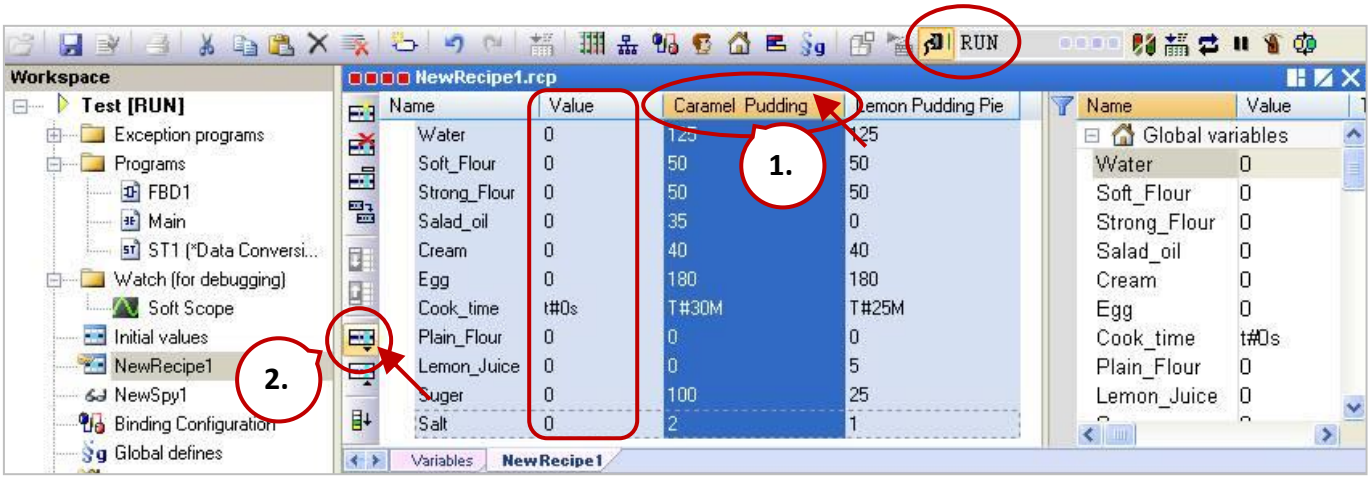
2. **Assign the variables.** Double-click **NewRecipe1** in the Workspace to open the window and drag all needed variables into it.

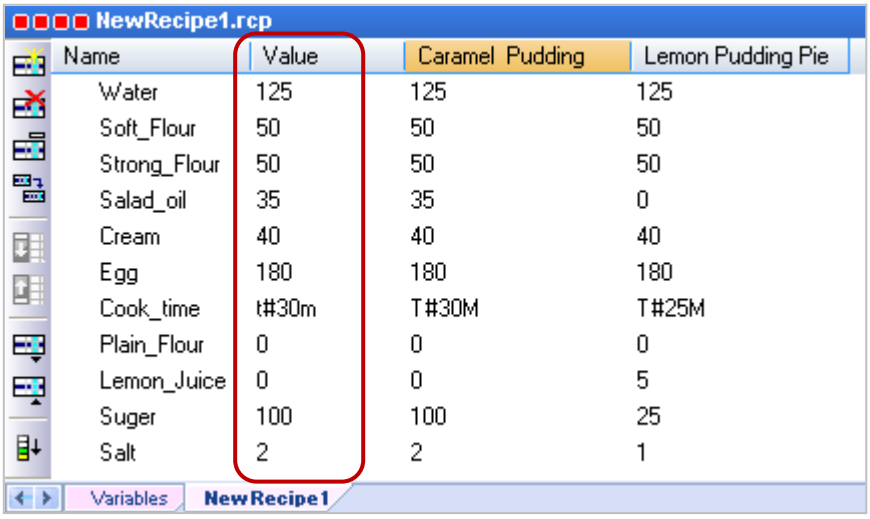


3. **Configure the Recipe.** Click the **Insert Column** button to add a column and enter a name. Double-click the blank cell to enter a value and press the Enter key.



4. **Apply the Recipe.** Download the project to the PAC and you can see all values are "0". Choose the desired product name and click the "Send Recipe" button to apply the configurations to the PAC.



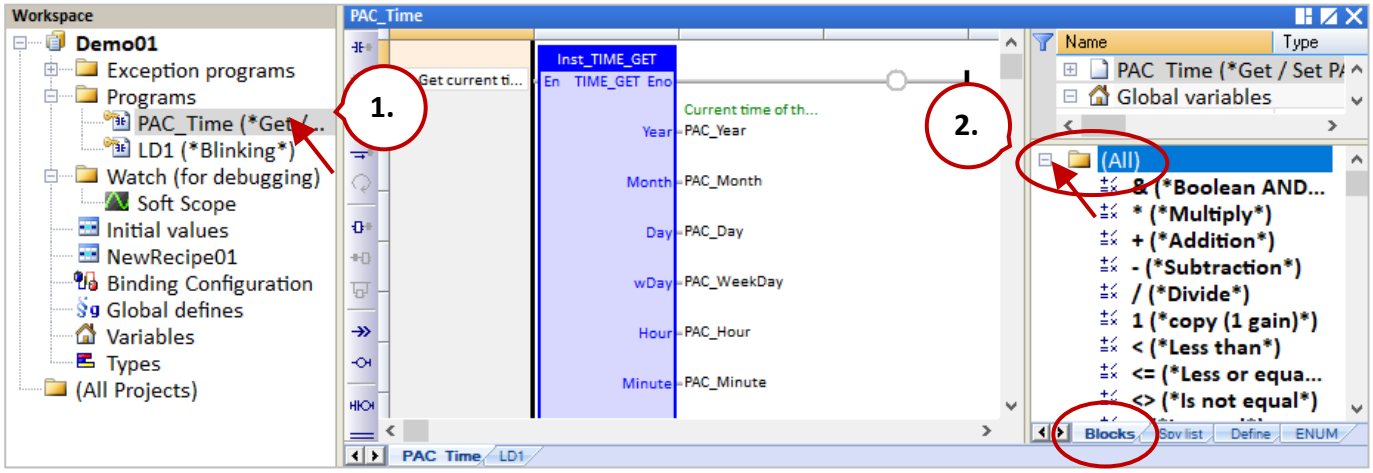


Note: Also, refer to Chapter 6 to use Retain Variables to ensure that all values will not disappear due to an unexpected power cut.

11.8 Functions and Function Blocks Supported by Win-GRAF PACs

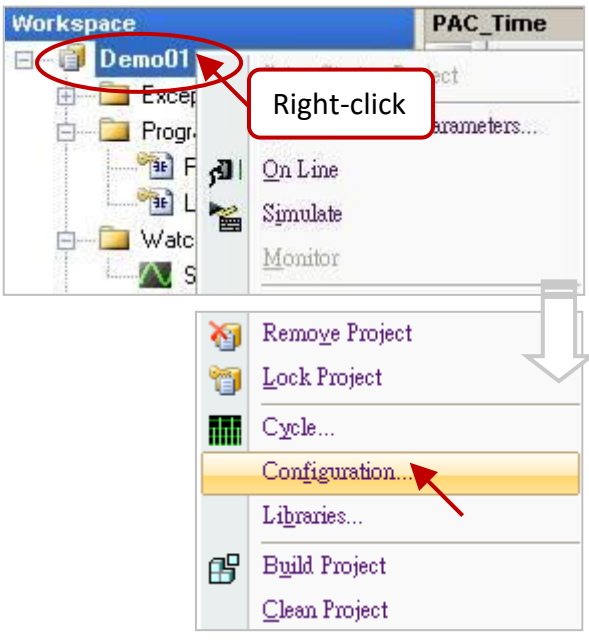
There are many functions and function blocks listed in the Win-GRAF **Block** pane, but some of them are not supported by Win-GRAF PACs. The section describes how to tell which ones are supported or not.

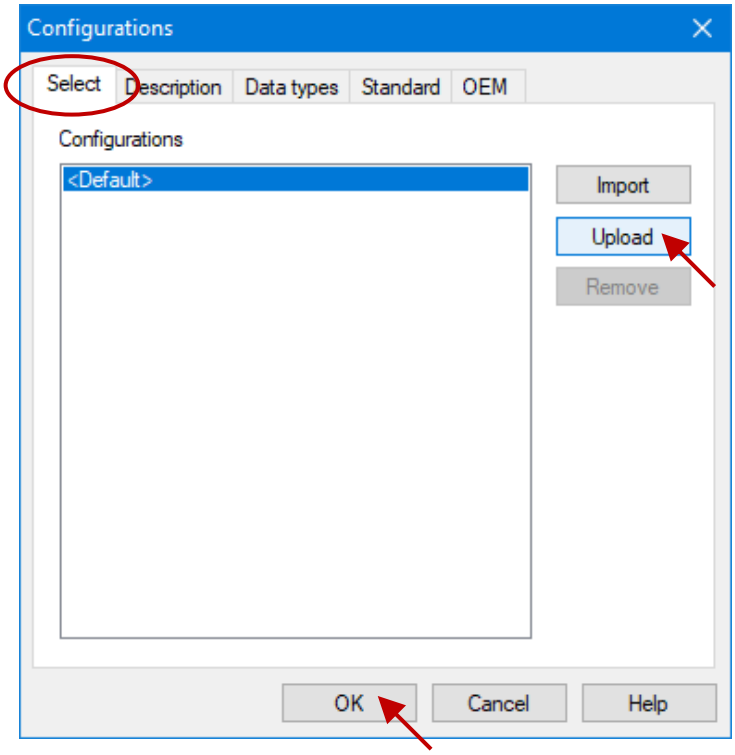
For easy to view all functions and FBs, expand the **All** folder in the **Blocks** pane in any program.



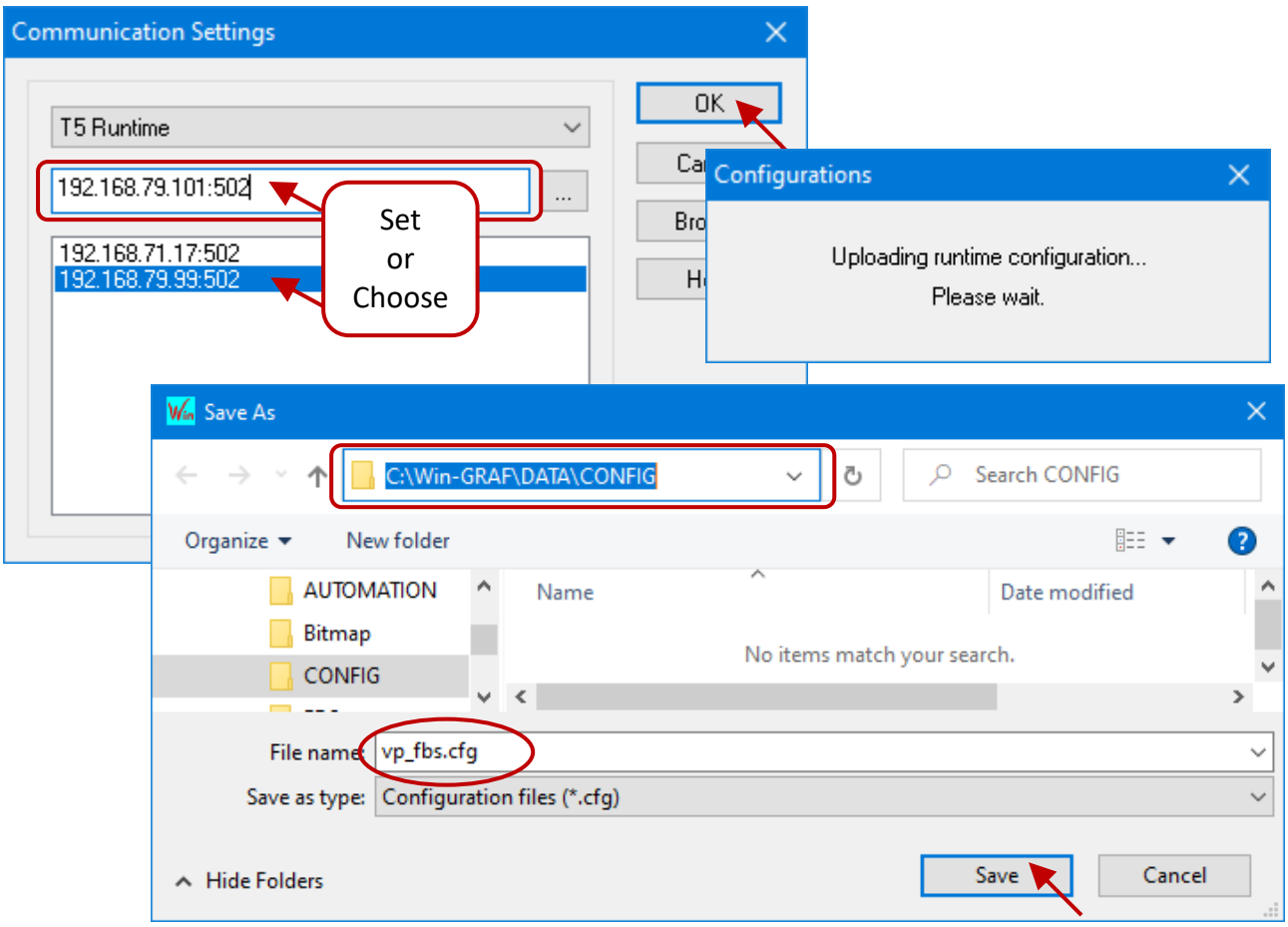
Follow the steps below to upload the configuration file from PAC:

1. Make sure the PAC is powered on and on the same network segment as the Win-GRAF/PC.
2. Right-click the Win-GRAF project name and select "Configuration", and then click the "Upload" button on the "Select" page to open the settings window.

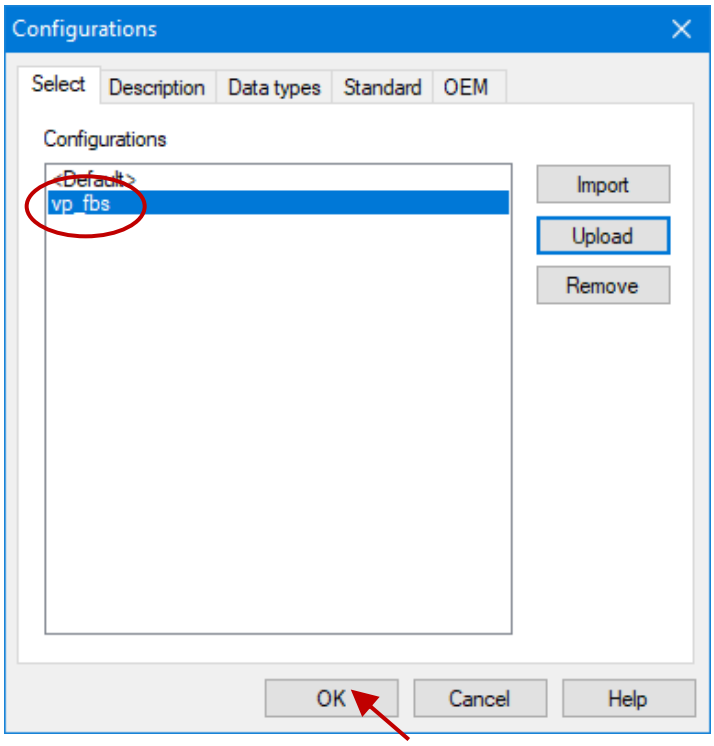


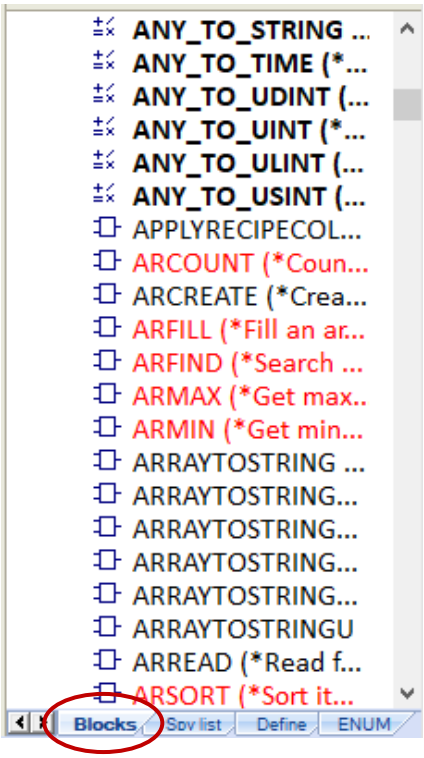


3. Set or choose an IP address of the PAC and click the "OK" button to start uploading. By default, the configuration file will be stored in the C:\Win-GRAF\DATA\CONFIG folder. Simply enter a file name (e.g., "vp_fbs.cfg") and click the "Save" button.



4. The file name will be displayed in the Configurations window, click OK to exit. Then, check the **Blocks** pane again, the functions and FBs marked in red are not supported by Win-GRAF PAC.





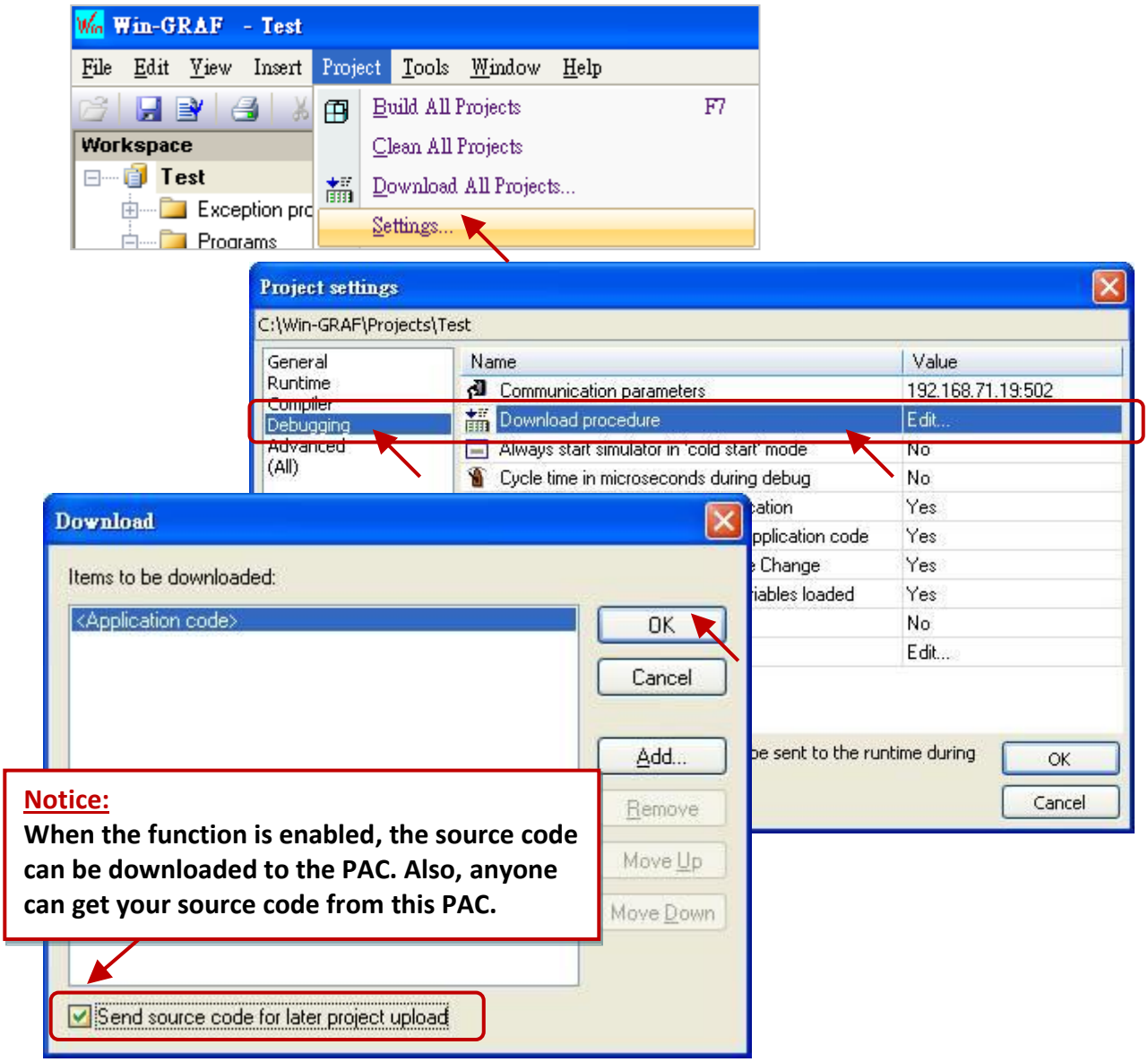
11.9 Upload the Win-GRAF Source Code

For some reason like to prevent the source code from being lost or an incomplete code was handed over by a previous worker, so it is necessary to upload the source code from the PAC to PC.

Otherwise, the user needs to enable the **Download Procedure** function and download the project to the PAC once. Next time, the Win-GRAF project can be uploaded with the source code.

Enable the Download Procedure function:

1. Click the menu command **Project - Settings** to open the settings window.
2. Double-click the **Download procedure** in the **Debugging** setting and check the "**Send source code for later project upload**" box, and then click **OK**.

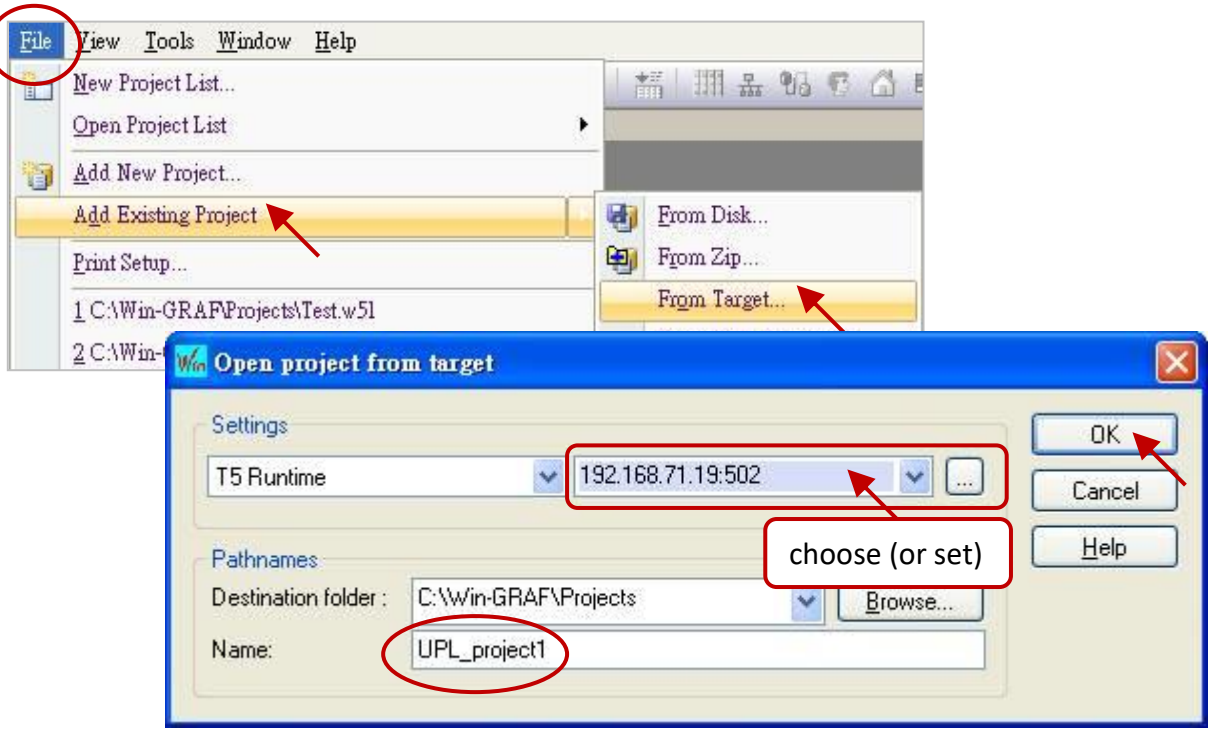


3. Click the menu command **Project - Build All Projects** to compile the project and download it to the PAC. The source code will be stored in **/Systen_Disk/Win-GRAF/t5.upl** on the PAC. The file size varies depending on the content of the project. When the project becomes large and complex, the file size may reach several hundred K Bytes or even more than 1 MB.

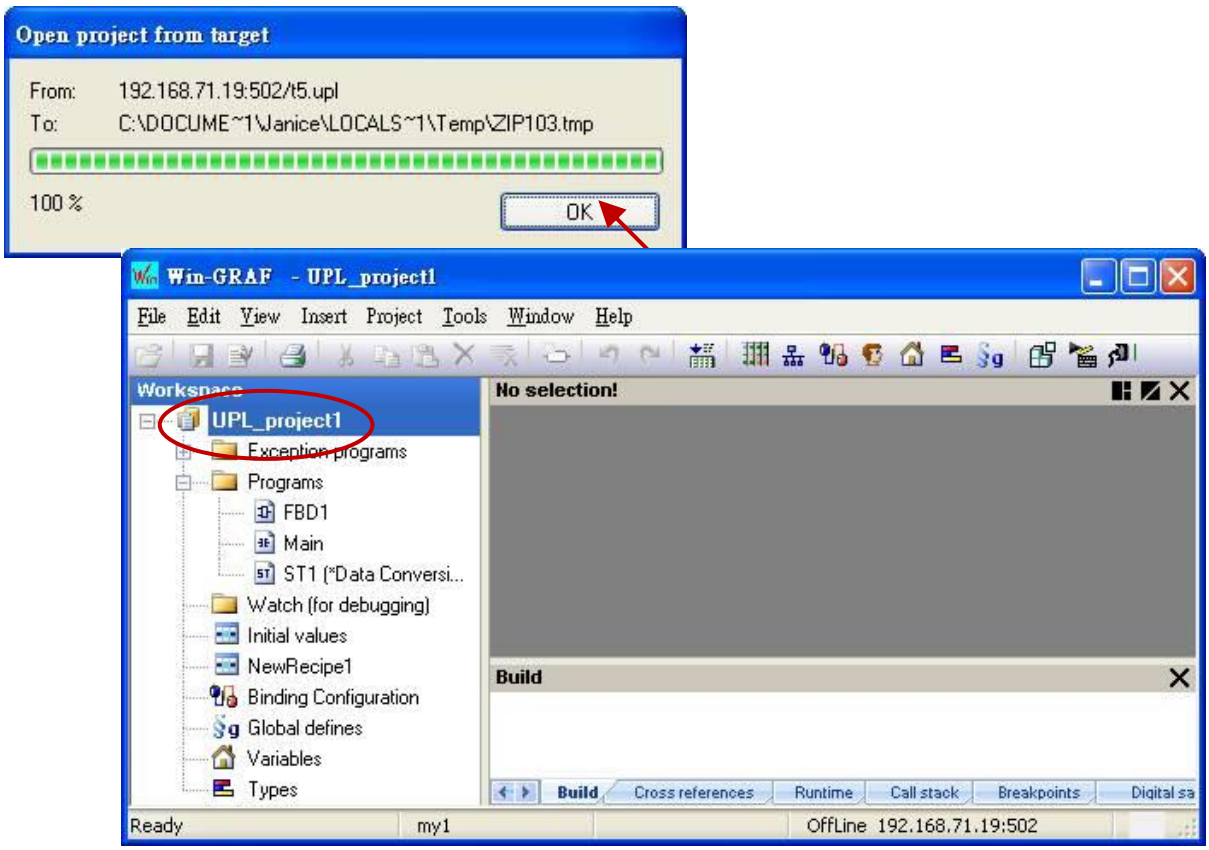
Upload the Source Code from the PAC:

First, click the menu command **File - Close Project List** to close all Win-GRAF projects.

4. Click the menu commands **File - Add Existing Project - From Target** and choose (or set) the PAC's IP address, and then set a project name (e.g., "UPL_project1"). Finally, click "OK" to start uploading.



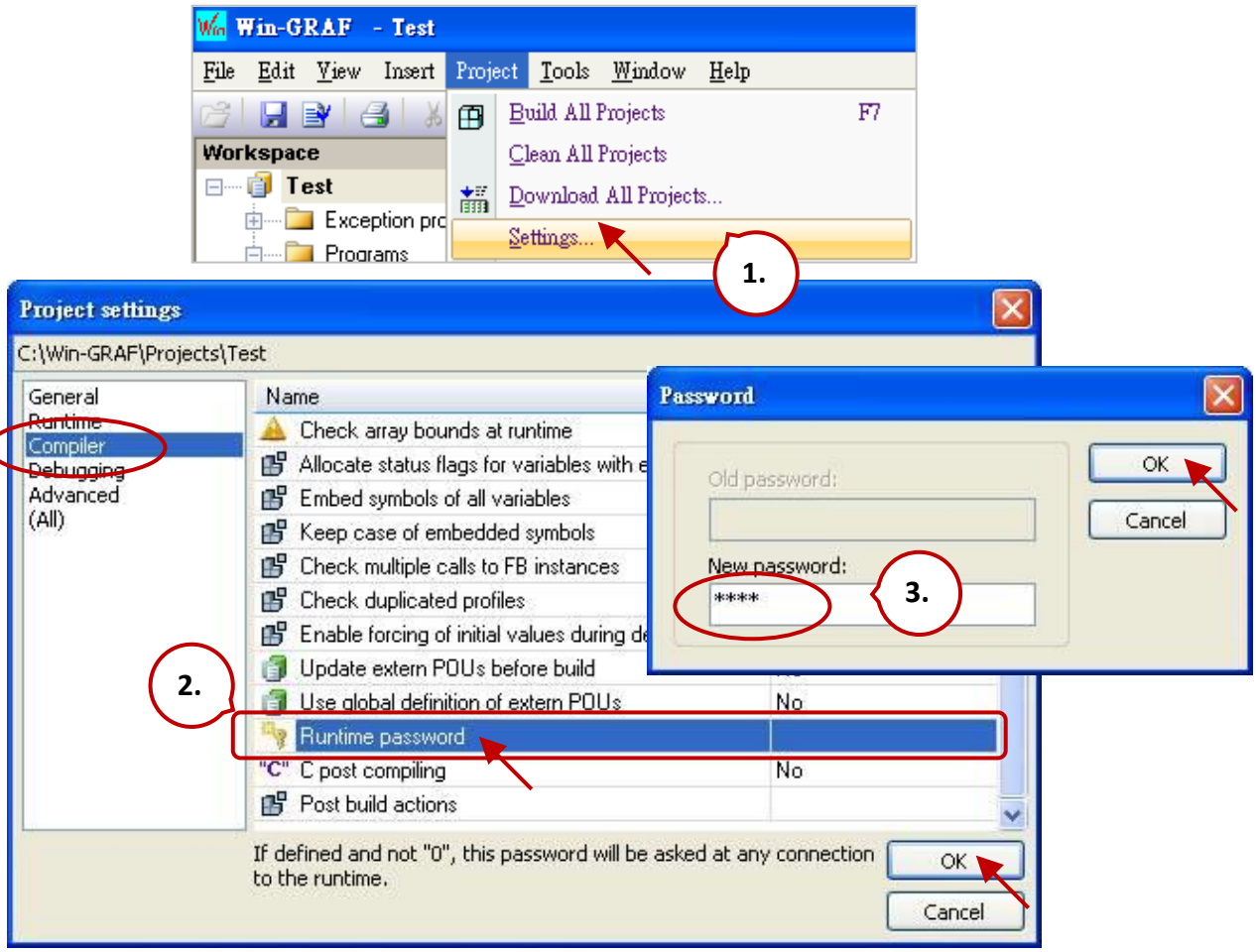
5. After uploading, click the "OK" button to open the Win-GRAF project automatically.



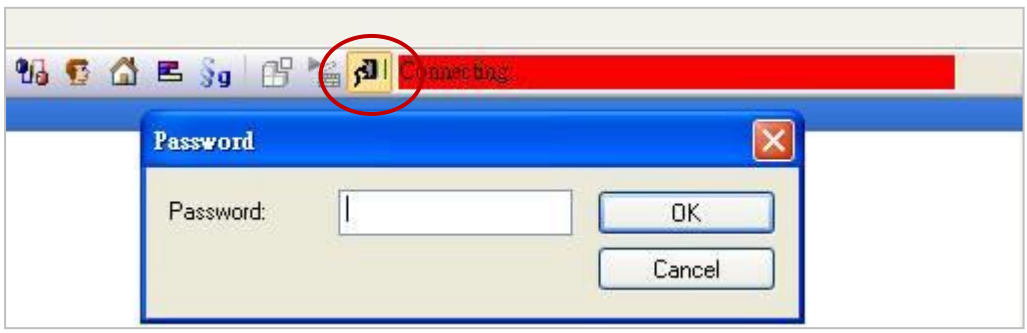
11.10 Set Up the PAC Password

To prevent the PAC from being maliciously changed or stopped by a connected PC while running an important program, you can set a password for the PAC to forbid unauthorized operation.

1. Click the menu command **Project - Settings** to open the settings window.
2. Double-click **Runtime password** in the **Compiler** setting to set a password and click "OK".



3. Click the menu command **Project - Build All Projects** to compile the project and download it to the PAC. Next time, the user needs to enter the password after clicking the "On Line" button.



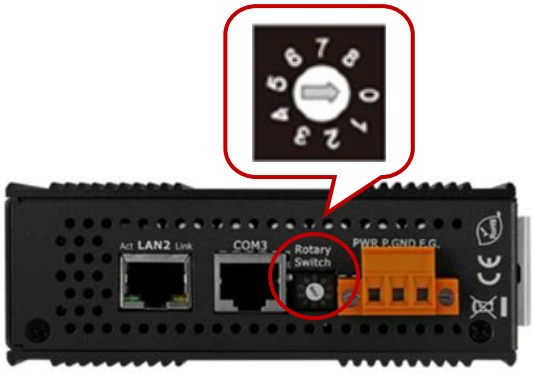
Note:

Remember the password after enabling the function or Win-GRAF cannot connect to the PAC.

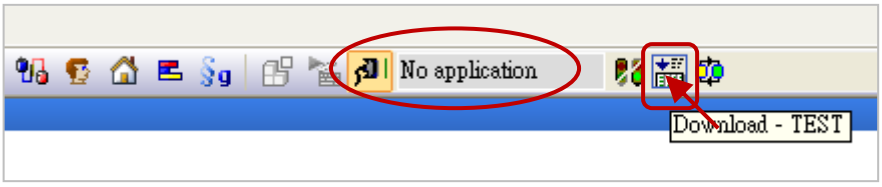
The Only Solution:

Set the rotary switch of the PAC to "**1**" (i.e., Safe mode) and reboot. After that, download the project again and Set the rotary switch of the PAC to "**0**" (i.e., Normal mode).

Position 0



旋轉開關位置	說明
0	Normal Mode
1	Do not perform the user's application
2	LAN1: DHCP Mode
3	LAN1: Static IP Mode IP: 192.168.255.1 Mask: 255.255.0.0
4	Firmware Update Mode



11.11 Using Function or Function Block in the ST Program

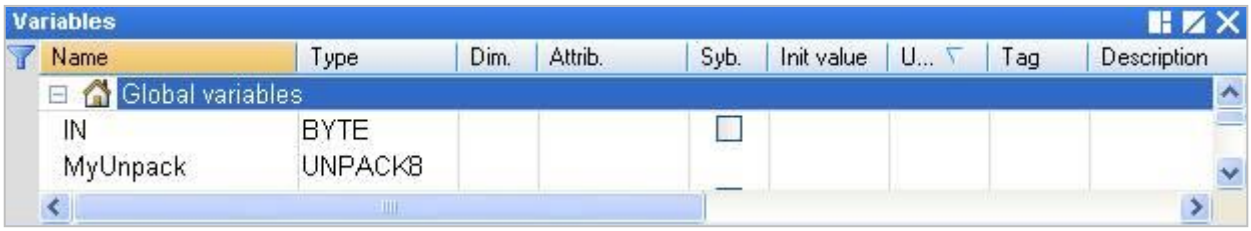
It is easy to use a Function in the ST program, simply call the Function and assign the corresponding parameters. The example below will open the COM1 at the beginning and then send a `Hello' string through the COM1 every 5 seconds.

```
(* Declare  "INIT1" as BOOL and has initial value TRUE,
   Declare  "TMP_BOO" as BOOL,  "TMR1"  as TIME  *)

IF    INIT1    THEN
   INIT1  :=  FALSE ;
      TMR1  :=  T#0s ;
   TSTART  (TMR1) ;
END_IF;
IF  COM_Status(1) = FALSE  THEN
   TMP_BOO := COM_open (1, `19200,N,8,1') ;
END_IF ;
IF    TMR1  >=  T#5s    THEN
   TMR1  :=  T#0s ;
   COM_send_str (1, `Hello: ') ;
END_IF;
```

To use a Function Block in the ST program, you must declare an FB instance variable in the **Variables** window. After that, the usage is similar to the Function as follows:

For example, unpack one Byte to 8 Booleans.

1. Declare the "MyUnpack" as a "UNPACK8" (FB) variable and "IN" as a "BYTE" variable.



2. Edit an ST program.

```
MyUnpack(IN) ;
Q0  :=  MyUnpack.Q0 ;
Q1  :=  MyUnpack.Q1 ;
Q2  :=  MyUnpack.Q2 ;
Q3  :=  MyUnpack.Q3 ;
Q4  :=  MyUnpack.Q4 ;
Q5  :=  MyUnpack.Q5 ;
Q6  :=  MyUnpack.Q6 ;
Q7  :=  MyUnpack.Q7 ;
```

11.12 Protect Your Win-GRAF Program to Avoid Unauthorized Use

When you finish developing a Win-GRAF application and prepare to deliver it to the customer, please think that your application on the PAC may be copied into another PAC with the same model by a third-party?! Be aware! Someone else may steal your hard outcome! The following provides a simple and easy way to protect your application.

Note:

If you give the source code of a Win-GRAF application to someone, then sorry, the following method cannot protect your program. Anyone who got the source code can modify the code and apply it to another PAC.

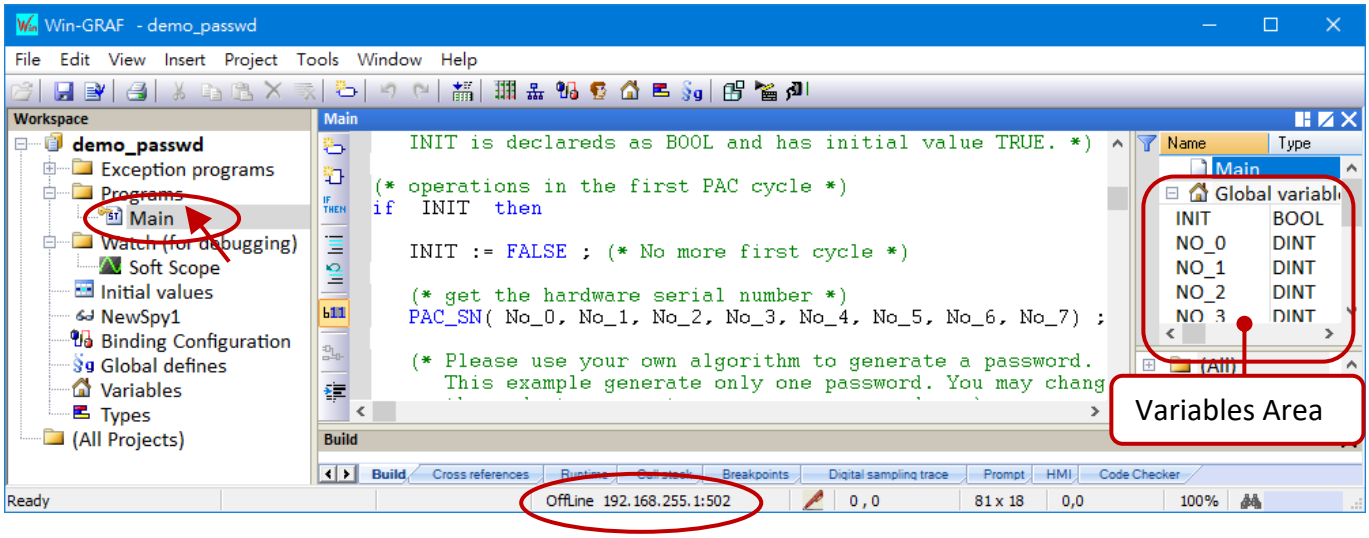
Each ICP DAS Win-GRAF PAC has a unique 64-bit serial number that can be used to combine with the custom algorithm to generate a password. Also, you can add the validation method to the project. When the project is downloaded to the PAC, it will check if the password is correct. If it is incorrect, the program will not be executed.

Description of demo programs:

1) "demo_passwd": Used to generate a password and save it as a file in the PAC.
2) "demo_my_ap": The project with the password verification function to give to the customer.

The user needs to download the "demo_passwd" project to the PAC and run it once to generate a unique password. Then, download the "demo_my_ap" project to the same PAC. After that, even if the Win-GRAF project is copied to the other PAC with the same model, it will not work due to the password validation failure.

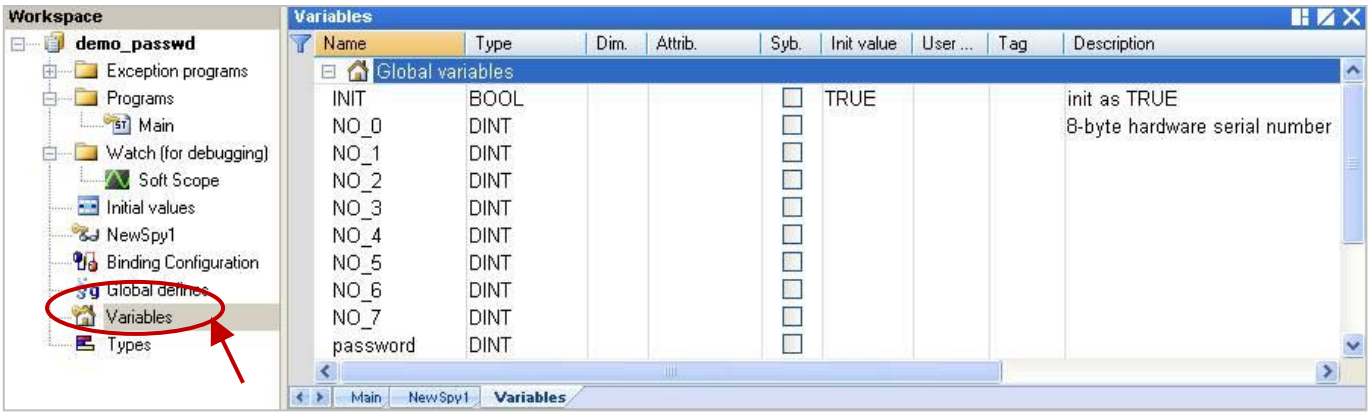
Download the demo program (**demo_passwd.zip** and **demo_my_ap.zip**) on the website. Click the menu commands "**File - Add Existing Project - From Zip...**" in the Workbench to open the project.



Win-GRAF Project: "demo_passwd"

First, using the "PAC_SN" function to read the serial number of the PAC and add the custom algorithm to generate a password, and then store the password in a file. The user can specify the file location.

Variable Declaration:

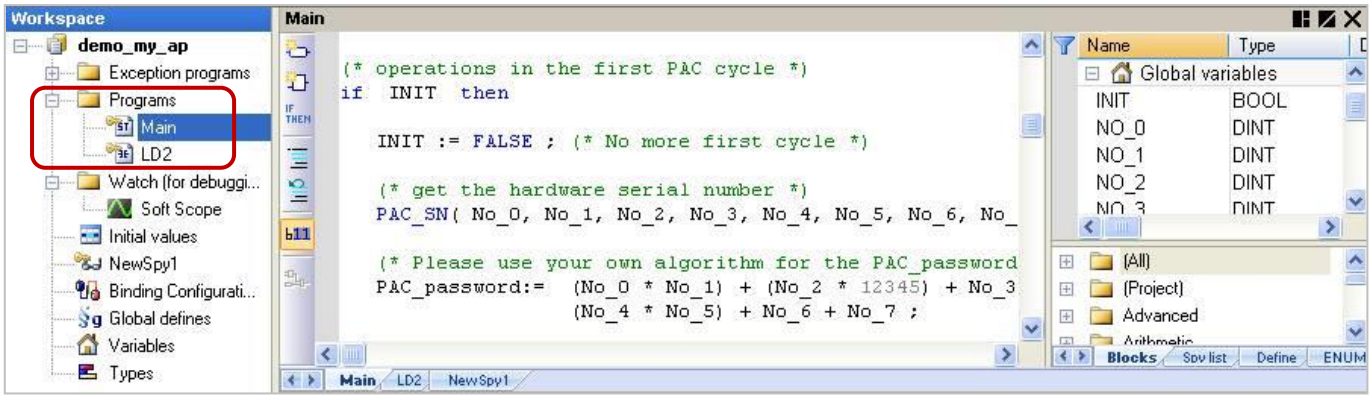


ST Program - Main:

```
(*  This "demo_passwd" project will generate a password by the 8-Byte Serial Number of the PAC
    and save it in the /System_disk/Win-GRAF/my_product.pwd on the PAC *)

(*  Declare  "No_0" ~ "No_7" and "password"  variables as  DINT.
     Declare  "INIT" variable as BOOL and has Initial value TRUE. *)

(* Operations in the first PAC Cycle *)
if  INIT  then
     INIT := FALSE ; (* No more first cycle *)

  (* Get the hardware serial number *)
  PAC_SN( No_0, No_1, No_2, No_3, No_4, No_5, No_6, No_7) ;

 (*  Please use your own algorithm to generate a password. This example generate only one
     password. You may change it to generate some passwords.  *)
  password := (No_0 * No_1) + (No_2 * 12345) + No_3 +  (No_4 * No_5) + No_6 + No_7 ;

  (* save the password in a file "my_product.pwd" in the /System_Disk/Win-GRAF *)
  file_name := '/System_Disk/Win-GRAF/my_product.pwd' ;
  file_id := f_wopen( file_name ) ;
  if  file_id = 0  then
    (* failed , do nothing *)
  else
    (* open file ok, save the password into it *)
    fm_write( file_id , Any_to_String(password) ) ;
    f_close( file_id );  (* close file *)
  end_if ;
end_if ;
```

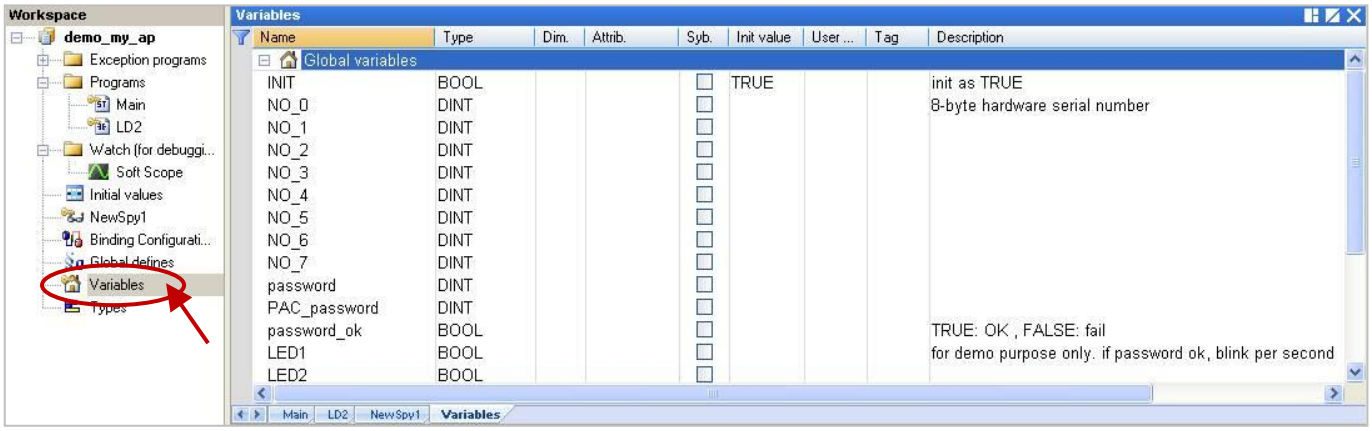
Win-GRAF Project: "demo_my_ap"

First, using the same way as described before to generate a password and compare the password that is stored in the file on the PAC. If the password is incorrect, quit the whole program.



Note: Refer to Section 2.1.2 to arrange programs in the execution order.

Variable Declaration:



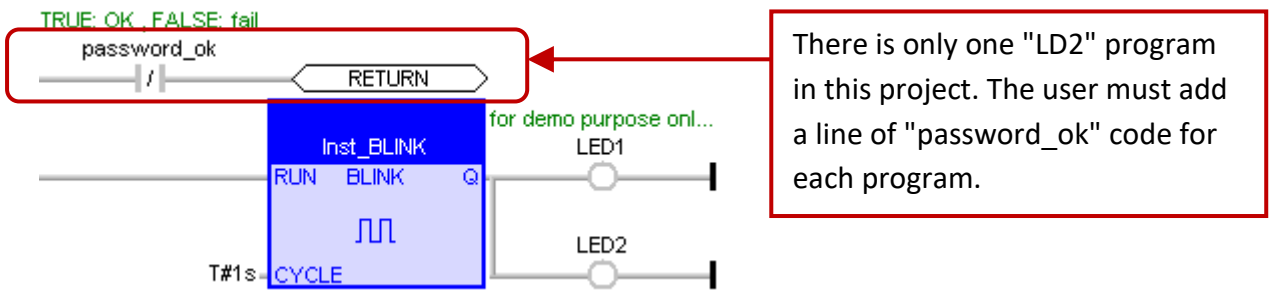
ST Program - Main:

```
(*  This "demo_my_ap" example can read the password in the /System_disk/Win-GRAF/
    my_product.pwd on the PAC, and check if match with the result that calculated from the user's
    own algorithm.  *)

(*  Declare  "No_0" ~ "No_7", "password" and "PAC_password" variables as  DINT.
    Declare  "INIT" variable as BOOL and has initial value TRUE.
    Declare  "password_ok" variable as BOOL  *)

(*  Operations in the first PAC cycle  *)
if  INIT  then
     INIT := FALSE ;  (*  No more first cycle  *)

  (*  get the hardware serial number  *)
  PAC_SN( No_0, No_1, No_2, No_3, No_4, No_5, No_6, No_7) ;
```

```
  (*  Please use your own algorithm for the "PAC_password" value  *)
 PAC_password:=  (No_0 * No_1) + (No_2 * 12345) + No_3 + (No_4 * No_5) + No_6 + No_7 ;

  (*  Read the password value from a file "my_product.pwd" in /System_Disk/Win-GRAF  *)
  file_name := '/System_disk/Win-GRAF/my_product.pwd' ;
  file_id := f_ropen( file_name ) ;
  if  file_id = 0  then
    (* can not open file, set password to 0 *)
    password := 0 ;
  else
    (* open file ok, read the password *)
    if  f_eof( file_id )  then
      (* reach the end of file *)
    else
      (* hasn't reached the end of file , read a string form it *)
      Tmp_string := fm_read( file_id ) ;
      (* Convert a string to a DINT value *)
      password := Any_to_DINT(Tmp_string) ;
    end_if ;
    f_close(file_id) ;  (* close file *)
  end_if ;

  (*  check if the password is correct?  *)
  password_ok := FALSE ;  (*  set it as "FALSE" in the beginning *)
  if  password = PAC_password  then
    password_ok := TRUE ;   (*  the password is correct  *)
  end_if ;
end_if ;
```

LD Program – LD2

If the "password_ok" is “FALSE“ which means the password is incorrect, it will quit the program. The program can be executed only when the password is correct so that protects your application from unauthorized use by others.



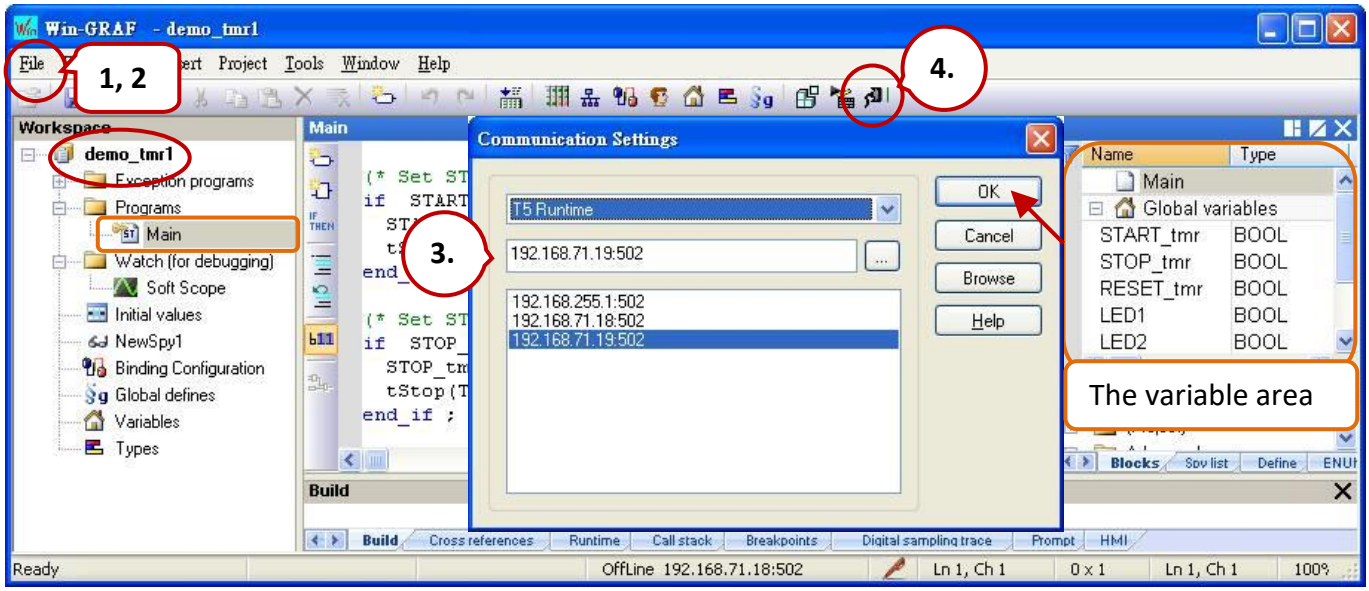
There is only one "LD2" program in this project. The user must add a line of "password_ok" code for each program.

Chapter 12 Description of Win-GRAF Demo Projects

The user can download Win-GRAF demo programs on the website:

Follow the steps below to execute the demo program:

1. Click the menu commands "**File - Close Project List**" to close all open projects.
2. Click the menu command "**File - Add Existing Project - From Zip**" to restore a demo program (e.g., "demo_tmr1.zip"). Also, refer to Section 11.4 to back up or restore the project.
3. Right-click the project name and select "**Communication Parameters**" to set the IP address of a PAC.
4. Click the menu command "**Project - On Line**" to download the project to the PAC. (Refer to Section2.3.4).



Before downloading the demo program to the PAC, make sure these settings are correct:

1. Win-GRAF communication IP, Modbus TCP Slave IP.
   Default IP addresses of RPAC - LAN1: 192.168.255.1, LAN2: 172.16.255.1, LAN3: 10.0.255.1
2. The setting of COM port or baud rate.
   COM1: RS-232, COM2: RS-485, COM3: RS-422
3. File location (e.g., /mnt/microSD/ or /System_Disk/Win-GRAF/).

Users can refer to Section 12.1 to modify parameters properly in the following window.

Main	LD Program		I/O Boards
Main	ST Program		I/O Drivers
Variables	Variables		Binding

12.1 The List of Demo Programs

Follow the steps as noted above to restore the demo program to the Win-GRAF Workbench.

File Name	Description	Chapter
win-graf-linux-demo-all.zip : Download all Win-GRAF demo programs. (674 KB)		
demo_et7060	Connect the remote ET-7060 Ethernet I/O module. I/O Drivers: Modbus Slave IP (10.0.255.2:502)	5.2.2
demo_et7018z	Connect the remote ET-7018Z Ethernet I/O module. I/O Drivers: Modbus Slave IP (10.0.255.2:502)	5.2.3
demo_tgw725	To access multiple Modbus RTU Slave via tGW-700 Gateway I/O Drivers: Modbus Slave IP (Port1): 10.0.255.2:502 Modbus Slave IP (Port2): 10.0.255.2:503	5.4
demo_retain	Use retain variables to store last data before a PAC power failure.	6.1
demo_wp5_retain	Store retain data to a file. Variables: File_Path1 ('/System_Disk/Win-GRAF/retain_real.txt') Variables: File_Path2 ('/System_Disk/Win-GRAF/retain_other.txt')	6.2
demo_binding	Exchange data between PACs (Data Binding) Binding: Exern (192.168.255.1/10.0.255.1)	7
Remote DCON I/O Module (8.2.1 ~ 8.2.8) **I/O Boards:** DCON (Port = 2, Baud rate = 9600)		
demo_d_7065	Connect the remote I-7065 I/O module. Function Block: D_7065 (Port = 2, Address = 2)	8.2.1
demo_d_7018z	Connect the remote I-7018Z I/O module. Function Block: D_7018Z (Port = 2, Address = 3)	8.2.2
demo_d_7083	Connect the remote I-7083 I/O module. Function Block: D_7083 (Port = 2, Address = 4)	8.2.3
demo_d_87084_fr	Connect the remote I-87084W module to measure frequency. Function Block: D_7084_FREQ (Port = 2, Address = 5)	8.2.4
demo_d_87084_c4	Connect the remote I-87084W module to measure the Counter. Function Block: D_7084_CNT4 (Port = 2, Address = 5) Function Block: D_7084_CNT8 (Port = 2, Address = 5)	8.2.5
demo_d_87084_c8		8.2.6
demo_dl_100T485	Connect the remote DL-100T485 (temperature/humidity) module. Function Block: DL_100T485 (Port = 2, Address = 1)	8.2.7
dmeo_gps721	Connect the remote GPS-721 GPS Receiver module Function Block: D_GPS721 (Port = 2, Address = 1)	8.2.8
demo_my_ap demo_passwd	Protect your Win-GRAF program and avoid being embezzled. Main (ST): file_name ('/System_Disk/Win-GRAF/my_product.pwd')	11.12
demo_tmr1	Use the "tStart" and the "tStop" functions to operate the Timer.	12.2.1
demo_tmr2	To do periodic operations for the Timer.	12.2.2
demo_tmr3	To do periodic operations more accurately for the Timer.	12.2.2

demo_com_port1	Send a string by the COM Port. Variables: Port_number (1: RS-232), Main (ST): '9600,N,8,1’	12.3.1
demo_com_port2	Request/answer the device by the COM Port. Variables: Port_number (2: RS-485), Main (SFC): '9600,N,8,1’	12.3.2
demo_com_port3	Wait for data coming from the remote device to the COM Port. Variables: Port_number (2: RS-485), Main (ST): '9600,N,8,1’	12.3.3
demo_com_port4	Report data periodically to the remote device by the COM Port. Variables: Port_number (2: RS-485), Main (SFC): '19200,E,8,2’	12.3.4
demo_file1	Write data to a file on the PAC. Main (ST): '/System_Disk/Real_data1.txt'	12.4.1
demo_file2	Read data from a file on the PAC. Variables: File_path ('/System_Disk/Real_data2.txt')	12.4.2
demo_datalog	Data Logging Main (ST): **/mnt/microSD/**yyyy-mm/yyyy-mm-dd.csv	12.4.3
demo_vb03	Use the C program to read/write Win-GRAF variables.	13
demo_vb04		
✧ Redundant System. **i_redundancy**: Active_IP = 192.168.71.37, Mask=255.255.0.0 **Fixed LAN2 IP**: 199.193.195.17 (Main)/ 199.193.195.9 (Backup) **Custom security mechanism**: Refer to the **RDN_control** program		14
demo_rdn_1	Function Block: D_87064、D_87018Z、D_7065 (Port = 2, Address = 2、3、4) I/O Boards: DCON (Baud rate = 9600)、i_redundancy、i_redundancy_rs485	
demo_rdn_2	I/O Boards: i_redundancy	
demo_rdn_3	LAN3, ET-7050 I/O Drivers: Modbus Slave IP (10.0.255.1:502) I/O Boards: i_redundancy	
demo_rdn_4	LAN3, iDCS-8830 I/O Drivers: Modbus Slave IP (10.0.79.200/10.0.79.201) I/O Boards: i_redundancy	
demo_schedule	Schedule control. I/O Boards: Schedule (Password: 0) Binding: Control Variables (ID: 5001 to 5030)	15
demo_sms	Use the 2G/3G GSM Modem to send/receive the SMS message. Variables: Phone_Nb (‘0900629879')	16
demo_extra_port	Enable a serial port for connecting the Win-GRAF Workbench. Variables: file1_name ('/System_Disk/Win-GRAF/Extra_Ports.txt') ST1 (ST): 'COM2:115200,N,8,1'	C
demo_pid_simple	The PID and regulator applications.	-

12.2 Timer Operations

12.2.1 Start, Stop and Reset the Timer

Open the project ("**demo_tmr1.zip**") and view variables in the variable area. Refer to Chapter 12.

ST Program:

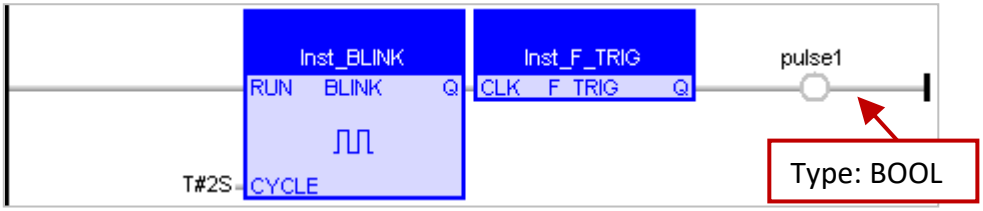
```
(* Declare “START_tmr”, “STOP_tmr”, “RESET_tmr”, “LED1”, “LED2” as BOOL
   Declare “TMR1” as TIME *)
(* Set START_tmr as TRUE to start ticking Timer TMR1 *)
 IF   START_tmr   THEN
   START_tmr := FALSE ;
   TSTART (TMR1) ;
END_IF;

(* Set STOP_tmr as TRUE to stop ticking Timer TMR1 *)
 IF   STOP_tmr   THEN
    STOP_tmr := FALSE ;
    TSTOP (TMR1) ;
 END_IF;

(* Set RESET_tmr as TRUE to reset TMR1 to a value T#0s *)
 IF   RESET_tmr   THEN
    RESET_tmr := FALSE ;
    TMR1 := T#0s ;
 END_IF;

(* Let LED1, LED2 ON during TMR1 = 3 ~ 10 second *)
 LED1 := FALSE ;
 LED2 := FALSE ;
 IF   (TMR1 >= T#3s) and (TMR1 <= T#10s) THEN
      LED1 := TRUE ;
      LED2 := TRUE ;
 END_IF;

(* Reset TMR1 as 0 when reachs 15 second *)
 IF   TMR1 >= T#15s THEN
      TMR1 := T#0s ;
 END_IF;
```

12.2.2 Periodic Operations

Open the project ("**demo_tmr2.zip**") and view variables in the variable area. Refer to Chapter 12.

In this example, using both the "**BLINK**" function and the "**F_TRIG**" function block generates a pulse TRUE at specific intervals. It can be applied to periodic operations.

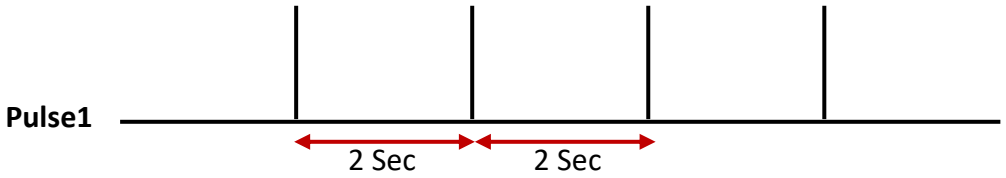
LD Program:



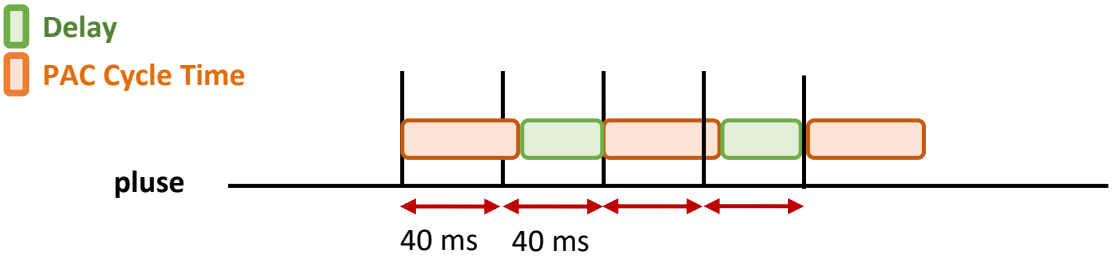
ST Program:

```
IF   pulse1   THEN
  (*  do periodic operations here *)
  • • •
END_IF;
```

The LD program above can be used to generate a pulse TRUE every two seconds. But, this method has a drawback:



When the time interval is shorter (e.g., less than 100 ms) or the PAC Cycle Time is larger (e.g., 20 to 50 ms, typically 3 to 15 ms), then the operating time will be inaccurate. For example, set it to execute the operation every 40 ms, as compared to 2 seconds or 250 ms, the interval time of 40 ms is very close to the PAC Cycle Time. Using this way is easy to accumulate the output delay time and the operating time finally becomes inaccurate.



To improve the situation, the following method of coding will be more accurate:

Open the project ("**demo_tmr3.zip**") and view variables in the variable area. Refer to Chapter 12.

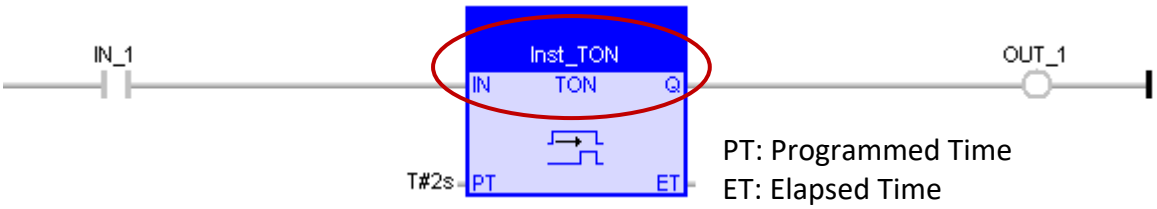
ST Program:

```
(* Declare "INIT"  as BOOL and has initial value  TRUE
   Declare "TMR1",  "TMR1_next" as TIME  *)

 IF   INIT   THEN
    INIT  :=  FALSE ;
    TMR1  :=  T#0s ;
   TMR1_next  :=  TMR1  +  T#50 ms ;
   TSTART (TMR1);
END_IF;

IF   TMR1  >=  TMR1_next   THEN
   IF   TMR1  >  T#10h  THEN
        TMR1  :=  T#0s ;
        TMR1_next  :=  T#0s ;
   END_IF;
   TMR1_next  :=  TMR1_next  +  T#50 ms ;
   (*  Do periodic operations here *)
   • • •
END_IF;
```

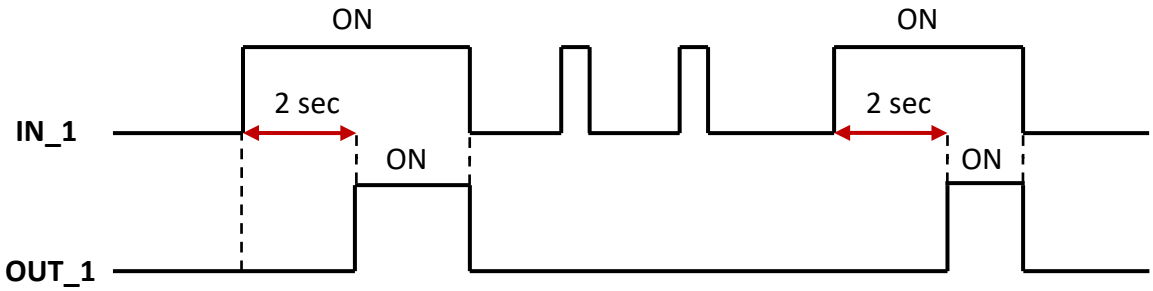
When the timer reach T#23h59m59s999ms, the value will overflow. Therefore, reset it automatically to "0" after 10 or 18 hours.

12.2.3 Detect the Steady ON or Steady OFF Signal

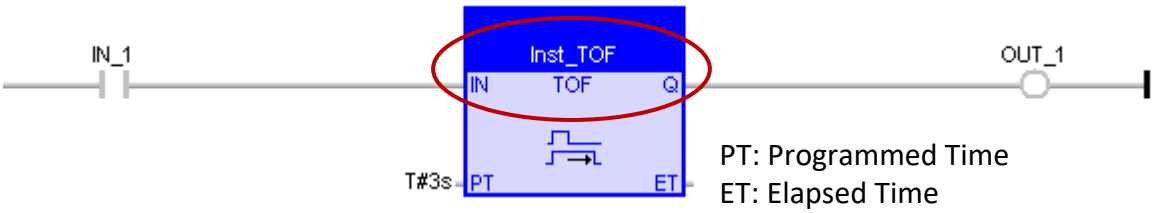
"TON" function block can detect a steady signal that remains "ON" for a specified time.



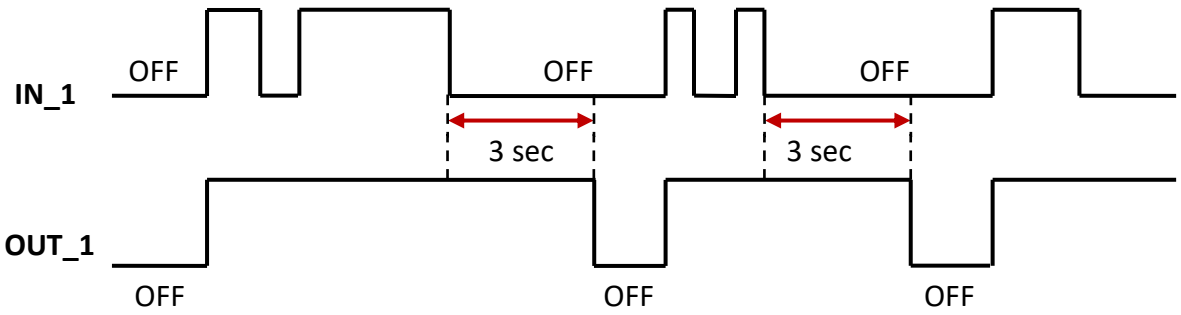
In the example, the function can detect a signal that remains "**ON**" for at least 2 seconds.



"TOF" function block can detect a steady signal that remains "**OFF**" for a specified time.

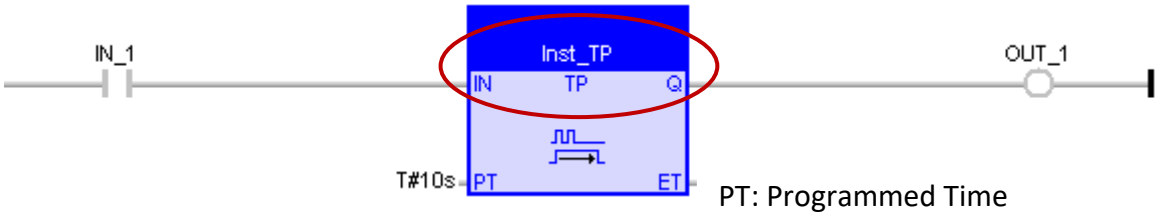


In the example, the function can detect a signal that remains "**OFF**" for at least 3 seconds.



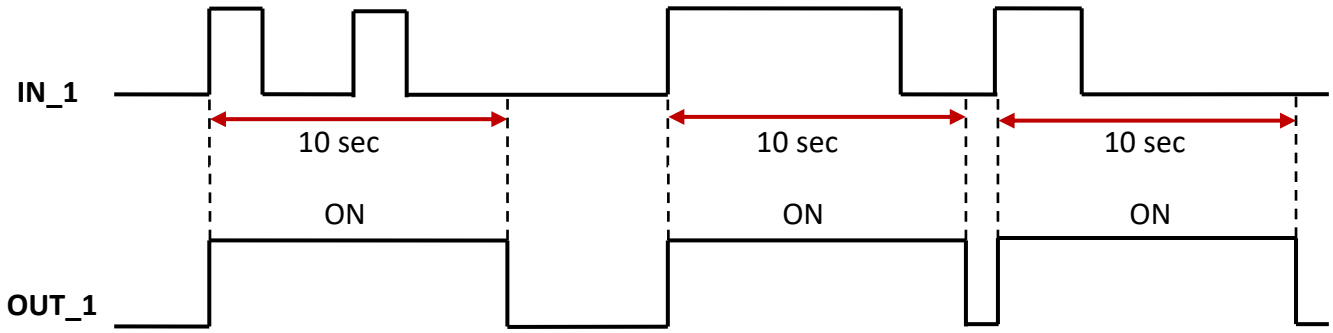
12.2.4 Keep Outputting ON for Some Time after Triggering

"TP" function block can keep outputting "**ON**" for a specified time after it is triggered (i.e., from OFF to ON).



PT: Programmed Time
ET: Elapsed Time

In the example, when the IN_1 is ON, it will keep outputting an "**ON**" signal in 10 seconds.

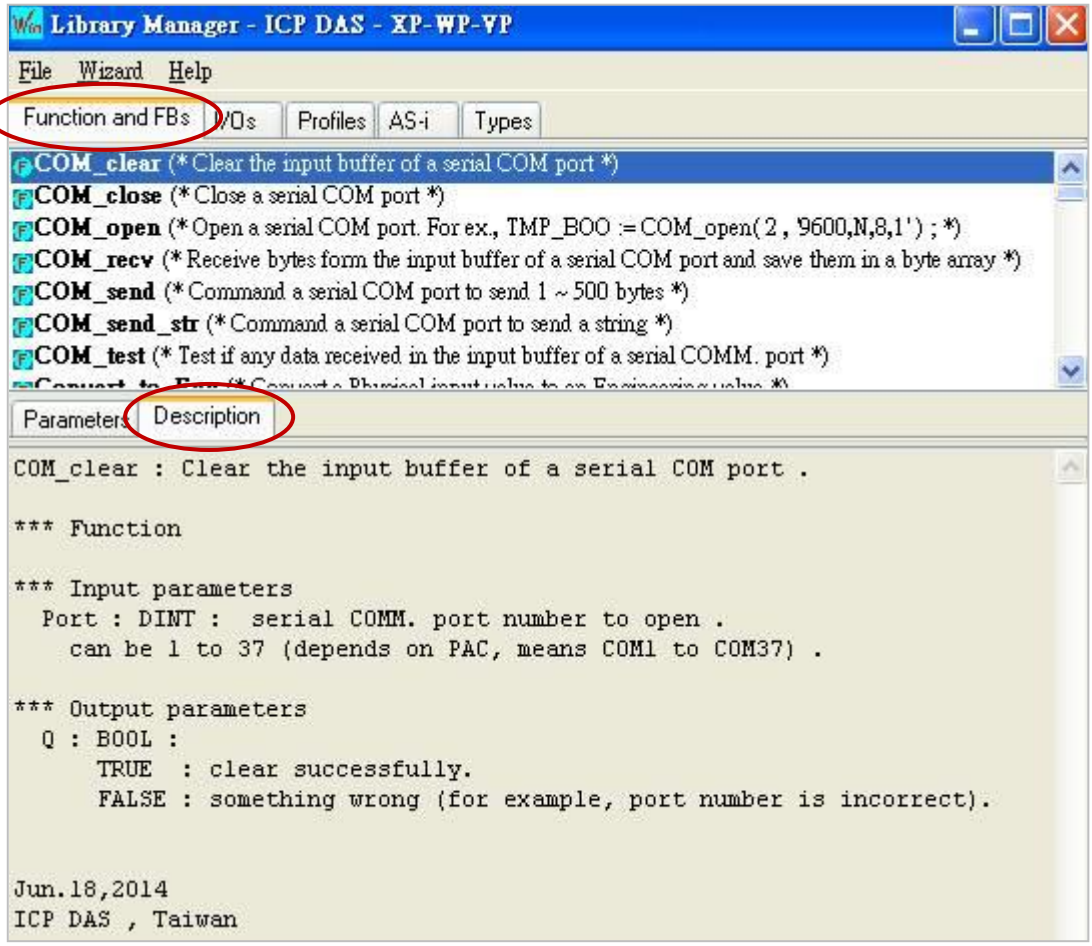


12.3 Operations of Serial Port Communication

Serial communication is often used either to control or to receive data, the user can use the following functions to directly perform specified operations to the serial port (e.g., RS-232, RS-485, or RS-422).

Functions	Description
COM_open	Open a serial port.
COM_close	Close a serial port.
COM_clear	Clear the input buffer of a serial port.
COM_test	Test if any Byte data received by a serial port?
COM_send	Send Byte data to a serial port.
COM_send_str	Send String data to a serial port.
COM_recv	Receive Bytes data from the input buffer of a serial port and save them in a Byte Array.
COM_status	Get the current status of a serial port.

Refer to Section 1.2.3 to open the Library Manager and look up the description of functions.



12.3.1 Send a String via a COM Port

Open the project ("**demo_com_port1.zip**") and view variables in the variable area. Refer to Chapter 12.

ST Program:

Used to send a String (e.g., <CNT1=1>...<CNT1=100>) every 2 seconds through the COM1 (`9600,N,8,1') of the PAC.

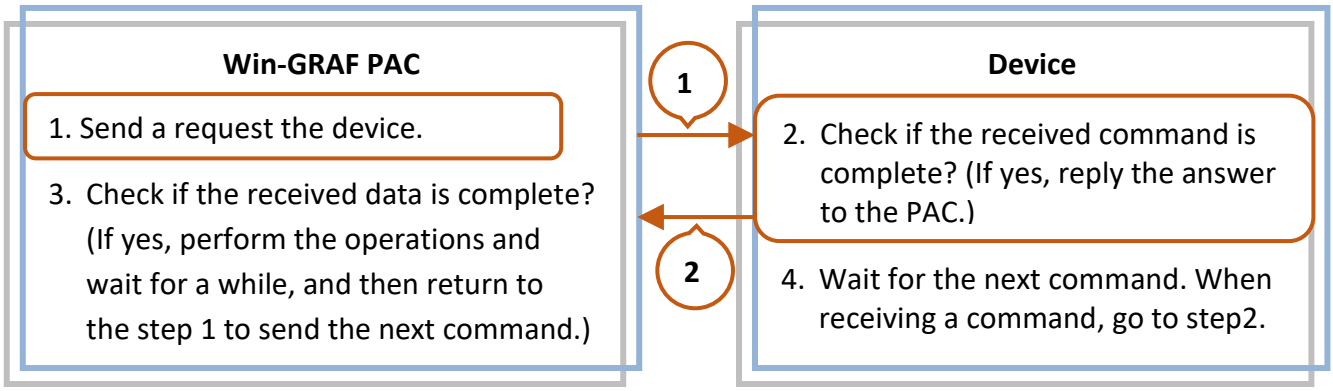
```
(*  Operations in the first PAC cycle  *)
if  INIT  then
  INIT := FALSE ;  (*  No more first cycle  *)
  CNT1 := 0 ;
  TMR1 := T#0s ;
  TMR1_next := TMR1 + T#2s ;
  (* start ticking TMR1 *)
  tStart(TMR1) ;
end_if ;
```

```
Declare  "INIT"  as BOOL and has initial value TRUE;
         "Port_OK"  as BOOL  ;
          "CNT1"  as DINT  ;
         "TMR1",  "TMR1_next" as TIME
          "Port_number" as  DINT  and has initial value "1"
```

```
  (* if the status of COM port becomes FALSE(not open), open it *)
if  COM_Status(Port_number) = FALSE  then
  (* open a serial COM port *)
  Port_OK := COM_open(Port_number, '9600,N,8,1' ) ;
end_if ;

  (* when time reached , ... *)
if  TMR1 >= TMR1_next  then

  (* to prevent TMR1 overflow (means reach T#23h59m59s999ms) *)
  if  TMR1 > T#10h  then
    TMR1 := T#0s ;
    TMR1_next := T#0s ;
  end_if ;

  (* Set new TMR1_next *)
  TMR1_next := TMR1_next + T#2s ;

  (* Send a string from COM port *)
  COM_send_str( Port_number, '<CNT1=' + Any_to_STRING(CNT1) + '>' ) ;

  (* reset CNT1 when reach 100 *)
  CNT1 := CNT1 + 1 ;
  if  CNT1 >= 100  then
    CNT1 := 0 ;
  end_if ;
end_if;
```

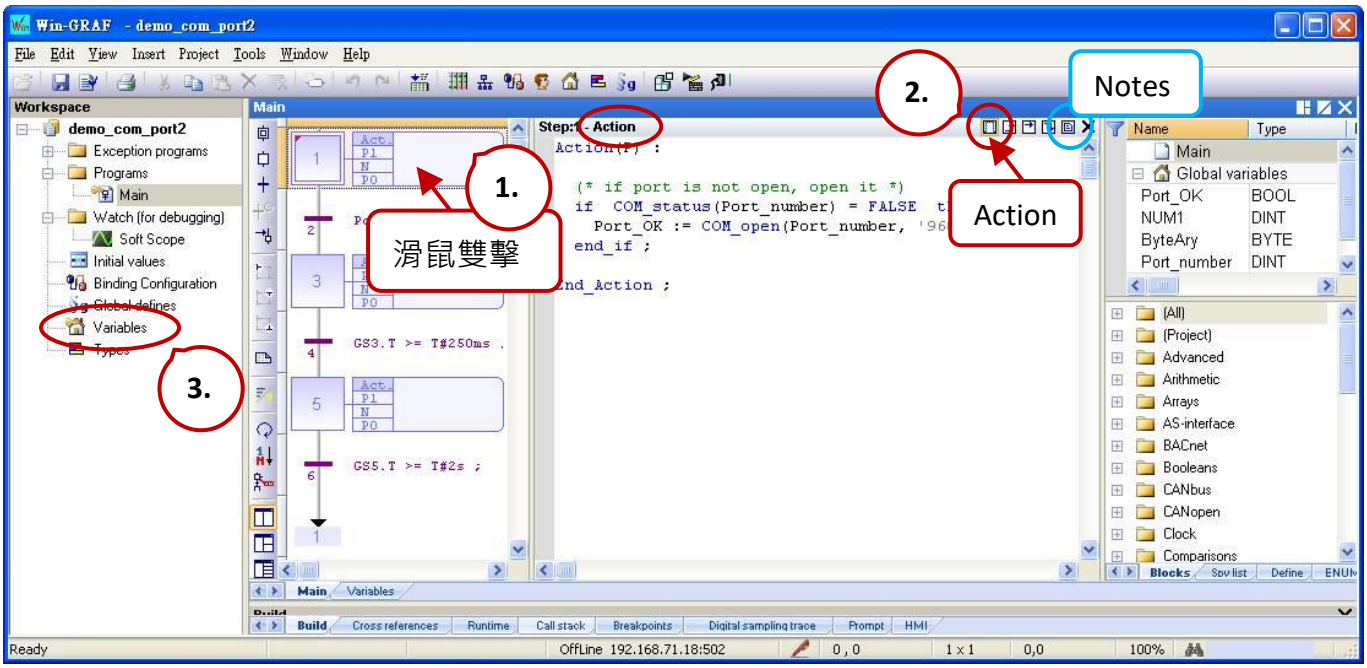
12.3.2 Request-Reply Communication via a COM Port

If it needs to get data from a device through an RS-232/RS-485/RS-422 port, using the Request-Reply communication as follows:

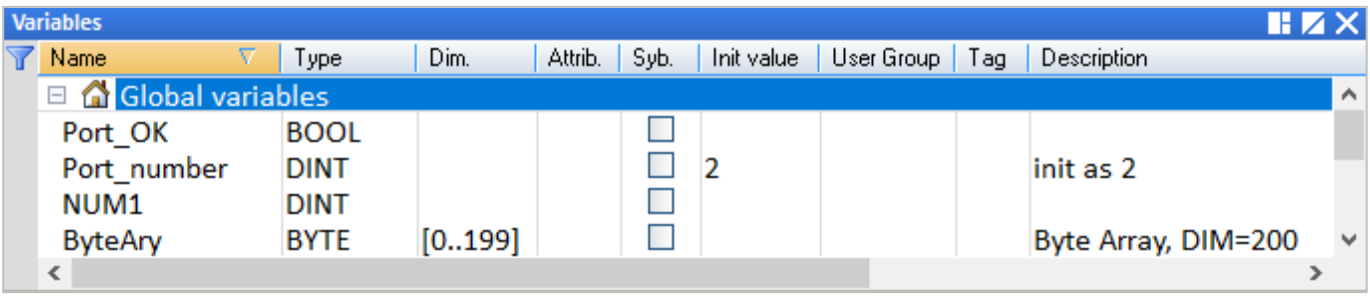


Open the project ("**demo_com_port2.zip**") and view variables in the variable area.

1. Double-click an **Action** to open the **Notes** window and change it to the “Action” window to view the code.



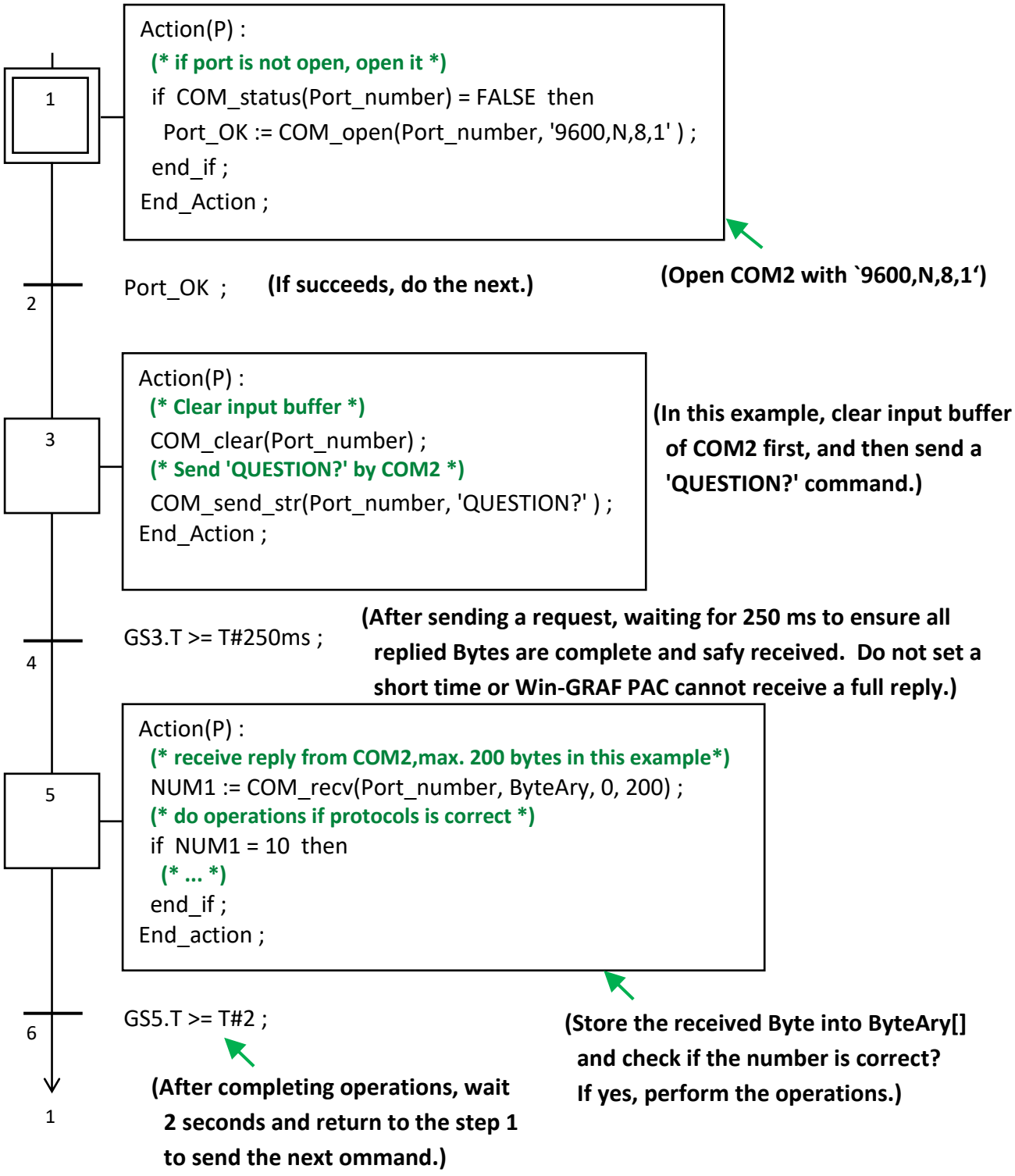
2. Also, click the “Variables” tag to open the Variables window.



In this example, the Win-GRAF PAC will send a 'QUESTION?' string to the device through the COM2 and waits for a complete reply before processing. After completing operations, wait for 2 seconds to send the next command, and repeats the procedure.

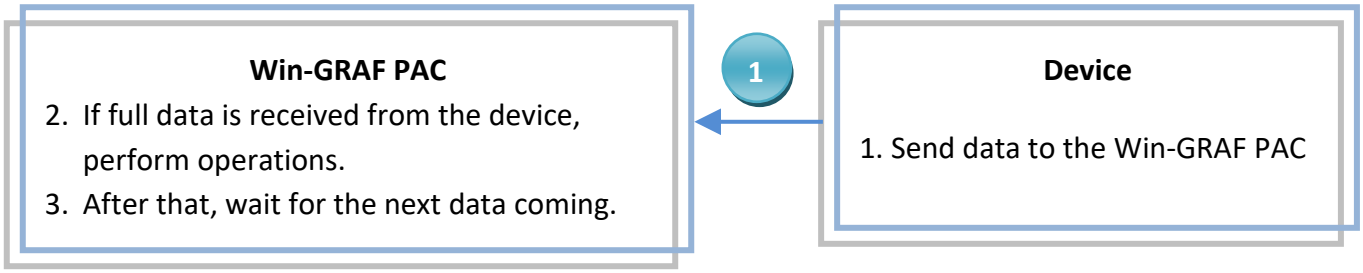
SFC Program:

(Declare "Port_OK" as BOOL ; "NUM1" as DINT ; "ByteAry" as BYTE and Dim. = "200" ;
"Port_number" as DINT and has an initial value "2")



12.3.3 Wait for a Remote Device to Send Data to a COM Port

This way is commonly used in convenience stores or supermarkets like the use of barcode reader. After reading the product barcode, it will send data to a COM Port (RS-232/485/422) of a Win-GRAF PAC, and no response is required.



Open the project ("**demo_com_port3.zip**") and view variables in the variable area. Refer to Chapter 12.

ST Program:

```
(* operations in first PAC cycle *)
if  INIT  then
  INIT := FALSE ;
  T1 := T#0s ;
  STEP1 := 0 ;
end_if ;
```

Declare "INIT" as BOOL and has initial value TRUE;
"Port_OK" as BOOL ;
"STEP1", "NUM1" as DINT ;
"T1" as TIME ;
"ByteAry" as BYTE and Dim. = "200" ;
“Port_number” as DINT and has initial value "2".

```
(* if port is not open, open it *)
if  COM_status(Port_number) = FALSE  then
  Port_OK := COM_open( Port_number , '9600,N,8,1' ) ;
end_if ;

(* If open port fail, exit this ST program *)
if  Port_OK = FALSE  then
  return ;
end_if ;

CASE  STEP1  OF

  (* if there is at least 1 byte coming *)
  0:
     if  COM_test(Port_number)  then
       STEP1 := 1 ;
       T1 := T#0s ;
       Tstart(T1) ;
     end_if ;
```

STEP1 = 0 means to wait and check if any data received by COM2? If returns TRUE means there is some data. Then set STEP1 to "1" and T1 to "0" to start timing.

```
  (* wait 250 ms, then receive all bytes by the COM port *)
1:
   if  T1 >= T#250ms  then
    Tstop(T1) ;
    T1 := T#0s ;
    STEP1 := 0 ;
```

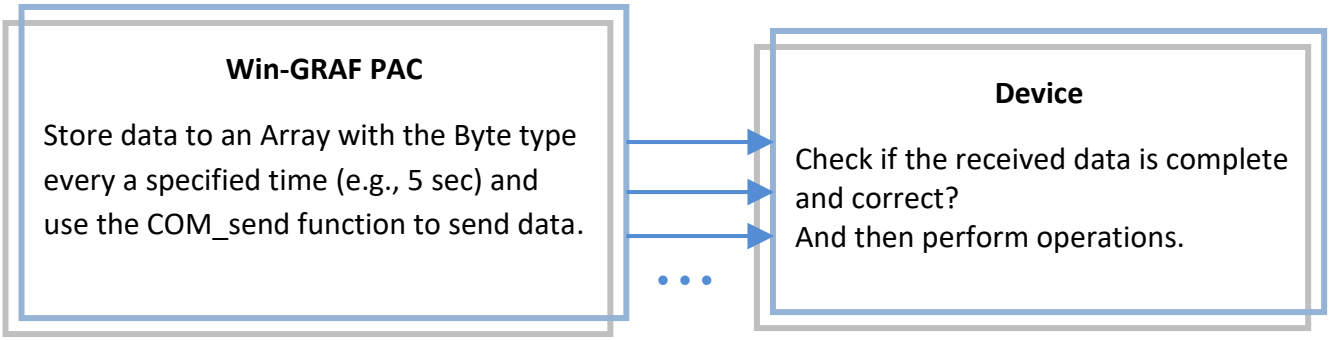
STEP = 1 means data is comeing, waits for 250 ms to receive fulll data and store them into an array. The waiting time concerns the specifications and Baud Rate of the device, set a short time may causes data lost. **Remember to set STEP1 to "0" to wait for next data coming.**

```
    (* receive max. 200 bytes *)
    NUM1 := COM_recv(Port_number , ByteAry , 0 , 200 ) ;

    (* do proper operations if protocol is correct ,
       here assume correct protocols has 25 bytes in this example*)
    if  NUM1 = 25  then
     (* ... *)
    end_if ;

   end_if ;

END_CASE ;
```

After receiveing full data, check if data is correct? If yes, perform the operations.

12.3.4 Reply Data to the Remote Device Periodically via a COM Port

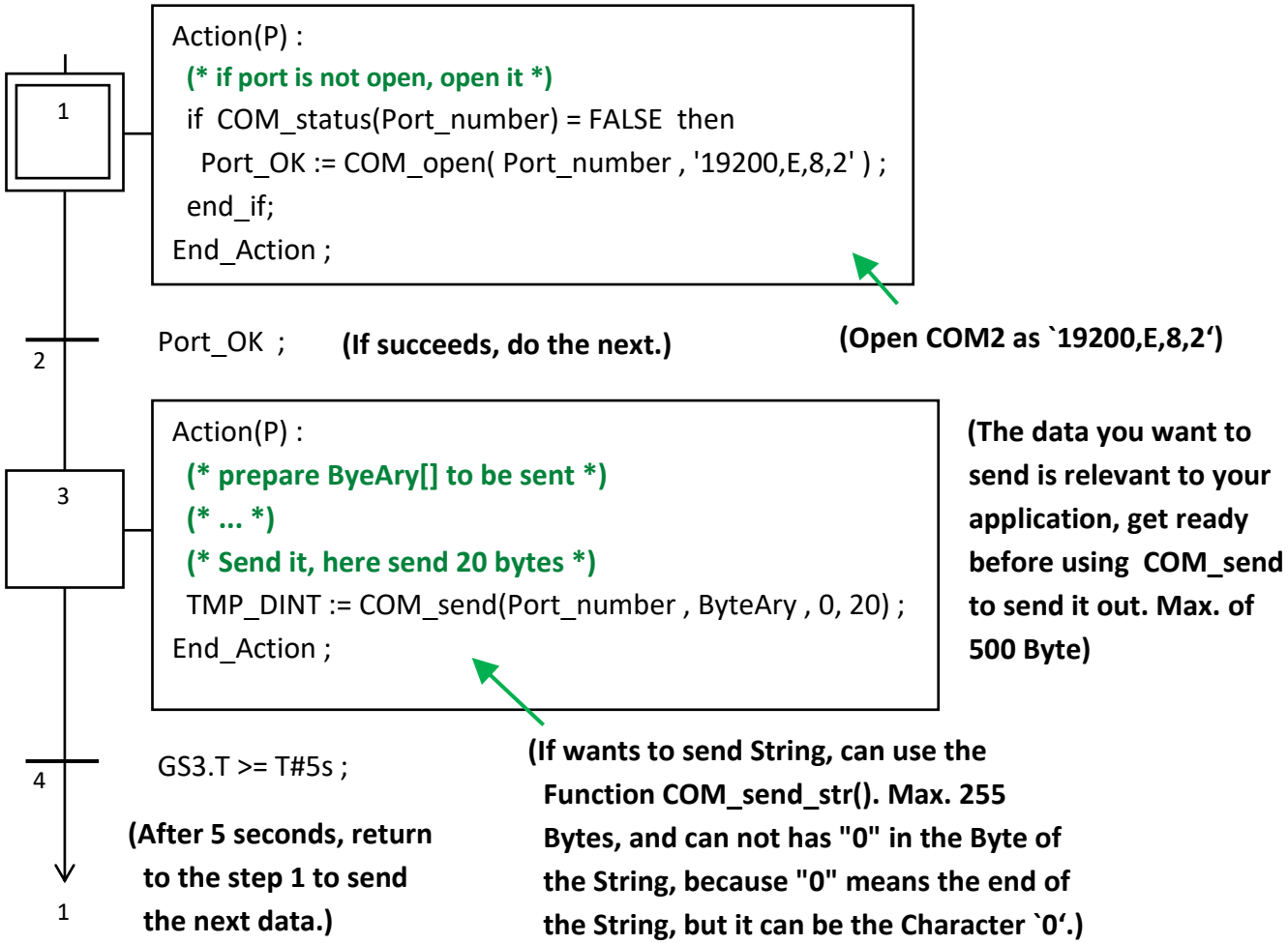
To periodically reply data to the remote device through an RS-232/RS-485/RS-422 port, the process of the method is as follows



Open the project ("**demo_com_port4.zip**") and view variables in the variable area. Refer to Chapter 12.

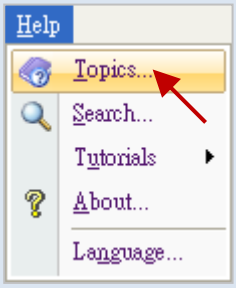
SFC Program:

(Declare "Port_OK" as BOOL ; "TMP_DINT" as DINT ; "ByteAry" as BYTE and Dim. = 100 ; "Port_number" as DINT and has an initial value "2".)



12.4 Read/Write Data from/to a File on the PAC

Win-GRAF Workbench provides the following functions to enable read/write operations in a file on the PAC. For more information, open the "HTML Help" window and search the word "**File**", and then select the **File Management functions** topic.

Functions	Descriptions
	
F_ROPEN	Open/Create a file for reading.
F_WOPEN	Open/Create a file for writing.
	If the file doesn't exist, it will be created automatically. If the file exists, all its content will be removed.
F_AOPEN	Create or open a file in append mode.
F_CLOSE	Close an open file.
F_EOF	Test if the end of the file is reached in a file open for reading.
FA_READ	Read a DINT integer from a binary file.
FA_WRITE	Write a DINT integer to a binary file.
FM_READ	Read a STRING value from a text file
FM_WRITE	Write a STRING value to a text file.
FB_READ	Read binary data from a file.
FB_WRITE	Write binary data to a file.
F_EXIST	Test if a file exists.
F_GETSIZE	Get the size of a file.
F_COPY	Copy a file.
F_DELETE	Remove a file.
F_RENAME	Rename a file.
Refer to Section 1.2.3 to look up the description of these functions.	
F_dir	Create a directory.
F_cp_dir	Copy a directory and all its contents to another directory.
F_del_dir	Delete a directory and all its contents.

Note: Win-GRAF Linux PAC does not support F_SAVERETAI and F_LOADERETAIN functions.

12.4.1 Write Data to a File on the PAC

Open the project ("**demo_file1.zip**") and view variables in the variable area. Refer to Chapter 12.

ST Program: This program can be used to write 10 "REAL" values to a file on the PAC.

```
(*  This "demo_file1" project will save 10 REAL value to a file in the  /System_Disk/Real_data1.txt .

      File Format :
        Each row contains one REAL value and ends with <CR><LF> characters.  Like:

        1.08
        2.786
        38.45
        41.5
        59.875
        60.76
        71.23
        80.5
        99.8
        100.7        *)

(*  Variables declaration:
    Write_File       : BOOL
    Tmp_string       : String, len=255
    File_ID          : DINT
    REAL_val[0..9] : REAL
    ii               : DINT
    File_Status      : String, len=128    *)

(* Set Write_File as TRUE to write data to the file *)
if  Write_File  then

  Write_File := FALSE ;
  File_ID := F_Wopen( '/System_Disk/Real_data1.txt' );

 if  File_ID = 0  then
    (* Can not open file in write mode *)
    File_Status := 'Can not open file in write mode !' ;
 else
     (* open file in write mode ok, save REAL[0] ~ [9] to file ,
           each row contains 1 REAL value and end with <CR><LF> *)
    File_Status := 'Open file ok.' ;
    for ii := 0 to 9  by 1  do
```

Because the capacity of /System_Disk/ is small; you can change the file path to **/mnt/microSD/**. Also, the SD Card must be installed in the PAC.

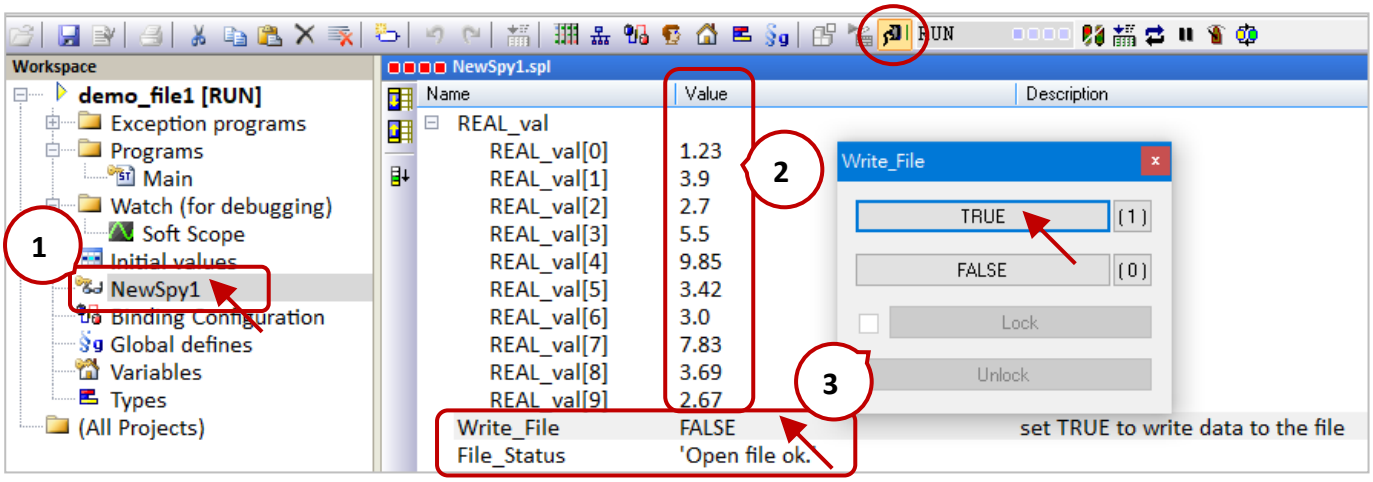
```
    Tmp_string := Any_to_string( REAL_val[ii] ) ;
    FM_write( File_ID , Tmp_string ) ;
  end_for ;

  (* close the file *)
  F_close( File_ID ) ;

 end_if ;
end_if ;
```

To write Integers by using the code below:
Tmp_string := Any_to_string(**DINT_val[ii]**) ;
and declare the "DINT_val" variable as DINT and the Dim. at least "10"

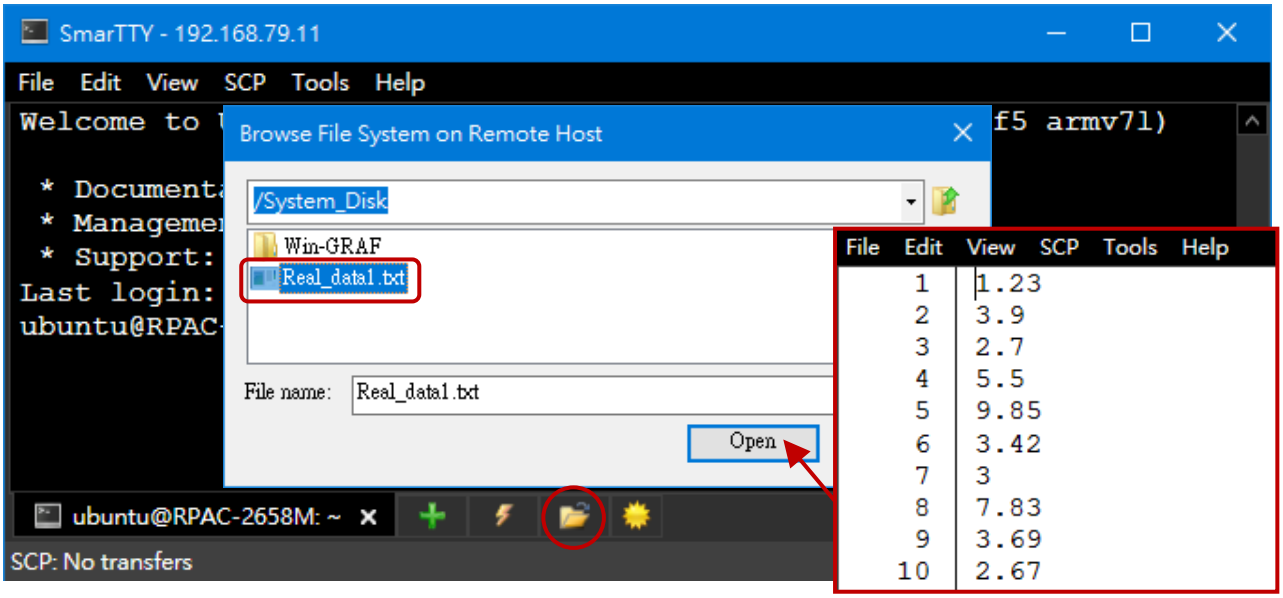
Test Program:

In this example, when the "Write_File" is set to "TRUE", it will write data to the **/System_Disk/ Real_data1.txt** on the PAC.

1. After downloading the project to the PAC, click "NewSpy1" to open a Spy List and fill in the values to be written, and then set the "Write_File" to "TRUE" to Write data. (If OK, "File_Status" will show "Open file ok".)



2. Find out the "Real_data1.txt" file on the PAC by using SSH (refer to Section 13.2).



12.4.2 Read Data from a File on the PAC

Open the project ("**demo_file2.zip**") and view variables in the variable area. Refer to Chapter 12.

ST Program: This program can be used to read 10 "REAL" values from a file on the PAC.

```
(*  this "demo_file2" project will read 10 REAL value from a file in the /System_Disk/Real_data2.txt .
    File format :
        Each row contains one REAL value and ends with <CR><LF> characters. Like:
        1.08
        2.786
        38.45
        41.5
        59.875
        60.76
        71.23
        80.5
        99.8
        100.7
*)

(*
  Variables Declaration:
  Write_File       : BOOL
  Tmp_string       : String, len=255
  File_ID          : DINT
  REAL_val[0..9] : REAL
  ii               : DINT
  File_path        : String, len = 128, initial val = '/System_Disk/Real_data2.txt'
  File_Status      : String, len=128
*)

(* Set Read_File as TRUE to read data from the file *)
if  Read_File  then

   Read_File := FALSE ;
   (* Check if file exists *)
   if  F_exist( File_path ) = FALSE  then
      (* file doesn't exist *)
      File_Status := 'File "' + File_path +'" does not exist !' ;
   else

     (* file does exist , open it in read mode *)
      File_ID := F_Ropen( File_path );

      if  File_ID = 0  then
         (* open file in read mode fail *)
         File_status := 'Can not open File "' + File_path +'" !' ;
      else
```

Because the capacity of /System_Disk/ is small; you can change the file path to **/mnt/microSD/**. Also, the SD Card must be installed in the PAC.

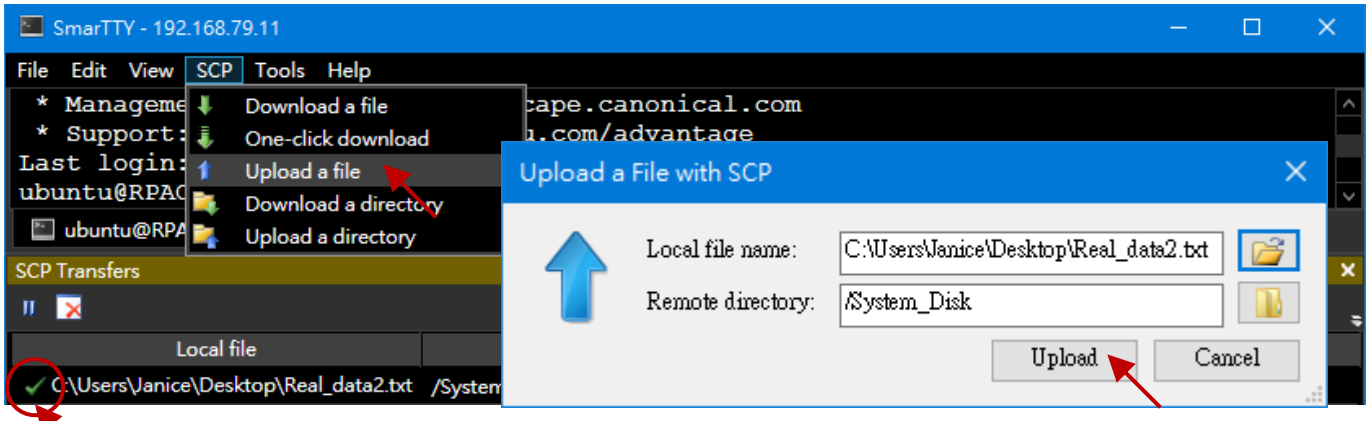
```
(* open file in read mode ok, read REAL[0] ~ [9] from file ,
    each row contains 1 REAL value and end with <CR><LF> *)
File_status := 'Open File "' + File_path +'" Ok .' ;
for ii := 0 to 9  by 1  do

  (*  test if the end of file is reached in a file open for read *)
  if  F_EOF( File_ID )  then
    (* reach the end of file, exit "for loop" *)
    exit ;
  end_if ;

  (* read one row in the file as a string *)
  Tmp_string := FM_READ( File_ID ) ;

  (* convert the string to become REAL value *)
  REAL_val[ii] := Any_to_REAL( Tmp_string ) ;

end_for ;

(* close the file *)
F_close( File_ID ) ;

  end_if ;
 end_if ;
end_if ;
```

To read Integers, use the code below:
DINT_val[ii] := Any_to_**DINT**(Tmp_string) ;
and declare the "DINT_val" variable as DINT and the Dim. at least "10" for this example.

Test Program:

In this example, when the "Read_File" is set to "TRUE", it will read data from the "**/System_Disk/ Real_data2.txt"** on the PAC.

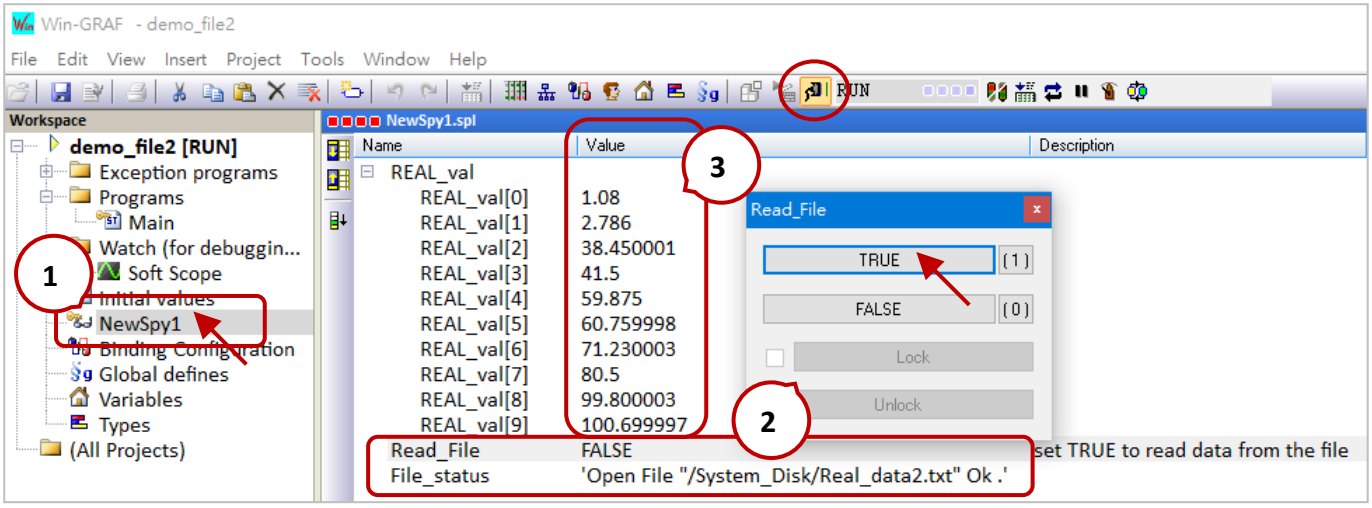
1. Upload the **Real_data2.txt** file to the **/System_Disk/** directory by using the SSH tool. (Refer to Section 13.2)



Upload successfully.

2. After downloading the project ("**demo_file2.zip**") to the PAC, click "NewSpy1" to open a Spy List and set the "Read_File" to "TRUE" to read the data.

   If OK, the read data will be displayed and "File_Status" will show 'Open File "/System_Disk/Real_data2.txt" Ok.'



Note:

- When using the Linux PAC, change the symbol of the path to **/** (slash).
- Also, refer to Section 6.2 to write multiple data to a file by using retain variables.

12.4.3 Data Logging

Refer to Chapter12 to open this project ("**demo_datalog.zip**"). The project provides a simple data logging function.

Description of the Project:

The project creates a spy list that contains a String variable (write_date), an Integer variable (int_data), and a Real variable (float_data). The variable data will be stored in a file per minute in the specified folder on the PAC.

The log file will be named according to the current date (e.g., Jan 02, 2021, the file name is "2021-1-2.csv"). Besides, the log file of the previous day will be moved to "/the specified folder/current month/" every day (e.g., Jan 01, 2021, the file will be moved to "**/mnt/microSD/**2021-1/").

Note: **I**n the example, the log file is stored in **/mnt/microSD/** on the PAC. The user needs to install an SD card in the PAC.

The sample of the CSV file format:

```
Time ,    int_val ,  float_val
20:18:30,  1236,   14.56
20:18:40,  3456,   34.56
20:18:50,  8932,   89.32
```

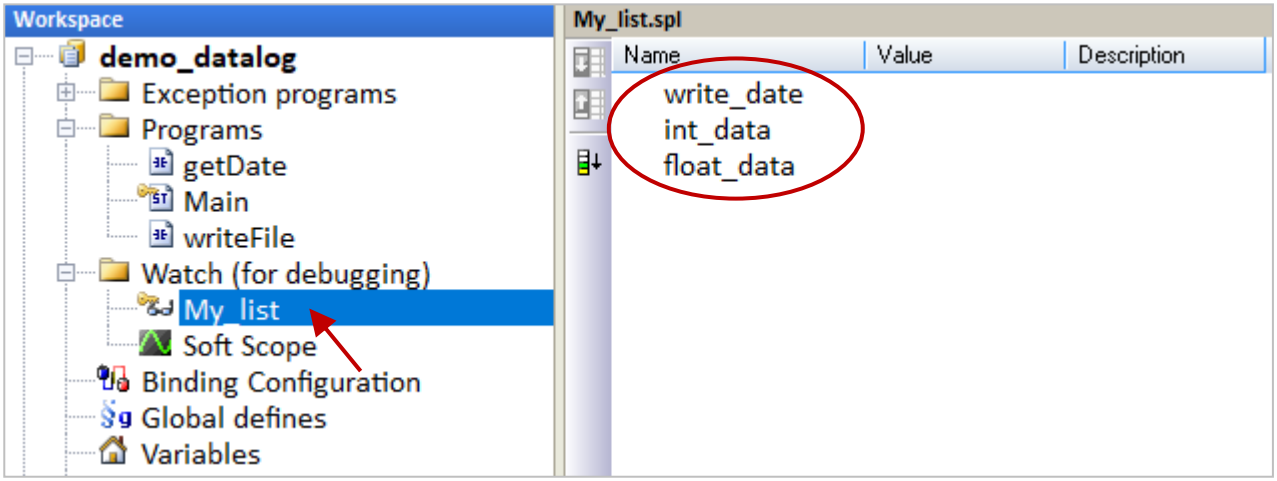
Description of variables: The user can view/set variables in the Win-GRAF "Variables" window.

Name	Data Type	Description
Year1	DINT	Used to get the PAC system time in the “PAC_Time” program.
Month1		
Day1		
WeekDay1		
Hour1		
Minute1		
Second1		
old_day		Used to know if the time is changed and then rename the log file in the “PAC_Time” program.
old_hour		
log_tmr1	TIME	Timer.
log_tmr2		
CSV_Path	STRING	The storage path of the CSV file.
CSV_Dir		The storage folder of the CSV file.
write_date		Used to record the time when writing data to a CSV file.
init	BOOL	Set it as TRUE to initialize. (Init value = TRUE)

Name	Data Type	Description
int_data	DINT	Used to record the data.
float_data	REAL	Used to record the data.
writecsv	BOOL	Set it as TRUE to write data.
File_ID	DINT	Used for the "F_WOPEN" function.
tmp_bval	BOOL	The temporary variable.
tmp_msg	STRING	

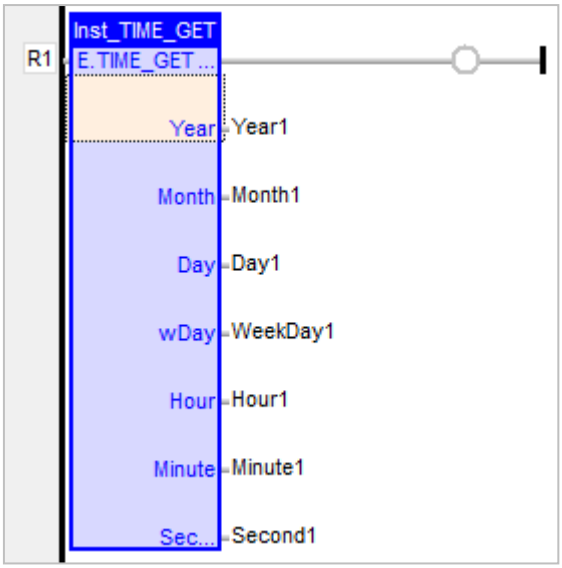
Spy List – "My_List":

This example creates a watch list called "My_list". (Refer to Section 11.3 for more details)



LD Program – "getDate":

To get/set the PAC system time.



Note: In the demo program, the file folder is **/mnt/microSD/**. The user needs to install an SD Card in the PAC.

ST Program – "Main":

To generate random values as the data of the log file (.csv) and to change the file storage path.

```
(* Declare "init" as "BOOL" ; Declare "log_tmr1" and "log_tmr2" as "TIME"  *)
(* Set "init" as TRUE to start ticking "TMR1"  *)
 IF    init    THEN
   init  :=  FALSE ;
     (* Enable the timer of the data logger *)
       TSTART (log_tmr1) ;
     (* Set the separator and decimal within the csv file  *)
      SetCsvOpt( ' , ' ,  ' . ' );
END_IF;

IF old_day<> day1 THEN
       tmp_msg:='/mnt/microSD/'+CSV_Dir+'/'+ANY_TO_STRING(year1)+'-'+
                  ANY_TO_STRING(month1)+'-'+ANY_TO_STRING(old_day)+'.csv';

     (* Copy the file to /mnt/microSD/ *)
       tmp_bval:=F_COPY (CSV_Path,tmp_msg);

     (* Delete the original file stored at /run/ *)
       tmp_bval:=F_DELETE(CSV_Path);

     (* Create the CSV file and its field name *)
      tmp_msg:='/run/'+any_to_string(year1)+'-'+any_to_string(month1)+'-'+
                   any_to_string(day1)+'.csv';
      CSV_Path:=tmp_msg;
      File_ID:= F_WOPEN(tmp_msg);

     IF File_ID<>0 THEN
          tmp_msg:='Time,Int_val,Float_val';
          tmp_bval:=FM_WRITE(File_ID,tmp_msg);
          tmp_bval:=F_CLOSE(File_ID);
      END_IF;

     old_day:=day1;
END_IF;
```

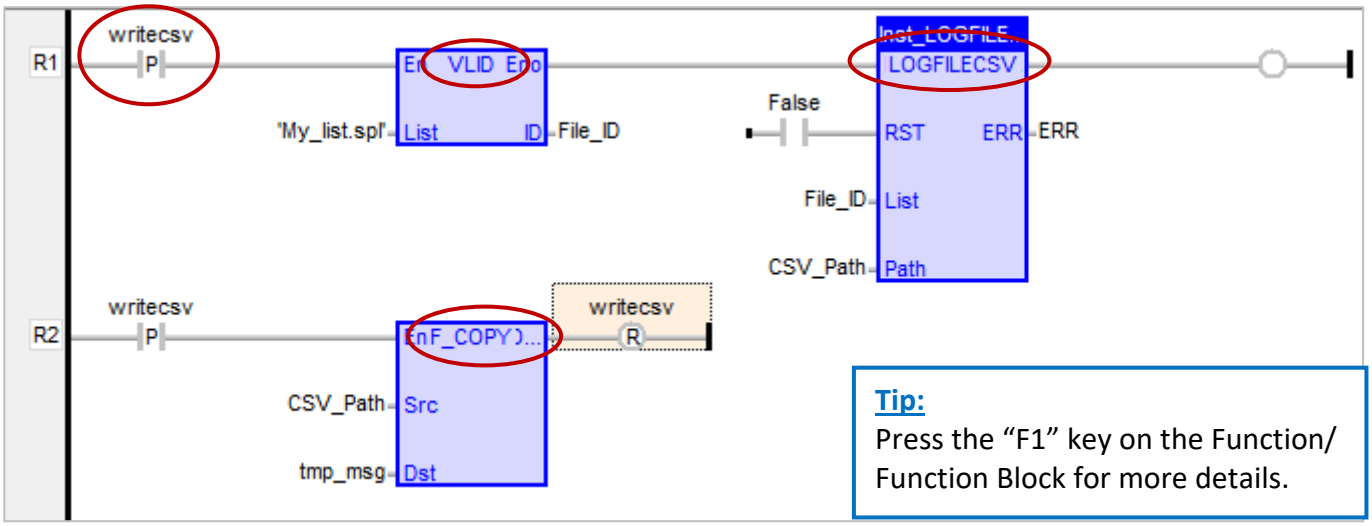
```
(* Create the folder every month, or when the PAC boot up. *)
if  old_month<>month1 then
      CSV_Dir:=ANY_TO_STRING(year1)+ANY_TO_STRING(month1);
      tmp_msg:='/mnt/microSD/'+CSV_Dir;
      F_DIR(tmp_msg);

end_if;

(* The function for writing data to a CSV file *)
IF  log_tmr1>=log_tmr2  THEN
   log_tmr1:=t#0s;

   (* Generate random values for the "Int_val" and the “Float_val" field *)
      int_data:=rand(1000);
      float_data:=ANY_TO_REAL(int_data)/100;

   (* The time for each data logging *)
     write_date:=ANY_TO_STRING(hour1)+':'+ ANY_TO_STRING (Minute1)+':'+
                 ANY_TO_STRING (Second1);

   (* Delete the previous file *)
     tmp_msg:='/mnt/microSD/'+CSV_Dir+'/'+ANY_TO_STRING(Year1) +'-'+
                 ANY_TO_STRING(month1)+'-'+ANY_TO_STRING(Day1)+'.csv';
```

Because the capacity of /System_Disk/ is small; you can change the file path to **/mnt/microSD/**. Also, the SD Card must be installed in the PAC.

```
   (* Delete the file stored at /mnt/microSD/ *)
      tmp_bval:=F_DELETE(tmp_msg);
   (* Trigging it to write data to a file *)
      writecsv:=true;
END_IF;
```

LD Program – “WriteFile”

Set the "writecsv" as "TRUE" to write one data to the CSV file.



VLID: Get the identifier of an embedded list of variables (i.e., Spy List - **My_list.spl**).

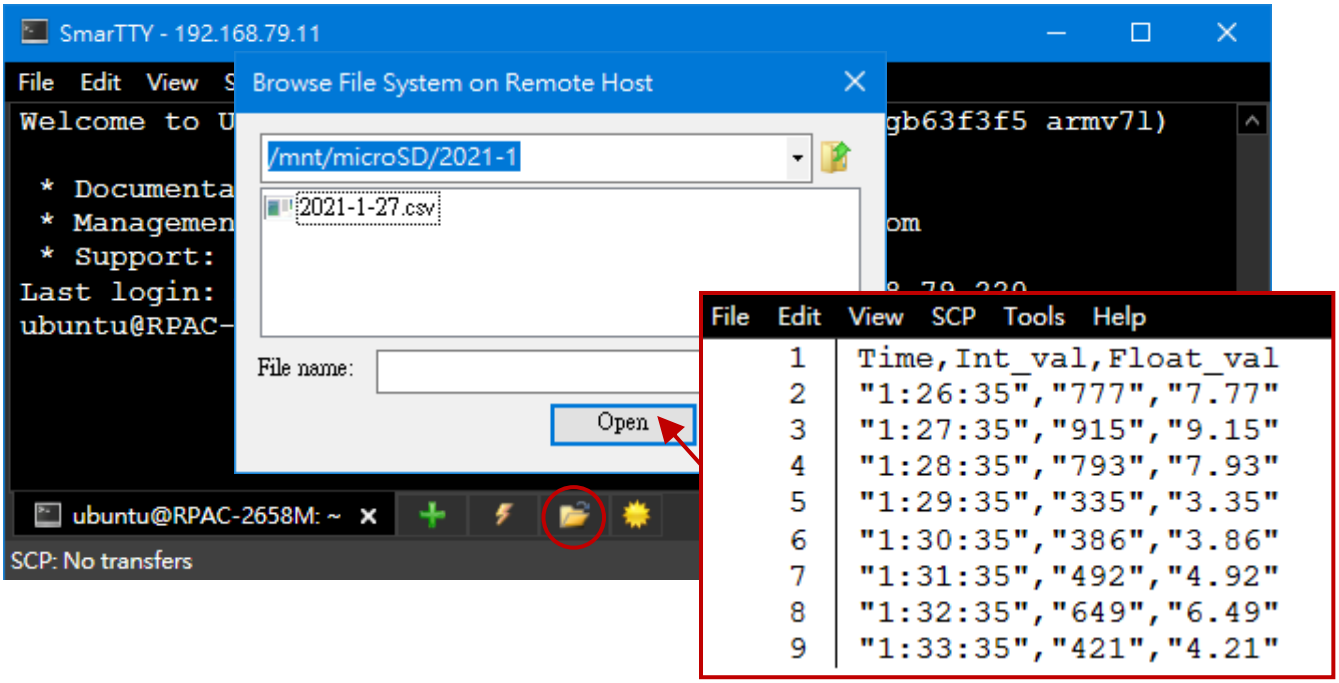
LogFileCSV: Generate a log file in CSV format for a list of variables.

F_COPY: Copy a file. Using “F_COPY” to copy a file from the “**/run/**” folder (RAM Disk, which can read/write data quickly) to “**/mnt/microSD/**”. (Refer to ST program – “Main”)

Test Program:

The write_date, int_data, and float_data variable data will be recorded and stored to **/mnt/microSD/** YYYY-MM/YYYY-MM-DD.csv (e.g., **/mnt/microSD/**2021-1/2021-1-27.csv).

Find out the .csv file on the PAC and check values by using SSH tools (refer to Section 13.2).



Chapter 13 Using a C Program to Access Win-GRAF Variables

The Chapter describes how to create public variables in the Win-GRAF project for the C program to read/write data. The C demo program is developed by using the GCC tool as well as two Win-GRAF projects are provided.

C demo program - Quicker_Demo.tgz
Used to read/write Win-GRAF Char, Short, Long, ULong, Int64, Float, Double, or String variable.

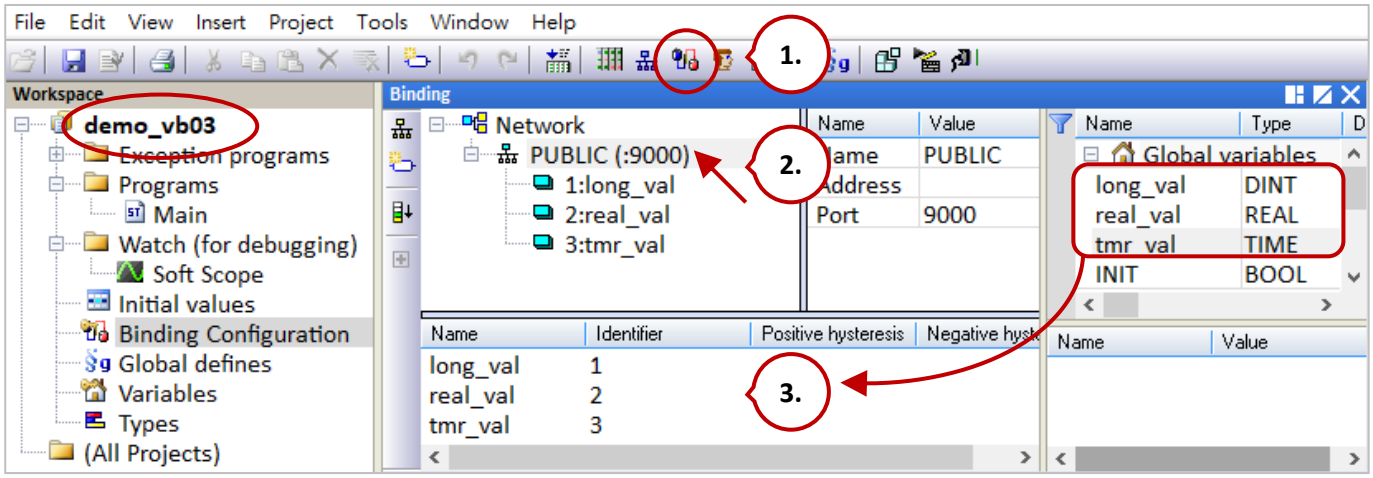
Win-GRAF demo programs - demo_vb03.zip and demo_vb04.zip
Download the demo program on the website and run the **File** menu commands **Add Existing Project** and then **From Zip...** in the Workbench to open the project.

13.1 Publishing the Win-GRAF Variable for C Program

In the Win-GRAF, except for the String variable, all the public variables and IDs must be set in the "Binding" window so that the C program can access data with the same address.

Refer to Chapter 7 to create public variables. After completing the settings, the results are shown below.

Note: Up to 8192 public variables can be created and the range of ID is 1 to 8192.



ST program (Main):

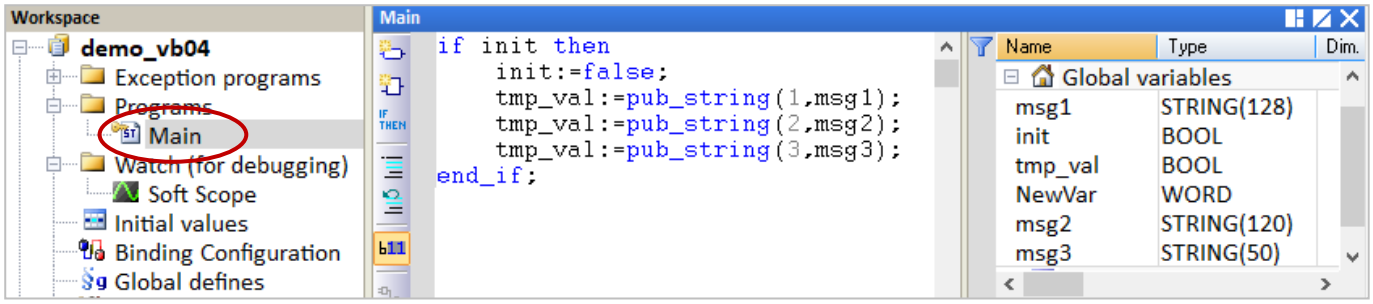
```
(* Execute in the 1st scan cycle *)
if INIT then
  INIT := False;  (* Do not execute after the 1st scan cycle *)
  tstart(tmr_val);
end_if;
```

The following procedure will show you how to use the “pub_string” function to publish the Win-GRAF String variable in the ST program.

Pub_string (Address, String_val) ;

Address: The public address number. It can be 1 to 1024
String_val: The name of String variable.

Also, restore the demo program (**demo_vb04.zip**) to view the codes.



The description of variables:

Name	Type	Description
Init	BOOL	To initialize. The initial value is TRUE.
Tmp_val	BOOL	To determine if the setting is ok. TRUE: Binding succeeds. FALSE: Binding fails.
msg1	STRING, Length is 100	The String variable to be policed. **NOTE: The String length can be 1 to 255.**
msg2	STRING, Length is 32	
msg3	STRING, Length is 60	

ST program (Main):

```
If  init  then
  Init := false;
(*add address 1 for share string val *)
  Tmp_val := pub_string(1,msg1);

(*add address 2 for share string val *)
  Tmp_val := pub_string(2,msg2);

(*add address 3 for share string val *)
  Tmp_val := pub_string(3,msg3);

End_if;
```

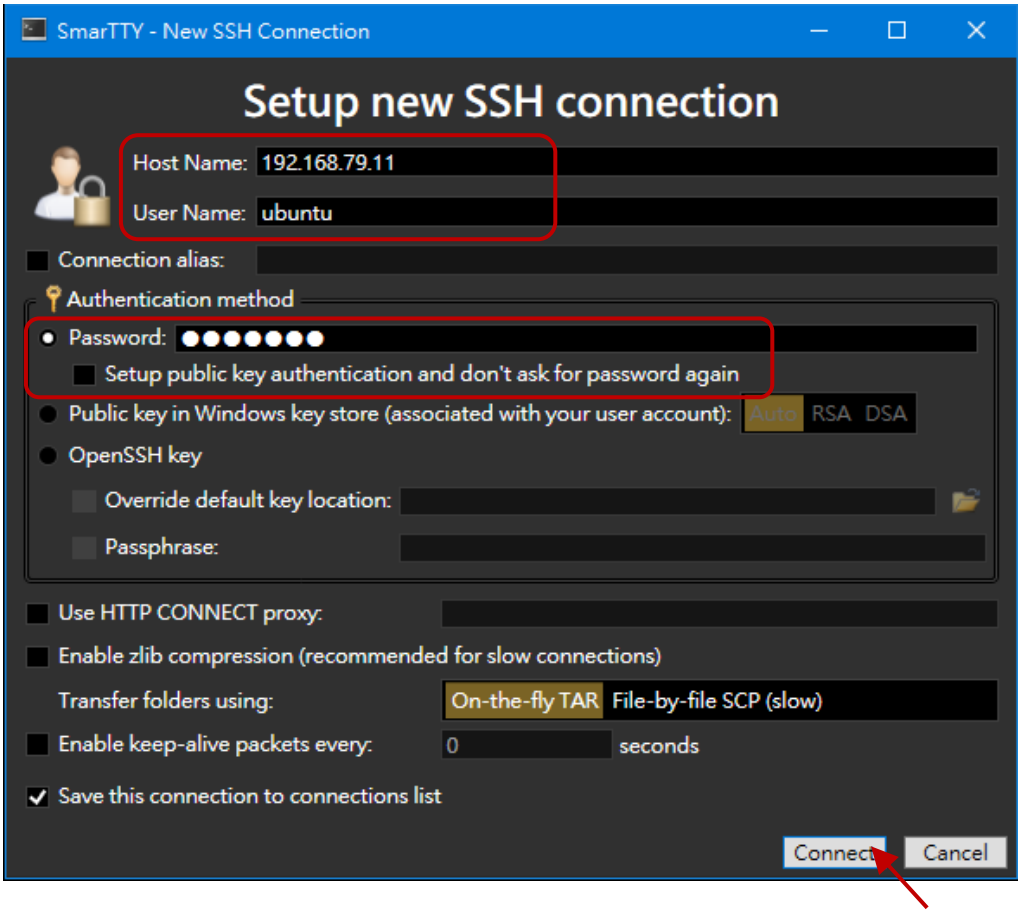
13.2 Downloading and Compiling the C Program

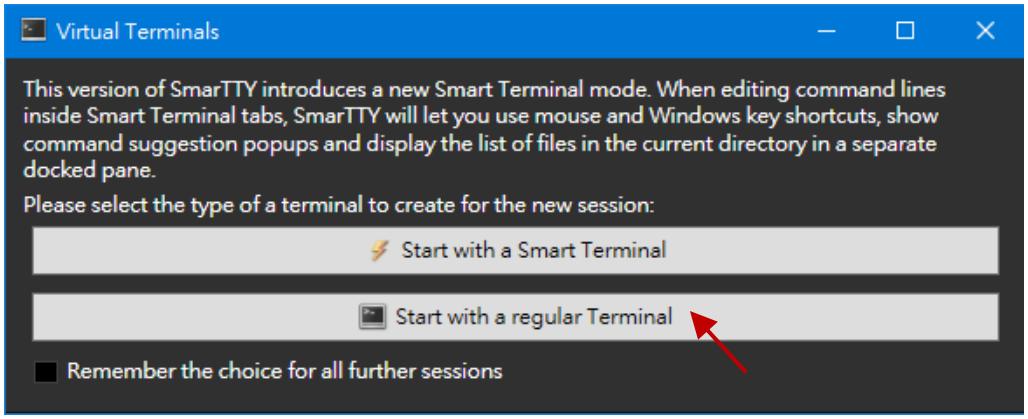
Follow the steps below to download and compile the C program:

1. Download an SSH tool such as **SmarTTY** that supports SCP commands.
https://sysprogs.com/SmarTTY/download/

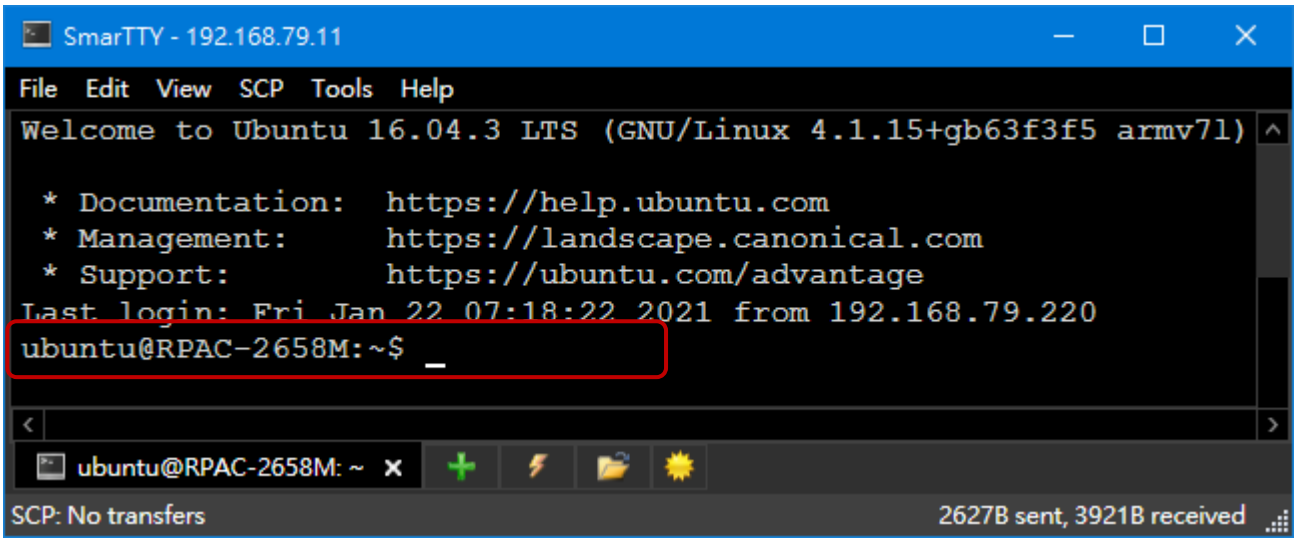
2. Connect to the PAC by using SmartTTY.
Execute the SmartTTY and set up a new SSH connection.

Host name: Enter the IP address of PAC. (LAN1's default IP is '192.168.255.1')
User name: Enter 'ubuntu'.
Password: Enter 'temppwd'.





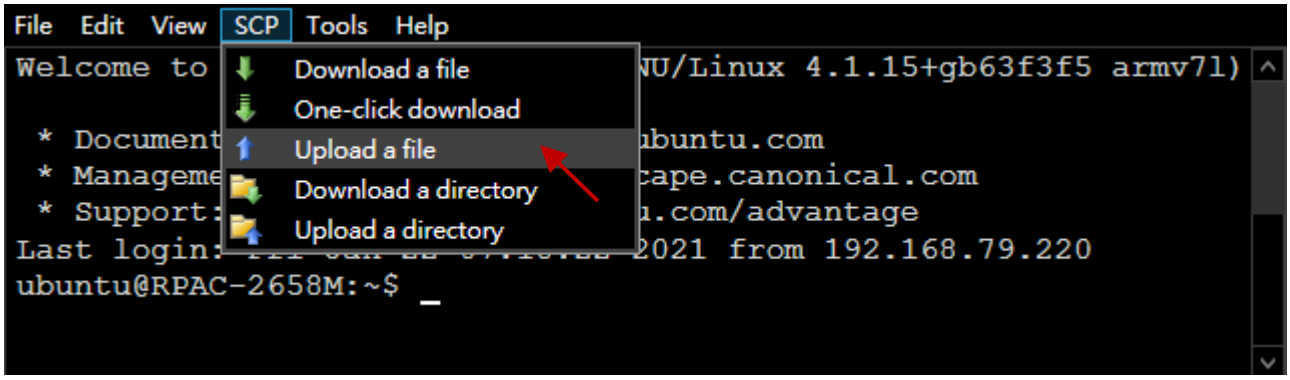
If the connection is successful, the following message will be displayed.



Note: The first time you log in to the PAC, you will be asked to change the password.

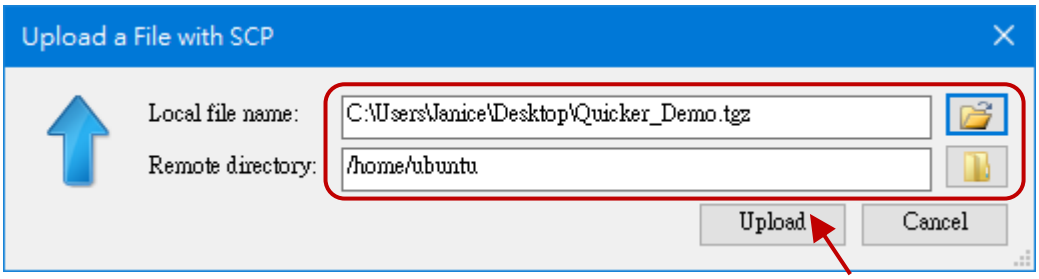
3. Upload the **Quicker_Demo.tgz** file to the PAC.

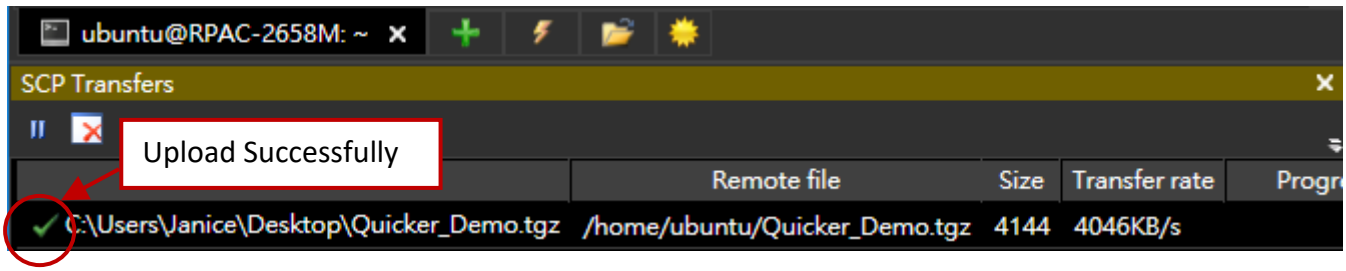
1) Run the **SCP** menu command "Upload a file".



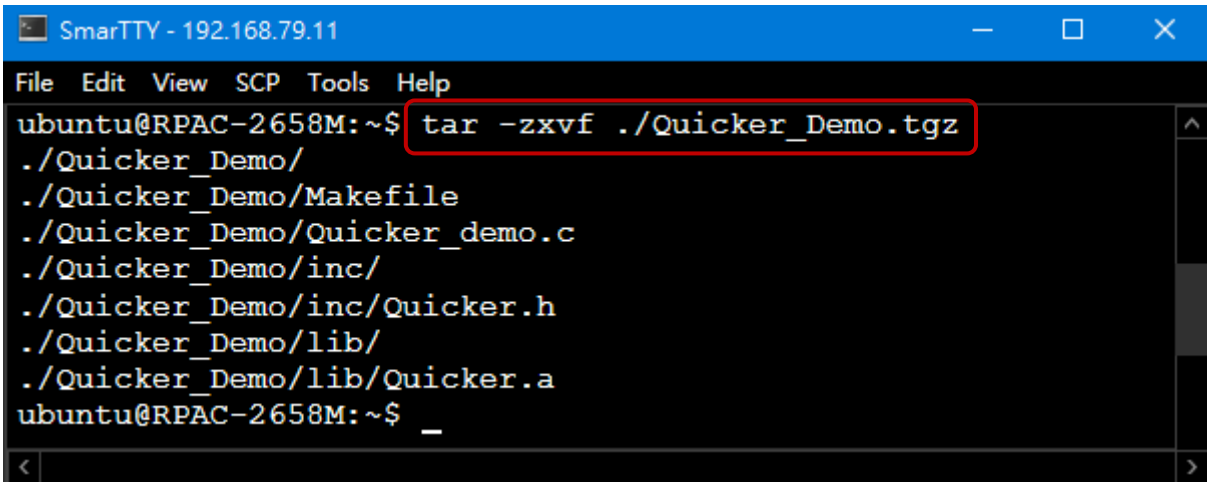
Note: LAN1, LAN2, and LAN3 must be set as different network segments, or the file cannot be uploaded due to a routing error

2) Download the **Quicker_Demo.tgz** file to the **/home/Ubuntu** directory.

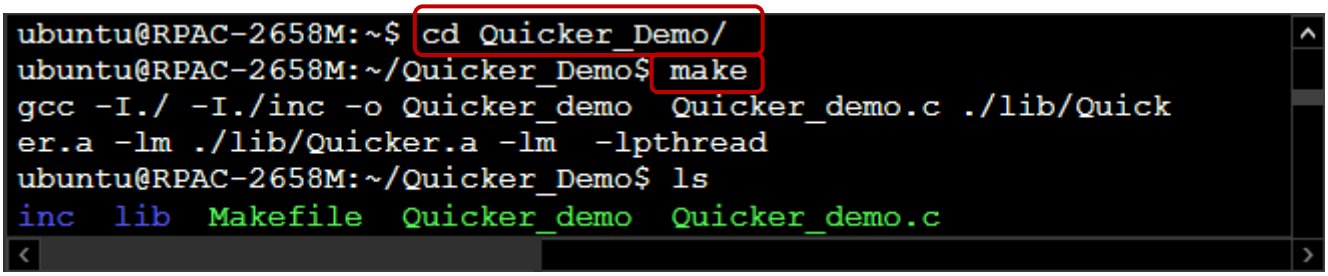




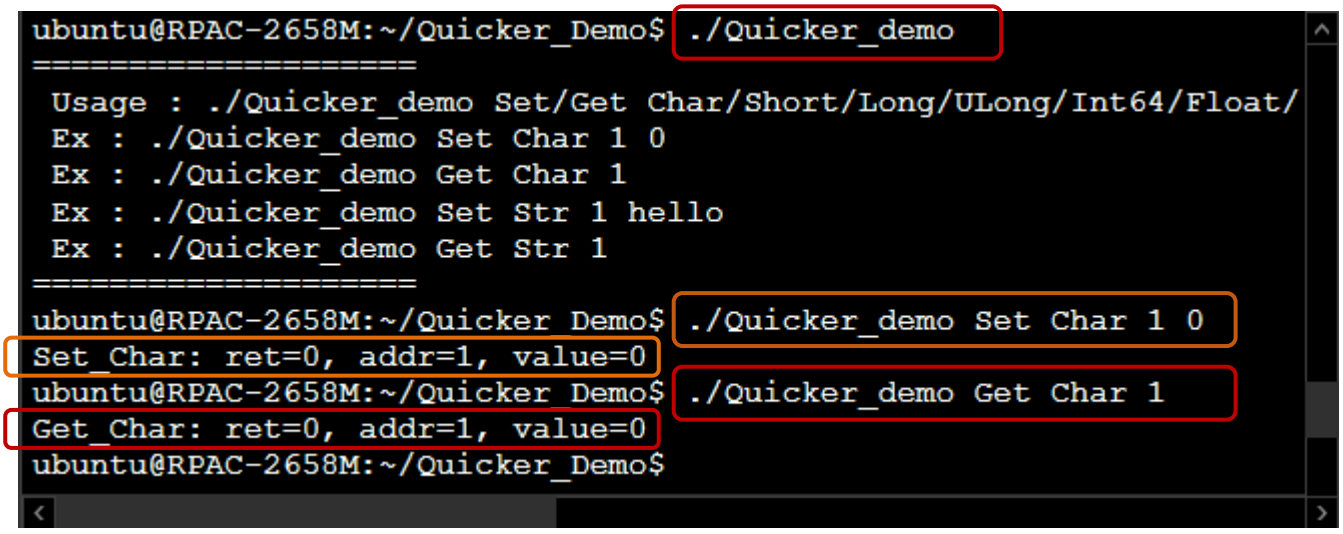
4. Extract the **Quicker_Demo.tgz** archive file.
Using the '**tar –zxvf ./Quicker_Demo.tgz**' command to extract files to the Quicker_Demo folder.



5. Run the **make** command to build the demo program.
1) Type the '**cd Quicker_Demo/**' command to enter the directory.
2) Type the '**make**' command to build the program in the current folder.

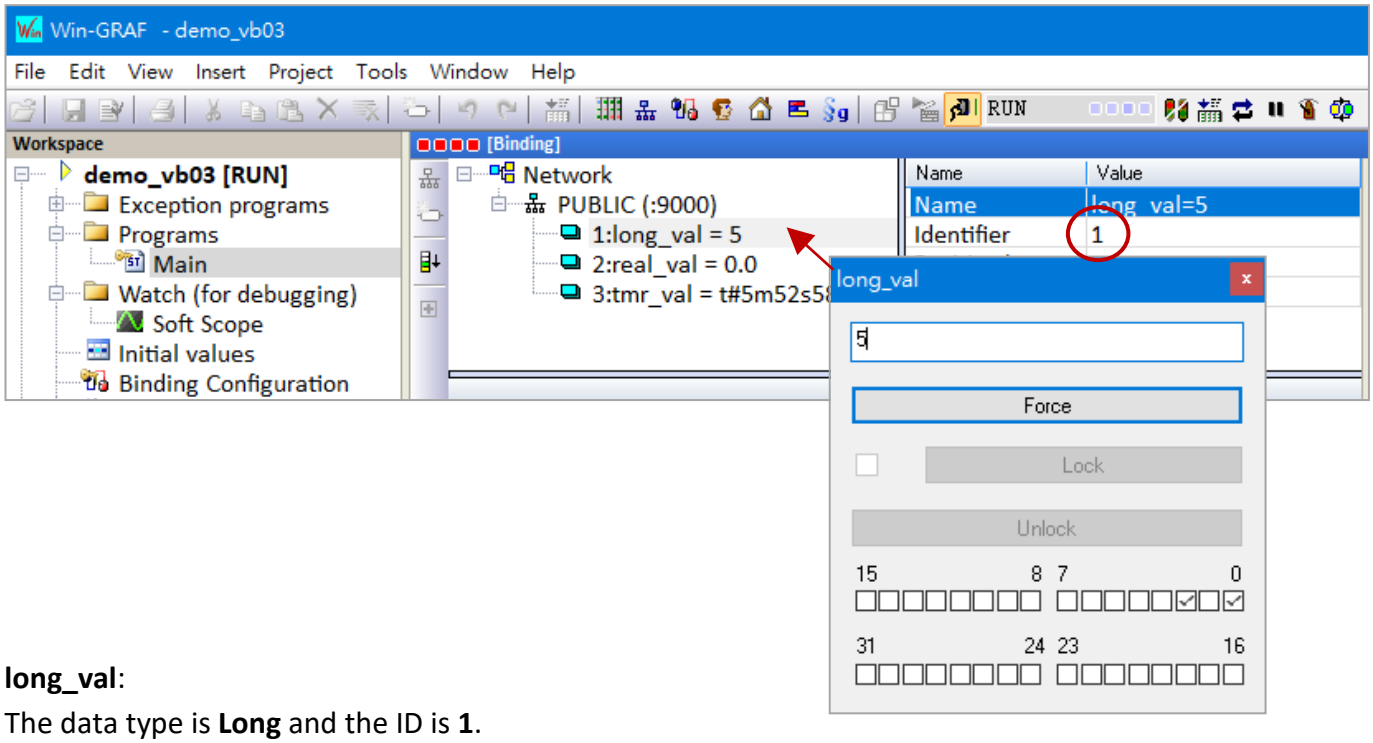


6. Run the demo program to access Win-GRAF variables.



13.3 Accessing Win-GRAF Variables

There are three pubic variables - **long_val**, **real_val**, and **tmr_val** in the Win-GRAF **demo_vb03** project.



long_val:

The data type is **Long** and the ID is **1**.

1) Double-click **long_val** in the Binding window to enter a value (e.g., 5).
2) Enter the **./Quicker_Demo Get Long 1** command to read data.

```
ubuntu@RPAC-2658M:~/Quicker_Demo$ ./Quicker_demo Get Long 1
Get_Long: ret=0, addr=1, value=5
ubuntu@RPAC-2658M:~/Quicker_Demo$ _
```

real_val:

The data type is **Float** and the ID is **2**.

1) Enter the **./Quicker_Demo Set Float 2 1.1** command to write data.
2) Enter the **./Quicker_Demo Get Float 2** command to read data.

```
ubuntu@RPAC-2658M:~/Quicker_Demo$ ./Quicker_demo Set Float 2 1.1
Set_Float: ret=0, addr=2, value=1.100000
ubuntu@RPAC-2658M:~/Quicker_Demo$ ./Quicker_demo Get Float 2
Get_Float: ret=0, addr=2, value=1.100000
ubuntu@RPAC-2658M:~/Quicker_Demo$ _
```

tmr_val:

The data type is **Ulong** and the ID is **3**. Enter the **./Quicker_Demo Get ULong 3** command to read data. The unit of timer is microseconds, so the value 1529275 (ms) equal to 25 m 29 s 275 ms.

```
Get_ULong: ret=0, addr=3, value=1529275
ubuntu@RPAC-2658M:~/Quicker_Demo$ _
```

13.4 Descriptions of Functions in the "Quicker.a"

The Section describes functions in the "Quicker.a" static library.

"Quicker.a" provides many functions for users to read/write Win-GRAF variables data as follows:

1. Read/Write Boolean
2. Read/Write the 8-bit, 16-bit, 32-bit, or 64-bit Integer
3. Read/Write the 32-bit or 64-bit REAL

13.4.1 R/W a Boolean Variable

Set_BOOL

Description:

Used to set the value of a Win-GRAF Boolean variable with the specific address no.

Syntax:

```
unsigned char UserShare_Set_Coil(unsigned short iUserAddress, unsigned char iStatus)
```

Parameter:

iUserAddress: The address no of the Win-GRAF variable (1 to 8192)
iStatus: Set a Boolean value. (1: True; 0: False)

Example:

'Set a Boolean variable value to True with address no.1

```
UserShare_Set_Coil(1, 1)
```

Get_BOOL

Description:

Used to get the value of a Win-GRAF Boolean variable with the specific address no.

Syntax:

```
unsigned char UserShare_Get_Coil(unsigned short iUserAddress, unsigned char *iStatus)
```

Parameter:

iUserAddress: The address no of the Win-GRAF variable no (1 to 8192)
iStatus: Get a Boolean value. (1: True; 0: False)

Example:

'Get a Boolean variable value with address no "1"

```
unsigned char value;
UserShare_Get_Coil(1, &value);
```

13.4.2 R/W an Integer Variable

■ Set_Char ■ Set_Short ■ Set_Long ■ Set_Int64
■ Set_UChar ■ Set_UShort ■ Set_ULong ■ Set_UInt64

Description:
Used to set the value of a Win-GRAF 8-bit, 16-bit, 32-bit, or 64-bit Integer variable with the specific address no.

Syntax:

```
unsigned char UserShare_Set_Char (unsigned short iUserAddress, char iStatus)
unsigned char UserShare_Set_Short (unsigned short iUserAddress, short iStatus)
unsigned char UserShare_Set_Long (unsigned short iUserAddress, long iStatus)
unsigned char UserShare_Set_Int64 (unsigned short iUserAddress, long long iStatus)
unsigned char UserShare_Set_ULong (unsigned short iUserAddress, unsigned long iStatus)
unsigned char UserShare_Set_UChar (unsigned short iUserAddress, unsigned char iStatus)
unsigned char UserShare_Set_UShort (unsigned short iUserAddress, unsigned short iStatus)
unsigned char UserShare_Set_UInt64 (unsigned short iUserAddress, unsigned long long iStatus)
```

Parameter:

iUserAddress:	The address no of the Win-GRAF variable (1 to 8192)
iStatus:	Set an 8-bit, 16-bit, 32-bit, or 64-bit Integer value

Example:

‘Set the value of a 32-bit integer variable with address no "1" to "1234567"
UserShare_Set_Long(1, 1234567)

‘Set the value of a 32-bit integer variable with address no "2" to "-1234"
UserShare_Set_Long(2, -1234)

‘Set the value of a 64-bit integer variable with address no "3" to "123456789012345"
UserShare_Set_Int64(3, 123456789012345)

‘Set the value of an 8-bit integer variable with address no "4" to "125"
UserShare_Set_Char(4, 125)

■ Get_Char ■ Get_Short ■ Get_Long ■ Get_Int64
■ Get_UChar ■ Get_UShort ■ Get_ULong ■ Get_UInt64

Description:

Used to get the value of a Win-GRAF 8-bit, 16-bit, 32-bit, or 64-bit Integer variable with the specific address no.

Syntax:

```
unsigned char UserShare_Get_Char (unsigned short iUserAddress, char *iStatus)
unsigned char UserShare_Get_Short (unsigned short iUserAddress, short *iStatus)
unsigned char UserShare_Get_Long (unsigned short iUserAddress, long *iStatus)
unsigned char UserShare_Get_Int64 (unsigned short iUserAddress, long long *iStatus)
unsigned char UserShare_Get_UChar (unsigned short iUserAddress, unsigned char *iStatus)
unsigned char UserShare_Get_UShort (unsigned short iUserAddress, unsigned short *iStatus)
unsigned char UserShare_Get_ULong (unsigned short iUserAddress, unsigned long *iStatus)
unsigned char UserShare_Get_UInt64 (unsigned short iUserAddress, unsigned long long *iStatus)
```

Parameter:

iUserAddress: The address no of the Win-GRAF variable (1 to 8192)
iStatus: Get an 8-bit, 16-bit, 32-bit, or 64-bit Integer value

Example:

```
long long Dlong_val;
long long_val;
short short_val;
char sbyte_val;

'Get the value of a 64-bit integer variable with address no "7"
UserShare_Get_Int64(7, &Dlong_val);

'Get the value of a 32-bit integer variable with address no "8"
UserShare_Get_Long(8, &long_val);

'Get the value of a 16-bit integer variable with address no "9"
UserShare_Get_Short(9, &short_val);

'Get the value of an 8-bit integer variable with address no "10"
UserShare_Get_Char(10, &sbyte_val);
```

13.4.3 R/W a REAL Variable

■ **Set_Float** ■ **Set_Double**

Description:
Used to set the value of a Win-GRAF 32-bit or 64-bit REAL variable with the specific address no.

Syntax:

```
unsigned char UserShare_Set_Float(unsigned short iUserAddress, float iStatus)
unsigned char UserShare_Set_Double(unsigned short iUserAddress, double iStatus)
```

Parameter:
iUserAddress: The address no of the Win-GRAF variable (1 to 8192)
iStatus: Set a floating-point number

Example:
'Set the value of a 64-bit REAL variable with address no "7" to "11234.234567"
UserShare_Set_Double(7, 11234.234567)

'Set the value of a 32-bit REAL variable with address no "8" to "123.12"
UserShare_Set_Float(8, 123.12);

■ **Get_Float** ■ **Get_Double**

Description:
Used to get the value of a Win-GRAF 32-bit or 64-bit REAL variable with the specific address no.

Syntax:

```
unsigned char UserShare_Get_Float(unsigned short iUserAddress, float *iStatus)
unsigned char UserShare_Get_Double(unsigned short iUserAddress, double *iStatus)
```

Parameter:
iUserAddress: The address no of the Win-GRAF variable (1 to 8192)
iStatus: Get a floating-point number

Example:

```
float float_val;
double double_val;
```

'Get the value of a 64-bit REAL variable with address no "7"
UserShare_Get_Double(7, &double_val);

'Get the value of a 32-bit REAL variable with address no "8"
UserShare_Get_Float(8, &float_val);

13.4.4 R/W a String Variable

Set_STRING

Description:

Used to set Win-GRAF String data with the specific address no.

Syntax:

```
int UserSetReg_Str(unsigned short iUserAddress, char iStatus, unsigned short len)
```

Parameter:

iUserAddress: The address no of the Win-GRAF variable (1 to 1024)
iStatus: Get/Set Win-GRAF String pointers
len: Set the length of a Win-GRAF String

Example:

```
char str_value[256];
//Set String data with address no "7"
sprintf(str_value,"hello world \n");
UserSetReg_Str(7, str_value, strlen(str_value));
```

Get_STRING

Description:

Used to get Win-GRAF String data with the specific address no.

Syntax:

```
int UserGetReg_Str(unsigned short iUserAddress, char *iStatus, unsigned short len);
```

Parameter:

iUserAddress: The address no of the Win-GRAF variable (1 to 1024)
iStatus: Get/Set Win-GRAF String pointers
len: Get the length of a Win-GRAF String

Example:

```
char str_value[256];
// Get String data with address no "7"
UserGetReg_Str(7, str_value, 256);
```

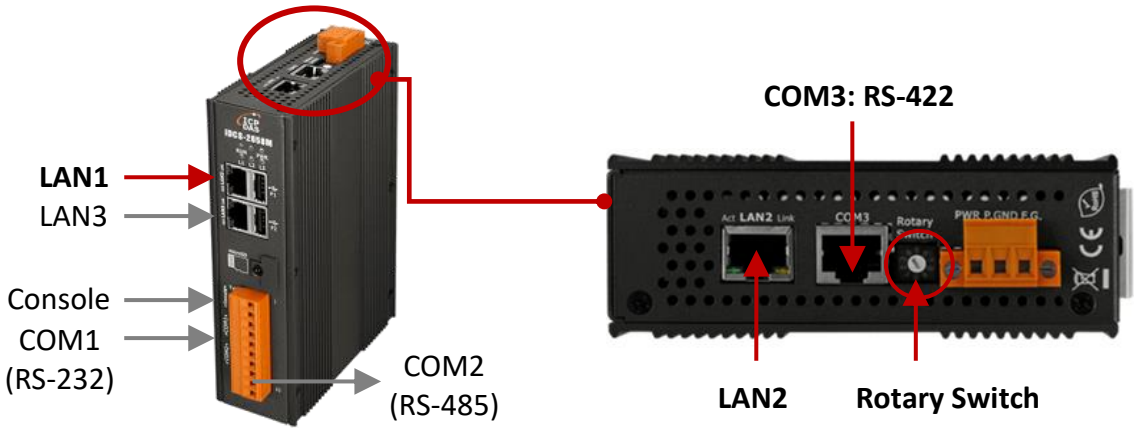
Chapter 14 Redundancy

ICP DAS **Win-GRAF Linux PAC - RPAC-2658M** supports the redundant system:

One redundant system is composed of two Win-GRAF PACs. When the PAC that running programs is crashed or needs to release its control authority by user-defined event, the PAC control authority will automatically switch to the normal one.

Name Definition	Description
Main-PAC	The rotary switch is set to 7
Backup-PAC	The rotary switch is set to 9
Active-PAC	The PAC that is running the program
Passive PAC	The PAC that is synchronizing data

Note: The Main-PAC is Active-PAC at the beginning and its control authority will automatically be changed to the other PAC depends on conditions.



Notice:

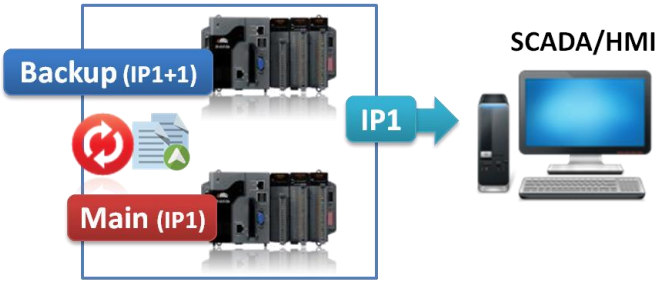
1. **Do not** use two Main-PACs or two Backup-PACs to form a redundant system.
2. Two PACs are communicating by using these communication ports.

Communication Port	Description
A. Public IP Port	**LAN1:** Used to communicate with Win-GRAF, SCADA, or HMI. **Note:** LAN3 is recommended if Ethernet I/O or devices are to be connected.
B. Replication Port	**LAN2**: Data Synchronization Port. Two PACs are connected with an Ethernet crossover cable for high-speed data synchronization. Do not connect an Ethernet switch or hub in-betweens.
C. Alive Port (or Heart-beat Port)	**COM3 (RS-422):** Used to detect if two PACs are running properly. Using the RJ-45 connector. Two PACs can be connected by using an Ethernet crossover cable

Features of the Win-GRAF Redundancy

1. **Higher safety:**
   The redundant system conducts communication through LAN1, LAN2, or COM3 (RS-422). When the Active-PAC detects hardware, software, or communication exceptions, the switching mechanism will be triggered which means the other PAC will take control of the system. So the program can work normally even if two cables are disconnected.

2. **Unique Public IP:**
   Win-GRAF redundant system provides a unique public IP address for SCADA/HMI to access it without needing to determine which one is the Active IP address.



3. **Easy maintenance:**
   If the redundant system is malfunctioning, the operator can power off and remove the broken one and replace a spare one without re-installing all files. The normal PAC will automatically send the Win-GRAF project and data to the new PAC.

   Notice:
   - **Do not shut down or dismounting the normal PAC, keep it running.**
   - Before powering on the PAC, adjust the rotary switch to the proper position, and connect all required communication cables or I/O modules.

   Exception:
   Except for the Win-GRAF project, if there are other projects such as C or HMI running in the redundant system, these files need to pre-installed to the spare RPAC (or a repaired PAC) before re-installing this PAC to the redundant system.

4. **Simplifying the programming process:**
   Users do not need to specify what files or data should be sent to spare PAC because the Win-GRAF redundant system will automatically handle these tasks.

5. **Custom security mechanism:**
   Users can design a security mechanism in the program, for example, if Active-PAC is failed to connect to SCADA because LAN1 or RS-485 is disconnected or damaged, it will automatically reboot which means release its control authority to the other PAC.

6. **I/O Redundancy:**
   Besides PAC Redundancy, users can choose the iDCS-8000 system to achieve I/O redundancy.

14.1 What Kinds of Data will Automatically Send to the Passive-PAC?

Win-GRAF redundant system will automatically send a part of data from Active-PAC to Passive-PAC.

What Kinds of Data can Backup Automatically:

1. The user's Win-GRAF applications.
2. The value of variables.
3. The private data of function block instance.
4. The PAC's RTC (Real Time Clock) time.
5. Retain memory.

The most common data that cannot BackUp to the Passive-PAC Automatically?

1. The status of the Timer variable (Ticking or Sleeping).
2. Some files on the Active-PAC. For example, files stored in the "/system_disk" or "/mnt/microSD" or non-Win-GRAF applications such as C or eLogger applications. These files should be pre-installed in the PAC before mounting them to the redundant system.
3. If using the COM_OPEN() function to open the serial port, which will not automatically be opened on the Passive-PAC.
4. The PAC's data stored in FRAM memory cannot be backed up automatically.

For some data that unable to send to the Passive-PAC automatically, users can deal with them by programming as follows. Refer to the "Retain_and_timer" program in the "demo_RDN_2" project.

```
if  is_first_cycle_just_after_switch  then

  (* Just in the cycle after switching. *)
  ……

end_if ;
```

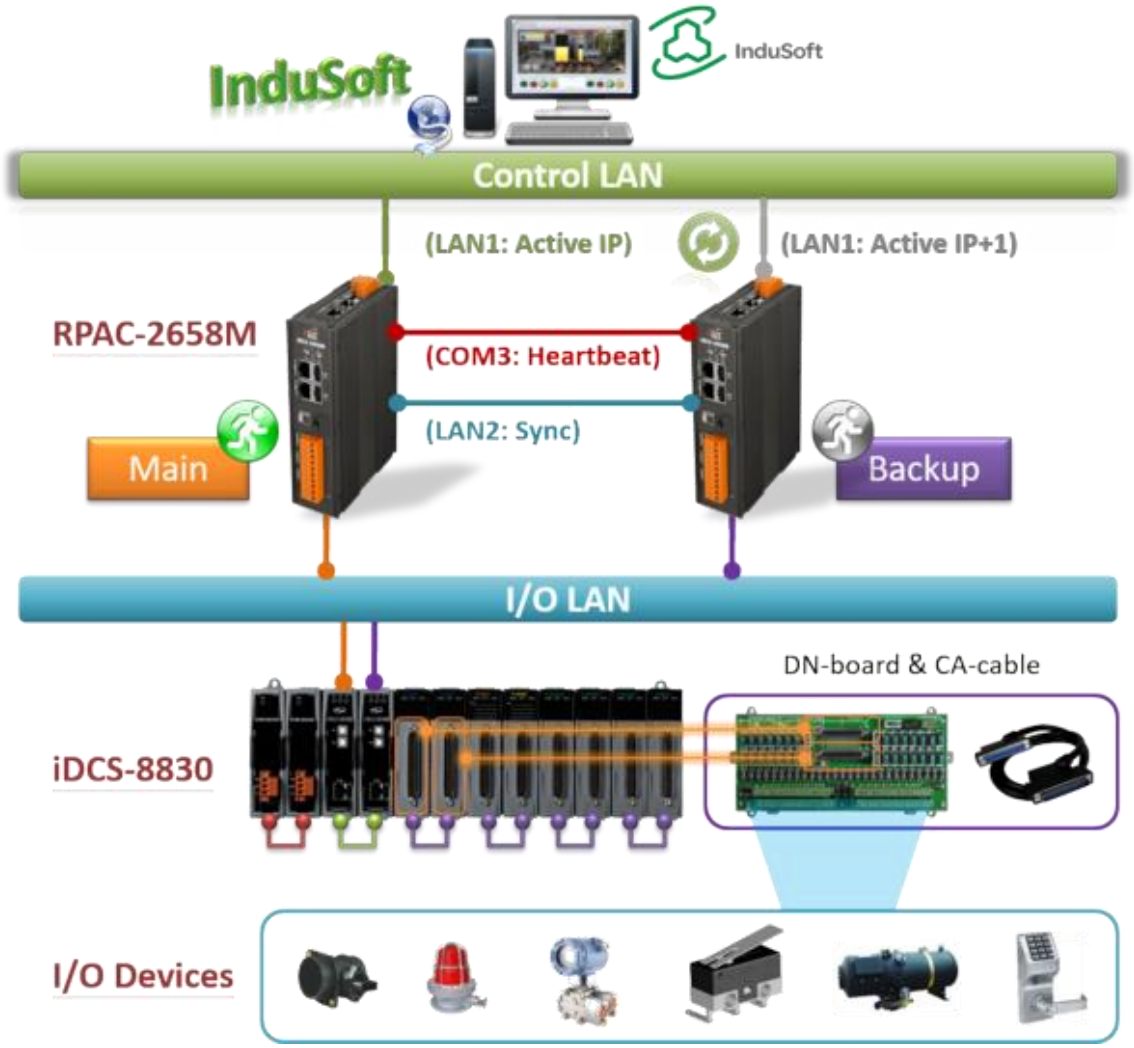
14.2 Redundant System (Rotary Switch: 7 & 9)

Set up the rotary switch and cables as follows:

Hardware		
Two RPAC	Rotary Switch	One is set to "**7**" (called Main-PAC) One is set to "**9**" (called Backup-PAC)
	LAN1	2 x Ethernet cable
	LAN2	1 x Ethernet crossover cable
	COM3 (RS-422)	1 x RS-422 crossover cable, also use Ethernet crossover cable.

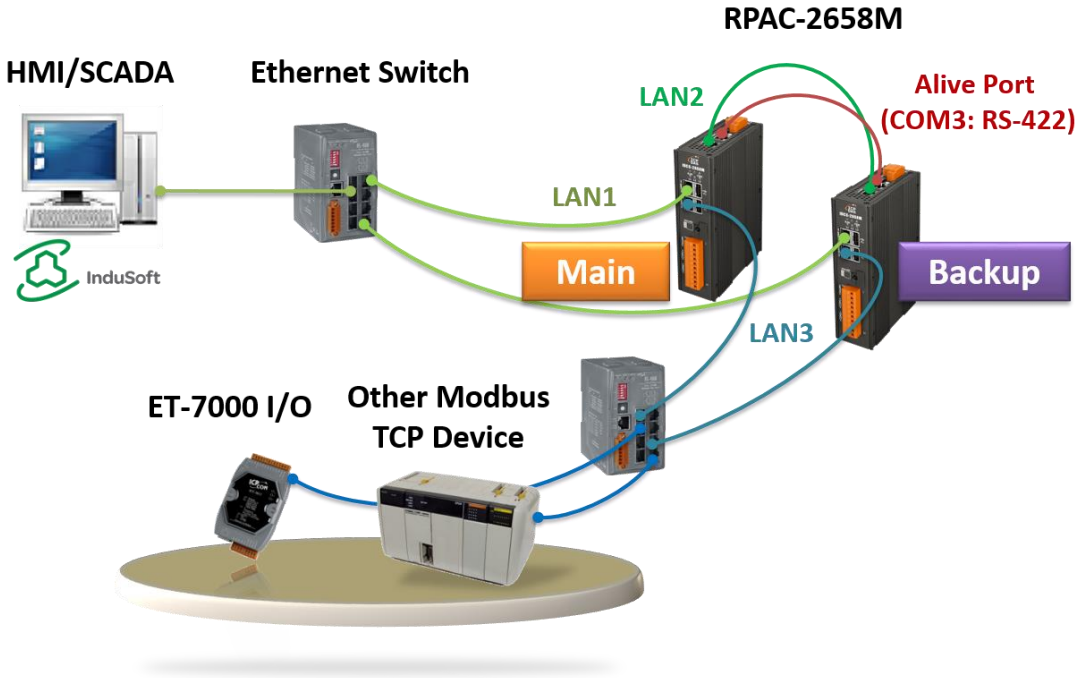
14.2.1 The Architecture of Win-GRAF Near-Field Redundant System

1. **A redundant system with the use of iDCS-8830 I/O modules:**
   iDCS-8830 Ethernet expansion unit and redundant I/O modules.

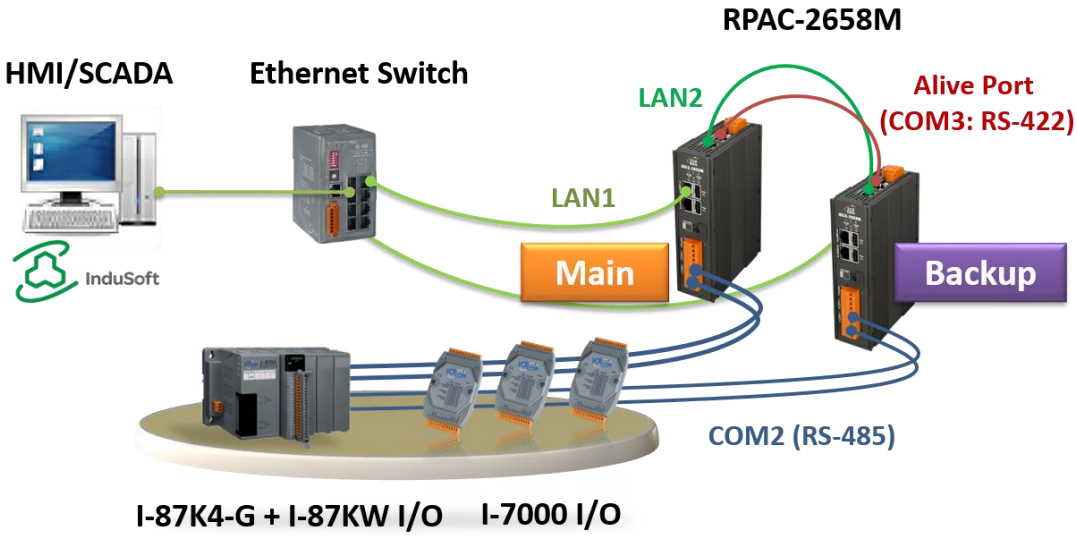


Note: Each pair of plugged-in redundant I/O modules with iDCS-8830 must be the same model numbers.

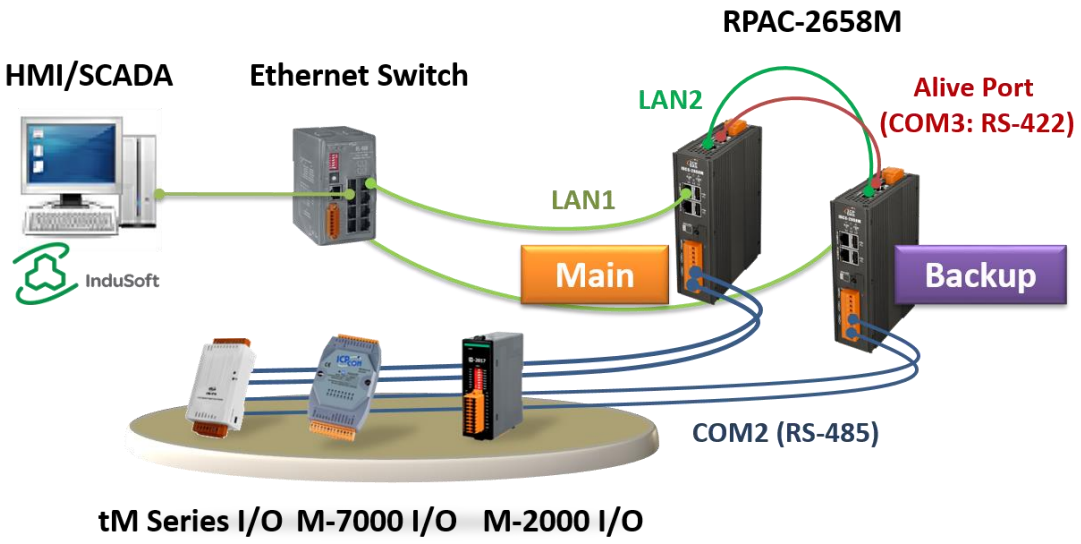
2. **A redundant system with the use of Modbus TCP I/O modules:**



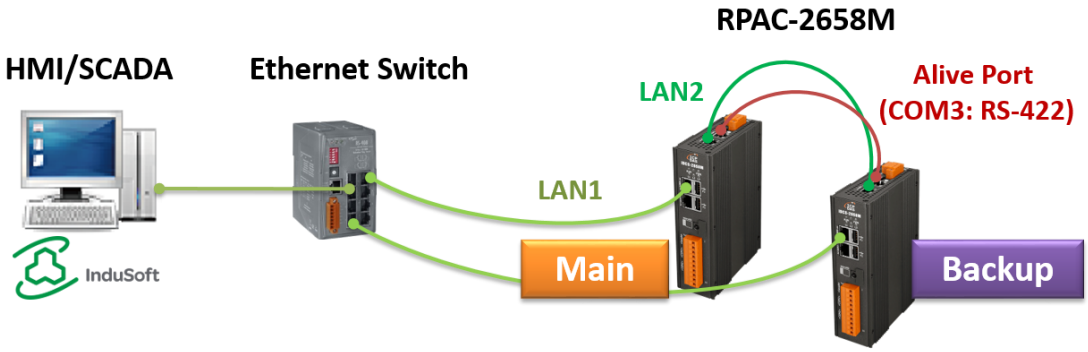
3. **A redundant system with the use of DCON I/O modules:**



4. Two PACs are equipped with Modbus RTU/ASCII I/O modules:

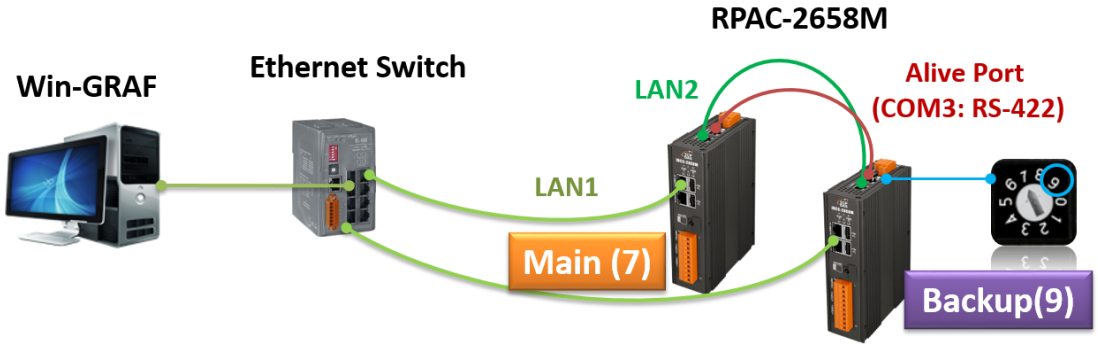


5. A redundant system without plugged-in I/O modules:



The above architecture of the I/O connection (2 to 5) can also be mixed with two or more.

14.2.2 Set up the Redundant System



Step 1: Turn on two PAC with normal mode (i.e., Rotary Switch = 0) and set the LAN1 address.

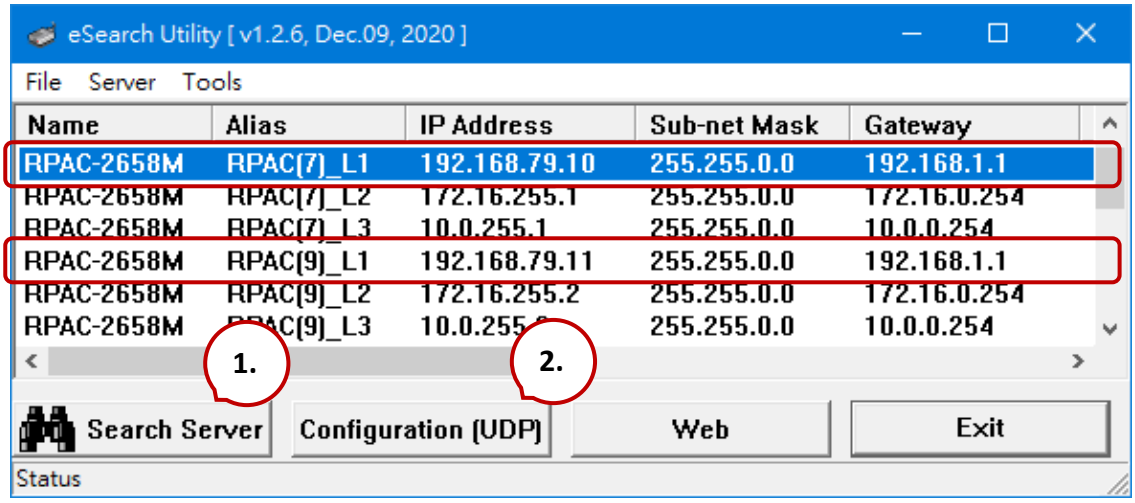
Step 2: Wiring three communication cables for each PAC and power on with redundant mode (i.e., Rotary Switch = 7 & 9).

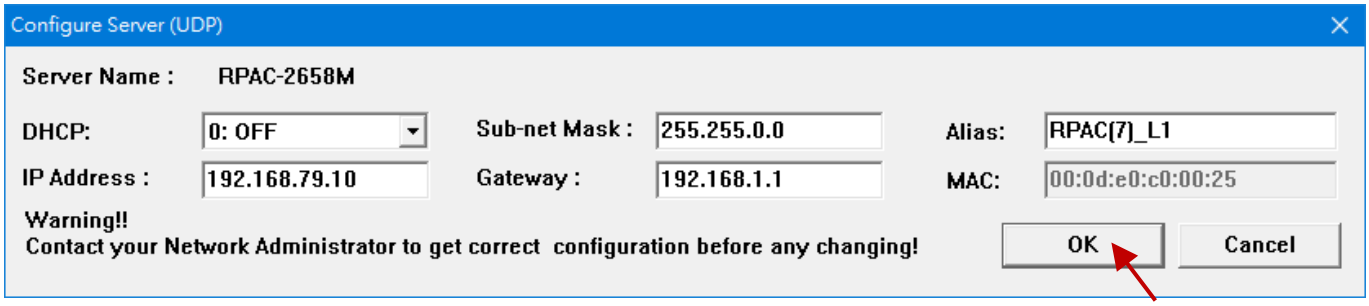
Step 3: Set both communication IP and Active IP in the Win-GRAF project, and download it to Main PAC (7). After that, the project will automatically send to Backup PAC (9) via LAN2.

Step 1: Set the LAN1 Address

A. Set the rotary switch to "0" for the redundant system and power on.

B. Execute eSearch Utility. Click the **Search Server** button to search devices and click the **Configuration** button to set the LAN1 IP address. Refer to Section 1.3 and both the IP addresses of PC/Win-GRAF and PAC must on the same network segment to connect properly.





Note: To avoid IP address conflicts after rebooting PACs, LAN1 and Win-GRAF Active_IP must be set to a different IP address.

Step 2: Set up communication cables and redundant mode (Rotary switch: 7 & 9)

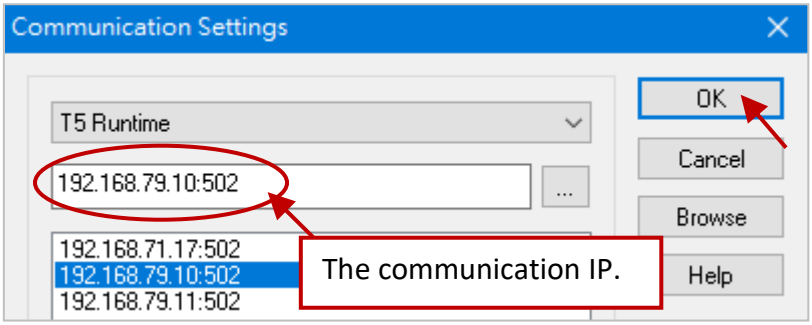
A. Connect each LAN1 port of PACs to an Ethernet switch.
B. In between, two LAN2 ports of PACs must connect to an Ethernet crossover cable.
C. In between, two COM3 ports of PACs must connect to an RS-422 crossover cable.
D. Set the rotary switch of PAC to "7“ (called Main-PAC) and the other one to "9" (called Backup-PAC).
E. Rebooting two PACs.

Note: In redundant mode, the PAC's LAN2 IP will automatically be set to 199.193.195.17/199.193.195.9.

Step 3: Download the Redundant Project

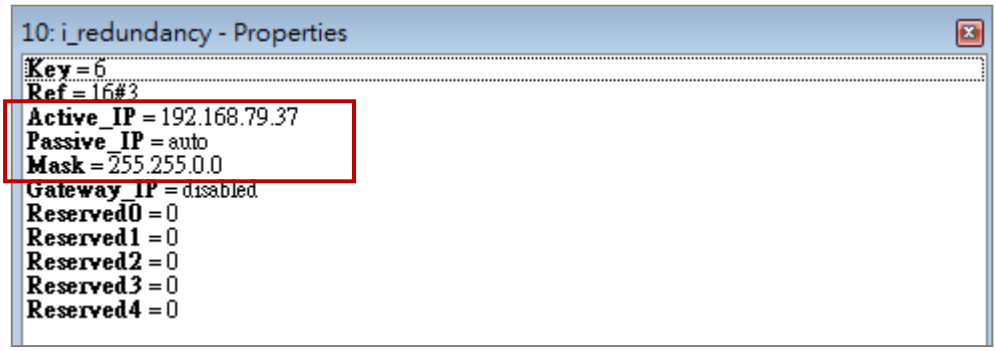
1. **Set the Win-GRAF communication IP address**

   For the first time to download the redundant project, e.g., demo_rdn_2, enter the LAN1 IP address of the Main-PAC (7). (If you are not familiar with the setting, refer to Section 2.3.4)



2. **Set the Active_IP address**

   Set the Active_IP and Mask address for the " i_redundancy" function depends on the network environment. (If you are not familiar with the setting, refer to Chapter 4)

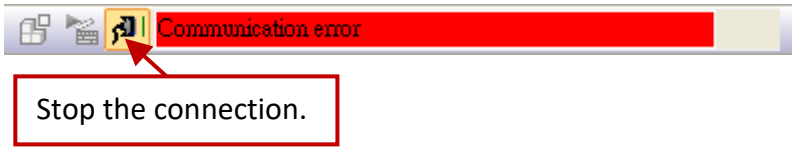


3. **Download the Win-GRAF project**

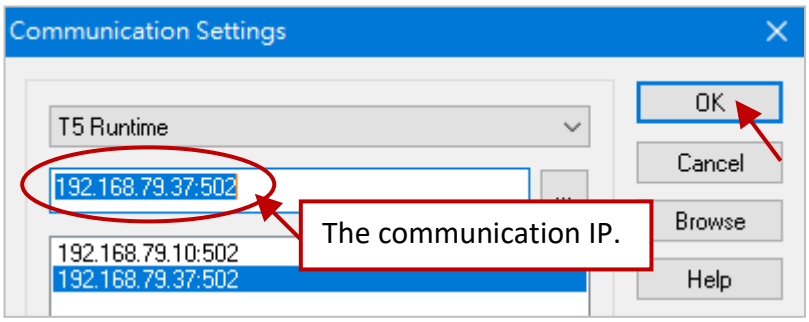
   Click the "On Line" button ( ) to connect and download the redundancy project to the Main-PAC. After completing the download, the PAC's LAN1 IP address will automatically be set to the Active_IP address (e.g., 192.168.79.37) and the LAN1 IP address of the Backup-PAC(9) will automatically be set to Active IP + 1 (e.g., 192.168.79.38).

4. Change the Win-GRAF communication IP address to the Active_IP address

Right now, Win-GRAF will show "Communication error" because the current IP address of the Active-PAC is automatically set to the Active_IP address. Click the “On Line” button again to stop the connection between the Win-GARF and the PAC.

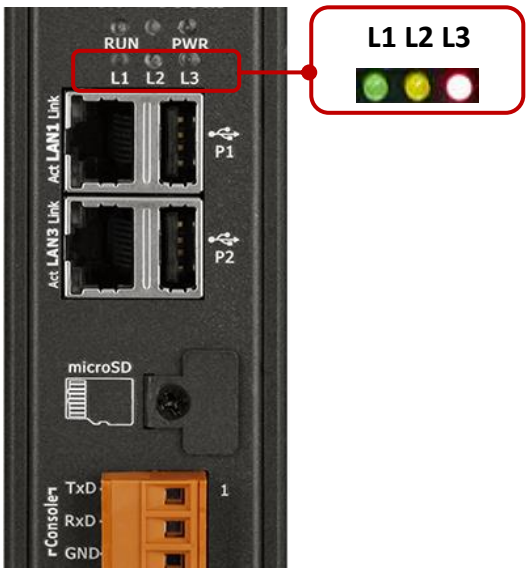


Next, change the Win-GRAF communication IP to the Active_IP address so that this project will always download to the Active-PAC whenever the user wants to update the program.



Note: Before updating the program of the Active-PAC, power off the Passive-PAC. After completing the update, power on the Passive-PAC to ensure that the redundant system is work properly.

Besides, the L3 LED is lit in red which means the PAC is Active-PAC.

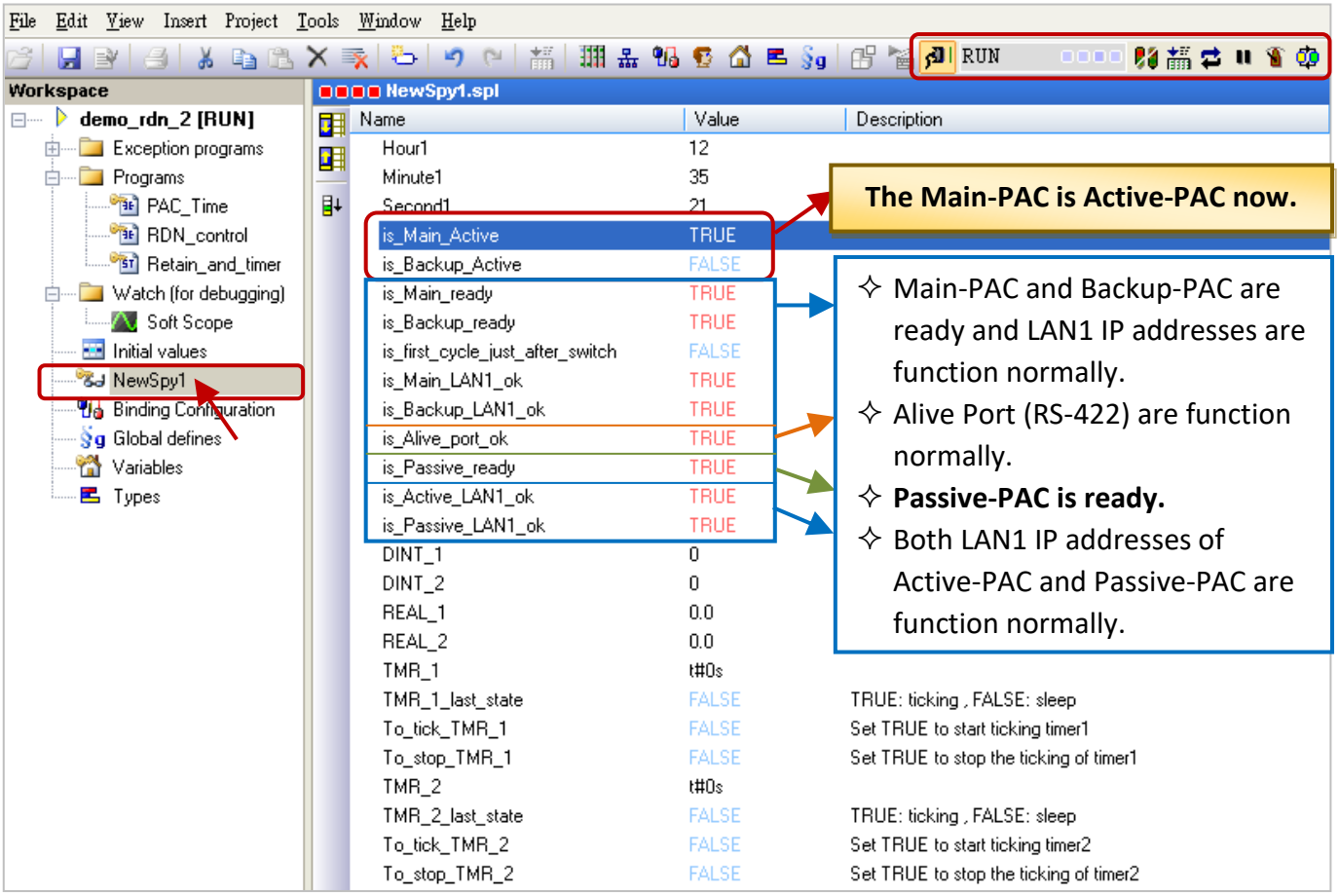


14.2.3 Test the Redundant Project (demo_rdn_2)

Refer to Section 14.4 for the description of demo projects.

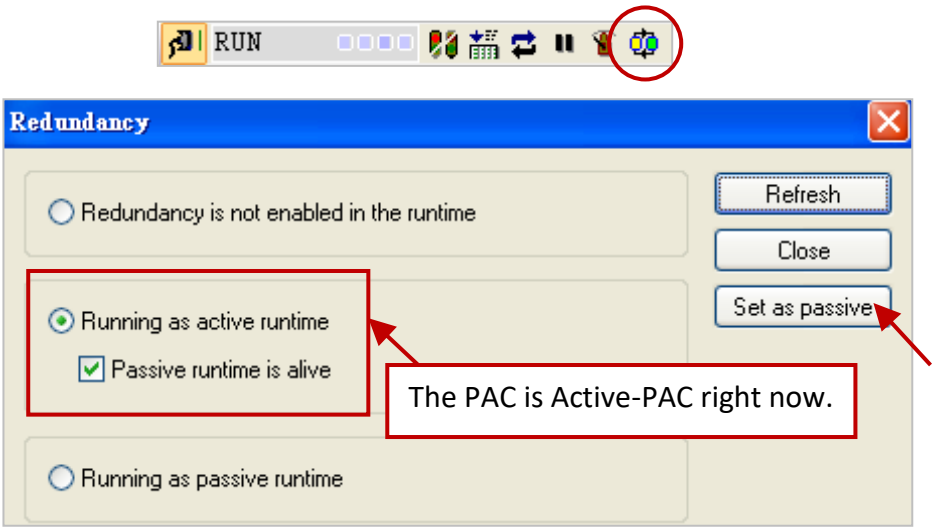
1. View the status of redundant PACs

After Win-GRAF connects to the PAC, click "NewSpy1" to open the spy list window and check the current status of the redundant system.



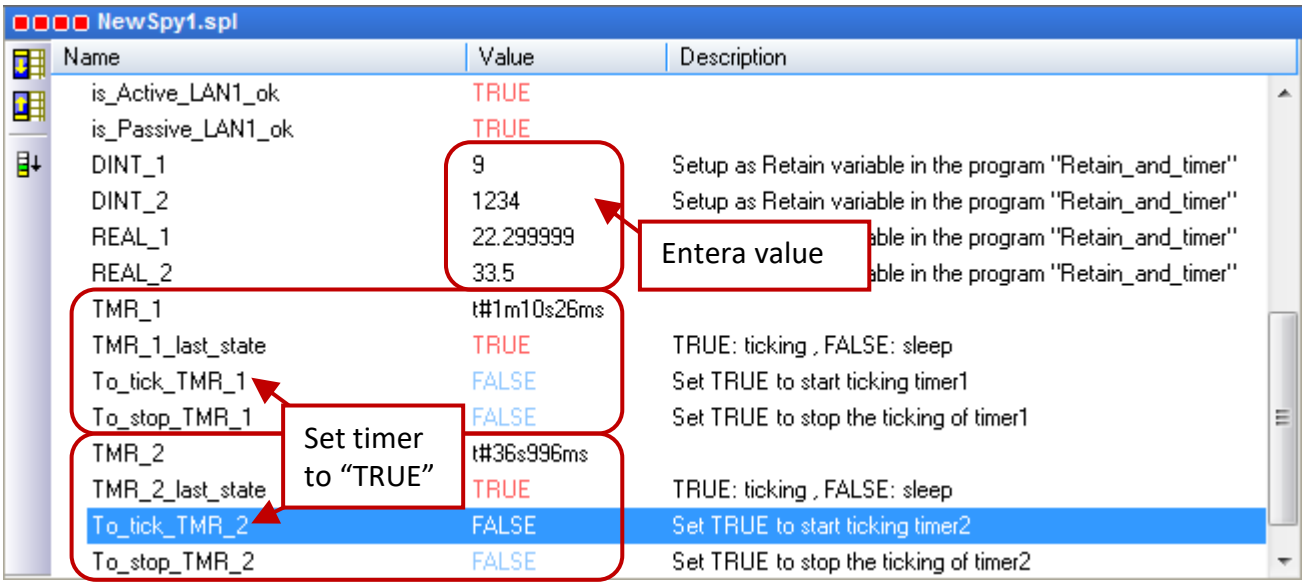
Note:

1) **Before switching the control authority, make sure the Passive-PAC is ready** (i.e., "is_Passive_ready" = TRUE).
2) Users can also click the "Redundancy" button ( ) to manually switch the control authority.



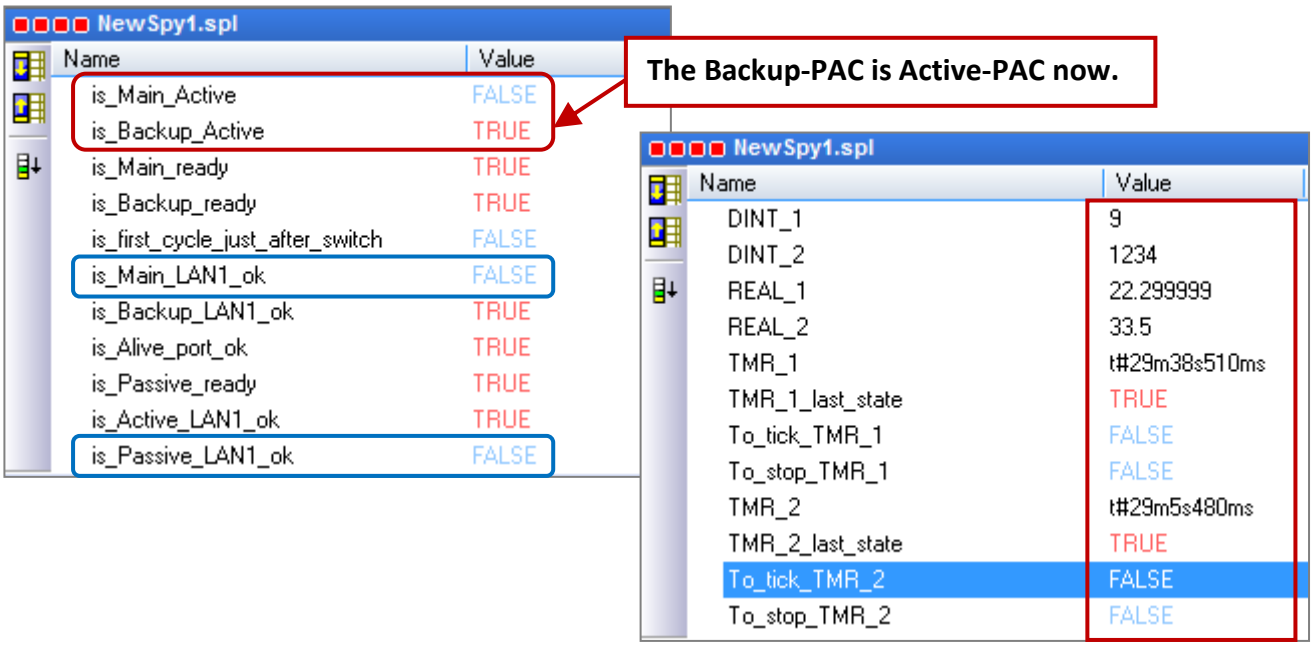
2. Set the value of variables

1) Assign a value to DINT_1, DINT_2, REAL_1, and REAL _2 variables.
2) Set the To_tick_TMR_1 and To_tick_TMR_2 to "TRUE“ (it will automatically reset to FALSE) to start the "TMR_1" and "TMR_2" ticking.



3. Test the redundant system (refer to the “RDN_control” program)

1) Make sure the Passive-PAC is ready (i.e., is_Passive_Ready is TRUE).
2) Remove the LAN1 cable of the Active-PAC and the control will pass to the other PAC. In this demo program (RDN_control), the PAC will automatically reboot after a specific time if the LAN1 cable is disconnected.
3) After rebooting, the Active-PAC is Backup PAC, all values still exist and timers are still ticking.



Note:

Plugin the LAN1 cable on the Main-PAC (Passive), the statuses of "is_Main_LAN1_ok" and "is_Passive_LAN1_ok" will become "TRUE“.

14.3 Description of Win-GRAF Demo Projects

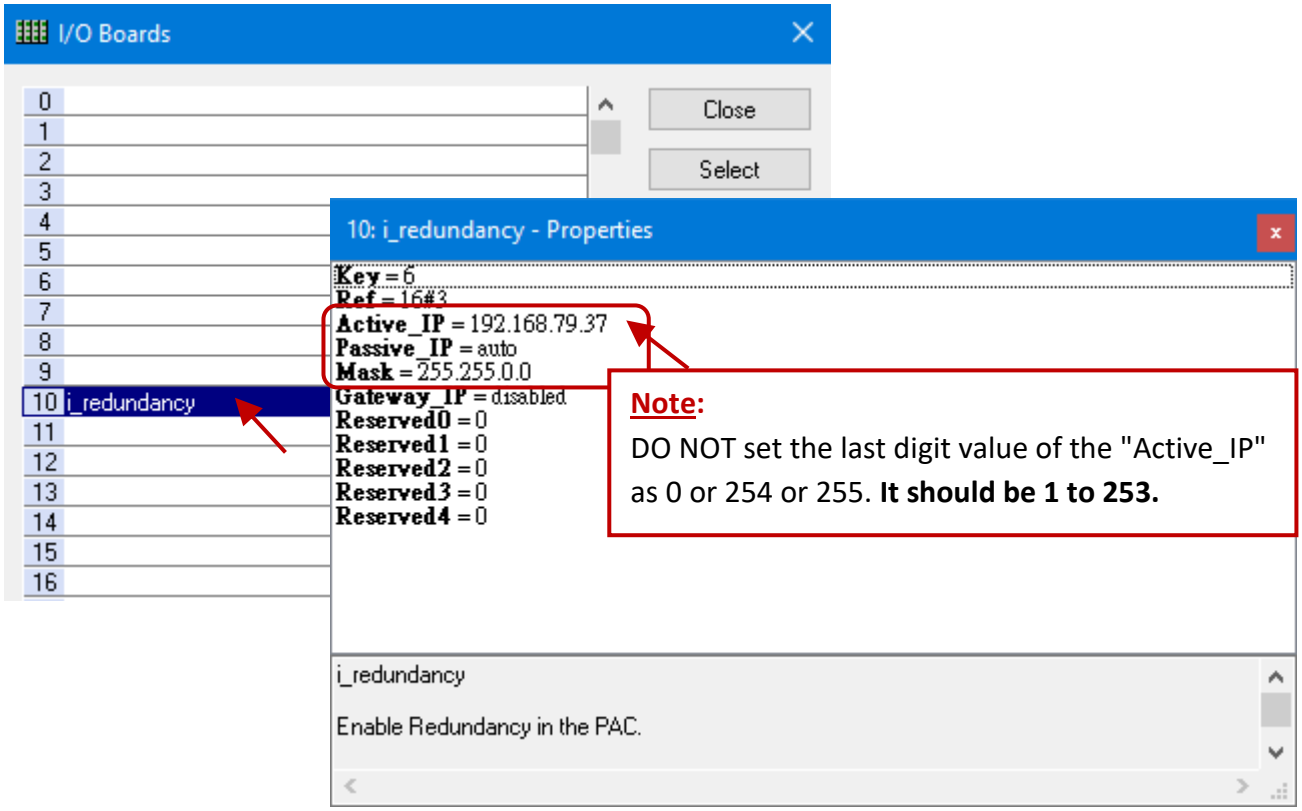
Download the demo program on the website (demo_RDN_1.zip, demo_RDN_2.zip, demo_RDN_3.zip, or and "demo_RDN_4.zip"). Click the menu commands **File - Add Existing Project - From Zip...** in the Workbench to open the project.

Project Name	Description	
demo_RDN_1	Two PACs are connected with three DCON I/O modules via COM2	Test
demo_RDN_2	Using two PACs without connecting any I/O module.	Test, Program
demo_RDN_3	Two PACs are connected with an ET-7050 (Modbus TCP I/O module) via LAN3 through an Ethernet switch.	Test
demo_RDN_4	Two PACs are connected with an iDCS-8830 via LAN3 through an Ethernet switch. iDCS-8830 supports redundant I/O modules.	Test, Program

14.3.1 [Important] "I/O Board" Settings (i_redundancy and i_redundancy_rs485)

✧ **Used for demo_RDN_1, demo_RDN_2, demo_RDN_3, and demo_RDN_4:**

For the redundant system can work normally, the user must add the "i_redundancy" function in the "I/O Boards" window (refer to Chapter4) and set the Active_IP and Mask.



Description of "i_redundancy" I/O board:

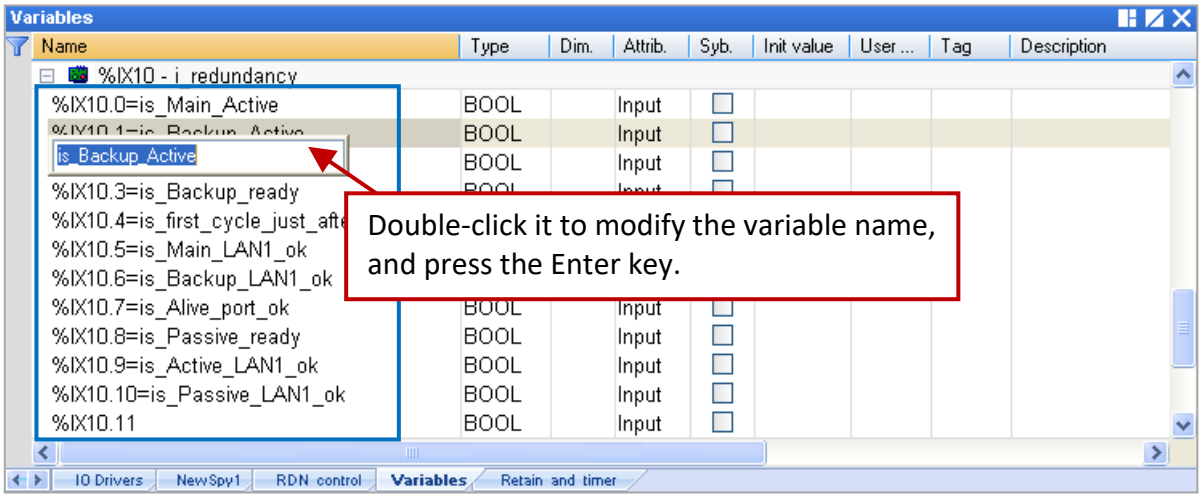
The function is used to display the current status of the Win-GRAF redundant system.

Active_IP: This IP address is public for the redundant system to communicate with HMI or SCADA. **DO NOT** set the last digit value of the "Active_IP" IP address as 0 or 254 or 255. It should be from 1 to 253.

Passive_IP: "Auto", the LAN1 IP address of the Passive-PAC will automatically be set as **Active_IP +1**. E.g., if the "Active_IP" is set as "192.168.71.37", the "Passive_IP" will be automatically set as "192.168.71.38".

Mask: The common settings are either 255.255.255.0 or 255.255.0.0 depending on the network environment.

Note: After adding the "i_redundancy" in the "I/O Boards" window, it will auto add 12 "BOOL" input channels in the "Variables" window that can be used to display the current status of the redundant system.

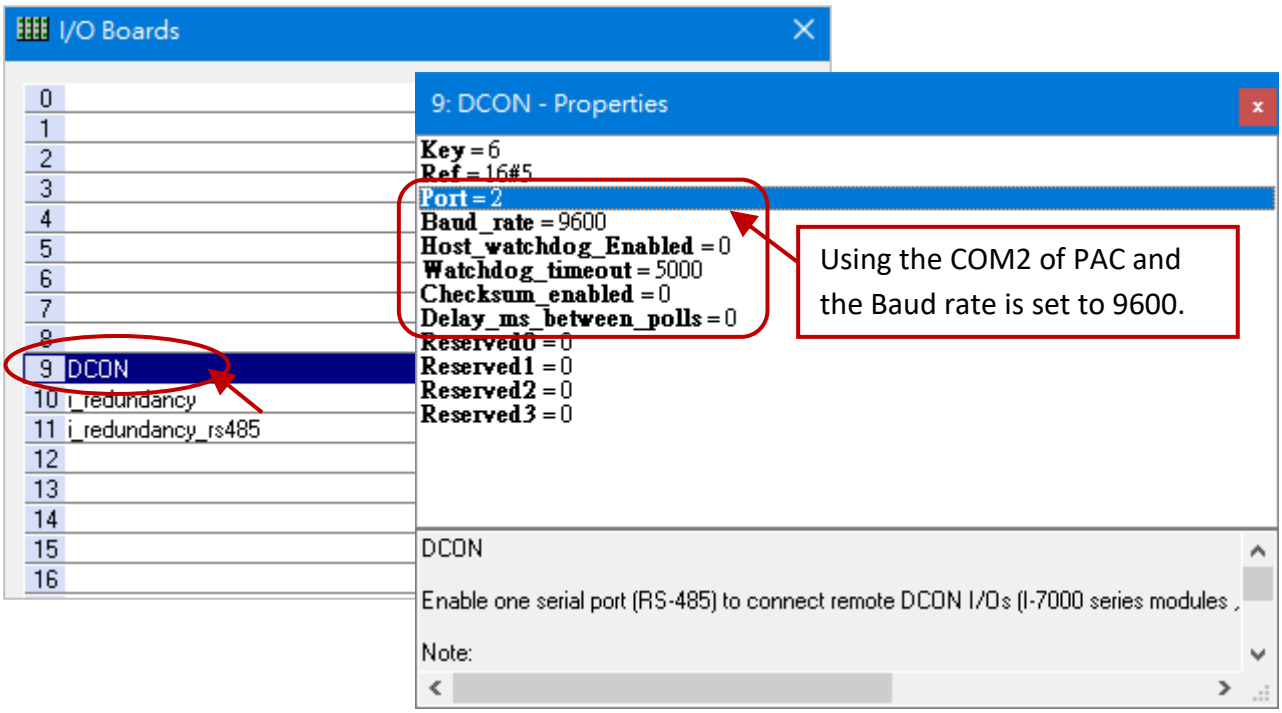


Channel	Variable Name	Description
Ch.0	**is_Main_Active**	**Is the Main-PAC active now?** TRUE: Active PAC. FALSE: Passive PAC.
Ch.1	**is_Backup_Active**	**Is the Backup-PAC active now?** TRUE: Active PAC. FALSE: Passive PAC.
Ch.2	**is_Main_ready**	**Is the Main-PAC ready?** If Ch.2 returns FALSE. The possible reason could be the following. (1) The Ethernet cable (LAN2) between Main-PAC and Backup-PAC is broken. (2) The Main-PAC is dead or crashed. (3) The rotary switch of the Main-PAC is not set at 7 (or 6).

Channel	Variable Name	Description
Ch.3	**is_Backup_ready**	**Is the Backup-PAC ready?** If Ch.3 returns FALSE. The possible reason could be the following. (1) The Ethernet cable (LAN2) between the Main and Backup-PAC is broken. (2) The Main-PAC is dead or crashed. (3) The rotary switch of the Main-PAC is not set at 9 (or 8).
Ch.4	**is_first_cycle_just_after_switch**	**For Active-PAC only.** True: Now is in the first cycle just after switching. False: Now is not in the first cycle after switching.
Ch.5	**is_Main_LAN1_ok**	**Is the LAN1 port of the Main-PAC ok?** TRUE: OK. FALSE: Fail or Ethernet cable is disconnected.
Ch.6	**is_Backup_LAN1_ok**	**Is the LAN1 port of the Backup-PAC ok?** TRUE: OK. FALSE: Fail or Ethernet cable is disconnected.
Ch.7	**is_Alive_port_ok**	**Is the communication status of Alive Port ok?** TRUE: OK. FALSE: Fail or the Passive-PAC is dead or crashed.
Ch.8	**is_Passive_ready**	**Is the Passive-PAC ready now?** If Ch.8 returns FALSE. The possible reason could be the following. (1) The Ethernet cable (LAN2) between the Main and Backup-PAC is broken. (2) The Passive-PAC is dead or crashed. (3) The rotary switch setting of the Passive-PAC is incorrect.
Ch.9	**is_Active_LAN1_ok**	**Is the LAN1 port of the Active-PAC ok?** TRUE: OK. FALSE: Fail or Ethernet cable is disconnected.
Ch.10	**is_Passive_LAN1_ok**	**Is the LAN1 port of the Passive-PAC ok?** TRUE: OK. FALSE: Fail or Ethernet cable is disconnected.

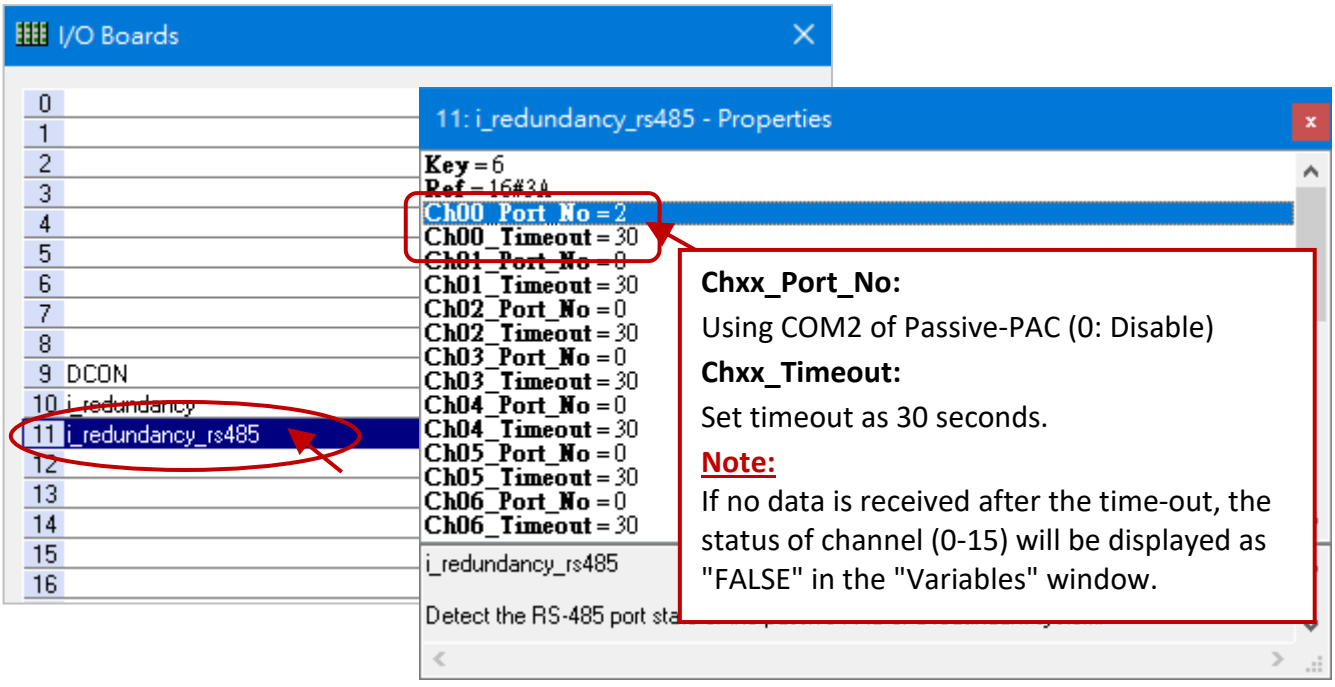
✧ **Used for demo_RDN_1:**

In this case, using the COM3 (RS-485) of PAC to connect DCON I/O modules. Besides "i_redundancy", also "**DCON**" must be added in the I/O Boards window. Moreover, the "**i_redundancy_rs485**" function is used to check if the RS-485 port of Passive-PAC can receive data normally.



Notice:

1. **The "i_redundancy_rs485" must use with the "i_redundancy" or it doesn't work.**
2. The "i_redundancy_rs485" only enables RS-485 ports of Passive-PAC to receive data but not to send data.



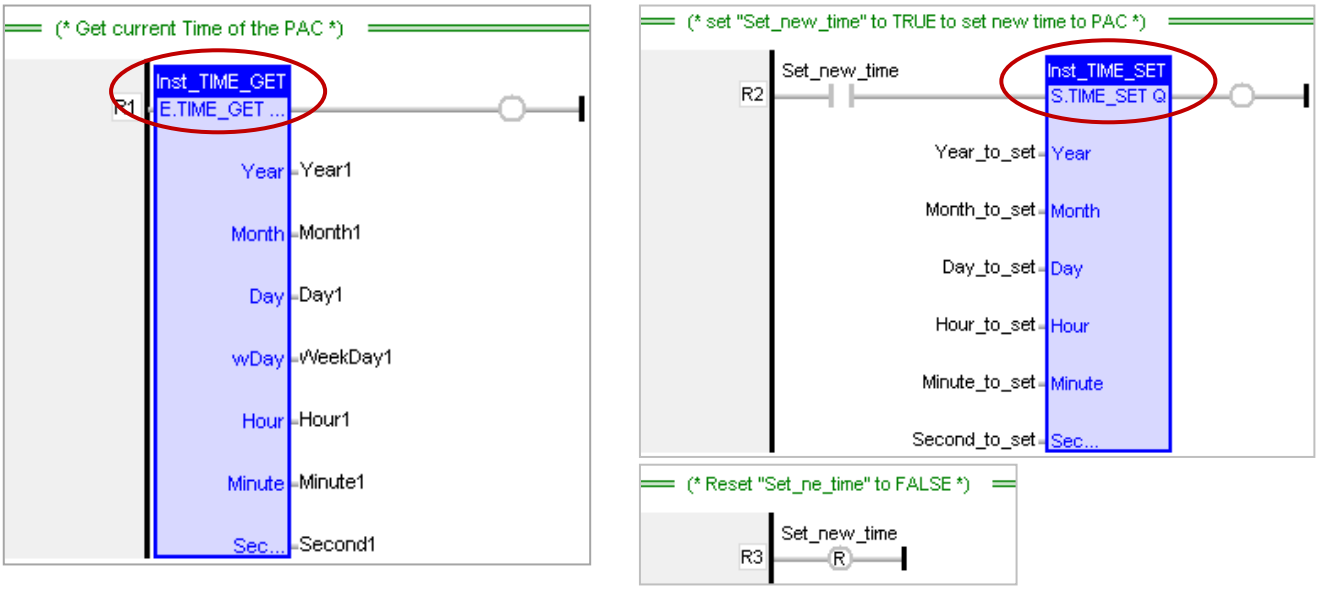
14.3.2 Declaring Variables (demo_RDN_2)

Users can view or add variables in the "Variable" window.

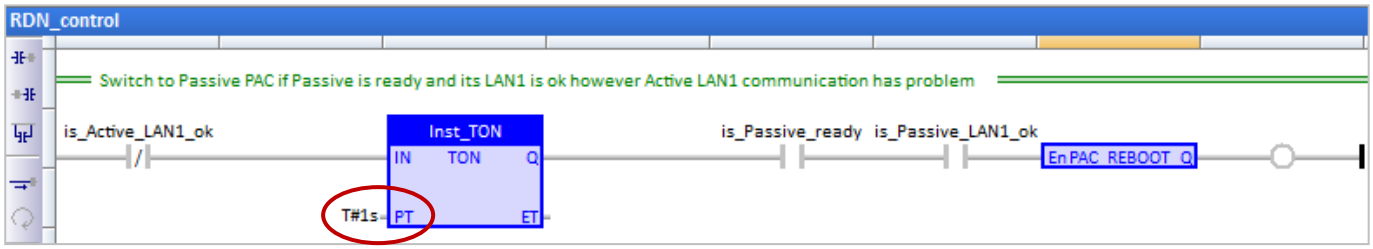
Name	Data Type	Description
Used in the “PAC_Time” program		
Year1	DINT	Used in the “PAC_Time” program: They are used to get the PAC’s system time.
Month1		
Day1		
WeekDay1		
Hour1		
Minute1		
Second1		
Set_new_time	BOOL	Set it as "TRUE" to set up a new system time.
Year_to_set	DINT	Used in the “PAC_Time” program: They are used to set the PAC’s system time.
Month_to_set		
Day_to_set		
Hour_to_set		
Minute_to_set		
Second_to_set		
Used in the “Retain_and_timer” program		
DINT_1	DINT	Used in the "Retain_and_timer" program: Set them as retain variables.
DINT_2		
REAL_1	REAL	
REAL_2		
TMR_1	TIME	Timer
TMR_2		
retain_done	BOOL	TRUE: Retain variables are well set up; FALSE: Not set up yet.
on_line_change_cycle	DINT	Non-zero means this is the first cycle just after On-Line change.
tmp_bool	BOOL	It used to return the Retain status.
TMR_1_last_state		TRUE: Ticking; FALSE: Sleeping.
TMR_2_last_state		TRUE: Ticking; FALSE: Sleeping.
To_tick_TMR_1		Set it as TRUE to start TIMER1.
To_tick_TMR_2		Set it as TRUE to start TIMER2.
To_stop_TMR_1		Set it as TRUE to stop TIMER1.
To_stop_TMR_2		Set it as TRUE to stop TIMER2.

14.3.3 Introduction of the "demo_RDN_2" Project

LD Program – **"PAC_Time"**
It used to get/set the system time of PAC.



LD Program – **"RDN_control"**
If the Passive-PAC's communication and health status is better than Active-PAC when there is a problem with Active PAC, the control authority will be changed to the Passive-PAC immediately. In this example program, PAC will automatically reboot after detecting LAN1 disconnection for more than 1 second.



ST Program – **"Retain_and_timer"**

```
(*   "on_line_change_cycle" is declared as DINT (nonezero means  it is in the cycle jsut
     after doing on line change) .
     "retain_done" is declared as BOOL and inited as FALSE .
     "tmp_bool" is declared as BOOL.  *)

on_line_change_cycle := GetSysInfo (_SYSINFO_CHANGE_CYCLE) ;
if (retain_done = FALSE) or
  (is_first_cycle_just_after_switch = TRUE) or
  (on_line_change_cycle <> 0) then
    retain_done := TRUE ;  (*just do it one time *)
    tmp_bool := Retain_Var( DINT_1 , 1) ; (* retain a DINT variable *)
    tmp_bool := Retain_Var( DINT_2 , 2) ;
    tmp_bool := Retain_Var( REAL_1 , 3) ; (* retain a REAL variable *)
    tmp_bool := Retain_Var( REAL_2 , 4) ;
```

```
        (* if Retain variables havn't been inited yet, use default value *)
    if  (DINT_1 < -1000000) or (DINT_1 > 1000000) or
        (DINT_2 < -2000000) or (DINT_2 > 2000000) or
         (REAL_1 < -9.9E10) or (REAL_1 > 9.9E10) or
        (REAL_2 < -9.9E10) or (REAL_2 > 9.9E10)  then
       DINT_1 := 0 ;
       DINT_2 := 0 ;
       REAL_1 := 0.0 ;
       REAL_2 := 0.0 ;
    end_if ;
end_if ;

(*  is_first_cycle_just_after_switch :
     TRUE  : just in the cycle after switching.
     FALSE : other cycle   *)
if  is_first_cycle_just_after_switch  then

  (*  The Timer ticking state is not auto-redundant. So we have to process them here.
      Ticking timer in the cycle just after switching if its last state is "ticking"   *)
  if  TMR_1_last_state  then
    tStart(TMR_1) ;
  end_if ;
  if  TMR_2_last_state  then
    tStart(TMR_2) ;
  end_if ;
end_if ;

(* Timer operation *)
if  To_tick_TMR_1  then
  To_tick_TMR_1 := FALSE ;
  tStart(TMR_1) ;
  TMR_1_last_state := TRUE ;
end_if ;
if  To_tick_TMR_2  then
  To_tick_TMR_2 := FALSE ;
  tStart(TMR_2) ;
  TMR_2_last_state := TRUE ;
end_if ;
if  To_stop_TMR_1  then
  To_stop_TMR_1 := FALSE ;
  tStop(TMR_1) ;
  TMR_1_last_state := FALSE ;
end_if ;
if  To_stop_TMR_2  then
  To_stop_TMR_2 := FALSE ;
  tStop(TMR_2) ;
  TMR_2_last_state := FALSE ;
end_if ;
```

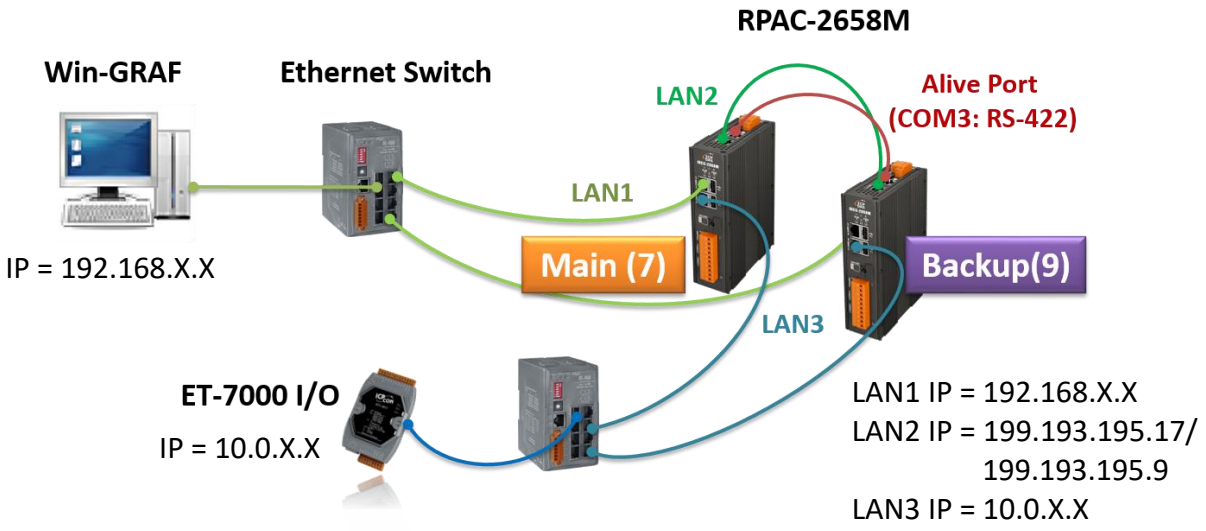
14.4 The Description for Other Redundant Projects

14.4.1 Test the Redundant Project (demo_rdn_3, demo_rdn_1)

Description (demo_rdn_3):

Using both the **LAN3** of two PACs to connect an ET-7050 (Modbus TCP I/O module, 12 DI, 6 DO) through one Ethernet switch.

Hardware Wiring:

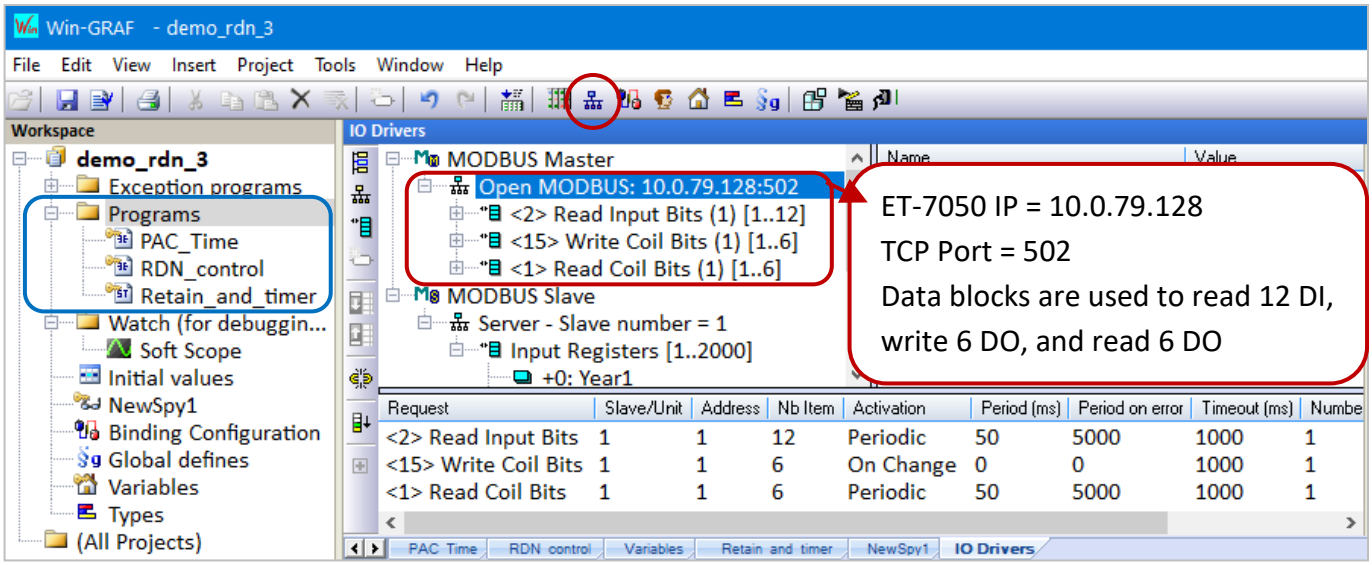


1. **Configure the ET-7000 module**

   Refer to the ET-7000 manual to set the IP address and I/O settings (also refer to Section 5.2.1).
   Manual: https://www.icpdas.com/en/download/show.php?num=2217&model=ET-7050

2. **View settings of the Win-GRAF project**

   In the "I/O Drivers" window, set the PAC as Modbus TCP Master to connect an ET-7050 module (Modbus TCP Slave, Addr. = 1) and create some data blocks to read/write DI and DO data (refer to Section 5.2 for details about the setting).



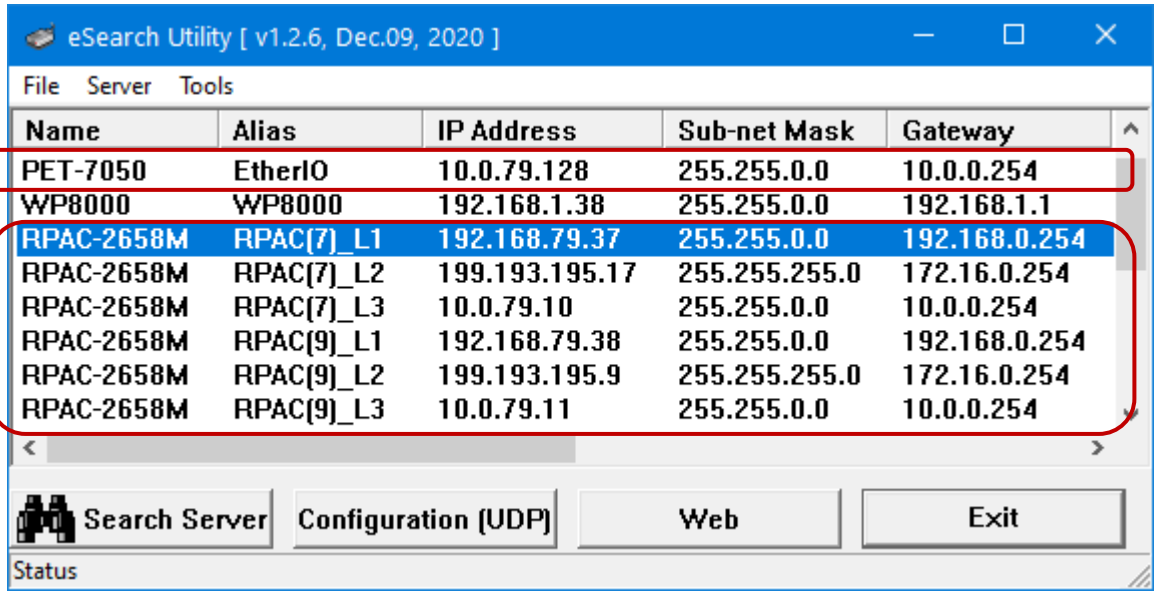
In the Workspace, double-click the program name to view the program.

3. Download the Win-GRAF project ("demo_rdn_3")

Configure the proper Win-GRAF communication IP, Active_IP, Mask, and ET-7000 IP addresses, and then download the project to the PAC.

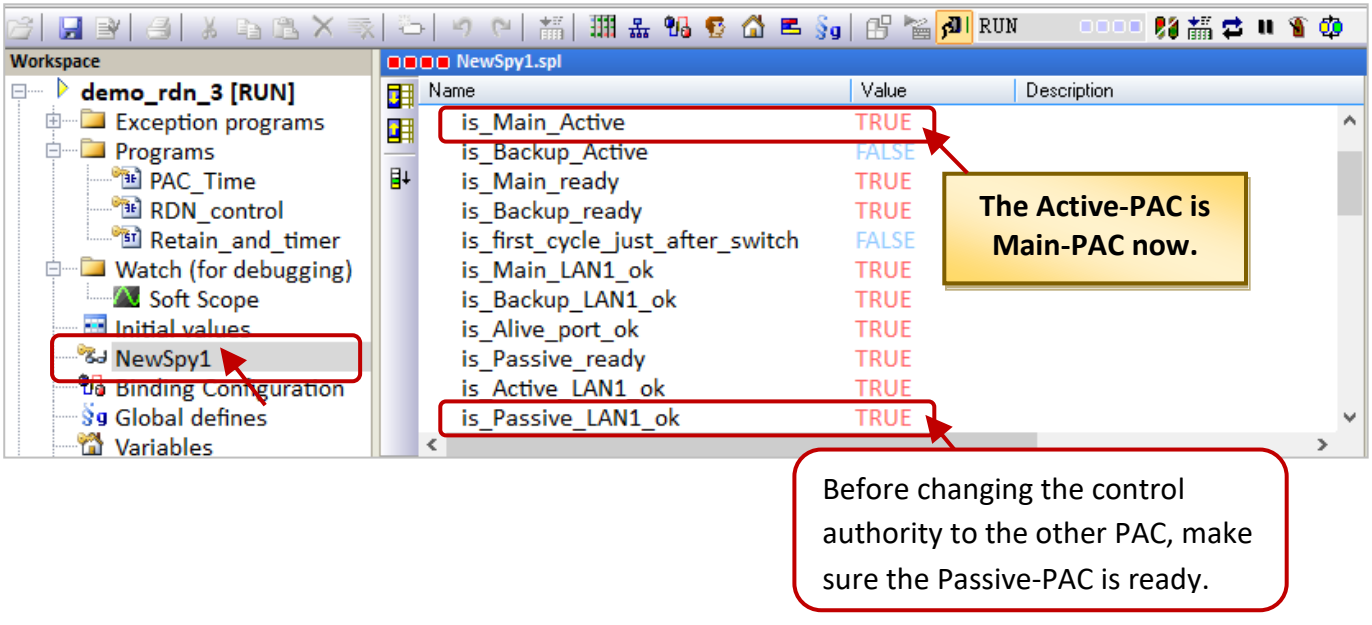
Note:

- Using eSearch Utility to check the IP address of devices.
- In the example, the IP address of LAN3 and PET-7050 must on the same network segment.
- In redundant mode, the LAN2 IP address of two PACs will automatically be set to 199.193.195.17 and 199.193.195.9.
- Power off the Passive-PAC before **updating the project** to the Active-PAC. After completing the update, power on the Passive-PAC to ensure that the redundant system is work properly.



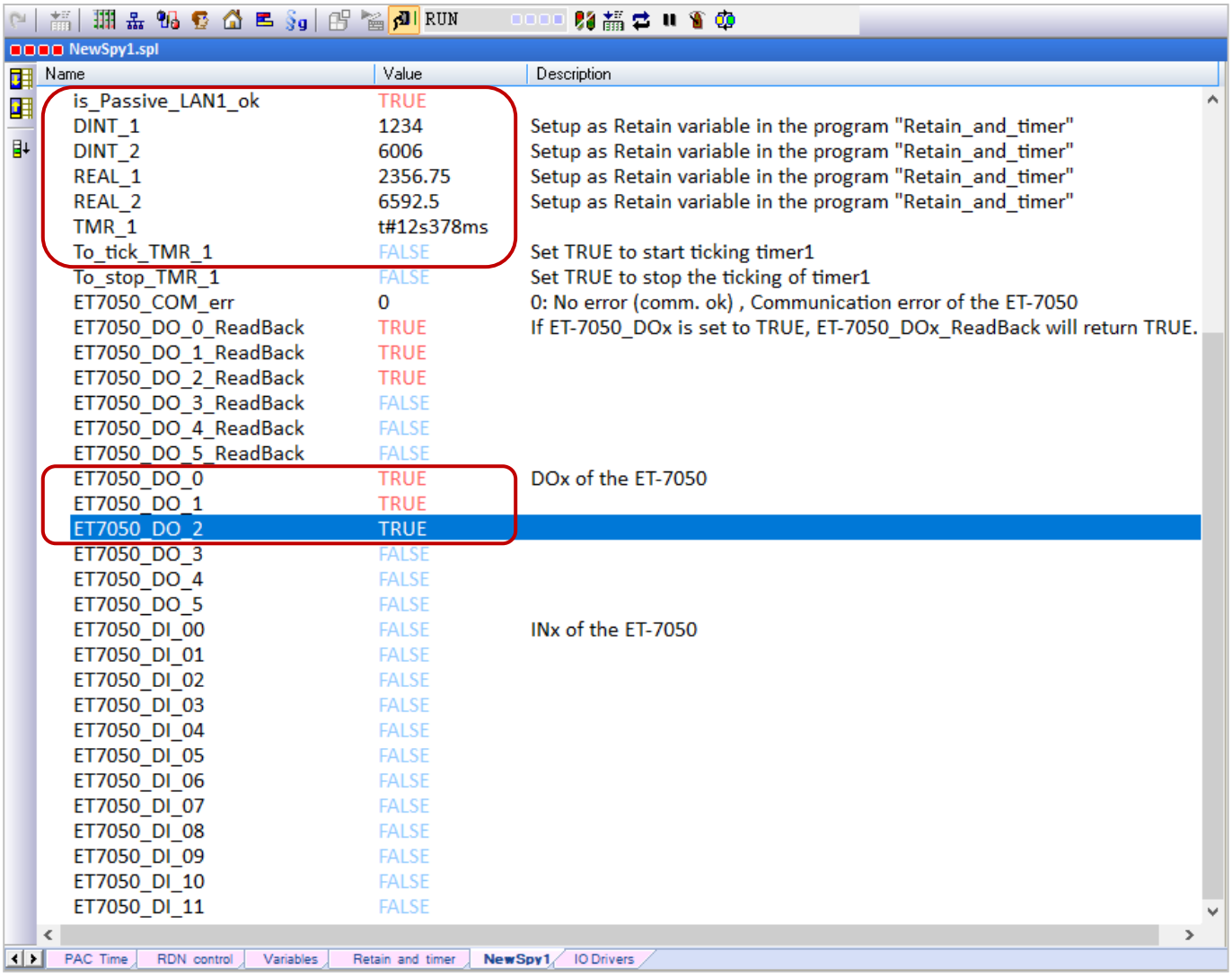
4. View the status of redundant PACs

Click "NewSpy1" to open the spy list, now the Main-PAC is the Active-PAC.



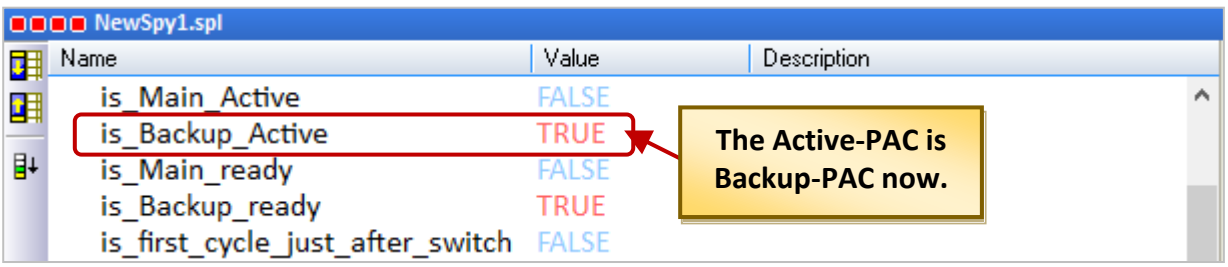
5. Set the value of variables

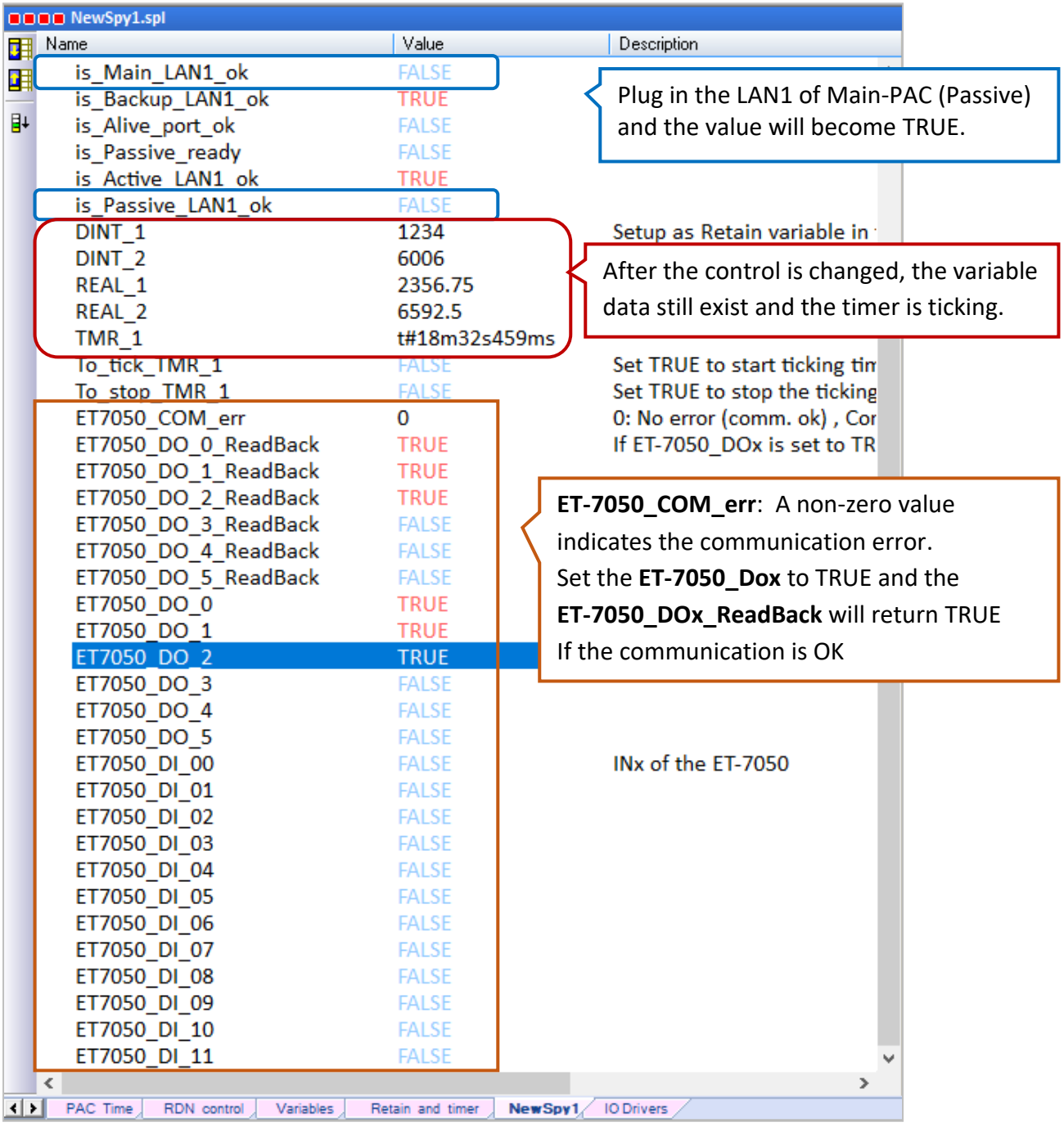
1) Assign values to DINT_1, DINT_2, REAL_1, and REAL _2 variables.
2) Set the To_tick_TMR_1 to "TRUE" to start the "TMR_1" ticking.
3) Set the ET-7050_Do_x to "TRUE" and the ET-7050_DO_x_ReadBack will return "TRUE".
   If the Ethernet cable of ET-7050 is removed, the "ET-7050_COM_err" will return a non-zero value to indicate the communication error.



6. Test the redundant system

1) Make sure the Passive-PAC is ready (i.e., is_Passive_Ready is TRUE).
2) Remove the LAN1 cable of the Active-PAC and the control will pass to the other PAC. In this demo program (RDN_control), the PAC will automatically reboot after a specific time if the LAN1 cable is disconnected.





Test the redundant project (demo_rdn_1)

Description:

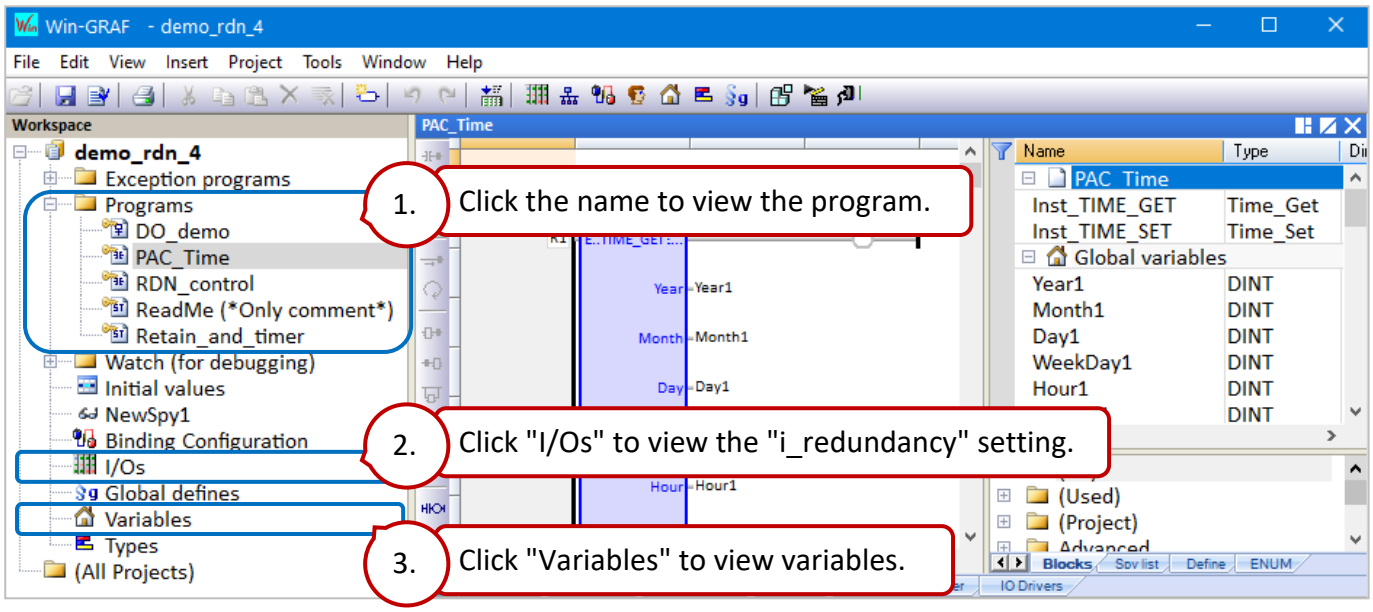
Using both the **COM2 (RS-485)** of two PAC to connect three DCON I/O modules (I-87064, I-87018ZW, I-7065).

1. Before connecting two PACs with remote DCON I/O modules, refer to Chapter 8 to configure each module by using DCON Utility and set the “DCON” I/O board.
2. In the Win-GRAF, view programs and set "i_redundancy“ and "i_redundancy_rs485“ I/O boards. (Refer to Section 14.3.1 I/O Boards Settings).
3. After downloading the Win-GRAF project ("demo_rdn_1"), refer to step5 and step6 of “demo_RDN_3" as noted before to test.

14.4.2 Introduction of the "demo_RDN_4" Project

Description:

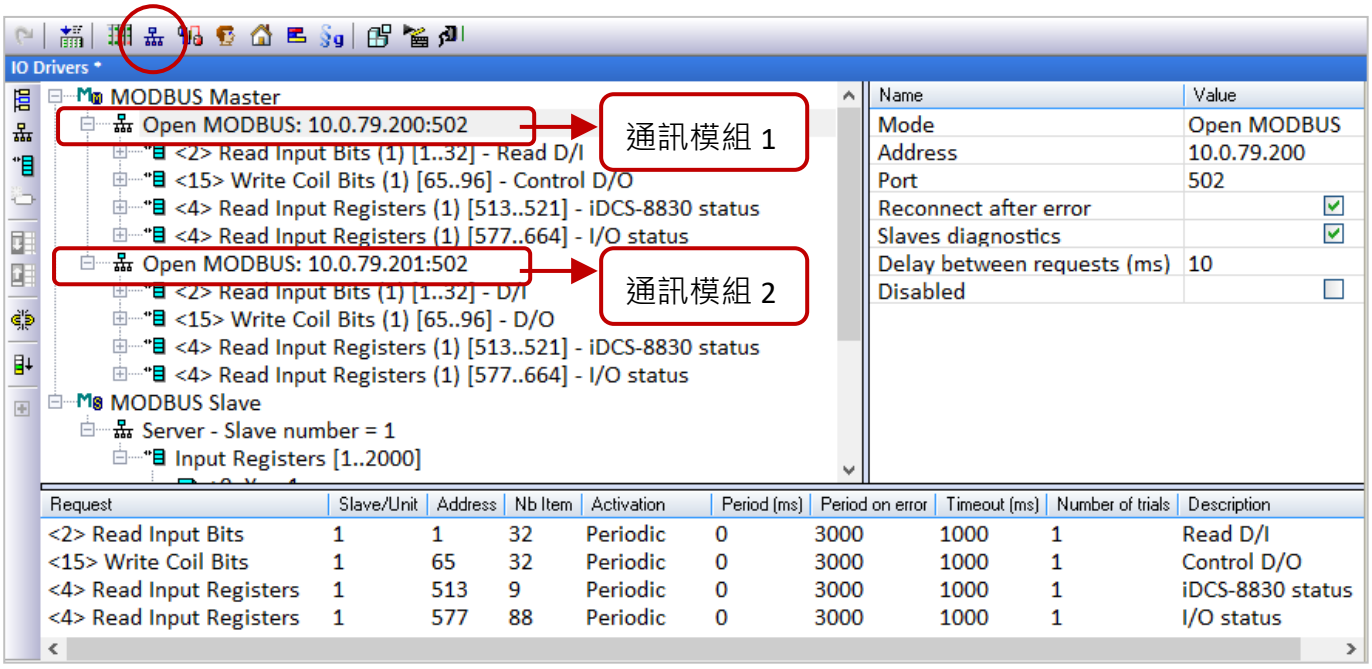
Two PACs are connected with an iDCS-8830 Redundancy I/O unit via **LAN3** through an Ethernet switch.



iDCS-8830 equips redundant power inputs, redundant Ethernet communication modules, and eight I/O slots. In the example, a set of redundant DI modules (i.e., F-8040 in slots 0 and 1) and a set of redundant DO modules (i.e., F-8041 in slots 2 and 3) are used.

<u>The "Modbus Master" Setting</u>

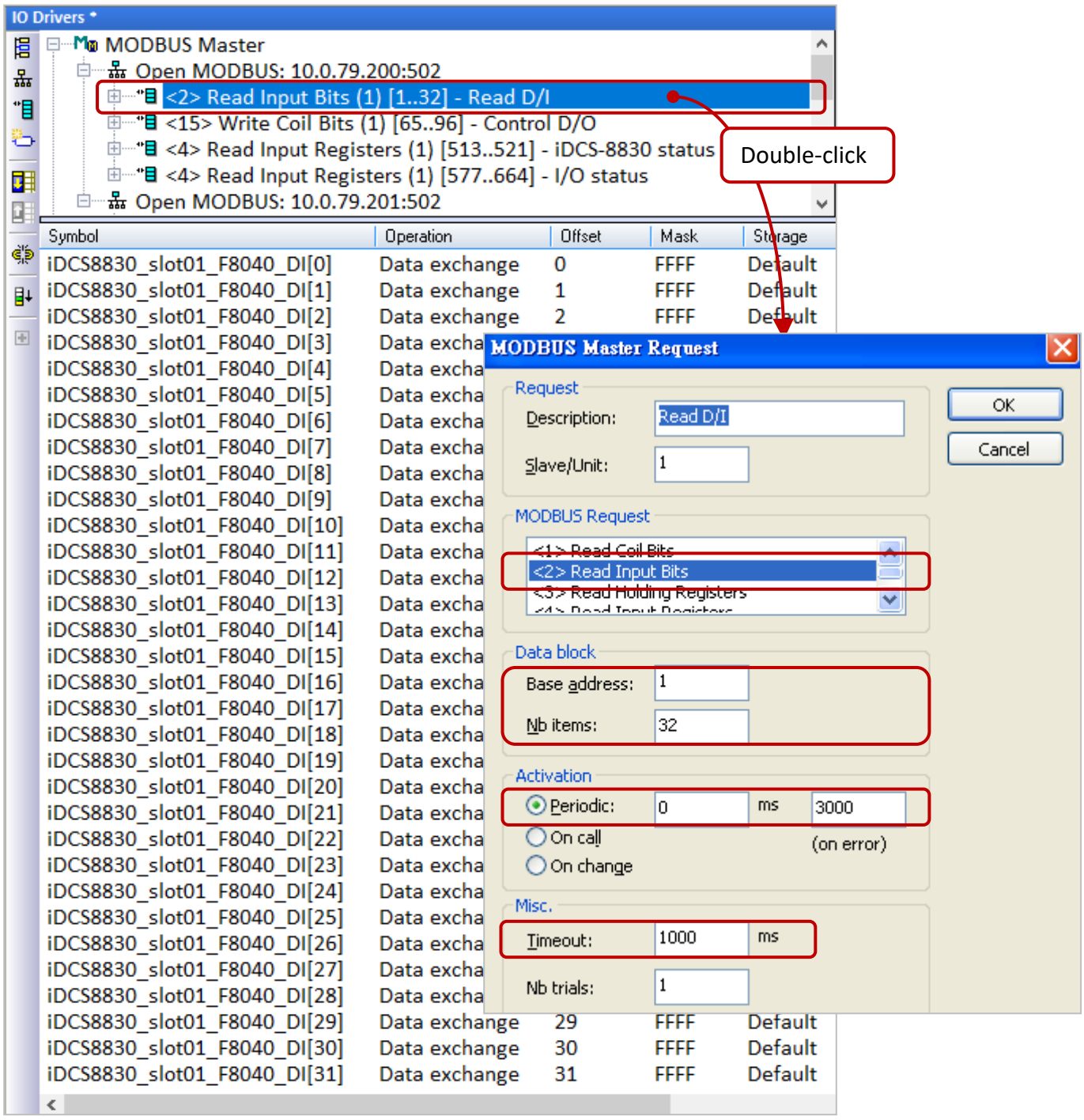
Click the "Open Fieldbus Configuration" button to open the "IO Drivers" window. In this example, the Modbus Master is enabled to connect two Modbus TCP Slaves (i.e., communication modules of iDCS-8830) to read/write data or the communication status.



Note: the IP addresses of LAN3 and communication modules must on the same network segment.

Read DI (F-8040 in slot 0 and 1):

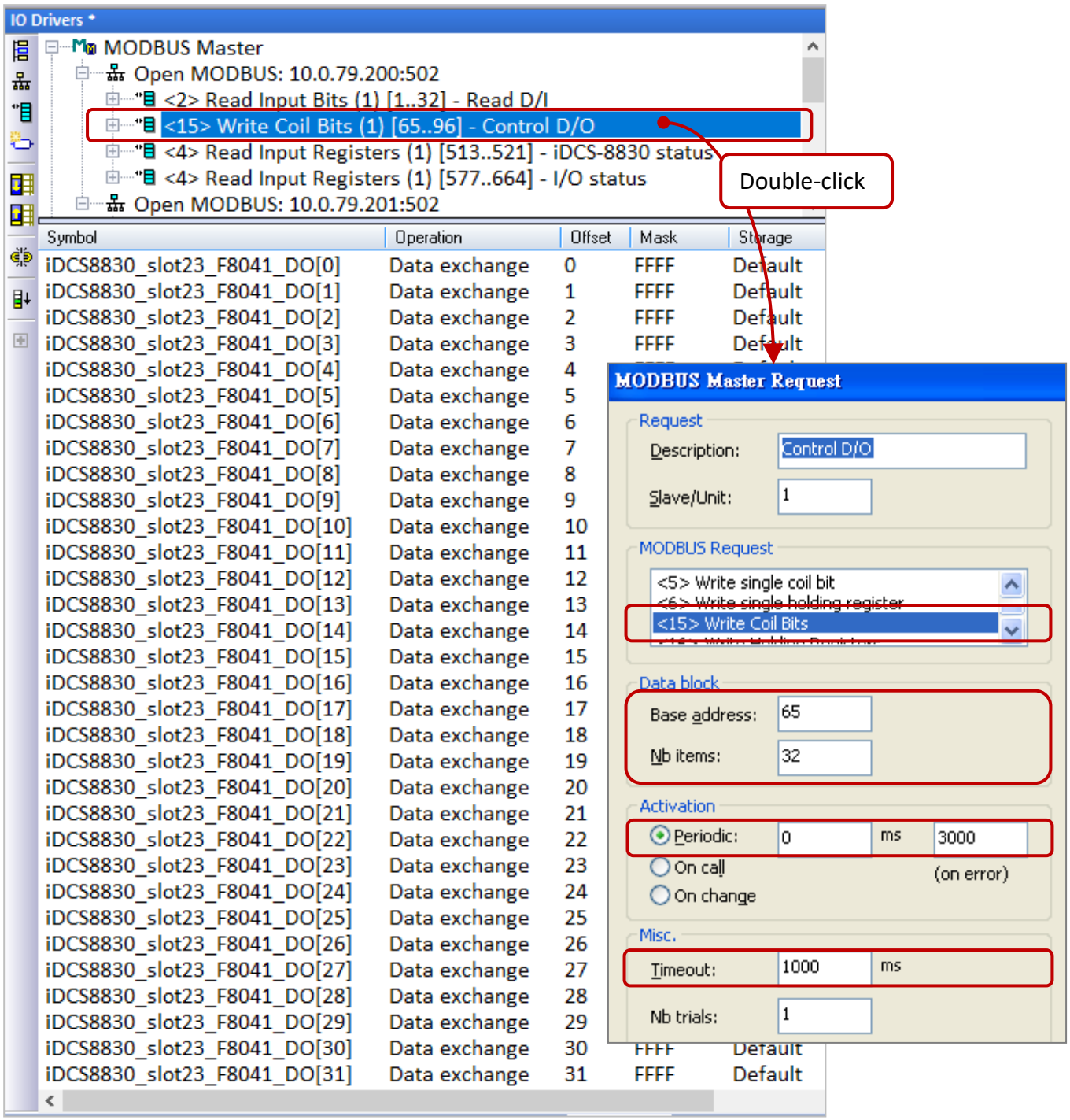
To read **32** DI data start from address **1** and send the request continuously. It will be sent after **3** seconds if an exception occurs. No response for **1** second can be regarded as abnormal.



Note: iDCS-8830 Ethernet redundant I/O unit is used in this example, refer to Section 4.1.1 of the FCM-MTCP software manual to set the "Base address".

Read DO (F-8041 in slot 2 and 3):

To read **32** DI data start from address **65** and send the request continuously. It will be sent after **3** seconds if an exception occurs. No response for **1** second can be regarded as abnormal.

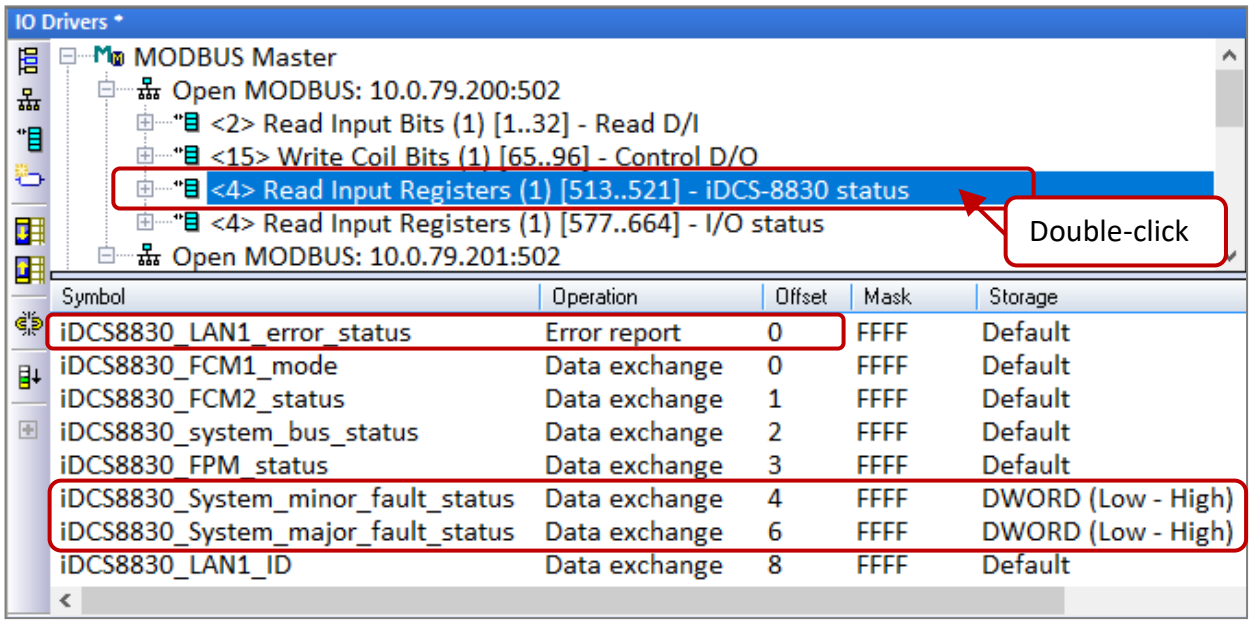


Note: Refer to Section 4.1.2 of the FCM-MTCP software manual to set the "Base address".

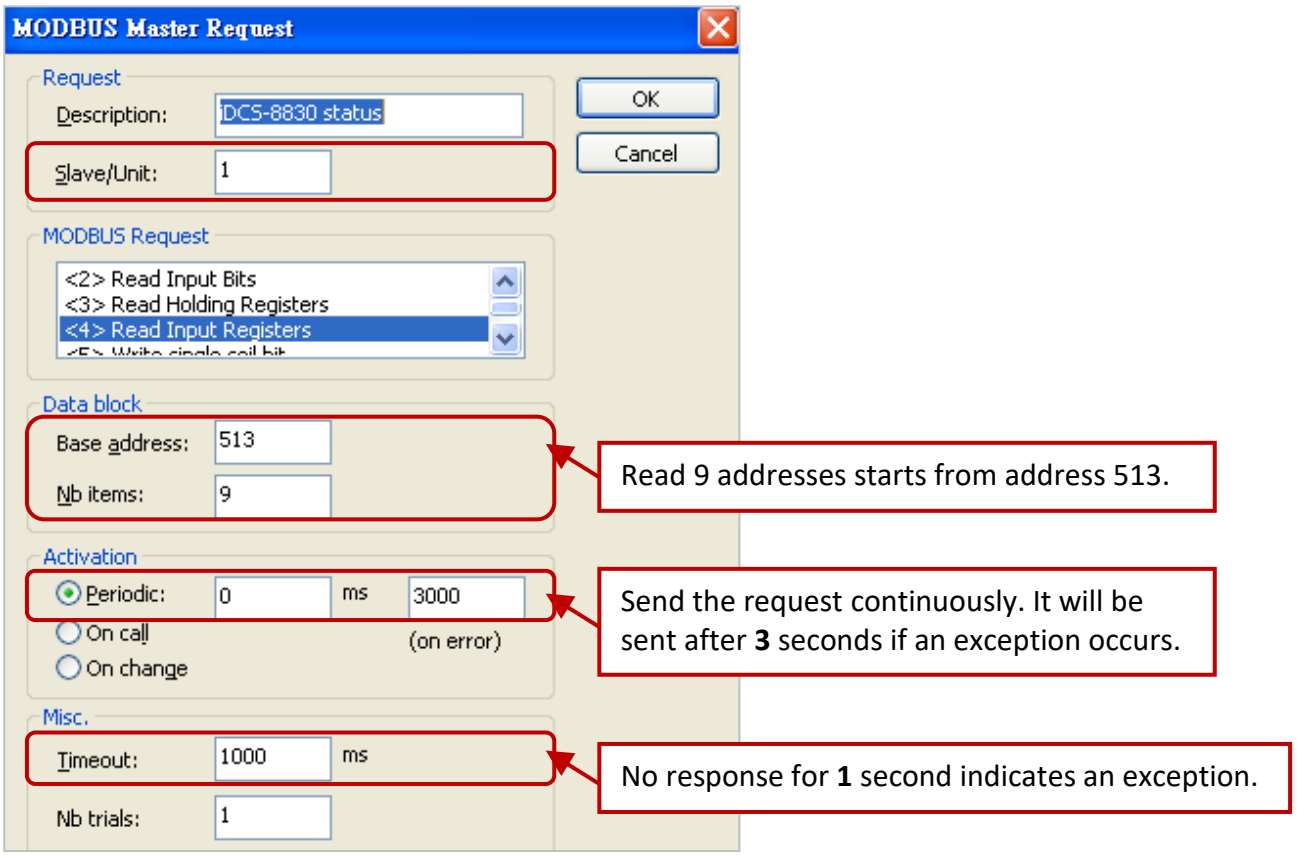
Read the status of iDCS-8830:

- Set the Operation field of "iDCS8830_LAN1_error_status" to "**Error report**" which means to return an error code when fails to read data and be reset to 0 when a successful read. Also, its "Offset" field must be set as "**0**".

- The data type of "iDCS8830_System_minor_fault_status" and "iDCS8830_System_major_fault_status" is "DWORD" (32-bit), so the "Offset" requires two Modbus addresses and the "Storage" must be set as "DWORD (Low-High)".



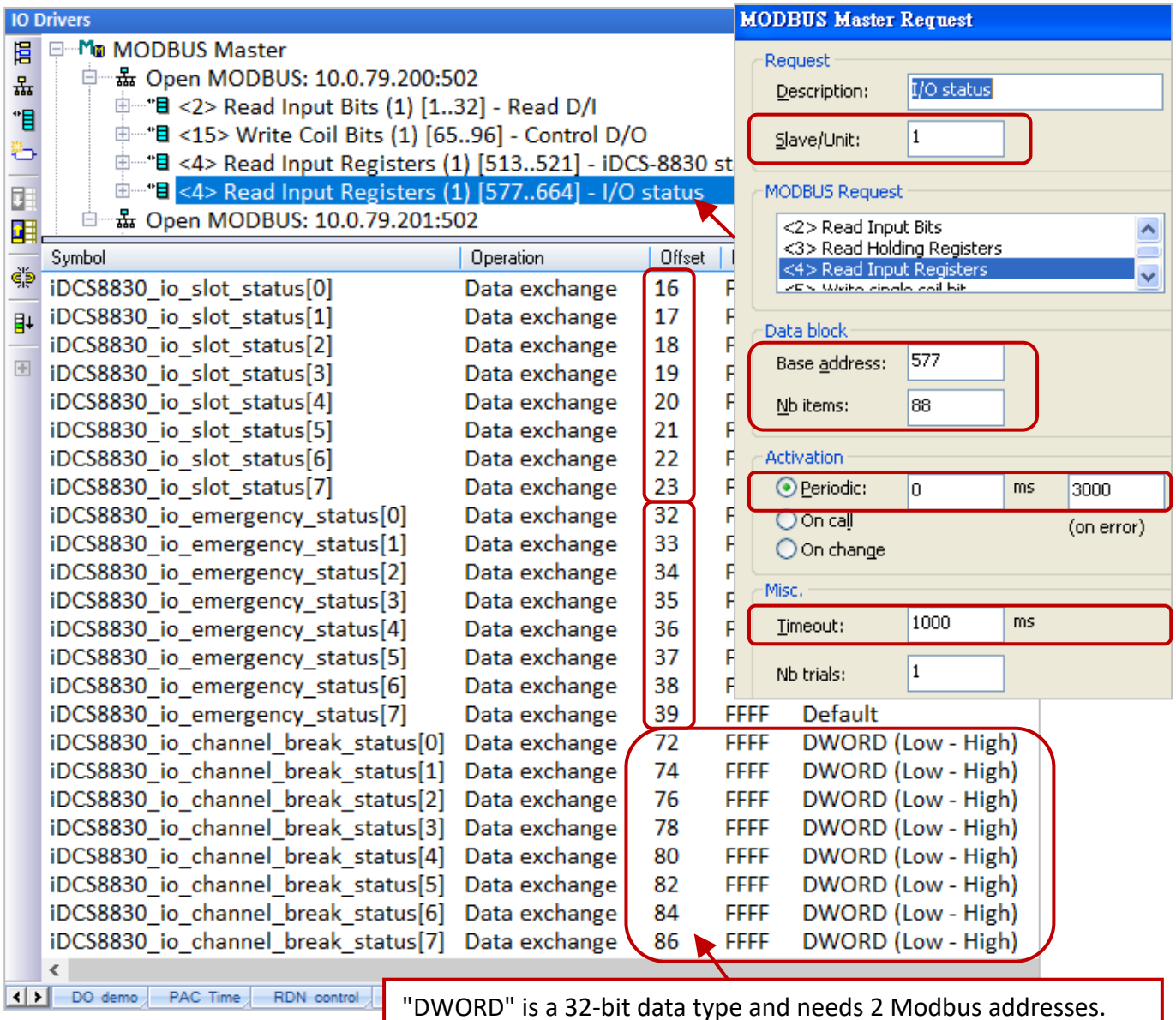
Note: Refer to Section 4.2.1 of the FCM-MTCP software manual to set the "Base address".



Read the I/O status of the iDCS-8830

To read the system information start from address **577** and send the request continuously. It will be sent after **3** seconds if an exception occurs. No response for **1** second can be regarded as abnormal. Refer to Section 4.2.2 of the FCM-MTCP software manual to set the "Base address" and "Offset".

Start Address:	
Module ID:	00577
I/O slot status:	00593
Emergency:	00609
Channel break status:	00649



"DWORD" is a 32-bit data type and needs 2 Modbus addresses. Also, the Storage must be set to"**DWORD (Low – High)**".

Refer to Chapter 5 for more details on Modbus Master settings and Chapter 3 for Modbus Slave settings.

14.4.3 Test the Redundant Project (demo_rdn_4)

Description:

Two PACs are connected with an iDCS-8830 Redundancy I/O unit via **LAN3** through an Ethernet switch.

The following table lists all used devices in this project:

Products	Quantity
RPAC-2658M	2
RS-408	1
iDCS-8830	1
F-8040	2
F-8041	2

Products	Quantity
DN-DI-32DW	1
DN-DO-16DR-A	1
DN-DO-16DR-B	1
CA-37**10**AM (1M Cable) or CA-37**20**AM/**30**AM/**50**AM/**100**AM	4

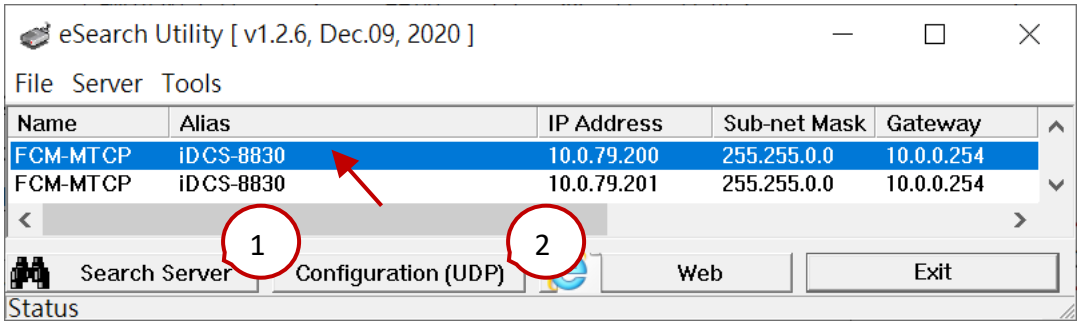
The following describes how to configure an iDCS-8830. Visit the web page to download the software and user manual.

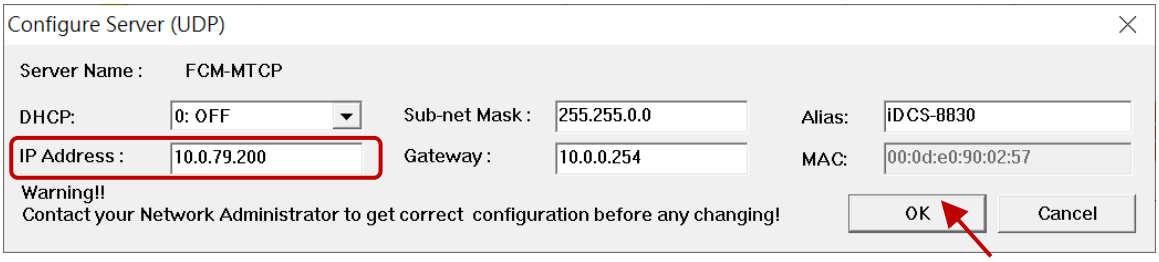
✧ iDCS-8000 Website: https://www.icpdas.com/en/product/iDCS-8830
✧ Download Center: https://www.icpdas.com/en/download/index.php?model=iDCS-8830
✧ Software Manual: https://www.icpdas.com/en/download/show.php?num=1677&model=iDCS-8830

FPM-D2440 * 2: Power module1, Power module2 (for power input, 24V).
FCM-MTCP * 2: MCU1 (set the SW2 to "C", the SW1 to "8", and the IP address to 10.0.79.**200**).
MCU2 (set the SW2 to "C", the SW1 to "9", and the IP address to 10.0.79.**201**).
SW2/SW1 means the fourth octet of IP address of the MCU. ($C8_{16}$ = 200; $C9_{16}$ = 201)

Configure IP addresses by using eSearch Utility:

Open "eSearch Utility" and click "Search Server" to search devices, then click "Configuration" to set the IP addresses. In this example, set two IP addresses to 10.0.79.**200** and 10.0.79.**201** and set the Mask address to "255.255.**0.0**".

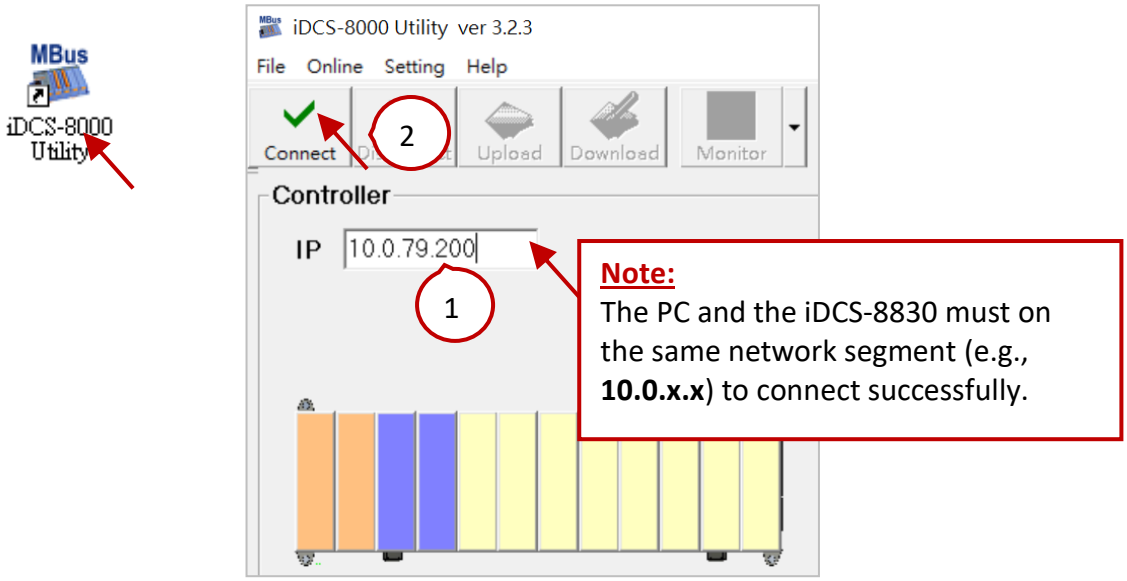




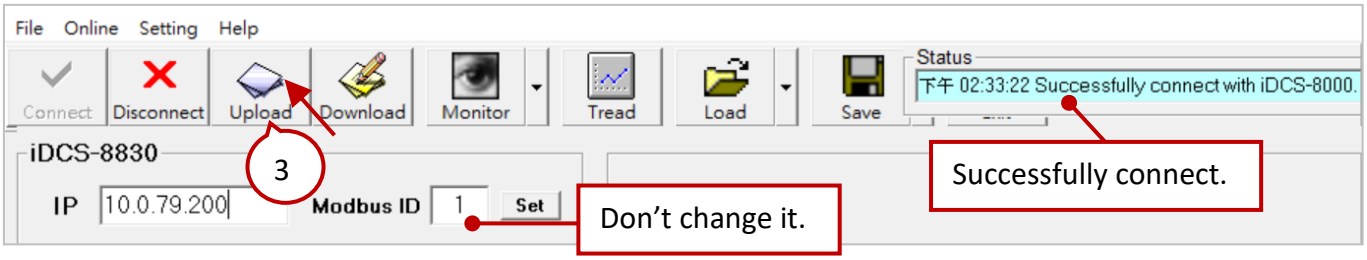
F-8040 * 2: 32-channel **DI** modules which plugged into **slot 0 and 1**.
F-8041 * 2: 32-channel **DO** modules which plugged into **slots 2 and 3**.

Configure I/O modules by using iDCS-8000

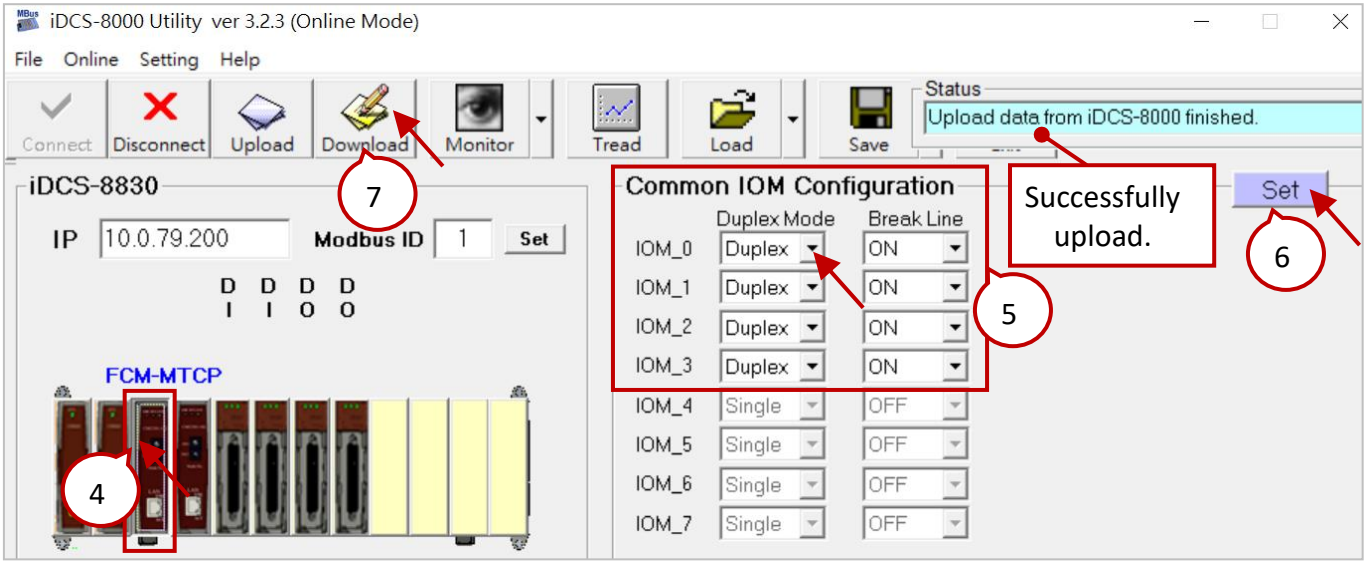
1) Open iDCS-8000 Utility and enter the IP address of the iDCS-8830 to connect.



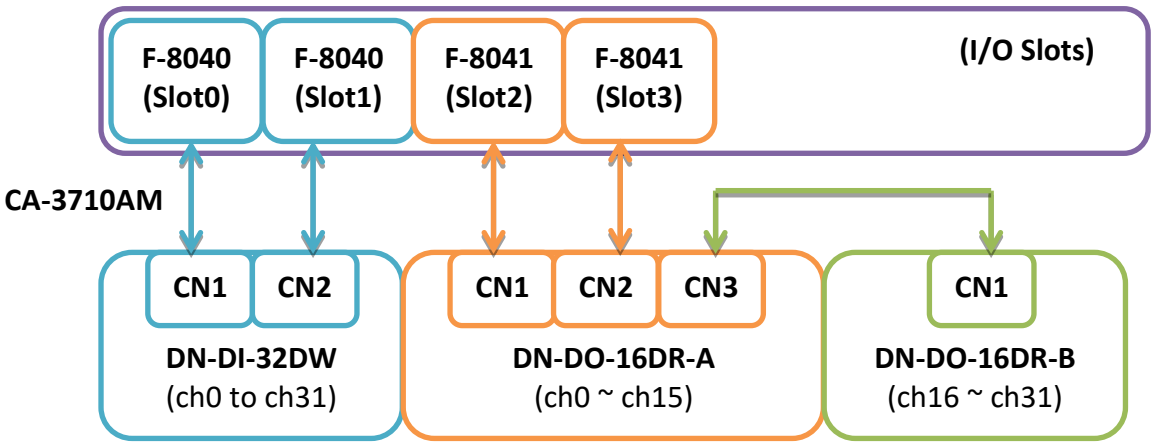
2) Click "Upload" to upload the current I/O settings used in the iDCS-8830.



3) Click the 1st FCM-MTCP (MCU1) and set "IOM_0 ~ 3" (F-8040/F-8041) as "Duplex" Mode.



3) Make sure that F-8040 and F-8041 modules have connected to the termination boards and set the “Break Line” (i.e., Open Wire Detection) as "ON".



5) Click “Set” and click “Download” to download settings to iDCS-8830 and close the iDCS-8830 utility.

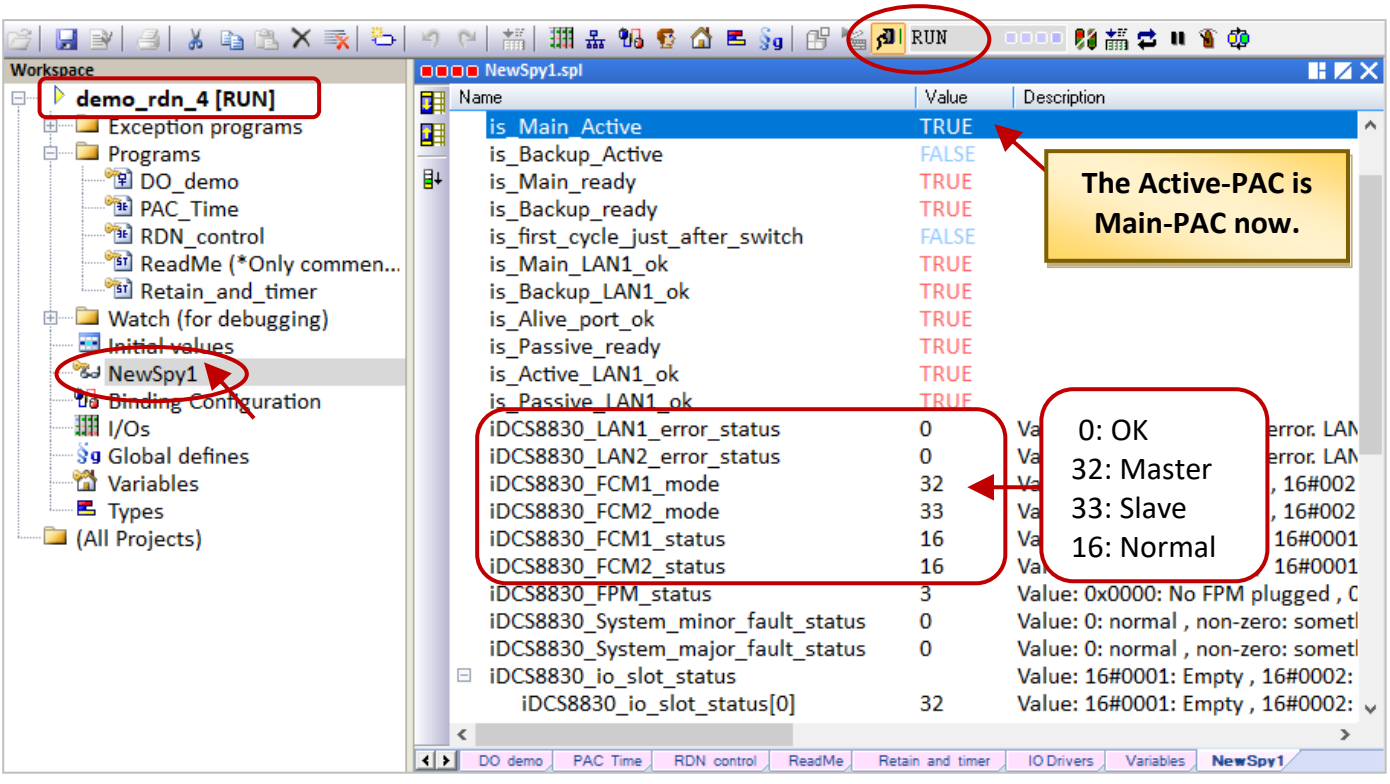
Start testing:

Make sure that all devices are connected properly and modify the Win-GRAF communication IP address, Active_IP address, and Mask address. Then download the project ("demo_RDN_4”) to the PAC. Both the IP address of PAC and PAC must on the same network segment.

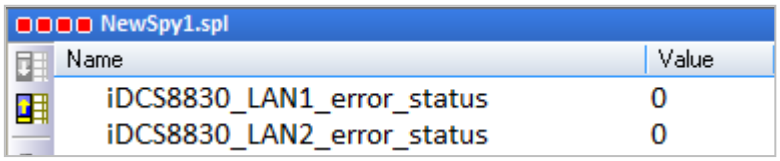
Note:

Power off the Passive-PAC before **updating the project** to the Active-PAC. After completing the update, power on the Passive-PAC to ensure that the redundant system is work properly.

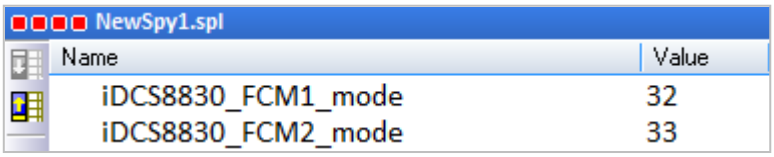
Click "New Spy1" to open the spy list.

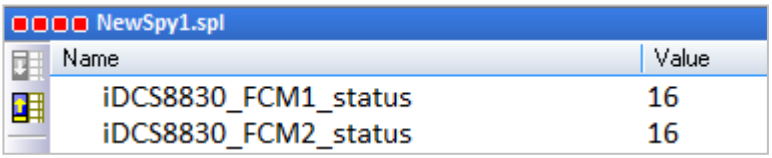


iDCS-8830's Communication Status:

Name	iDCS8830_LAN1_error_status	iDCS8830_LAN2_error_status
Description	The communication status of LAN1	The communication status of LAN2
Value	**0**: OK **Non-zero**: Error	

FCM-MTCP's Redundancy Mode and Status:

Name	iDCS8830_FCM1_mode	iDCS8830_FCM2_mode
Description	The redundancy mode of the FCM1	The redundancy mode of the FCM2
	The start Modbus address: **513**	
Value	**32** (16#0020): Master **33** (16#0021): Slave	

Name	iDCS8830_FCM1_status	iDCS8830_FCM2_status
Description	The status of the FCM1	The status of the FCM2
	The start Modbus address: **514**	
Value	**0** (16#0000): Empty **1** (16#0001): Timeout **2** (16#0002) Undefinded **16** (16#0010) Normal;	

Test Steps

1. Unplug the LAN1 cable and the FCM2 will change to Master (32) and take over.

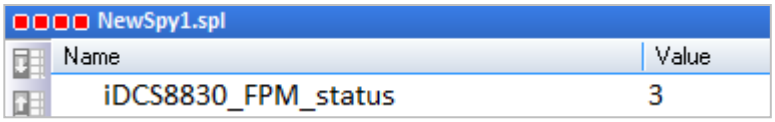
iDCS8830_LAN1_error_status=130	**iDCS8830_LAN2_error_status = 0**
iDCS8830_FCM1_mode = 33	**iDCS8830_FCM2_mode = 32**
iDCS8830_FCM1_status = 1	**iDCS8830_FCM2_status = 16**

2. Plug in the LAN1 cable

iDCS8830_LAN1_error_status=0	**iDCS8830_LAN2_error_status = 0**
iDCS8830_FCM1_mode = 33	**iDCS8830_FCM2_mode = 32**
iDCS8830_FCM1_status = 16	**iDCS8830_FCM2_status = 16**

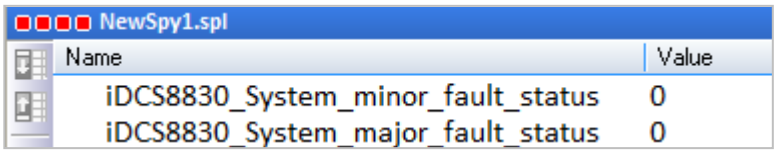
iDCS-8830's Power Status:

Name	iDCS8830_FPM_status
Description	The status of two power modules
	The start Modbus address: **516**
Value	**0**: No FPM is plugged **1**: FPM1 Off / FPM2 Good **2**: FPM1 Good / FPM2 Off **3**: Two FPM are Good.



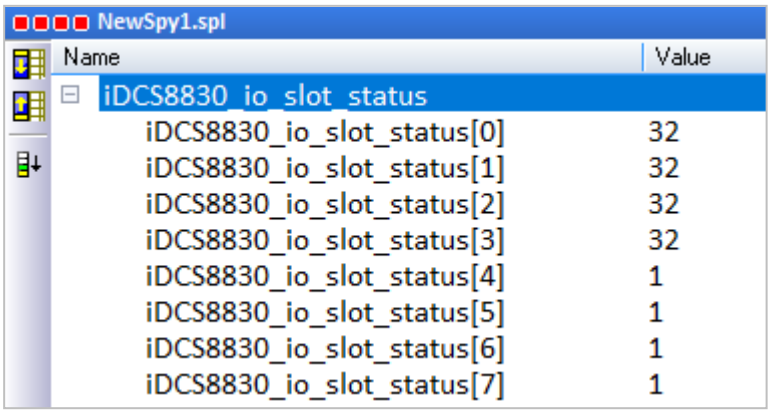
iDCS-8830's Failuure Status:

Name	iDCS8830_System_minor_fault_status	iDCS8830_System_major_fault_status
Description	The status of a glitch or major failure of the system	
	The start Modbus address: **517 / 518**	The start Modbus address: **519 / 520**
Value	**0**: Empty **Non-zero**: Something wrong, refer to Section 4.2.1 of the FCM-MTCP software manual	

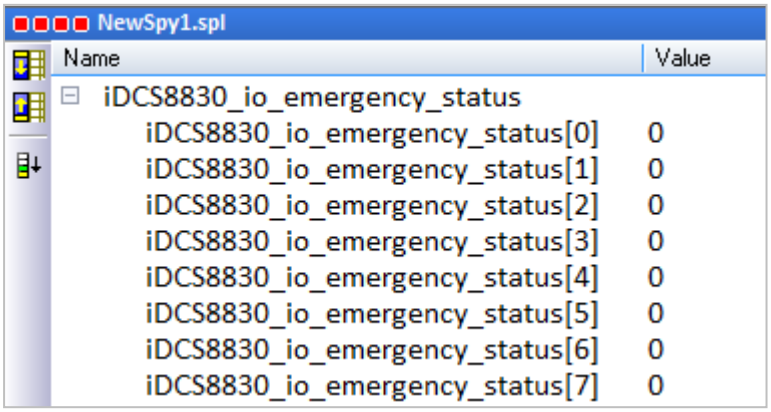


The Status of I/O Slots

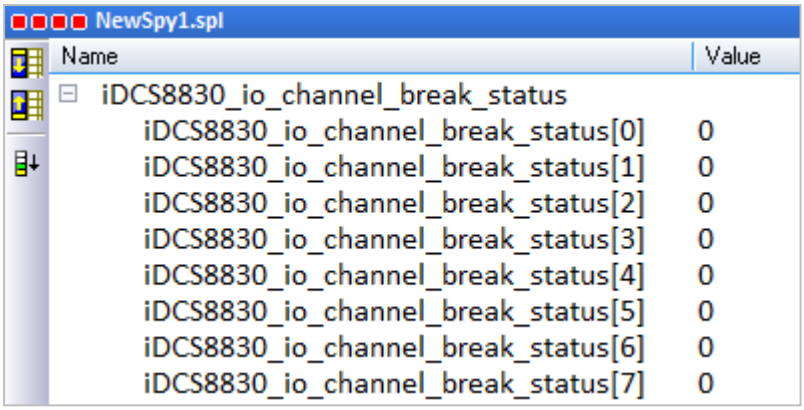
Name	iDCS8830_FPM_status
Description	The status of I/O slots 0 to 7
	The start Modbus address: **593**
Value	**1** (16#0001): Empty **2** (16#0002): Halt **4** (16#0004): Bootup **8** (16#0008): Bootloade **16** (16#0010): Pre-operation **32** (16#0020): Operation **64** (16#0040): Stop



The Status of I/O Emergency:

Name	iDCS8830_io_emergency_status
Description	The Emergency status of two power modules **Note:** "iDCS8830_io_emergency_status[x]" only works when the "IOM_x" is set to "**Duplex**".
	The start Modbus address: **609**
Value	**0**: OK, **32** (16#0020): CJC Error **256** (16#0100): Cable Break-off 

The Status of I/O Channels:

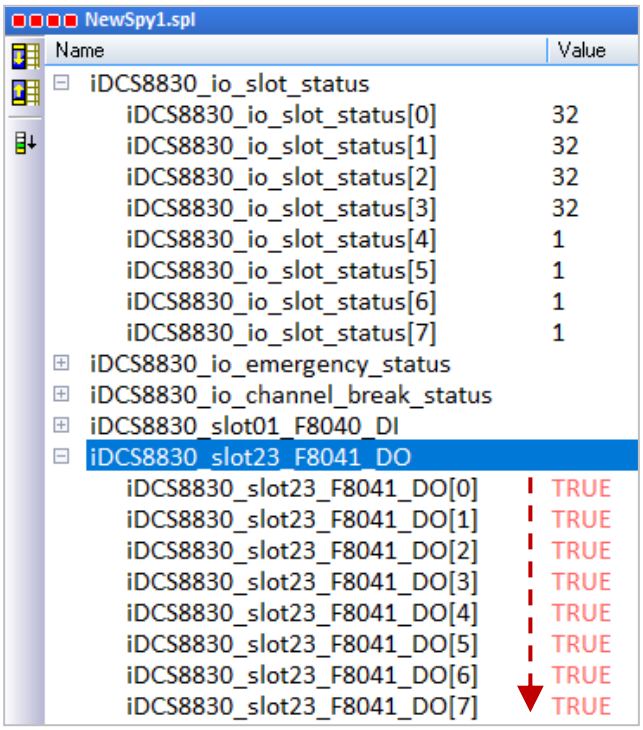
Name	iDCS8830_io_channel_break_status
Description	Open wire detection for I/O channels 0 to 7
	The start Modbus address: **649**
Value	**0**: Normal, **Non-zero:** The corresponding bit is 1, which means the wire is disconnected. 

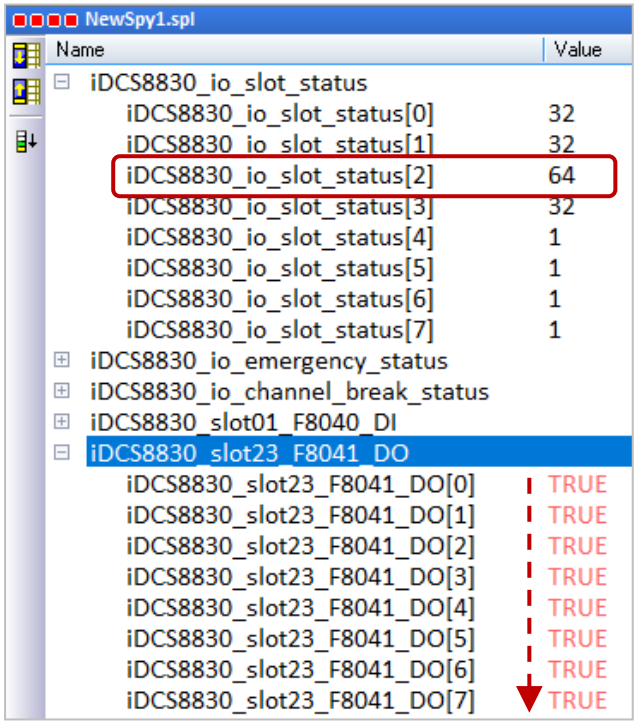
The Status of F8040 (DI) and F8041 (DO) Modules

Name	iDCS8830_slot01_F8040_DI	iDCS8830_slot23_F8041_DO
Description	The status of F8040 (32 DI) modules on I/O slots 0 and 1	The status of F8041 (32 DO) modules on I/O slots 2 and 3
	The start Modbus address: **1**	The start Modbus address: **65**
Test Steps		

At first, the LEDs (DO0 – 7) of the F-8041 module on slot2 will light up sequentially.

1. Unplug the F-8041 module on slot2 and, the F-8041 module on slot3 will take over and light up LEDs in turns.
2. At this time, the status value of the iDCS8830_io_slot_status[2] is changed to "64" which means the module on slot2 stops working. Plug the module into the I/O slot again.





The above is the description of testing the iDCS-8830 redundant I/O, also refer to Section 14.4.1 to test the RPAC redundant system.

Chapter 15 Schedule Control

Introduction:

All Win-GRAF PACs support the Schedule-Control function. One PAC can control the scheduling for up to 10 Targets. Each Target can control one boolean (BOOL), integer (DINT), and real (REAL) variables and up to 5 weekly schedules can be set for each year or four seasons. Also, up to 15 periods of time can be set for one schedule.

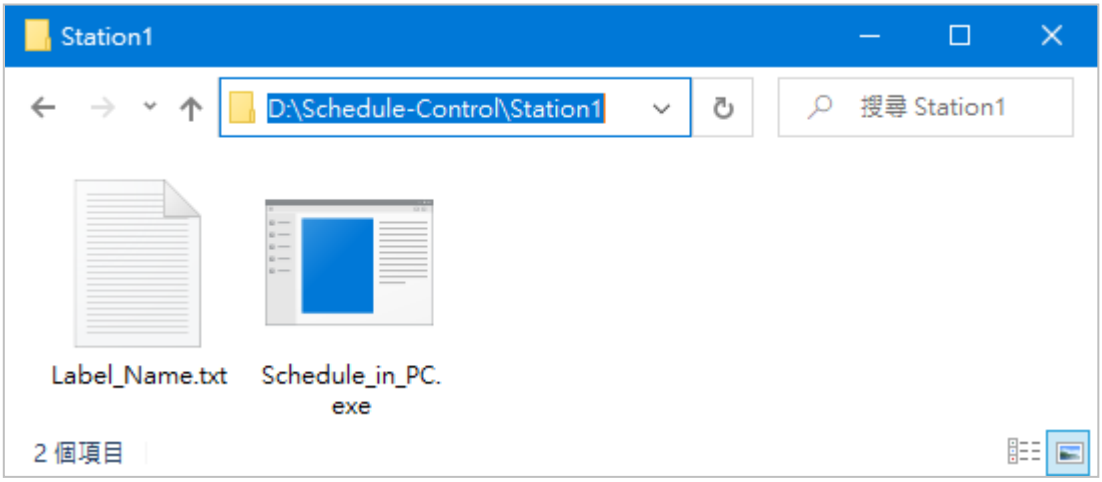
ICP DAS provides free software – "Schedule-Control Utility" for users to simply and quickly complete the scheduling configurations.

15.1 Install the Schedule-Control Utility and Restore the Win-GRAF Project

The user can download the Schedule-Control Utility ("**Schedule_in_PC.zip**") on the website.

Install the Schedule-Control Utility:

Extract the zip file and copy Schedule_in_PC.exe and label_name.txt to your PC. It is recommended to store files in the project directory (e.g., D:\Schedule-Control**Station1**\).

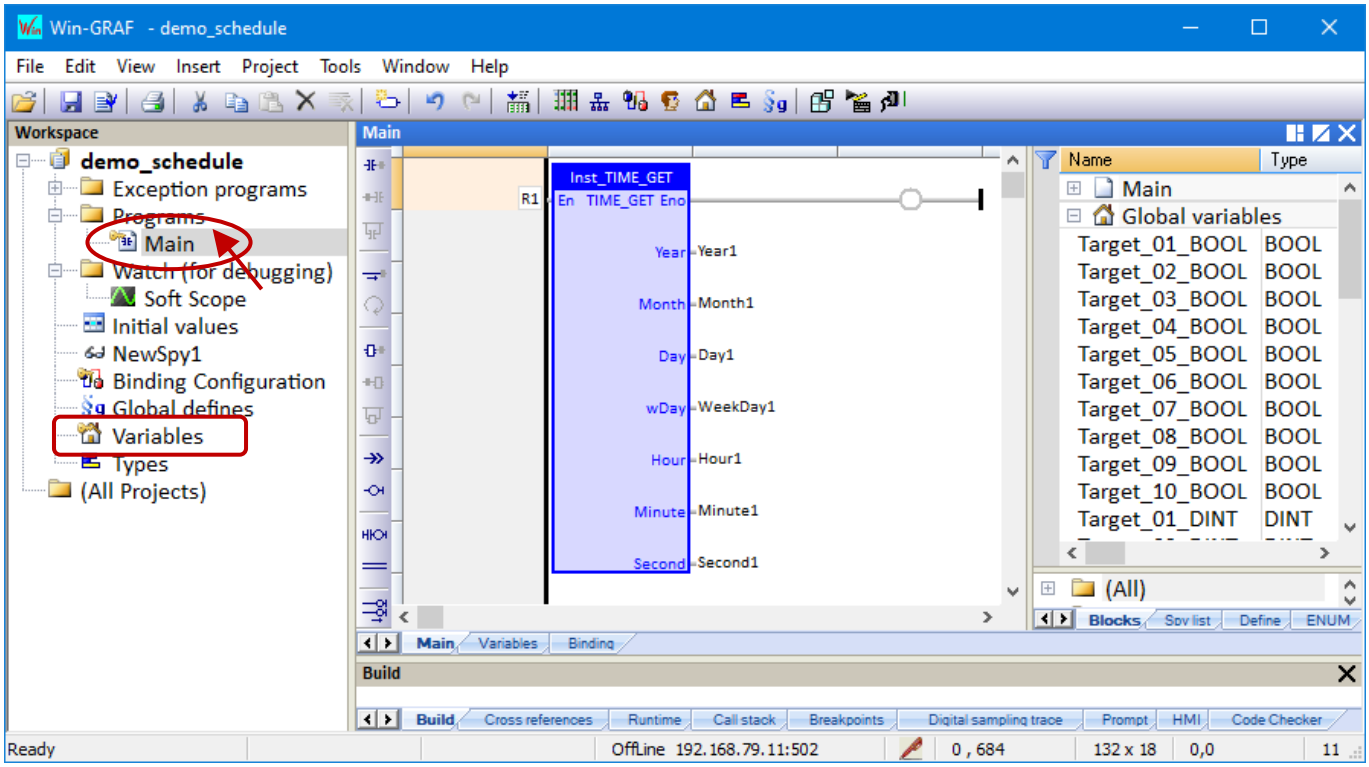


Restore a Win-GRAF Demo Program:

Download the demo program on the website (**demo_schedule.zip**). Click the menu commands **File - Add Existing Project - From Zip...** in the Workbench to open the project. (Refer to Section 11.4)

15.2 Introduction of the “demo_schedule” Project

This project includes one LD program (Main) that used to get the system time.

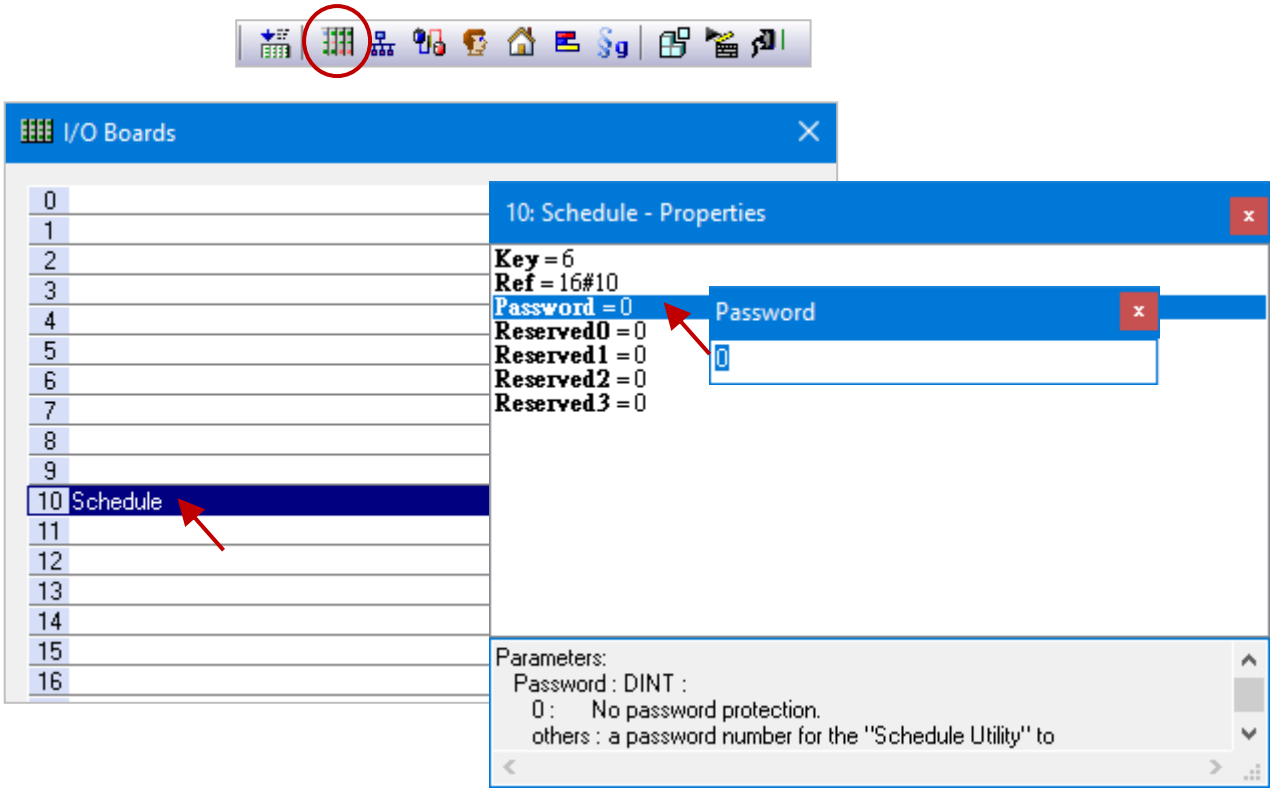


Double-click "Variable" can view or set variables.

Name	Data Type	Description
Target_01_BOOL to Target_10_BOOL	BOOL	For ten targets to control Boolean variable
Target_01_DINT to Target_10_DINT	DINT	For ten targets to control DINT variable
Target_01_REAL to Target_10_REAL	REAL	For ten targets to control REAL variable
Year1	DINT	Used for “TIME_GET” function block
Month1		
Day1		
WeekDay1		
Hour1		
Minute1		
Second1		

15.2.1 "I/O Boards" Setting (Schedule)

To enable the Schedule-control on the PAC, the user must add the "Schedule" function in the "I/O Boards" window (refer to Chapter4)



Parameters:

Password: DINT:

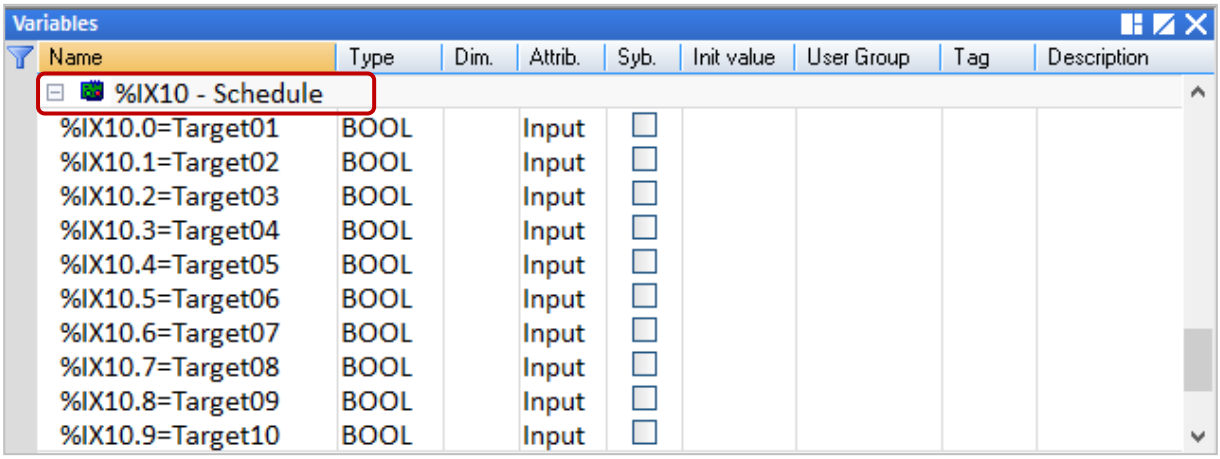
0: No password protection.

Others: A password number for the "Schedule Utility" to access the Win-GRAF PAC.

10-Ch Boolean Inputs: (TRUE: Enabled; FALSE: Disabled)

To indicate the related target of the schedule-control is enabled or disabled. For example, if Ch.0 and Ch.5 return TRUE, it means the Target 1 and Target 6 of the schedule control are enabled.

The password is “0” in this example. If a password is set, the user will be asked for entering the password before downloading the settings to the PAC by using “Schedule-Control Utility”. After adding the "Schedule" I/O board, there are 10 Boolean variables that are used to display the status of the schedule-control will automatically be added.

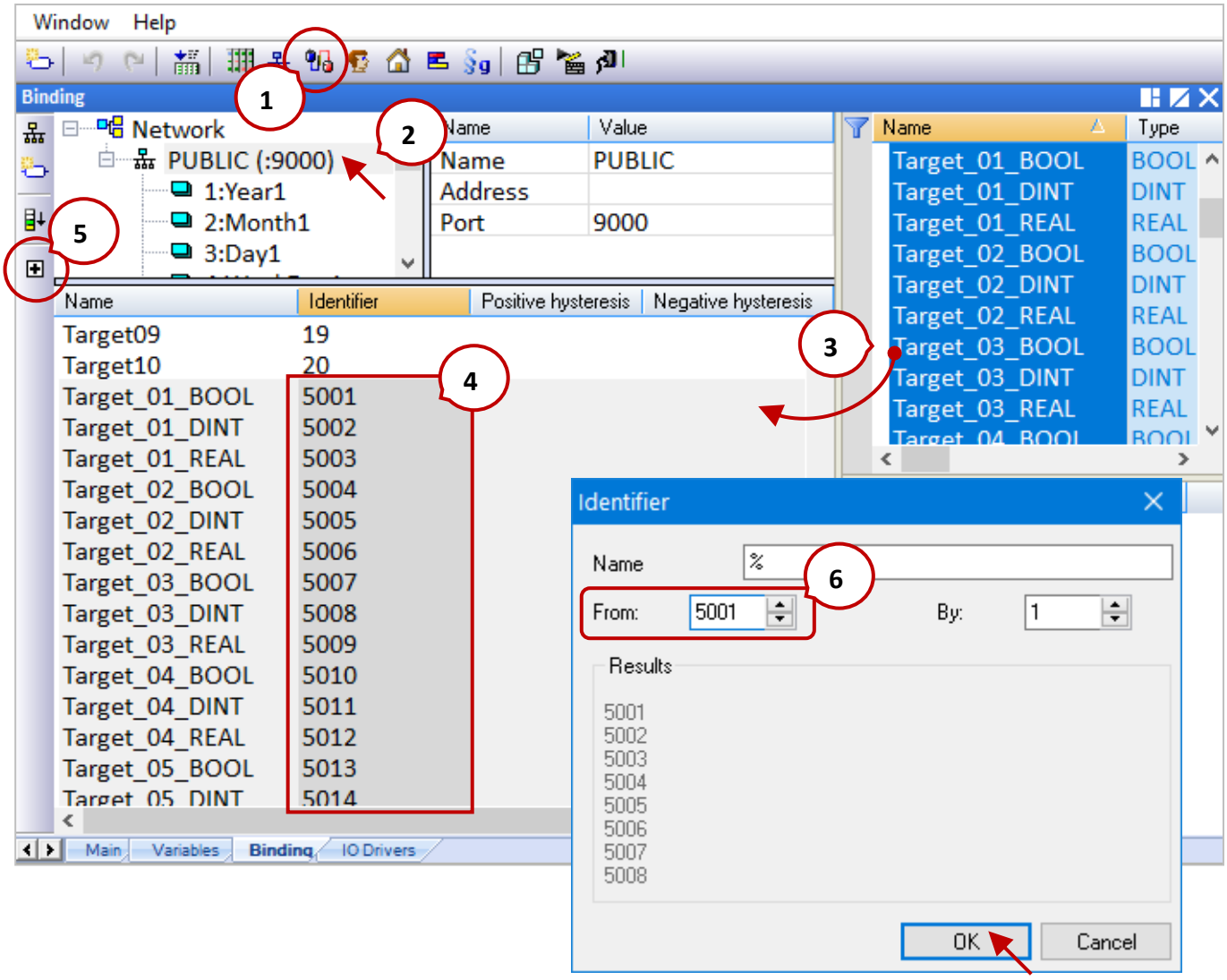


15.2.2 Public the Variable Data (Data Binding)

To public Win-GRAF variables for other PACs to read/write data, add the needed PUBLIC variables in the “Binding” window and specify the identifier, refer to Chapter 7. And, the ID of control variables must be set to **5001 to 5030**.

Follow the steps:

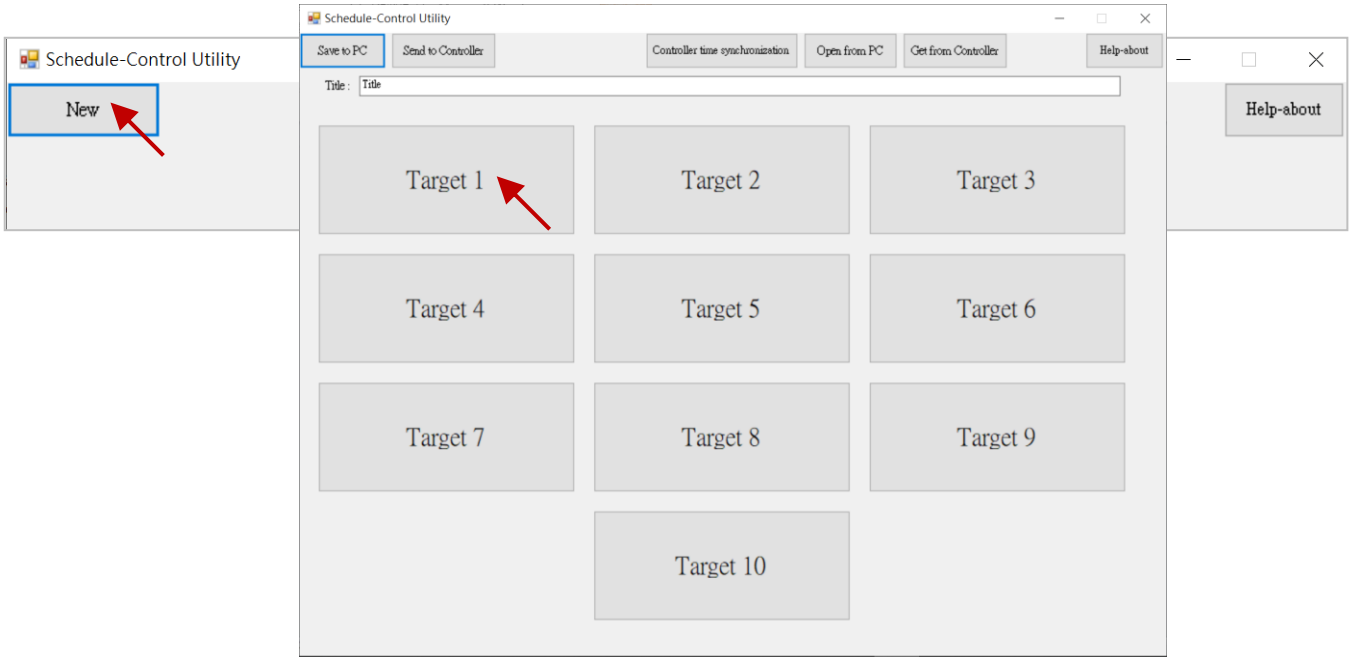
1. Click the **Open Binding Configuration** button to display the **Binding** window.
2. Click on the **PUBLIC (:9000)** setting.
3. Drag variables to be published to the ID mapping area.
4. Select multiple **Identifier** fields that you want to set a continuous number.
5. Click the **Iterate Property** button.
6. Enter the starting ID number in the **From** field.



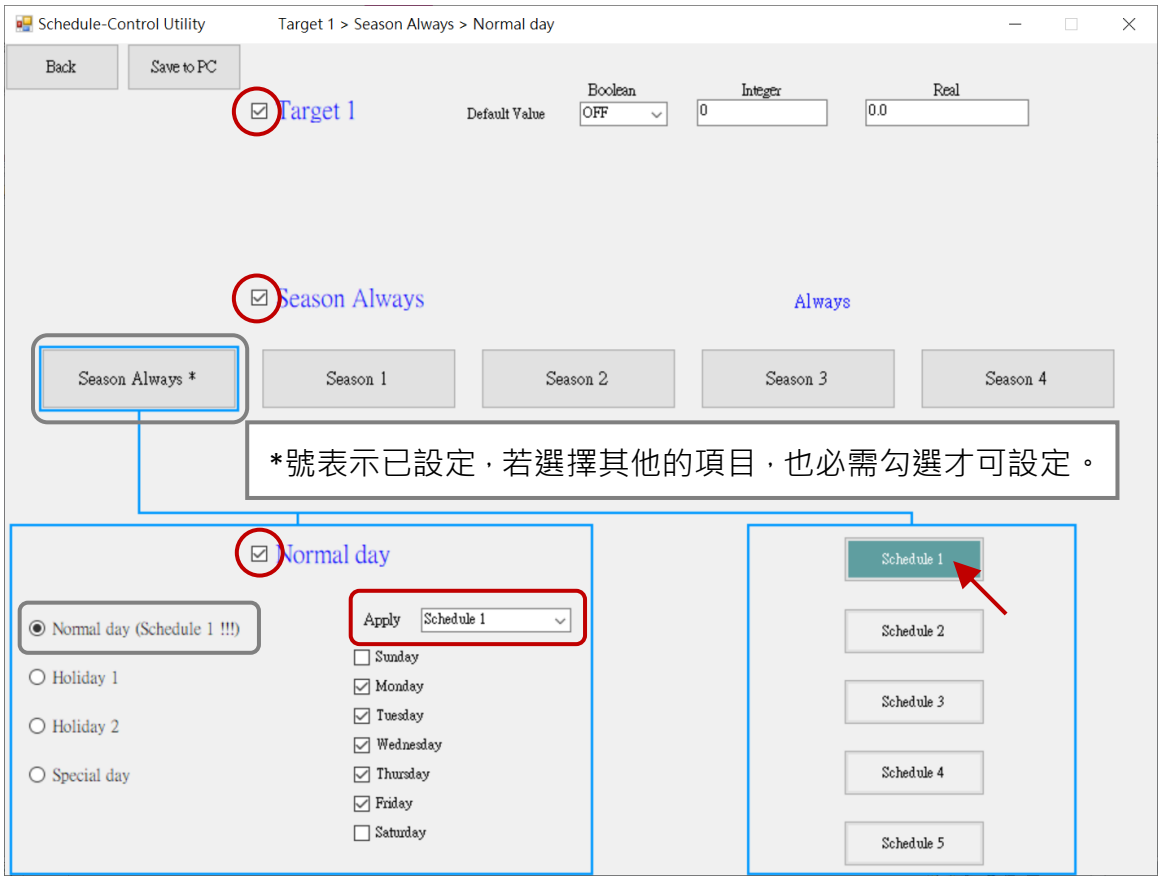
15.3 Description of the Schedule-Control Utility Example

This example briefly describes how to set Target 1, also refer to Section 15.5 to learn more about the settings.

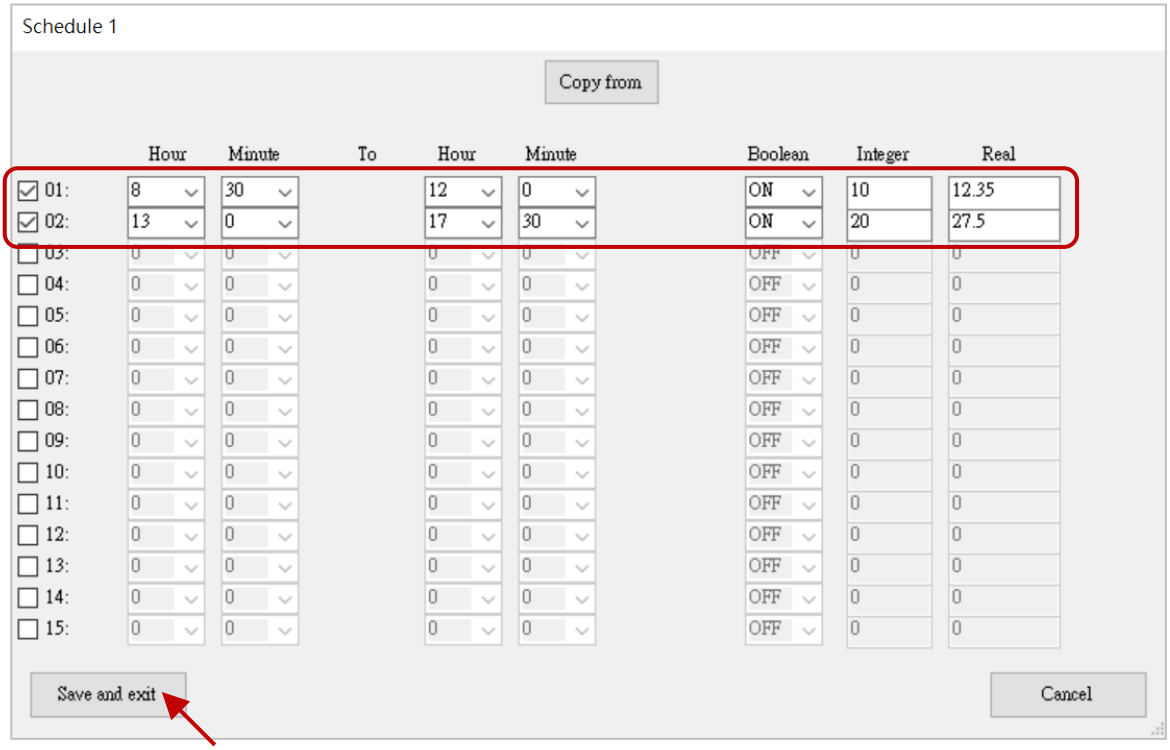
1. Execute the Schedule-Control Utility (Schedule_in_PC.exe) and click the **New** button and then click **Target 1** to display the settings screen.



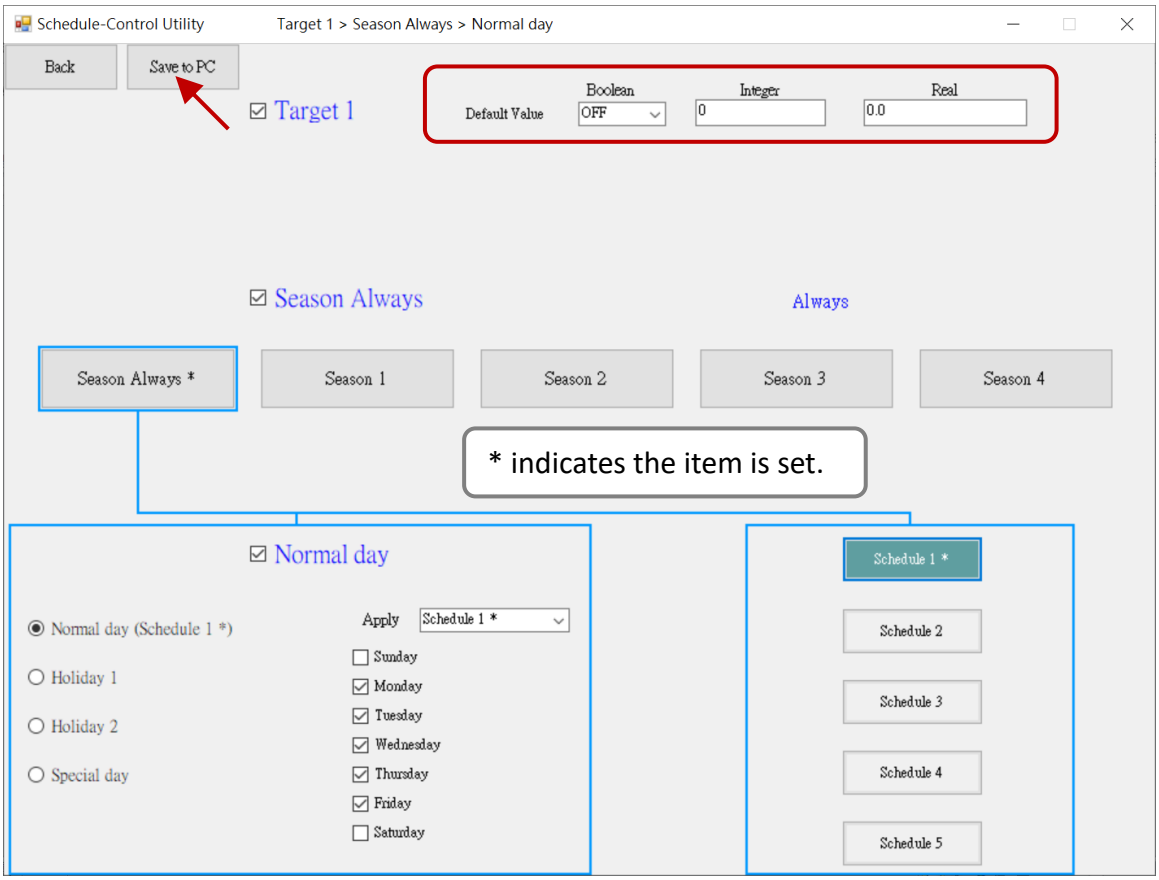
2. Tick the **Target 1**, **Season Always**, and **Normal day** to enable the setting.
3. Select the scheduling options you want to use (ex: Schedule 1) and click the **Schedule 1** button to open the window.



4. Each schedule can set up to 15 periods of time, check the box to set the desired time and the value of control variables. After completing the setting, click the "Save and exit" button to save and exit the window.



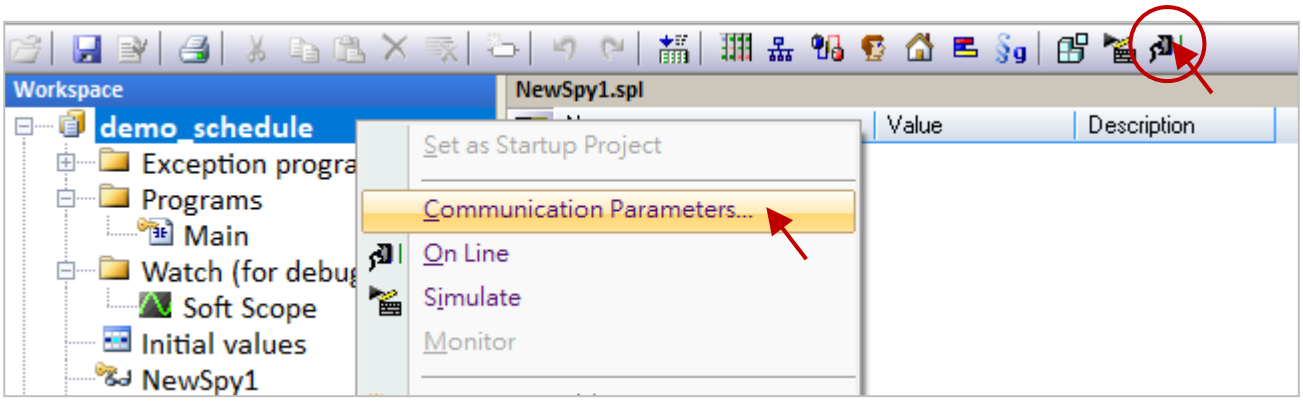
5. The “Default Value” settings will be applied if the time is not in the scheduled period. Click the “Save to PC” button to save the configuration file (e.g., test1.txt).



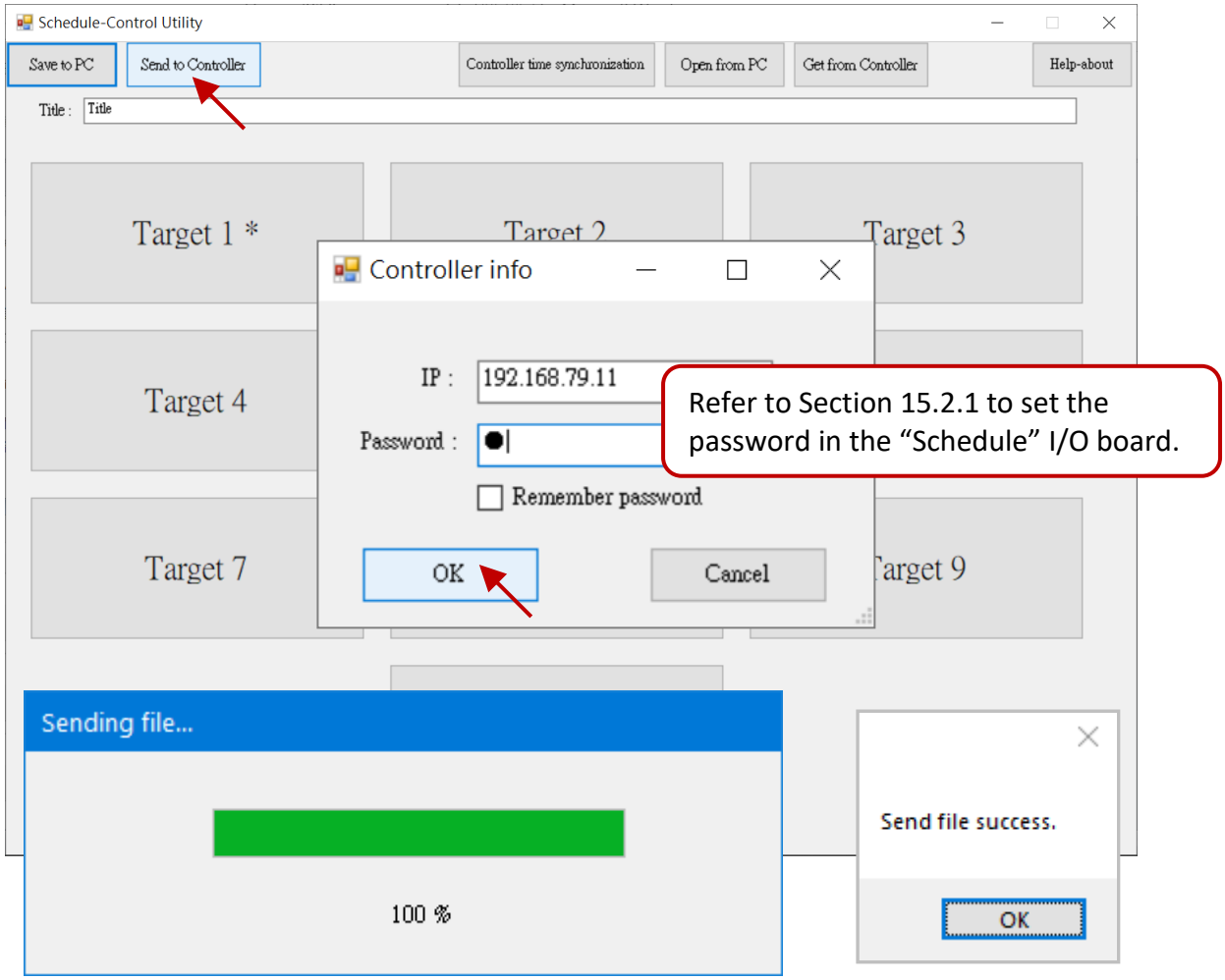
15.4 Testing the Schedule-Control on the PAC

Before testing, download the scheduling settings and the Win-GRAF project to the Win-GRAF PAC.

1. First, download the Win-GRAF project (demo_schedule) to the PAC.
   Modify the communication IP address and click the **On Line** button to download the project.

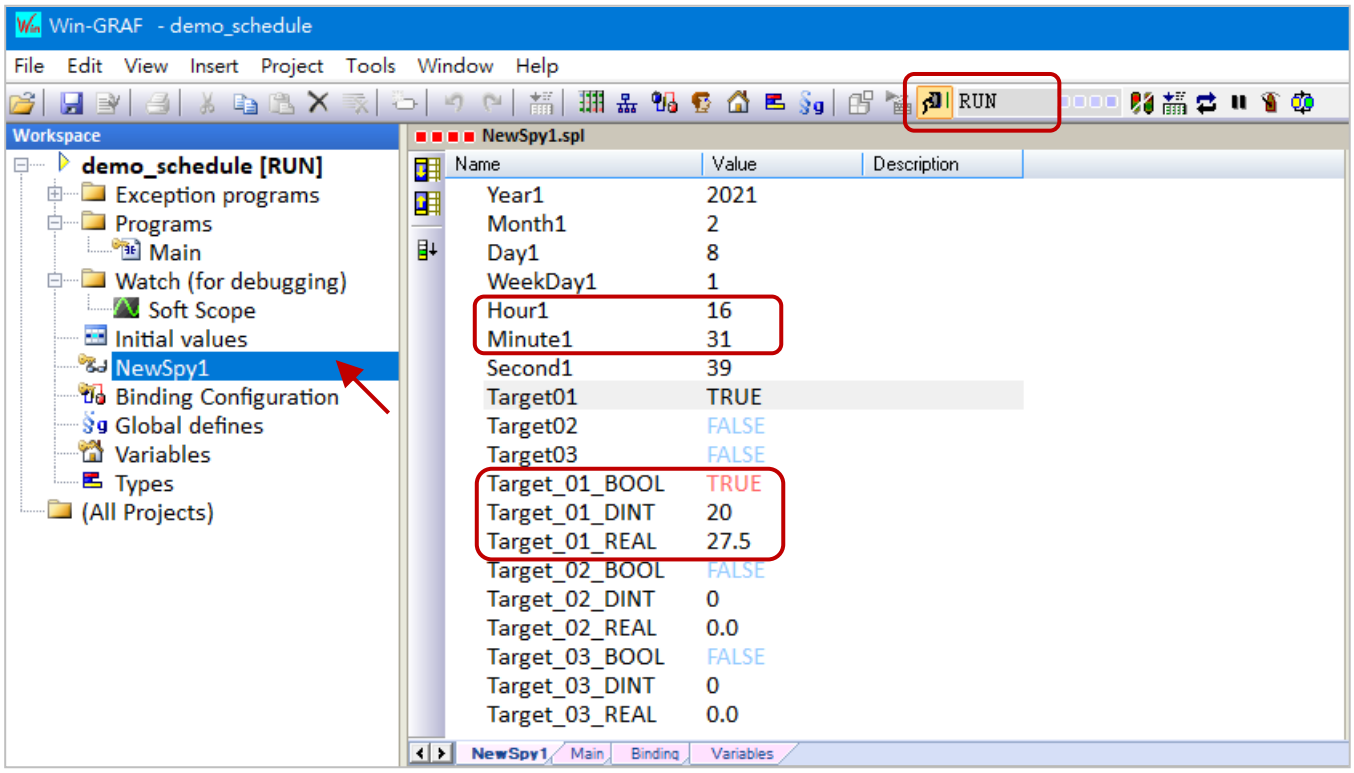


2. Download the scheduling settings to the PAC by using Schedule-Control Utility.
   Click the **Send to Controller** button, enter the IP address of the PAC and the password for the scheduling function (in this case, 0), and click OK. After sending files, the "Send file success" dialog will be displayed and the utility will be automatically closed.



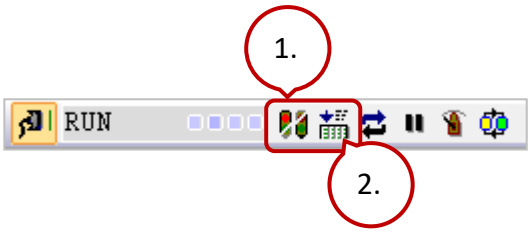
3. Test the scheduling. Open the Win-GRAF "NewSpy1" window and you can see "Target_01 = TRUE" which means **Target 1** is enabled. In the example, we set two periods of time.

01:	08:30 ~ 12:00	Boolean=ON,	Integer=10,	Real=12.35
02:	13:00 ~ 17:30	Boolean=ON,	Integer=20,	Real=27.5



Note:

If the scheduling setting is changed and downloaded to the PAC, you can reboot the PAC or click the "Stop application" button and click the "Download" button in Win-GRAF to apply the settings.



15.5 How do I Use the Schedule-Control Utility

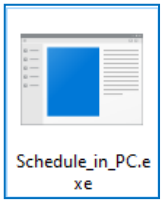
15.5.1 Address for Control Variables (BOOL, DINT, and REAL)

Up to 10 targets can be configured in the Schedule-Control Utility. Each Target can control one BOOL variable, one DINT variable, and one REAL variable.

To publish the following variables for other devices to access data, all variables must be declared in the Variables window (refer to Section 15.2) and add them into the Win-GRAF **Binding** widow also specify the identifier that must be **5001 to 5030** (refer to Section 15.2.2).

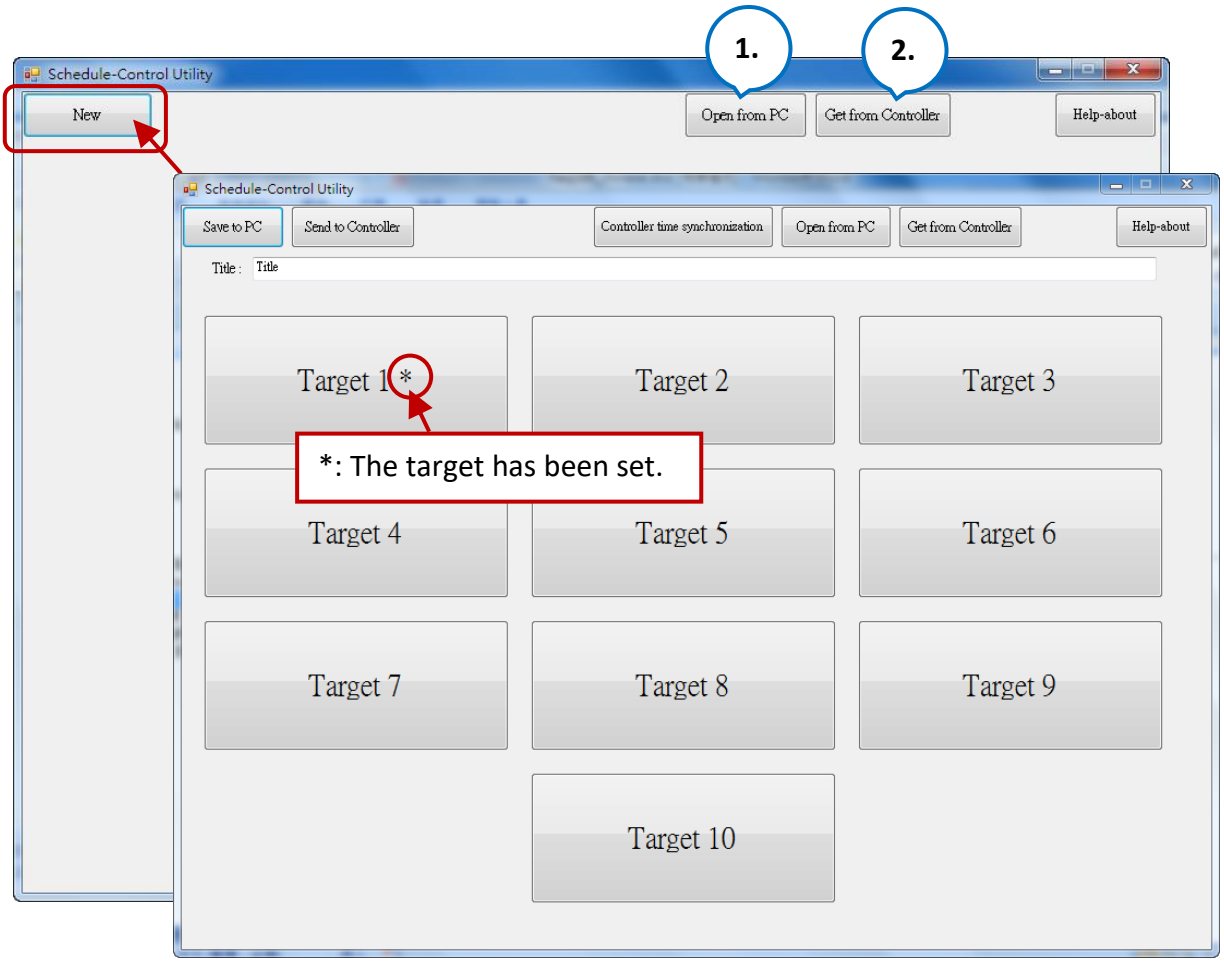
Address	Type	Description	Address	Type	Description
5001	BOOL	BOOL, DINT, and REAL variable controlled by **Target 1**	5016	BOOL	BOOL, DINT, and REAL variable controlled by **Target 6**
5002	DINT		5017	DINT	
5003	REAL		5018	REAL	
5004	BOOL	BOOL, DINT, and REAL variable controlled by **Target 2**	5019	BOOL	BOOL, DINT, and REAL variable controlled by **Target 7**
5005	DINT		5020	DINT	
5006	REAL		5021	REAL	
5007	BOOL	BOOL, DINT, and REAL variable controlled by **Target 3**	5022	BOOL	BOOL, DINT, and REAL variable controlled by **Target 8**
5008	DINT		5023	DINT	
5009	REAL		5024	REAL	
5010	BOOL	BOOL, DINT, and REAL variable controlled by **Target 4**	5025	BOOL	BOOL, DINT, and REAL variable controlled by **Target 9**
5011	DINT		5026	DINT	
5012	REAL		5027	REAL	
5013	BOOL	BOOL, DINT, and REAL variable controlled by **Target 5**	5028	BOOL	BOOL, DINT, and REAL variable controlled by **Target 10**
5014	DINT		5029	DINT	
5015	REAL		5030	REAL	

15.5.2 Configure the Target



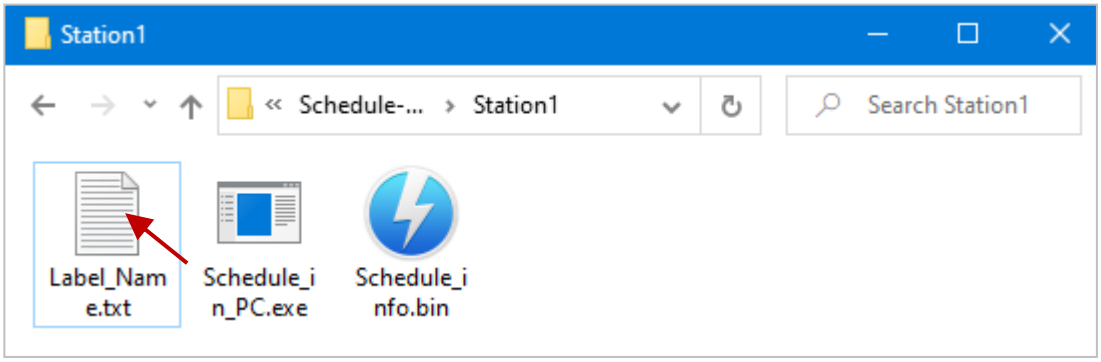
Execute the Schedule-Control Utility and click “New” to display the settings screen. Each Win-GRAF PAC can control a maximum of 10 Targets and the default name is Target 1 to Target 10.

1) **Open from PC**: Open an existing configuration file.
2) **Get from Controller:** Load the current settings from the PAC.



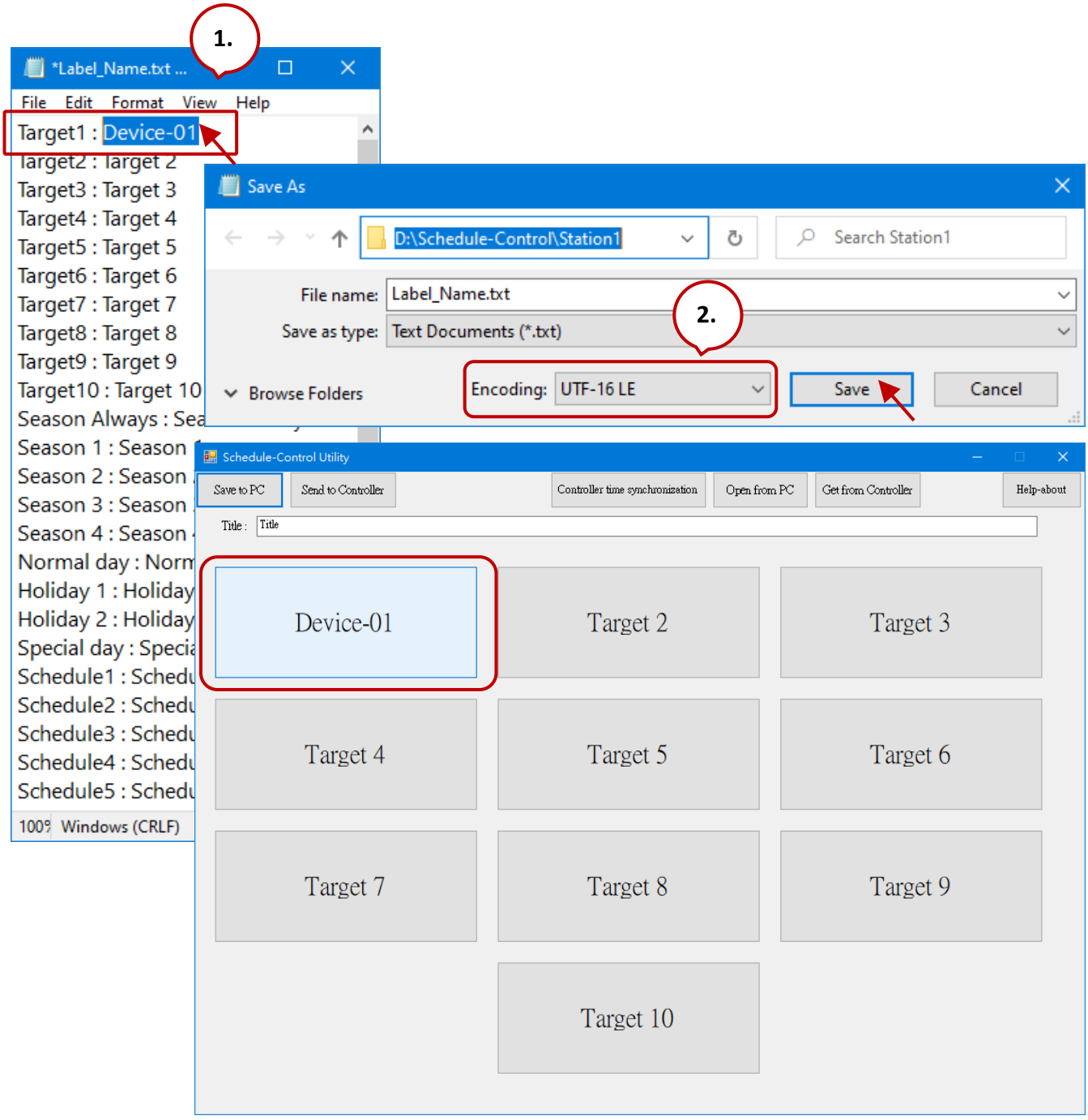
<u>Change the Target Name</u>:

There is a “**Label_Name.txt**” file in the project folder (e.g., D:\Schedule-Control\Station1\). The name of Target, Season, and so on in the file can be changed according to the needs of the application. The default name will be displayed if the file does not exist.



Edit the "Label_Name.txt" File:

1. Enter the desired name after the colon (": ") symbol. E.g., Device-01 and the prefix/suffix of spaces will be erased.
2. After completing the edit, saving the file in a Unicode format.
3. Both the "Label_Name.txt" and "Schedule_in_PAC.exe" must be stored in the same folder.

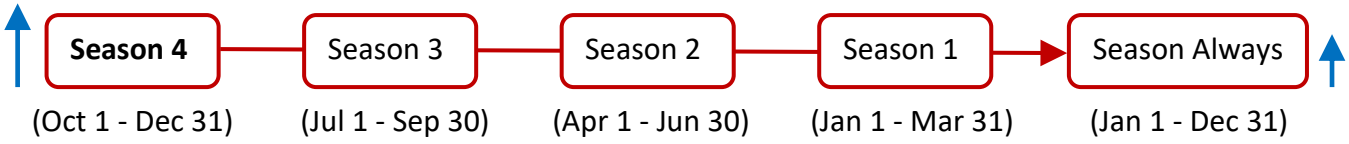


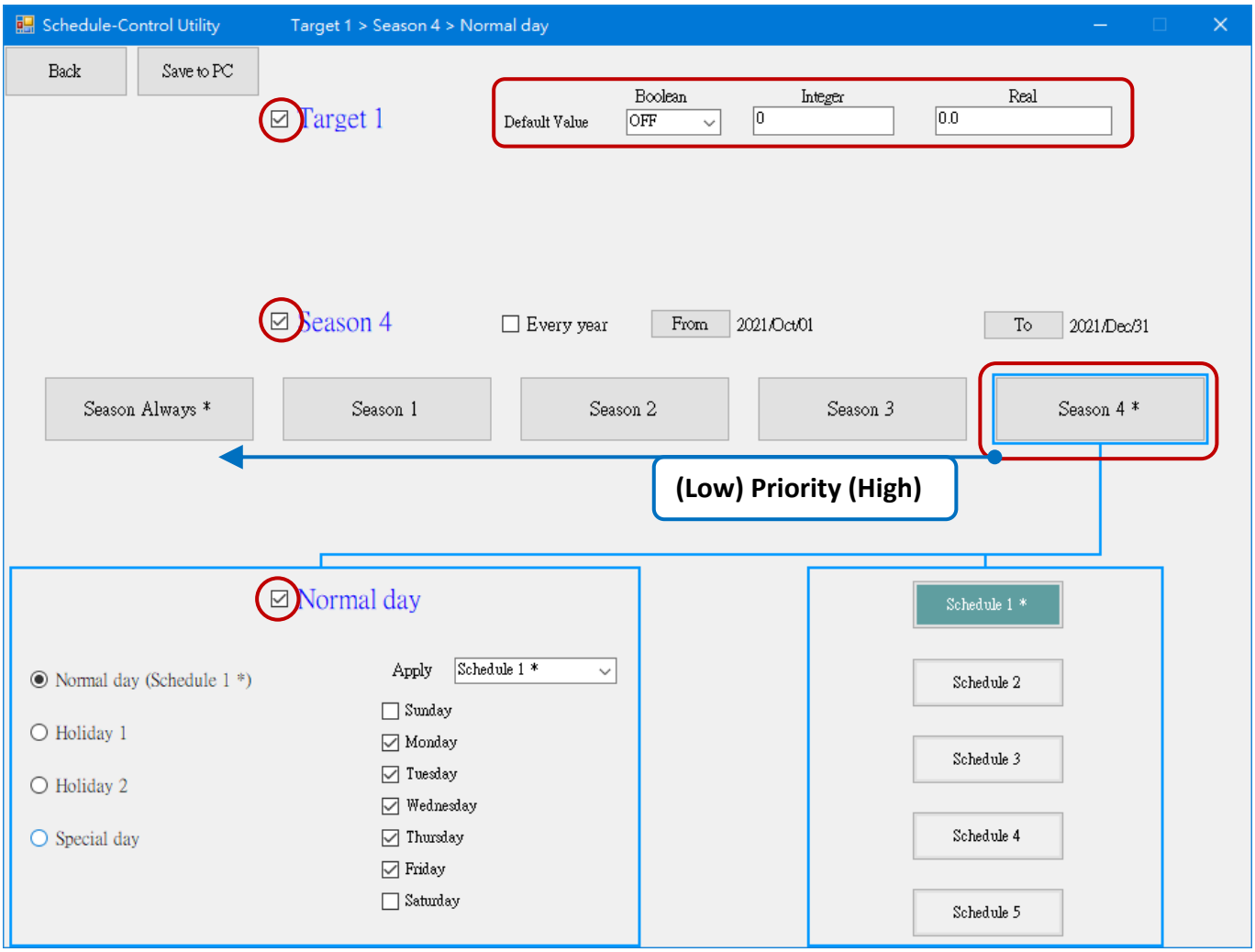
15.5.3 Configure the Season

Each Target provides the Season Always, Season 1, Season 2, Season 3, and Season 4 settings that can be used to configure all year-round or four seasons schedules. Note that check the checkbox before setting.

The Searching Priority of Seasons:

It is recommended to enable the **Season Always** setting. If multiple Season settings are enabled, the PAC will apply the Boolean, Integer, and Real data in the following orders. The default setting (OFF, 0, 0.0) will be applied if no available setting is found.





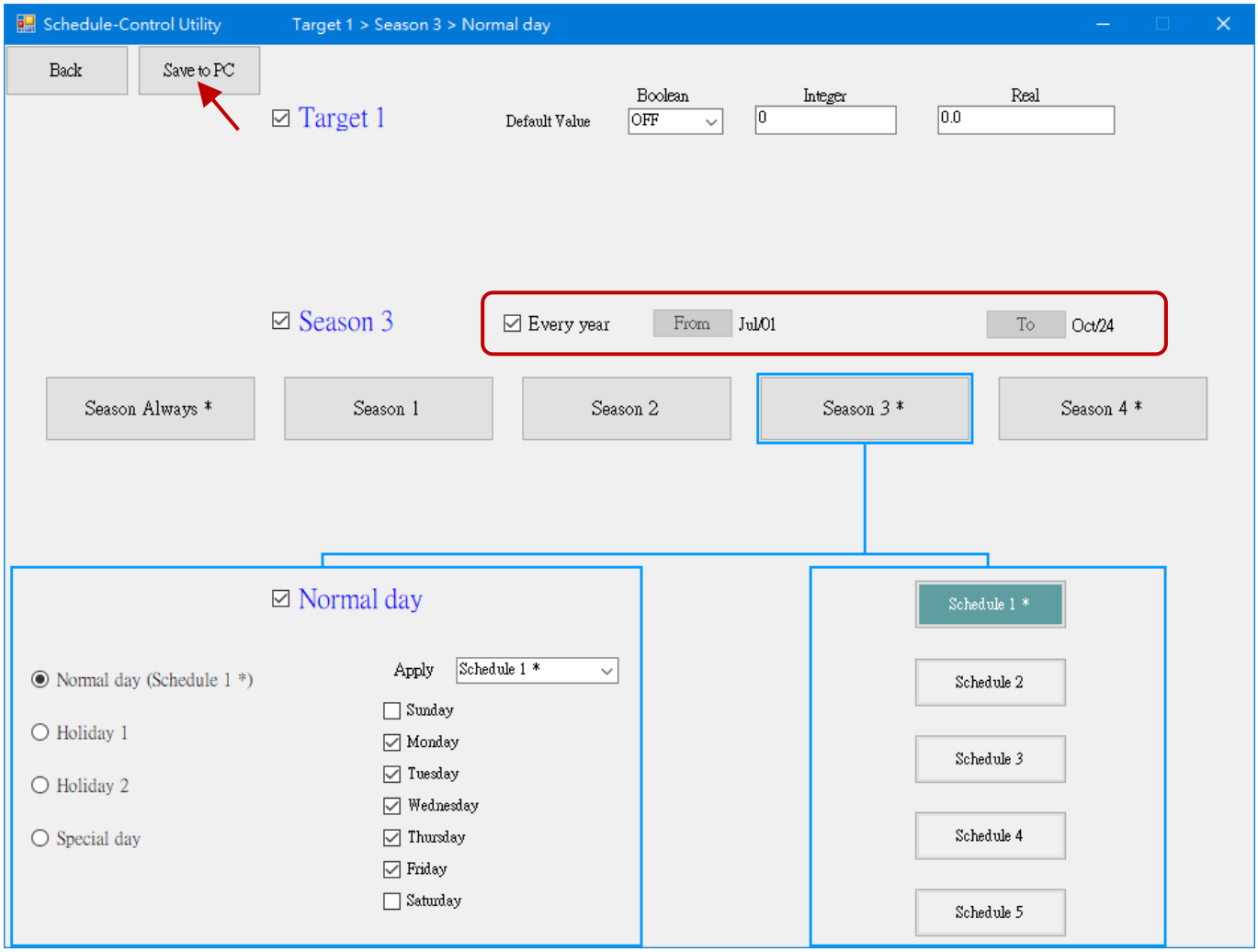
The Season Setting:

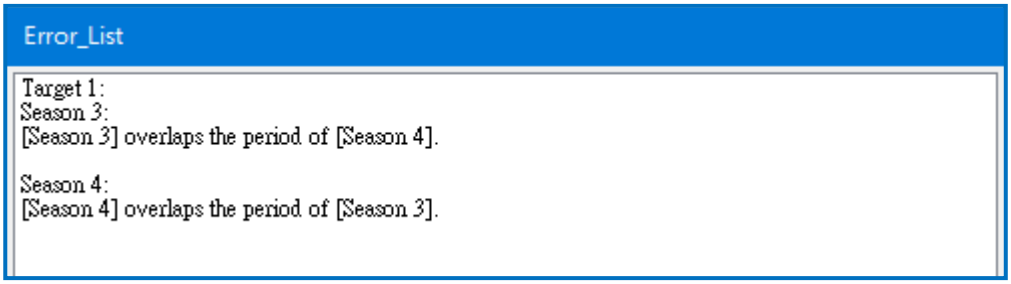
To enable Season (1 to 4), the date range must be set. Also, check the “Every year” box to apply the setting every year.

Notice:

1. The specified date range of Season 1 to 4 cannot overlap and the start date (From) must earlier than the end date (To).
2. Uncheck the “Every year” box before changing the date.

For example,
The date range of Season 4 is **Oct 01** - Dec 31. If the date range of Season 3 is set to Jul 01 - **Oct 24**, an error message will be displayed when clicking the **Save to PC** button to save the settings.



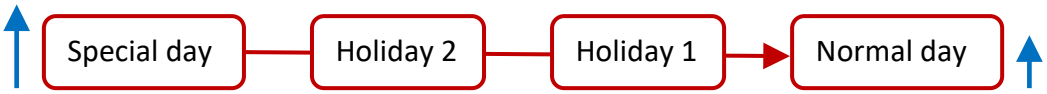


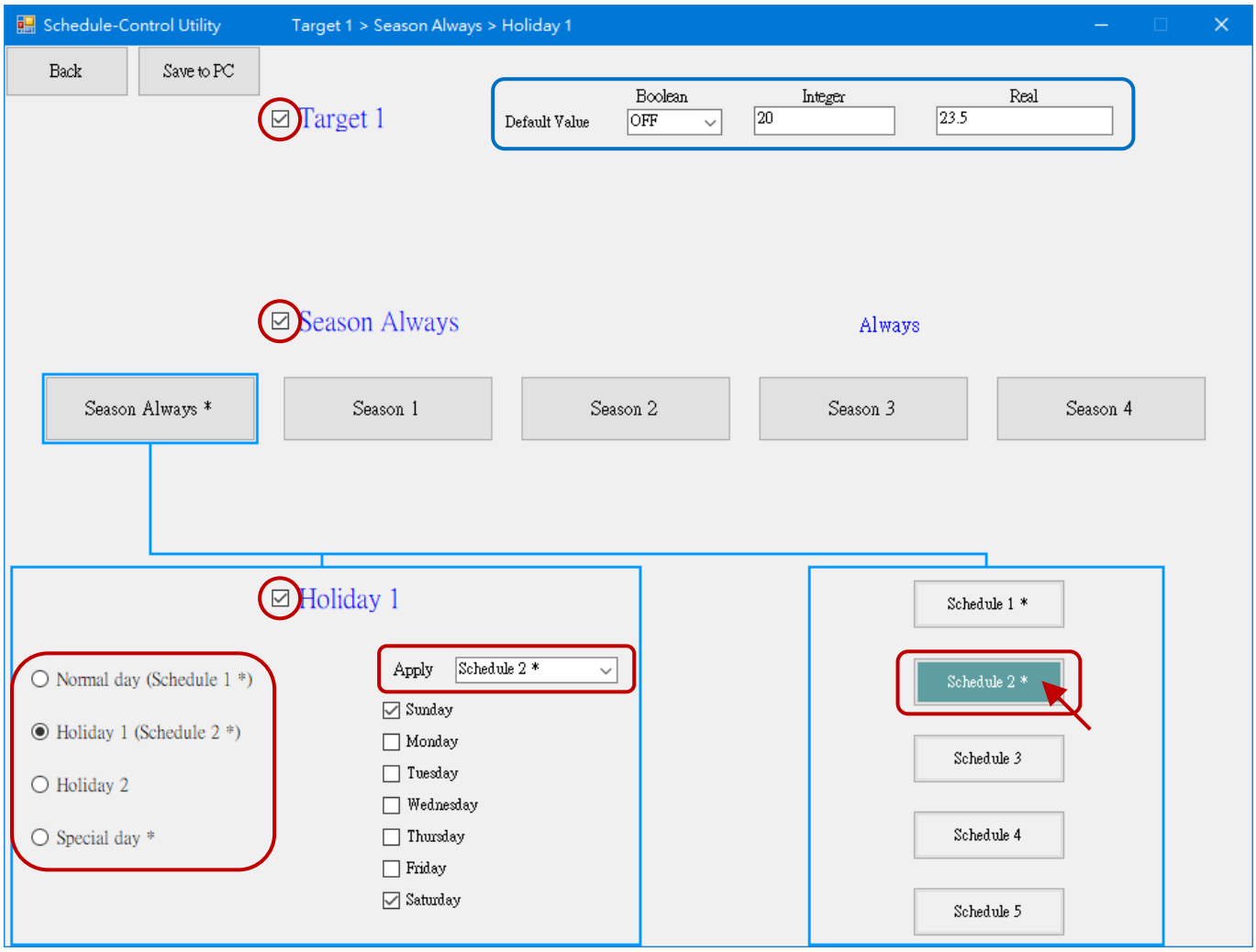
15.5.4 Configure the Normal Day / Holiday / Special Day

Each Season provides the Normal day, Holiday 1, Holiday 2, and Special day setting that can be used to configure the schedule for a weekday, holiday, or specific day. Note that check the checkbox before setting.

The Searching Priority of Days:

If multiple Day settings are enabled, the PAC will apply the Boolean, Integer, and Real data in the following orders. The default setting (OFF, 0, 0.0) will be applied if no available setting is found.

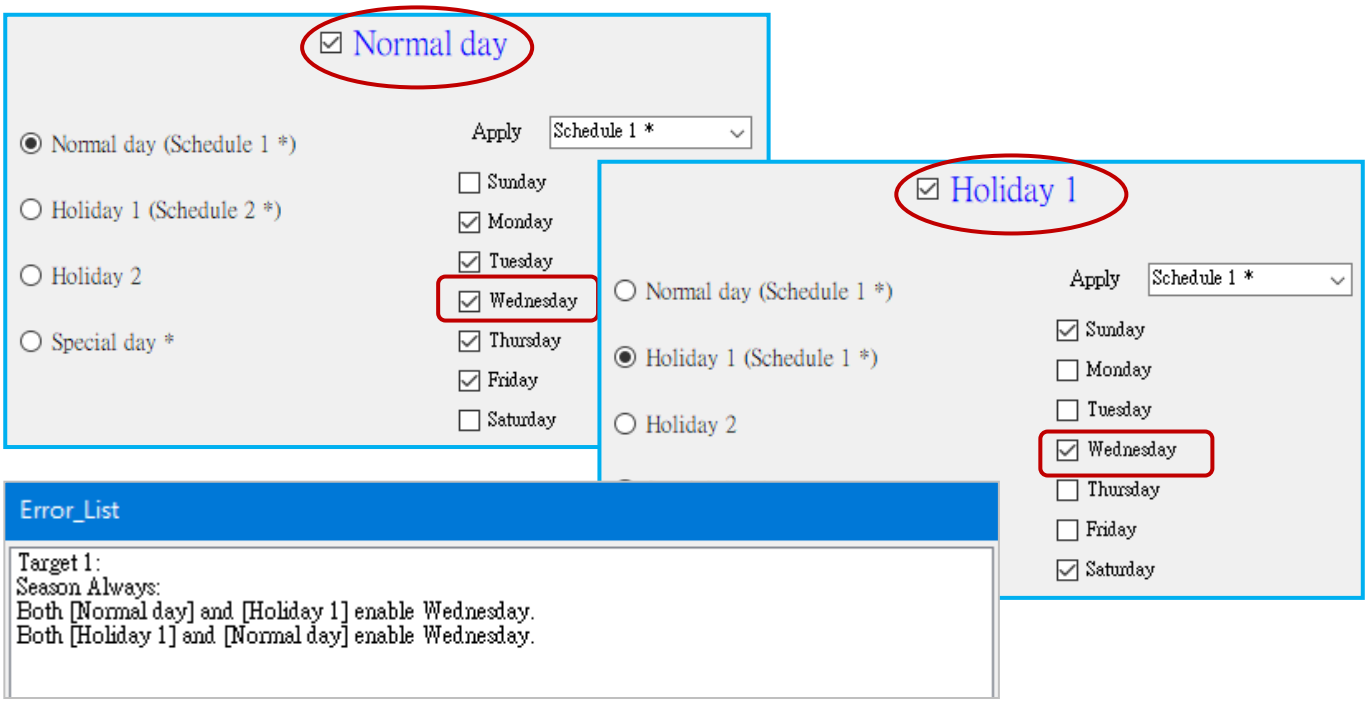




Normal day	Working days, usually Monday to Friday.	If the Day setting is enabled, at least one Schedule (1-5) must be set.
Holiday 1	Normally set to Saturday and Sunday.	
Holiday 2	It can be set from Monday to Sunday.	
Special Day	Used to specify a date, e.g., Oct 10, Jul 4, etc. Up to 50 days can be set for each Season.	

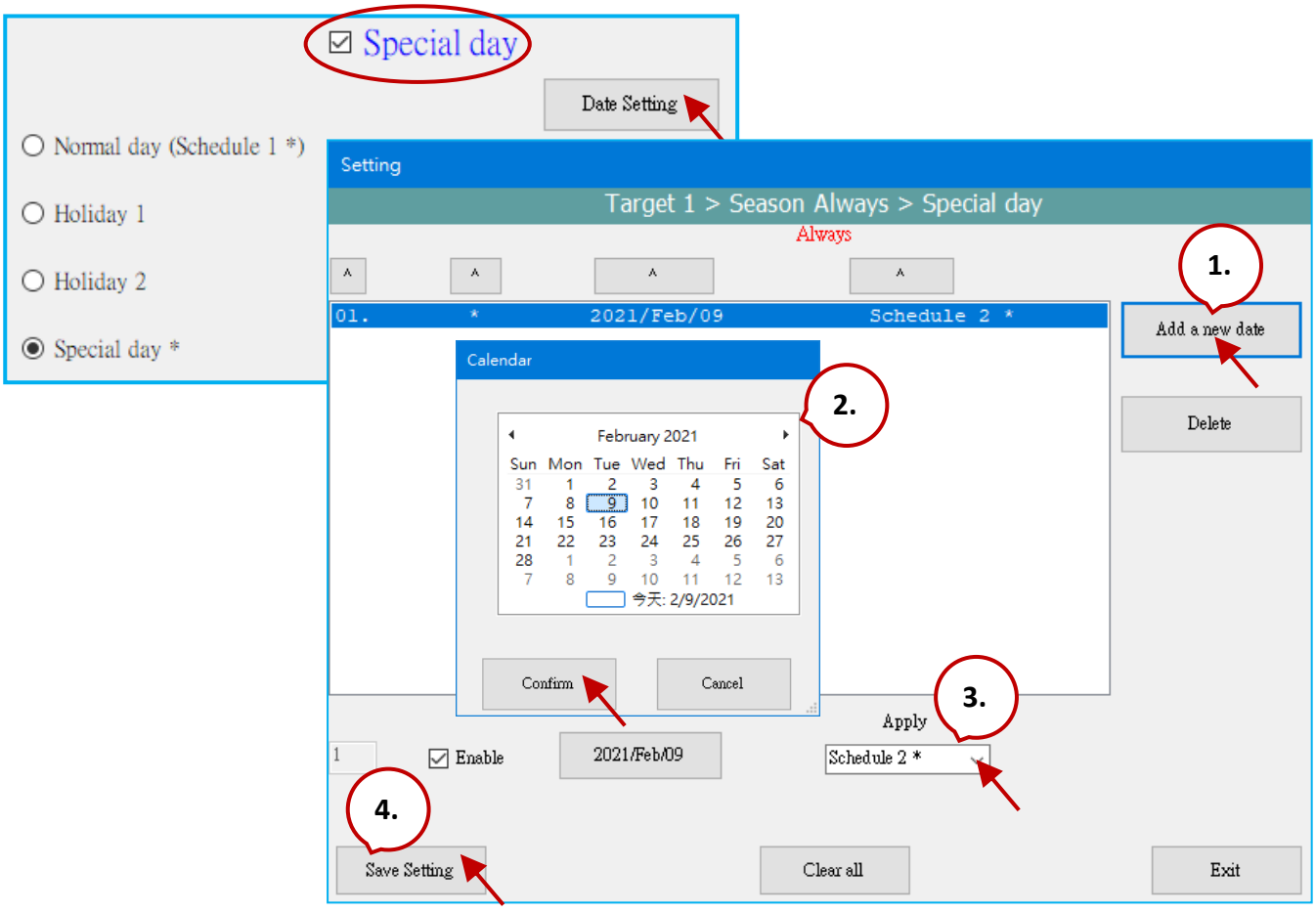
Notice:

The specified day of the Normal day, Holiday 1, and Holiday 2 cannot overlap. E.g., Both the Wednesday of the Normal day and Holiday 1 are checked, an error message will be displayed when saving the settings.



Specify the date of the Special Day:

"Special Day" can be used to specify a maximum of 50 dates for each "Season" setting.

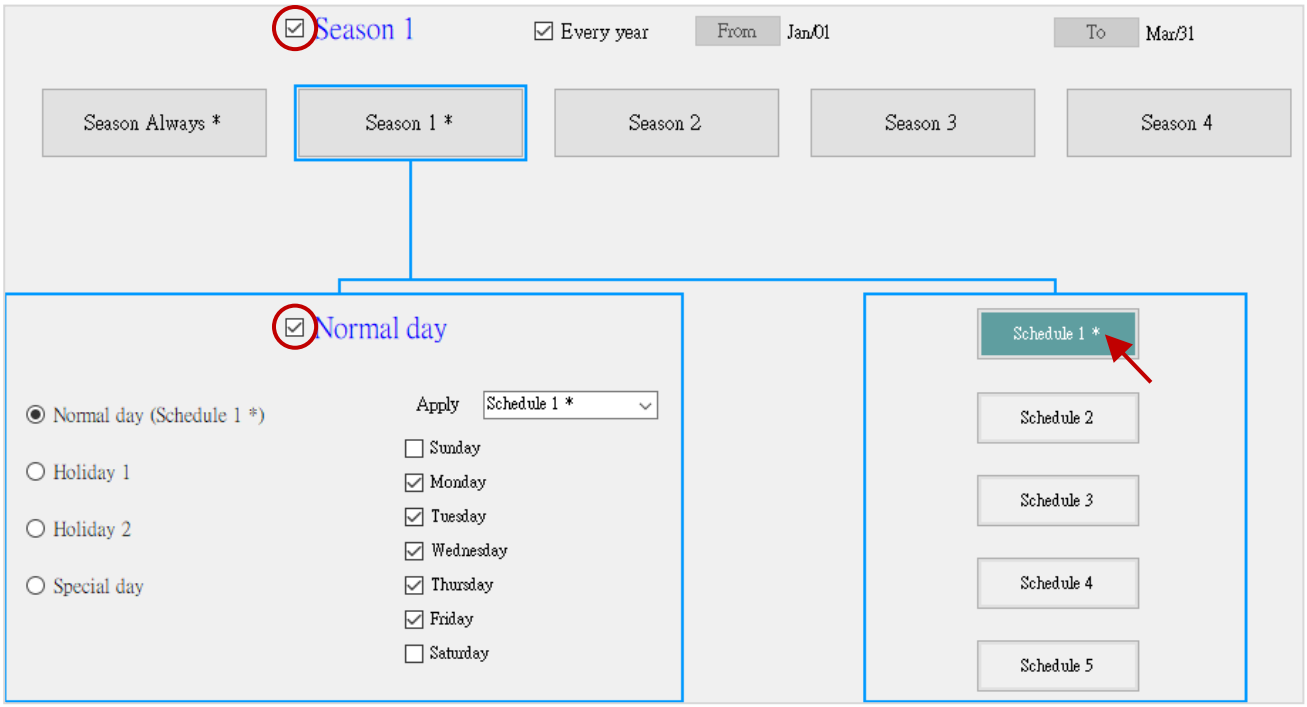


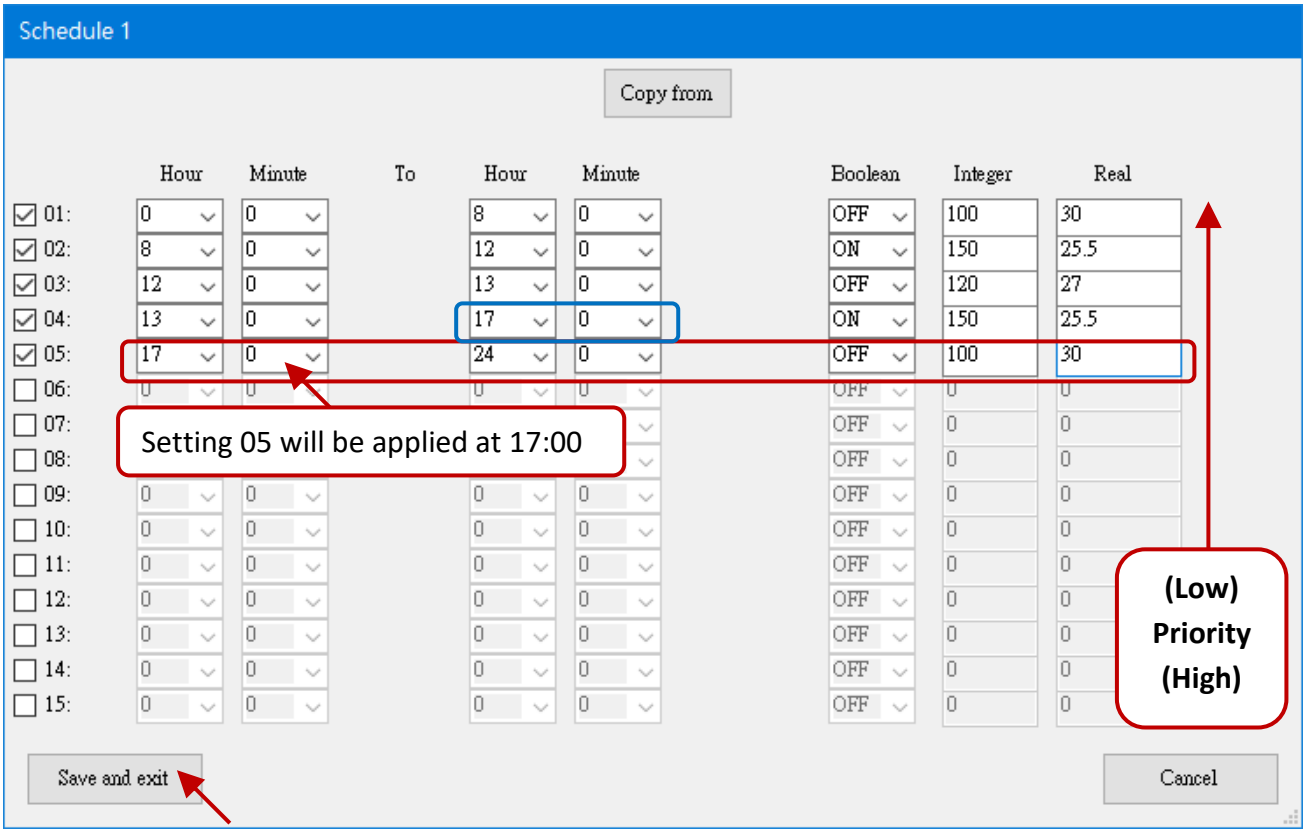
15.5.5 Configure the Schedule Time

Up to five Schedules (1 to 5) can be set in each Season setting and up to fifty periods of time can be set in each Schedule setting. The minimum unit of time is minutes, which must be within the range of 00:00 to 24:00. Note that check the checkbox before setting.

The Searching Priority of Schedule Time:

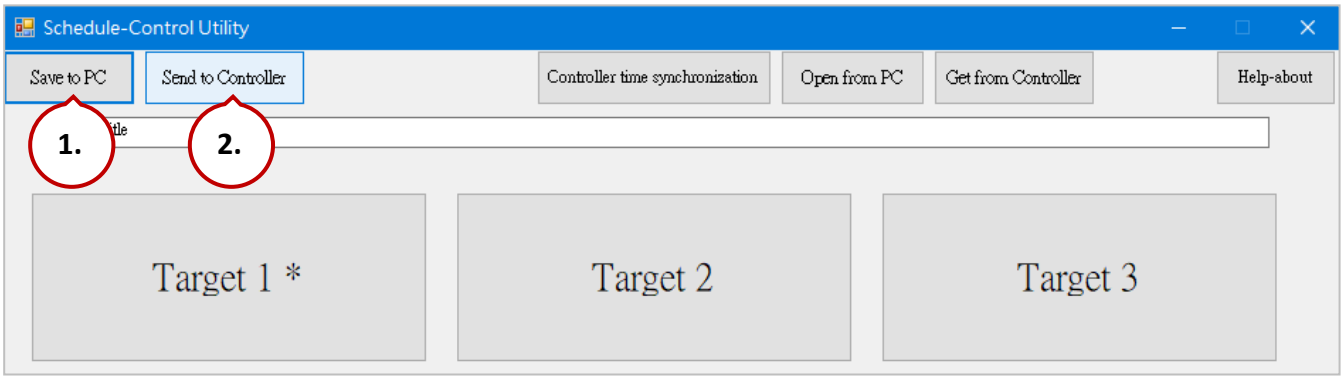
If multiple schedule times are enabled, the PAC will apply the Boolean, Integer, and Real values according to the settings with the larger number. The default values (OFF, 0, 0.0) will be applied if no available setting is found.





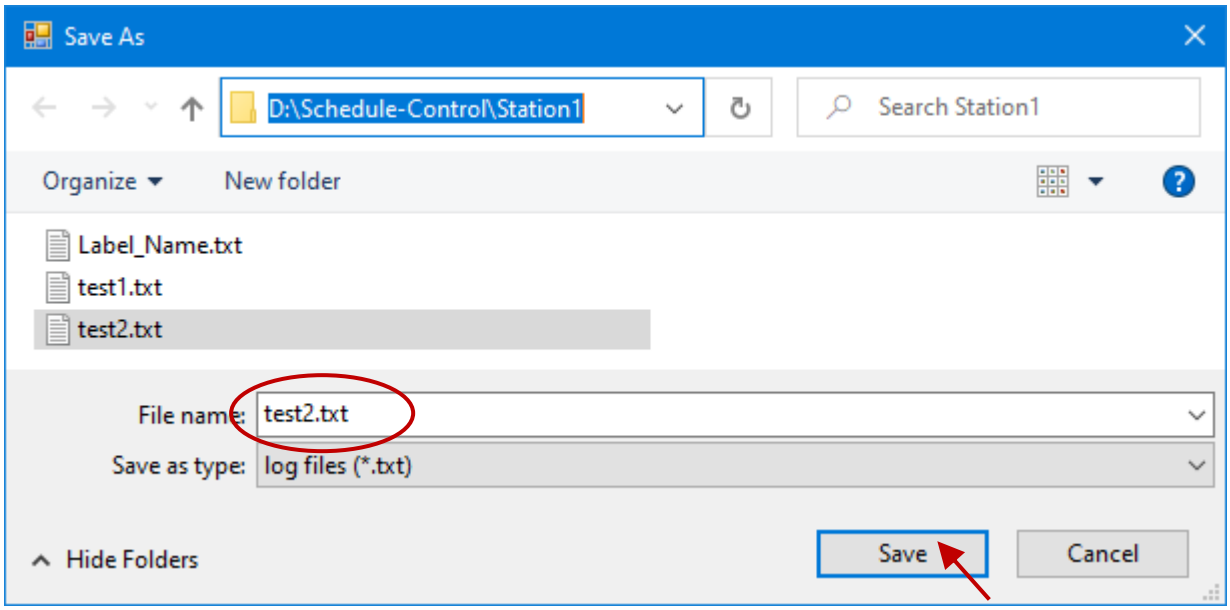
15.5.6 Save and Send the Configuration File

After completing the setting, save it as a file on PC and then send configurations to the PAC.



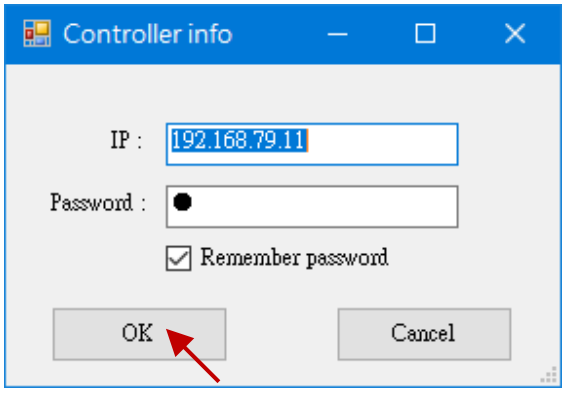
1. Save the file on PC

   Click the **Save to PC** button to save the configuration file (*.txt).



2. Send the Configuration to the PAC

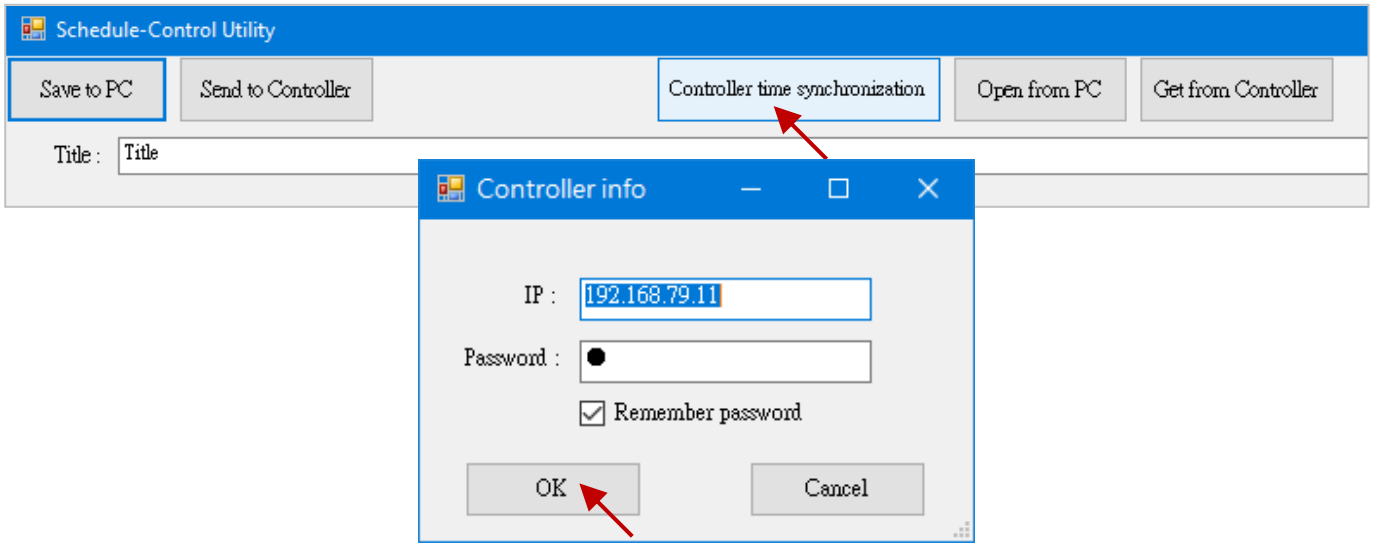
   Click the **Send to Controller** button to send configurations to the PAC. Enter the IP address of the PAC and the password (defaults: 0).



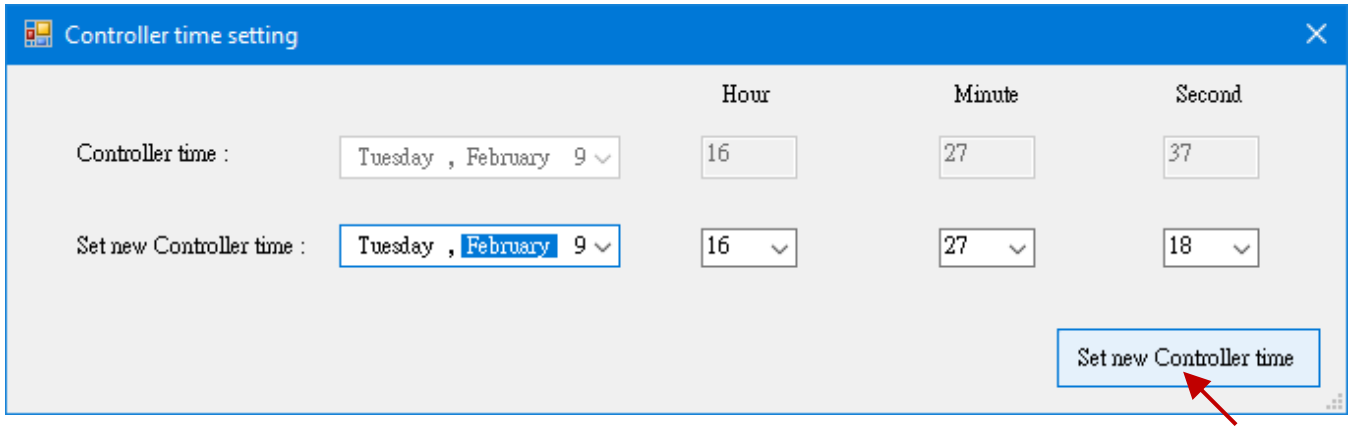
15.5.7 Time Synchronization

If the system time of the PAC is different from the current time, for conducting the scheduling on time, run the time calibration in the Schedule-Control Utility to correct the PAC time. Make sure that both the PC and Win-GRAF PAC are online.

1. Click the **Controller Time Synchronization** button and enter the IP address of the PAC and the password (defaults: 0). Then, click OK.



2. Confirm the date, hour, minute, and second, then click the **Set new Controller time** button to do the time synchronization.

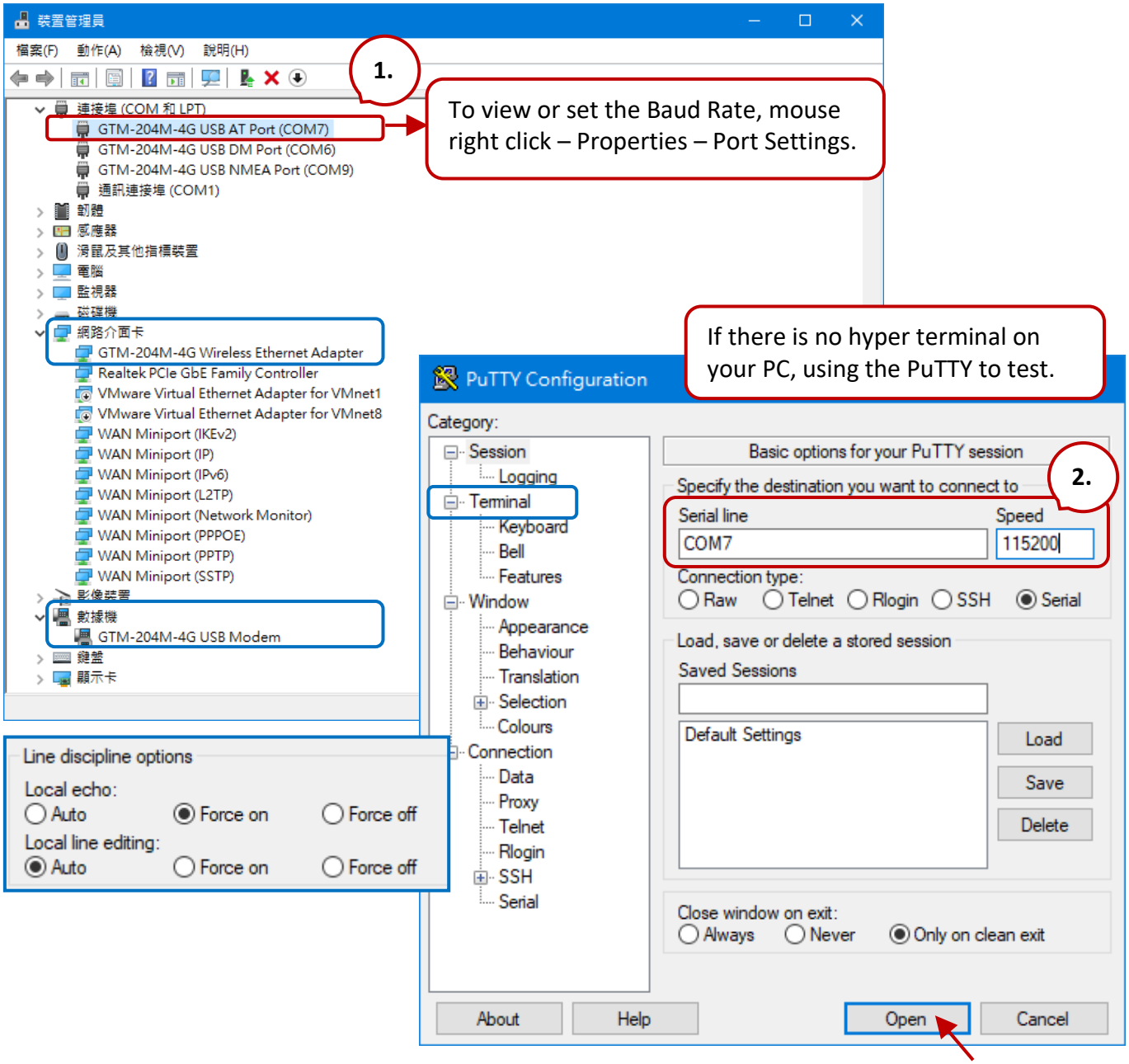


Chapter 16 Win-GRAF SMS Function

This chapter shows the way to send/receive an SMS message by using the RPAC-2568M that comes with the GTM-204M-4GE 3G/4G wireless module.

Note:

1. Due to the product certification issue, these modules can sale in certain areas. Please contact our agents for more information.
2. Visit the download page of the GTM-204M-4GE and refer to the quick start guide to install the USB driver and test the module.

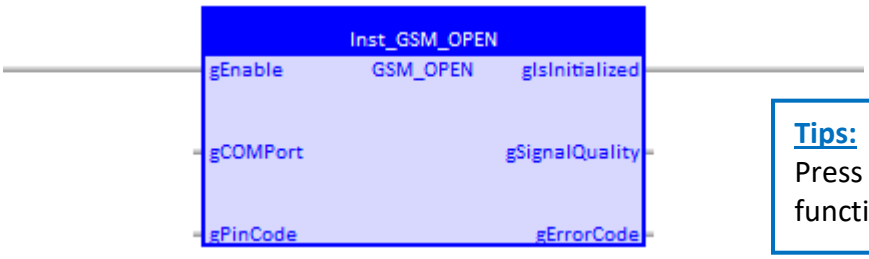


16.1 "GSM_Open", "Send_SMS" and "Read_SMS" Functions

The following three functions can be used to send or receive an SMS message in the Win-GRAF PAC.

The "GSM_OPEN" function:

To open/close the GSM module.



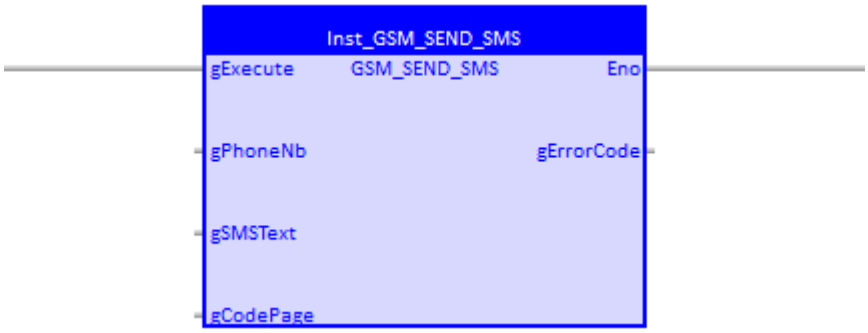
Tips:
Press the "F1" key on the function block for Help.

Inputs	Data Type	Description
gEnable	BOOL	TRUE: Open the specified COM port to connect the GSM module and initialize it. FALSE: Disconnect the GSM module and close the COM Port.
gCOMPort	DINT	The COM port number that connected to the GSM module.
gPinCode	STRING	Enter the PIN code if it is set in a SIM card. (E.g., '0000').

Outputs	Data Type	Description
gIsInitialized	BOOL	TRUE: The COM Port has been opened and the GSM module has been initialized. FALSE: Failed to open the COM Port or initialize the GSM module.
gSignalQuality	SINT	0 ~ 31: The higher the value, the better the signal quality. 99: The signal is unknown or not detected.
gErrorCode	INT	0 : No error. -1: The GSM module is not connected. -2: The SIM card is not inserted properly. -3: The PIN code of the SIM card is wrong. -4: SIM configuration error. -5: Cannot open the specified COM port.

The "GSM_SEND_SMS" Function:

To send the SMS message via the GSM module.



Notice:

Before using this "GSM_SEND_SMS" function, using the "GSM_OPEN" function block to open the COM Port that is connected to the GSM module, or this function will not work.

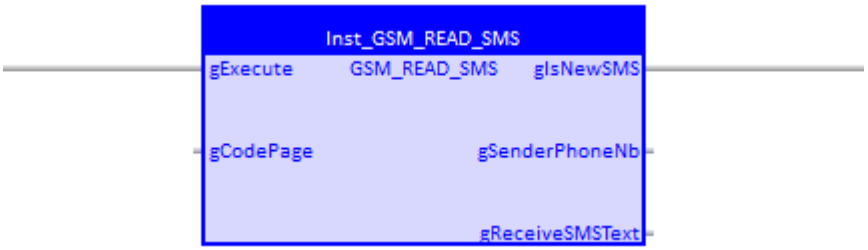
Inputs	Data Type	Description
gExecute	BOOL	Pulse TRUE: to send an SMS message.
gPhoneNb	STRING	Destination-Address (i.e., the phone number).
gSMSText	STRING	The SMS message.
gCodePage	UDINT	The code page of the text. 0: English 866: Russian 932: Japanese 936: Simplified Chinese 950: Traditional Chinese **Notice:** Only the maximum number of characters will be sent if exceeds.

Value	The maximum length of a text message
0	160 characters
non-zero	70 characters

Outputs	Data Type	Description	
gErrorCode	INT	4: Send SMS succeeds 3: Sending SMS 2: Send SMS is pending 1: Prepare to send SMS 0: No operation.	-1: No GSM module detected. -2: SIM card is not inserted. -4: SIM card configuration error. -5: Cannot open the PAC's COM port. -6: No recipient's phone number. -7: Send SMS message failed.

The "GSM_READ_SMS" Function:

To read the SMS message via the GSM module.



Notice:

Before using this "GSM_READ_SMS" function, using the "GSM_OPEN" function block to open the COM Port that is connected to the GSM module, or this function will not work.

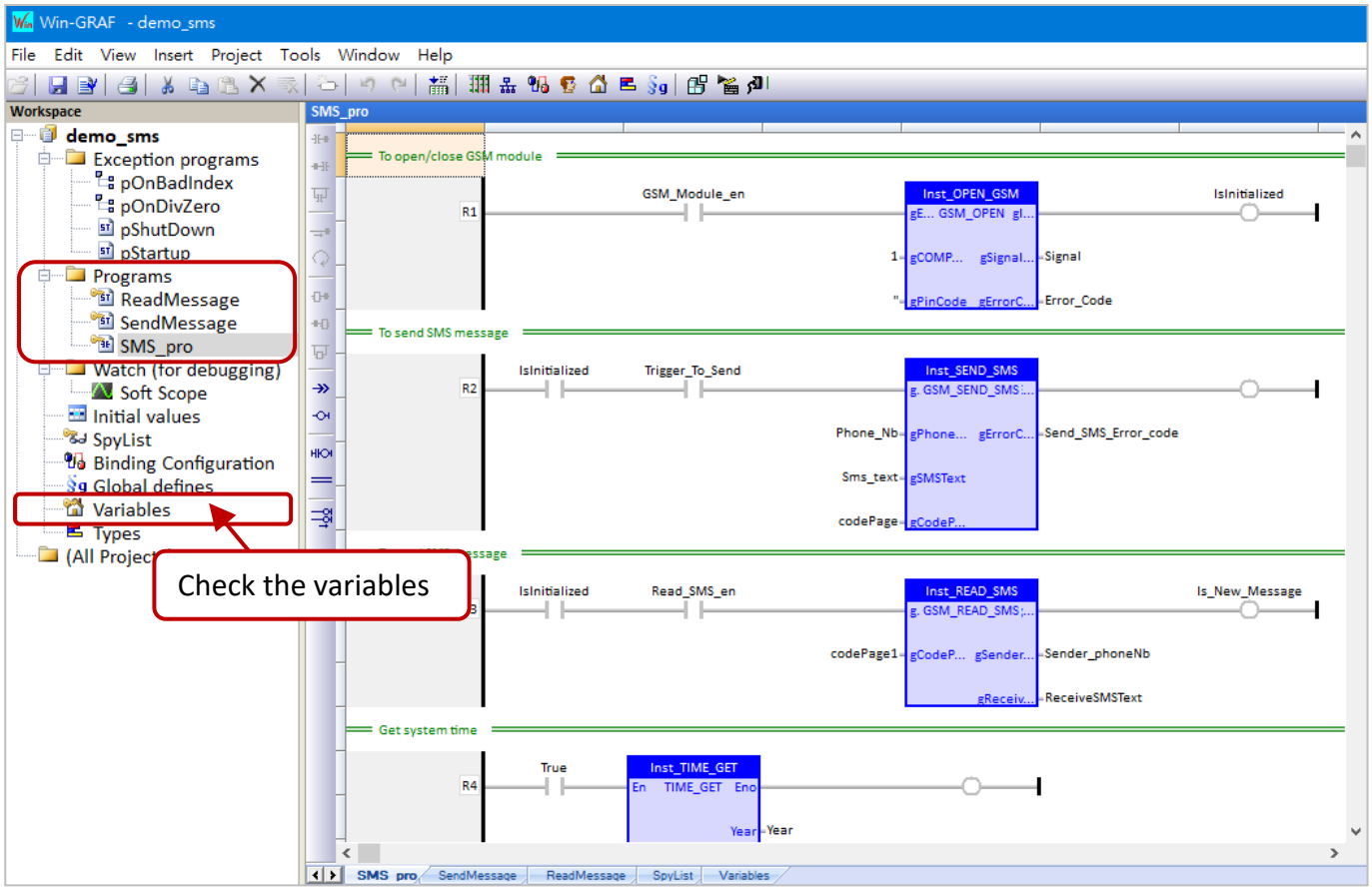
Inputs	Data Type	Description
gExecute	BOOL	TRUE: Enable the GSM module to read an SMS message. FALSE: Disable the GSM module to read an SMS message.
gCodePage	UDINT	The code page of the text. 0: English 866: Russian 932: Japanese 936: Simplified Chinese 950: Traditional Chinese

Outputs	Data Type	Description
gIsNewSMS	BOOL	Pulse TRUE: A new message is coming.
gSenderPhoneNb	STRING	Originating-Address (i.e., the phone number).
gReceiveSMSText	STRING	The SMS message.

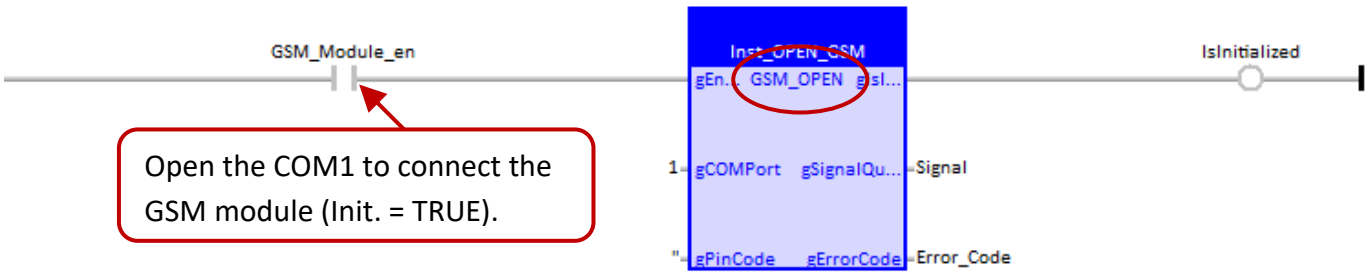
16.2 Introduction of the "Demo_SMS" Project

Download the demo program on the website (**demo_sms.zip**). Click the menu commands **File - Add Existing Project - From Zip...** in the Workbench to open the project. (Also refer to Section 11.4)

The project includes two ST programs (SendMessage and ReadMessage) and an LD program (SMS_pro).

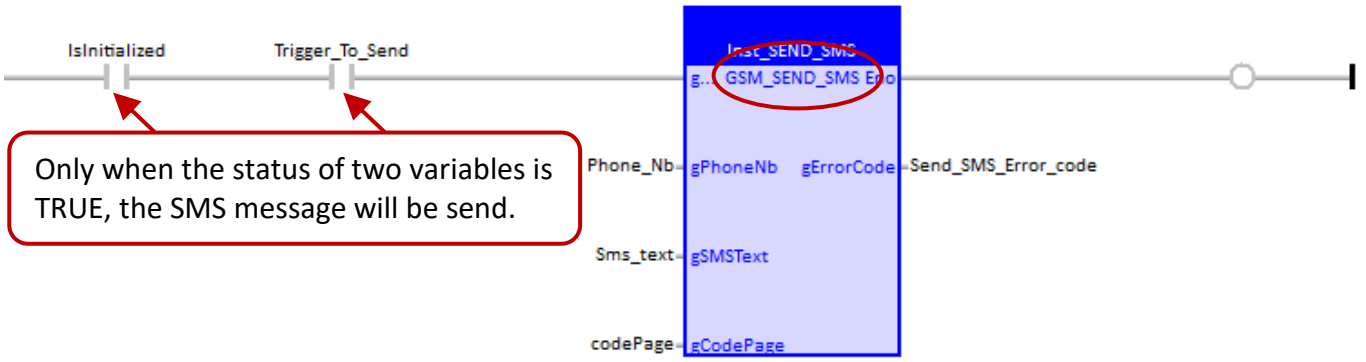


LD Program (SMS_pro): The "GSM_OPEN" function (refer to Section 16.1 for more details)



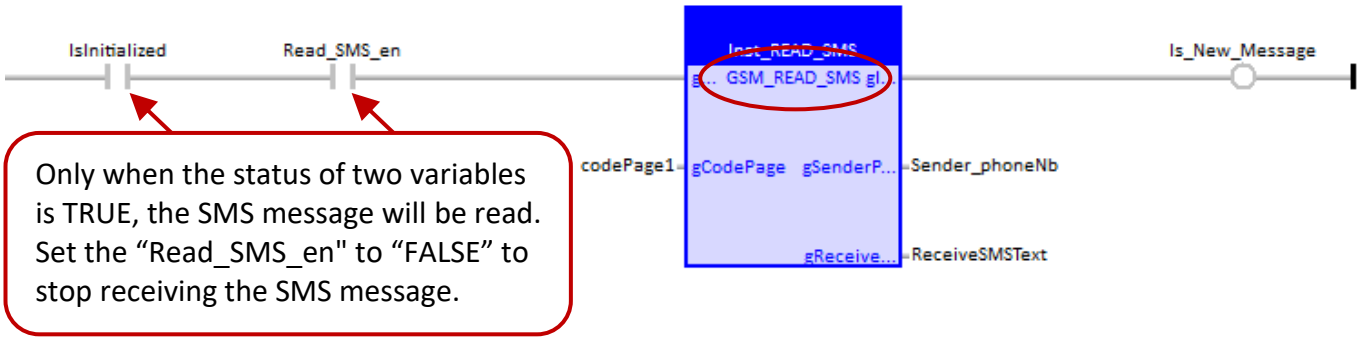
Name	Data Type	Description
GSM_Module_en	BOOL	Set it to TRUE to open the specified COM port to connect and initialize the GSM module (Init. = TRUE).
Signal	SINT	The signal quality of the GSM module.
Error_Code	INT	The error code of the GSM module.
IsInitialized	BOOL	The initialization results of the GSM module.

The "GSM_SEAD_SMS" function (refer to Section 16.1 for more details)



Name	Data Type	Description
Trigger_To_Send	BOOL	Set it to “TRUE” to send an SMS message
Phone_Nb	STRING(255)	The phone number of the recipient. (Init. value: '0900629879')
Sms_text	STRING(255)	The content of the SMS message. (Init. value: 'This message is sent from Win-GRAF PAC')
codePage	UDINT	The code page of the text.
Send_SMS_Error_code	INT	The error code of the SMS message.

The "GSM_READ_SMS" function (refer to Section 16.1 for more details)



Name	Data Type	Description
Read_SMS_en	BOOL	Sec it to “TRUE” to read an SMS message. (Init value: TRUE)
codePage1	UDINT	The code page of the text. (Init value: UDINT#950)
Sender_phoneNb	STRING(255)	The sender’s phone number.
ReceiveSMSText	STRING(255)	The content of the SMS message.
Is_New_Message	BOOL	To check if there is any new SMS message.

The ST Program (SendMessage)

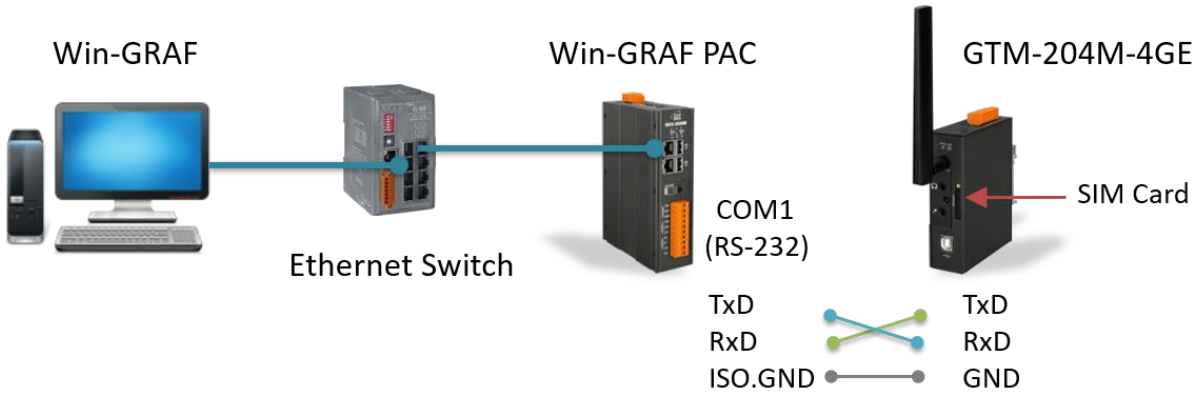
```
if Trigger_To_Send then

  if IsInitialized then
    if Send_SMS_Error_code < 0 then
      (* Send SMS failed *)
      Trigger_To_Send := false;

      (* TODO: Add failed processing here *)

    elsif Send_SMS_Error_code = 4 then
      (* Send SMS successed *)
      Trigger_To_Send := false;

      (* TODO: Add success processing here *)

    end_if;
  else
    (* GSM module is not initialized *)
    (* TODO: Add failed processing here *)

  end_if;
end_if;
```

The ST Program (ReadMessage)

```
(* Get new message *)
if Is_New_Message then

  (* If the user need more time to deal with the new message, *)
  (* the user could set "Read_SMS_en" as “FALSE” to disable the Read_SMS function *)
  (* and then set it as “TRUE” after dealing *)
  (* Notice : If the user set "Read_SMS_en" as “FALSE”, it will stop reading the SMS message. *)
  (* Read_SMS_en := false; *)

  Got_New_Message := ReceiveSMSText;
  Got_Message_from_who := Sender_phoneNb;

  (* do more operating *)

end_if;
```

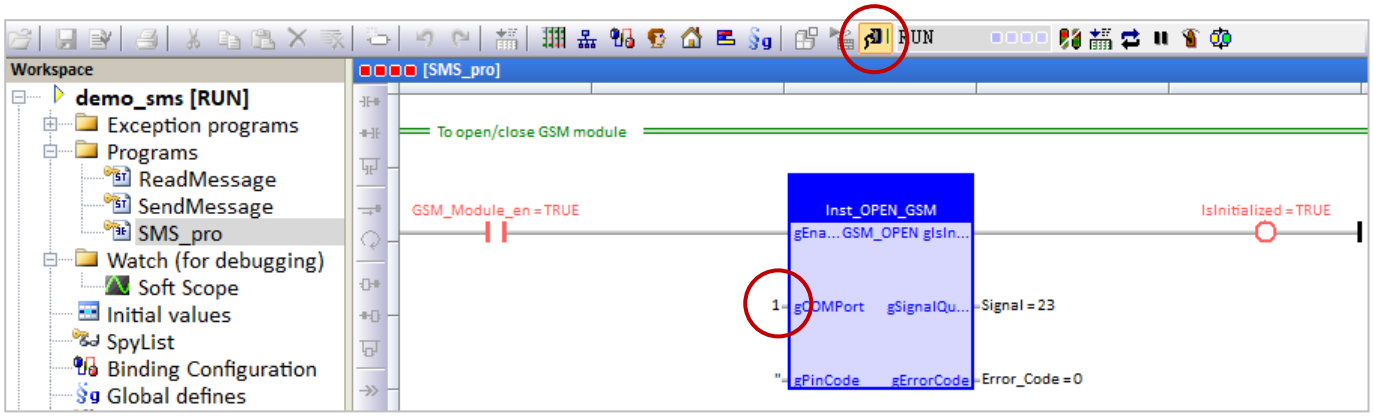
16.3 Test for SMS Messaging

In this example, the SMS message is sent or received by using the RPAC-2568M PAC and the GTM-204M-4GE module. First, insert the SIM card into the module and connect to the PAC via the RS-232 port, and then power on.



Download the project

1. Modify the Win-GRAF communication IP to the current IP address of the PAC.
2. Enter the number of PAC's COM port that is connected to the GSM module.
3. Click the **On Line** button to download the "dmeo_sms" project to the Win-GRAF PAC. (Also refer to Section 2.3.4 if you are not familiar with the setting.)



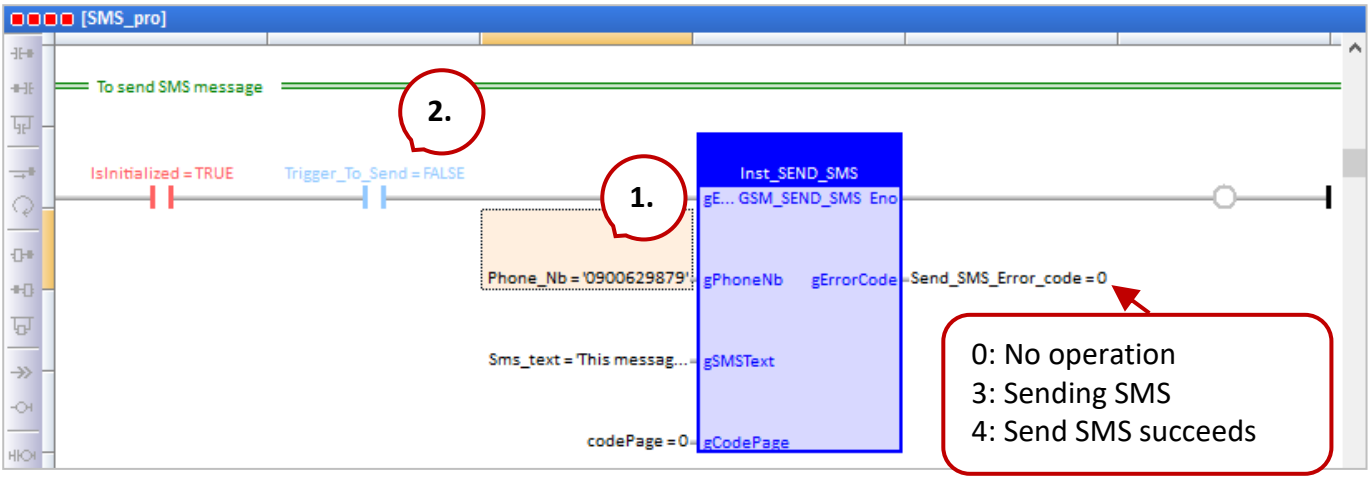
Name	Description
GSM_Module_en	Open the COM1 to connect and initialize the GSM module. (Init. = TRUE).
Signal	The signal quality of the GSM module is "23".
Error_Code	The error code of the GSM module is "0" (no error).
IsInitialized	Successful to open COM1 and initialize the GSM module. Note that if the initialization is failed, reboot the PAC.

Send an SMS Message

1. Double-click on the parameter to enter the phone number, the text, or the code page. For example,

Name	Description
Phone_Nb	The initial value is set to '0900629879' (or '886900629879').
Sms_text	The initial value is set to 'This message is sent from Win-GRAF PAC'.
codePage	0: to send an SMS message in English.

2. Double-click the "Trigger_To_Send" to set it to “TRUE” to send an SMS message.

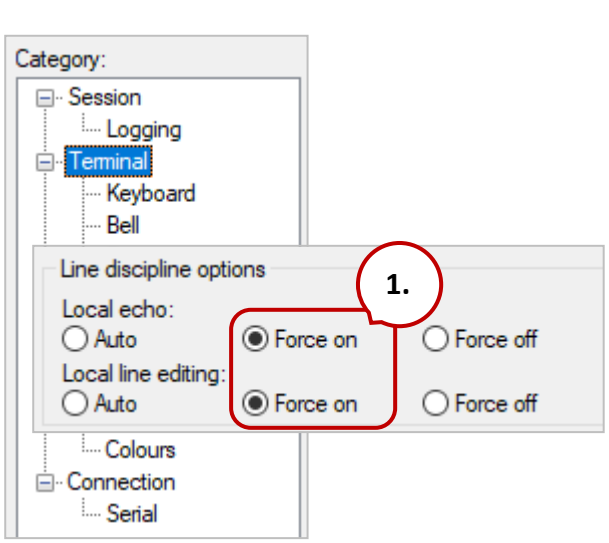


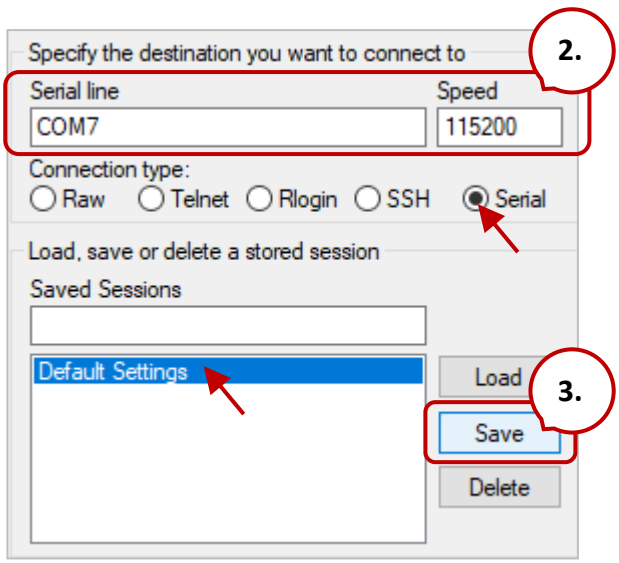
Receive an SMS Message

By default, the GTM-204M series send an SMS notification by using the USB port, and you can use the command **AT+QURCCFG?** to check the current setting.

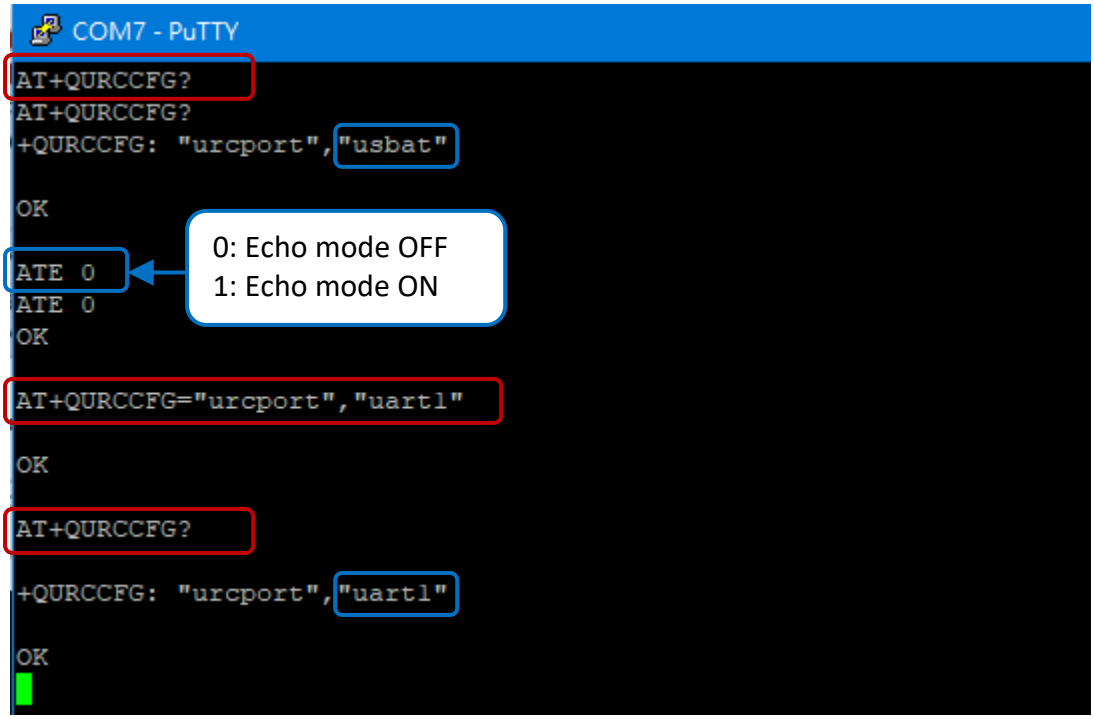
- Using an USB port to receive an SMS message: AT+QURCCFG="urcport","usbat"
- Using an UART port to receive an SMS message: **AT+QURCCFG="urcport","uart1"**

To test with PuTTY (also refer to Chapter 16), set both the **Local echo** and **Local line editing** in **Terminal** to **Force on**, fill in the port settings in **Session**, and click the **Save** button.

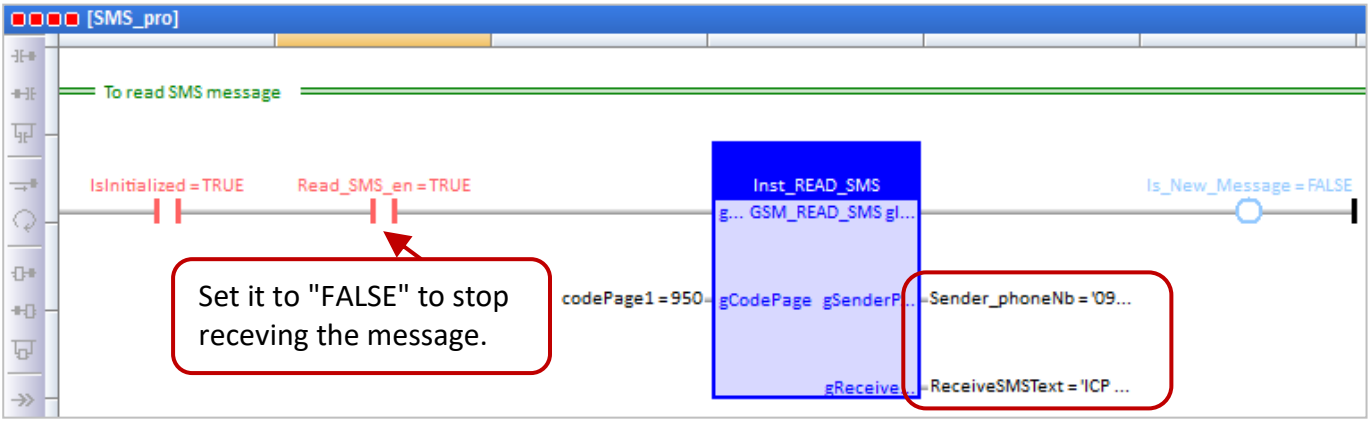




1. Enter the command **AT+QURCCFG?** to check the current setting.
2. Enter the command **AT+QURCCFG="urcport","uart1"** to change to use the UART port to receive SMS notifications.
3. Enter the command **AT+QURCCFG?** to check the setting again.

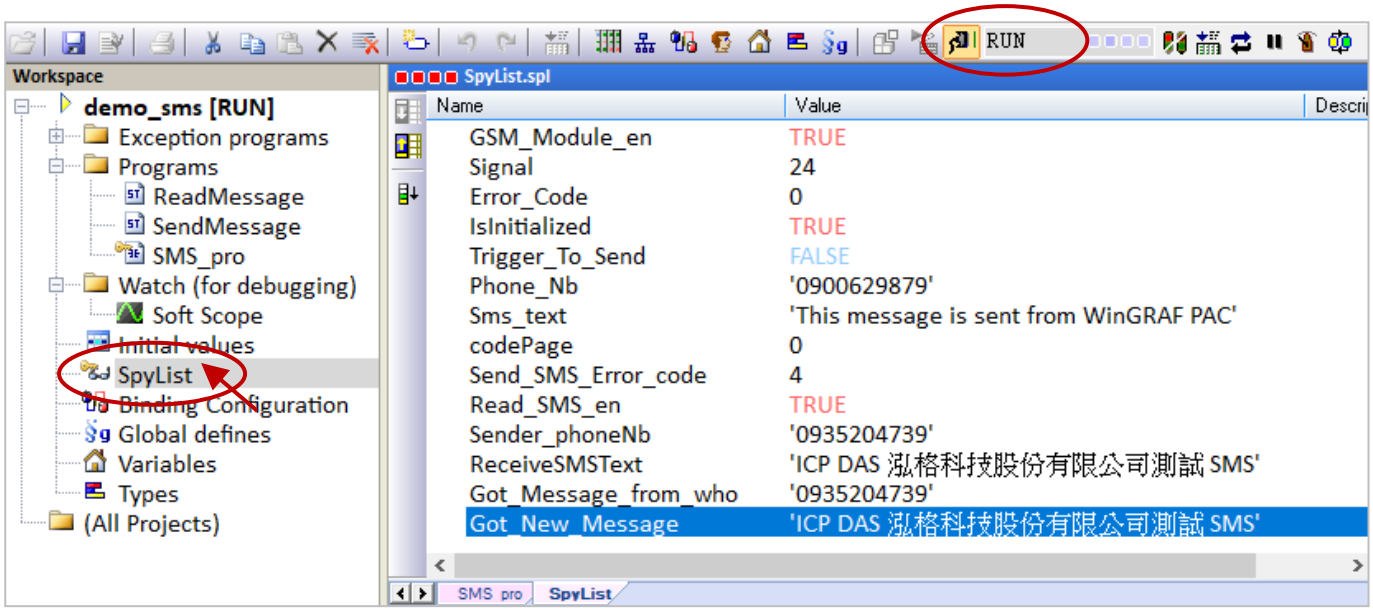


Please send an SMS message in Chinese from your cell phone (e.g., ICP DAS 泓格科技股份有限公司測試 SMS) to the phone number of the SIM card.



Name	Description
codePage1	950: Chinese ; 0: English
Sender_phoneNb	The phone number of the sender.
ReceiveSMSText	E.g., ICP DAS 泓格科技股份有限公司測試 SMS
Is_New_Message	TRUE: An new SMS message is coming.

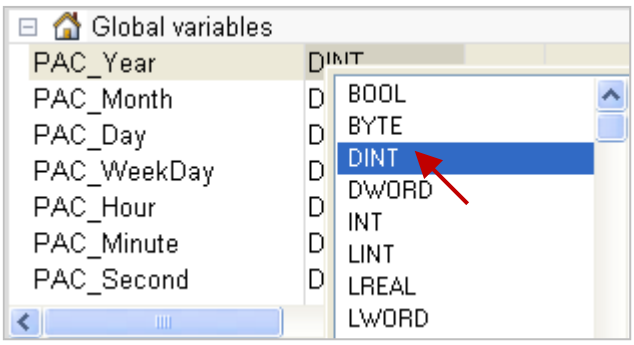
Moreover, double-click the **SpyList** to open the variables list for the test of sending or receiving an SMS message.



Visit the web pages for more related FAQ:
https://www.icpdas.com/en/faq/index.php?page=2&model=GTM-204M-4GE#399

Appendix A Data types and Ranges

Users can specify the data type of variables in the Variables Area (refer to Section 2.2.1) or in the Variables window (refer to Section 2.2.2).



Below are the available basic data types and ranges:

(*) stands for the commonly used data type.

Data types	Size in Bits		Range of Values
BOOL (*)	Boolean		TRUE or FALSE
STRING (*)	Character string		A max. of 255 characters
SINT	8-bit	Signed small integer	-128 to +127
USINT		Unsigned small integer	0 to +255
BYTE			
INT	16-bit	Signed integer	-32768 to +32767
UINT		Unsigned integer	0 to +65535
WORD			
DINT (*)	32-bit	Signed double integer	-2147483648 to +2147483647
UDINT		Unsigned double integer	0 to +4294967295
DWORD			
REAL (*)		Floating point	$\pm 3.4\times10^{-38}$ to $\pm 3.4\times10^{38}$
TIME (*)		Time of day	T#0ms to T#23h59m59s999ms
LINT	64-bit	Signed long integer	-2^{63} to $+(2^{63}-1)$
LREAL		Floating point	$\pm 1.7\times10^{-308}$ to $\pm 1.7\times10^{308}$
ULINT		Unsigned long integer	0 to $+(2^{64}-1)$
LWORD			
Note: "ULINT" and "LWORD" are not supported for Win-GRAF PAC.			

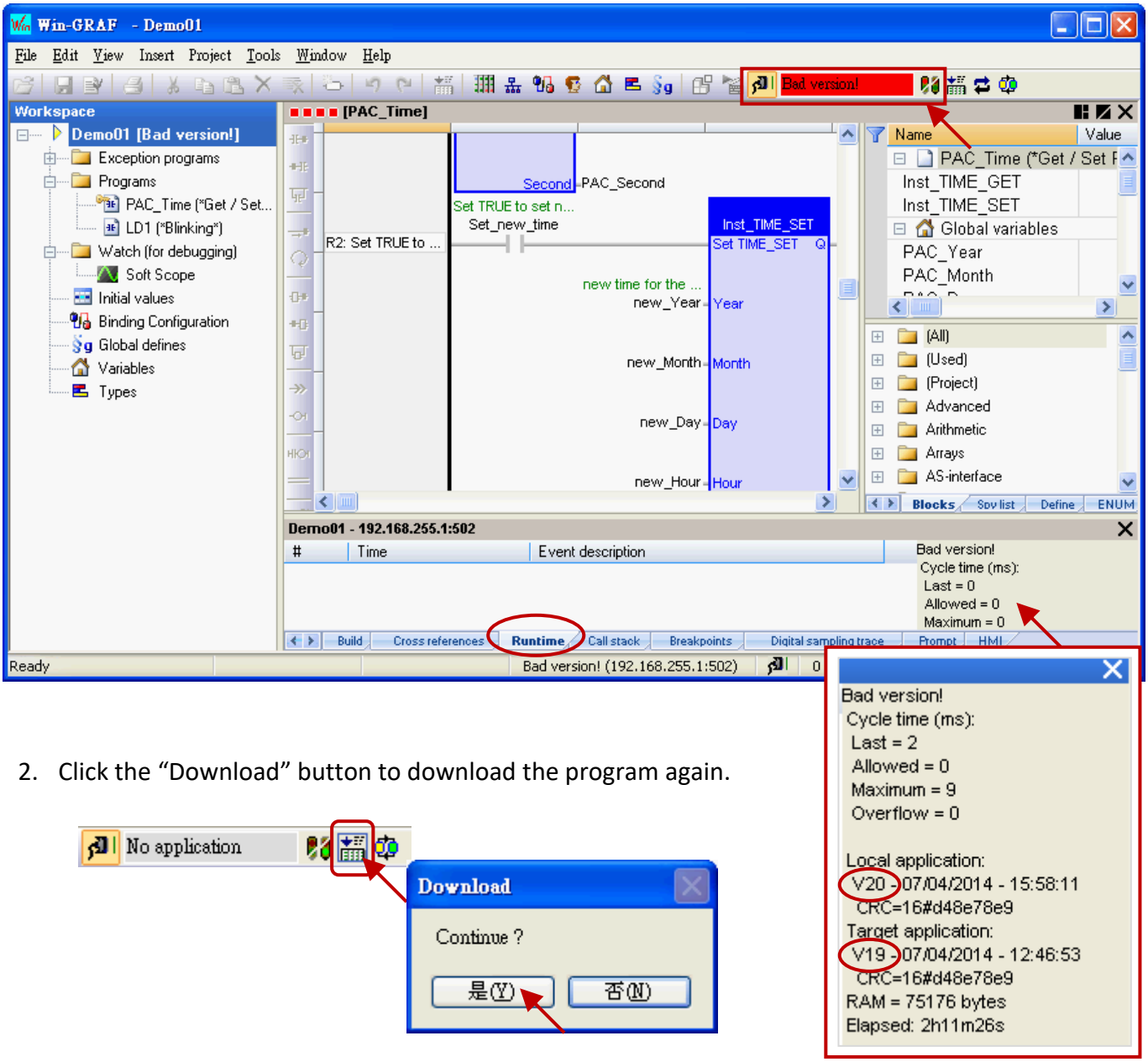
Appendix B Troubleshooting while On-Line the PAC

As the screenshot below, if the error message is showing up after connecting to the Win-GRAF PAC, follow the instruction to solve the problem.

- **The "Bad version!" error message:**
  It means that the compiled version between the PC and the PAC is different. The most common reason is that users have modified and re-compiled the program.

To solve the problem

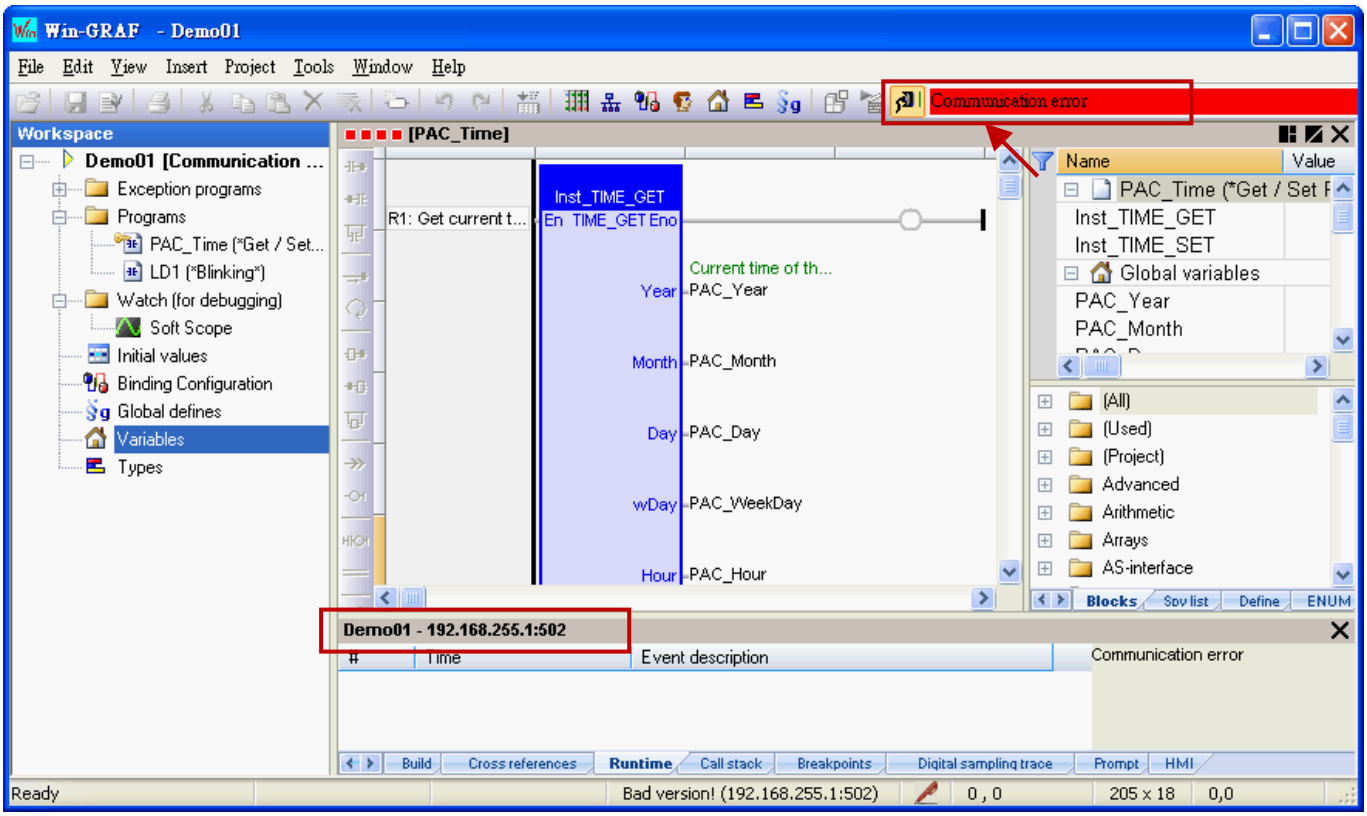
1. Click the "Stop application" button to stop the running program.



2. Click the "Download" button to download the program again.

3. The "RUN" message means that the program is working properly.

- **The “Communication error” error message:**
  A communication failure occurs between the PC and the PAC.



To solve the problem

1. Make sure your Win-GRAF PAC is started, and the network communication between the PC and the PAC is functioning properly.
2. Make sure the Win-GRAF communication IP address is set to the current IP address of the PAC. (refer to Section 2.3.4, the IP address is “192.168.255.1:502” in this example).
3. Make sure both IP addresses of the PC and PAC are on the same network segment.

Appendix C Allows the Win-GRAF to Connect via PAC's Serial Port

Generally, the Win-GRAG can connect to the PAC to download projects or perform debugging via the Ethernet port. If the user needs to enable a PAC's serial port (normally RS-232 or RS-485) to allow Win-GRAF to connect with it, follow the instructions below:

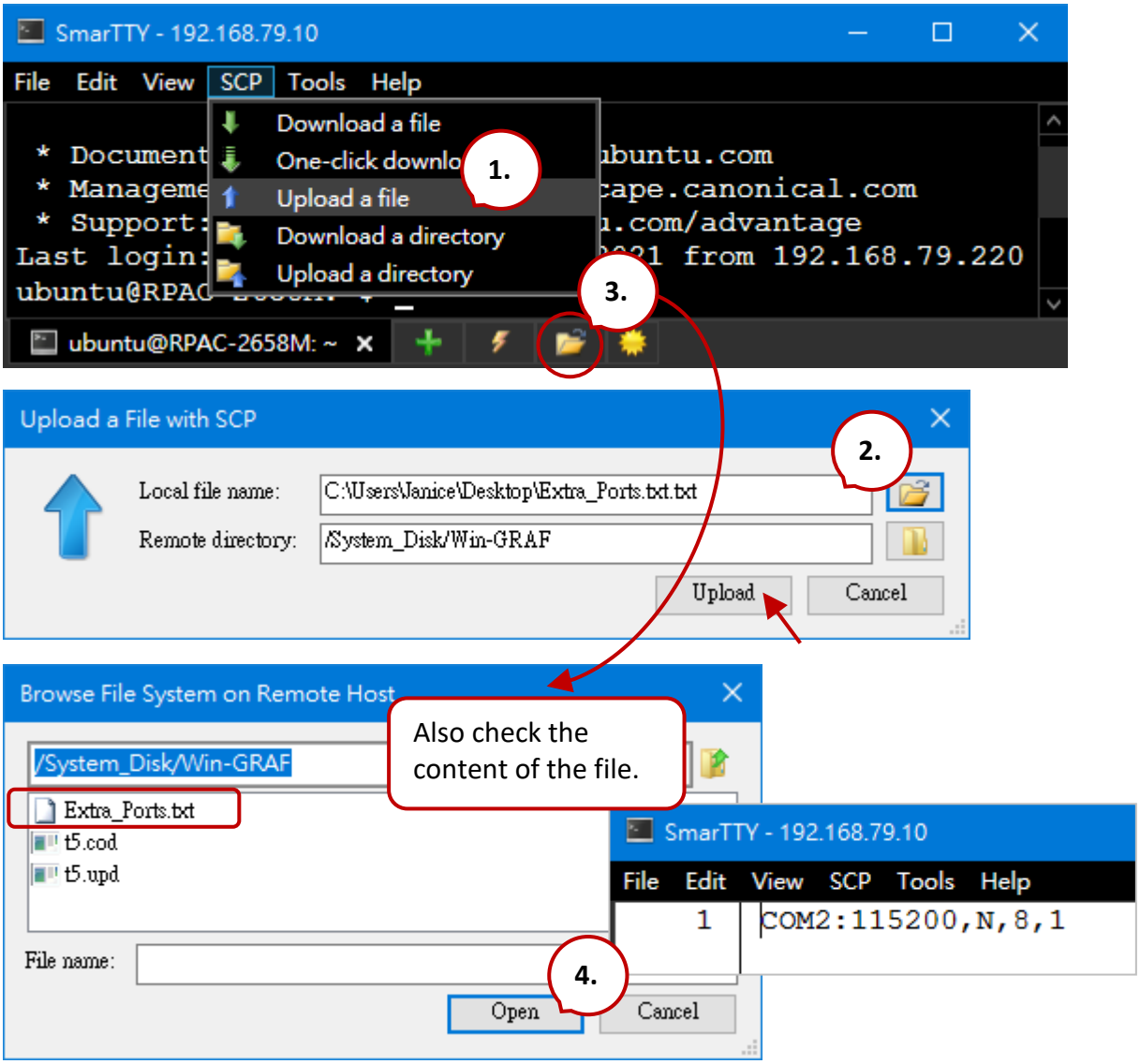
Method 1:

Once the Win-GRAF PAC is booted, it will try to read the "Extra_Ports.txt" file from the "/System_Disk/ Win-GRAF/" directory. If you want to enable the COM2 of the PAC, enter the text as follows:

COM2:115200,N,8,1

It means to enable the COM2 and the Baud Rate is 115200 bps

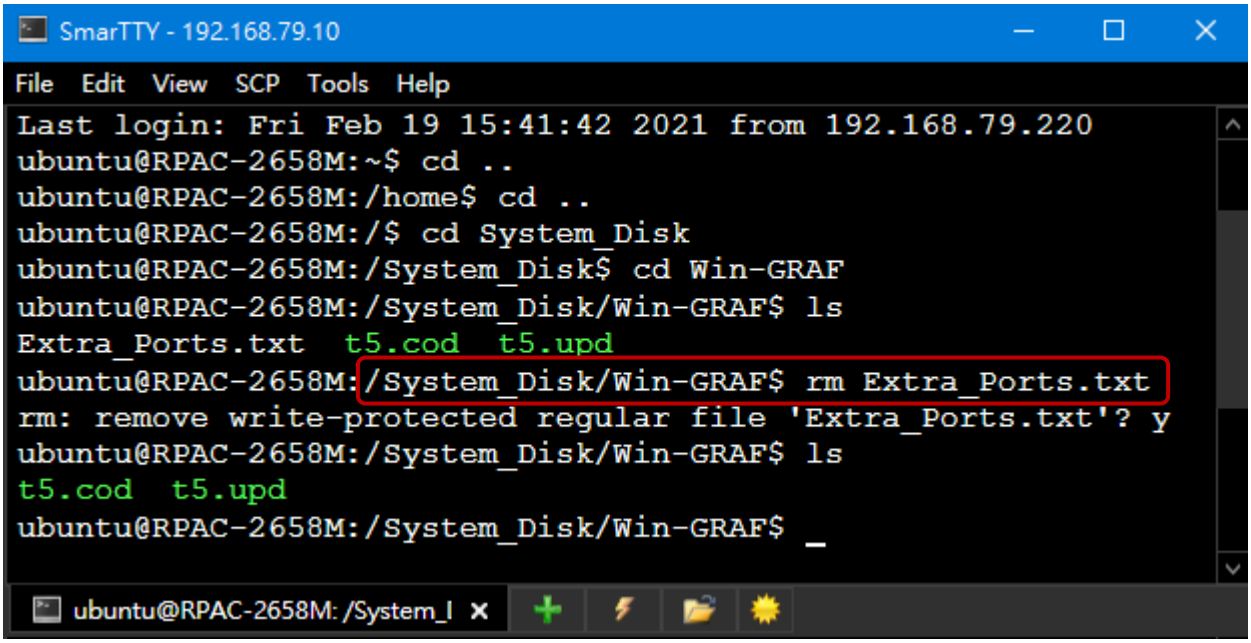
To upload the "Extra_Ports.txt" file to the "/System_Disk/ Win-GRAF/" directory by using the SSH tool (also refer to Section 13.2) and then reboot the PAC.



Disable the COM Port:

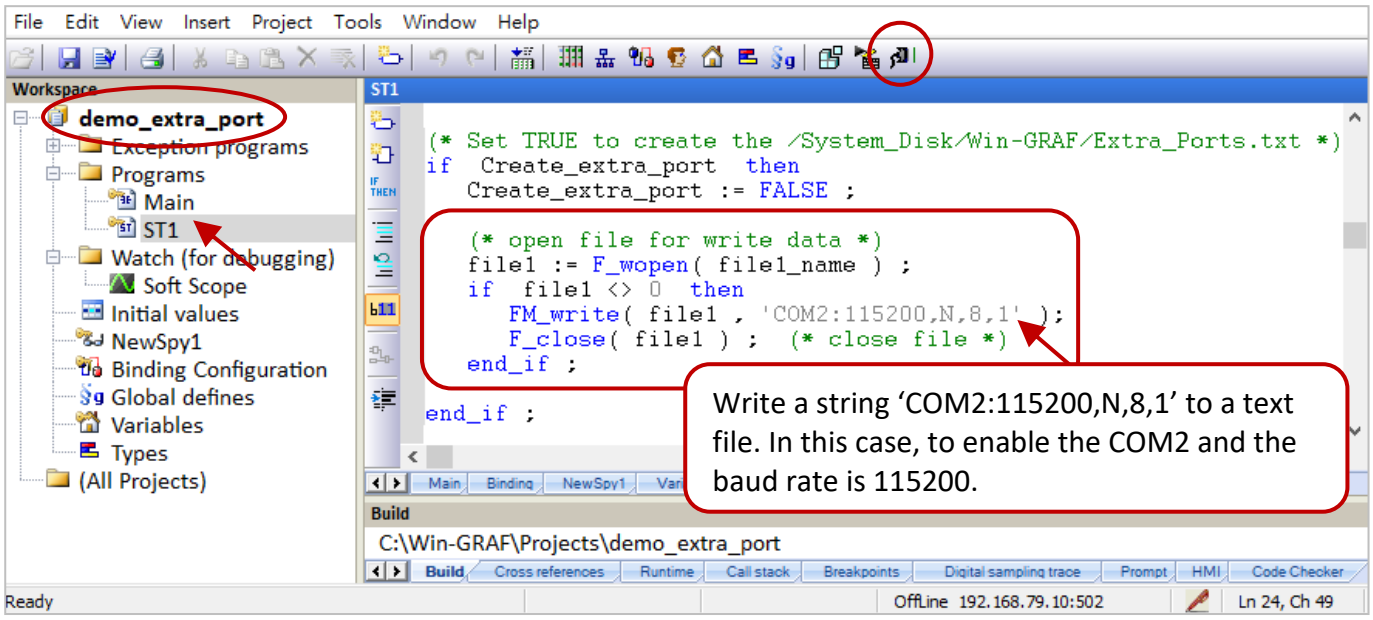
If you want to disable the COM port, simply remove the "Extra_Ports.txt" file in the " /System_Disk/

Win-GRAF/ ” directory and then reboot the PAC.

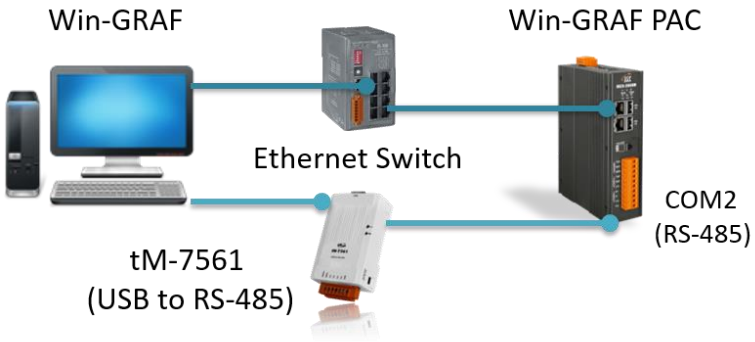


Method 2:

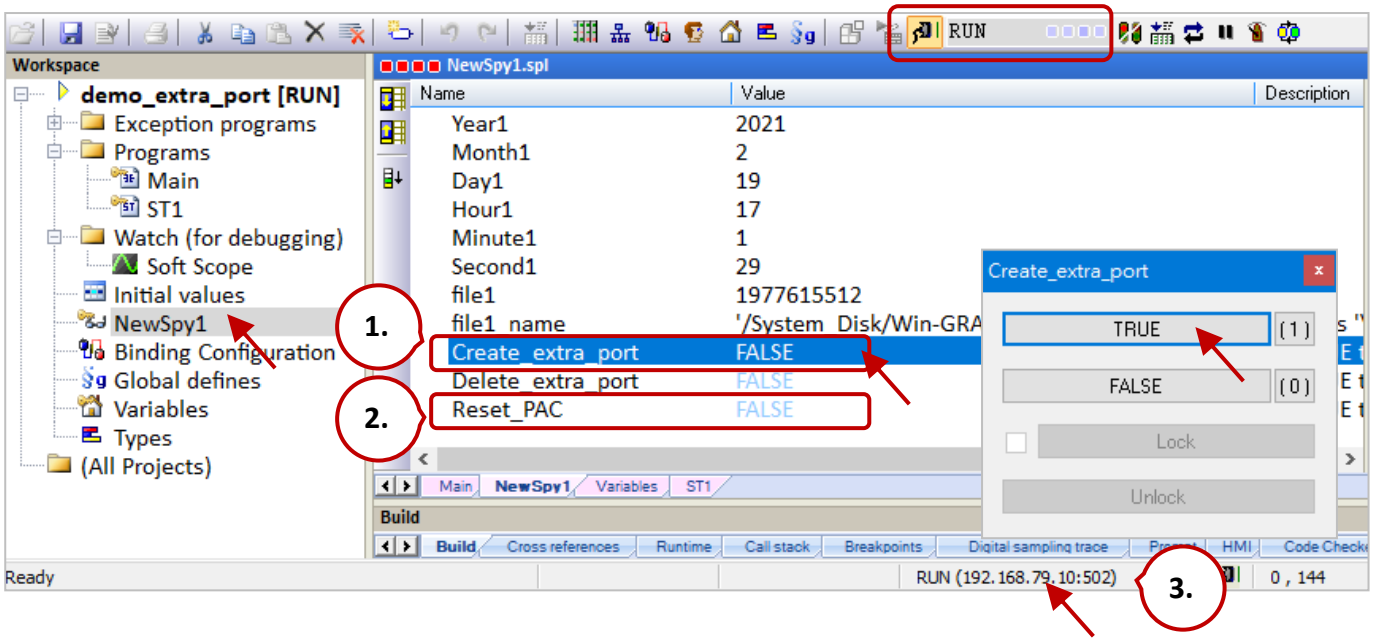
1. Download the demo program on the website (**demo_extra_port.zip**). Click the menu commands **File - Add Existing Project - From Zip...** in the Workbench to open the project.
2. Double-click the ST1 program to modify the COM port and baud rate (e.g., ‘COM2:115200,N,8,1’) of the PAC.
3. Click the On Line button to download the project to the PAC via Ethernet.



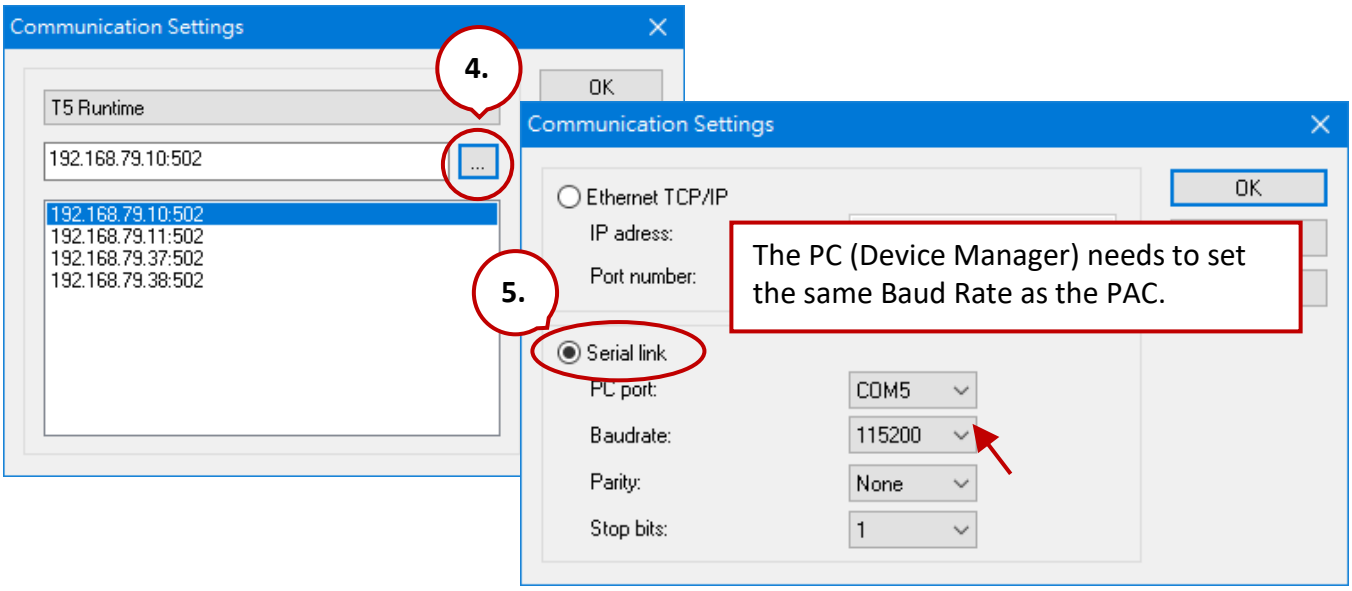
Test the Project



1. After connecting to the Win-GRAF PAC, set the "Create_extra_port" to "TRUE" in the variables list. In this example, the "Extra_Ports.txt" file will be added into the /System_Disk/Win-GRAF/ directory and write a string 'COM2:115200,N,8,1' to the file.
2. Set the "Reset_PAC" to "TRUE" to reboot the PAC automatically and apply the setting.

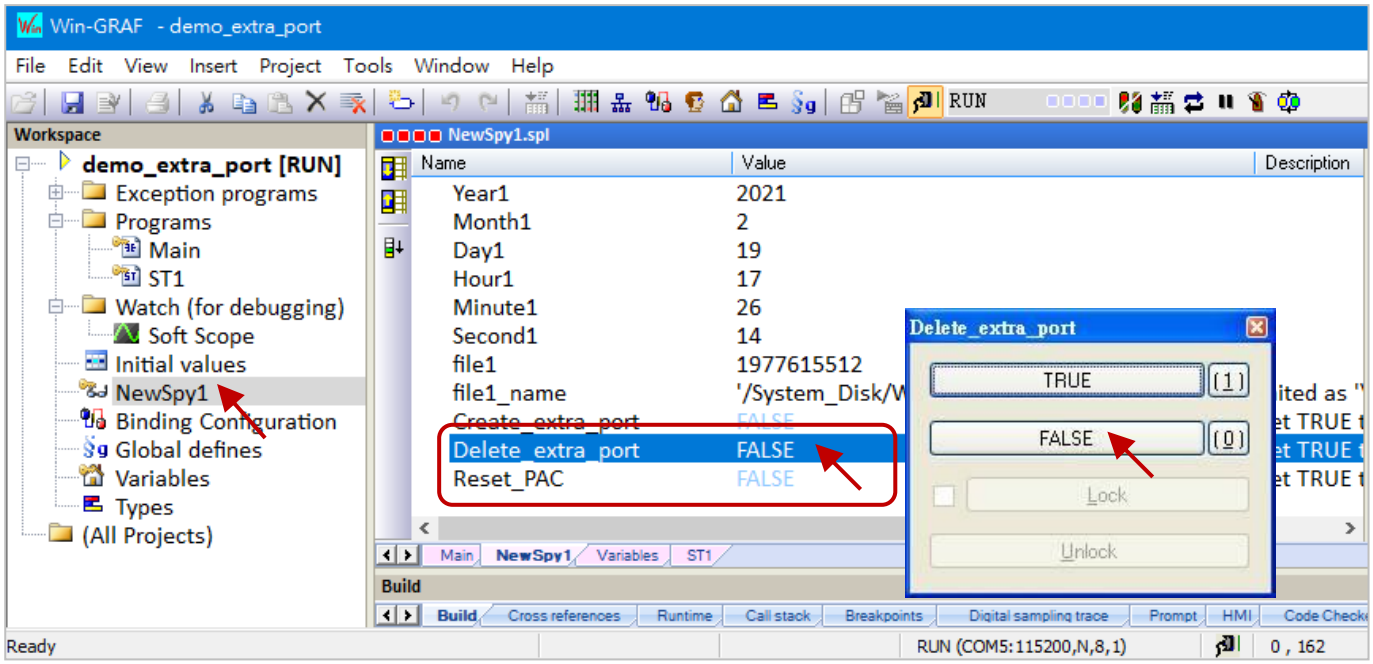


3. Disconnect the PAC and modify the Win-GRAF communication settings. Double-click the IP address on the status bar and then click the ... button. Select the **Serial link** radio button and choose the available COM port of the PC.



Disable the COM Port:

1. Set the "Delete_extra_port" to "TRUE" to remove the Extra_Ports.txt file in the " /System_Disk/ Win-GRAF/ " directory.
2. Set the "Reset_PAC" to "TRUE" to reboot the PAC.



Note:

When using method2 to download the "Extra_Ports.txt" file to the PAC, the file cannot be overwritten by using method1.

Appendix D Pin Assignment of PAC's Serial Ports

RPAC-2658M:

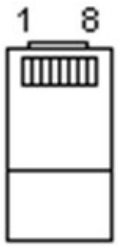
Pin assignment of COM1, COM2, and COM3.

Pin	Signal	Description
1	TxD	Console (RS-232)
2	RxD	
3	GND	
4	TxD	**COM1** (RS-232)
5	RxD	
6	ISO.GND	
7	D+	**COM2** (RS-485)
8	D-	
9	ISO.GND	
10	F.GND	Frame Ground

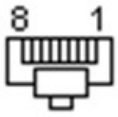
Pin	Signal	Description
1	Tx+	**COM3** (RS-422, RJ-45)
2	Tx-	
3	Rx+	
4	-	
5	-	
6	Rx-	
7	-	
8	-	

RJ-45

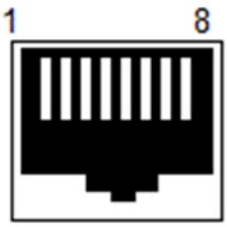
Top View



Front View



Plug



Front View

Receptacle