HMIWorks

The Development Software for the TouchPAD Series

User Manual Version 1.5

ICP DAS CO., LTD.
service@icpdas.com
https://www.icpdas.com

Warning

ICP DAS assumes no liability for any damage resulting from the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, or for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

The names used for identification only may be registered trademarks of their respective companies.

Support

ICP DAS takes your problem as ours.
If you have any problem, please feel free to contact us.
You can count on us for quick response.

Email: service@icpdas.com

Also, the web site of ICP DAS has contents about TouchPAD which you may be interested in. We believe that those contents may be helpful to your work.

web site:https://www.icpdas.com/en/product/guide+Panel__Products+TouchPAD+TPD__Series

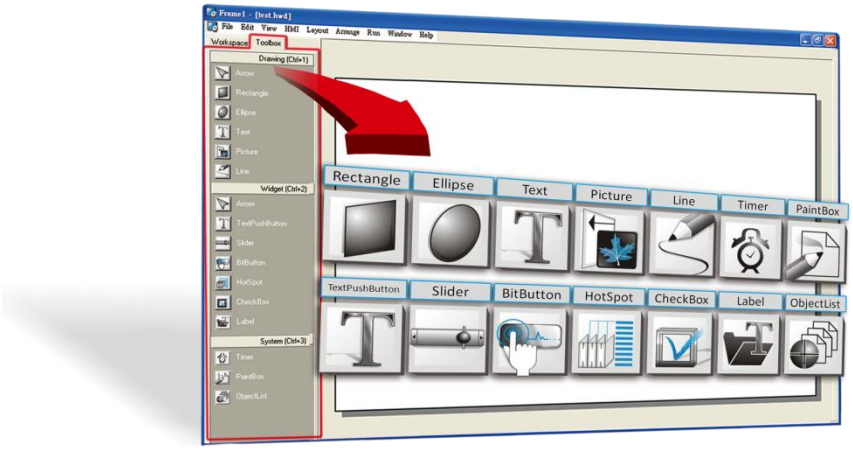
Table of Contents

1. Introduction

HMIWorks is free development software for TouchPAD series products of ICP DAS. It features of many widgets, built-in extensible graphics library, intuitive design, C programming, Ladder Diagram supporting, fully I/O integration... etc. Using with TouchPAD series devices, HMIWorks can help users to short the development time and design the sophisticated, cost effective solutions for the complex systems.

Support Many Widgets - Shorten Development Time

There are many widgets included in the HMIWorks development tool, including Rectangle, Ellipse, Text, Picture, Line, TextPushButton, Slider, BitButton, HotSpot, CheckBox, Label, Timer, PaintBox, ObjectList, providing the most commonly-used functions, such as drawings, event handlers, and timing control, which effectively shortens development time.



C and Ladder Diagram Programming

65536 Colors - Bright and Clear

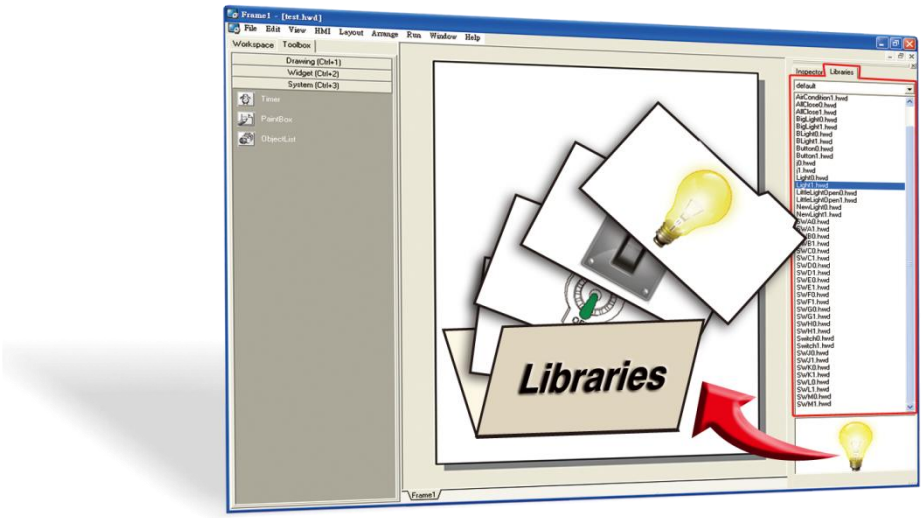
Presently, LCD touch screens are available at 2.8”, 3.5”, 4.3” , 7” and the TouchPAD series includes different resolutions from 240 x 320, 480 x 272, 800 x 480. ICP DAS will expand this range in the future.

Intuitive Design

HMIWorks provides an intuitive graphical design interface that allows users to focus on what they want to do. By getting rid of the programming details and being more intuitive, everyone can easily finish their projects.

Built-in the Extensible Graphics Library

HMIWorks supports simple graphics functions and provides users with a variety of built-in graphics for common situations. Users can also add their own graphics to the library by the common painting or photo editing software.



Drag-and-drop Design - fully integrate with I/O (support third party modules)

ICP DAS now supports many I/O devices, such as ET-7000/PET-7000 series Modbus TCP modules, M-7000 series Modbus RTU modules, I-7000 series DCON modules and user-defined third party Modbus TCP devices. Users can expect that additional I/O devices will be supported by HMIWorks for the TouchPAD series in the future.

1.1 Features

Features of HMIWorks include:

- FREE of charge (for ICP DAS TouchPAD devices)
- Two programming types, ladder diagram and Standard C
- Plenty of widgets
- Plenty of demos shorten development time
- Advanced search for I/O modules
- Detail error messages
- Easy downloading after building
- Automatic generated codes for user-designed frames
- Multi-frame design
- Abstract graphics as simple APIs
- Easy learning IDE to raise productivity in short time
- Data exchange function

1.2 Support in ICP DAS Products

The following is a summary of TPD/VDP Series produced by ICP DAS that support the HMIWorks software.

TPD High Speed Series Models	TPD-280-H, TPD-280U-H, TPD-283-H, TPD-280-M1, TPD-280-M2, TPD-280-M3, TPD-283-M1, TPD-283-M2, TPD-283-M3, TPD-283U-M1, TPD-283U-M2, TPD-283U-M3, TPD-430-H, TPD-433-H, TPD-433F-H, TPD-432F-H, TPD-433-M2, TPD-703, TPD-703-64
VPD High Speed Series Models	VPD-130-H, VPD-130N-H, VPD-132-H, VPD-132N-H, VPD-133-H, VPD-133N-H, VPD-130-H2, VPD-130N-H2, VPD-133-H2, VPD-133N-H2, VPD-142-H, VPD-142N-H, VPD-143-H, VPD-143N-H VPD-173N , VPD-173N-64, VPD-173X , VPD-173X-64

The following products are phase out: The last supported version is HMIWorks v2.10.61.

TPD Series Models	TPD-280, TPD-280U, TPD-283, TPD-283U, TPD-430, TPD-430-EU, TPD-433, TPD-433-EU, TPD-432F, TPD-433F
VPD Series Models	VPD-130, VPD-130N, VPD-132, VPD-132N, VPD-133, VPD-133N, VPD-142, VPD-142N, VPD-143, VPD-143N

2. Software Installation

The following is a detailed description of the process for obtaining, installing and removing the HMIWorks development software.

2.1 Obtaining the development software Installation Package

The installation package for the HMIWorks development software can be obtained from the FTP site or the ICP DAS web site. The locations and addresses are indicated below:

Note: TPD-280/283/280U/238U, TPD-430/433/432F/433F, VPD-130(N)/132(N)/133(N) and VPD-142(N)/143(N) has be phased out. The last supported version is HMIWorks v2.10.61, and the versions after HMIWorks v2.30.xx no longer support the above products.

https://www.icpdas.com/en/download/show.php?num=944

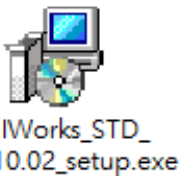
Operating system of Windows requirement

32-bit(x86)	64-bit(x64)
Microsoft Windows 7	Microsoft Windows 7
Microsoft Windows 2008	Microsoft Windows 2008
Microsoft Windows 8	Microsoft Windows 8
Microsoft Windows 2012	Microsoft Windows 2012
Microsoft Windows 10	Microsoft Windows 10

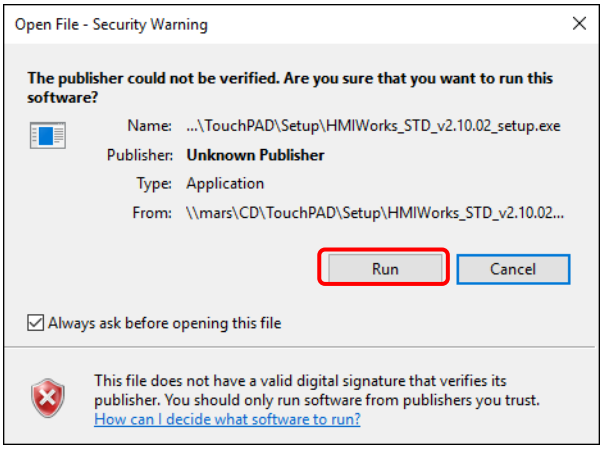
2.2 Development software Installation Procedure

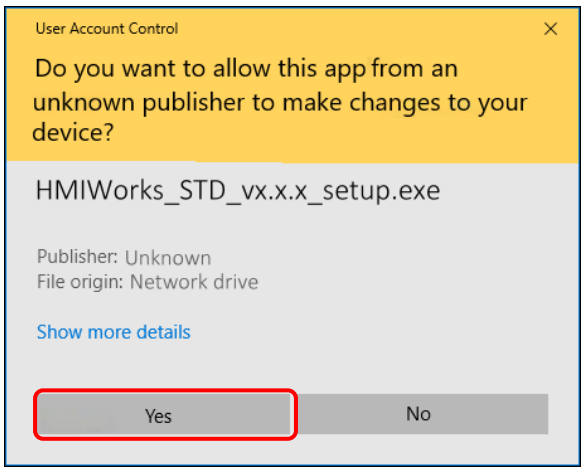
Here, the Windows 10 is used as an example. To install the HMIWorks development software, follow the procedure described below:

Step 1: Double-click the **"HMIWorks_STD_vxxx_setup.exe"** file icon to execute the development software installation program.

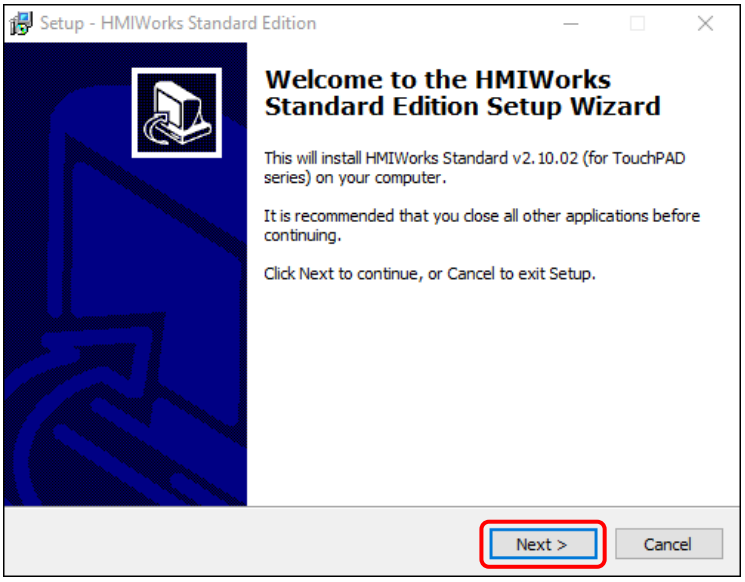


Note: More recent operating system, such as Windows 10, will display secruity warning message asking you to confirm whether you wish to install the software. Click the **"Run"** and **"Yes"** button to continue.

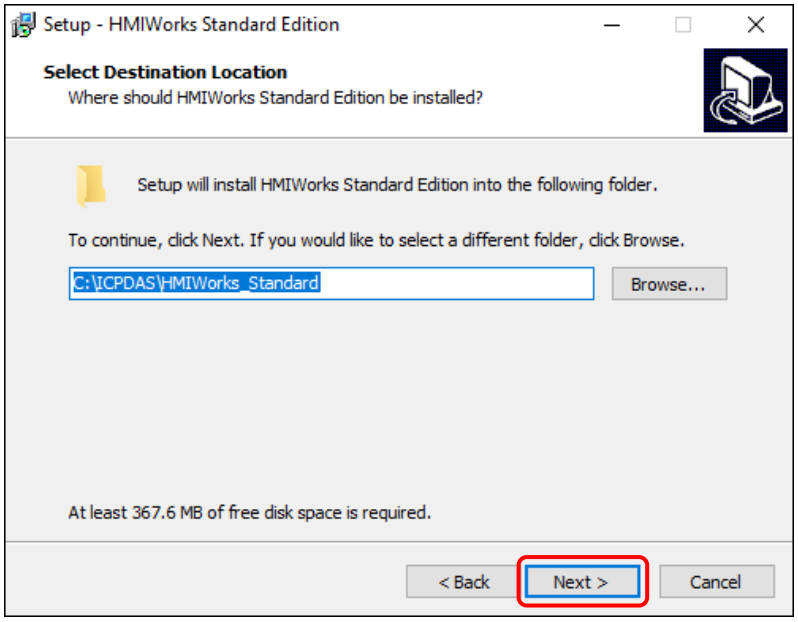




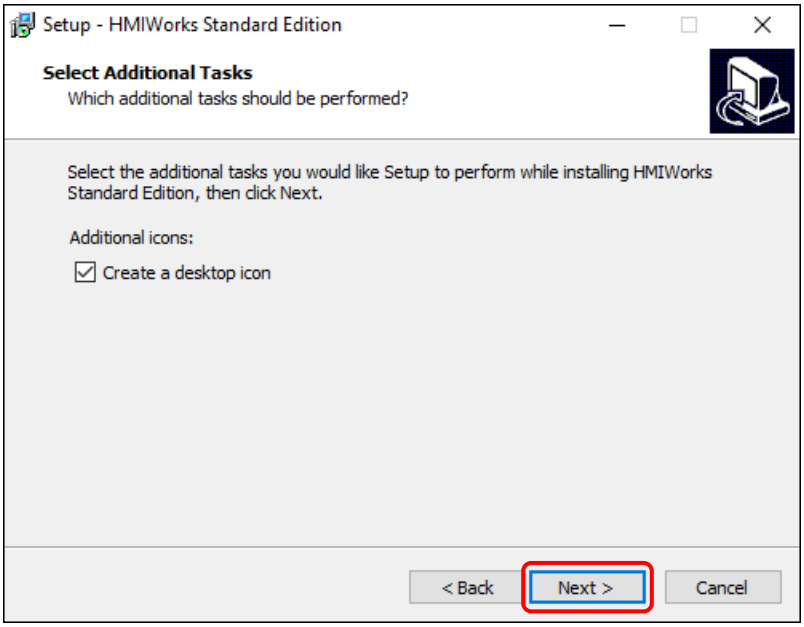
Step 2: Once the **"Setup – HMIWorks Standard Edition"** Installation Wizard screen is displayed, click the **"Next>"** button to start the installation.



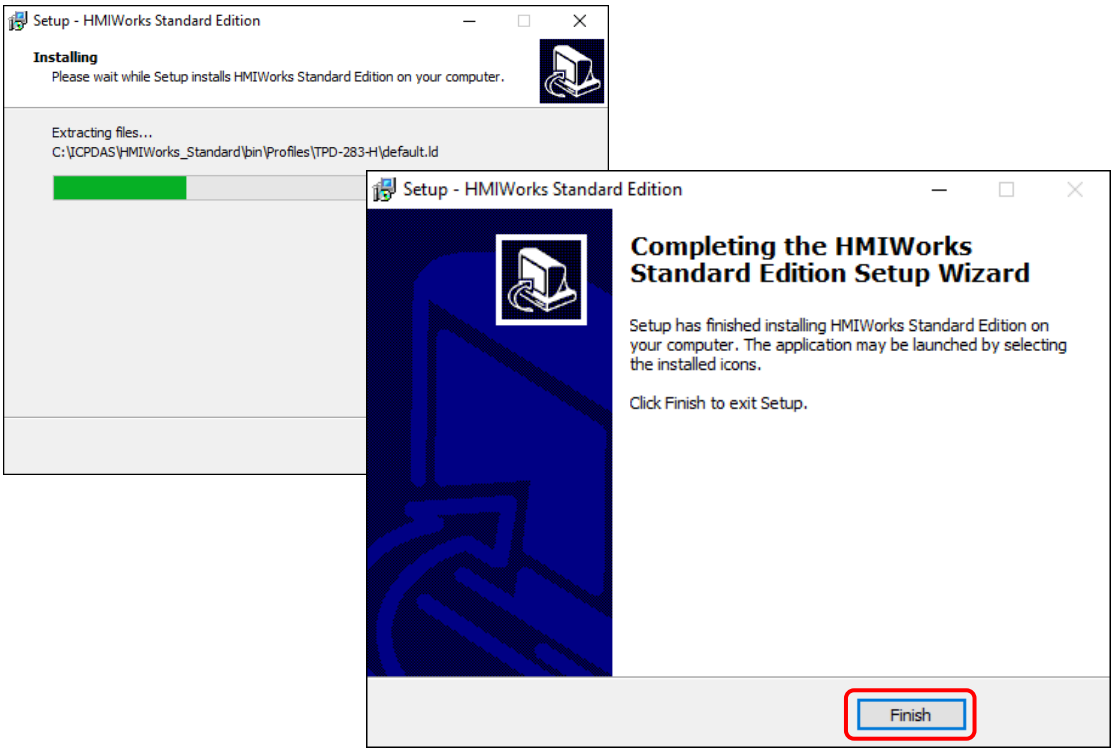
Step 3: Select the destination location. The **default path is C:\ICPDAS\HMIWorks_Standard.** Verify that the destination path is correct and click the **"Next >"** button, or click the **"Browse…"** button to install the development software in a different location. It is strongly recommended that the development software is installed in the default location.



Step 4: Click the **"Next >"** button on the **"Select Additional Tasks"** screen to continue.

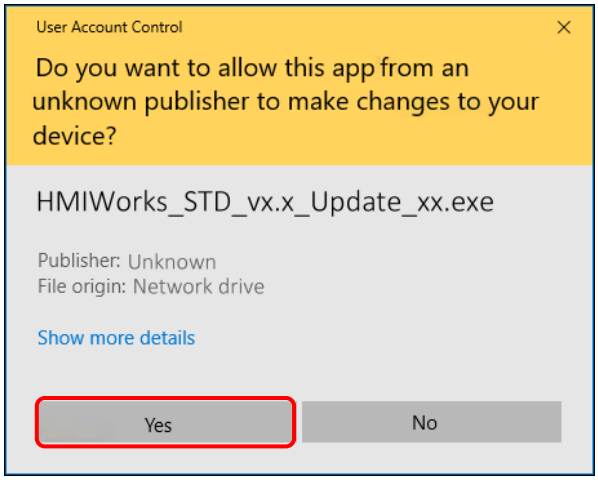


Step 5: Click the **"Finish"** button to complete the installation.

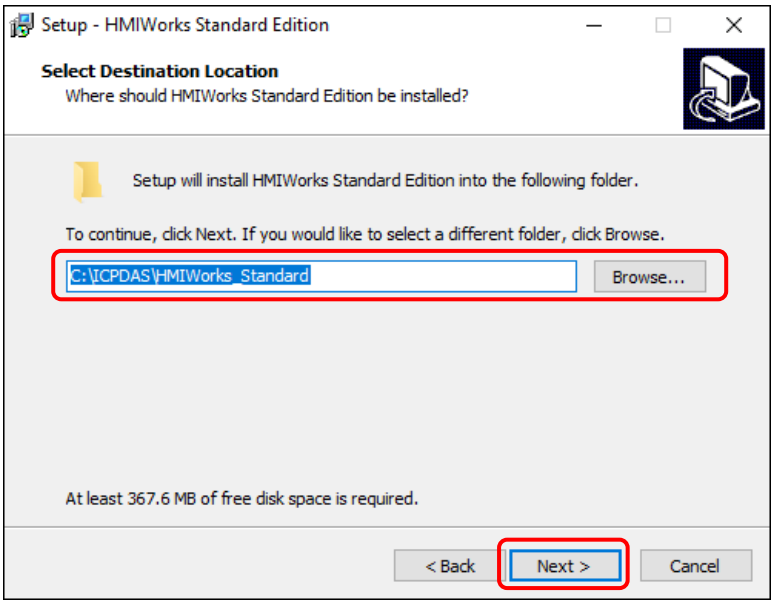


Step 6: Once the development software installation is complete, double-click the **"HMIWorks_STD_vxxx_Update_xx.exe"** file icon to execute the development software installation update program.

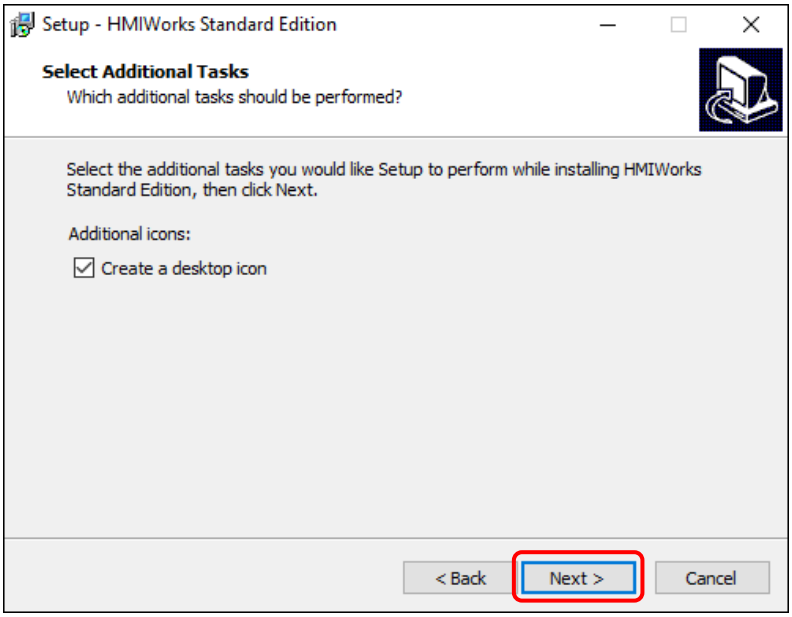
Note: More recent operating system, such as Windows 10, will display secruity warning message asking you to confirm whether you wish to install the software. Click the **"Yes"** button to continue.



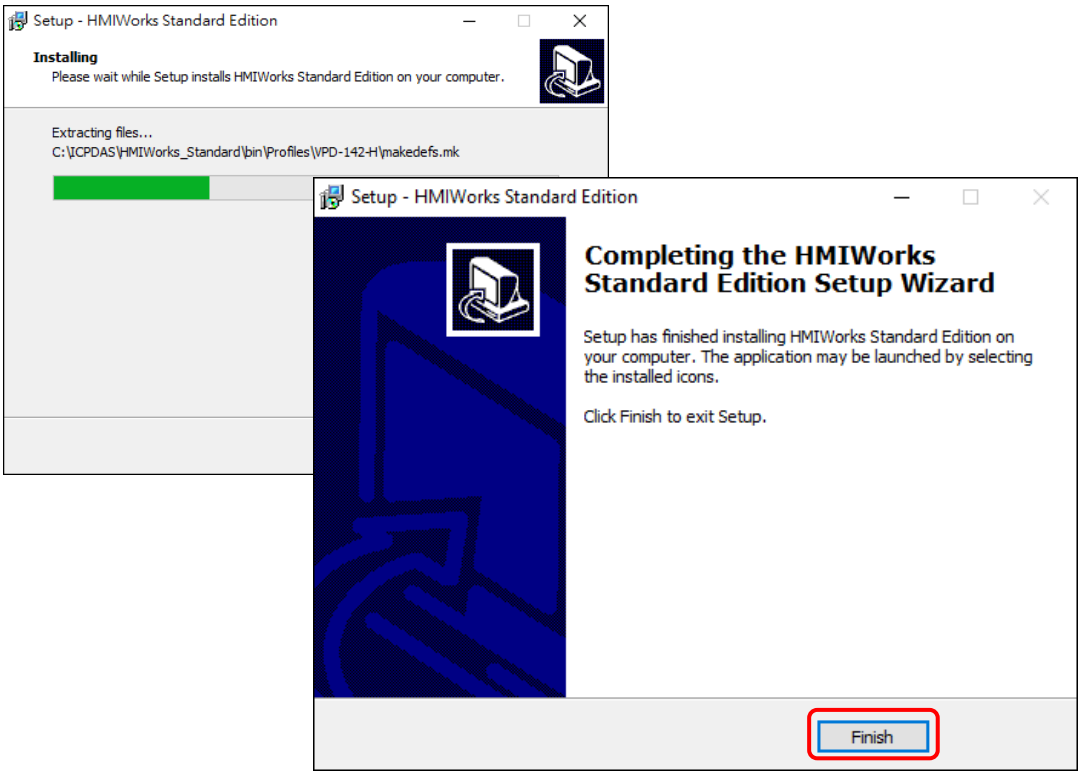
Step 7: Select the destination location. The **default path is C:\ICPDAS\HMIWorks_Standard.** Verify that the destination path is correct and click the **"Next >"** button, or click the **"Browse…"** button to install the development software in a different location. It is strongly recommended that the development software is installed in the default location.



Step 8: Click the **"Next >"** button on the **"Select Additional Tasks"** screen to continue.



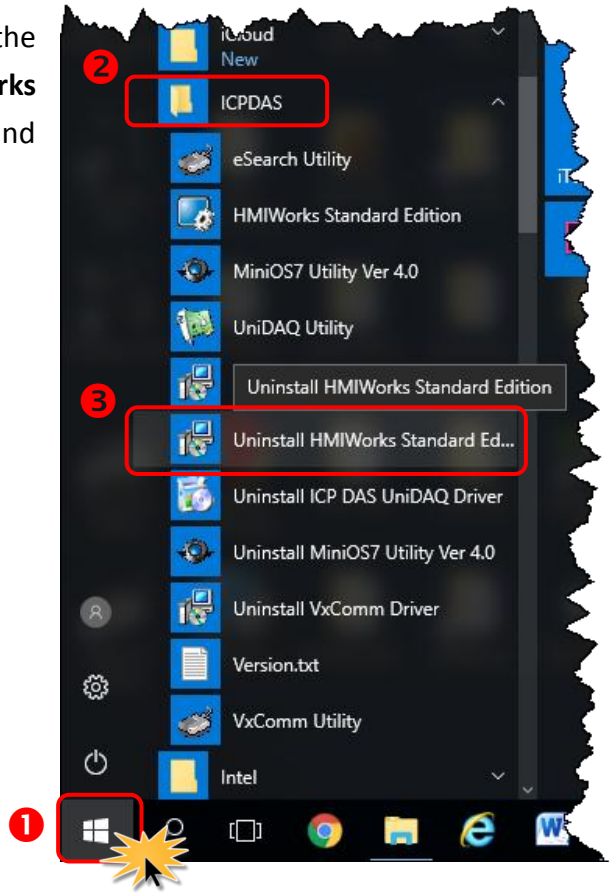
Step 9: Click the **"Finish"** button to complete the installation.



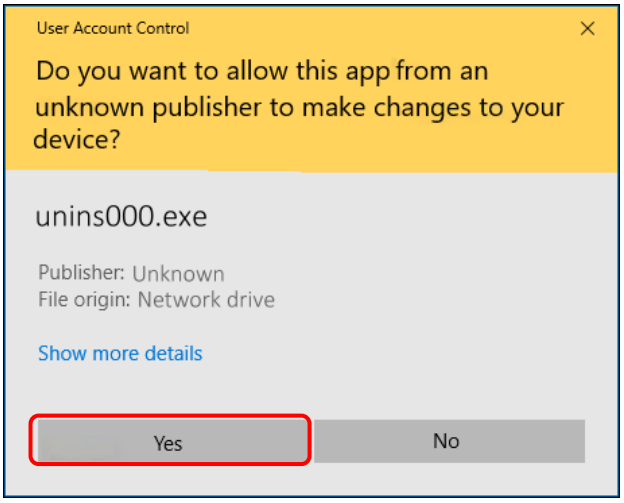
2.3 Uninstalling the development software

The HMIWorks development software includes an uninstallation utility that allows the software to be removed from the computer if necessary. Here, the Windows 10 is used as an example. To uninstall the software, follow the procedure described below:

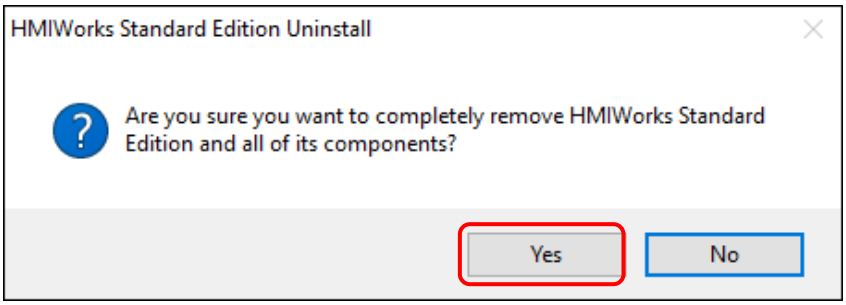
Step 1: Click the Windows **"Start"** button and click the **"ICP DAS"** folder**,** then click the **"Uninstall HMIWorks Standard Edition"** item to run the uninstall process and remove the development software.



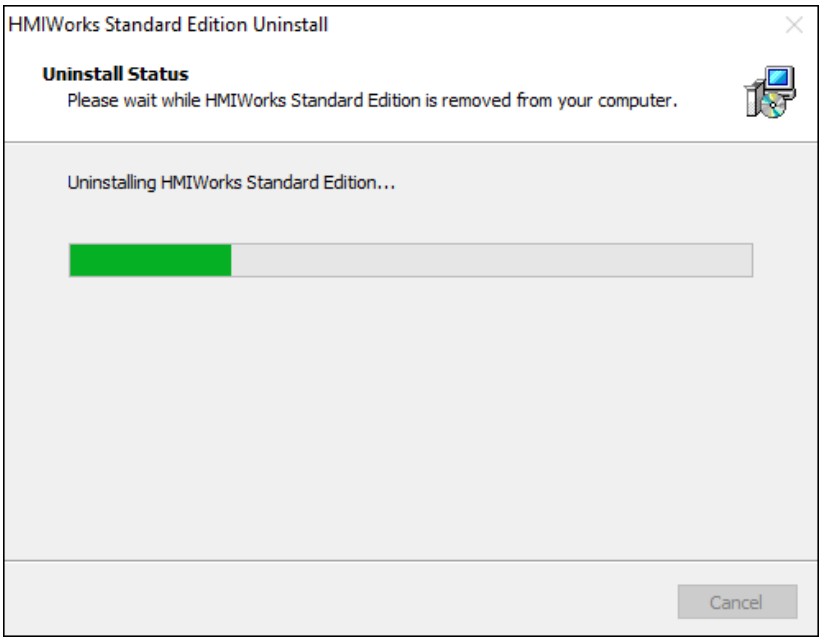
Note: More recent operating system, such as Windows 10, will display secruity warning message asking you to confirm whether you wish to allow software from an unknown publisher to make changes to the computer. Click the **"Yes"** button to continue.



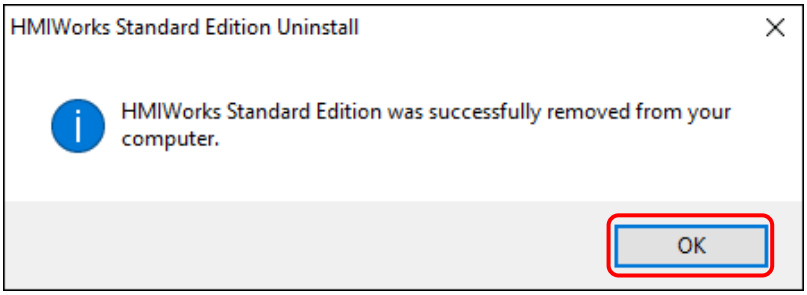
Step 2: A dialog box will be displayed asking for confirmation that you want to remove the HMIWorks Standard Edition. Click the **"Yes"** button to continue.



Step 3: Uninstalling HMIWorks Standard Edition on the **"Uninstall Status"** screen.



Step 4: After the uninstallation process is complete, a dialog box will be displayed to indicate that the development software was successfully removed. Click the **"OK"** button to finish the uninstallation process.



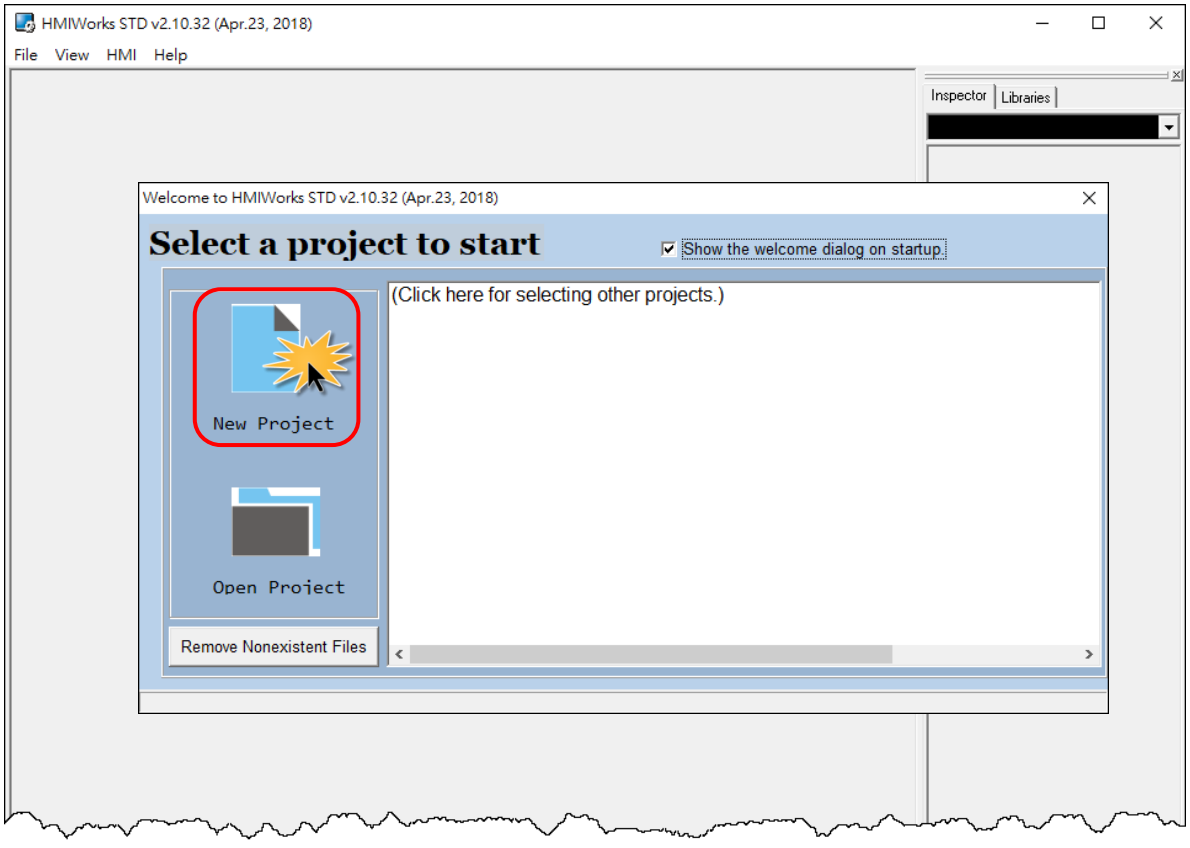
3. HMIWorks Working Environment

Once the HMIWorks development software installation is complete, a shortcut to the HMIWorks_Standard Utility will be created on the Windows desktop. Double click the shortcut to open the HMIWorks_Standard Utility, each of which will be described in more detail below.

3.1 The Construction of HMIWorks

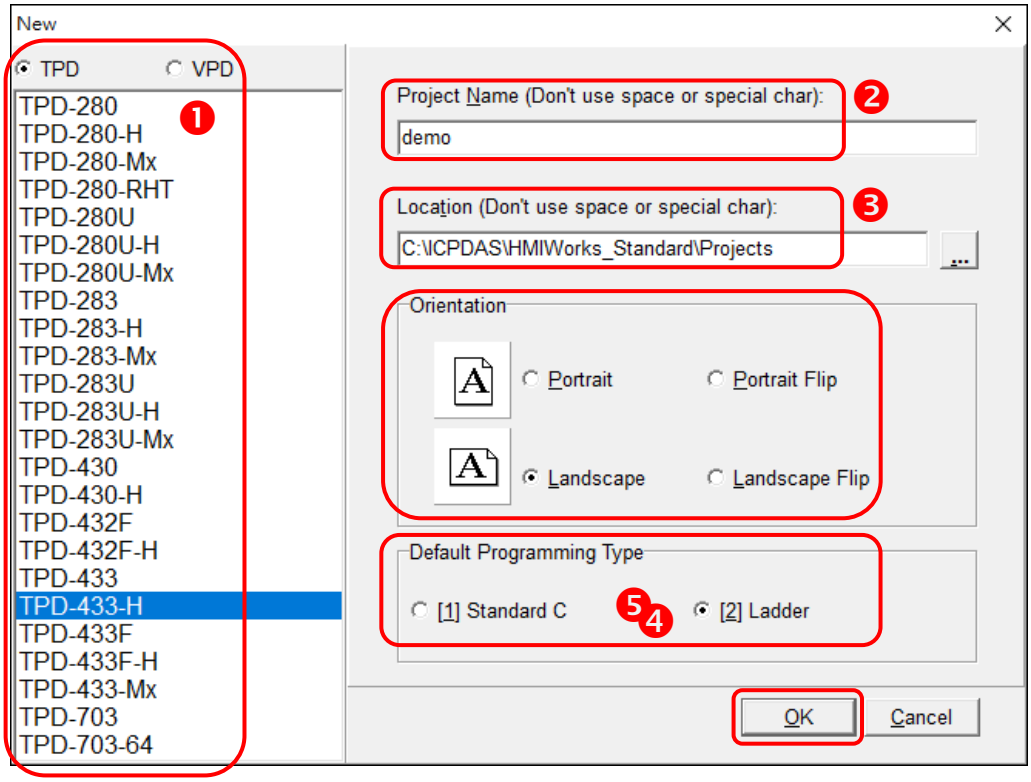
Before showing the construction of HMIWorks, create a new project first.

Step 1: Click the "**New Project**" icon to create a new project.
(or click the **"New…"** from the **"File"** menu to create a new project.)



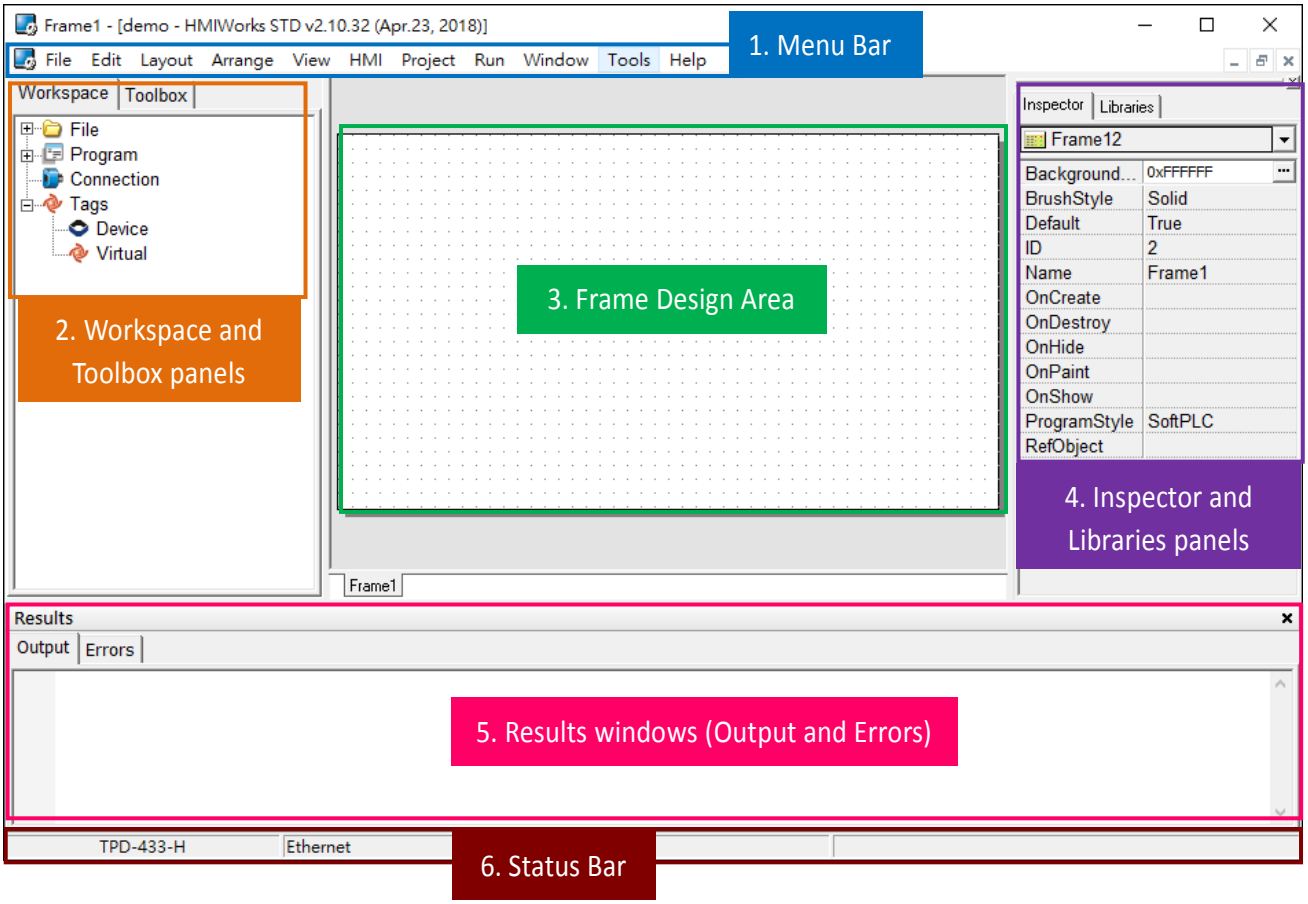
Step 2: In the **"New"** window, configure the parameters for the new project as follows:

1. Click the name of the TouchPAD model to select it (e.g., TPD-433-H).
2. Enter a name for the project (e.g., demo).
3. Select the location where the project should be saved (Use the default path).
4. Select the orientation for the display (e.g., Landscape).
5. Select the Default Programming Type (e.g., Ladder).
6. Click the **"OK"** button to save the configuration and close the window.



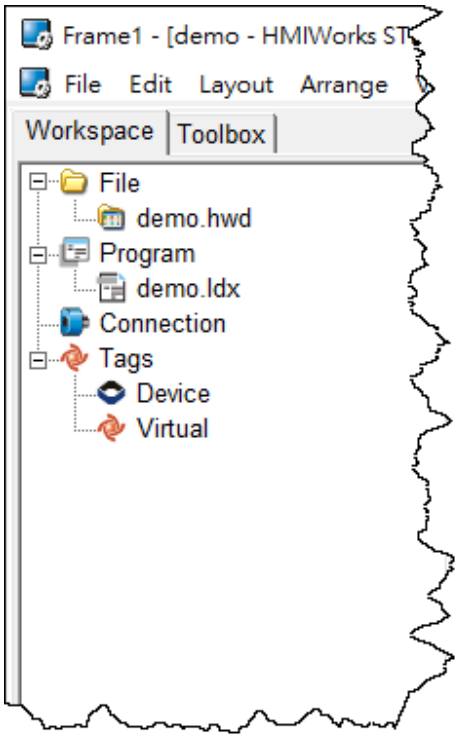
Note: A valid project name is a sequence of one or more letters, digits or underscore characters (_). It must not begin with a digit. Besides, it is of suggested length 100 characters (including its path).

➢ HMIWorks integrated design environment shows as below.



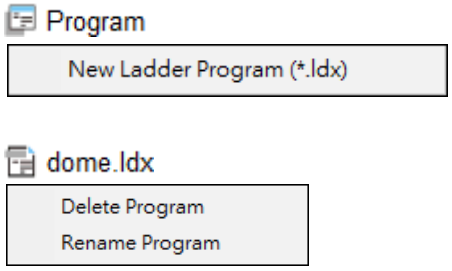
There are several parts of HMIWorks		
1.	**Menu bar**	This is the main menu of the HMIWorks. Refer to Section 3.5 Menus for more detailed information.
2.	**Workspace panel**	Refer to the next page will have more detailed information about Workspace.
	Toolbox panel	Refer to Section 3.4 Frames and Components for more detailed information about Toolbox.
3.	**Frame Design area**	You can set up an application program in this area. Refer to Section 3.4 Frames and Components for more detailed information.
4.	**Inspector panel**	Refer to Section 3.4 Frames and Components for more detailed information.
	Libraries panel	
5.	**Results window (Output and Errors)**	This window will show the output and error status when the execution compile and download.
6.	**Status bar**	Shows the status of the TouchPAD device.

- In the "**Workspace"** panel provides allowing items such as the **"File"**, **"Program"**, **"Connection"** and **"Tags"** to be configured (add, delete and edit). For example: right click on the **"File"** item will pop-up function menus, as shown below.

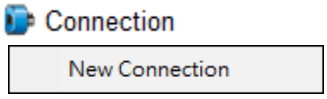


File	Display and configure the Project file.

Program	Display and configure the Program file.

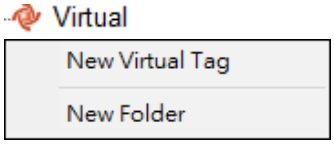


Connection	Display and configure the TCP/IP and serial port connection for Modbus TCP/RTU device.



Tags	Display and configure the virtual tags and Modbus TCP/RTU device.

Refer to Section 3.3.6 Associate Tags with Tools and Section 3.3.7 User-Defined I/O Modules for more detailed information.



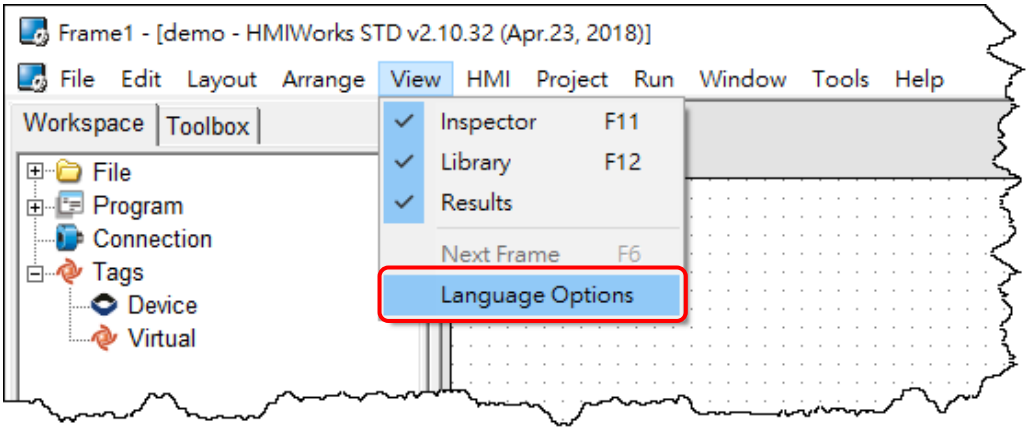
Refer to Section 3.3 Ladder Designer for more detailed information.

3.2 The Options of TouchPAD

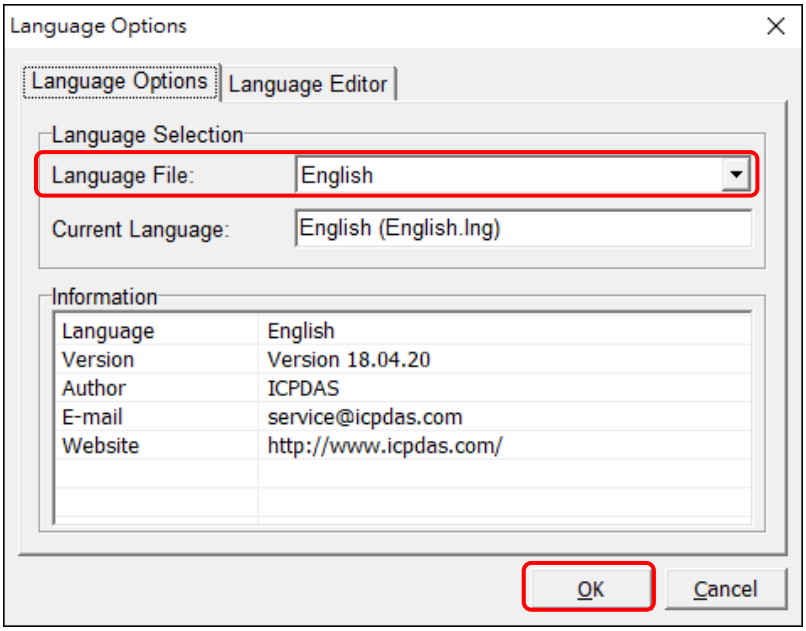
3.2.1 Language Options

The following instructions guide you to set the HMIWorks interface language

Step 1: Click the **"Language Options"** from the **"View"** menu.



Step 2: In the "Language Options" window, **select a language** from the **"Language File"** drop down options and click the **"OK"** button.

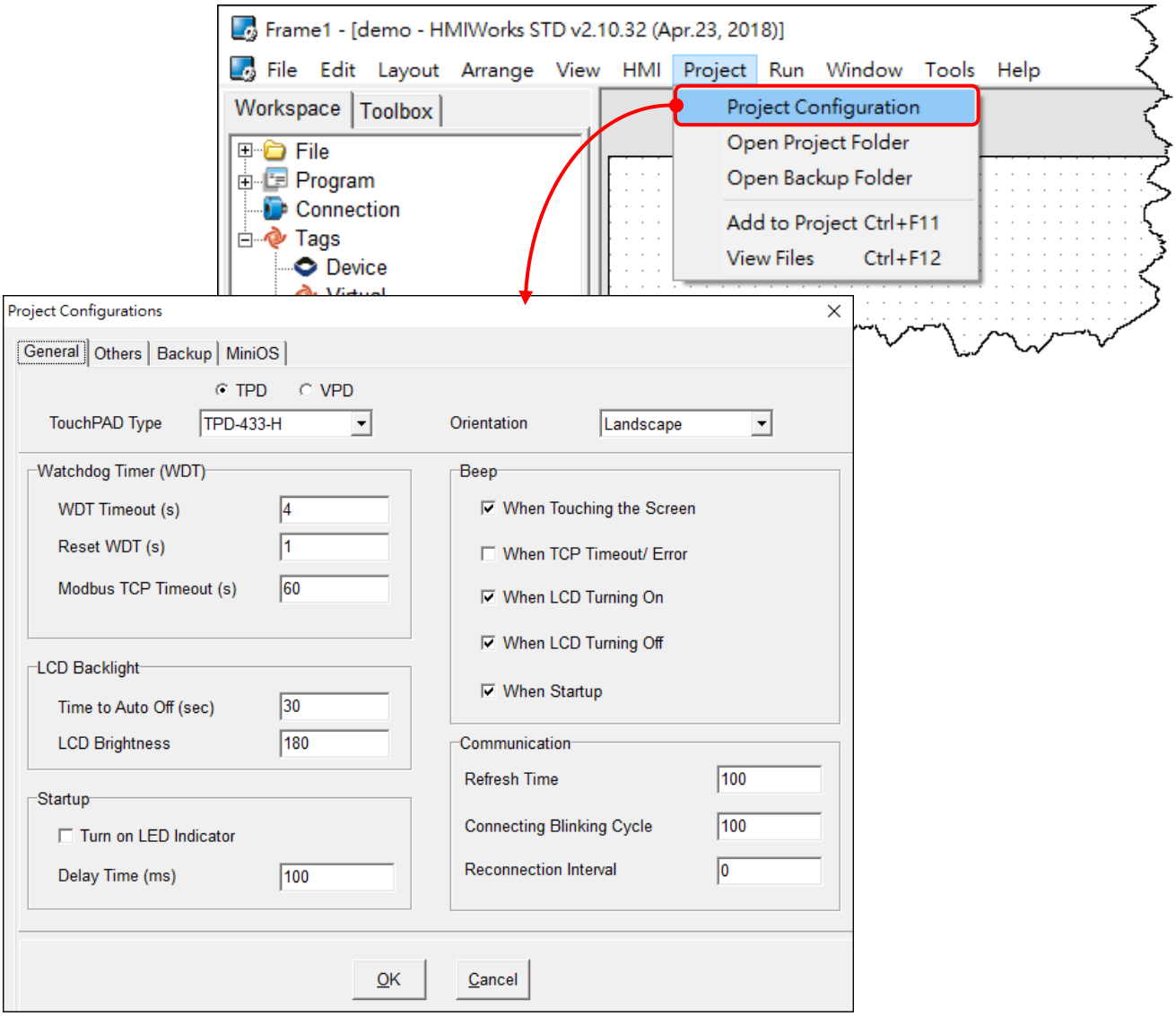


3.2.2 Project Configurations

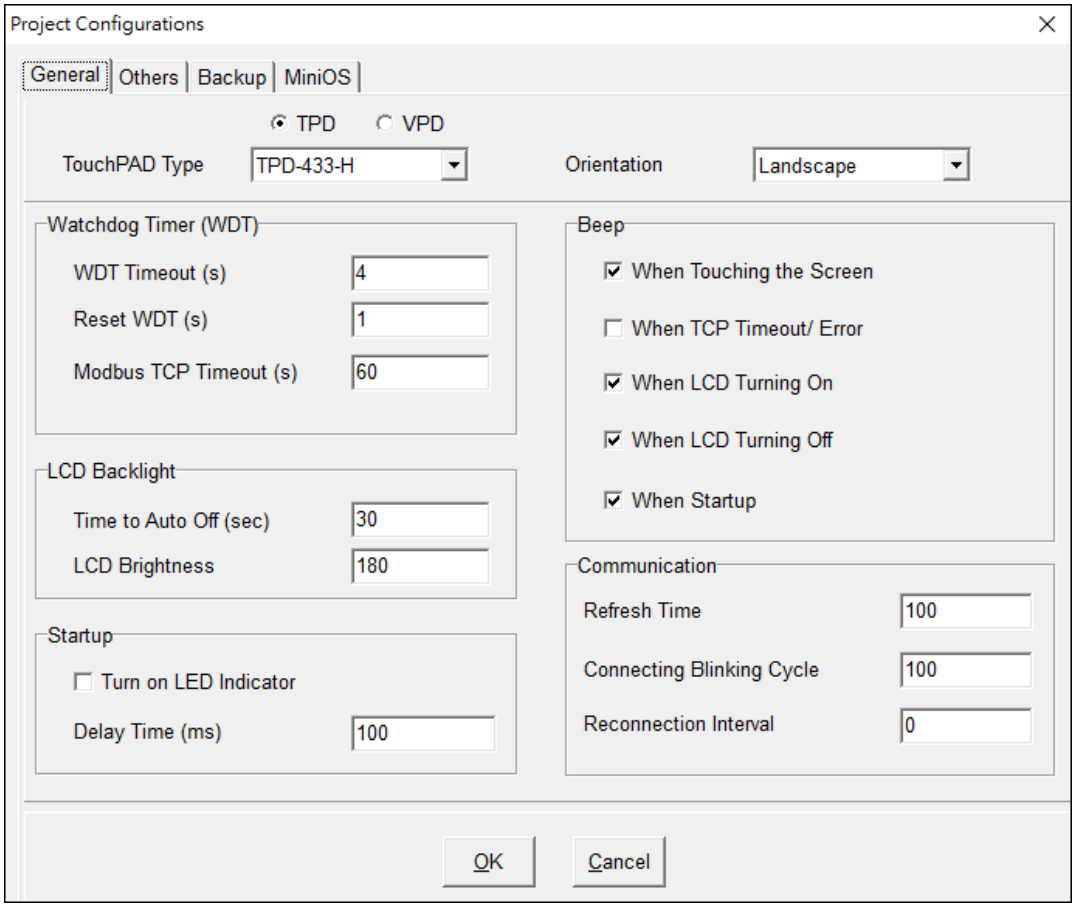
The **"Project Configurations"** provides functions allowing items such as the Watchdog timer, LCD backlight, communication and backup, etc. to be configured, each of which will be described in more detail below.

Open the Project Configuration

Click the **"Project Configuration"** from the **"Project"** menu to open the "Project Configurations" window.



General

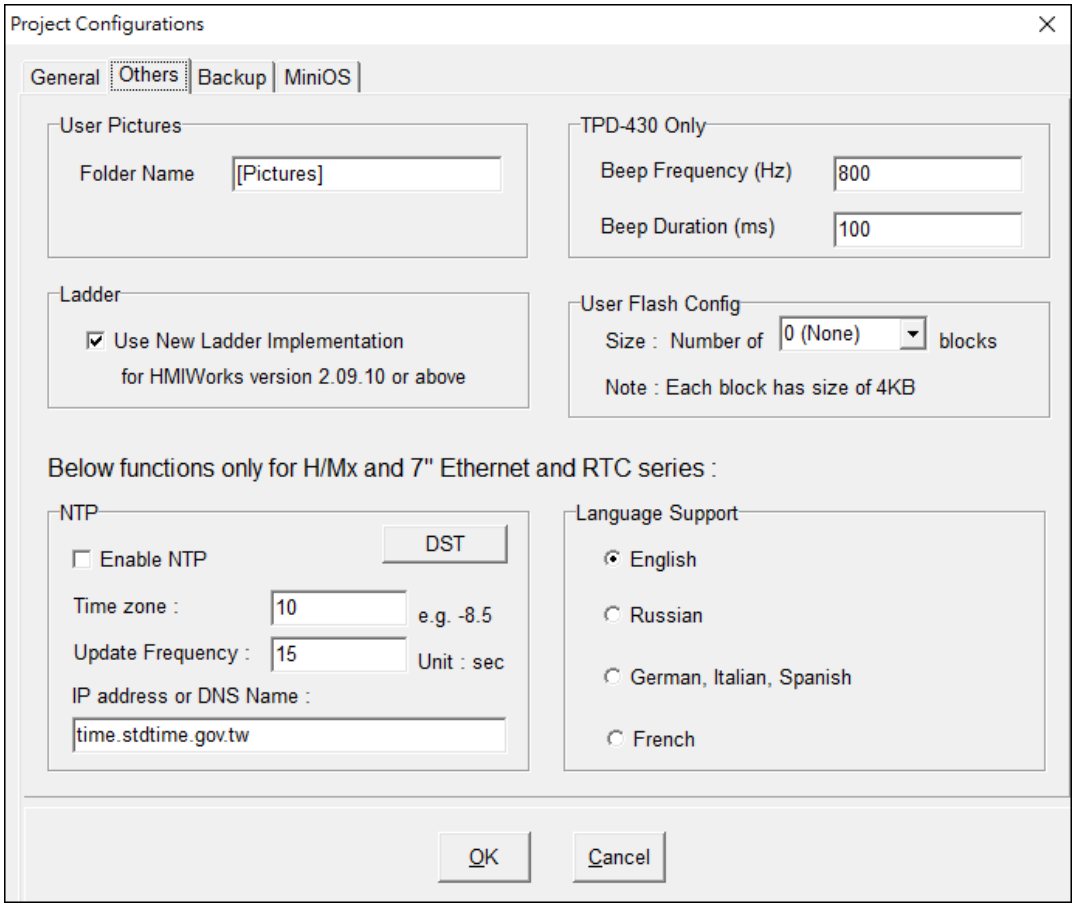


The following is an overview of the functions contained in the **General** section:

Option		Descriptions
TouchPAD Type		After changing these two options, HMIWorks automatically scale the size of every frame and every widget to maintain the relative positions between each other. **Note:** the Text component is not scaled.
Orientation		
Watchdog (WDT)	**WDT Timeout (s)**	The timeout value in seconds to reboot. Valid Range: 1 ~ 50 s
	Reset WDT (s)	The period to reset the Watchdog timer to prevent rebooting in seconds. (Suggested: 25% of the timeout value)
	Modbus TCP Timeout (s)	The timeout value of Modbus TCP in seconds to reboot. Valid Range: 10 ~ 10,000 s

Option		Descriptions
LCD Backlight	**Time to Auto Off (sec)**	Time to turn off the LCD backlight automatically when touch screen is idle in second. (Default: 30 sec)
	LCD Brightness	Specify the brightness level of the screen. (Default: 180) Valid Range: 0 ~ 255. 0: the darkest, 255: the brightest
Startup	**Turn on LED Indicator**	Turn on LED indicator when TouchPAD starts up.
	Delay Time (ms)	Time to delay TouchPAD on start up in millisecond. (Default: 100 ms)
Beep	**When Touching the Screen**	Make TouchPAD issue a beep when the screen is touched. If this item is checked, the **hmi_PlaySong** function becomes useless.
	When TCP Timeout/Error	Make TouchPAD issue a beep when the TCP communication has timeout or error.
	When LCD Turning On	Make TouchPAD issue a beep when the LCD backlight turns on.
	When LCD Turning Off	Make TouchPAD issue a beep when the LCD backlight turns off.
	When Startup	Make TouchPAD issue a beep when it starts up.
Communication	**Refresh Time**	Interval of I/O and Ladder scan time (Default: 100 ms)
	Connecting Blinking Cycle	Used for communications of Modbus TCP master polling (remote slave devices), the Connecting Blinking Cycle defines the blinking period of “ERROR” tag used in devices which can be found in the Workspace.
	Reconnection Interval	The interval between two groups of 7 consecutive connections tries.
OK		Click this button to save the revised settings.
Cancel		Click this button to stop and closing the window.

Others

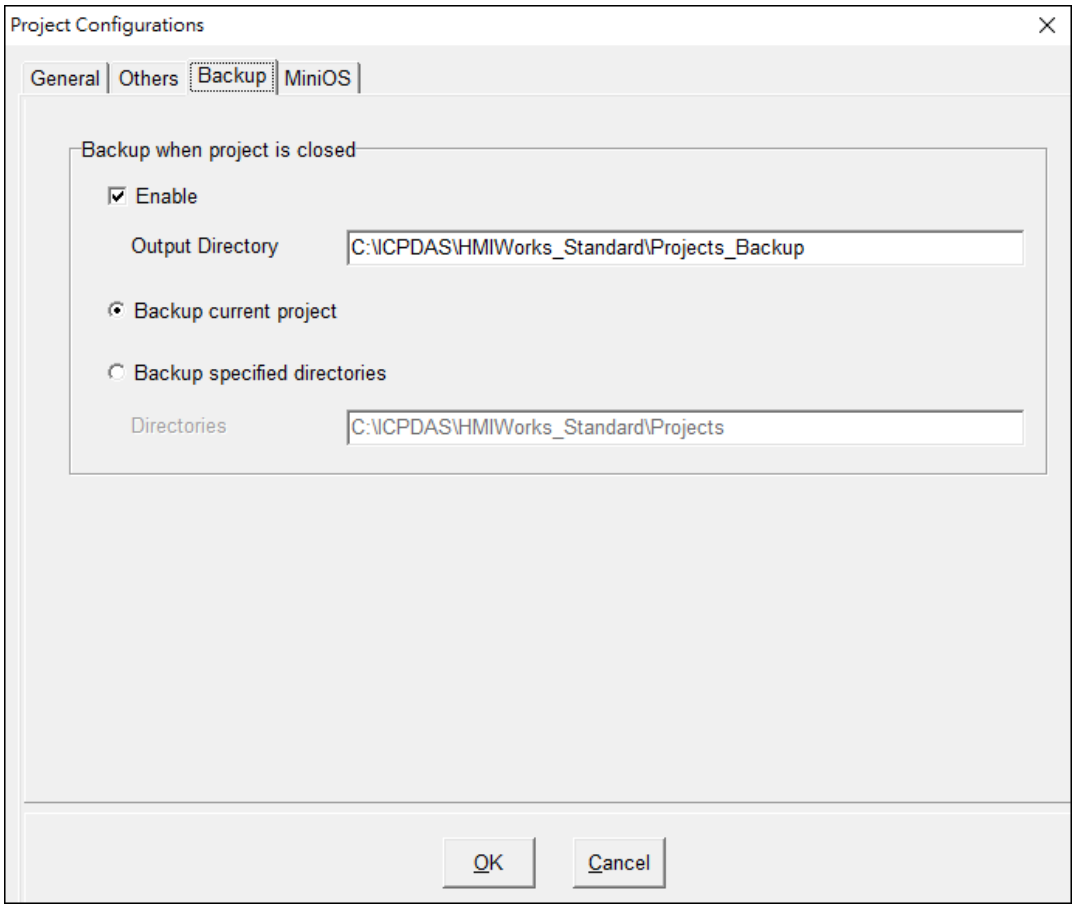


The following is an overview of the functions contained in the **Others** section:

Option		Descriptions
User Pictures	**Folder Name**	The folder name (relative path) that stores user's pictures under project.
Ladder	**Use New Ladder Implementation for HMIWorks version 2.09.10 or above**	If your original project uses Ladder program, and is created by HMIWorks v2.09.09 or older versions, please unchecked this item to disable the new Ladder mode. **New ladder mode:** The Coil-Set and Coil-Reset change the coil state and lock it (industrial standard) until reset or set. Other coil operations will not unlock or change it. **Old ladder mode:** There is no lock feature.

Option		Descriptions
TPD-430 Only	**Beep Frequency (Hz)**	Specify the frequency of the beep. (Default 800 Hz) Valid Range: 30 ~ 4,000 Hz
	Beep Duration (ms)	Specify the duration of the beep. (Default 25 ms)
User Flash Config	**Size**	Specify you need flash size. (Default 0) The flash is used to store the project program in general situation. Users can cut part of the flash space for other purposes. For example, do data logging function. **Note:** It will reduces the size of storable project files and has 100,000 write limits for each location of memory.
Below functions only for H/Mx and 7" Ethernet and RTC Series:		
NTP	**Enable NTP**	We can get time from the NTP server automatically after NTP is enabled.
	Time Zone	Set the Time Zone according to your real location.
	Update Frequency	Set update rate. (Unit: sec)
	IP address or DNS Name	Set NTP Server.
Language Supports		Built-in multilingual support that includes **English, Russian, German/Italian/Spanish (European) and French**. The default is English, if you want to use language other than English, refer to FAQ: How to display multilingual text on TouchPAD by using the HMIWorks built-in fonts? for more details. If there are some languages not list, such as CJK (Chinese/Japanese/Korean), etc., you can install the **ebFonts** to support more languages, refer to FAQ: How to install ebFonts to support multilingual feature? For more details.
OK		Click this button to save the revised settings.
Cancel		Click this button to stop and closing the window.

Backup



The following is an overview of the functions contained in the **Backup** section:

Option		Descriptions
Backup when project is closed	**Enable**	Enable backup which is executed when a project is closed. The backup files are compressed in the format, .7z.
	Output Directory	The Location where the backup compressed files are placed.
	Backup current project	-
	Backup specified directories	Directory: Directories to be backed up. Use semicolon (;) to separate directories.
OK		Click this button to save the revised settings.
Cancel		Click this button to stop and closing the window.

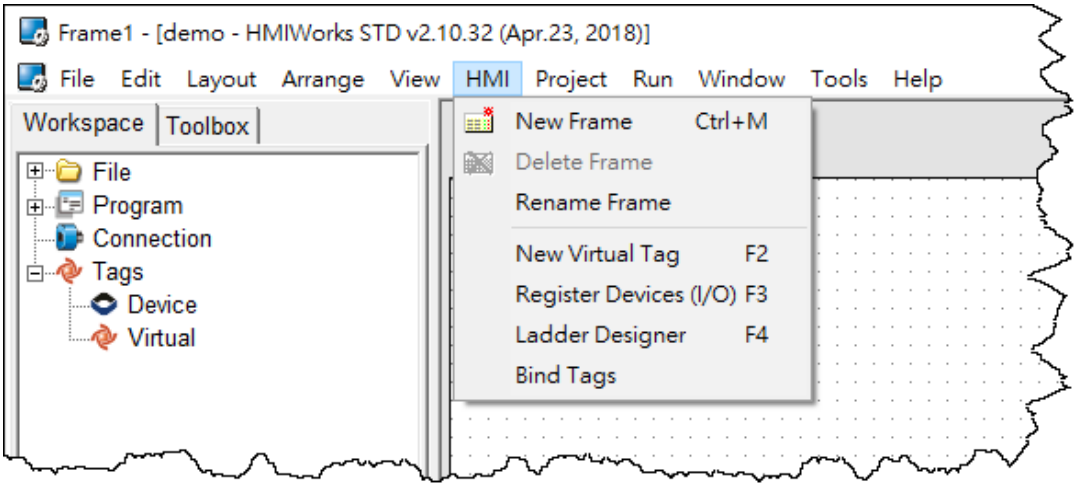
3.3 Ladder Designer

One of the most important features of HMIWorks is Ladder Designer. The ladder logic is defined by the followings:

1. A Ladder Diagram consists of many rungs.
2. Each rung resembles a circuit which is formed by relays.
3. All of the rungs are executed serially in a loop.

Click the **HMI** menu to use ladder diagram.

Note: Users can manage their ladder design in the "Workspace" panel.



The following is an overview of the functions contained in the **HMI** menu:

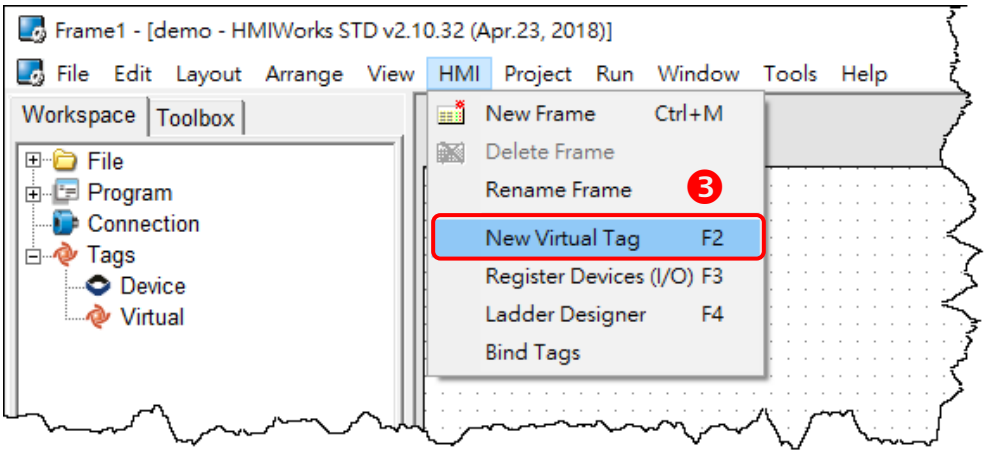
Option	Shortcut keys	Descriptions
New Frame	**Ctrl + M**	Add the new design frame.
Delete Frame		Delete a design frame.
Rename Frame		Rename the design frame.
New Virtual Tag	**F2**	Defines your own variables
Register Devices (I/O))	**F3**	Uses I/O devices of ICP DAS on the networks
Ladder Designer	**F4**	Designs your ladder logics
Bind Tags		Refer to Section 3.3.8 Data exchange for more detailed information.

3.3.1 Getting Started

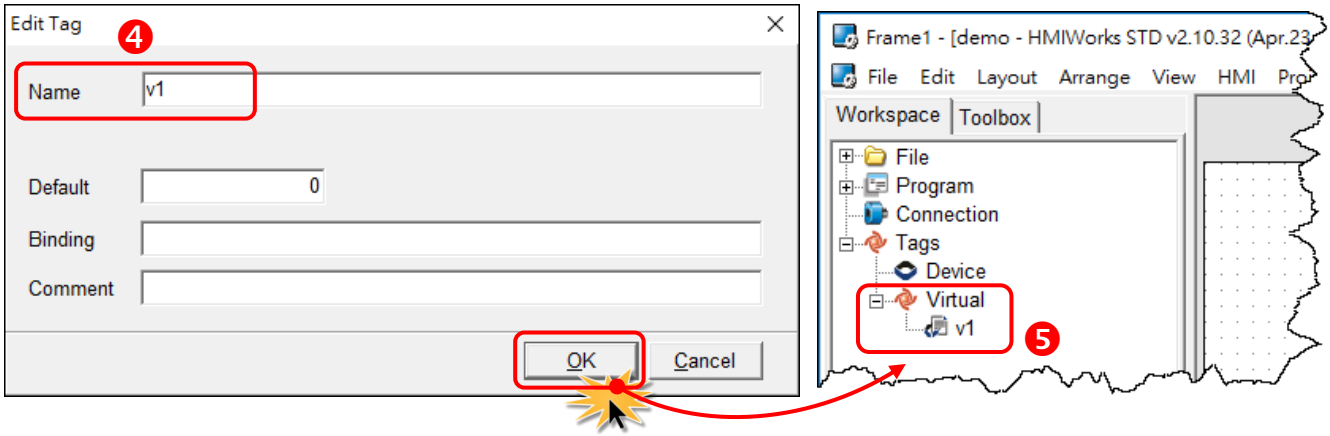
Step 1: Run **HMIWork_Standard.exe** and click the "**New Project**" icon to create a new project.
Step 2: In the **"New"** window, configure the parameters for the new project.

Refer to Section 3.1 The Construction of HMIWorks **for an illustration of how to perform the above steps.**

Step 3: Click the **"New Virtual Tag (F2)"** from the **"HMI"** menu to open the "Edit Tag" window.
or right click on the **"Virtual"** item and select the **"New Virtual Tag"** in the **"Workspace"** panel.

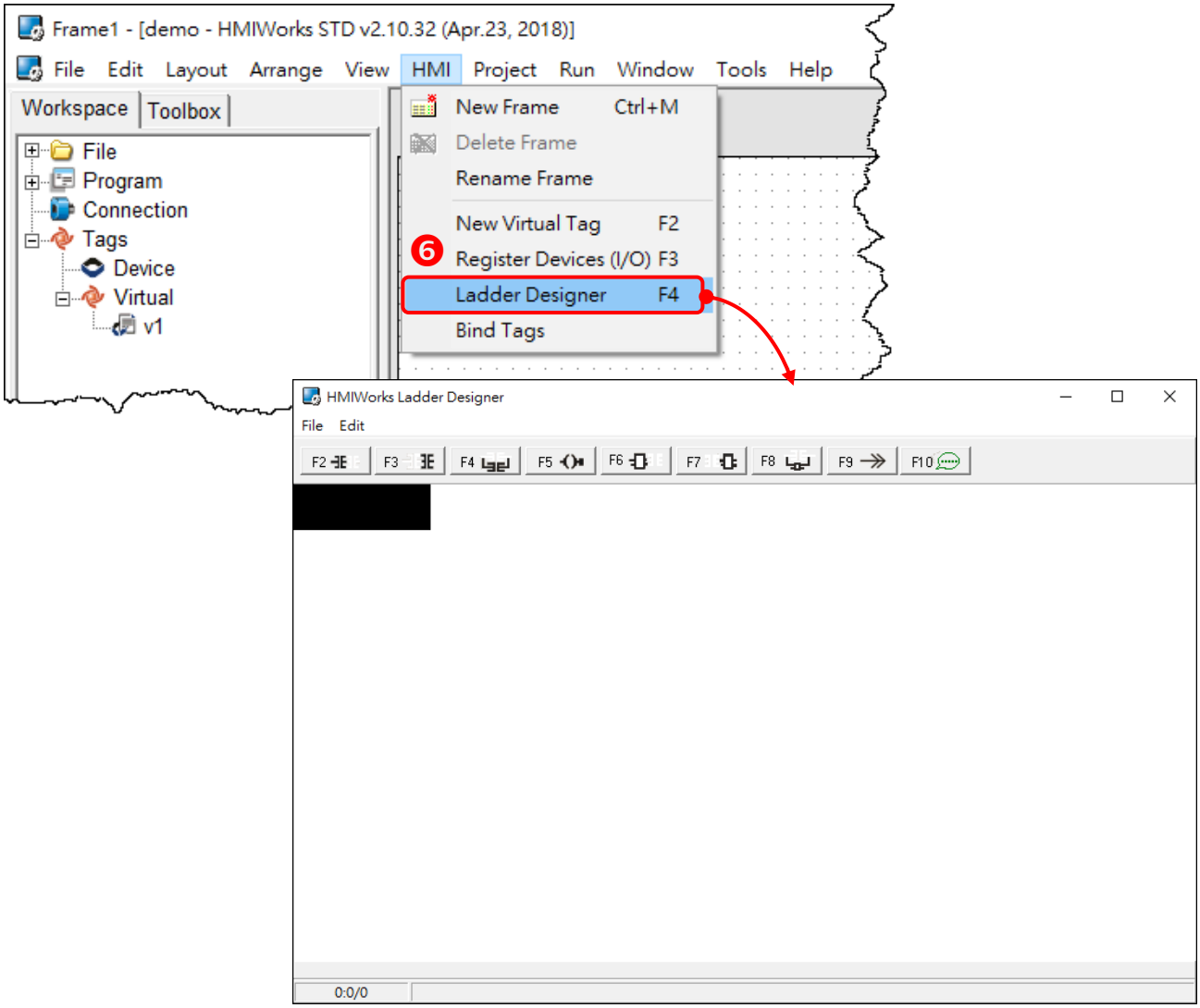


Step 4: Type a tag name (e.g., v1) in the **"Name"** field and click the **"OK"** button.
Step 5: The new tag (e.g., v1) is now shown under **"Virtual"** item in the **"Workspace"** panel.



Note: Refer to Section 3.3.3 Operating the Ladder Designer for more detailed information.

Step 6: Click the **"Ladder Designer (F4)"** from the **"HMI"** menu to open the "HMIWorks Ladder Designer" window. For detailed information about the interface, function block and operations of the Ladder Designer, refer to Section 3.3.2 Introduction and Section 3.3.3 Operating the Ladder Designer.

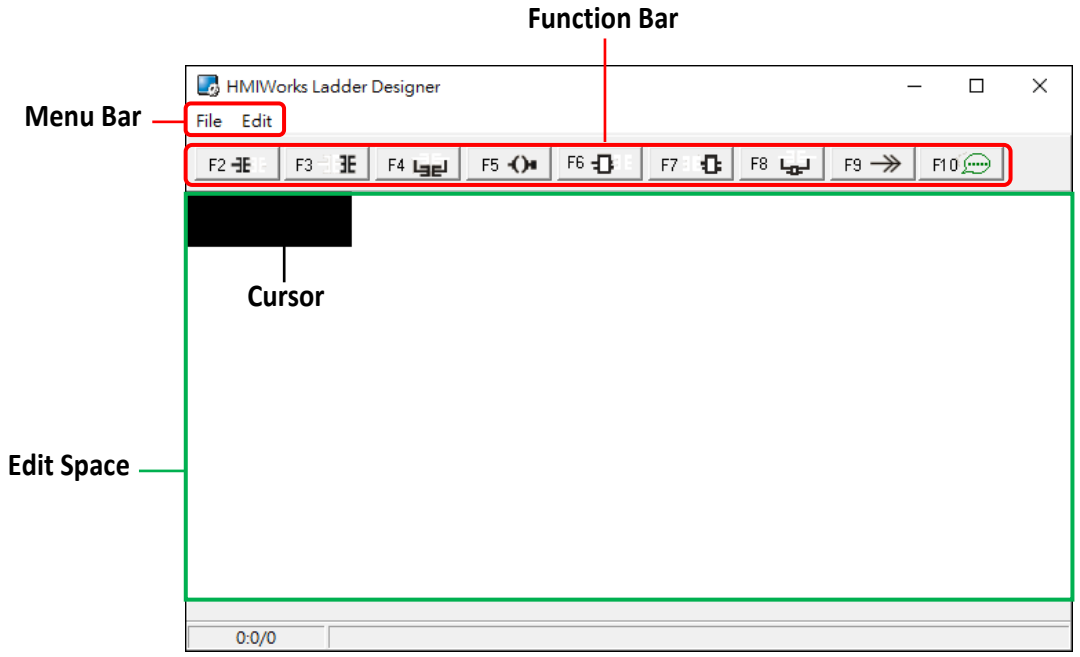


3.3.2 Introduction

This Section provides a basic overview of Ladder Designer interface, including the menu bar and function bar, etc., and function block definition.

3.3.2.1 Appearance

The Ladder Designer interface has been successfully opened in the Section 3.3.1 Getting Started.
A Ladder Designer is a tool to implement the ladder logic according to users' design. The Ladder Designer consists of four parts, the menu bar, the function bar, the edit space and the cursor, each of which will be described in more detail below.



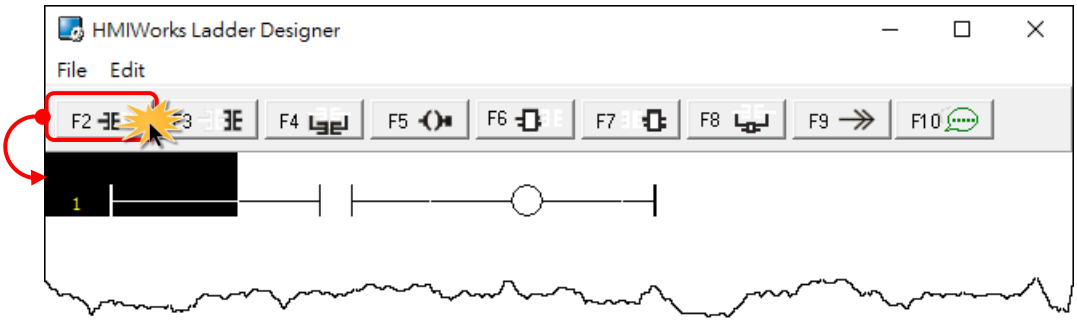
The briefings of the menu bar:

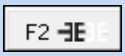
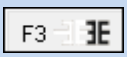
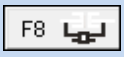
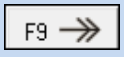
Option		Shortcut keys	Descriptions
File	New	Ctrl + N	Create a new Ladder Designer file.
	Open	Ctrl + O	Pick an existing Ladder Designer file to load.
	Save	Ctrl + S	Save the Ladder Designer file.
	Save as...	Ctrl + A	Save the Ladder Designer file under a new filename.
	Save & Close	Ctrl + K	Save the Ladder Designer file then close the window.
	Exit	Ctrl + X	Exiting the Ladder Designer window.

Option			Shortcut keys	Descriptions
Edit	New Rung	Insert Before	Ctrl + I	Insert a Rung up.
		Insert After	Ctrl + M	Insert a Rung down
	Duplicate		Ctrl + D	Copy and paste the selected Rung.
	Copy		Ctrl + C	Copy selected Rung to the clipboard.
	Paste		Ctrl + V	Paste a copy from the clipboard.

The briefings of the function bar:

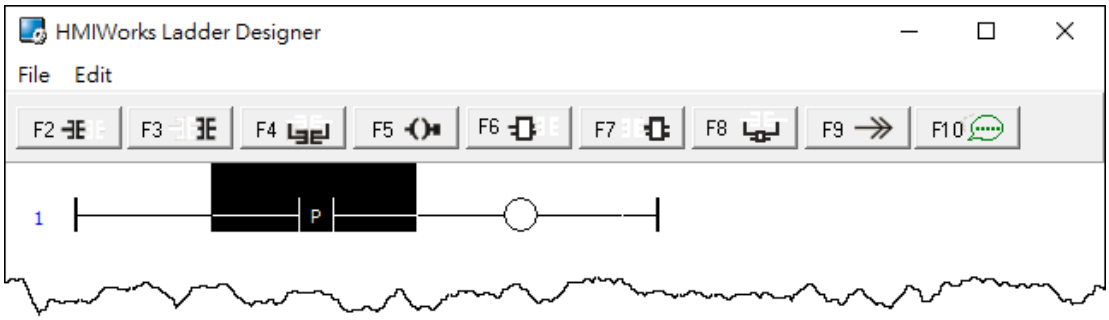
For example: Click the  button to create a contact input in the edit space, as shown below.



Option	Shortcut keys	Descriptions
	F2	Insert a contact input in the left of the cursor.
	F3	Insert a contact input in the right of the cursor.
	F4	Insert a contact input which is parallel to the cursor.
	F5	Insert a coil output.
	F6	Insert a function block in the left of the cursor.
	F7	Insert a function block in the right of the cursor.
	F8	Insert a function block which is parallel to the cursor.
	F9	Insert a Jump which is parallel to the cursor.
	F10	Add comments.

The briefings of the contact input type:

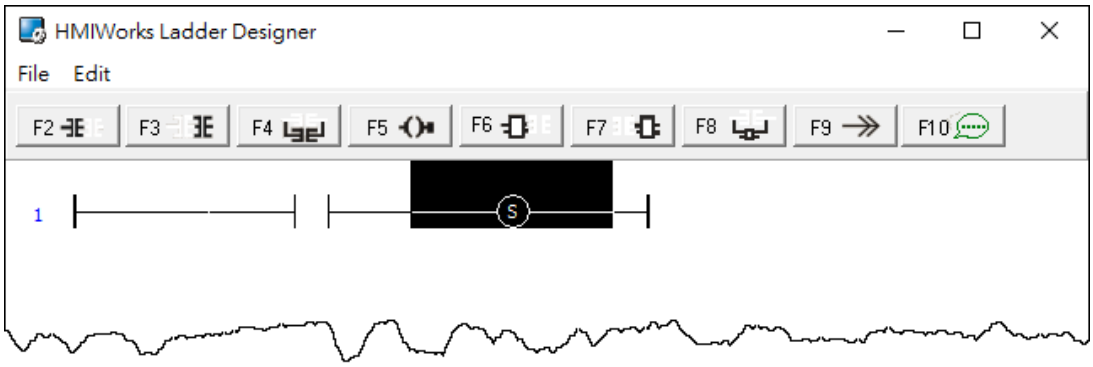
For example: Click the contact input to select it and press **<P>** key (or press **<Spacebar>** key continuously to adjust input type), as shown below.



Option	Select Key	Descriptions
	**Spacebar**	A normally-open contact input.
	**Spacebar or **	A normally-closed contact input.
	**Spacebar or P**	A positive transition contact input. When the state from OFF to ON, trigger one shot.
	**Spacebar or N**	A negative transition contact input. When the state from ON to OFF, trigger one shot.

The briefings of the coil output type:

For example: Click the coil output to select it and press **<S>** key (or press the **<Spacebar>** key continuously to adjust output type), as shown below.



Option	Select Key	Descriptions
	**Spacebar**	A normally-open coil output.
	**Spacebar or **	A normally-closed coil output.
	**Spacebar or S**	A “Set” coil output. Once triggered, the coil remains ON until a reset.
	**Spacebar or R**	A “Reset” coil output. Once triggered, the coil remains OFF until a set.
	**Spacebar or P**	A positive transition coil output. When the state from OFF to ON, trigger one shot.
	**Spacebar or N**	A negative transition coil output. When the state from ON to OFF, trigger one shot.

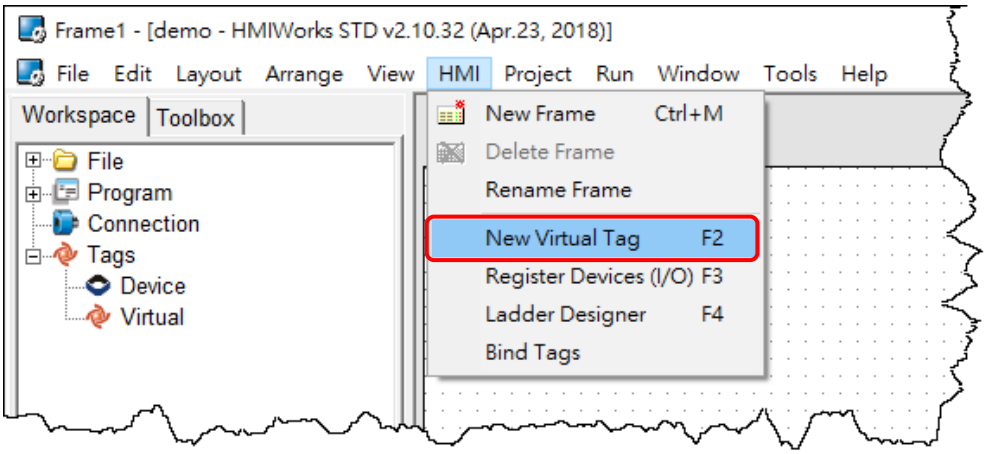
3.3.3 Operating the Ladder Designer

This Section provides a basic overview of how to use the Ladder Designer.

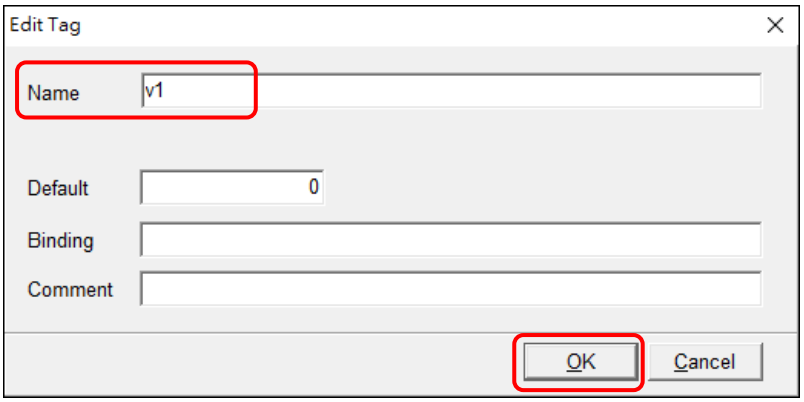
3.3.3.1 Add the New Virtual Tags (F2)

To use the **Ladder Designer**, add tags for the **Ladder Designer** first, as following the procedure described below:

Step 1: Click the **"New Virtual Tag (F2)"** from the **"HMI"** menu to open the "Edit Tag" window. or right click on the **"Virtual"** item and select the **"New Virtual Tag"** in the **"Workspace"** panel.



Step 2: Define a new tag in the **"Name"** field and optionally fill the other fields.
Step 3: Finally, click the "**OK"** button to take effect.

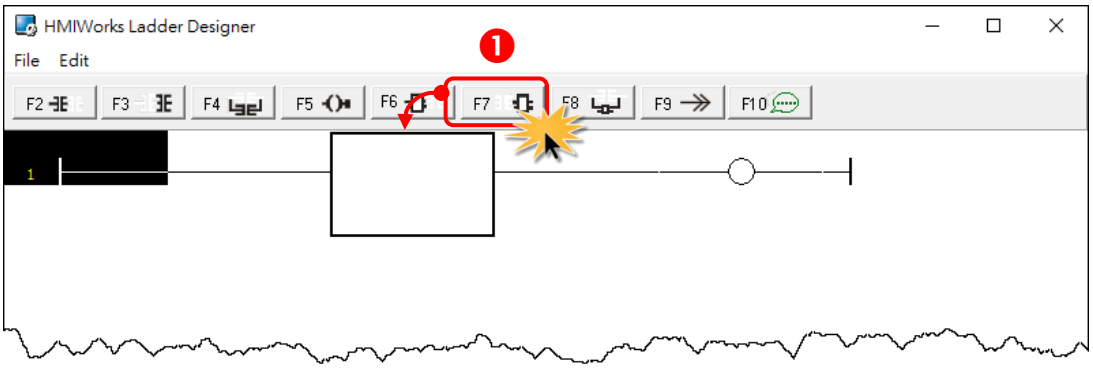


Here, we add three variable v1, v2 and v3 for example in the next sections.

3.3.3.2 Assigning Tags and Constants

The following description of the **math formula: v3 = 1 + 2 and v2 = v1** are used as an example.

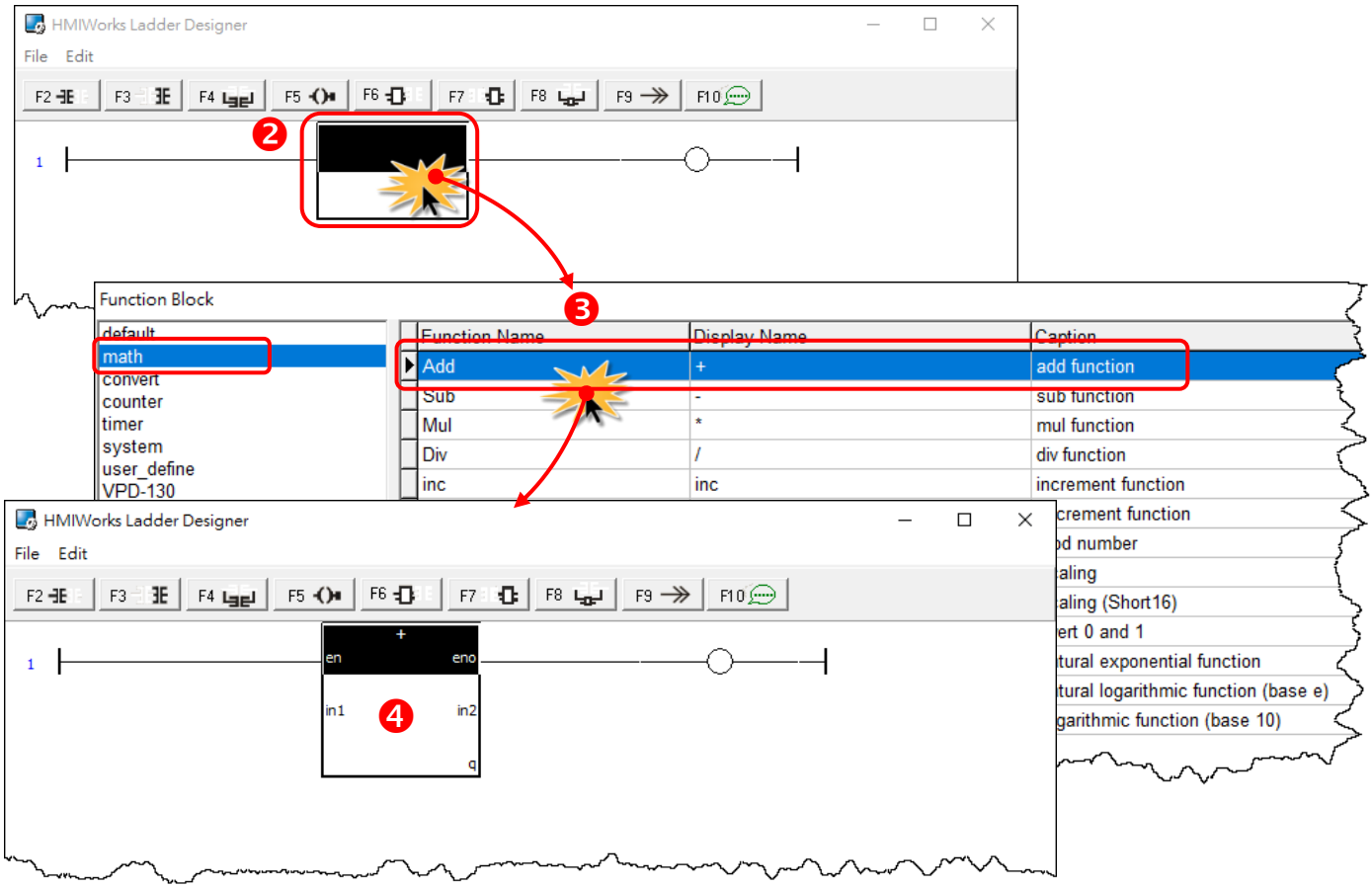
Step 1: Press **<F4>** key to open the **Ladder Designer**, click the F7 button to create a function block.



Step 2: Double-click it to open "Function Block" window.

Step 3: Click the **"math"** item and double-click the **"Add"** function.

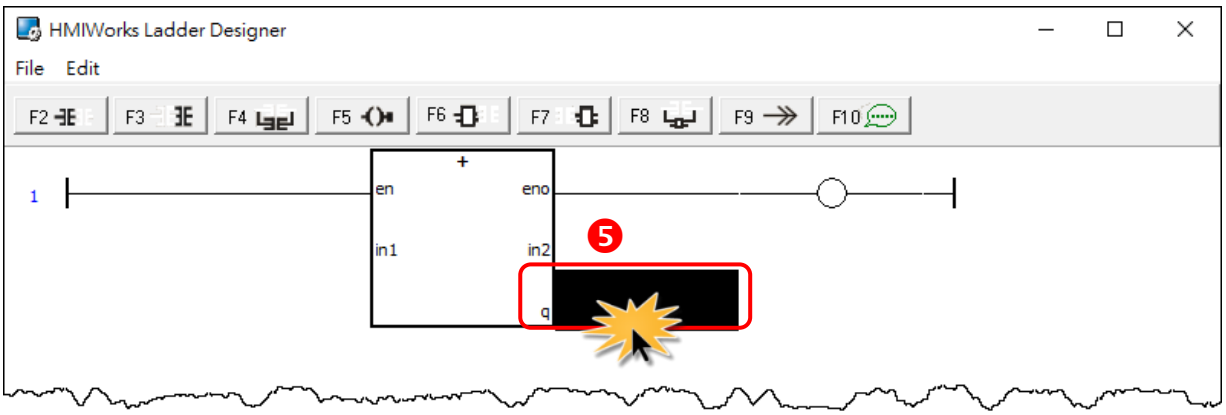
Step 4: Open the **"Add"** function block.



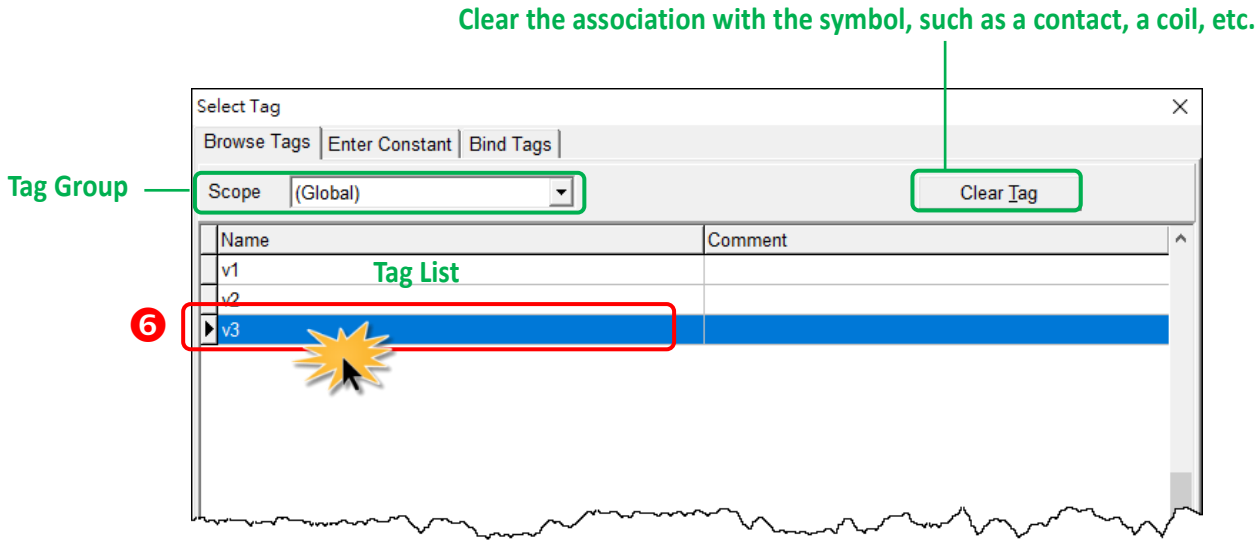
Browse Tags and Enter Constant

For example: Math Formula: v3 = 1 + 2

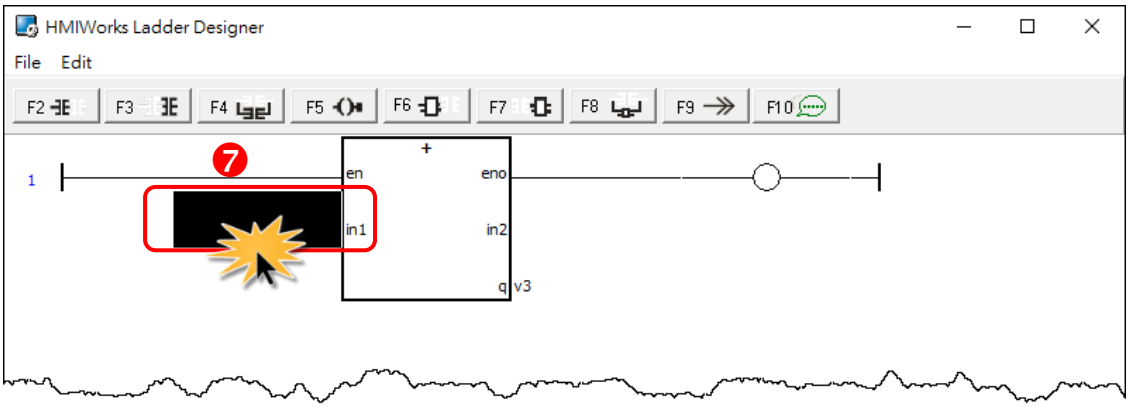
Step 5: Double-click on the **"q" symbol** to open the "**Select Tag**" window.



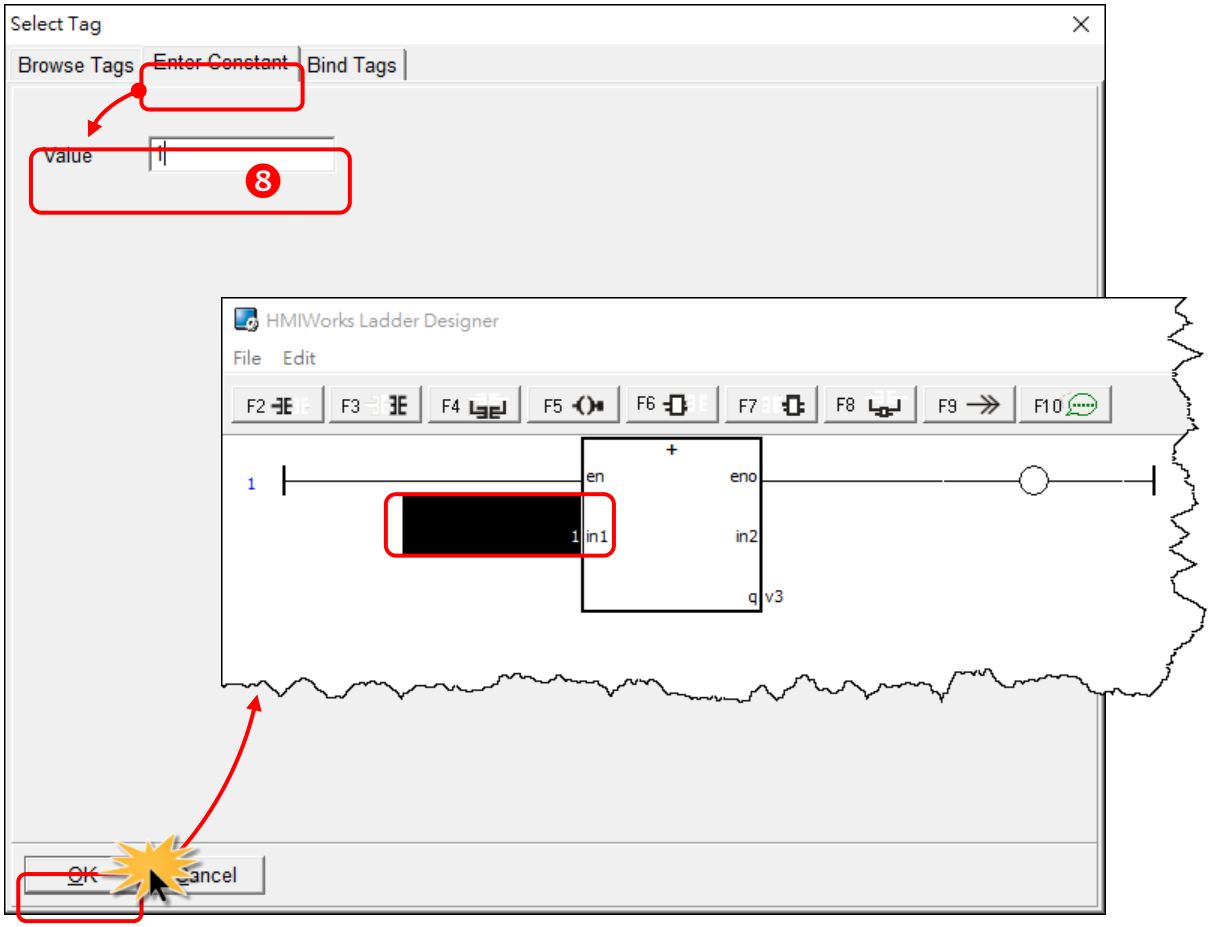
Step 6: Double-click the **"v3"** to select it in the **"Browse Tags"** tab.



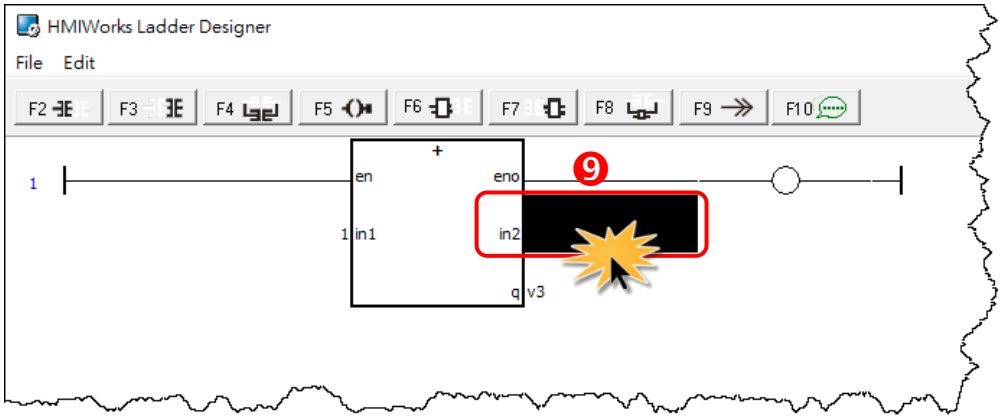
Step 7: Double-click on the **“in1” symbol** to open the “**Select Tag**” window.



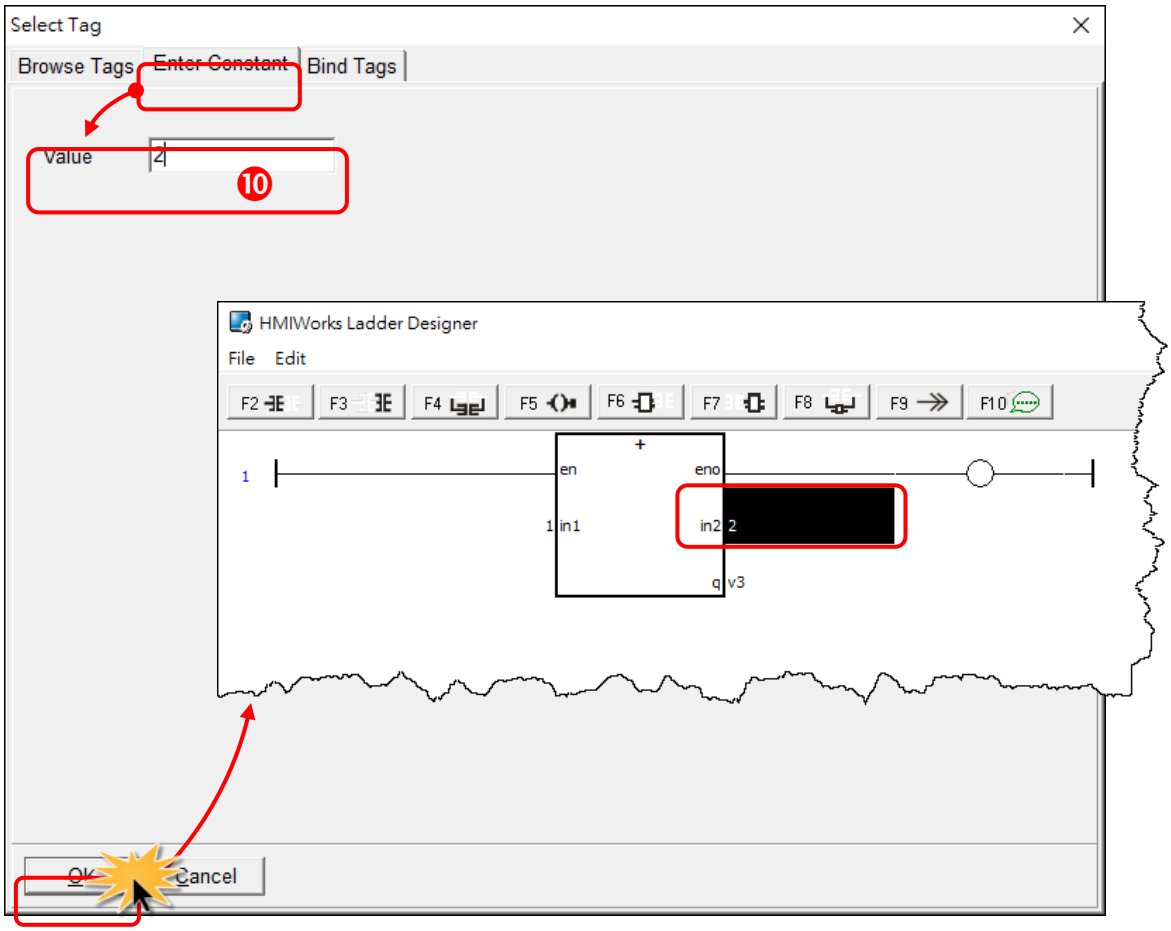
Step 8: Click the **“Enter Constant”** tab and type the **“1”** in **“Value”** filed then click the **“OK”** button.



Step 9: Double-click on the **“in2” symbol** to open the “**Select Tag**” window.



Step 10: Click the **“Enter Constant”** tab and type the **“2”** in **“Value”** filed then click the **“OK”** button.

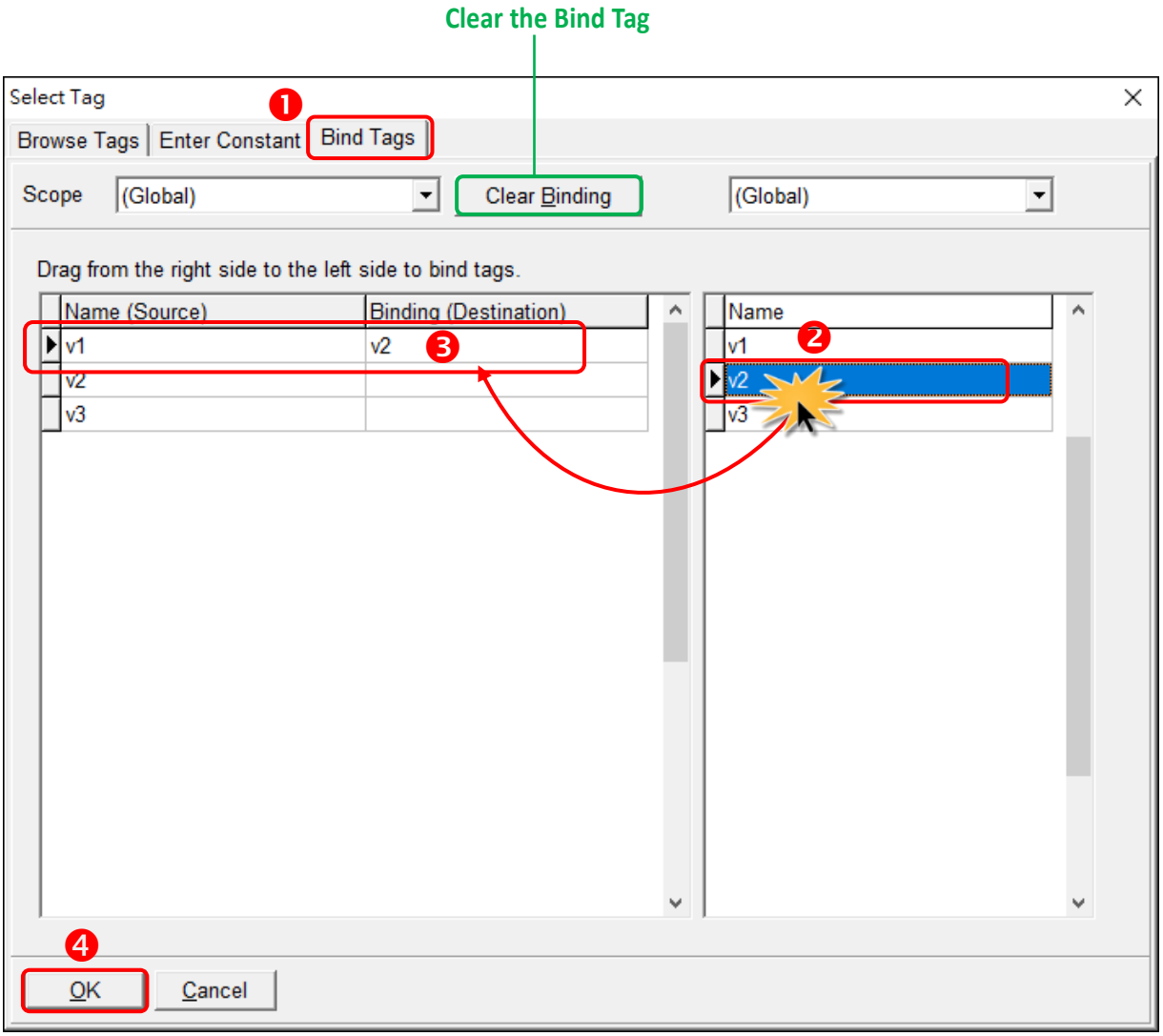


Bind Tags

For example: v2 = v1

Drag from the tag of right side to the tag of left side to bind tags. For example, when v2 drag to v1, if v1 changed, then v2 = v1. For detail application can refer to Section 3.3.8 Data exchange.

Step 1: Click the **"Bind Tags"** tab.
Step 2: Click the **"v2"** in the right-hand tag list.
Step 3: **Drag to "Binding (Destination)" field of v1** in the left-hand tag list.
Step 4: Click the **"OK"** button.

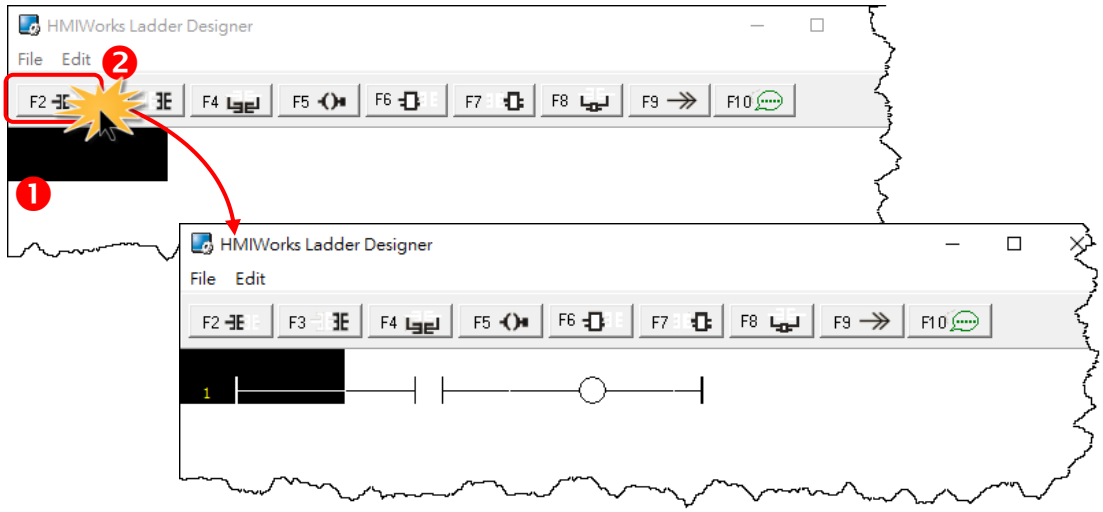


3.3.3.3 Inserting and Deleting a Rung

Insert a rung:

Step 1: Move the cursor (the highlighted area) to the empty place.

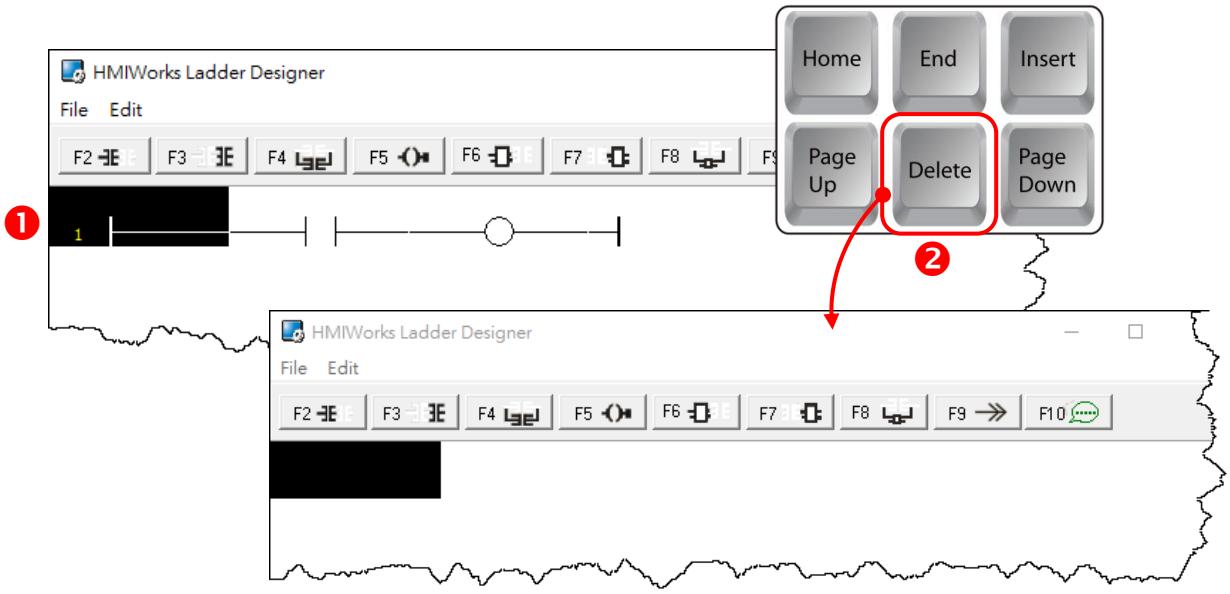
Step 2: Click the F2 button (or press **<F2>** key) to insert a rung.



Delete a rung:

Step 1: Move the cursor to the starting point of the rung.

Step 2: Press **<Delete>** key to delete a rung.



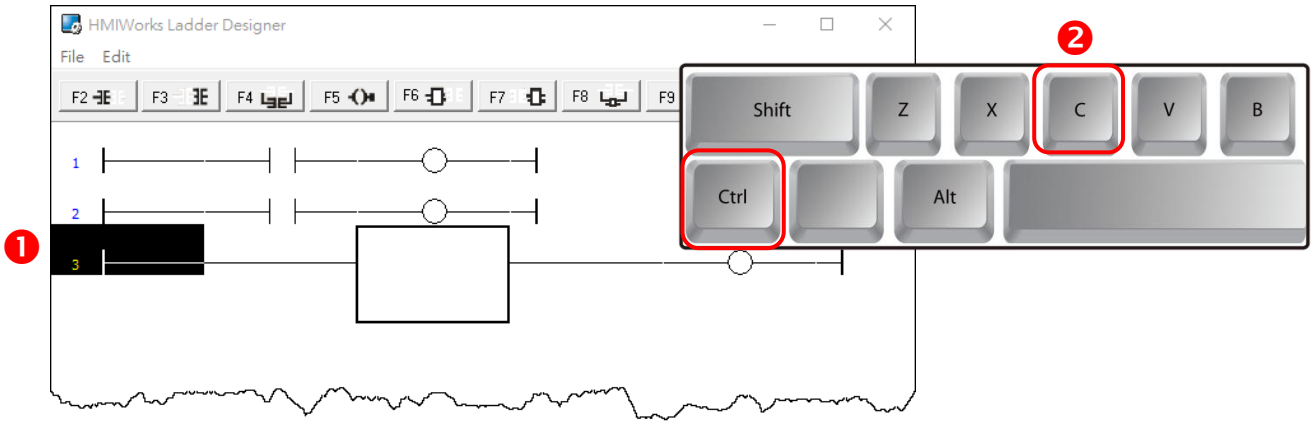
3.3.3.4 Copying and Pasting a Rung

Supposed that we have three rungs and we want to copy the third rung and insert it between the first and the second rungs.

Copy a rung:

Step 1: Move the cursor to the third rung.

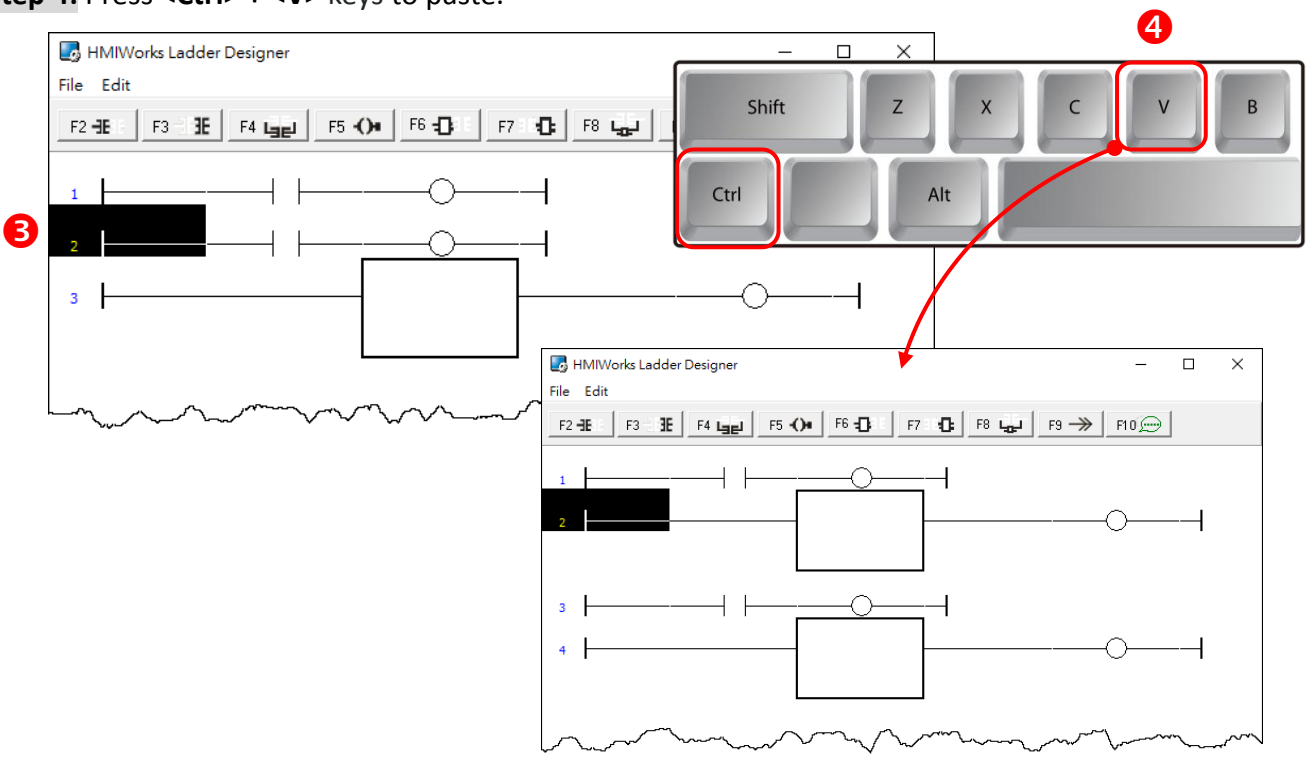
Step 2: Press **<Ctrl> + <C>** keys to copy a rung.



Paste a rung:

Step 3: Move the cursor to the second rung.

Step 4: Press **<Ctrl> + <V>** keys to paste.

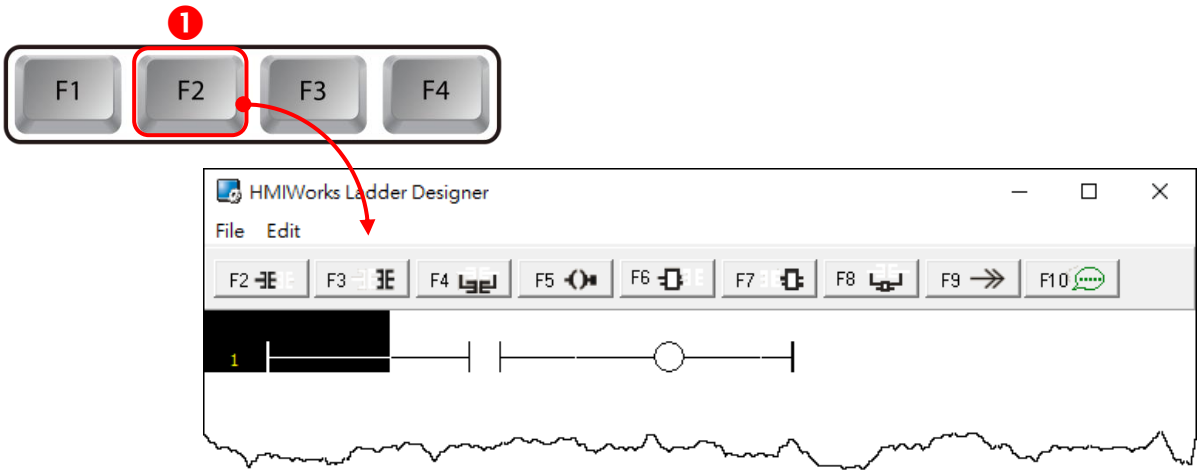


3.3.3.5 Inserting and Deleting a Contact Input

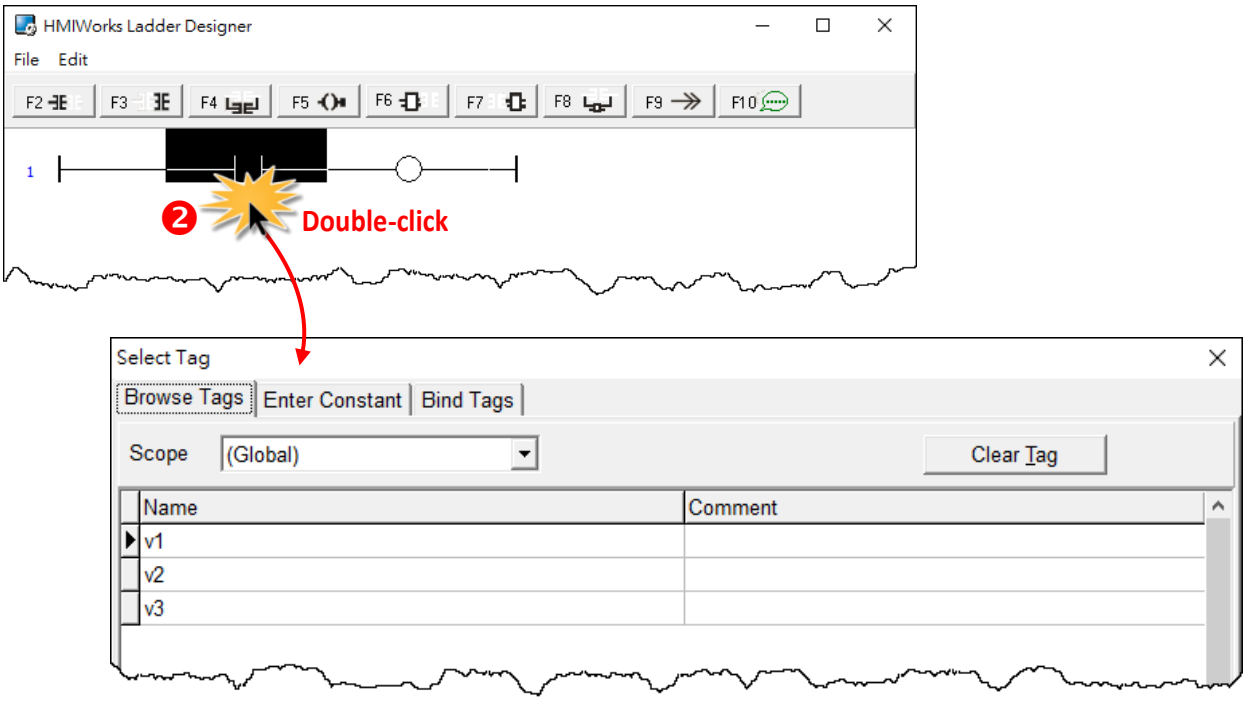
To demonstrate how to insert or delete a contact input and other related issues, go through the steps below.

Associate a Tag to a contact input:

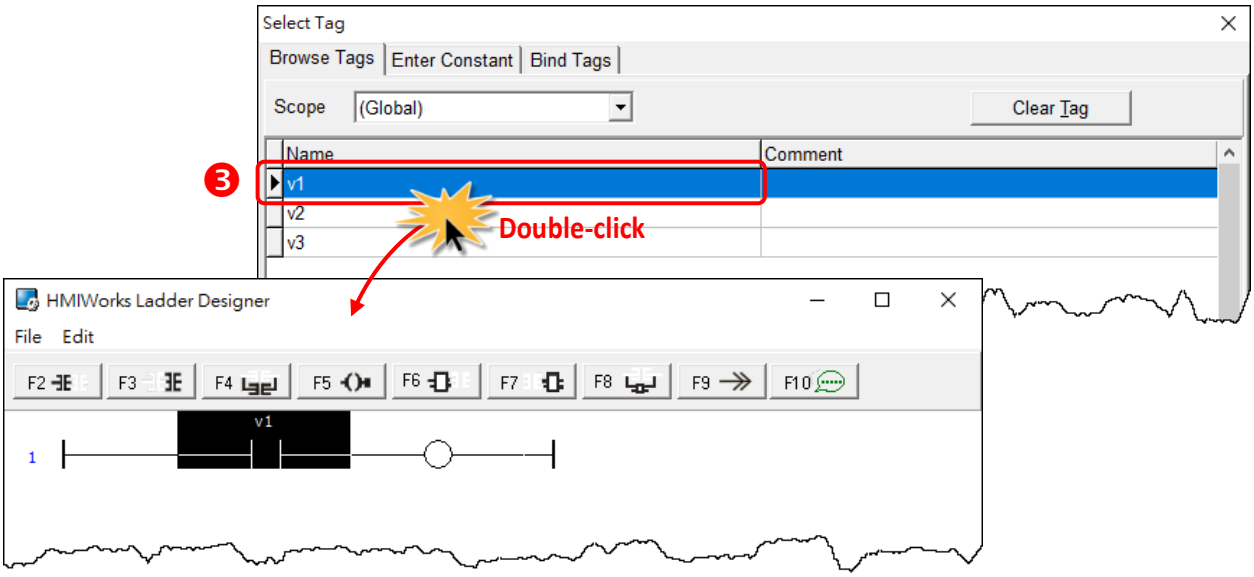
Step 1: Press **<F2>** key to insert a new rung with a contact input and a coil output.



Step 2: In the new rung, double-click on the contact input to open the "**Select Tag**" window to select a tag and assign it to the contact input.



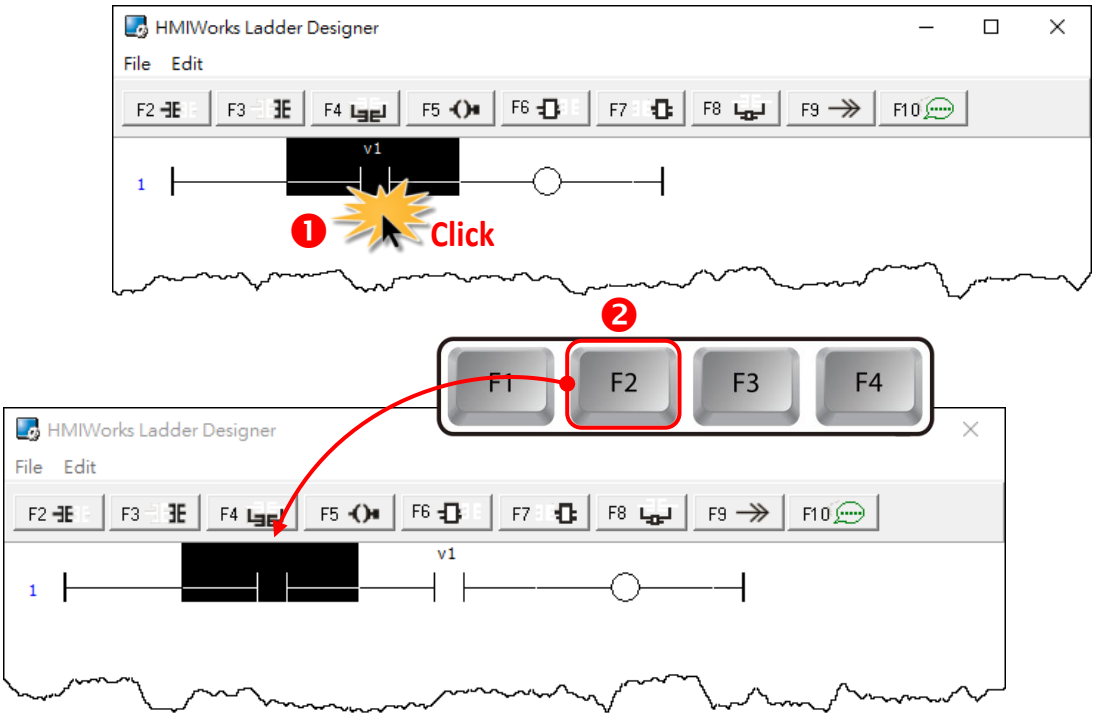
Step 3: Double-click on the **"v1"** tag and set to the contact input.



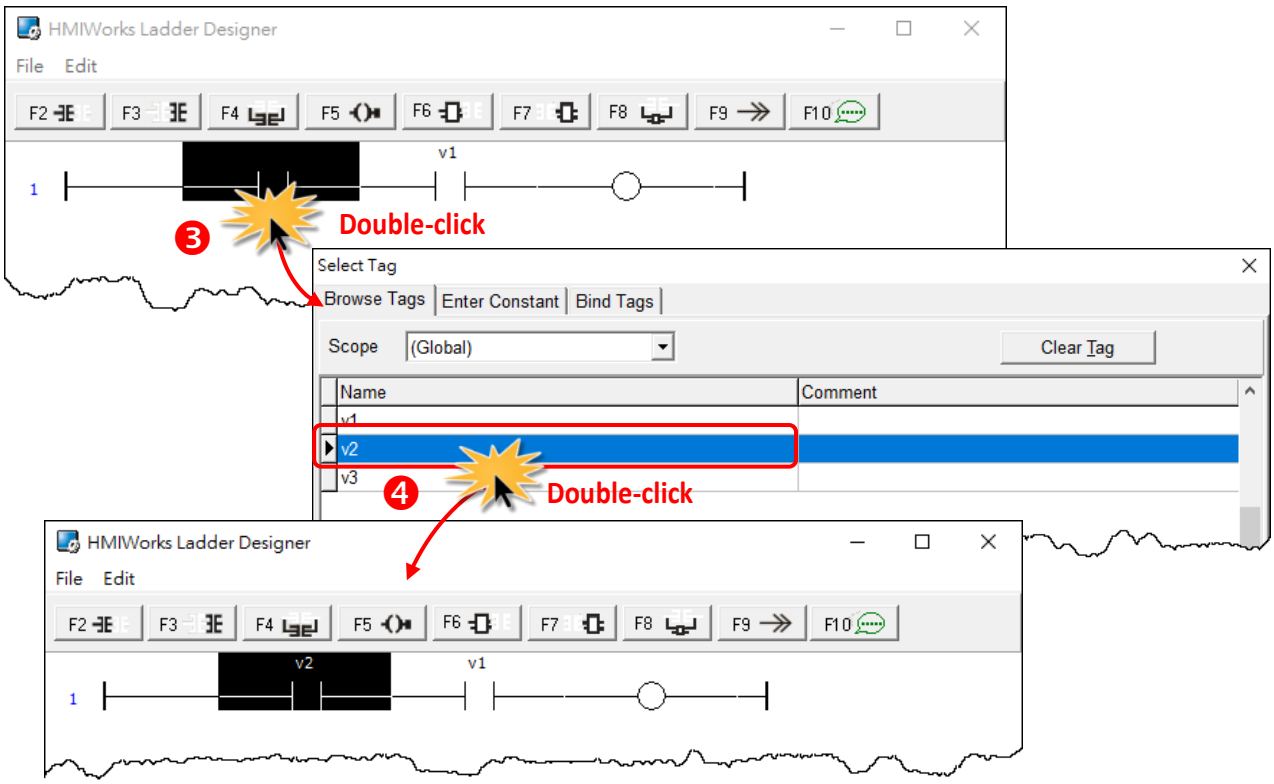
Insert a new contact input in the left of the cursor (F2)

Step 1: Move the cursor to the **"v1"** contact input.

Step 2: Press <**F2**> key.



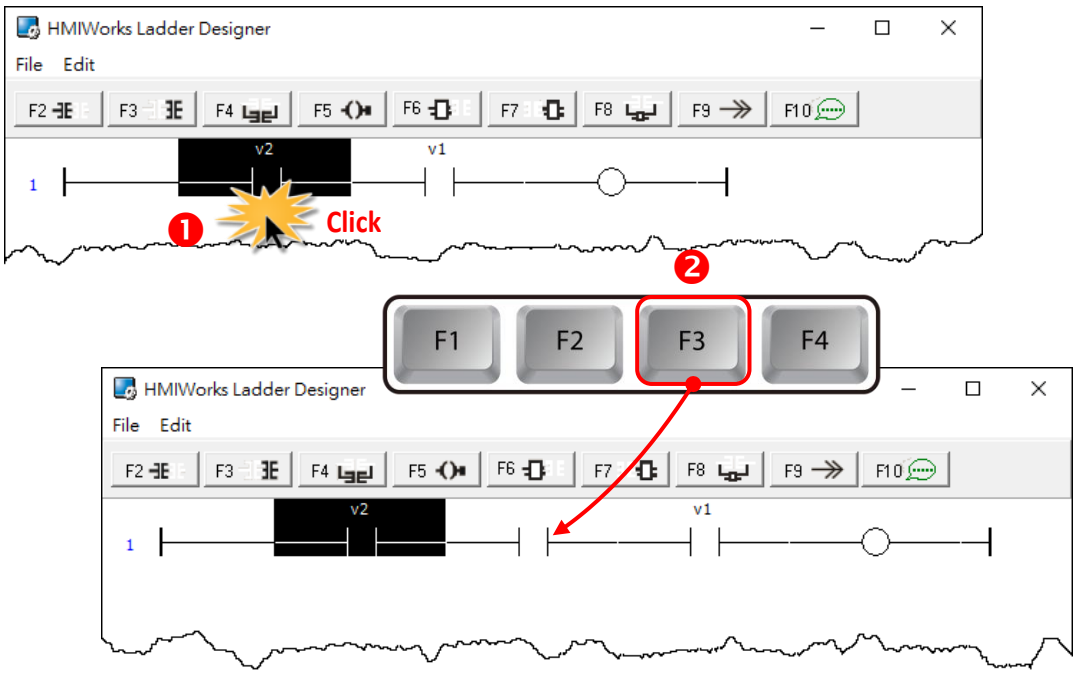
Step 3: Associate tag **"v2"** to the newly-inserted contact input.



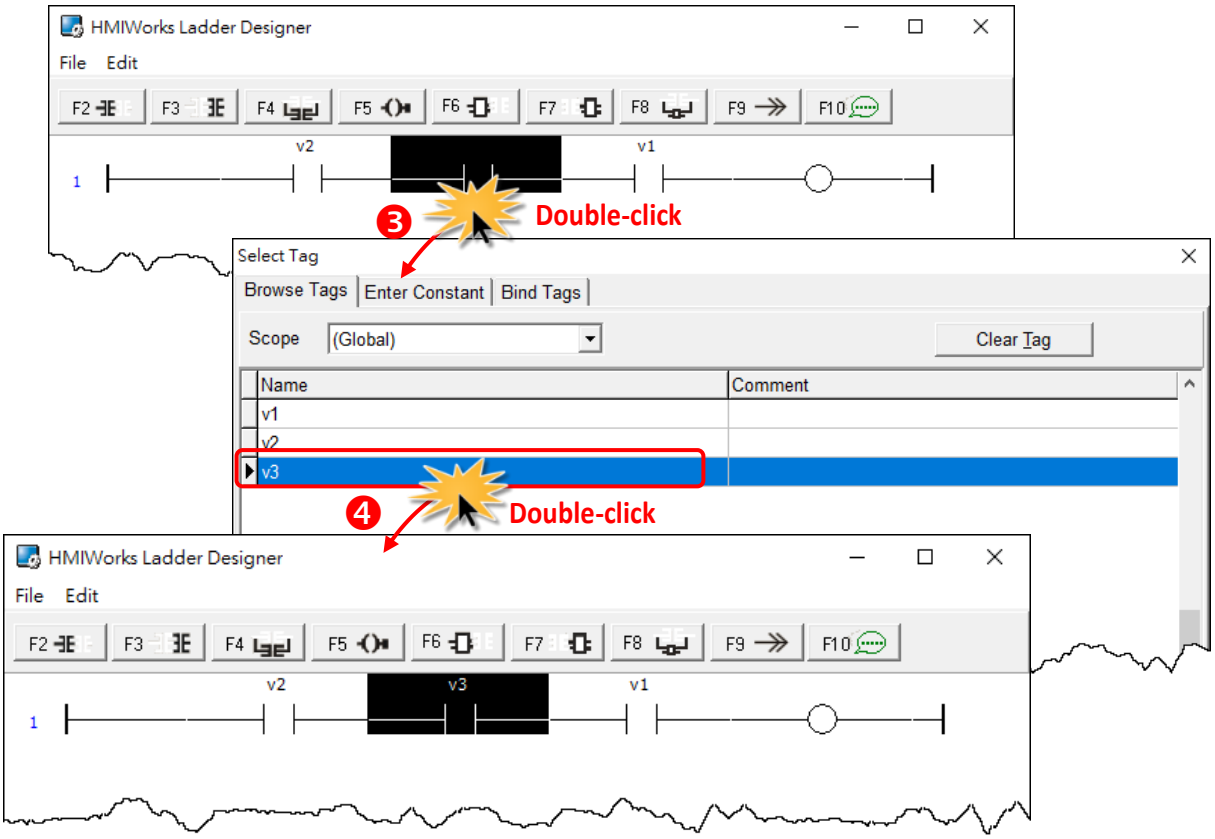
Insert a new contact input in the right of the cursor (F3)

Step 1: Move the cursor to the **"v2"** contact input.

Step 2: Press <**F3**> key.



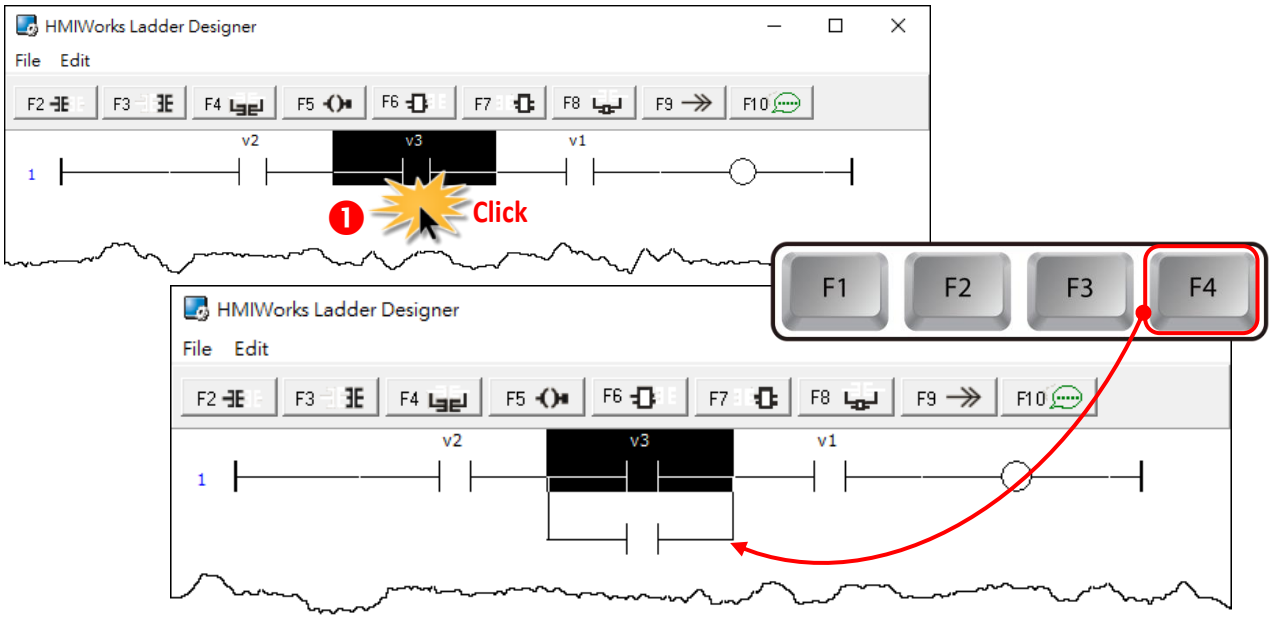
Step 3: Associate tag **"v3"** to the newly-inserted contact input.



Insert a new contact input which is parallel to the cursor (F4)

Step 1: Move the cursor to the **"v3"** contact input.

Step 2: Press **<F4>** key.

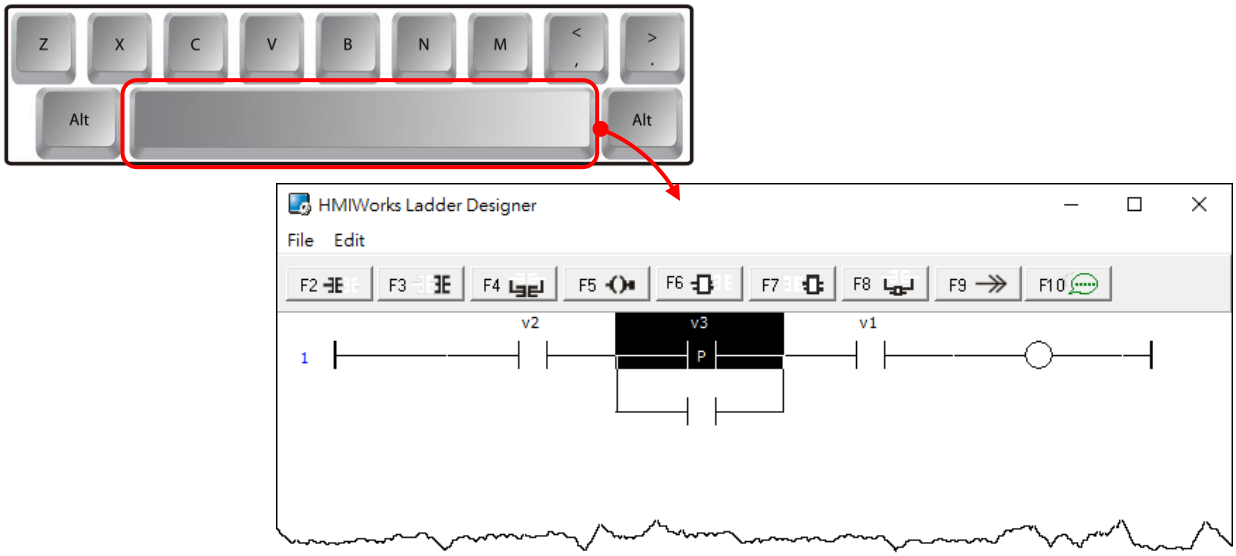


Set the type of a contact input

Move the cursor to a contact input and then press the **"Spacebar"** to change the type of the contact input.

Step 1: We move the cursor to the **"v3"** contact input.

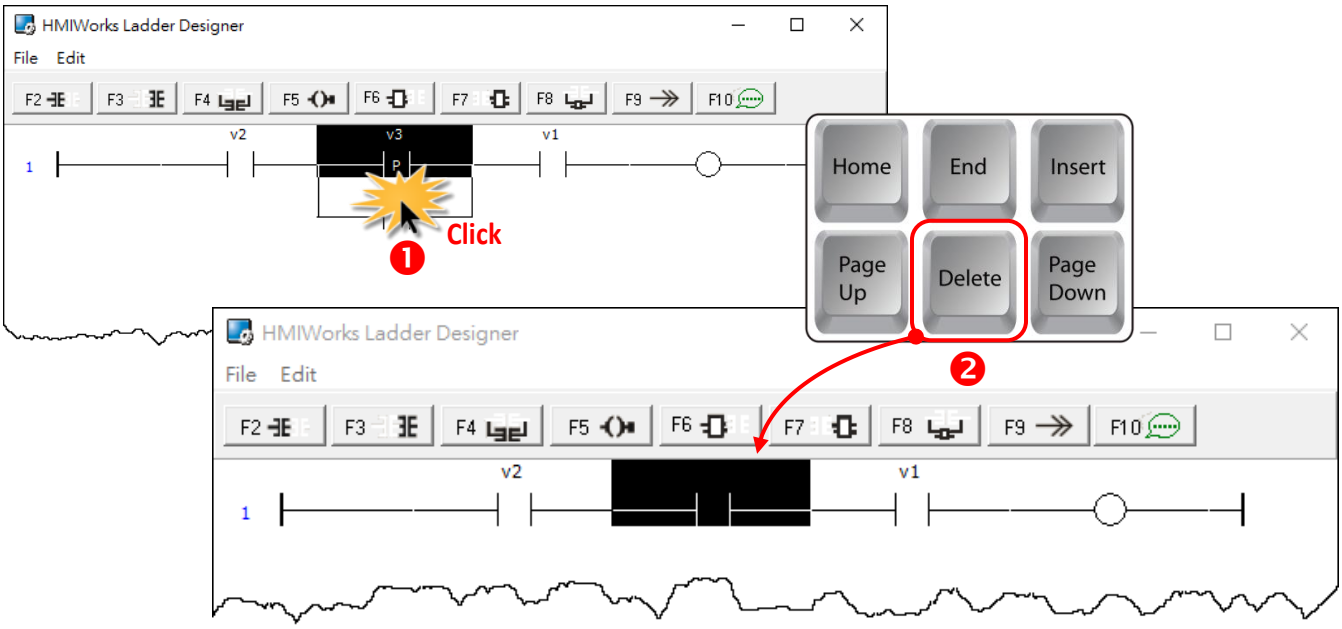
Step 2: Press **<Spacebar>** key twice to set the type of the contact input to pulse contact input.



Delete a contact input in the rung

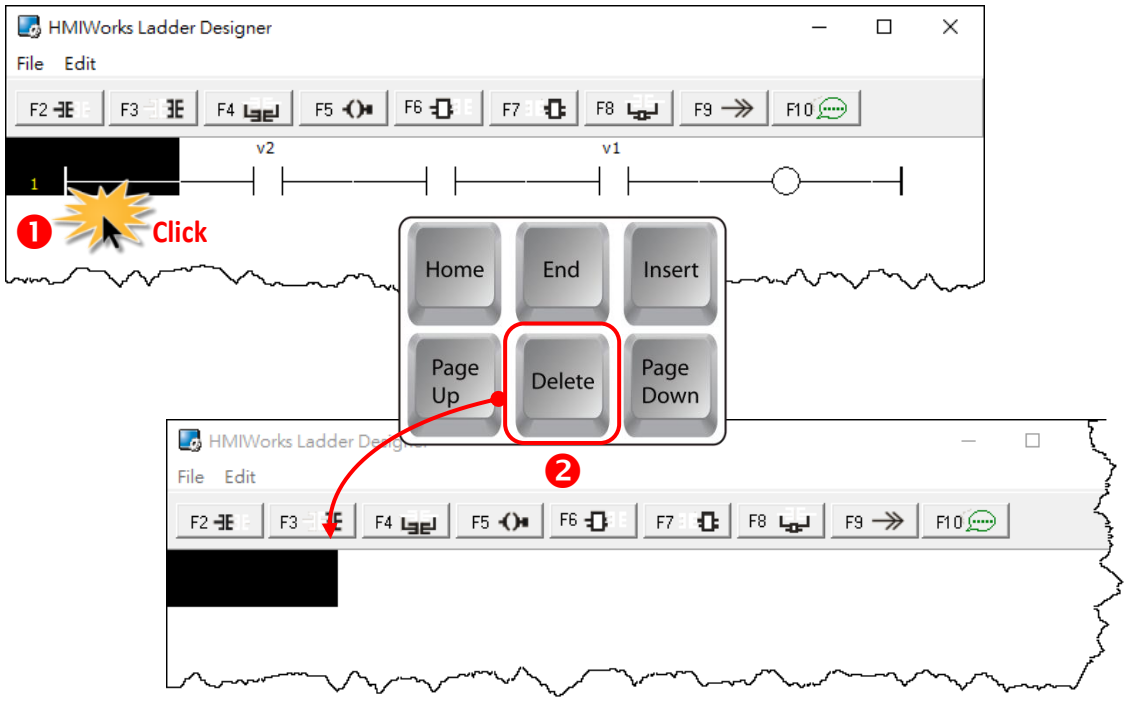
Move the cursor to the contact input you want to delete and press **<Delete>** key.

For example, we move the cursor to the **"v3"** contact input and press the **<Delete>** key.



Delete the rung

Move the cursor to the **starting point** of the rung and press **<Delete>** key.

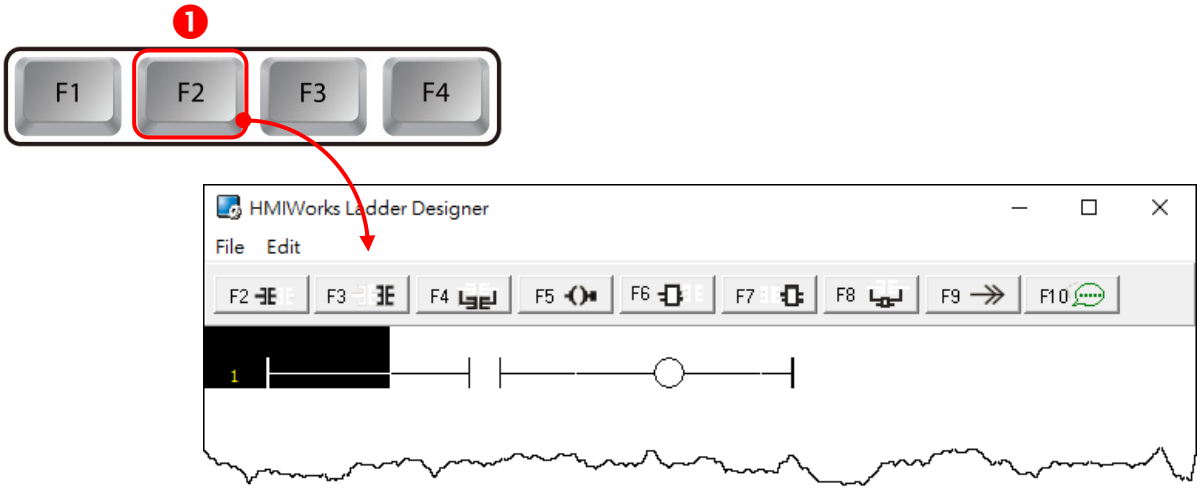


3.3.3.6 Inserting and Deleting a Coil Output

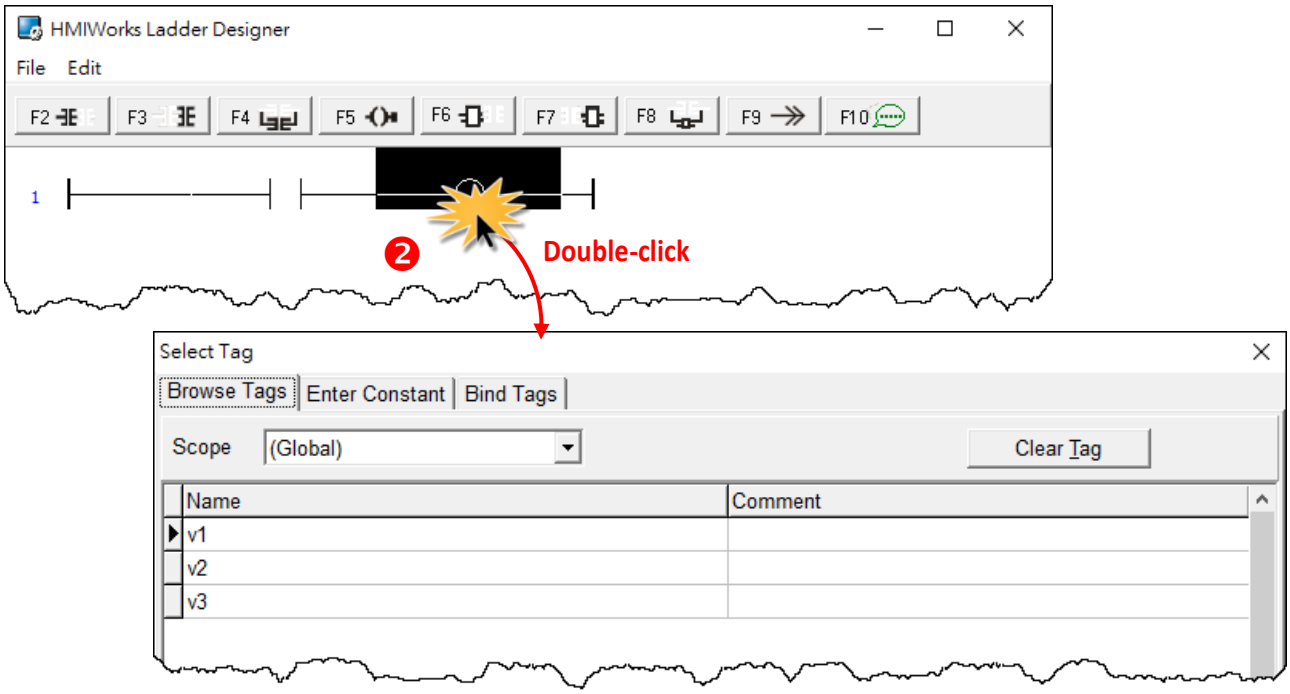
To demonstrate how to insert or delete a coil output and other related issues, see the figure below.

Associate a Tag to a Coil Output:

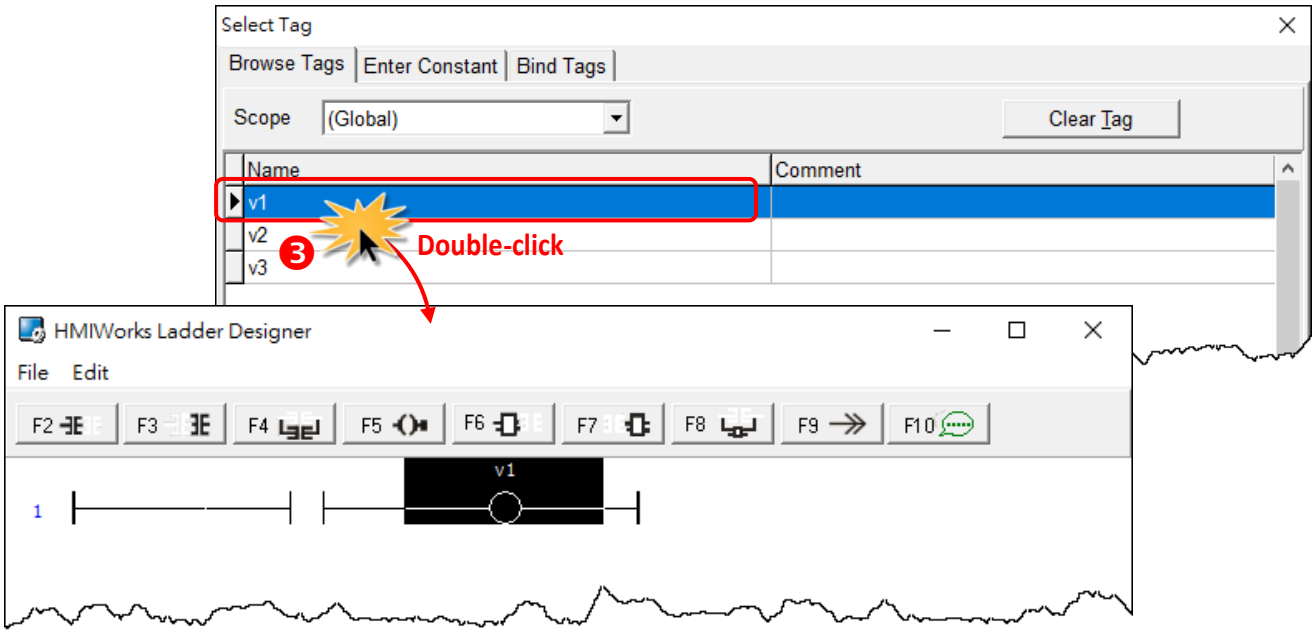
Step 1: Press **<F2>** key to insert a new rung with a contact input and a coil output.



Step 2: Double-click on the coil to open the **"Select Tag"** window.

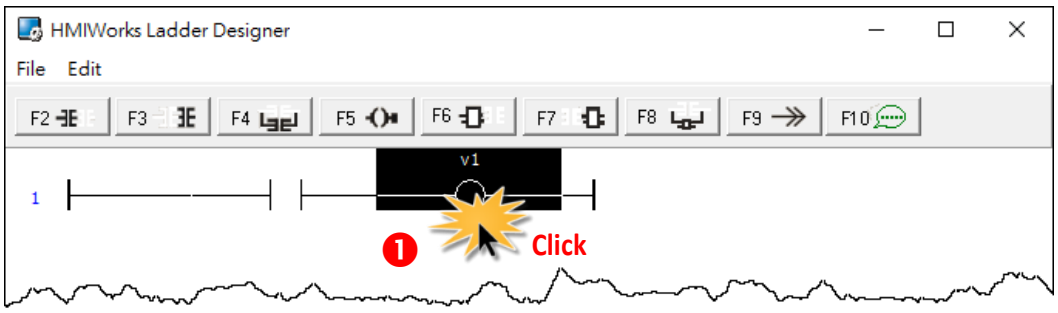


Step 3: Associate the tag **"v1"** to the coil.

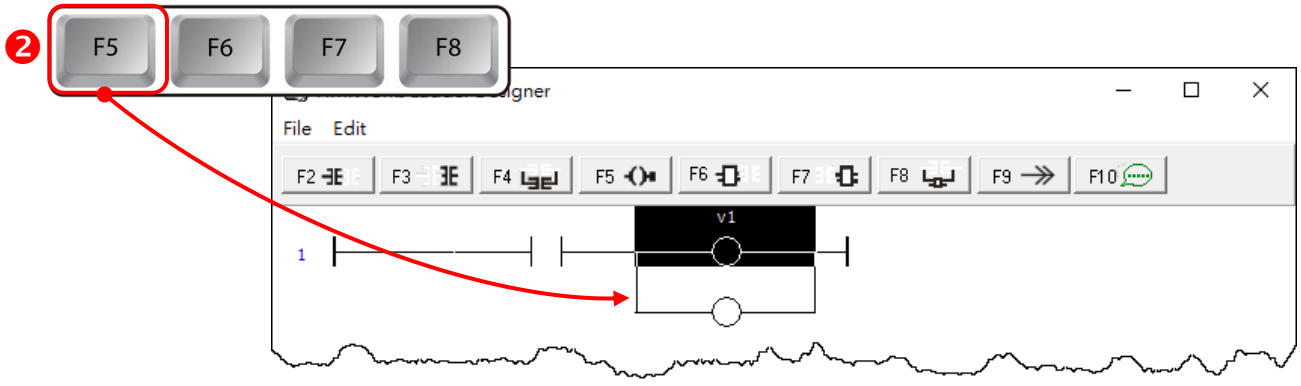


Insert a new Coil Output which is parallel to the cursor (F5)

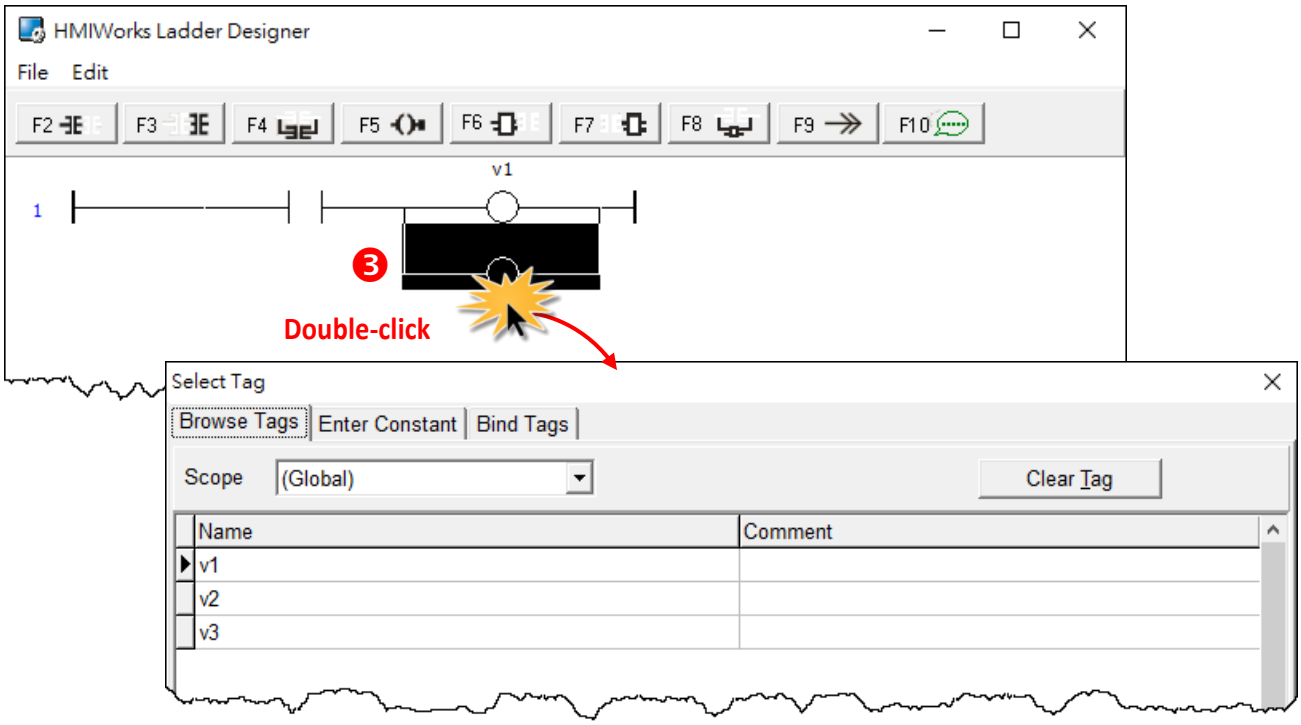
Step 1: Move the cursor to the coil **"v1"**



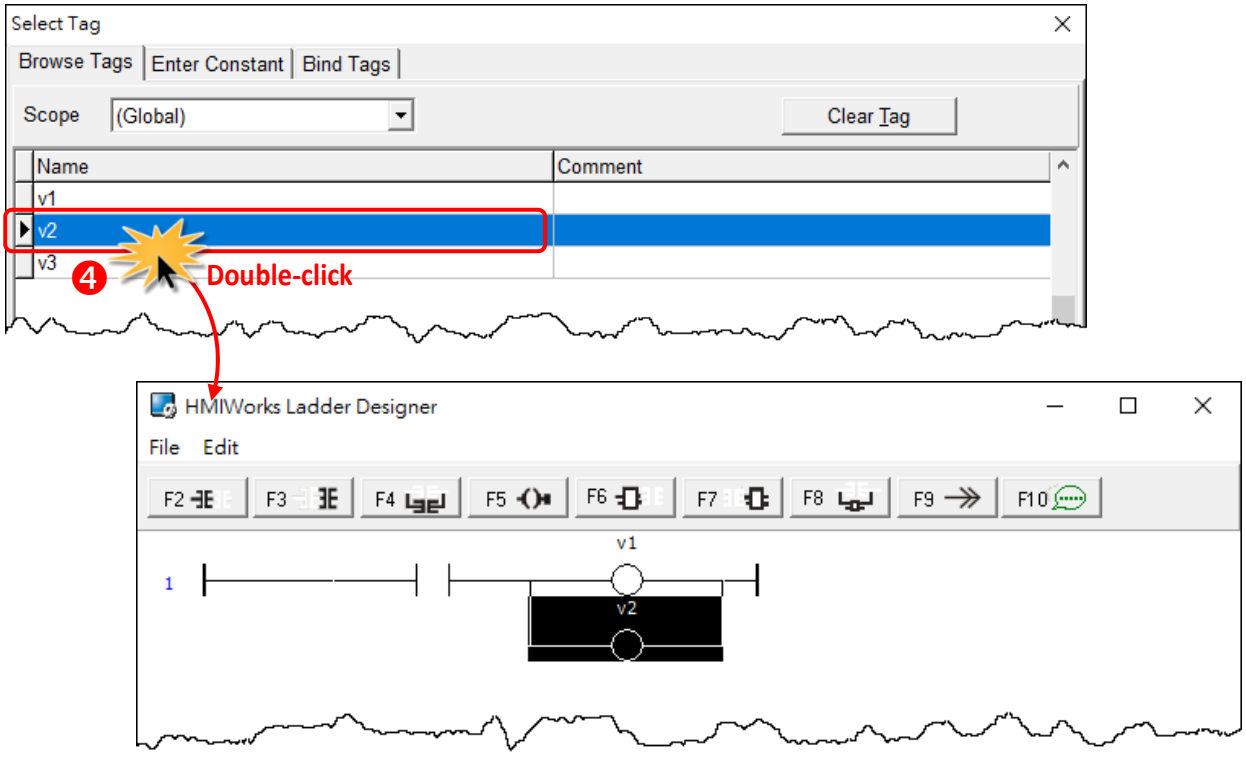
Step 2: Press **<F5>** key to insert a new parallel coil.



Step 3: Double-click on the coil to open the **"Select Tag"** window.



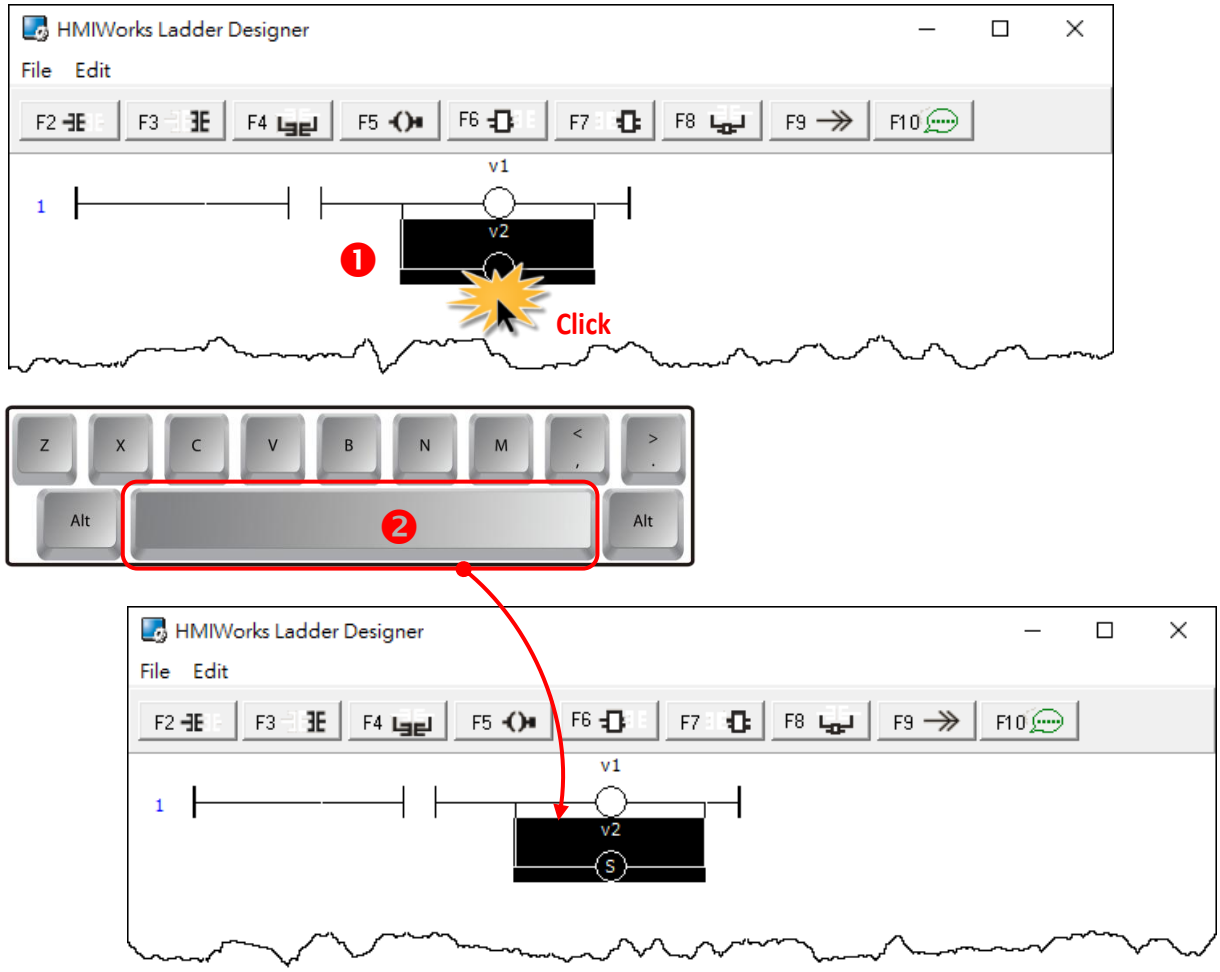
Step 4: Associate the tag **"v2"** to the coil.



Set the type of a Coil Output

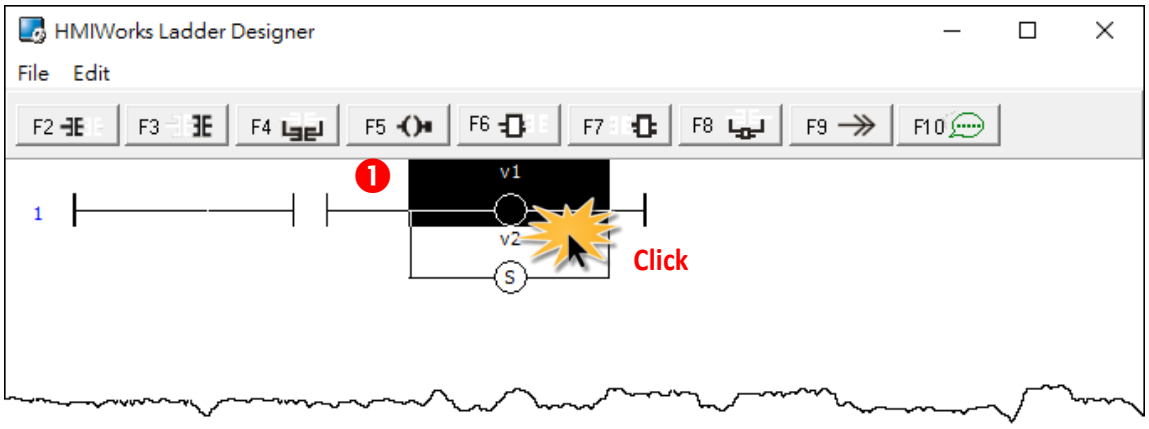
Step 1: We move the cursor to the **"v2"** coil output.

Step 2: Press **<Spacebar>** key twice to change the coil type to **"set"** coil.

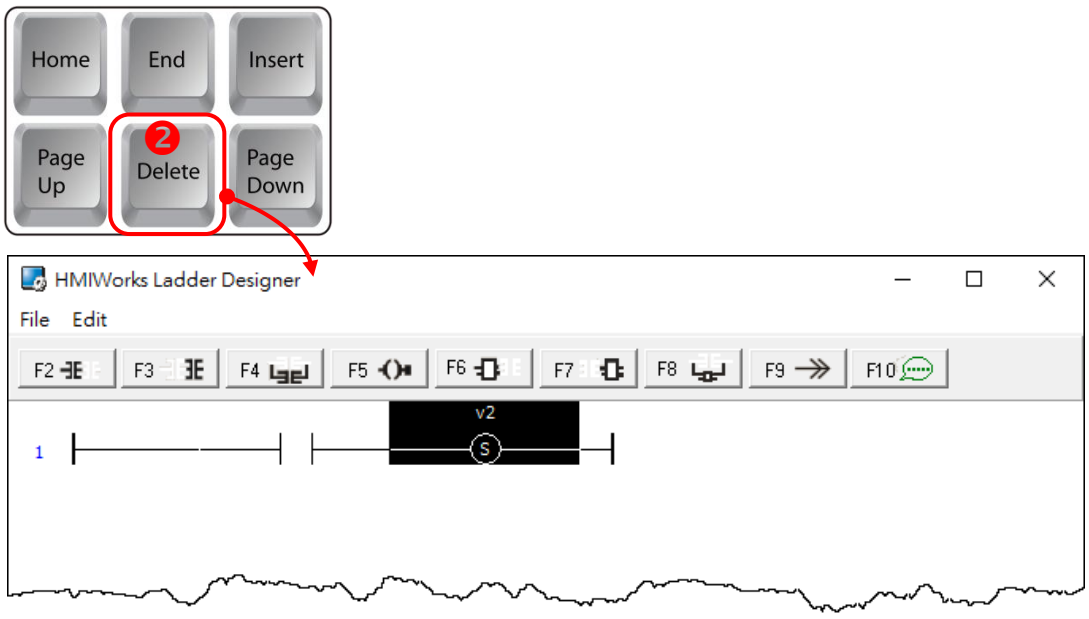


Delete a Coil Output in the rung

Step 1: Move the cursor to the coil **"v1"**.

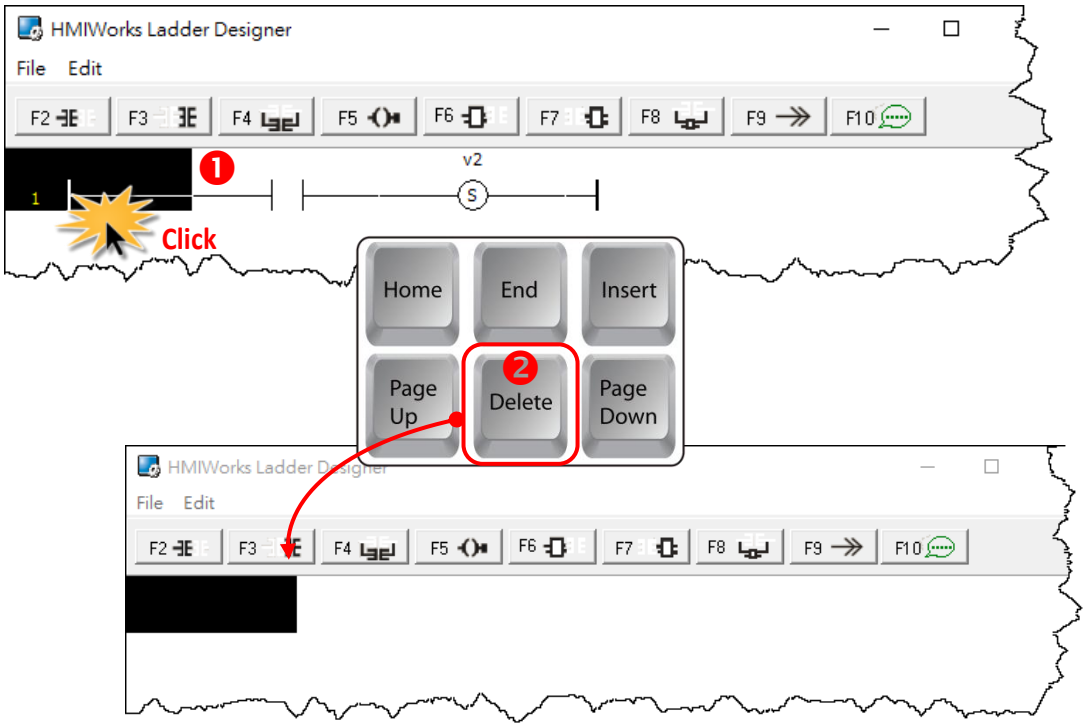


Step 2: Press **<Delete>** key to delete coil **"v1"**.



Delete the rung

Move the cursor to the **starting point** of the rung and press **<Delete>** key.



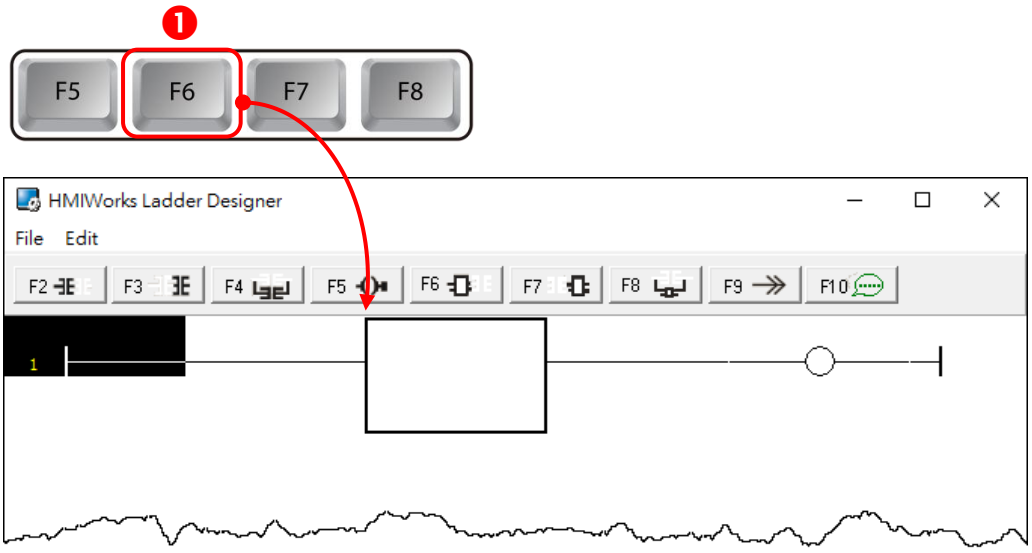
3.3.3.7 Inserting and Deleting a Function Block

To demonstrate how to insert or delete a function block and other related issues, go through the following steps.

Set the function type to a function block

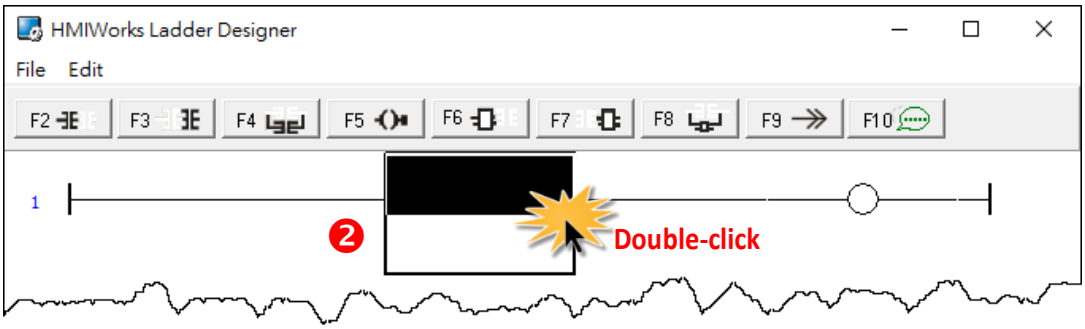
1. Insert a new rung

Step 1: Press <**F6**> key to insert a new rung with a function block and a coil output.

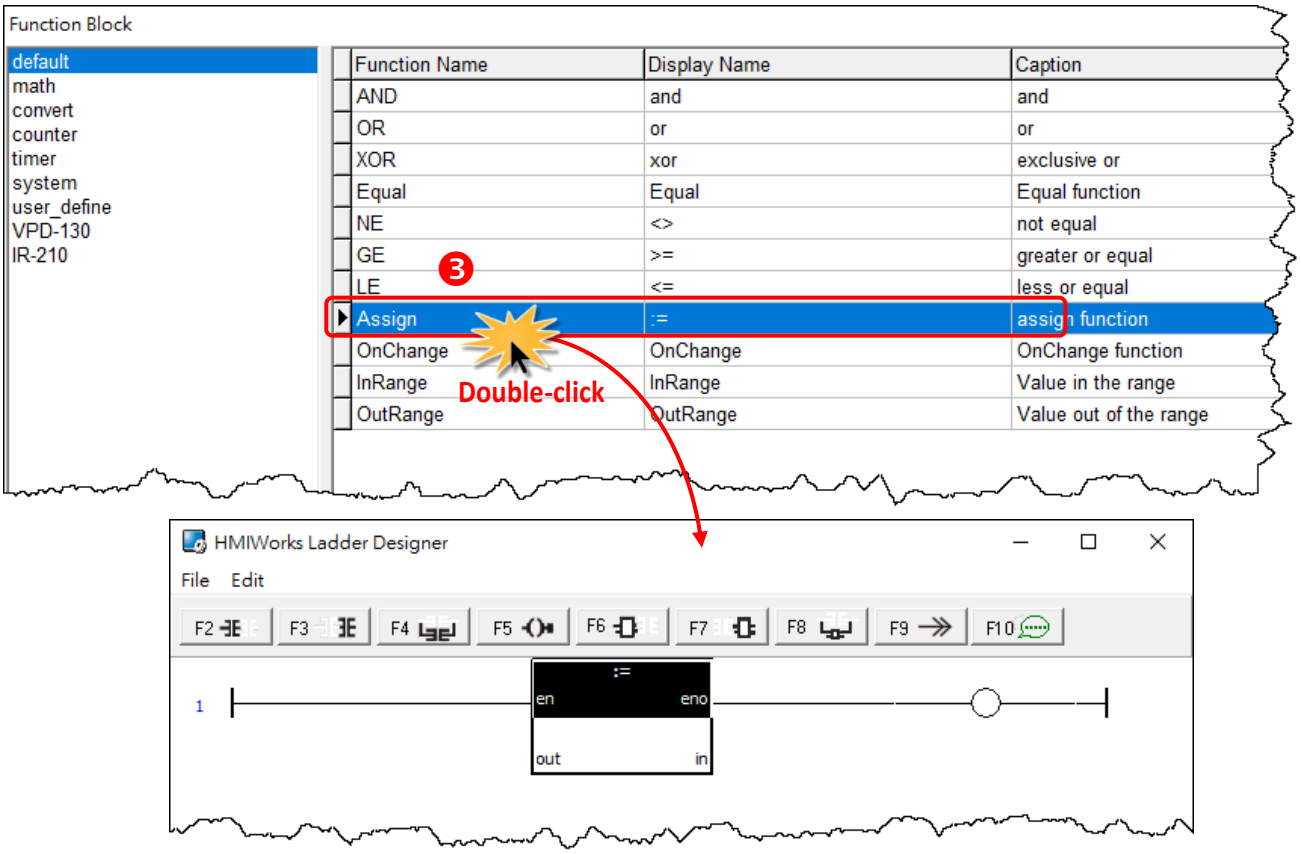


2. Choose function type

Step 2: In the new rung, double-click on the function block to open the "**Function Block**" window.

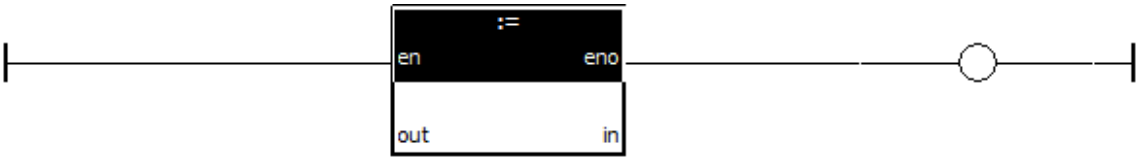


Step 3: Double-click on the **"Function Name"** field in the list to set the type of the function.
For example, we double-click on the Function **"Assign"** in the default group and set to the function block.



3. Assign the tag to the function

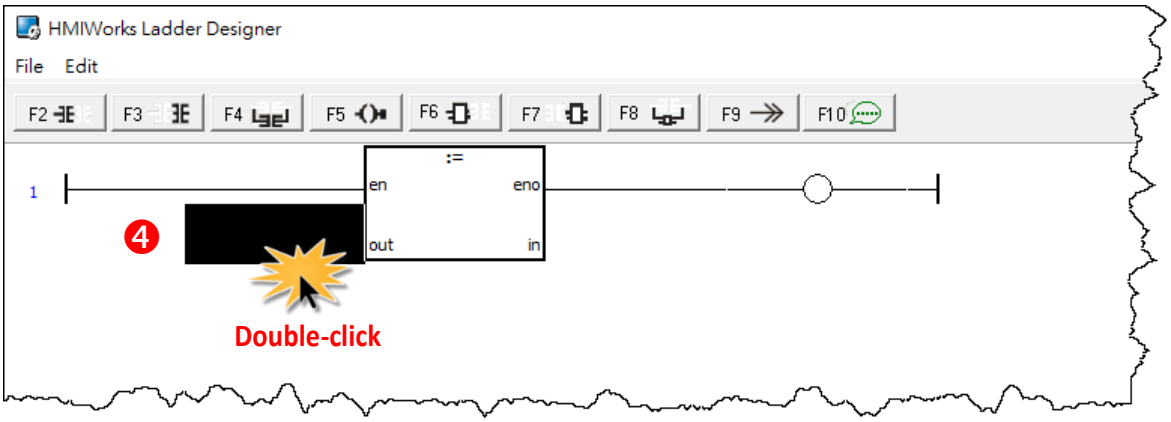
Now, we should assign the variable to the function **"Assign"**. As you can see, there are four tags: **"en"**, **"eno"**, **"out"** and **"in"**.



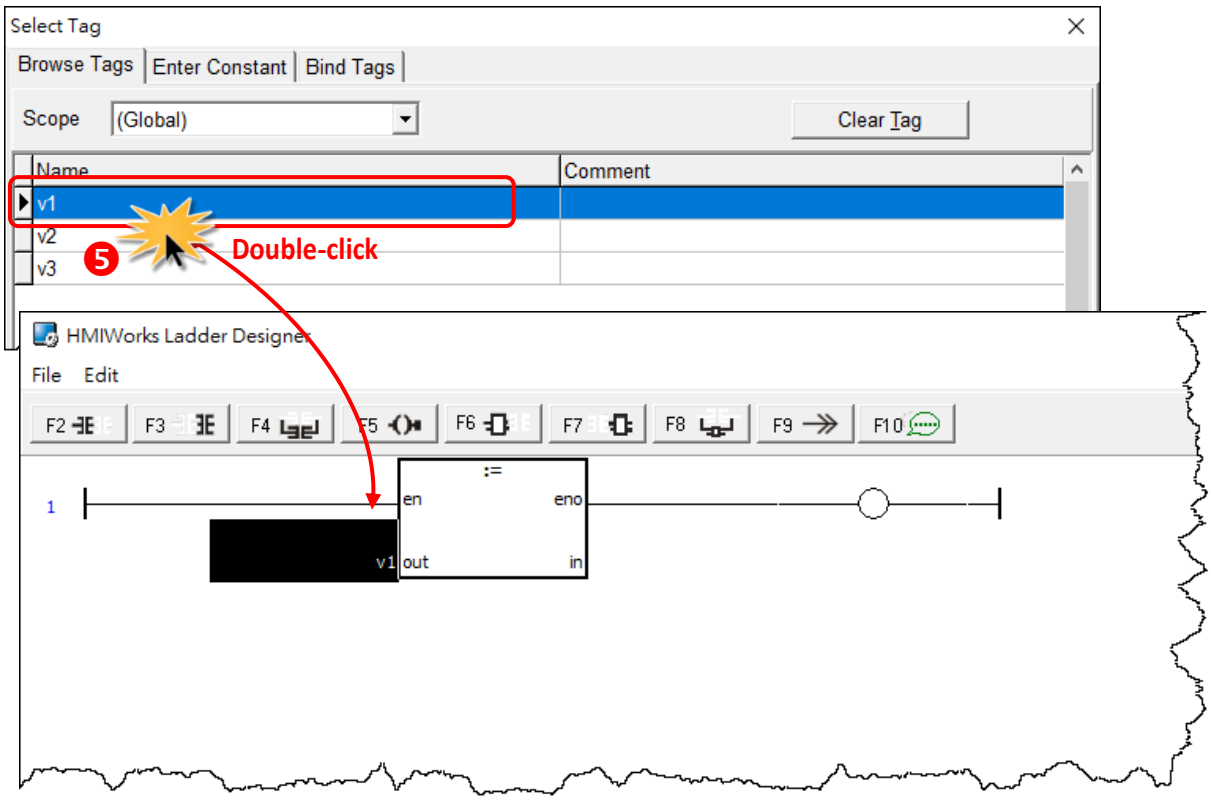
- Both **"en"** and **"eno"** cannot associate tags by users.
- We can associate **"out"** and **"in"** with the tags we define by "New Virtual Tags".

For example, we associate **"v1"** to **"out"** and **"v2"** to **"in"**. The v1, v2 and v3 are the tags defined in from the "**Edit Tag**" window. Refer to the Section 3.3.3.1 Add the New Virtual Tags (F2).

Step 4: Move the cursor just beside **“out”** but not in the function block. Double-click on **just beside “out”** to open “**Select Tag**” window.

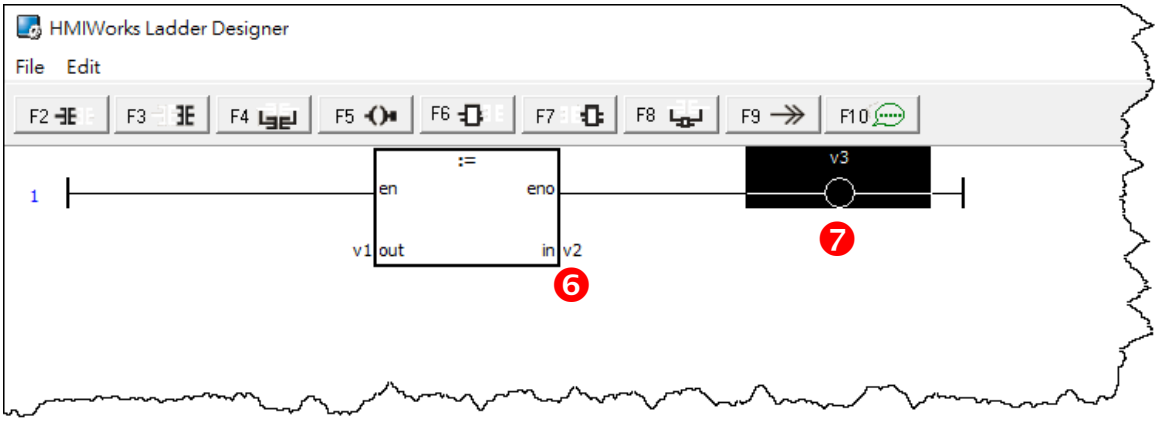


Step 5: Associate the tag **“v1”** to **“out”**. Double-click on the tag in the list to assign the tag to **“out”**. For example, we double-click on the variable **“v1”** and set to **“out”** of **“Assign”** function.



Step 6: Set **“v2”** to **“in”** of **“Assign”** function in the same way.

Step 7: Finally, set **“v3”** to the coil output.



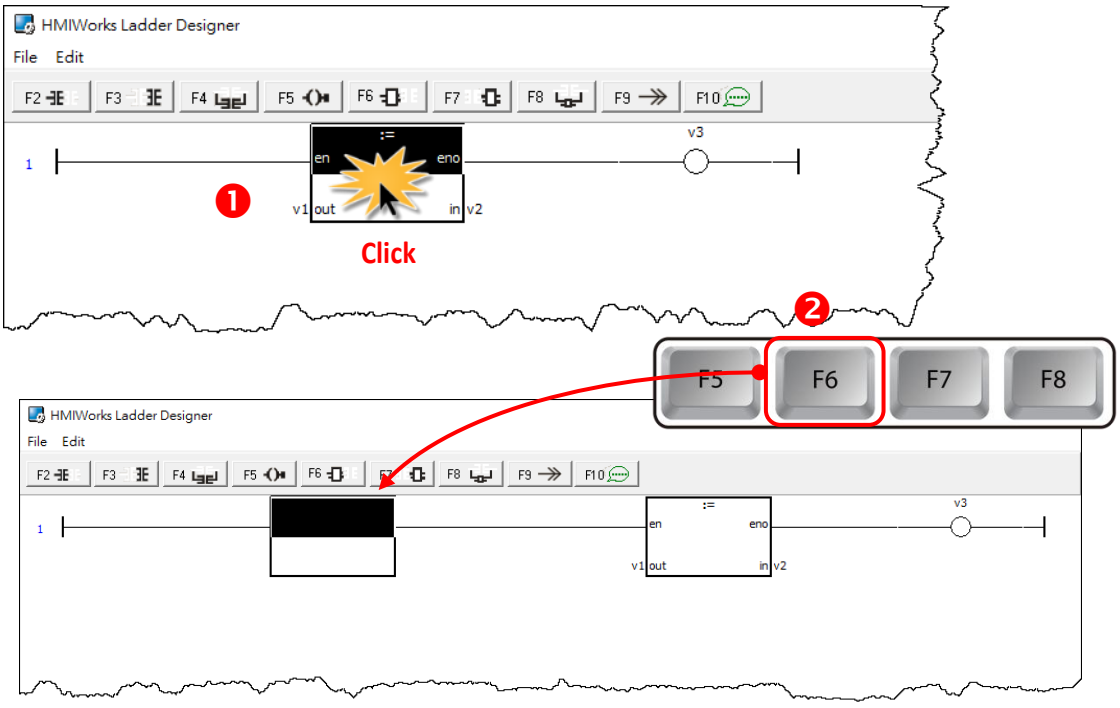
This function assigns **“v2”** to **“v1”** if en is set to high.

The coil output **“v3”** is purely defined by **“eno”**, where **“eno”** = **“en”**.

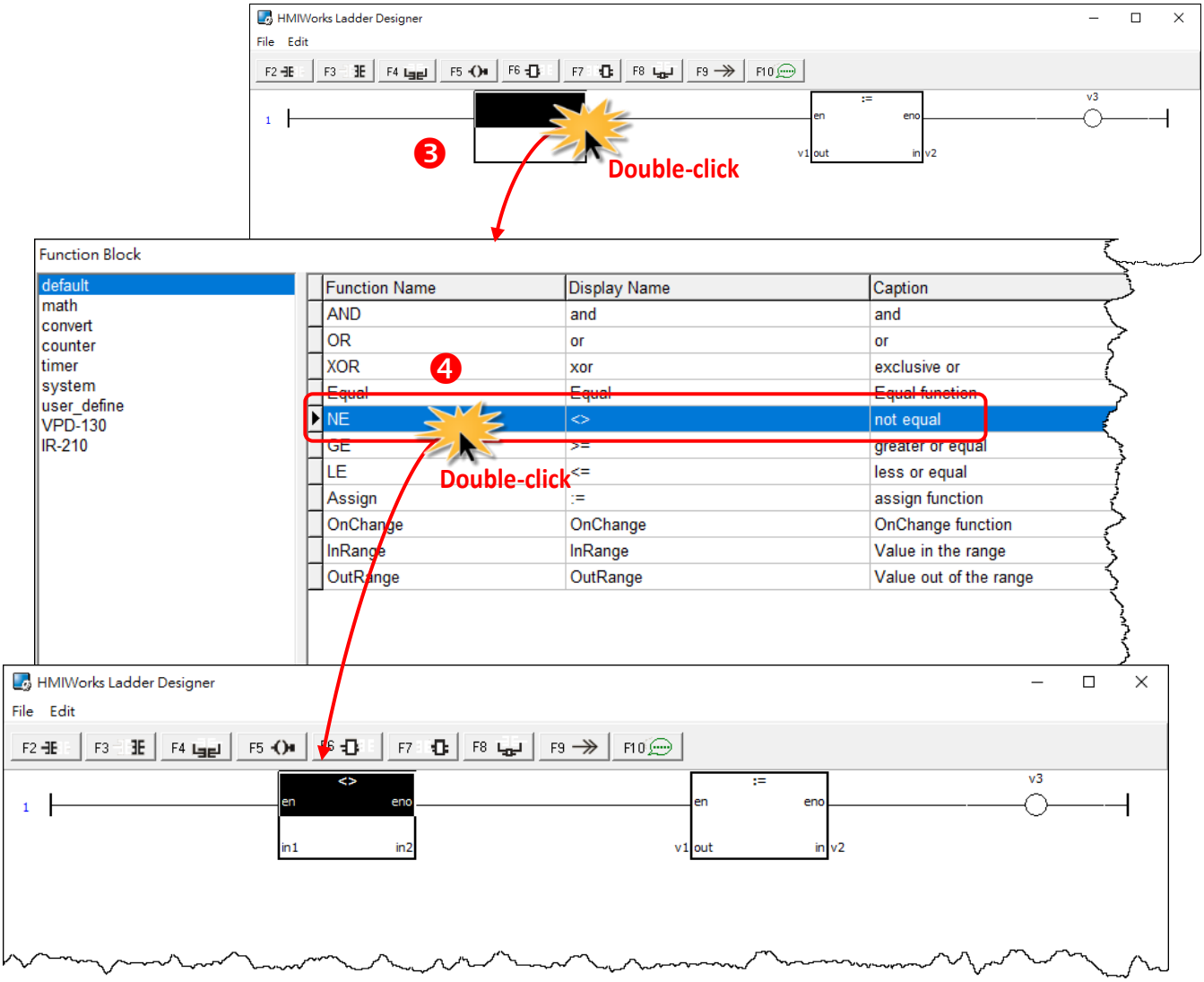
Insert a new function block in the left of the cursor (F6)

Step 1: Move the cursor to the **“Assign”** function block

Step 2: Press <**F6>** key.

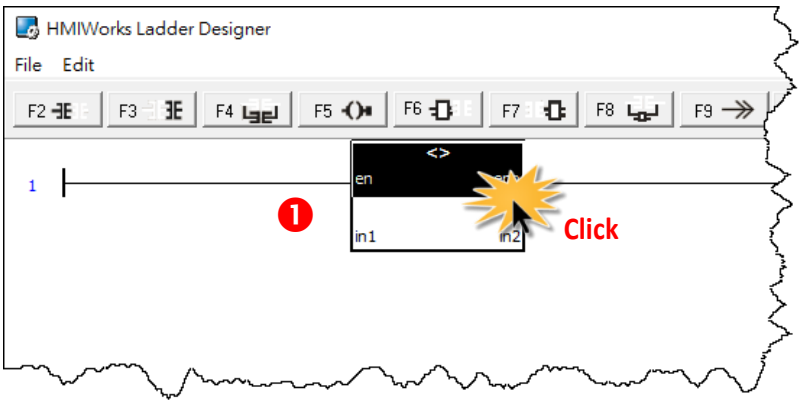


Step 3: And to make things clear, set the newly-inserted function block as **"NE" (not equal)**.

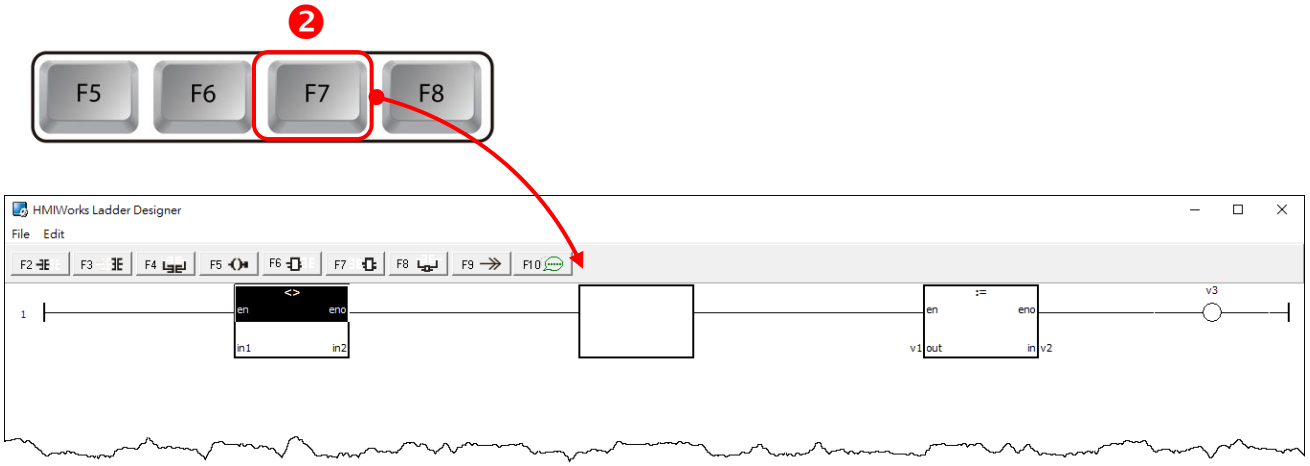


Insert a new function block in the right of the cursor (F7)

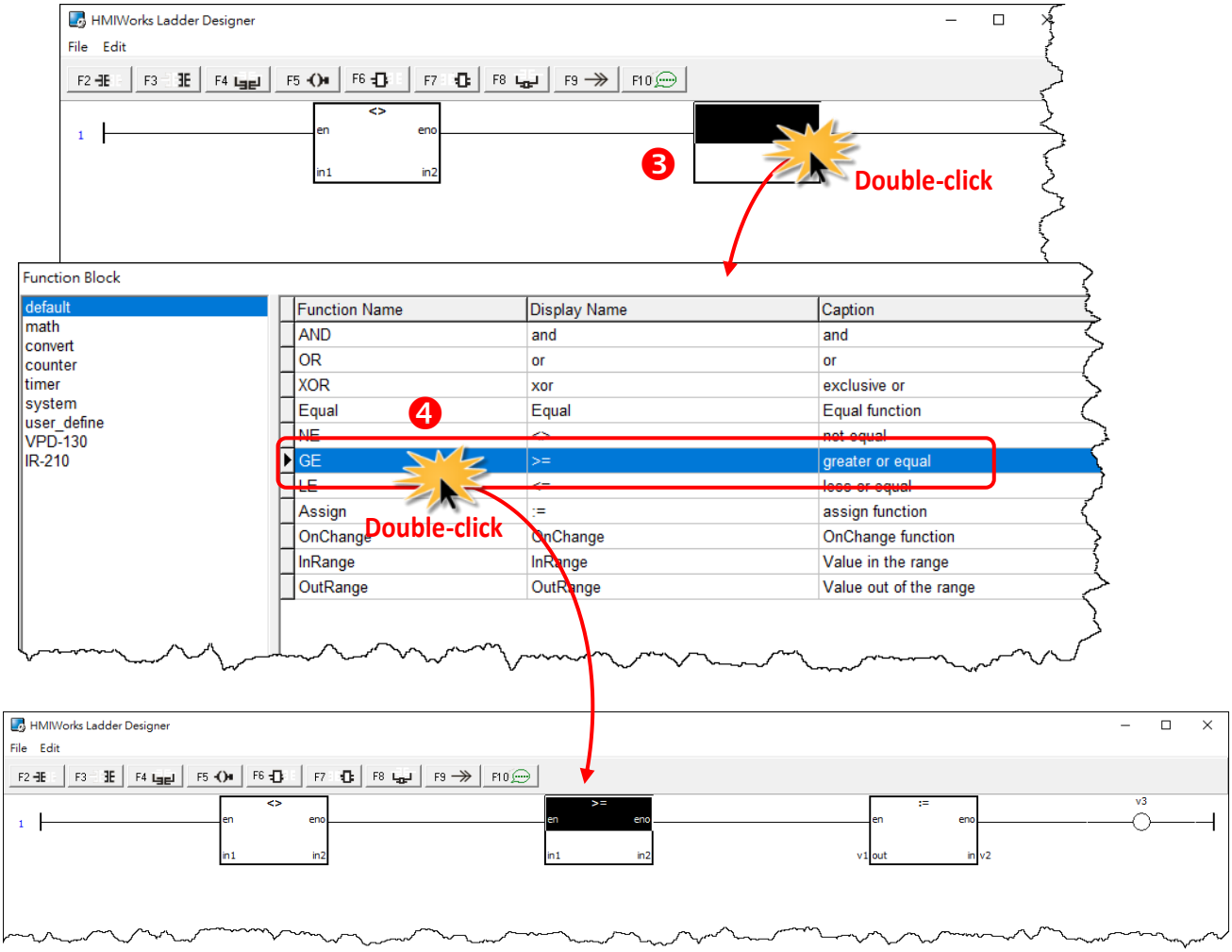
Step 1: Move the cursor to the **"NE"** function block



Step 2: Press <**F7>** key.

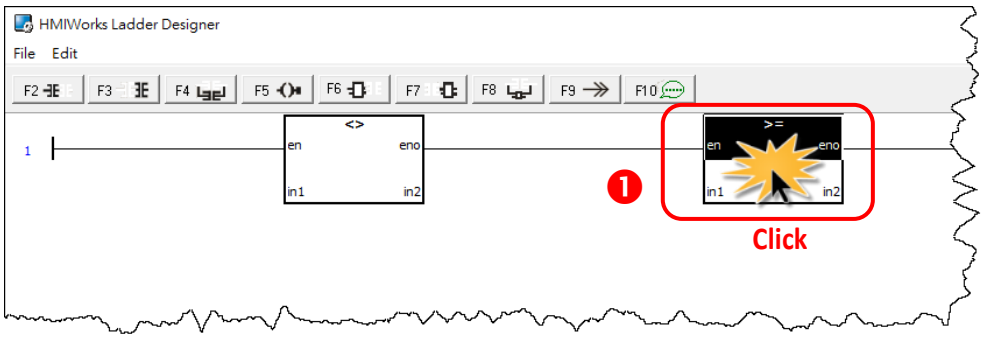


Step 3: Set the newly-inserted function block as **"GE" (greater than or equal)**.

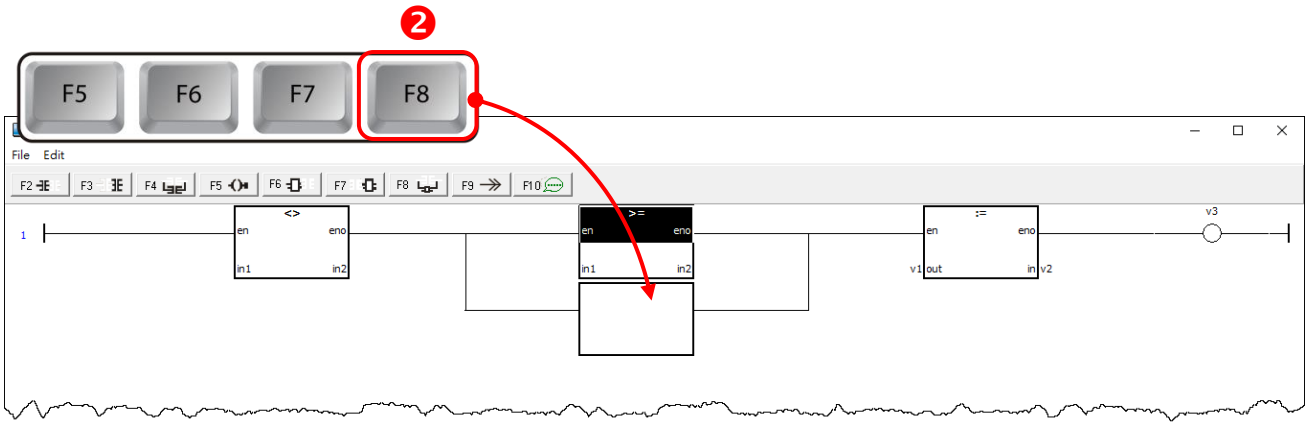


Insert a new function block which is parallel to the cursor (F8)

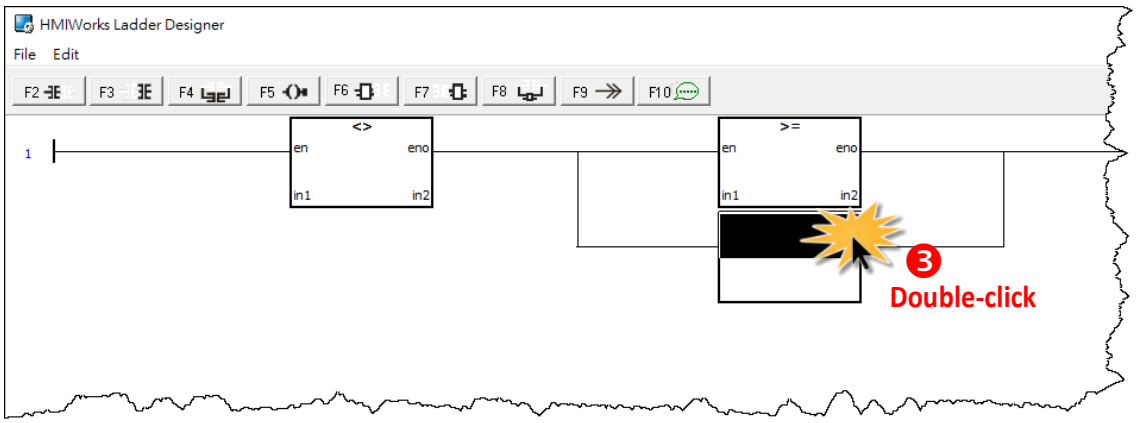
Step 1: Move the cursor to the **"GE"** function block

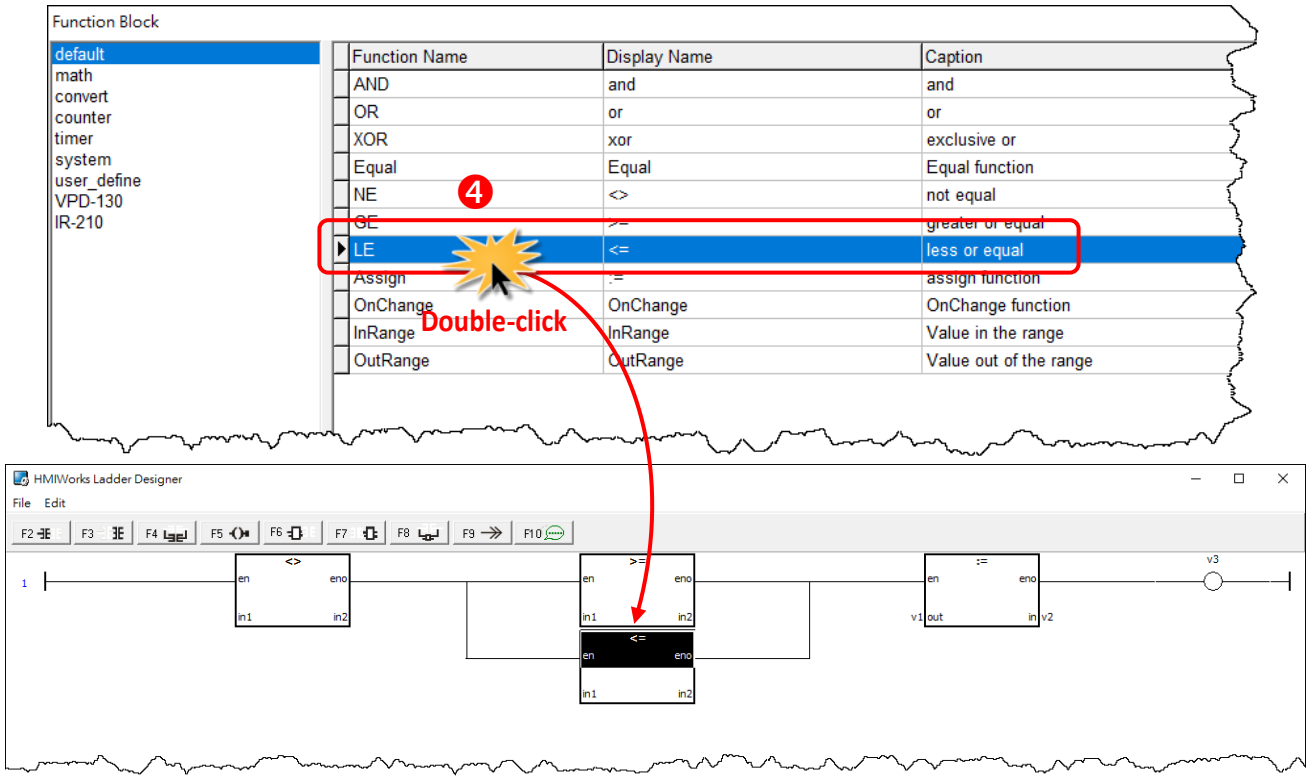


Step 2: Press **<F8>** key.



Step 3: Set the newly-inserted function block as **"LE" (less than or equal)**.

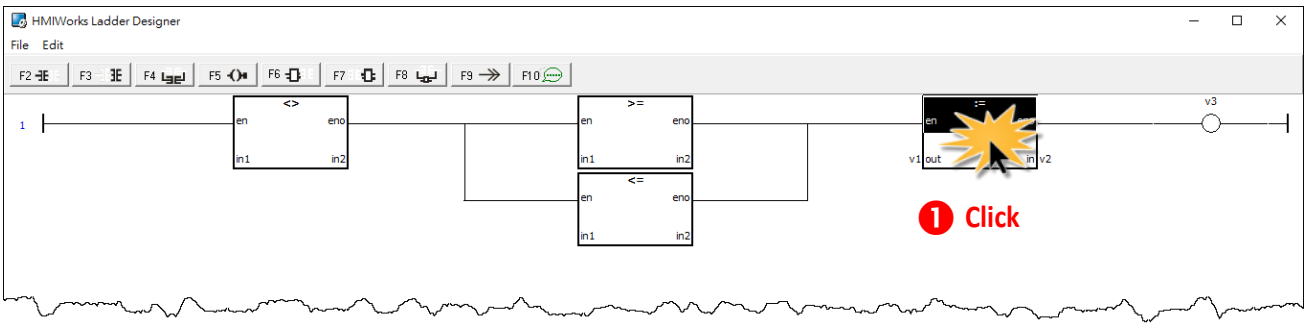




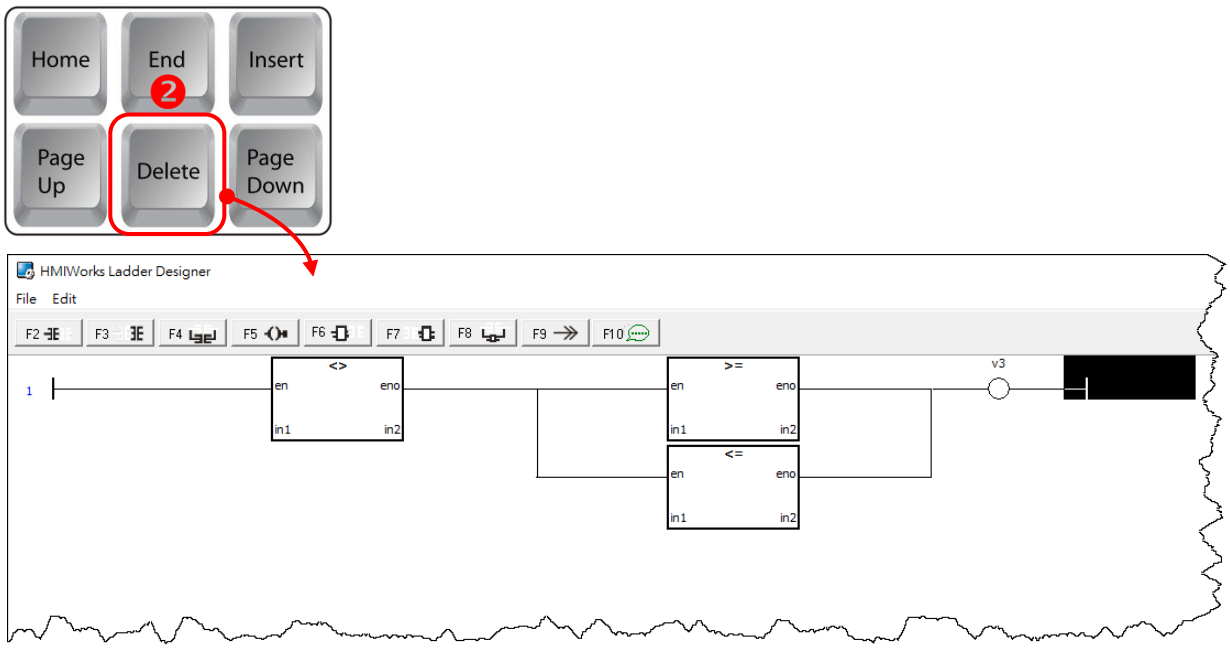
Delete a function block in the rung

Move the cursor to the function block you want to delete and press **<Delete>** key.

Step 1: Move the cursor to the **"Assign"** function block.

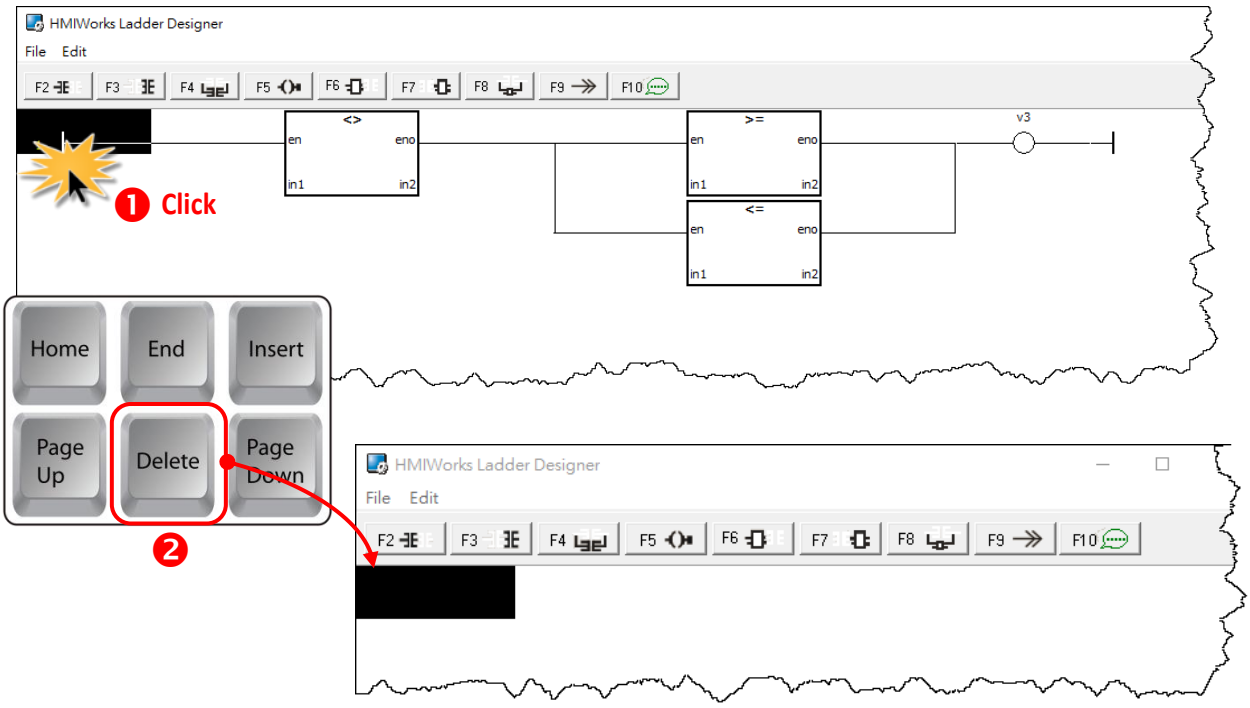


Step 2: Press **<Delete>** key to delete **"Assign"** function block.



Delete the rung

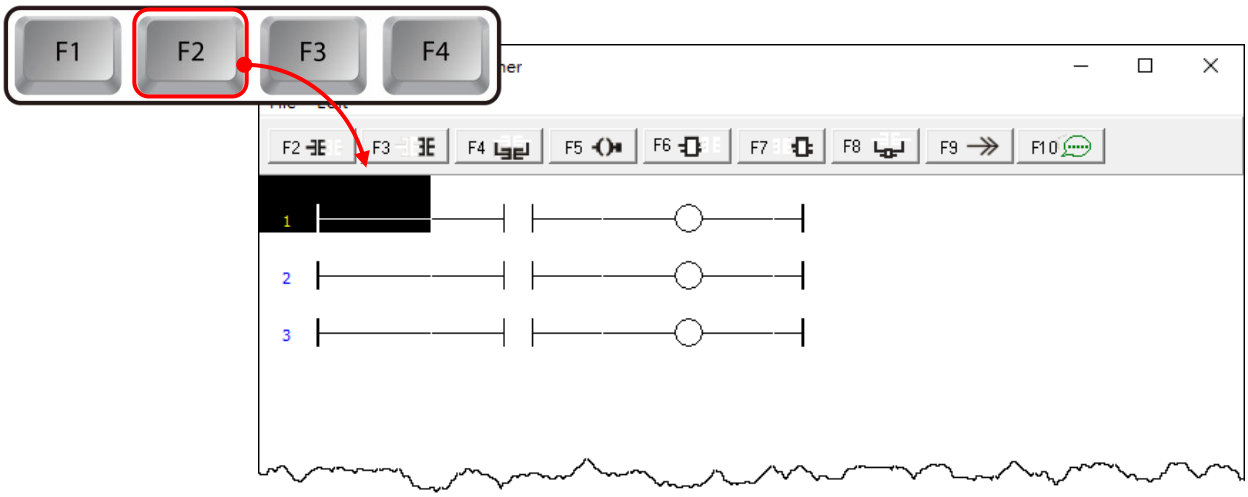
Move the cursor to the **starting point** of the rung and press **<Delete>** key.



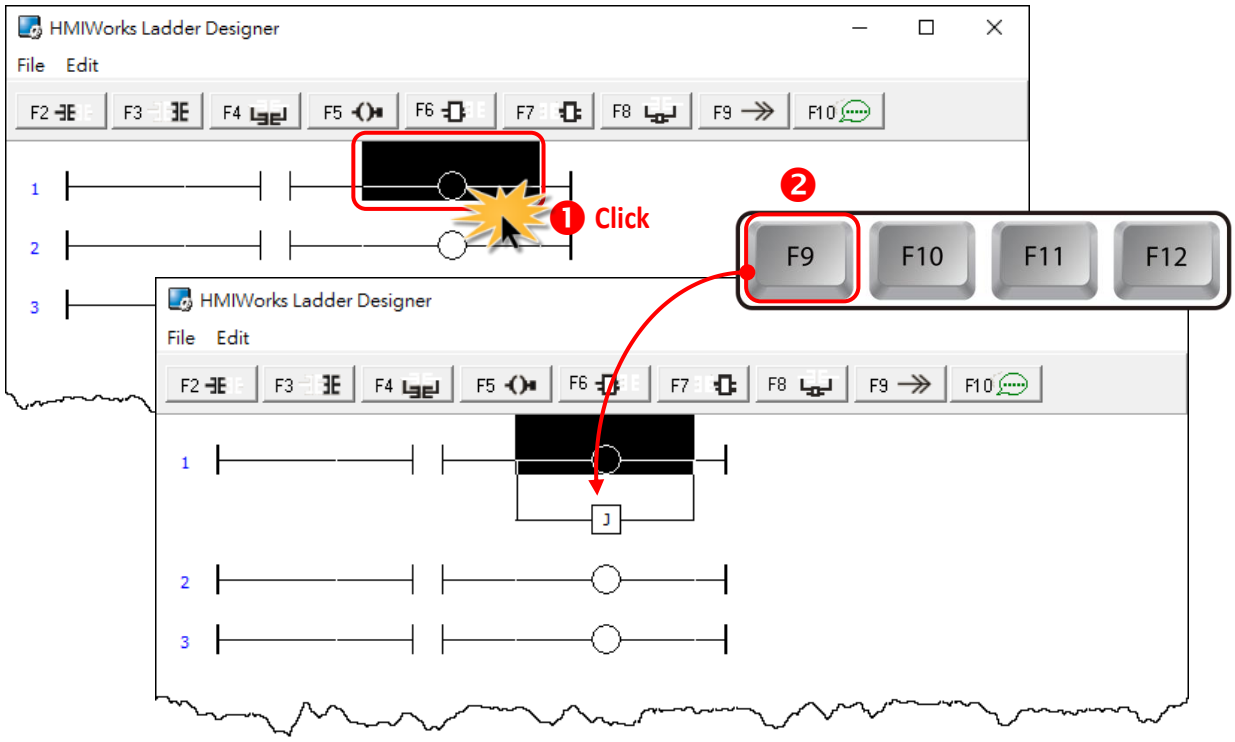
3.3.3.8 Jump to a Label

To demonstrate how to jump to a label, first we create three rungs and then explain how to skip the second rung and jump to the third.

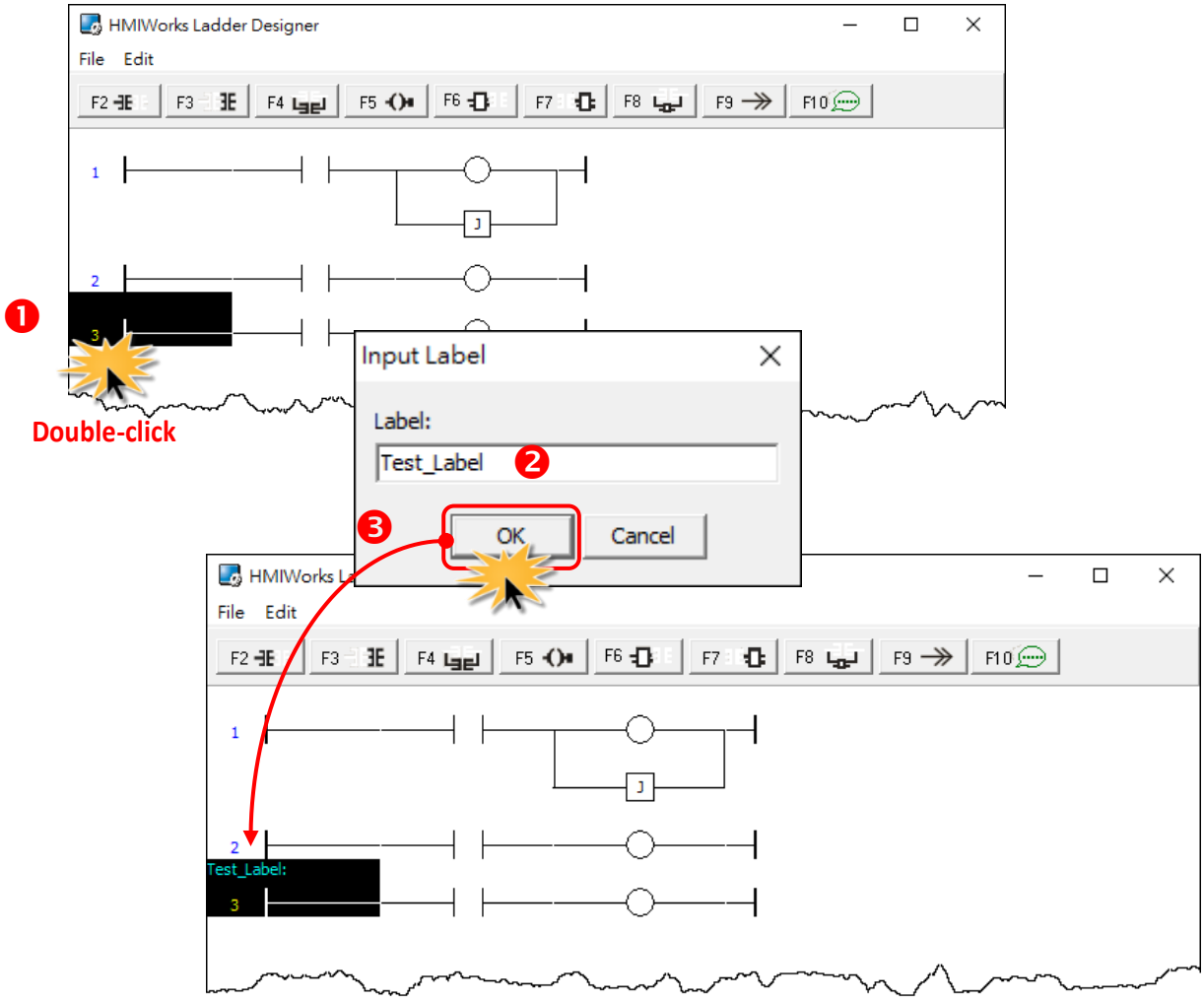
1. Press **<F2>** key three times to create three rungs for example.



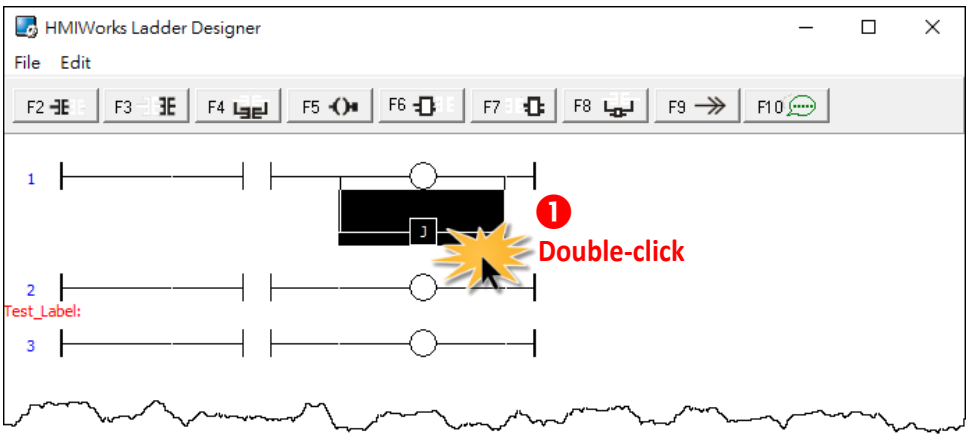
2. Move the cursor to the coil output of the first rung and press **<F9>** key to add a "**Jump**".

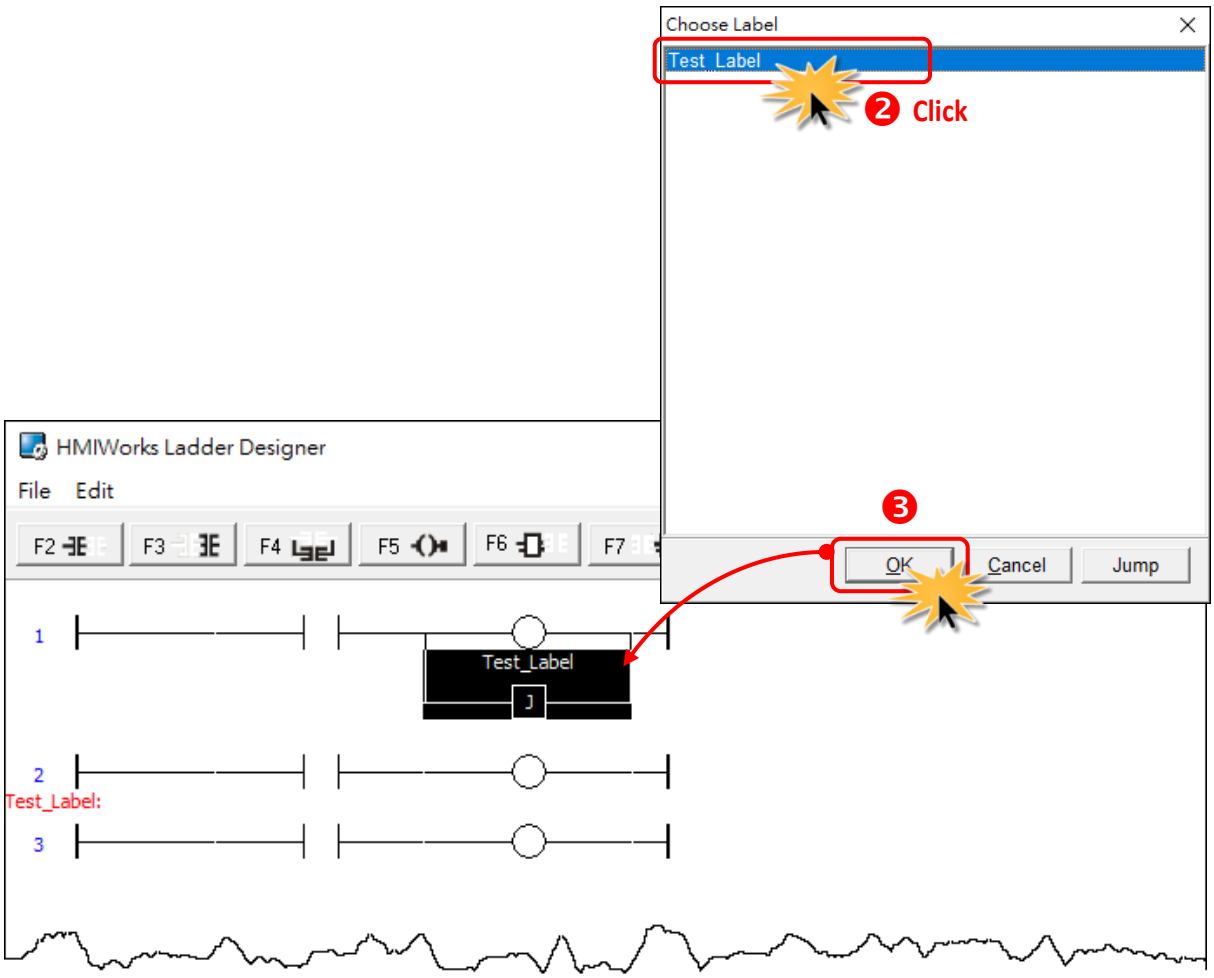


3. Double click on the **starting point** of the third rung to add a label **"Test_Label"** to it.



4. Double-click on the **"Jump"** of the first rung to associate with the label of the third rung.





5. When running the ladder logic, set the coil output of the first rung to high, skip the second rung and jump to the third rung if the contact input of the first rung is closed.

3.3.4 Function Block

The ladder provides a variety of function blocks for the user to application, including the math, convert, counter, timer and system function, etc. and provides these functions of the source code, refer to "**C:\ICPDAS\HMIWorks_Standard\bin\FunctionBlock**" for more details.

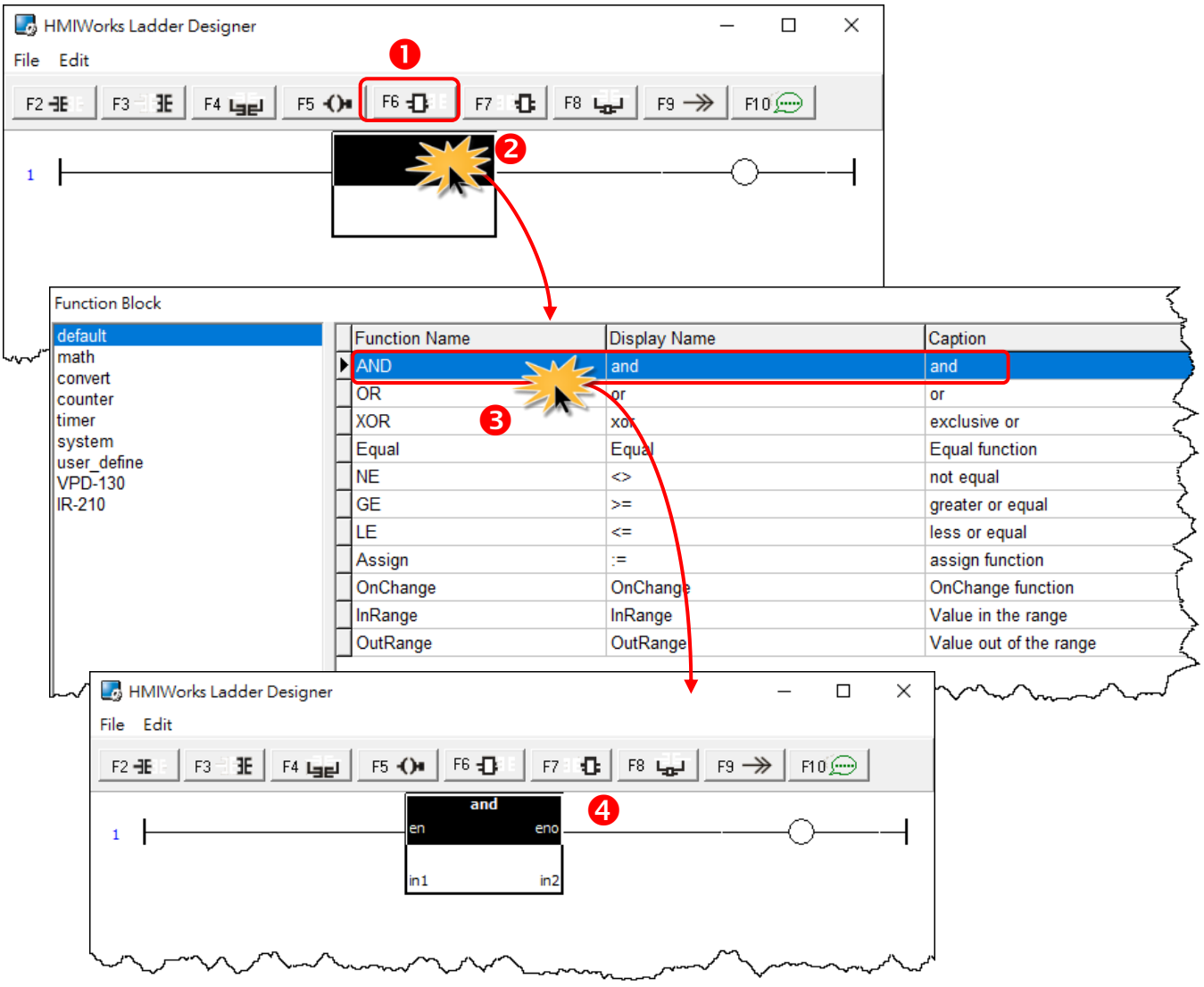
For example:

Step 1: Click the F6 button to create a function block.

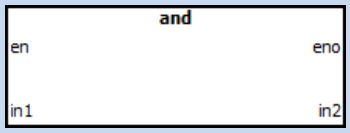
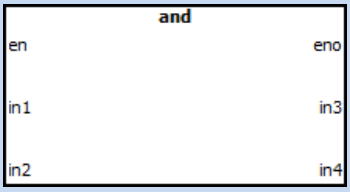
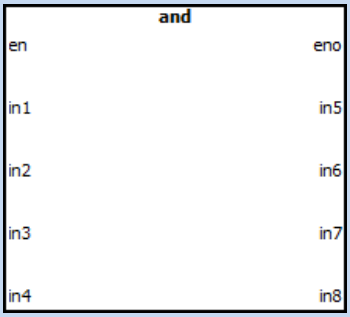
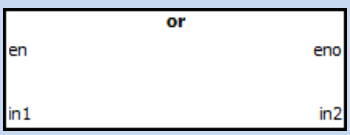
Step 2: Double-click it to open **"Function Block"** window.

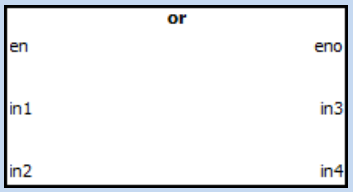
Step 3: Double-click a **function name (e.g., AND)** for you need.

Step 4: Open a **"AND"** function block.



3.3.4.1 Default Group

<table>
<tr><th>Function Block</th><th>Description and Pseudo Code</th></tr>
<tr><td> 
</td><td>
AND (Logical operator: And)
➢ Parameter:
in1: [Input] Input Value/Tag
in2: [Input] Input Value/Tag
➢ Examples:
<table>
<tr><th colspan="2">Input</th><th>Output</th></tr>
<tr><th>in1</th><th>in2</th><th>eno</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
➢ Pseudo Code:
If en == 1
eno = in1 & in2;
Else eno = 0;
</td></tr>
<tr><td> 
</td><td>
AND4
➢ Parameter:
in1: [Input] Input Value/Tag in3: [Input] Input Value/Tag
in2: [Input] Input Value/Tag in4: [Input] Input Value/Tag
➢ Pseudo Code:
If en == 1
eno = in1 & in2 & in3 & in4;
Else eno = 0;
</td></tr>
<tr><td> 
</td><td>
AND8
➢ Parameter:
in1: [Input] Input Value/Tag in5: [Input] Input Value/Tag
in2: [Input] Input Value/Tag in6: [Input] Input Value/Tag
in3: [Input] Input Value/Tag in7: [Input] Input Value/Tag
in4: [Input] Input Value/Tag in8: [Input] Input Value/Tag
➢ Pseudo Code:
If en == 1
eno = in1 & in2 & in3 & in4 & in5 & in6 & in7 & in8;
Else eno = 0;
</td></tr>
<tr><td> 
</td><td>
OR (Logical operator: Or)
➢ Parameter:
in1: [Input] Input Value/Tag
in2: [Input] Input Value/Tag
➢ Examples:
<table>
<tr><th colspan="2">Input</th><th>Output</th></tr>
<tr><th>in1</th><th>in2</th><th>eno</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
➢ Pseudo Code:
If en == 1
eno = in1 | in2;
Else eno = 0;
</td></tr>
</table>

<table>
<tr><th>Function Block</th><th>Description and Pseudo Code</th></tr>
<tr><td> 
</td><td>

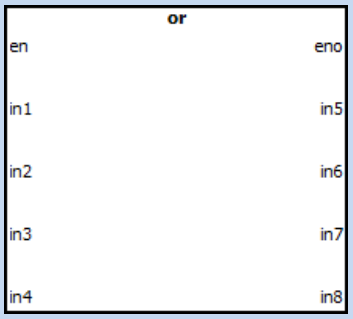
OR4

➢ **Parameter:**

in1: [Input] Input Value/Tag ***in3:*** [Input] Input Value/Tag

in2: [Input] Input Value/Tag ***in4***: [Input] Input Value/Tag

➢ **Pseudo Code:**

If en == 1
eno = in1 | in2 | in3 | in4;
Else eno = 0;
</td></tr>
<tr><td> 
</td><td>

OR8

➢ **Parameter:**

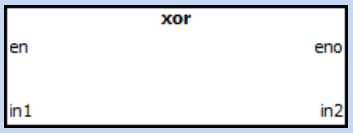
in1: [Input] Input Value/Tag ***in5:*** [Input] Input Value/Tag

in2: [Input] Input Value/Tag ***in6***: [Input] Input Value/Tag

in3: [Input] Input Value/Tag ***in7:*** [Input] Input Value/Tag

in4: [Input] Input Value/Tag ***in8***: [Input] Input Value/Tag

➢ **Pseudo Code:**

If en == 1
eno = in1 | in2 | in3 | in4 | in5 | in6 | in7 | in8;
Else eno = 0;
</td></tr>
<tr><td> 
</td><td>

XOR (Logical operator: Exclusive Or)

➢ **Parameter:**

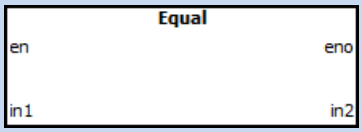
in1: [Input] Input Value/Tag

in2: [Input] Input Value/Tag

➢ **Examples:**

Input		Output
in1	in2	eno
0	0	0
0	1	1
1	0	1
1	1	0

➢ **Pseudo Code:**

If en == 1
eno = in1 ^ in2;
Else eno = 0;
</td></tr>
<tr><td> 
</td><td>

Equal (Mathematical Symbols: Equality)

➢ **Parameter:**

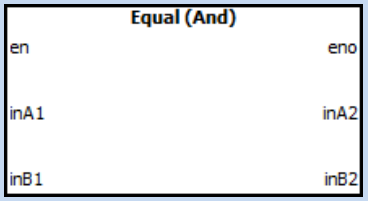
in1: [Input] Input Value/Tag

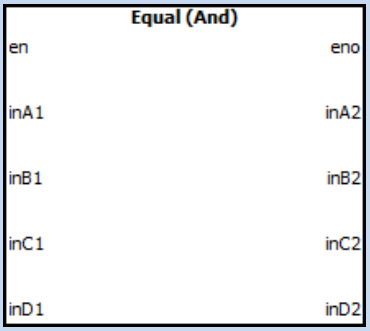
in2: [Input] Input Value/Tag

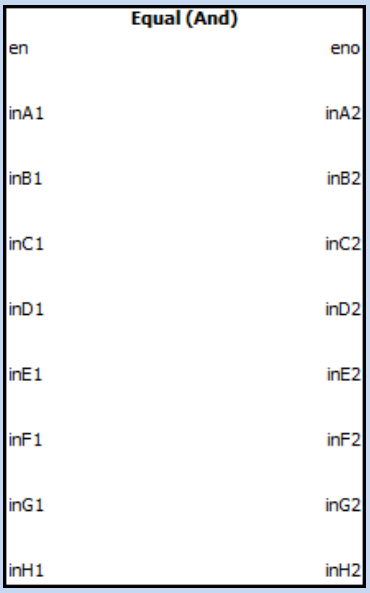
➢ **Pseudo Code**:

If (en == 1 and in1 is equal to in2)
eno = 1;
Else eno = 0;
</td></tr>
</table>

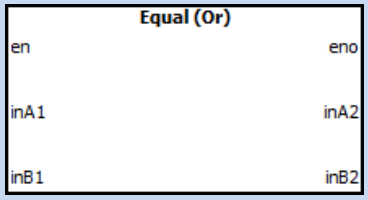
Function Block	Description and Pseudo Code

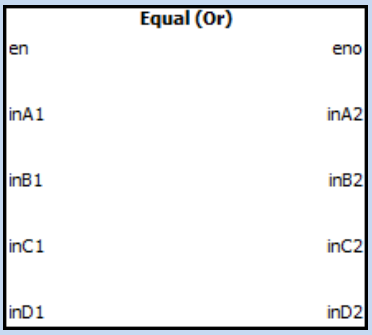
	**Equal_And2**
	➢ **Parameter:**
	inA1: [Input] Input Value/Tag ***inA2*:** [Input] Input Value/Tag
	inB1: [Input] Input Value/Tag ***inB2*:** [Input] Input Value/Tag
	➢ **Pseudo Code**:
	If (en == 1 and
	inA1 is equal to inA2 and inB1 is equal to inB2)
	eno = 1;
	Else eno = 0;

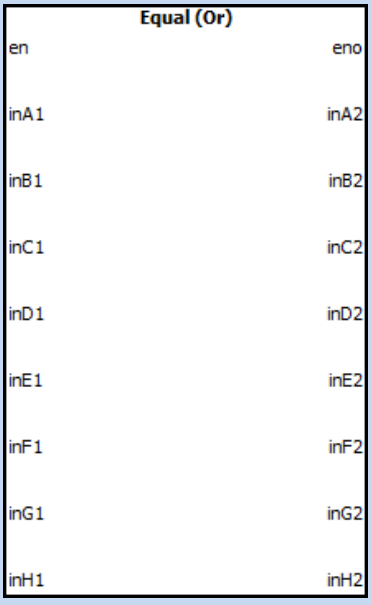
	**Equal_And4**
	➢ **Parameter:**
	inA1: [Input] Input Value/Tag ***inA2*:** [Input] Input Value/Tag
	inB1: [Input] Input Value/Tag ***inB2*:** [Input] Input Value/Tag
	inC1: [Input] Input Value/Tag ***inC2*:** [Input] Input Value/Tag
	inD1: [Input] Input Value/Tag ***inD2:*** [Input] Input Value/Tag
	➢ **Pseudo Code**:
	If (en == 1 and
	inA1 is equal to inA2 and inB1 is equal to inB2 and
	inC1 is equal to inC2 and inD1 is equal to inD2)
	eno = 1;
	Else eno = 0;

	**Equal_And8**
	➢ **Parameter:**
	inA1: [Input] Input Value/Tag ***inA2*:** [Input] Input Value/Tag
	inB1: [Input] Input Value/Tag ***inB2*:** [Input] Input Value/Tag
	inC1: [Input] Input Value/Tag ***inC2*:** [Input] Input Value/Tag
	inD1: [Input] Input Value/Tag ***inD2:*** [Input] Input Value/Tag
	inE1: [Input] Input Value/Tag ***inE2*:** [Input] Input Value/Tag
	inF1: [Input] Input Value/Tag ***inF2*:** [Input] Input Value/Tag
	inG1: [Input] Input Value/Tag ***inG2:*** [Input] Input Value/Tag
	inH1: [Input] Input Value/Tag ***inH2*:** [Input] Input Value/Tag
	➢ **Pseudo Code**:
	If (en == 1 and
	inA1 is equal to inA2 and inB1 is equal to inB2 and
	inC1 is equal to inC2 and inD1 is equal to inD2 and
	inE1 is equal to inE2 and inF1 is equal to inF2 and
	inG1 is equal to inG2 and inH1 is equal to inH2)
	eno = 1;
	Else eno = 0;

Function Block	Description and Pseudo Code

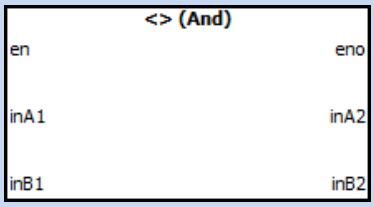
	**Equal_Or2**
	➢ **Parameter:**
	inA1: [Input] Input Value/Tag ***inA2*:** [Input] Input Value/Tag
	inB1: [Input] Input Value/Tag ***inB2*:** [Input] Input Value/Tag
	➢ **Pseudo Code**:
	If (en == 1 and
	inA1 is equal to inA2 or inB1 is equal to inB2)
	eno = 1;
	Else eno = 0;

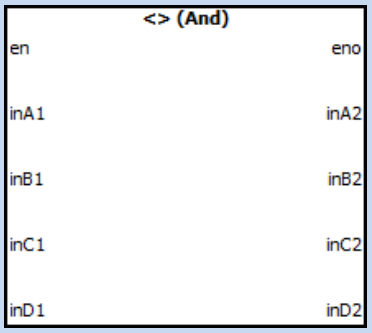
	**Equal_Or4**
	➢ **Parameter:**
	inA1: [Input] Input Value/Tag ***inA2*:** [Input] Input Value/Tag
	inB1: [Input] Input Value/Tag ***inB2*:** [Input] Input Value/Tag
	inC1: [Input] Input Value/Tag ***inC2*:** [Input] Input Value/Tag
	inD1: [Input] Input Value/Tag ***inD2:*** [Input] Input Value/Tag
	➢ **Pseudo Code**:
	If (en == 1 and
	inA1 is equal to inA2 or inB1 is equal to inB2 or
	inC1 is equal to inC2 or inD1 is equal to inD2)
	eno = 1;
	Else eno = 0;

	**Equal_Or8**
	➢ **Parameter:**
	inA1: [Input] Input Value/Tag ***inA2*:** [Input] Input Value/Tag
	inB1: [Input] Input Value/Tag ***inB2*:** [Input] Input Value/Tag
	inC1: [Input] Input Value/Tag ***inC2*:** [Input] Input Value/Tag
	inD1: [Input] Input Value/Tag ***inD2:*** [Input] Input Value/Tag
	inE1: [Input] Input Value/Tag ***inE2*:** [Input] Input Value/Tag
	inF1: [Input] Input Value/Tag ***inF2*:** [Input] Input Value/Tag
	inG1: [Input] Input Value/Tag ***inG2:*** [Input] Input Value/Tag
	inH1: [Input] Input Value/Tag ***inH2*:** [Input] Input Value/Tag
	➢ **Pseudo Code**:
	If (en == 1 and
	inA1 is equal to inA2 or inB1 is equal to inB2 or
	inC1 is equal to inC2 or inD1 is equal to inD2 or
	inE1 is equal to inE2 or inF1 is equal to inF2 or
	inG1 is equal to inG2 or inH1 is equal to inH2)
	eno = 1;
	Else eno = 0;

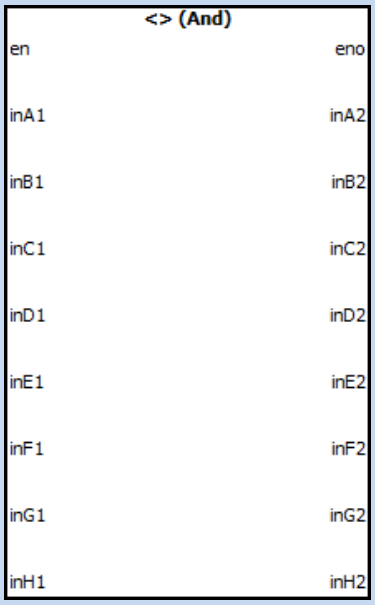
Function Block	Description and Pseudo Code

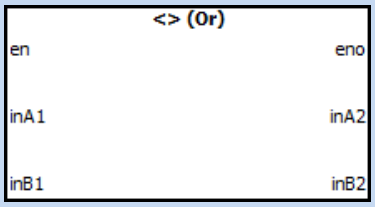
	**NE** (Mathematical Symbols: Not Equality)
	➢ **Parameter:**
	in1: [Input] Input Value/Tag
	in2: [Input] Input Value/Tag
	➢ **Pseudo Code:**
	If (en == 1 and in1 is not equal to in2)
	eno = 1;
	Else eno = 0;

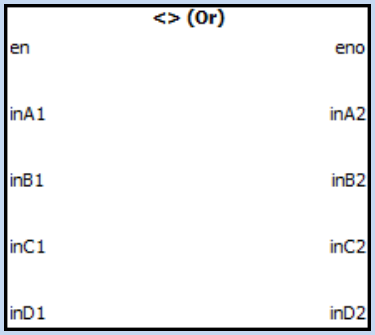
	**NE_And2**
	➢ **Parameter:**
	inA1: [Input] Input Value/Tag ***inA2***: [Input] Input Value/Tag
	inB1: [Input] Input Value/Tag ***inB2***: [Input] Input Value/Tag
	➢ **Pseudo Code:**
	If (en == 1 and
	inA1 is not equal to inA2 and
	inB1 is not equal to inB2)
	eno = 1;
	Else eno = 0;

	**NE_And4**
	➢ **Parameter:**
	inA1: [Input] Input Value/Tag ***inA2***: [Input] Input Value/Tag
	inB1: [Input] Input Value/Tag ***inB2***: [Input] Input Value/Tag
	inC1: [Input] Input Value/Tag ***inC2***: [Input] Input Value/Tag
	inD1: [Input] Input Value/Tag ***inD2:*** [Input] Input Value/Tag
	➢ **Pseudo Code:**
	If (en == 1 and
	inA1 is not equal to inA2 and
	inB1 is not equal to inB2 and
	inC1 is not equal to inC2 and
	inD1 is not equal to inD2)
	eno = 1;
	Else eno = 0;

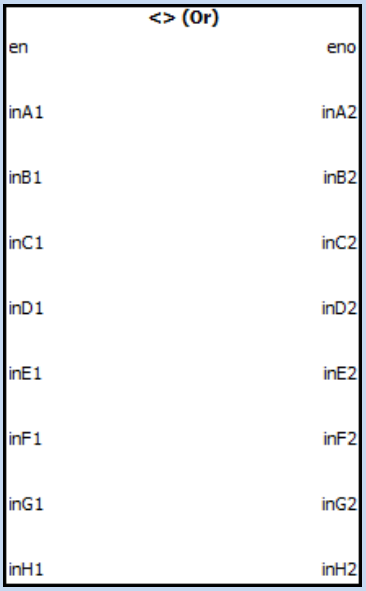
Function Block	Description and Pseudo Code

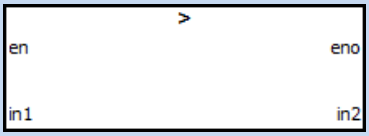
	**NE_And8**
	➢ **Parameter:**
	inA1: [Input] Input Value/Tag ***inA2*:** [Input] Input Value/Tag
	inB1: [Input] Input Value/Tag ***inB2*:** [Input] Input Value/Tag
	inC1: [Input] Input Value/Tag ***inC2*:** [Input] Input Value/Tag
	inD1: [Input] Input Value/Tag ***inD2:*** [Input] Input Value/Tag
	inE1: [Input] Input Value/Tag ***inE2*:** [Input] Input Value/Tag
	inF1: [Input] Input Value/Tag ***inF2*:** [Input] Input Value/Tag
	inG1: [Input] Input Value/Tag ***inG2:*** [Input] Input Value/Tag
	inH1: [Input] Input Value/Tag ***inH2*:** [Input] Input Value/Tag
	➢ **Pseudo Code:**
	If (en == 1 and
	inA1 is not equal to inA2 and inB1 is not equal to inB2 and
	inC1 is not equal to inC2 and inD1 is not equal to inD2 and
	inE1 is not equal to inE2 and inF1 is not equal to inF2 and
	inG1 is not equal to inG2 and inH1 is not equal to inH2)
	eno = 1;
	Else eno = 0;

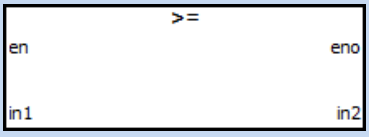
	**NE_Or2**
	➢ **Parameter:**
	inA1: [Input] Input Value/Tag ***inA2*:** [Input] Input Value/Tag
	inB1: [Input] Input Value/Tag ***inB2*:** [Input] Input Value/Tag
	➢ **Pseudo Code:**
	If (en == 1 and
	inA1 is not equal to inA2 or inB1 is not equal to inB2)
	eno = 1;
	Else eno = 0;

	**NE_Or4**
	➢ **Parameter:**
	inA1: [Input] Input Value/Tag ***inA2*:** [Input] Input Value/Tag
	inB1: [Input] Input Value/Tag ***inB2*:** [Input] Input Value/Tag
	inC1: [Input] Input Value/Tag ***inC2*:** [Input] Input Value/Tag
	inD1: [Input] Input Value/Tag ***inD2:*** [Input] Input Value/Tag
	➢ **Pseudo Code:**
	If (en == 1 and
	inA1 is not equal to inA2 or inB1 is not equal to inB2 or
	inC1 is not equal to inC2 or inD1 is not equal to inD2)
	eno = 1;
	Else eno = 0;

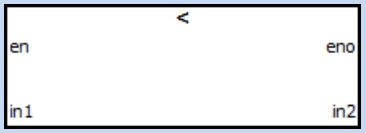
Function Block	Description and Pseudo Code

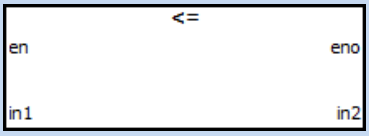
	**NE_Or8**
	➢ **Parameter:**
	inA1: [Input] Input Value/Tag ***inA2*:** [Input] Input Value/Tag
	inB1: [Input] Input Value/Tag ***inB2*:** [Input] Input Value/Tag
	inC1: [Input] Input Value/Tag ***inC2*:** [Input] Input Value/Tag
	inD1: [Input] Input Value/Tag ***inD2:*** [Input] Input Value/Tag
	inE1: [Input] Input Value/Tag ***inE2*:** [Input] Input Value/Tag
	inF1: [Input] Input Value/Tag ***inF2*:** [Input] Input Value/Tag
	inG1: [Input] Input Value/Tag ***inG2:*** [Input] Input Value/Tag
	inH1: [Input] Input Value/Tag ***inH2*:** [Input] Input Value/Tag
	➢ **Pseudo Code:**
	If (en == 1 and
	inA1 is not equal to inA2 or
	inB1 is not equal to inB2 or
	inC1 is not equal to inC2 or
	inD1 is not equal to inD2 or
	inE1 is not equal to inE2 or
	inF1 is not equal to inF2 or
	inG1 is not equal to inG2 or
	inH1 is not equal to inH2)
	eno = 1;
	Else eno = 0;

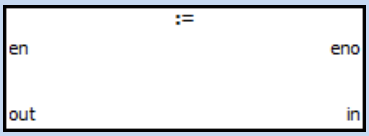
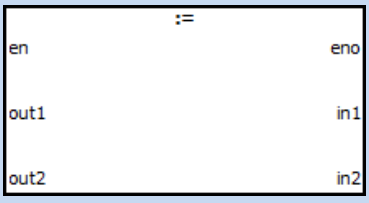
	**Greater** (Mathematical Symbols: Greater)
	➢ **Parameter:**
	***in1*:** [Input] Input Value/Tag
	***in2*:** [Input] Input Value/Tag
	➢ **Pseudo Code:**
	If (en == 1 and in1 > in2)
	eno = 1;
	Else eno = 0;

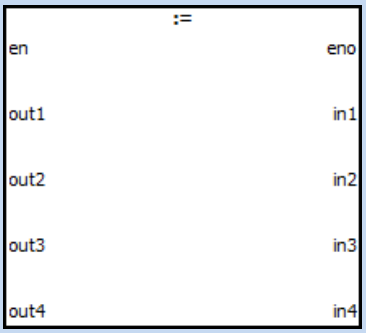
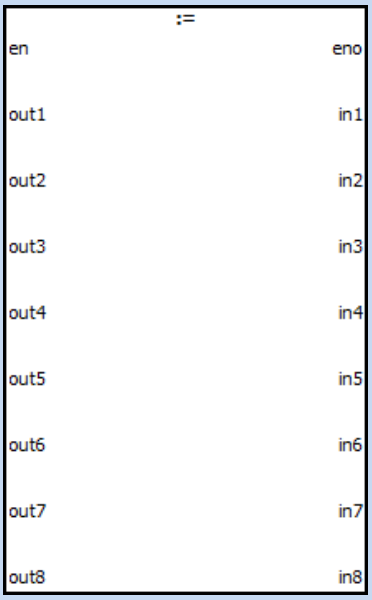
	**GE** (Mathematical Symbols: Greater or Equal)
	➢ **Parameter:**
	***in1*:** [Input] Input Value/Tag
	***in2*:** [Input] Input Value/Tag
	➢ **Pseudo Code:**
	If (en == 1 and in1 >= in2),
	eno = 1;
	Else eno = 0;

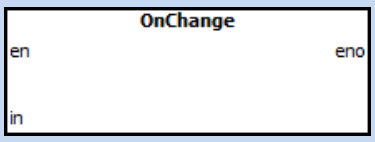
Function Block	Description and Pseudo Code

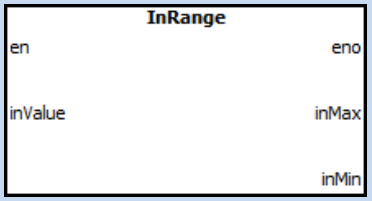
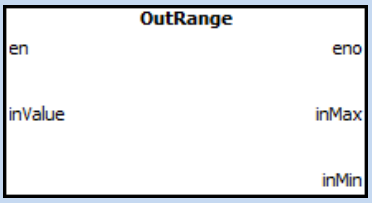
	**Less** (Mathematical Symbols: Less)
	➢ **Parameter:**
	in1: [Input] Input Value/Tag
	in2: [Input] Input Value/Tag
	➢ **Pseudo Code:**
	If (en == 1 and in1 < in2)
	eno = 1;
	Else eno = 0;

	**LE** (Mathematical Symbols: Less or Equal)
	➢ **Parameter:**
	in1: [Input] Input Value/Tag
	in2: [Input] Input Value/Tag
	➢ **Pseudo Code:**
	If (en == 1 and in1 <= in2),
	eno = 1;
	Else eno = 0;

	**Assign** (Mathematical Symbols: Assign the tag a value) ➢ **Parameter:** ***out***: [Output] Tag ***in***: [Input] Value/Tag ➢ **Pseudo Code:** If en == 1 “out” is assigned with “in” eno = 1; Else eno = 0;
	**Assign2** ➢ **Parameter:** ***Out1***: [Output] Tag ***in1***: [Input] Value/Tag ***Out2***: [Output] Tag ***in2***: [Input] Value/Tag ➢ **Pseudo Code:** If en == 1 “out1” is assigned with “in1” “out2” is assigned with “in2” eno = 1; Else eno = 0;

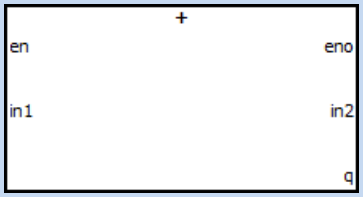
Function Block	Description and Pseudo Code
	**Assign4** ➢ **Parameter:** ***Out1***: [Output] Tag ***in1***: [Input] Value/Tag ***Out2***: [Output] Tag ***in2***: [Input] Value/Tag ***Out3***: [Output] Tag ***in3***: [Input] Value/Tag ***Out4***: [Output] Tag ***in4***: [Input] Value/Tag ➢ **Pseudo Code:** If en == 1 “out1” is assigned with “in1” “out2” is assigned with “in2” “out3” is assigned with “in3” “out4” is assigned with “in4” eno = 1; Else eno = 0;
	**Assign8** ➢ **Parameter:** ***Out1***: [Output] Tag ***in1***: [Input] Value/Tag ***Out2***: [Output] Tag ***in2***: [Input] Value/Tag ***Out3***: [Output] Tag ***in3***: [Input] Value/Tag ***Out4***: [Output] Tag ***in4***: [Input] Value/Tag ***Out5***: [Output] Tag ***in5***: [Input] Value/Tag ***Out6***: [Output] Tag ***in6***: [Input] Value/Tag ***Out7***: [Output] Tag ***in7***: [Input] Value/Tag ***Out8***: [Output] Tag ***in8***: [Input] Value/Tag ➢ **Pseudo Code:** If en == 1 “out1” is assigned with “in1” “out2” is assigned with “in2” “out3” is assigned with “in3” “out4” is assigned with “in4” “out5” is assigned with “in5” “out6” is assigned with “in6” “out7” is assigned with “in7” “out8” is assigned with “in8” eno = 1; Else eno = 0;

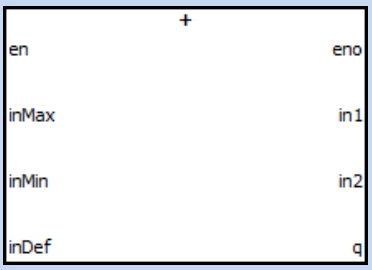
	**OnChange** (Check if the value has changed)
	➢ **Parameter:**
	in: [Input] Input Tag
	➢ **Pseudo Code:**
	If (en == 1 and “in” is changed)
	eno = 1;
	Else eno = 0;

Function Block	Description and Pseudo Code
	**InRange** (Check if the input value is within range) ➢ **Parameter:** ***inValue***: [Input] Input Value/Tag ***inMax***: [Input] The maximum Value/Tag of the input range ***inMin***: [Input] The minimum Value/Tag of the input range ➢ **Pseudo Code:** If en == 1 and (inMin <= inValu<=inMax), eno = 1; Else eno = 0;
	**OutRange** (Check if the input value is out of range) ➢ **Parameter:** ***inValue***: [Input] Input Value/Tag ***inMax***: [Input] The maximum Value/Tag of the input range ***inMin***: [Input] The minimum Value/Tag of the input range ➢ **Pseudo Code:** If en == 1 and (inValue < inMin or inValue >inMax), eno = 1; Else eno = 0;

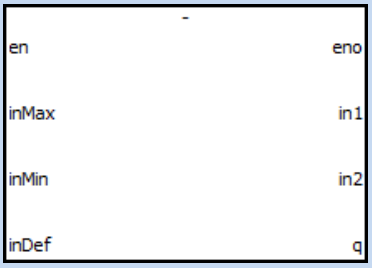
3.3.4.2 Math Group

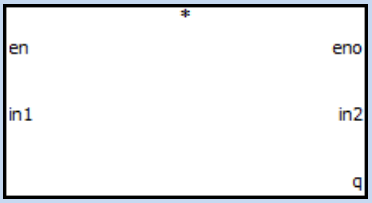
Function Block	Description and Pseudo Code

	**Add** (Mathematical Symbols: Addition)
	➢ **Parameter:**
	in1: [Input] Input Value/Tag
	in2: [Input] Input Value/Tag
	q: [Output] Tag
	➢ **Pseudo Code:**
	If en == 1
	q = in1 + in2;
	eno = 1;
	Else eno = 0;

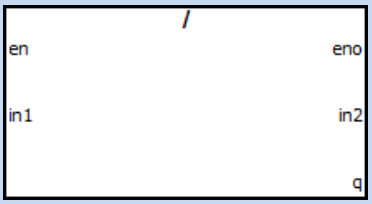
	**AddRange**
	➢ **Parameter:**
	inMax: [Input] Input Value/Tag
	inMin: [Input] Input Value/Tag
	inDef: [Input] Input Value/Tag
	in1: [Input] Input Value/Tag
	in2: [Input] Input Value/Tag
	q: [Output] Tag
	➢ **Pseudo Code:**
	If en == 1
	q = in1 + in2;
	if q < inMin or q >inMax
	q = inDef;
	eno = 1;
	Else eno = 0;

	**Sub** (Mathematical Symbols: Subtraction)
	➢ **Parameter:**
	in1: [Input] Input Value/Tag
	in2: [Input] Input Value/Tag
	q: [Output] Tag
	➢ **Pseudo Code:**
	If en == 1
	q = in1 - in2;
	eno = 1;
	Else eno = 0;

	**SubRange**
	➢ **參數:**
	inMax: [Input] Input Value/Tag
	inMin: [Input] Input Value/Tag
	inDef: [Input] Input Value/Tag
	in1: [Input] Input Value/Tag
	in2: [Input] Input Value/Tag
	q: [Output] Tag
	➢ **Pseudo Code:**
	If en == 1
	q = in1 - in2;
	if q < inMin or q >inMax
	q = inDef;
	eno = 1;
	Else eno = 0;

	**Mul** (Mathematical Symbols: Multiplication)
	➢ **Parameter:**
	in1: [Input] Input Value/Tag
	in2: [Input] Input Value/Tag
	q: [Output] Tag
	➢ **Pseudo Code:**
	If en == 1
	q = in1 * in2;
	eno = 1;
	Else eno = 0;

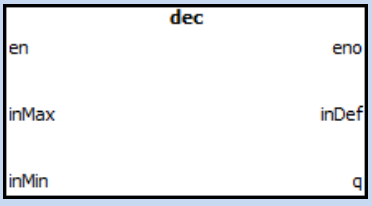
Function Block	Description and Pseudo Code

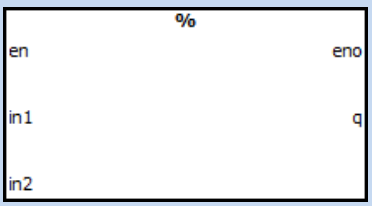
	**Div** (Mathematical Symbols: Division)
	➢ **Parameter:**
	in1: [Input] Input Value/Tag
	in2: [Input] Input Value/Tag
	q: [Output] Tag
	➢ **Pseudo Code:**
	If en == 1
	q = in1 / in2;
	eno = 1;
	Else eno = 0;

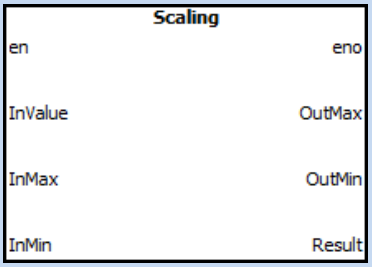
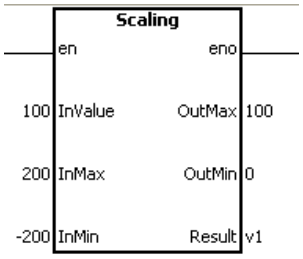
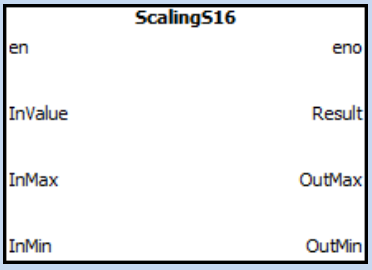
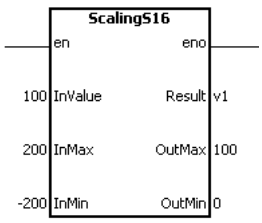
	**Inc** (Increment function) ➢ **Parameter:** ***in***: [Input/Output] Input Tag ➢ **Pseudo Code:** If en == 1 increment "in" by 1; eno = 1; Else eno = 0;

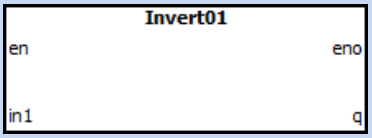
	**incRange**
	➢ **參數:**
	inMax: [Input] Input Value/Tag
	inMin: [Input] Input Value/Tag
	inDef: [Input] Input Value/Tag
	q: [Output] Tag
	➢ **Pseudo Code:**
	If en == 1
	q = q + 1;
	if q < inMin or q >inMax
	q = inDef;
	eno = 1;
	Else eno = 0;

	**Dec** (decrement function) ➢ **Parameter:** ***in***: [Input/Output] Input Tag ➢ **Pseudo Code:** If en == 1 decrement "in" by 1; eno=1; Else eno = 0;

	**decRange**
	➢ **參數:**
	inMax: [Input] Input Value/Tag
	inMin: [Input] Input Value/Tag
	inDef: [Input] Input Value/Tag
	q: [Output] Tag
	➢ **Pseudo Code:**
	If en == 1
	q = q - 1;
	if q < inMin or q >inMax
	q = inDef;
	eno = 1;
	Else eno = 0;

	**Mod** (Mathematical Symbols: Modulo)
	➢ **Parameter:**
	in1: [Input] Input Value/Tag
	in2: [Input] Input Value/Tag
	q: [Output] Tag
	➢ **Pseudo Code:**
	If en == 1
	q = in1 % in2;
	eno = 1;
	Else eno = 0;

Function Block	Description and Pseudo Code
	**Scaling** (Type: int) ➢ **Parameter:** ***InValue***: [Input] Input Value/Tag ***InMax***: [Input] The maximum Value/Tag of the input range ***InMin***: [Input] The minimum Value/Tag of the input range ***OutMax***: [Input] The maximum Value/Tag of the output range ***OutMin***: [Input] The minimum Value/Tag of the output range ***Result***: [Output] Scaled Tag ➢ **Examples:**  (InValue - InMin) / (InMax - InMin) = 0.75 Result(v1) = 0.75*(OutMax-OutMin)+OutMin = 75 ➢ **Pseudo Code:** If en == 1 Do the conversion function; eno = 1; Else eno = 0;
	**ScalingS16** (Type : short 16) ➢ **Parameter:** ***InValue***: [Input] Input Value/Tag ***InMax***: [Input] the maximum Value/Tag of the input range ***InMin:*** [Input] the minimum Value/Tag of the input range ***OutMax***: [Input] the maximum Value/Tag of the Output range ***OutMin***: [Input] the minimum Value/Tag of the Output range ***Result***: [Output] Scaled Tag ➢ **Examples:**  (InValue - InMin) / (InMax - InMin) = 0.75 Result(v1)= 0.75*(OutMax-OutMin)+OutMin = 75 ➢ **Pseudo Code:** If en == 1 Do the conversion function; eno = 1; Else eno = 0;

<table>
<tr><th>Function Block</th><th>Description and Pseudo Code</th></tr>
<tr><td> 
</td><td>

Invert01 (Logical operator: NOT)

➢ **Parameter:**

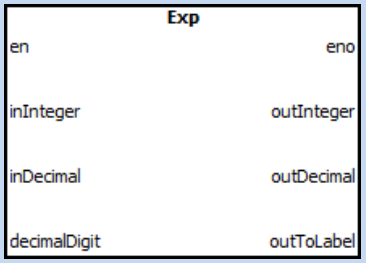
in1: [Input] Input Value/Tag, Invert the value between 0 and 1.
q: [Output] Tag

➢ **Examples:**

If “in1” is True => q = 0;
If “in1” is False => q = 1;

➢ **Pseudo Code:**

```
If en == 1,
   q = invert( in1 );
   eno = 1;
Else eno = 0;
```

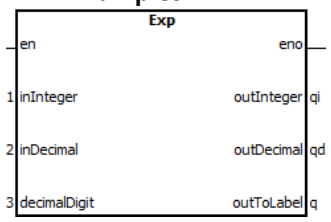
</td></tr>
<tr><td> 
</td><td>

Exp (e^x)

➢ **Parameter:**

inInteger: [Input] integer of the x
inDecimal: [Input] decimal of the x
decimalDigit: [Input] number of decimal places of the x (0~5)
outInteger: [Output] integer of the output
outDecimal: [Output] decimal of the output
outToLabel: [Output] The output is used to display in Label (used with property “DecimalDigits”).

➢ **Examples:**

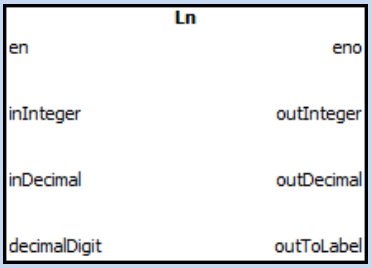


e^1.002 = 2.723

qi = 2
qd = 723
q =2723

➢ **Pseudo Code:**

```
If en == 1,
   Do the exponential function;
   eno = 1;
Else eno = 0;
```

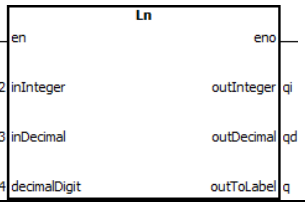
</td></tr>
<tr><td> 
</td><td>

Ln (ln(x))

➢ **Parameter:**

inInteger: [Input] integer of the x
inDecimal: [Input] decimal of the x
decimalDigit: [Input] number of decimal places of the x (0~5)
outInteger: [Output] integer of the output
outDecimal: [Output] decimal of the output
outToLabel: [Output] The output is used to display in Label (used with property “DecimalDigits”).

➢ **Example:**



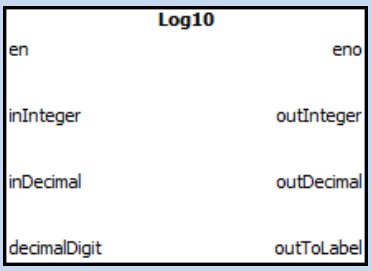
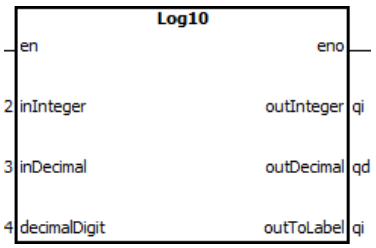
ln(2.0003)=0.6932

qi = 0
qd = 6932
q =6932

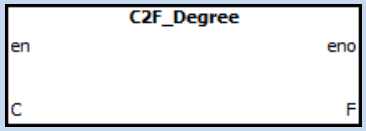
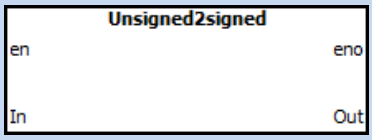
➢ **Pseudo Code:**

```
If en == 1,
   Do the natural
   logarithm function;
   eno = 1;
Else eno = 0;
```

</td></tr>
</table>

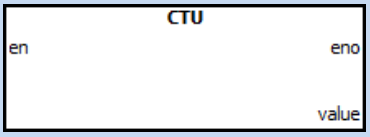
Function Block	Description and Pseudo Code
	**Log** (log(x)) ➢ **Parameter:** ***inInteger***: [Input] integer of the x ***inDecimal***: [Input] decimal of the x ***decimalDigit***: [Input] number of decimal places of the x (0~5) ***outInteger:*** [Output] integer of the output ***outDecimal:*** [Output] decimal of the output ***outToLabel:*** [Output] The output is used to display in Label (used with property "DecimalDigits"). ➢ **Example:**  log(2.0003) = 0.3010 qi = 0 qd = 3010 q =3010 ➢ **Pseudo Code:** If en == 1, Do the log(x) function; eno = 1; Else eno = 0;

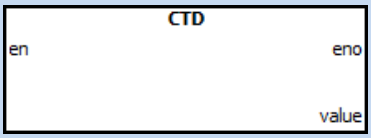
3.3.4.3 Convert Group

Function Block	Description and Pseudo Code
	**C2F_Degree** (Celsius to Fahrenheit Degree Converter) ➢ **Parameter:** ***C***: [Input] Input Value/Tag (Celsius degree) ***F***: [Output] Output Tag (Fahrenheit degree) ➢ **Pseudo Code:** If en == 1, F = (9/5)*C + 32; eno = 1; Else eno = 0;
	**Unsigned2signed** (Convert the value from unsigned to signed) ➢ **Parameter:** ***In***: [Input] Input Value/Tag ***Out***: [Output] Output Tag ➢ **Pseudo Code:** If en == 1, Do the conversion function; eno = 1; Else eno = 0;

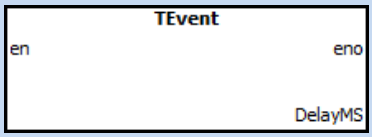
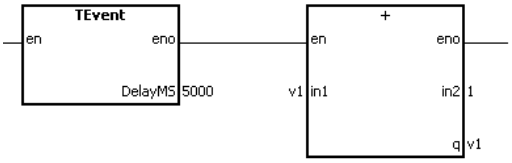
3.3.4.4 Counter Group

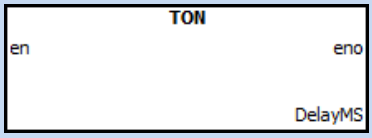
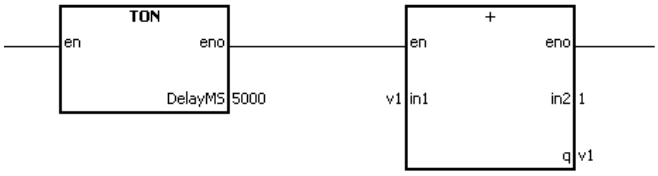
Function Block	Description and Pseudo Code

	**CTU** (The counter counts up until equals the value)
	➢ **Parameter:**
	***value*:** [Input] Input Value/Tag
	➢ **Pseudo Code:**
	If en == 1,
	Count=0;
	Loop: Count up until count>=value;
	During counting, eno = 0;
	When End, eno = 1;
	Else
	Reset count to 0,
	eno = 0;
	Note: the counting period depends on the number of rungs

	**CTD** (The counter counts down until equals zero)
	➢ **Parameter:**
	***value*:** [Input] Input Value/Tag
	➢ **Pseudo Code:**
	If en == 1,
	Count=value;
	Loop: Count down until count<=0;
	During counting, eno = 0,
	When End, eno = 1;
	Else
	Reset count to value,
	eno = 0;
	Note: the counting period depends on the number of rungs

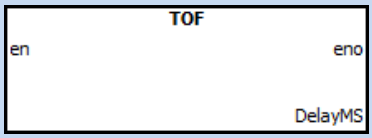
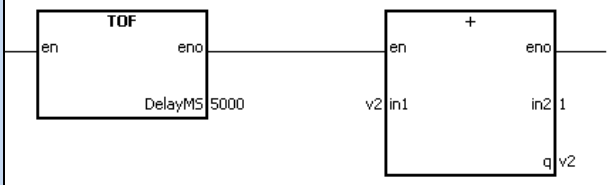
3.3.4.5 Timer Group

<table>
<tr><th>Function Block</th><th>Description and Pseudo Code</th></tr>
<tr><td> 
</td><td>
TimerEventH
(Period Timer Event, Triggered once after each timer, unit=ms)
➢ Parameter:
DelayMS: [Input] Set the interval time. The timer resolution is about 10 ms.
➢ Examples:
Calculate v1 = v1 + 1 for each 5 seconds cycle


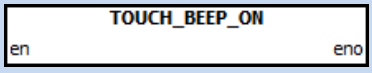
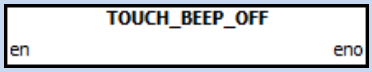
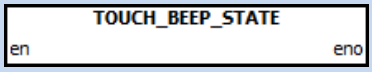
➢ Pseudo Code:
If en == 1
  If timer < DelayMS
    Start the timer;
    eno = 0;
  Else
    Timer = 0;
    eno = 1;
Else
  Reset the timer;
  eno = 0;
</td></tr>
<tr><td> 
</td><td>
TON (Timer On, Continued trigger after the timer, unit=ms)
➢ Parameter:
DelayMS: [Input] Set the interval time. The timer resolution is about 10 ms.
➢ Examples:
After 5 seconds, calculate v1 = v1 + 1 for each count cycle


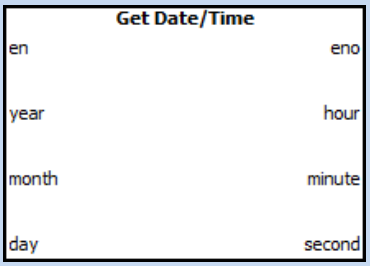
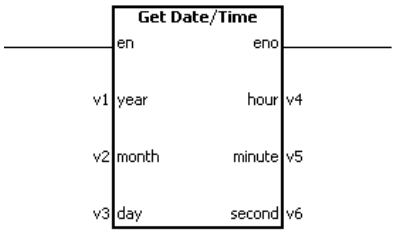
➢ Pseudo Code:
If en == 1
  Start the timer if not;
  Stop the timer when timer>=DelayMS;
  If timer runs
    eno = 0;
  else
    eno = 1;
Else
  Reset the timer;
  eno = 0;
</td></tr>
</table>

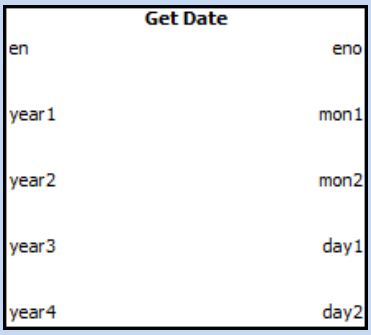
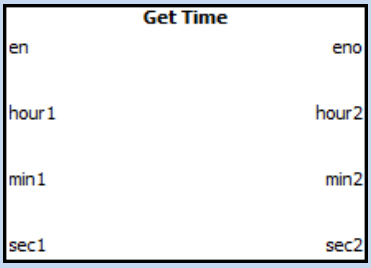
Function Block	Description and Pseudo Code

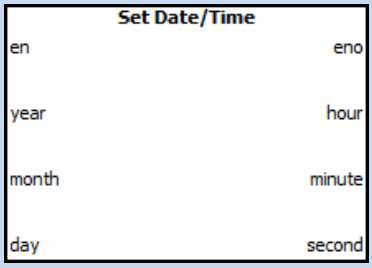
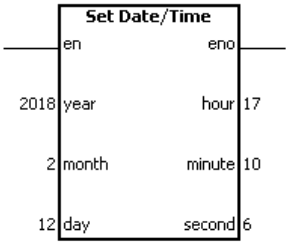
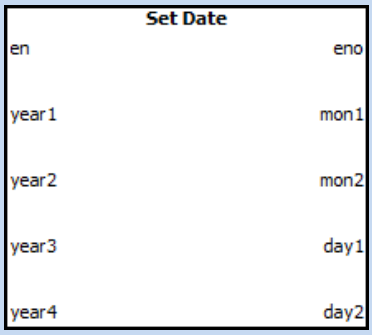
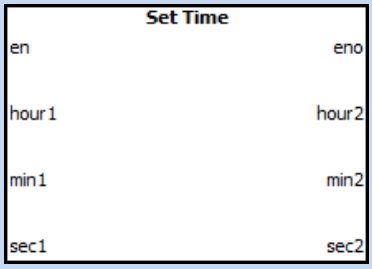
	**TOF** (Timer Off, Trigger only during timer, unit=ms)
	➢ **Parameter:**
	***DelayMS*: [Input]** Set the interval time. The timer resolution is about 10 ms.
	➢ **Examples:**
	Only in 5 seconds, calculate v2 = v2 + 1 for each count cycle
	 
	➢ **Pseudo Code:**
	If en == 1,
	Start the timer if not;
	Stop the timer when timer>=DelayMS;
	If timer runs
	Eno=1;
	Else
	Eno=0;
	Else
	Reset the timer;
	eno = 0;

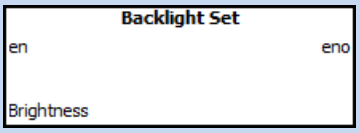
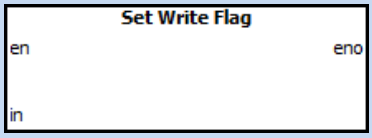
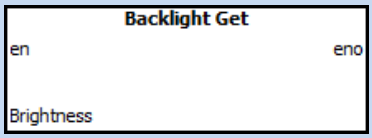
3.3.4.6 System Group

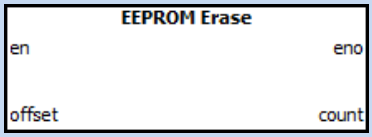
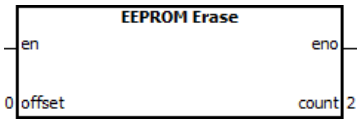
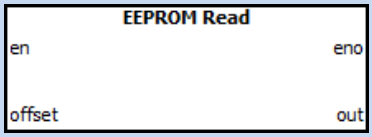
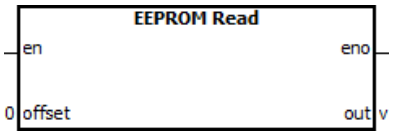
Function Block	Description and Pseudo Code
	**Beep** (Sound the beep.) ➢ **Pseudo Code:** If en == 1, beep and eno = 1; Else eno = 0;
	**TOUCH_BEEP_ON** (Beep when user click on the screen.) ➢ **Pseudo Code:** If en == 1, Set the beep function to ON; eno = 1; Else eno = 0;
	**TOUCH_BEEP_OFF** (Disable the beep function) ➢ **Pseudo Code:** If en == 1, Set the beep function to OFF; eno = 1; Else eno = 0;
	**TOUCH_BEEP_STATE** (Check the beep function state) ➢ **Pseudo Code:** If en == 1, Beep function is ON, eno =1; Beep function if OFF, eno =0; Else eno = 0;

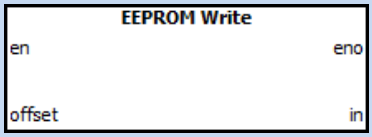
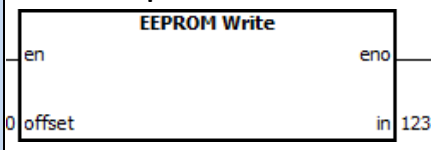
<table>
<tr><th>Function Block</th><th>Description and Pseudo Code</th></tr>
<tr><td> 
</td><td>
<h2>Get_Date_Time</h2>
(Get the date and time from the RTC chip on the TouchPAD devices.)
➢ Parameter:
year: [Output] Represent the year
month: [Output] Represent the month
day: [Output] Represent the day
hour: [Output] Represent the hour
minute: [Output] Represent the minute
second: [Output] Represent the second
➢ Examples:


v1=year
v2=month
v3=day
v4=hour
v5=minute
v6=second
➢ Pseudo Code:
If en == 1,
Get the RTC’s date and time;
eno =1;
Else eno = 0;
</td></tr>
<tr><td> 
</td><td>
<h2>Get_Date_Digit</h2>
(Get the date from the RTC chip on the TouchPAD devices.)
➢ Parameter:
year1: [Output] year(thousands digit)
year2: [Output] year(hundreds digit)
year3: [Output] year(tens digit)
year4: [Output] year(ones digit)
mon1: [Output] month (tens digit)
mon2: [Output] month (ones digit)
day1: [Output] day (tens digit)
day2: [Output] day (ones digit)
➢ Pseudo Code:
If en == 1,
Get the RTC’s date;
eno =1;
Else eno = 0;
</td></tr>
<tr><td> 
</td><td>
<h2>Get_Time_Digit</h2>
(Get the time from the RTC chip on the TouchPAD devices.)
➢ Parameter:
hour1: [Output] hour (tens digit)
min1: [Output] minute (tens digit)
sec1: [Output] second (tens digit)
hour2: [Output] hour (ones digit)
min2: [Output] minute (ones digit)
sec2: [Output] second (ones digit)
➢ Pseudo Code:
If en == 1,
Get the RTC’s time;
eno =1;
Else eno = 0;
</td></tr>
</table>

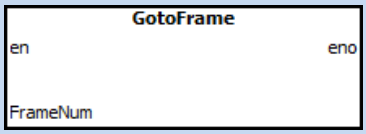
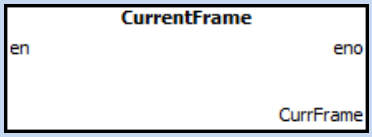
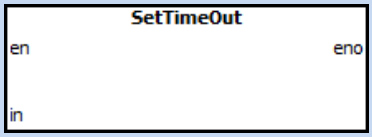
Function Block	Description and Pseudo Code
	**Set_Date_Time** (Set the date and time to the RTC chip on the TouchPAD devices.) ➢ **Parameter:** ***year***: [Input] Represent the year ***hour***: [Input] Represent the hour ***month***: [Input] Represent the month ***minute***: [Input] Represent the minute ***day***: [Input] Represent the day ***second***: [Input] Represent the second ➢ **Examples:**  Set Date and time as 2018-2-12,17:10:06 ➢ **Pseudo Code:** If en == 1, Set the date and time to RTC; eno =1; Else eno = 0;
	**Set_Date_Digit** (Set the date to the RTC chip on the TouchPAD devices.) ➢ **Parameter:** ***year1***: [Input] year(thousands digit) ***mon1***: [Input] month (tens digit) ***year2***: [Input] year(hundreds digit) ***mon2***: [Input] month (ones digit) ***year3***: [Input] year(tens digit) ***day1***: [Input] day (tens digit) ***year4***: [Input] year(ones digit) ***day2***: [Input] day (ones digit) ➢ **Pseudo Code** If en == 1, Set the date to RTC; eno =1; Else eno = 0;
	**Set_Time_Digit** (Set the time to the RTC chip on the TouchPAD devices.) ➢ **Parameter:** ***hour1***: [Input] hour (tens digit) ***hour2***: [Input] hour (ones digit) ***min1***: [Input] minute (tens digit) ***min2***: [Input] minute (ones digit) ***sec1***: [Input] second (tens digit) ***sec2***: [Input] second (ones digit) ➢ **Pseudo Code** If en == 1, Set the time to RTC; eno =1; Else eno = 0;

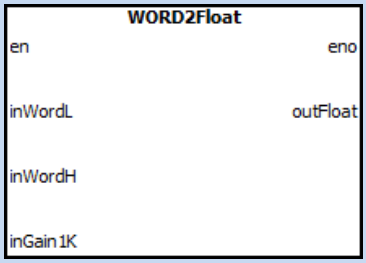
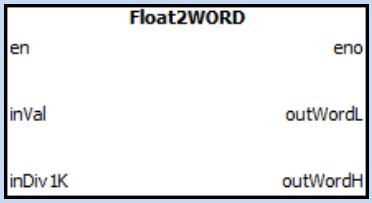
Function Block	Description and Pseudo Code
	**Backlight Set** (Set the brightness of the TouchPAD series.) ➢ **Parameter:** ***Brightness*: [Input]** Specify the brightness of TouchPAD. Range: 0 ~ 255. 0=the darkest, ..., 255=the brightest. ➢ **Pseudo Code:** If en == 1 Set the brightness value as “Brightness”; eno =1; Else eno = 0;
	**Set_Write_Flag** (Set the write flag of a I/O tag, so the I/O tag should be updated to remote device at next I/O scan.) ➢ **Parameter:** ***in*: [Input/Output]** A I/O tag. ➢ **Pseudo Code:** If en == 1 Set the “write” flag of a I/O tag; eno =1; Else eno = 0;
	**BacklightGet** (Get the brightness of the TouchPAD series.) ➢ **Parameter:** ***Brightness*: [Output]** Get the brightness of TouchPAD ➢ **Pseudo Code:** If en == 1 “Brightness”=brightness value; eno =1; Else eno = 0;

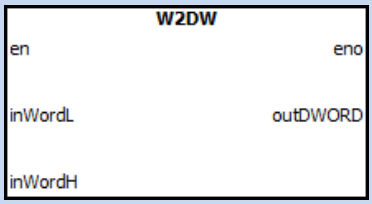
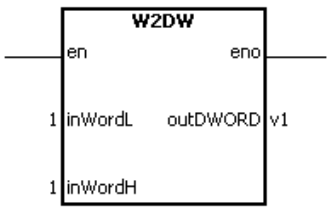
Function Block	Description and Pseudo Code
	**EepromErase** (Erase the content of the EEPROM) ➢ **Parameter:** ***offset*: [Input]** Specify the offset address of the EEPROM to be erased. Possible range: 0 ~ 511. **Note:** offset + count cannot be larger than 512 ***count*: [Input]** Specify the number of 32-bit data space that is to be erased. Possible range: 1 ~ 512. **Note:** offset + count cannot be larger than 512 ➢ **Examples:**  Erase two 32-bit space values starting from the area "0". ➢ **Pseudo Code:** If en == 1 Erase data of the EEPROM; eno =1; Else eno = 0; **Note:** There is 100,000 times write limitation for the EEPROM. Frequently usage may damage the EEPROM.
	**EepromRead** (Get 32-bit data from the EEPROM.) ➢ **Parameter:** ***offset*: [Input]** Specify the offset address of the EEPROM to be read. Possible range: 0 ~ 511. ***out*: [Output]** Specify the value to store the data got from the EEPROM. ➢ **Examples:**  v = Read the value of 32-bit space from area “0”. ➢ **Pseudo Code:** If en == 1 Read data from the EEPROM; eno =1; Else eno = 0;

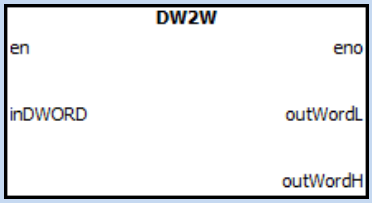
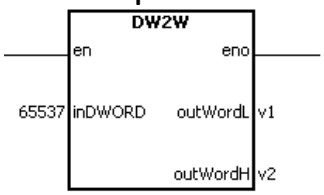
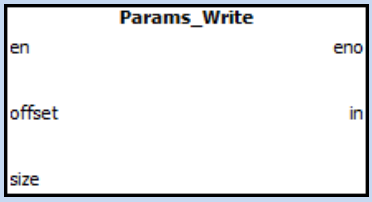
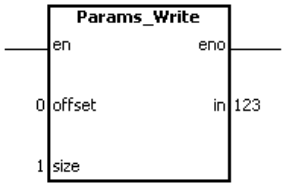
Function Block	Description and Pseudo Code
	**EepromWrite** (Write 32-bit data to the EEPROM.) ➢ **Parameter:** ***offset*: [Input]** Specify the offset address of the EEPROM to be written to. Possible range: 0 ~ 511. ***in*: [Input]** Specify the value which is used to write to the EEPROM. ➢ **Examples:**  Write the value "123" to area “0”. ➢ **Pseudo Code:** If en == 1, Write data to the EEPROM; eno = 1; Else eno = 0; **Note:** There is 100,000 times write limitation for the EEPROM. Frequently usage may damage the EEPROM.

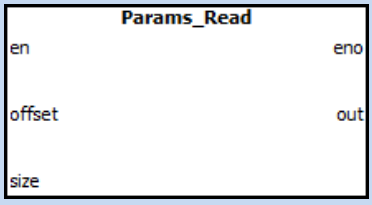
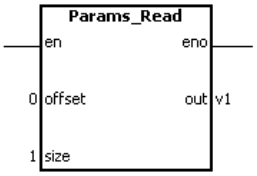
3.3.4.7 User_define Group

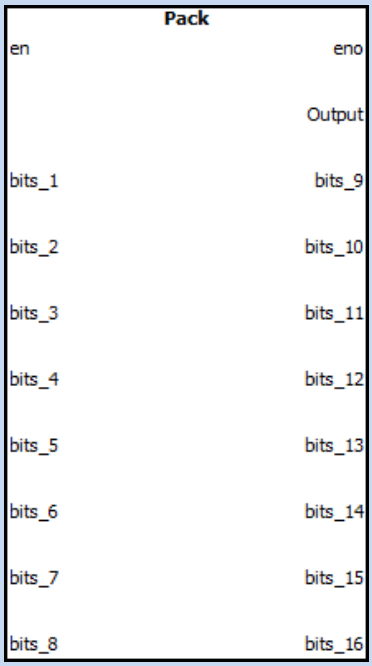
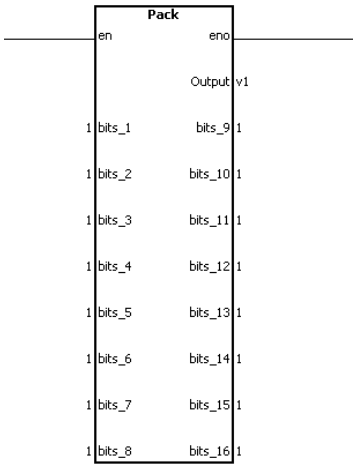
Function Block	Description and Pseudo Code
	**GotoFrame** (Go to the frame number) ➢ **Parameter:** ***FrameNum*: [Input]** Set the frame number. The frame number is indexed from 1, not depending on ID number. ➢ **Pseudo Code:** If en == 1 Go to the frame number; eno =1; Else eno = 0;
	**CurrentFrame** (Get the current frame number) ➢ **Parameter:** ***CurrFrame*: [Output]** The current frame number ➢ **Pseudo Code:** If en == 1 CurrFrame= current frame number; eno =1; Else eno = 0;
	**SetTimeOut** (Sets the uart fucntions's timeout timer.) ➢ **Parameter:** ***in*: [input]** Timeout value/Tag ➢ **Pseudo Code:** If en == 1 TimeOut value = “in”; eno =1; Else eno = 0;

Function Block	Description and Pseudo Code
	**WORD2Float** (Convert 2 WORD to float) ➢ **Parameter:** ***inWordL***: [Input] Low word Value ***inWordH***: [Input] High word Value ***inGain1K***: [Input] the result *1000 or not ***outFloat***: [Output] float Tag ➢ **Pseudo Code:** If en == 1, Do the conversion function; eno = 1; Else eno = 0;
	**Float2WORD** (Convert float to 2 WORD) ➢ **Parameter:** ***inVal***: [Input] float Value/Tag ***inDiv1K***: [Input] the result /1000 or not ***outWordL***: [Output] Low word Tag ***outWordH***: [Output] High word Tag ➢ **Pseudo Code:** If en == 1, Do the conversion function; eno = 1; Else eno = 0;

	**WORD2DWORD**
	(Convert 2 WORD to a single DWORD)
	➢ **Parameter:**
	inWordL: [Input] High word Value
	inWordH: [Input] Low word Value
	outDWORD: [Output] DWORD Tag
	 
	➢ **Examples:**
	unsigned char v1[4];
	v1[1]=(inWordL>>8) & 0xFF;
	v1[0]=inWordL & 0xFF;
	v1[3]=(inWordH>>8) & 0xFF;
	v1[2]=inWordH & 0xFF;
	You can get v1 = 65537;
	➢ **Pseudo Code:**
	If en == 1,
	Do the conversion function;
	eno = 1;
	Else eno = 0;

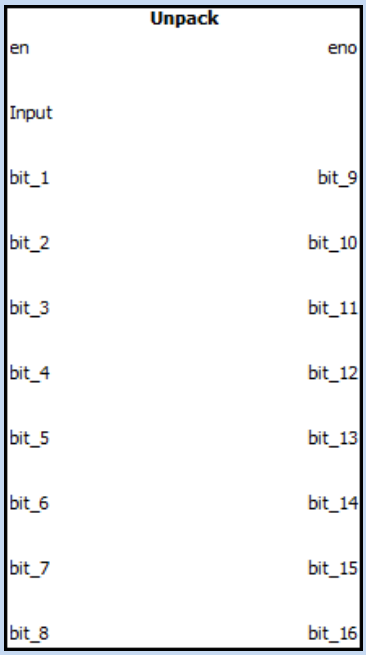
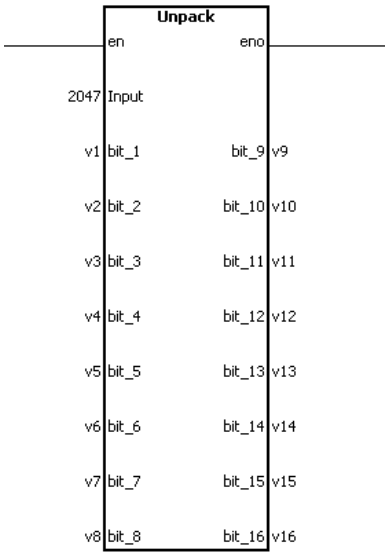
Function Block	Description and Pseudo Code
	**DWORD2WORD** (Convert a single DWORD to 2 WORD) ➢ **Parameter:** ***inDWORD*:** [Input] DWORD Value/Tag ***outWordL*:** [Output] High-word tag ***outWordH*:** [Output] Low-word tag ➢ **Examples:**  V1 = inDWORD & 0xFFFF = 1 V2=(inDWORD>>16) & 0xFFFF =1 ➢ **Pseudo Code:** If en == 1, Do the conversion function; eno = 1; Else eno = 0;
	**Params_Write** (Set data to the 256-byte parameter area in the MCU (MicroController Unit) internal flash.) ➢ **Parameter:** ***offset*:** [Input] Specify the offset to the base of the 256-byte parameter area to write data to it. Possible range: 0 ~ 255. **Note:** iOffset + iSize cannot be larger than 256 ***size*:** [Input] Specify the number of bytes to write to the 256-byte parameter area. Possible range: 1 ~ 256. **Note:** iOffset + iSize cannot be larger than 256 ***in*:** [Input] Specify the value which is used to write to the 256-byte parameter area. ➢ **Examples:**  Write the value "123" to area “0”, the space size is “1”. ➢ **Pseudo Code:** If en == 1, Write data to internal flash; eno = 1; Else eno = 0; There is 100,000 times write limitation for the flash. Frequently usage may damage the flash.

<table>
<tr><th>Function Block</th><th>Description and Pseudo Code</th></tr>
<tr><td> 
</td><td>
<h2>Params_Read</h2>
(Get data from the 256-byte parameter area in the MCU (MicroController Unit) internal flash.)
➢ Parameter:
offset: [Input] Specify the offset to the base of the 256-byte parameter area to read data from it. Possible range: 0 ~ 255.
Note: iOffset + iSize cannot be larger than 256
size: [Input] Specify the size of the data to read from the 256-byte parameter area. Possible range: 1 ~ 256.
Note: iOffset + iSize cannot be larger than 256
out: [Output] Specify the value to store the data got from the 256-byte parameter area
➢ Examples:


v1 = Read the value of space size “1” from area “0”.
➢ Pseudo Code:
If en == 1,
Read data from internal flash;
eno = 1;
Else eno = 0;
</td></tr>
<tr><td> 
</td><td>
<h2>Pack</h2> (Binary to Decimal)
Convert 16 tags to a single tag with 16-bit data.
➢ Parameter:
Bits_1: [Input] Input Value/Tag
Bits_2: [Input] Input Value/Tag
……
Bits_15: [Input] Input Value/Tag
Bits_16: [Input] Input Value/Tag
Output: [Output] Output Tag


➢ Examples:
for(i=0 ; i<16 ; i++)
v1= v1+ (bits_[i]<<i);
➔ v1=65535
➢ Pseudo Code:
If en == 1,
for(i=0 ; i<16 ; i++)
Output= Output+ (bits_[i]<<i);
eno = 1;
Else eno = 0;
</td></tr>
</table>

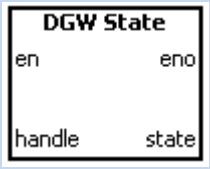
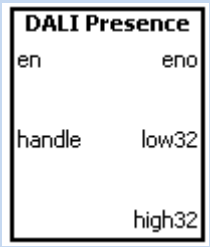
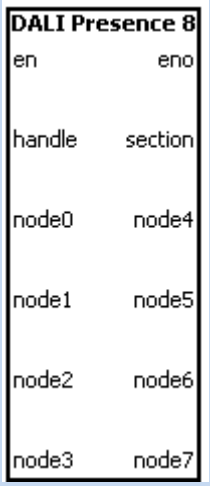
Function Block	Description and Pseudo Code

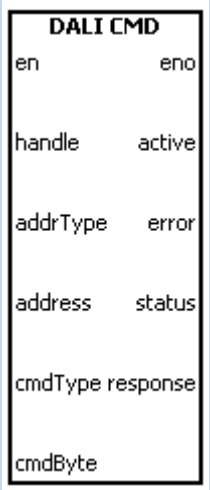
	**UnPack** (Decimal to Binary)
	Convert a single tag with 16-bit data to 16 tags.
	➢ **Parameter:**
	Input: [Input] Input Value/Tag
	Bits_1: [Output] Output Tag
	Bits_2: [Output] Output Tag
	……
	Bits_15: [Output] Output Tag
	Bits_16: [Output] Output Tag
	➢ **Examples:**
	 
	v1=v2=v3=v4=v5=v6=v7=v8=v9=v10=v11=1;
	v12=v13=v14=v15=v16=0
	➢ **Pseudo Code:**
	If en == 1,
	bit_1 = (input>>0)&1;
	bit_2 = (input>>1)&1;
	bit_3 = (input>>2)&1;
	…..
	bit_16 = (input>>15)&1;
	eno = 1;
	Else eno = 0;

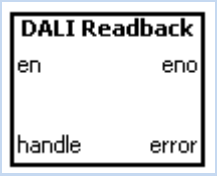
3.3.4.8 VPD-130 Group

Function Block	Description and Pseudo Code
	**GetPanelKey** (Only support VPD series) ➢ **Parameter:** ***out*: [Output]** A tag to store the value of pressed panel key. ➢ **Pseudo Code:** If en == 1, “out” = the panel key number; eno = 1; Else eno = 0;
	**ShowPanelLed** (Setting the LED state) ➢ **Pseudo Code:** LED state = en If en == 1, LED state = ON; If en == 0, LED state = OFF;

3.3.4.9 DGW-521 Group

Function Block	Description and Pseudo Code
	**DGW State**(Gets state of the DGW-521 module) ➢ **Parameter:** **handle:** [Intput] Handle value of the DGW-521 module. **state:** [Output] >= 0, State value of the DGW-521 module. < 0, error code.
	**DALI Scan** (Scan DALI slaves connected on the DGW-521) ➢ **Parameter:** **handle:** [Intput] Handle value of the DGW-521 module. **active:** [Input] Set 1 to start. This is auto-cleared after finished. **error:** [Output] Error code. return (eno): 1 = finished and the $active is auto-cleared. 0 = idle, failure or processing.
	**DALI Presence**(Gets DALI presence (slave devices) of the DGW-521 module, and then store the presence-state into two 32-bit tags.) ➢ **Parameter:** **handle:** [Intput] Handle value of the DGW-521 module. **Low32:** [Output] State of DALI presence. **High32:** [Output] State of DALI presence.
	**DALI Presence 8**(Gets DALI presence (slave devices) of the DGW-521 module, and then store the presence-state into individual nodes.) ➢ **Parameter:** **handle:** [Intput] Handle value of the DGW-521 module. **section:** [Input] Set Section Section 0 = address 0 - 7, Section 1 = address 8 - 15. Section 2 = address 16 - 23, Section 3 = address 24 - 31. Section 4 = address 32 - 39, Section 5 = address 40 - 47. Section 6 = address 48 - 55, Section 7 = address 56 - 63. node0 ~ node7: [Output] State of DALI presence (slave devices). 1 = presence. 0 = absent.

Function Block	Description and Pseudo Code
	**DALI Scene**(Sends DALI goto-scene command and receives response in non-blocking mode. This FB should be continuously called after setting $active = 1, until $active back to 0 (finished / idle).) ➢ **Parameter:** **handle:** [Intput] Handle value of the DGW-521 module. **addrType:** [Input] Address type 0=Lamp Address, 1=Group Address, 2=Broadcast. **addrss:** [Input] DALI address 0-63 for short address, 0-15 for group address. **scene:** [Input] 0-15. **active:** [Input] Set 1 to start. This is auto-cleared after finished. **error:** [Output] Error code. **status:** [Output] Status of the command. **response:** [Output] Response from the DALI slave. return (eno): 1 = finished and the $active is auto-cleared. 0 = idle, failure or processing. Refer to the DGW-521 user manual for more information about $scene, $status and $response.
	**DALI CMD**(Sends DALI command and receives response in non-blocking mode. This FB should be continuously called after setting $active = 1 until $active back to 0 (finished / idle)). ➢ **Parameter:** **handle:** [Intput] Handle value of the DGW-521 module. **addrType:** [Input] Address type 0=Lamp Address, 1=Group Address, 2=Broadcast. **addrss:** [Input] DALI address 0-63 for short address, 0-15 for group address. **cmdType:** [Input] Command type 0=Direct Lamp Power Level, 1=DALI Command. **cmdByte:** [Input] Value of lamp power level, or DALI command depending on cmdType. **active:** [Input] Set 1 to start. This is auto-cleared after finished. **error:** [Output] Error code. **status:** [Output] Status of the command. **response:** [Output] Response from the DALI slave. Refer to the DGW-521 user manual for details about $cmdByte, $status and $response.

Function Block	Description and Pseudo Code
	**DALI Raw**(Sends raw DALI command and receives response in non-blocking mode. This FB should be continuously called after setting $active = 1 until $active back to 0 (finished / idle).) ➢ **Parameter:** **handle:** [Intput] Handle value of the DGW-521 module. **active:** [Input] Set 1 to start. This is auto-cleared after finished. **addrByte:** [Input] Address byte (including address, type, broacast settings) of the DALI command. **cmdByte:** [Input] Command byte can be lamp power level or DALI command depening on the $addrByte. **error:** [Output] Error code. **status:** [Output] Status of the command. **response:** [Output] Response from the DALI slave. Refer to the DGW-521 user manual for more information about $addrByte, $cmdByte, $status and $response.
	**DALI Readback**(Sends DALI command to get current level of lamps. The lamp level is then updated (stored) in DGW-521 control block later.) ➢ **Parameter:** **handle:** [Intput] Handle value of the DGW-521 module. **error:** [Output] Error code.

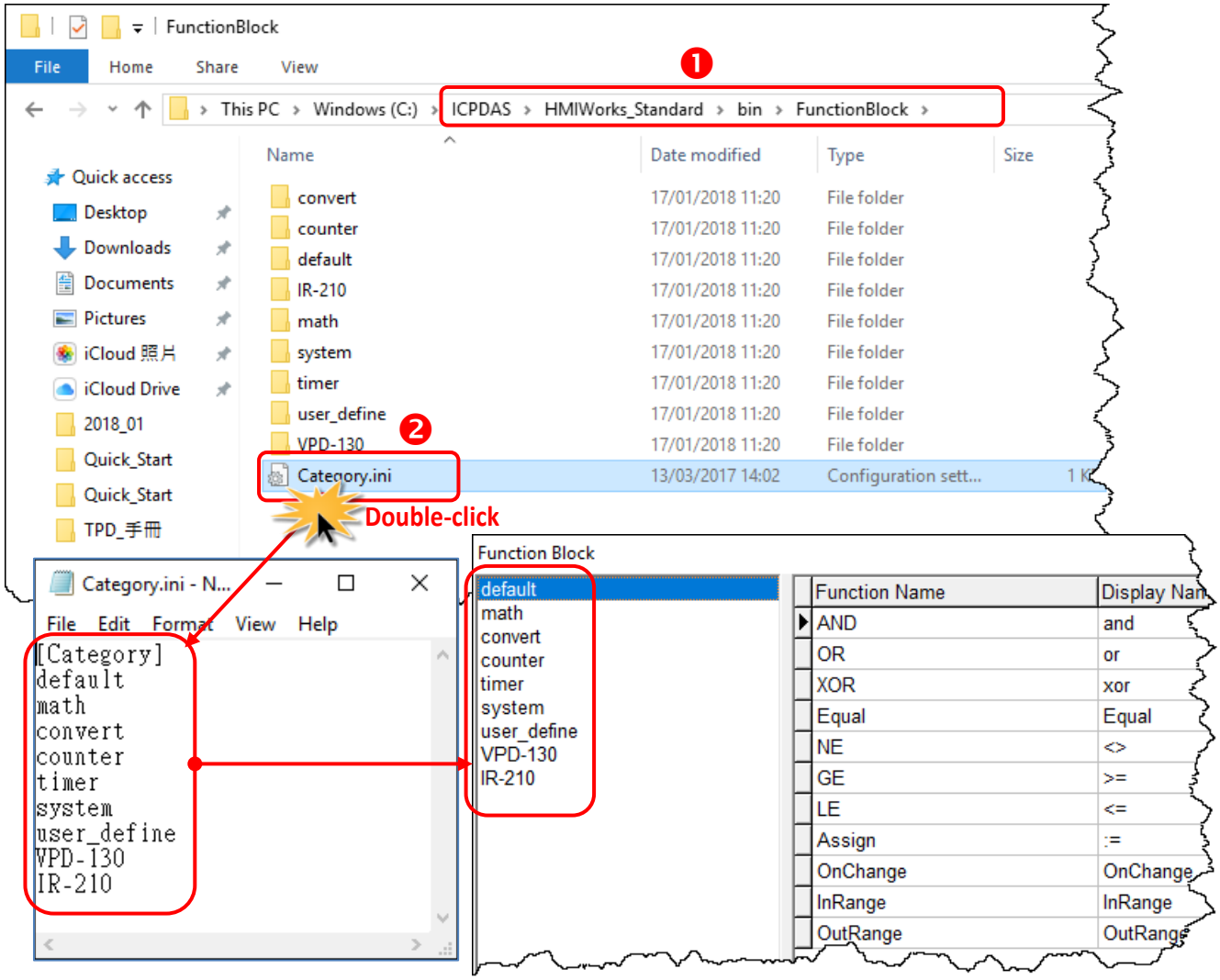
3.3.5 User-Defined Function Block

Why should we use function block? There may be cases that using only ladders is too complex. At that time, using a function block may be a good choice.

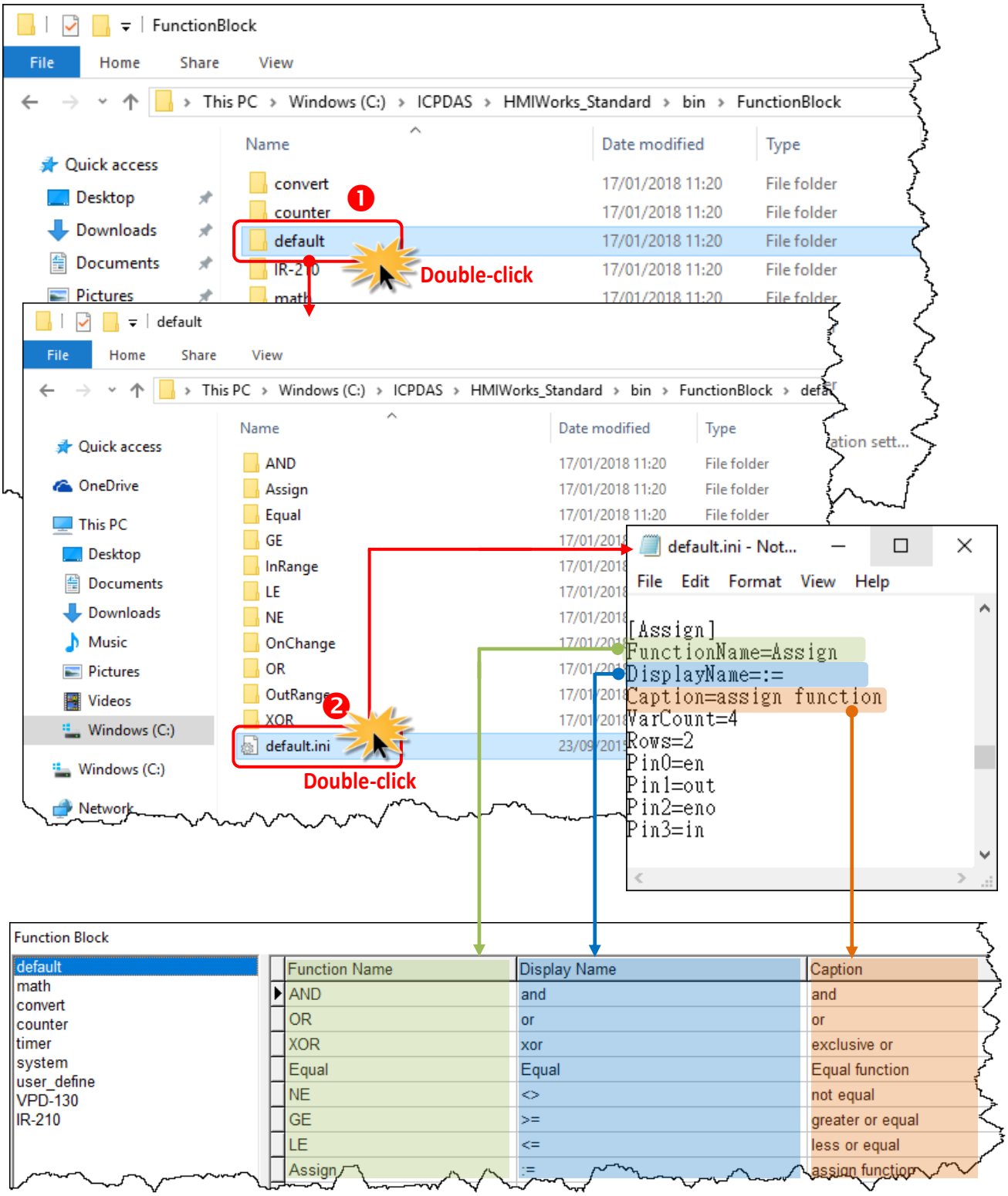
To know how to add a user-defined function block, we first explain how HMIWorks uses these function blocks. Take **"Assign"** function block in the **"default"** group for example.

3.3.5.1 How HMIWorks Uses Function Blocks

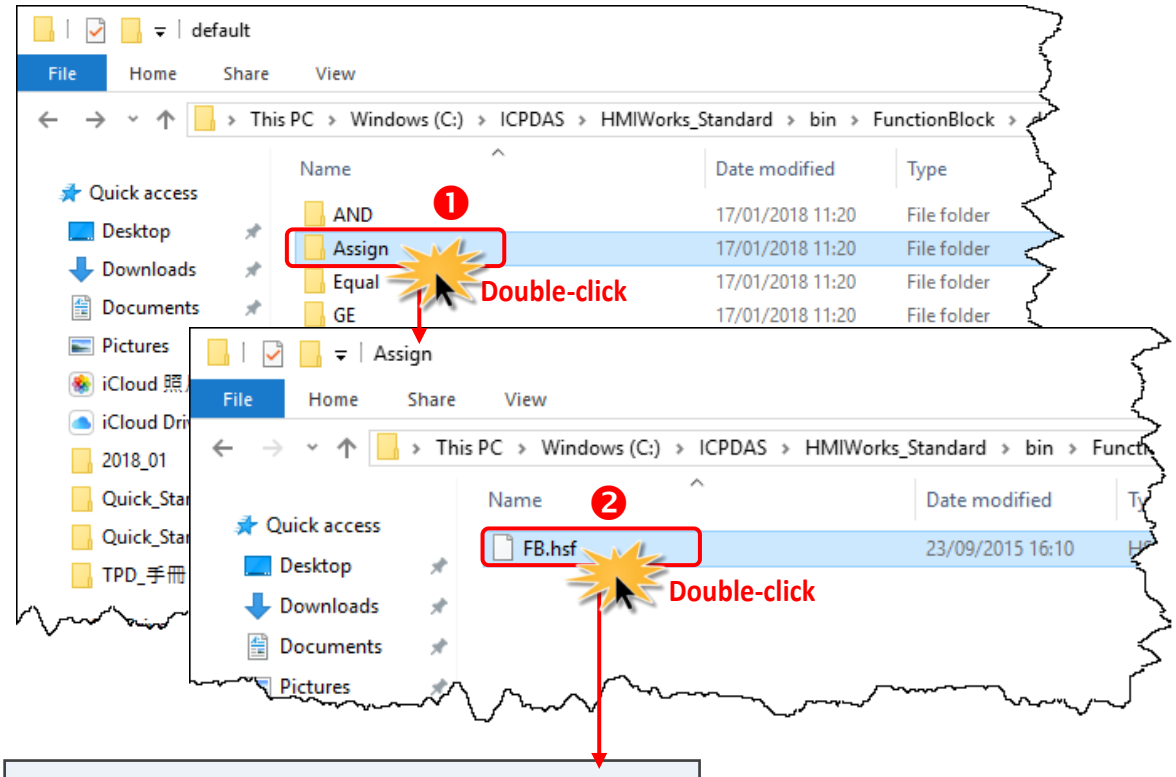
1. Go to the installation path of the HMIWorks software. In the sub-directory, **"bin\FunctionBlock"**, of that installation path, open the file **"Category.ini"** to load the groups.



2. If we choose the **“default”** group, then HMIWorks opens the matching-name sub-directory and then loads from the matching-name **“.ini”** file in that sub-directory. That is, the **“default.ini”** in the sub-directory **“default”**.



3. Double-click on the **"Assign"** to use it in the **Ladder Designer**. The **Ladder Designer** uses the logics defined in the file **"FB.hsf"** in the sub-directory **"Assign"**. **FB.hsf** is based on the C language. The following figure explains what **FB.hsf** of the **"Assign"** function does.

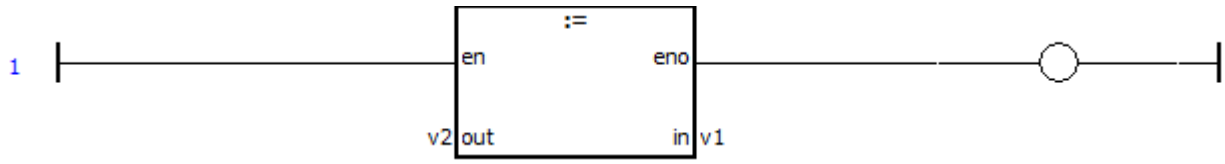



```
if (!en) return 0;

if (VAR_VALUE($out) == VAR_VALUE($in)) return 1;

#ifdef LAD_MODS_VER_1
VAR_SET($out, VAR_VALUE($in));
#else
VAR_VALUE($out) = VAR_VALUE($in);
#endif

VAR_SET_DIRTY($out);
VAR_FLAG_SET_WRITE($out);
return 1;
```

If **en** is set to low, **eno** is set to low and return.

If **v1** is equal to **v2**, **eno** is set to high and return. **v2** is assigned with **v1**.

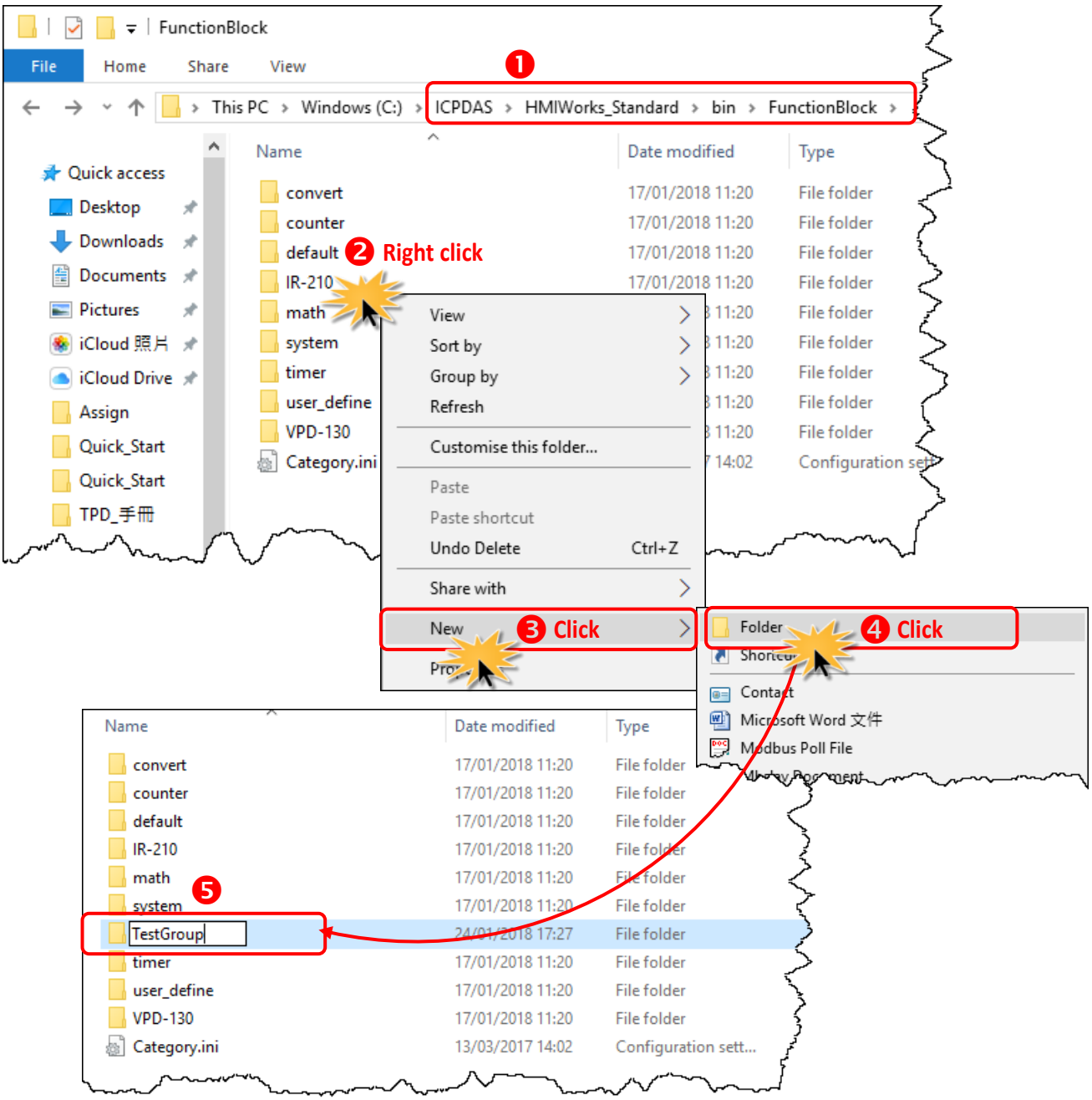
Show the value of **v2** to the associated widget on TouchPAD.
eno is set to high and return.



3.3.5.2 Adding a User-Defined Function Block

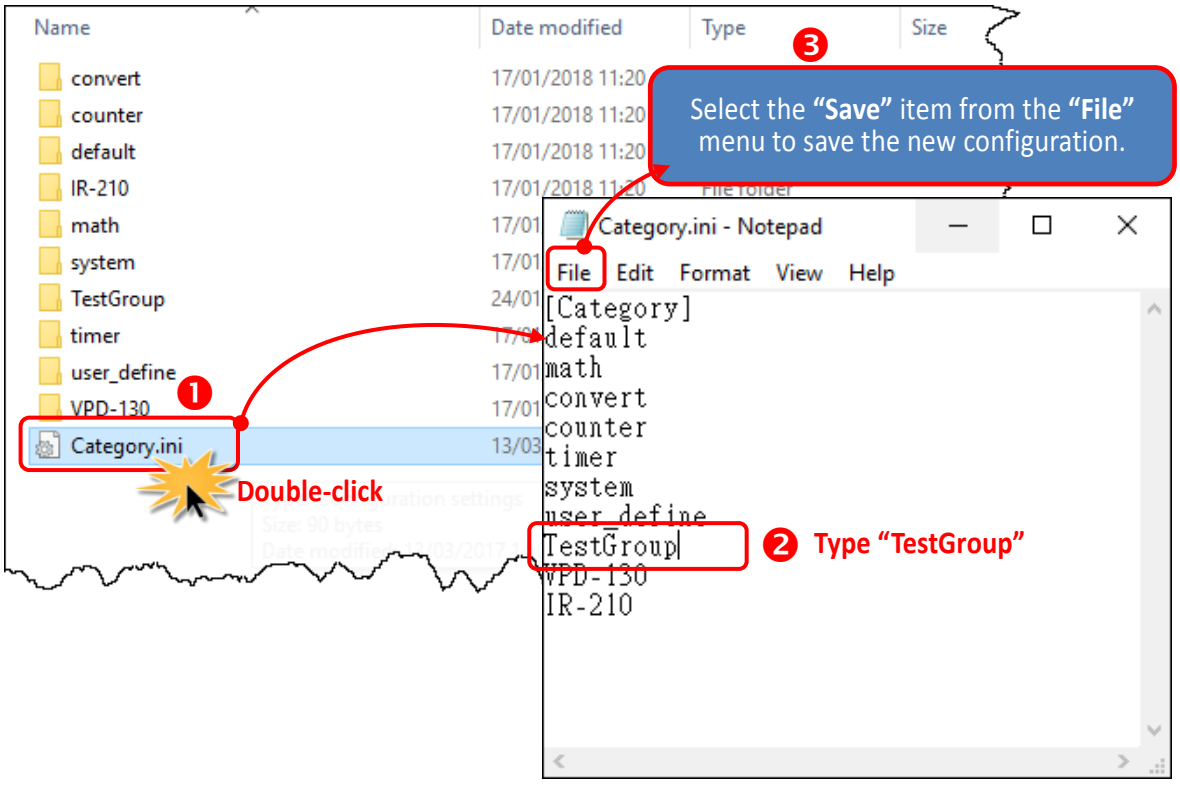
Now, we introduce how to add a user-defined function block.

Step 1: Go to the installation path of HMIWorks. In the sub-directory of **"bin\FunctionBlock"**, create a new directory **"TestGroup"**.

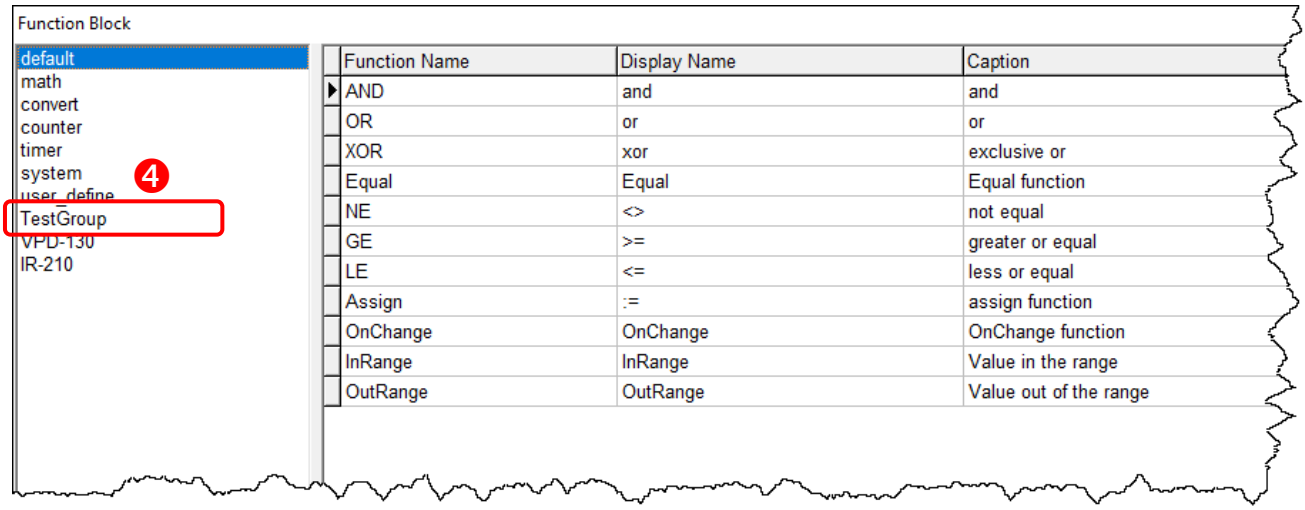


Step 2: Open the file **"Category.ini"** to add a new item to represent the new group.

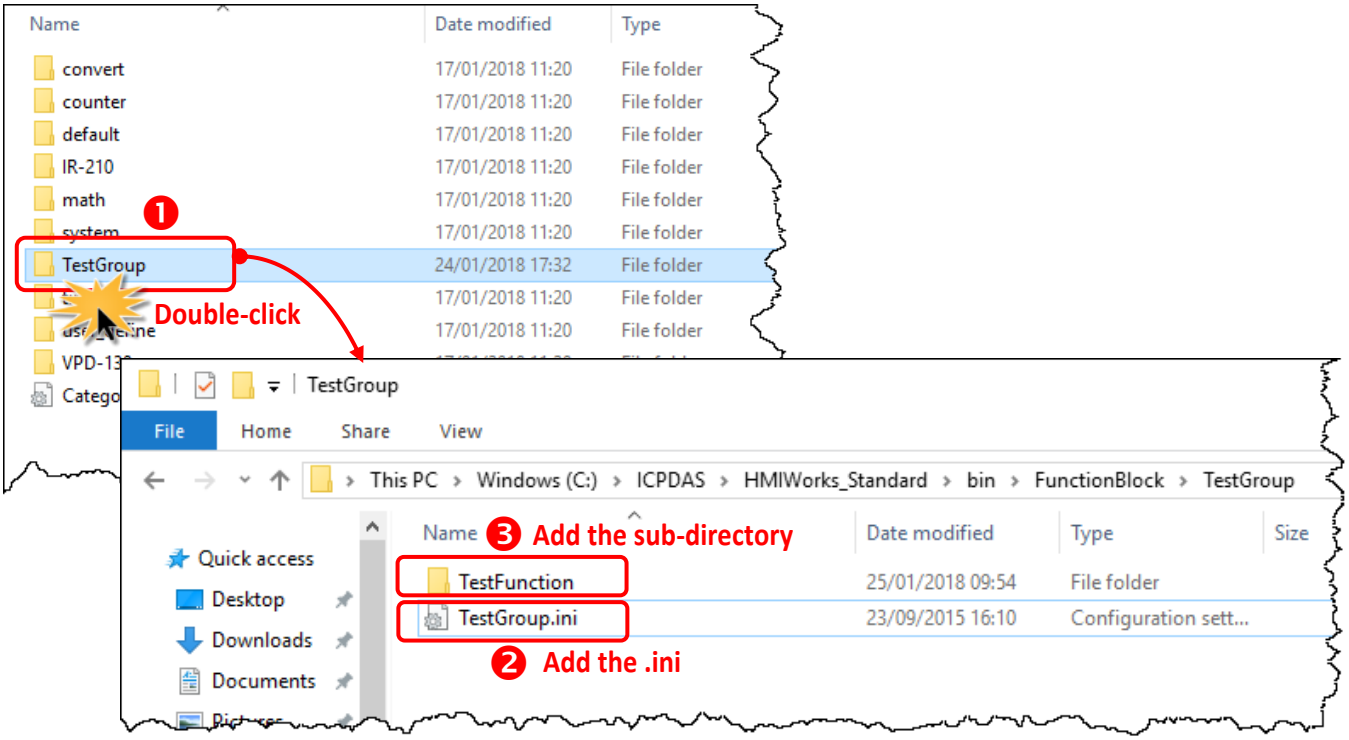
Note: The name of the new item in the "Category.ini" **must** be exactly the same as the name of the newly-created directory.



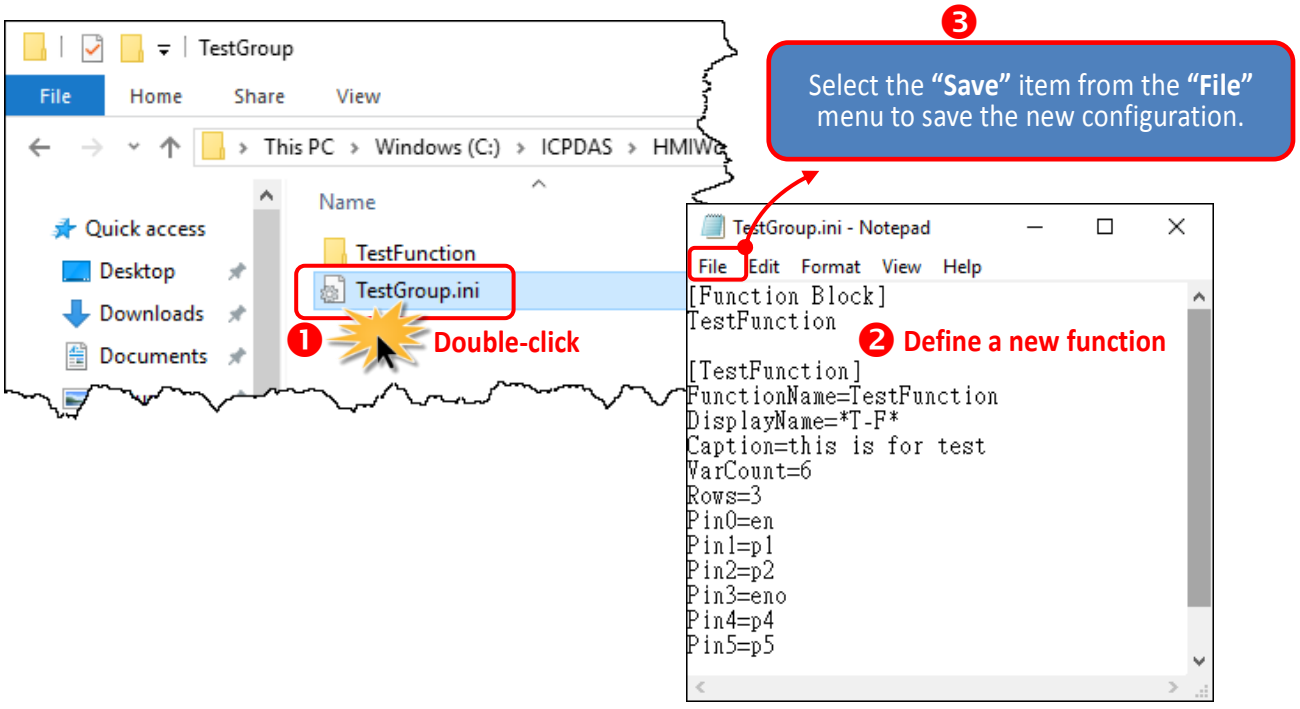
You can open the "Function Block" window to check new group (e.g., TestGroup) has been added.



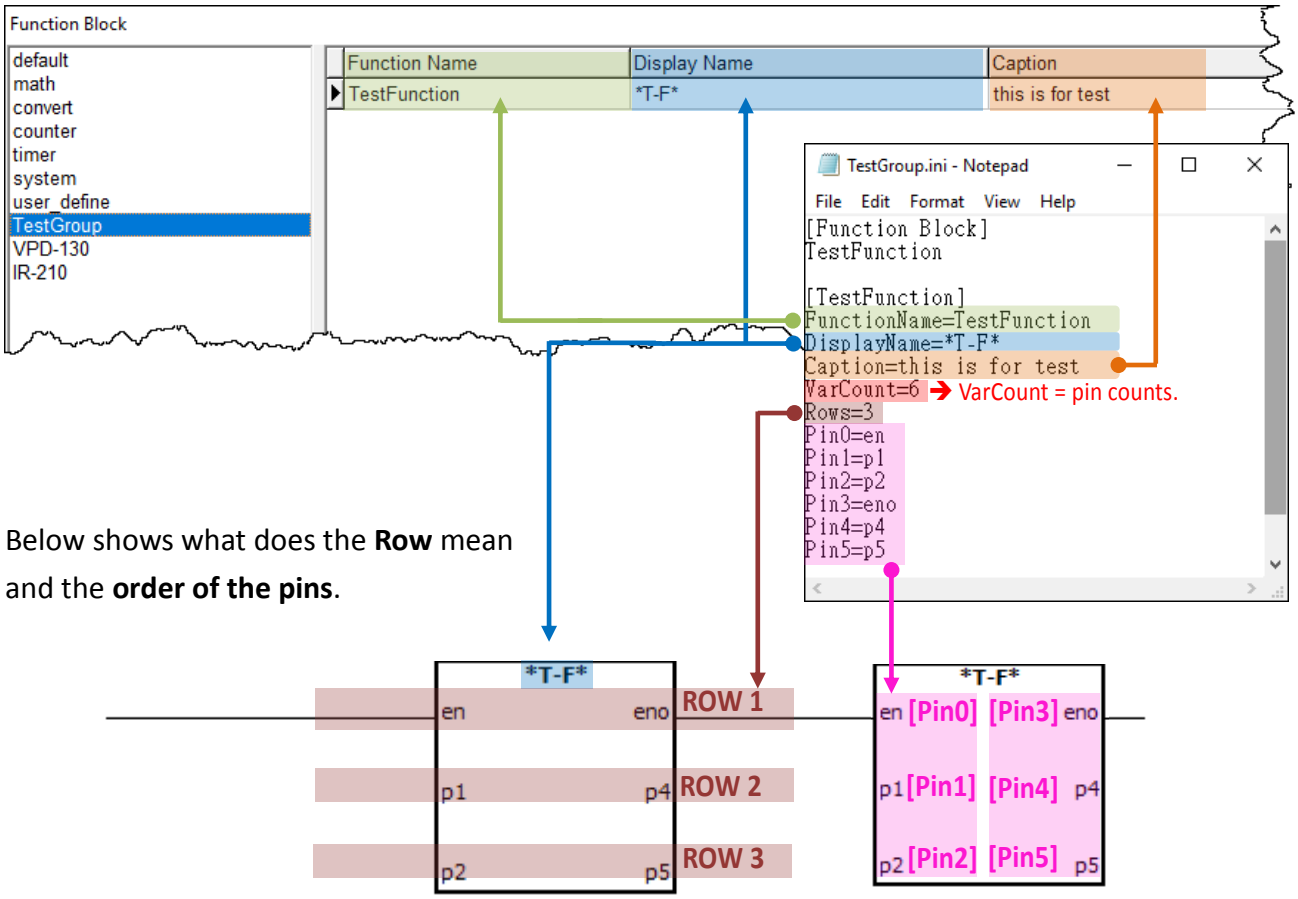
Step 3: Go to the directory **"TestGroup"**, create a **".ini"** file of the exactly same name as that of the group, that is, **"TestGroup"**, and create a sub-directory of the **"TestGroup"** directory and we may call the sub-directory **"TestFunction"**.



Step 4: Finally, define a new function (e.g., **TestFunction**) in the file **"TestGroup.ini"**.

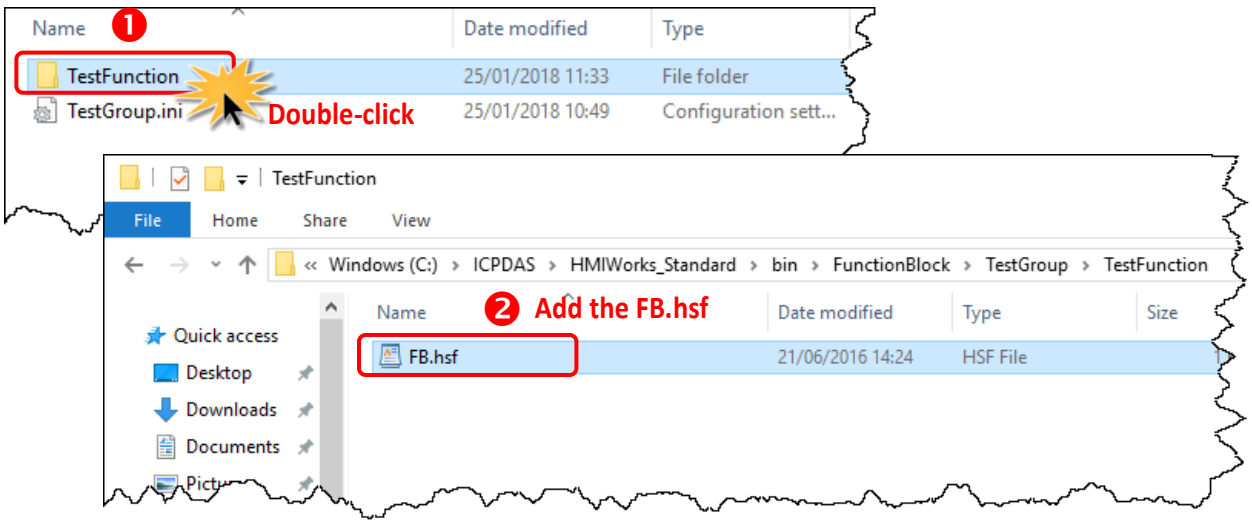


You can open the “Function Block” window to check new function (e.g., **TestFunction**) has been added.



Below shows what does the **Row** mean and the **order of the pins**.

Step 5: In the directory **“TestFunction”**, create a new file “**FB.hsf”** to implement the user-defined function.



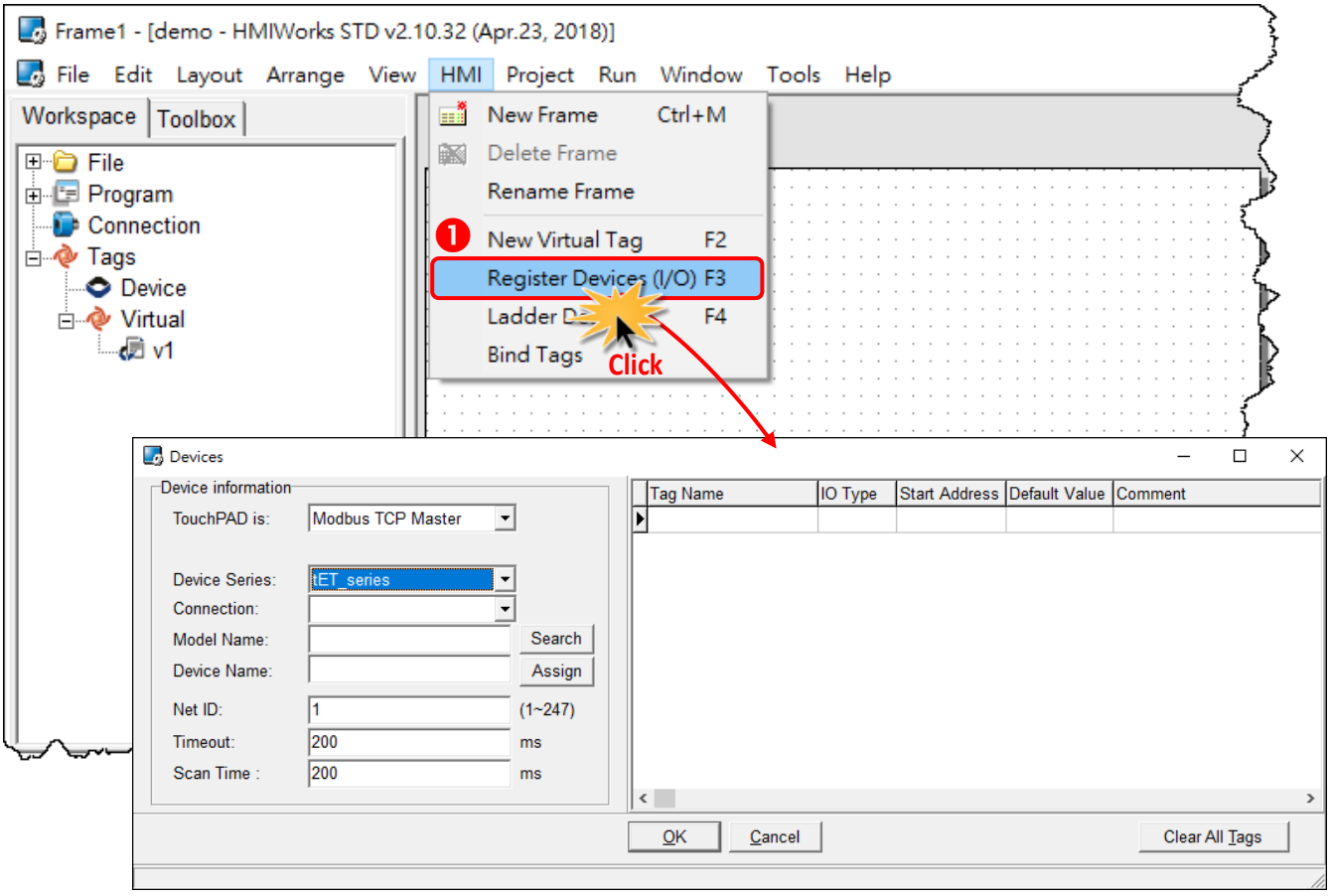
3.3.6 Associate Tags with Tools

In order to use **Ladder Designer** to build HMI of TouchPAD, we should associate tags with tools. There are three methods to associate tools with tags. Every change of the tag in the **Ladder Designer** is updated to the tool in the run time after association.

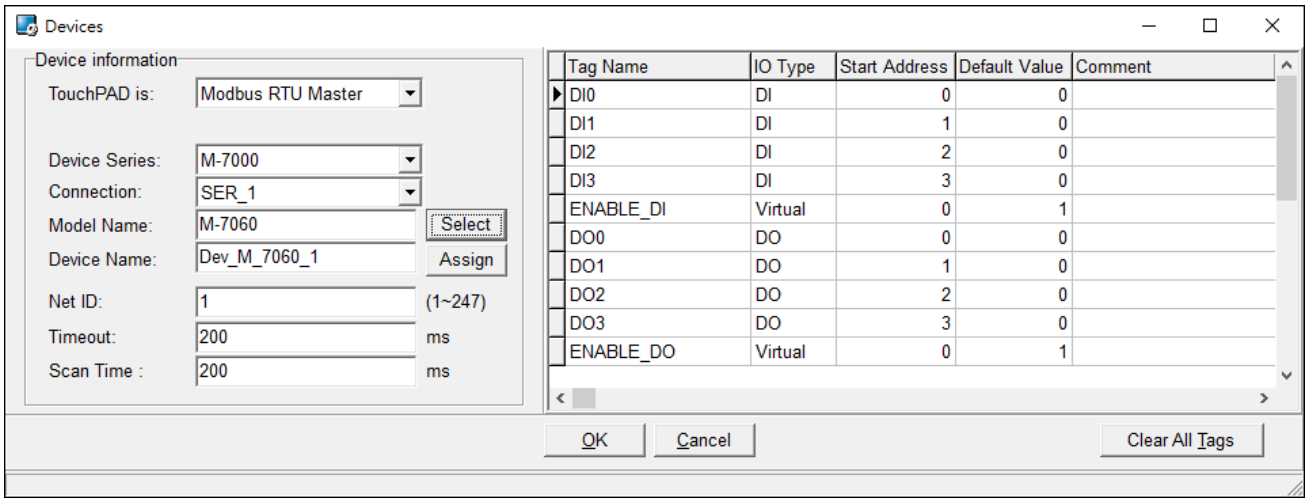
3.3.6.1 Add the New Device Tags (F3)

To associate tools with tags, add **"Device"** tags for the first, as following the procedure described below:

Step 1: Click the **"Register Devices (I/O) (F3)"** from the **"HMI"** menu to open the **"Device"** window. or right click on the **"Device"** item and select the **"New Device"** in the **"Workspace"** panel.



Step 2: In the **"Device"** window, configure the device information and click the **"OK"** button to import tags. Here, the M-7060 module is used as an example.



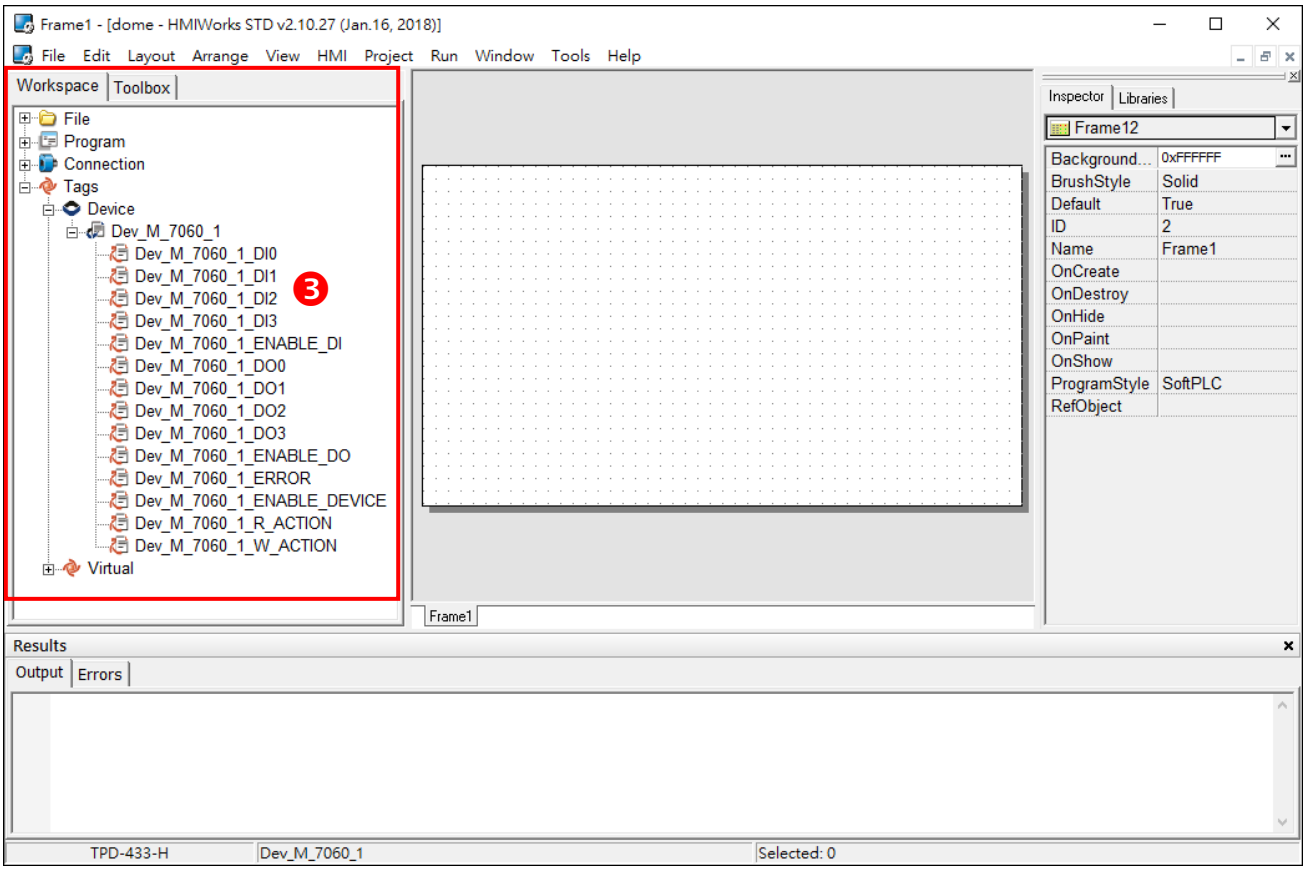
The following is an overview of the functions contained in the **Device Information** section:

Option	Descriptions
TouchPAD is	Specify the TouchPAD acts as master or slave device and protocol, refer to following table 3-1 for more detail.
Device Series	Specify the device type, refer to following table 3-1 for more detail.
Connection	Specify an existing connection approach (TCPIP, XV-board or COM Port) or create a new one to connect to the I/O module. **Note:** when TouchPAD is **Modbus TCP slave**, check the **"As a Server"** option in the "**New/Edit Connection**" window which is called by right-click on the connection item in the **"Workspace"** panel.
Model Name	Specify the model name of the I/O module to connect.
Device Name	Specify the name of the I/O module. Users can assign a name they want.
Net ID	When **TouchPAD acts as a master device**, Net ID is the specified ID of the I/O module in the network. When **TouchPAD acts as a slave device**, Net ID is the specified ID of TouchPAD itself.
Timeout	Set the communication timeout value (default: 200 ms).
Scan Time	Set the update time of device tags (default: 200 ms).

Table 3-1: The following is an overview of the **"TouchPAD is"** and **"Device Series"** options:

TouchPAD is	Device Series	Device Series Descriptions
Modbus RTU Master	**M-7000**	Remote I/O modules over Modbus RTU protocol
	DL_series_MRTUM	Remote temperature and humidity over Modbus RTU protocol
	tM_series	Tiny series remote I/O modules over Modbus RTU protocol
	LC_series	Lighting control module over Modbus RTU protocol
	PM_series	Power meter over Modbus RTU protocol
	IR_series	IR learning remote module over Modbus RTU protocol
	PIR_series	PIR motion sensor and temperature sensor module over Modbus RTU protocol
	XVBoard	VPD series I/O expansion boards
	User_Define(MRTUM)	Remote Modbus RTU I/O modules of third parties
	GateWay(MRTUM)	DALI Gateway over Modbus RTU protocol
	Example(MRTUM)	Other example module (Customize Modules)
Modbus RTU Slave	**Profiles(MRTUS)**	TouchPAD is treated as a Modbus RTU slave device and wait for some master devices to control
Modbus ASCII Master	**User_Define(MASCM)**	Remote Modbus ASCII I/O modules of third parties
Modbus ASCII Slave	**Profiles(MASCS)**	TouchPAD is treated as a Modbus ASCII slave device and wait for some master devices to control
DCON Master	**I-7000**	Remote I/O modules over DCON protocol
	DL_series_DCON	Remote temperature and humidity over DCON protocol
	tM_series_DCON	Tiny series remote I/O modules over DCON protocol
Modbus TCP Master	**tET_series**	Tiny series remote I/O modules over Modbus TCP protocol
	PET-7000	Remote I/O modules over Modbus TCP protocol
	WISE-7000	WISE (Web Inside, Smart Engine) devices
	User_Define(MTCPM)	Remote Modbus TCP I/O modules of third parties
	Example(MTCPM)	Other example module (Customize Modules)
Modbus TCP Slave	**Profiles(MTCPS)**	TouchPAD is treated as a Modbus TCP slave device and wait for some master devices to control

Step 3: The creation of the **"Dev_M_7060_1"** device is now complete and check these imported tags in the **"Workspace"** panel.

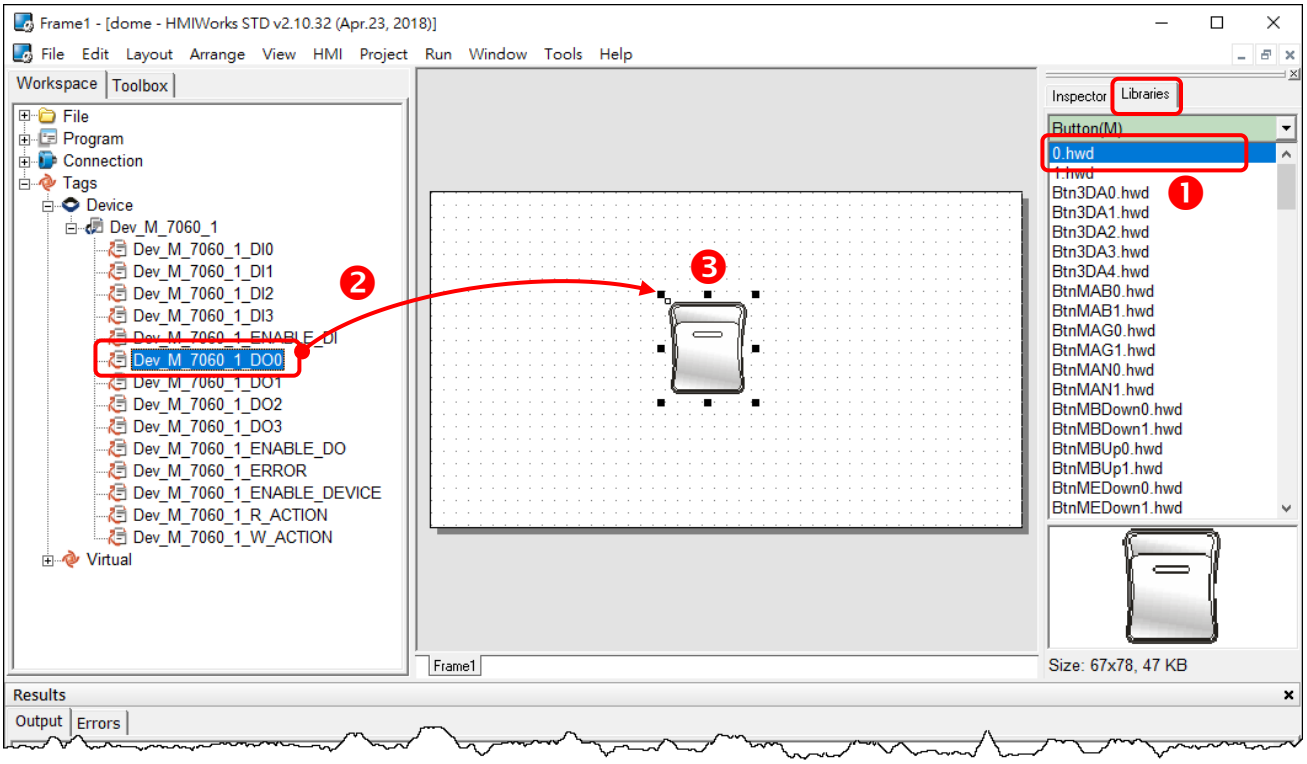


3.3.6.2 Three Methods to Associate Tools with Tags

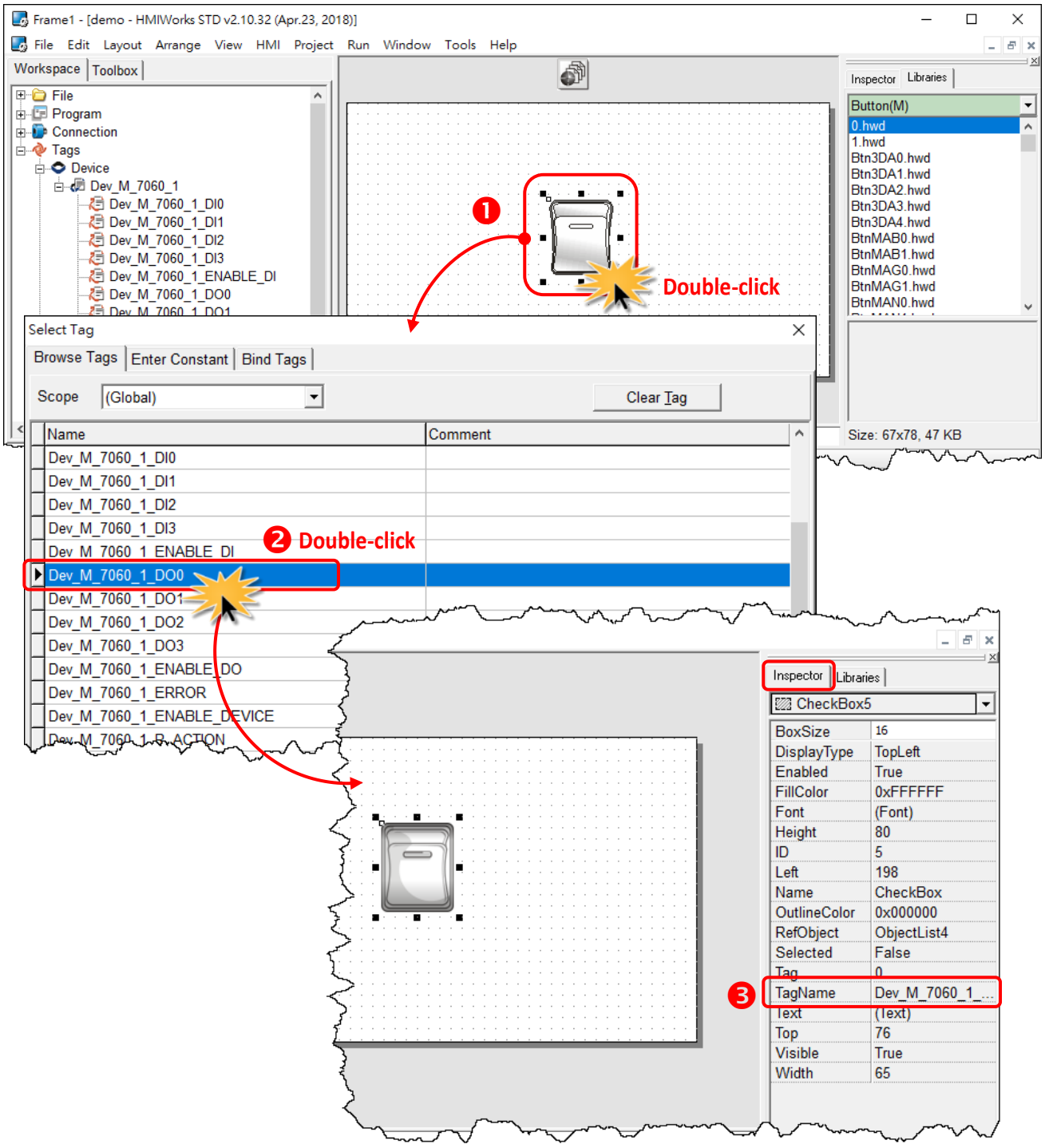
Method 1: Simply drag and drop the tags in the "**Workspace"** panel to the frame design area. A **CheckBox** component is created with the tag associated.

Note: this feature is only supported for the **CheckBox** components.

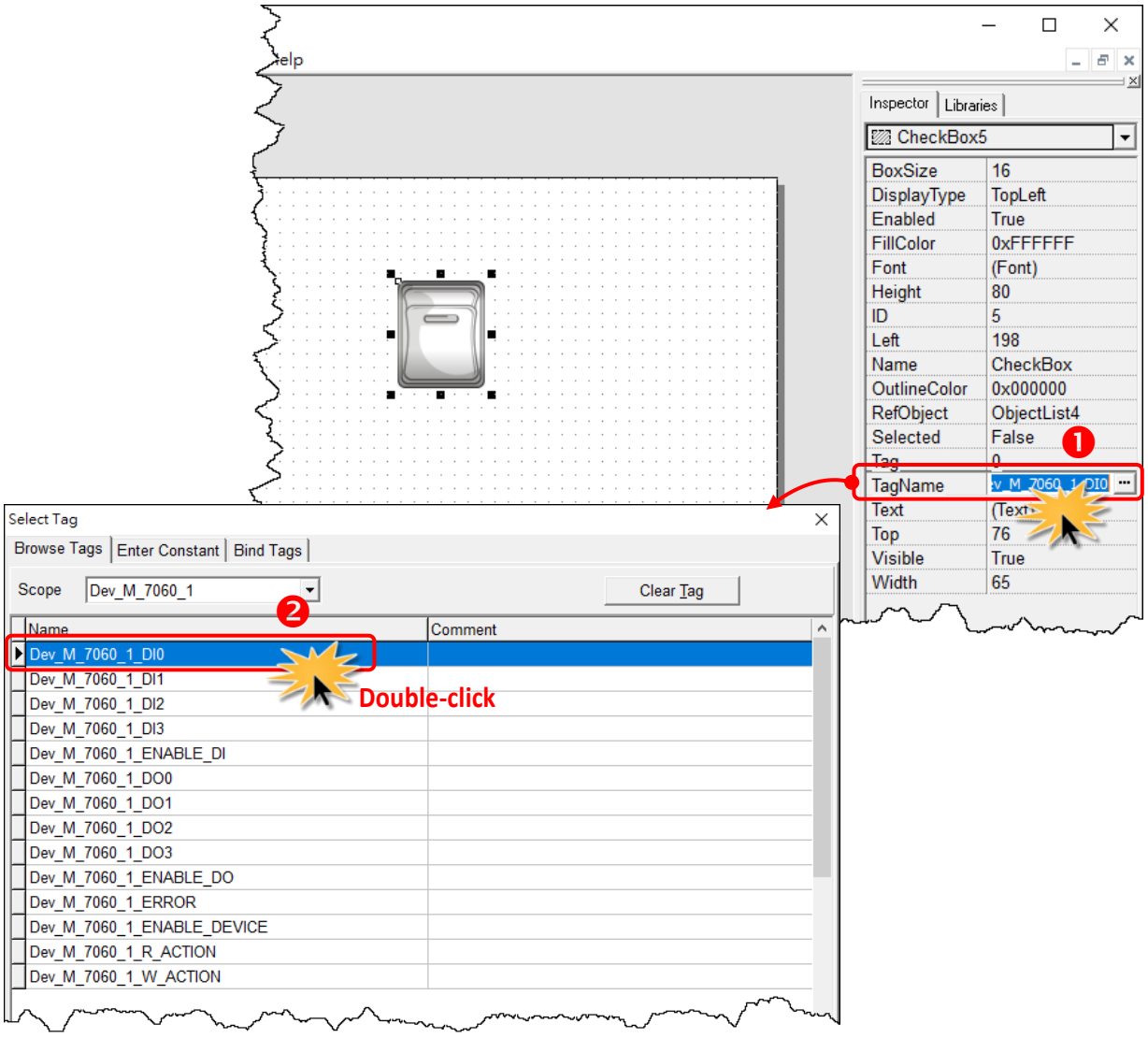
1. Pick an icon to represent the tag in the **"Libraries"** panel.
2. Click a tag.
3. Drag and drop the tag on the frame design space.



Method 2: Double-click the widget (e.g., CheckBox) on the frame design area to open the "**Select Tag**" window, and double-click on the tag Name you want to associate with the widget (e.g., CheckBox). Then you can see the tag is associated with the widget (e.g., CheckBox) by setting the property "**TagName"** in the **"Inspector"** panel to the name of the tag.



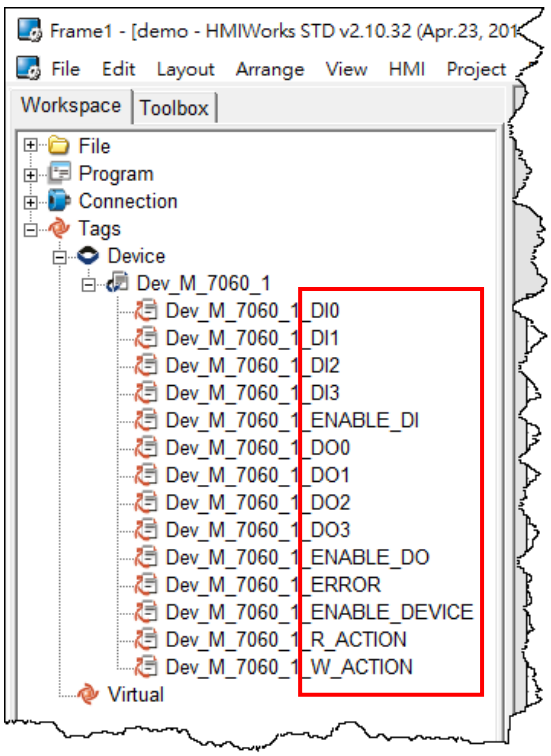
Method 3: Click the **"…"** button from the **"TagName"** field in the **"Inspector"** panel to open the "**Select Tag**" window.



Note: Refer to Section 3.4.17 ObjectList. Set the **RefObject** property of a **CheckBox** component to an **ObjectList** component which contains images and then associate a tag to the **CheckBox** component. Then every time the tag changes its value, the **CheckBox** component toggles the images. This feature is especially useful when building switches.

3.3.6.3 Introduction to Device Tags

The following is a detailed description of the device tags, including the **ENABLE_DO**, **R_ACTION**, **W_ACTION**, **ERROR** and **ENABLE_DEVICE** tags, etc., each of which will be described in more detail below.



Option	Descriptions
DIn	Digital Input Channels
DOn	Digital Output Channels
AIn	Analog Input Channels
AOn	Analog Output Channels
ENABLE_DI	Is used to enable/disable the DI group. **1:** Enable, **0:** Disable
ENABLE_DO	Is used to enable/disable the DO group. **1:** Enable, **0:** Disable
ENABLE_AI	Is used to enable/disable the AI group. **1:** Enable, **0:** Disable
ENABLE_AO	Is used to enable/disable the AO group. **1:** Enable, **0:** Disable
ENABLE_DEVICE	Is used to enable/disable all read write operations on this device. **1:** Enable, **0:** Disable

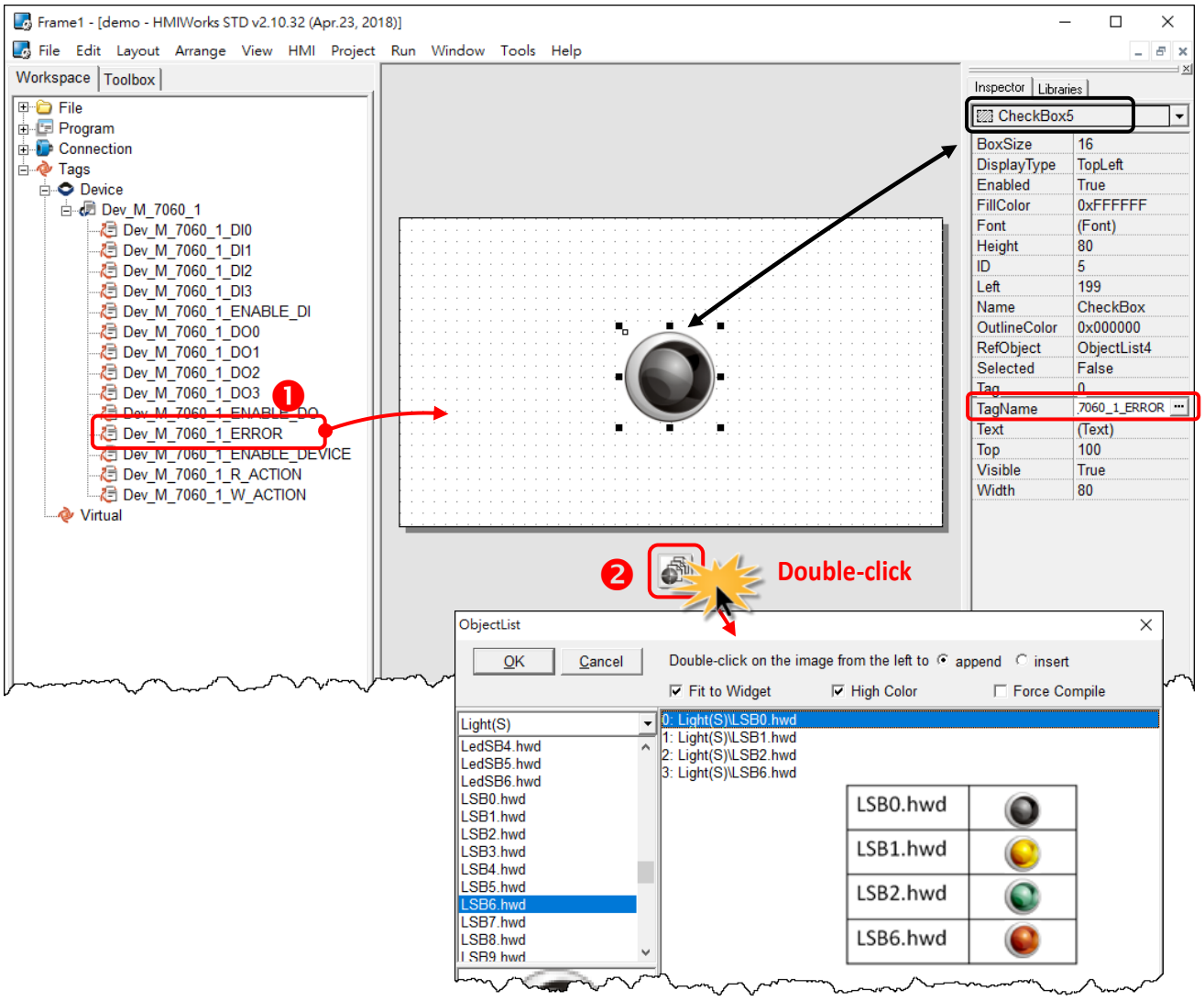
Option	Descriptions
R_ACTION	Is used to enable/disable the Read Action including DO or AO groups. **1:** Enable, **0:** Disable
W_ACTION	Is used to enable/disable the Write Action including DO or AO groups. **1:** Enable, **0:** Disable
ADDR_BASE	Is used to configure the shift of the base-address. There is no standard on Modbus protocol, so users have to refer to the device manual and assign correct address. Some devices use address based on 0, while others use 1.
ERROR	Is used to determine the connection status. **1:** Enable; **0:** Disable. In the next page, we will use the blinking period of “ERROR” tag.

Connecting Blinking Cycle

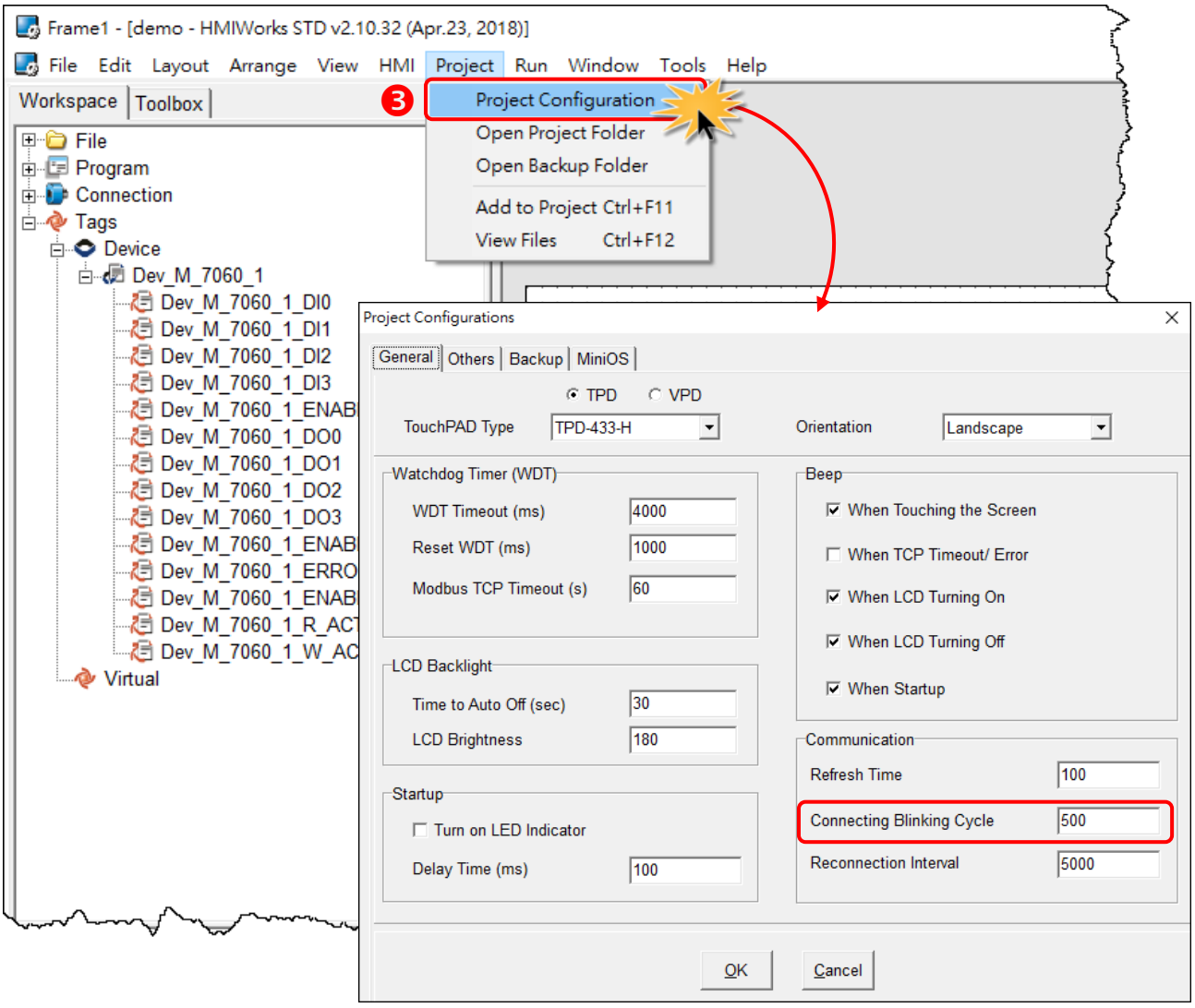
Used for communications of Modbus TCP master polling (remote slave devices) only, the Connecting Blinking Cycle defines the blinking period of **"ERROR"** tag used in devices which can be found in the "**Workspace"** panel.

Follow the procedure described below to demonstrate the usage of Connecting Blinking Cycle.

1. Drag and drop the **"ERROR"** tag on the frame design space.
   A "**CheckBox"** is used to be a signal of communication status of a remote Modbus TCP slave device. (Of course, TouchPAD must be a Modbus TCP master device.)
2. Double-click **"ObjectList"** to open **"ObjectList"** window and assigned the four images.
   To compatible with the old versions of HMIWorks, the first and the second images must represent **"communication normal"** (connected) and **"communication error"** (disconnected). The third and the fourth images toggle when TouchPAD is in its connecting status.



3. Open the **“Project Configuration”** window form the **“Project”** menu to set the **“Connecting Blinking Cycle”**.



3.3.7 User-Defined I/O Modules

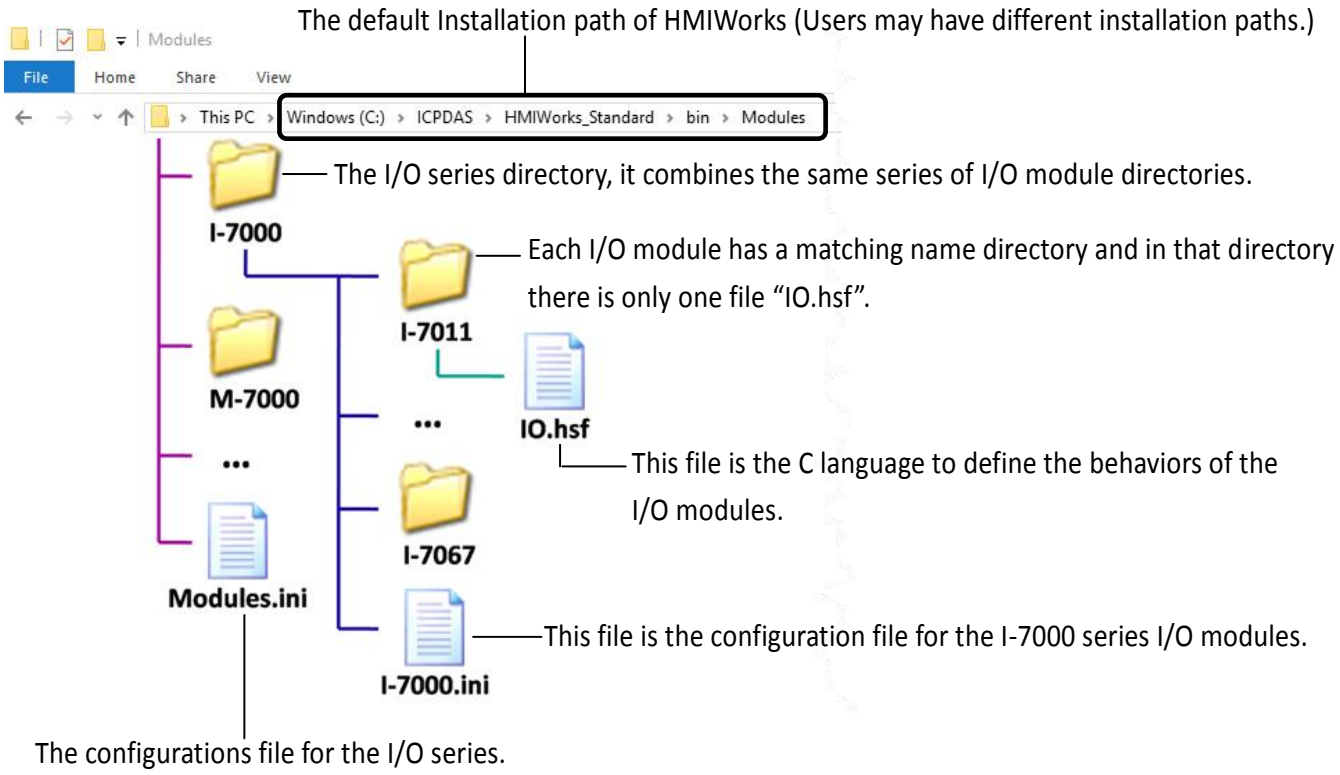
To know how to add a user-defined I/O module, we first explain how HMIWorks uses these I/O modules.

There are several kinds of I/O modules by ICP DAS, as follows:

Model	Description
I-7000 Series	DCON I/O Modules WebSite:https://www.icpdas.com/en/product/guide+Remote_I_O_Module_and_Unit+RS-485_I_O_Modules+I-7000
M-7000 Series	Modbus RTU I/O Modules WebSite:https://www.icpdas.com/en/product/guide+Remote_I_O_Module_and_Unit+RS-485_I_O_Modules+I-7000
ET/PET-7000 Series	Modbus TCP I/O modules WebSite:https://www.icpdas.com/en/product/guide+Remote_I_O_Module_and_Unit+Ethernet_I_O_Modules+ET-7000_ET-7200

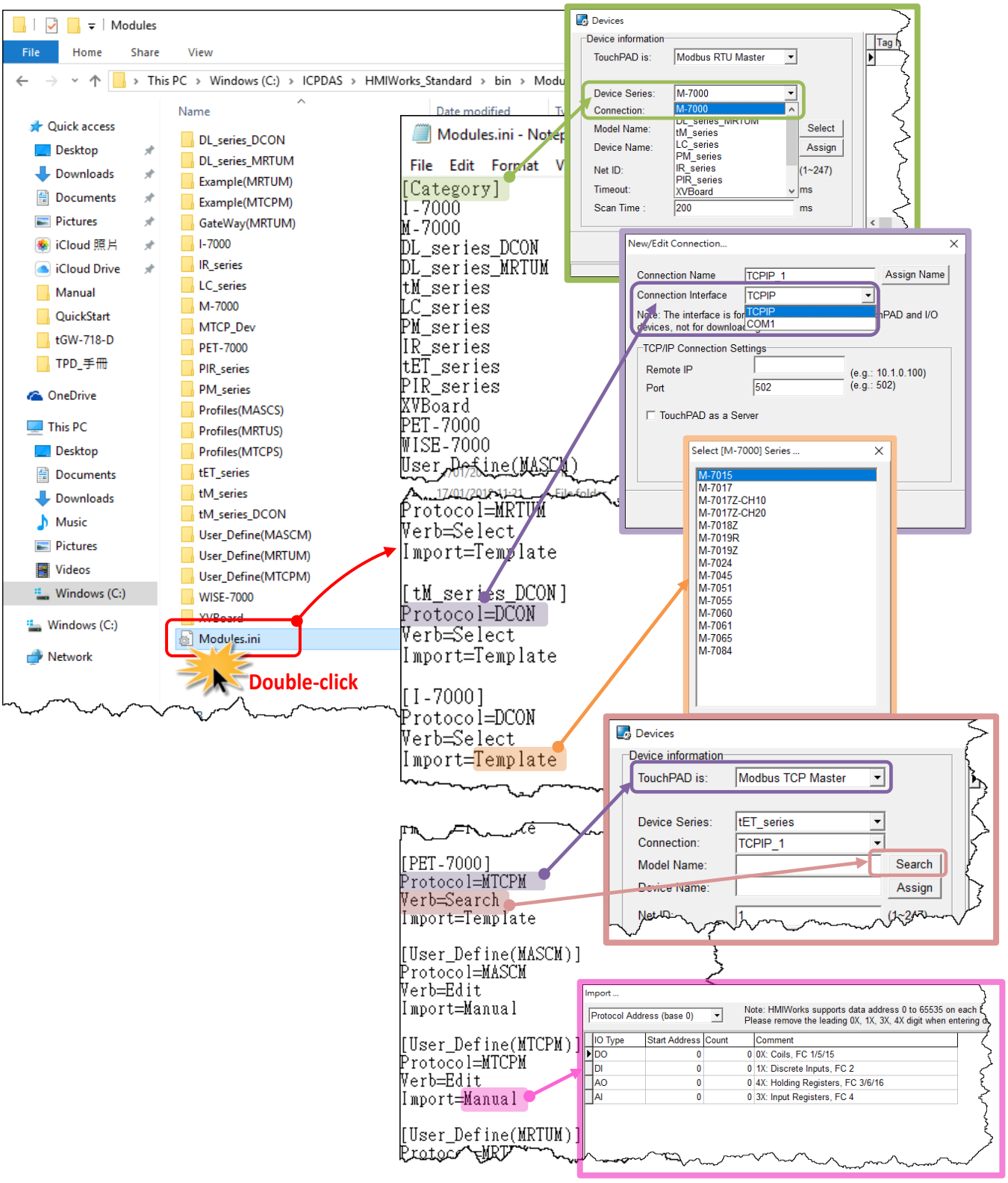
3.3.7.1 Where HMIWorks Put I/O Module Information

HMIWorks puts I/O module information in the following locations.



What "Module.ini Describes

Go to the installation path of the HMIWorks software. In the sub-directory, **"bin\Modules"**, of that installation path, open the file **"Module.ini"** to load the groups.

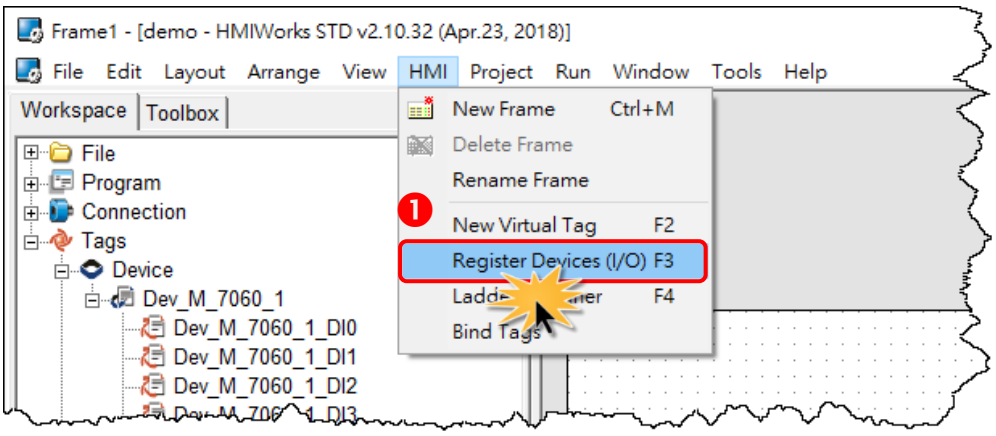


The following is an overview of the functions contained in the **Module.ini** section:

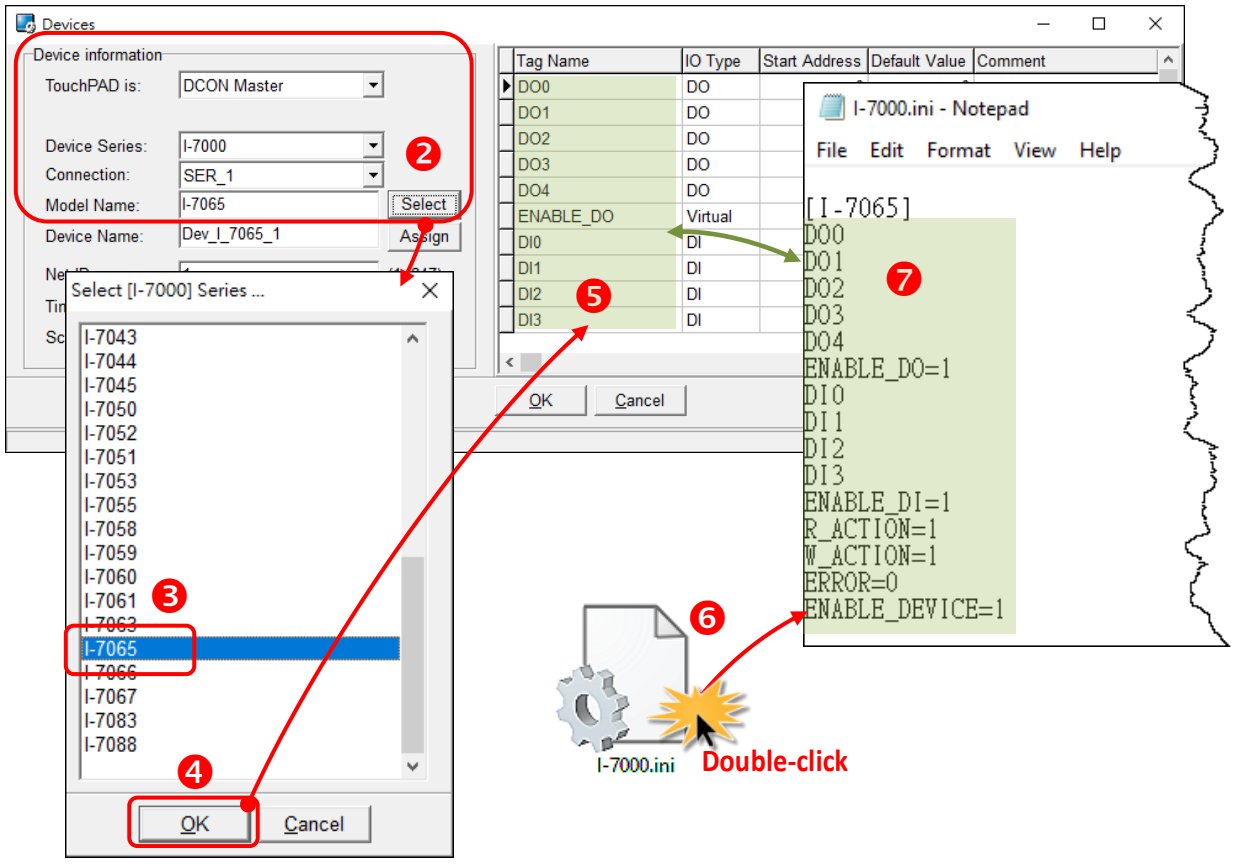
Item		Description
Category		This section keeps the list of the device series which HMIWorks supports. When **registering device (F3)**, the "Devices" window gets the information of device series from this **"Category"** section.
Protocol	**MTCPM**	**"Protocol=MTCPM"** in the **Module.ini** is corresponding to **"TouchPAD is Modbus TCP Master"** in the "Devices" window and **"Protocol=TCPIP"** in the "New/Edit Connection" window.
	MTCPS	**"Protocol=MTCPS"** in the **Module.ini** is corresponding to **"TouchPAD is Modbus TCP Slave"** in the "Devices" window and **"Protocol=TCPIP"** in the "New/Edit Connection" window.
	MASCM	**"Protocol=MASCM"** in the **Module.ini** is corresponding to **"TouchPAD is Modbus ASCII Master"** in the "Devices" window and **"Protocol=COM Port"** in the "New/Edit Connection" window.
	MRTUM	**"Protocol=MRTUM"** in the **Module.ini** is corresponding to **"TouchPAD is Modbus RTU Master"** in the "Devices" window and **"Protocol=COM Port"** in the "New/Edit Connection" window.
	MASCS	**"Protocol=MASCS"** in the **Module.ini** is corresponding to **"TouchPAD is Modbus ASCII Salve"** in the "Devices" window and **"Protocol=COM Port"** in the "New/Edit Connection" window in the **Workspace** panel.
	MRTUS	**"Protocol=MRTUS"** in the **Module.ini** is corresponding to **"TouchPAD is Modbus RTU Salve"** in the "Devices" window and **"Protocol=COM Port"** in the "New/Edit Connection" window.
	DCON	**"Protocol=DCON"** in the **Module.ini** is corresponding to **"TouchPAD is DCON Master"** in the "Devices" window and **"Protocol=COM Port"** in the "New/Edit Connection" window.
Verb	**Search**	HMIWorks scans through the network to find out I/O modules.
	Select	HMIWorks pops up a list of I/O modules to let users select one. The list of I/O modules is loaded from the file whose name is **[Device_Series_Name].ini**
	Edit	HMIWorks opens the "**Import**" window to let users decide the I/O points for the I/O module.
Import	**Template**	HMIWorks imports the tags of the I/O module from the I/O module configuration file. For example, HMIWorks imports tags of I-7011 from the template in the file of **I-7000.ini**.
	Manual	HMIWorks imports the tags of the I/O module by the manually-decided I/O points.

Generating Tags by "Register Devices (F3)"

Click the **"Register Devices (I/O) (F3)"** from the **"HMI"** menu to open the **"Device"** window. (or press **<F3>** key)



The I/O modules configuration file has templates for all the I/O modules in the I/O series. For example, "**I-7000.ini"** is the configuration file for the I-7000 series I/O modules. Take I-7065 in the I-7000 series for example as shown in the following figure.



Defining I/O Behaviors in "IO.hsf"

Take I-7065 for example (I-7000 series I/O module), open the **IO.hsf** in the directory **"[HMIWorks install path]\bin\Modules\I-7000\I-7065\"**. The codes in **IO.hsf** are of C language as below:

```
BEGIN_FUNCTION_BLOCK(); //this line is necessary

DWORD v_do = 0;
DWORD v_di = 0;
int      gWriteCount = 0;

uart_SetTimeout($DEVICE, $TIMEOUT);

//$W_ACTION: a tag used in Ladder to enable/disable writing actions
//$ENABLE_DO: a tag used in Ladder to enable/disable the part of DOs
if ( VAR_VALUE($ENABLE_DO) && VAR_VALUE($W_ACTION))
{
     int iWrite = 0; //To decide if there's a need to write any DO channel
     v_do      = 0;

     // Update the status for each channel if it has been changed.
     iWrite += VAR_GET_WRITE_U32(&v_do, $DO0, 0);
     iWrite += VAR_GET_WRITE_U32(&v_do, $DO1, 1);
     iWrite += VAR_GET_WRITE_U32(&v_do, $DO2, 2);
     iWrite += VAR_GET_WRITE_U32(&v_do, $DO3, 3);
     iWrite += VAR_GET_WRITE_U32(&v_do, $DO4, 4);

     if ( iWrite )    // Write only when need
     {
          gWriteCount++;
          if ( ! dcon_WriteDO($DEVICE, $NETID, 5, v_do & 0xFF) )
           // dcon_WriteDO: the DO writing API function of I-7000 I/O series.
           // I-7000 I/O series uses the DCON protocol.
                return HMI_ERROR;
     }
}

if ( gWriteCount ) return HMI_OK;
// Skip reading to reduce the device loading

if ( (VAR_VALUE($ENABLE_DO) || VAR_VALUE($ENABLE_DI)) && VAR_VALUE($R_ACTION)) {
//$R_ACTION: a tag used in Ladder to enable/disable reading actions
//$ENABLE_DO: a tag used in Ladder to enable/disable the part of DOs
//$ENABLE_DI: a tag used in Ladder to enable/disable the part of Dis
if (dcon_ReadDIO($DEVICE, $NETID, 4, 5, &v_di, &v_do))
```

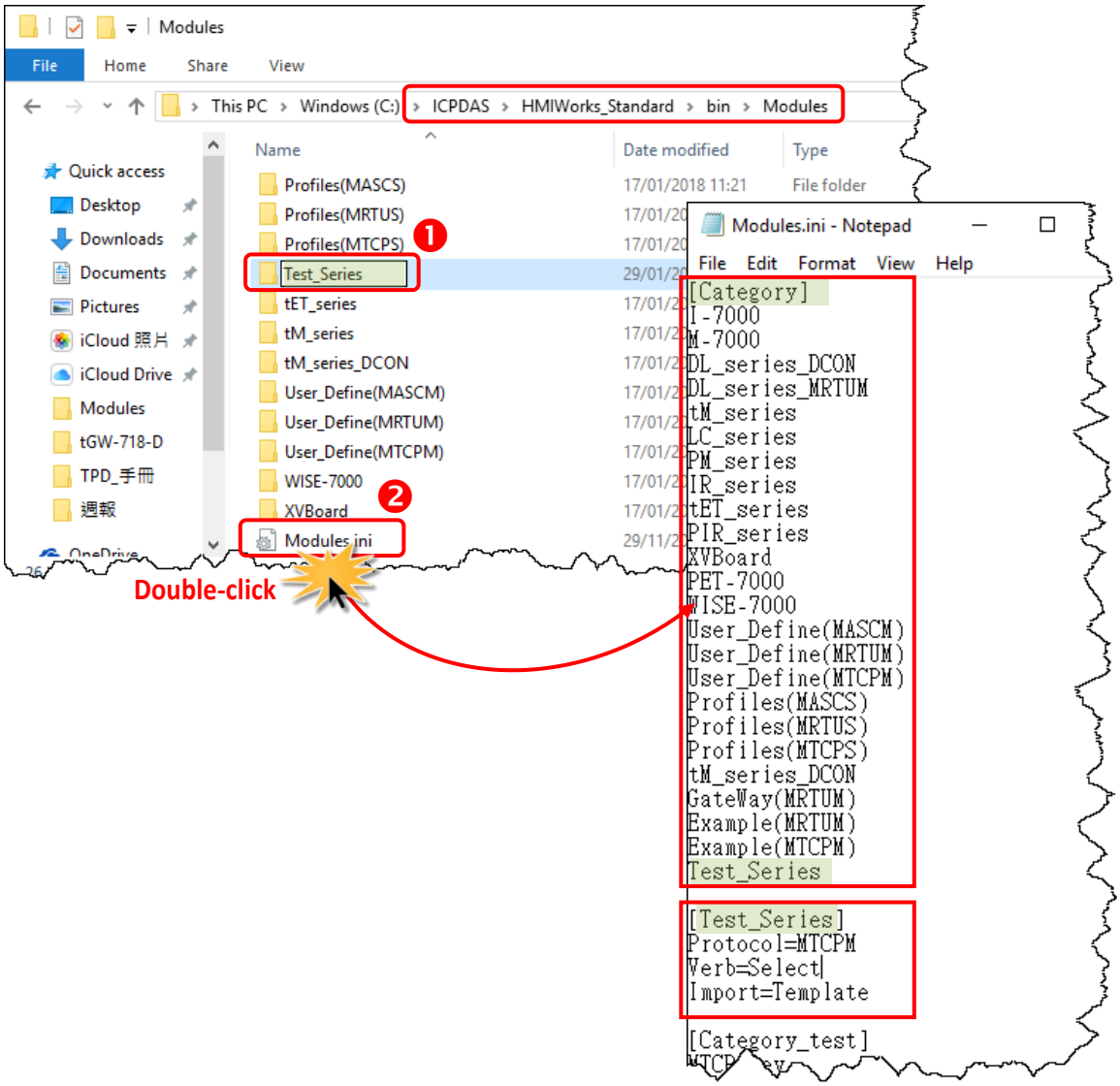
```
// dcon_ReadDIO: the DI/DO reading API function of I-7000 I/O series.
// I-7000 I/O series uses the DCON protocol.
{
        VAR_SET($DI0, v_di & (1<<0));
       // VAR_SET: used to set the value of this channel to its tag
        VAR_SET($DI1, v_di & (1<<1));
        VAR_SET($DI2, v_di & (1<<2));
        VAR_SET($DI3, v_di & (1<<3));

        VAR_SET($DO0, v_do & (1<<0));
        VAR_SET($DO1, v_do & (1<<1));
        VAR_SET($DO2, v_do & (1<<2));
        VAR_SET($DO3, v_do & (1<<3));
        VAR_SET($DO4, v_do & (1<<4));
} else
      return HMI_ERROR;
}

END_FUNCTION_BLOCK(); //this line is necessary
```

3.3.7.2 Creating a User-Defined I/O Module

Now, we introduce how to add a user-defined I/O module.

Step 1: Go to the installation path of HMIWorks. In the directory, **"[HMIWorks install path]\bin\Modules\"**, create a new I/O series directory whose name is **"Test_Series"**.
Step 2: Open the file **"Modules.ini"** to add a new item (e.g., Test_Series) and save the new configuration to notify HMIWorks that there is a new I/O series called **"Test_Series"**.

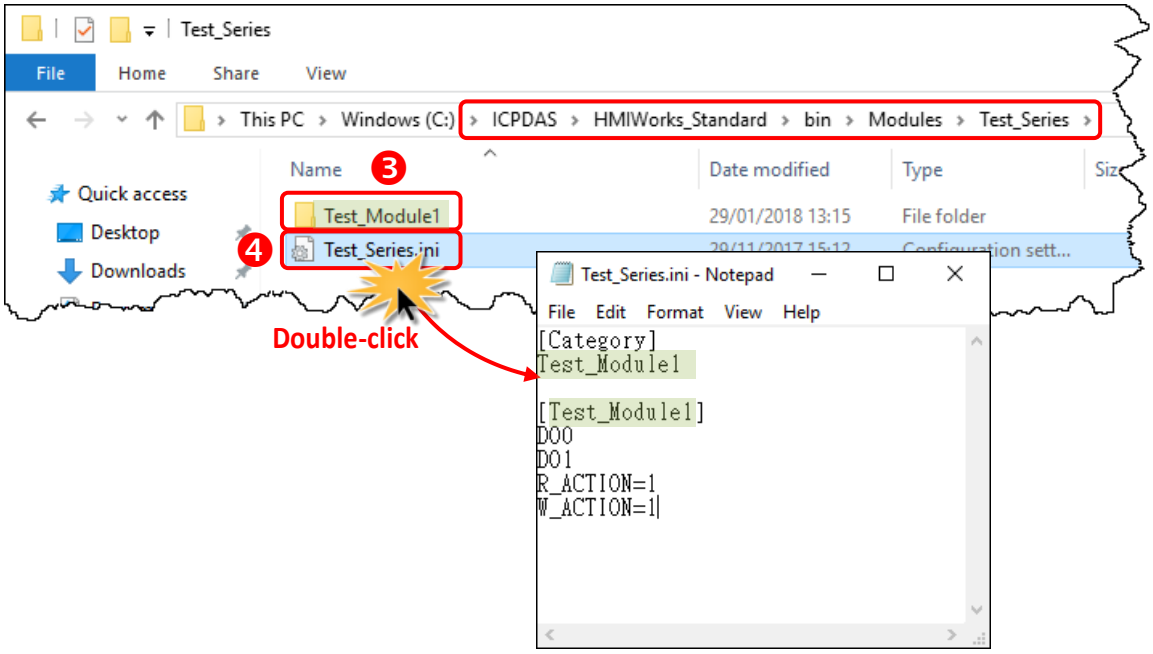
Note: the series directory name and the name in the **Modules.ini** must be **the same**.



Step 3: In the I/O series directory **"Test_Series"**, we create a new I/O module directory whose name is **"Test_Module1"**.

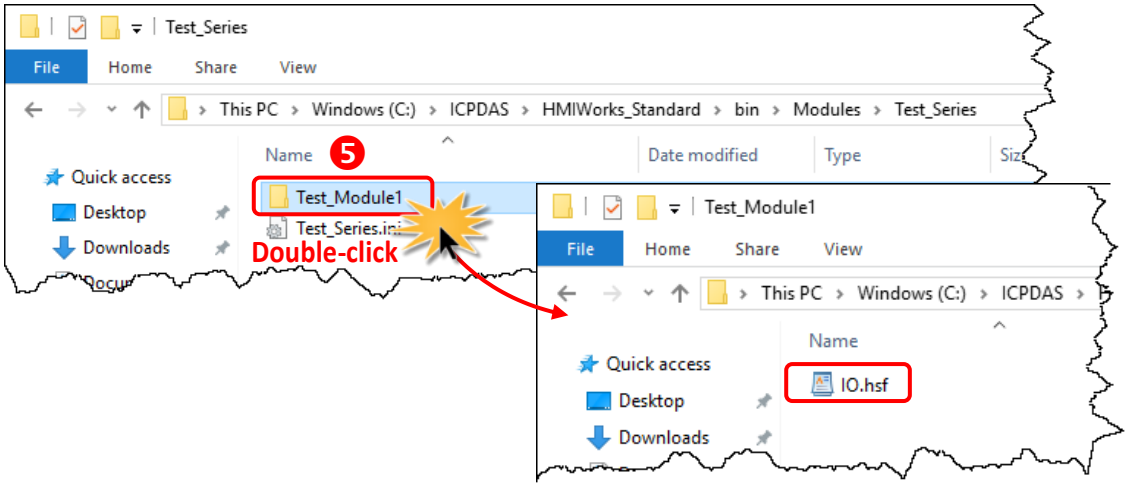
Step 4: Create a I/O modules configuration file **"Test_Series.ini"**, to depict the template of the newly-created I/O module **"Test_Module1"**.

⚠ **Note:** the module directory name and the name in the **Test_Series.ini** must be **the same**.



Step 5: Implement the **IO.hsf** which is created in I/O module directory **"Test_Module1"**, to describe the behaviors of the I/O module **"Test_Module1"**.

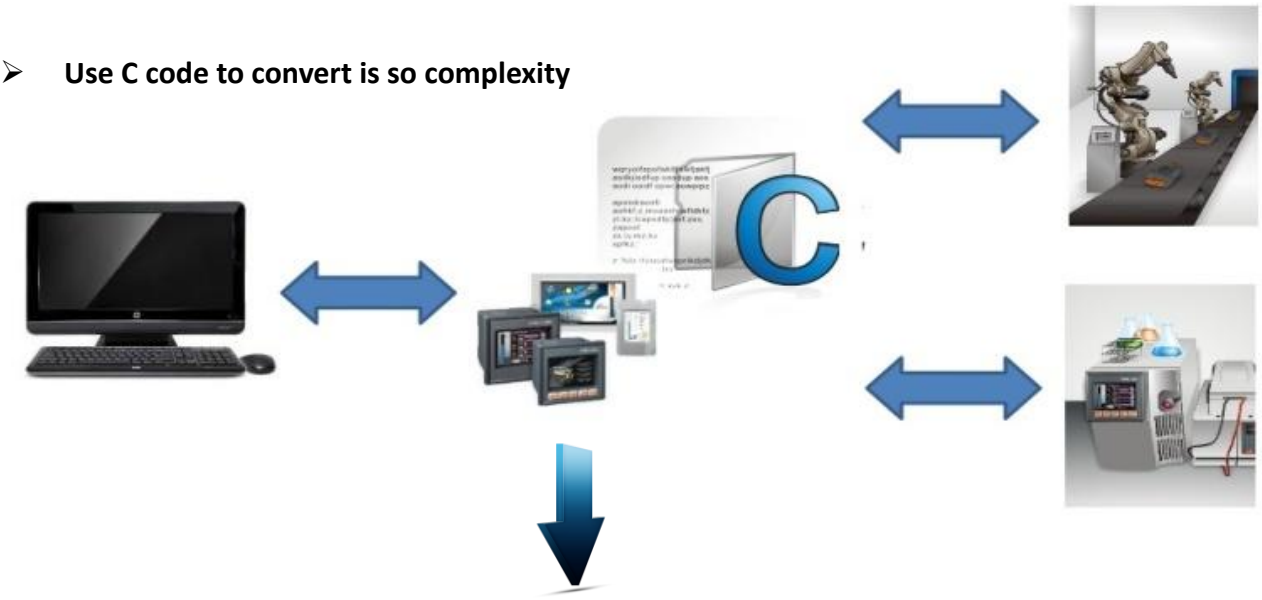
- If using the Modbus TCP protocol, refer to **IO.hsf** of PET-7000 series.
- If using the Modbus RTU protocol, refer to **IO.hsf** of M-7000 series.
- If using the DCON protocol, refer to **IO.hsf** of I-7000 series.
- All are similar to the example of the **IO.hsf** of I-7065 above.



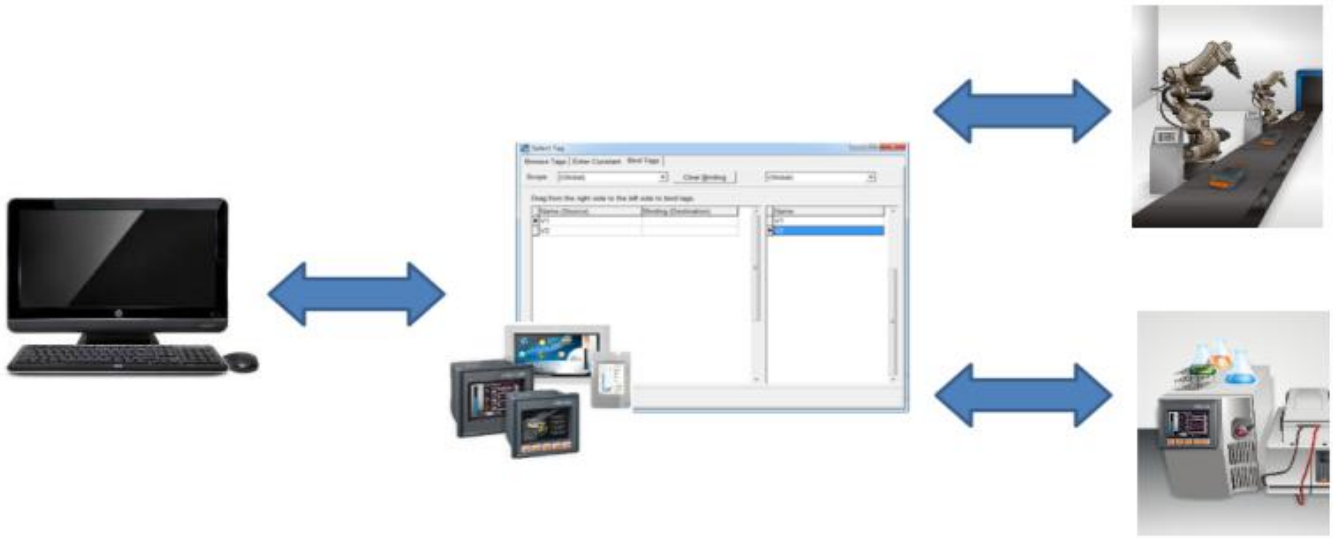
3.3.8 Data Exchange

Uniform standards data format by the TouchPAD and served as the role of protocol conversion to the exchange of information between the different agreements and resolve master and slave exchange of information between the problem of data transfer between the device to automatically **"Agreement"**, "Handle "and" Respond "and let live applications more flexible.

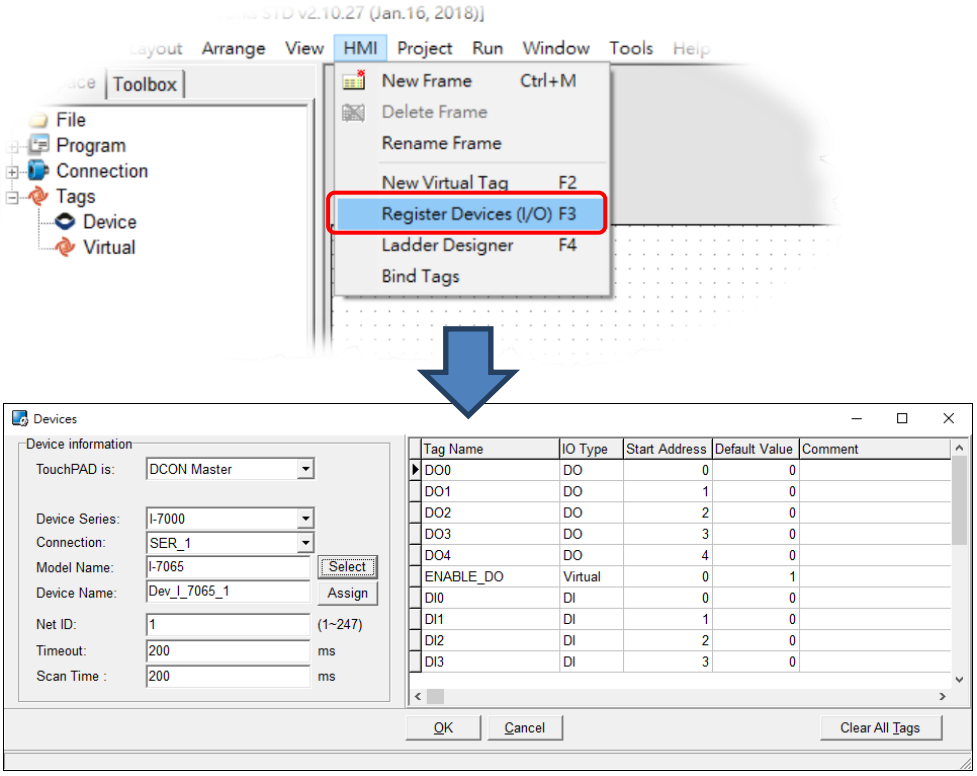
- **Use C code to convert is so complexity**



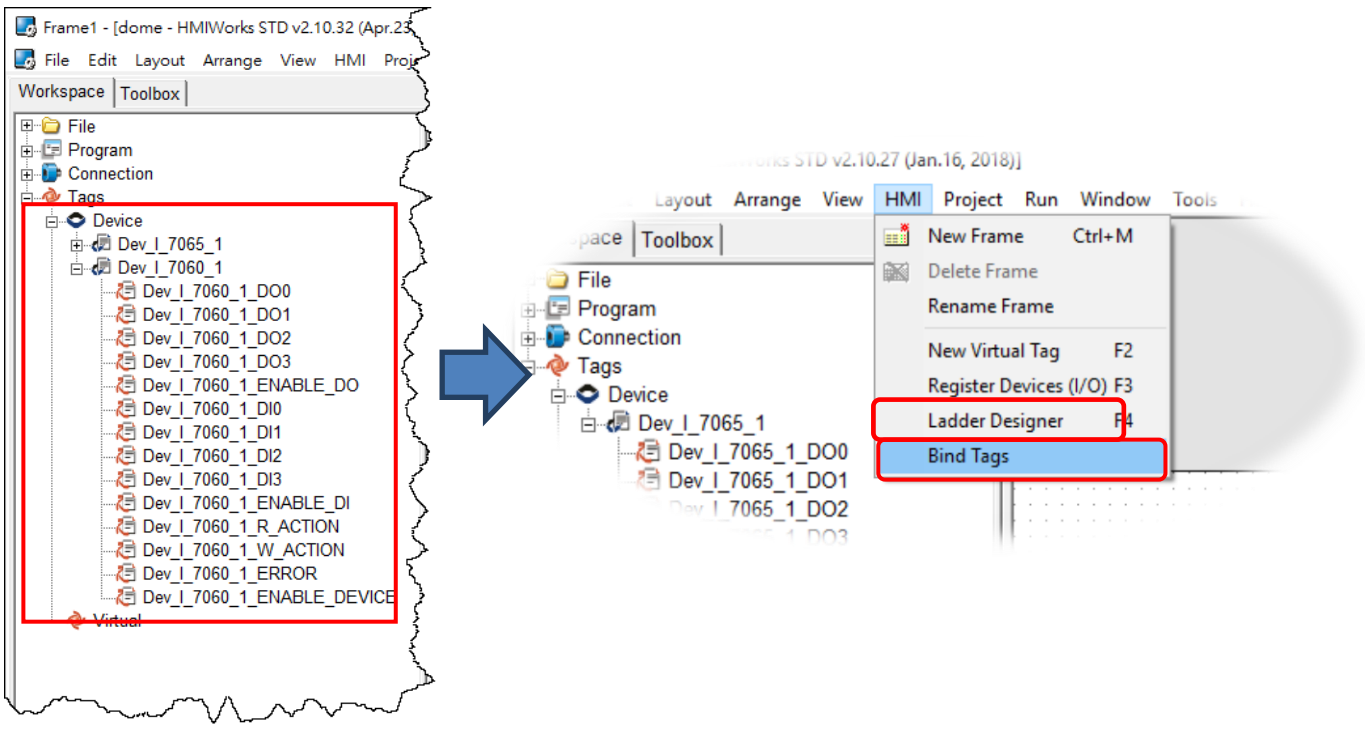
- **And use data exchange function after, it's so easy.**



Step 1: Click the **"Register Devices (I/O) (F3)"** from the **"HMI"** menu to add device in the HMIWorks. (or press **<F3>** key.)



Step 2: Add more devices (e.g., I-7065 and I-7060) after, you can see the **"Workspace"** panel and add tag, then click the **"Bind Tags"** from the **"HMI"** menu.

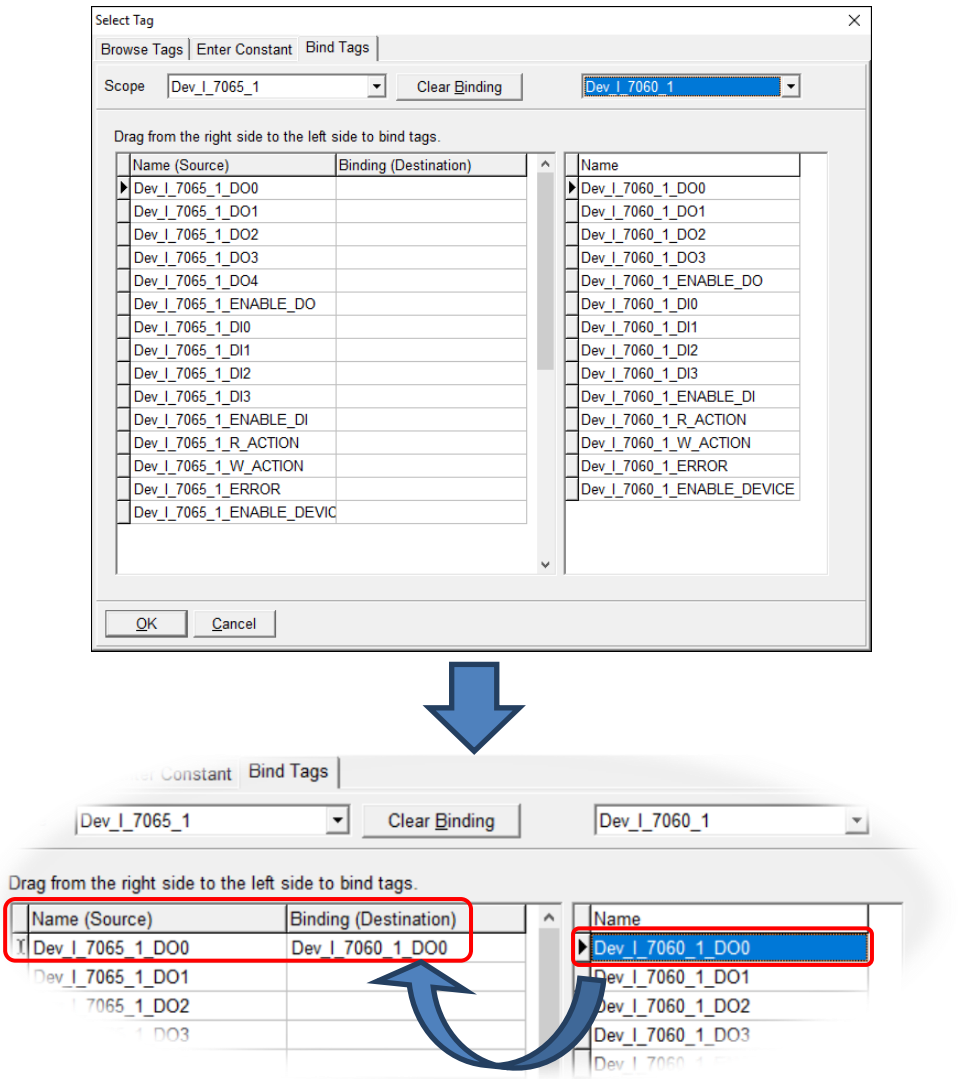


Step 3: Drag the tag that make the relatedness, and the instructions for use as below.

Showed as above pic:

1. Drag the tag from right to left.
2. When the **“Dev_I_7060_1DO0”** drag to **“Dev_ I _7065_1_DO0”** :

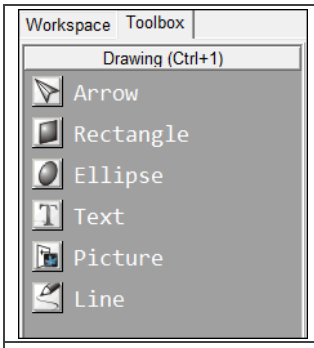
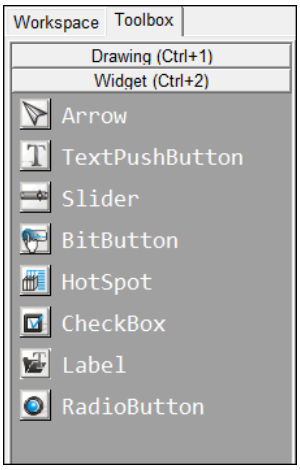
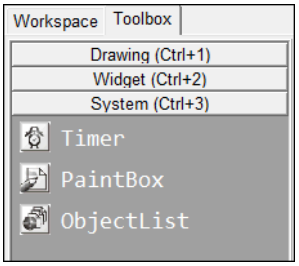
   If **“Dev_ I _7065_1_DO0”** changed, then the **“Dev_ I _7060_1DO0”** value will upade to **“Dev_ I _7065_1_DO0”** value: **Dev_ I _7060_1DO0 = Dev_ I _7065_1_DO0**
3. For example, when B drag to A, C drag toB, if A changed, then B=A, C=A



3.4 Frames and Components

This section introduces properties and usages of frames and components from the **"Toolbox"** panel.

In the **"Toolbox"** panel, there are three kinds of components, the **Drawing**, the **Widget** and the **System** components, each of which will be described in more detail below.

	**Drawing (Ctrl+1):** 1. **Rectangle**: draw a rectangle. 2. **Ellipse**: draw ellipse. 3. **Text**: put string (text) on screen. 4. **Picture**: load an image file on a frame. 5. **Line**: draw a line.
	**Widget (Ctrl+2)**: 1. **TextPushButton**: create a button. 2. **Slider**: show or decide the percentage. 3. **BitButton**: create an image button. 4. **HotSpot**: create a hot spot that can issue an **OnClick** event. 5. **CheckBox**: provide an alternative. 6. **Label**: provide a string that can be modified during the run-time. 7. **RadioButton**: provide a "one-of-many" selection
	**System (Ctrl+3):** 1. **Timer**: periodically execute codes. 2. **PaintBox**: draw shapes in the run time. 3. **ObjectList**: maintain a list of library objects which can be used through property **"RefObject"** of **TextPushButton** and **CheckBox**.

Notes:

1. Make sure that widget component should not overlap or unexpected behavior may happen when clicking.
2. The minimum gap between two components is 12 pixels. If the gap is smaller than 12 pixels, pressing one component may trigger the other's event handler due to calibration accuracy.

3.4.1 Commons of Components and Frames

This section describes the common characteristics of frames and components from the **"Toolbox"** panel.

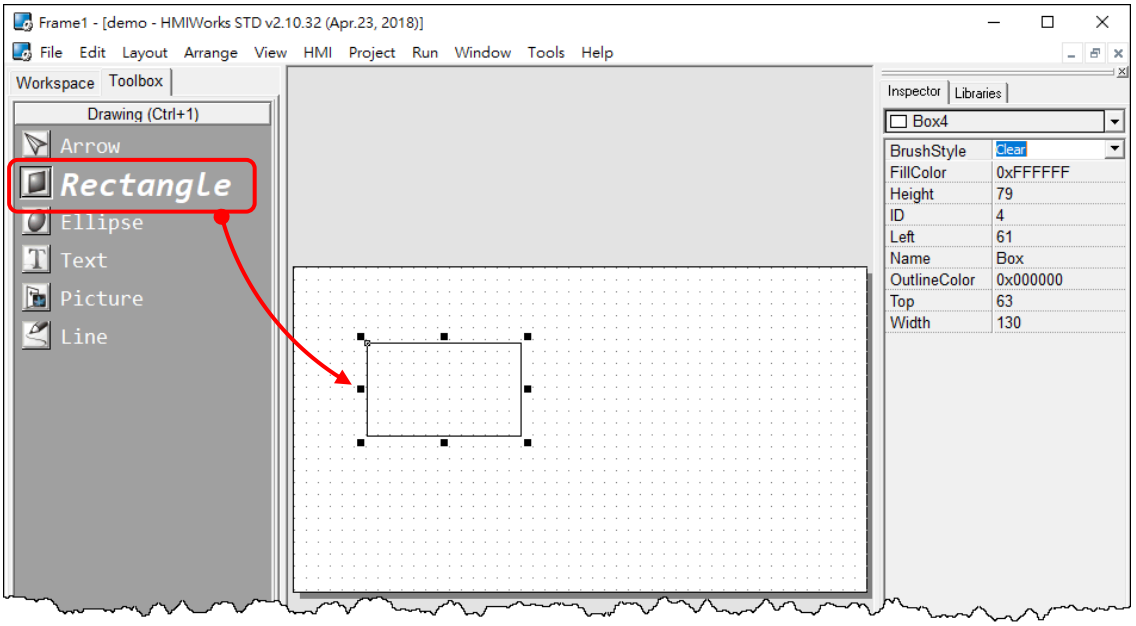
Putting a component on the frame

Two ways to put a component on the frame:

1. Drag a suitable sized rectangle.
2. Simply click on the frame to decide the location after selecting a component.

To drag a suitable sized rectangle, take a Rectangle for example to describe how to put a component (such as a **Rectangle**, a **Slider**, etc.) on the frame.

Step 1: Click the **Rectangle** icon from the **Drawing** panel of the **"Toolbox"** panel.
Step 2: Move your mouse to the frame design area and click and drag a suitable sized rectangle.



To draw a square

What to do if I want to draw a square?

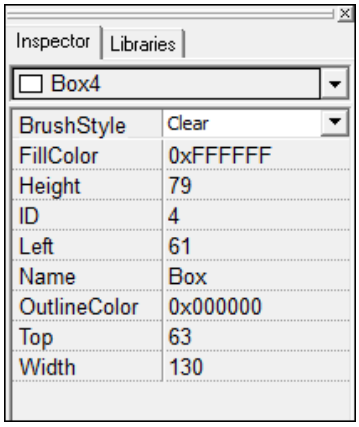
Step 2 with the **<Ctrl>** key pressed at the same time when drawing a Rectangle.

To draw a circle

What to do if I want to draw a circle?

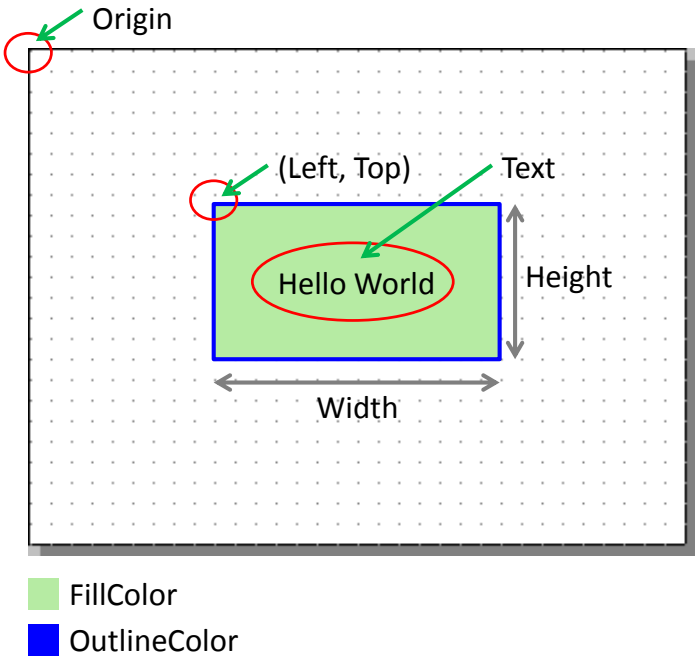
Step 2 with the **<Ctrl>** key pressed at the same time when drawing an Ellipse.

Common Properties



Where can we access properties of a component?

Click on the component (or the frame) and then the properties of the component can be accessed in the **"Inspector"** panel.



The following is an overview of the functions contained in the **Inspector** section:

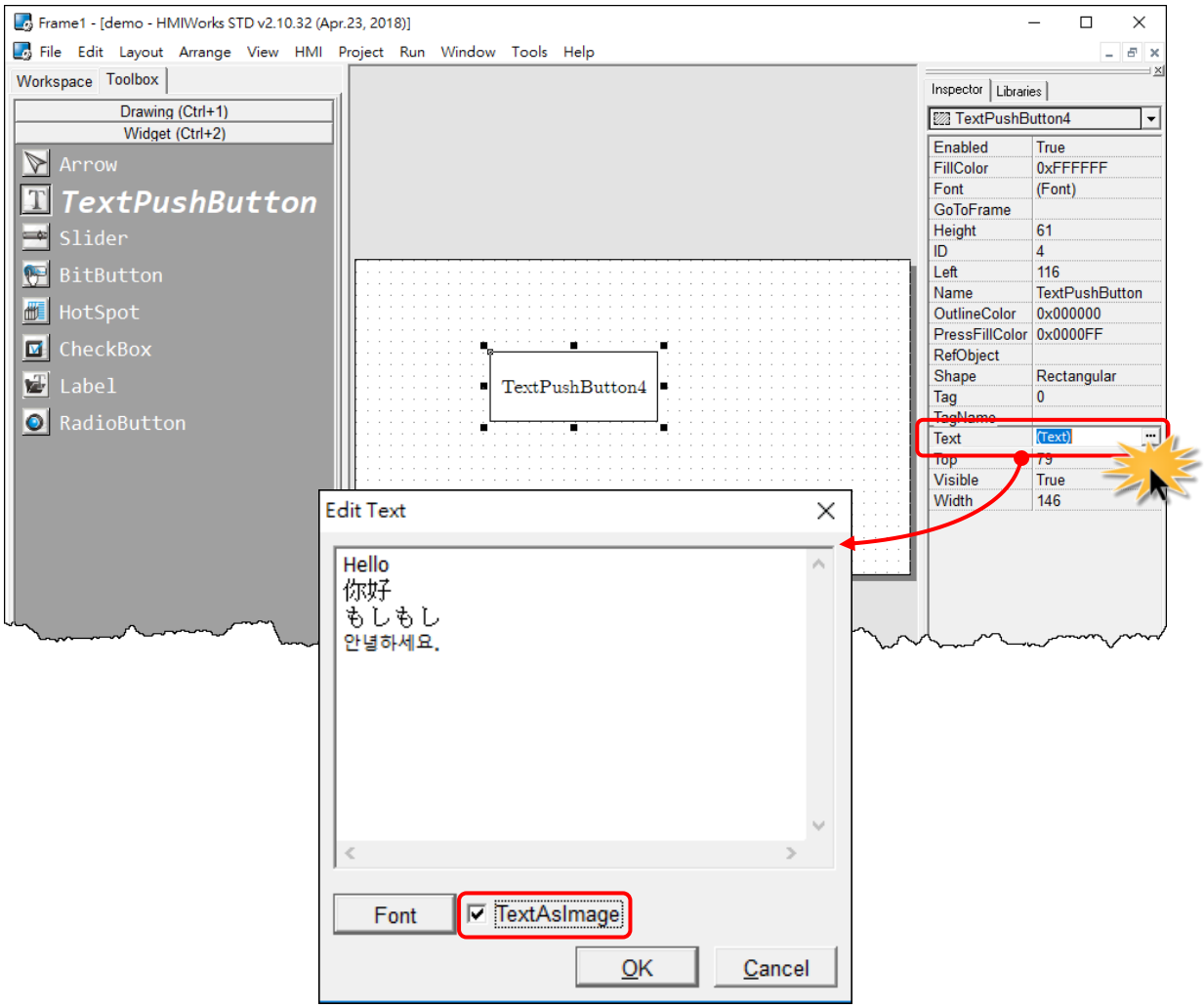
Property	Description
FillColor	The color used to fill the rectangle which encloses the component. The color is represented by a three byte value in the hexadecimal form. From the highest byte to the lowest, it is the blue, the green, and the red byte in sequence.
OutlineColor	The outline color of the rectangle which encloses the component
Height	The length of the vertical side of the rectangle which encloses the component
Width	The length of the horizontal side of the rectangle which encloses the component
Left	The x-coordinate of the left-top vertex of the rectangle which encloses the component
Top	The y-coordinate of the left-top vertex of the rectangle which encloses the component

Property	Description
Name	The name of the component
ID	The serial numbers of the components in the "Toolbox" panel and of the frames. These serial numbers are used to identify them.
Font	The font of the Text property
Text	The strings of the component to be displayed
GoToFrame	Go to the specified frame. That is, pressing on the owner of this property switches to the frame which is specified in this property. **Note:** the **priority** of the property "GoToFrame" is higher than that of "OnClick".
RefObject	The reference to the specified object list. An **ObjectList** is a component that can be selected in the "Toolbox" panel to maintain a list of the images of the library. Refer to Section 3.4.17 ObjectList section for more information.
Tag	The variable used for programming purpose. For example, it can be assigned a unique number for each **TextPushButton** component in order to identify them. Refer to the <<HMIWorks API Reference>> for functions to get/set the Tag property. **Note:** This Tag property has nothing to do with the **"Tag"** which the **TagName** property refers to in the Ladder mode.
TagName	Associate a variable (tag) in Ladder Designer. Note: The property is supported only in programming type **"Ladder"**.
Enabled	Whether the component is activated or not
Visible	Whether the component is able to be seen or not

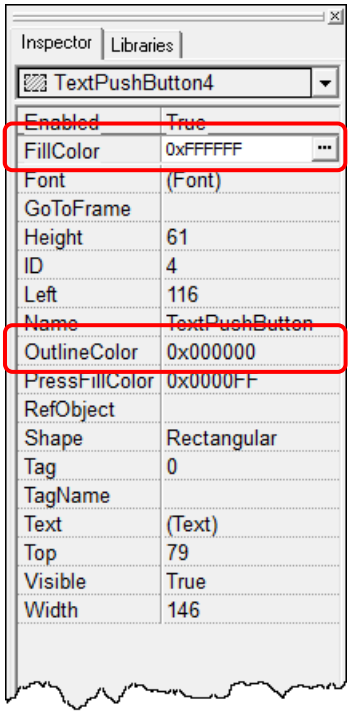
Text into Image and Multi-language Display

There are **three components (TextPushButton**, **CheckBox** and **RadioButton)** whose "**Text"** properties are not like other components and can be used to support multi-language display by transforming strings into images.

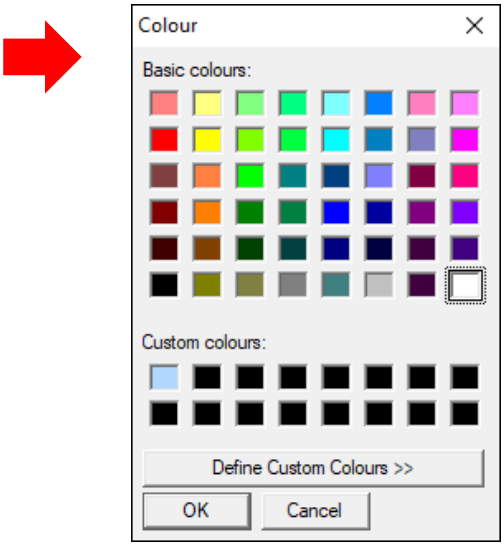
1. Check the "**TextAsImage"** option. If checked, the **Text** property can have multiple strings.
2. Each string of the **Text** property is generated into one image and each image corresponds to one state of the components. Refer to "Using the RefObject property" below for more information.



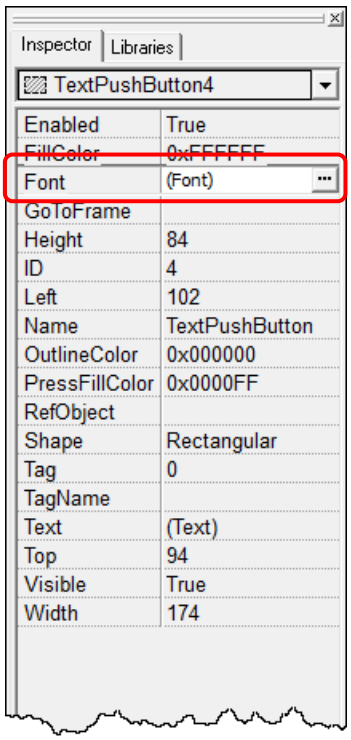
Changing the Color



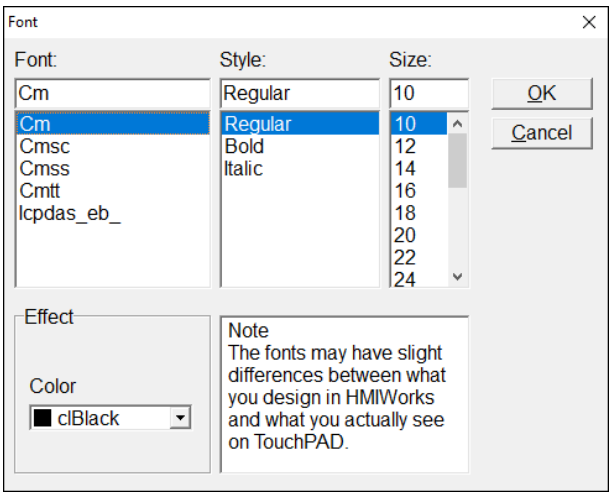
To change the **FillColor** property of a component, click the component first to select it and then click the **"FillColor"** field in the "**Inspector"** panel. Then click on the **"…"** button to open the color dialog to select a color. Repeat the same procedure for the "**OutlineColor"** field.



Changing the Font



To change the **Font** property of a component, click the component first to select it and then click the **"Font"** field in the "**Inspector"** panel. Then click on the **"…"** button to open the font dialog to change the font.



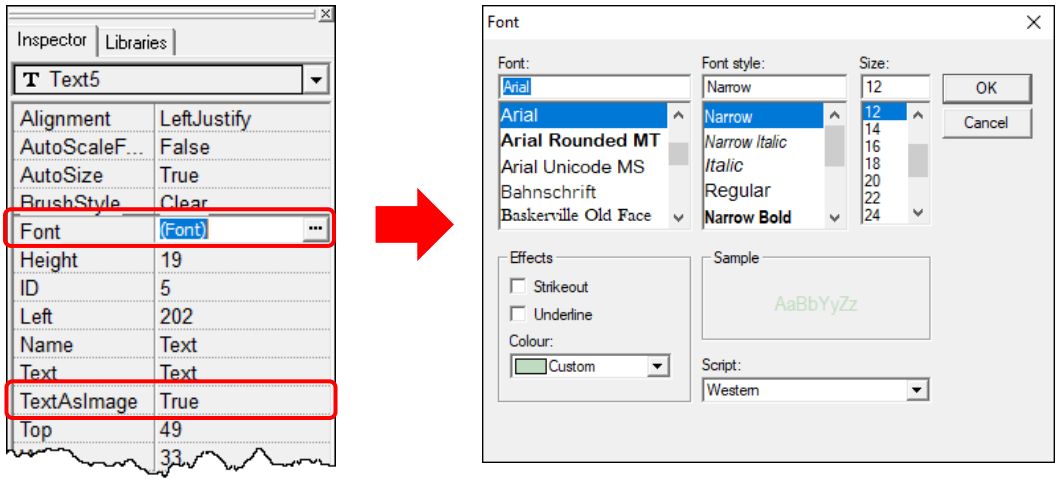
There are two font dialogs when choosing fonts.

- **The same font dialog as the PC's.**
  1. If this font dialog is opened, fonts are stored as image in TouchPAD after download and therefore cost more memory space. (e.g. the same two letters, such as 'A' and 'A' cost)
  2. Widgets that use this font dialog: **Text**, **BitButton**.

- **The custom font dialog that shows only fonts supported by TouchPAD.**
  1. The fonts does not stored as image after download. That is, the same two letters, such as 'A' and 'A', only are stored with the space one 'A' takes.
  2. Widgets that use this font dialog: **Text**, **TextPushButton**, **Slider**, **Checkbox**, **Label**, **RadioButton**.
  3. To support language other than English, refer to
     FAQ: How to display multilingual test on TouchPAD by using the HMIWorks built-in fonts?
     FAQ: How to show multilingual text with ebFont on TouchPAD?

Note:

To use the font dialog of PC's, the **"TextAsImage"** property of a **"Text"** component needs to be set to **"True"**.



Using GoToFrame to switch to another frame

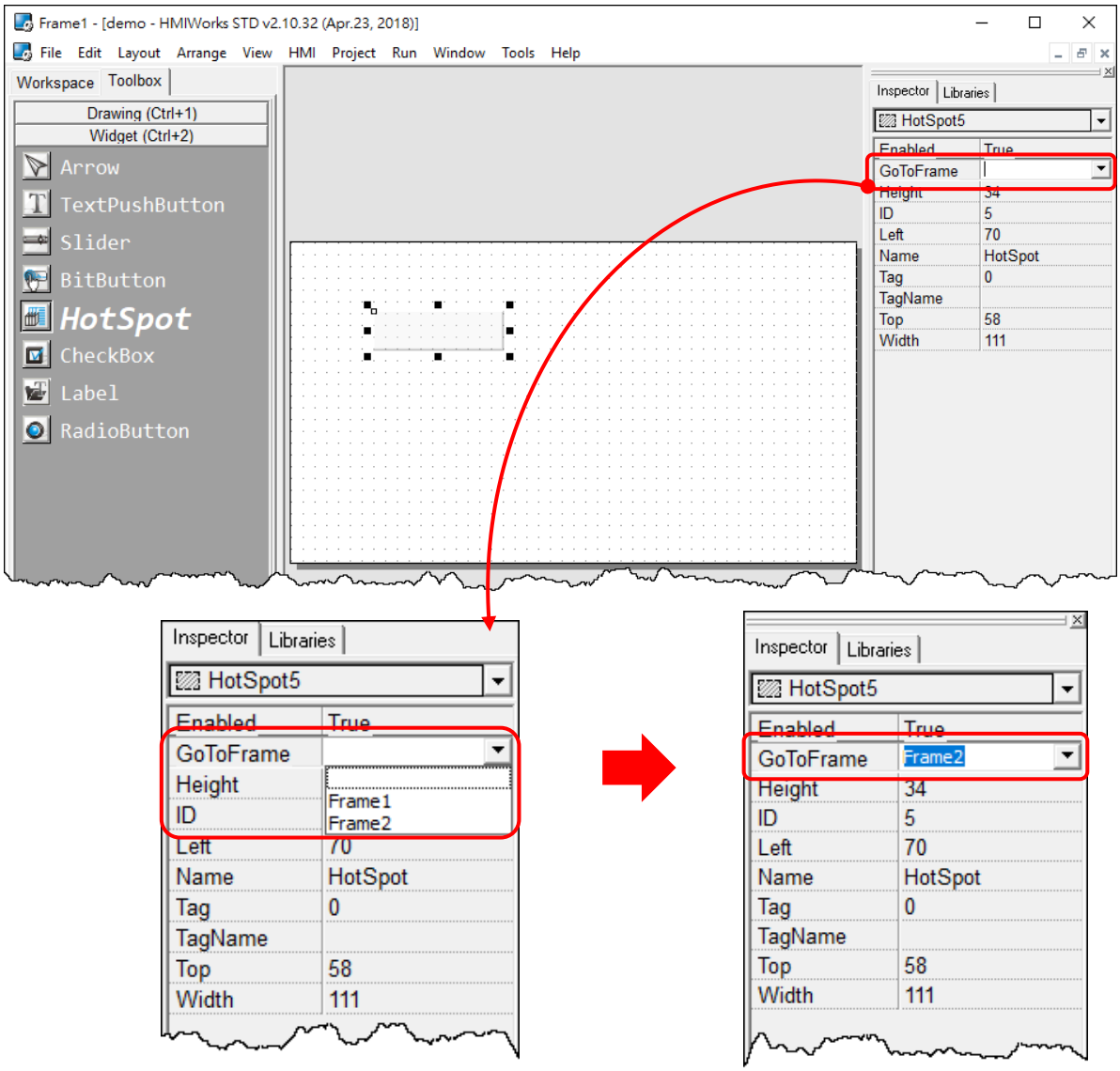
The **GoToFrame** property is used as an event of go-to-specified-frame.

It has higher priority than other events, such as **OnClick** event. Thus specifying an option of the **GoToFrame** property disables the **OnClick** event.

How to add a new Frame?

1. Press **<Ctrl> + <M>**
2. Click the **"New Frame"** from the **"HMI"** menu.

It's easy to specify a value to the **GoToFrame** property. Simply click the **"GoToFrame"** field in the "**Inspector"** panel and then choose the frame to go.



Using the RefObject property

We use the **RefObject** properties to replace the display of **TextPushButton**, **Slider**, **CheckBox**, and **RadioButton** with images of the assigned **ObjectList**. The state (or value) of a component is used as an index to determine which image in the **ObjectList** is displayed if the **ObjectList** is assigned to the **RefObject** property. The state can be changed by human touch, API functions (e.g., **CheckBoxValueSet**), and tags which are specified by the **TagName** property.
Supposed that an **ObjectList** called OL is assigned to the **RefObject** of a component (e.g., **CheckBox**) and it has n images, OL[0], OL[1], …, OL[n-1].

Component/Frame	Max No. of Image	Component Behavior
Frame	1	OL[0] is the background image. **Note1**: Any more images in the **ObjectList**, OL, has no effect, they simply waste memory. **Note2**: Assigning the **RefObject** property of the default frame (the frame which has its default property equal to True) automatically assign the same **ObjectList** to all the frames in the project.
TextPushButton	Unlimited, theoretically	OL[0] is the background image. When the **TextPushButton** is in the released state, it displays the OL[0]. And when it is in the pressed state, it displays OL[1] for the first click, OL[2] for the second click after releasing the first click, and so on. While the **TextPushButton** reaches the last image, OL[n-1], it will start to display from the beginning again for the next click, that is, OL[1], and go on the next round.
Slider	Unlimited, theoretically	OL[0] is the background image. The Slider is divided into n-1 segments and draws the corresponding image according to the value of the Slider. See the table below for example.
CheckBox	Unlimited, theoretically	Every click on the **CheckBox** changes the display image, started from OL[0] to OL[n-1], one by one. Once reaching the last image, OL[n-1], it restart to display from the first image for the next click, OL[0], again.
Label	1	OL[0] is the background image. **Note:** any more images in the ObjectList, OL, has no effect, they simply waste memory.
RadioButton	2	OL[0] is the background image. OL[1] is the selected image. **Note:** any more images in the ObjectList, OL, has no effect, they simply waste memory.

➢ **Slider** example for the **RefObject** property

Example	Description
	6 Images in the **ObjectList**, OL. From left to right, they are OL[0], OL[1], …, OL[5].
	OL[0] is taken as a background image. The **Slider** is divided into 5 segment, 20% for each one, and is drawn by its value: 0% ~ 20%: OL[1] 20% ~ 40%: OL[2] 40% ~ 60%: OL[3] 60% ~ 80%: OL[4] 80% ~ 100%: OL[5] As shown in the left column.

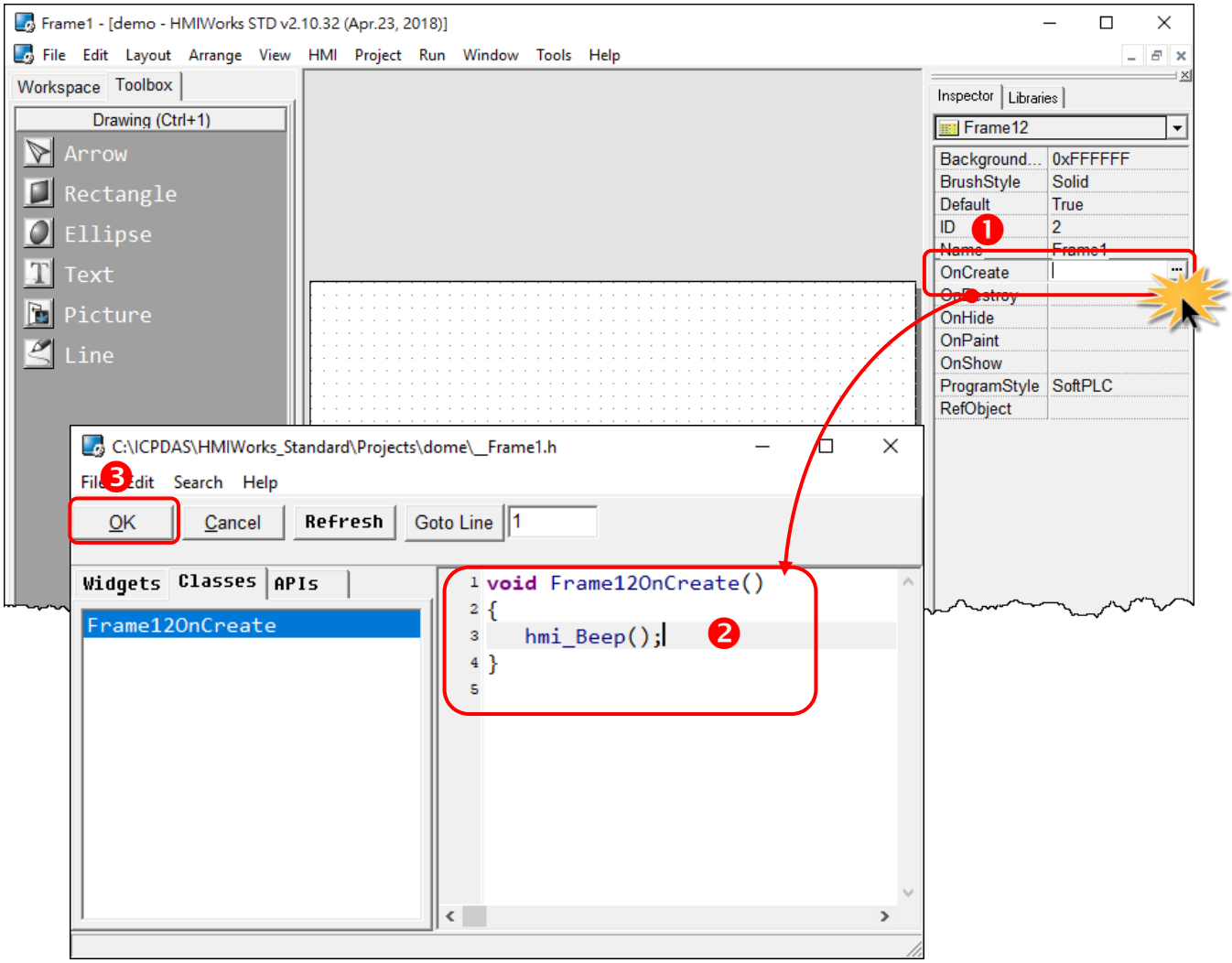
Implementing event handlers

The event handler is supported only in the frame of C, not Ladder. By default, double clicking on the component opens the programming window of the **OnClick** event handler if more than one event handlers that a component has.

Component /Frame	Existing Event Handler
Frame	OnCreate, OnDestroy OnHide, OnShow OnPaint
TextPushButton, BitButton, HotSpot,	OnClick, OnRelease
Slider	OnSliderChange
CheckBox	OnChange
Timer	OnExecute
PaintBox	OnPaint
RadioButton	OnRadioChange

➢ Take **OnCreate** event handler of a **Frame** for example.

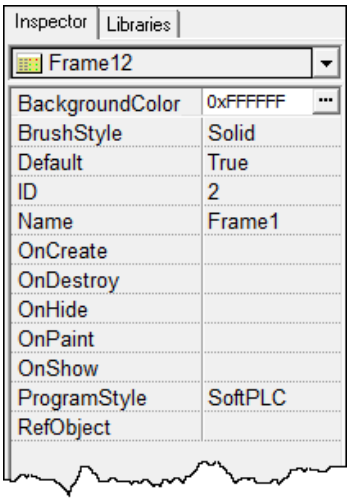
1. Click on the **"OnCreate"** field in the "**Inspector"** panel. Then click on the "**…**" button to open the programming window.
2. Here we use **hmi_Beep()** to sound a beep for example.
3. Click **"OK"** button to save the file and leave.



3.4.2 Frame

Unique Properties of a Frame

Click on the frame, and the properties of the frame are shown in the "**Inspector"** panel.



Properties	Description
BackgroundColor	The background color of the frame. The color is represented by a three-byte value in the hexadecimal form. From the highest byte to the lowest, it is the blue byte, the green byte, the red byte in sequence.
BrushStyle	Solid or Clear. If **BrushStyle** is set to **"Solid"**, then the setting of the **"BackgroundColor"** property does take effect. However this may make the **screen flash** if background color is quite different from the loaded picture. Setting **BrushStyle** Clear disables the **"BackgroundColor"** property and prevents the screen from flashing.
Default	Whether this frame is default frame or not. The default frame is displayed first after the TouchPAD device turns on.
ProgramStyle	Standard C or Ladder

Event handlers of a frame

For example, we have a frame which is named **"frame1"**, and

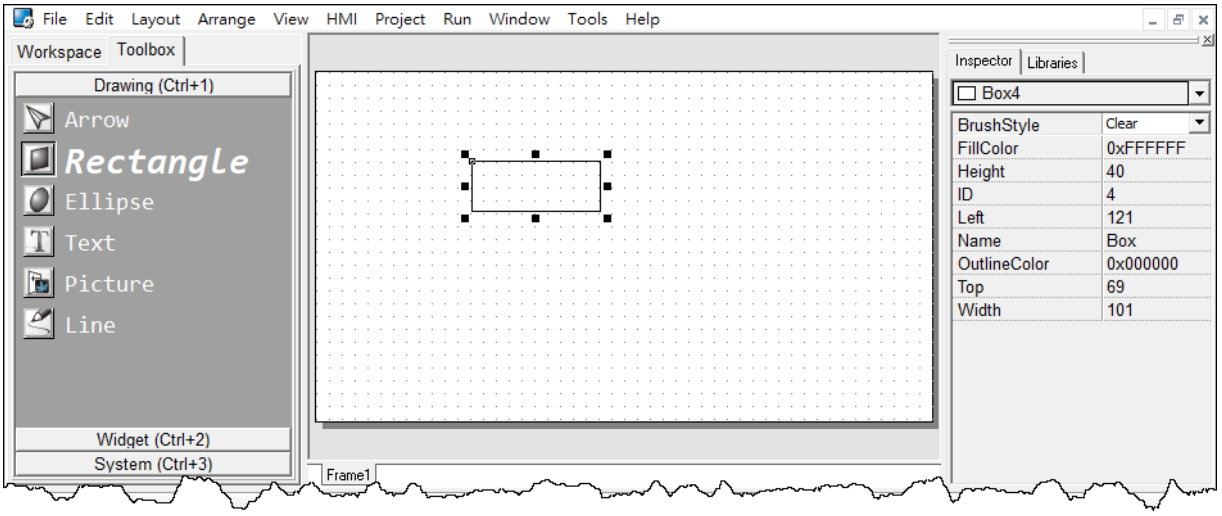
When entering the **frame1**,

- **OnCreate**: TouchPAD executes this **OnCreate** event handler of **frame1** first.
- **OnShow**: TouchPAD adds all the widgets used in the **frame1** after **OnCreate** is executed. Then executes the **OnShow**. (So **OnShow** has widgets to use)
- **OnPaint**: whenever TouchPAD needs to paint its screen. **OnPaint** is executed after **OnShow** when TouchPAD just switches to the **frame1**.

When leaving the frame1,

- **OnHide**: TouchPAD executes **OnHide** first,
- **OnDestroy**: TouchPAD removes all the widgets used in the **frame1** after **OnHide** is executed. Then executes the **OnDestroy**.

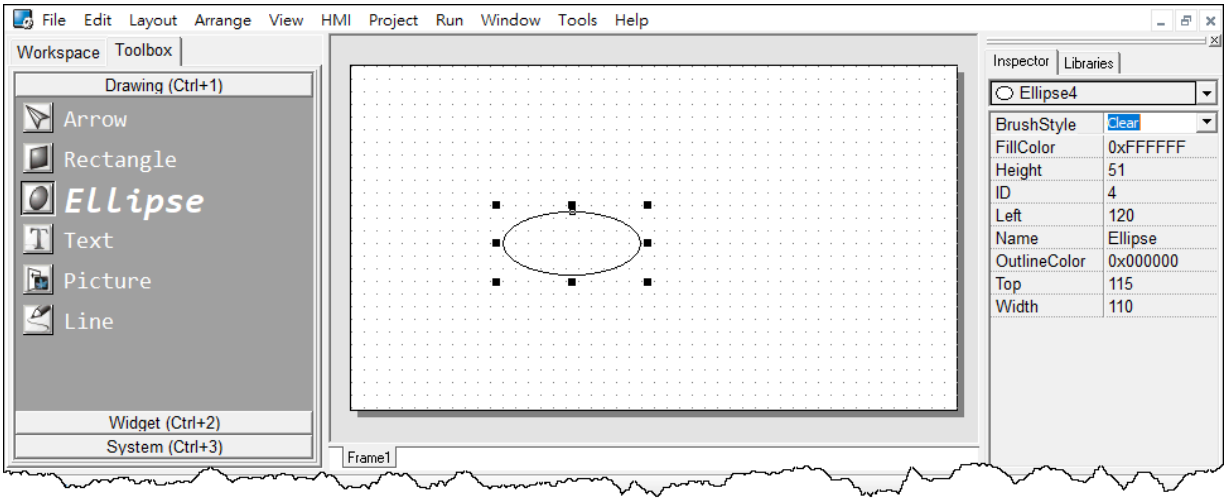
3.4.3 Rectangle



Unique Properties of **Rectangle**:

Properties	Description
BrushStyle	The style used to fill to a rectangle

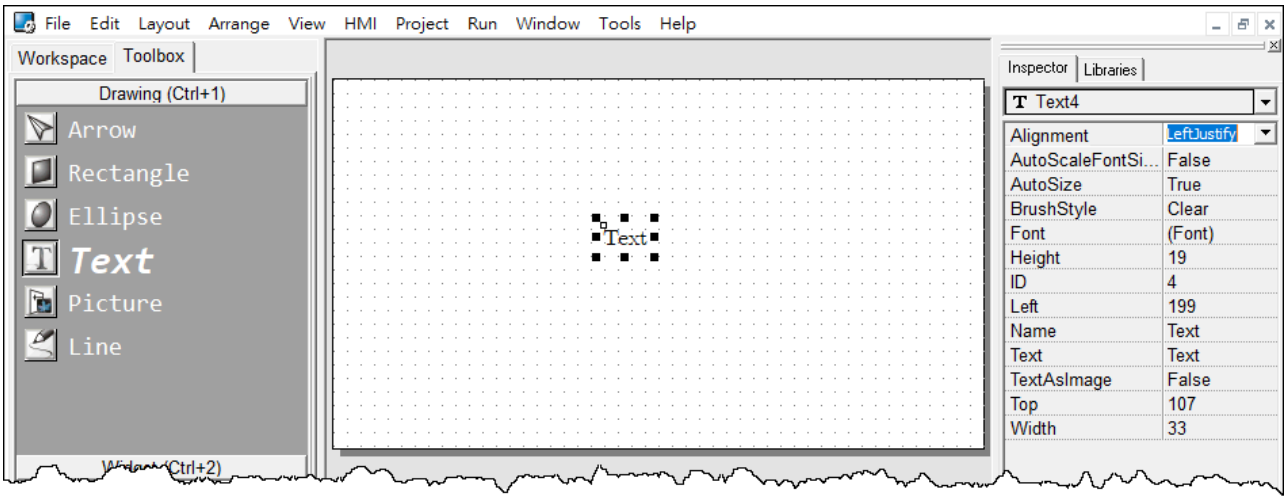
3.4.4 Ellipse



Unique Properties of **Ellipse**:

Properties	Description
BrushStyle	The style used to fill to an ellipse

3.4.5 Text



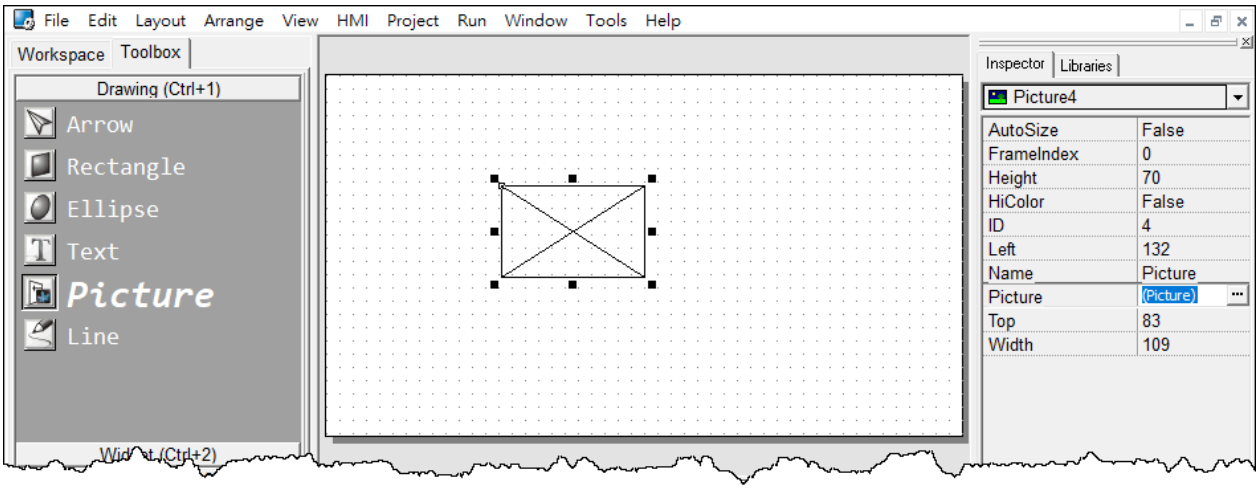
Another way to put a Text (a string) on the frame

Simply copy an text from the clipboard and paste it on the frame design area of HMIWorks. HMIWorks then create a **Text** component and then load the string from clipboard automatically.

Unique Properties of **Text**:

Properties	Description
Alignment	This property determines where to locate the string, Left, right, or center. (LeftJustify, RightJustify, or Center) **Note:** This property is enabled only when **AutoSize=True**
AutoScaleFontSize	Automatically scale the font size to fit the rectangle which encloses the Text. **Note:** This property is enabled only when **AutoSize=True**
AutoSize	True or False. This property is used to indicate that whether the size of the rectangle which encloses Text can be automatically changed to cover the whole string.
BrushStyle	The style used to fill the rectangle that encloses the Text
TextAsImage	True or False. Whether the text is stored as an image or not. If the text is treated as an image, it will take more space to store and more time to download.

3.4.6 Picture



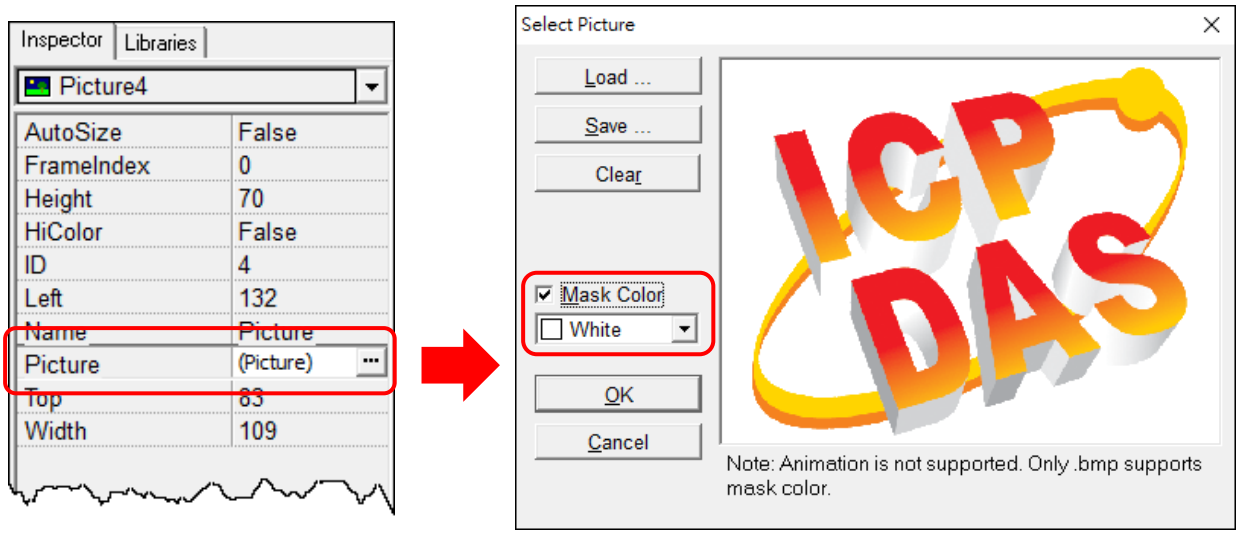
Unique Properties of **Picture**:

Properties	Description
AutoSize	True or False. This property is used to indicate that whether the size of the **Picture** can be changed or not.
FrameIndex	Ignored
HiColor	True or False. This property decides whether the loaded picture is stored as 16-bit color (True) or 8-bit color (False). **True:** The color depth for the picture is now set to 16-bit. Although 16-bit images occupy much more memory space, they provide much better quality. **False (default):** The color depth for the image will be set to 8-bit or less. The image will, of course, occupy less memory space, but will be of a much poorer quality.
Picture	The picture to be loaded

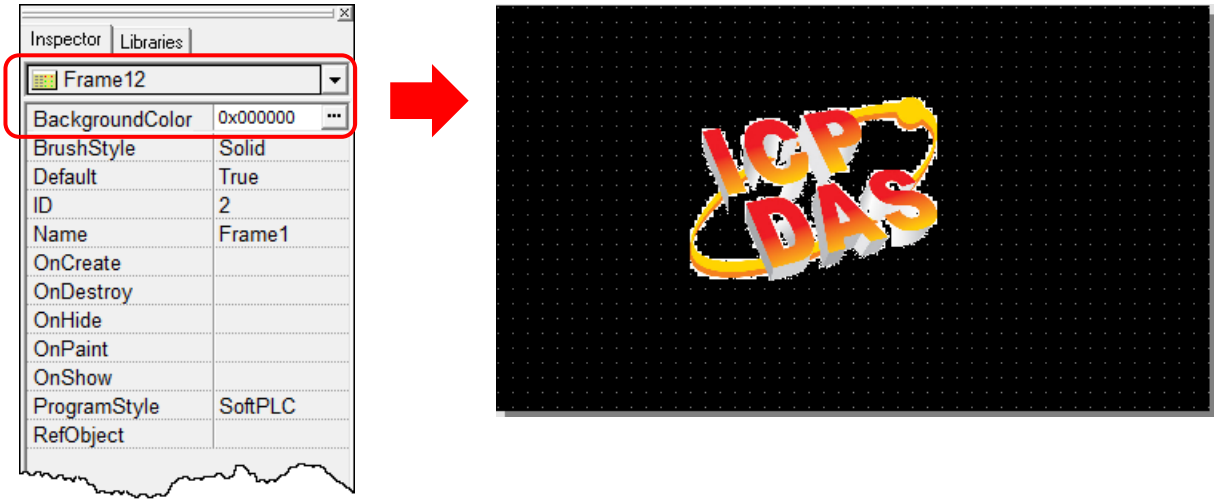
Loading a Picture

1. You can just copy an image from the clipboard and paste it on the frame design area of HMIWorks. HMIWorks create a Picture component and then load the image from clipboard automatically.

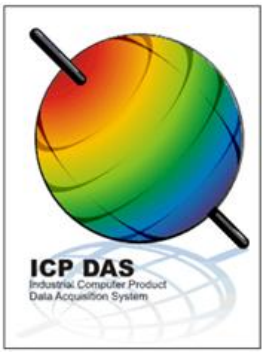
2. Click the **"Picture"** field in the **Inspector** (the **"…"** button) to open the "**Select Picture**" dialog to load a picture. There's a **"Mask Color"** option to achieve transparency as shown below.
   Note: Only "**.bmp**" files are supported for the "**Mask Color**" option.



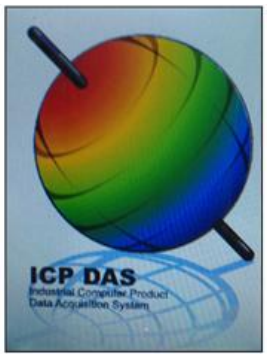
As you can see below, we select the **"Mask Color"** option as white to mask the **white** color, that is, the area of white color becomes transparent. Change the **background color** of the frame to **black** to illustrate the effect.



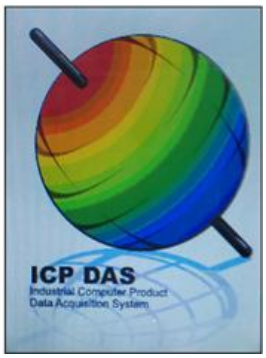
Trade-off between firmware size and resolution



In HMIWorks



HiColor = True
(16-bit color)
On TouchPAD (189KB)



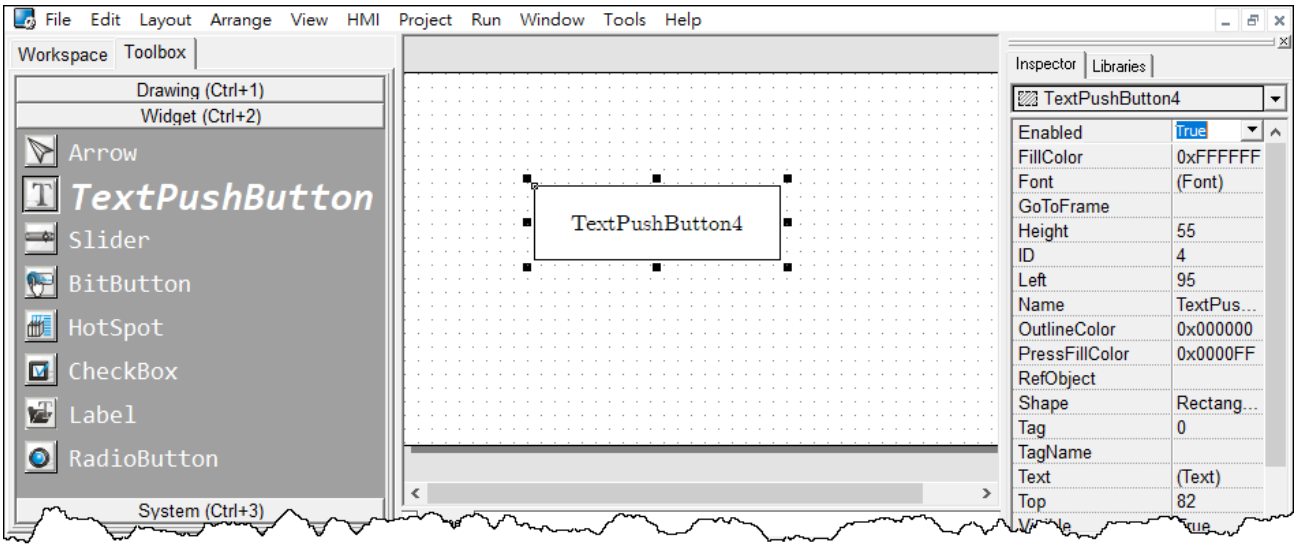
HiColor = False
(8-bit color)
On TouchPAD (69KB)

Above is the comparison between **"HiColor = True"** and **"HiColor = False"**. The left picture is original one in HMIWorks. The two right-side pictures are real photos. One is **"HiColor = True"** and the other is **"HiColor = False"**.

As you can see, setting **HiColor to False** (8-bit color) makes the photo have a not-smooth gradient part while setting **HiColor to True** (16-bit color) does not. Because 8-bit color does not have enough color (256 only) to represent the picture, similar colors are represented by the same color and this results in not-smooth gradient.

However, preventing pictures from not-smooth gradient costs TouchPAD bigger size of memory. Take above picture for example, setting **HiColor to True** (16-bit color) uses memory of 189 KB but setting **HiColor to False** (8-bit color) costs only 69 KB.

3.4.7 TextPushButton



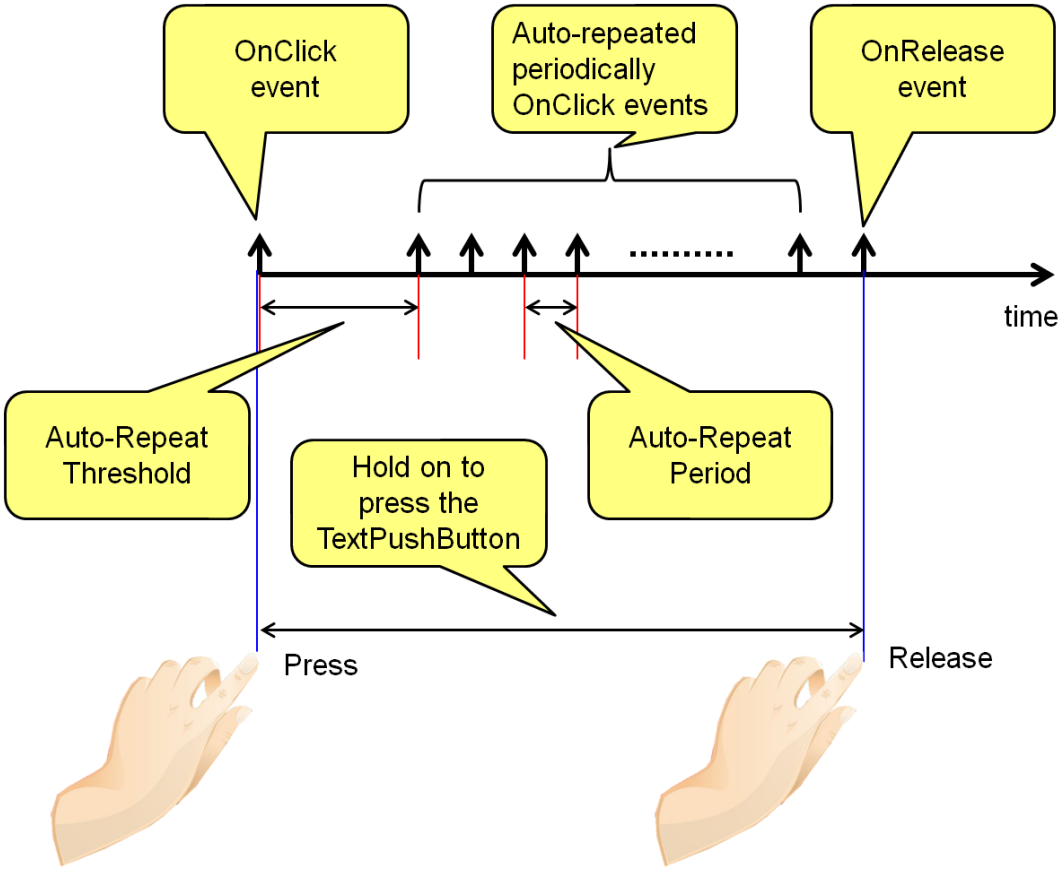
What is a TextPushButton?

A **TextPushButton** is a button with a **Text** on it. When a **TextPushButton** is pressed and not released, the status is changed. But the status is restored back to the original state after you release it.

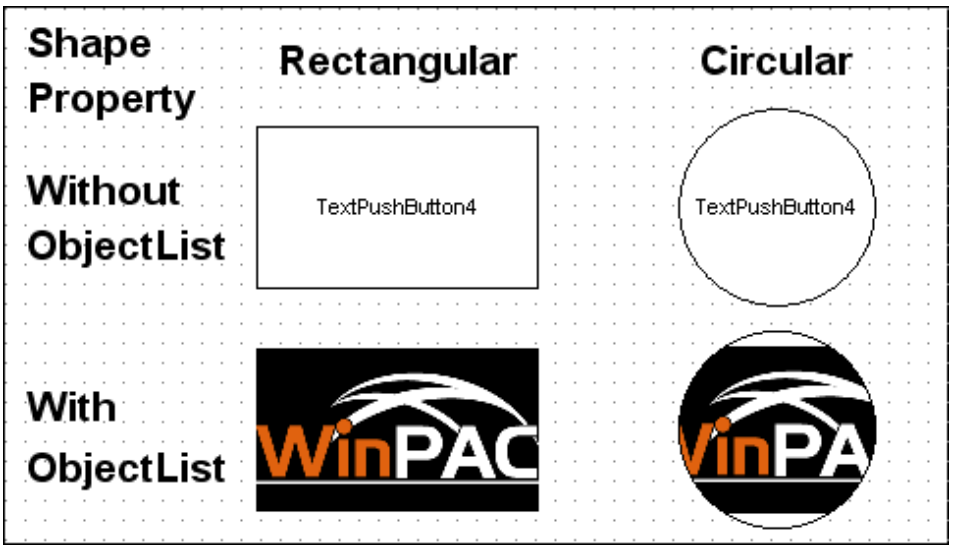
Unique Properties of **TextPushButton**:

Properties	Description
AutoRepeatPeriod	The period to hold on to press the **TextPushButton** to trigger one **OnClick** event again when in the programming type **"Standard C"**. (unit: ms)
AutoRepeatThreshold	After pressing the **TextPushButton** to trigger the **OnClick** event and continuing pressing, this property determines the threshold of the time that is required to trigger the first periodical **OnClick** event (not the first **OnClick** event) when in the programming type **"Standard C"**. (unit: ms)
PressFillColor	The color used to fill the **TextPushButton** when the **TextPushButton** is touched (but not yet released)
Shape	The shape of a **TextPushButton**, **Circular** or **Rectangular**.

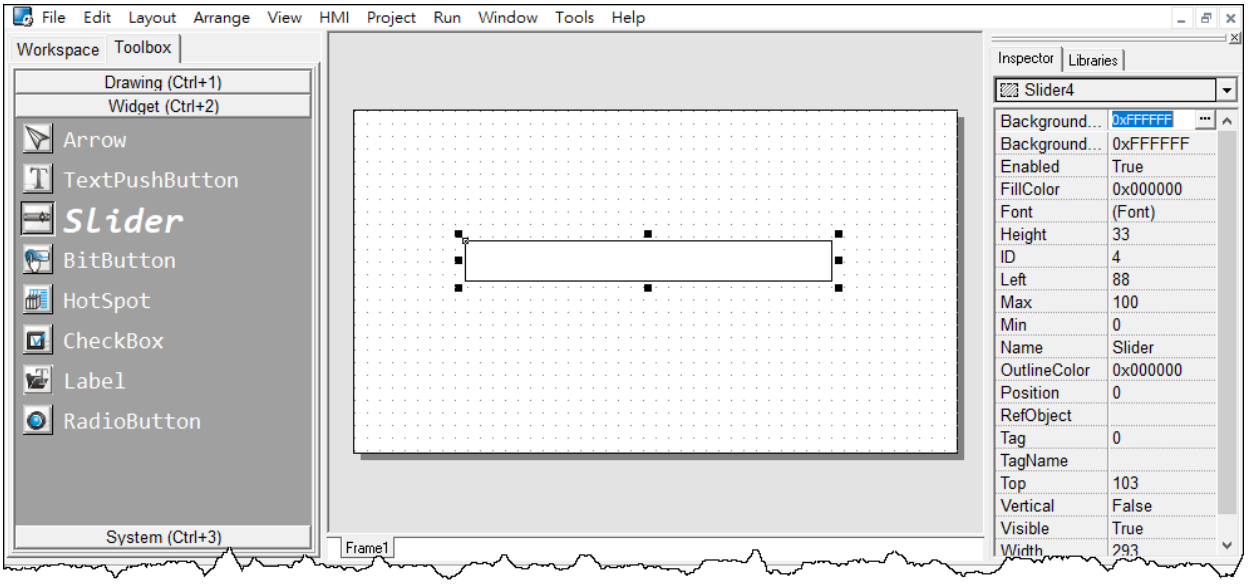
Triggered events



An example demonstrate the Shape property



3.4.8 Slider



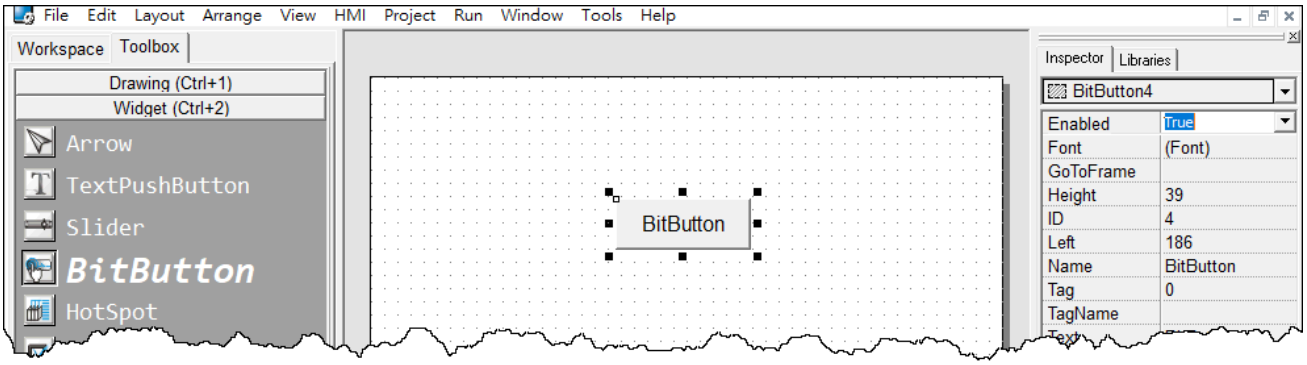
What is a Slider?

A Slider is a control element used to set levels. Usually, a Slider is used in volume control.

Unique Properties of **Slider**:

Properties	Description
BackgroundFillColor	The color used to fill the background of the Slider. The color is represented by a three byte value in the hexadecimal form. From the highest byte to the lowest, it is the blue byte, the green byte, the red byte in sequence.
BackgroundTextColor	The color of the text in the background of the Slider. The color is represented by a three byte value in the hexadecimal form. From the highest byte to the lowest, it is the blue byte, the green byte, the red byte in sequence.
Max	The maximum value of the Position
Min	The minimum value of the Position
Position	The value where the slider locate (between Max. and Min.)
Vertical	The direction of the Slider

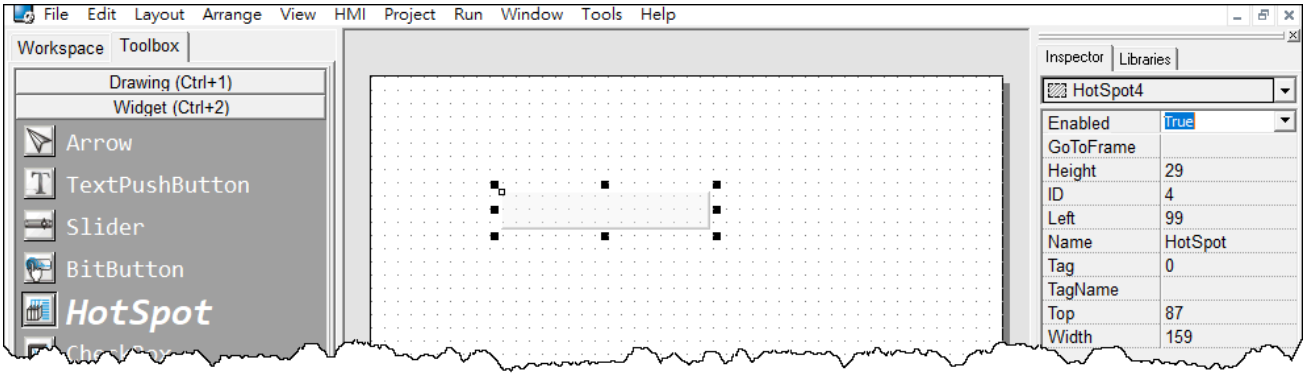
3.4.10 BitButton



What is a BitButton?

A **BitButton** is a button with 3D appearance and the status rebounds back if releasing the pressed button. When you press it, you can see that the **BitButton** is pressed “down”. This 3D-like appearance is achieved by two images so that it takes more spaces to store and more time to download than a **TextPushButton**.

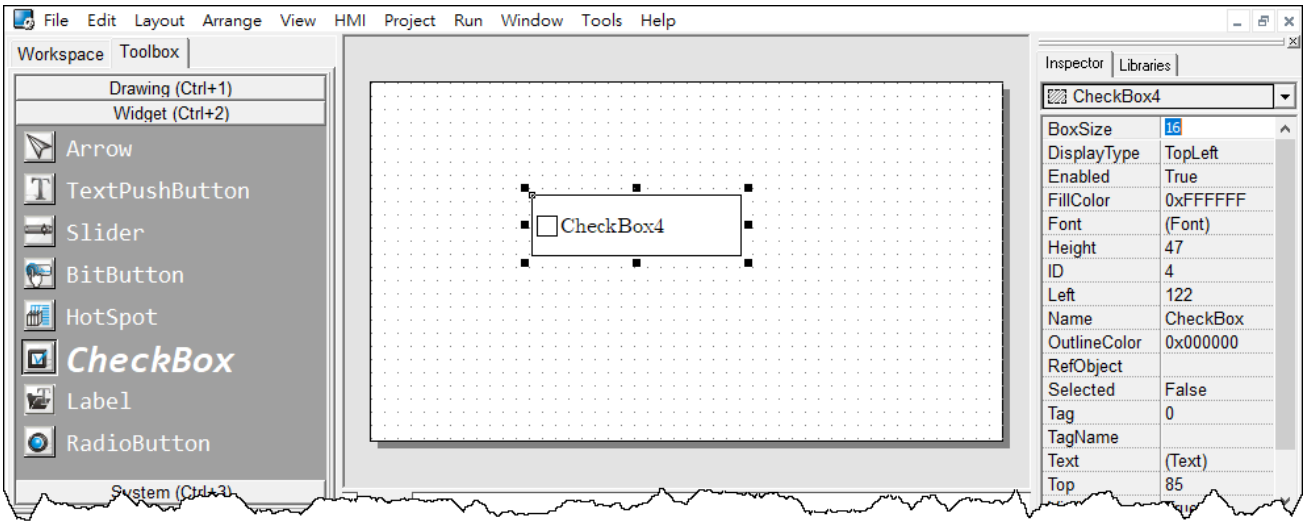
3.4.11 HotSpot



What is a HotSpot?

HotSpot decides an area which is capable of responding to on-click events. Usually, putting a **HotSpot** on the Drawing components (that is, Rectangles, Ellipses, Texts, Pictures, and Lines) makes them to respond to on-click events. After downloading to TouchPAD, a **HotSpot** is invisible.

3.4.12 CheckBox



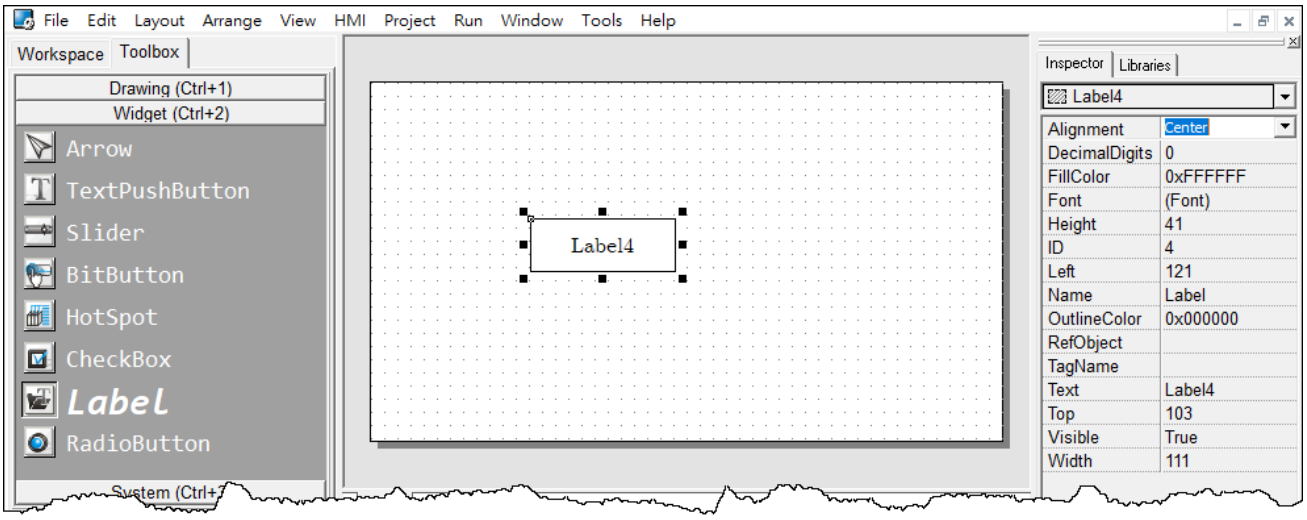
What is a CheckBox?

A **CheckBox** is a control element that provides a yes-no choice.

Unique Properties of **CheckBox**:

Properties	Description
BoxSize	The size of the checking box.
DisplayType	How to display the pictures which are loaded from **RefObject** property.
Selected	True or false. Whether the **CheckBox** is checked or not.

3.4.13 Label



What is a Label?

A **Label** is a Text put on TouchPAD to give information that may change at the run time.

Unique Properties of **Label**:

Properties	Description
Alignment	This property determines where to locate the string, left, right, or center. (LeftJustify, RightJustify, or Center)
DecimalDigits	The power to which ten must be raised to produce the value, say divisor, which is used to divide the value of the associated tag of this Label. The value of the tag must be divided by the divisor to show on the screen to represent decimal digits. **Note**: The property is supported only in programming type **"Ladder"**.

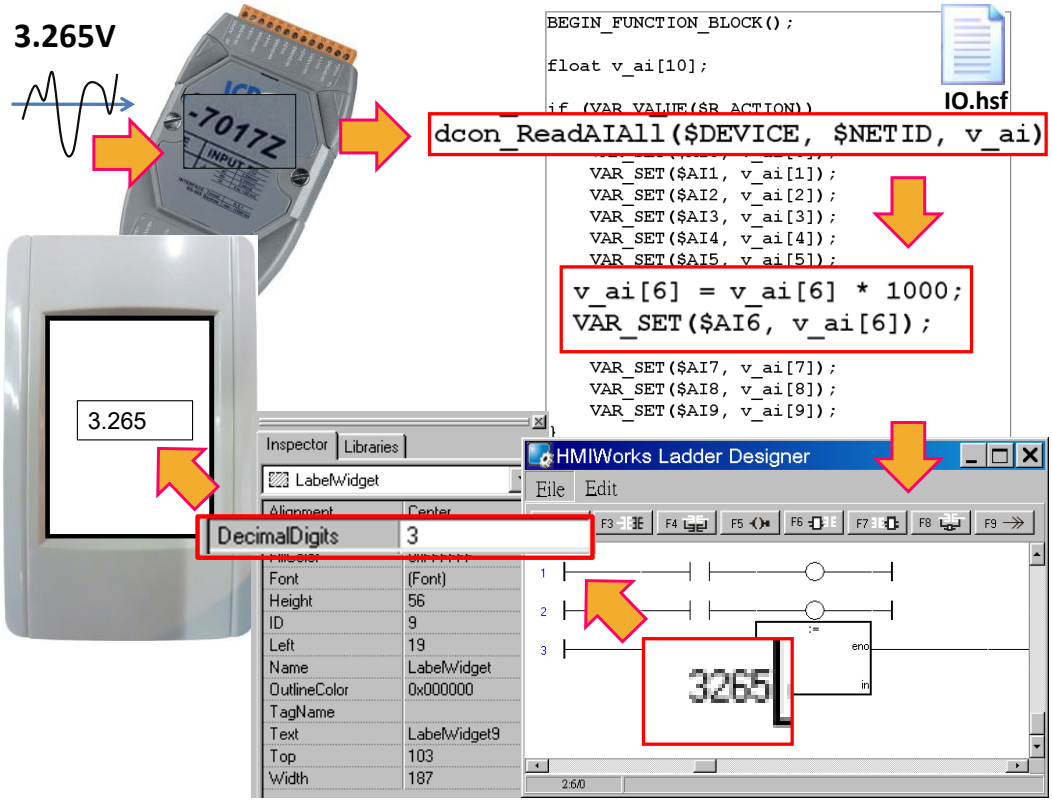
Representing decimals for Ladder Designer

The numbers used in the **Ladder Designer** are all integers. The decimals are not accepted in the **Ladder Designer**. However, in some cases, users may need to calculate or display decimals. So we provide a work-around method to handle these cases.

Take the I-7017Z module for example. Supposed that we use the I-7017Z module to read an analog value 3.265 V back from the remote side and we want to display decimals on the TouchPAD devices. But the **Ladder Designer** supports only integers. So we must handle this drawback to directly read back the AI value from the I-7017Z module in the **Ladder Designer**.

1. Set the property **"DecimalDigits"** to the number of digits in the right of the decimal point. For example, we set **DecimalDigits to 3**.
2. Modify the I/O module's **IO.hsf**. Let the read back AI value multiplied by ten of the **n**-th power where **n** is the value of **"DecimalDigits"**. You can find out I/O module's **IO.hsf** file in the following locations: "**[HMIWorks_install_path]\ bin\Modules**".
   For example, **IO.hsf** of I-7017Z is located in "**C:\ICPDAS\HMIWorks_Standard\bin\Modules\I-7000\I-7017Z**", where "**C:\ICPDAS\HMIWorks_Standard**" is the installation path of HMIWorks.
   And we modify the **IO.hsf** to make **v_ai[6] = v_ai[6] * 1000**; Supposed we use channel 6 to read back AI value.

As shown in the figure below, you can see that the tag **"$AI6"** in the **Ladder Designer** is 1000 times of the real value. With **DecimalDigits set to 3**, the correct value 3.265 is displayed on TouchPAD.



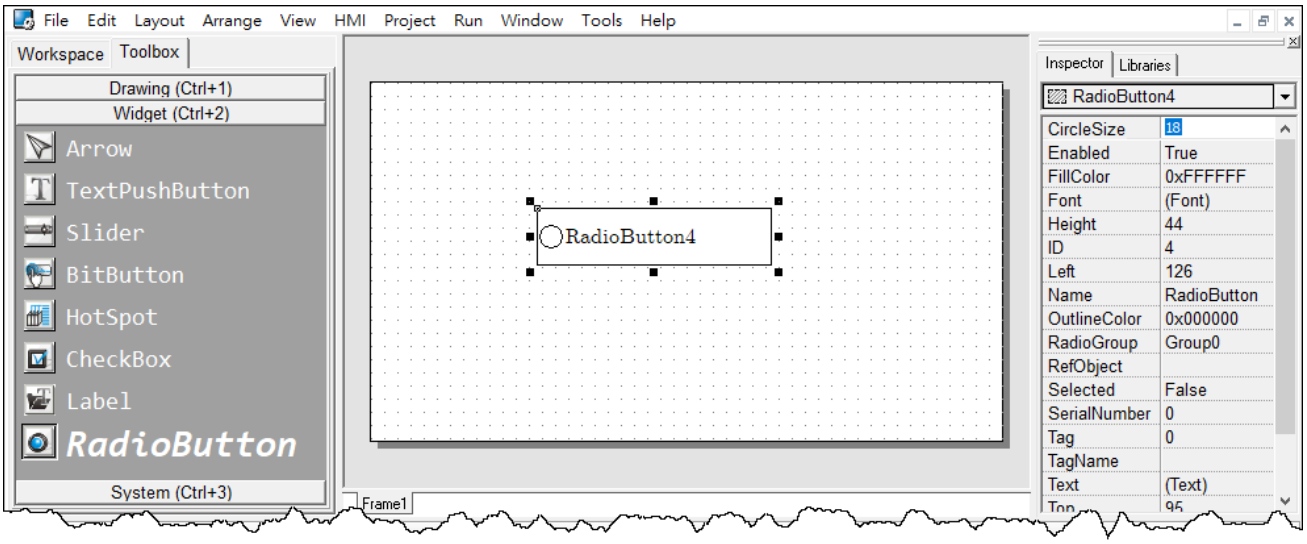
Representing decimals in the C language

In the frame of **“Standard C”**, representing decimals may be difficult since **“sprintf”** function is not supported in HMIWorks.

We use **“usprintf”** (or **“usnprintf”**) to replace **“sprintf”**, but **“usprintf”** does not support the argument **“%f”**. In order to display a floating-point value, we provide a new API function for this purpose, the **“FloatToStr”** function as shown in the example below.

```
void TextPushButton4OnClick(tWidget *pWidget)
{
    float ret_sin;
    float angle = 1.57;
    static char str_sin[16];

    // sin
    ret_sin = sin(angle);

    // int FloatToStr(char *buf, float fVal, int precision);
    // the precision determine the number of the digits after the decimal point
    FloatToStr(str_sin, ret_sin, 3);
    LabelTextSet(&Label5, str_sin);    // The result is 1.000
}
```

3.4.14 RadioButton



What is a RadioButton?

The **RadioButtons** is used for a **"one-of-many"** selection. That is, only one of the **RadioButtons** in a particular group (we call it **RadioGroup**) can be selected.

Unique Properties of **RadioButton:**

Properties	Description
CircleSize	The size of the checking circle.
RadioGroup	The group in which only one **RadioButtons** can be selected. Each frame has at most 8 **RadioGroups**, from Group0 to Group7.
Selected	True or false. Whether the **RadioButton** is selected or not.
SerialNumber	The unique number started from 0 which is used to identify a **RadioButton** in a particular **RadioGroup**. The **SerialNumber** property is used only for users to know about which **RadioButton** is to use, for example, when using the **RadioButtonGroupValueSet** function. **Note1**: this is a read-only property and is assigned automatically. **Note2**: when a **RadioButton** assigned a tag with the **TagName** property, then all the other **RadioButtons** in the same **RadioGroup** are assigned the same tag to their **TagName** property at the same time. Depending on the value of the tag (usually, the tag represent a I/O from the remote side), certain **RadioButton** is selected if its **SerialNumber** property is equal to the value of tag.

TagName property has different behavior

Unlike other widgets, several **RadioButtons** in the same **RadioGroup** have the same **TagName** property. Since **RadioButtons** together provide a **"one-of-many"** selection, the value of the **TagName** property is the same among all the **RadioButtons** in a particular **RadioGroup**.

For example, supposed we have 3 **RadioButtons**, 0, 1, 2, where 0, 1, 2 are their **SerialNumbers**. And they are all specified in a **RadioGroup**, Group0. If we specified the **TagName** with an AI tag, named Dev_AI0, then we have the following behaviors:

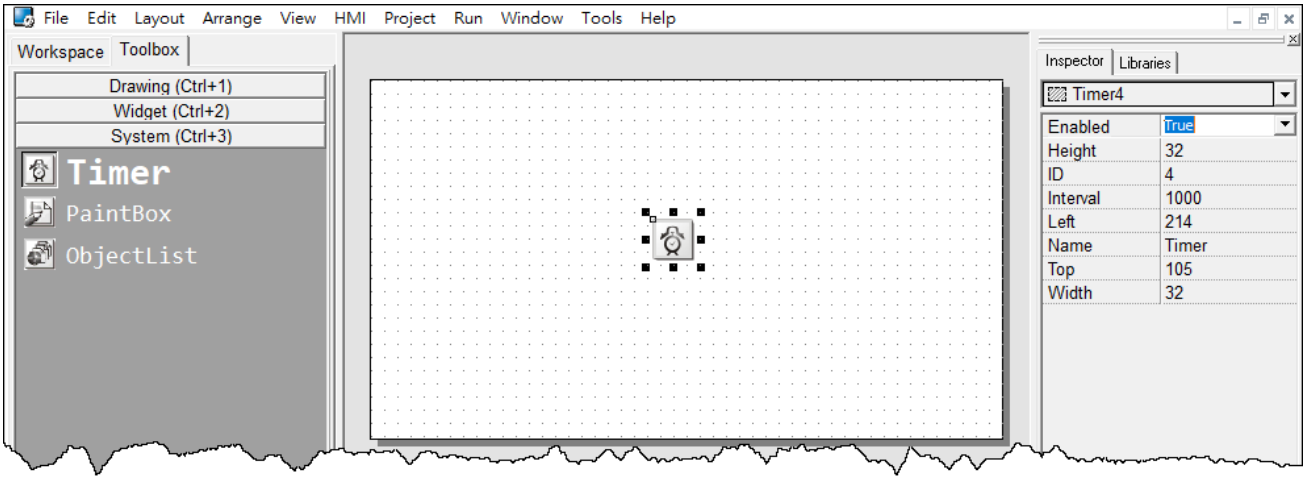
1. When Dev_AI0 = 0, only **RadioButton** with **SerialNumber** 0 is selected (while the other two are unselected).
2. When Dev_AI0 = 1, only **RadioButton** with **SerialNumber** 1 is selected.
3. When Dev_AI0 = 2, only **RadioButton** with **SerialNumber** 2 is selected.

OnRadioChange property

Unlike the **TagName** property, each **RadioButton** has its own **OnRadioChange** event handler. An example as shown below:

```
void RadioButton6OnRadioChange(tWidget *pWidget, unsigned char ucValue)
{
        //
        // ucValue is the serial number of the selected RadioButton in the
        // RadioGroup which contains this Radio Button. The RadioButton which
        // triggers this OnRadioChange event handler is not necessarily the
        // same as the selected RadioButton.
        //

}
```

3.4.15 Timer



Note: This component is supported only in programming type **"Standard C"**.

What is a Timer?

A Timer is a component that executes the **OnExecute** event handler every specified interval.

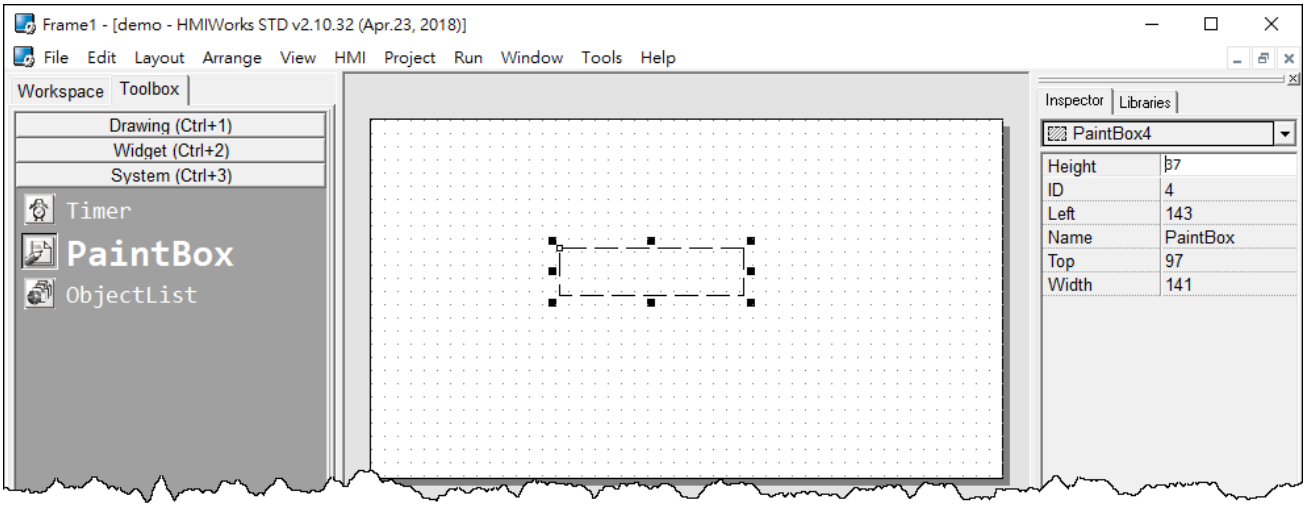
Using a Timer

Note that you should not worry about the size or the location of the Timer because the **Timer** is invisible when downloaded to the TouchPAD. Also it's not necessary to put the **Timer** on the frame panel.

Unique Properties of **Timer**:

Properties	Description
Enabled	Whether the Timer is enabled or not.
Interval	The time span of two consecutive **OnExecute** events

3.4.16 PaintBox



Note: This component is supported only in programming type **"Standard C"**.

What is a PaintBox?

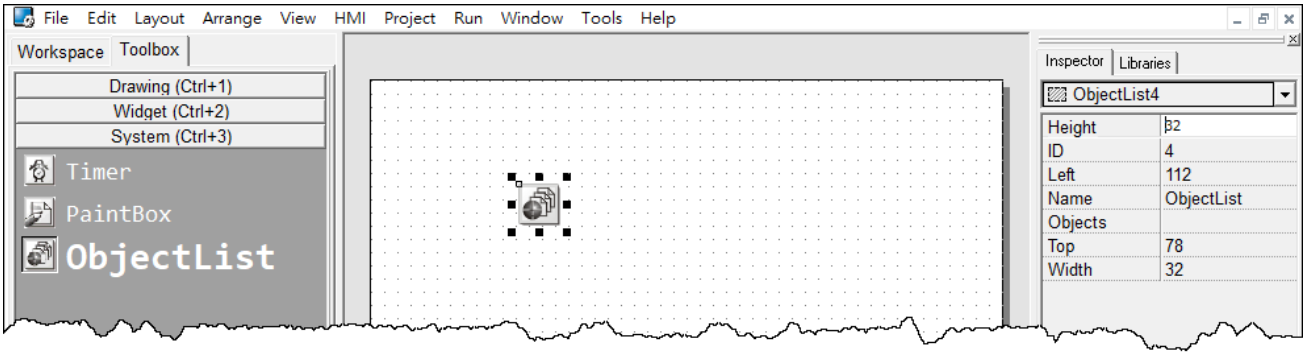
A **PaintBox** is a component which is used to paint shapes, such as rectangles, ellipses, etc., in the runtime.

Clearing a PaintBox

Use the **"hmi_SetForeground"** function to paint a white rectangle to clear the **PaintBox** in the example below. Refer to the Section 1.12 hmi_SetForeground of HMIWorks API Reference for more details.

```
//white; R-G-B; used to clear the PaintBox
hmi_SetForeground(pContext, 0xFFFFFF);

hmi_FillRect(pContext,
             widgetLeft(pWidget),
             widgetTop(pWidget),
             widgetRight(pWidget),
             widgetBottom(pWidget));
```

3.4.17 ObjectList



What is an ObjectList?

An **ObjectList** is a component which is used to maintain a list of library objects. Combined with **"RefObject"** properties of **TextPushButton**, **Slider**, **CheckBox**, and **RadioButton** components, users can easily toggle two or multiple images.

Unique Properties of **ObjectList**:

Properties	Description
Objects	The maintained library objects.

Options about images in **ObjectList** dialog:

Properties	Description	Default
Fit to Widget	Resize the images in the **ObjectList** to cover the whole area of the widget which references to it.	True
High Color	Render the images in the **ObjectList** as 16-bit color (high color) or 8-bit color when compiling.	True
Force Compile	Force HMIWorks to compile the images of this **ObjectList** which is not used by any widgets.	False

Note:

To display transparent color (mask color) correctly, the following conditions must be satisfied.

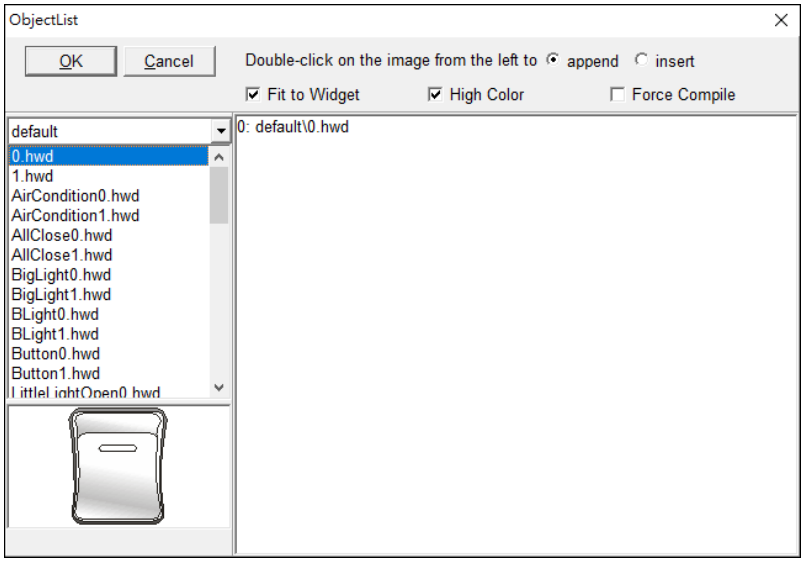
1. The **"Fit to Widget"** option in the **ObjectList** dialog must be checked.
2. Each object of the **ObjectList** must contain only one Picture component. (Note that when you **"add to library"** the picture, it is grouped.)
3. **TextPushButton** with an **ObjectList** assigned to its **RefObject** property does not support the transparent (mask) color function when its Shape property set to **"Circular"**.

Using an ObjectList

1. Note that you should not worry about the size or the location of the **ObjectList** component because the **ObjectList** component is invisible after downloaded to the TouchPAD device.

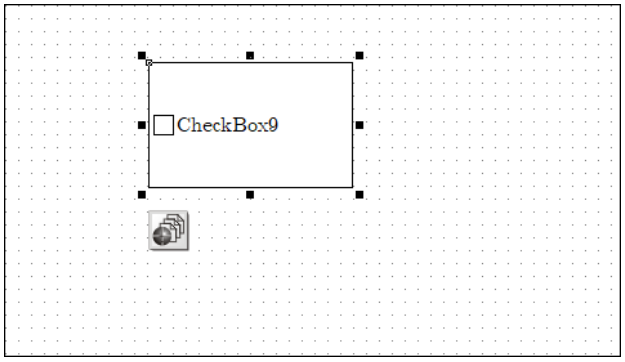
2. The **ObjectList** component maintains a list of a library objects and is used in a component (e.g. CheckBox) with the **RefObject** property. After downloading to the TouchPAD device, the images of the library objects replace the original display of the component. When the state/value of the component changed, users see only the images of the library objects displays in the order in the **ObjectList** according to the state/value of the component.

3. For example, add two library objects in the **ObjectList** by double clicking the **ObjectList** icon. Then the **"ObjectList"** window is displayed. Double click on the list of the library objects to add them to the right side panel.

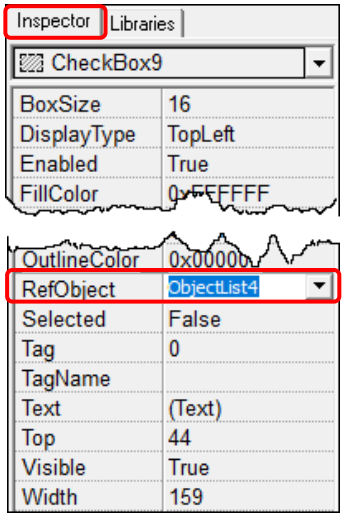


Note:
To delete the library objects in the **"ObjectList"** window, double click on the objects in the right-side panel.

4. Click and drag a **CheckBox** component on the frame panel for example. Be sure to make the size of the **CheckBox** component large enough to cover the whole image of the library object.



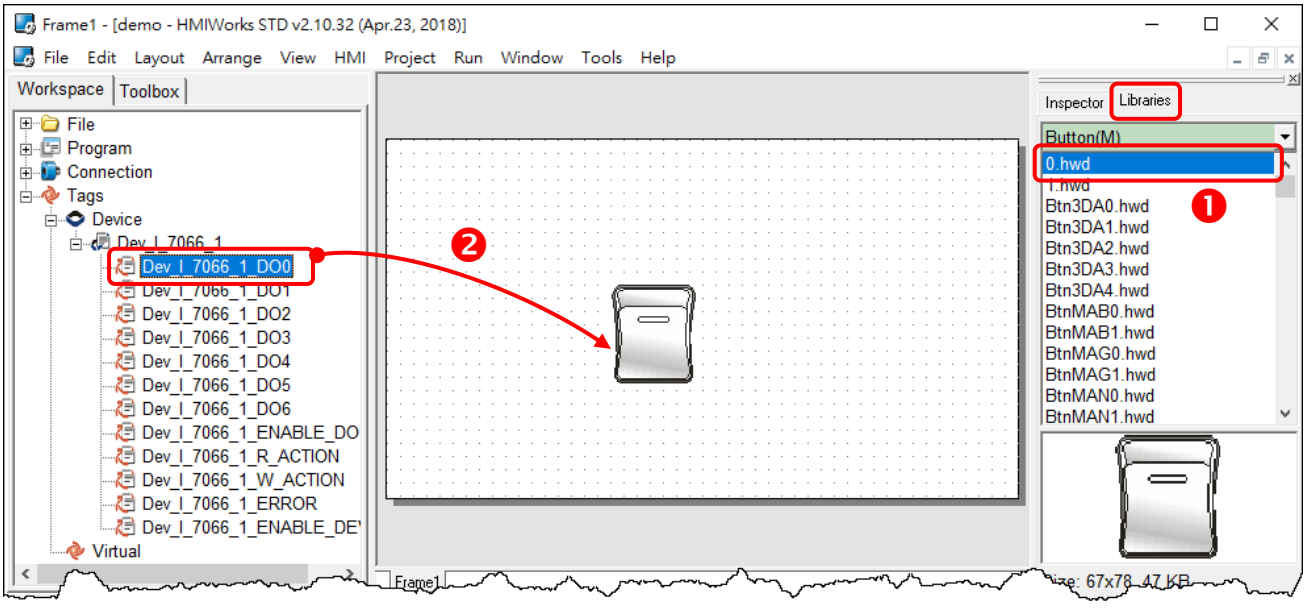
5. Go to the “**Inspector”** panel to select an option from the **“RefObject”** field for the **CheckBox** component. The selected **ObjectList** component is connected to the **CheckBox** component.



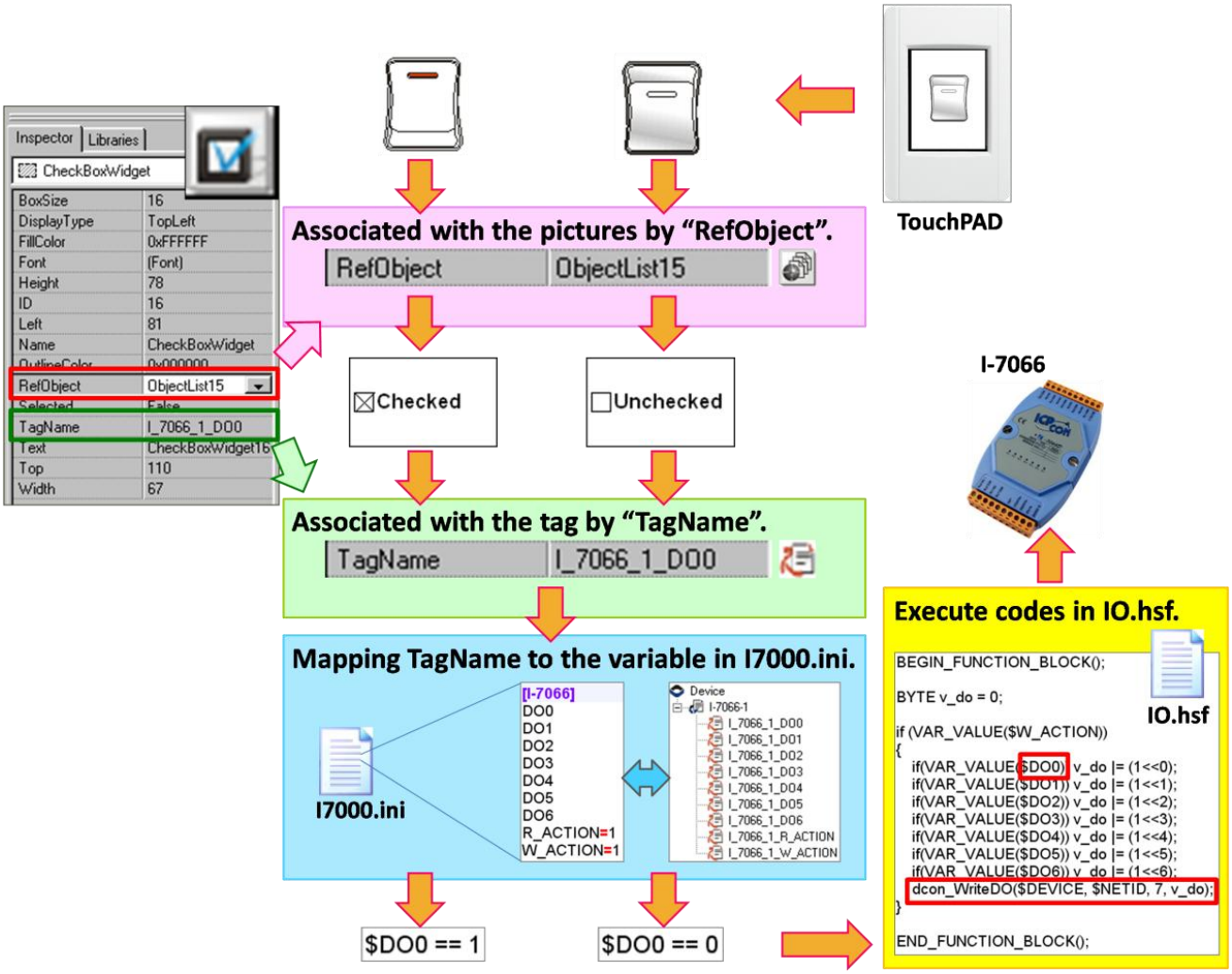
6. Build and download the project. You can see two images of the library objects toggle when the state the **CheckBox** component changes.

Relationships between TouchPAD and I/O module

Take the I-7066 module for example, click on the “**Register Devices(I/O) F3**” option from the “**HMI**” menu or press **<F3>** key to automatically generate tags and then drag and drop the tag on the frame.



HMIWorks does the followings to build the relationships between the TouchPAD device and I/O modules.



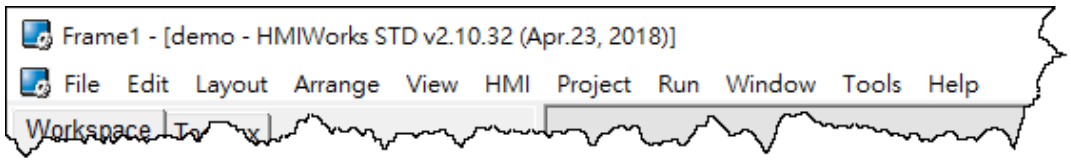
Note:

The **TagName** property takes effect only in the programming type Ladder. (It's easier in programming type **"Standard C"**. Control the I/O by using API function, **dcon_WriteDO**, in the event handler of the **CheckBox**.)

3.5 Menus

All the menus can be accessed from **“menu bar”** or the **“popup menu”**, which will be described in more detail below.

3.5.1 Menu Bar



The following is an overview of the **“Menu Bar”**, including a description of the usage of each function.

File

The options on the **File** menu enable you to open, close and save HMIWorks project. Refer to the Section 3.5.2.5 Import Images to Library will have more detailed information about **“Import Images to Library”**.

Edit

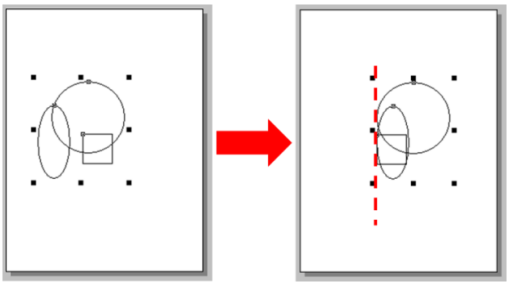
The options on the **Edit** menu enable you to modify components of the HMIWorks project, including copy, cut, paste, delete, rotate and flip, etc.

Layout

The options on the **Layout** menu enable you to align objects along axes, including vertical, top edge, left, and so on.

Note: All alignment functions refer to the last shape you draw. In below example, all alignment functions refer to the square.

For example, draw three shapes and select all the shapes then click “Align Left”, the result as following:



Arrange

The options on the **Arrange** menu enable you to make the selected object go down a level of the stacks and put components (the Drawing, the Widget and the system components) together as a set, that is, a group.

View

The options on the **View** menu enable you to set the HMIWorks interface, including change language (refer to Section 3.2.1 Language Options) and display Inspector, Library and Results panels.

HMI

The options on the **HMI** menu enable you to management Frame (add, delete and rename Frame) and create the tags (device and virtual) and Ladder Designer, etc. (refer to Section 3.3 Ladder Designer).

Project

The options on the **Project** menu enable you to configure settings project (refer to Section 3.2.2 Project Configurations), open the project folder and view project file, etc.

Run

The options on the **Run** menu enable you to set up TouchPAD, build current project and download to TouchPAD, etc. (refer to Chapter 4 Making a Simple Project).

Window

The options on the **Tools** menu enable you to setting display methods of the multiple windows, including cascade, tile horizontally and tile vertically, etc.

Tools

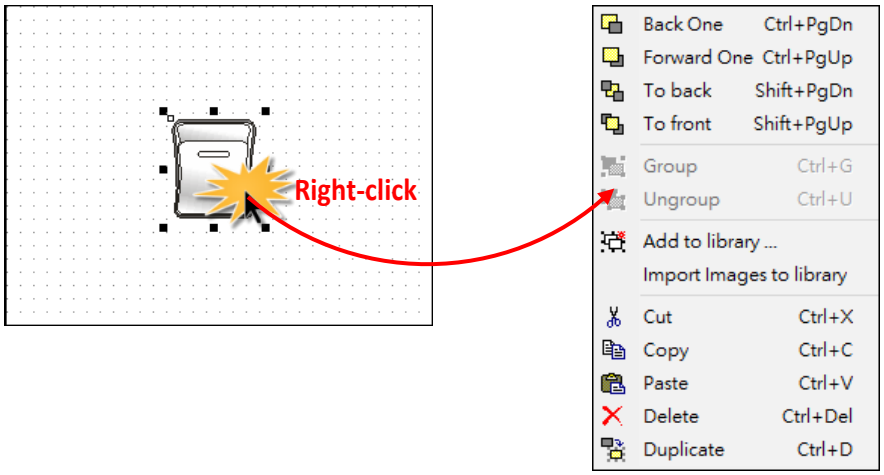
The options on the **Tools** menu enable you to update MiniOS8 of TouchPAD.
Note: This function is only suitable for the TPD-280U/238U/430/433/432F/433F and VPD-130(N)/132(N)/133(N)/142(N)/143(N).

Help

The options on the **Help** menu enable you to setting display welcome screen and view information about HMIWorks version number and computer memory, etc.

3.5.2 Popup Menu, Library Management

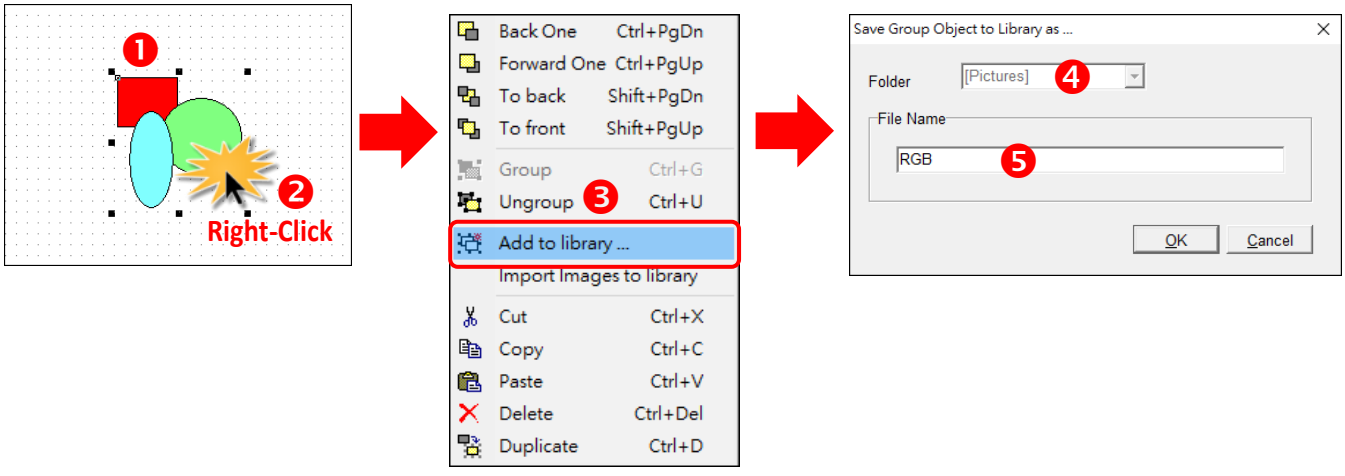
In frame design area, right-click on the component, a popup menu is displayed.

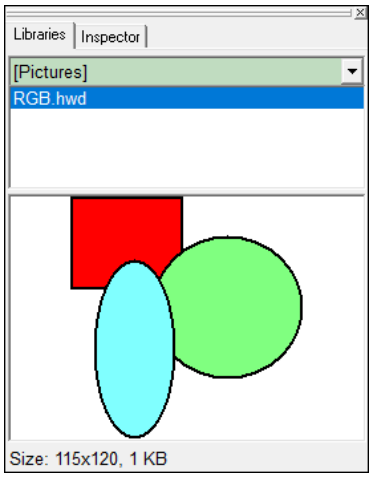


3.5.2.1 Adding items to library

All the items added have the file extension **"hwd"**. For example as described below:

1. Draw three shapes and group the selected items if necessary.
2. Right-click on the object we want to add to open the popup menu.
3. Click on **"Add to library ..."**
4. Specify the folder we want the added object locates in the drop-down menu. The default is **[Pictures]**.
5. Specify the name of the added object and save it to the library.





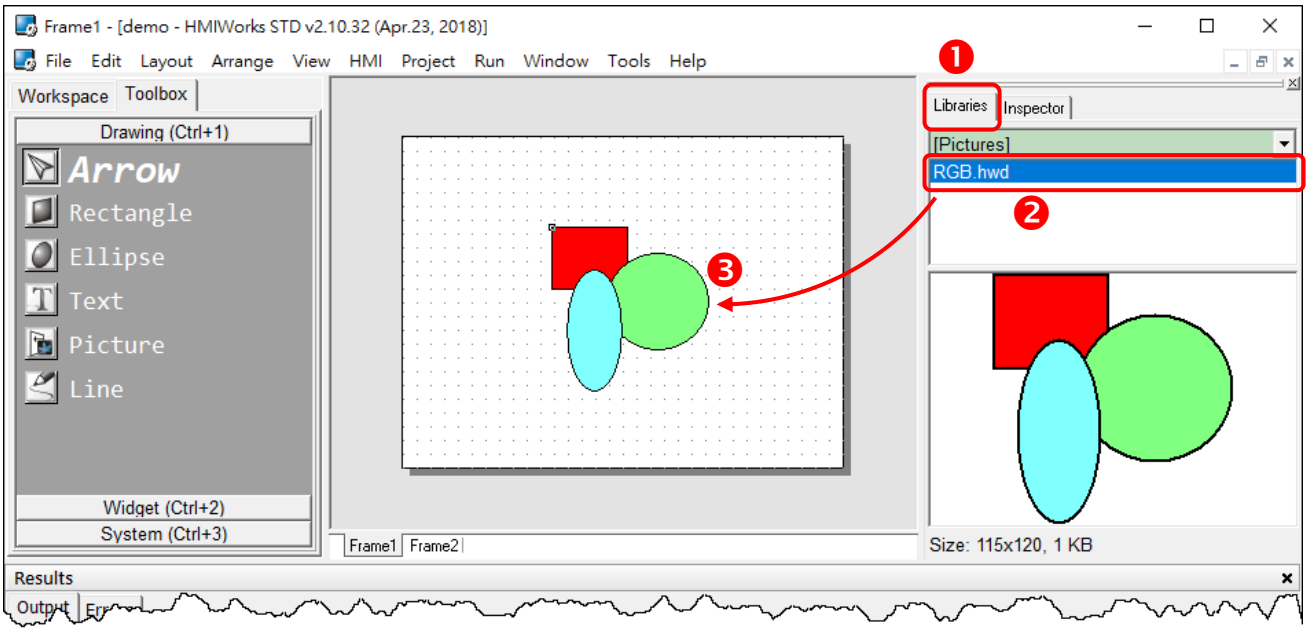
Note:

You can preview the library object in the "**libraries**" panel and the "**size**" information of that library object.

3.5.2.2 Using items from library

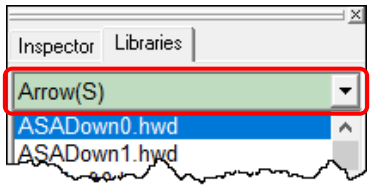
For example as below:

1. Click the "**Libraries**" tab to show the library panel.
2. Pick the object you want. You can preview the object in the preview box below.
3. Click (and not released) on the item in the preview box (or in the list) and then drag the item and drop it on the frame design area.

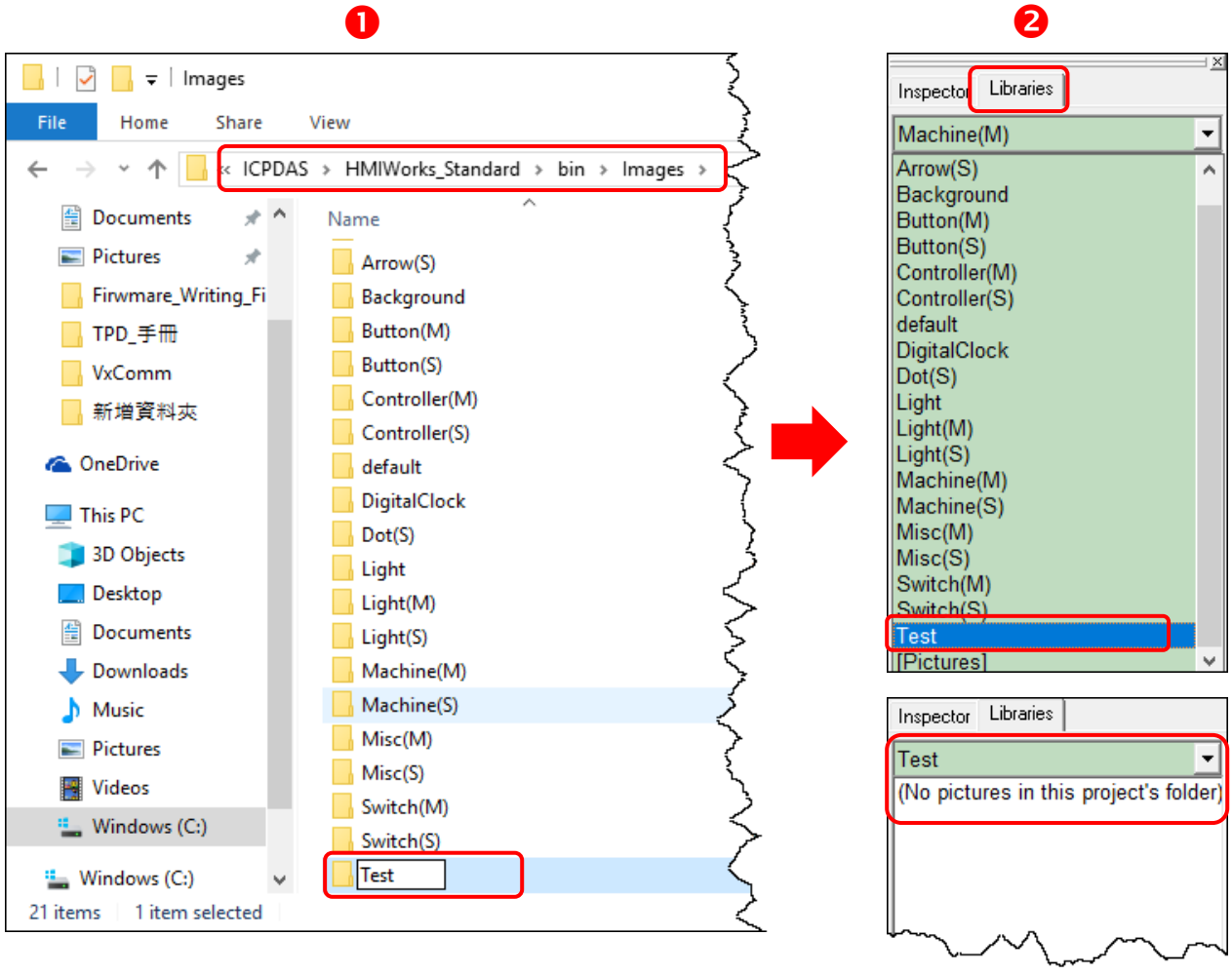


3.5.2.3 Adding a new folder into libraries panel

To add a new folder into the "**libraries**" panel, create a new folder in the following path:
"**HMIWorks_install_path\bin\Images**" where the HMIWorks_install_path is the installation path of HMIWorks.



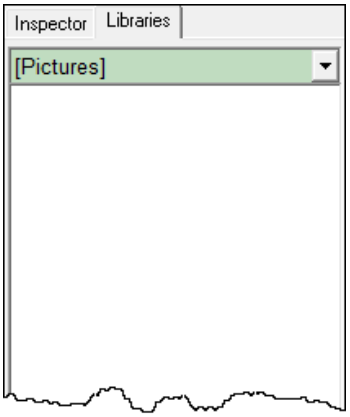
1. Supposed the installation path of HMIWorks is "**C:\ICPDAS\HMIWorks_Standard**". We want to add a new folder named **"Test"** into the "**libraries**" panel. Then all we have the do is creating a new folder named "**Test**" in the directory of "**C:\ICPDAS\HMIWorks_Standard\bin\Images**".
2. Re-open the "**libraries**" panel, you can see that the new folder **"Test"**. Of course, there's no library item in it. You should add items yourself.



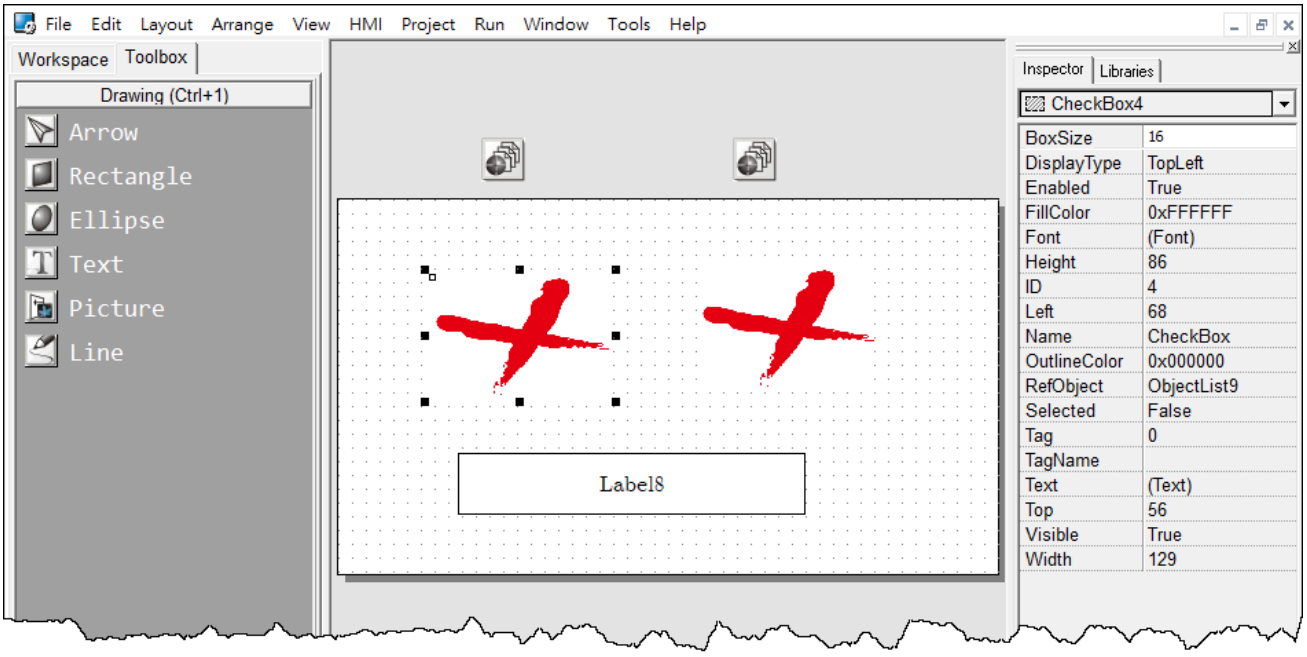
3.5.2.4 Special [Picture] directory in the project directory

Click the "**Libraries**" tab, select the **"[Picture]"** directory from the dropdown menu as shown in the picture below.

Unlike others options in that dropdown menu, **"[Picture]"** directory is at the location of the project directory. Any library that is added to the **"[Picture]"** directory is always together with the project and makes the project portable among different computers.

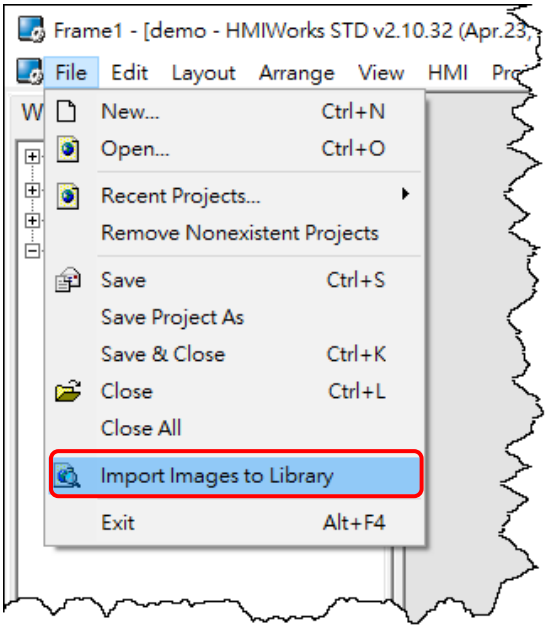


When opening a project, a red cross will be shown on the frame panel if HMIWorks fails to load the image as shown in the below picture, refer to [FAQ: How to fix the broken image (Red Cross) issue?] for instructions to resolve this issue.



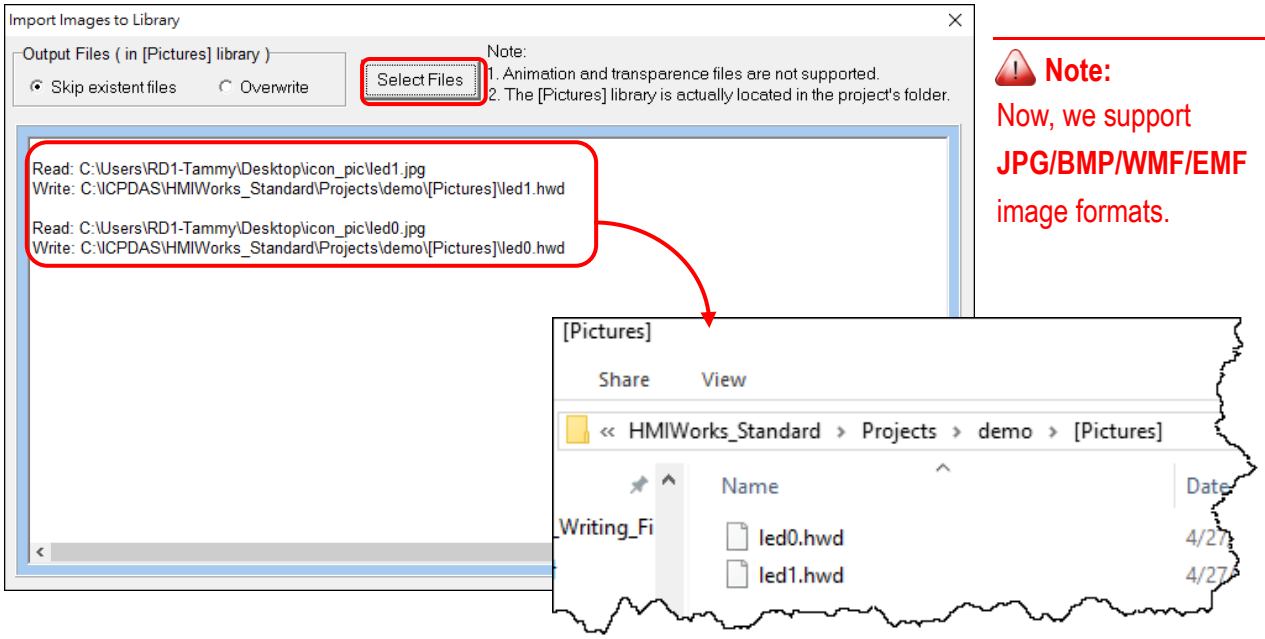
3.5.2.5 Import Images to Library

Click the **“Import Images to Library”** option from the **“File”** menu to select more than one image files, transform them into the **“.hwd”** file format which HMIWorks can recognize and finally put these files in the **[Pictures]** folder in the current project directory.



Note:
Since the transformed “**.hwd**” files are put in the **[Pictures]** folder of a project, users should create or open a project to execute this option.

As shown below, click the “**Select files**” button to execute.



Note:
Now, we support **JPG/BMP/WMF/EMF** image formats.

4. Making a Simple Project

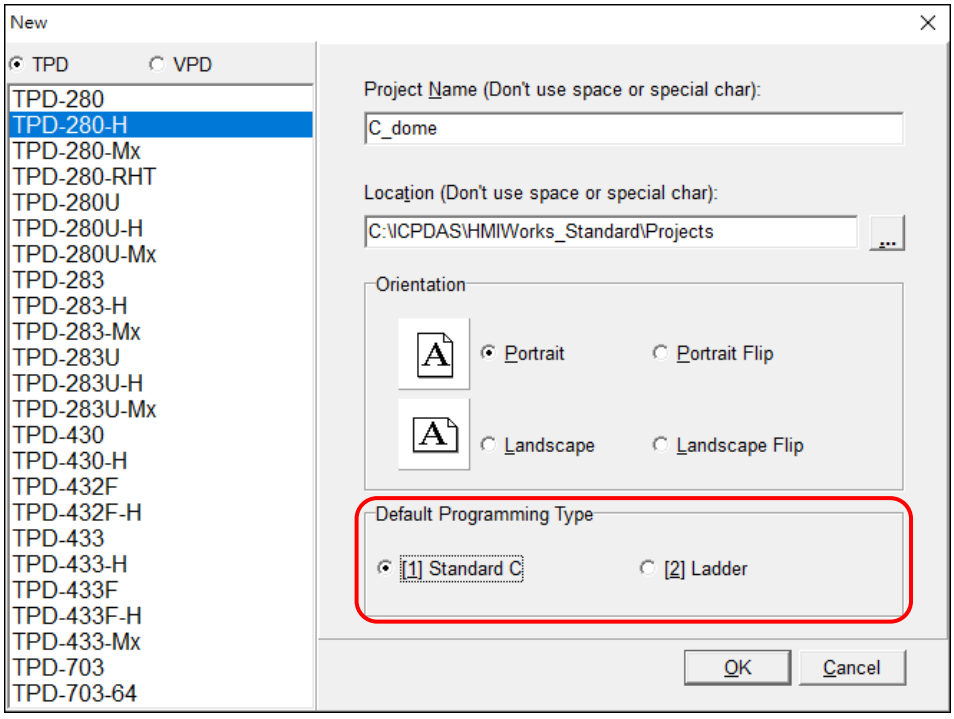
There are two programming types (Standard C and Ladder) in the HMIWorks. In this chapter, we introduce how to build your first project for each programming type and how to integrated TouchPAD with I/O modules.

4.1 Your First Project Using Standard C

Here, the TPD-280-H is used as an example, the following for a detailed description of the configuration process:

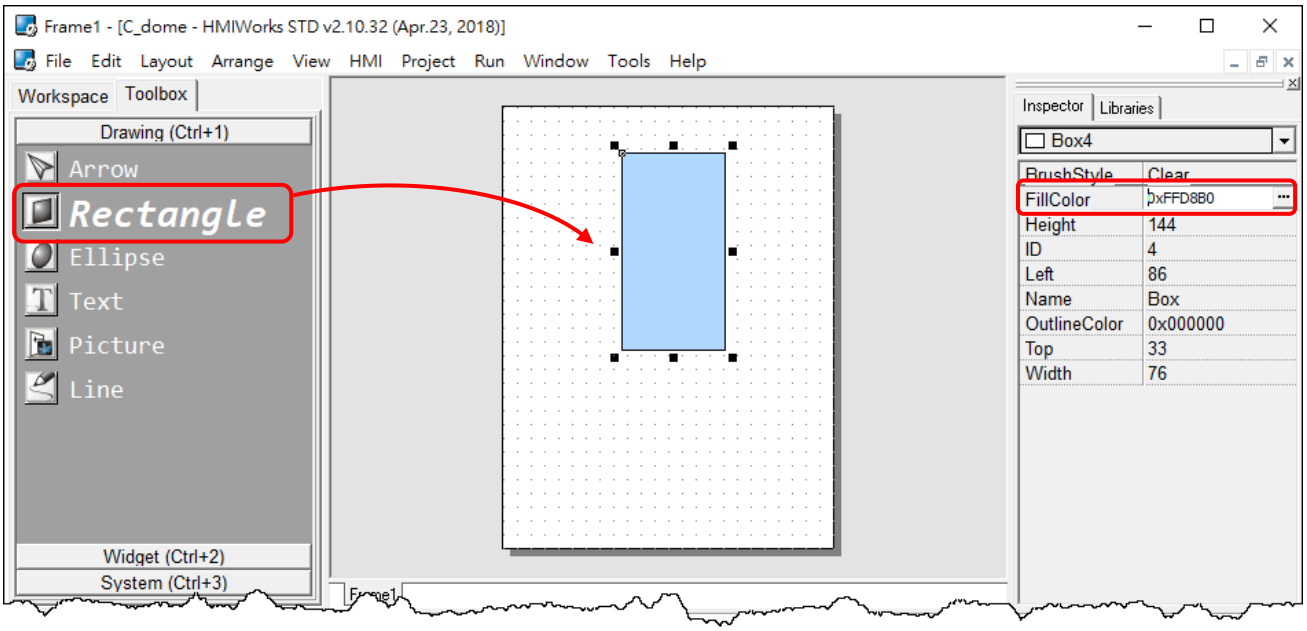
Step 1 Creating a new project

Click the "**New…**" option from the "**File**" menu and select the name of the TouchPAD model, specify the Project name, the Location, the Orientation, and the Programming Type. Here we choose **programming type** as **"[1] Standard C"**.

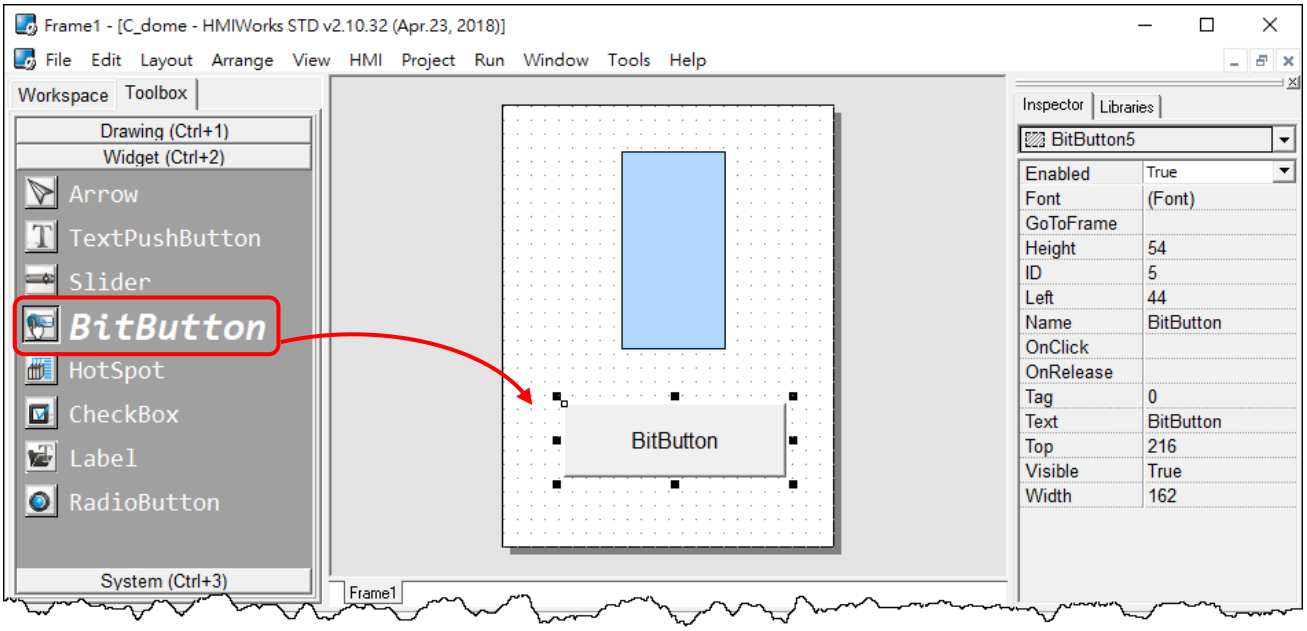


Step 2 Designing the Graphic User Interface

For example, draw a rectangular and fill the color. Of course, you can draw more complex and beautiful figures. Here, we simply demonstrate how to make a simple project.

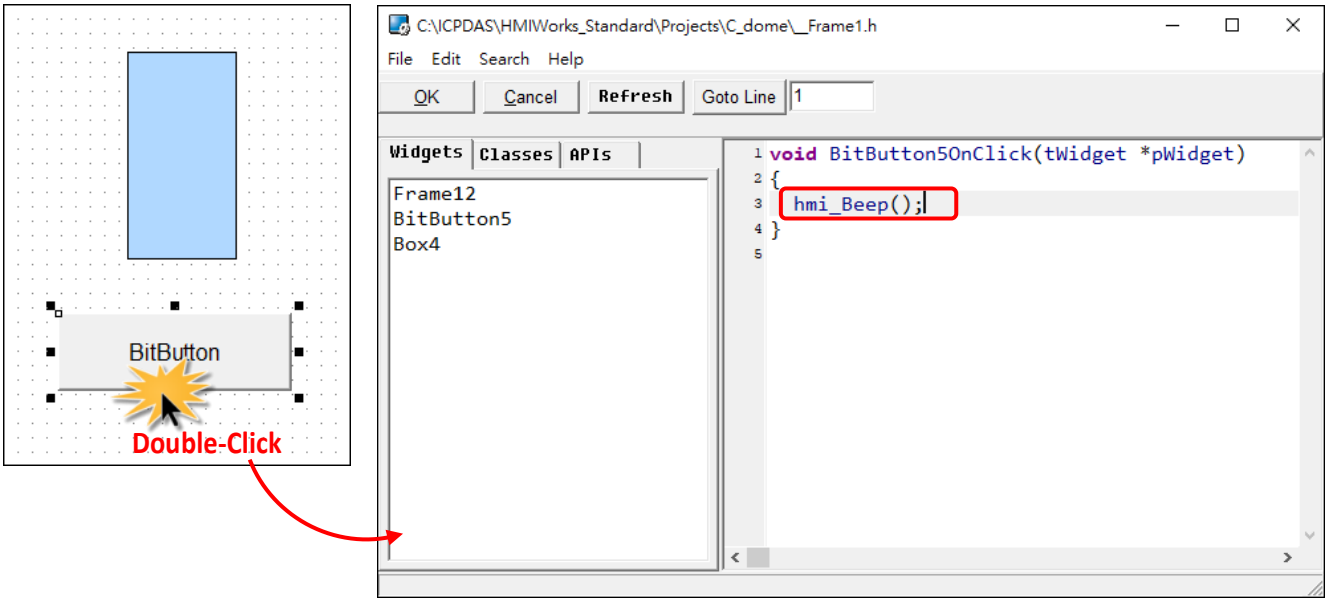


And then select a **Widget**. For example, pick a **BitButton**.



Step 3 Modifying Source Codes

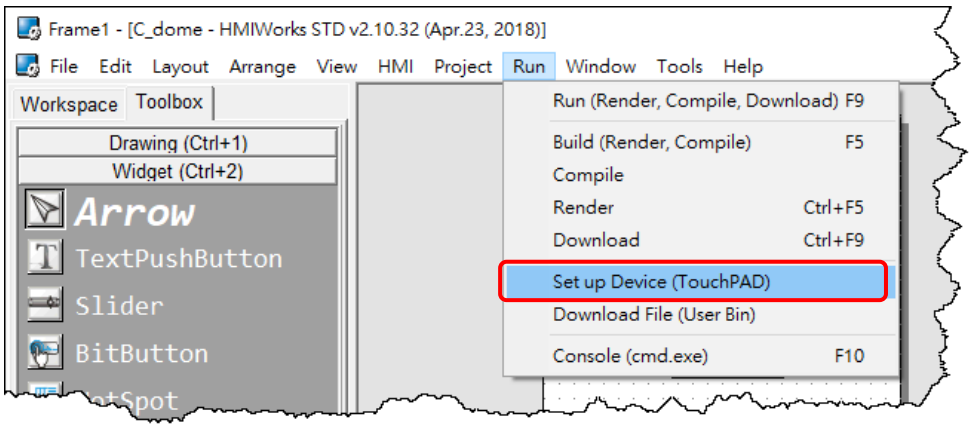
Double click the **BitButton** in the frame design area to open the programming window. Use **"hmi_Beep();"** to sound a beep for example, then click the **"OK"** button.



Step 4 Setup Device

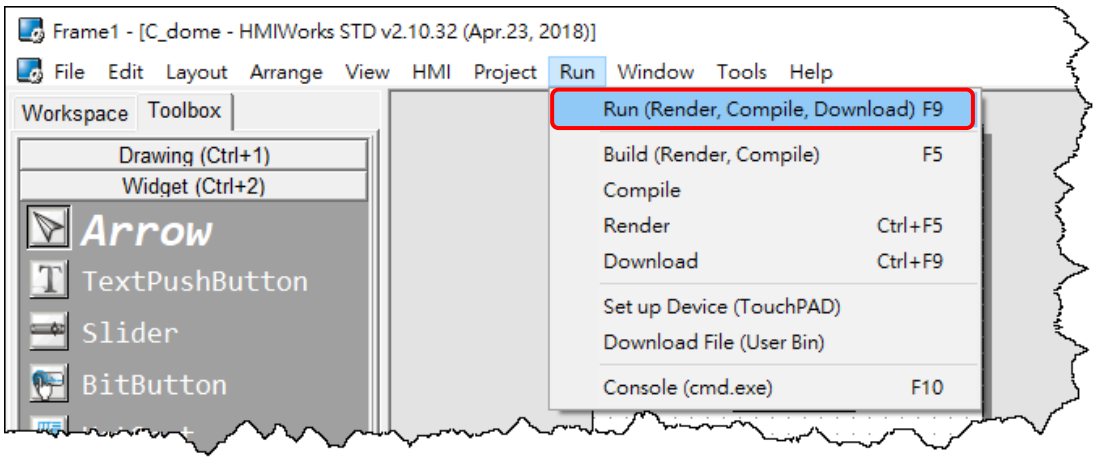
The downloading program method to the TouchPAD depends on the type of TouchPAD device, refer to the Section 3.4 Downloading Methods for TouchPAD of the TouchPAD Hardware User Manual for more detailed information.

In this example, we use the TPD-280-H device connect the Host PC via RS-485wiring and turn the rotary switch to "Update Only" mode (position 1) then reboot TouchPAD device.
Click the **"Set up Device (TouchPAD)"** option from the **"Run"** menu to select correct COM Port.

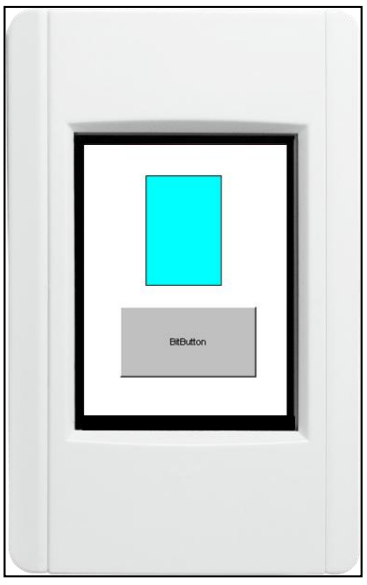


Step 5 Compiling and Downloading to Run

Click the **“Run (Render, Compile, Download) F9”** option from the **“Run”** menu, or press **<F9>** key. Once the download is complete, set the rotary switch to “Run Only” (position 0) and reboot TouchPAD device.



As shown in the figure below, pressing the button makes TouchPAD device sound a beep.

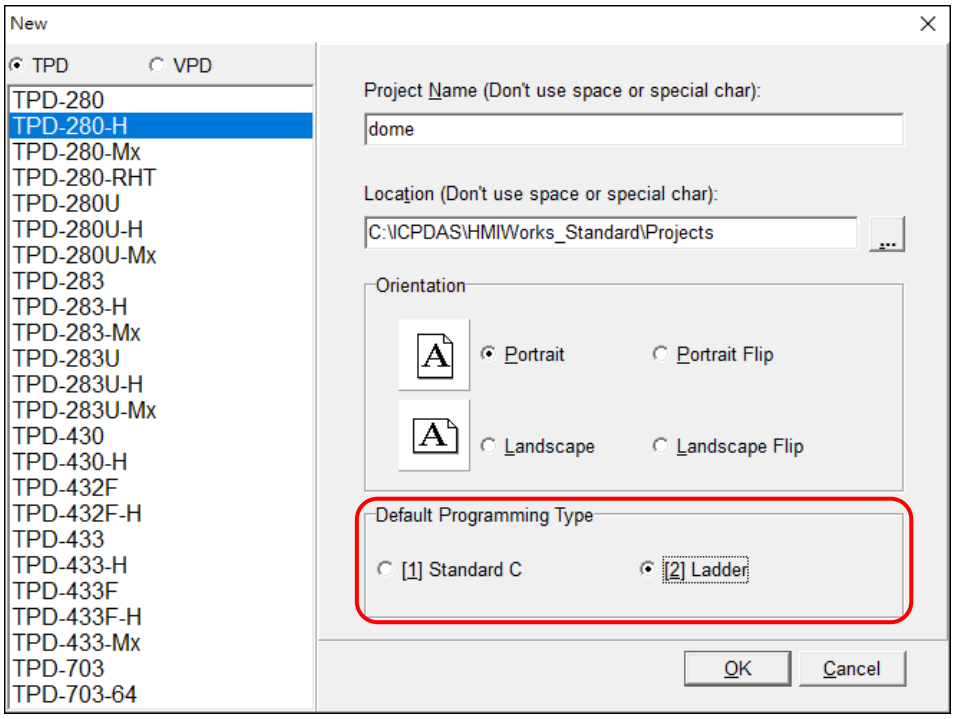


4.2 Your First Project Using Ladder

Here, the TPD-280-H is used as an example, the following for a detailed description of the configuration process:

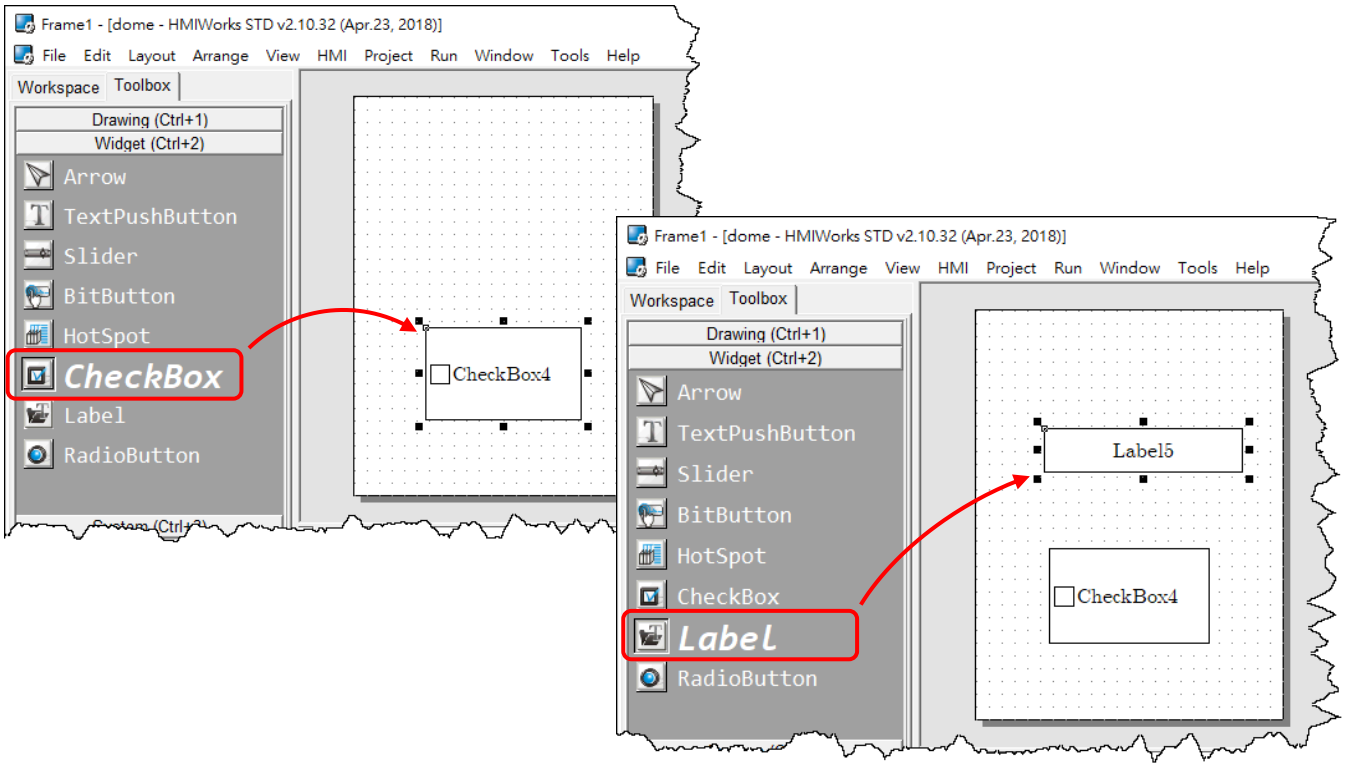
Step 1 Creating a new project

Click the "**New…**" option from the "**File**" menu and select the name of the TouchPAD model, specify the Project name, the Location, the Orientation, and the Programming Type. Here we choose **programming type** as **"[2] Ladder"**.

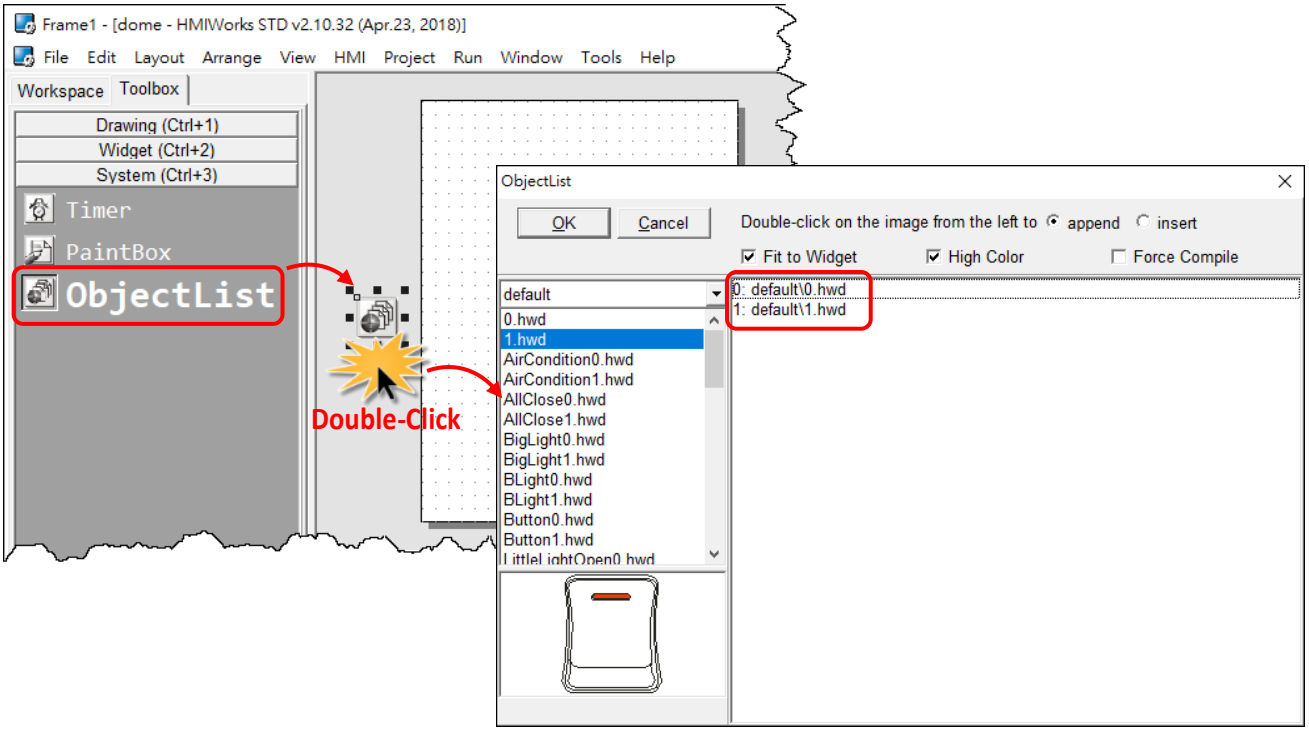


Step 2 Designing the Graphic User Interface

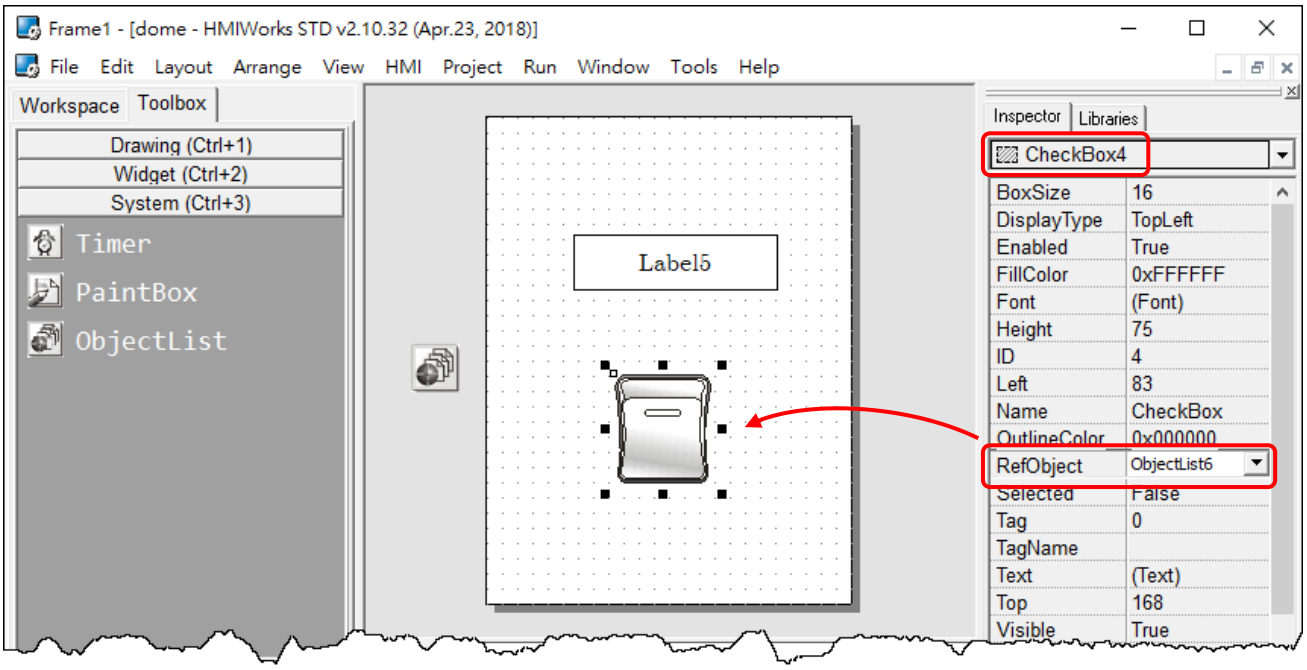
1. For example, place a **CheckBox** component and a **Label** component on the frame panel. Here, we plan to take the **CheckBox** component as an input and the **Label** component as an output.



2. Select an **ObjectList** component and click on the frame design area. Double click the **ObjectList** icon to open the "**ObjectList**" window. In the "**ObjectList**" window, double click to select the pictures you want. Users need to double click on two pictures, one is for the checked state of the **CheckBox** component and the other is for the unchecked state. Click the "**OK"** button to finish this step.

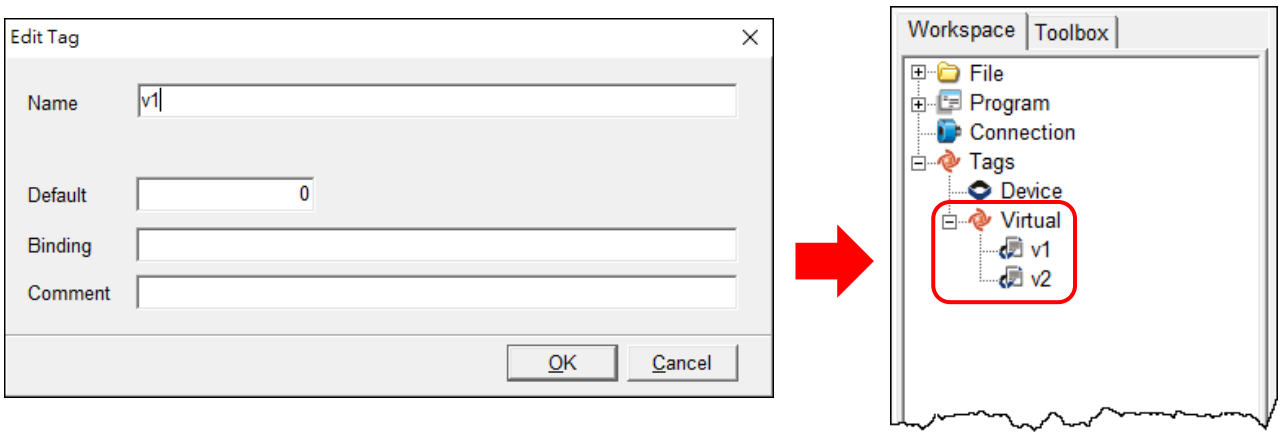


3. Make the **CheckBox** component refer to the **ObjectList** component by setting the property **"RefObject"** to the **ObjectList** component. Now toggling the states of the **CheckBox** component becomes the switching of the pictures in the **ObjectList** component.

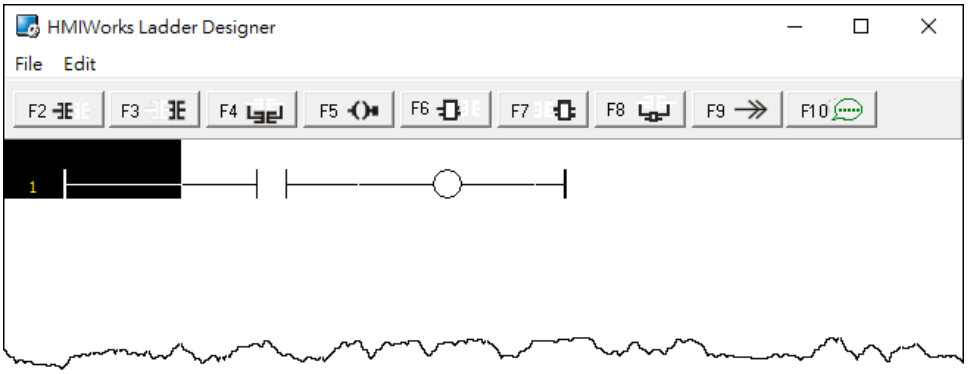


Step 3 Designing the Ladder Diagram

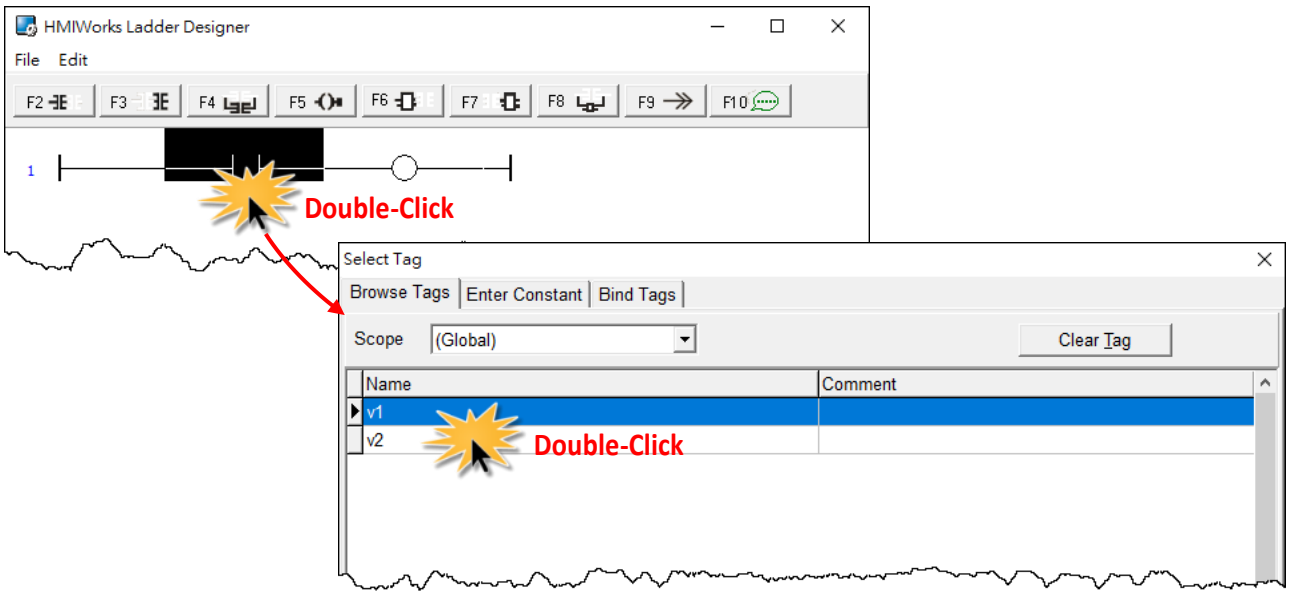
1. First, add virtual tags (variables) for the ladder diagram. Press **<F2>** key or click the "**New Virtual Tag F2**" option from the "**HMI**" menu. Here, we add two tags, v1 and v2, for example. After adding the tags, users can verify in the "**Workspace"** panel.



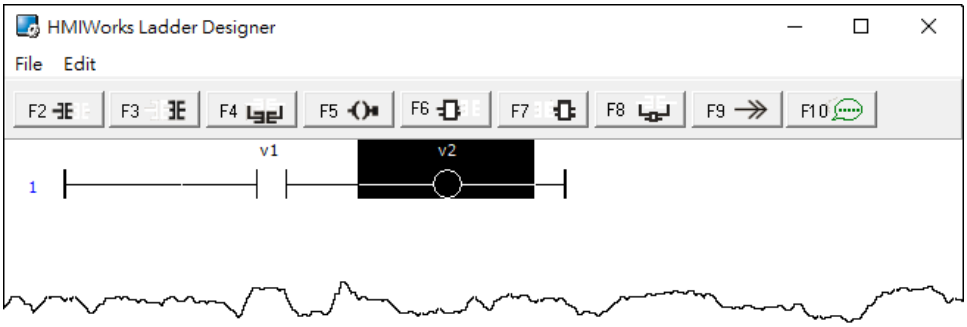
2. Press **<F4>** key or click the "**Ladder Designer F4**" option from the "**HMI**" menu to open the "**Ladder Designer"** window. In the **Ladder Designer** window, press **<F2>** key to create a new rung.



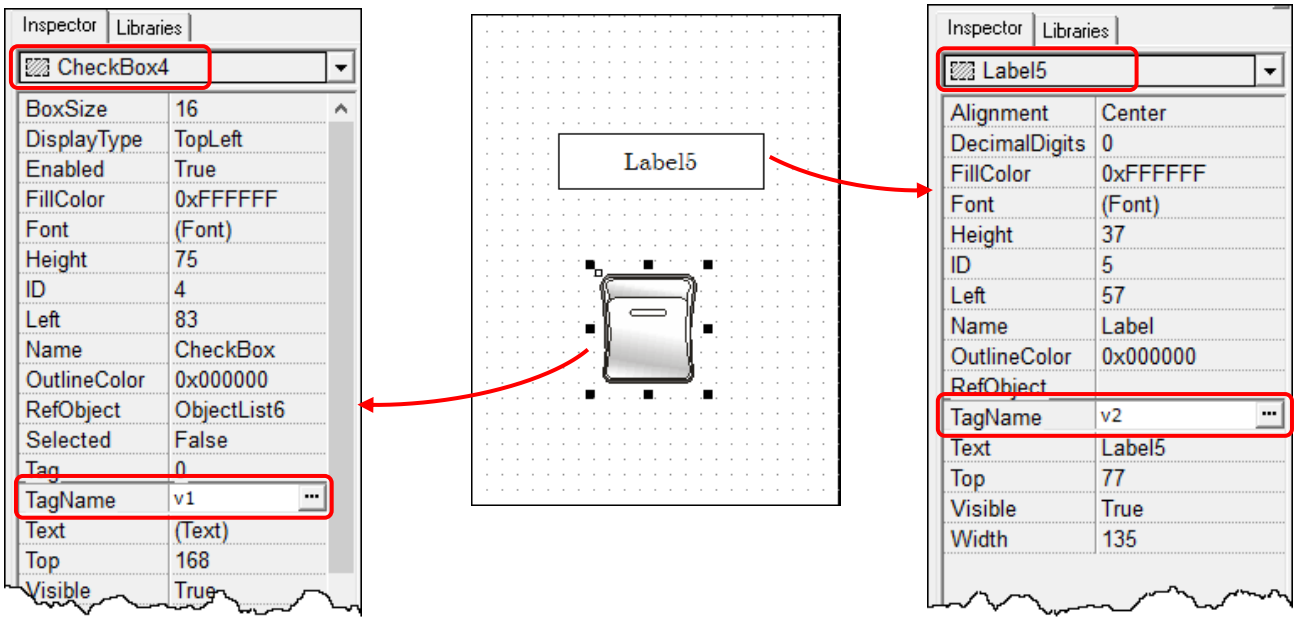
3. Double click the contact input of the first rung in the "**Ladder Designer"** window. Then the "**Select variable**" window is displayed. Choose the variable to associate with the contact input.



4. Here, we select variable v1 to associate the contact input. Repeat the same procedure to associate v2 with the coil output.



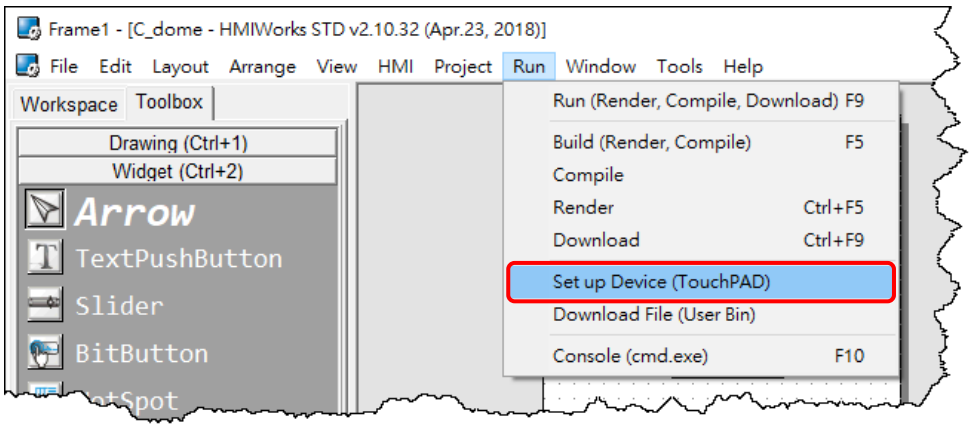
5. We associate the **CheckBox** component with the **v1** tag and the **Label** component with the **v2** tag by the **"TagName"** properties of themselves. After setting the **"TagName"** properties, users can verify in the "**Inspector"** panel.



Step 4 Setup Device

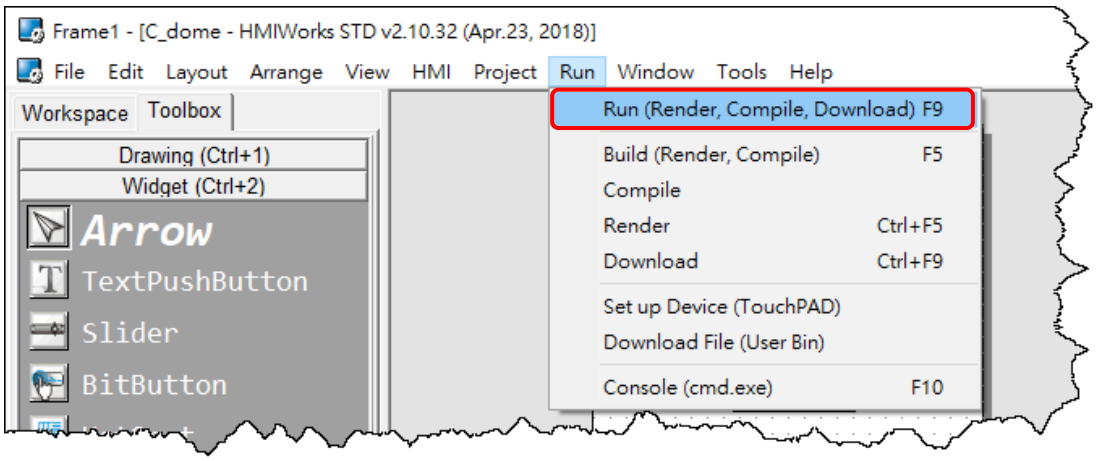
The downloading program method to the TouchPAD depends on the type of TouchPAD device, refer to the Section 3.4 Downloading Methods for TouchPAD of the TouchPAD Hardware User Manual for more detailed information.

In this example, we use the TPD-280-H device connect the Host PC via RS-485 wiring and turn the rotary switch to "Update Only" mode (position 1) then reboot TouchPAD device.
Click the **"Set up Device (TouchPAD)"** option from the **"Run"** menu to select correct COM Port.

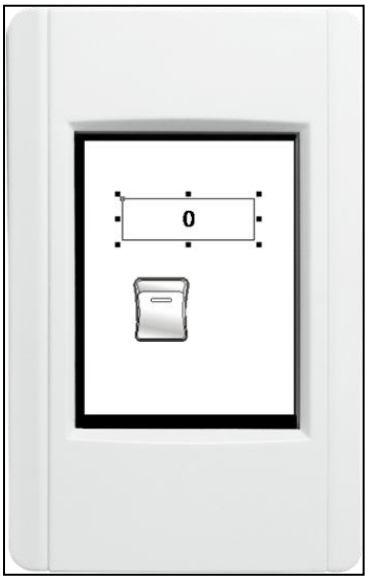


Step 5 Compiling and Downloading to Run

Click the **“Run (Render, Compile, Download) F9”** option from the **“Run”** menu, or press **<F9>** key. Once the download is complete, set the rotary switch to “Run Only” (position 0) and reboot TouchPAD device.



As shown in the figure below, pressing the button switches the value of the **Label** from 0 → 1, or 1 → 0.



4.3 Integrating TouchPAD with I/O Modules

This Section provides connection methods for three series of I/O modules, the PET-7000, the I-7000, and the M-7000 series for ICP DAS, which will be described in more detail below.

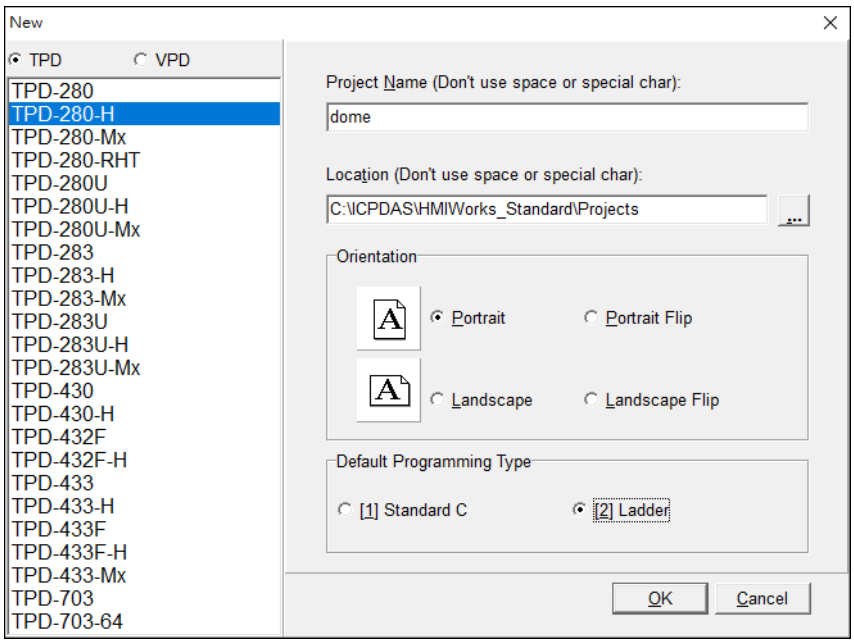
If your slave device is a third party Modbus RTU or TCP device, refer to the following FAQ for detailed instructions.
FAQ: How do I access a third-party Modbus RTU slave device by using TouchPAD?
FAQ: How do I access a third-party Modbus TCP slave device by using TouchPAD?

4.3.1 Access M-7000 by using TouchPAD

In this example, we use the TPD-280-H device to control an M-7060 module **(Modbus RTU I/O device)**, the 4-channel Digital Input and 4-channel Relay Output module of ICP DAS. First, put the M-7060 module in the same RS-485 network of the TPD-280-H device and configure the settings of the M-7060 module, including the Baud Rate, Data Bit, Parity, Stop Bit, Net ID, etc.

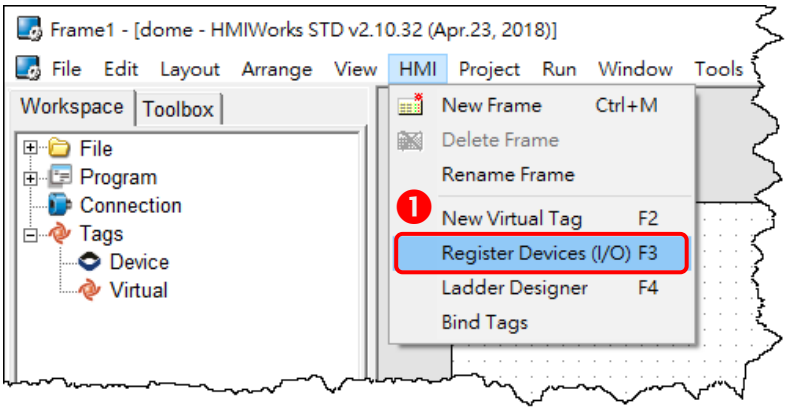
Step 1 Creating a new project

Click the "**New…**" option from the "**File**" menu and select the name of the TouchPAD model, specify the Project name, the Location, the Orientation, and the Programming Type.



Step 2 Configure the device (I/O) tags

1. Press **<F3>** key or click the "**Register Devices (I/O) F3**" option from the "**HMI**" menu to open the "**Devices**" window to register the M-7060 module.



2. Select **"Modbus RTU Master"** from the "TouchPAD is" drop down menu.

3. Select **"M-7000"** from the "Device Series" drop down menu.

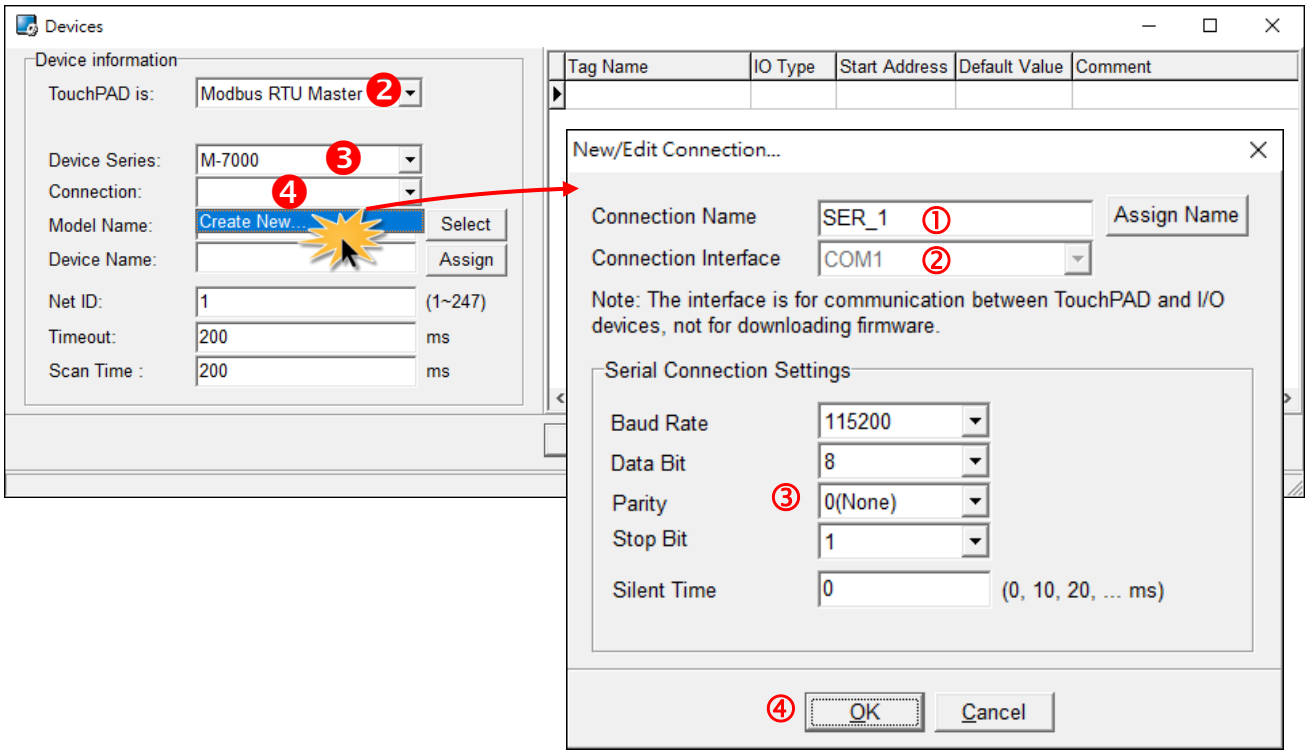
4. Select **"Create New…"** from the "Connection" drop down menu to open the "New/Edit Connection…"window, configure the connection information of the M-7060 module in the following manner:

① Enter a name for the connection (e.g., SER_1) in the "Connection Name" field.

② Select **"COM1"** from the "Connection Interface" drop down menu.

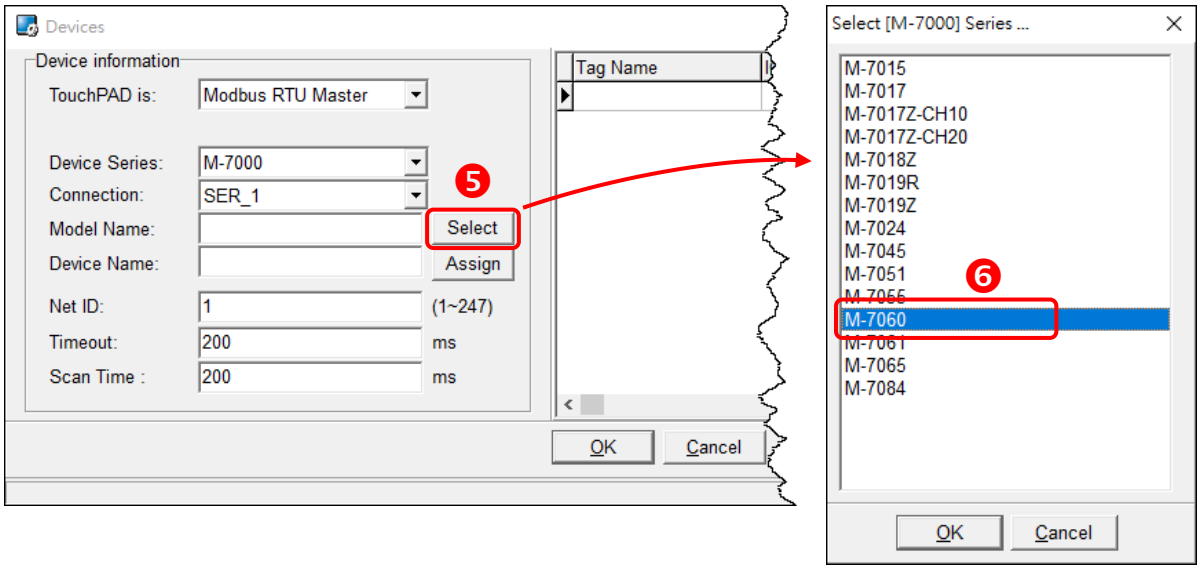
③ Select the **Baud Rate and Data Format of the M-7060** module in the "Baud Rate", "Data Bit", "Parity" and "Stop Bit" drop down menu.

④ Click the **"OK"** button to save the configuration.



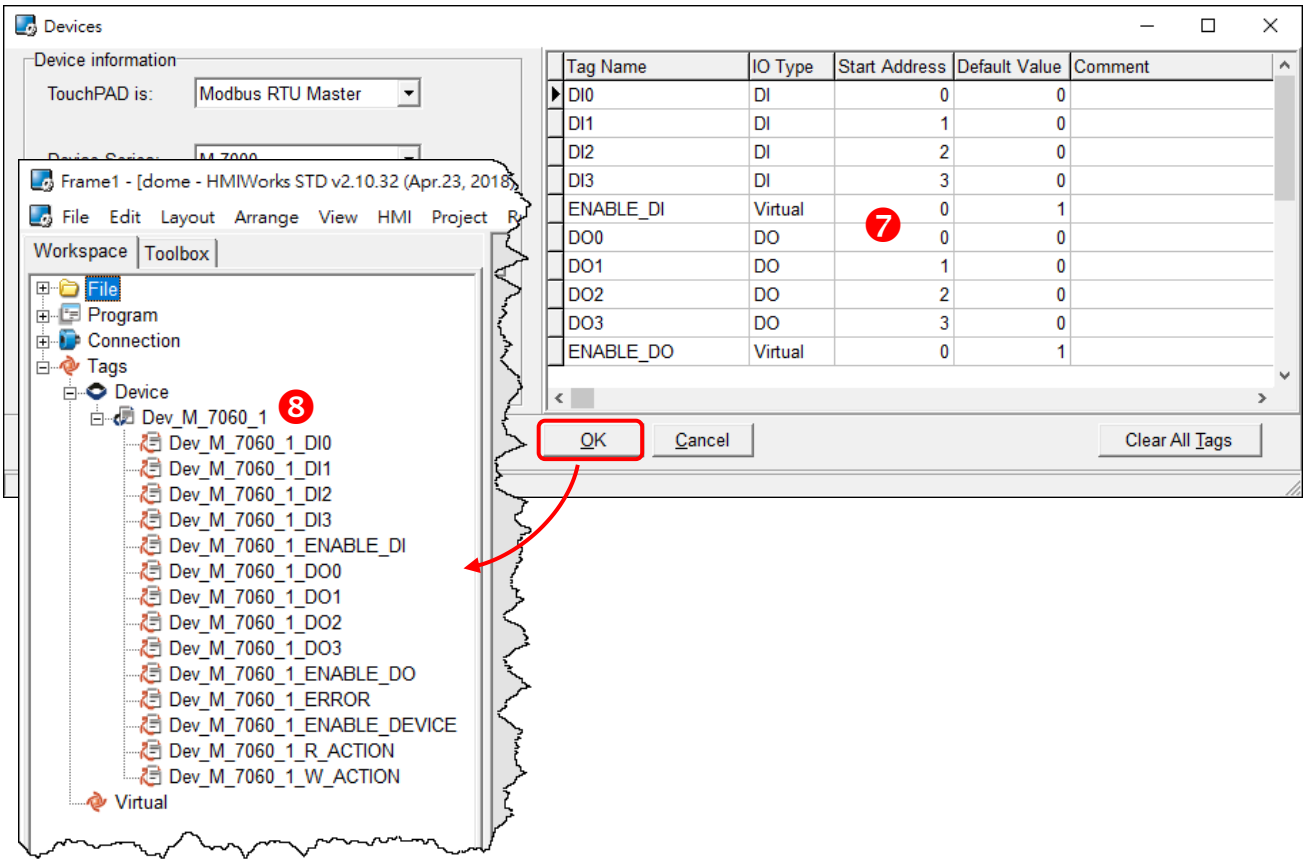
5. Click the **"Select"** button to open the "Select [M-7000] Series…" window.

6. In the **"Select [M-7000] Series…"** window, select the M-7060 module and click the **"OK"** button.



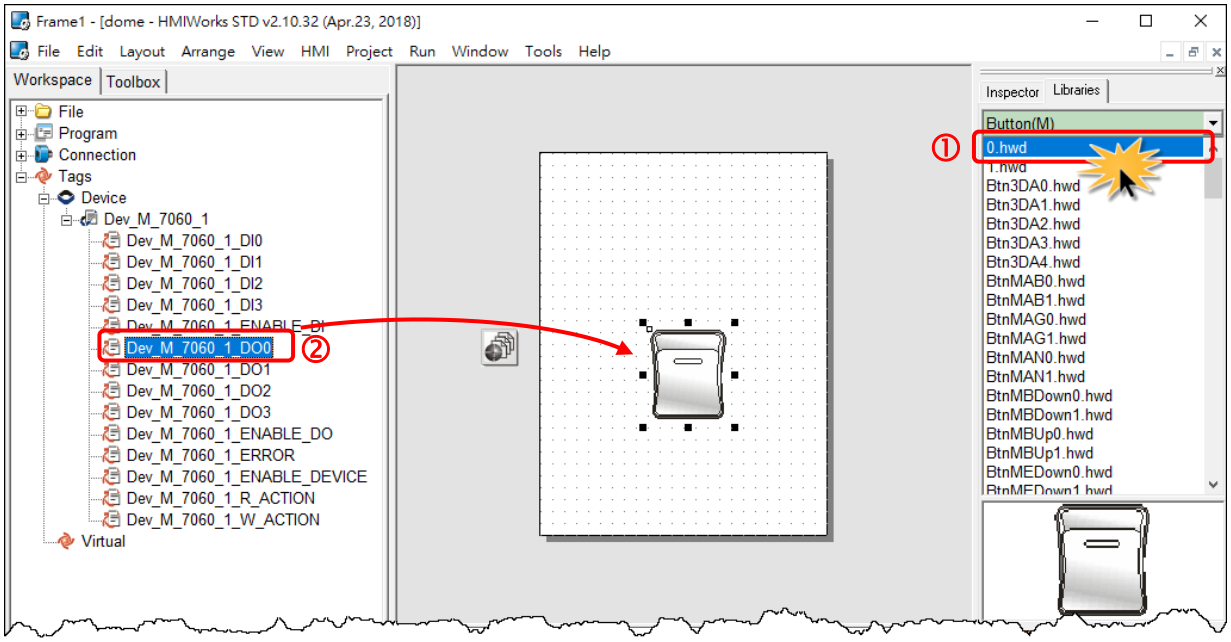
7. Verify that the **information for M-7060 module is correct** (e.g., the Device Name, Net ID, Tag Name, IO Type, Start Address and Default Value, etc.) and click the **"OK"** button.

8. In the **"Workspace"** panel, the creation of the **"Dev_M_7060_1"** device is now complete.



Step 3 Designing the Ladder Diagram

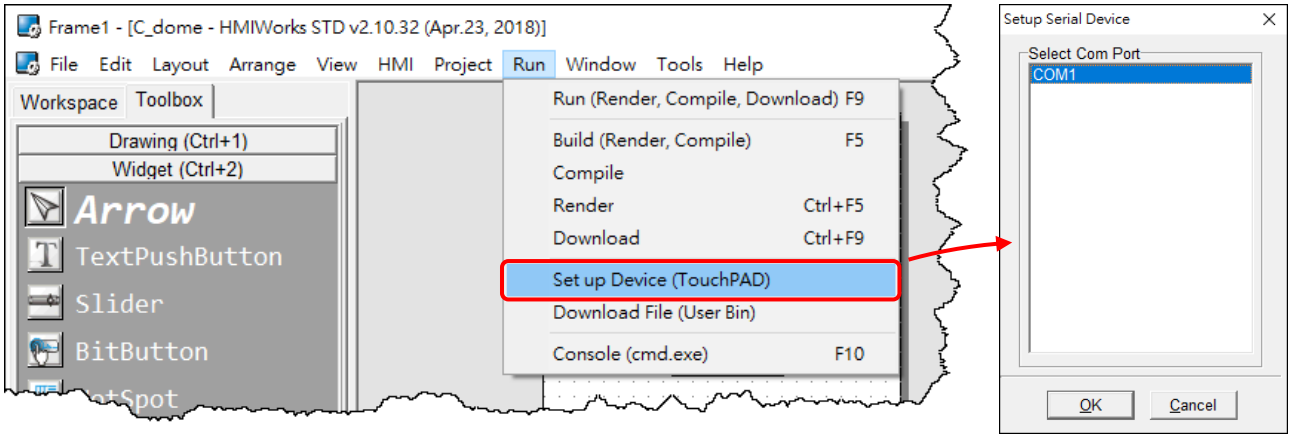
Click the "**Libraries**" tab to select a picture to represent the tag in the "**Libraries**" panel. Drag and drop the tag that is corresponding to the DO0 of M-7060. On the frame design area, the picture you just select is now on the frame.



Step 4 Setup Device

The setup device methods depends on the type of TouchPAD device and download methods, refer to the Section 3.4 Downloading Methods for TouchPAD of the TouchPAD Hardware User Manual for more detailed information.

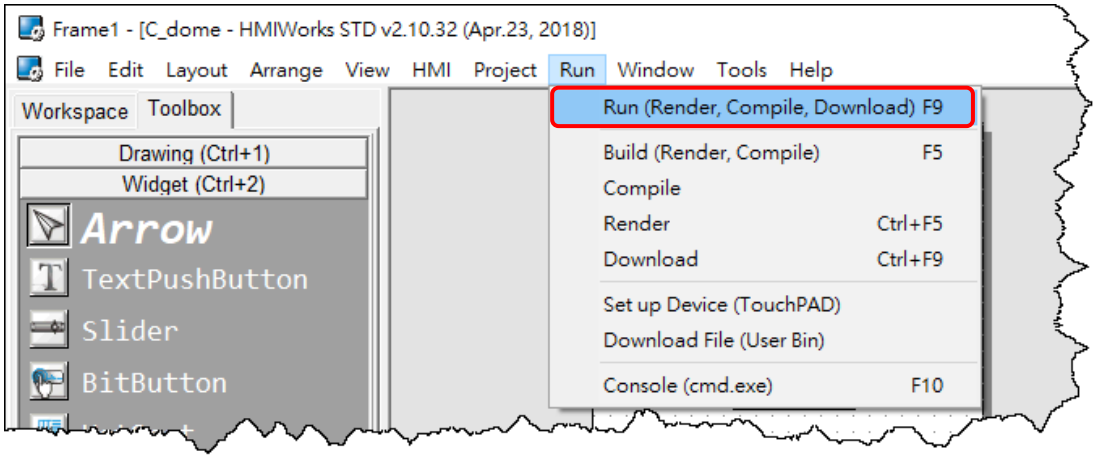
In this example, we use the TPD-280-H device to connect the Host PC via RS-485 wiring and turn the rotary switch to "Update Only" mode (position 1) then reboot TouchPAD device.
Click the **"Set up Device (TouchPAD)"** option from the **"Run"** menu to select correct COM Port.



Step 5 Compiling and Downloading to Run

The downloading program method to the TouchPAD depends on the type of TouchPAD device, refer to the Section 3.4 Downloading Methods for TouchPAD of the TouchPAD Hardware User Manual for more detailed information.

Click the **"Run (Render, Compile, Download) F9"** option from the **"Run"** menu, or press **<F9>** key. Once the download is complete, set the rotary switch to "Run Only" (position 0) and reboot TouchPAD device.



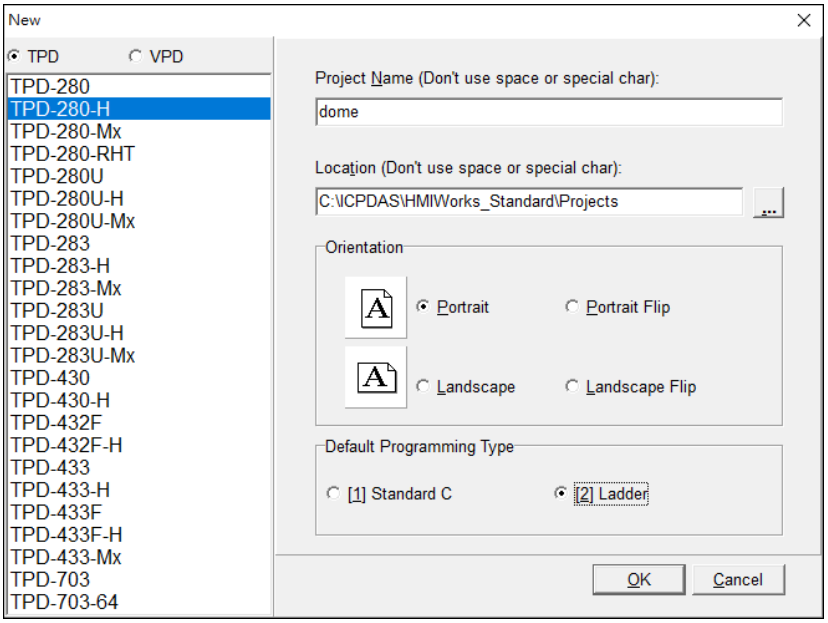
As shown in the figure below, pressing the button switches the output of channel 0 of the M-7060 module.

4.3.2 Access I-7000 by using TouchPAD

In this example, we use the TPD-280-H device to control an I-7066 module **(DCON I/O device)**, the 7-channel PhotoMOS Relay Output module of ICP DAS. First, put the I-7066 module in the same RS-485 network of the TPD-280-H device and configure the settings of the I-7066 module, including the Baud Rate, Data Bit, Parity, Stop Bit, Net ID, etc.

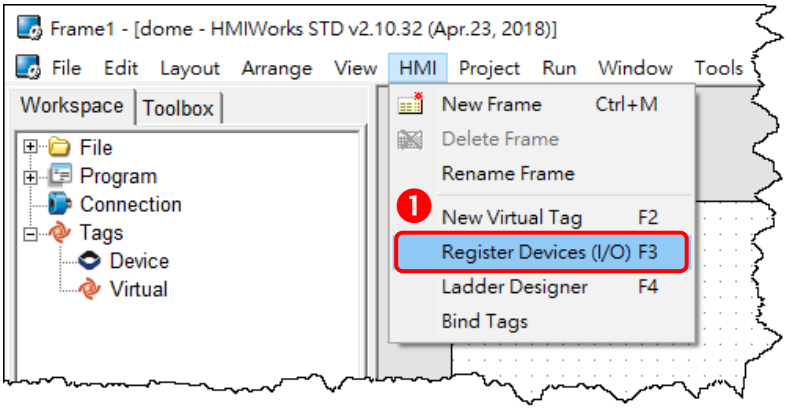
Step 1 Creating a new project

Click the "**New…**" option from the "**File**" menu and select the name of the TouchPAD model, specify the Project name, the Location, the Orientation, and the Programming Type.



Step 2 Configure the device (I/O) tags

1. Press **<F3>** key or click the "**Register Devices (I/O) F3**" option from the "**HMI**" menu to open the "**Devices**" window to register the I-7066 module.



2. Select **“DCON Master”** from the “TouchPAD is” drop down menu.

3. Select **“I-7000”** from the “Device Series” drop down menu.

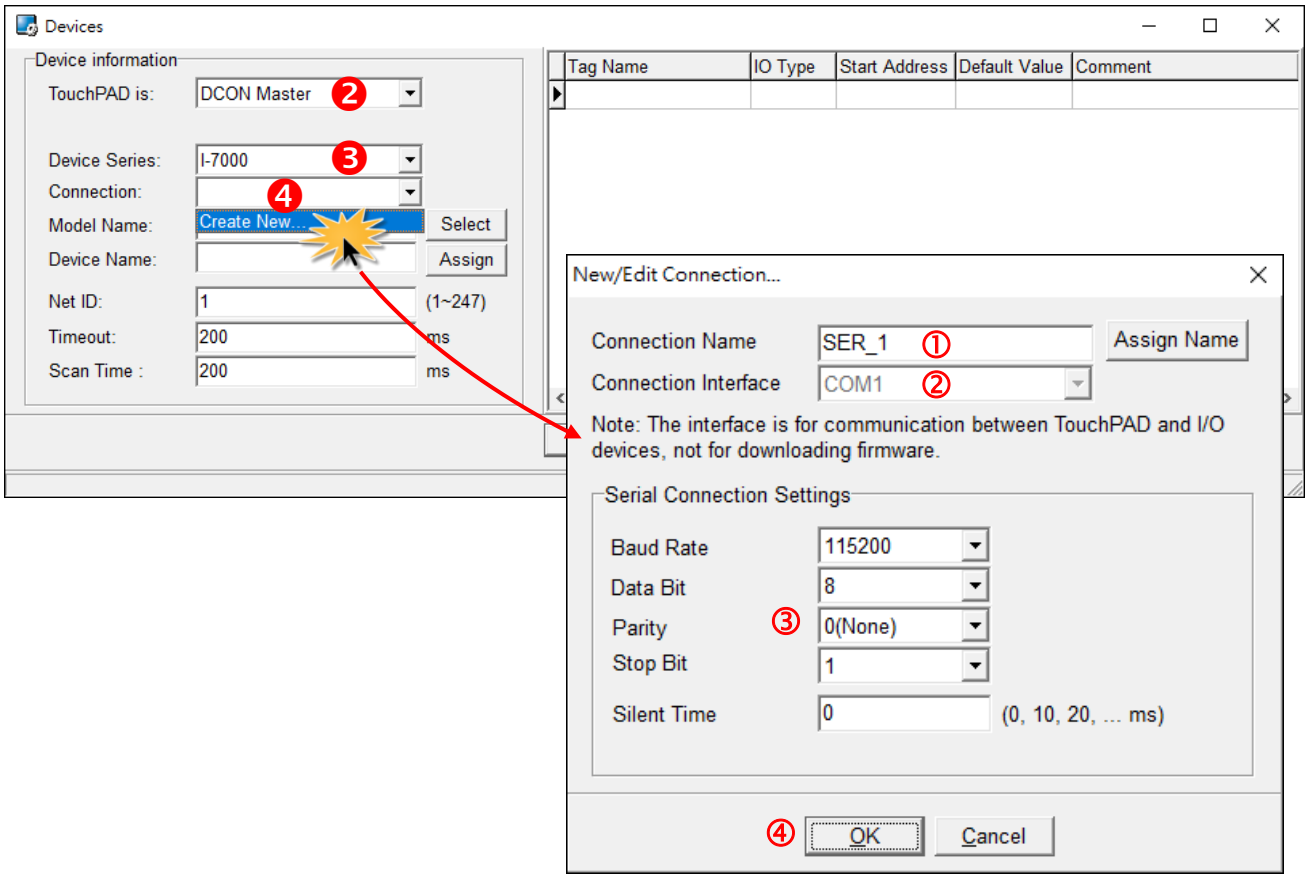
4. Select **“Create New…”** from the “Connection” drop down menu to open the “New/Edit Connection…”window, configure the connection information of the I-7066 module in the following manner:

① Enter a name for the connection (e.g., SER_1) in the “Connection Name” field.

② Select **“COM1”** from the “Connection Interface” drop down menu.

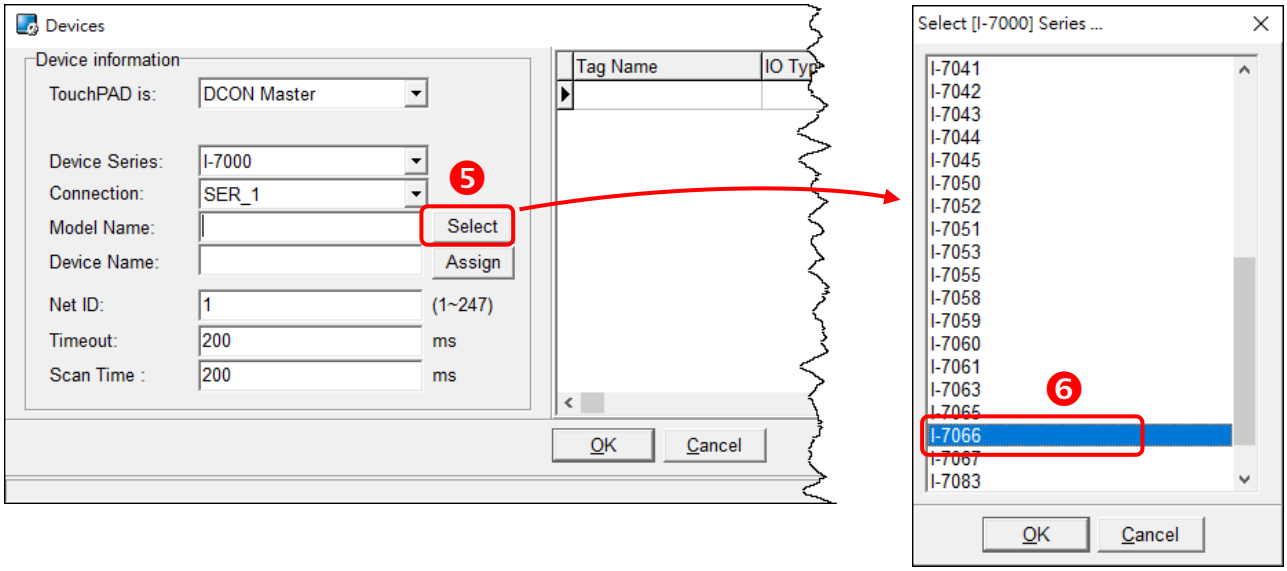
③ Select the **Baud Rate and Data Format of the I-7066** module in the “Baud Rate”, “Data Bit”, “Parity” and “Stop Bit” drop down menu.

④ Click the **“OK”** button to save the configuration.



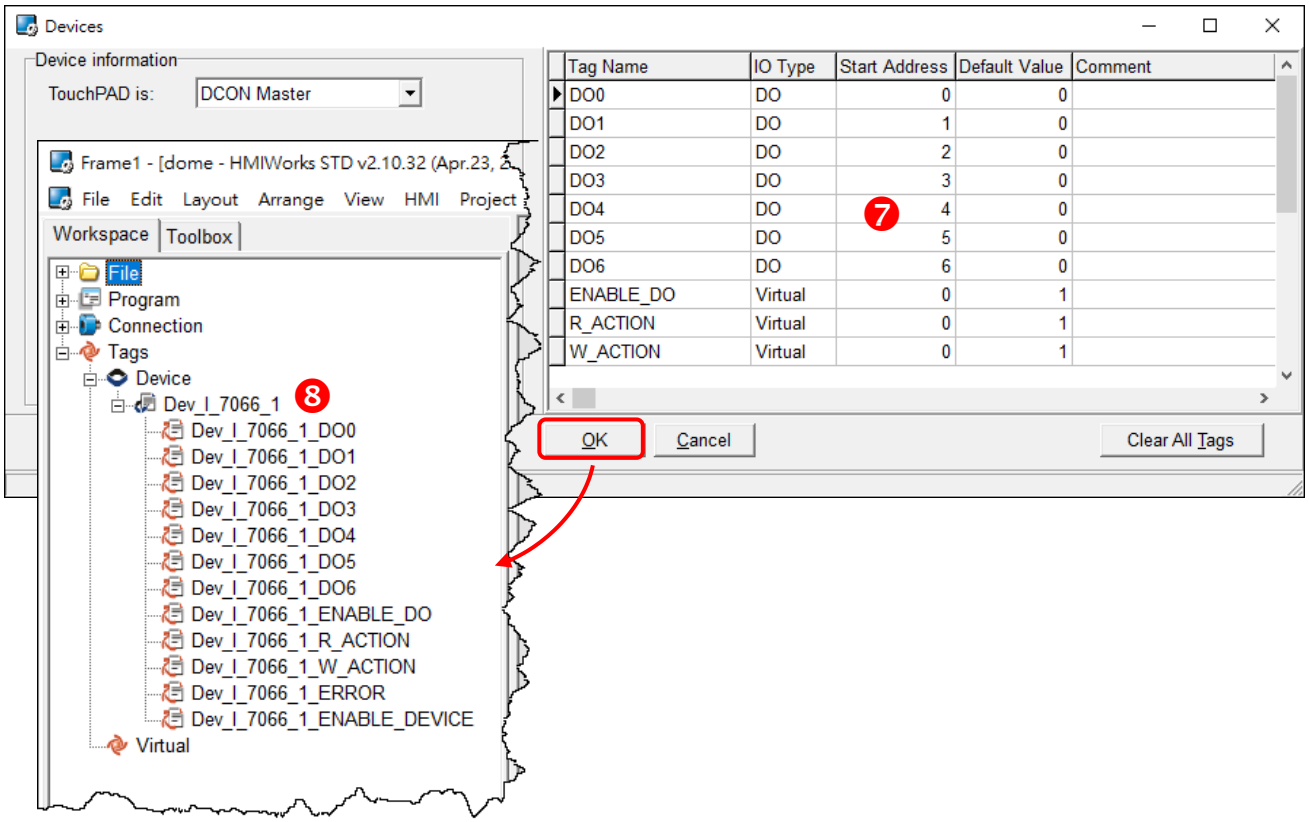
5. Click the **"Select"** button to open the "Select [I-7000] Series…" window.

6. In the **"Select [I-7000] Series…"** window, select the I-7066 module and click the **"OK"** button.



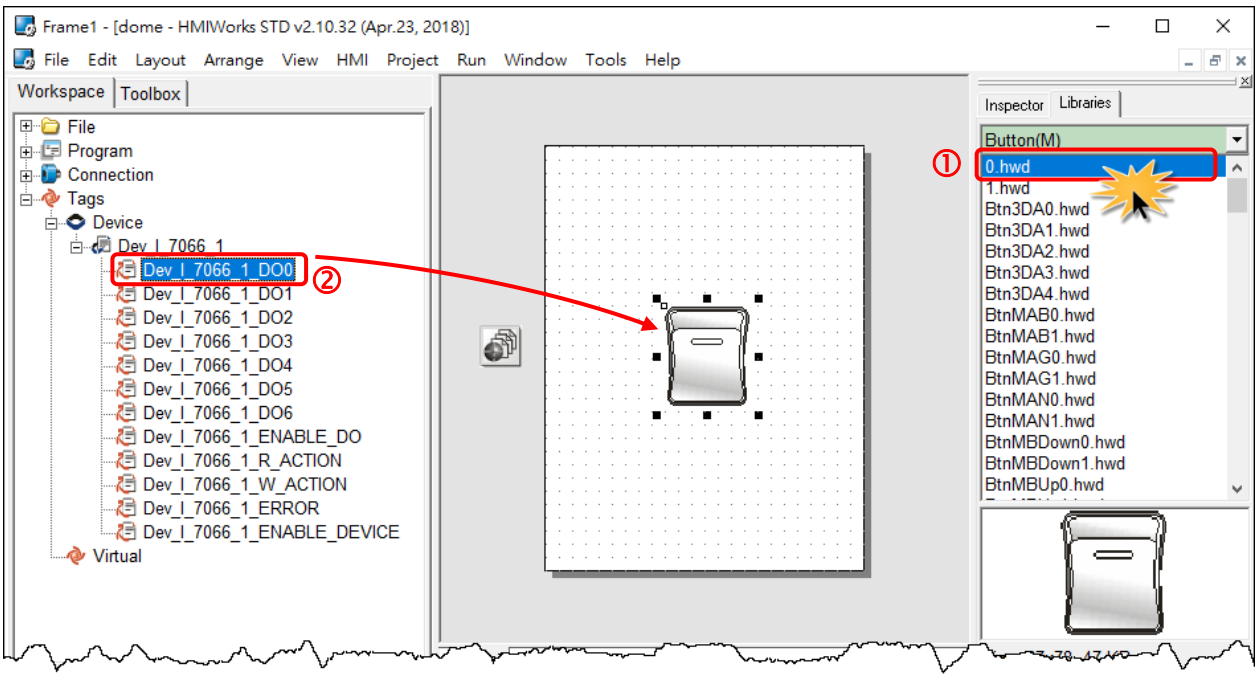
7. Verify that the **information for I-7066 module is correct** (e.g., the Device Name, Net ID, Tag Name, IO Type, Start Address and Default Value, etc.) and click the **"OK"** button.

8. In the **"Workspace"** panel, the creation of the **"Dev_I_7066_1"** device is now complete.



Step 3 Designing the Ladder Diagram

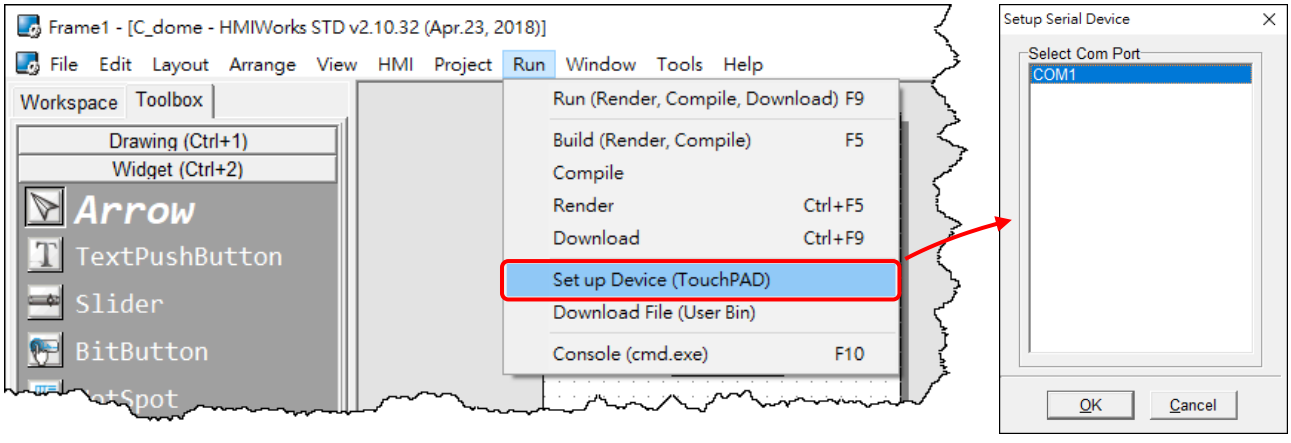
Click the "**Libraries**" tab to select a picture to represent the tag in the "**Libraries**" panel. Drag and drop the tag that is corresponding to the DO0 of I-7066. On the frame design area, the picture you just select is now on the frame.



Step 4 Setup Device

The setup device methods depends on the type of TouchPAD device and download methods, refer to the Section 3.4 Downloading Methods for TouchPAD of the TouchPAD Hardware User Manual for more detailed information.

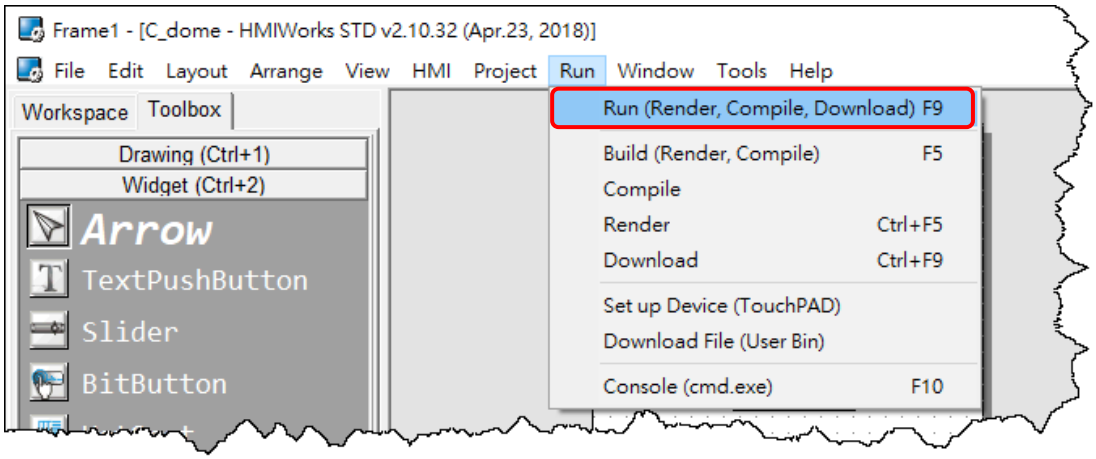
In this example, we use the TPD-280-H device to connect the Host PC via RS-485 wiring and turn the rotary switch to "Update Only" mode (position 1) then reboot TouchPAD device.
Click the **"Set up Device (TouchPAD)"** option from the **"Run"** menu to select correct COM Port.



Step 5 Compiling and Downloading to Run

The downloading program method to the TouchPAD depends on the type of TouchPAD device, refer to the Section 3.4 Downloading Methods for TouchPAD of the TouchPAD Hardware User Manual for more detailed information.

Click the **"Run (Render, Compile, Download) F9"** option from the **"Run"** menu, or press **<F9>** key. Once the download is complete, set the rotary switch to "Run Only" (position 0) and reboot TouchPAD device.



As shown in the figure below, pressing the button switches the output of channel 0 of the I-7066 module.

4.3.3 Access PET-7000 by using TouchPAD

In this example, we use the TPD-283-H device to control a PET-7060 module **(Modbus TPC I/O device)**, the 6-channel Digital Input and 6-channel Relay Output module of ICP DAS. First, connect the PET-7060 and TPD-283-H to the same hub or the same sub-network as the Host PC.

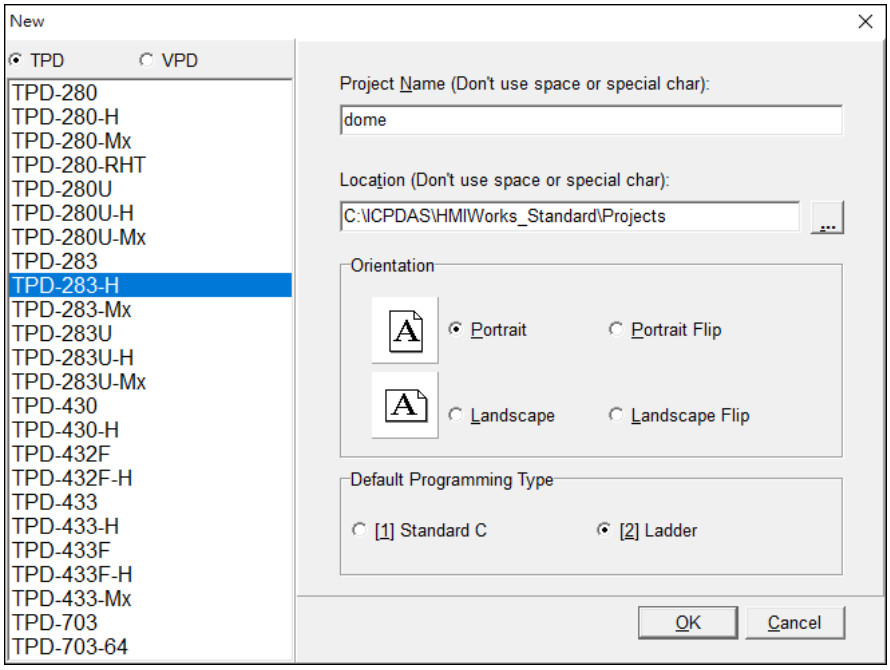
Step 1 Configuring the PET-7060

Ensure that the network settings on your PC are configured correctly and attach a power supply to the PET-7060. Configure the correct network settings for the PET-7060 module.

Refer to the PET-7060 Quick Start Guide for more detailed information.

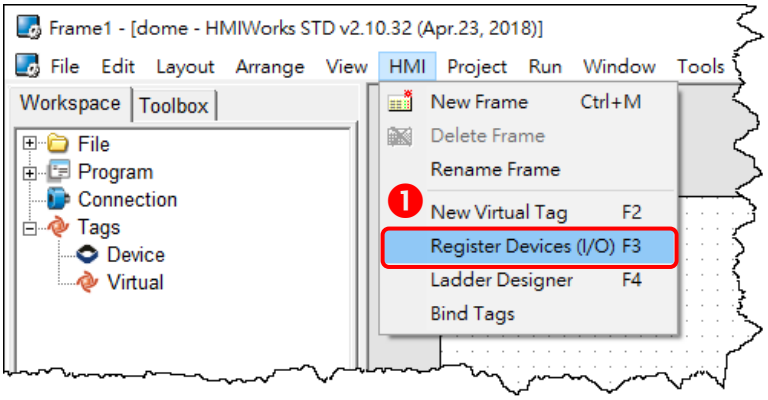
Step 2 Creating a new project

Click the "**New…**" option from the "**File**" menu and select the name of the TouchPAD model, specify the Project name, the Location, the Orientation, and the Programming Type.



Step 3 Configure the device (I/O) tags

1. Press **<F3>** key or click the "**Register Devices (I/O) F3**" option from the "**HMI**" menu to open the "**Devices**" window to register the PET-7060 module.



2. Select **"Modbus TCP Master"** from the "TouchPAD is" drop down menu.

3. Select **"PET-7000"** from the "Device Series" drop down menu.

4. Select **"Create New…"** from the "Connection" drop down menu to open the "New/Edit Connection…"window, configure the connection information of the PET-7060 in the following manner:

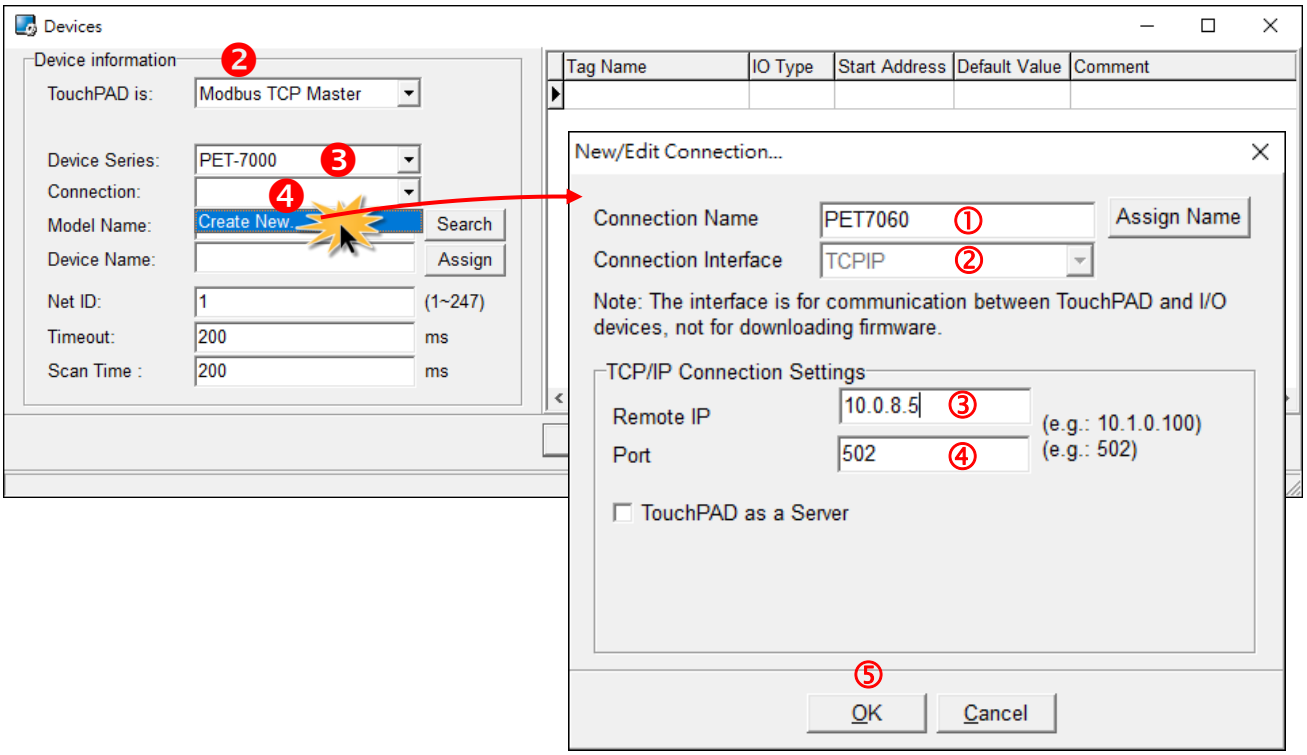
① Enter a name for the connection (e.g., PET7060) in the "Connection Name" field.

② Select **"TCPIP"** from the "Connection Interface" drop down menu.

③ Enter the **IP Address of the PET-7060** module in the "IP Address" field.

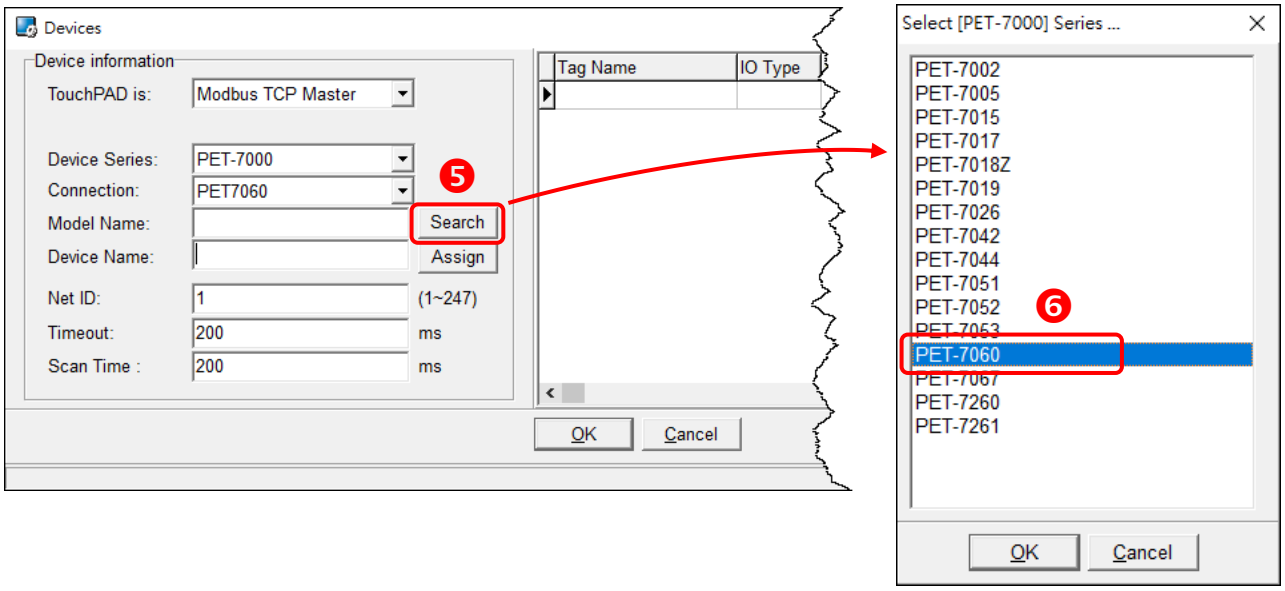
④ Enter the **TCP Port of the PET-7060** module in the "Port" field.

⑤ Click the **"OK"** button to save the configuration.



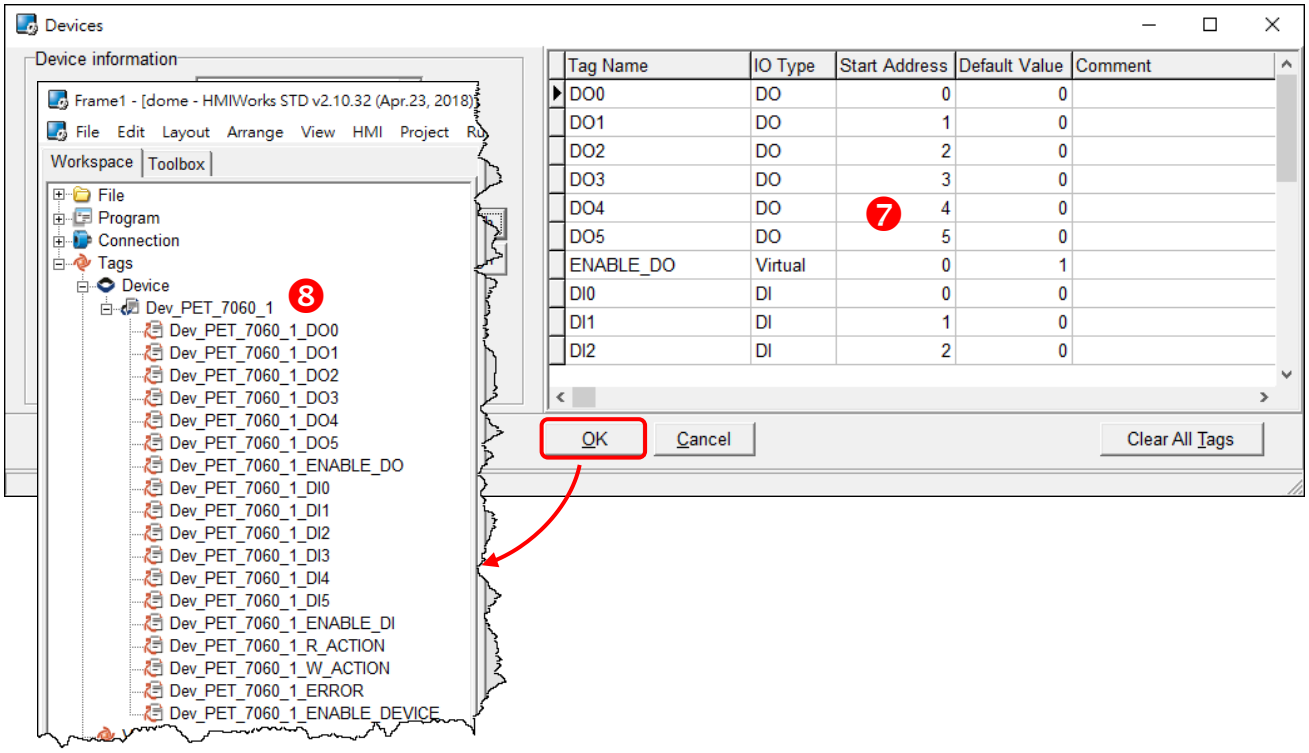
5. Click the **"Select"** button to open the "Select [PET-7000] Series…" window.

6. In the **"Select [PET-7000] Series…"** window, select the PET-7060 module and click the **"OK"** button.



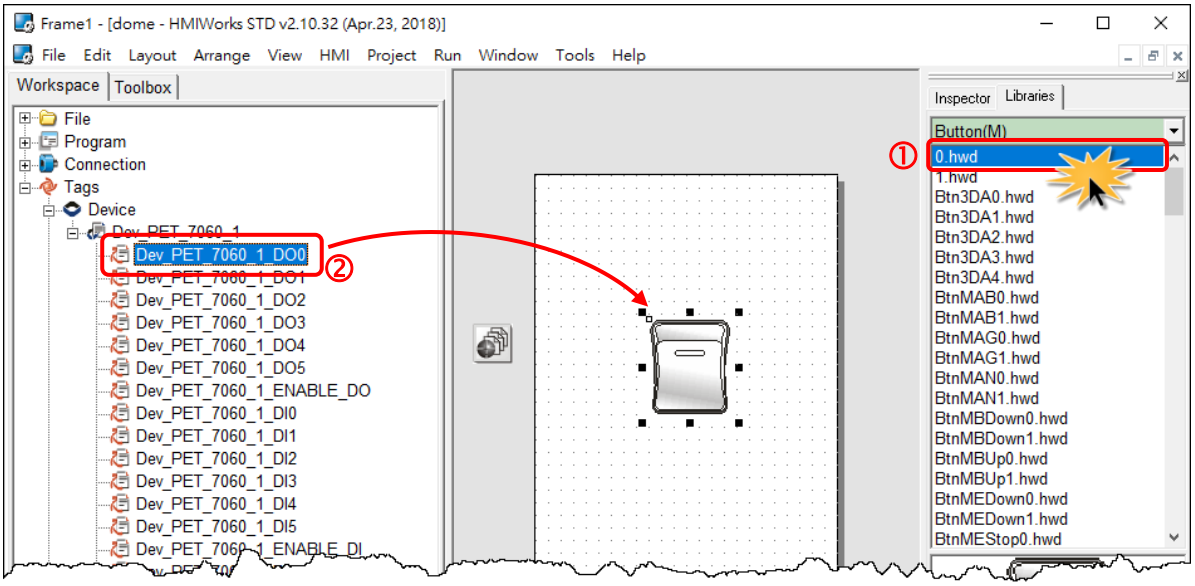
7. Verify that the **information for PET-7060 module is correct** (e.g., the Device Name, Net ID, Tag Name, IO Type, Start Address and Default Value, etc.) and click the **"OK"** button.

8. In the **"Workspace"** panel, the creation of the **"Dev_PET_7060_1"** device is now complete.



Step 4 Designing the Ladder Diagram

Click the "**Libraries**" tab to select a picture to represent the tag in the "**Libraries**" panel. Drag and drop the tag that is corresponding to the DO0 of PET-7060. On the frame design area, the picture you just select is now on the frame.

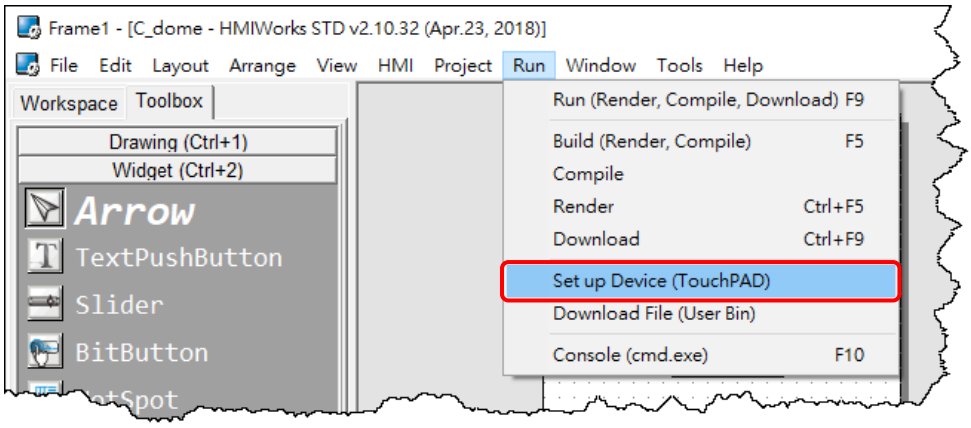


Step 5 Setup Device

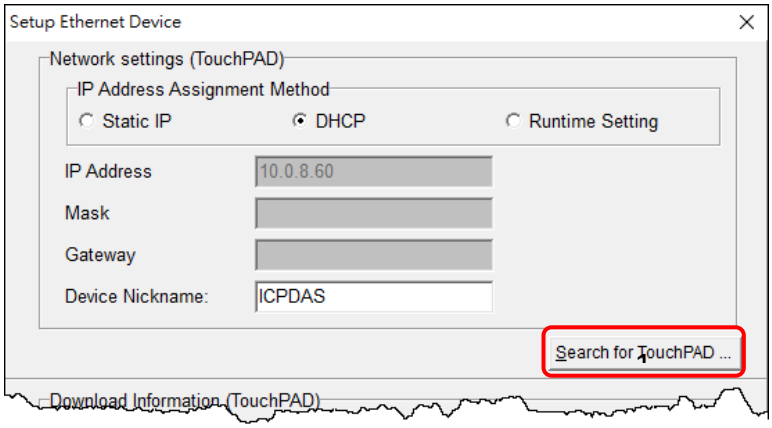
The setup device methods depends on the type of TouchPAD device and download methods, refer to the Section 3.4 Downloading Methods for TouchPAD of the TouchPAD Hardware User Manual for more detailed information.

In this example, we use the TPD-283-H device to connect the same hub or the same sub-network as the Host PC via Ethernet cable, and turn the rotary switch to "Run & Update mode" mode (position 0) then reboot TouchPAD device.

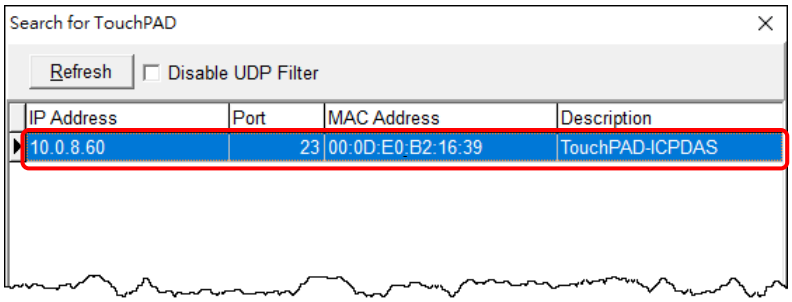
1. Click the **"Set up Device (TouchPAD)"** option from the **"Run"** menu to assign correct runtime IP address and download information.



2. In the "Setup Ethernet Device" window, click the **"Search for TouchPAD…"** button to open "Search for TouchPAD" window.

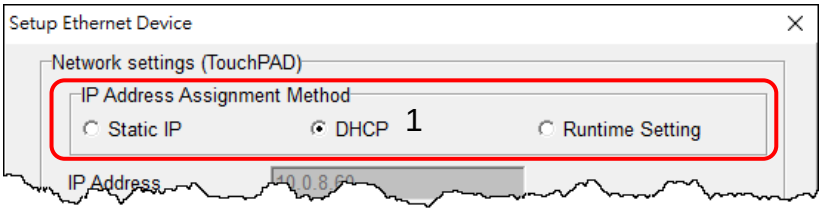


3. If the TouchPAD device is found and displayed in the list on the "Search for TouchPAD" window, **select the TouchPAD item depending on MAC Address of TPD-283-H** and click the **"OK"** button.



Note: You can also find the MAC address on the back of the TouchPAD.

4. In the "Setup Ethernet Device" window, select the **"DHCP", "Static IP" or "Runtime Setting"** (e.g., DHCP) in the "IP Address Assignment Method" field. This setting is used for TouchPAD runtime.

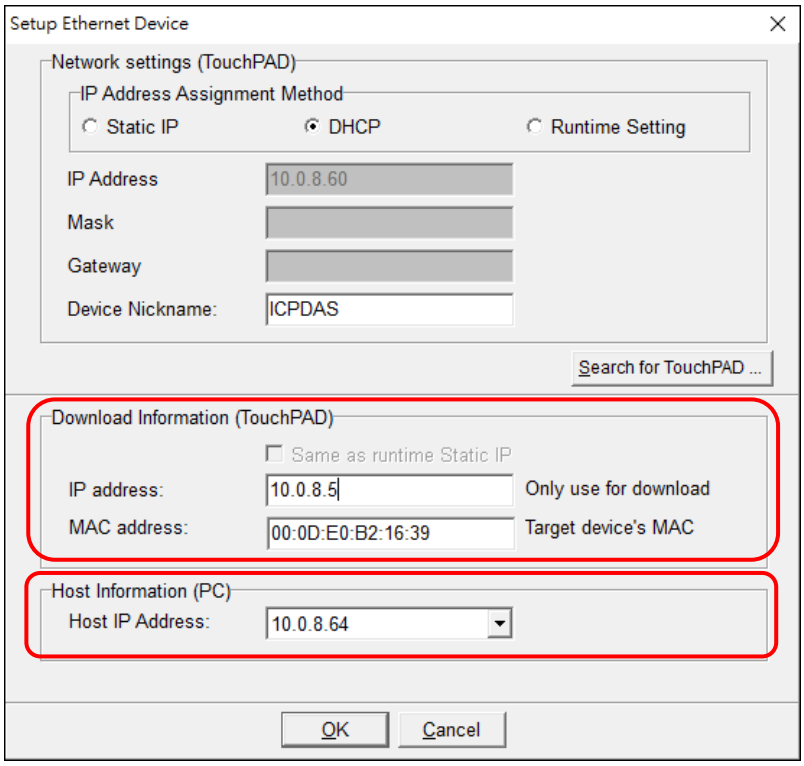


Option	Descriptions
Static IP	The IP address of the TouchPAD is configured in HMIWorks, and it is stored as a part of the program image.
DHCP	The IP address of the TouchPAD is dynamically allocated from a DHCP server. Please ensure that there is a DHCP server in the environment.
Runtime Setting	TouchPAD loads the IP information from the flash at the runtime. Before IP settings are used, be sure to set the IP settings into the flash by the related API functions. We have demo to do this as well.

Note: Downloading new program image into TouchPAD is required for changing the operation mode between Static IP, DHCP and Runtime Setting, or changing the IP address of the Static IP settings.

5. Verify that the **“IP Address”** of the download information is in the same subnet of the **“Host IP Address”**. This setting is used for downloading application only.

6. Verify that **“TouchPAD MAC Address”** must match the MAC Address of your TouchPAD device, and click the **“OK”** button.



Notes:

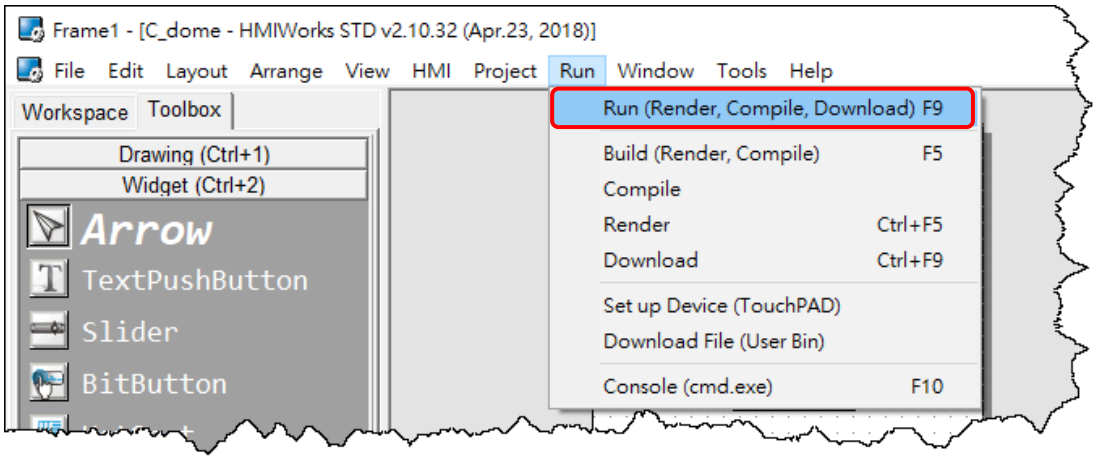
1. You can find the MAC address on the back of the TouchPAD.

2. It's normal that the searched IP address of the TouchPAD is 0.0.0.0 when force update. You just need to assign a valid IP address in the Download Information, and the TouchPAD can then be updated via the new specified IP address.

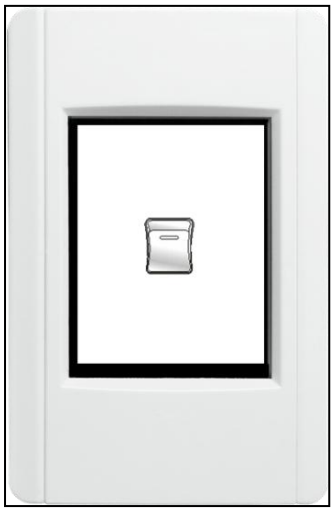
Step 6 Compiling and Downloading to Run

The downloading program method to the TouchPAD depends on the type of TouchPAD device, refer to the Section 3.4 Downloading Methods for TouchPAD of the TouchPAD Hardware User Manual for more detailed information.

Click the **"Run (Render, Compile, Download) F9"** option from the **"Run"** menu, or press **<F9>** key.



Once the upload is complete, as shown in the figure below, pressing the button switches the output of channel 0 of the PET-7060 module.



4.4 TCP/IP Communication

TouchPAD users can develop custom applications for TCP/IP communication. Refer to below example for more information about creating a TCP client or server with TouchPAD.

4.4.1 How to use TouchPAD as TCP Client?

In this example, we use PC as TCP server to receive data from TouchPAD (TCP Client), which will be described in more detail below.

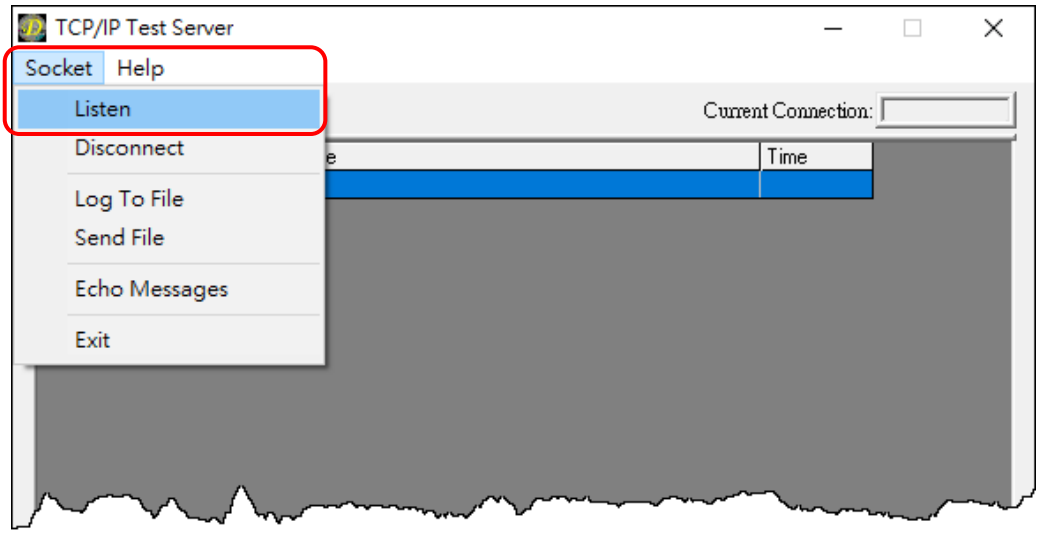
Step 1 Configure your computer to be a Server

⚠ **Note:** Ensure that the Windows firewall or any Anti-Virus firewall software is correctly configured or temporarily disable these functions; otherwise the TcpipEcho.exe may not work as required.

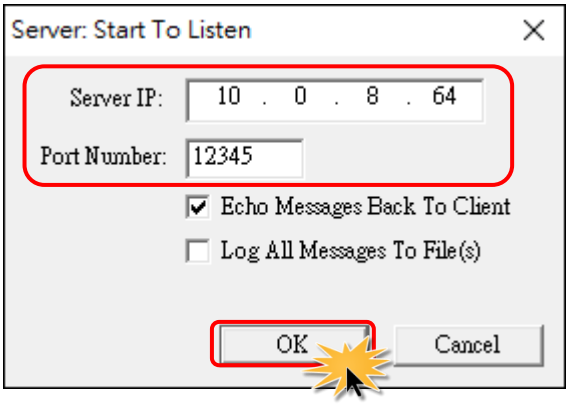
1. Install **TcpipEcho.exe (TCP/IP Test Server Program)** on your PC. The location of the download addresses are shown below:

   http://www.brothersoft.com/tcp-ip-test-server-27898.html

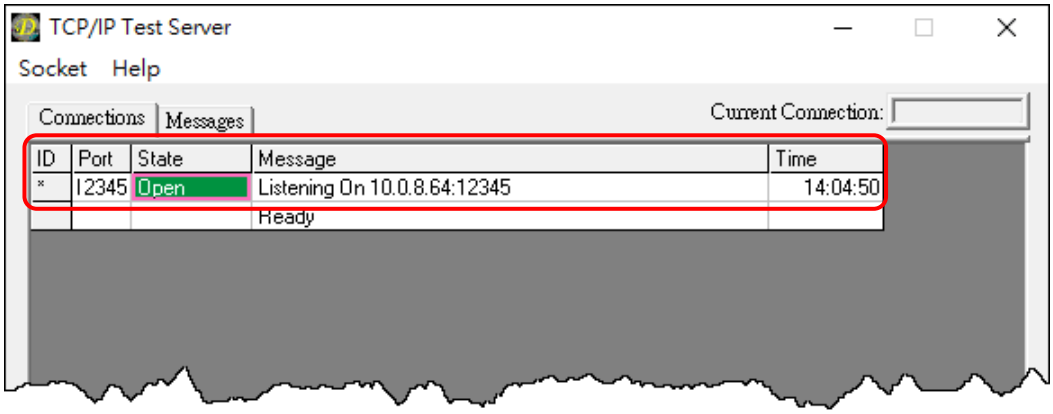
2. Launch the **TCPIPEcho.exe** program. Click the **"Listen"** option from the **"Socket"** menu to open the "Server: Start To Listen" dialog.



3. Type the **IP address** and **Port number** of the TCP Server (e.g., PC) in the "**Server IP**" and "**Port Number**" field (e.g., "**10.0.8.64" and "12345**").
4. Click the **"Echo Messages Back To Client"** check box.
5. After clicking the **"OK"** button, the server will begin listening on the specific IP/Port.



6. This will be indicated an **"Open"** line in the TCP/IP Test Server dialog box.



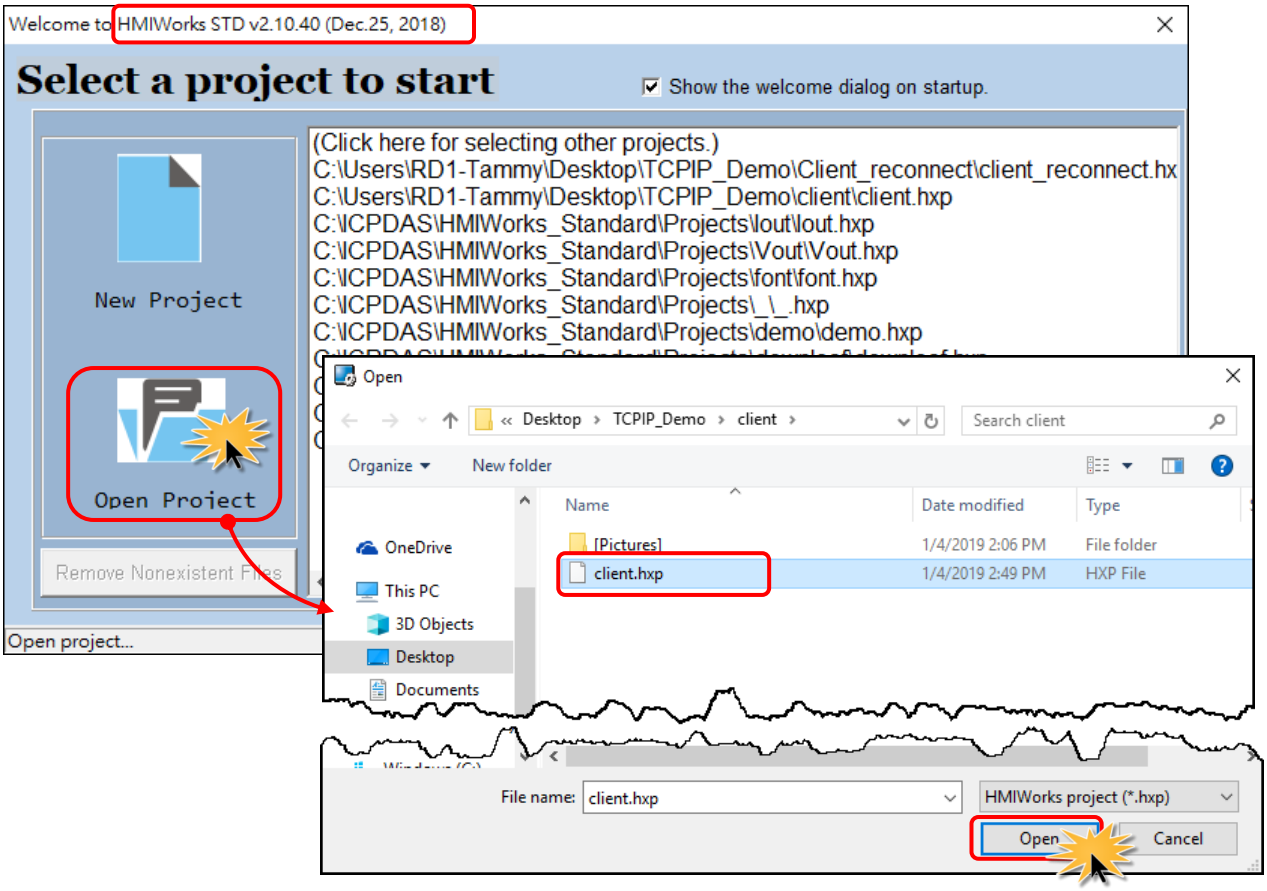
Step 2 Configure TouchPAD to be a Client

1. Download and unzip the TCP/IP demo.
   Download the file from the ICP DAS web site. The location of the download address is shown below:

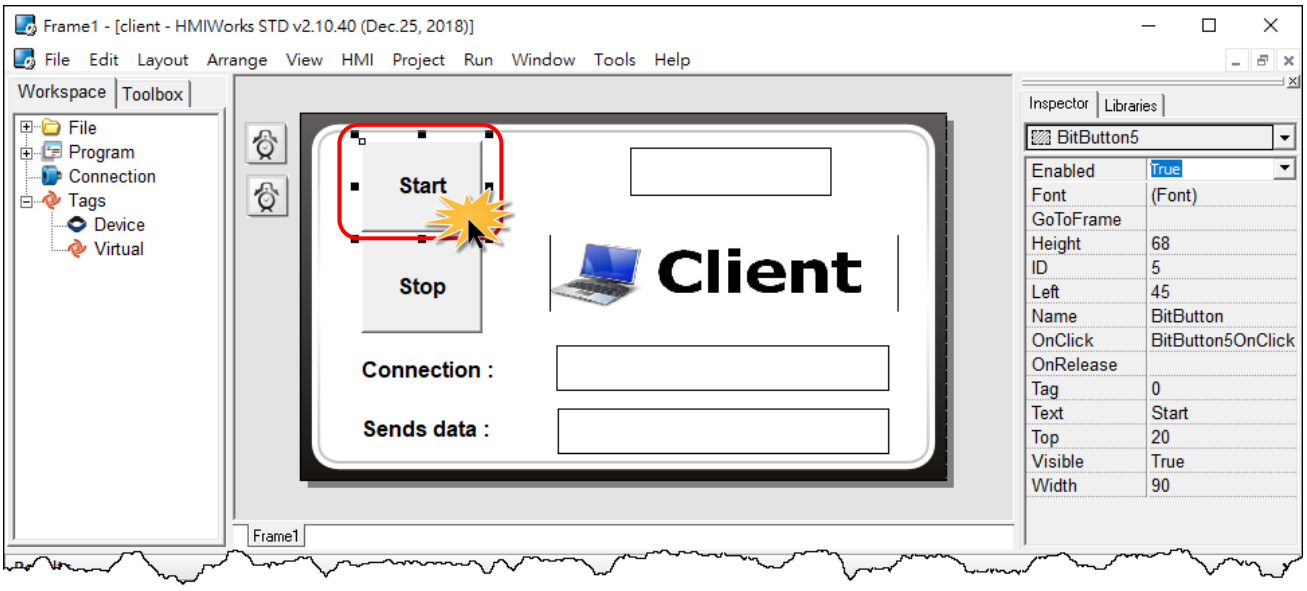
https://www.icpdas.com/en/download/show.php?num=1000

2. Launch the HMIWorks Standard software and open an existing **"client.hxp"** project.

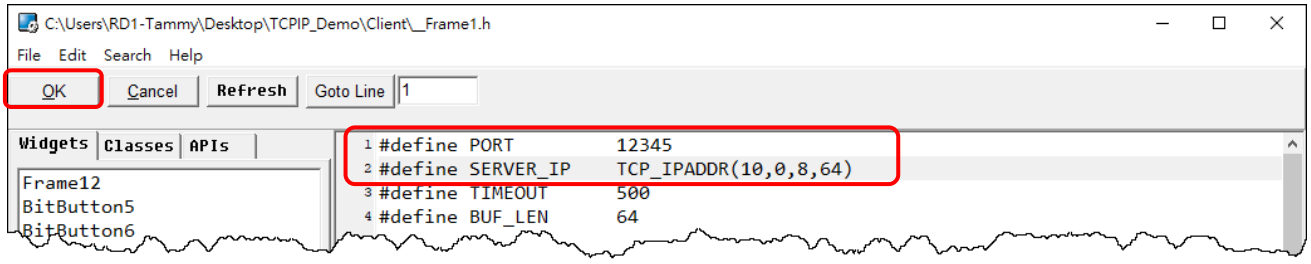
Note: Check that your HMIWorks version is v2.10.40 or later. If your HMIWorks version is earlier than v2.10.40, the HMIWorks must be updated to the latest version.



3. Double-click the **"Start"** BitButton component to implement its OnClick event handler in the displayed programming window.



4. Assign an **IP address and TCP Port** of the TCP Server (e.g., PC) in the define PORT and SERVER_IP lines (e.g., "**10.0.8.64" and "12345**") and click the **"OK"** button to save the file and leave.



The following code example establishes a TCP client connection, the process involves opening the connection, e.g., **hmi_TCPNew()** and **hmi_TCPOpen()**, sending and receiving the data, e.g., **hmi_TCPSendCmdEx()**, closing the connection, e.g., **hmi_TCPClose()**. Refer to the [<< HMIWorks API Reference>>] for details about TCP API.

```
#define PORT            12345
#define SERVER_IP       TCP_IPADDR(10,0,8,64)
#define TIMEOUT         500
#define BUF_LEN         64

static tHandle h = INVALID_HANDLE;    // handle for TCP Communications
int index = 0;                  // timer execution count
int missingCount = 0;       // the missing count (receiving error count)

void BitButton5OnClick(tWidget *pWidget) // Start
{
    if ( h > INVALID_HANDLE ) return; // already have a connection

    // Allocate a session. Check if h < 0 to prevent using another tHandle
    h = hmi_TCPNew();           //Allocate a TCP session if possible

    if (h > INVALID_HANDLE)    // if allocating a new session successfully
    {
        LabelTextSet(&Label4, "Connecting");
        //used in a client to establish a TCP session for connecting to a server.
        hmi_TCPOpen(h, SERVER_IP, PORT, PORT);
    }
    else
        h = INVALID_HANDLE; // don't keep error code in h
}

void BitButton6OnClick(tWidget *pWidget)      // Stop
{
    if (h > INVALID_HANDLE)
    {
        hmi_TCPClose(h);    //closes and deallocates a TCP session.
        h = INVALID_HANDLE;
        LabelTextSet(&Label4, "OFF");
    }
}
```



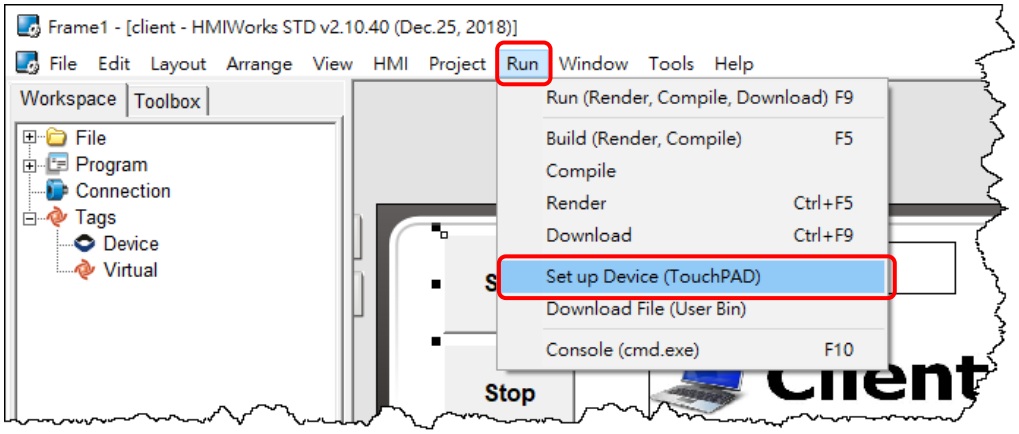
```
void Timer10OnExecute(tWidget *pWidget)                    //Connection status
{
    if (hmi_TCPState(h) == STATE_TCP_CONNECTED)      //gets the state of the TCP session.
    {
        LabelTextSet(&Label9, "CONNECTED");
        LabelTextSet(&Label4, "ON");
    }
    else
    {
        LabelTextSet(&Label9, "DISCONNECTED");
    }
}

void Timer11OnExecute(tWidget *pWidget)              //Send data
{
    static unsigned char send_buf[BUF_LEN];
    static unsigned char recv_buf[BUF_LEN];

    if ( h == INVALID_HANDLE ) return; // not ready yet

    index++;

    if (hmi_TCPState(h) == STATE_TCP_CONNECTED)
    {
        usprintf((char*)send_buf, "DATA%05d", index);
        //sends data and then receives data through a TCP session.
        hmi_TCPSendCmdEx(h, send_buf, BUF_LEN, recv_buf, BUF_LEN, TIMEOUT);
        recv_buf[BUF_LEN -1] = 0;          // null-terminated
        LabelTextSet(&Label12, (char*)send_buf);
    }
}

void TextPushButton15OnClick(tWidget *pWidget)
{
    static unsigned char send_buf[BUF_LEN];
    static unsigned char recv_buf[BUF_LEN];

    if ( h == INVALID_HANDLE ) return; // not ready yet

    index++;

    if (hmi_TCPState(h) == STATE_TCP_CONNECTED)
    {
        usprintf((char*)send_buf, "DATA%05d", index);
        //sends data and then receives data through a TCP session.
        hmi_TCPSendCmdEx(h, send_buf, BUF_LEN, recv_buf, BUF_LEN, TIMEOUT);
        recv_buf[BUF_LEN -1] = 0;          // null-terminated
        LabelTextSet(&Label12, (char*)send_buf);
    }
}
```

5. Setup device.

The setup device methods depends on the type of TouchPAD device and download methods, refer to the Section 3.4 Downloading Methods for TouchPAD of the TouchPAD Hardware User Manual for more detailed information.

In this example, we use the TPD-433-H device to connect the Host PC via USB wiring and turn the rotary switch to position 9 (USB update mode) then reboot TouchPAD device.

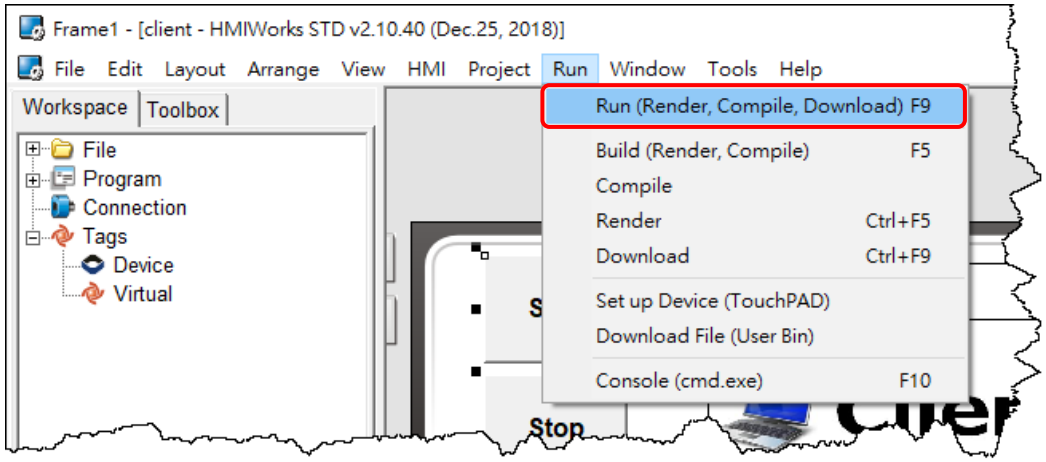
Click the menu item **"Run → Setup Device (TouchPAD)"** to configure the network settings (e.g., DHCP) and select the USB download interface.



6. Compiling and Downloading to Run.

The downloading program method to the TouchPAD depends on the type of TouchPAD device, refer to the Section 3.4 Downloading Methods for TouchPAD of the TouchPAD Hardware User Manual for more detailed information.

Click the menu item **"Run →Run (Render, Compile, Download) F9"**, or press **<F9>** key to download the **"client"** program to the TouchPAD device. Once the download is complete, set the rotary switch to position 0 (Run mode) and reboot TouchPAD device.



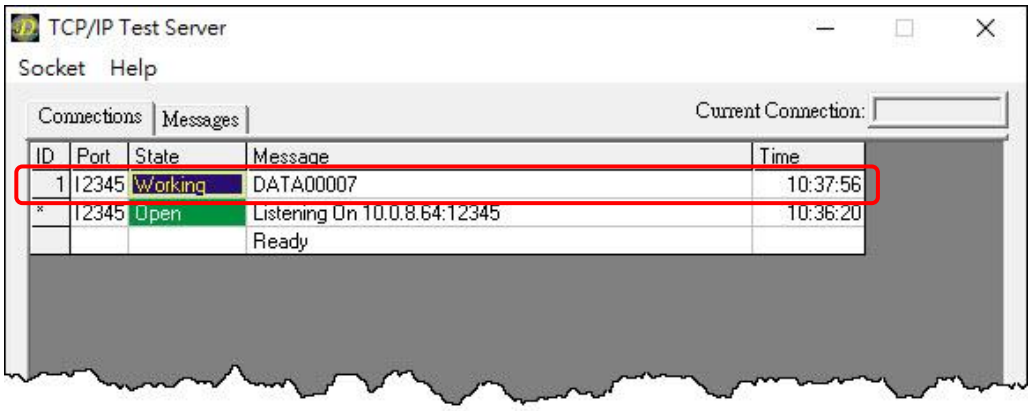
7. The TouchPAD device will then display the **“client”** program.

Step 3 TCP Testing Application

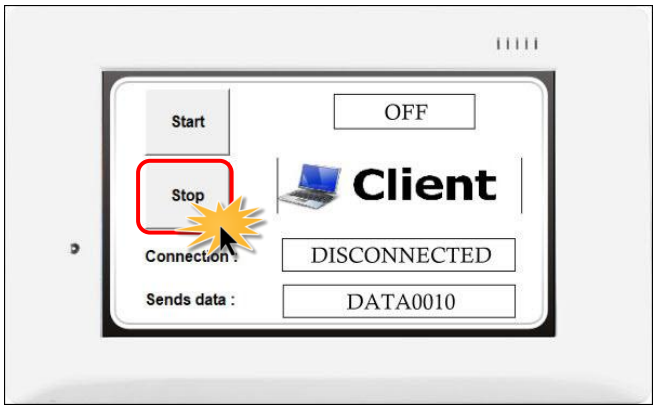
1. Click the **“Start”** button to connect to TCP Server (e.g., PC) on the TouchPAD device.

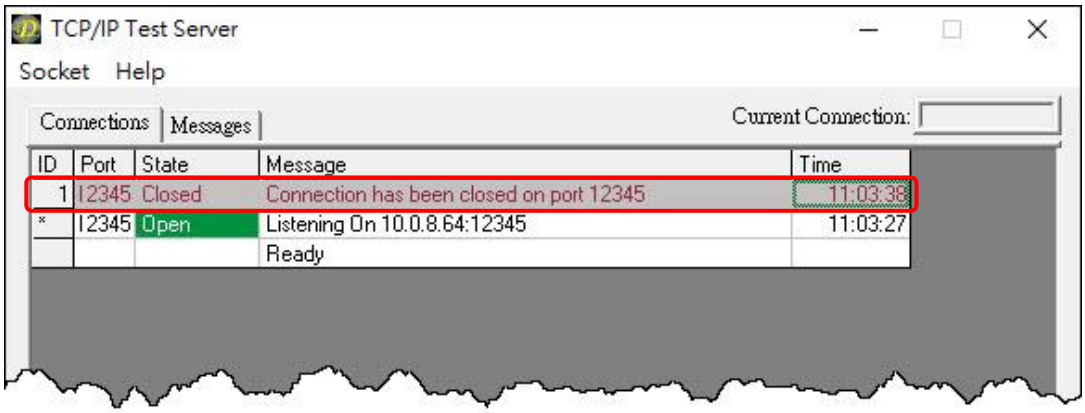
2. Verify that connection status is **“CONNECTED”** in the **“Connection”** field and send the message in the **“Sends data”** field on the TouchPAD device.

3. On the TCP/IP Test Server dialog box, verify that TCP server (e.g., PC) will receive this message in the **"Working"** line.



4. Click the **"Stop"** button to disconnect on the TouchPAD device.





4.4.2 How to use TouchPAD as TCP Server?

In this example, we use TouchPAD #1 as TCP server to receive data from TouchPAD #2 (TCP Client), which will be described in more detail below.

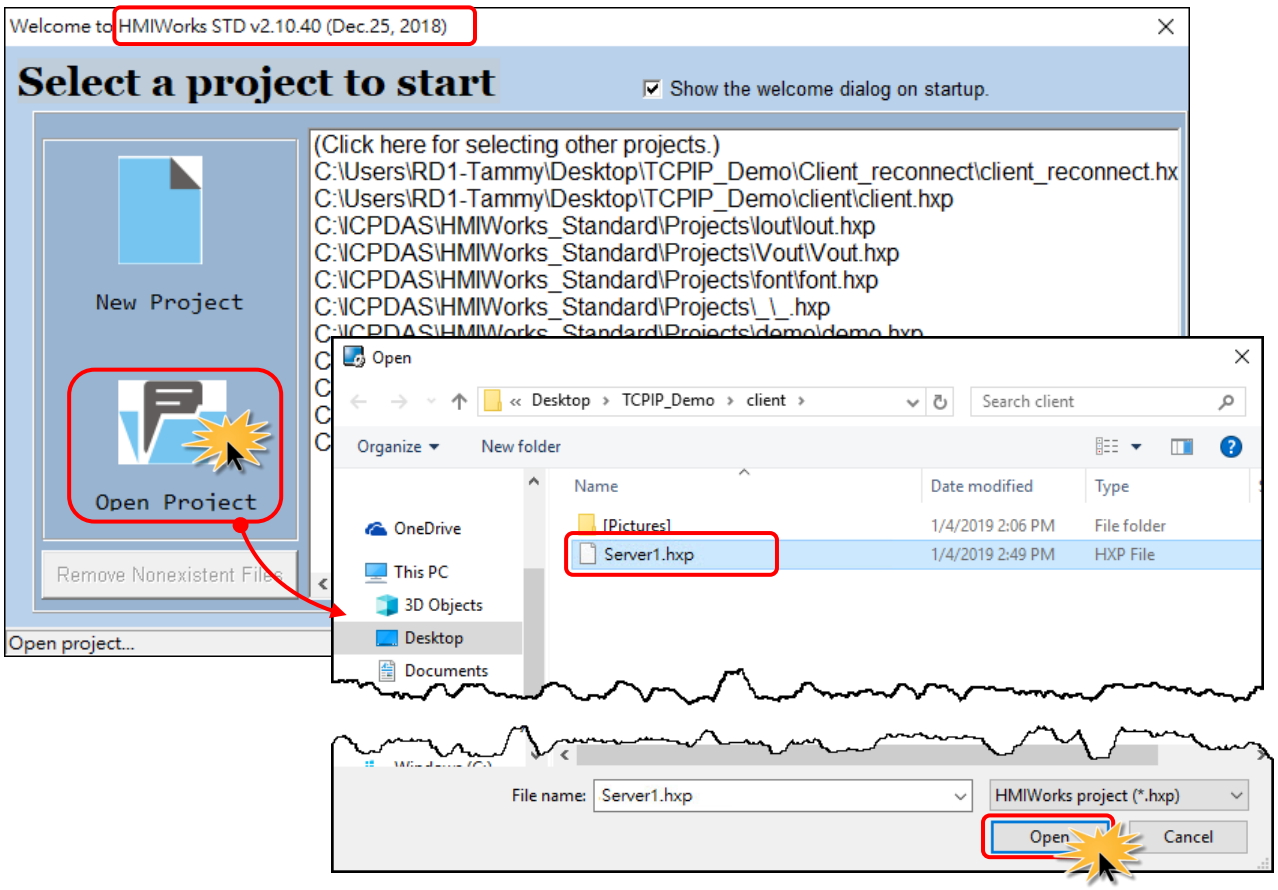
Step 1 Configure TouchPAD #1 to be a Server

1. Download and unzip the TCP/IP demo.
Download the file from the ICP DAS web site. The location of the download address is shown below:

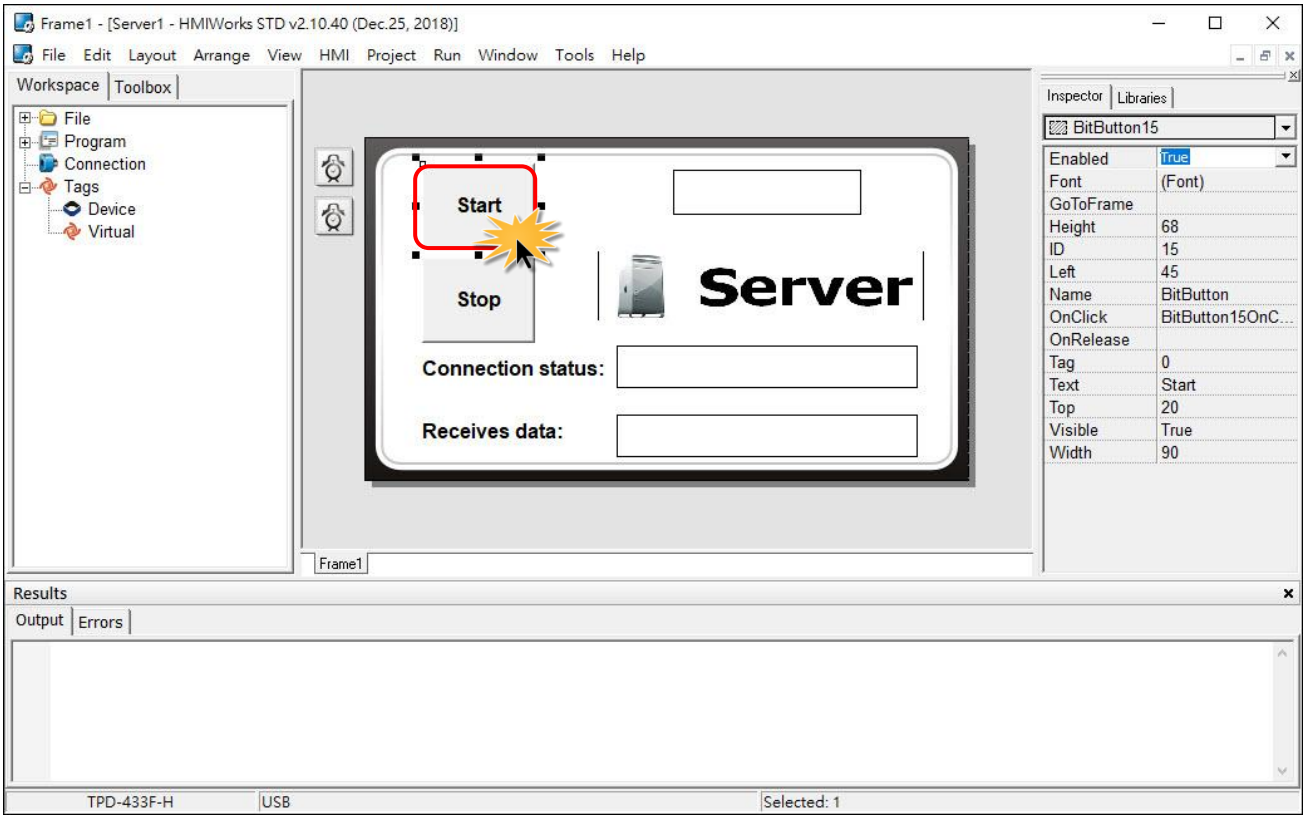
https://www.icpdas.com/en/download/show.php?num=1000

2. Launch the HMIWorks Standard software and open an existing **"Server1.hxp"** project.

Note: Check that your HMIWorks version is v2.10.40 or later. If your HMIWorks version is earlier than v2.10.40, the HMIWorks must be updated to the latest version.



3. Double-click the **"Start"** BitButton component to implement its OnClick event handler in the displayed programming window.



4. Assign a **TCP Port** of the TCP Server (e.g., 12345) in the define PORT line and click the **"OK"** button to save the file and leave.

The following code example establishes a TCP Server connection, the process involves opening the connection, e.g., **hmi_TCPNew()** and **hmi_TCPListen()**, receiving and sending the data, e.g., **hmi_TCPReadEx()** and **hmi_TCPOutput()**, closing the connection, e.g., **hmi_TCPClose()**. Refer to the <<HMIWorks API Reference>> for details about TCP API.

```
#define PORT            12345
#define TIMEOUT         500
#define BUF_LEN         24

static tHandle h = INVALID_HANDLE;    // handle for TCP Communications

void BitButton15OnClick(tWidget *pWidget)      // Start
{
   if ( h > INVALID_HANDLE ) return; // already have a connection

   // Allocate a session. Check if h < 0 to prevent using another tHandle
   h = hmi_TCPNew();          //Allocate a TCP session if possible

   if ( h > INVALID_HANDLE )          // if allocating a new session successfully
   {
        LabelTextSet(&Label4, "Listening");
        hmi_TCPListen(h, PORT);
   }
   else
   {
        static char szMsg[20];
        usprintf(szMsg, "Err= %d", h);
        LabelTextSet(&Label4, szMsg);
        h = INVALID_HANDLE; // don't keep the error code in h
   }
}

void BitButton6OnClick(tWidget *pWidget)          // Stop
{
   if (h >= 0)
   {
      hmi_TCPClose(h);    //closes and deallocates a TCP session.
      h = -1;
      LabelTextSet(&Label4, "OFF");
   }
}

void Timer14OnExecute(tWidget *pWidget)           //Connection status
{
    if (hmi_TCPState(h) == STATE_TCP_LISTEN)                    //gets the state of the TCP
session.
    {
        LabelTextSet(&Label10, "LISTENED");
    }
    if (hmi_TCPState(h) == STATE_TCP_CONNECTED)
    {
        LabelTextSet(&Label10, "CONNECTED");
    }
    else
```



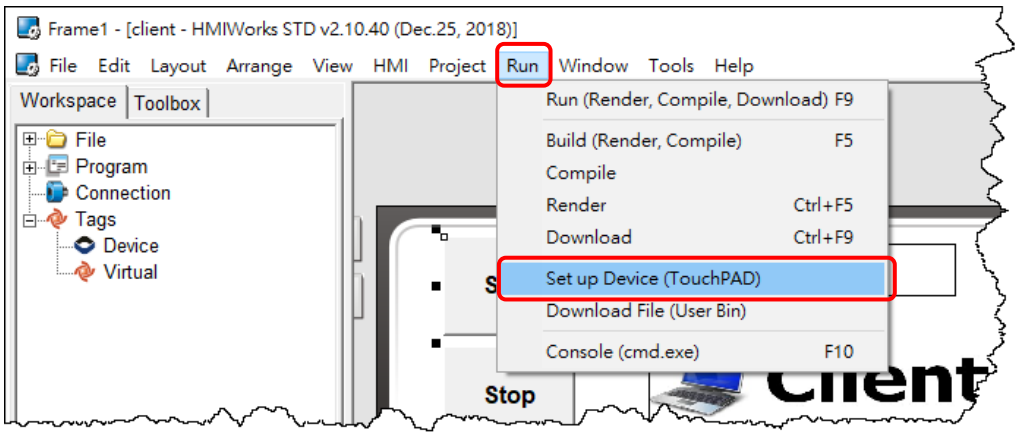
```
        {
            LabelTextSet(&Label10, "NO CONNECTED");
        }
    }

    void Timer9OnExecute(tWidget *pWidget) //Receiver data
    {
        static unsigned char recv_buf[64];
        int ret = 0;

        if ( h == INVALID_HANDLE ) return; // server does not ready

        if (hmi_TCPState(h) == STATE_TCP_CONNECTED) // client connected
        {
            //reads data through a TCP session.
            ret = hmi_TCPReadEx(h, recv_buf, BUF_LEN, TIMEOUT);
              LabelTextSet(&Label12, (char*)recv_buf);
                if (ret > 0)
                {
              //write back to the session immediately (no waiting in the queue).
                    hmi_TCPOutput(h, recv_buf, ret);
            }
        }
    }
```

5. Setup device.

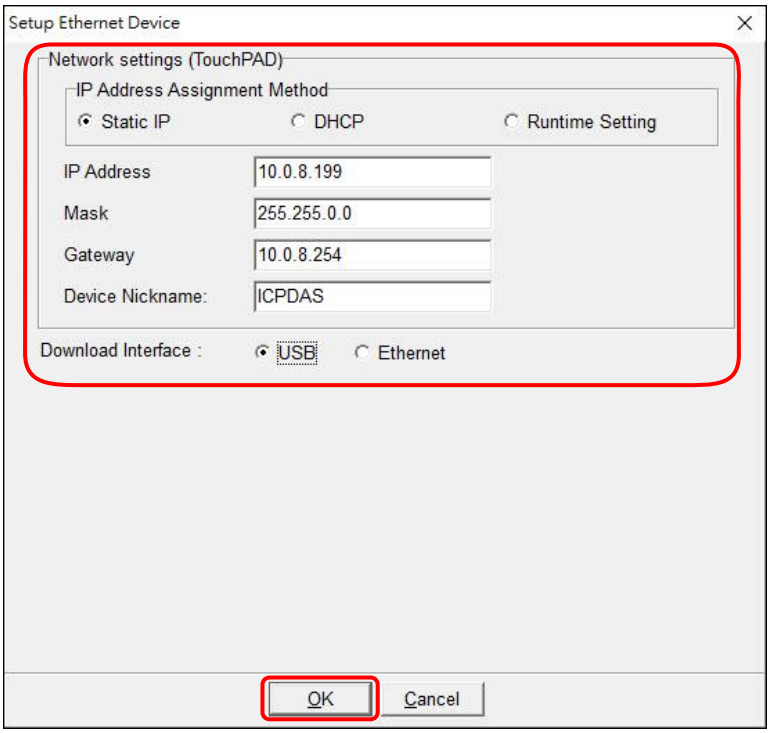
The setup device methods depends on the type of TouchPAD device and download methods, refer to the Section 3.4 Downloading Methods for TouchPAD of the TouchPAD Hardware User Manual for more detailed information.

In this example, we use the TPD-433-H device to connect the Host PC via USB wiring and turn the rotary switch to position 9 (USB update mode) then reboot TouchPAD device.

Click the menu item **"Run → Setup Device (TouchPAD)"** to open the “Setup Ethernet Device” dialog box.



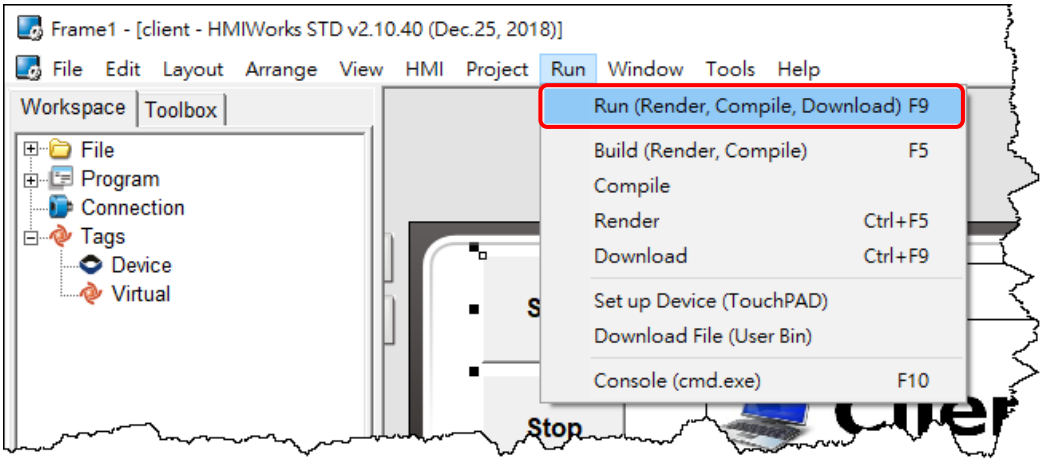
Configure the network settings and select the USB download interface and then click the **“OK”** button to save the file and leave.



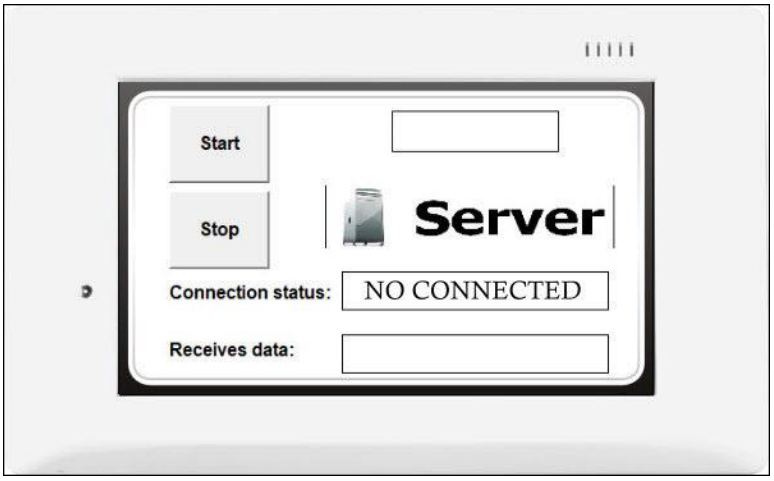
6. Compiling and Downloading to Run.

The downloading program method to the TouchPAD depends on the type of TouchPAD device, refer to the Section 3.4 Downloading Methods for TouchPAD of the TouchPAD Hardware User Manual for more detailed information.

Click the menu item **“Run →Run (Render, Compile, Download) F9”**, or press **<F9>** key to download the **“client”** program to the TouchPAD device. Once the download is complete, set the rotary switch to position 0 (Run mode) and reboot TouchPAD device.



7. The TouchPAD device will then display the **"Server1"** program.



Step 2 Configure TouchPAD #2 to be a Client

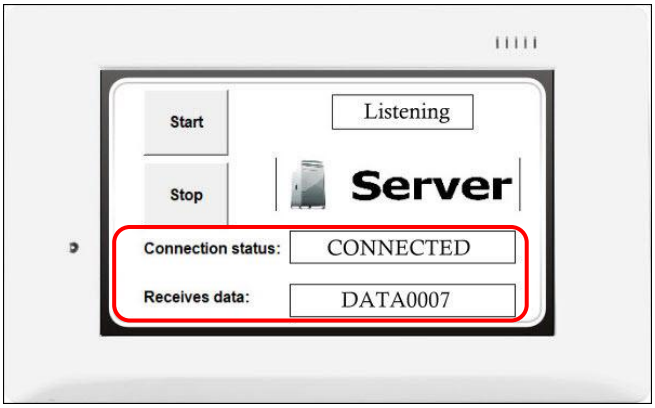
Refer to the **Step 2** in the Section 4.4.1 How to use TouchPAD as TCP Client for more details.

Step 3 TCP Testing Application

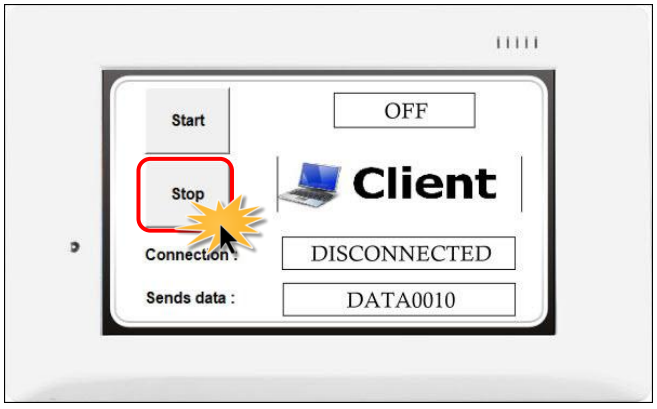
1. Click the **"Start"** button to start listening on the TouchPAD #1 (Server).
2. Click the **"Start"** button to connect to TCP Server on the TouchPAD #2 (Client).

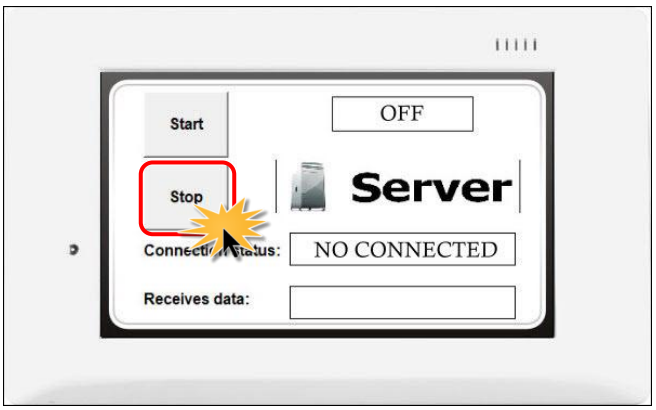
3. Verify that connection status is **“CONNECTED”** in the **“Connection”** field, and send the message in the **“Sends data”** field on the TouchPAD #2 (Client).

4. Verify that connection status is **“CONNECTED”** in the **“Connection status”** field and receive the message in the **“Receives data”** field on the TouchPAD #1 (Server).



5. Click the **“Stop”** button to disconnect on the TouchPAD #1 (Server) and TouchPAD #2 (Client).





5. Advanced Programming in C

We have an API reference for TouchPAD.
https://www.icpdas.com/en/download/show.php?num=958

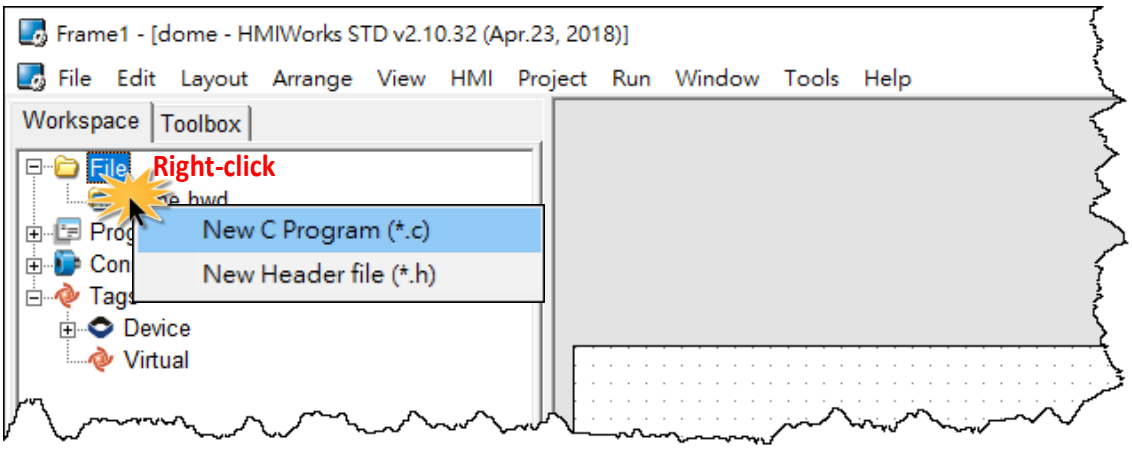
Though you can refer to the generated codes to learn how to use these API functions, all the API functions are defined in header files in the following path: "**C:\ICPDAS\HMIWorks_Standard\include\grlib**" and "**C:\ICPDAS\HMIWorks_Standard\include**", where "**C:\ICPDAS\HMIWorks_Standard**" is the installation path.

We give some examples in this chapter.

5.1 Adding a New File to Project

Before introducing the details, first we show how to add a new file (**"*.c"** or **"*.h"**) to the project.

1. Go to **"Workspace"** panel.
2. Right-click on the "**File**" item and a pop-up menu is displayed.
3. On that pop-up menu, choose the type (**"*.c"** or **"*.h"**) of the file you want to add.



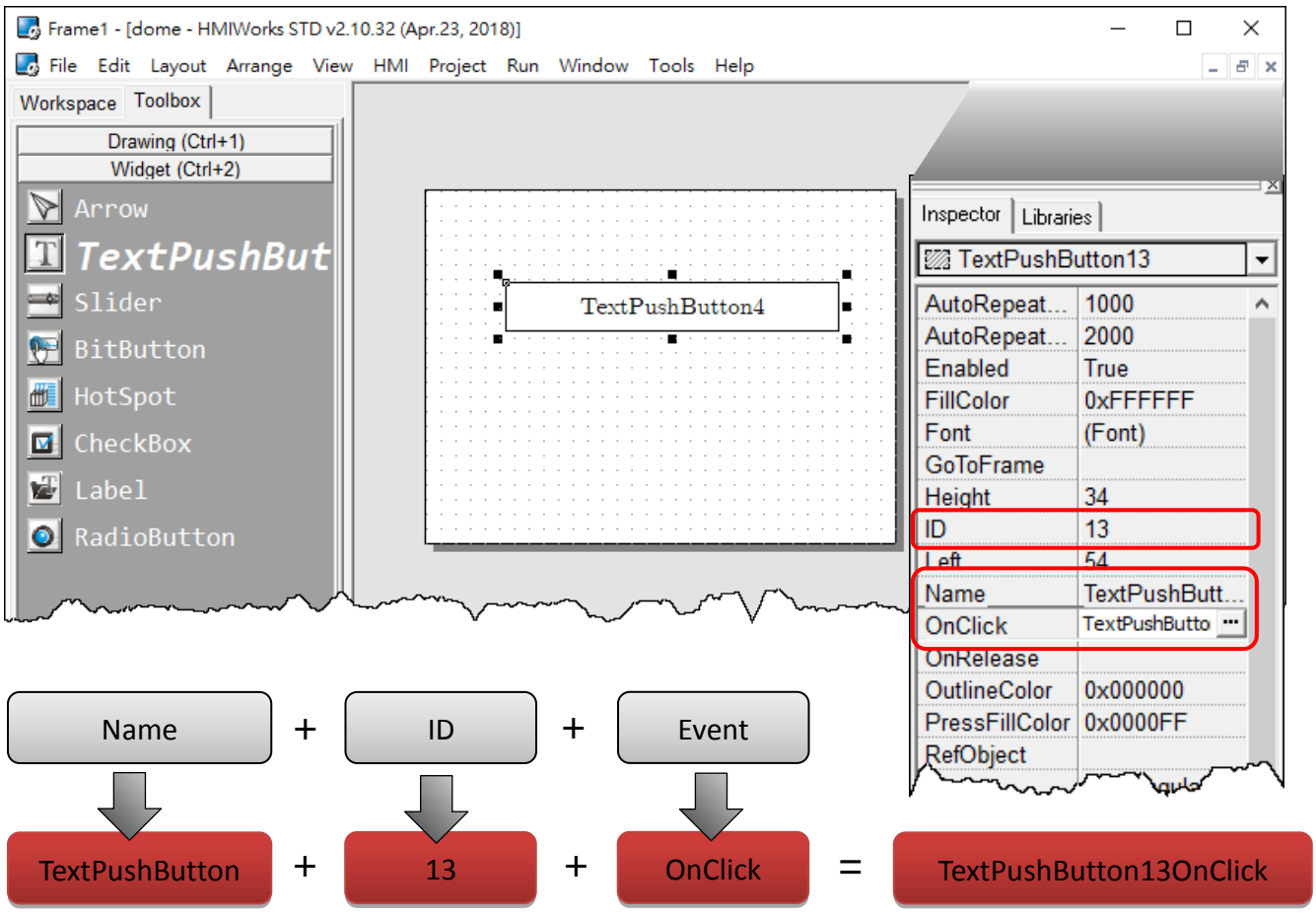
5.2 Updating Properties in Run Time

It is a bit more complicated to change the properties of widgets in the run time.
In this section, we demonstrate some commonly-used cases, including:

1. The **"FillColor"** and **"Text"** properties of a **TextPushButton** component. Refer to Section 5.2.1 for more detailed information.
2. The percentage of a **Slider** component. Refer to Section 5.2.2 for more detailed information.
3. The **"Selected"** property of a **CheckBox** component. Refer to Section 5.2.3 for more detailed information.
4. The **"Font"**, the **"Text"** and the **"TextColor"** properties of a **Label** component. Refer to Section 5.2.4 for more detailed information.

Updating properties is implemented in the event handlers of the widgets.

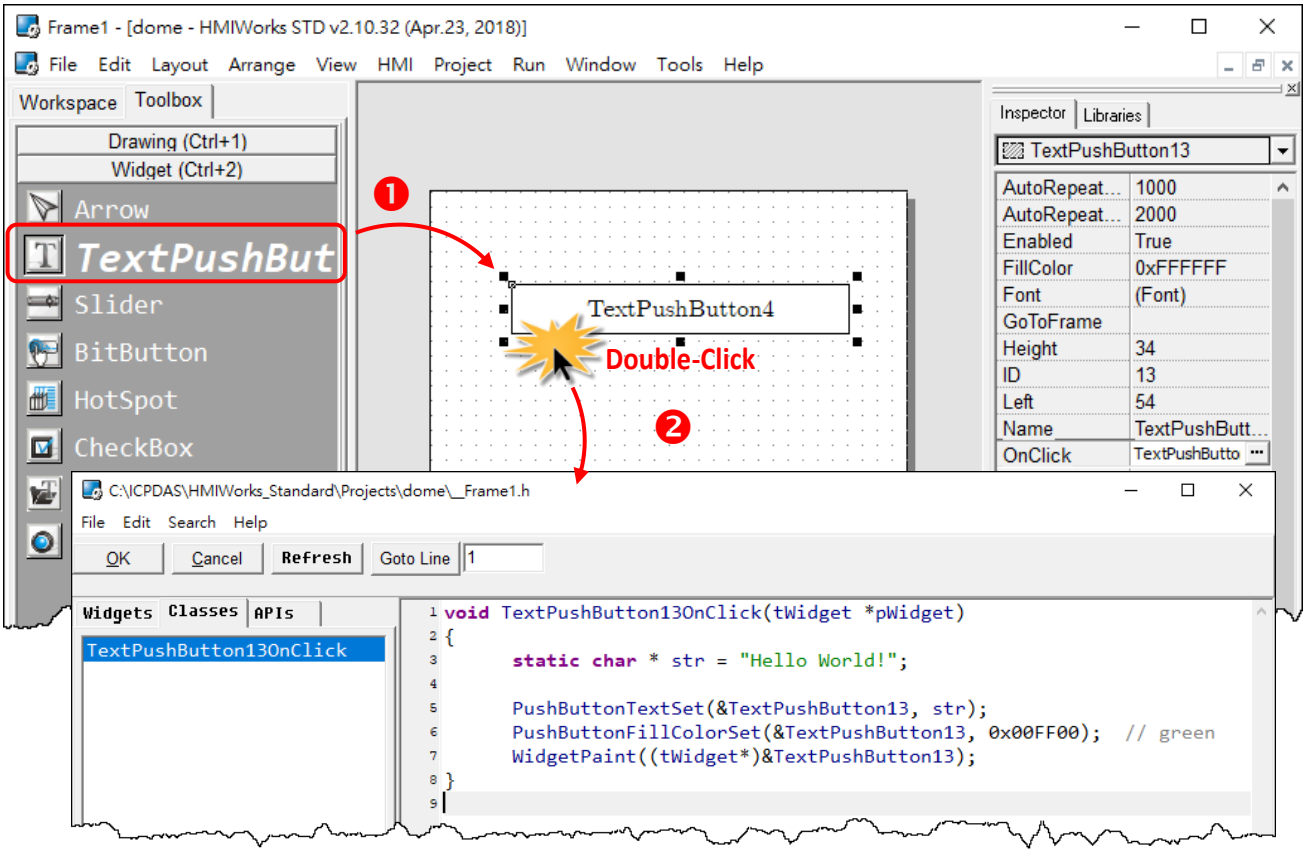
Note: The naming convention of the event handler of the widget (here the widget is the **TextPushButton** component) is shown as below:



5.2.1 FillColor and Text of a TextPushButton

This section shows how to change the **"FillColor"** and the **"Text"** properties of a **TextPushButton** component. Simply follow the steps below.

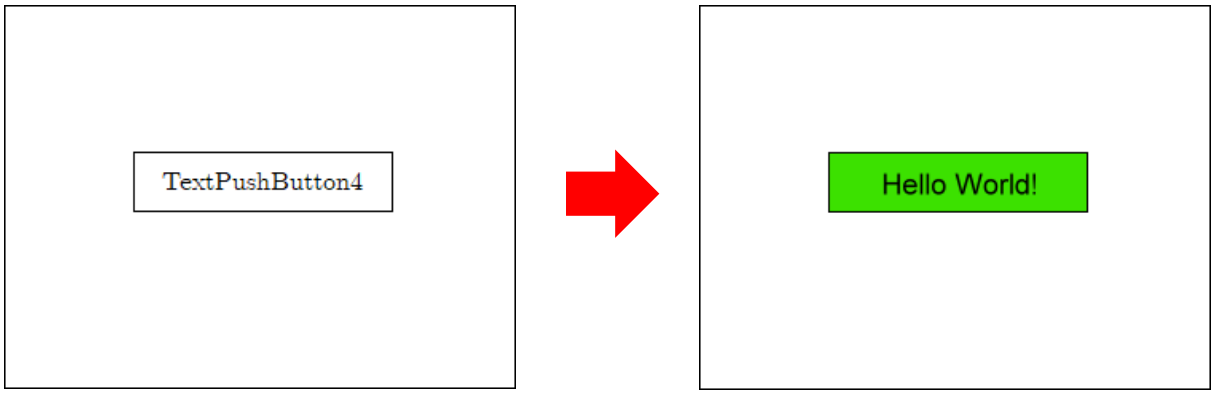
Step 1: Click the **TextPushButton** icon in the "**Toolbox**" panel and move your mouse to the frame design area. Click and drag a suitable sized **TextPushButton**.
Step 2: Double-click the **TextPushButton** component to implement its **OnClick** event handler in the displayed programming window. Then click the **OK** button to save the file and leave.



In order to make it clearer, we copy the above codes below.

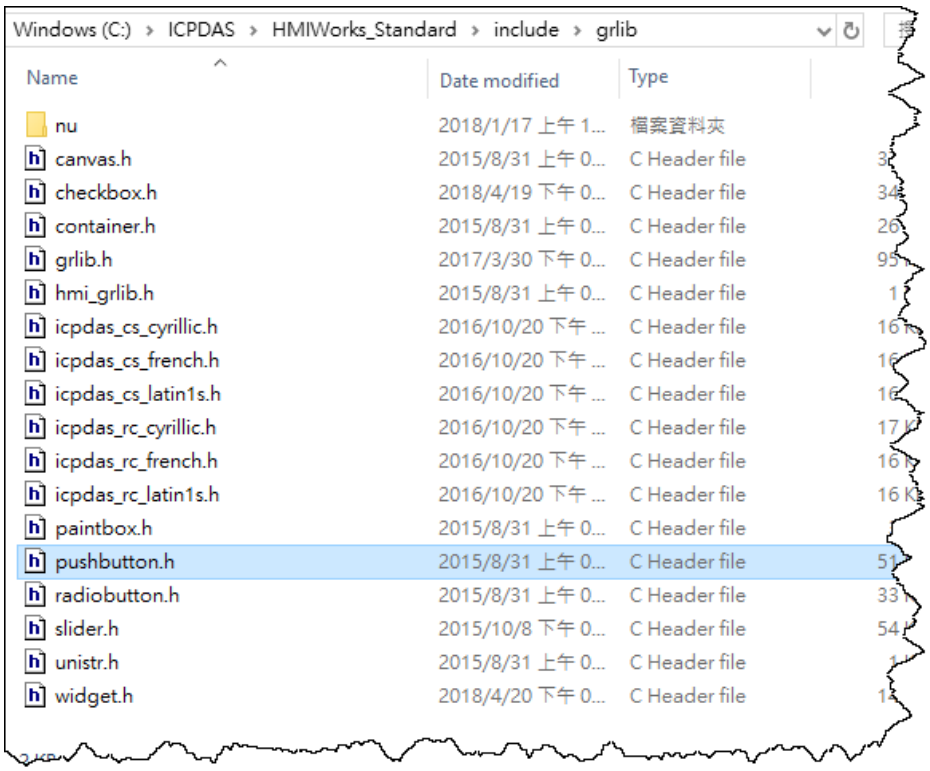
```
void TextPushButton13OnClick(tWidget *pWidget)
{
        static char * str = "Hello World!";

        PushButtonTextSet(&TextPushButton13, str);
        PushButtonFillColorSet(&TextPushButton13, 0x00FF00);    // green
        WidgetPaint((tWidget*)&TextPushButton13);
}
```


The effect of the **OnClick** event handler:



To set the **"Text"** property of a **TextPushButton**, we provide another function "**TextButtonTextSet**" for your convenience. Refer to the API reference for more details. The API reference can be downloaded from: https://www.icpdas.com/en/download/show.php?num=958

For more API functions of the **TextPushButton** component, refer to **pushbutton.h** in the following path: "**C:\ICPDAS\HMIWorks_Standard\include\grlib**", where "**C:\ICPDAS\HMIWorks_Standard**" is the installation path.

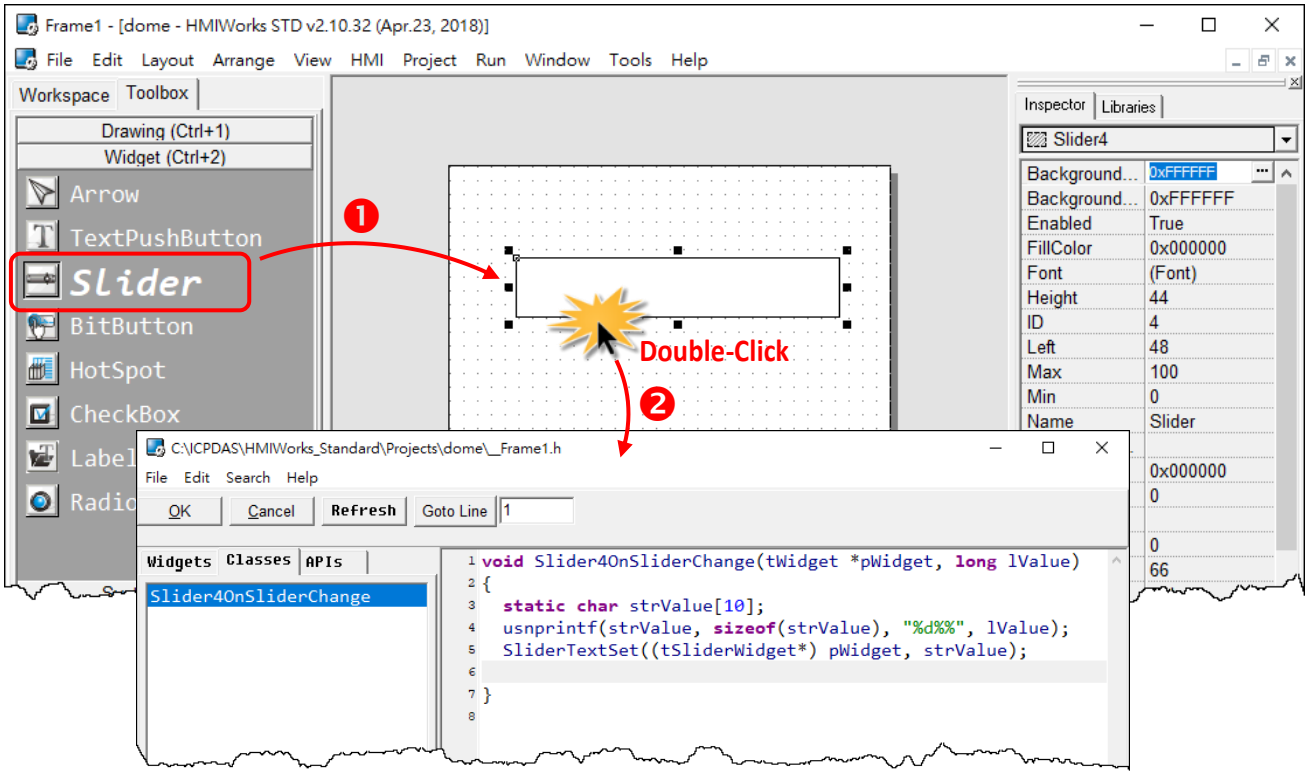


5.2.2 Percentage of a Slider

Simply follow the steps below to display the percentage of a **Slider** when it changes its position.

Step 1: Click the **Slider** icon in the "**Toolbox**" panel and move your mouse to the frame design area. Click and drag a suitable sized **Slider**.

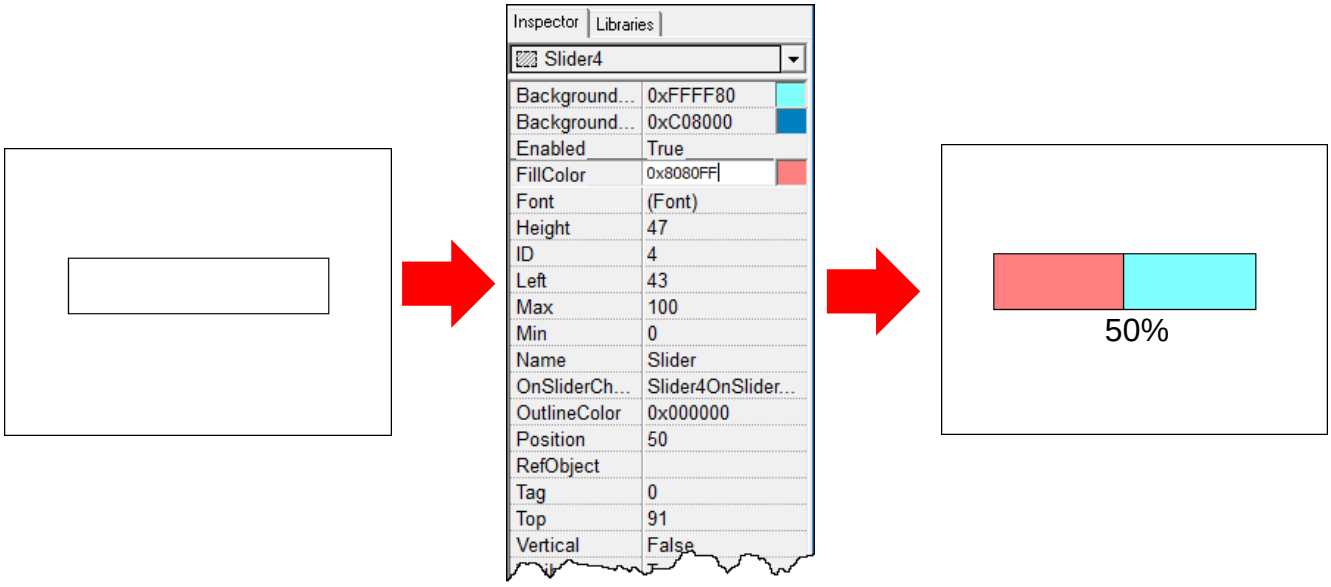
Step 2: Double-click the **Slider** component to implement its **OnSliderChange** event handler in the displayed programming window. Then click the **OK** button to save the file and leave.



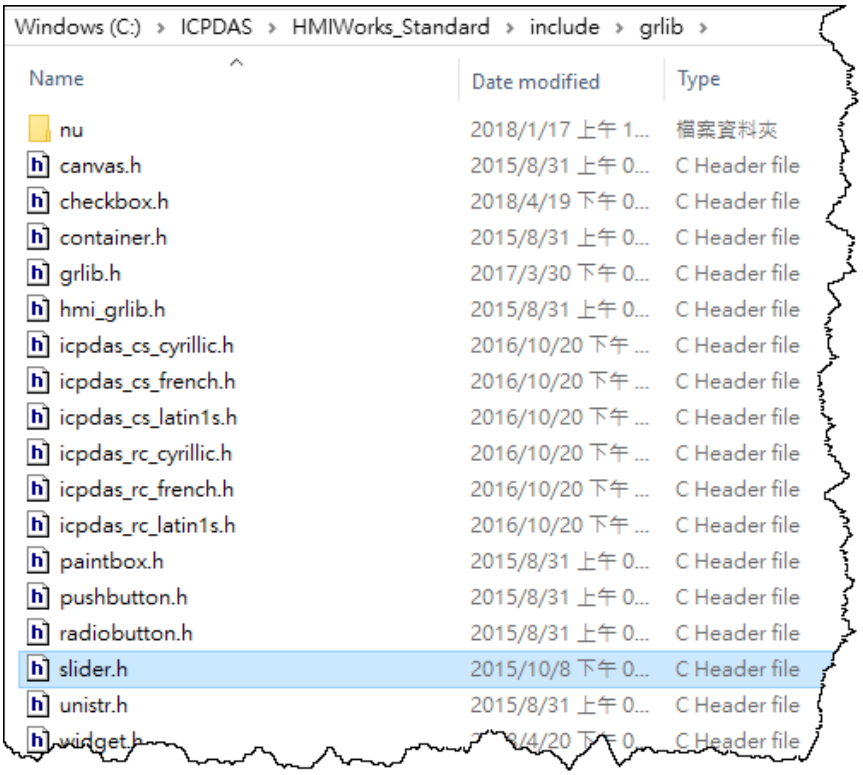
In order to make it clearer, we copy the above codes below.

```
void Slider4OnSliderChange(tWidget *pWidget, long lValue)
{
    static char strValue[10];
    usnprintf(strValue, sizeof(strValue), "%d%%", lValue);
    SliderTextSet((tSliderWidget*) pWidget, strValue);
}
```

The effect of the **OnSliderChange** function (after selecting colors):



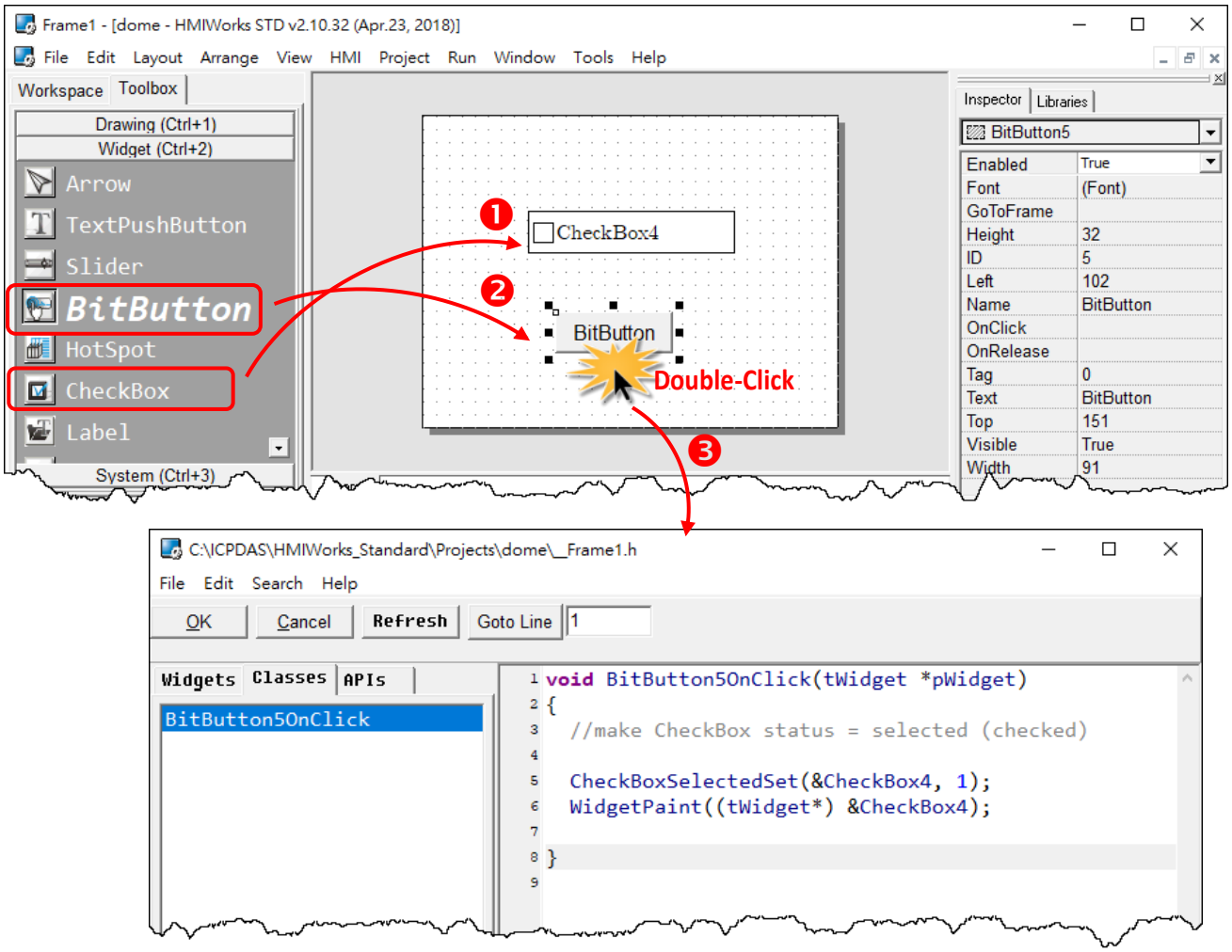
For more API functions of Slider, refer to **slider.h** in the following path: "**C:\ICPDAS\HMIWorks_Standard\include\grlib**", where "**C:\ICPDAS\HMIWorks_Standard**" is the installation path.



5.2.3 Selected of a CheckBox

Take the steps below for example to change the **"Selected"** property of a **CheckBox** component in the run time.

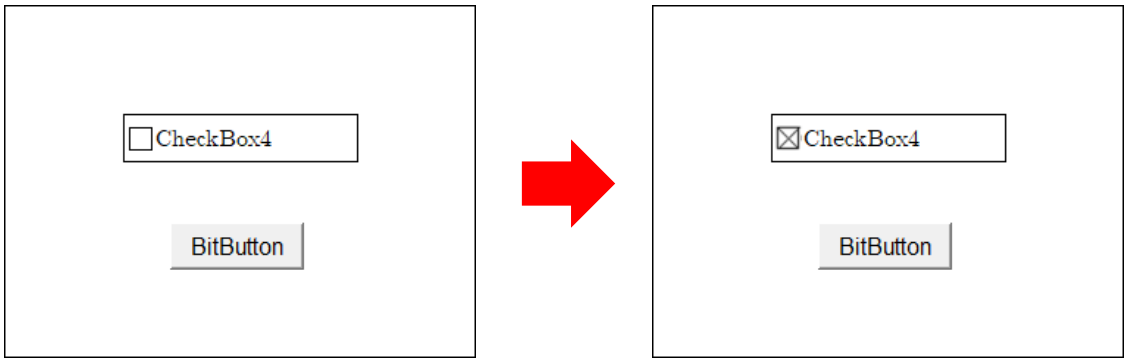
Step 1: Click the **CheckBox** icon in the "**Toolbox**" panel and move your mouse to the frame design area. Click and drag a suitable sized **CheckBox**.
Step 2: Repeat the same procedure as that of the **CheckBox** component for a **BitButton** component.
Step 3: Double-click the **BitButton** component to implement its **OnClick** event handler in the displayed programming window. Then click the **OK** button to save the file and leave.



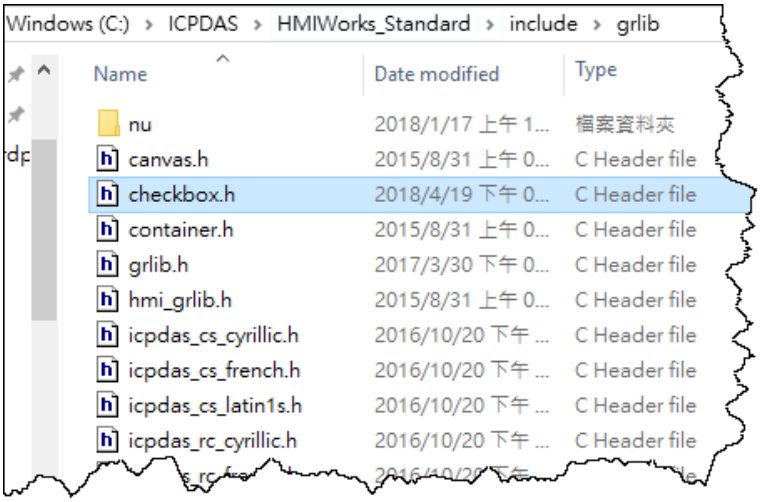
In order to make it clearer, we copy the above codes below.

```
void BitButton5OnClick(tWidget *pWidget)
{
    //make CheckBox status = selected (checked)

    CheckBoxSelectedSet(&CheckBox4, 1);
    WidgetPaint((tWidget*) &CheckBox4);
}
```

The effect of the **OnClick** function:



For more API functions of **CheckBox**, refer to **checkbox.h** in the following path: "**C:\ICPDAS\HMIWorks_Standard\include\grlib**", where "**C:\ICPDAS\HMIWorks_Standard**" is the installation path.



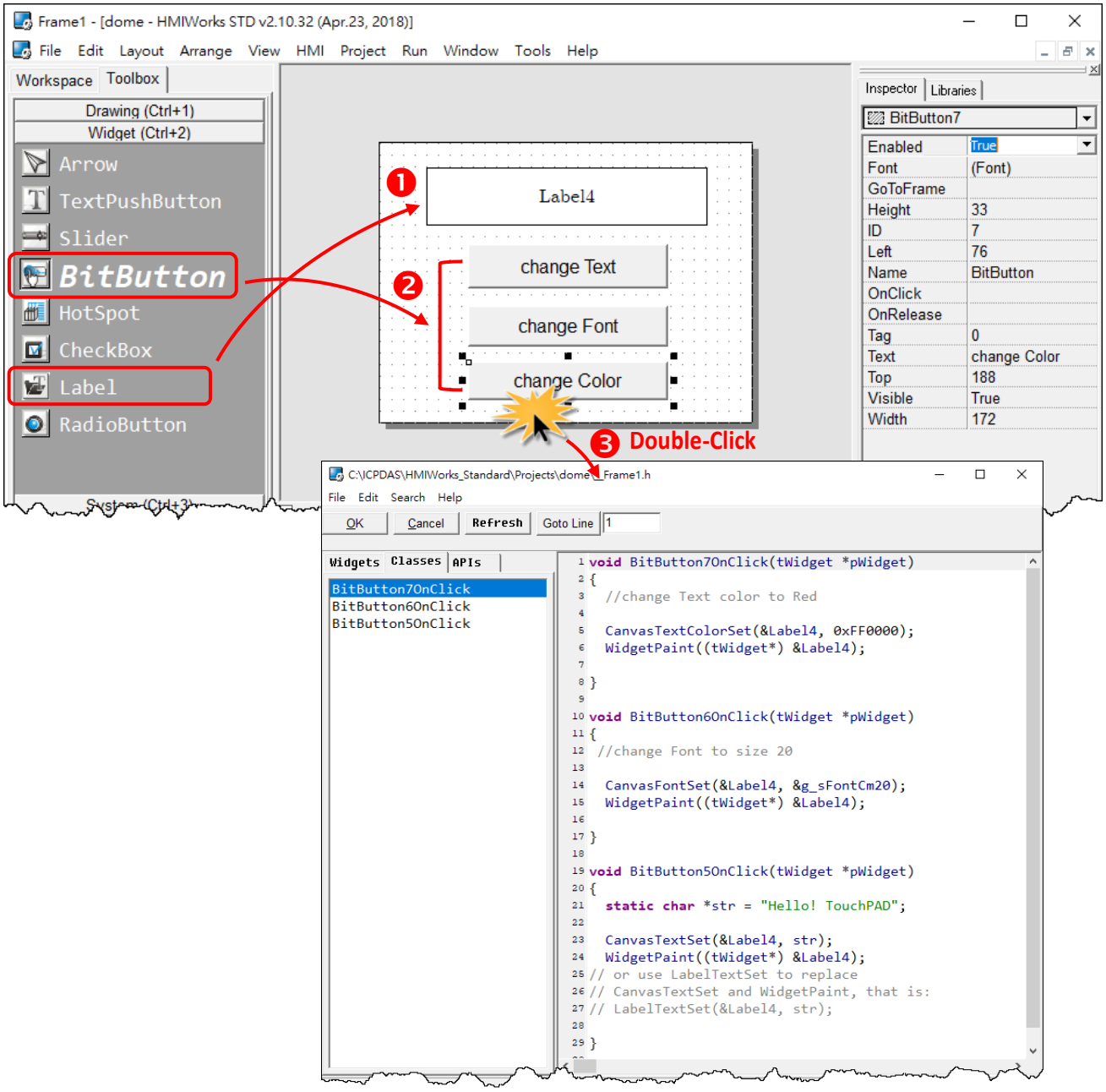
5.2.4 Font, Text and TextColor of a Label

Take the steps below for example to update properties of a **Label** component in the run time.

Step 1: Click the **Label** icon in the "**Toolbox**" panel and move your mouse to the frame design area. Click and drag a suitable sized **Label**.

Step 2: Repeat the same procedure as that of the **Label** component above for three **BitButton** components.

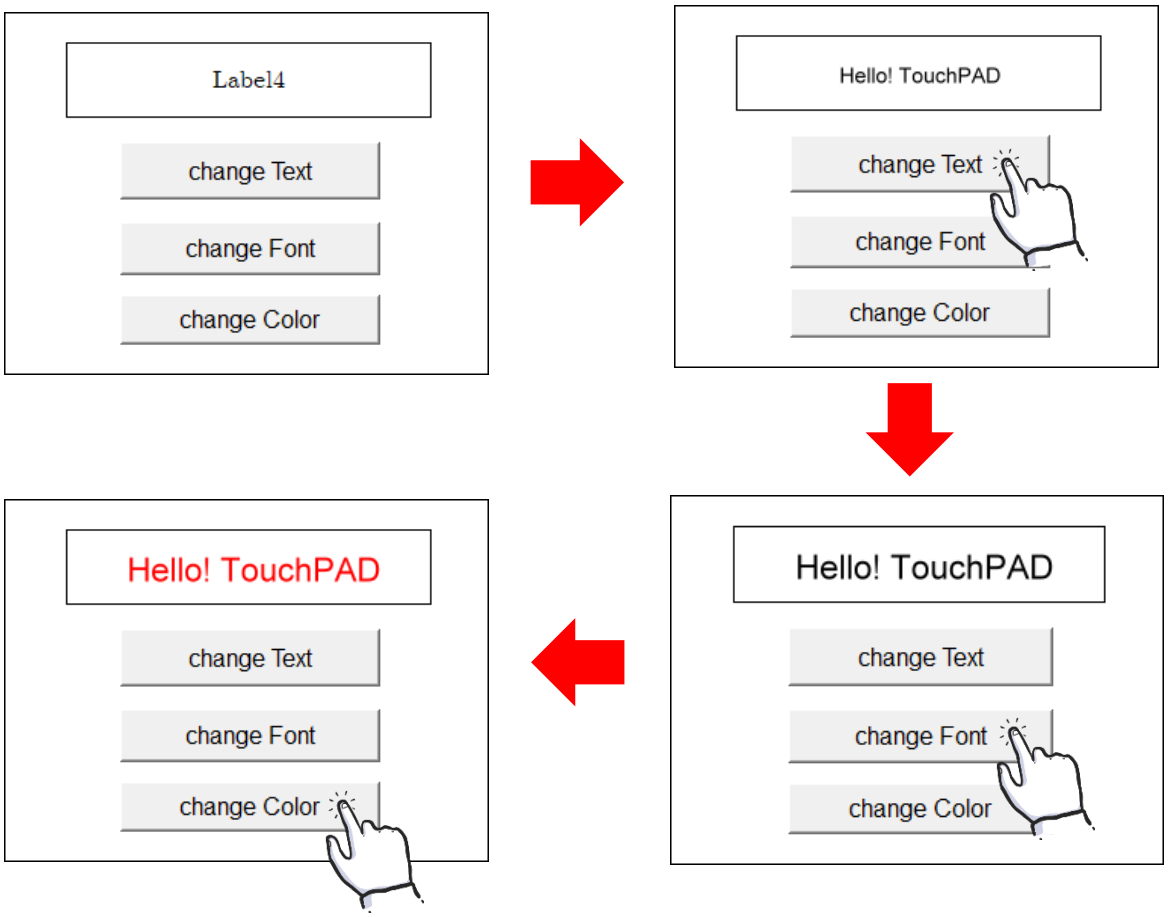
Step 3: Double-click the **BitButton** component to implement its **OnClick** event handler in the displayed programming window. Then click the **OK** button to save the file and leave.



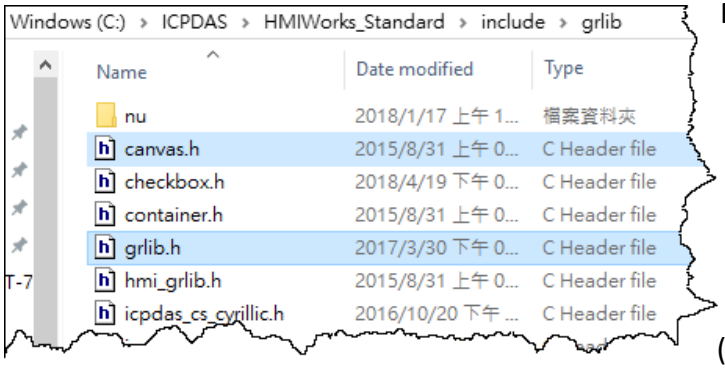
In order to make it clearer, we copy the above codes below.

```
//Click on BitButton7 “change Color”
void BitButton7OnClick(tWidget *pWidget)
{
    //change Text color to Red

    CanvasTextColorSet(&Label4, 0xFF0000);
    WidgetPaint((tWidget*) &Label4);
}

//Click on BitButton6 “change Font”
void BitButton6OnClick(tWidget *pWidget)
{
    //change Font to size 20

    CanvasFontSet(&Label4, &g_sFontCm20);
    WidgetPaint((tWidget*) &Label4);
}

//Click on BitButton5 “change Text”
void BitButton5OnClick(tWidget *pWidget)
{
    static char *str = "Hello! TouchPAD";

    CanvasTextSet(&Label4, str);
    WidgetPaint((tWidget*) &Label4);
// or use LabelTextSet to replace
// CanvasTextSet and WidgetPaint, that is:
// LabelTextSet(&Label4, str);
}
```

The effect of the **OnClick** function for three **BitButton** components:



To set the **"Text"** property of a **Label** component, we provide another function "**LabelTextSet**" for your convenience. Refer to the API reference for more details. The API reference can be downloaded from: https://www.icpdas.com/en/download/show.php?num=958



For more API functions of **Label**, refer to **canvas.h** in the following path: "**C:\ICPDAS\HMIWorks_Standard\include\grlib**", where "**C:\ICPDAS\HMIWorks_Standard**" is the installation path.

In the same path, there is a header file (**grlib.h**). The **grlib.h** contains prototypes for the pre-defined fonts, such as g_sFontCm20.

5.3 Accessing Tags in Ladder

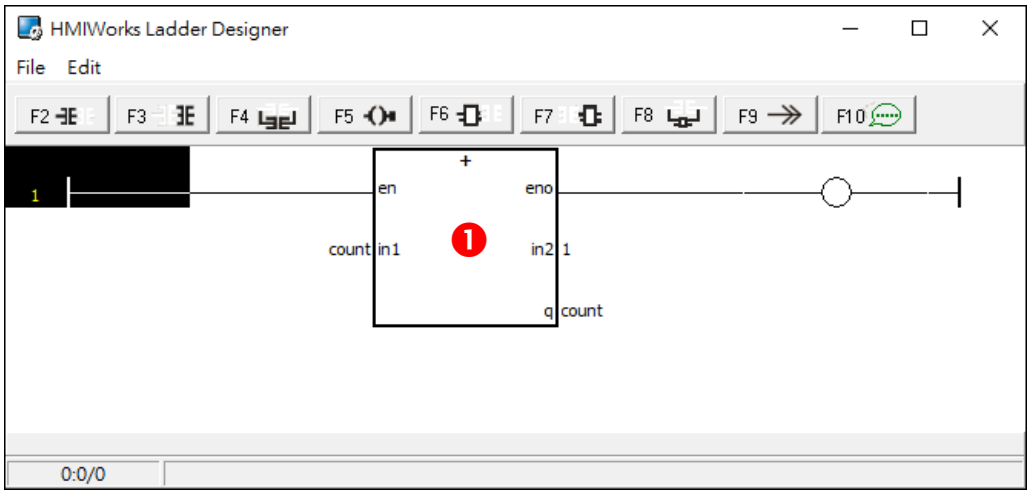
In HMIWorks, users can design a project with many frames of two different types, **"Standard C"** and **"Ladder"**. The variables (tags) used in the Ladder is transformed into a structure of the C language after building the project and thus the tags can be accessed in the frame of programming type **"Standard C"**.

Two macros are provided for this purpose:

1. **VAR_GET**: get the value from the tag in the Ladder
2. **VAR_SET**: set a value to the tag in the Ladder

Supposed that we have a tag named **"count"** incremented in the **Ladder**, and we can get the value of the **"count"** tag and set the **"count"** tag to zero as shown in the example below.

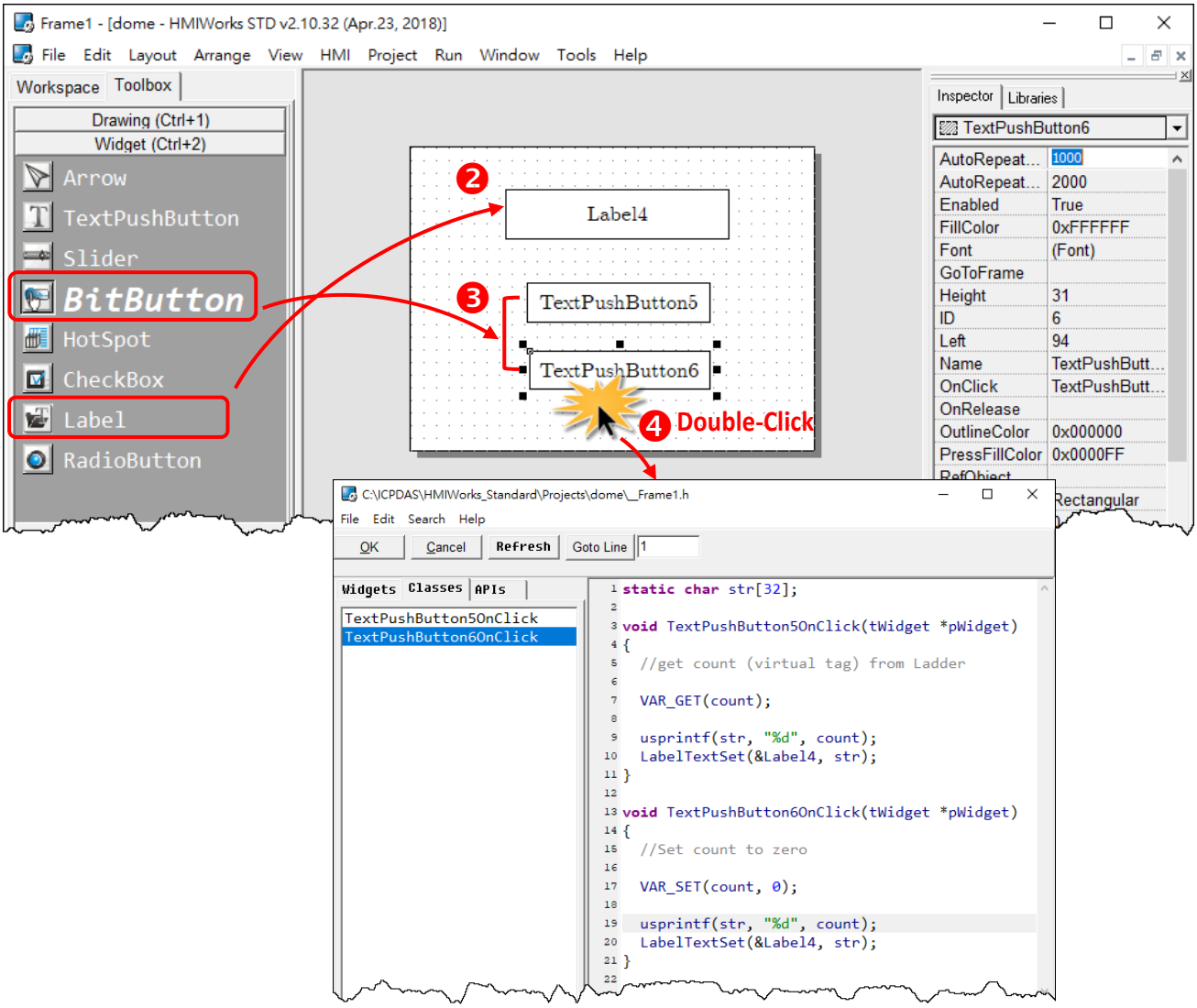
Step 1: Press **<F2>** key to add the **"count"** tag and press **<F4>** key to open the "HMIWorks Ladder Designer" window, and then create **"count"** incremented.



Step 2: Click the **Label** icon in the "**Toolbox**" panel and move your mouse to the frame design area. Click and drag a suitable sized **Label**.

Step 3: Repeat the same procedure as that of the **Label** component above for two **BitButton** components.

Step 4: Double-click the **BitButton** component to implement its **OnClick** event handler in the displayed programming window. Then click the **OK** button to save the file and leave.



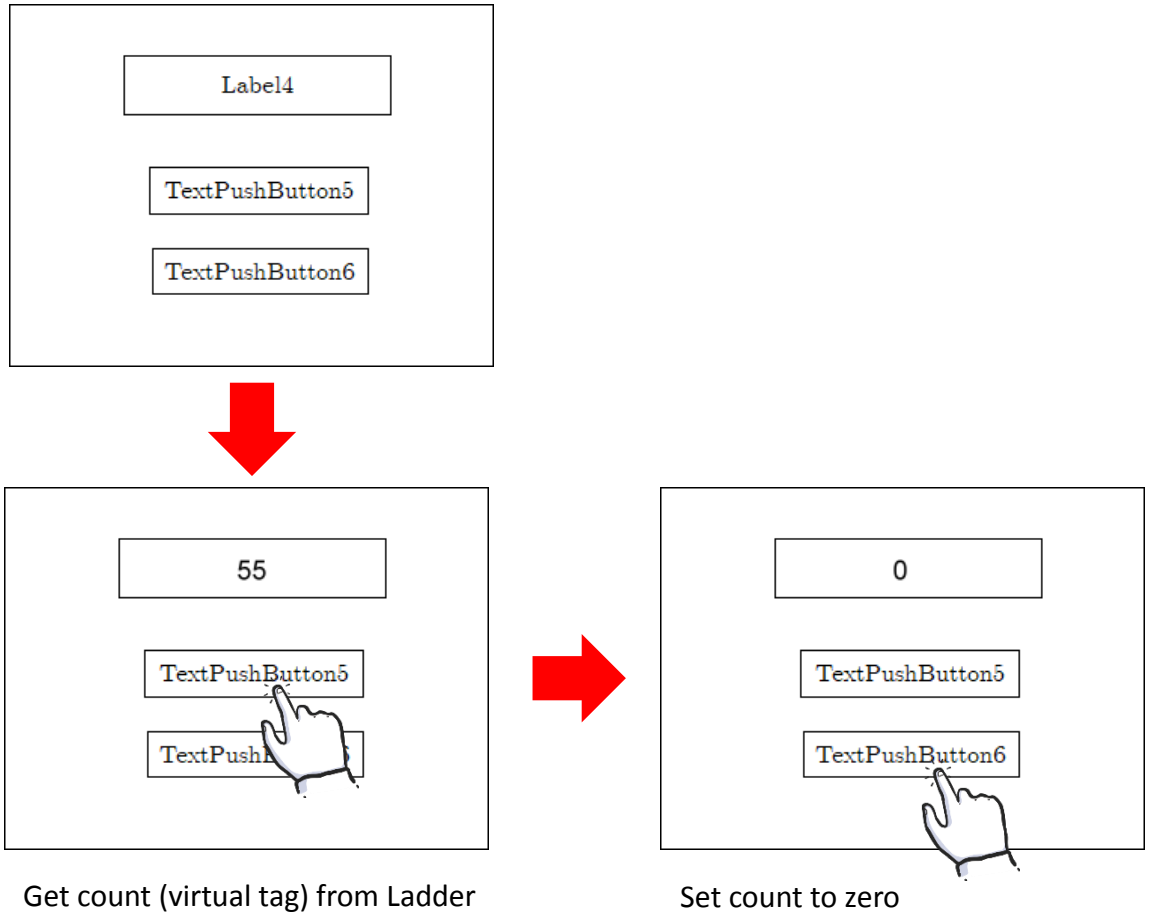
In order to make it clearer, we copy the above codes below.

```
static char str[32];

//Click on BitButton5 to get count (virtual tag) from Ladder
void TextPushButton5OnClick(tWidget *pWidget)
{
   VAR_GET(count);

   usprintf(str, "%d", count);
   LabelTextSet(&Label4, str);
}
```



```
// Click on BitButton6 to Set count to zero
void TextPushButton6OnClick(tWidget *pWidget)
{
    VAR_SET(count, 0);

    usprintf(str, "%d", count);
    LabelTextSet(&Label4, str);
}
```

The effect of the **OnClick** function for two **BitButton** components:



Get count (virtual tag) from Ladder

Set count to zero

Appendix

A. FAQ

For more detailed FAQ, refer to
https://www.icpdas.com/en/faq/index.php?fkw=hmiworks#882

A.1.What to do if screen flashes?

Refer to Section 3.4.2 Frame for more details.

A.2.How can I improve the picture quality on the TouchPAD?

Refer to Section 3.4.6 Picture for more details.

A.3.How does a TouchPAD control I/O?

Refer to Section 3.3.6 Associate Tags with Tools and Section 3.4.17 ObjectList for more details.

A.4.How to change Font of Text?

Refer to Section 3.4.5 Text for more details.

A.5.How to represent decimals for Ladder Designer?

Refer to Section 3.4.13 Label for more details.

A.6.How to clear the paint box?

Refer to Section 3.4.16 PaintBox for more details.

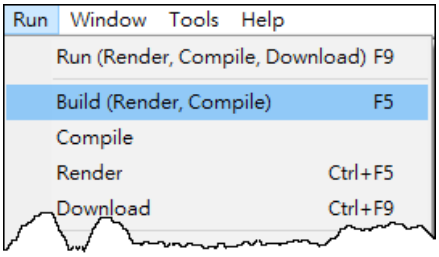
A.7.How to remove the startup beep of the TouchPAD?

Some TouchPAD devices sound a beep when startup, refer to Section 3.2.2 Project Configurations for more details.

A.8.How to customize the generated code?

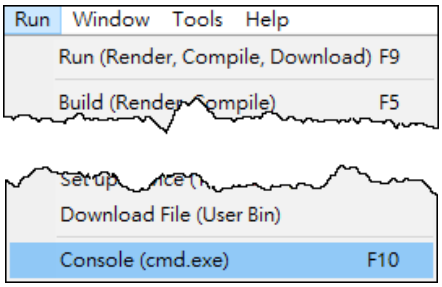
Every time when building a project, HMIWorks generates source codes to build. Below is the procedure to customize the generated source codes.

1. After finishing designing the project, press **<F5>** key (build) instead of **<F9>** key (run) to generate codes.



2. In the directory of the project, open the source file (**.c files**).

3. Edit the source files (**.c files**).

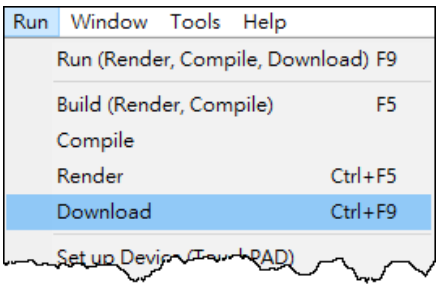


4. Press **<F10>** key and a Command Prompt window (cmd.exe) are displayed. Enter "**make**" in the Command Prompt window (cmd.exe) to re-make the project.

5. For the TPD-283U-H/TPD-283U-Mx, there are additional steps that need to be executed after entering **make**.

Enter **"make genbix"**

6. Press **<Ctrl>** + **<F9>** key to download the **.bin** (or **.bix**) file.



A.9.How to store data in the flash?

For users' convenience, there are two sets of API functions for data storage in the flash on the TouchPAD devices. One is for the MCU (micro-controller unit) internal flash and the other is the external serial flash.

To user these features, install the HMIWorks software with version 2.03 or above. The HMIWorks software can be downloaded from: https://www.icpdas.com/en/download/show.php?num=944

No.	1	2
Target Flash	MCU internal flash	External serial flash
Possible Target Device	All devices in the TouchPAD series	All devices in the TouchPAD series, except TPD-280 and TPD-283 (for those having external flash)
API Functions Provided*	hmi_UserParamsGet, hmi_UserParamsSet	hmi_UserFlashReadEx, hmi_UserFlashWriteEx, hmi_UserFlashConfig, hmi_UserFlashErase
Size of Storage	256 byte	4 KB ~ 7 MB
Suggested Users	Any TouchPAD users	**For advanced users only.** Any undetermined use will damage the application image.

* Refer to the API reference for more details. The API reference can be downloaded from: https://www.icpdas.com/en/download/show.php?num=958

A.10. How to use soft reset?

There are two methods to reset a TouchPAD by software.

Method 1: Use the API function of **hmi_SoftwareReset**.

Method 2: Use the Watchdog.

1. Configure watchdog.

Click the **"Project Configuration"** from the **"Project"** menu to configure the watchdog option.

2. Use infinite loop to start up watchdog.

For example: while(1){}

If you need to use this function in ladder, refer to the Section3.3.5 User-Dfined Function Block for more details.

A.11. How to use TouchPAD as Modbus RTU/TCP Slave?

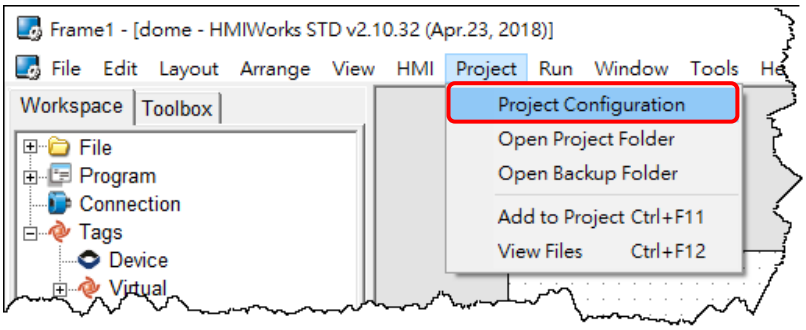
Refer to the FAQ: How to use TouchPAD as Modbus RTU Slave? and FAQ: How to use TouchPAD as Modbus TCP Slave? for more details.

A.12. How do I Project migrations form Non -H to -H Version of TouchPAD?

For example, migrating projects from TPD-433F to TPD-433F-H:

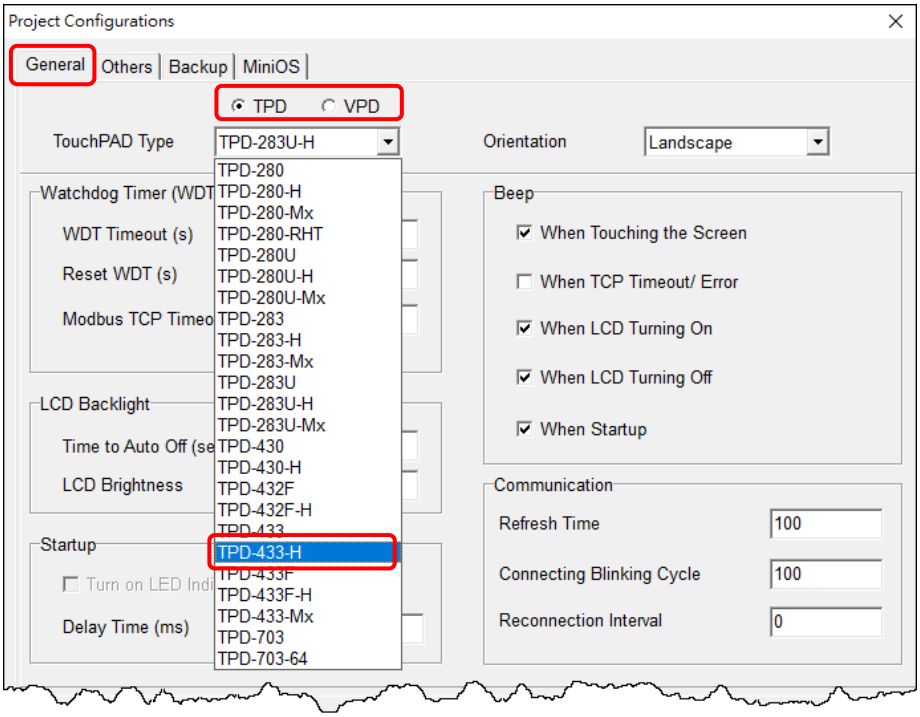
1. Open original project by using HMIWorks v2.10.22 or last version.

2. Click the HMIWorks menu item **"Project" → "Project Configuration"**.



3. Click the **"General"** page and select the **"TPD"** option.

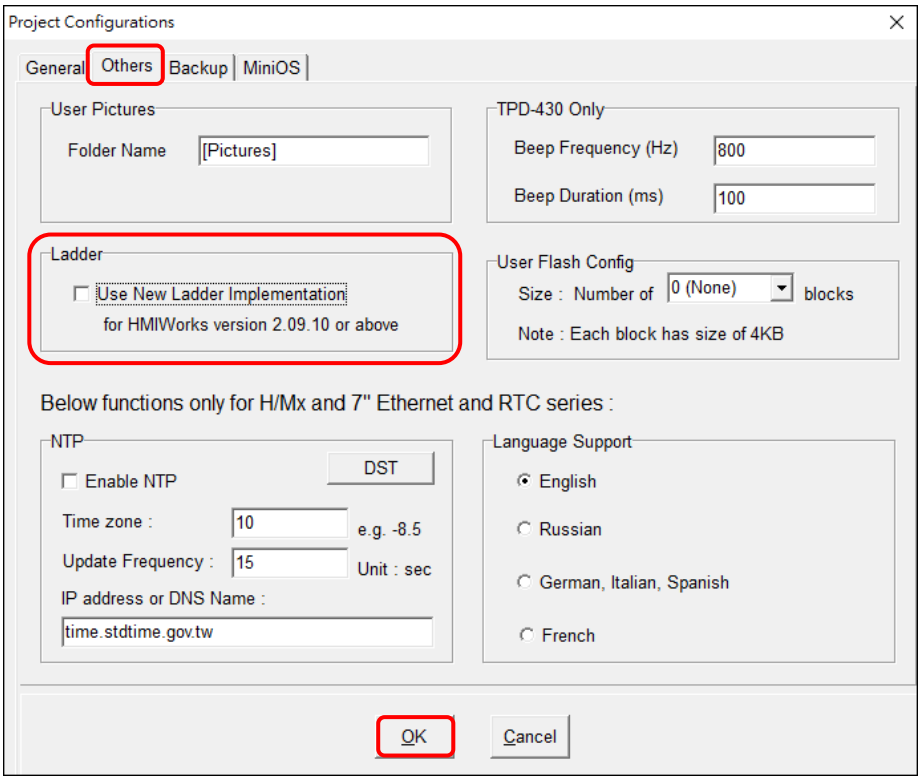
4. Change the **"TouchPAD Type"** to TPD-433F-H.



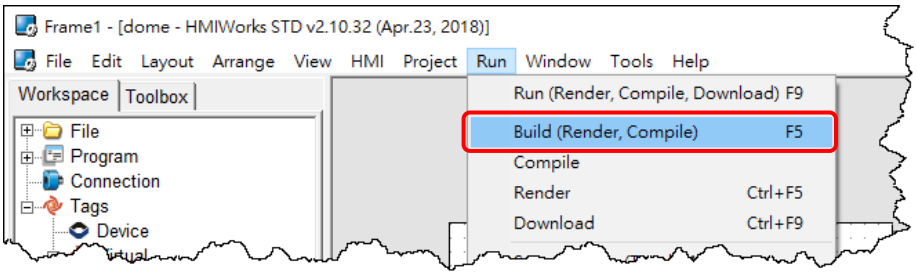
If your original project uses Ladder program, and is created by HMIWorks v2.09.09 or older versions, please follow the **steps 5 - 6** to disable the new Ladder mode.

New ladder mode: The Coil-Set and Coil-Reset change the coil state and lock it (industrial standard) until reset or set. Other coil operations will not unlock or change it.
Old ladder mode: There is no lock feature.

5. Click the **"Others"** page.
6. Unchecked the **"Use New Ladder Implementation"** item in the **"Ladder"** pane.
7. Click **"OK"** button.



8. Click the HMIWorks menu item **"Run" → "Build (Render, Compile)"** to rebuild project or press **<F5>** key.



B. Revision History

This chapter provides revision history information to this document.

The table below shows the revision history.

Revision	Date	Description
1.0.25	May. 2015	Initial issue
1.1.0	Jul. 2015	The user manual is divided into HMIWorks software and TouchPAD Hardware.
1.2.0	May. 2018	Update the HMIWorks (v2.10.32) operation picture. Update the function blocks of Ladder. Add Section 4.3 Integrating TouchPAD with I/O Modules. Add FAQ: How do I project migrations form Non -H to -H version of TouchPAD.
1.3.0	Jan. 2019	Add Section 4.4 TCP/IP Communication Add Section 4.4.1 How to use TouchPAD as TCP Client Add Section 4.4.2 How to use TouchPAD as TCP Server
1.4.0	Jun. 2020	Add introduction of new function blocks, changed to section 3.3.4.
1.5.0	Jul. 2022	Add introduction of new function blocks (DGW-521).