Win-GRAF User Manual

ICP DAS CO., LTD. would like to congratulate you own your purchase of our Win-GRAF PACs. The ease to 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.

Win-GRAF PAC (Programming Automation Controller) Series of ICP DAS includes:

ViewPAC-2000:	VP-25W8
ViewPAC-4000:	VP-4138
WinPAC-5000:	WP-5148, WP-5238, WP-5248
WinPAC-8000:	WP-8148, WP-8448, WP-8848
	WP-8138, WP-8438, WP-8838
XPAC-8000:	XP-8048, XP-8448, XP-8848
XPAC-8000-CE6:	XP-8148-CE6, XP-8348-CE6, XP-8748-CE6
XPAC-8000-Atom-CE6:	XP-8148-Atom-CE6, XP-8348-Atom-CE6, XP-8748-Atom-CE6

Legal Liability

ICP DAS CO., LTD. assumes no liability for any and all damages that may be incurred by the user as a consequence of this product. ICP DAS CO., LTD. reserves the right to change this manual at any time without notice.

ICP DAS CO., LTD. constantly strives to provide our customers with the most reliable and accurate information possible regarding our products. However, ICP DAS CO., LTD. assumes no responsibility for its use, or for any infringements of patents or other rights of third parties resulting from its use.

Trademark & Copyright Notice

The names of products are used for identification purposes only, and are the registered trademarks of their respective owners or companies.

Technical Service

Please contact local agent or email problem-report to service@icpdas.com.

Table of Contents

Chapter 1 Software Installation & Hardware Setting

1.1 Installing the Win-GRAF Workbench

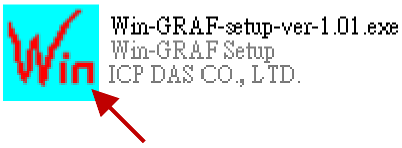
Before installing the Win-GRAF Workbench, check the installation environment on your PC.

System requirements:

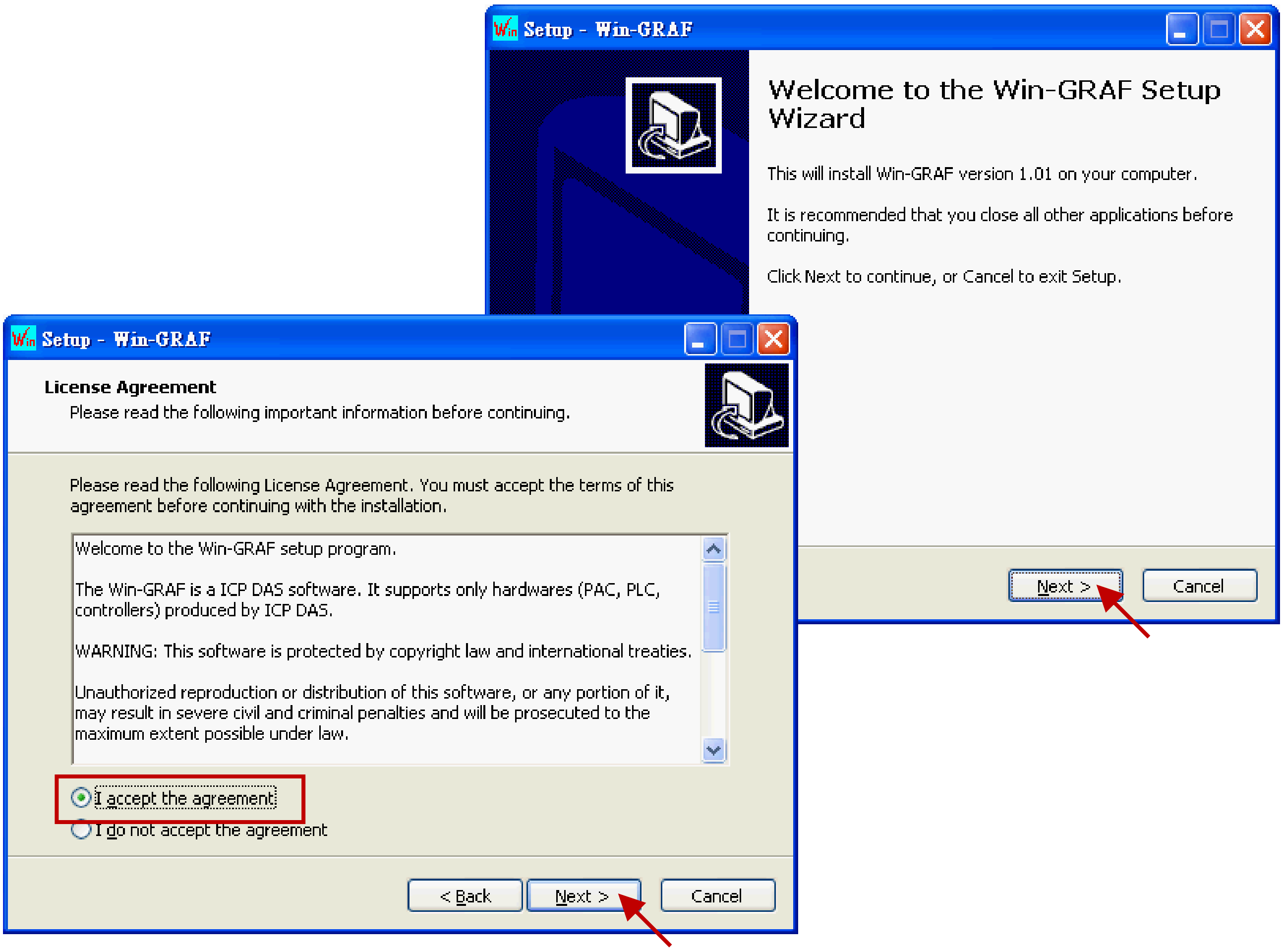
- **O.S.**: Windows XP, Windows Vista, Windows 7, Windows 8 (32-bits or 64-bits)
- **Microsoft .Net Framework 3.5** (Download it on the Microsoft web site: http://www.microsoft.com/zh-tw/download/details.aspx?id=22)
- **RAM:** 1 GB minimum (Recommended: 2 GB or more)
- **Available hard-disk space:**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

Installation Steps:

1. Double-click the "Win-GRAF-setup-ver-x.xx.exe" file in the Win-GRAF installation CD to begin the process.



2. Click "Next" to continue and then select "I accept the agreement", then click "Next" to continue.



Welcome to the Win-GRAF Setup Wizard

This will install Win-GRAF version 1.01 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

License Agreement

Please read the following important information before continuing.

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

Welcome to the Win-GRAF setup program.

The Win-GRAF is a ICP DAS software. It supports only hardwares (PAC, PLC, controllers) produced by ICP DAS.

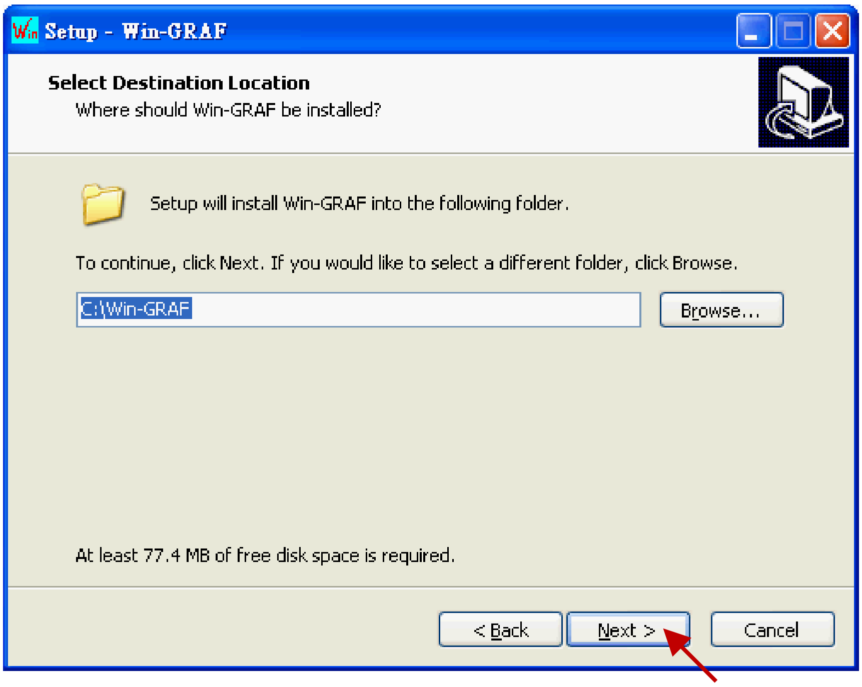
WARNING: This software is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this software, or any portion of it, may result in severe civil and criminal penalties and will be prosecuted to the maximum extent possible under law.

● I accept the agreement

○ I do not accept the agreement

3. Recommend to use the default installation path (i.e., “C:\Win-GRAF”) and then click “Next” to continue.



Select Destination Location

Where should Win-GRAF be installed?

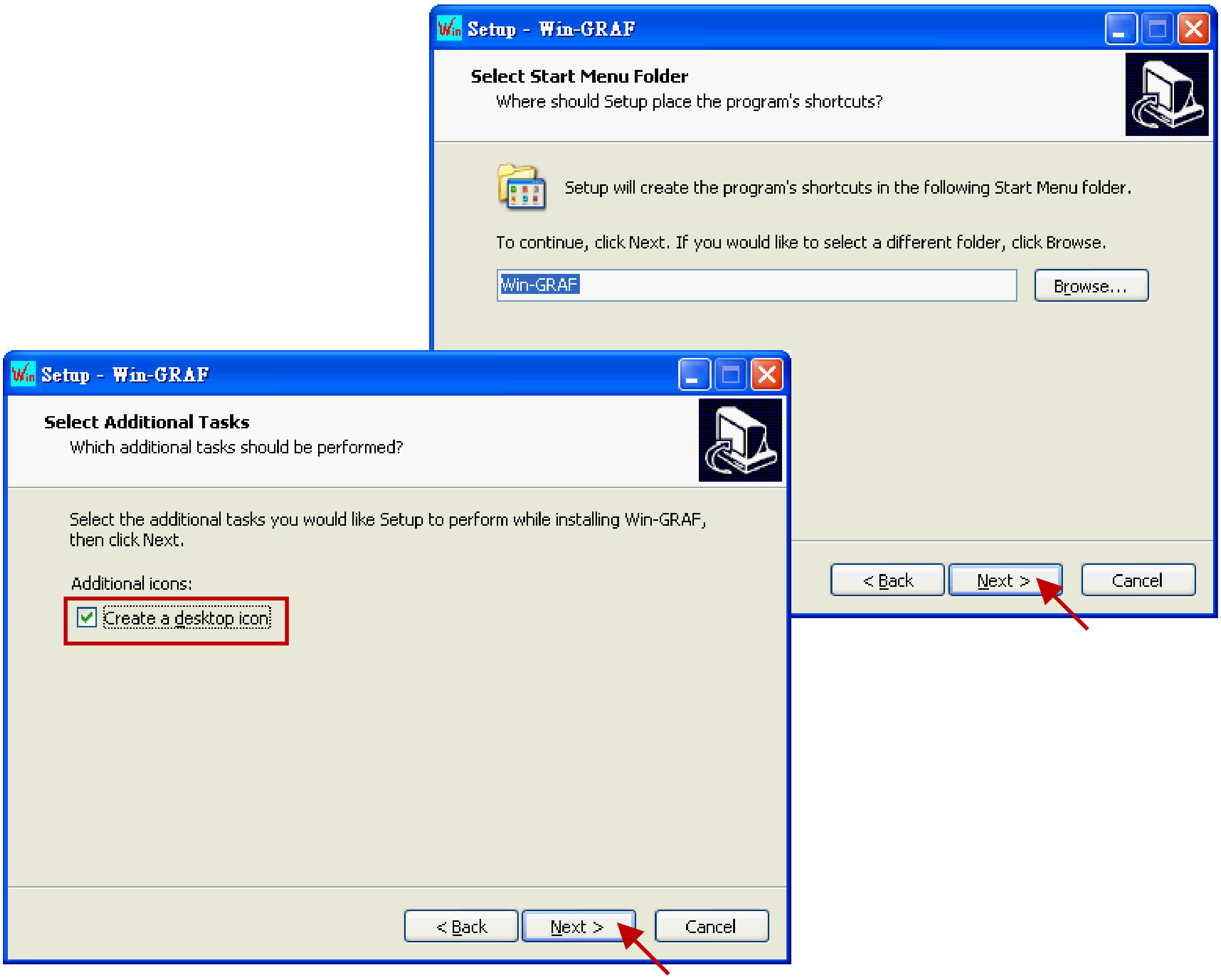
Setup will install Win-GRAF into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Win-GRAF

At least 77.4 MB of free disk space is required.

4. Click “Next” to add a “Win-GRAF” folder shortcut in the “Start” menu, and then select “Create a desktop icon” to add a desktop shortcut, then click “Next” to continue.



Select Start Menu Folder

Where should Setup place the program's shortcuts?

Setup will create the program's shortcuts in the following Start Menu folder.

To continue, click Next. If you would like to select a different folder, click Browse.

Win-GRAF

Select Additional Tasks

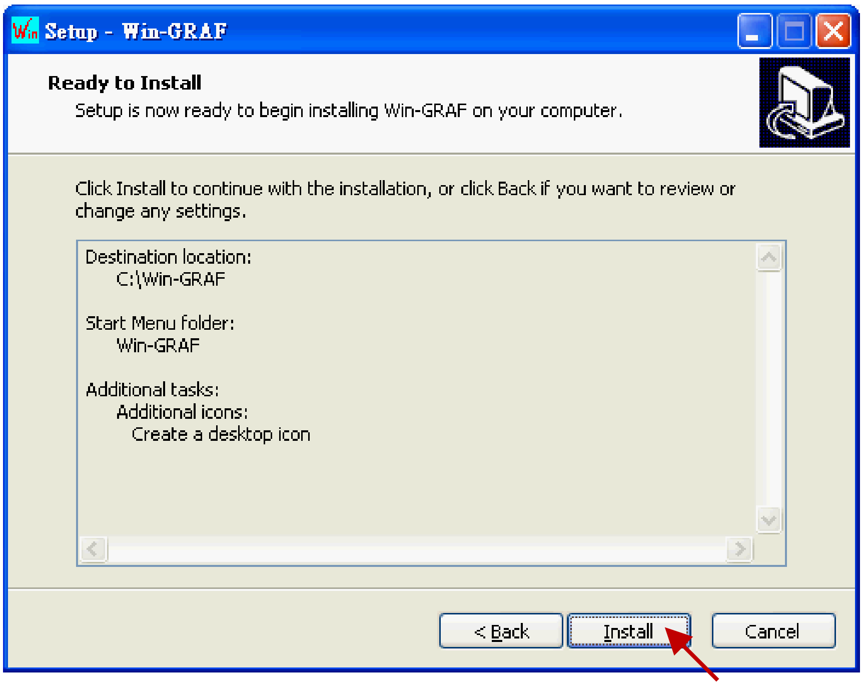
Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing Win-GRAF, then click Next.

Additional icons:

☑ Create a desktop icon

5. Click “Install” to begin installing the Win-GRAF Workbench.



Ready to Install

Setup is now ready to begin installing Win-GRAF on your computer.

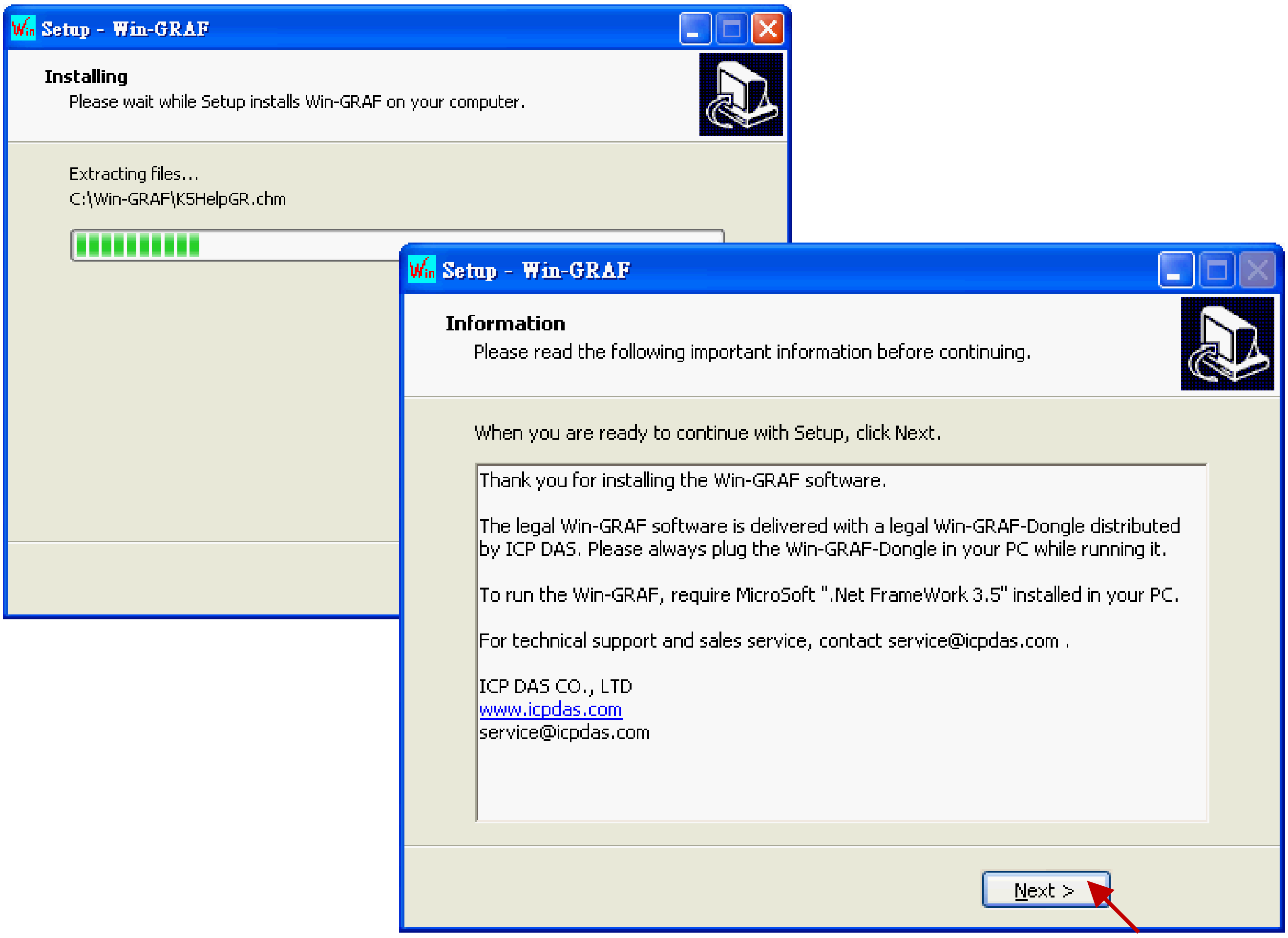
Click Install to continue with the installation, or click Back if you want to review or change any settings.

Destination location:
C:\Win-GRAF

Start Menu folder:
Win-GRAF

Additional tasks:
Additional icons:
Create a desktop icon

6. Before the end of the installation, you will see a pop up window and it displays:
   a. The legal Win-GRAF Workbench is delivered with a legal Win-GRAF Dongle distributed by ICP DAS. Please always plug the Win-GRAF Dongle in your PC while running it.
   b. To run the Win-GRAF, require Microsoft “.Net FrameWork 3.5” installed in your PC.



Installing

Please wait while Setup installs Win-GRAF on your computer.

Extracting files...
C:\Win-GRAF\K5HelpGR.chm

Information

Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

Thank you for installing the Win-GRAF software.

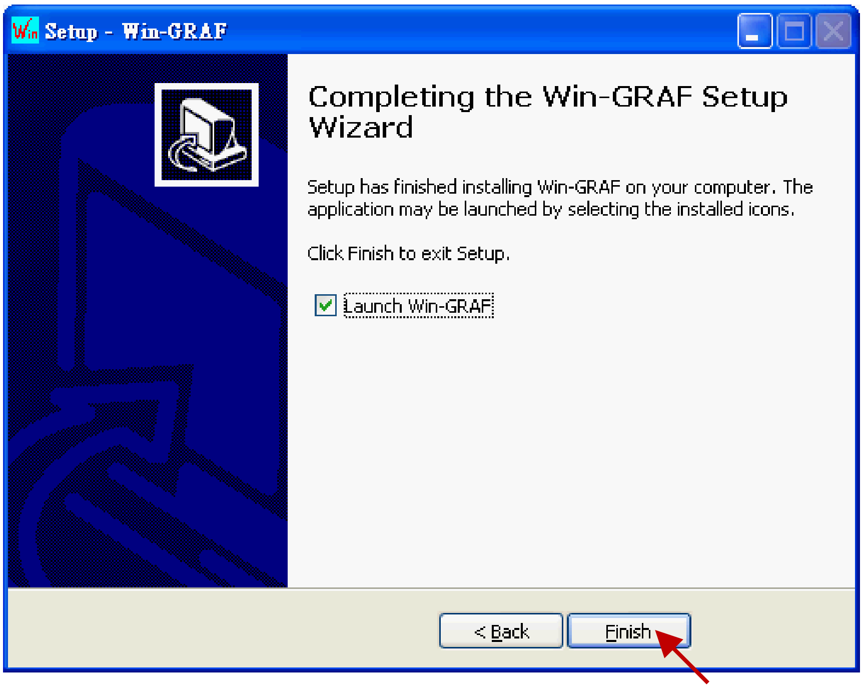
The legal Win-GRAF software is delivered with a legal Win-GRAF-Dongle distributed by ICP DAS. Please always plug the Win-GRAF-Dongle in your PC while running it.

To run the Win-GRAF, require MicroSoft ".Net FrameWork 3.5" installed in your PC.

For technical support and sales service, contact service@icpdas.com .

ICP DAS CO., LTD
www.icpdas.com
service@icpdas.com

7. By now, you have completed the Win-GRAF installation, then click “Finish” to exit this window. (Select “Launch Win-GRAF” to auto-run the Win-GRAF after completing the setup. If there is no Win-GRAF Dongle in your PC, the Win-GRAF Workbench will run in Demo mode.)



Completing the Win-GRAF Setup Wizard

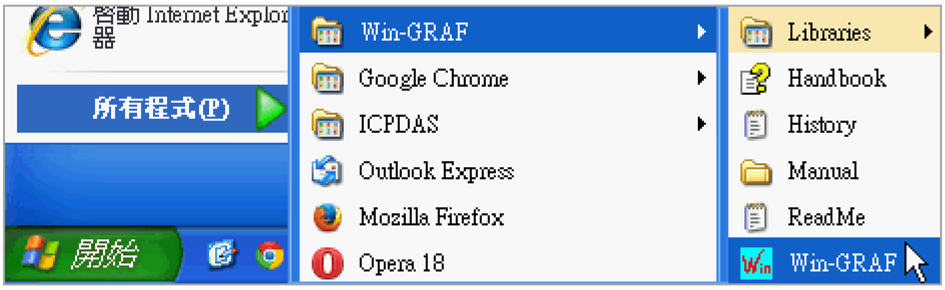
Setup has finished installing Win-GRAF on your computer. The application may be launched by selecting the installed icons.

Click Finish to exit Setup.

☑ Launch Win-GRAF

1.2 Run the Win-GRAF Workbench

Before running the Win-GRAF Workbench, make sure the Win-GRAF Dongle is plugged into your PC.



Description of the “Win-GRAF” folder:

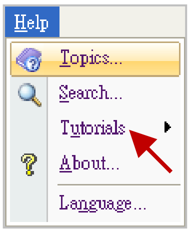
Libraries: For users to create their own function or modify an exist function.

Handbook: The manual details the software interface, programming environment, programming languages, and so on, provided by COPALP.
(Or click the [Help] > [Topics] from the Win-GRAF menu bar)

History: The modification history and features added of the Win-GRAF Workbench.

Manual: The Win-GRAF manual provided by ICP DAS.
(Or click the [Help] > [Tutorials] from the Win-GRAF menu bar, the manual is located in the path “C:\Win-GRAF\Tutorials”)

ReadMe: The notice for the Win-GRAF Workbench.



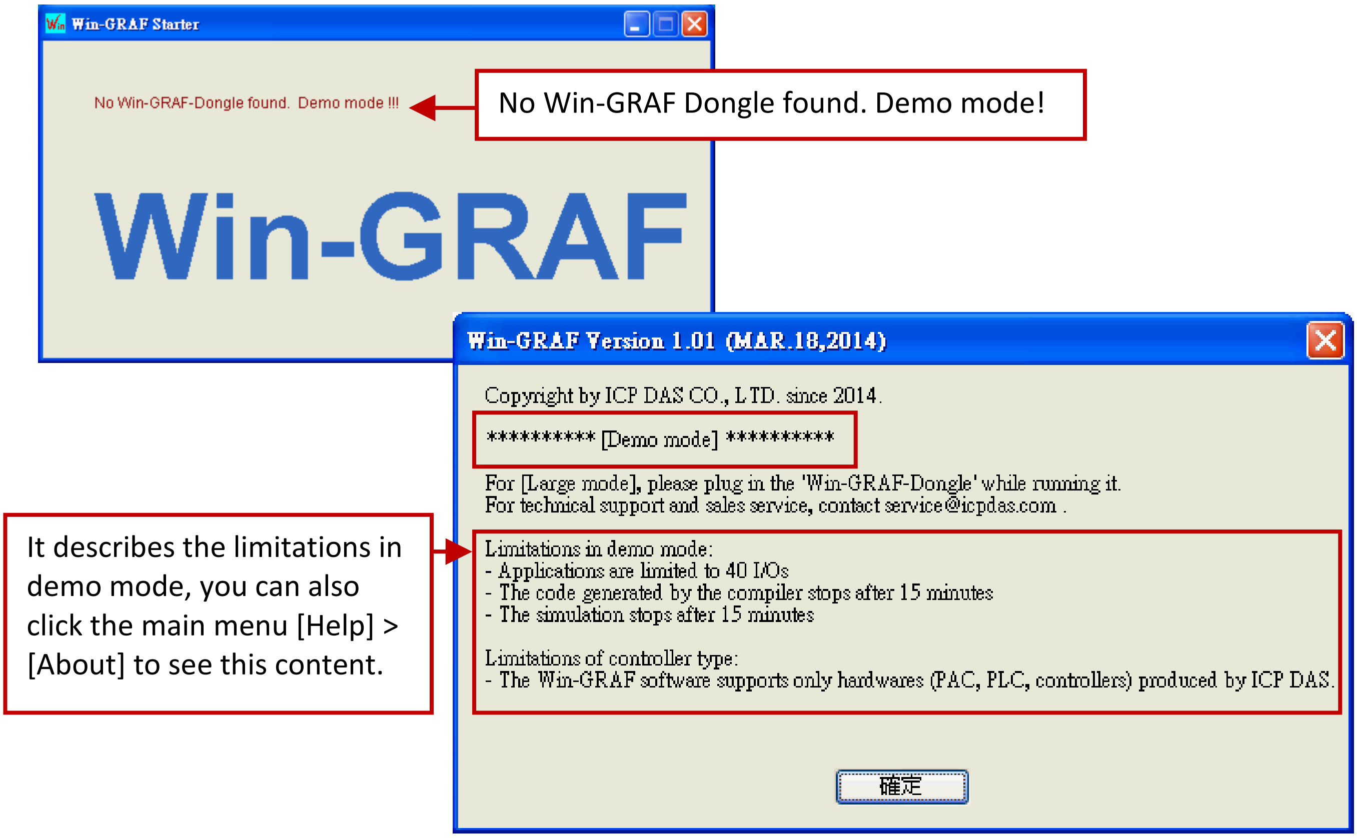
1.2.1 Win-GRAF Operating Mode

The Win-GRAF Workbench provides two operating modes:

"Demo Mode": Without using a Win-GRAF Dongle. The compiled Win-GRAF project can run for 15 minutes in the PAC. Once the time limit has expired, users must Stop/Start this project again and it only supports up to 40 I/O tags.

"Large Mode": Using a Win-GRAF Dongle. The project can run in the PAC without the time limit.

Demo Mode **-** Without using a Win-GRAF Dongle.
The start screen will show as below after running the Win-GRAF Workbench.



No Win-GRAF-Dongle found. Demo mode !!!

No Win-GRAF Dongle found. Demo mode!

Copyright by ICP DAS CO., LTD. since 2014.

********** [Demo mode] **********

For [Large mode], please plug in the 'Win-GRAF-Dongle' while running it.
For technical support and sales service, contact service@icpdas.com .

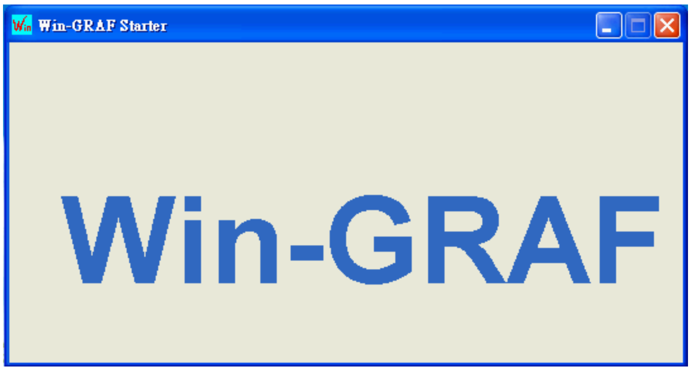
Limitations in demo mode:
- Applications are limited to 40 I/Os
- The code generated by the compiler stops after 15 minutes
- The simulation stops after 15 minutes

Limitations of controller type:
- The Win-GRAF software supports only hardwares (PAC, PLC, controllers) produced by ICP DAS.

It describes the limitations in demo mode, you can also click the main menu [Help] > [About] to see this content.

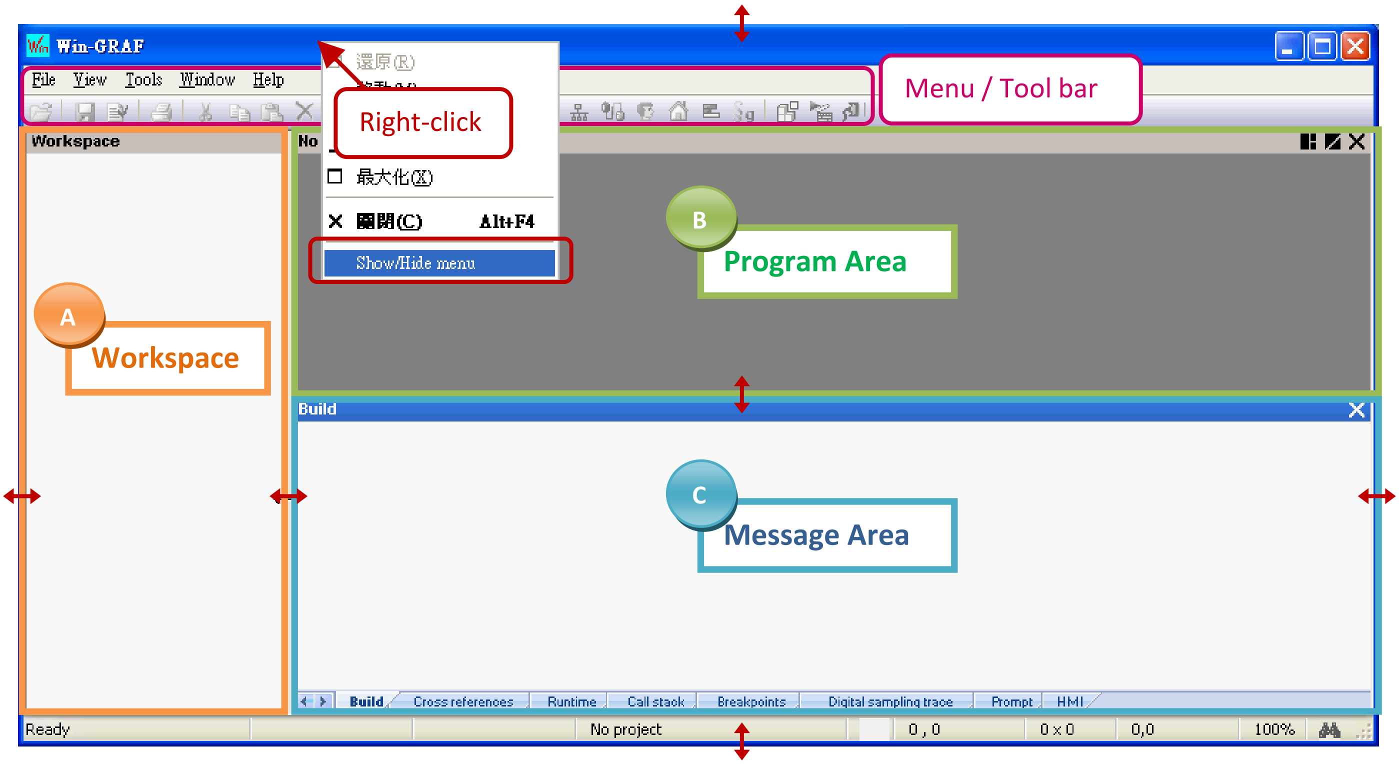
Note: If you install the Win-GRAF Dongle in the Demo Mode, you must close Win-GRAF Workbench and then start it again to make it become Large Mode.

Large Mode **-** Using a Win-GRAF Dongle.
The start screen will show as below after running the Win-GRAF Workbench.



1.2.2 Win-GRAF Operating Environment

Run the Win-GRAF and then the main screen will show as below:



Note: Mouse right-click on the top of the Window to Show/Hide the menu bar.

A. The Workspace: It allows users to create project lists, and add/open the Win-GRAF program as well as the related settings. Moreover, the new project can be created by using a project template.

B. The Program Area: It used to show/edit the program and can be divided into more function area. (Refer the Section 2.2.1)

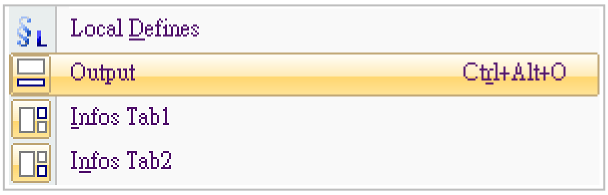
C. The Message Area: It used to show compiler messages and provides more diagnostic tools.

Tips:

1. To resize a window, click and drag the side or corner of the window to change its size.
2. Press the “F1” key to open the user manual (i.e., HTML Help).

Hind or Show the window

If you carelessly closed the Variables pane or the Message Area during the programming, you can click the menu bar "View" and select the following options to open this window again.



Output: It means the Message Area.

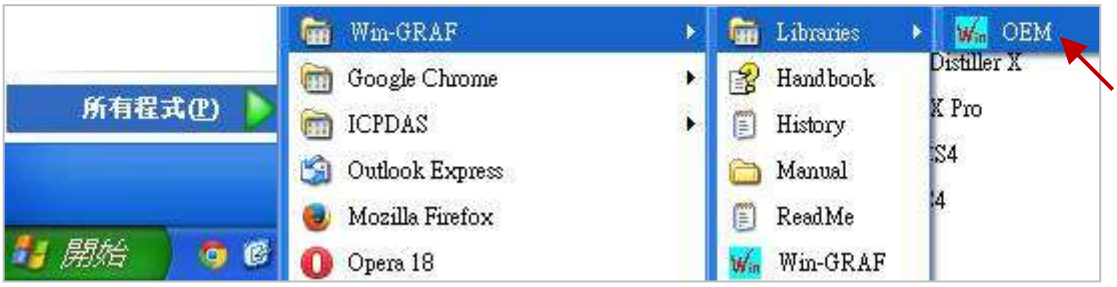
Infos Tab1: It means the Program Area – Variables pane (refer the Section 2.2.1).

Infos Tab2: It means the Program Area - Function Blocks pane (refer the Section 2.2.1).

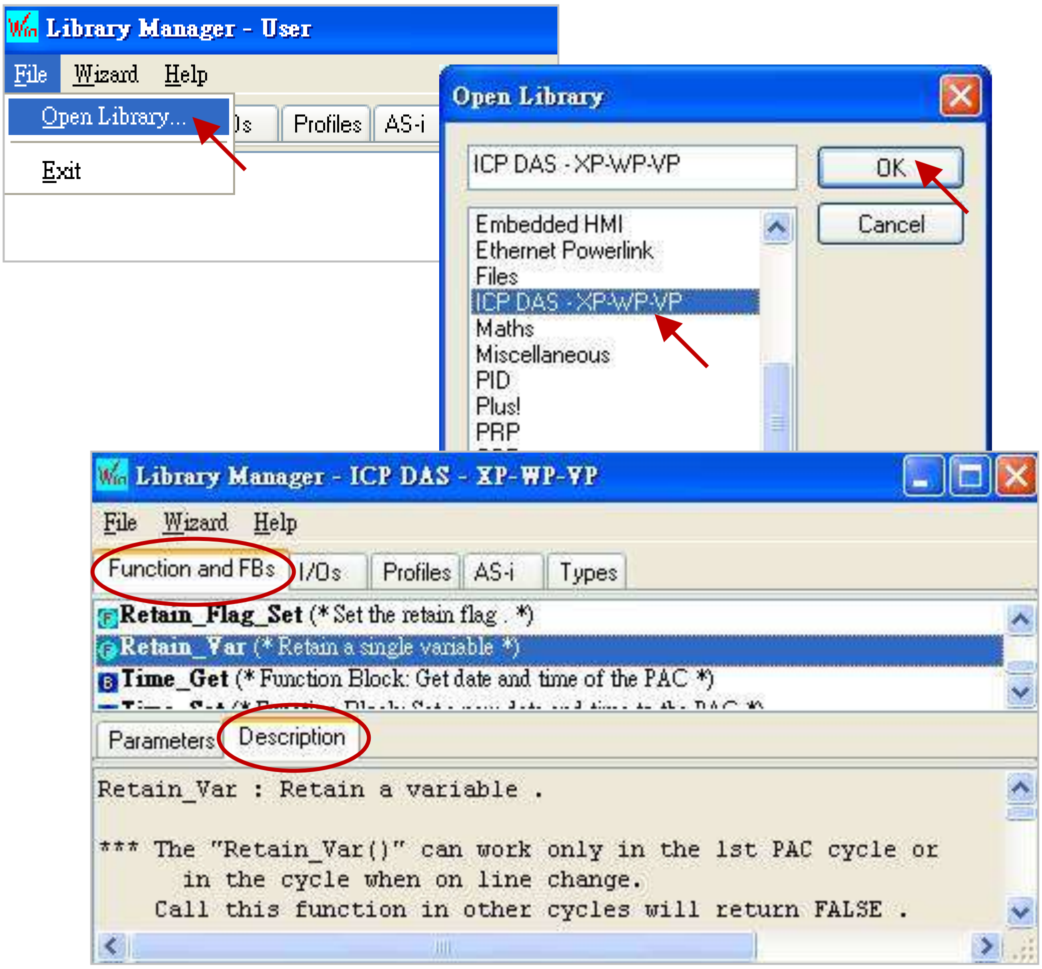
1.2.3 Win-GRAF Library Manager

Win-GRAF Workbench provides a Library Manager that can be used to look up all descriptions for Functions, Function Blocks and I/O Boards.

1. To begin this, click the Start button and click "All Programs" > "Win-GRAF" > "Libraries" > “OEM”.



2. In the “Library Manager” window, click the menu bar “File” > “Open Library” and select “ICP DAS – XP-WP-VP” then click “OK”.
3. Select any title in the “Function and FBs” tab and click the “Description” to view the usage of this Function or Function Block; Select any title in the “I/Os” tab and click the “Description” to view the usage of this I/O Boards.

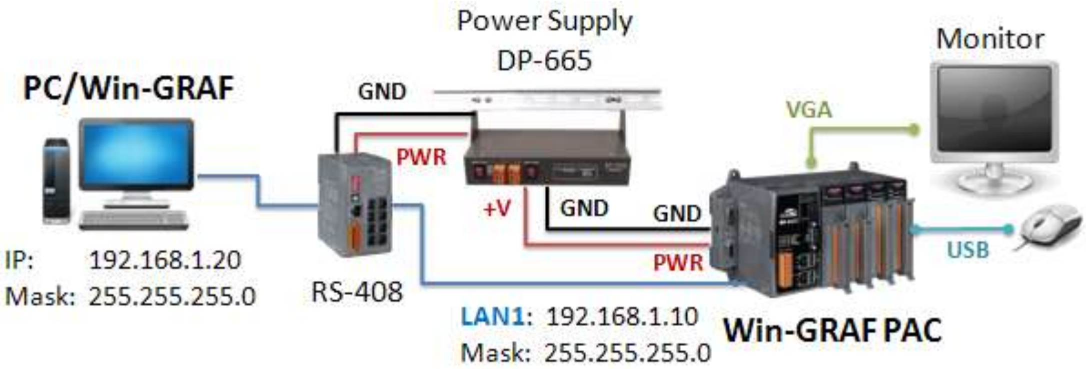


```
Retain_Var : Retain a variable .

*** The "Retain_Var()" can work only in the 1st PAC cycle or
      in the cycle when on line change.
    Call this function in other cycles will return FALSE .
```

1.3 Setting the Win-GRAF PAC's IP Address

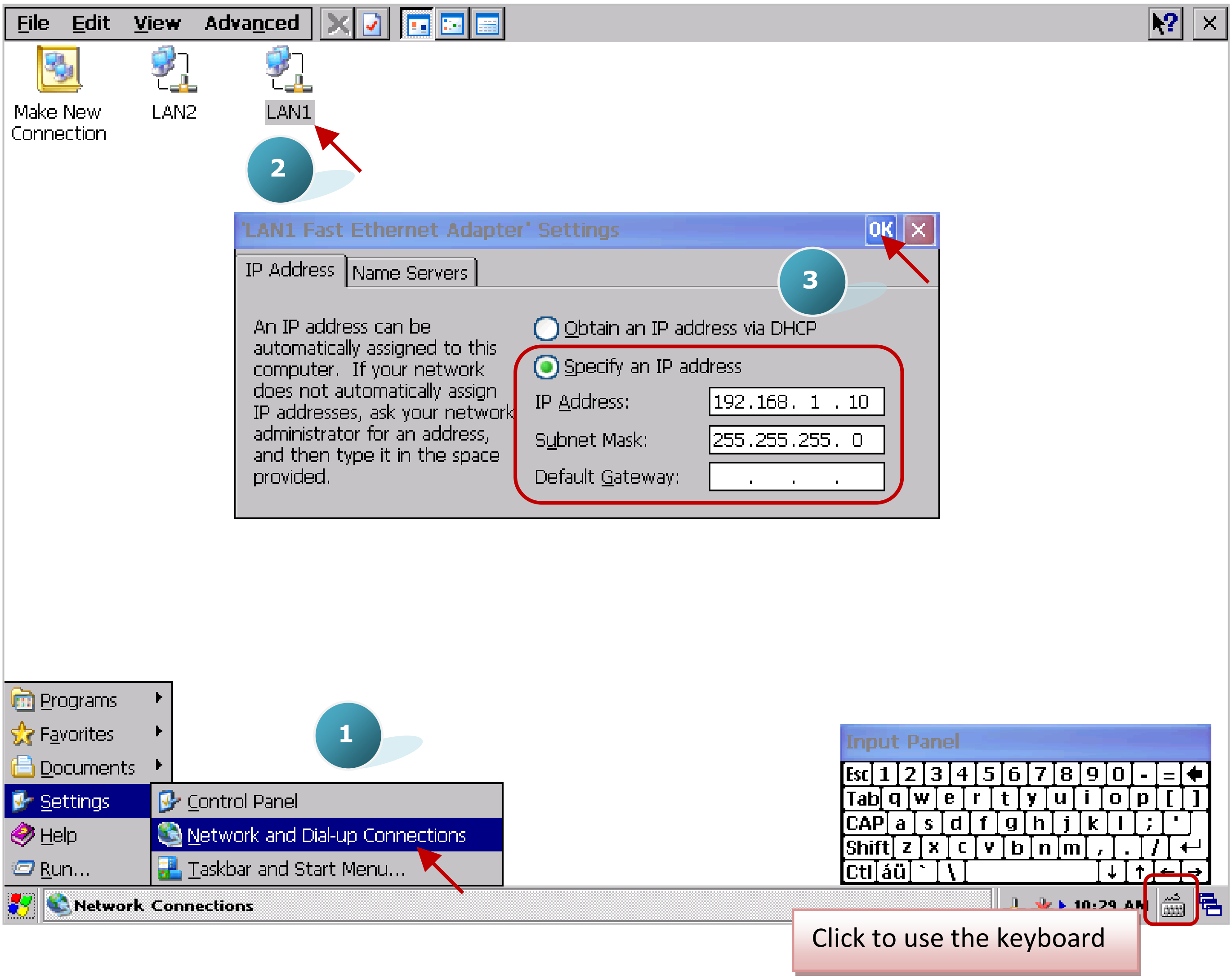
For connecting with the PAC, the Win-GRAF Workbench needs to know the PAC IP. The following will show you how to set up the PAC IP. Using the XPAC (XP-8xx8, XP-8xx8-CE6, XP-8xx8-Atom-CE6) and the WinPAC (WP-8xx8, WP-5xx8) as the example:

Hardware Wiring Diagram



PAC Side

Using the USB mouse that connected to the PAC, and click "Start" > "Settings" > "Network and Dial-up Connections" on the lower left corner of the monitor, then double-click the "LAN1" (or LAN2), then fill in a proper IP address.



An IP address can be automatically assigned to this computer. If your network does not automatically assign IP addresses, ask your network administrator for an address, and then type it in the space provided.

○ Obtain an IP address via DHCP

● Specify an IP address

IP Address: 192.168. 1 . 10

Subnet Mask: 255.255.255. 0

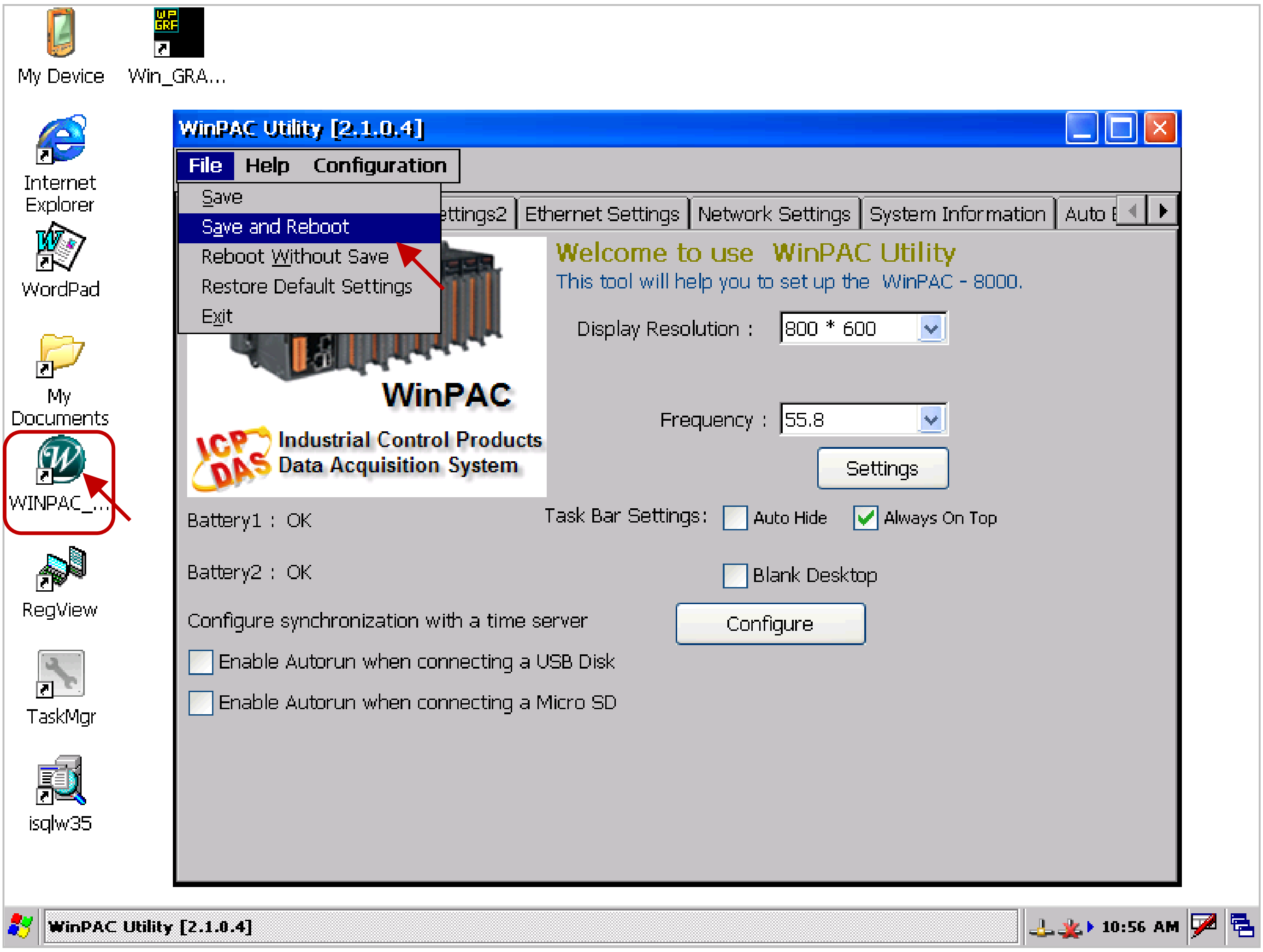
Default Gateway: . . .

Click to use the keyboard

Open the “WinPAC_Utility.exe” (or “XPAC_Utility.exe”) on the desktop (or \System Disk\tools\).Then, click “File” > “Save and Reboot” to reboot the PAC.

Notice:

For connecting properly, the PC/Win-GRAF IP and the PAC IP must in the same network segment.
For example, set the PC’s IP to “192.168.1.20” (Mask: 255.255.255.0).



Welcome to use WinPAC Utility
This tool will help you to set up the WinPAC - 8000.

Display Resolution : 800 * 600

Frequency : 55.8

Battery1 : OK

Battery2 : OK

Task Bar Settings: □ Auto Hide ☑ Always On Top

□ Blank Desktop

Configure synchronization with a time server

□ Enable Autorun when connecting a USB Disk

□ Enable Autorun when connecting a Micro SD

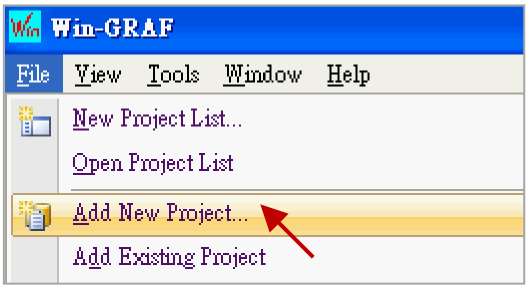
Chapter 2 A Simple Win-GRAF Program

2.1 Creating a New Win-GRAF Project

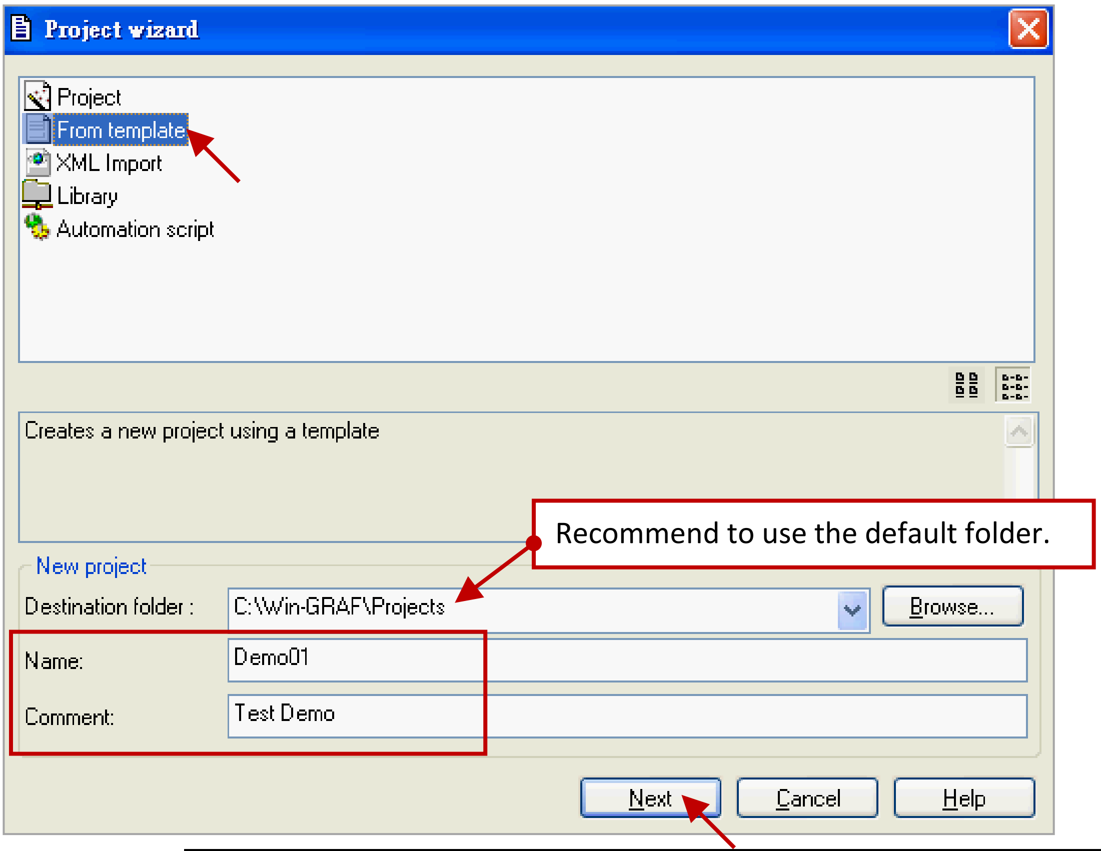
The following sections will introduce you to a simple template project that used to get/set (read/write) the Win-GRAF PAC‘s system time. Follow the steps below to complete this demo program.

2.1.1 Creating a template project (Demo01)

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



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 a "ICPDAS_template" option provided by Win-GRAF Workbench, just click "Next" to continue.



Creates a new project using a template

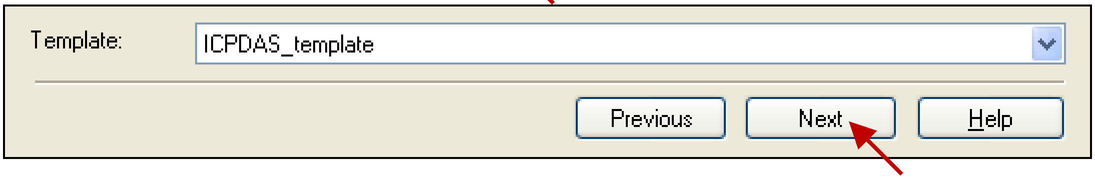
Recommend to use the default folder.

New project

Destination folder : C:\Win-GRAF\Projects

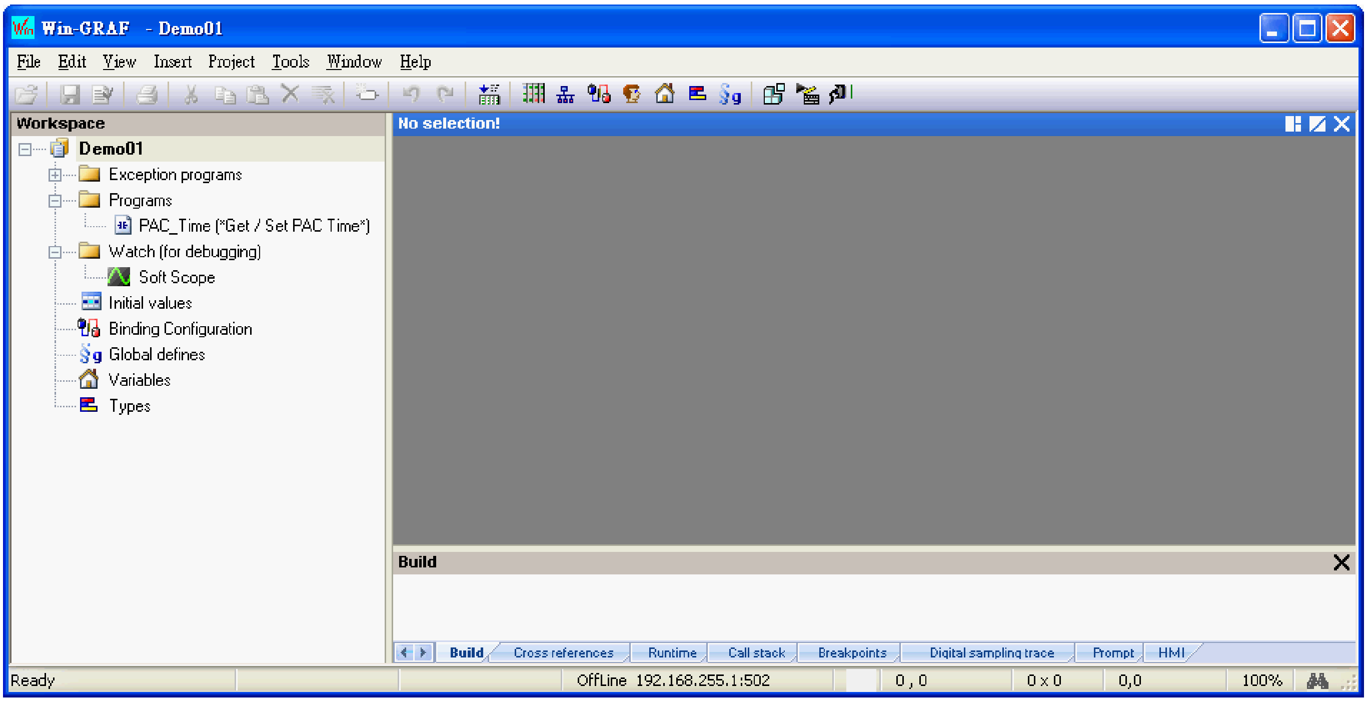
Name: Demo01

Comment: Test Demo

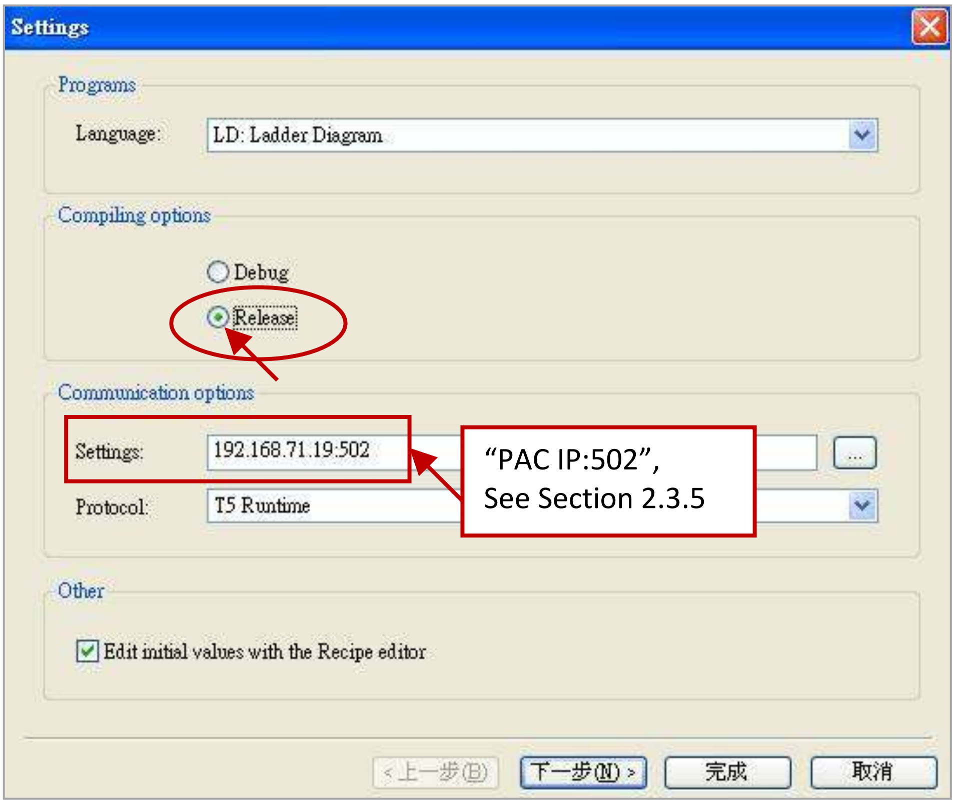


Template: ICPDAS_template

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



Note: In the demo01, we use a “From template” way to create this project. If you select "Project" in the step2, click the "Release" in the “Compiling options" setting. The others setting can be done in the following sections, just click "Next" and then "OK" button to end the settings.


Settings

Programs

Language: LD: Ladder Diagram

Compiling options

○ Debug

● Release

Communication options

Settings: 192.168.71.19:502

Protocol: T5 Runtime

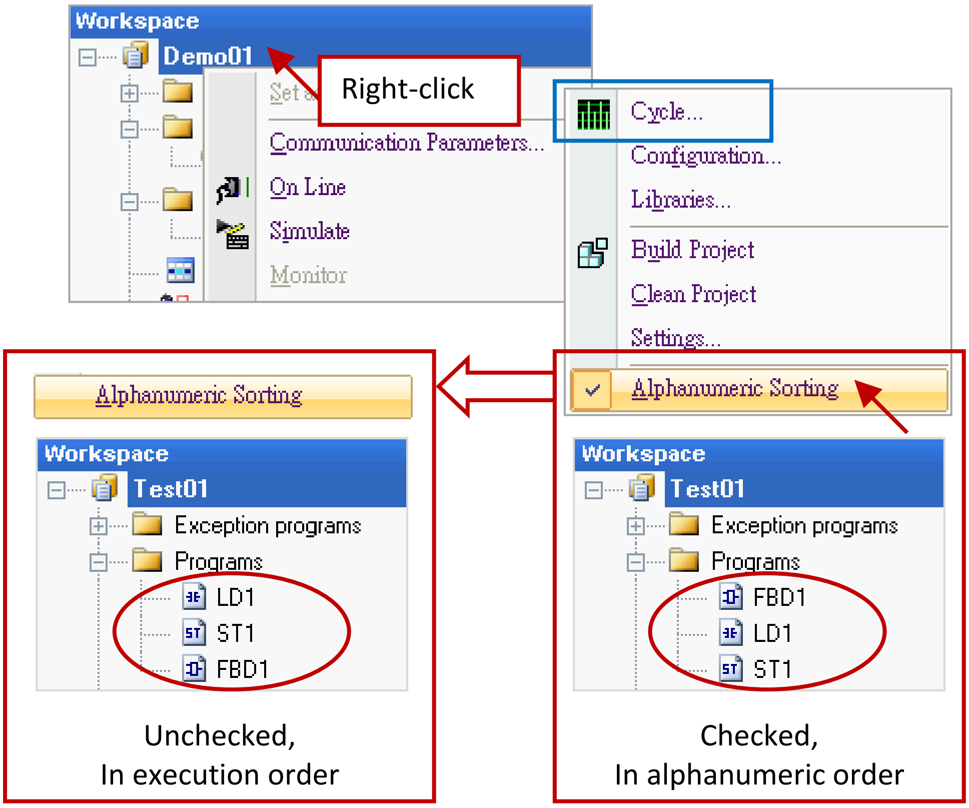
Other

☑ Edit initial values with the Recipe editor

2.1.2 Important project settings

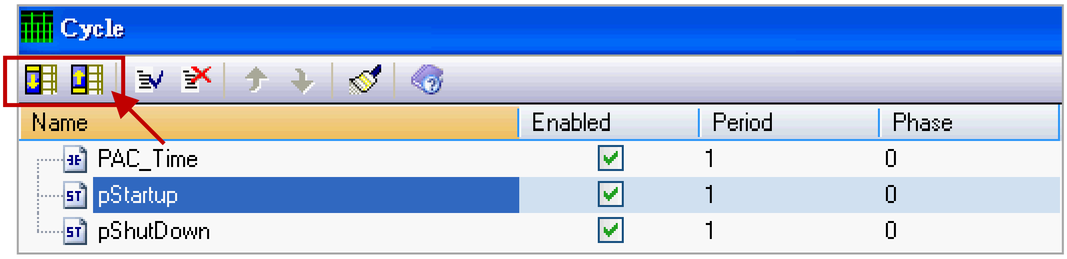
There are two important settings must be done after creating the project.

1. In the "Workspace", mouse right-click the project name (e.g., "Demo01") and then uncheck the "Alphanumeric Sorting" option (the last one). If unchecked, means the programs are in execution order ; If checked, means the programs are in alphanumeric order (e.g., FBD1, LD1, ST1).



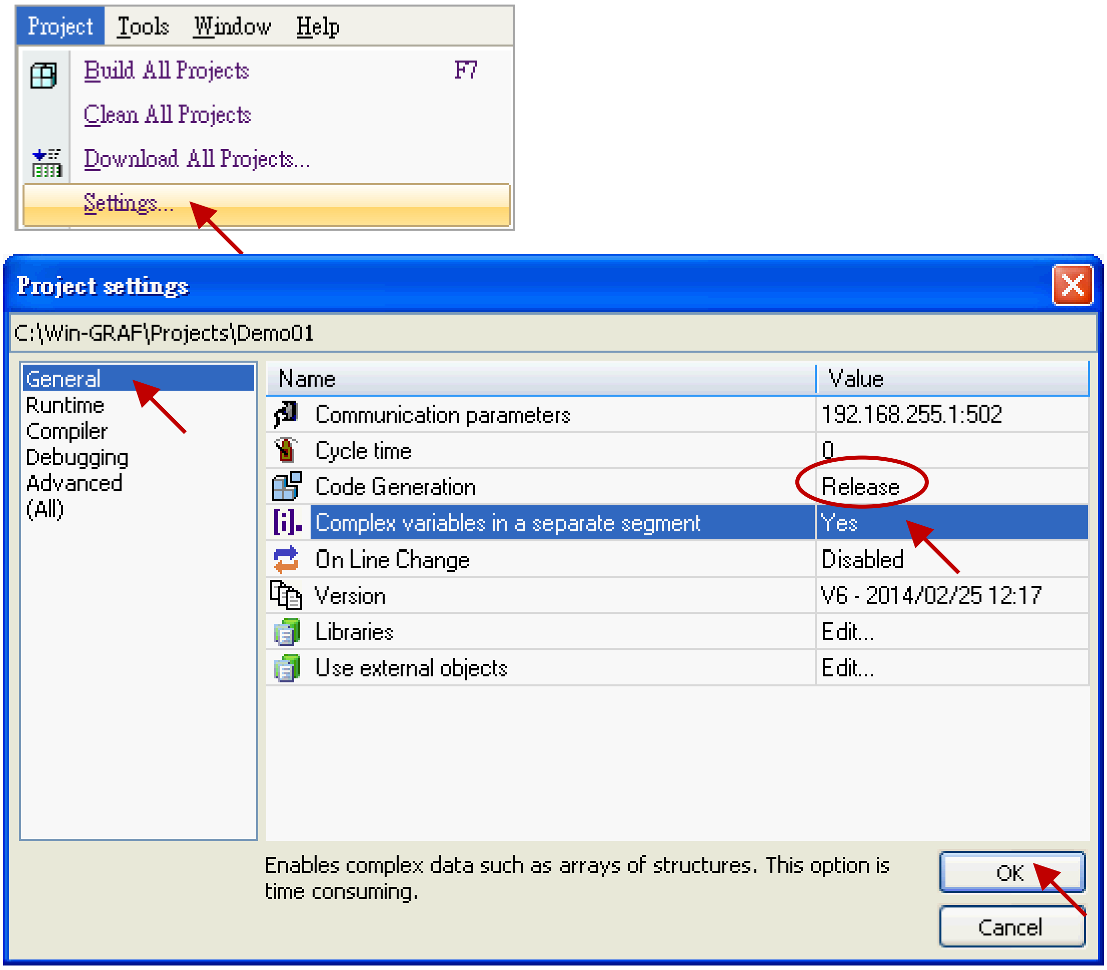
Change the execution order of programs:

Note: If you want to change the execution order of programs, mouse right-click the project name (e.g., "Demo01") and click "Cycle" (as the screenshot above) to open the settings window, then click the "Move Up" or "Move Down" button to change the order.


Cycle

Name	Enabled	Period	Phase
PAC_Time	☑	1	0
pStartup	☑	1	0
pShutDown	☑	1	0

2. If using the "Project" way to create a new project (in this example, we use the “From template” way, refer the Section 2.1.1 - Step 2), click the"Project" > "Settings..." from the menu bar to open the "Project settings" window. Click the "General" option and set the "Complex variables in a separate segment" to "Yes" to allow the using of complex data structures, such as arrays. Finally, click "OK" to exit the window.



C:\Win-GRAF\Projects\Demo01

Name	Value
Communication parameters	192.168.255.1:502
Cycle time	0
Code Generation	Release
Complex variables in a separate segment	Yes
On Line Change	Disabled
Version	V6 - 2014/02/25 12:17
Libraries	Edit...
Use external objects	Edit...

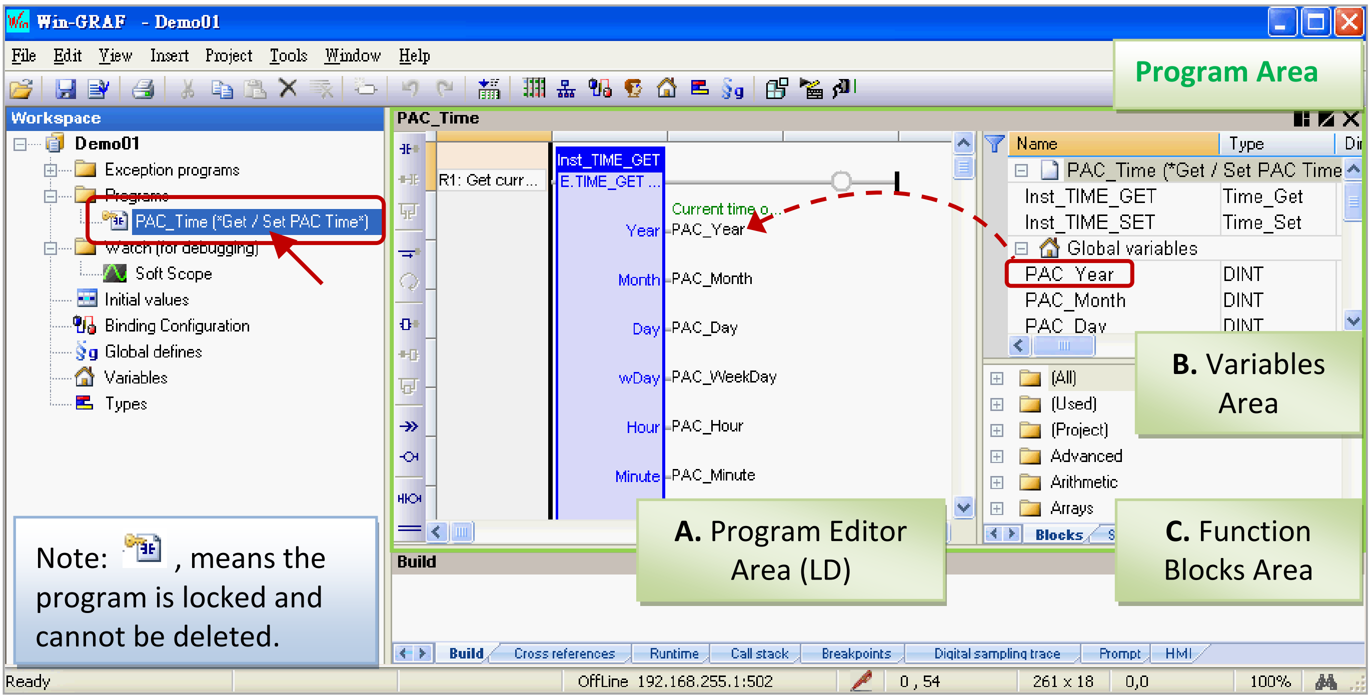
Enables complex data such as arrays of structures. This option is time consuming.

(**Note:** The "Code Generation" must be set to "Release".)

2.2 Introduction of the Project

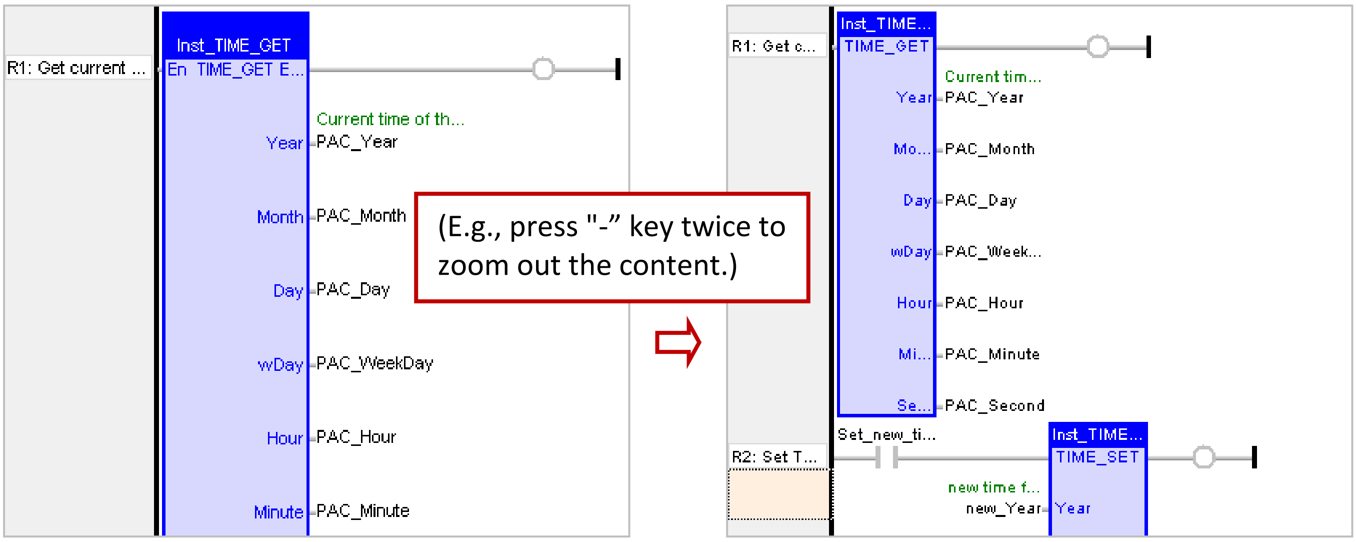
2.2.1 Demo01 - LD Program

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:



Note: 🔒, means the program is locked and cannot be deleted.

Tips: Mouse click the Program Editor Area, and press the "+" or "-" key to zoom in or zoom out the content.

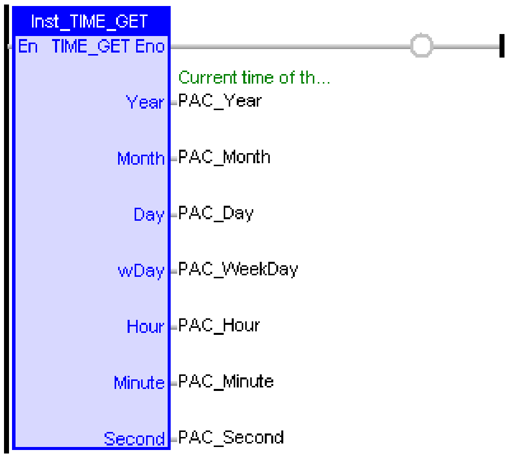


(E.g., press "-" key twice to zoom out the content.)

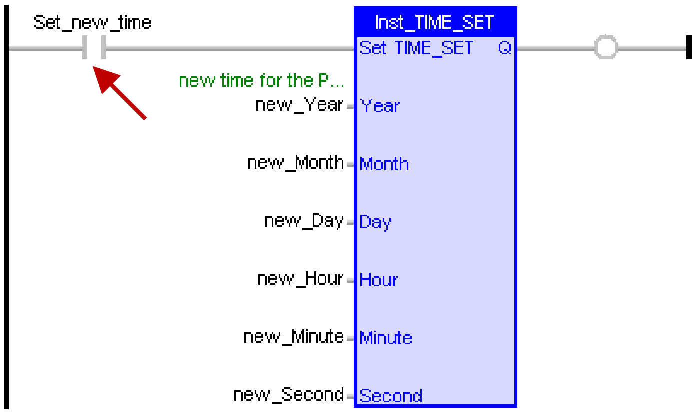
A. Program Editor Area (LD):

This area allows to edit or display this LD program, you can click the object button (on the left of Program Area) to add a program, and then drag-and-drop variables (in the Variables Area) onto the function block one-by-one.

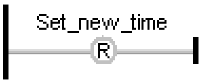
R1: Get current time of the PAC



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

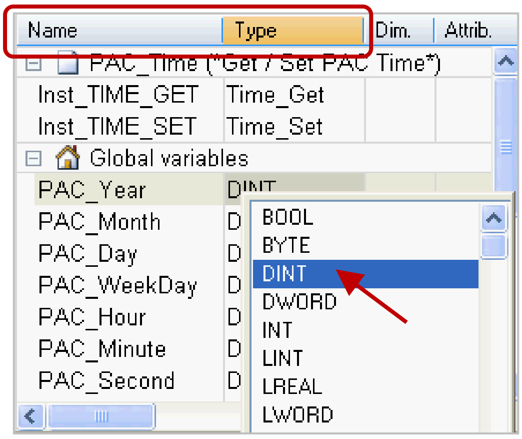


R3: Reset it to "False"

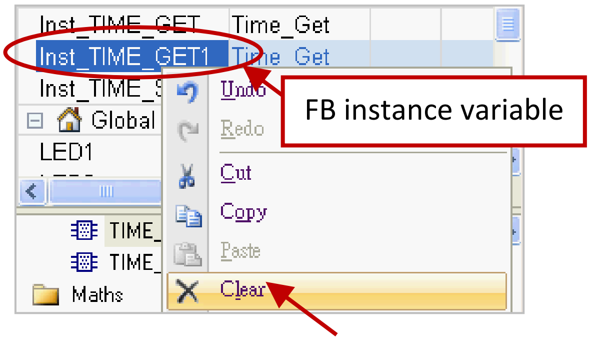


B. Variables Area:

This area shows the function blocks and variables that used in this program. Mouse double-click the "Name" or "Type" item to modify its name or data type, then click "Enter" to complete the setting. (Refer the Section 2.3.1 for the details about variable declaration.)

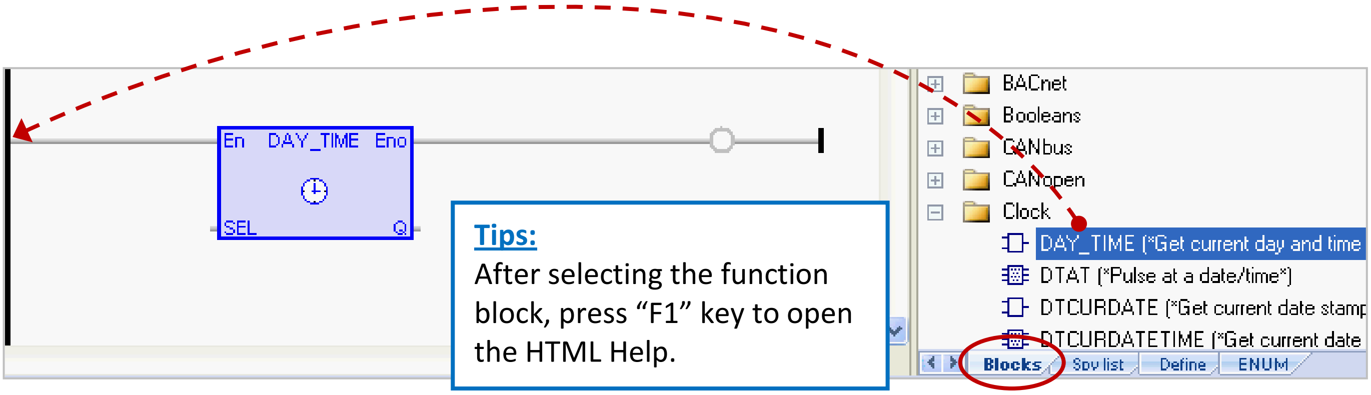


Note: For the function block can work correctly, the "Inst_xxx.." FB instance variable will be automatically added when using one function block. For the safety reasons, this FB instance variable will not be automatically deleted even if the function block has been removed in the editor area. So, users can right-click the unwanted variable and select "Clear" to manually remove it.



C. Function Blocks Area:

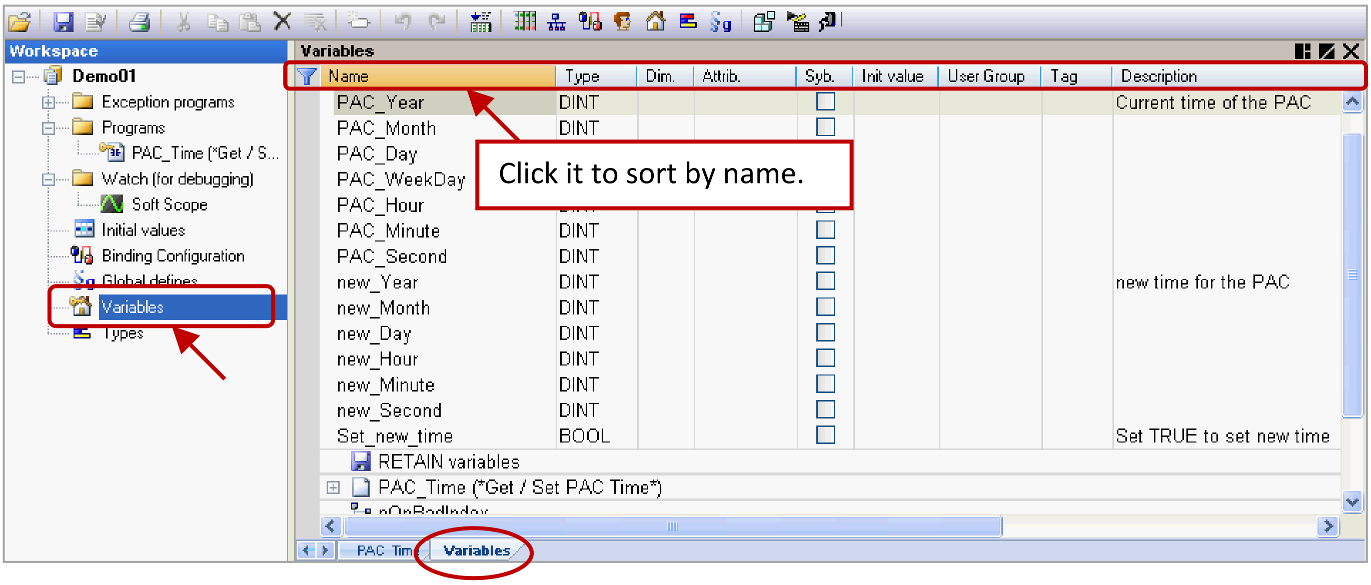
In the "Blocks" tab, it provides many types of the function block for users to drag and drop them to the editor area.


En DAY_TIME Eno
SEL Q
BACnet
Booleans
CANbus
CANopen
Clock
DAY_TIME (*Get current day and time
DTAT (*Pulse at a date/time*)
DTCURDATE (*Get current date stamp
DTCURDATETIME (*Get current date
Blocks
Spy list
Define
ENUM

Tips:
After selecting the function block, press "F1" key to open the HTML Help.

2.2.2 Demo01 - Variables

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



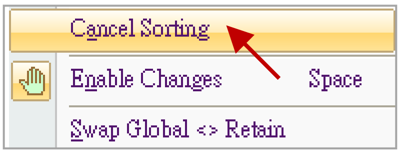
Name	Type	Dim.	Attrib.	Syb.	Init value	User Group	Tag	Description
PAC_Year	DINT			☐				Current time of the PAC
PAC_Month	DINT			☐				
PAC_Day								
PAC_WeekDay								
PAC_Hour								
PAC_Minute	DINT			☐				
PAC_Second	DINT			☐				
new_Year	DINT			☐				new time for the PAC
new_Month	DINT			☐				
new_Day	DINT			☐				
new_Hour	DINT			☐				
new_Minute	DINT			☐				
new_Second	DINT			☐				
Set_new_time	BOOL			☐				Set TRUE to set new time

RETAIN variables

PAC_Time (*Get / Set PAC Time*)

nOnBadIndex

Tips: In the Variables window, users can click any title field (e.g., “Name”) for sorting. If you want to go back to the original sort order, mouse right-click anywhere and select the “Cancel Sorting” option.



Field description: (Press the “F1” key to look up the details)
Mouse double-click any field item 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 the Appendix A for the value range)

Dim.: To specify the range of an array.
(E.g., enter “10”, means the use of the Counter [**0**] to [**9**]).

Attrib.: Double-click this field item to set it to “Read Olny” that means users can only read this variable but cannot modify it.

Syb.: If checked, the variable name will also be downloaded into 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.

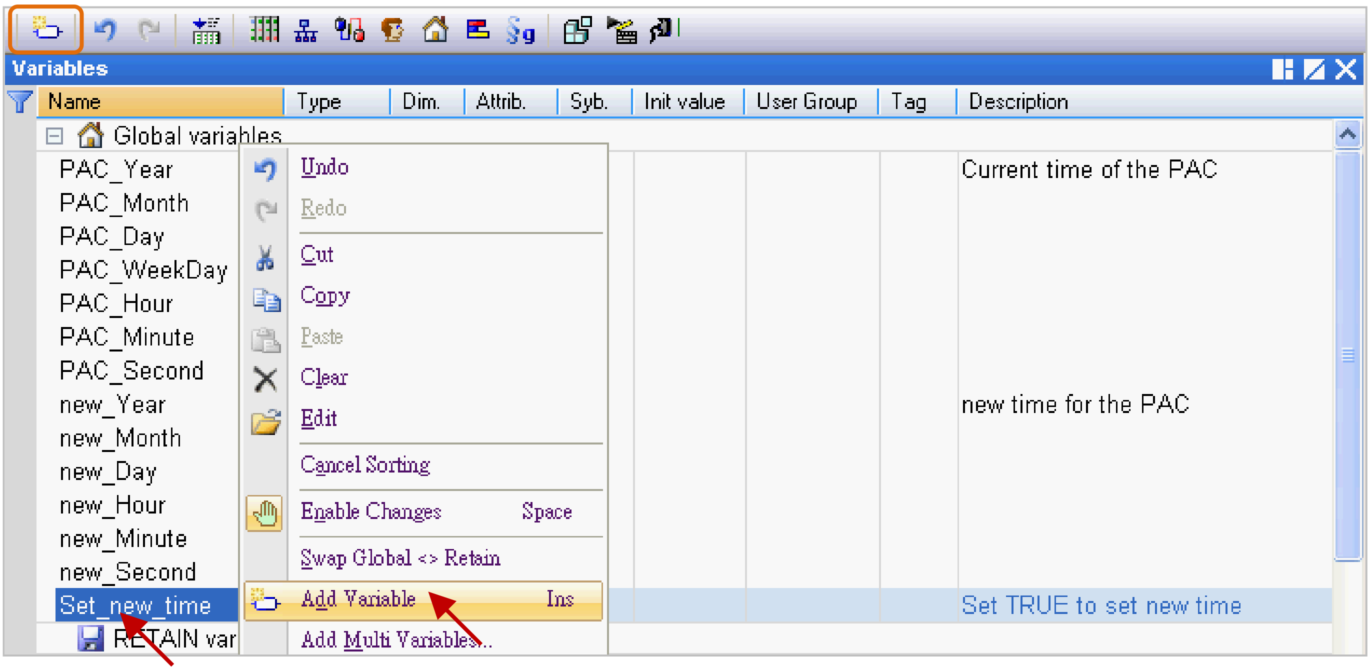
2.3 Give it a Try

As mentioned before, we have described the LD program (Section 2.2.1) and variables (Section 2.2.2) in the "Demo01" project. The following sections will show you how to declare variables and add an LD program with the blinking function in this project.

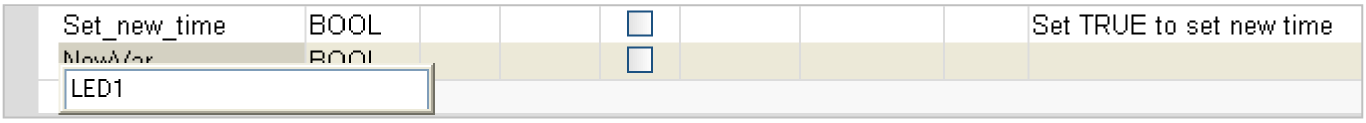
2.3.1 Declaring the Win-GRAF Project Variables

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

1. In the "Variables" window, mouse right-click any item in the "Name" field and select "Add Variable" (or press the "Ins" key or click the tool button) to add a variable.



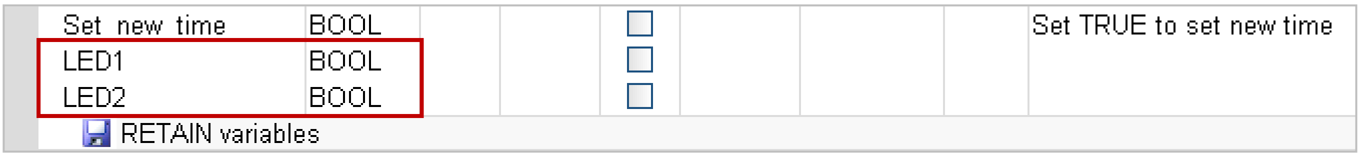
2. Double-click the new "NewVar" item and change its name to "LED1", then click "Enter" to finish the setting. In this case, the data type is "BOOL".



Set_new_time	BOOL		☐				Set TRUE to set new time
NewVar	BOOL		☐				
LED1							

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

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

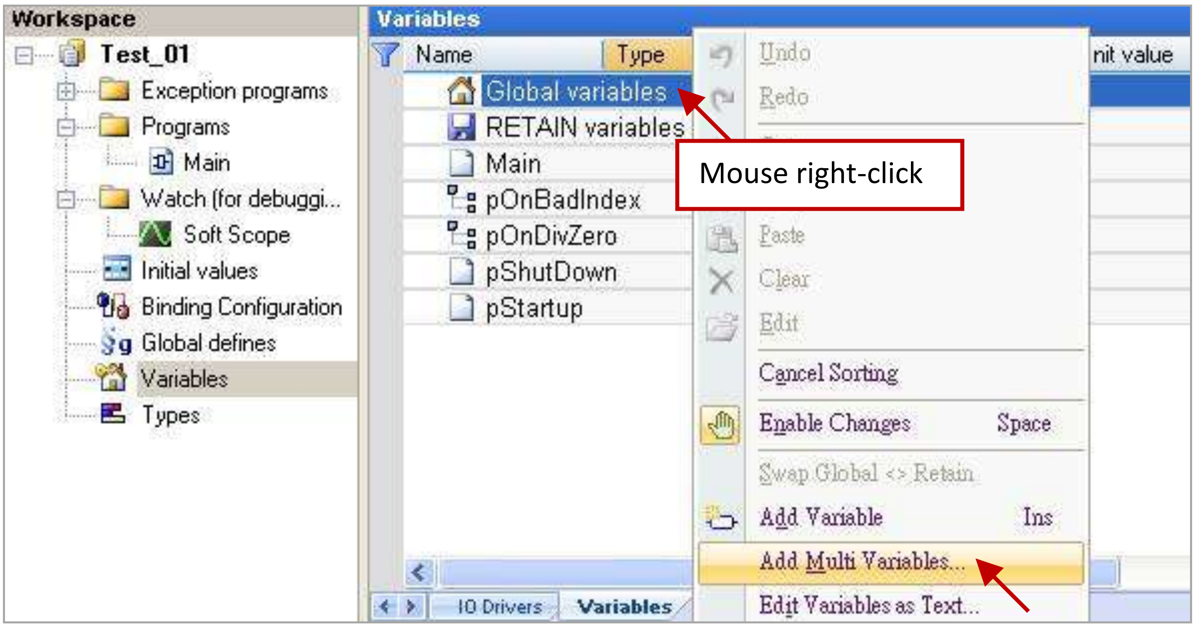


Set_new_time	BOOL		☐				Set TRUE to set new time
LED1	BOOL		☐				
LED2	BOOL		☐				
RETAIN variables							

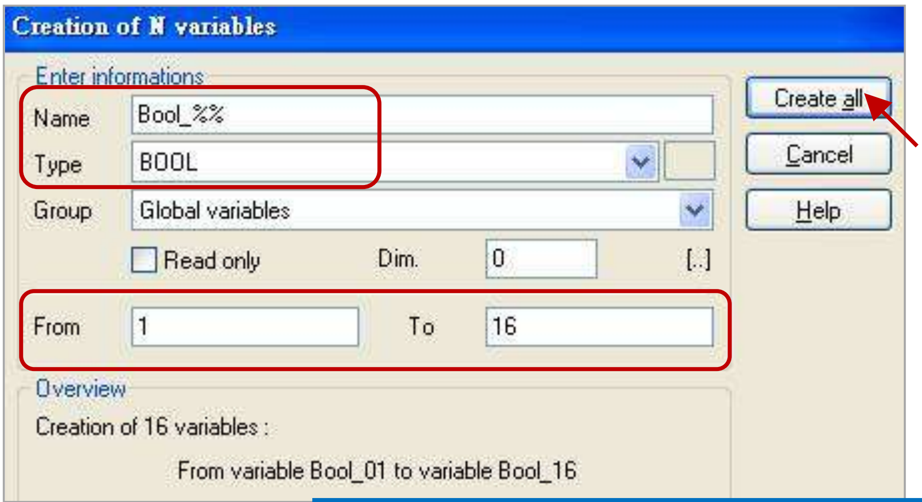
Tips: To set up multiple ordinal variables, enter the name "LED" (as the step2) and then press "Ctrl+C" and "Ctrl+V" twice to create "LED**1**" and "LED**2**" (auto sequential numbering), finally, delete the first variable (i.e., "LED").

Tip #2:

1. If you need to add multiple variables (e.g., “Boo_01 to Boo_16”), simply right-click the “Global variables” and select the “Add Multi Variables”.



2. Follow the settings like the figure below (Name: “Bool_%%”; Type: “BOOL”; From: 1; To: 16) to create Boolean variables (i.e., “Bool_01” to “Boo_16”) and then click “Create all” button to complete the settings.



Name: Bool_%%

Type: BOOL

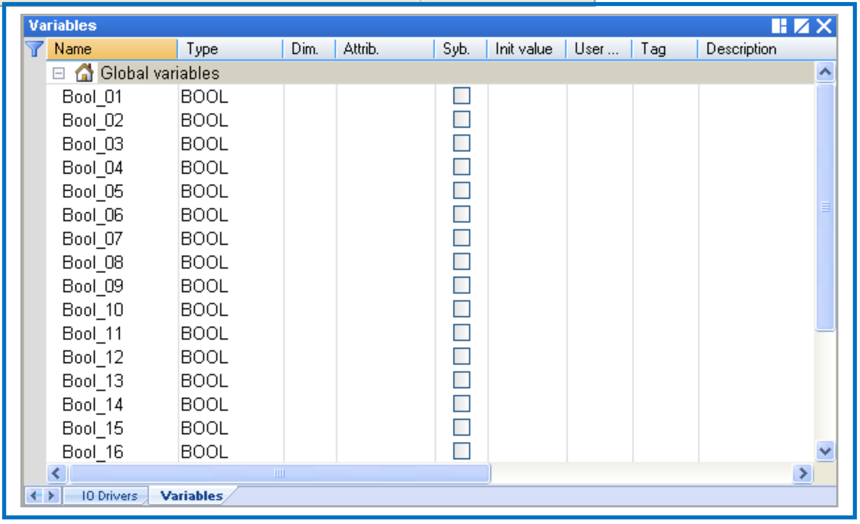
Group: Global variables

□ Read only Dim. 0 [..]

From: 1 To: 16

Creation of 16 variables :

From variable Bool_01 to variable Bool_16

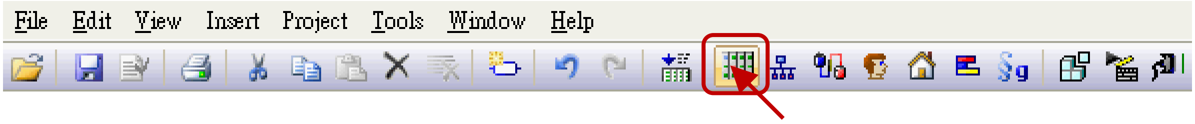


Name	Type	Dim.	Attrib.	Syb.	Init value	User ...	Tag	Description
Bool_01	BOOL			□				
Bool_02	BOOL			□				
Bool_03	BOOL			□				
Bool_04	BOOL			□				
Bool_05	BOOL			□				
Bool_06	BOOL			□				
Bool_07	BOOL			□				
Bool_08	BOOL			□				
Bool_09	BOOL			□				
Bool_10	BOOL			□				
Bool_11	BOOL			□				
Bool_12	BOOL			□				
Bool_13	BOOL			□				
Bool_14	BOOL			□				
Bool_15	BOOL			□				
Bool_16	BOOL			□				

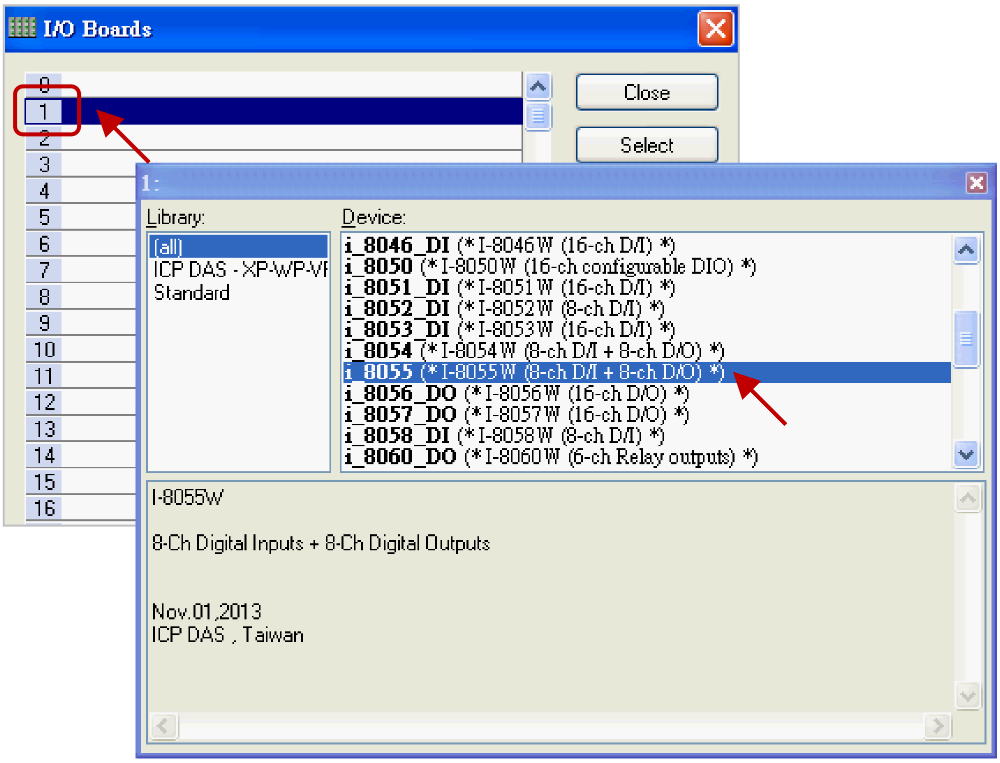
2.3.2 Declaring the I/O Variables

In this example, the I-8055W module that used to show the blinking feature must be plugged in the PAC's slot1. So, we need to add an I/O link to correspond to the real I/O module.

1. Click the "Open I/Os" tool button to add an I/O link.



2. Mouse double-click on "Slot 1" and then double-click the "i_8055" to select this I/O board.



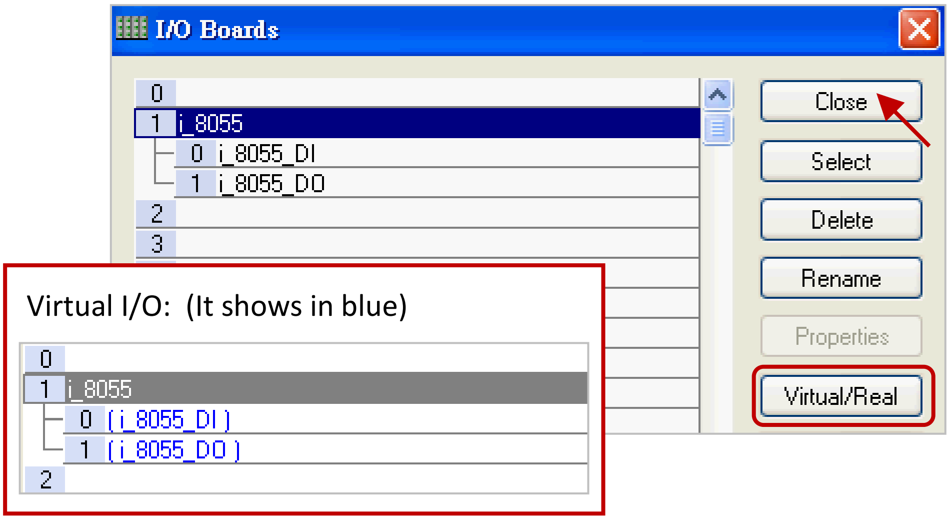
i_8046_DI (*I-8046W (16-ch D/I) *)
i_8050 (*I-8050W (16-ch configurable DIO) *)
i_8051_DI (*I-8051W (16-ch D/I) *)
i_8052_DI (*I-8052W (8-ch D/I) *)
i_8053_DI (*I-8053W (16-ch D/I) *)
i_8054 (*I-8054W (8-ch D/I + 8-ch D/O) *)
i_8055 (*I-8055W (8-ch D/I + 8-ch D/O) *)
i_8056_DO (*I-8056W (16-ch D/O) *)
i_8057_DO (*I-8057W (16-ch D/O) *)
i_8058_DI (*I-8058W (8-ch D/I) *)
i_8060_DO (*I-8060W (6-ch Relay outputs) *)

I-8055W

8-Ch Digital Inputs + 8-Ch Digital Outputs

Nov.01,2013
ICP DAS , Taiwan

3. Click the "Close" button to exit the "I/O Boards" window.
<span style="color:red">Note:</span> Click the "Virtual/Real" button to change to the Virtual I/O (for testing) or the Real I/O. (The Real I/O is used in this example).

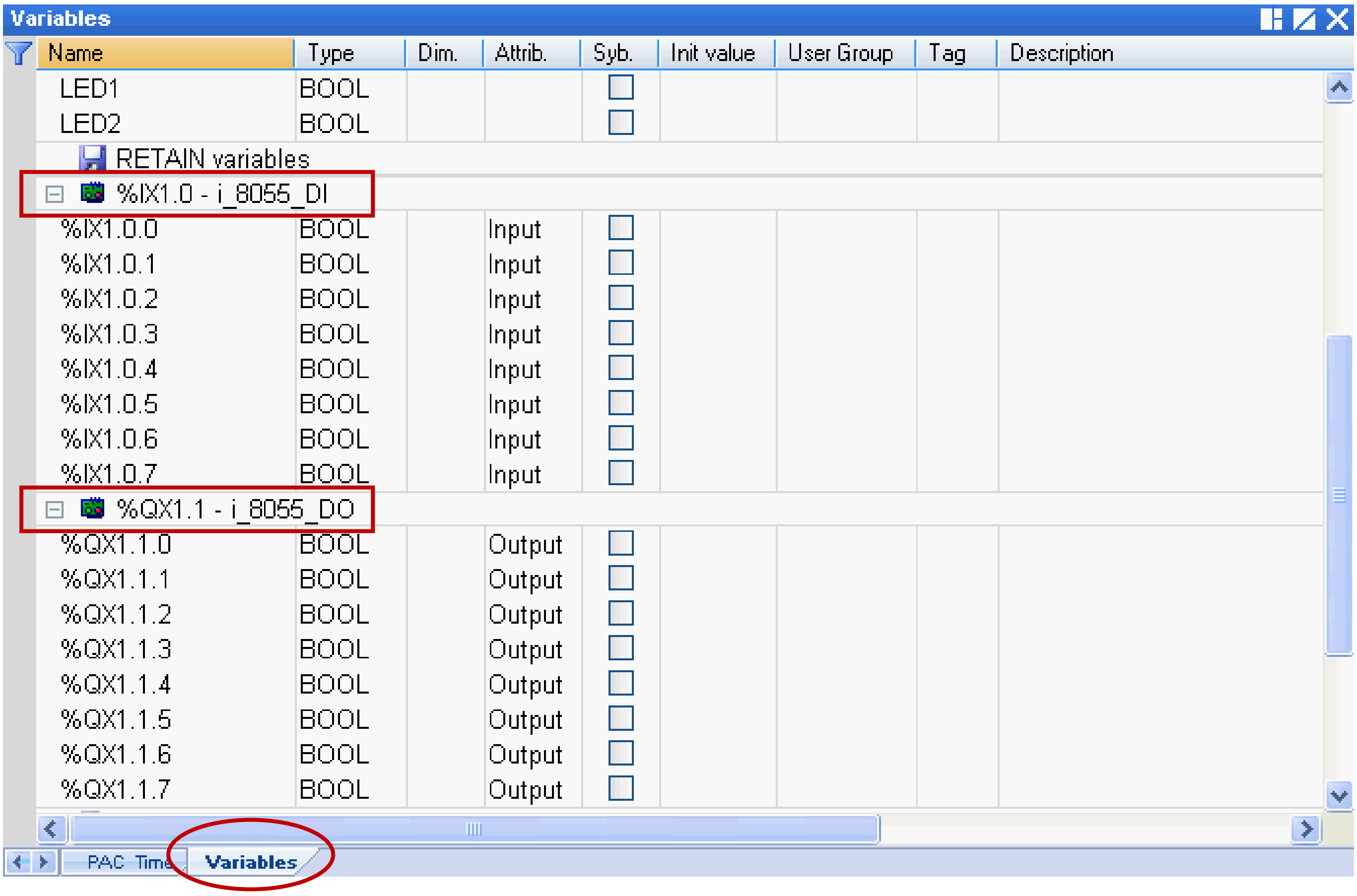


0
1 i_8055
0 i_8055_DI
1 i_8055_DO
2
3

Virtual I/O: (It shows in blue)

0
1 i_8055
0 (i_8055_DI)
1 (i_8055_DO)
2

After linking the “i_8055” I/O board, it will automatically add 8 input & output variables in the “Variables” window.



Variables

Name	Type	Dim.	Attrib.	Syb.	Init value	User Group	Tag	Description
LED1	BOOL			□				
LED2	BOOL			□				
RETAIN variables								
%IX1.0 - i_8055_DI								
%IX1.0.0	BOOL		Input	□				
%IX1.0.1	BOOL		Input	□				
%IX1.0.2	BOOL		Input	□				
%IX1.0.3	BOOL		Input	□				
%IX1.0.4	BOOL		Input	□				
%IX1.0.5	BOOL		Input	□				
%IX1.0.6	BOOL		Input	□				
%IX1.0.7	BOOL		Input	□				
%QX1.1 - i_8055_DO								
%QX1.1.0	BOOL		Output	□				
%QX1.1.1	BOOL		Output	□				
%QX1.1.2	BOOL		Output	□				
%QX1.1.3	BOOL		Output	□				
%QX1.1.4	BOOL		Output	□				
%QX1.1.5	BOOL		Output	□				
%QX1.1.6	BOOL		Output	□				
%QX1.1.7	BOOL		Output	□				

%IX1.0 – i_8055_DI

“I” means **“Input”**

“X” means **“Boolean”**

“1” means **“Slot 1”**

%QX1.1 – i_8055_DO

“Q” means **“Output”**

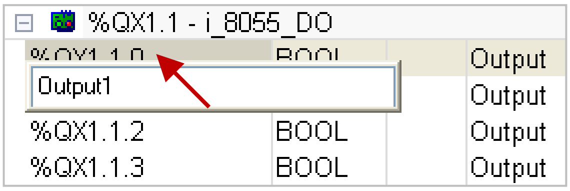
“X” means **“Boolean”**

“1” means **“Slot 1”**

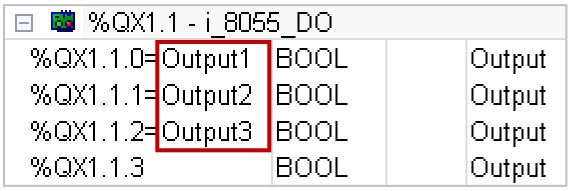
%ID or %QD

“D” means **“Integer/Real”**

There are three output variables are used in this example, and you can modify the name for easy use. Mouse double-click the item and fill in a name, then press “Enter” key to finish the setting.



%QX1.1 - i_8055_DO		
%QX1.1.0	BOOL	Output
Output1		Output
%QX1.1.2	BOOL	Output
%QX1.1.3	BOOL	Output

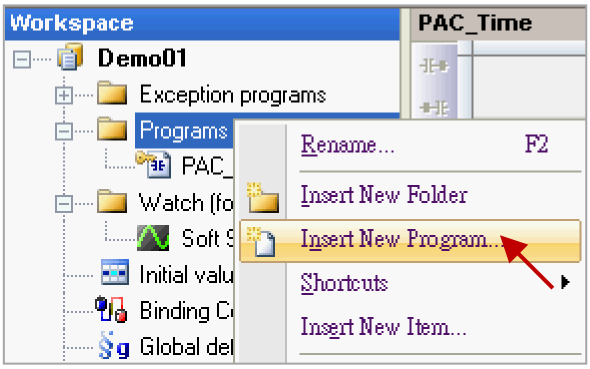


%QX1.1 - i_8055_DO		
%QX1.1.0=Output1	BOOL	Output
%QX1.1.1=Output2	BOOL	Output
%QX1.1.2=Output3	BOOL	Output
%QX1.1.3	BOOL	Output

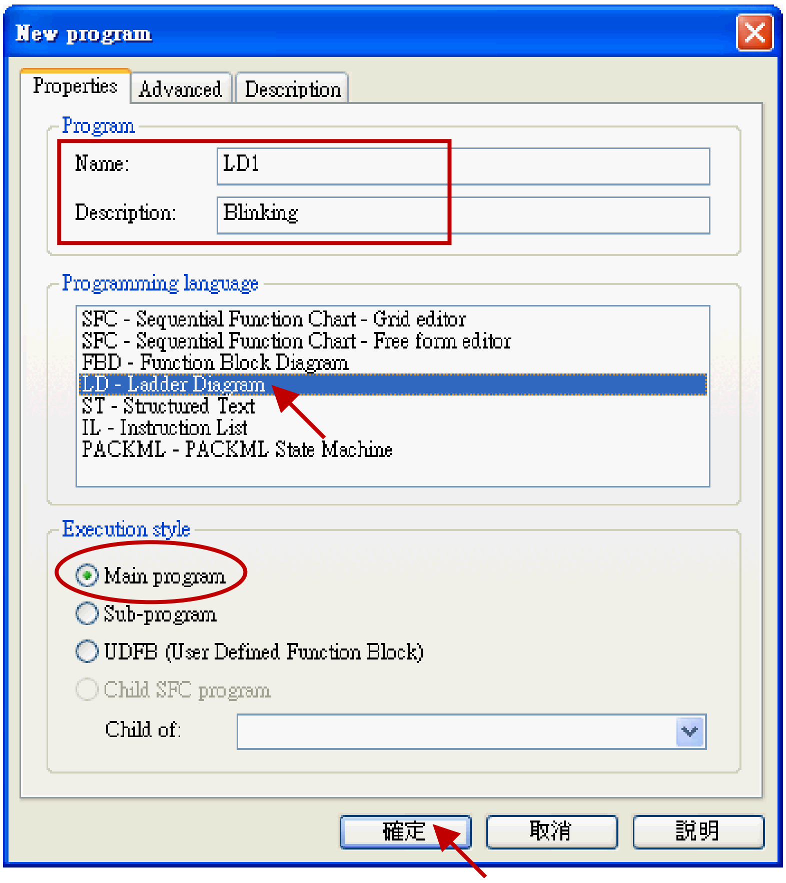
2.3.3 Creating an LD Program

In the “Demo01” project, we want to create a “LD1” program to show the blinking. To begin, follow these steps:

1. In the workspace, mouse 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.

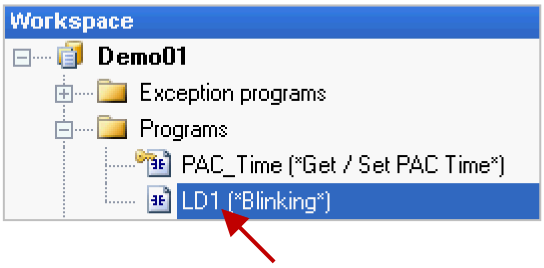


Name: LD1

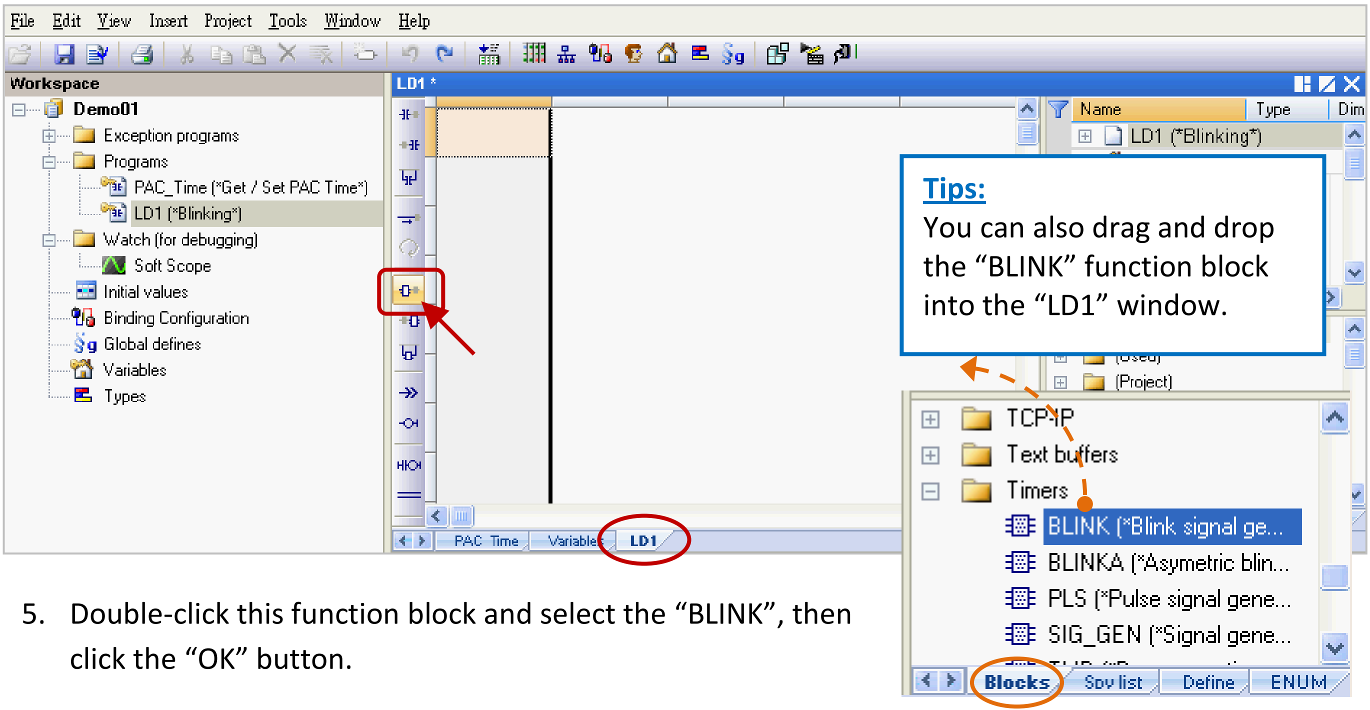
Description: Blinking

Programming language
SFC - Sequential Function Chart - Grid editor
SFC - Sequential Function Chart - Free form editor
FBD - Function Block Diagram
LD - Ladder Diagram
ST - Structured Text
IL - Instruction List
PACKML - PACKML State Machine
Execution style
● Main program
○ Sub-program
○ UDFB (User Defined Function Block)
○ Child SFC program
Child of:
確定
取消
說明

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

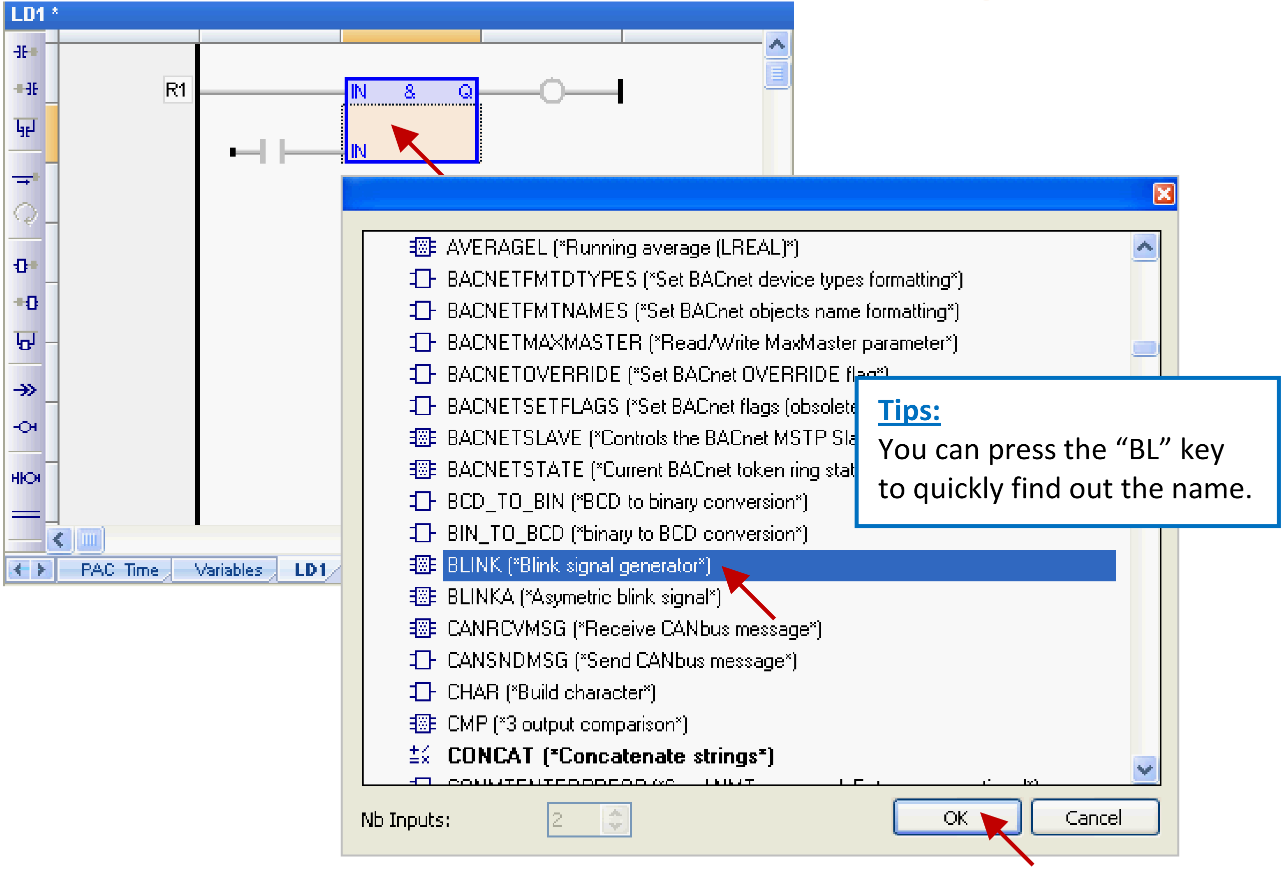


4. Click the “Insert FB..” button on the left of the “LD1” window to add a function block.



Tips:
You can also drag and drop the “BLINK” function block into the “LD1” window.

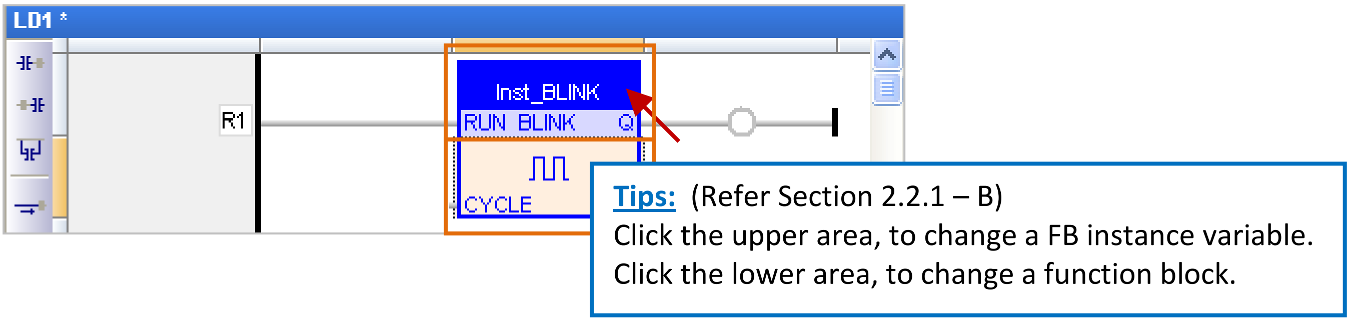
5. Double-click this function block and select the “BLINK”, then click the “OK” button.



AVERAGEL (*Running average (LREAL)*)
BACNETFMTDTYPES (*Set BACnet device types formatting*)
BACNETFMTNAMES (*Set BACnet objects name formatting*)
BACNETMAXMASTER (*Read/Write MaxMaster parameter*)
BACNETOVERRIDE (*Set BACnet OVERRIDE flag*)
BACNETSETFLAGS (*Set BACnet flags (obsolete
BACNETSLAVE (*Controls the BACnet MSTP Sla
BACNETSTATE (*Current BACnet token ring stat
BCD_TO_BIN (*BCD to binary conversion*)
BIN_TO_BCD (*binary to BCD conversion*)
BLINK (*Blink signal generator*)
BLINKA (*Asymetric blink signal*)
CANRCVMSG (*Receive CANbus message*)
CANSNDMSG (*Send CANbus message*)
CHAR (*Build character*)
CMP (*3 output comparison*)
CONCAT (*Concatenate strings*)

Nb Inputs: 2

Tips:
You can press the “BL” key to quickly find out the name.

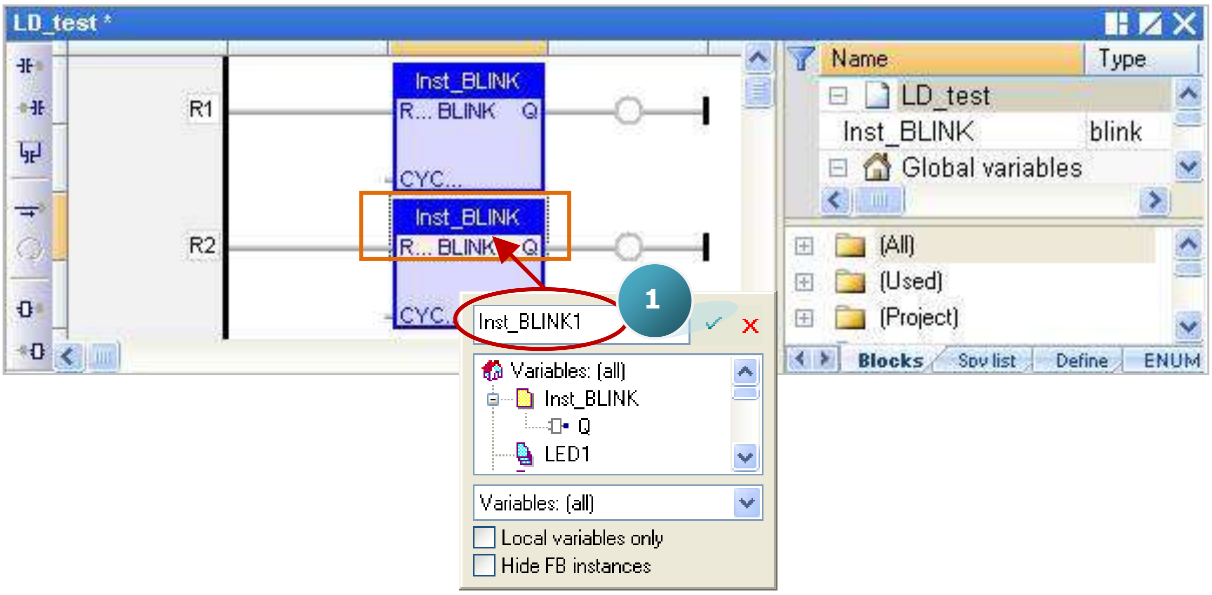


Tips: (Refer Section 2.2.1 – B)
Click the upper area, to change a FB instance variable.
Click the lower area, to change a function block.

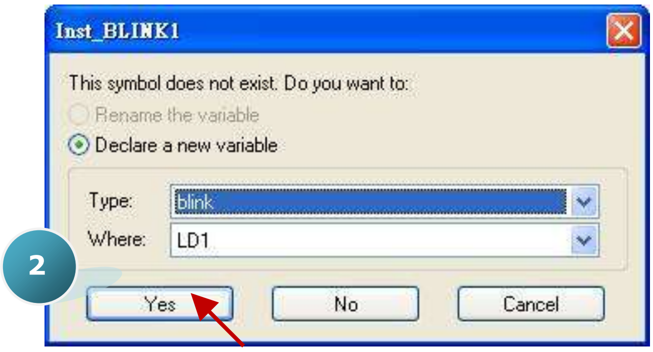
Important Notice:

When programming, users may copy & paste an existing function block to create a new one. But, this way will cause a function exception due to the same function instances. Therefore, users must create a new name for the function instance.

1. Mouse double-click the function block and enter a new name (e.g., “Inst_BLINK1”), then click the ✓ button to complete the setting.



2. In the “Inst_BLINK1” window, click “Yes” to create this function instance.



Inst_BLINK1

This symbol does not exist. Do you want to:

○ Rename the variable

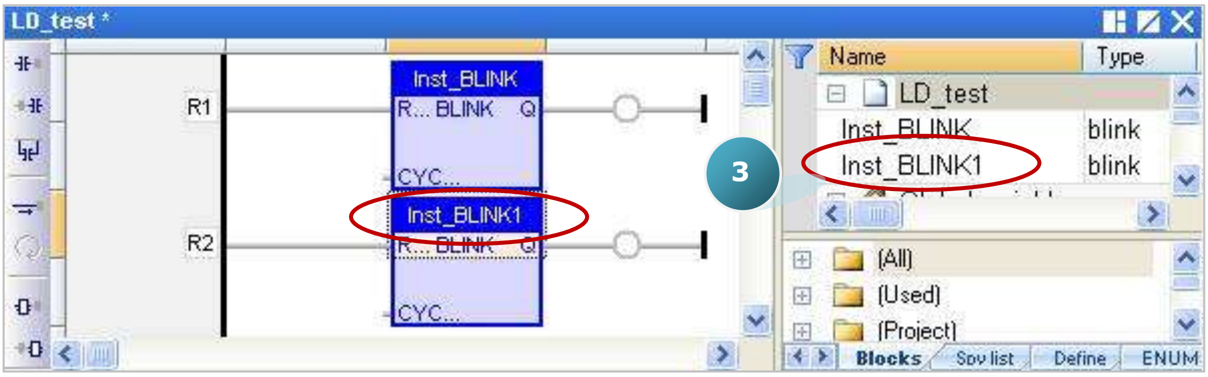
● Declare a new variable

Type: blink

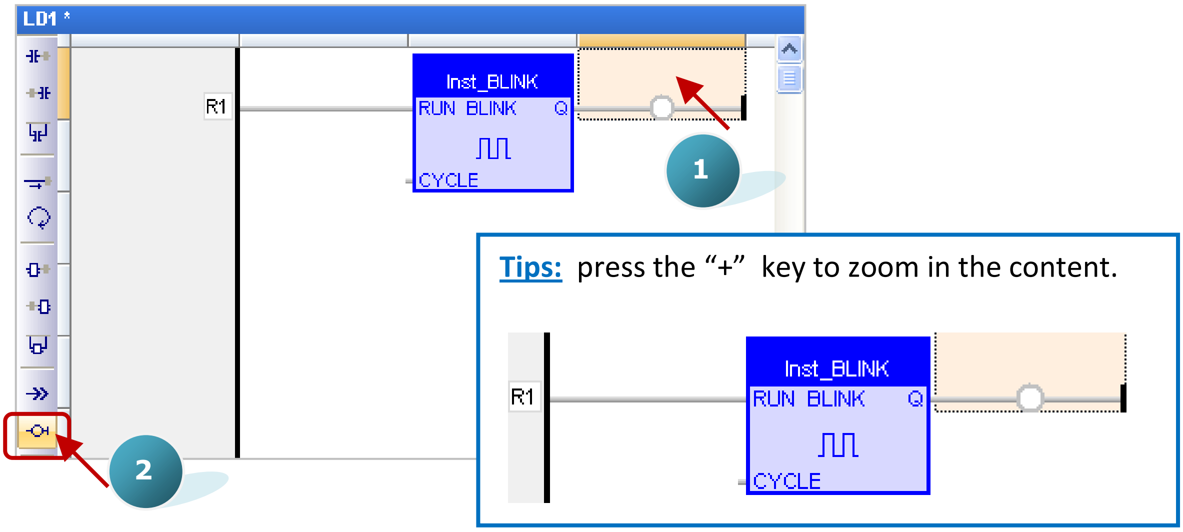
Where: LD1

Tips:
You can also use the same way to double-click on the right-side of the “Coil” to create or assign a variable. (See Step7)

3. Now, there is an “Inst_BLINK1” function instance added in the Variables Area.

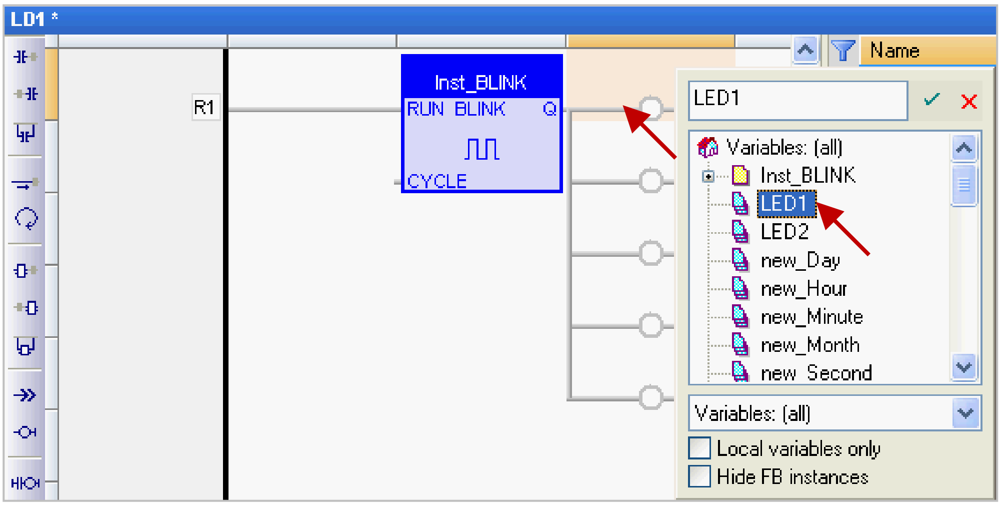


6. Click the “Coil” on the right of the “BLINK” function block, and continuously click the “Insert Coil” button to add four “Coil”.

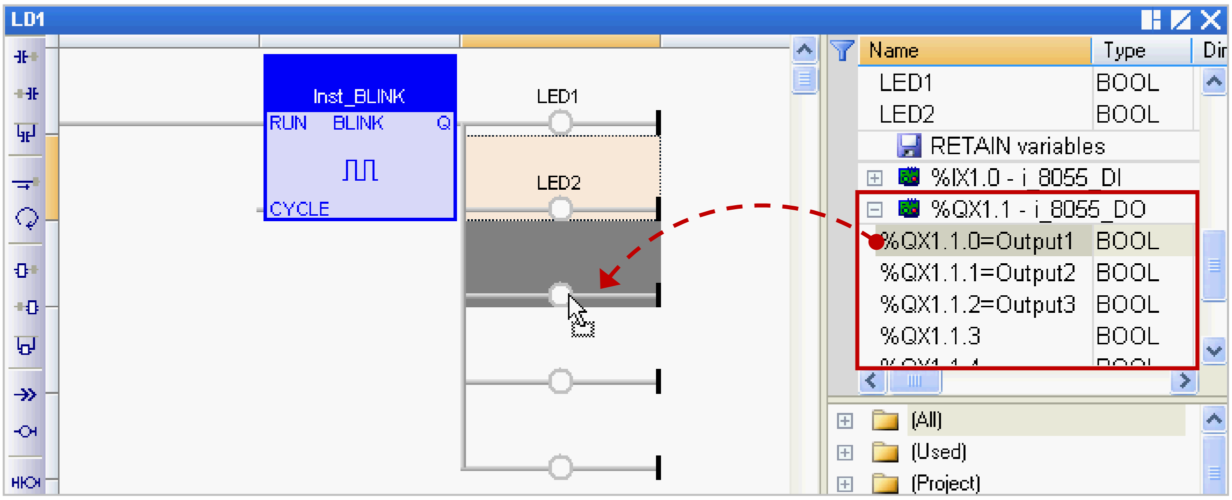


Tips: press the “+” key to zoom in the content.

7. Mouse double-click the first “Coil” and double-click “LED1” to assign it. Follow the same way to assign the “LED2” variable to the second “Coil”.

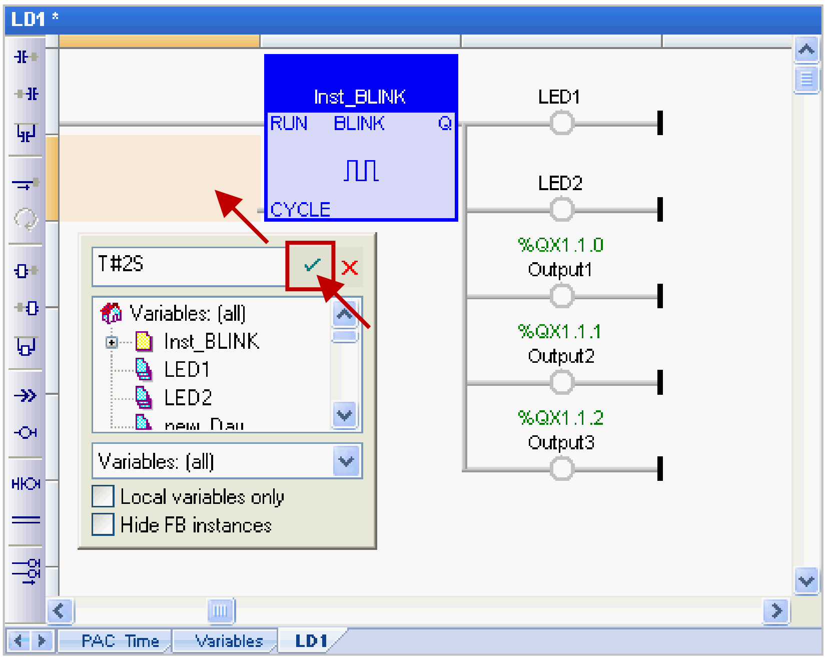


8. As the screenshot below, mouse drag-and-drop the “Output1”, “Output2” and “Output3” variables to the 3th, 4th and 5th “Coil”.

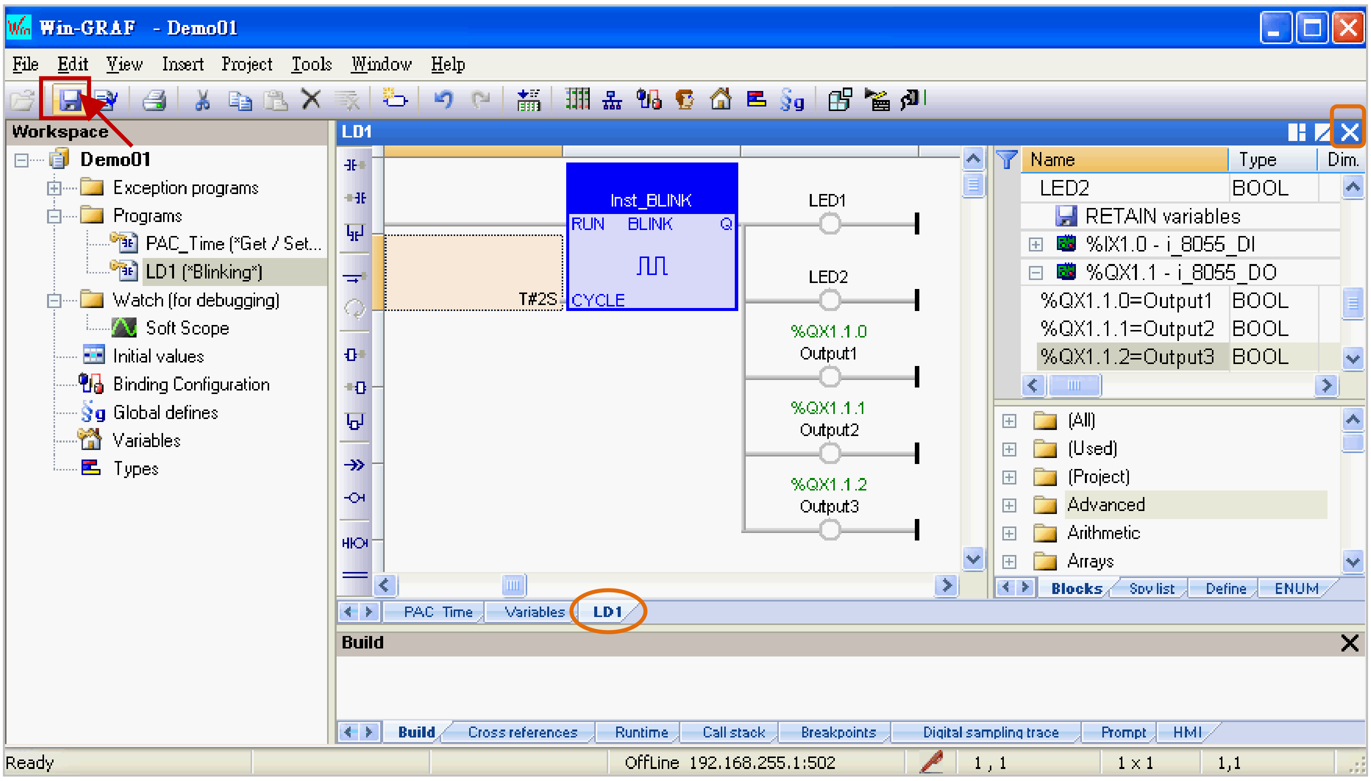


Name	Type
LED1	BOOL
LED2	BOOL
RETAIN variables	
%IX1.0 - i_8055_DI	
%QX1.1 - i_8055_DO	
%QX1.1.0=Output1	BOOL
%QX1.1.1=Output2	BOOL
%QX1.1.2=Output3	BOOL
%QX1.1.3	BOOL

9. Mouse double-click on the left of the "CYCLE" and enter "T#2S" (to blink every two seconds), and then click ✓ to finish the setting.



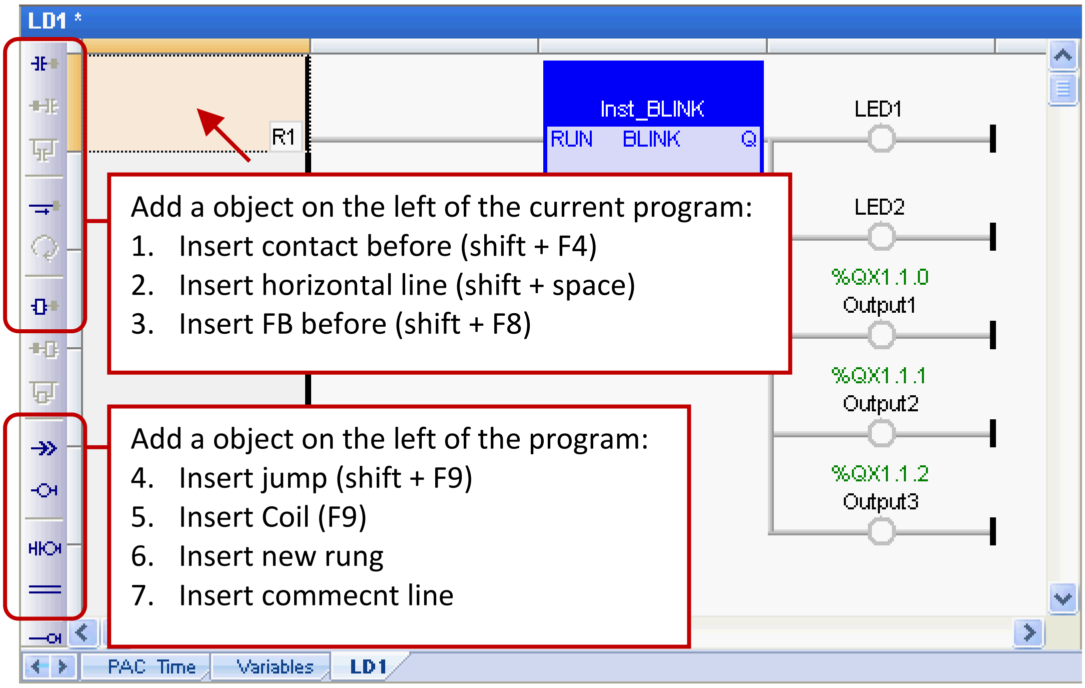
10. Finally, click the "Save" button to save the "LD1" program.



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

If you want to add a program to the first line after completing the programming, follow these steps:

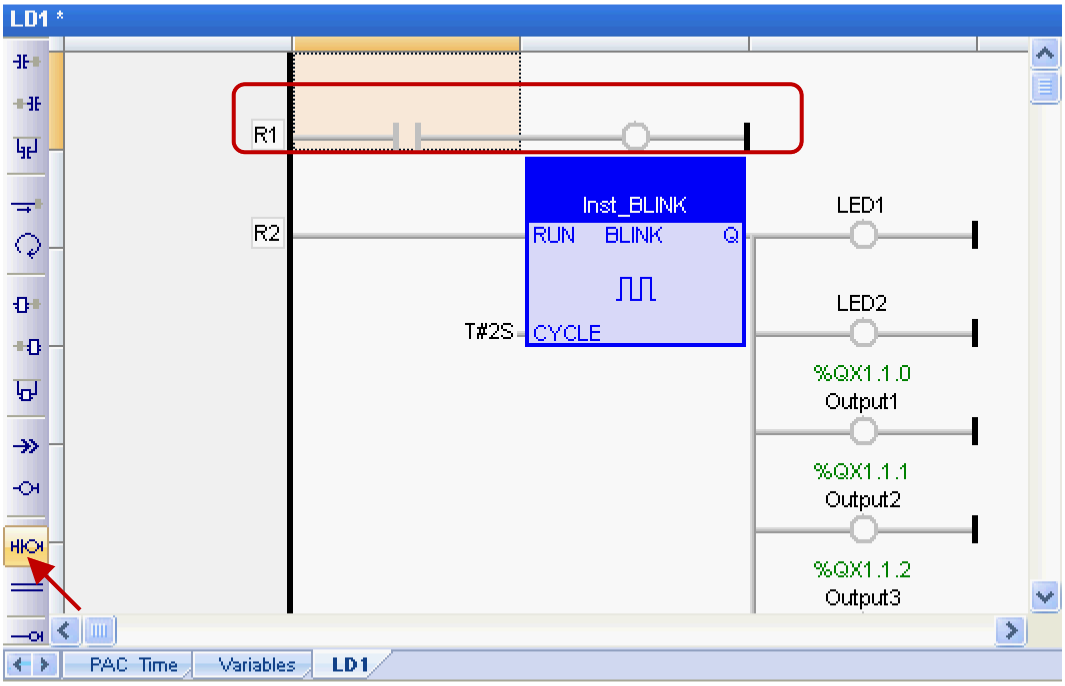
1. Click the upper-left corner of the Program Editor Area, and you will see the selectable items on the Object bar. Select an object (4 to 7, as the screenshot below) to add a program to the first line. Note: You can also select an object (1 to 3) to add it on the left of the current program.



Add a object on the left of the current program:
1. Insert contact before (shift + F4)
2. Insert horizontal line (shift + space)
3. Insert FB before (shift + F8)

Add a object on the left of the program:
4. Insert jump (shift + F9)
5. Insert Coil (F9)
6. Insert new rung
7. Insert commecnt line

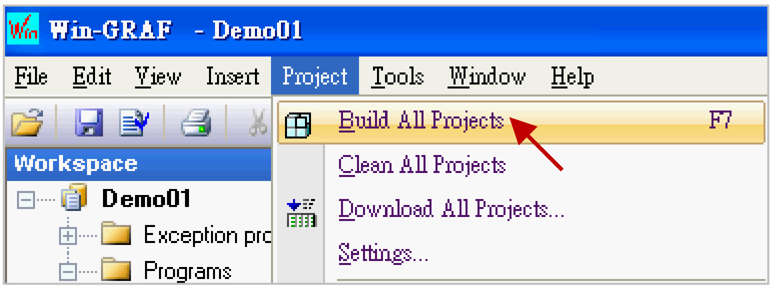
2. In this case, click “Insert new rung” to add it to the first line.



2.3.4 Compiling the Program

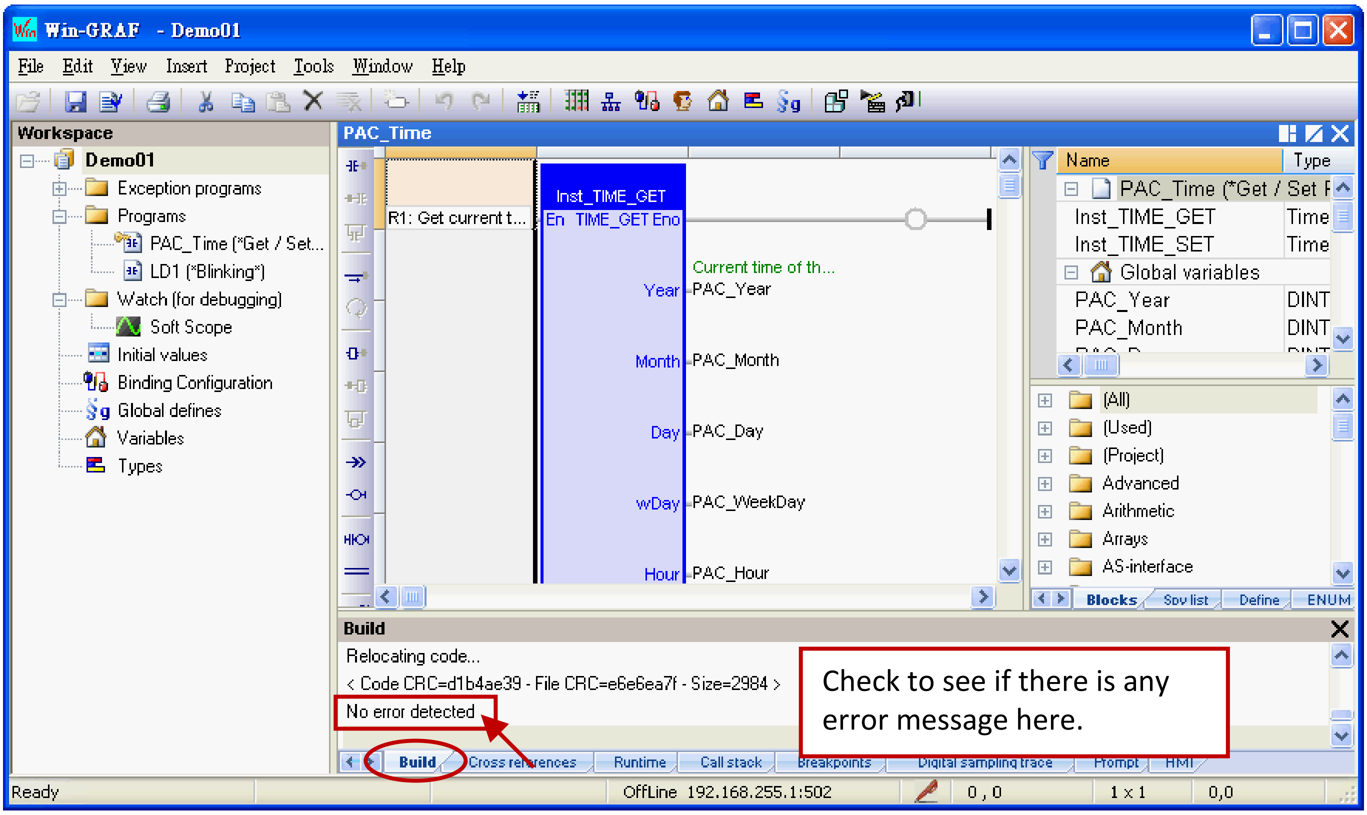
In the previous section, we have added and saved the LD program. For the Win-GRAF project can function properly in the PAC, we need to compile the programs. To begin, follow these steps:

1. On the menu bar, click "Project > Build All Projects" to compile all programs.



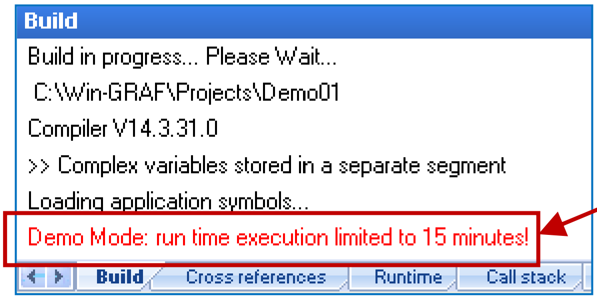
2. If a "No error detected" message is appear that means the project was successfully compiled.

Note: **If you modify and save the program after compiling it,** click the "Clean All Projects" to clean the previous results and then do the step1 again.



```
Relocating code...
< Code CRC=d1b4ae39 - File CRC=e6e6ea7f - Size=2984 >
No error detected
```

Check to see if there is any error message here.



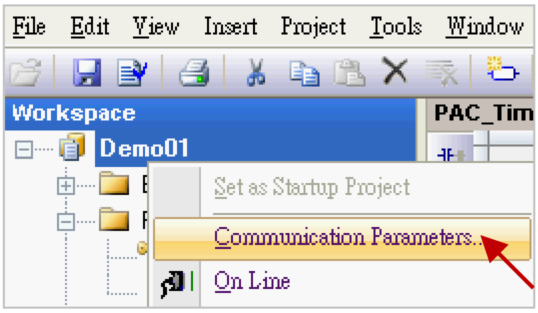
```
Build in progress... Please Wait...
 C:\Win-GRAF\Projects\Demo01
Compiler V14.3.31.0
>> Complex variables stored in a separate segment
Loading application symbols...
Demo Mode: run time execution limited to 15 minutes!
```

If the Win-GRAF is running in Demo Mode, this message means the Win-GRAF project can run for up to 15 minutes in the PAC.

2.3.5 Download the Program to PAC

Before downloading the program, you need to set up the communication parameters. (By now, it only supports the Ethernet TCP/IP).

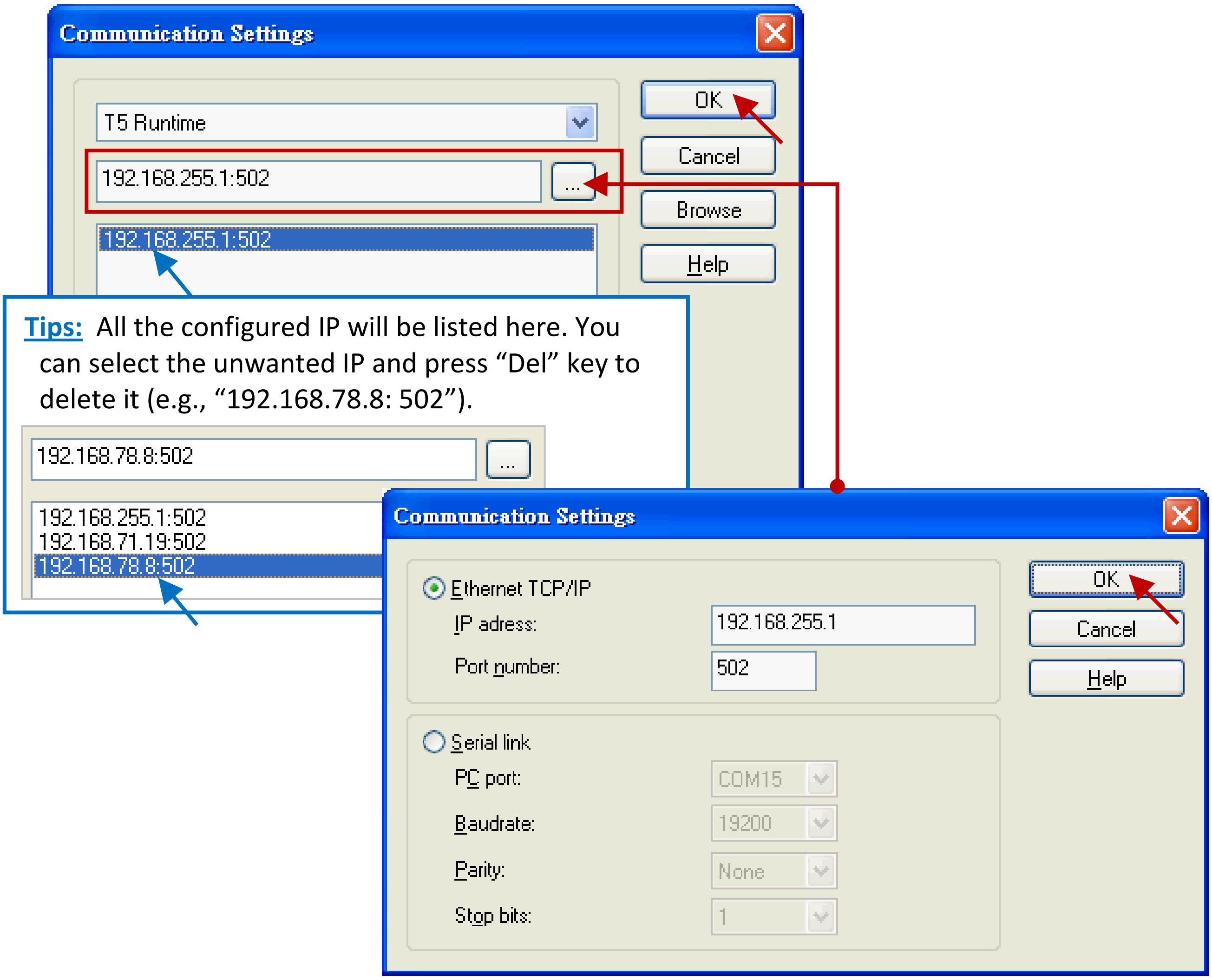
1. Mouse right-click the project name (i.e., "Demo01") and select “Communication Parameters…” to open the settings window.



2. Enter the “PAC IP:502” (e.g., “192.168.255.1:502”) to add an IP address and then click “OK”.

   It can also click the ... button to add/modify the IP address.

   (Note: the default PAC IP is “192.168.255.1” and the fixed port number of Win-GRAF PAC is “502”)



Communication Settings

T5 Runtime

192.168.255.1:502

192.168.255.1:502

Tips: All the configured IP will be listed here. You can select the unwanted IP and press “Del” key to delete it (e.g., “192.168.78.8: 502”).

192.168.78.8:502

192.168.255.1:502
192.168.71.19:502
192.168.78.8:502

Communication Settings

● Ethernet TCP/IP

IP adress: 192.168.255.1

Port number: 502

○ Serial link

PC port: COM15

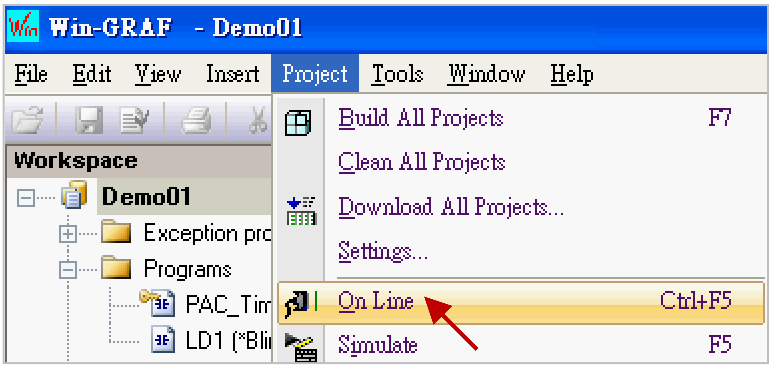
Baudrate: 19200

Parity: None

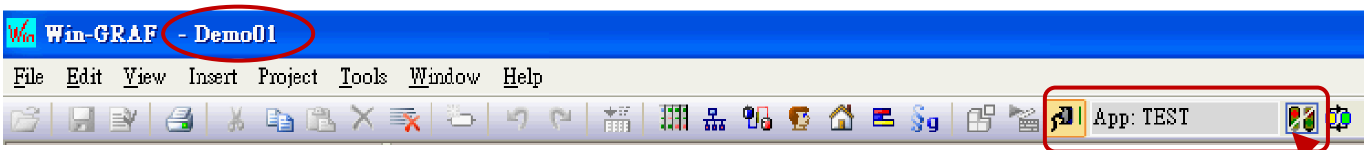
Stop bits: 1

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

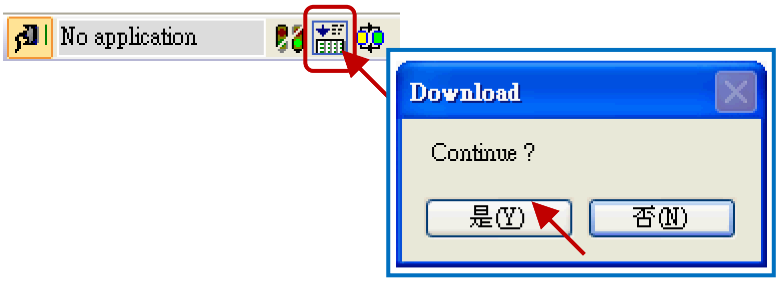
4. Click the menu bar “Project” and select “On Line”, or click the tool button to establish a connection.

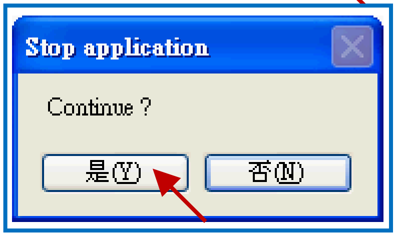


5. As the screenshot, if it shows “App: TEST”, different to the current project name (i.e., “Demo01”), that means there is a project (name: “TEST”) running in the PAC. Click “Stop application” tool button to stop the “TEST” project.

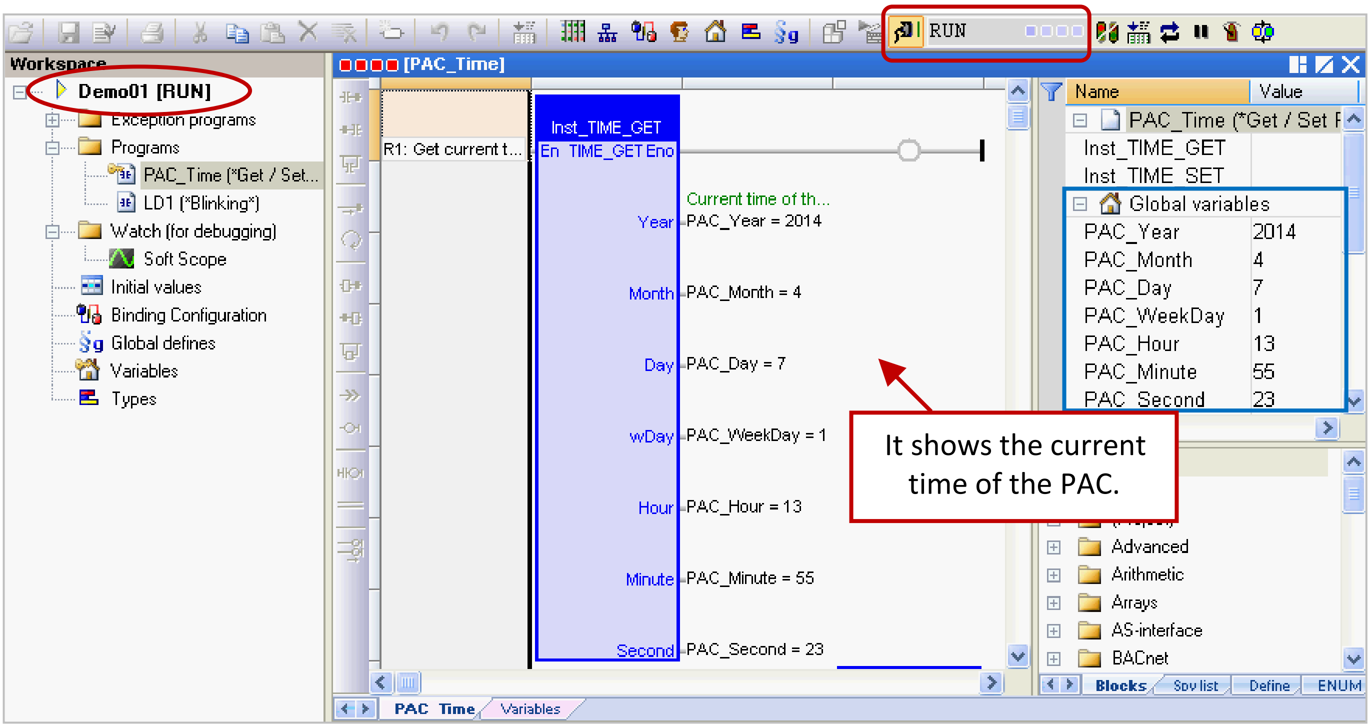


6. Click the “Download” tool button to download the “Demo01” project.





7. If it shows “RUN” that means the "Demo01" project is successfully running in the PAC.



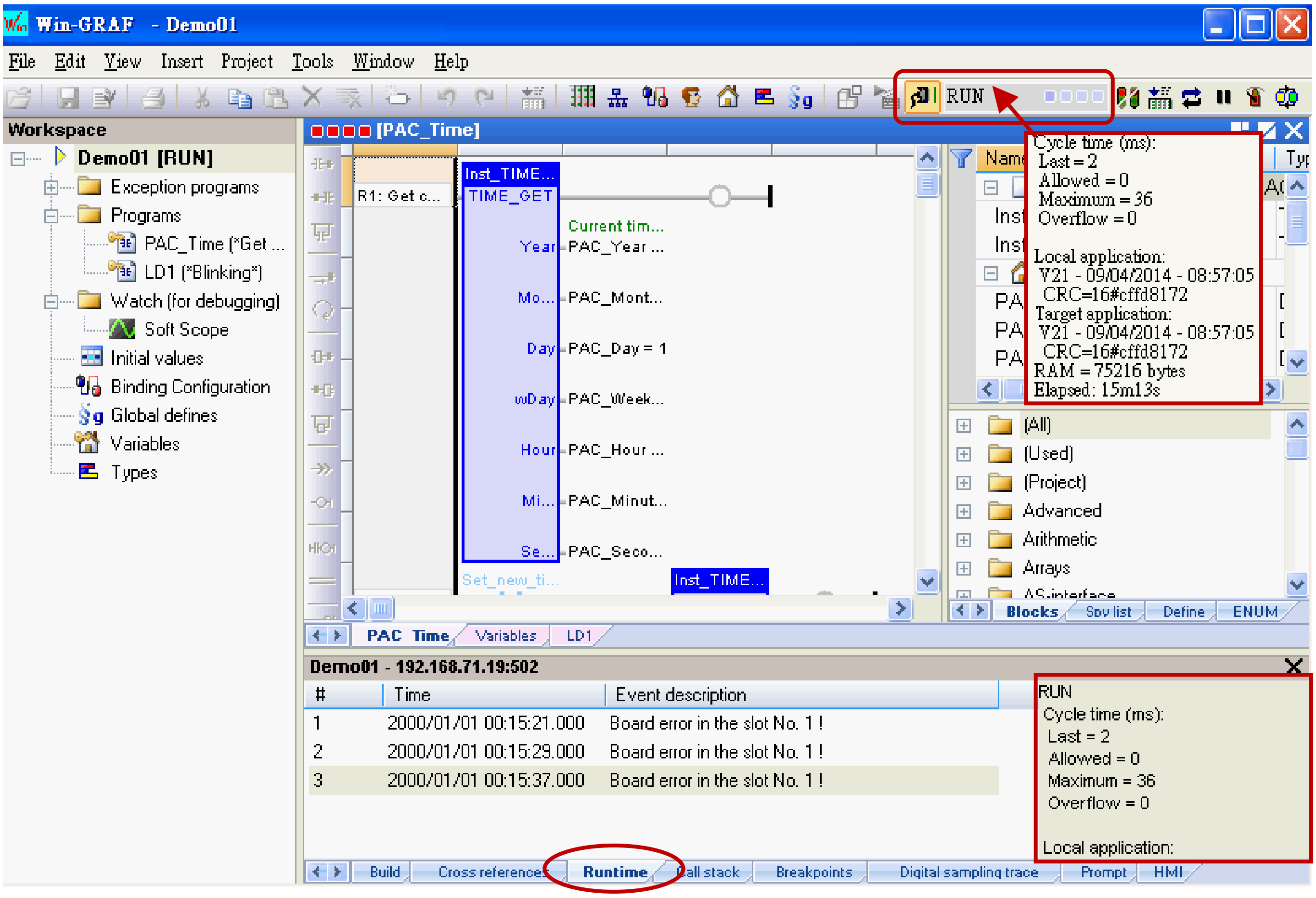
Name	Value
PAC_Time (*Get / Set F	
Inst_TIME_GET	
Inst_TIME_SET	
Global variables	
PAC_Year	2014
PAC_Month	4
PAC_Day	7
PAC_WeekDay	1
PAC_Hour	13
PAC_Minute	55
PAC_Second	23

It shows the current time of the PAC.

Note: If there is any error message show up during the download process, refer the Appendix B to get the solution.

Cycle time

When on-line with the PAC, move your mouse over the “RUN” position on the toolbar to view the current cycle time of the program in the PAC. You can also view the cycle time in the bottom-right corner of the message area.



Cycle time (ms):
Last = 2
Allowed = 0
Maximum = 36
Overflow = 0

Local application:
V21 - 09/04/2014 - 08:57:05
CRC=16#cffd8172
Target application:
V21 - 09/04/2014 - 08:57:05
CRC=16#cffd8172
RAM = 75216 bytes
Elapsed: 15m13s

Demo01 - 192.168.71.19:502

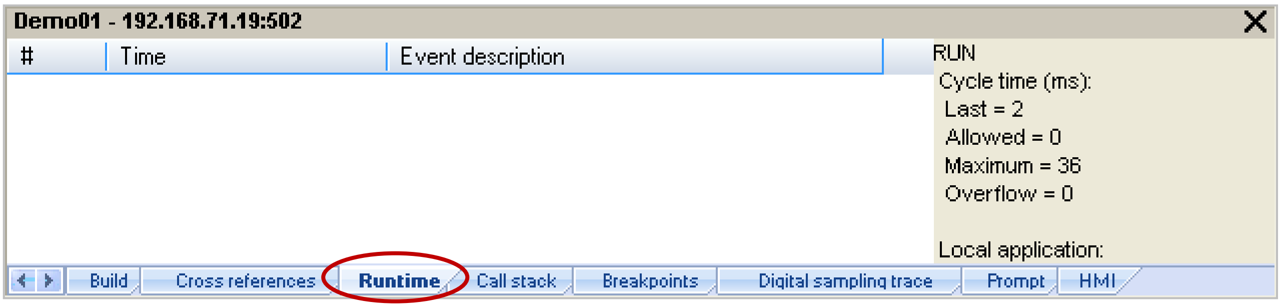
#	Time	Event description
1	2000/01/01 00:15:21.000	Board error in the slot No. 1 !
2	2000/01/01 00:15:29.000	Board error in the slot No. 1 !
3	2000/01/01 00:15:37.000	Board error in the slot No. 1 !

RUN
Cycle time (ms):
Last = 2
Allowed = 0
Maximum = 36
Overflow = 0

Local application:

When doing the “On Line” () operation, it will automatically switch to the “Runtime” tab and you can see if there is any error message for the downloaded program. (E.g., in this example, we need to plug the I-8055W module in the Slot1 of the PAC, and the message “Board error in the slot No. 1 !” means there is no I/O module in the Slot1 or an I/O exception.)

Shutting down the PAC, and plug one I-8055W module in the PAC’s Slot1 then reboots. Then, click “On Line” () button to connect to the PAC.



Build Cross references Runtime Call stack Breakpoints Digital sampling trace Prompt HMI

Demo01 - 192.168.71.19:502

#	Time	Event description

RUN
Cycle time (ms):
Last = 2
Allowed = 0
Maximum = 36
Overflow = 0

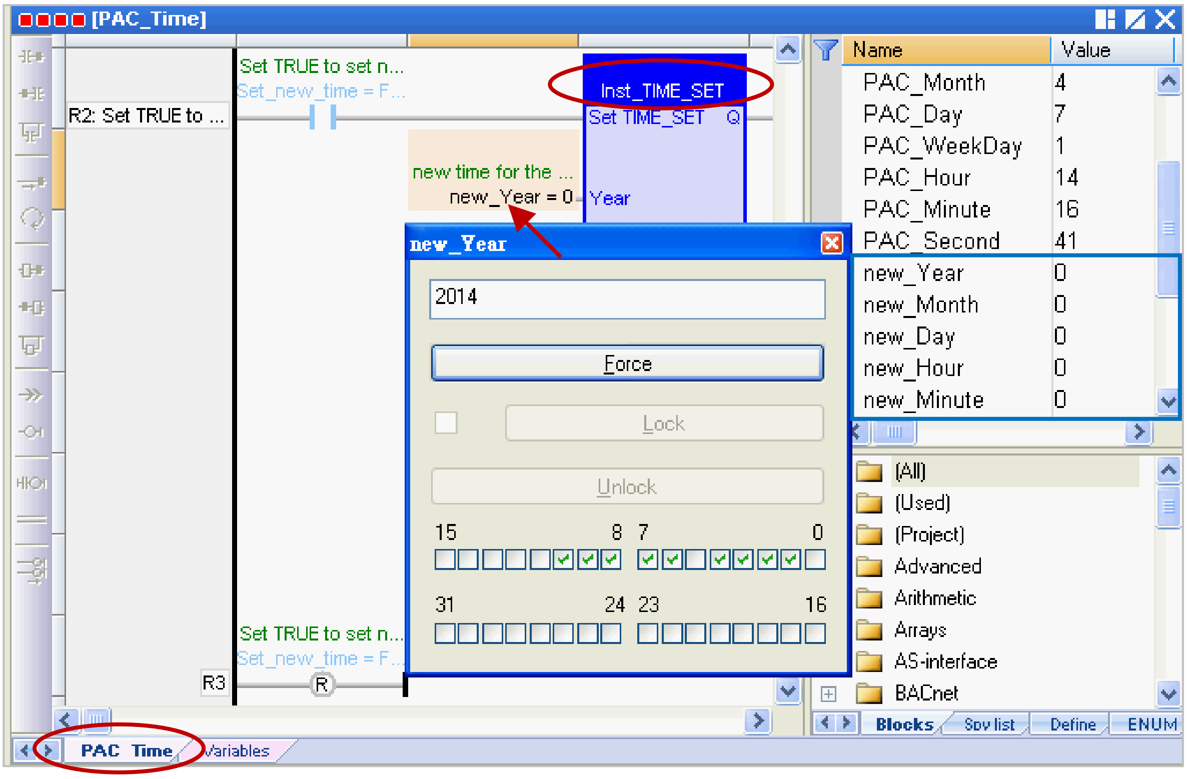
Local application:

2.3.6 Testing the Program

In the previous section, you have successfully downloaded the “Demo01” project and the following will describe how to test the program.

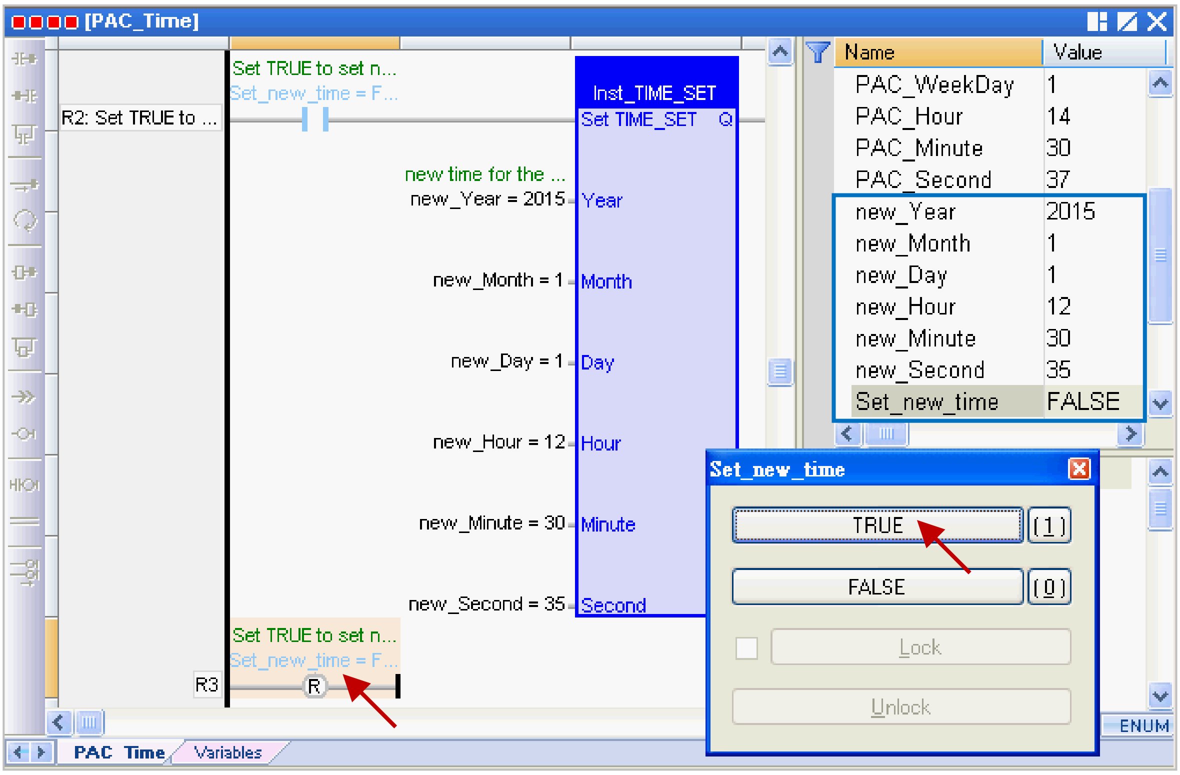
The “PAC_Time” Program:

1. Mouse double-click the variable name (e.g., “new_Year”) in the “TIME_**SET**” function block (or the Variables Area) one-by-one to change the PAC’s system time (e.g., to change it as January 1, 2015 12:30:35).



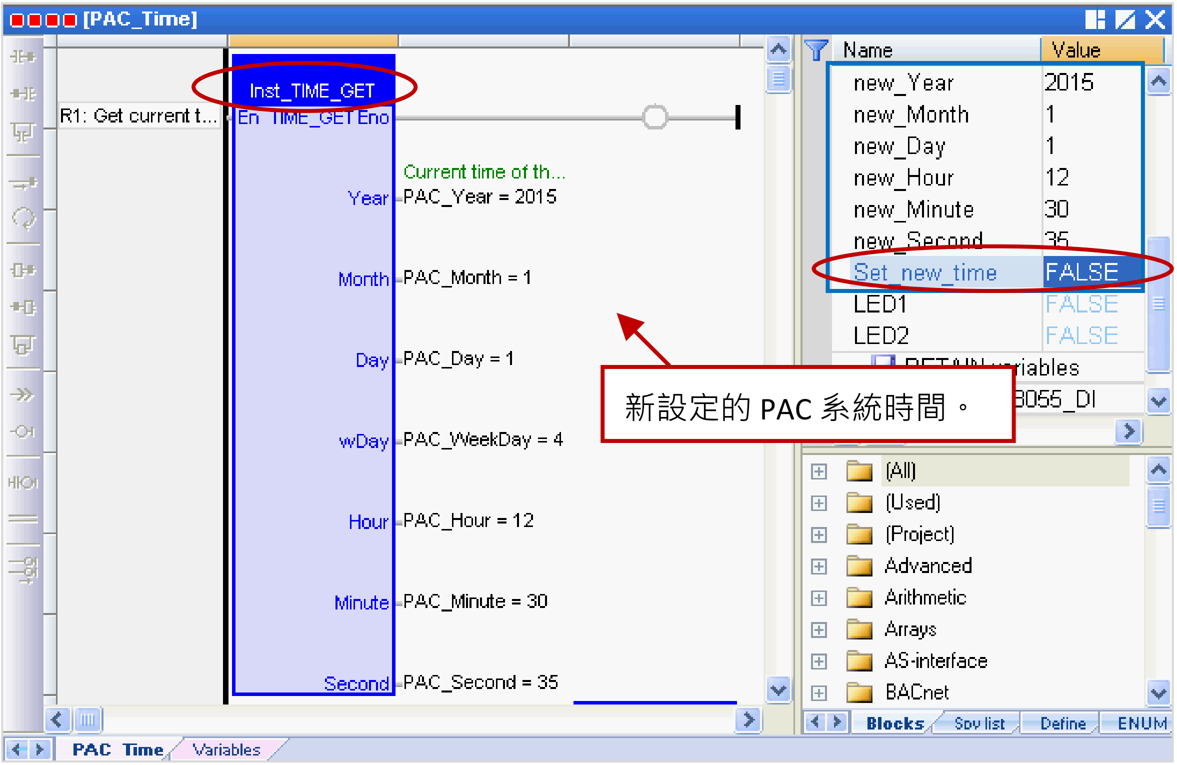
Name	Value
PAC_Month	4
PAC_Day	7
PAC_WeekDay	1
PAC_Hour	14
PAC_Minute	16
PAC_Second	41
new_Year	0
new_Month	0
new_Day	0
new_Hour	0
new_Minute	0

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



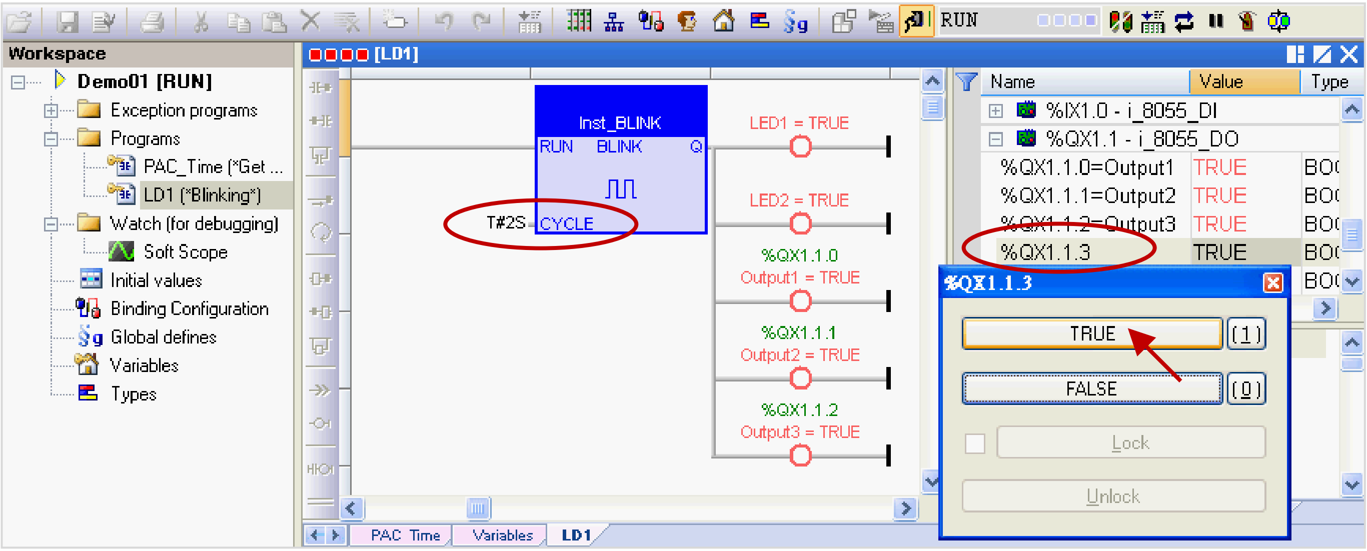
Name	Value
PAC_WeekDay	1
PAC_Hour	14
PAC_Minute	30
PAC_Second	37
new_Year	2015
new_Month	1
new_Day	1
new_Hour	12
new_Minute	30
new_Second	35
Set_new_time	FALSE

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 check to see if the DO0, DO1 and DO2 tags of the I-8055W I/O module that plugged in the slot1 of the PAC is blinking every two seconds (like the value "T#2S" we set before). You can also assign a "TIME" variable on the left side of the "CYCLE" for easy to change the time setting. Refer the Section 2.3.1 for the setting way.
5. If the "%QX1.1.3" variable is set to "TURE" in the Variable Area, the LED4 (i.e., "DO3", at the top of the I-8055W I/O module) will light up.



6. Click the tool button again to cancel the PAC connection.

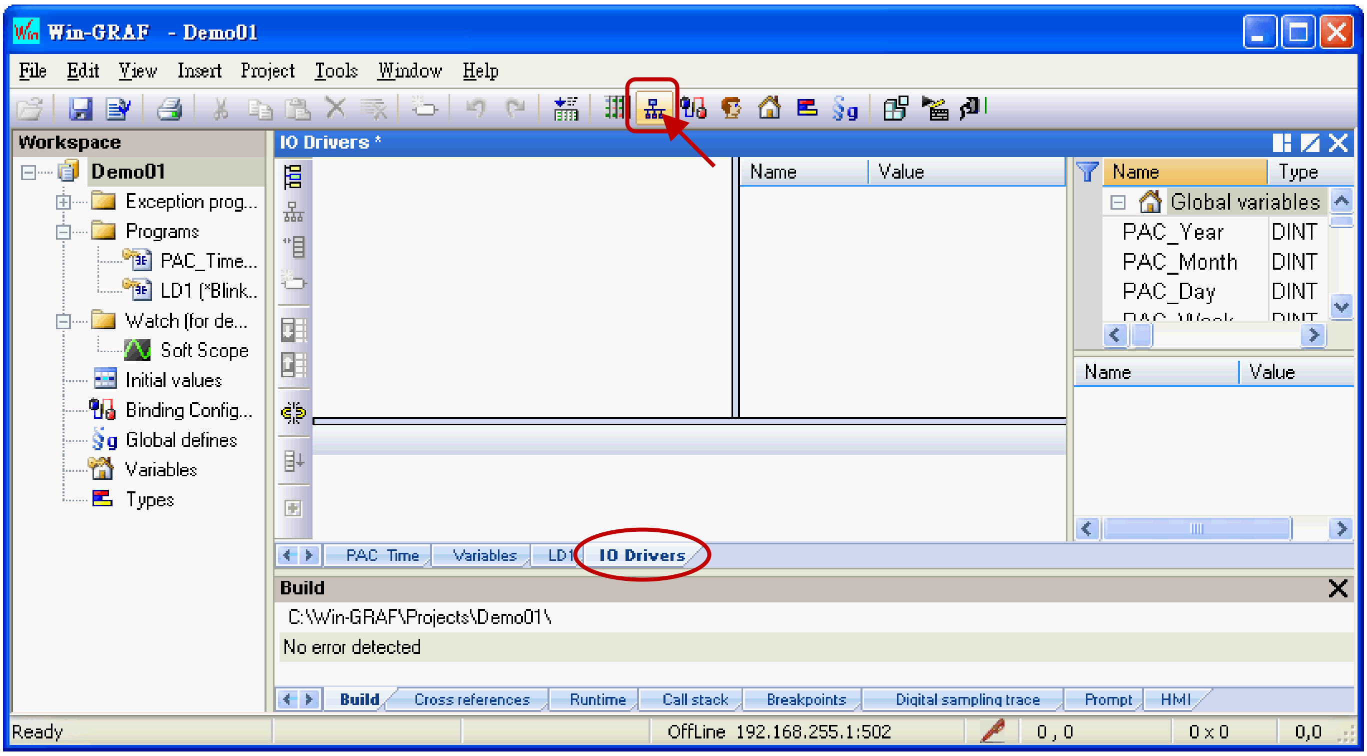
Note: Do NOT click the "Stop Application" button; it will stop the running project in the PAC.

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

In Chapter2, we have described how to get/set the PAC system time (i.e., the "PAC_Time" program) and create a blinking function (i.e., the "LD1" program) in the "Demo01" project. The following sections describe how to allow the SCADA/HMI software (e.g., "InduSoft") to access Win-GRAF variables that defined in the "Demo01" project. The Win-GRAF Workbench provides two ways to open the PAC data, one way is to enable the Win-GRAF PAC as a Modbus **TCP** Slave and the second way is to enable the Win-GRAF PAC as a Modbus **RTU** Slave (you must first complete all the Modbus Slave settings in Section 3.1, and then refer the Section 3.2). To begin, follow these steps:

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

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

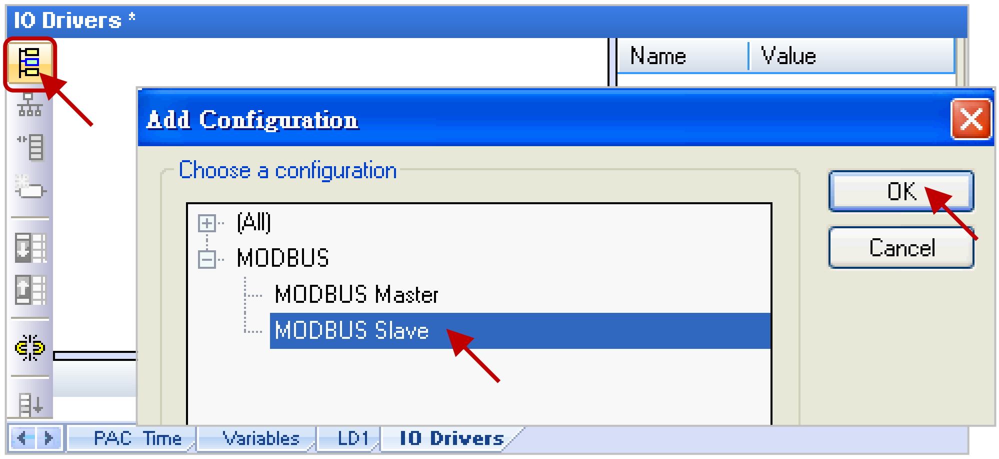


Build

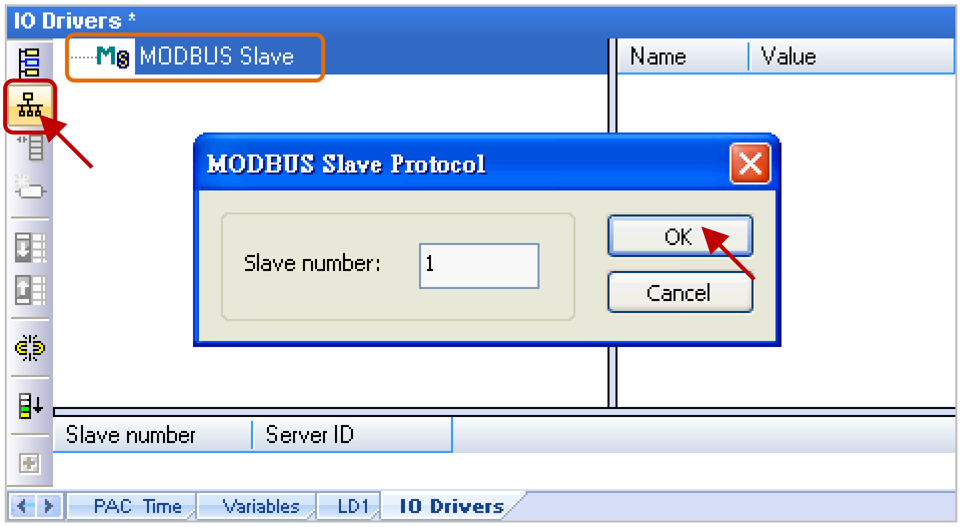
C:\Win-GRAF\Projects\Demo01\

No error detected

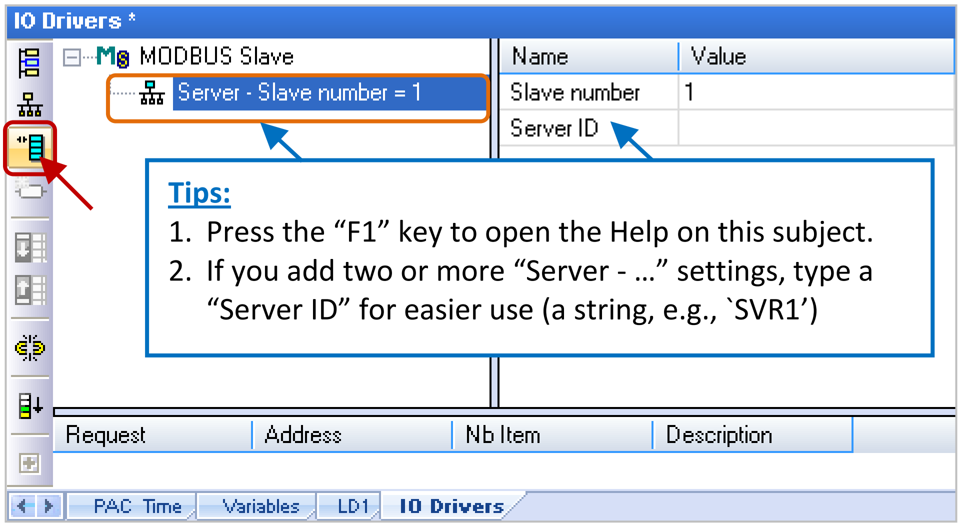
2. Click the "Insert Configuration" button on the left side of the "IO Drivers" window and then select the "MOSBUS Slave" and click "OK" to enable a Modbus TCP Slave.



3. Click the “Insert Master/Port” button on the left side to set the “Slave number” (In this case, the value is “1”), and click the “OK” button.



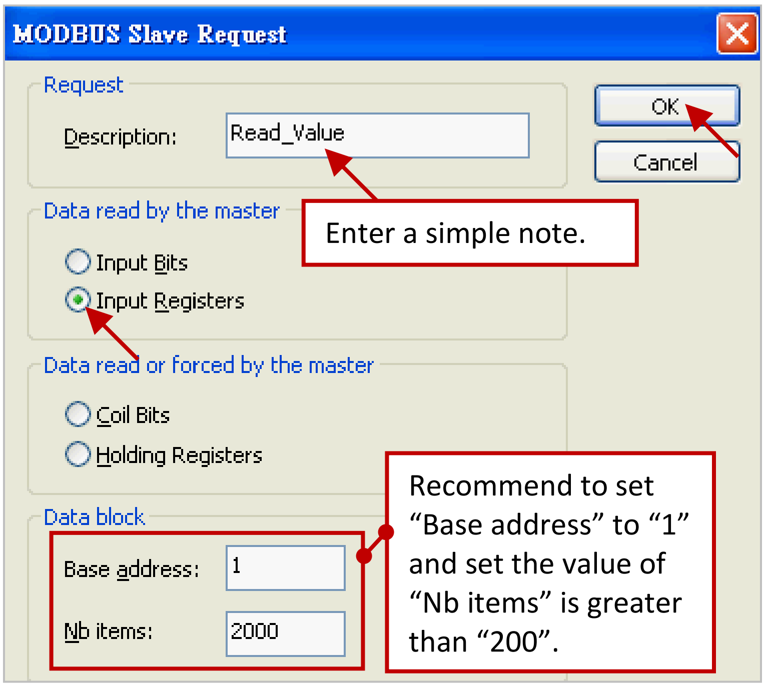
4. Click the “Insert Slave/Data Block” button on the left side to open the “MODBUS Slave Request” window.



Tips:

1. Press the “F1” key to open the Help on this subject.
2. If you add two or more “Server - …” settings, type a “Server ID” for easier use (a string, e.g., `SVR1’)

5. Enter a simple note in the “Description” field and then click the "Input Registers” option.



Enter a simple note.

Recommend to set “Base address” to “1” and set the value of “Nb items” is greater than “200”.

For Modbus Master to Read data:

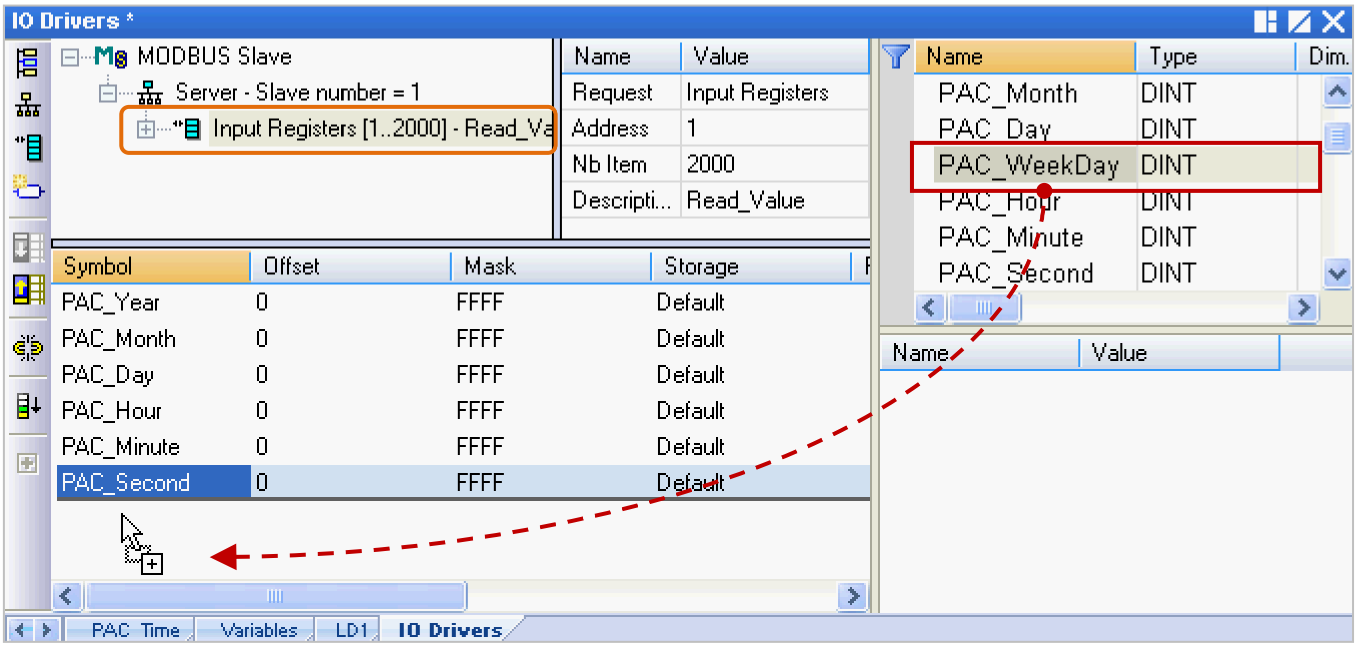
Options	**Data types**
Input-bits	BOOL
Input Registers	BYTE, INT, DINT, REAL, etc.

For Modbus Master to Write data:

Options	**Data types**
Coil-bits	BOOL
Holding Registers	BYTE, INT, DINT, REAL, etc.

(Refer the Appendix A to see more data type)

6. As the screenshot above, it's recommended to set the "Base address" to "1" and the "Nb items" refers to how much variable data can be provided by one "Data block". If the data address requested from the Modbus Master (e.g., the SCADA software) is greater than this value (in this example, the value is "2000"), the Modbus Slave (i.e., Win-GRAF PAC) will not respond.
7. Mouse drags all the needed variables (e.g., "PAC_xxx", data type: "DINT") one-by-one from the Variables area and then drop it to the "Symbol" field.


IO Drivers *
MODBUS Slave
Server - Slave number = 1
Input Registers [1..2000] - Read_Va
PAC_Time
Variables
LD1
IO Drivers

Name	Value
Request	Input Registers
Address	1
Nb Item	2000
Descripti...	Read_Value

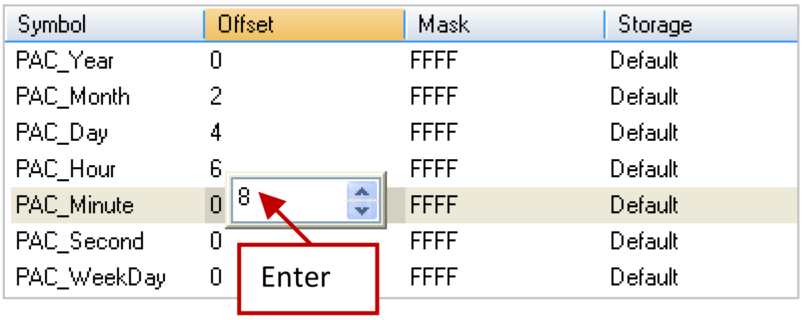
Symbol	Offset	Mask	Storage
PAC_Year	0	FFFF	Default
PAC_Month	0	FFFF	Default
PAC_Day	0	FFFF	Default
PAC_Hour	0	FFFF	Default
PAC_Minute	0	FFFF	Default
PAC_Second	0	FFFF	Default

Name	Type	Dim.
PAC_Month	DINT	
PAC_Day	DINT	
PAC_WeekDay	DINT	
PAC_Hour	DINT	
PAC_Minute	DINT	
PAC_Second	DINT	

8. Mouse double-click the "Offset" field and fill in a value, then press "Enter" key to finish the setting.

Note: (1) The "Offset" value starts at "0" and the Modbus address of variable is equal to this value plus 1 (Base address).

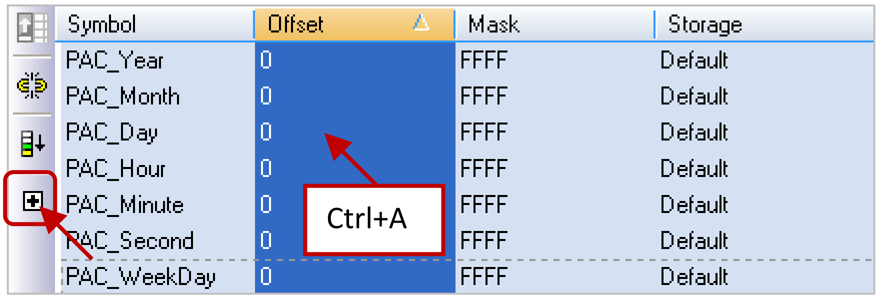
(2) If using a 32-bits (or more than 32-bits) data type (e.g., "DINT", refer the Appendix A), it requires two Modbus addresses, as the table below, the "Offset" values are 0, 2, 4, 6, etc.



Symbol	Offset	Mask	Storage
PAC_Year	0	FFFF	Default
PAC_Month	2	FFFF	Default
PAC_Day	4	FFFF	Default
PAC_Hour	6	FFFF	Default
PAC_Minute	0 8	FFFF	Default
PAC_Second	0	FFFF	Default
PAC_WeekDay	0	FFFF	Default

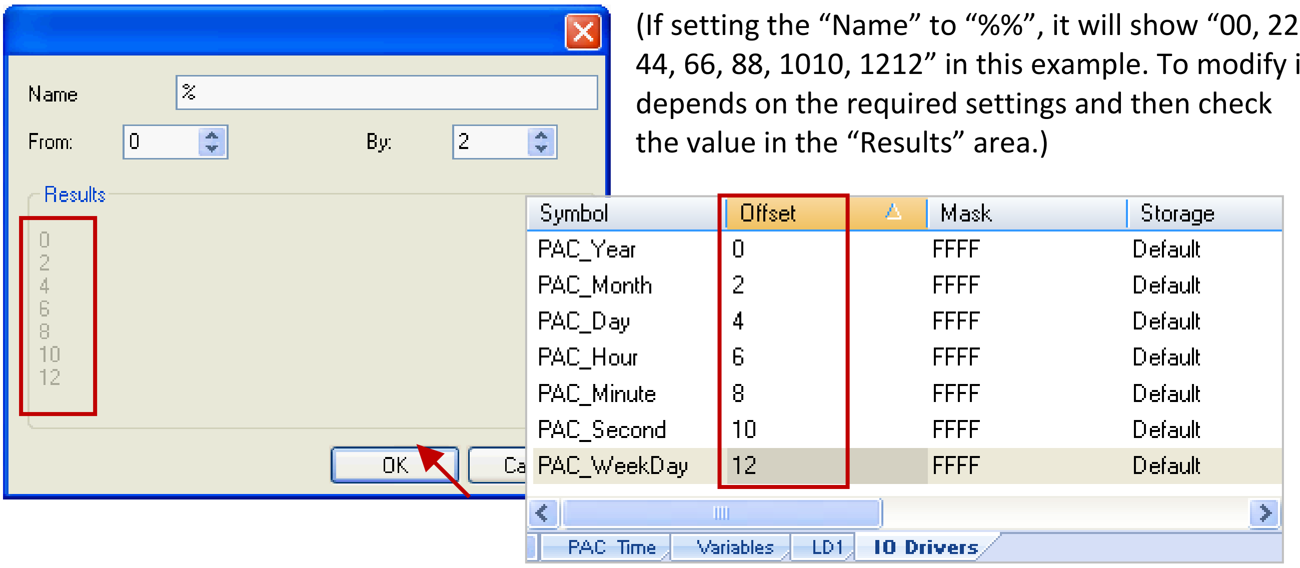
Tips:

Mouse click the "Offset" field and press the keyboard "Ctrl+A" to select all items, and then click the "Iterate Property" button at the left-side to open the settings window.



Symbol	Offset	Mask	Storage
PAC_Year	0	FFFF	Default
PAC_Month	0	FFFF	Default
PAC_Day	0	FFFF	Default
PAC_Hour	0	FFFF	Default
PAC_Minute	0	FFFF	Default
PAC_Second	0	FFFF	Default
PAC_WeekDay	0	FFFF	Default

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



Name: %

From: 0 By: 2

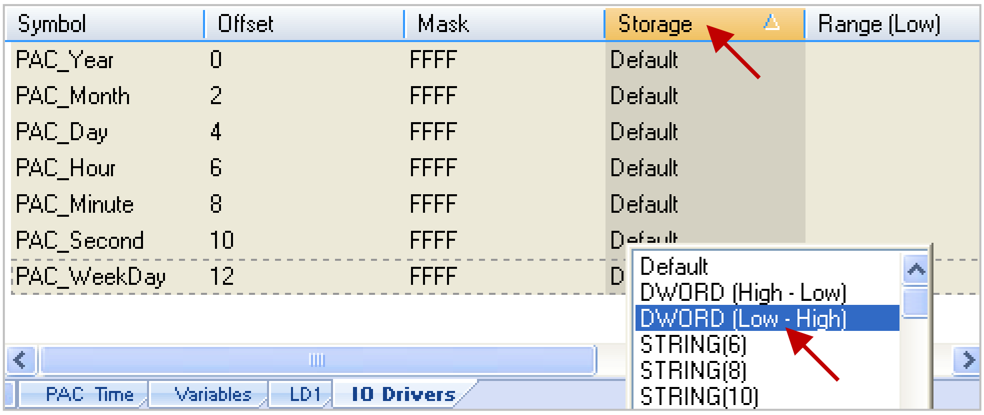
Results

0
2
4
6
8
10
12

(If setting the “Name” to “%%”, it will show “00, 22, 44, 66, 88, 1010, 1212” in this example. To modify it depends on the required settings and then check the value in the “Results” area.)

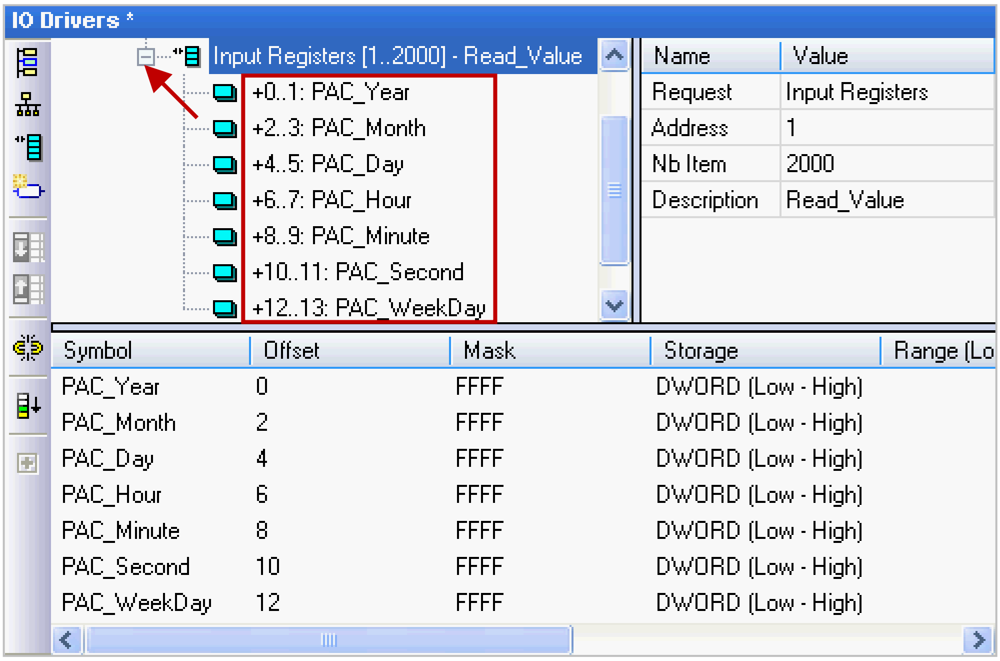
Symbol	Offset	Mask	Storage
PAC_Year	0	FFFF	Default
PAC_Month	2	FFFF	Default
PAC_Day	4	FFFF	Default
PAC_Hour	6	FFFF	Default
PAC_Minute	8	FFFF	Default
PAC_Second	10	FFFF	Default
PAC_WeekDay	12	FFFF	Default

9. Click “Storage” to select entire columns and then press “Enter” key to display a drop-down menu. Then, select “DWORD (Low – High)” and press “Enter” key to complete the setting. (If using a 16-bits or below, it’s no need to set the “Storage” item.)



Symbol	Offset	Mask	Storage	Range (Low)
PAC_Year	0	FFFF	Default	
PAC_Month	2	FFFF	Default	
PAC_Day	4	FFFF	Default	
PAC_Hour	6	FFFF	Default	
PAC_Minute	8	FFFF	Default	
PAC_Second	10	FFFF	Default	
PAC_WeekDay	12	FFFF	D	

To expand this “Data Block” and you can see the Modbus addresses of all variables. It equals to the “Offset” value plus 1 (Base address).



IO Drivers *

Input Registers [1..2000] - Read_Value

+0..1: PAC_Year
+2..3: PAC_Month
+4..5: PAC_Day
+6..7: PAC_Hour
+8..9: PAC_Minute
+10..11: PAC_Second
+12..13: PAC_WeekDay

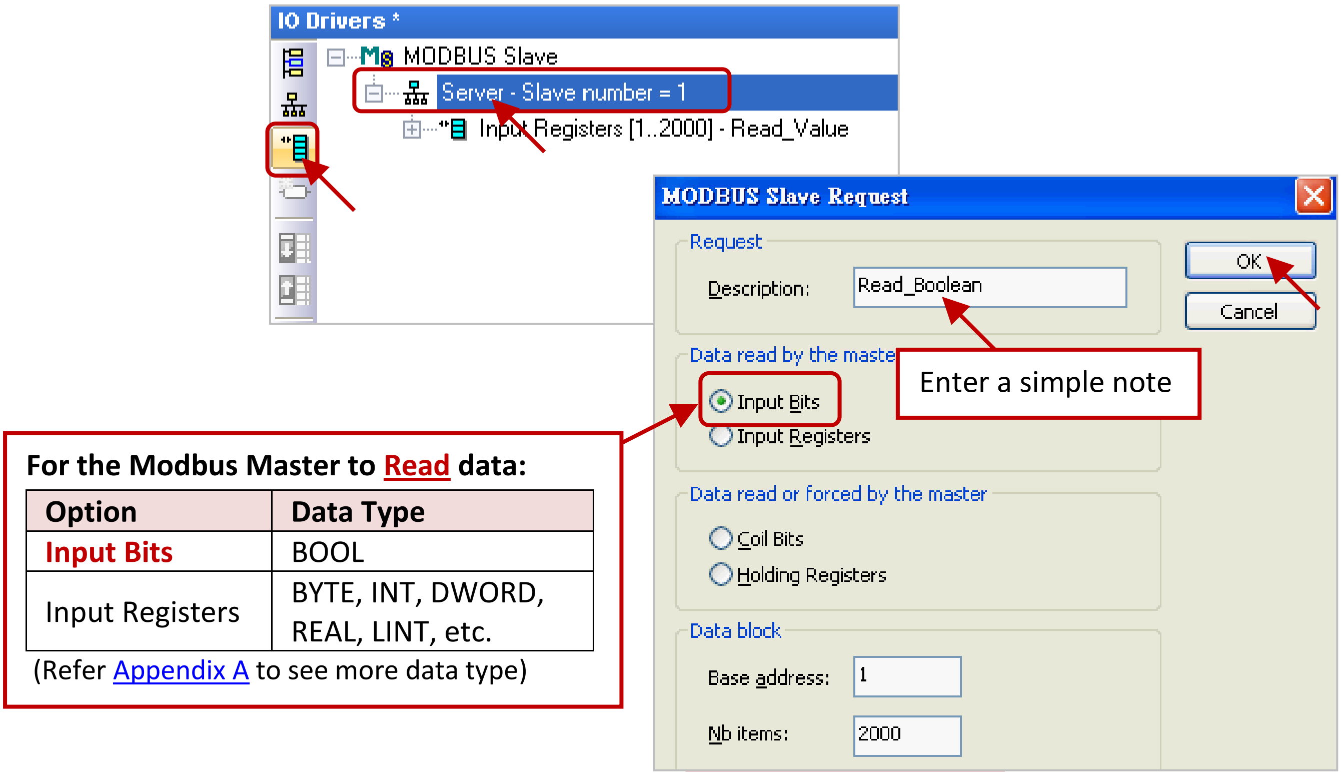
Name	Value
Request	Input Registers
Address	1
Nb Item	2000
Description	Read_Value

Symbol	Offset	Mask	Storage	Range (Lo
PAC_Year	0	FFFF	DWORD (Low - High)	
PAC_Month	2	FFFF	DWORD (Low - High)	
PAC_Day	4	FFFF	DWORD (Low - High)	
PAC_Hour	6	FFFF	DWORD (Low - High)	
PAC_Minute	8	FFFF	DWORD (Low - High)	
PAC_Second	10	FFFF	DWORD (Low - High)	
PAC_WeekDay	12	FFFF	DWORD (Low - High)	

10. Then, we need to add the second "Data Block" for the Modbus Master to read the Boolean data. This configure way is similar to the step 4 to 8:

(1) Click the "Server - …" item and click the "Insert Slave/Data Block" button at the left side to open the settings window.

(2) In the "MODBUS Slave Request" window, enter a simple note and select the "Input-bits" option, then set "Base address" to "1" and set "Nb items" to "2000".



IO Drivers *

- MODBUS Slave
  - Server - Slave number = 1
    - Input Registers [1..2000] - Read_Value

MODBUS Slave Request

Request

Description: Read_Boolean

Enter a simple note

Data read by the master

● Input Bits

○ Input Registers

Data read or forced by the master

○ Coil Bits

○ Holding Registers

Data block

Base address: 1

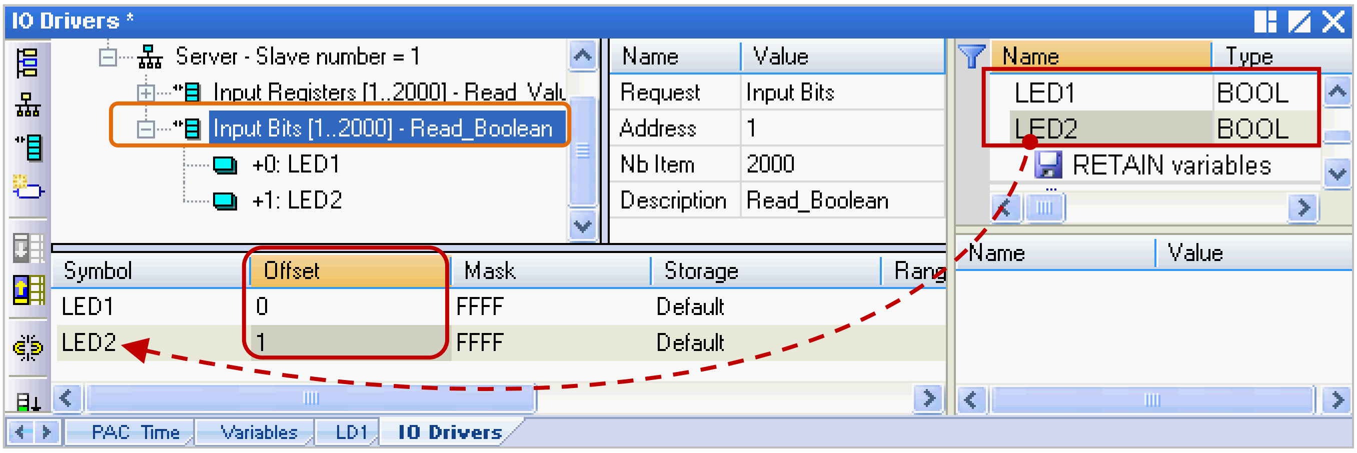
Nb items: 2000

For the Modbus Master to Read data:

Option	Data Type
Input Bits	BOOL
Input Registers	BYTE, INT, DWORD, REAL, LINT, etc.

(Refer Appendix A to see more data type)

(3) Mouse drags the Boolean variables (i.e., "LED1", "LED2"; data type: BOOL) one-by-one and drop them to the "Symbol" area, and then set the "Offset" to "0" and to "1".



IO Drivers *

- Server - Slave number = 1
  - Input Registers [1..2000] - Read_Valu
  - Input Bits [1..2000] - Read_Boolean
    - +0: LED1
    - +1: LED2

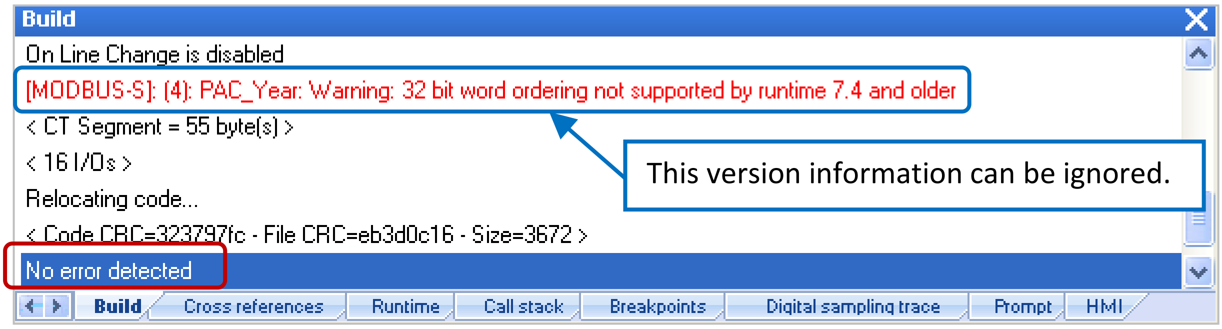
Name	Value
Request	Input Bits
Address	1
Nb Item	2000
Description	Read_Boolean

Name	Type
LED1	BOOL
LED2	BOOL
RETAIN variables	

Symbol	Offset	Mask	Storage	Rang
LED1	0	FFFF	Default	
LED2	1	FFFF	Default	

You have completed the settings for the Modbus Slave. Finally, follow the way below to re-compile the program and download it to the Win-GRAF PAC.

11. Click “Project” > “Build All Projects” from the menu bar to compile this program again (refer the Section 2.3.4). If a message informs you “No error detected” that means this process is successful.



Build

On Line Change is disabled

[MODBUS-S]: (4): PAC_Year: Warning: 32 bit word ordering not supported by runtime 7.4 and older

< CT Segment = 55 byte(s) >

< 16 I/Os >

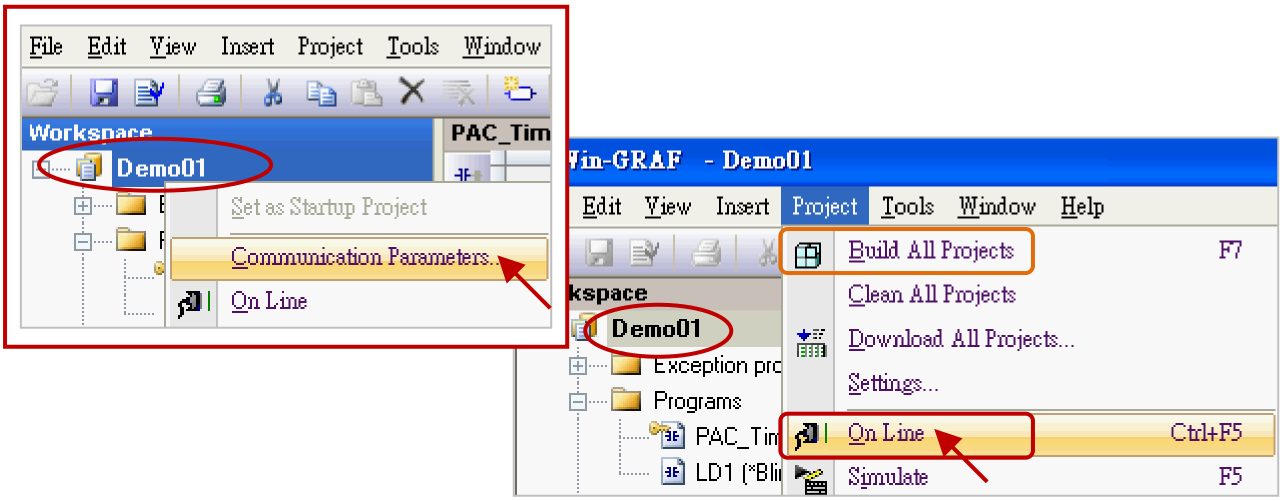
Relocating code...

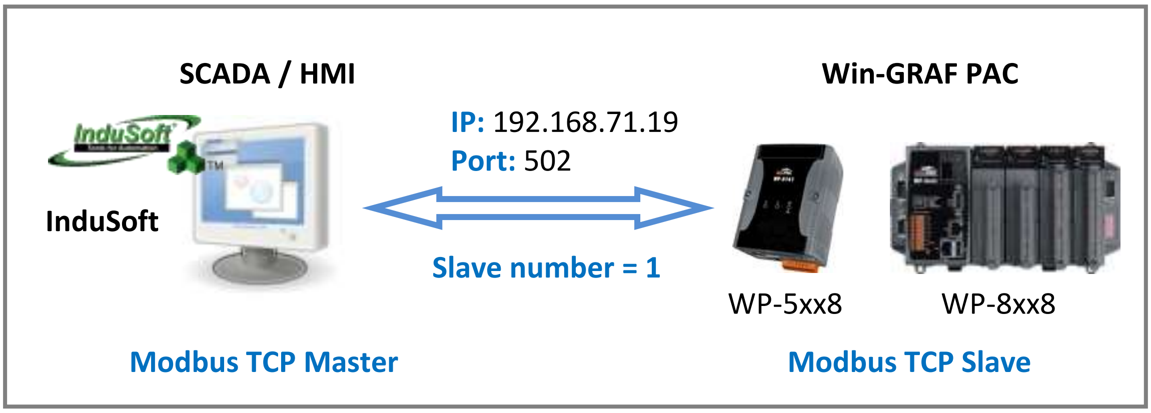
< Code CRC=323797fc - File CRC=eb3d0c16 - Size=3672 >

No error detected

This version information can be ignored.

12. Mouse right-click the project name (i.e., “Demo01”) and select the “Communication Parameters...” to set the PAC IP (e.g., “192.168.71.19:502”) and then click the menu bar “Project” > “On Line” (or) to establish a connection and download this project to the Win-GRAF PAC. (Refer the Section 2.3.5).





(Refer the P1-1 to view all PAC models)

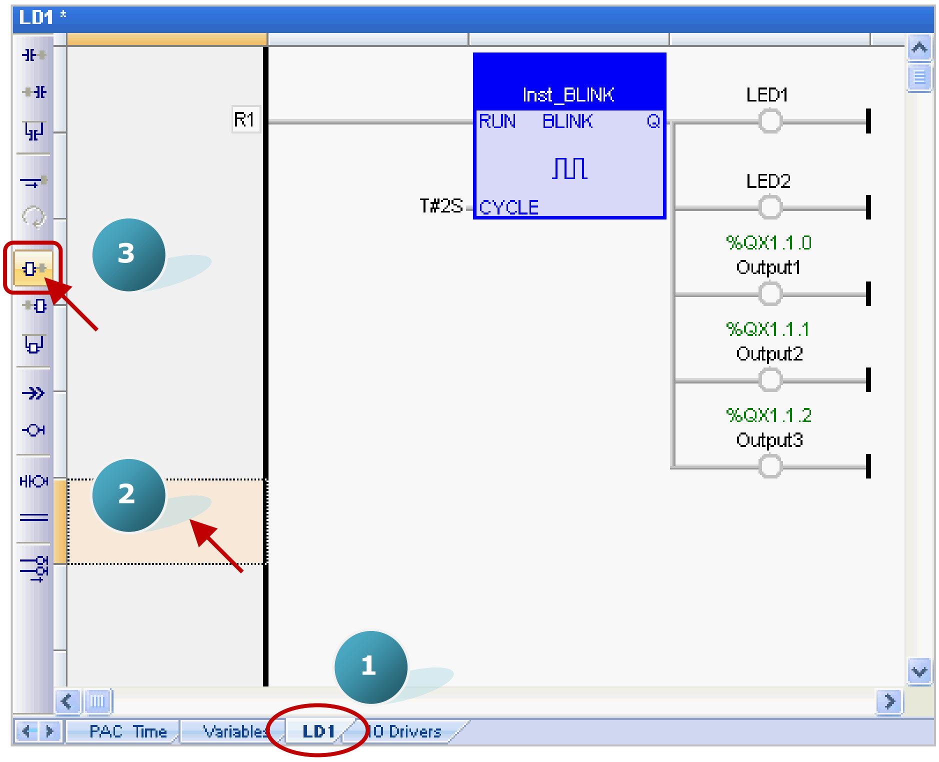
After completing all the steps, the HMI/SCADA software can access to all the Win-GRAF variables listed above via Modbus TCP protocol.

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

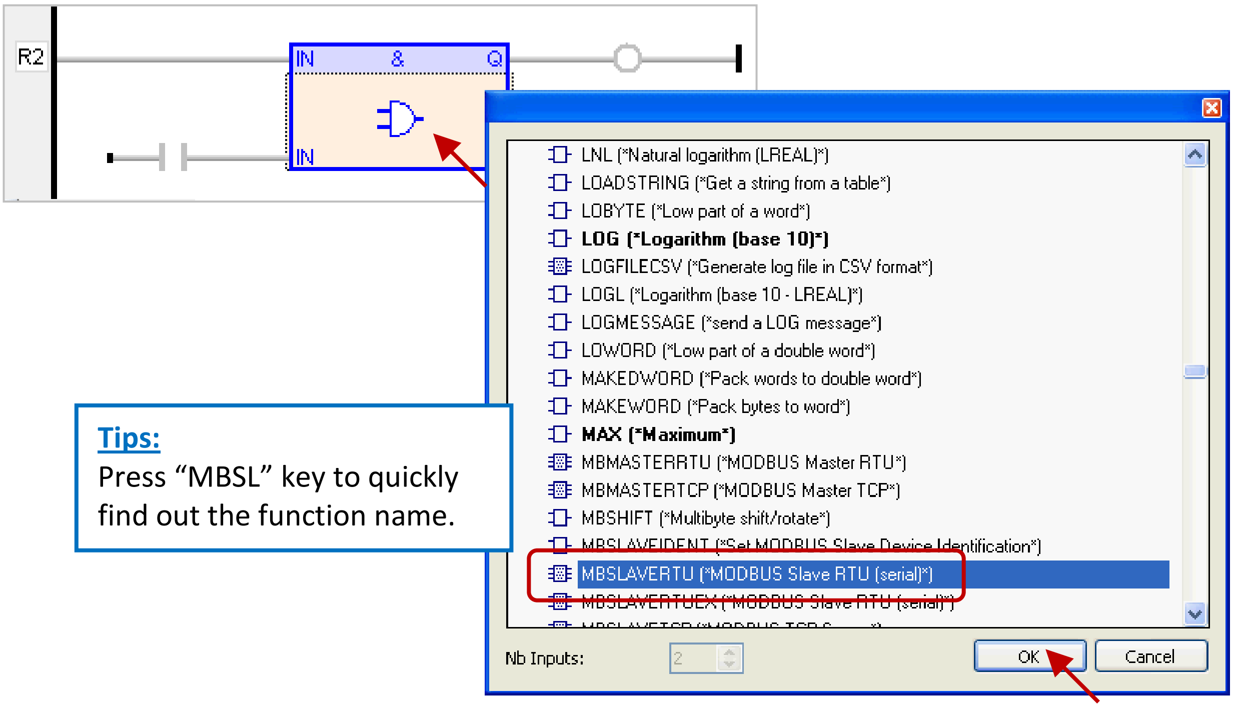
Before doing this, you must complete all the content that described in Section 3.1 to open the Modbus Slave data. The way to enable the Win-GRAF PAC as the Modbus RTU Slave is to add the “MBSLAVERTU” or the “MBSLAVERTU**EX**” function block in the program. To begin, follow these steps:

Add the “MBSLAVERTU” function block

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



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



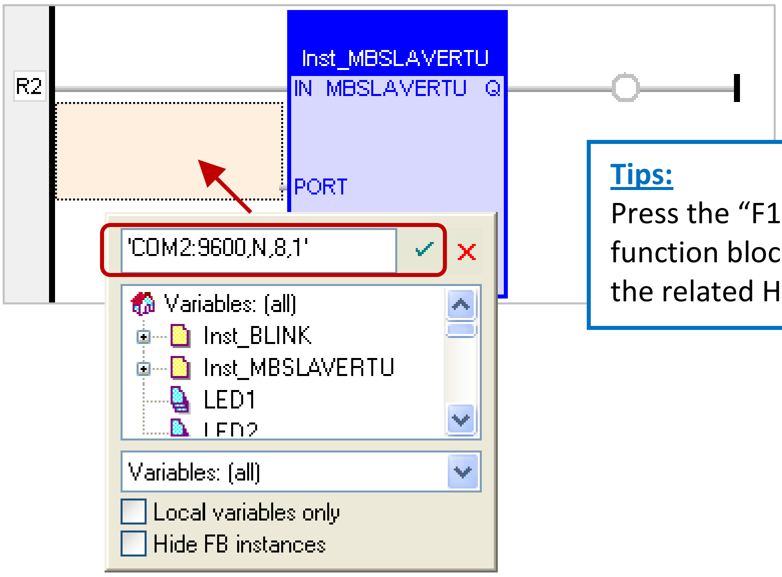
LNL (*Natural logarithm (LREAL)*)
LOADSTRING (*Get a string from a table*)
LOBYTE (*Low part of a word*)
LOG (*Logarithm (base 10)*)
LOGFILECSV (*Generate log file in CSV format*)
LOGL (*Logarithm (base 10 - LREAL)*)
LOGMESSAGE (*send a LOG message*)
LOWORD (*Low part of a double word*)
MAKEDWORD (*Pack words to double word*)
MAKEWORD (*Pack bytes to word*)
MAX (*Maximum*)
MBMASTERRTU (*MODBUS Master RTU*)
MBMASTERTCP (*MODBUS Master TCP*)
MBSHIFT (*Multibyte shift/rotate*)
MBSLAVEIDENT (*Set MODBUS Slave Device Identification*)
MBSLAVERTU (*MODBUS Slave RTU (serial)*)
MBSLAVERTUEX (*MODBUS Slave RTU (serial)*)
MBSLAVETCP (*MODBUS TCP S

Nb Inputs: 2

OK Cancel

Tips:
Press “MBSL” key to quickly find out the function name.

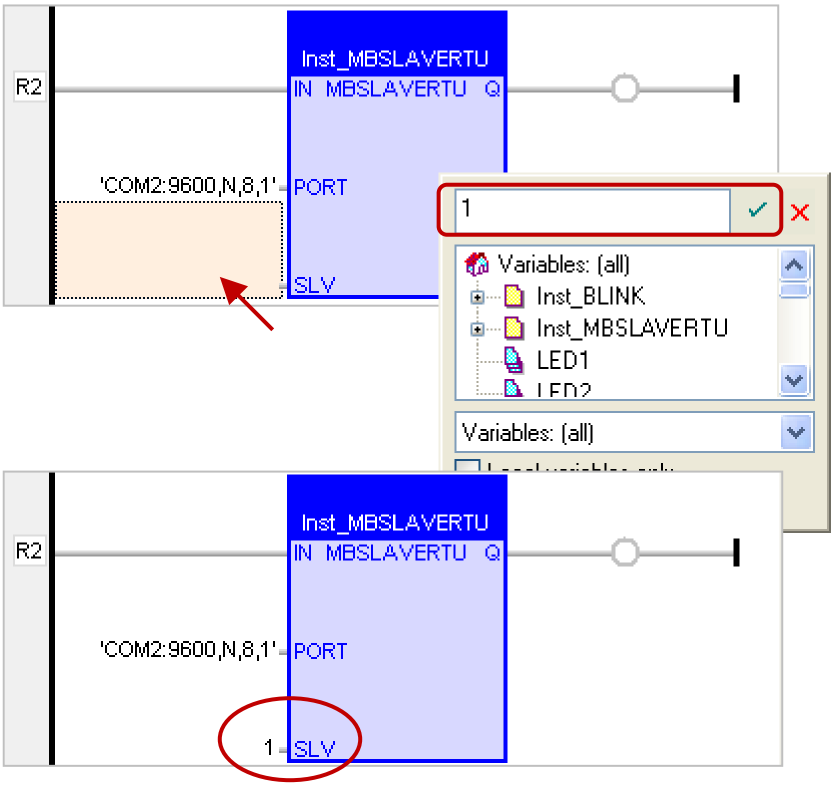
3. In the “MBSLAVERTU” function block, mouse double-click the left side of the “PORT” and enter a string 'COM2:9600,N,8,1' (it means using the Win-GRAF PAC’s COM2 to communication with the Modbus Master) and then click ✓ to complete the settings.



'COM2:9600,N,8,1'

Tips:
Press the “F1” key on this function block to show up the related Help.

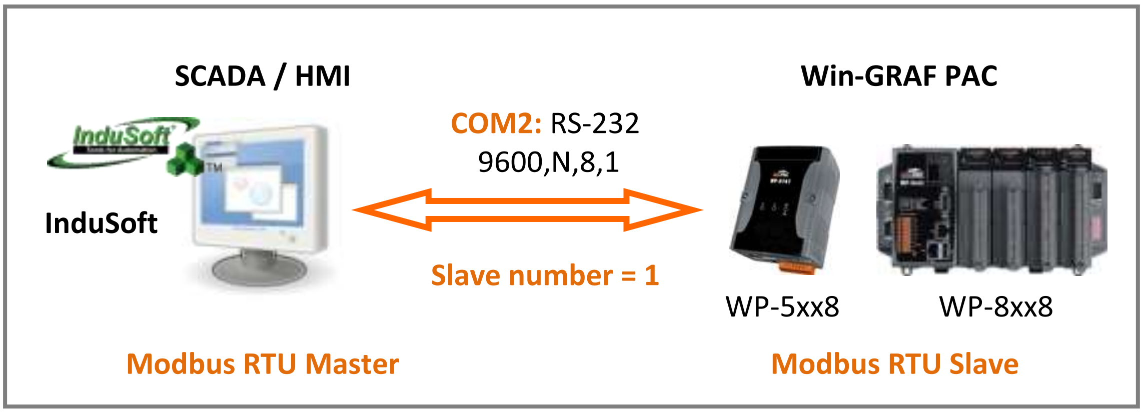
4. Double-click the left side of the “SLV” and then enter “1” (the value set in the Section 3.1 - Step 3), then click ✓ to finish the setting.



1

Now, you have completed the setting of the “MBSLAVERTU” function block, then re-compile the program and download it to the Win-GRAF PAC. (Refer the Section 2.3.4, Section 2.3.5)

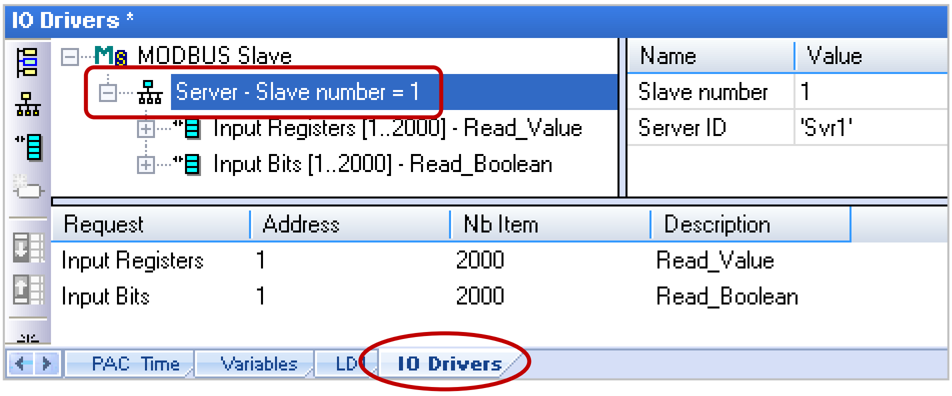
Note: Users can enable multiple Modbus RTU Slave ports for each PAC (recommend not over 16 Ports), the way is to add multiple “MBSLAVERTU” function blocks and set the different “Port” value.



(Refer the P1-1 to view all PAC models)

Add the "MBSLAVERTUEX" function block

If you have added multiple "Server - …" settings (recommend to set one) in the "IO Drivers" window, the "MBSLAVERTU**EX**" function block must be used.



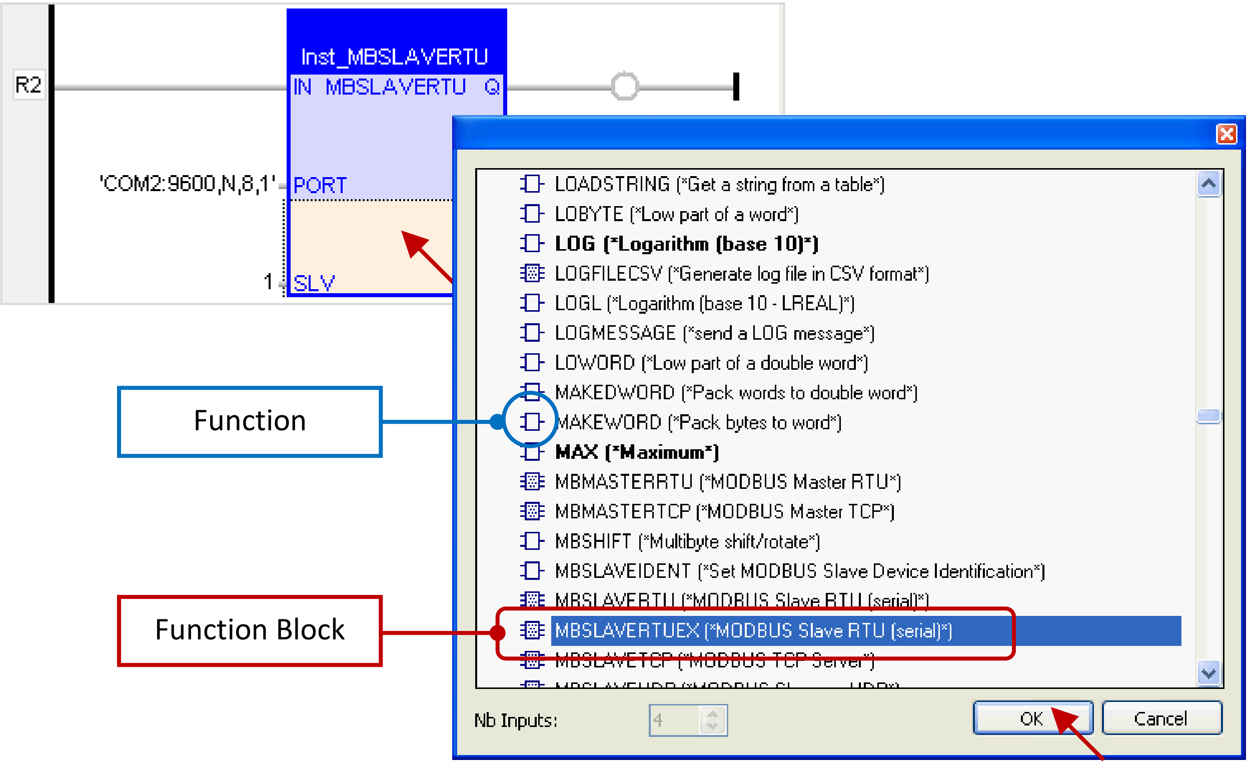
IO Drivers *

- MODBUS Slave
  - Server - Slave number = 1
    - Input Registers [1..2000] - Read_Value
    - Input Bits [1..2000] - Read_Boolean

Name	Value
Slave number	1
Server ID	'Svr1'

Request	Address	Nb Item	Description
Input Registers	1	2000	Read_Value
Input Bits	1	2000	Read_Boolean

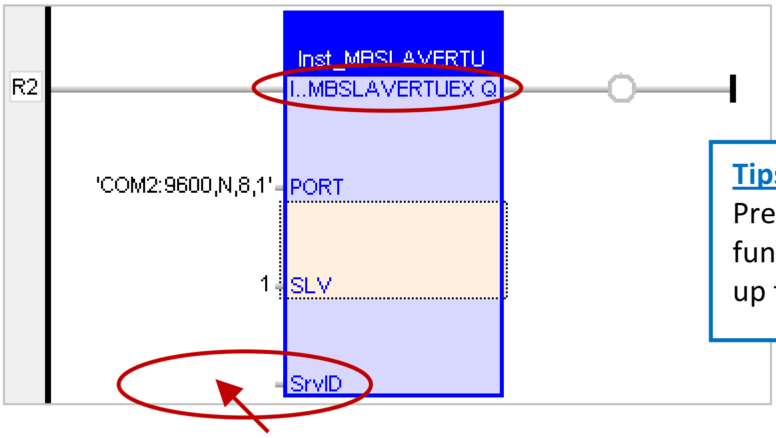
1. Follow the step1 to 4 described above to add the "MBSLAVERTU**EX**" function block. If you want to change the usage for existing function block, mouse double-click the "MBSLAVERTU" and change it to the"MBSLAVERTU**EX**", and then click the "OK" button.


R2
Inst_MBSLAVERTU
IN MBSLAVERTU Q
'COM2:9600,N,8,1'
PORT
1
SLV
LOADSTRING (*Get a string from a table*)
LOBYTE (*Low part of a word*)
LOG (*Logarithm (base 10)*)
LOGFILECSV (*Generate log file in CSV format*)
LOGL (*Logarithm (base 10 - LREAL)*)
LOGMESSAGE (*send a LOG message*)
LOWORD (*Low part of a double word*)
MAKEDWORD (*Pack words to double word*)
MAKEWORD (*Pack bytes to word*)
MAX (*Maximum*)
MBMASTERRTU (*MODBUS Master RTU*)
MBMASTERTCP (*MODBUS Master TCP*)
MBSHIFT (*Multibyte shift/rotate*)
MBSLAVEIDENT (*Set MODBUS Slave Device Identification*)
MBSLAVERTU (*MODBUS Slave RTU (serial)*)
MBSLAVERTUEX (*MODBUS Slave RTU (serial)*)
MBSLAVETCP (*MODBUS TCP Server*)
Nb Inputs:
4
OK
Cancel
Function
Function Block

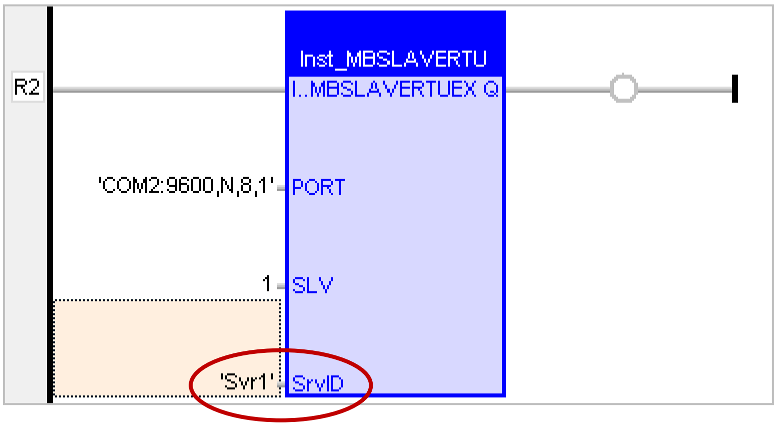
2. The “MBSLAVERTU**EX**” has a “SrvID” setting. Double click the left side of the “SrvID” and enter a needed “Server ID” (using a string format, e.g., ‘Svr1’).

   Note:

   Using the “MBSLAVERTU” function block means the first Mobus Slave setting will be enabled. Using the “MBSLAVERTU**EX**” function block means to enable the Modbus Slave setting depends on the “Server ID”.



Tips:
Press the “F1” on this function block to show up the related Help.



Now, you have finished the settings for the “MBSLAVERTU**EX**” function block and then re-compile the program and download it to the Win-GRAF PAC. (Refer the Section 2.3.4, Section 2.3.5)

Chapter 4 Linking "I/O Boards"

This section lists the usage of the "I/O Boards" function in the Win-GRAF Workbench to link the Real I/O modules or to enable other I/O functions. First, you need to know the slot numbering and supported I/O modules for each PAC:

PAC Model	Slot No. (from the left to the right)	The supported PAC I/O modules
WP-8xx8	0 to 7	**Supported:** I-8K and I-87K series (High Profile) I/O modules. (E.g., I-8017HW and I-87055W) **Not Supported:** I-8K and I-87K series (Low Profile) I/O modules. (E.g., I-8017H and I-87055)
XP-8x48 (*)	1 to 7	
XP-8x48-CE6 (*)		
XP-8x48-Atom-CE6		
VP-25W8	0 to 2	
VP-4138		
WP-5xx8	-	Palm-size PAC, it can support one XV board. (E.g., XV107, XV116, XV308, etc.) The XV board belongs to the Modbus slave I/O board. (Refer the Section 5.1.6 to 5.1.11 for using XV-boards.)

(*) : The XP-8048 and XP-8048-CE6 are the 0-slot PAC. (Refer P1-1 for all PAC models)

Add the "I/O board"

"I/O board" refers to the I/O functionality in the Win-GRAF (e.g., "i_8037_DO") and "I/O module" refers to the hardware device (e.g., "I-8037W").

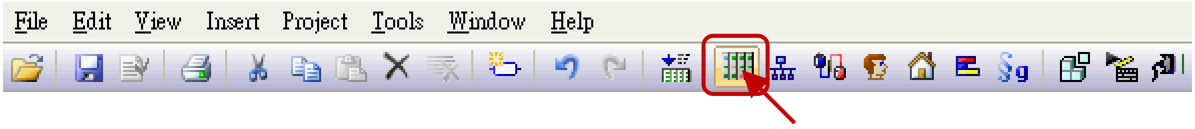
Notice:

Besides the software setting for the I/O board, there are some kinds of I/O modules need to set the hardware Jumper (e.g., Single-ended and Differential Jumper). So, go to the website to look up the product information, or the description printed on the module cover, or the attached shipment document.

I-8K and I-87K series product website:

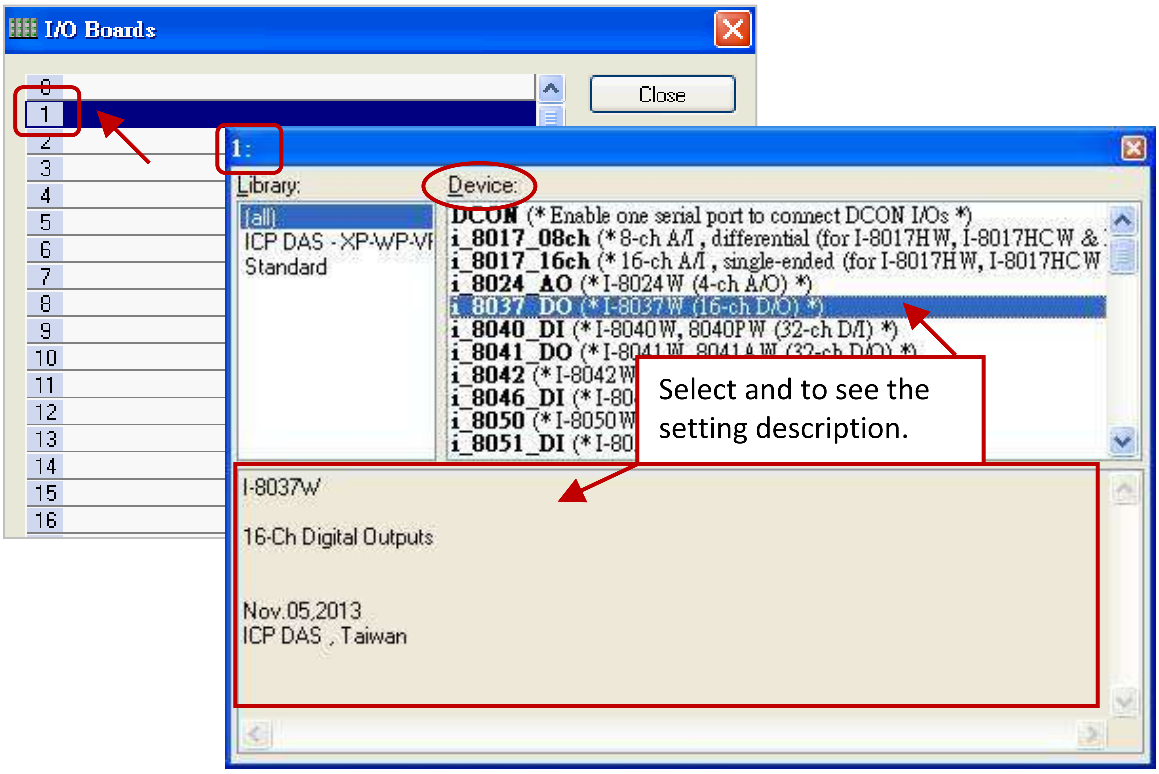
http://www.icpdas.com/root/product/solutions/remote_io/rs-485/i-8k_i-87k/i-8k_i-87k_selection.html

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



2. As the figure below, mouse double click the slot number that corresponds to the real I/O module and then select the proper I/O board (e.g., i_8037_DO).

Note: The Slot 0 to Slot 7 are reserved for real I/O modules that plugged into the PAC, and the slot 8 or above are for other usage.



Library:
- (all)
- ICP DAS - XP-WP-VI
- Standard

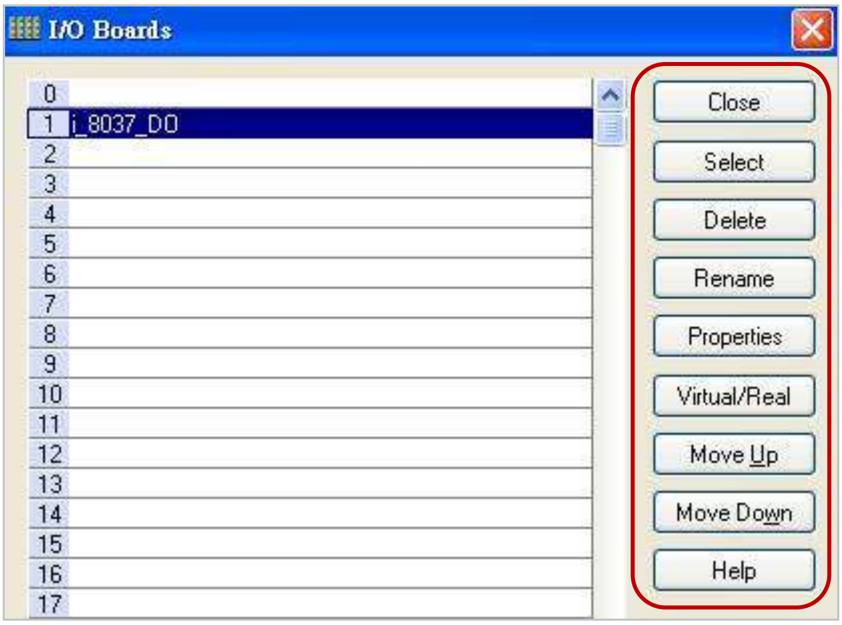
Device:

```
DCON (* Enable one serial port to connect DCON I/Os *)
i_8017_08ch (* 8-ch A/I , differential (for I-8017HW, I-8017HCW &
i_8017_16ch (* 16-ch A/I , single-ended (for I-8017HW, I-8017HCW
i_8024_AO (* I-8024W (4-ch A/O) *)
i_8037_DO (* I-8037W (16-ch D/O) *)
i_8040_DI (* I-8040W, 8040PW (32-ch D/I) *)
i_8041_DO (* I-8041W, 8041AW (32-ch D/O) *)
i_8042 (* I-8042W
i_8046_DI (* I-80
i_8050 (* I-8050W
i_8051_DI (* I-80
```

Select and to see the setting description.

```
I-8037W

16-Ch Digital Outputs


Nov.05,2013
ICP DAS , Taiwan
```

Buttons Description: (Click the following buttons to modify the settings)



"Close":
Close this window.

"Select":
Open the I/O selecting window.
(Hot Key - "Enter": to open ;
Hot Key - "ESC" : to exit)

"Delete":
Delete this I/O board.

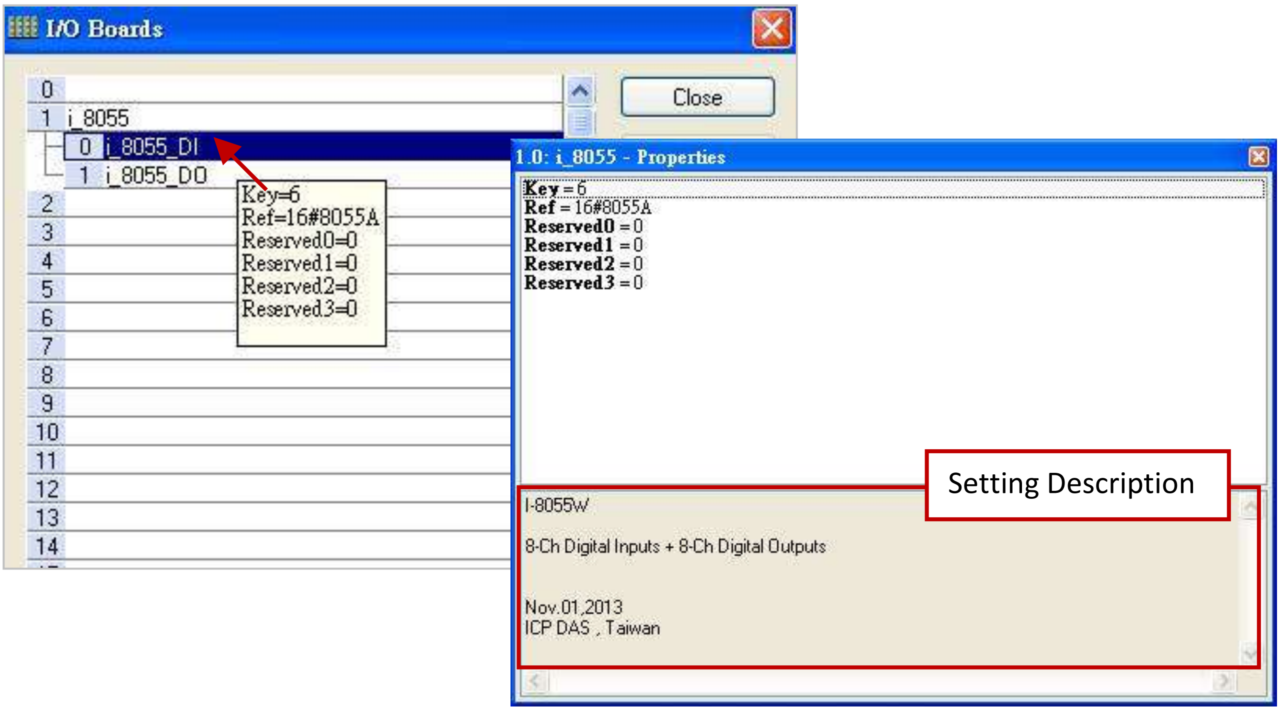
"Rename":
Rename this I/O board.

"Properties": Look up the usage of this I/O board.
"Virtual/Real": Swith the I/O board to a Virtual I/O (for testing) or a Real I/O. (Hot Key – "Space")
"Move Up": Move up this I/O board.
"Move Down": Move down this I/O board.
"Help": To see the description on "I/O devices".

4.1 DI/DO Boards

Here use the "I-8055W" as an example, users can refer the Chapter 4 (P4-1) to add this I/O board.

1. Double-click the “i_8055_DI” (or the “i_8055_DO”) to open the “Properties” window.
   Note: A mouse-over showing the details on the “i_8055_DI” (or the “i_8055_DO”).



```
0
1  i_8055
   0  i_8055_DI
   1  i_8055_DO
2
3
4
5
6
7
8
9
10
11
12
13
14
```

```
Key=6
Ref=16#8055A
Reserved0=0
Reserved1=0
Reserved2=0
Reserved3=0
```

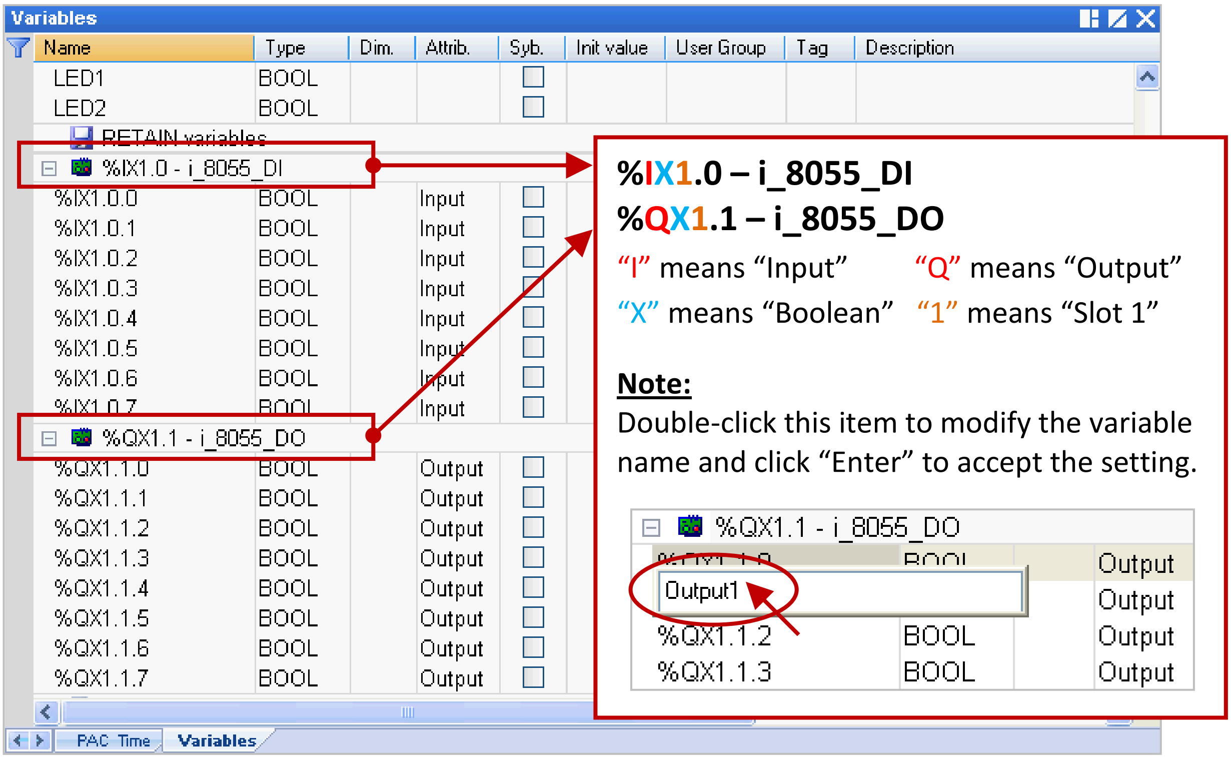
1.0: i_8055 - Properties

```
Key = 6
Ref = 16#8055A
Reserved0 = 0
Reserved1 = 0
Reserved2 = 0
Reserved3 = 0
```

Setting Description

```
I-8055W

8-Ch Digital Inputs + 8-Ch Digital Outputs


Nov.01,2013
ICP DAS , Taiwan
```

2. After linking the “i_8055” I/O board, it will auto add 8 Input and 8 output variables in the “Variables” window that can be used in the program.



Name	Type	Dim.	Attrib.	Syb.	Init value	User Group	Tag	Description
LED1	BOOL			□				
LED2	BOOL			□				
RETAIN variables								
⊟ %IX1.0 - i_8055_DI								
%IX1.0.0	BOOL		Input	□				
%IX1.0.1	BOOL		Input	□				
%IX1.0.2	BOOL		Input	□				
%IX1.0.3	BOOL		Input	□				
%IX1.0.4	BOOL		Input	□				
%IX1.0.5	BOOL		Input	□				
%IX1.0.6	BOOL		Input	□				
%IX1.0.7	BOOL		Input	□				
⊟ %QX1.1 - i_8055_DO								
%QX1.1.0	BOOL		Output	□				
%QX1.1.1	BOOL		Output	□				
%QX1.1.2	BOOL		Output	□				
%QX1.1.3	BOOL		Output	□				
%QX1.1.4	BOOL		Output	□				
%QX1.1.5	BOOL		Output	□				
%QX1.1.6	BOOL		Output	□				
%QX1.1.7	BOOL		Output	□				

%IX1.0 – i_8055_DI
%QX1.1 – i_8055_DO

“I” means “Input” “Q” means “Output”
“X” means “Boolean” “1” means “Slot 1”

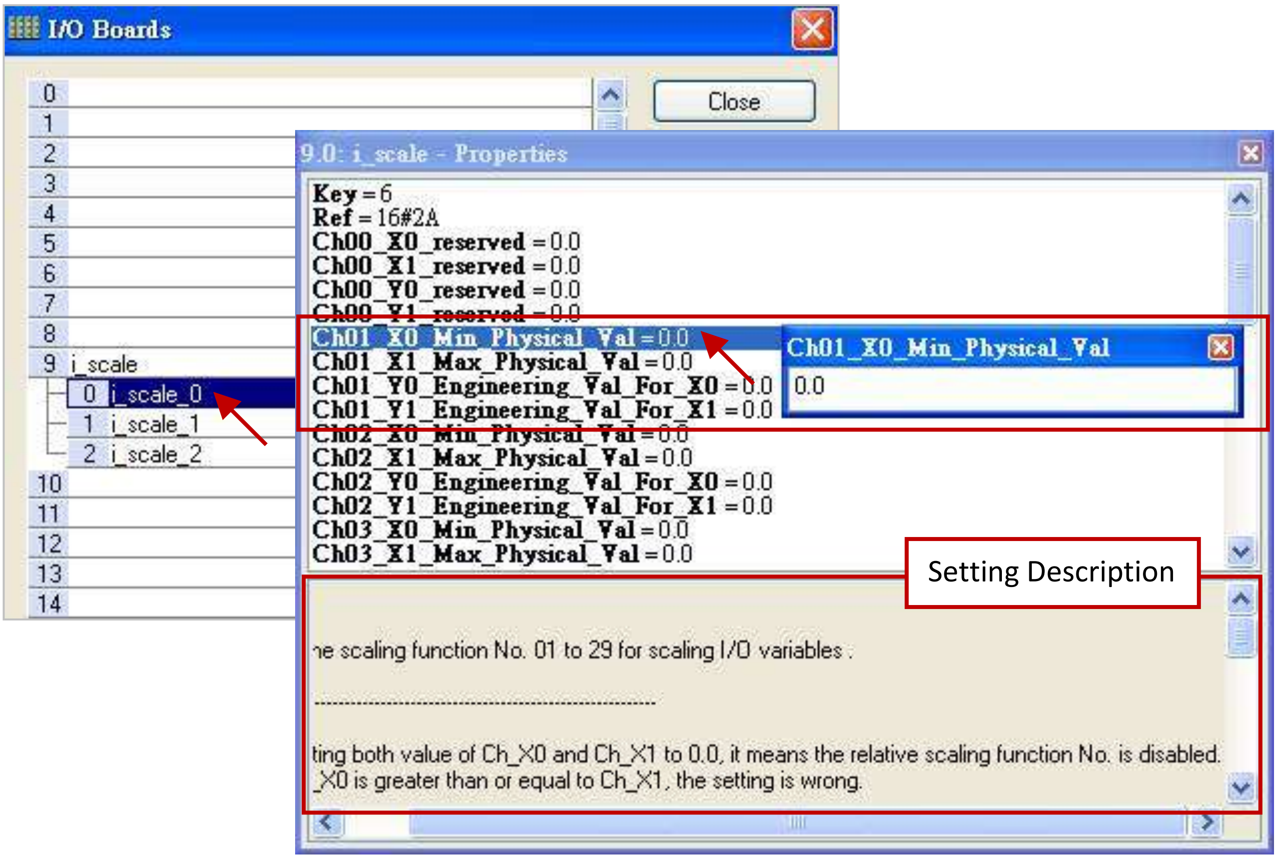
Note:
Double-click this item to modify the variable name and click “Enter” to accept the setting.

⊟ %QX1.1 - i_8055_DO		
%QX1.1.0	BOOL	Output
Output1		Output
%QX1.1.2	BOOL	Output
%QX1.1.3	BOOL	Output

4.2 i_scale (Conversion Table)

The “i_scale” function can set up to 29 scaling functions to convert values for the AI or AO module that plugged in the slot 0 to slot 7. See the Chapter4 (P4-1) to add this I/O board.

1. Mouse double click the "i_scale_0" (or "i_scale_1" or "i_scale_2") to open the “Properties” window, and then to see the setting description.
   Note: Using the slot 8 or above No. because the slot 0 to slot 7 are reserved for the real I/O module.



```
Key = 6
Ref = 16#2A
Ch00_X0_reserved = 0.0
Ch00_X1_reserved = 0.0
Ch00_Y0_reserved = 0.0
Ch00_Y1_reserved = 0.0
Ch01_X0_Min_Physical_Val = 0.0
Ch01_X1_Max_Physical_Val = 0.0
Ch01_Y0_Engineering_Val_For_X0 = 0.0
Ch01_Y1_Engineering_Val_For_X1 = 0.0
Ch02_X0_Min_Physical_Val = 0.0
Ch02_X1_Max_Physical_Val = 0.0
Ch02_Y0_Engineering_Val_For_X0 = 0.0
Ch02_Y1_Engineering_Val_For_X1 = 0.0
Ch03_X0_Min_Physical_Val = 0.0
Ch03_X1_Max_Physical_Val = 0.0
```

Ch01_X0_Min_Physical_Val: 0.0

```
ne scaling function No. 01 to 29 for scaling I/O variables .
------------------------------------------------------------
ting both value of Ch_X0 and Ch_X1 to 0.0, it means the relative scaling function No. is disabled.
_X0 is greater than or equal to Ch_X1, the setting is wrong.
```

Parameters: (“Ch” means the Ch01 to Ch29, "Ch00” is a reserved item)

Ch_X0_Min_Physical_Val: The min. value of AI (or AO) boards (X0).

Ch_X1_Max_Physical_Val: The max. value of AI (or AO) boards (X1).

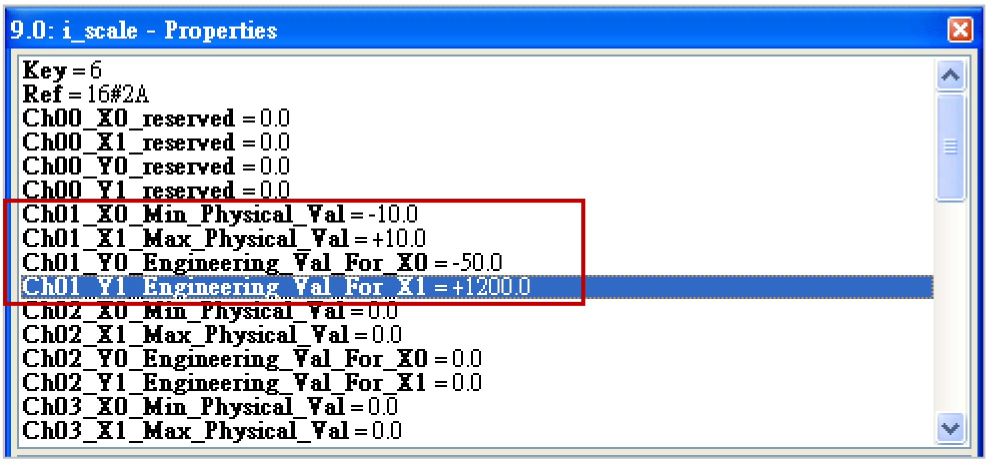
Ch_Y0_Engineering_Val_For_X0: The engineering value after scaling X0.

Ch_Y1_Engineering_Val_For_X1: The engineering value after scaling X1.

2. Double click the item and fill in a value, then press the “Enter” key to complete the setting.
   Notice:
   1. If set both value of Ch_X0 and Ch_X1 to "0.0", it means the relative scaling function No. is disabled.
   2. If Ch_X0 is greater than or equal to Ch_X1, the setting is wrong.
   3. If Ch_Y0 is equal to Ch_Y1, the setting is wrong.

   For example, if the AI board's value is 4 to 20 mA and wish to scale as 0 to 10000, then set Ch_X0 as "4.0", Ch_X1 as "20.0", Ch_Y0 as "0.0", Ch_Y1 as "10000.0".

For example, if the AO board's value is -10 to +10 V and their respective engineering value is -50 to 1200, then set Ch_X0 as "-10.0", Ch_X1 as "+10.0", Ch_Y0 as "-50.0", Ch_Y1 as "+1200.0".

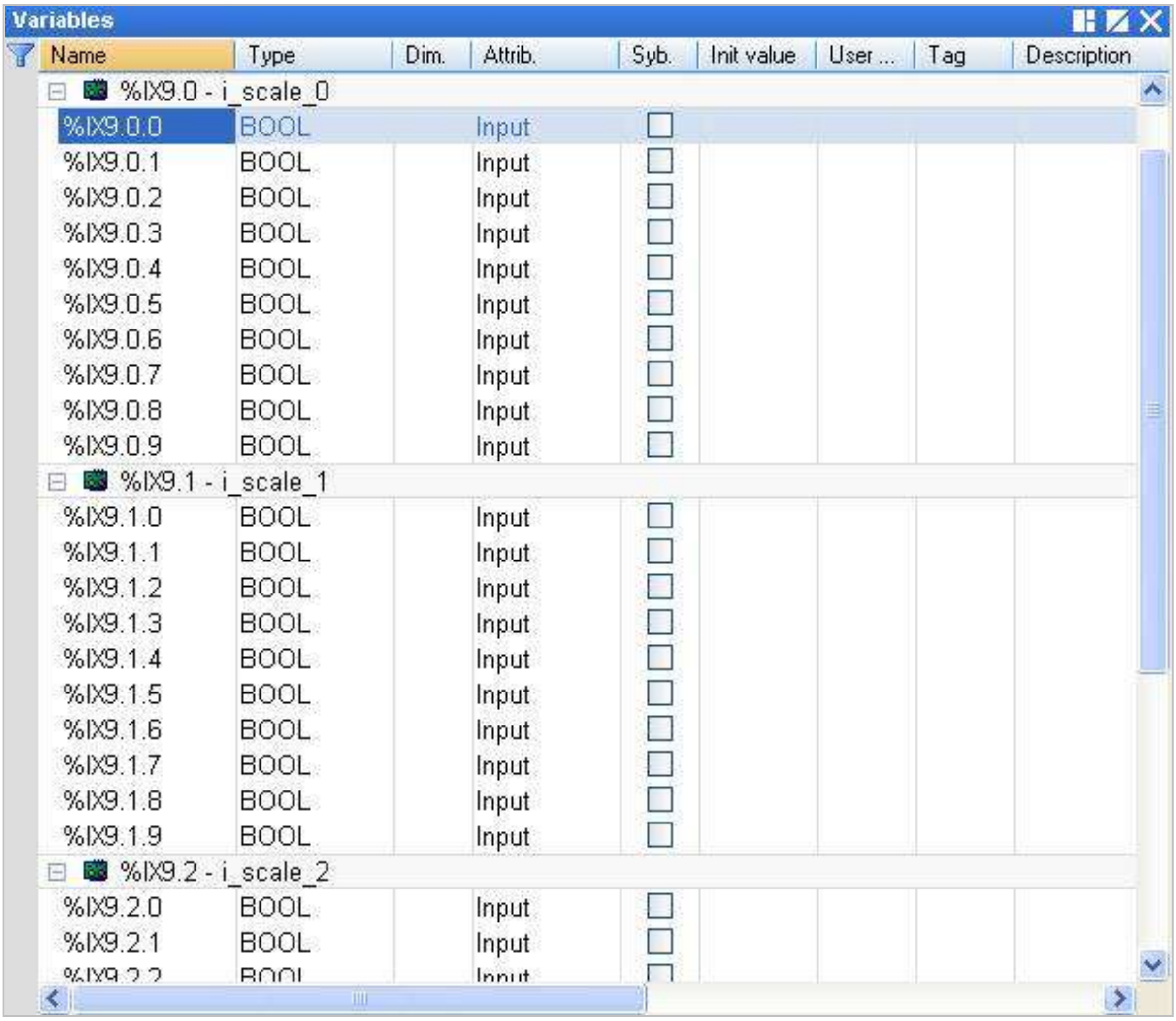


9.0: i_scale - Properties

Key = 6
Ref = 16#2A
Ch00_X0_reserved = 0.0
Ch00_X1_reserved = 0.0
Ch00_Y0_reserved = 0.0
Ch00_Y1_reserved = 0.0
Ch01_X0_Min_Physical_Val = -10.0
Ch01_X1_Max_Physical_Val = +10.0
Ch01_Y0_Engineering_Val_For_X0 = -50.0
Ch01_Y1_Engineering_Val_For_X1 = +1200.0
Ch02_X0_Min_Physical_Val = 0.0
Ch02_X1_Max_Physical_Val = 0.0
Ch02_Y0_Engineering_Val_For_X0 = 0.0
Ch02_Y1_Engineering_Val_For_X1 = 0.0
Ch03_X0_Min_Physical_Val = 0.0
Ch03_X1_Max_Physical_Val = 0.0

3. After linking the “i_scale” in the “I/O Boards” window, it will auto add 30 Boolean variables in the “Variables” window. When the Win-GRAF connects the PAC, it will display the state of each scaling function.

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



Variables

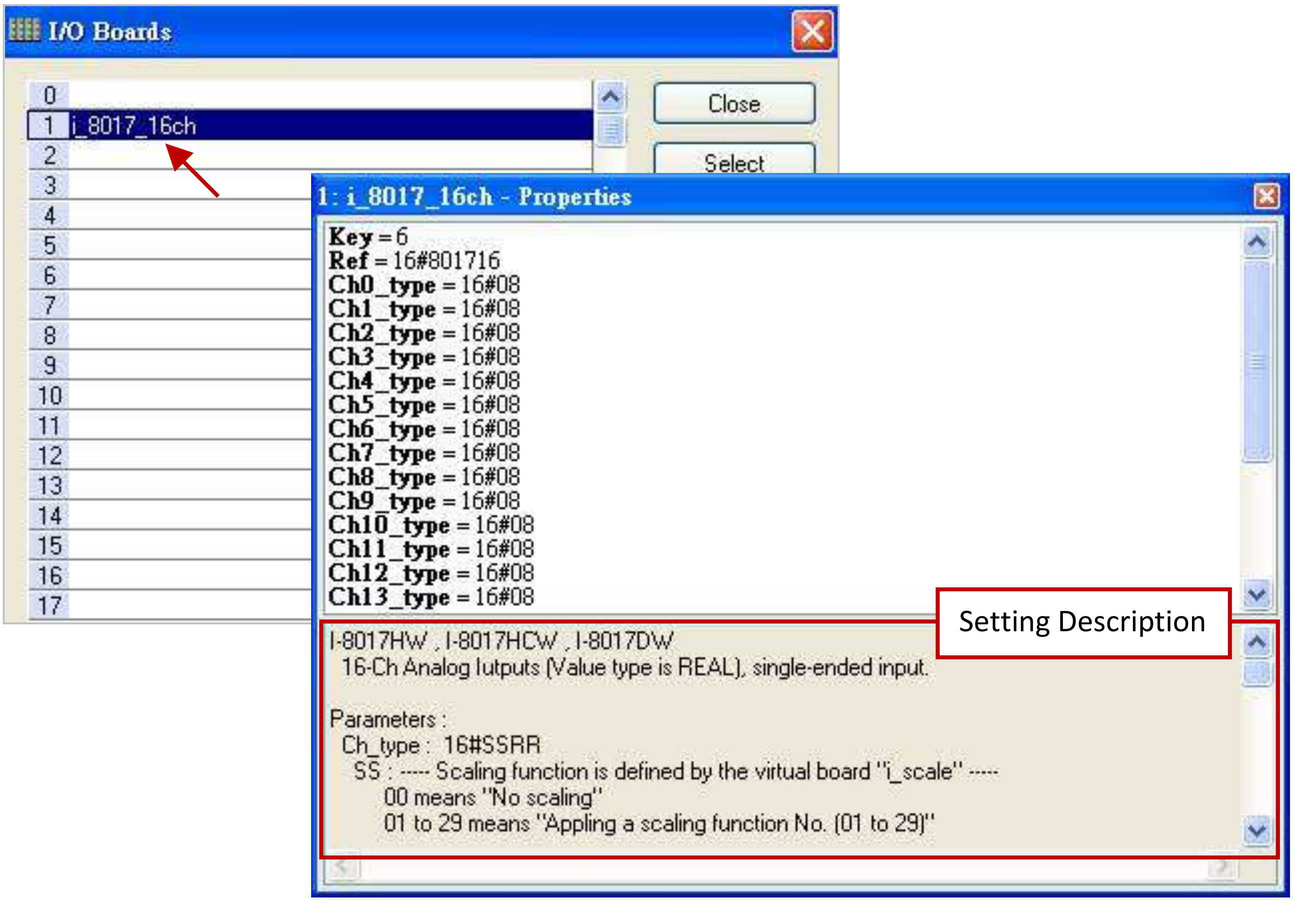
Name	Type	Dim.	Attrib.	Syb.	Init value	User ...	Tag	Description
%IX9.0 - i_scale_0								
%IX9.0.0	BOOL		Input	□				
%IX9.0.1	BOOL		Input	□				
%IX9.0.2	BOOL		Input	□				
%IX9.0.3	BOOL		Input	□				
%IX9.0.4	BOOL		Input	□				
%IX9.0.5	BOOL		Input	□				
%IX9.0.6	BOOL		Input	□				
%IX9.0.7	BOOL		Input	□				
%IX9.0.8	BOOL		Input	□				
%IX9.0.9	BOOL		Input	□				
%IX9.1 - i_scale_1								
%IX9.1.0	BOOL		Input	□				
%IX9.1.1	BOOL		Input	□				
%IX9.1.2	BOOL		Input	□				
%IX9.1.3	BOOL		Input	□				
%IX9.1.4	BOOL		Input	□				
%IX9.1.5	BOOL		Input	□				
%IX9.1.6	BOOL		Input	□				
%IX9.1.7	BOOL		Input	□				
%IX9.1.8	BOOL		Input	□				
%IX9.1.9	BOOL		Input	□				
%IX9.2 - i_scale_2								
%IX9.2.0	BOOL		Input	□				
%IX9.2.1	BOOL		Input	□				
%IX9.2.2	BOOL		Input	□				

4.3 i_8017HW (8/16 channels AI)

The I-8017HW can be an 8-channel differential or a 16-channel single-ended analog input module (Data type: "REAL"). The following will describe a 16-channel module, you can see the Chapter4 (P4-1) to add this I/O board.

Note: Before using the I-8017HW I/O module, it requires to set the Differential or Single-ended Jumper in the hardware.

1. Mouse double click the "i_8017_16ch" to open the "Properties" window, and then to see the setting description.



```
Key = 6
Ref = 16#801716
Ch0_type = 16#08
Ch1_type = 16#08
Ch2_type = 16#08
Ch3_type = 16#08
Ch4_type = 16#08
Ch5_type = 16#08
Ch6_type = 16#08
Ch7_type = 16#08
Ch8_type = 16#08
Ch9_type = 16#08
Ch10_type = 16#08
Ch11_type = 16#08
Ch12_type = 16#08
Ch13_type = 16#08
```

```
I-8017HW , I-8017HCW , I-8017DW
  16-Ch Analog Iutputs (Value type is REAL), single-ended input.

Parameters :
  Ch_type :  16#SSRR
    SS : ----- Scaling function is defined by the virtual board "i_scale" -----
         00 means "No scaling"
         01 to 29 means "Appling a scaling function No. (01 to 29)"
```

Parameters:

Ch_type: 16#SSRR

SS : Scaling function is defined by the "i_scale" I/O board (see the Section 4.2).

00 means "No scaling".

01 to 29 means "Appling a scaling function No. (01 to 29)".

Setting "SS" as other value will use the default value 00.

RR : Range definition of signals.

05 means "physical input signal is -2.5 to +2.5 Volt".

06 means "physical input signal is -20 to +20 mA".

07 means "physical input signal is -1.25 to +1.25 Volt".

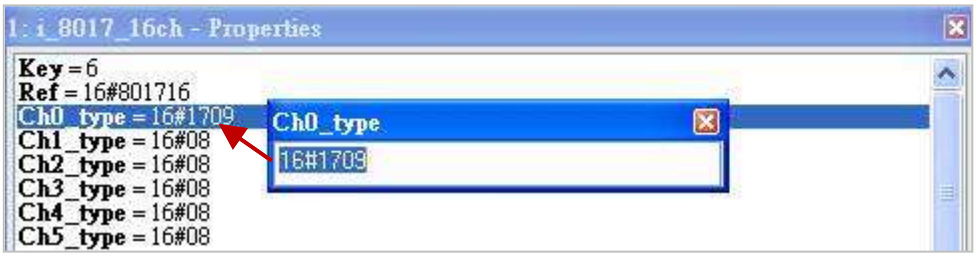
08 means "physical input signal is -10 to +10 Volt".

09 means "physical input signal is -5 to +5 Volt".

Setting "RR" as other value will use the default value 08.

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

For example, 16#08 means the physical input signal is -10 to +10 V.
Channel value 5.67 means the input signal is 5.67 V.
For example, 16#209 means the physical input signal is -5 to +5 V with the scaling function 2.
Signal 5.67 V will be scaled to an engineering value by the scaling function 2.
For example, 16#1709 means the physical input signal is -5 to +5 V with the scaling function 17.
Signal 5.67 V will be scaled to an engineering value by the scaling function 17.



1: i_8017_16ch - Properties

Key = 6
Ref = 16#801716
Ch0_type = 16#1709
Ch1_type = 16#08
Ch2_type = 16#08
Ch3_type = 16#08
Ch4_type = 16#08
Ch5_type = 16#08

Ch0_type
16#1709

Noise_Filter_Max: The max. of the physical value to be considered as noise.
The filter will filter out the signal value beyond it, default setting is "9999.9".
For example, set as “7.9”, signal larger than 7.9 V (or 7.9 mA) will be filtered out.

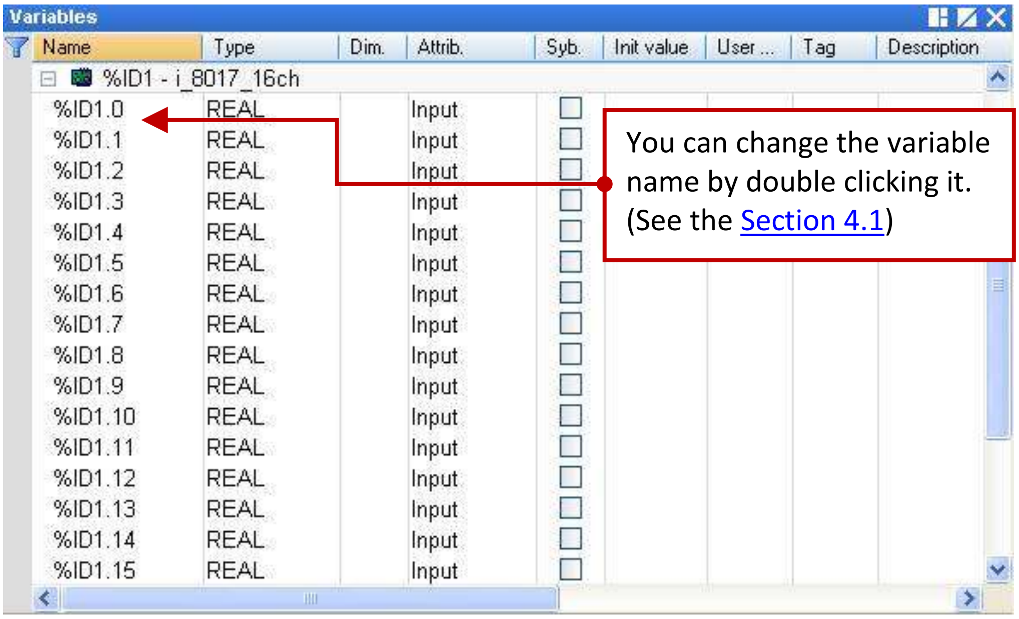
Noise_Filter_Min: The min. of the physical value to be considered as noise.
The filter will filter out the signal value beyond it, default setting is "-9999.9".
For example, set as “1.5”, signal smaller than 1.5 V (or 1.5 mA) will be filtered out.

Note:

If setting Noise_Filter_Min >= Noise_Filter_Max, filter is disabled.
If setting Noise_Filter_Min < -1000 and Noise_Filter_Max > 1000, the filter is disabled.

Sample_Number: The number of sampled data to be averaged as one data.
Default is “1” (range: 1 to 500). Set a bigger value will reduce the sampling rate, however the signal curve is smoother than setting a small value.

3. After linking the “i_8017_16ch” in the “I/O Boards” window, it will auto add 16 "REAL” input variables in the “Variables” window. These variables can be used in the program.



Variables

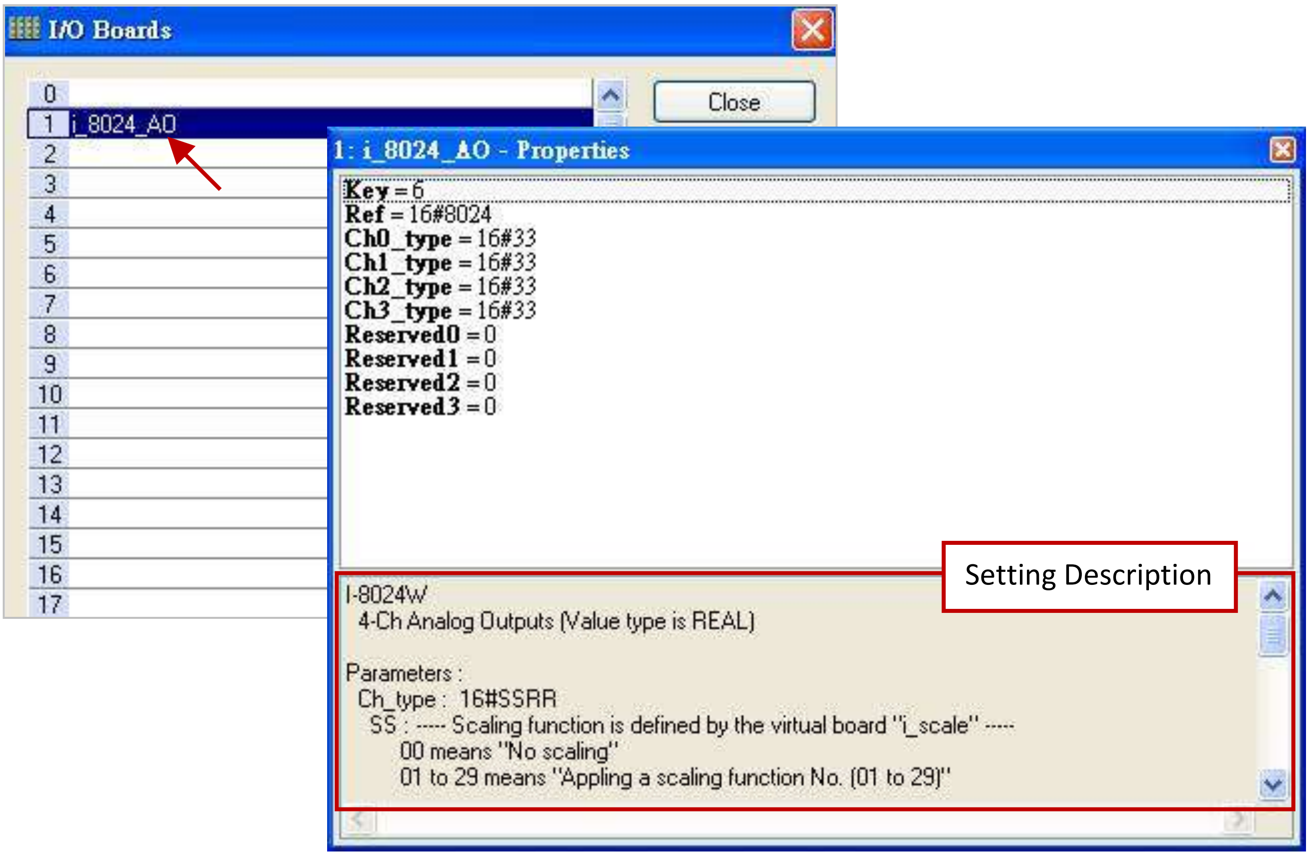
Name	Type	Dim.	Attrib.	Syb.	Init value	User ...	Tag	Description
%ID1 - i_8017_16ch								
%ID1.0	REAL		Input	□				
%ID1.1	REAL		Input	□				
%ID1.2	REAL		Input	□				
%ID1.3	REAL		Input	□				
%ID1.4	REAL		Input	□				
%ID1.5	REAL		Input	□				
%ID1.6	REAL		Input	□				
%ID1.7	REAL		Input	□				
%ID1.8	REAL		Input	□				
%ID1.9	REAL		Input	□				
%ID1.10	REAL		Input	□				
%ID1.11	REAL		Input	□				
%ID1.12	REAL		Input	□				
%ID1.13	REAL		Input	□				
%ID1.14	REAL		Input	□				
%ID1.15	REAL		Input	□				

You can change the variable name by double clicking it. (See the Section 4.1)

4.4 i_8024 (4-channel AO)

The I-8024W is a 4-channel analog output module (Data type: "REAL") that can be used to output +/- 10 V or 0 to +20 mA signal. See the Chapter4 (P4-1) to add this I/O board.

1. Mouse double-click the "i_8024_AO" to open the “Properties” window and then to see the setting description.



Key = 6
Ref = 16#8024
Ch0_type = 16#33
Ch1_type = 16#33
Ch2_type = 16#33
Ch3_type = 16#33
Reserved0 = 0
Reserved1 = 0
Reserved2 = 0
Reserved3 = 0

I-8024W
4-Ch Analog Outputs (Value type is REAL)

Parameters :
Ch_type : 16#SSRR
SS : ----- Scaling function is defined by the virtual board "i_scale" -----
00 means "No scaling"
01 to 29 means "Appling a scaling function No. (01 to 29)"

Parameters:

Ch_type: 16#SSRR

SS: Scaling function is defined by the "i_scale" I/O board (see the Section 4.2).

00 means "No scaling".

01 to 29 means "Appling a scaling function No. (01 to 29)".

Setting “SS” as other value will use the default value 00.

RR: Range definition of signals

30 means "physical output signal is 0 to 20 mA"

33 means "physical output signal is -10 to +10 Volt"

Setting “RR” as other value will use the default value 33.

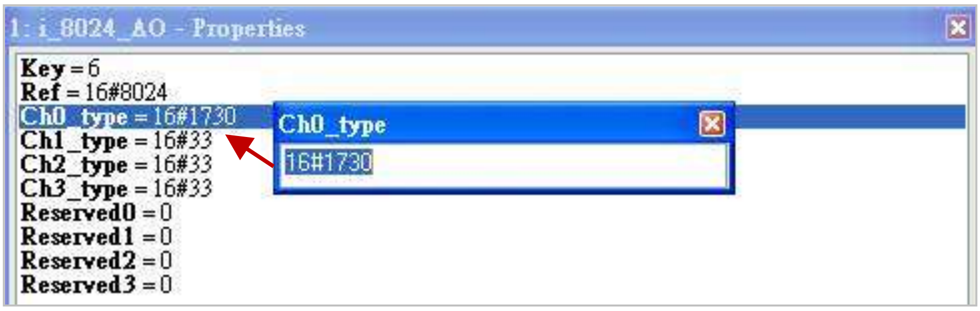
2. Double click the item and fill in a value, then press the “Enter” key to complete the setting.

For example, 16#33 means the physical output signal is -10 to +10 V.
Channel value 5.67 is to output 5.67 V ; value -3.752 is to output -3.752 V.

For example, 16#133 means the physical output signal is -10 to +10 V with the scaling function 1.
Channel value is a user-defined engineering value will be converted first by the scaling function 1 (i.e., "Ch01") before output it as -10 to +10 V.

For example, 16#30 means the physical output signal is 0 to 20 mA.
Channel value 12.5 is to output 12.5 mA ; value 6.27 is to output 6.27 mA.

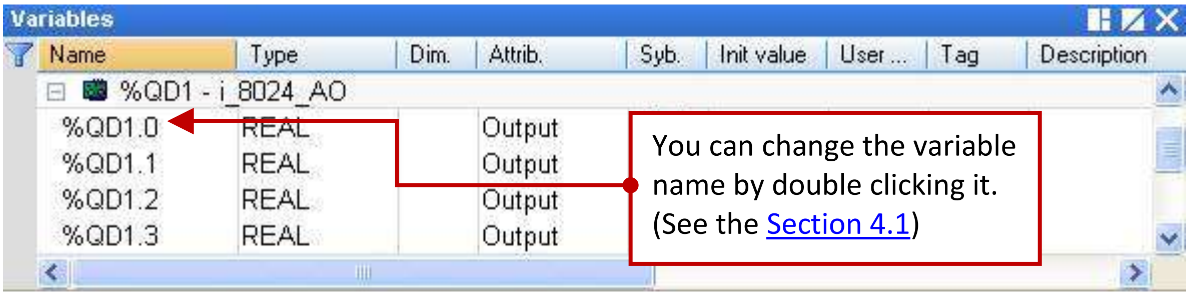
For example, 16#1730 means the physical output signal is 0 to 20 mA with the scaling function 17.
Channel value is a user-defined engineering value will be converted first by the scaling function 17 (i.e., "Ch17") before output it as 0 to 20 mA.



Key = 6
Ref = 16#8024
Ch0_type = 16#1730
Ch1_type = 16#33
Ch2_type = 16#33
Ch3_type = 16#33
Reserved0 = 0
Reserved1 = 0
Reserved2 = 0
Reserved3 = 0

16#1730

3. After linking the “i_8024_AO” in the “I/O Boards” window, it will auto add 4 "REAL” Output variables in the “Variables” window. These variables can be used in the program.



Name	Type	Dim.	Attrib.	Syb.	Init value	User ...	Tag	Description
%QD1 - i_8024_AO								
%QD1.0	REAL		Output					
%QD1.1	REAL		Output					
%QD1.2	REAL		Output					
%QD1.3	REAL		Output					

You can change the variable name by double clicking it. (See the Section 4.1)

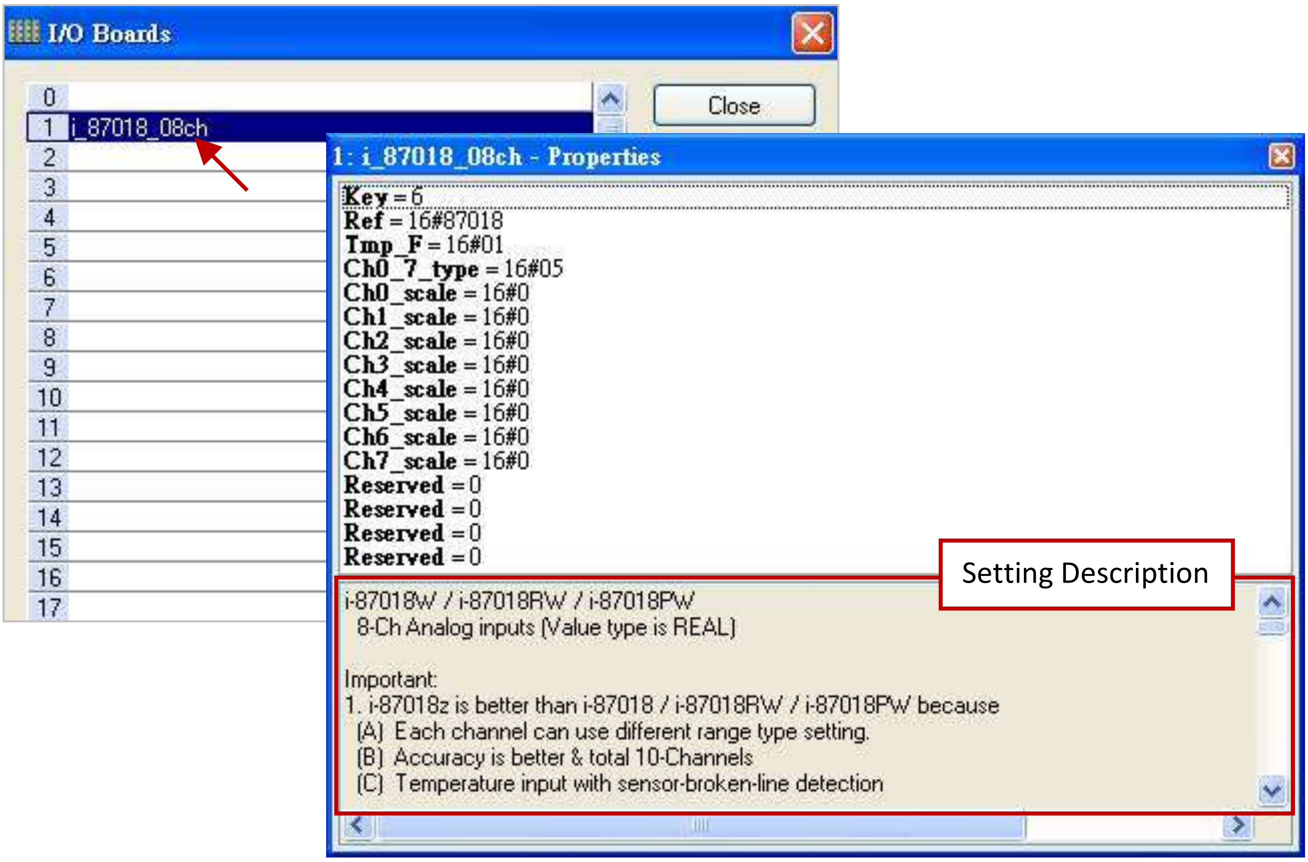
4.5 i_87018W (8-channel AI)

The I-87018W is an 8-channel analog input module (Data type: "REAL") that provides thermocouple input, current input (-20 mA to +20 mA) and voltage input (+/- 15 mV, +/- 50 mV, +/- 100 mV, +/- 500 mV, +/- 1 V, +/- 2.5 V). See the Chapter4 (P4-1) to add this I/O board.

Important Notice:

- I-87018ZW is better than I-87018W / I-87018RW / I-87018PW because
  (A) Each channel can use different range type setting.
  (B) Accuracy is better and total 10-Channels.
  (C) Temperature input with sensor-broken-line detection.
  Please visit http://www.icpdas.com/products/Remote_IO/i-87k/i-87018z.htm
- I-87018W does not support sensor-broken line function.

1. Mouse double-click the "i_8018_08ch" to open the “Properties” window, and then to see the setting description.



Key = 6
Ref = 16#87018
Tmp_F = 16#01
Ch0_7_type = 16#05
Ch0_scale = 16#0
Ch1_scale = 16#0
Ch2_scale = 16#0
Ch3_scale = 16#0
Ch4_scale = 16#0
Ch5_scale = 16#0
Ch6_scale = 16#0
Ch7_scale = 16#0
Reserved = 0
Reserved = 0
Reserved = 0
Reserved = 0

i-87018W / i-87018RW / i-87018PW
8-Ch Analog inputs (Value type is REAL)

Important:
1. i-87018z is better than i-87018 / i-87018RW / i-87018PW because
(A) Each channel can use different range type setting.
(B) Accuracy is better & total 10-Channels
(C) Temperature input with sensor-broken-line detection

Parameters:

Tmp_F: 16#FF

FF: Temperature format, It only apply to the channel type is Thermocouple.

01 means the unit of the input value is Degree Celsius

02 means the unit of the input value is Degree Fahrenheit

Ch0_7_type: 16#RR

RR: Range definition of signals.

Normal Range: (For mA or Volt)

Type Code	Physical Input Signal
00	-0.015 to +0.015 V
01	-0.05 to +0.05 V
02	-0.1 to +0.1 V
03	-0.5 to +0.5 V
04	-1 to +1 V
05	-2.5 to +2.5 V
06	20 to +20 mA

When I-87018 and I-87018R are connected to a current source and set to “06” type code, an optional external 125 Ohm resistor is required.

Thermocouple Range: (For temperature)

Type Code	Type	Physical Input Signal
0E	J	-210 to +760 °C
0F	K	-270 to +1372 °C
10	T	-270 to +400 °C
11	E	-270 to +1000 °C
12	R	0 to +1768 °C
13	S	0 to +1768 °C
14	B	0 to +1820 °C
15	N	0 to +2320 °C
17	L	-200 to +800 °C
18	M	-200 to +100 °C
19	$\mathrm{L}_{\mathrm{DIN43710}}$	-200 to +900 °C

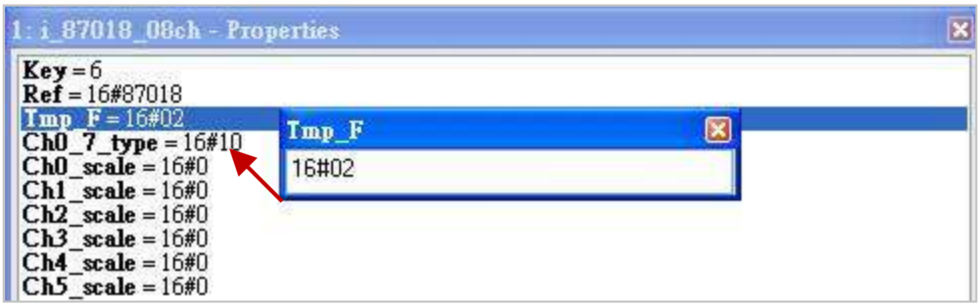
Setting RR as other value will use the default value “05”

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

    For example, 16#05 means the physical input signal is -2.5 to +2.5 V.
    Channel value 1.28 means the input signal is 1.28 V.
    Channel value -0.752 means the input signal is -0.752 V.

    For example, If "Ch0_7_type" set as 16#0F and "Tmp_F" set as 16#01 means the physical input signal is -270 to +1372 degree Celsius.
    Channel value 25.75 means 25.75 degree Celsius.

    For example, If "Ch0_7_type" set as 16#10 and "Tmp_F" set as 16#01 means the physical input signal is -454 to +752 degree Fahrenheit.
    Channel value 25.75 means 25.75 degree Fahrenheit.


1: i_87018_08ch - Properties

Key = 6
Ref = 16#87018
Tmp_F = 16#02
Ch0_7_type = 16#10
Ch0_scale = 16#0
Ch1_scale = 16#0
Ch2_scale = 16#0
Ch3_scale = 16#0
Ch4_scale = 16#0
Ch5_scale = 16#0

Tmp_F

16#02

Note: If using a temperature module with a broken-line detection function (e.g., I-87018ZW), and the temperature value is greater than "9000.0", it means,

1. The temperature sensor may be broken-line.
2. The temperature sensor may be damaged.
3. The DCON module is not configured well to fit the connected temperature sensors.
4. The ohm measured by the connected sensor is not correct.

Ch0_scale to Ch7_scale: 16#SS

SS: Scaling function is defined by the "i_scale" I/O board (refer the Section 4.2).

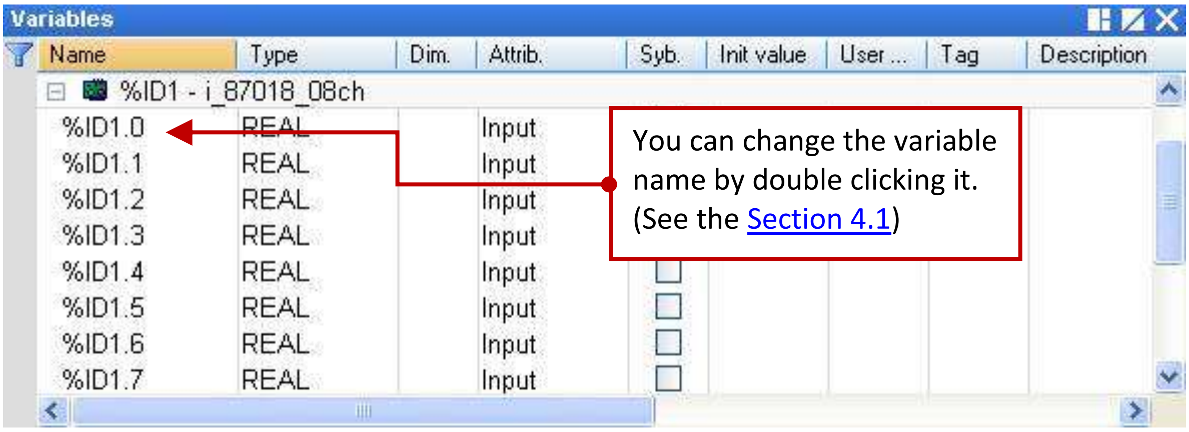
00 means "No scaling".

01 to 29 means "Appling a scaling function No. (01 to 29)"

Setting SS as other value will use the default value 00.

For example, 16#17 means the physical input signal is converted with the scaling function 17.

3. After linking the “i_87018_08ch” in the “I/O Boards” window, it will auto add 8 "REAL” input variables in the “Variables” window that are available for programing.



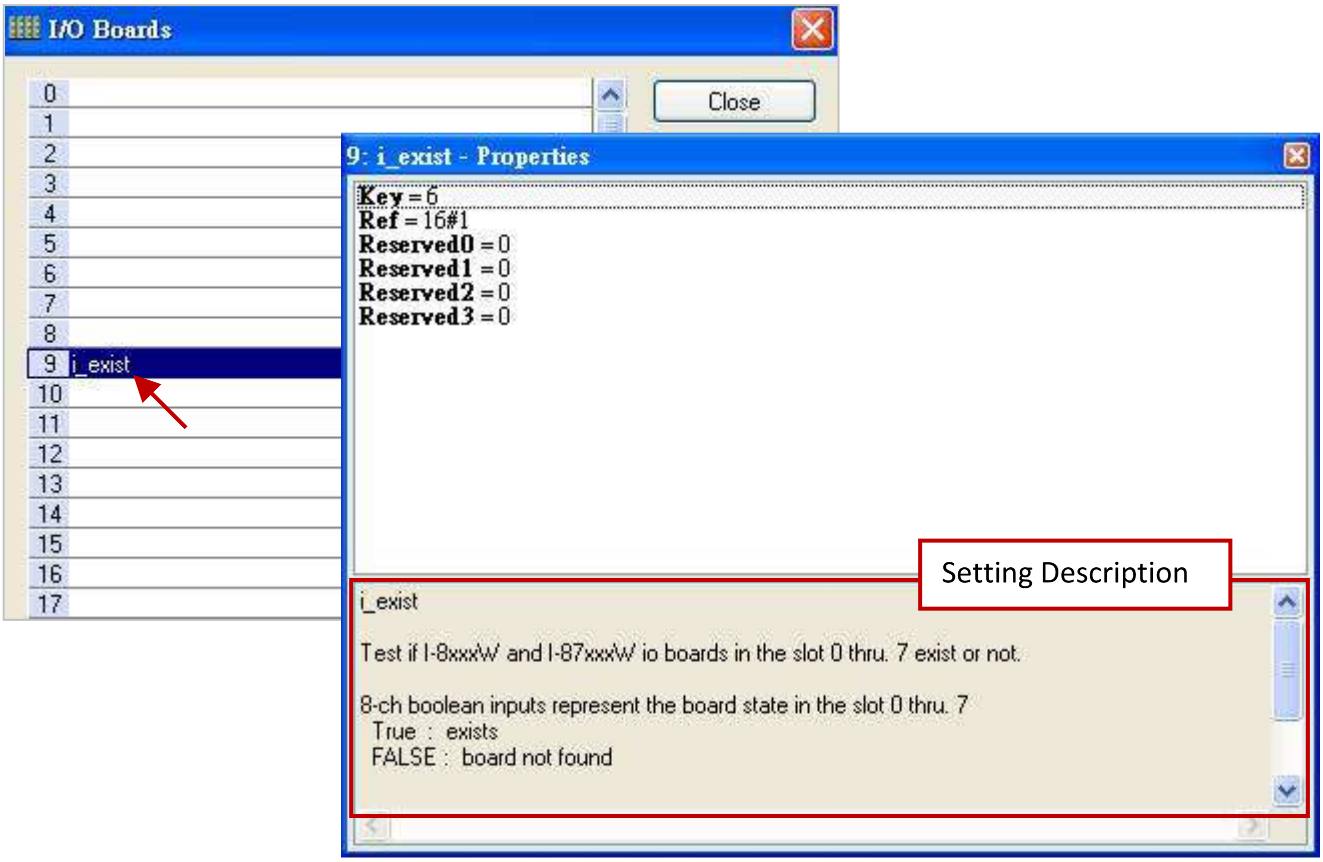
Name	Type	Dim.	Attrib.	Syb.	Init value	User ...	Tag	Description
%ID1 - i_87018_08ch								
%ID1.0	REAL		Input					
%ID1.1	REAL		Input					
%ID1.2	REAL		Input					
%ID1.3	REAL		Input					
%ID1.4	REAL		Input	□				
%ID1.5	REAL		Input	□				
%ID1.6	REAL		Input	□				
%ID1.7	REAL		Input	□				

You can change the variable name by double clicking it. (See the Section 4.1)

4.6 i_exist (Test if the I/O module exists?)

The “i_exist” is used to check if the I-8K and I-87K series I/O modules exist in the PAC‘s slot 0 to 7. See the Chapter4 (P4-1) to add this I/O board.

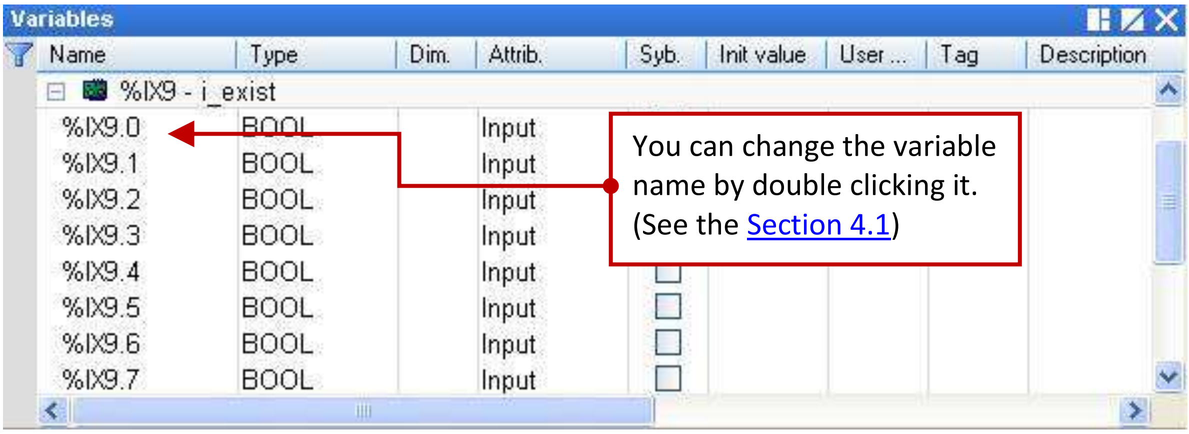
1. Mouse double-click the "i_exist" to open the “Properties” window, and then to see the setting description.
   Note: Using the Slot 8 or above No. because the slot 0 to slot 7 are reserved for the real I/O module.



```
Key = 6
Ref = 16#1
Reserved0 = 0
Reserved1 = 0
Reserved2 = 0
Reserved3 = 0
```

```
i_exist

Test if I-8xxxW and I-87xxxW io boards in the slot 0 thru. 7 exist or not.

8-ch boolean inputs represent the board state in the slot 0 thru. 7
  True  :  exists
  FALSE :  board not found
```

2. After linking the “i_exist” in the “I/O Boards” window, it will auto add 8 "BOOL” input variables in the “Variables” window and display the state of the I/O module from slot 0 to slot 7 when connecting the Win-GRAF PAC.
   "TRUE" means the I/O module exists.
   "FALSE" means can not find this I/O module.



Name	Type	Dim.	Attrib.	Syb.	Init value	User ...	Tag	Description
⊟ %IX9 - i_exist								
%IX9.0	BOOL		Input					
%IX9.1	BOOL		Input					
%IX9.2	BOOL		Input					
%IX9.3	BOOL		Input					
%IX9.4	BOOL		Input	□				
%IX9.5	BOOL		Input	□				
%IX9.6	BOOL		Input	□				
%IX9.7	BOOL		Input	□				

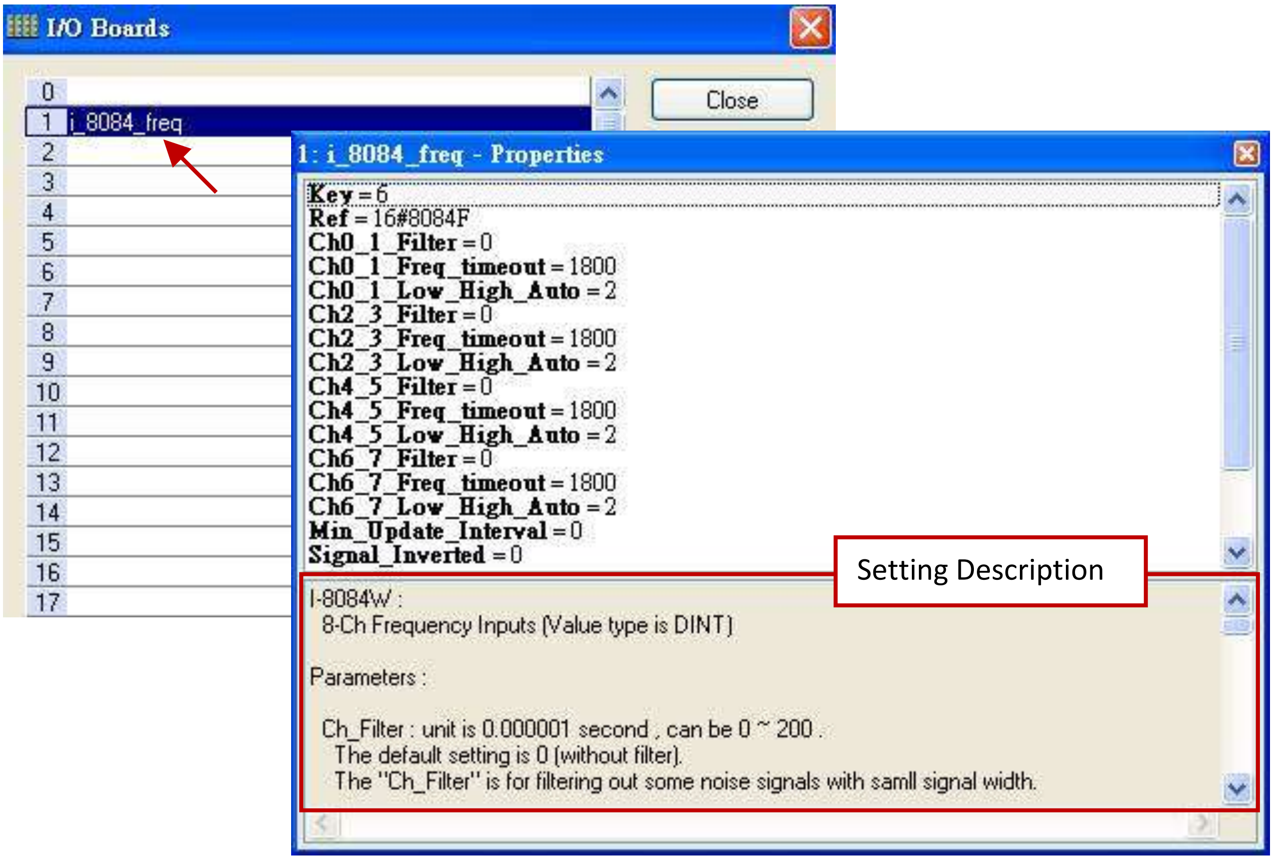
You can change the variable name by double clicking it. (See the Section 4.1)

4.7 i_8084 (Frequency, UP/Down Counter, UP Counter)

The I-8084W is a 4/8-channel high speed Frequency/Counter (Data type: “DINT”) that can be used to measure frequency or as a UP/Down Counter or as a UP Counter. The following will describe these three modes. See the Chapter4 (P4-1) to add this I/O board.

4.7.1 i_8084_freq (8-channel Frequency)

1. Mouse double-click the "i_8084_freq" to open the “Properties” window, and then to see the setting description.



```
Key = 6
Ref = 16#8084F
Ch0_1_Filter = 0
Ch0_1_Freq_timeout = 1800
Ch0_1_Low_High_Auto = 2
Ch2_3_Filter = 0
Ch2_3_Freq_timeout = 1800
Ch2_3_Low_High_Auto = 2
Ch4_5_Filter = 0
Ch4_5_Freq_timeout = 1800
Ch4_5_Low_High_Auto = 2
Ch6_7_Filter = 0
Ch6_7_Freq_timeout = 1800
Ch6_7_Low_High_Auto = 2
Min_Update_Interval = 0
Signal_Inverted = 0
```

Setting Description

```
I-8084W :
 8-Ch Frequency Inputs (Value type is DINT)

Parameters :

 Ch_Filter : unit is 0.000001 second , can be 0 ~ 200 .
  The default setting is 0 (without filter).
  The "Ch_Filter" is for filtering out some noise signals with samll signal width.
```

Parameters:

Ch_Filter: The unit is 0.000001 second (μs), the value can be 0 to 200.
The default setting is 0 (without filter). The "Ch_Filter" is for filtering out some noise signals with smaller signal width. (Recommend 0: if there is no noise consideration or need a real-time measurement.) The following setting is recommended:

Max Input Signal (Hz)	Recommend Filter Value
1K	200
2K	100
5K	40
10K	20
20K	10
100K	2
450K	1
450K	0 (without filter)

Ch_Freq_Timeout:

The unit is 0.001 second (ms), the value can be 20 to 1800. Set as other value will use the default value 1800. If there is no signal wave input to the I-8084W in the "Ch_Freq_Timeout" interval, the frequency value of the related channel will be assigned as 0.

For example, if set it as 100 ms and the input is 500 Hz (that means, one signal wave takes about 2 ms to happen), the frequency is updated normally. When the input frequency drop to 9 Hz (that means, one signal wave take about 111 ms to happen), this "111" exceeds the setting "100" ms (Freq_Timeout). So the frequency value will be assigned as 0 because there is no signal wave coming in this 100 ms interval.

When setting as 20 ms, the frequency value below 50 Hz is not detectable (become 0).
When setting as 100 ms, the frequency value below 10 Hz is not detectable (become 0).
When setting as 1800 ms, the frequency value of 0 Hz , 1 Hz to 450 KHz is detectable.

Ch_Low_High_Auto: (recommend setting as "2: Auto".)

0 means a Low frequency mode.
1 means a High frequency mode.
2 means Auto switching between Low and High frequency mode.
Set as other value will use the default value "2: Auto".

Mode 2 will auto change the frequency mode. It will auto change to High mode when the input frequency is larger than 3500 Hz, while auto change to Low mode if input frequency is less than 1000 Hz.

DO NOT set as 1 (High frequency mode) if the input signal is normally less than 1000 Hz, or the frequency value will be incorrect frequently. Recommended don't set as 0 (Low frequency mode) if the input signal is normally larger than 3500 Hz.

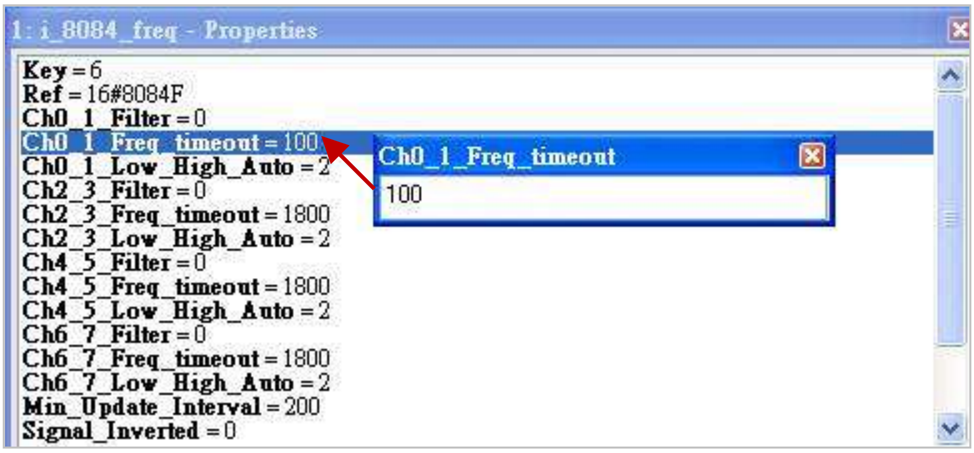
Min_Update_Interval:

The unit is 0.001 second (ms), the value can be 0, or 20 to 1000.
Default value 0 means "Update frequency every PAC cycle".
Other means "Update frequency when each Interval time reached".
The frequency update time also depends on the Win-GRAF PAC cycle time. If the PAC cycle time is big, for example 200 ms, then the real frequency update time will become 200 ms when setting the "Min_Update_Interval" less than 200. Setting bigger "Min_Update_Interval" will get smooth frequency curve value, however the frequency value is updated slowly.

Signal_Inverted:

0 : input signal is normal (no inverted).
1 : input signal is inverted (means voltage HIGH will be processed as LOW, and voltage LOW will be processed as HIGH).

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



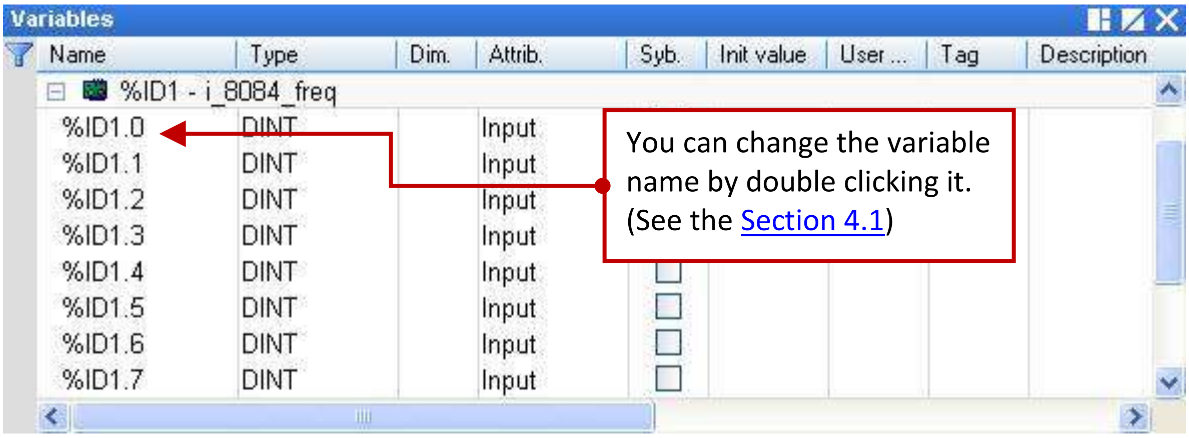
1: i_8084_freq - Properties

```
Key = 6
Ref = 16#8084F
Ch0_1_Filter = 0
Ch0_1_Freq_timeout = 100
Ch0_1_Low_High_Auto = 2
Ch2_3_Filter = 0
Ch2_3_Freq_timeout = 1800
Ch2_3_Low_High_Auto = 2
Ch4_5_Filter = 0
Ch4_5_Freq_timeout = 1800
Ch4_5_Low_High_Auto = 2
Ch6_7_Filter = 0
Ch6_7_Freq_timeout = 1800
Ch6_7_Low_High_Auto = 2
Min_Update_Interval = 200
Signal_Inverted = 0
```

Ch0_1_Freq_timeout

100

3. After linking the “i_8084_freq” in the “I/O Boards” window, it will auto add 8 “DINT” input variables in the “Variables” window. When the Win-GRAF connects the PAC, it will display the frequency value for each channel.



Variables

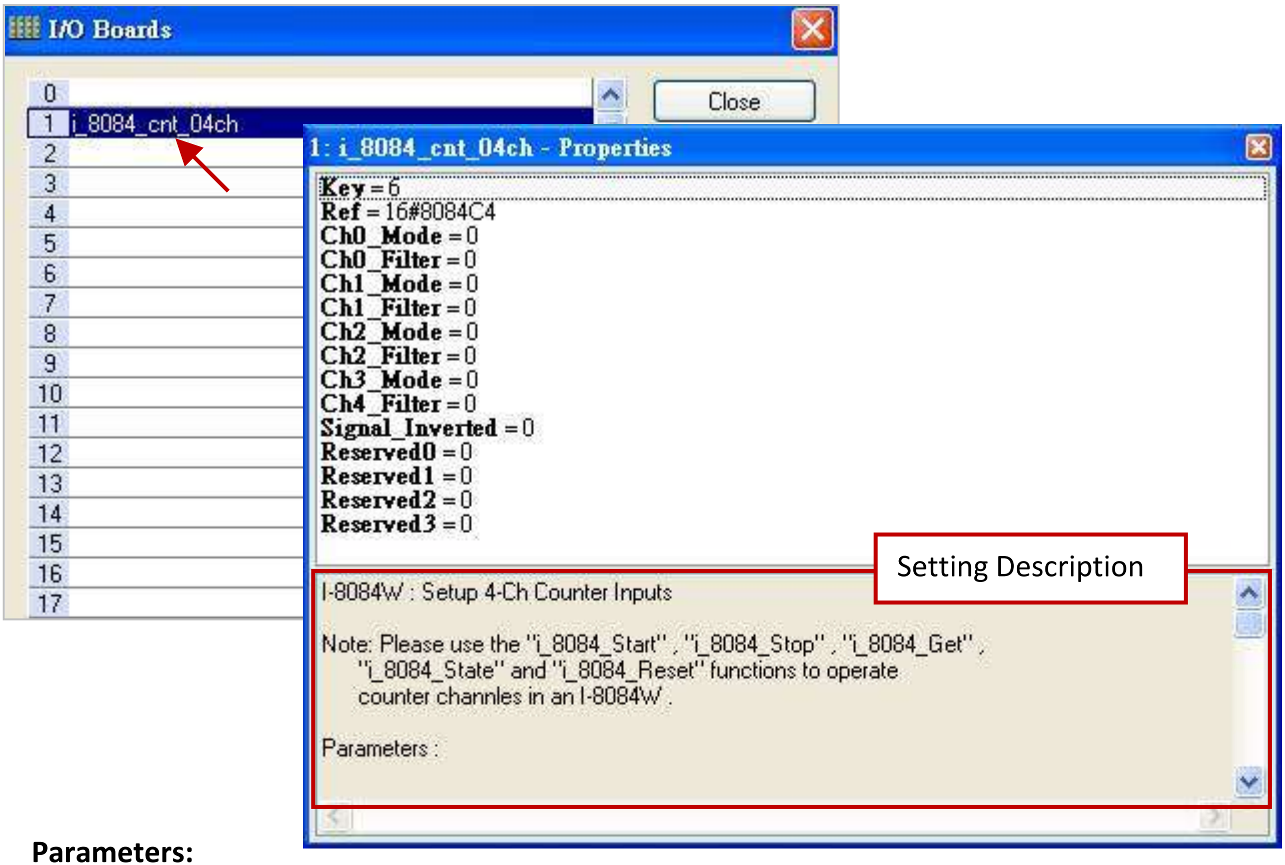
Name	Type	Dim.	Attrib.	Syb.	Init value	User ...	Tag	Description
%ID1 - i_8084_freq								
%ID1.0	DINT		Input					
%ID1.1	DINT		Input					
%ID1.2	DINT		Input					
%ID1.3	DINT		Input					
%ID1.4	DINT		Input	□				
%ID1.5	DINT		Input	□				
%ID1.6	DINT		Input	□				
%ID1.7	DINT		Input	□				

You can change the variable name by double clicking it. (See the Section 4.1)

4.7.2 i_8084_cnt_ch04 (4-channel UP/Down Counter)

Note: Using the "COUNTER_START", "COUNTER_STOP", "COUNTER _GET", "COUNTER_STATE" and "COUNTER_RESET" functions in the Win-GRAF Workbench to operate counter channels in an I-8084W.

1. Mouse double-click the "i_8084_cnt_ch04" to open the "Properties" window, and then to see the setting description.



Key = 6
Ref = 16#8084C4
Ch0_Mode = 0
Ch0_Filter = 0
Ch1_Mode = 0
Ch1_Filter = 0
Ch2_Mode = 0
Ch2_Filter = 0
Ch3_Mode = 0
Ch4_Filter = 0
Signal_Inverted = 0
Reserved0 = 0
Reserved1 = 0
Reserved2 = 0
Reserved3 = 0

I-8084W : Setup 4-Ch Counter Inputs

Note: Please use the "i_8084_Start" , "i_8084_Stop" , "i_8084_Get" , "i_8084_State" and "i_8084_Reset" functions to operate counter channles in an I-8084W .

Parameters :

Parameters:

Ch_Mode: Input mode, can be 0 , 1 and 4. Set other value will use 0.

0: Pulse/DIR mode.

1: UP/DOWN mode.

4: A/B phase (Quard.) mode.

Ch_Filter: The unit is 0.000001 second (μs), the value can be 0 to 200.

The default setting is 0 (without filter). The "Ch_Filter" is for filtering out some noise signals with smaller signal width. (Recommend 0: if there is no noise consideration or need a real-time measurement.) The following setting is recommended:

Max Input Signal (Hz)	Recommend Filter Value
1K	200
2K	100
5K	40
10K	20
20K	10
100K	2
450K	1
450K	0 (without filter)

Signal_Inverted:

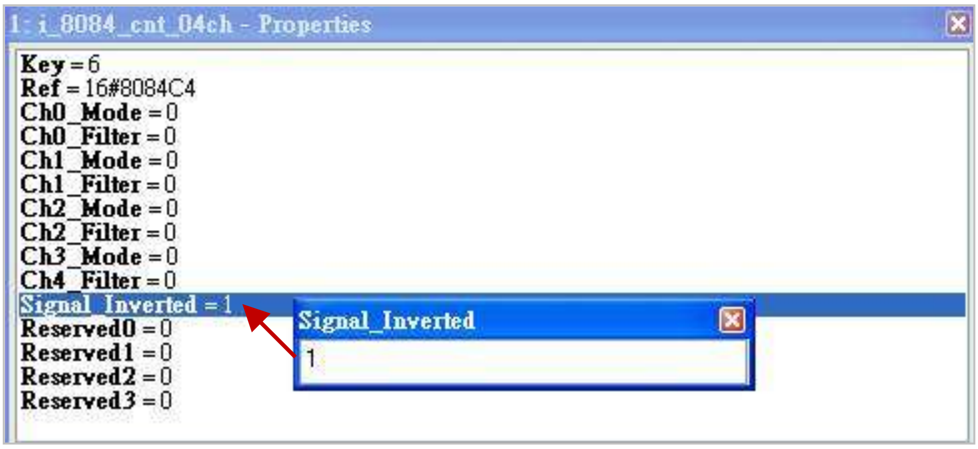
0: Input signal is normal (no inverted).

1: Input signal is inverted (means voltage HIGH will be processed as LOW, and voltage LOW will be processed as HIGH).

For example:

If setting "Signal_Inverted" as 0 (no inverted) and Ch_Mode is 0 (Pulse/DIR),
the counter value will count up if "DIR" signal is High.
If setting "Signal_Inverted" as 1 (inverted) and Ch_Mode is 0 (Pulse/DIR),
the counter value will count down if "DIR" signal is High.

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



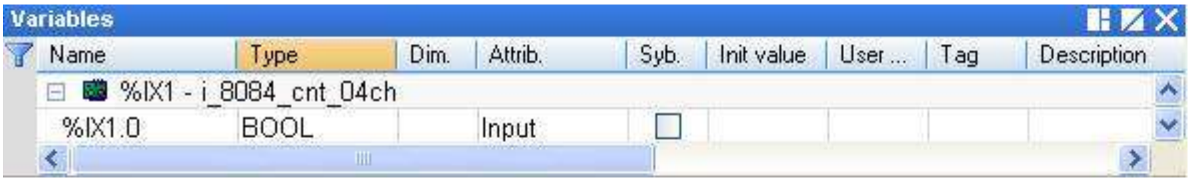
1: i_8084_cnt_04ch - Properties

Key = 6
Ref = 16#8084C4
Ch0_Mode = 0
Ch0_Filter = 0
Ch1_Mode = 0
Ch1_Filter = 0
Ch2_Mode = 0
Ch2_Filter = 0
Ch3_Mode = 0
Ch4_Filter = 0
Signal_Inverted = 1
Reserved0 = 0
Reserved1 = 0
Reserved2 = 0
Reserved3 = 0

Signal_Inverted

1

3. After linking the “i_8084_cnt_ch04” in the “I/O Boards” window, it will auto add one “BOOL” Input variable (no meaning, always “FALSE”) in the “Variables” window.



Variables

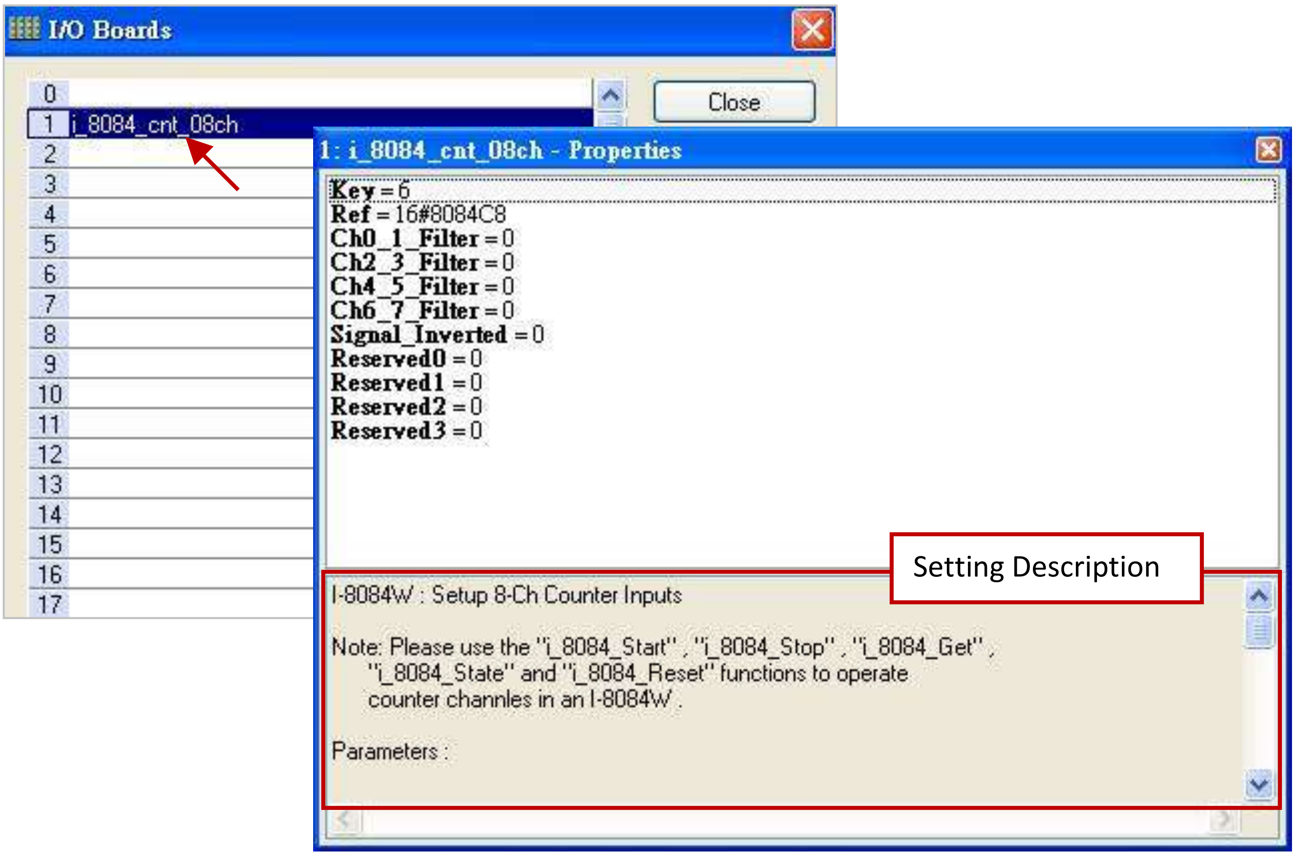
Name	Type	Dim.	Attrib.	Syb.	Init value	User ...	Tag	Description
%IX1 - i_8084_cnt_04ch								
%IX1.0	BOOL		Input	□				

4. After linking the “I/O board”, refer the Section 4.9 to use "COUNTER_START", "COUNTER_STOP", "COUNTER _GET", "COUNTER_STATE" and “COUNTER_RESET” functions in the LD or ST program to operate the Counter channel of the I-8084W.

4.7.3 i_8084_cnt_ch08 (8-channel UP Counter)

Note: Using the "COUNTER_START", "COUNTER_STOP", "COUNTER _GET", "COUNTER_STATE" and "COUNTER_RESET" functions in the Win-GRAF Workbench to operate counter channels in an I-8084W.

1. Mouse double-click the "i_8084_cnt_ch08" to open the “Properties” window, and then to see the setting description.



Key = 6
Ref = 16#8084C8
Ch0_1_Filter = 0
Ch2_3_Filter = 0
Ch4_5_Filter = 0
Ch6_7_Filter = 0
Signal_Inverted = 0
Reserved0 = 0
Reserved1 = 0
Reserved2 = 0
Reserved3 = 0

I-8084W : Setup 8-Ch Counter Inputs

Note: Please use the "i_8084_Start" , "i_8084_Stop" , "i_8084_Get" , "i_8084_State" and "i_8084_Reset" functions to operate counter channles in an I-8084W .

Parameters :

Parameters:

Ch_Filter: The unit is 0.000001 second (μs), the value can be 0 to 200. The default setting is 0 (without filter). The "Ch_Filter" is for filtering out some noise signals with smaller signal width. (Recommend 0: if there is no noise consideration or need a real-time measurement.) The following setting is recommended:

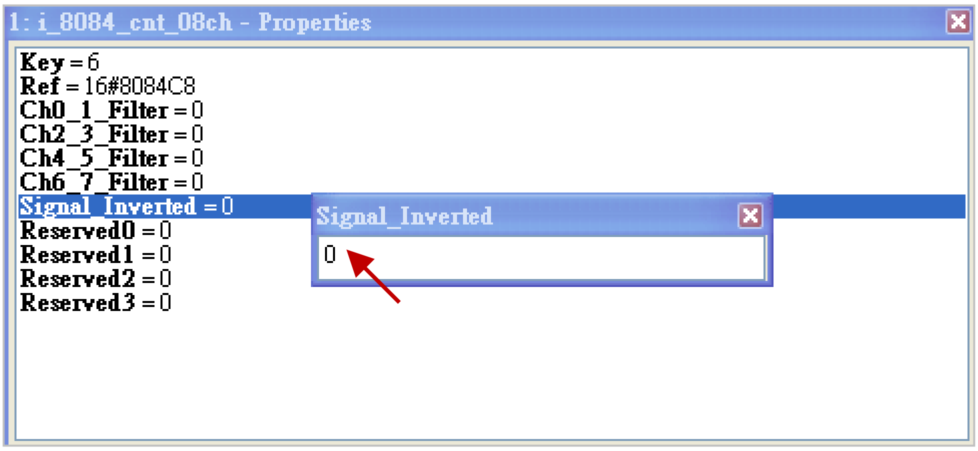
Max Input Signal (Hz)	Recommend Filter Value
1K	200
2K	100
5K	40
10K	20
20K	10
100K	2
450K	1
450K	0 (without filter)

Signal_Inverted:

0: Input signal is normal (no inverted)

1: Input signal is inverted (means voltage HIGH will be processed as LOW, and voltage LOW will be processed as HIGH).

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



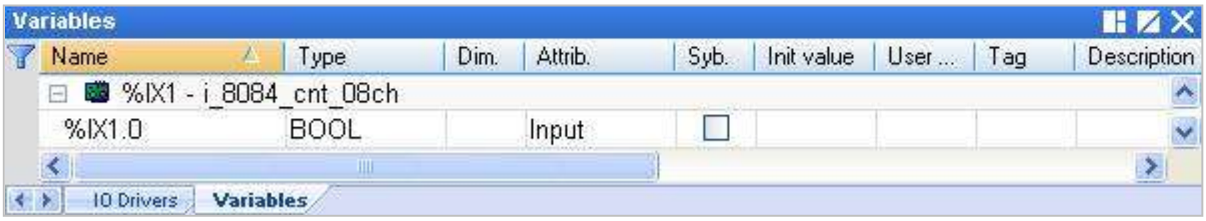
1: i_8084_cnt_08ch - Properties

```
Key = 6
Ref = 16#8084C8
Ch0_1_Filter = 0
Ch2_3_Filter = 0
Ch4_5_Filter = 0
Ch6_7_Filter = 0
Signal_Inverted = 0
Reserved0 = 0
Reserved1 = 0
Reserved2 = 0
Reserved3 = 0
```

Signal_Inverted

0

3. After linking the “i_8084_cnt_ch08” in the “I/O Boards” window, it will auto add one “BOOL” Input variable (no meaning, always “FALSE”) in the “Variables” window.



Variables

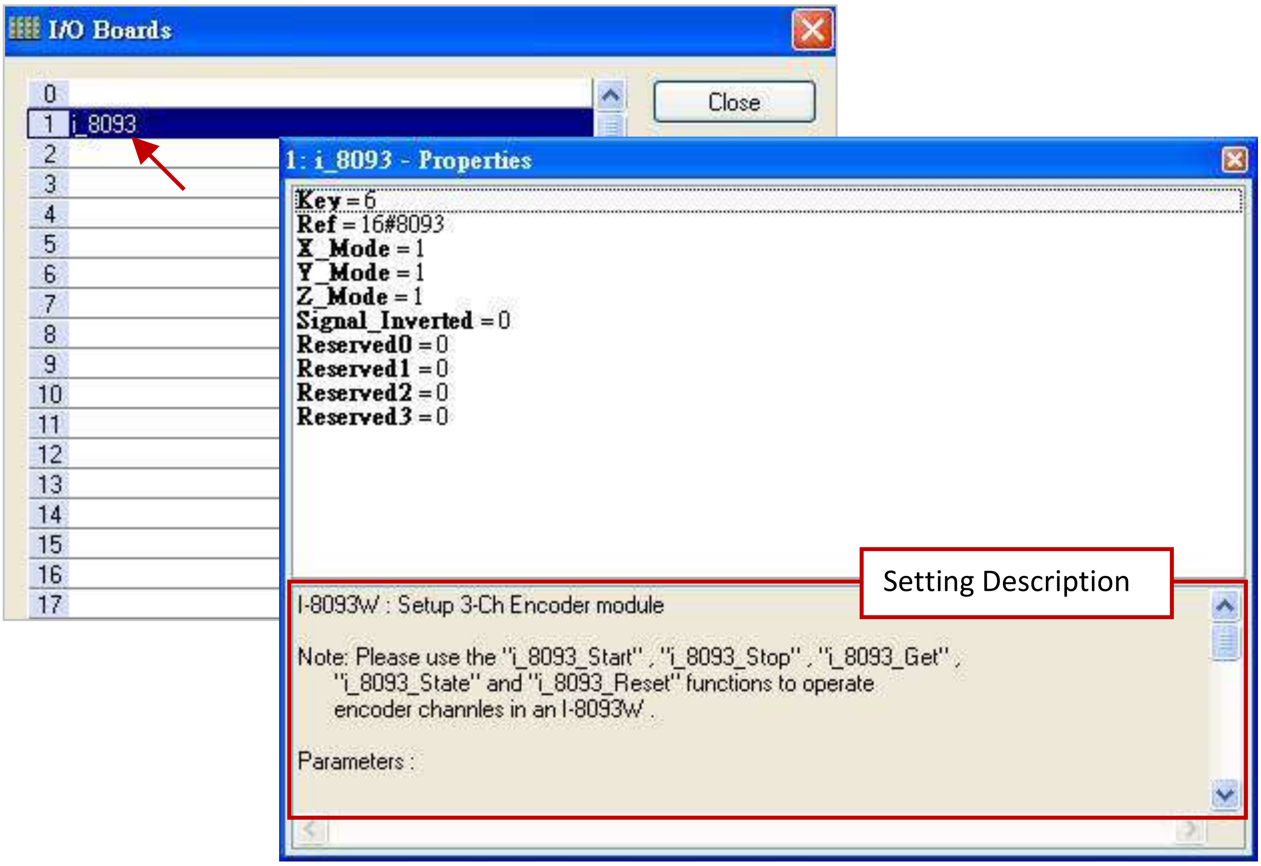
Name	Type	Dim.	Attrib.	Syb.	Init value	User ...	Tag	Description
%IX1 - i_8084_cnt_08ch								
%IX1.0	BOOL		Input	□				

4. After linking the “I/O board”, refer the Section 4.9 to use "COUNTER_START", "COUNTER_STOP", "COUNTER _GET", "COUNTER_STATE" and “COUNTER_RESET” functions in the LD or ST program to operate the Counter channel of the I-8084W.

4.8 i_8093 (3-axis High Speed Encoder Module)

The I-8093W is a 3-axis high speed encoder module that can be independently configured as one of the Quadrant, Pulse/Direction or CW/CCW input mode for each channel. If not familiar with the way to add the this I/O board, see the Chapter4 (P4-1).

1. Mouse double-click the "i_8093" to open the “Properties” window, and then to see the setting description.



Key = 6
Ref = 16#8093
X_Mode = 1
Y_Mode = 1
Z_Mode = 1
Signal_Inverted = 0
Reserved0 = 0
Reserved1 = 0
Reserved2 = 0
Reserved3 = 0

I-8093W : Setup 3-Ch Encoder module

Note: Please use the "i_8093_Start" , "i_8093_Stop" , "i_8093_Get" , "i_8093_State" and "i_8093_Reset" functions to operate encoder channles in an I-8093W .

Parameters :

Parameters:

X_Mode, Y_Mode, Z_Mode:

The input mode of X, Y, Z axis, can be 1, 2 and 3. Set other value will use 1.

1: CW/CCW counting mode.

2: Pulse/Directioncounting mode.

3: A/B phase (quadrant) counting mode.

Signal_Inverted:

0: Input signal is normal (no inverted)

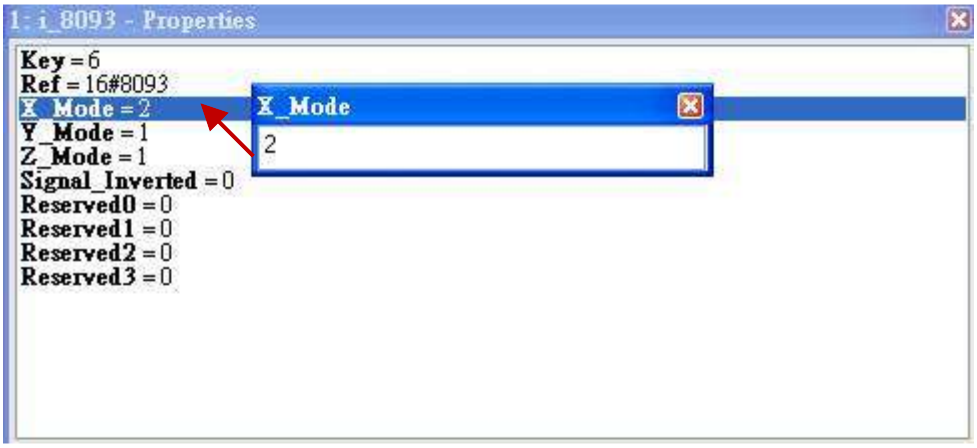
1: Input signal is inverted (means voltage HIGH will be processed as LOW, and voltage LOW will be processed as HIGH).

For example:

If setting "Signal_Inverted" as 0 (no inverted) and X_Mode is 2 (Pulse/Direction), the encoder value will increase if "Dirextion" signal is High.

If setting "Signal_Inverted" as 1 (inverted) and X_Mode is 2 (Pulse/Direction), the encoder value will decrease if "Dirextion" signal is High.

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



1: i_8093 - Properties

Key = 6
Ref = 16#8093
X_Mode = 2
Y_Mode = 1
Z_Mode = 1
Signal_Inverted = 0
Reserved0 = 0
Reserved1 = 0
Reserved2 = 0
Reserved3 = 0

X_Mode

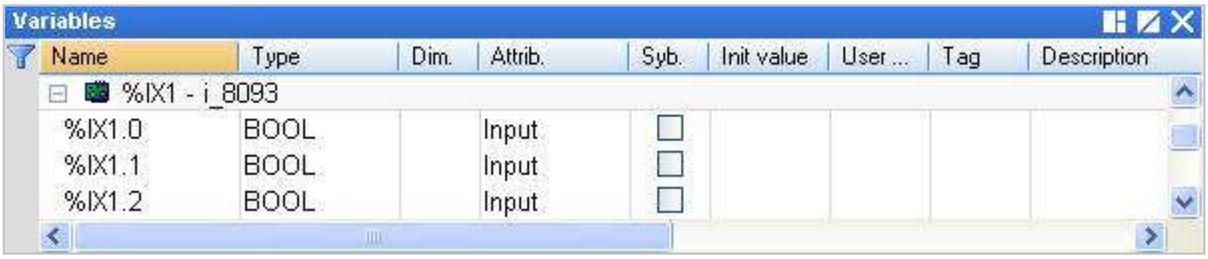
2

3. After linking the “i_8093” in the “I/O Boards” window, it will auto add 3 “BOOL” input variables in the “Variables” window that are available for programing.

Ch0: Z-index of X axis.

Ch1: Z-index of Y axis.

Ch2: Z-index of Z axis.



Variables

Name	Type	Dim.	Attrib.	Syb.	Init value	User ...	Tag	Description
%IX1 - i_8093								
%IX1.0	BOOL		Input	□				
%IX1.1	BOOL		Input	□				
%IX1.2	BOOL		Input	□				

4. After linking the “I/O Boards”, refer the Section 4.9 to use "COUNTER_START", "COUNTER_STOP", "COUNTER _GET", "COUNTER_STATE" and “COUNTER_RESET” functions in the LD or ST program to operate the Encoder channel of the I-8093W.

4.9 Using the Count Function for I-8084W, I-8093W, I-87082W, I-87084W, I-7083 and I-7080 Modules

This section lists the way to use the "COUNTER_START", "COUNTER_STOP", "COUNTER _GET", "COUNTER_STATE" and "COUNTER_RESET" functions in the LD or ST program to operate the Counter and Encoder modules. If not familiar with the way to create a program or a function block, see the Section 2.3.3.

Note:

1. **In the following content, we use I-8084W and I-8093W modules as examples.**
2. **Before using these function blocks, first go to Section 4.7.2 (UP/Down Counter), Section 4.7.3 (UP Counter) and Section 4.8 (Encoder) to link I/O Boards.**

4.9.1 COUNTER_START

For example: Using the I-8084W module in the PAC's slot2 and start counting the channel 5.

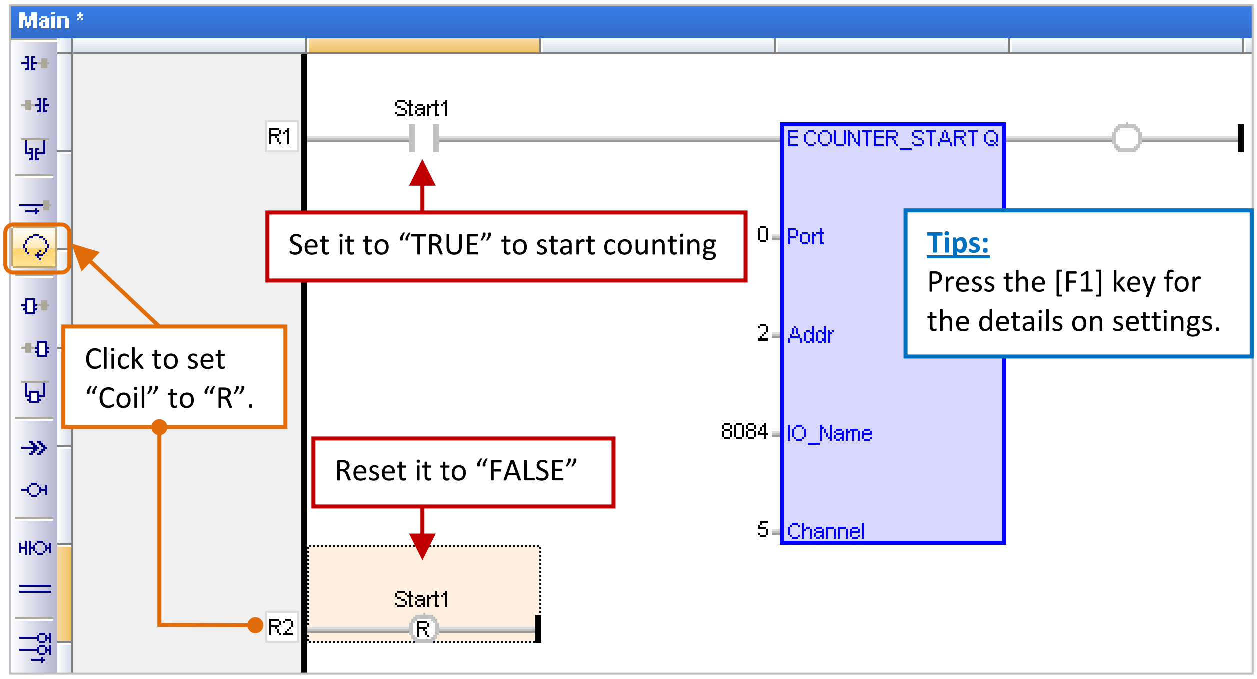
ST program:

```
IF   Start1 = TRUE   THEN
  Start1  :=  FALSE ;
  TMP_BOOL  :=  Counter_Start (0, 2, 8084, 5) ;
END_IF ;
```

Note:
First, add two BOOL variables ("Start1", "TMP_BOOL") in the Variables Area.

LD program:

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



Set it to "TRUE" to start counting

Click to set "Coil" to "R".

Reset it to "FALSE"

Tips:
Press the [F1] key for the details on settings.

Port: (Data type: “DINT”)
For a module in the slot 0 to 7 of the PAC, set it as “0”. For a DCON module connected to a serial COM port, can be “1 to 37” (depends on PAC, means COM1 to COM37).

Addr: (Data type: “DINT”)
For a module in the slot 0 to 7 of the PAC, set it as the slot number (0 to 7).
For a module connected to a serial COM port, set it as the Net-ID address of the module (1 to 255).

IO_Name: (Data type: “DINT”)
The name of relative Counter/Encoder module, it can be set to "8084”, "8093", "87084", "87082", "7083" and "7080".

Channel: (Data type: “DINT”)
The channel No. of the Counter/Encoder module, it can be set to "0", "1", and so on, depends on the module. For example, when using the I-8093W module, "0" means the X-axis, "1" means the Y-axis and "2" means the Z-axis.

Q: (Data type: “BOOL”)
“TRUE”: OK ; “FALSE”: Error.

4.9.2 COUNTER_STOP

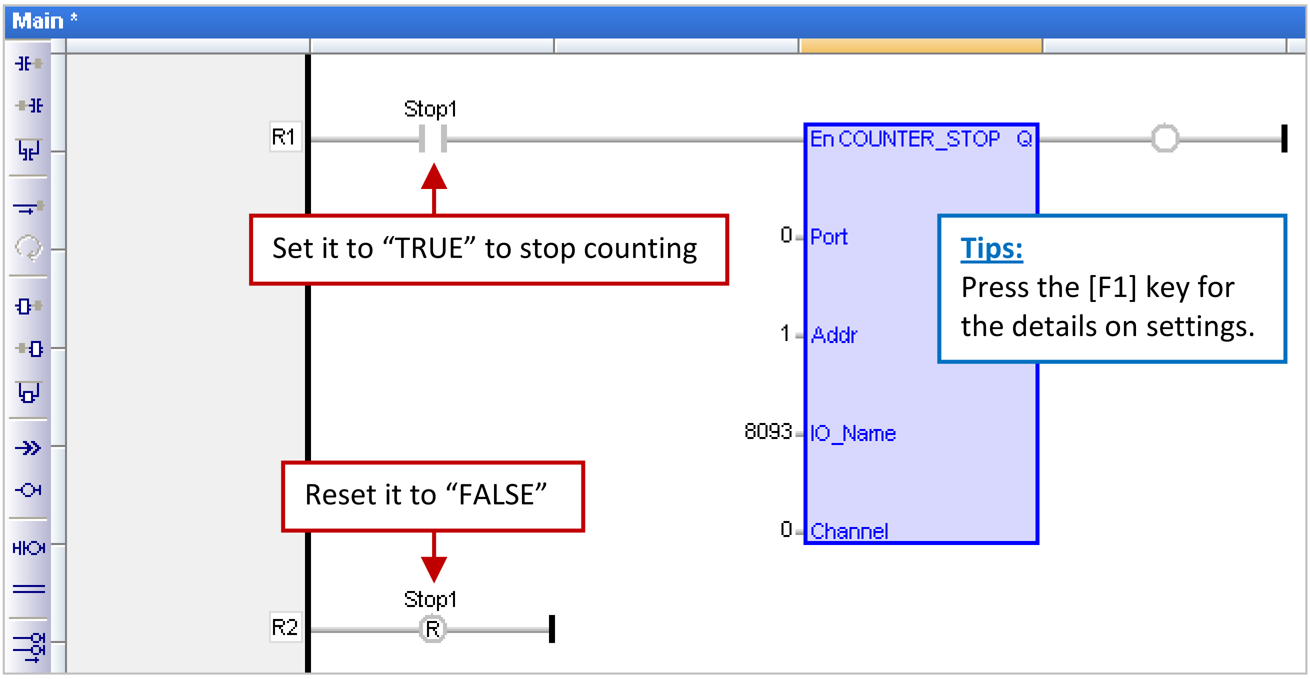
For example: Using the I-8093W module in the PAC's slot1 and stop counting the X-axis.

ST program:

```
IF   Stop1 = TRUE   THEN
  Stop1  :=  FALSE ;
  TMP_BOOL  :=  Counter_Stop (0, 1, 8093, 0) ;
END_IF ;
```

Note:
First, add two BOOL variables ("Stop1", "TMP_BOOL") in the Variables Area.

LD program:
("Stop1": Boolean, set it to "TRUE" to start counting and then reset "Stop1" to "FALSE".)



Set it to "TRUE" to stop counting

Reset it to "FALSE"

Tips:
Press the [F1] key for the details on settings.

Port: (Data type: "DINT")
For a module in the slot 0 to 7 of the PAC, set it as "0". For a DCON module connected to a serial COM port, can be "1 to 37" (depends on PAC, means COM1 to COM37).

Addr: (Data type: "DINT")
For a module in the slot 0 to 7 of the PAC, set it as the slot number (0 to 7).
For a module connected to a serial COM port, set it as the Net-ID address of the module (1 to 255).

IO_Name: (Data type: "DINT")
The name of relative Counter/Encoder module, it can be set to "8084", "8093", "87084", "87082", "7083" and "7080".

Channel: (Data type: "DINT")
The channel No. of the Counter/Encoder module, it can be set to "0", "1", and so on, depends on the module. For example, when using the I-8093W module, "0" means the X-axis, "1" means the Y-axis and "2" means the Z-axis.

Q: (Data type: "BOOL")
"TRUE": OK ; "FALSE": Error.

4.9.3 COUNTER_GET

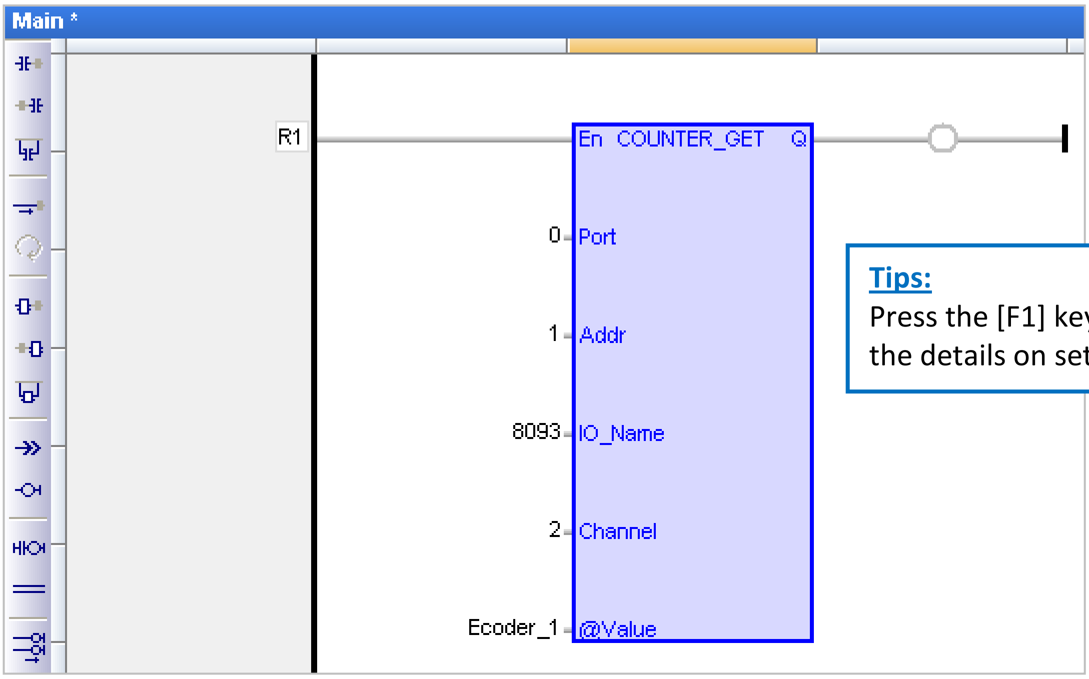
For example: Using the I-8093W module in the PAC's slot1 and get the Encoder value of the Z-axis.

ST program:

```
TMP_BOOL := Counter_Get (0, 1, 8093, 2, Encoder_1) ;
```

Note: First, add variables in the Variables Area (see section 2.2.1).
"TMP_BOOL" (BOOL).
"Encoder_1" (DINT).

LD program:



Tips:
Press the [F1] key for the details on settings.

Port: (Data type: "DINT")
For a module in the slot 0 to 7 of the PAC, set it as "0". For a DCON module connected to a serial COM port, can be "1 to 37" (depends on PAC, means COM1 to COM37).

Addr: (Data type: "DINT")
For a module in the slot 0 to 7 of the PAC, set it as the slot number (0 to 7).
For a module connected to a serial COM port, set it as the Net-ID address of the module (1 to 255).

IO_Name: (Data type: "DINT")
The name of relative Counter/Encoder module, it can be set to "8084", "8093", "87084", "87082", "7083" and "7080".

Channel: (Data type: "DINT")
The channel No. of the Counter/Encoder module, it can be set to "0", "1", and so on, depends on the module. For example, when using the I-8093W module, "0" means the X-axis, "1" means the Y-axis and "2" means the Z-axis.

@Value: (The data type can be "DINT", "UDINT", "DWORD", "LINT" and "ULINT")
It returns the current counter or encoder value. (Refer the Appendix A for the range of values)

Q: (Data type: "BOOL")
Counting state. "TRUE": Counting ; "FALSE": Stopped.

4.9.4 COUNTER_STATE

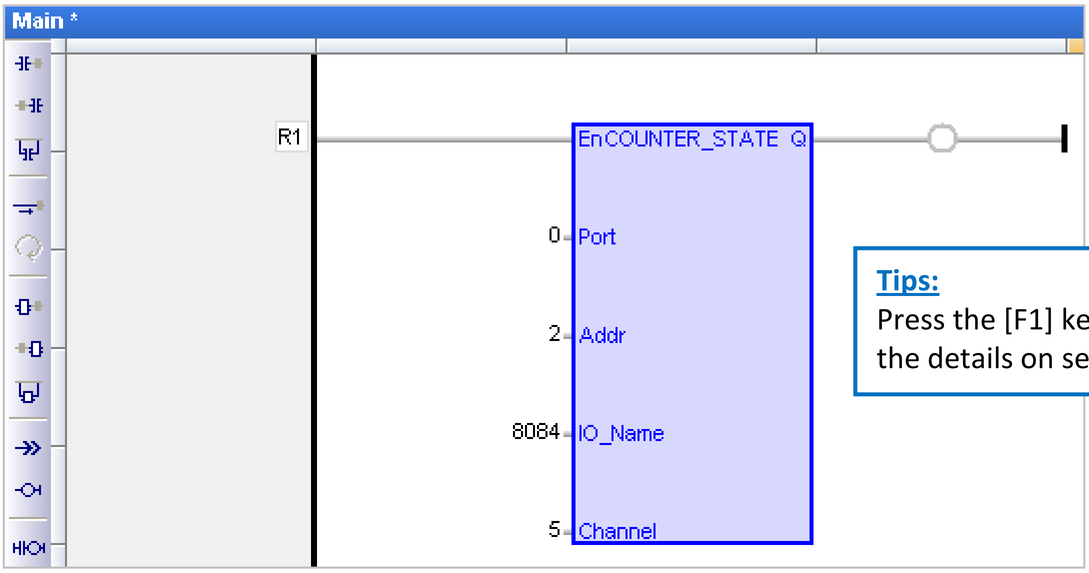
For example: Using the I-8084W module in the PAC‘s slot2 and to get the counting status of the channel 5.

ST program:

```
TMP_BOOL := Counter_State (0, 2, 8084, 5) ;
```

Note: First, add a "TMP_BOOL" BOOL variable in the Variable Area.

LD program:



Tips:
Press the [F1] key for the details on settings.

Port: (Data type: “DINT”)
For a module in the slot 0 to 7 of the PAC, set it as “0”. For a DCON module connected to a serial COM port, can be “1 to 37” (depends on PAC, means COM1 to COM37).

Addr: (Data type: “DINT”)
For a module in the slot 0 to 7 of the PAC, set it as the slot number (0 to 7).
For a module connected to a serial COM port, set it as the Net-ID address of the module (1 to 255).

IO_Name: (Data type: “DINT”)
The name of relative Counter/Encoder module, it can be set to "8084”, "8093", "87084", "87082", "7083" and "7080".

Channel: (Data type: “DINT”)
The channel No. of the Counter/Encoder module, it can be set to "0", "1", and so on, depends on the module. For example, when using the I-8093W module, "0" means the X-axis, "1" means the Y-axis and "2" means the Z-axis.

Q: (Data type: “BOOL”)
“TRUE”: OK ; “FALSE”: Error.

4.9.5 COUNTER_RESET

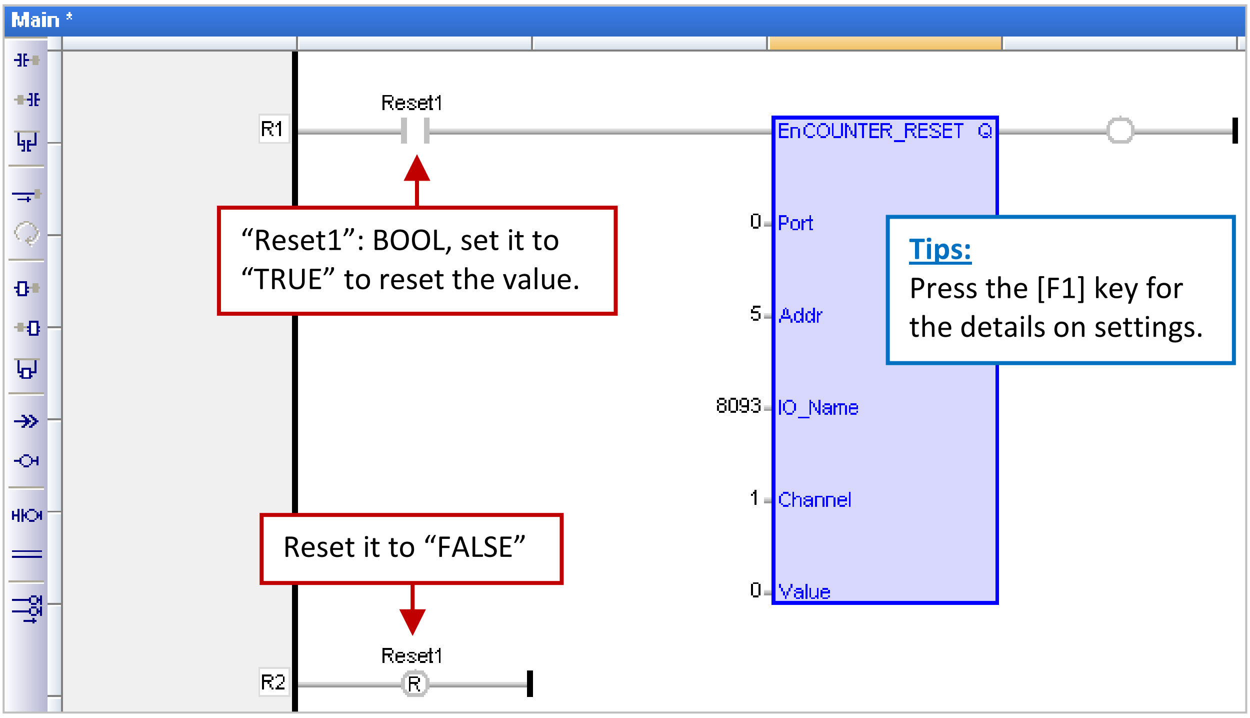
For example: Using the I-8093W module in the PAC's slot5 and reset the Encoder value of the Y-axis as "0".

ST program:

```
IF   Reset1 = TRUE   THEN
  Reset1  :=  FALSE ;
  TMP_BOOL  :=  Counter_Reset (0, 5, 8093, 1, 0) ;
END_IF ;
```

Note:
First, add two BOOL variables ("Reset1", "TMP_BOOL") in the Variable Area.

LD program:



"Reset1": BOOL, set it to "TRUE" to reset the value.

Tips:
Press the [F1] key for the details on settings.

Reset it to "FALSE"

Port: (Data type: "DINT")
For a module in the slot 0 to 7 of the PAC, set it as "0". For a DCON module connected to a serial COM port, can be "1 to 37" (depends on PAC, means COM1 to COM37).

Addr: (Data type: "DINT")
For a module in the slot 0 to 7 of the PAC, set it as the slot number (0 to 7).
For a module connected to a serial COM port, set it as the Net-ID address of the module (1 to 255).

IO_Name: (Data type: "DINT")
The name of relative Counter/Encoder module, it can be set to "8084", "8093", "87084", "87082", "7083" and "7080".

Channel: (Data type: "DINT")
The channel No. of the Counter/Encoder module, it can be set to "0", "1", and so on, depends on the module. For example, when using the I-8093W module, "0" means the X-axis, "1" means the Y-axis and "2" means the Z-axis.

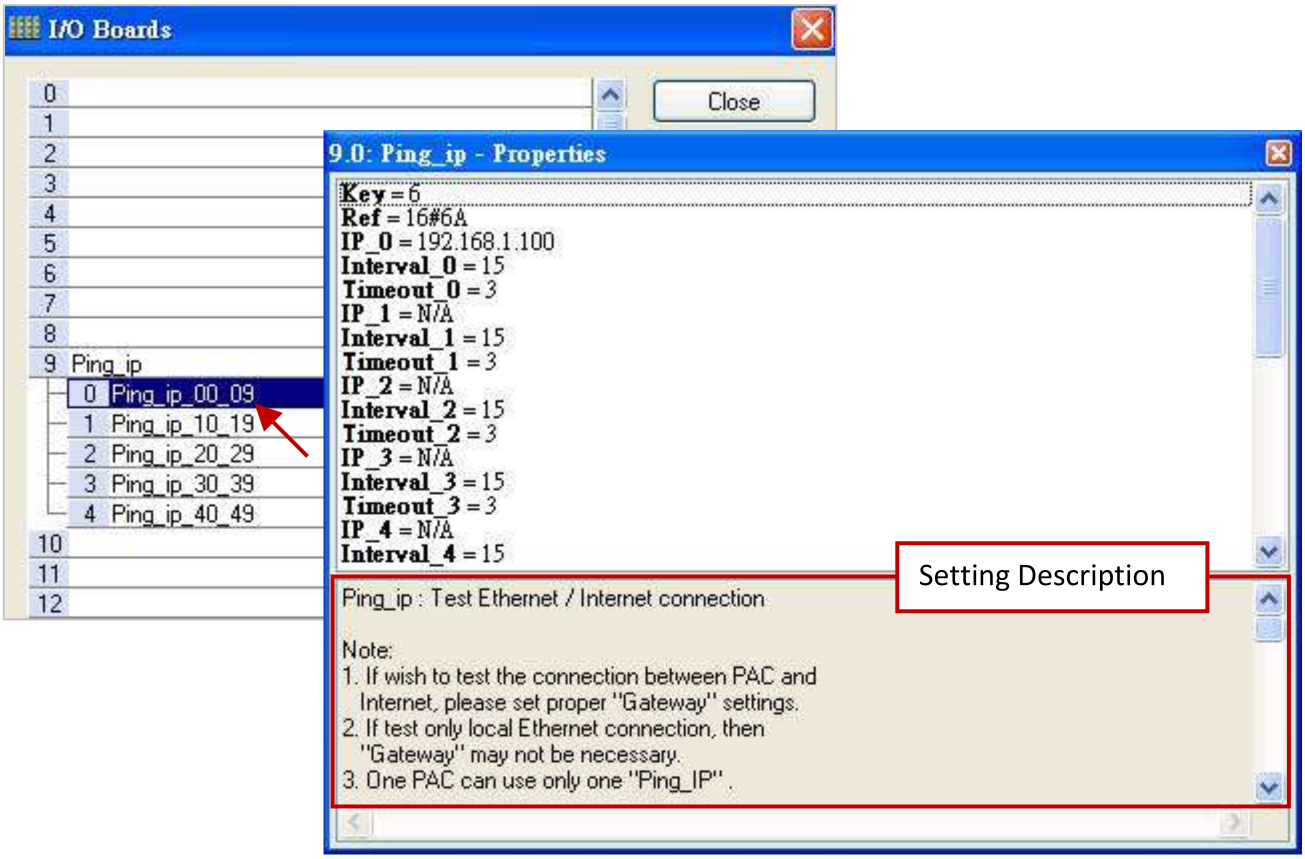
Value: (The data type can be "DINT", "UDINT", "DWORD", "LINT" and "ULINT")
The new Counter or Encoder value wish to set.

Q: (Data type: "BOOL") "TRUE": OK ; "FALSE": Error.

4.10 Ping_ip (Test an Ethernet/Internet Connection)

The "Ping_ip" function is used to test if the connection of the remote Ethernet/ Internet device is working properly. (It supports a max. of 50 IP settings.) If not familiar with the way to add this I/O board, see the Chapter4 (P4-1).

1. Mouse double-click the "Ping_ip" to open the "Properties" window, and then to see the setting description.
   Note: Using the slot 8 or above No. because the slot 0 to 7 are reserved for the real I/O module.



Key = 6
Ref = 16#6A
IP_0 = 192.168.1.100
Interval_0 = 15
Timeout_0 = 3
IP_1 = N/A
Interval_1 = 15
Timeout_1 = 3
IP_2 = N/A
Interval_2 = 15
Timeout_2 = 3
IP_3 = N/A
Interval_3 = 15
Timeout_3 = 3
IP_4 = N/A
Interval_4 = 15

Ping_ip : Test Ethernet / Internet connection

Note:
1. If wish to test the connection between PAC and Internet, please set proper "Gateway" settings.
2. If test only local Ethernet connection, then "Gateway" may not be necessary.
3. One PAC can use only one "Ping_IP" .

Note:

1. If wish to test the connection between PAC and Internet, please set proper "Gateway" settings.
2. If test only local Ethernet connection, then "Gateway" may not be necessary.
3. One PAC can use only one "Ping_IP". (Don't use two or more)
4. When Ping success, return Boolean channel as TRUE.
5. When Ping fails, it will try one more time. If still fail, then return Boolean channel as FALSE.

Parameters:

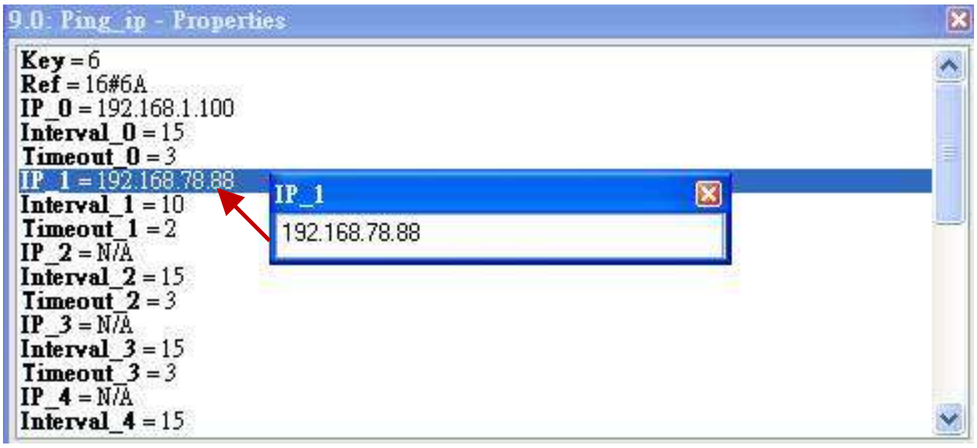
IP_01 to IP_49: (Data type: "STRING")
The IP address of targets. Set as 'N/A' if wish to disable it.
For example, 192.168.1.100 or 52.19.125.242 or N/A.

Interval_01 to Interval_49: (Data type: "DINT")
The unit is second. The interval to send one "ping" command. Value can be 6 to 86,400 seconds. Setting smaller than 6 will use as 6. Setting greater than 86400 (24 hours) will use as 86400.

Timeout_01 to Timeout_49: (Data type: "DINT")
The unit is second. The timeout settings of the "ping" command. Value can be 2 to 30 seconds. Setting smaller than 2 will use as 2. Setting greater than 30 will use as 30.

Note: The "Interval_xx" value should be **at least triple** of the "Timeout_x" value. Or the PAC will use the "Interval_x" value as a triple of the "Timeout_x" value.
For example, if "Timeout_00" is set as 10 however "Interval_00" is set as 20, then PAC will use "Interval_00" as 30.

2. Double click the item and fill in a value, then press the "Enter" key to complete the setting.



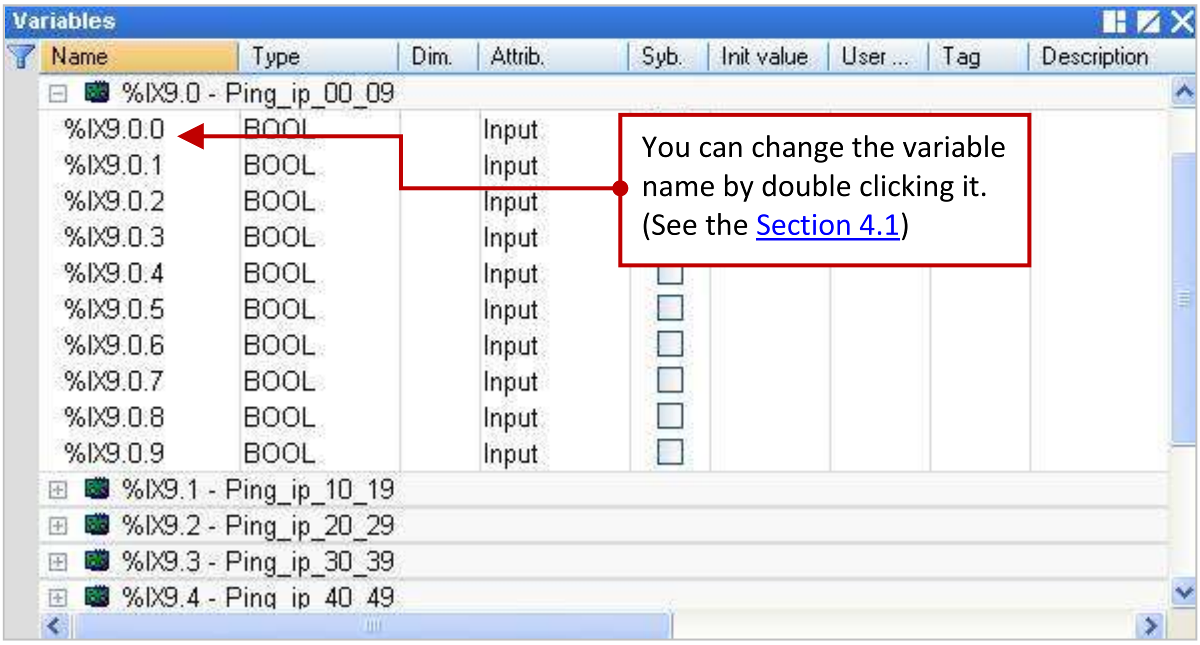
9.0: Ping_ip - Properties

```
Key = 6
Ref = 16#6A
IP_0 = 192.168.1.100
Interval_0 = 15
Timeout_0 = 3
IP_1 = 192.168.78.88
Interval_1 = 10
Timeout_1 = 2
IP_2 = N/A
Interval_2 = 15
Timeout_2 = 3
IP_3 = N/A
Interval_3 = 15
Timeout_3 = 3
IP_4 = N/A
Interval_4 = 15
```

IP_1

192.168.78.88

3. After linking the "Ping_ip" in the "I/O Boards" window, it will auto add 50 "BOOL" input variables in the "Variables" window. When the Win-GRAF connects the PAC, it will display the online status.
True: The connection is ok.
FALSE: Connection failed or cable problem.



Variables

Name	Type	Dim.	Attrib.	Syb.	Init value	User ...	Tag	Description
%IX9.0 - Ping_ip_00_09								
%IX9.0.0	BOOL		Input					
%IX9.0.1	BOOL		Input					
%IX9.0.2	BOOL		Input					
%IX9.0.3	BOOL		Input					
%IX9.0.4	BOOL		Input	□				
%IX9.0.5	BOOL		Input	□				
%IX9.0.6	BOOL		Input	□				
%IX9.0.7	BOOL		Input	□				
%IX9.0.8	BOOL		Input	□				
%IX9.0.9	BOOL		Input	□				
%IX9.1 - Ping_ip_10_19								
%IX9.2 - Ping_ip_20_29								
%IX9.3 - Ping_ip_30_39								
%IX9.4 - Ping_ip_40_49								

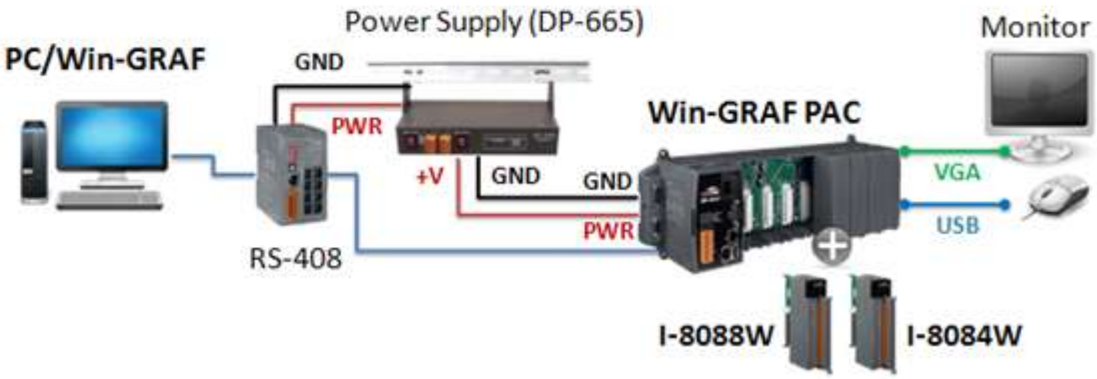
You can change the variable name by double clicking it. (See the Section 4.1)

4.11 I-8088W (8-channel PWM Output Module)

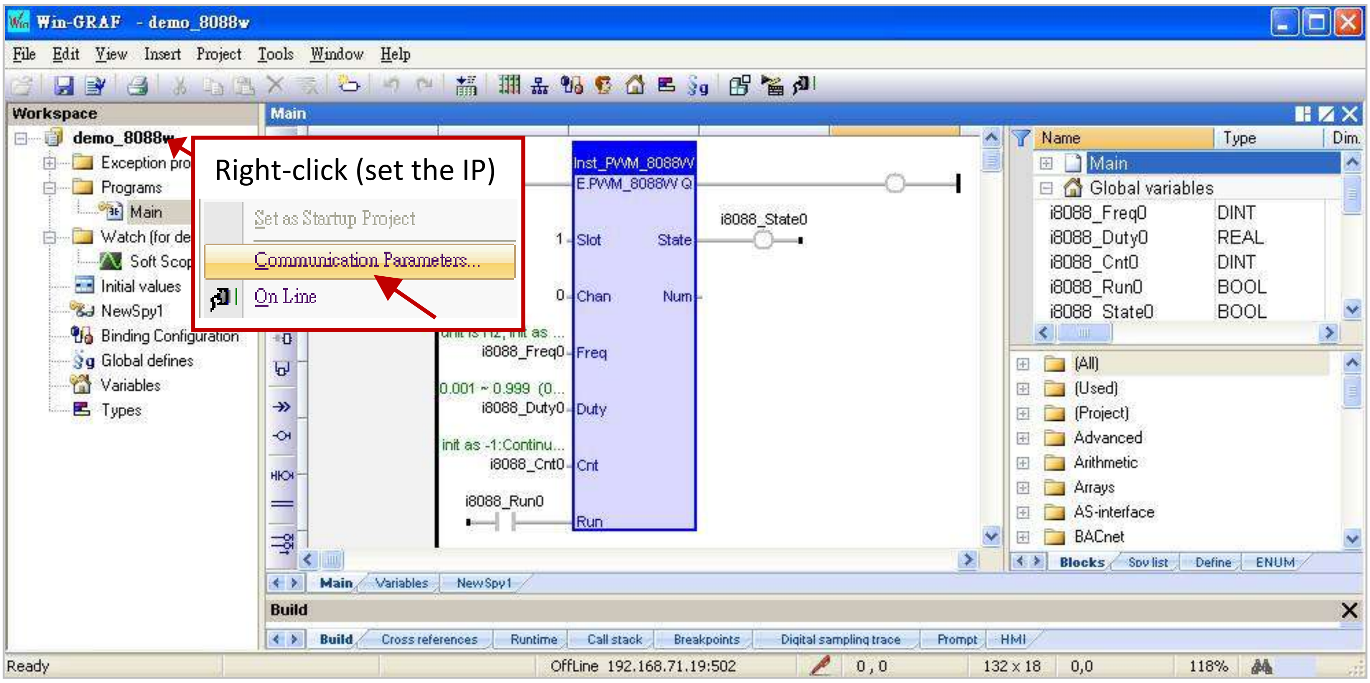
The I-8088W is an 8-channel PWM (Pulse Width Modulation) output module. The duty range (Duty = High / (High + Low)) of the PWM output signal can be from 0.1% to 99.9%. Its output frequency in the Win-GRAF PAC is from 1 Hz to 500 KHz. The I-8088W support two PWM output modes, one is the "Continuous" mode. It outputs always. The other one is the "Burst" mode. It outputs the required pulse count and then stop. Please visit http://www.icpdas.com/products/Remote_IO/i-8ke/i-8088w.htm for other specifications.

Hardware Connection Diagram:

This example uses the I-8084W (Slot 2) to measure the frequency of the I-8088W (Slot 1) PWM output signal (the I-8084W is not necessary in the actual application). Then, connect the I-8088W's PWM output channel 0 (PW0) to the I-8084W's frequency input channel 0 (C0A+).

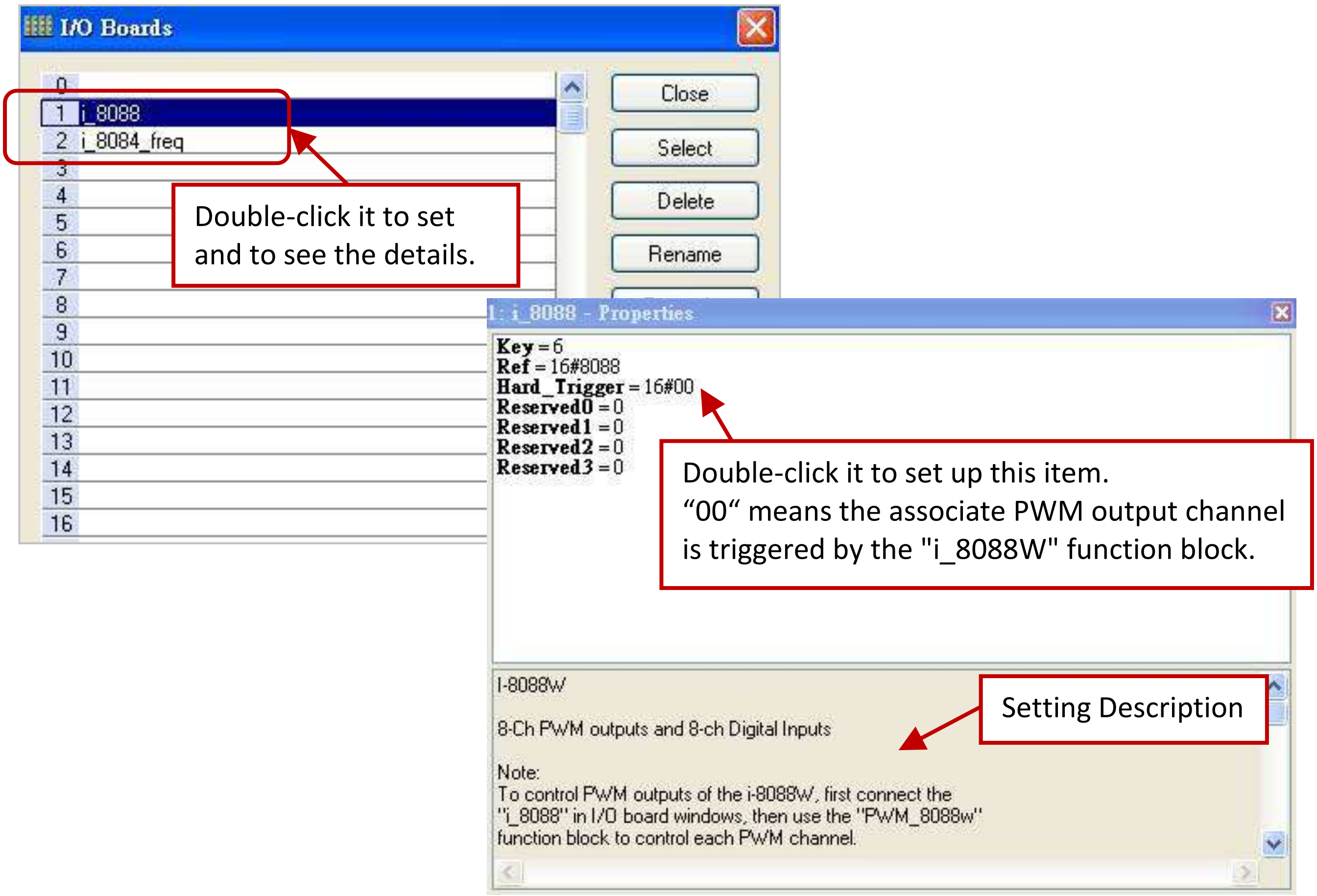


The demo program (demo_8088w.zip) that we will describe below is located in the shipment CD (\Napdos\Win-GRAF\demo-project) , refer the Chapter 12 to restore/open this project and set up the current PAC's IP address.



I/O Boards:

In this case, add the "i_8088" and the "i_8084_freq" to the corresponding I/O slot No. in the "I/O Boards" window (see the Chapter 4). Then, mouse double-click the Slot No. to open the "Properties" window and you can see the setting description to set this I/O board.



Double-click it to set and to see the details.

Key = 6
Ref = 16#8088
Hard_Trigger = 16#00
Reserved0 = 0
Reserved1 = 0
Reserved2 = 0
Reserved3 = 0

Double-click it to set up this item.
“00“ means the associate PWM output channel is triggered by the "i_8088W" function block.

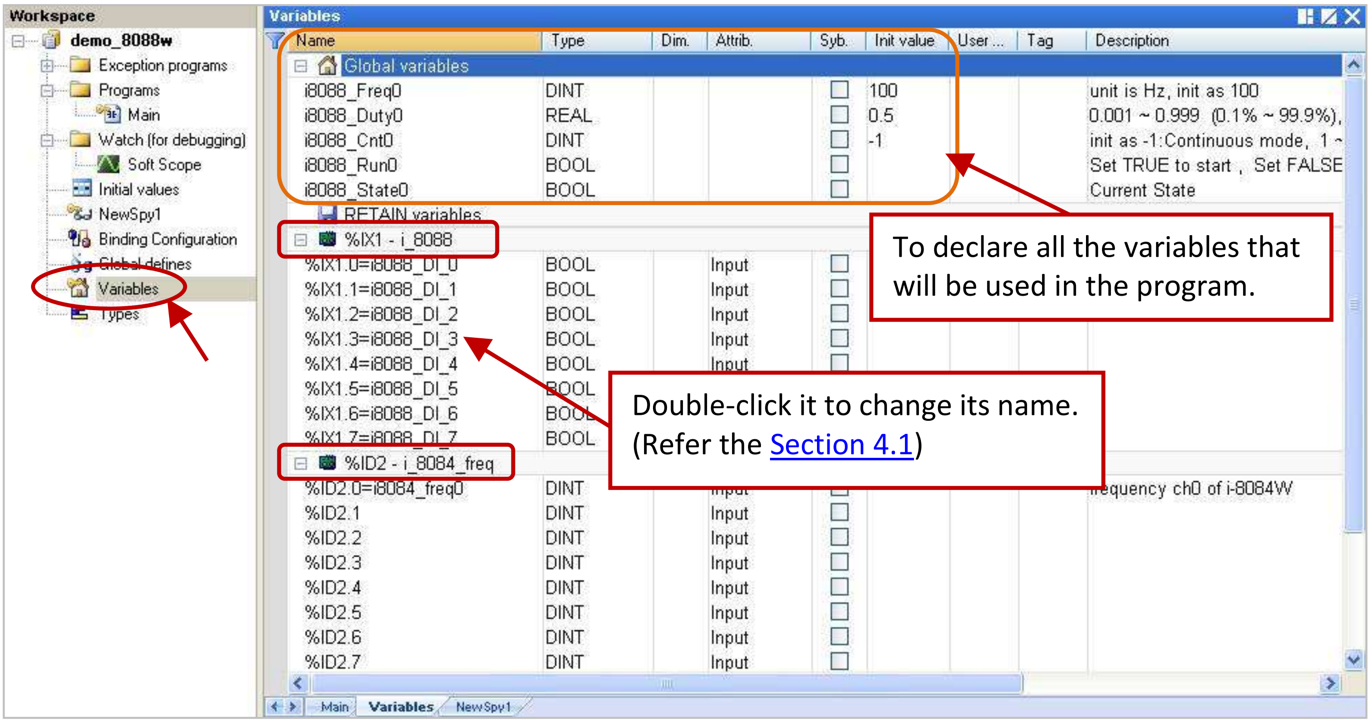
I-8088W

8-Ch PWM outputs and 8-ch Digital Inputs

Note:
To control PWM outputs of the i-8088W, first connect the "i_8088" in I/O board windows, then use the "PWM_8088w" function block to control each PWM channel.

Setting Description

After linking the "i_8088" and the "i_8084_freq" I/O boards, it will auto add related variables in the "Variables" window (or Variables Area). And, you can also declare all the variables that will be used in the program here (refer the Section 2.3.1).



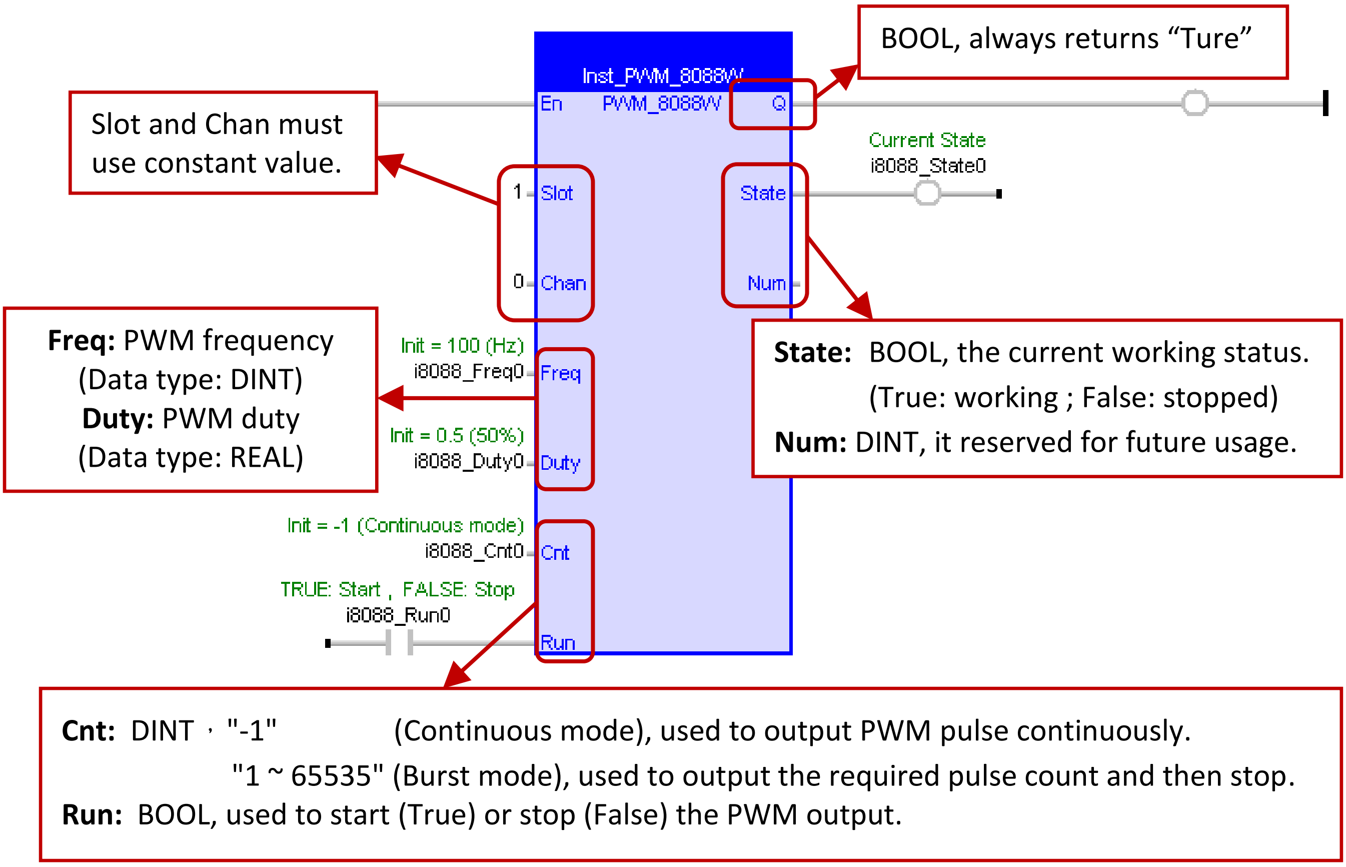
Name	Type	Dim.	Attrib.	Syb.	Init value	User ...	Tag	Description
Global variables								
i8088_Freq0	DINT			□	100			unit is Hz, init as 100
i8088_Duty0	REAL			□	0.5			0.001 ~ 0.999 (0.1% ~ 99.9%),
i8088_Cnt0	DINT			□	-1			init as -1:Continuous mode, 1 ~
i8088_Run0	BOOL			□				Set TRUE to start , Set FALSE
i8088_State0	BOOL			□				Current State
RETAIN variables								
%IX1 - i_8088								
%IX1.0=i8088_DI_0	BOOL		Input	□				
%IX1.1=i8088_DI_1	BOOL		Input	□				
%IX1.2=i8088_DI_2	BOOL		Input	□				
%IX1.3=i8088_DI_3	BOOL		Input	□				
%IX1.4=i8088_DI_4	BOOL		Input	□				
%IX1.5=i8088_DI_5	BOOL							
%IX1.6=i8088_DI_6	BOOL							
%IX1.7=i8088_DI_7	BOOL							
%ID2 - i_8084_freq								
%ID2.0=i8084_freq0	DINT		Input	□				requency ch0 of i-8084W
%ID2.1	DINT		Input	□				
%ID2.2	DINT		Input	□				
%ID2.3	DINT		Input	□				
%ID2.4	DINT		Input	□				
%ID2.5	DINT		Input	□				
%ID2.6	DINT		Input	□				
%ID2.7	DINT		Input	□				

To declare all the variables that will be used in the program.

Double-click it to change its name. (Refer the Section 4.1)

"PWM_8088W" Function Block:

Then, using the function block “PWM_8088W” to control the PWM output for each channel, such as the LD program below.



BOOL, always returns “Ture”

Slot and Chan must use constant value.

Freq: PWM frequency (Data type: DINT)
Duty: PWM duty (Data type: REAL)

State: BOOL, the current working status. (True: working ; False: stopped)
Num: DINT, it reserved for future usage.

Cnt: DINT ‧ "-1" (Continuous mode), used to output PWM pulse continuously.
"1 ~ 65535" (Burst mode), used to output the required pulse count and then stop.
Run: BOOL, used to start (True) or stop (False) the PWM output.

Parameters:

Slot: The used I/O slot No., and it must be a constant value, not a changeable value.
In this case, the value is "1". (Data type: DINT)

Chan: The used I/O channel No., and it must be a constant value, not a changeable value.
In this case, the value is "0". (Data type: DINT)

Freq: The PWM output frequency. (Data type: DINT ; Unit: Hz)
The value can be from 1 to 500,000 (i.e., 1 Hz to 500 KHz) .
In this case, the initial value is "100" Hz.

Duty: The PWM output. (Data type: REAL)
The value can be from 0.001 to 0.999 (i.e., 0.1 % to 99.9 %).
In this case, the initial value is 0.5 (i.e., 50 %).

Cnt: The output mode (Data type: DINT)
Continuous mode: set it as “-1” (in this case) to output PWM pulse continuously.
Burst mode: it can be from "1" to "65535", to output the required pulse count and then stop.

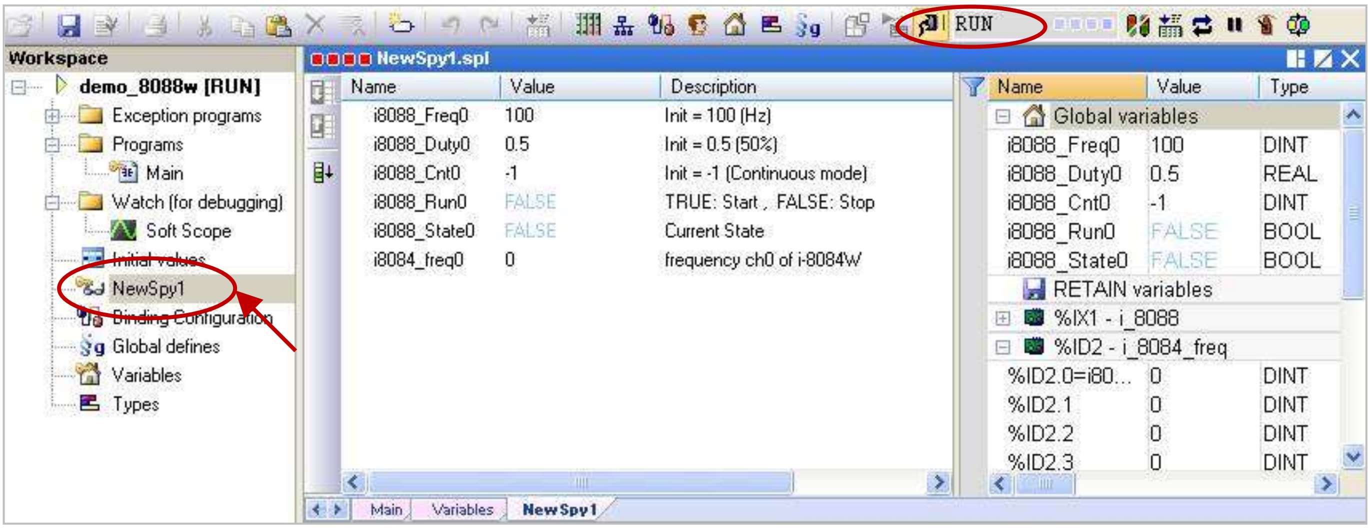
Run: Using a BOOL variable to trigger the PWM output. (True: Start ; False: Stop)

State: The current working status. (Data type: BOOL). (True: working ; False: stopped)

Test the program:

Before testing, make sure you have set the PAC's IP address and then compile and download this program to the PAC. (If not familiar with the operation, refer the Section 2.3.4 and Section 2.3.5.)

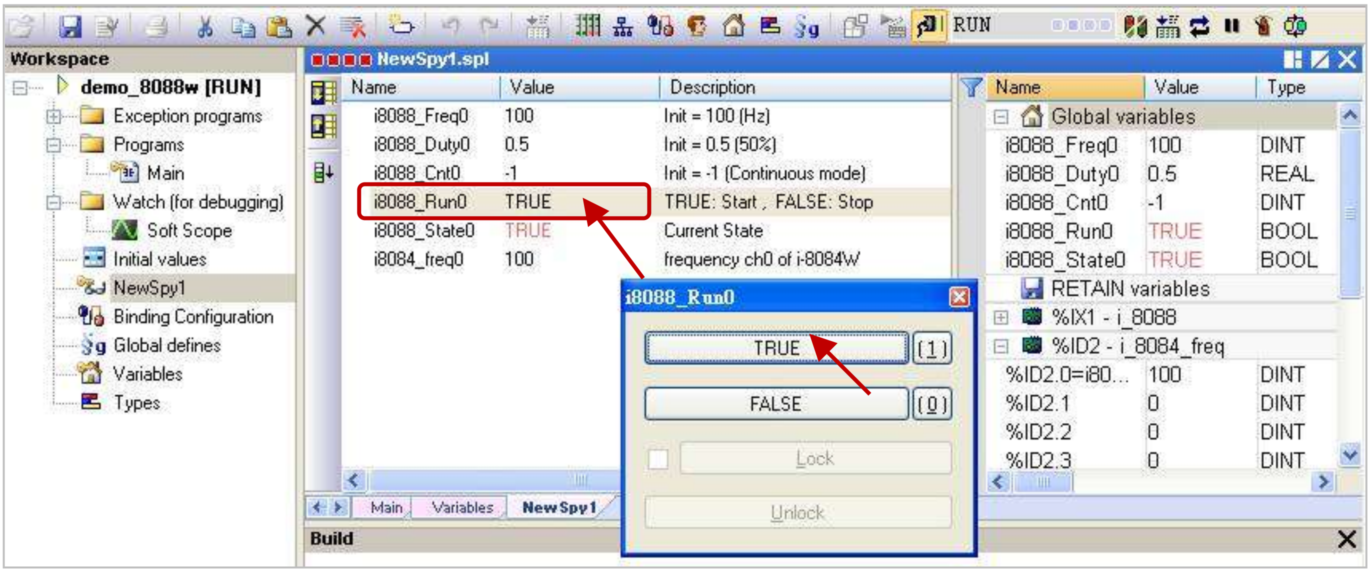
When connecting with the PAC, the SPY List (refer the Section 11.3) will show that the I-8088W's PWM initial frequency ("i8088_Freq0") is 100 Hz, the initial duty cycle ("i8088_Duty0") is 50%, using the Continuous mode ("i8088_Cnt0" = -1) and the currently measured frequency of the I-8084W is 0 Hz.



Name	Value	Description
i8088_Freq0	100	Init = 100 (Hz)
i8088_Duty0	0.5	Init = 0.5 (50%)
i8088_Cnt0	-1	Init = -1 (Continuous mode)
i8088_Run0	FALSE	TRUE: Start , FALSE: Stop
i8088_State0	FALSE	Current State
i8084_freq0	0	frequency ch0 of i-8084W

Name	Value	Type
Global variables		
i8088_Freq0	100	DINT
i8088_Duty0	0.5	REAL
i8088_Cnt0	-1	DINT
i8088_Run0	FALSE	BOOL
i8088_State0	FALSE	BOOL
RETAIN variables		
%IX1 - i_8088		
%ID2 - i_8084_freq		
%ID2.0=i80...	0	DINT
%ID2.1	0	DINT
%ID2.2	0	DINT
%ID2.3	0	DINT

Now, set the "i8088_Run0" as "TRUE" to start the PWM output. At this time, the "i8088_State" will change from "FALSE" to "TRUE" and output a PWM signal to the I-8084W module, and then the value of the "i8084_Freq0" will change from 0 Hz to 100 Hz.



Name	Value	Description
i8088_Freq0	100	Init = 100 (Hz)
i8088_Duty0	0.5	Init = 0.5 (50%)
i8088_Cnt0	-1	Init = -1 (Continuous mode)
i8088_Run0	TRUE	TRUE: Start , FALSE: Stop
i8088_State0	TRUE	Current State
i8084_freq0	100	frequency ch0 of i-8084W

Name	Value	Type
Global variables		
i8088_Freq0	100	DINT
i8088_Duty0	0.5	REAL
i8088_Cnt0	-1	DINT
i8088_Run0	TRUE	BOOL
i8088_State0	TRUE	BOOL
RETAIN variables		
%IX1 - i_8088		
%ID2 - i_8084_freq		
%ID2.0=i80...	100	DINT
%ID2.1	0	DINT
%ID2.2	0	DINT
%ID2.3	0	DINT

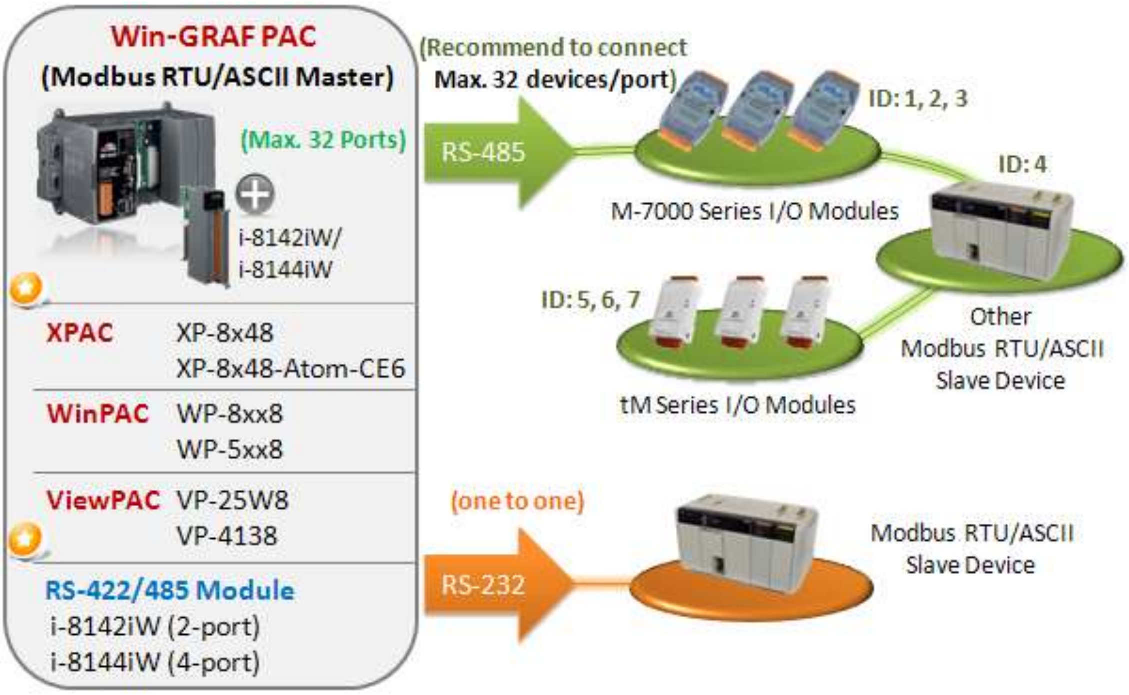
If set the "i8088_Cnt0" as 500 (Burst mode), the "i8084_Freq0" value will become 0 after the I-8088W outputs 500 PWM pulses. You can try to change the "i8088_Cnt0" value and then set the "i8088_Run0" as "TRUE" to view the changes of output.

Chapter 5 Modbus Master: connect to Modbus Slave Device

This chapter lists the way to enable the Win-GRAF PAC as a Modbus Master to connect a Modbus RTU/ASCII Slave or Modbus TCP/UDP Slave device. If you want to use one XV board in the WP-5000, refer the Section 5.1.6 to Section 5.1.11.

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

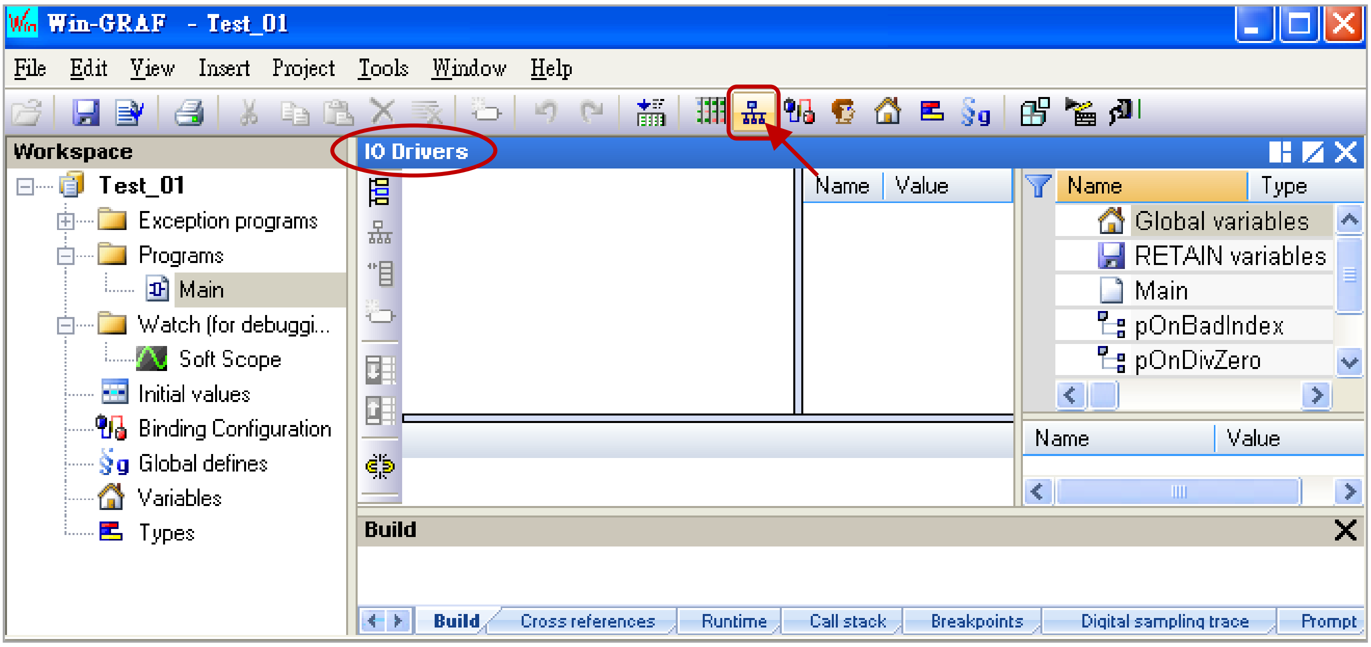
Application Diagram:



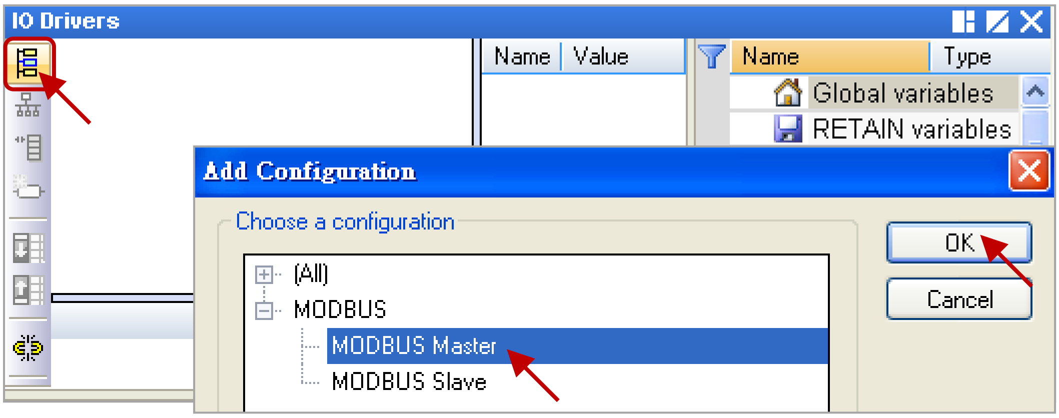
(Refer the P1-1 to view all PAC models)

Follow these steps:

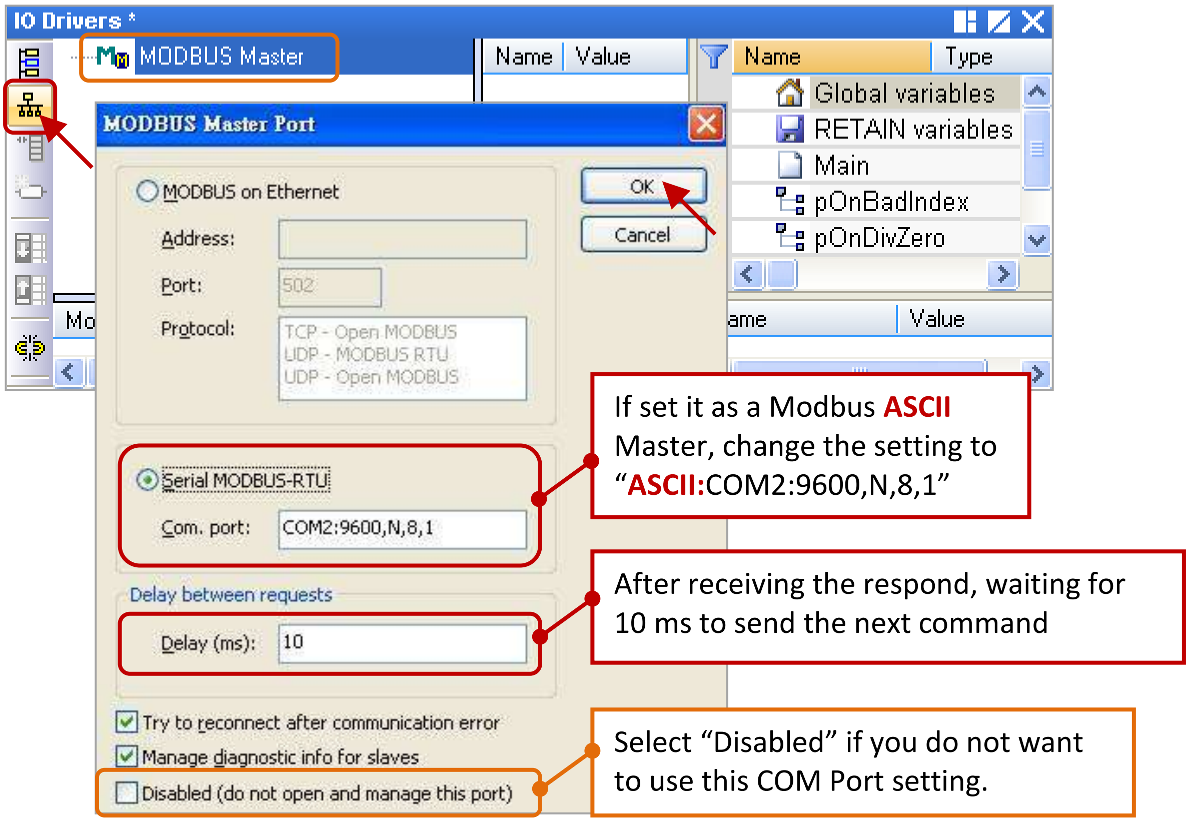
1. Mouse click the "Open Fieldbus Configuration" tool button to open "IO Drivers" window.



2. Click the “Insert Configuration” button on the left of the “IO Drivers” window, then click the “MOSBUS Master” and “OK” to enable the Modbus Master setting.



3. Click the “Insert Master/Port” button on the left side to open the setting window. Then, select the “Serial MODBUS-RTU”, set COM Port (e.g., “COM2:9600,N,8,1”) and Delay time (recommended value: 10 ms, it can be 0 to 10000), and then click “OK”.



○ MODBUS on Ethernet

Address:

Port: 502

● Serial MODBUS-RTU

Com. port: COM2:9600,N,8,1

Delay (ms): 10

☑ Try to reconnect after communication error

☑ Manage diagnostic info for slaves

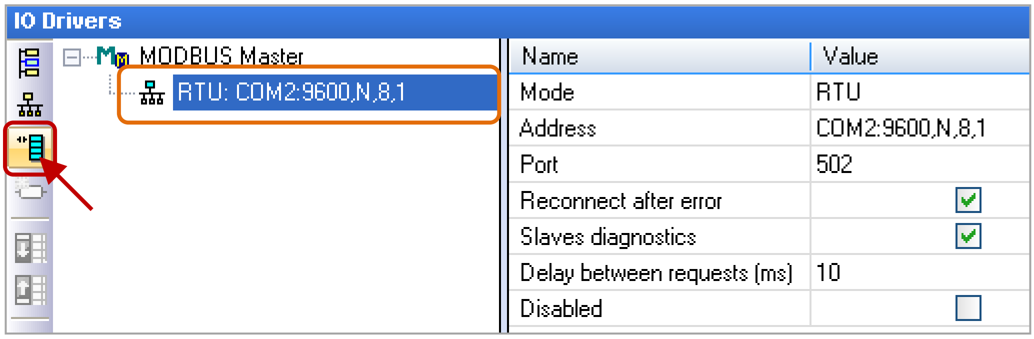
□ Disabled (do not open and manage this port)

If set it as a Modbus **ASCII** Master, change the setting to “**ASCII:**COM2:9600,N,8,1”

After receiving the respond, waiting for 10 ms to send the next command

Select “Disabled” if you do not want to use this COM Port setting.

4. Click the “Insert Slave/Data Block” button on the left side to create a data block.



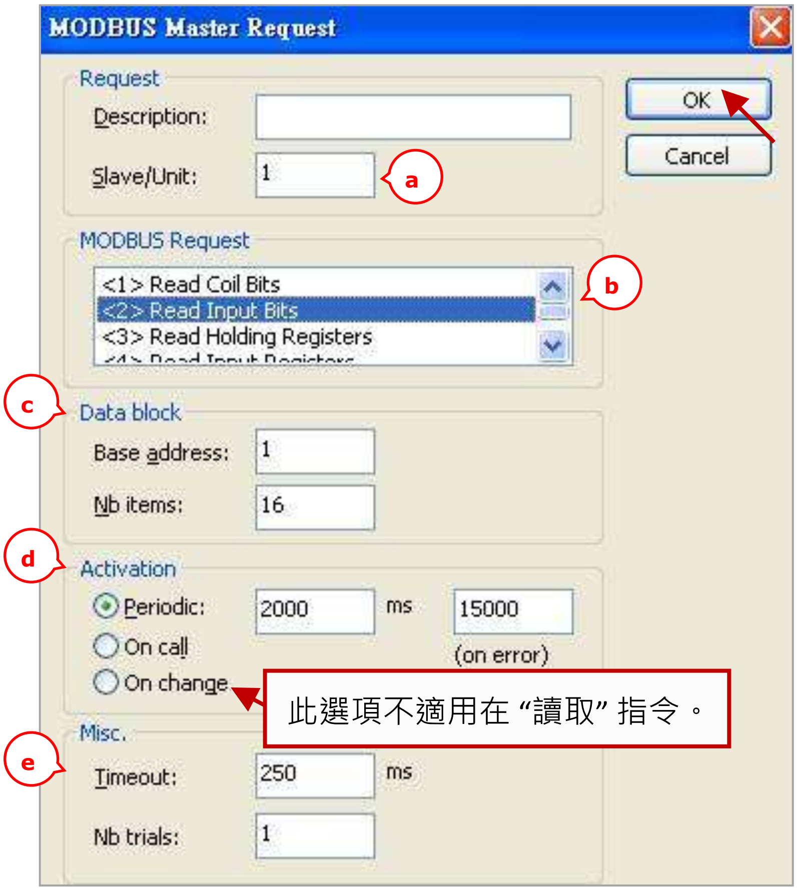
Name	Value
Mode	RTU
Address	COM2:9600,N,8,1
Port	502
Reconnect after error	☑
Slaves diagnostics	☑
Delay between requests (ms)	10
Disabled	□

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

Item	Function Code	Modbus Request	Description
1	2	Read Input-bits	Read the DI data
2	5	Write single coil-bit	Write the DO data
3	4	Read Input Registers	Read the AI data
4	6	Write single holding register	Write the AO data (16-bit)
5	16	Write Holding Registers	Write the AO data (32-bit)

5.1.1 Read the DI data

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



MODBUS Master Request

Request
Description:
Slave/Unit: 1

MODBUS Request
<1> Read Coil Bits
<2> Read Input Bits
<3> Read Holding Registers
<4> Read Input Registers

Data block
Base address: 1
Nb items: 16

Activation
● Periodic: 2000 ms 15000 (on error)
○ On call
○ On change

Misc.
Timeout: 250 ms
Nb trials: 1

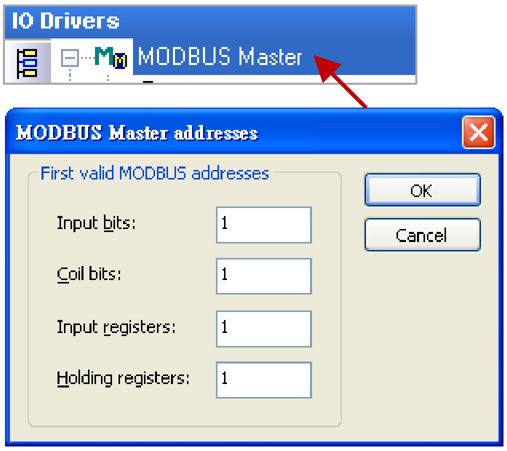
a. Slave/Unit:
Enter the Net-ID of the Slave device.
(In this case, the Net-ID is “1”).

b. MODBUS Request:
Select “<2> Read Input Bits” option.

c. Base address:
Start from “1” by default.
Nb items:
The number of DI signals to read.
(In this case, the number is “16”).

Note:

If you want to change the “Base address”, right-click the “MODBUS Master” and then select the “MODBUS Master Addresses” to modify the value.



MODBUS Master addresses

First valid MODBUS addresses
Input bits: 1
Coil bits: 1
Input registers: 1
Holding registers: 1

d. Activation: The way to send the Modbus request.

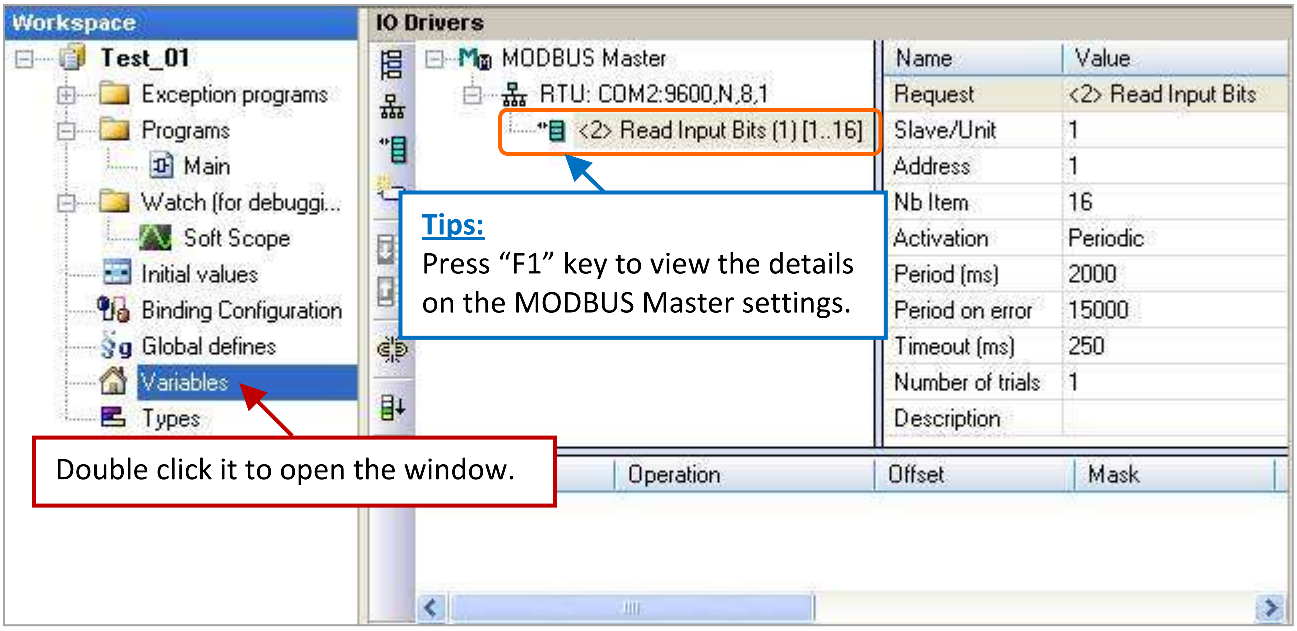
Periodic: Sending the request periodically.
(In this case, to send once every two seconds.)
“on error” means the next sending time when an exception occurred (e.g., 15 seconds).

On call: The request is activated when a program makes function call.

On change: In case of a write request, means that the request is activated each time a variable changes.

e. Timeout: Set a timeout value. (When time-out occurred, it will show the defined error code.)
(The recommended value for the Modbus RTU/ASCII device is 200 to 1000 ms.
In this case, the value is 250 ms)

2. Next, open the “Variables” window and then declare variables that are available for the program.

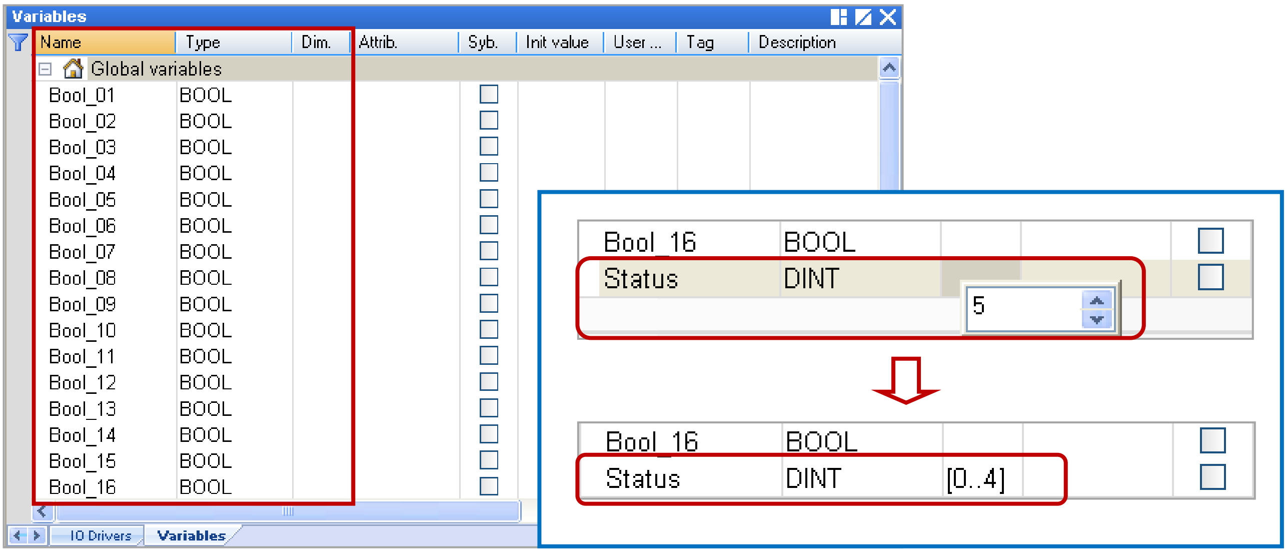


Name	Value
Request	<2> Read Input Bits
Slave/Unit	1
Address	1
Nb Item	16
Activation	Periodic
Period (ms)	2000
Period on error	15000
Timeout (ms)	250
Number of trials	1
Description	

Tips:
Press “F1” key to view the details on the MODBUS Master settings.

Double click it to open the window.

Declaring 16 variables to read data (Name: “Boo_01 to Boo_16”; Type: BOOL) and one array variable to record the state of data access (Name: “Status"; Dim.: 5; Type: DINT). Refer the Sectin 2.3.1 for the way to declare variables , and the figure below shows defined variables.



Name	Type	Dim.	Attrib.	Syb.	Init value	User ...	Tag	Description
Bool_01	BOOL			□				
Bool_02	BOOL			□				
Bool_03	BOOL			□				
Bool_04	BOOL			□				
Bool_05	BOOL			□				
Bool_06	BOOL			□				
Bool_07	BOOL			□				
Bool_08	BOOL			□				
Bool_09	BOOL			□				
Bool_10	BOOL			□				
Bool_11	BOOL			□				
Bool_12	BOOL			□				
Bool_13	BOOL			□				
Bool_14	BOOL			□				
Bool_15	BOOL			□				
Bool_16	BOOL			□				

Bool_16	BOOL		□
Status	DINT	5	□

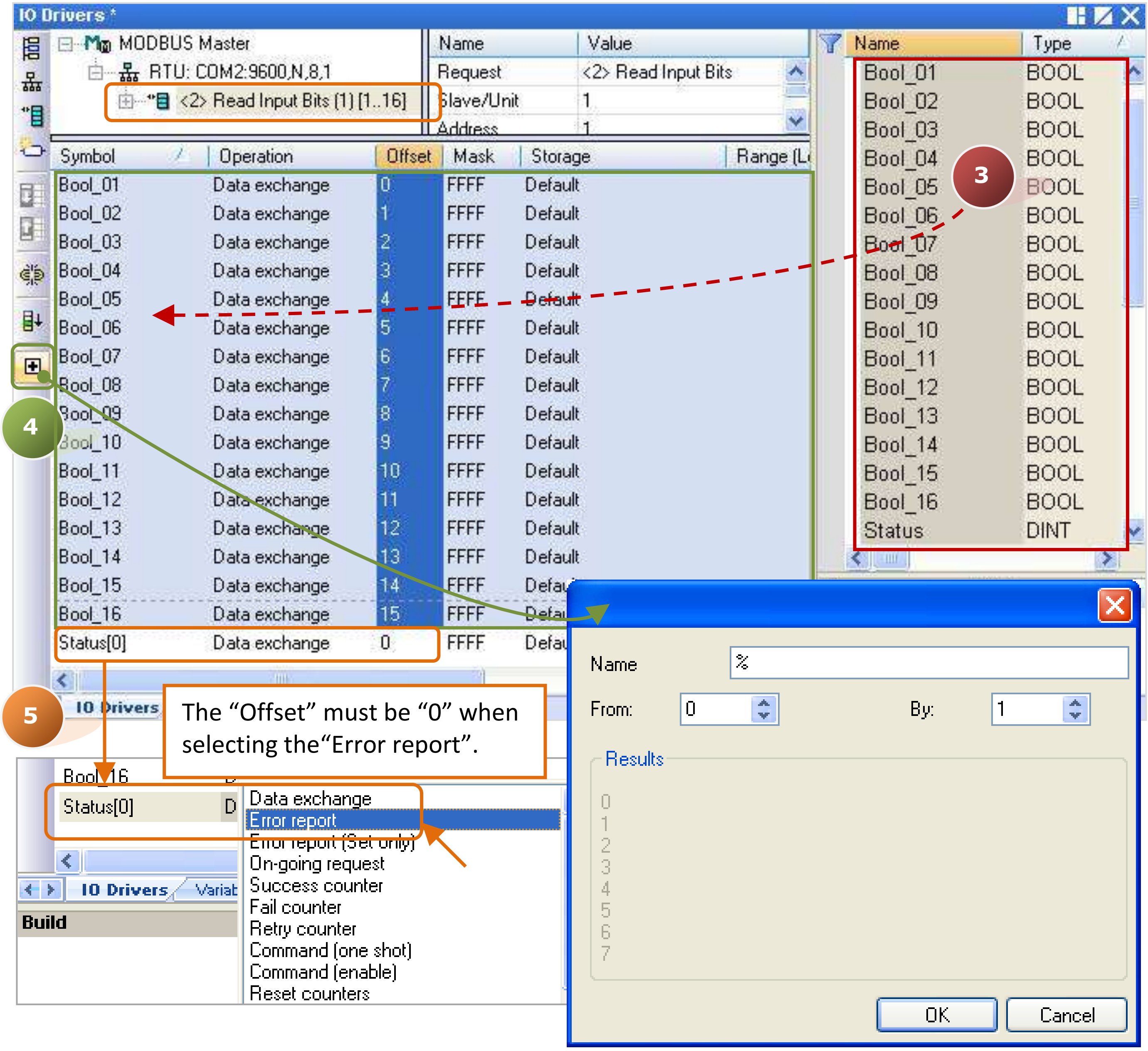
⇩

Bool_16	BOOL		□
Status	DINT	[0..4]	□

3. In the "IO Drivers" window like the figure below, drag all required variables in the Variables Area (i.e., “Bool_01” to “Bool_16” and “Status”) and drop them to the “Symbol” area of the first data block.
Note: The “Status” is an array variable, so, the Status[0] to Status[4] will show on the “Symbol” area.
Click the “Del” key to delete the Status[1] to Status[4].

4. Next, select “Offset” field from “Boo_01” to “Boo_16” and then click the “Iterate Property” button on the left side to set the “Offset” value (From: “0” ; By: “1”, refer the Section 3.1 – Step8).

5. In the “Operation” field, set the “Status[0]” as “Error report” which means the return value is an “Error Code” if a read error occurred and the value will be reset to “0” if read successfully.



Name	Value
Request	<2> Read Input Bits
Slave/Unit	1
Address	1

Symbol	Operation	Offset	Mask	Storage	Range (L
Bool_01	Data exchange	0	FFFF	Default	
Bool_02	Data exchange	1	FFFF	Default	
Bool_03	Data exchange	2	FFFF	Default	
Bool_04	Data exchange	3	FFFF	Default	
Bool_05	Data exchange	4	FFFF	Default	
Bool_06	Data exchange	5	FFFF	Default	
Bool_07	Data exchange	6	FFFF	Default	
Bool_08	Data exchange	7	FFFF	Default	
Bool_09	Data exchange	8	FFFF	Default	
Bool_10	Data exchange	9	FFFF	Default	
Bool_11	Data exchange	10	FFFF	Default	
Bool_12	Data exchange	11	FFFF	Default	
Bool_13	Data exchange	12	FFFF	Default	
Bool_14	Data exchange	13	FFFF	Default	
Bool_15	Data exchange	14	FFFF	Defau	
Bool_16	Data exchange	15	FFFF	Defau	
Status[0]	Data exchange	0	FFFF	Defau	

Name	Type
Bool_01	BOOL
Bool_02	BOOL
Bool_03	BOOL
Bool_04	BOOL
Bool_05	BOOL
Bool_06	BOOL
Bool_07	BOOL
Bool_08	BOOL
Bool_09	BOOL
Bool_10	BOOL
Bool_11	BOOL
Bool_12	BOOL
Bool_13	BOOL
Bool_14	BOOL
Bool_15	BOOL
Bool_16	BOOL
Status	DINT

The “Offset” must be “0” when selecting the“Error report”.

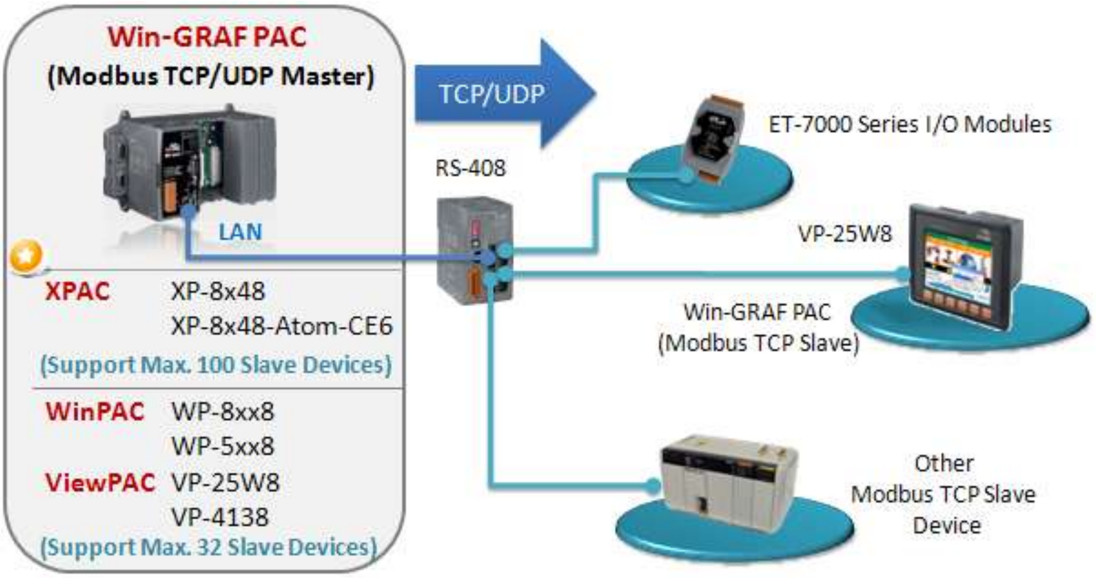
You can also press “F1” in this “IO Drivers” window to see details on Modbus Master Configuration.

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	Server is busy.	130	RS-232 communication error.

Under construction ...

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

Application Diagram:



(Refer the P1-1 to view all PAC models)

Under construction ...

5.3 Connecting the Modbus TCP Slave Device with 2 IP Addresses

Under construction ...

Chapter 6 Retain Variable and Data Storage

Under construction ...

Chapter 7 Data Binding

Under construction ...

Chapter 8 Linking DCON I/O Modules

Under construction ...

Chapter 9 On Line Change

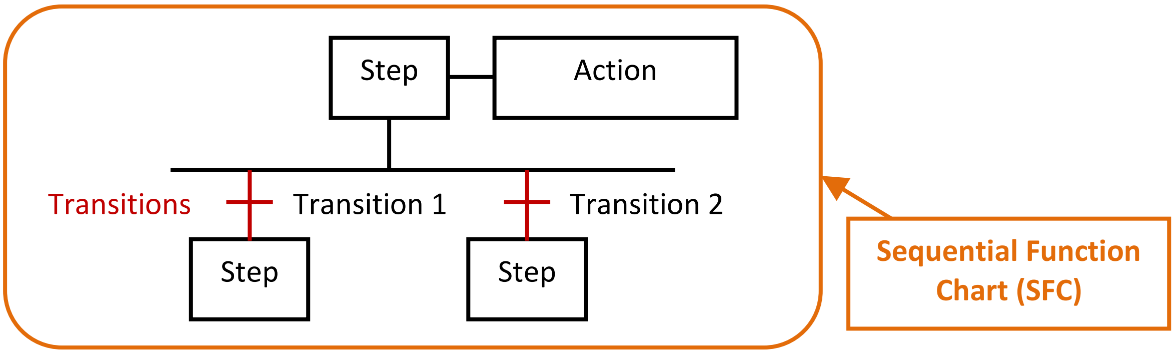
"On Line Change" function allows Win-GRAF PAC change its application to a modified one during the PAC running time. The modified application must be the same name as the original one that currently running in the PAC. The "On Line Change" is primarily used for emergency, such as, when the application is not allowed to be disabled or stopped for a while or cannot find time to replace a new application (for example, the device needs 24 hours operation and cannot be stopped). Except the above situations, it is not recommended to use this function! You had better stop the running application, then download the modified application to the PAC (see Section 2.3.5), which is more safe.

9.1 Limitations of "On Line Change"

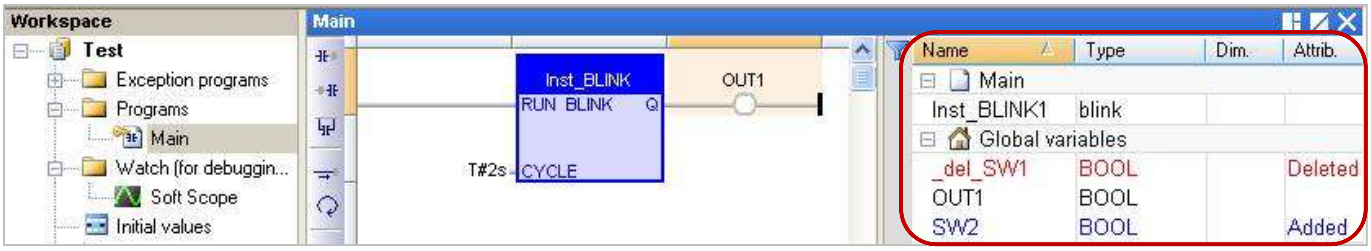
Please notice the important limitations before enabling the "On Line Change"!

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 a SFC transition or the actions of a SFC step.



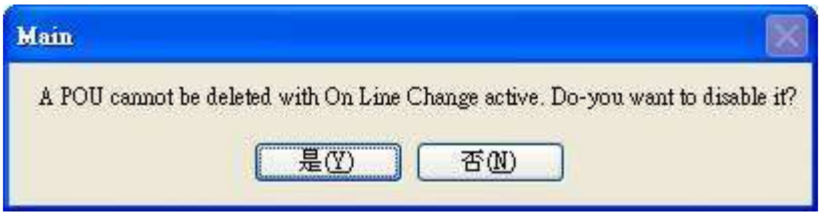
- Create, rename or delete global and local variables.
- Create, rename or delete global and local function block instances.



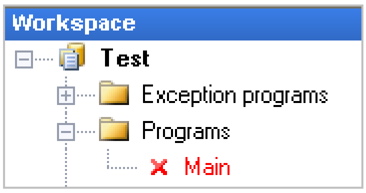
Name	Type	Dim.	Attrib.
Main			
Inst_BLINK1	blink		
Global variables			
_del_SW1	BOOL		Deleted
OUT1	BOOL		
SW2	BOOL		Added

The following kinds of changes are not allowed:

- Create, delete or rename a program. (It will appear a warning message if delete a program.)


Main

A POU cannot be deleted with On Line Change active. Do-you want to disable it?

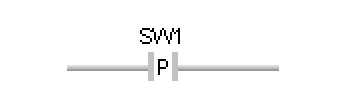
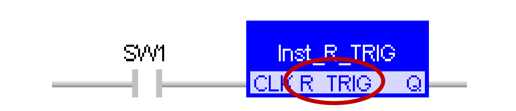
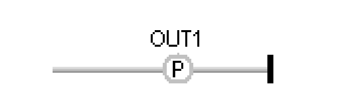
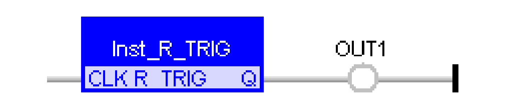
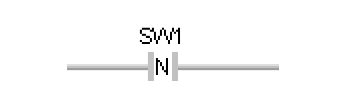
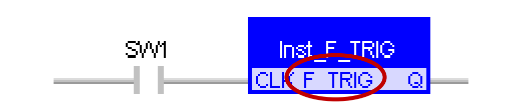
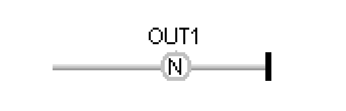
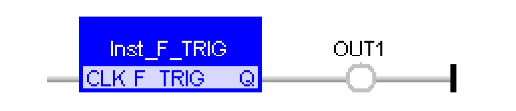


- Change SFC charts.
- Change the local parameters and variables of a UDFB.
- Change the type or dimension (or string length) of a variable or function block instance.
- Change the set of I/O boards.
- Change the definition of RETAIN variables.

In addition, 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		
	Before Enable	**After Enable**
P **(False > True)**		
		
Decreased Pulse Detection		
N **(True > False)**		
		

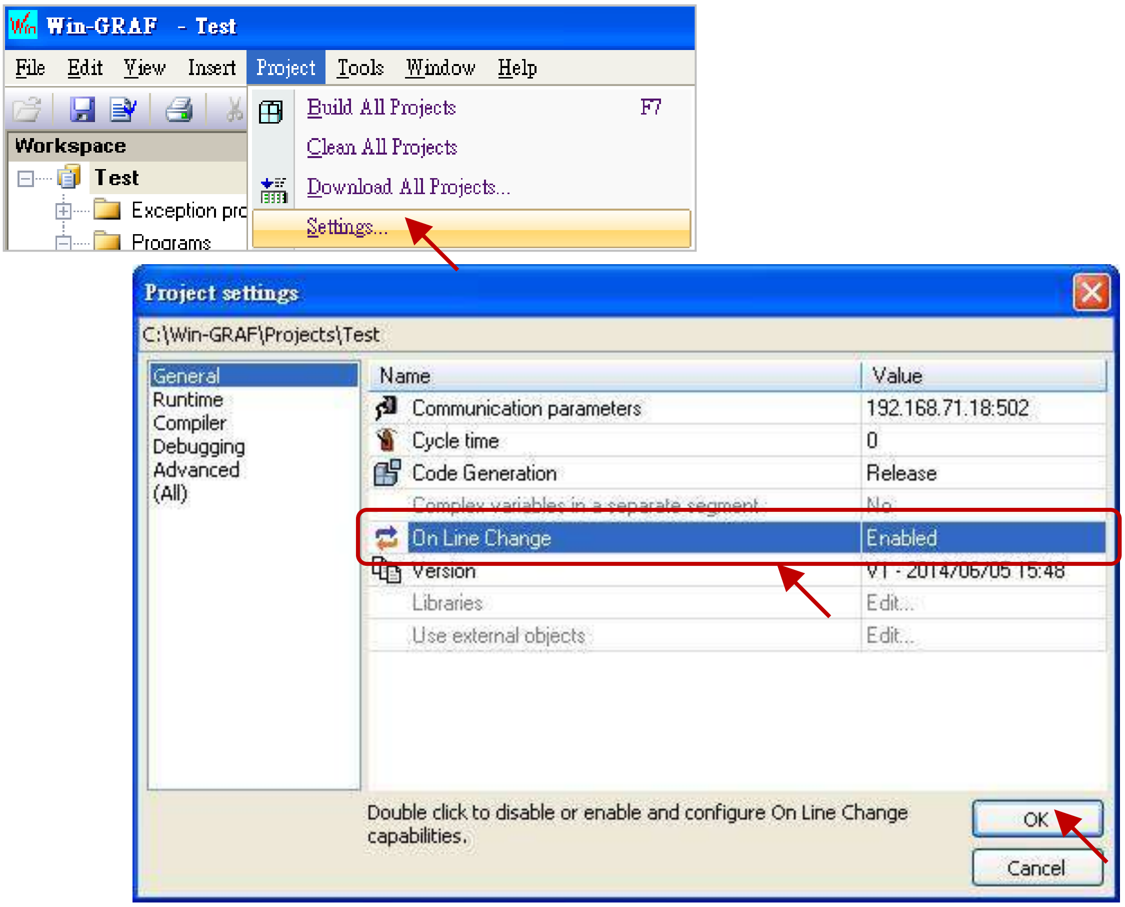
- Loops in FBD with no declared variable linked.

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

9.2 Using "On Line Change"

Enable The "On Line Change" Function:

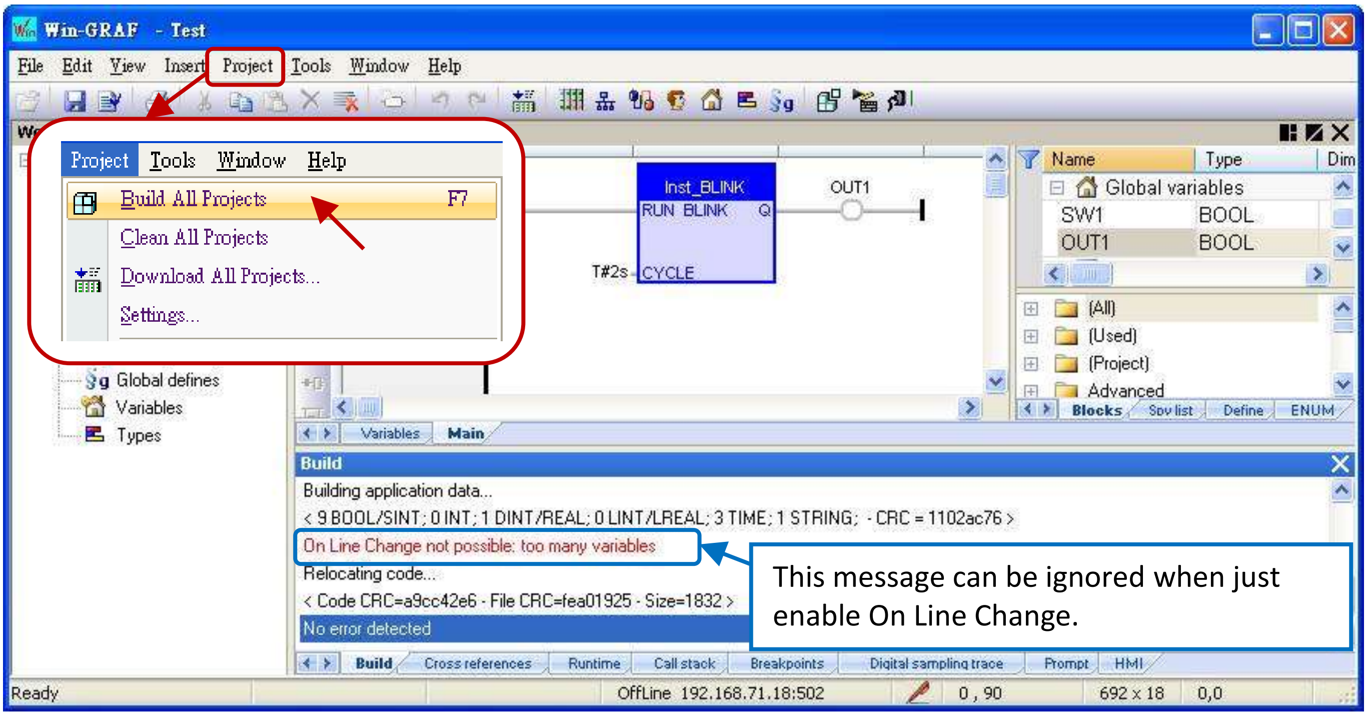
1. Mouse click "Project > Settings..." from the menu bar, then double click "On Line Change" item to set it "Enabled".



Name	Value
Communication parameters	192.168.71.18:502
Cycle time	0
Code Generation	Release
Complex variables in a separate segment	No
On Line Change	Enabled
Version	V1 - 2014/06/05 15:48
Libraries	Edit...
Use external objects	Edit...

Double click to disable or enable and configure On Line Change capabilities.

2. Click "Project > Build All Projects" from the menu bar. Must compile the program first, then can set up the following steps.



```
Building application data...
< 9 BOOL/SINT; 0 INT; 1 DINT/REAL; 0 LINT/LREAL; 3 TIME; 1 STRING; - CRC = 1102ac76 >
On Line Change not possible: too many variables
Relocating code...
< Code CRC=a9cc42e6 - File CRC=fea01925 - Size=1832 >
No error detected
```

This message can be ignored when just enable On Line Change.

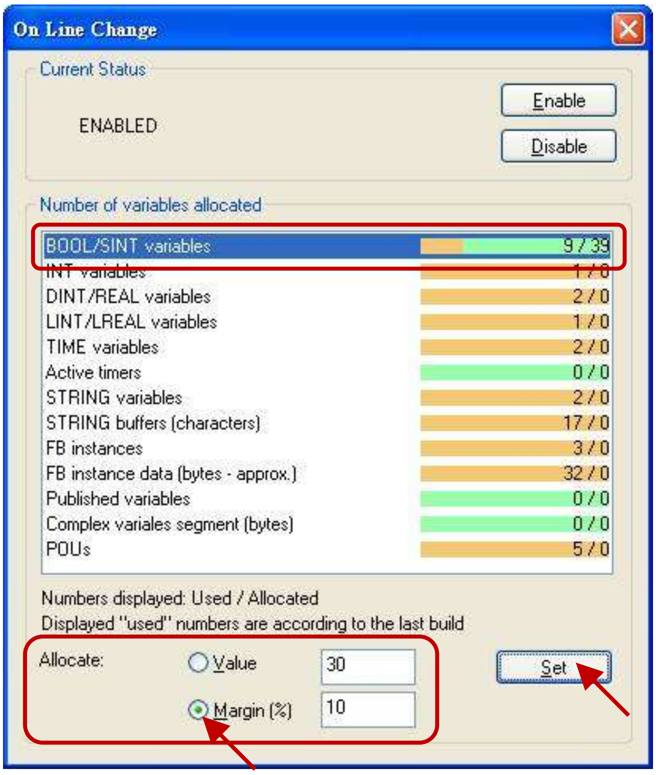
Setup the Number of Variables:

When the "On Line Change" is enabled, you have to set up the number of variables reserved for the declaration of the new variables and Function Blocks for future on-line change usage.

3. The same as the step 1, mouse click "Project > Settings..." from the menu bar, then double click on the "On Line Change" to show the setting window.
   Please set the needed new number in the "Value" or "Margin" fields.
   Note: If both "Value" and "Margin" have set values, it will use the larger value. In this example, "Value" is set as "30" and "Margin" is set as "10", then the displayed value x 10% is smaller than 30, so it will use the larger value "30".

4. Click the needed Variable Type, then click the "Set" button.
   (E.g., Click "BOOL/SINT variables" and "Set" button, then the number become 9 + 30 = 39.)
   Note: "STRING buffers (characters)", "FB instance data (bytes – approx.)" and "Complex variables segment (bytes) " need to set a larger number (This example uses "5000").

5. After setting (as the picture below), click the "X" in the upper right corner to exit the setting.



On Line Change

Current Status

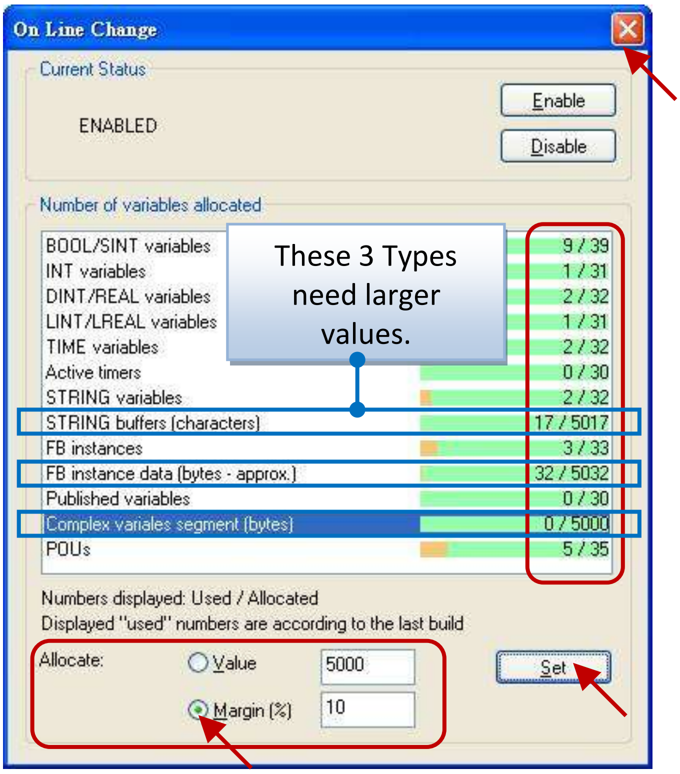
ENABLED

Number of variables allocated

BOOL/SINT variables	9 / 39
INT variables	1 / 0
DINT/REAL variables	2 / 0
LINT/LREAL variables	1 / 0
TIME variables	2 / 0
Active timers	0 / 0
STRING variables	2 / 0
STRING buffers (characters)	17 / 0
FB instances	3 / 0
FB instance data (bytes - approx.)	32 / 0
Published variables	0 / 0
Complex variales segment (bytes)	0 / 0
POUs	5 / 0

Numbers displayed: Used / Allocated
Displayed "used" numbers are according to the last build

Allocate: ○ Value 30
● Margin (%) 10



On Line Change

Current Status

ENABLED

Number of variables allocated

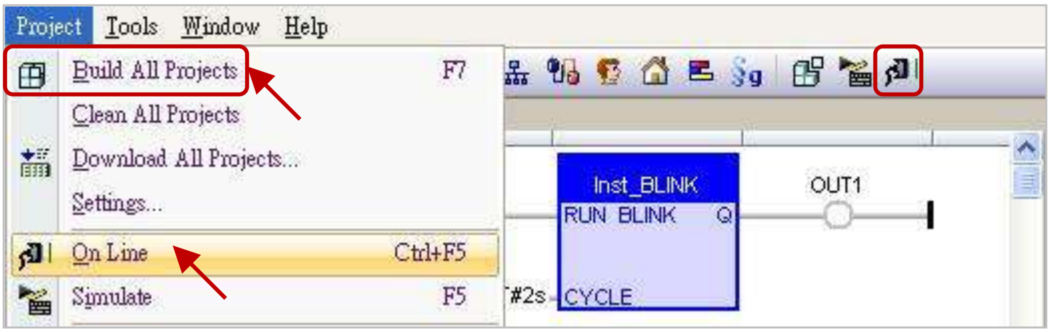
BOOL/SINT variables	9 / 39
INT variables	1 / 31
DINT/REAL variables	2 / 32
LINT/LREAL variables	1 / 31
TIME variables	2 / 32
Active timers	0 / 30
STRING variables	2 / 32
STRING buffers (characters)	17 / 5017
FB instances	3 / 33
FB instance data (bytes - approx.)	32 / 5032
Published variables	0 / 30
Complex variales segment (bytes)	0 / 5000
POUs	5 / 35

These 3 Types need larger values.

Numbers displayed: Used / Allocated
Displayed "used" numbers are according to the last build

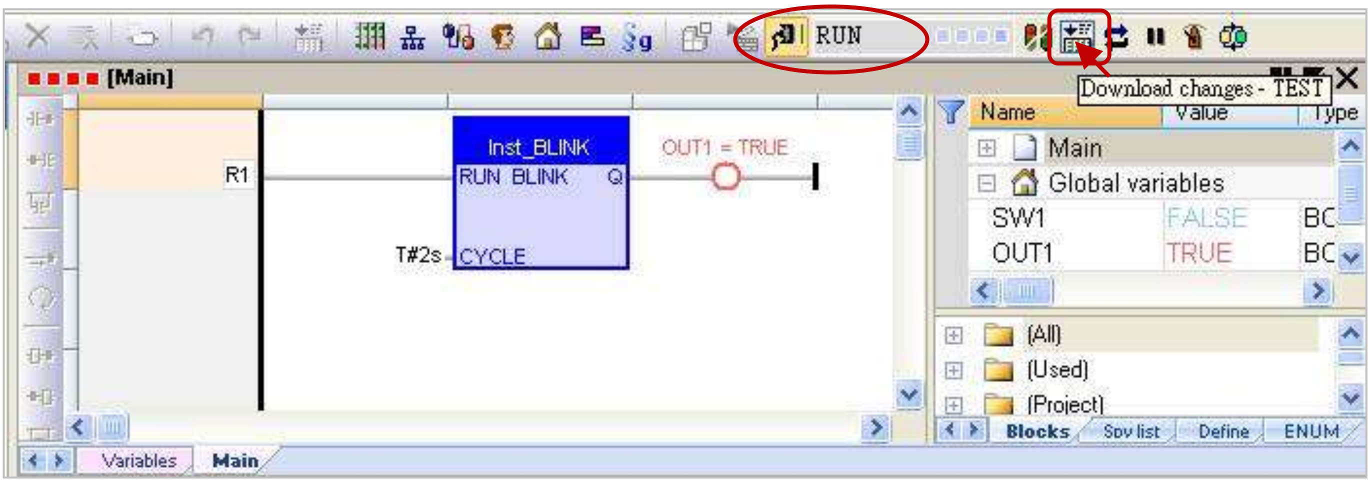
Allocate: ○ Value 5000
● Margin (%) 10

6. Click "Project > Build All Projects" from the menu bar, compile the program again. Then click "On Line" or click the tool icon to link to the PAC. (Refer the Section 2.3.5.)



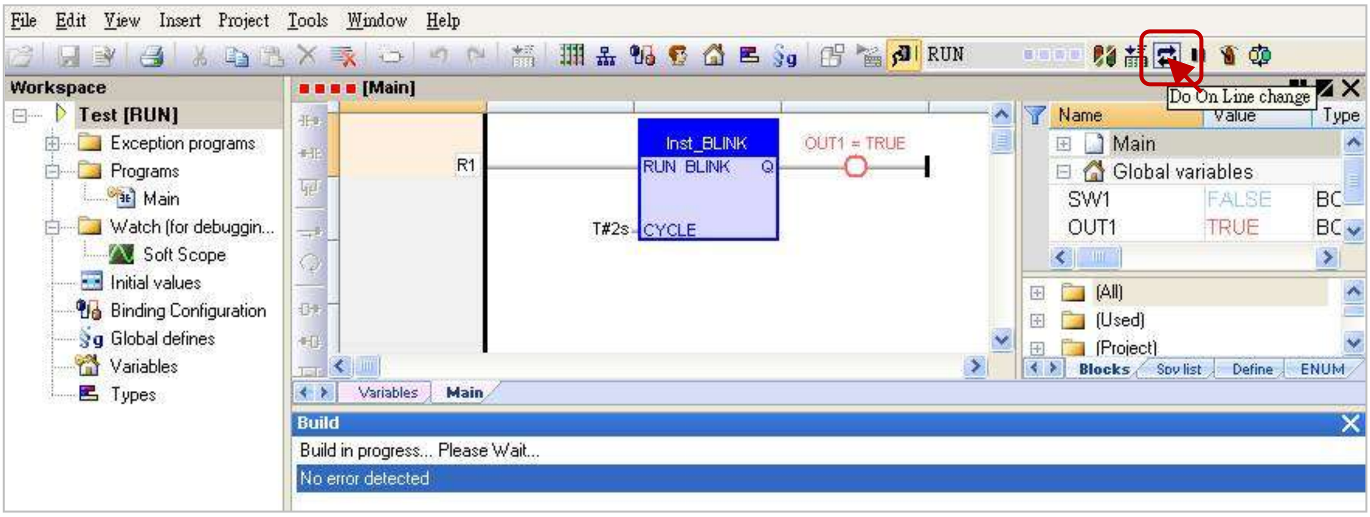
7. After successfully link to the PAC, click the tool icon "Download changes" to download the program to the PAC.

Note: "On Line Change" is only suitable for the program that just has a little change (Do not need to stop the application). If the program name is different from the running one in the PAC, the user has to stop and download the program again (refer the Appendix B).



8. Click on the tool icon "Do On Line change" to execute the program.

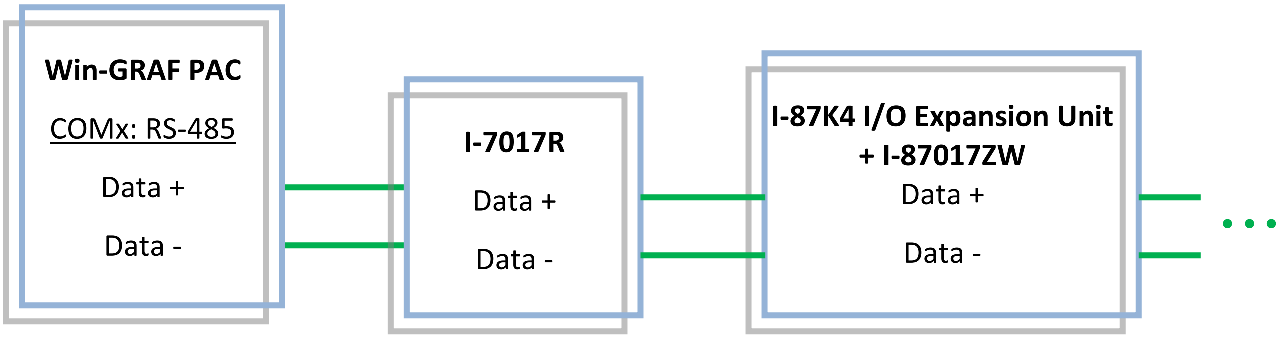
Note: After executing the "On Line Change", there are some using restrictions for protecting the system normal operation (refer the Section 9.1). So, make sure the program is correct, then perform this function.



Build in progress... Please Wait...
No error detected

10.1 AI Data Conversion

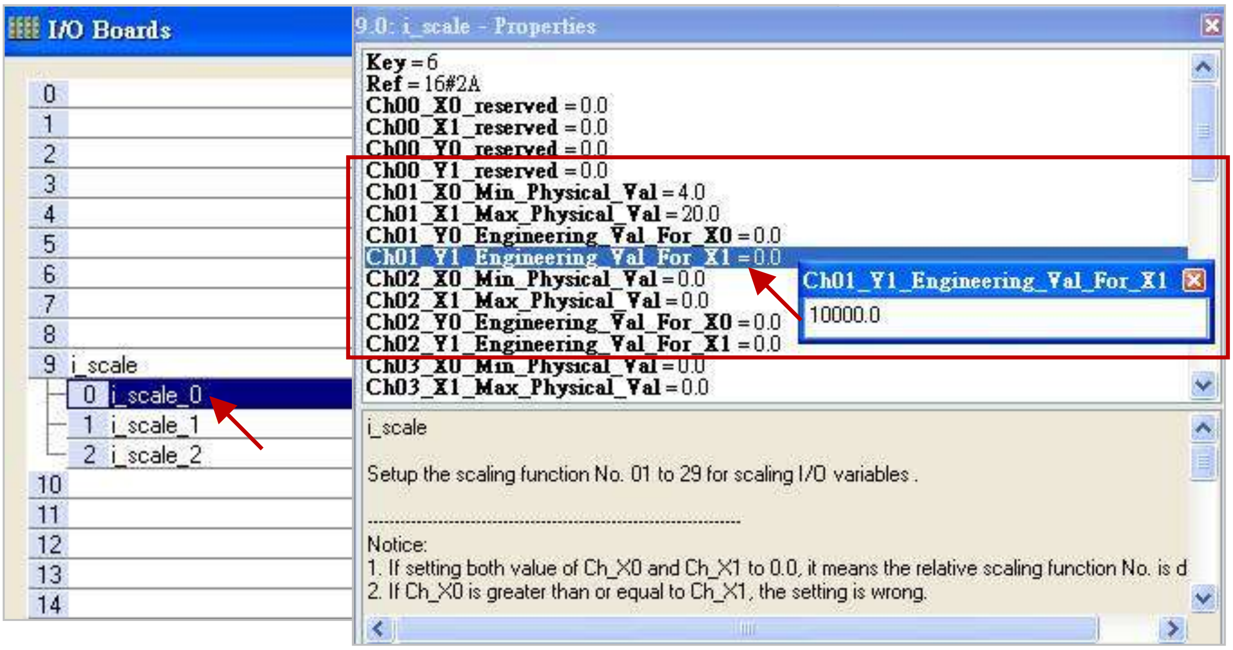
If you are using AI modules in the PAC's Slot 0 to 7 (for example: I-8017HW), and want to convert the AI input signal (for example: "4 to 20 mA" or "0 to 10 V") to the user engineering value (for example: 0 to 10000), refer the Section 4.3. However, if you are using the remote AI modules (e.g., through PAC's RS-485 Port to connect to the I-87017ZW or I-7017R module), you can refer the following settings:



1. First, connect "i_scale" in the "I/O Boards" window, and double click on "i_scale_x" to open the Properties window (refer the Section 4.2 for detail steps)
   Note: "I/O Boards" supports ONLY ONE "i_scale" (DO NOT connect 2 or more "i_scale").

2. Set up the number and value of the Conversion Function that need to be enabled.
   (E.g., Use Function 1 to convert "4 to 20 mA" into "0 to 10000").

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"



```
Key = 6
Ref = 16#2A
Ch00_X0_reserved = 0.0
Ch00_X1_reserved = 0.0
Ch00_Y0_reserved = 0.0
Ch00_Y1_reserved = 0.0
Ch01_X0_Min_Physical_Val = 4.0
Ch01_X1_Max_Physical_Val = 20.0
Ch01_Y0_Engineering_Val_For_X0 = 0.0
Ch01_Y1_Engineering_Val_For_X1 = 0.0
Ch02_X0_Min_Physical_Val = 0.0
Ch02_X1_Max_Physical_Val = 0.0
Ch02_Y0_Engineering_Val_For_X0 = 0.0
Ch02_Y1_Engineering_Val_For_X1 = 0.0
Ch03_X0_Min_Physical_Val = 0.0
Ch03_X1_Max_Physical_Val = 0.0
```

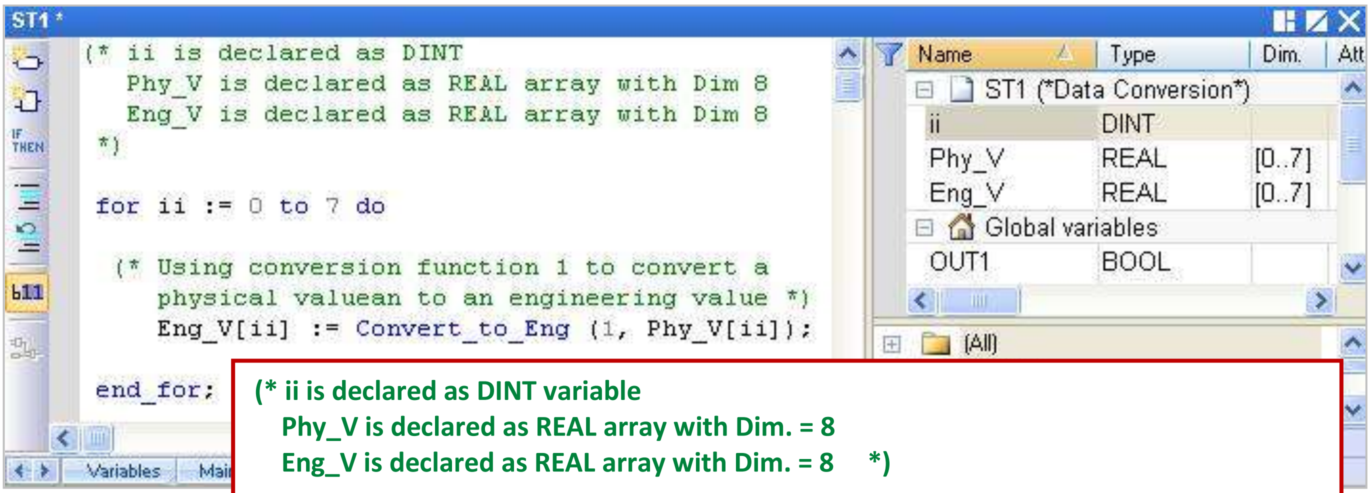
Ch01_Y1_Engineering_Val_For_X1: 10000.0

i_scale

Setup the scaling function No. 01 to 29 for scaling I/O variables .

Notice:
1. If setting both value of Ch_X0 and Ch_X1 to 0.0, it means the relative scaling function No. is d
2. If Ch_X0 is greater than or equal to Ch_X1, the setting is wrong.

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]). (Refer the Section 2.3.3 for detail setting steps.)



```
(* ii is declared as DINT
   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 1 to convert a
     physical valuean to an engineering value *)
     Eng_V[ii] := Convert_to_Eng (1, Phy_V[ii]);

end_for;
```

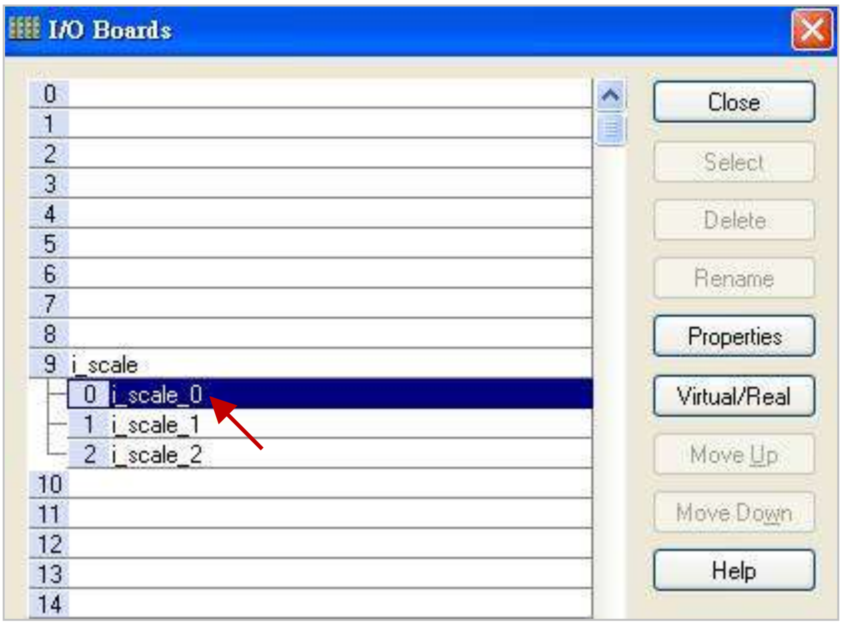
Name	Type	Dim.	Att
ST1 (*Data Conversion*)			
ii	DINT		
Phy_V	REAL	[0..7]	
Eng_V	REAL	[0..7]	
Global variables			
OUT1	BOOL		

```
(* 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 1 to convert a physical value to an engineering value *)
   Eng_V[ii] := Convert_to_Eng (1, Phy_V[ii]);

end_for;
```

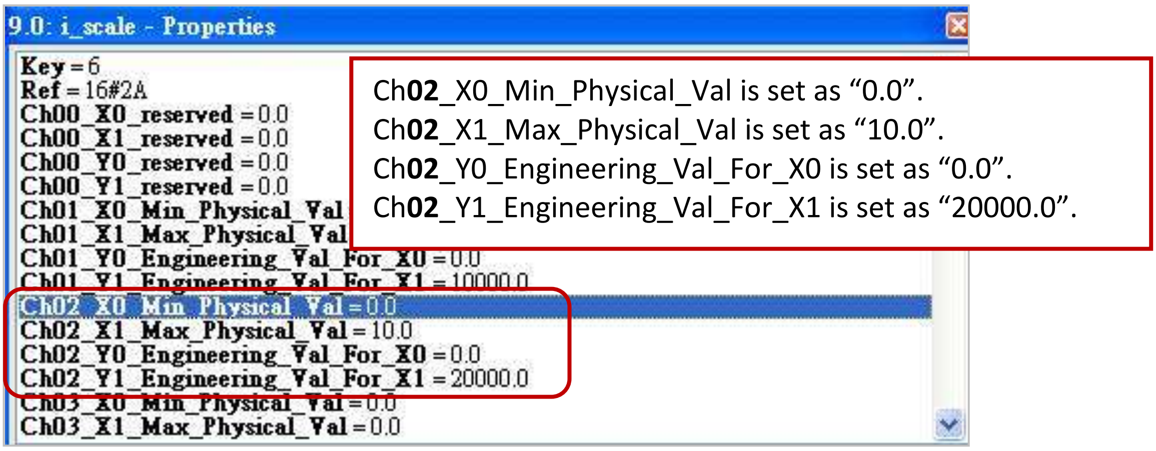
10.2 AO Data Conversion

If you are using AO modules in the PAC's Slot 0 to 7 (for example: I-8024W), and want to convert the engineering value to the AO output signal (e.g., convert "0 to 20000" to "0 to 10 V"), refer the Section 4.4. However, if you are using the DCON remote AO modules (e.g., through PAC's RS-485 Port to connect to the I-7024 module), you can refer to the following settings:

1. Connect "i_scale" in the "I/O Boards" window, and double click "i_scale_x" to open the Properties window (refer the Section 10.1 for detail steps)
   Note: "I/O Boards" supports ONLY ONE "i_scale" (DO NOT connect 2 or more "i_scale").



2. Set up the number and value of the Conversion Function (e.g., use Function 2 to convert "0 to 20000" to "0 to 10 V").

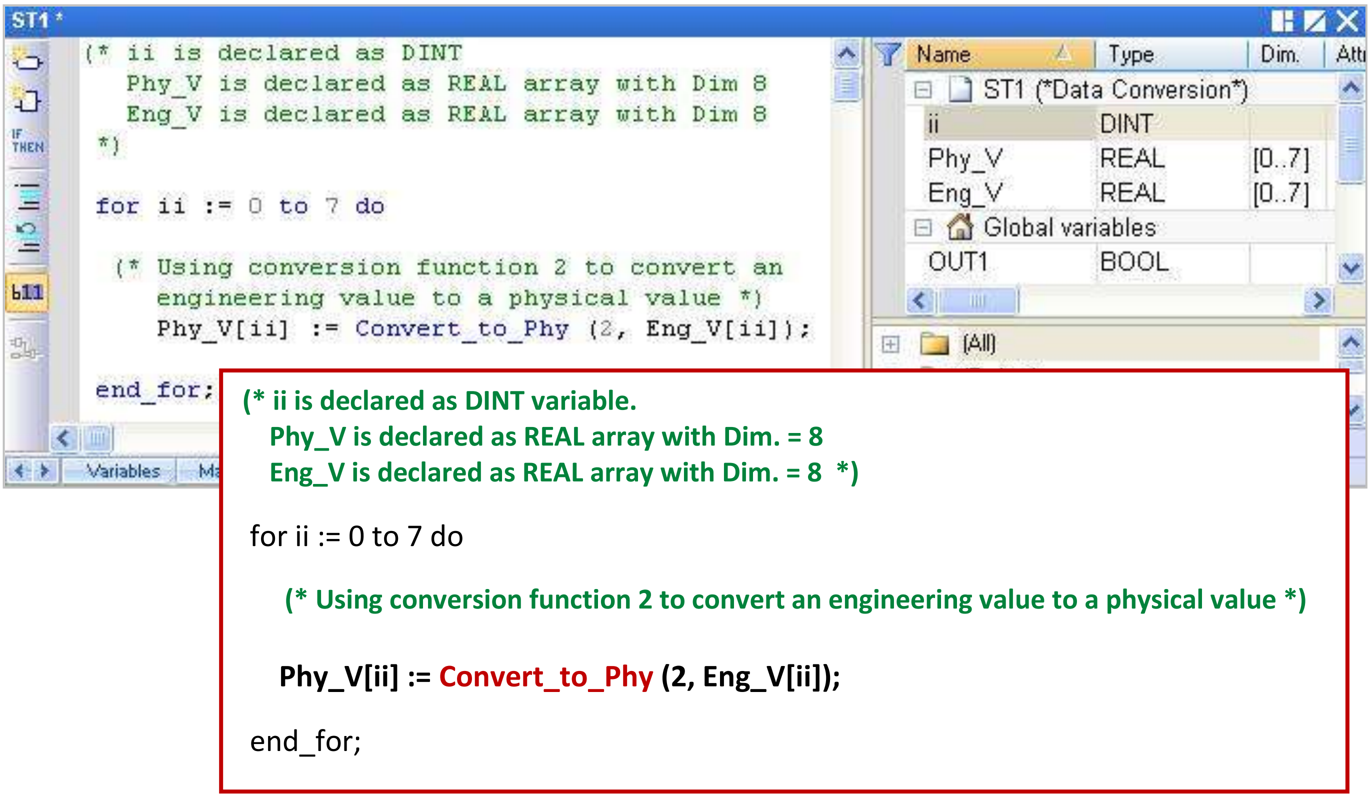


9.0: i_scale - Properties

```
Key = 6
Ref = 16#2A
Ch00_X0_reserved = 0.0
Ch00_X1_reserved = 0.0
Ch00_Y0_reserved = 0.0
Ch00_Y1_reserved = 0.0
Ch01_X0_Min_Physical_Val
Ch01_X1_Max_Physical_Val
Ch01_Y0_Engineering_Val_For_X0 = 0.0
Ch01_Y1_Engineering_Val_For_X1 = 10000.0
Ch02_X0_Min_Physical_Val = 0.0
Ch02_X1_Max_Physical_Val = 10.0
Ch02_Y0_Engineering_Val_For_X0 = 0.0
Ch02_Y1_Engineering_Val_For_X1 = 20000.0
Ch03_X0_Min_Physical_Val = 0.0
Ch03_X1_Max_Physical_Val = 0.0
```

Ch**02**_X0_Min_Physical_Val is set as “0.0”.
Ch**02**_X1_Max_Physical_Val is set as “10.0”.
Ch**02**_Y0_Engineering_Val_For_X0 is set as “0.0”.
Ch**02**_Y1_Engineering_Val_For_X1 is set as “20000.0”.

3. Edit an ST Program to convert an engineering value to a physical value (e.g., Eng_V[0] to [7]). (Refer the Section 2.3.3 for detail setting steps.)



ST1 *

```
(* ii is declared as DINT
   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;
```

Name	Type	Dim.	Attr
ST1 (*Data Conversion*)			
ii	DINT		
Phy_V	REAL	[0..7]	
Eng_V	REAL	[0..7]	
Global variables			
OUT1	BOOL		

(* 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 Data Type Conversion

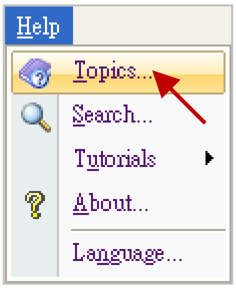
When different types of data want to do "+, -, *, /" calculation or ">, <, =, <=, > =, <> (not equal)" operation, or when the variable types of the parameters in the function are different, you must first use the type conversion function (in the following table) to convert them into the same data type, then can use the data normally.

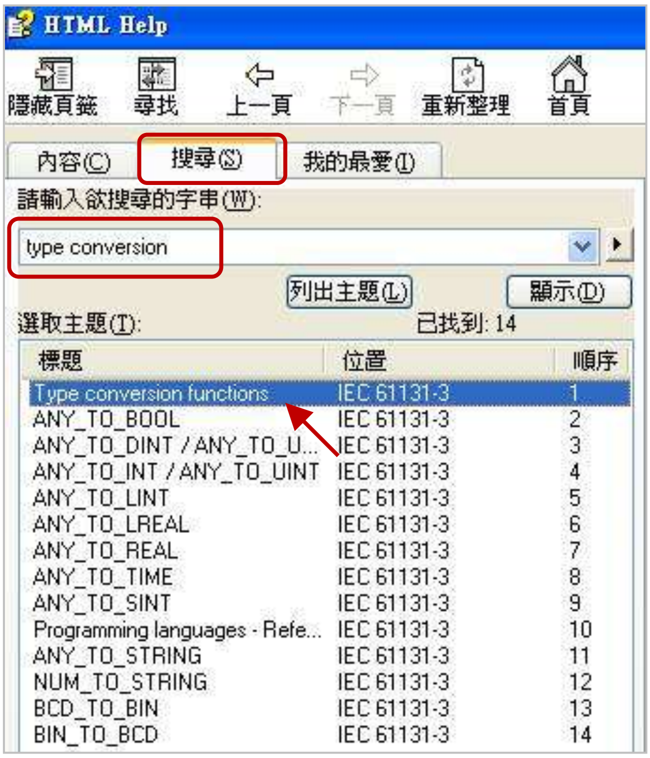
Data Conversion Functions	Descriptions	Data Conversion Functions	Descriptions
ANY_TO_BOOL	Convert to Boolean	ANY_TO_REAL	Convert to Real
ANY_TO_SINT	Convert to Short Integer (8-bit)	ANY_TO_LREAL	Convert to Double
ANY_TO_INT	Convert to Integer (16-bit)	ANY_TO_STRING	Convert to String
ANY_TO_DINT	Convert to Long Integer (32-bit – Default)	NUM_TO_STRING	Convert Number to String. Can set the decimal digital number after converting
ANY_TO_LINT	Convert to Large Integer (64-bit)	ATOH	Convert Hexadecimal String to Integer
ANY_TO_TIME	Convert to Timer	HTOA	Convert Integer to Hexadecimal String

For example, the following ST program will convert the DINT variable to a Real first, and then do the calculation.

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

You can open the "HTML Help" from the menu bar, and enter the searching key word you want to see the detail setup instructions.





type conversion

已找到: 14

標題	位置	順序
Type conversion functions	IEC 61131-3	1
ANY_TO_BOOL	IEC 61131-3	2
ANY_TO_DINT / ANY_TO_U...	IEC 61131-3	3
ANY_TO_INT / ANY_TO_UINT	IEC 61131-3	4
ANY_TO_LINT	IEC 61131-3	5
ANY_TO_LREAL	IEC 61131-3	6
ANY_TO_REAL	IEC 61131-3	7
ANY_TO_TIME	IEC 61131-3	8
ANY_TO_SINT	IEC 61131-3	9
Programming languages - Refe...	IEC 61131-3	10
ANY_TO_STRING	IEC 61131-3	11
NUM_TO_STRING	IEC 61131-3	12
BCD_TO_BIN	IEC 61131-3	13
BIN_TO_BCD	IEC 61131-3	14

10.4 BCD Conversion

BCD 4-bit code is used to represent decimal numbers from 0 to 9. Suppose there is a decimal value "132", if converted to BCD code is "000100110010" and if converted to binary is 10000100 (e.g., $2^7 + 2^2$ = 128 + 4 = 132).

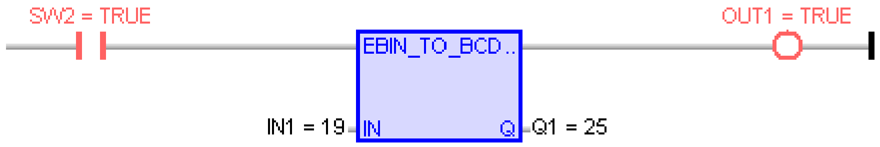
Decimal	BCD				Descriptions
	2^3	2^2	2^1	2^0	
0	0	0	0	0	0
1	0	0	0	1	$2^0 = 1$
2	0	0	1	0	$2^1 = 2$
3	0	0	1	1	$2^1 + 2^0 = 3$
4	0	1	0	0	$2^2 = 4$
5	0	1	0	1	$2^2 + 2^0 = 5$
6	0	1	1	0	$2^2 + 2^1 = 6$
7	0	1	1	1	$2^2 + 2^1 + 2^0 = 7$
8	1	0	0	0	$2^3 = 8$
9	1	0	0	1	$2^3 + 2^0 = 9$

NOTE:
BCD code can only represent numbers from 0 to 9.
It can not be used for these six values (1010, 1011, 1100, 1101, 1110, 1111), and will return "0".

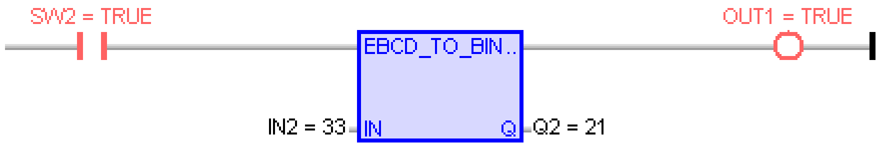
The function table below can be used for BCD (Binary Coded Decimal) value conversion.

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

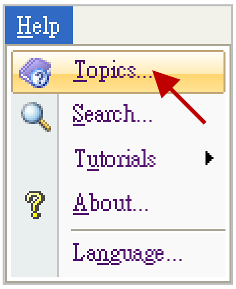
BIN _TO_BCD:
19$_{(10)}$ = 0001 1001$_{(BCD)}$
= $2^4 + 2^3 + 2^0$ = 16 + 8 + 1 = **25**

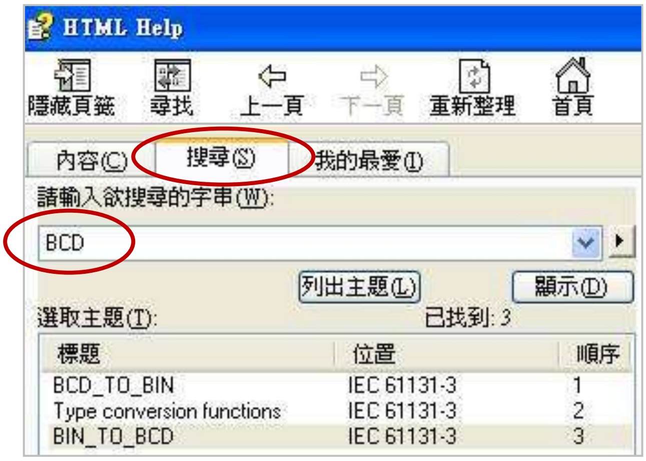


BCD_TO_BIN:
33$_{(10)}$ = 0010 0001$_{(2)}$ = **21**$_{(BCD)}$
15$_{(10)}$ = 0000 ~~1111~~$_{(2)}$ = **0**$_{(BCD)}$



You can open the "HTML Help" from the menu bar, and enter the searching key word you want to see the detail setup instructions.



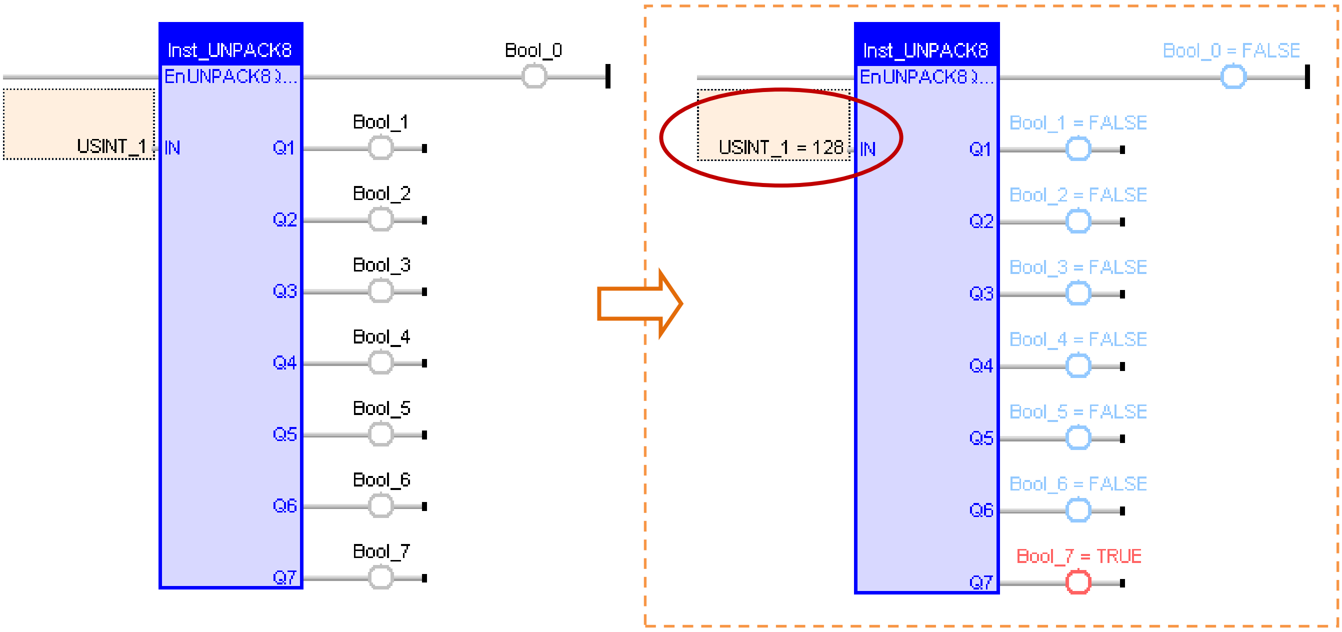


標題	位置	順序
BCD_TO_BIN	IEC 61131-3	1
Type conversion functions	IEC 61131-3	2
BIN_TO_BCD	IEC 61131-3	3

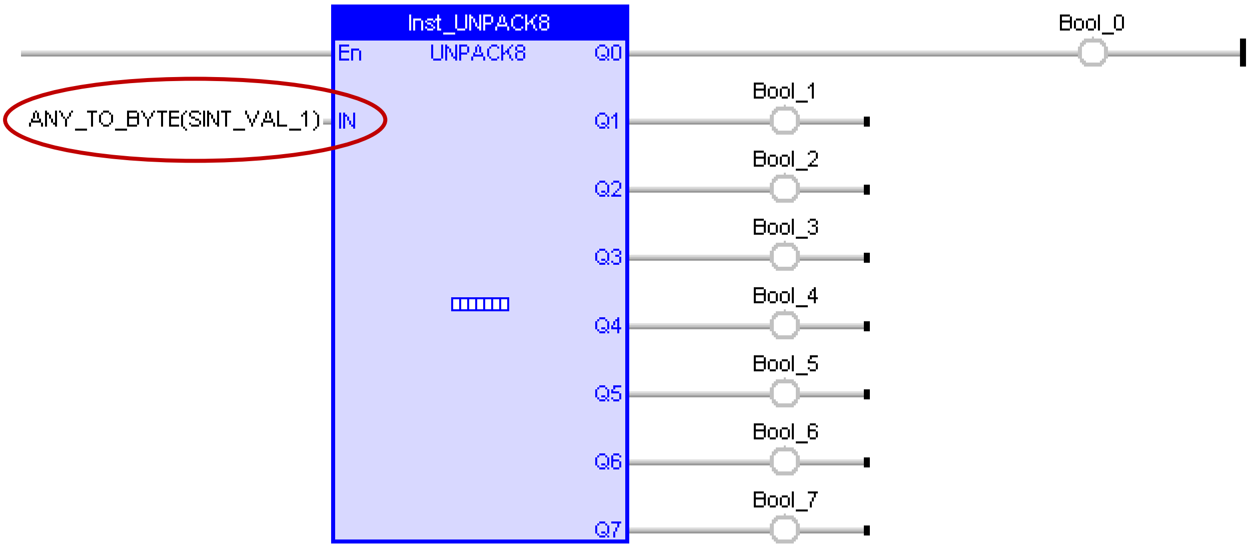
10.5 Pack/Unpack Integer or Boolean

Unpack Integer to Boolean:

If want to unpack one BYTE (or USINT, range: 0 to 255) to 8 Booleans, you can use "UNPACK8" Function Block.



If want to unpack one SINT to 8 Booleans, you must first use the ST program "ANY_TO_BYTE ()" to convert the SINT to be a BYTE type, as follows:



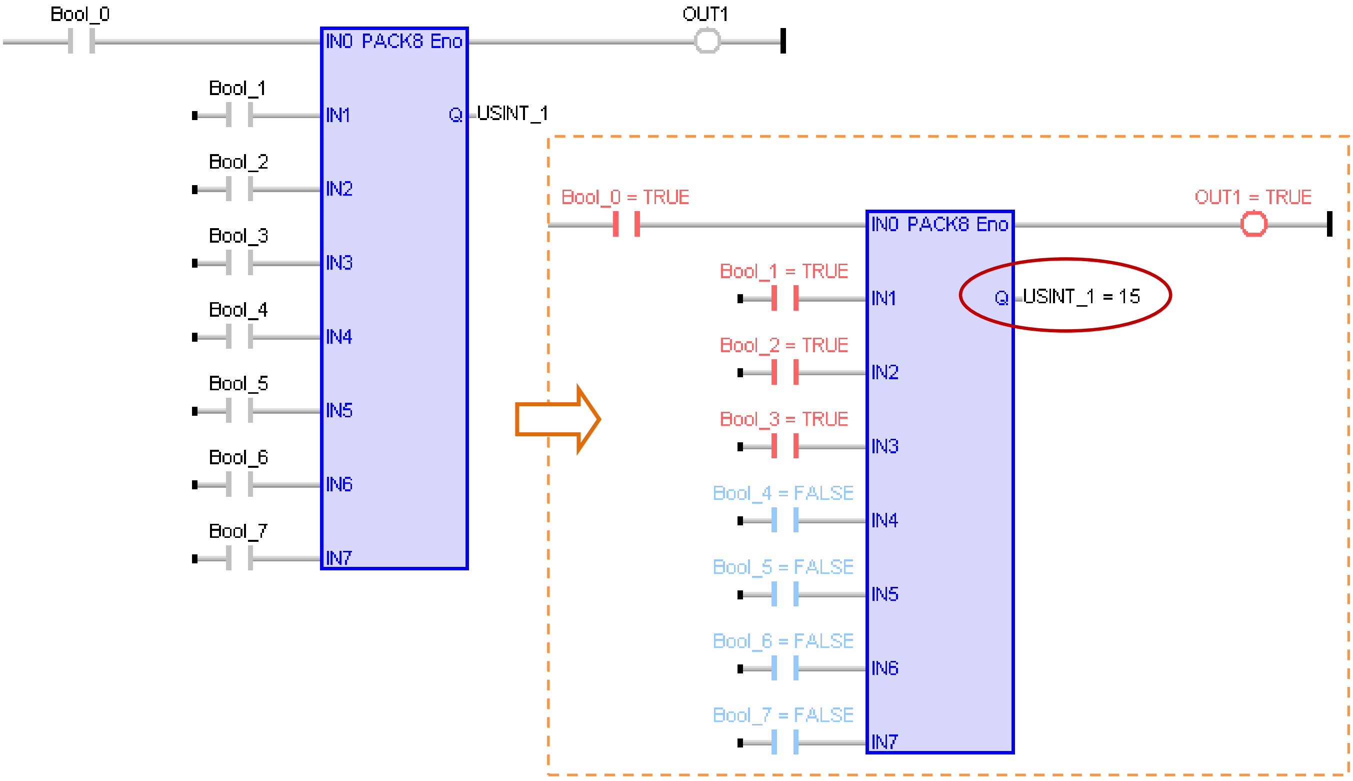
Pack Boolean Into Integer:

If want to pack 8 Booleans into one BYTE (or USINT, range: 0 to 255), you can use "PACK8" Function.

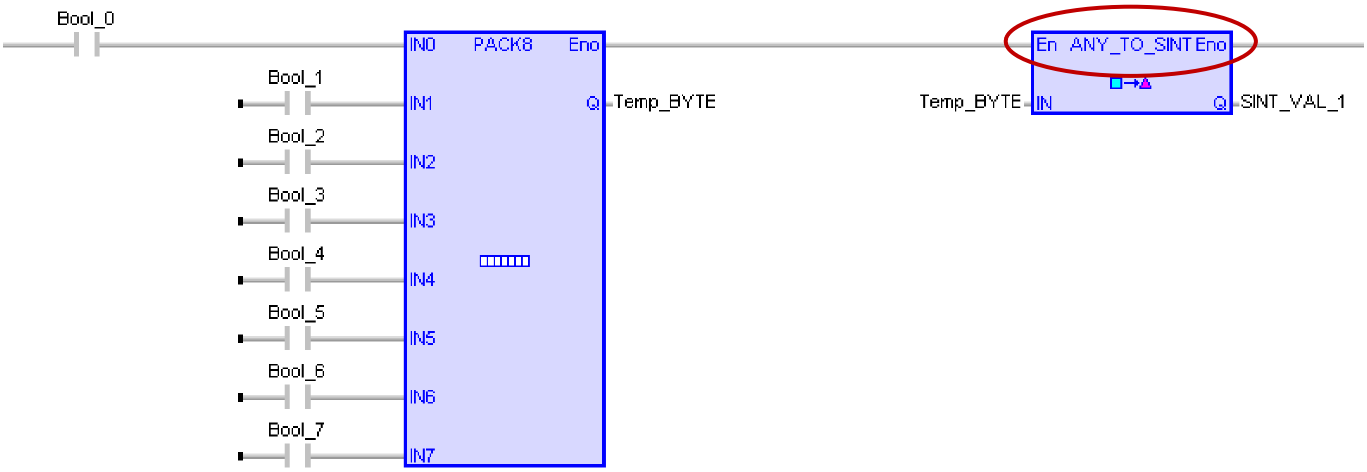
ST Program:

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

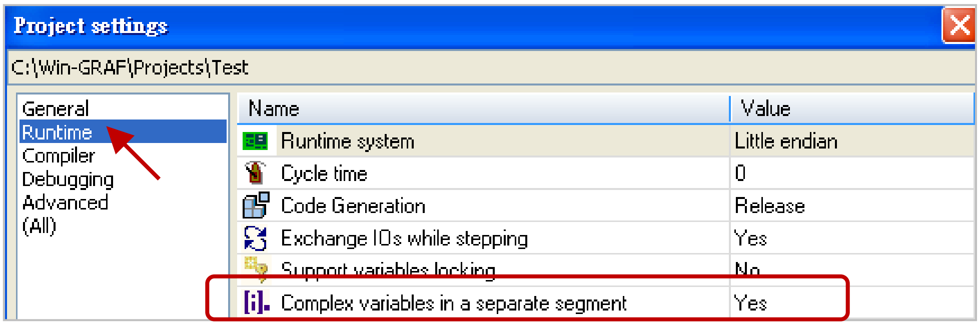
LD Program:



If want to pack 8 Booleans into one SINT, you must assign a "BYTE" variable to the output(Q) to save the value temporary, and use a "ANY_TO_SINT" Function to convert BYTE into SINT type, as follows:



Note: If the compiling fails, please click "Project" > "Settings" from the menu bar to check if the setting of "Complex variables in a separate segment" in the "Runtime" is "Yes".


Project settings

C:\Win-GRAF\Projects\Test

Name	Value
Runtime system	Little endian
Cycle time	0
Code Generation	Release
Exchange IOs while stepping	Yes
Support variables locking	No
Complex variables in a separate segment	Yes

10.6 Pack/Unpack BYTE, WORD, DWORD

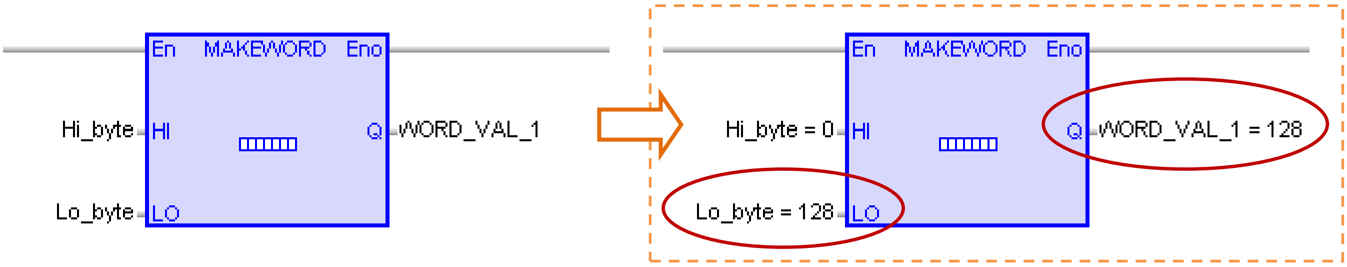
Pack Two 8-bit Data into One 16-bit Data

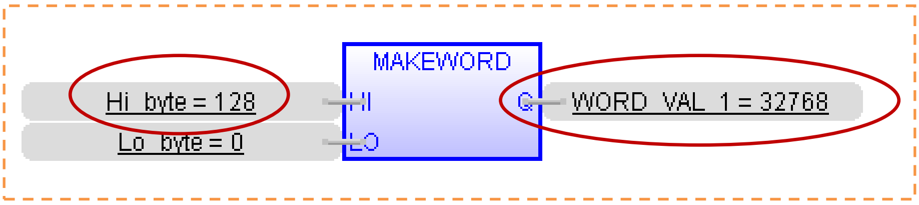
If want to pack 2 BYTE (or USINT) into one WORD (or UINT), you can use a "MAKEWORD" Function.

ST Program:

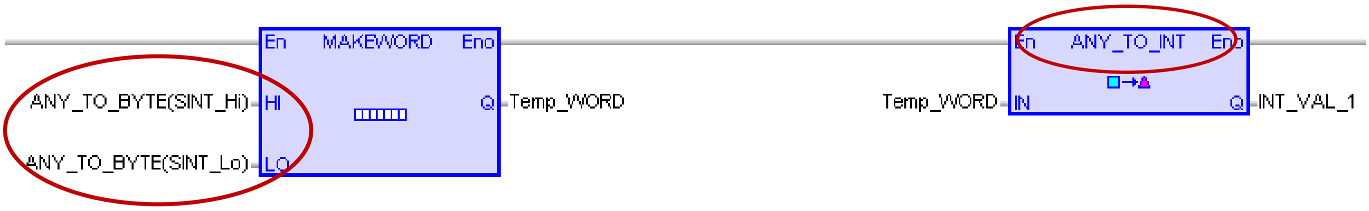
```
WORD_VAL_1  :=   MAKEWORD (Hi_byte, Lo_byte);
```

LD/FBD Program:





If want to pack 2 SINT into one INT, you must first use an ST program "ANY_TO_BYTE ()" to convert SINT into BYTE, and then use an "ANY_TO_INT" Function to convert the packed WORD into INT type.



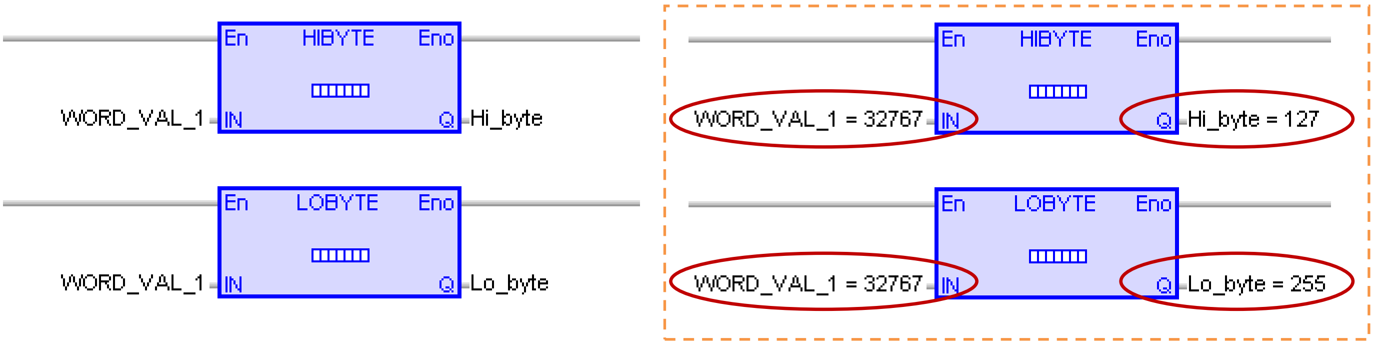
Unpack One 16-bit Data to Two 8-bit Data

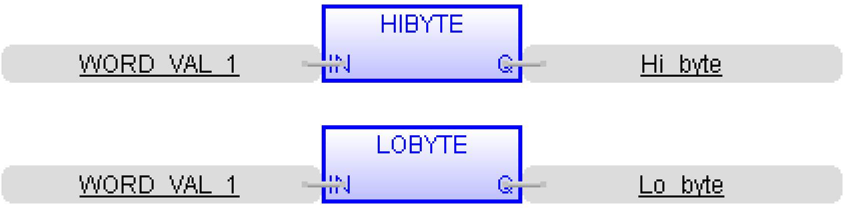
If want to unpack one WORD (or UINT) to 2 Byte (or USINT), you can use "HIBYTE", "LOBYTE" Functions.

ST Program:

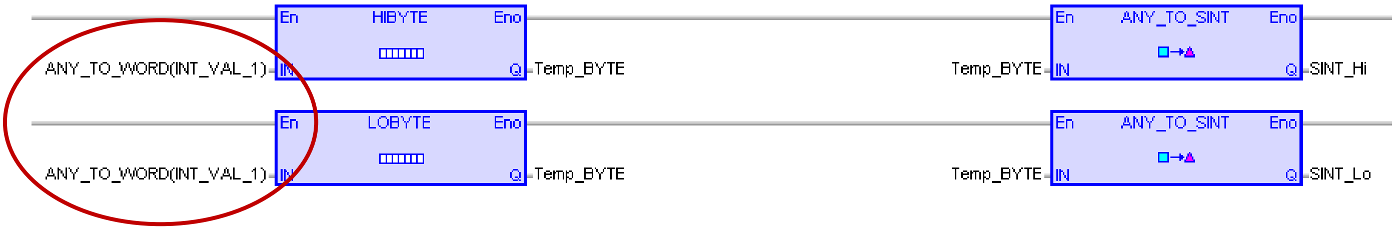
```
Hi_byte  :=   HIBYTE (WORD_VAL_1);
Lo_byte  :=   LOBYTE (WORD_VAL_1);
```

LD/FBD Program:





If want to unpack one INT to 2 SINT, you must first use an ST program "ANY_TO_WORD()" to convert INT into WORD, and then use an "ANY_TO_SINT" Function to convert the unpacked BYTE into SINT type.



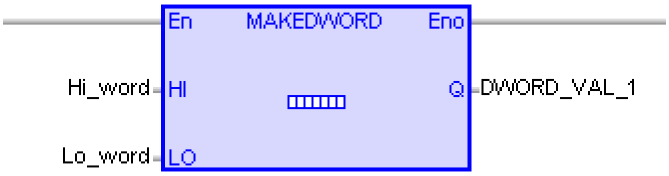
Pack Two 16-bit Data into One 32-bit Data

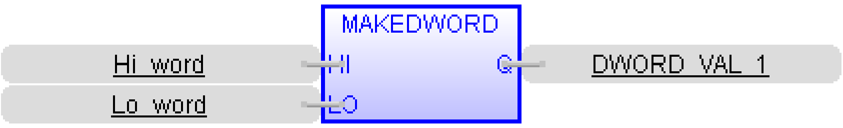
If want to pack 2 WORD (or UINT) into a DWORD (or UDINT), you can use a "MAKEDWORD" Functions.

ST Program:

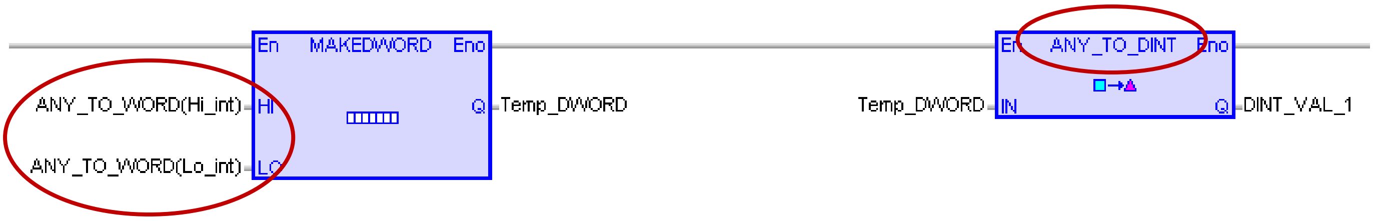
```
DWORD_VAL_1  :=   MAKEDWORD (Hi_word, Lo_word);
```

LD/FBD Program:





If want to pack 2 INT into 1 DINT, you must first use an ST program "ANY_TO_WORD()" to convert INT to WORD, and then use an "ANY_TO_DINT" Function to convert the unpacked DWORD into DINT type.



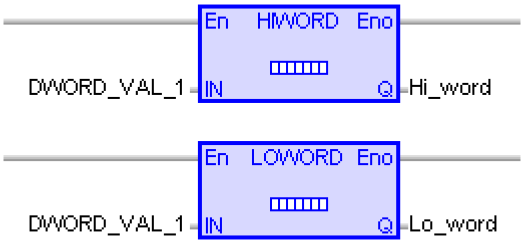
Unpack One 32-bit Data to Two 16-bit Data

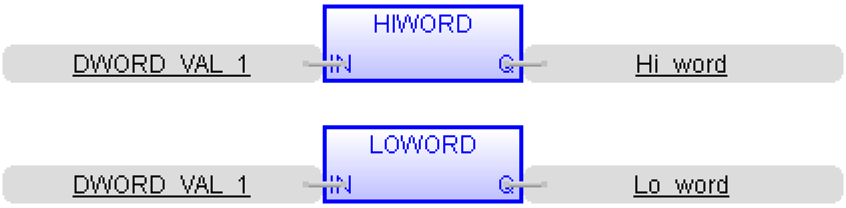
If want to unpack one DWORD (or UDINT) to 2 WORD (or UINT), you can use "HIWORD", "LOWORD" Function Blocks.

ST Program:

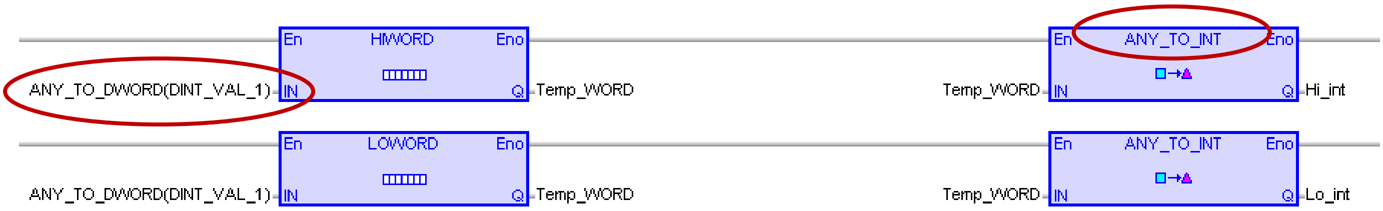
```
Hi_word  :=  HIWORD (DWORD_VAL_1);
Lo_word  :=  LOWORD (DWORD_VAL_1);
```

LD/FBD Program:





If want to unpack one DINT to 2 INT, you must first use an ST program "ANY_TO_DWORD()" to convert DINT to DWORD, and then use an "ANY_TO_INT" FB to convert the unpacked WORD into INT type.

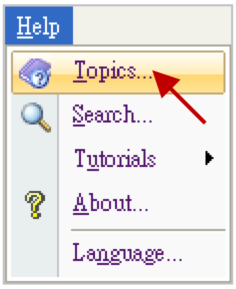


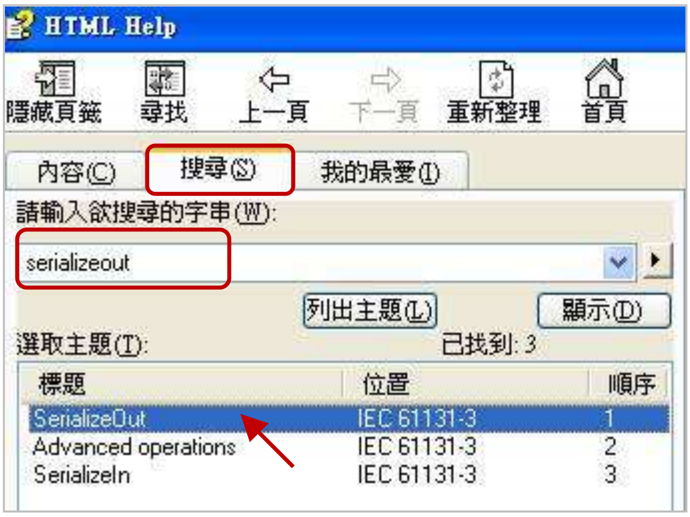
10.7 Unpack Variable to Byte Array or Pack Byte Array into Variable

"SerializeOut" Function can unpack a Win-GRAF Variable value to a Byte Array (or USINT Array);
"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 as at least "8".
2. This "Serialize" Function can not use the STRING variable.

You can open the "HTML Help" from the menu bar, and enter the searching key word to see the detail setup instructions.





If the SerializeOut() and SerializeIn() return "0", it means the saving location is wrong or the Array's space is not enough.

(* 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 the location 2 of the BYTE Array in the "Little Endian" sequence. *)

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

Note: The last parameter is "FALSE", that means to use the "Little Endian" sequence (To save the Low Byte to the starting 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 the "Big Endian" sequence. *)

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

Note: The last parameter is "TRUE", that means to use the "Big Endian" sequence (To save the High Byte to the starting address).

Example 3

(* To pack the buf[0], buf[1] , buf[2] and buf[3] in the BYTE Array into one REAL_Val in the "Little Endian" sequence. *)

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

Note: The last parameter is "FALSE", that means to use the "Little Endian" sequence (To save the Low Byte to the starting address).

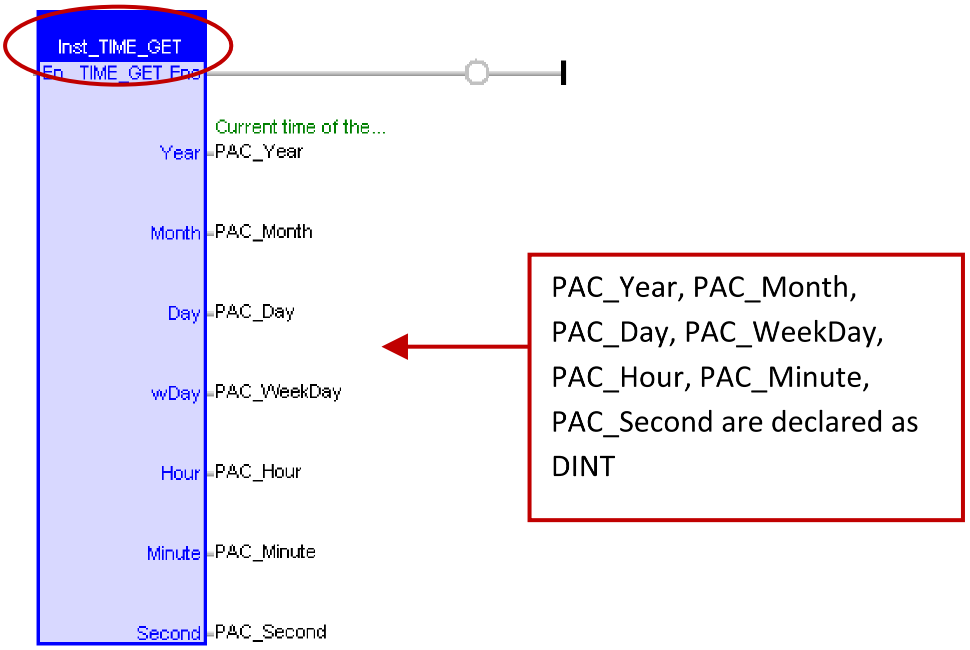
Example 4

(* To map one DINT_Val to one REAL_Val in the "Little Endian" sequence. *)

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

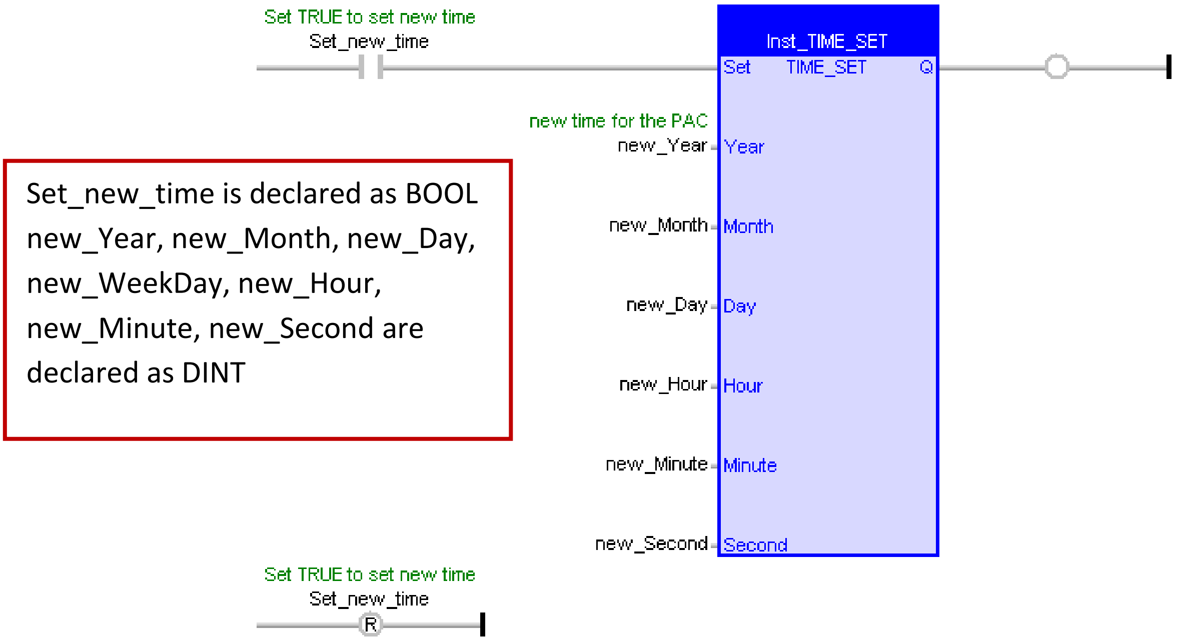
10.8 Get/Set the PAC Time

If you want to get the current time of a Win-GRAF PAC, you can use a "TIME_GET" Function Block. (Refer the Section 2.2.1)



PAC_Year, PAC_Month, PAC_Day, PAC_WeekDay, PAC_Hour, PAC_Minute, PAC_Second are declared as DINT

If want to adjust the Win-GRAF PAC time, you can use "TIME_SET" Function Block. (Refer the Section 2.3.6) 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.



Set_new_time is declared as BOOL
new_Year, new_Month, new_Day, new_WeekDay, new_Hour, new_Minute, new_Second are declared as DINT

Chapter 11 Commonly Used Tools and Useful Tips

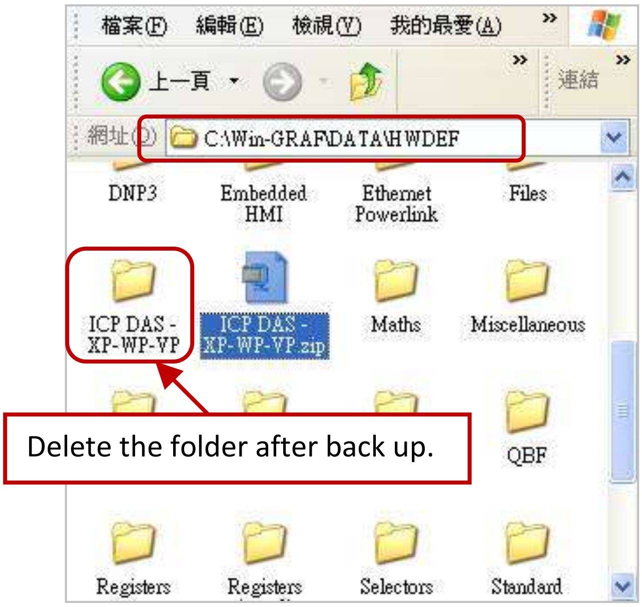
11.1 Upgrade Win-GRAF Libraries

Note: We will provide a web link for the libraries in the future.

The Win-GRAF Libraries (Including Function, Function Block and I/O Board definitions) are saved in the folder of “ICP DAS - XP-WP-VP” under the directory of “C:\Win-GRAF\DATA\HWDEF”.

In some situations, you need to upgrade the Win-GRAF Libraries to the new version for supporting more Functions or new I/O Board. Please follow the steps below:

1. First, close all Win-GRAF Workbench windows.
2. You can compress the original “ICP DAS - XP-WP-VP” folder and backup to other directory (ex: D:\temp\xxx.zip), and then delete the folder.



Delete the folder after back up.

3. Copy the new “ICP DAS - XP-WP-VP” folder into the directory “C:\Win-GRAF\DATA\HWDEF”, and execute the Win-GRAF Workbench again.

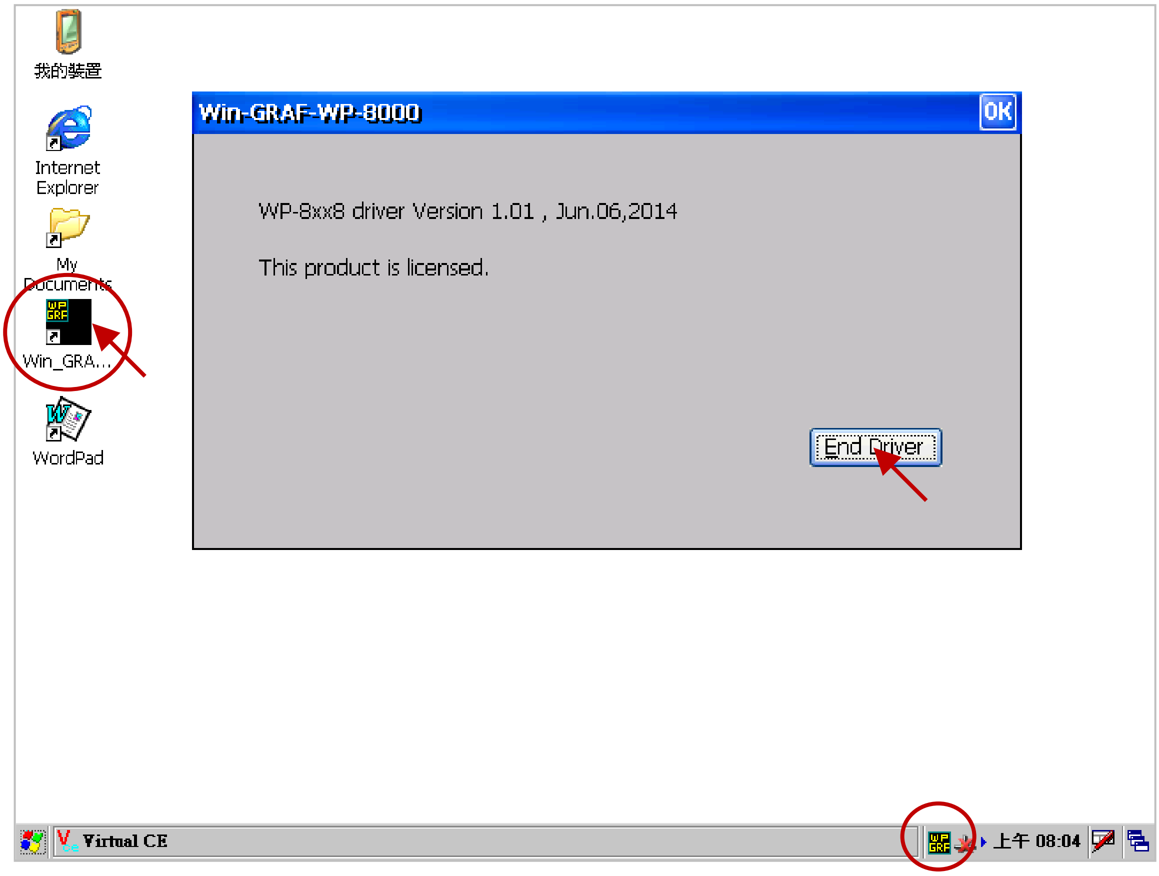
11.2 Upgrade Win-GRAF Driver

Note: We will provide a web link for the Drivers in the future.

For updating add-on functions, I/O boards or other purposes, ICP DAS will release a new version of Win-GRAF drivers in the future. Please get the latest version from the website and follow the steps below to upgrade the new driver into the PAC.

The Win-GRAF Driver of the XPAC (XP-8xx8, XP-8xx8-CE6, XP-8xx8-Atom-CE6), WinPAC (WP-8xx8, WP-5xx8) and ViewPAC (VP-25W8, VP-4138) will be placed in the directory "\System_Disk\Win-GRAF\" inside the PAC.

1. On the desktop of a PAC (use WP-8xx8 in this example), double click on the "Win-GRAF_WP_8000" icon and then click "End Driver" to stop the currently running driver.



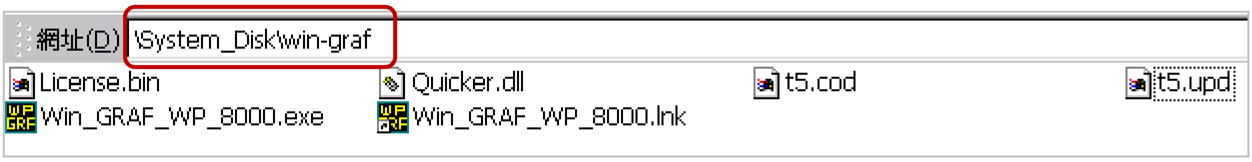
Win-GRAF-WP-8000

WP-8xx8 driver Version 1.01 , Jun.06,2014

This product is licensed.

2. In the PC, copy the new driver into the PAC's directory “\Temp\” by using FTP method.

3. In the PAC, copy the new driver from “\Temp\” into “\the System_Disk\Win-GRAF\” directory to replace the old one, and then reboot the PAC.



網址(D) \System_Disk\win-graf

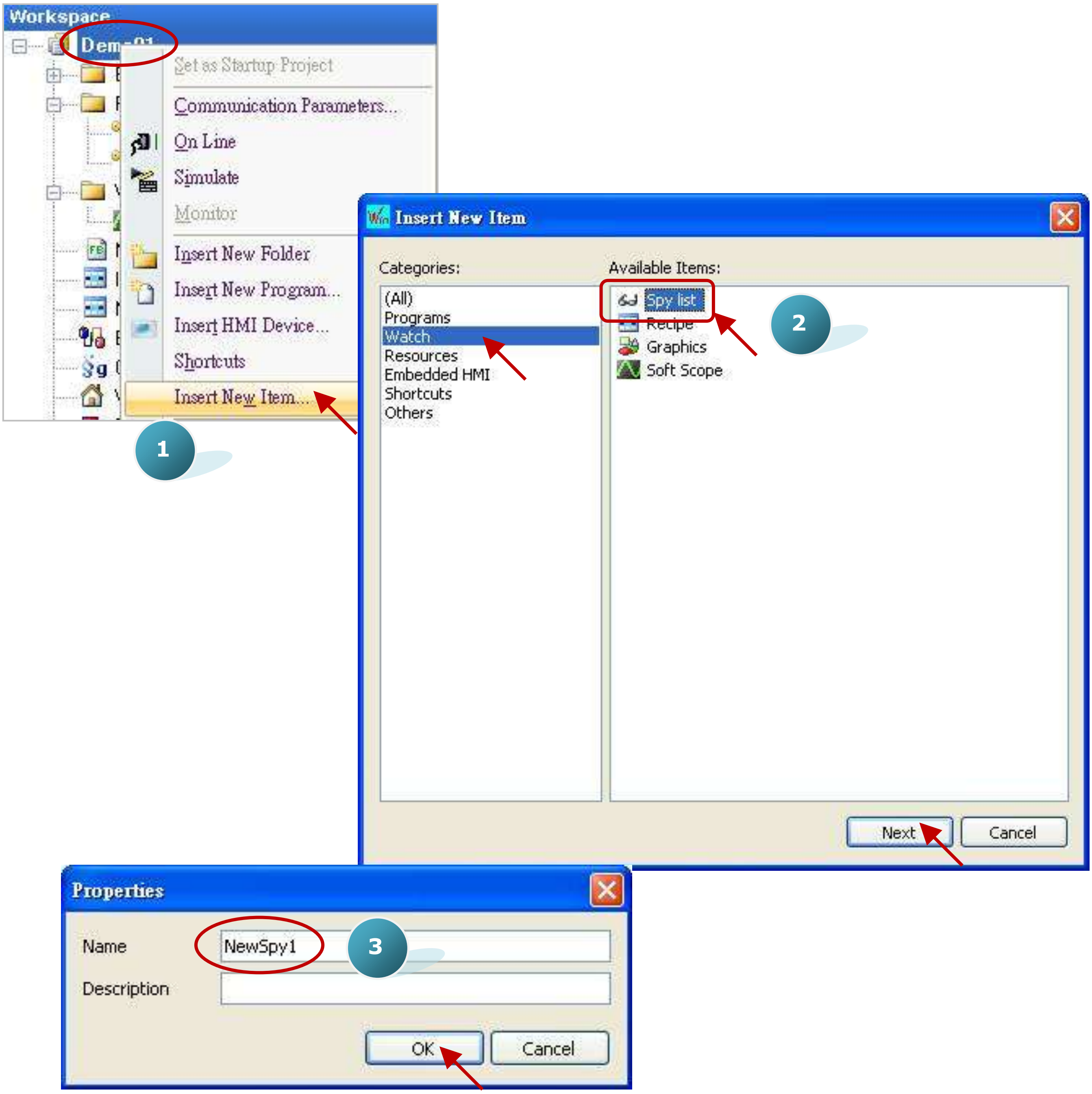
License.bin
Quicker.dll
t5.cod
t5.upd
Win_GRAF_WP_8000.exe
Win_GRAF_WP_8000.lnk

11.3 Spy List

When a program is running, the Spy List lets users quickly know the variable's value or status. Sometimes, a program may declare hundreds or thousands of variables, users do not need to look for them, just simply switch to the pre-created Spy List window to see the wanted information.

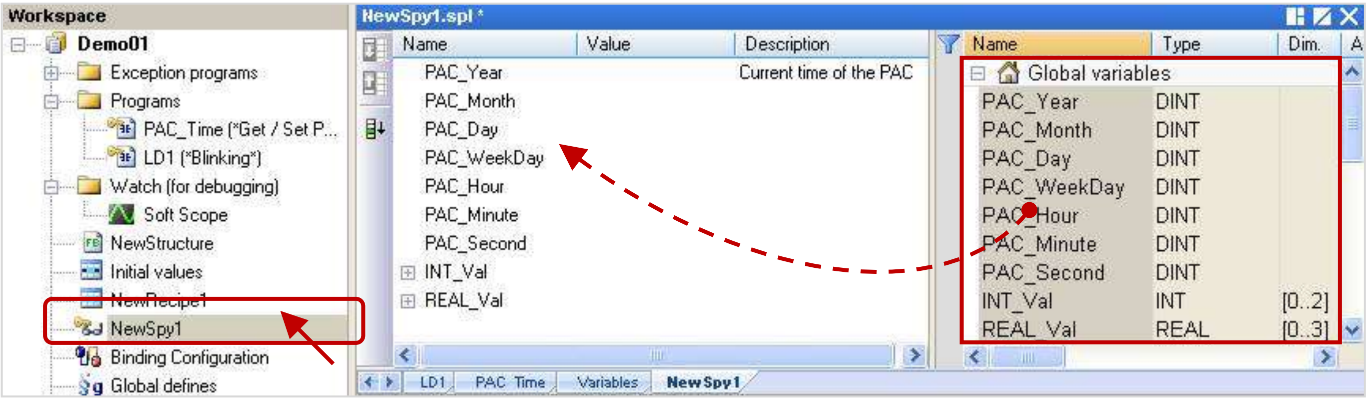
Steps:

1. Right-click on the project name (e.g., "Demo01") and select "Insert New Item".
2. Select "Spy List" of the "Watch", then click "Next" to the next step.
3. Then, key in a list name (e.g., "NewSpy1") and press "OK".



Name: NewSpy1

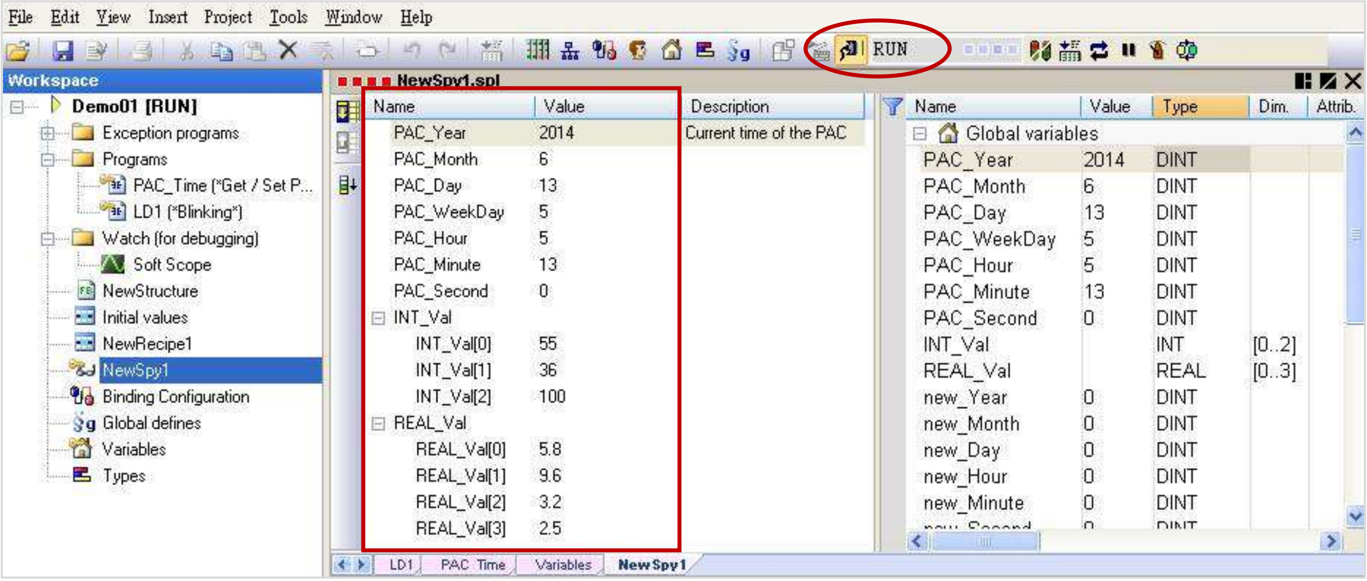
4. Double click "NewSpy1" on the left side to open the setting window and drag the variables you want to put into the window.



Name	Value	Description
PAC_Year		Current time of the PAC
PAC_Month		
PAC_Day		
PAC_WeekDay		
PAC_Hour		
PAC_Minute		
PAC_Second		
⊞ INT_Val		
⊞ REAL_Val		

Name	Type	Dim.
⊟ Global variables		
PAC_Year	DINT	
PAC_Month	DINT	
PAC_Day	DINT	
PAC_WeekDay	DINT	
PAC_Hour	DINT	
PAC_Minute	DINT	
PAC_Second	DINT	
INT_Val	INT	[0..2]
REAL_Val	REAL	[0..3]

5. When the Win-GRAF and the PAC are connected, the "NewSpy1" window will show the variables information clearly.



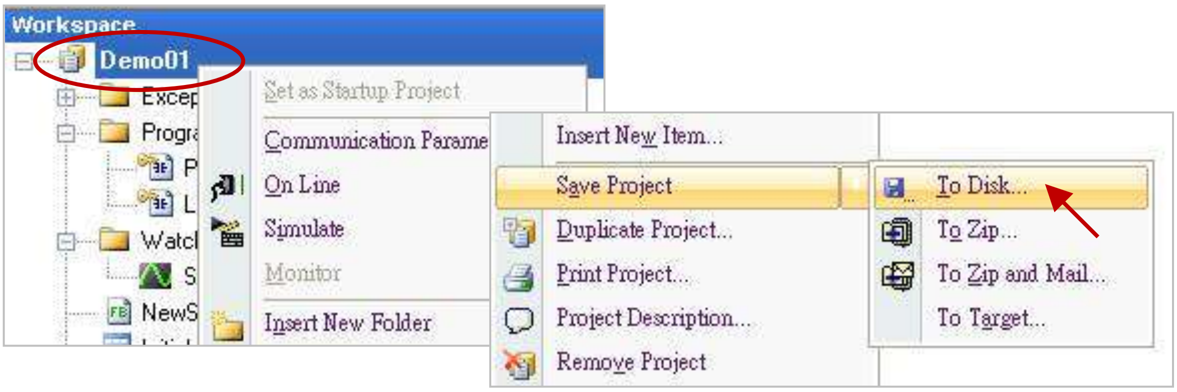
Name	Value	Description
PAC_Year	2014	Current time of the PAC
PAC_Month	6	
PAC_Day	13	
PAC_WeekDay	5	
PAC_Hour	5	
PAC_Minute	13	
PAC_Second	0	
⊟ INT_Val		
INT_Val[0]	55	
INT_Val[1]	36	
INT_Val[2]	100	
⊟ REAL_Val		
REAL_Val[0]	5.8	
REAL_Val[1]	9.6	
REAL_Val[2]	3.2	
REAL_Val[3]	2.5	

Name	Value	Type	Dim.	Attrib.
⊟ Global variables				
PAC_Year	2014	DINT		
PAC_Month	6	DINT		
PAC_Day	13	DINT		
PAC_WeekDay	5	DINT		
PAC_Hour	5	DINT		
PAC_Minute	13	DINT		
PAC_Second	0	DINT		
INT_Val		INT	[0..2]	
REAL_Val		REAL	[0..3]	
new_Year	0	DINT		
new_Month	0	DINT		
new_Day	0	DINT		
new_Hour	0	DINT		
new_Minute	0	DINT		
new_Second	0	DINT		

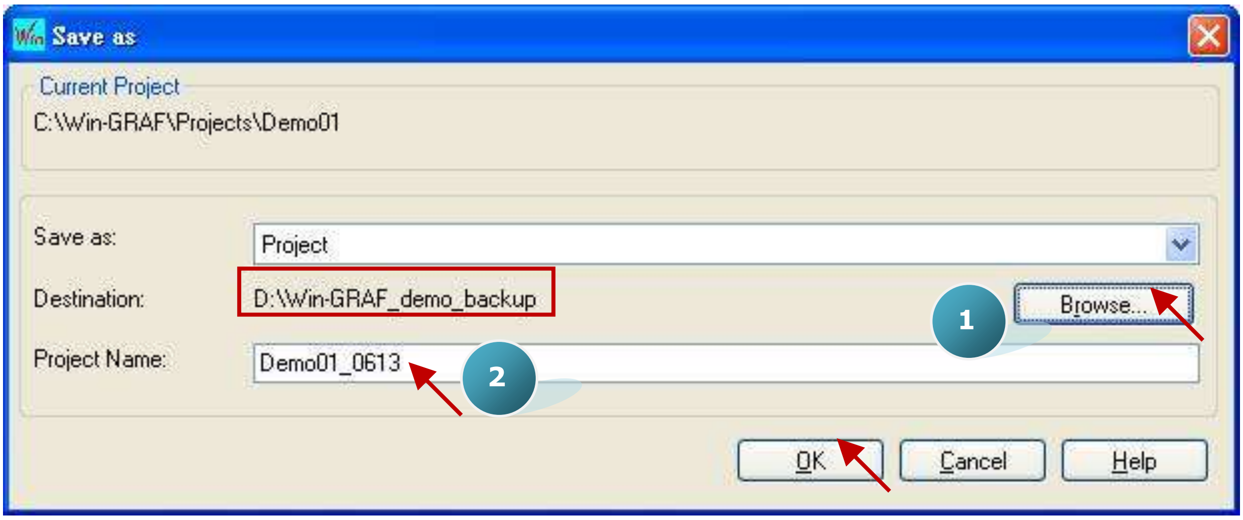
11.4 Backup/Restore a Win-GRAF Project

Backup A Win-GRAF Project:

1. Mouse right-click on the project name (e.g., "Demo01") and select "Save Project" and then "To Disk".



2. Click on the "Browser" button to assign a directory you want to save the project (e.g., D:\Win-GRAF_demo_backup), fill in the project name (e.g., "Demo01_0613"), and then click "OK" to backup the project.


Save as

Current Project
C:\Win-GRAF\Projects\Demo01

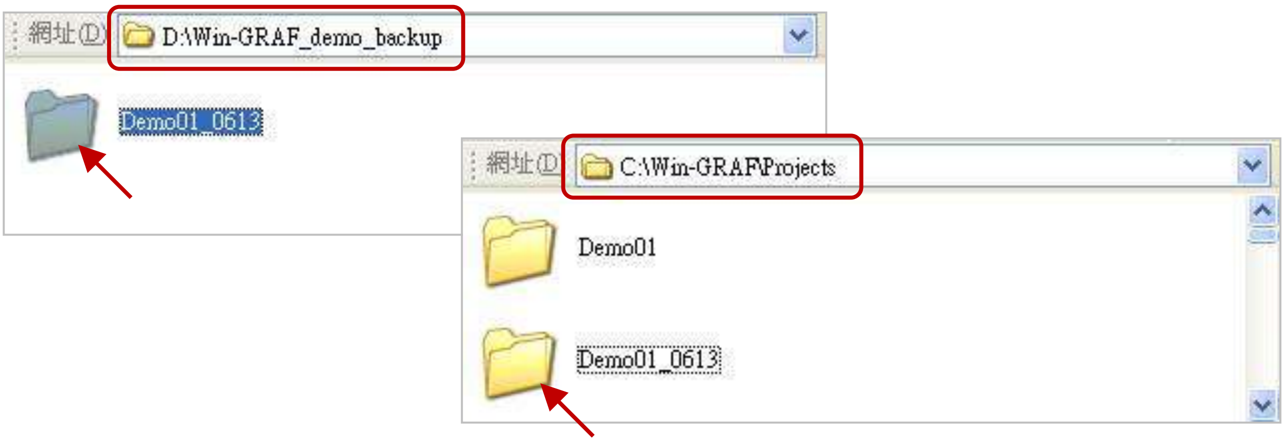
Save as: Project

Destination: D:\Win-GRAF_demo_backup

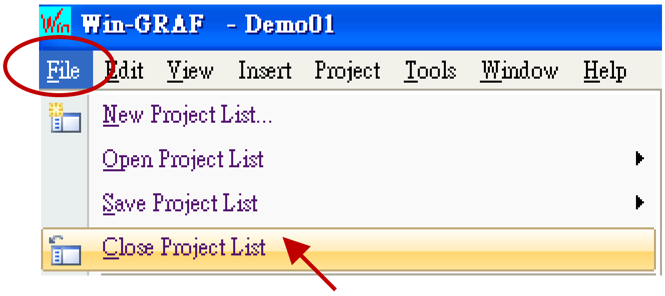
Project Name: Demo01_0613

Restore A Win-GRAF Project:

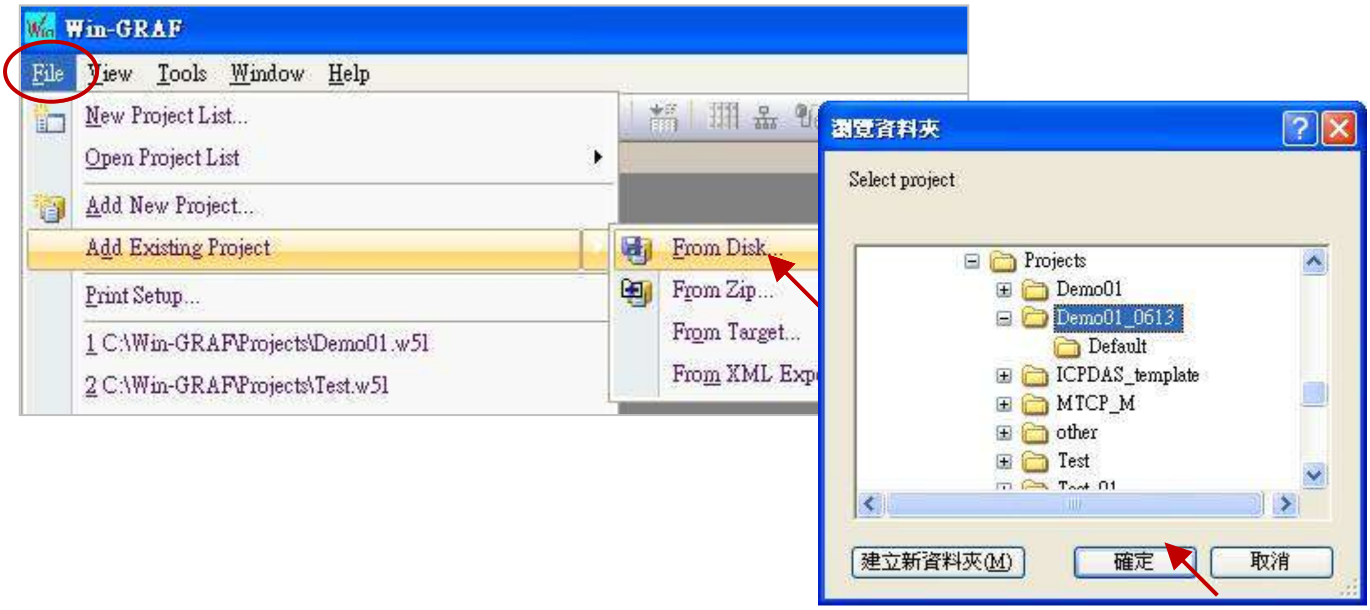
1. Copy the previously backed up the project folder (e.g., "Demo01_0613") into "C:\Win-GRAF\Projects".

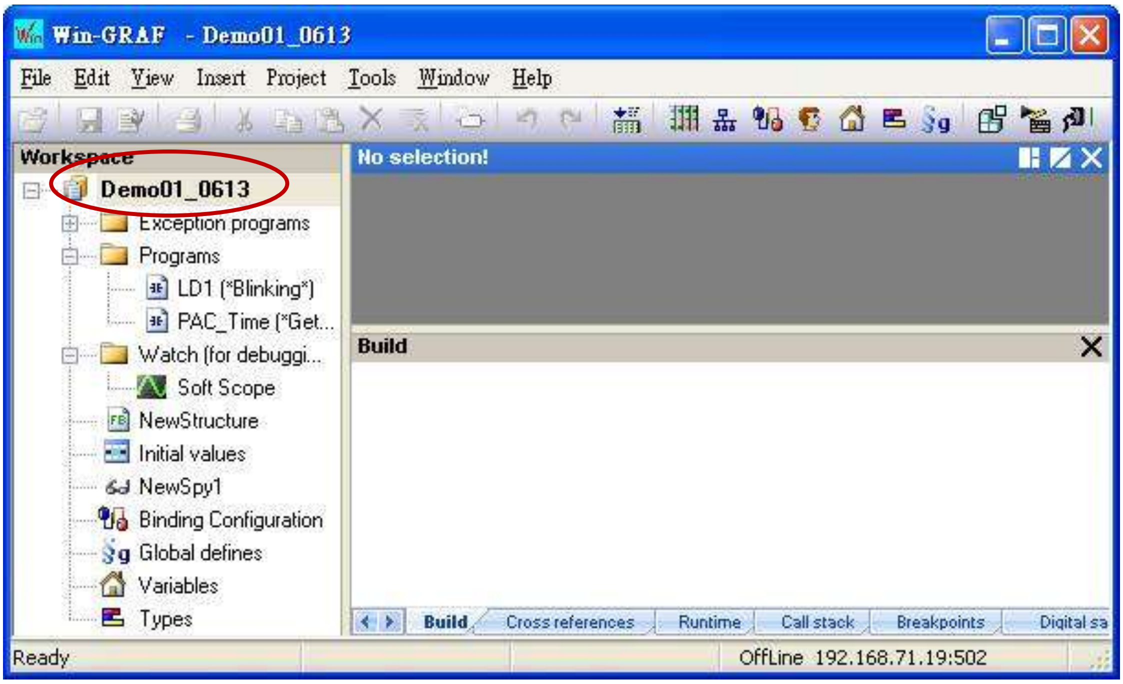


2. Click the menu bar "File" > "Close Project List" to close all opened project windows.



3. Click the menu bar "File" > "Add Existing Project" > "From Disk", select the project you want (e.g., "Demo01_0613") in the "C:\Win-GRAF\" directory, and then click "OK" to restore the project.





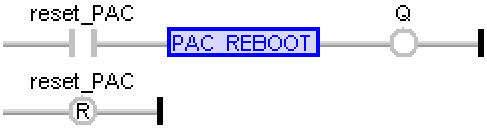
11.5 Software Reboot a PAC

Based on some cases, users may want to reboot the PAC in a software way. The Win-GRAF provides a Function "PAC_Reboot" for users to restart the PAC.

Note: Please 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 and has initial "FALSE"
    "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 ;
```



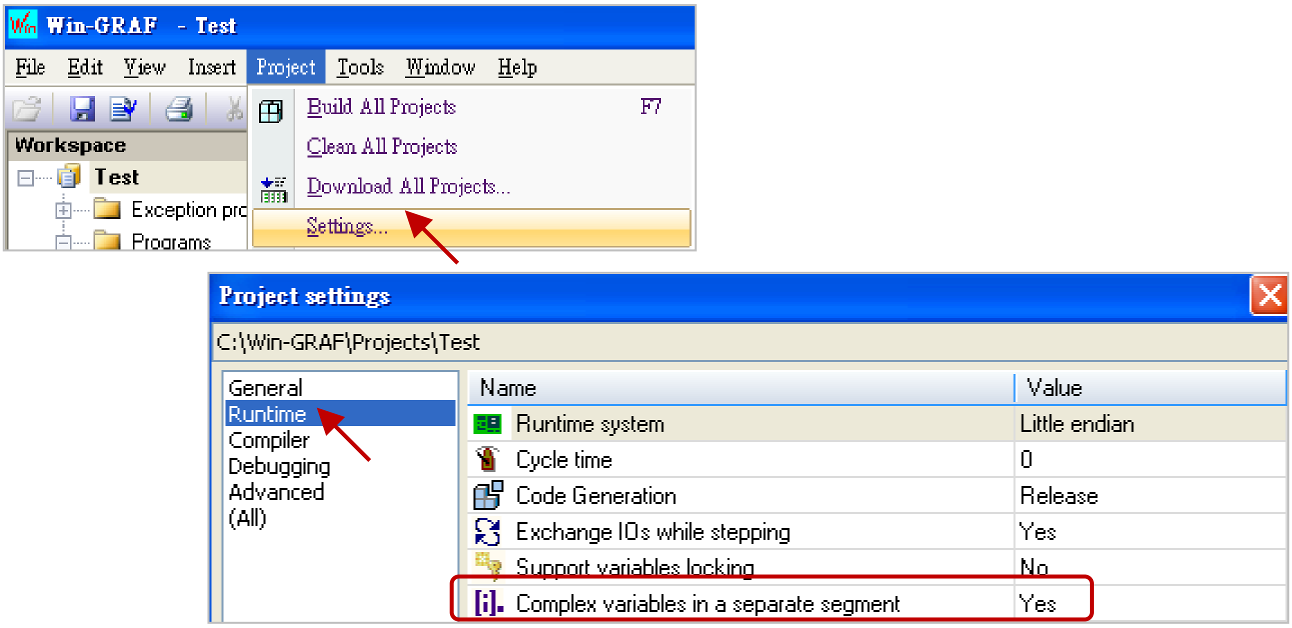
Dangerous Coding:

```
(*  "TMP_BOOL" is declared as BOOL *)
(* Dangerous ! This coding method will let the PAC reboot always and cannot stop. *)
    TMP_BOOL  :=  PAC_Reboot( ) ;
```

If a mistake to reboot the PAC always, turns the rotary switch of the Win-GRAF PAC to "1" and reset it. Then it will boot up in safe mode. Then you may rename the Win-GRAF application code in the PAC to an invalid name. Then when the rotary switch is turned back to "0" and reboot, it will boot up normally (No application). The Win-GRAF application code in the XP-8xx8, XP-8xx8-CE6, WP-8xx8, WP-5xx8, VP-25W8 and VP-4138 is "\System_Disk\Win-GRAF\t5.cod".

11.6 Using ST Syntax in LD and FBD

The Win-GRAF Workbench allows users using simple ST syntax in Ladder (LD) and Function Block Diagram (FBD) to facilitate programming. Before use, go to the menu bar "Project"> "Settings" > "Runtime", and set the "Complex variables in a separate segment" to "Yes" to enable this function.



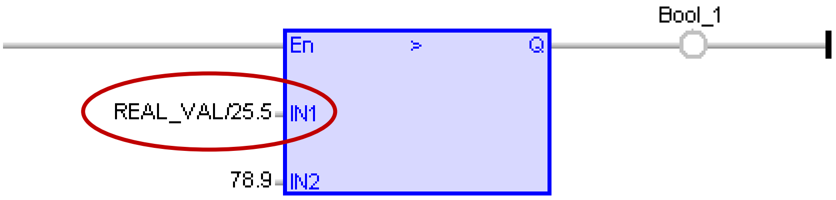
C:\Win-GRAF\Projects\Test

Name	Value
Runtime system	Little endian
Cycle time	0
Code Generation	Release
Exchange IOs while stepping	Yes
Support variables locking	No
Complex variables in a separate segment	Yes

Example:

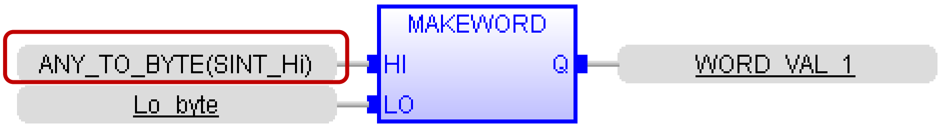
LD Program:

Using division (REAL_VAL/25.5).



FBD Program:

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

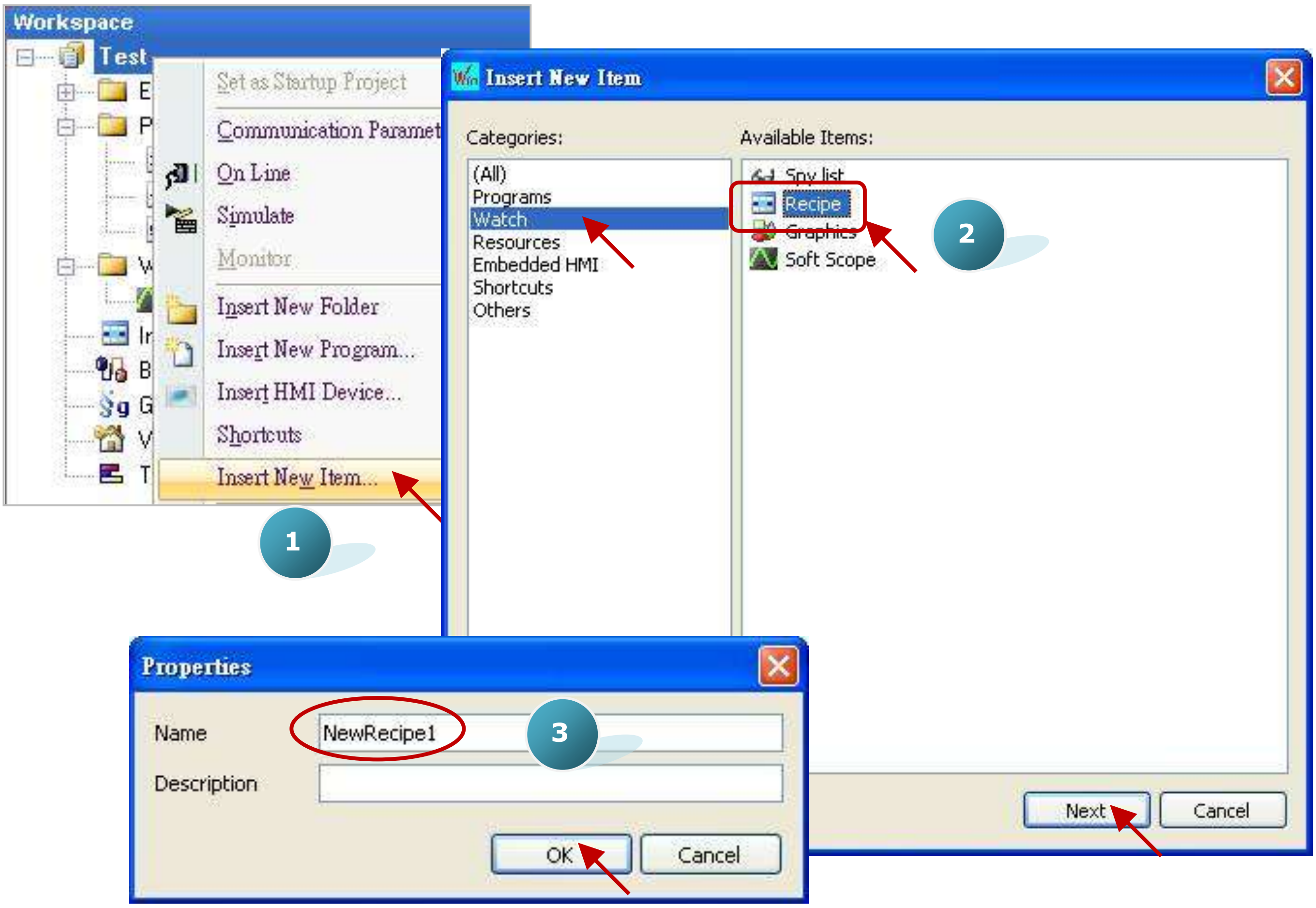


11.7 Apply a Recipe in the PAC

Some applications use the pre-defined Recipe and Value for processing different products, and this Recipe can be mapped to a combination of variables within a Win-GRAF PAC. When one day want to change the PAC process to produce a different product, you can use the Win-GRAF Workbench to connect with the PAC and select a new Recipe you want to replace, and then apply it to the PAC.

Steps:

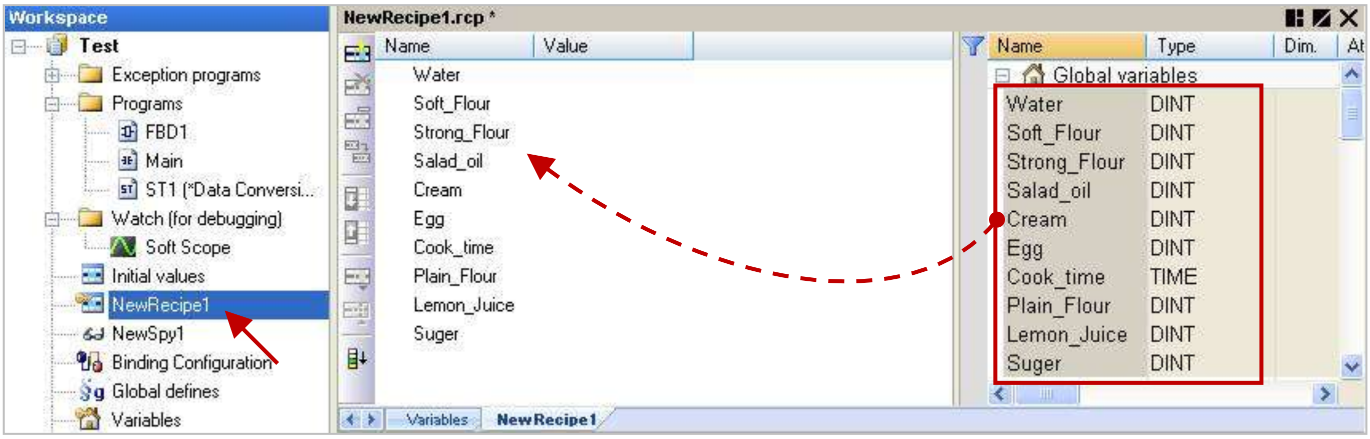
1. Mouse right-click on the project name (e.g., "Test") and select "Insert New Item"; then click on "Watch" > "Recipe" and "Next" button, fill in the Recipe name (e.g., "NewRecipe1") and then click "OK".



Name: NewRecipe1

Description:

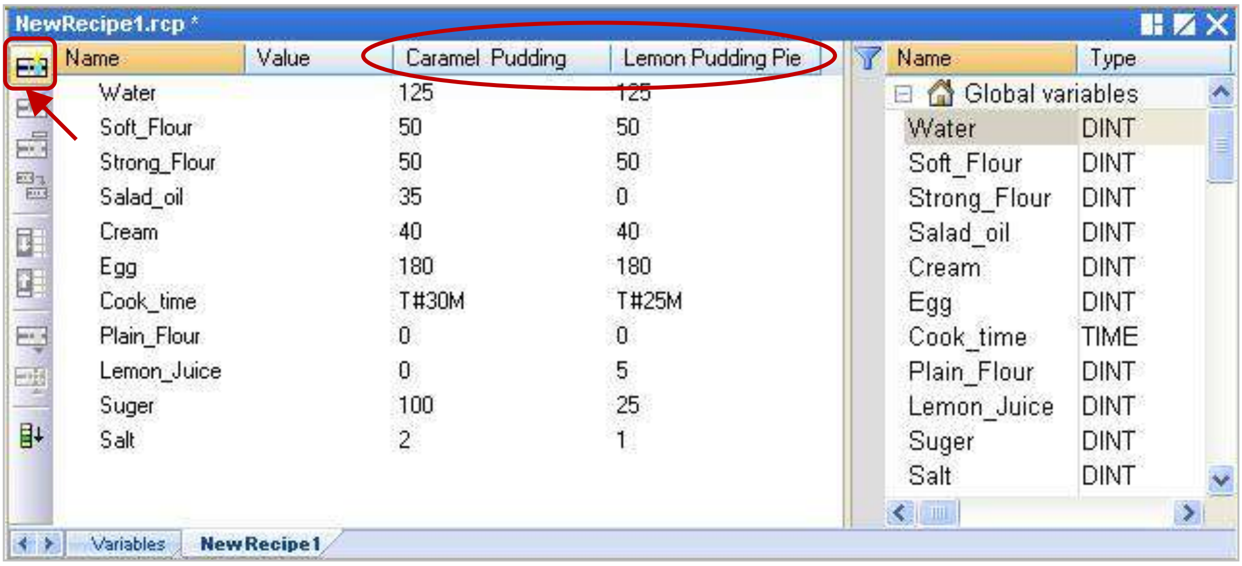
2. Double click "NewRecipe1" on the left side to open the setting window and drag the variables you want to put into the window.



Name	Value
Water	
Soft_Flour	
Strong_Flour	
Salad_oil	
Cream	
Egg	
Cook_time	
Plain_Flour	
Lemon_Juice	
Suger	

Name	Type	Dim.	At
Water	DINT		
Soft_Flour	DINT		
Strong_Flour	DINT		
Salad_oil	DINT		
Cream	DINT		
Egg	DINT		
Cook_time	TIME		
Plain_Flour	DINT		
Lemon_Juice	DINT		
Suger	DINT		

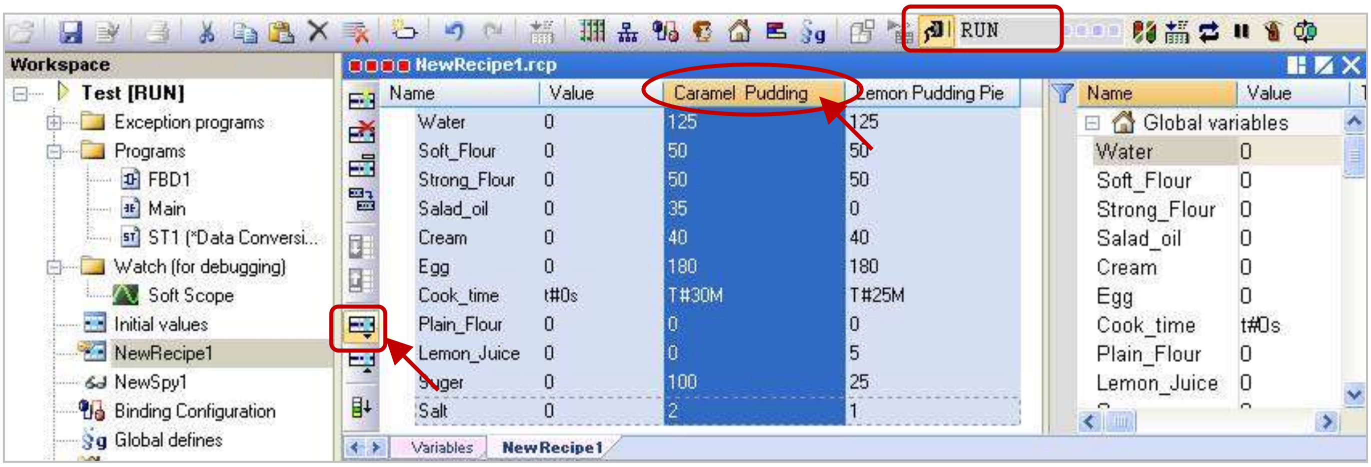
3. Click on the "Insert Column" icon to add the new Recipe, and then fill in the suitable Values.



Name	Value	Caramel Pudding	Lemon Pudding Pie
Water		125	125
Soft_Flour		50	50
Strong_Flour		50	50
Salad_oil		35	0
Cream		40	40
Egg		180	180
Cook_time		T#30M	T#25M
Plain_Flour		0	0
Lemon_Juice		0	5
Suger		100	25
Salt		2	1

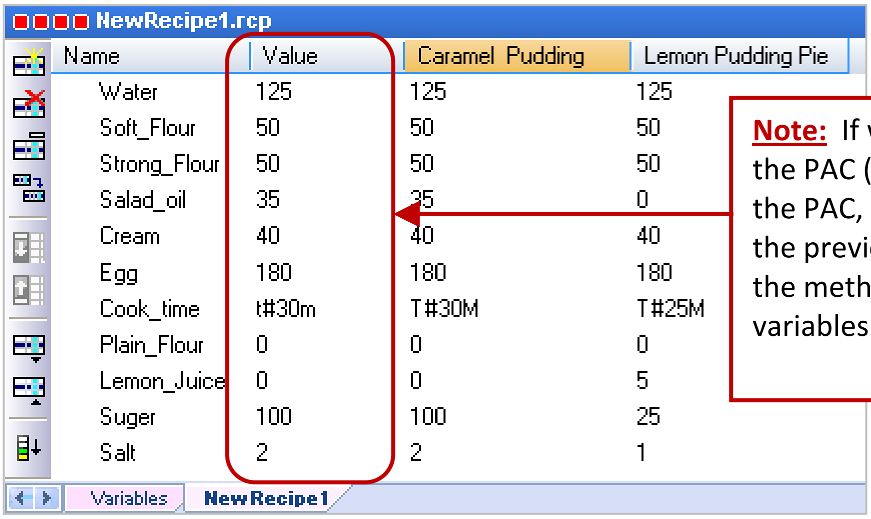
Name	Type
Global variables	
Water	DINT
Soft_Flour	DINT
Strong_Flour	DINT
Salad_oil	DINT
Cream	DINT
Egg	DINT
Cook_time	TIME
Plain_Flour	DINT
Lemon_Juice	DINT
Suger	DINT
Salt	DINT

4. Click on the "On Line" button to connect the PAC. At first the Values are all "0", please select the product column and then click the icon "Send Recipe" to apply this Recipe into the PAC.



Name	Value	Caramel Pudding	Lemon Pudding Pie
Water	0	125	125
Soft_Flour	0	50	50
Strong_Flour	0	50	50
Salad_oil	0	35	0
Cream	0	40	40
Egg	0	180	180
Cook_time	t#0s	T#30M	T#25M
Plain_Flour	0	0	0
Lemon_Juice	0	0	5
Suger	0	100	25
Salt	0	2	1

Name	Value
Global variables	
Water	0
Soft_Flour	0
Strong_Flour	0
Salad_oil	0
Cream	0
Egg	0
Cook_time	t#0s
Plain_Flour	0
Lemon_Juice	0

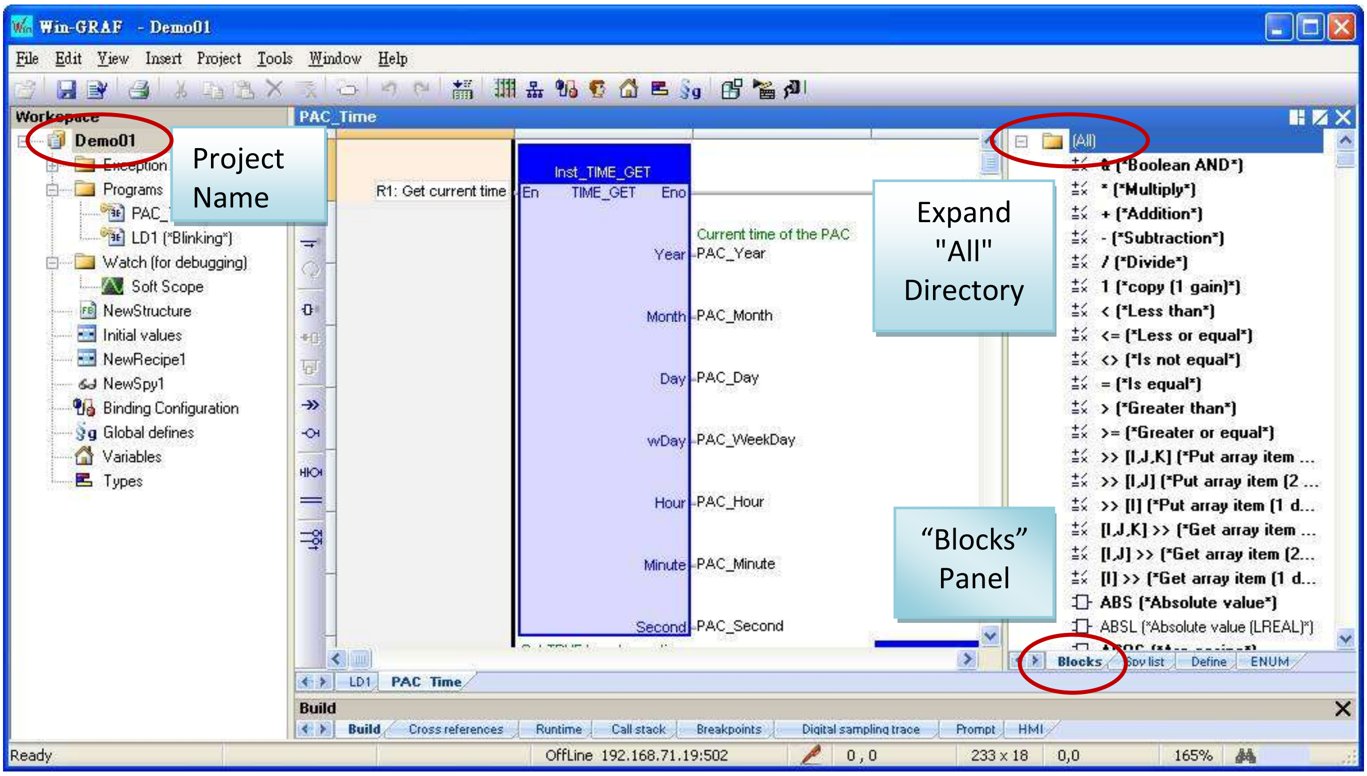


Name	Value	Caramel Pudding	Lemon Pudding Pie
Water	125	125	125
Soft_Flour	50	50	50
Strong_Flour	50	50	50
Salad_oil	35	35	0
Cream	40	40	40
Egg	180	180	180
Cook_time	t#30m	T#30M	T#25M
Plain_Flour	0	0	0
Lemon_Juice	0	0	5
Suger	100	100	25
Salt	2	2	1

Note: If want to save the Values in the PAC (e.g. Power off and restart the PAC, the Recipe can still retain the previous Values), please refer the method of using the retain variables in the Section 6.1.

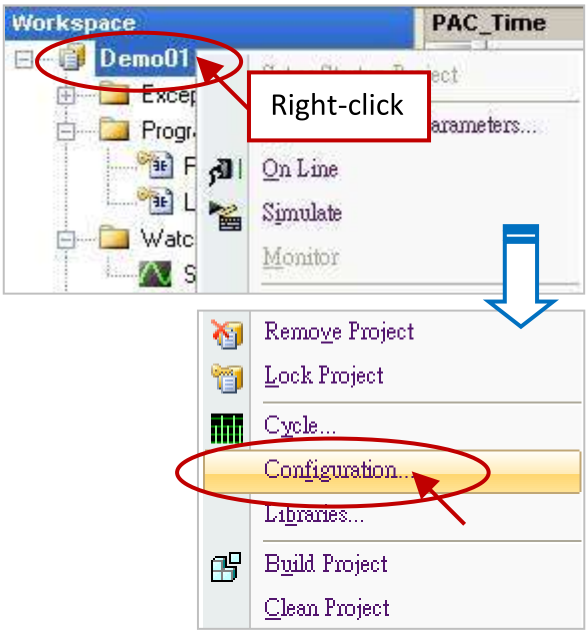
11.8 Get the Functions and Function Blocks that Supported by the PAC

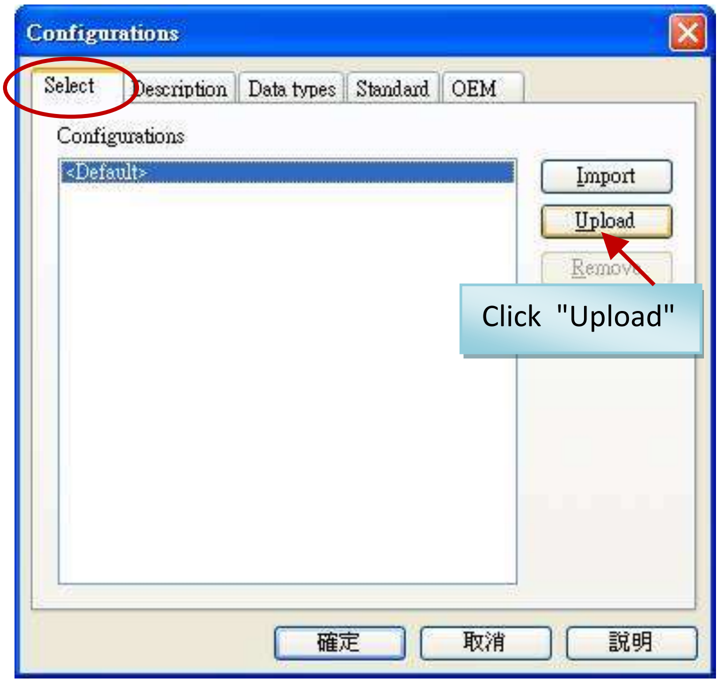
In the Win-GRAF Workbench window, user can expand the "All" directory in the "Blocks" panel to see quite a lot of Functions and Function Blocks, however, some are not supported in the Win-GRAF PAC. The following will show how to quickly get the Functions or Function Blocks that supported by the PAC.



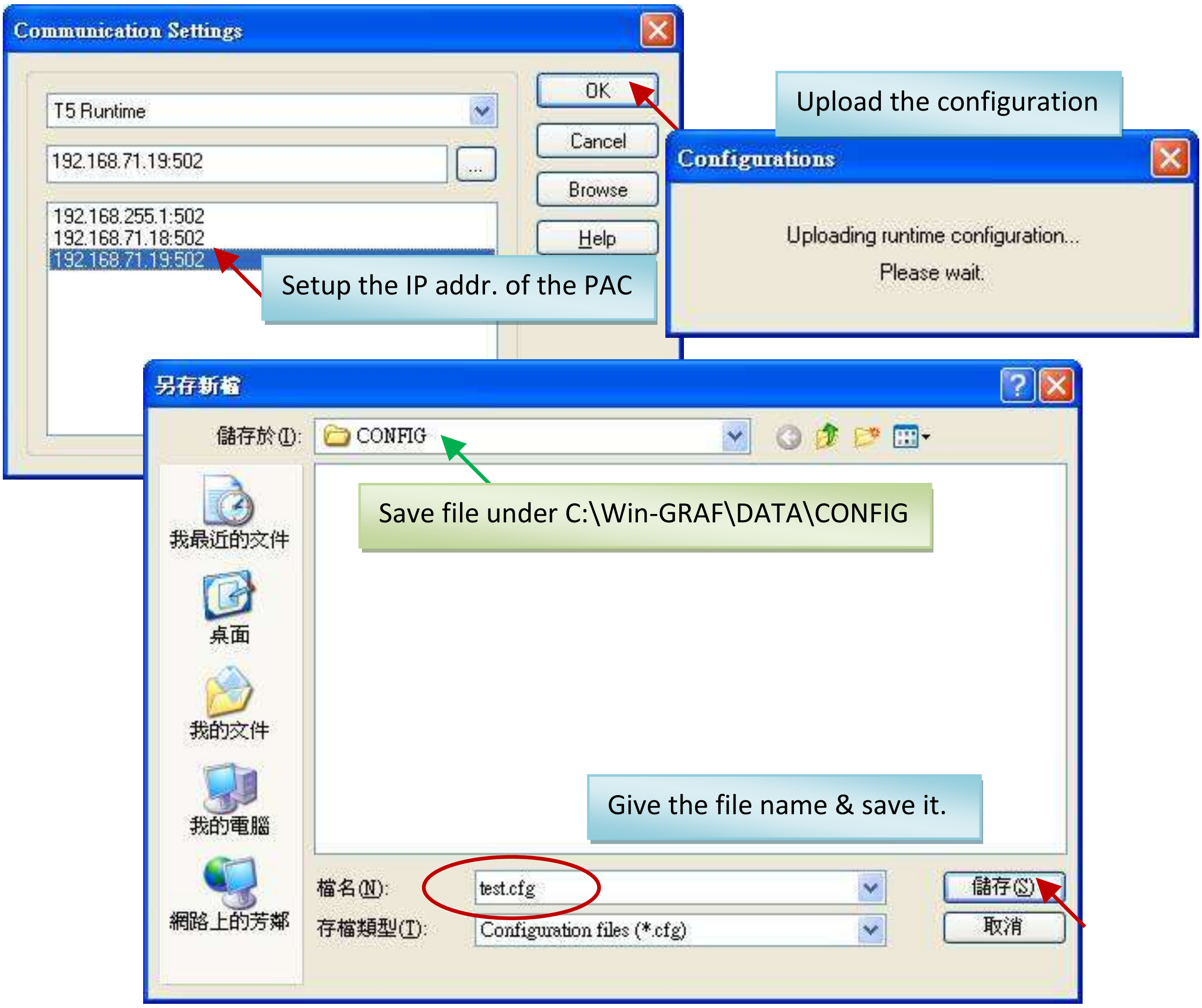
Setting Steps:

1. Make sure the PAC is powered on and connected with a PC via an Ethernet cable.
2. In the Win-GRAF Workbench, right-click on the project name (e.g., "Demo01") and then select "Configuration", and click on the "Upload" button in the "Select" tab to open the setting window.





3. Configure/Select the IP address of the PAC and click on "OK" button, the PAC will upload the configuration file. Next, key in the file name (e.g., "test.cfg") and click "Save" to save the configuration file.



Communication Settings

T5 Runtime

192.168.71.19:502

192.168.255.1:502
192.168.71.18:502
192.168.71.19:502

Setup the IP addr. of the PAC

Upload the configuration

Configurations

Uploading runtime configuration...
Please wait.

另存新檔

儲存於(I): CONFIG

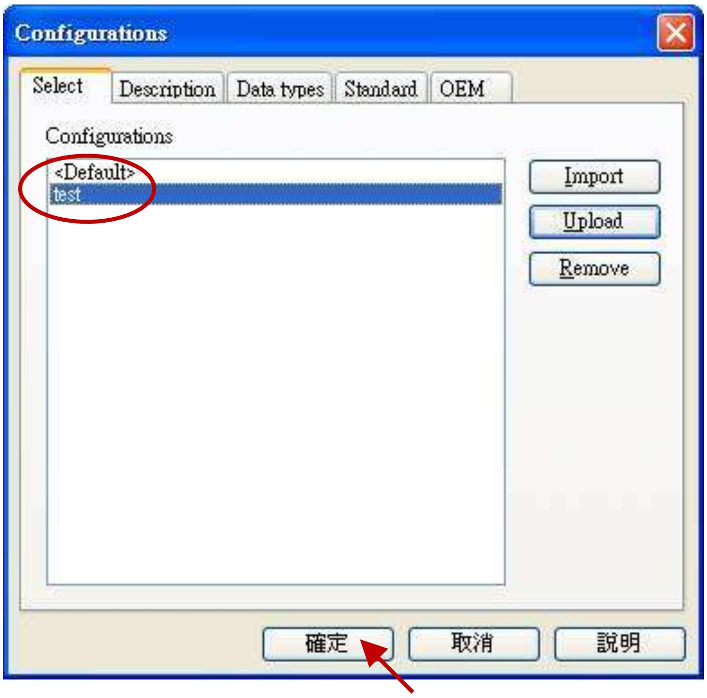
Save file under C:\Win-GRAF\DATA\CONFIG

Give the file name & save it.

檔名(N): test.cfg

存檔類型(T): Configuration files (*.cfg)

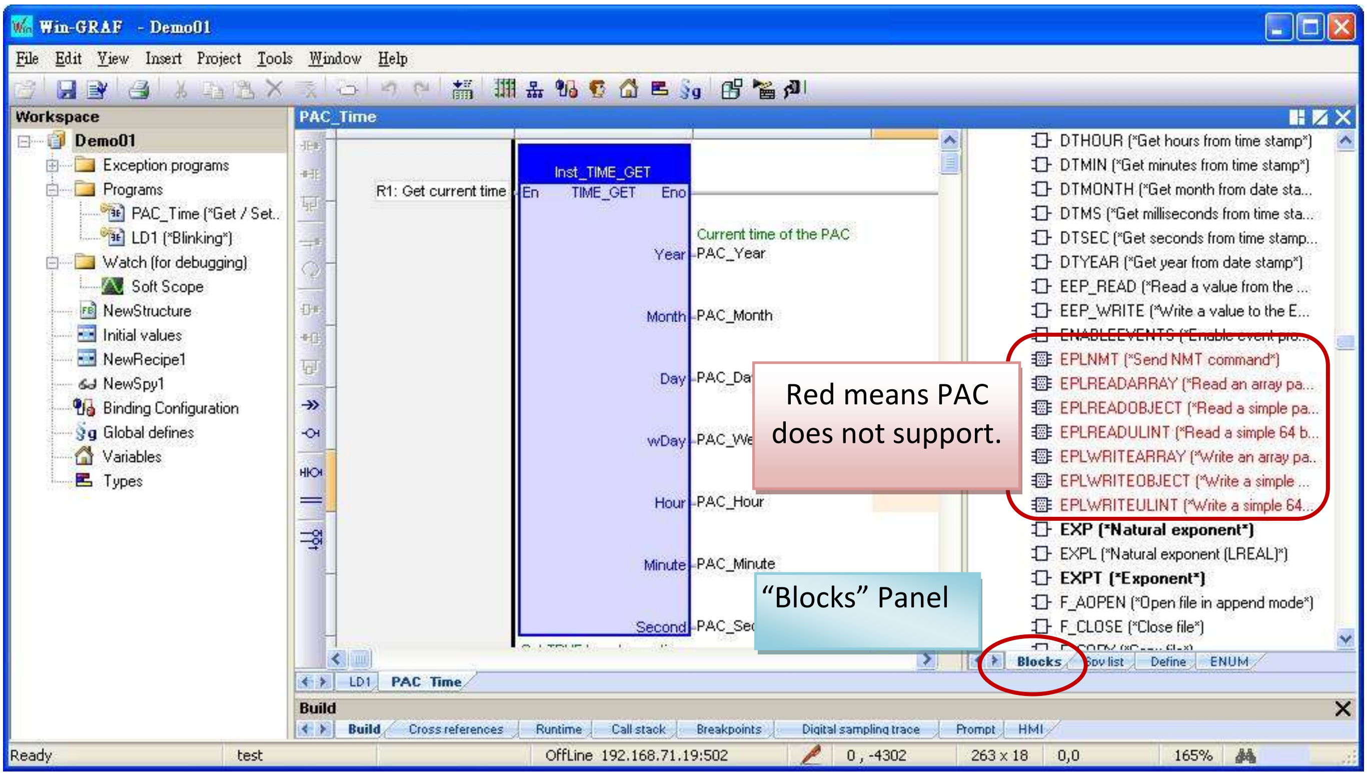
4. Back to the "Configurations" window, the configuration file (test) will show in the list, and then click "OK" to leave this window.



Configurations

Configurations
<Default>
test

5. In the "Blocks" panel, the red Functions and Function Blocks are not supported by this PAC.



Red means PAC does not support.

“Blocks” Panel

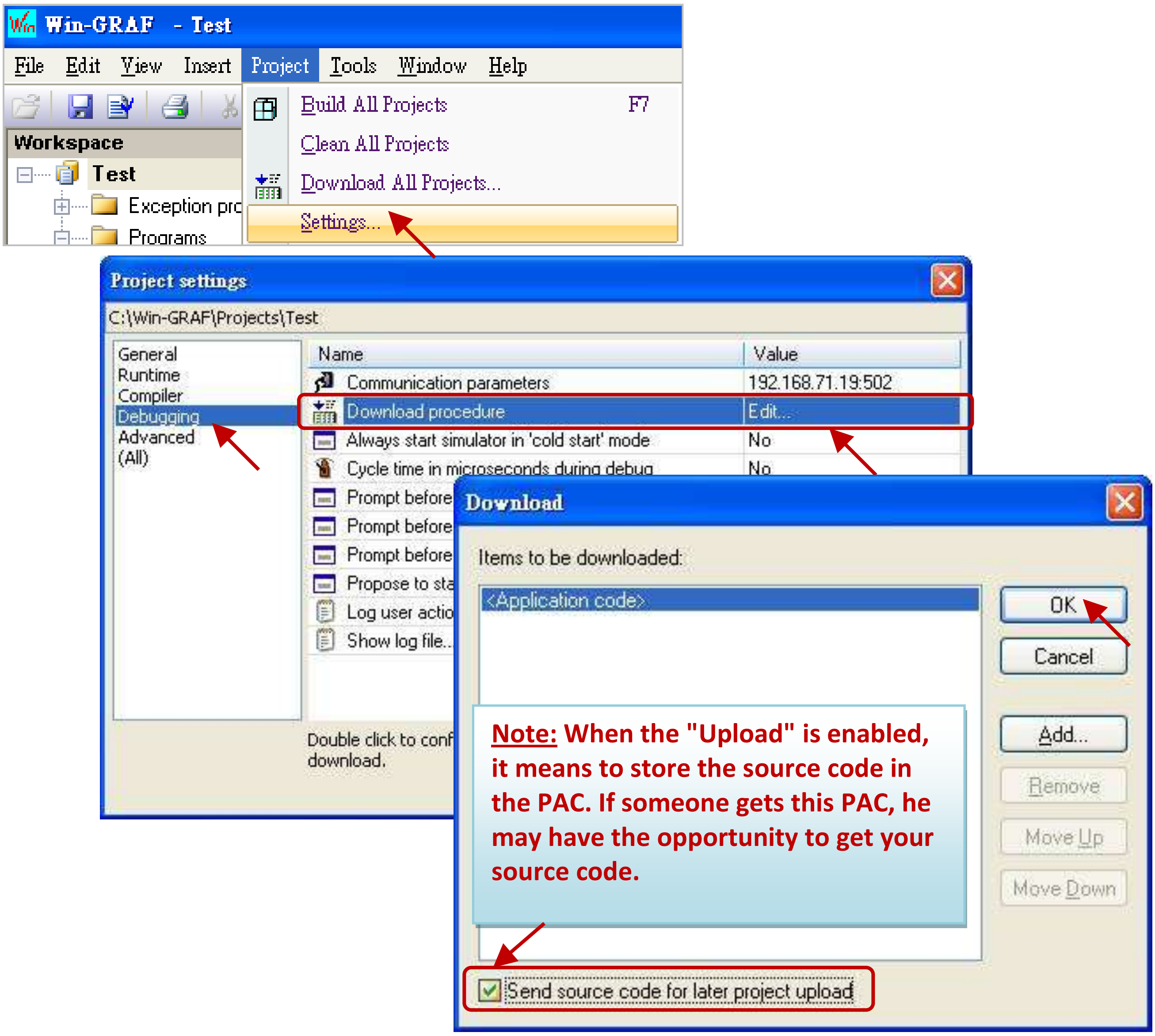
Win-GRAF - Demo01
File Edit View Insert Project Tools Window Help
Workspace
Demo01
Exception programs
Programs
PAC_Time (*Get / Set..
LD1 (*Blinking*)
Watch (for debugging)
Soft Scope
NewStructure
Initial values
NewRecipe1
NewSpy1
Binding Configuration
Global defines
Variables
Types
PAC_Time
R1: Get current time
Inst_TIME_GET
TIME_GET
En
Eno
Current time of the PAC
Year
PAC_Year
Month
PAC_Month
Day
wDay
Hour
PAC_Hour
Minute
PAC_Minute
Second
DTHOUR (*Get hours from time stamp*)
DTMIN (*Get minutes from time stamp*)
DTMONTH (*Get month from date sta...
DTMS (*Get milliseconds from time sta...
DTSEC (*Get seconds from time stamp...
DTYEAR (*Get year from date stamp*)
EEP_READ (*Read a value from the ...
EEP_WRITE (*Write a value to the E...
ENABLEEVENTS (*Enable event pro...
EPLNMT (*Send NMT command*)
EPLREADARRAY (*Read an array pa...
EPLREADOBJECT (*Read a simple pa...
EPLREADULINT (*Read a simple 64 b...
EPLWRITEARRAY (*Write an array pa...
EPLWRITEOBJECT (*Write a simple ...
EPLWRITEULINT (*Write a simple 64...
EXP (*Natural exponent*)
EXPL (*Natural exponent (LREAL)*)
EXPT (*Exponent*)
F_AOPEN (*Open file in append mode*)
F_CLOSE (*Close file*)
Blocks
Spy list
Define
ENUM
LD1
PAC_Time
Build
Build
Cross references
Runtime
Call stack
Breakpoints
Digital sampling trace
Prompt
HMI
Ready
test
OffLine 192.168.71.19:502
0 , -4302
263 x 18
0,0
165%

11.9 Upload the Win-GRAF Source Code

For some applications, users need to get the source code of the Win-GRAF project from the PAC to the PC, this is called "Upload". This function can prevent the project source code from missing or incomplete handover from the previous worker, you can still get the project source code inside the PAC.

Enable/Download The Project Source Code:

1. Click on the menu bar "Project" > "Settings" to open the setting window.
2. Double click on the "Download procedure" of the "Debugging" and select "Send source code for later project upload", and then click "OK ".



C:\Win-GRAF\Projects\Test

Name	Value
Communication parameters	192.168.71.19:502
Download procedure	Edit...
Always start simulator in 'cold start' mode	No
Cycle time in microseconds during debug	No
Prompt before	
Prompt before	
Prompt before	
Propose to sta	
Log user actio	
Show log file..	

Double click to conf download.

Note: When the "Upload" is enabled, it means to store the source code in the PAC. If someone gets this PAC, he may have the opportunity to get your source code.

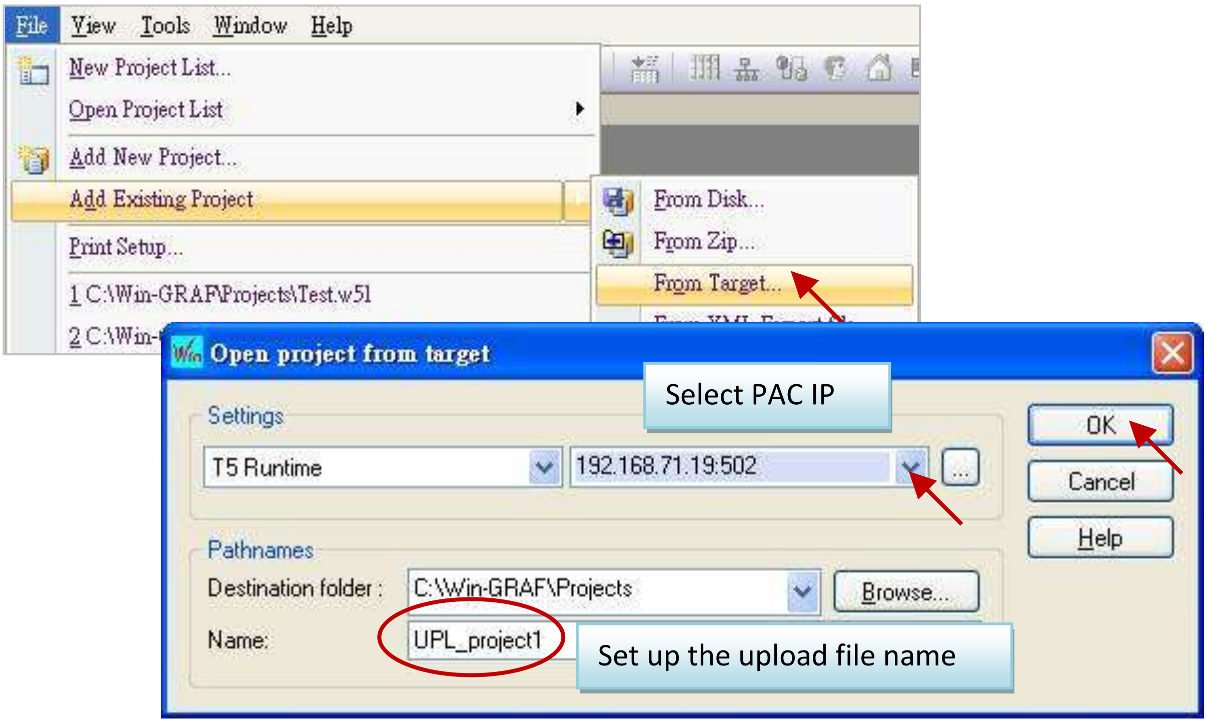
☑ Send source code for later project upload

3. Click on the menu bar "Project" > "Build All Projects" to compile the program, and then click on the tool icon to connect the PAC, and next, download the current project to the PAC. (Refer the Section 2.3.5 for the detail steps). After downloading, the source code will be stored in the file of the directory "\Systen_Disk\Win-GRAF\t5.upl" of the PAC. This file will be larger when the project increases. If the project becomes very large and complex, the file size may reach several hundreds K Bytes or even more than 1 MB.

Upload the Porject Source Code:

Please close all opened Win-GRAF windows (Click on the menu bar "File" > "Close Project List").

4. Click on the menu bar "File" > "Add Existing Project" > "From Target", then select the PAC‘s IP address and set up the upload file name (e.g., "UPL_project1"), and then click "OK" to upload the file.



Select PAC IP

Settings: T5 Runtime | 192.168.71.19:502

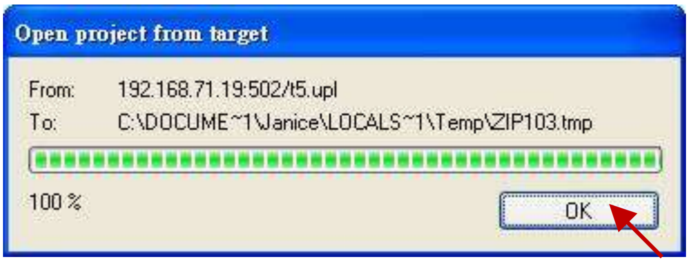
Pathnames

Destination folder : C:\Win-GRAF\Projects

Name: UPL_project1

Set up the upload file name

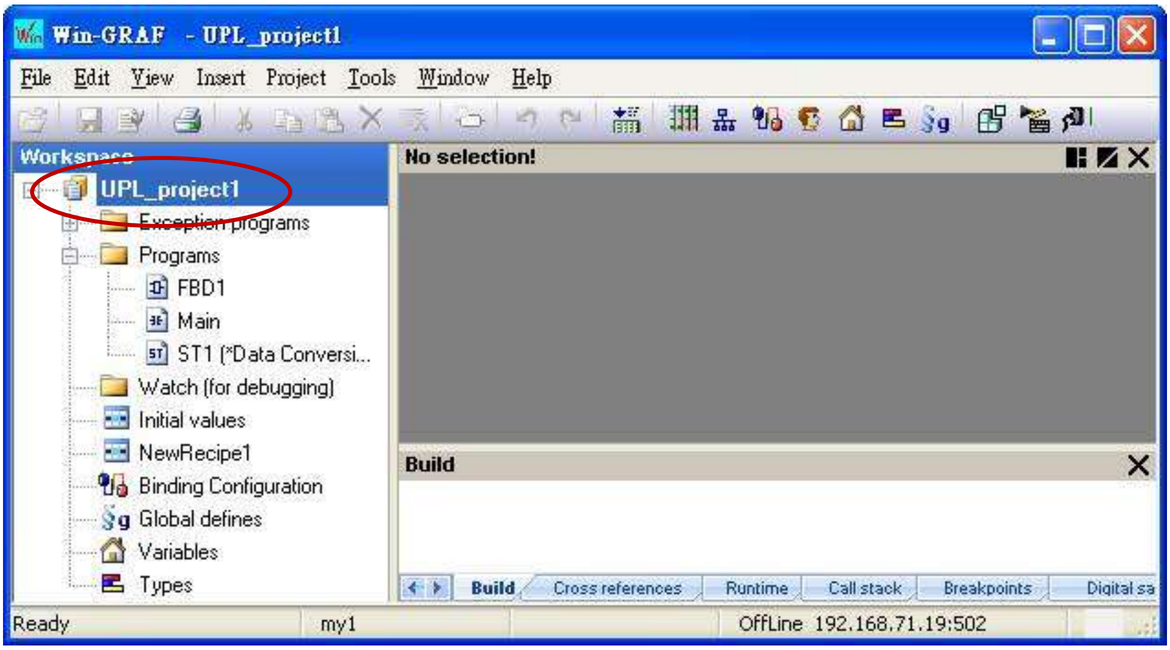
5. After uploading, click on "OK" button, and then the Win-GRAF will open the project automatically.



From: 192.168.71.19:502/t5.upl

To: C:\DOCUME~1\Janice\LOCALS~1\Temp\ZIP103.tmp

100 %

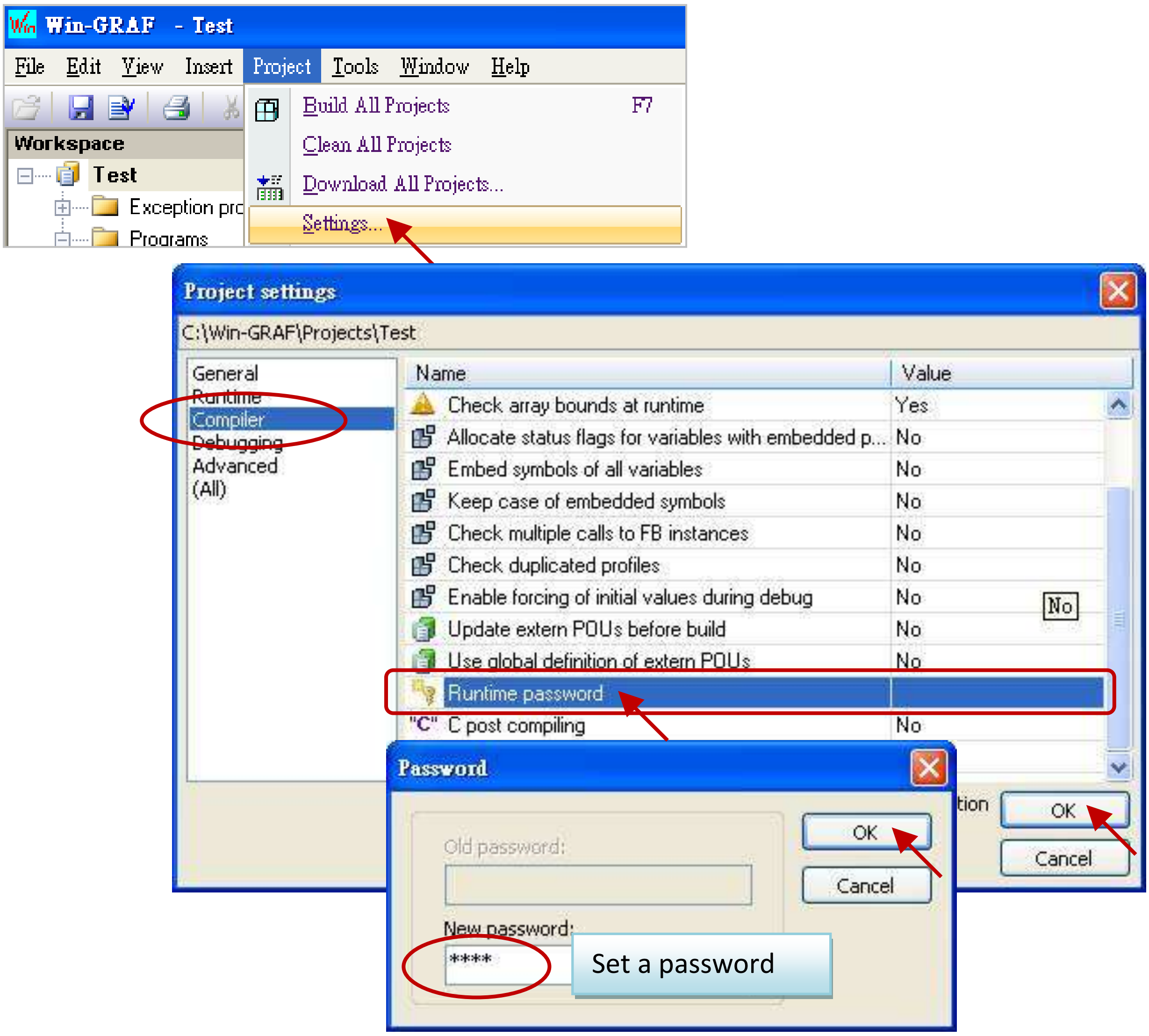


Ready | my1 | OffLine 192.168.71.19:502

11.10 Set Up the PAC Password

In order to avoid the important program running in the PAC is changed or stopped by an unfriendly connecting PC, you can set up a password for the PAC to prevent unauthorized operation.

1. Click on the menu bar "Project" > "Settings" to open the setting window.
2. Double click on the "Runtime password" of the "Compiler", set a password, and then click "OK".



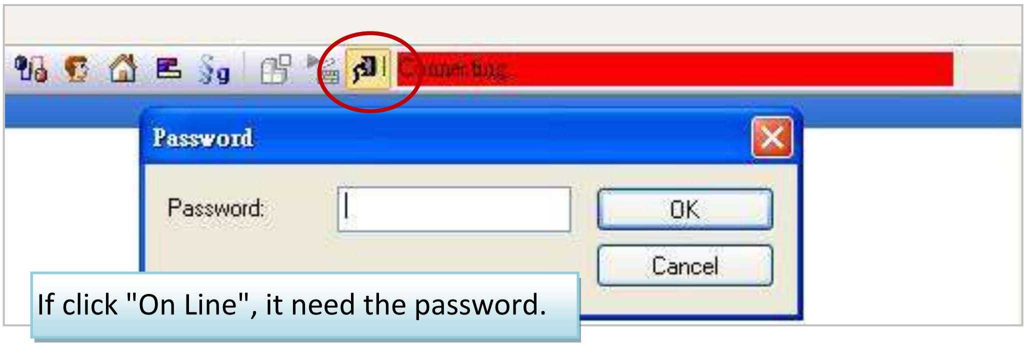
Name	Value
Check array bounds at runtime	Yes
Allocate status flags for variables with embedded p...	No
Embed symbols of all variables	No
Keep case of embedded symbols	No
Check multiple calls to FB instances	No
Check duplicated profiles	No
Enable forcing of initial values during debug	No
Update extern POUs before build	No
Use global definition of extern POUs	No
Runtime password	
C post compiling	No

Old password:

New password: ****

Set a password

3. Click on the menu bar "Project" > "Build All Projects" to compile the program again, and then download the current project to the PAC (Refer the Section 2.3.5 for detail steps.). When the next time to click the "On Line" icon for connection, it will require the password.



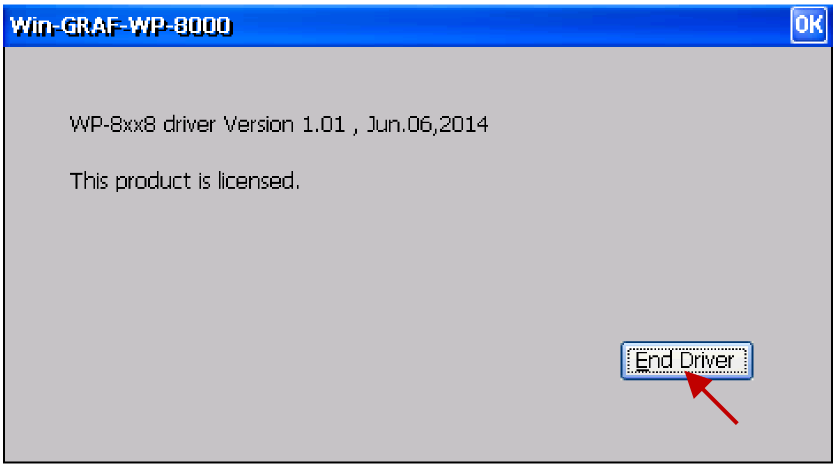
Password:

If click "On Line", it need the password.

Note: After enabling the password, please remember your password, or you will not connect the PAC.

The Only Solution:

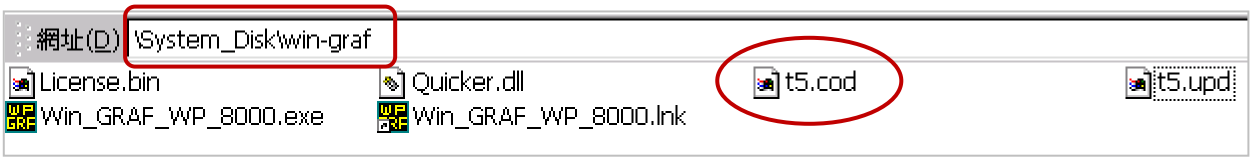
1. Connect the PAC with a USB mouse and screen.
2. In the PAC, execute the Win-GRAF Driver and then click on "End Driver" button. (Refer the Section 11.2).



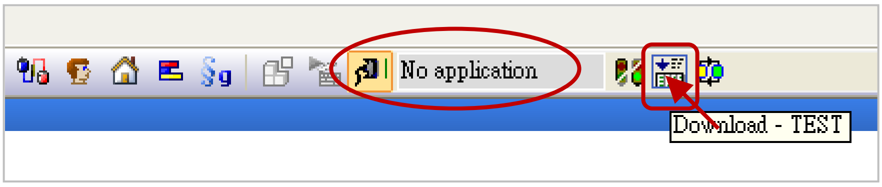
WP-8xx8 driver Version 1.01 , Jun.06,2014

This product is licensed.

3. Rename the file "t5.cod" in the directory "\System_Disk\win-graf" (e.g., "t5.cod1") or delete it. Then reboot the PAC.



Then, it will become "No application" in the PAC. Now, you can connect and download the application from the Win-GRAF Workbench again.



11.11 Using Function Block in the ST Program

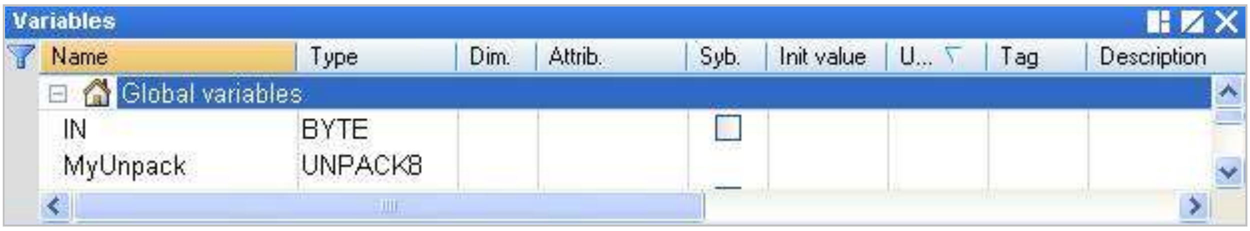
It is easy to use the Function in the ST program, just call the Function and assign the corresponding parameters. The example below will open COM3 at the beginning, and then send a String `Hello‘ from COM3 every 5 seconds.

```
(* 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(3) = FALSE  THEN
    TMP_BOO := COM_open (3, `19200,N,8,1') ;
 END_IF ;
 IF    TMR1  >=  T#5s    THEN
    TMR1  :=  T#0s ;
     COM_send_str (3, `Hello: ') ;
 END_IF;
```

To use a Function Block in the ST, you must first declare an Instance Variable of the Function Block in the variable region, after that, the using steps are similar to the steps of using the Function, as follows:

The following code can unpack one Byte to become 8 BOOLs:

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



Name	Type	Dim.	Attrib.	Syb.	Init value	U...	Tag	Description
Global variables								
IN	BYTE			□				
MyUnpack	UNPACK8							

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 How to Protect Your Win-GRAF Program to Avoid Unauthorized Copied?

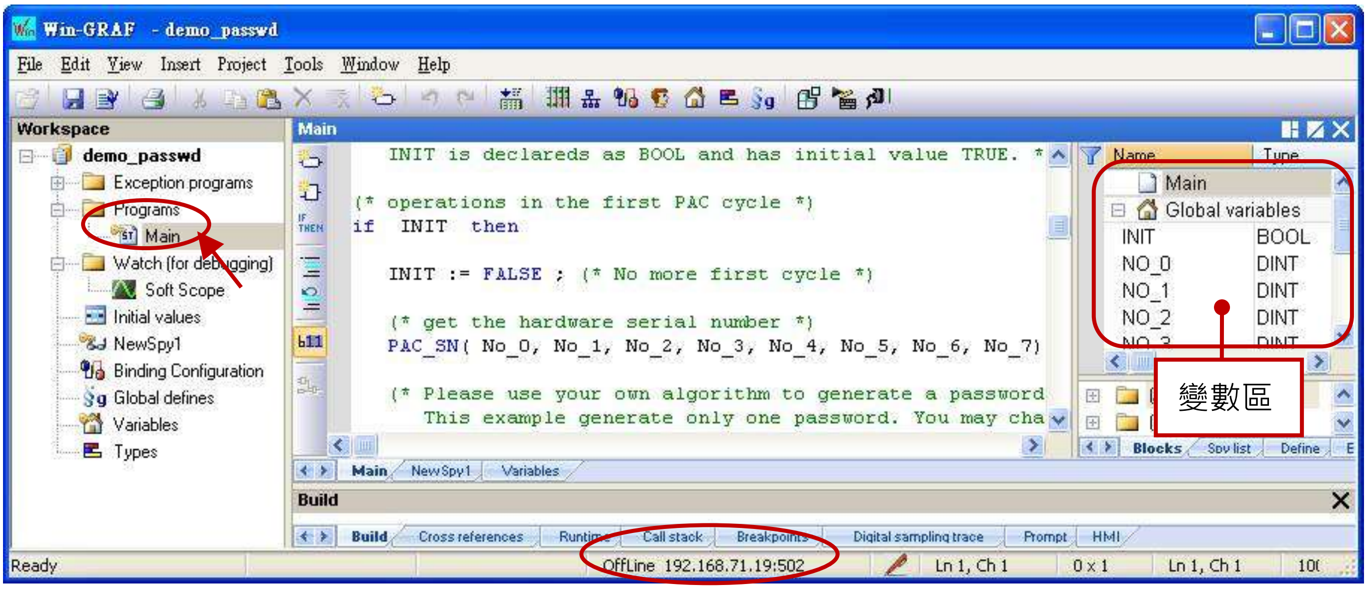
When you finish a Win-GRAF application development and prepare for delivery to the customer, please think about the possibility that your Win-GRAF application in the PAC may will be copied into another same model PAC?! Be careful! Someone else may steal your hard outcome! The following provides a simple and easy way to protect your application.

Note: If you give the Win-GRAF application Source Code to the customer, then sorry, the following method will not protect your program from stolen. Because having the Source Code, anyone can modify the code and apply into another PAC.

Each ICP DAS Win-GRAF PAC has a Serial Number that has 8 Bytes (also known as 64-Bit), and each PAC has the different and unique Serial Number. Therefore, you can use this serial number combine with your own algorithm to generate a password, and pre-store this password into the PAC's EEPROM. Then, verify this password in your application. If not passed, the application will not be allowed to execute.

The steps are as follows:
This example uses two Win-GRAF projects, one is "demo_passwd" used to generate a PAC password and save it into the EEPROM; the other is "demo_my_ap" that is a developed application and ready to ship to the customer. Before shipping a PAC to the customer, user needs to download the project "demo_passwd" into the PAC and runs it once to generate a unique password for that PAC. Then, downloads the project "demo_my_ap" into the same PAC, and then can ship the PAC to the customer. After that, if someone copies the Win-GRAF application in this PAC to another same type of PAC and the operation will fail because of the password validation failure.

There are two example projects (demo_passwd.zip and demo_my_ap.zip) in the shipment CD (\Napdos\Win-GRAF\demo-project), please refer the Chapter 12 to restore the projects (Execute File > Add Existing Project > From Zip) and set up the IP address of the current PAC.



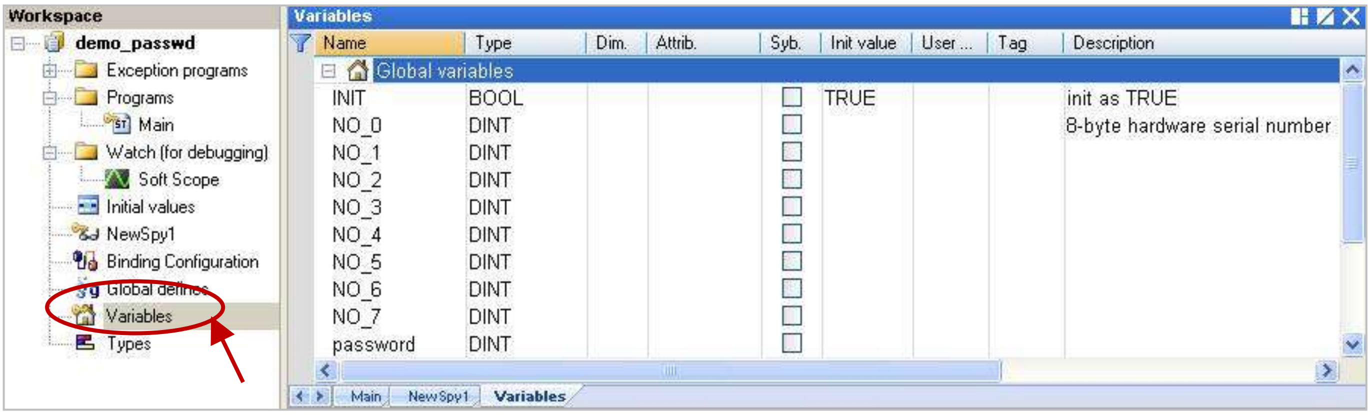
```
   INIT is declareds as BOOL and has initial value TRUE. *
(* operations in the first PAC cycle *)
if  INIT  then

   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 cha
```

Name	Type
Main	
Global variables	
INIT	BOOL
NO_0	DINT
NO_1	DINT
NO_2	DINT

"demo_passwd" Project:

This program first uses "PAC_SN" Function to read out the Serial Number, and then uses a user-defined algorithm to generate a password. Finally, save the password to an address within the EEPROM memory (user can decide where you want to store).

Variable Declaration:



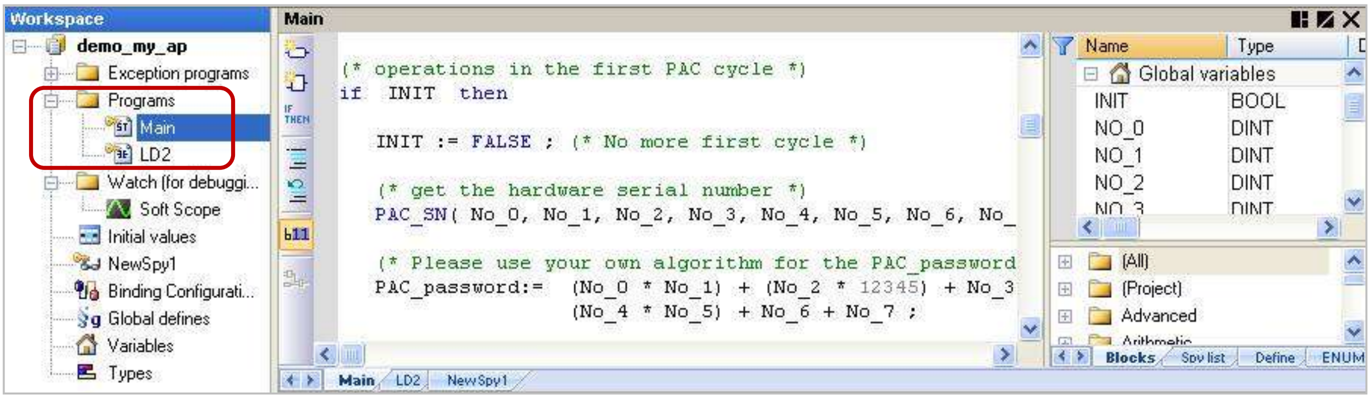
Name	Type	Dim.	Attrib.	Syb.	Init value	User ...	Tag	Description
Global variables								
INIT	BOOL			□	TRUE			init as TRUE
NO_0	DINT			□				8-byte hardware serial number
NO_1	DINT			□				
NO_2	DINT			□				
NO_3	DINT			□				
NO_4	DINT			□				
NO_5	DINT			□				
NO_6	DINT			□				
NO_7	DINT			□				
password	DINT			□				

ST Program:

```
(*  This "demo_passwd" example will generate a password by the 8-Byte Serial Number of the PAC
    and save it into the EEPROM of 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 into the EEPROM, address is "157" *)
   EEP_Write( 157 , password ) ;

end_if ;
```

"demo_my_ap" Project:

This project first uses "PAC_SN" Function to read out the Serial Number, then calculates the password, and then compares the password that read from EEPROM to check if the password correct.

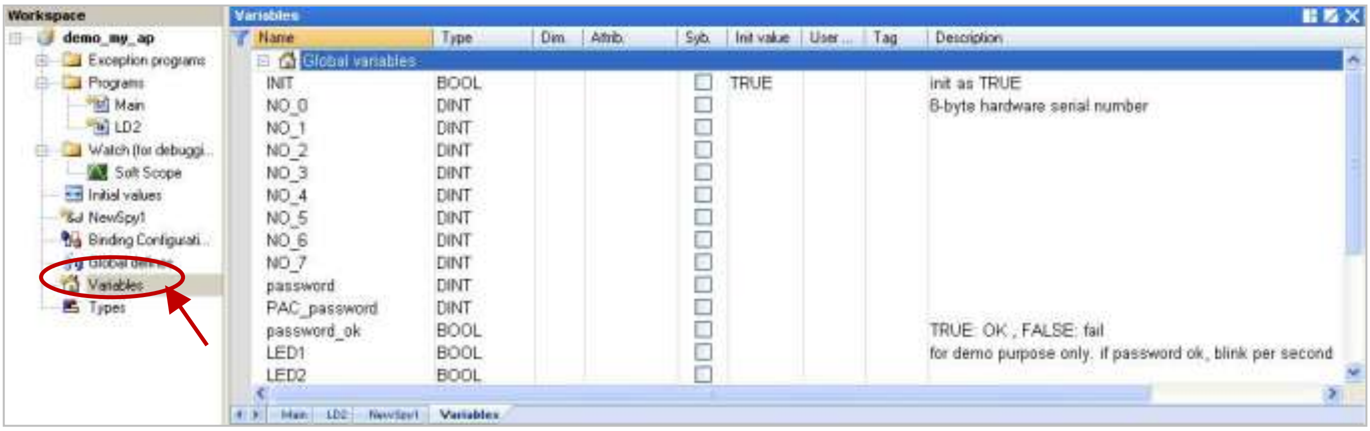


```
(* operations in the first PAC cycle *)
if  INIT  then

   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_

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

Name	Type
Global variables	
INIT	BOOL
NO_0	DINT
NO_1	DINT
NO_2	DINT
NO_3	DINT

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

Variable Declaration:



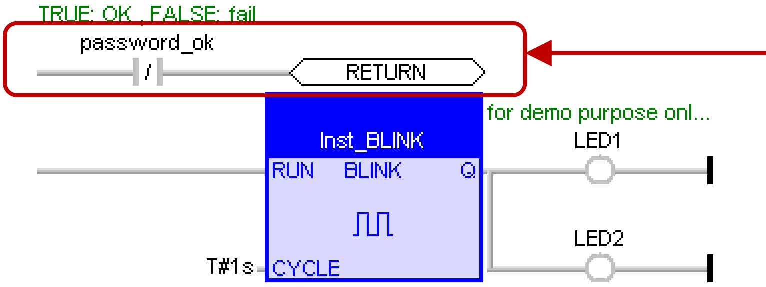
Name	Type	Dim.	Attrib.	Syb.	Init value	User ...	Tag	Description
Global variables								
INIT	BOOL			□	TRUE			init as TRUE
NO_0	DINT			□				8-byte hardware serial number
NO_1	DINT			□				
NO_2	DINT			□				
NO_3	DINT			□				
NO_4	DINT			□				
NO_5	DINT			□				
NO_6	DINT			□				
NO_7	DINT			□				
password	DINT			□				
PAC_password	DINT			□				
password_ok	BOOL			□				TRUE: OK , FALSE: fail
LED1	BOOL			□				for demo purpose only. if password ok , blink per second
LED2	BOOL			□				

ST Program - Main:

```
(* This "demo_my_ap" example can read the password from the EEPROM of 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 pre-saved password from the EEPROM, address is "157"  *)
   EEP_Read( 157 , password ) ;

   (*  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“, it means the password is incorrect, then will exit the program. Only when the password is correct can the program execute continuously, and then your application can be protected from the unauthorized access.

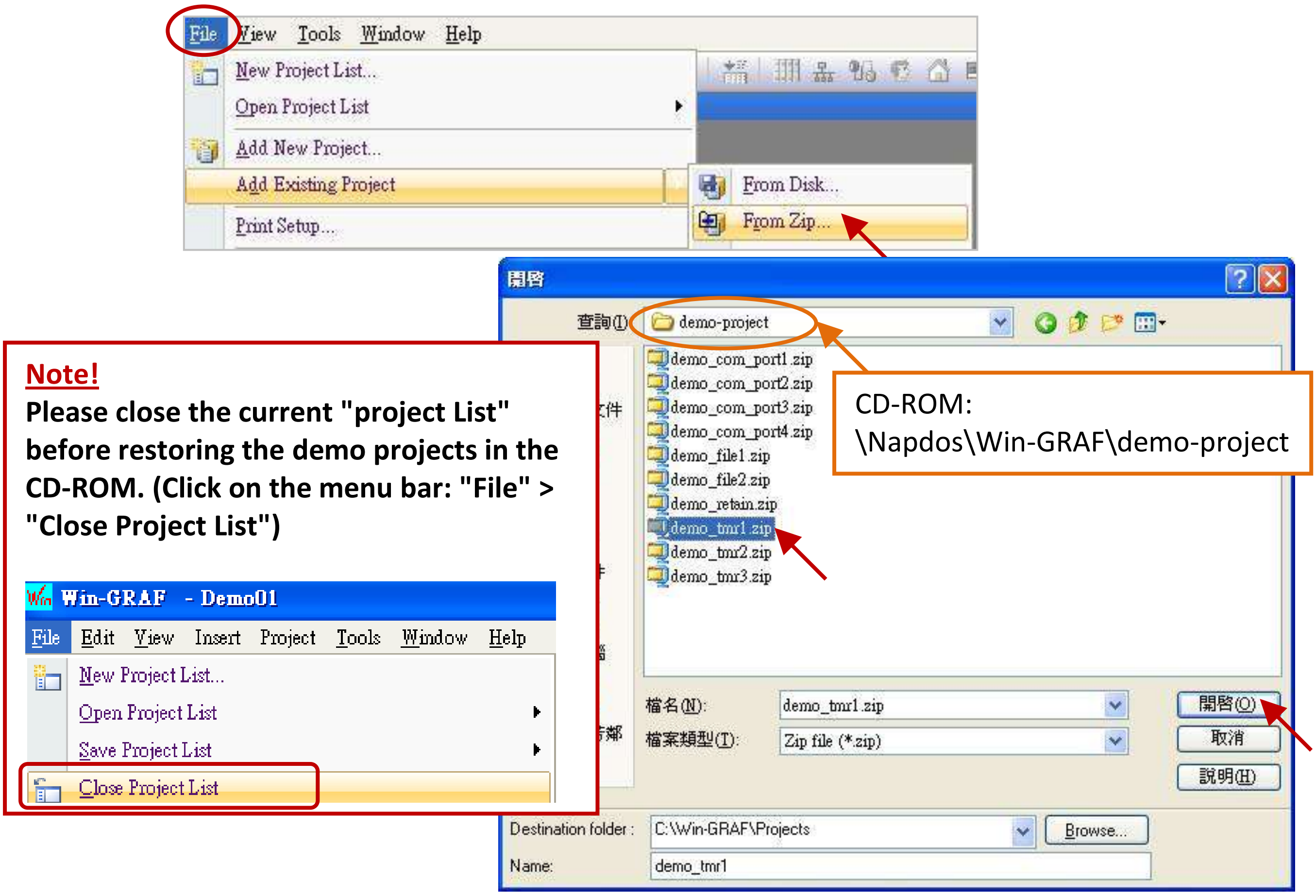


This example uses only one "LD2" program, if your application as well as other programs, please add the "password_ok" checking code for each program.

Chapter 12 Demo Project Descriptions

There are some demo projects in the Win-GRAF shipping CD (CD-ROM: **\Napdos\Win-GRAF\demo-project**) which some of them will be introduced in the following sections.
Before using the demo projects, follow the steps below:

1. Click on the menu bar "File" > "Add Existing Project" > "From Zip" to open a project (e.g., "demo_tmr1.zip").



檔名(N): demo_tmr1.zip

檔案類型(T): Zip file (*.zip)

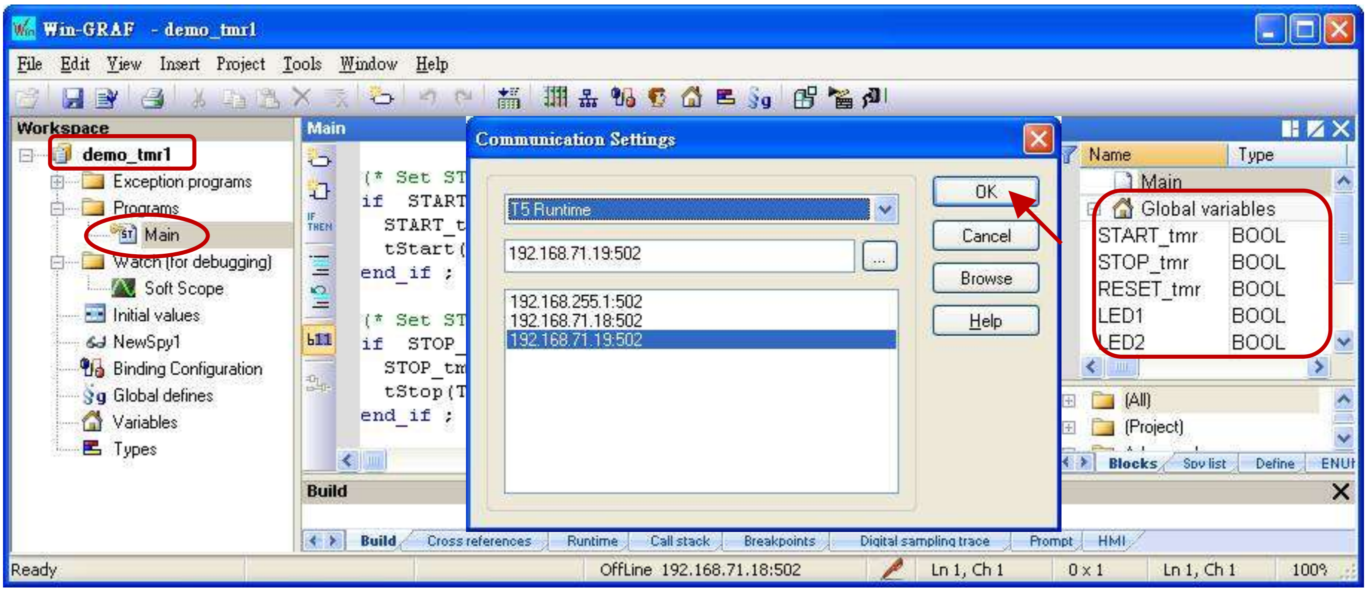
Destination folder : C:\Win-GRAF\Projects

Name: demo_tmr1

Note!
Please close the current "project List" before restoring the demo projects in the CD-ROM. (Click on the menu bar: "File" > "Close Project List")

CD-ROM:
\Napdos\Win-GRAF\demo-project

2. Double click "Main" to open the ST program, and can view/add variables in the variable area.
3. Mouse right-click on the project name ("demo_tmr1"), and select "Communication Parameters" to set up the IP Address of your PAC. (Refer Section 2.3.5)



```
(* Set ST
if  START
  START_t
  tStart (
end_if ;

(* Set ST
if  STOP
  STOP_tm
  tStop (T
end_if ;
```

T5 Runtime

192.168.71.19:502

192.168.255.1:502
192.168.71.18:502
192.168.71.19:502

Name	Type
START_tmr	BOOL
STOP_tmr	BOOL
RESET_tmr	BOOL
LED1	BOOL
LED2	BOOL

OffLine 192.168.71.18:502   Ln 1, Ch 1   0 x 1   Ln 1, Ch 1   100%

12.1 Timer Operations

12.1.1 Start, Stop and Reset the Timer

Refer P12-1 to open the project ("demo_tmr1.zip"), and can view/add variables in the variable area.

ST Program:

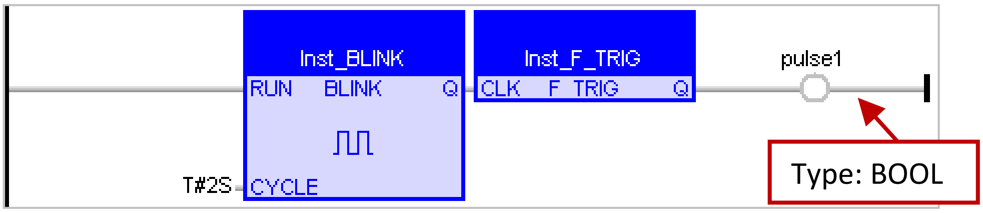
```
(* Declare “START_tmr”,  “STOP_tmr”,  “RESET_tmr”,  “LED1”,  “LED2” as BOOL
    Declare “TMR1” as TIME  *)
(* Set  START_tmr as TRUE to start ticking Timer TMR1  *)
 IF    START_tmr    THEN
   START_tmr  := FALSE ;
   TSTART  (TMR1) ;
END_IF;

(* 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;
```

12.1.2 Periodic Operations

Refer P12-1 to open the project ("demo_tmr2.zip"), and can view/add variables in the variable area.

Function "BLINK" plus Function Block "F_TRIG" can produce a Pluse TRUE at regular intervals, so it can be applied in the periodic operations.

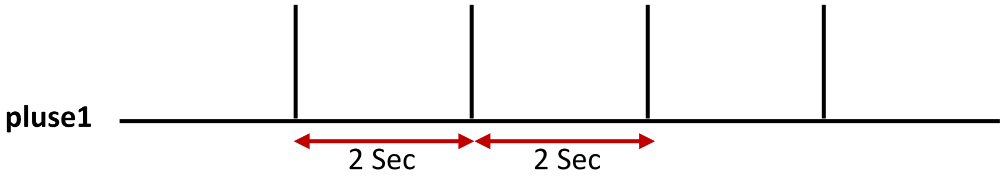
LD Program:



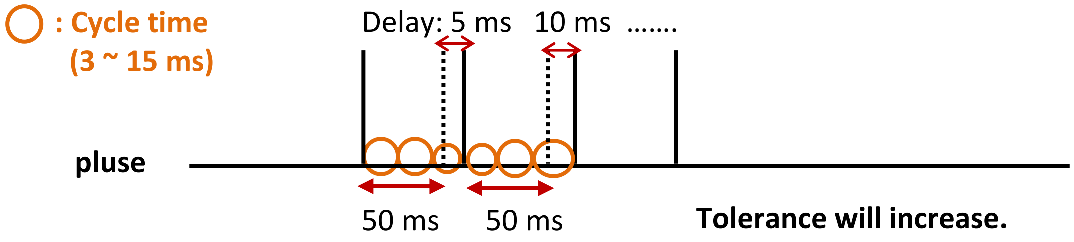
ST Program:

```
 IF   pluse1  THEN
   (*  do periodic operations here *)
    • • •
END_IF;
```

The "BLINK" and "F_TRIG" above will produce a Pluse TRUE every two seconds, but this method has a drawback:



If the interval time of the period is shorter (e.g., 100 ms per period or less; or the PAC Cycle Time is larger, such as 20 ~ 50 ms, generally 3 ~ 15 ms), then the period operations will be inaccurate. For example, to do a periodic operation every 50 ms, as compared to 250 ms or 2 seconds, the interval time of 50 ms is very close to the PAC Cycle Time, if using Function Blocks **"BLink" plus "F_TRIG"**, it is easy to accumulate the output delay time, therefore the final operation time will become inaccurate.



To improve the situation above, the following coding method will be more accurate:
 (Tolerance will not increase.)

Refer P12-1 to open the project ("demo_tmr3.zip"), and can view/add variables in the variable area.

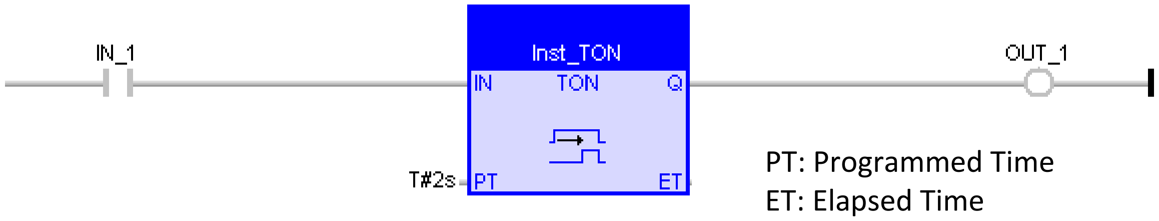
ST Program:

```
(* 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, please reset it automatically to "0" after 10 or 18 hours.

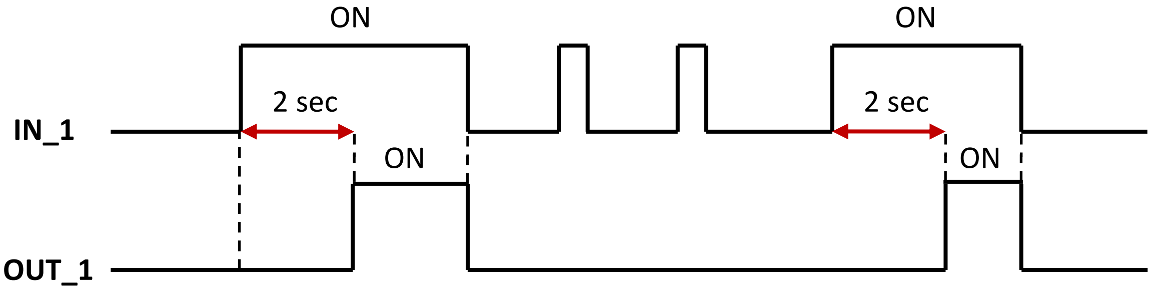
12.1.3 Detect the Steady ON or Steady OFF Signal

"TON" Function Block can detect the steady "**ON**" signal. (Keeps "**ON**" for a minimum period of time.)

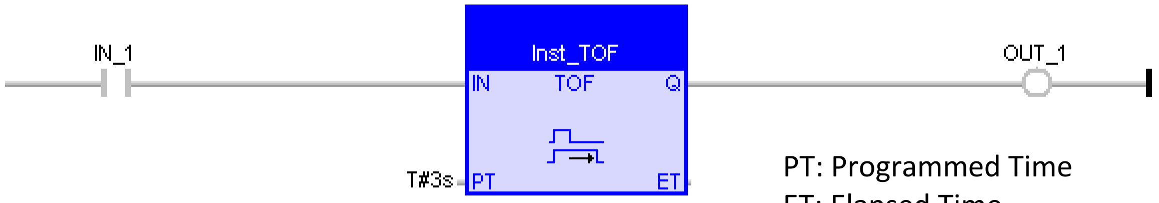


PT: Programmed Time
ET: Elapsed Time

As the picture above, this function can detect the steady "**ON**" signal that keep at least 2 seconds.

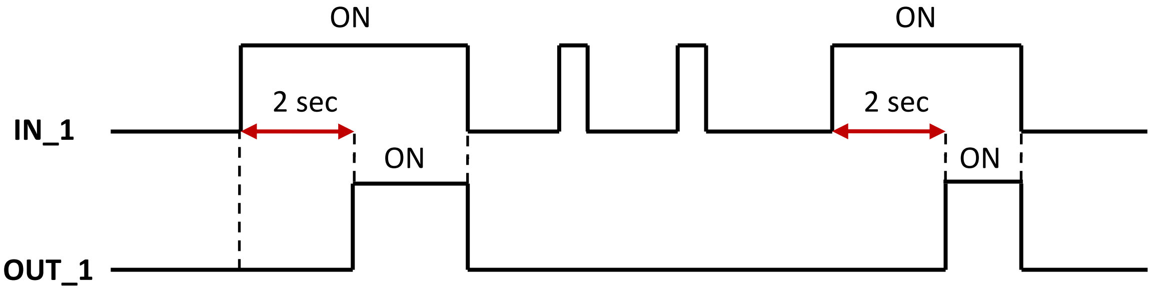


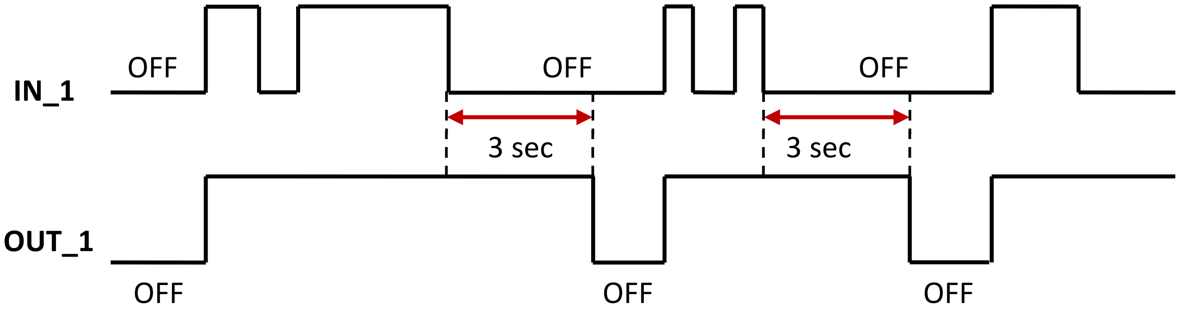
"TOF" Function Block can detect the steady **"OFF"** signal. (Keeps **"OFF"** for a minimum period of time.) that can keep **"OFF"** for a period of time.



PT: Programmed Time
ET: Elapsed Time

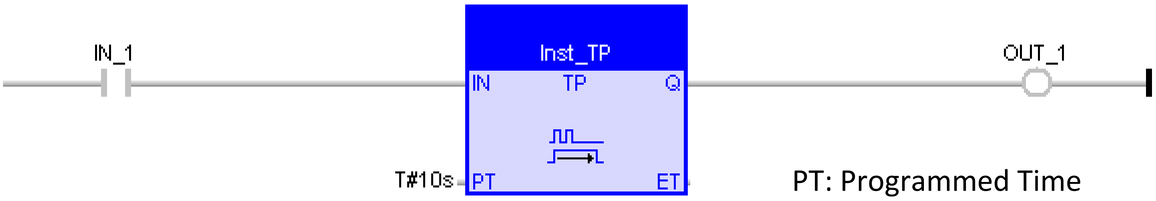
As the picture above, this function can detect the steady **"OFF"** signal that keep at least 3 seconds.





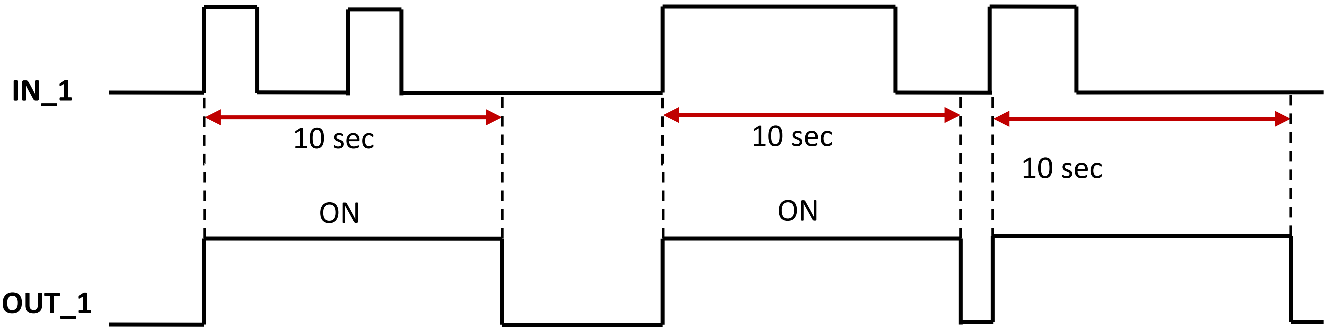
12.1.4 Keep Outputting ON for Some Time after Triggering

"TP" Function Block can keep outputting "ON" for some time after triggering (e.g., from OFF to ON).



PT: Programmed Time
ET: Elapsed Time

As the picture above, after triggering, it can keep outputting **"ON"** signal for 10 seconds.

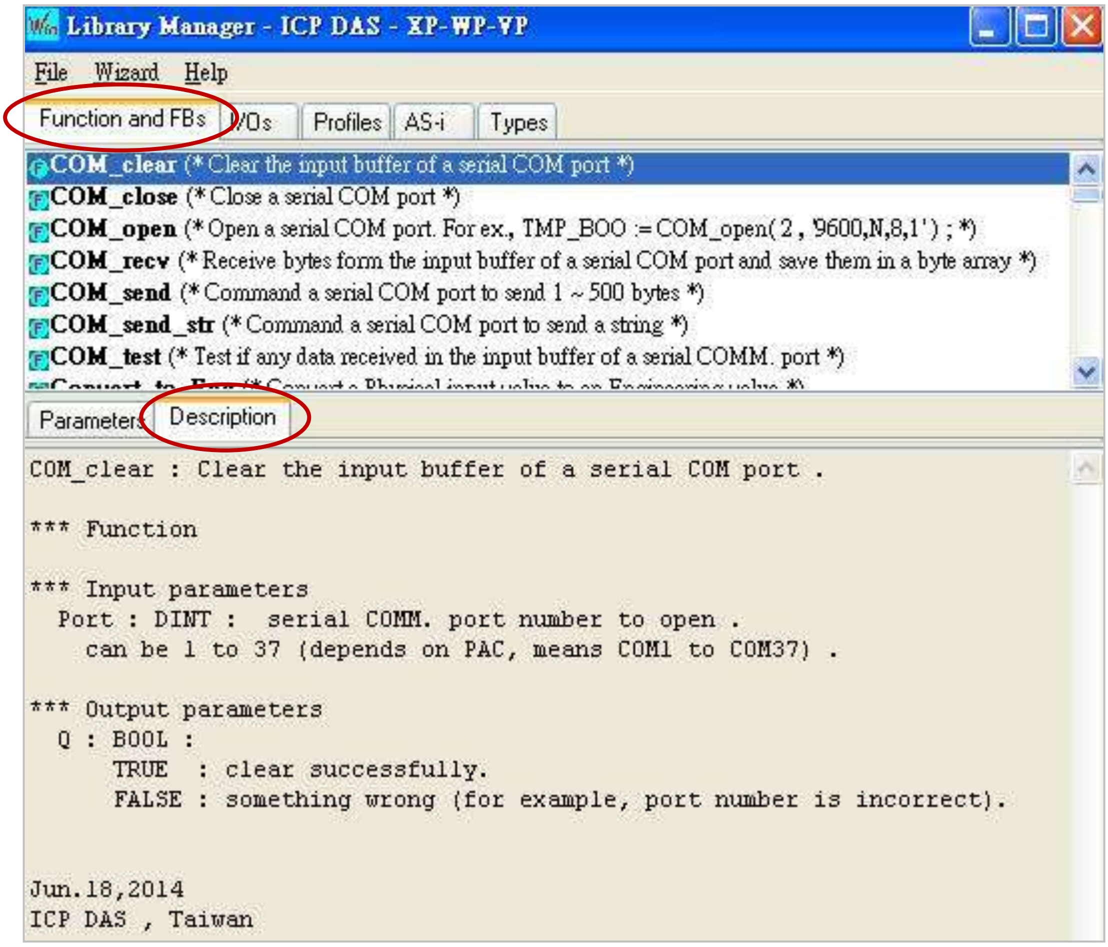


12.2 Operations of Serial Port Communication

Users can directly operate the serial port (e.g., RS-232, RS-485 or RS-422 Port) to achieve some specific communication protocol. The following Functions can be used to directly operate the serial port.

Functions	Descriptions
COM_open	Open a serial COM port.
COM_close	Close a serial COM port.
COM_clear	Clear the input buffer of a serial COM port.
COM_test	Test if any data received in the input buffer of a serial COMM port.
COM_send	Command a serial COM port to send 1~500 bytes.
COM_send_str	Command a serial COM port to send a String.
COM_recv	Receive bytes from the input buffer of a serial COM port and save them in a byte array.
COM_status	Get the current status of a serial COM port.

Please refer Section 1.2.3 to open the Library Manager and find the detail Function descriptions.



COM_clear (* Clear the input buffer of a serial COM port *)
COM_close (* Close a serial COM port *)
COM_open (* Open a serial COM port. For ex., TMP_BOO := COM_open(2 , '9600,N,8,1') ; *)
COM_recv (* Receive bytes form the input buffer of a serial COM port and save them in a byte array *)
COM_send (* Command a serial COM port to send 1 ~ 500 bytes *)
COM_send_str (* Command a serial COM port to send a string *)
COM_test (* Test if any data received in the input buffer of a serial COMM. port *)
Convert_to_Eng (* Convert a Physical input value to an Engineering value *)

```
COM_clear : Clear the input buffer of a serial COM port .

*** Function

*** Input parameters
  Port : DINT :  serial COMM. port number to open .
    can be 1 to 37 (depends on PAC, means COM1 to COM37) .

*** Output parameters
  Q : BOOL :
      TRUE  : clear successfully.
      FALSE : something wrong (for example, port number is incorrect).


Jun.18,2014
ICP DAS , Taiwan
```

12.2.1 Send a String by COM Port

Refer P12-1 to open the project ("demo_com_port1.zip"), and view/add variables in the variable area.

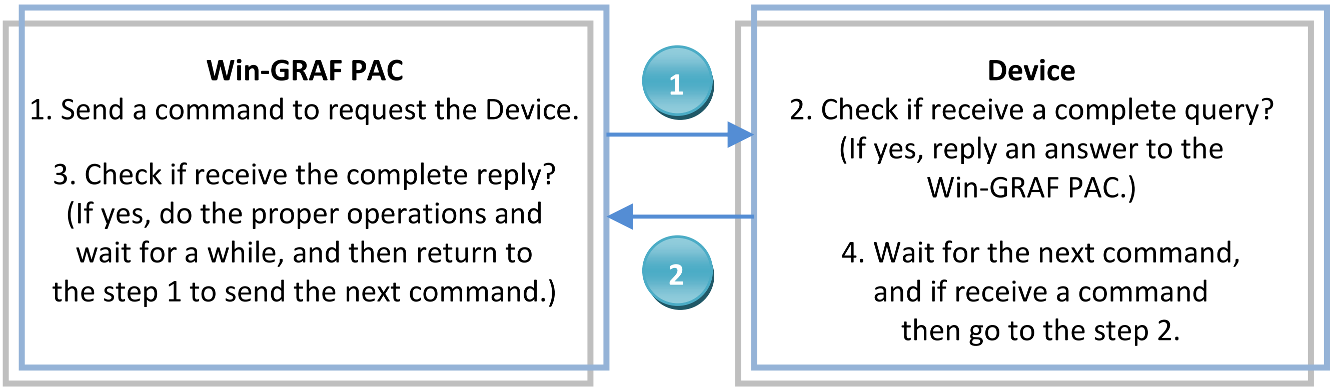
ST Program: This program can send a String every 2 seconds by the PAC COM1 (parameters: `9600,N,8,1‘) (e.g., < CNT1 = 1 > or < CNT1 = 25 >).

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”

```
  (* 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 ;

    (* 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;
```

12.2.2 Request/Answer the Device by COM Port

If an application needs to use RS-232/485/422 Port to get the data from other devices, the steps are as the following request and answer:



Win-GRAF PAC

1. Send a command to request the Device.

3. Check if receive the complete reply? (If yes, do the proper operations and wait for a while, and then return to the step 1 to send the next command.)

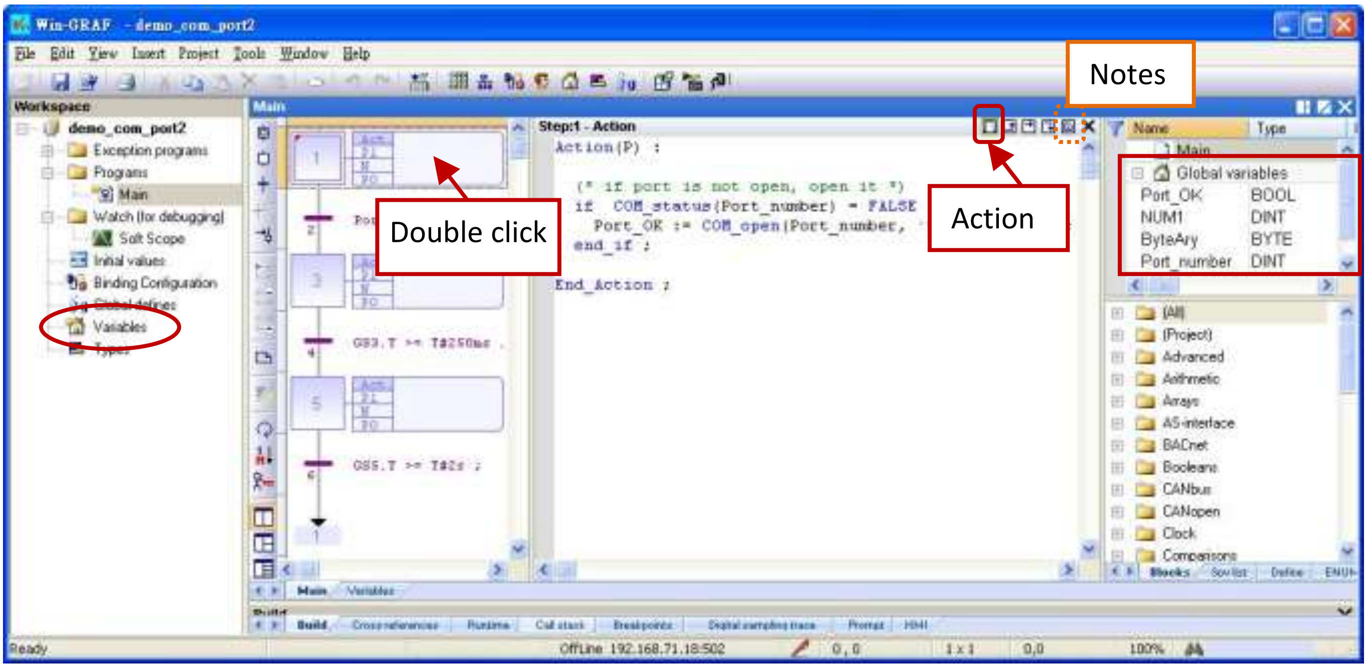
Device

2. Check if receive a complete query? (If yes, reply an answer to the Win-GRAF PAC.)

4. Wait for the next command, and if receive a command then go to the step 2.

Refer P12-1 to open the project ("demo_com_port2.zip"), and view/add variables in the variable area.

Note: Double click “Action” will first open a “Notes” window, then click “Action” icon to see the code.

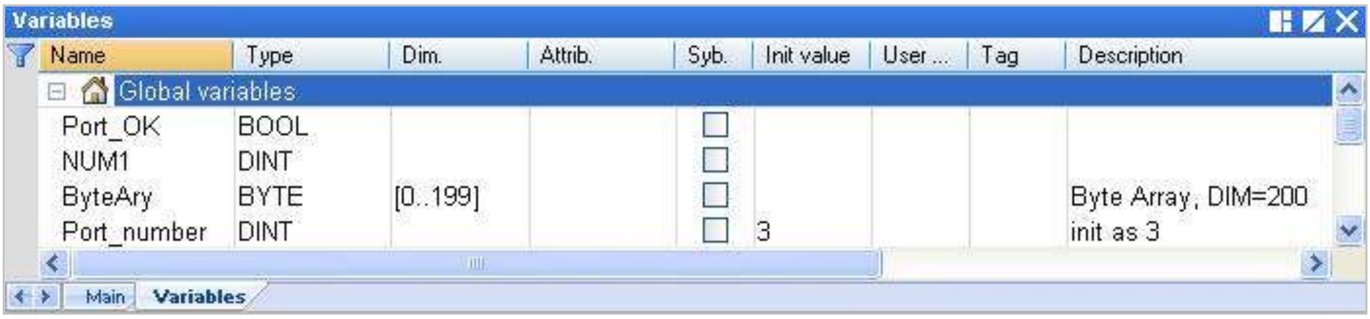


```
Action(P) :

  (* if port is not open, open it *)
  if  COM_status(Port_number) = FALSE
    Port_OK := COM_open(Port_number,
  end_if ;

End_Action ;
```

```
GS3.T >= T#250ms ;
GS5.T >= T#2s ;
```

Name	Type
Port_OK	BOOL
NUM1	DINT
ByteAry	BYTE
Port_number	DINT

You can click the “Variables” tag to open the Variables window.



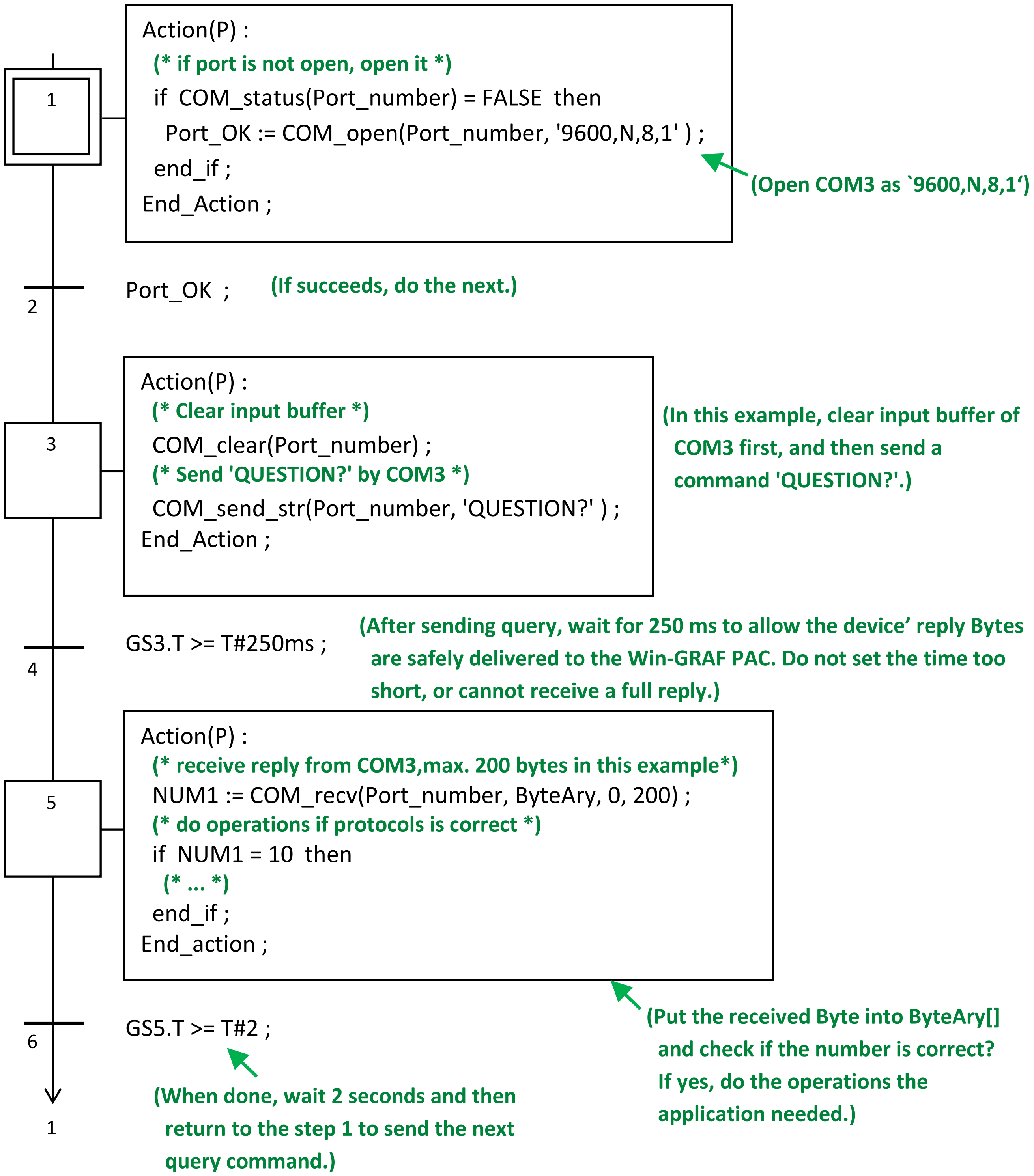
Variables

Name	Type	Dim.	Attrib.	Syb.	Init value	User ...	Tag	Description
Global variables								
Port_OK	BOOL			□				
NUM1	DINT			□				
ByteAry	BYTE	[0..199]		□				Byte Array, DIM=200
Port_number	DINT			□	3			init as 3

In this example, the Win-GRAF PAC sends a string 'QUESTION?' by COM3 to the device, and then wait for the reply and does operations. After the operations, waits 2 seconds, and then sends the same command 'QUESTION?', and repeated.

SFC Program:

(Declare "Port_OK" as BOOL ; "NUM1" as DINT ; "ByteAry" as BYTE and Dim. = "200" ;
"Port_number" as DINT and has initial value "3")



```
Action(P) :
 (* 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 ;
End_Action ;
```

(Open COM3 as `9600,N,8,1‘)

```
Port_OK ;
```

(If succeeds, do the next.)

```
Action(P) :
 (* Clear input buffer *)
 COM_clear(Port_number) ;
 (* Send 'QUESTION?' by COM3 *)
 COM_send_str(Port_number, 'QUESTION?' ) ;
End_Action ;
```

(In this example, clear input buffer of COM3 first, and then send a command 'QUESTION?'.)

```
GS3.T >= T#250ms ;
```

(After sending query, wait for 250 ms to allow the device' reply Bytes are safely delivered to the Win-GRAF PAC. Do not set the time too short, or cannot receive a full reply.)

```
Action(P) :
 (* receive reply from COM3,max. 200 bytes in this example*)
 NUM1 := COM_recv(Port_number, ByteAry, 0, 200) ;
 (* do operations if protocols is correct *)
 if NUM1 = 10 then
   (* ... *)
 end_if ;
End_action ;
```

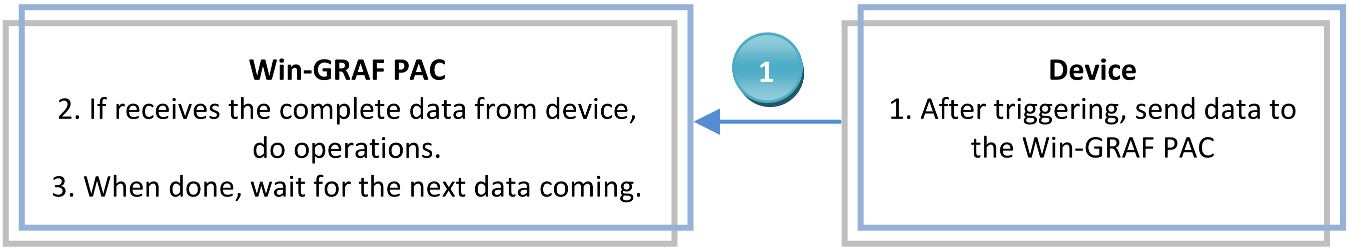
(Put the received Byte into ByteAry[] and check if the number is correct? If yes, do the operations the application needed.)

```
GS5.T >= T#2 ;
```

(When done, wait 2 seconds and then return to the step 1 to send the next query command.)

12.2.3 Wait Data Coming from Remote Device by COM Port

This way is common in the general store or supermarket, such as using the barcode readers. After reading the barcode of the product, it will send the barcode data to the Win-GRAF PAC's COM Port (RS-232/485/422), and need not to reply any messages.



Win-GRAF PAC
2. If receives the complete data from device, do operations.
3. When done, wait for the next data coming.

Device
1. After triggering, send data to the Win-GRAF PAC

Refer P12-1 to open the project ("demo_com_port3.zip"), and view/add variables in the variable area.

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 "3".

```
(* 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 waiting, and will test if COM3 has data?
If returns TRUE, means COM3 has data.
Then set STEP1 to "1", T1 to "0" and start timing.

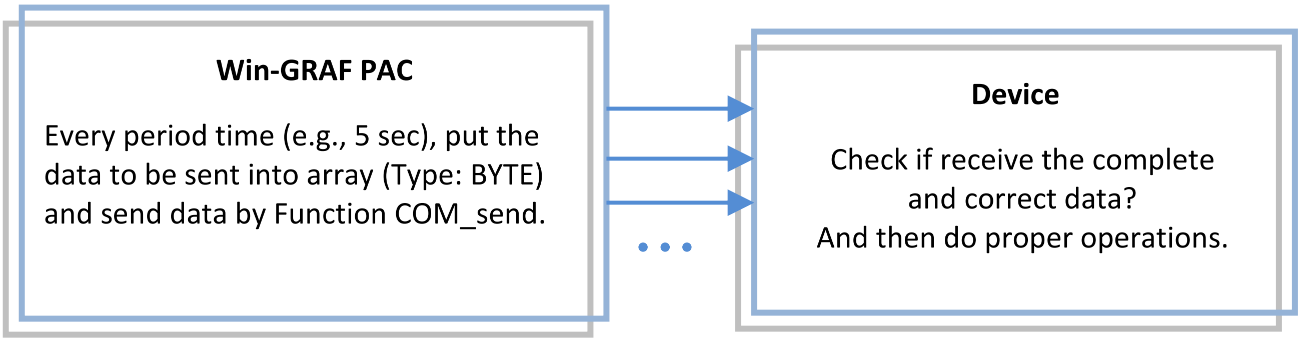
```
  (* wait 250 ms, then receive all bytes form COM port *)
 1:
   if  T1 >= T#250ms  then
    Tstop(T1) ;
    T1 := T#0s ;
    STEP1 := 0 ;

    (* 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 ;
```

STEP = 1: means the data is sending in, and will wait 250 ms to receive all data and put them into an array. The waiting time concerns the device specifications and the Baud Rate. If set the time too short, may receive data incompleted. Remember to set STEP1 to "0" to wait for the data coming next time.

**When receive data, check if data is correct?
If yes, do the operations the application needed.**

12.2.4 Report Data Periodically to Remote Device by COM Port

If wants to periodically report data to other devices by RS-232/485/422 Port, do as follows.



Win-GRAF PAC

Every period time (e.g., 5 sec), put the data to be sent into array (Type: BYTE) and send data by Function COM_send.

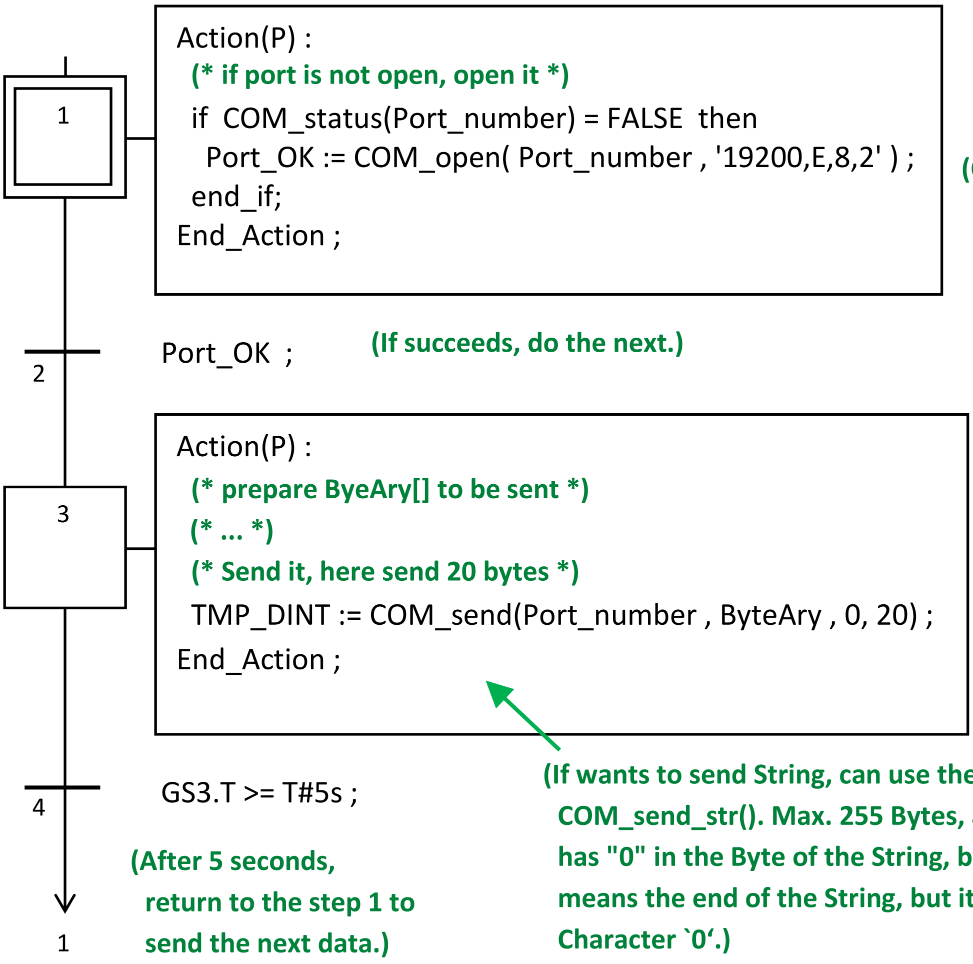
Device

Check if receive the complete and correct data?
And then do proper operations.

Refer P12-1 to open the project ("demo_com_port4.zip"), and view/add variables in the variable area.

SFC Program: (Refer Section 12.2.2 to open the "Action" window.)

(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".)



```
Action(P) :
  (* if port is not open, open it *)
  if  COM_status(Port_number) = FALSE  then
    Port_OK := COM_open( Port_number , '19200,E,8,2' ) ;
  end_if;
End_Action ;
```

(Open COM2 as `19200,E,8,2‘)

```
Port_OK  ;
```

(If succeeds, do the next.)

```
Action(P) :
  (* prepare ByeAry[] to be sent *)
  (* ... *)
  (* Send it, here send 20 bytes *)
  TMP_DINT := COM_send(Port_number , ByteAry , 0, 20) ;
End_Action ;
```

(The sent data concers with your application, please use COM_send to send data after the data is ready. Max. 500 Bytes)

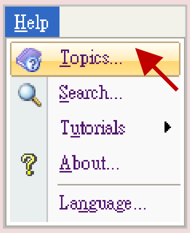
```
GS3.T >= T#5s ;
```

(After 5 seconds, return to the step 1 to send the next data.)

(If wants to send String, can use the Function COM_send_str(). Max. 255 Bytes, and can not has "0" in the Byte of the String, because "0" means the end of the String, but it can be the Character `0‘.)

12.3 Read/Write Data from/to a File in The PAC

The Win-GRAF Workbench provides the following Functions to enable sequential read/write operations in disk files of the Win-GRAF PAC.

<table>
<tr><th>Functions</th><th>Descriptions</th></tr>
<tr><td colspan="2">Please click the menu bar "Help" > "Topics" and type the searching key word “File” to see more detail information in the topic of the “File Management functions”.  </td></tr>
<tr><td>F_ROPEN</td><td>Open a file for reading.</td></tr>
<tr><td>F_WOPEN</td><td>Create or reset a file and open it for writing.</td></tr>
<tr><td>F_AOPEN</td><td>Create or open a file in append mode.</td></tr>
<tr><td>F_CLOSE</td><td>Close an open file.</td></tr>
<tr><td>F_EOF</td><td>Test if the end of file is reached in a file open for read.</td></tr>
<tr><td>FA_READ</td><td>Read a DINT integer from a binary file.</td></tr>
<tr><td>FA_WRITE</td><td>Write a DINT integer to a binary file.</td></tr>
<tr><td>FM_READ</td><td>Read a STRING value from a text file</td></tr>
<tr><td>FM_WRITE</td><td>Write a STRING value to a text file.</td></tr>
<tr><td>FB_READ</td><td>Read binary data from a file.</td></tr>
<tr><td>FB_WRITE</td><td>Write binary data to a file.</td></tr>
<tr><td>F_EXIST</td><td>Test if a file exists.</td></tr>
<tr><td>F_GETSIZE</td><td>Get the size of a file.</td></tr>
<tr><td>F_COPY</td><td>Copy a file.</td></tr>
<tr><td>F_DELETE</td><td>Remove a file.</td></tr>
<tr><td>F_RENAME</td><td>Rename a file.</td></tr>
<tr><td colspan="2">Refer Section 1.2.3 to find the detailed descriptions for the following Functions.</td></tr>
<tr><td>F_dir</td><td>Create a directory.</td></tr>
<tr><td>F_cp_dir</td><td>Copy all files in a directory to another directory (exclude files in sub-directories).</td></tr>
<tr><td>F_del_dir</td><td>Delete a directory and all files inside it (exclude sub-directories and files inside sub-directories).</td></tr>
</table>

Note: The Win-GRAF PAC of ICP DAS does not support Functions “F_SAVERETAIN” and “F_LOADERETAIN”.

12.3.1 Write Data to a File in the PAC

Refer P12-1 to open the project ("demo_file1.zip"), and can view/add variables in the variable area.

ST Program: This program can be used to write 10 "REAL" values to a file in 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 size of \System_Disk\ is small, recommend you may change the directory to below (Depends on your application):

WP-8xx8, WP-5xx8, VP-25W8, VP-4138:
\Micro_SD or

XP-8x48, XP-8x48-CE6, XP-8x48-Atom-CE6:
\System_Disk2

```
      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 ;
```

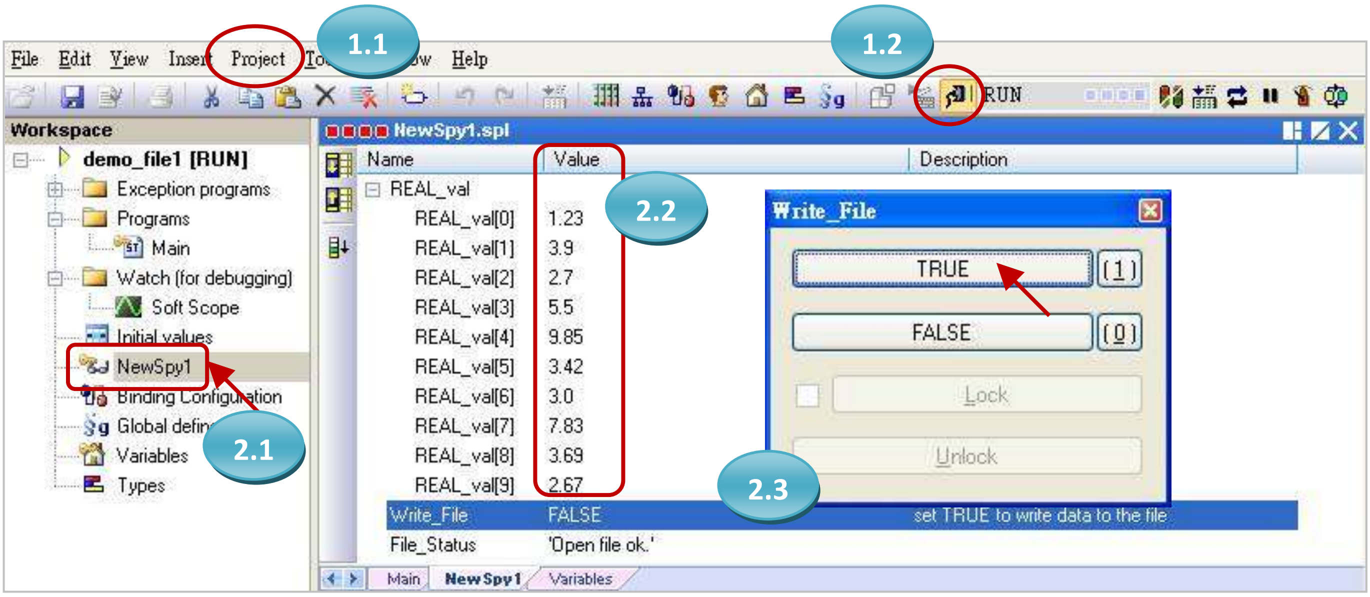
If want to save data to Integer, use the code:
Tmp_string := Any_to_string(**DINT_val[ii]**) ;
and declare Variable "DINT_val" as DINT and
Dim. at least "10" for this example.

Test Program:

In this example, when the "Write_File" is set to "TRUE", the values will be written into the file \System_Disk\Real_data1.txt in the PAC.

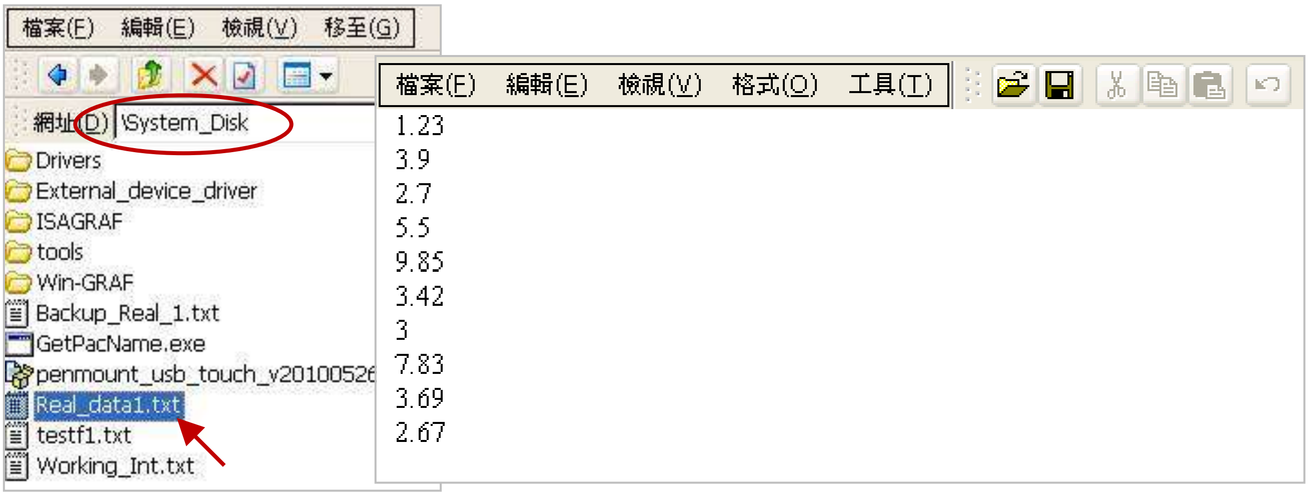
1. Please set up IP configurations (Refer P12-1), compile and download the program to the PAC. (Click on "Project" > "Build All Projects" / "On Line", if not familiar with the operation, refer to Section 2.3.4 , Section 2.3.5)

2. Click "NewSpy1" to open a Spy List and fill in the values to be written, and then set the "Write_File" become "TRUE" to write the data. (If OK, "File_Status" will show "Open file ok".)



Name	Value	Description
REAL_val		
REAL_val[0]	1.23	
REAL_val[1]	3.9	
REAL_val[2]	2.7	
REAL_val[3]	5.5	
REAL_val[4]	9.85	
REAL_val[5]	3.42	
REAL_val[6]	3.0	
REAL_val[7]	7.83	
REAL_val[8]	3.69	
REAL_val[9]	2.67	
Write_File	FALSE	set TRUE to write data to the file
File_Status	'Open file ok.'	

3. In the PAC, open the file "Real_data1.txt", can see the values filled in the step 2.



```
1.23
3.9
2.7
5.5
9.85
3.42
3
7.83
3.69
2.67
```

12.3.2 Read Data from a File in the PAC

Refer P12-1 to open the project ("demo_file2.zip"), and can view/add variables in the variable area.

ST Program: This program can be used to read 10 "REAL" values from a file in 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 size of \System_Disk\ is small, recommend you amy change the directory to below (Depends on your application):

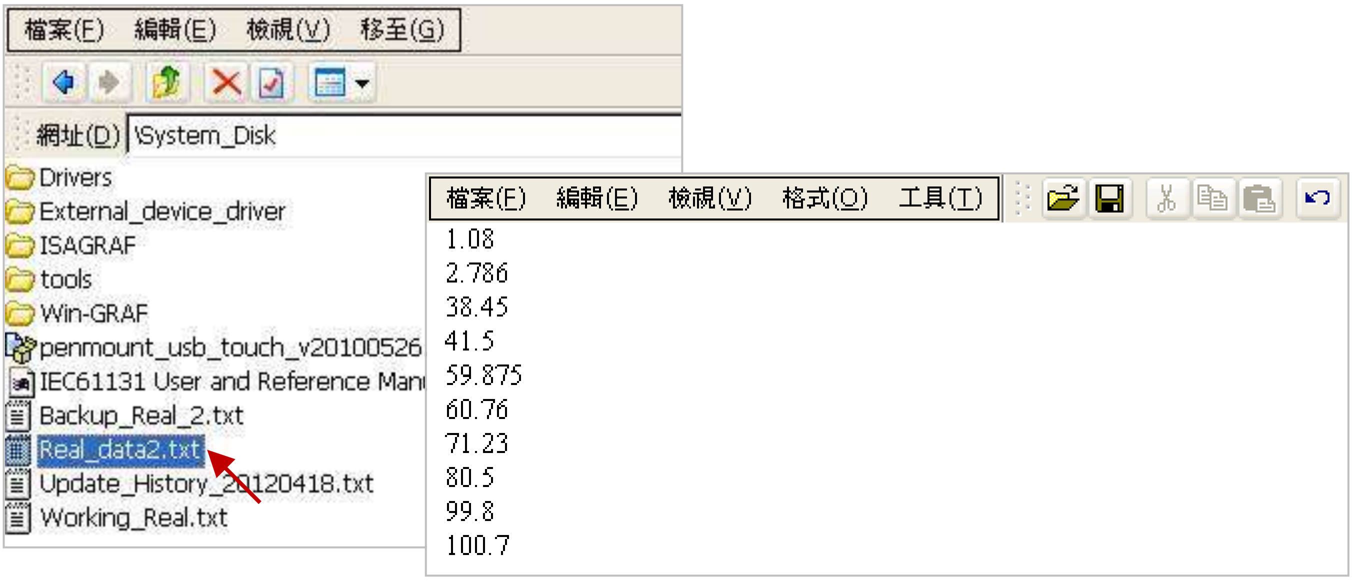
WinPAC, ViewPAC Series:
\Micro_SD or

XP PAC Series:
\System_Disk2

```
      (* 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 ;
```

If want to read Integer, use the code below:
DINT_val[ii] := Any_to_**DINT**(Tmp_string) ;
and declare Variable "DINT_val" as DINT and Dim. at least "10" for this example..

Note: In this example, when the "Read_File" is set to "TRUE", it will read the file in the PAC "\System_Disk\Real_data2.txt", please make sure the file already exists in the PAC.



\System_Disk

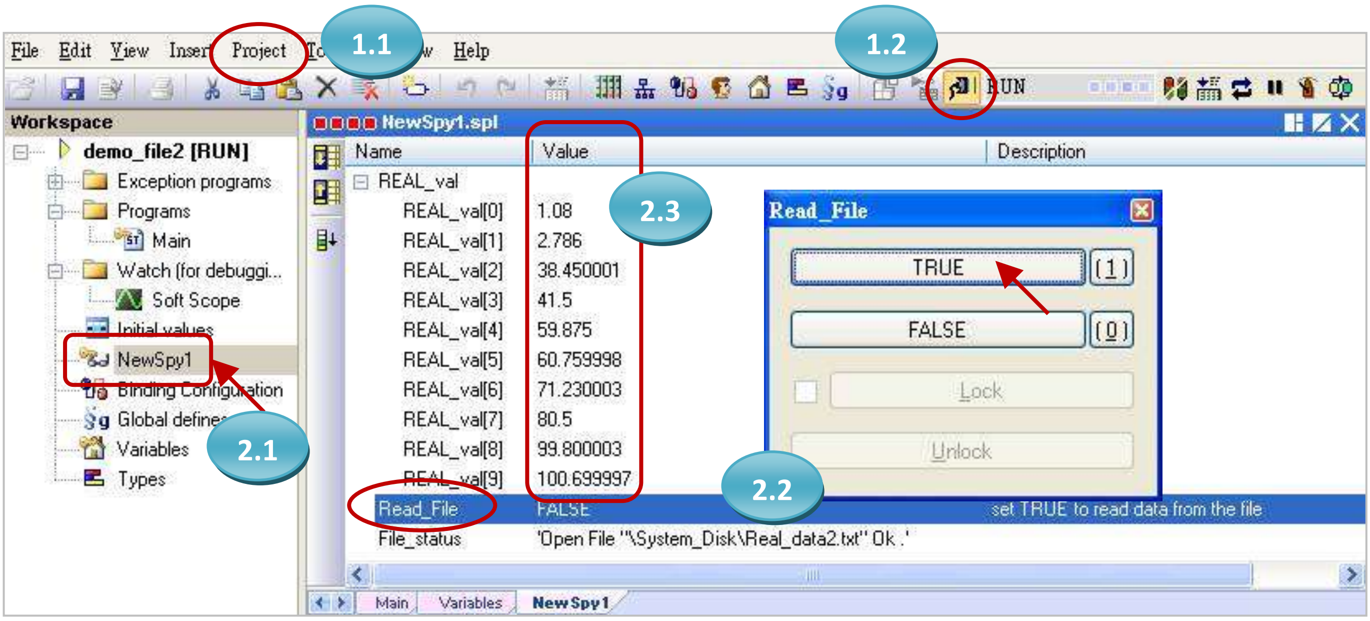
Drivers
External_device_driver
ISAGRAF
tools
Win-GRAF
penmount_usb_touch_v20100526
IEC61131 User and Reference Manu
Backup_Real_2.txt
Real_data2.txt
Update_History_20120418.txt
Working_Real.txt

```
1.08
2.786
38.45
41.5
59.875
60.76
71.23
80.5
99.8
100.7
```

Test Program:

1. Please set up IP configurations (Refer P12-1), compile and download the program to the PAC. (Click on "Project" > "Build All Projects" / "On Line", if not familiar with the operation, refer to Section 2.3.4 , Section 2.3.5)

2. Click "NewSpy1" to open a Spy List and set the "Read_File" to "TRUE" to read the data. (If OK, "File_Status" will show "Open File "\System_Disk\Real_data2.txt" Ok.")



Name	Value	Description
REAL_val		
REAL_val[0]	1.08	
REAL_val[1]	2.786	
REAL_val[2]	38.450001	
REAL_val[3]	41.5	
REAL_val[4]	59.875	
REAL_val[5]	60.759998	
REAL_val[6]	71.230003	
REAL_val[7]	80.5	
REAL_val[8]	99.800003	
REAL_val[9]	100.699997	
Read_File	FALSE	set TRUE to read data from the file
File_status	'Open File ''\System_Disk\Real_data2.txt'' Ok .'	

Note: There is one another file operation example listed in the Section 6.2. It handles many data in the file, you may refer it.

Chapter 13 VB.net 2008 Program Running in WP-8xx8 Access to Win-GRAF Variables

This chapter lists the procedure for creating the first demo program by Visual Studio .NET 2008 development tool. There are some sample programs in the WP-8xx8 CD-ROM.

VB .NET example:

CD-ROM : \napdos\Win-GRAF\demo-project\vb.net_2008_demo\

demo_vb01 :	Digital I/O demo with one I-87055W in slot 0 of the WP-8xx8.
demo_vb02 :	Analog I/O demo with one I-87024W in slot 1, one I-8017HW in slot 2.
demo_vb03 :	Read/Write Win-GRAF internal integers, timers & real variables. (No I/O)
demo_vb04 :	Read/Write Win-GRAF internal String variables. (No I/O)

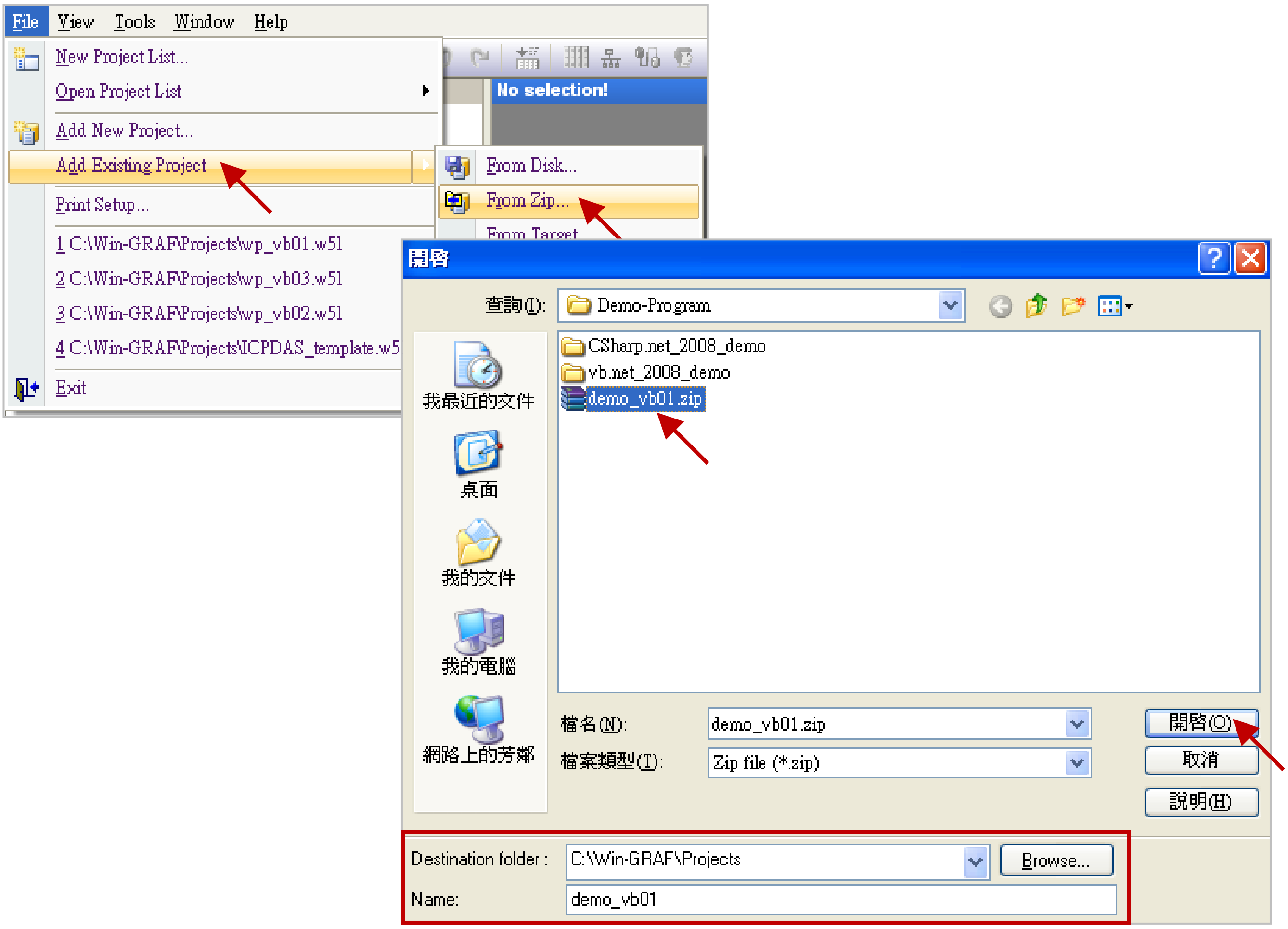
Win-GRAF example:

CD-ROM : \napdos\Win-GRAF\demo-project\
"demo_vb01.zip", "demo_vb02.zip", "demo_vb03.zip", "demo_vb04.zip"

13.1 Add an Existing Win-GRAF Project from a ZIP

Please follow these steps to restore the Win-GRAF project.

First Click “File” -> “Add Exiting Project” -> “FromZip…”. Then choose the Win-GRAF project zip file which you would like to restore. After restoring the project, you have to build the project, and then download it to the PAC.



檔名(N): demo_vb01.zip

檔案類型(T): Zip file (*.zip)

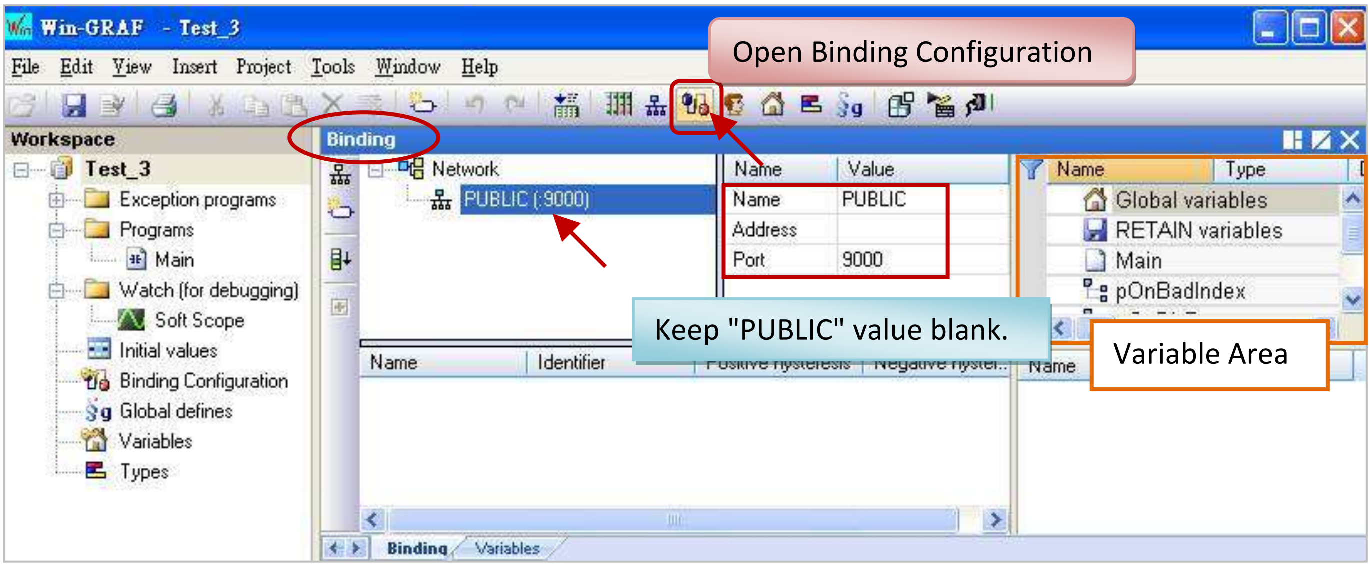
Destination folder : C:\Win-GRAF\Projects

Name: demo_vb01

13.2 Publishing the Win-GRAF Variable for .NET

If users wish to use .NET program to Read/Write the Win-GRAF variables. Except for String variable all of the variables need to use the "Open Binding Configuration" function to set an address. The following demonstrates how to publish Win-GRAF variable:

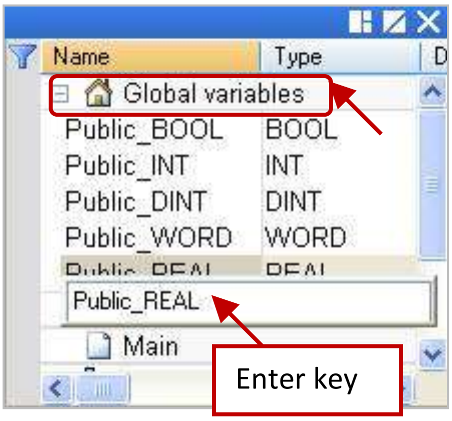
1. Click “Open Binding Configuration” on the toolbar to open the “Binding” setup window.
2. Click "PUBLIC (:9000)". Then users can modify the “Name” value. But keep the "Address" value is blank and "Port" value is fixed to 9000.



Name	Value
Name	PUBLIC
Address	
Port	9000

3. Before publishing these variables, make sure you have declared them in the Variables Area. Click “Global variables” and press the “Ins” key to insert a new variable. The following table demonstrates variables using in the “Test_3” project and you can declare them according to the needs of your application.

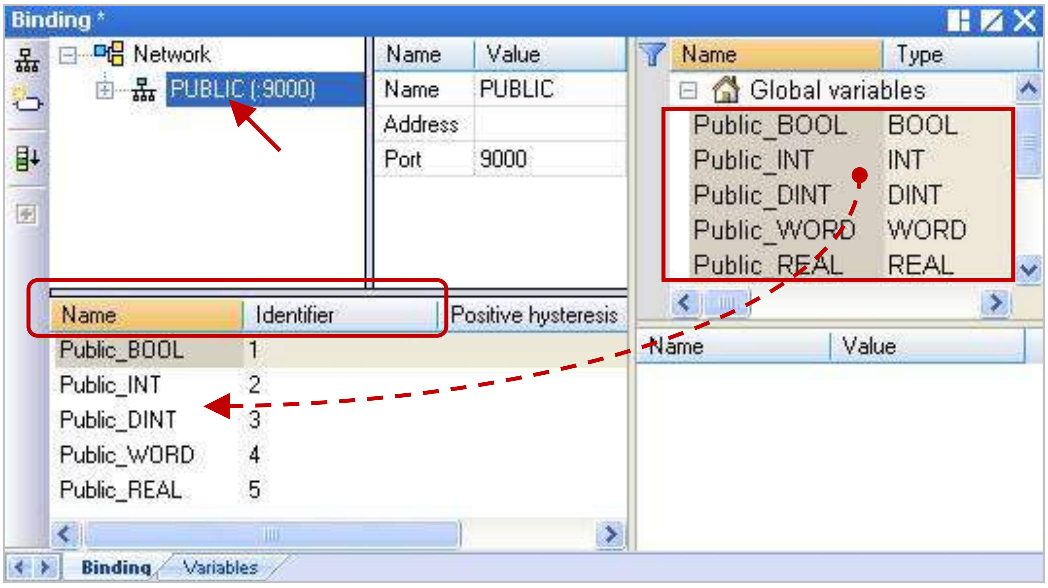
Variable name	Type
Public_BOOL	BOOL
Public_INT	INT
Public_DINT	DINT
Public_WORD	WORD
Public_REAL	REAL



4. As the figure below, click on "PUBLIC(:9000)" and drag all the needed variables to the "Name" area. The "Identifier" will generate an address number automatically. If any other VB or .NET program wants to use these public variables, it must set to the same address number (ID).

NOTE:

The "PUBLIC" allows to use up to 8192 variables, and the "Identifier" number JUST can be 1 to 8192.



Binding *

Name	Value
Name	PUBLIC
Address	
Port	9000

Name	Identifier	Positive hysteresis
Public_BOOL	1	
Public_INT	2	
Public_DINT	3	
Public_WORD	4	
Public_REAL	5	

Name	Type
Global variables	
Public_BOOL	BOOL
Public_INT	INT
Public_DINT	DINT
Public_WORD	WORD
Public_REAL	REAL

The following procedure will show you how to use the "pub_string" function to publish the Win-GRAF String variable in the ST program.

Syntax:

```
Pub_string(Address, String_val) ;
```

Address: The public address number, and its range can be 1 to 1024
String_val: The name of String variable.

Variables description:

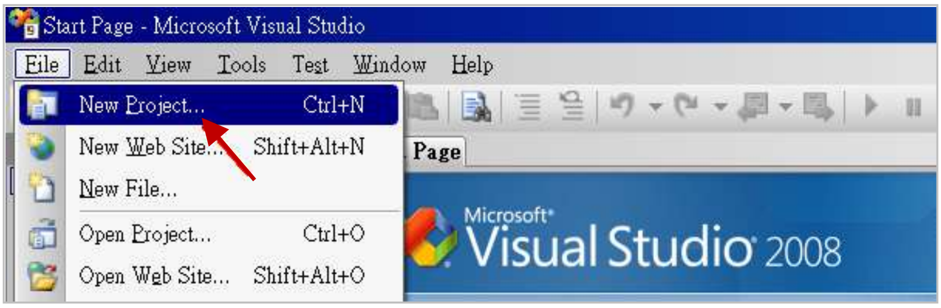
Name	Type	Description
Init	BOOL	Set the initial value as TRUE.
Tmp_val	BOOL	TRUE: Binding succeeds. FALSE: Binding fails.
msg1	STRING, Length is 100	String variable for demo purpose. **NOTE: The String length could be 1 to 255.**
msg2	STRING, Length is 32	
msg3	STRING, Length is 60	

ST program:

```
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;
```

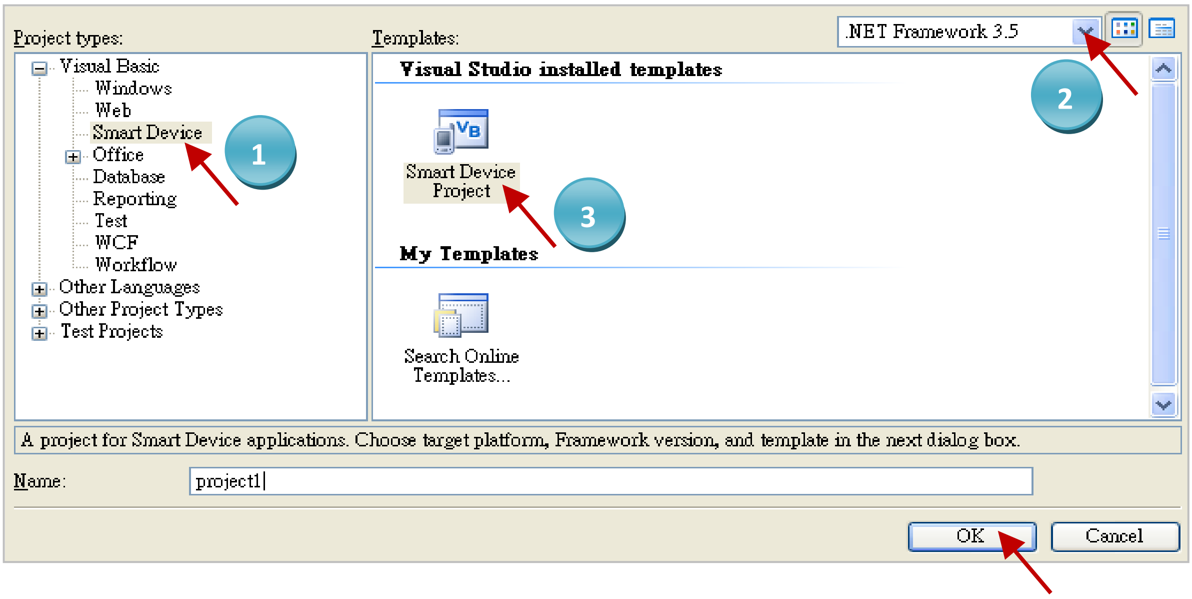
13.3 Create a new VB.NET project

1. First, run Microsoft Visual Studio .NET 2008 software, and then choose “File” > “New Project”.



Start Page - Microsoft Visual Studio
File Edit View Tools Test Window Help
New Project... Ctrl+N
New Web Site... Shift+Alt+N
New File...
Open Project... Ctrl+O
Open Web Site... Shift+Alt+O
Page
Microsoft Visual Studio 2008

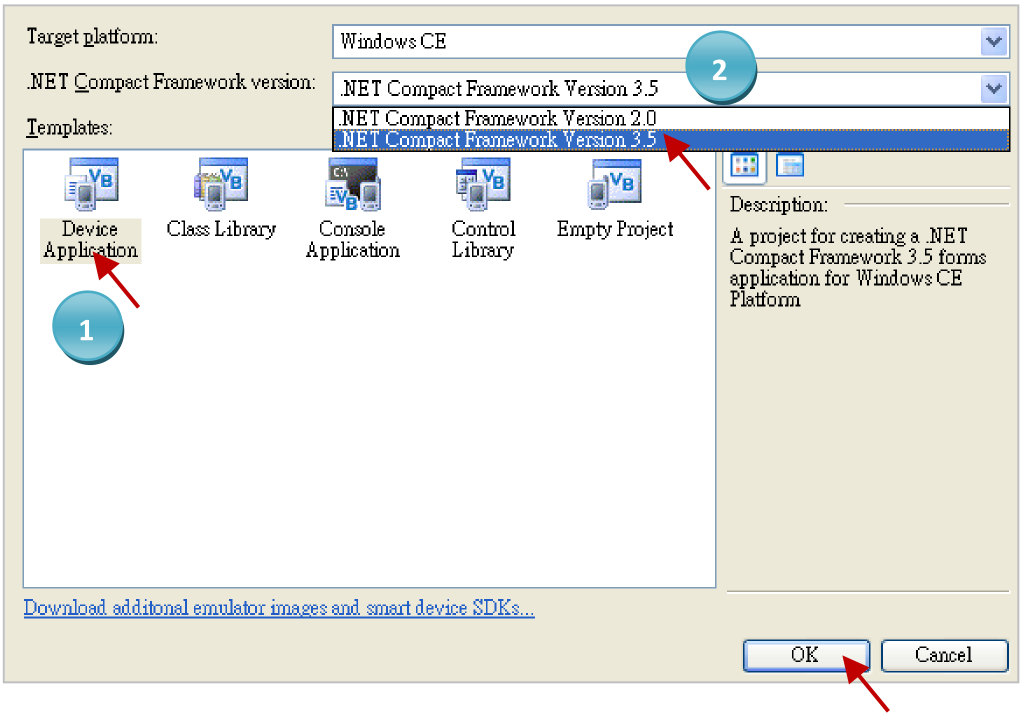
2. Click “Smart Device” on the left, and then select “.NET Framework 3.5” and “Smart Device Project”. Entering a proper project name and click “OK”.



A project for Smart Device applications. Choose target platform, Framework version, and template in the next dialog box.

Name: project1

3. Select the "Device Application" and "Windows CE" and “.NET Compact Framework Version 3.5”, then click “OK”.



Target platform: Windows CE

.NET Compact Framework version: .NET Compact Framework Version 3.5

.NET Compact Framework Version 2.0
.NET Compact Framework Version 3.5
Templates:
Device Application
Class Library
Console Application
Control Library
Empty Project
1
2
Download additonal emulator images and smart device SDKs...
OK
Cancel

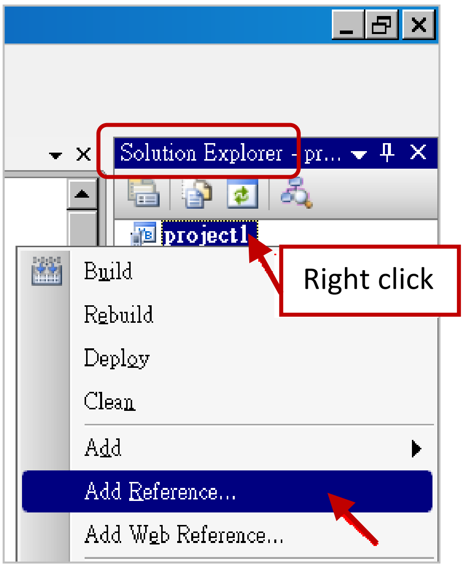
Description: A project for creating a .NET Compact Framework 3.5 forms application for Windows CE Platform

13.3.1 Add Project Reference

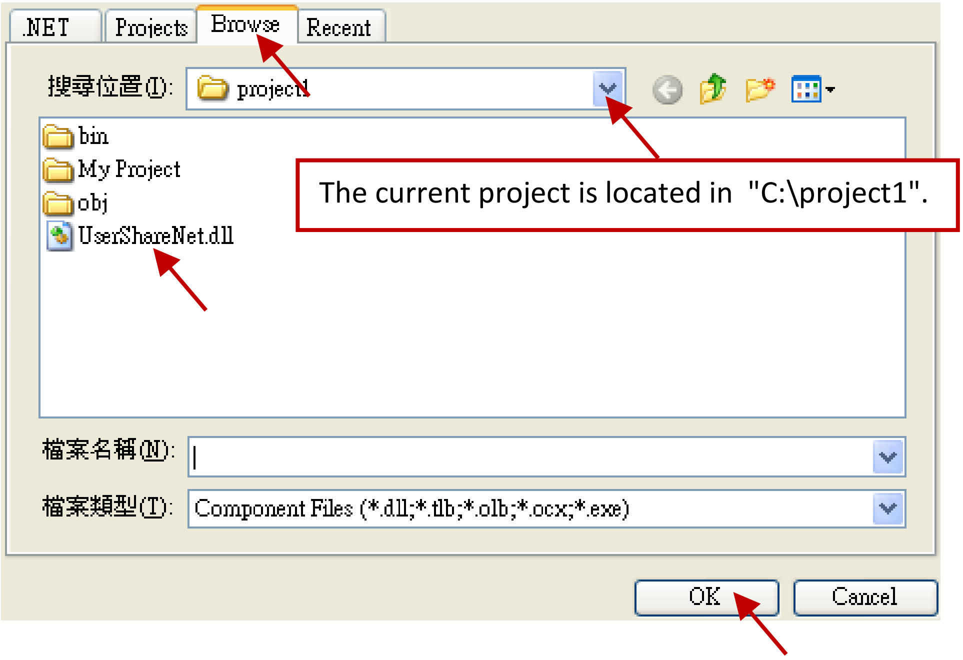
The “UserShareNet.dll” library contains all functions of data exchange with Win-GRAF variables. Before you use the “UserShareNet” keyword in the program, you must add the “UserShareNet.dll” into the reference list of your application.

1. Copy the “UserShareNet.DLL” from Win-PAC’s shipment CD (\napdos\Win-GRAF\WP-8xx8\vb.net_2008_demo\wp_vb01\vb01\) to your project folder (e.g., “C:\project1\”)

2. Right click on the project name (e.g., “project1”) in the “Solution Explorer” window, and then select “Add Reference …”.



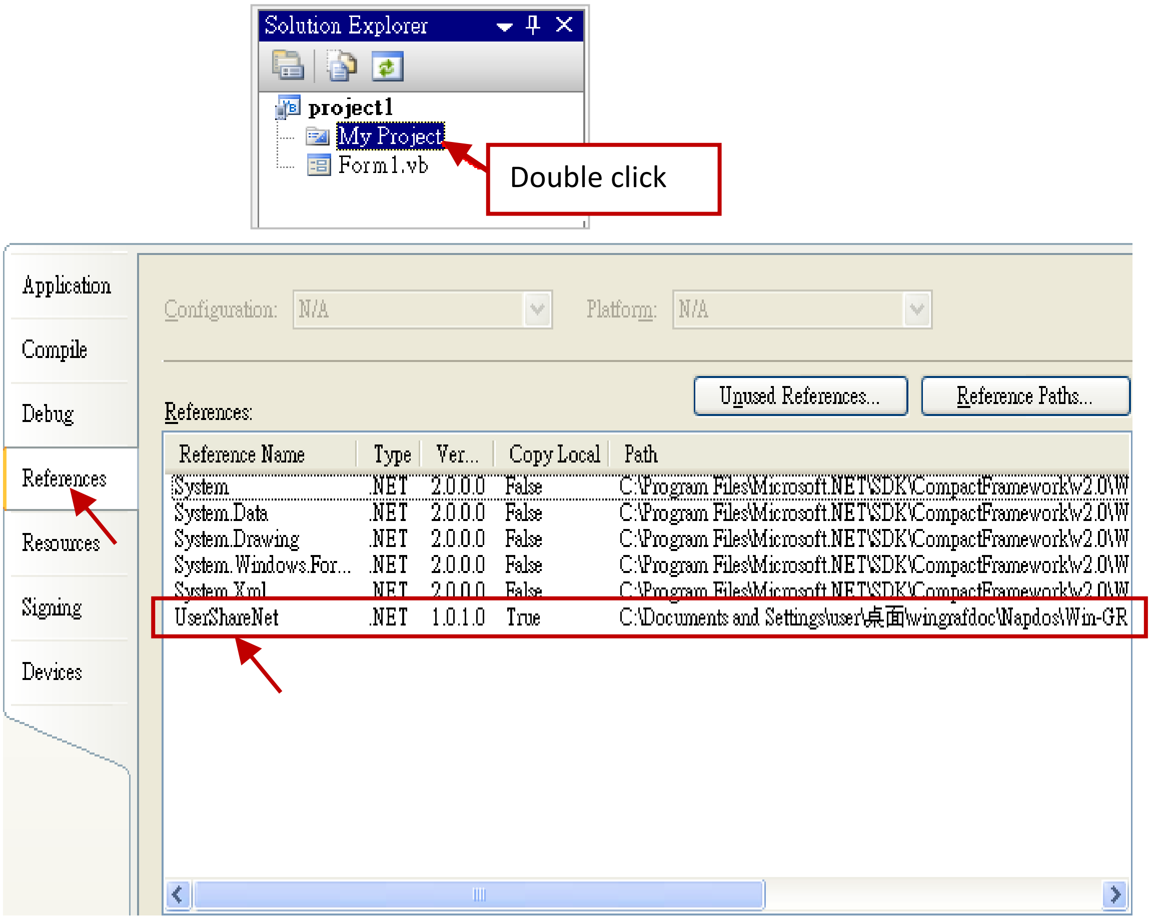
3. Click the “Browse” tab and select the “UserShareNet.dll” from your project location.



The current project is located in "C:\project1".

Note: You may copy the “UserShareNet.dll” from the CD-ROM to your current project path first. Then add it to the project reference.

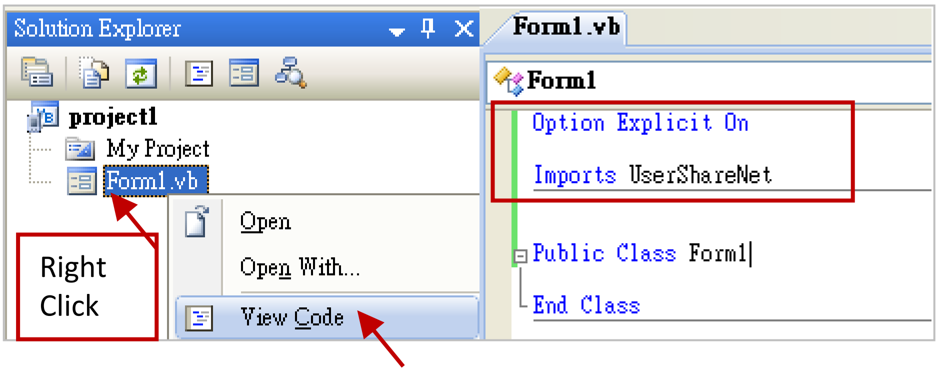
4. When "UserShareNet.dll" is added, please double click on "My Project" to check if the "UserShareNet.dll" is well added.



References:

Reference Name	Type	Ver...	Copy Local	Path
System	.NET	2.0.0.0	False	C:\Program Files\Microsoft.NET\SDK\CompactFramework\v2.0\W
System.Data	.NET	2.0.0.0	False	C:\Program Files\Microsoft.NET\SDK\CompactFramework\v2.0\W
System.Drawing	.NET	2.0.0.0	False	C:\Program Files\Microsoft.NET\SDK\CompactFramework\v2.0\W
System.Windows.For...	.NET	2.0.0.0	False	C:\Program Files\Microsoft.NET\SDK\CompactFramework\v2.0\W
System.Xml	.NET	2.0.0.0	False	C:\Program Files\Microsoft.NET\SDK\CompactFramework\v2.0\W
UserShareNet	.NET	1.0.1.0	True	C:\Documents and Settings\user\桌面\wingrafdoc\Napdos\Win-GR

5. Right-click on the "Form1.vb" and select "View Code" from the pop-up. Move cursor to top and insert the "Option Explicit On" and "Imports UserShareNet" in the first two statements.



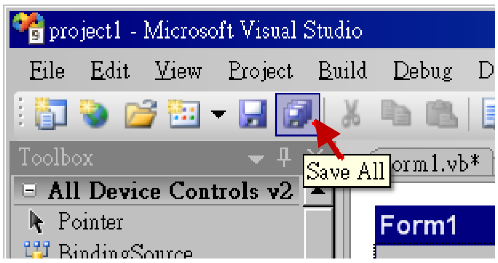
```
Option Explicit On

Imports UserShareNet


Public Class Form1

End Class
```

Then you can design all required objects and actions inside your VB Forms.
(Refer the Chapter 13.5 for more information about using functions in the "UserShareNet.dll".)

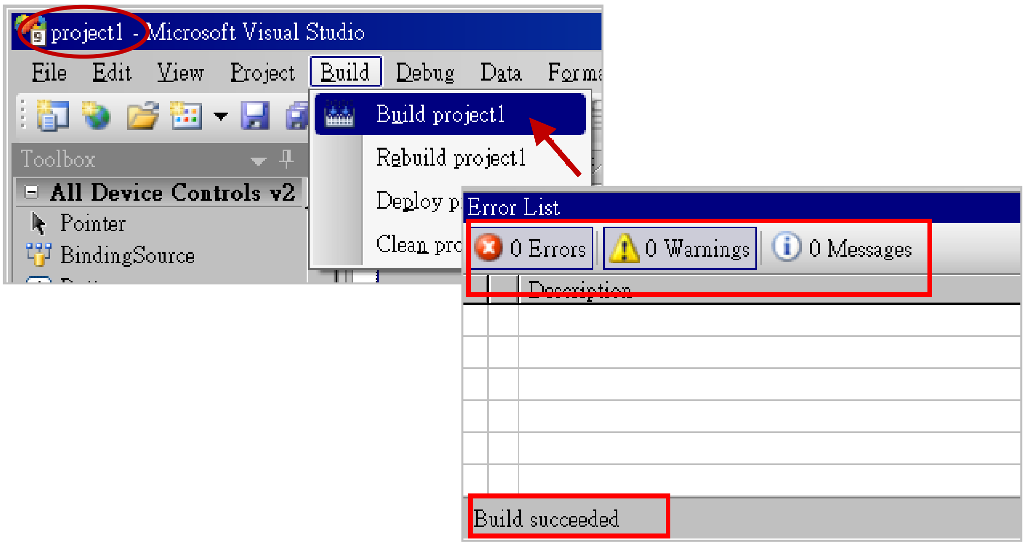
13.4 Compiling the Application

When you have finished writing a program, you can build(compile) an application by the following steps.

1. Remember to save at any time for safety.


project1 - Microsoft Visual Studio
File Edit View Project Build Debug D
Toolbox
Save All
orm1.vb*
All Device Controls v2
Pointer
BindingSource
Form1

2. Then compile (Build) the project. The result is listed in the "Error List" windows at the bottom.



3. You can find the execution file in

<Your VB.net Project folder> \bin\Release\ **<project_name>.exe**

Please copy this execution file to the WP-8xx8's \System_Disk\Win-GRAF\ path to run it.

Note:

User may copy the VB.net execution file to other path to run it but there should contain at least two DLL files with it or it cannot run correctly.

For ex, the project1.exe can run in the \Micro_SD\ path if there is three files in it. The "project1.exe" , "UserShareNet.dll" and "Quicker.dll" . (The "UserShareNet.dll" and "Quicker.dll" can be copied from the Win-GRAF PAC's "\System_disk\Win-GRAF\" path)

13.5 UserShareNet.dll

This section we will focus on the description of the application example of UserShareNet.dll functions. There are some functions that can be used to Read/Write data from/to the Win-GRAF soft-logic. The functions of UserShareNet.dll can be divided into as listed below:

1. R/W Boolean
2. R/W 8-bit Integer
3. R/W 16-bit Integer
4. R/W 32-bit Integer
5. R/W 64-bit Integer
6. R/W 32-bit Float
7. R/W 64-bit Float
8. R/W String

※ **Refer the "Appendix A" to get familiar with the definition of Win-GRAF variables.**

13.5.1 R/W Boolean Functions

■ **Set_BOOL**

Description:

This function is to set a value to a Win-GRAF Boolean variable.

Syntax:

```
UserShare.Set_BOOL ( iUserAddress  As System.UInt16 , ByVal iStatus As byte) as Byte
```

Parameter:

iUserAddress : Address of the Variable (1 to 8192)

iStatus : Set the status. For instance, iStatus = 1 for True, iStatus = 0 for False

Example:

```
'Set the Win-GRAF BOOL variable with address 1 to True.
UserShare.Set_BOOL(Convert.ToUInt16(1), 1)
```

Demo program :

CD-ROM: \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb01

■ Get_BOOL

Description:

This function is to get the value from a Win-GRAF BOOL variable.

Syntax:

```
UserShare.Get_BOOL ( iUserAddress As System.UInt16 , ByRef iStatus As byte)
```

Parameter:

iUserAddress :	Address of the Variable (1 to 8192)
iStatus :	Get the variable value , iStatus = 1 for True, iStatus = 0 for False

範例:

```
‘Get the value of Win-GRAF BOOL variable with address 1.
Dim iStatus As Byte
UserShare.Get_BOOL(Convert.ToUInt16(1), iStatus)
```

Demo Program :

CD-ROM: \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb01

13.5.2 Integer R/W Functions

■ **Set_SINT** ■ **Set_INT** ■ **Set_DINT** ■ **Set_LINT**

Description:

These functions are to set 8-bit Integer, 16-bit Integer, 32-bit integer & 64-bit Integer value to Win-GRAF integer variables.

Syntax:

```
UserShare.Set_SINT (ByVal iUserAddress As System.UInt16 , ByVal iStatus As SByte) As Byte

UserShare.Set_INT (ByVal iUserAddress As System.UInt16 , ByVal iStatus As Short) As Byte

UserShare.Set_DINT (ByVal iUserAddress As System.UInt16 , ByVal iStatus As Integer) As Byte

UserShare.Set_LINT (ByVal iUserAddress As System.UInt16 , ByVal iStatus As long) As Byte
```

Parameter:

iUserAddress : Address of Variable. (1 to 8192)

iStatus : the value of 8-bit Integer, 16-bit Integer, 32-bit Integer or 64-bit Integer.

Example:

```
'Set a 32-bit integer value "1234567" to the Win-GRAF DINT variable with Address "1".
UserShare.Set_DINT(Convert.ToUInt16(1), Convert.ToInt32(1234567) )

'Set a 16-bit integer value "-1234" to the Win-GRAF INT variable with Address "2".
UserShare.Set_INT(Convert.ToUInt16(3), Convert.ToInt16(-1234) )

'Set a 64-bit Integer value "123456789012345" to the Win-GRAF LINT variable with Address "3".
UserShare.Set_LINT(Convert.ToUInt16(3), Convert.ToInt64(123456789012345) )

 'Set a 8-bit Integer value "125" to the Win-GRAF SINT variable with Address "4".
UserShare.Set_SINT(Convert.ToUInt16(3), Convert.ToSByte(125) )
```

Demo Program :

CD-ROM :

1. \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb02 for R/W analog I/O
2. \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb03 for R/W internal long integer, Timer and Real (floating-point) values.

■ Get_SINT ■ Get_INT ■ Get_DINT ■ Get_LINT

Description:

These functions are to get 8-bit integer, 16-bit integer, 32-bit integer & 64-bit integer value from Win-GRAF integer variables.

Syntax:

UserShare. Get_SINT (ByVal *iUserAddress* As System.UInt16 , ByRef *iStatus* As SByte) As Byte

UserShare. Get_INT (ByVal *iUserAddress* As System.UInt16 , ByRef *iStatus* As Short) As Byte

UserShare.Get_DINT (ByVal *iUserAddress* As System.UInt16 , ByRef *iStatus* As Integer) As Byte

UserShare. Get_LINT (ByVal *iUserAddress* As System.UInt16 , ByRef *iStatus* As long) As Byte

Parameter:

iUserAddress : Address of Variable (1 to 8192)

iStatus : Get the 8-bit integer, 16-bit integer, 32bit-integer or 64-bit integer value.

Example:

```
Dim Dlong_val As Int64
Dim short_val As Int16
Dim long_val As Int32
Dim Sbyte_val as byte

‘Get 64-bit integer value from the Win-GRAF LINT variable with address “7”.
UserShare.Get_LINT(Convert.ToUInt16(7), Dlong_val)

‘Get 32-bit integer value from the Win-GRAF DINT variable with address “8”.
UserShare.Get_DINT(Convert.ToUInt16(8), long_val)

‘Get 16-bit integer value from the Win-GRAF INT variable with address “9”.
UserShare.Get_INT(Convert.ToUInt16(9), short_val)

‘Get 8-bit integer value from the Win-GRAF SINT variable with address “10”.
UserShare.Get_SINT(Convert.ToUInt16(9), sbyte_val)
```

Demo program:

CD-ROM:

1. R/W analog I/O :
   \napdos\Win-GRAF\demo-project\vb.net_2008_demo \demo_vb02
2. R/W internal long integer, Timer and Real (floating-point) values :
   \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb03

13.5.3 R/W Real variable Functions

■ **Get_REAL** ■ **Get_LREAL**

Description:

These functions are to get 32-bit REAL and 64-bit double from the Win-GRAF REAL/LREAL variable.

Syntax:

UserShare. Get_REAL (ByVal *iUserAddress* As System.UInt16 , ByRef *iStatus* As Single) As Byte

UserShare. Get_LREAL(ByVal *iUserAddress* As System.UInt16 , ByRef *iStatus* As Double) As Byte

parameter:

iUserAddress : Address of Variable (1 to 8192)

iStatus : Get the 32-bit REAL or 64-bit double value.

example:

```
Dim float_val As Single
Dim double_val As Double

'Get 64-bit double value from the Win-GRAF LREAL variable with address "7".
UserShare.Get_LREAL(Convert.ToUInt16(7), double_val)

'Get 32-bit REAL value from the Win-GRAF REAL variable with Address "8".
UserShare.Get_REAL(Convert.ToUInt16(8), float_val)
```

Demo program:

CD-ROM:

1. R/W analog I/O :
   \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb02
2. R/W internal long integer, Timer and Real (floating-point) values :
   \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb03

■ Set_REAL ■ Set_LREAL

Description:

These functions are to set 32-bit REAL and 64-bit double value to the Win-GRAF REAL/LREAL variable.

Syntax:

UserShare. Set_REAL (ByVal *iUserAddress* As System.UInt16, ByVal *iStatus* As Single) As Byte

UserShare. Set_LREAL(ByVal *iUserAddress* As System.UInt16, ByVal *iStatus* As Double) As Byte

Parameter:

iUserAddress : Address of Variable. (1 to 8192)
iStatus : Set the 32-bit REAL or 64-bit double.

Example:

```
'Set a 64-bit double value "11234.234567" to the Win-GRAF LREAL variable with address "1".
UserShare.Set_LREAL(Convert.ToUInt16(7),Convert.ToDouble(11234.234567))

'Set a 32-bit REAL value "123.12" to the Win-GRAF REAL variable with Address "8".
UserShare.Set_REAL(Convert.ToUInt16(8), Convert.ToSingle (123.12))
```

Demo program:

CD-ROM:

1. R/W analog I/O : \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb02
2. R/W internal long integer, Timer and Real (floating-point) values : \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb03

13.5.4 R/W String variable Functions

■ Get_STRING

Description:

This function is to get a Win-GRAF String variable.

Syntax:

UserShare. Get_STRING (ByVal *iUserAddress* As System.UInt16, ByVal *msg()* As Byte) As Byte

Parameter:

iUserAddress : Address of Variable (1 to 1024)

msg() : Get the string value.

Example:

```
Dim str_val As String
Dim msg() As Byte

'Get String value of the Win-GRAF String variable with address "7".
UserShare.Get_STRING(Convert.ToUInt16(7),msg )
str_val= byte_array_to_unicode(msg)

 Private Function byte_array_to_unicode(ByVal buf() As Byte) As String
       Dim tmpmsg As String
       If buf.Length > 255 Then
          Return Nothing
       End If
tmpmsg = System.Text.Encoding.GetEncoding("UTF-8").GetString(buf, 0,  buf.Length)

       Return tmpmsg
   End Function
```

Demo program:

CD-ROM:

1. R/W String variable:
   \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb04

■ Set_STRING

Description:

This functions is to set String value to the Win-GRAF String variable.

Syntax:

```
UserShare. Set_STRING (ByVal iUserAddress As System.UInt16, ByVal msg() As Byte) As Byte
```

Parameter:

iUserAddress : Address of Variable. (1 to 1024)

msg() : the string value.

Example:

```
Dim str_val As String=”Hello World”
Dim msg() As Byte

msg= unicode_to_byte_array(str_val)

‘Set a string value “Hello World” to the Win-GRAF String variable with Address “7”.
UserShare.Set_STRING(Convert.ToUInt16(7),msg )

‘Convert String to byte array.
Private Function unicode_to_byte_array(ByVal msg As String) As Byte()
    Dim tmpbuf() As Byte
    If msg.Length > 255 Then
        Return Nothing
    End If
    tmpbuf = System.Text.Encoding.GetEncoding("UTF-8").GetBytes(msg)
    Return tmpbuf
End_Function
```

Demo program:

CD-ROM:

1. R/W String variable :
   \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb04

13.5.5 How to use VB.NET R/W to Win-GRAF String variable

Before .NET program write to Win-GRAF String variable. The String-type has to convert to byte array. If you need to read Win-GRAF String variable. Then you have to convert byte array to String. There is an VB.NET example to show how to convert each other.

(Encode :UTF-8) :

Convert String to byte array

```
  Private Function unicode_to_byte_array(ByVal msg As String) As Byte()
     Dim tmpbuf() As Byte
     If msg.Length > 255 Then
        Return Nothing
     End If

     tmpbuf = System.Text.Encoding.GetEncoding("UTF-8").GetBytes(msg)
     Return tmpbuf
  End Function
```

Convert byte array to string

```
  Private Function byte_array_to_unicode(ByVal buf() As Byte) As String
     Dim tmpmsg As String
     If buf.Length > 255 Then
        Return Nothing
     End If
tmpmsg = System.Text.Encoding.GetEncoding("UTF-8").GetString(buf, 0, buf.Length)

     Return tmpmsg
  End Function
```

Chapter 14 C# .net 2008 Program Running In WP-8xx8 Access To Win-GRAF Variables

This chapter lists the procedure for creating the first demo program by Visual Studio .NET 2008 development tool. There is some sample programs in the Wp-8xx8 CD-ROM.

C# demo:

CD-ROM : \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\
demo_CSharp01 : Digital I/O demo with one I-87055W in slot 0 of the Wp-8xx8.
demo_CSharp02 : Analog I/O demo with one I-87024W in slot 1 and one I-8017HW in slot 2.
demo_CSharp03 : Read / Write Win-GRAF internal integers, timers and real variables. (No I/O)
demo_CSharp04 : Read/Write Win-GRAF internal String variables. (No I/O)

Win-GRAF demo:

CD-Rom : \napdos\Win-GRAF\demo-project\
"demo_vb01.zip", "demo_vb02.zip", "demo_vb03.zip", "demo_vb04.zip"

14.1 Add an Existing Win-GRAF project from a ZIP

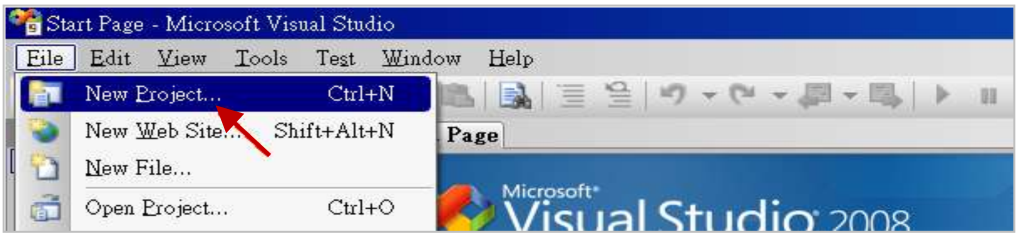
Please refer the Chapter 13.1

14.2 Publishing the Win-GRAF Variable for .NET

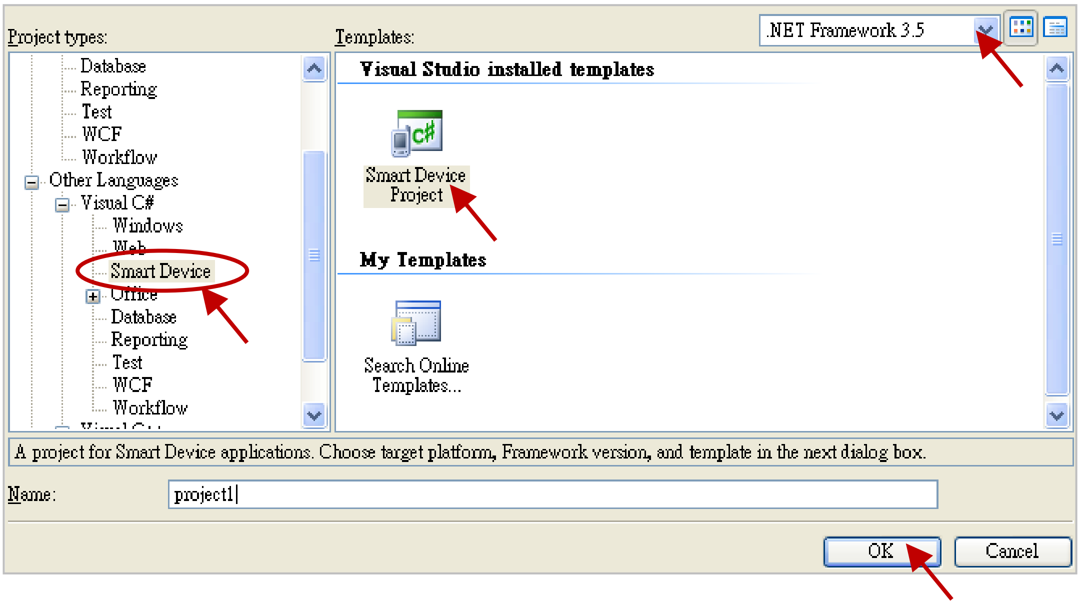
Please refer the Chapter 13.2

14.3 Create a New C# Project

1. First, users need to open Microsoft Visual Studio .NET 2008 software. And then in the menu of “File”, please run the “New Project” .


Start Page - Microsoft Visual Studio
File Edit View Tools Test Window Help
New Project... Ctrl+N
New Web Site... Shift+Alt+N
New File...
Open Project... Ctrl+O
Page
Microsoft
Visual Studio 2008

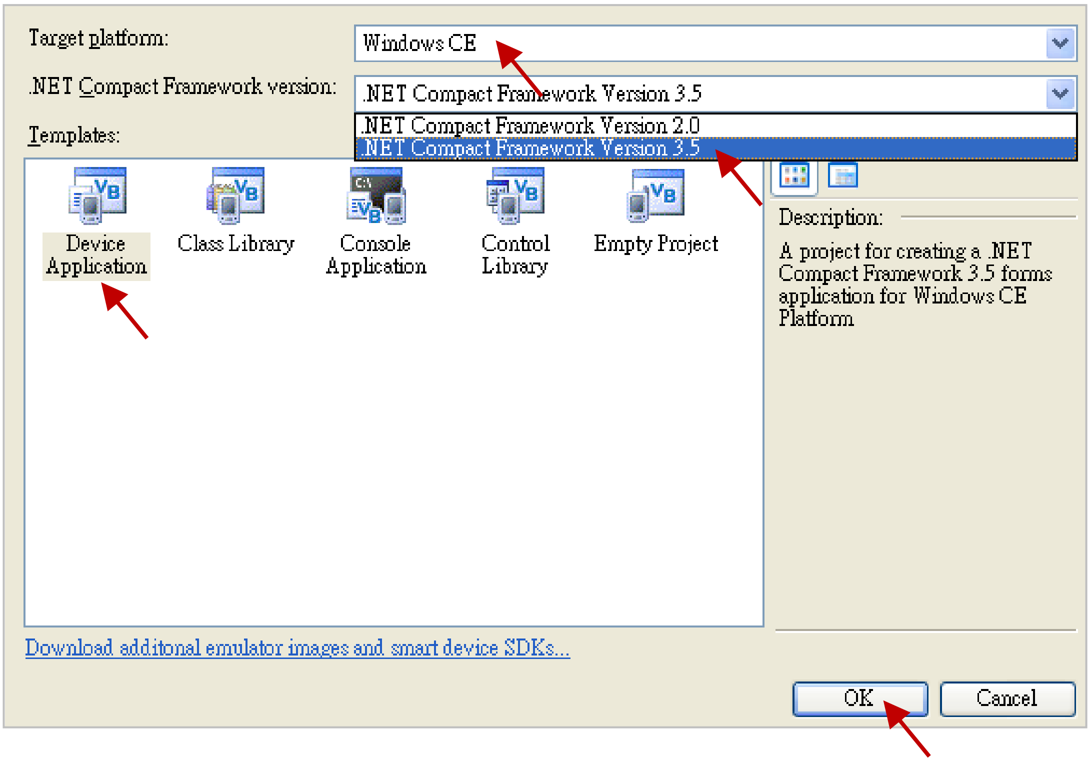
2. Check the “Smart Device” on the left, then selecting the “.NET framework 3.5” and “Smart Device Project”. Then entering a proper project name and the last click on “OK”.



A project for Smart Device applications. Choose target platform, Framework version, and template in the next dialog box.

Name: project1

3. Select the "**Device Application**" and "**Windows CE**" and “**.NET Compact Framework Version 3.5**”, then click on “OK”.



Target platform: Windows CE

.NET Compact Framework version: .NET Compact Framework Version 3.5

.NET Compact Framework Version 2.0
.NET Compact Framework Version 3.5
Templates:
Device Application
Class Library
Console Application
Control Library
Empty Project
Download additonal emulator images and smart device SDKs...
OK
Cancel

Description:

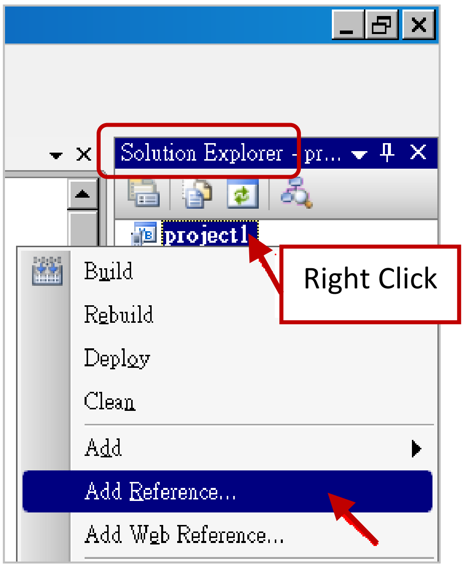
A project for creating a .NET Compact Framework 3.5 forms application for Windows CE Platform

14.3.1 Add C# Project Reference

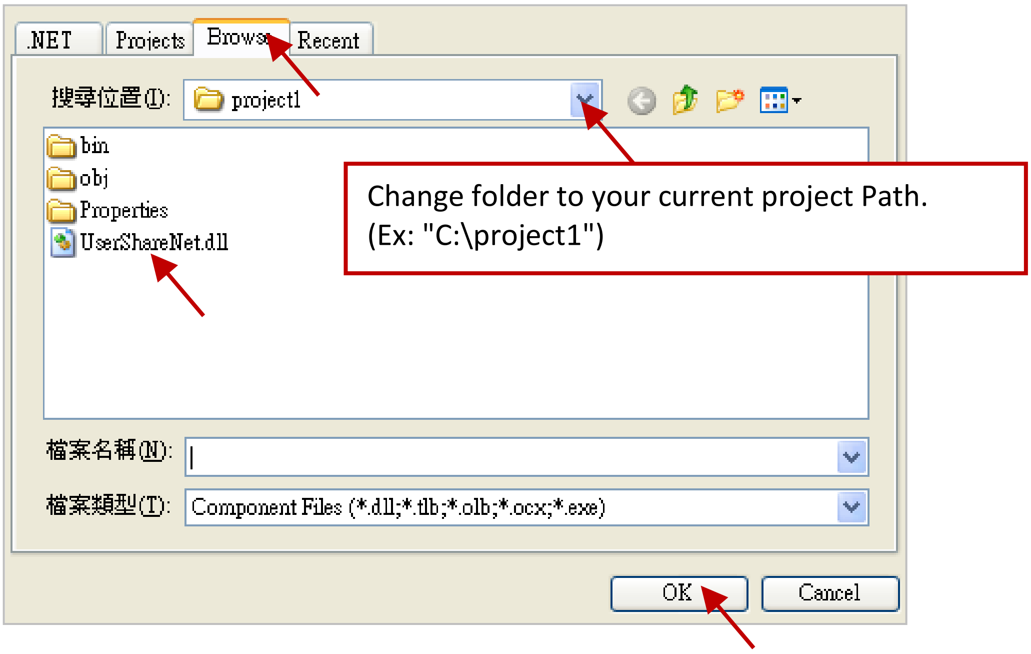
The “UserShareNet” library contains all modules’ functions. Before you use the “UserShare” keyword in the program, you must add the “UserShareNet.dll” into the reference list of your application.

1. Copy the “UserShareNet.DLL” from WP-8xx8 CD-ROM: **\napdos\Win-GRAF\WP-8xx8\CSharp.net_2008_demo\demo_CSharp01\** to your project folder(ex: C:\project1\)

2. Right click on the Project name on the right hand side , then select “Add Reference …”



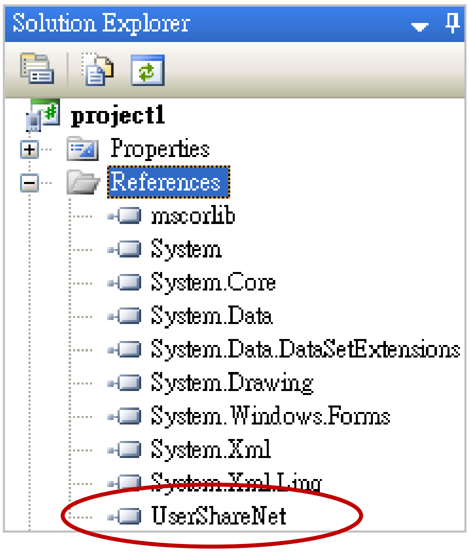
3. Click the “**Browse**” button. Select the “UserShareNet.dll” from your project location.



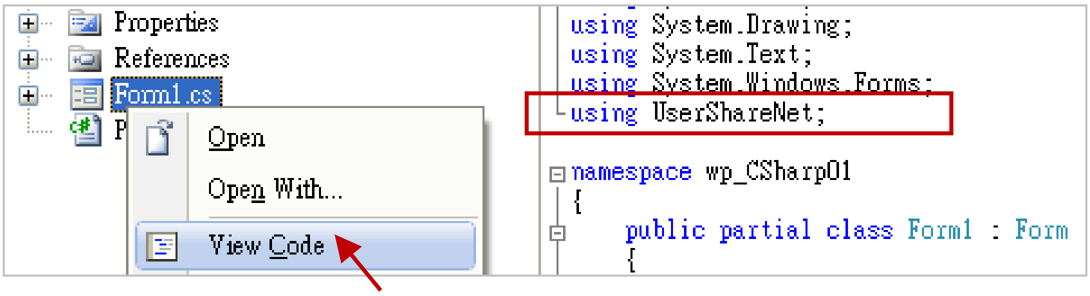
Change folder to your current project Path.
(Ex: "C:\project1")

Note: **You may copy the “UserShareNet.dll” from the CD-ROM to your current project path first. Then add it to the project reference.**

4. When “UserShareNet.dll” are added, you can see them in the solution explorer as below.



5. Right-click on the “**Form1.cs**” and select “**View Code**” from the pop-up. Move cursor to top and insert the “**using UserShareNet;**” in the first statements.



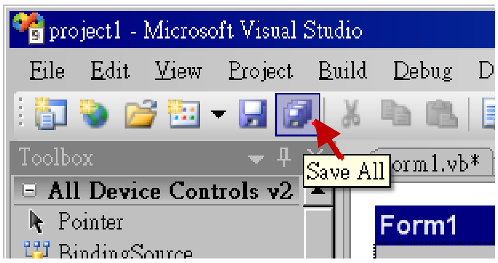
```
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using UserShareNet;

namespace wp_CSharp01
{
    public partial class Form1 : Form
    {
```

Then you can design all required objects and actions inside your C# Forms.
(Refer the Section 14.5 for more information about using functions in the “UserShareNet.dll”.)

14.4 Compiling the Application Program

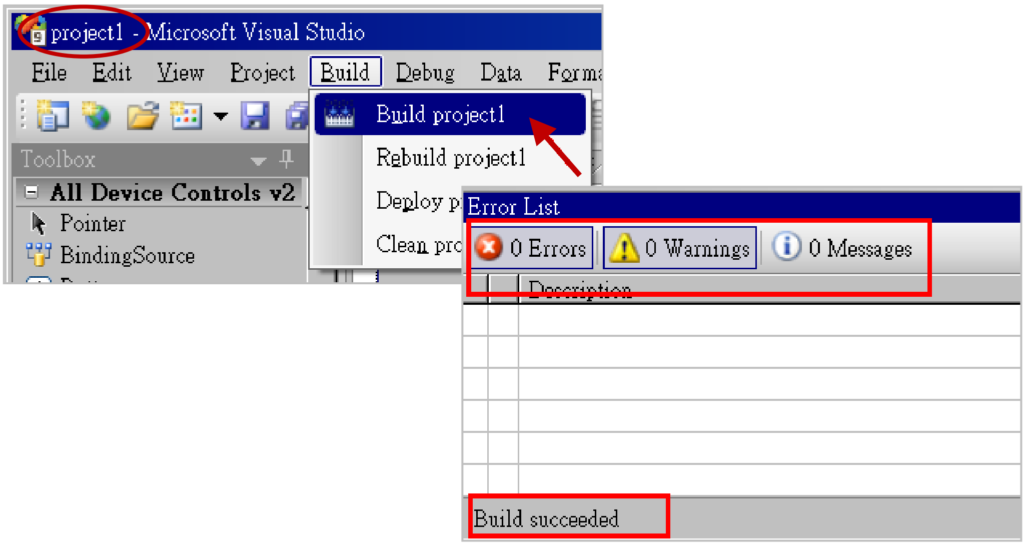
When you have finished writing a program, you can build(compile) an application by the following steps.

1. Remember to save at any time for safety.



project1 - Microsoft Visual Studio
File Edit View Project Build Debug D
Toolbox
All Device Controls v2
Pointer
BindingSource
Save All
orm1.vb*
Form1

2. Then compile (Build) the project . The result is listed in the "Error List" windows at the bottom .



3. You can find the execution file in

```
<Your C# .net Project folder> \bin\Release\ <project_name>.exe
```

Please copy this execution file to the Wp-8xx8's \System_Disk\Win-GRAF\ path to run it.

Note:

User may copy the C#.net execution file to other path to run it but there should contain at least two DLL files with it or it can not run correctly. For ex, the project1.exe can run in the \Micro_SD\ path if there is three files in it. The "project1.exe", "UserShareNet.dll" and, "Quicker.dll" .

(The "UserShareNet.dll" and "Quicker.dll" can be copied from the Win-GRAF PAC's "\System_disk\Win-GRAF\" path)

14.5 UserShareNet.DLL

This section we will focus on the description of the application example of UserShareNet.DLL functions. There are some functions that can be used to Read/Write data from/to the Win-GRAF variable. The functions of UserShareNet.DLL can be divided into as listed below

1. R/W Boolean
2. R/W 8-bit Integer
3. R/W 16-bit Integer
4. R/W 32-bit Integer
5. R/W 64-bit Integer
6. R/W 32-bit Float
7. R/W 64-bit Float
8. R/W 32-bit String

※ Refer the "Appendix A" to get familiar with the definition of Win-GRAF variables.

14.5.1 R/W Boolean Functions

■ **Set_BOOL**

Description :

This function is to set a value to a Win-GRAF Boolean variable.

Syntax:

```
UserShare.Set_BOOL(ushort iUserAddress, byte iStatus)
```

Parameter:

iUserAddress : Address of Variable (1 to 8192)

iStatus : Set the status. For instance, iStatus = 1 for True, iStatus = 0 for False

Example:

```
// Set the Win-GRAF BOOL variable with address 1 to True.
 UserShare.Set_BOOL(Convert.ToUInt16(1), 1);
```

Demo program :

CD-ROM : \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp01

■ Get_BOOL

Description :

This function is to get the value from a Win-GRAF BOOL variable.

Syntax:

```
UserShare.Get_BOOL(ushort iUserAddress, out byte iStatus)
```

Parameter:

iUserAddress : Address of Variable. (1 to 8191)

iStatus : Get the variable status , iStatus = 1 for True, iStatus = 0 for False.

Example::

```
Byte iStatus=0;
// Get the value of Win-GRAF BOOL variable with address 1.
UserShare.Get_BOOL(Convert.ToUInt16(1),out iStatus);
```

Demo program :

CD-ROM: \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp01

14.5.2 R/W Integer Functions

■ **Set_SINT** ■ **Set_INT** ■ **Set_DINT** ■ **Set_LINT**

Description:

These functions are to set 8-bit Integer, 16-bit Integer, 32-bit integer & 64-bit Integer value to Win-GRAF integer variables.

Syntax:

```
UserShare.Set_SINT(ushort iUserAddress , sbyte iStatus)

UserShare.Set_INT(ushort iUserAddress , short iStatus)

UserShare.Set_DINT(ushort iUserAddress, int iStatus)

UserShare.Set_LINT(ushort iUserAddress, long iStatus)
```

Parameter:

iUserAddress : Address of Variable. (1 to 8192)

iStatus : Set the 8-bit Integer, 16-bit Integer, 32-bit Integer or 64-bit Integer.

Example:

```
// Set a 32-bit integer value "1234567" to the Win-GRAF DINT variable with Address "1".
int temp1=1234567;
UserShare.Set_DINT(Convert.ToUInt16(1), temp );

// Set a 16-bit integer value "-1234" to the Win-GRAF INT variable with Address "2".
short temp2= -1234;
UserShare.Set_INT(Convert.ToUInt16(2), temp2 );

// Set a 64-bit Integer value "123456789012345" to the Win-GRAF LINT variable with Address "3".
long temp3=123456789012345;
UserShare.Set_LINT(Convert.ToUInt16(3),  temp3 );


// Set a 8-bit Integer value "125" to the Win-GRAF SINT variable with Address "4".
Sbyte temp4=125;
UserShare.Set_SINT(Convert.ToUInt16(4),  temp4 );
```

Demo program:

CD-ROM:

1. R/W analog I/O:
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp02
2. R/W internal Boolean ,long integer, Timer and Real (floating-point) values :
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp03

■ Get_SINT ■ Get_INT ■ Get_DINT ■ Get_LINT

Description:

These functions are to get 8-bit integer, 16-bit integer, 32-bit integer & 64-bit integer value from Win-GRAF integer variables.

Syntax:

```
UserShare.Get_SINT(ushort iUserAddress, out sbyte iStatus)

UserShare.Get_INT(ushort iUserAddress, out short iStatus)

UserShare.Get_DINT(ushort iUserAddress, out int iStatus)

UserShare.Get_LINT(ushort iUserAddress, out long iStatus)
```

Parameter:

iUserAddress : Address of Variable (1 to 8192)

iStatus : Get the value of Win-GRAF integer variables.

example:

```
Int64 Dlong_val;
Int16 short_val;
Int32 long_val ;
sbyte sbyte_val;

// Get 64-bit integer value from the Win-GRAF LINT variable with address “7”.
UserShare.Get_LINT(Convert.ToUInt16(7),out Dlong_val);

// Get 32-bit integer value from the Win-GRAF DINT variable with address “8”.
UserShare.Get_DINT(Convert.ToUInt16(8),out long_val);

// Get 16-bit integer value from the Win-GRAF INT variable with address “9”.
UserShare.Get_INT(Convert.ToUInt16(9),out short_val);

// Get 8-bit integer value from the Win-GRAF SINT variable with address “10”.
UserShare.Get_SINT(Convert.ToUInt16(9),out sbyte_val)
```

Demo program:

CD-Rom:

1. R/W analog I/O:
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp02
2. R/W internal Boolean ,long integer, Timer and Real (floating-point) values:
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp03

14.5.3 R/W Real variable Functions

■ **Get_REAL** ■ **Get_LREAL**

Description:

These functions are to get 32-bit REAL and 64-bit double from the Win-GRAF.

Syntax:

UserShare. Get_REAL (System.UInt16 *iUserAddress*, out float *iStatus*)

UserShare. Get_LREAL(ByVal *iUserAddress* As System.UInt16 , out Double *iStatus*)

Parameter:

iUserAddress : Address of Variable (1 to 8192)
iStatus : Get the 32-bit REAL or 64-bit double value.

Example:

```
float float_val;
double double_val;

// Get 64-bit double value from the Win-GRAF LREAL variable with address “7”.
UserShare.Get_LREAL(Convert.ToUInt16(7),out double_val);

// Get 32-bit REAL value from the Win-GRAF REAL variable with Address “8”.
UserShare.Get_REAL(Convert.ToUInt16(8),out float_val);
```

Demo program:

CD-Rom:

1. R/W analog I/O:
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp02
2. R/W internal long integer, Timer and Real (floating-point) values :
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_ demo_CSharp03

■ Set_REAL ■ Set_LREAL

Description:

These functions are to set 32-bit REAL and 64-bit double value to the Win-GRAF REAL/LREAL variable.

Syntax:

> **UserShare. Set_REAL (ushort *iUserAddress* , float *iStatus*)**
>
> **UserShare. Set_LREAL(ushort *iUserAddress* , Double *iStatus*)**

Parameter:

iUserAddress : Address of Variable. (1 to 8192)

iStatus : Set the 32-bit REAL or 64-bit double.

Example:

```
// Set a 64-bit double value “11234.234567” to the Win-GRAF LREAL variable with address “7”
UserShare.Set_LREAL(Convert.ToUInt16(7),Convert.ToDouble(11234.234567));

// Set a 32-bit REAL value “123.12” to the Win-GRAF REAL variable with Address “2”.
UserShare.Set_REAL(Convert.ToUInt16(8), Convert.ToSingle (123.12));
```

Demo program :

CD-ROM:

1. R/W analog I/O:
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp02
2. R/W internal long integer, Timer and Real (floating-point) values :
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp03

14.5.4 R/W String variable Functions

■ Set_STRING

Description:

This function is to get a Win-GRAF String variable.

Syntax:

```
UserShare.Set_STRING (ushort  addr , Byte []  msg)
```

Parameter:

addr : Address of Variable (1 to 1024)

msg[] : Get the string value.

Example:

```
String str_val;
Byte[] msg;

// Get String value of the Win-GRAF String variable with address “7”.
msg= unicode_to_byte_array(str_val);
UserShare.Set_STRING(Convert.ToUInt16(7),msg );

//Convert String to byte array.
private byte[] unicode_to_byte_array(string msg)
    {
        byte[] tmpbuf;
        if (msg.Length > 255)
            return null;

        tmpbuf = Encoding.GetEncoding("UTF-8").GetBytes(msg);
        return tmpbuf;
    }
```

Demo program:

CD-Rom:

1. R/W String variable :
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp04

■ Get_STRING

Description:

This functions is to set String value to the Win-GRAF String variable.

Syntax :

```
UserShare.Set_STRING (ushort  addr , Byte []  msg)
```

Parameter:

addr : Address of Variable. (1 to 1024)

msg[] : Set the string value.

Example:

```
String str_val= “Hello World”;
Byte[] msg;

// Set a string value “Hello World” to the Win-GRAF String variable with Address “7”.
UserShare.Get_STRING(Convert.ToUInt16(7),msg );
str_val= byte_array_to_unicode(msg);

//Convert byte array to String
private string byte_array_to_unicode(byte[] buf)
 {
        string tmpmsg;
        if (buf.Length > 255)
           return null;

        tmpmsg = Encoding.GetEncoding("UTF-8").GetString(buf, 0, buf.Length);
        return tmpmsg;
  }
```

Demo program:

CD-Rom:

1. R/W String variable : \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp04

14.5.5 How to use C# to convert Win-GRAF String variable

Before .NET program write to Win-GRAF String variable. The String-type has to convert to byte array. (According your .NET program Encode. Ex: UTF-8) If you need to read Win-GRAF String variable. Then you have to convert byte array to String. There is an C# example to show how to convert each other.

Example (Encode is UTF-8):

```
//Convert String to byte array
private byte[] unicode_to_byte_array(string msg)
    {
      byte[] tmpbuf;
      if (msg.Length > 255)
        return null;

      tmpbuf = Encoding.GetEncoding("UTF-8").GetBytes(msg);
      return tmpbuf;
    }

//byte array to string
private string byte_array_to_unicode(byte[] buf)
 {
      string tmpmsg;
      if (buf.Length > 255)
        return null;

      tmpmsg = Encoding.GetEncoding("UTF-8").GetString(buf, 0, buf.Length);
      return tmpmsg;
  }
```

Chapter 15 Using Soft-GRAF HMI in the Win-GRAF PAC

This chapter list the way to run Soft-GRAF HMI in the Win-GRAF PAC. The Soft-GRAF is a software which allows user to create colorful HMI. User can easily edit HMI screen by mouse dragging and dropping the object and run both HMI and Win-GRAF softlogic in the same Win-GRAF PAC.

※**Note:** Please refer the ISaGRAF FAQ 146 to get more information about using Soft-GRAF.
ISaGRAF FAQ146:
http://www.icpdas.com/faq/isagraf/146.htm

15.1 Restore the Win-GRAF project

Please copy all demo programs into your PC.
CD-ROM of Win-GRAF PAC: \napdos\Win-GRAF\demo-project\Soft-GRAF-demo\

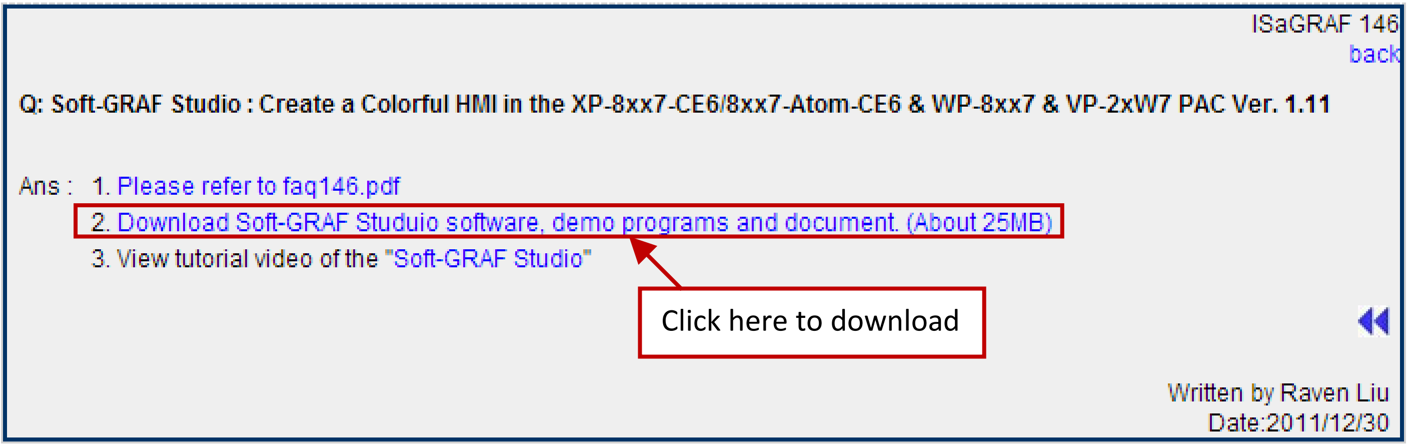
Demo project for both Win-GRAF and Soft-GRAF:

Win-GRAF demo	Soft-GRAF demo	Description
demo_soft-graf01	demo01	Demonstrate Soft-GRAF basic HMI object.
demo_soft-graf02	demo02	
demo_soft-graf03	demo03	
demo_soft-graf04	demo04	
demo_soft-graf05	demo05	Picture and animation picture (.gif).
demo_soft-graf06	demo06	Trend graph and gauge.
demo_soft-graf07	demo07a demo07b	Alarm
demo_soft-graf08	demo08	Data logger

You may restore the “demo_soft_graf01.zip” to your Win-GRAF workbench to get familiar with it. Then download to the Win-GRAF PAC. If you have no idea how to restore the Win-GRAF project. Please refer the Section 13.1.

15.1.1 Install the Soft-GRAF Studio

Please download the “faq146_demo.zip” from the ISaGARF FAQ146 web page.
http://www.icpdas.com/faq/isagraf/146.htm



ISaGRAF 146
back

Q: Soft-GRAF Studio : Create a Colorful HMI in the XP-8xx7-CE6/8xx7-Atom-CE6 & WP-8xx7 & VP-2xW7 PAC Ver. 1.11

Ans : 1. Please refer to faq146.pdf
2. Download Soft-GRAF Studuio software, demo programs and document. (About 25MB)
3. View tutorial video of the "Soft-GRAF Studio"

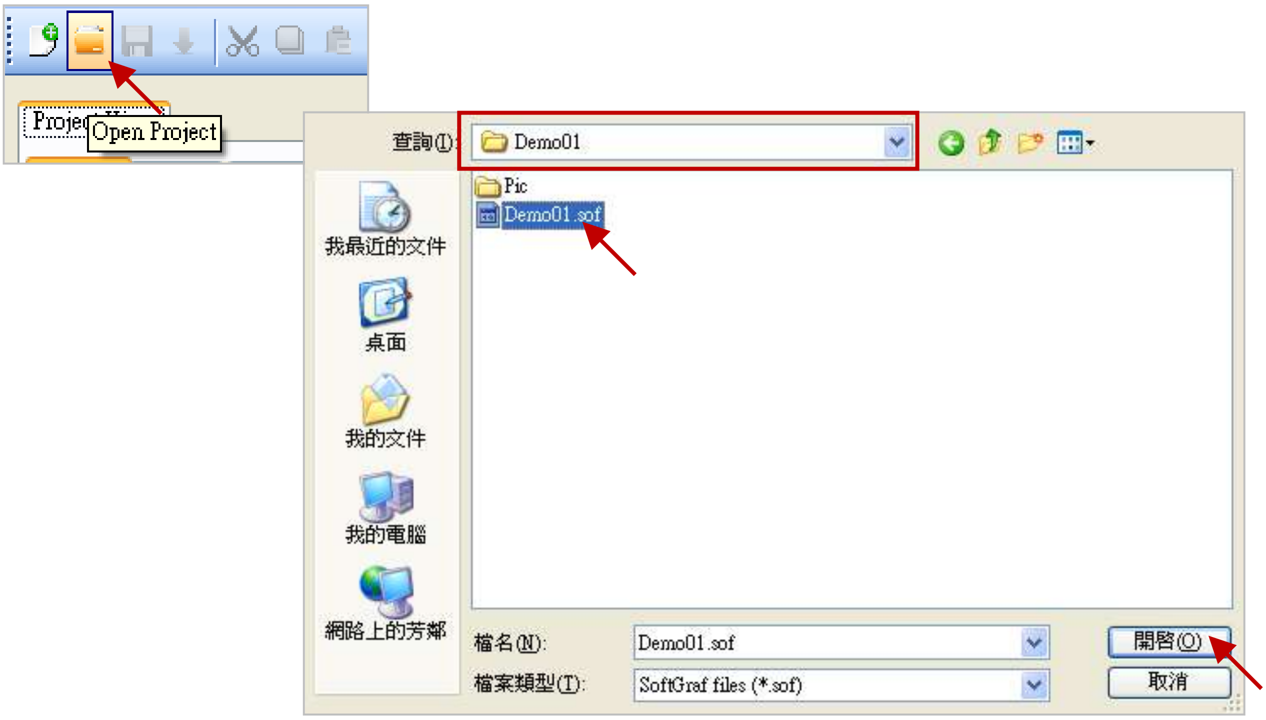
Click here to download

Written by Raven Liu
Date:2011/12/30

After unzip the file, please copy the “Soft-GRAF Studio” folder into “D:\”. Then, it becomes “D:\Soft-GRAF Studio”.

15.1.2 Download the Soft-GRAF HMI project to the Win-GRAF PAC

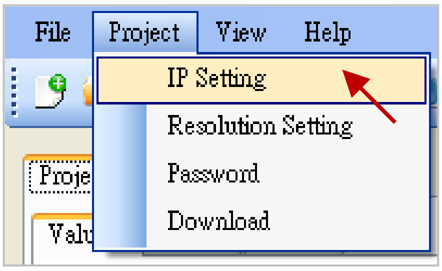
Executes the “Soft-GRAF Studio.exe” from the “D:\Soft-GRAF Studio”. Then click the “open project” on the upper left screen. Open the “demo01.sof”.

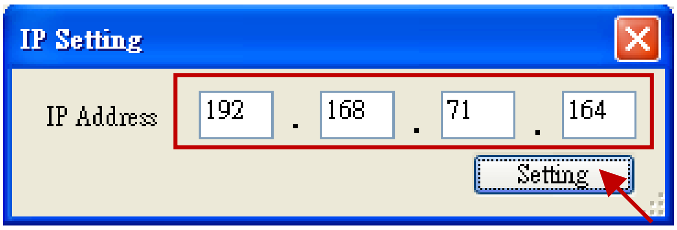


檔名(N): Demo01.sof

檔案類型(T): SoftGraf files (*.sof)

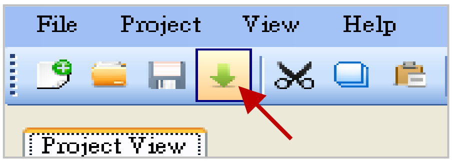
Click “project” -> “IP Setting”. Setup the IP address of the Win-GRAF PAC.





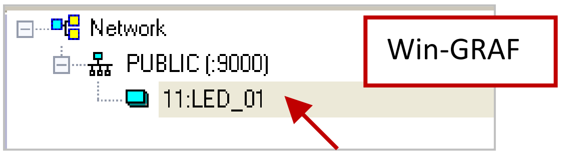
IP Address 192 . 168 . 71 . 164

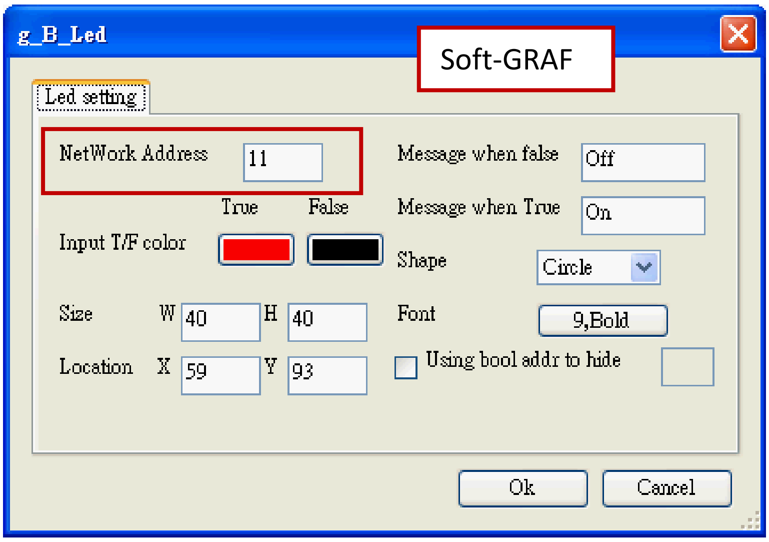
Click “download” to download HMI project to the Win-GRAF PAC. If download success, you can see HMI on the Win-GRAF PAC screen.



15.2 How to develop Soft-GRAF HMI

Before user develops Soft-GRAF HMI, users have to publish Win-GRAF project variables. Then, Soft-GRAF HMI could read/write the Win-GRAF variables. Please refer the Section 13.2 to get information about publishing Win-GRAF variables.





NetWork Address: 11

Message when false: Off

Message when True: On

Input T/F color: True 🟥 False ⬛

Shape: Circle

Size W 40 H 40

Font: 9,Bold

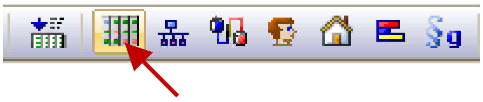
Location X 59 Y 93

□ Using bool addr to hide

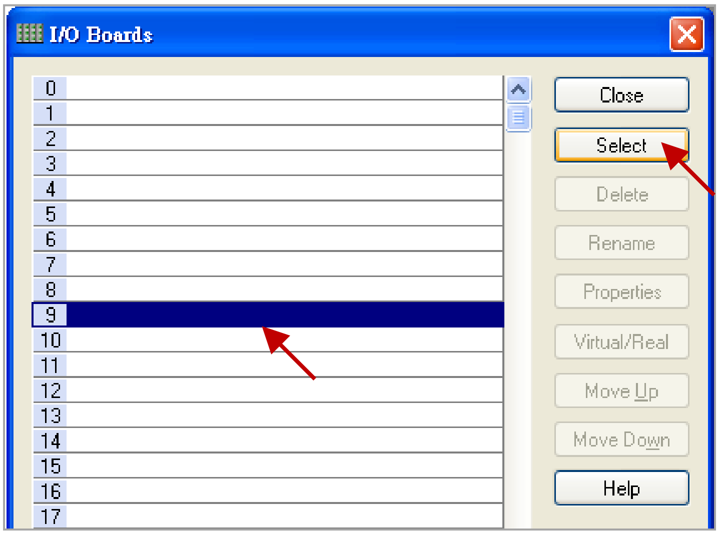
The left example shows a Win-GRAF variable “LED_01” with address “11” is published. If user wish to use Soft-GRAF to read/write this variable. Then user has to set “NetWork Address” to 11 in Soft-GRAF Studio object. (This Soft-GRAF HMI object is “g_B_Led”.)

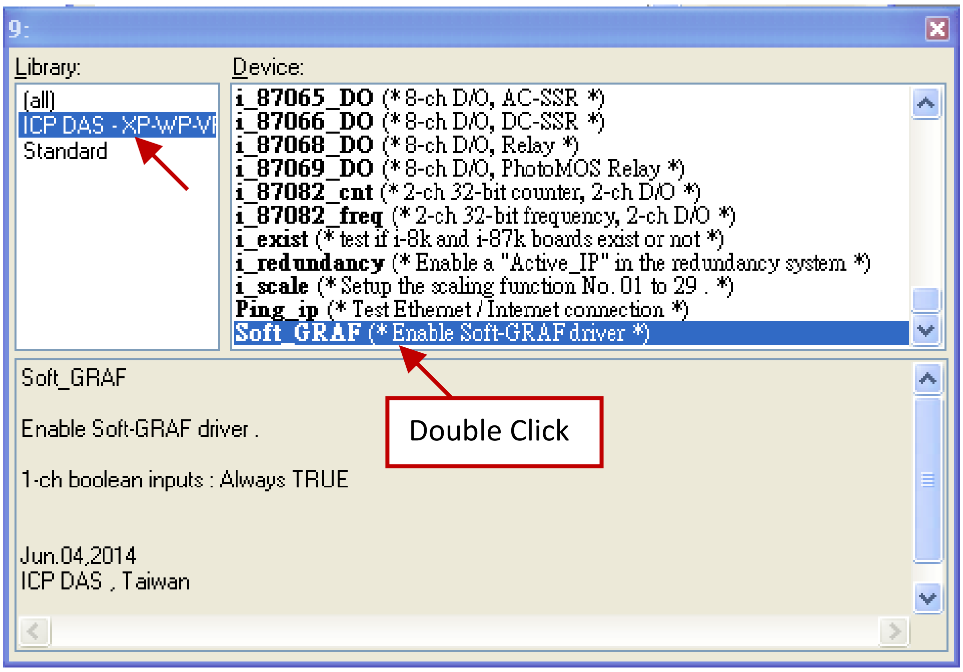
There is another setting in the Win-GRAF workbench before user develops Soft-GRAF HMI. User needs to add “Soft-GRAF” in the Win-GRAF “IO boards”. The following steps demonstrate how to add it.

Click “Open I/Os” on Win-GRAF toolbar.



Click Slot 8 or larger slot NO. (suggest value is 9) position. Then click “Select” button. Finally, mouse double click “Soft-GRAF” in the dialog.





(all)
ICP DAS - XP-WP-VI
Standard

i_87065_DO (* 8-ch D/O, AC-SSR *)
i_87066_DO (* 8-ch D/O, DC-SSR *)
i_87068_DO (* 8-ch D/O, Relay *)
i_87069_DO (* 8-ch D/O, PhotoMOS Relay *)
i_87082_cnt (* 2-ch 32-bit counter, 2-ch D/O *)
i_87082_freq (* 2-ch 32-bit frequency, 2-ch D/O *)
i_exist (* test if i-8k and i-87k boards exist or not *)
i_redundancy (* Enable a "Active_IP" in the redundancy system *)
i_scale (* Setup the scaling function No. 01 to 29 . *)
Ping_ip (* Test Ethernet / Internet connection *)
Soft_GRAF (* Enable Soft-GRAF driver *)

Soft_GRAF

Enable Soft-GRAF driver .

1-ch boolean inputs : Always TRUE

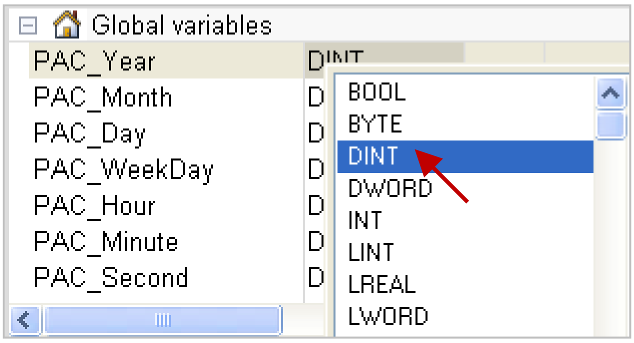
Jun.04,2014
ICP DAS , Taiwan

After adding “I/O boards”, please compile the Win-GRAF project. Then download to the Win-GRAF PAC. Then users can start to develop your Soft-GRAF HMI.

※**Note:** Please refer ISaGRAF FAQ146 chapter 1.2 to get more details about Soft-GRAF Studio.
ISaGRAF FAQ146:
http://www.icpdas.com/faq/isagraf/146.htm

Appendix A Data types and Ranges

Users can specify the data type of variables in the Variables Area (refer the Section 2.2.1) or in the Variables window (refer the Section 2.2.2).



Below are the available basic data types and ranges:

Data types	Size in-bits	Range of Values
BOOL (*)	---	TRUE, FALSE
SINT	8-bits (Small int, signed)	-128 to +127
USINT	8-bits (Unsigned small int)	0 to +255
BYTE		
INT	16-bits (Int, signed)	-32768 to +32767
UINT	16-bits (Unsigned int)	0 to +65535
WORD		
DINT (*)	32-bits (Double int, signed)	-2147483648 to +2147483647
UDINT	32-bits (Unsigned double int)	0 to +4294967295
DWORD		
LINT	64-bits (Large int, signed)	-2^{63} to $+(2^{63}-1)$
ULINT	64-bits (Unsigned large int)	0 to $+(2^{64}-1)$
LWORD		
REAL (*)	32-bits (Floating point)	$\pm3.4\times10^{-38}$ to $\pm3.4\times10^{38}$
LREAL	64-bits (Floating point)	$\pm1.7\times10^{-308}$ to $\pm1.7\times10^{308}$
STRING (*)	A max. of 255 characters	---
TIME (*)	32-bits	T#0ms to T#23h59m59s999ms

(*): The common used data type.

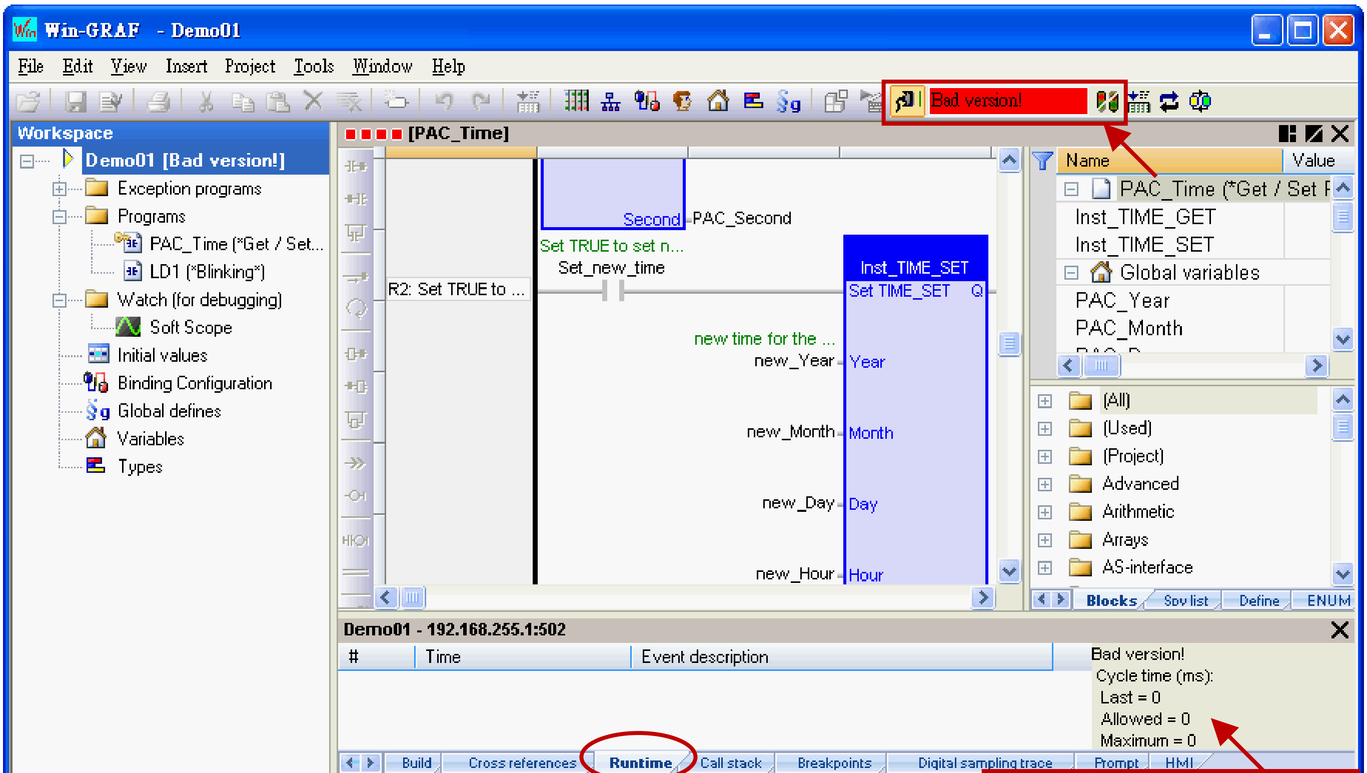
Appendix B Troubleshooting while On-Line the PAC

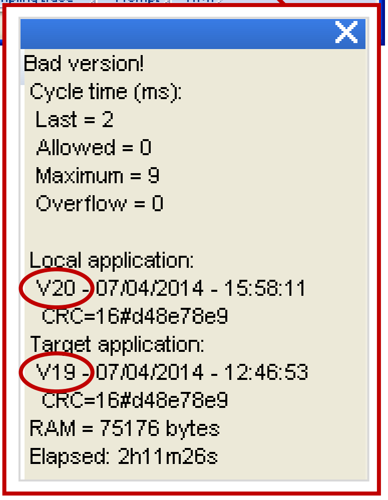
If the error message is showing up (as the screenshot below) after connecting to the Win-GRAF PAC, refer the following content to solve the problem.

- **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.

<u>To solve the problem</u>

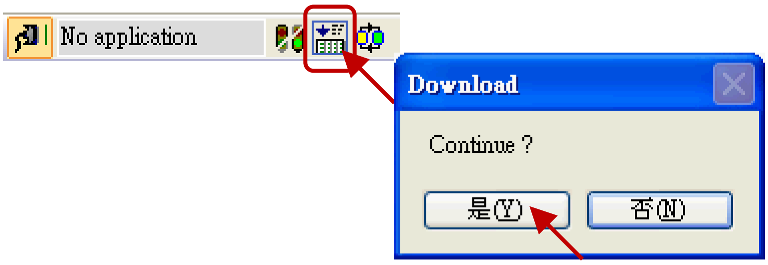
1. Click the "Stop application" button to stop the running program.



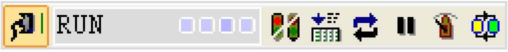

Bad version!
Cycle time (ms):
Last = 2
Allowed = 0
Maximum = 9
Overflow = 0

Local application:
V20 - 07/04/2014 - 15:58:11
CRC=16#d48e78e9
Target application:
V19 - 07/04/2014 - 12:46:53
CRC=16#d48e78e9
RAM = 75176 bytes
Elapsed: 2h11m26s

2. Click the "Download" button to download the program again.

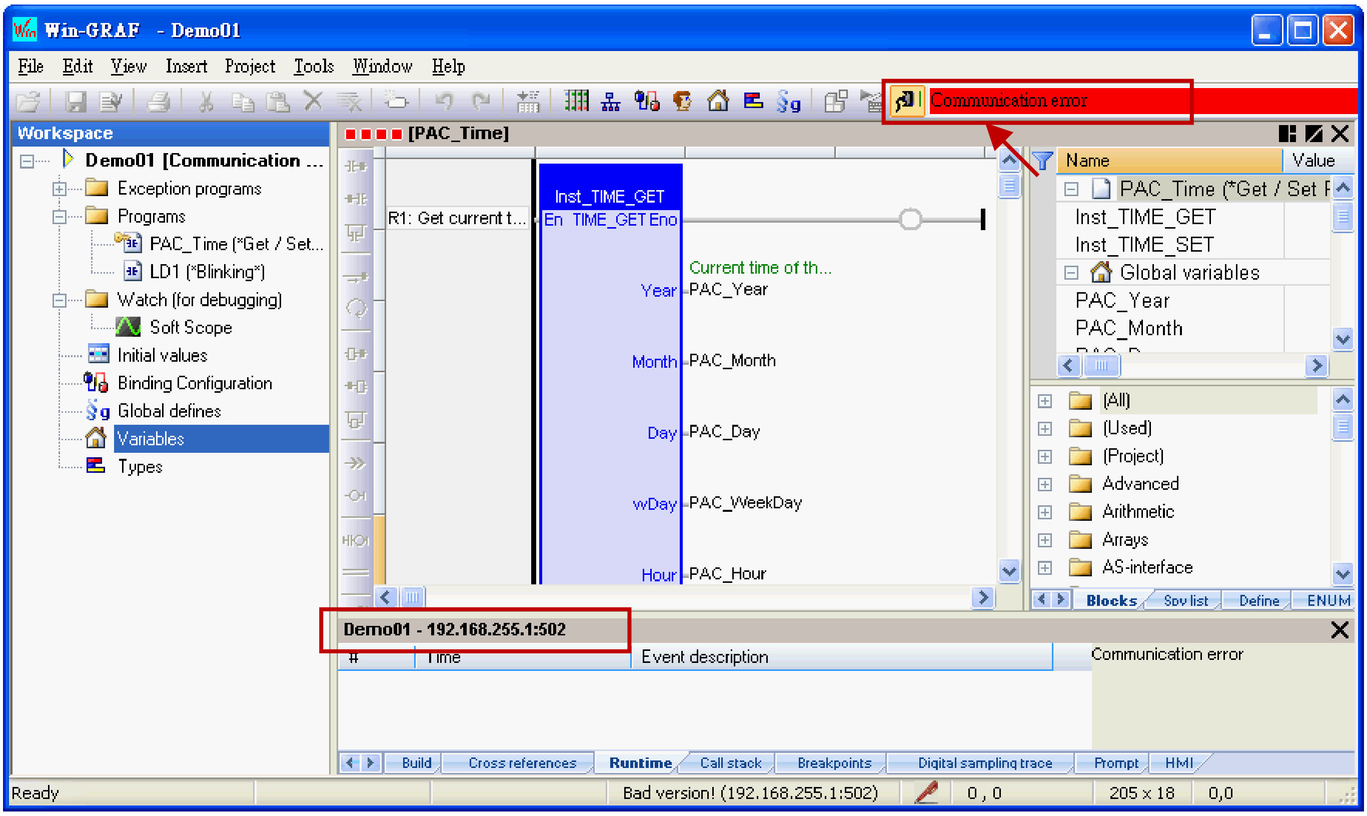


3. The "RUN" message means that the program is working properly.



- **The “Communication error” error message:**

  A communications failure has occurred between the PC and the PAC.



Communication error

Demo01 - 192.168.255.1:502

Bad version! (192.168.255.1:502)

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 IP setting of the Win-GRAF project is the same as the PAC IP (refer the Section 2.3.5, in this example, the IP address is “192.168.255.1:502”).
3. Make sure the network communication of your PC is working.

Appendix C Enable the Screen Saver of WinCE PAC

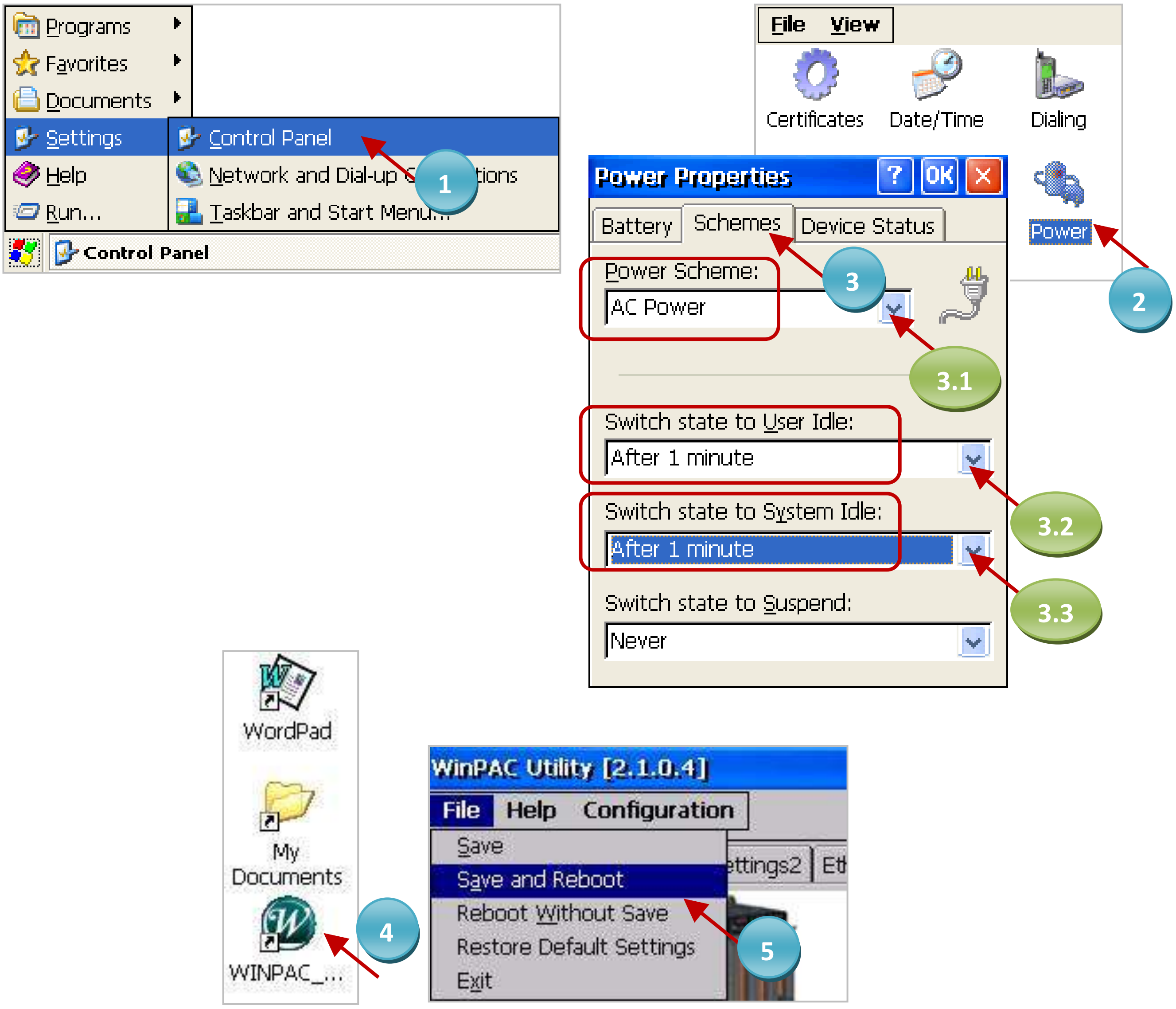
Please set the following two items to enable the screen of WinCE PAC.

1. Choose **“Control Panel**” > “Power” > “Schemes”, and set the “Power Scheme” as “AC Power”, set the “User Idle” and the “System Idle” to the same value (or set the “System Idle” value larger than the “User Idle” value).
2. Then, remember to run “**WinPAC Utility**” > “File” > “Save and Reboot” to save the settings and auto reboot the PAC.

<u>Using the WP-8xx8 as an example:</u>

If users do not touch the screen or button until the time out (e.g., 1 minute), the WP-8xx8 will turns off the backlight for enabling the screen saver. Whenever users touch the screen or button, the WP-8xx8 will turns on the backlight again.

The way to disable the screen saver is to set the “User Idle” and the “System Idle” as “Never”, and then remember to run “WinPAC Utility” > “File” > “Save and Reboot” to save the settings and auto reboot the PAC.



Power Scheme: AC Power

Switch state to User Idle: After 1 minute

Switch state to System Idle: After 1 minute

Switch state to Suspend: Never

Appendix D Using Expansion RS-232/485/422

The Win-GRAF PAC (See P1-1) expand more COM port in its slot No. 0 to 7 by using following modules.

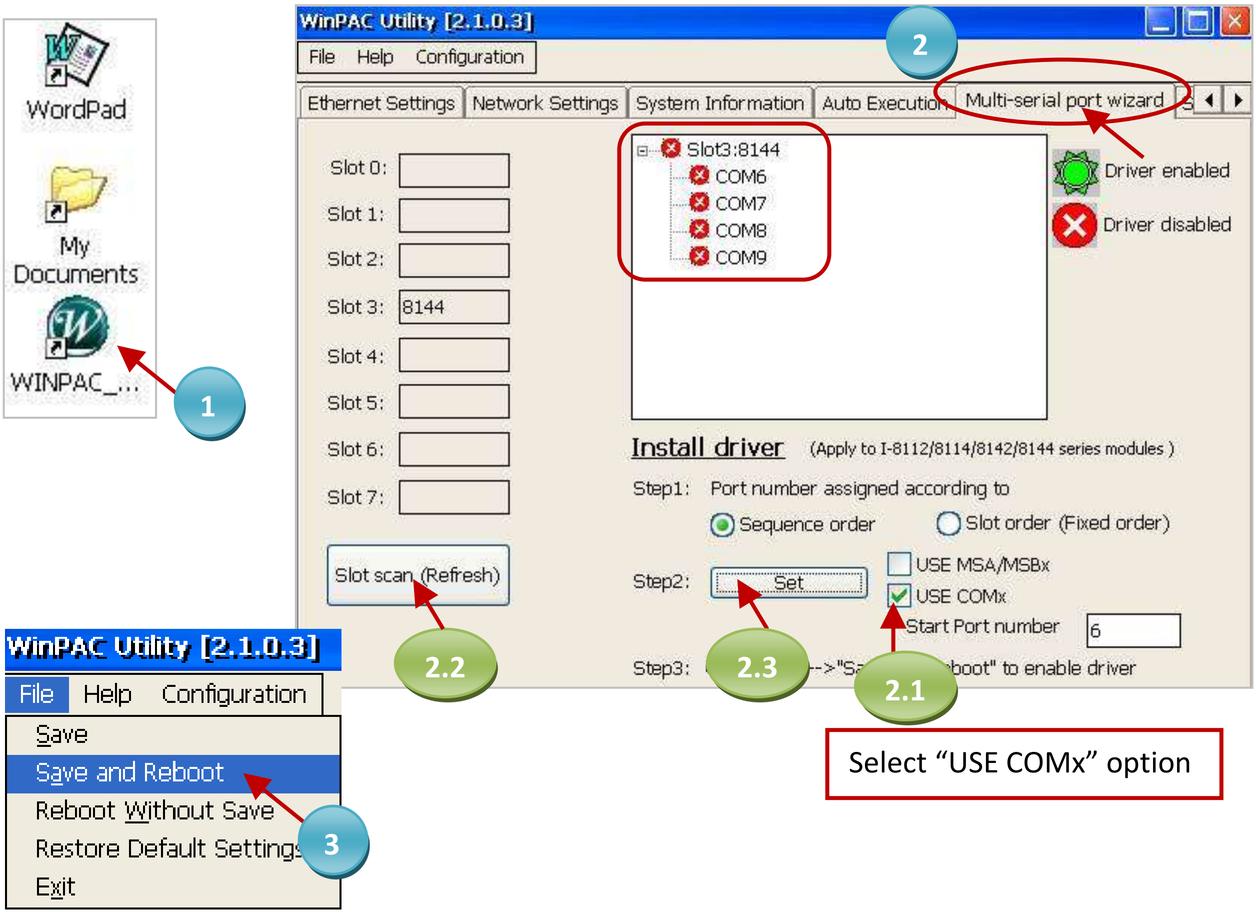
I-8112iW : 2-port Isolated RS-232 module
I-8114iW : 4-port Isolated RS-232 module
I-8114W : 4-port RS-232 module
I-8142iW : 2-port Isolated RS-422/RS-485 module
I-8144iW : 4-port Isolated RS-422/RS-485 module

Note: The **WP-5xx8** does not support **XW-5xx** series XW-board. (This PAC can not expand COM port.)

Using the WP-8xx8 as an example:

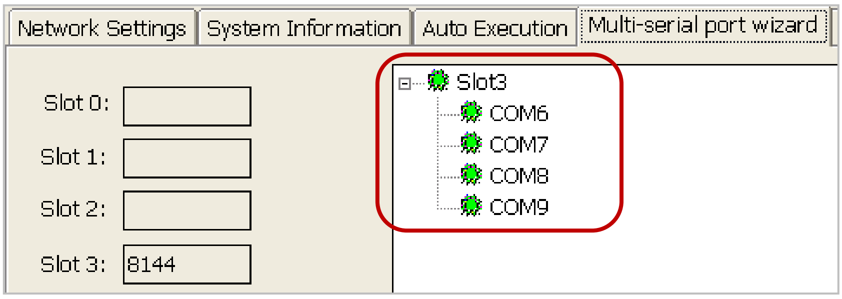
Before using these modules, please configure then by the “WinPAC Utility”. First, plug the module in the WP-8xx8’s slot 0 to 7 (It is better to be in slot 0 to 3), and then run the “WinPAC Utility”.

- Click the “Multi-serial port wizard” tab.
- Check the“USE COMx” option. (**Note:** The Win-GRAF doesn’t support “USE MSA/MSBx”)
- Click the “Slot scan” button, and then the current found serial-port expansion module will be listed on the left. (The earlier COM port setting is listed on the right if you have already set it before.)
- Click the “Set” button to refresh the new setting like the figure below.
- Click [File] > [Save and Reboot] to save the new setting and auto reboot the WP-8xx8.



Select “USE COMx” option

After the configuration succeeds, the COM port No. for the expansion is COM6 to COM37 in the Win-GRAF definition. (In this case, it expands the COM6 to COM9).


Network Settings | System Information | Auto Execution | Multi-serial port wizard

Slot 0:

Slot 1:

Slot 2:

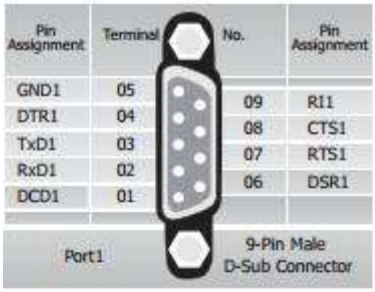
Slot 3: 8144

- Slot3
  - COM6
  - COM7
  - COM8
  - COM9

Pin Assignment:

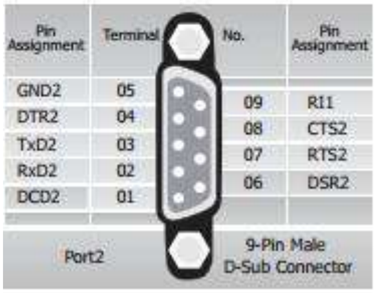
I-8112iW

2-port Isolated RS-232



Pin Assignment	Terminal	No.	Pin Assignment
GND1	05	09	RI1
DTR1	04	08	CTS1
TxD1	03	07	RTS1
RxD1	02	06	DSR1
DCD1	01		

Port1 — 9-Pin Male D-Sub Connector

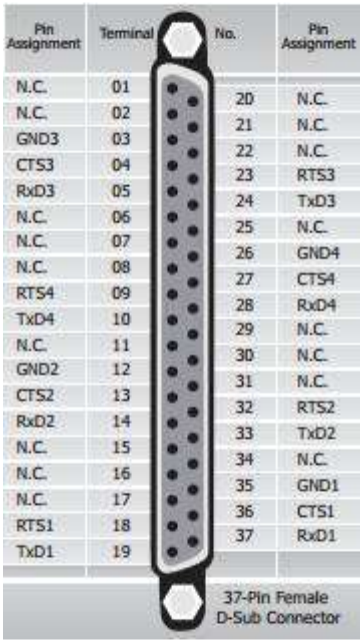


Pin Assignment	Terminal	No.	Pin Assignment
GND2	05	09	RI1
DTR2	04	08	CTS2
TxD2	03	07	RTS2
RxD2	02	06	DSR2
DCD2	01		

Port2 — 9-Pin Male D-Sub Connector

I-8114iW

4-port Isolated RS-232

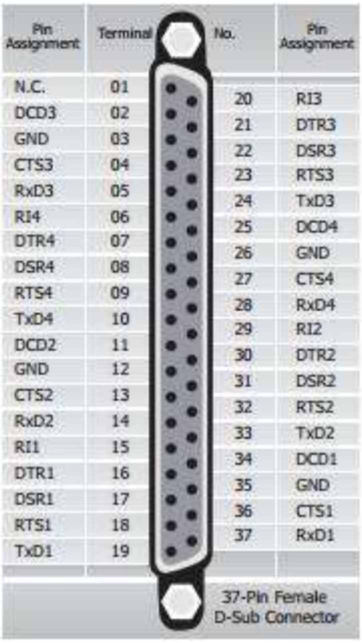


Pin Assignment	Terminal	No.	Pin Assignment
N.C.	01	20	N.C.
N.C.	02	21	N.C.
GND3	03	22	N.C.
CTS3	04	23	RTS3
RxD3	05	24	TxD3
N.C.	06	25	N.C.
N.C.	07	26	GND4
N.C.	08	27	CTS4
RTS4	09	28	RxD4
TxD4	10	29	N.C.
N.C.	11	30	N.C.
GND2	12	31	N.C.
CTS2	13	32	RTS2
RxD2	14	33	TxD2
N.C.	15	34	N.C.
N.C.	16	35	GND1
N.C.	17	36	CTS1
RTS1	18	37	RxD1
TxD1	19		

37-Pin Female D-Sub Connector

I-8114W

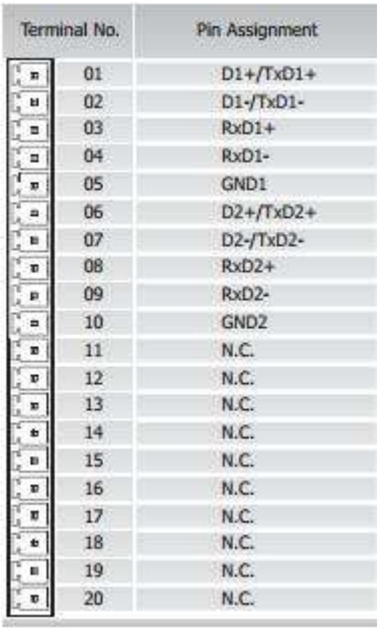
4-port RS-232



Pin Assignment	Terminal	No.	Pin Assignment
N.C.	01	20	RI3
DCD3	02	21	DTR3
GND	03	22	DSR3
CTS3	04	23	RTS3
RxD3	05	24	TxD3
RI4	06	25	DCD4
DTR4	07	26	GND
DSR4	08	27	CTS4
RTS4	09	28	RxD4
TxD4	10	29	RI2
DCD2	11	30	DTR2
GND	12	31	DSR2
CTS2	13	32	RTS2
RxD2	14	33	TxD2
RI1	15	34	DCD1
DTR1	16	35	GND
DSR1	17	36	CTS1
RTS1	18	37	RxD1
TxD1	19		

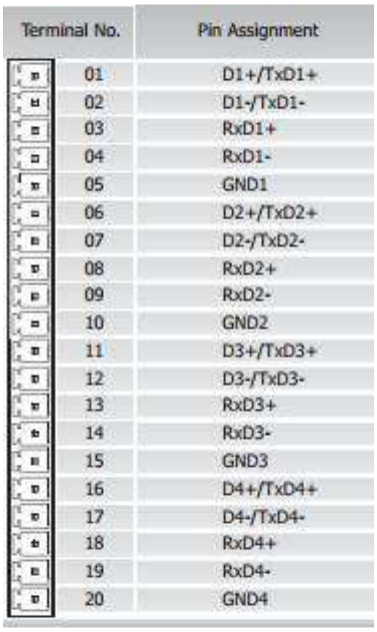
37-Pin Female D-Sub Connector

I-8142iW



Terminal No.	Pin Assignment
01	D1+/TxD1+
02	D1-/TxD1-
03	RxD1+
04	RxD1-
05	GND1
06	D2+/TxD2+
07	D2-/TxD2-
08	RxD2+
09	RxD2-
10	GND2
11	N.C.
12	N.C.
13	N.C.
14	N.C.
15	N.C.
16	N.C.
17	N.C.
18	N.C.
19	N.C.
20	N.C.

I-8144iW



Terminal No.	Pin Assignment
01	D1+/TxD1+
02	D1-/TxD1-
03	RxD1+
04	RxD1-
05	GND1
06	D2+/TxD2+
07	D2-/TxD2-
08	RxD2+
09	RxD2-
10	GND2
11	D3+/TxD3+
12	D3-/TxD3-
13	RxD3+
14	RxD3-
15	GND3
16	D4+/TxD4+
17	D4-/TxD4-
18	RxD4+
19	RxD4-
20	GND4

I-8142iW (2-port Isolated RS-422/485)

RS-485 port1: (D1+ , D1-)
RS-485 port2: (D2+ , D2-)

RS-422 port1: (TxD1+ , TxD1-, RxD1+, RxD1-)
RS-422 port2: (TxD2+ , TxD2-, RxD2+, RxD2-)

I-8144iW (4-port Isolated RS-422/485)

RS-485 port1: (D1+ , D1-)
RS-485 port2: (D2+ , D2-)
RS-485 port3: (D3+ , D3-)
RS-485 port4: (D4+ , D4-)

RS-422 port1: (TxD1+ , TxD1-, RxD1+, RxD1-)
RS-422 port2: (TxD2+ , TxD2-, RxD2+, RxD2-)
RS-422 port3: (TxD3+ , TxD3-, RxD3+, RxD3-)
RS-422 port4: (TxD4+ , TxD4-, RxD4+, RxD4-)