EasyBuilder8000 User's Manual

For EasyBuilder8000 Version 4.65.08

Chapter 1 EasyBuilder Installation and Startup Guide

1.1 EasyBuilder Installation

Software:

Download EasyBuilder from CD or visit Weintek Labs, Inc.'s website at http://www.weintek.com. The language versions include Simplified Chinese, Traditional Chinese, English, Italian, Korean, Spanish, Russian, and French. The latest upgraded files can be downloaded too.

Hardware Requirements (Recommended):

CPU: INTEL Pentium II or higher
Memory: 256MB or higher
Hard Disk: 2.5GB or higher (Disc space available at least 500MB)
CD-ROM: 4X or higher
Display: 1024 x 768 resolution or greater
Keyboard and Mouse
Ethernet: for project downloading/uploading
USB Port 2.0: for project downloading/uploading
RS-232 COM: for on-line simulation
Printer

Operating System:

Windows XP / Windows Vista / Windows 7.

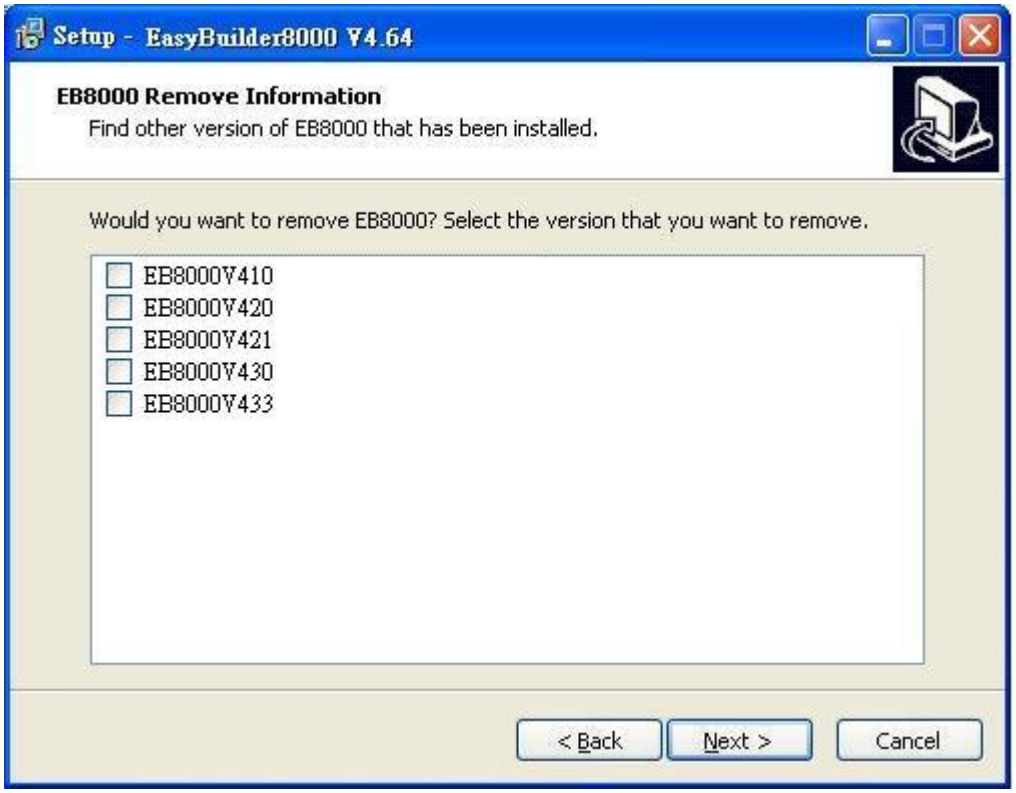
1.2 Steps to Install EasyBuilder

1. Installing EasyBuilder:

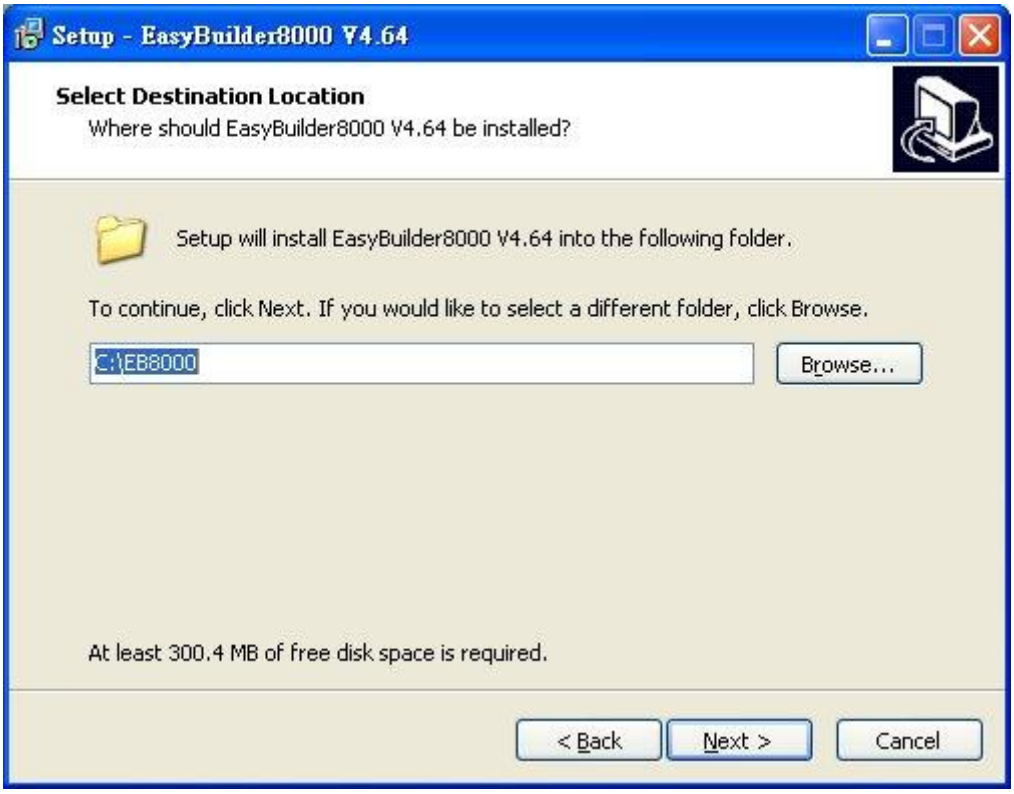
Put the disk into the CD drive. The computer will run the program automatically or execute under the root directory **[Autorun.exe]** manually. The installation screen is shown below.

2. Click **[Install]**, the dialog below is shown, select the language and click **[Next].**

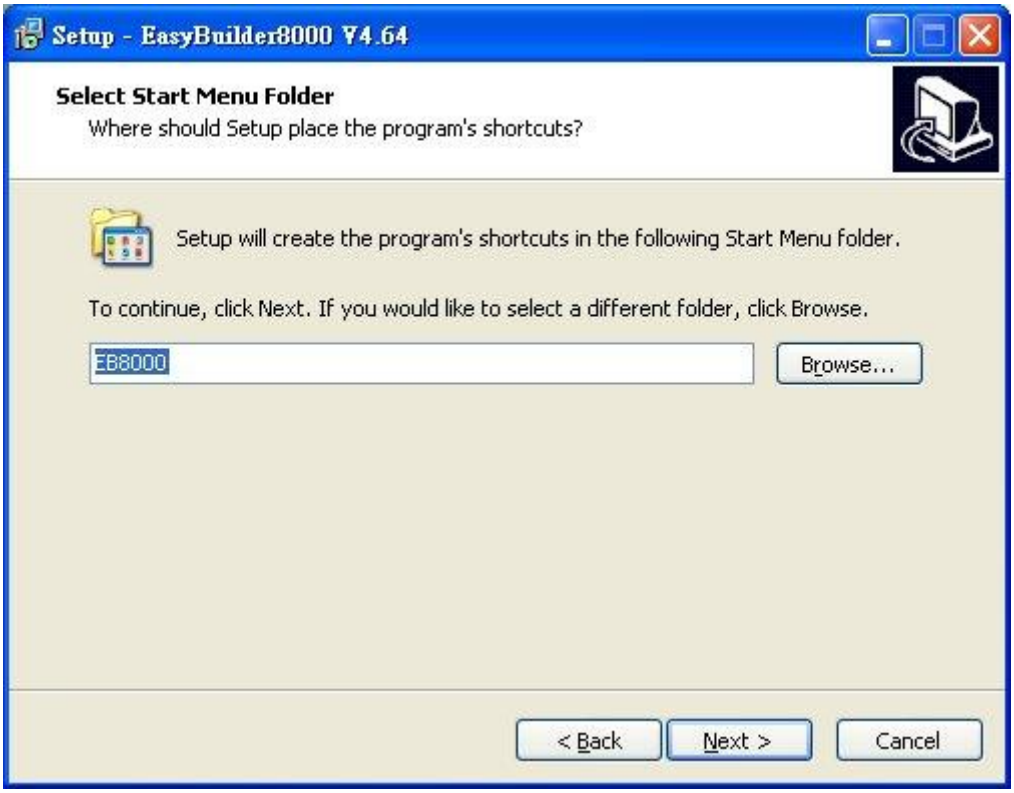
3. To remove the old versions of EasyBuilder, please select the check box and click **[Next]**.



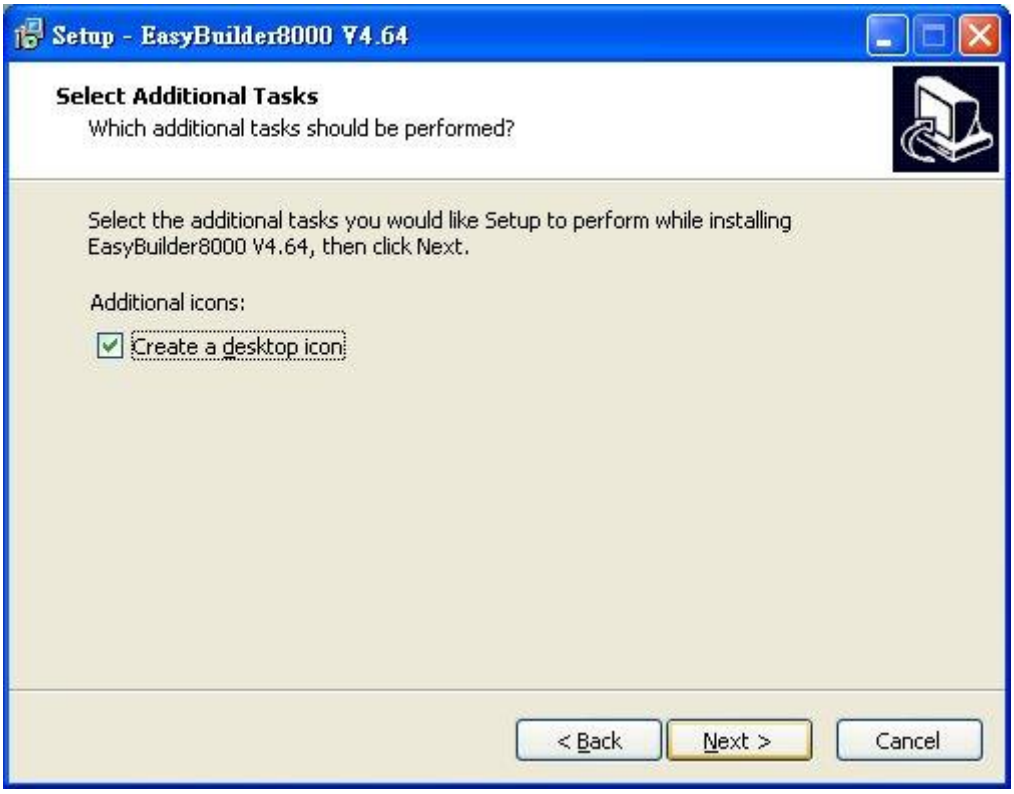
4. Designate a new folder for EasyBuilder installation or use the folder recommended and then click **[Next]**.



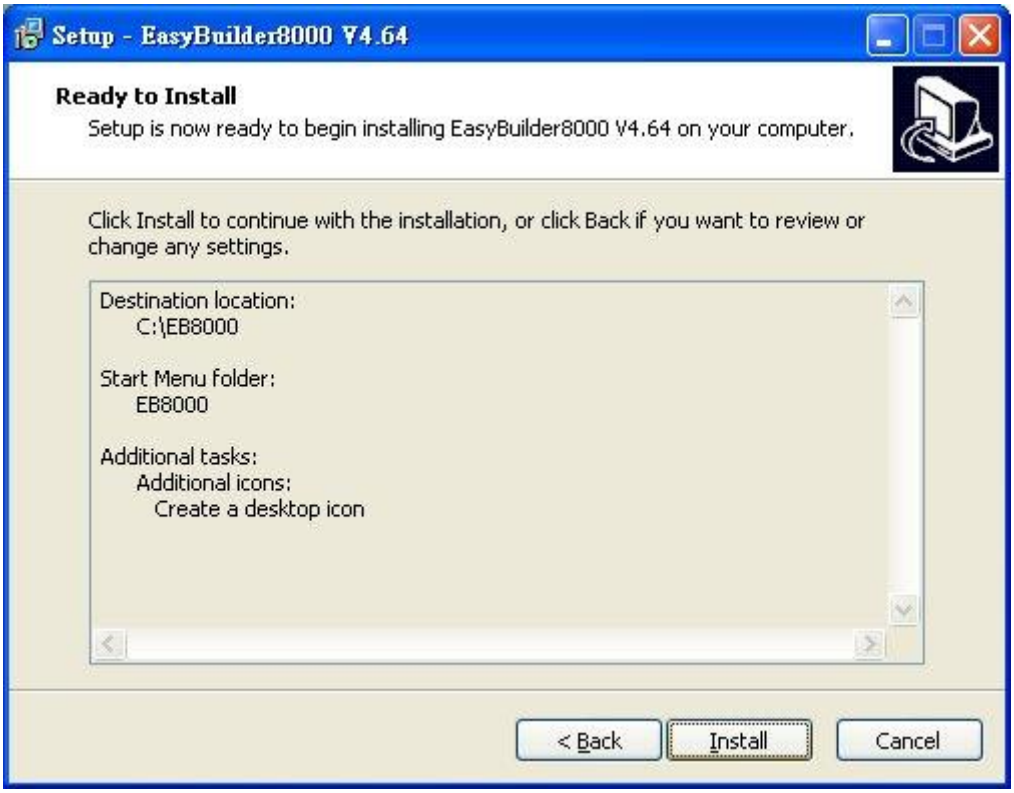
5. Select a start menu folder to save the program's shortcuts. Click **[Browse]** to designate a folder or use the folder recommended then click **[Next]**.



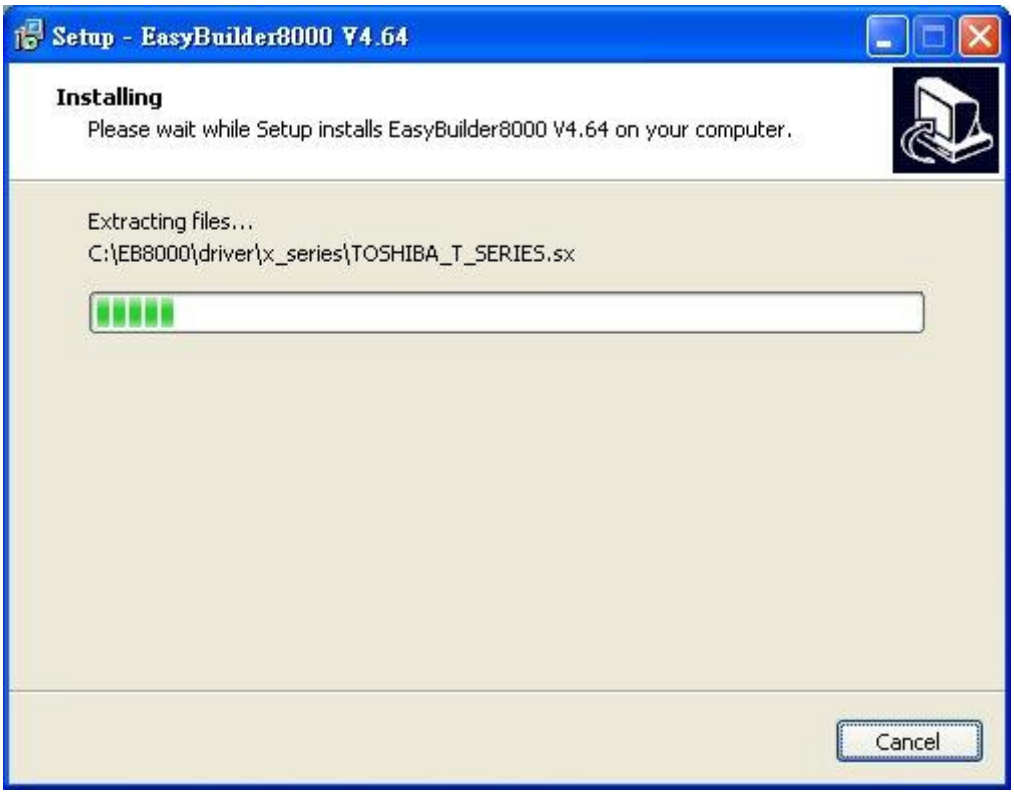
6. Users will be enquired if there are any additional tasks to be done. For example: **[Create a desktop icon]**. Select it if needed then click **[Next]** to continue.



7. When finish settings, please check if they are all correct. If any changes need to be made, click **[Back]** to change the settings or click **[Install]** to start installing.

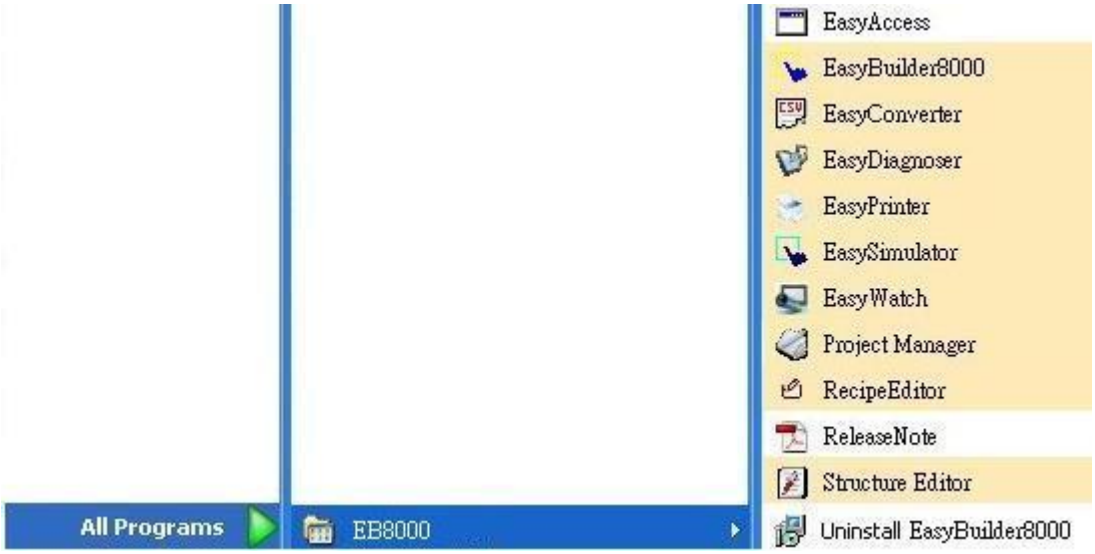


8. Installation processing.



9. Click **[Finish]** to complete the installation.

10. The EasyBuilder shortcuts can be found in **[Start]** » **[All Programs]** » **[EasyBuilder]**.



The description of each item in EasyBuilder menu:

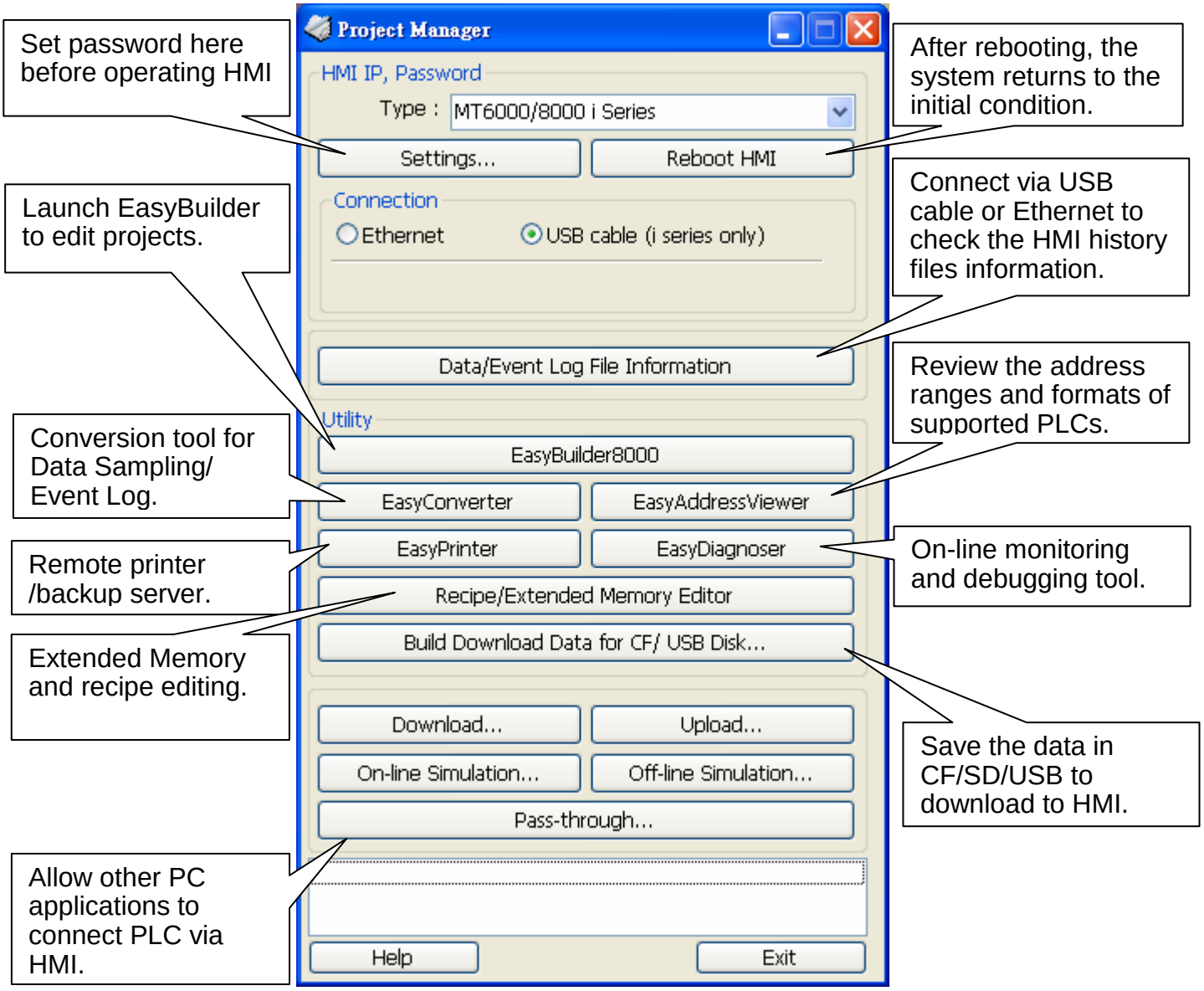
Installed file	Description
EasyAccess	Managing tool for HMIs connected in network.
EasyBuilder 8000	EasyBuilder project editor.
EasyConverter	Conversion tool for Data Sampling and Event Log.
EasyDiagnoser	Tool for detecting the connection between HMI and PLC.
EasyPrinter	Remote screen hardcopy and backup server.
EasySimulator	Execute simulation without installing EasyBuilder.
EasyWatch	Via PC to monitor or set HMI and PLC address value.
Project Manager	EasyBuilder managing tool.
Recipe Editor	Tool for setting the format of Recipe data. Users can open Recipe data or data in the External Memory here.
Release Note	Software release notes.
Structure Editor	Supports AB TAG and improve the flexibility to read / write an object.

Refer to

■ HMI support downloading/uploading projects via USB cable. After installing EasyBuilder, please go to [Computer Management] » [Device Manager] to check if USB driver is installed, if not, please install it manually.

Chapter 2 Project Manager

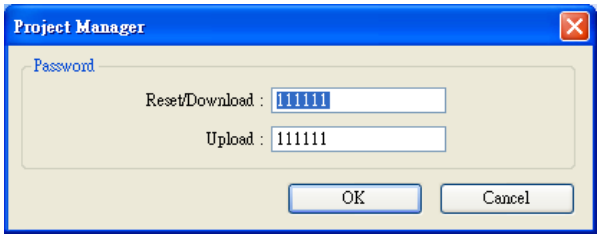
After installing EasyBuilder, double click **[Project Manager]** shortcut on the desktop to start. Project Manager is for launching several utilities and it is a stand-alone program.
To launch EasyWatch, please open EasyBuilder first.



2.1 HMI IP, Password

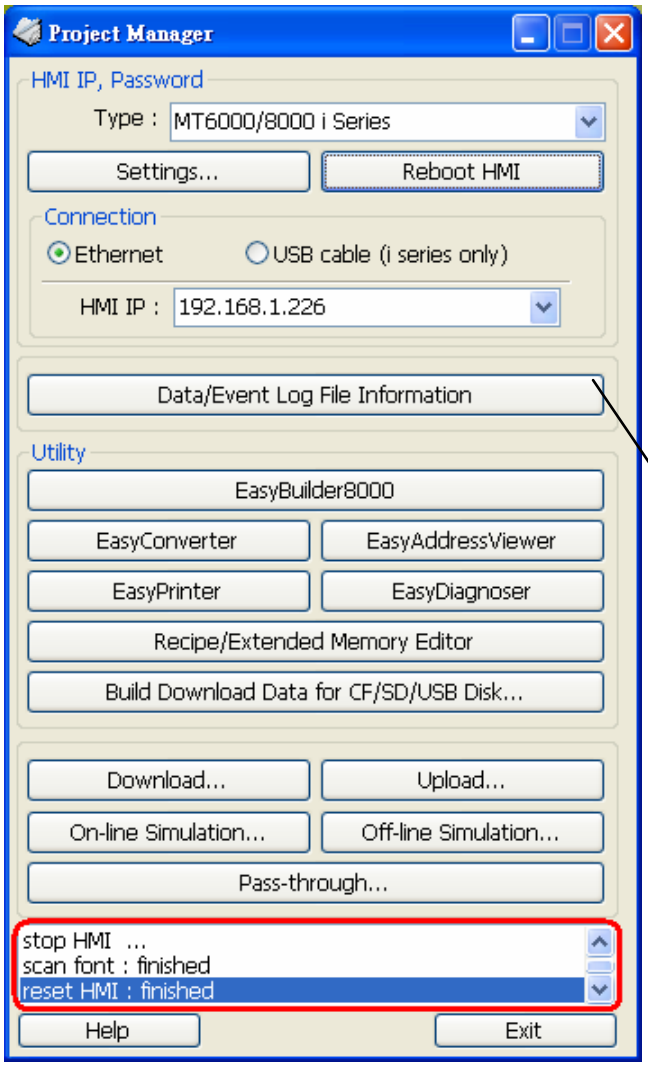
[Settings]

When operating HMI via Ethernet or USB cable, please set the password for HMI to protect against unauthorized access.



[Reset / Download] functions share the same password while **[Upload]** function uses another one.

- Please remember the password, otherwise, while restoring HMI default settings, the project files and data in HMI will be completely erased.

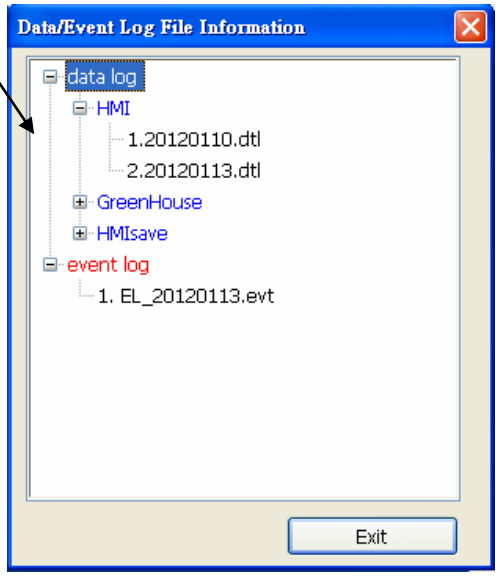


[Reboot HMI]

Reboot the HMI without unplugging. After reboot, the system returns to the initial state. Set the correct IP address when rebooting HMI via Ethernet.

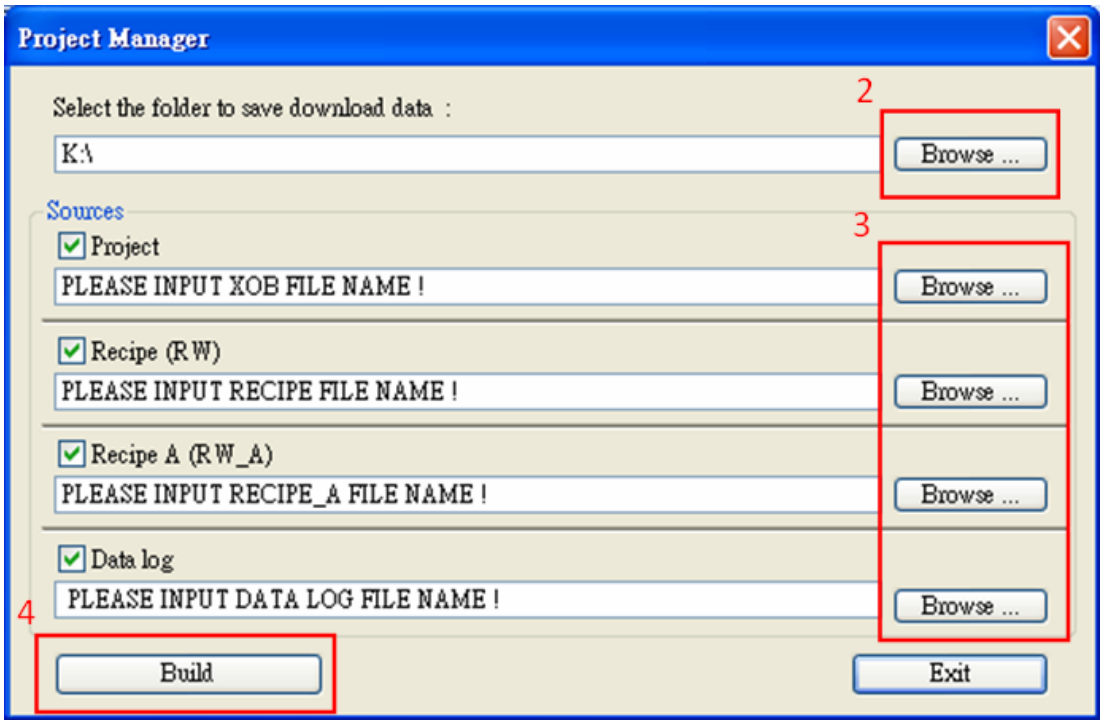
[Data/Event Log File Information]

After setting, connect with HMI to check the number of history files in HMI.



2.2 Editing Tools

2.2.1 Build Download Data for Saving in CF/SD Card or USB Disk



1. Insert the external device (CF/SD card or USB disk) to PC.
2. Assign the folder to store data.
3. Select source files for project, recipe (A) and data log.
4. Select [Build] to create files for downloading.

The source files will be saved in the inserted device for users to download to HMI without connecting via a USB cable or Ethernet.

2.2.2 Steps to Download Project to HMI via USB Disk or CF/SD Card

Assume we will download data in the folder named "**123" (K:\123)** on an USB disk.

1. Insert USB (in which project saved) to HMI.
2. In **[Download / Upload]** dialog box select **[Download]**.
3. Enter Download Password.
4. In **[Download Settings]** dialog box, select **[Download project files]** and **[Download history files]** check boxes.
5. Press **[OK]**.
6. In **[Pick a Directory]** dialog box, select directory: ***usbdisk/device-0/123***.
7. Press **[OK]**.

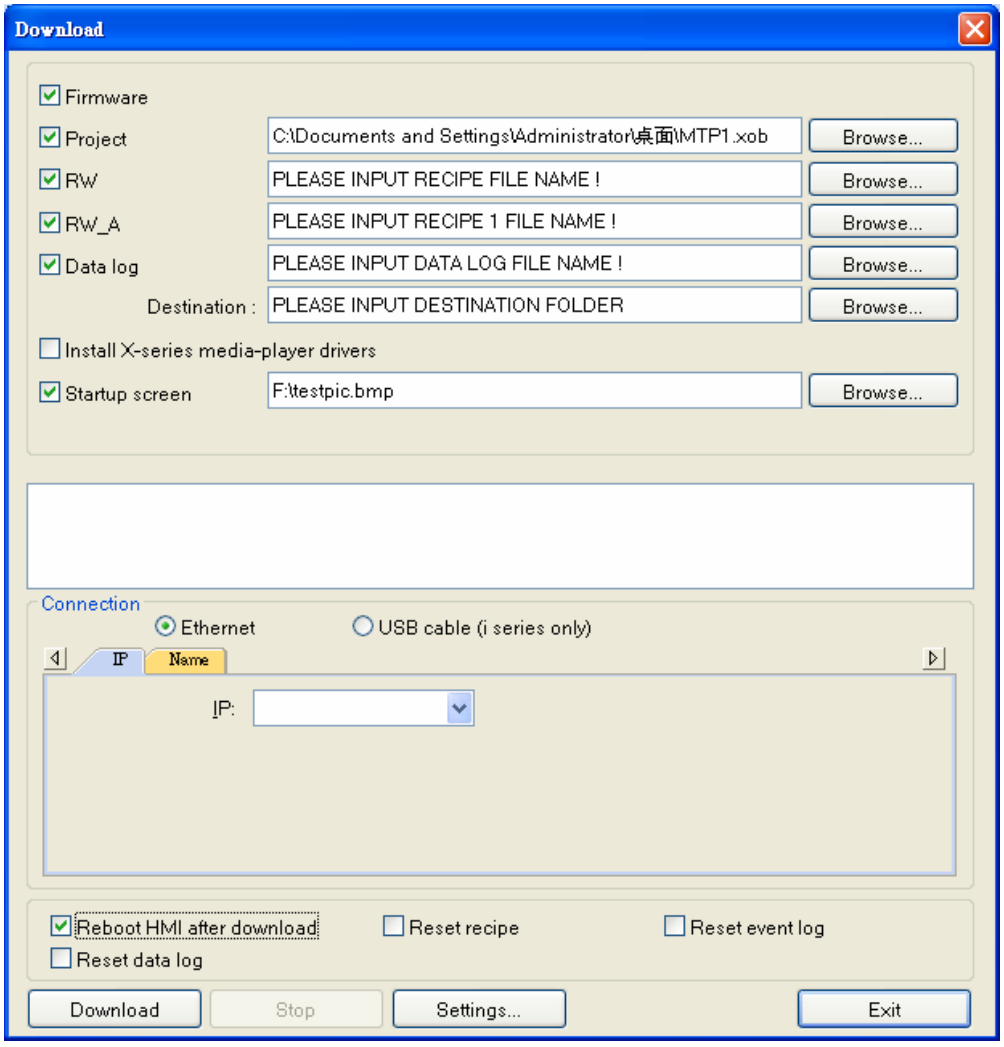
Project will then be updated.

- If only the history files are downloaded, it is necessary to reboot HMI to update files.

2.3 Transfer

2.3.1 Download

Download files to HMI via Ethernet or USB cable.



[Firmware]
Update HMI kernel programs. The firmware must be downloaded at the first time downloading data to HMI.

[Project]
Select an xob file.

[RW / RW_A]
Select a rcp file.

[Data log]

Select a dtl file in datalog folder. Select the data sampling folder in HMI and then select a dtl file.

[Install X-series media-player drivers]

Necessary when first time download data to X series HMI using EasyBuilder8000.

[Startup screen]

Download a bmp bitmap file to HMI. After HMI is rebooted, this bmp file will be shown before project starts.

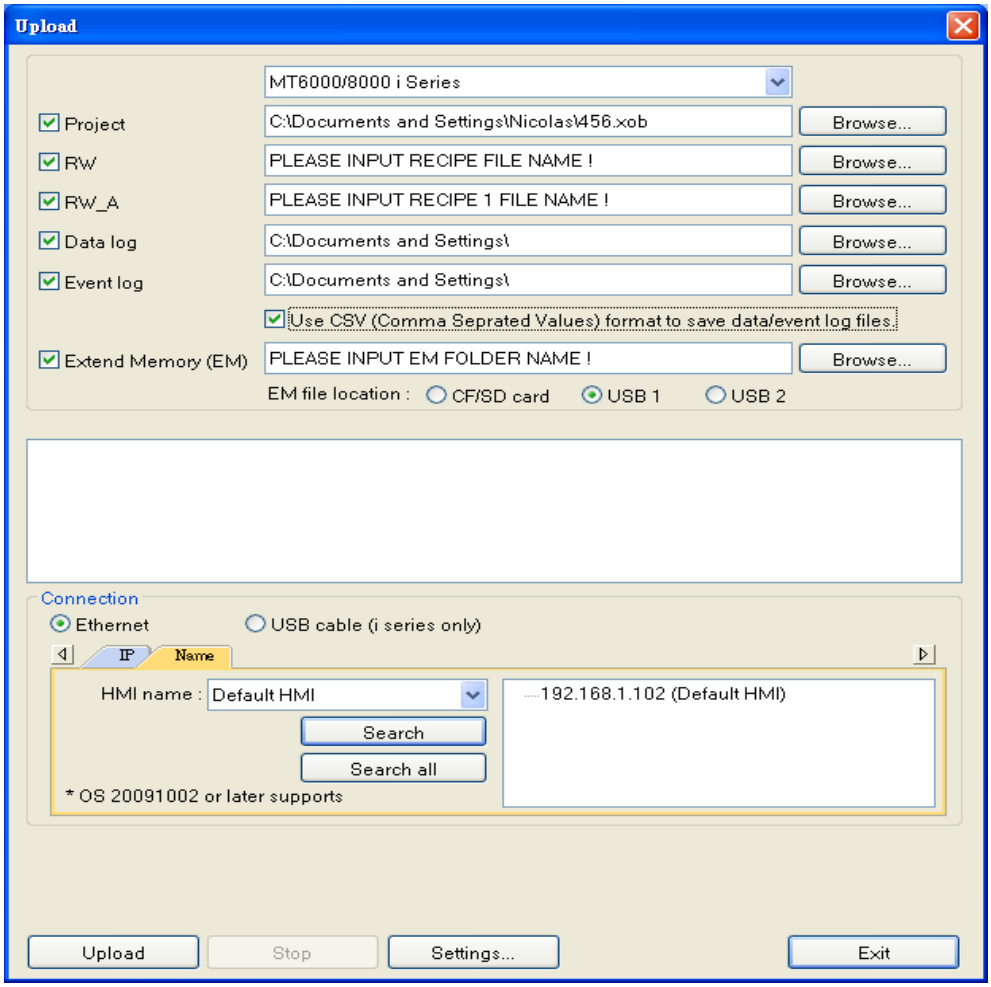
[Reboot HMI after download]

Automatically reboot after download.

[Reset recipe] [Reset event log] [Reset data log] Erase the selected files in HMI before download.

2.3.2 Upload

Upload files from HMI to PC via Ethernet or USB cable.
Click [Browse] and assign the file path before uploading.



For **[Project], [RW / RW_A],** or **[Data log],** please refer to 2.3.1.

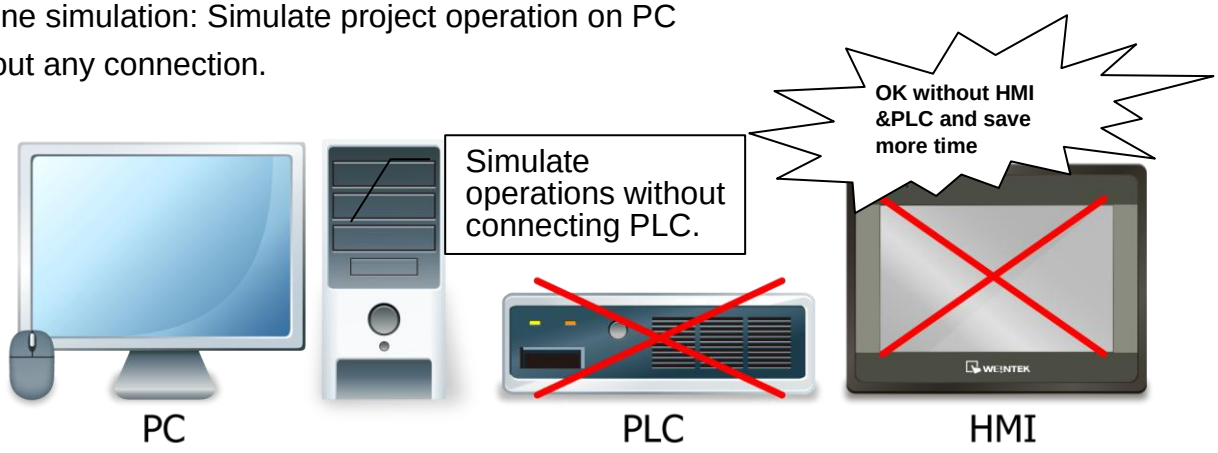
[Event log]
Upload the evt file in HMI to PC.
[Extended Memory (EM)]
Upload the emi file saved in CF/SD card or USB disk to PC.

- The file will be uploaded to PC in xob format. Please decompile it into editable mtp file first and open the mtp file in EasyBuilder.

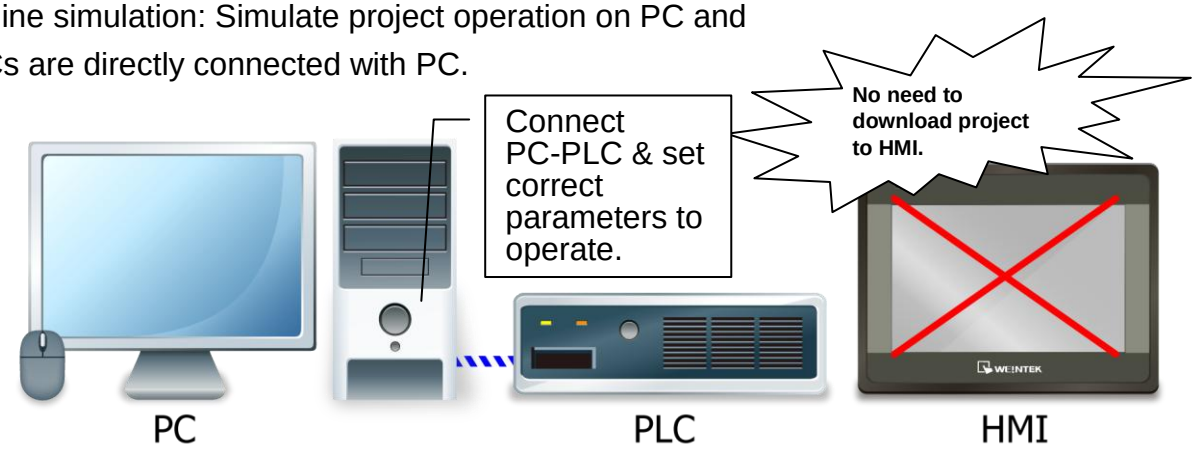
2.4 Simulation

2.4.1 Off-line Simulation / On-line Simulation

Off-line simulation: Simulate project operation on PC without any connection.



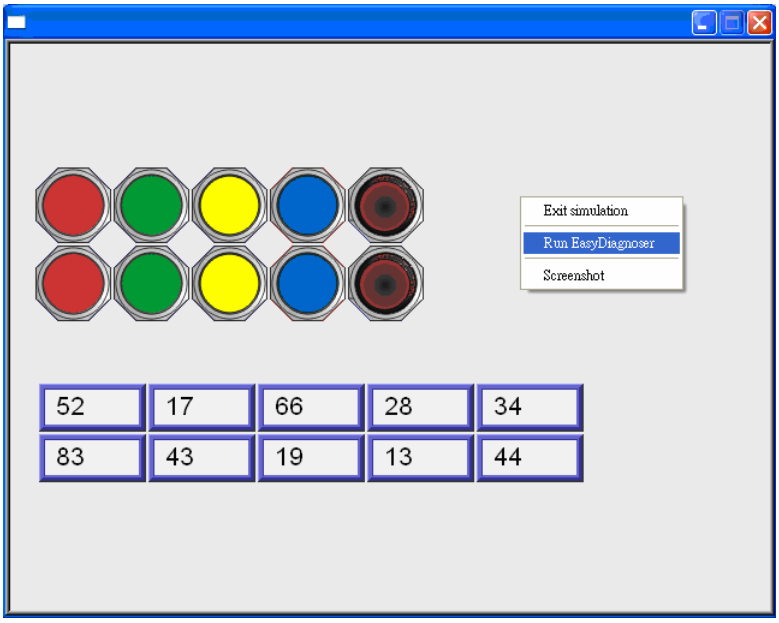
On-line simulation: Simulate project operation on PC and PLCs are directly connected with PC.



- When using **[On-line simulation]** on PC, if the target device is a local PLC (the PLC is directly connected to PC), there is a **10 minutes simulation limit.**

Before executing On-line/Off-line Simulation, please select the source *.xob file.

When executing on-line/off-line simulation, right click to use these functions:



[Exit simulation]
Stop simulating.

[Run EasyDiagnoser]
To monitor current communication status.

[Screenshot]
Capture and save current screen image as a picture file in the screenshot folder under the installation directory.

2.5 Pass-Through

This function allows the PC application to connect PLC via HMI. In this case, the HMI works like a converter.

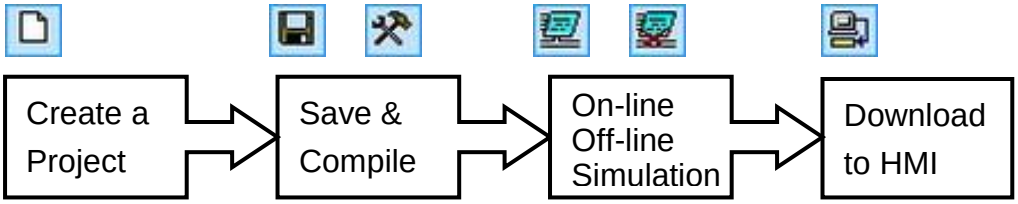
Pass-through provides two modes: **[Ethernet]** and **[COM port]**.

When using **[Ethernet]**, please install the virtual serial port driver first.

For more detail, please refer to “Chapter 29 Pass Through Function”.

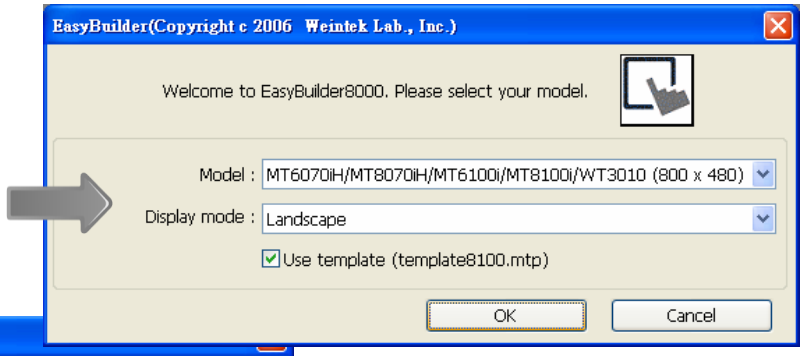
Chapter 3 Create an EasyBuilder Project

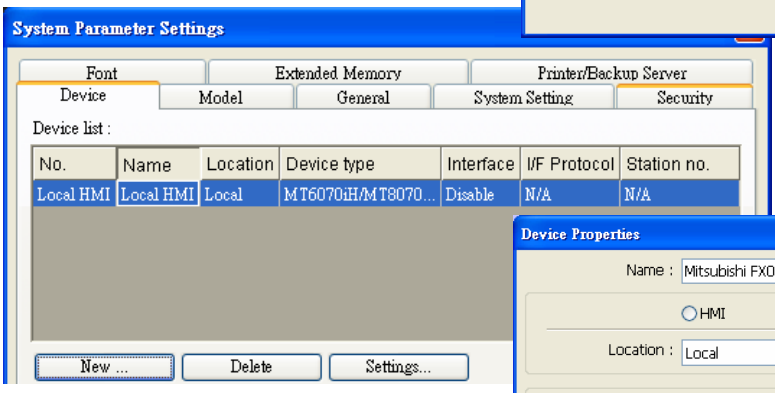
Click on the icons to see illustration.



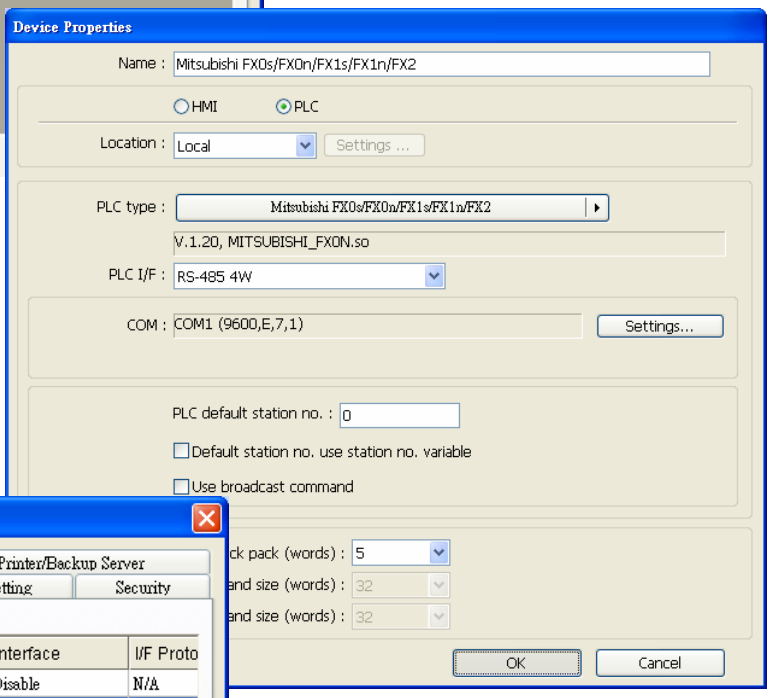
3.1 Create a New Project

1. Click on icon New.
2. Select **[Model]**.
3. Tick **[Use template]**.
4. Click **[OK]**.

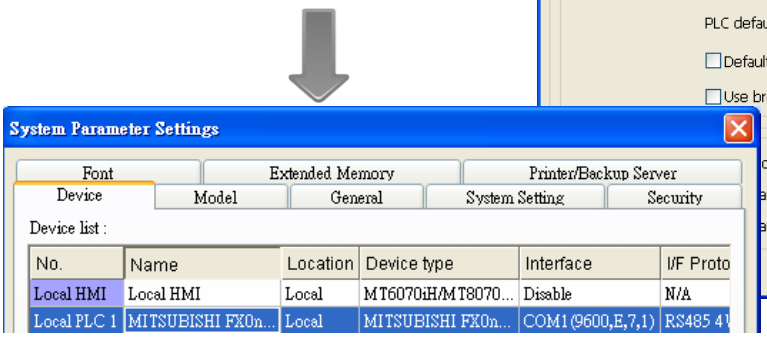




5. Click **[New]**.
6. Set correct parameters.
7. Click **[OK]**.

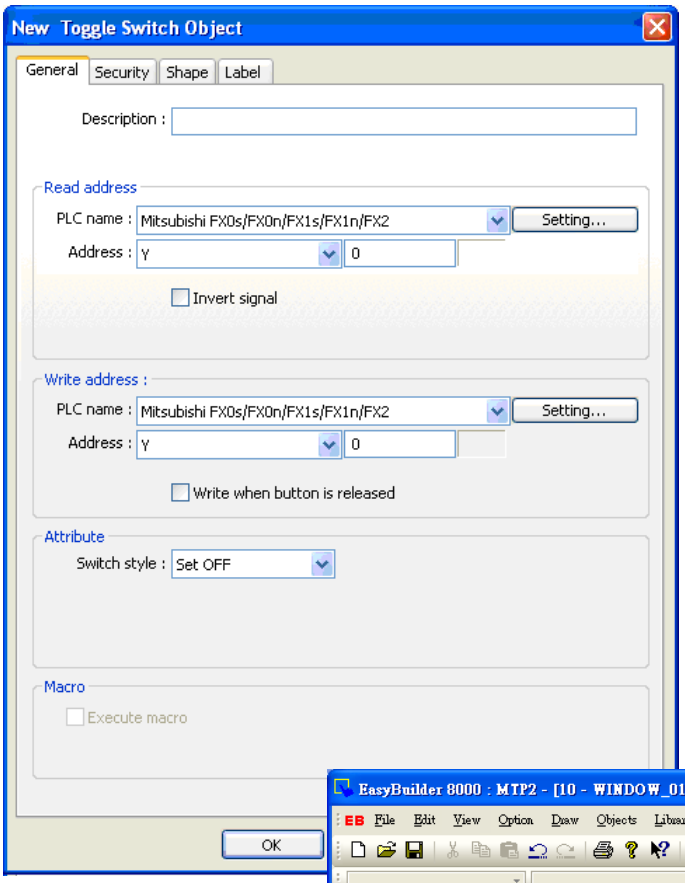


8. Device “MISUBISHI FX0s/FX0n/FX1s/FX2” is added to the **[Device List]**.

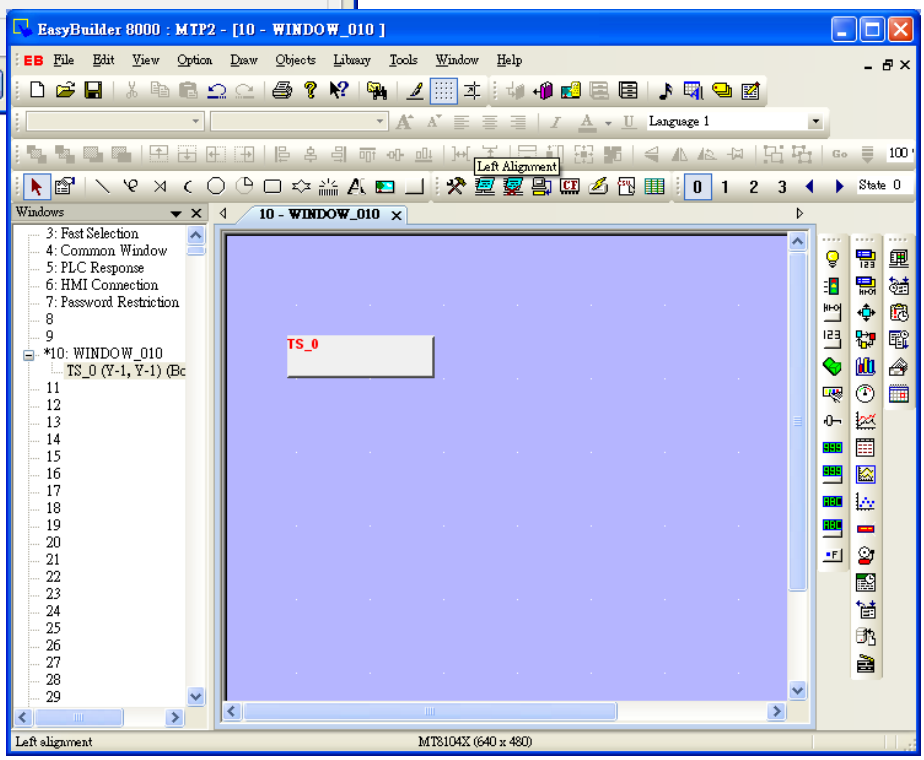


Now let's add a new object.

1. Click on the object icon  Toggle Switch Object.
2. Set correct parameters.

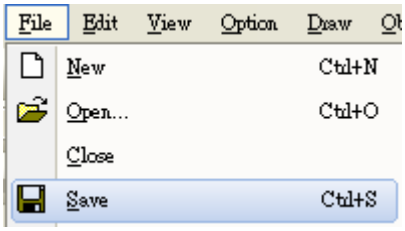


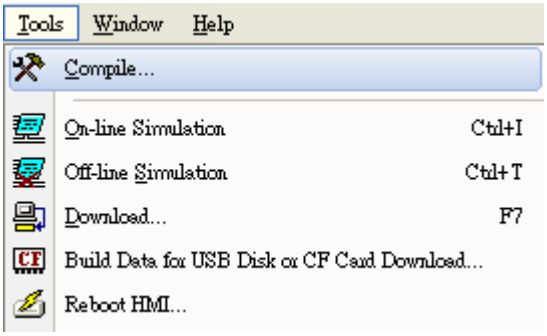
3. Place the object wherever you like on window.
4. A project with one object is now created.



3.2 Save and Compile the Project

On EasyBuilder Tool Bar:

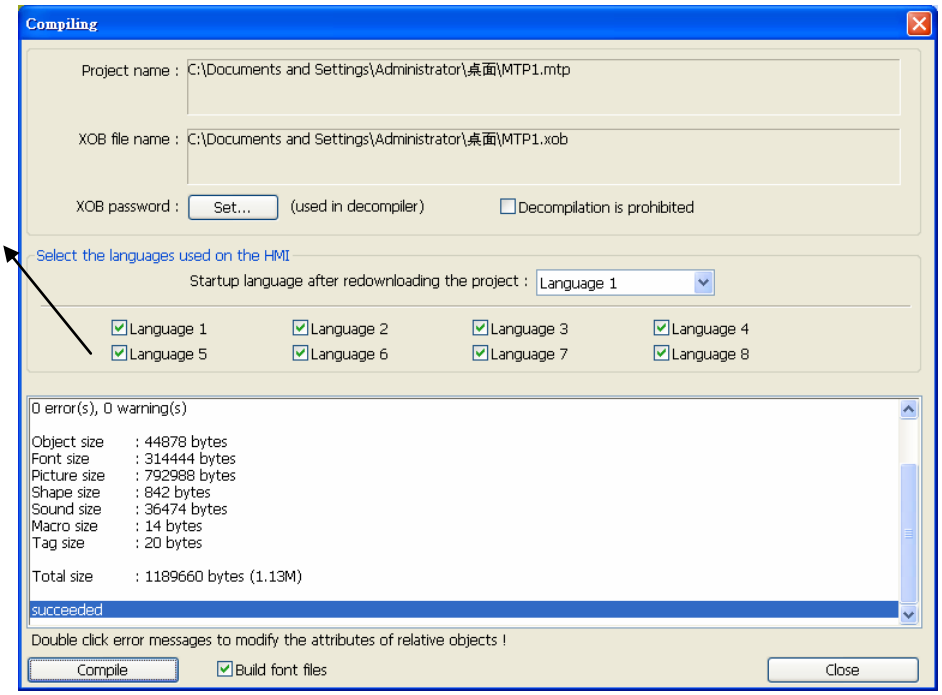




1. Click to **[Save]** *.mtp file.

2. Click to **[Compile]** to *.xobfile for downloading to HMI, this also checks if the project can run correctly.

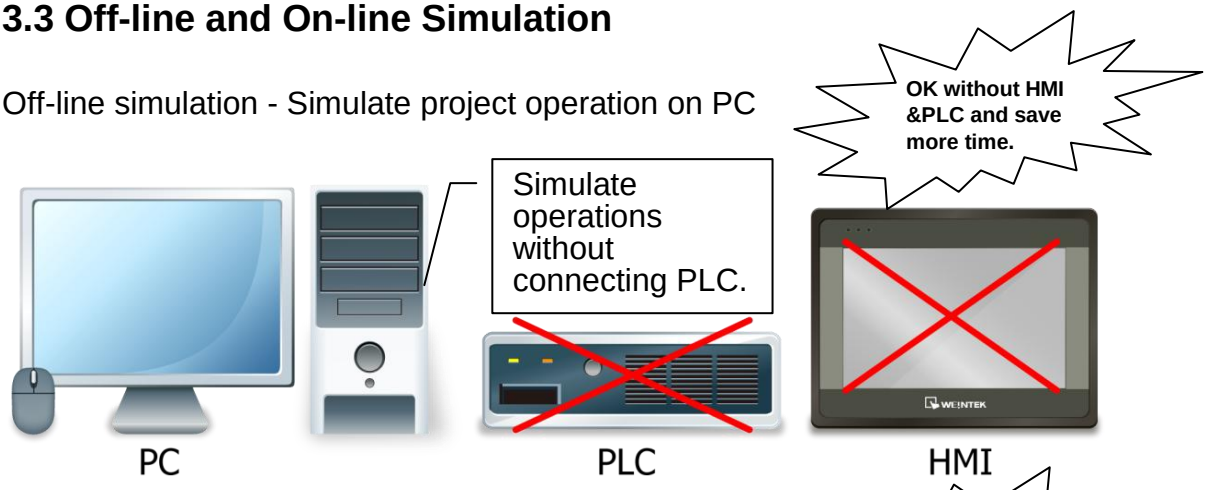
Users are allowed to select the languages needed for the project and download to HMI, up to 8 languages can be selected.



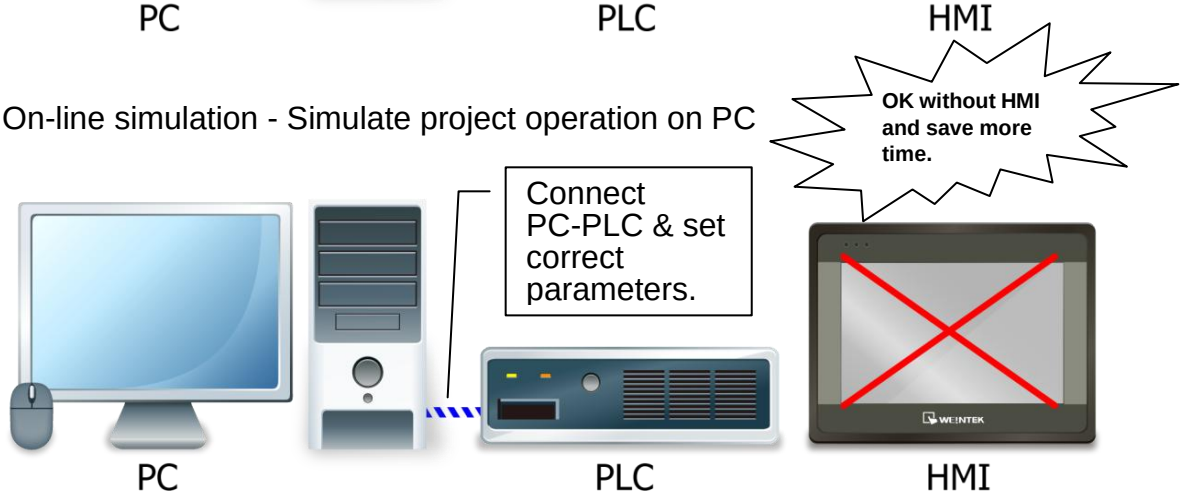
A successfully compiled file will get this dialog box.

3.3 Off-line and On-line Simulation

Off-line simulation - Simulate project operation on PC



On-line simulation - Simulate project operation on PC

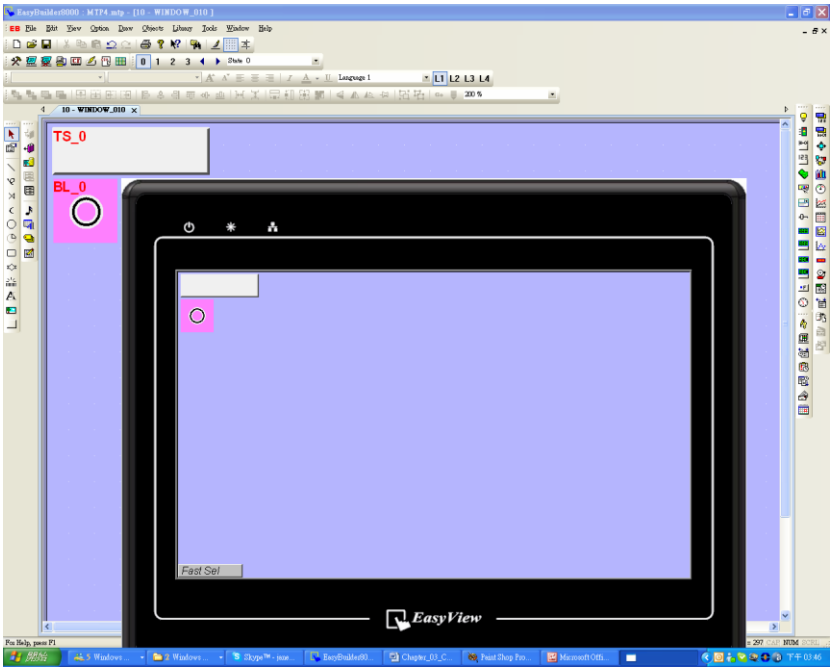


When On-line simulating on PC, if the control target is a local PLC (i.e. the PLC directly connected to PC), there is **10 minutes simulation limit.**

Off-line

On-line

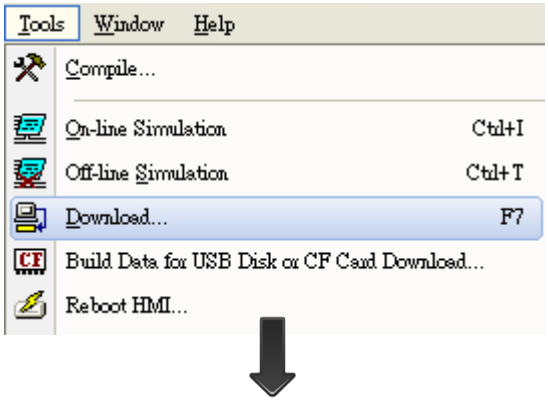
Click after correctly connecting the device.



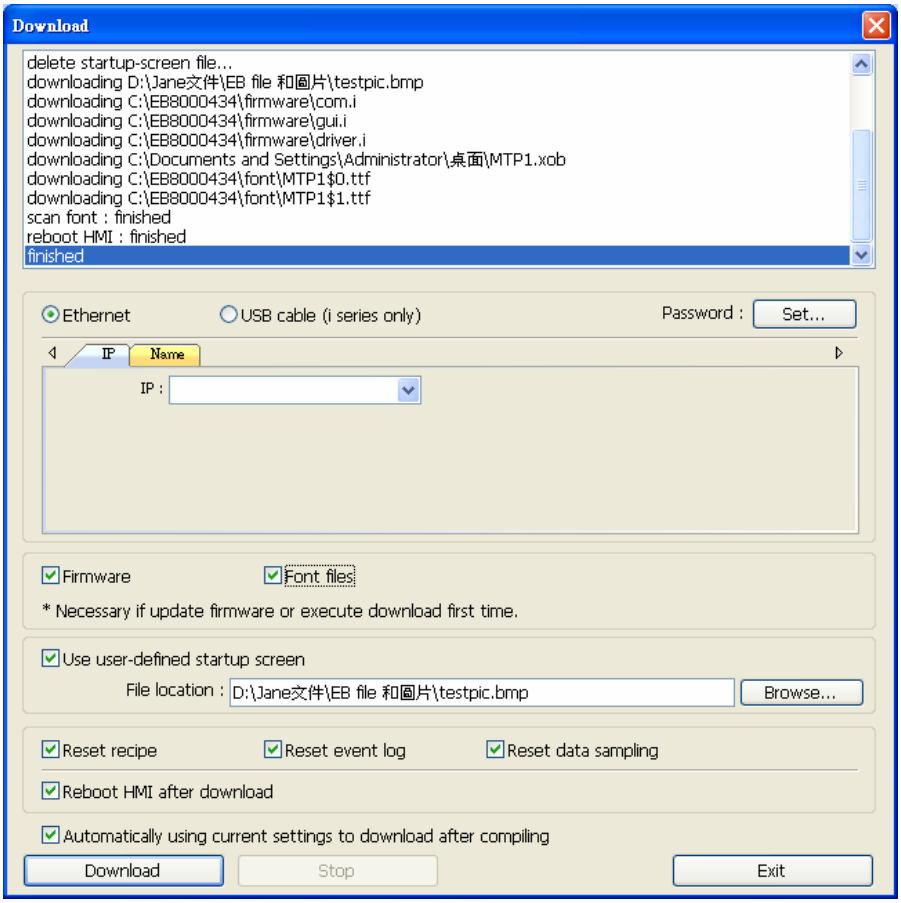
3.4 Download the Project to HMI

■ Way 1 [Ethernet] / HMI IP

Before **[Download]**, be sure to check if all the settings are correct.



Input **[Password]** & Specify **[HMI IP]**.



Firmware

Update HMI kernel programs. Must do this when first time download files to HMI.

Font files

Download the font used in project.

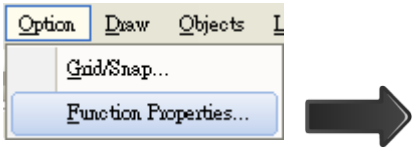
The selected files will be erased before downloading.

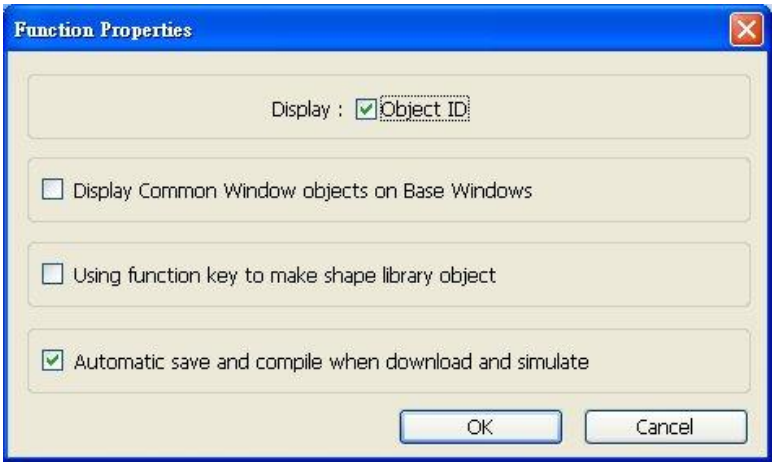
Reboot HMI after download HMI will reboot after downloading.

Automatically using current settings to download after compiling

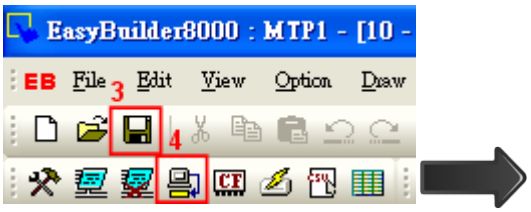
If this is checked, system will download project to HMI according to last settings. Please see illustration below.

The way to enable this function:

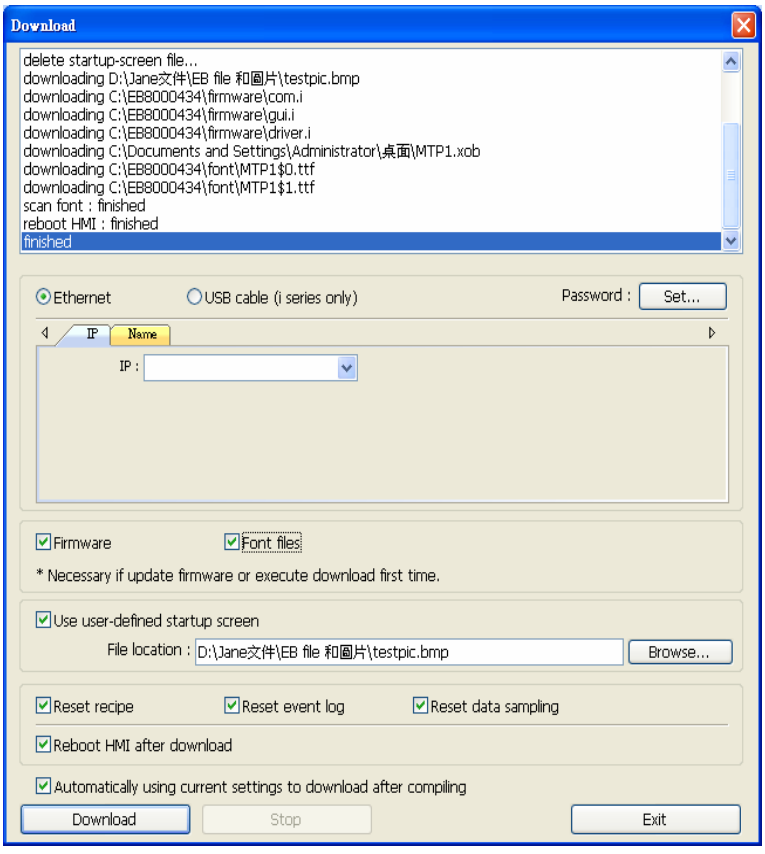




1. Click **[Function Properties]**.
2. Tick **[Automatic save and compile when download and simulate]**.

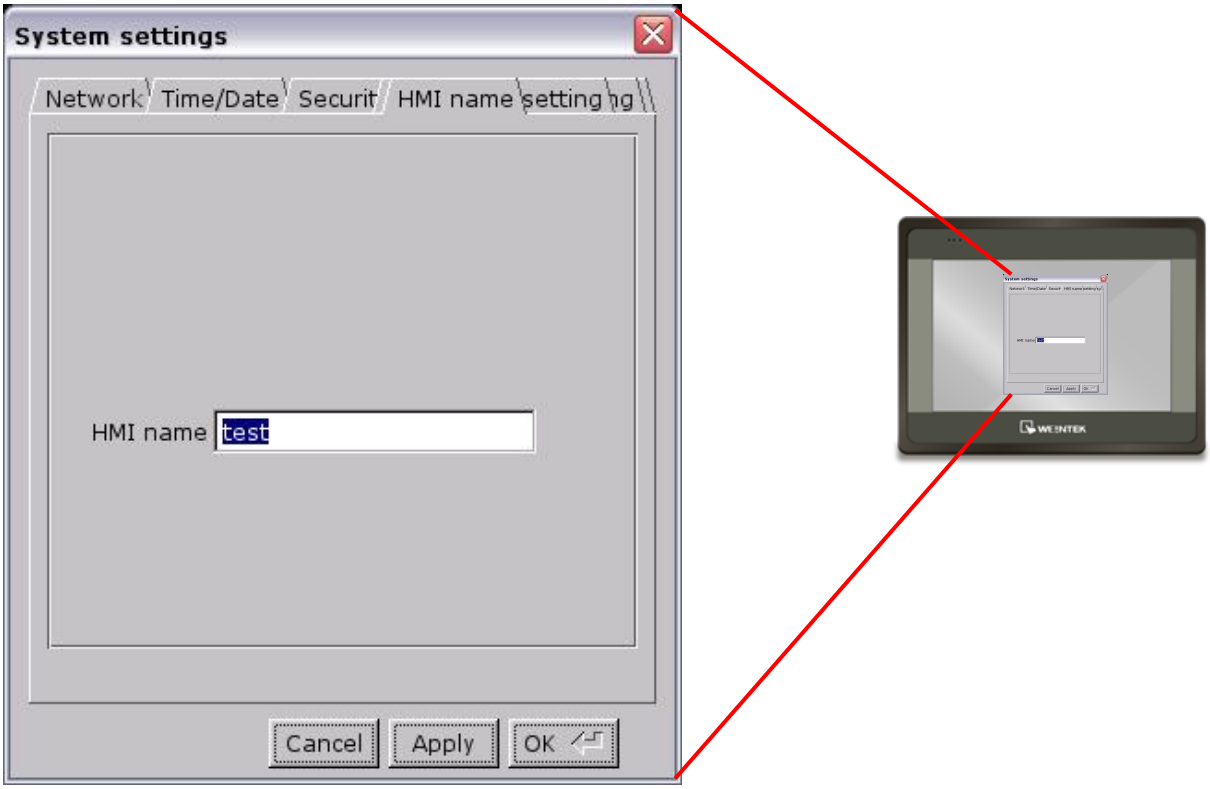


3. **[Save]** project.
4. Click **[Download]**.
5. On dialog box, tick **[Automatically using current settings to download after compiling]**.
6. Click **[Download]**.
7. After finish setting, next time when click **[Download]**, EasyBuilder will automatically compile and download project to the latest target HMI.

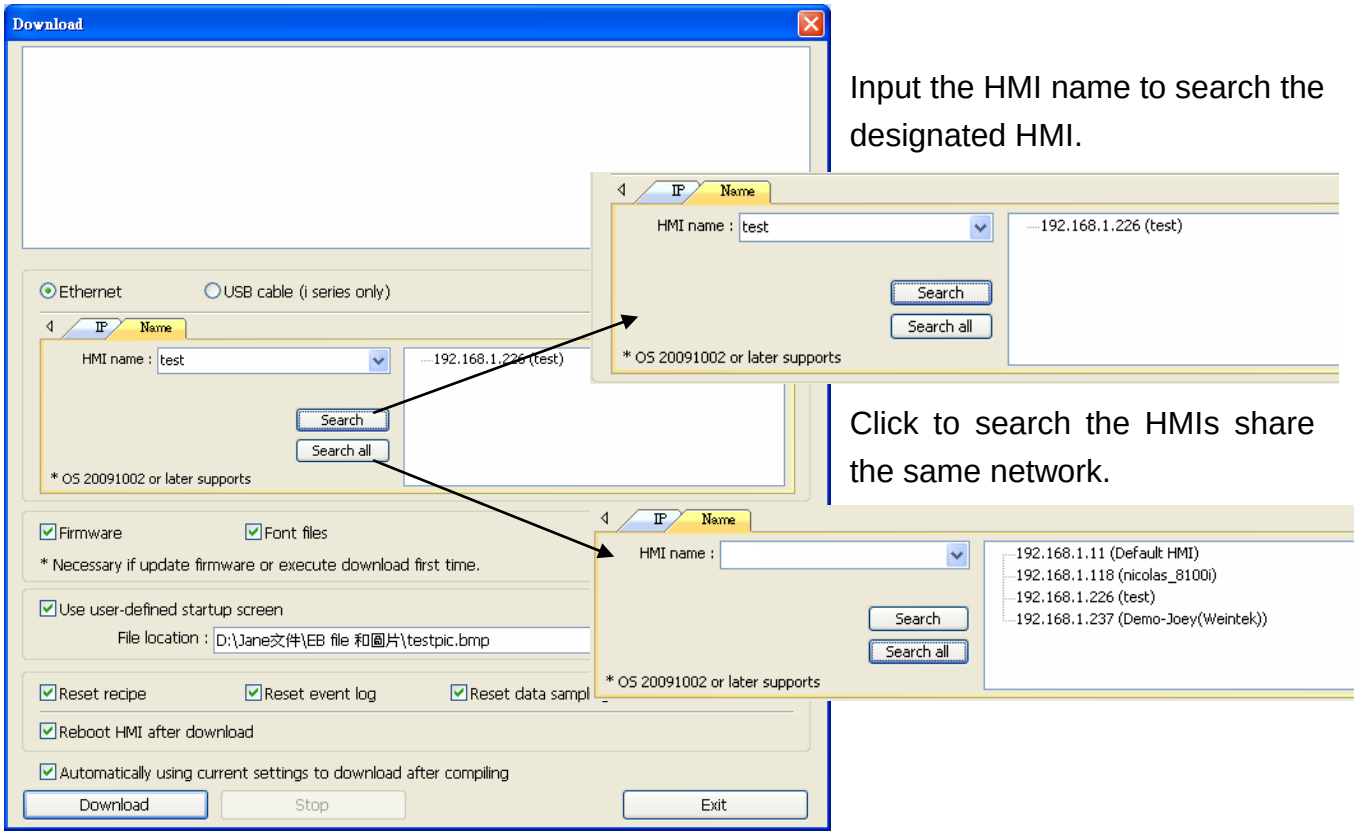


■ Way 2 [Ethernet] / HMI Name

1. On HMI set HMI name first.



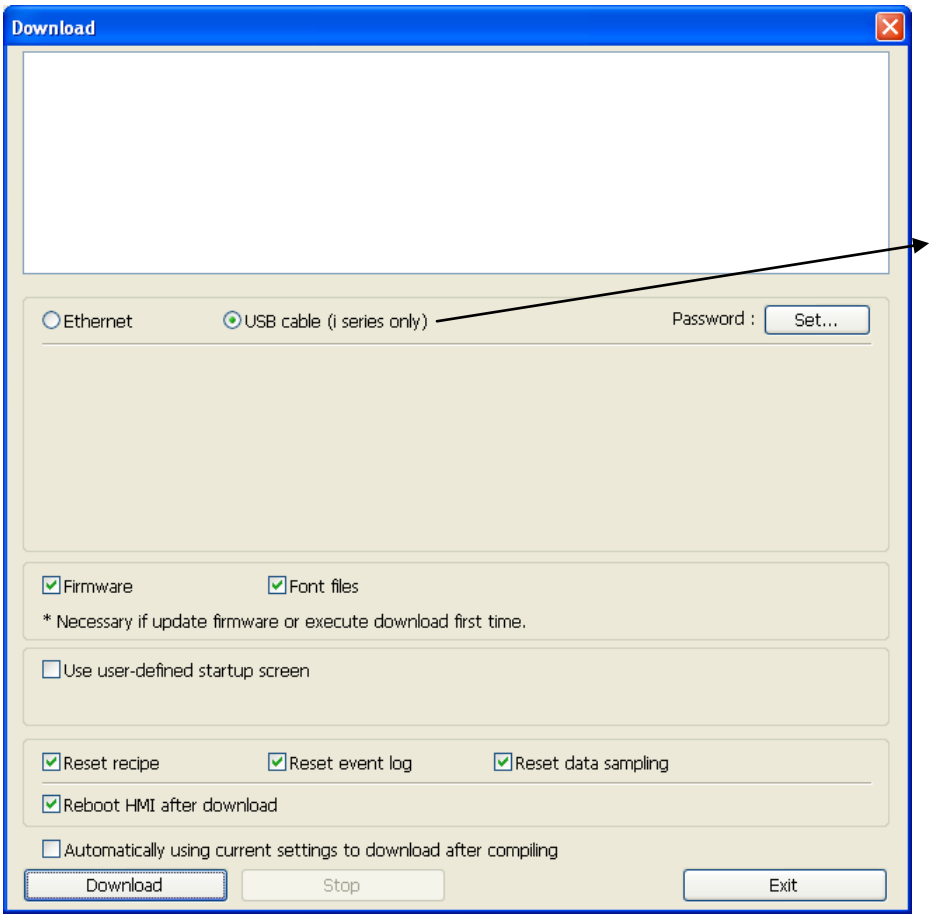
2. On PC, select the set HMI name and start downloading.



Input the HMI name to search the designated HMI.

Click to search the HMIs share the same network.

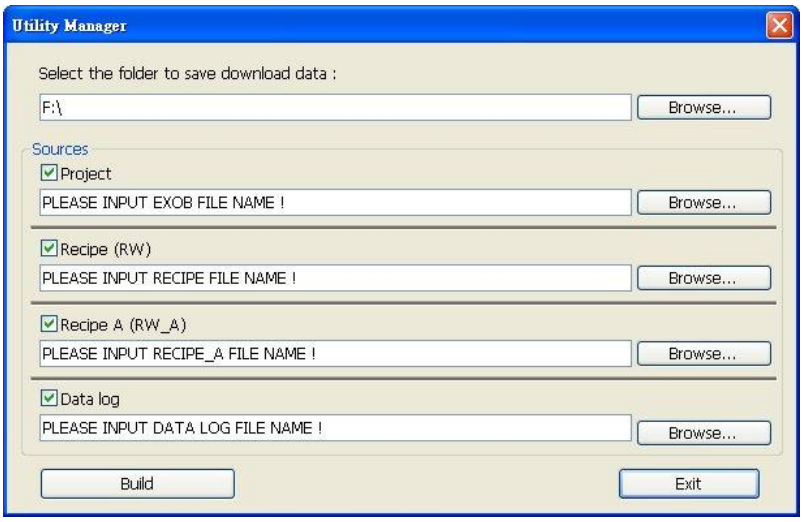
■ Way 3 [USB Cable]



Select USB cable to download project to HMI. The way of setting is same as Way 1 mentioned above. USB cable only works for i Series HMI.

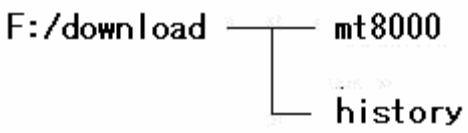
■ Before downloading via USB cable, please make sure the USB driver is correctly installed. Go to [Computer Management] / [Device Manager] to check if USB driver is installed, if not, please refer to installation steps to manually install.

■ Way 4 [USB Disk / SD Card]

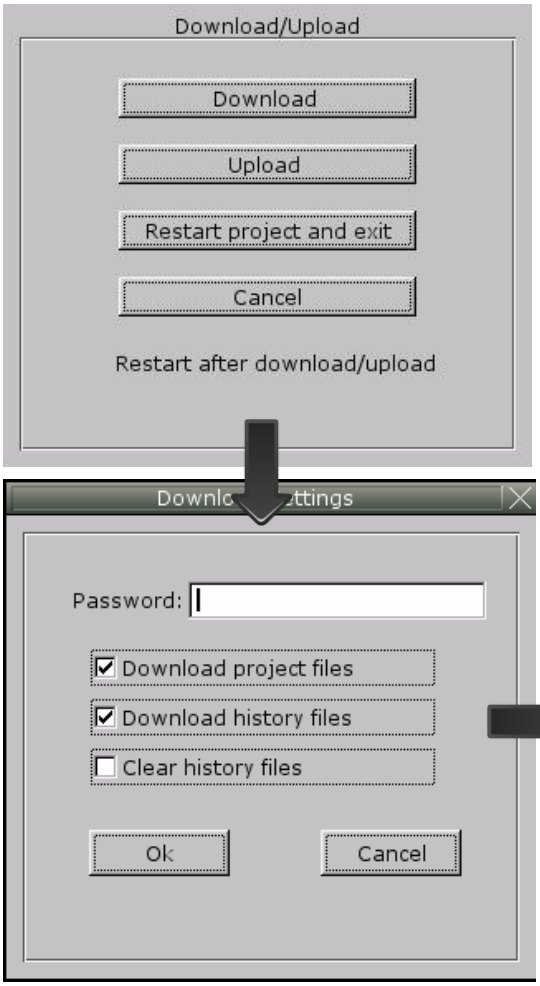


1. In Utility Manager click **[Build Download Data for CF / SD / USB Disk]** to build the data to be downloaded first. Generally divided into 2 directories, if set as the way shown:

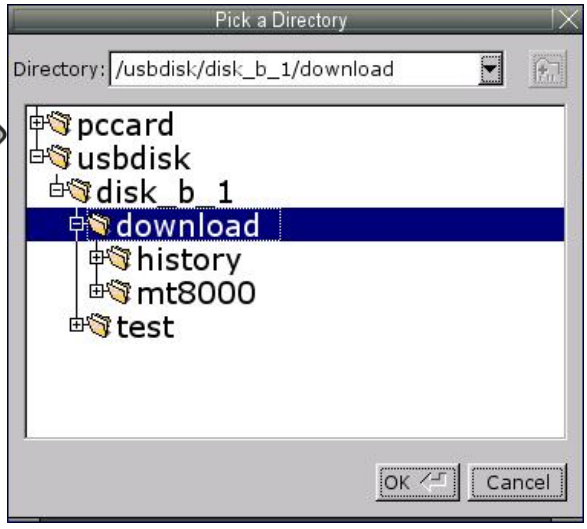
The download data storing structure:



This directory is generated when downloading history data.



2. Insert external devices to HMI.
3. Select **[Download]** and input correct password.
4. Password confirmed, show directories in external device.(pccard: SD/CF Card；usbdisk: USB Disk)
5. Select a directory for storing project then click **[OK]** to start downloading.

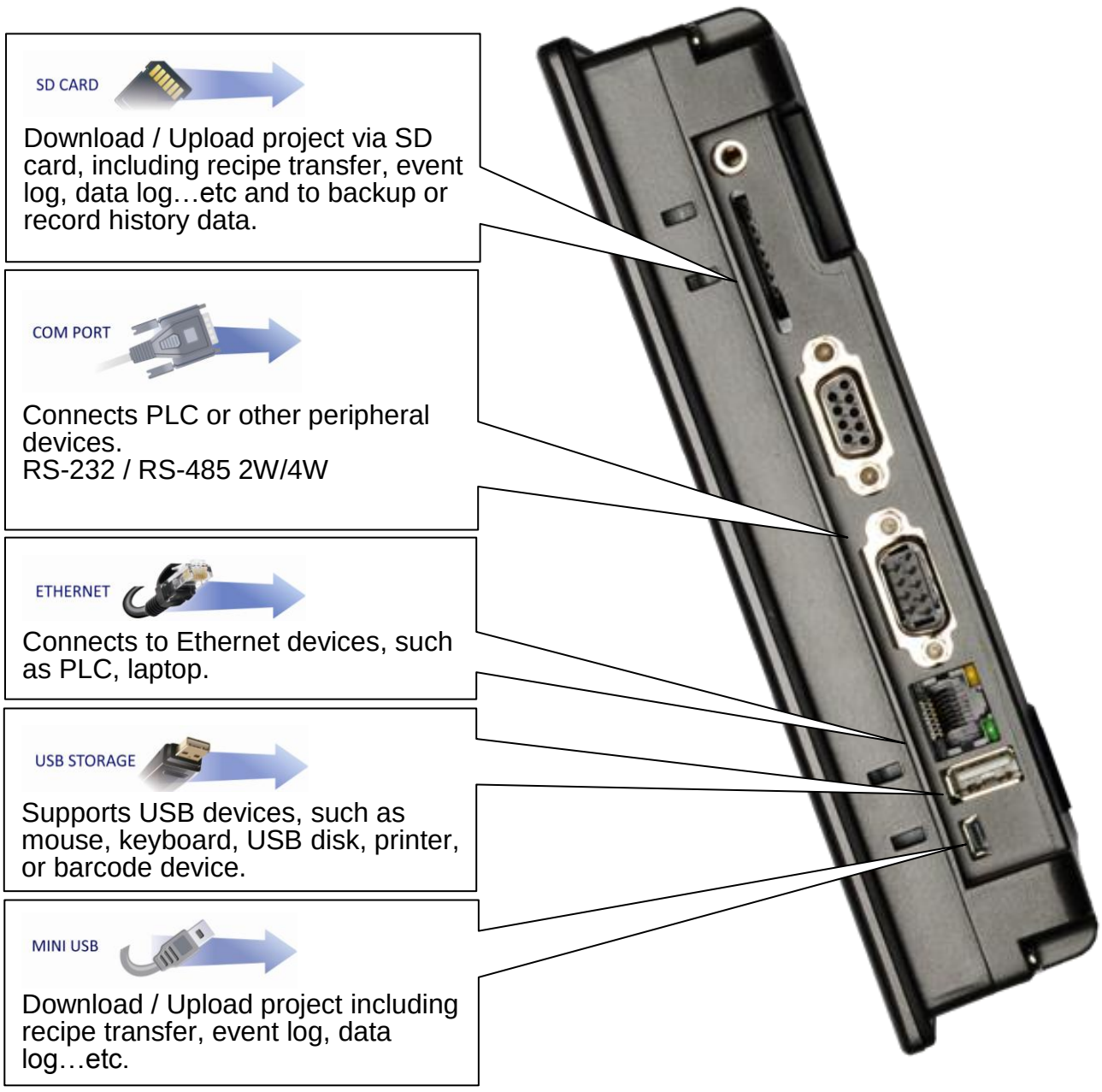


Note Please select **the top layer directory of the target file** when downloading. For the structure above, select ***download***, not ***mt8000*** or ***history*****.**

Chapter 4 Hardware Settings

4.1 I/O Ports

The I/O ports are different from one HMI type to another.



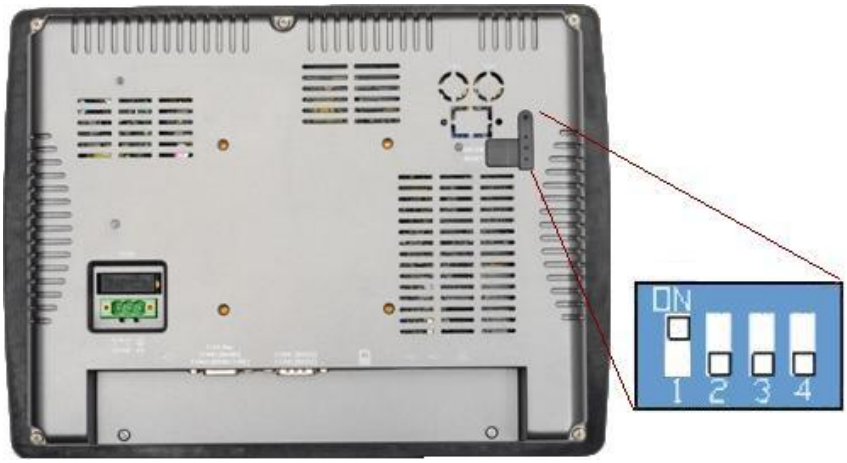
In addition, Weintek provides [MT8 – Multi-Connector Cable] to expand the COM port for easier operation.

4.2 System Settings

For the first time operating HMI, please complete the following system settings. When finished, the project files designed using EasyBuilder can be used on HMI.

4.2.1 System Reset

Each HMI is equipped with a reset button and a DIP switch. When using DIP switch to change modes, the corresponding functions will be triggered.

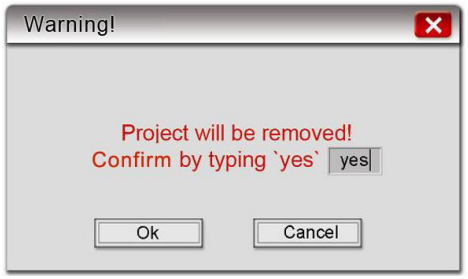


If system password is lost or forgotten, please flip DIP Switch 1 to ON and the rest to OFF, and then reboot HMI.

HMI will switch to touch screen calibration mode.

1. A “+” sign appears on the screen, touch the center of the sign, after all 5 signs are touched, “+” disappears and the touch screen parameters will be stored in HMI system.

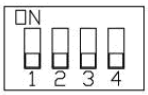
2. After calibration, confirm to restore the default password, select **[Yes]**.



3. Confirm to restore the default password again by typing **[yes]** and clicking **[OK]**. The project files and history records stored in HMI will all be removed. (The default local password is 111111. However, other passwords, such as download/upload passwords have to be reset.)

The above shows the steps to restore factory settings of T and i Series HMI. For X Series, users will need a connected USB keyboard, and press any key (or space key) right when the first image displayed as HMI power ON to enter the menu. Select "Factory Mode", the window mentioned will pop up when system displays project. In case users may miss the very first image shown, to press space key continuously since HMI power ON will ensure entering the system setting window.

Dip Switch



SW1	SW2	SW3	SW4	Mode
ON	OFF	OFF	OFF	Touch screen calibration mode (X Series excluded)
OFF	ON	OFF	OFF	Hide system toolbar (T Series excluded)
OFF	OFF	ON	OFF	Boot loader mode (X Series excluded)
OFF	OFF	OFF	ON	Enable front panel power switch (X Series only)
OFF	OFF	OFF	OFF	Normal

- Dip Switch 4 is set ON or OFF according to the LCD used. If it should be ON when out from factory, the Dip Switch 4 would be set ON and cut off. If it should be OFF, the Dip Switch 4 would be set OFF but the switch is not cut.

4.2.2 System Toolbar

After rebooting HMI, users can set the system with [System Toolbar] at the bottom of the screen. Normally, this bar is hidden automatically. Only by touching the target at the bottom-right corner of the screen will the System Toolbar pop up.

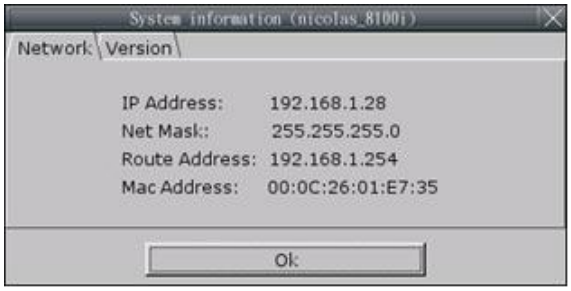
Screen Calibration shortcut, X Series only, for other series, turn SW1 to ON. When X Series touch screen drifting problem occurs, please connect an USB mouse to select this mode.

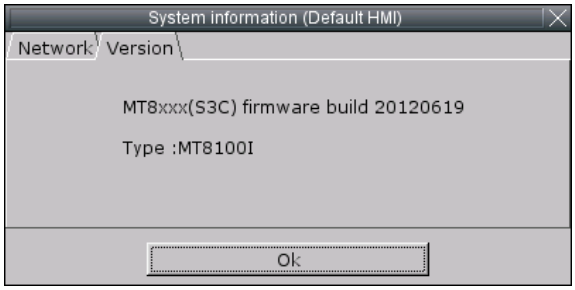
■ How to hide HMI System Setting Toolbar

When [DIP Switch 2] is set ON, the system setting toolbar is disabled. When set OFF; the system setting toolbar is enabled. Please restart HMI to enable/disable the toolbar.
System register [LB-9020] can also enable/disable system setting toolbar
When [LB-9020] is set ON, the toolbar is displayed, and set OFF to hide the toolbar.
Note: [LB-9020] is available for all HMI series. [DIP Switch 2] is available for i and X Series.

4.2.3 System Information

Network: Displays network information & HMI IP. **Version:** Displays HMI system version.





4.2.4 System Setting

Set or modify system parameters.
Confirm password for security.

■ Network

Download project to HMI via Ethernet.
Confirm IP address of target HMI.
[Obtain an IP Address Automatically] or [IP address get from below].

■ Security

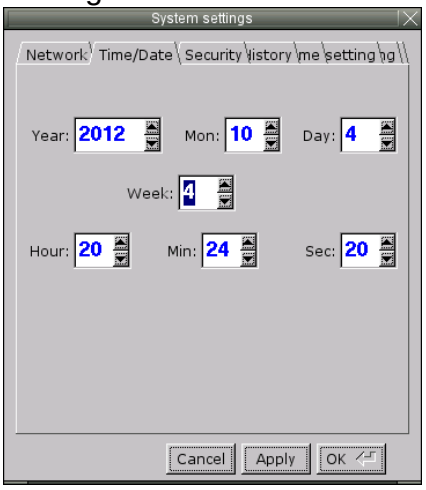
Password protection, the default is 111111.

[Password for entering system]
[Password for uploading project]
[Password for downloading project]
[Password for uploading history data]

Password confirmation dialog:

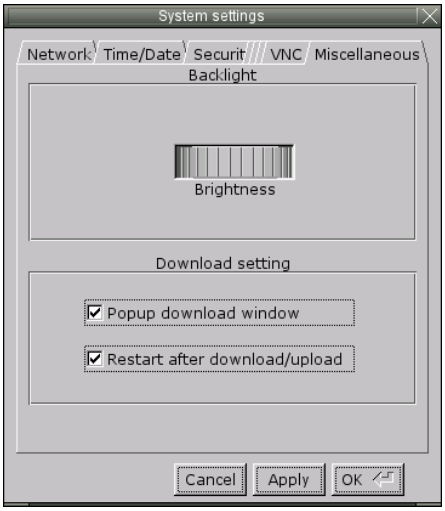
■ Time/Date

Setting HMI local time/date.



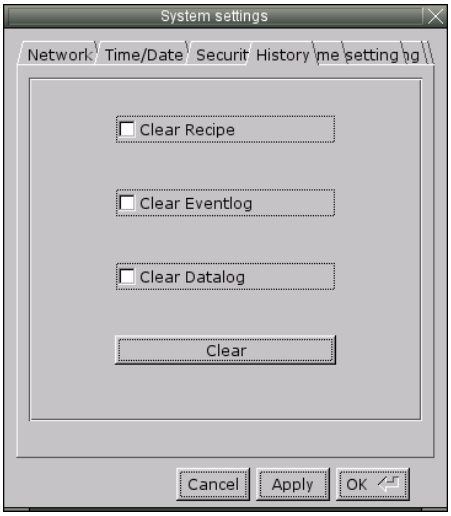
■ Miscellaneous

Rotary switch for adjusting LCD brightness.



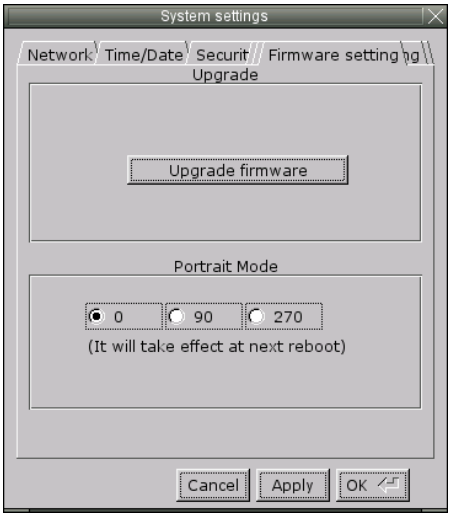
■ History

Clears history data in HMI.
[Recipe] / [Eventlog] / [Datalog]



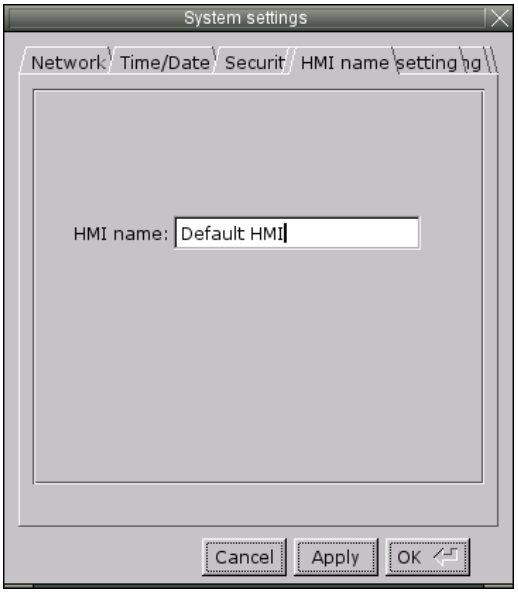
■ Firmware setting

Upgrades firmware / enable portrait mode.



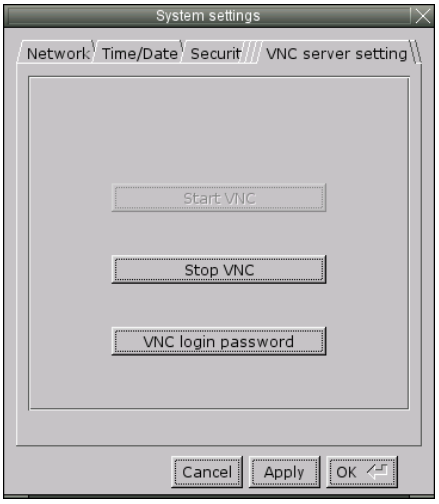
■ HMI name

Set HMI name to download/upload project.



■ VNC server

Remote HMI monitoring and controlling via Ethernet.

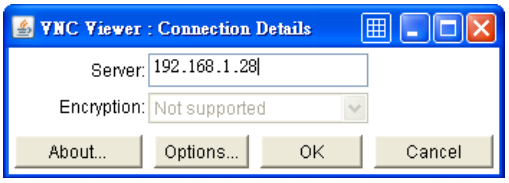


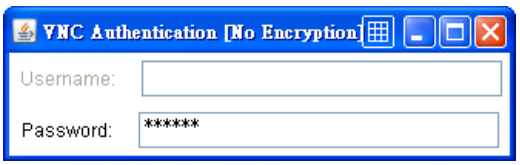
Step 1. Enable HMI VNC server, set password.

Step 2. Install Java IE or VNC Viewer on PC.

Step 3-1 Input remote HMI IP in Java IE, example: http://192.168.1.28

Step 3-2 In VNC Viewer input remote HMI IP and password.



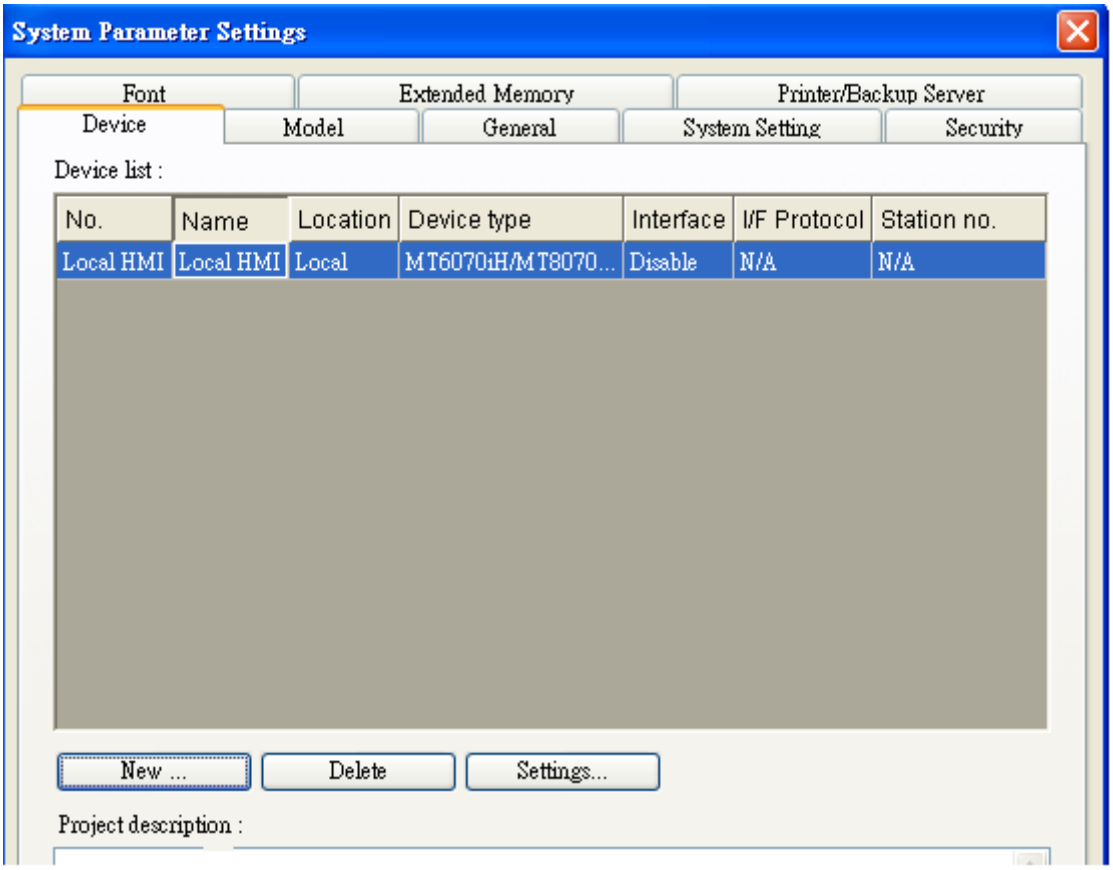


Note

One HMI allows only one user to log in VNC server at one time. If there is no activity for more than one hour, HMI system will log out automatically.

Chapter 5 System Parameter Settings

Launch EasyBuilder, in the main menu select **[Edit] » [System Parameters]** to open the **[System Parameter Settings]** dialog:

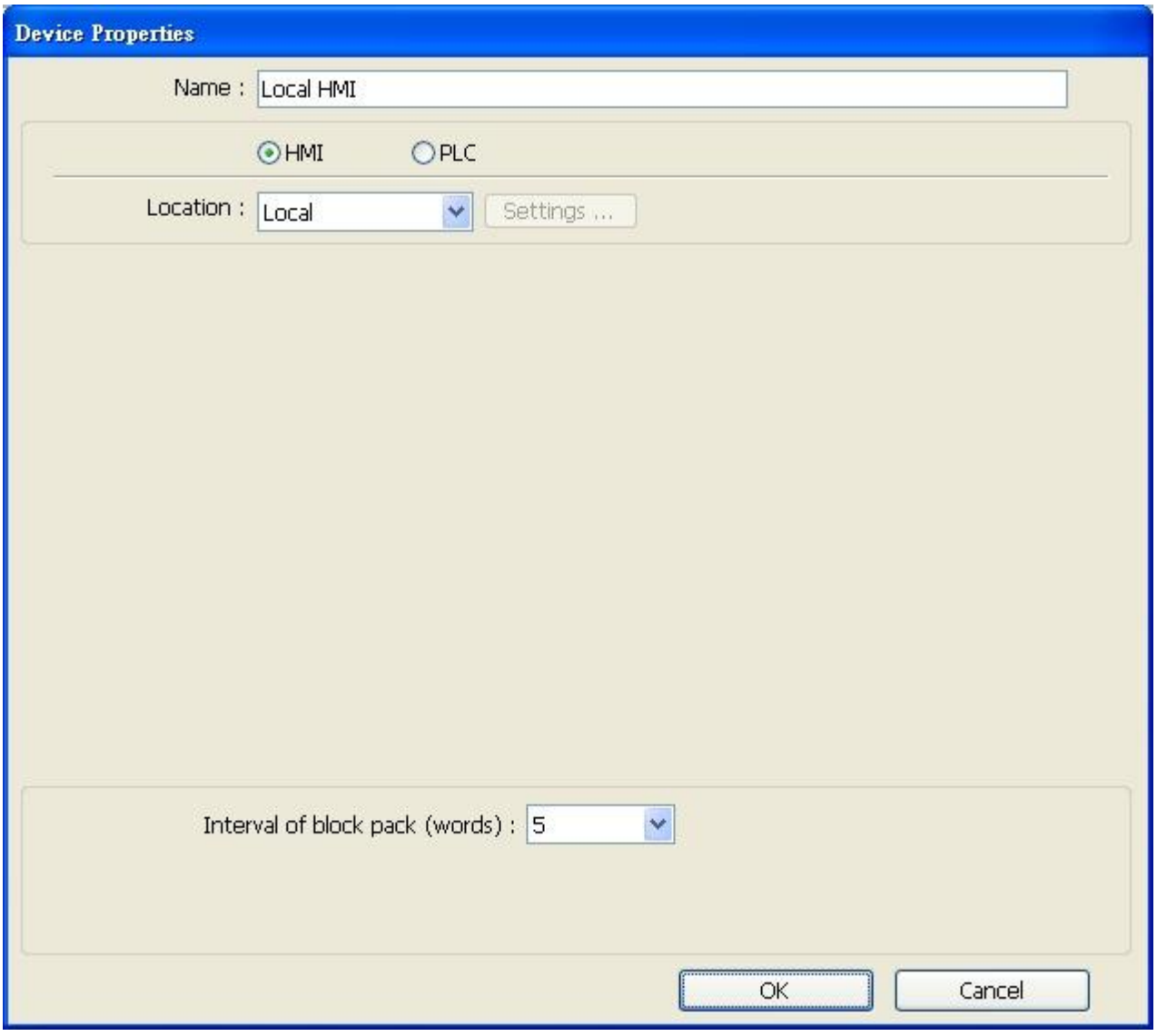


System Parameter Settings are divided into several parts: **[Device]**, **[Model]**, **[General]**, **[System Setting]**, **[Security]**, **[Font]**, **[Extended Memory]**, and **[Printer/Backup Server]**. These will be introduced respectively in this chapter.

5.1 Device

Parameters in this tab determine the attributes of each device connected with HMI. The device can be a Local / Remote HMI / PLC.

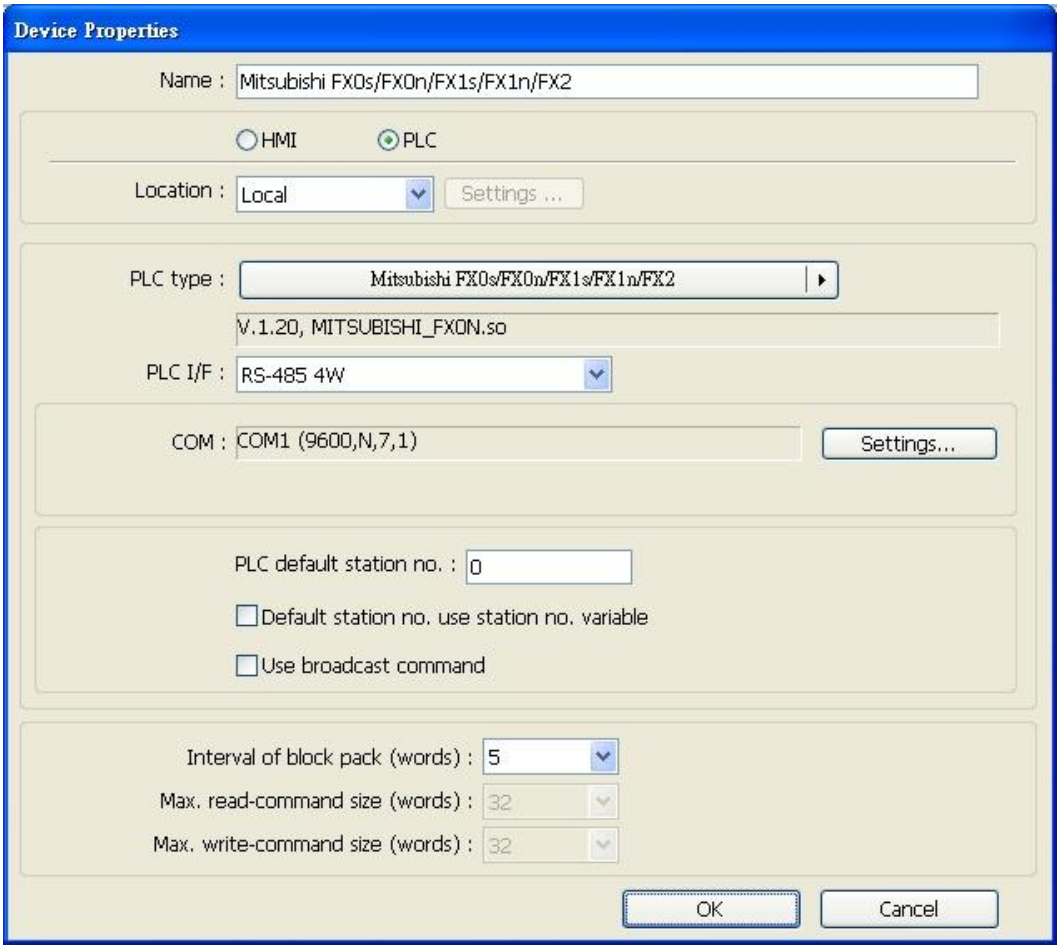
When creating a new project file, there is a default device "Local HMI" which indicates the HMI that will be updated and programmed. To change the relevant device settings, click **[System Parameter Settings] » [Settings]** to open **[Device Properties]** dialog as shown below:



5.1.1 How to Control a Local PLC

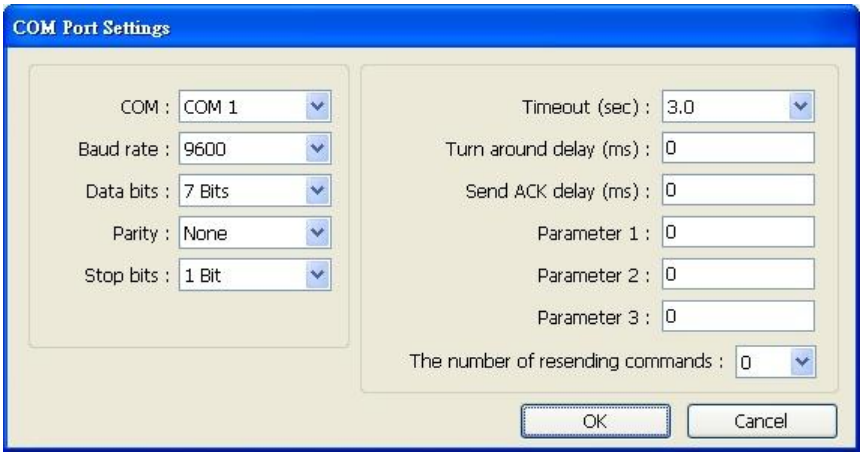
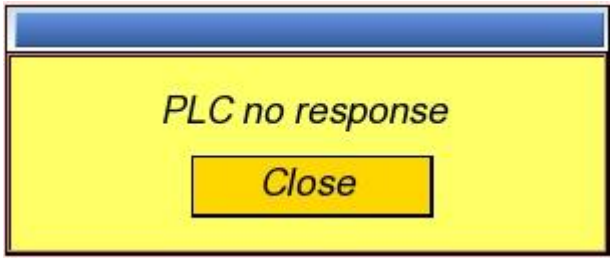
"Local PLC" means the PLC is connected to the local HMI. To control/connect a Local PLC, add this type of device first. Click **[System Parameter Settings] » [New]** to open **[Device Properties]** dialog as shown below:

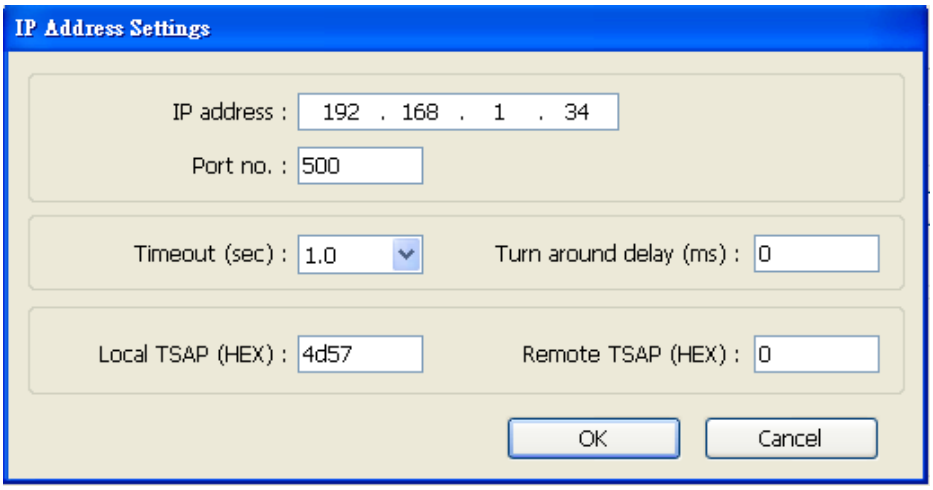
For example, we want to connect "Mitsubishi FX0s/FX0n/FX1s/FX1n/FX2" as a Local PLC:

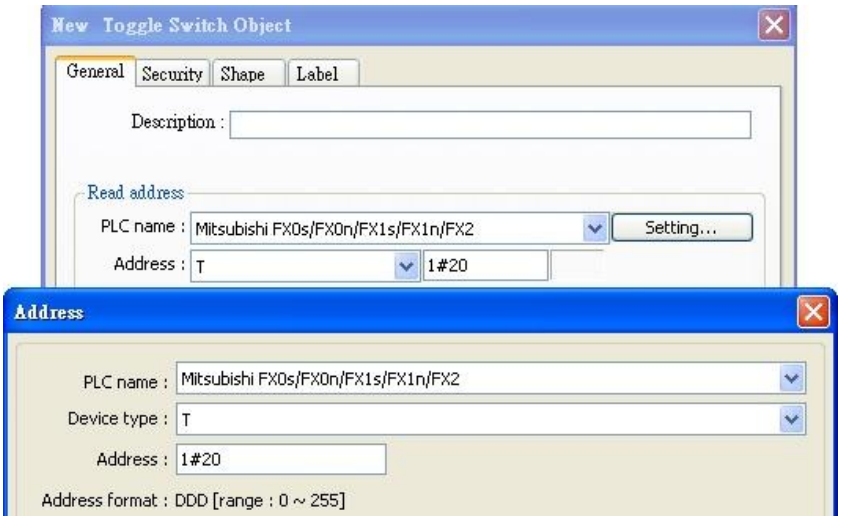


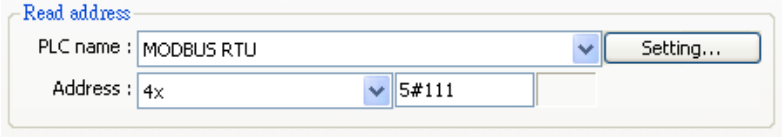
The settings are described below:

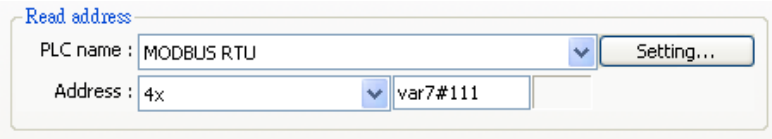
Setting	Description
Name	The name of the device.
HMI / PLC	In this example the device used is a PLC, so select **[PLC]**.

Location	Select **[Local]** or **[Remote]**. In this example the PLC is connected to the Local HMI, so select **[Local]**.
PLC type	Select the type of the PLC.
PLC I/F	The available PLC interface: **[RS-232]**, **[RS-485 2W]**, **[RS-485 4W]**, **[Ethernet]**, and **[USB]**. If the interface used is **[RS-232]**, **[RS-485 2W]**, or **[RS-485 4W]**, configure communication parameters by clicking **[Device Properties] » [Settings]** and then **[Com Port Settings]** dialog opens, as shown below:  **[Timeout]** If the communication has been disconnected for more than preset time limit configured in **[Timeout]** (in sec), Window No. 5 will pop up and show “PLC No Response” message, as shown below:  **[Turn around delay]** While sending the next command to PLC, HMI will delay the sending according to the time interval set in **[Turn around delay]**. This may influence the efficiency of the communication between HMI and PLC. Default value is **“0”**. Note: If the PLC used is **SIEMENS S7-200 Series**, it is recommended to assign “**5**” to **[Turn around delay]** and “**30**” to **[Send ACK delay]**.

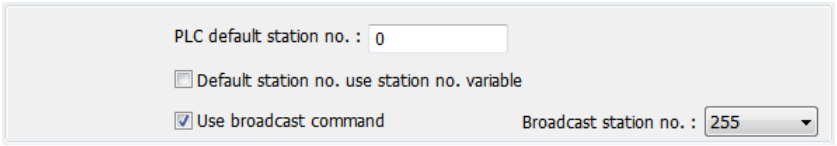
	If the interface used is **[Ethernet]**, click **[Device Properties] » [Settings]** and the **[IP Address Settings]** dialog opens. Please set correct PLC IP address and port number. For example, use a S7-1200 as the connected PLC, as shown below:  If the interface is **[USB]**, no further setting is required. Please check the settings in **[Device Properties]**. If the interface is [CAN (Controller Area Network) Bus], please check the PLC connect guide for “CANopen” and import the eds device file.
PLC default station no.	The default station number for PLC address if the PLC station number is not included in the address, as explained later. PLC station no. can be set in PLC address. The address format: ABC#DEFGH ABC stands for PLC station number and ranges from 0 to 255. DEFGH stands for PLC address. And the “#” sign separates the station number and the address. As shown below, the data is read from PLC station number 1, and address T20.

<table>
<tr><td></td><td> 
</td></tr>
<tr><td>Default station no. use station no. variable</td><td>Use the station number variables as the default PLC station number. Select one from LW-10000 to LW-10015 as the station number variables. If the station no. is not specified in PLC address, the station number will be determined by the station no. variable.
For example, if var7 is set for default station no:
 

The followings demonstrate some examples:
a. The PLC station number is “5”.
 

b. The PLC station number is determined by var7 (LW-10007)
 

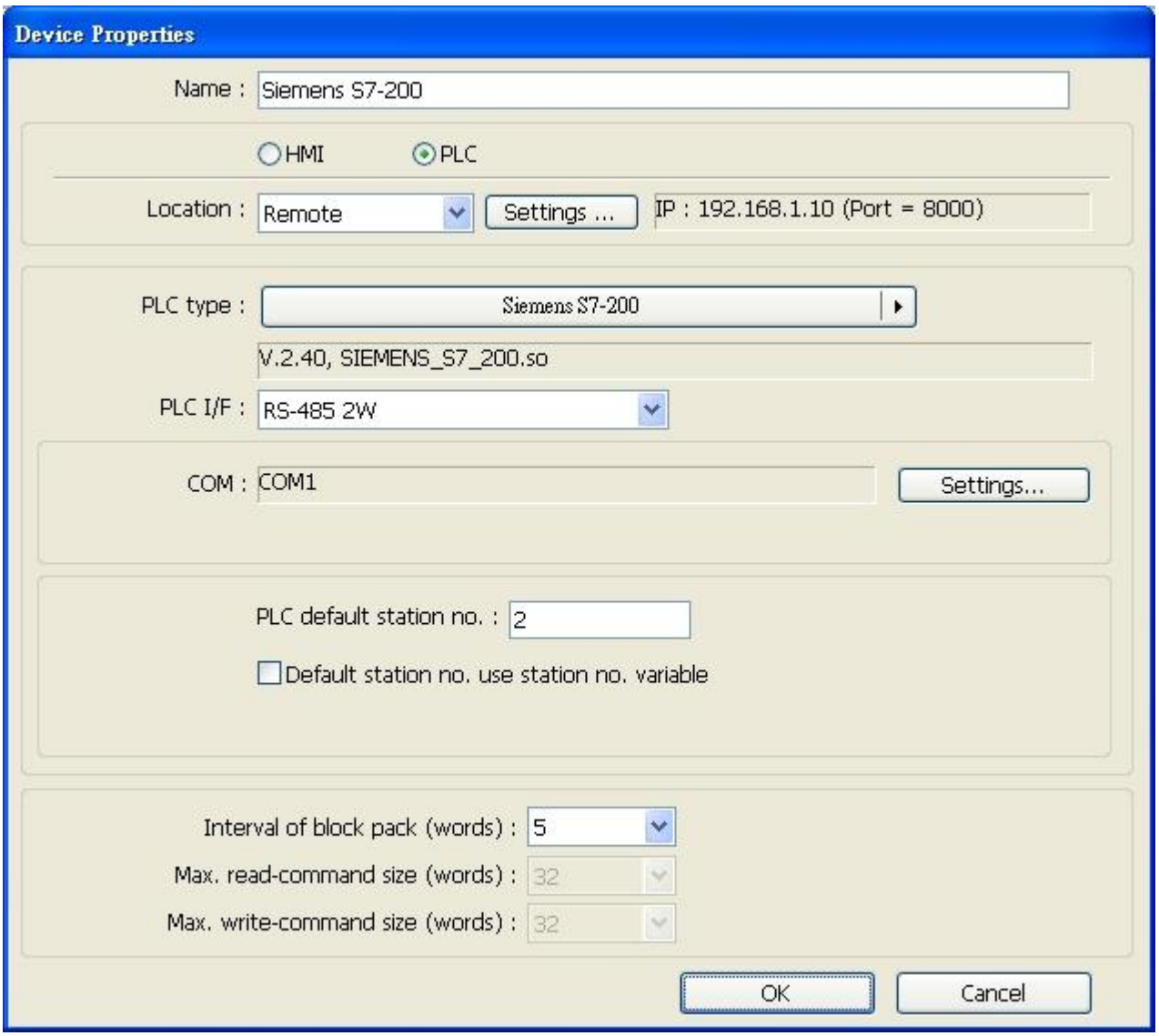
c. PLC address is set to “111”, since PLC station no. is not specified, and the default station no. is using var3, the PLC station no. is determined by var3 (LW-10003).
 
</td></tr>
<tr><td>Use broadcast command</td><td>When [Use broadcast command] check box is select, please fill in [Broadcast station no.] according to the broadcast station number defined by PLC. When HMI sends a broadcast command to the station number set</td></tr>
</table>

	here, PLC will only receive the command and not reply to HMI.  And set as below:  When HMI sends a command to address 255#200, all the PLCs will receive this command and will not reply. Note: Only PLCs that support broadcast command can use this function.
Interval of block pack (words)	If the interval between read addresses of different commands is less than this value, the commands can be combined to one. But the combining function is disabled if this value is set to “0”. For example, the interval value is set to “5”, to read 1 word from LW3 and 2 words from LW6 respectively (read from LW6 to LW7), since the interval of addresses between LW3 and LW6 is less than 5, these two commands can be combined to one. The result is to read 5 consecutive words from LW3 to LW7. Note: The maximum size of command combination data must be less than **[Max. read-command size]**.
Max. read - command size (words)	The maximum data size to read from the device at one time. Unit: word
Max. write - command size (words)	The maximum data size to write to the device at one time. Unit: word.

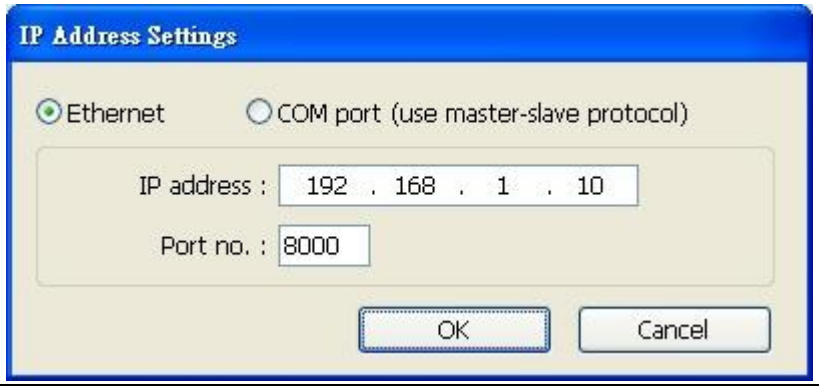
After all settings are completed, a new device named “Local PLC 1” is added to the **[Device list]**.

5.1.2 How to Control a Remote PLC

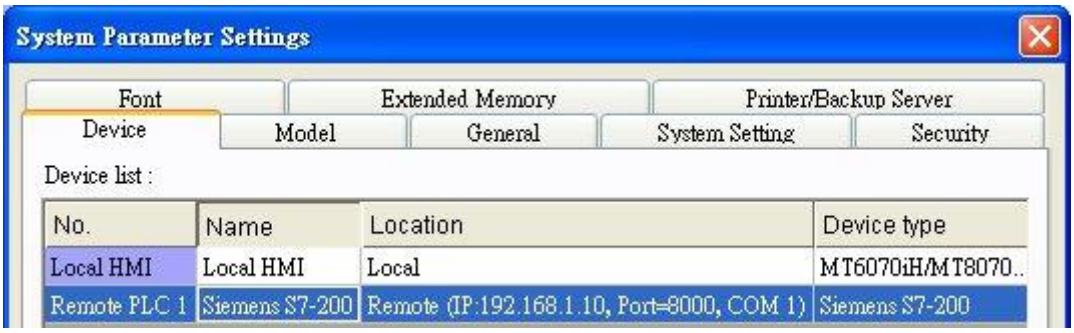
"Remote PLC" is a PLC being connected to a remote HMI. To control a remote PLC, add this type of device first. Please click **[System Parameter Settings] » [New]** to open **[Device Properties]** dialog. For example, use SIEMENS S7-200 as the Remote PLC:



The settings are described below:

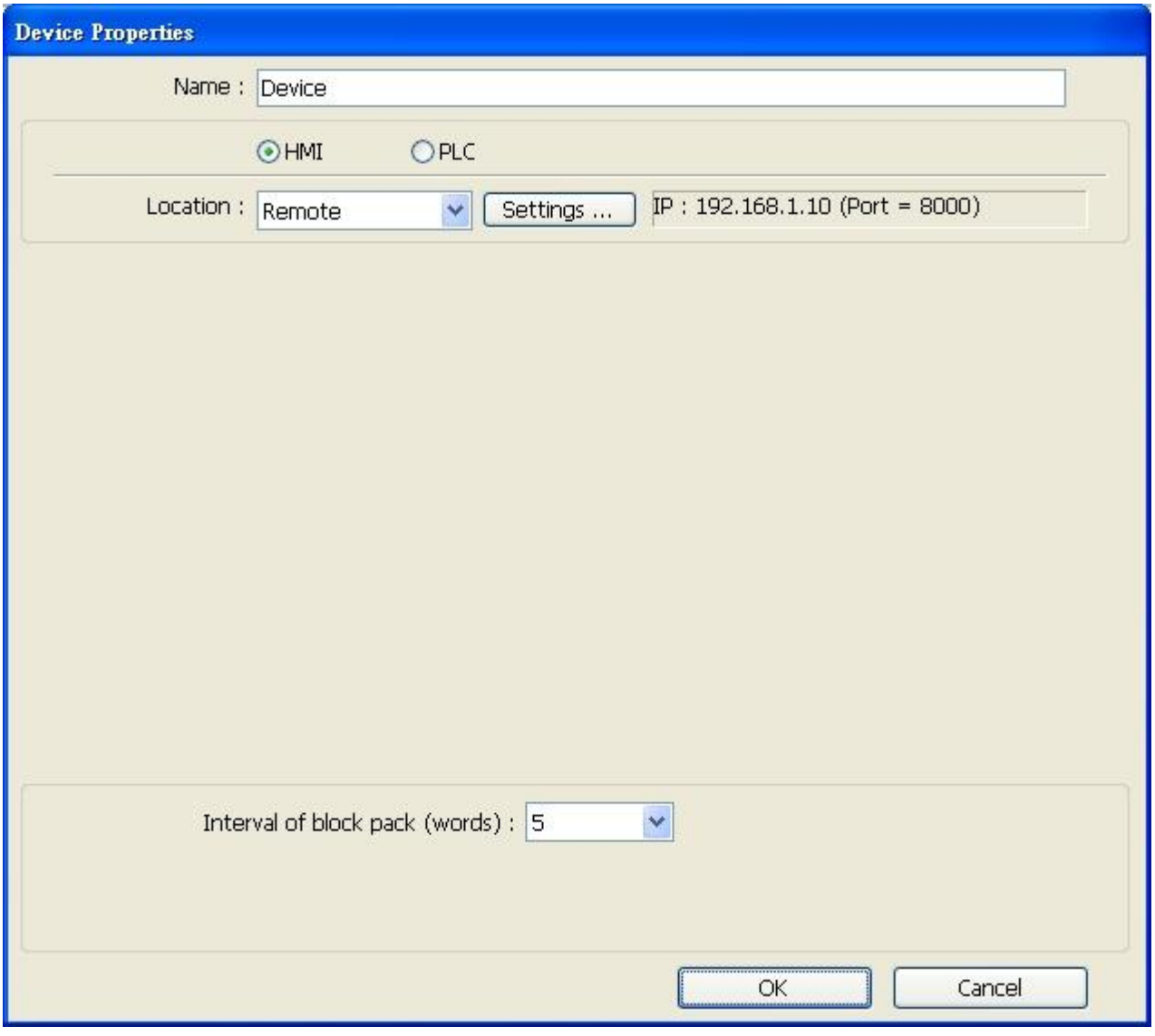
Setting	Description
HMI or PLC	In this example, the device used is a PLC, so select **[PLC]**.
Location	Select **[Local]** or **[Remote]**. In this example the PLC is connected to Remote HMI, so select **[Remote]**. Set the IP address and port number of the Remote HMI by clicking **[Settings]** next to **[Remote]**. 
PLC Type	Select the type of the PLC.
PLC I/F	The interface used for Remote PLC. If the remote PLC uses a COM port, select **[RS-232]**, **[RS-485 2W]**, or **[RS485 4W]**.
PLC default station no.	Set the station number of Remote PLC.
COM	Set the COM port used by the Remote PLC, this setting must be correct.

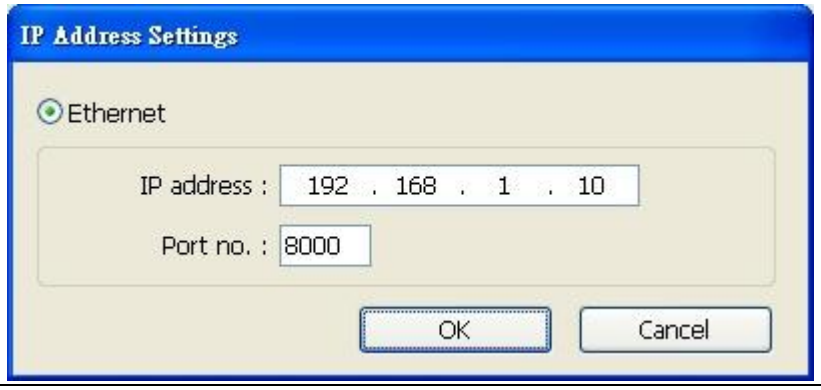
After all settings are completed, a new device named “Remote PLC” is added to the **[Device list]**.



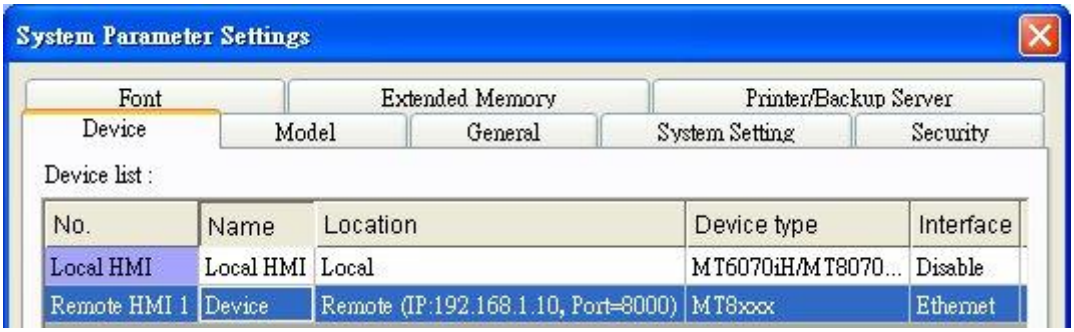
5.1.3 How to Control a Remote HMI

"Remote HMI" is the HMI other than "Local HMI", and PC is also a "Remote HMI". To control a Remote HMI, add this type of device first. Click **[System Parameter Settings] » [New]** to open **[Device Properties]** dialog as shown below:



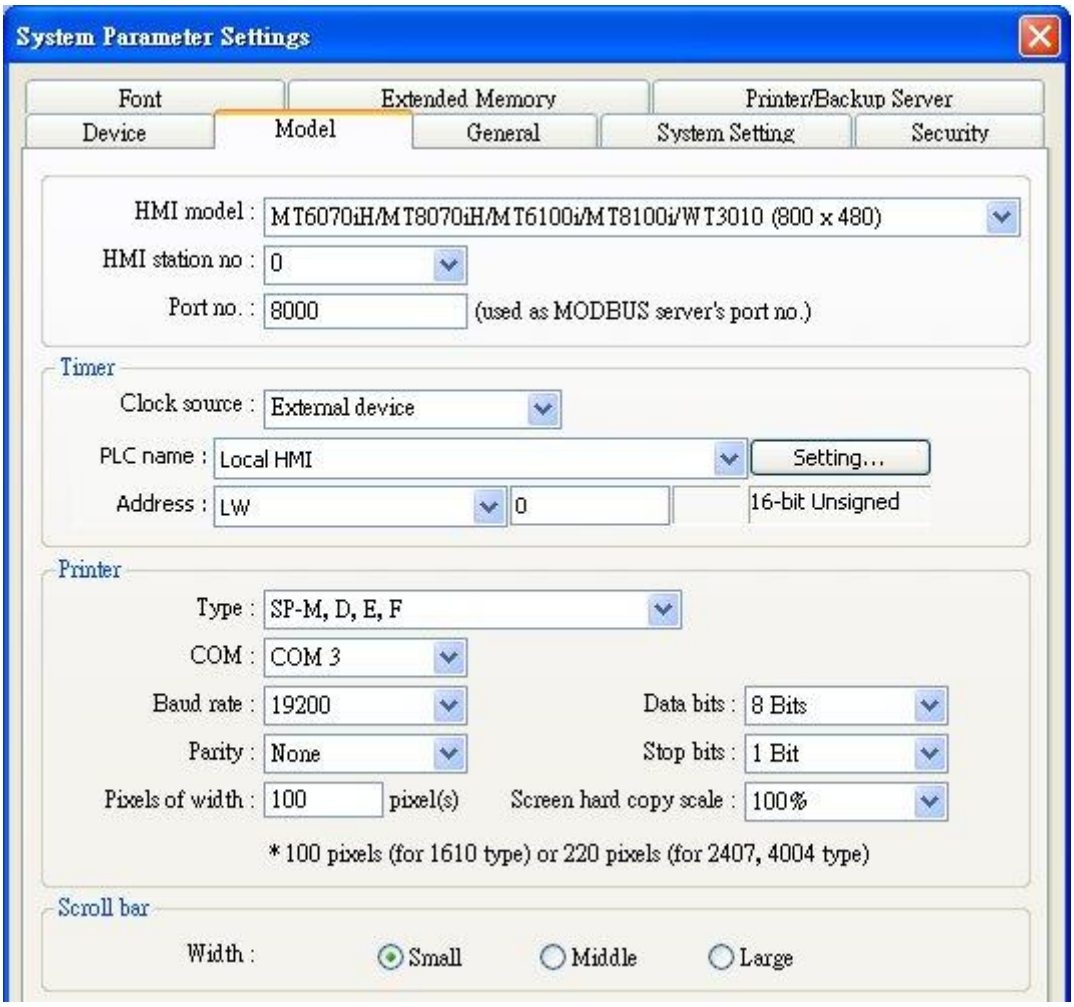
Setting	Description
HMI or PLC	In this example, the device used is a HMI, so select **[HMI]**.
Location	Select **[Local]** or **[Remote]**. In this example Remote HMI is used, select **[Remote]**. Set the IP address and port number of the Remote HMI by clicking **[Device Properties] » [Settings]**. 

After all settings are completed, a new device named “Device” with “No.” as “Remote HMI 1” is added to the **[Device list]**.

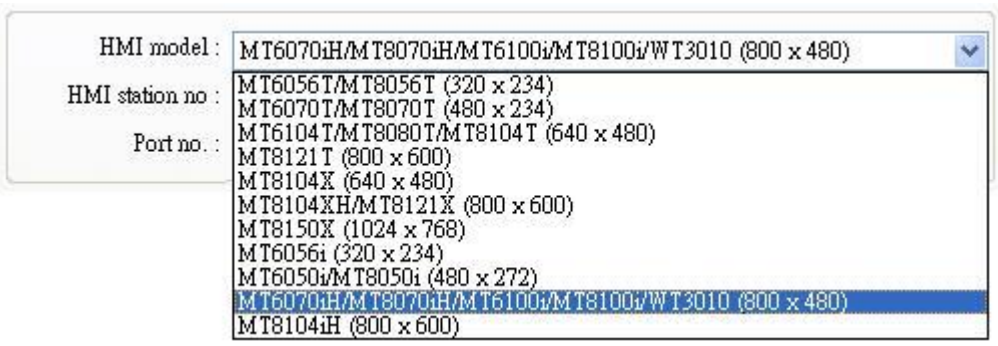


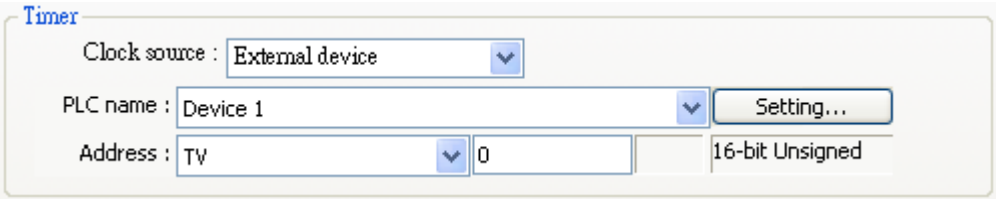
5.2 Model

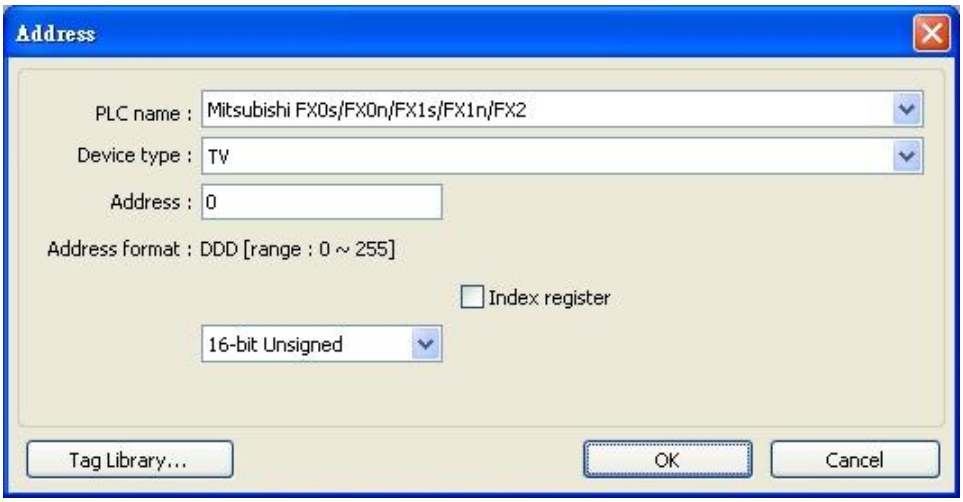
Configure the **[HMI model]**, **[Timer]**, **[Printer]** and **[Scroll bar]** settings.



The settings are described below:

Setting	Description
HMI model	Select the HMI model to use.  If the HMI model is changed, the **[Resize pop-up windows / objects]** dialog will pop up, as shown below. Select required adjustment and click **[OK]**. In most cases, please select all options.

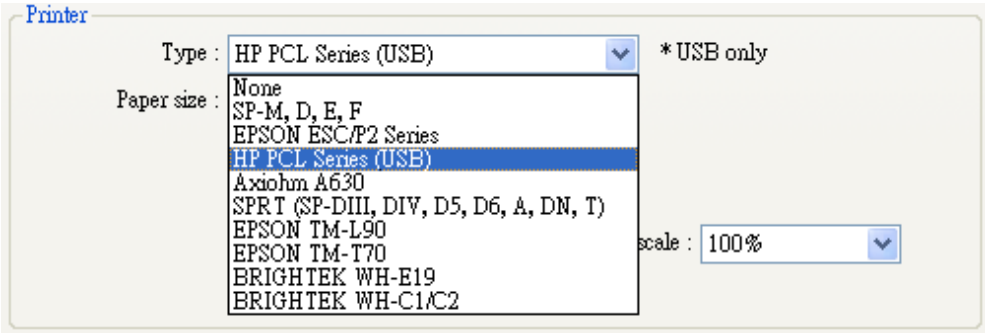
	
HMI station no.	Set the station number of current HMI. The default value is "**0**".
Port no.	Set the port number of current HMI. It is also used in MODBUS TCP server. The default value is "**8000**".
Timer	**[Clock source]** Set the source device of the clock/time information. It is used by [Data Sampling], [Event Log], …., etc. a. If **[HMI RTC]** is selected, the time information comes from the internal clock of the HMI. b. If **[External device]** is selected, the time information comes from an external device. The address of the source device must set correctly. As shown below, the time information is from "TV" address type of the "Local PLC". The addresses of "TV" start from 0 and contain 6 consecutive words and each of them store the following information: TV 0 → Second (range: 0~59) TV 1 → Minute (range: 0~59) TV 2 → Hour (range: 0~23) TV 3 → Day (range: 1~31) TV 4 → Month (range: 1~12) TV 5 → Year (range: 1970~2037) 



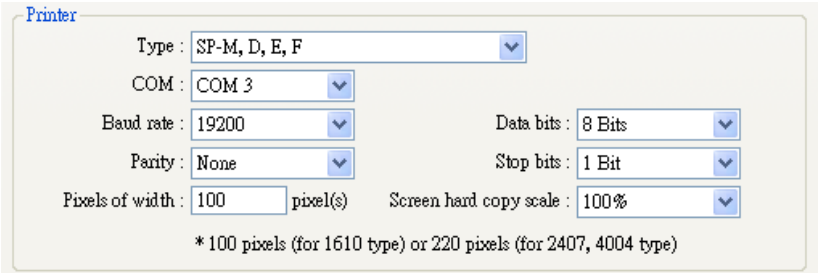
Printer

[Type]

A printer can be connected with the HMI. For HP PCL Series, it has to be connected through USB interface while other printers through a COM port. For more information, please refer to “Chapter 23 HMI Supported Printers”.



If the printer is connected through [COM], configure the parameters correctly. If the printer type is **[SP-M, D, E, F]**, the **[pixels of width]** has to be set accurately, i.e. the set pixel(s) cannot exceed printer’s default setting, or the HMI will fail to print data.



Scroll bar

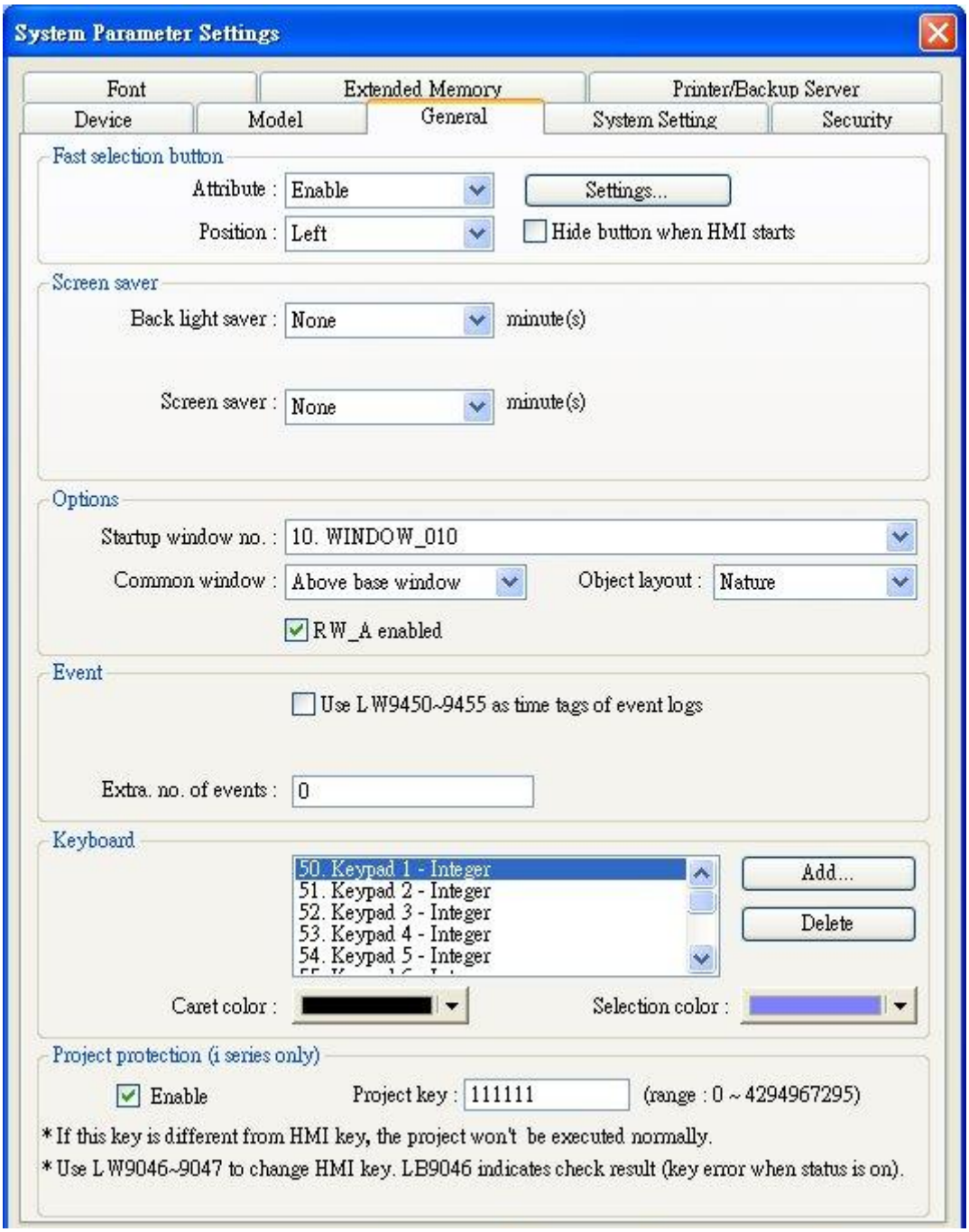
[Width]

Set to **[Small]**, **[Middle]** or **[Large]** as required.

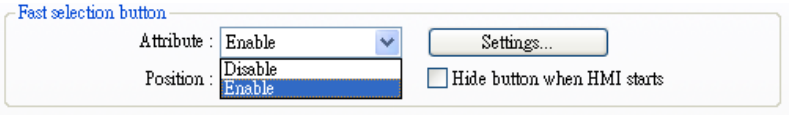
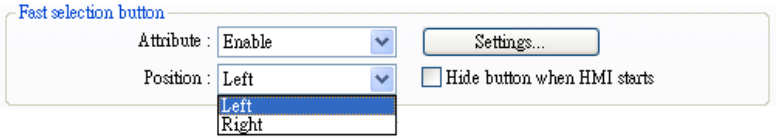
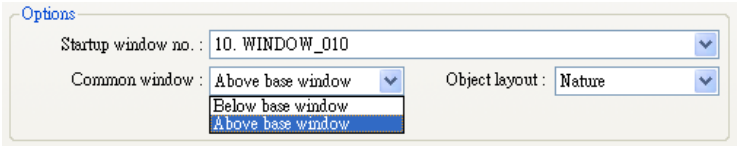
Set the width of Scroll Bar, when the size of the object is too small to display the contents, a scroll bar is displayed in the object.

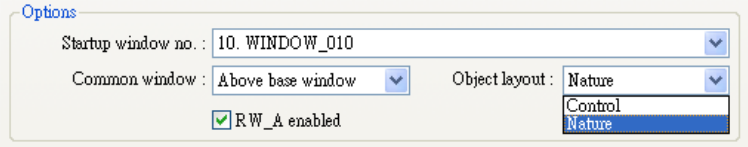
5.3 General

Configure the properties related to screen display.



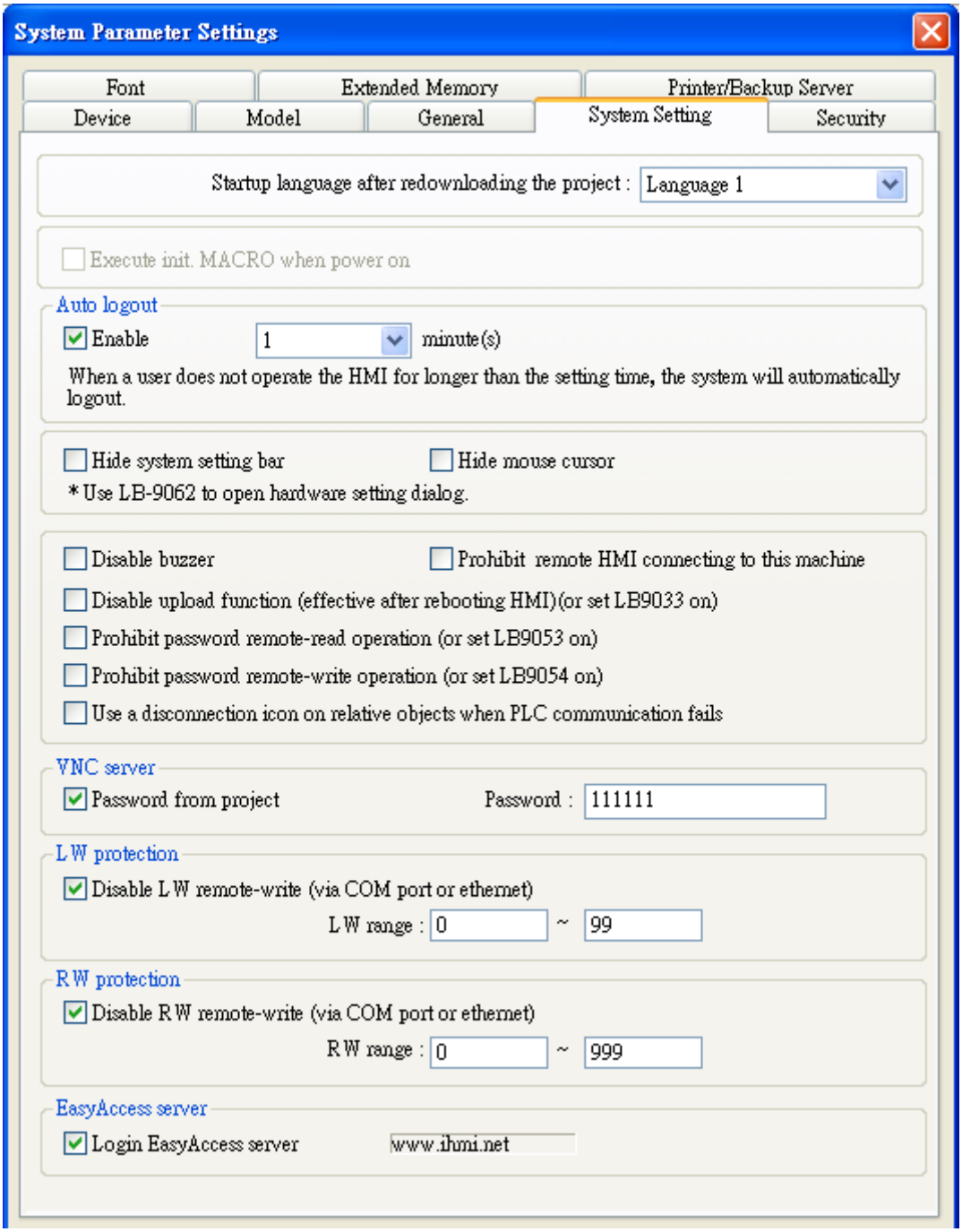
The settings are described below:

Setting	Description
Fast selection button	Setting the attributes for fast selection button for Window No. 3. To use the fast selection button, create Window No. 3 first. **a. [Attribute]**  Enable or disable fast selection window. Select **[Enable]** and click **[Settings]** to set the attributes, including color and text of the button. **b. [Position]**  Select the button position on the screen. If **[Left]** is chosen, the button will show up in at bottom left side of the screen; if **[Right]** is chosen, the button will show at the bottom right side of the screen
Screen saver	**a. [Back light saver]** If the screen is left untouched and reaches the time limit set here, the back light will be turned off. The unit is minute. Back light will be on again once the screen is touched. If **[none]** is set, the back light will always be on. **b. [Screen saver]** If the screen is left untouched and reaches the time limit set here. The current screen will automatically switch to a window assigned in **[Saver window no.]**.The setting unit is minute. If **[none]** is set, this function is disabled. **c. [Saver window no.]** To assign a window for screen saver.
Option	**a. [Startup window no.]** Designate the window shown when start up HMI. **b. [Common window]**  The objects in the common window (Window No. 4) will be shown in each base window. This determines that the objects in common window

	are placed above or below the objects in the base window. **c. [Keyboard caret color]** Set the color of caret that appears when entering data in [Numeric Input] and [Word Input] objects. **d. [Object layout]**  If **[Control]** mode is selected, when operating HMI, **[Animation]** and **[Moving Shape]** objects will be displayed above other kinds of objects neglecting the sequence that the objects are created. If **[Nature]** mode is selected, the display will follow the sequence that the objects are created, the first created will be displayed first. **e. [RW_A enabled]** Enable or disable recipe data RW_A. Enable this, the objects can then control RW_A .The size of RW_A is 64K.
Event	**[Extra no. of events]** The default number of the events in the system is 1000. For additional number of events, modify this setting. The maximal is 10000.
Keyboard	The window number in which the keyboard is placed. When using **[Numeric Input]** or **[ASCII Input]** objects, the type of keyboards can be selected. Up to 32 keyboards can be added. To design a keyboard, a window should be designated for creating it. Press **[add]** after creating, and add the window to the list. For more information, please see “Chapter 12 Key Pad Design and Usage”
Project protection (i series only)	User’s project can be restrained and executed on specific HMI (only for i series HMI). Please refer to “Chapter 30 Project protection” for more information.

5.4 System Setting

[**System Setting**] is used to configure different functions of EasyBuilder.



Some functions are duplicated from system registers, such as, [Hide system setting bar (LB-9020)], [Hide mouse cursor (LB-9018)], [Disable buzzer (LB-9019)], [Prohibit remote HMI connecting to this machine (LB-9044)], and [Disable upload function (LB-9033)]. Users can also set these functions via system tag.

To select a system tag, select **[Address] » [System tag]** check box when adding a new object and then select the **[Device Type]**.

To browse all the system tags, Select **[Library] » [Tag] » [System]** from the main menu of EasyBuilder.

[Startup language after redownloading the project]
Set the language to use when HMI starts after the project is re-downloaded.
[Execute init. MACRO when power on]
Designate the macro to be executed when HMI power on.
[Auto logout]
If leaving HMI untouched for longer than the set time, the objects protected by security classes will not be able to operate. The user ID and password must be entered again to unlock it.
[Hide System Setting Bar]
Hide the system setting bar in the bottom right corner of the HMI screen.
[Hide Mouse Cursor]
Hide the mouse cursor in HMI screen.
[Disable Buzzer]
Mute HMI.
[Prohibit remote HMI connecting to this machine]
Prohibit the connection with a remote HMI. The remote HMI will not be able to control the local HMI.
[Disable upload function (effective after rebooting HMI) (or set LB9033 ON)]
Disable HMI to upload project, after downloading, HMI must be rebooted to disable uploading project.
[Prohibit password remote-read operation (or set LB9053 ON)]
Prohibit Remote HMI to read Local HMI password.
[Prohibit password remote-write operation (or set LB9054 ON)]
Prohibit Remote HMI to write Local HMI password.
[Use a disconnection icon or relative objects when PLC communication fails]
Decide whether or not to display a disconnection icon on relevant objects when failing to communicate with PLC.

When using this function and fail to communicate with PLC, this icon will be shown in the lower right corner of the object as below:

[VNC Server]

Set the password to log in VNC server.

[LW protection], [RW protection]

If select **[Disable LW/RW remote-write]** check boxes and set the protect range in **[LW/RW range]**, values within the protected range cannot be adjusted using Remote HMI.

[Easy Access server]

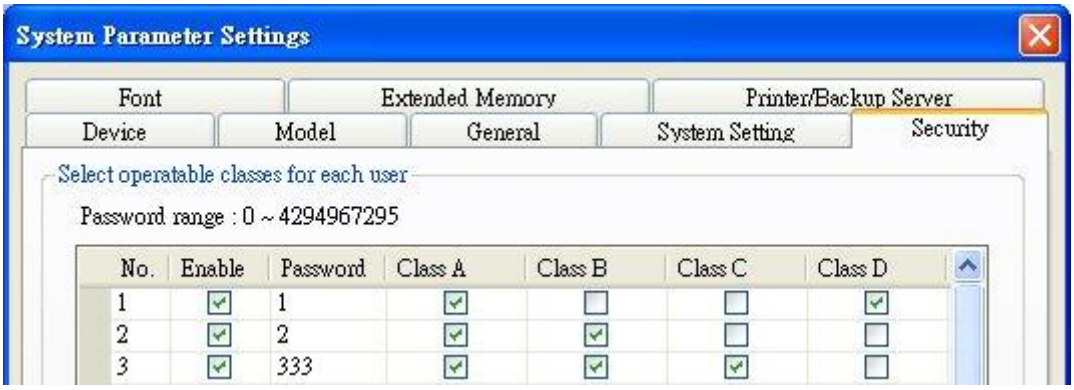
Through this technology, users can easily access to any HMI connected to the internet and operate them on PC just like holding touch screen in hand.

Easy Access does not transmit updated graphic images directly but only the real time data. This makes transmission really quick and efficient. Please refer to “EasyAccess Manual” for more information.

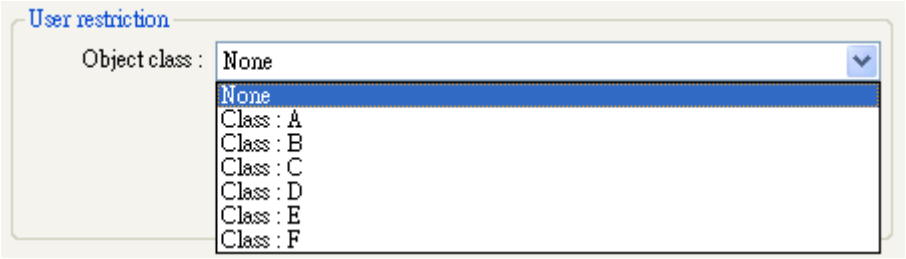
5.5 Security

Parameters in this tab configure the user passwords and security classes.

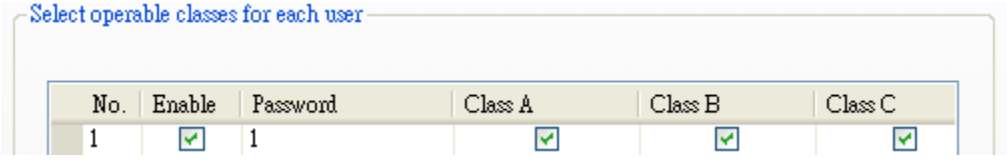
5.5.1 Select operatable classes for each user



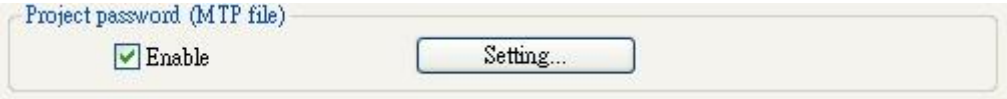
Up to 12 sets of user and password are available. Password should be one non-negative integer. Once the password is entered, the objects that the user can operate are classified. There are six security classes available: A to F.
If **[None]** is selected for an object, every user can access this object.



For example, when the security class of User1 is set as below, User1 could only access objects of classes A, B, C and “none”. For more information, please refer to “Chapter 10 Security”.

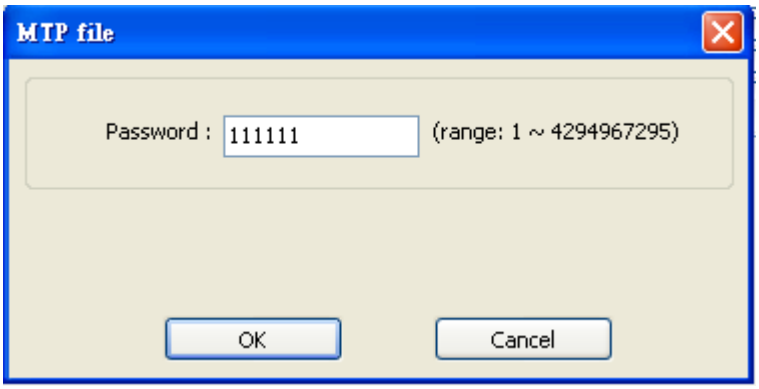


[Project password (MTP file)]



Users can set password to protect the project (mtp) files. The password set here must be entered when editing the project file.

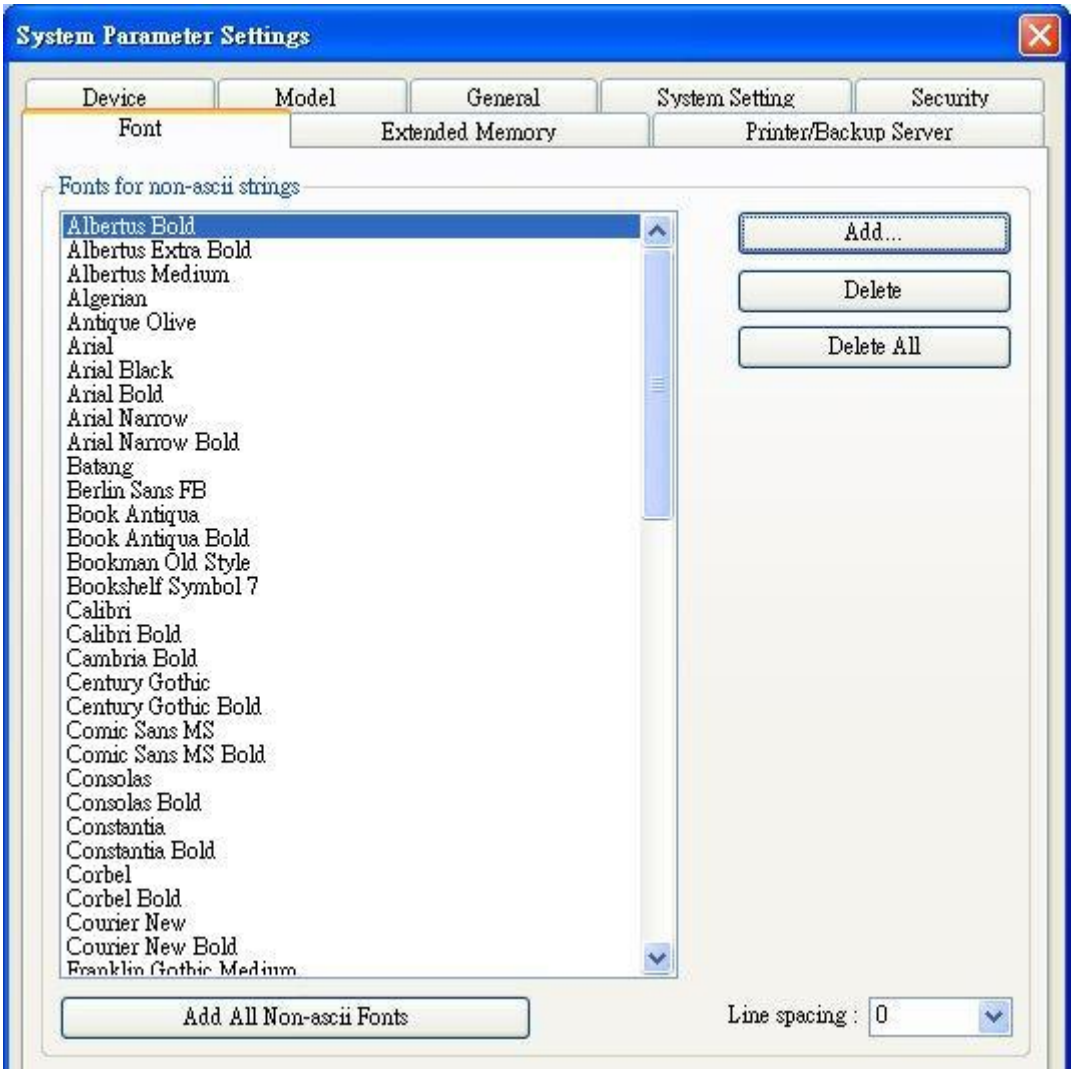
Select **[Enable]** then click **[Setting]**, and the dialog below opens.



Before editing a project, a popup window is shown for entering the password. Only when the password is correct can the user edit this project.

5.6 Font

Parameters in this tab determine the non-ASCII fonts.



[Fonts for non- ascii strings]

The non-ASCII fonts are listed above. When using non-ASCII characters or double byte characters (including Simplified or Traditional Chinese, Japanese, or Korean) which are not listed in **[Fonts for non-ascii strings]** table, EasyBuilder will select a font from the list to substitute for it automatically.

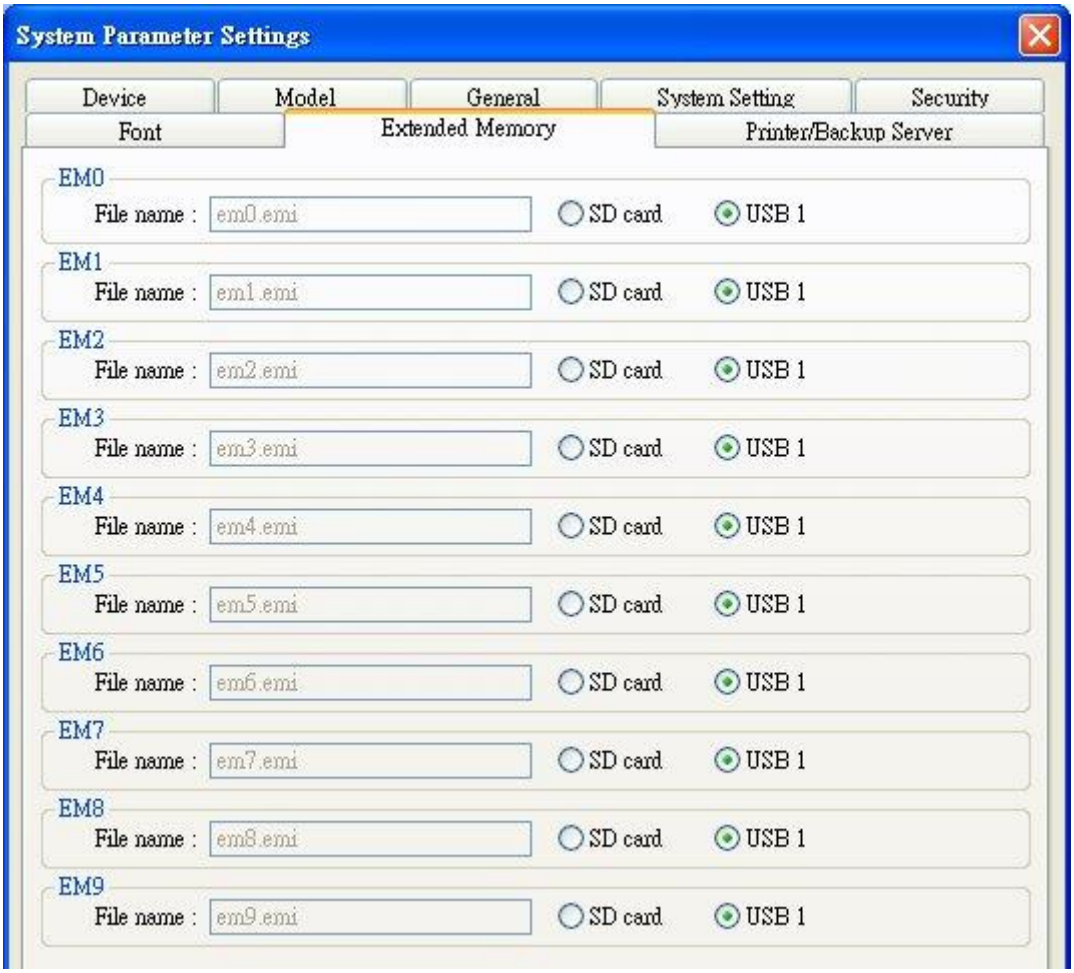
The non-ASCII fonts in Windows can be added to the **[Fonts for non-ascii strings]** table.

[Line spacing]

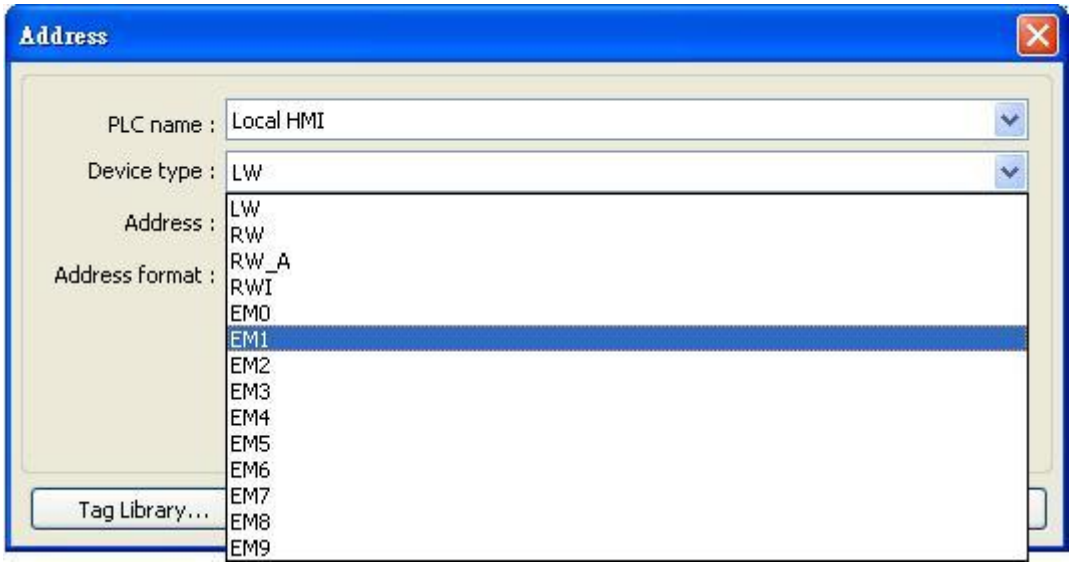
Decide the space between lines in the text.

5.7 Extended Memory

Parameters in this tab determine the location of the extended memory.



Extended Memory is numbered from EM0 to EM9. It works in a way similar to other device types (i.e. LW or RW address). Users can simply select from **[Device type]** list while adding a new object. Size of each extended memory is up to 2G word.



Extended memories are saved as files in **[SD card]** or **[USB disk]**. **[EM0]** to **[EM9]** are saved as “em0.emi” to “em9.emi” respectively. Users can use **RecipeEditor.exe** to open these files and edit the data in the extended memory.

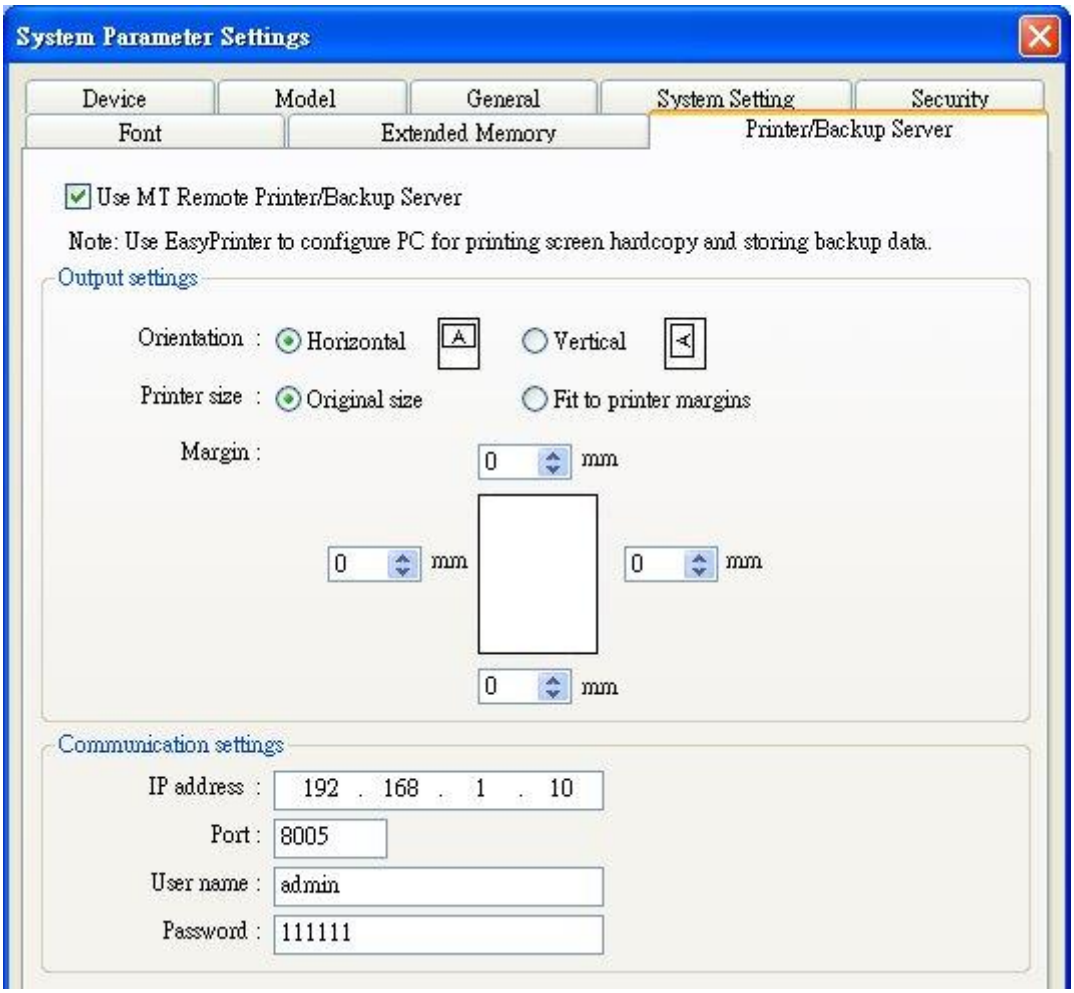
Data in extended memory will not be erased when power is cut, which means next time when start up HMI again, data in the extended memory remains the same as before power off. This is similar to recipe data (RW, RW_A). The difference is that users can specify the location to store data. (SD card, USB disk)

When the device of extended memory does not exist and to read data in it, the data content will be “0”; to write data to a device that does not exist, the "PLC no response" message will be shown in HMI.

HMI supports "Hot Swapping" function for SD card and USB devices. Users can insert or remove the device for extended memory without cutting the power. With this function, users can update or take data in extended memory.

5.8 Printer/Backup Server

Configure remote printer / backup server.



The settings are described below:

Setting	Description
Output settings	**[Orientation]** Set how will words or pictures be printed out, **[horizontal]** or **[vertical]**. **[Printer size]** Set to print out in **[Original size]** or to **[Fit to printer margins]**. **[Margin]** Set the top, bottom, right and left margin width.
Communication settings	**[IP address]** Assign the IP address of the printer via network. **[Port]**, **[User name]**, **[Password]**

	Specify the data to log in printer. Port can be set from 1 to 65535. Maximum length of user name or password is 12 characters.

Please refer “Chapter 26 EasyPrinter” for more information.

Chapter 6 Window Operations

A window is a basic element in a project. With a window, all kinds of information like objects, pictures, and texts can be displayed on HMI screen. Total 1997 windows numbered from 3 ~ 1999 in EasyBuilder can be built and edited.

6.1 Window Types

There are 4 types of windows, each with different functions and usages:
(1) Base Window (2) Fast Selection Window (3) Common Window
(4) System Message Window

6.1.1 Base Window

The most frequently used window, except for main screen, it can also be:
■ A background of other windows.
■ A keyboard window.
■ A pop-up window of [function key] object.
■ A pop-up window of [direct window] and [indirect window] objects.
■ A screen saver.

■ Base Window should be in same size as the HMI screen. Therefore, the resolution of the base window should set to the resolution of HMI.

6.1.2 Fast Selection Window

Window no. 3 is the default Fast Selection Window. This window can co-exist with base window. Generally, it is used to place the frequently-used buttons on the lower-left side or the lower-right side on the screen. Please create window no. 3 first, and set the relevant properties in **[System Parameter Settings] » [General]** tab. Apart from showing or hiding fast selection window with the button on the screen, there are system registers to do so:

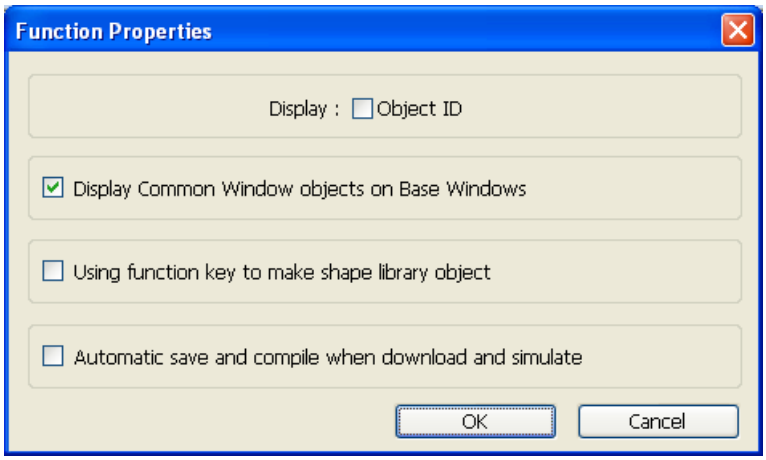
[LB-9013] FS window control [hide(ON)/show(OFF)]
[LB-9014] FS button control [hide(ON)/show(OFF)]
[LB-9015] FS window/button control [hide(ON)/show(OFF)]

6.1.3 Common Window

Window no. 4 is the default Common Window. Objects in this window will be displayed in other base windows, not including popup windows. Therefore, the common objects in different windows are often placed in common window.

When operating HMI, select **[Function Key]** » **[Change common window]** to change the source of common window.

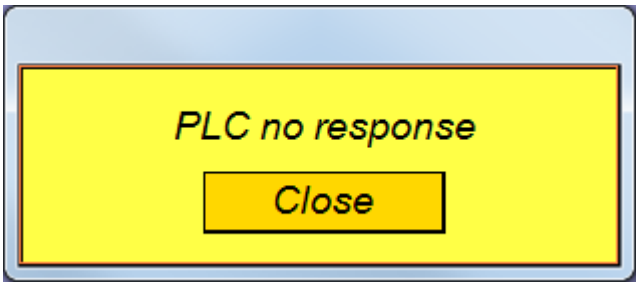
In menu **[Option]** » **[Function Properties]** select whether or not to **[Display Common Window objects on Base Windows]** when editing a project. This can avoid overlapping objects in base window with objects in common window.



6.1.4 System Message Window

Windows No. 5, 6, 7, 8 are the default System Message Windows:

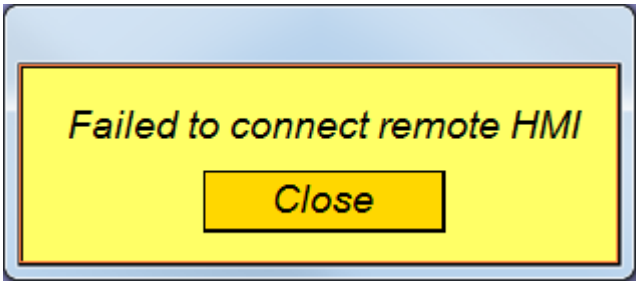
[Window No. 5: PLC Response]
When the communication between PLC and HMI is disconnected, this message window will pop up automatically right on the base window currently opened.



Note

■ “PLC no response” window can be disabled by system registers.
Please refer to “Chapter 22 System Reserved Words & Bits” for more information.

[Window No. 6: HMI Connection]
When failing to connect HMI with a remote HMI, this message window will pop up automatically.

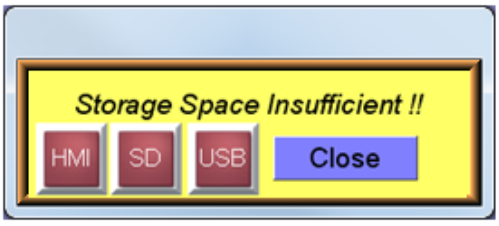


[Window No. 7: Password Restriction]

When attempting to control an object without authorization, this window may pop up as a warning depending on the settings of the object.

[Window No.8: Storage Space Insufficient]

When HMI flash memory, USB disk or SD card run out of storage space, this message window will pop up automatically. (When the memory space is under 4 MB)



The following system registers can be used to check the free memory space in HMI, USB disk, or SD card:

[LW-9072] HMI current free space (K bytes)
[LW-9074] SD current free space (K bytes)
[LW-9076] USB 1 current free space (K bytes)
[LW-9078] USB 2 current free space (K bytes)

To check if there is sufficient storage in the devices, the following system registers can be used. These addresses will set ON when the space is under 4 MB.

[LB-9035] HMI free space insufficiency alarm (when ON)
[LB-9036] SD card free space insufficiency alarm (when ON)
[LB-9037] USB 1 free space insufficiency alarm (when ON)
[LB-9038] USB 2 free space insufficiency alarm (when ON)

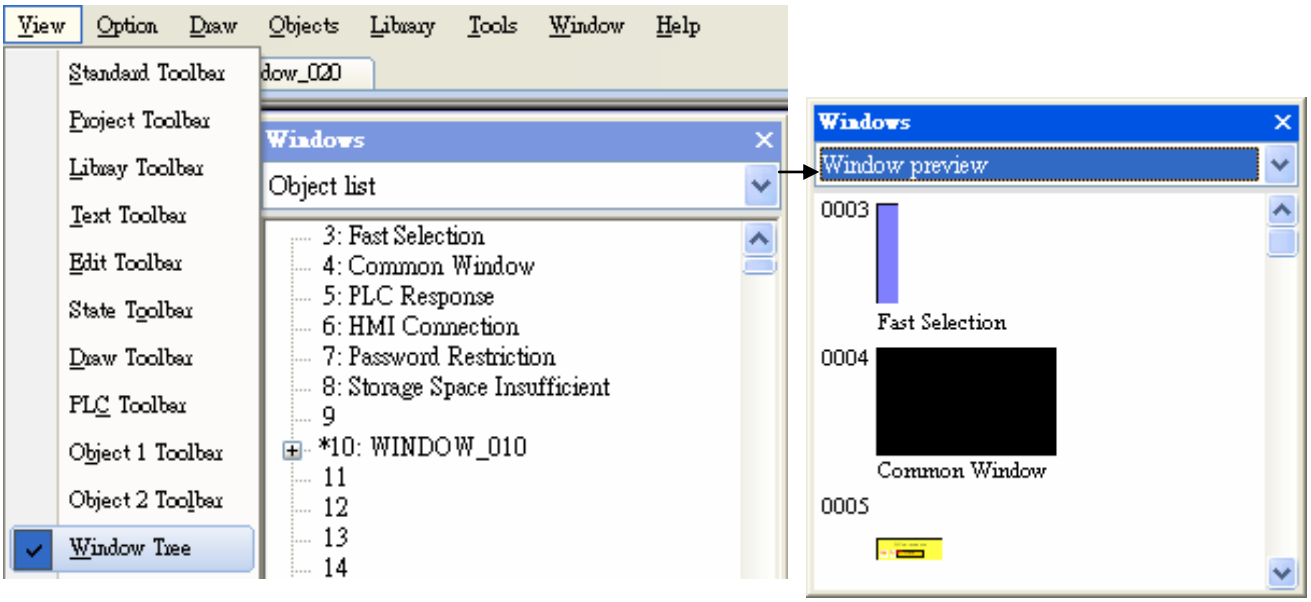
The text shown in windows no. 5 ~ 8 can be edited for easier reference.

Note

- A screen can display up to 16 popup windows simultaneous including System Message Window, Direct Window and Indirect Window.
- The system does not allow opening the same window with two Direct (or Indirect) Windows in one base window.
- Windows no. 3 to 9 are used by the system only, and windows no. 10 to 1999 can be edited based on actual usage.

6.2 Create, Set, and Delete a Window

Check the existing windows in **[View] » [Window Tree]**.

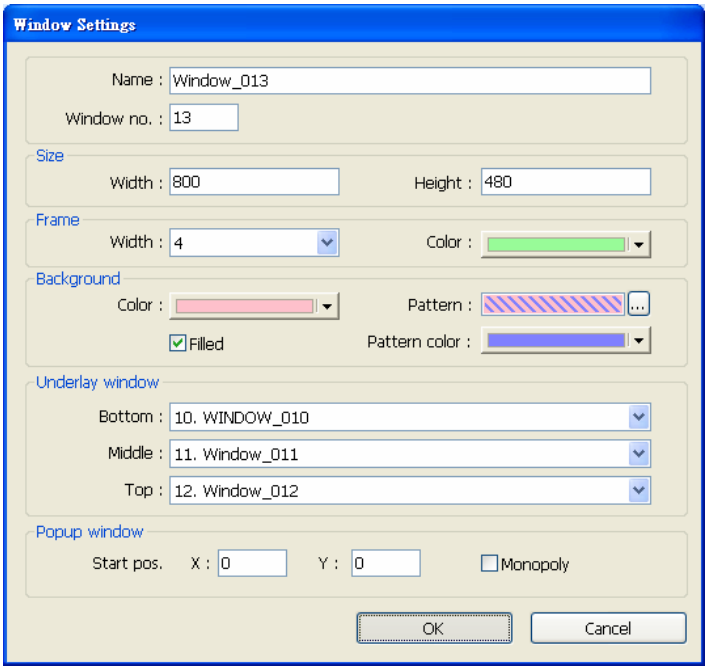


[Object list] displays window numbers and window names. Opened windows are marked with (*) sign. Press the (+) sign to see the object ID, address and description in this window.

[Window preview] displays the thumbnails of windows.

6.2.1 Creating and Setting a Window

In window tree right click on a window number then select **[New]**.



[Name] The name will appear on the title bar and also in window tree.

[Window no.] can be 3 to 1999

[Size] Set the window size in accordance with the HMI resolution.

[Underlay window] [Popup window] Please refer to the description below.

[Underlay window]

Underlay Window can be seen as an extra Common Window. When designing the project, the often used objects may be placed in different windows but not all windows. These objects can be placed in underlay window.

Each base window can set three underlay windows as background, from **[Bottom]** to **[Top]**. The objects in underlay windows are displayed in this order in base window.

[Popup window]

Base window can also be used as a pop-up window. Use **[X]** and **[Y]** to set the coordinates indicating where in the screen will this base window pop up. The origin of the coordinates is the upper-left corner of the window.

[Monopoly]

If the option is selected, when the base window pops up, no operations of other popup windows and background windows are allowed until the monopoly window is closed. If a base window is used as a keyboard window, “Monopoly" is automatically enabled.

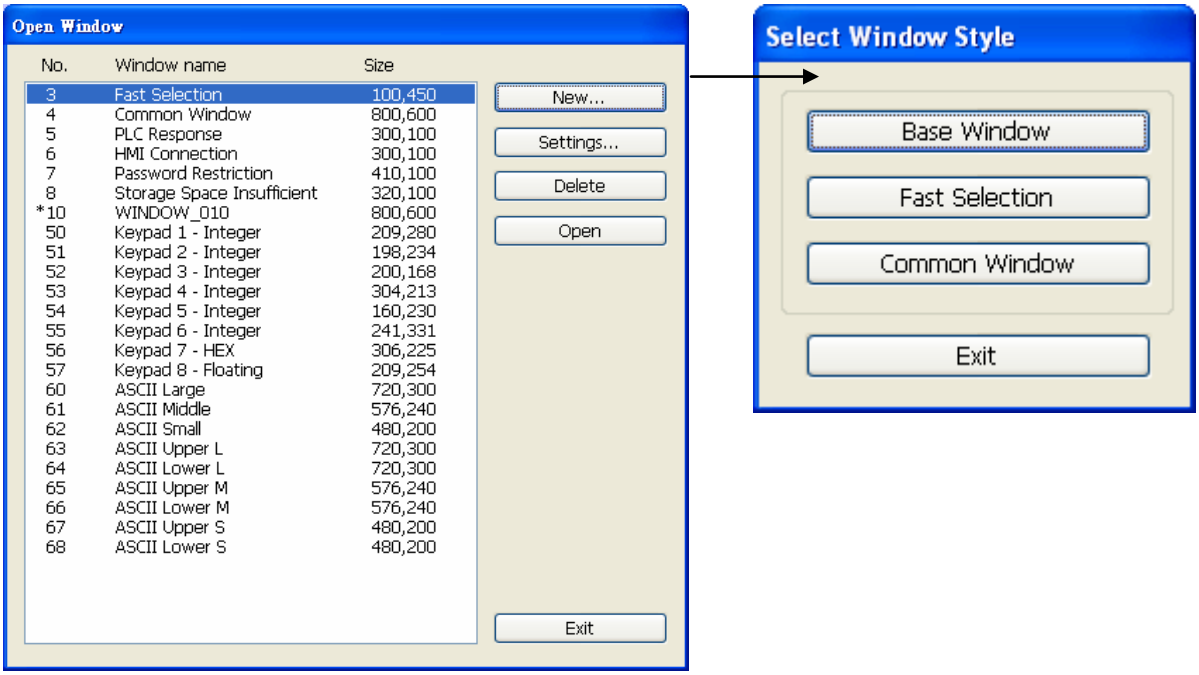
■ The objects in underlay window cannot be edited from the base window that displays them. To edit those objects, please open the underlay window where they are located.

■ When the window number of the underlay window used by the base window is identical to the popup window, the popup window is disabled.

■ When base window and popup window use the same underlay window, the objects in the underlay window cannot be displayed in popup window.

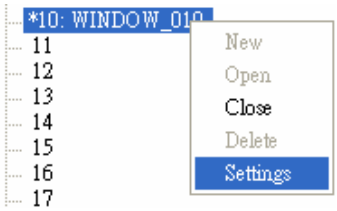
[Window] » [Open Window]

Click **[New]** and select the type of the window and click **[OK]**.



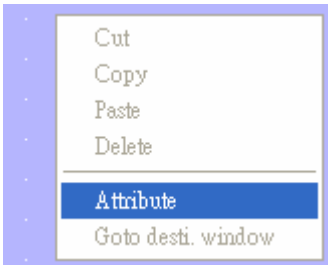
The way to call up **[Window Settings]** dialog:

■ Right click on the window number in the window tree and select **[Settings]**.



■ In **[Window] » [Open Window]** select the window then click **[Settings]**.

■ In the window, right click when no object is selected, and select **[Attribute]**.



6.2.2 Open, Close and Delete a Window

Open an existing window:

■ Double click on the window number in the window tree.

■ In the window tree, select the window, right click, then select **[Open]**.

■ In **[Window] » [Open Window]** select the window then click **[Open]**.

Close or delete an existing window:

■ In the window tree, select the window; right click, then select **[Close]** or **[Delete]**.

■ In **[Window] » [Open Window]** select the window then click **[Delete]**.

To delete a window, please close it first.

Chapter 7 Event Log

Define Event content and trigger condition.

Trigger Event according to the condition.

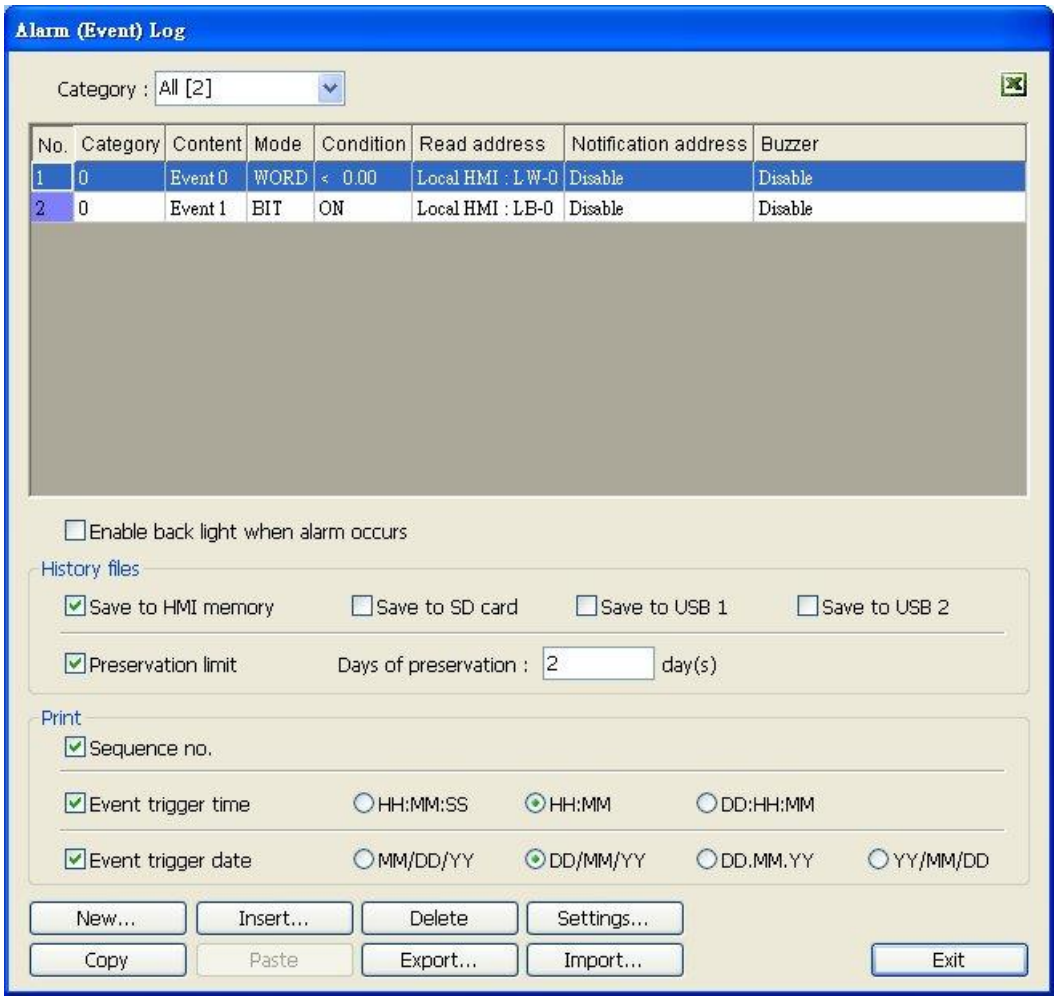
Save the Event Log to specified device.

View the process of Event using the object.

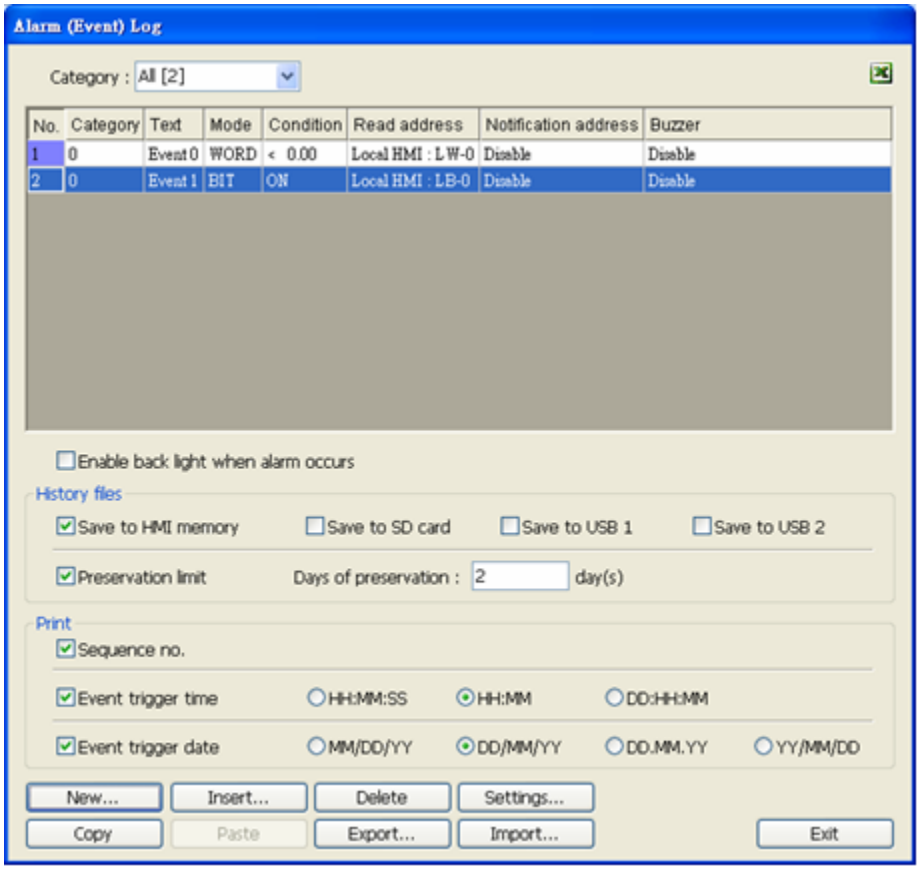
7.1 Event Log Management

Alarm Bar / Alarm Display / Event Display

Using these objects to view the process of the whole event from triggering→waiting to be processed→return to normal. Define the event content first.



The settings are described below:



[Category]

Classifies events by dividing them into 0 ~ 255 categories. Select one category to add or view event log. In the bracket “[ ]”, it shows how many events are in this category.

[History files]

To specify the storage location of an event log. However, when executing On-line or Off-line Simulation on PC, the files will be saved in the HMI_memory / SD_card / USB folder under the installation directory.

[Preservation limit]

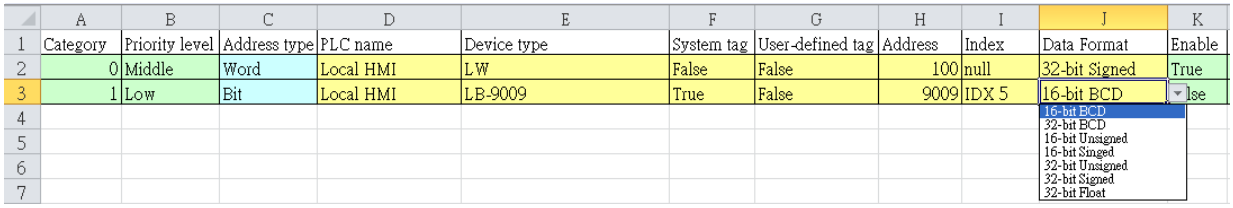
This setting determines how many days the data is preserved. For example, the **[Days of preservation]** is set to two days; the data of yesterday and the day before yesterday will be kept. Data that is built before this period will be deleted automatically to prevent the storage space from running out.

[Print]

In **[System Parameter Settings] » [Model]**, select a printer. The printing format can then be set.

7.1.1 Excel Editing

Click on the Excel icon in Event Log setting dialog to open the Excel template for a reference of editing.
This template is under the installation directory, the file name is EventLogExample.xls. This template includes the ready-made dropdown lists and validation mechanism.



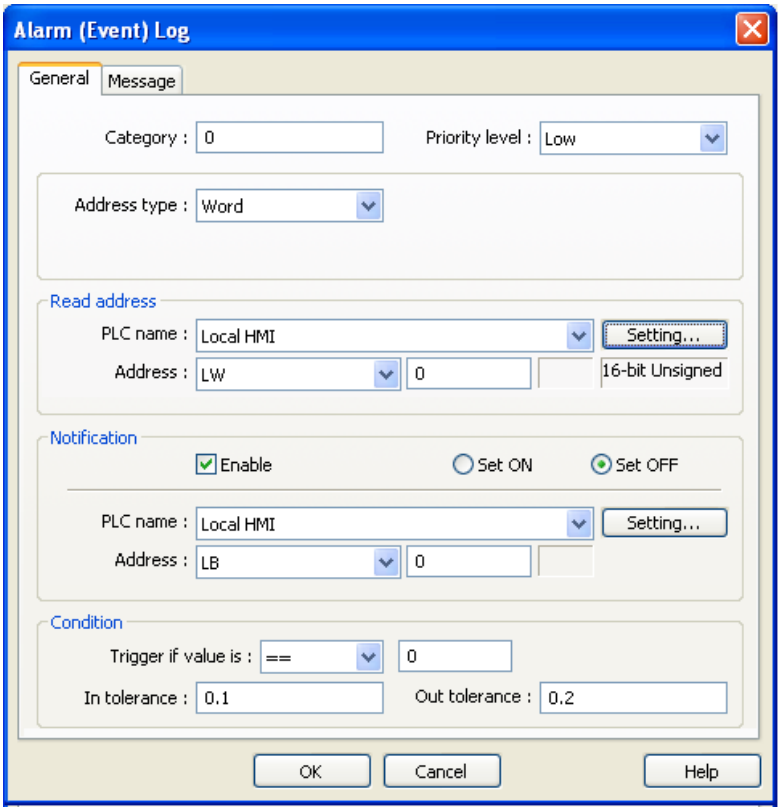
	A	B	C	D	E	F	G	H	I	J	K
1	Category	Priority level	Address type	PLC name	Device type	System tag	User-defined tag	Address	Index	Data Format	Enable
2	0	Middle	Word	Local HMI	LW	False	False	100	null	32-bit Signed	True
3	1	Low	Bit	Local HMI	LB-9009	True	False	9009	IDX 5	16-bit BCD	lse
4										16-bit BCD	
5										32-bit BCD	
6										16-bit Unsigned	
7										16-bit Singed	
										32-bit Unsigned	
										32-bit Signed	
										32-bit Float	

1. **[System tag]** and **[User-defined tag]** cannot set to true simultaneously, otherwise, the system will view the User-defined tag to be a System tag, and **[User-defined tag]** to be false. If setting **[Device type]** to **[User-defined tag]**, please set **[System tag]** to false.
2. **[Color]** format is R:G:B, each should be an integer form 0 to 255.
3. When setting **[User-defined tag]** to true, if the system compares the **[Device type]** with the user-defined tag in the system, and no suitable tag is found, the system will set the **[User-defined tag]** in event log to false.
4. Before importing Label Library / Sound Library, please make sure the library names exist in the system.

7.2 Create a New Event Log

7.2.1 Event (Alarm) Log General Settings

Click **[New]** in the **[Event (Alarm) Log]** dialog, the dialog below appears. Select **[General]** tab.



[Category]

Select event category, 0 ~ 255.

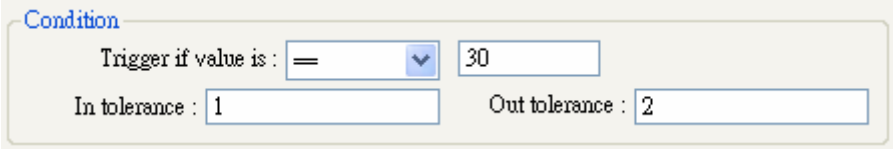
[Priority level]
When the number of events equals to the max number available in the system (default 1000), the lower priority events will be deleted and new events will be added in.

[Read address]
System reads data from this address to check if the event matches the trigger condition.

[Notification]
When enabled, the system will set the specified address ON or OFF when the event is triggered.

[Condition]
When [Bit] is selected, Event Log will detect the state of a Bit address.
When [Word] is selected, Event Log will detect the value of a Word address to check if it is greater than, less than, or equals to a specified value.

Example 1



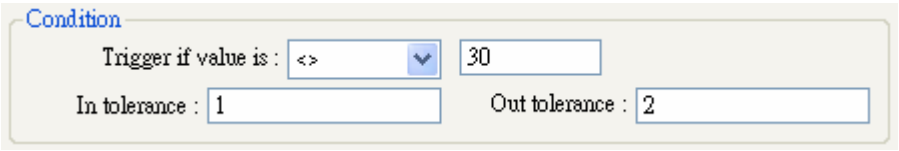
The setting above indicates:
When **[Read address]** value is greater than or equals to 29 (= 30 - 1)
Or less than or equals to 31 (= 30 + 1), the event will be triggered. The trigger condition:

29 ≤ [Read address] value ≤ 31

After the event is triggered, when **[Read address]** value is greater than 32 (= 30 + 2) or less than 28 (= 30 - 2) the system will return to normal condition:

[Read address] value < 28 or [Read address] value > 32

Example 2



The setting above indicates:
When **[Read address]** value is less than 29 (= 30 - 1)
or greater than 31 (= 30 + 1), the event will be triggered. The trigger condition:

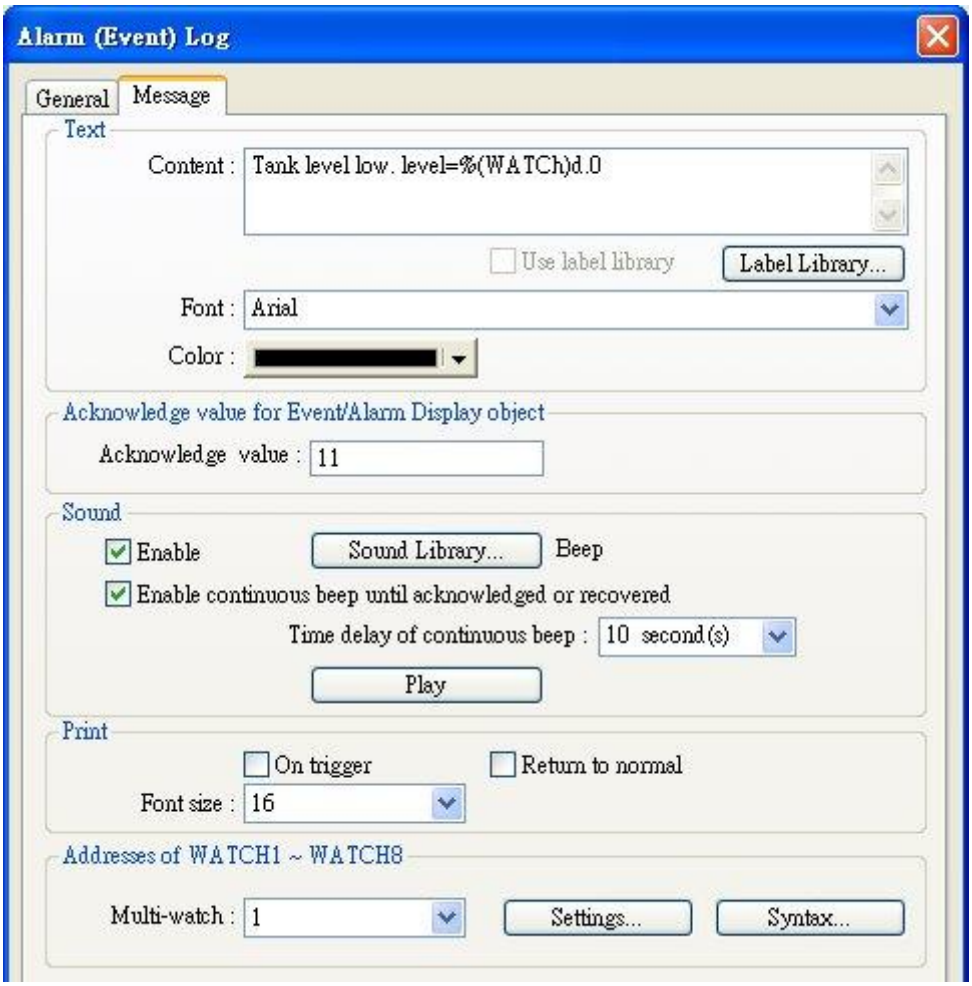
[Read address] value < 29 or [Read address] value > 31

After the event is triggered, when **[Read address]** value is greater than or equals to 28 (= 30 - 2) or less than or equals to 32 (= 30 + 2) the system will return to normal condition:

28 ≤ [Read address] value ≤ 32

7.2.2 Event (Alarm) Log Message Settings

Select **[Message]** tab:



The settings are described below:

[Content]

The text content displayed in [Alarm Bar], [Alarm Display], and [Event Display] objects. Use the formats in the following two examples or WATCH addresses to use register data in content.

Example 1

The data of the LW register can be used in the content displayed when an event is triggered:
Format: **%#d** (% -> initial sign, # -> address, d -> end sign)
When an event is triggered, if the value in LW-20 is 13:
Setting: “High Temperature = %20d”→ Display: “High Temperature = 13”

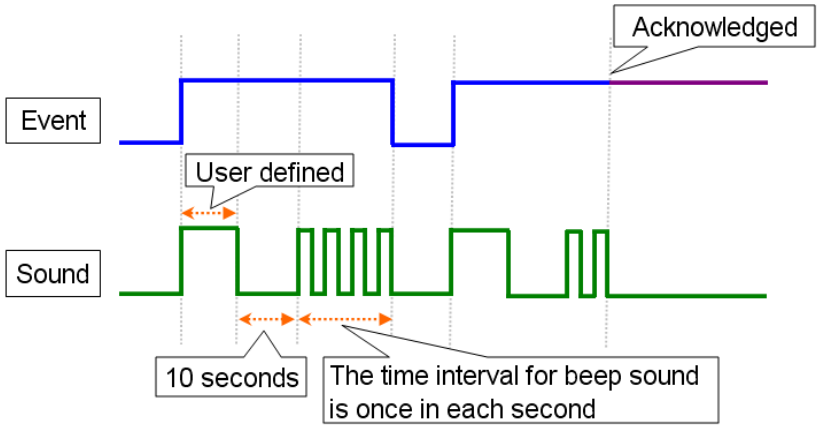
Example 2

The data in the specified address when the event is triggered can be included in the content displayed. The address should be set to the **[Read address]** of Event Log, take MODBUS RTU 4x address as an example:
Format: **$#d** ($ -> initial sign, # -> address, d -> end sign)
When an event is triggered, if the value in MODBUS 4x-15 is 42:
Setting: “High Temperature = $15d” → Display: “High Temperature = 42”

[Font] / [Color]
The font and color can be set differently for each event. The setting determines the font and color shown in [Alarm Bar], [Alarm Display] or [Event Display] objects.

[Write value for Event/Alarm Display object]
When an event in [Event Display] or [Alarm Display] is acknowledged, the value is written to the assigned **[Write address]**.

[Sound]



If enabled, the selected sound will be played when an event is triggered. Continuous beep can also be enabled, which only stops when the event is acknowledged or recovered.
For continuous beep, a delay time can be set between triggering the alarm and the start of beeping.

[Address of WATCH 1 ~ WATCH 8]
Click **[Syntax]** to edit and display the value in watch address when the event is triggered. Up to 8 watch addresses can be set.

Chapter 8 Data Sampling

Define how the data is sampled
1. sampling time
2. address
3. data length

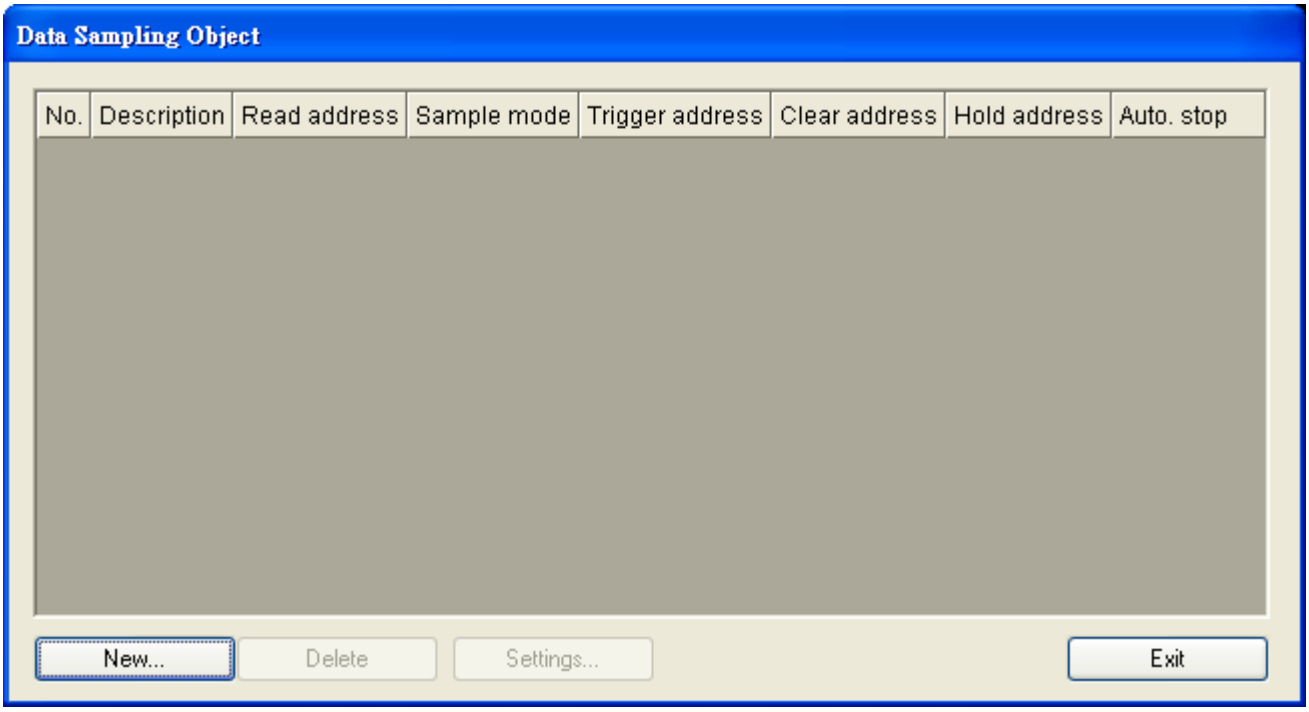
Save the sampled data to the designated location, such as HMI, SD card, USB disk.

Use [Trend Display] and [History Data Display] objects to display sampling records.

8.1 Data Sampling Management

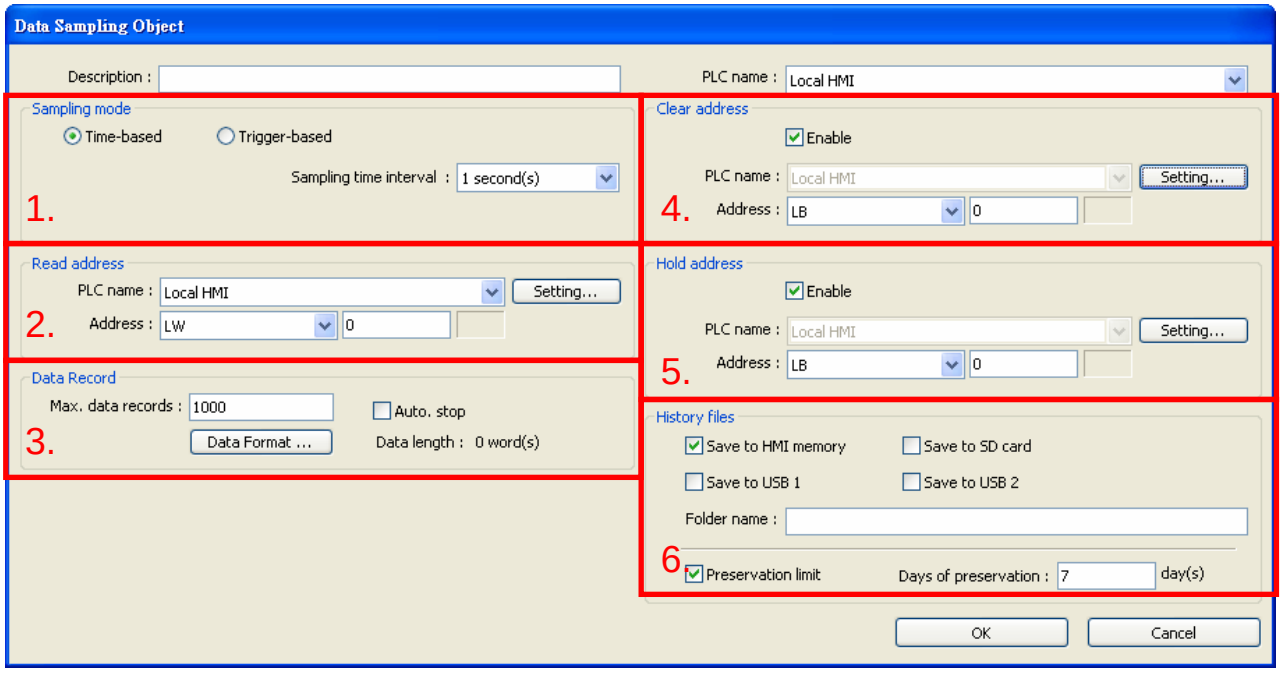
Create a new Data Sampling object first by the following steps:

1. From the menu select **[Objects]** and click on **[Data Sampling]**.
2. Click **[New]** to finish relevant settings:



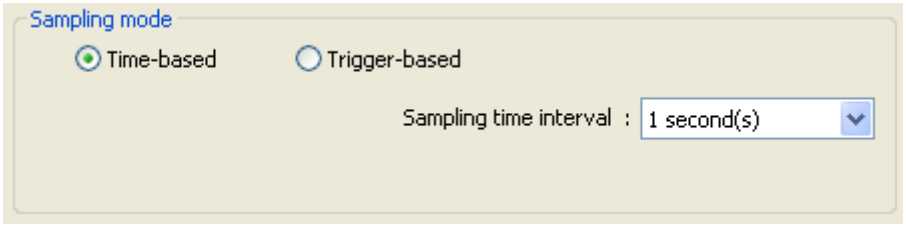
8.2 Create a New Data Sampling

The following introduces how to add a new Data Sampling:



1. [Sampling mode]

[Time-based] mode samples data in a fixed frequency. The **[Sampling time interval]** can be set from “0.1 second(s) to 120 mins”.



[Trigger-based] mode triggers data sampling by the status of a designated address.

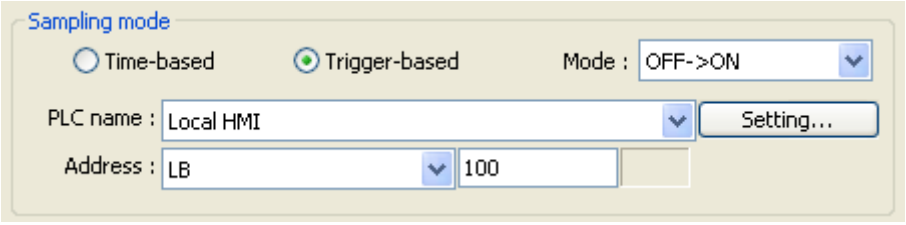
[Mode]

Conditions to trigger Data Sampling:

[OFF -> ON] Trigger sampling when the status of the address changes from OFF to ON.

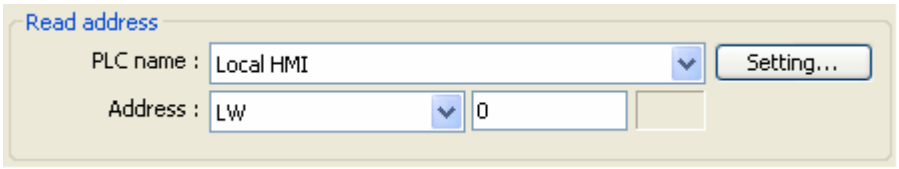
[ON -> OFF] Trigger sampling when the status of the address changes from ON to OFF.

[OFF <-> ON] Trigger sampling when the status of the address changes.



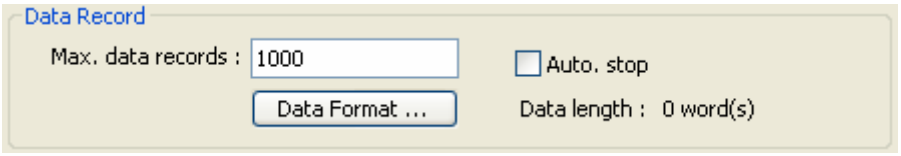
2. [Read address]

Specify an address to be the source of Data Sampling.



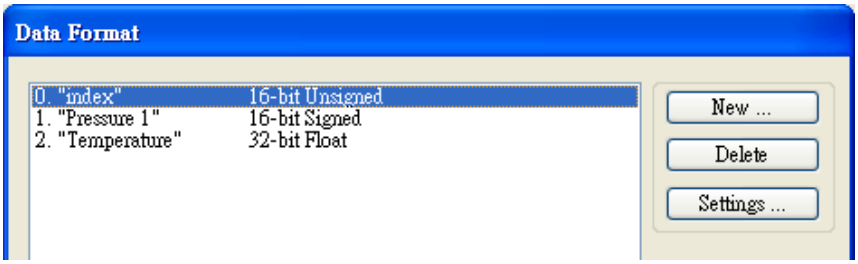
3. [Data Record]

The max. number of data records can be saved by one Data Sampling in one day is 86400. (1 record per second for 24hours) If **[sampling time interval]** is set to “0.1 second”, the max. number of data records is still 86400.



[Data Format]

Available to read several data in different format.

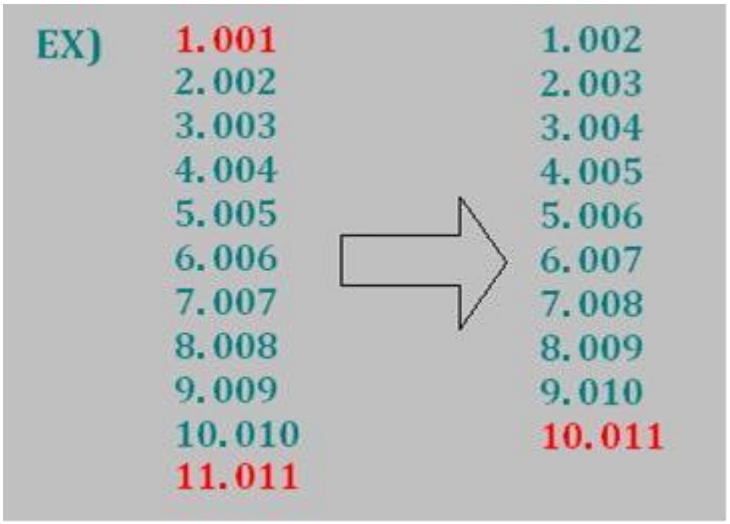


[Auto. stop]

This function depends on the arrangement of different objects and modes. (Set [Max. data records] to n.)

Objects	**not** selecting **[Auto. stop]**	selecting **[Auto. stop]**
Trend Display - Real Time	Deletes the earlier records and displays the latest number of records (n) in Trend Display. Please refer to the following figure.	Stops after reaching the specified number of data records (n).
Trend Display - Historical	Keeps on sampling data and displays all history data in Trend Display.	Stops after reaching the specified number of data records (n).
History Data Display	Keeps on sampling data and displays all history data in History Data Display.	Stops after reaching the specified number of data records (n).
Data Sampling	Keeps on sampling new data.	Stops sampling after reaching the specified number of data records (n).

The figure illustrates how the data is sampled in **Trend Display – Real Time** mode when **Auto. stop** check box is not selected.

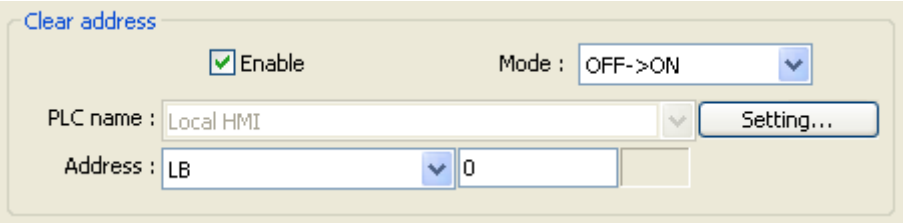


As shown in the preceding figure, if **Data length** is set to 10, when the 11th data is generated, the oldest data is deleted, and the newest data is added.

- A Data Sampling may include more than one type of records. Data Sampling can retrieve different types of records at the same time. For example, user defines three types of data, 4 words in total. In this way, system retrieves a 4-word data each time from the designated address to be the content in one Data Sampling.
- When running simulation and save data sampling records, to change the data format, be sure to delete the previous data records in the installation directory to avoid the system to read the old data records.

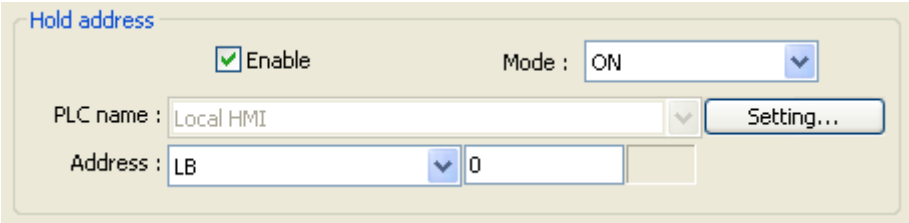
4. [Clear address]

Set when the bit address status changes from [OFF -> ON] or [ON -> OFF], clear the sampled data in Trend Display Real-time Mode. The number of data records returns zero but the data records that are already saved will not be cleared.

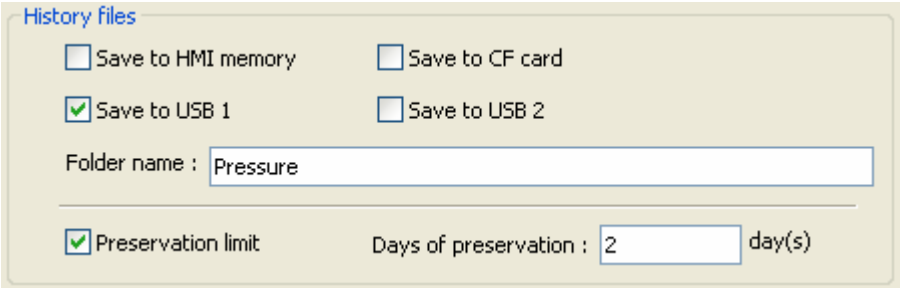


5. [Hold address]

If the status of the designated address is set ON or OFF, sampling will be paused until the status of the designated address returns.



6. [History files]



[Save to HMI momery]

Save Data Sampling to HMI only when its size reaches 4kb. Or, use system register [LB-9034] to force storing data.

[Save to SD card / USB 1 / USB 2]

Save Data Sampling to the specified external device.

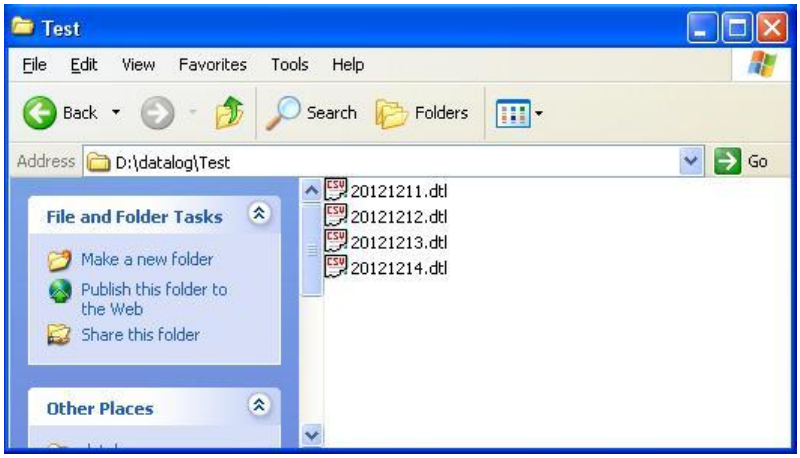
[Folder name]

Specify Data Sampling file name which must be all in ASCII characters.

The folder name will be written as: **[Storage Location] \ [Folder Name] \ yyyymmdd.dtl**

The files are saved in the specified folder according to the date the file is generated.

As shown in the following figure, the files are saved in the **Test** folder according to date.



[Preservation limit]

This determines how many days the data is to be preserved.

- If [Preservation limit] is set to 2 days, the data of yesterday and the day before yesterday will be kept. Data that is not built in this period will be deleted to prevent the storage space from running out. EX: if today were July 1st, data of June 30th and June 29th will be preserved and data of June 28th be deleted.
- When running simulation on PC, all data sampling will be saved to the **datalog** folder in the storage location.

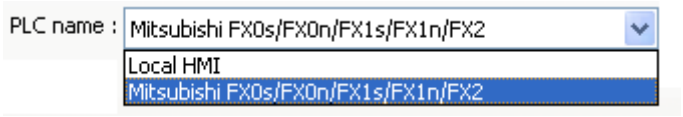
Chapter 9 Object General Properties

The setting of general properties of an object includes:

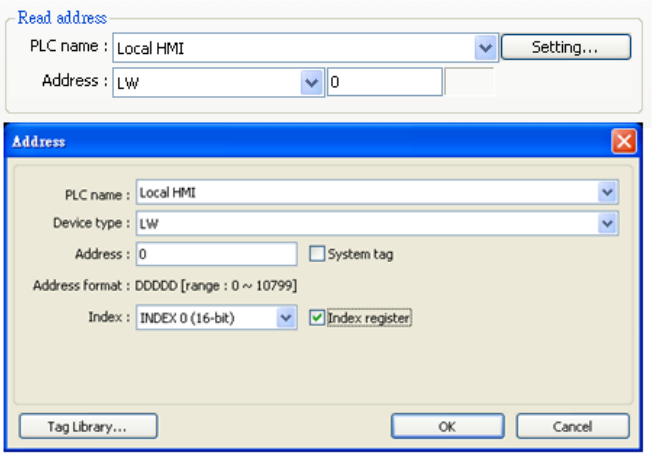
1. Selecting the PLC.
2. Setting read and write address
3. Using Shape Library and Picture Library
4. Setting label text
5. Adjusting profile size

9.1 Selecting the PLC

Some objects are for controlling PLCs. As shown, **[PLC name]** represents the PLC to control. In this example there are two devices: “Local HMI” and “Mitsubishi FX0s/FX0n/FX1s/FX1n/FX2.” The listed available devices come from **[System Parameters Settings]** » **[Device List]**



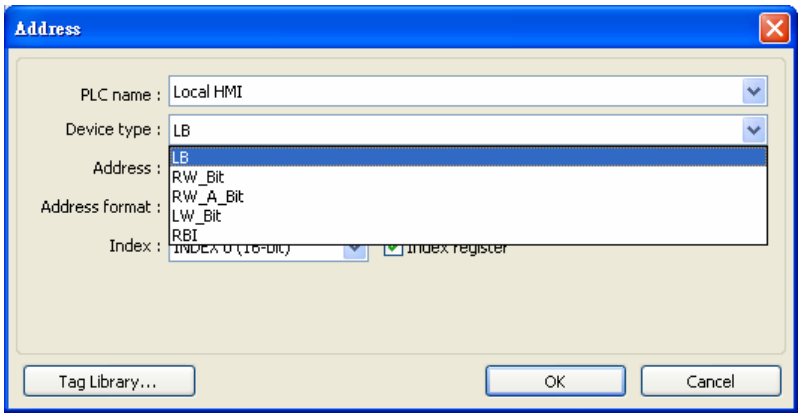
9.1.1 Setting Read and Write Address



The settings of read and write address:

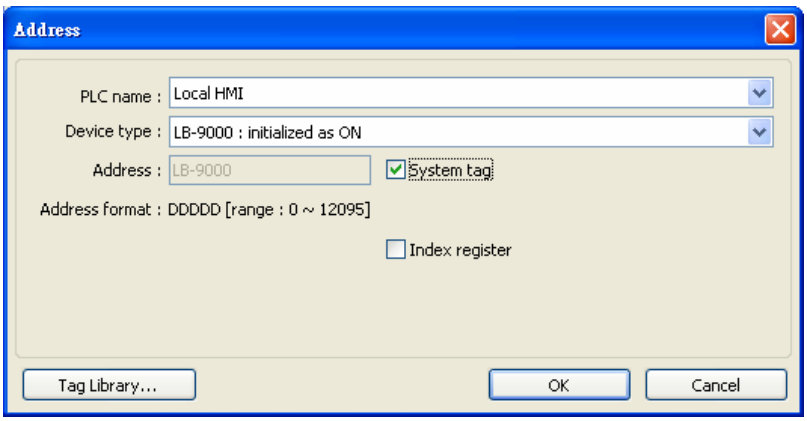
[PLC name]
Select the PLC type.

[Device type]
Different PLC has different device type.

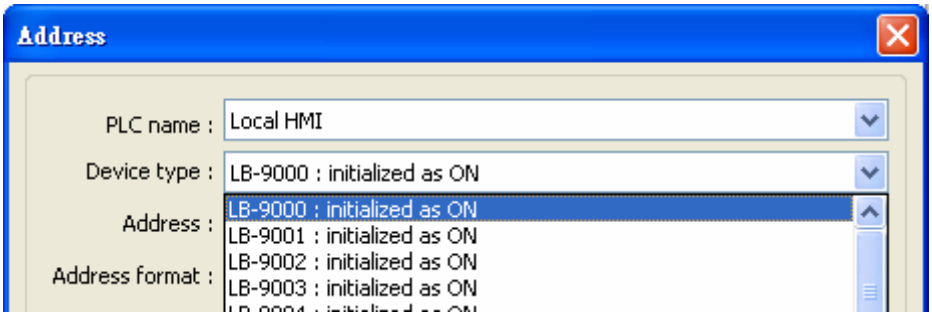


[Address]
Set the read and write address.

[System tag]
Address tags include “system tag” and “user-defined tag.” Click **[Setting…]** beside **[PLC name]** and select **[system tag]** check box**.** This allows users to use the preserved addresses by system for particular purpose.
The address tags are divided into bit or word (LB or LW).
After selecting **[System tag],** not only will the **[Device type]** displays the chosen tag, **[Address]** will also display the chosen tag as shown below.



The illustration below shows a part of system tags. For more information, please refer to “Chapter 16 Address Tag Library” and “Chapter 22 System Reserved Words and Bits”.

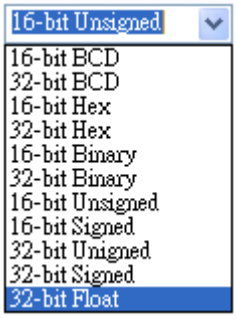


[Index register]

Select this check box to use the index register, please refer to “Chapter 11 Index Register” for more information.

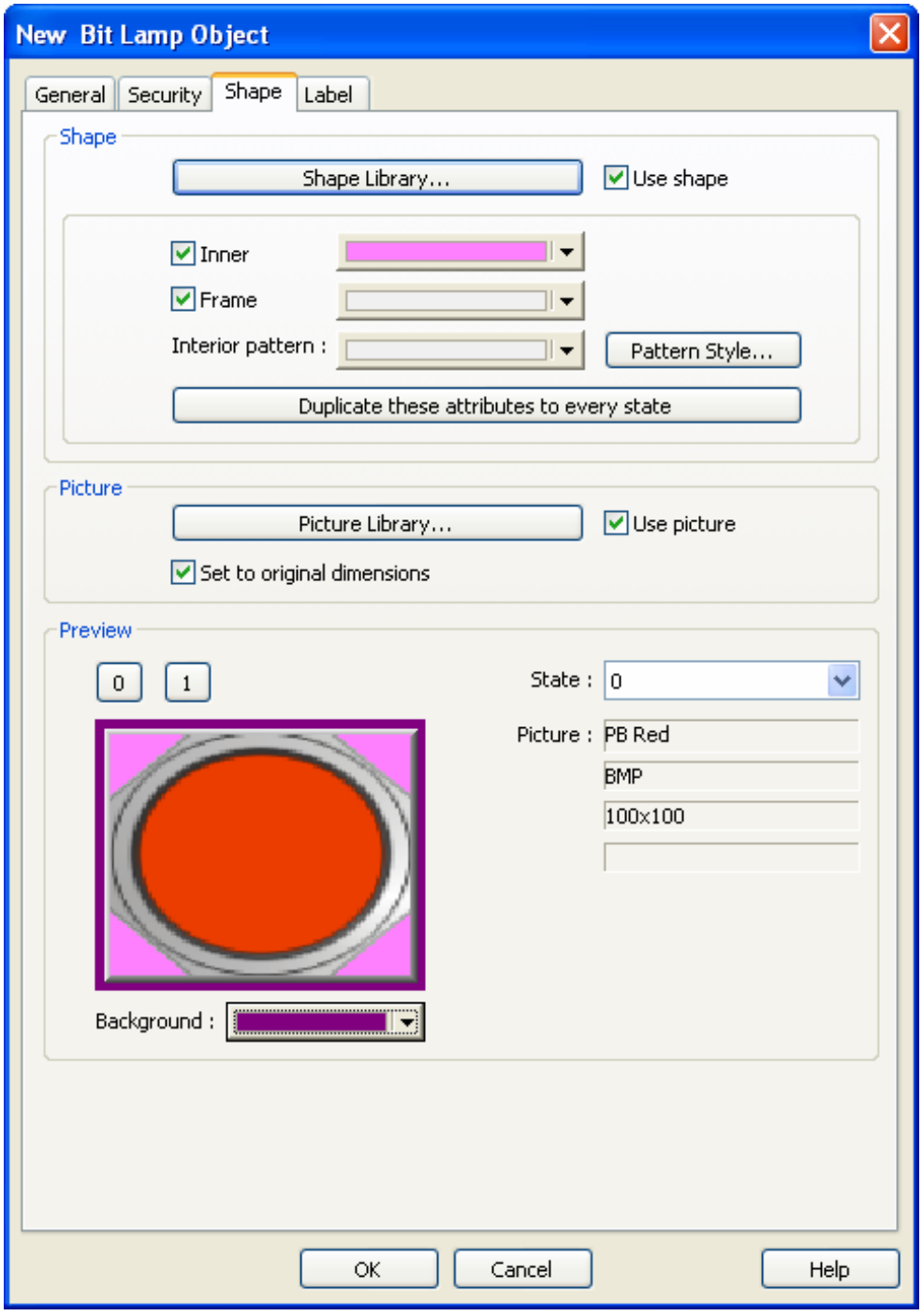
Selecting Data Type

EasyBuilder supports data types that are listed below. Selecting correct data type is necessary especially when using address tag.



9.2 Using Shape Library and Picture Library

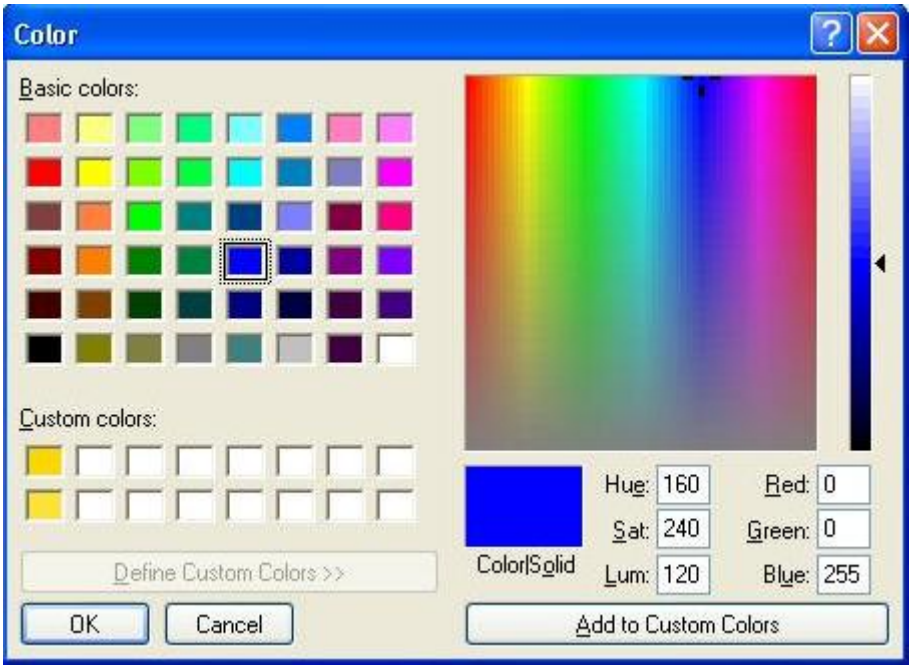
Shape Library and Picture Library are used for adding visual effect on objects. Select **[Shape]** tab when creating an object to use the libraries:



9.2.1 Settings of Shape Library

[Shape Library…]
Select **[Use shape]** check box to select a shape from the library.

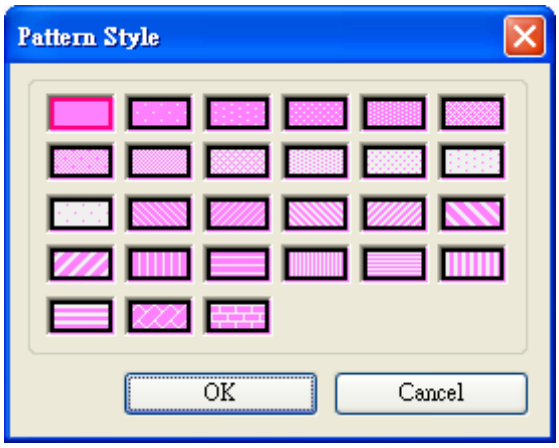
[Inner]
Select this check box to set the inner part of a shape. Click the drop down button to select a color or customize a color and click **[Add to Custom Colors]**. EasyBuilder will save this color.



[Frame]
Select this check box to set the frame of a shape. Click the drop down button to select a color.

[Interior Pattern]
Set the color of the interior pattern of the shape.

[Pattern Style]
Click this button to select a pattern.

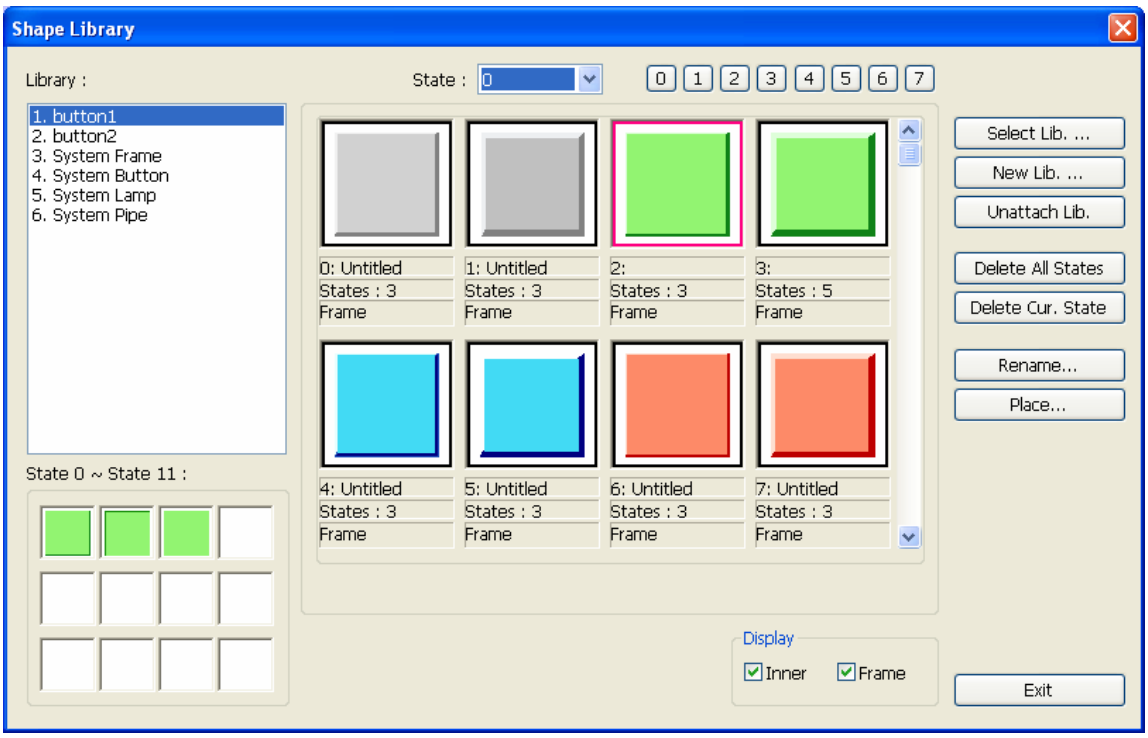


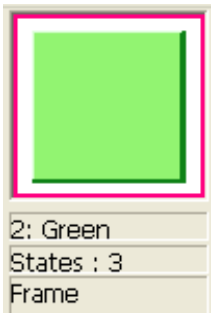
[Duplicate these attributes to every state]

Duplicate all attributes of the current state to the other states.

How to set [Shape Library…]

Click [Shape Library…] button. The currently selected shape is marked by a red frame.

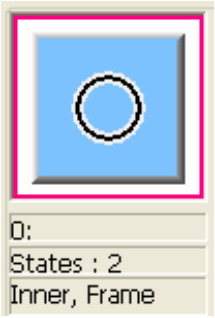




The illustration above provides information of one of the Shapes in the Shape Library as follows:

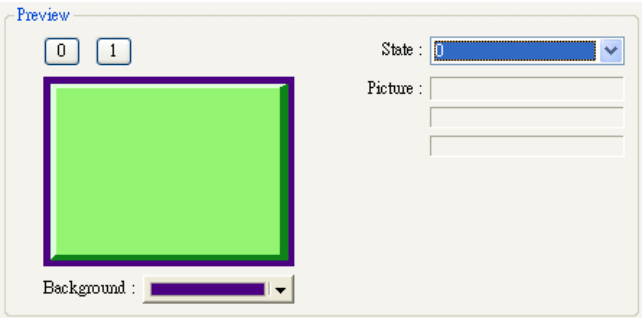
2: Green	The number and the name of the shape.
States: 3	The number of the states of the shape.
Frame	Indicates that the Shape only has a frame.

The illustration below shows that the Shape has “inner” and “frame.”



Please refer to “Chapter 14 Shape Library and Picture Library” for more details.

When finished, click **[OK]** and preview the design of the shape.



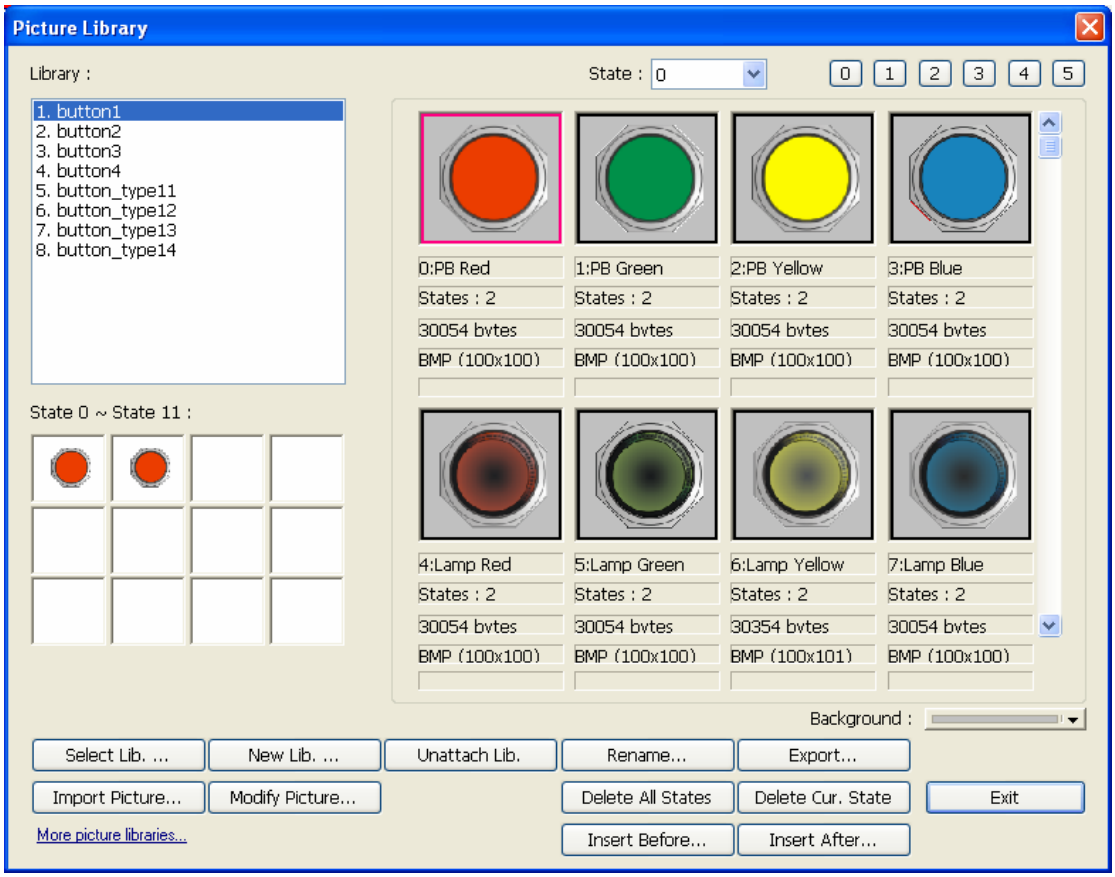
9.2.2 Settings of Picture Library

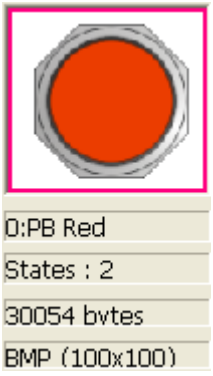
[Picture Library]

Select **[Use picture]** check box to select a shape from the library.

How to set [Picture Library…]

Click **[Picture Library…]** button. The currently selected picture is marked by a red frame.



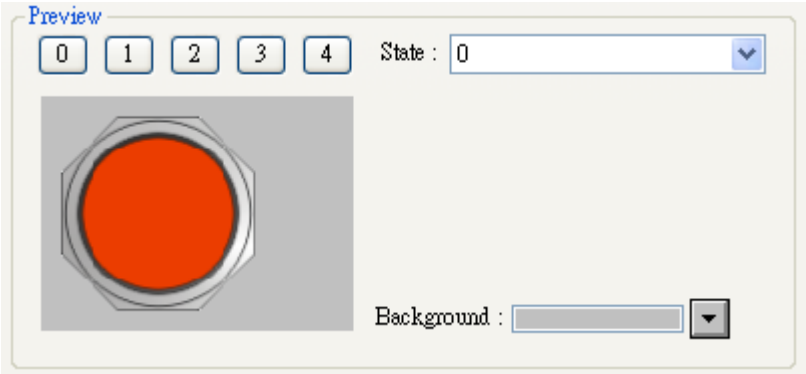


The illustration above provides information of one of the Pictures in the Picture Library as follows:

0 : PB Red	The number and the name of the Picture
States: 2	The number of the states of the Picture
30054 bytes	The size of the Picture
BMP (100x100)	The format and resolution of the Picture; *.bmp means bitmap picture. The format can also be *.jpg, *.png, *.dpd, or *.gif. Picture length: 100 pixels and height: 100 pixels.

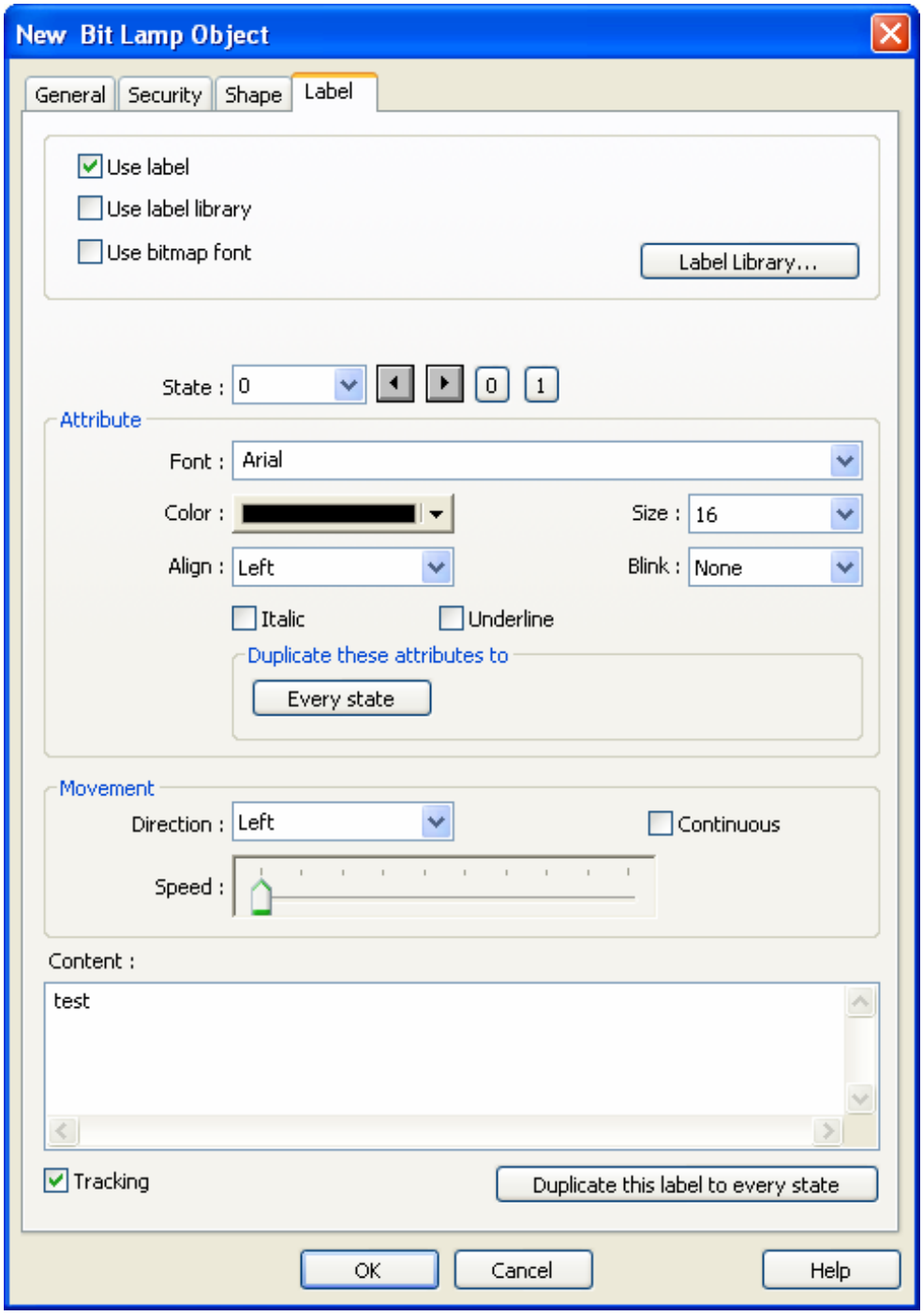
Please refer to “Chapter 14 Shape Library and Picture Library” for more details.

When finished, click **[OK]** and preview the design of the picture.



9.3 Setting Label Text

Select **[Label]** tab when creating an object to use the libraries:

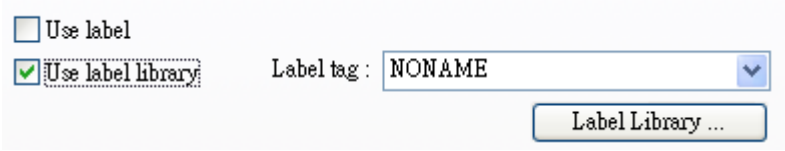


[Use label]

Select this check box to use a label for the object. EasyBuilder supports Windows true-font.

[Use label library]

Select this check box to choose a label in Label Library.

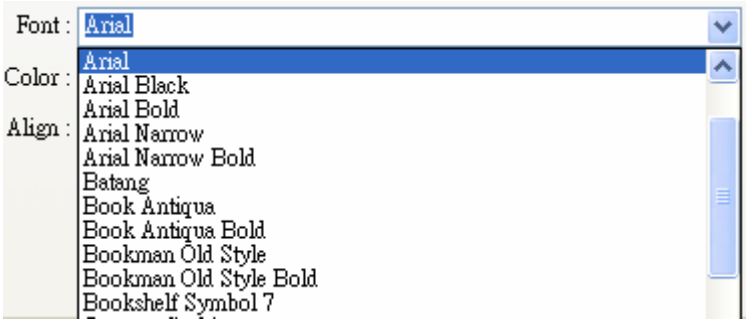


[Label Library…]

Please refer to “Chapter 15 Label Library and Multi-Language Usage” for more details.

[Font]

Select a font from the list. EasyBuilder supports Windows true-font.



[Color]

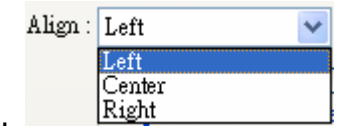
Select the font color.

[Size]

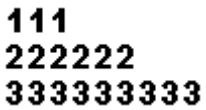
Select the font size.

[Align]

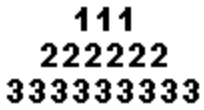
Align the multiple lines of the text.



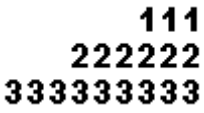
The text aligned **[Left]**.



The text aligned **[Center]**.



The text aligned **[Right]**.



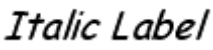
[Blink]

Specify the way the text blinks:

Choose **[None]** to disable this feature or set the blinking interval to **[1 second]** or **[0.5 seconds]**.

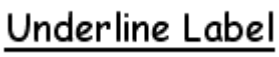
[Italic]

Use Italic font.



[Underline]

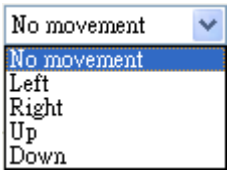
Use Underline font.



[Movement] setting

[Direction]

Set the direction of the marquee effect.



[Continuous]
Specify how the marquee effect is displayed:

If **not** selecting this check box, the next text appears only when the previous text disappears completely. See the picture below.

If selecting this check box, the text will be displayed continuously.

[Speed]
Adjust the speed of the text movement.

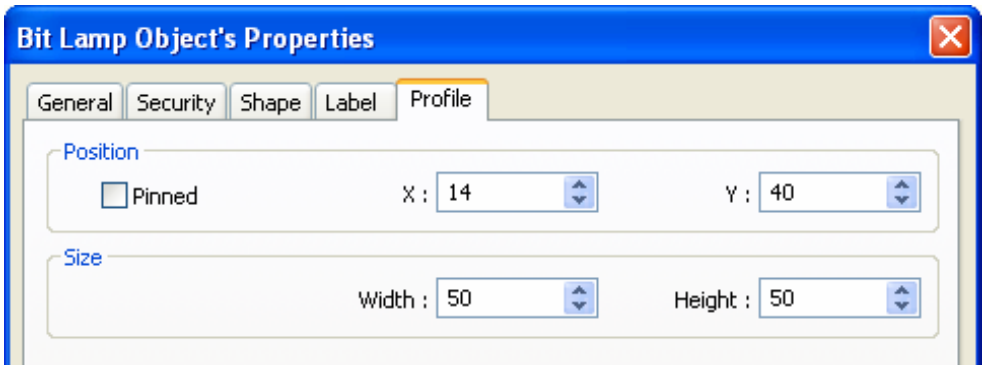
[Content]
Set the content of the text. If using **[Label Library]**, the content comes from Label Library.

[Tracking]
If this check box is selected, changing the text of one state will also change the text of the other states.

[Duplicate this label to other states]
Duplicate the current text to the other states.

9.4 Adjusting Profile Size

When an object is created, double click it and select the [Profile] tab to adjust the position and size of the object.



Position

[Pinned] When this check box is selected, the position and the size of the object cannot be changed.

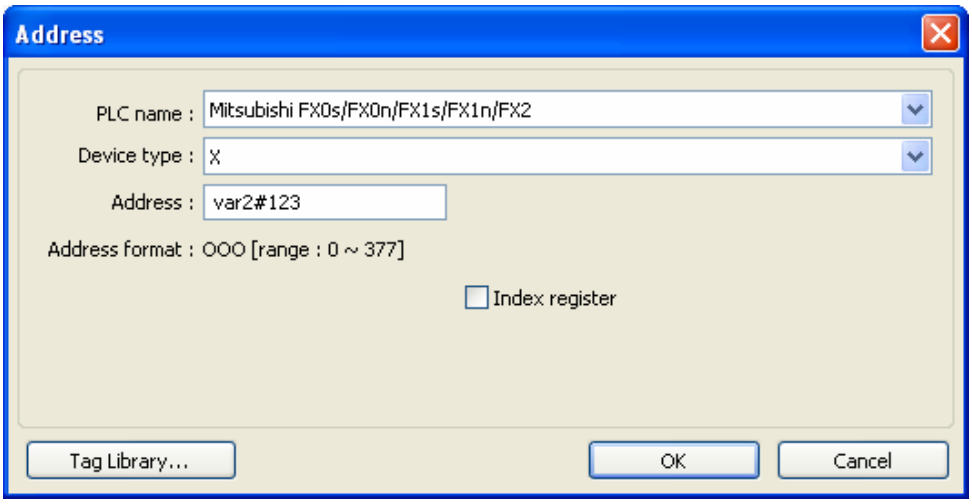
[X] and **[Y]** are coordinates of the left-top corner of the object.

Size

Adjust the **[width]** and **[height]** of the object.

9.5 Variables of Station Number

EasyBuilder allows users to set the variables of station number in PLC address. As shown below, “var2” is one of 16 station number variables.



The syntax of the variable of a station number: varN#address

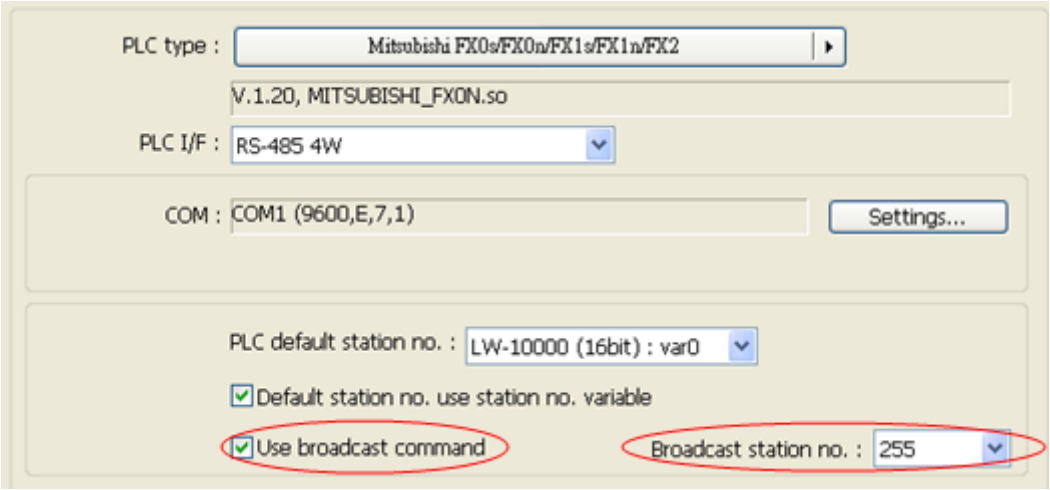
The range of N is integer from 0~15; address means PLC address.
16 variables are availble：var0 ~ var15. These variables of station number read values from address LW-10000~LW-10015. The list below shows the variables and the corresponding system reserved addresses：

var0	LW-10000	var8	LW-10008
var1	LW-10001	var9	LW-10009
var2	LW-10002	var10	LW-10010
var3	LW-10003	var11	LW-10011
var4	LW-10004	var12	LW-10012
var5	LW-10005	var13	LW-10013
var6	LW-10006	var14	LW-10014
var7	LW-10007	var15	LW-10015

For example, “var0” reads value from LW-10000, when the value in LW-10000 is “32”, var0#234 = 32#234, the station number is 32.
Similarly, “var13” reads value from LW-10013, when the value in LW10013 is” 5”, var13#234 = 5#234.

9.6 Broadcast Station Number

HMI provides two ways for users to enable broadcast command. First is to set the PLC parameter directly in **[system parameter settings]**:



The second way is to use system tag to enable or disable broadcast station number or to change it.
The corresponding system tags are listed below:

LB-9065 disable/enable COM 1 broadcast station no.
LB-9066 disable/enable COM 2 broadcast station no.
LB-9067 disable/enable COM 3 broadcast station no.

LW-9565 COM 1 broadcast station no.
LW-9566 COM 2 broadcast station no.
LW-9567 COM 3 broadcast station no.

Chapter 10 User Password and Object Security

This chapter discusses the protection for operations provided by setting up user passwords and security classes.

To set up the protection system, please:

1. Set user password and operable classes.
2. Set object class for objects.

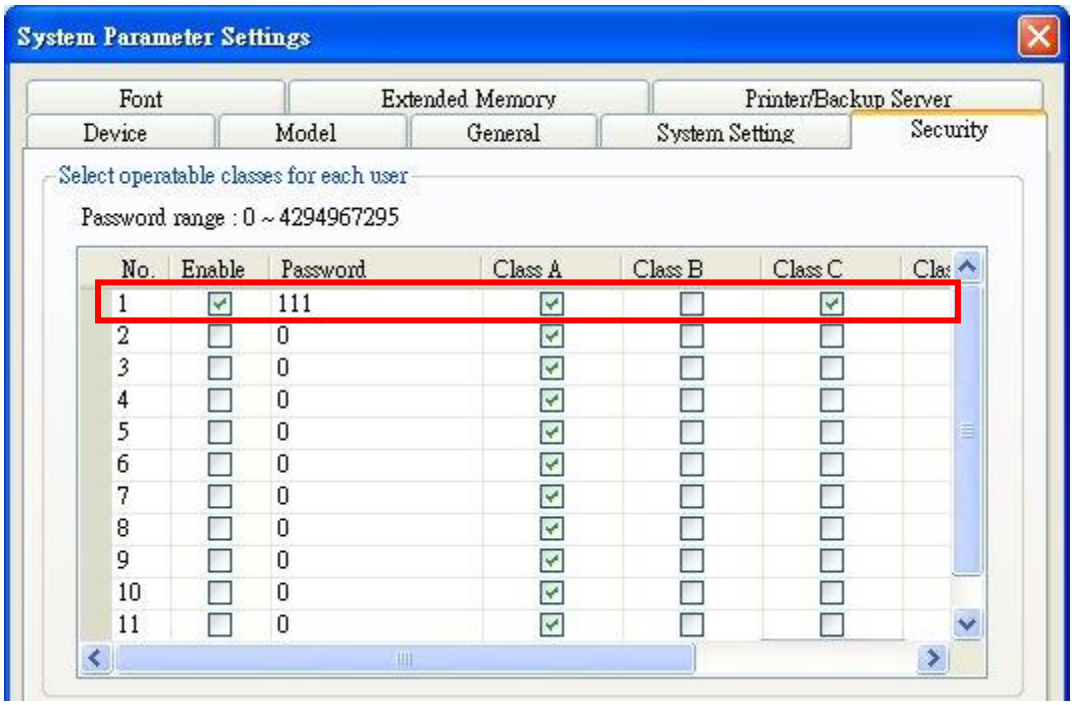
An object belongs only to one security class. Setting the object class to “None” means any user can operate this object.

10.1 User Password and Operable Object Classes

The security parameters can be found in **[Edit] » [System Parameter Settings] » [Security]**.

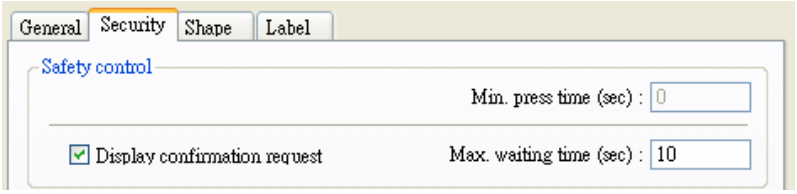
Up to 12 sets of user and password are available. Password should be one non-negative integer. There are six security classes: A to F.

Once the password is entered, the objects that the user can operate are classified. As shown below, “User 1” can only operate objects with class A or class C.

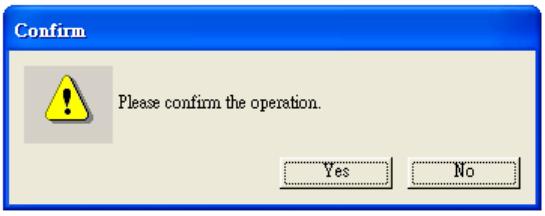


10.2 Object Security Settings

[Safety control]

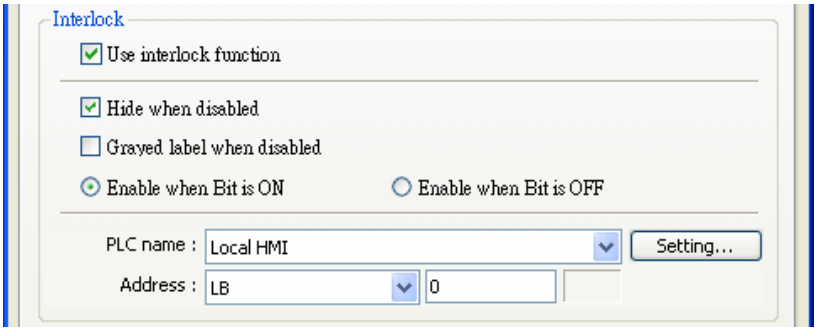


Use **[Min. press time (sec)]** to avoid miss operations. Press and hold the object longer than the [Min. press time] set here to activate the object.



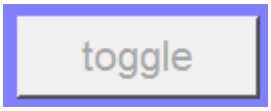
[Display confirmation request] After pressing the object, a dialog appears for operation confirmation. If the response to this dialog comes later than the set **[Max. waiting time (sec)]**, this dialog disappears automatically and the operation will be canceled.

[Interlock]



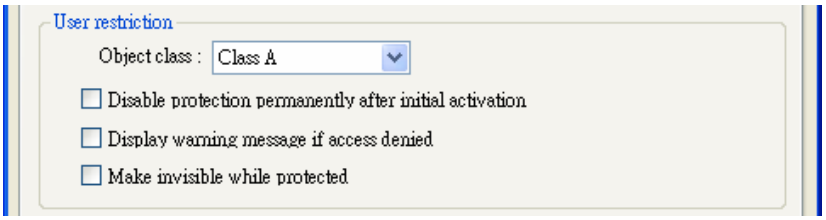
When this check box is selected, the specified Bit address is used to enable or disable the object. As shown, if LB-0 is ON, the object is enabled.

[Hide when disabled] When the specified Bit is OFF, hide the object.



[Grayed label when disabled] When the specified Bit is OFF, the label font of the object turns gray.

[User restriction]



Set the security class of the object to be operated by an authorized user.

[Object class]

"None" means any user can operate this object.

[Disable protection permanently after initial activation] Once the permitted class of the user matches that of the object, the system will stop checking the security class permanently, that means, any user can operate this object freely after it is unlocked.

[Display warning message if access denied] When an unauthorized user attempts to operate the object, a warning dialog (Window no. 7) appears. The content of the message in the dialog can be modified.

[Make invisible while protected] When the user's privilege does not match the object class, hide the object.

10.3 Example of Object Security Settings

The following shows an example of setting object security class:

1. Create a project, go to **[System Parameter Settings] » [Security] » [General]** to enable 3 users:

User 1 =

Operable class: A

User 2 =

Operable class: A, B

User 3 =

Operable class: A, B, C

2. Design Window no. 10 as shown:

Create two **[Numeric Input]** objects:

[LW-9219]

User no. (1~12)

Length = 1word

[LW-9220]

For entering user password.

Length = 2 words

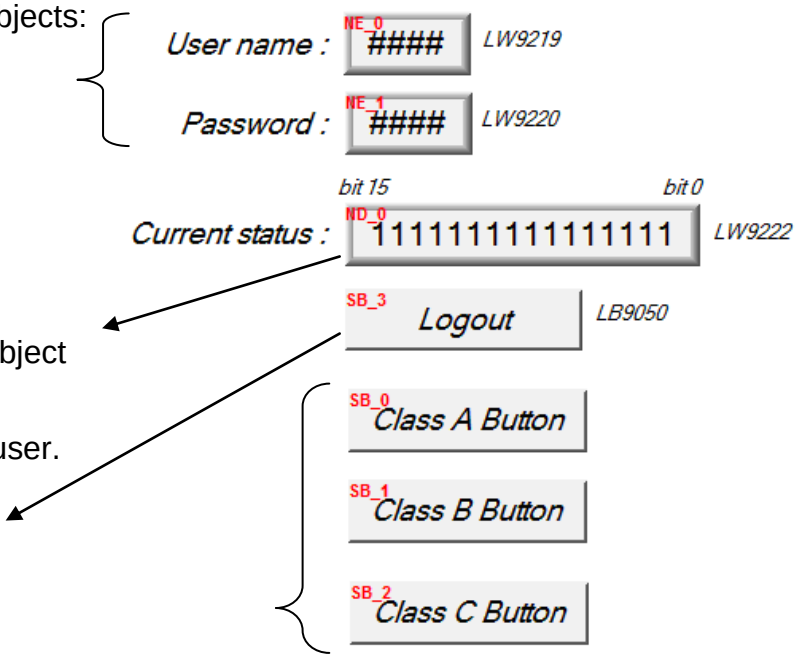
Create a **[Numeric Display]** object

[LW-9222]

Displays the status of current user. (16-bit Binary)

Create a **[Set Bit]** object

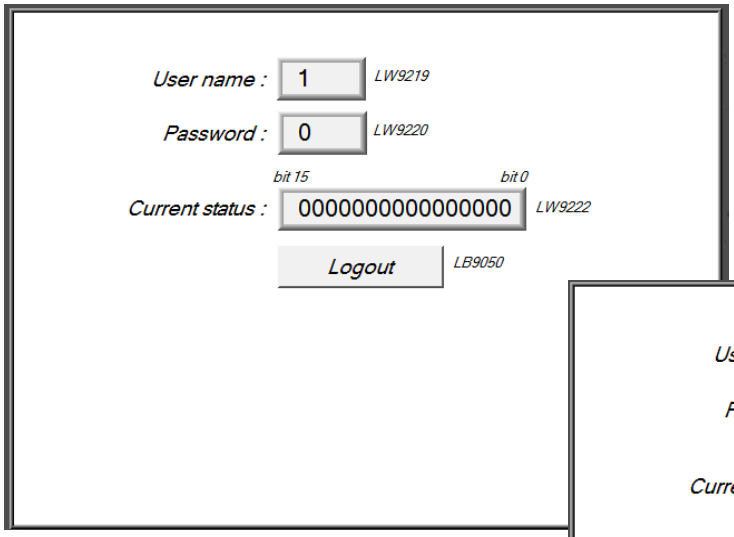
[LB-9050] logout



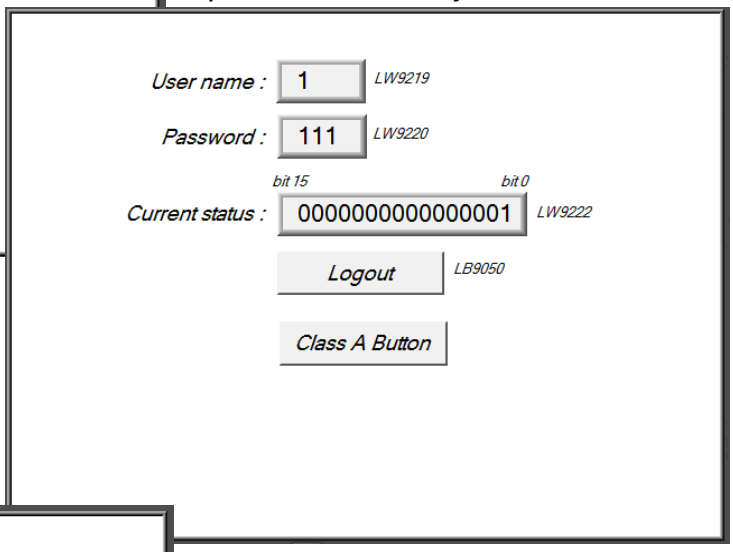
Create three **[Set Bit]** objects, each set to different classes but all select **[Made invisible while protected]**.

After setting, please save and compile the project and execute off-line simulation. The below shows how it works when simulating.

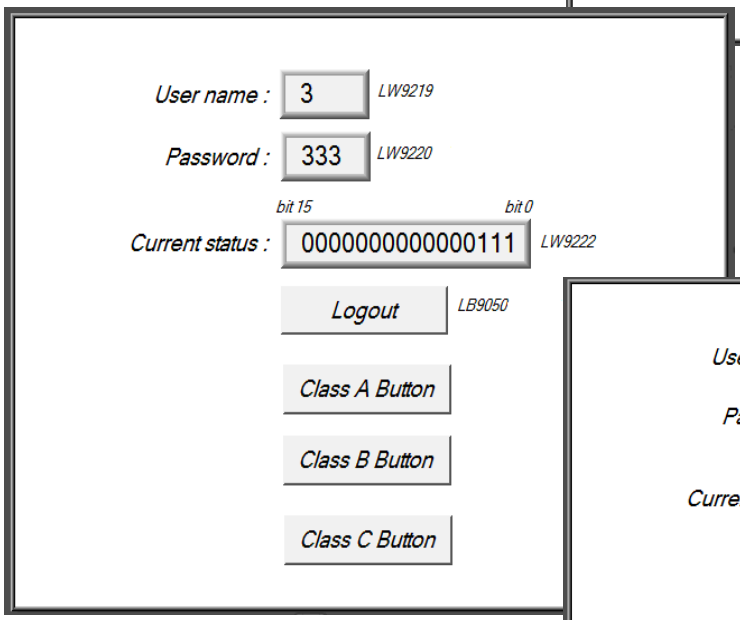
3. Before entering the password, it displays “0000000000000000”, which means that the user operable object class is “None”. [Class A Button] ~ [Class C Button] objects are classified from “A” to “C” and selected **[Made invisible while protected]**; therefore they are hidden at this moment.



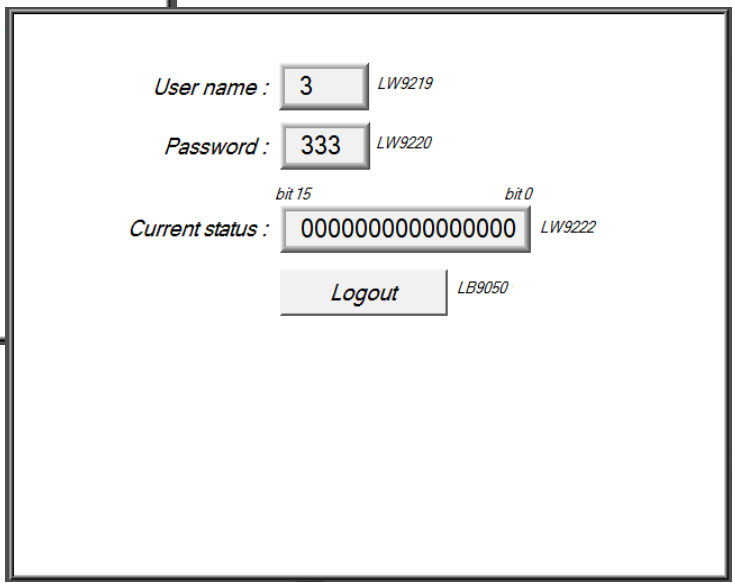
4. Enter User 1 password “111”. Since User 1 is only allowed to operate class A objects, [Class A Button] object appears for operating. [LW-9222] bit 0 turns to “1” means that user can operate class A objects.



5. Enter User 3 password “333”. Since User 3 is allowed to operate class A, B, C objects, [LW-9222] bit 0 ~ bit 2 turns to “1”, means that user can operate class A ~ C objects.



6. Click [Logout] button to log out, the system will return to the initial state, and current user can only operate class “None” objects.



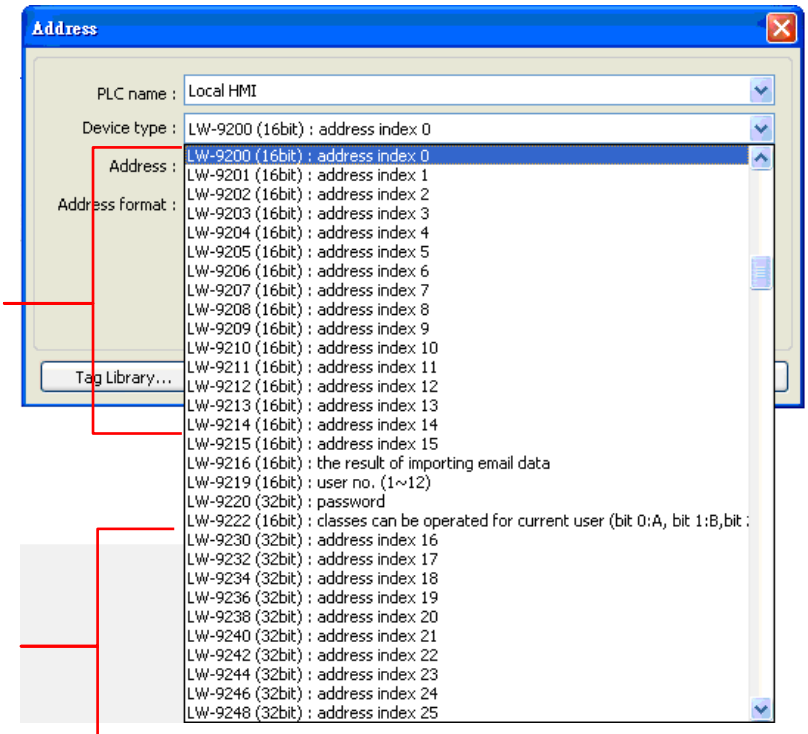
■ **Password input:** If the password is incorrect, [LB-9060] will be ON; if the password is correct, [LB-9060] will be OFF. All user passwords (User 1 to User 12) can be obtained from system registers [LW-9500] ~ [LW-9522], 24 words in total.

■ **Changing password directly on HMI:** When [LB-9061] is set ON, the system will read data in [LW-9500] ~ [LW-9522] to update user password. The new password will be used in the further operations. Please note that the user operable object classes will not be changed due to the change of password.

Chapter 11 Index Register

11.1 Introduction

EasyBuilder provides Index Registers for users to change addresses flexibly. With Index Registers, users can change object's read/write address directly on HMI without changing its settings. There are 32 Index Registers, divided into 16-bit and 32-bit.



16-bit Index Register:
Index 0 [LW-9200] (16-bit)~
Index 15 [LW-9215] (16-bit)
Max. address range: 65536 words.

32-bit Index Register:
Index 16 [LW-9230] (32-bit)~
Index 31 [LW-9260] (32-bit)
Max. address range: 4294967296 words.

While using **[Index register]**, the address is designated by the following equation:
The constant set in **[Address]** + the value in the chosen Index Register.

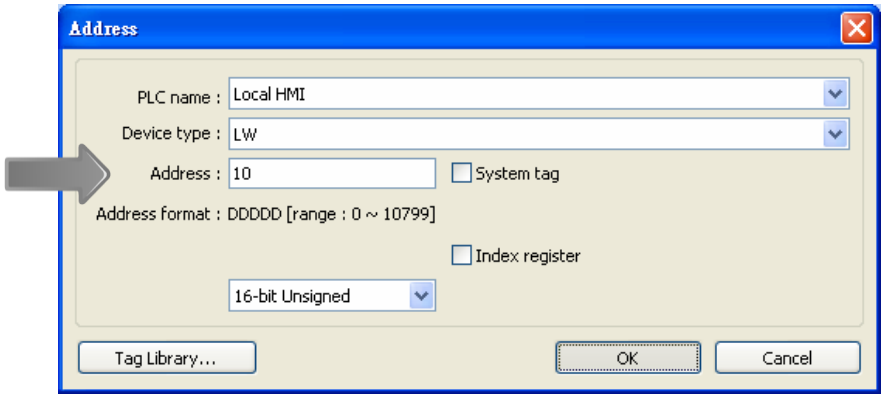
Note

Index Registers work for all devices in **[System Parameter Settings] » [Device list]**, but are limited to word registers only.

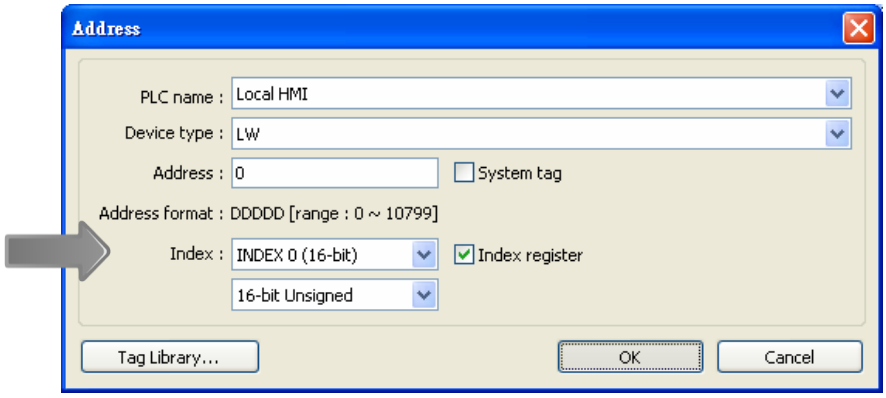
11.2 Examples of Index Register

The following examples show how to use Index Registers.

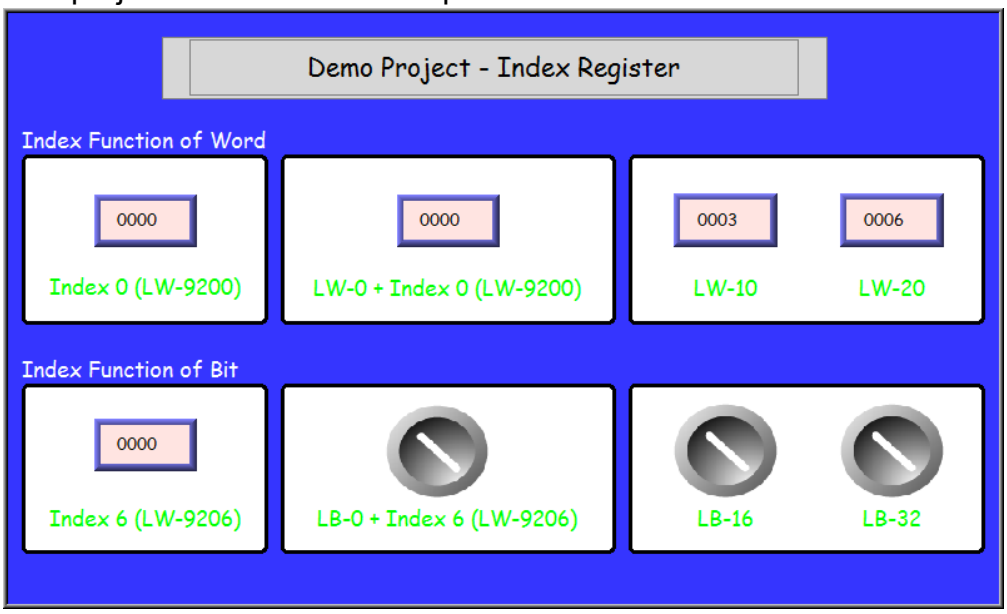
If **not** selecting **[Index register]** check box and set read address to [LW-10]. The system will directly read / write LW-10.



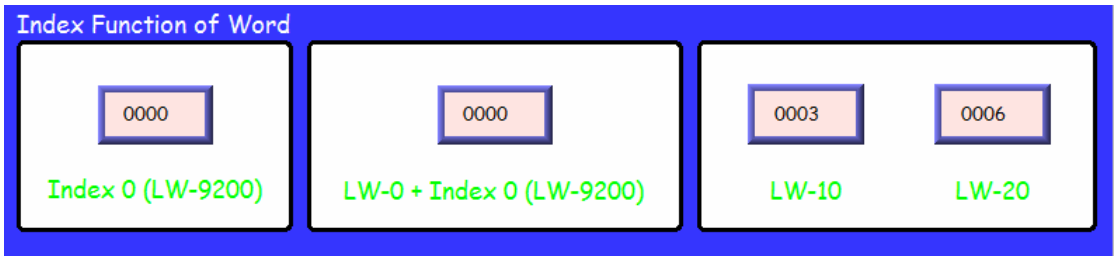
If select **[Index register]** check box and set index register to **[INDEX 0]**. Read address = [LW-0 + INDEX 0]
As shown in the beginning, Index 0 indicates register [LW-9200]. If the data in [LW-9200] is “5”, read address is set to [LW(0+5)] = [LW-5].



Here’s a demo project shown as an example:



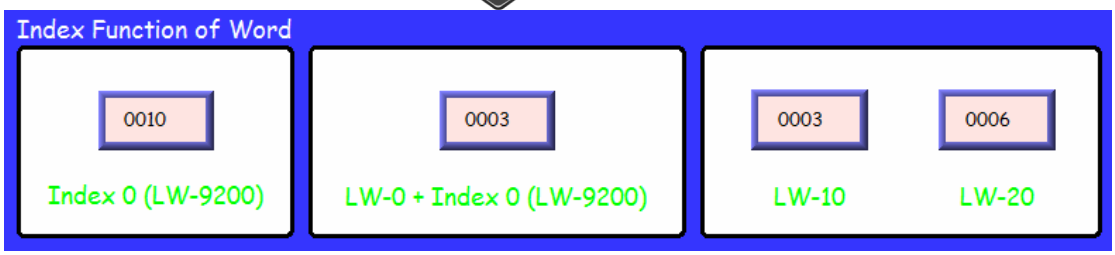
Example 1 Index Function of Word



If the value in Index 0 (LW-9200) is 0.

Read [LW-0 + Index 0] = Read [LW-0]

[LW-10] is set to 3
[LW-20] is set to 6



If the value in Index 0 (LW-9200) is 10

Read [LW-0 + Index 0] = Read [LW-10] = 3

Example 2 Index Function of Bit

In the same way, Index Register can be used for Bit address.
1 Word = 16 Bit, adding 1 in the value of index register = adding 16

Index 6 (LW-9206) is set to 1

The switch [LB-0 + Index 6] reads LB-16 = ON

Index 6 is set to 2

The switch [LB-0 + Index 6] reads LB-32 = OFF

- When using Index Registers for bit addresses, 16 bit addresses will be calculated as one unit. EX: If the target is LB-0, and set the value in Index Register to 1, LB-16 will be activated. If set the value in Index Register to 2, LB-32 will be activated.

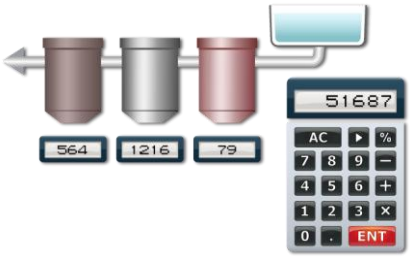
Index Register is used to change addresses. Through changing the data in Index Register, user can make an object to read and write different addresses without changing its own address settings. Therefore user can transmit or exchange data among different addresses.

Please confirm your internet connection before downloading the demo project.

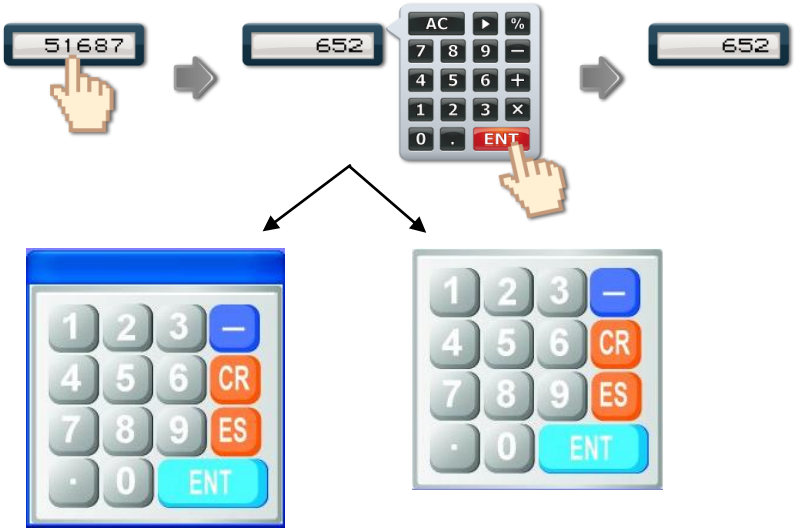
Chapter 12 Keyboard Design and Usage

Numeric Input and ASCII Input objects need keyboard as an input tool.
Both numeric keyboard and ASCII keyboard are created with Function Key object.
The types of the keyboards are:

1. Fixed Keyboard.



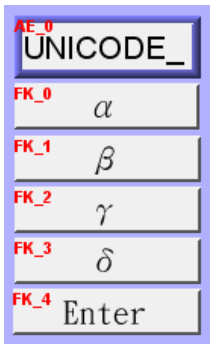
2. Popup Keyboard.



Popup keyboard with title bar.

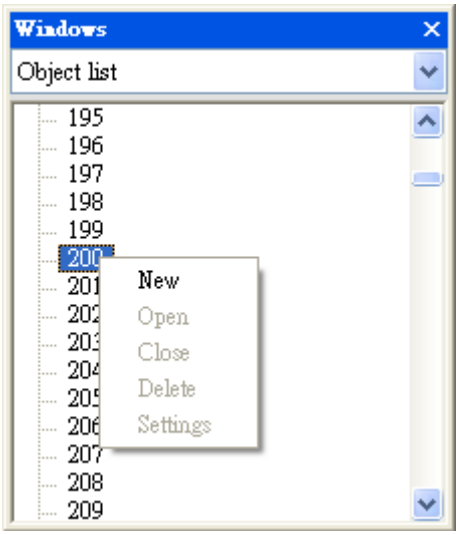
Popup keyboard without title bar.

3. UNICODE Keyboard.



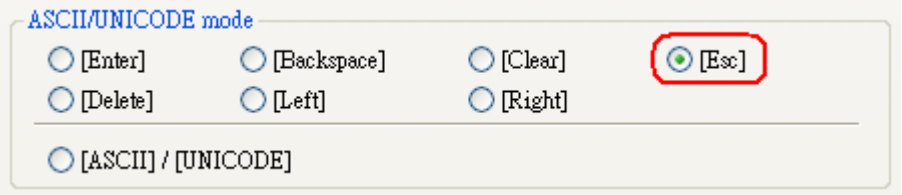
12.1 Steps to Design a Popup Keyboard

Step 1 Create and open a window for the new keyboard. For example, set to “window no. 200”.

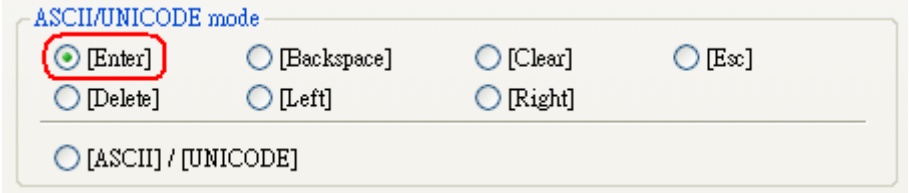


Step 2 Adjust the height and width of “window no. 200” and create a variety of Function Key objects in **[ASCII/UNICODE mode]**.
For example:

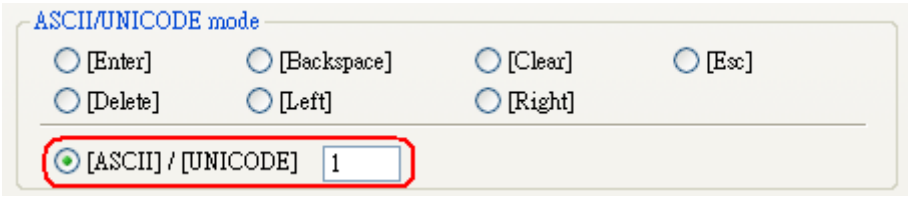
[FK_11] is used as the **[Esc]** key.



[FK_14] is used as the **[Enter]** key.

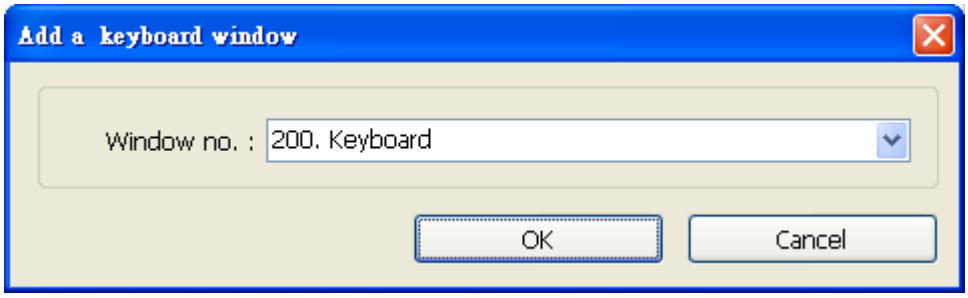


The rest are mostly used to enter numbers. For example, [FK_0] is used for entering number "1".

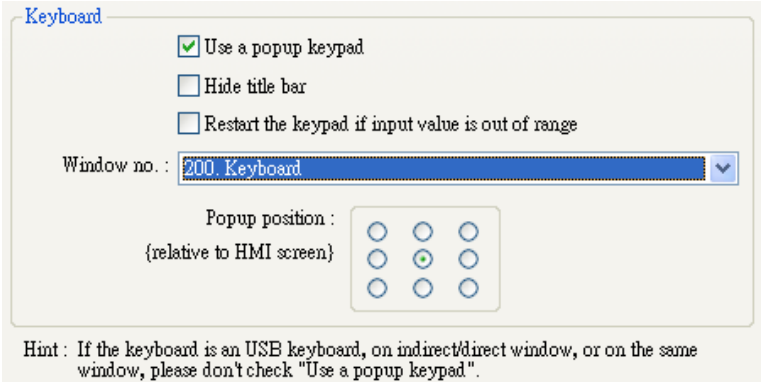


Step 3 Select a suitable picture for each Function Key object and placed at the bottom layer as a background.

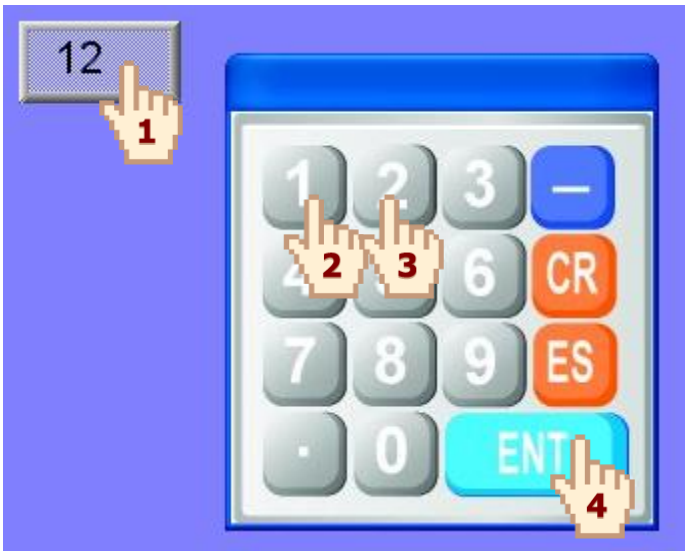
Step 4 Select **[System Parameter Settings] » [General] » [Keyboard] » [Add]** to add "window no. 200". Up to 32 keyboards can be added.



Step 5 After the keyboard window is added, when creating Numerical Input and ASCII Input objects, "200. Keyboard" can be found in **[Data Entry] » [Keyboard] » [Window no.]**. The **[Popup position]** is for designating the display position of the keyboard on the screen. The system divides the screen into 9 areas.

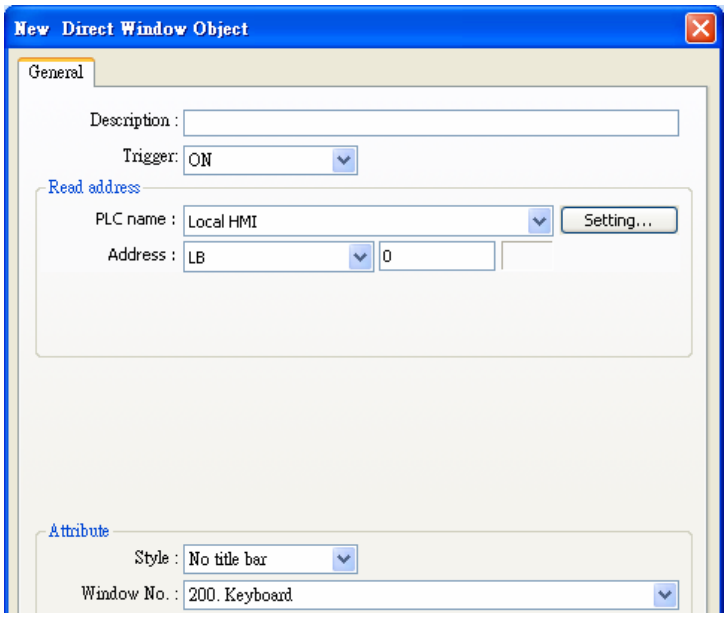


Step 6 Select “200.Keyboard”. When users press Numerical Input or ASCII Input objects on the screen, “window no. 200” will pop up. Users can press the keys on the keyboard to enter data.

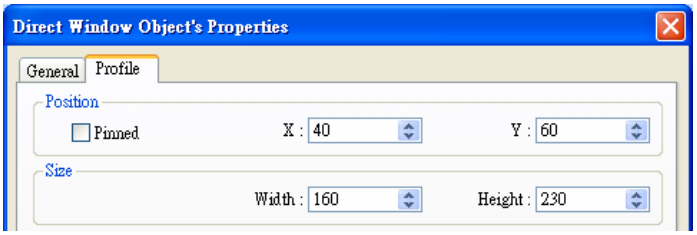


12.2 Steps to Design a Keyboard with Direct Window

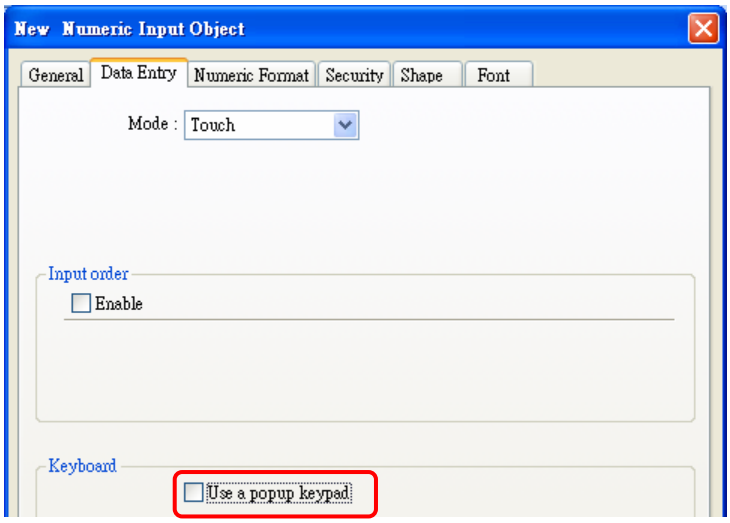
Step 1 Create a Direct Window object and set a read address to activate it.
In **[General] » [Attribute]** select **[No title bar]** and the correct **[Window No.]**.



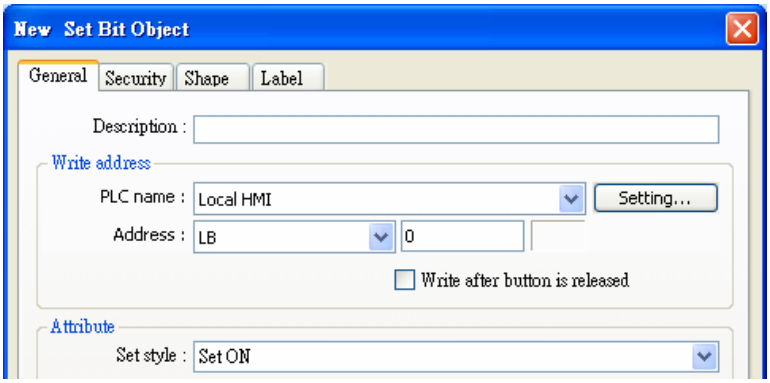
Step 2 Open the setting dialog of Direct Window again to set the [Profile] to the same size as the created keyboard window.



Step 3 Create a Numeric Input object, and **don't** select **[Use a popup keypad]** check box.



Step 4 Create a Set Bit object, set address to **[LB-0]** and set **[Set style]** to **[Set ON]**. Overlay it on the Numeric Input object. Pressing the Numeric Input object will open the keyboard and the Direct Window.

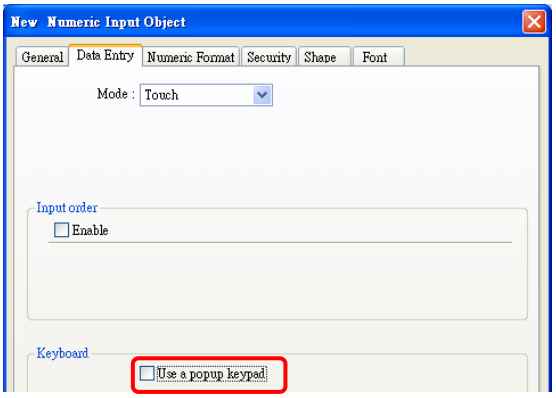


Step 5 Add Set Bit objects on the **[Enter]** and **[ESC]** function keys respectively. Set address to **[LB-0]** and **[Set style]** to **[Set OFF]**. In this way when pressing either [Enter] or [ESC] key will close the keyboard and the Direct Window.

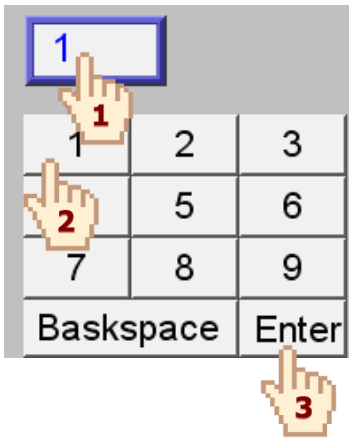
12.3 Steps to Design a Fixed Keyboard on Screen

Users can also place a fixed keyboard on the screen instead of popup keyboard or Direct Window. This type of keyboard can't be moved or closed.

Step 1
Create a Numeric Input object, in **[Data Entry] » [Keyboard] don't** select **[Use a popup keypad]** check box.



Step 2
Use Function Keys to design the keyboard and place it on the screen.



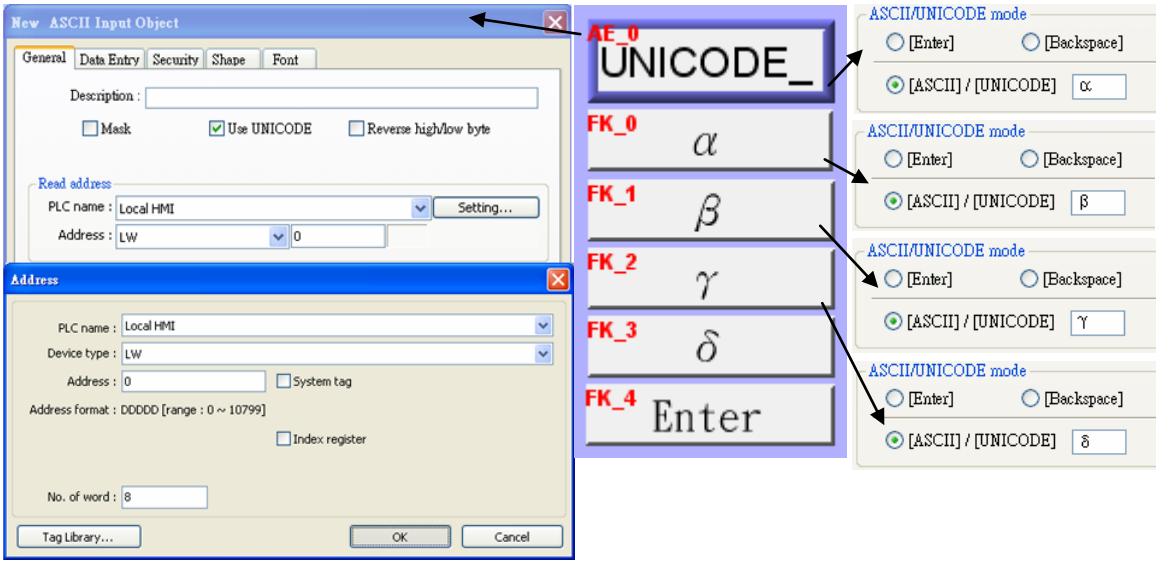
Step 3
Press the Numeric Input object and enter a value with function keys directly.

12.4 Steps to Design a UNICODE Keyboard

Create a UNICODE keyboard with Function Keys.

Step 1 Place an ASCII Input object on the window and select [Use UNICODE] check box.

Step 2 Create Function Keys **[α] [β] [γ] [δ]** as shown, and an **[Enter]** key. A simple UNICODE keyboard is created.



Chapter 13 Objects

This chapter explains the ways of using and setting different kinds of objects. To set the general properties of the objects, please refer to the relevant chapters as listed below:

Shape, Label, Profile

Please refer to “Chapter 9 Object General Properties”.

Security

Please refer to “Chapter 10 Security”

Index Register

Please refer to “Chapter 11 Index Register”

Label Tag Library

Please refer to “Chapter 15 Label Library and Multi-language Usage”

Address Tag Library

Please refer to “Chapter 16 Address Tag Library”

13.1 Bit Lamp

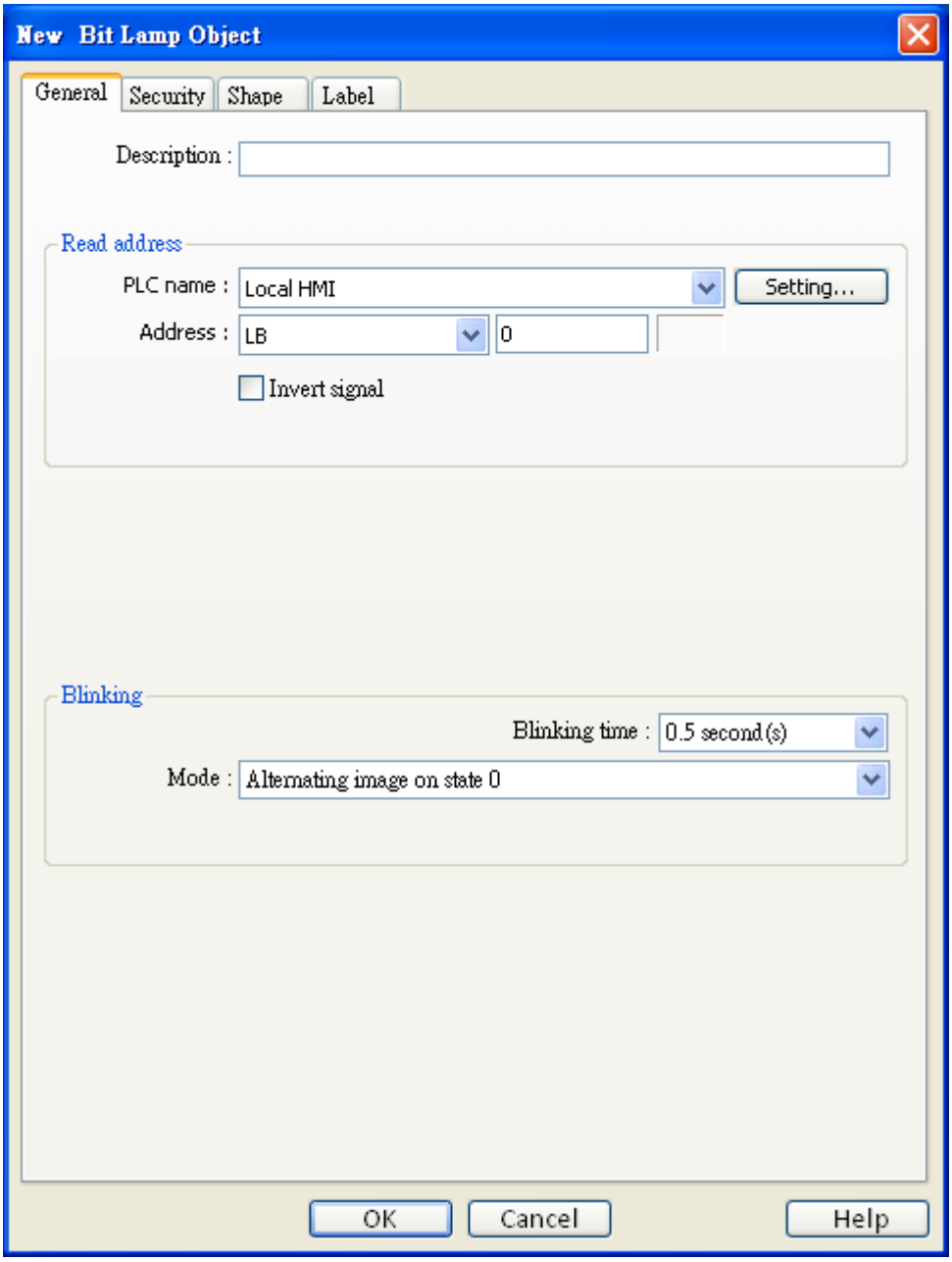
Overview

[Bit Lamp] object displays the state of a designated bit address. If the bit state is OFF, the State 0 shape will be displayed. If the bit state is ON, the State 1 shape will be displayed.

Configuration

Click the [Bit Lamp] icon on the toolbar to open a [Bit Lamp] object property dialog. Set up the properties, press OK button, and a new [Bit Lamp] object will be created.

.

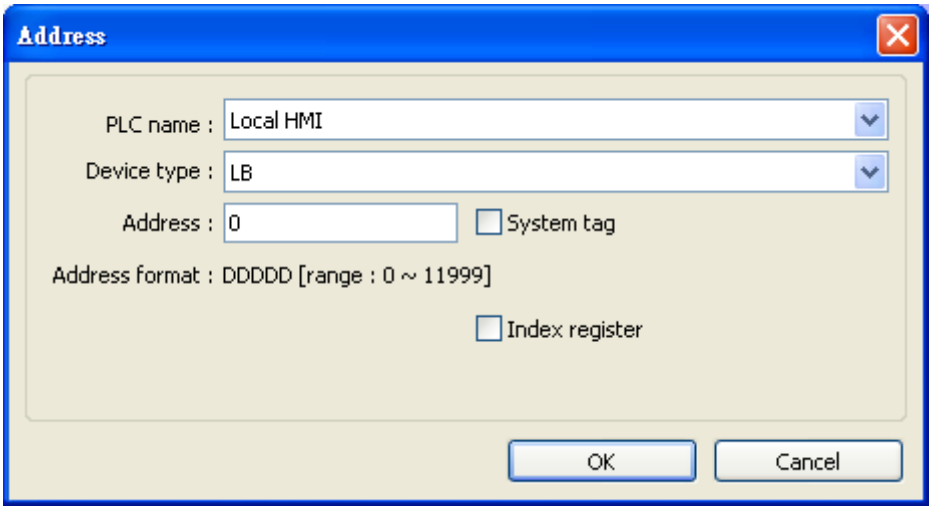


Description

User can describe the information of the object.

Read address

Click [Setting] to select the [PLC name], [Address], [Device type], [System tag], [Index register] of the bit device that controls the [Bit Lamp] object. Users can also set address in [General] tab while adding a new object.



[Invert signal]

Reverses the display of ON / OFF states. For example, if [Invert signal] check box is selected, when the designated bit is OFF, the object displays ON state.

Blinking

The appearance of the object may alternate between states when the bit is ON or OFF.

a. None

No blinking.

b. Alternating image on state 0

The appearance of the object alternates between State 0 and 1 when the bit is OFF.

c. Alternating image on state 1

The appearance of the object alternates between State 0 and 1 when the bit is ON.

d. Blinking on state 0

The State 0 appearance of the object will blink when the bit is OFF.

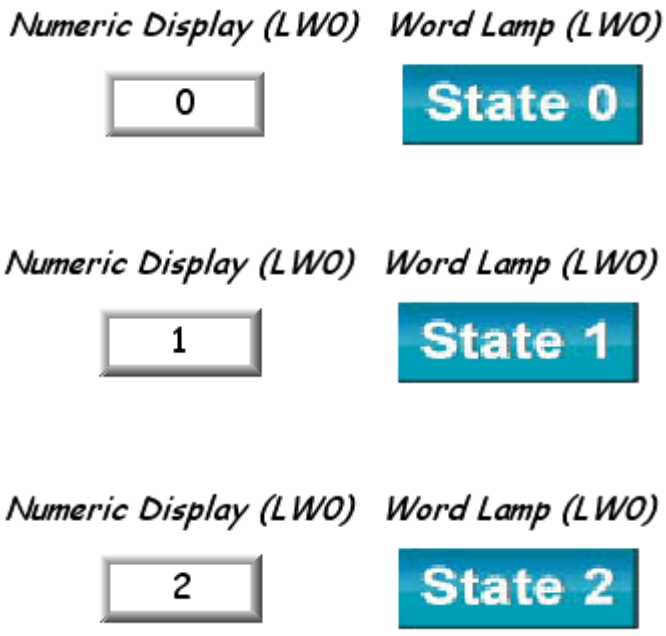
e. Blinking on state 1

The State 1 appearance of the object will blink when the bit is ON.

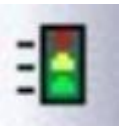
13.2 Word Lamp

Overview

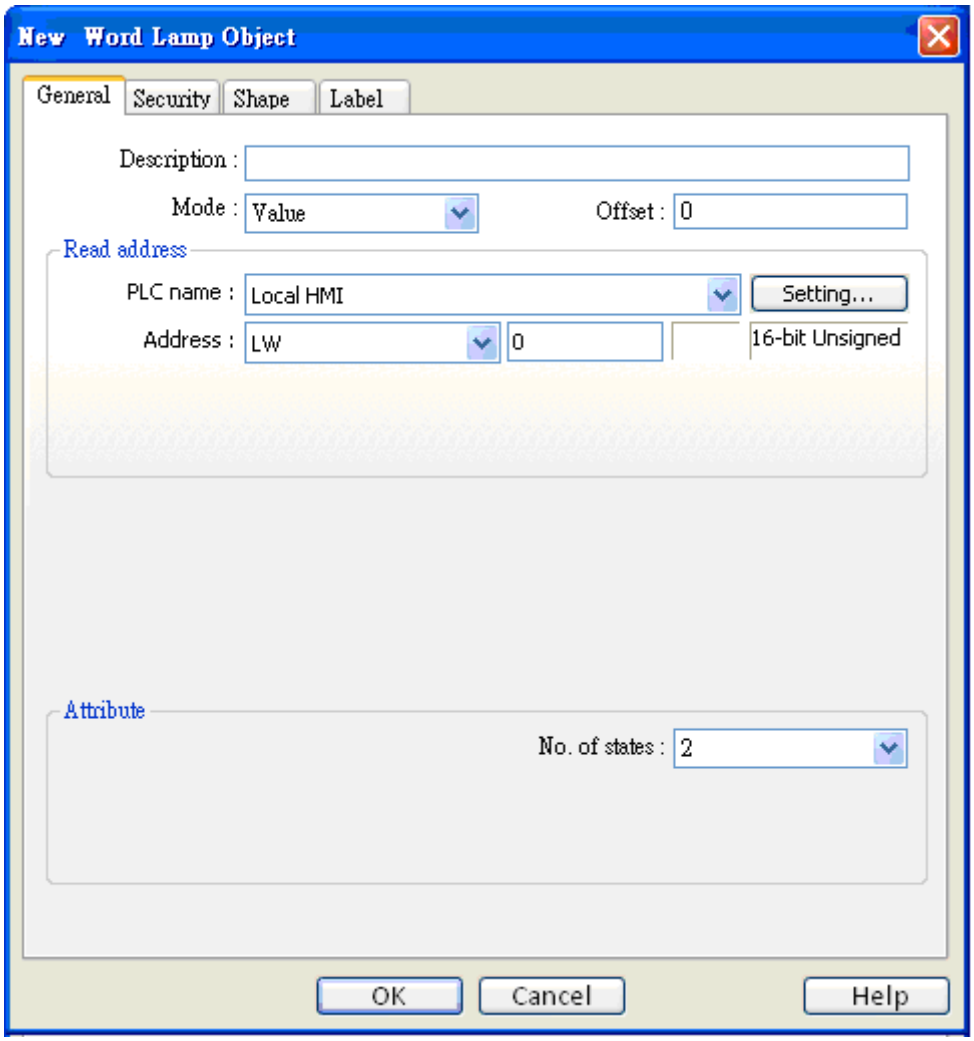
[Word Lamp] object displays the state according to the value of a designated word register. Up to 256 states are available. When the value of the register is 0, State 0 appearance of the object is displayed, and value 1 displays State 1, and so on.



Configuration



Click the [Word Lamp] icon on the toolbar to open a [Word Lamp] object property dialog. Set up the properties, press OK button, and a new [Word Lamp] object will be created.



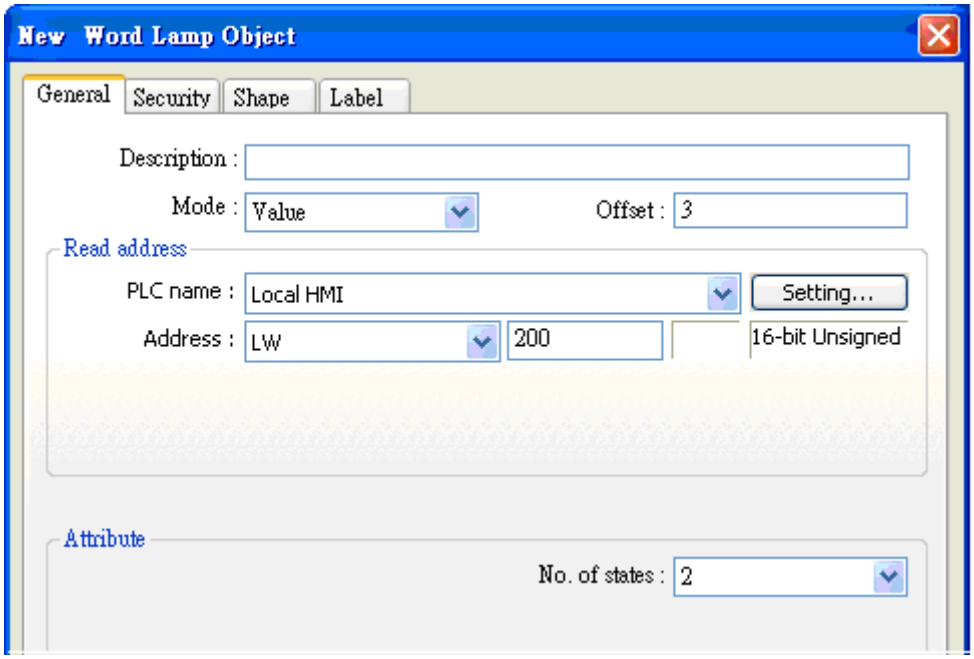
[Mode] / [Offset]

Word lamp object offers the following three modes for selection:

a. Value

The state is displayed according to the value in the designated word address and plus or minus the **[Offset]**.

As shown below, if the value within LW-200 is 3, since the offset is set to 3, the shape of state 6 is displayed. (value 3 + offset 3)



b. LSB

Convert the value from decimal to binary. The least significant active bit in a binary data word selects the state displayed.

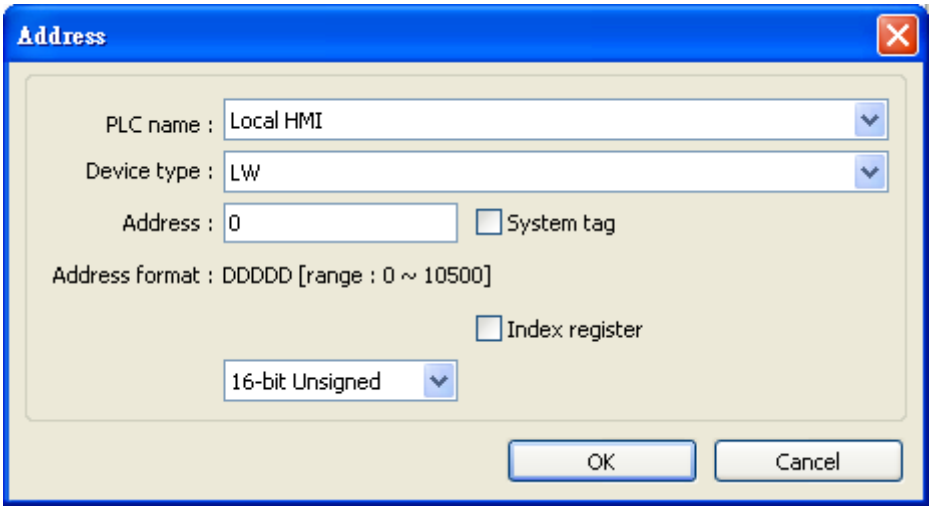
Decimal value	Binary value	Displayed state
0	0000	State 0 displayed. All the bits are 0.
1	0001	State 1 displayed. The least significant active bit is bit 0.
2	0010	State 2 displayed. The least significant active bit is bit 1.
3	0011	State 1 displayed. The least significant active bit is bit 0.
4	0100	State 3 displayed. The least significant active bit is bit 2.
5	0101	State 1 displayed. The least significant active bit is bit 0.
6	0110	State 2 displayed. The least significant active bit is bit 1.
7	0111	State 1 displayed. The least significant active bit is bit 0.
8	1000	State 4 displayed. The least significant active bit is bit 3.

c. Change state by time

The state displayed changes on a time-base. The frequency can be set.

Read address

Click [Setting] to select the [PLC name], [Address], [Device type], [System tag], [Index register] of the word device that controls the [Word Lamp] object. Users can also set address in [General] tab while adding a new object.



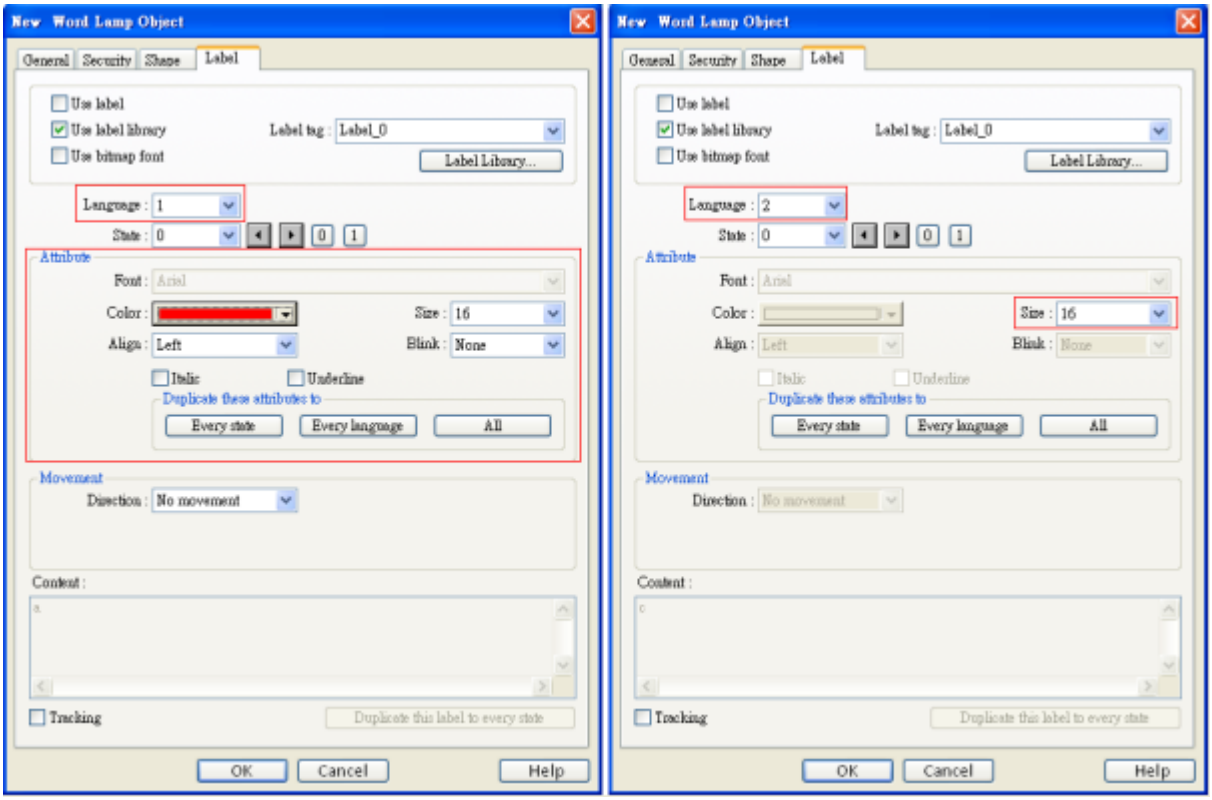
Attribute

[No. of states]

The number of states is utilized by the object. The state is numbered from 0, so the number of states minus 1 will be the state number. If the value within the word register is ≥ **[No. of states]** defined in Attribute, the highest state will be displayed.

If the number of states is set to 8, the valid states will be 0, 1, 2, …, 7. In this case if the word value is 8 or higher, the system will display the state 7 shape.

In [Label] tab, Language 1 determines the relevant settings of the font. For Language 2~8, only the font size can be changed and other settings follow Language 1.

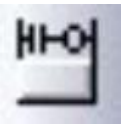


13.3 Set Bit

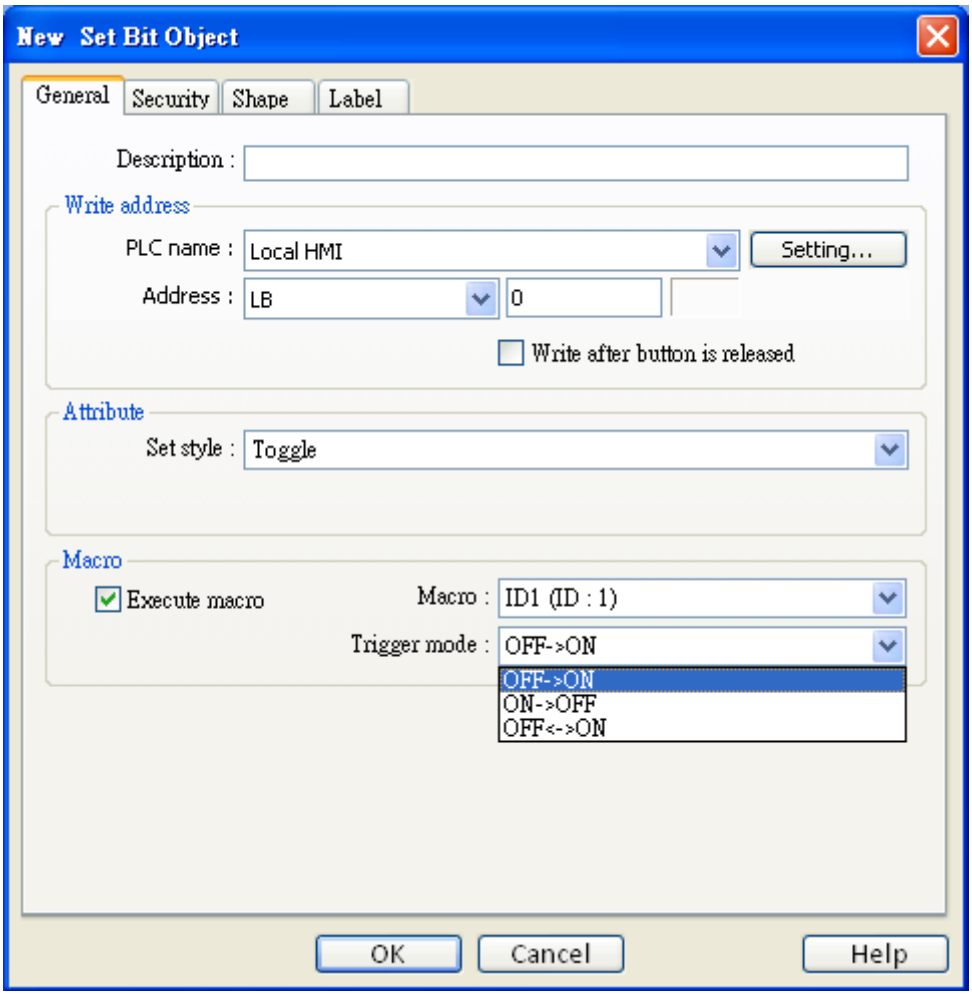
Overview

The **[Set Bit]** object provides two operation modes: manual or automatic. Manual mode can trigger a designated bit address to change the state between ON and OFF when the object is touched. In automatic mode, the bit is automatically activated when a pre-defined condition occurs, touching the button will not be effective.

Configuration

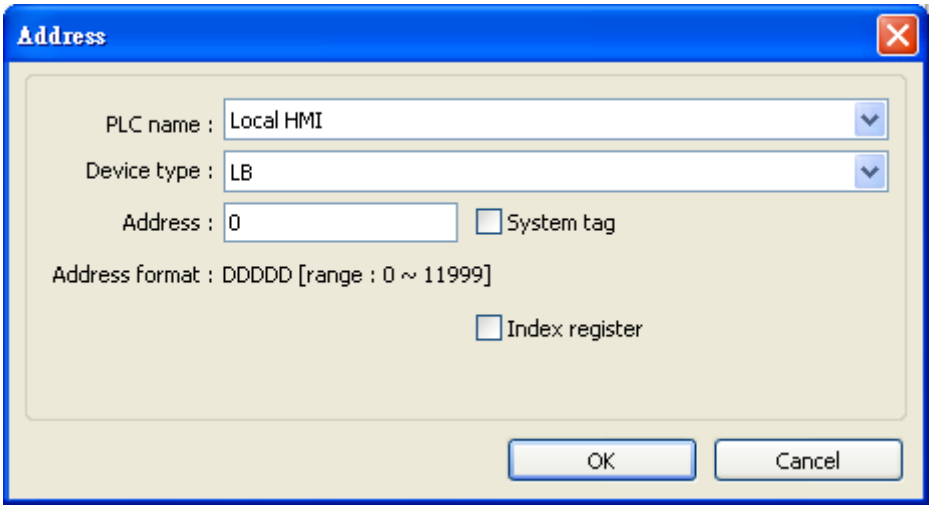


Click the [Set Bit] icon on the toolbar to open a [Set Bit] object property dialog. Set up the properties, press OK button, and a new [Set Bit] object will be created.



Write address

Click [Setting] to select the [PLC name], [Address], [Device type], [System tag], [Index register] of the bit device that controls the [Set Bit] object. Users can also set address in [General] tab while adding a new object.



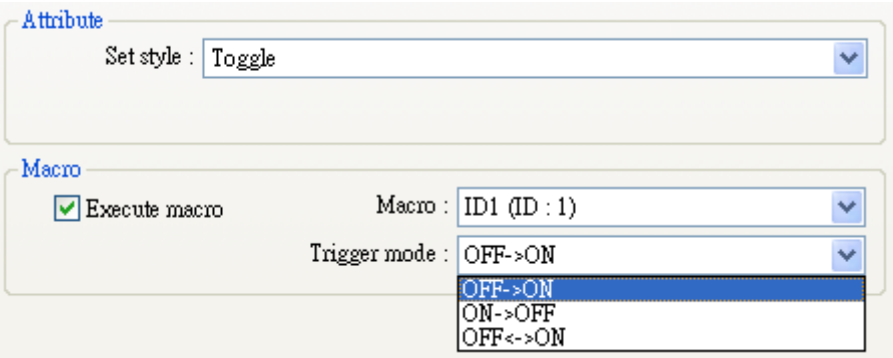
[Write after button is released]
If this function is selected, the action is delayed till button is released; otherwise, the action is executed once the button is pressed. This function does not work with momentary buttons.

Attribute
[Set Style] Please refer to the following description for different types of operation mode.

Set style	Description
Set ON	Set ON the designated bit of the device.
Set OFF	Set OFF the designated bit of the device.
Toggle	Alternates the bit state each time pressed.
Momentary	Holds the bit ON only while button is pressed.
Periodical toggle	Set a designated bit ON and OFF at a set time interval. Time interval can be selected，the range: 0.1~25.5 seconds.
Set ON when window opens	Set ON the bit within the window when the window opens.
Set OFF when window opens	Set OFF the bit within the window when the window opens.
Set ON when window closes	Set ON the bit within the window when the window closes.
Set OFF when window closes	Set OFF the bit within the window when the window closes.
Set ON when backlight on	Set the bit ON when the backlight is turned ON.
Set OFF when backlight on	Set the bit OFF when the backlight is turned ON.
Set ON when backlight off	Set the bit ON when the backlight is turned OFF.
Set OFF when backlight off	Set the bit OFF when the backlight is turned OFF.

Macro
[Set Bit] object can trigger the start of a Macro routine when the Macro has been created in advance. Please refer to “Chapter 18 Macro Reference” for more information.

Trigger Mode



If [Set style] is set to [Toggle], there is a further selection to make of whether the macro operates after:

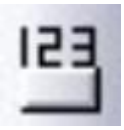
OFF->ON	OFF to ON transition
ON->OFF	ON to OFF transition
ON<->OFF	At both of the changes of state.

13.4 Set Word

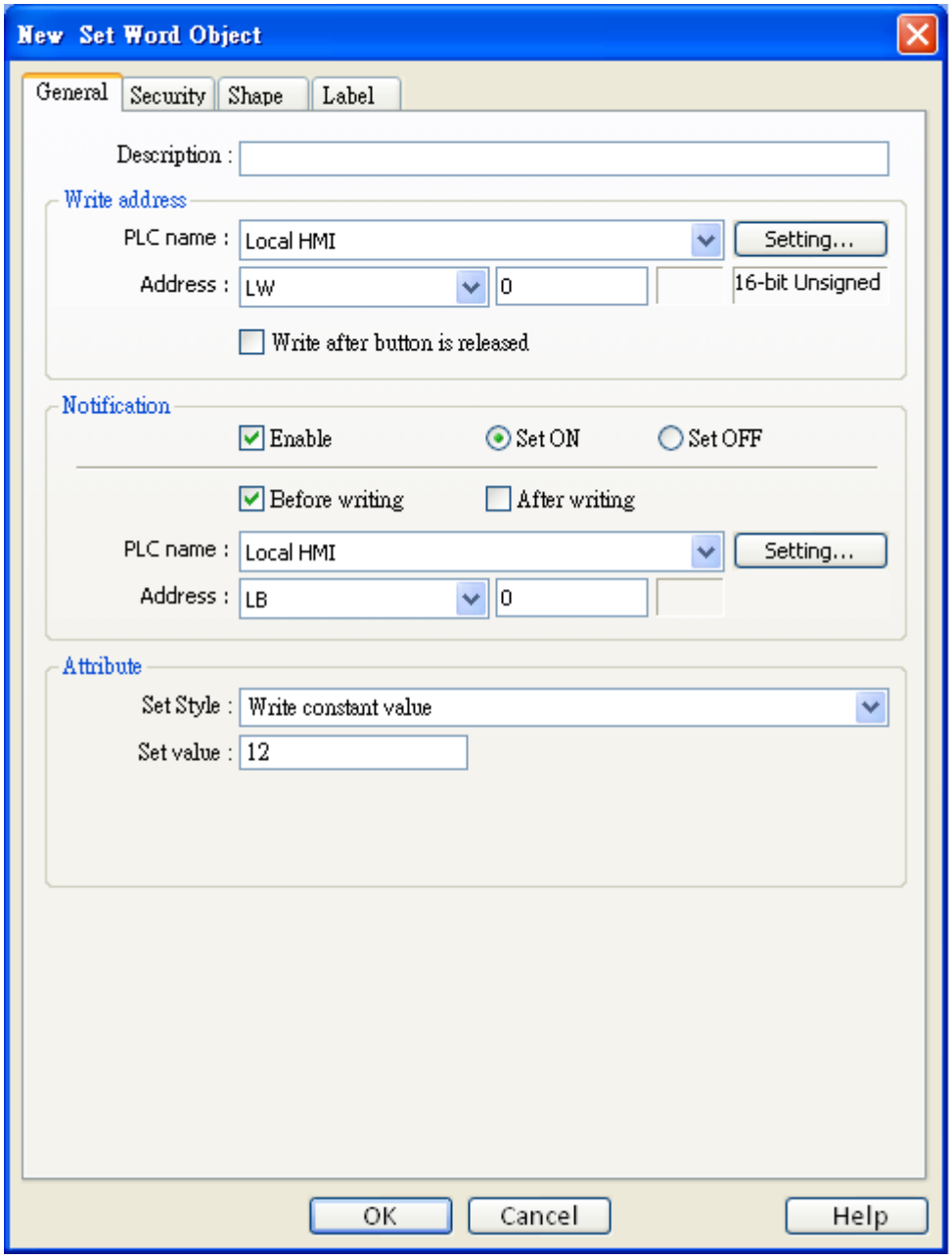
Overview

The [Set Word] object provides two operation modes: manual or automatic. Manual mode can change the value in a designated word address when the object is touched. In automatic mode, the word register is automatically activated when a pre-defined condition occurs, touching the button will not be effective.

Configuration

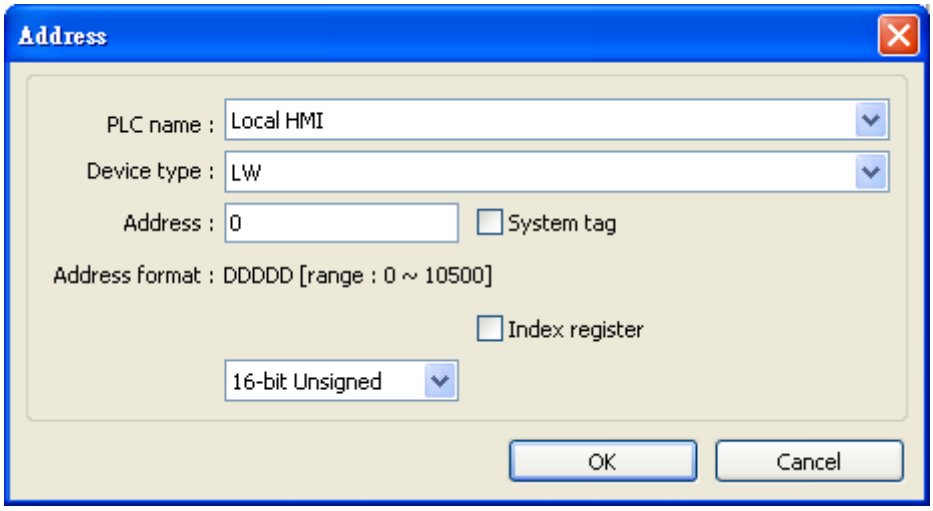


Click the [Set Word] icon on the toolbar to open a [Set Word] object property dialog. Set up the properties, press OK button, and a new [Set Word] object will be created.



Write address

Click [Setting] to select the [PLC name], [Address], [Device type], [System tag], [Index register] of the word device that controls the [Set Word] object. Users can also set address in [General] tab while adding a new object.



[Write after button is released]

If this function is selected, the action is delayed till button is released; otherwise, the action is executed once the button is pressed.

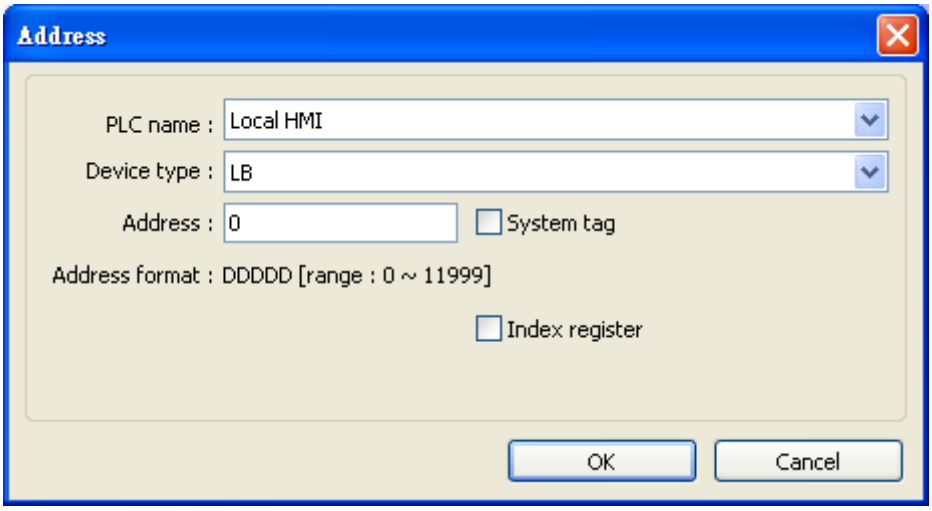
Notification

If this check box is selected, it will notify a designated bit address (setting ON or OFF).

[Before writing] / [After writing]

Set the state of the designated bit address before or after the manual operation.

Click [Setting] to select the [PLC name], [Address], [Device type], [System tag], [Index register] of the Notification bit. Users can also set address in [General] tab while adding a new object.



Attribute

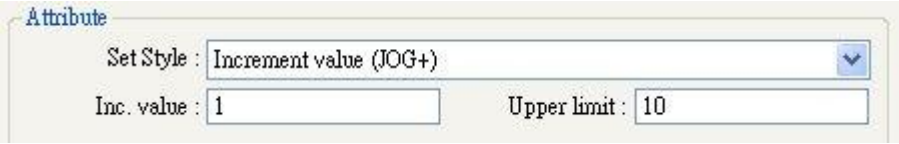
[Set style] Select the button action from the drop down list:

■ Write constant value

Preset a register with the value entered. Each time when the button is pressed, it writes the **[Set value]** to the designated register. Data format is as set by the **[Write address]**; it can be 16-bit BCD, 32-bit BCD, …32-bit float. As shown below, when the button is pressed, preset the register with 12.

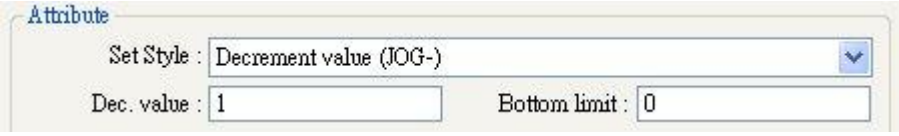
■ Increment value (JOG+)

Increase value in register by a set amount in [Inc. value], each time when the button is pressed, to the [Upper limit]. As shown below, each time when pressing the button, increase 1 in the designated register, till the value is 10.



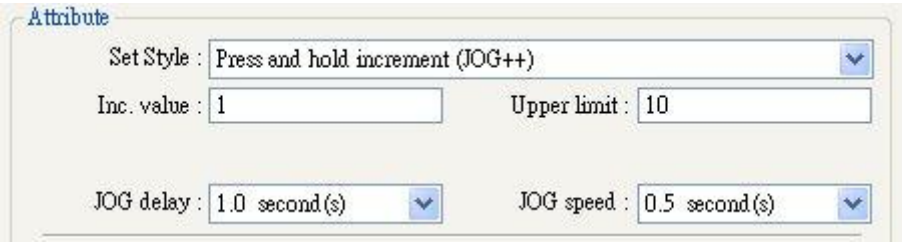
■ Decrement Value (JOG-)

Decrease value in register by a set amount in [Dec. value], each time when the button is pressed, to the [Bottom limit]. As shown below, each time when pressing the button, decrease 1 in the designated register, till the value is 0.



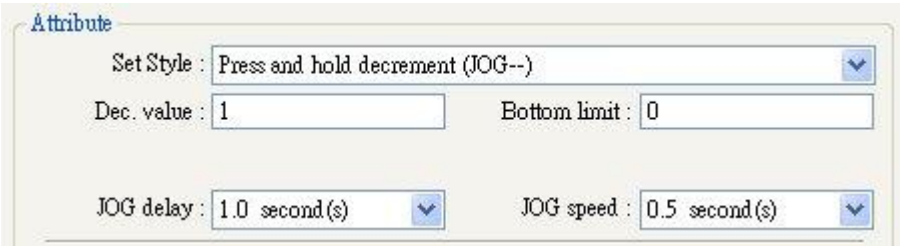
■ Press and hold increment (JOG++)

When the button is held longer than a set time in [JOG delay], it will increase the value in a register by a set amount in [Inc. value] at a set rate in [JOG speed], to the [Upper limit].
As shown below, when the button is pressed, increase the value in the designated register by 1. When the button is held longer than 1 second, increase the value in register by 1 every 0.5 second, till the value is 10.



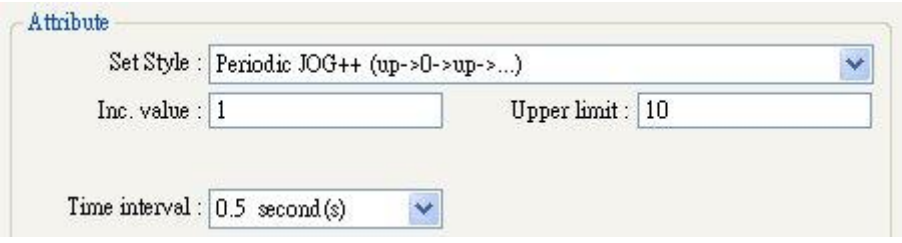
■ Press and hold increment (JOG--)

When the button is held longer than a set time in [JOG delay], it will decrease the value in a register by a set amount in [Dec. value] at a set rate in [JOG speed], to the [Bottom limit].
As shown below, when the button is pressed, decrease the value in the designated register by 1. When the button is held longer than 1 second, decrease the value in register by 1 every 0.5 second, till the value is 0.



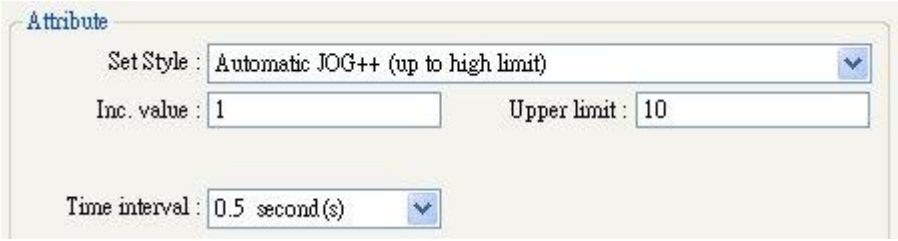
■ Periodical JOG++

An automatic function which operate the designated word address to increase the value in the register by a set amount in [Inc. value], at a set rate in [Time interval], to the [Upper limit].
As shown below, the system will automatically increase the value in the register by 1 every 0.5 second, till the value is 10. Then the value returns to 0 and add 1 every 0.5 second again.



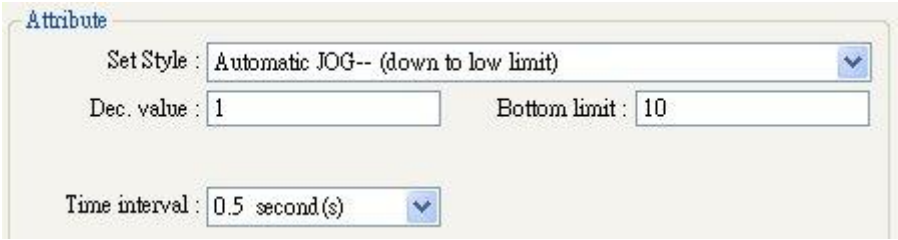
■ Automatic JOG++

An automatic function which operate the designated word address to increase the value in the register by a set amount in [Inc. value], at a set rate in [Time interval], to the [Upper limit], then hold this value. As shown below, the system will automatically increase the value in the register by 1 every 0.5 second, till the value is 10, and then stops.



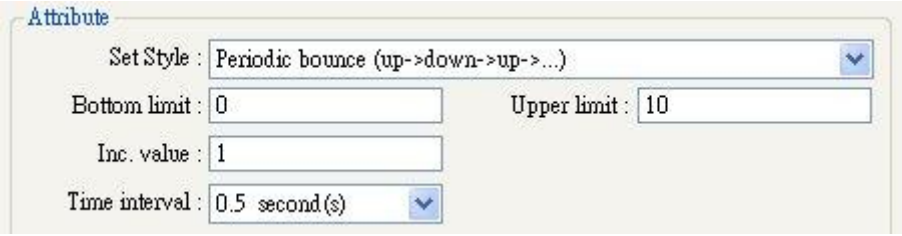
■ Automatic JOG--

An automatic function which operate the designated word address to decrease the value in the register by a set amount in [Dec. value], at a set rate in [Time interval], to the [Bottom limit], then hold this value. As shown below, the system will automatically decrease the value in the register by 1 every 0.5 second, till the value is 10, and then stops.



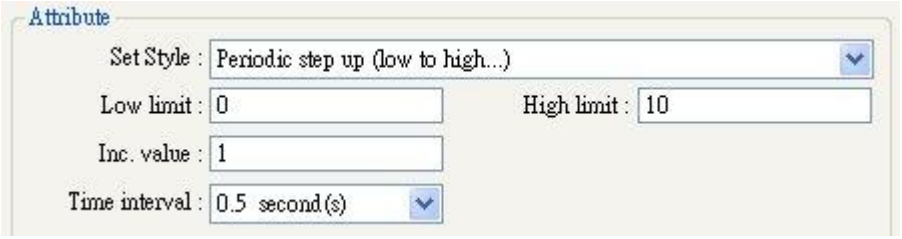
■ Periodical bounce

Increases the word address value to the [High limit] by a [Inc. value] at a set rate in [Time interval], then decreases to the [Low limit] by the same value at the same rate. As shown below, the system will increase the value in the designated register by 1 every 0.5 second, till the value is 10, and then decrease the value by 1 every 0.5 second till the value is 0 whenever the screen is active.



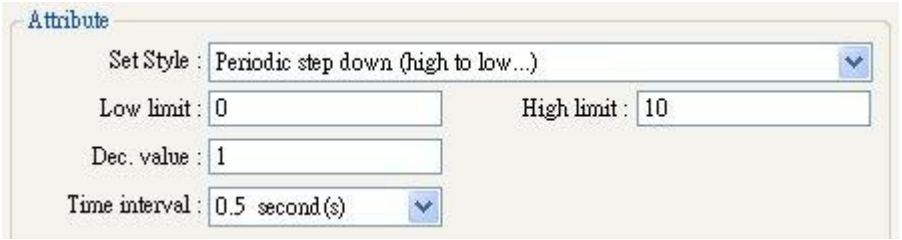
■ Periodical step up

Step up to the [High limit] by [Inc. value] at a set rate in [Time interval], then reset immediately to the [Low limit]. The action repeats whenever the screen is active. As shown below, the system will increase the value in the designated register by 1 every 0.5 second, till the value is 10, and then reset to 0 and increase again, and the action repeats.



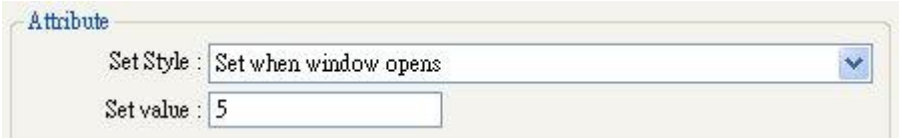
■ Periodical step down

Step down to the [Low limit] by [Dec. value] at a set rate in [Time interval], then reset immediately to the [High limit]. The action repeats whenever the screen is active. As shown below, the system will decrease the value in the designated register by 1 every 0.5 second, till the value is 0, and then reset to 10 and decrease again, and the action repeats.



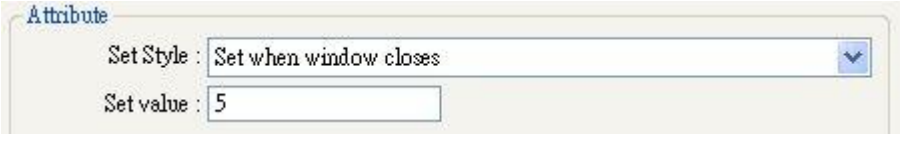
■ Set when window opens

Automatic function occurs whenever the screen is active. The value entered in [Set value] is set into the word address when the action occurs. As shown below, when the window opens, the system enters 5 into the designated register.



■ Set when window closes

Automatic function occurs whenever the screen is inactive. The value entered in [Set value] is set into the word address when the action occurs. As shown below, when the window closes, the system enters 5 into the designated register.



■ Set when backlight on

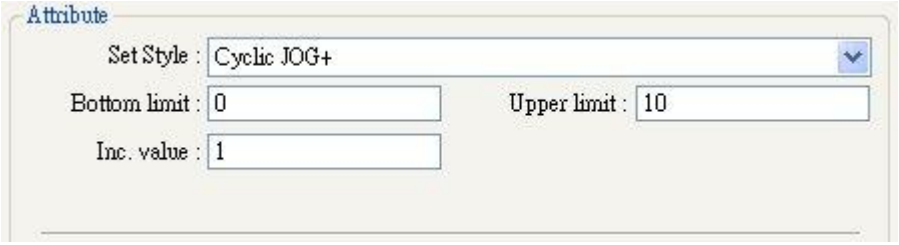
Automatic function occurs whenever the backlight is active. The value entered in [Set value] is set into the word address when the action occurs. As shown below, when the backlight turns ON, the system set 5 into the designated register.

■ Set when backlight off

Automatic function occurs whenever the backlight is inactive. The value entered in [Set value] is set into the word address when the action occurs. As shown below, when the backlight turns OFF, the system set 5 into the designated register.

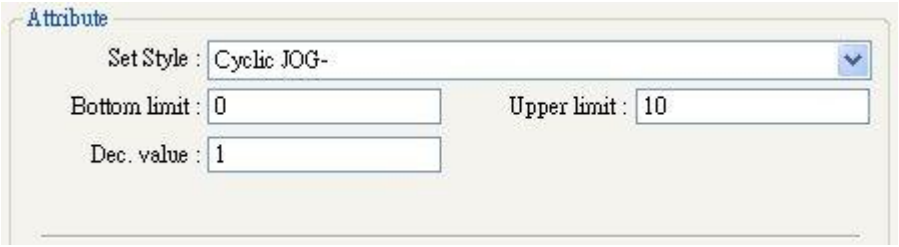
■ Cyclic JOG+

Each time when the button is pressed, increase the word address value to the [Upper limit] by [Inc. value] then reset to the [Bottom limit]. As shown below, each time when pressing the button, the system will increase the value in the designated register by 1, till the value is 10, and then reset to 0 and increase again by pressing the button.



■ Cyclic JOG-

Each time when the button is pressed, decrease the word address value to the [Bottom limit] by [Dec. value] then reset to the [Upper limit]. As shown below, each time when pressing the button, the system will decrease the value in the designated register by 1, till the value is 0, and then reset to 10 and decrease again by pressing the button.

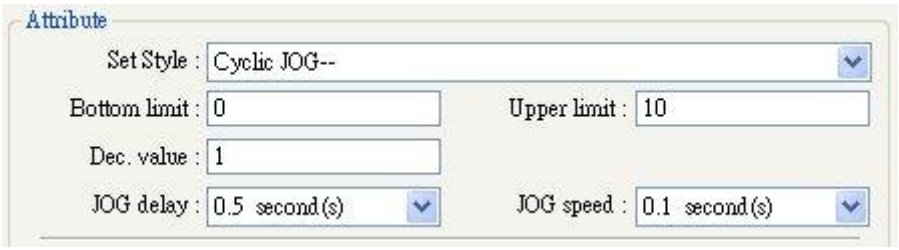


■ Cyclic JOG++

When the button is held longer than a set time in [JOG delay], increase the value in a register by a set amount in [Inc. value] at a set rate in [JOG speed], to the [Upper limit], then reset to the [Bottom limit]. As shown below, when the button is held longer than 0.5 second, increase the value in the designated register by 1 every 0.1 second, till the value is 10, and then reset to 0 and increase again by holding the button.

■ Cyclic JOG- -

When the button is held longer than a set time in [JOG delay], decrease the value in a register by a set amount in [Dec. value] at a set rate in [JOG speed], to the [Bottom limit], then reset to the [Upper limit]. As shown below, when the button is held longer than 0.5 second, decrease the value in the designated register by 1 every 0.1 second, till the value is 0, and then reset to 10 and decrease again by holding the button.



[Dynamic limits]

Set the [Bottom limit] and [Upper limit] by a designated register. When Dynamic Address is LW-n, where n is an arbitrary number, the rule of setting Upper / Bottom limit is:

Content	16-bit	32-bit
Dynamic Address	LW-n	LW-n
Bottom limit	LW-n	LW-n
Upper limit	LW-n+1	LW-n+2

When Dynamic Address is LW-100, the rule of setting Upper / Bottom limit is:

Content	16-bit	32-bit
Dynamic Address	LW-100	LW-100
Bottom limit	LW-100	LW-100
Upper limit	LW-101	LW-102

13.5 Function Key

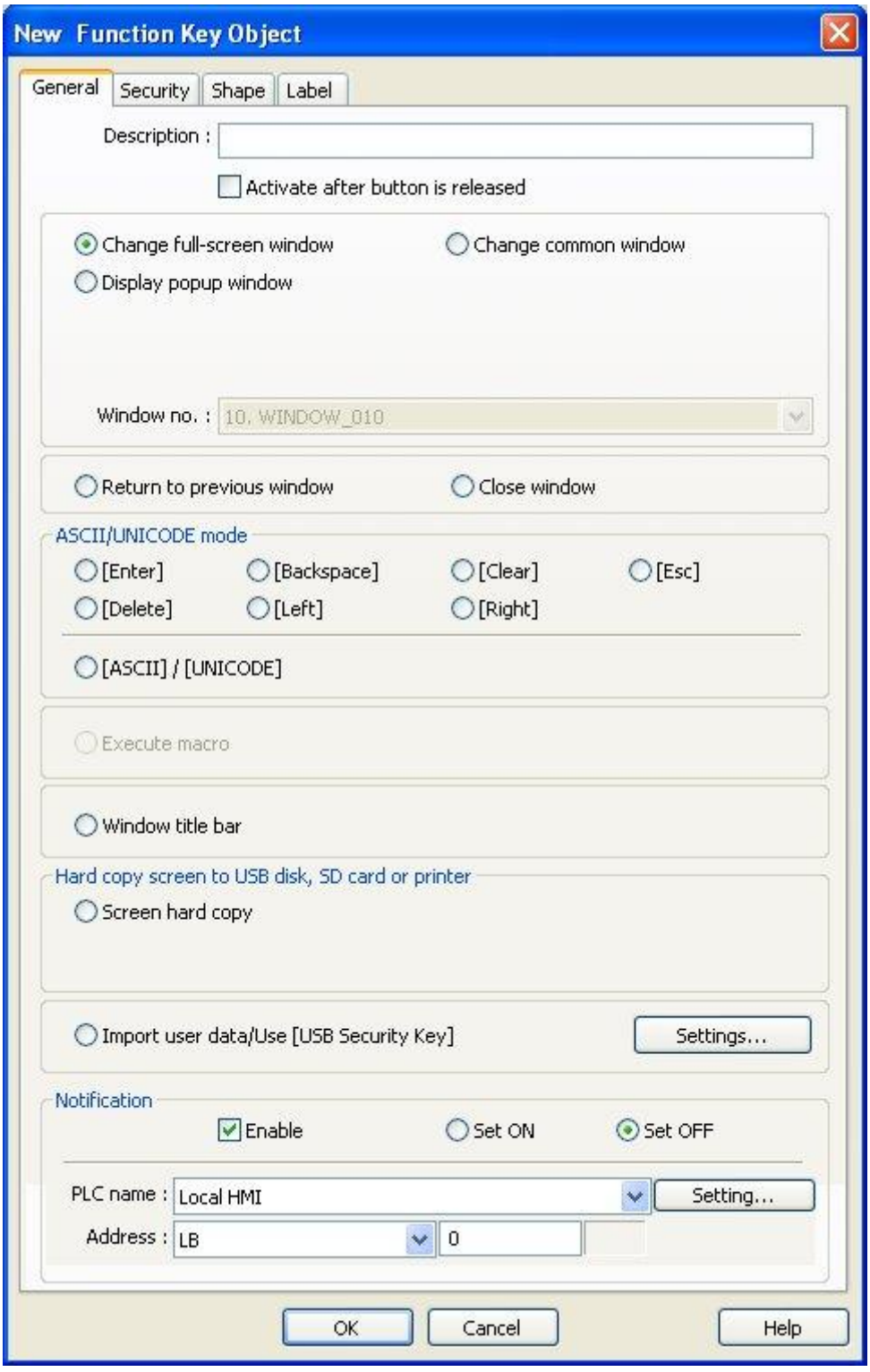
Overview

The [Function Key] object can be used for several tasks:

1. Open / close / return to a screen window.
2. For keypad design
3. Execute a Macro
4. Print screen

Configuration

Click the [Function Key] icon on the toolbar to open a [Function Key] object property dialog. Set up the properties, press OK button, and a new [Function Key] object will be created.



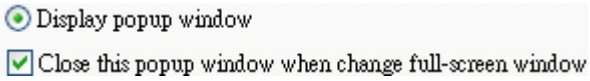
[Activate after button is released]

If this function is selected, the action is delayed till button is released, otherwise, the action is executed once the button is pressed.

[Change full-screen window] Change to another base window.

[Change common window] Change common window.

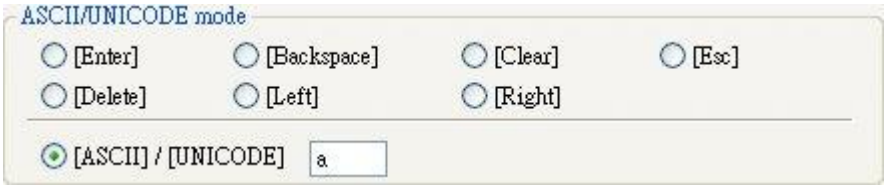
[Display popup window]
A pop up window displays in the base window. If **[Close this popup window when parent window is closed]** check box is selected, the pop up window will be closed when change the base window to another window. Otherwise, a function key in the pop up window is needed to close it.



[Return to previous window]
If this is selected, the Function Key will change from the current screen to the previous one displayed. For example, when window no. 10 is changed to window no. 20, press the function key to return to window no. 10. This function is only available for base window.

[Close window] Close any active popup windows, message windows included.

[ASCII/UNICODE mode]
Configures the button as a keypad key, and the character it enters, via [Numeric Input] or [ASCII Input] objects.



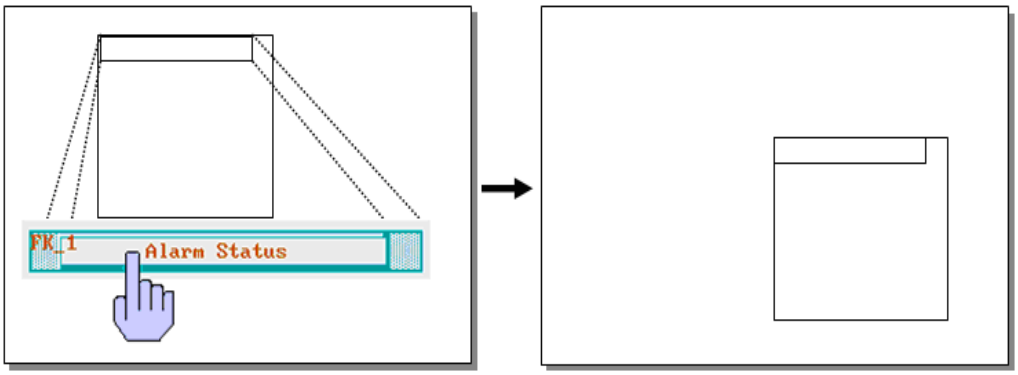
[Enter] Same as the keyboard's "enter" function.
[Backspace] Same as the keyboard's "backspace" function.
[Clear] Clear the value in the word register.
[Esc] Same as the **[Close window]** function, it is used to close the keyboard window.
[Delete] Same as the keyboard's "Delete" function, deletes the number or character on the right side of the text cursor.
[Left] Same as the keyboard's "←" key, moves the text cursor to the left side of the previous number or character.
[Right] Same as the keyboard's "→" key, moves the text cursor to the left side of the next number or character.
[ASCII/UNICODE] Specify the character to be entered by this key.

[Execute Macro]

Select this check box to execute one of the Macros from the drop down list that has already been configured by users.

[Window title bar]

[Function Key] defined can be used to move a popup window which is no [window title bar] to a preferred position on screen. Select the popup window and then click on a preferred position, the window will be moved.



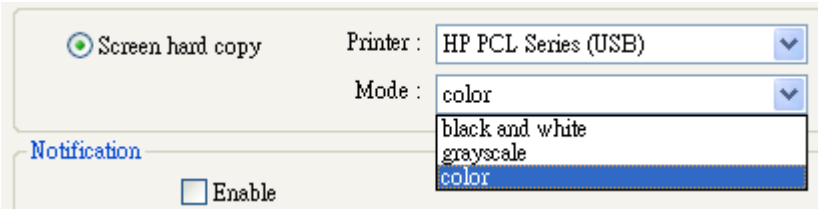
Select the window title bar firstly.

Touching the screen for the new position the popup window will be moved.

[Screen hard copy]

Print the current window. Before using this function, choose a printer model in **[System Parameter Settings] » [Model] » [Printer]**.

If a single color printer is used, selecting [grayscale] can get a better print result, but the text may not be clearly printed. To improve text printing, it is not necessary to select [grayscale].



Notification

If this selection is enabled, it will notify a designated bit address to set ON or OFF, each time the button is pressed.

Refer to

1. Chapter 6 Window Operations
2. Chapter 12 Keypad Design and Usage

Example 1 Design Non-ASCII character keyboard

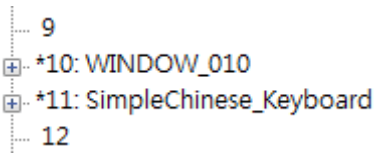
The following explains how to enter and display non-ascii characters in HMI, such as Traditional Chinese, Simplified Chinese, Japanese, Greek and so on.
Please follow the steps.

Step1: Setting non-ascii fonts

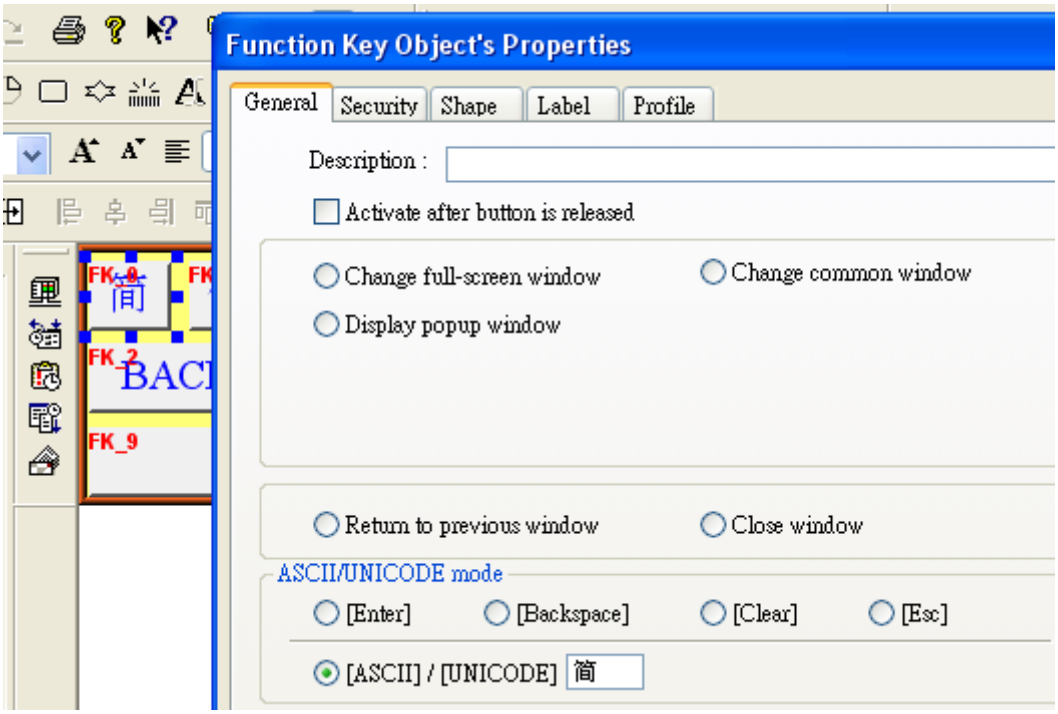
Add the needed non-ascii fonts in [System Parameter Settings] » [Font].
Please use the required font for the language used.

Step2: Design non-ascii keyboard

Create “window no.11” and design the non-ascii keyboard. The following use Simplified Chinese as an example.



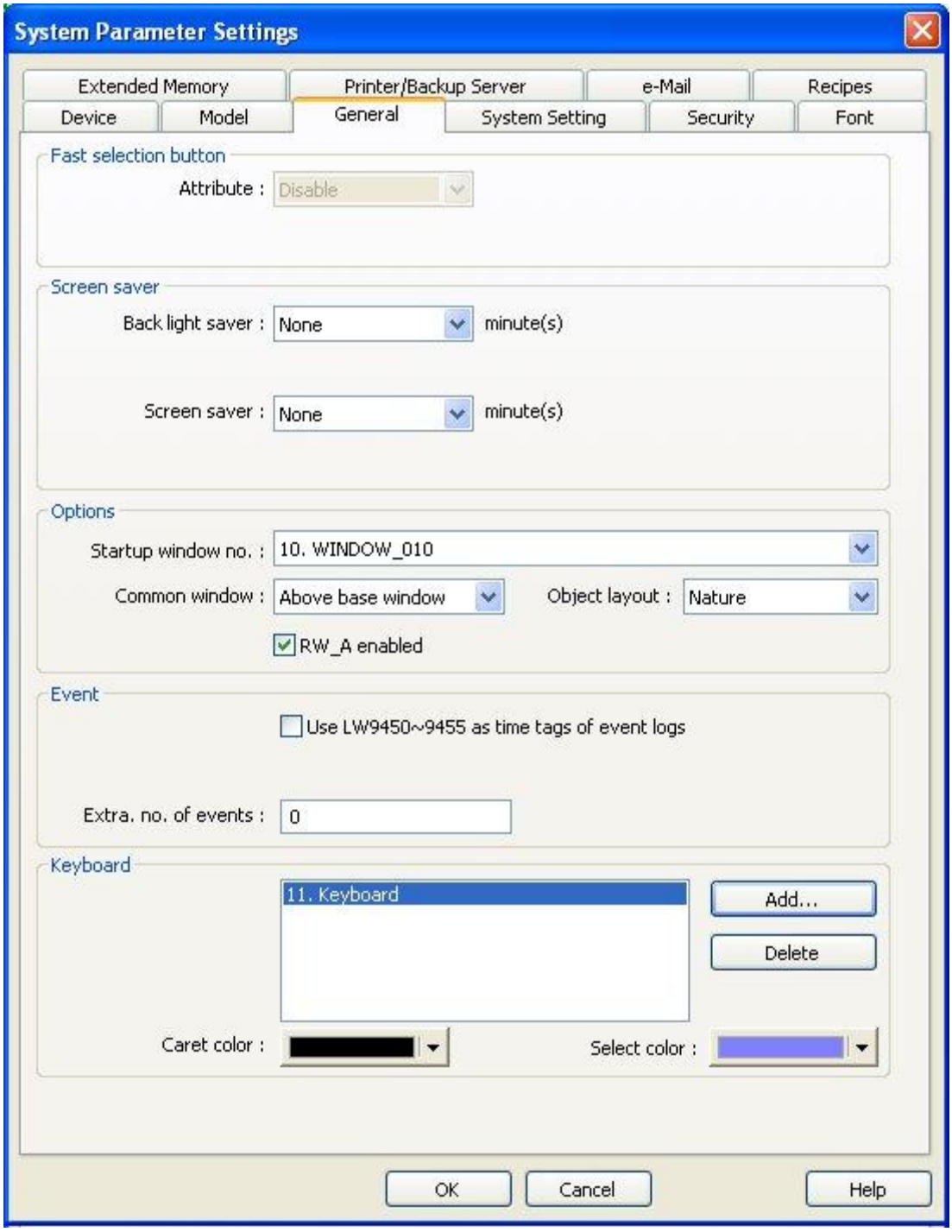
The objects in this window are all Function Keys that are set to the needed functions as labeled. Take the “简” Function Key as an example, create a function key in [ASCII] / [UNICODE] mode, as shown below.



In the [Label] tab, select [Use label], type “简” in [Content] and select “AR MingtiM GB”, which must be a font set in step 1, as shown below.

The Function Keys used for typing non-ascii characters in the same keyboard must all use the same font. For example, in a Simplified Chinese keyboard, use “AR MingtiM GB” in each key for entering characters.

When finished, add window no. 11 to [System Parameter Settings] » [General] » [Keyboard] as shown below.



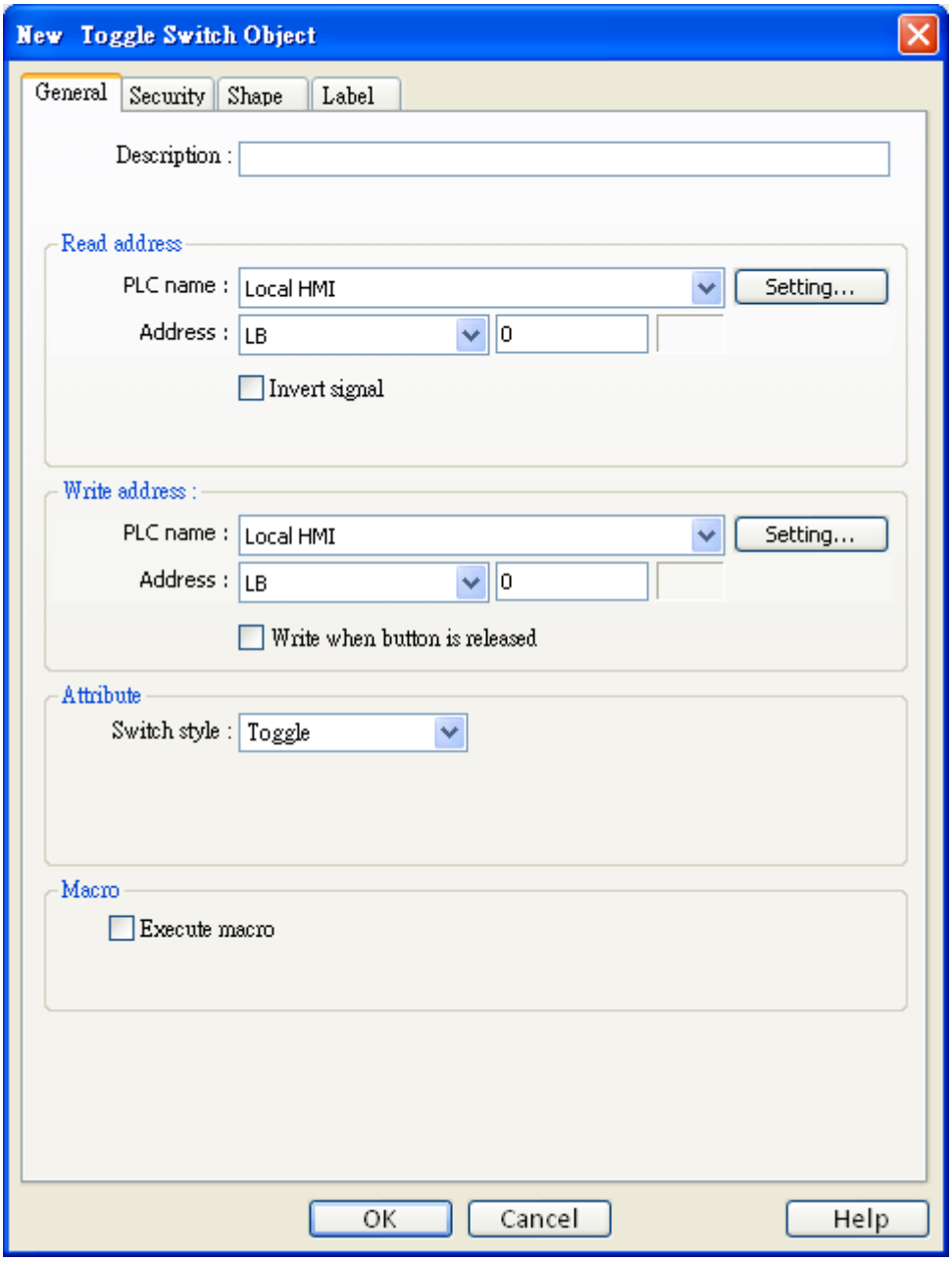
13.6 Toggle Switch

Overview

[Toggle Switch] object is a combination of [Bit Lamp] object and [Set Bit] object. The appearance of the object is controlled by the ON / OFF state of the read bit address when pressing the button.

Configuration

Click the [Toggle Switch] icon on the toolbar to open a [Toggle Switch] object property dialog. Set up the properties, press OK button, and a new [Toggle Switch] object will be created.



Read address

Click [Setting] to select the [PLC name], [Address], [Device type], [System tag], [Index register] of the bit device that controls the [Toggle Switch] object. Users can also set address in [General] tab while adding a new object.

[Invert signal]

Reverses the display of ON / OFF states. For example, if [Invert signal] check box is selected, when the designated bit is OFF, the object displays ON state.

Write address

Click [Setting] to select the [PLC name], [Address], [Device type], [System tag], [Index register] of the bit device that controls the [Toggle Switch] object. Users can also set address in [General] tab while adding a new object. The address can be the same or different from [Read address].

[Write after button is released]

If this function is selected, the action is delayed till button is released, otherwise, the action is executed once the button is pressed.

Attribute

Switch style	Description
Set ON	Press the button to set ON the designated register.
Set OFF	Press the button to set OFF the designated register.
Toggle	Press the button to set the designated register to an inverse state. Set OFF when the state is ON, set ON when the state is OFF.
Momentary	Hold the button to set ON in the designated register and OFF when released.

Macro

[Toggle Switch] object can trigger the start of a Macro routine when the Macro has been created in advance. Please refer to “Chapter 18 Macro Reference” for more information

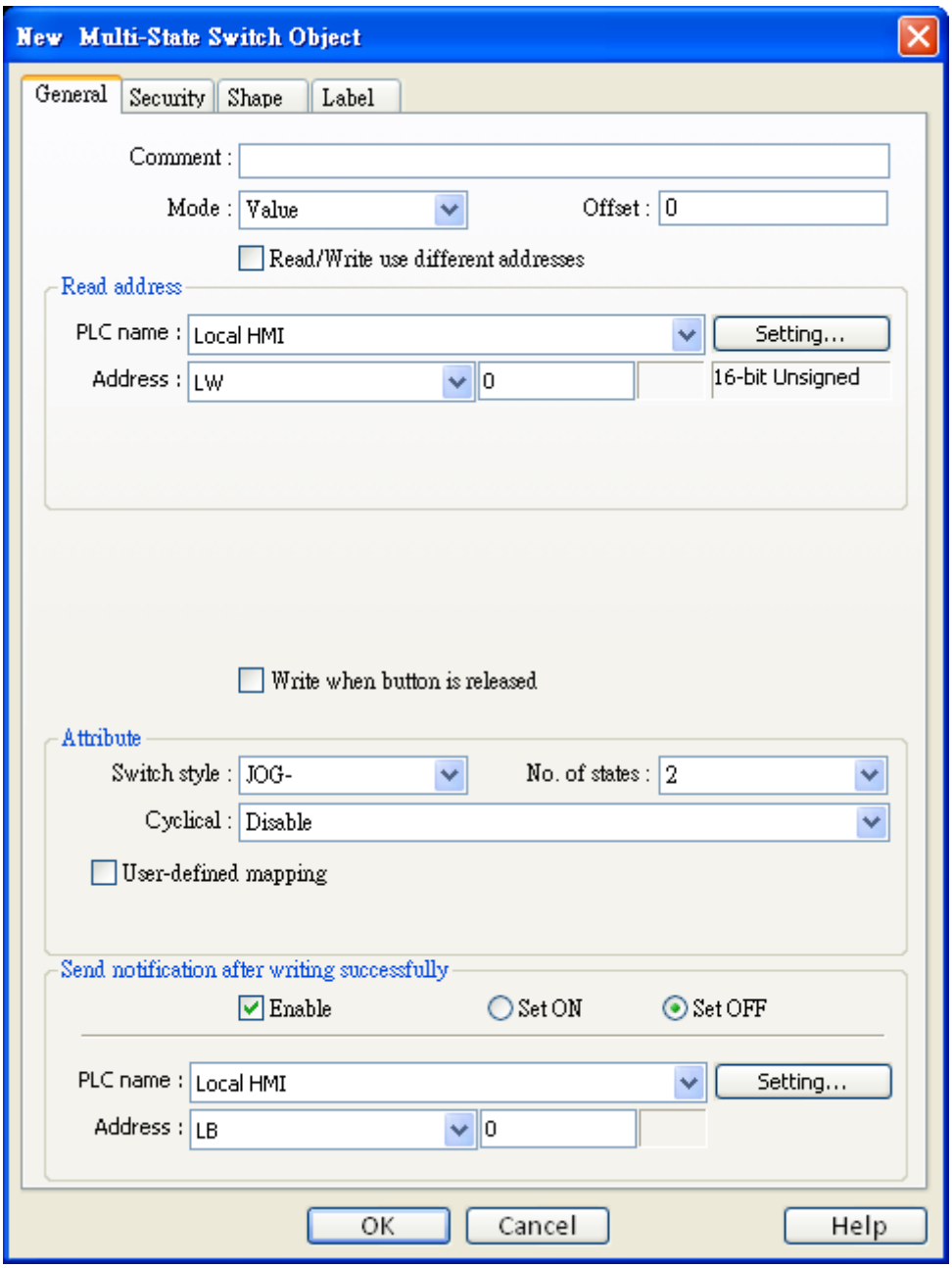
13.7 Multi-State Switch

Overview

[Multi-state Switch] object is a combination of [Word Lamp] object and [Set Word] object. The appearance of the object is controlled by the value of the read word address when pressing the button.

Configuration

Click the [Multi-State Switch] icon on the toolbar to open a [Multi-State Switch] object property dialog. Set up the properties, press OK button, and a new [Multi-State Switch] object will be created.



[Mode] / [Offset]

Offers [Value] and [LSB] display mode. Please refer to the “Word Lamp Object” section of this chapter for more information.

Read address

Click [Setting] to select the [PLC name], [Address], [Device type], [System tag], [Index register] of the word device that controls the [Multi-state Switch] object. Users can also set address in [General] tab while adding a new object.

Write address

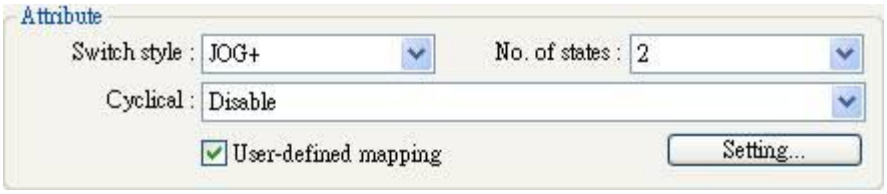
Click [Setting] to select the [PLC name], [Address], [Device type], [System tag], [Index register] of the word device that controls the [Multi-state Switch] object. Users can also set address in [General] tab while adding a new object. The address can be same or different from [Read address].

[Write after button is released]

If this function is selected, the action is delayed till button is released, otherwise, the action is executed once the button is pressed.

Attribute

Select the object's operation mode.

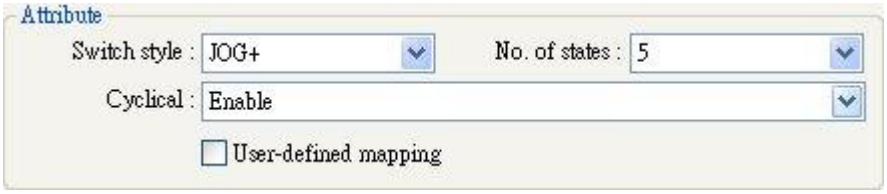


[Switch style]

There are [JOG+] and [JOG-] selections. When the [Offset] is not 0, state displayed is "[no. of state] -1 + [Offset]".

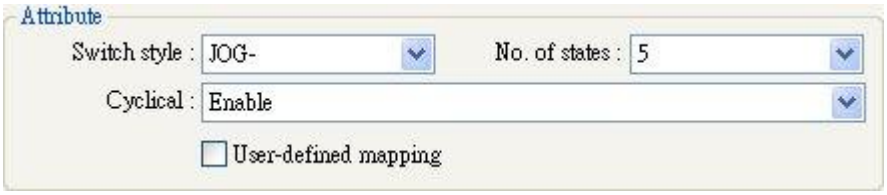
a. JOG+

Increase the value of a designated register by 1 each time when pressing the button, till the value equals to [No. of states]. A cyclic action can be enabled. As shown below, each time when pressing the button, the state number will add 1 start from state 0, till state 4 ([no. of state]-1), and returns to 0 and step up again.



b. JOG-

Decrease the value of the designated register by 1 each time when pressing the button, till the value equals to 0. A cyclic action can be enabled. As shown below, each time when pressing the button, the state number will minus 1 start from state 4 ([no. of state]-1), till state 0, and returns to state 4 and step down again.



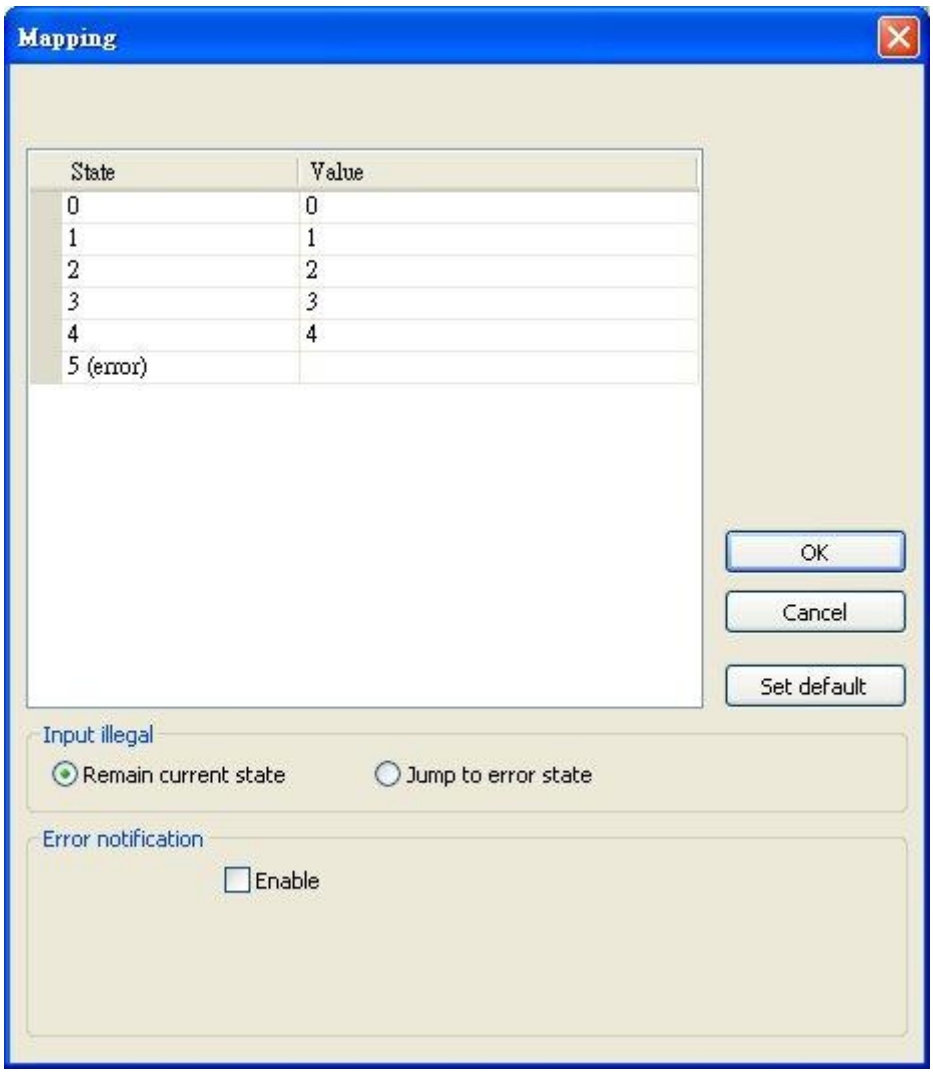
[User-defined mapping]

The value placed in the write register of each selection can be set, also the action taken when an illegal value is entered or notify a designated bit address.

[Remain current state]: If an illegal value is entered, [Multi-state Switch] will remain at the current state.

[Jump to error state]: If an illegal value is entered, [Multi-state Switch] will jump to the error state.

[Error notification] If an illegal value is entered, automatically set the value placed in the designated register.



Send notification after writing successfully

If the [Enable] check box is selected, it will notify a designated bit address (setting ON or OFF) after the command is successfully executed. Click [Setting] to select the [PLC name], [Address], [Device type], [System tag], [Index register] of the bit device that controls the object, or configure in [General] tab.

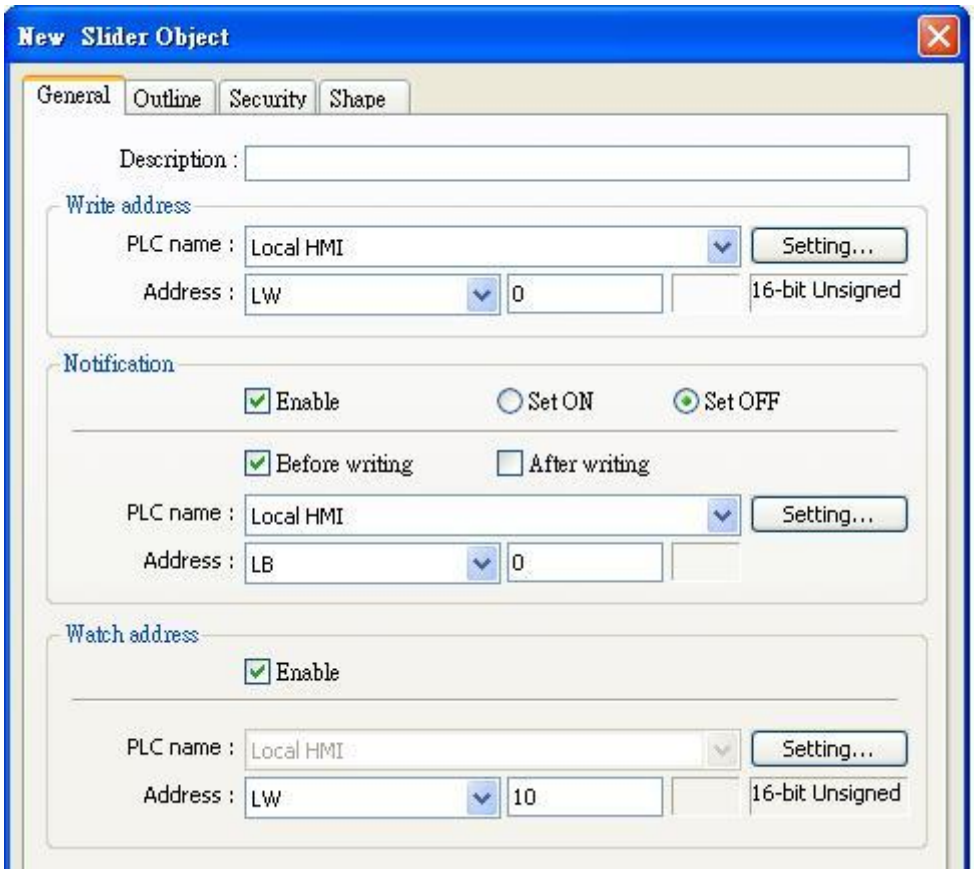
13.8 Slider

Overview

[Slider] object is used to change the value in a designated word register address by moving the roller on the screen.

Configuration

Click the [Slider] icon on the toolbar to open a [Slider] object property dialog. Set up the properties, press OK button, and a new [Slider] object will be created.



Write address

Click [Setting] to select the [PLC name], [Address], [Device type], [System tag], [Index register] of the word device that controls the [Slider] object. Users can also set address in [General] tab while adding a new object.

Notification

If enabled, the state of a designated bit address will be set to ON or OFF, either before, or after the Slider is slid.

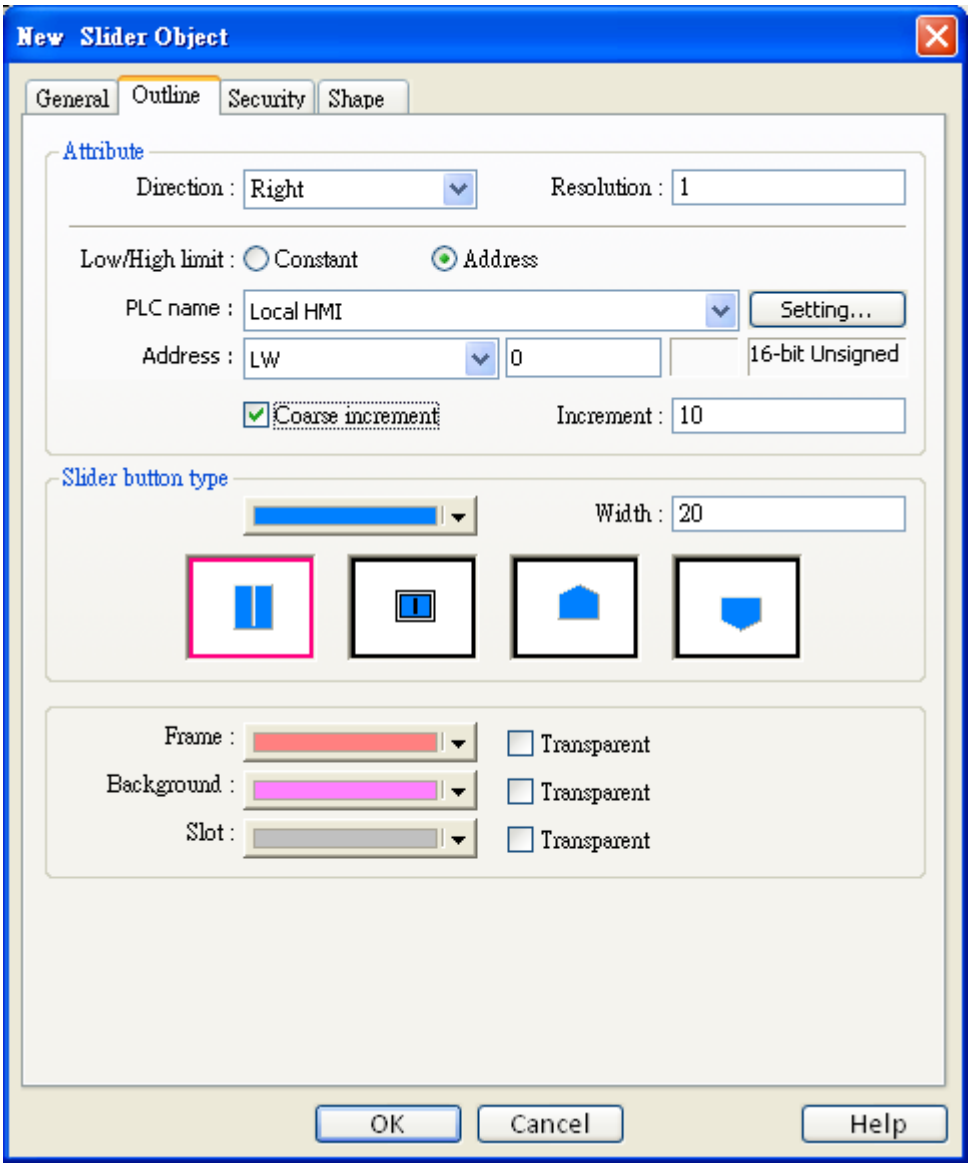
Click [Setting] to select the [PLC name], [Address], [Device type], [System tag], [Index register] of the bit device that controls the notification settings. Users can also set address in [General] tab while adding a new object.

[Before writing] / [After writing]

Change the state of a designated bit register before, or after the Slider is slid.

Watch address

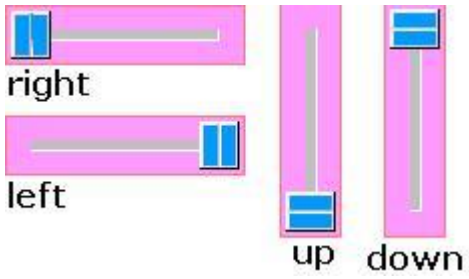
When moving the roller, the new value written to the word register address can be displayed in real time.



Attribute

[Direction]

Select the direction of the slider. (Right, Up, Left, Down)



[Resolution]

Sets the value change in the word register for each step of the Slider. For example, if set to 10, the register value changes by 10 points for each increment or decrement on the Slider.

[Low limit & High limit]

a. Constant

Sets the range of the Slider. EX: If set [Low limit] to 5, and [High limit] to 100, the Slider will enter values between 5 and 100.

b. Address

Set the [Low/High limit] by a designated register. When [Address] is LW-n, where n is an arbitrary number, the rule of setting [Low/High limit] is:

Content	16-bit	32-bit
Address	LW-n	LW-n
Low limit	LW-n	LW-n
High limit	LW-n+1	LW-n+2

For example, when [Address] is LW-100, the rule of setting [Low/High limit] is:

Content	16-bit	32-bit
Dynamic Address	LW-100	LW-100
Low limit	LW-100	LW-100
High limit	LW-101	LW-102

[Coarse increment:]

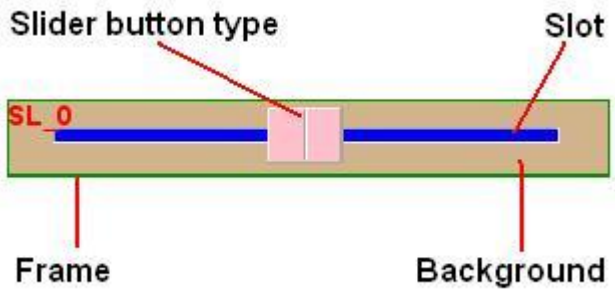
Apart from moving the roller to change the value as in [Resolution], if this option is selected, the word value will increase / decrease by the [Increment] value each time the object is touched.

Slider button type

A choice of different slider shapes, and the width of the Slider may be set.

Color

Colors of the frame, background, and slot may be chosen.



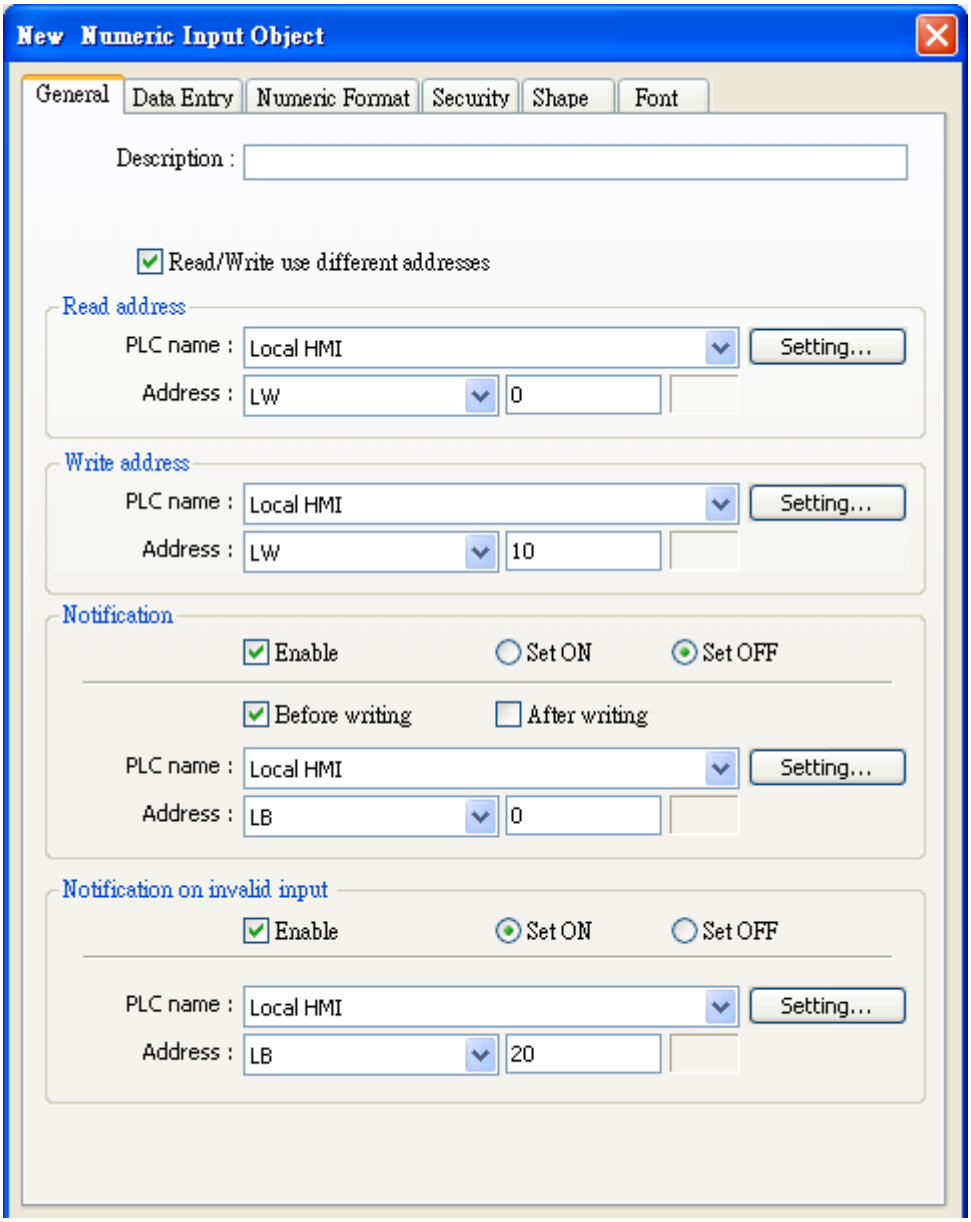
13.9 Numeric Input and Numeric Display

Overview

[Numeric Input] object and [Numeric Display] object can be used to display the value of a designated word register. [Numeric Input] object can be used to input a value into a register via a keyboard.

Configuration


Click the [Numeric Input] or [Numeric Display] icon on the toolbar to open a [Numeric Input] or [Numeric Display] object property dialog. Set up the properties, press OK button, and a new [Numeric Input] or [Numeric Display] object will be created.



[Read/Write use different address]

Set [Read address] and [Write address] differently.

Read address

Click [Setting] to select the [PLC name], [Address], [Device type], [System tag], [Index register] of the word device that displays the value. Users can also set address in [General] tab while adding a new object.

Write address

Select the [PLC name], [Device type], [Address] of the word device that system writes to.

Notification

If enabled, the state of a designated address will be set to ON or OFF, either before or after the object writes the new value to the word register address.

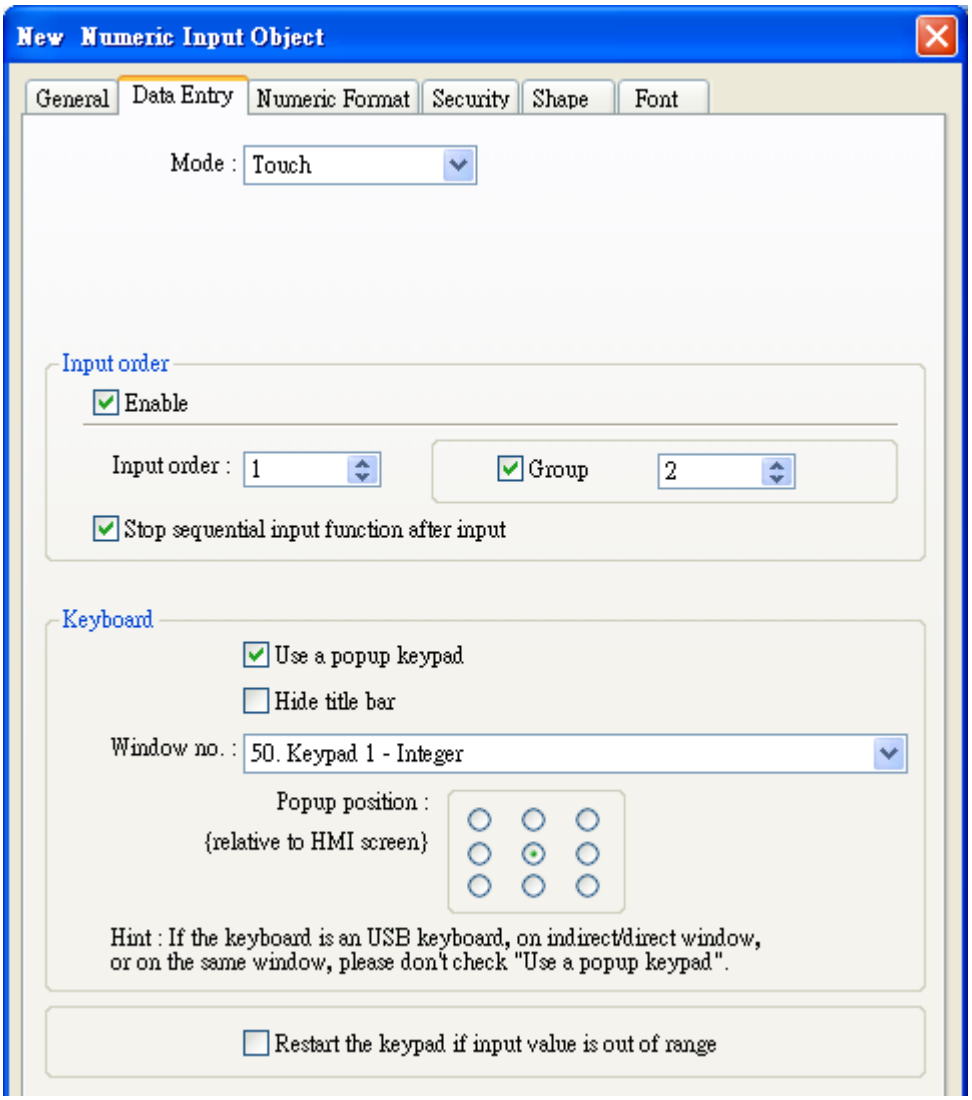
Click [Setting] to select the [PLC name], [Address], [Device type], [System tag], [Index register] of the bit device that controls the notification settings. Users can also set address in [General] tab while adding a new object.

[Before writing] / [After writing]

Set the value of the designated register before, or after the object writes the new value to the write register address.

Notification on valid input

If an illegal value is entered, automatically set the state of a designated register.



Mode

- **[Touch]**

Used when data entry is initiated by touching the screen object.

- **[Bit control]**

Used when data entry is enabled by turning ON a designated bit, and entry ends when the bit goes OFF.

[Allow input bit address]

Specify a bit address that enables or ends data entry. The order of data entry is specified in [Input order] and an external USB keyboard is needed for data entry.

Input order

Perform continuous input by setting [Input order] and [Group].

- **Criterion of searching the next input object**
  a. The range of [Input order]: 1 ~ 511. The range of [Group]: 1 ~ 15.
  b. If [Group] is not selected, its input order is 0.
  c. The system only searches for the objects with the same Group.
  d. The lower number of order is entered before the higher number of order.
  e. For the two objects in the same group and input order, the object placed in the lower layer is entered first.

Keyboard

- **Select [Use a popup keypad]**
  A pre-designed popup keypad can be chosen by selecting a check box, and selecting the relative position on the HMI screen. When data entry is enabled, the popup keypad displays in the selected position, and closed when data entry ends.
- **Not selecting [Use a popup keypad]**
  When data entry is enabled, the popup keypad is not displayed. Users may:
  a. Create a custom design on the same screen window.
  b. Use a USB keyboard.
- **[Hide title bar]**
  Use a keypad without the title bar.
- **[Restart the keypad if input value is out of range]**
  When entering data, if the value entered is not within the valid range, the system will automatically restart the keypad.

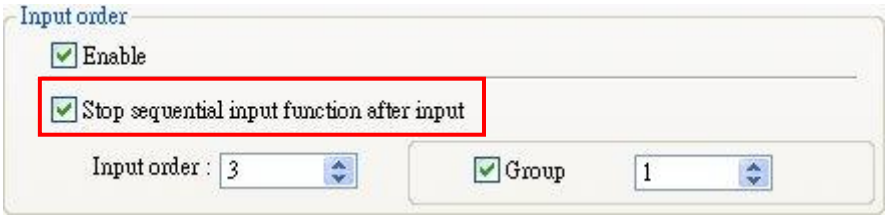
Example 1 Design a group of Numeric Input Object

This example demonstrates how to use [Input Order] and [Group] to perform continuous input in several [Numeric Input] objects. After entering data in one object, entry will be passed to the next input order object which is in the same group.

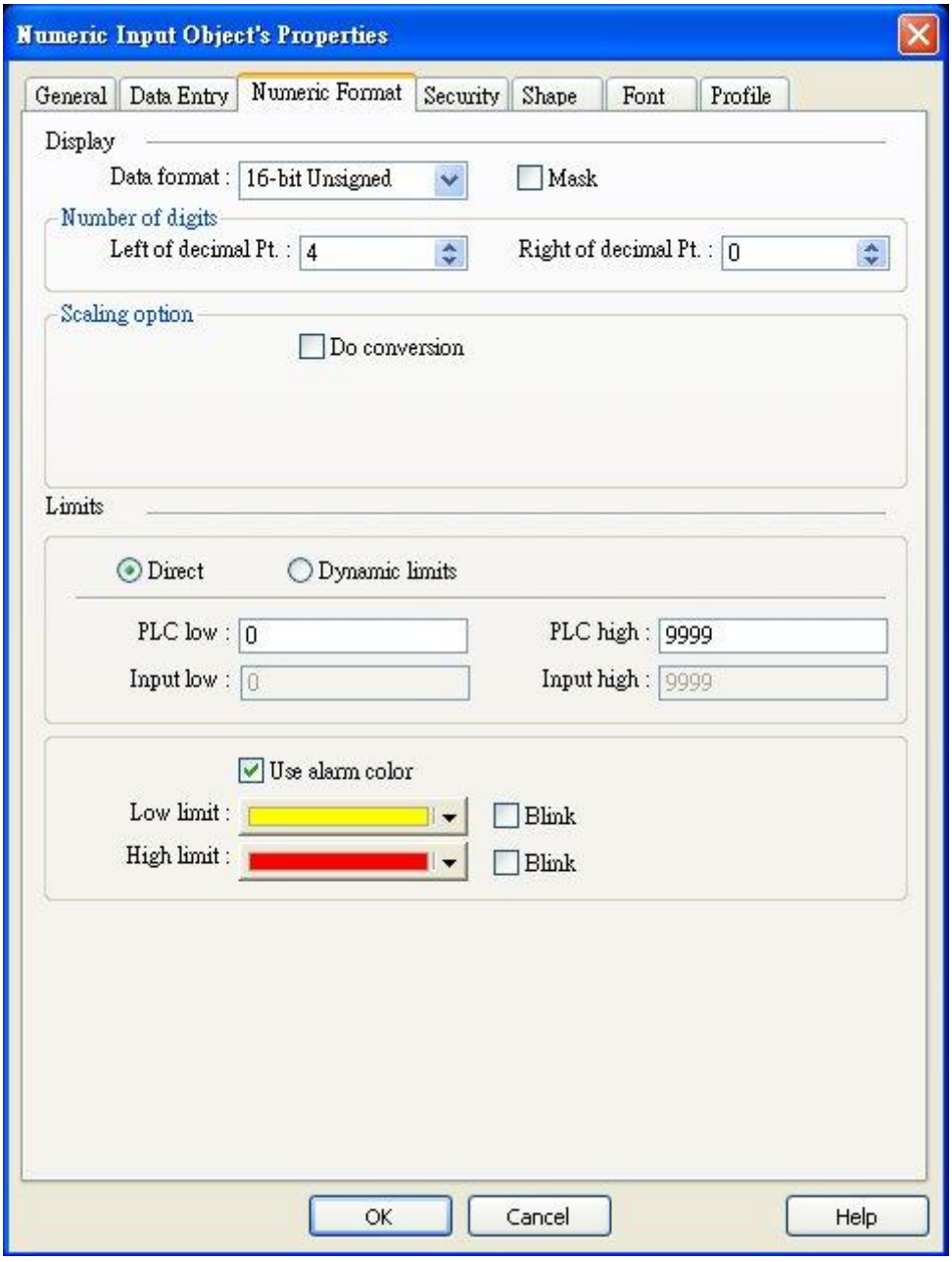
- Create three Numeric Input Objects, and set [Input order] to 1, 2, and 3 respectively. Include the three objects in [Group 1] as shown below.

Address	Setting
LW – 0	Input order Enable Stop sequential input function after input Input order : 1 Group 1
LW – 1	Input order Enable Stop sequential input function after input Input order : 2 Group 1
LW – 2	Input order Enable Stop sequential input function after input Input order : 3 Group 1

- When finish entering data in the last object, to end data entry of all objects, please select **[Stop sequential input function after input]** check box.



The following shows the [Numeric Format] tab of [Numeric Input] and [Numeric Display] objects for setting the properties of displaying value.



Display

[Data format]

Set the data format of a designated word register. The selections are listed below. 16-bit uses 1 word where 32-bit uses two words.

Format
16-bit BCD
32-bit BCD
16-bit Hex
32-bit Hex
16-bit Binary
32-bit Binary
16-bit Unsigned
16-bit Signed
32-bit Unsigned
32-bit Signed
32-bit Float

[Mask]

If selected, any values entered will be hidden by displaying them as ****.

Number of digits

[Left of decimal Pt.]

The number of digits before the decimal point.

[Right of decimal Pt.]

The number of digits after the decimal point.

Scaling option

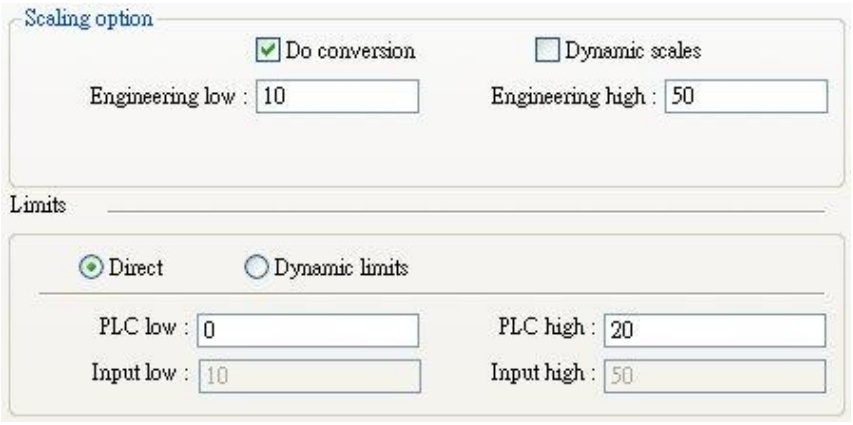
[Do conversion]

If this check box is selected, [Engineering low] and [Engineering high] boxes appear. Values entered in these boxes correspond to the display range required. The setting also requires [Input low] and [Input hight] in the limits section. If A indicates the original data and B indicates the displayed data:

B = [Engineering low] + (A - [Input low]) × ratio

where, the ratio = ([Engineering high] - [Engineering low]) / ([Input high] - [Input low])

As shown below, the original data is 15, after conversion where: 10 + (15 – 0) × (50 – 10) / (20 - 0) = 40. As a result, 40 will be displayed.



[Dynamic limits]

Set the [Engineering low] and [Engineering high] by a designated register. When Dynamic Address is LW-n, where n is an arbitrary number, the rule of setting Low / high limit is:

Content	16-bit	32-bit
Dynamic Address	LW-n	LW-n
Low limit	LW-n	LW-n
High limit	LW-n+1	LW-n+2

For example, when Dynamic Address is LW-100, the rule of setting Low / High limit is:

Content	16-bit	32-bit
Dynamic Address	LW-100	LW-100
Low limit	LW-100	LW-100
High limit	LW-101	LW-102

Limits

This section allows users to apply display limits to the values held in the input register. The color when the register value is outside limits can be set.

[Constant]

Sets the range of values set by entering values in [Input low] and [Input high]. If the value entered is outside the limits, the value in the register cannot be changed.

[Address]

Set the low / high limit by a designated register. When [Address] is LW-n, where n is an arbitrary number, the rule of setting limits is:

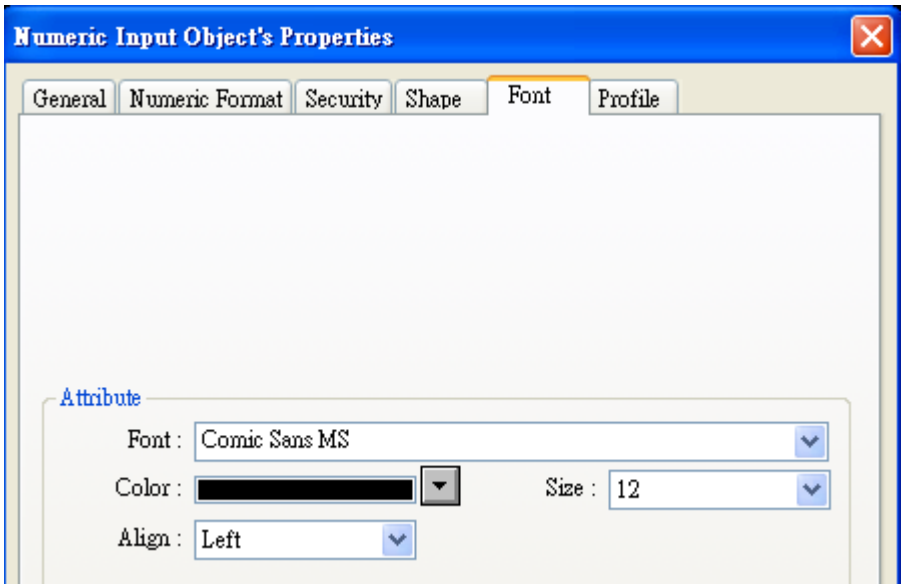
Content	16-bit	32-bit
Address	LW-n	LW-n
Low limit	LW-n	LW-n
High limit	LW-n+1	LW-n+2

For example, when [Address] is LW-100, the rule of setting limits is:

Content	16-bit	32-bit
Dynamic Address	LW-100	LW-100
Low limit	LW-100	LW-100
High limit	LW-101	LW-102

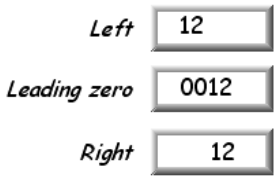
[Low limit]
When the value in the register is outside the [Low limit], display digit color set.
[High limit]
When the value in the register is outside the [High limit], display digit color set.
[Blink]
When the value in the register is outside the limits, the digits flash.

The following shows the [Font] tab of [Numeric Input] and [Numeric Display] objects for setting the properties of the displayed digits including [Font], [Color], [Size], and [Align].



Attribute

[Color]
When the value is within the limits, display digit color set in this tab.
[Align]
There are three selections: [Left], [Leading zero], and [Right].



[Size] Set the font size.

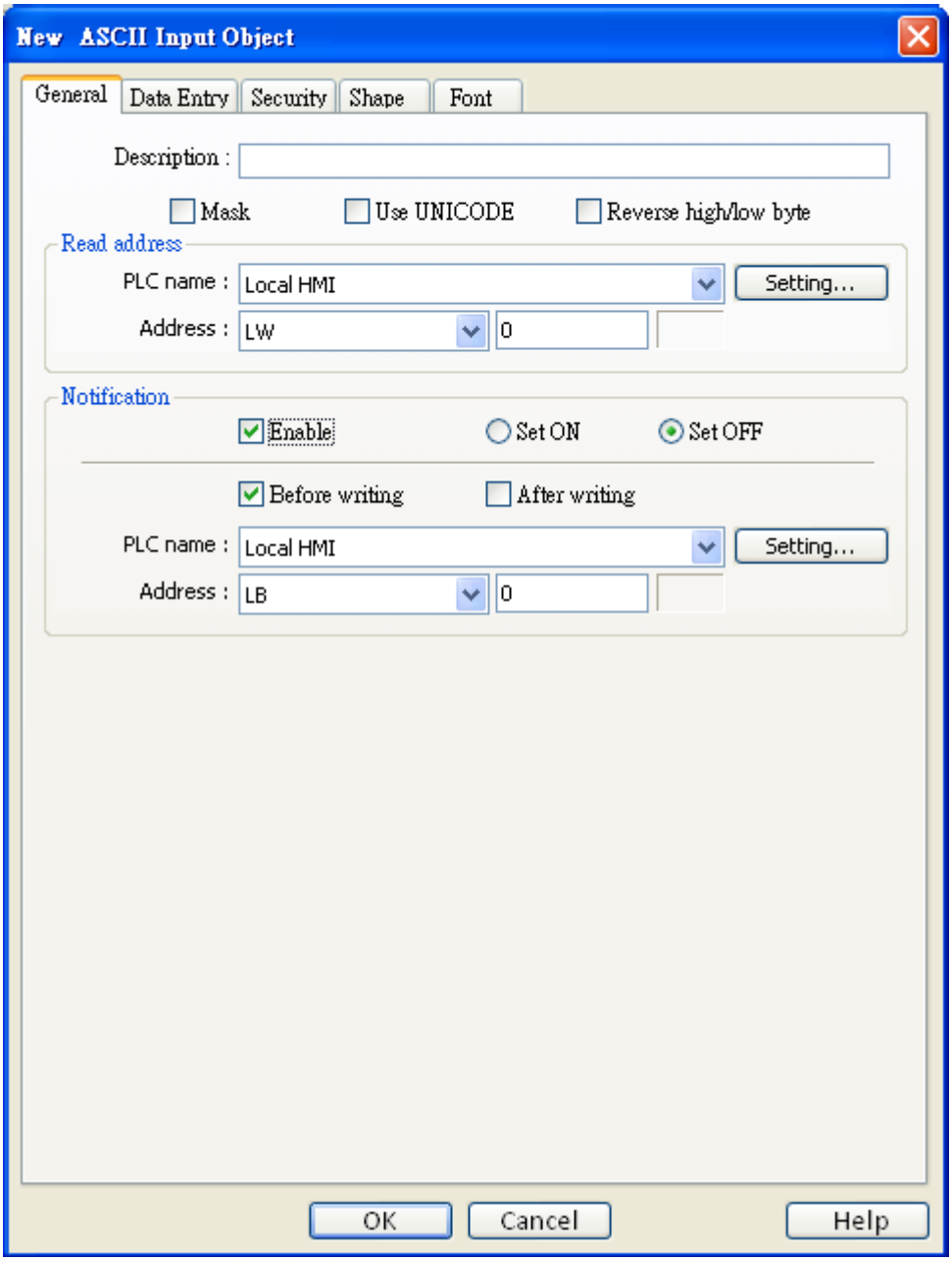
13.10 ASCII Input and ASCII Display

Overview

[ASCII Input] object and [ASCII Display] object can be used to display ASCII or UNICODE characters held in a number of sequential registers form a designated word register. [ASCII Input] object can be used to input a value into a register via a keyboard.

Configuration


Click the [ASCII Input] or [ASCII Display] icon on the toolbar to open a [ASCII Input] or [ASCII Display] object property dialog. Set up the properties, press OK button, and a new [ASCII Input] or [ASCII Display] object will be created.



[Mask]

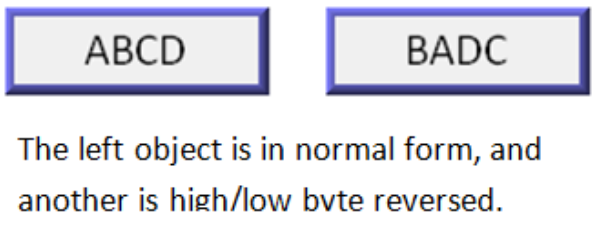
If selected, any values entered will be hidden by displaying them as ****.

[Use UNICODE]

Select this check box to display data in UNICODE format. If not selected, the characters are displayed in ASCII format. This feature can be used with the [Function Key] object [UNICODE].

[Reverse high/low byte]

Normally an ASCII code is displayed in “high byte”, “low byte” order. Reverse selection makes the system display ASCII characters in “low byte”, “high byte” order.

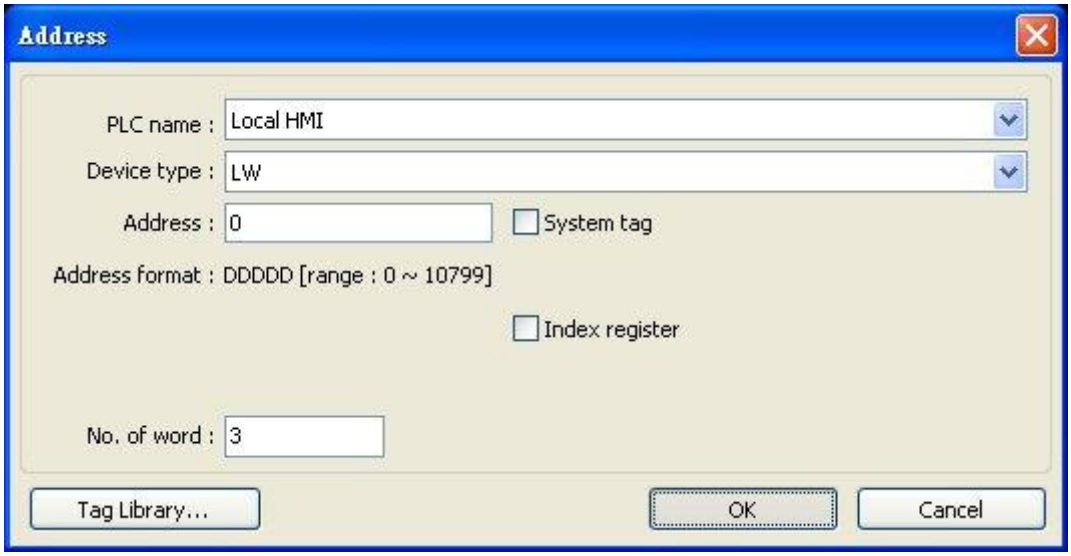


Read address

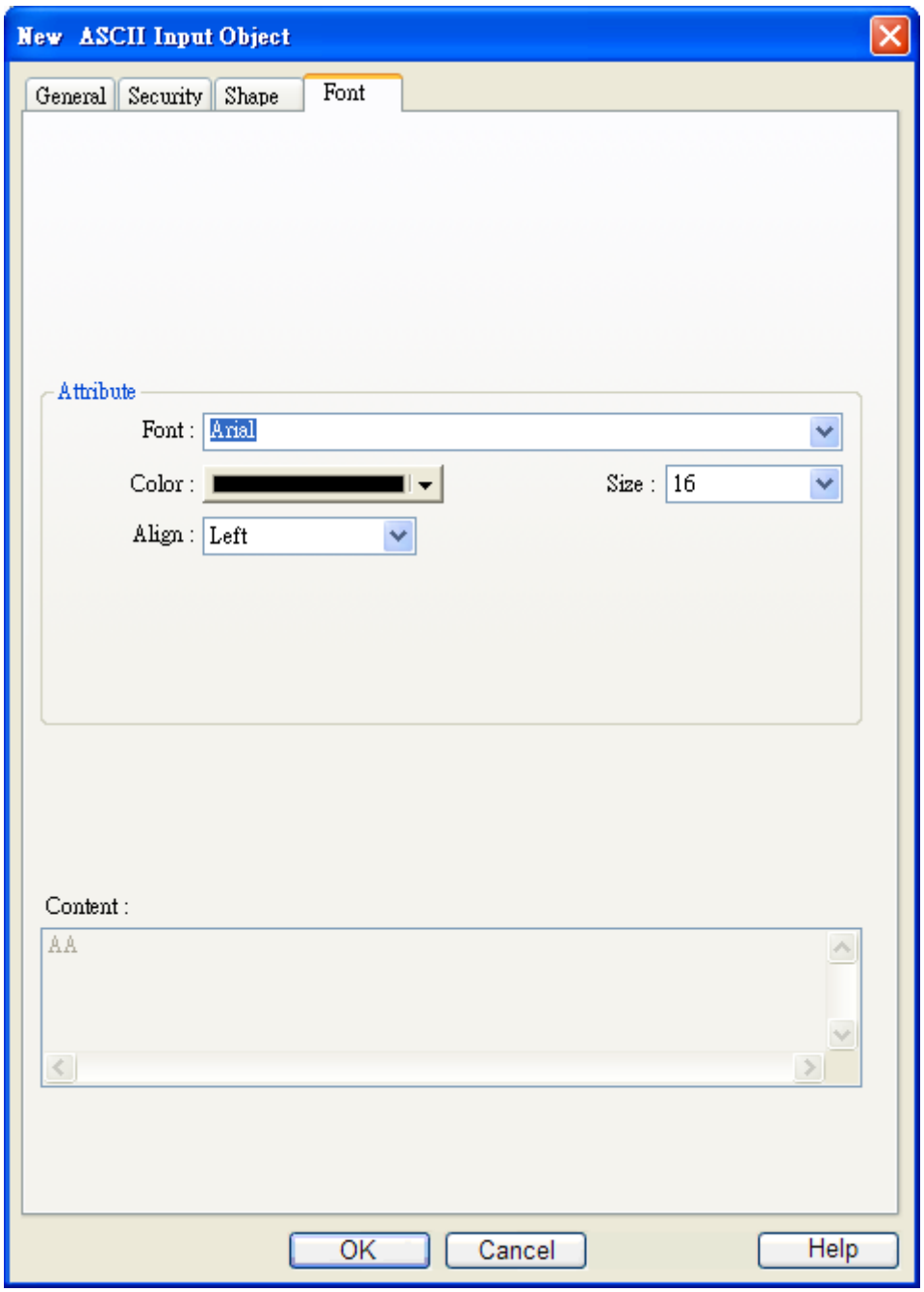
Click [Setting] to select the [PLC name], [Address], [Device type], [System tag], [Index register] of the word device that displays characters. Users can also set address in [General] tab while adding a new object.

[No. of words]

Click [Setting] to select the maximum number of words to be displayed.



An UNICODE character uses 1 word, and an ASCII character uses 1 byte. Therefore 1 word can be used as 1 UNICODE character or 2 ASCII characters. (1 word equals to 2 bytes)

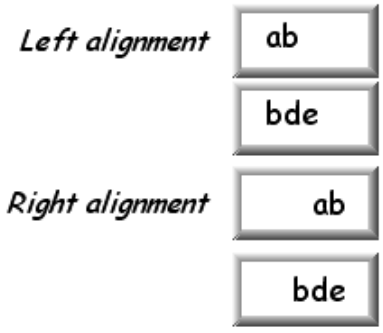


Attribute

In the [Font] tab of [ASCII Input] object and [ASCII Display] object, the font, size, color, and alignment can be set..

[Align]

Left or Right justified as shown.



13.11 Indirect Window

Overview

[Indirect Window] object calls a popup window where the window number corresponds to the value in a designated word register. The control word register is set in Indirect Window, entering a value in the register calls the corresponding number of popup window.

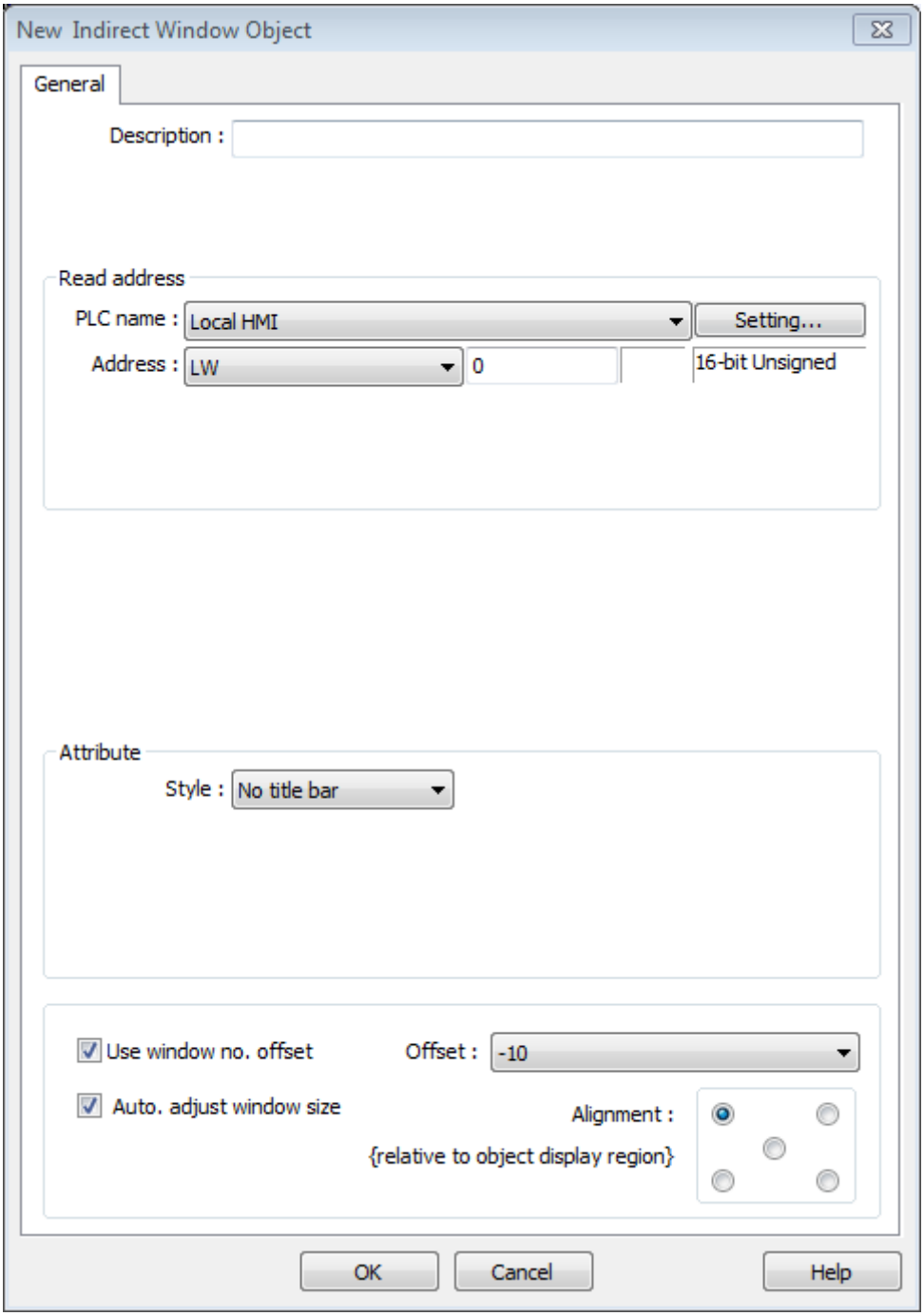
There are two ways to use Indirect Window object:

1. Preset an area for Indirect Window, in this way the popup windows all have the same size.
2. Indirect Window is automatically resized according to the size of the popup window to be displayed.

The difference between [Direct Window] and [Indirect Window] is that the display of [Direct Window] is controlled by the state of a designated bit register where the [Indirect Window] is controlled by the value in a designated word register that calls the popup window number.

Configuration

Click the [Indirect Window] icon on the toolbar to open a [Indirect Window] object property dialog. Set up the properties, press OK button, and a new [Indirect Window] object will be created.



Read address

Click [Setting] to select the [PLC name], [Address], [Device type], [System tag], [Index register] of the word device that calls the popup window. Users can also set address in [General] tab while adding a new object.

Attribute

[Style]

Set the display style of the popup window. There are two styles, [No title bar] / [With title bar].

a. No title bar

The position of the window is fixed as pre-defined in set up.

b. “With title bar”

The position of the window can be dragged during operation.

[Use window no. offset]

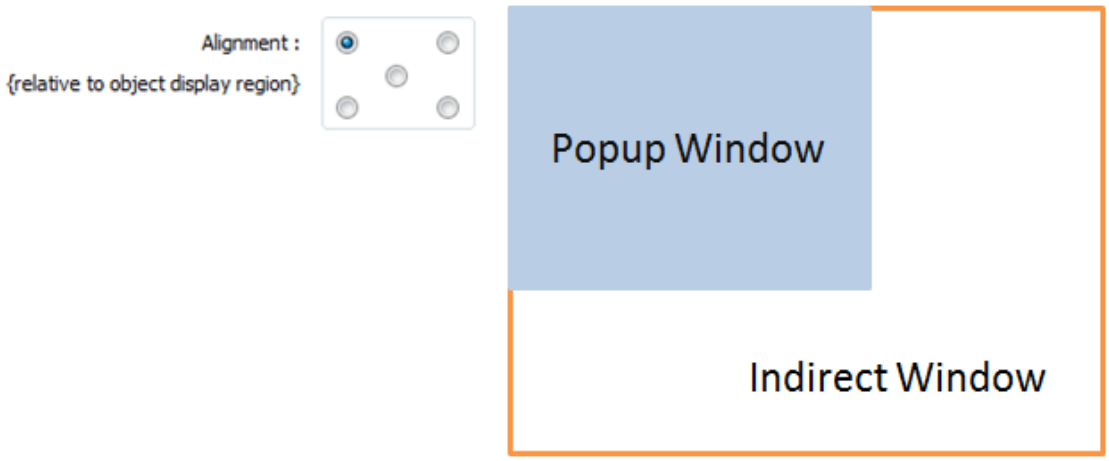
Sets the offset for selecting the popup window. The popup window number to be displayed is calculated by the following equation: value in the word register + offset = window number. The offset can be set from -1999 to 1999, not including 0.

[Auto. adjust window size]

Automatically adjust the size of the Indirect Window to the size of the popup window. Not necessary to preset the popup window size.

[Alignment:]

When [Auto. adjust window size] is selected, the system generates 5 reference points. If one reference point is selected, the popup window is displayed according to the point. See the following example, the upper-left reference point is selected, the following figure shows the position of a popup window.



Note: To use this function, please note the size and direction of the popup window to avoid covering the objects in the main window or the object size exceeds the range of the main window.

Here is an example of using Indirect Window. The setting is shown in the following figure, set the address to [LW-100] which calls the window number. Create window no. 35 and 36 first.

Use the [Set Word] object and set the value of [LW-100] to 35, the display is shown in the following figure.

Use the [Set Word] object to set the value of [LW-100] to 36, the display is shown in the following figure.

To close window no. 35 or 36, use [Set Word] object to set the value of [LW-100] to 0. Another way is to place a [Function Key] object in window no. 35 or 36, and set the key to [Close window].

- Up to 16 windows can be displayed simultaneously at run time.
- The system does not allow opening the same window with two Direct (or Indirect) windows in one base window.

13.12 Direct Window

Overview

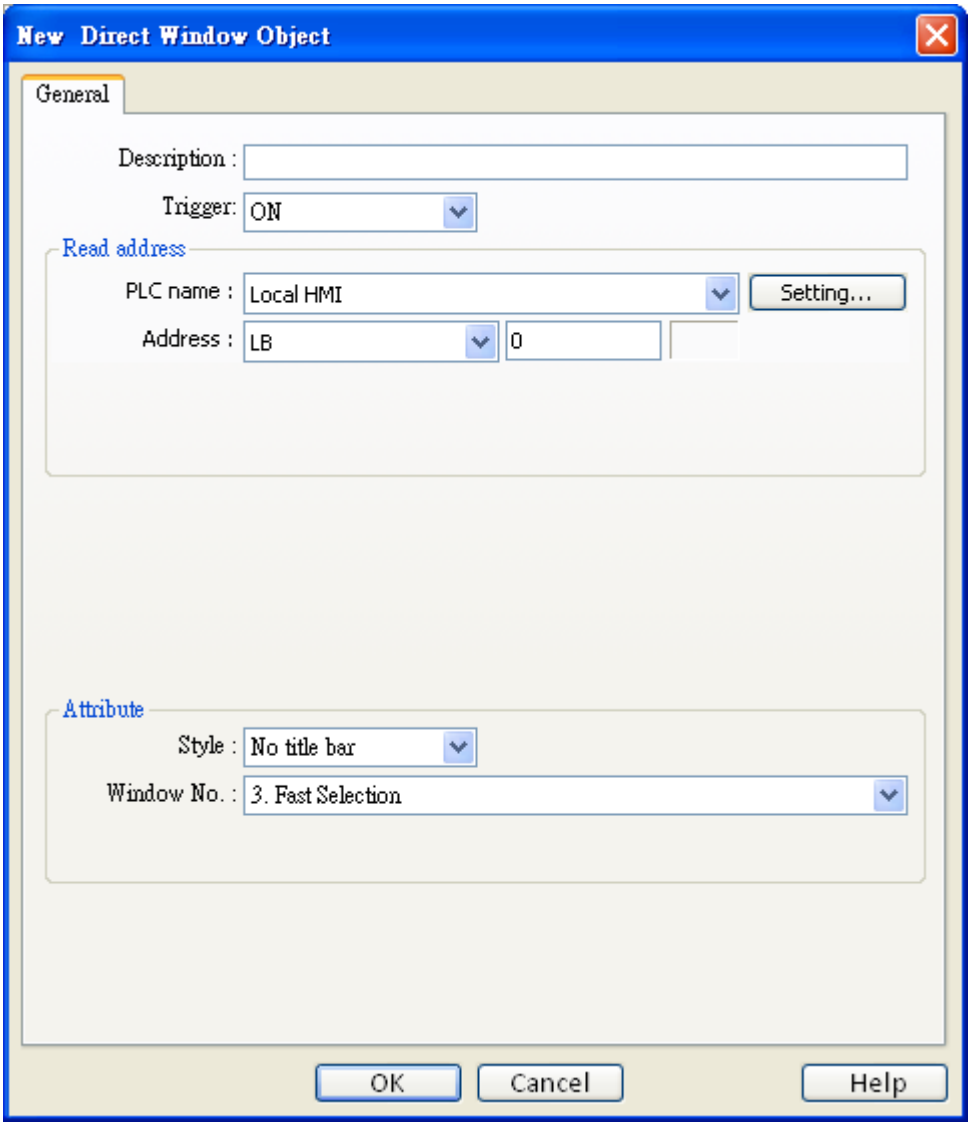
[Direct Window] object defines the position and size of a popup window location on a window. When the content of the bit register is changed, the window will pop up at the predefined location. The display area for the popup window is limited by the size of predefined location. Restore the value of the bit register to close the popup window,.

The difference between the [Direct Window] and the [Indirect Window] is that [Direct Window] object has a predefined window number. For both of them, users can use the state of the bit device register to open or close the popup window.

Simply speaking, [Direct Window] is controlled by a bit register and [Indirect Window] is controlled by a word register.

Configuration

Click the [Direct Window] icon on the toolbar to open a [Direct Window] object property dialog. Set up the properties, press OK button, and a new [Direct Window] object will be created.



Read address

Click **[Setting]** to select the **[PLC name], [Device type], [Address], [System tag], [Index register]** of the bit device that control the window popup. Users can also set the address in **[General]** tab while adding a new object.

Attribute

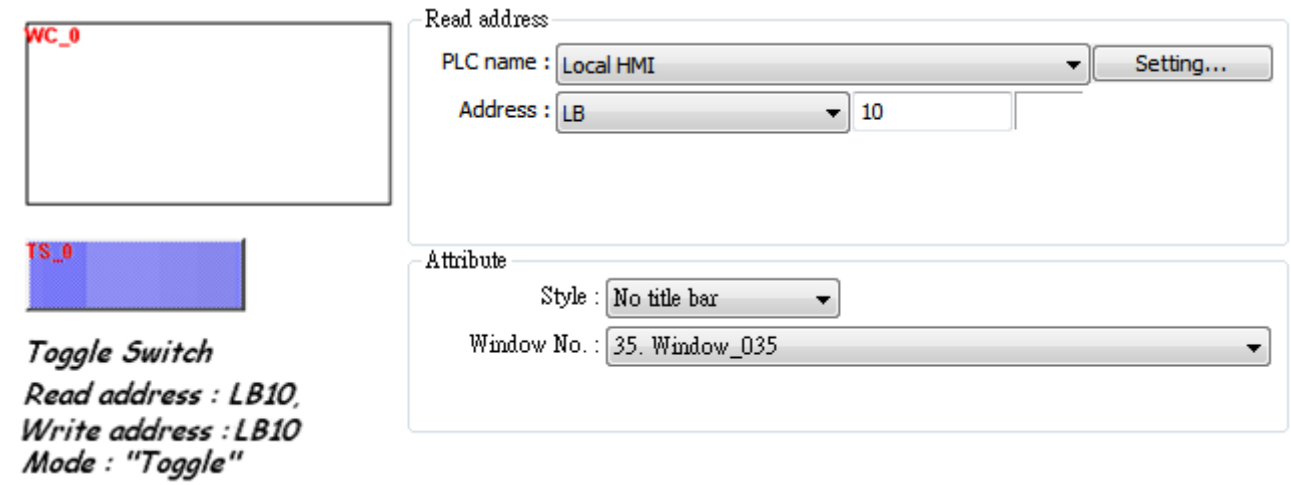
[Style]

Define the popup window style. Two styles are available, [No title bar] and [With title bar]

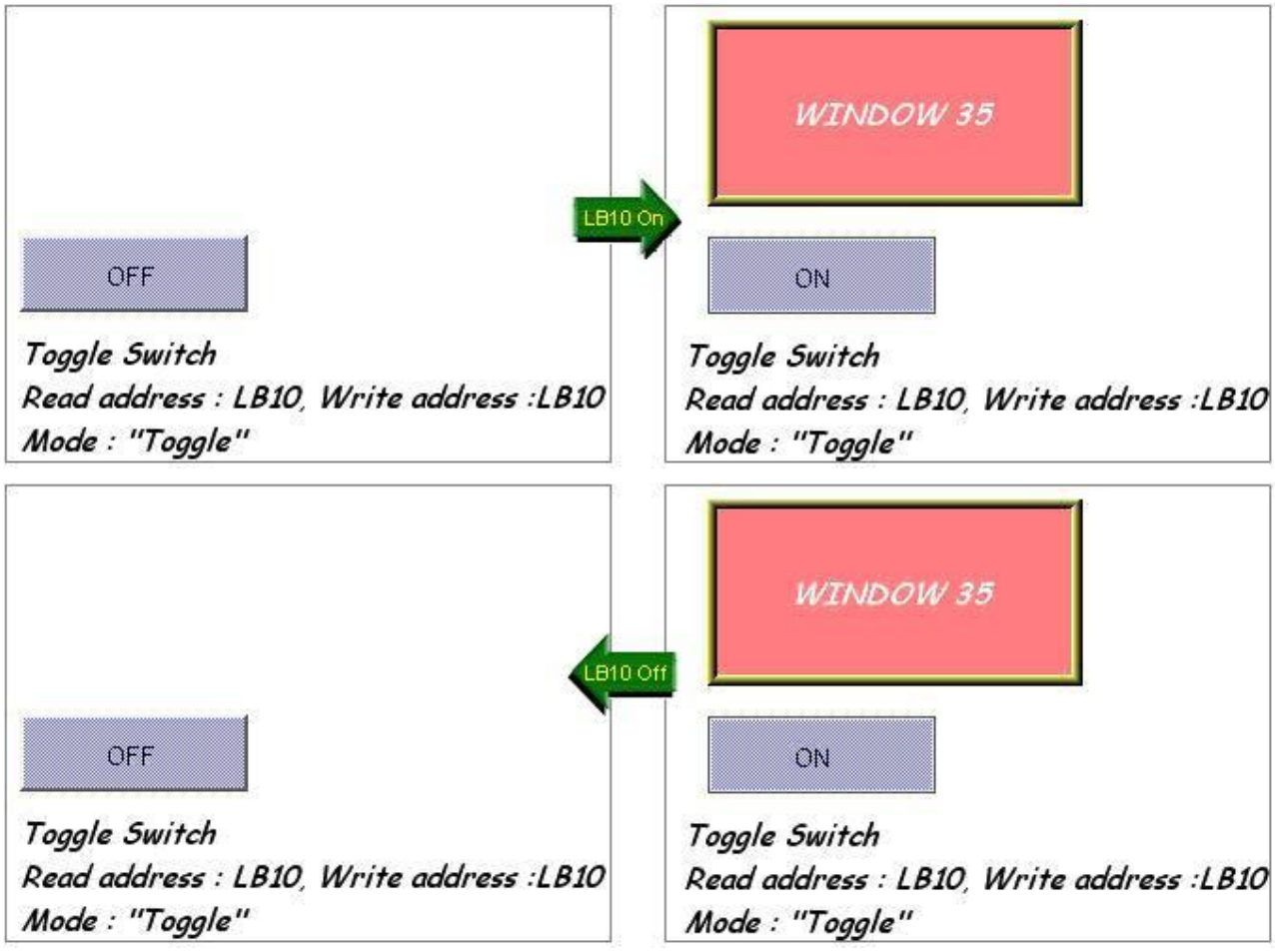
[Window no.]

Set the popup window number.

Here is an example to explain how to use the [Direct Window] object. The picture below shows the settings of the [Direct Window] object. In the example, use LB-10 to call up window no. 35.



If the state of LB-10 turned ON, window no. 35 will be popup; if the state of LB-10 turned OFF, window no. 35 will be closed. See the picture below.



Note

- A screen can display up to 16 popup windows simultaneous including System Message Window, Direct Window and Indirect Window.
- The system does not allow opening the same window with two Direct (or Indirect) Windows in one base window.

13.13 Moving Shape

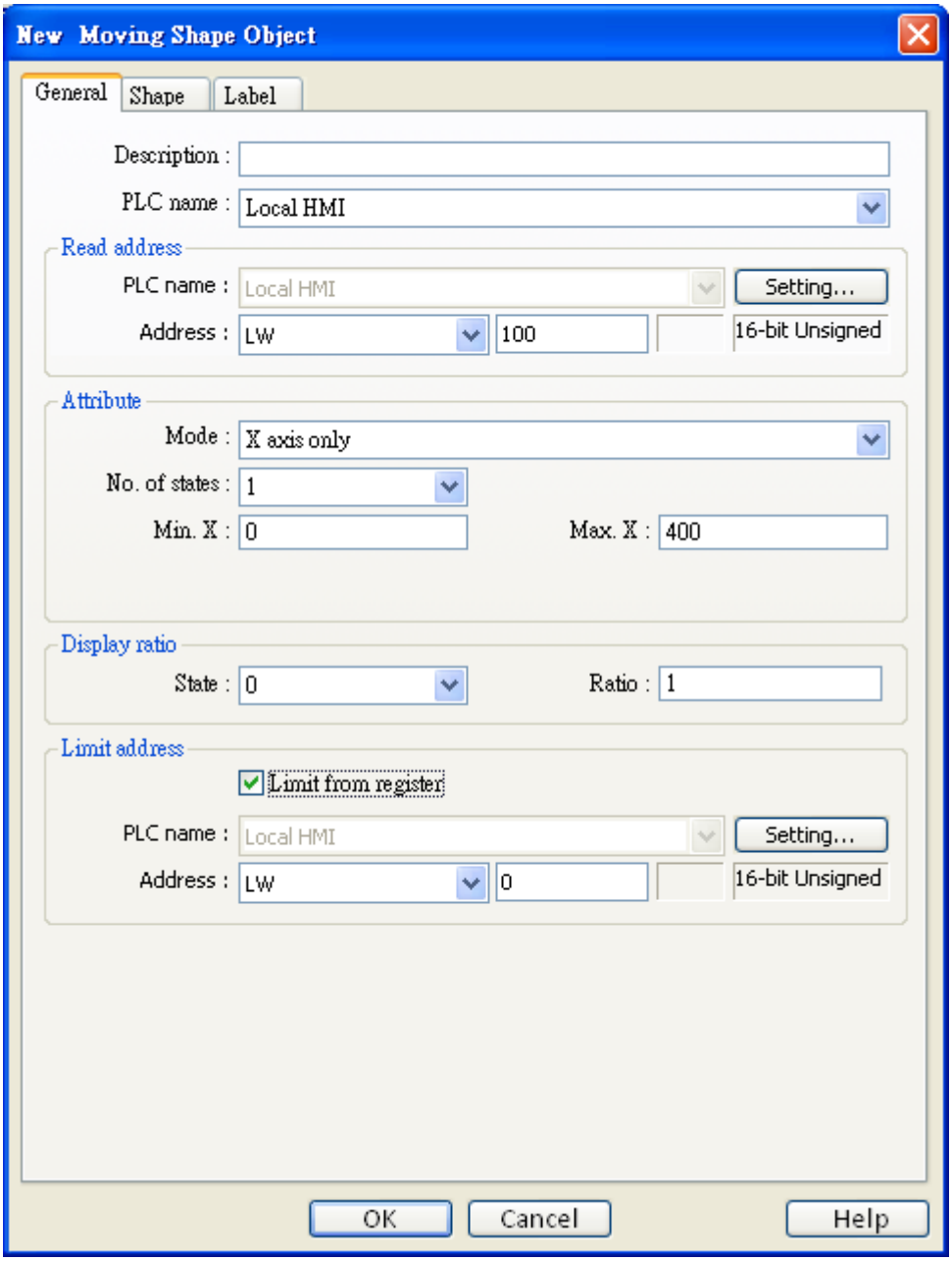
Overview

[Moving Shape] object defines the states and moving distance of an object. The state and the location of the object depend on three consecutive PLC registers. The first register controls the state of the object, the second register controls the horizontal position (X), and the third register controls the vertical position (Y).

Data format	Object state	Moving Distance on the X-axis	Moving distance on the Y-axis
16-bit	Address (LW-n)	Address + 1 (LW-n+1)	Address + 2 (LW-n+2)
32-bit	Address (LW-n)	Address + 2 (LW-n+2)	Address + 4 (LW-n+4)

Configuration

Click the [Moving Shape] icon on the toolbar to create a [Moving Shape] object. Set up the properties, press OK button, and a new [Moving Shape] object will be created.



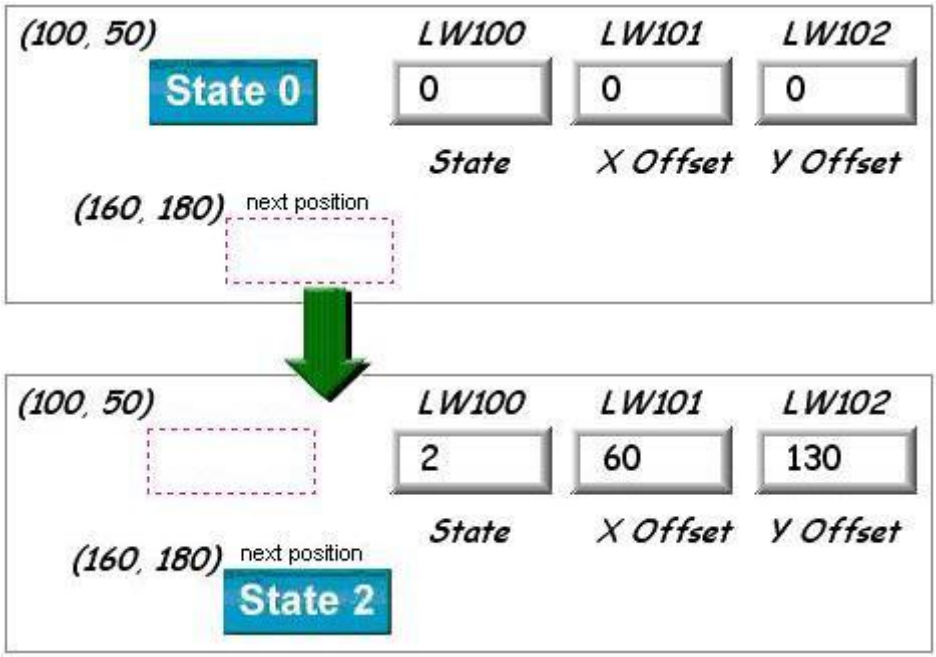
Read address

Click **[Setting]** to configure the **[PLC name]**, **[Device type]**, **[Address]**, **[System tag]**, or **[Index register]** of the word devices that control the display of object's state and moving distance. Users can also set the address in **[General]** tab while adding a new object. The table below shows the address in different data format:

Data format	Object state	Moving Distance on the X-axis	Moving distance on the Y-axis
16-bit	Address (LW-n)	Address + 1 (LW-n+1)	Address + 2 (LW-n+2)
32-bit	Address (LW-n)	Address + 2 (LW-n+2)	Address + 4 (LW-n+4)

For example, if the object's read address is LW-100 and the data format is [16-bit Unsigned], LW-100 is used to control the object's state, LW-101 is used to control the object's moving distance on the X-axis, and LW-102 is used to control the object's moving distance on the Y-axis.

The picture below shows that the object's read address is LW-100 and initial position is (100, 50). Supposed you want to move the object to the position (160,180) and change its state to State 2, then, assign 2 to LW-100, 160-100 = 60 to LW-101, 180-50 = 130 to [LW102].

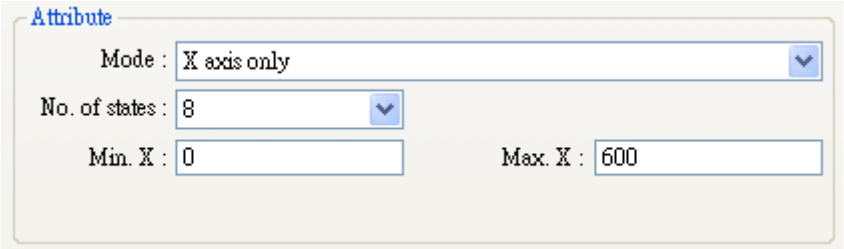


Attribute

Select the object's movement mode and range.

a. X axis only

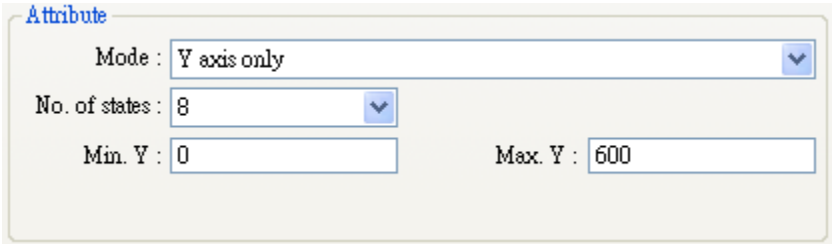
The object is only allowed to move along the X-axis. The moving distance ranges from [Min. X] to [Max. X].



Data format	Object state	Moving Distance on the X-axis
16-bit format	Address (LW-n)	Address + 1 (LW-n+1)
32-bit format	Address (LW-n)	Address + 2 (LW-n+2)

b. Y axis only

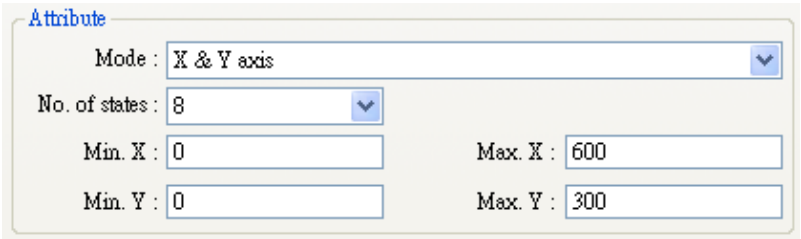
The object is only allowed to move along the Y-axis. The moving distance ranges from [Min. Y] to [Max. Y].



Data format	Object state	Moving Distance on the Y-axis
16-bit format	Address (LW-n)	Address + 1 (LW-n+1)
32-bit format	Address (LW-n)	Address + 2 (LW-n+2)

c. X & Y axis

The object is allowed to move along the X-axis and Y-axis. The moving range in X and Y direction is defined by [Min. X], [Max. X] and [Min. Y], [Max. Y] respectively.



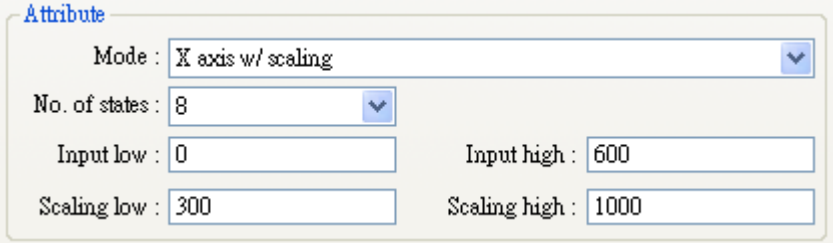
Data format	Object state	Moving Distance on the X-axis	Moving distance on the Y-axis
16-bit format	Address (LW-n)	Address + 1 (LW-n+1)	Address + 2 (LW-n+2)
32-bit format	Address (LW-n)	Address + 2 (LW-n+2)	Address + 4 (LW-n+4)

d. X axis w/ scaling

The object moves in X-axis only with scaling. Supposed that the value of the designated register is DATA, the system uses the following formula to calculate the moving distance on the X-axis.

X axis move distance =

(DATA – [Input low]) * ([Scaling high – Scaling low]) / ([Input high] – [Input low])



For example, the object is only allowed to move within 0-600, but the range of the register's value is 300-1000. Set [Input low] to 300 and [Input high] to 1000, and set [Scaling low] to 0 and [Scaling high] to 600. And the object will move within the defined range.

Data format	Object state	Moving Distance on the X-axis
16-bit format	Address (LW-n)	Address + 1 (LW-n+1)
32-bit format	Address (LW-n)	Address + 2 (LW-n+2)

e. Y axis w/ scaling

The object is for Y axis movement with scale, and the formula to calculate the moving distance on the Y-axis is the same as the one in [X axis w/ scaling].

Data format	Object state	Moving Distance on the Y-axis
16-bit format	Address (LW-n)	Address + 1 (LW-n+1)
32-bit format	Address (LW-n)	Address + 2 (LW-n+2)

f. X axis w/ reverse scaling

This works in the way as [X axis w/ scaling], but the moving direction is in reverse.

g. Y axis w/ reverse scaling

This works in the way as [Y axis w/ scaling], but the moving direction is in reverse.

Display ratio

The size of shape in different states can be set individually as shown in the picture below.

Limit address

The object's moving range can be set not only by [Min. X], [Max. X] and [Min. Y] [Max. Y], but also by the designated registers. Supposed that the object's moving range is set by the value of the designated register "Address", then the address of [Min. X], [Max. X] and [Min. Y] [Max. Y] are listed in the following table.

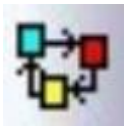
Data format	[Min. X] address	[Max. X] address	[Min. Y] address	[Max. Y] address
16-bit format	Address	Address + 1	Address + 2	Address + 3
32-bit format	Address	Address + 2	Address + 4	Address + 6

13.14 Animation

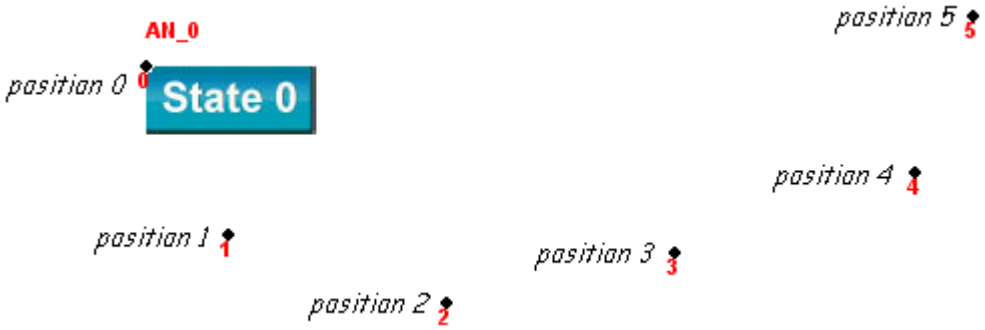
Overview

[Animation] object is defined by a pre-defined path and states. It will move to the given point and show in the given state as defined by registers. The object state and position depend on current value of two consecutive registers. The first register controls the state of the object and the second register controls the position along the predefined path.

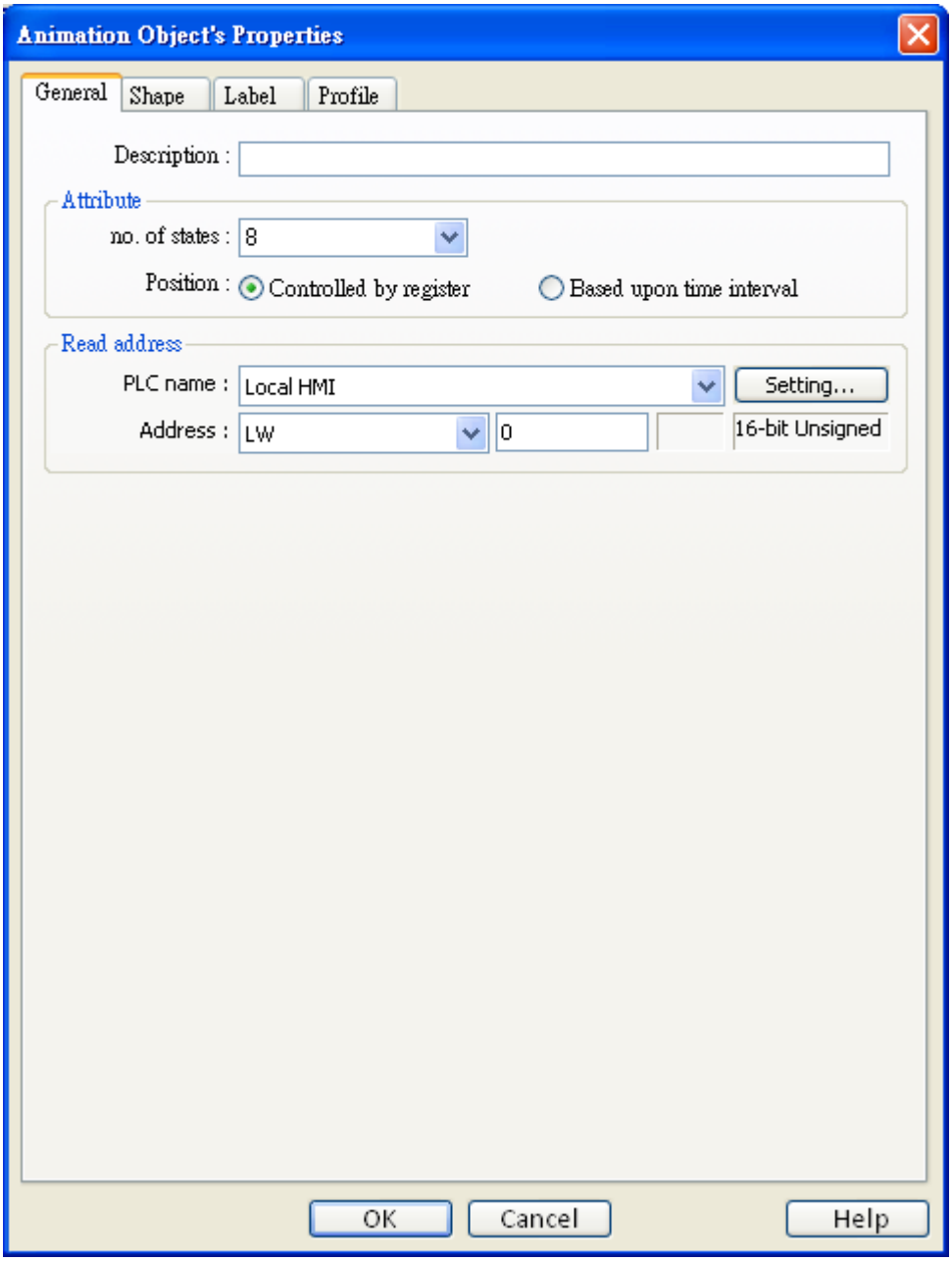
Configuration



Click the [Animation] icon on the toolbar. First, create the pre-defined path. Move the mouse to each moving position, and click the left button to define positions one by one. When it is done, right click on the screen, set up the properties, press OK button, and a new [Animation] object will be created.



To change the object's attributes, you can double click on the object to open [Animation] object's properties dialog box, as shown in the picture below.



Attribute

[Total no. of states]

This configures the number of the states for this object.

a. Controlled by register

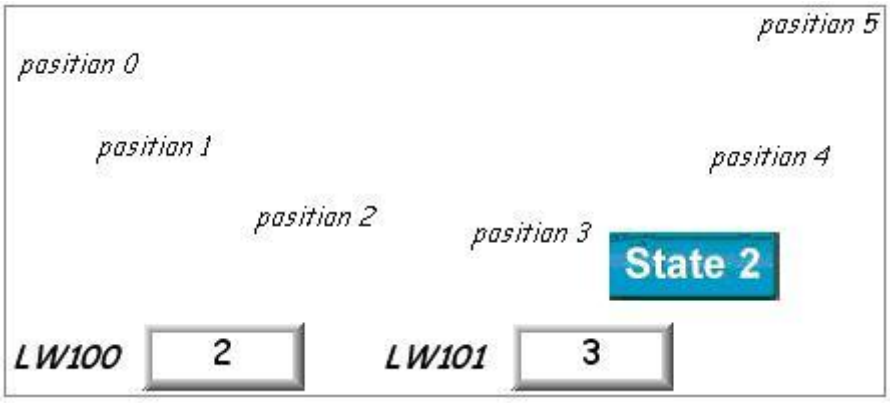
Select [Controlled by register] to use designated registers to control the object's state and position.

Read address

The object's state and position are determined by the registers, and the addresses must be configured correctly, as in the following table:

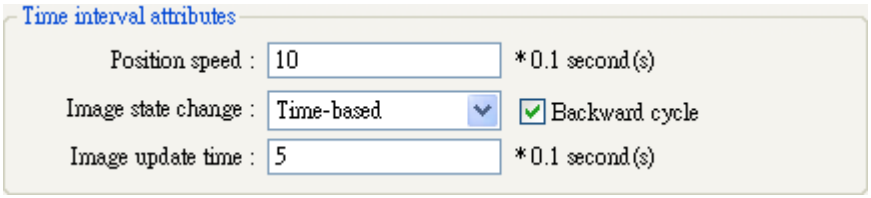
Data Format	Object's state	Object's position
16-bit format	Address (LW-n)	Address + 1 (LW-n+1)
32-bit format	Address (LW-n)	Address + 2 (LW-n+2)

For example, if the designated register is LW-100 and the data format is [16-bit Unsigned], then LW-100 represents object's state, LW-101 represents position. In the picture below, LW-100 = 2, LW-101 = 3, so the object's state is 2 and position is 3.



b. Based upon time interval

If [Based upon time interval] is chosen, the object's state and position will change from time to time. [Time interval attributes] is used to set the time interval for states and positions.



[Position speed]

The speed of change of position. The unit is 0.1 second. Supposed that [Speed] is set to 10, the object position will change each second.

[Backward cycle]

Assumed the object has four positions: position 0, position 1, position 2, and position 3, and [Backward cycle] is not selected. When the object moves to the last position (position 3), the next position will be back to the initial position 0, and repeat the same when it moves to position 3 again. The moving path is shown as follows:

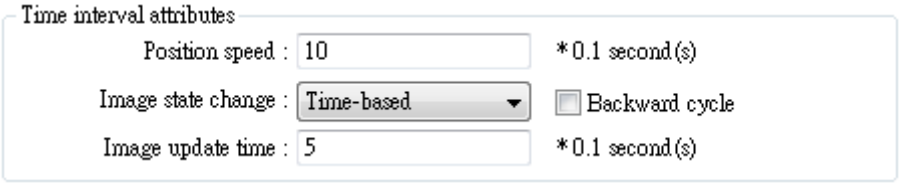
position 0 → position 1 → position 2 → position 3 → position 0 → position 1 → position 2…

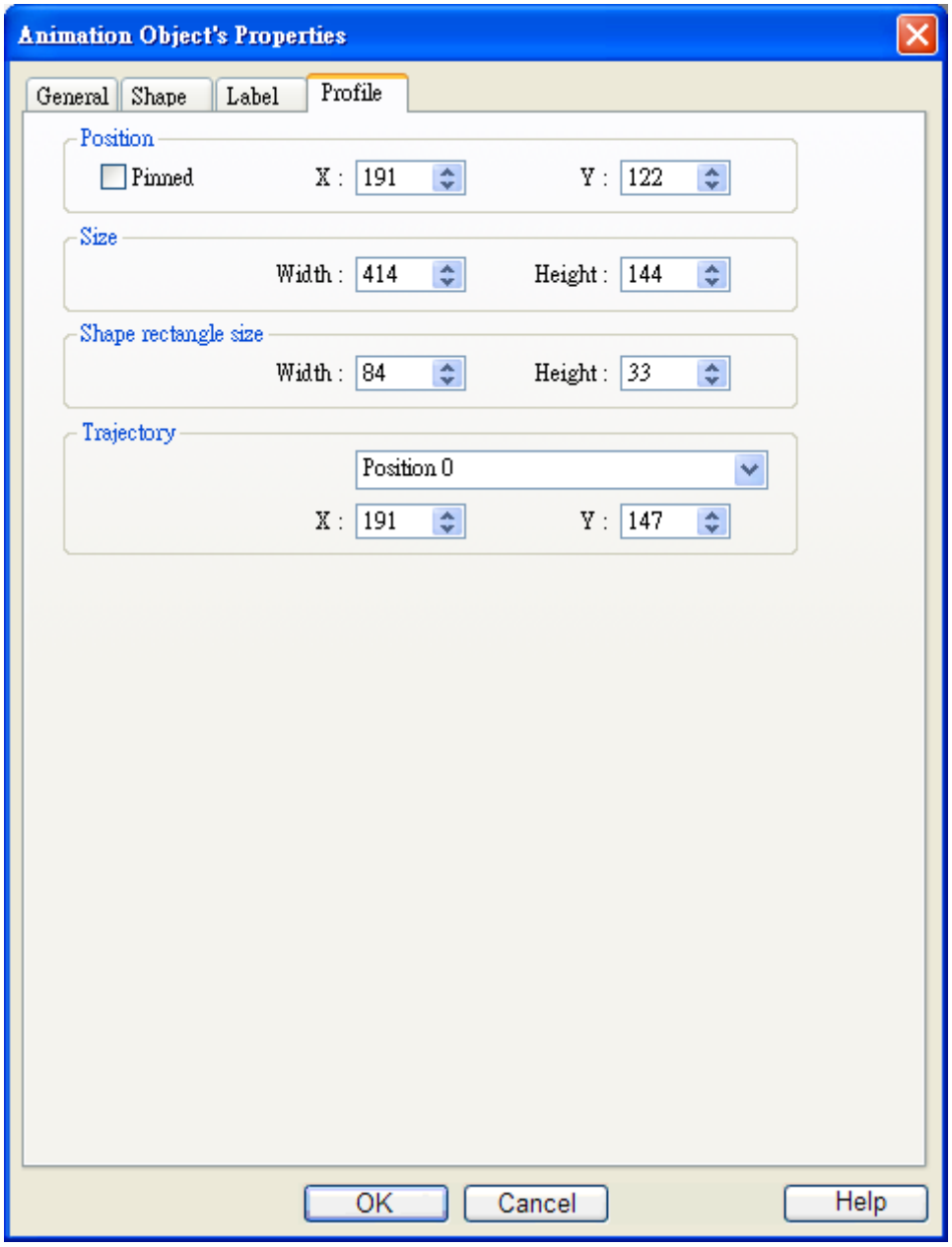
If [Backward cycle] is selected, when the object moves to the last position (position 3), it will move backwards to position 2, position 1 and then the initial position 0, and start over again. The moving path is shown as follows.

position 0 → position 1 → position 2 → position 3 → position 2 → position 1 → position 0…

[Image state change]

Determine how state changes, either [Position dependant] or [Time-based]. If [Position dependant] is selected, the object state will change when position changes. If [Time-based] is selected, the object position will change based on [Position speed] and the object state will change based on [Image update time], as shown below:





Shape rectangle size

To set the size of the shape.

Trajectory

To set the position of each point on the moving path.

- Because multiple pictures might be used by an [Animation] object, [Set to original dimension] will not return all pictures to the original size.

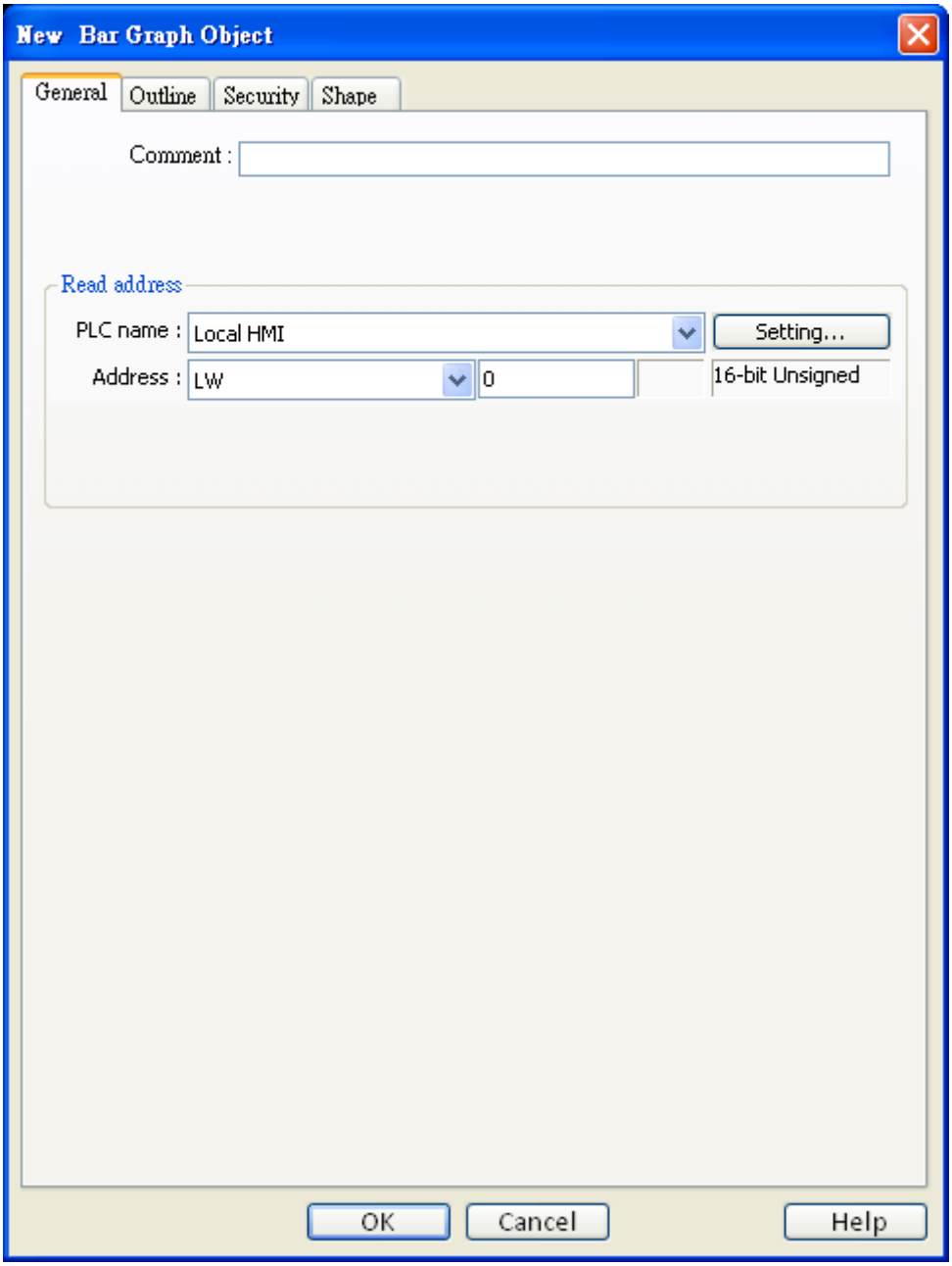
13.15 Bar Graph

Overview

[Bar graph] object displays data as a bar graph in proportion to its value.

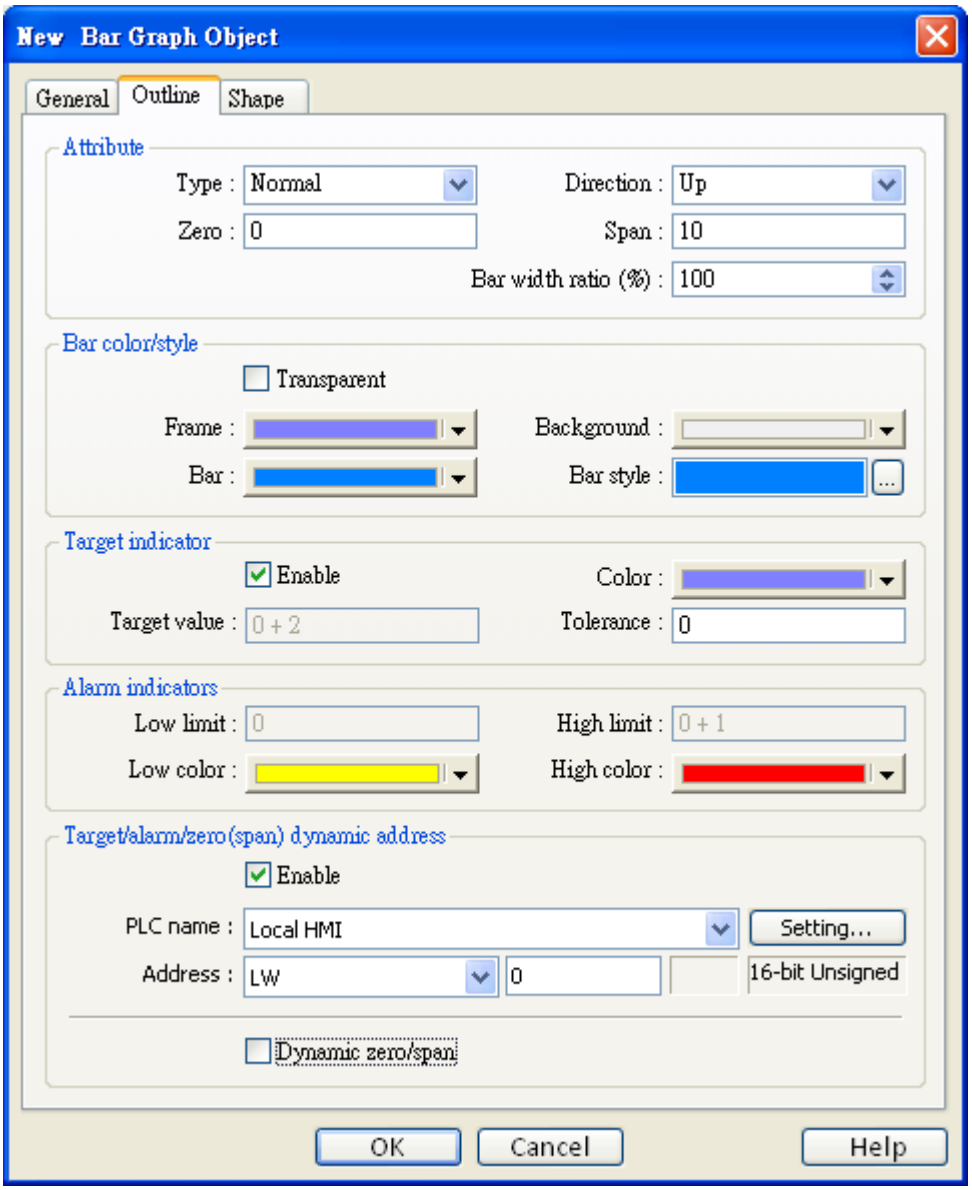
Configuration

Click [Bar Graph] icon on the toolbar to open [Bar Graph] dialog box. Fill in properties, click OK button, and then create a new [Bar Graph] object.



Read address

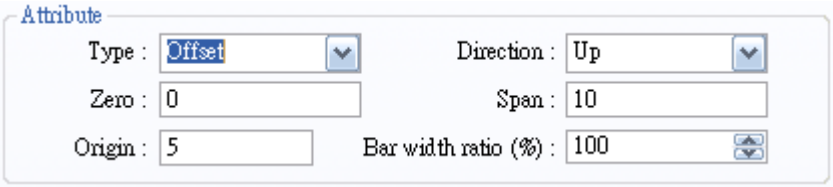
Click **[Setting]** to Select the **[PLC name], [Device type], [Address], [System tag]**, and **[Index register]** of the word devices that controls how the bar graph displays. Users can also configure address in **[General]** tab while adding a new object.



Attribute

[Type]

Choose either [Normal] or [Offset]. When [Offset] is selected, an original value must be entered for reference. Please refer the illustration below.



[Direction]

Determine the bar graph direction. Available options are [Up], [Down], [Right], and [Left].

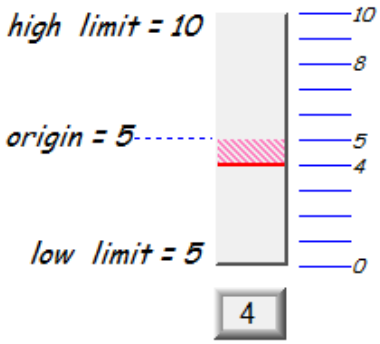
[Zero], [Span]

The percentage of filling can be calculated by the following formula:

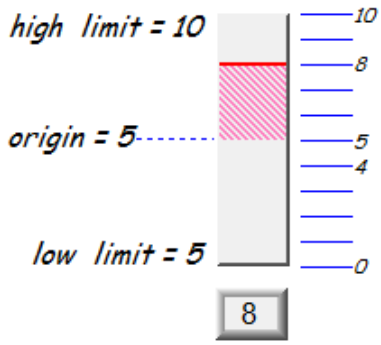
The percentage of filling = (Register value – [Zero]) / [Span] – [Zero]) * 100%

Assume [Offset] is selected. If (Register value – [Zero]) is greater than 0, the bar will fill up from [Origin]. If (Register value – Zero) is less than 0, the bar will be drawn below [Origin].

For example, [Origin] is 5, [Span] is 10, and [Zero] is 0.
For different value in read address, it will display as below:
If the value at read address is 4:

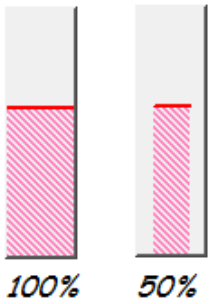


If the value at read address is 8,



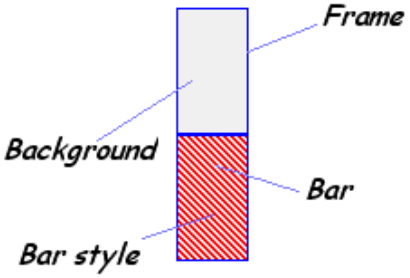
[Bar width ratio(%)]

It is the ratio of bar to object width. The figure below shows two ratios, 50% and 100%.



Bar color/style

To set the bar's frame and background color, bar style, and bar color. See the picture below.

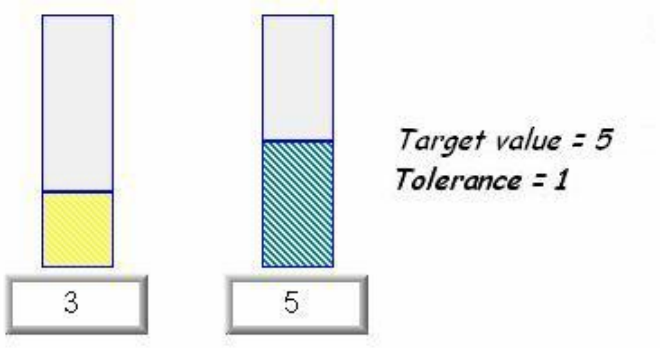


Target Indicator

When the register value meets the following condition, the color of filled area will change to the target color.

[Target Value] - [Tolerance] ≤ Register value ≤ [Target Value] + [Tolerance]

Assume [Target Value] is 5 and [Tolerance] is 1. As shown below, if the register value is equal to or larger than 4 (=5-1) and equal to or less than 6 (=5+1), the filled area's color of the bar will change to the target color.

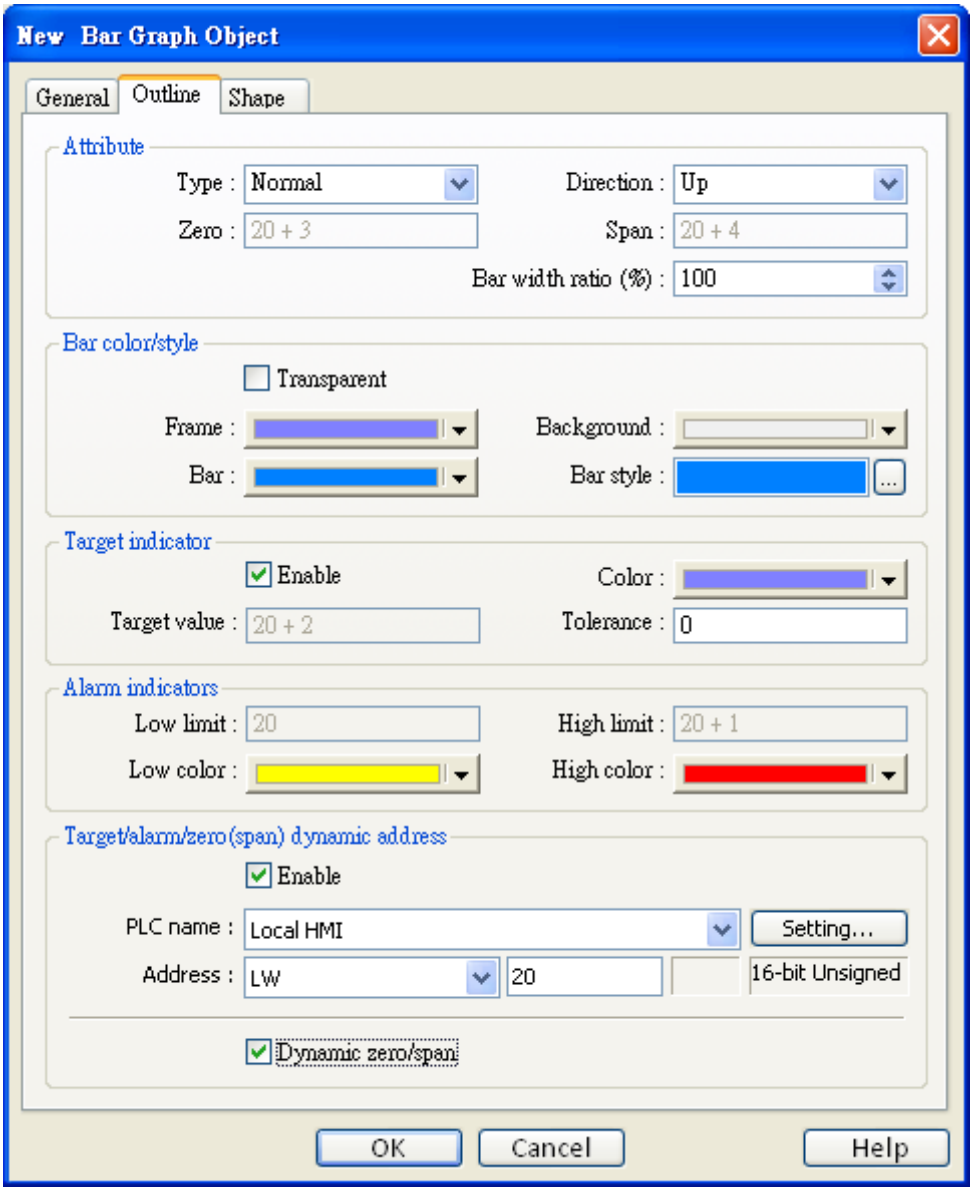


Alarm Indicator

If the register value is larger than [High limit], the color of filled area will change to [High color]. If the register value is smaller than [Low limit], the color of filled area will change to [Low color].

Target/Alarm Dynamic Address

When [Enable] is ticked, the [Low limit] and [High limit] of [Alarm indicator] and the [Target Value] of [Target indicator] will use from the value of designated registers. See the picture below.



The following table shows the read address of low limit, high limit, and target. The “Address” means the device address. For example, if the device address is LW-20 and data format is

16-bit, [Low limit] is LW 20, [High limit] is LW21, [Target indicator] is LW22, [Zero] is LW23, and [Span] is LW24.

Data Format	Alarm Low limit	Alarm High limit	Target indicator	Zero	Span
16-bit format	Address	Address + 1	Address + 2	Address + 3	Address + 4
32-bit format	Address	Address + 2	Address + 4	Address + 6	Address + 8

13.16 Meter Display

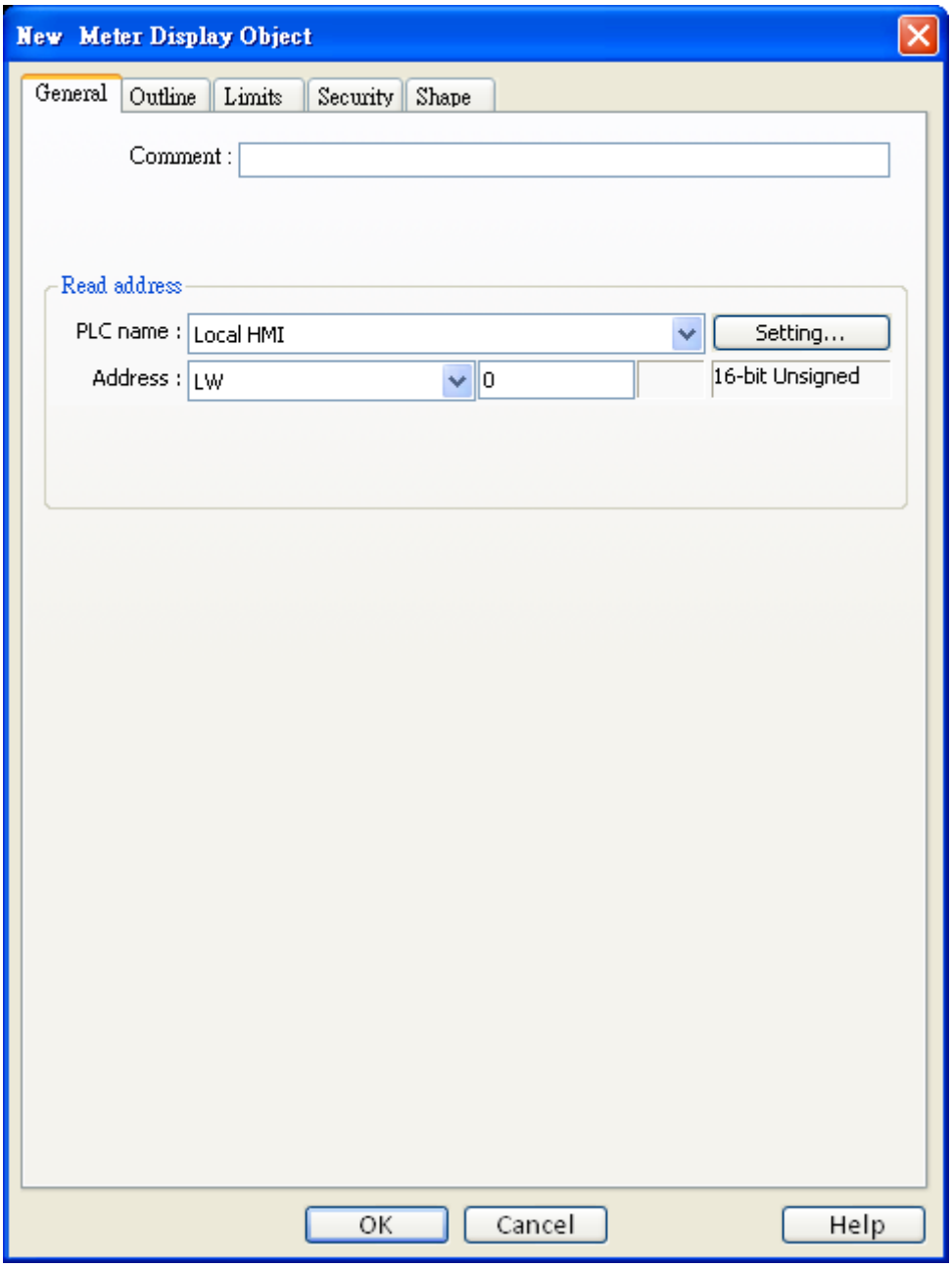
Overview

[Meter] object can display the value of word device by meter.

Configuration

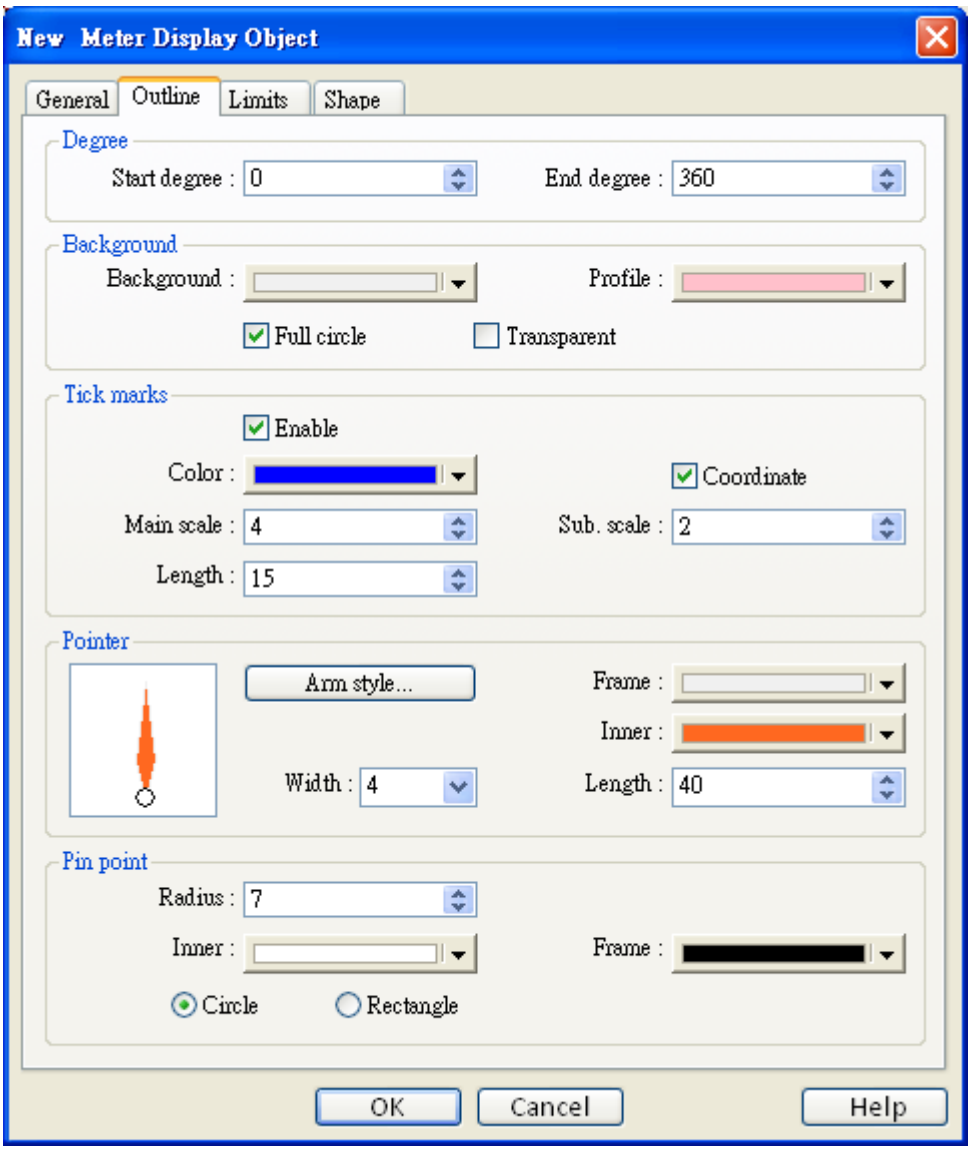
Click [Meter Display] icon on the toolbar to open [Meter Display] dialog box. Fill in properties, click OK button, and then create a new [Meter Display] object.

The picture below shows the [General] tab in the [Meter Display] object properties dialog:

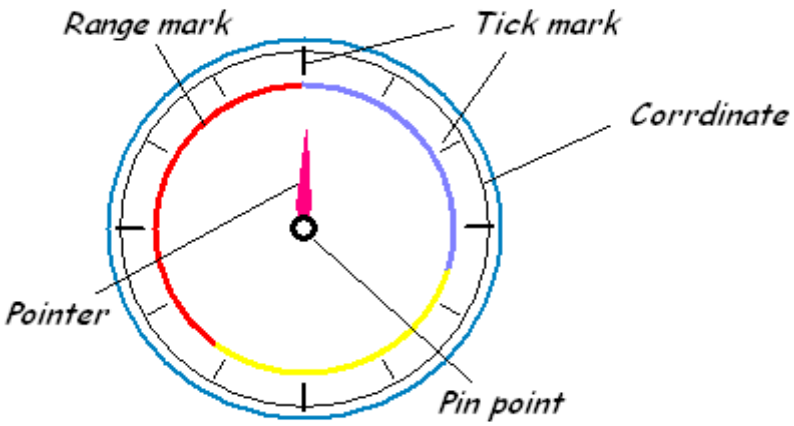


Read address

Click **[Setting]** to select the **[PLC name], [Device type], [Address], [System tag],** and **[Index register]** of the word devices that controls the display of meter. Users can also set address in **[General]** tab while adding a new object.

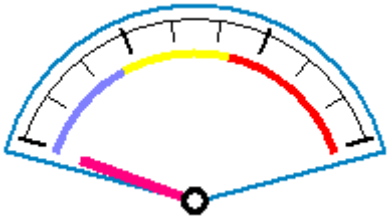


In the above dialog box, users can set the meter display object's outline. Refer to the picture below for the names of each part of the meter.

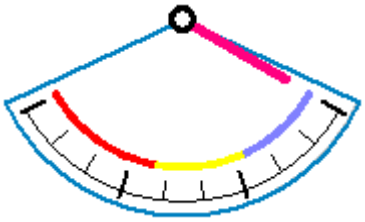


Degree

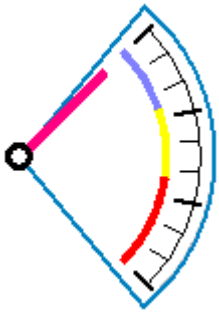
Set the object's start degree and end degree. The angle range is 0-360 degrees. The following pictures show several results of different settings.



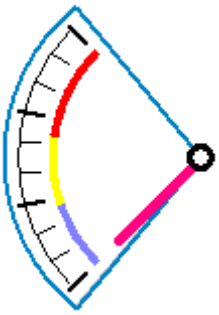
[Start degree] = 290, [End degree] = 70



[Start degree] = 120, [End degree] = 240



[Start degree] = 40, [End degree] = 140



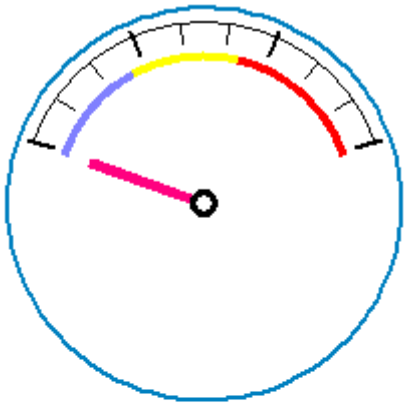
[Start degree] = 225, [End degree] = 315

Background

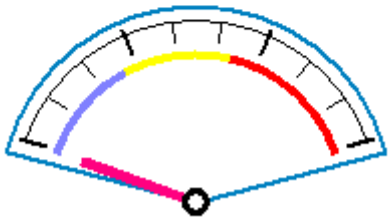
Set the object's background color and profile color.

[Full circle]

When [Full circle] is selected, the object will display the whole circle. Otherwise, the object will display the defined degree range. See the picture below.



Full circle



non-full circle

[Transparent]

When [Transparent] is selected, the object will not display the background and profile color.

Tick marks

Configure the tick mark's number and color.

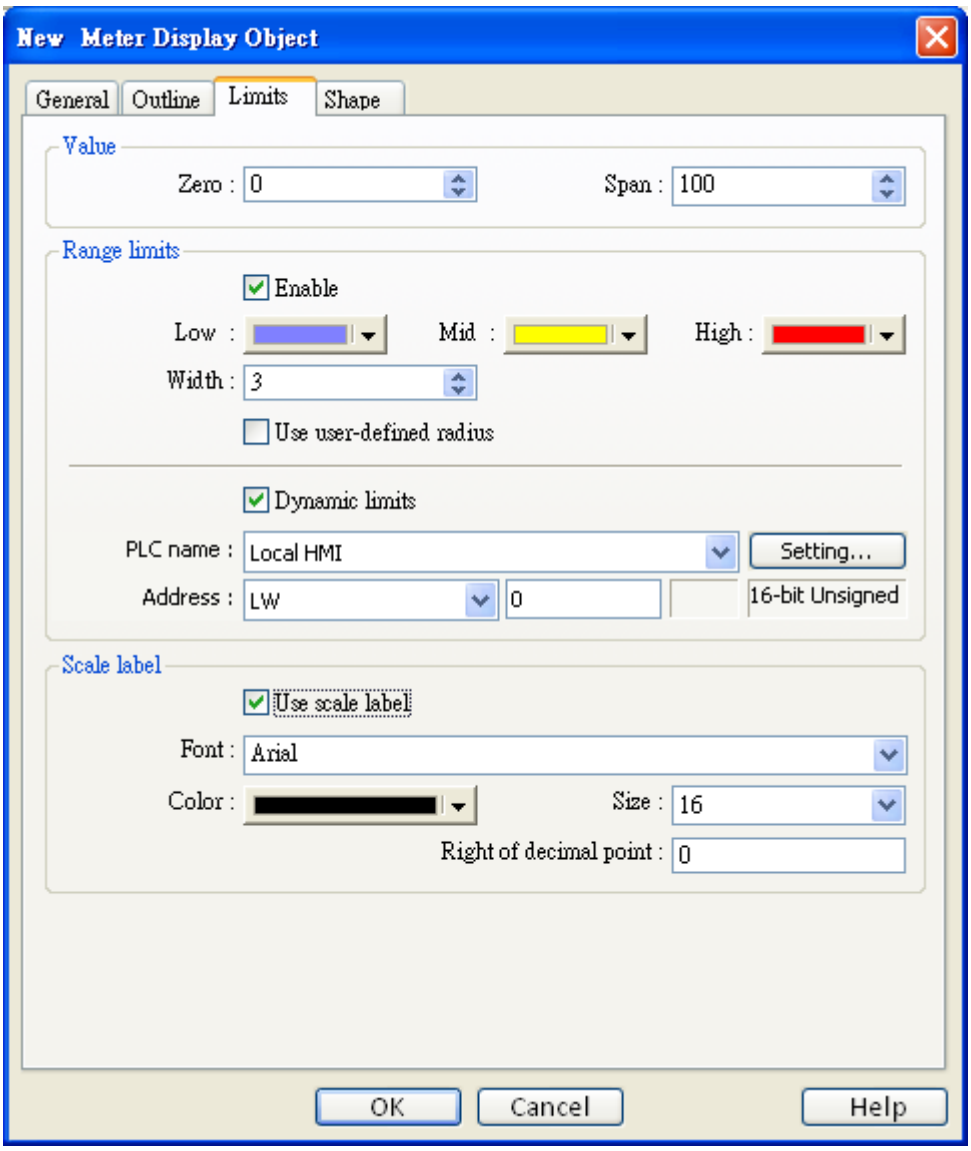
Pointer

Configure pointer's style, length, width, and color.

Pin point

Configure the style, radius, and color of the pin point.

The following picture shows the [Limit] tab.



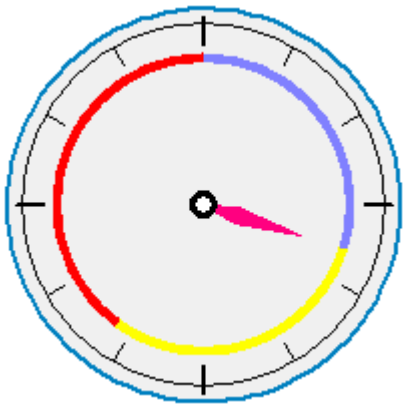
Value

To set object's display range. [Meter Display] object will use the value of [Zero] and [Span] and the value of register to calculate the pointer's position. For example, supposed that [Zero] is 0, [Span] is 100, when the value of register is 30, [Start degree] is 0, and [End degree] is 360, then the degree indicated by the pointer is:

{ (30 – [Zero]) / ([Span] – [Zero]) } * ([End degree] - [Start degree]) =

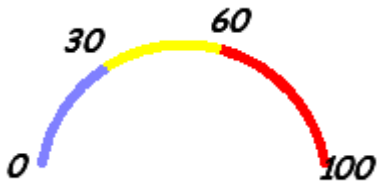
{(30 – 0) / (100 - 0)} * (360 – 0) = 108

Pointer will be pointing at 108 degrees. See the picture below.

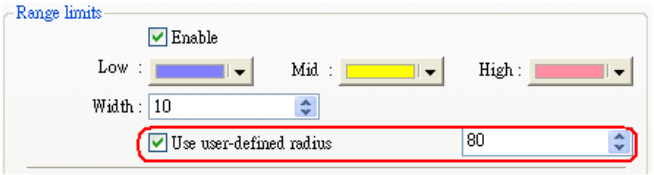


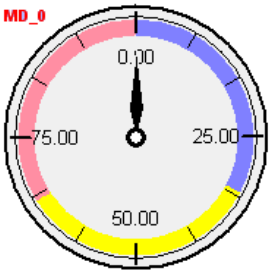
Range limits

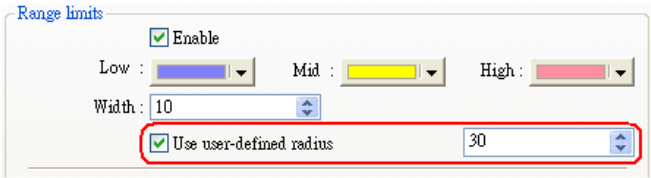
Configure the values of [Low limit], [High limit], their corresponding display colors, and the width.

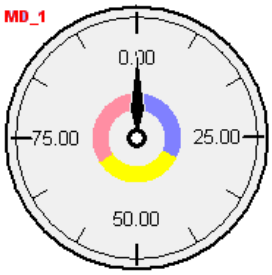


[Use user-defined radius]









[Dynamic Limits]

The low limit and high limit are decided by the register.

The following table shows the read address of low limit and high limit. When address is LW-n , the register's address:

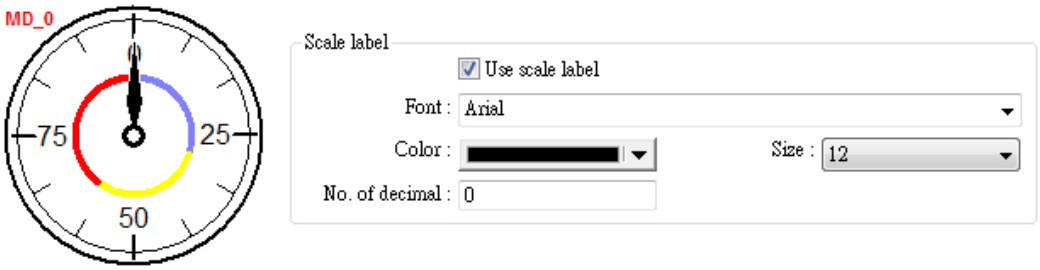
Data format	Low limit's read address	High limit's read address
16-bit format	LW-n	LW-n + 1
32-bit format	LW-n	LW-n + 2

For example, if the address is LW-100, the corresponding addresses will be:

Data format	Low limit's read address	High limit's read address
16-bit format	LW-100	LW-101
32-bit format	LW-100	LW-102

Scale label

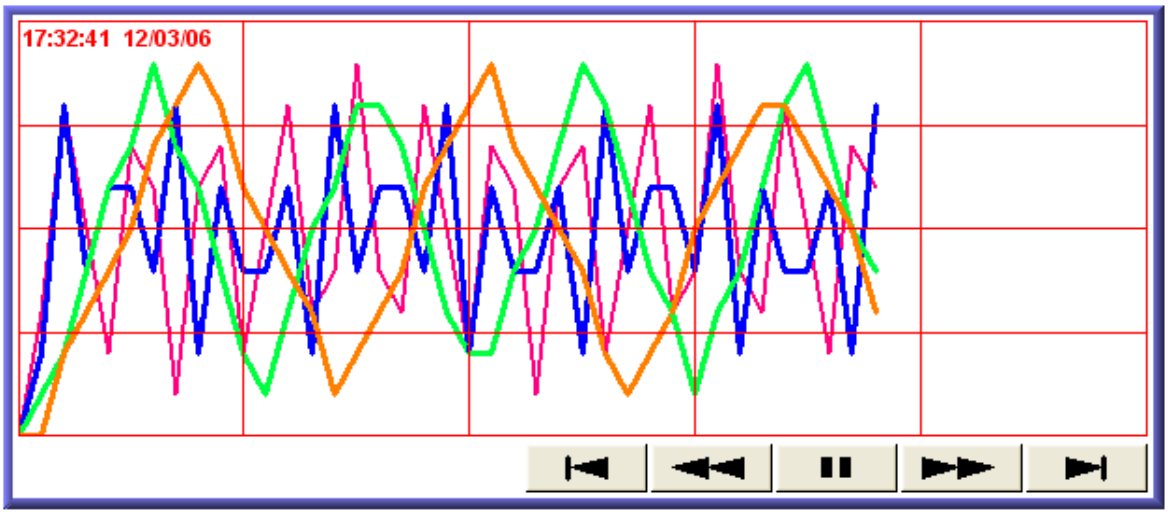
Select the attribute of scale label on [Meter Display].



13.17 Trend Display

Overview

[Trend display] object uses curves to represent the data recorded by [Data Sampling] object. The sampling operation is conducted by [Data Sampling] objects and the [Trend Display] object displays the result of sampling. The following picture shows an example of trend display object.



Those buttons on the screen mean:


Go to the beginning of the sampling data, and stop auto-scrolling.


Go to previous time interval and stop auto-scrolling.


Enable auto-scrolling. This shows when auto-scrolling is turned off.


Go to next time interval.

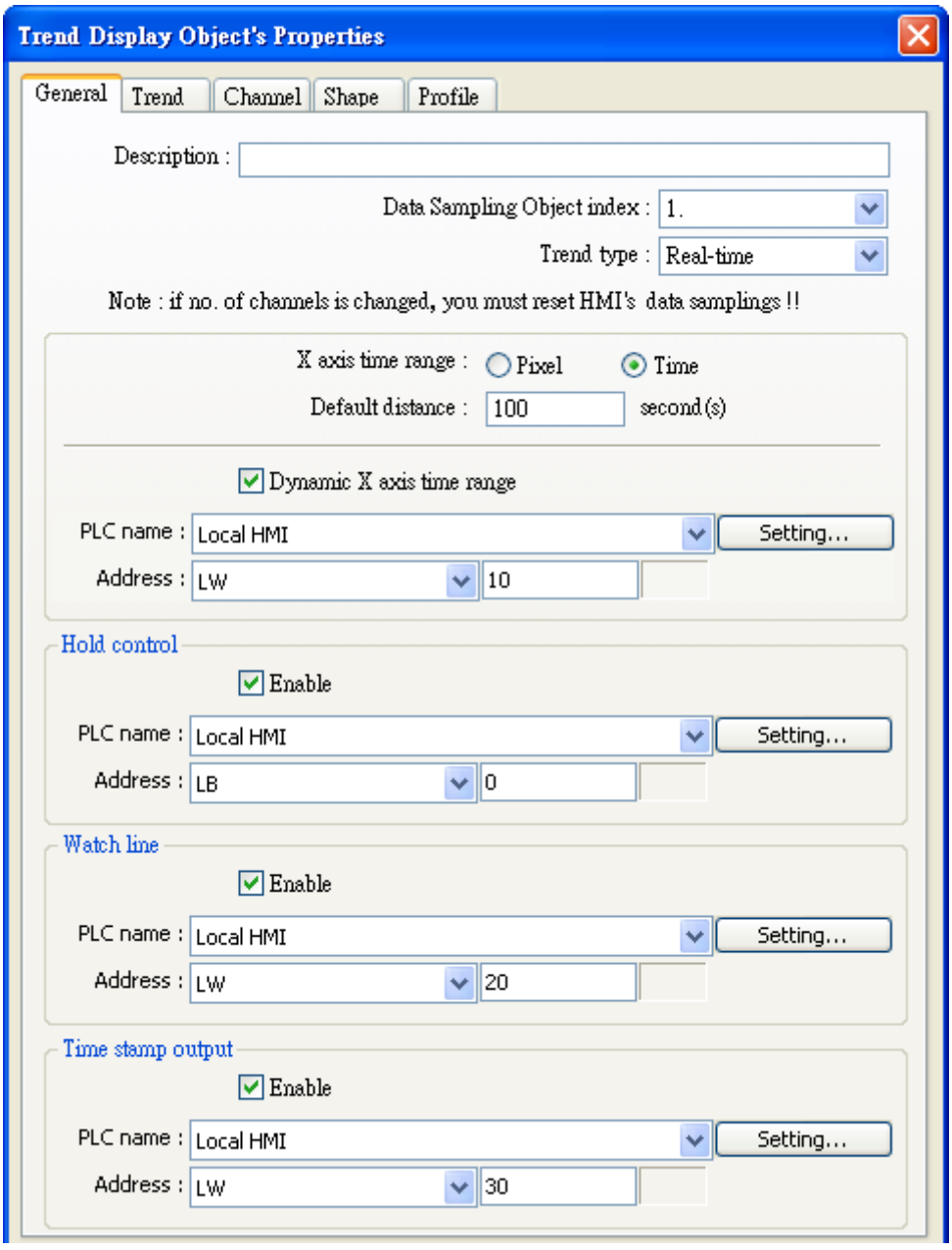

Go to the latest sampling data.


Click to stop auto-scrolling. This shows when auto-scrolling is turned on.

Configuration


Click [Trend Display] icon on the toolbar to open [Trend Display] dialog box. Fill in properties, click OK button, and then create a new [Trend Display] object.

The following picture shows the [General] tab in the [Trend Display] object properties dialog box.



[Data Sampling Object index]

Select a [Data Sampling] object as the source data.

[Trend mode]

Select the mode of data source, either [Real-time] or [History].

a. Real-time

In this mode, it displays the sampling data from the moment HMI starts to present. If other data are needed, select [History] mode to read the data from history data.

[Hold control]

Suspend the update of [Trend Display]. It does not stop the sampling process of [Data Sampling] object. The picture below shows the [Hold control] setting.

b. History

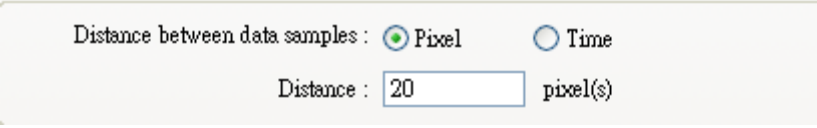
In this mode, the data come from the history data of the [Data Sampling] object defined by [Data Sampling Object index]. The sampling data is sorted by dates. The system uses [History control] to select the history data that are created in different dates.

The system sorts the history data of sampling data by date; the latest file is record 0 (Normally it is the sampling data today), the second latest file is record 1, and so on. If the value of designated register in “History control” is n, the [Trend Display] object will display data record n.

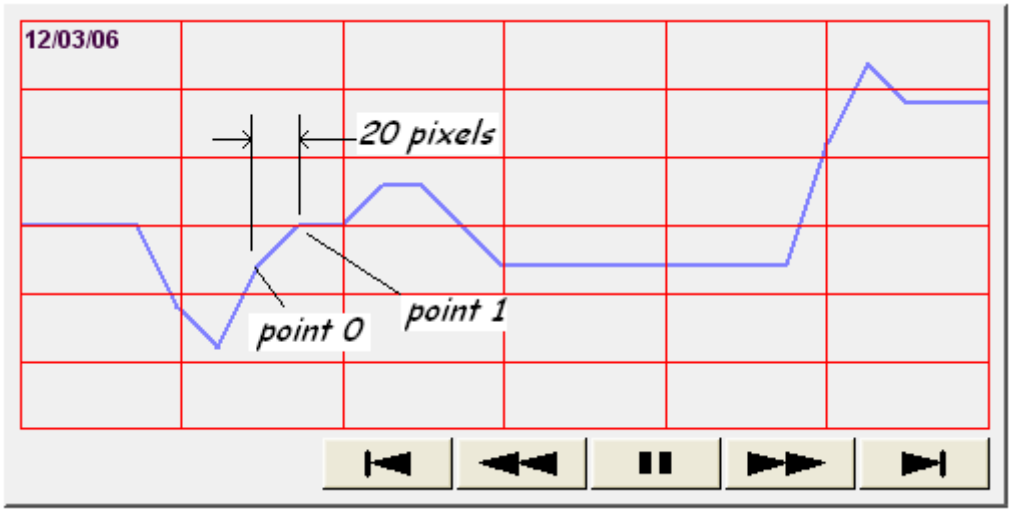
Here is an example to explain [History control]. If the designated register is LW-0, and the sampling data available in the files are pressure_20061120.dtl, pressure_20061123.dtl, pressure_20061127.dtl, and pressure_20061203.dtl, and it is 2006/12/3 today, based on the value of LW-0, the sampling data file which is selected by [Trend Display] is shown as follows:

Value of LW-0	Selected sampling history data
0	pressure_20061203.dtl
1	pressure_20061127.dtl
2	pressure_20061123.dtl
3	pressure_20061120.dtl

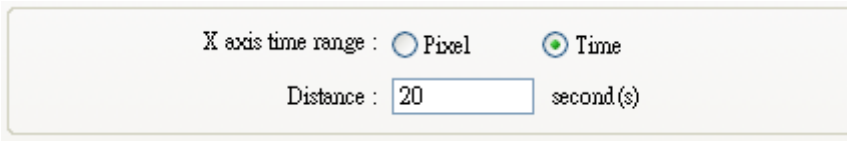
[Distance between data samples] ([Pixel] is selected)



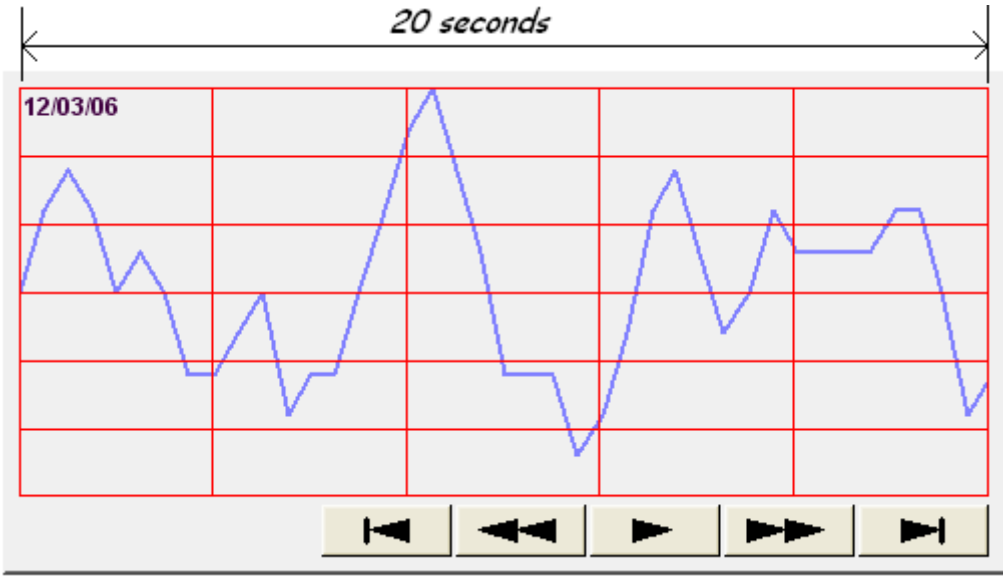
Select [Pixel]. Then, [Distance] can be used to set the distance between two sampling points. See the picture below.



[X axis time range] ([Time] is selected)



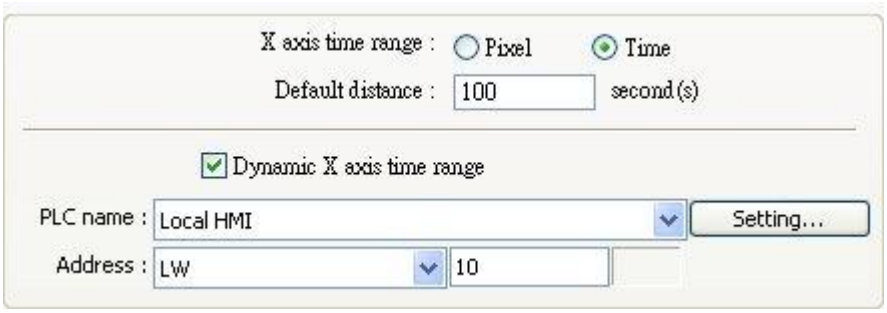
Select [Time]. Then, [Distance] is used to set the X-axis in unit of time elapsed. See the picture below.



Other than these two methods, select [Time] for [X axis time range] and go to [Trend] » [Grid] and enable [Time scale]. Please refer to [Time scale] in the later section.

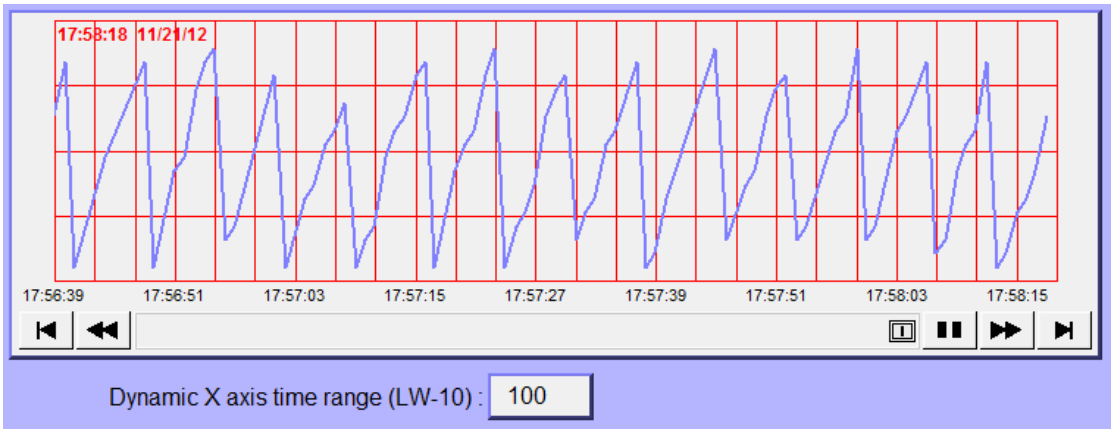
[Dynamic distance between data samples] / [Dynamic X axis time range]

If selected, a word register can be designated to adjust the distance between data samples (in pixel) or X axis time range (in second) directly on HMI.

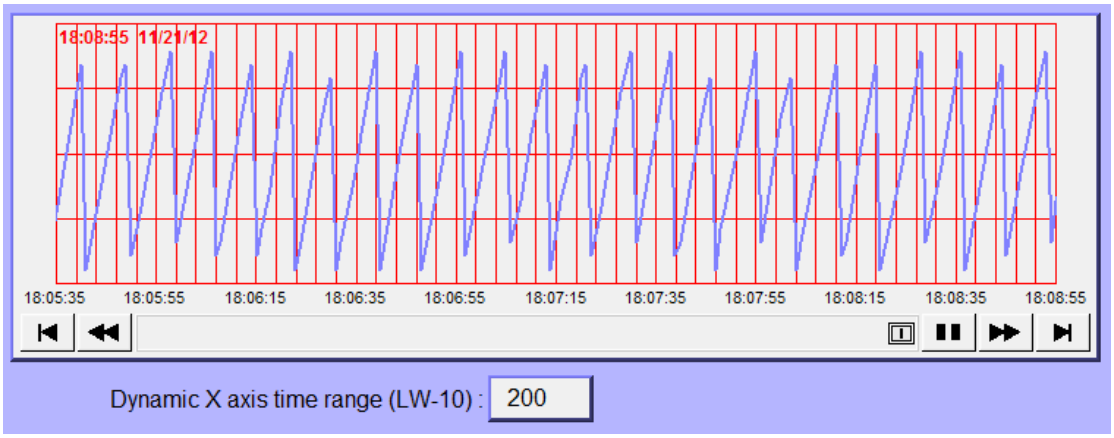


When this function is enabled, a valid default value must be set, that is, when the value of the designated register is 0, Trend Display will still be calculated according to the value set here.

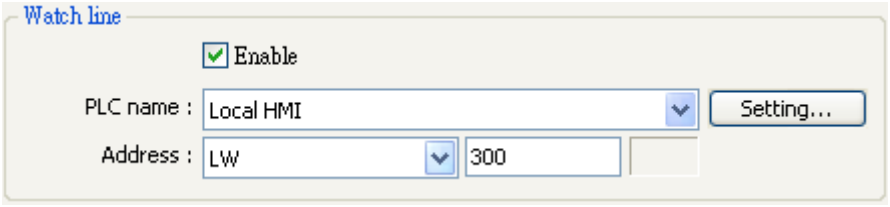
The following shows the trend curve displayed when X axis time range is set to 100.



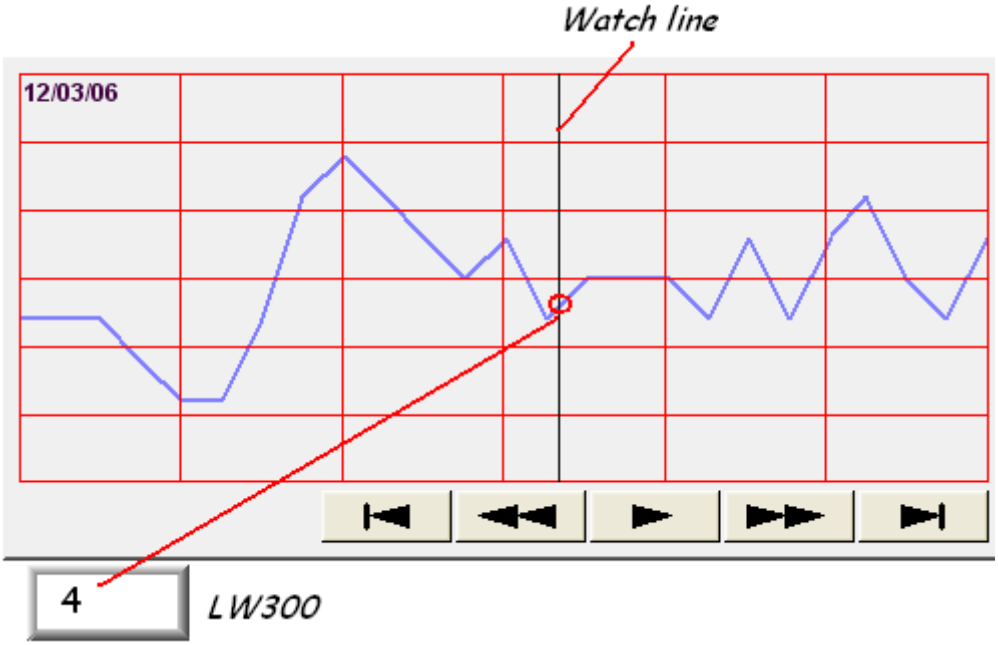
The following shows the trend curve displayed when X axis time range is set to 200.



Watch line



Use the [Watch line] function to display a “watch line” when user touches the [Trend Display] object. It will also export the sampling data at the position of watch line to the designated word device and use [Numeric Display] objects to display the results. Please refer to the following picture.



[Watch line] can also export sampling data with multiple channels. The system will consecutively write each channel to the specified address and the following addresses, in the same order in [Data Sampling] object. The address assigned to [Watch line] is the start address, and sampling data for each channel will be exported to the word devices starting from “start address.” If the data format of each channel is different, the corresponding address of each channel is arranged from the first to the last. If the watch register is LW-300:

[LW-300]	Ch. 0 : 16-bit Unsigned	(1 word)
[LW-301]	Ch. 1 : 32-bit Unsigned	(2 words)
[LW-303]	Ch. 2 : 32-bit Unsigned	(2 words)
[LW-305]	Ch. 3 : 16-bit Signed	(1 word)

Time stamp output

If selected, the system will start counting time from the first data sample, and output the elapsed time counted of the latest data sample to the register designated in [Time stamp output + 2].

When pressing a point on the trend curve, the relative time of the nearest data sample is then output to [Time stamp output address].

Note: the format of the register designated in [Time Stamp Output] and [Time Stamp Output + 2] must be 32-bit. [Time stamp output + 2] is only available for Trend Display - real time mode while [Time stamp output] is available for real time mode and history mode.

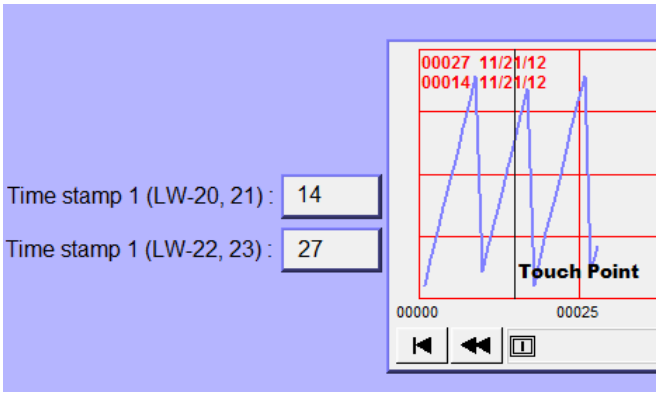
This function is only available when [Relative time mode] is selected.

When the designated register is 16-bit, the table below shows how the data of time stamp is stored in the register.

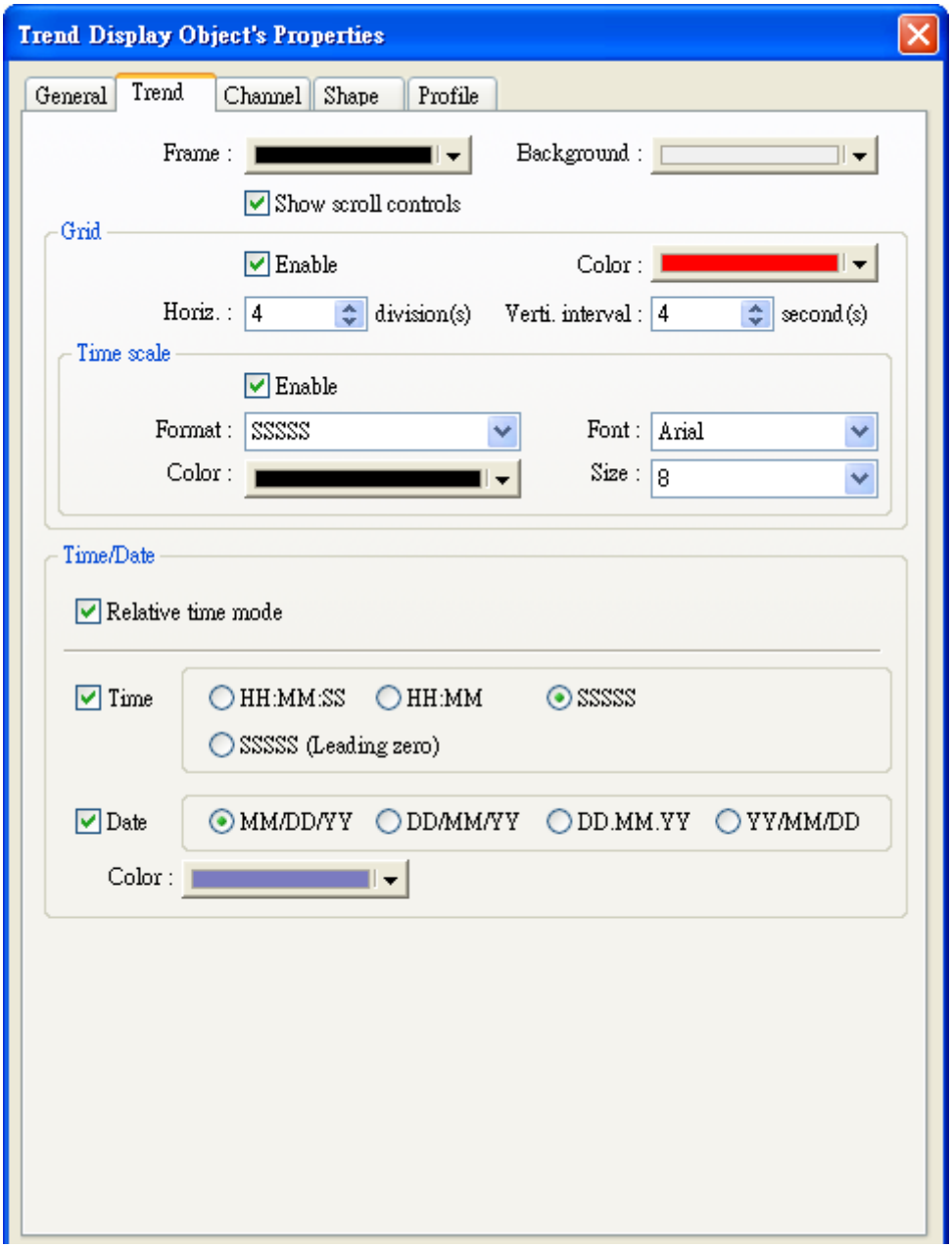
Address	The low word of the nearest sampling time to the touch point.
Address + 1	The high word of the nearest sampling time to the touch point.
Address + 2	The low word of the latest sampling time.
Address + 3	The high word of the latest sampling time.

The following demonstrates the operation when [Time stamp output] is enabled.

[LW-20, 21] The 14 (seconds) displayed in the register represents the nearest sampling time to the touch point.
[LW-22, 23] The 27 (seconds) displayed in the register represents the latest sampling time.



The picture below shows the attribute of [Trend Display]:



[Frame]

The color of frame.

[Background]

The color of background.

[Show scroll controls]

To enable / disable scroll control on the bottom of [Trend Display] object.

Grid

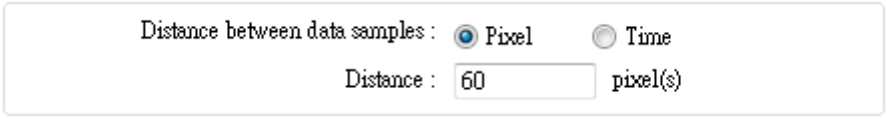
Set the distance and the color of grid.

[Horiz.]

The number of horizontal lines.

[Verti. interval]

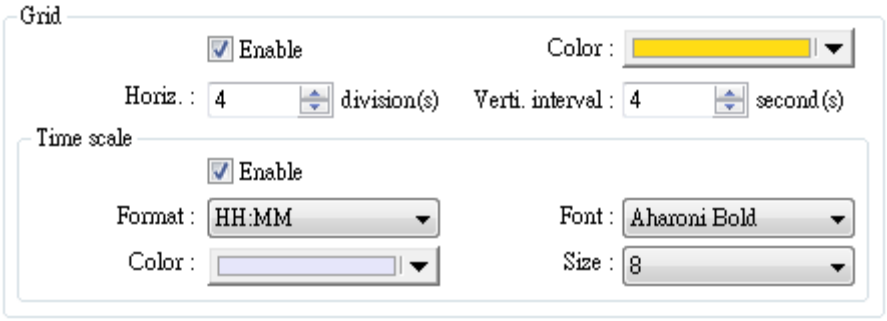
a. Pixel



When [Pixel] is selected in [General] tab, [Verti. interval] is used to select how many sampling point will be included between two vertical grid lines. See the picture below.

b. Time

When [Time] is selected, [Verti. interval] is used to select the time range between two vertical grid lines. See the picture below.



According to these settings, the system will calculate the number of vertical grid line automatically.

Time Scale

Select[Enable] to enable the time scale on the bottom of trend display.

[Format]

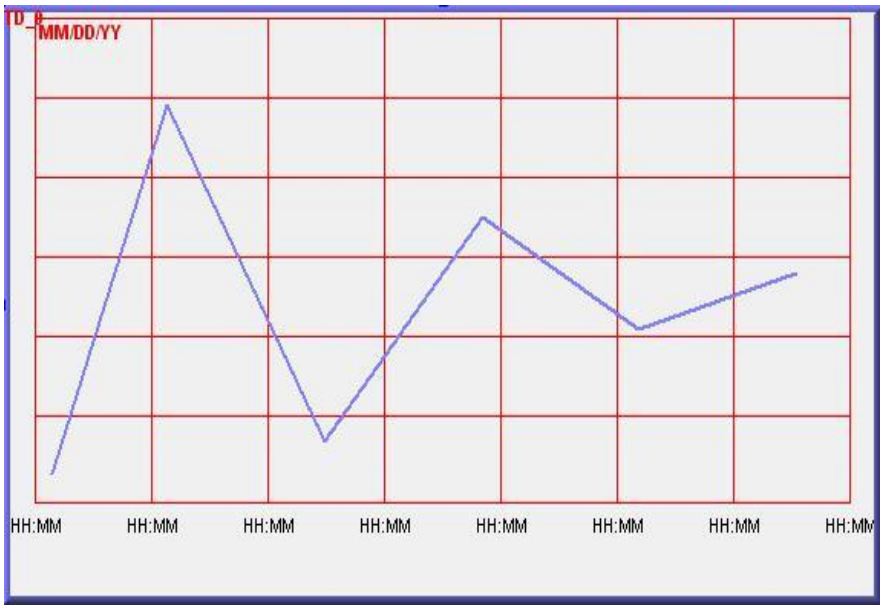
Selects the time scale displayed.

[Font]

Selects font style

[Size]

Selects font size. Default font size: 8.

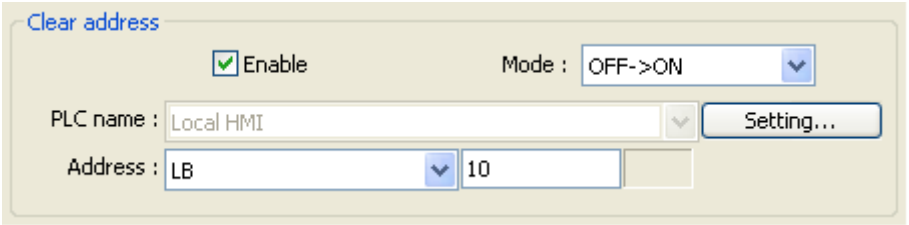


Relative time mode

If selected, the system will start counting time from the first data sample. The time displayed on the upper-left corner of the object and the range of X axis starts from “00:00:00”, “00:00”, “0”, or “00000”.

In addition, the time tag displayed on the upper-left corner of the object, and on the X axis can be set to [SSSSS] or [SSSSS (leading zero)] formats, and are based in seconds.

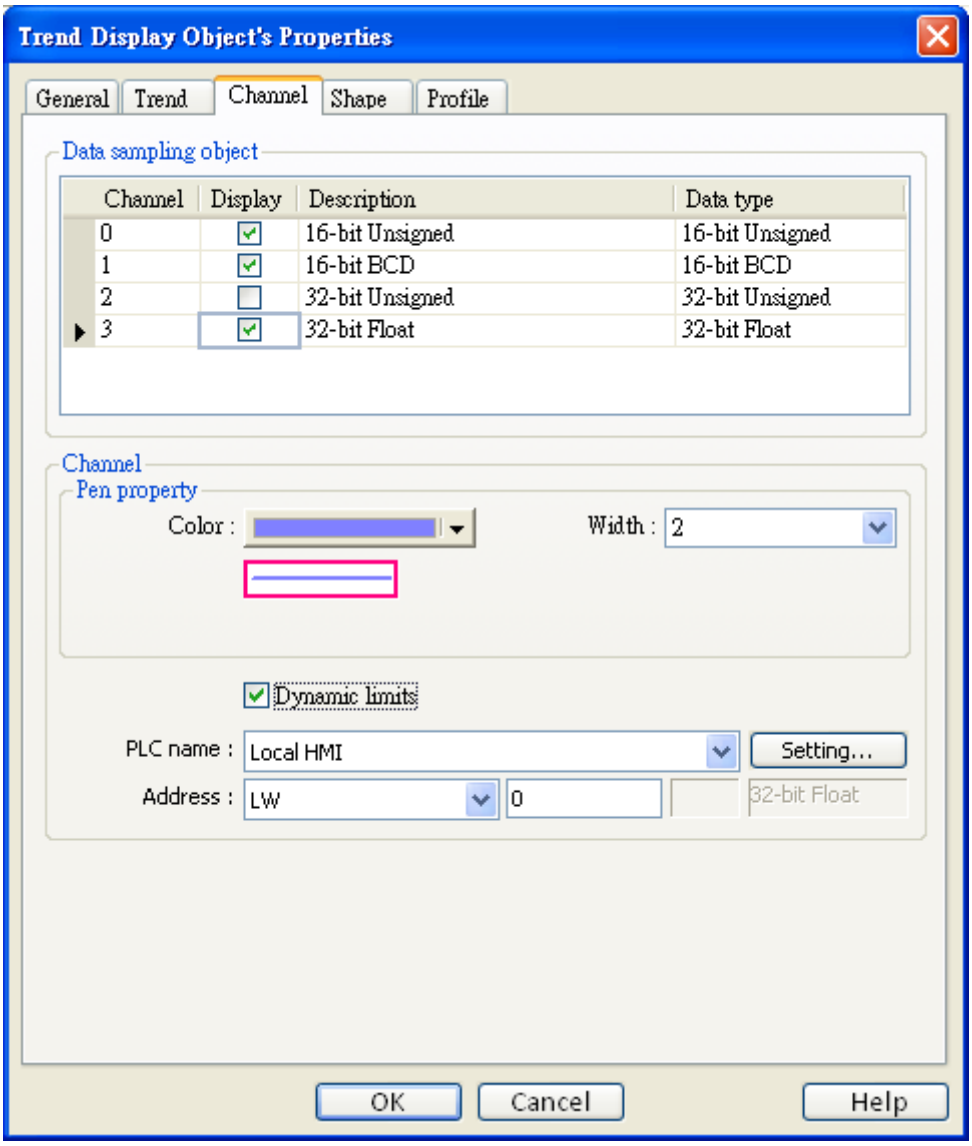
If the [Clear address] in [Data Sampling] object is enabled, the sampled data will be cleared, and the sampling time is reset to the start, that is, the time starts counting from the first data sample after reset. The figure below shows the settings of [Clear address].



Time / Date

The time of latest sampling data will be marked on the top left corner of the object. It is used to set the time display format and color.

The picture below shows the attribute of [Channel] tab:



[Channel]

Configure each sampling line's format and color, and the displayed data's low limit and high limit. At most 64 channels could be configured.

[Dynamic limits] unchecked

[Zero]、[Span]

[Zero] and [Span] are used to set the low limit and high limit of sampling data. If the low limit is 50 and the high limit is 100 for one sampling line, [Zero] and [Span] must be set as [50] and [100], so all the sampling data can be displayed in the trend display object.

[Dynamic limits] checked

The low limit and the high limit are read from the designated word devices, as shown below. When address is LW-n , the register's address:

Data Format	Low limit	High limit
16-bit format	LW-n	LW-n+1
32-bit format	LW-n	LW-n+2

For example, if LW-100 is used here, the low limit and the high limit will be read from:

Data Format	Low limit	High limit
16-bit format	LW-100	LW-101
32-bit format	LW-100	LW-102

A typical usage of this is used for zoom-in and zoom-out of [Trend Display].

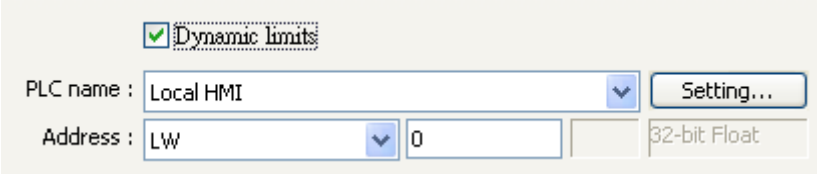
[Channel visibility control]

If [Enable] is selected, the bits of the assigned word register will be used to show/hide each channel. First bit controls first channel, second bit controls second channel, and so on. For example, there are 5 channels and LW-0 is used like the figure above, channels which will be shown are:

Channel	Controlling Bit	State	Will Be Displayed?
1	LW_bit-000	OFF	YES
2	LW_bit-001	ON	NO
3	LW_bit-002	ON	NO
4	LW_bit-003	OFF	YES
5	LW_bit-004	OFF	NO

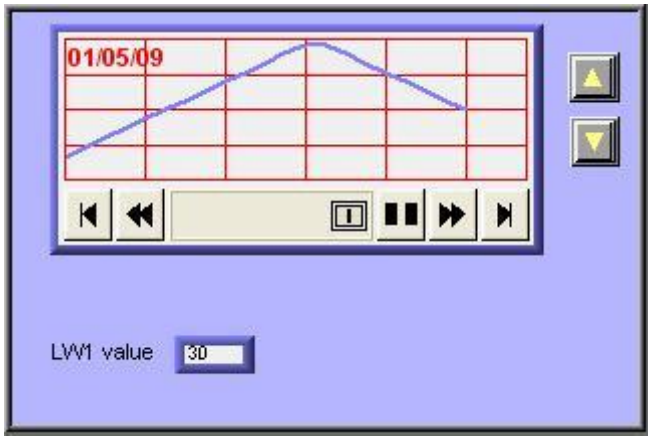
Example 1 Example of zoom in/out function

To zoom in / out the trend graph, check the **[Dynamic limits]** as picture below.

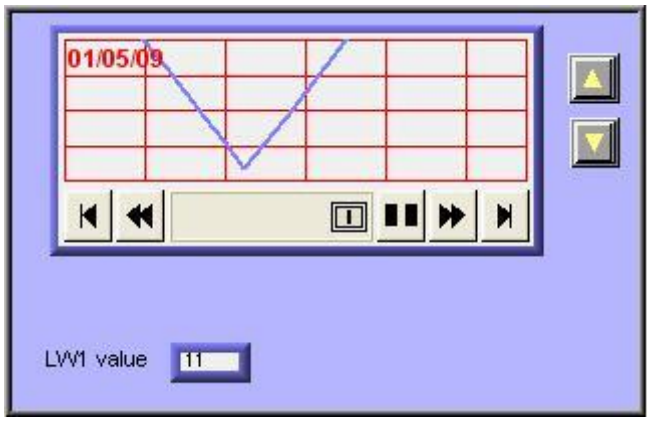


For example, the LW0 and LW1 are to control low limit and high limit, you may change the value of LW1 to zoom in / out.

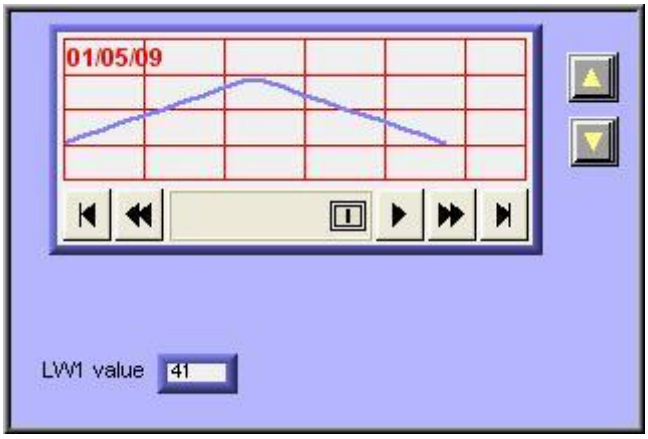
The following picture is in original size. The range of trend is between 0~30. The arrow on the right side are [Set Word] (LW1, increment (JOG+) and LW-1, decrement (JOG-)) for controlling the zoom in and zoom out function.



Decrease the value of LW-1 to exhibit zoom in function as shown below:
The value of LW-1 decreased to 11.



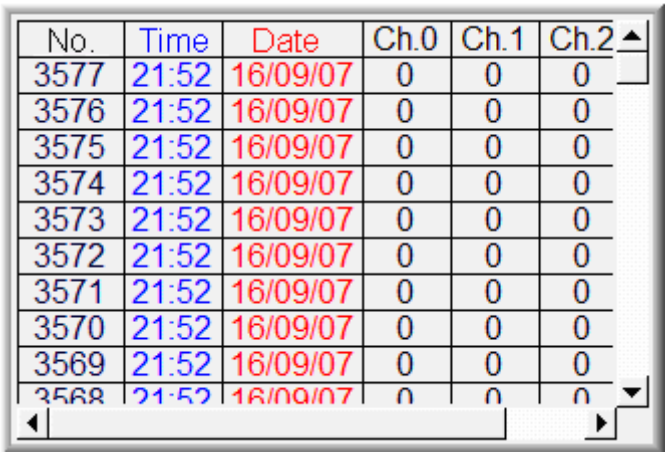
Increase the value of LW-1 to exhibit zoom out function as shown below:
The value of LW-1 increased to 41.



13.18 History Data Display

Overview

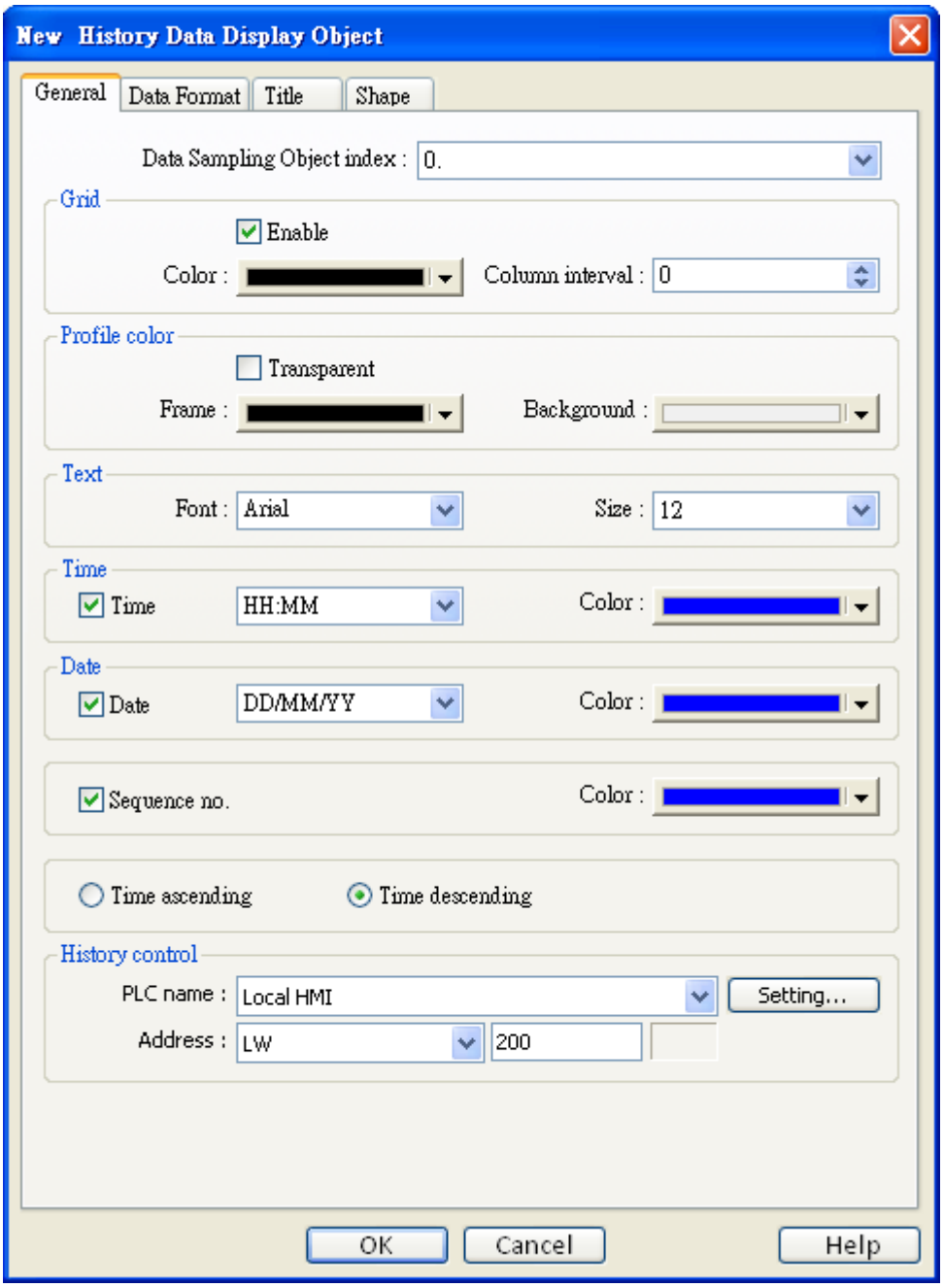
[History Data Display] object displays data stored by [Data Sampling] object. The difference from [Trend Display] is that [History Data Display] objects use tables to display numbers. Please note that the [History Data Display] will not refresh the table automatically when the data updates. It shows only the data retrieved from the designated record as the time window popped up.



No.	Time	Date	Ch.0	Ch.1	Ch.2
3577	21:52	16/09/07	0	0	0
3576	21:52	16/09/07	0	0	0
3575	21:52	16/09/07	0	0	0
3574	21:52	16/09/07	0	0	0
3573	21:52	16/09/07	0	0	0
3572	21:52	16/09/07	0	0	0
3571	21:52	16/09/07	0	0	0
3570	21:52	16/09/07	0	0	0
3569	21:52	16/09/07	0	0	0
3568	21:52	16/09/07	0	0	0

Configuration

Click the [History Data Display] icon on the toolbar to open [History Data Display] dialog box. Fill in properties, click OK button and then create a new [History Data Display] object.

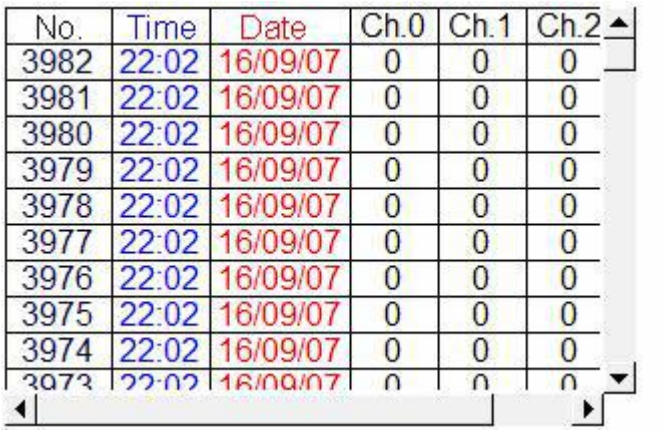


[Data Sampling object index]

Select a [Data Sampling] object as the source data.

Grid

Enable to show grids between rows and columns, like shown below:



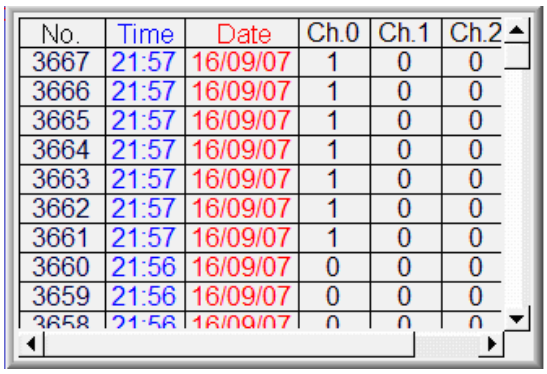
No.	Time	Date	Ch.0	Ch.1	Ch.2
3982	22:02	16/09/07	0	0	0
3981	22:02	16/09/07	0	0	0
3980	22:02	16/09/07	0	0	0
3979	22:02	16/09/07	0	0	0
3978	22:02	16/09/07	0	0	0
3977	22:02	16/09/07	0	0	0
3976	22:02	16/09/07	0	0	0
3975	22:02	16/09/07	0	0	0
3974	22:02	16/09/07	0	0	0

[Color]

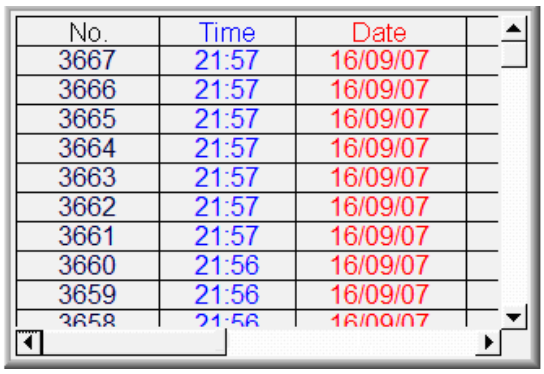
Change the color of grids.

[Column interval]

Change the width of each column. The figures below are the examples.



No.	Time	Date	Ch.0	Ch.1	Ch.2
3667	21:57	16/09/07	1	0	0
3666	21:57	16/09/07	1	0	0
3665	21:57	16/09/07	1	0	0
3664	21:57	16/09/07	1	0	0
3663	21:57	16/09/07	1	0	0
3662	21:57	16/09/07	1	0	0
3661	21:57	16/09/07	1	0	0
3660	21:56	16/09/07	0	0	0
3659	21:56	16/09/07	0	0	0



No.	Time	Date
3667	21:57	16/09/07
3666	21:57	16/09/07
3665	21:57	16/09/07
3664	21:57	16/09/07
3663	21:57	16/09/07
3662	21:57	16/09/07
3661	21:57	16/09/07
3660	21:56	16/09/07
3659	21:56	16/09/07

Profile color

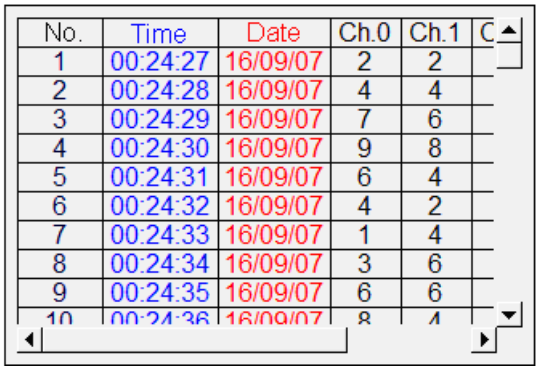
Change the color of frame and background. Use [Transparent] to hide frames and background.

[Time] and [Date]

Enable/disable showing the time and date and configure its format.

[Time ascending]

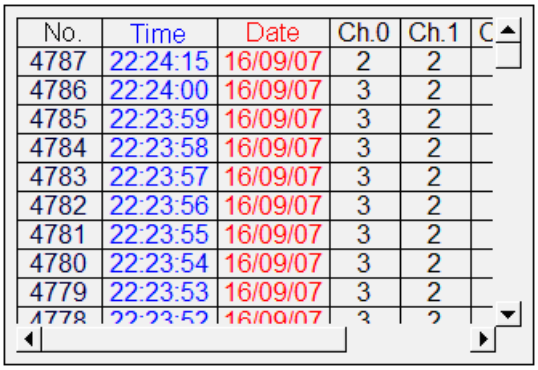
Put earlier data at the top and the latest data at the bottom.



No.	Time	Date	Ch.0	Ch.1	C
1	00:24:27	16/09/07	2	2	
2	00:24:28	16/09/07	4	4	
3	00:24:29	16/09/07	7	6	
4	00:24:30	16/09/07	9	8	
5	00:24:31	16/09/07	6	4	
6	00:24:32	16/09/07	4	2	
7	00:24:33	16/09/07	1	4	
8	00:24:34	16/09/07	3	6	
9	00:24:35	16/09/07	6	6	

[Time descending]

Put the latest data at the top and the earlier data at the bottom.

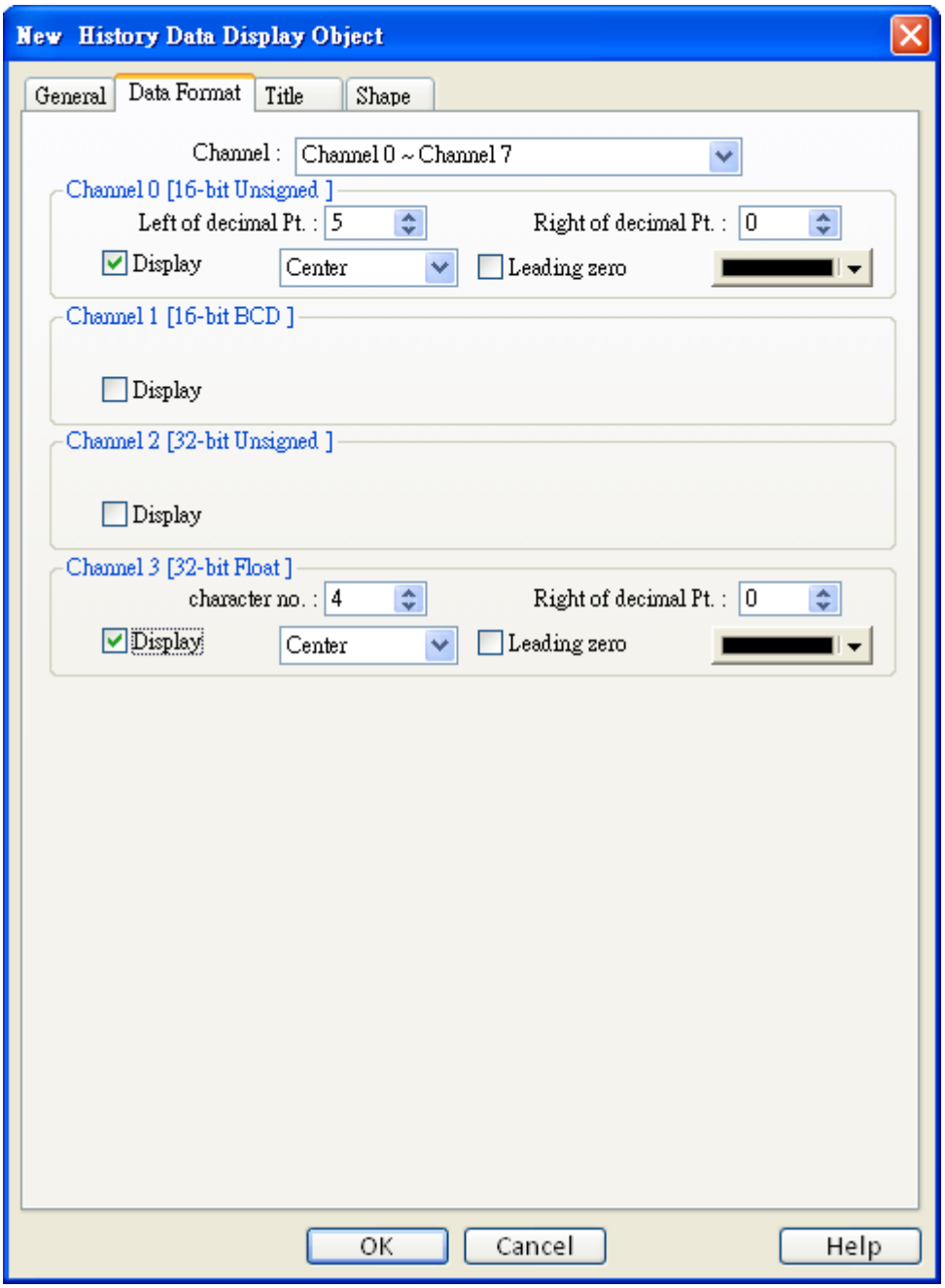


No.	Time	Date	Ch.0	Ch.1	C
4787	22:24:15	16/09/07	2	2	
4786	22:24:00	16/09/07	3	2	
4785	22:23:59	16/09/07	3	2	
4784	22:23:58	16/09/07	3	2	
4783	22:23:57	16/09/07	3	2	
4782	22:23:56	16/09/07	3	2	
4781	22:23:55	16/09/07	3	2	
4780	22:23:54	16/09/07	3	2	
4779	22:23:53	16/09/07	3	2	
4778	22:23:52	16/09/07	3	2	

History Control

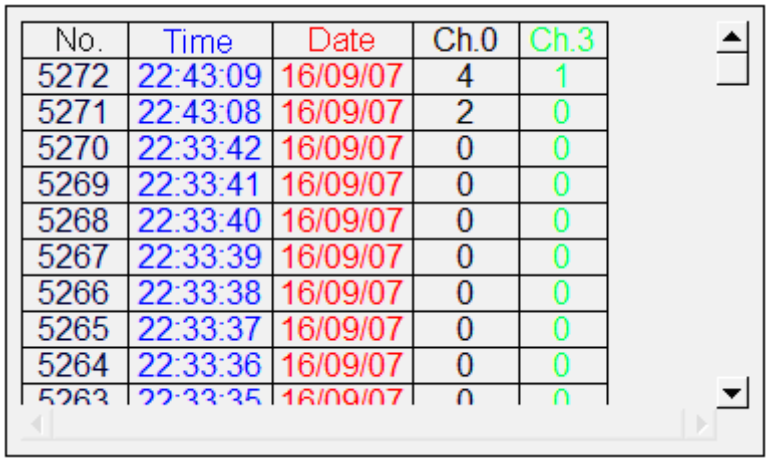
The history files are sorted with date and given an index. The latest one is assigned index 0 (mostly today), the second latest file is assigned index 1, and so on. [History Control] is used to select the history data to be shown.

The picture below shows the attribute of [Data Format] tab:



Each [History Data Display] object can display up to 64 channels. Use [Display] to select the channels to be shown on the screen.

In the figure above, there are 4 channels (channel 0 to channel 3) in the [Data Sampling] object, and only Ch.0 and Ch.3 are selected. The data formats are shown next to channel name. The data format of each channel is decided by the corresponding [Data Sampling] objects. The result is shown below:

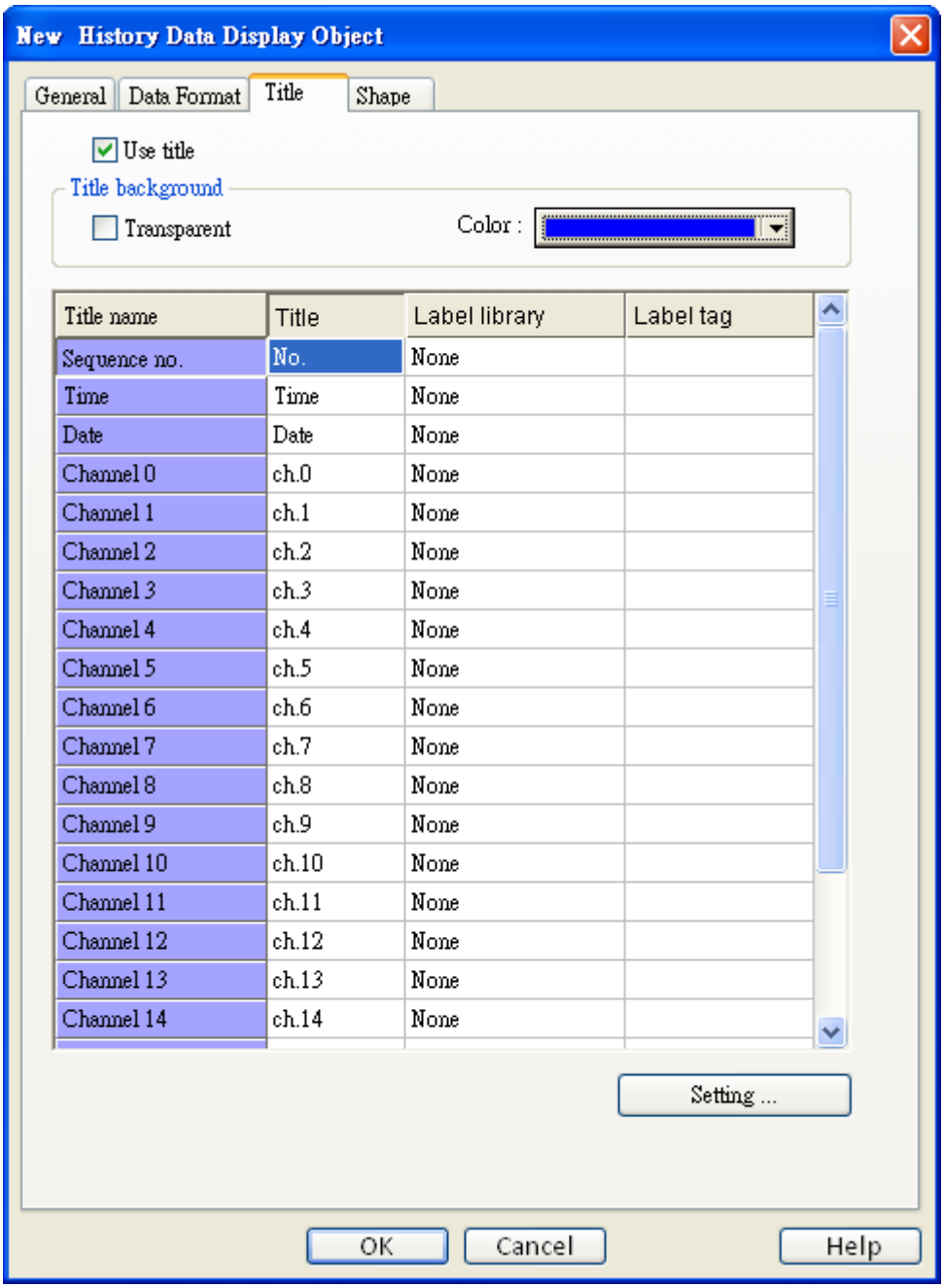


No.	Time	Date	Ch.0	Ch.3
5272	22:43:09	16/09/07	4	1
5271	22:43:08	16/09/07	2	0
5270	22:33:42	16/09/07	0	0
5269	22:33:41	16/09/07	0	0
5268	22:33:40	16/09/07	0	0
5267	22:33:39	16/09/07	0	0
5266	22:33:38	16/09/07	0	0
5265	22:33:37	16/09/07	0	0
5264	22:33:36	16/09/07	0	0
5263	22:33:35	16/09/07	0	0

When display [String] format in [History Data Display] object, users may choose:

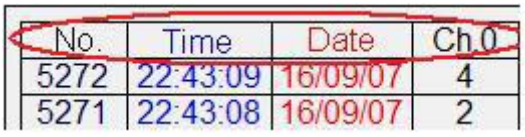
a. Display in [UNICODE] mode
b. Reverse high byte and low byte data then display.

The picture below shows the attribute of [Title] tab:



[Use title]

Enable or disable title, which is marked as shown below:



Title background

[Transparent]

When selected, hide the background for title area.

[Background color]

Set the background color of title.

[Setting]

This dialog window defines the text to be shown on the title.

No.	Time	Date	Ch.0
5272	22:43:09	16/09/07	4
5271	22:43:08	16/09/07	2

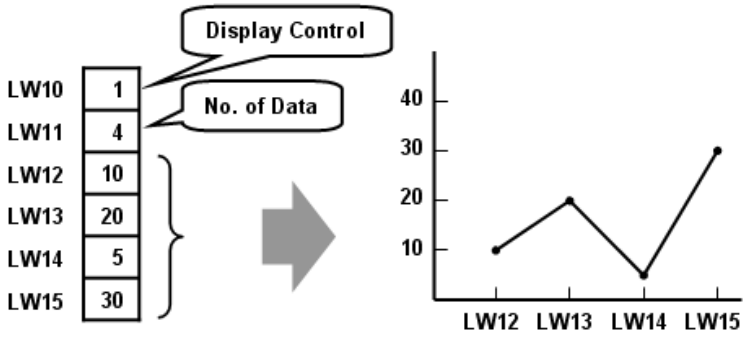
You can also use [Label Tag Library] to use multi-language text for titles. Click [Setting] and select one label tag from label library, as shown below:

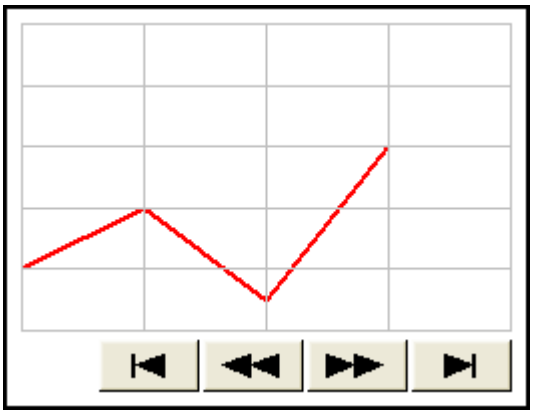
If the format of sampling data is changed during off-line simulation, please delete previous data records in C:\EasyBuilder \HMI_memory\datalog to prevent the system from misinterpreting the old data records.

13.19 Data Block Display

Overview

[Data Block Display] is a combination of several word devices with continuous address, for example, LW-12, LW-13, LW-14, LW-15, and so on. Use [Data Block Display] object to draw multiple data blocks. For example, it can display two data blocks LW-12~LW-15 and RW-12~RW-15 in trend curve simultaneously. It is very useful to observe and compare the difference of trend curves. The following displays the data block of LW-12~LW-15.

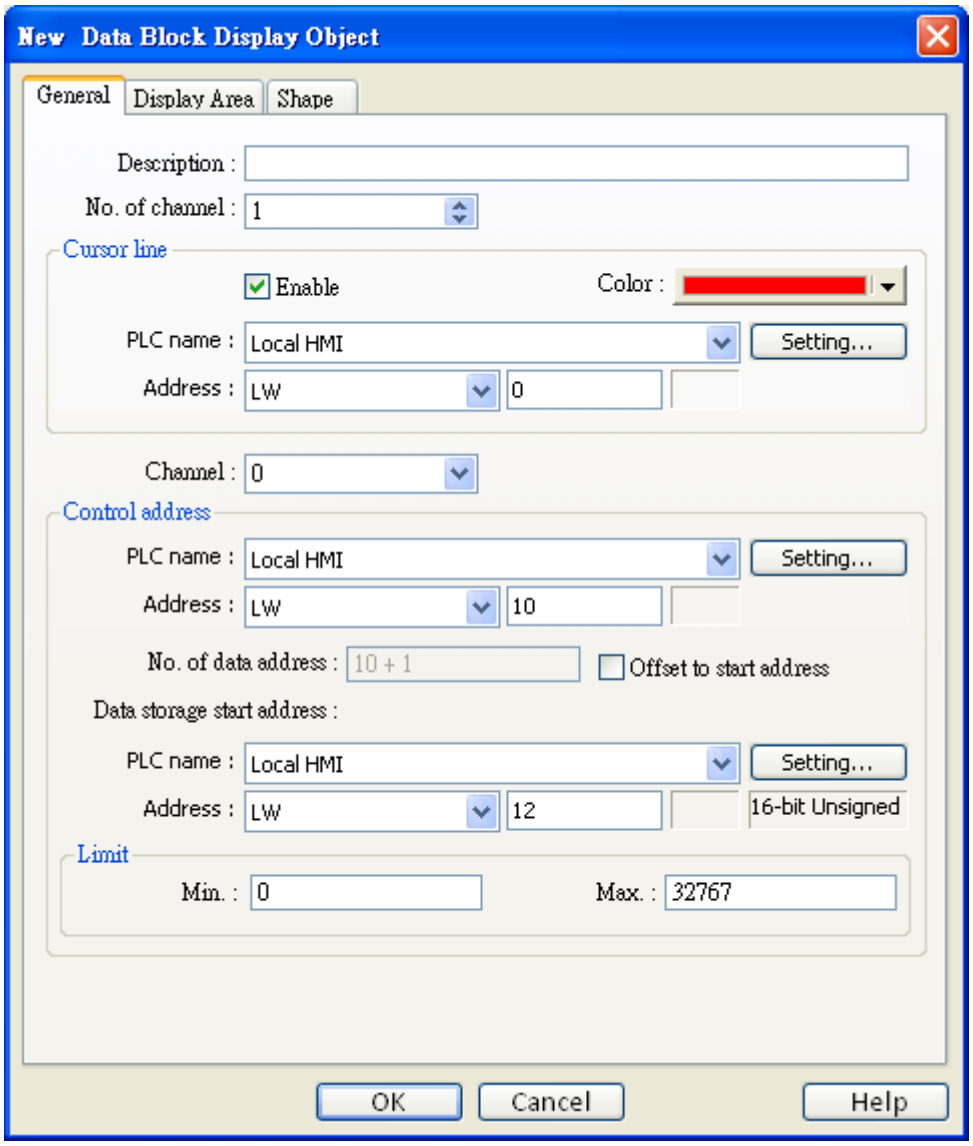




Snapshot of Data Block Display

Configuration

Click [Data Block Display] icon on the toolbar to open [Data Block Display] properties dialog box as follows:



[No. of channel]

Set the no of channel for this object. Each channel represents one data block. The maximal number of channels is 12.

Cursor Line

If [Cursor Line] is enabled, when user touches the [Data Block Display] object, it will display a vertical cursor line on it, and store the data on the line to the designated registers.

[Channel]

Select the channel to be configured.

Control address

[PLC name]

Select the PLC where the target data block is located.

[Device type]

Select the device type where the target data block is located.

[Control word address]

[Control word address] is used to control and clear the drawn curve.

0 = No action (default)

1 = Draw (Without clear first)

2 = Clear

3 = Redraw

After executing the operation above, the system will reset the control word to zero.

[No. of data address]

The default for [No. of data address] is [Control word address] + 1.

[No. of data address] stores the number of word devices in each data block, i.e. the number of data. The maximum value is 1024.

[Data storage start address]

If [Offset to start address] is not selected, please select the start address which stores the data.

[Offset value storage address]

If [Offset to start address] is enabled, the [Offset value storage address] will be set as [Control word address] + 2.

[Format]

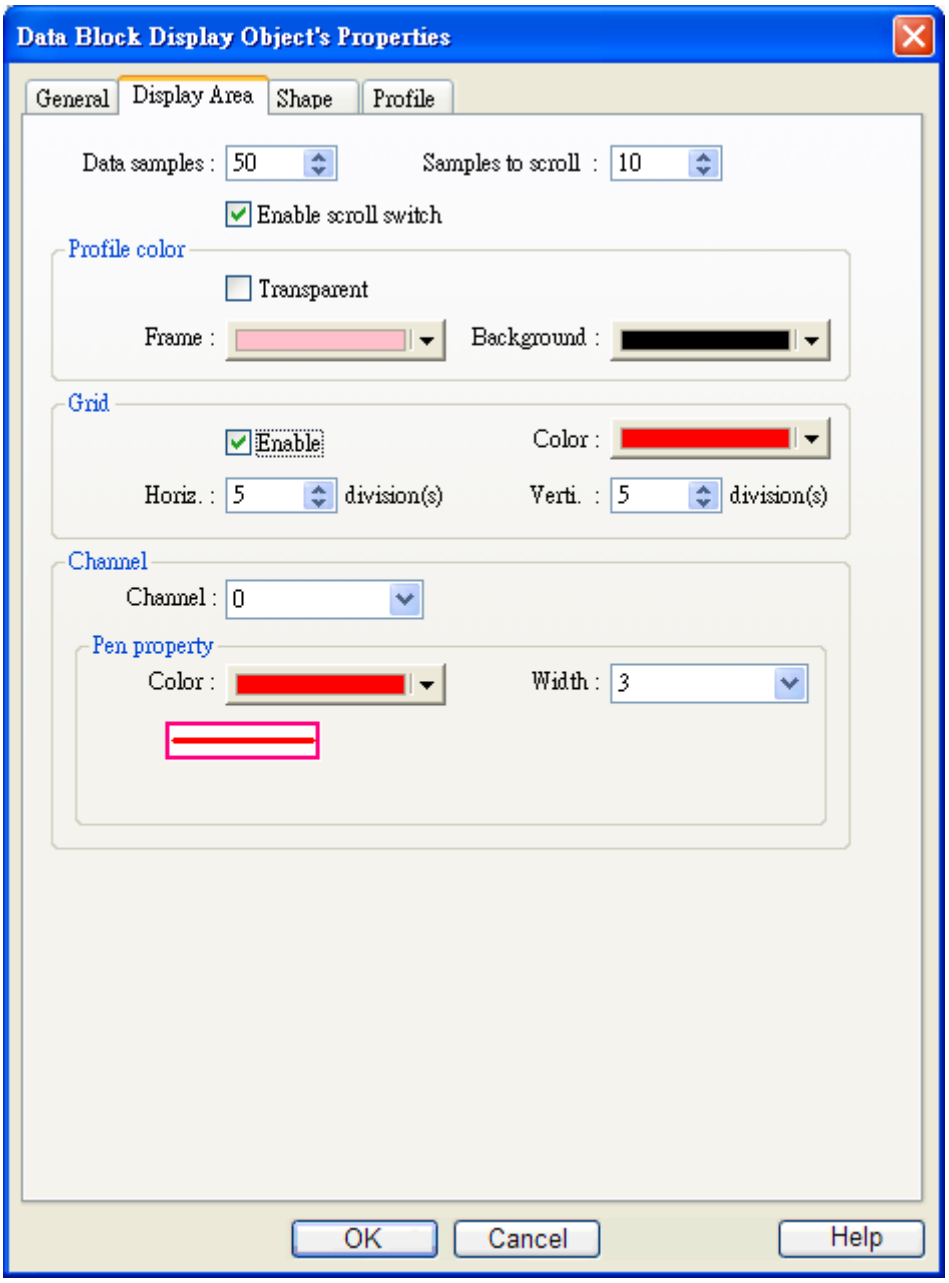
If you select 16-bit data format, the address for each data will be start address, start address + 1, start address + 2 and so on.

If you select 32-bit data format, the address for each data will be start address, start address + 2, start address + 4 and so on.

Limit

Set the minimum and maximum limit for the curve.

The figure below show the settings in the [Display Area] tab:

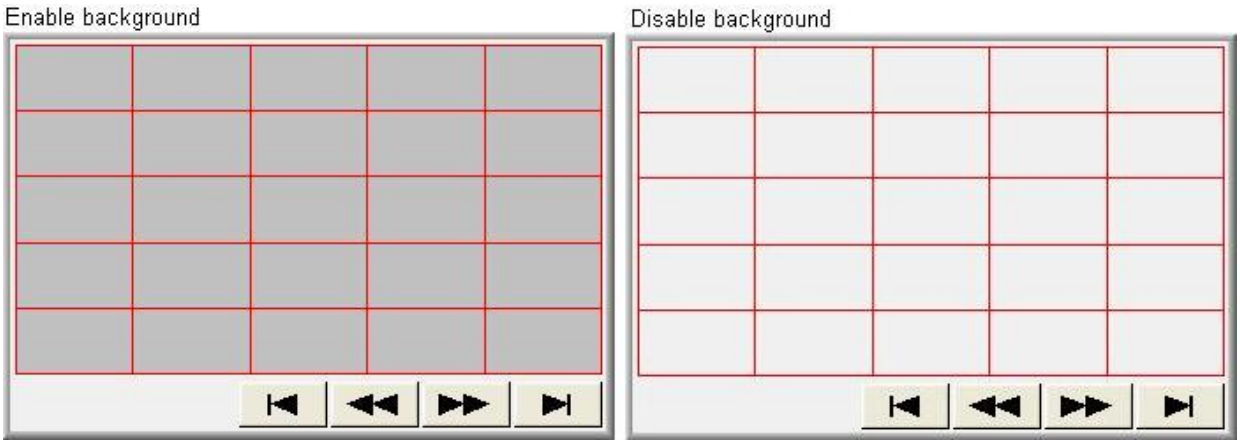


[Data samples]

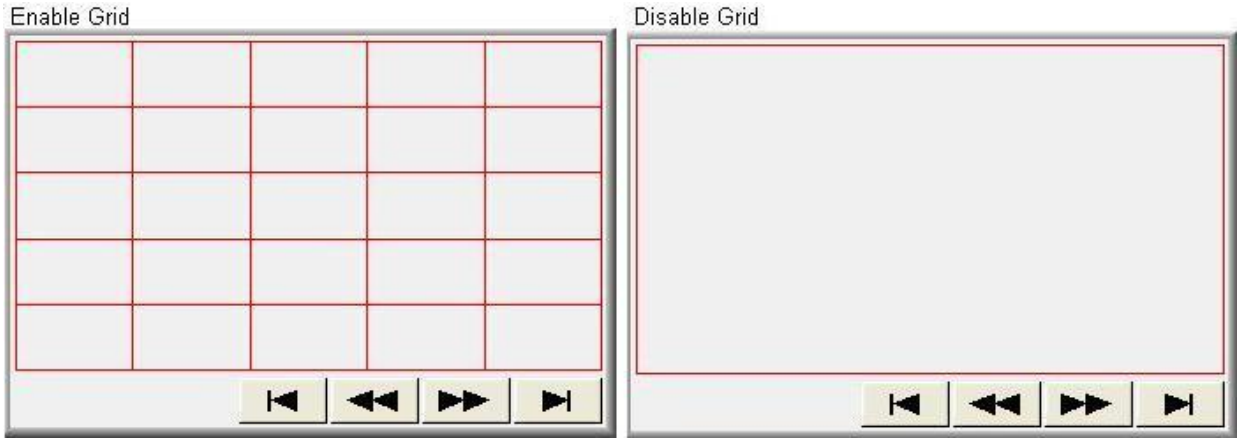
Configure the maximal number of data samples (points) to be displayed.

[Samples to scroll]

Configure the number of data samples being scrolled.

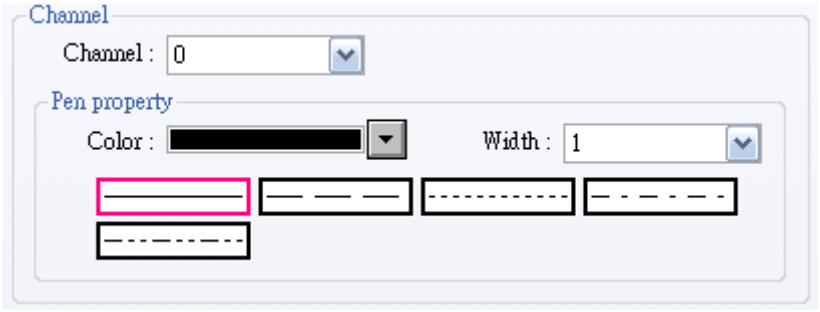


Grid



Channel

Set the color, width and style of each curve.



Examples:

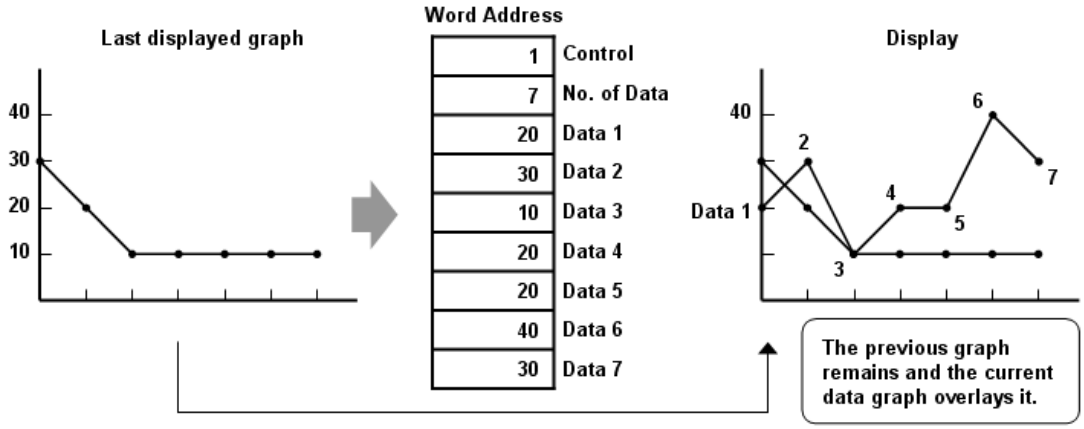
1. How to show a data block

a. Write the number of data to [No. of data address], i.e. “control word address+1”

b. Store the data consecutively beginning at [Data storage start address].

c. Write “1” to [Control word address] to draw the curve without cleaning the plot. All previous curves will not be erased.

d. The system will write “0” to [Control word address] after it is plotted.

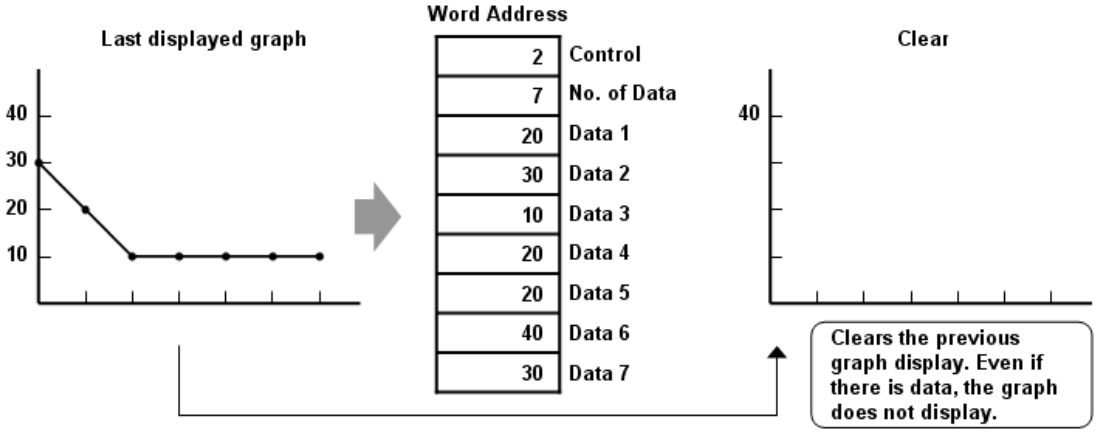


- During the period between c and d, do not change the content of [Control word address], [No. of data address] and [Data storage start address], it might cause error for trend curve plot.

2. How to clear the graph

a. Write “2” to [Control word address], all the trend curves will be cleared.

b. The system will write “0” to [Control word address] after the trend curve is cleared.



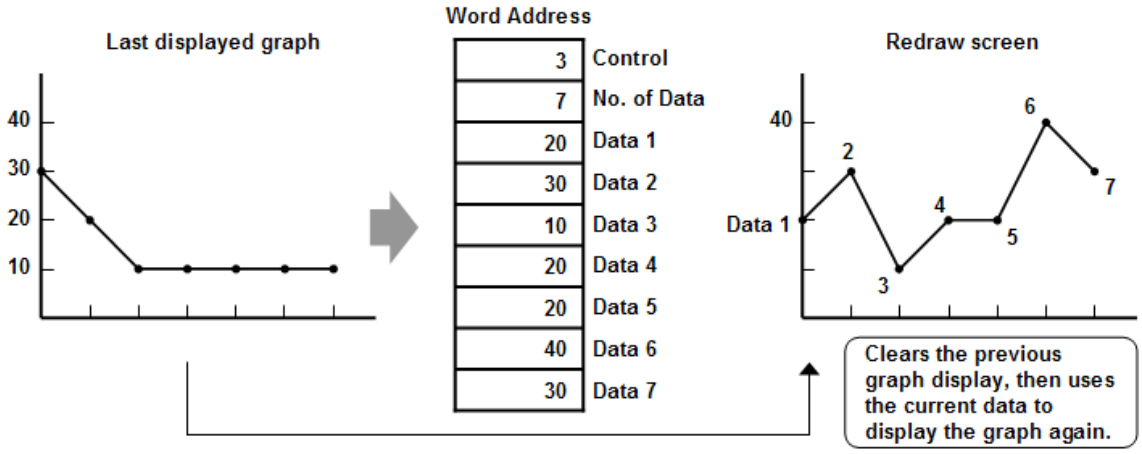
3. How to clear the previous trend curve and display new one

a. Write the number of data to [No. of data address], i.e. “control word address+1”

b. Store the data consecutively beginning at [Data storage start address].

c. Write “3” to [Control word address], the previous trend curves will be cleared and the new content in data block will be plotted on the screen.

d. The system will write “0” to [Control word address] after the trend curve displayed.

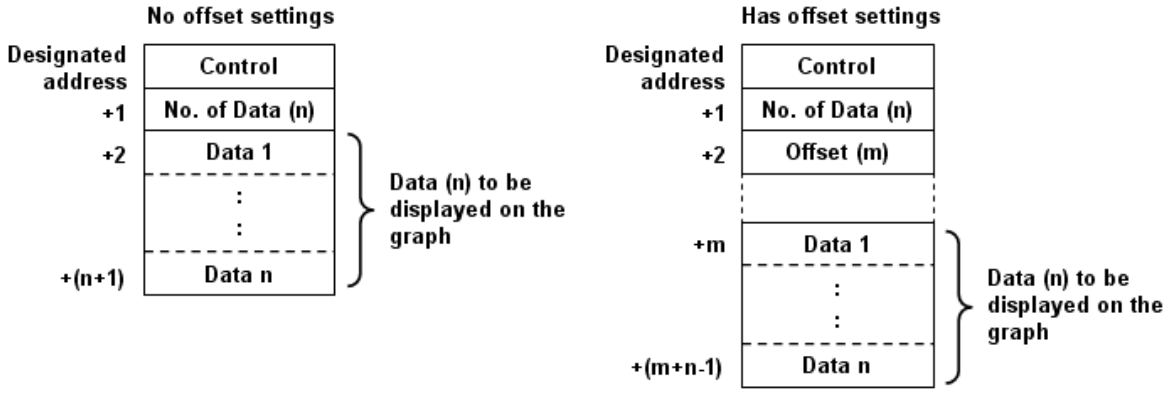


4. How to use offset mode

If [Offset to start address] is selected, [Control word address], [No. of data address], and [Offset value storage address] will use 3 consecutive addresses.

For example, assume the total number of channels is 3 (start from 0 to 2), and 3 [Control word address] are LW-0, LW-100, and LW-200. Then, other addresses are set as follows: (In the example, format 16-bit Unsigned is used and [Offset value storage address] are all m)

Item	Control Address	No. of data address	Offset value storage address	Data 1	Data 2	…
Channel 0	LW-0	LW-1	LW-2 (=m)	LW-0+m	LW-1+m	…
Channel 1	LW-100	LW-101	LW-102 (=m)	LW-100+m	LW-101+m	…
Channel 2	LW-200	LW-201	LW-202 (=m)	LW-200+m	LW-201+m	…



- When [Control word address] is set to LW-n, [No. of data address] and [Offset value storage address] are as follows:

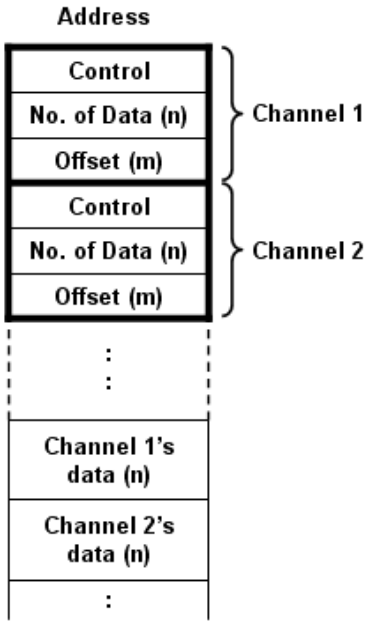
Address	16 Bit	32 Bit
Control word address	LW-n	LW-n
No. of data address	LW-n+1	LW-n+2
Offset value storage address	LW-n+2	LW-n+4

- If the control registers are 32-bit devices, only bit 0-15 will be used as control purpose, bit 16-31 will be ignored. (as illustrated below)

32 bit device

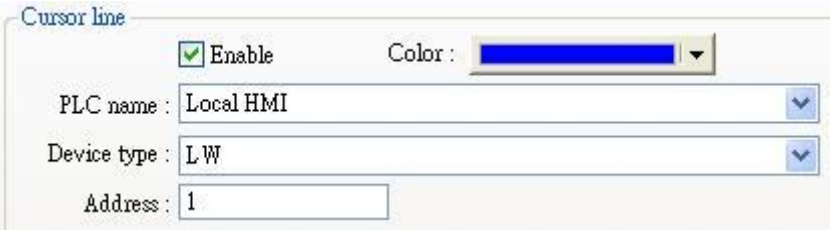
	31 ... 16	15 ... 0
+0	0	Control
+1	0	No. of Data
+2	0	Offset

- When the value in [Control word address] is not zero, the system will read [No. on data address] and [Offset value storage address].
- It is recommended to use [Offset to start address] for data block display with multiple channels and the same device type. You can use [Control word address] at continuous address for each channel. The system will read the control words of all the channels in one read command and it shall speed up the response time. Please refer to the following picture. The control words of channel 1 is located from LW-n, the control words of channel 2 is located from LW-n+3, and so on. As they are all continuous addresses, the system could read all the control words in one read command.

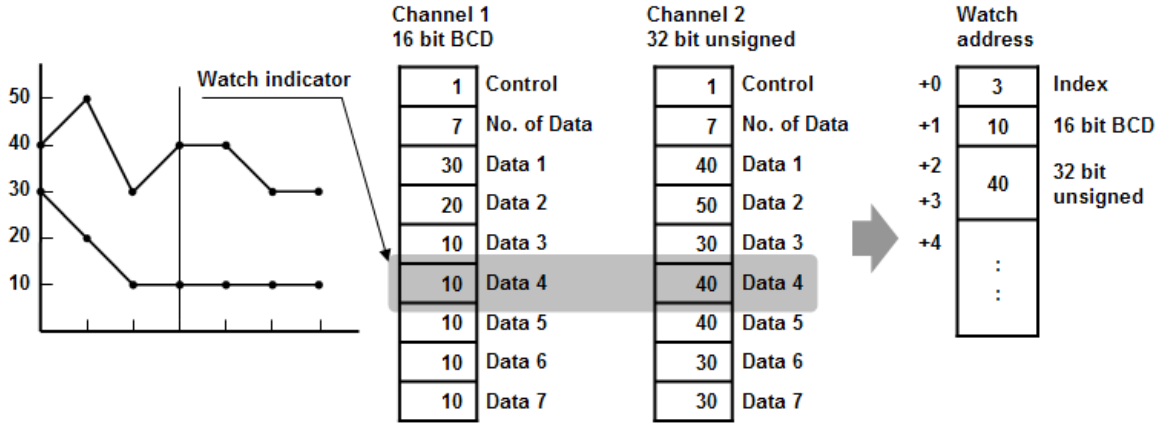


How to use watch (Cursor Line) feature

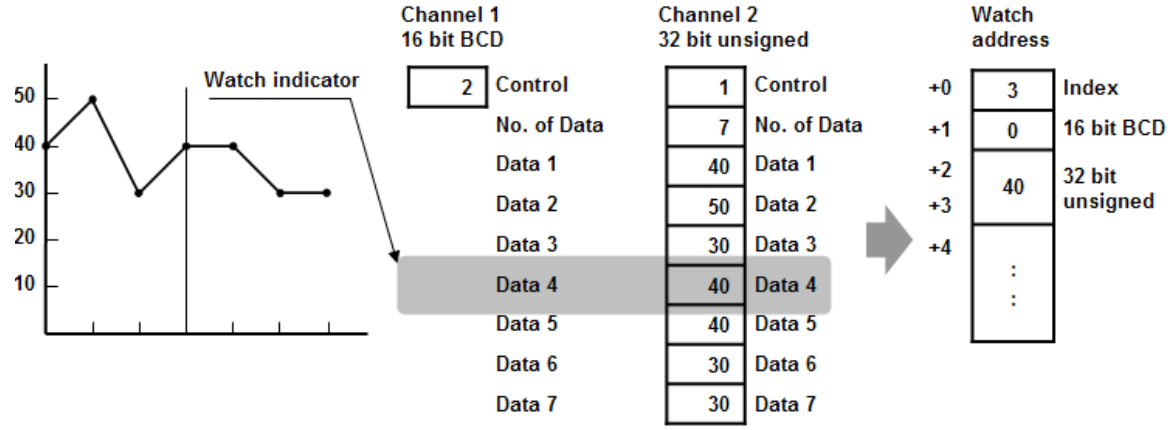
You may use the “Watch” function to check the value of any point of the curve. When the user touches [Data Block] object, it will display a “cursor line”, and the system will write the index and value of that data on the cursor line to the designated address.



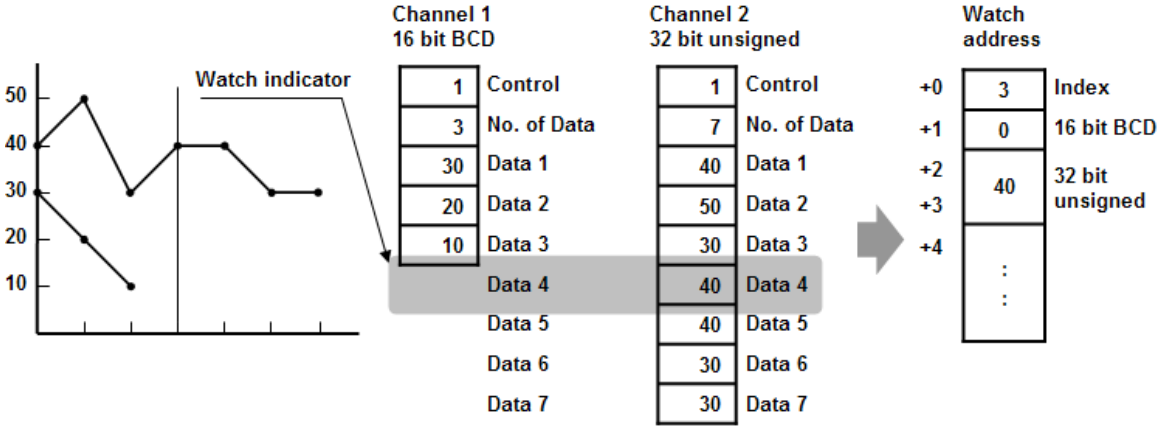
When watch address is set to LW-n, the value written into LW-n represents the channel index number to be called up. (start form 0)



- [Data Index] is a 16 bit unsigned integer; when the designated register of cursor line is 32 bit device, it will be stored in the bit 0-15.
- If the trend curve is cleared, when the cursor line is moved, “0” will be displayed, as shown below. In the example, there are no data in channel 1, when the cursor points at Data 4, “0” will be displayed as shown below.



- If there are fewer data in Channel 1, when position the cursor in Data 4, “0” will be displayed, as shown below.



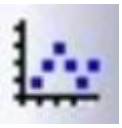
1. The maximum number of channels is 12.
2. The system can draw at most 32 trend curves.
3. The system can draw at most 1024 points for each channel.

13.20 XY Plot

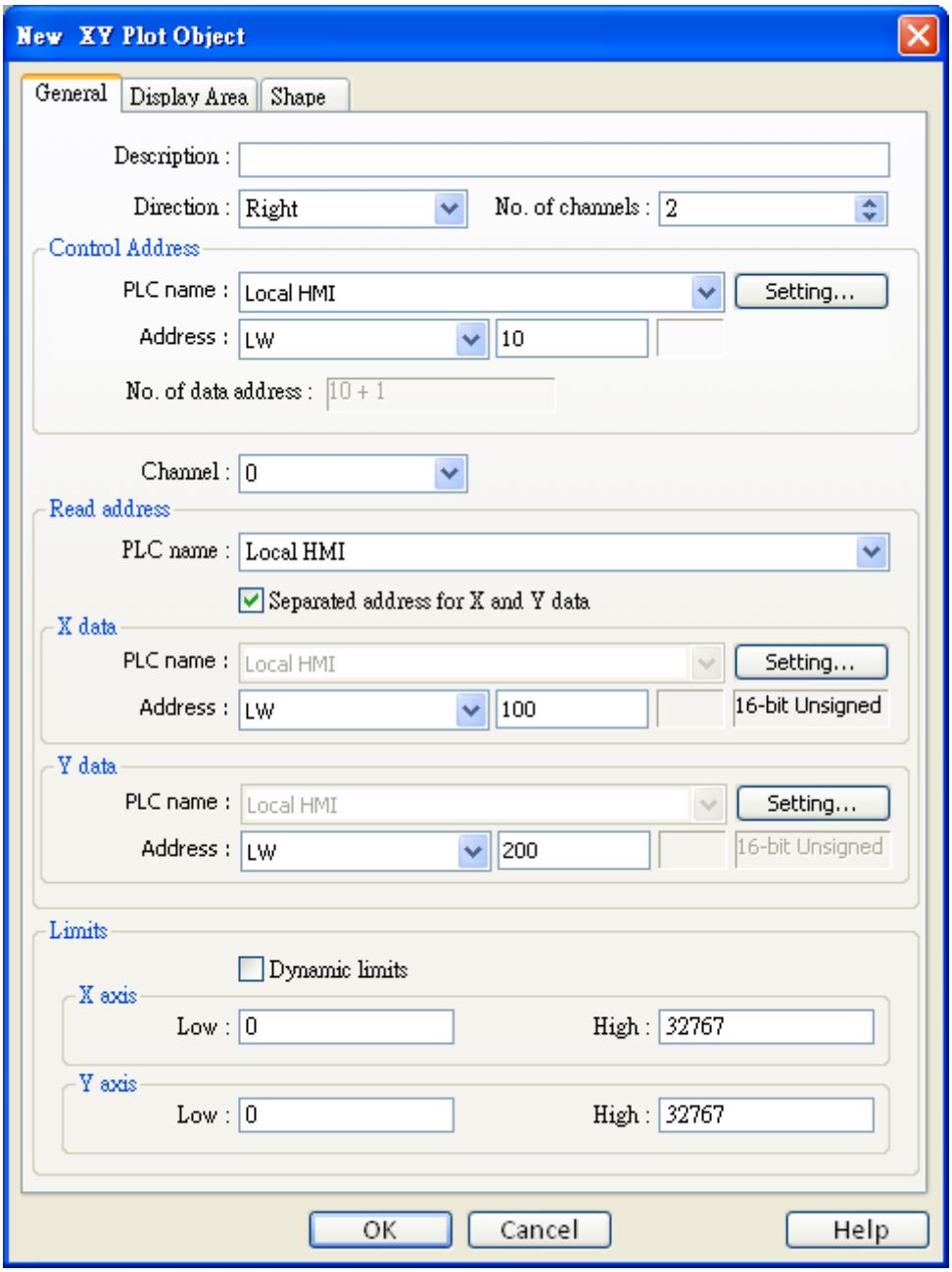
Overview

[XY Plot] is drawn where pair of word registers control the X and Y-axis. Up to 16 channels can be displayed simultaneously. This object is for easier data observation, and negative numbers can be displayed as well.

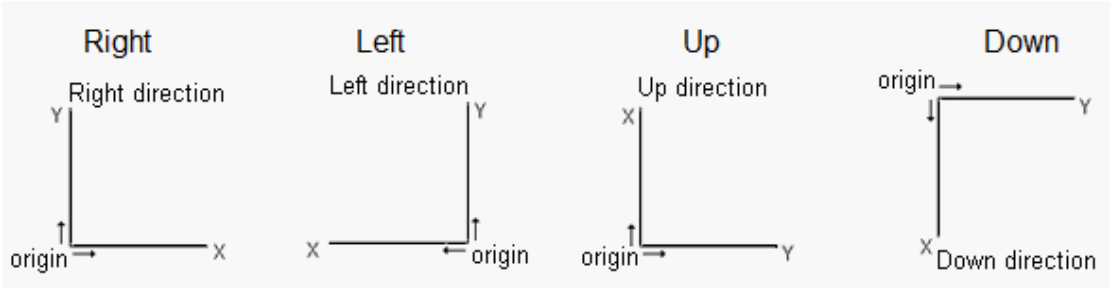
Configuration



Click the [XY Plot] icon on the toolbar to open a [XY Plot] object property dialog. Set up the properties, press OK button, and a new [XY Plot] object will be created.



[Direction] There are four selections, right, left, up or down.



[No. of channels]
Up to 16 independent channels may be selected for observation.

Control address
[Address]
Control the operation of all channels simultaneously. When the control address is set to LW-n, different values entered in it represents different commands, and the following one address LW-n+1 will also be designated to control the number of data points plotted.

Control address	value	Result
LW-n	1	Plot point on XY curve. (The plotted points are kept.)
	2	Clear all XY curves.
	3	Clear then plot new XY curve.
LW-n+1	Any number	control the number of data points plotted.

After operation [Control address] will be set 0, which represents action completed, ready for next operation.
[No. of data address]
Controls the number of data points. Each channel has a selection of up to 1023 points per plot.
[Channel]
Select a channel to configure.
Read Address
[PLC name]
Select a PLC which will be the source of [X data] and [Y data] and designate a read address. The format of the data register blocks used for the display channels depends on whether [Separated address for X and Y data] has been selected, and if [Dynamic limits] has been selected. The following explains the situations while 16-bit register is used:

1. If [Separated address for X and Y data] is not selected, and set [Read address] to LW-n:

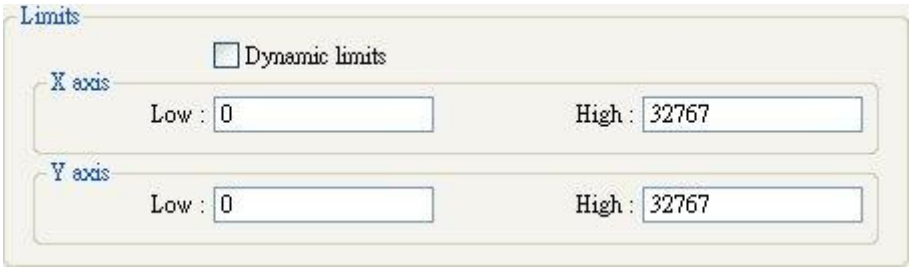
	Select **[Dynamic limits]**		Not select **[Dynamic limits]**	
	X data	Y data	X data	Y data
Low Limit	LW-n	LW-n+1	Constant	Constant
High Limit	LW-n+2	LW-n+3	Constant	Constant
1st data	LW-n+4	LW-n+5	LW-n+0	LW-n+2
2nd data	LW-n+6	LW-n+7	LW-n+1	LW-n+3
3rd data	LW-n+8	LW-n+9	LW-n+4	LW-n+5
4th data	LW-n+10	LW-n+11	LW-n+6	LW-n+7

2. If [Separated address for X and Y data] is selected, and set [X data] to LW-0, [Y data] to LW-100:

	Select **[Dynamic limits]**		Not select **[Dynamic limits]**	
	X data	Y data	X data	Y data
Low Limit	LW-m+0	LW-n+0	Constant	Constant
High Limit	LW-m+1	LW-n+1	Constant	Constant
1st data	LW-m+2	LW-n+2	LW-m+0	LW-n+0
2nd data	LW-m+3	LW-n+3	LW-m+1	LW-n+1
3rd data	LW-m+4	LW-n+4	LW-m+2	LW-n+2
4th data	LW-m+5	LW-n+5	LW-m+3	LW-n+3

Limits

When [Dynamic limits] is not selected, the Low and High limits can be set:



The Low and High limits are used for counting X and Y range in percentage.

$$\text{Scale } (\%) = \frac{\text{Read Address Value} - \text{Low Limit}}{\text{High Limit} - \text{Low Lmit}}$$

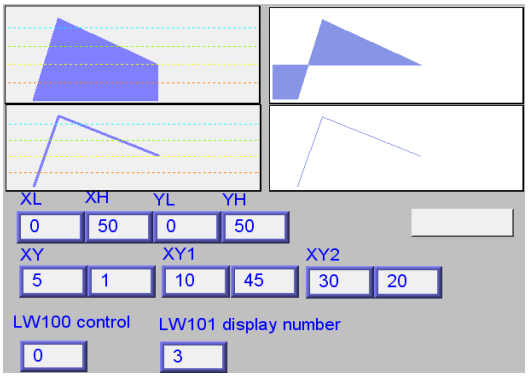
[Dynamic Limits]

When the format is 32-bit, and the address is LW-100 (n), the corresponding data will as table below: (If [Separated address for X and Y data] is not selected)

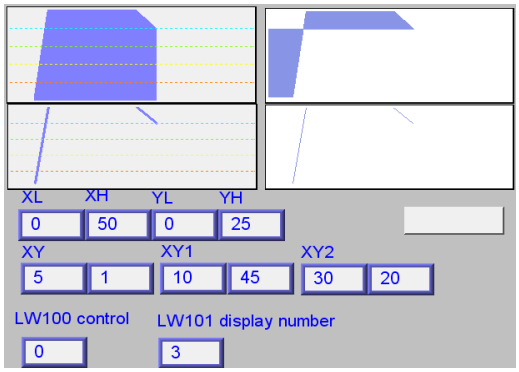
Content	X data		Y data	
Read Address	LW100 (n)			
Low Limit	LW100	n+0	LW104	n+4
High Limit	LW102	n+2	LW106	n+6
Data 0	LW108	n+8	LW110	n+10
Data 1	LW112	n+12	LW114	n+14
Data 2	LW116	n+16	LW118	n+18

If **[Dynamic limits]** is selected, a zoom effect can be created by changing the setting of Low / High Limits. (Please refer to Trend Display Object)

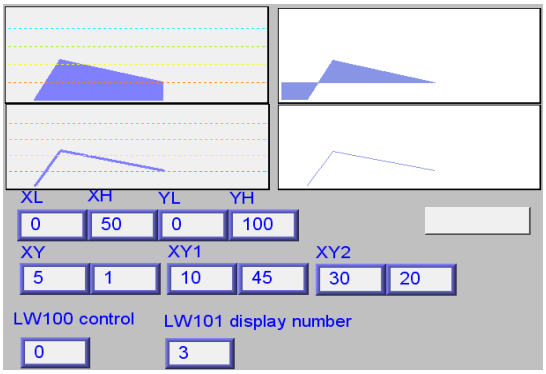
In the following example, XL=X low limit, XH=X high limit, YL=Y low limit, YH=Y high limit, and XY, XY1, XY2 are three XY data. When changing the high limits of X and Y axis, the result is shown below:



Original



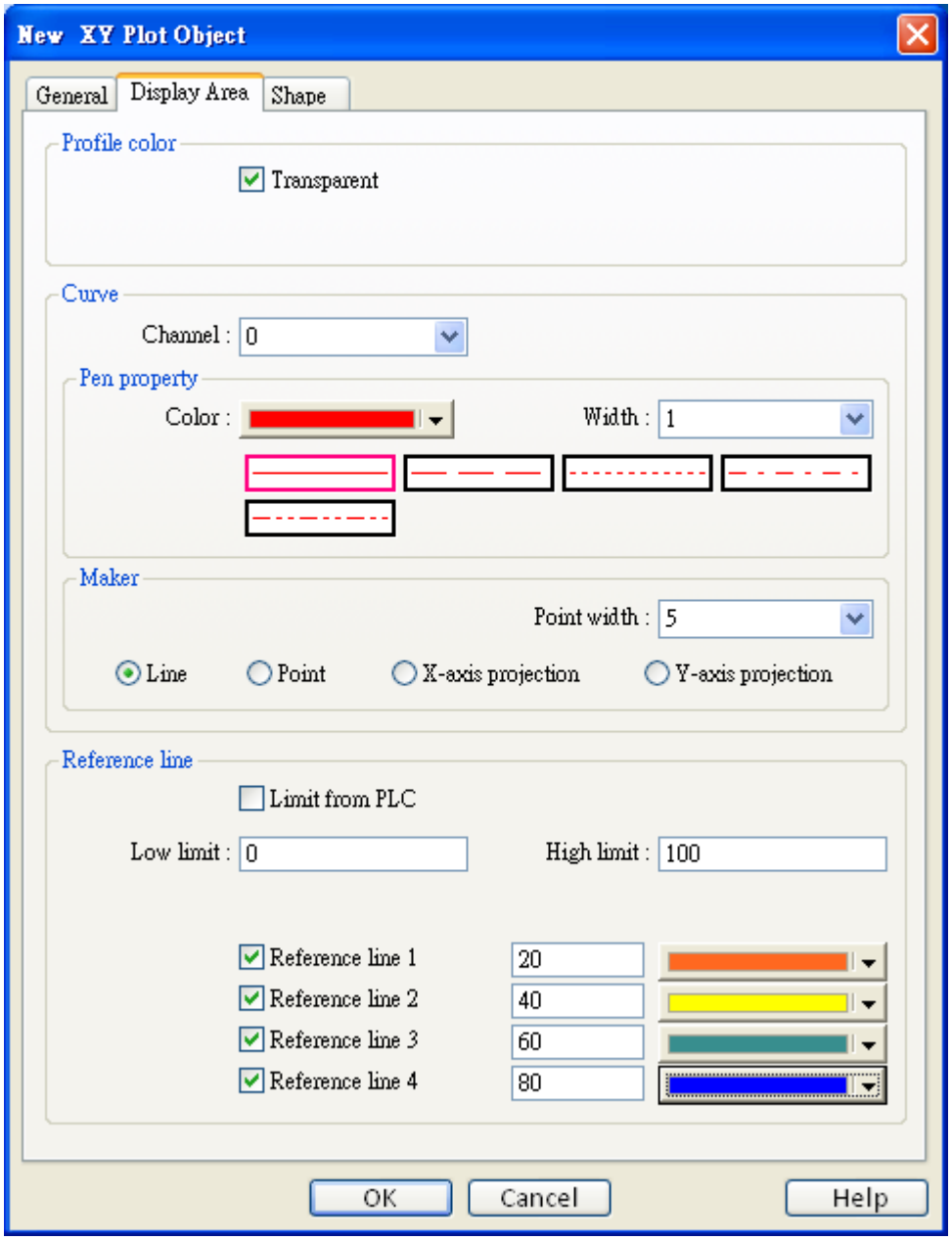
Change the high limit of Y axis to 25. (zoom in)



Change the high limit of Y axis to 100 (zoom out)

- X and Y data can be set to different formats, EX: If X data uses 16-bit unsigned, Y data uses 32-bit signed, please note the address setting.
- When using a Tag PLC, such as AB tag PLC, X and Y must be in the same format. When using different formats a warning will be shown.

[Display Area] tab



Profile color

Select frame and background colors, or select [Transparent] check box.

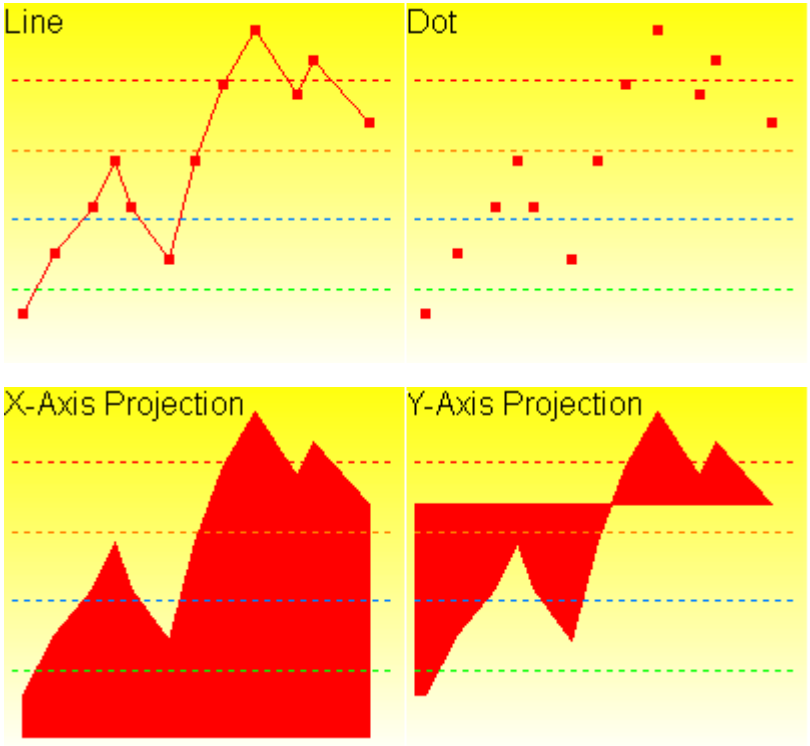
Curve

For each channel select the properties of color, width, and line style.

Maker

There are four different types of XY plot:

The result is shown below:



Example:

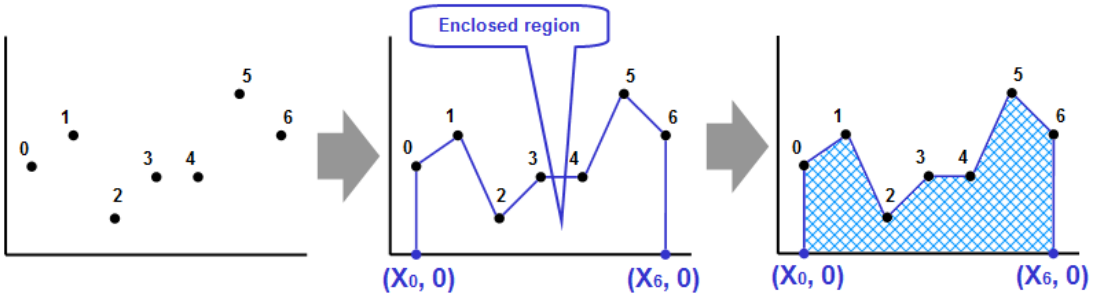
The curve shown below is drawn with 7 points numbered from P0 to P6.

The steps the system draws the X-axis Projection:

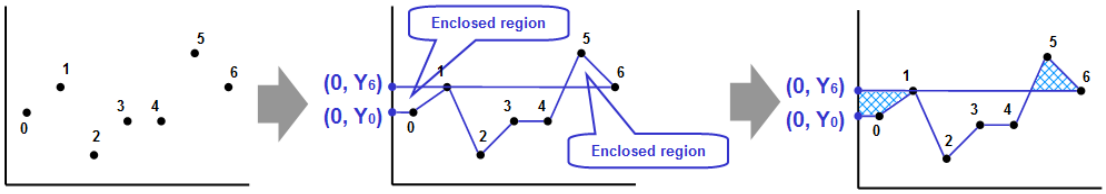
a. Calculates the two points in X-axis – (X_0, 0) and (X_6, 0).

b. Link all the points in the order of (X_0, 0), P0, P1… P6, (X_6, 0) and returns to (X_0, 0) at last.

c. Fill out all enclosed areas.

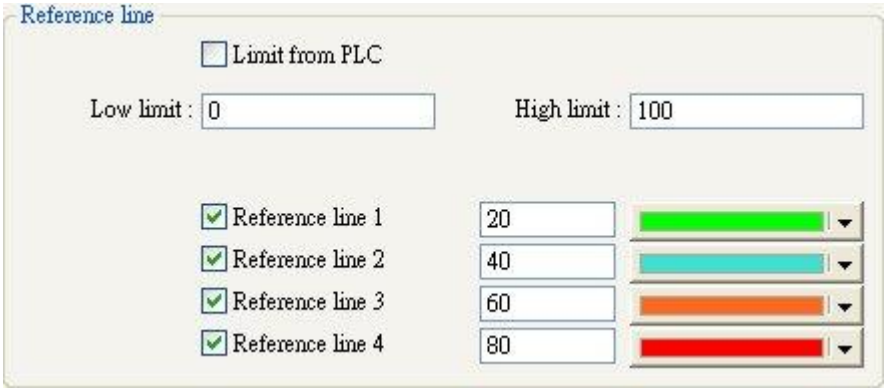


Similarly for Y-axis projection:

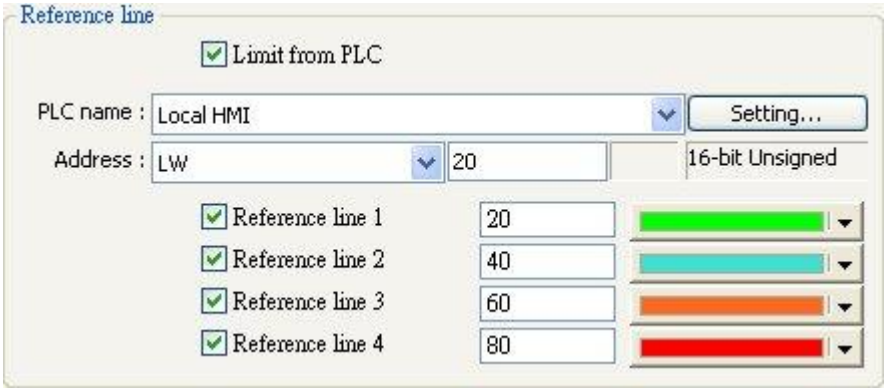


Reference line

Up to 4 horizontal reference lines can be configured on the graph. Fill in high, low limits and Y axis coordinate percentage values with choice of color, for each reference line.



If [Limit form PLC] is selected, designate a register to be the read address of reference line.



XY Plot can be drawn repeatedly up to 32 times:

1 channel➔32 times

2 channels➔16 times

The way to calculate: 32 divided by the number of channels

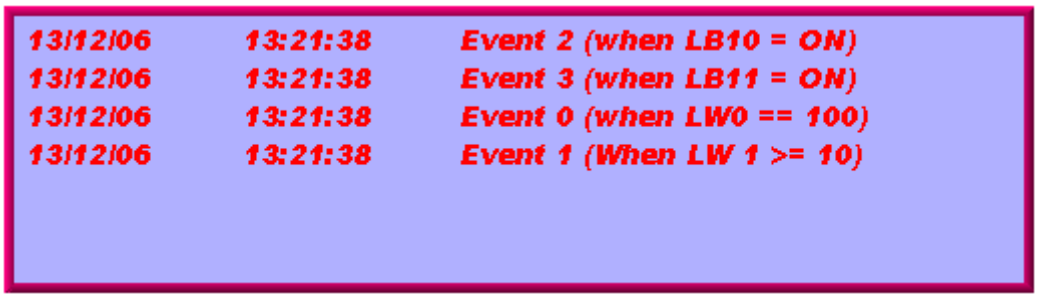
13.21 Alarm Bar and Alarm Display

Overview

[Alarm Bar] and [Alarm Display] objects are used to display alarm messages which are login in [Event (Alarm) Log]. When designated addresses meet the trigger condition, events or alarms will be displayed in time/date order as they occurred in [Alarm Bar] or [Alarm Display].

[Alarm Bar] scrolls all alarm messages in one single display line, where [Alarm Display] shows active alarm messages in multiple lines. Please refer to the relevant chapters about Event Log.

Alarm Bar - Displays alarm messages in one single line.

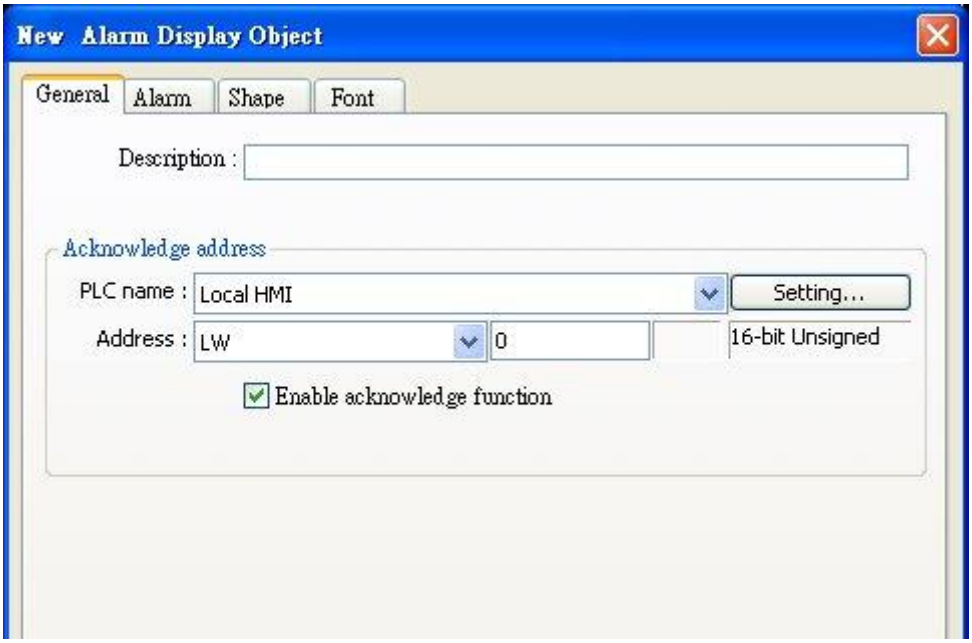


Alarm Display – Displays active alarm messages in multiple lines.

Configuration

Click the [Alarm Display] or [Alarm Bar] icon on the toolbar to open a [Alarm Display] or [Alarm Bar] object property dialog. Set up the properties, press OK button, and a new [Alarm Display] or [Alarm Bar] object will be created.

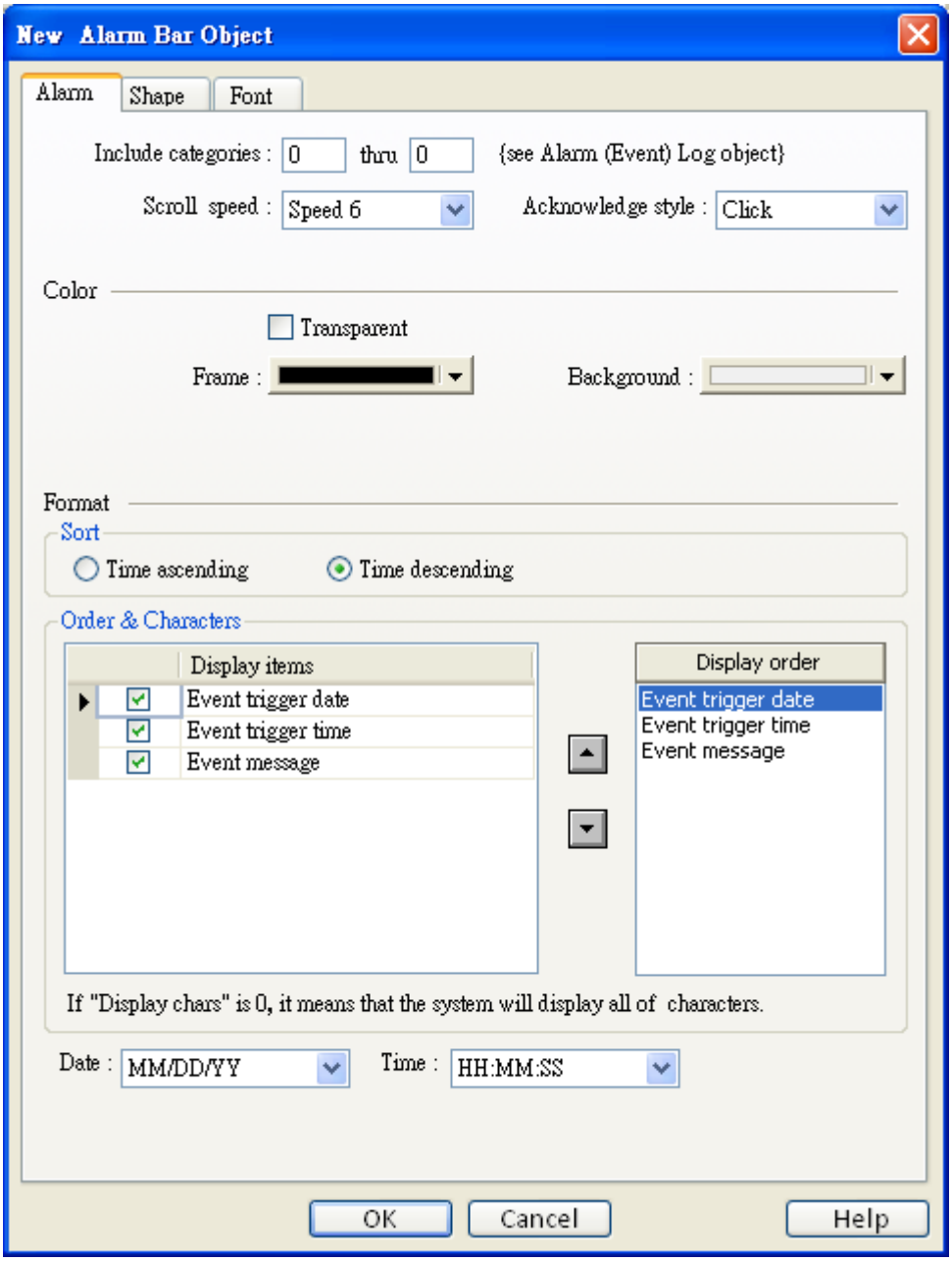
The difference between these two objects is that [Alarm Display] can be controlled by a designated [Acknowledge address].



[Enable acknowledge function]

If this check box is selected, the [Acknowledge value] selected for the associated event, specified in [Alarm (Event) Log] will be written to the [Acknowledge address] designated in [Alarm Display].

The following explains the general settings of these two objects:



[Include categories]

Select the categories displayed in [Alarm Display] or [Alarm Bar]. Alarms within these categories will be displayed only.

For example, if the category is set to 2 to 4, only events in categories 2, 3, 4 will be displayed. Please refer to “Chapter 7 Event Log”.

[Scroll Speed] and [Acknowledge style]

Select one of the speed settings at which the messages scroll, and whether to acknowledge the alarm by a single or double click. The selection of scroll speed is only available in [Alarm Bar].

Color

Use color to indicate different state of alarms. Frame and background may be made invisible by selecting [Transparent] check box.

Sort

Sort alarms in time ascending or descending order.

[Time ascending]

Latest alarm is placed last in the list.

[Time descending]

Latest alarm is placed first in the list.

Order & Characters

Use the up and down arrow buttons to adjust the display order of the alarms.

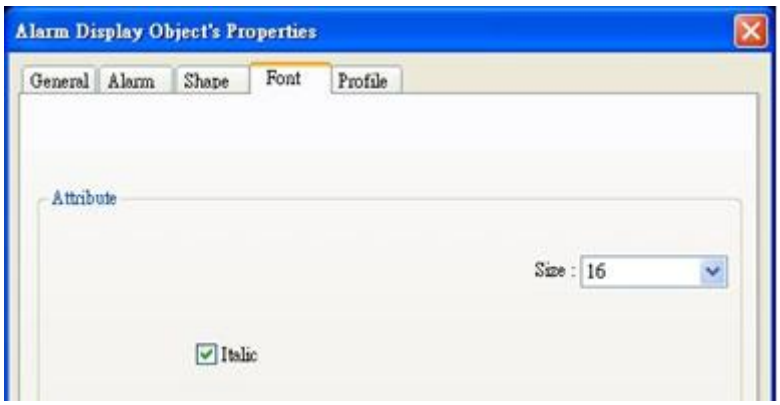
[Date]

Displays the date tag with each alarm message. The four formats of date tag:
MM/DD/YY、DD/MM/YY、DD/MM/YY、YY/MM/DD

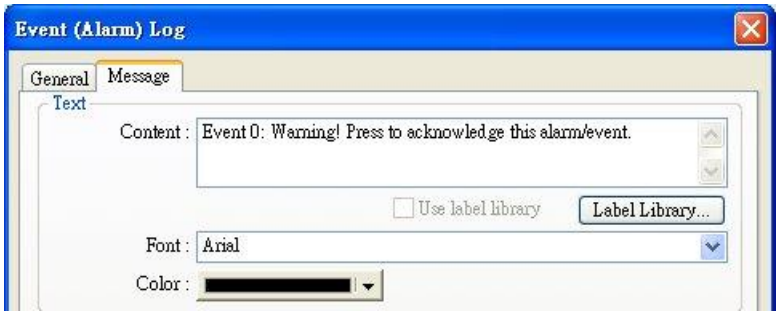
[Time]

Displays the time tag with each alarm message. The four formats of time tag:
HH:MM:SS、HH:MM、DD:HH:MM、HH

In the [Font] tab set the size of the font or select [Italic].



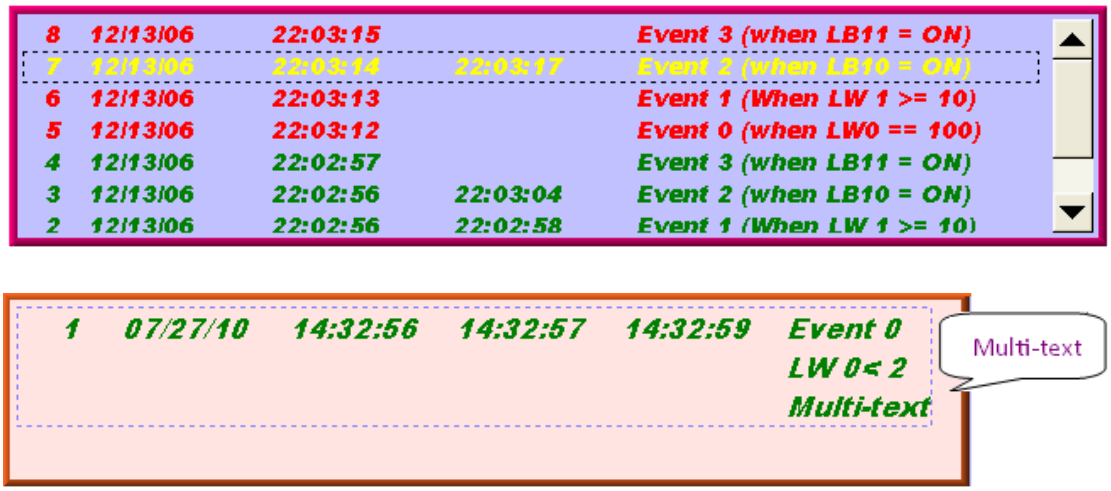
The font and color of the alarm messages is set in [Alarm (Event) Log] object:



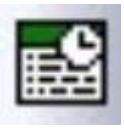
13.22 Event Display

Overview

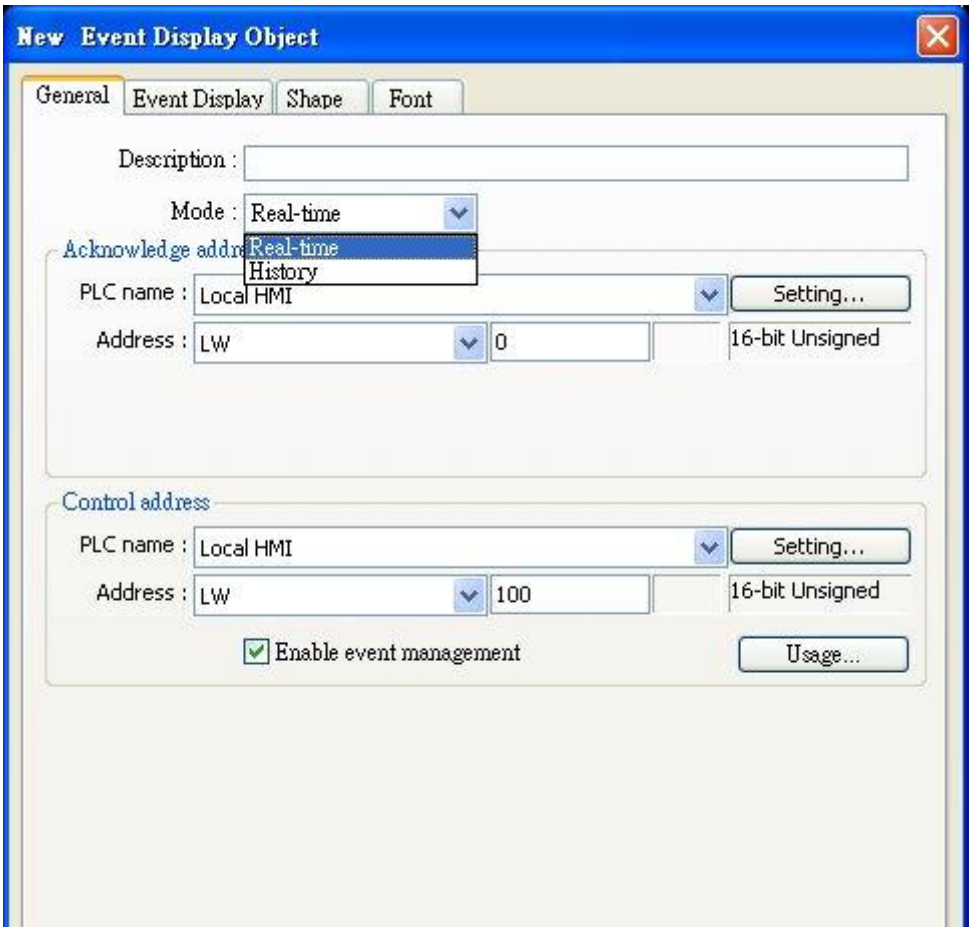
[Event Display] object is used to display alarm messages which are login in [Event (Alarm) Log]. When designated addresses meet the trigger condition, events or alarms will be displayed in time/date order as they occurred in [Event Display]. [Event Display] object displays real-time active events or historical files as they are triggered, acknowledged, and cleared, in multiple lines.



Configuration



Click the [Event Display] icon on the toolbar to open a [Event Display] object property dialog. Set up the properties, press OK button, and a new [Event Display] object will be created.



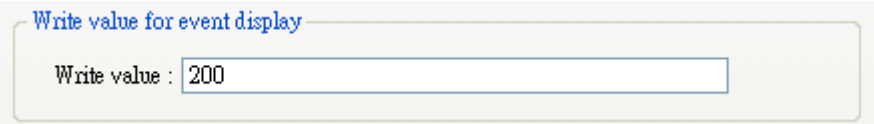
[Mode]

Select [Real-time] to work with current events, or [Historical] to view data from previous [Alarm (Event) Log] files.

a. Real-time

Acknowledge address

When in Real-time mode, and an event is acknowledged by touching an active display line, the [Acknowledge value] selected for the associated event, specified in [Alarm (Event) Log] will be written to the [Acknowledge address] designated in [Event Display]. Please refer to “Chapter 7 Event Log”.



[Enable event management]

If this check box is selected, writing a specific value into a register LW-n and LW-n+1, where n is an arbitrary number, will control [Event Display] object with different commands as shown below:

Address	Value	Command
LW-n	0	Display all events.
	1	Hide [Confirmed] events.
	2	Hide [Recovered] events.
	3	Hide [Confirmed] or [Recovered] events.
	4	Hide [Confirmed] and [Recovered] events.
LW-n+1	1	Delete a single selected event.

b. History Control

[Enable reading multiple histories]

- If this check box is **Not** selected

Daily event log files can be viewed and interrogated. A history control address can be designated:

The designated register contains a value which is used as an index to select historical files.
Index value 0 calls the latest file
Index value 1 calls the second latest, etc.

The example below shows how to use the historical control address. Set control address to LW-100. Assume four data logs exist dated:

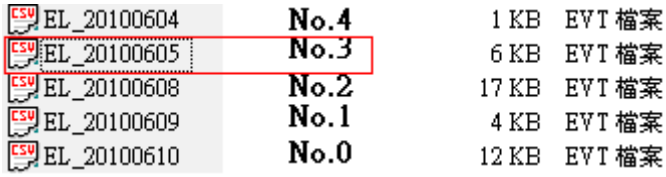
EL_20061120.evt,
EL_20061123.evt,
EL_20061127.evt
EL_20061203.evt,

The value in the control word selects the following records:

Value in [LW-100]	The selected record
0	EL_20061203.evt
1	EL_20061127.evt
2	EL_20061123.evt
3	EL_20061120.evt

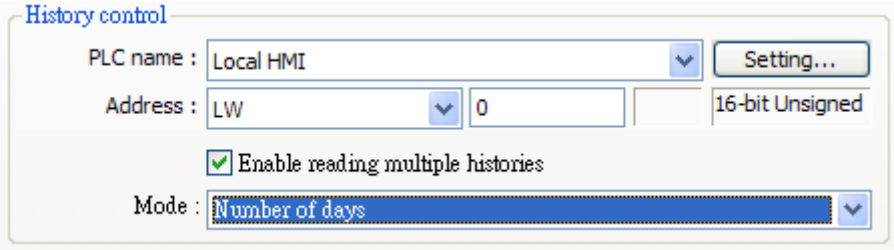
- If this check box is selected

Displays a list of events triggered in multiple days. If [History control] address is set to LW-0, LW-0 to LW-1 forms a range of log selection. Value in LW-0 represents the first history data. Example: As illustrated below, for showing it clearer, the history data is numbered according to the date they occur, (No.0、No.1、No.2…). If enter 3 in LW-0, the first data displayed will be data No. 3.



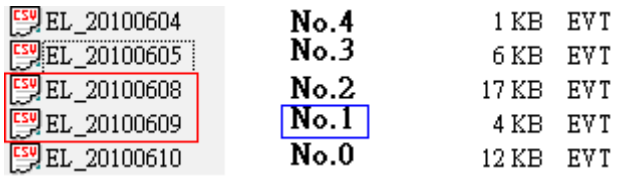
LW-1 has two modes:

a. Number of days

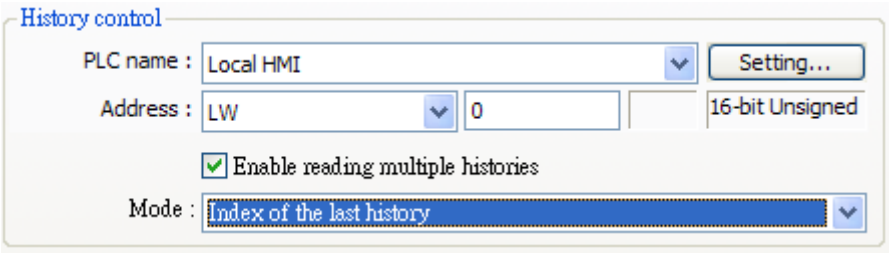


The data range starts from the number in LW-0. The value in LW1 represents how many days to be included from the start to days before.

Example: As illustrated below, if enter 1 in LW-0, enter 3 in LW-1, then the range of data will start form 20100609, and include data of 2 days before (while 20100609 is included). Since data of 20100607 does not exist in this example, the data displayed will only include 20100609 and 20100608.

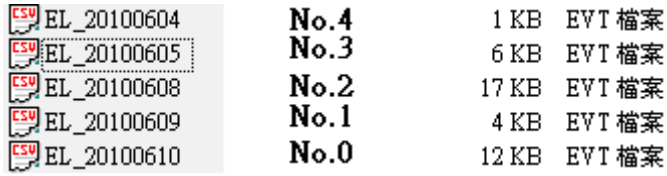


b. Index of the last history



The range of data will start form the number in LW-0 and end in LW-1.

Example: Enter 1 in LW-0, enter 3 in LW-1, the data displayed will include data No.1, No.2, No.3.



The maximum size of data that can be displayed is 4MB; the exceeding part will be ignored. The following shows how data will be stored while the data size is too big.

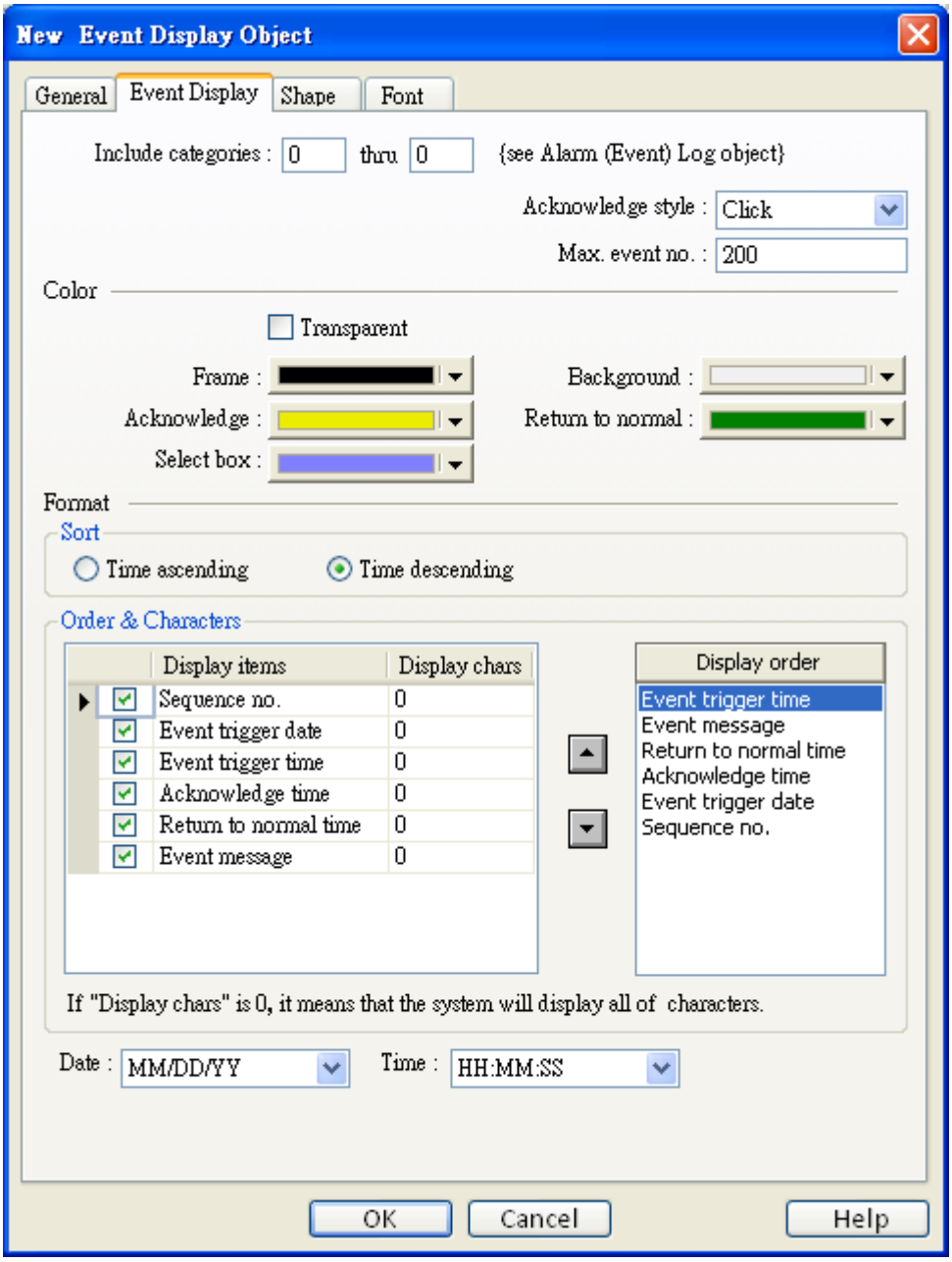
Example:

a. 5 history data, each is 0.5MB → The size of data displayed is 8 x 0.5MB
b. 5 history data, each is 1MB → The size of data displayed is 4 x 1MB
c. 5 history data, each is 1.5MB → The size of data displayed is 2 x 1.5MB+1 x 1MB (partial)

[Enable event management]

If this check box is selected, writing a specific value into register LW-n and LW-n+1, where n is an arbitrary number, will control [Event Display] object with different commands as shown below:

Address	Value	Command
LW-n	0	Display all events.
	1	Hide [Confirmed] events.
	2	Hide [Recovered] events.
	3	Hide [Confirmed] / [Recovered] events.
	4	Hide [Confirmed] and [Recovered] events.
LW-n+1	1	Delete a single selected event.



[Include categories]

Select the categories displayed in [Event Display]. Events within these categories will be displayed only.

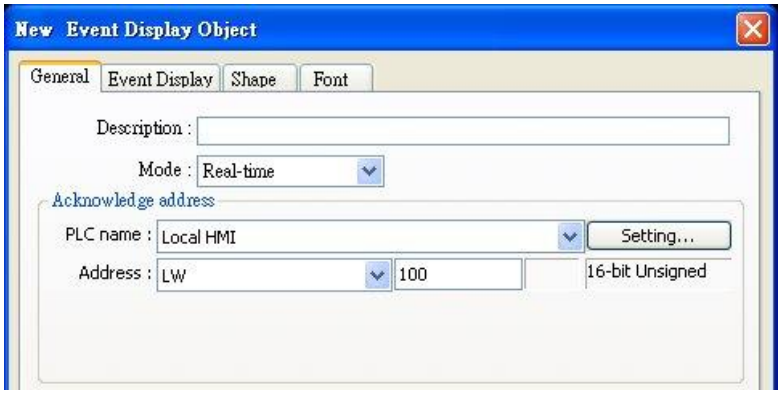
For example, if the category is set to 2 to 4, only events in categories 2, 3, 4 will be displayed. Please refer to “Chapter 7 Event Log”.

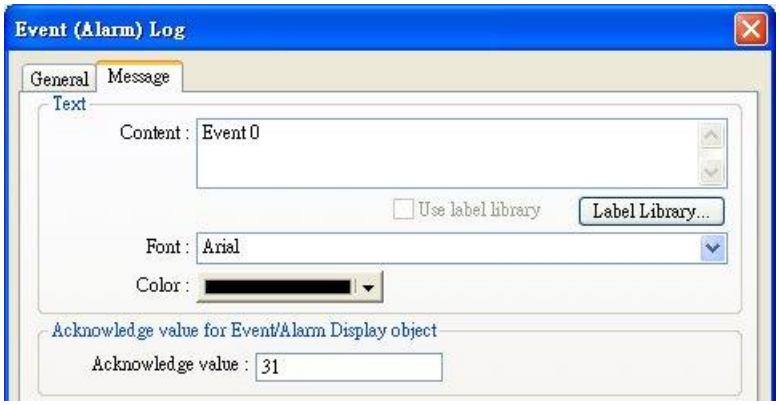
[Acknowledge style]

Select of Click or Double Click to acknowledge each single event. When a new event occurs the operator can tap the event line once or twice to acknowledge the new event.

When acknowledged, the text color of the event will change to that selected in the color section, and [Write value] associated with that event will be sent to the register designated in the Alarm (Event) log.

Users could configure an [Indirect Window] object so that when an event is acknowledged the [Write value] is written into the read address of the [Indirect Window] to call a popup window with warning message.
As shown below, if address is set to LW-100, when the event is confirmed, write 31 to the address. When users acknowledge the event, 31 is written to LW-100.





[Max. event no.]
The maximum number of events to be displayed in this Event Display.

When the amount of events displayed has reached [Max. event no.] set here, the oldest event will be removed and add a new event.

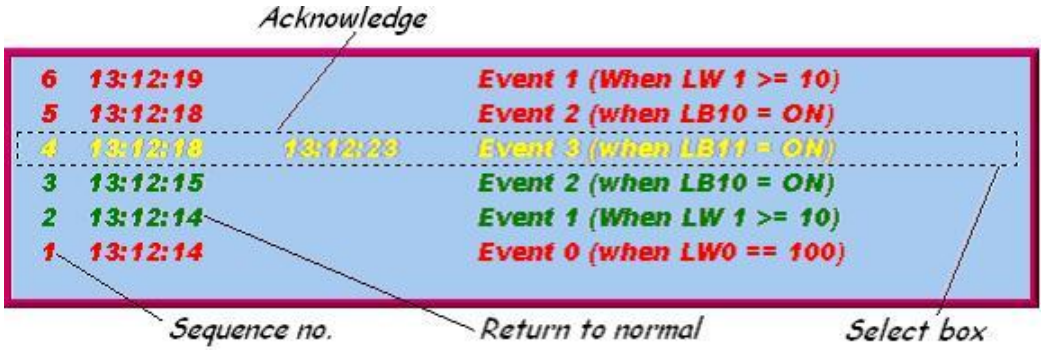
[Color]
Indicate the different event states. Frame and background may be made invisible by selecting [Transparent] check box.

a. Acknowledge

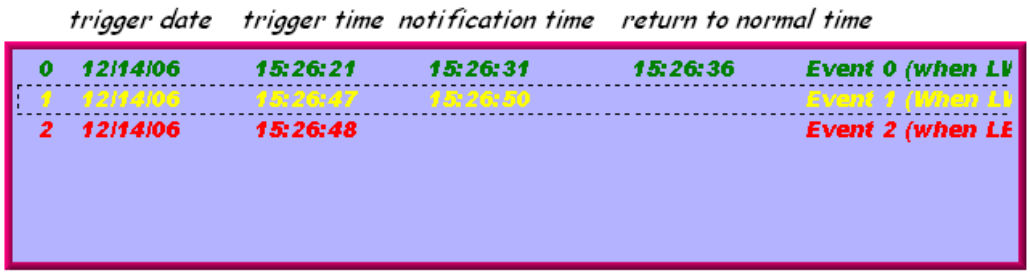
b. Return to normal

c. Select box –

The system draw a highlight box around the latest acknowledged event.



Format



Sort

Sort alarms in time ascending or descending order.

[Time ascending]

Latest alarm is placed last in the list.

[Time descending]

Latest alarm is placed first in the list.

Order & Characters

Use the up and down arrow buttons to adjust the display order of the alarms.

[Date]

Displays the date tag with each alarm message. The four formats of date tag:

MM/DD/YY、DD/MM/YY、DD/MM/YY、YY/MM/DD

[Time]

Displays the time tag with each alarm message. The four formats of time tag:

HH:MM:SS、HH:MM、DD:HH:MM、HH

In the [Font] tab set the size of the font or select [Italic].

The font and color of the alarm messages is set in [Alarm (Event) Log] object.

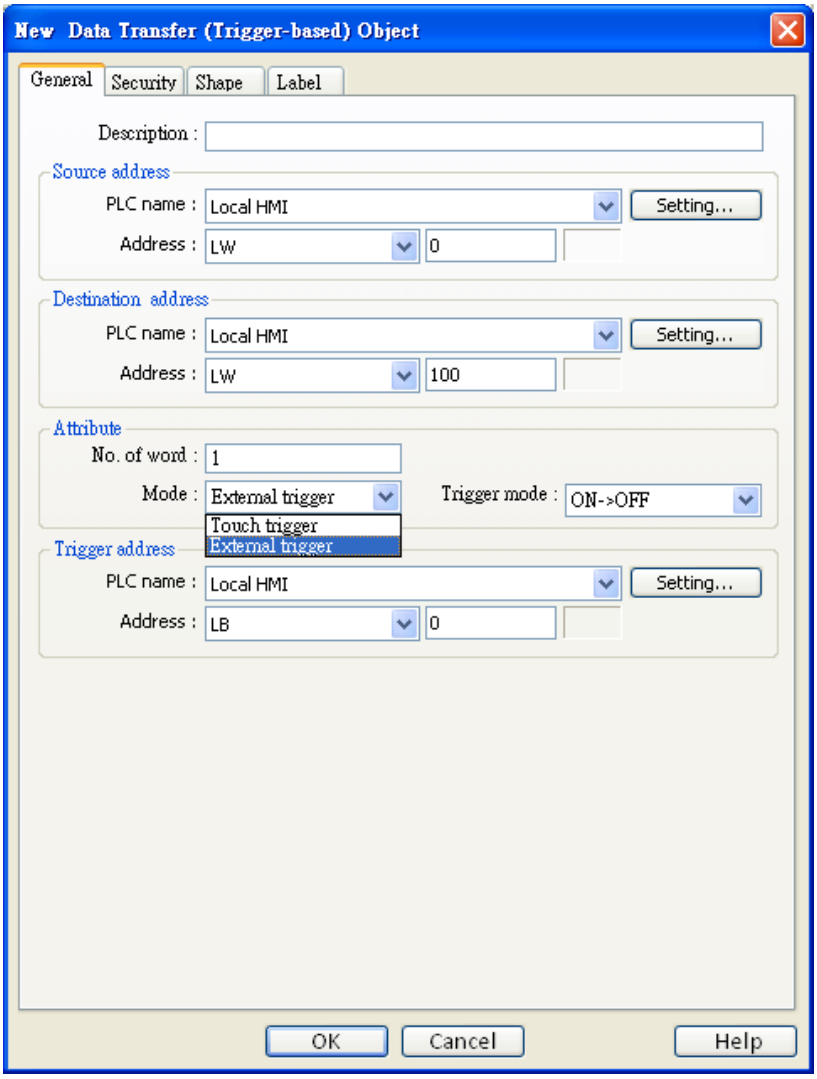
13.23 Data Transfer (Trigger-based)

Overview

[Data Transfer (Trigger-based)] object can transfer values from the source registers to the destination registers. The data transfer operation can also be activated by changing state of bit register.

Configuration

Click the **[Data Transfer (Trigger-based)]** icon on the toolbar to open a [Data Transfer (Trigger-based)] object property dialog. Set up the properties, press OK button, and a new **[Data Transfer (Trigger-based)]** object will be created.



Source address

[Data Transfer] object reads the data from [Source Address].

Destination address

[Data Transfer] object writes the data to [Destination Address].

Attribute

[No. of words]

It is the number of words to be transferred from [Source Address] to [Destination Address]. The unit is word (16-bit).

[Mode]

Set the trigger mode of data transfer.

a. **Touch trigger**
   Press the object to activate data transfer operation.

b. **External trigger**
   Activated when specify bit address changes state.

[ON >> OFF]

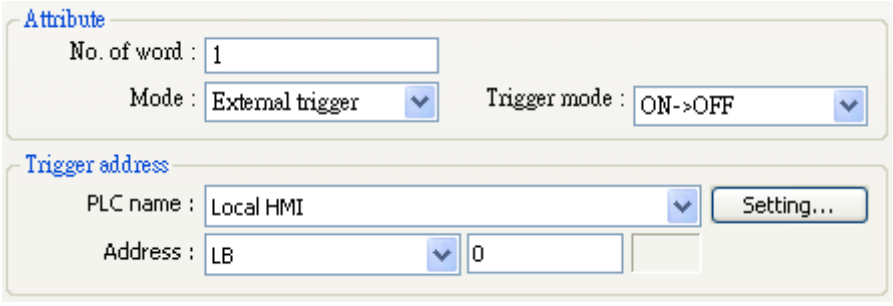
Transfer the data when the state of [Trigger address] changed from ON to OFF.

[OFF >> ON]

Transfer the data when the state of [Trigger address] changed from OFF to ON.

[ON <-> OFF]

Transfer the data when the state of [Trigger address] changes.



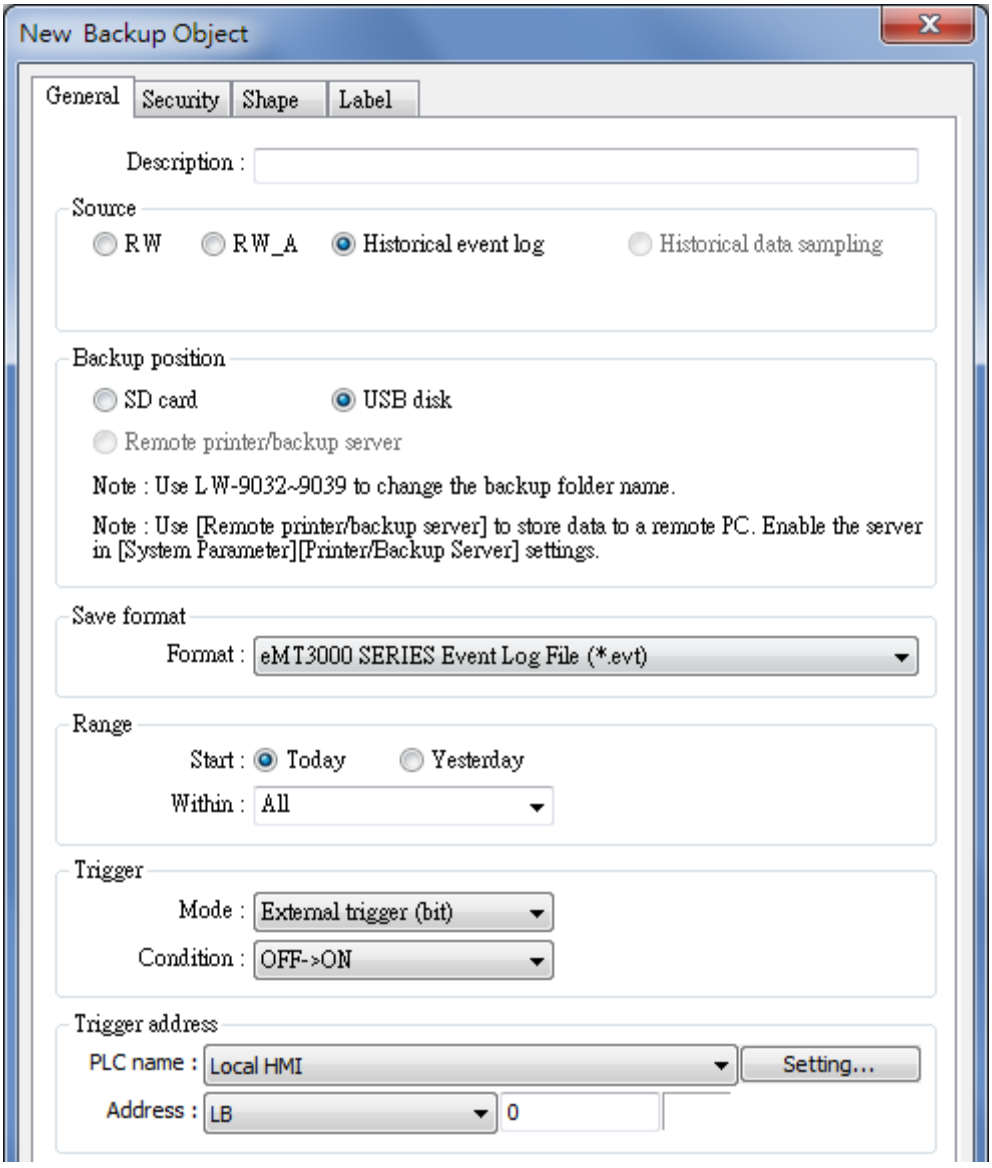
13.24 Backup

Overview

[Backup] object can transmit recipe data (RW, RW_A), event log and sampling data to external device or Remote printer/backup server. Users can also use [LB-9039] to monitor the backup status. If the system is backing up, the status of [LB-9039] will be turned ON.

Configuration

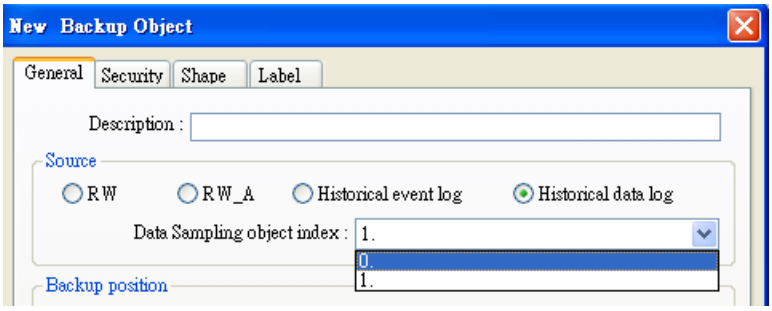
Click **[Backup]** icon on the toolbar to open a **[Backup]** object property dialog. Set up the properties, press OK button, and a new **[Backup]** object will be created.



Source

[RW], [RW_A], [Historical event log], [Historical data sampling]

Select one from the above for the source. There may be several data sampling objects registered in the project. When backing up **[Historical data log]**, use **[Data Sampling object index]** to select the right one to back up, as shown below.



Backup Position

Select the destination where the source files will be copied to.

a. **SD card** or **USB Disk**

   The external device connected to HMI.

b. **Remote printer/backup server**

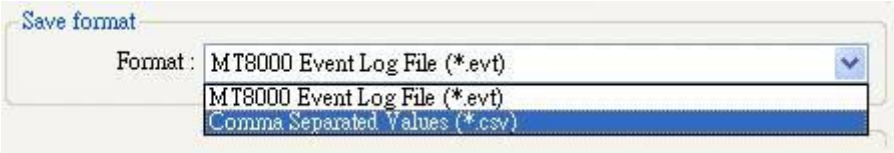
   To select this, users have to enable ***MT remote printer/backup server*** at: [Menu] » [Edit] » [System Parameters] » [Printer/Backup Server]

Save format

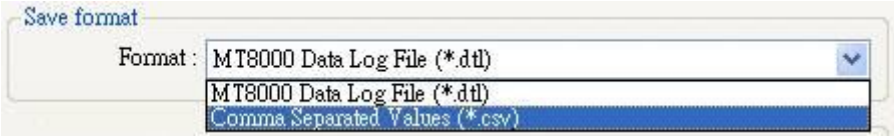
User can select the desired format to back up the file.

1. **HMI Event Log File (*.evt) / HMI Data Log File (*.dtl)**
2. **Comma Separated Values (*.csv)**

➢ **Event Log saved as a csv file**



➢ **Data Log saved as a csv file**



When back up event log in csv format, users can open the csv file in EXCEL, as shown on the right.

The [Data] column means:

0 = Event is triggered

1 = Event is acknowledged

2 = Event returns to normal

	A	B	C	D	E
1	[Creation time]				
2	Fri Oct 29 10:59:28 2010				
3	[Data fields]				
4	event	category	time	message	
5	[Data]				
6	0	0	11:19:42	"Emergency"	
7	0	5	11:19:43	"5"	
8	2	0	11:19:46	"LOW"	
9	2	5	11:19:49	"5"	
10	1	0	11:19:52	"Word"	
11	2	0	11:19:52	"Word"	

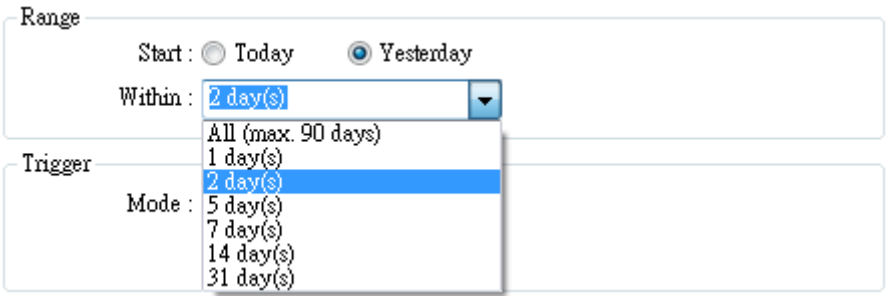
Range

[Start]

Choose either **[Today]** or **[Yesterday]**.

[Within]

Select the range of time period. For example, [Yesterday] is selected at [Start], and [2 day(s)] is selected here, which means the files obtained yesterday and the day before yesterday will be backed up. Select [All] to save all files in 90 days in the system.



Trigger

There are three ways to activate Backup function.

1. **Touch trigger**

   Touch the object to activate backup operation.

2. **External trigger (bit)**

   Register a bit device to trigger the backup operation.

   [ON >> OFF]

   Bit device change from ON to OFF to activate backup operation.

   [OFF >> ON]

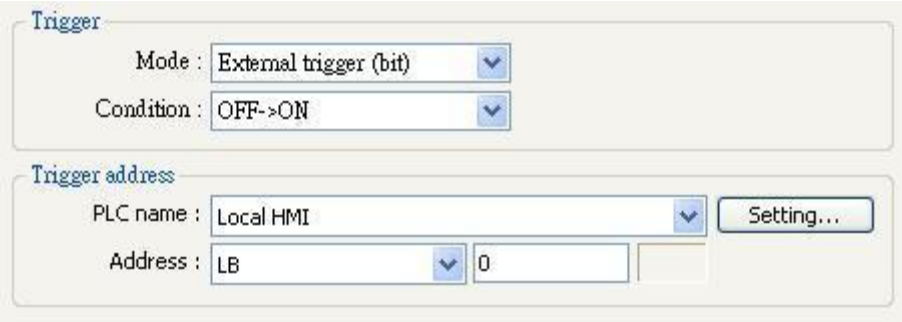
   Bit device changes from OFF to ON to activate backup operation.

   [ON <-> OFF]

Bit device change state to activate backup operation.

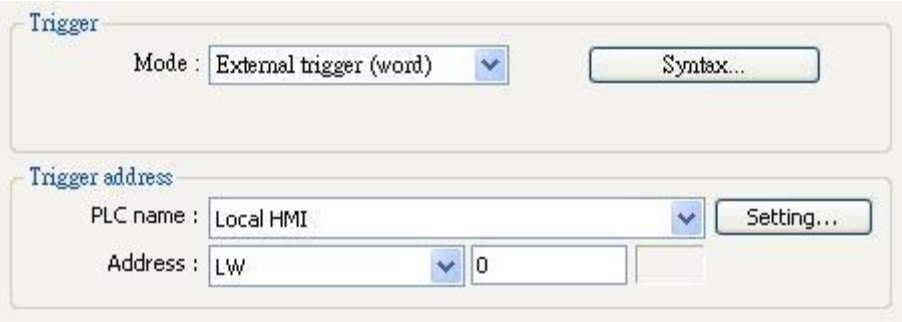
3. Trigger address

When use [External trigger], assign a bit device as shown below.



4. External trigger (word)

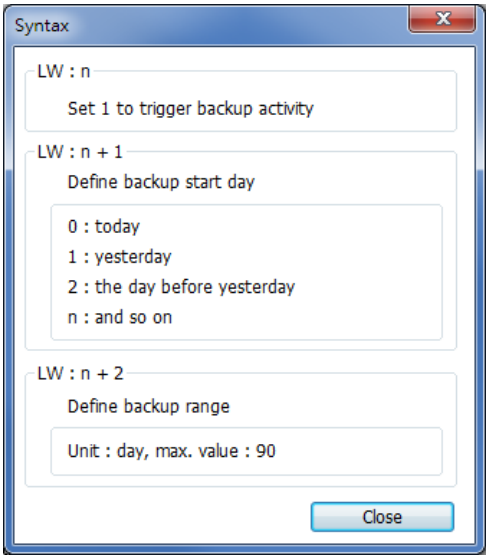
When selecting [External trigger (word)], users can specify the number of days to backup data using [Trigger address].



[Trigger address] usage (suppose LW-n is used):

- LW-n: Will start to back up when the value changes from 0 to 1.
- LW-n+1: The start date of backup.
- LW-n+2: The number of days for backup.

[Syntax] dialog is shown on the right. (The max. no of days is 90).



Note

1. All history files should have been saved in memory, either HMI memory, USB stick or SD card. Otherwise, the [Backup] object will not work.
2. The maximum number of days for backup is 90 days.

13.25 Media Player

When the first time a project using [Media Player] object, it is necessary to download the project via Ethernet. Therefore, Media Player drivers will be installed in HMI.

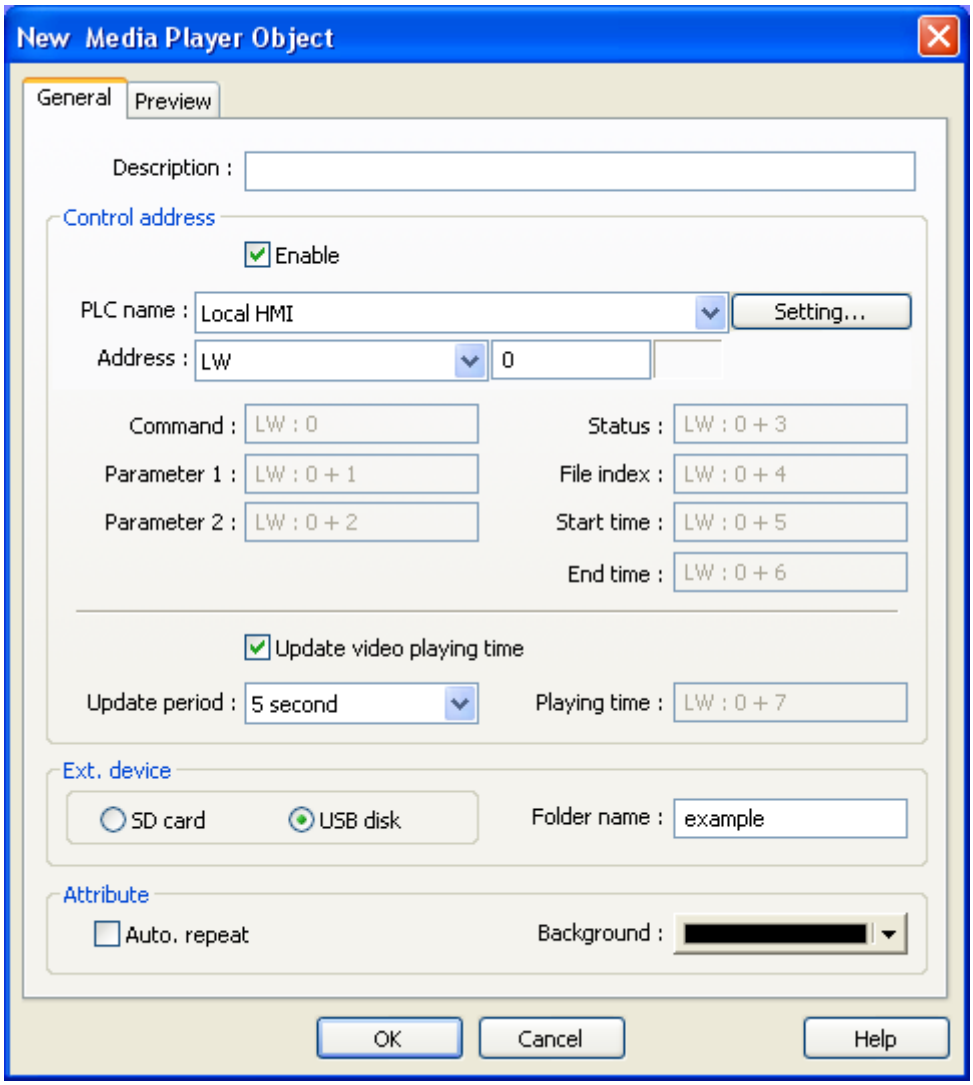
Overview

[Media Player] object will play video files with controls Seek, Zoom, and Volume to provide maintenance instructions or procedures on video so as to enable on-site operators to perform tasks efficiently.

Note: Only MT8000X-series supports this function.

Click [Media Player] object icon on the toolbar, or select from [Objects] » [Media Player]. [New Media Player Object] dialog box will appear.

General tab



Setting		Description
Control address	**Enable**	● Selected a. Control the [Media Player] operations. b. Designate a word register to control the object operations. ● Not selected No manual control. Video will be played automatically when the designated window opens.
	Command	Enter a value in the Command register to designate which action is required. ➢ Command (control address + 0)
	Parameter 1	Enter a value in Parameter 1 associated with each command action (see below for details.)

			➢ Parameter 1 (control address + 1)
	Parameter 2		Enter a value in Parameter 2 associated with each command action (see below for details.) ➢ Parameter 2 (control address + 2)
	Status		Indicate the status or errors. ➢ Status (control address + 3)
	File index		The file number in the designated folder. It is recommended to file the video name with a number. ➢ File index (control address + 4)
	Start time		The beginning of time (second). 0, normally. ➢ Start time (control address + 5)
	End time		The ending of time. (The period of time) ➢ End time (control address + 6)
	Video playing time	**Update video playing time**	● Enable The elapsed playing time of video will be writing into [Playing time] register at a rate set by [Update period] in seconds.
		Update period	Update period of [Playing time], range from 1 to 60 (second).
		Playing time	The elapsed playing time of video. (Second) ➢ Playing time (control address + 7)
Video file store location	**SD**		Play video files in SD card.
	USB 1 / USB 2		Play video files in USB 1 or USB 2 disk.
	Folder name		The folder name of video files stored in SD card or USB disk. Files must be stored in root directory. Subdirectories won't be accepted.(EX: root\sub is an invalid directory.)  1. [Folder name] cannot be empty. 2. [Folder name] cannot include **/\:*?"<>|**. 3. A folder name must be in ASCII characters.
Attribute	**Auto. repeat**		When finish playing all the video files, replay from the first file. Ex: Video 1 > Video 2 > Video 1 > Video 2
	Background		The background color of the object.

The data format for control address is 16-bit Unsigned or 16-bit Signed. If using 32-bit Unsigned or 32-bit Signed, only the previous 16 bits will be effective.

Control command：

a. Play index file

[Command] = 1
[Parameter 1] = file index
[Parameter 2] = ignore (set 0)

- Files are stored with file names in ascending order, the index 0 file is the first file and so on.
- If the file cannot be found, it will set [Status] bit 8 ON.
- Please stop the playing video before switching to another.

b. Play previous file

[Command] = 2
[Parameter 1] = ignore (set 0)
[Parameter 2] = ignore (set 0)

- If [File index] of previous file was zero it replays the same file.
- If the file cannot be found, it will set [Status] bit 8 ON.

c. Play next file

[Command] = 3
[Parameter 1] = ignore (set 0)
[Parameter 2] = ignore (set 0)

- If there are no more files it plays the index 0 file..
- If the file cannot be found, it will set [Status] bit 8 ON.

d. Pause / Play Switch

[Command] = 4
[Parameter 1] = ignore (set 0)
[Parameter 2] = ignore (set 0)

e. Stop playing and close file

[Command] = 5
[Parameter 1] = ignore (set 0)
[Parameter 2] = ignore (set 0)

f. Start playing at designated target location

[Command] = 6
[Parameter 1] = target time (second)
[Parameter 2] = ignore (set 0)

- Parameter 1 (target time) must be less than the ending of time or it will play the last second.

g. Forward

[Command] = 7
[Parameter 1] = target time (second)
[Parameter 2] = ignore (set 0)

- Going Forward to the designated second in [Parameter 1]. If the video is paused, the forwarding action will be started by playing.
- When the designed time is after than the ending of time, it will play the last second.

h. Backward

[Command] = 8
[Parameter 1] = target time (second)
[Parameter 2] = ignore (set 0)

- Going Backward to the designated second in [Parameter 1], If the video is paused, the backward action will be started by playing.
- When the designed time is earlier than the beginning of time, it will play from beginning.

i. Adjust volume

[Command] = 9
[Parameter 1] = volume (0 ~ 128)
[Parameter 2] = ignore (set 0)

- Default volume is 128.

j. **Set video display size**

[Command] = 10

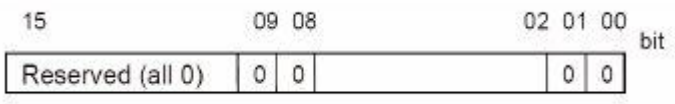
[Parameter 1] = display size (0 ~ 16)

[Parameter 2] = ignore (set 0)

- [Parameter 1 = 0]：Fit video image to object size.
- [Parameter 1 = 1 ~ 16]：Magnification from 25% ~ 400% in 25% increments where 1 = 25%, 2 = 50%, 3 = 75% and so on.

k. Status (control address + 3)

- When playing a video the system will turn ON file open and file playing bits 00 and 01. If the file cannot be scanned, or an unacceptable command is entered, the bit 08 will be set ON. If the file format is not supported, or a disk I/O error occurs, during playback (eg. USB disk unplugged), the file error bit 09 is set ON.



00: File Opened / Closed (0 = closed, 1 = opened)

01: File Playing (0 = not playing, 1 = playing)

08: Command Error (0 = accepted, 1 = incorrect)

09: File Error (0 = accepted, 1 = incorrect)

- The figure shows the status value associated with each state:

  Stop = 0
  Pause = 1
  Playing = 3

- [Command], [Parameter 1], and [Parameter 2] are write addresses. All others are read only.

Preview tab：

Users can test whether the video format is supported by using the preview function.

[Load]

Select the testing video to preview

[Play / Pause]

Select to start playing video or pausing.

[Forward << / Backward >>]

Go forward of the video of go backward.(in minutes)

[Stop]

Stop playing and close the file. If testing another video is needed, please stop playing the current video first.

- Only one video file can play at one time.
- If [control address] is not enabled and [Auto. repeat] is not selected, after finish playing the first file, the system will stop playing and close.
- If [control address] is not enabled, the system will find the first file in the designated folder and start to play (in ascending order of the file name).
- If the file can be previewed, the format is supported. If the video image quality is poor, please adjust the resolution.
- The supported formats: mpeg4, xvid, flv…etc.

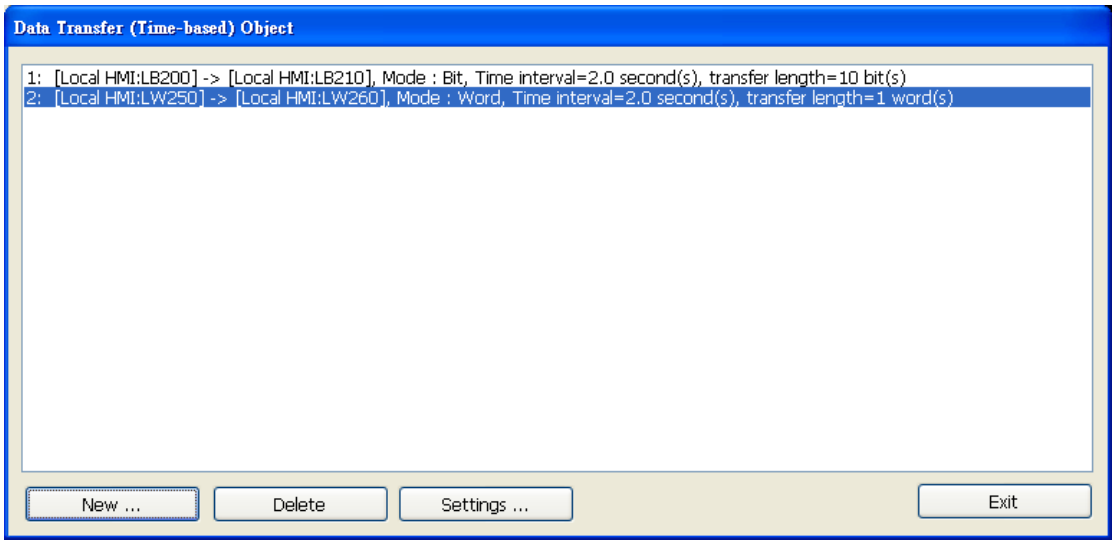
13.26 Data Transfer (Time-based)

Overview

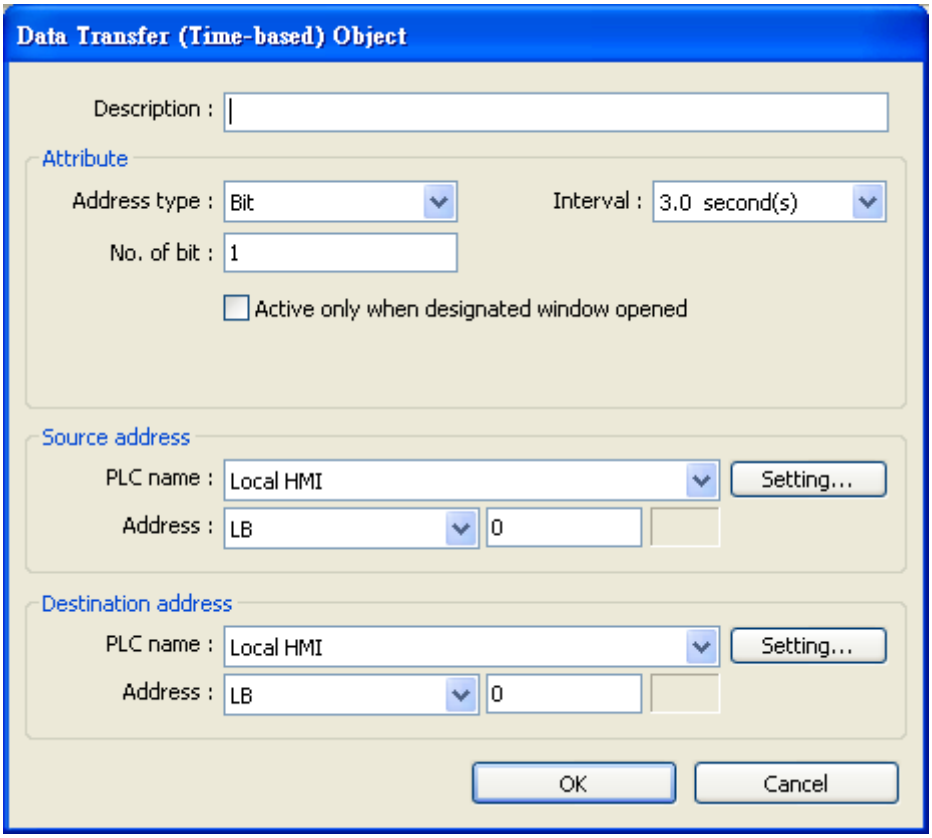
[Data transfer (Time-based)] object is similar to [Data transfer (Trigger-based)] object that they all transfer the data from source to destination register. The difference is that the [Data transfer (time-based)] object transfers data based on time schedule, and is able to transfer data in bits.

Configuration

Click **[Data Transfer (Time-based)]** icon on the toolbar to open the [Data Transfer (Time-based) Object] management dialog, as shown below:



Press the [New] button in the above dialog box to open a [Data Transfer (Time-based)] object property dialog, as shown below, set up the properties, press OK button, and a [Data Transfer (Time-based)] object will be created.



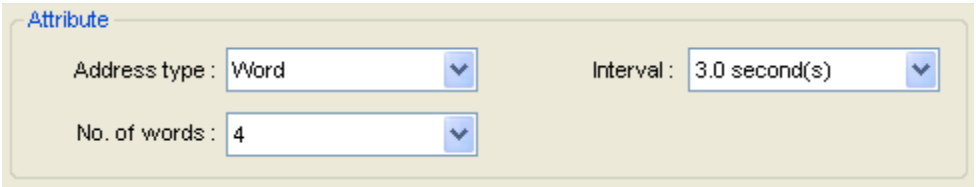
Attribute

[Address type]

Select the data transfer unit, either [Bit] or [Word].

[No. of words] or [No. of bits]

[No. of words] will be shown when [Word] is selected in [Address type], meaning the data transfer unit is word. See the picture below.



When **[Bit]** is selected, the unit of data transfer is bit. See the picture below.

[Interval]

Select the time interval for each data transfer, for example, when 3 seconds is set, the system will transfer data every 3 seconds.

- Specifying a short time interval or a big number of data to transfer may cause an overall performance of system decrease. Therefore, it is recommended that users choose a longer time interval and a smaller amount of data to transfer.
- When a short interval is inevitable, be aware of the interval must be longer than the data transfer operation. For example, if the data transfer operation takes 2 seconds, you must set the interval longer than 2 seconds.

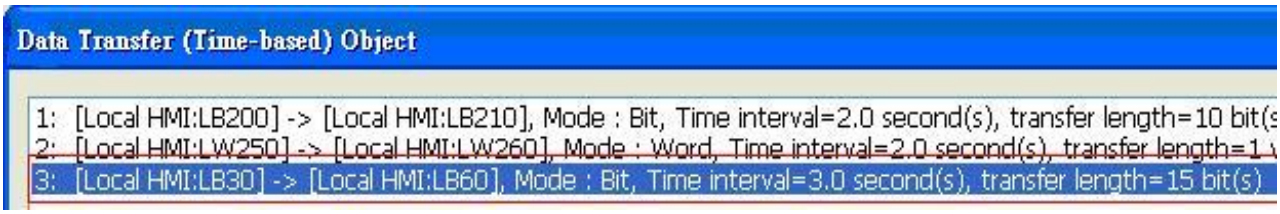
Source address

[Data Transfer] object reads the data from [Source Address].

Destination address

[Data Transfer] object writes the data to [Destination Address].

After all settings are completed, press [OK] button, and a new [Data Transfer (Time-based)] object will be created. The [Data Transfer (Time-based)] management dialog displays brief information for each object as shown below.



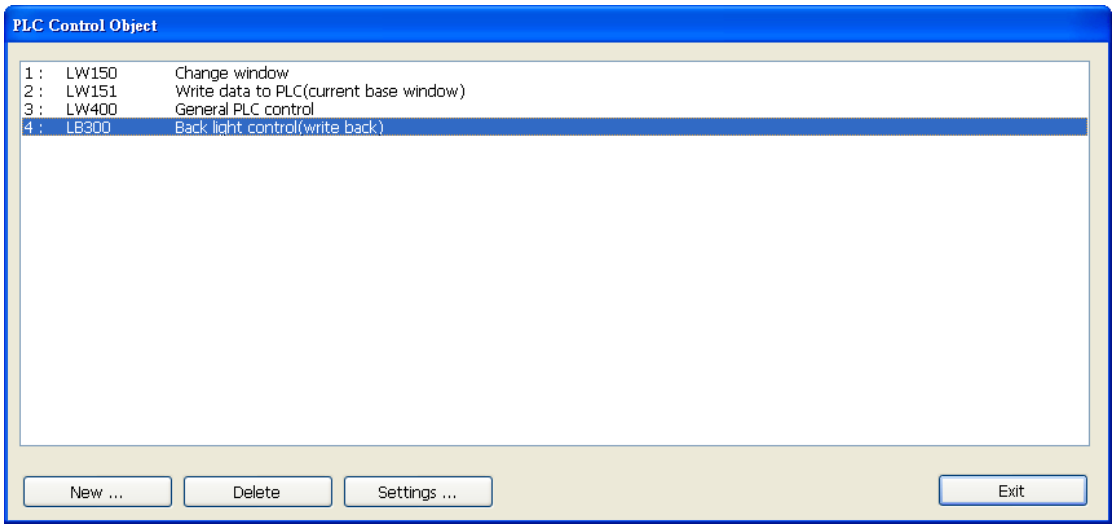
13.27 PLC Control

Overview

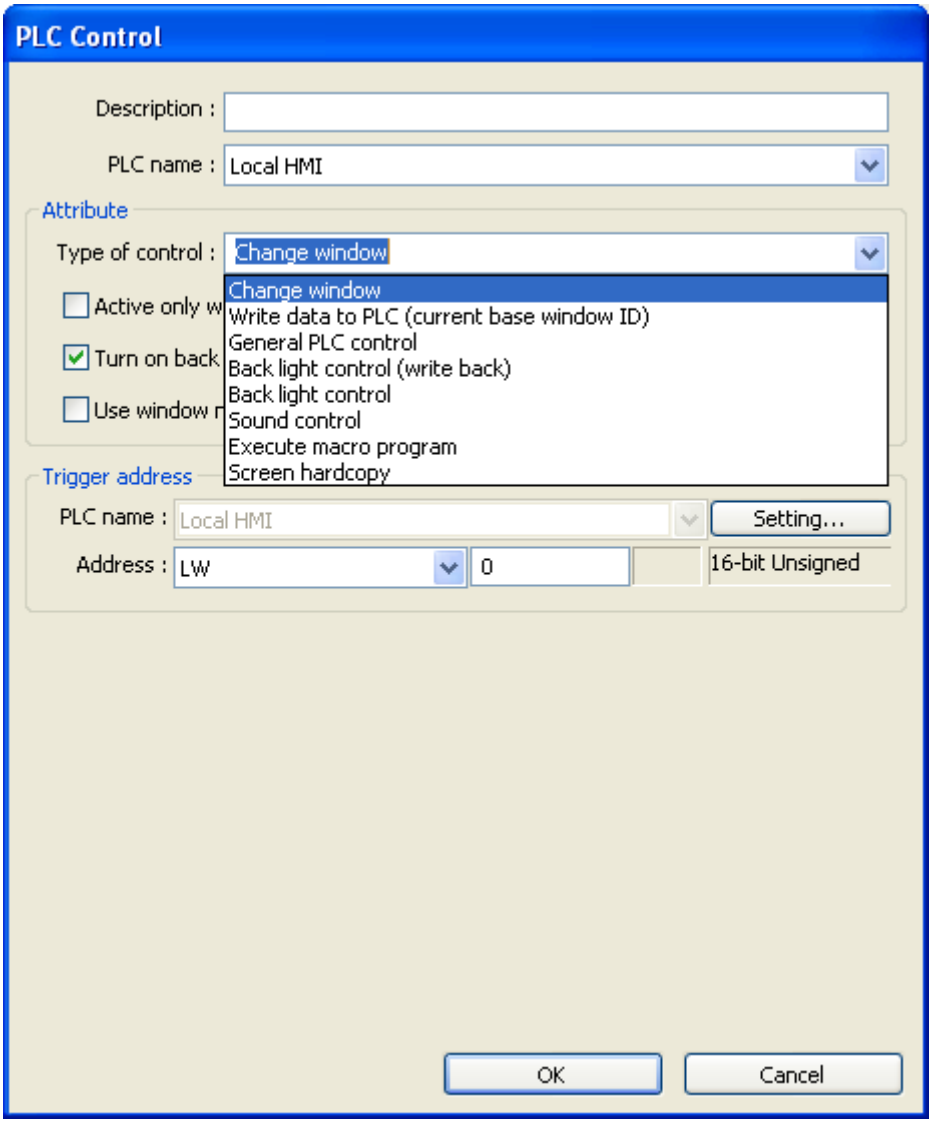
[PLC Control] object can execute commands when it is triggered. These commands include [Change window], [Back light control]… etc.

Configuration

Click the **[PLC Control]** icon on the toolbar to open the **[PLC Control Object]** management dialog. To add a **[PLC Control]** object, click [New], set up the properties, press OK button and a new **[PLC Control]** object will be created.



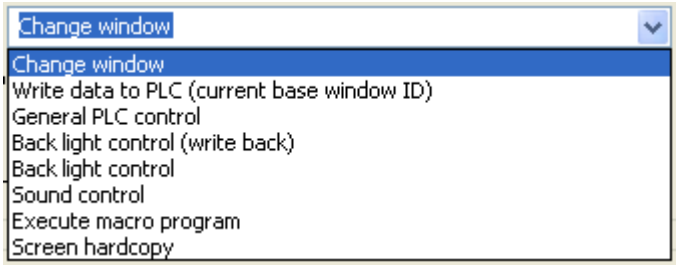
Click [New] and the following dialog box will appear.



Attribute

[Type of control]

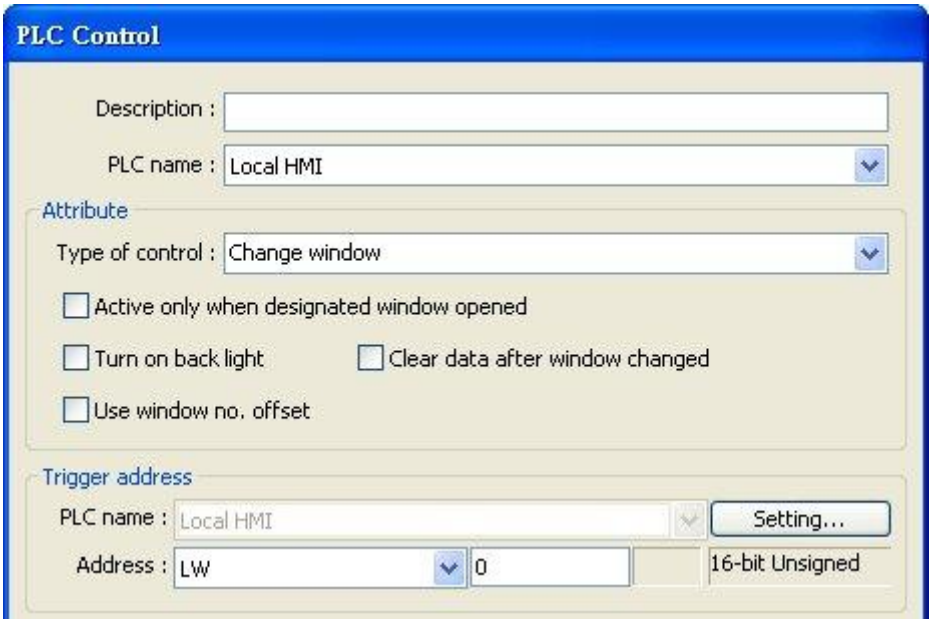
Select a type of control form the drop down list.



a. Change window

Place a valid window number in the designated trigger address to change the base screen to the new window number. By default the new window number is written back into a designated address.

For example, if current window is window no. 10 and the setting is as shown below:



When LW-0 is changed to 11, the system will change the current window to window no. 11, and then write 11 to LW-1.

When the window is changed, the new window number is written back into the address that is calculated by [Trigger address] and the data format, as shown in the table below.

Data Format	Trigger address	Write address
16-bit BCD	Address	Address + 1
32-bit BCD	Address	Address + 2
16-bit Unsigned	Address	Address + 1
16-bit Signed	Address	Address + 1
32-bit Unsigned	Address	Address + 2
32-bit Signed	Address	Address + 2

[Activate only when designated window opened]
Allow this operation only if a particular screen is displayed.
[Clear data after window changed]
Reset the value at trigger address to zero after the [PLC Control] object is activated.
[Turn on back light]
Illuminate the screen when the [PLC Control] object is activated.
[Use window no. offset]
Select the check box and select an window offset, the new window no. to change to will be the value in [Trigger address] plus the offset.

For example, if [Trigger address] is LW-0 and offset is set to 5. When the value in LW-0 is 10, the new window number will be window no. 15 (10+5). The range of the offset is -1024 to 1024.

- If [LB-9017] is set ON, the write-back function will be disabled.

b. Write data to PLC (current base window)

Each time the base window is changed, the new window number will be written into the [Trigger address].

c. General PLC Control

Transfer word data blocks from PLC to HMI, and vise-versa, and the transfer direction is controlled by the value contained in the [Trigger address].

Value in [Trigger Address]	Action
1	Transfer data from PLC register → HMI RW register
2	Transfer data from PLC register → HMI LW register
3	Transfer data from HMI RW register → PLC register
4	Transfer data from HMI LW register → PLC register

Four consecutive word registers are used as discussed in the following table:

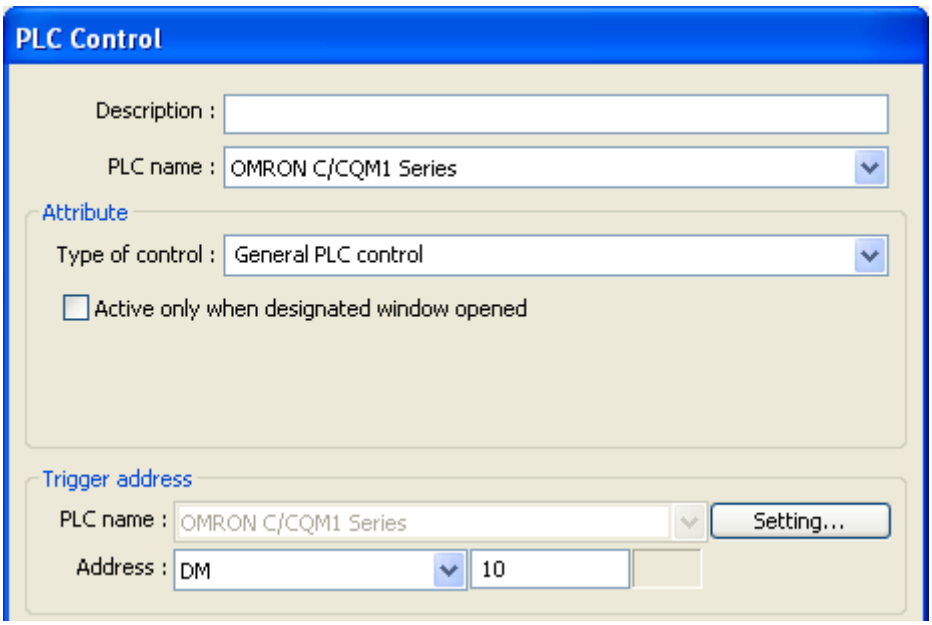
Address	Purpose
[Trigger address]	**Determine the direction of data transfer** The valid values are listed in the above table. When a new control code is written into the register, HMI will start to transfer. After data transfer is finished, the value will be set to 0.

[Trigger address+1]	Number of words to transfer.
[Trigger address+2]	**Offset to the start address of PLC register** Assume the value is “n”, where n is an arbitrary number, the start address of PLC register is [Trigger address + 4 + n]. Take an OMRON PLC as an example: If [Trigger address] uses DM-100, [Trigger address + 2] will be DM-102. If the value in DM-102 is 5, the start address of data source would be DM-109 (100 + 4 + 5 = 109).
[Trigger address+3]	**Offset to the start address of LW or RW memory in HMI** Take OMRON PLC as an example: If set [Trigger address] to DM-100, [Trigger address + 3] will be DM-103. If the value in DM-103 is 100, the start address of memory in HMI is RW-100 or LW-100.

Example 1

We want to use [PLC Control] object to transfer 16 words data in OMRON PLC, starting from address DM-100, to the HMI address, starting from RW-200. The setting is shown below:

(a) Firstly, create a PLC Control object, set [Type of control] to [General PLC control], and set [Trigger address] to DM-10, that is, to use the four sequential registers start from DM-10 to control data transfer.



(b) Confirm the data size and the offset addresses.
Set DM-11 to 16, since the number of words to transfer is 16 words.

Set DM-12 to 86, which indicates the address of data source is DM-100 (100=10+4+86).

Set DM-13 to 200, which indicates the destination address is RW-200.

(c) Set DM-10 according to the direction of data transfer.

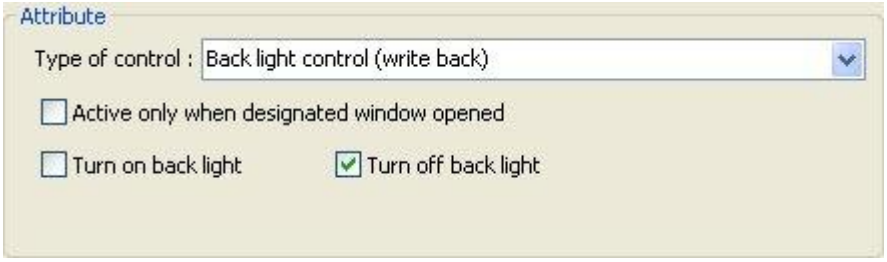
If set DM-10 to 1, the data will be transferred from PLC to HMI RW register,

If set DM-10 to 3, the data will be transferred from HMI RW register to PLC.

Setting DM-10 to 2 or 4 works the same, the difference is that the HMI memory is LW.

d. Back light control (write back)

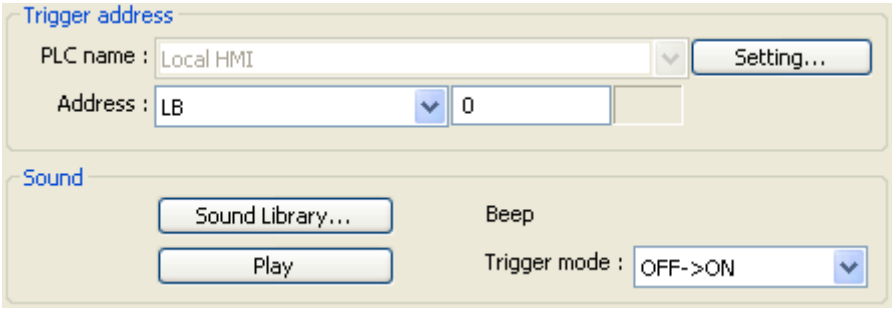
When [Trigger address] is turned ON, HMI backlight will be turned ON/OFF and [Trigger address] will be turned OFF. Any touch on the screen will turn the backlight on.



e. Back light control

When [Trigger address] is turned ON, HMI backlight will turn ON/OFF and [Trigger address] will not be changed.

f. Sound control



When the state of the designated [Trigger address] changes, the HMI will play the sound selected from the sound library.

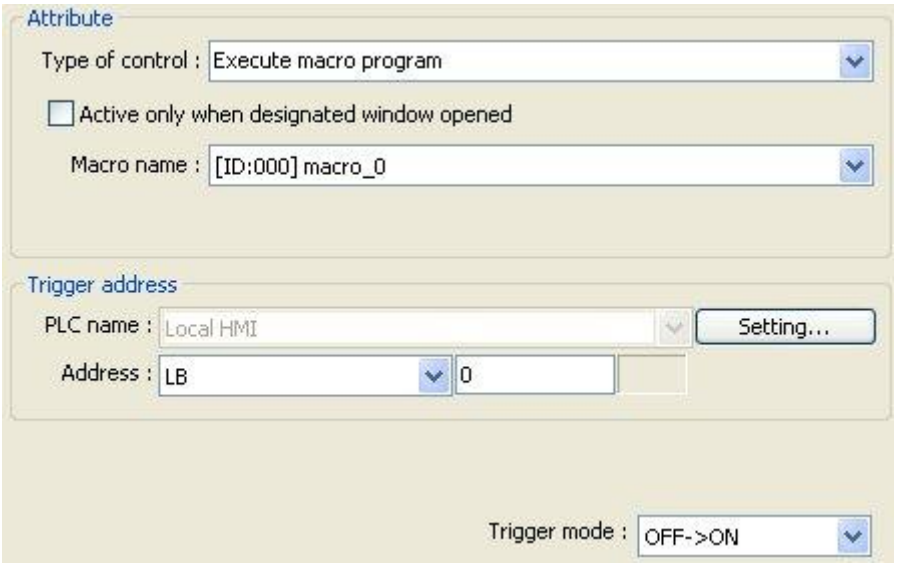
To activate the sound by [Trigger address]:

(1) Bit goes ON (OFF->ON)

(2) Bit goes OFF (ON->OFF)

(3) State change (either ON->OFF or OFF->ON)

g. Execute macro program

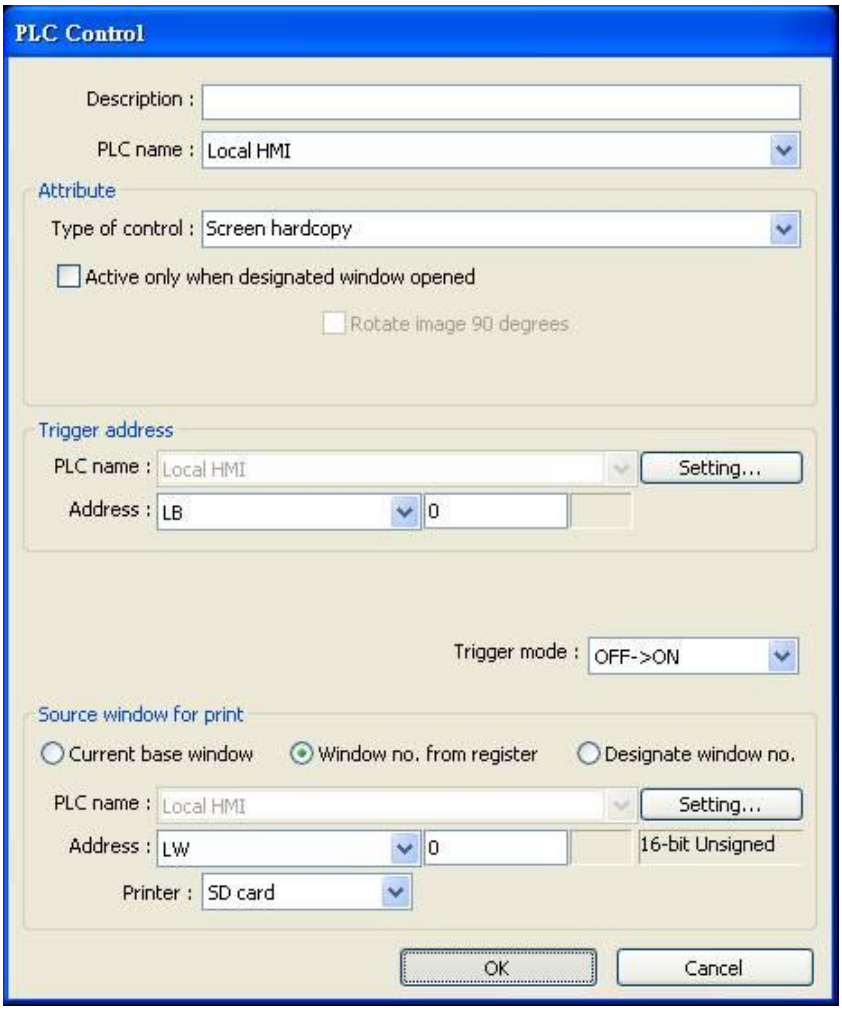


Select a pre-defined Macro from the drop-down list. When the state of the designated [Trigger address] changes, the selected Macro is executed.
The ways to execute Macro by [Trigger address]:
(1) Bit goes ON (OFF->ON)
(2) Bit goes OFF (ON->OFF)
(3) State change (either ON->OFF or OFF->ON)
(4) Execute the selected Macro when bit is ON. When the bit remains ON, Macro will be executed repeatedly.

h. Screen hardcopy

When the state of the designated [Trigger address] changes, print the selected screen.
The ways to activate screen hardcopy by [Trigger address]:
(1) Bit goes ON (OFF->ON)
(2) Bit goes OFF (ON->OFF)
(3) State change (either ON->OFF or OFF->ON)

There are three options to specify the source window for hardcopy:



[Current base window]

Print the base window at the time the operation is activated.

[Window no. from register]

Print the window designated by the value in a PLC address. If the window number is valid, the screen is printed as shown below:

[Designate window no.]

Select a base window to be printed as shown below:

[Printer]

If not specifying any printer, there are other selections such as SD card or USB disk.

The printer can be set in [System Parameter Settings] » [Model]

- A background printing procedure is performed when the printed window is not the current base window.
- If the hard-copied window is not current base window, its [Direct Window] and [Indirect Window] objects will not be printed.

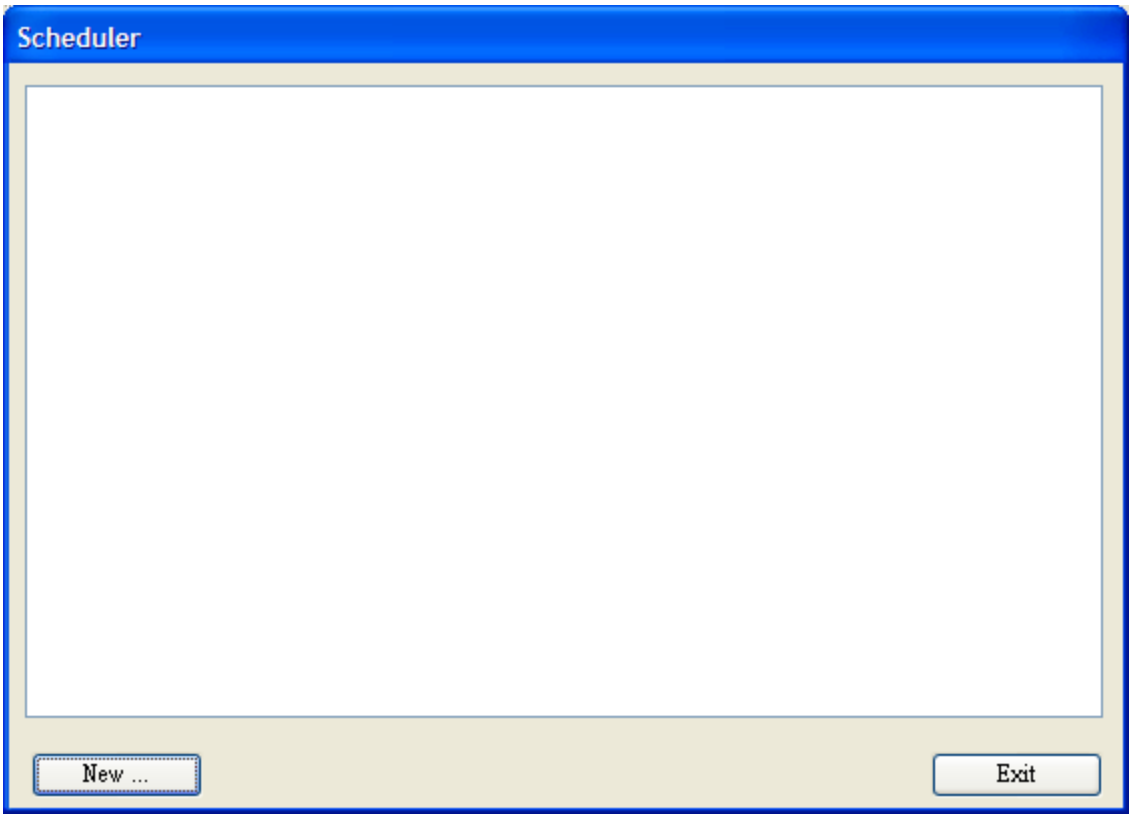
13.28 Schedule

Overview

[Scheduler] object turns bits ON/OFF, or writes values to word registers at designated start times. It works on a daily or weekly basis.

Configuration

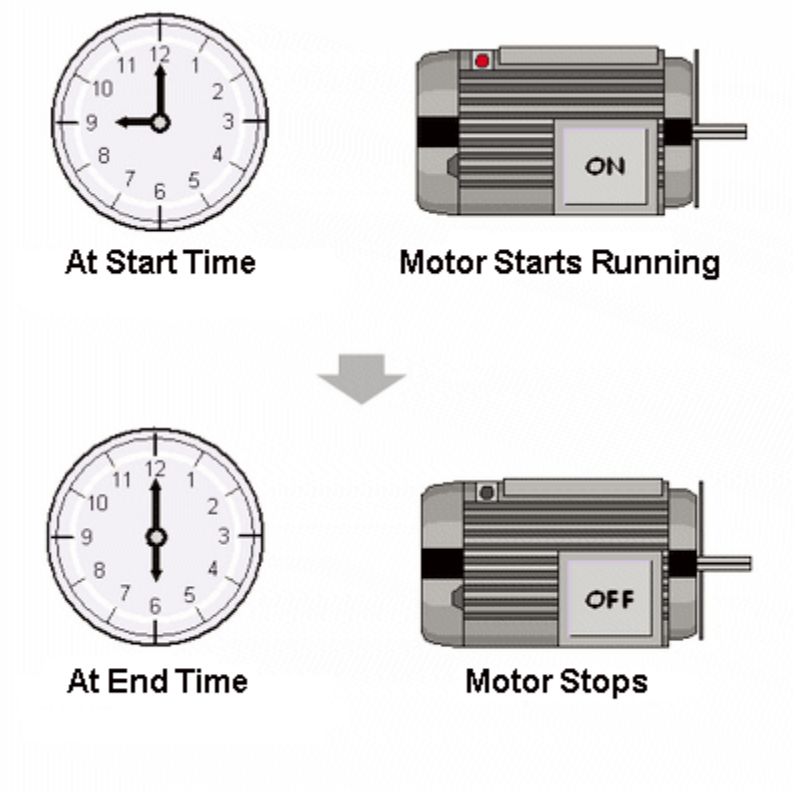
Click the [Schedule] icon on the toolbar to open the [Scheduler] management dialog. To add new [Scheduler], click [New], Set up the properties, press OK button, and a new [Scheduler] object is created.



There are two demonstrations to help you understand the usage of Schedule.

Example 1

A motor is scheduled to power - ON at 9:00 and power – OFF at 18:00, Monday to Friday. We are using LB-100 to control the motor state. LB-100 will be set ON at 9:00 and OFF at 18:00.

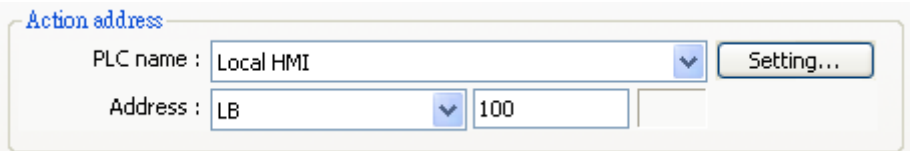


Click [Scheduler] icon on the toolbar or select [Objects] » [Scheduler] to configure the [Scheduler] object.

[General]

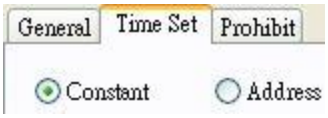
1. Select [Bit ON] in [Action mode].

2. Use LB-100 as [Action address].



[Time Set]

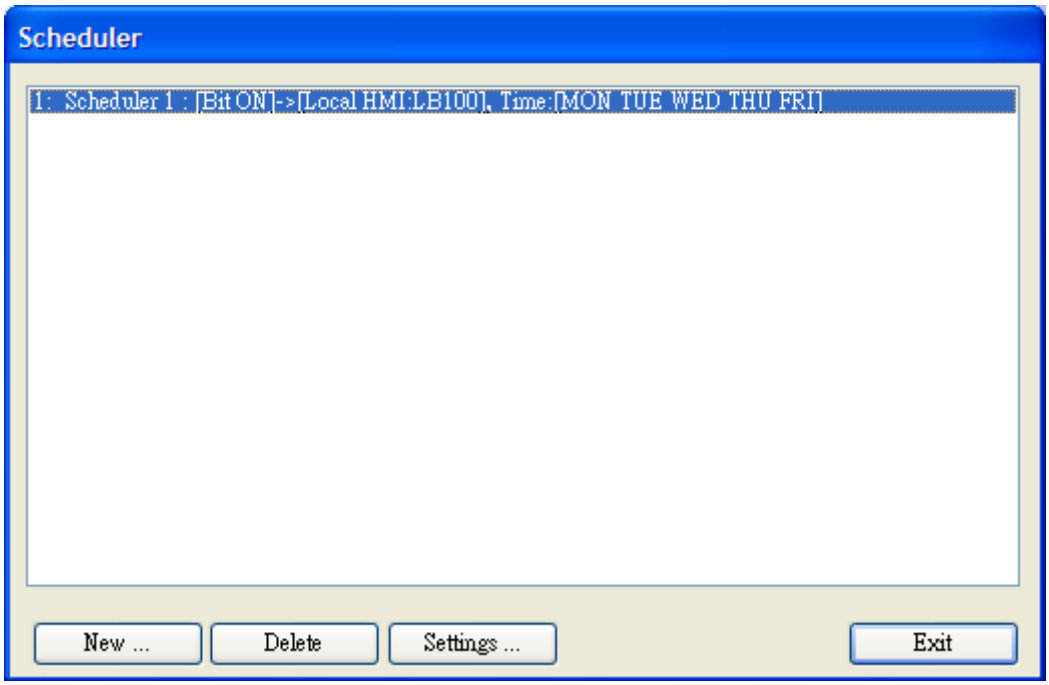
3. In [Time Set] tab, select [Constant].



4. Enter [Start] time as 9:00:00 and select Monday to Friday. Deselect [Setting on individual day].

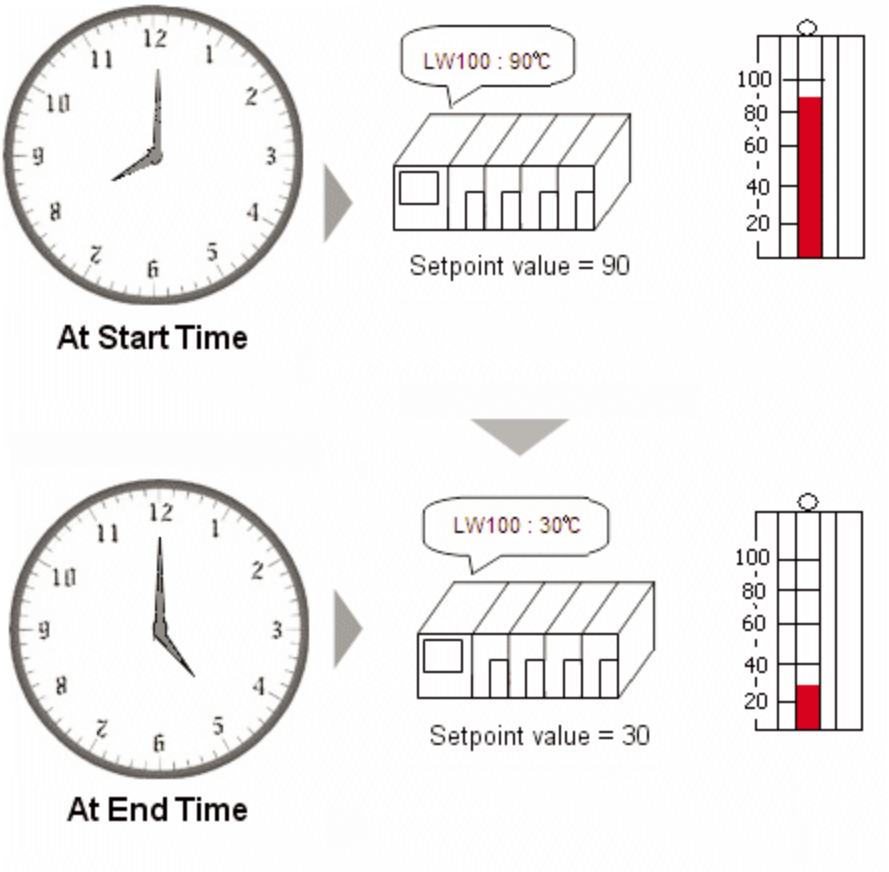
5. Enter [End] time as 18:00:00 and select [Enable termination action] check box.

6. Click [OK], a new schedule object will be created on the [Scheduler] list.

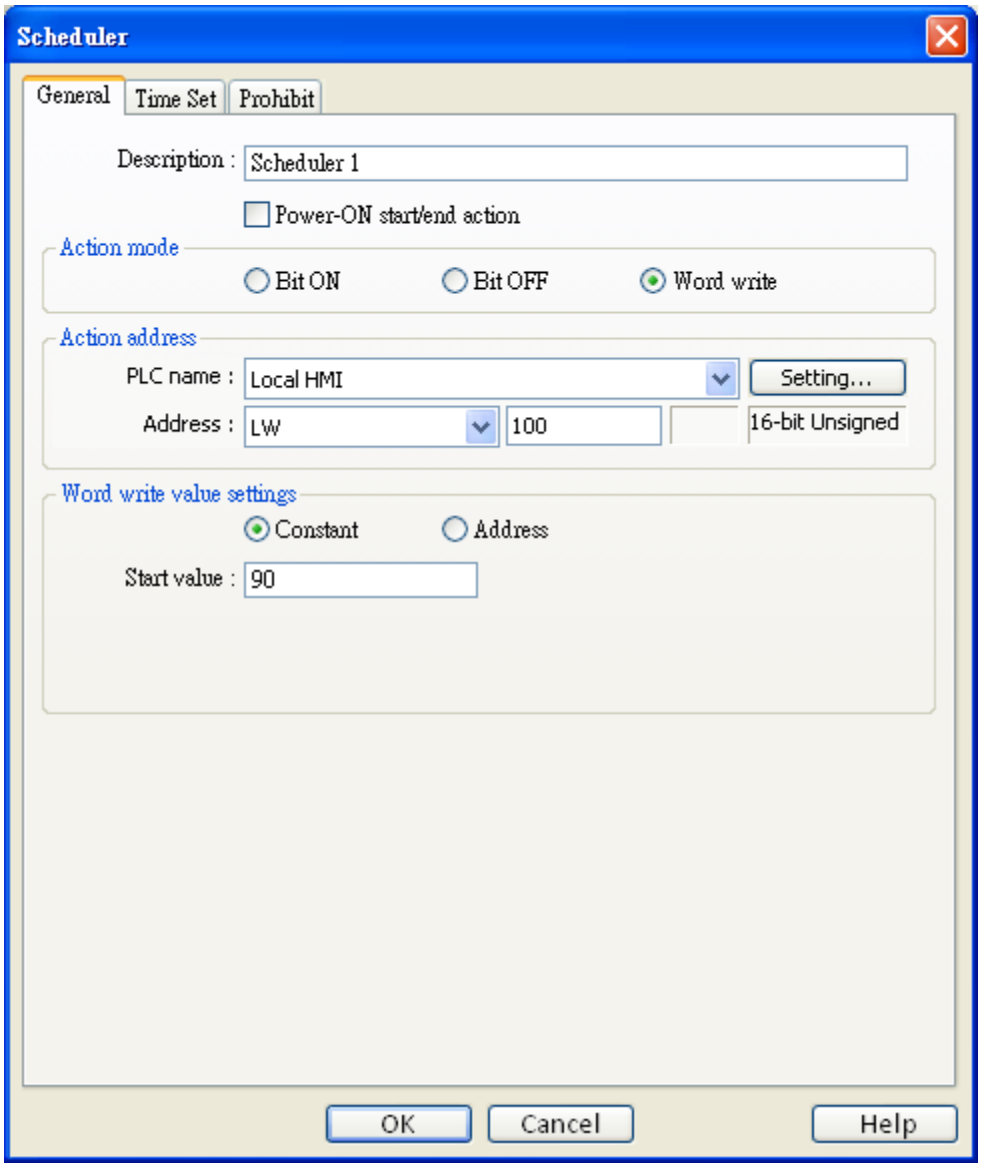


Example 2

A thermal heater is scheduled to heat up to 90°C at 08:00 and cool down to 30°C at 17:00, Monday to Friday. LW-100 is used to store the set point value.

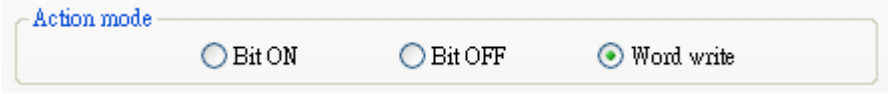


Click [Scheduler] icon on the toolbar or select [Objects] » [Scheduler] to configure the [Scheduler] object. Click [New] to add a new object.

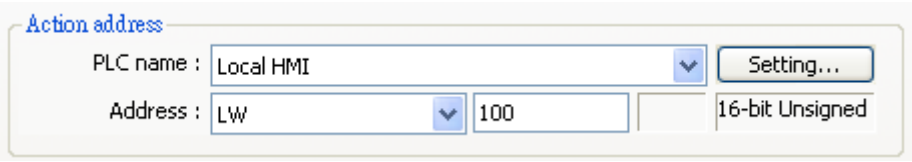


[General tab]

1. Select [Word write] in [Action mode].



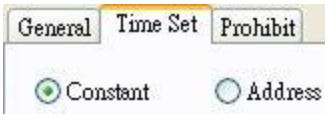
2. Set LW-100 in [Action address].



3. Select [Constant] for [Word write value settings] and enter 90 in [Start value].

[Time Set tab]

4. In [Time set] tab select [Constant].



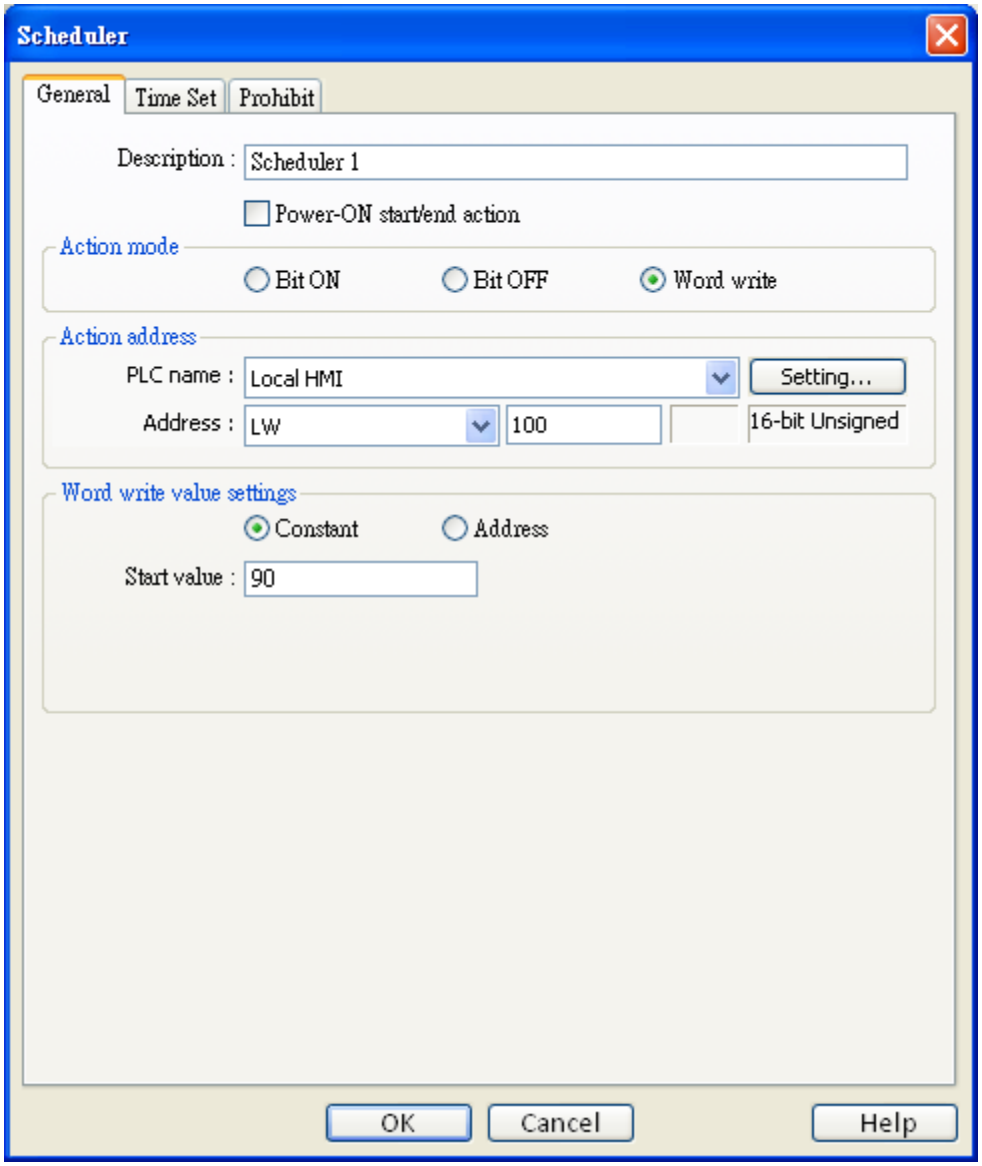
5. Enter [Start] time as 8:00:00 and select Monday to Friday. Deselect [Setting on individual day].

6. Enter [End] time as 17:00:00 and select [Enable termination action] check box.

7. Return to [General] tab and enter 30 in [Write end value].

8. Click [OK], a new schedule object will be created on the [Scheduler] list.

■ General tab



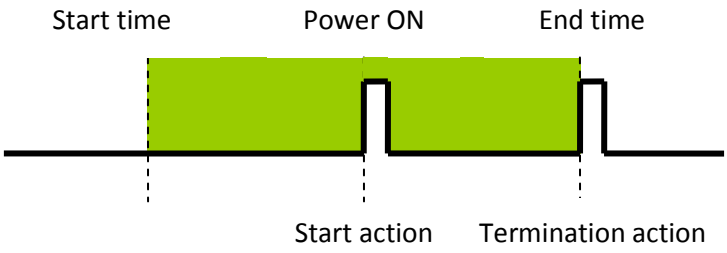
[Power-ON start/end action]

Check the condition when the HMI is powered ON.

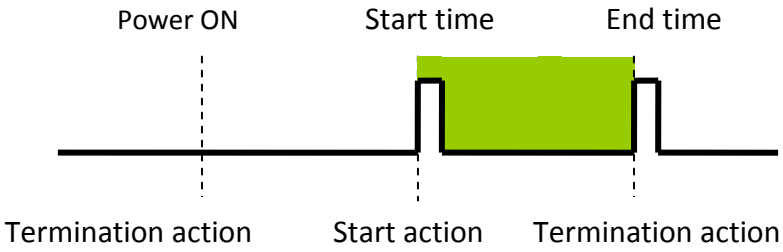
- Enabled

  When HMI is powered ON within the scheduled time range, the start action will be performed automatically. When HMI is powered ON outside the scheduled time range, the end action will be executed.

Inside scheduled range



Outside scheduled range



- Disabled
  When the HMI is powered ON at a time later than the start time, the start action will not be performed, but the end action will be performed. When the end action is not defined the scheduled range is not recognized and no action is performed.

Action Mode

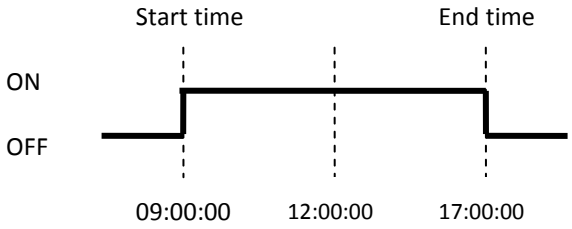
Choose the action to do at the given time.

[Bit ON]

At the start time, set the designated bit ON. At the end time, set the designated bit OFF.

Example: Start time : 09:00:00

End time : 17:00:00

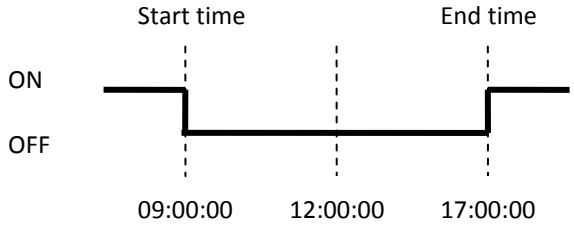


[Bit OFF]

At the start time, set the designated bit OFF. At the end time, set the designated bit ON.

Example: Start time : 09:00:00

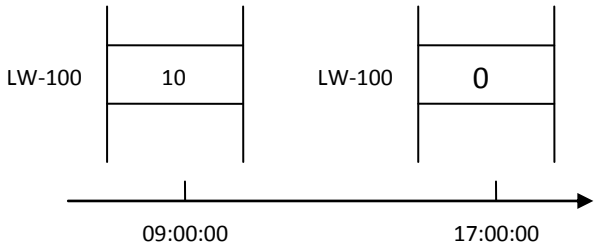
End time : 17:00:00



[Word write]

The [Write start value] entered here is transferred to the designated [Action address] word register at the start time. At end time, the [Write end value] entered here is written to the [Action address].

Example: Device address: LW100
Start time: 09:00:00
End time: 17:00:00
Write start value: 10
Write end value: 0



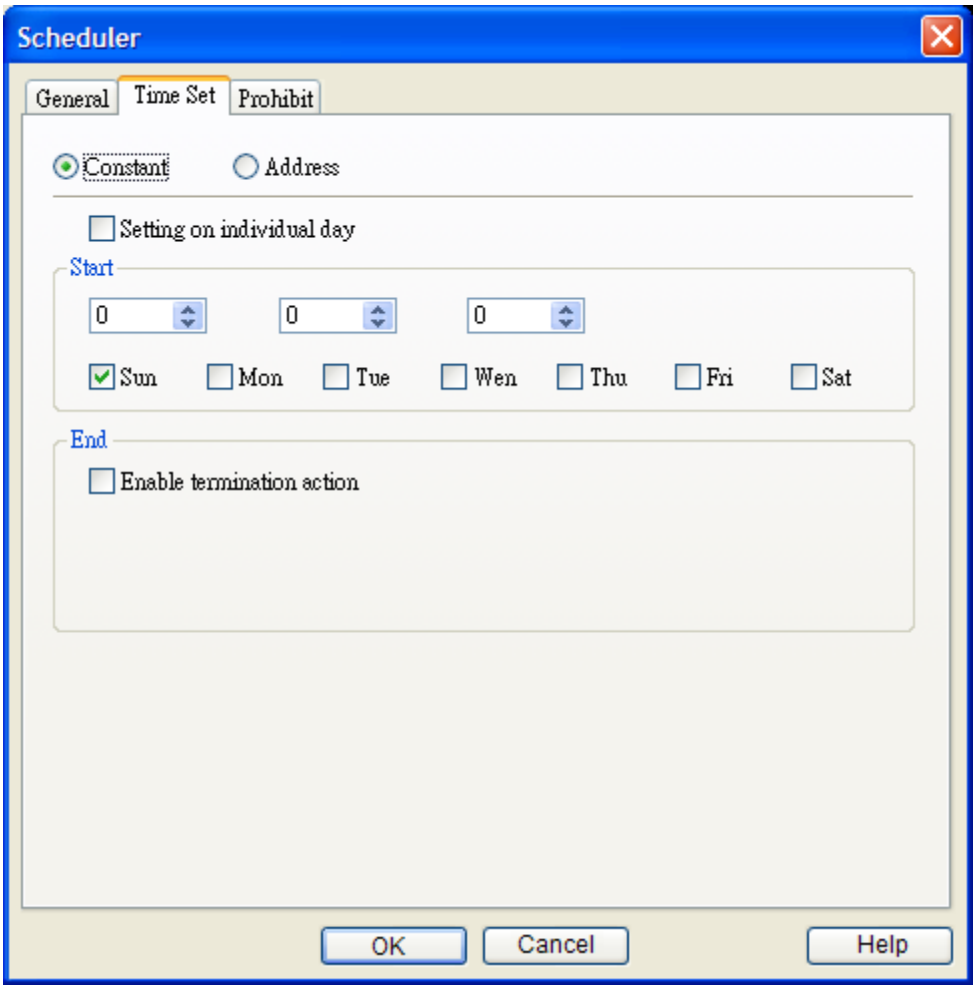
- Only if an [End time] is enabled and entered on the [Time set] tab will the [Write end value] box appear.

■ Time Set

[Constant] / [Address]

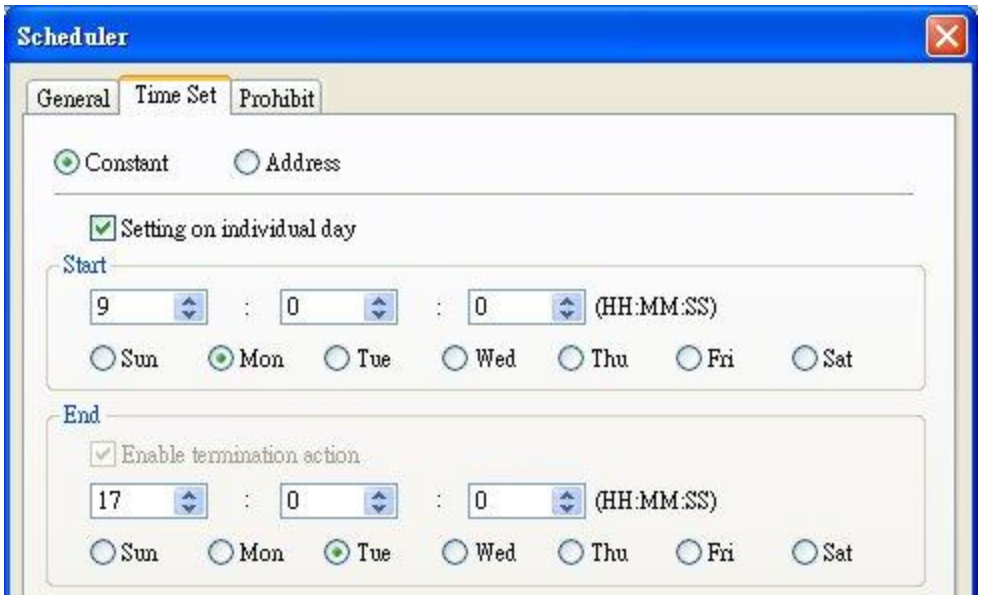
Specify start time and end time. [Constant] allows specifying a date or period and time. [Address] allows controlling the time by a designated address.

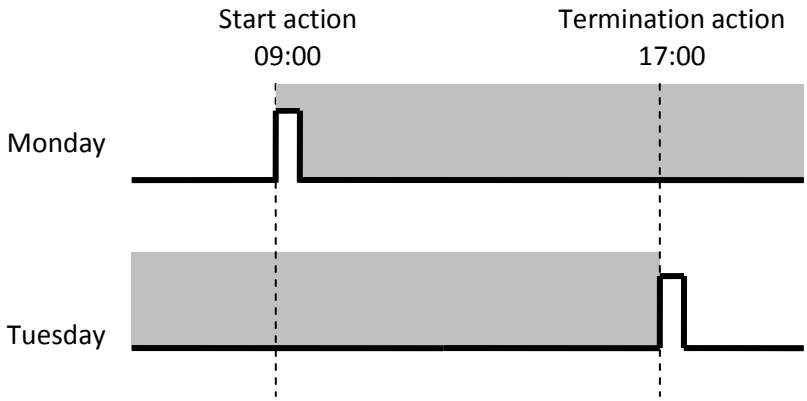
When [Constant] is selected:



[Setting on individual day]

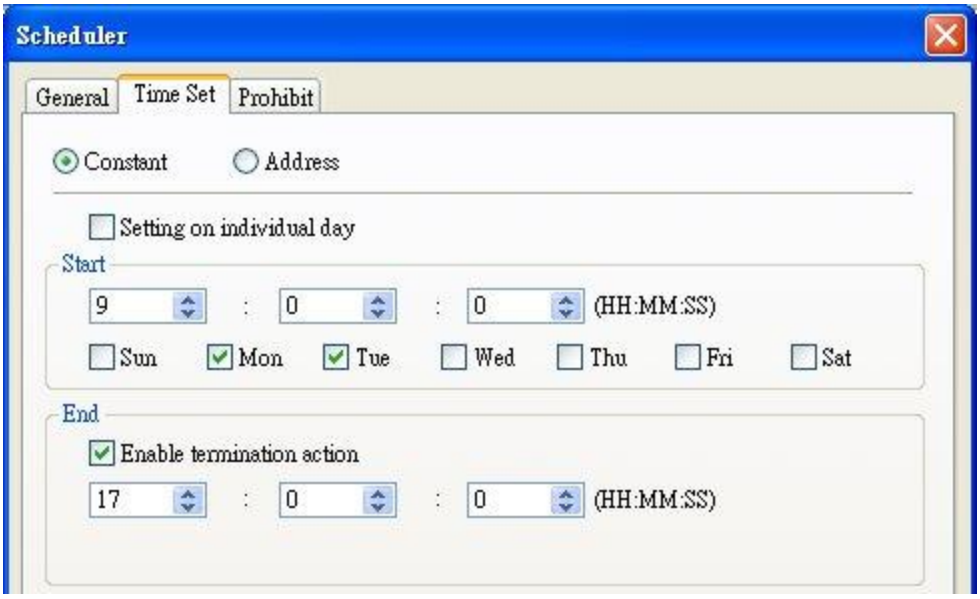
- [Setting on individual day] is selected
  Start and end times can be assigned to different days of the week.

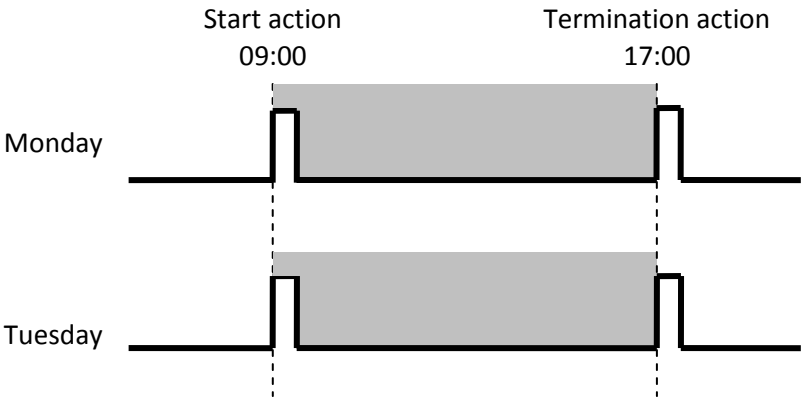




- Start and end time must be entered.
- Start and end time must be on a different time, or same time but different day.

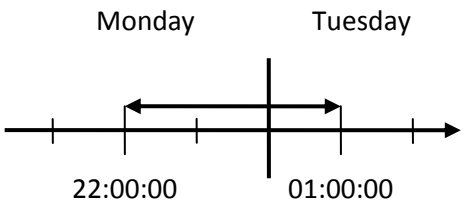
- [Setting on individual day] is not selected
  Start and end times entered must start and end within 24-hours.





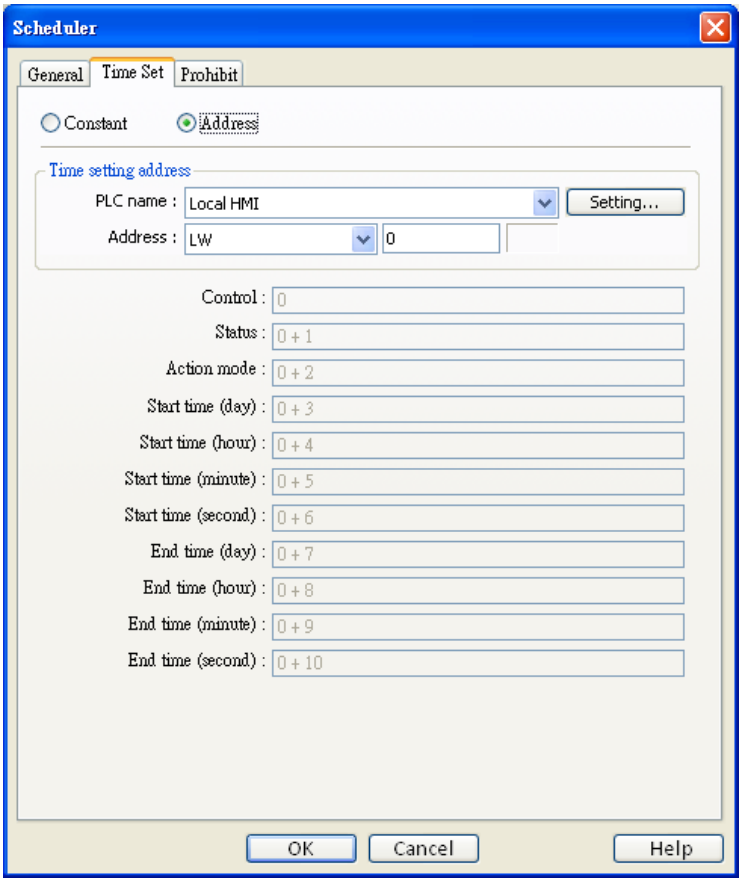
- Start and end time must be on a different time, different day.
- If an end time is earlier than a start time the end action will occur early the next day.

Start date: Monday
Start time: 22:00
End time: 01:00



When [Address] is selected:

The scheduler object retrieves the start/end time and day of week information from word registers, enabling all parameters to be set and changed under PLC or user control.



[Time setting address]

Designated as the top address in a block of 11 sequential registers which are used to store time setting data.

The format of the 11 word registers should normally be 16-unsigned integer. If a 32-bit word address is chosen, only bits 0-15 are effective, and bits 16-31 should be written as zero.

a. Control (Time setting address + 0)

Turn [Control] ON to tell the HMI to read and update [Action mode], [Start time], and [End time] values.

15	0	Bit
Reserved (0 fixed)	0	

Bit 0: no action 1: read times/action mode

- HMI will not regularly read the data from [Action mode] (address + 2) to [End time] (address + 10). Please turn [Control] ON when the settings are changed.

b. Status (Time setting address + 1)

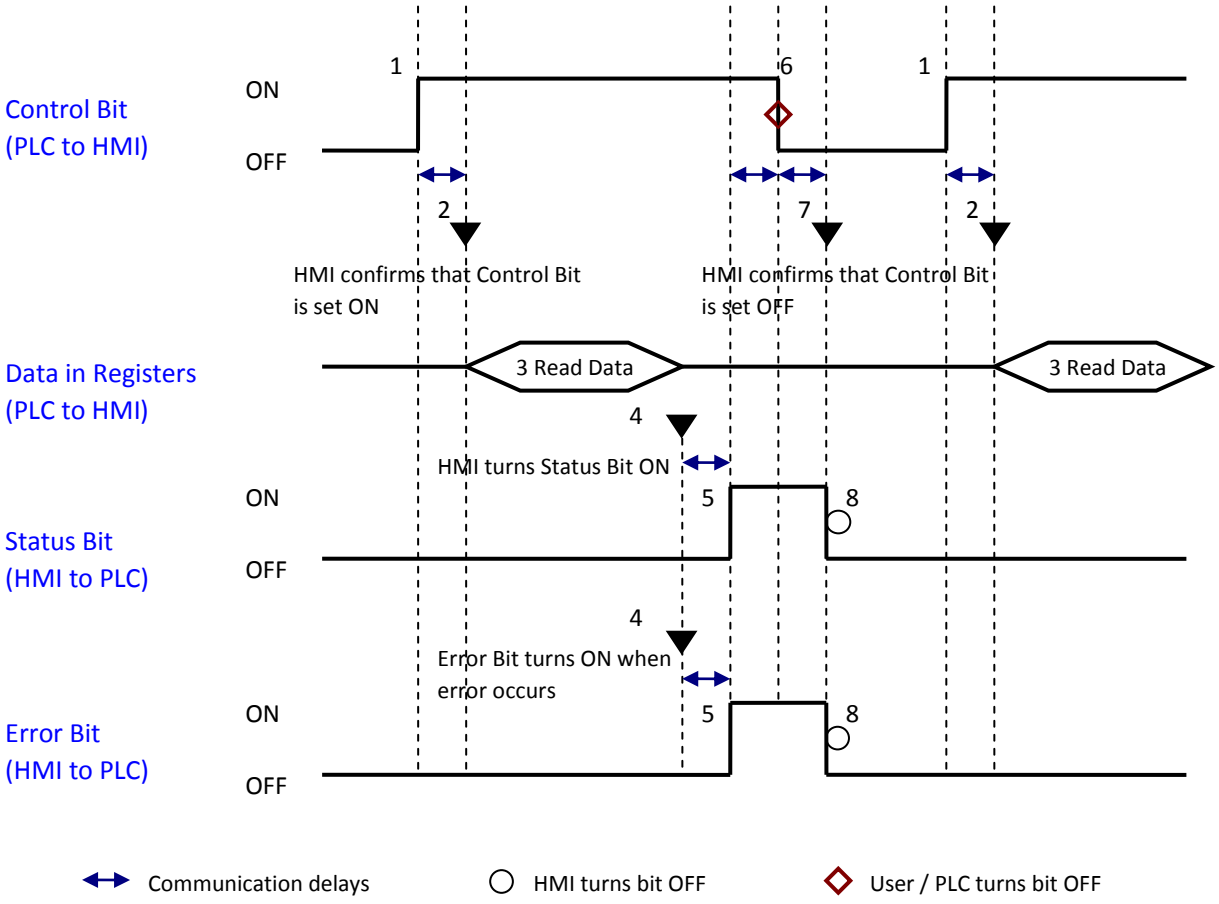
When the read operation above completes bit0 of this register truns ON. If time data read is out of range or incorrect in any way bit1 turns ON..

15 ... 02	01	00	Bit
Reserved (0 fixed)	0	0	

Bit 00: Status bit: Read operation completed. (0: reading or reading not started. 1: reading completed.)

Bit 01: Error bit: Start or end time format incorrect. (0: corrected 1: error)

- After the scheduler reads the data and the status is turned ON (The value in [Address + 1] = 01), the control bit must be turned OFF (address = 0). The status bit and error bit will be turned OFF (1→0) at the same time.



c. Action mode (Time setting address + 2)

Enable/disable [Enable termination action] and [Setting on individual day]. Whatever the [Enable termination action] bit is, all the time data, from [Control] to [End time (second)], will be read.

15 … 02	01	00	Bit
Reserved (0 fixed)	0	0	

Bit 00 Enable termination action(0: Disabled 1: Enabled)

Bit 01 Setting on individual day (0: Disabled 1: Enabled)

- If [Setting on individual day] is OFF, all 11 registers are still read but end time is ignored.
- If [Setting on individual day] is ON, make sure that all start end times are entered. If more than one start / end day bit is ON, and error will occur.

d. Start/End Day (Start Day: Time setting address + 3, End Day: Time setting address + 7)

Designates which day of week is used to trigger the start or end action.

15 ... 07	06	05	04	03	02	01	00	Bit
Reserved (0 fixed)	Sat	Fri	Thu	Wed	Tue	Mon	Sun	

Bit 00 Sunday (0: not used 1: used)

Bit 01 Monday (0: not used 1: used)

Bit 02 Tuesday (0: not used 1: used)

Bit 03 Wednesday (0: not used 1: used)

Bit 04 Thursday (0: not used 1: used)

Bit 05 Friday (0: not used 1: used)

Bit 06 Saturday (0: not used 1: used)

e. Start/End Time (Start Time: Time setting address + 4 to + 6, End Time: Time setting address + 8 to + 10)

Hour: 0 - 23

Minute: 0 - 59

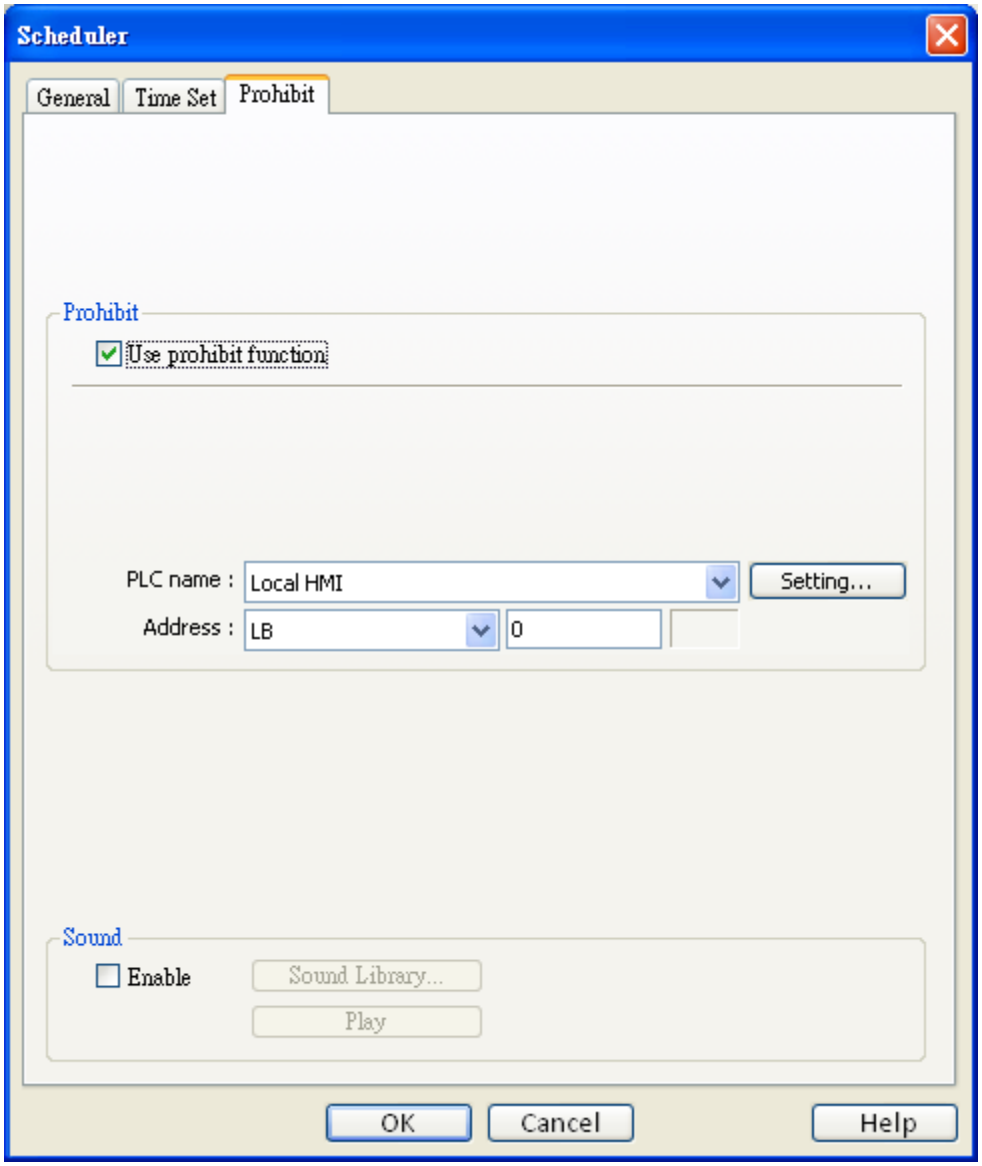
Second: 0 - 59

Values outside these ranges will set the error bit in the Status word.

- **16-bit unsigned integer** format must be used. Does not support BCD format here.
- End time depends on [Action mode] (address + 2). [Enable termination action] (bit 00) and [Setting individual day] (bit 01) are related:

Setting individual day	Enabled	Disabled	
Enable termination action	Enabled	Enabled	Disabled

■ Prohibit tab



Prohibit

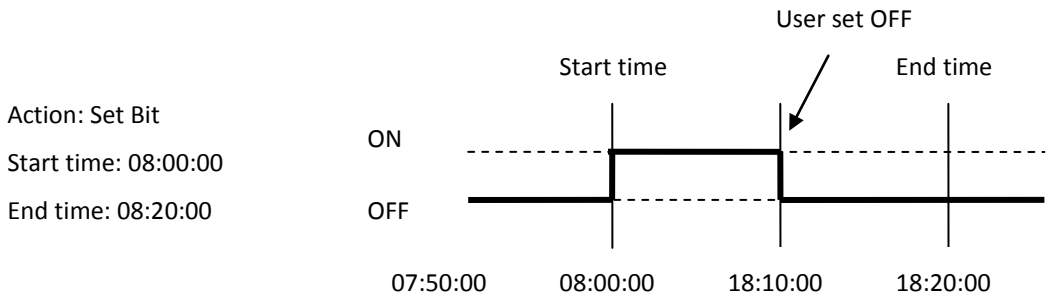
Before the scheduled action is performed, the HMI will read the specified bit state. If it is ON, the scheduled start/end action will be skipped. Otherwise, it will be performed normally.

Sound

Enable the sound output function to play the chosen sound when action.

Restrictions:

1. Up to 32 scheduler entries are allowed.
2. A time schedule applies one action only, when the start time is reached.



3. [Write start/end value] and [Prohibit] bit is read only once before start action. After that, even to change the state or [Prohibit] bit or [Write start/end value], the end action and the value written will not be affected. Also, to read data of [Write start/end value] and [Prohibit] bit, there is a delay of start action due to the communication.
4. Each time RTC data is changed, schedule list entries that possess both start and end times will be checked for in-range or out-range conditions. For in-range, the start action will occur. If the end action is not set, the new range is not recognized, the action will not occur.
5. If several schedule objects are set to the same start time or end time, the action is performed in ascending order of the schedule number.
6. In [Time Set] » [Address] mode, the system will read [Control] word regularly. The length of the period depends on the system.
7. In [Time Set] » [Address] mode, when start time and end time is out- range, error occurs in the set action time. (Note: BCD is not an acceptable format)

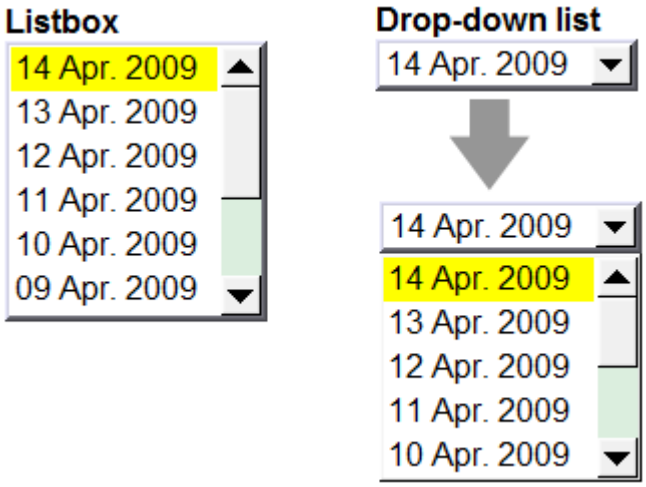
- In [Time Set] » [Address] mode, the action will not start up until the first time the time data is successfully updated.

13.29 Option List

Overview

[Option List] object displays a list of items that the user can view and select. Once the user selects an item, the corresponding value will be written to a word register.

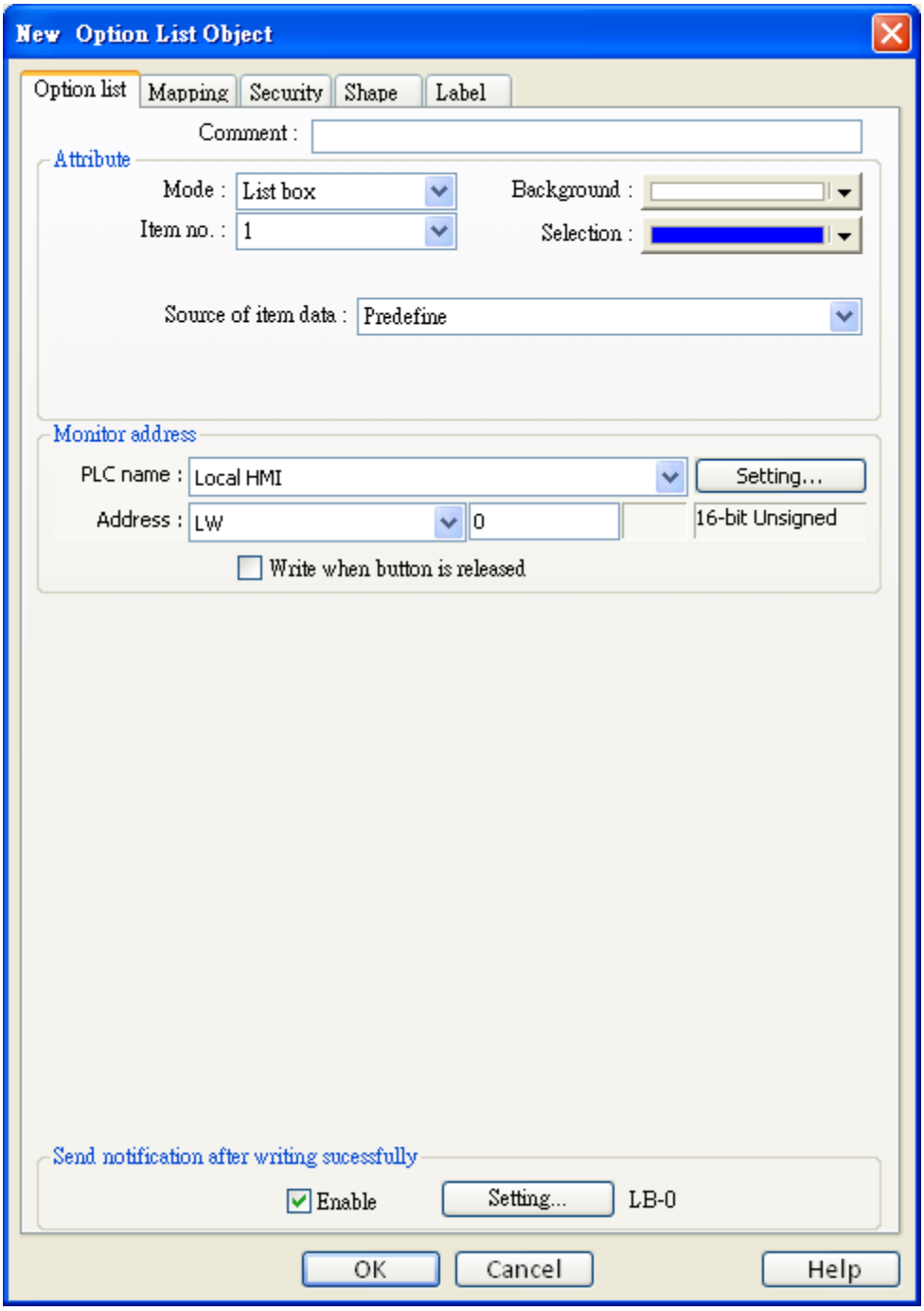
There are two forms for this object – Listbox and Drop-down list. The listbox lists all items and highlights the selected one. The drop-down list normally displays only the selected item. Once the user presses it, the system will display a listbox (which is similar to Listbox) beneath the object.



Configuration

Click the [Option List] icon on the toolbar to open a [Option List] object property dialog. Set up the properties, press OK button, and a new [Option List] object will be created.

Option list



Attribute

1. **[Mode]**
   The list style, either List box or Drop-down list.
2. **[Item no.]**
   Set the number of items for the object. Each item represents a state displayed in the list and a value to be written to the [Monitor address]
3. **[Background]**

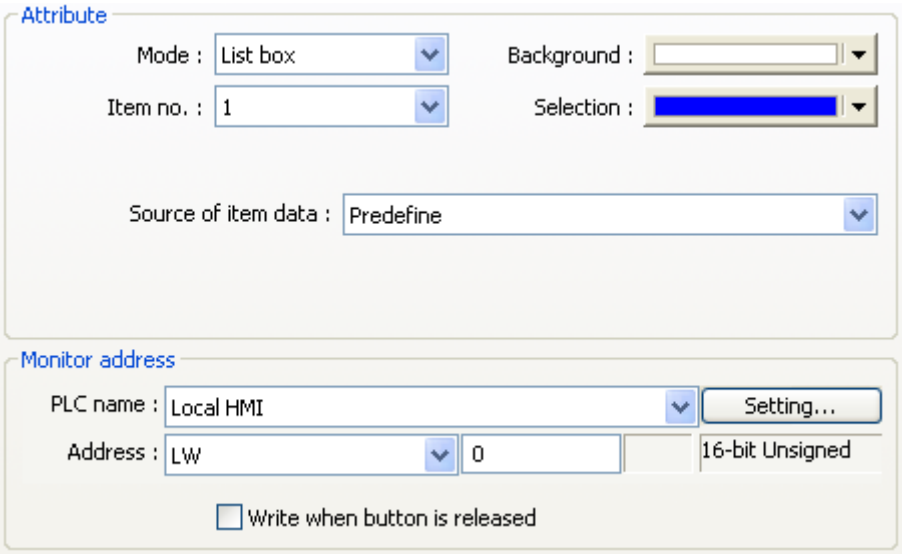
Change background color.

4. **[Selection]**
   Change background color for the selected item.
5. **[Source of item data]**
   There are 3 sources available: [Predefine], [Dates of historical data], and [Item address].
6. **[Monitor address]**
   The value of selected item will be written to [Monitor address].
   [Write when button is released]
   If this option is selected, the selected item value will be written to [Monitor address] after the button is released. (This option is only available in List Box style.)

Source of item data:

(1) [Predefine]

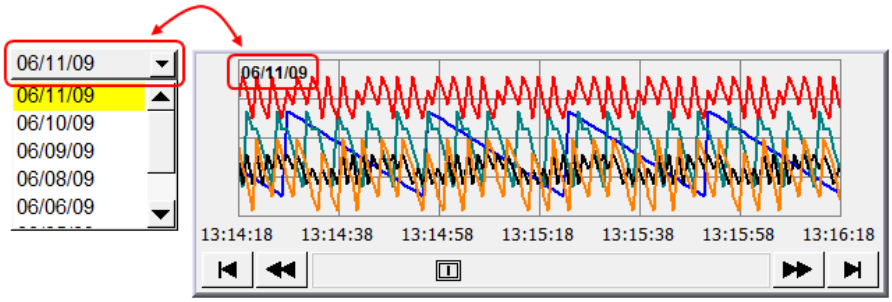
The list is manually defined in [Mapping] tab.



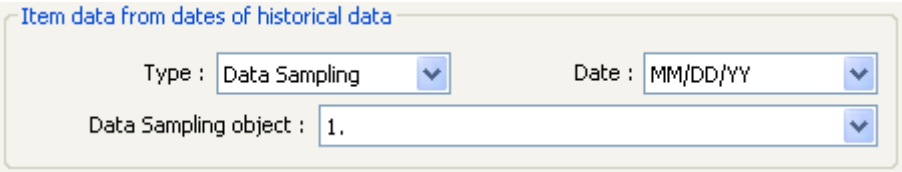
The number of items can be adjusted by **[Item no.]**, and each item represents one state. And each item has a value which will be written to **[Monitor address]**.

(2) [Dates of historical data]

[Option List] object can be used with [Trend Display], [History Data Display], [Event Display] to control which history file should be shown. The figure below is an example of [Option List] used with [History Data Display].



The available options are:



a. [Type]

Two options are available: [Event (Alarm) log], which is used for [Event Display], and [Data sampling], which is used for [Trend Display] and [History Data Display]

b. [Date]

Set the date format.

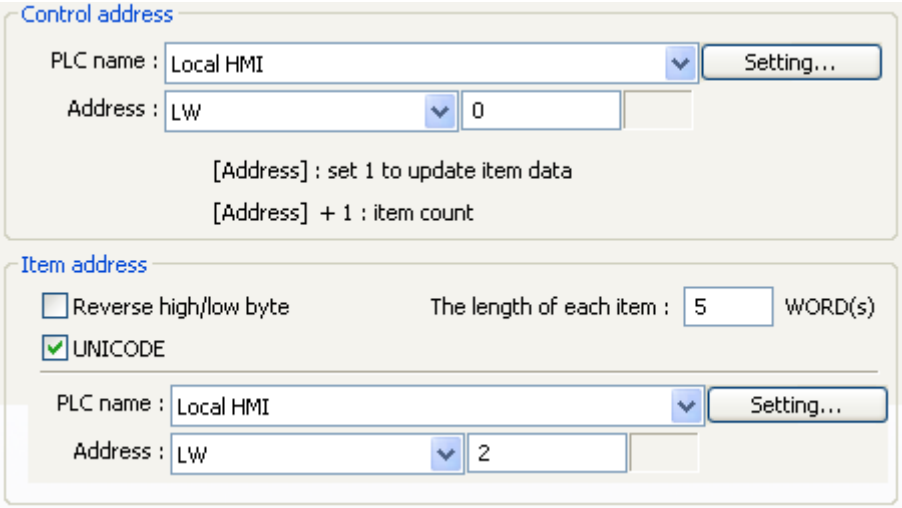
c. [Data Sampling object]

Select which [Data sampling] object is displayed when [Type] is [Data Sampling], and it should be the same as the [Data sampling object index] configured in [Trend Display] or [History Data Display].

- The system automatically disables [Mapping] tab in [Dates of historical data], and [Item address] mode.
- In [Dates of historical data] mode, When users select ”Drop-Down List” in [Attribute] and enable History Index mode, the Option List displays “?” in Error State.

(3) [Item address]

The list will be read from given [Item address] and controlled by [Control address]. The following options will be available:



a. [Control address]

It is used to update and assign the number of items.

[Address]: If the value at this address is changed to 1, the option list would be replaced by items defined at **[Item address]**. After updating, the value will be restored to 0.

[Address + 1]: Define the number of items in **[Item address]**.

b. [Item address]

Assign the item address

- **[UNICODE]**

  The item will use UNICODE characters, such as Chinese characters.

  The UNICODE characters used here should be used by other objects, so EasyBuilder will compile the needed fonts and download these fonts to HMI, and the UNICODE letters could be displayed correctly.

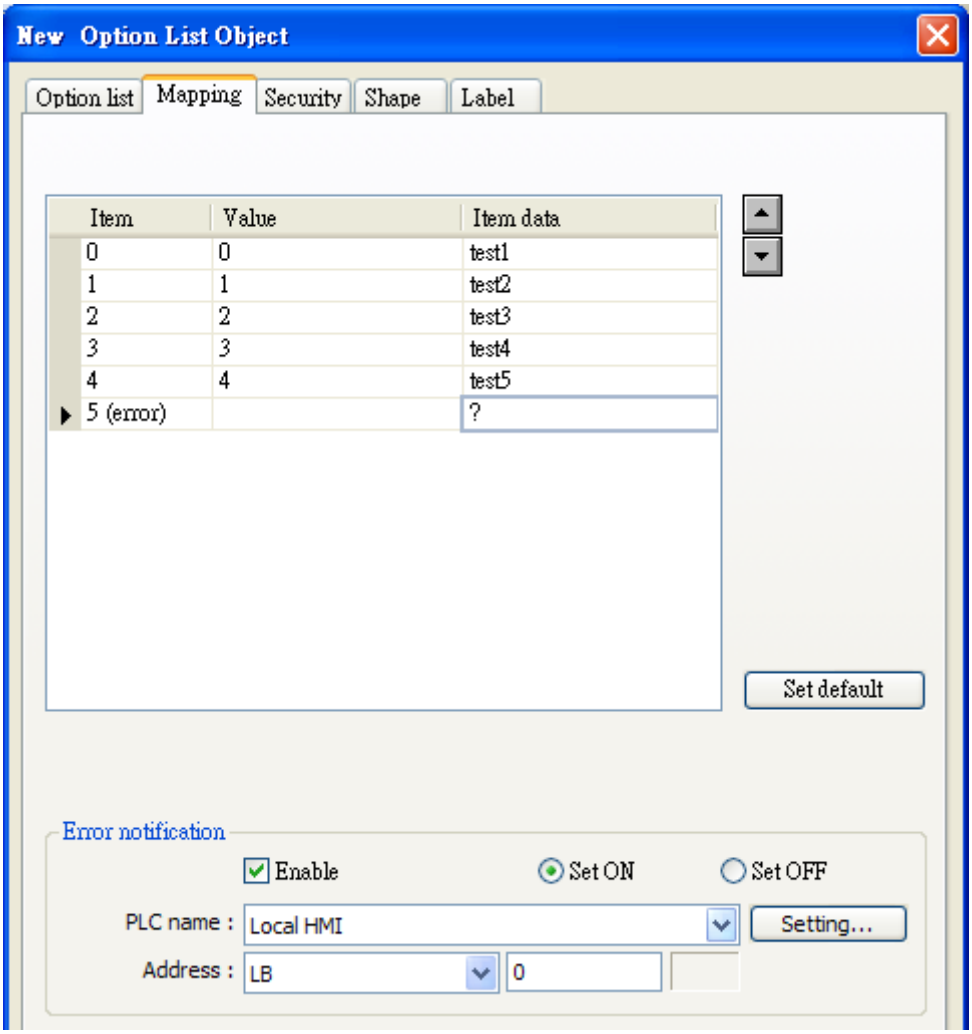
- **[The length of each item]**

  Define the number of letters for each item. The number of items multiplied by [The Length of each item] must be less than 1024 words.

Send notification after writing successfully

If the [Enable] check box is selected, it will notify a designated bit address (setting ON or OFF) after the command is successfully executed. Click [Setting] to select the [PLC name], [Address], [Device type], [System tag], [Index register] of the bit device that controls the object, or configure in [General] tab.

Mapping



Mapping table

This table displays all available states/items, their item data and values. To change the number of available items, please refer to [Option list tab] » [Attribute] » [Item no.].

[Item]

The system lists all available items. Each item represents a state that will be displayed in the list. This field is read-only.

[Value]

Here user can assign value for each item, basing on the following two criteria:

1. [For reading]: If the content from [Monitor address] is changed, the object compares the content with these values and selects first-matched item. If no item is matched, the status goes to error state and signals the notification bit register (if requested).
2. [For writing]: The system writes this value to [Monitor address] when user selects an item.

[Item data]

Text displayed for each item. The option list object displays the text of all items in the list for users to review and select.

[Error state]

On error state, the listbox-style option list removes the highlight to represent no item is selected and the drop-down list displays the data of error state.

Only the drop-down-style lists use Error state. Listbox-style lists are not able to use Error state.

For example, item 8 is the error state when specifying 8 in [Item no.]. Similarly, if you set Item no.] to 11 then state 11 would be the error state, and so on. (The states are counts from state 0)

[Set default]

Reset all values or states to default value, i.e. set 0 for item 0, 1 for item 1, and so on.

Error Notification

The system will set ON/OFF to the specified bit register when error is detected. The signal of the bit register could be used to trigger a procedure for correcting the error.

13.30 Timer

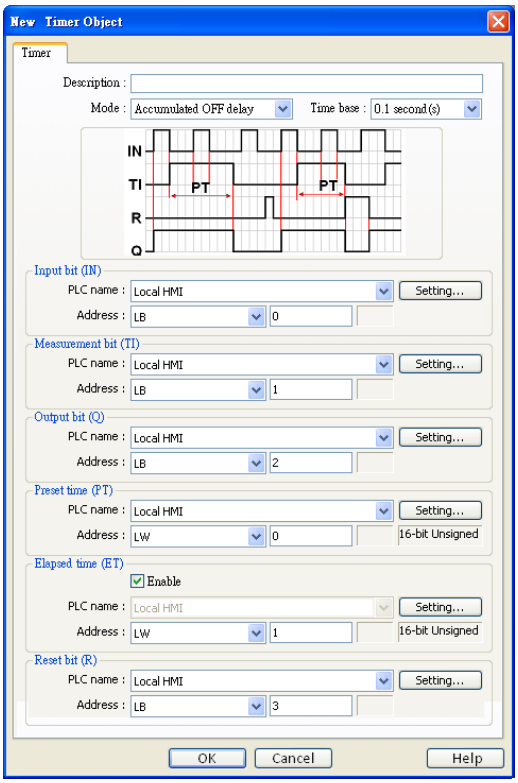
Overview

[Timer] is a time switch, and could be a Delay-On/Off, Pulse, and Cumulative On/Off switch. [Time] object uses the following 6 variables:

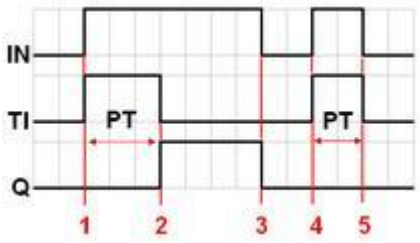
Timer Variable	Variables Type	Description
Input bit (IN)	Bit type	The main switch of timer.
Measurement bit (TI)	Bit type	Turn ON when the timer begin counting.
Output bit (Q)	Bit type	Activate when the timer finish counting.
Preset time (PT)	Word type	Set the timer value.
Elapsed time (ET)	Word type	Display current elapsed value of timer.
Reset bit (R)	Bit type	Reset the elapsed time (ET) to 0.

Configuration

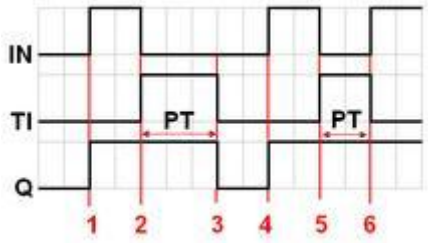
Click the **[Timer]** icon on the toolbar to open [Timer] object property dialog. Set up the properties, press OK button, and a new [Timer] object will be created.



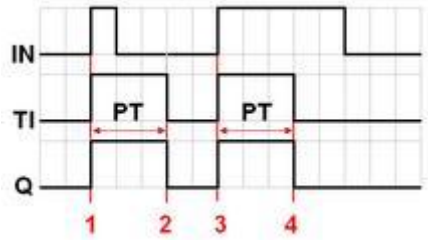
On Delay switch

Mode	Description
	• Input bit (IN): Main switch of [Timer] • Measurement bit (TI) • Output bit (Q) • Preset time (PT) • Elapsed time (ET)
Description (See the figure above)	
Period 1: When the IN is turned ON, TI is turned ON and the elapsed time ET starts counting. The Q remains OFF. **Period 2**: When the ET equals the PT, the Q be turned ON and the TI be turned OFF. **Period 3**: When the IN turns OFF, the Q be turned OFF and the ET reset to 0. **Period 4**: When the IN turns ON, the TI be turned ON and the elapsed time ET starts counting. **Period 5**: Turn IN OFF before the ET reaches the PT, the TI would be turned OFF, and the ET reset to 0, the Q remains OFF.	

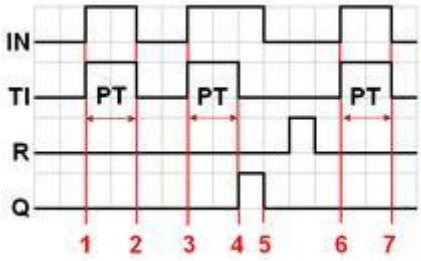
■ **Off Delay switch**

Mode	Description
	• Input bit (IN): Main switch of [Timer] • Measurement bit (TI) • Output bit (Q) • Preset time (PT) • Elapsed time (ET)
Description (See the figure above)	
Period 1: When the IN turns ON, the TI remains OFF and the Q be turned ON, and the ET reset to 0. **Period 2**: When the IN turns OFF, the TI be turned ON and the elapsed time ET starts counting, the Q remains ON. **Period 3**: When the ET equals the PT, the Q and TI are turned OFF. **Period 4**: When the IN turns ON, the Q be turned ON and the ET reset to 0. **Period 5**: When the IN turns OFF, the TI be turned ON and the elapsed time ET starts counting, the Q remains ON. **Period 6**: Turn the IN to ON before the ET reaches the PT, the TI be turned OFF, and the ET reset to 0, the Q remains ON.	

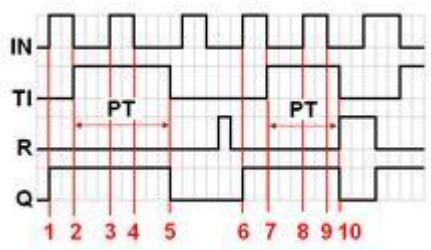
■ Pulse switch

Mode	Description
	• Input bit (IN): Main switch of [Timer] • Measurement bit (TI) • Output bit (Q) • Preset time (PT) • Elapsed time (ET)
Description (See the figure aboe)	
Period 1: When the IN turns ON, the TI and Q are turned ON, and the elapsed time ET starts counting. **Period 2**: When the ET equals PT, the TI and Q are turned OFF. **Period 3**: When the IN turns ON, the TI and Q are turned ON, and the elapsed time ET starts counting. **Period 4**: When the ET equals the PT, the TI and Q are turned OFF.	

■ Accumulated On delay

Mode	Description
	• Input bit (IN): Main switch of [Timer] • Measurement bit (TI) • Output bit (Q) • Preset time (PT) • Elapsed time (ET) • Reset bit (R): Reset ET to 0
Description (See the figure above)	
Period 1: When the IN turns ON, the TI be turned ON and the elapsed time ET starts counting, the Q remains OFF. **Period 2**: When the IN turns OFF, and if the ET is less than the PT, the TI be turned OFF. The ET is in the retentive state. **Period 3**: When the IN turns ON, the TI be turned ON. The timer measurement starts again and the ET starts counting from the kept value. The Q remains OFF. **Period 4**: When the ET reaches the PT, the TI be turned OFF and the Q be turned ON. **Period 5**: When the IN turns OFF, the Q be turned OFF. (Reset the ET to 0 by using Reset bit (R).)	

■ Accumulated Off delay

Mode	Description
	• Input bit (IN): Main switch of [Timer] • Measurement bit (TI) • Output bit (Q) • Preset time (PT) • Elapsed time (ET) • Reset bit (R): Reset ET to 0

Description (See the figure above)

Period 1: When the IN turns ON, the Q be turned ON and TI remains OFF.

Period 2: When the IN turns OFF, the TI be turned ON and the elapsed time ET starts counting, the Q remains ON.

Period 3: When the IN turns ON, the TI and Q remains ON, the timer measurement ET pauses.

Period 4: When the IN turns OFF, the paused timer measurement ET continues.

Period 5: When the ET equals the PT, the TI and Q are turned OFF. (Reset the ET to 0 by using Reset bit (R).)

13.31 Video In

Overview

Specific HMI models provide the video input function. Users can install surveillance camera, then monitor the factory any time they want. The video images can also be stored in devices and analyzed on PC.

This function can be utilized in different aspects. Apart from monitoring factory, it can also be used in mobile vehicles or building automation monitoring.

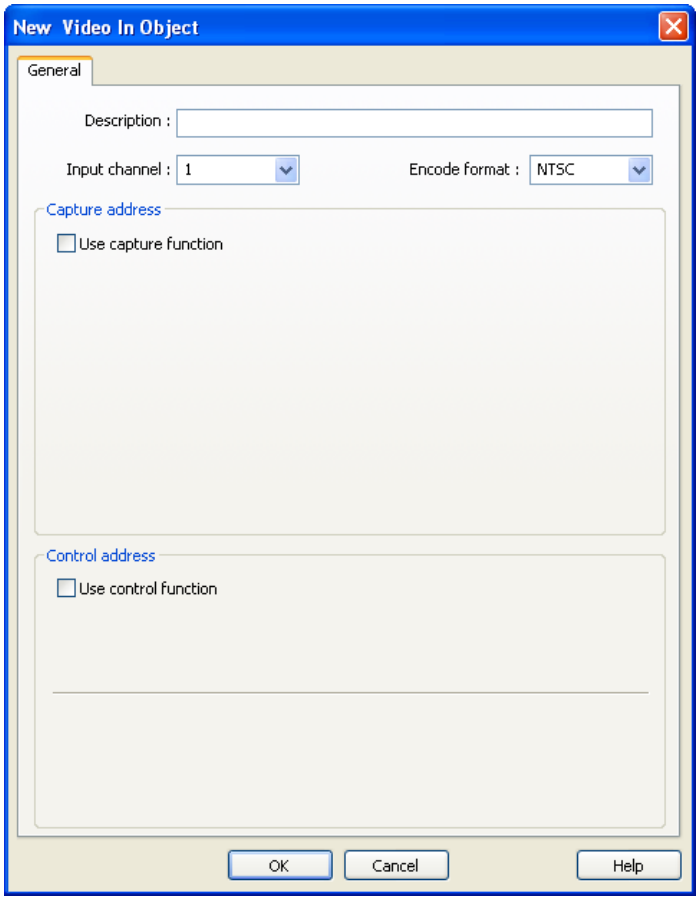
For hardware, HMI provides 2 channels for video input. Users can switch the monitored channel, and capture images without being influenced by playback function, such as pause. The captured images are still the real-time external video images.

Supported formats are NTSC and PAL.

Note: For MT-series, only MT8000X-series supports video inputs.

Configuration

Click [Video Input] icon on the toolbar to open a [Video Input] object property dialog. Set up the properties, press OK button, and a new [Video Input] object will be created.



[Input Channel]

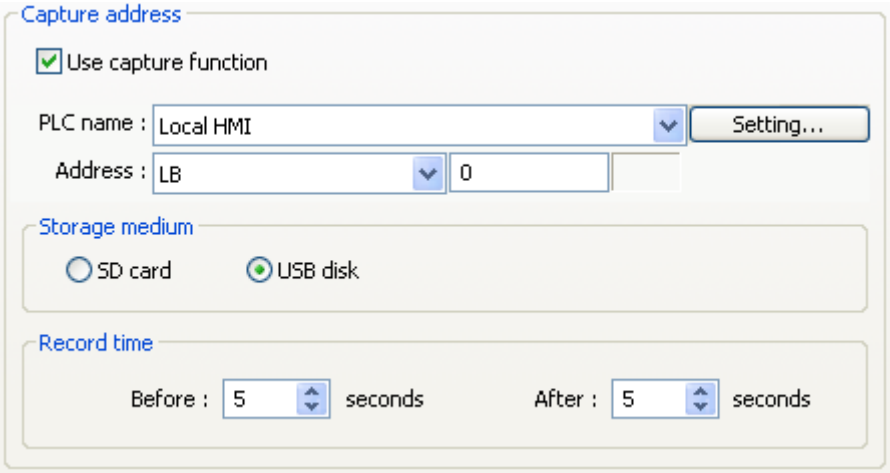
Video Input 1 or Video Input 2 can be selected.

[Encode format]

NTSC or PAL can be selected.

Use Capture address

Enable [Use capture function] to capture the image of the input video

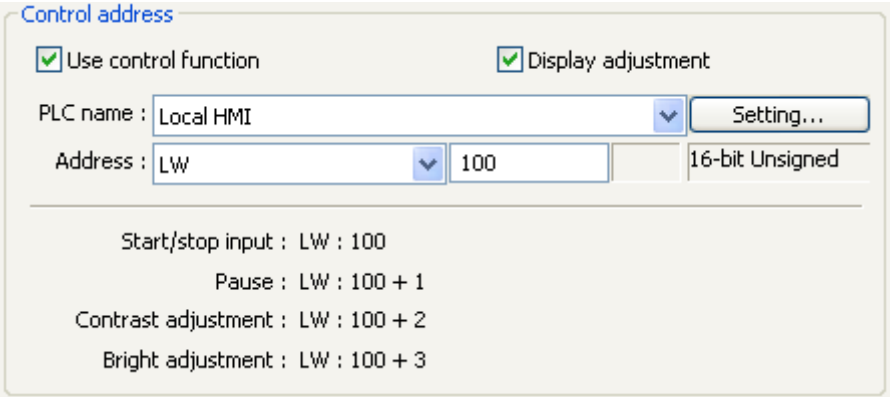


1. **[Capture address]**
   Configure the address that triggers image capturing of the video.
2. **[Storage medium]**
   Select the storage medium to save captured images, either SD card or USB disk.
   - Images of video channel 1 will be saved in directory “VIP1” in the chosen storage and images of video channel 2 in directory “VIP2”.
3. **[Record time]**
   Set a period of time for image capturing.
   - The longest period can be set starts from 10 seconds before triggering **[Capture address]** to 10 seconds after triggering. In this case there will be 21 images captured, including the one captured at the triggering moment.
   - The time interval for capturing is once in each second.
   - The captured .jpg file will be named in the following format:
     Before or after [Capture address] is triggered: YYYYMMDDhhmmss.jpg
     The moment that [Capture address] is triggered: YYYYMMDDhhmmss@.jpg

The figure above as an example, set **[Record time]** “Before” and “After” to “5” seconds, when **[Capture address]** changes from OFF to ON, system will start to capture one image per second, from 5 seconds before the triggering time to 5 seconds after the triggering time.

Use Control address

Select [Use control function] to control the video input.



Suppose **[Control Address]** is designated as “LW100”:

A. Users can set [Control Address+ 0] to enable/stop Video Input function.
   [LW100] = 0 → Stop Playing.
   [LW100] = 1 → Open video input channel 1 and display it on screen.
   [LW100] = 2 → Open video input channel 2 and display it on screen.
   [LW100] = 3 → Open video input channel 1 but don’t display it on screen. Users can still execute Capture image.
   [LW100] = 4 → Open video input channel 2 but don’t display it on screen. Users can still execute Capture image.
B. Users can set [Control Address +1] to stop or continue playing video:
   [LW101] = 1 → Pause/Continue playing.
C. If users change the value in [Control Address + 0], the system will keep the new value.
D. If users change the value in [Control Address + 1], system will first execute the corresponding command and then erase the new value and set it back to “0”.
E. If **[Use control function]** is not selected, system will play the channel set in **[Input channel]**.

[Display adjustment]

If it is selected, the screen brightness and contrast ratio can be adjusted. If specify “LW100” as the control address:

A. Adjust Contrast Ratio [Control Address + 2]: LW102, range: 1~100.
B. Adjust Brightness [Control Address + 3]: LW103, range: 1~100.

1. [Video In] object can only be used MT8000X-series.
2. Only 1 video input channel can be opened at a time.
3. Capture function will not be affected by using “pause” function. The captured images will

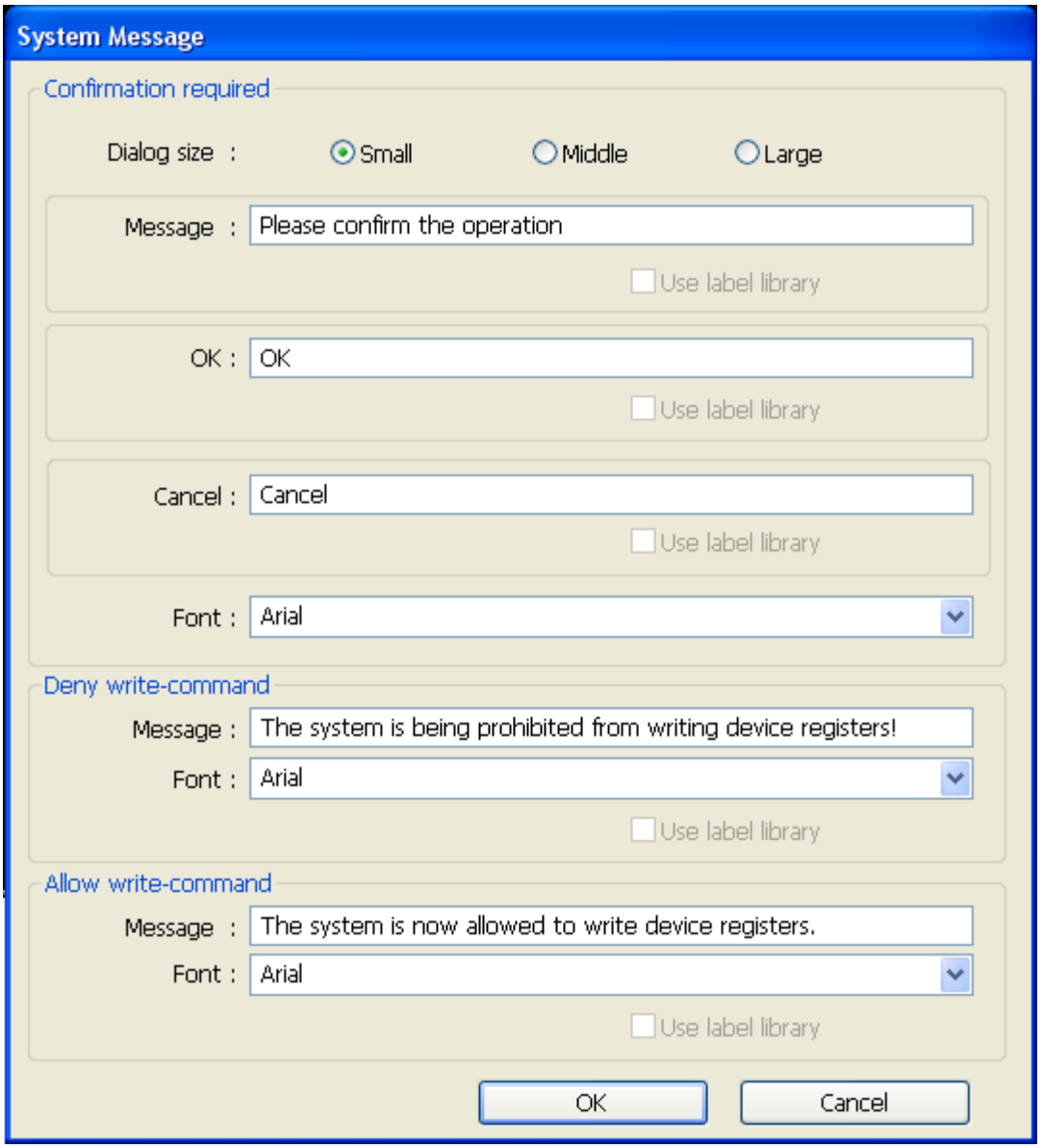
still be the real-time video input images.

4. Recommended Format and Resolution:

	1:1	50%
NTSC	720 x 480	360 x 240
PAL	720 x 576	360 x 288

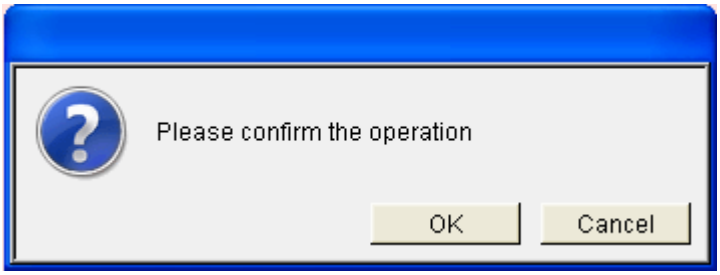
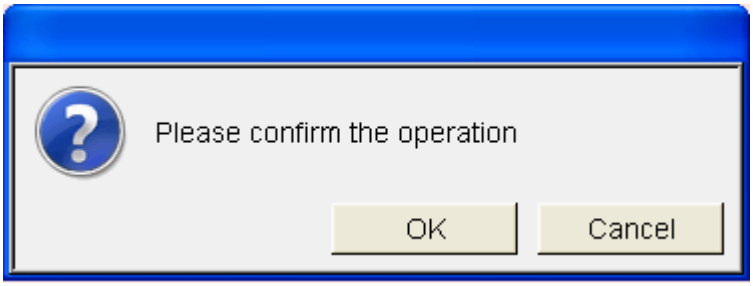
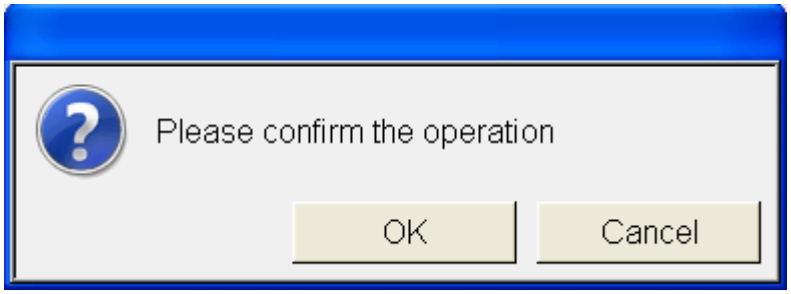
13.32 System Message

If objects use [Display confirmation request] or [local HMI supports monitor function only] is turned on/off, the corresponding messages configured here will be displayed in popup message boxes.



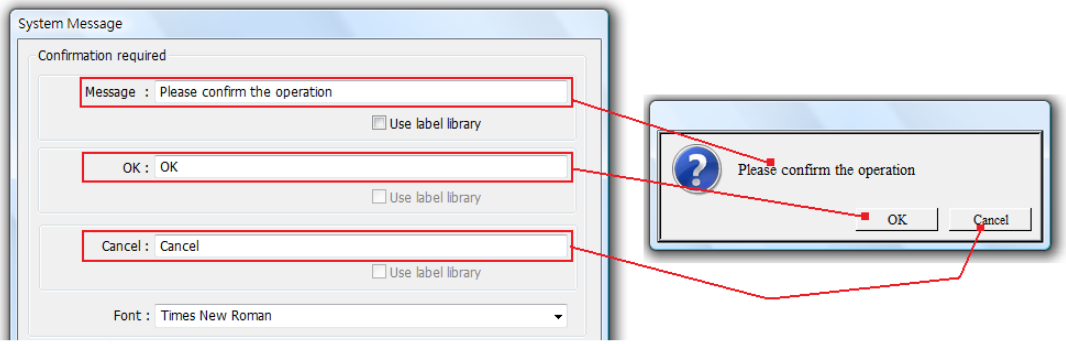
Dialog Size

Select the size for pop-up window and texts

Size	Reference
Small	
Middle	
Large	

Confirmation required

If an object uses [Display confirmation request], this message would pop up when the object is used. [Message] shown on confirmation dialog, and the text label of the 2 buttons, [OK] and [Cancel], can be set. Please use the same font for the labels of [Message], [OK] and [Cancel]. Additionally, only when selecting [Label Library] for [Message], the use of Label Library for [OK] and [Cancel] buttons can be enabled.



Deny write-command

Display when system tag LB-9196 (local HMI supports monitor function only) is turned ON.

Allow write-command

Display when system tag LB-9196 (local HMI supports monitor function only) is turned OFF.

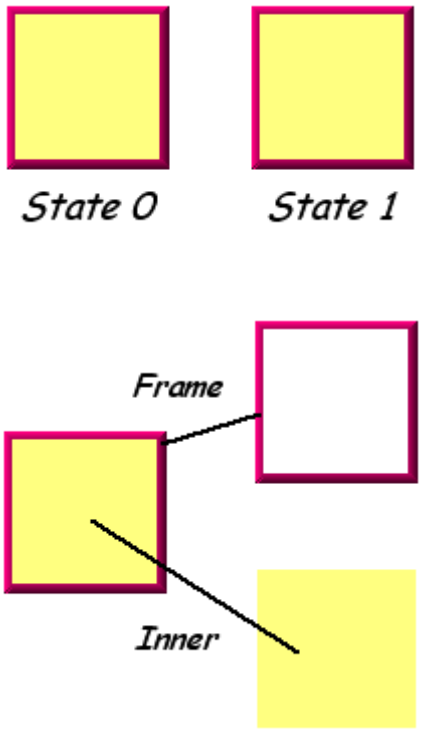
Chapter 14 Shape Library and Picture Library

EasyBuilder provides Shape Library and Picture Library to add visual effects on objects. Each Shape and Picture includes up to 256 states. This chapter explains how to create Shape Library and Picture Library.

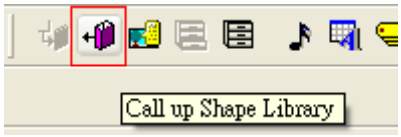
For more details, please refer to “Chapter 9 Object General Properties”.

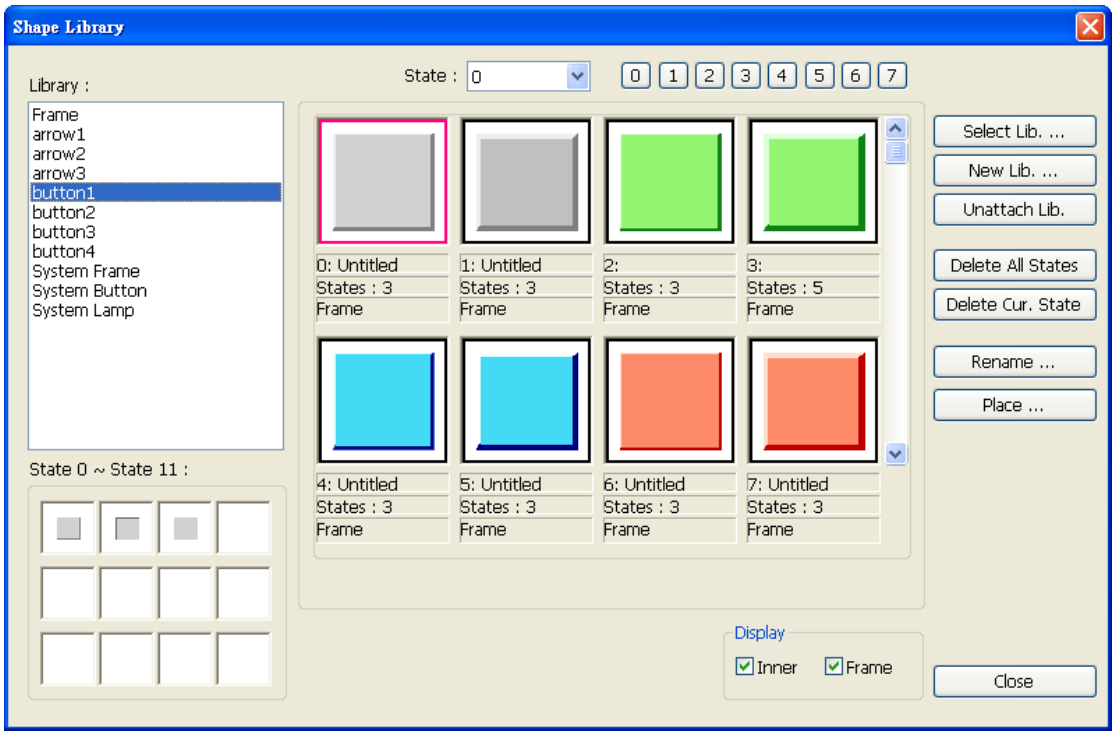
14.1 Creating Shape Library

Shapes are vector graphics in the use of lines, curves or polygons. A complete Shape can have more than one state, and each state includes two parts: frame and inner as shown below:

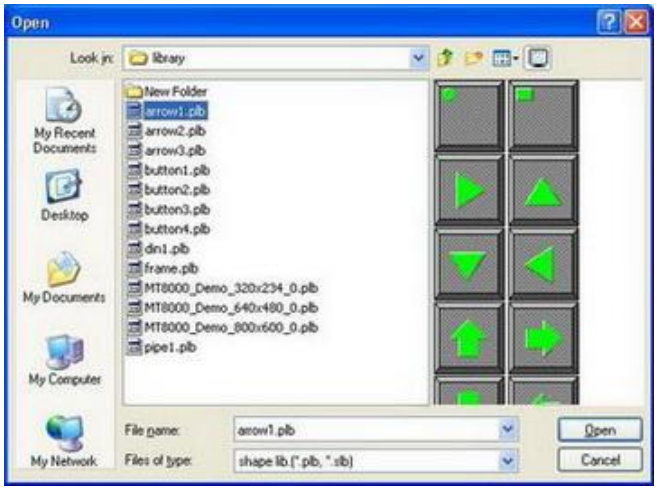
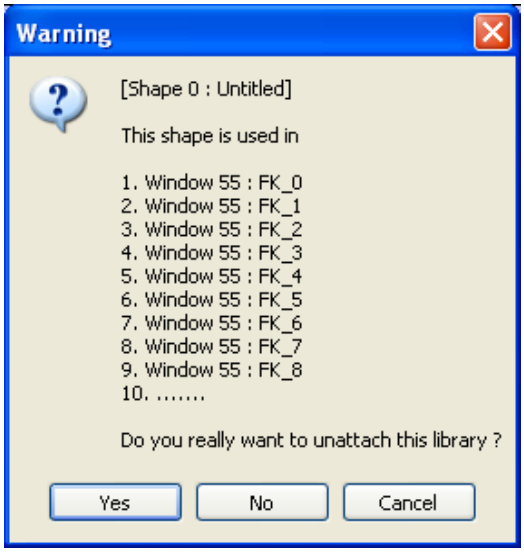


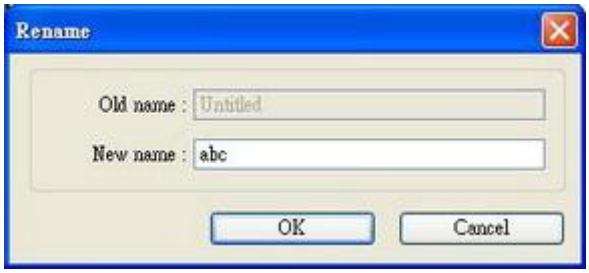
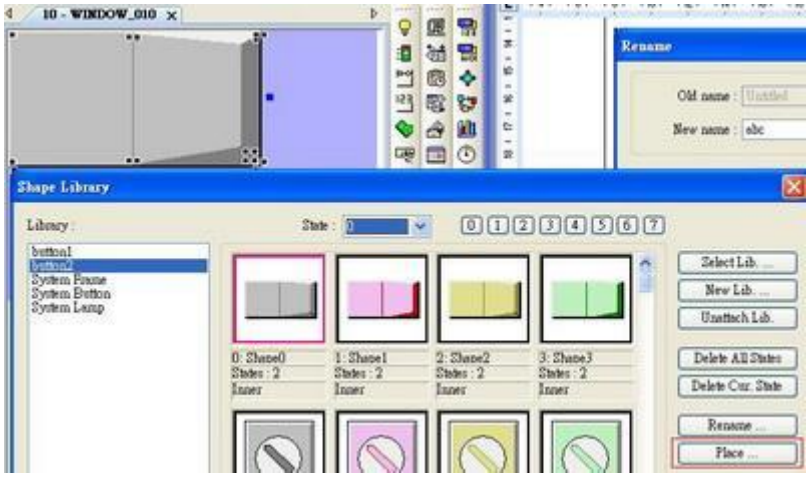
An object can use only frame or inner or both. Click **[Call up Shape Library]**, and the **[Shape Library]** dialog appears as below:





Setting	Description
Library	Shape Libraries which have been added into the current project. Select a library from the list.
State	Select the state represented by the current Shape. If the selected state doesn't have a shape, it means that the Shape does not exist or the state of the Shape isn't defined.
Select Lib.	Select a Shape Library to add into the project. By previewing the content of the library on the right side of the window, users can select a suitable library.

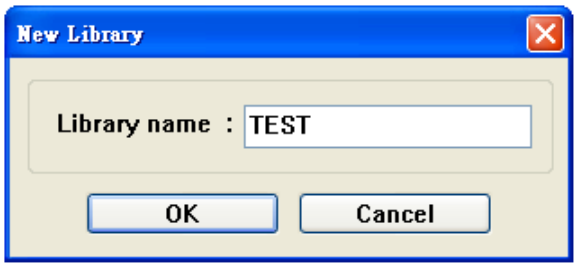
	
New Lib.	Add a new Shape Library. 
Unattach Lib.	Delete a Shape Library in the project. 
Delete All States	Delete all the states of the selected Shape.
Delete Cur. State	Delete the current state of the selected Shape.
Rename	Rename the selected Shape.

	
Place	Place the shape in the window. 

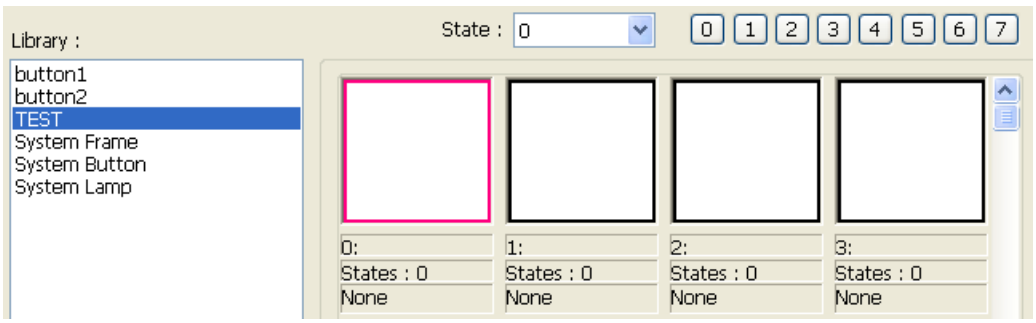
The following shows how to create a new Shape Library and add a Shape with two states into the library.

Step 1

Click **[New Lib.]** and enter the name of the new Shape Library.

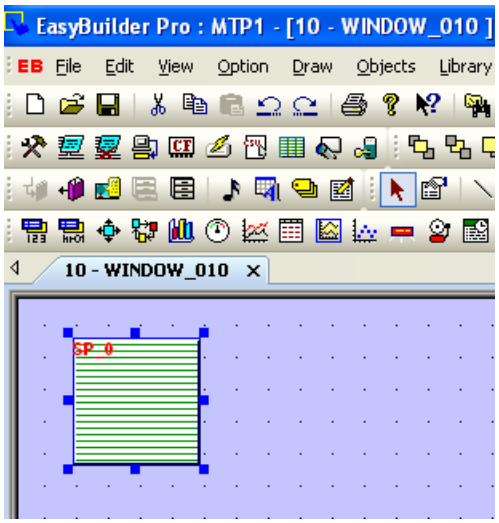


A new Shape Library “TEST” will be added. At this moment, no Shape is in this library.

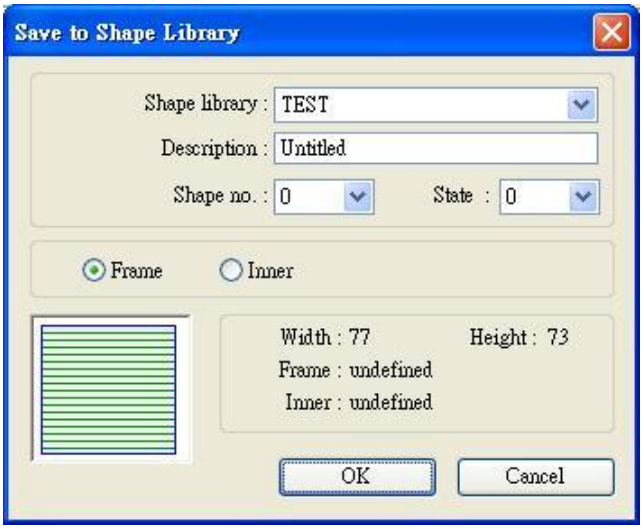


Step 2

Add a state to the selected Shape. First, use the drawing tools to draw a shape in the window and select it to add to the Shape Library.

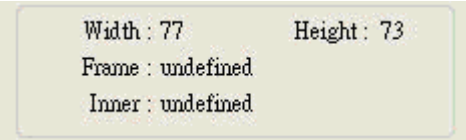


Chick **[Save to Shape Library]** button in the toolbar and the following dialog appears.



Setting	Description
Shape library	Select the Shape Library to add the selected shape. In this example, “TEST” library is selected.
Description	The name of the Shape.
Shape no.	The number in the Shape Library that the shape belongs to.
State	Select the state that the shape represents. In this case the state is set to “0”. Each shape can have up to 256 states.
Frame	If **[Frame]** is selected, the shape is defined as a frame.
Inner	If **[Inner]** is selected, the shape is defined as the inner part.

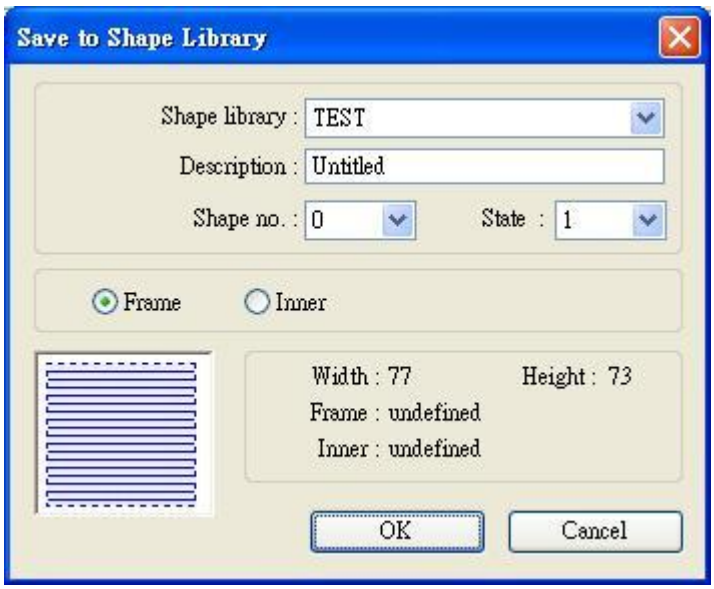
The following shows the information of the shape numbered 0 in the TEST library. Its state 0 is not defined as a frame or inner.



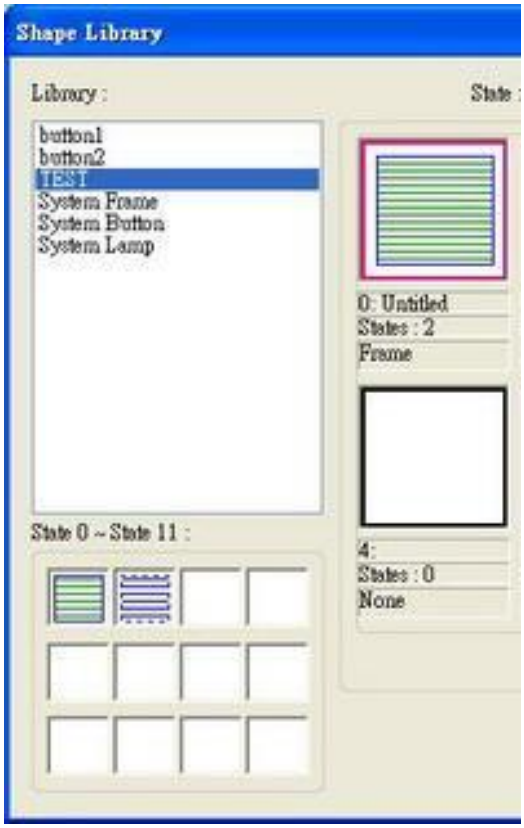
Click **[OK]**, the shape will be added to the library. The below shows the shape number 0 in this library has only 1 state and is defined as a frame.

Step 3

Do the step 2 again, this time the shape is set to state 1.

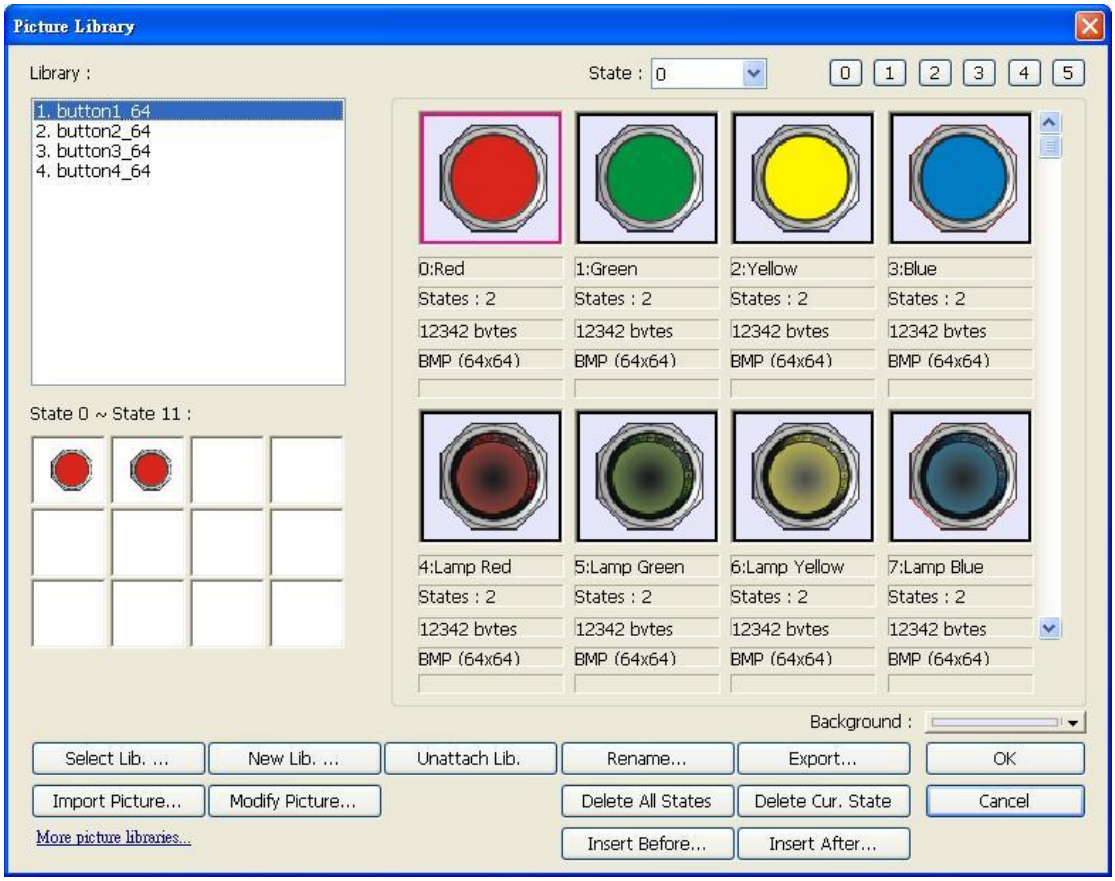


A complete Shape with two states is created as shown below.

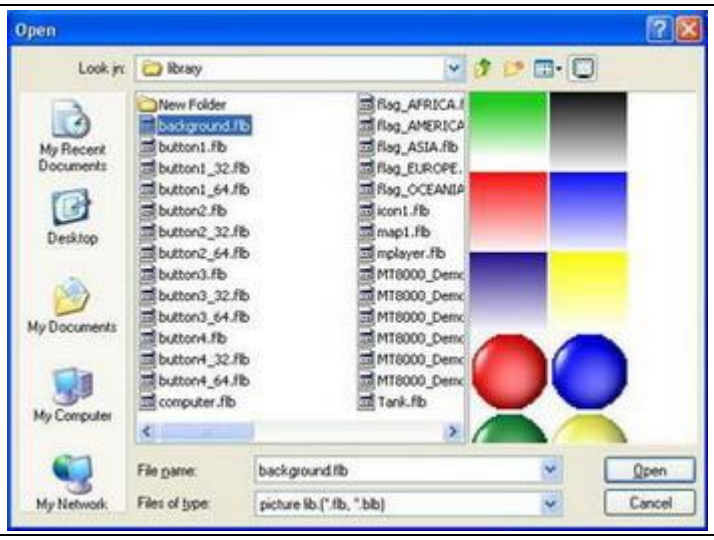
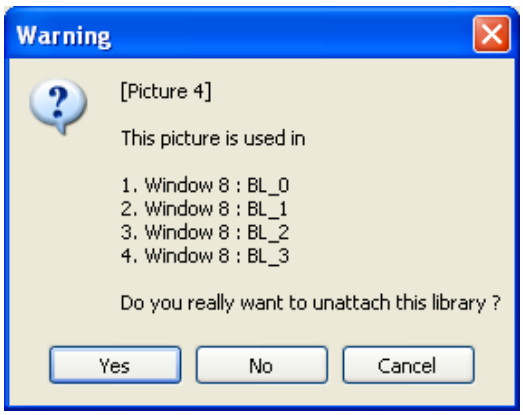


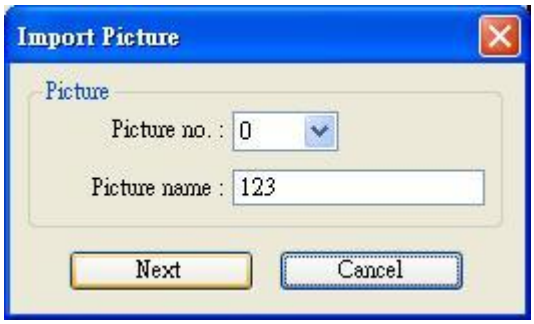
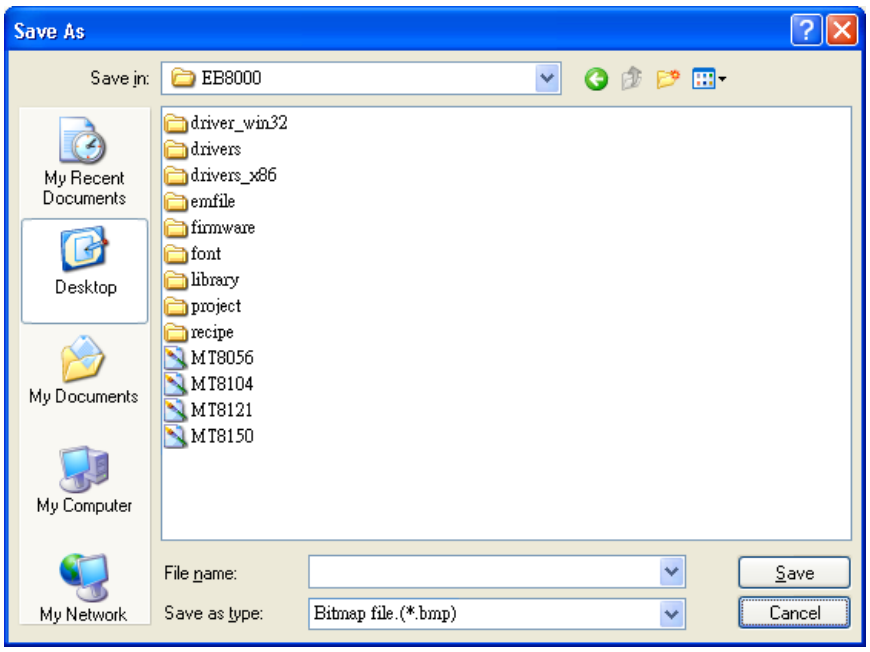
14.2 Creating Picture Library

Click **[Call up Picture Library]** button in the toolbar and the **[Picture Library]** dialog appears.

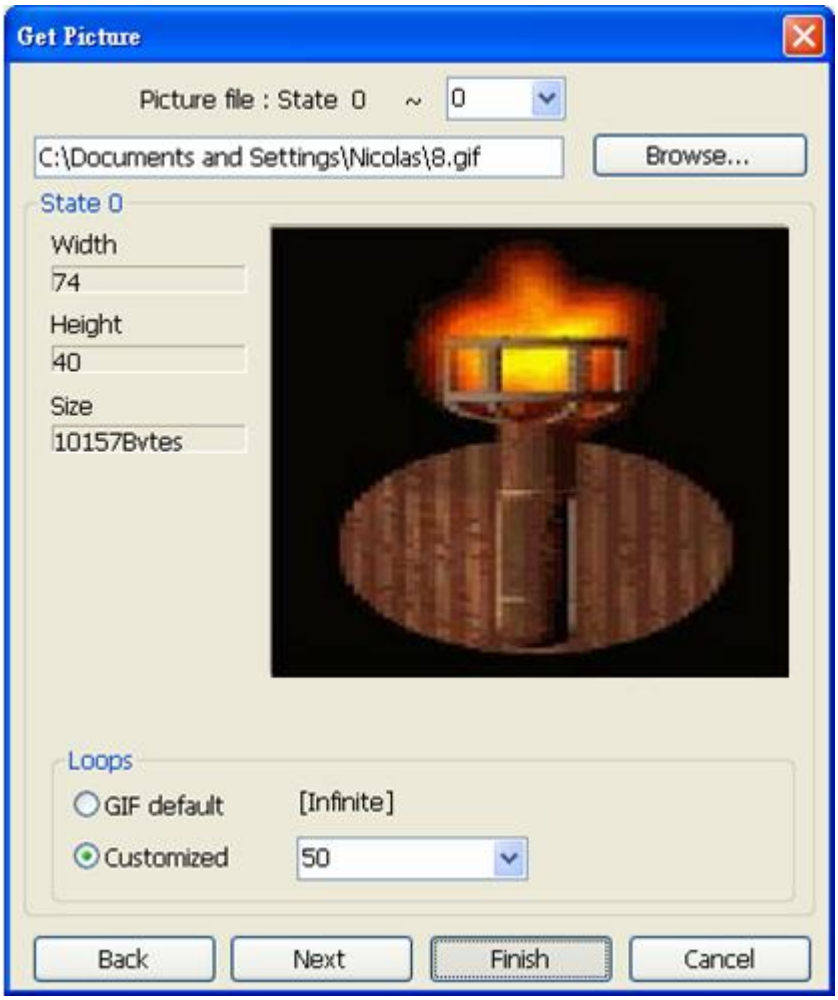


Setting	Description
Library	Picture Libraries which have been added into the current project. Select a library from the list.
State	Select the state represented by the current Picture. If the selected state doesn't have a picture, it means that the picture does not exist or the state of the picture isn't defined.
Select Lib.	Select a Picture Library to add into the project. By previewing the content of the library on the right side of the window, users can select a suitable library.

	
New Lib.	Add a new Picture Library. 
Unattach Lib.	Delete a Picture Library in the project. 
Delete All States	Delete all the states of the selected Picture.
Delete Cur. State	Delete the current state of the selected Picture.
Rename	Rename the selected Picture. 
Insert Before	Add a new state before the current state.

Insert After	Add a new state after the current state.
Import Picture	Add a new picture to the Picture Library. 
Modify Picture	Modify the selected picture.
Export	Export the selected picture to the designated place. As shown below, users can get the original picture. 

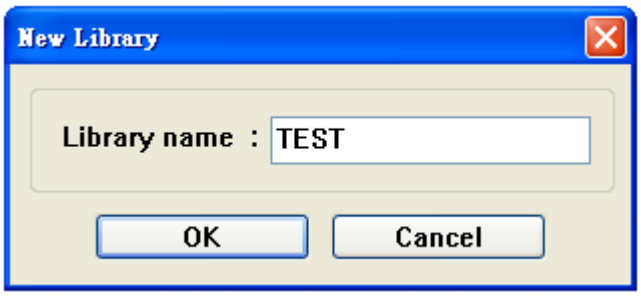
- The compatible picture formats are *.bmp, *.jpg, *.gif, *.dpd, and *.png. When adding a GIF picture in Picture Library, if this picture file is animated, the loop times of this animated picture can be set.



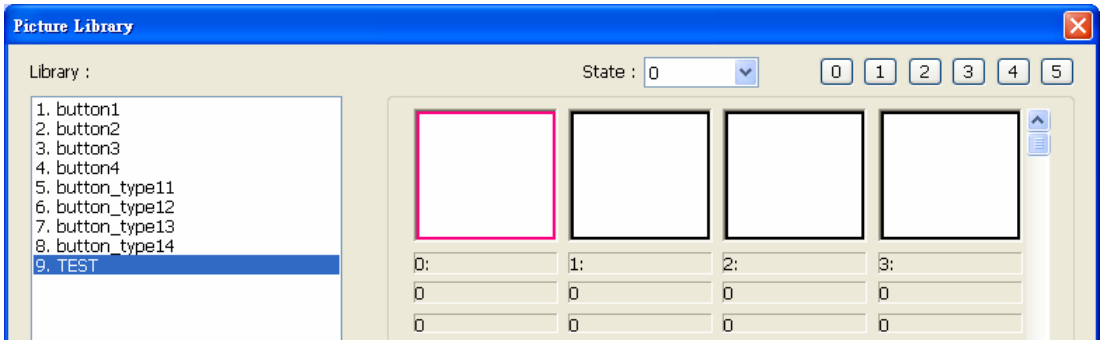
The example below shows how to create a new Picture Library and add a Picture with two states into it.

Step 1

Click **[New Lib.]** and enter the name of the new Picture Library.

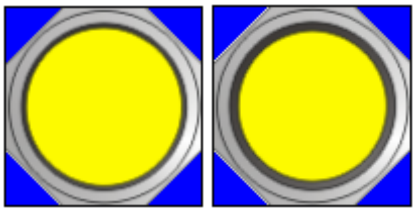


A new Picture Library “TEST” will be added. At this moment, there is no Picture in the library.

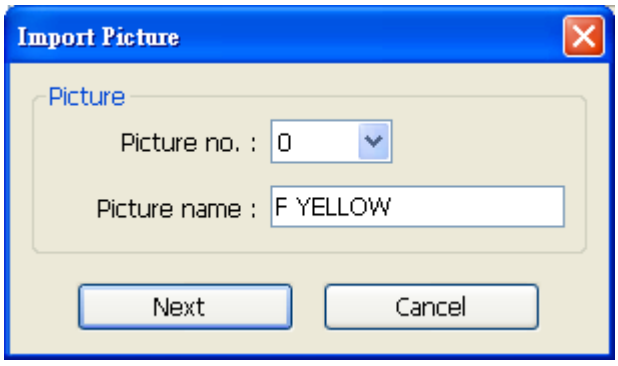


Step 2

Draw the pictures to be added; suppose the two pictures below are used to represent state 0 and state 1 respectively.

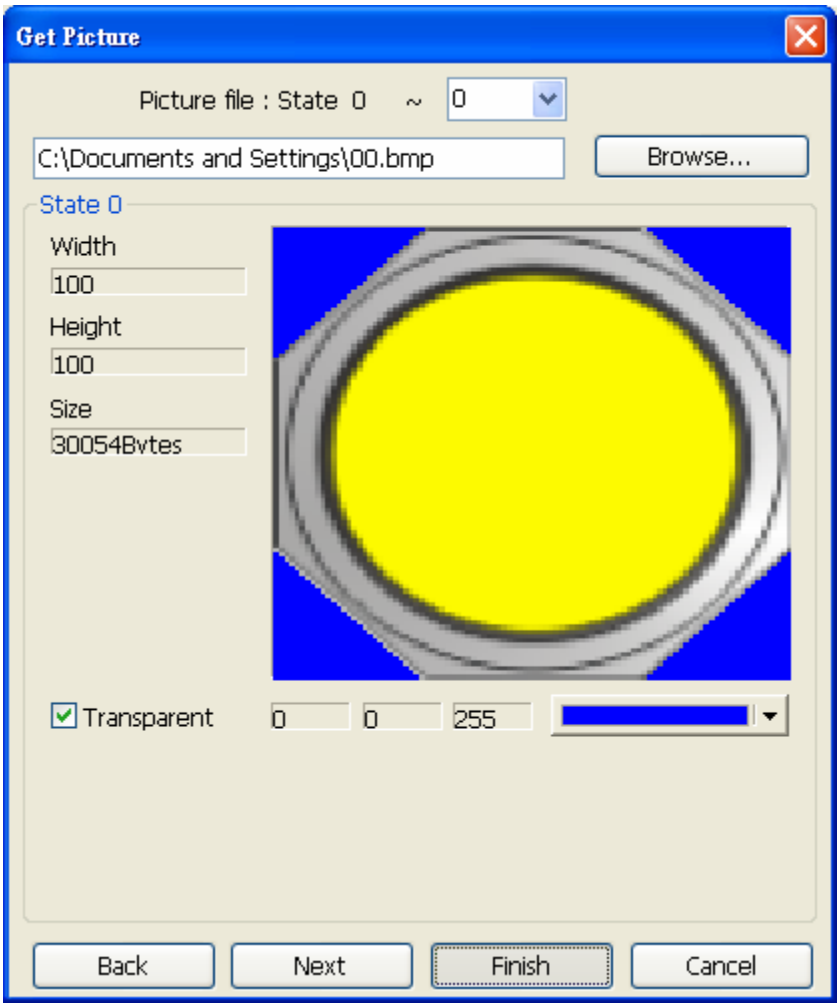


Click **[Import Picture]** and a dialog appears as below. Set **[Picture no.]** and **[Picture name]**, and then click **[Next]**.



Step 3

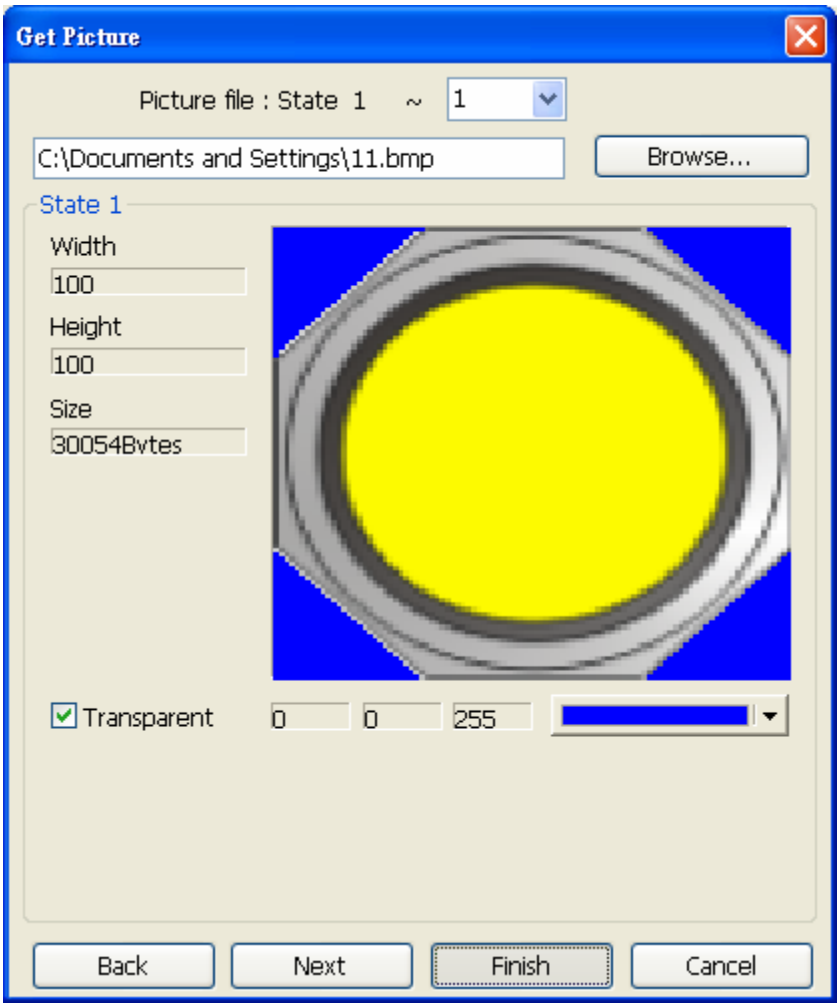
When the dialog below is shown, select a picture for state 0. If select **[Transparent]** check box and set to RGB (0, 0, 255), the blue area of this picture will be transparent. After setting state 0, click **[Next]** button to set state 1.



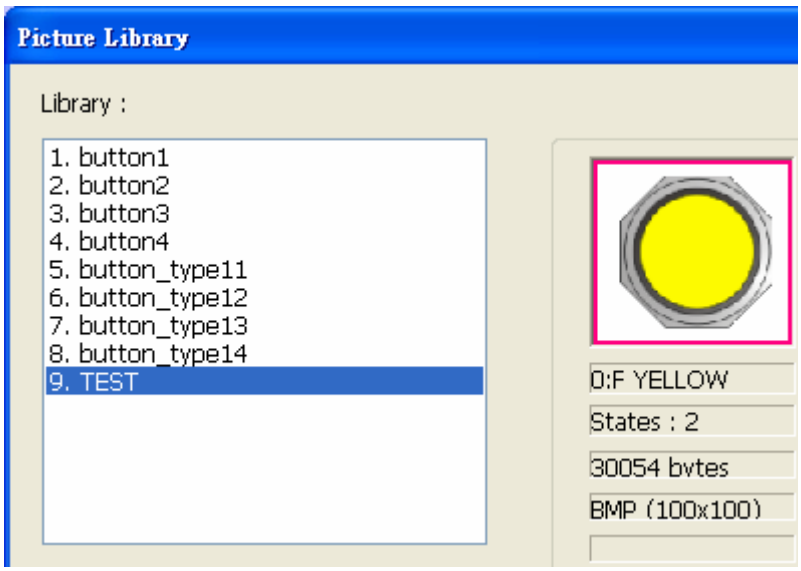
Select **[Transparent]** check box and click on an area of the picture, the RGB value of the area will be displayed. Take the settings above as an example, the actual picture shown is:

Step 4

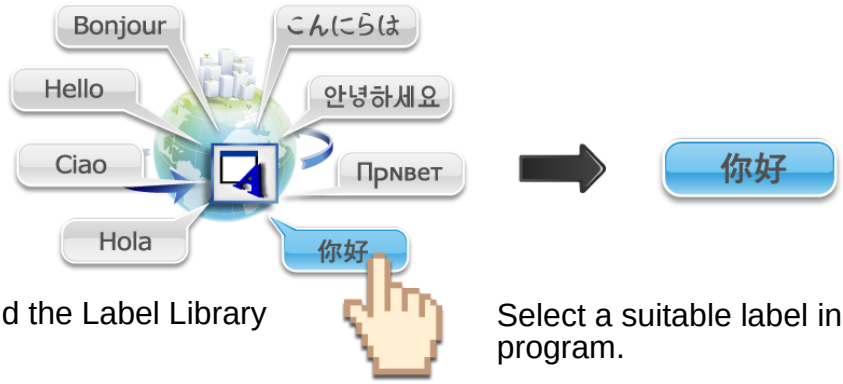
Do the steps above again, set the transparent area of the picture for state 1, and then click **[Finish]**. A picture with two states will be created.



When finished, a new picture “F Yellow” is added to the library. As shown, this picture is in BMP format and has two states.



Chapter 15 Label Library and Multi-Language

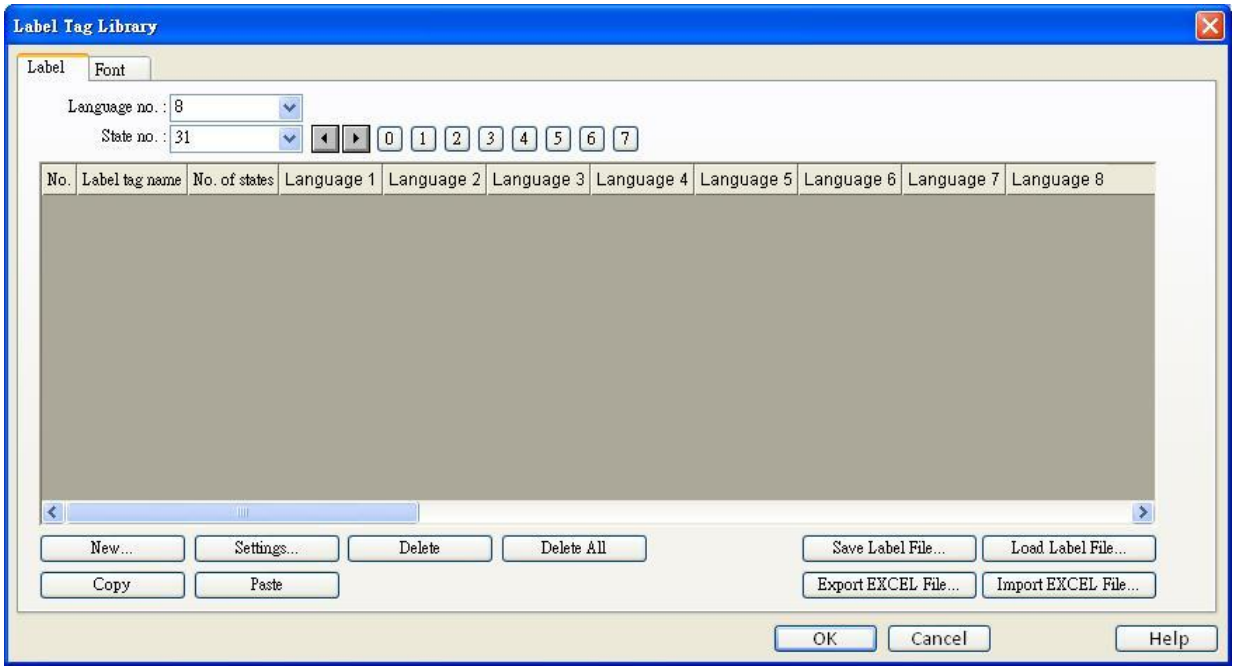


Build the Label Library

Select a suitable label in the program.

15.1 Introduction

In run time the project displays the corresponding language based on the settings.



[Language no.]
EasyBuilder supports the editing of 8 languages in maximum.

[State no.]
Indicates the current state. Each Label has a maximum of 256 states (state no. 0 ~ 255). The state no. is determined by **[Language no.]** selected. If 8 languages are used, the maximum state no. is 256 / 8 = 32.

[New]
Add a new label.

[Settings]
Settings of the selected label.

[Save Label File]
Save all labels in *.lbl format.

[Load Label File]
Load the existing *.lbl file to Label Library.

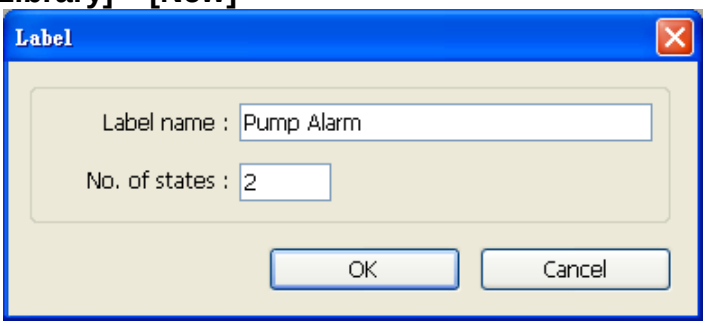
[Export EXCEL File]
Load the existing *.csv or *.xls file to label library.

[Import EXCEL File]
Load the existing *.lbl file to Label Library.

- Unicode is not supported when importing and exporting Excel file.

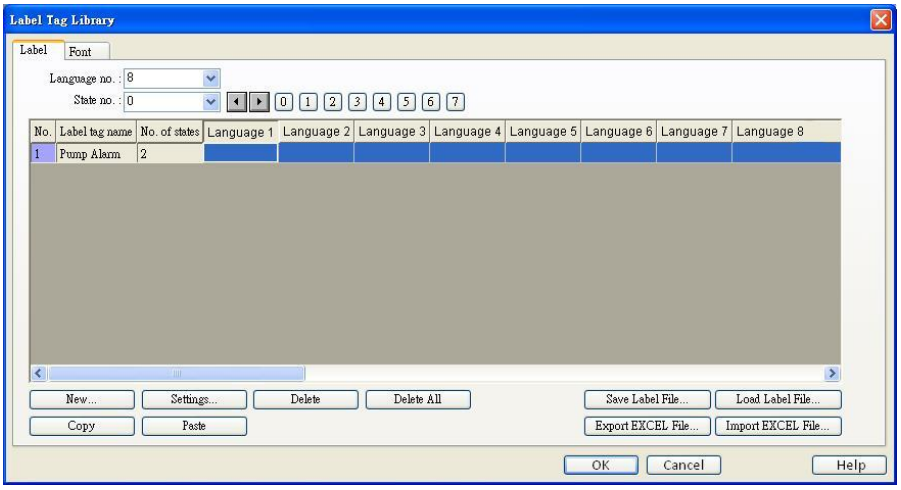
15.2 Building Label Library

1. Open **[Label Tag Library] » [New]**

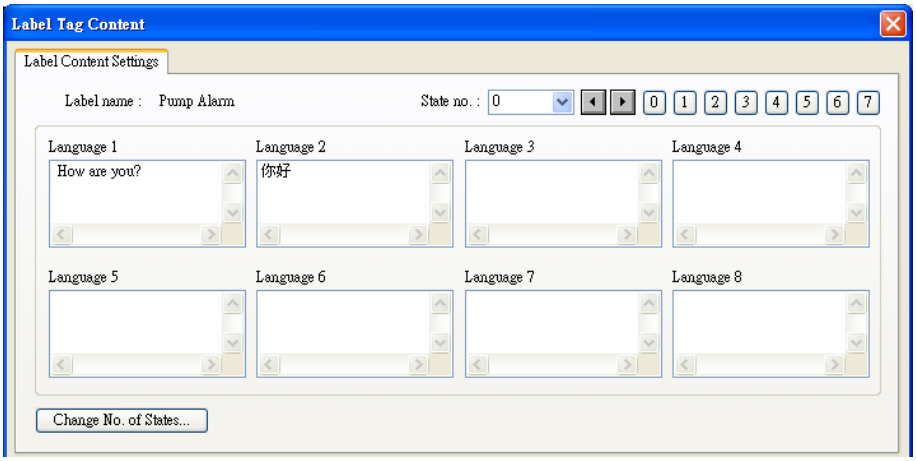


[Label name]
Specify the name of the Label.
[No. of states]
The number of states to be displayed by this Label.

2. Click **[OK]**, a new label is added to the Label Library, select it and click **[Settings]** to set its content.

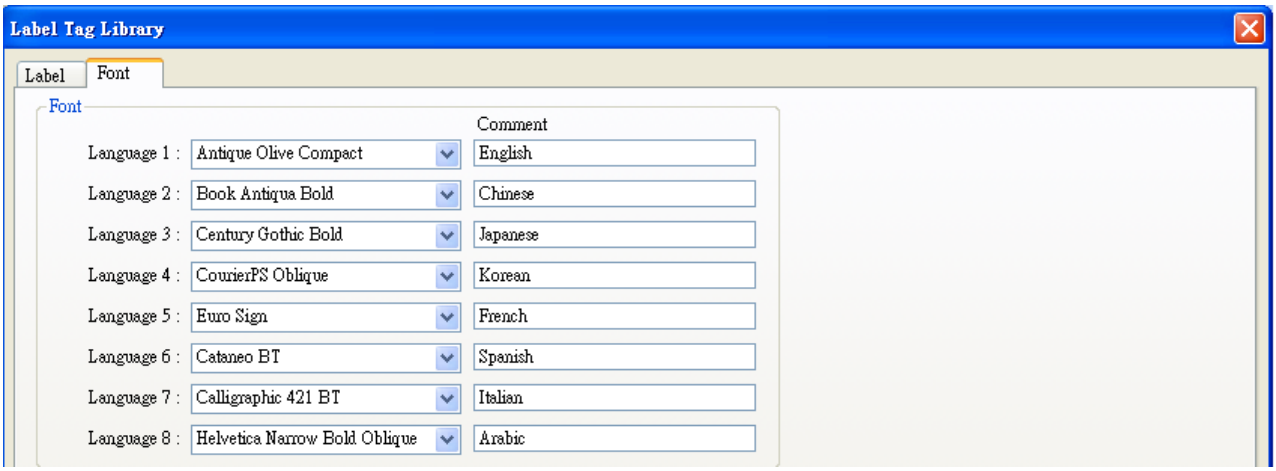


3. Set up the corresponding language contents.



15.3 Setting Label Font

Select **[Label Tag Library] » [Font]** to see the languages the Label contains and set the font. Different languages can use different
fonts.



[Font]

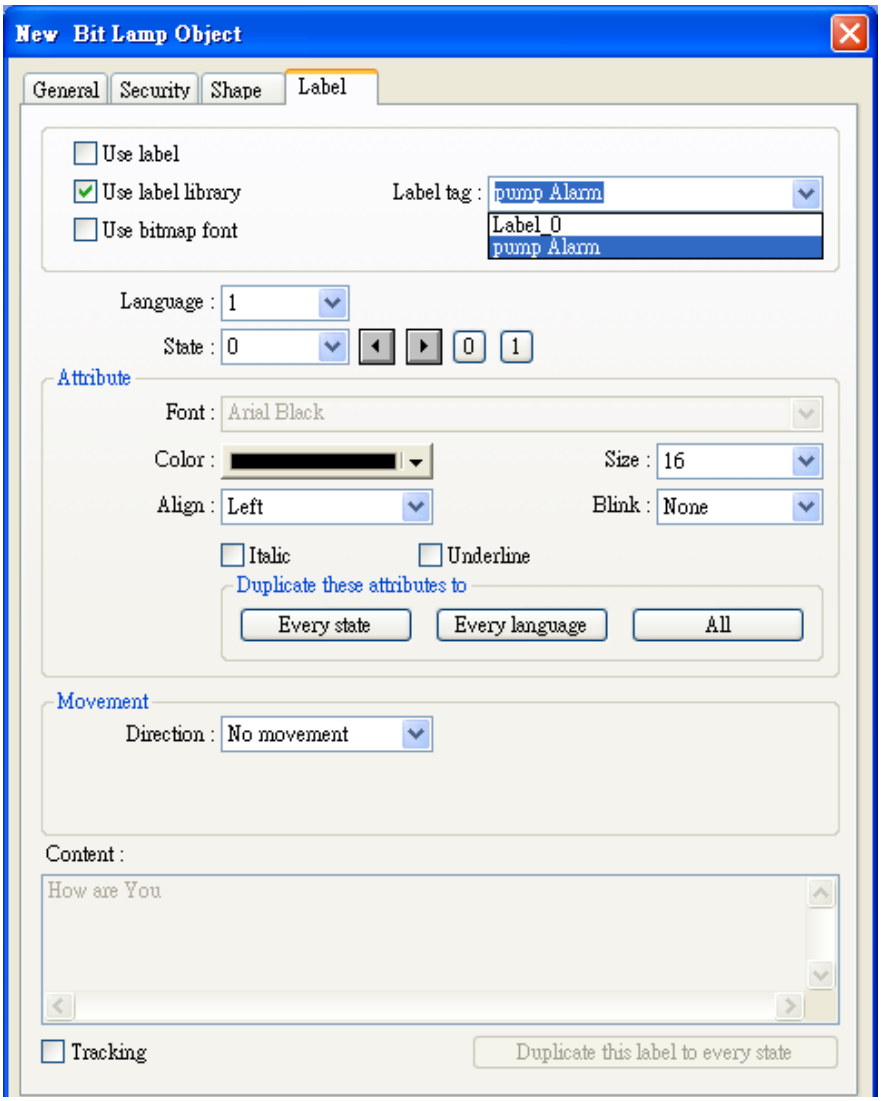
When using a Label to show different languages, different fonts can be selected.

[Comment]

The memo for each font.

15.4 Using Label Library

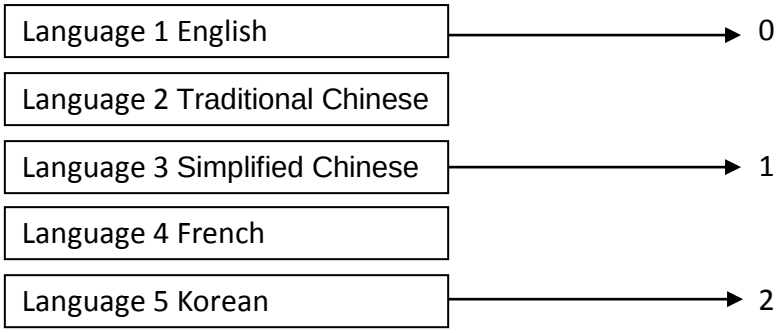
When there are defined labels in Label Library, the labels can be found in the object's **[Label]** tab. Select **[Use label library]**, and select the label from the pull down list of **[Label tag]**.



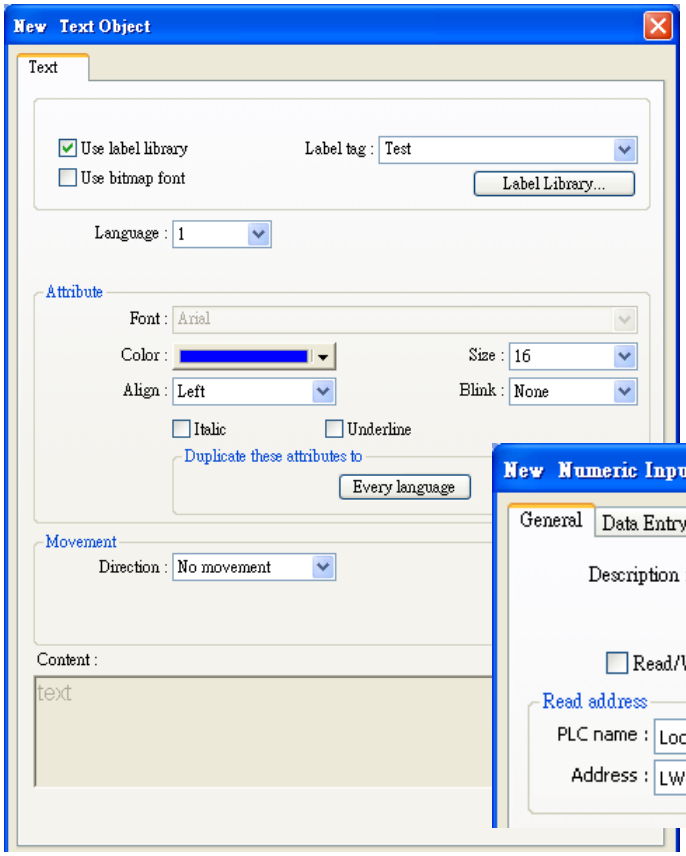
When a tag is selected, the content of the selected tag is shown in the **[Content]** field, in the font selected. Please note that languages 2 ~ 8 can only set the Font **[Size]**, other settings such as **[Color], [Align], [Blink]** etc. will follow the settings of language 1.

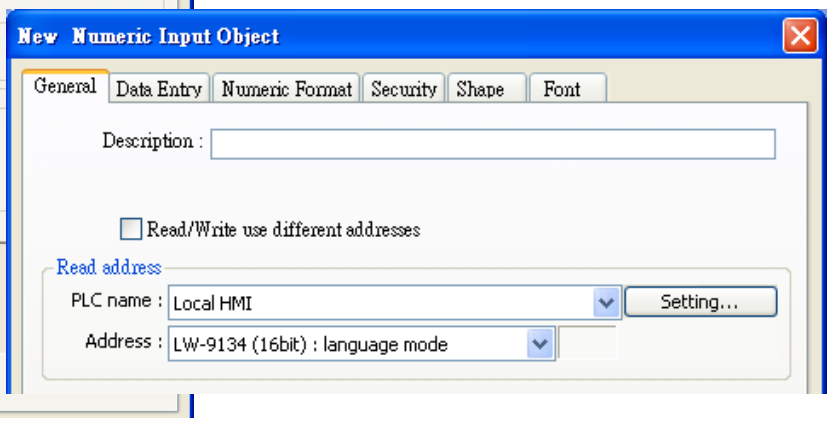
15.5 Settings of Multi-Language (System Register LW-9134)

When displaying the texts in multiple languages, except for using Label Library, the system register “[LW-9134]: language mode” should be used too.
The value range of [LW-9134] is 0 ~ 7. Different values correspond to different languages. Although up to 8 languages can be used in EasyBuilder, if the languages are not all chosen in compiling and downloading, [LW-9134] will work differently.
For example, user defines 5 languages in Label Library. Only languages 1, 3, 5 are chosen in compiling, then the corresponding values of [LW-9134] are:



How to use multi-language:

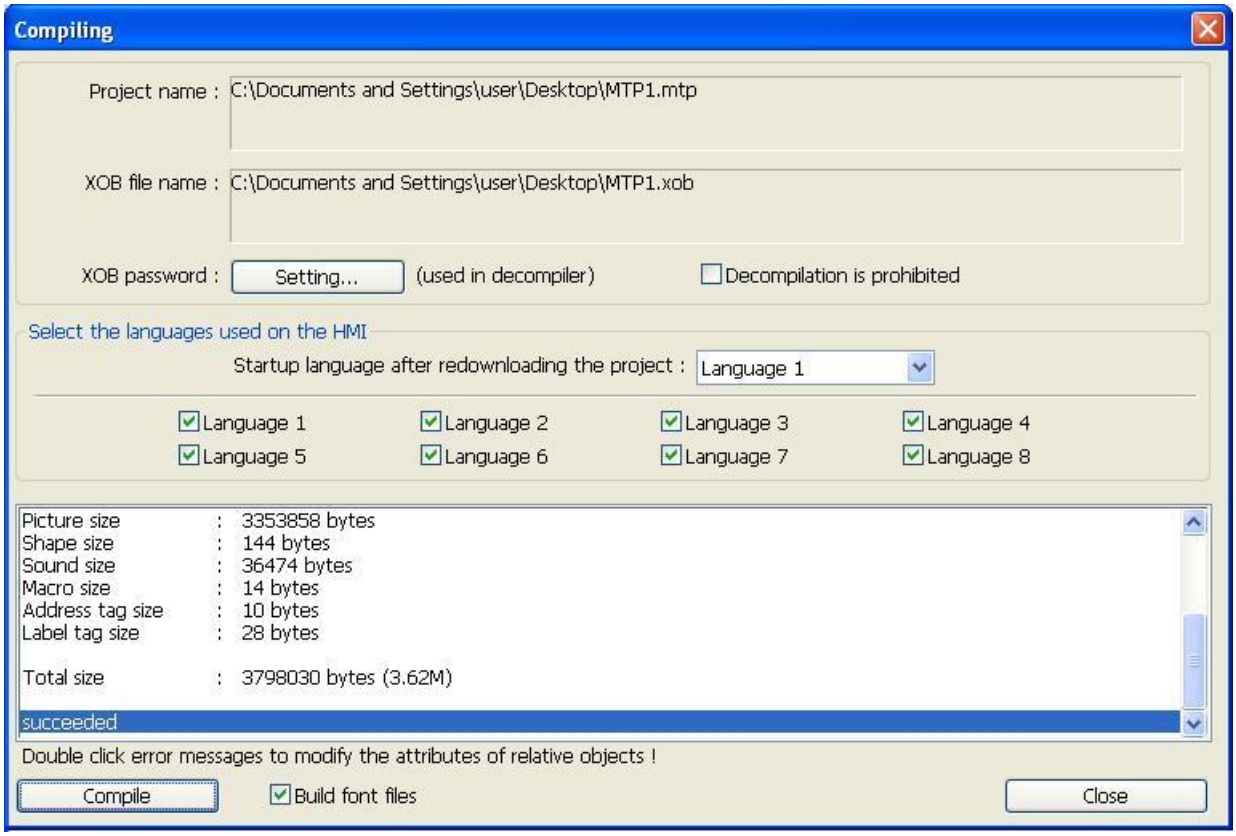




1. Create a Text Object and select **[Use label library]** check box.

2. Create a Numeric Input Object and use system register **[LW-9134]**.

When compiling, select the languages.



The simulation is shown below, if change the value of [LW-9134]; the content displayed by the Text Object will be changed.

English

LW9134 : language mode 0

简体中文(SIMPLE)

LW9134 : language mode 2

한국어 웹(KOREAN)

LW9134 : language mode 4

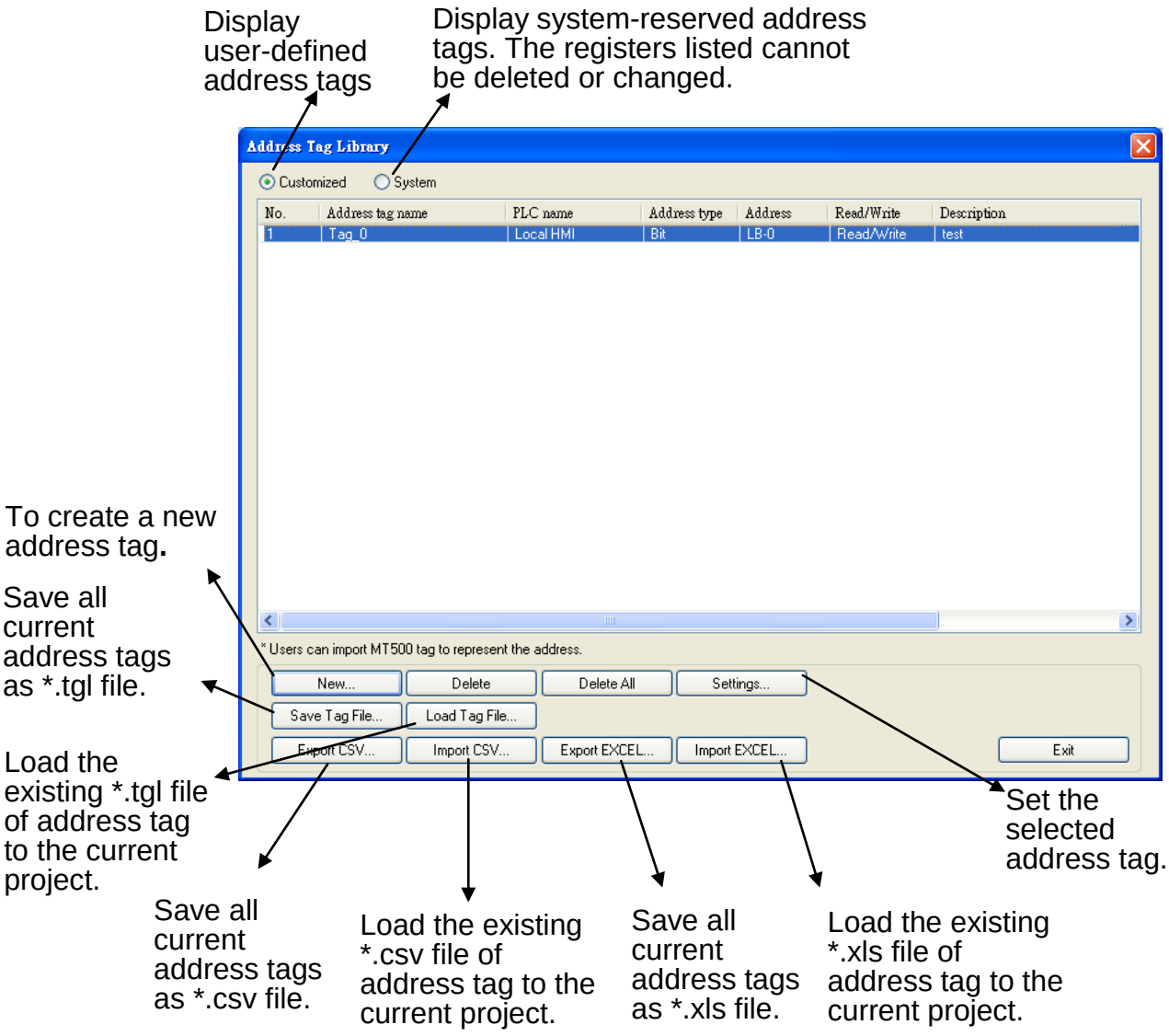
- At most 8 languages can be downloaded to HMI simultaneously.

Please confirm your internet connection before downloading the demo project.

Chapter 16 Address Tag Library

16.1 Building Address Tag Library

Generally it is recommended to define the commonly used addresses in Address Tag Library when start to build a project. It not only avoids setting the addresses repeatedly but also expresses the function of an address more clearly.



Click **[New]** and set the relevant properties.

The information of the address tag.

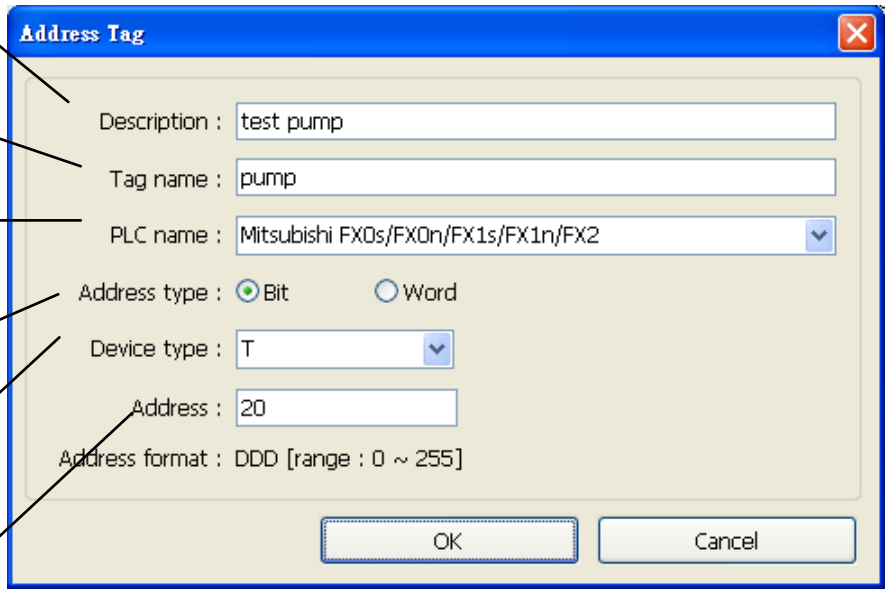
The name of the address tag.

As defined in **[System Parameter Settings] » [Device list]**.

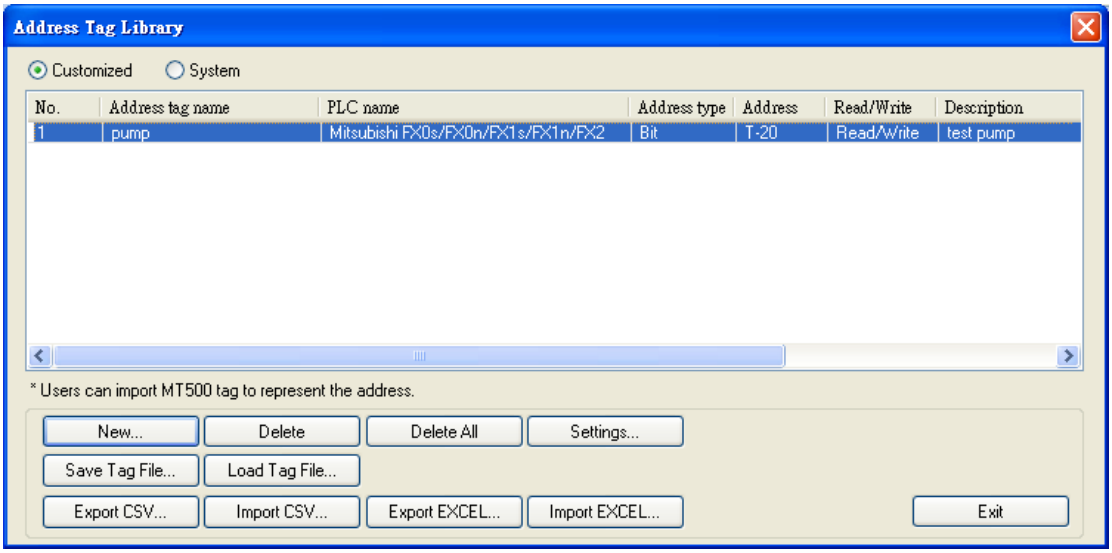
The tag address type; **[Bit]** / **[Word]**.

Available device types are limited by **[PLC name]** & **[Address type]**.

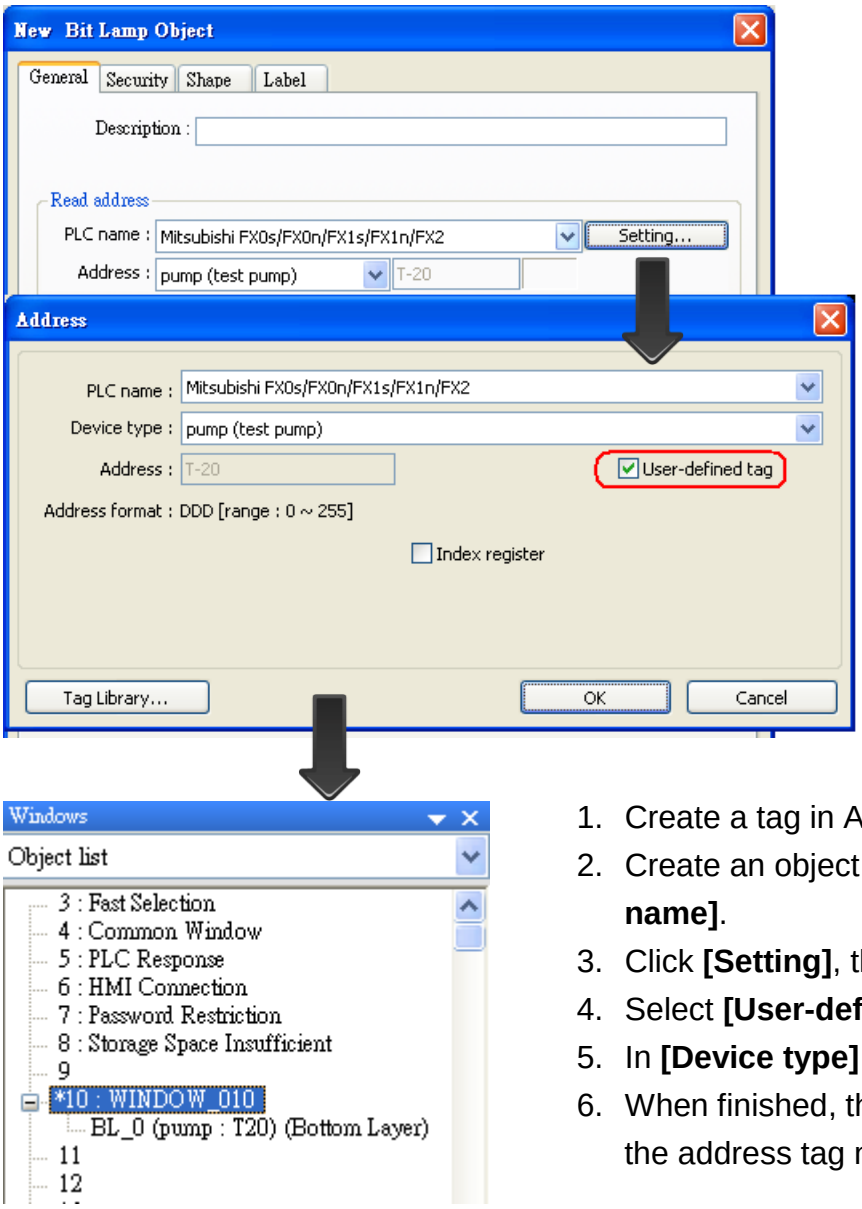
Address of the Tag.



Click **[OK]**, a newly added tag can be found in the **[Customized]** library.



16.2 Using Address Tag Library

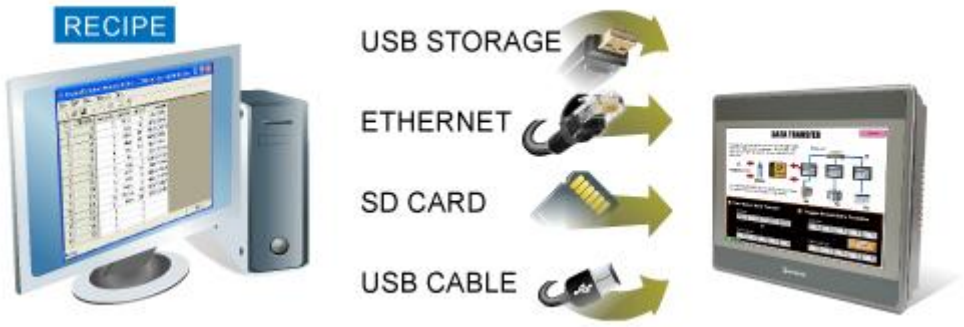


1. Create a tag in Address Tag Library.
2. Create an object, select **[General] » [PLC name]**.
3. Click **[Setting]**, the setting dialog pops up.
4. Select **[User-defined tag]** check box.
5. In **[Device type]** select the defined tag.
6. When finished, the window tree will show the address tag name used by the object.

Chapter 17 Transferring Recipe Data

Recipe Data refers to the data stored in RW and RW_A addresses. The way of reading and writing these addresses is the same as operating a word register. The difference is that recipe data is stored in HMI, when restarting HMI, the latest data records in RW and RW_A are kept.

The size of recipe data a RW address can store is 1024KB words, and RW_A is 128KB. Users can update recipe data with SD card, USB disk, USB cable or Ethernet and use the data to update PLC data. Recipe Data can also be uploaded to PC; furthermore, PLC data can be saved in recipe data. The following explains the ways of transferring recipe data.

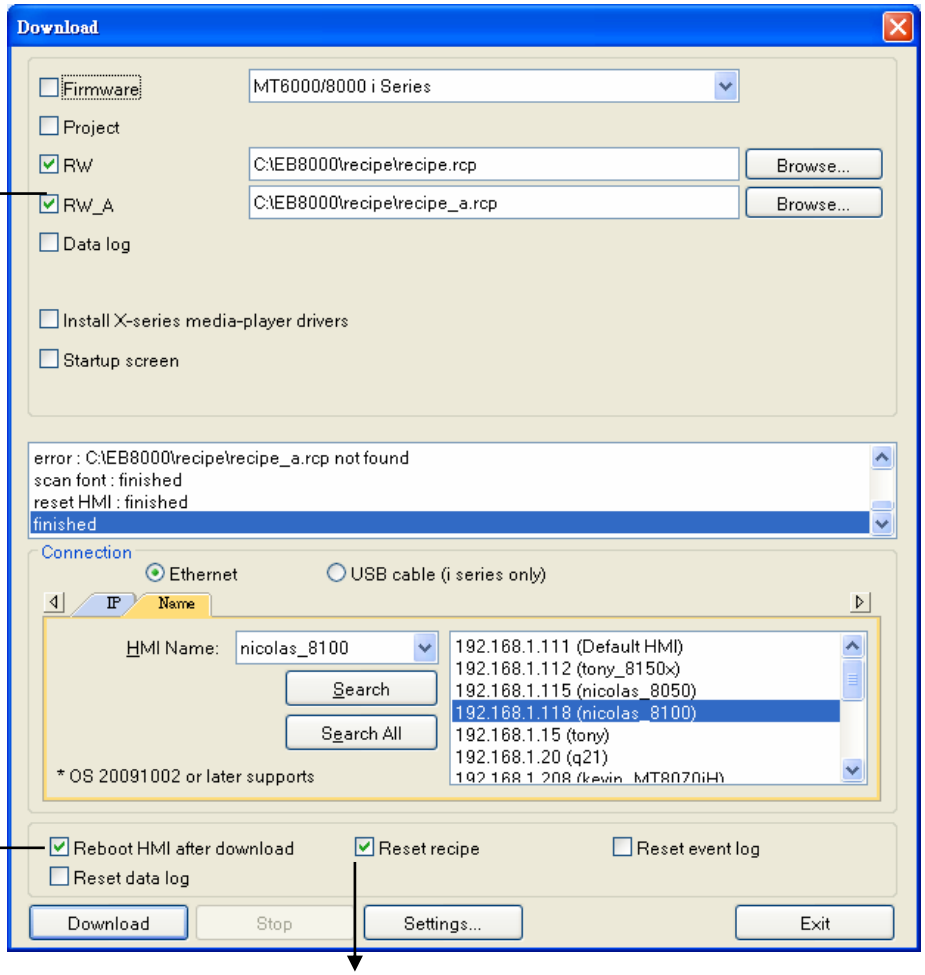


17.1 Updating Recipe Data with Ethernet or USB Cable

1. Open Project Manager and click **[Download]**.
2. Select **[RW]** and **[RW_A]** and **[Browse]** the source file.
3. After downloading, restart HMI, RW and RW_A will be updated.

Select the file source.

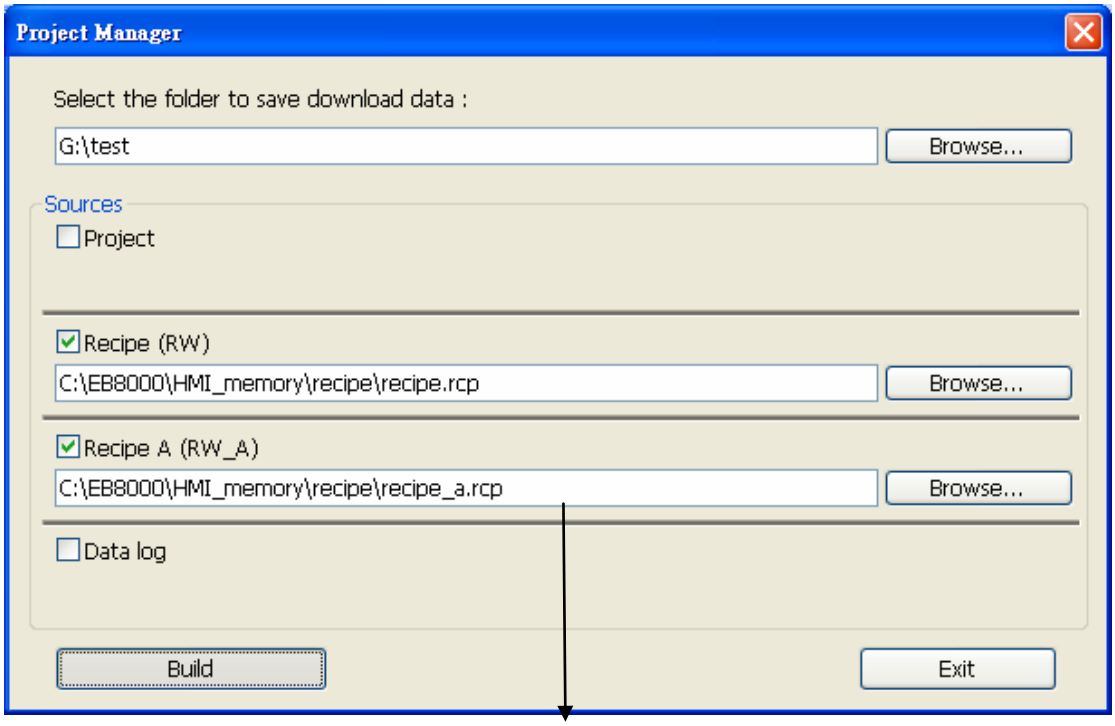
Automatically reboot HMI after downloading.



When **[Reset recipe]** check box is selected, the system will clear all the data in **[RW]** and **[RW_A]** before downloading.

17.2 Updating Recipe Data with SD Card or USB Disk

1. Open Project Manager and click **[Build Download Data for SD Card or USB Disk]**.
2. Insert a SD card or USB disk into PC
3. Click **[Browse]** to designate the file path.
4. Click **[Build]**, EasyBuilder will save the data in SD card or USB disk.



Select the source files.

■ When download data is successfully built, two folders can be found: *history* and *mt8000. mt8000* is for storing project file; *history* is for storing recipe data and data sampling / event log records.

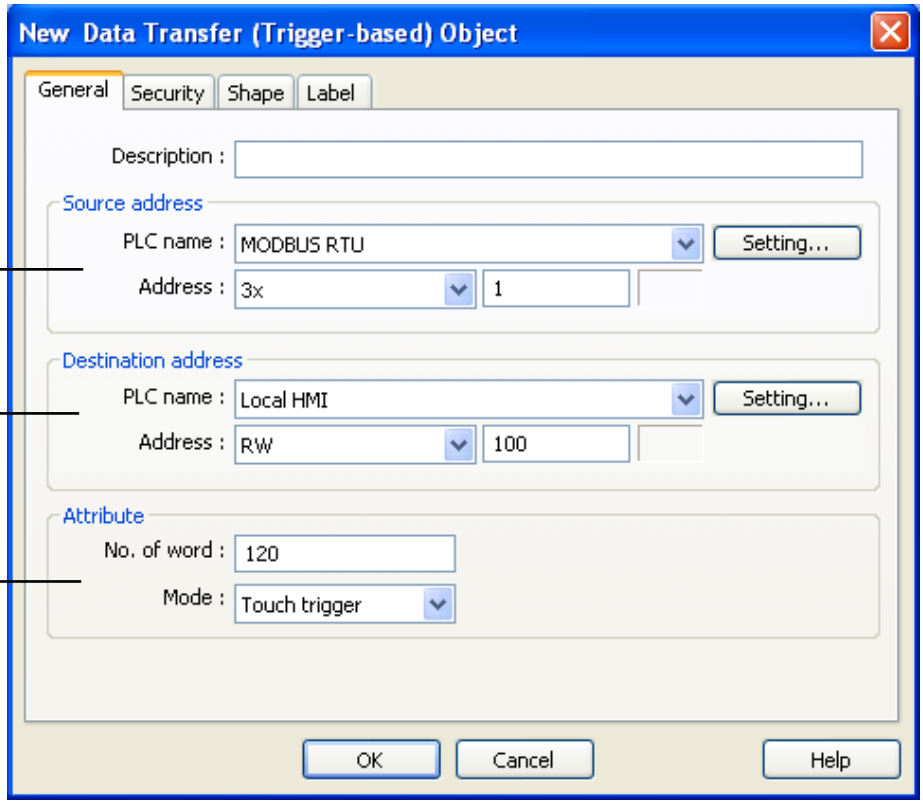
17.3 Transferring Recipe Data

Use **[Data Transfer (Trigger-based) Object]** to transfer recipe data to a specific address, or save the data of this address in [RW] and [RW_A].
Use [Data Transfer (Trigger-based) object] to transfer Recipe Data to the designated address, or save the data of the designated address to [RW] and [RW_A].

Set the source of the data to transfer from.

Set the destination of the data to transfer to.

Set the number of words to transfer from source to destination.



17.4 Saving Recipe Data Automatically

In order to prolong the life span of HMI memory, system will automatically save the recipe data to HMI **every minute**. To avoid losing data when turning HMI off during the interval of saving data, system register [LB-9029: Save all recipe data to machine (set ON)] is provided. Set ON LB-9029 will force system to save recipe data once. Set ON [LB-9028: Reset all recipe data (set ON)], system will clear all recipe data.

Chapter 18 Macro Reference

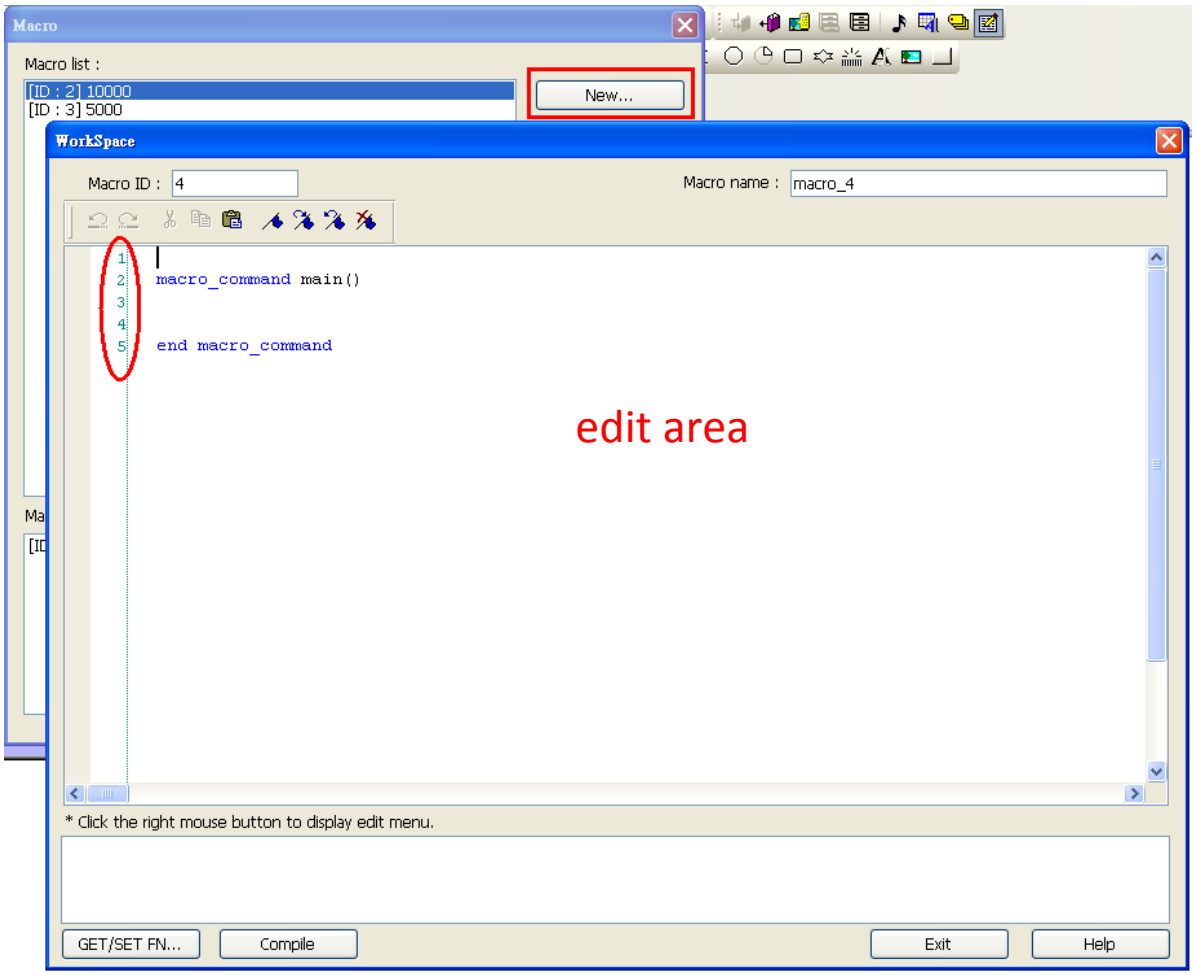
Macros provide the additional functionality your application may need. Macros are automated sequences of commands that are executed at run-time. Macros allow you to perform tasks such as complex scaling operations, string handling, and user interactions with your projects. This chapter describes syntax, usage, and programming methods of macro commands.

18.1 Instructions to use the Macro Editor

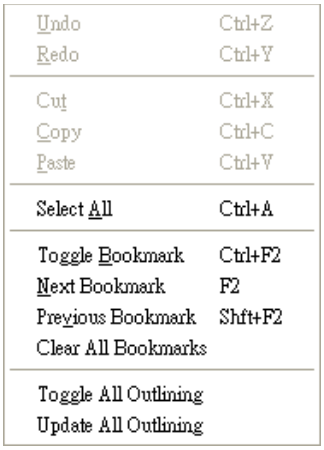
1. Macro editor provides the following functions:
    a. Display line number
    b. Undo / Redo
    c. Cut / Copy / Paste
    d. Select All
    e. Toggle Bookmark / Previous Bookmark / Next Bookmark / Clear All Bookmarks
    f. Toggle All Outlining
    g. Security -> Use execution condition
    h. Periodical execution
    i. Execute one time when HMI starts

    The instructions below show you how to use these functions.

2. Open the macro editor; you'll see the line numbers displayed on the left-hand side of the edit area.



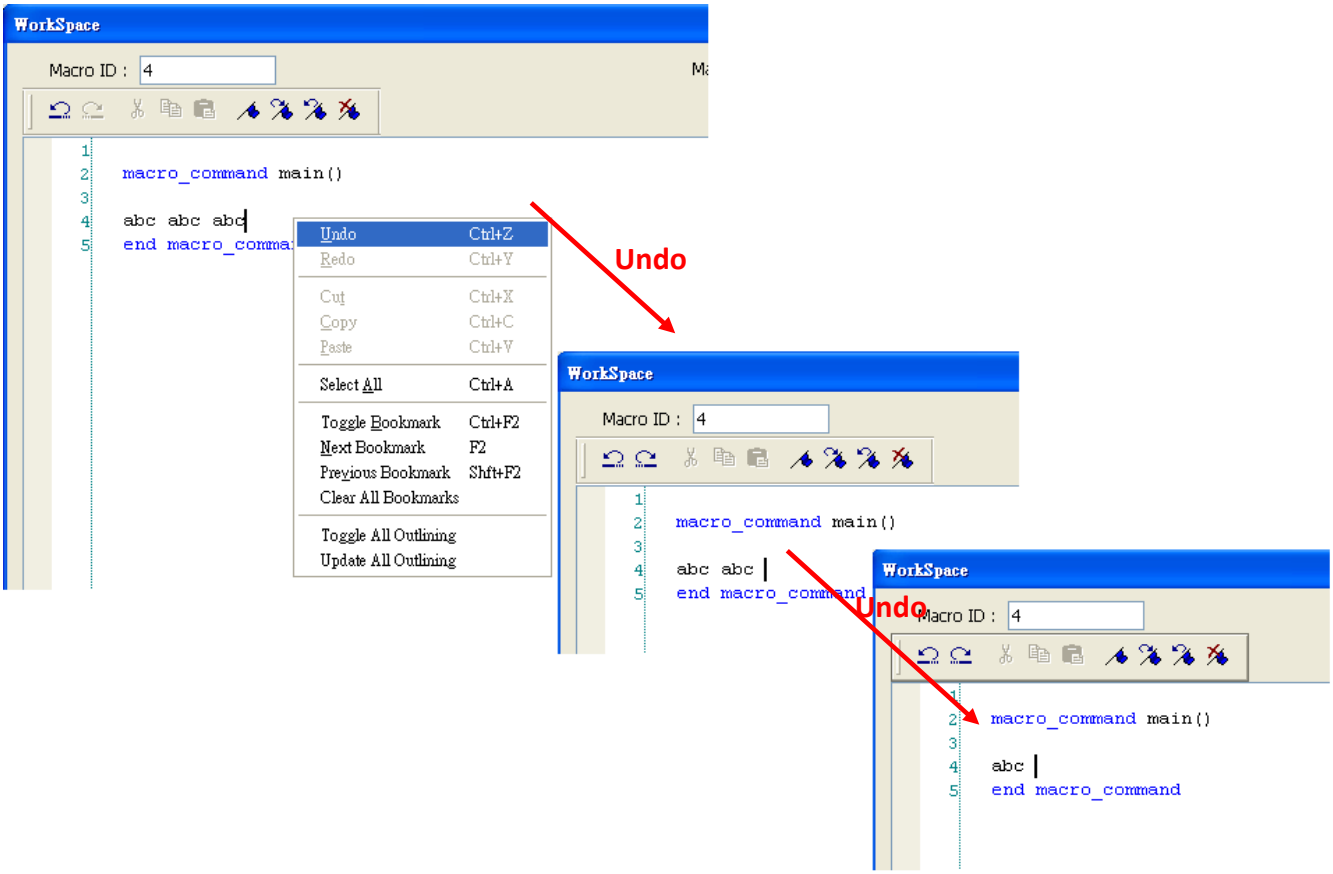
3. Right click on the edit area to open the pop-up menu as shown below:

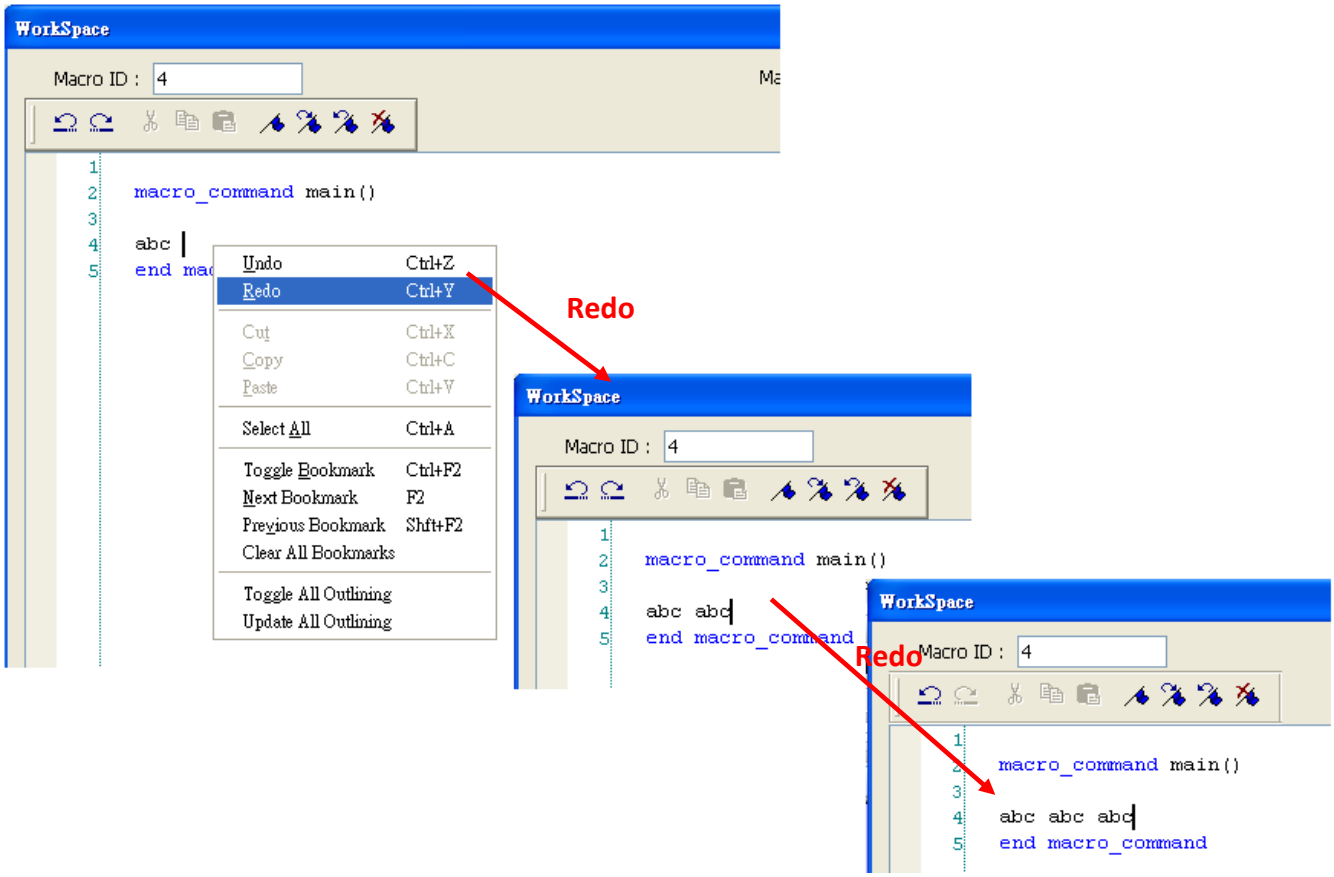


Disabled operations are colored grey, which indicates that it is not possible to use that function in the current status of the editor. For example, you should select some text to enable the copy function, otherwise it will be disabled. Keyboard shortcuts are also shown.

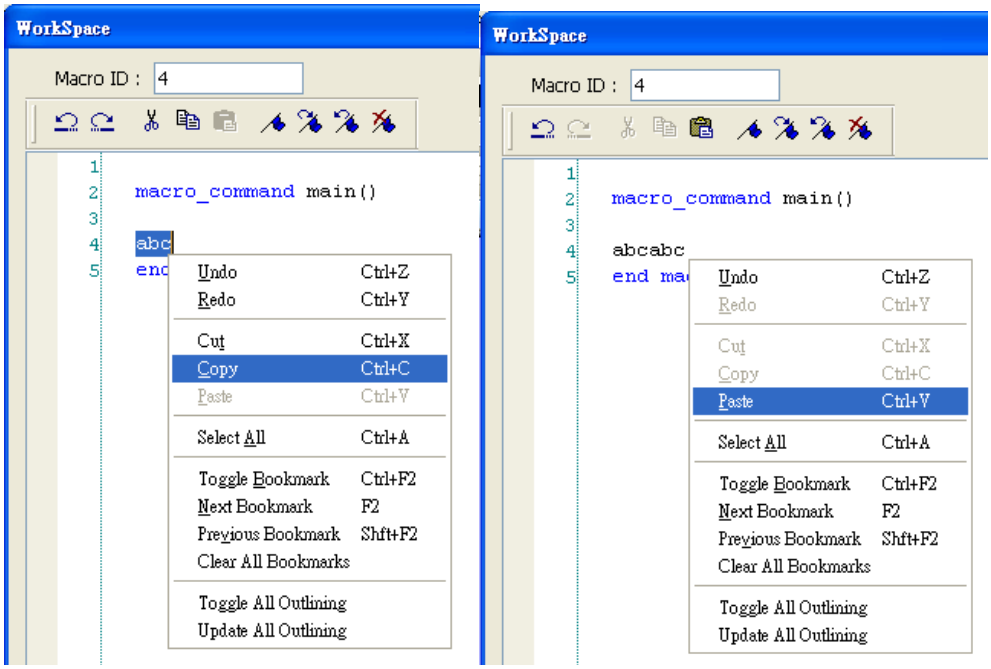
4. The toolbar provides [Undo], [Redo], [Cut], [Copy], [Paste], [Toggle Bookmark], [Next Bookmark], [Previous Bookmark] and [Clear All Bookmarks] buttons.

5. Any modification will enable the [Undo] function. [Redo] function will be enabled after the undo action is used. To perform the undo/redo, right click to select the item or use the keyboard shortcuts. (Undo: Ctrl+Z, Redo: Ctrl+Y).

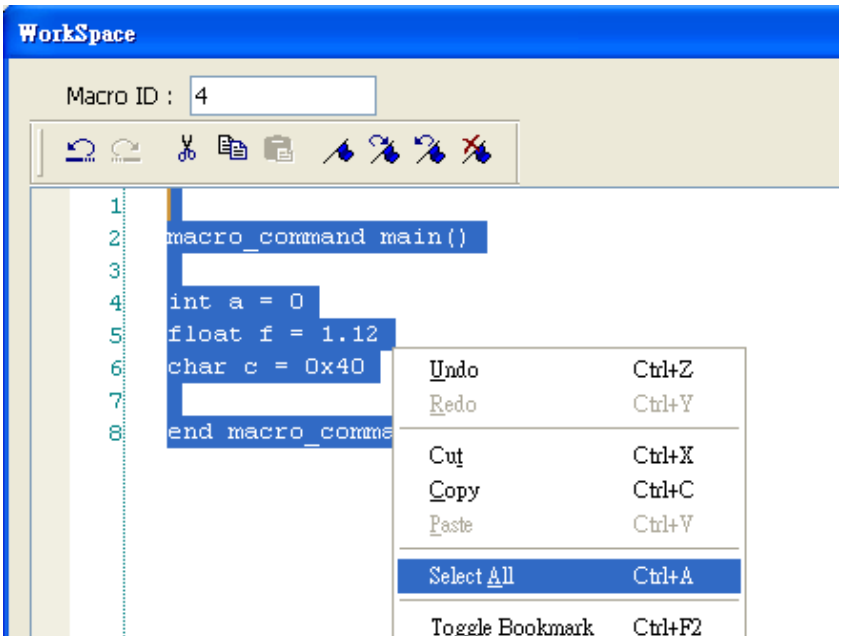




6. Select a word in the editor to enable the [Cut] and [Copy] function. After [Cut] or [Copy] is performed, [Paste] function is enabled.

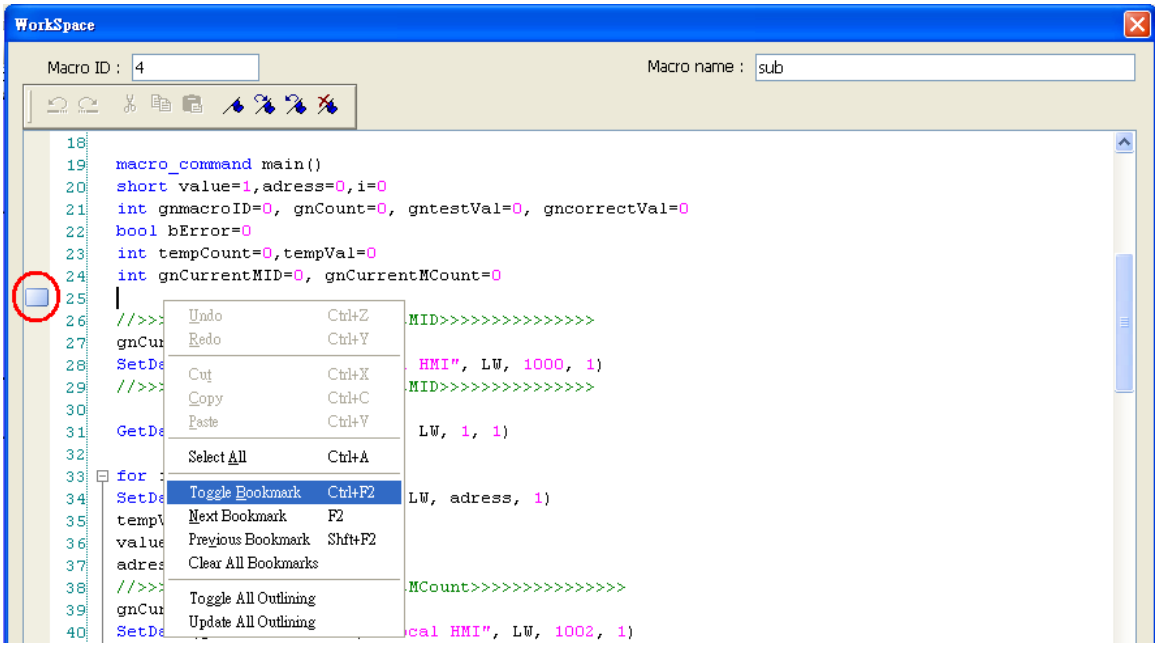


7. Use [Select All] to include all the content in the edit area.



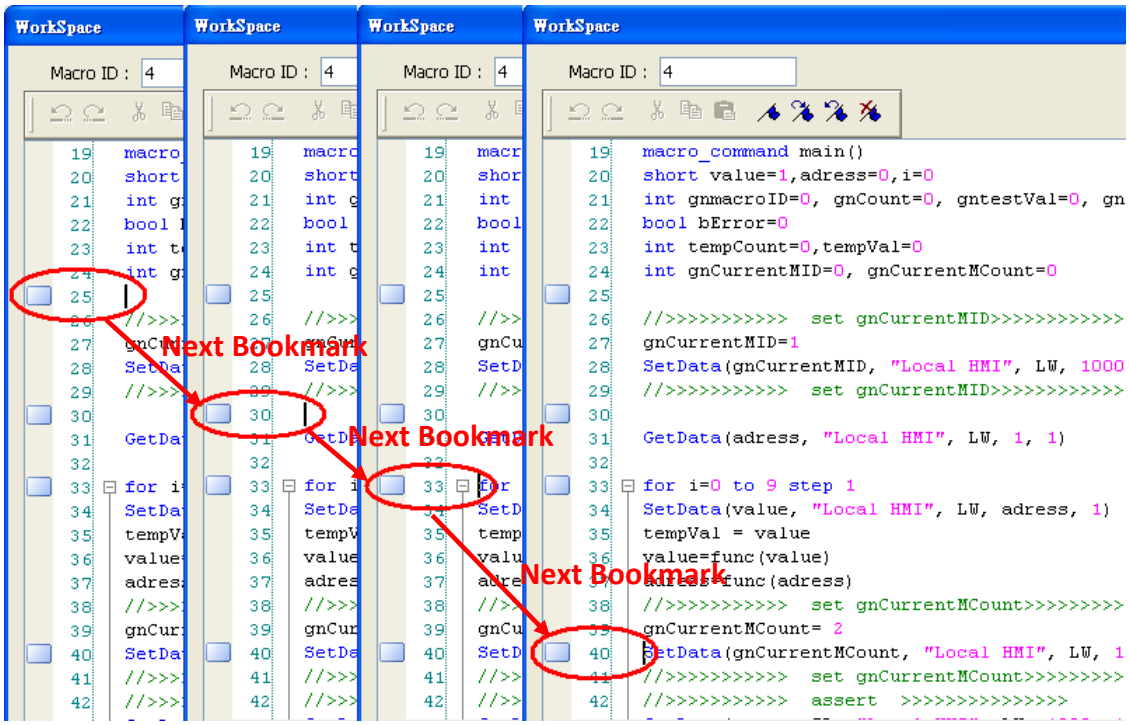
8. If the macro is too long, use bookmarks to manage and read the code with ease. The illustration below shows how it works.

   a. Move your cursor to the position in the edit area where to insert a bookmark. Right click, select [Toggle Bookmark]. There will be a blue little square that represents a bookmark on the left side of edit area.



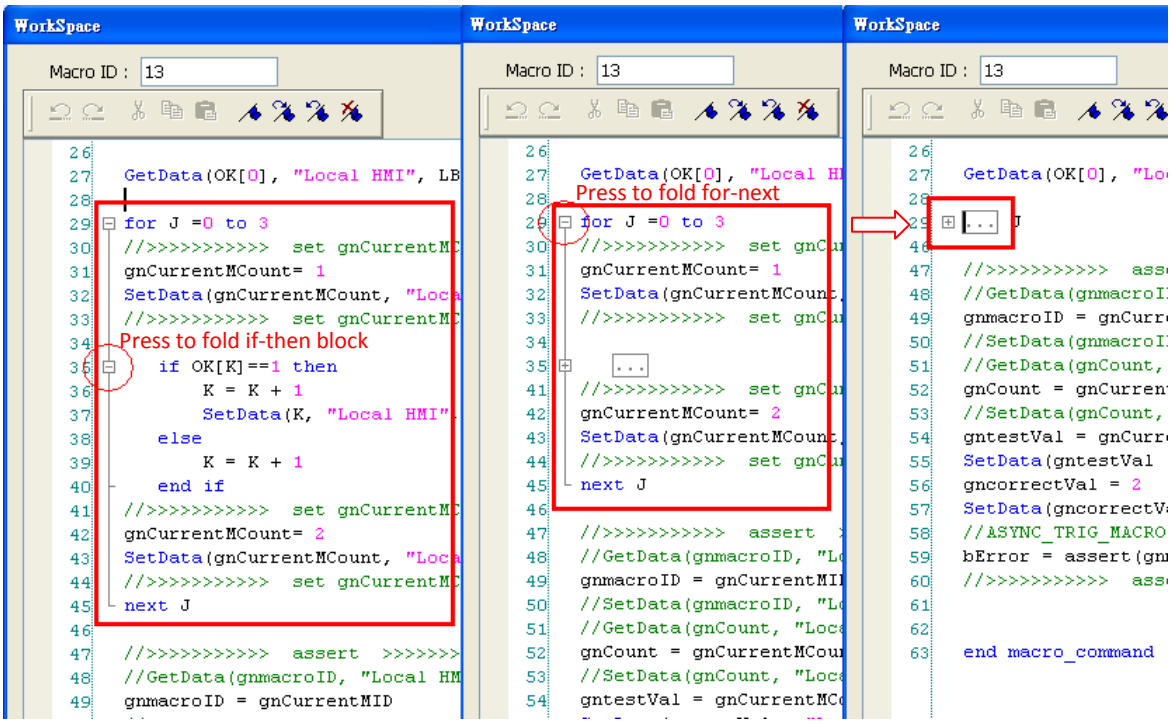
b. If there is already a bookmark where the cursor is placed, select [Toggle Bookmark] to close it, otherwise to open it.

c. Right click and select [Next Bookmark], the cursor will move to where the next bookmark locates. Selecting [Previous Bookmark] will move the cursor to the previous bookmark.

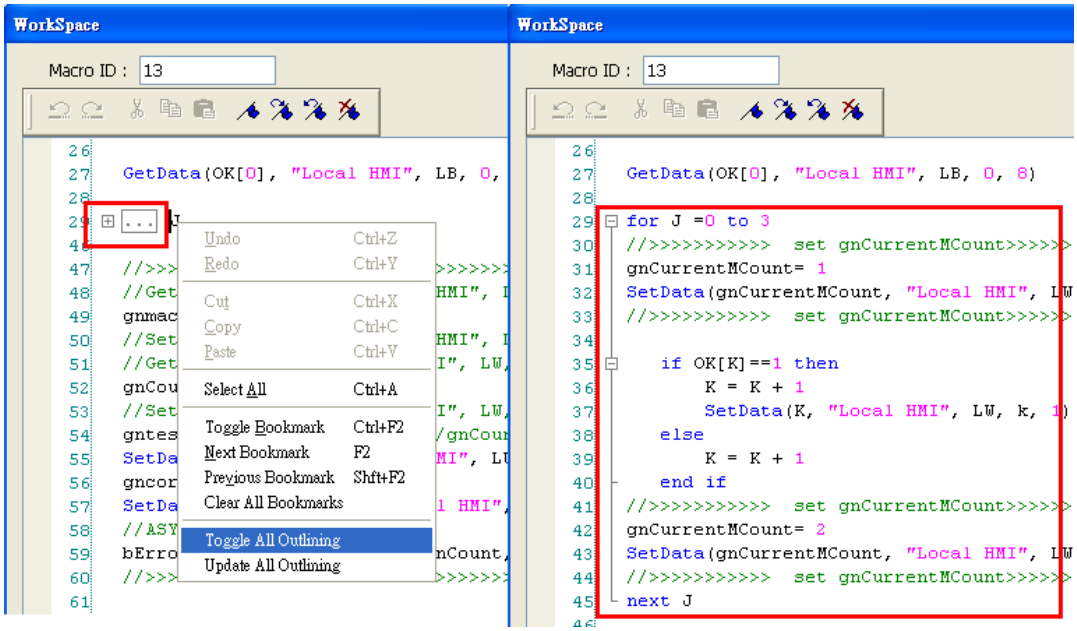


d. Selecting [Clear All Bookmarks] will delete all bookmarks.

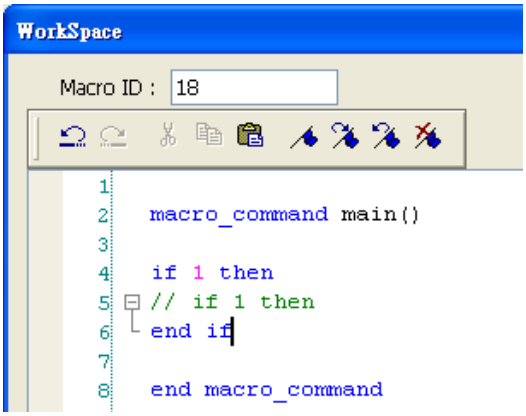
9. Macro editor provides outlining (or code-folding). Outlining will hide macro codes that belong to the same block, and display them as ⊞ ... . There will be a tree diagram on the left side of edit area. Users can click ⊟ to hide the block or ⊞ to open, as shown below:



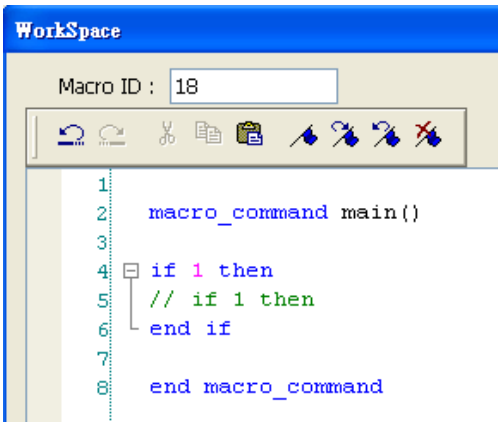
10. Right click to select [Toggle All Outlining] to open all folded macro code blocks.



11. Sometimes the outlining might be incorrect since that the keywords are misjudged. For example:



To solve this problem, right click and select [Update All Outlining].



12. The statements enclosed in the following keywords are called a “block” of the macro code:

a. Function block: sub – end sub

b. Iterative statements:

i. for – next

ii. while – wend

c. Logical statements:

i. if – end if

d. Selective statements: select case – end select

18.2 Macro Construction

A macro contains statements. The statements contain constants, variables and operations. The statements are put in a specific order to create the desired output.

A macro has the following structure:

```
Global Variable Declaration          ---------- Optional

Sub Function Block Declarations      ---------- Optional
    Local Variable Declarations
End Sub

macro_command main()                 ---------- Required
    Local Variable Declarations
    [Statements]
end macro_command                    ---------- Required
```

Macro must have one and only one main function which is the execution start point of macro. The format is:

macro_command main()

end macro_command

Local variables are used within the main macro function or in a defined function block. Its value remains valid only within the specific block.

Global variables are declared before any function blocks and are valid for all functions in the macro. When local variables and global variables have the same declaration of name, only the local variables are valid.

The example below shows a simple macro which includes a variable declaration and a function call.

```
macro_command main()
   short pressure = 10                                  //  local variable declaration
   SetData(pressure, "Allen-Bradley DF1", N7, 0, 1)     //  function calling
end macro_command
```

18.3 Syntax

18.3.1 Constants and Variables

18.3.1.1 Constants

Constants are fixed values and can be directly written into statements. The formats are:

Constant Type	Note	Example
Decimal integer		345, -234, 0, 23456
Hexadecimal	Must begin with 0x	0x3b, 0xffff, 0x237
ASCII	Single character must be enclosed in single quotation marks and a string (group of characters) must be enclosed by double quotation marks.	‘a’, "data", "name"
Boolean		true, false

Here is an example using constants:

```
macro_command main()
short A, B      //   A and B are variables
A = 1234
B = 0x12        //   1234 and 0x12 are constants
end macro_command
```

18.3.1.2 Variables

Variables are names that represent information. The information can be changed as the variable is modified by statements.

Naming Rules for Variables

1. A variable name must start with an alphabet.
2. Variable names longer than 32 characters are not allowed.
3. Reserved words cannot be used as variable names.

There are 8 different Variable types, 5 for signed data types and 3 for unsigned data types:

Variable Type	Description	Range
bool (boolean)	1 bit (discrete)	0, 1
char (character)	8 bits (byte)	+127 to -128
short (short integer)	16 bits (word)	+32767 to -32768
int (integer)	32 bits (double word)	+2147483647to -2147483648
float (floating point)	32 bits (double word)	
unsigned char	8 bits (byte)	0 to 255
unsigned short	16 bits (word)	0 to 65535
unsigned int	32 bits (double word)	0 to 4,294,967,295

Declaring Variables

Variables must be declared before being used. To declare a variable, specify the type before the variable name.

Example:

```
int          a
short        b, switch
float        pressure
unsigned short c
```

Declaring Arrays

Macros support one-dimensional arrays (zero-based index). To declare an array of variables, specify the type and the variable name followed by the number of variables in the array enclosed in brackets “[]”. The length of an array could be 1 to 4096. (Macros only support at most 4096 variables per macro).

Example:

```
int          a[10]
short        b[20], switch[30]
float        pressure[15]
```

The minimum array index is 0 and the maximum is (array size – 1).

Example:

```
char data [100]          //    array size is 100
```

In this case, the minimum of array index is 0 and maximum of array index is 99 (=100-1)

Variable and Array Initialization

There are two ways variables can be initialized:

1. By statement using the assignment operator (=)

Example:

```
int a
float b[3]
a = 10
b[0] = 1
```

2. During declaration

```
char a = '5', b = 9
```

The declaration of arrays is a special case. The entire array can be initialized during declaration by enclosing comma separated values inside curly brackets "{}".

Example:

```
float data[4] = {11, 22, 33, 44}    //     now data[0] is 11, data[1] is 22….
```

18.3.2 Operators

Operators are used to designate how data is manipulated and calculated.

Operator	Description	Example
=	Assignment operator	pressure = 10

Arithmetic Operators	Description	Example
+	Addition	A = B + C
-	Subtraction	A = B – C
*****	Multiplication	A = B * C
/	Division	A = B / C
%	Modulo division (return remainder)	A = B % 5

Comparison Operators	Description	Example
<	Less than	if A < 10 then B = 5
<=	Less than or equal to	if A <= 10 then B = 5
>	Greater than	if A > 10 then B = 5
>=	Greater than or equal to	if A >= 10 then B = 5
==	Equal to	if A == 10 then B = 5
<>	Not equal to	if A <> 10 then B = 5

Logic Operators	Description	Example
and	Logical AND	if A < 10 and B > 5 then C = 10
or	Logical OR	if A >= 10 or B > 5 then C = 10
xor	Logical Exclusive OR	if A xor 256 then B = 5
not	Logical NOT	if not A then B = 5

Shift and bitwise operators are used to manipulate bits of signed/unsigned character and integer variables. The priority of these operators is from left to right within the statement.

Shift Operators	Description	Example
<<	Shifts the bits in a bitset to the left a specified number of positions	A = B << 8
>>	Shifts the bits in a bitset to the right a specified number of positions	A = B >> 8

Bitwise Operators	Description	Example
&	Bitwise AND	A = B & 0xf
|	Bitwise OR	A = B | C
^	Bitwise XOR	A = B ^ C
~	One’s complement	A = ~B

Priority of All Operators

The overall priority of all operations from highest to lowest is as follows:

Operations within parenthesis are carried out first
Arithmetic operations
Shift and Bitwise operations
Comparison operations
Logic operations
Assignment

Reserved Keywords

The following keywords are reserved for system. These keywords cannot be used as variable, array, or function names.

+, -, *, /, %, >=, >, <=, <, <>, ==, and, or, xor, not, <<, >>,=, &, |, ^, ~
exit, macro_command, for, to, down, step, next, return, bool, short, int, char, float, void, if, then, else, break, continue, set, sub, end, while, wend, true, false
SQRT, CUBERT, LOG, LOG10, SIN, COS, TAN, COT, SEC, CSC, ASIN, ACOS, ATAN, BIN2BCD, BCD2BIN, DEC2ASCII, FLOAT2ASCII, HEX2ASCII, ASCII2DEC, ASCII2FLOAT, ASCII2HEX, FILL, RAND, DELAY, SWAPB, SWAPW, LOBYTE, HIBYTE, LOWORD, HIWORD, GETBIT, SETBITON, SETBITOFF, INVBIT, ADDSUM, XORSUM, CRC, INPORT, OUTPORT, POW, GetError, GetData, GetDataEx, SetData, SetDataEx, SetRTS, GetCTS, Beep, SYNC_TRIG_MACRO, ASYNC_TRIG_MACRO, TRACE, FindDataSamplingDate, FindDataSamplingIndex, FindEventLogDate, FindEventLogIndex
StringGet, StringGetEx, StringSet, StringSetEx, StringCopy, StringMid, StringDecAsc2Bin, StringBin2DecAsc, StringDecAsc2Float, StringFloat2DecAsc, StringHexAsc2Bin, StringBin2HexAsc, StringLength, StringCat, StringCompare, StringCompareNoCase, StringFind, StringReverseFind, StringFindOneOf, StringIncluding, StringExcluding, StringToUpper, StringToLower, StringToReverse, StringTrimLeft, StringTrimRight, StringInsert

18.4 Statement

18.4.1 Definition Statement

This covers the declaration of variables and arrays. The formal construction is as follows:

```
type      name
```

This defines a variable with name as “name” and type as “type”.

Example:

```
int A              // define a variable A as an integer
```

```
type      name[constant]
```

This defines an array variable called “name” with size as “constant” and type as “type”.

Example:

```
int B[10]        // where define a variable B as a one-dimensional array of size 10
```

18.4.2 Assignment Statement

Assignment statements use the assignment operator to move data from the expression on the right side of the operator to the variable on the left side. An expression is the combination of variables, constants and operators to yield a value.

```
VariableName = Expression
```

Example

```
A = 2                    where a variable A is assigned to 2
```

18.4.3 Logical Statements

Logical statements perform actions depending on the condition of a boolean expression. The syntax is as follows:

Single-Line Format

```
if <Condition> then
    [Statements]
else
    [Statements]
end if
```

Example:

```
if a == 2 then
    b = 1
else
    b = 2
end if
```

Block Format

```
If <Condition> then
    [Statements]
else if <Condition – n> then
[Statements]
else
    [Statements]
end if
```

Example:

```
if a == 2 then
    b = 1
else if a == 3 then
    b = 2
else
    b = 3
end if
```

Syntax description:

if	Must be used to begin the statement
<Condition>	Required. This is the controlling statement. It is FALSE when the <Condition> evaluates to 0 and TRUE when it evaluates to non- zero.
then	Must precede the statements to execute if the <Condition> evaluates to TRUE.
[Statements]	It is optional in block format but necessary in single-line format without else. The statement will be executed when the <Condition> is TRUE.
else if	Optional. The else if statement will be executed when the relative <Condition-n> is TRUE.
<Condition-n>	Optional. see <Condition>
else	Optional. The else statement will be executed when <Condition> and <Condition-n> are both FALSE.
end if	Must be used to end an if-then statement.

18.4.4 Selective Statements

The select-case construction can be used like multiple if-else statements and perform selected actions depending on the value of the given variable. When the matched value is found, all the actions below will be executed until a break statement is met. The syntax is as follows:

Format without a Default Case

```
Select Case [variable]
Case [value]
    [Statements]
    break
end Select
```

Example:

```
Select Case A
    Case 1
        b=1
    break
end Select
```

Format with a Default Case (Case else)

```
Select Case [variable]
Case [value]
    [Statements]
    break
    Case else
    [Statements]
    break

end Select
```

Example:

```
Select Case A
    Case 1
        b=1
    break
    Case else
        b=0
    break
end Select
```

Multiple cases in the same block

```
Select Case [variable]
Case [value1]
    [Statements]
Case [value2]
    [Statements]
    break

end Select
```

Example:

```
Select Case A
    Case 1
    Case 2
        b=2
    Case 3
        b=3
    break
end Select
```

Syntax description:

Select Case	Must be used to begin the statement
[variable]	Required. The value of this variable will be compared to the value of each case.
Case else	Optional. It represents the default case. If none of the cases above are matched, the statements under default case will be executed. When a default case is absent, it will skip directly to the end of the select-case statements if there is no matched case.
break	Optional. The statements under the matched case will be executed until the break command is reached. If a break command is absent, it simply keeps on executing next statement until the end command is reached.
end Select	Indicates the end of the select-case statements

18.4.5 Iterative Statements

Iterative statements control loops and repetitive tasks depending on condition. There are two types of iterative statements.

18.4.5.1 for-next Statements

The for-next statement runs for a fixed number of iterations. A variable is used as a counter to track the progress and test for ending conditions. Use this for fixed execution counts. The syntax is as follows:

```
for [Counter] = <StartValue> to <EndValue> [step <StepValue>]
    [Statements]
next [Counter]
```

or

```
for [Counter] = <StartValue> down <EndValue> [step <StepValue>]
    [Statements]
next [Counter]
```

Example:

```
for a = 0 to 10 step 2
    b = a
next a
```

Syntax description:

for	Must be used to begin the statement
[Counter]	Required. This is the controlling statement. The result of evaluating the variable is used as a test of comparison.
<StartValue>	Required. The initial value of [Counter]
to/down	Required. This determines if the <step> increments or decrements the <Counter>. “to” increments <Counter> by <StepValue>. “down” decrements <Counter> by <StepValue>.
<EndValue>	Required. The test point. If the <Counter> is greater than this value, the macro exits the loop.
step	Optional. Specifies that a <StepValue> other than one is to be used.
[StepValue]	Optional. The increment/decrement step of <Counter>. It can be omitted when the value is 1 If [step <StepValue>] are omitted the step value defaults to 1.
[Statements]	Optional. Statements to execute when the evaluation is TRUE. “for-next” loops may be nested.
next	Required.
[Counter]	Optional. This is used when nesting for-next loops.

18.4.5.2 while-wend Statements

The while-wend statement runs for an unknown number of iterations. A variable is used to test for ending conditions. When the condition is TRUE, the statements inside are executed repetitively until the condition becomes FALSE. The syntax is as follows.

```
while <Condition>
    [Statements]
wend
```

Example:

```
while a < 10
    a = a + 10
wend
```

Syntax description:

while	Must be used to begin the statement
continue	Required. This is the controlling statement. When it is TRUE, the loop begins execution. When it is FALSE, the loop terminates.
return [value]	Statements to execute when the evaluation is TRUE.
wend	Indicates the end of the while-end statements

18.4.5.3 Other Control Commands

break	Used in for-next and while-wend. It skips immediately to the end of the iterative statement.
continue	Used in for-next and while-wend. It ends the current iteration of a loop and starts the next one.
return	The return command inside the main block can force the macro to stop anywhere. It skips immediately to the end of the main block.

18.5 Function Blocks

Function blocks are useful for reducing repetitive codes. It must be defined before use and supports any variable and statement type. A function block could be called by putting its name followed by parameters in parenthesis. After the function block is executed, it returns the value to the caller function where it is used as an assignment value or as a condition. A return type is not required in function definition, which means that a function block does not have to return a value. The parameters can also be ignored in function definition while the function has no need to take any parameters from the caller. The syntax is as follows:

Function definition with return type:

```
sub type <name> [(parameters)]
    Local variable declarations
    [Statements]
    [return [value]]
end sub
```

Example:

```
sub int Add(int x, int y)
    int result
    result = x +y
    return result
 end sub

macro_command main()
    int a = 10, b = 20, sum
    sum = Add(a, b)
end macro_command
```

or:

```
sub int Add()
    int result, x=10, y=20
    result = x +y
    return result
 end sub
macro_command main()
    int   sum
    sum = Add()
```

```
end macro_command
```

Function definition without return type:

```
sub <name> [(parameters)]
    Local variable declarations
    [Statements]
end sub
```

Example:

```
sub Add(int x, int y)
    int result
    result = x +y
 end sub

macro_command main()
    int   a = 10, b = 20
    Add(a, b)
end macro_command
```

or:

```
sub Add()
    int result, x=10, y=20
    result = x +y
 end sub

macro_command main()
    Add()
end macro_command
```

Syntax description:

sub	Must be used to begin the function block
type	Optional. This is the data type of value that the function returns. A function block is not always necessary to return a value.
(parameters)	Optional. The parameters hold values that are passed to the function. The passed parameters must have their type declared in the parameter field and assigned a variable name. For example: sub int MyFunction(int x, int y). x and y would be integers passed to the function. This function is called by a statement that looks similar to this: ret = MyFunction(456, pressure) where “pressure” must be integer according to the definition of function. Notice that the calling statement can pass hard coded values or variables to the function. After this function is executed, an integer values is return to ‘ret’.
Local variable declaration	Variables that are used in the function block must be declared first. This is in addition to passed parameters. In the above example x and y are variables that the function can used. Global variables are also available for use in function block.
[Statements]	Statements to execute
[return [value]]	Optional. Used to return a value to the calling statement. The value can be a constant or a variable. Return also ends function block execution. A function block is not always necessary to return a value, but, when the return type is defined in the beginning of the definition of function, the return command is needed.
end sub	Must be used to end a function block.

18.6 Built-In Function Block

EasyBuilder has many built-in functions for retrieving and transferring data to the PLC, data management and mathematical functions.

18.6.1 Mathematical Functions

Name	SQRT
Syntax	SQRT(*source*, *result*)
Description	Calculate the square root of *source* and store the result into *result*. *source* can be a constant or a variable. *result* must be a variable. *source* must be a nonnegative value.
Example	macro_command main() float source, result SQRT(15, result) source = 9.0 SQRT(source, result)// result is 3.0 end macro_command

Name	CUBERT
Syntax	CUBERT(*source*, *result*)
Description	Calculate the cube root of source and store the *result* into result. *source* can be a constant or a variable. *result* must be a variable. *source* must be a nonnegative value.

Example	macro_command main()
	float source, result
	
	CUBERT (27, result) // result is 3.0
	
	source = 27.0
	CUBERT(source, result)// result is 3.0

	end macro_command

Name	POW
Syntax	POW(*source1*, *source2*, *result*)
Description	Calculate *source1* to the power of *source2*. *source1* and *source2* can be a constant or a variable. *result* must be a variable. *source1* and *source2* must be a nonnegative value.
Example	macro_command main() float y, result y = 0.5 POW (25, y, result) // result = 5 end macro_command

Name	SIN
Syntax	SIN(*source*, *result*)
Description	Calculate the sine of *source* (degree) into *result*. *source* can be a constant or a variable. *result* must be a variable.

Example	macro_command main()
	float source, result
	
	SIN(90, result)// result is 1
	
	source = 30
	SIN(source, result)// result is 0.5
	
	end macro_command

Name	COS
Syntax	COS(*source*, *result*)
Description	Calculate the cosine of *source* (degree) into *result*. *source* can be a constant or a variable. *result* must be a variable.
Example	macro_command main() float source, result COS(90, result)// result is 0

	source = 60 GetData(source, "Local HMI", LW, 0, 1) COS(source, result)// result is 0.5 end macro_command

Name	TAN
Syntax	TAN(*source*, *result*)
Description	Calculate the tangent of *source* (degree) into *result*. *source* can be a constant or a variable. *result* must be a variable.

Example	macro_command main()
	float source, result
	
	TAN(45, result)// result is 1
	
	source = 60
	TAN(source, result)// result is 1.732
	
	end macro_command

Name	COT
Syntax	COT(*source*, *result*)
Description	Calculate the cotangent of *source* (degree) into *result*. *source* can be a constant or a variable. *result* must be a variable.

Example	macro_command main()
	float source, result
	
	COT(45, result)// result is 1
	
	source = 60
	COT(source, result)// result is 0.5774
	
	end macro_command

Name	SEC
Syntax	SEC(*source*, *result*)
Description	Calculate the secant of *source* (degree) into *result*.

	source can be a constant or a variable. *result* must be a variable.

Example	macro_command main()
	float source, result
	
	SEC(45, result)// result is 1.414
	
	source = 60
	SEC(source, result)// if source is 60, result is 2
	
	end macro_command

Name	CSC
Syntax	CSC(*source*, *result*)
Description	Calculate the cosecant of *source* (degree) into *result*. *source* can be a constant or a variable. *result* must be a variable.

Example	macro_command main()
	float source, result
	
	CSC(45, result)// result is 1.414
	
	source = 30
	CSC(source, result)// result is 2
	
	end macro_command

Name	ASIN
Syntax	ASIN(*source*, *result*)
Description	Calculate the arc sine of *source* into *result* (degree). *source* can be a constant or a variable. *result* must be a variable.

Example	macro_command main()
	float source, result
	
	ASIN(0.8660, result)// result is 60
	
	source = 0.5
	ASIN(source, result)// result is 30

	end macro_command

Name	ACOS
Syntax	ACOS(*source*, *result*)
Description	Calculate the arc cosine of *source* into *result*. *source* can be a constant or a variable. *result* must be a variable.

Example	macro_command main()
	float source, result
	
	ACOS(0.8660, result)// result is 30
	
	source = 0.5
	ACOS(source, result)// result is 60
	
	end macro_command

Name	ATAN
Syntax	ATAN(*source*, *result*)
Description	Calculate the arc tangent of *source* into *result*. *source* can be a constant or a variable. *result* must be a variable.

Example	macro_command main()
	float source, result
	
	ATAN(1, result)// result is 45
	
	source = 1.732
	ATAN(source, result)// result is 60
	
	end macro_command

Name	LOG
Syntax	LOG (*source*, *result*)
Description	Calculates the natural logarithm of a number. *source* can be either a variable or a constant. *result* must be a variable.
Example	macro_command main() float source = 100, result

	LOG (source, result)// result is approximately 4.6052 end macro_command

Name	LOG10
Syntax	LOG10(*source*, *result*)
Description	Calculates the base-10 logarithm of a number. *source* can be either a variable or a constant. *result* must be a variable.
Example	macro_command main() float source = 100, result LOG10 (source, result) // result is 2 end macro_command

Name	RAND
Syntax	RAND(result)
Description	Calculates a random integer and save into *result*. *result* must be a variable.
Example	macro_command main() short result RAND (result) //result is not a fixed value when executes macro every time end macro_command

18.6.2 Data Transformation

Name	BIN2BCD
Syntax	BIN2BCD(*source*, *result*)
Description	Transforms a binary-type value (*source*) into a BCD-type value (*result*). *source* can be a constant or a variable. result must be a variable.
Example	macro_command main()

	short source, result
	
	BIN2BCD(1234, result)// result is 0x1234
	
	source = 5678
	BIN2BCD(source, result)// result is 0x5678
	
	end macro_command

Name	BCD2BIN
Syntax	BCD2BIN(*source*, *result*)
Description	Transforms a BCD-type value (*source*) into a binary-type value (*result*). *source* can be a constant or a variable. *result* must be a variable.

Example	macro_command main()
	
	short source, result
	
	BCD2BIN(0x1234, result)// result is 1234
	
	source = 0x5678
	BCD2BIN(source, result)// result is 5678
	
	end macro_command

Name	DEC2ASCII
Syntax	DEC2ASCII(*source*, *result[start]*, *len*)
Description	Transforms a decimal value (*source*) into an ASCII string and save it to an array (*result*). *len* represents the length of the string and the unit of length depends on result's type., i.e. if result's type is "char" (the size is byte), the length of the string is (byte * *len*). If result's type is "short" (the size is word), the length of the string is (word * *len*), and so on. The first character is put into *result[start]*, the second character is put into *result[start + 1]*, and the last character is put into result[start + (len -1)]. *source* and *len* can be a constant or a variable. *result* must be a variable.

	start must be a constant.

Example	macro_command main()
	short source
	char result1[4]
	short result2[4]
	
	source = 5678
	DEC2ASCII(source, result1[0], 4)
	// result1[0] is '5', result1[1] is '6', result1[2] is '7', result1[3] is '8'
	// the length of the string (result1) is 4 bytes(= 1 * 4)
	
	DEC2ASCII(source, result2[0], 4)
	// result2[0] is '5', result2[1] is '6', result2[2] is '7', result2[3] is '8'
	// the length of the string (result2) is 8 bytes(= 2 * 4)
	
	end macro_command

Name	HEX2ASCII
Syntax	HEX2ASCII(*source*, *result[start]*, *len*)
Description	Transforms a hexadecimal value (*source*) into ASCII string saved to an array (*result*). *len* represents the length of the string and the unit of length depends on result's type., i.e. if result's type is "char" (the size is byte), the length of the string is (byte * *len*). If result's type is "short" (the size is word), the length of the string is (word * *len*), and so on. *source* and *len* can be a constant or a variable. *result* must be a variable. *start* must be a constant.
Example	macro_command main() short source char result[4] source = 0x5678 HEX2ASCII (source, result[0], 4) // result[0] is '5', result[1] is '6', result[2] is '7', result[3] is '8' end macro_command

Name	FLOAT2ASCII

Syntax	FLOAT2ASCII(*source*, *result[start]*, *len*)
Description	Transforms a floating value (*source*) into ASCII string saved to an array (*result*). *len* represents the length of the string and the unit of length depends on result's type., i.e. if result's type is "char" (the size is byte), the length of the string is (byte * *len*). If result's type is "short" (the size is word), the length of the string is (word * *len*), and so on. *source* and *len* can be a constant or a variable. *result* must be a variable. *start* must be a constant.
Example	macro_command main() float source char result[4] source = 56.8 FLOAT2ASCII (source, result[0], 4) // result[0] is '5', result[1] is '6', result[2] is '.', result[3] is '8' end macro_command

Name	ASCII2DEC
Syntax	ASCII2DEC(*source[start], result, len*)
Description	Transforms a string (*source*) into a decimal value saved to a variable (*result*). The length of the string is *len*. The first character of the string is *source[start].* *source* and *len* can be a constant or a variable. *result* must be a variable. *start* must be a constant.
Example	macro_command main() char source[4] short result source[0] = '5' source[1] = '6' source[2] = '7' source[3] = '8' ASCII2DEC(source[0], result, 4) // result is 5678

	end macro_command

Name	ASCII2HEX
Syntax	ASCII2HEX (*source[start], result, len*)
Description	Transforms a string (source) into a hexadecimal value saved to a variable (result). The length of the string is *len*. The first character of the string is *source[start].* *source* and *len* can be a constant or a variable. *result* must be a variable. *start* must be a constant.
Example	macro_command main() char source[4] short result source[0] = '5' source[1] = '6' source[2] = '7' source[3] = '8' ASCII2HEX (source[0], result, 4) // result is 0x5678 end macro_command

Name	ASCII2FLOAT
Syntax	ASCII2FLOAT(*source[start], result, len*)
Description	Transforms a string (*source*) into a float value saved to a variable (result). The length of the string is len. The first character of the string is *source[start]*. *source* and *len* can be a constant or a variable. *result* must be a variable. *start* must be a constant.
Example	macro_command main() char source[4] float result source[0] = '5' source[1] = '6' source[2] = '.'

	source[3] = '8' ASCII2FLOAT (source[0], result, 4) // result is 56.8 end macro_command

18.6.3 Data Manipulation

Name	FILL
Syntax	FILL(*source[start], preset, count*)
Description	Sets the first *count* elements of an array (*source*) to a specified value (*preset*). *source* and start must be a variable, and *preset* can be a constant or variable.

Example	macro_command main()
	char result[4]
	char preset
	
	FILL(result[0], 0x30, 4)
	// result[0] is 0x30, result[1] is 0x30, , result[2] is 0x30, , result[3] is 0x30
	
	preset = 0x31
	FILL(result[0], preset, 2) // result[0] is 0x31, result[1] is 0x31
	
	end macro_command

Name	SWAPB
Syntax	SWAPB(*source*, *result*)
Description	Exchanges the high-byte and low-byte data of a 16-bit *source* into *result*. *source* can be a constant or a variable. *result* must be a variable.

Example	macro_command main()
	short source, result
	
	SWAPB(0x5678, result)// result is 0x7856
	
	source = 0x123
	SWAPB(source, result)// result is 0x2301
	
	end macro_command

Name	SWAPW
Syntax	SWAPW(*source*, *result*)
Description	Exchanges the high-word and low-word data of a 32-bit *source* into *result*. *source* can be a constant or a variable. *result* must be a variable.

Example	macro_command main()
	int source, result
	
	SWAPW (0x12345678, result)// result is 0x56781234
	
	source = 0x12345
	SWAPW (source, result)// result is 0x23450001
	
	end macro_command

Name	LOBYTE
Syntax	LOBYTE(*source*, *result*)
Description	Retrieves the low byte of a 16-bit source into result. *source* can be a constant or a variable. *result* must be a variable.

Example	macro_command main()
	short source, result
	
	LOBYTE(0x1234, result)// result is 0x34
	
	source = 0x123
	LOBYTE(source, result)// result is 0x23
	
	end macro_command

Name	HIBYTE
Syntax	HIBYTE(*source*, *result*)
Description	Retrieves the high byte of a 16-bit source into result. *source* can be a constant or a variable. *result* must be a variable.
Example	macro_command main() short source, result

	HIBYTE(0x1234, result)// result is 0x12
	
	source = 0x123
	HIBYTE(source, result)// result is 0x01
	
	end macro_command

Name	LOWORD
Syntax	LOWORD(*source*, *result*)
Description	Retrieves the low word of a 32-bit source into result. *source* can be a constant or a variable. *result* must be a variable.

Example	macro_command main()
	int source, result
	
	LOWORD(0x12345678, result)// result is 0x5678
	
	source = 0x12345
	LOWORD(source, result)// result is 0x2345
	
	end macro_command

Name	HIWORD
Syntax	HIWORD(*source*, *result*)
Description	Retrieves the high word of a 32-bit source into result. *source* can be a constant or a variable. *result* must be a variable.

Example	macro_command main()
	int source, result
	
	HIWORD(0x12345678, result)// result is 0x1234
	
	source = 0x12345
	HIWORD(source, result)// result is 0x0001
	
	end macro_command

18.6.4 Bit Transformation

Name	GETBIT
Syntax	GETBIT(*source*, *result*, *bit_pos*)
Description	Gets the state of designated bit position of a data (*source*) into result. *result* value will be 0 or 1. *source* and *bit_pos* can be a constant or a variable. *result* must be a variable.

Example	macro_command main()
	int source, result
	short bit_pos
	
	GETBIT(9, result, 3)// result is 1
	
	source = 4
	bit_pos = 2
	GETBIT(source, result, bit_pos)// result is 1
	
	end macro_command

Name	SETBITON
Syntax	SETBITON(*source, result, bit_pos*)
Description	Changes the state of designated bit position of a data (*source*) to 1, and put changed data into *result*. *source* and *bit_pos* can be a constant or a variable. *result* must be a variable.

Example	macro_command main()
	int source, result
	short bit_pos
	
	SETBITON(1, result, 3)// result is 9
	
	source = 0
	bit_pos = 2
	SETBITON (source, result, bit_pos)// result is 4

	end macro_command

Name	SETBITOFF
Syntax	SETBITOFF(*source*, *result*, *bit_pos*)
Description	Changes the state of designated bit position of a data (*source*) to 0, and put in changed data into *result*. *source* and *bit_pos* can be a constant or a variable. *result* must be a variable.

Example	macro_command main()
	int source, result
	short bit_pos
	
	SETBITOFF(9, result, 3)// result is 1
	
	source = 4
	bit_pos = 2
	SETBITOFF(source, result, bit_pos)// result is 0
	
	end macro_command

Name	INVBIT
Syntax	INVBIT(*source, result, bit_pos*)
Description	Inverts the state of designated bit position of a data (*source*), and put changed data into *result*. *source* and *bit_pos* can be a constant or a variable. *result* must be a variable.

Example	macro_command main()
	int source, result
	short bit_pos
	
	INVBIT(4, result, 1)// result = 6
	
	source = 6
	bit_pos = 1
	INVBIT(source, result, bit_pos)// result = 4

	end macro_command

18.6.5 Communication

Name	DELAY
Syntax	DELAY(*time*)
Description	Suspends the execution of the current macro for at least the specified interval (*time*). The unit of *time* is millisecond. *time* can be a constant or a variable.

Example	macro_command main()
	int time == 500
	
	DELAY(100)// delay 100 ms
	DELAY(time)// delay 500 ms
	
	end macro_command

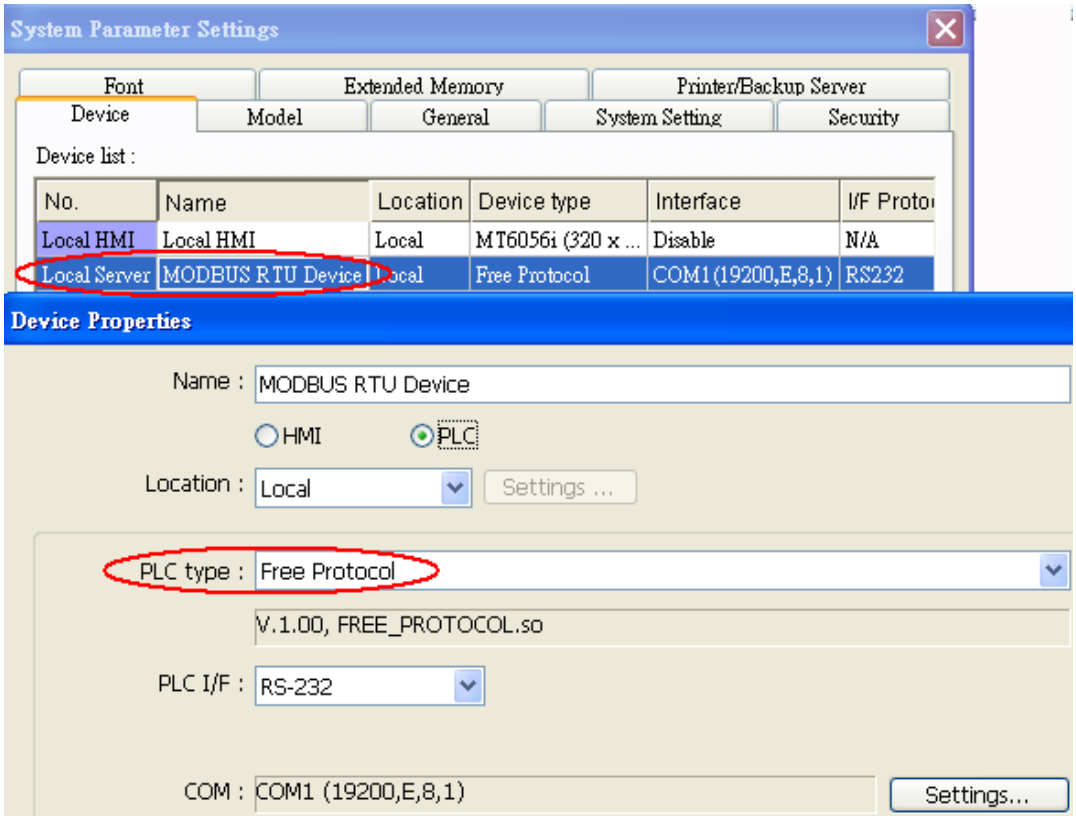
Name	ADDSUM
Syntax	ADDSUM(*source[start]*, *result*, *data_count*)
Description	Adds up the elements of an array (*source*) from *source[start]* to *source*[*start* + *data_count* - 1] to generate a checksum. Puts in the checksum into *result*. *result* must be a variable. *data_count* is the amount of the accumulated elements and can be a constant or a variable.
Example	macro_command main() char data[5] short checksum data[0] = 0x1 data[1] = 0x2 data[2] = 0x3 data[3] = 0x4 data[4] = 0x5

	ADDSUM(data[0], checksum, 5)// checksum is 0xf end macro_command

Name	XORSUM
Syntax	XORSUM(*source*[*start*], *result*, *data_count*)
Description	Uses an exclusion method to calculate the checksum from *source*[*start*] to *source*[*start* + *data_count* - 1]. Puts the checksum into *result*. *result* must be a variable. *data_count* is the amount of the calculated elements of the array and can be a constant or a variable.
Example	macro_command main() char data[5] = {0x1, 0x2, 0x3, 0x4, 0x5} short checksum XORSUM(data[0], checksum, 5)// checksum is 0x1 end macro_command

Name	CRC
Syntax	CRC(*source*[*start*], *result*, *data_count*)
Description	Calculates 16-bit CRC of the variables from *source*[*start*] to *source*[*start* + *data_count* - 1]. Puts in the 16-bit CRC into *result*. *result* must be a variable. *data_count* is the amount of the calculated elements of the array and can be a constant or a variable.
Example	macro_command main() char data[5] = {0x1, 0x2, 0x3, 0x4, 0x5} short 16bit_CRC CRC(data[0], 16bit_CRC, 5)// 16bit_CRC is 0xbb2a end macro_command

Name	OUTPORT

Syntax	OUTPORT(*source*[*start*], *device_name*, *data_count*)
Description	Sends out the specified data from *source*[*start*] to *source*[*start* + *count* -1] to PLC via a COM port or the ethernet. *device_name* is the name of a device defined in the device table and the device must be a “Free Protocol”-type device. *data_count* is the amount of sent data and can be a constant or a variable.
Example	To use an OUTPORT function, a “Free Protocol” device must be created first as follows:  The device is named ”MODBUS RTU Device”. The port attribute depends on the setting of this device. (the current setting is “19200,E, 8, 1”) Below is an example of executing an action of writing single coil (SET ON) to a MODBUS device. macro_command main() char command[32] short address, checksum

```
FILL(command[0], 0, 32)//    command initialization

command[0] = 0x1//    station no
command[1] = 0x5//    function code : Write Single Coil

address = 0
HIBYTE(address, command[2])
LOBYTE(address, command[3])

command[4] = 0xff//    force bit on
command[5] = 0

CRC(command[0], checksum, 6)

LOBYTE(checksum, command[6])
HIBYTE(checksum, command[7])

//    send out a “Write Single Coil” command
OUTPORT(command[0], "MODBUS RTU Device", 8)

end macro_command
```

Name	INPORT
Syntax	INPORT(*read_data*[*start*], *device_name*, *read_count*, *return_value*)
Description	Reads data from a COM port or the ethernet. These data is stored to *read_data*[*start*]~ *read_data*[*start* + *read_count* - 1]. *device_name* is the name of a device defined in the device table and the device must be a “Free Protocol”-type device. *read_count* is the required amount of reading and can be a constant or a variable. If the function is used successfully to get sufficient data, *return_value* is 1, otherwise is 0.
Example	Below is an example of executing an action of reading holding registers of a MODBUS device. // Read Holding Registers macro_command main()

```
char command[32], response[32]
short address, checksum
short read_no, return_value, read_data[2]

FILL(command[0], 0, 32)//    command initialization
FILL(response[0], 0, 32)

command[0] = 0x1//    station no
command[1] = 0x3//    function code : Read Holding Registers

address = 0
HIBYTE(address, command[2])
LOBYTE(address, command[3])

read_no = 2//    read 2 words (4x_1 and 4x_2)
HIBYTE(read_no, command[4])
LOBYTE(read_no, command[5])

CRC(command[0], checksum, 6)

LOBYTE(checksum, command[6])
HIBYTE(checksum, command[7])

//    send out a ‘Read Holding Registers” command
OUTPORT(command[0], "MODBUS RTU Device", 8)

//    read responses for a ‘Read Holding Registers” command
INPORT(response[0], "MODBUS RTU Device", 9, return_value)

if return_value > 0 then
read_data[0] = response[4] + (response[3] << 8)//    data in 4x_1
read_data[1] = response[6] + (response[5] << 8)//    data in 4x_2

SetData(read_data[0], "Local HMI", LW, 100, 2)
end if

end macro_command
```

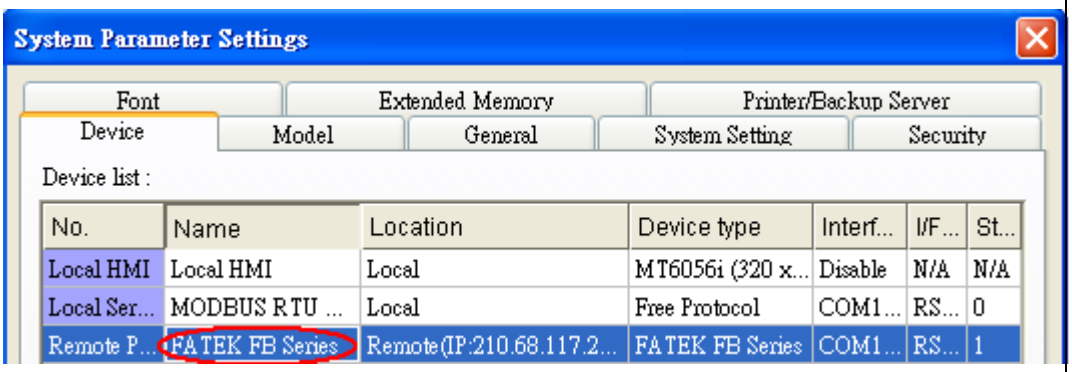
Name	INPORT2
Syntax	INPORT2(*response*[*start*], *device_name*, *receive_len*, *wait_time*)
Description	Read data from a communication port (COM Port or Ethernet Port). The data read will be saved in *response*. The description of *device_name* is the same as OUTPORT. *receive_len* stores the length of the data received, this must be a variable. *receive_len* total length can't exceed the size of *response*. *wait_time* (in millisecond) can be a constant or variable. After the data is read, if there's no upcoming data during the designated time interval, the function returns.

Example	macro_command main()
	
	short wResponse[6], receive_len, wait_time=20
	
	INPORT2(wResponse[0], "Free Protocol", receive_len, wait_time)
	// wait_time unit : millisecond
	
	if receive_len > 0 then
	SetData(wResponse[0], "Local HMI", LW, 0, 6)
	// set responses to LW0
	end if
	
	end macro_command

Name	GetData
Syntax	GetData(*read_data*[*start*], *device_name*, *device_type*, *address_offset*, *data_count*) or GetData(*read_data, device_name, device_type, address_offset, 1*)
Description	Receives data from the PLC. Data is stored into *read_data*[*start*]~ *read_data*[*start* + *data_count* - 1]. *data_count* is the amount of received data. In general, *read_data* is an array, but if *data_count* is 1, *read_data* can be an array or an ordinary variable. Below are two methods to read one word data from PLC. macro_command main()

short read_data_1[2], read_data_2
GetData(read_data_1[0], “FATEK KB Series”, RT, 5, 1)
GetData(read_data_2, “FATEK KB Series”, RT, 5, 1)
end macro_command

Device_name is the PLC name enclosed in the double quotation marks (“) and this name has been defined in the device list of system parameters as follows (see FATEK KB Series):

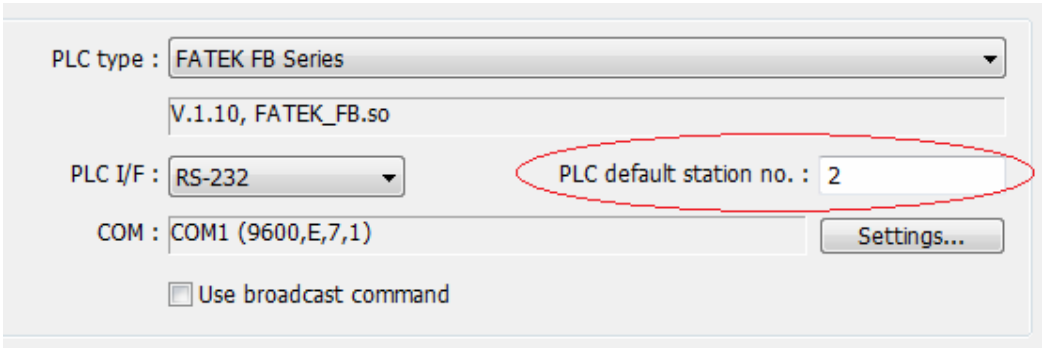


Device_type is the device type and encoding method (binary or BCD) of the PLC data. For example, if device_type is LW_BIN, it means the register is LW and the encoding method is binary. If use BIN encoding method, “_BIN” can be ignored.

If device_type is LW_BCD, it means the register is LW and the encoding method is BCD.

Address_offset is the address offset in the PLC.
For example, GetData(read_data_1[0], “FATEK KB Series”, RT, 5, 1) represents that the address offset is 5.

If address_offset uses the format – “N#AAAAA”, N indicates that PLC’s station number is N. AAAAA represents the address offset. This format is used while multiple PLCs or controllers are connected to a single serial port. For example, GetData(read_data_1[0], “FATEK KB Series”, RT, 2#5, 1) represents that the PLC’s station number is 2. If GetData() uses the default station number defined in the device list as follows, it is not necessary to define station number in address_offset.



The number of registers actually read from depends on both the type of the read_data variable and the value of the number of data_count.

type of read_data	data_count	actual number of 16-bit register read
char (8-bit)	1	1
char (8-bit)	2	1
bool (8-bit)	1	1
bool (8-bit)	2	1
short (16-bit)	1	1
short (16-bit)	2	2
int (32-bit)	1	2
int (32-bit)	2	4
float (32-bit)	1	2
float (32-bit)	2	4

When a GetData() is executed using a 32-bit data type (int or float), the function will automatically convert the data. For example,

```
macro_command main()
float f
GetData(f, "MODBUS", 6x, 2, 1)      // f will contain a floating point value
end macro_command
```

Example

```
macro_command main()
bool a
```

	bool b[30]
	short c
	short d[50]
	int e
	int f[10]
	double g[10]
	
	// get the state of LB2 to the variable a
	GetData(a, “Local HMI”, LB, 2, 1)
	
	// get 30 states of LB0 ~ LB29 to the variables b[0] ~ b[29]
	GetData(b[0], “Local HMI”, LB, 0, 30)
	
	// get one word from LW2 to the variable c
	GetData(c, “Local HMI”, LW, 2, 1)
	
	// get 50 words from LW0 ~ LW49 to the variables d[0] ~ d[49]
	GetData(d[0], “Local HMI”, LW, 0, 50)
	
	// get 2 words from LW6 ~ LW7 to the variable e
	// note that the type of e is int
	GetData(e, “Local HMI”, LW, 6, 1)
	
	// get 20 words (10 integer values) from LW0 ~ LW19 to variables f[0] ~ f[9]
	// since each integer value occupies 2 words
	GetData(f[0], “Local HMI”, LW, 0, 10)
	
	// get 2 words from LW2 ~ LW3 to the variable f
	GetData(f, “Local HMI”, LW, 2, 1)
	
	end macro_command

Name	GetDataEx
Syntax	GetDataEx(*read_data*[*start*], *device_name*, *device_type, address_offset, data_count*) or

	GetDataEx(*read_data, device_name, device_type, address_offset, 1*)
Description	Receives data from the PLC and continue executing next command even if no response from this device. Descriptions of *read_data, device_name, device_type, address_offset* and *data_count* are the same as GetData.

Example	macro_command main()
	bool a
	bool b[30]
	short c
	short d[50]
	int e
	int f[10]
	double g[10]
	
	// get the state of LB2 to the variable a
	GetDataEx (a, “Local HMI”, LB, 2, 1)
	
	// get 30 states of LB0 ~ LB29 to the variables b[0] ~ b[29]
	GetDataEx (b[0], “Local HMI”, LB, 0, 30)
	
	// get one word from LW2 to the variable c
	GetDataEx (c, “Local HMI”, LW, 2, 1)
	
	// get 50 words from LW0 ~ LW49 to the variables d[0] ~ d[49]
	GetDataEx (d[0], “Local HMI”, LW, 0, 50)
	
	// get 2 words from LW6 ~ LW7 to the variable e
	// note that he type of e is int
	GetDataEx (e, “Local HMI”, LW, 6, 1)
	
	// get 20 words (10 integer values) from LW0 ~ LW19 to f[0] ~ f[9]
	// since each integer value occupies 2 words
	GetDataEx (f[0], “Local HMI”, LW, 0, 10)
	
	// get 2 words from LW2 ~ LW3 to the variable f
	GetDataEx (f, “Local HMI”, LW, 2, 1)

	end macro_command

Name	SetData
Syntax	SetData(*send_data[start], device_name, device_type, address_offset, data_count*) or SetData(*send_data, device_name, device_type, address_offset, 1*)
Description	Send data to the PLC. Data is defined in *send_data*[*start*]~ *send_data*[*start* + *data_count* - 1]. *data_count* is the amount of sent data. In general, *send_data* is an array, but if *data_count* is 1, *send_data* can be an array or an ordinary variable. Below are two methods to send one word data. macro_command main() short send_data_1[2] = { 5, 6}, send_data_2 = 5 SetData(send_data_1[0], “FATEK KB Series”, RT, 5, 1) SetData(send_data_2, “FATEK KB Series”, RT, 5, 1) end macro_command *device_name* is the PLC name enclosed in the double quotation marks (“) and this name has been defined in the device list of system parameters. *device_type* is the device type and encoding method (binary or BCD) of the PLC data. For example, if *device_type* is LW_BIN, it means the register is LW and the encoding method is binary. If use BIN encoding method, “_BIN” can be ignored. If *device_type* is LW_BCD, it means the register is LW and the encoding method is BCD. *address_offset* is the address offset in the PLC. For example, SetData(read_data_1[0], “FATEK KB Series”, RT, 5, 1) represents that the address offset is 5. If *address_offset* uses the format – “N#AAAAA”, N indicates that PLC’s station number is N. AAAAA represents the address offset. This format is used while multiple PLCs or controllers are connected to a single serial port. For example, SetData(read_data_1[0], “FATEK KB Series”, RT, 2#5,

1) represents that the PLC's station number is 2. If SetData () uses the default station number defined in the device list, it is not necessary to define station number in address_offset.

The number of registers actually sends to depends on both the type of the send_data variable and the value of the number of data_count.

type of read_data	data_count	actual number of 16-bit register send
char (8-bit)	1	1
char (8-bit)	2	1
bool (8-bit)	1	1
bool (8-bit)	2	1
short (16-bit)	1	1
short (16-bit)	2	2
int (32-bit)	1	2
int (32-bit)	2	4
float (32-bit)	1	2
float (32-bit)	2	4

When a SetData() is executed using a 32-bit data type (int or float), the function will automatically send int-format or float-format data to the device. For example,

```
macro_command main()
float f = 2.6
SetData(f, "MODBUS", 6x, 2, 1)      // will send a floating point value to the device
end macro_command
```

Example

```
macro_command main()
int i
bool a = true
bool b[30]
short c = false
short d[50]
```

```
int e = 5
int f[10]

for i = 0 to 29
b[i] = true
next i

for i = 0 to 49
d[i] = i * 2
next i

for i = 0 to 9
f [i] = i * 3
next i

//   set the state of LB2
SetData(a, “Local HMI”, LB, 2, 1)

//   set the states of LB0 ~ LB29
SetData(b[0], “Local HMI”, LB, 0, 30)

//   set the value of LW2
SetData(c, “Local HMI”, LW, 2, 1)

//   set the values of LW0 ~ LW49
SetData(d[0], “Local HMI”, LW, 0, 50)

//   set the values of LW6 ~ LW7,   note that the type of e is int
SetData(e, “Local HMI”, LW, 6, 1)

//   set the values of LW0 ~ LW19
// 10 integers equal to 20 words, since each integer value occupies 2 words.
SetData(f[0], “Local HMI”, LW, 0, 10)

end macro_command
```

Name	SetDataEx
Syntax	SetDataEx (*send_data[start], device_name, device_type, address_offset, data_count*) or SetDataEx (*send_data, device_name, device_type, address_offset, 1*)
Description	Send data to the PLC and continue executing next command even if no response from this device. Descriptions of *send_data, device_name, device_type, address_offset and data_count* are the same as SetData.

Example	macro_command main()
	int i
	bool a = true
	bool b[30]
	short c = false
	short d[50]
	int e = 5
	int f[10]
	
	for i = 0 to 29
	b[i] = true
	next i
	
	for i = 0 to 49
	d[i] = i * 2
	next i
	
	for i = 0 to 9
	f [i] = i * 3
	next i
	
	// set the state of LB2
	SetDataEx (a, “Local HMI”, LB, 2, 1)
	
	// set the states of LB0 ~ LB29
	SetDataEx (b[0], “Local HMI”, LB, 0, 30)
	
	// set the value of LW2

	SetDataEx (c, “Local HMI”, LW, 2, 1)
	
	// set the values of LW0 ~ LW49
	SetDataEx (d[0], “Local HMI”, LW, 0, 50)
	
	// set the values of LW6 ~ LW7, note that the type of e is int
	SetDataEx (e, “Local HMI”, LW, 6, 1)
	
	// set the values of LW0 ~ LW19
	// 10 integers equal to 20 words, since each integer value occupies 2 words.
	SetDataEx (f[0], “Local HMI”, LW, 0, 10)
	
	end macro_command

Name	GetError
Syntax	GetError (*err*)
Description	Get an error code.

Example	macro_command main()
	short err
	char byData[10]
	
	GetDataEx(byData[0], “MODBUS RTU”, 4x, 1, 10)// read 10 bytes
	
	// if err is equal to 0, it is successful to execute GetDataEx()
	GetErr(err)// save an error code to err
	
	end macro_command

Name	PURGE
Syntax	PURGE (*com_port*)
Description	*com_port* refers to the COM port number which ranges from 1 to 3. It can be either a variable or a constant. This function is used to clear the input and output buffers associated with the COM port.

Example	macro_command main() int com_port=3 PURGE (com_port) PURGE (1) end macro_command

Name	SetRTS
Syntax	SetRTS(*com_port, source*)
Description	Set RTS state for RS232. *com_port* refers to the COM port number. It can be either a variable or a constant. *source* can be either a variable or a constant. This command raise RTS signal while the value of *source* is greater than 0 and lower RTS signal while the value of *source* equals to 0.

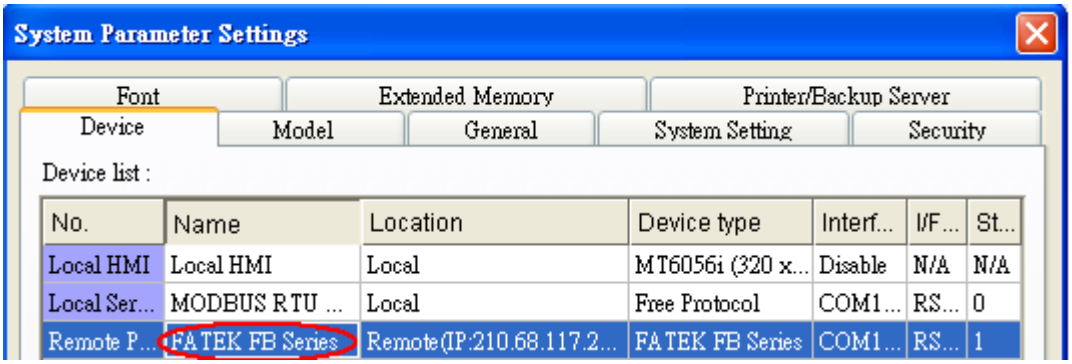
Example	macro_command main()
	char com_port=1
	char value=1
	
	SetRTS(com_port, value) // raise RTS signal of COM1 while value>0
	
	SetRTS(1, 0) // lower RTS signal of COM1
	
	end macro_command

Name	GetCTS
Syntax	GetCTS(*com_port*, *result*)
Description	Get CTS state for RS232. *com_port* refers to the COM port number. It can be either a variable or a constant. *result* is used for receiving the CTS signal. It must be a variable. This command receives CTS signal and stores the received data in the *result* variable. When the CTS signal is pulled high, it writes 1 to *result*, otherwise, it writes 0.
Example	macro_command main() char com_port=1 char result

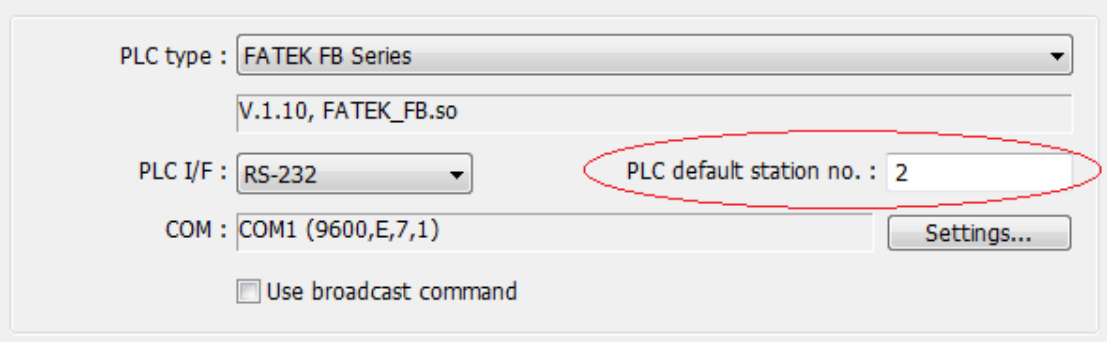
	GetCTS(com_port, result) // get CTS signal of COM1
	
	GetCTS (1, result) // get CTS signal of COM1
	
	end macro_command

Name	Beep
Syntax	Beep()
Description	Plays beep sound. This command plays a beep sound with frequency of 800 hertz and duration of 30 milliseconds.
Example	macro_command main() Beep() end macro_command

18.6.6 String Operation Functions

Name	StringGet
Syntax	StringGet(read_data[start], device_name, device_type, address_offset, data_count)
Description	Receives data from the PLC. The String data is stored into read_data[start]~ read_data[start + data_count - 1]. read_data must be a one-dimensional char array. Data_count is the number of received characters, it can be either a constant or a variable. Device_name is the PLC name enclosed in the double quotation marks (“) and this name has been defined in the device list of system parameters as follows (see FATEK KB Series):  Device_type is the device type and encoding method (binary or BCD) of the PLC data. For example, if device_type is LW_BIN, it means the register is LW and the encoding method is binary. If use BIN encoding method, “_BIN” can be ignored. If device_type is LW_BCD, it means the register is LW and the encoding method is BCD. Address_offset is the address offset in the PLC. For example, StringGet(read_data_1[0], “FATEK KB Series”, RT, 5, 1) represents that the address offset is 5. If address_offset uses the format – “N#AAAAA”, N indicates that PLC’s station

number is N. AAAAA represents the address offset. This format is used while multiple PLCs or controllers are connected to a single serial port. For example, StringGet(read_data_1[0], “FATEK KB Series”, RT, 2#5, 1) represents that the PLC’s station number is 2. If StringGet() uses the default station number defined in the device list as follows, it is not necessary to define station number in address_offset.



The number of registers actually read from depends on the value of the number of data_count since that the read_data is restricted to char array.

type of read_data	data_count	actual number of 16-bit register read
char (8-bit)	1	1
char (8-bit)	2	1

1 WORD register(16-bit) equals to the size of 2 ASCII characters. According to the above table, reading 2 ASCII characters is actually reading the content of one 16-bit register.

Example

```
macro_command main()
char str1[20]

//   read 10 words from LW0~LW9 to the variables str1[0] to str1[19]
//   since that 1 word can store 2 ASCII characters, reading 20 ASCII
//   characters is actually reading 10 words of register
StringGet(str1[0], “Local HMI”, LW, 0, 20)
```

	end macro_command

Name	StringGetEx
Syntax	StringGetEx (read_data[start], device_name, device_type, address_offset, data_count)
Description	Receives data from the PLC and continue executing next command even if no response from this device. Descriptions of read_data, device_name, device_type, address_offset and data_count are the same as GetData.

Example	macro_command main()
	char str1[20]
	short test=0
	
	// macro will continue executing test = 1 even if the MODBUS device is
	// not responding
	StringGetEx(str1[0], "MODBUS RTU", 4x, 0, 20)
	test = 1
	
	// macro will not continue executing test = 2 until MODBUS device responds
	StringGet(str1[0], "MODBUS RTU", 4x, 0, 20)
	test = 2
	
	end macro_command

Name	StringSet
Syntax	StringSet(send_data[start], device_name, device_type, address_offset, data_count)
Description	Send data to the PLC. Data is defined in send_data[start]~ send_data[start + data_count - 1]. send_data must be a one-dimensional char array. data_count is the number of sent characters, it can be either a constant or a variable. device_name is the PLC name enclosed in the double quotation marks (“) and this name has been defined in the device list of system parameters. device_type is the device type and encoding method (binary or BCD) of the PLC data. For example, if device_type is LW_BIN, it means the register is LW and the encoding method is binary. If use BIN encoding method, “_BIN” can be ignored.

If device_type is LW_BCD, it means the register is LW and the encoding method is BCD.

address_offset is the address offset in the PLC.
For example, StringSet(read_data_1[0], “FATEK KB Series”, RT, 5, 1) represents that the address offset is 5.

If address_offset uses the format – “N#AAAAA”, N indicates that PLC’s station number is N. AAAAA represents the address offset. This format is used while multiple PLCs or controllers are connected to a single serial port. For example, StringSet(read_data_1[0], “FATEK KB Series”, RT, 2#5, 1) represents that the PLC’s station number is 2. If SetData () uses the default station number defined in the device list, it is not necessary to define station number in address_offset.

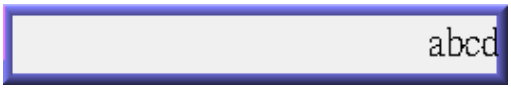
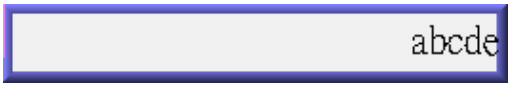
The number of registers actually sends to depends on the value of the number of data_count, since that send_data is restricted to char array.

type of read_data	data_count	actual number of 16-bit register send
char (8-bit)	1	1
char (8-bit)	2	1

1 WORD register(16-bit) equals to the size of 2 ASCII characters. According to the above table, sending 2 ASCII characters is actually writing to one 16-bit register. The ASCII characters are stored into the WORD register from low byte to high byte. While using the ASCII display object to display the string data stored in the registers, data_count must be a multiple of 2 in order to display full string content. For example:

```
macro_command main()
char src1[10]="abcde"
StringSet(src1[0], "Local HMI", LW, 0, 5)
end macro_command
```

The ASCII display object shows:

	 If data_count is an even number that is greater than or equal to the length of the string, the content of string can be completely shown: macro_command main() char src1[10]="abcde" StringSet(src1[0], "Local HMI", LW, 0, 6) end macro_command 

Example	macro_command main()
	
	char str1[10]=”abcde”
	
	// Send 3 words to LW0~LW2
	// Data are being sent until the end of string is reached.
	// Even though the value of data_count is larger than the length of string
	// , the function will automatically stop.
	StringSet(str1[0], "Local HMI", LW, 0, 10)
	
	end macro_command

Name	StringSetEx
Syntax	StringSetEx (send_data[start], device_name, device_type, address_offset, data_count)
Description	Send data to the PLC and continue executing next command even if no response from this device. Descriptions of send_data, device_name, device_type, address_offset and data_count are the same as StringSet.

Example	macro_command main()
	char str1[20]=”abcde”
	short test=0
	
	// macro will continue executing test = 1 even if the MODBUS device is
	// not responding
	StringSetEx(str1[0], "MODBUS RTU", 4x, 0, 20)
	test = 1

	// macro will not continue executing test = 2 until MODBUS device responds
	StringSet(str1[0], "MODBUS RTU", 4x, 0, 20)
	test = 2
	
	end macro_command

Name	StringCopy
Syntax	success = StringCopy (“source”, destination[start]) or success = StringCopy (source[start], destination[start])
Description	Copy one string to another. This function copies a static string (which is enclosed in quotes) or a string that is stored in an array to the destination buffer. The source string parameter accepts both static string (in the form: “source”) and char array (in the form: source[start]). destination[start] must be an one-dimensional char array. This function returns a Boolean indicating whether the process is successfully done or not. If successful, it returns true, otherwise it returns false. If the length of source string exceeds the max. size of destination buffer, it returns false and the content of destination remains the same. The success field is optional.

Example	macro_command main()
	char src1[5]="abcde"
	char dest1[5]
	bool success1
	success1 = StringCopy(src1[0], dest1[0])
	// success1=true, dest1=”abcde”
	
	char dest2[5]
	bool success2
	success2 = StringCopy("12345", dest2[0])
	// success2=true, dest2=”12345”
	
	char src3[10]="abcdefghij"
	char dest3[5]
	bool success3
	success3 = StringCopy(src3[0], dest3[0])
	// success3=false, dest3 remains the same.

	char src4[10]="abcdefghij" char dest4[5] bool success4 success4 = StringCopy(src4[5], dest4[0]) // success4=true, dest4=”fghij” end macro_command

Name	StringDecAsc2Bin
Syntax	success = StringDecAsc2Bin(source[start], destination) or success = StringDecAsc2Bin(“source”, destination)
Description	This function converts a decimal string to an integer. It converts the decimal string in source parameter into an integer, and stores it in the destination variable. The source string parameter accepts both static string (in the form: “source”) and char array (in the form: source[start]). Destination must be a variable, to store the result of conversion. This function returns a Boolean indicating whether the process is successfully done or not. If successful, it returns true, otherwise it returns false. If the source string contains characters other than ‘0’ to ‘9’, it returns false. The success field is optional.

Example	macro_command main()
	char src1[5]="12345"
	int result1
	bool success1
	success1 = StringDecAsc2Bin(src1[0], result1)
	// success1=true, result1 is 12345
	
	char result2
	bool success2
	success2 = StringDecAsc2Bin("32768", result2)
	// success2=true, but the result exceeds the data range of result2
	
	char src3[2]="4b"
	char result3

	bool success3 success3 = StringDecAsc2Bin (src3[0], result3) // success3=false, because src3 contains characters other than '0' to '9' end macro_command

Name	StringBin2DecAsc
Syntax	success = StringBin2DecAsc (source, destination[start])
Description	This function converts an integer to a decimal string. It converts the integer in source parameter into a decimal string, and stores it in the destination buffer. Source can be either a constant or a variable. Destination must be an one-dimensional char array, to store the result of conversion. This function returns a Boolean indicating whether the process is successfully done or not. If successful, it returns true, otherwise it returns false. If the length of decimal string after conversion exceeds the size of destination buffer, it returns false. The success field is optional.

Example	macro_command main()
	int src1 = 2147483647
	char dest1[20]
	bool success1
	success1 = StringBin2DecAsc(src1, dest1[0])
	// success1=true, dest1="2147483647"
	
	short src2 = 0x3c
	char dest2[20]
	bool success2
	success2 = StringBin2DecAsc(src2, dest2[0])
	// success2=true, dest2="60"
	
	int src3 = 2147483647
	char dest3[5]
	bool success3
	success3 = StringBin2DecAsc(src3, dest3[0])
	// success3=false, dest3 remains the same.

	end macro_command

Name	StringDecAsc2Float
Syntax	success = StringDecAsc2Float (source[start], destination) or success = StringDecAsc2Float (“source”, destination)
Description	This function converts a decimal string to floats. It converts the decimal string in source parameter into float, and stores it in the destination variable. The source string parameter accepts both static string (in the form: “source”) and char array (in the form: source[start]). Destination must be a variable, to store the result of conversion. This function returns a Boolean indicating whether the process is successfully done or not. If successful, it returns true, otherwise it returns false. If the source string contains characters other than ‘0’ to ‘9’ or ‘.’, it returns false. The success field is optional.

Example	macro_command main()
	char src1[10]="12.345"
	float result1
	bool success1
	success1 = StringDecAsc2Float(src1[0], result1)
	// success1=true, result1 is 12.345
	
	float result2
	bool success2
	success2 = StringDecAsc2Float("1.234567890", result2)
	// success2=true, but the result exceeds the data range of result2, which
	// might result in loss of precision
	
	char src3[2]="4b"
	float result3
	bool success3
	success3 = StringDecAsc2Float(src3[0], result3)
	// success3=false, because src3 contains characters other than ‘0’ to ‘9’ or
	// ‘.’
	end macro_command

Name	StringFloat2DecAsc

Syntax	success = StringFloat2DecAsc(source, destination[start])
Description	This function converts a float to a decimal string. It converts the float in source parameter into a decimal string, and stores it in the destination buffer. Source can be either a constant or a variable. Destination must be an one-dimensional char array, to store the result of conversion. This function returns a Boolean indicating whether the process is successfully done or not. If successful, it returns true, otherwise it returns false. If the length of decimal string after conversion exceeds the size of destination buffer, it returns false. The success field is optional.

Example	macro_command main()
	float src1 = 1.2345
	char dest1[20]
	bool success1
	success1 = StringFloat2DecAsc(src1, dest1[0])
	// success1=true, dest1=" 1.2345"
	
	float src2 = 1.23456789
	char dest2 [20]
	bool success2
	success2 = StringFloat2DecAsc(src2, dest2 [0])
	// success2=true, but it might lose precision
	
	float src3 = 1.2345
	char dest3[5]
	bool success3
	success3 = StringFloat2DecAsc(src3, dest3 [0])
	// success3=false, dest3 remains the same.
	
	end macro_command

Name	StringHexAsc2Bin
Syntax	success = StringHexAsc2Bin (source[start], destination) or success = StringHexAsc2Bin ("source", destination)
Description	This function converts a hexadecimal string to binary data. It converts the

	hexadecimal string in source parameter into binary data , and stores it in the destination variable. The source string parameter accepts both static string (in the form: “source”) and char array (in the form: source[start]). Destination must be a variable, to store the result of conversion. This function returns a Boolean indicating whether the process is successfully done or not. If successful, it returns true, otherwise it returns false. If the source string contains characters other than ‘0’ to ‘9’, ‘a’ to ‘f’ or ‘A’ to ‘F’, it returns false. The success field is optional.

Example	macro_command main()
	char src1[5]="0x3c"
	int result1
	bool success1
	success1 = StringHexAsc2Bin(src1[0], result1)
	// success1=true, result1 is 3c
	
	short result2
	bool success2
	success2 = StringDecAsc2Bin("1a2b3c4d", result2)
	// success2=true, result2=3c4d.The result exceeds the data range of
	// result2
	
	char src3[2]="4g"
	char result3
	bool success3
	success3 = StringDecAsc2Bin (src3[0], result3)
	// success3=false, because src3 contains characters other than ‘0’ to ‘9’
	// , ‘a’ to ‘f’ or ‘A’ to ‘F’
	
	end macro_command

Name	StringBin2HexAsc
Syntax	success = StringBin2HexAsc (source, destination[start])
Description	This function converts binary data to a hexadecimal string. It converts the binary data in source parameter into a hexadecimal string, and stores it in the destination buffer. Source can be either a constant or a variable.

	Destination must be an one-dimensional char array, to store the result of conversion. This function returns a Boolean indicating whether the process is successfully done or not. If successful, it returns true, otherwise it returns false. If the length of hexadecimal string after conversion exceeds the size of destination buffer, it returns false. The success field is optional.

Example	macro_command main()
	int src1 = 20
	char dest1[20]
	bool success1
	success1 = StringBin2HexAsc(src1, dest1[0])
	// success1=true, dest1=”14”
	
	short src2 = 0x3c
	char dest2[20]
	bool success2
	success2 = StringBin2HexAsc(src2, dest2[0])
	// success2=true, dest2=”3c”
	
	int src3 = 0x1a2b3c4d
	char dest3[6]
	bool success3
	success3 = StringBin2HexAsc(src3, dest3[0])
	// success3=false, dest3 remains the same.
	
	end macro_command

Name	StringMid
Syntax	success = StringMid (source[start], count, destination[start]) or success = StringMid (“string“, start, count, destination[start])
Description	Retrieve a character sequence from the specified offset of the source string and store it in the destination buffer. The source string parameter accepts both static string (in the form: “source”) and char array (in the form: source[start]). For source[start], the start offset of the substring is specified by the index value. For static source string(“source”), the second parameter(start) specifies the start offset of the

	substring. The count parameter specifies the length of substring being retrieved. Destination must be an one-dimensional char array, to store the retrieved substring. This function returns a Boolean indicating whether the process is successfully done or not. If successful, it returns true, otherwise it returns false. If the length of retrieved substring exceeds the size of destination buffer, it returns false. The success field is optional.

Example	macro_command main()
	char src1[20]="abcdefghijklmnopqrst"
	char dest1[20]
	bool success1
	success1 = StringMid(src1[5], 6, dest1[0])
	// success1=true, dest1=”fghijk”
	
	char src2[20]="abcdefghijklmnopqrst"
	char dest2[5]
	bool success2
	success2 = StringMid(src2[5], 6, dest2[0])
	// success2=false, dest2 remains the same.
	
	char dest3[20]="12345678901234567890"
	bool success3
	success3 = StringMid("abcdefghijklmnopqrst", 5, 5, dest3[15])
	// success3= true, dest3=” 123456789012345fghij”
	
	end macro_command

Name	StringLength
Syntax	length = StringLength (source[start]) or length = StringLength (“source“)
Description	Obtain the length of a string. It returns the length of source string and stores it in the length field on the left-hand side of ‘=’ operator. The source string parameter accepts both static string (in the form: “source”) and char array (in the form: source[start]). The return value of this function indicates the length of the source string.

Example	macro_command main()
	char src1[20]="abcde"
	int length1
	length1= StringLength(src1[0])
	// length1=5
	
	char src2[20]={'a', 'b', 'c', 'd', 'e'}
	int length2
	length2= StringLength(src2[0])
	// length2=20
	
	char src3[20]="abcdefghij"
	int length3
	length3= StringLength(src3 [2])
	// length3=8
	
	end macro_command

Name	StringCat
Syntax	success = StringCat (source[start], destination[start]) or success = StringCat (“source”, destination[start])
Description	This function appends source string to destination string. It adds the contents of source string to the last of the contents of destination string. The source string parameter accepts both static string (in the form: “source”) and char array (in the form: source[start]). Destination must be an one-dimensional char array. This function returns a Boolean indicating whether the process is successfully done or not. If successful, it returns true, otherwise it returns false. If the length of result string after concatenation exceeds the max. size of destination buffer, it returns false. The success field is optional.
Example	macro_command main() char src1[20]="abcdefghij" char dest1[20]="1234567890" bool success1 success1= StringCat(src1[0], dest1[0]) // success1=true, dest1=”123456790abcdefghij”

	char dest2 [10]="1234567890"
	bool success2
	success2= StringCat("abcde", dest2 [0])
	// success2=false, dest2 remains the same.
	
	char src3[20]="abcdefghij"
	char dest3[20]
	bool success3
	success3= StringCat(src3[0], dest3[15])
	// success3=false, dest3 remains the same.
	
	end macro_command

Name	StringCompare
Syntax	ret = StringCompare (str1[start], str2[start]) ret = StringCompare (“string1“, str2[start]) ret = StringCompare (str1[start], “string2”) ret = StringCompare (“string1”, “string2”)
Description	Do a case-sensitive comparison of two strings. The two string parameters accept both static string (in the form: “string1”) and char array (in the form: str1[start]). This function returns a Boolean indicating the result of comparison. If two strings are identical, it returns true. Otherwise it returns false. The ret field is optional.

Example	macro_command main()
	char a1[20]="abcde"
	char b1[20]="ABCDE"
	bool ret1
	ret1= StringCompare(a1[0], b1[0])
	// ret1=false
	
	char a2[20]="abcde"
	char b2[20]="abcde"
	bool ret2
	ret2= StringCompare(a2[0], b2[0])
	// ret2=true

	char a3 [20]="abcde" char b3[20]="abcdefg" bool ret3 ret3= StringCompare(a3[0], b3[0]) // ret3=false end macro_command

Name	StringCompareNoCase
Syntax	ret = StringCompareNoCase(str1[start], str2[start]) ret = StringCompareNoCase(“string1“, str2[start]) ret = StringCompareNoCase(str1[start], “string2”) ret = StringCompareNoCase(“string1”, “string2”)
Description	Do a case-insensitive comparison of two strings. The two string parameters accept both static string (in the form: “string1”) and char array (in the form: str1[start]). This function returns a Boolean indicating the result of comparison. If two strings are identical, it returns true. Otherwise it returns false. The ret field is optional.

Example	macro_command main()
	char a1[20]="abcde"
	char b1[20]="ABCDE"
	bool ret1
	ret1= StringCompareNoCase(a1[0], b1[0])
	// ret1=true
	
	char a2[20]="abcde"
	char b2[20]="abcde"
	bool ret2
	ret2= StringCompareNoCase(a2[0], b2[0])
	// ret2=true
	
	char a3 [20]="abcde"
	char b3[20]="abcdefg"
	bool ret3
	ret3= StringCompareNoCase(a3[0], b3[0])
	// ret3=false

	end macro_command

Name	StringFind
Syntax	position = StringFind (source[start], target[start]) position = StringFind (“source“, target[start]) position = StringFind (source[start], “target”) position = StringFind (“source“, “target“)
Description	Return the position of the first occurrence of target string in the source string. The two string parameters accept both static string (in the form: “source”) and char array (in the form: source[start]). This function returns the zero-based index of the first character of substring in the source string that matches the target string. Notice that the entire sequence of characters to find must be matched. If there is no matched substring, it returns -1.

Example	macro_command main()
	char src1[20]="abcde"
	char target1[20]="cd"
	bool pos1
	pos1= StringFind(src1[0], target1[0])
	// pos1=2
	
	char target2[20]="ce"
	bool pos2
	pos2= StringFind("abcde", target2[0])
	// pos2=-1
	
	char src3[20]="abcde"
	bool pos3
	pos3= StringFind(src3[3], "cd")
	// pos3=-1
	
	end macro_command

Name	StringReverseFind
Syntax	position = StringReverseFind (source[start], target[start]) position = StringReverseFind (“source“, target[start]) position = StringReverseFind (source[start], “target”)

	position = StringReverseFind (“source“, “target“)
Description	Return the position of the last occurrence of target string in the source string. The two string parameters accept both static string (in the form: “source”) and char array (in the form: source[start]). This function returns the zero-based index of the first character of substring in the source string that matches the target string. Notice that the entire sequence of characters to find must be matched. If there exists multiple substrings that matches the target string, function will return the position of the last matched substring. If there is no matched substring, it returns -1.

Example	macro_command main()
	char src1[20]="abcdeabcde"
	char target1[20]="cd"
	bool pos1
	pos1= StringReverseFind(src1[0], target1[0])
	// pos1=7
	
	char target2[20]="ce"
	bool pos2
	pos2= StringReverseFind("abcdeabcde", target2[0])
	// pos2=-1
	
	char src3[20]="abcdeabcde"
	bool pos3
	pos3= StringReverseFind(src3[6], "ab")
	// pos3=-1
	
	end macro_command

Name	StringFindOneOf
Syntax	position = StringFindOneOf (source[start], target[start]) position = StringFindOneOf (“source“, target[start]) position = StringFindOneOf (source[start], “target”) position = StringFindOneOf (“source“, “target“)
Description	Return the position of the first character in the source string that matches any character contained in the target string. The two string parameters accept both static string (in the form: “source”) and char array (in the form: source[start]).

	This function returns the zero-based index of the first character in the source string that is also in the target string. If there is no match, it returns -1.

Example	macro_command main()
	char src1[20]="abcdeabcde"
	char target1[20]="sdf"
	bool pos1
	pos1= StringFindOneOf(src1[0], target1[0])
	// pos1=3
	
	char src2[20]="abcdeabcde"
	bool pos2
	pos2= StringFindOneOf(src2[1], "agi")
	// pos2=4
	
	char target3 [20]="bus"
	bool pos3
	pos3= StringFindOneOf("abcdeabcde", target3[1])
	// pos3=-1
	
	end macro_command

Name	StringIncluding
Syntax	success = StringIncluding (source[start], set[start], destination[start]) success = StringIncluding (“source“, set[start], destination[start]) success = StringIncluding (source[start], “set“, destination[start]) success = StringIncluding (“source“, “set“, destination[start])
Description	Retrieve a substring of the source string that contains characters in the set string, beginning with the first character in the source string and ending when a character is found in the source string that is not in the target string. The source string and set string parameters accept both static string (in the form: “source”) and char array (in the form: source[start]). This function returns a Boolean indicating whether the process is successfully done or not. If successful, it returns true, otherwise it returns false. If the length of retrieved substring exceeds the size of destination buffer, it returns false.
Example	macro_command main() char src1[20]="cabbageabc"

	char set1[20]="abc"
	char dest1[20]
	bool success1
	success1 = StringIncluding(src1[0], set1[0], dest1[0])
	// success1=true, dest1="cabba"
	
	char src2[20]="gecabba"
	char dest2[20]
	bool success2
	success2 = StringIncluding(src2[0], "abc", dest2[0])
	// success2=true, dest2=""
	
	char set3[20]="abc"
	char dest3[4]
	bool success3
	success3 = StringIncluding("cabbage", set3[0], dest3[0])
	// success3=false, dest3 remains the same.
	
	end macro_command

Name	StringExcluding
Syntax	success = StringExcluding (source[start], set[start], destination[start]) success = StringExcluding (“source“, set[start], destination[start]) success = StringExcluding (source[start], “set“, destination[start]) success = StringExcluding (“source“, “set“, destination[start])
Description	Retrieve a substring of the source string that contains characters that are not in the set string, beginning with the first character in the source string and ending when a character is found in the source string that is also in the target string. The source string and set string parameters accept both static string (in the form: “source”) and char array (in the form: source[start]). This function returns a Boolean indicating whether the process is successfully done or not. If successful, it returns true, otherwise it returns false. If the length of retrieved substring exceeds the size of destination buffer, it returns false.
Example	macro_command main() char src1[20]="cabbageabc" char set1[20]="ge"

	char dest1[20]
	bool success1
	success1 = StringExcluding(src1[0], set1[0], dest1[0])
	// success1=true, dest1="cabba"
	
	char src2[20]="cabbage"
	char dest2[20]
	bool success2
	success2 = StringExcluding(src2[0], "abc", dest2[0])
	// success2=true, dest2=""
	
	char set3[20]="ge"
	char dest3[4]
	bool success3
	success3 = StringExcluding("cabbage", set3[0], dest3[0])
	// success3=false, dest3 remains the same.
	
	end macro_command

Name	StringToUpper
Syntax	success = StringToUpper (source[start], destination[start]) success = StringToUpper ("source", destination[start])
Description	Convert all the characters in the source string to uppercase characters and store the result in the destination buffer. The source string parameter accepts both static string (in the form: "source") and char array (in the form: source[start]). This function returns a Boolean indicating whether the process is successfully done or not. If successful, it returns true, otherwise it returns false. If the length of result string after conversion exceeds the size of destination buffer, it returns false.
Example	macro_command main() char src1[20]="aBcDe" char dest1[20] bool success1 success1 = StringToUpper(src1[0], dest1[0]) // success1=true, dest1="ABCDE" char dest2[4]

	bool success2 success2 = StringToUpper("aBcDe", dest2[0]) // success2=false, dest2 remains the same. end macro_command

Name	StringToLower
Syntax	success = StringToLower (source[start], destination[start]) success = StringToLower ("source", destination[start])
Description	Convert all the characters in the source string to lowercase characters and store the result in the destination buffer. The source string parameter accepts both static string (in the form: "source") and char array (in the form: source[start]). This function returns a Boolean indicating whether the process is successfully done or not. If successful, it returns true, otherwise it returns false. If the length of result string after conversion exceeds the size of destination buffer, it returns false.

Example	macro_command main()
	char src1[20]="aBcDe"
	char dest1[20]
	bool success1
	success1 = StringToUpper(src1[0], dest1[0])
	// success1=true, dest1="abcde"
	
	char dest2[4]
	bool success2
	success2 = StringToUpper("aBcDe", dest2[0])
	// success2=false, dest2 remains the same.
	
	end macro_command

Name	StringToReverse
Syntax	success = StringToReverse (source[start], destination[start]) success = StringToReverse ("source", destination[start])
Description	Reverse the characters in the source string and store it in the destination buffer. The source string parameter accepts both static string (in the form: "source") and char array (in the form: source[start]).

	This function returns a Boolean indicating whether the process is successfully done or not. If successful, it returns true, otherwise it returns false. If the length of reversed string exceeds the size of destination buffer, it returns false.

Example	macro_command main()
	char src1[20]="abcde"
	char dest1[20]
	bool success1
	success1 = StringToUpper(src1[0], dest1[0])
	// success1=true, dest1="edcba"
	
	char dest2[4]
	bool success2
	success2 = StringToUpper("abcde", dest2[0])
	// success2=false, dest2 remains the same.
	
	end macro_command

Name	StringTrimLeft
Syntax	success = StringTrimLeft (source[start], set[start], destination[start]) success = StringTrimLeft (“source“, set[start], destination[start]) success = StringTrimLeft (source[start], “set“, destination[start]) success = StringTrimLeft (“source“, “set“, destination[start])
Description	Trim the leading specified characters in the set buffer from the source string. The source string and set string parameters accept both static string (in the form: "source") and char array (in the form: source[start]). This function returns a Boolean indicating whether the process is successfully done or not. If successful, it returns true, otherwise it returns false. If the length of trimmed string exceeds the size of destination buffer, it returns false.
Example	macro_command main() char src1[20]= "# *a*#bc" char set1[20]="# *" char dest1[20] bool success1 success1 = StringTrimLeft (src1[0], set1[0], dest1[0]) // success1=true, dest1="a*#bc"

```
char set2[20]={'#', ' ', '*'}
char dest2[4]
success2 = StringTrimLeft ("# *a*#bc", set2[0], dest2[0])
// success2=false, dest2 remains the same.

char src3[20]="abc *#"
char dest3[20]
bool success3
success3 = StringTrimLeft (src3[0], "# *", dest3[0])
// success3=true, dest3="abc *#"

end macro_command
```

Name	StringTrimRight
Syntax	success = StringTrimRight (source[start], set[start], destination[start]) success = StringTrimRight (“source“, set[start], destination[start]) success = StringTrimRight (source[start], “set“, destination[start]) success = StringTrimRight (“source“, “set“, destination[start])
Description	Trim the trailing specified characters in the set buffer from the source string. The source string and set string parameters accept both static string (in the form: "source") and char array (in the form: source[start]). This function returns a Boolean indicating whether the process is successfully done or not. If successful, it returns true, otherwise it returns false. If the length of trimmed string exceeds the size of destination buffer, it returns false.

Example	macro_command main()
	char src1[20]= "# *a*#bc# * "
	char set1[20]="# *"
	char dest1[20]
	bool success1
	success1 = StringTrimRight(src1[0], set1[0], dest1[0])
	// success1=true, dest1="# *a*#bc"
	
	char set2[20]={'#', ' ', '*'}
	char dest2[20]
	success2 = StringTrimRight("# *a*#bc", set2[0], dest2[0])
	// success2=true, dest2="# *a*#bc"

	char src3[20]="ab**c *#" char dest3[4] bool success3 success3 = StringTrimRight(src3[0], "# *", dest3[0]) // success3=false, dest3 remains the same. end macro_command

Name	StringInsert
Syntax	success = StringInsert (pos, insert[start], destination[start]) success = StringInsert (pos, “insert“, destination[start]) success = StringInsert (pos, insert[start], length, destination[start]) success = StringInsert (pos, “insert“, length, destination[start])
Description	Insert a string in a specific location within the destination string content. The insert location is specified by the pos parameter. The insert string parameter accepts both static string (in the form: "source") and char array (in the form: source[start]). The number of characters to insert can be specified by the length parameter. This function returns a Boolean indicating whether the process is successfully done or not. If successful, it returns true, otherwise it returns false. If the length of string after insertion exceeds the size of destination buffer, it returns false.

Example	macro_command main()
	
	char str1[20]="but the question is"
	char str2[10]=", that is"
	char dest[40]="to be or not to be"
	bool success
	
	success = StringInsert(18, str1[3], 13, dest[0])
	// success=true, dest="to be or not to be the question"
	
	success = StringInsert(18, str2[0], dest[0])
	// success=true, dest="to be or not to be, that is the question"
	
	success = StringInsert(0, "Hamlet:", dest[0])

	// success=false, dest remains the same. end macro_command

18.6.7 Miscellaneous

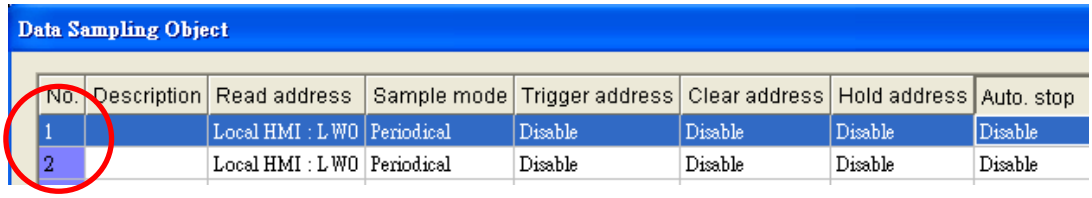
Name	SYNC_TRIG_MACRO
Syntax	SYNC_TRIG_MACRO(*macro_id*)
Description	Trigger the execution of a macro synchronously (use *macro_id* to designate this macro) in a running macro. The current macro will pause until the end of execution of this called macro. *macro_id* can be a constant or a variable.
Example	macro_command main() char ON = 1, OFF = 0 SetData(ON, “Local HMI”, LB, 0, 1) SYNC_TRIG_MACRO(5)// call a macro (its ID is 5) SetData(OFF, “Local HMI”, LB, 0, 1) end macro_command

Name	ASYNC_TRIG_MACRO
Syntax	ASYNC_TRIG_MACRO (*macro_id*)
Description	Trigger the execution of a macro asynchronously (use *macro_id* to designate this macro) in a running macro. The current macro will continue executing the following instructions after triggering the designated macro; in other words, the two macros will be active simultaneously. *macro_id* can be a constant or a variable.
Example	macro_command main() char ON = 1, OFF = 0 SetData(ON, “Local HMI”, LB, 0, 1) ASYNC_TRIG_MACRO(5)// call a macro (its ID is 5) SetData(OFF, “Local HMI”, LB, 0, 1) end macro_command

Name	TRACE
Syntax	TRACE(*format*, *argument*)
Description	Use this function to send specified string to the EasyDiagnoser. Users can print out the current value of variables during run-time of macro for debugging. When TRACE encounters the first format specification (if any), it converts the value of the first argument after format and outputs it accordingly. *format* refers to the format control of output string. A format specification, which consists of optional (in [ ]) and required fields (in bold), has the following form: **%**[flags] [width] [.precision] **type** Each field of the format specification is described as below: *flags* (optional): - + *width* (optional): A nonnegative decimal integer controlling the minimum number of characters printed. *precision* (optional): A nonnegative decimal integer which specifies the precision and the number of characters to be printed. *type*: C or c : specifies a single-byte character. d : signed decimal integer. i : signed decimal integer. o : unsigned octal integer. u : unsigned decimal integer. X or x : unsigned hexadecimal integer. E or e : Signed value having the form. [–]*d***.***dddd* **e** [*sign*]*ddd* where *d* is a single decimal digit, *dddd* is one or more decimal digits, *ddd* is exactly three decimal digits, and *sign* is + or –. f : Signed value having the form [–]*dddd***.***dddd*, where *dddd* is one or more decimal digits. The length of output string is limited to 256 characters. The extra

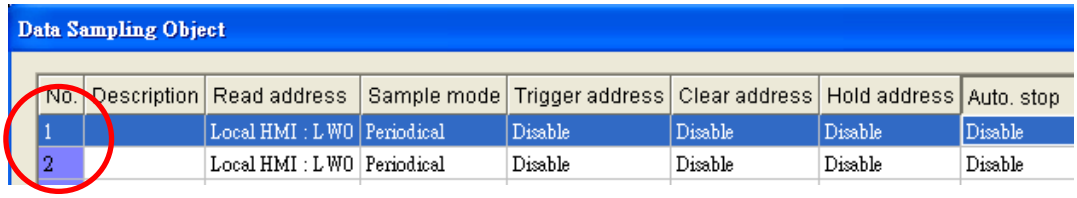
	characters will be ignored. The *argument* part is optional. One format specification converts exactly one argument.

Example	macro_command main()
	char c1 = 'a'
	short s1 = 32767
	float f1 = 1.234567
	
	TRACE("The results are") // output: The results are
	TRACE("c1 = %c, s1 = %d, f1 = %f", c1, s1, f1)
	// output: c1 = a, s1 = 32767, f1 = 1.234567
	
	end macro_command

Name	FindDataSamplingDate
Syntax	return_value = FindDataSamplingDate (*data_log_number*, *index*, *year*, *month*, *day*) or FindDataSamplingDate (*data_log_number*, *index*, *year*, *month*, *day*)
Description	A query function for finding the date of specified data sampling file according to the data sampling no. and the file index. The date is stored into *year*, *month* and *day* respectively in the format of YYYY, MM and DD.  data sampling no. The directory of saved data: [Storage location]\[filename]\yyyymmdd.dtl. The data sampling files under the same directory are sorted according to the file name and are indexed starting from 0. The most recently saved file has the smallest file index number. For example, if there are four data sampling files as follows: 20101210.dtl 20101230.dtl 20110110.dtl 20110111.dtl The file index are:

	20101210.dtl -> index is 3 20101230.dtl -> index is 2 20110110.dtl -> index is 1 20110111.dtl -> index is 0 *return_value* equals to 1 if referred data sampling file is successfully found, otherwise it equals to 0. *data_log_number* and *index*can be constant or variable. *year*, *month*, *day* and *return_value* must be variable. *return_value* is optional.

Example	macro_command main()
	short data_log_number = 1, index = 2, year, month, day
	short success
	
	// if there exists a data sampling file named 20101230.dtl, with data sampling // number 1 and file index 2.
	// the result after execution: success == 1, year == 2010, month == 12 and //day == 30
	success = FindDataSamplingDate(data_log_number, index, year, month, day)
	
	end macro_command

Name	FindDataSamplingIndex
Syntax	return_value = FindDataSamplingIndex (*data_log_number*, *year*, *month*, *day*, *index*) or FindDataSamplingIndex (*data_log_number*, *year*, *month*, *day*, *index*)
Description	A query function for finding the file index of specified data sampling file according to the data sampling no. and the date. The file index is stored into *index*. *year*, *month* and *day* are in the format of YYYY, MM and DD respectively.  data sampling no. The directory of saved data: [Storage location]\[filename]\yyyymmdd.dtl. The data sampling files under the same directory are sorted according to the file name and are indexed starting from 0. The most recently saved file has the smallest file index number. For example, if there are four data

	sampling files as follows: 20101210.dtl 20101230.dtl 20110110.dtl 20110111.dtl The file index are: 20101210.dtl -> index is 3 20101230.dtl -> index is 2 20110110.dtl -> index is 1 20110111.dtl -> index is 0 *return_value* equals to 1 if referred data sampling file is successfully found, otherwise it equals to 0. *data_log_number*, *year*, *month* and *day* can be constant or variable. *index* and *return_value* must be variable. *return_value* is optional.

Example	macro_command main()
	short data_log_number = 1, year = 2010, month = 12, day = 10, index
	short success
	
	// if there exists a data sampling file named 20101210.dtl, with data sampling // number 1 and file index 2.
	// the result after execution: success == 1 and index == 2
	success = FindDataSamplingIndex (data_log_number, year, month, day, index)
	
	end macro_command

Name	FindEventLogDate
Syntax	return_value = FindEventLogDate (*index*, *year*, *month*, *day*) or FindEventLogDate (*index*, *year*, *month*, *day*)
Description	A query function for finding the date of specified event log file according to file index. The date is stored into *year*, month and *day* respectively in the format of YYYY, MM and DD. The event log files stored in the designated position (such as HMI memory storage or external memory device) are sorted according to the file name and are indexed starting from 0. The most recently saved file has the smallest file index number. For example, if there are four event log files as follows: EL_20101210.evt

	EL_20101230.evt EL_20110110.evt EL_20110111.evt The file index are: EL_20101210.evt -> index is 3 EL_20101230.evt -> index is 2 EL_20110110.evt -> index is 1 EL_20110111.evt -> index is 0 *return_value* equals to 1 if referred data sampling file is successfully found, otherwise it equals to 0. *index* can be constant or variable. *year*, *month*, *day* and *return_value* must be variable. *return_value* is optional.

Example	macro_command main()
	short index = 1, year, month, day
	short success
	
	// if there exists an event log file named EL_20101230.evt，with index 1
	// the result after execution: success == 1, year == 2010, month == 12, day //== 30
	success = FindEventLogDate (index, year, month, day)
	
	end macro_command

Name	FindEventLogIndex
Syntax	return_value = FindEventLogIndex (*year*, *month*, *day*, *index*) or FindEventLogIndex (*year*, *month*, *day*, *index*)
Description	A query function for finding the file index of specified event log file according to date. The file index is stored into *index*. *year*, *month* and *day* are in the format of YYYY, MM and DD respectively. The event log files stored in the designated position (such as HMI memory storage or external memory device) are sorted according to the file name and are indexed starting from 0. The most recently saved file has the smallest file index number. For example, if there are four event log files as follows: EL_20101210.evt EL_20101230.evt EL_20110110.evt EL_20110111.evt

	The file index are: EL_20101210.evt -> index is 3 EL_20101230.evt -> index is 2 EL_20110110.evt -> index is 1 EL_20110111.evt -> index is 0 *return_value* equals to 1 if referred data sampling file is successfully found, otherwise it equals to 0. *index* can be constant or variable. *year*, *month*, *day* and *return_value* must be variable. *return_value* is optional.

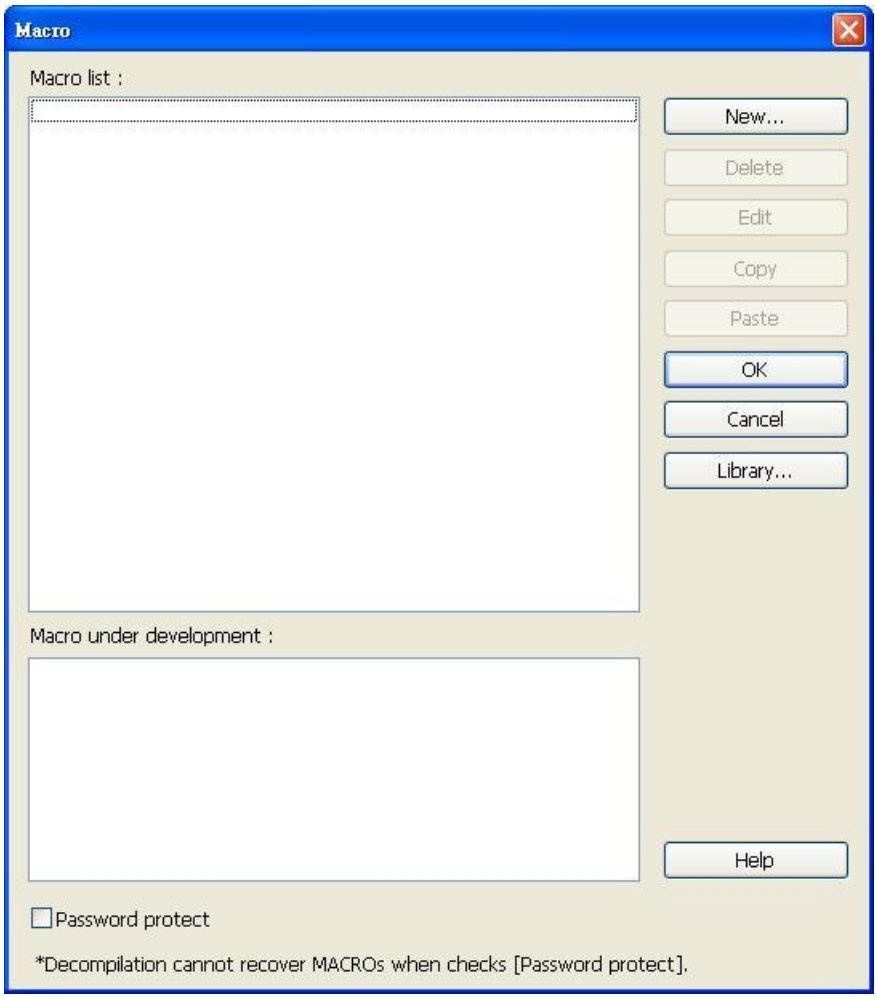
Example	macro_command main()
	short year = 2010, month = 12, day = 10, index
	short success
	
	// if there exists an event log file named EL_20101210.evt, with index 2
	// the result after execution: success == 1, index == 2
	success = FindEventLogIndex (year, month, day, index)
	
	end macro_command

18.7 How to Create and Execute a Macro

18.7.1 How to Create a Macro

Please follow the steps below to create a macro:.

Step 1:
Click on [Macro Manager] icon on the tool bar in EasyBuilder to open Macro Manager dialogue box as follows.



In Macro Manager, all macros compiled successfully are displayed in “Macro list”, and all macros which is under development or cannot be compiled are displayed in “Macro under development”. The following is a description of the various buttons.

[New]

Opens a blank “WorkSpace” editor for creating a new macro.

[Delete]

Deletes the selected macro.

[Edit]

Opens the “WorkSpace” editor, and loads the selected macro.

[Copy]

Copies the selected macro into the clipboard.

[Paste]

Pastes the macro in the clipboard into the list, and creates a new name for the macro.

[OK]

Confrim all the edited Macros and click this button to save the new contents before leaving this dialog.

[Cancel]

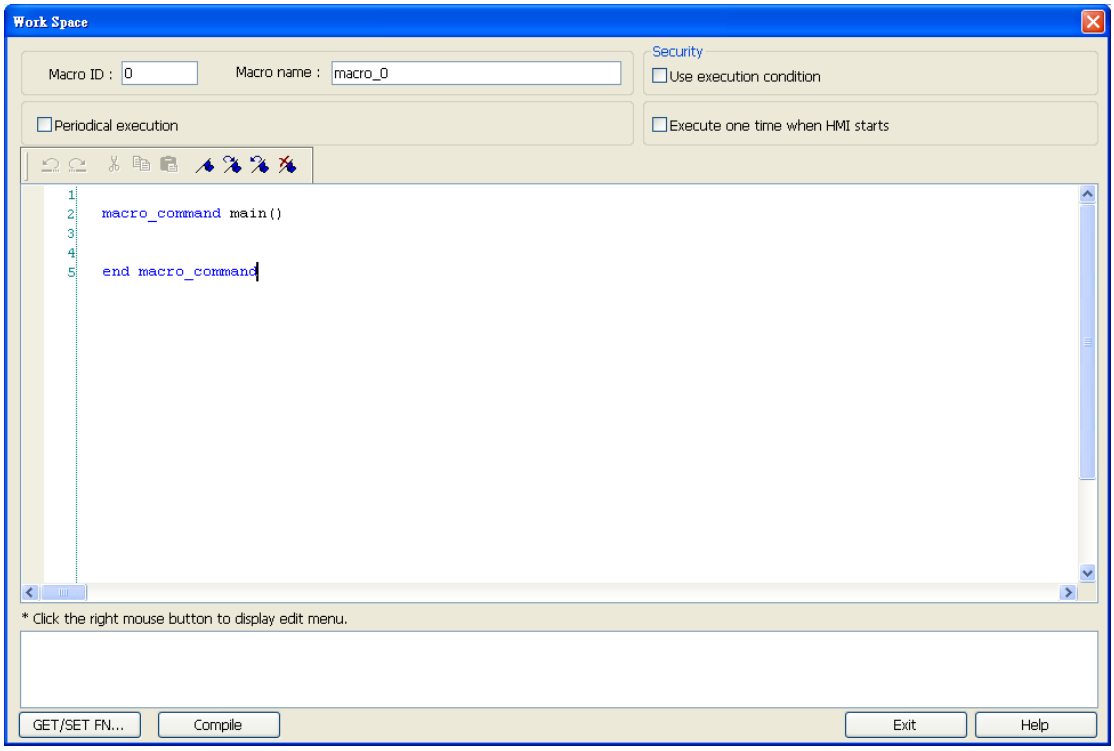
Cancel the editing and leave Macro editing dialog.

[Library…]

Open Macro Funtion Library managing dialog.

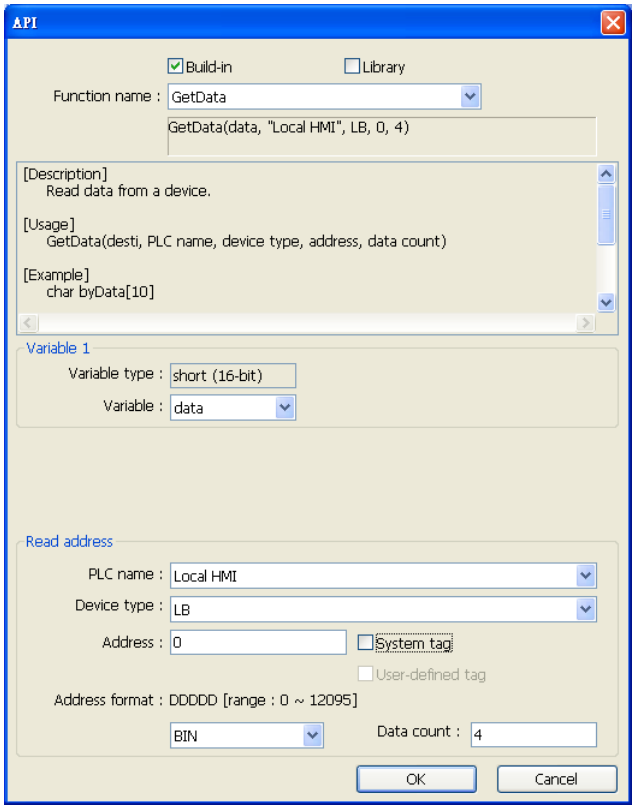
Step 2:

Press the “New” button to create a empty macro and open the macro editor. Every macro has a unique number defined at [Macro ID], and must have a macro name, otherwise an error will appear while compiling.



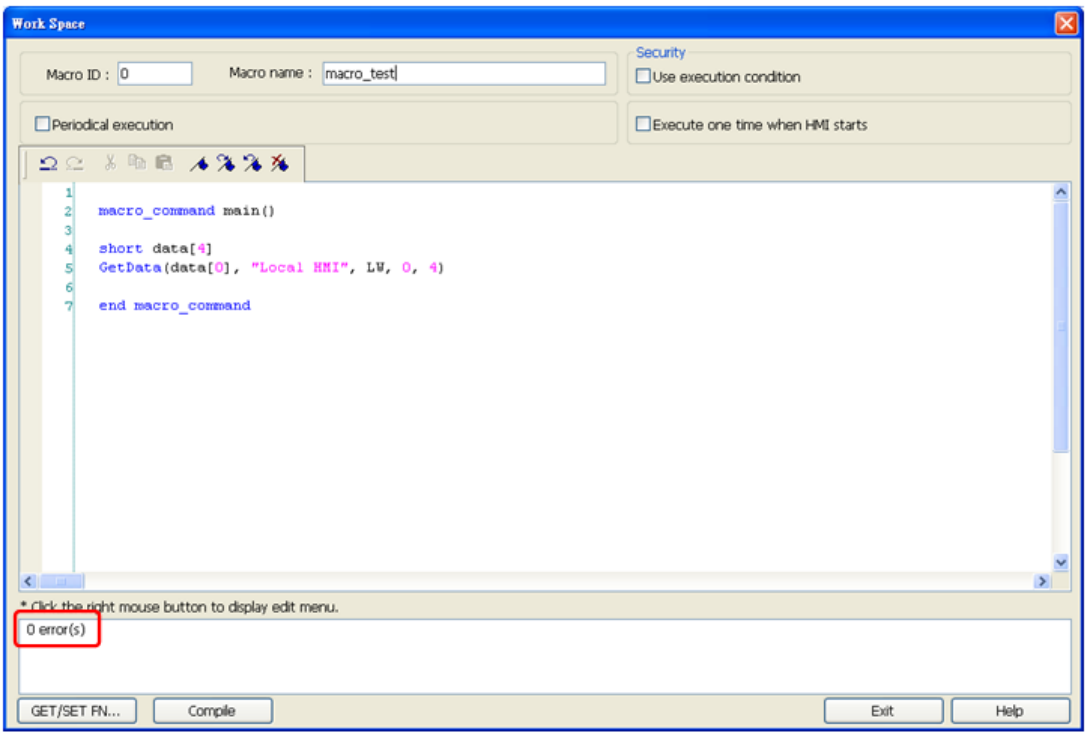
Step 3:

Design your macro. To use built-in functions (like SetData() or Getdata()), press ‘Get/Set FN…” button to open API dialog and select the function and set essential parameters.

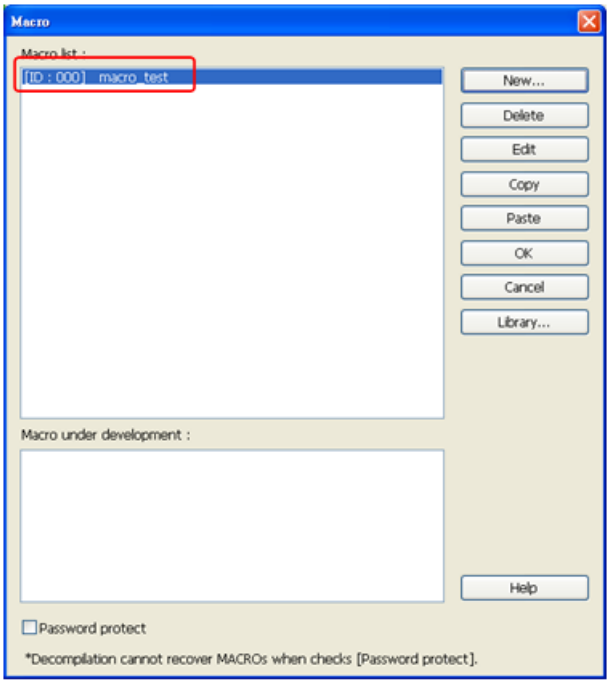


Step 4:

After the completion of a new macro, press ‘Compile” button to compile the macro.



If there is no error, press “Exit” button and a new macro “macro_test” will be in “Macro list”.



18.7.2 Execute a Macro

There are several ways to execute a macro.

a. Use a [PLC Control] object
   Execute the macro when the [Trigger address] changes as defined.
   1. Open [PLC Control] and add one [PLC Control] object with the [Type of control] as [Execute macro program].
   2. Select the macro in [Macro name]. Choose a bit and select a trigger condition to trigger the macro. In order to guarantee that the macro will run only once, consider latching the trigger bit, and then resetting the trigger condition within the macro.
   3. Use a [Set Bit] or [Toggle Switch] object to change the bit to activate the macro.

b. Use a [Set Bit] or [Toggle Switch] object
   The macro will be executed when [Set Bit] or [Toggle Switch] is pressed by user.
   1. On the [General] tab of the [Set Bit] or [Toggle Switch] dialog, select the [Execute Macro] option.
   2. Select the macro to execute.
   3. The macro will be executed one time when the button is activated.

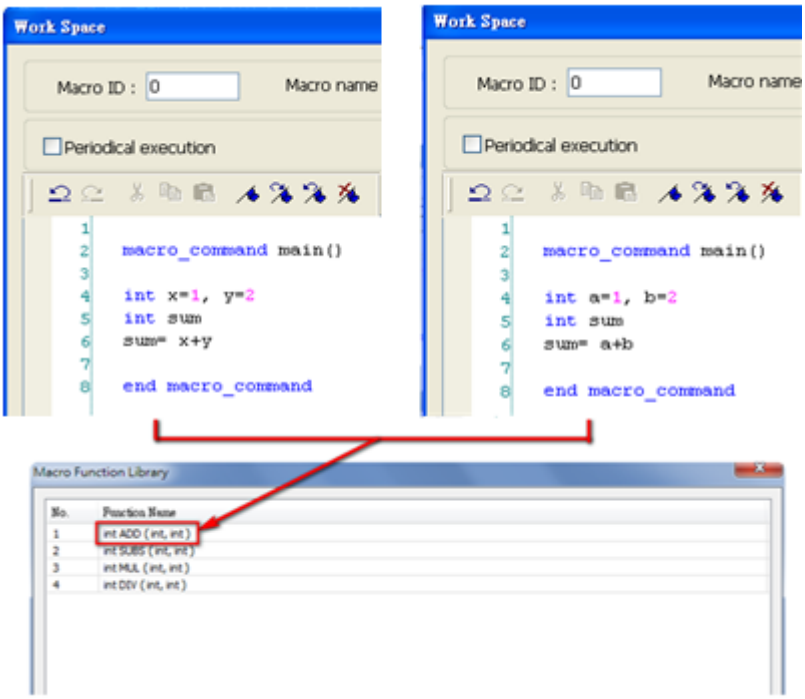
- If [Set Bit] uses [Periodic Toggle], the macro will be executed every time [Set Bit] toggles.

c. Use a [Function Key] object
   1. On the [General] tab of the [Function Key] dialog, select the [Execute Macro] option.
   2. Select the macro to execute.
   3. The macro will execute one time when the button is activated.

d. In macro editor, use
   1. [Periodical Execution]: Macro will be triggered periodically.
   2. [Execute one time when HMI starts]: Macro will be executed once HMI starts.

18.8 User Defined Macro Function

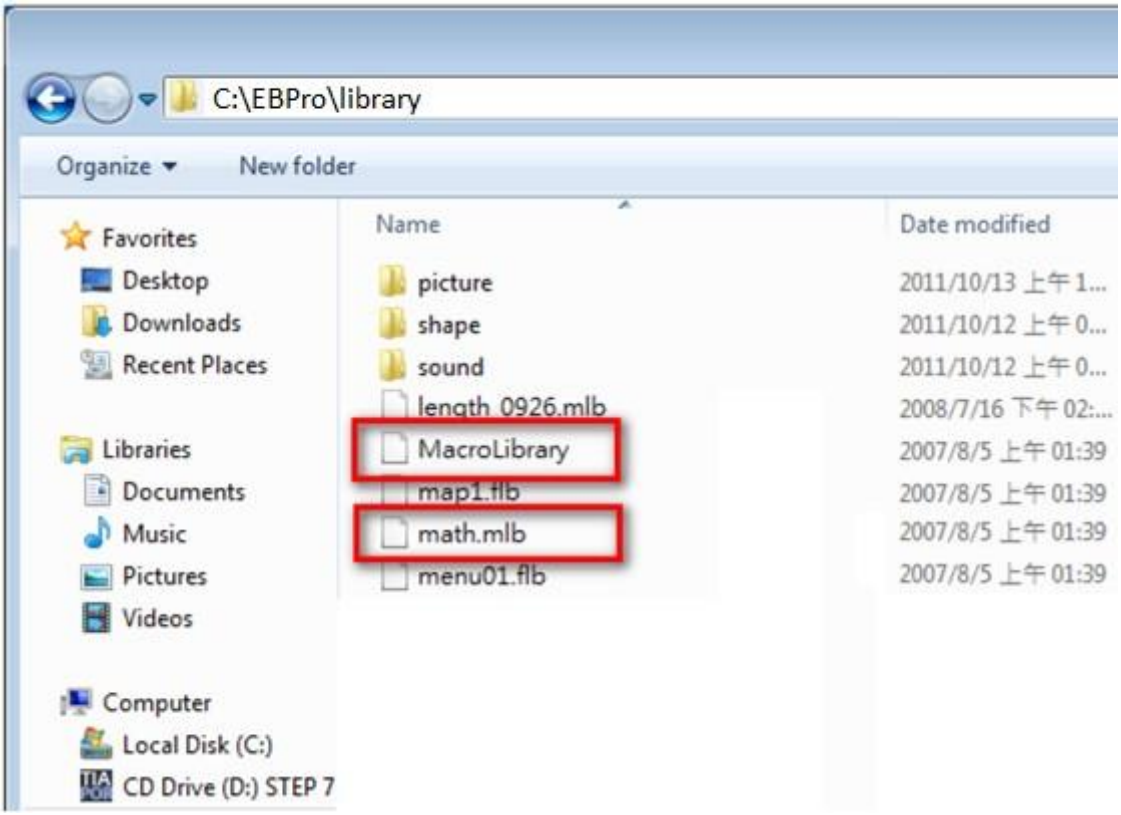
When editing Macro, to save time of defining functions, user may search for the needed from built-in Macro Function Library. However, certain functions, though frequently used, may not be found there. In this case, user may define the needed function and save it for future use. Next time when the same function is required, the saved functions can be called from [Macro Function Library] for easier editing. Additionally, [Macro Function Library] greatly enhances the portability of user-defined functions. Before building a function please check the built-in functions or online function library to see if it exists.



18.8.1 Import Function Library File

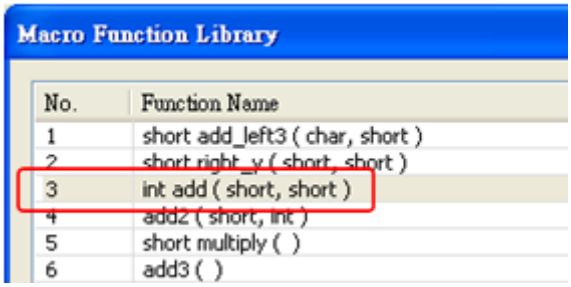
Open a project in HMI programming software, the default Function Library File will be read automatically and the function information will be loaded in. At this moment if a user-defined function is called, the relevant *.mlb file must be imported first.

1. Default Function Library File Name: MacroLibrary (without filename extension)
2. Function Library Directory: HMI programming software installation directory\library (folder)
3. \library (folder) contains two types of function library files:
   - Without filename extension: MacroLibrary, the Default Function Library for HMI programming software to read at the beginning.
   - With filename extension (*.mlb): Such as ”math.mlb”. The files to be read / written when users import / export. These files are portable and can be called from the folder when needed.
4. When opening HMI programming software, only the functions in Default Function Library will be loaded in, to use functions in *.mlb files, please import them first.

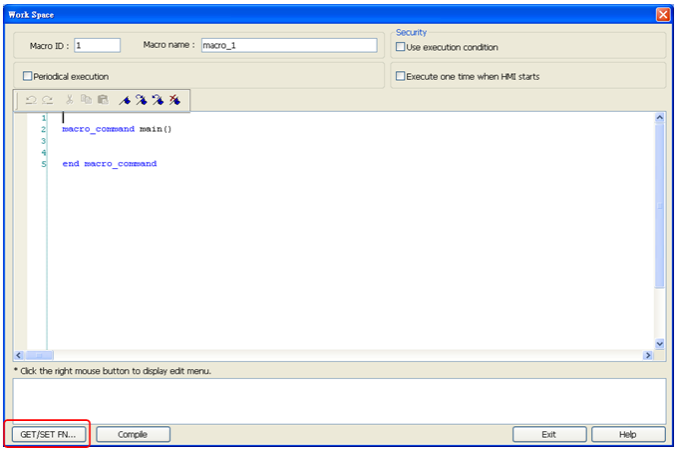


18.8.2 How to Use Macro Function Library

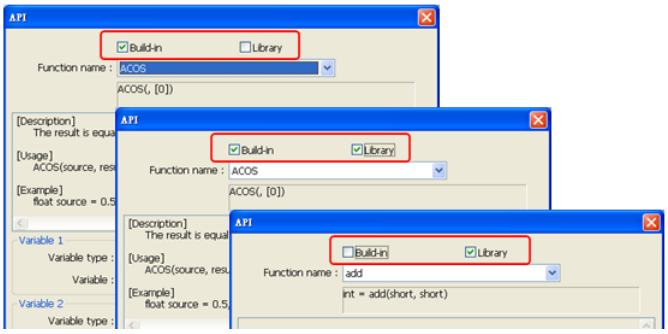
1. Select the function directly from Macro Function Library.



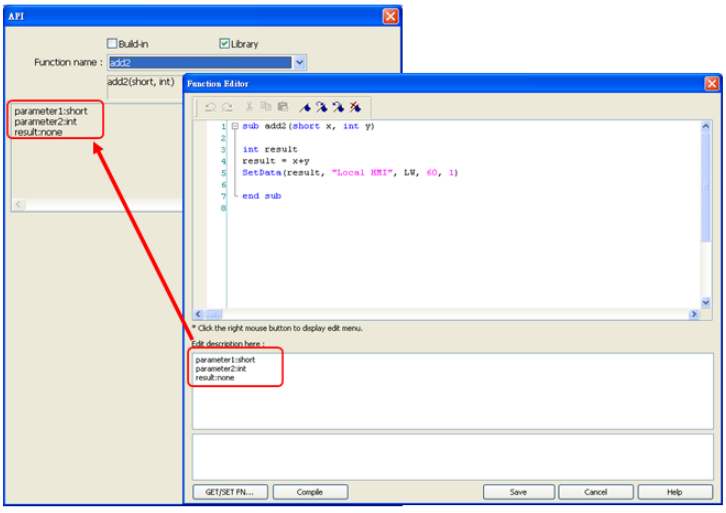
2. In WorkSpace click [GET/SET FN…] to open API dialog box.



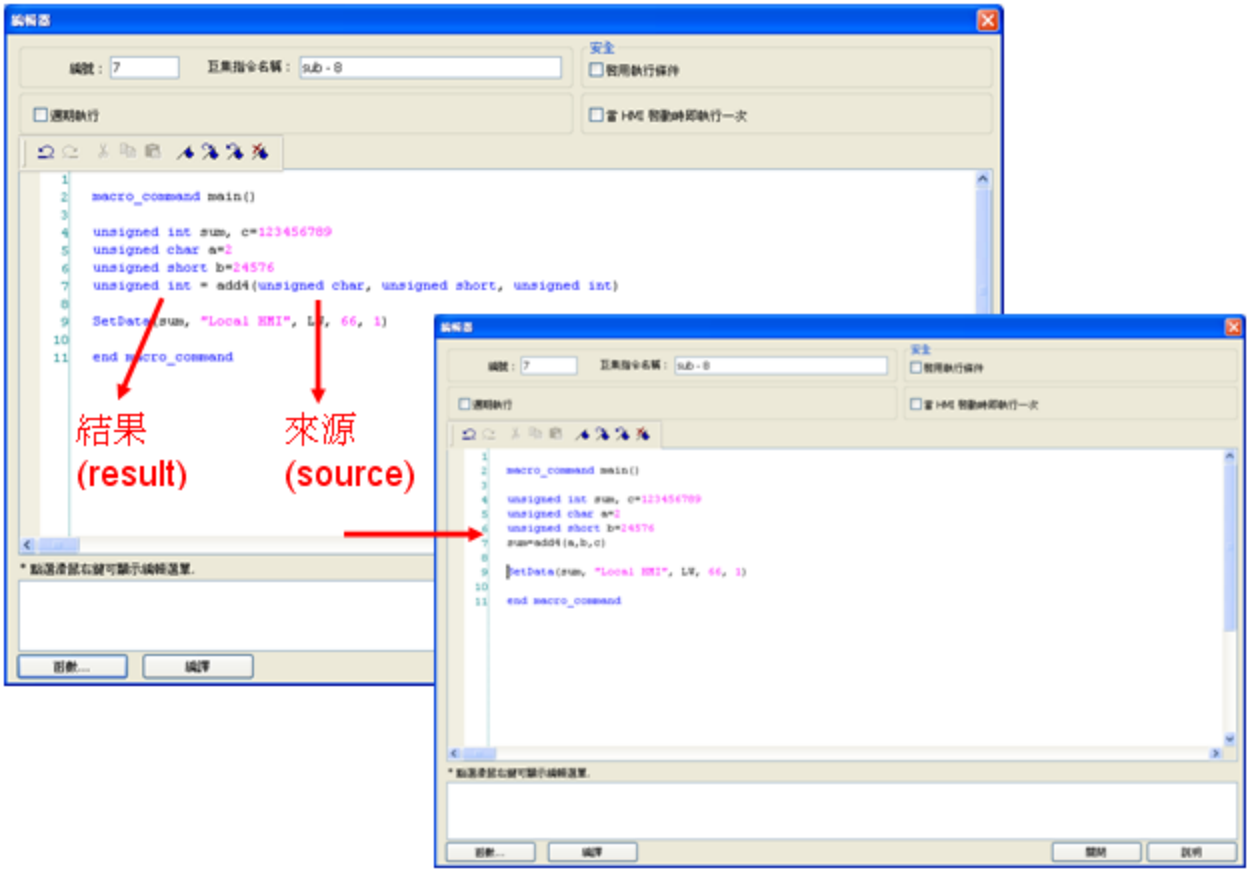
3. At least check one from [Library] or [Build-in] and select the function to be used.



4. The description displayed in API dialog is the same as written in Function Editor.



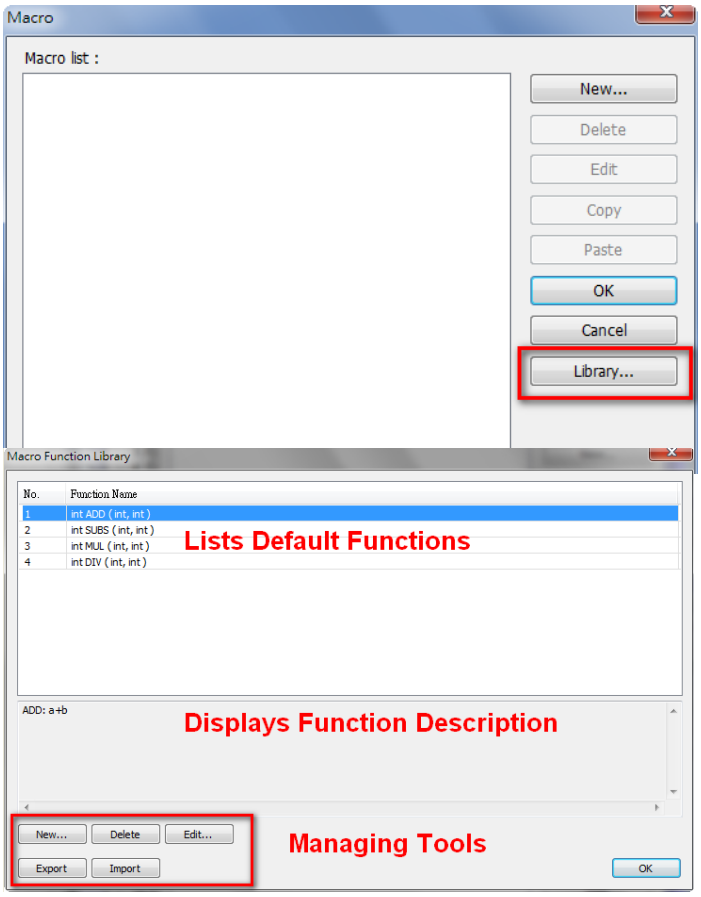
5. Select the function to be used, fill in the corresponding variables according to the data type.



Upon completion of the steps above, user-defined functions can be used freely without defining the same functions repeatedly.

18.8.3 Function Library Management Interface

1 Open macro management dialog, click [Library] to open [Macro Function Library] dialog.



2 A list of functions will be shown. When the project is opened, the software will load all the functions in the Default Function Library.

3. Each listed function has the following format:

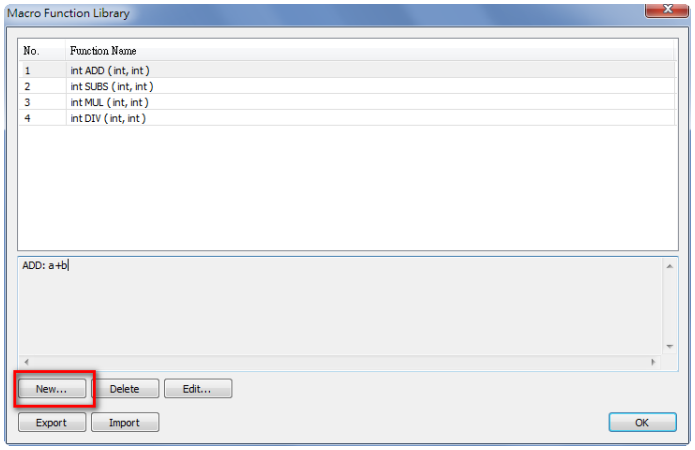
return_type function_name (*parameter_type1, …, parameter_typeN*)

return_type indicates the type of the return value. If this value does not exist, this column will be omitted. function_name indicates the name of the function. “N” in *parameter_typeN* stands for the number of parameter types. If this function does not need any parameter, this column will be omitted.

```
1  sub int ADD(int a, int b)
2    int ret
3    ret = a+b
4    return ret
5  end sub
6
```

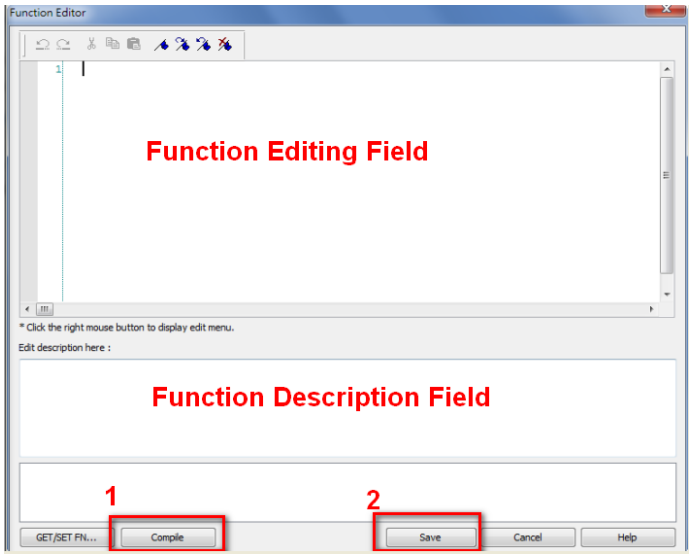
18.8.3.1 Create a Function

1. Click [New] to enter Function Editor.

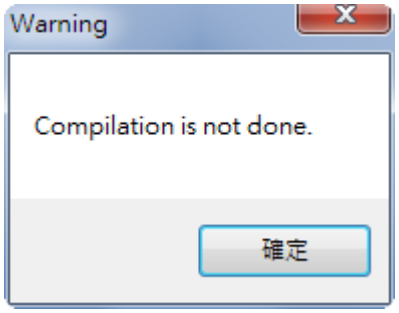


2. Edit function in Function Editor.

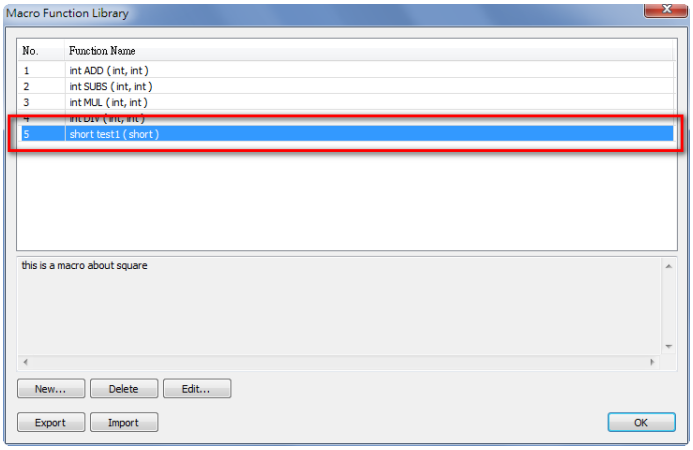
3. Edit the function description to describe what the specification is, how to use … etc.



4. After editing, click [Compile] and [Save] to save this function to the Library. Otherwise, a warning will be shown.



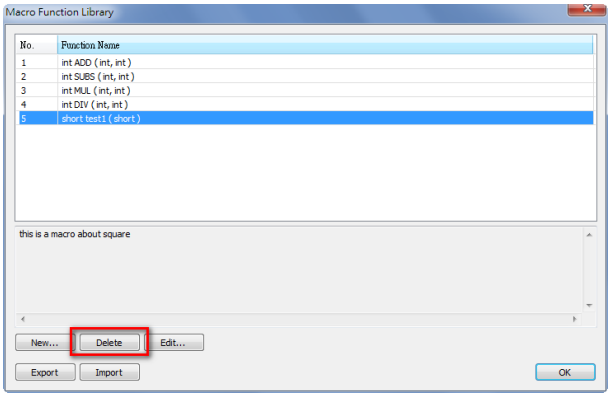
5. Successfully add a function into Macro Function Library.



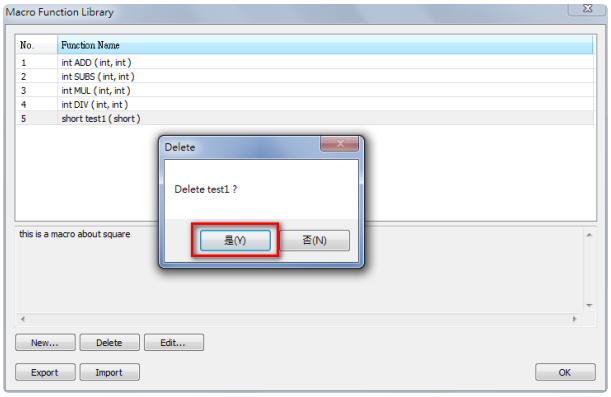
1. The total size of data type can be declared in a function is 4096 bytes.
2. Function name must only contain alphanumeric characters, and cannot start with a number.

18.8.3.2 Delete a Function

1. In function list select the function to be deleted and click [Delete].

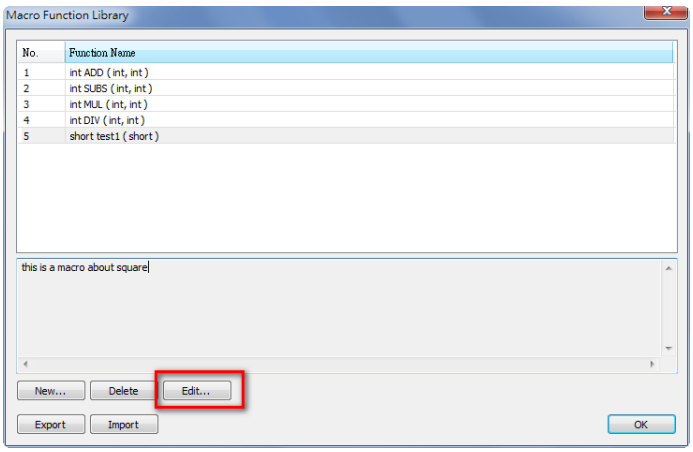


2. Click [Yes] to confirm, [No] to cancel the deletion.
3. Click [Yes] to delete MAX_SHORT function.

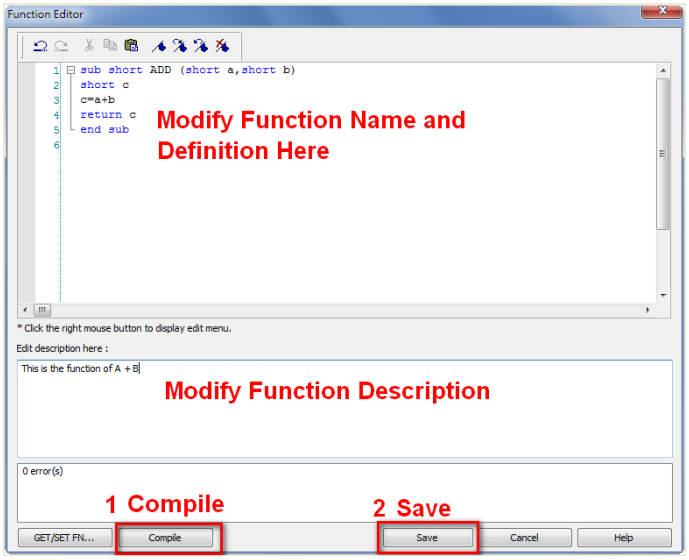


18.8.3.3 Modify a Function

1. Users can modify the functions exist in the Library.
2. Select a function to modify by clicking [Edit] to enter Function Editor

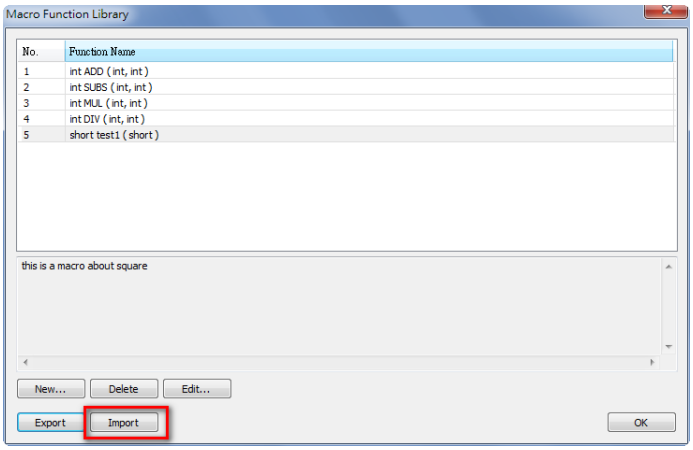


3. Double click on the function to be modified can also enter Function Editor.
4. After modifying, [Compile] then [Save] before leaving.

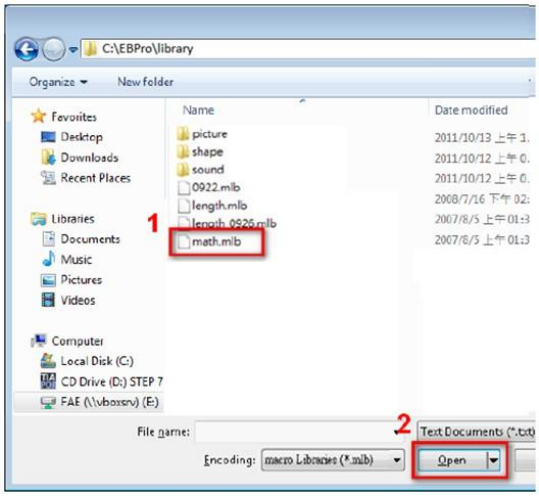


18.8.3.4 Import a Function

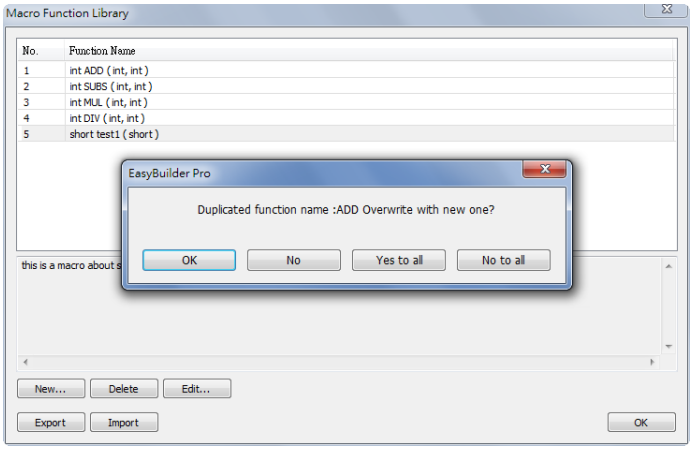
1. Functions can be imported using an external mlb file.



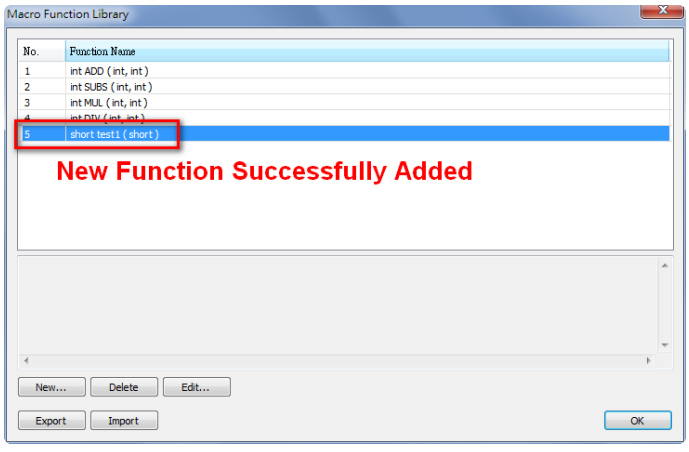
2. For example, import a function library “math.mlb” which contains a function “test1”.
3. Click [Open].



4. When importing a function which already exists in the Library, a confirmation popup will be shown. The buttons are:
   - [OK]: Overwrite the existing function with the imported one.
   - [NO]: Cancel the importing of the function with the same name.
   - [Yes to all]: Overwrite using all the imported functions with the same name.
   - [No to all]: Cancel the importing of all the functions with the same name.

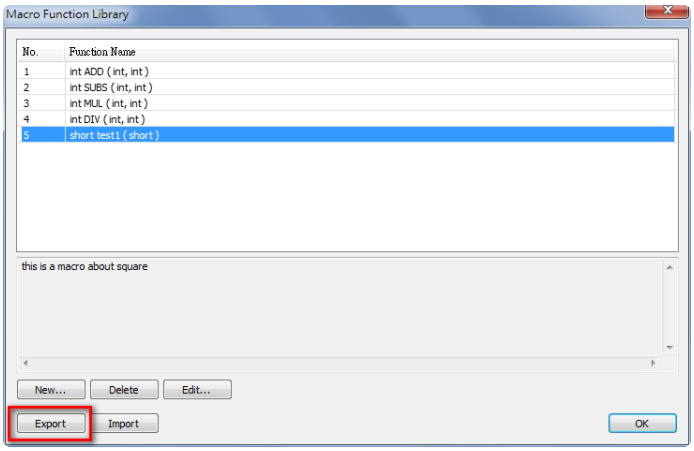


5. The imported functions will be saved in Default Function Library, so if “math.mlb” file is deleted, “test1” will still exist in the Library, even restarting EasyBuilder.

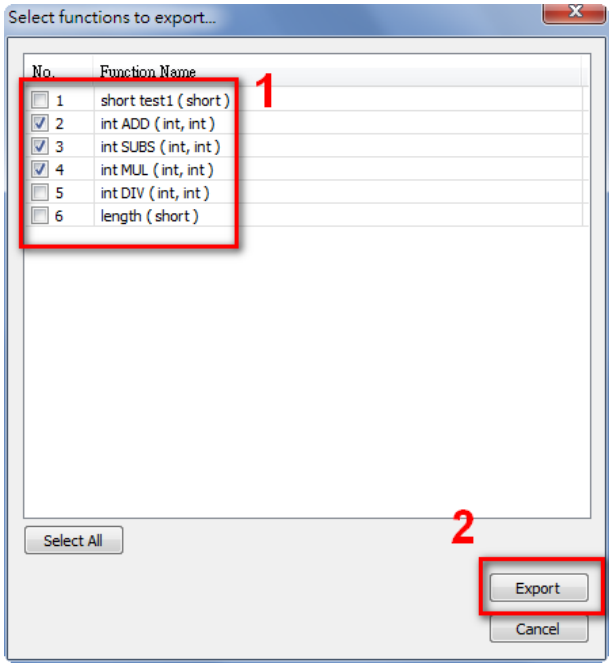


18.8.3.5 Export a Function

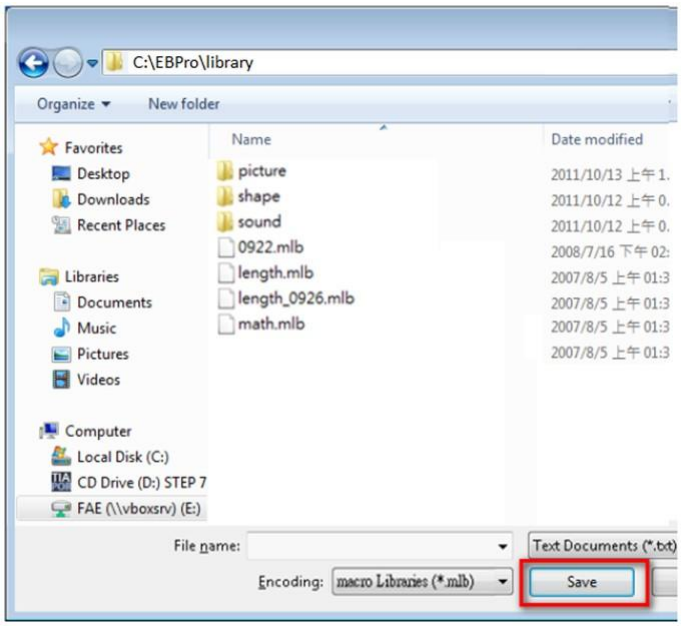
1. Export the function from Function Library and save as mlb file.
2. Click [Export].



3. Select the function to be exported, and click [Export].
4. A “math.mlb” file can be found under export directory. This file contains 4 functions: ADD, SUBS, MUL, and DIV.



5. The exported mlb file can be imported on another PC. Open HMI programming software, import, then the functions in this file can be used.



18.9 Some Notes about Using the Macro

1. The maximum storage space of local variables in a macro is 4K bytes. So the maximum array size of different variable types are as follows:

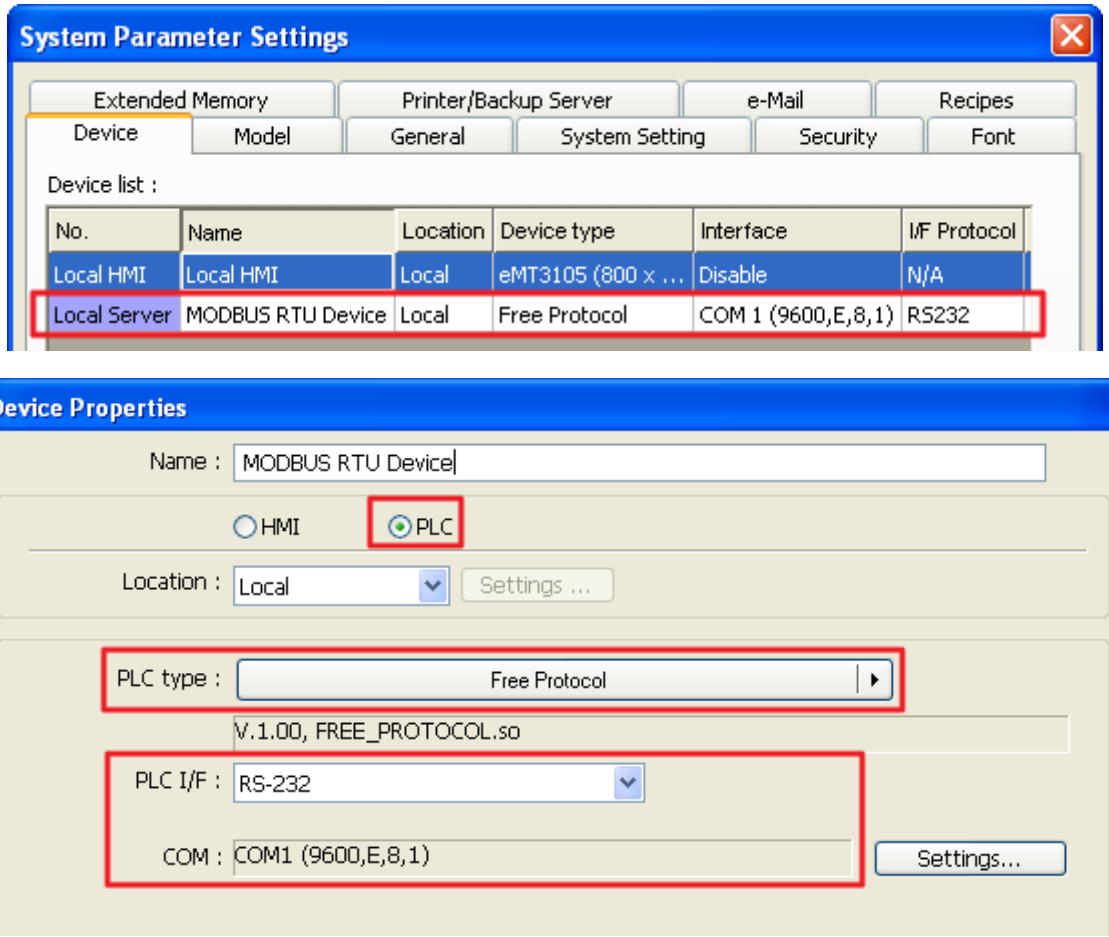
char	a[4096]
bool	b[4096]
short	c[2048]
int	d[1024]
float	e[1024]

2. A maximum of 255 macros are allowed in an EasyBuilder project.
3. A macro may cause the HMI unresponsive. Possible reasons are:
   - A macro contains an infinite loop with no PLC communication.
   - The size of an array exceeds the storage space in a macro.
4. The PLC communication speed affects the running time for the macro to execute. Also, too many macros may slow down the communication between HMI and PLC.

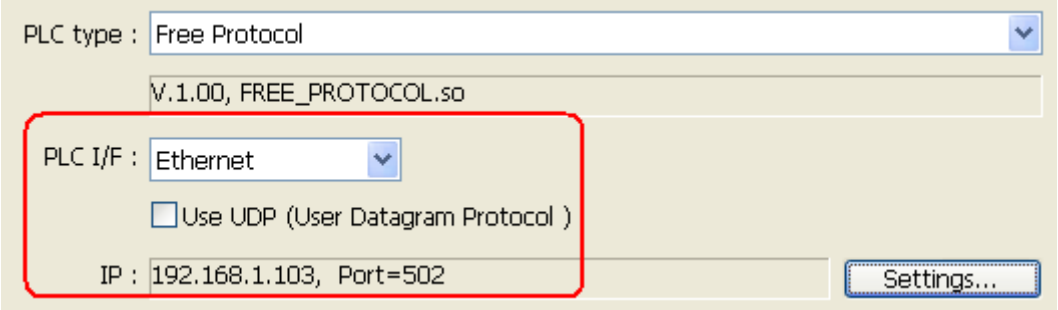
18.10 Use the Free Protocol to Control a Device

If EasyBuilder does not provide a driver for a specific device, users can use OUTPORT and INPORT built-in functions to control the device. The data sent by OUTPORT and INPORT must follow the communication protocol of the device. The following example explains how to use these two functions to control a MODBUS RTU device.

First, create a new device in the device table. The device type of the new device is set to “Free Protocol” and named with “MODBUS RTU device” as follows:



The interface of the device (PLC I/F) uses [RS-232]. If a MODBUS TCP/IP device is connected, the interface should be [Ethernet] with correct IP and port number as follows:



Suppose that the HMI will read the data of 4x_1 and 4x_2 on the device. First, utilize OUTPORT to send out a read request to the device. The format of OUTPORT is:

OUTPORT(command[start], device_name, cmd_count)

Since “MODBUS RTU device” is a MODBUS RTU device, the read request must follow MODBUS RTU protocol. The request uses”Reading Holding Registers (0x03)” command to read data. The following picture displays the content of the command. (The items of the station number (byte 0) and the last two bytes (CRC) are ignored).

Request

Function code	1 Byte	0x03
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Registers	2 Bytes	1 to 125 (0x7D)

Response

Function code	1 Byte	0x03
Byte count	1 Byte	2 x N*
Register value	N* x 2 Bytes	

*N = Quantity of Registers

Error

Error code	1 Byte	0x83
Exception code	1 Byte	01 or 02 or 03 or 04

Depending on the protocol, the content of a read command as follows (The total is 8 bytes):

command[0] : station number (BYTE 0)
command[1] : function code (BYTE 1)
command[2] : high byte of starting address (BYTE 2)
command[3] : low byte of starting address (BYTE 3)
command[4] : high byte of quantity of registers (BYTE 4)
command[5] : low byte of quantity of registers (BYTE 5)
command[6] : low byte of 16-bit CRC (BYTE 6)
command[7] : high byte of 16-bit CRC (BYTE 7)

So a read request is designed as follows:

```
char command[32]
short address, checksum

FILL(command[0], 0, 32)    //    initialize command[0]~command[31] to 0
```

```
command[0] = 0x1     // station number
command[1] = 0x3     // read holding registers (function code is 0x3)

address = 0          // starting address (4x_1) is 0
HIBYTE(address, command[2])
LOBYTE(address, command[3])

read_no = 2          // the total words of reading is 2 words
HIBYTE(read_no, command[4])
LOBYTE(read_no, command[5])

CRC(command[0], checksum, 6)    // calculate 16-bit CRC

LOBYTE(checksum, command[6])
HIBYTE(checksum, command[7])
```

Lastly, use OUPORT to send out this read request to PLC

```
OUTPORT(command[0], "MODBUS RTU Device", 8)      // send read request
```

After sending out the request, use INPORT to get the response from PLC. Depending on the protocol, the content of the response is as follows (the total byte is 9):

command[0] : station number (BYTE 0)
command[1] : function code (BYTE 1)
command[2] : byte count (BYTE 2)
command[3] : high byte of 4x_1 (BYTE 3)
command[4] : low byte of 4x_1 (BYTE 4)
command[5] : high byte of 4x_2 (BYTE 5)
command[6] : high byte of 4x_2 (BYTE 6)
command[7] : low byte of 16-bit CRC (BYTE 7)
command[8] : high byte of 16-bit CRC (BYTE 8)

The format of INPORT is:

```
INPORT(response[0], "MODBUS RTU Device", 9, return_value)  // read response
```

Where the real read count is restored to the variable return_value (unit is byte). If return_value is 0, it means reading fails in executing INPORT.
According to the MODBUS RTU protocol specification, the correct response[1] must be equal to 0x03. After getting correct response, calculate the data of 4x_1 and 4x_2 and put in the data into LW100 and LW101 of HMI.

```
if (return_value >0 and response[1] == 0x3) then
   read_data[0] = response[4] + (response[3] << 8)//   4x_1
   read_data[1] = response[6] + (response[5] << 8)//   4x_2

   SetData(read_data[0], "Local HMI", LW, 100, 2)
end if
```

The complete macro is as follows:

```
// Read Holding Registers
macro_command main()

char command[32], response[32]
short address, checksum
short read_no, return_value, read_data[2], i

FILL(command[0], 0, 32)//   initialize command[0]~command[31] to 0
FILL(response[0], 0, 32)

command[0] = 0x1//   station number
command[1] = 0x3//   read holding registers (function code is 0x3)

address = 0
address =   0//   starting address (4x_1) is 0
HIBYTE(address, command[2])
LOBYTE(address, command[3])

read_no = 2/   the total words of reading is 2 words
HIBYTE(read_no, command[4])
LOBYTE(read_no, command[5])

CRC(command[0], checksum, 6)//   calculate 16-bit CRC
```

```
LOBYTE(checksum, command[6])
HIBYTE(checksum, command[7])

OUTPORT(command[0], "MODBUS RTU Device", 8 )//    send request
INPORT(response[0], "MODBUS RTU Device", 9, return_value)//    read response

if (return_value > 0 and response[1] == 0x3) then
   read_data[0] = response[4] + (response[3] << 8)//    4x_1
   read_data[1] = response[6] + (response[5] << 8)//    4x_2

   SetData(read_data[0], "Local HMI", LW, 100, 2)
end if

end macro_command
```

The following example explains how to design a request to set the status of 0x_1. The request uses ”Write Single Coil(0x5)” command.

Request

Function code	1 Byte	**0x05**
Output Address	2 Bytes	0x0000 to 0xFFFF
Output Value	2 Bytes	0x0000 or 0xFF00

Response

Function code	1 Byte	**0x05**
Output Address	2 Bytes	0x0000 to 0xFFFF
Output Value	2 Bytes	0x0000 or 0xFF00

Error

Error code	1 Byte	**0x85**
Exception code	1 Byte	01 or 02 or 03 or 04

The complete macro is as follows:

```
//    Write Single Coil (ON)
macro_command main()

char command[32], response[32]
short address, checksum
short i, return_value
```

```
FILL(command[0], 0, 32)//    initialize command[0]~ command[31] to 0
FILL(response[0], 0, 32)

command[0] = 0x1//    station number
command[1] = 0x5//    function code : write single coil

address = 0
HIBYTE(address, command[2])
LOBYTE(address, command[3])

command[4] = 0xff//    force 0x_1 on
command[5] = 0

CRC(command[0], checksum, 6)

LOBYTE(checksum, command[6])
HIBYTE(checksum, command[7])

OUTPORT(command[0], "MODBUS RTU Device", 8)//    send request
INPORT(response[0], "MODBUS RTU Device", 8, return_value)//    read response

end macro_command
```

18.11 Compiler Error Message

1. Error Message Format:

error C# : error description
(# is the error message number)
Example: error C37 : undeclared identifier : i

When there are compile errors, the description of the error can be found by the compiler error message number.

2. Error Description

(C1) syntax error：'identifier'

There are many possibilities to cause compiler error.

For example:

```
macro_command main()
char i, 123xyz    //    this is an unsupported variable name
      end macro_command
```

(C2) 'identifier' used without having been initialized

Macro must define the size of an array during declaration.

For example:

```
macro_command main()
char i
int g[i]    //    i must be a numeric constant
end macro_command
```

(C3) redefinition error : 'identifier'

The name of variable and function within its scope must be unique.

For example:

```
macro_command main()
int g[10]，g    //    error
end macro_command
```

(C4) function name error : 'identifier'

Reserved keywords and constant can not be the name of a function

For example：

sub int if() // error

(C5) parentheses have not come in pairs

Statement missing "(" or ")"

For example：

macro_command main) // missing "("

(C6) illegal expression without matching 'if'

Missing expression in "if" statement

(C7) illegal expression (no 'then') without matching 'if'

Missing "then" in "if" statement

(C8) illegal expression (no 'end if')

Missing "end if"

(C9) illegal 'end if' without matching 'if'

Unfinished "If' statement before "End If"

(C10) illegal 'else'

The format of "if" statement is :

if [logic expression] then

[else [if [logic expression] then]]

end if

Any format other than this format will cause a compile error.

(C17) illegal expression (no 'for') without matching 'next'

"for" statement error : missing "for" before "next"

(C18) illegal variable type (not integer or char)

Should be integer or char variable

(C19) variable type error

Missing assign statement

(C20) must be keyword ‘to’ or ‘down’

Missing keyword “to” or “down”

(C21) illegal expression (no 'next')

The format of “for” statement is:

for [variable] = [initial value] to [end value] [step]

next [variable]

Any format other than this format will cause a compile error.

(C22) ‘wend’ statement contains no ‘while’

“While” statement error : missing “while” before “Wend”

(C23) illegal expression without matching ‘wend’

The format of “While” statement is :

while [logic expression]

wend

Any format other than this format will cause a compile error.

(C24) syntax error : ‘break’

“break” statement can only be used in “for”, “while” statement.

(C25) syntax error : ‘continue’

“continue” statement can only be used in “for” statement, or “while” statement.

(C26) syntax error

Error in expression.

(C27) syntax error

The mismatch of an operation object in expression can cause a compile error.

For example :

```
macro_command main( )
int a, b
for a = 0 to 2
b = 4 + xyz //    illegal : xyz is undefined
next a
end macro_command
```

(C28) must be 'macro_command'

There must be 'macro_command'

(C29) must be key word 'sub'

The format of function declaration is:

```
sub [data type] function_name(…)
……..
end sub
```

For example::

```
sub int pow(int exp)
…….
end sub
```

Any format other than this format will cause a compile error.

(C30) number of parameters is incorrect

Mismatch of the number of parameters

(C31) parameter type is incorrect

Mismatch of data type of parameter. When a function is called, the data type and the number of parameters should match the declaration of function, otherwise it will cause a compile error.

(C32) variable is incorrect

The parameters of a function must be equivalent to the arguments passing to a function to avoid compile error.

(C33) function name : undeclared function

(C34) expected constant expression

Illegal array index format.

(C35) invalid array declaration

(C36) array index error

(C37) undeclared identifier : i 'identifier'

Any variable or function should be declared before use.

(C38) un-supported PLC data address

The parameter of GetData(…) , SetData(…) should be legal PLC address. If the address is illegal, this error message will be shown.

(C39) 'idenifier' must be integer, char or constant

The format of array is:

Declaration: array_name[constant] (constant is the size of the array)

Usage: array_name[integer, character or constant]

Any format other than this format will cause a compile error.

(C40) execution syntax should not exist before variable declaration or constant definition

For example :

```
macro_command main(    )
int a, b
for a = 0 To 2
  b = 4 + a
int h , k //    illegal – definitions must occur before any statements or expressions
          //    for example, b = 4 + a
next a
end macro_command
```

(C41) float variables cannot be contained in shift calculation

(C42) function must return a value

(C43) function should not return a value

(C44) float variables cannot be contained in calculation

(C45) PLC address error

(C46) array size overflow (max. 4k)

(C47) macro command entry function is not only one

(C48) macro command entry function must be only one

The only one main entrance of macro is :

```
macro_command function_name( )
end macro_command
```

(C49) an extended addressee's station number must be between 0 and 255

For example :

```
SetData(bits[0] , "PLC 1", LB , 300#123, 100)
//   illegal : 300#123 means the station number is 300, but the maximum is 255
```

(C50) an invalid PLC name

PLC name is not defined in the device list of system parameters.

(C51) macro command do not control a remote device

A macro can only control a local machine.

For example :

SetData(bits[0] , "PLC 1", LB , 300#123, 100)

"PLC 1" is connected with the remote HMI ,so it can not work.

18.12 Sample Macro Code

1. ”for” statement and other expressions (arithmetic, bitwise shift, logic and comparison)

```
macro_command main()
int a[10], b[10], i

b[0]= (400 + 400 << 2) / 401
b[1]= 22 *2 - 30 % 7
b[2]= 111 >> 2
b[3]= 403 > 9 + 3 >= 9 + 3 < 4 + 3 <= 8 + 8 == 8
b[4]= not 8 + 1 and 2 + 1 or 0 + 1 xor 2
b[5]= 405 and 3 and not 0
b[6]= 8 & 4 + 4 & 4 + 8 | 4 + 8 ^ 4
b[7]= 6 – (~4)
b[8]= 0x11
b[9]= 409

for i = 0 to 4 step 1
    if (a[0] == 400) then
        GetData(a[0],”Device 1”, 4x, 0,9)
        GetData(b[0],”Device 1”, 4x, 11,10)
end If
next i
end macro_command
```

2. “while”, “if” and “break” statements

```
macro_command main()
int b[10], i
i = 5
while i == 5 - 20 % 3
    GetData(b[1], ”Device 1”, 4x, 11, 1)

    if b[1] == 100 then
        break
    end if
```

```
wend
end macro_command
```

3. Global variables and function call

```
char g
sub int fun(int j, int k)
int y

SetData(j, “Local HMI”, LB, 14, 1)
GetData(y, “Local HMI”, LB, 15, 1)
g = y

return y
end Sub

macro_command main()
int a, b, i

a = 2
b = 3
i = fun(a, b)
SetData(i, “Local HMI”, LB, 16, 1)
end macro_command
```

4. ”if” statement

```
macro_command main()
int k[10], j

for j = 0 to 10
    k[j] = j
next j

if k[0] == 0 then
    SetData(k[1], “Device 1”, 4x, 0, 1)
end if
```

```
if k[0] == 0 then
    SetData(k[1], “Device 1”, 4x, 0, 1)
else
    SetData(k[2], “Device 1”, 4x, 0, 1)
end if

if k[0] == 0 then
    SetData(k[1], “Device 1”, 4x, 1, 1)
else if k[2] == 1 then
    SetData(k[3], “Device 1”, 4x, 2, 1)
end If

if k[0] == 0 then
    SetData(k[1], “Device 1”, 4x, 3, 1)
else if k[2] == 2 then
    SetData(k[3], “Device 1”, 4x, 4, 1)
else
    SetData(k[4], “Device 1”, 4x, 5, 1)
end If
end macro_command
```

5. “while” and wend” statements

```
macro_command main()
char i = 0
int a[13], b[14], c = 4848

b[0] = 13

while b[0]
    a[i] = 20 + i * 10

    if a[i] == 120 then
        c =200
        break
    end if

    i = i + 1
```

```
wend

SetData(c, “Device 1”, 4x, 2, 1)
end macro_command
```

6. “break” and “continue” statements

```
macro_command main()
char i = 0
int a[13], b[14], c = 4848

b[0] = 13

while b[0]
    a[i] = 20 + i * 10

    if a[i] == 120 then
    c =200
        i = i + 1
        continue
    end if

    i = i + 1

    if c == 200 then
    SetData(c, “Device 1”, 4x, 2, 1)
    break
    end if
wend
end macro_command
```

7. Array

```
macro_command main()
int a[25], b[25], i

b[0] = 13
```

```
for i = 0 to b[0] step 1
      a[i] = 20 + i * 10
next i

SetData(a[0], “Device 1”, 4x, 0, 13)
end macro_command
```

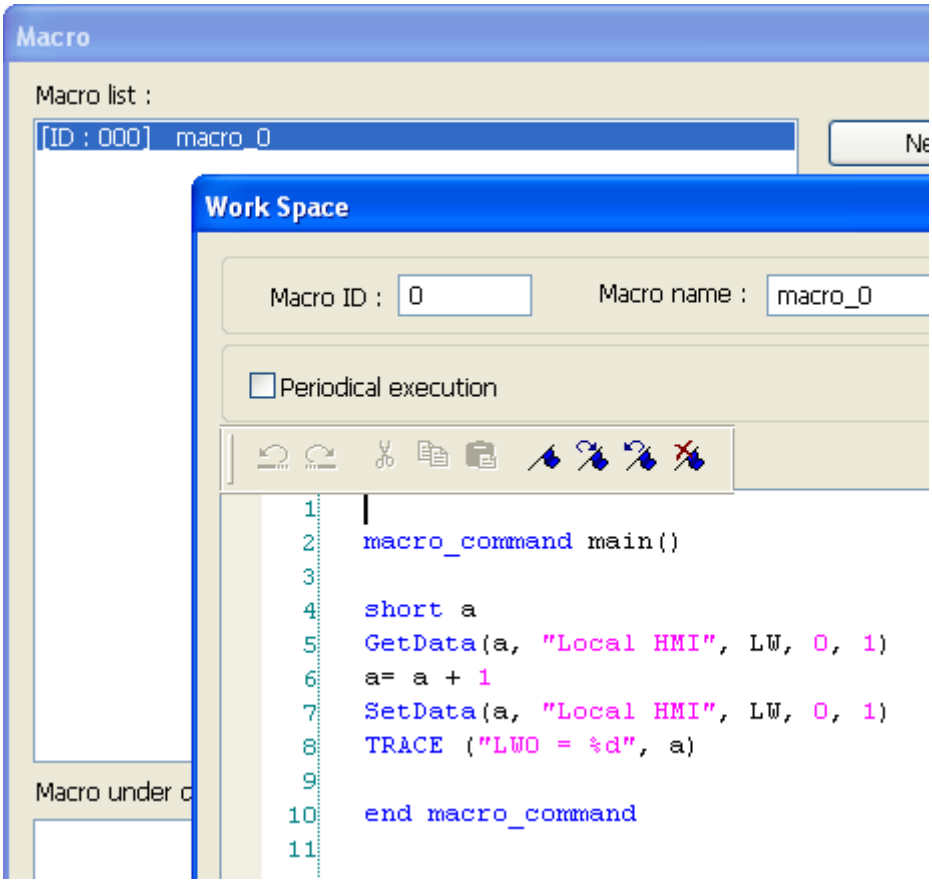
18.13 Macro TRACE Function

TRACE function can be used with EasyDiagnoser to show the current content of the variables. The following example illustrates how TRACE function could be used in macro.

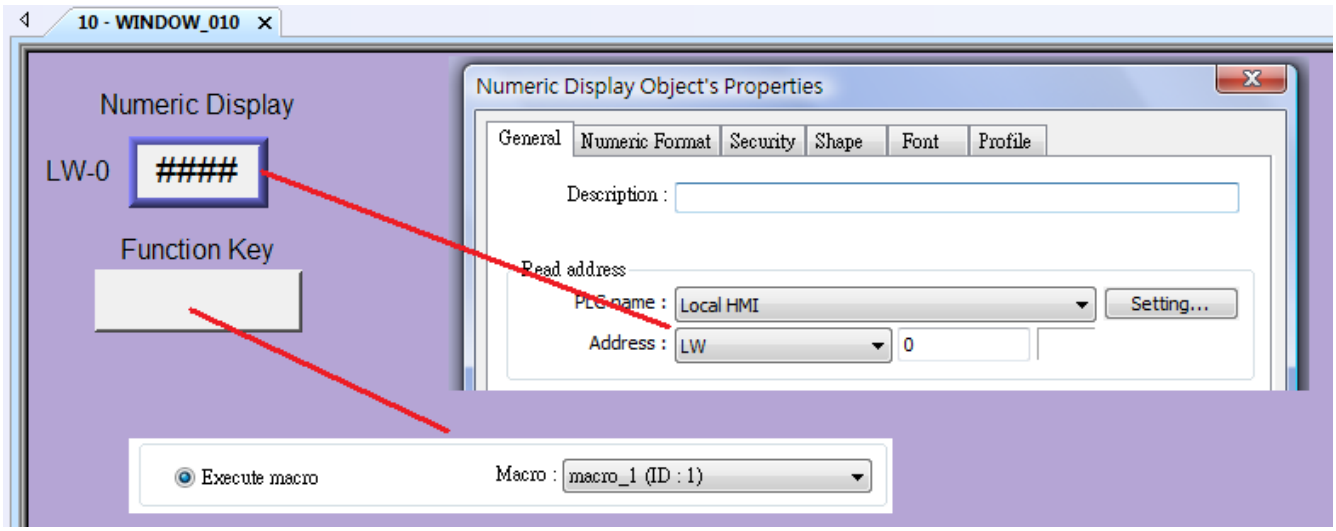
First of all, add a new macro “macro_1” in the project, and in “macro_1” add TRACE (“LW = %d”, a). “%d” indicates display current value of LW in decimal format. The content of macro_1 is as follows:

```
macro_command main()

short a
GetData(a, "Local HMI", LW, 0, 1)
a= a + 1
SetData(a, "Local HMI", LW, 0, 1)
TRACE ("LW0 = %d", a)

end macro_command
```

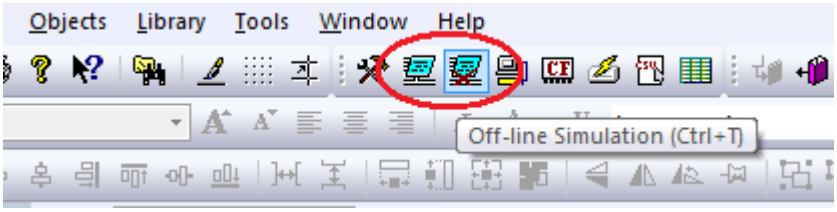
(For the detailed usage of TRACE function, please refer to the following paragraph.)



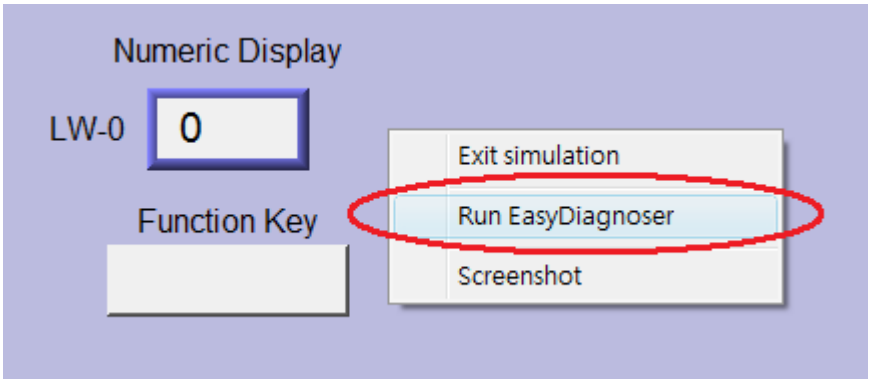
Secondly, add a [Numeric Display] object and a [Function Key] object in window no. 10 of the project. The settings of these objects are shown below. [Function Key] object is used to execute macro_1.



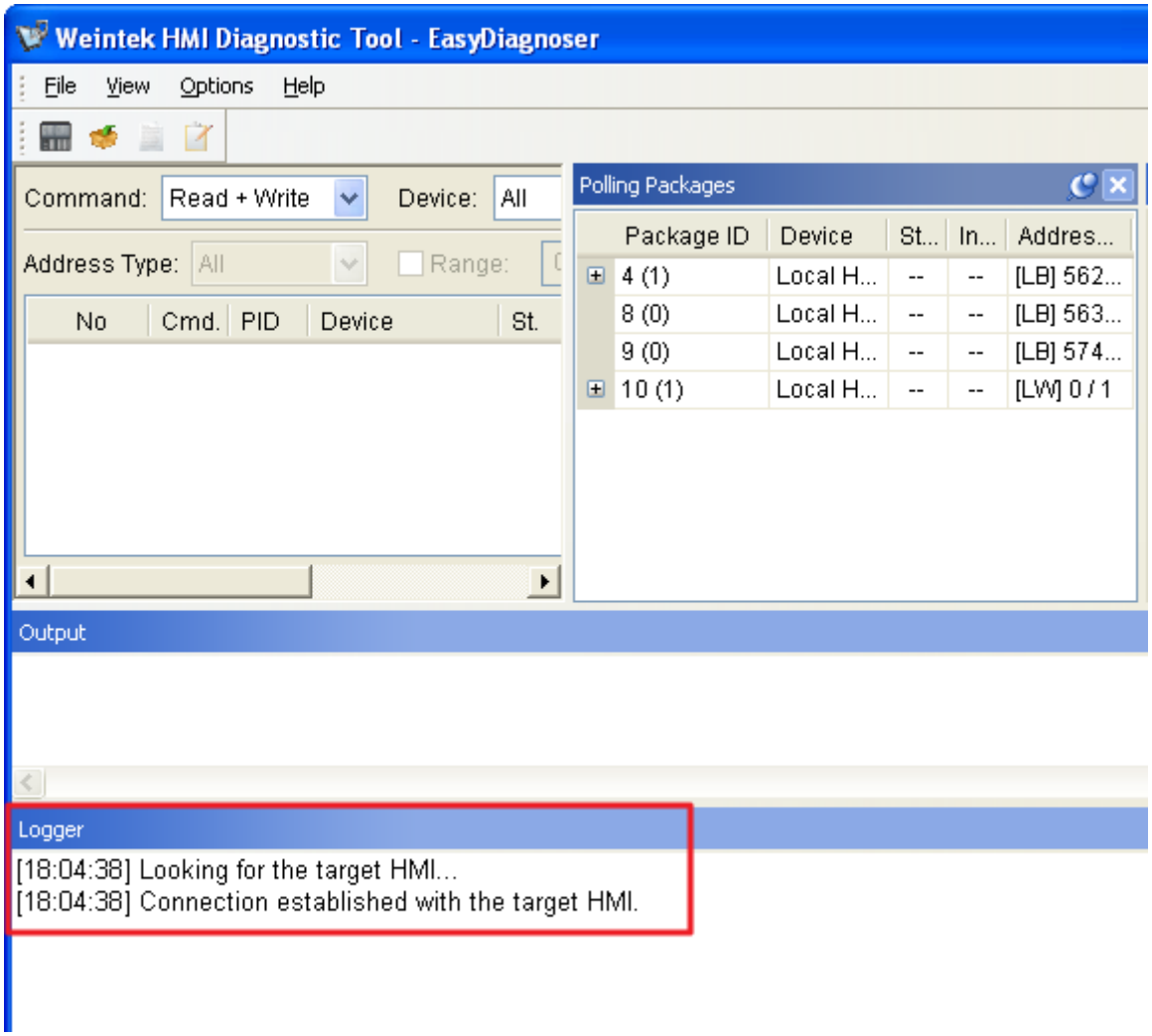
Lastly, compile the project and execute [Off-line simulation] or [On-line simulation].



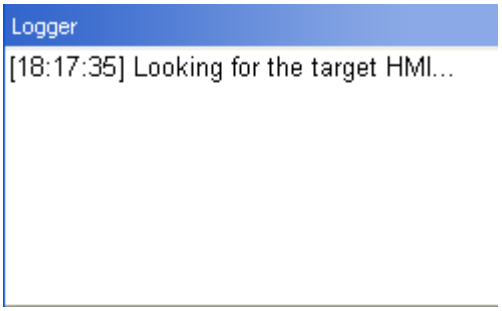
When processing simulation on PC, right click and select “Run EasyDiagnoser” in the pop-up menu.



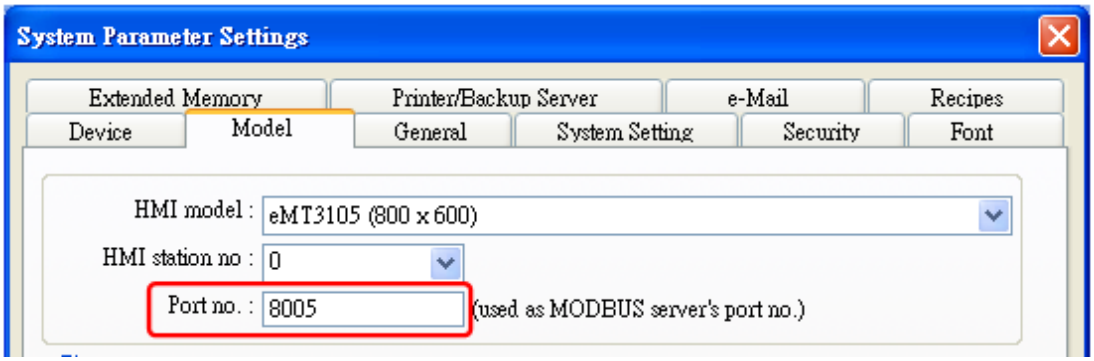
Afterwards, EasyDiagnoser will be started. [Logger] window displays whether EasyDiagnoser is able to connect with the HMI to be watched or not. [Output] window displays the output of the TRACE function. The illustration below shows that EasyDiagnoser succeeds in connecting with HMI.



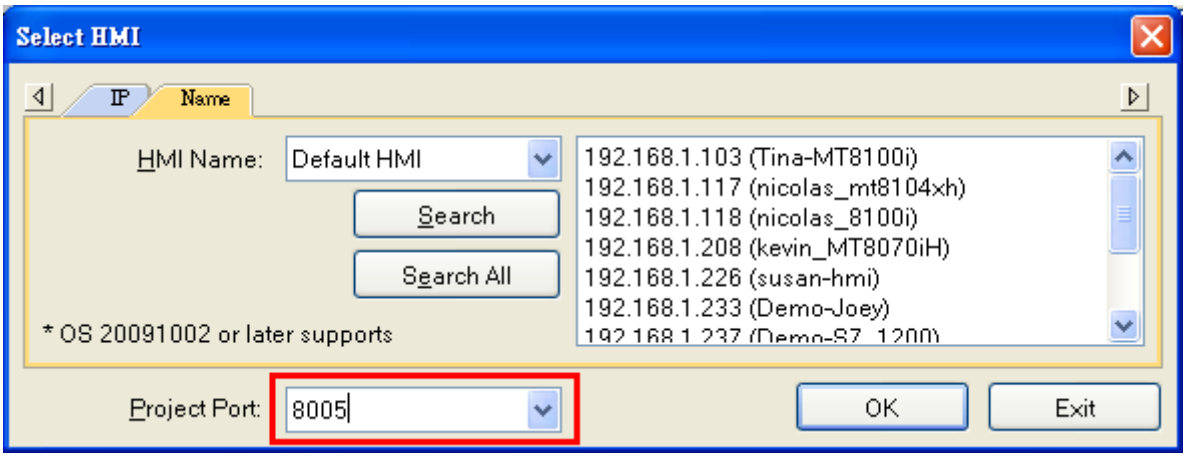
When EasyDiagnoser is not able to connect with HMI, [Logger] window displays content as shown below:



The possible reason of not being able to get connection with HMI can be failure in executing simulation on PC. Another reason is that the Port No. used in project for simulation on PC is incorrect (or occupied by system). Please change Port No. as shown, compile project then do simulation again.



In EasyDiagnoser, the Port No. should be set the same as the Port No. in the project.



The three consecutive ports of the project port no. are preserved for HMI communication. In the setting above as an example, Port No. is set as 8005. Port 8005, 8006 and 8007 should be reserved. In this case when executing simulation on PC, please make sure that these ports are not occupied by other programs.

1. TRACE Syntax List：

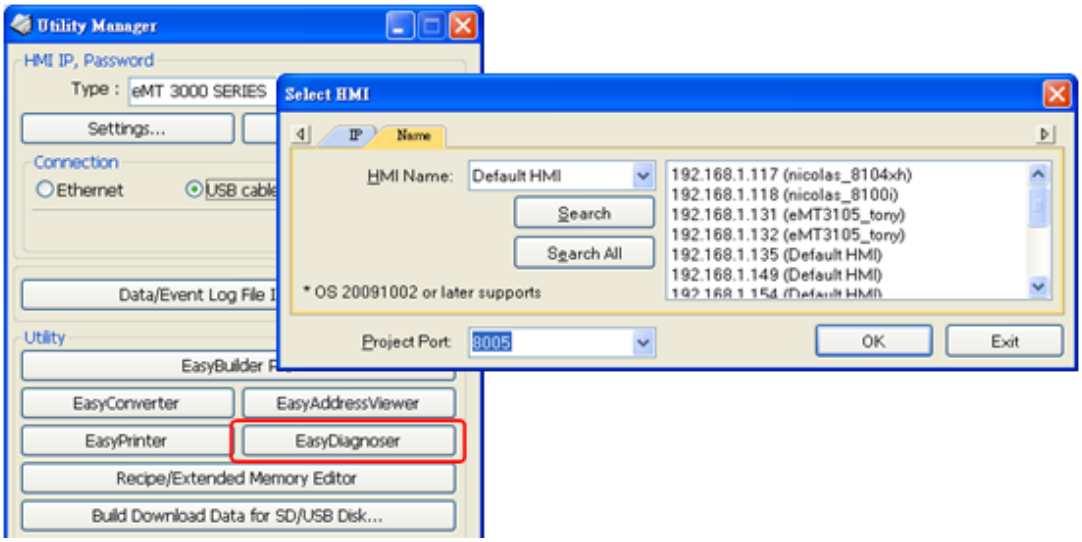
Name	TRACE
Syntax	TRACE(format, argument)
Description	Use this function to send specified string to the EasyDiagnoser. Users can print out the current value of variables during run-time of macro for debugging. When TRACE encounters the first format specification (if any), it converts the value of the first argument after format and outputs it accordingly. *format* refers to the format control of output string. A format specification, which consists of optional (in [ ]) and required fields (in bold), has the following form: **%**[flags] [width] [.precision] **type** Each field of the format specification is described as below:

	flags (optional): - + *width* (optional): A nonnegative decimal integer controlling the minimum number of characters printed. *precision* (optional): A nonnegative decimal integer which specifies the precision and the number of characters to be printed. *type*: C or c : specifies a single-byte character. d : signed decimal integer. i : signed decimal integer. o : unsigned octal integer. u : unsigned decimal integer. X or x : unsigned hexadecimal integer. E or e : Signed value having the form. [–]*d***.***dddd* **e** [*sign*]*ddd* where *d* is a single decimal digit, *dddd* is one or more decimal digits, *ddd* is exactly three decimal digits, and *sign* is + or –. f : Signed value having the form [–]*dddd***.***dddd*, where *dddd* is one or more decimal digits. The length of output string is limited to 256 characters. The *argument* part is optional.

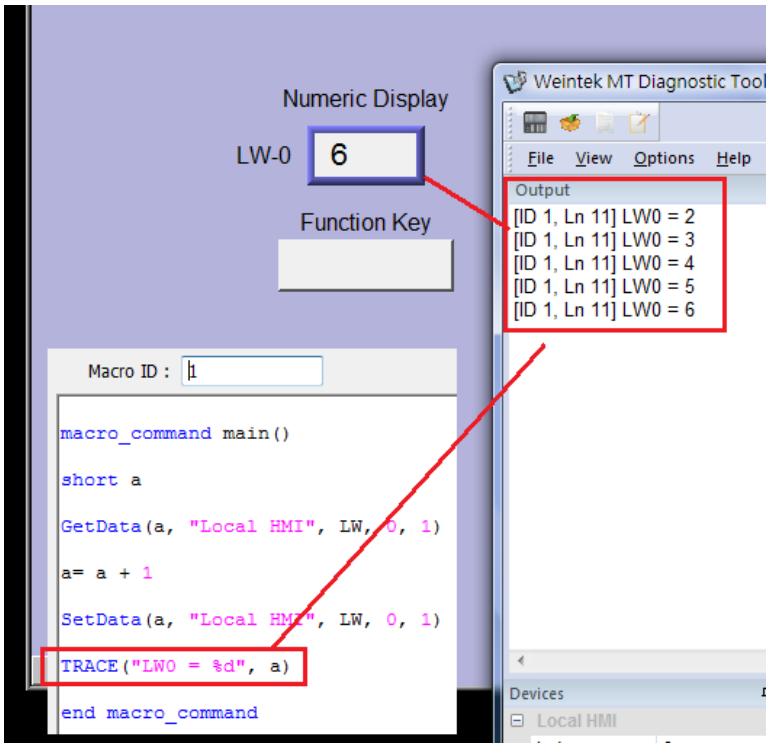
Example	macro_command main()
	char c1 = 'a'
	short s1 = 32767
	float f1 = 1.234567
	
	TRACE("The results are") // output: The results are
	TRACE("c1 = %c, s1 = %d, f1 = %f", c1, s1, f1)
	// output: c1 = a, s1 = 32767, f1 = 1.234567
	
	end macro_command

2. Use LB9059 to disable MACRO TRACE function (when ON). When set ON, the output message of TRACE won't be sent to EasyDiagnoser.

3. Users can directly execute EasyDiagnoser.exe from Utility Manager. In Utility Manager, current HMI on line will be listed; users can simply select the HMI to be watched. Please note that Project Port should be the same as Port No. used in project file.

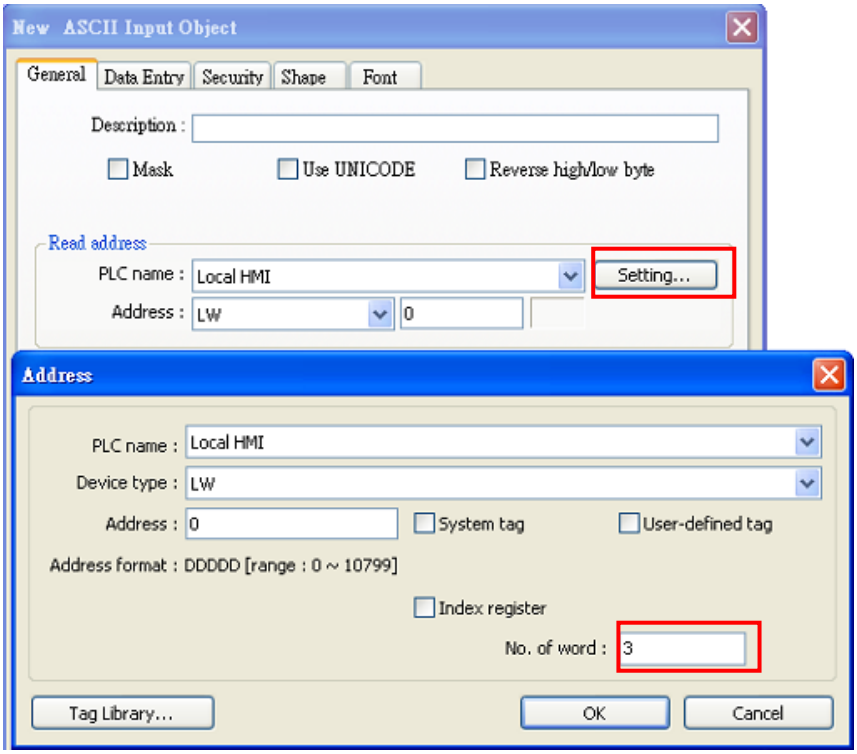


4. Download the project to HMI and start the project. If EasyDiagnoser is unable to get connection with the HMI to be watched, it is possible that HMI power is not ON, or Port No. is incorrect. This may cause EasyDiagnoser to connect then disconnect with HMI continuously. Please check the Port No. in EasyDiagnoser settings.
5. When EasyDiagnoser succeeds in connecting with HMI, simply execute macro_1, [Output] window will then display the output of the TRACE function.



18.14 Example of String Operation Functions

String operation functions are added to macro to provide a convenient way to operate strings. The term “string” means a sequence of ASCII characters, and each of them occupies 1 byte. The sequence of characters can be stored into 16-bit registers with least significant byte first. For example, create an [ASCII Input] object and setup as follows:



Run simulation and input “abcdef”:

The string “abcdef” is stored in LW0~LW2 as follows (LB represents low byte and HB represents high byte):

	HB	LB
LW0	'B'	'A'
LW1	'D'	'C'
LW2	'F'	'E'
LW3		
LW4		
LW5		

The ASCII input object reads 1 word (2 bytes) at a time as described in the previous chapter. Suppose an ASCII input object is set to read 3 words as shown in the above example, it can actually read at most 6 ASCII characters since that one ASCII character occupies 1 byte.

The functionality of each string operation function is described in the following table:

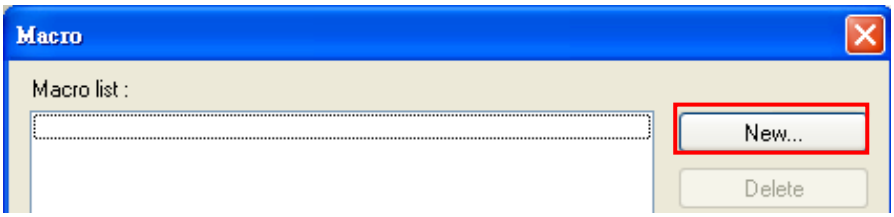
Function name	Description
StringGet	Read string data from a device.
StringGetEx	Read string data from a device and continue executing next command even if no response from that device.
StringSet	Write string data to a device.
StringSetEx	Write string data to a device and continue executing next command even if no response from that device.
StringCopy	Copy one string to another.
StringMid	Retrieve a substring.
StringDecAsc2Bin	Convert a decimal string to an integer.
StringBin2DecAsc	Convert an integer to a decimal string.
StringDecAsc2Float	Convert a decimal string to floats.
StringFloat2DecAsc	Convert a float to a decimal string.
StringHexAsc2Bin	Convert a hexadecimal string to binary data.
StringBin2HexAsc	Convert binary data into a hexadecimal string.
StringLength	Obtain the length of a string.
StringCat	Append source string to destination string.
StringCompare	Do a case-sensitive comparison of two strings.
StringCompareNoCase	Do a case-insensitive comparison of two strings.
StringFind	Find a substring inside a larger string.
StringReverseFind	Find a substring inside a larger string; starts from the end.
StringFindOneOf	Find the first matching character from a set.
StringIncluding	Extracts a substring that contains only the characters in a set.
StringExcluding	Extracts a substring that contains only the characters not in a set.
StringToUpper	Convert the characters of a string to uppercase.
StringToLower	Convert the characters of a string to lowercase.
StringToReverse	Reverse the characters of a string.
StringTrimLeft	Trim the leading specified characters in a set from the source string.
StringTrimRight	Trim the trailing specified characters in a set from the source string.
StringInsert	Insert a string in a specific location within another

	string.

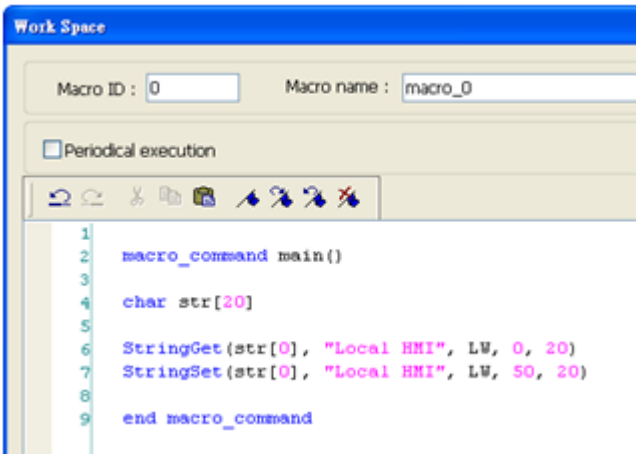
For more detailed information of the above string operation functions, please check out the “Built-In Function Block” section. In order to demonstrate the powerful usage of string operation functions, the following examples will show you step by step how to create executable project files using the new functions; starts from creating a macro, ends in executing simulation.

1. How to read (or write) a string from a device.

Create a new macro:

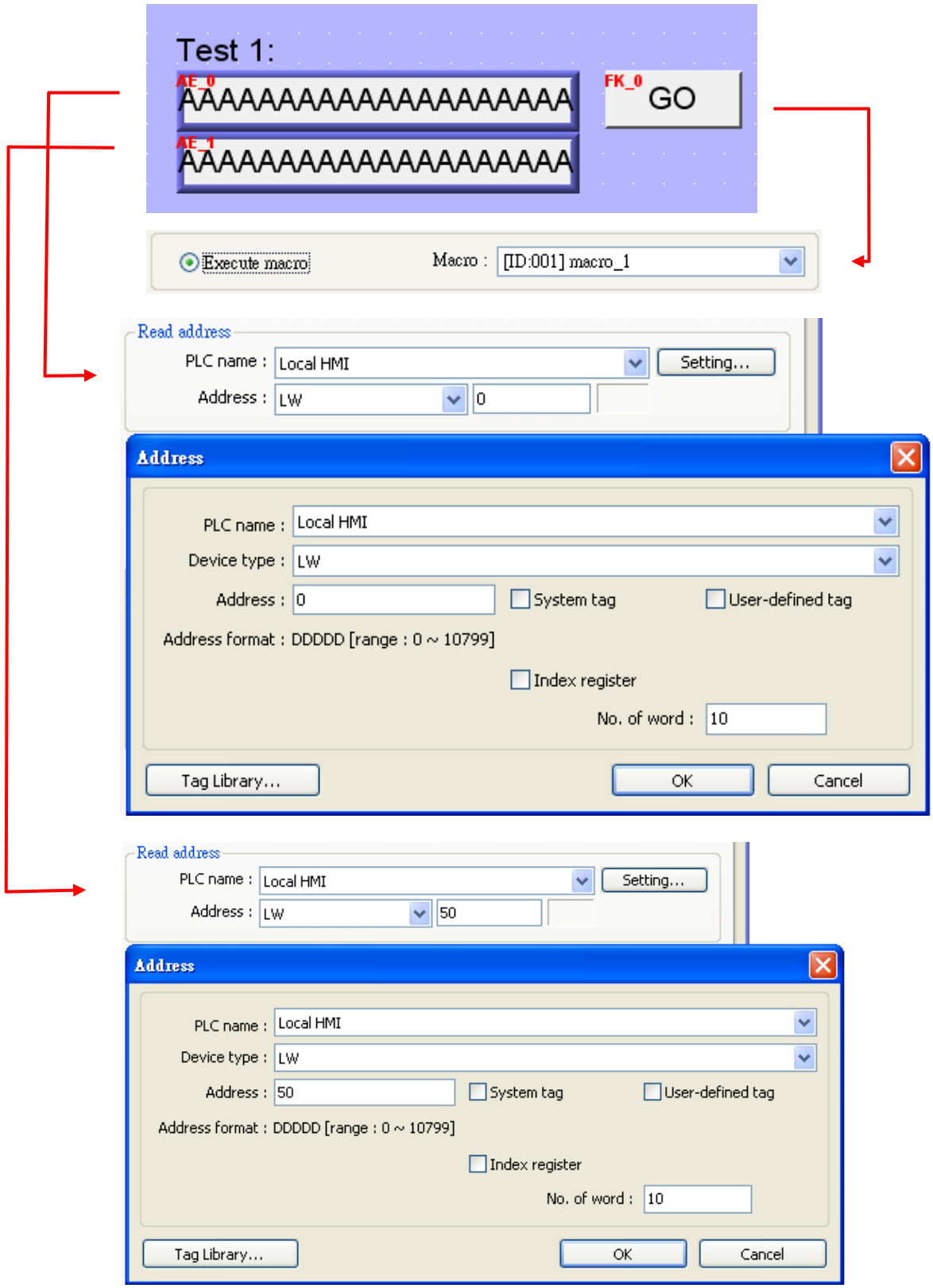


Edit the content:

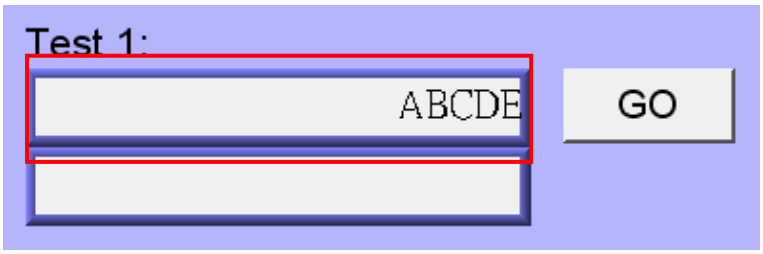


The first function “StringGet” is used to read a string from LW0~LW19, and store it into the str array. The second function “StringSet” is used to output the content of str array.

Add one [ASCII Input] object and one [Function Key] object in window 10 of the project. The settings of these objects are shown as below. Function Key object is used to execute macro_1.

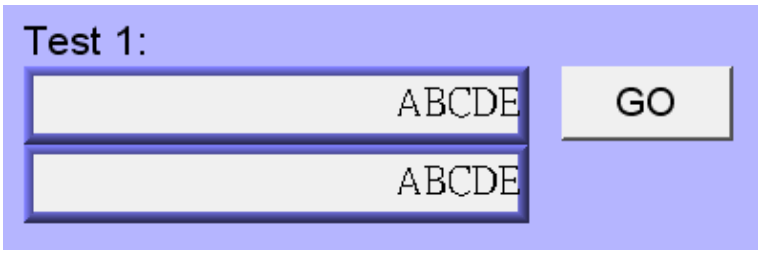


Lastly, use [Compile] to compile the project and execute [Off-line simulation] or [On-line simulation]. Follow the steps below to operate the executing project:



Step 1: input string

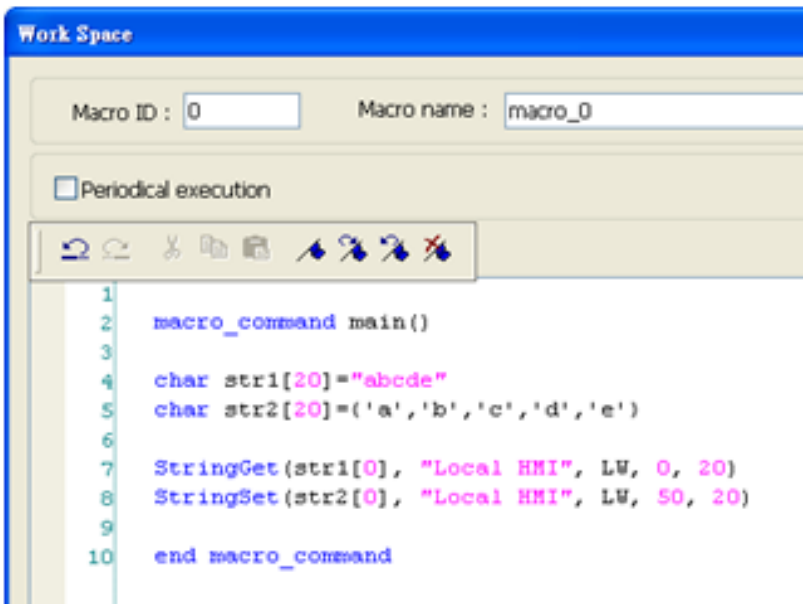
Step 2: press “GO” button



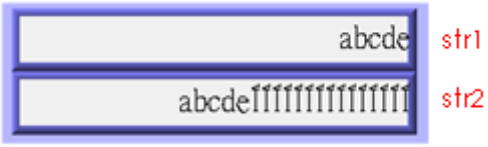
Step 3: output string

2. Initialization of a string.

Create a new macro and edit the content:



The data enclosed in double quotation mark (“”) is viewed as a string. str1 is initialized as a string while str2 is initialized as a char array. The following snapshot of simulation shows the difference between str1 and str2 using two ASCII input objects.

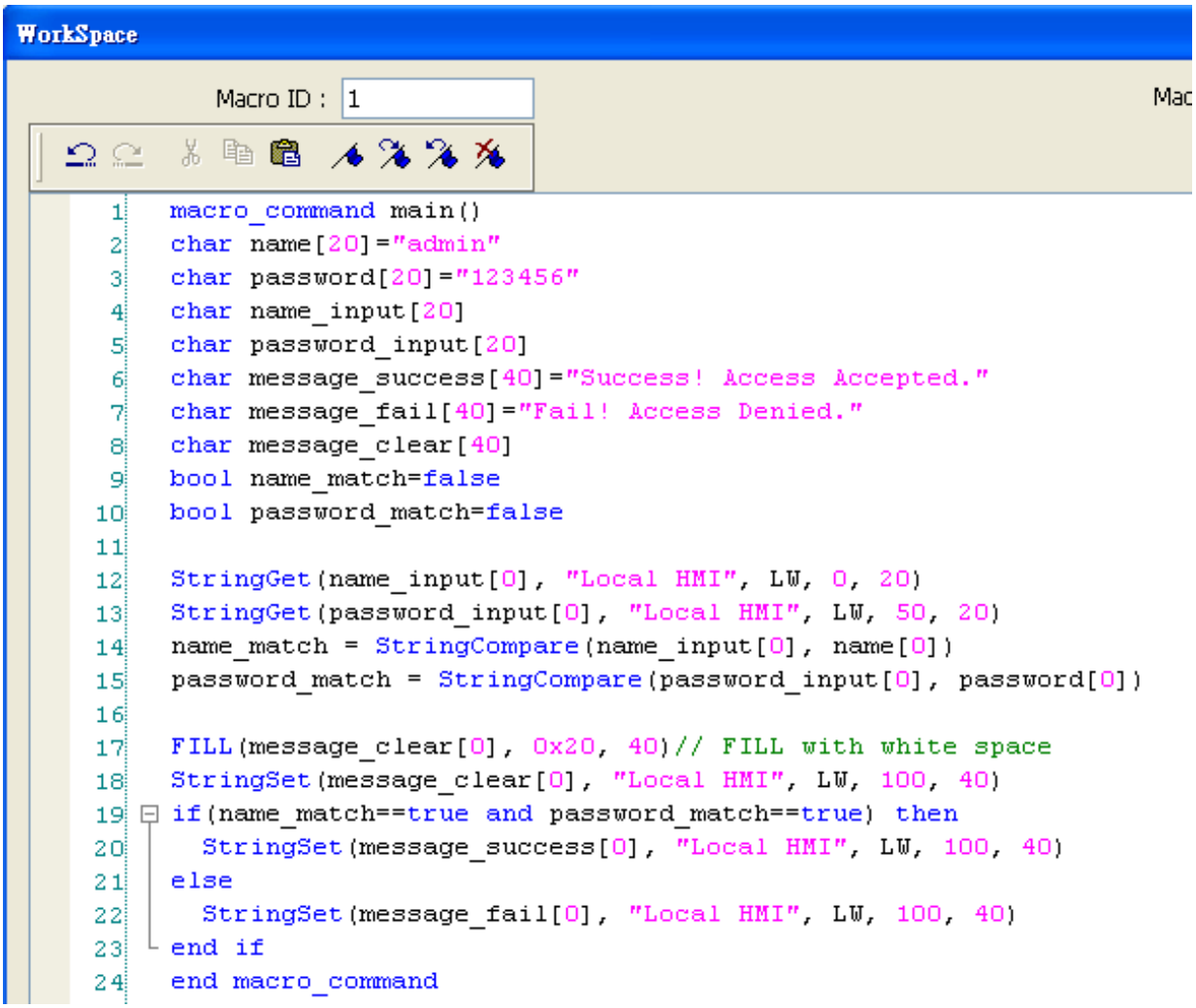


Macro compiler will add a terminating null character (‘\0’) at the end of a string. The function “StringSet” will send each character of str1 to registers until a null character is reached. The extra characters following the null character will be ignored even if the data count is set to a larger value than the length of string.

On the contrary, macro compiler will not add a terminating null character (‘\0’) at the end of a char array. The actual number of characters of str2 being sent to registers depends on the value of data count that is passed to the “StringSet” function.

3. A simple login page.

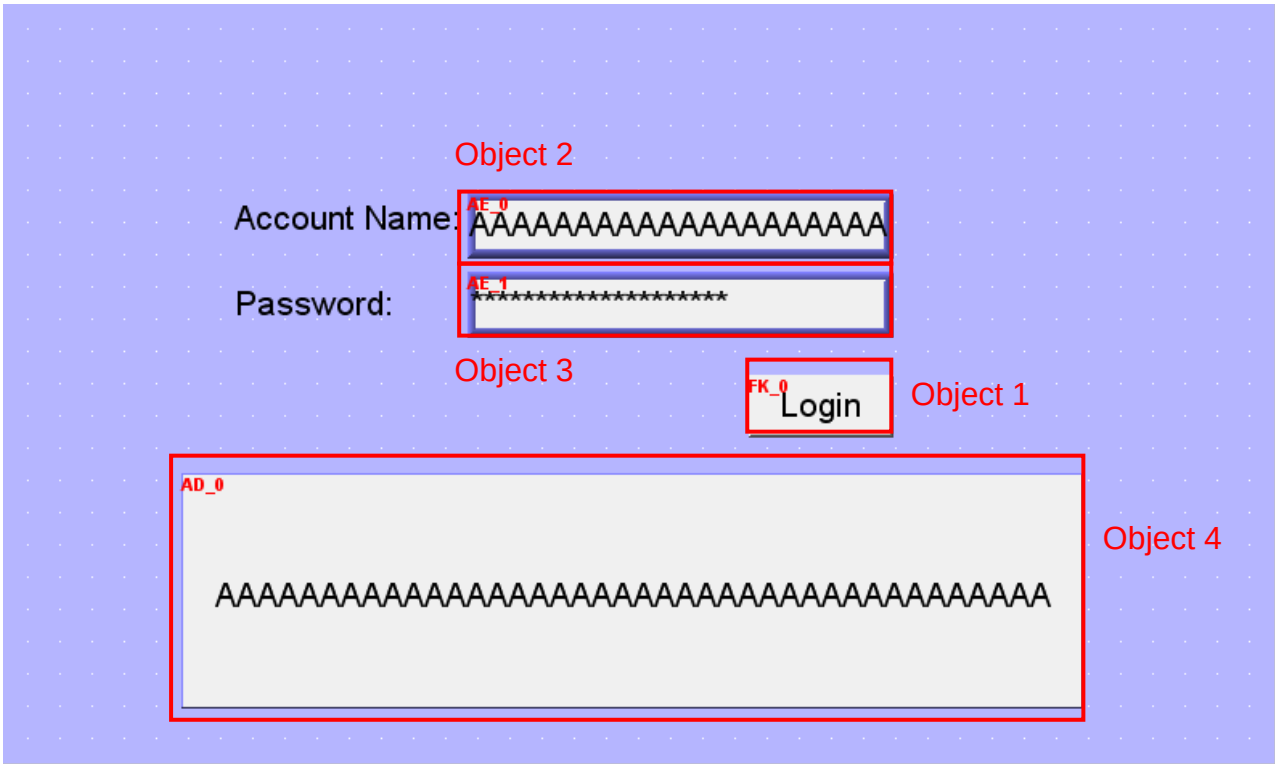
Create a new macro and edit the content:




```
1  macro_command main()
2  char name[20]="admin"
3  char password[20]="123456"
4  char name_input[20]
5  char password_input[20]
6  char message_success[40]="Success! Access Accepted."
7  char message_fail[40]="Fail! Access Denied."
8  char message_clear[40]
9  bool name_match=false
10 bool password_match=false
11
12 StringGet(name_input[0], "Local HMI", LW, 0, 20)
13 StringGet(password_input[0], "Local HMI", LW, 50, 20)
14 name_match = StringCompare(name_input[0], name[0])
15 password_match = StringCompare(password_input[0], password[0])
16
17 FILL(message_clear[0], 0x20, 40)// FILL with white space
18 StringSet(message_clear[0], "Local HMI", LW, 100, 40)
19 if(name_match==true and password_match==true) then
20   StringSet(message_success[0], "Local HMI", LW, 100, 40)
21 else
22   StringSet(message_fail[0], "Local HMI", LW, 100, 40)
23 end if
24 end macro_command
```

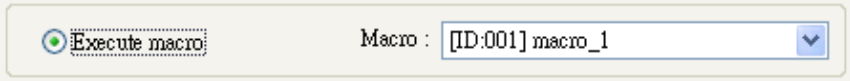
The first two “StringGet” functions will read the strings input by users and store them into arrays named name_input and password_input separately. Use the function “StringCompare” to check if the input account name and password are matched. If the account name is matched, name_match is set true; if the password is matched, password_match is set true. If both name_match and password_match are true, output the string “Success! Access Accepted.”. Otherwise, output the string “Fail! Access Denied.”.

Add ASCII Input and Function Key objects in window 10 of the project. The settings of these objects are shown as below. Function Key object is used to execute macro_1.

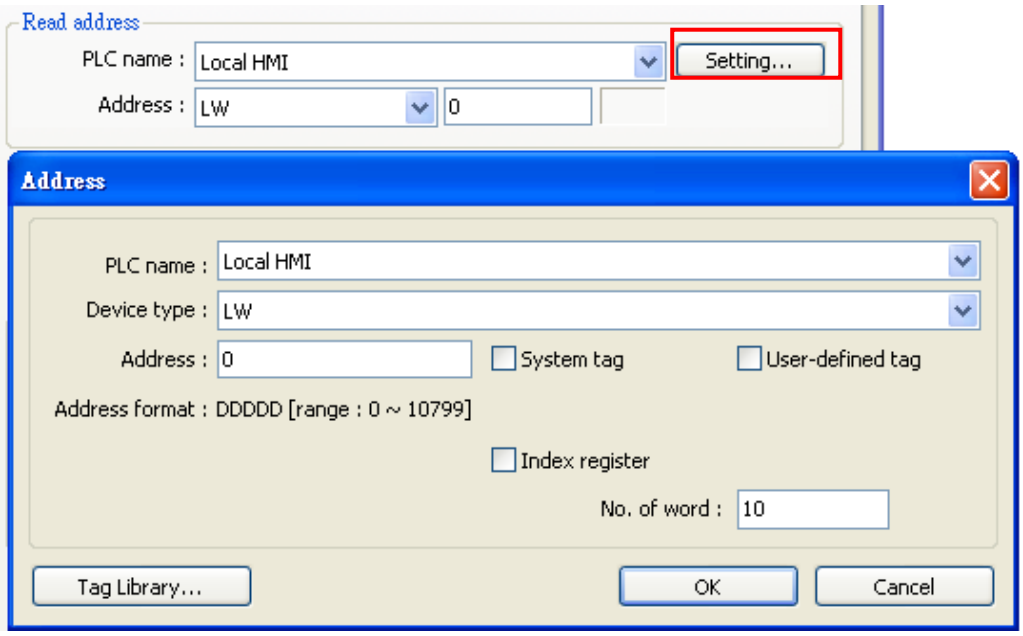


Object settings:

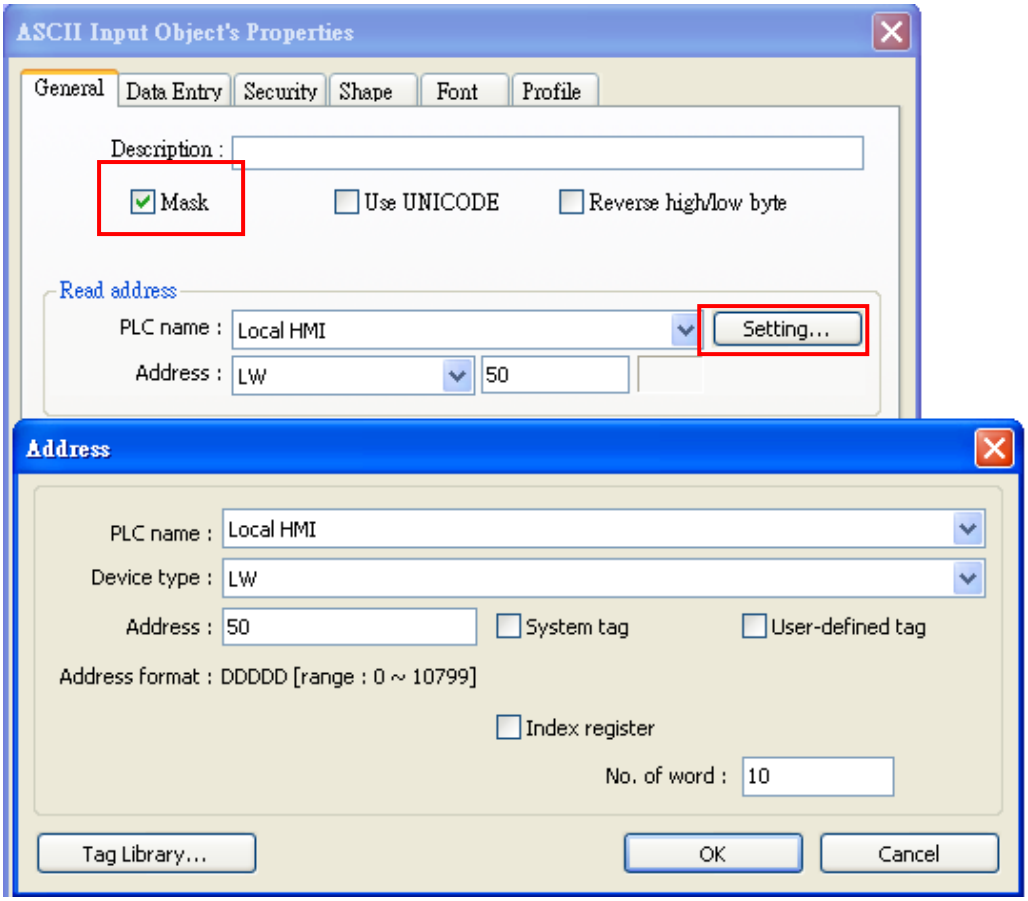
Object 1: Function Key



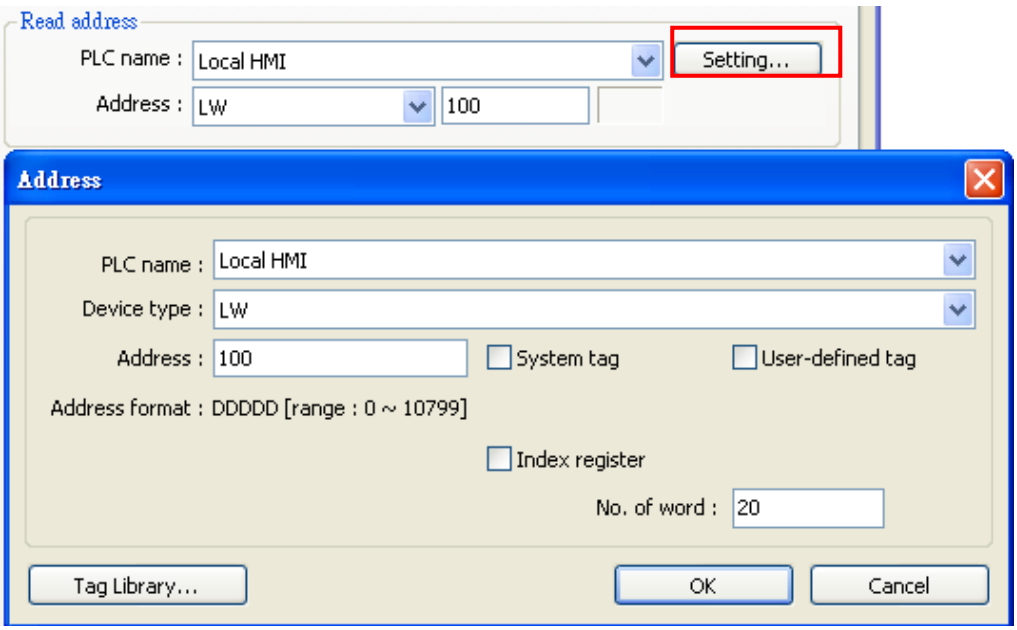
Object 2: ASCII Input



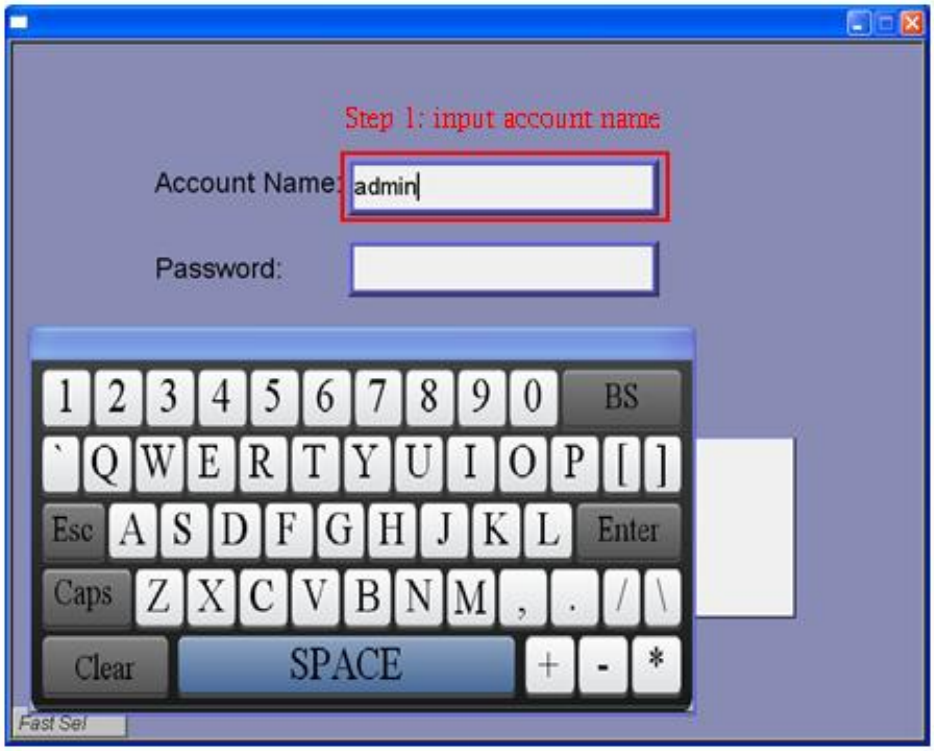
Object 3: ASCII Input

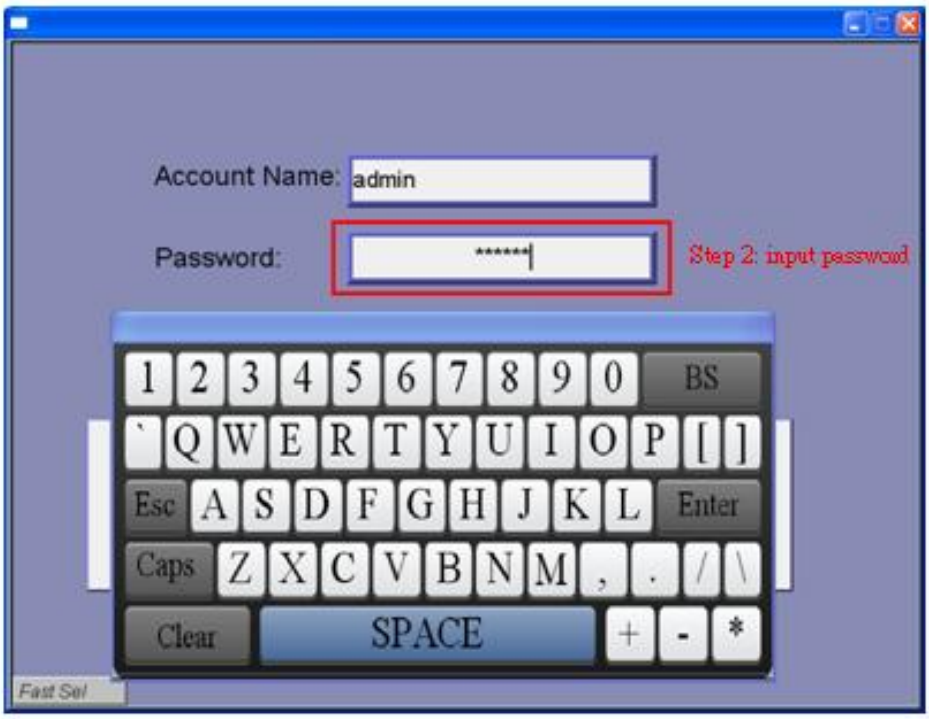


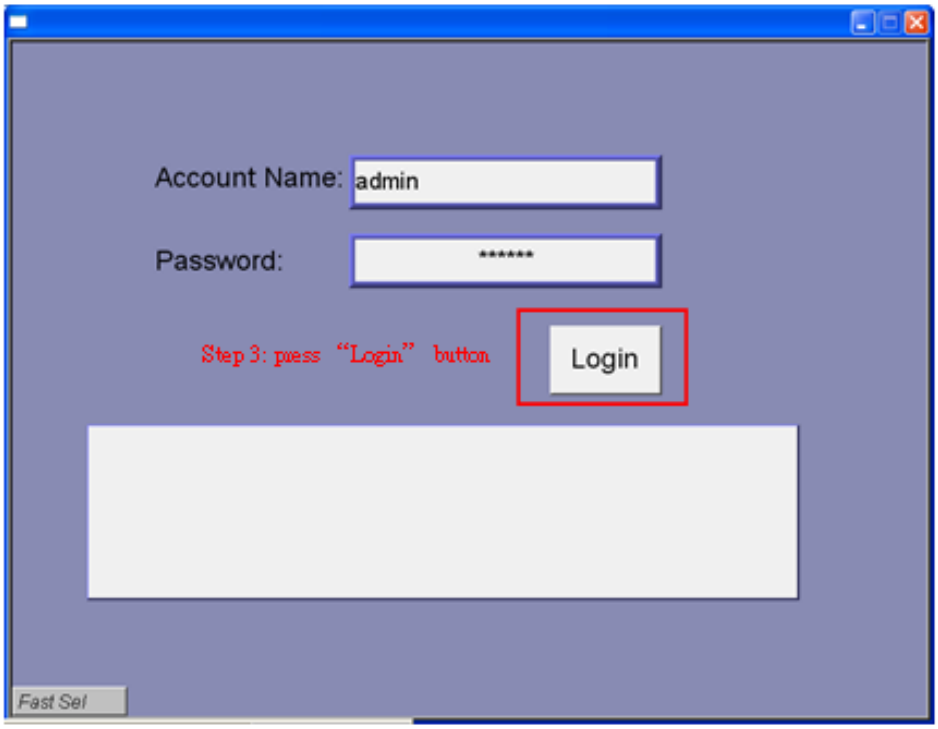
Object 4: ASCII Display



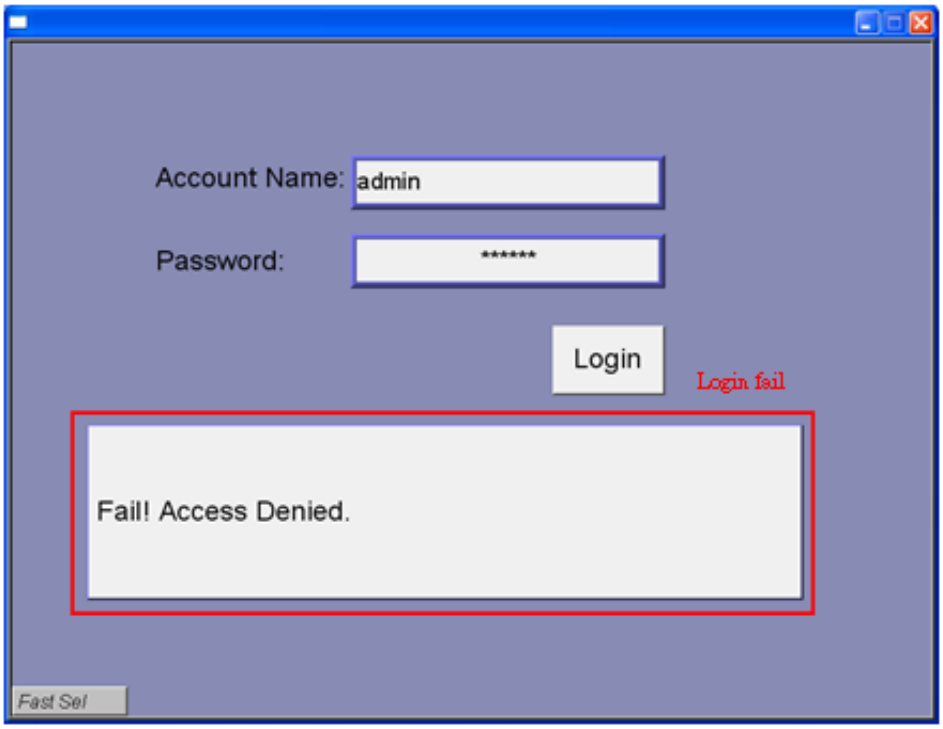
Lastly, use [Compile] to compile the project and execute [Off-line simulation] or [On-line simulation]. Follow the steps below to operate the executing project:



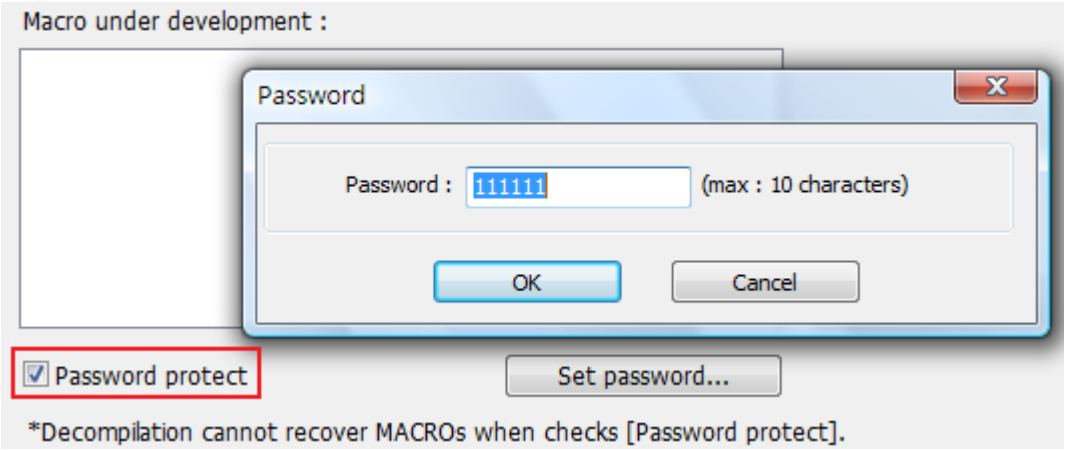

Account Name:
admin
Password:
******
Step 2: input password
1 2 3 4 5 6 7 8 9 0 BS
` Q W E R T Y U I O P [ ]
Esc A S D F G H J K L Enter
Caps Z X C V B N M , . / \
Clear SPACE + - *
Fast Sel


Account Name:
admin
Password:
******
Step 3: press "Login" button
Login
Fast Sel


Account Name:
admin
Password:
******
Login
Login success
Success! Access Accecpted.
Fast Sel

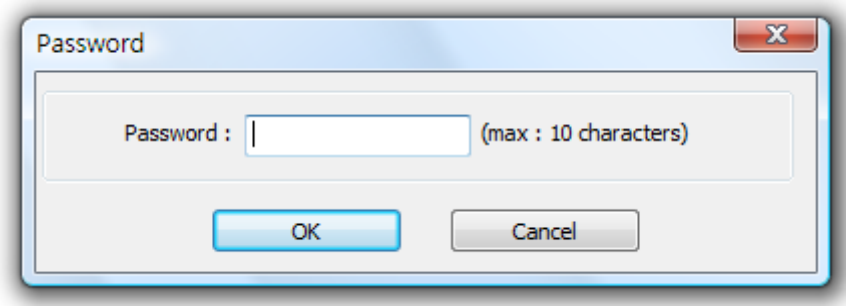

Account Name:
admin
Password:
******
Login
Login fail
Fail! Access Denied.
Fast Sel

18.15 Macro Password Protection

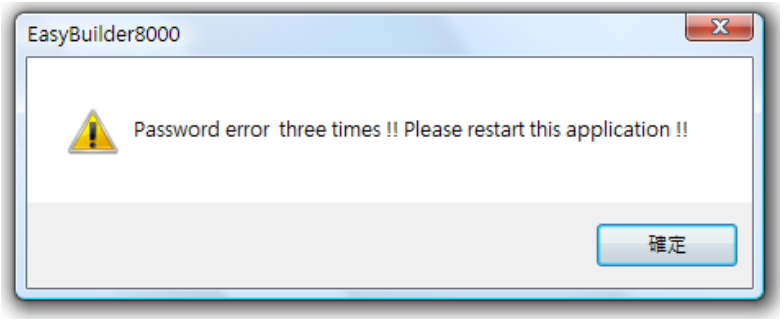


On MACRO editing window there's the [Password protect] selection, tick it and click [Set password…] to set a password less than or equals to 10 characters (support ASCII character only, ex. "a$#*hFds").

After setting MACRO password, users will have to input correct password when opening MACRO editing window.



EasyBuilder should be rebooted for typing the password again after 3 incorrect attempts.

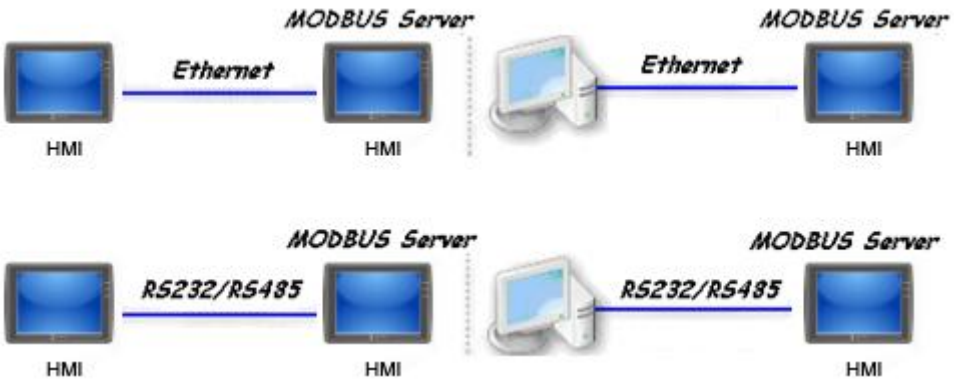


- When MACRO is password protected, de-compilation of XOB file will not be able to restore MACRO contents.

Chapter 19 Configure HMI as a MODBUS Server

19.1 Configure HMI as a MODBUS Device

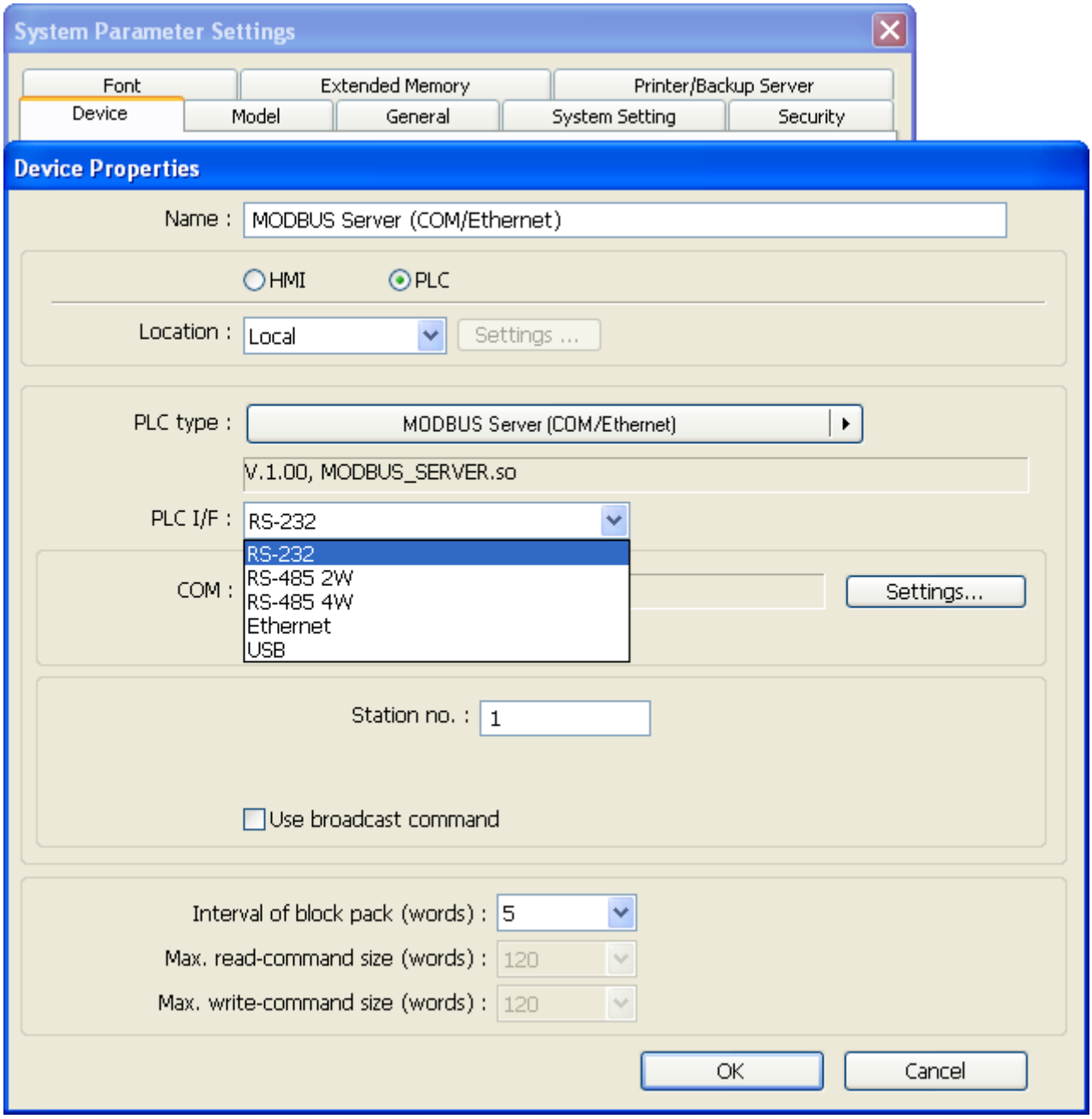
Once HMI is configured as a MODBUS device, the data of HMI can be read or written via MODBUS protocol.



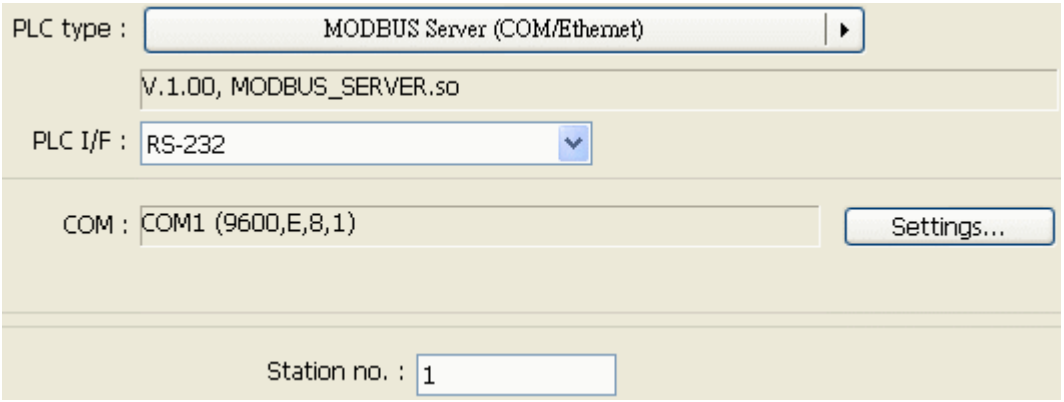
As shown above, HMI is set as a MODBUS device (also called MODBUS Server). The HMI, PC or other devices can use MODBUS protocol to read or write HMI data via Ethernet or RS-232 / RS-485 interface. Please follow the steps below.

19.1.1 Creating a MODBUS Server

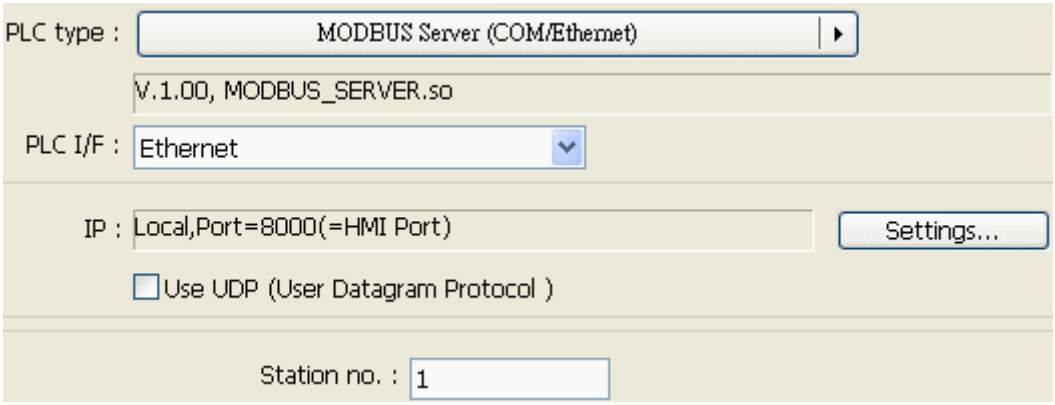
To configure HMI as a MODBUS device, add a new device in the **[Device]** tab first. In **[PLC type]**, select “MODBUS Server”. In **[PLC I/F],** select RS-232, RS-485 2W, RS-485 4W, or Ethernet.



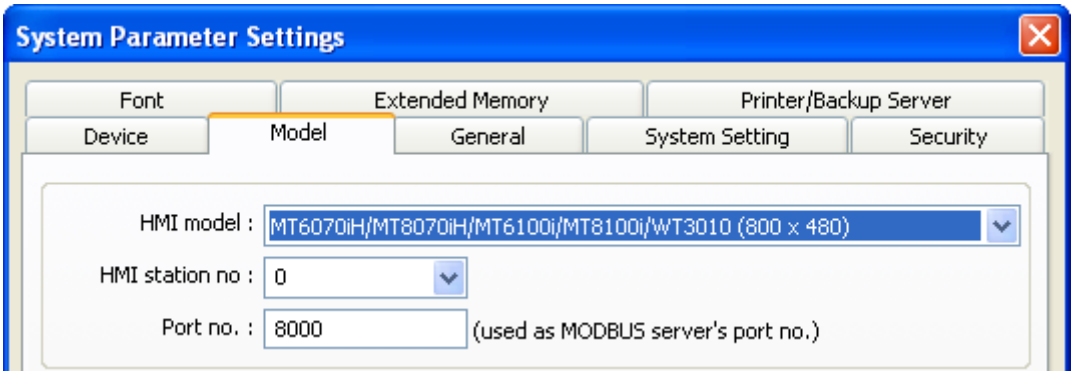
If **[PLC I/F]** is set to **[RS-232]** or **[RS-485]**, please select **[COM]** (COM 1 ~ COM 3) and set correct communication parameters as below. MODBUS Server **[Station no.]** is set to 1.



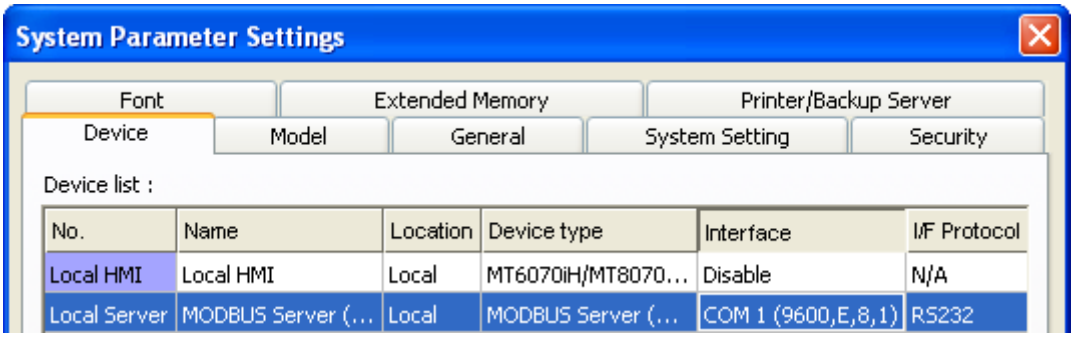
If **[PLC I/F]** is set to **[Ethernet]**, please set **[Port no.]**:



The **[Port no.]** of MODBUS Server and HMI must be the same. To change the port number please set in the **[Model]** tab.

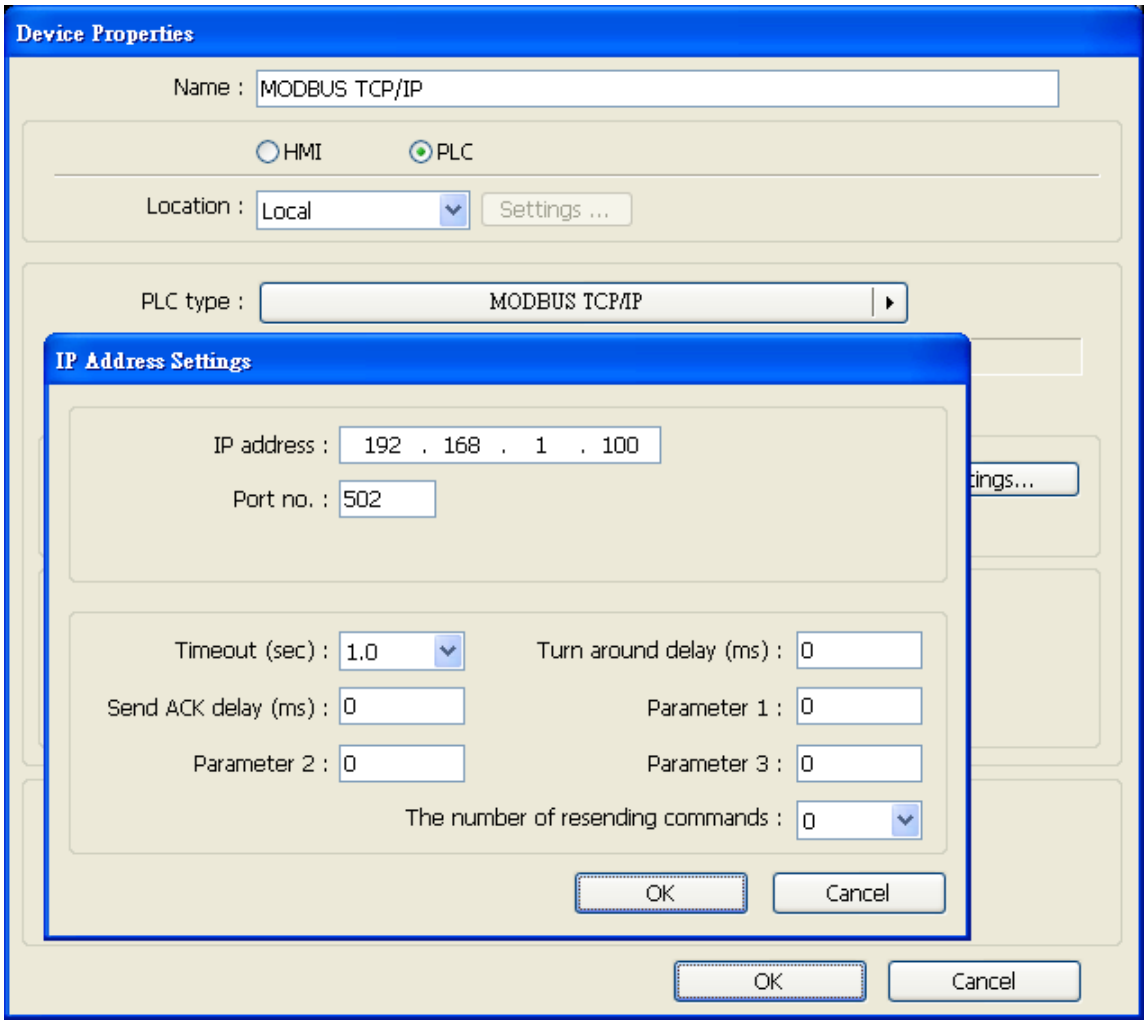


When finished, MODBUS Server is listed in **[Device]** tab. The configuration of MODBUS device is completed. Compile the mtp file and download the compiled xob file to HMI, then, HMI data can be read or written using MODBUS protocol.

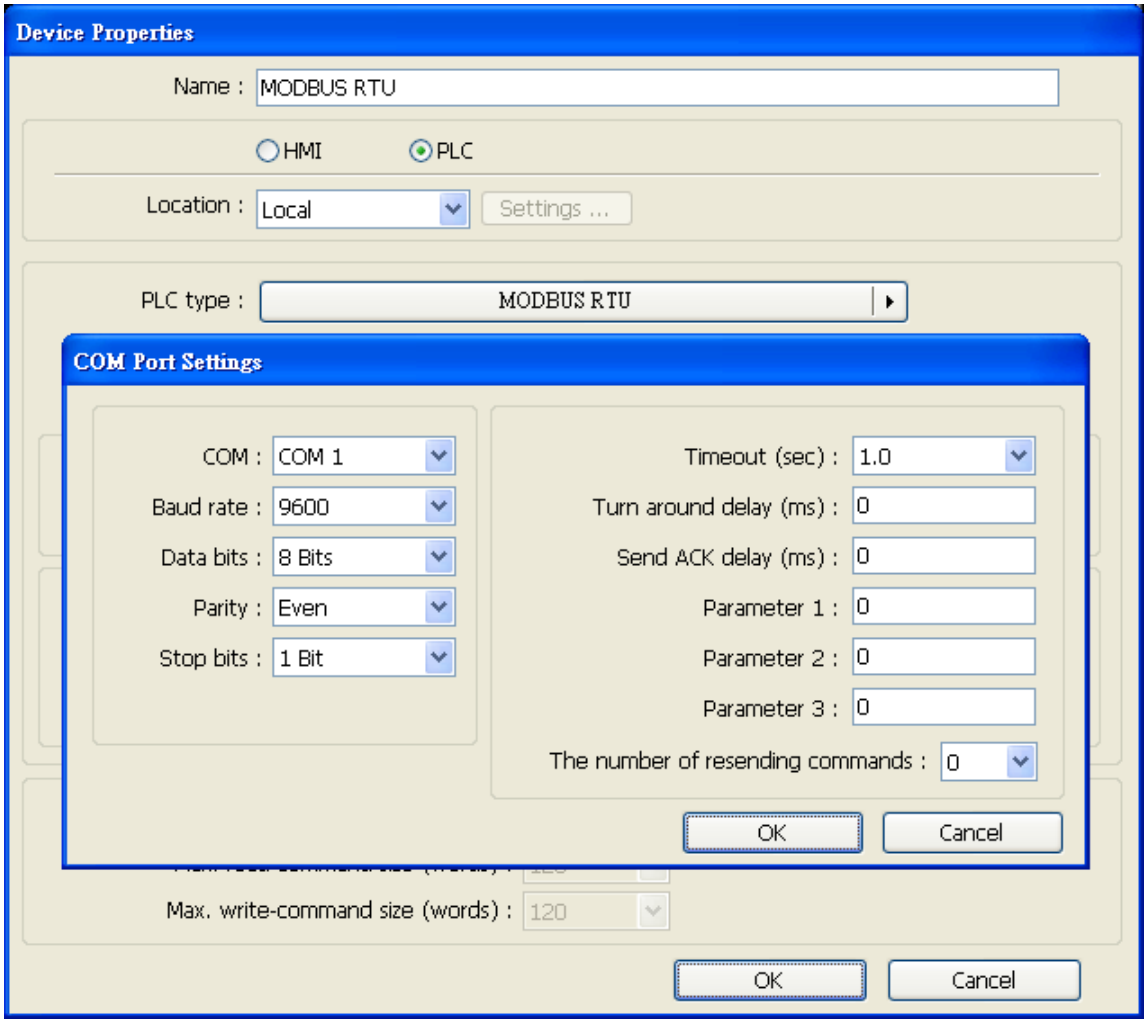


19.1.2 Access a MODBUS Server

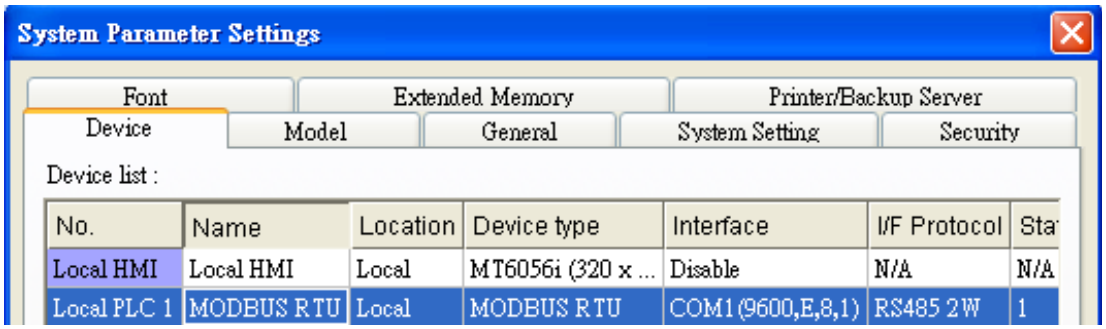
Two HMI can be configured as one MODBUS client and one MODBUS server to communicate and exchange data. First, add a new device in client's device list. If the client uses **[Ethernet]**, set **[PLC type]** to "MODBUS TCP/IP" and fill in the correct **[IP address]** (the IP of MODBUS Server), **[Port no.]**, and **[Station no.]**.



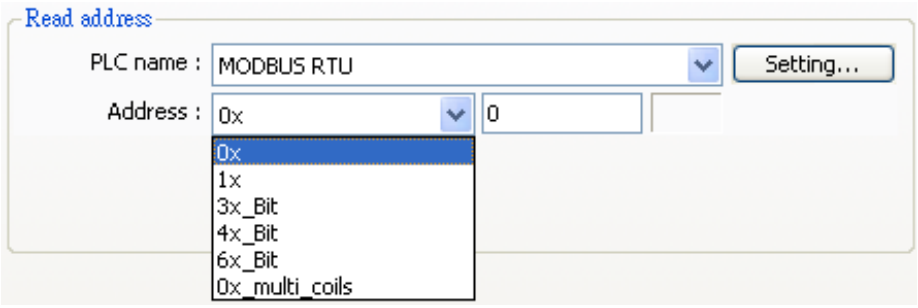
If the client uses **[RS-232]** or **[RS-485]** interface, the **[PLC type]** must set to “MODBUS RTU”, and configure the communication parameters correctly.



When finished, click **[OK]**, then a new device “MODBUS RTU” is listed in the **[Device]** tab.



In the setting page of each object, select “MODBUS RTU” in **[PLC name]**, and set the address of MODBUS RTU.



Since the server is a HMI, the corresponding read and write address are as follows：

read / write 0x/1x (1 ~ 12096)	= read / write LB (0 ~ 12095)
read / write 3x/4x/5x (1 ~ 9999)	= read / write LW (0 ~ 9998)
read / write 3x/4x/5x (10000 ~ 65535)	= read / write RW (0 ~ 55535)

19.2 Changing MODBUS Server Station Number in Runtime

EasyBuilder provides the following system registers to change MODBUS Server station number in runtime.

[LW-9541] MODBUS/ASCII server station no. (COM 1)
[LW-9542] MODBUS/ASCII server station no. (COM 2)
[LW-9543] MODBUS/ASCII server station no. (COM 3)
[LW-9544] MODBUS/ASCII server station no. (Ethernet)

19.3 About MODBUS Address Type

Address types in EasyBuilder MODBUS protocol are 0x, 1x, 3x, 4x, 5x, 6x, 3x_bit and 4x_bit. MODBUS RTU function codes are listed below:

0x: Coils. A read and write device type. When reading a bit with this device type, the function code is 01H. When writing a bit, the function code is 05H. When writing multiple bits, the function code is 0FH.
1x: Discrete Inputs. A read only device type. When reading a bit the function code is 02H.
3x: Input Registers. A read only device type. When reading data, the function code is 04H.
4x: Holding Register. A read and write device type. When reading data, the function code is 03H. When writing data, the function code is 10H.
5x: The function codes are the same as 4x. The difference is that 5x makes double word swap when the format is 32-bit unsigned. If the data read by 4x is 0x1234, the data read by 5x is 0x3412.
6x: A read and write device type. When reading data, the function code is 03H. The difference from 4x is that when writing data, the function code is 06H, meaning to write a single register.
3x_bit: The function code is the same as 3x. The difference is that 3x_bit reads a single bit in the data.
4x_bit: The function code is the same as 4x. The difference is that 4x_bit reads a single bit in the data.

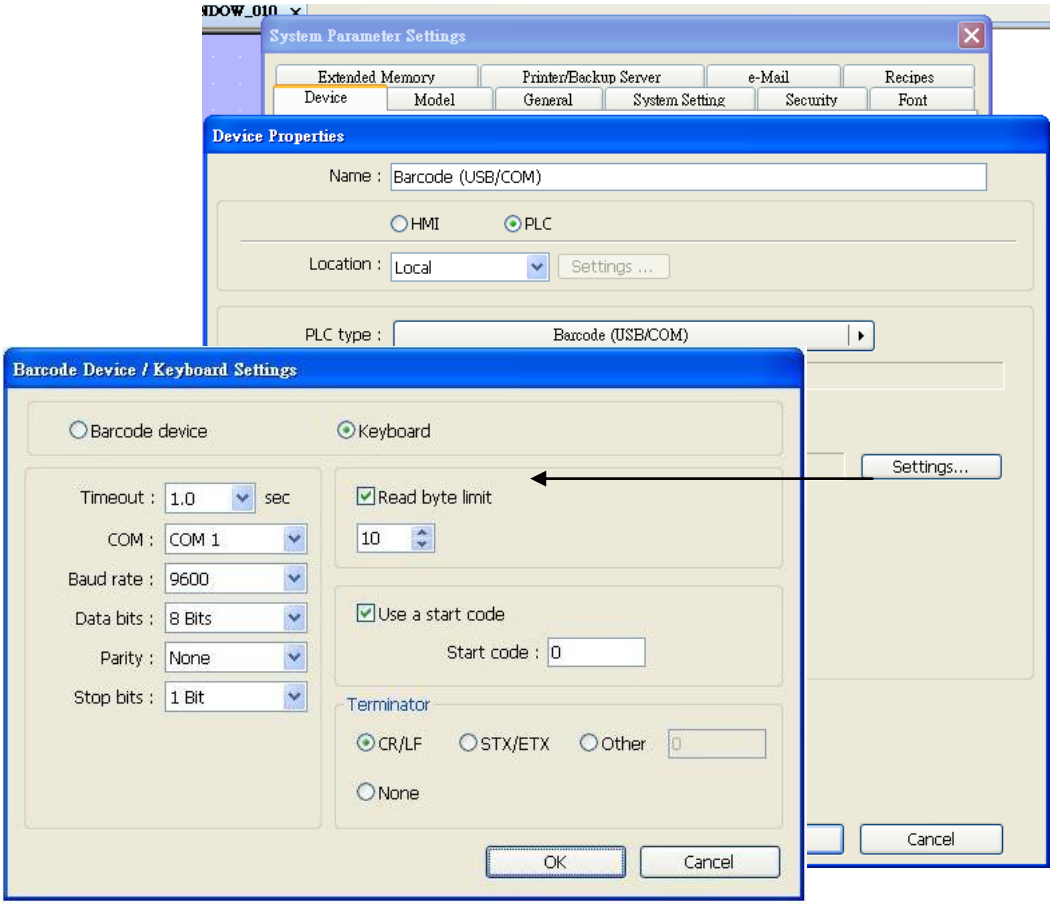
Chapter 20 How to Connect a Barcode Reader

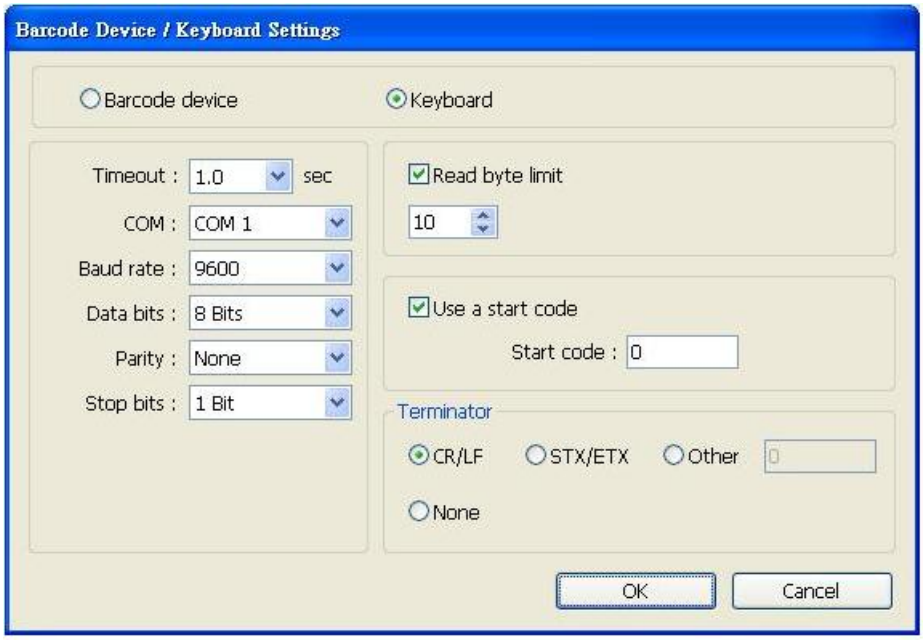
Barcode reader interfaces:

20.1 How to Connect with a Barcode Reader

HMI can connect with USB/COM port barcode reader or keyboard. Please add a new device in **[Edit] » [System Parameter Settings] » [Device list]** as shown below.

Click **[Settings]** and finish **[Barcode Device / Keyboard Settings]**:





[Timeout]

When select [Keyboard], set a time range for keyboard entries. The system starts counting time from the first entry.

[COM], [Baud rate], [Data bits], [Parity], [Stop bits]

Barcode reader can be connected with HMI via USB or COM port. When using COM port, please set the communication parameters correctly. When using USB, no need to set the parameters.

[Read byte limit]

This function will restrict the number of bytes a barcode reader reads in order to prevent overloading. The range is 10 to 512.

For example:

[Read byte limit] is set to “10”. If the data read by barcode device is:

0x34 0x39 0x31 0x32 0x30 0x30 0x34 0x37 0x30 0x38 0x33 0x38. (12 bytes)

Only the first 10 bytes will be read in this case.

0x34 0x39 0x31 0x32 0x30 0x30 0x34 0x37 0x30 0x38

[Use a start code]

If this check box is selected, the data is only valid when the first data identifies with the start code, otherwise the data will be ignored. The start code will not be stored in the address of barcode reader.

For example: if the start code is 255 (0xff), and the data read:

0xff 0x34 0x39 0x31 0x32 0x30 0x30 0x34 0x37

The data saved in the designated barcode reader address will be:

0x34 0x39 0x31 0x32 0x30 0x30 0x34 0x37

[Terminator]

Terminator means the end of data, when terminator is detected; it stands for the end of data stream.

[CR/LF] 0x0a or 0x0d stands for the end of data.

[STX/ETX] 0x02 or 0x03 stands for the end of data.

[Other] Users can set the terminator.

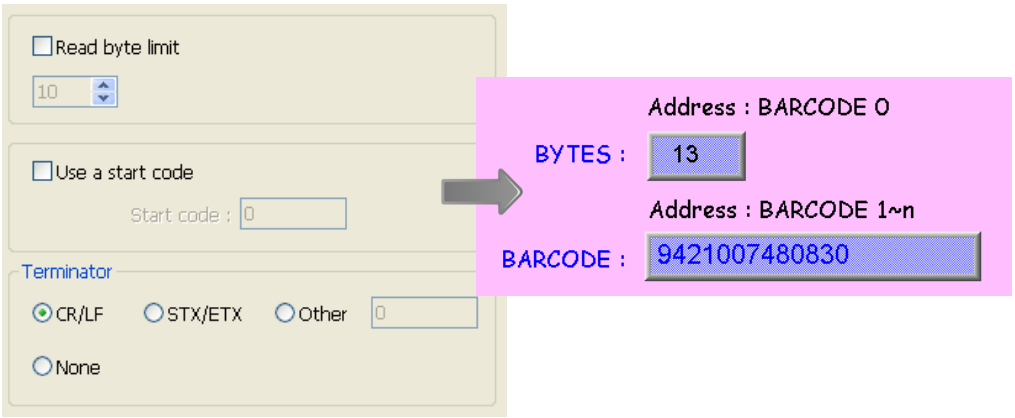
[None] If this option is selected, HMI will save all the data to the designated address of barcode reader.

When finish setting, a new device will be added to the **[Device list]**.

Now the barcode reader can be selected in **[PLC type]** when creating an object. The address types:

Address type	Address name	Description
Bit	FLAG	**FLAG 0** indicates the status of data reading. When finish reading data, the status of FLAG 0 is changed from OFF to ON and will not return OFF automatically.
Word	BARCODE	**BARCODE 0** Number of bytes currently read. **BARCODE 1 ~ n** Store the data read.

The following is a setting example, the barcode is 9421007480830. BARCODE 0 is the address of Numeric Display Object (BYTES) and BARCODE 1 ~ n is the address of ASCII Display object (BARCODE).



In the example the data stored in the barcode reader address:

Barcode Reader Address	Data
BARCODE 0	13 bytes (decimal) The data saved in this address is 14 bytes = 7 words. If the number of bytes is odd, the system will add a byte (0x00) to make it an even number.
BARCODE 1	3439 (HEX)
BARCODE 2	3132 (HEX)
BARCODE 3	3030 (HEX)
BARCODE 4	3437 (HEX)
BARCODE 5	3038 (HEX)
BARCODE 6	3338 (HEX)
BARCODE 7	0030 (HEX)

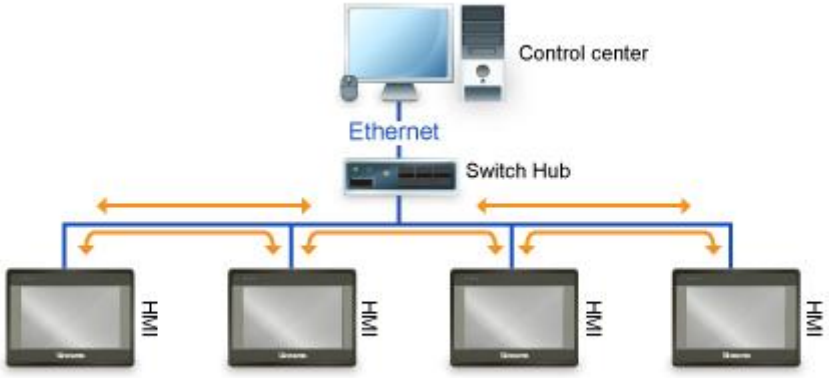
■ HMI can only connect with one USB barcode reader. When the device list in the project includes this kind of device, the system reserved address LB-9064: [enable USB barcode device (disable keyboard) (when ON)] will set ON. To enable USB keyboard again and stop using USB barcode reader, please set LB-9064 OFF.

Please confirm your Internet connection before downloading the demo project.

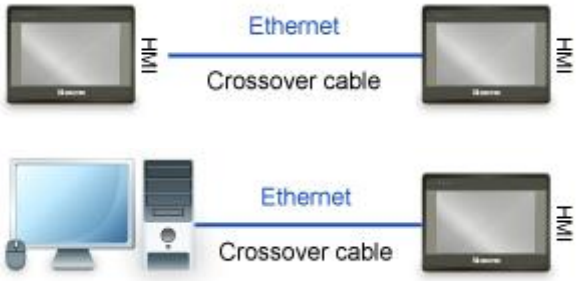
Chapter 21 Ethernet Communication and Multi-HMI Connection

There are two ways of Ethernet communication:

1. Use RJ45 straight through cable + hub.



2. Use RJ45 crossover cable and without hub, but this is limited to the condition of point to point connection (HMI to HMI or PC to HMI).



Through Ethernet network, the system provides the following methods for data transmission:

1. HMI to HMI communication.
2. PC to HMI communication.
3. Operating the PLC connected to another HMI.

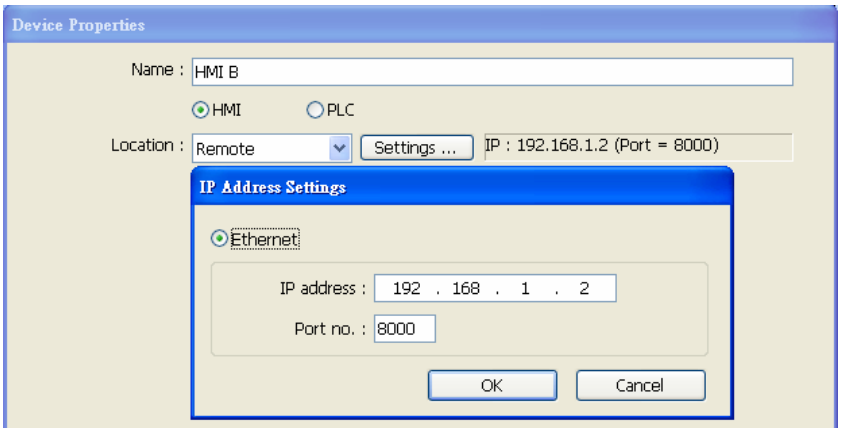
21.1 HMI to HMI Communication

To exchange data between one HMI and another HMI, add a new remote HMI device in [System Parameter Settings].

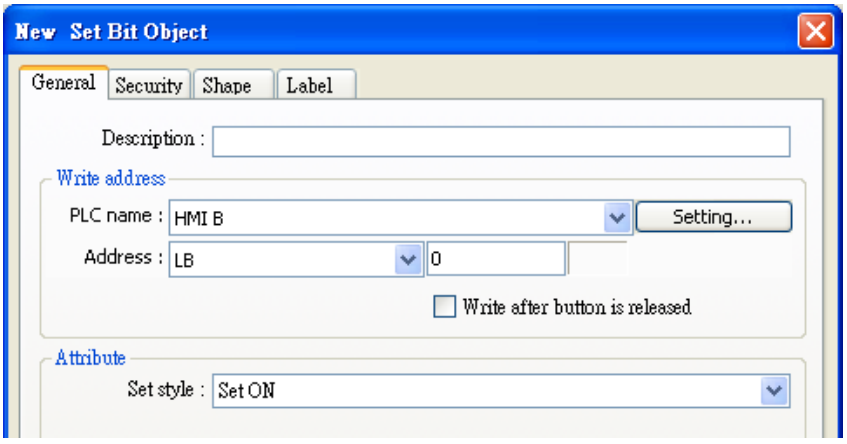
Assume there are HMI A and HMI B, and we want to use a Set Bit object on HMI A to control [LB-0] on HMI B:

1. Set the IP address of the two HMIs, i.e.: HMI A: 192.168.1.1, HMI B: 192.168.1.2.

2. In HMI A project:
[System Parameter Settings] » [Device list]
Add a remote HMI B (IP: 192.168.1.2).



3. In HMI A project:
Create a Set Bit Object, select “HMI B” in **[PLC name]** to control the address of the remote HMI.



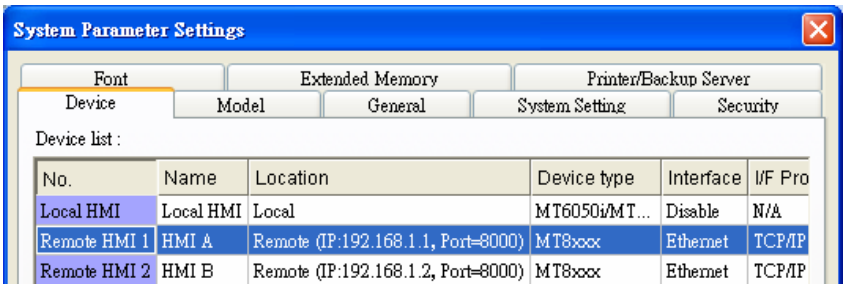
■ One HMI can handle requests from a maximum of 64 HMIs simultaneously.

21.2 PC to HMI Communication

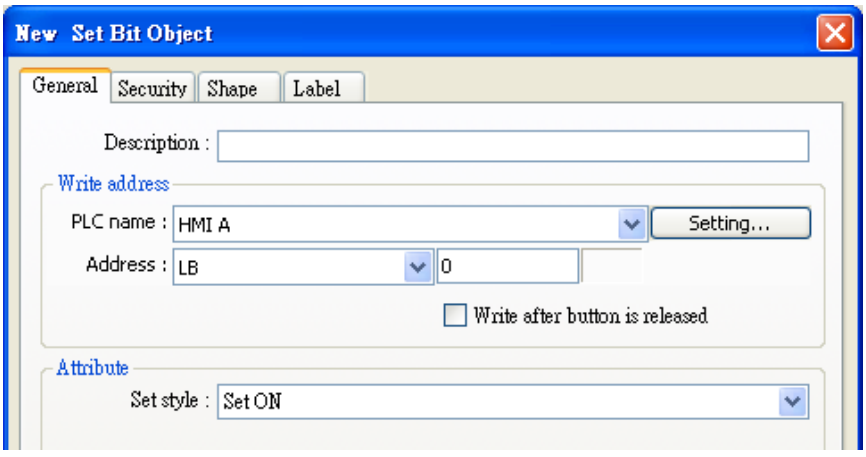
With On-line Simulation, PC can collect data from HMI through Ethernet network and save the data files to PC.
To connect PC with two HMIs (HMI A and HMI B), the settings of the project on PC is shown below:

1. Set the IP address of the two HMIs, i.e.: HMI A: 192.168.1.1, HMI B: 192.168.1.2.

2. In PC project: **[System Parameter Settings] » [Device list]**, Add a remote HMI A (IP: 192.168.1.1) & HMI B (IP: 192.168.1.2).

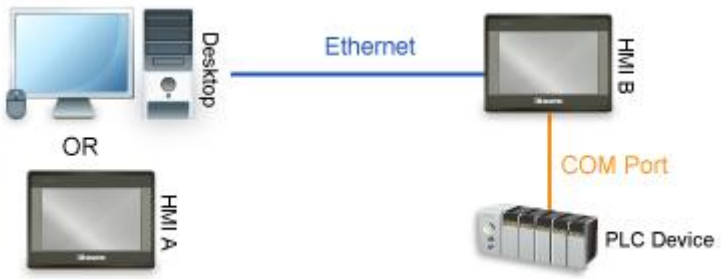


3. In PC project: Create a Set Bit Object, select “HMI A” in **[PLC name]** to control the address of the remote HMI A. Same for the HMI B.



■ A PC can control at most 64 HMIs simultaneously.
■ As shown above, HMI can also control PC. PC can be seen as another HMI, that is, adding a remote HMI in the project files of HMI A / HMI B, and the IP of the remote HMI is set to the IP of PC.

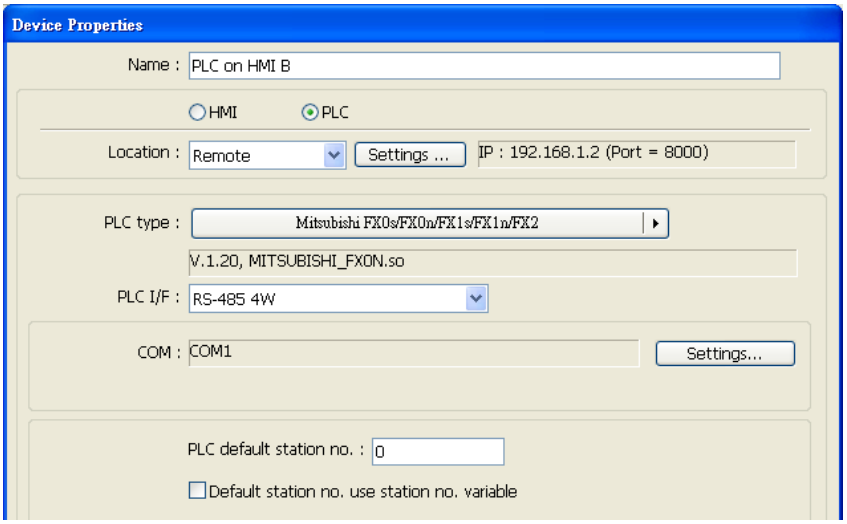
21.3 Operate the PLC Connected with Other HMI



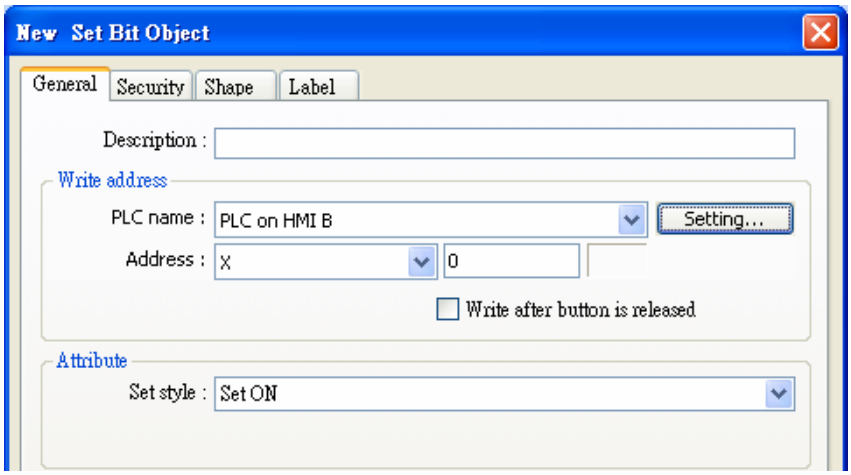
Through Ethernet network, PC or HMI can operate the PLC that is connected to another HMI; as shown above, a PLC is connected to COM 1of HMI B. When using PC or HMI A to read PLC data, the settings of PC or HMI A project is shown below:

1. Set the IP address of HMI B, i.e.: 192.168.1.2.

2. In PC or HMI A project: **[System Parameter Settings] » [Device list]**, Add a remote PLC, and set **[Name]** to “PLC on HMI B”. Set correct parameters. Since this PLC is connected to remote HMI B, set the IP address to HMI B (IP: 192.168.1.2).

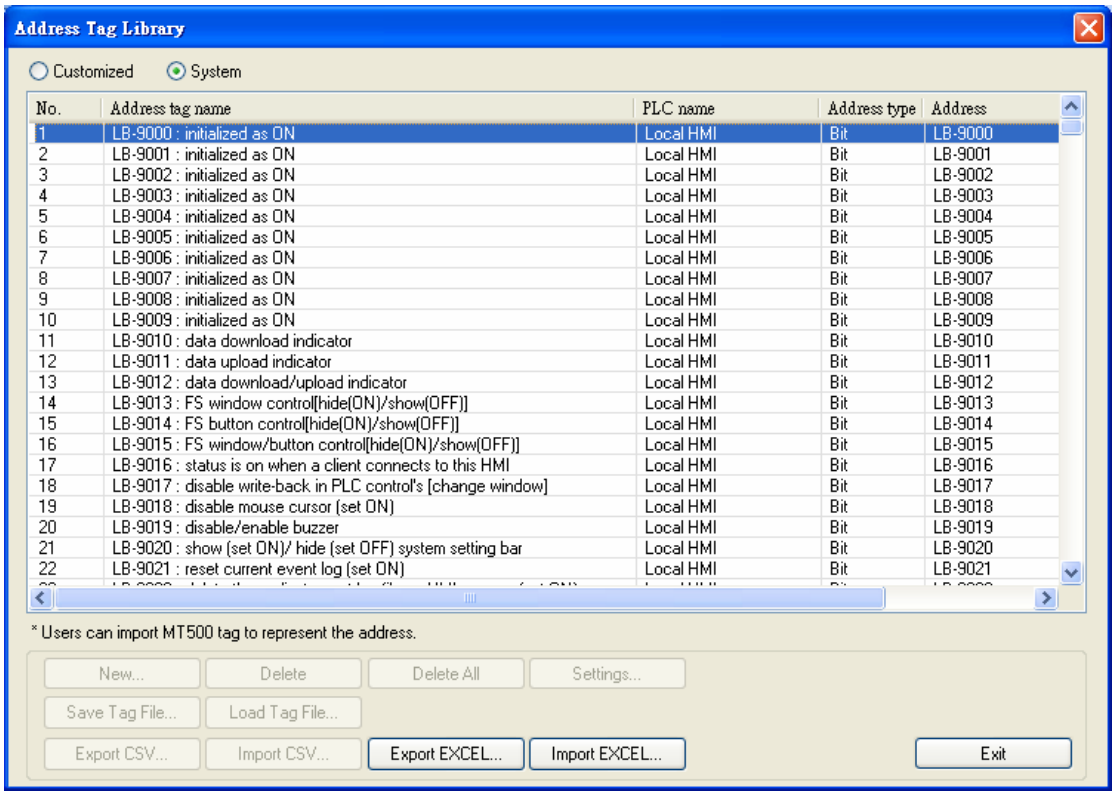


3. In PC or HMI A project: Create a Set Bit Object, select “PLC on HMI B” in **[PLC name]** to control the PLC connected with the remote HMI B.



Chapter 22 System Reserved Words / Bits

Some Local Words and Local Bits are reserved for system usage. These registers are all with different functions described below:



22.1 The Address Ranges of Local HMI Memory

22.1.1 Bits

Memory	Device Type	Range	Format
Local Memory Bits	LB	0 - 12095	DDDDD
Local Word Bits	LW_BIT	0 - 1079915	DDDDDdd DDDDD: address dd: bit no. (00 - 15)
Retentive Memory Bit Index	RBI	0 - 65535f	DDDDDh DDDDD: address h: bit no. (0 - f) Use LW-9000 as Index Register, and correspond to RW_Bit Example: When LW-9000 = 1, RBI-01 = RW_Bit-11
Retentive Memory Word Bits	RW_Bit	0 - 524287f	DDDDDh DDDDD: address h: bit no. (0 - f)
Retentive Memory RW_A Word Bits	RW_A_Bit	0 - 65535f	DDDDDh DDDDD: address h: bit no. (0 - f)

22.1.2 Words

Memory	Device Type	Range	Format
Local Memory Words	LW	0 - 10799	DDDDD
Retentive Memory Words	RW	0 - 524287	DDDDDD
Retentive Memory Word Index	RWI	0 - 65535	DDDDD Use LW-9000 as Index Register, and correspond to RW Example: When LW-9000 = 10, RWI-5 = RW-15
Retentive Memory A Word	RW_A	0 - 65535	DDDDD
Extended Memory Words	EM0 - EM9	0 - 1073741823	DDDDDDDDDD

22.2 HMI Time

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LB-11958	time setting error (when ON) *Note 3	R	R	R
LW-9010	(16bit-BCD) : local second	R/W	R/Y	R/Y
LW-9011	(16bit-BCD) : local minute	R/W	R/Y	R/Y
LW-9012	(16bit-BCD) : local hour	R/W	R/Y	R/Y
LW-9013	(16bit-BCD) : local day	R/W	R/Y	R/Y
LW-9014	(16bit-BCD) : local month	R/W	R/Y	R/Y
LW-9015	(16bit-BCD) : local year	R/W	R/Y	R/Y
LW-9016	(16bit-BCD) : local week	R	R	R
LW-9017	(16bit) : local second	R/W	R/Y	R/Y
LW-9018	(16bit) : local minute	R/W	R/Y	R/Y
LW-9019	(16bit) : local hour	R/W	R/Y	R/Y
LW-9020	(16bit) : local day	R/W	R/Y	R/Y
LW-9021	(16bit) : local month	R/W	R/Y	R/Y
LW-9022	(16bit) : local year *Note 1	R/W	R/Y	R/Y
LW-9023	(16bit) : local week *Note 2	R	R	R
LW-9030	(32bit) : system time (unit : 0.1 second)	R	R	R
LW-9048	(16bit) : time (0 : AM, 1 : PM)	R/W	R/Y	R/Y
LW-9049	(16bit) : local hour (12-hour format)	R/W	R/Y	R/Y

1. Value range: 2000 - 2037
2. Value range: 1 - 7, meaning Monday – Sunday.
3. When use LW-9010 to LW-9023 to update RTC time, the system will check if RTC time is successfully updated. If the system still fails to update RTC time, the system register [LB-11958: time setting error] will be set ON, and restore to the time before update. Updating time on PC during simulation by using LW-9010 to LW-9023 does not work.

22.3 User Name and Password

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LB-9050	user logout	W	Y	Y
LB-9060	password error	R	R	R
LB-9061	update password (set ON)	W	Y	Y
LW-9082	(16bit) : auto logout time (unit : minute, 0 : disable the function)	R/W	R/Y	R/Y
LW-9219	(16bit): user no. (1-12)	R/W	R/Y	R/Y
LW-9220	(32bit) : password	R/W	R/Y	R/Y
LW-9222	(16bit) : classes can be operated for current user (bit 0:A, bit 1:B,bit 2:C, ...)	R	R	R
LW-9500	(32bit) : user 1's password	R/W	R/Y	R/Y
LW-9502	(32bit) : user 2's password	R/W	R/Y	R/Y
LW-9504	(32bit) : user 3's password	R/W	R/Y	R/Y
LW-9506	(32bit) : user 4's password	R/W	R/Y	R/Y
LW-9508	(32bit) : user 5's password	R/W	R/Y	R/Y
LW-9510	(32bit) : user 6's password	R/W	R/Y	R/Y
LW-9512	(32bit) : user 7's password	R/W	R/Y	R/Y
LW-9514	(32bit) : user 8's password	R/W	R/Y	R/Y
LW-9516	(32bit) : user 9's password	R/W	R/Y	R/Y
LW-9518	(32bit) : user 10's password	R/W	R/Y	R/Y
LW-9520	(32bit) : user 11's password	R/W	R/Y	R/Y
LW-9522	(32bit) : user 12's password	R/W	R/Y	R/Y

Please confirm your Internet connection before downloading the demo project.

22.4 Data Sampling

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LB-9025	delete the earliest data sampling file on HMI memory (set ON)	W	Y	Y
LB-9026	delete all data sampling files on HMI memory (set ON)	W	Y	Y
LB-9027	refresh data sampling information on HMI memory (set ON)	W	Y	Y
LB-9034	save event/data sampling to HMI, USB disk, SD card (set ON)	W	Y	Y
LB-11949	delete the earliest data sampling file on SD card (set ON)	W	Y	Y
LB-11950	delete all data sampling files on SD card (set ON)	W	Y	Y
LB-11951	refresh data sampling information on SD card (set ON)	W	Y	Y
LB-11952	delete the earliest data sampling file on USB 1 (set ON)	W	Y	Y
LB-11953	delete all data sampling files on USB 1 (set ON)	W	Y	Y
LB-11954	refresh data sampling information on USB 1 (set ON)	W	Y	Y
LB-11955	delete the earliest data sampling file on USB 2 (set ON)	W	Y	Y
LB-11956	delete all data sampling files on USB 2 (set ON)	W	Y	Y
LB-11957	refresh data sampling information on USB 2 (set ON)	W	Y	Y
LW-9063	(16bit) : no. of data sampling files on HMI memory	R	R	R
LW-9064	(32bit) : size of data sampling files on HMI memory	R	R	R
LW-10489	(16bit) : no. of data sampling files on SD card	R	R	R
LW-10490	(32bit) : size of data sampling files on SD card	R	R	R
LW-10492	(16bit) : no. of data sampling files on USB 1	R	R	R
LW-10493	(32bit) : size of data sampling files on USB 1	R	R	R
LW-10495	(16bit) : no. of data sampling files on USB 2	R	R	R
LW-10496	(32bit) : size of data sampling files on USB 2	R	R	R

1. The registers for deleting or updating data samplings do not work during simulation on PC.

22.5 Event Log

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LB-9021	reset current event log (set ON)	W	Y	Y
LB-9022	delete the earliest event log file on HMI memory (set ON)	W	Y	Y
LB-9023	delete all event log files on HMI memory (set ON)	W	Y	Y
LB-9024	refresh event log information on HMI memory (set ON)	W	Y	Y
LB-9034	save event/data sampling to HMI, USB disk, SD card (set ON)	W	Y	Y
LB-9042	acknowledge all alarm events (set ON)	W	Y	Y
LB-9043	unacknowledged events exist (when ON)	R	R	R
LB-11940	delete the earliest event log file on SD card (set ON)	W	Y	Y
LB-11941	delete all event log files on SD card (set ON)	W	Y	Y
LB-11942	refresh event log information on SD card (set ON)	W	Y	Y
LB-11943	delete the earliest event log file on USB 1 (set ON)	W	Y	Y
LB-11944	delete all event log files on USB 1 (set ON)	W	Y	Y
LB-11945	refresh event log information on USB 1 (set ON)	W	Y	Y
LB-11946	delete the earliest event log file on USB 2 (set ON)	W	Y	Y
LB-11947	delete all event log files on USB 2 (set ON)ON)	W	Y	Y
LB-11948	refresh event log information on USB 2 (set ON)	W	Y	Y
LW-9060	(16bit) : no. of event log files on HMI memory	R	R	R
LW-9061	(32bit) : size of event log files on HMI memory	R	R	R
LW-9450	(16bit) : time tag of event log – second *Note1	R/W	R/Y	R/Y
LW-9451	(16bit) : time tag of event log – minute *Note1	R/W	R/Y	R/Y
LW-9452	(16bit) : time tag of event log – hour *Note1	R/W	R/Y	R/Y
LW-9453	(16bit) : time tag of event log – day *Note1	R/W	R/Y	R/Y
LW-9454	(16bit) : time tag of event log – month *Note1	R/W	R/Y	R/Y
LW-9455	(16bit) : time tag of event log – year *Note1	R/W	R/Y	R/Y
LW-10480	(16bit) : no. of event log files on SD card	R	R	R
LW-10481	(32bit) : size of event log files on SD card	R	R	R
LW-10483	(16bit) : no. of event log files on USB 1	R	R	R
LW-10484	(32bit) : size of event log files on USB 1	R	R	R
LW-10486	(16bit) : no. of event log files on USB 2	R	R	R
LW-10487	(32bit) : size of event log files on USB 2	R	R	R

1. If LW-9450 - LW-9455 are used as tags of Event Log time source, please set [System parameters] \ [General] correctly.
2. The registers for deleting or updating event logs do not work during simulation on PC.

The following link refers to a demo project of using the system registers LW-9450 to LW-9455 to be the time tag of event log.

Please confirm your Internet connection before downloading the demo project.

22.6 HMI Hardware Operation

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LB-9018	disable mouse cursor (set ON)	R/W	R/Y	R/Y
LB-9019	disable/enable buzzer	R/W	R/Y	R/Y
LB-9020	show (set ON)/ hide (set OFF) system setting bar	R/W	R/Y	R/Y
LB-9033	disable(when on)/enable (when off) HMI upload function(i series only) *Note1	R/W	R/Y	R
LB-9040	backlight up (set ON) *Note2	W	Y	Y
LB-9041	backlight down (set ON) *Note2	W	Y	Y
LB-9047	reboot HMI (set ON when LB9048 is on)	W	Y	Y
LB-9048	reboot-HMI protection	R/W	R/Y	R/Y
LB-9062	open hardware setting dialog (set ON)	W	Y	Y
LB-9063	disable(set ON)/enable(set OFF) popuping information dialog while an USB disk is plugged (i series support only)	R/W	R/Y	R/Y
LW-9008	(32bit-float) : battery voltage (i series supports only) *Note3	R	R	R
LW-9025	(16bit) : CPU loading (x 100%)	R	R	R
LW-9026	(16bit) : OS version (year)	R	R	R
LW-9027	(16bit) : OS version (month)	R	R	R
LW-9028	(16bit) : OS version (day)	R	R	R
LW-9040	(16bit) : backlight index *Note2	R	R	R
LW-9080	(16bit) : backlight saver time (unit : minute)	R/W	R/Y	R/Y
LW-9081	(16bit) : screen saver time (unit : minute)	R/W	R/Y	R/Y

Note

1. After changing the settings, please reboot HMI for updating.
2. LW-9040 used together with LB-9040 - LB-9041 can adjust the backlight brightness with level 0 - 31.
3. For LW-9008, when the battery voltage level goes below 2.80V, it is recommended to replace the battery.

22.7 Local HMI Network Information

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LW-9125	(16bit) : HMI ethernet gateway 0 (machine used only)	R/W	R/Y	R/Y
LW-9126	(16bit) : HMI ethernet gateway 1 (machine used only)	R/W	R/Y	R/Y
LW-9127	(16bit) : HMI ethernet gateway 2 (machine used only)	R/W	R/Y	R/Y
LW-9128	(16bit) : HMI ethernet gateway 3 (machine used only)	R/W	R/Y	R/Y
LW-9129	(16bit) : HMI ethernet IP 0 (machine used only)	R/W	R/Y	R/Y
LW-9130	(16bit) : HMI ethernet IP 1 (machine used only)	R/W	R/Y	R/Y
LW-9131	(16bit) : HMI ethernet IP 2 (machine used only)	R/W	R/Y	R/Y
LW-9132	(16bit) : HMI ethernet IP 3 (machine used only)	R/W	R/Y	R/Y
LW-9133	(16bit) : ethernet port no.	R	R	R
LW-9135	(16bit) : media access control (MAC) address 0	R	R	R
LW-9136	(16bit) : media access control (MAC) address 1	R	R	R
LW-9137	(16bit) : media access control (MAC) address 2	R	R	R
LW-9138	(16bit) : media access control (MAC) address 3	R	R	R
LW-9139	(16bit) : media access control (MAC) address 4	R	R	R
LW-9140	(16bit) : media access control (MAC) address 5	R	R	R
LW-10750	(16bit) : HMI ethernet Mask 0 (machine used only)	R/W	R/Y	R/Y
LW-10751	(16bit) : HMI ethernet Mask 0 (machine used only)	R/W	R/Y	R/Y
LW-10752	(16bit) : HMI ethernet Mask 0 (machine used only)	R/W	R/Y	R/Y
LW-10753	(16bit) : HMI ethernet Mask 0 (machine used only)	R/W	R/Y	R/Y

22.8 Recipe and Extended Memory

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LB-9028	reset all recipe data (set ON)	W	Y	Y
LB-9029	save all recipe data to machine (set ON)	W	Y	Y
LB-9460	EM0's storage device (SD card) does not exist (when ON)	R	R	R
LB-9461	EM1's storage device (SD card) does not exist (when ON)	R	R	R
LB-9462	EM2's storage device (SD card) does not exist (when ON)	R	R	R
LB-9463	EM3's storage device (SD card) does not exist (when ON)	R	R	R
LB-9464	EM4's storage device (SD card) does not exist (when ON)	R	R	R
LB-9465	EM5's storage device (SD card) does not exist (when ON)	R	R	R
LB-9466	EM6's storage device (SD card) does not exist (when ON)	R	R	R
LB-9467	EM7's storage device (SD card) does not exist (when ON)	R	R	R
LB-9468	EM8's storage device (SD card) does not exist (when ON)	R	R	R
LB-9469	EM9's storage device (SD card) does not exist (when ON)	R	R	R
LB-9470	EM0's storage device (USB1 disk) does not exist (when ON)	R	R	R
LB-9471	EM1's storage device (USB1 disk) does not exist (when ON)	R	R	R
LB-9472	EM2's storage device (USB1 disk) does not exist (when ON)	R	R	R
LB-9473	EM3's storage device (USB1 disk) does not exist (when ON)	R	R	R
LB-9474	EM4's storage device (USB1 disk) does not exist (when ON)	R	R	R
LB-9475	EM5's storage device (USB1 disk) does not exist (when ON)	R	R	R
LB-9476	EM6's storage device (USB1 disk) does not exist (when ON)	R	R	R
LB-9477	EM7's storage device (USB1 disk) does not exist (when ON)	R	R	R
LB-9478	EM8's storage device (USB1 disk) does not exist (when ON)	R	R	R
LB-9479	EM9's storage device (USB1 disk) does not exist (when ON)	R	R	R
LB-9480	EM0's storage device (USB2 disk) does not exist (when ON)	R	R	R
LB-9481	EM1's storage device (USB2 disk) does not exist (when ON)	R	R	R
LB-9482	EM2's storage device (USB2 disk) does not exist (when ON)	R	R	R
LB-9483	EM3's storage device (USB2 disk) does not exist (when ON)	R	R	R
LB-9484	EM4's storage device (USB2 disk) does not exist (when ON)	R	R	R
LB-9485	EM5's storage device (USB2 disk) does not exist (when ON)	R	R	R
LB-9486	EM6's storage device (USB2 disk) does not exist (when ON)	R	R	R
LB-9487	EM7's storage device (USB2 disk) does not exist (when ON)	R	R	R
LB-9488	EM8's storage device (USB2 disk) does not exist (when ON)	R	R	R
LB-9489	EM9's storage device (USB2 disk) does not exist (when ON)	R	R	R

22.9 Storage Space Management

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LB-9035	HMI free space insufficiency alarm (when ON)	R	R	R
LB-9036	SD card free space insufficiency alarm (when ON)	R	R	R
LB-9037	USB 1 free space insufficiency alarm (when ON)	R	R	R
LB-9038	USB 2 free space insufficiency alarm (when ON)	R	R	R
LB-12048	USB 1 status (exists when ON)	R	R	R
LB-12049	USB 2 status (exists when ON)	R	R	R
LW-9070	(16bit) : free space insufficiency warning (Mega bytes)	R	R	R
LW-9071	(16bit) : reserved free space size (Mega bytes)	R	R	R
LW-9072	(32bit) : HMI current free space (K bytes)	R	R	R
LW-9074	(32bit) : SD current free space (K bytes)	R	R	R
LW-9076	(32bit) : USB 1 current free space (K bytes)	R	R	R
LW-9078	(32bit) : USB 2 current free space (K bytes)	R	R	R

Want to know how to use LW-9072 - LW-9078 together with Backup object?

Please confirm your Internet connection before downloading the demo project.

22.10 Touch Position

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LW-9041	(16bit) : touch status word(bit 0 on = user is touching the screen)	R	R	R
LW-9042	(16bit) : touch x position	R	R	R
LW-9043	(16bit) : touch y position	R	R	R
LW-9044	(16bit) : leave x position	R	R	R
LW-9045	(16bit) : leave y position	R	R	R

Want to know how to trigger relevant registers to change window with finger slide?

Please confirm your Internet connection before downloading the demo project.

22.11 Station Number Variables

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LW-10000	(16bit) : var0 - station no variable (usage : var0#address)	R/W	R/Y	R/Y
LW-10001	(16bit) : var1 - station no variable (usage : var1#address)	R/W	R/Y	R/Y
LW-10002	(16bit) : var2 - station no variable (usage : var2#address)	R/W	R/Y	R/Y
LW-10003	(16bit) : var3 - station no variable (usage : var3#address)	R/W	R/Y	R/Y
LW-10004	(16bit) : var4 - station no variable (usage : var4#address)	R/W	R/Y	R/Y
LW-10005	(16bit) : var5 - station no variable (usage : var5#address)	R/W	R/Y	R/Y
LW-10006	(16bit) : var6 - station no variable (usage : var6#address)	R/W	R/Y	R/Y
LW-10007	(16bit) : var7 - station no variable (usage : var7#address)	R/W	R/Y	R/Y
LW-10008	(16bit) : var8 - station no variable (usage : var8#address)	R/W	R/Y	R/Y
LW-10009	(16bit) : var9 - station no variable (usage : var9#address)	R/W	R/Y	R/Y
LW-10010	(16bit) : var10 - station no variable (usage : var10#address)	R/W	R/Y	R/Y
LW-10011	(16bit) : var11 - station no variable (usage : var11#address)	R/W	R/Y	R/Y
LW-10012	(16bit) : var12 - station no variable (usage : var12#address)	R/W	R/Y	R/Y
LW-10013	(16bit) : var13 - station no variable (usage : var13#address)	R/W	R/Y	R/Y
LW-10014	(16bit) : var14 - station no variable (usage : var14#address)	R/W	R/Y	R/Y
LW-10015	(16bit) : var15 - station no variable (usage : var15#address)	R/W	R/Y	R/Y

Please confirm your Internet connection before downloading the demo project.

22.12 Index Register

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LW-9200	(16bit) : address index 0	R/W	R/Y	R/Y
LW-9201	(16bit) : address index 1	R/W	R/Y	R/Y
LW-9202	(16bit) : address index 2	R/W	R/Y	R/Y
LW-9203	(16bit) : address index 3	R/W	R/Y	R/Y
LW-9204	(16bit) : address index 4	R/W	R/Y	R/Y
LW-9205	(16bit) : address index 5	R/W	R/Y	R/Y
LW-9206	(16bit) : address index 6	R/W	R/Y	R/Y
LW-9207	(16bit) : address index 7	R/W	R/Y	R/Y
LW-9208	(16bit) : address index 8	R/W	R/Y	R/Y
LW-9209	(16bit) : address index 9	R/W	R/Y	R/Y
LW-9210	(16bit) : address index 10	R/W	R/Y	R/Y
LW-9211	(16bit) : address index 11	R/W	R/Y	R/Y
LW-9212	(16bit) : address index 12	R/W	R/Y	R/Y
LW-9213	(16bit) : address index 13	R/W	R/Y	R/Y
LW-9214	(16bit) : address index 14	R/W	R/Y	R/Y
LW-9215	(16bit) : address index 15	R/W	R/Y	R/Y
LW-9230	(32bit) : address index 16	R/W	R/Y	R/Y
LW-9232	(32bit) : address index 17	R/W	R/Y	R/Y
LW-9234	(32bit) : address index 18	R/W	R/Y	R/Y
LW-9236	(32bit) : address index 19	R/W	R/Y	R/Y
LW-9238	(32bit) : address index 20	R/W	R/Y	R/Y
LW-9240	(32bit) : address index 21	R/W	R/Y	R/Y
LW-9242	(32bit) : address index 22	R/W	R/Y	R/Y
LW-9244	(32bit) : address index 23	R/W	R/Y	R/Y
LW-9246	(32bit) : address index 24	R/W	R/Y	R/Y
LW-9248	(32bit) : address index 25	R/W	R/Y	R/Y
LW-9250	(32bit) : address index 26	R/W	R/Y	R/Y
LW-9252	(32bit) : address index 27	R/W	R/Y	R/Y
LW-9254	(32bit) : address index 28	R/W	R/Y	R/Y
LW-9256	(32bit) : address index 29	R/W	R/Y	R/Y
LW-9258	(32bit) : address index 30	R/W	R/Y	R/Y
LW-9260	(32bit) : address index 31	R/W	R/Y	R/Y

Please confirm your Internet connection before downloading the demo project.

22.13 Project File Information

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LW-9100	(16bit) : project name (16 words)	R	R	R
LW-9116	(32bit) : project size in bytes	R	R	R
LW-9118	(32bit) : project size in K bytes	R	R	R
LW-9120	(32bit) : compiler version	R	R	R
LW-9122	(16bit) : project compiled date [year]	R	R	R
LW-9123	(16bit) : project compiled date [month]	R	R	R
LW-9124	(16bit) : project compiled date [day]	R	R	R

22.14 MODBUS Server Communication

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LB-9055	MODBUS server (COM 1) receives a request (when ON)	R	R	R
LB-9056	MODBUS server (COM 2) receives a request (when ON)	R	R	R
LB-9057	MODBUS server (COM 3) receives a request (when ON)	R	R	R
LB-9058	MODBUS server (ethernet) receives a request (when ON)	R	R	R
LW-9270	(16bit): request's function code - MODBUS server (COM 1)	R	R	R
LW-9271	(16bit): request's starting address - MODBUS server (COM 1)	R	R	R
LW-9272	(16bit): request's quantity of registers - MODBUS server (COM 1)	R	R	R
LW-9275	(16bit): request's function code - MODBUS server (COM 2)	R	R	R
LW-9276	(16bit): request's starting address - MODBUS server (COM 2)	R	R	R
LW-9277	(16bit): request's quantity of registers - MODBUS server (COM 2)	R	R	R
LW-9280	(16bit): request's function code - MODBUS server (COM 3)	R	R	R
LW-9281	(16bit): request's starting address - MODBUS server (COM 3)	R	R	R
LW-9282	(16bit): request's quantity of registers - MODBUS server (COM 3)	R	R	R
LW-9285	(16bit) : request's function code - MODBUS server (ethernet)	R	R	R
LW-9286	(16bit) : request's starting address - MODBUS server (ethernet)	R	R	R
LW-9287	(16bit) : request's quantity of registers - MODBUS server (ethernet)	R	R	R
LW-9288	(16bit) : last error code - MODBUS server (ethernet)	R	R	R
LW-9541	(16bit) : MODBUS/ASCII server station no. (COM 1)	R/W	R/Y	R/Y
LW-9542	(16bit) : MODBUS/ASCII server station no. (COM 2)	R/W	R/Y	R/Y
LW-9543	(16bit) : MODBUS/ASCII server station no. (COM 3)	R/W	R/Y	R/Y
LW-9544	(16bit) : MODBUS/ASCII server station no. (ethernet)	R/W	R/Y	R/Y
LW-9570	(32bit) : received data count (bytes) (COM 1 MODBUS server)	R	R	R
LW-9572	(32bit) : received data count (bytes) (COM 2 MODBUS server)	R	R	R
LW-9574	(32bit) : received data count (bytes) (COM 3 MODBUS server)	R	R	R
LW-9576	(32bit) : received data count (bytes) (Ethernet MODBUS server)	R	R	R

22.15 Communication Parameters Settings

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LB-9030	update COM 1 communication parameters (set ON)	R/W	R/Y	R/Y
LB-9031	update COM 2 communication parameters (set ON)	R/W	R/Y	R/Y
LB-9032	update COM 3 communication parameters (set ON)	R/W	R/Y	R/Y
LB-9065	disable/enable COM1 broadcast station no.	R/W	R/Y	R/Y
LB-9066	disable/enable COM2 broadcast station no.	R/W	R/Y	R/Y
LB-9067	disable/enable COM3 broadcast station no.	R/W	R/Y	R/Y
LW-9550	(16bit) : COM 1 mode(0:RS232,1:RS485 2W,2:RS485 4W)	R/W	R/Y	R/Y
LW-9551	(16bit) : COM 1 baud rate (7:1200,8:2400,0:4800,1:9600,10:14400,2:19200,11:28800,3:38400,4:57600,..)	R/W	R/Y	R/Y
LW-9552	(16bit) : COM 1 databits (7 : 7 bits, 8 : 8 bits)	R/W	R/Y	R/Y
LW-9553	(16bit) : COM 1 parity (0:none, 1:even, 2:odd, 3:mark, 4:space)	R/W	R/Y	R/Y
LW-9554	(16bit) : COM 1 stop bits (1 : 1 bit, 2 : 2 bits)	R/W	R/Y	R/Y
LW-9555	(16bit) : COM 2 mode(0:RS232,1:RS485 2W,2:RS485 4W)	R/W	R/Y	R/Y
LW-9556	(16bit) : COM 2 baud rate (7:1200,8:2400,0:4800,1:9600,10:14400,2:19200,11:28800,3:38400,4:57600,..)	R/W	R/Y	R/Y
LW-9557	(16bit) : COM 2 databits (7 : 7 bits, 8 : 8 bits)	R/W	R/Y	R/Y
LW-9558	(16bit) : COM 2 parity (0:none, 1:even, 2:odd, 3:mark, 4:space)	R/W	R/Y	R/Y
LW-9559	(16bit) : COM 2 stop bits (1 : 1 bit, 2 : 2 bits)	R/W	R/Y	R/Y
LW-9560	(16bit) : COM 3 mode(0:RS232,1:RS485 2W)	R/W	R/Y	R/Y
LW-9561	(16bit) : COM 3 baud rate (7:1200,8:2400,0:4800,1:9600,10:14400,2:19200,11:28800,3:38400,4:57600,..)	R/W	R/Y	R/Y
LW-9562	(16bit) : COM 3 databits (7 : 7 bits, 8 : 8 bits)	R/W	R/Y	R/Y
LW-9563	(16bit) : COM 3 parity (0:none, 1:even, 2:odd, 3:mark, 4:space)	R/W	R/Y	R/Y
LW-9564	(16bit) : COM 3 stop bits (1 : 1 bit, 2 : 2 bits)	R/W	R/Y	R/Y
LW-9565	(16bit) : COM 1 broadcast station no.	R/W	R/Y	R/Y
LW-9566	(16bit) : COM 2 broadcast station no.	R/W	R/Y	R/Y
LW-9567	(16bit) : COM 3 broadcast station no.	R/W	R/Y	R/Y
LW-10500	(16bit) : PLC 1 timeout (unit : 100ms)	R/W	R/Y	R/Y
LW-10501	(16bit) : PLC 1 turn around delay (unit : ms)	R/W	R/Y	R/Y
LW-10502	(16bit) : PLC 1 send ACK delay (unit : ms)	R/W	R/Y	R/Y

LW-10503	(16bit) : PLC 1 parameter 1	R/W	R/Y	R/Y
LW-10504	(16bit) : PLC 1 parameter 2	R/W	R/Y	R/Y
LW-10505	(16bit) : PLC 2 timeout (unit : 100ms)	R/W	R/Y	R/Y
LW-10506	(16bit) : PLC 2 turn around delay (unit : ms)	R/W	R/Y	R/Y
LW-10507	(16bit) : PLC 2 send ACK delay (unit : ms)	R/W	R/Y	R/Y
LW-10508	(16bit) : PLC 2 parameter 1	R/W	R/Y	R/Y
LW-10509	(16bit) : PLC 2 parameter 2	R/W	R/Y	R/Y
LW-10510	(16bit) : PLC 3 timeout (unit : 100ms)	R/W	R/Y	R/Y
LW-10511	(16bit) : PLC 3 turn around delay (unit : ms)	R/W	R/Y	R/Y
LW-10512	(16bit) : PLC 3 send ACK delay (unit : ms)	R/W	R/Y	R/Y
LW-10513	(16bit) : PLC 3 parameter 1	R/W	R/Y	R/Y
LW-10514	(16bit) : PLC 3 parameter 2	R/W	R/Y	R/Y
LW-10515	(16bit) : PLC 4 timeout (unit : 100ms)	R/W	R/Y	R/Y
LW-10516	(16bit) : PLC 4 turn around delay (unit : ms)	R/W	R/Y	R/Y
LW-10517	(16bit) : PLC 4 send ACK delay (unit : ms) (SIEMENS S7/400 Link type)	R/W	R/Y	R/Y
LW-10518	(16bit) : PLC 4 parameter 1 (SIEMENS S7/400 rack)	R/W	R/Y	R/Y
LW-10519	(16bit) : PLC 4 parameter 2 (SIEMENS S7/400 CPU slot)	R/W	R/Y	R/Y
LW-10520	(16bit) : PLC 5 timeout (unit : 100ms)	R/W	R/Y	R/Y
LW-10521	(16bit) : PLC 5 turn around delay (unit : ms)	R/W	R/Y	R/Y
LW-10522	(16bit) : PLC 5 send ACK delay (unit : ms) (SIEMENS S7/400 Link type)	R/W	R/Y	R/Y
LW-10523	(16bit) : PLC 5 parameter 1 (SIEMENS S7/400 rack)	R/W	R/Y	R/Y
LW-10524	(16bit) : PLC 5 parameter 2 (SIEMENS S7/400 CPU slot)	R/W	R/Y	R/Y
LW-10525	(16bit) : PLC 6 timeout (unit : 100ms)	R/W	R/Y	R/Y
LW-10526	(16bit) : PLC 6 turn around delay (unit : ms)	R/W	R/Y	R/Y
LW-10527	(16bit) : PLC 6 send ACK delay (unit : ms) (SIEMENS S7/400 Link type)	R/W	R/Y	R/Y
LW-10528	(16bit) : PLC 6 parameter 1 (SIEMENS S7/400 rack)	R/W	R/Y	R/Y
LW-10529	(16bit) : PLC 6 parameter 2 (SIEMENS S7/400 CPU slot)	R/W	R/Y	R/Y
LW-10530	(16bit) : PLC 7 timeout (unit : 100ms)	R/W	R/Y	R/Y
LW-10531	(16bit) : PLC 7 turn around delay (unit : ms)	R/W	R/Y	R/Y
LW-10532	(16bit) : PLC 7 send ACK delay (unit : ms) (SIEMENS S7/400 Link type)	R/W	R/Y	R/Y
LW-10533	(16bit) : PLC 7 parameter 1 (SIEMENS S7/400 rack)	R/W	R/Y	R/Y
LW-10534	(16bit) : PLC 7 parameter 2 (SIEMENS S7/400 CPU slot)	R/W	R/Y	R/Y

LW-10535	(16bit) : PLC 8 timeout (unit : 100ms)	R/W	R/Y	R/Y
LW-10536	(16bit) : PLC 8 turn around delay (unit : ms)	R/W	R/Y	R/Y
LW-10537	(16bit) : PLC 8 send ACK delay (unit : ms) (SIEMENS S7/400 Link type)	R/W	R/Y	R/Y
LW-10538	(16bit) : PLC 8 parameter 1 (SIEMENS S7/400 rack)	R/W	R/Y	R/Y
LW-10539	(16bit) : PLC 8 parameter 2 (SIEMENS S7/400 CPU slot)	R/W	R/Y	R/Y

22.16 Communication Status with PLC (COM)

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LB-9150	auto. connection for PLC 1 (COM1) (when ON)	R/W	R/Y	R/Y
LB-9151	auto. connection for PLC 2 (COM2) (when ON)	R/W	R/Y	R/Y
LB-9152	auto. connection for PLC 3 (COM3) (when ON)	R/W	R/Y	R/Y
LB-9200	PLC 1 status (SN0, COM1), set on to retry connection *Note1	R/W	R/Y	R/Y
LB-9201	PLC 1 status (SN1, COM1), set on to retry connection	R/W	R/Y	R/Y
LB-9202	PLC 1 status (SN2, COM1), set on to retry connection	R/W	R/Y	R/Y
LB-9203	PLC 1 status (SN3, COM1), set on to retry connection	R/W	R/Y	R/Y
LB-9204	PLC 1 status (SN4, COM1), set on to retry connection	R/W	R/Y	R/Y
LB-9205	PLC 1 status (SN5, COM1), set on to retry connection	R/W	R/Y	R/Y
LB-9206	PLC 1 status (SN6, COM1), set on to retry connection	R/W	R/Y	R/Y
LB-9207	PLC 1 status (SN7, COM1), set on to retry connection	R/W	R/Y	R/Y
LB-9500	PLC 2 status (SN0, COM2), set on to retry connection *Note2	R/W	R/Y	R/Y
LB-9501	PLC 2 status (SN1, COM2), set on to retry connection	R/W	R/Y	R/Y
LB-9502	PLC 2 status (SN2, COM2), set on to retry connection	R/W	R/Y	R/Y
LB-9503	PLC 2 status (SN3, COM2), set on to retry connection	R/W	R/Y	R/Y
LB-9504	PLC 2 status (SN4, COM2), set on to retry connection	R/W	R/Y	R/Y
LB-9505	PLC 2 status (SN5, COM2), set on to retry connection	R/W	R/Y	R/Y
LB-9506	PLC 2 status (SN6, COM2), set on to retry connection	R/W	R/Y	R/Y
LB-9507	PLC 2 status (SN7, COM2), set on to retry connection	R/W	R/Y	R/Y
LB-9800	PLC 3 status (SN0, COM3), set on to retry connection *Note3	R/W	R/Y	R/Y
LB-9801	PLC 3 status (SN1, COM3), set on to retry connection	R/W	R/Y	R/Y
LB-9802	PLC 3 status (SN2, COM3), set on to retry connection	R/W	R/Y	R/Y
LB-9803	PLC 3 status (SN3, COM3), set on to retry connection	R/W	R/Y	R/Y
LB-9804	PLC 3 status (SN4, COM3), set on to retry connection	R/W	R/Y	R/Y
LB-9805	PLC 3 status (SN5, COM3), set on to retry connection	R/W	R/Y	R/Y
LB-9806	PLC 3 status (SN6, COM3), set on to retry connection	R/W	R/Y	R/Y
LB-9807	PLC 3 status (SN7, COM3), set on to retry connection	R/W	R/Y	R/Y
LB-12030	COM 1 status (OFF : normal, ON : open failed) *Note4	R	R	R
LB-12031	COM 2 status (OFF : normal, ON : open failed)	R	R	R
LB-12032	COM 3 status (OFF : normal, ON : open failed)	R	R	R

LB-12033	COM 4 status (OFF : normal, ON : open failed)	R	R	R
LB-12034	COM 5 status (OFF : normal, ON : open failed)	R	R	R
LB-12035	COM 6 status (OFF : normal, ON : open failed)	R	R	R
LB-12036	COM 7 status (OFF : normal, ON : open failed)	R	R	R
LB-12037	COM 8 status (OFF : normal, ON : open failed)	R	R	R
LB-12038	COM 9 status (OFF : normal, ON : open failed)	R	R	R

1. LB-9200 to LB-9455 (station number 0 to 255, COM 1) are registers relevant to the communication status of COM 1.
2. LB-9500 to LB-9755 (station number 0 to 255, COM 2) are registers relevant to the communication status of COM 2.
3. LB-9800 to LB-10055 (station number 0 to 255, COM 3) are registers relevant to the communication status of COM 3.
4. The ON state of COM is for checking if COM is occupied by other program during simulation on PC.

22.17 Communication Status with PLC (Ethernet)

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LB-9153	auto. connection for PLC 4 (ethernet) (when ON) *Note1	R/W	R/Y	R/Y
LB-9154	auto. connection for PLC 5 (ethernet) (when ON)	R/W	R/Y	R/Y
LB-9155	auto. connection for PLC 6 (ethernet) (when ON)	R/W	R/Y	R/Y
LB-9156	auto. connection for PLC 7 (ethernet) (when ON)	R/W	R/Y	R/Y
LB-9157	auto. connection for PLC 8 (ethernet) (when ON)	R/W	R/Y	R/Y
LB-9158	auto. connection for PLC 9 (ethernet) (when ON)	R/W	R/Y	R/Y
LB-10070	forced to reconnect PLC 4 (ethernet) when IP or system parameters changed on-line (set ON) *Note2	R/W	R/Y	R/Y
LB-10071	forced to reconnect PLC 5 (ethernet) when IP or system parameters changed on-line (set ON)	R/W	R/Y	R/Y
LB-10072	forced to reconnect PLC 6 (ethernet) when IP or system parameters changed on-line (set ON)	R/W	R/Y	R/Y
LB-10073	forced to reconnect PLC 7 (ethernet) when IP or system parameters changed on-line (set ON)	R/W	R/Y	R/Y
LB-10074	forced to reconnect PLC 8 (ethernet) when IP or system parameters changed on-line (set ON)	R/W	R/Y	R/Y
LB-10075	forced to reconnect PLC 9 (ethernet) when IP or system parameters changed on-line (set ON)	R/W	R/Y	R/Y
LB-10100	PLC 4 status (ethernet), set on to retry connection *Note3	R/W	R/Y	R/Y
LB-10400	PLC 5 status (ethernet), set on to retry connection	R/W	R/Y	R/Y
LB-10700	PLC 6 status (ethernet), set on to retry connection	R/W	R/Y	R/Y
LB-11000	PLC 7 status (ethernet), set on to retry connection	R/W	R/Y	R/Y
LB-11300	PLC 8 status (ethernet), set on to retry connection	R/W	R/Y	R/Y
LB-11600	PLC 9 status (ethernet), set on to retry connection	R/W	R/Y	R/Y
LB-11900	PLC 10 status (ethernet), set on to retry connection	R/W	R/Y	R/Y
LB-11901	PLC 11 status (ethernet), set on to retry connection	R/W	R/Y	R/Y
LB-11902	PLC 12 status (ethernet), set on to retry connection	R/W	R/Y	R/Y
LB-11903	PLC 13 status (ethernet), set on to retry connection	R/W	R/Y	R/Y
LB-11904	PLC 14 status (ethernet), set on to retry connection	R/W	R/Y	R/Y
LB-11905	PLC 15 status (ethernet), set on to retry connection	R/W	R/Y	R/Y
LB-11906	PLC 16 status (ethernet), set on to retry connection	R/W	R/Y	R/Y
LW-9600	(16bit) : PLC 4's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y

	*Note4			
LW-9601	(16bit) : PLC 4's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9602	(16bit) : PLC 4's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9603	(16bit) : PLC 4's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9604	(16bit) : PLC 4's port no.	R/W	R/Y	R/Y
LW-9605	(16bit) : PLC 5's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9606	(16bit) : PLC 5's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9607	(16bit) : PLC 5's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9608	(16bit) : PLC 5's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9609	(16bit) : PLC 5's port no.	R/W	R/Y	R/Y
LW-9610	(16bit) : PLC 6's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9611	(16bit) : PLC 6's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9612	(16bit) : PLC 6's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9613	(16bit) : PLC 6's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9614	(16bit) : PLC 6's port no.	R/W	R/Y	R/Y
LW-9615	(16bit) : PLC 7's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9616	(16bit) : PLC 7's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9617	(16bit) : PLC 7's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9618	(16bit) : PLC 7's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9619	(16bit) : PLC 7's port no.	R/W	R/Y	R/Y
LW-9620	(16bit) : PLC 8's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9621	(16bit) : PLC 8's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9622	(16bit) : PLC 8's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9623	(16bit) : PLC 8's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9624	(16bit) : PLC 8's port no.	R/W	R/Y	R/Y
LW-9625	(16bit) : PLC 9's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9626	(16bit) : PLC 9's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9627	(16bit) : PLC 9's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9628	(16bit) : PLC 9's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9629	(16bit) : PLC 9's port no.	R/W	R/Y	R/Y

1. LB-9153 to LB-9189 are registers relevant to auto connection with PLC (PLC 4 to 40).
2. LB-10070 to LB-10081 are registers relevant to re-connection with PLC (PLC 4 to 15).
3. LB-10100 to LB-11939 are registers relevant to communication status with PLC (PLC 4 to 49).
4. LW-9600 to LW-9769 are registers relevant to setting IP address of PLC (PLC 4 to 37).

22.18 Communication Status with PLC (USB)

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LB-9190	auto. connection for PLC (USB 1) (when ON)	R/W	R/Y	R/Y
LB-9191	PLC status (USB 1), set on to retry connection	R/W	R/Y	R/Y
LB-9193	auto. connection for PLC (USB 2) (when ON)	R/W	R/Y	R/Y
LB-9194	PLC status (USB 2), set on to retry connection	R/W	R/Y	R/Y

22.19 Communication Status with Remote HMI

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LB-9068	auto. connection for remote HMI 1 (when ON) *Note1	R/W	R/Y	R/Y
LB-9069	auto. connection for remote HMI 2 (when ON)	R/W	R/Y	R/Y
LB-9070	auto. connection for remote HMI 3 (when ON)	R/W	R/Y	R/Y
LB-9071	auto. connection for remote HMI 4 (when ON)	R/W	R/Y	R/Y
LB-9072	auto. connection for remote HMI 5 (when ON)	R/W	R/Y	R/Y
LB-9073	auto. connection for remote HMI 6 (when ON)	R/W	R/Y	R/Y
LB-9074	auto. connection for remote HMI 7 (when ON)	R/W	R/Y	R/Y
LB-9075	auto. connection for remote HMI 8 (when ON)	R/W	R/Y	R/Y
LB-9100	remote HMI 1 status (set on to retry connection) *Note2	R/W	R/Y	R/Y
LB-9101	remote HMI 2 status (set on to retry connection)	R/W	R/Y	R/Y
LB-9102	remote HMI 3 status (set on to retry connection)	R/W	R/Y	R/Y
LB-9103	remote HMI 4 status (set on to retry connection)	R/W	R/Y	R/Y
LB-9104	remote HMI 5 status (set on to retry connection)	R/W	R/Y	R/Y
LB-9105	remote HMI 6 status (set on to retry connection)	R/W	R/Y	R/Y
LB-9106	remote HMI 7 status (set on to retry connection)	R/W	R/Y	R/Y
LB-9107	remote HMI 8 status (set on to retry connection)	R/W	R/Y	R/Y
LB-9149	forced to reconnect remote HMI when IP changed on-line (set ON)	R/W	R/Y	R/Y
LW-9800	(16bit) : remote HMI 1's IP0 (IP address = IP0:IP1:IP2:IP3) *Note3	R/W	R/Y	R/Y
LW-9801	(16bit) : remote HMI 1's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9802	(16bit) : remote HMI 1's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9803	(16bit) : remote HMI 1's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9804	(16bit) : remote HMI 1's port no.	R/W	R/Y	R/Y
LW-9805	(16bit) : remote HMI 2's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9806	(16bit) : remote HMI 2's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y

LW-9807	(16bit) : remote HMI 2's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9808	(16bit) : remote HMI 2's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9809	(16bit) : remote HMI 2's port no.	R/W	R/Y	R/Y
LW-9810	(16bit) : remote HMI 3's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9811	(16bit) : remote HMI 3's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9812	(16bit) : remote HMI 3's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9813	(16bit) : remote HMI 3's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9814	(16bit) : remote HMI 3's port no.	R/W	R/Y	R/Y
LW-9815	(16bit) : remote HMI 4's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9816	(16bit) : remote HMI 4's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9817	(16bit) : remote HMI 4's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9818	(16bit) : remote HMI 4's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9819	(16bit) : remote HMI 4's port no.	R/W	R/Y	R/Y
LW-9820	(16bit) : remote HMI 5's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9821	(16bit) : remote HMI 5's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9822	(16bit) : remote HMI 5's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9823	(16bit) : remote HMI 5's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9824	(16bit) : remote HMI 5's port no.	R/W	R/Y	R/Y
LW-9825	(16bit) : remote HMI 6's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9826	(16bit) : remote HMI 6's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9827	(16bit) : remote HMI 6's IP2 (IP address =	R/W	R/Y	R/Y

	IP0:IP1:IP2:IP3)			
LW-9828	(16bit) : remote HMI 6's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9829	(16bit) : remote HMI 6's port no.	R/W	R/Y	R/Y
LW-9830	(16bit) : remote HMI 7's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9831	(16bit) : remote HMI 7's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9832	(16bit) : remote HMI 7's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9833	(16bit) : remote HMI 7's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9834	(16bit) : remote HMI 7's port no.	R/W	R/Y	R/Y
LW-9835	(16bit) : remote HMI 8's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9836	(16bit) : remote HMI 8's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9837	(16bit) : remote HMI 8's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9838	(16bit) : remote HMI 8's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9839	(16bit) : remote HMI 8's port no.	R/W	R/Y	R/Y
LW-9905	(16bit) : remote HMI 21's IP0 (IP address = IP0:IP1:IP2:IP3) *Note4	R/W	R/Y	R/Y
LW-9906	(16bit) : remote HMI 21's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9907	(16bit) : remote HMI 21's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9908	(16bit) : remote HMI 21's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9909	(16bit) : remote HMI 21's port no.	R/W	R/Y	R/Y
LW-9910	(16bit) : remote HMI 22's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9911	(16bit) : remote HMI 22's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9912	(16bit) : remote HMI 22's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y

LW-9913	(16bit) : remote HMI 22's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9914	(16bit) : remote HMI 22's port no.	R/W	R/Y	R/Y
LW-9915	(16bit) : remote HMI 23's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9916	(16bit) : remote HMI 23's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9917	(16bit) : remote HMI 23's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9918	(16bit) : remote HMI 23's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9919	(16bit) : remote HMI 23's port no.	R/W	R/Y	R/Y
LW-9920	(16bit) : remote HMI 24's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9921	(16bit) : remote HMI 24's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9922	(16bit) : remote HMI 24's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9923	(16bit) : remote HMI 24's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9924	(16bit) : remote HMI 24's port no.	R/W	R/Y	R/Y
LW-9925	(16bit) : remote HMI 25's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9926	(16bit) : remote HMI 25's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9927	(16bit) : remote HMI 25's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9928	(16bit) : remote HMI 25's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9929	(16bit) : remote HMI 25's port no.	R/W	R/Y	R/Y
LW-9930	(16bit) : remote HMI 26's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9931	(16bit) : remote HMI 26's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9932	(16bit) : remote HMI 26's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9933	(16bit) : remote HMI 26's IP3 (IP address =	R/W	R/Y	R/Y

	IP0:IP1:IP2:IP3)			
LW-9934	(16bit) : remote HMI 26's port no.	R/W	R/Y	R/Y
LW-9935	(16bit) : remote HMI 27's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9936	(16bit) : remote HMI 27's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9937	(16bit) : remote HMI 27's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9938	(16bit) : remote HMI 27's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9939	(16bit) : remote HMI 27's port no.	R/W	R/Y	R/Y
LW-9940	(16bit) : remote HMI 28's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9941	(16bit) : remote HMI 28's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9942	(16bit) : remote HMI 28's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9943	(16bit) : remote HMI 28's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9944	(16bit) : remote HMI 28's port no.	R/W	R/Y	R/Y
LW-9945	(16bit) : remote HMI 29's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9946	(16bit) : remote HMI 29's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9947	(16bit) : remote HMI 29's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9948	(16bit) : remote HMI 29's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9949	(16bit) : remote HMI 29's port no.	R/W	R/Y	R/Y
LW-9950	(16bit) : remote HMI 30's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9951	(16bit) : remote HMI 30's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9952	(16bit) : remote HMI 30's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9953	(16bit) : remote HMI 30's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y

LW-9954	(16bit) : remote HMI 30's port no.	R/W	R/Y	R/Y
LW-9955	(16bit) : remote HMI 31's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9956	(16bit) : remote HMI 31's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9957	(16bit) : remote HMI 31's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9958	(16bit) : remote HMI 31's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9959	(16bit) : remote HMI 31's port no.	R/W	R/Y	R/Y
LW-9960	(16bit) : remote HMI 32's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9961	(16bit) : remote HMI 32's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9962	(16bit) : remote HMI 32's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9963	(16bit) : remote HMI 32's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9964	(16bit) : remote HMI 32's port no.	R/W	R/Y	R/Y

1. LB-9068 to LB-9099 are registers relevant to auto. connection with remote HMI (HMI 1 to 32).
2. LB-9100 to LB-9148 are registers relevant to communication status with remote HMI (HMI 1 to 49).
3. LW-9800 to LW-9899 are registers relevant to setting IP address of remote HMI (HMI 1 to 20).
4. LW-9905 to LW-9999 are registers relevant to setting IP address of remote HMI (HMI 21 to 39).

22.20 Communication Status with Remote PLC

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LW-10050	(16bit) : IP0 of the HMI connecting to remote PLC 1 (IP address = IP0:IP1:IP2:IP3) *Note1	R/W	R/Y	R/Y
LW-10051	(16bit) : IP1 of the HMI connecting to remote PLC 1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10052	(16bit) : IP2 of the HMI connecting to remote PLC 1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10053	(16bit) : IP3 of the HMI connecting to remote PLC 1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10054	(16bit) : port no. of the HMI connecting to remote PLC 1	R/W	R/Y	R/Y
LW-10055	(16bit) : IP0 of the HMI connecting to remote PLC 2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10056	(16bit) : IP1 of the HMI connecting to remote PLC 2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10057	(16bit) : IP2 of the HMI connecting to remote PLC 2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10058	(16bit) : IP3 of the HMI connecting to remote PLC 2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10059	(16bit) : port no. of the HMI connecting to remote PLC 2	R/W	R/Y	R/Y
LW-10060	(16bit) : IP0 of the HMI connecting to remote PLC 3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10061	(16bit) : IP1 of the HMI connecting to remote PLC 3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10062	(16bit) : IP2 of the HMI connecting to remote PLC 3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10063	(16bit) : IP3 of the HMI connecting to remote PLC 3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10064	(16bit) : port no. of the HMI connecting to remote PLC 3	R/W	R/Y	R/Y
LW-10065	(16bit) : IP0 of the HMI connecting to remote PLC 4 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10066	(16bit) : IP1 of the HMI connecting to remote	R/W	R/Y	R/Y

	PLC 4 (IP address = IP0:IP1:IP2:IP3)			
LW-10067	(16bit) : IP2 of the HMI connecting to remote PLC 4 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10068	(16bit) : IP3 of the HMI connecting to remote PLC 4 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10069	(16bit) : port no. of the HMI connecting to remote PLC 4	R/W	R/Y	R/Y
LW-10300	(16bit) : remote PLC 1's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10301	(16bit) : remote PLC 1's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10302	(16bit) : remote PLC 1's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10303	(16bit) : remote PLC 1's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10304	(16bit) : remote PLC 1's port no.	R/W	R/Y	R/Y
LW-10305	(16bit) : remote PLC 2's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10306	(16bit) : remote PLC 2's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10307	(16bit) : remote PLC 2's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10308	(16bit) : remote PLC 2's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10309	(16bit) : remote PLC 2's port no.	R/W	R/Y	R/Y
LW-10310	(16bit) : remote PLC 3's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10311	(16bit) : remote PLC 3's IP1 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10312	(16bit) : remote PLC 3's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10313	(16bit) : remote PLC 3's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10314	(16bit) : remote PLC 3's port no.	R/W	R/Y	R/Y
LW-10315	(16bit) : remote PLC 4's IP0 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10316	(16bit) : remote PLC 4's IP1 (IP address =	R/W	R/Y	R/Y

	IP0:IP1:IP2:IP3)			
LW-10317	(16bit) : remote PLC 4's IP2 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10318	(16bit) : remote PLC 4's IP3 (IP address = IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-10319	(16bit) : remote PLC 4's port no.	R/W	R/Y	R/Y

1. LW-10050 to LW-10299 are registers relevant to the IP address of the HMI connected to remote PLC. (The HMI connected to PLC 1 to the HMI connected to PLC 50)

22.21 Communication Error Messages & No. of Pending Cmd.

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LW-9350	(16bit) : pending command no. in local HMI	R	R	R
LW-9351	(16bit) : pending command no. in PLC 1 (COM 1) *Note1	R	R	R
LW-9352	(16bit) : pending command no. in PLC 2 (COM 2)	R	R	R
LW-9353	(16bit) : pending command no. in PLC 3 (COM 3)	R	R	R
LW-9354	(16bit) : pending command no. in PLC 4 (ethernet)	R	R	R
LW-9355	(16bit) : pending command no. in PLC 5 (ethernet)	R	R	R
LW-9356	(16bit) : pending command no. in PLC 6 (ethernet)	R	R	R
LW-9357	(16bit) : pending command no. in PLC 7 (ethernet)	R	R	R
LW-9390	(16bit) : pending command no. in PLC (USB)	R	R	R
LW-9400	(16bit) : error code for PLC 1 *Note2	R	R	R
LW-9401	(16bit) : error code for PLC 2	R	R	R
LW-9402	(16bit) : error code for PLC 3	R	R	R
LW-9403	(16bit) : error code for PLC 4	R	R	R
LW-9404	(16bit) : error code for PLC 5	R	R	R
LW-9405	(16bit) : error code for PLC 6	R	R	R
LW-9406	(16bit) : error code for PLC 7	R	R	R
LW-9407	(16bit) : error code for PLC 8	R	R	R
LW-9490	(16bit) : error code for USB PLC	R	R	R

1. LW-9351 to LW-9389 are registers relevant to the number of pending commands in PLC (PLC 1 to 39).
2. LW-9400 to LW-9449 are registers relevant to the errors occur during communication with PLC (PLC 1 to 50).

22.22 Miscellaneous Functions

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LB-9000 - LB-9009	initialized as ON	R/W	R/Y	R/Y
LB-9010	data download indicator	R	R	R
LB-9011	data upload indicator	R	R	R
LB-9012	data download/upload indicator	R	R	R
LB-9016	status is on when a client connects to this HMI	R	R	R
LB-9017	disable write-back in PLC control's [change window]	R/W	R/Y	R/Y
LB-9039	status of file backup activity (backup in process if ON)	R	R	R
LB-9045	memory-map communication fails (when ON)	R	R	R
LB-9049	enable (set ON)/disable (set OFF) watch dog (i series support only) *Note1	R/W	R/Y	R/Y
LB-9059	disable MACRO TRACE function (when ON) *Note2	R/W	R/Y	R/Y
LB-9064	enable USB barcode device (disable keyboard) (when ON) *Note3	R/W	R/Y	R
LW-9006	(16bit) : connected client no.	R	R	R
LW-9024	(16bit) : memory link system register	R/W	R/Y	R/Y
LW-9032	(8 words) : folder name of backup history files to SD, USB memory	R/W	R/Y	R/Y
LW-9050	(16bit) : current base window ID	R	R	R
LW-9134	(16bit) : language mode *Note4	R/W	R/Y	R/Y
LW-9141	(16bit) : HMI station no.	R/W	R/Y	R/Y
LW-9300	(16bit) : driver ID of local PLC 1	R	R	R
LW-9301	(16bit) : driver ID of local PLC 2	R	R	R
LW-9302	(16bit) : driver ID of local PLC 3	R	R	R
LW-9303	(16bit) : driver ID of local PLC 4	R	R	R
LW-9900	(16bit) : HMI run mode (0 : normal mode, 1-3 : test mode (COM 1-COM 3)	R/W	R/Y	R/Y

1. When LB-9049 watch dog function is enabled, if there's a failure in communication for i Series HMI, system will reboot 10 seconds later.
2. LB-9059 Demo project: disable MACRO TRACE function.

   Please confirm your Internet connection before downloading the demo project.
3. LB-9064 Demo project: enable USB barcode device (disable keyboard).

   Please confirm your Internet connection before downloading the demo project.
4. When users would like to have the object's text to show multi-language, except for using Label Library, it needs to use the system reserved register [LW-9134: language mode]. The value of LW-9134 can be set from 0 to 7. Different data of LW-9134 corresponds to different Languages. The way of using LW-9134 will differ if the languages are not all chosen when compiling the downloaded file.
   For example: If 5 languages are defined by user in Label Library as Language 1 (Traditional Chinese), Language 2 (Simplified Chinese), Language 3 (English), Language 4 (French), and Language 5 (Japanese). If only Language 1, 3, 5 are downloaded by user, the corresponding language of the value in LW-9134 will be 0 -> Language 1 (Traditional Chinese), 1 -> Language 3 (English), 2 -> Language 5 (Japanese).
   Want to know how to swith languages using Option List object toghther with LW-9134?

   Please confirm your Internet connection before downloading the demo project.

22.23 Remote Print/Backup Server

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LB-10069	forced to reconnect remote printer/backup server when IP changed on-line (set ON)	R/W	R/Y	R/Y
LB-12040	remote printer/backup server disconnection alarm (when ON)	R	R	R
LW-9770	(16bit) : remote printer/backup server IP0 (IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9771	(16bit) : remote printer/backup server IP1 (IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9772	(16bit) : remote printer/backup server IP2 (IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9773	(16bit) : remote printer/backup server IP3 (IP0:IP1:IP2:IP3)	R/W	R/Y	R/Y
LW-9774	(6 words) : remote printer/backup server user name *Note1	R/W	R/Y	R/Y
LW-9780	(6 words) : remote printer/backup server password *Note1	R/W	R/Y	R/Y

1. When change settings using LW-9774 and LW-9780, please reboot HMI to enable the new settings.

Please confirm your Internet connection before downloading the demo project.

22.24 EasyAccess

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LB-9051	disconnect (set OFF)/connect (set ON) EasyAccess server	R/W	R/Y	R/Y
LB-9052	status of connecting to EasyAccess server	R	R	R
LB-9196	local HMI supports monitor function only (when ON)	R/W	R/Y	R/Y
LB-9197	support monitor function only for remote HMIs (when ON)	R/W	R/Y	R/Y

For further information on EasyAccess, please visit http://www.ihmi.net/.

Please confirm your Internet connection before downloading the demo project.

22.25 Pass-Through Settings

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LW-9901	(16bit) : pass-through source COM port (1-3 : COM 1-COM 3)	R/W	R/Y	R/Y
LW-9902	(16bit) : pass-through destination COM port (1-3 : COM 1-COM 3)	R/W	R/Y	R/Y
LW-9903	(16bit) : pass-through control (0 : normal, 1 : pause, 2 : stop communications between HMI and PLC when executing pass-through)	R/W	R/Y	R/Y

Please confirm your Internet connection before downloading the demo project.

22.26 Disable PLC No Response Window

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LB-9192	disable USB 1 PLC's "PLC No Response" dialog (when ON)	R/W	R/Y	R/Y
LB-9195	disable USB 2 PLC's "PLC No Response" dialog (when ON)	R/W	R/Y	R/Y
LB-11960	disable PLC 1's "PLC No Response" dialog (when ON) *Note1	R/W	R/Y	R/Y
LB-11961	disable PLC 2's "PLC No Response" dialog (when ON)	R/W	R/Y	R/Y
LB-11962	disable PLC 3's "PLC No Response" dialog (when ON)	R/W	R/Y	R/Y
LB-11963	disable PLC 4's "PLC No Response" dialog (when ON)	R/W	R/Y	R/Y
LB-11964	disable PLC 5's "PLC No Response" dialog (when ON)	R/W	R/Y	R/Y
LB-11965	disable PLC 6's "PLC No Response" dialog (when ON)	R/W	R/Y	R/Y
LB-11966	disable PLC 7's "PLC No Response" dialog (when ON)	R/W	R/Y	R/Y
LB-11967	disable PLC 8's "PLC No Response" dialog (when ON)	R/W	R/Y	R/Y

1. LB-11960 to LB-12026 are registers for disabling the “PLC No Response” pop-up dialog for PLC 1 to 67.

22.27 HMI and Project Key

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LB-9046	project key is different from HMI key (when ON)	R	R	R
LW-9046	(32bit) : HMI key (i series only) *Note1	R/W	R/Y	R

1. When change HMI key using LW-9046, please reboot HMI to enable the new settings.

Please confirm your Internet connection before downloading the demo project.

22.28 Fast Selection Window Control

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LB-9013	FS window control [hide(ON)/show(OFF)]	R/W	R/Y	R/Y
LB-9014	FS button control [hide(ON)/show(OFF)]	R/W	R/Y	R/Y
LB-9015	FS window/button control [hide(ON)/show(OFF)]	R/W	R/Y	R/Y

22.29 Input Object Function

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LW-9002	(32bit-float) : input high limit	R	R	R
LW-9004	(32bit-float) : input low limit	R	R	R
LW-9052	(32bit-float) : the previous input value of the numeric input object	R	R	R
LW-9150	(32 words) : keyboard's input data (ASCII)	R	R	R
LW-9540	(16bit) : reserved for caps lock	R/W	R/Y	R/Y

22.30 Local/Remote Operation Restrictions

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	MACRO R/Y	Remote HMI R/Y
LB-9044	disable remote control (when ON)	R/W	R/Y	R/Y
LB-9053	prohibit password remote-read operation (when ON)	R/W	R/Y	R/Y
LB-9054	prohibit password remote-write operation (when ON)	R/W	R/Y	R/Y
LB-9196	local HMI supports monitor function only (when ON)	R/W	R/Y	R/Y
LB-9197	support monitor function only for remote HMIs (when ON)	R/W	R/Y	R/Y
LB-9198	disable local HMI to trigger a MACRO (when ON)	R/W	R/Y	R/Y
LB-9199	disable remote HMI to trigger a MACRO (when ON)	R/W	R/Y	R/Y

22.31 VNC Control

Address	Description	Read(R)/Write(W)/Control(Y)		
		Local HMI	Macro R/Y	Remote HMI R/Y
LB-12090	a VNC client connecting to HMI (when ON) *Note1	R	R	R
LB-12091	disable auto-logout function when a VNC client is connected (when ON) *Note1	R/W	R/Y	R/Y
LB-12092	enable VNC (set ON), disable VNC (set OFF)	R/W	R/Y	R/Y
LW-9530	(8 words) : VNC server password	R/W	R/Y	R/Y

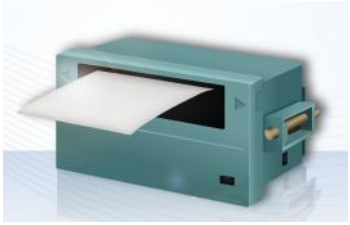
For i-Series HMI, only OS ver. 20120621 or later versions support this function.

Chapter 23 HMI Supported Printers

23.1 The Supported Printer Types

HMI supported printer drivers include the following types:

■ SP-M, D, E, F



Serial printers, please configure communication parameters to match the printer. [Pixels of width] must be correctly set and can't exceed printer default setting:
100 pixels for 1610 series printers.
220 pixels for 2407, 4004 series printers.
The driver uses EPSON ESC Protocol for Serial Micro Printer.

■ EPSON ESC/P2 Series

Serial printers, please configure communication parameters to match the printer. The EPSON ESC/P2 printer protocol is used.
Impact Printer:
LQ-300, LQ-300+, LQ-300K+ (RS232)
LQ-300+II (RS232)
Inkjet Printer: Stylus Photo 750
Laser Printer: EPL-5800

■ HP PCL Series (USB)

HP compatible USB printers that support HP PCL5 level 3 protocol.
• PCL 5 was released on HP LaserJet III in March 1990, added Intellifont font scaling (developed by Compugraphic, now part of Agfa), outline fonts and HP-GL/2 (vector) graphics.
• PCL 5e (PCL 5 enhanced) was released on HP LaserJet 4 in October 1992 and added bi-directional communication between printer and PC, and Windows fonts.

For more details of HP PCL series, please refer to 23.2.

■ Axiohm A630

Micro printer from France connects via serial port; please configure communication parameters to match the printer.

■ **SPRT**

Serial printers, please configure communication parameters to match the printer. [Pixels of width] must be correctly set and can't exceed printer default setting "100".

■ **EPSON TM-L90**

Serial printers, please configure communication parameters to match the printer. [Pixels of width] must be correctly set and can't exceed printer default setting "576".

■ **EPSON TM-T70**

Serial printers, please configure communication parameters to match the printer. [Pixels of width] must be correctly set and can't exceed printer default setting "576".
The paper cutting mode can be selected: [No cut] / [Partial cut].

■ **BRIGHTEK WH-A19**

Supported models:
A92R10-00E72A: 72 in model number represents hexadecimal printer, and A represents wide voltage 5~9V. This is the same as the A6 16 impact printer.

■ **BRIGHTEK WH-E19**

Serial printers, please configure the same communication parameters as the printer.

■ **BRIGHTEK WH-E22**

Supported models:
E22R10-00E725: Same as A7 16 impact printer. A7 represents A72R90-31E72A.
E221R90-00E11740GA: Serial printer, connects through RS-485 port, please use a RS232-to-RS485 converter.

■ BRIGHTEK WH-C1/C2

Serial printers, please configure communication parameters to match the printer. The paper cutting mode can be selected: [No cut] / [Half cut] / Full cut].

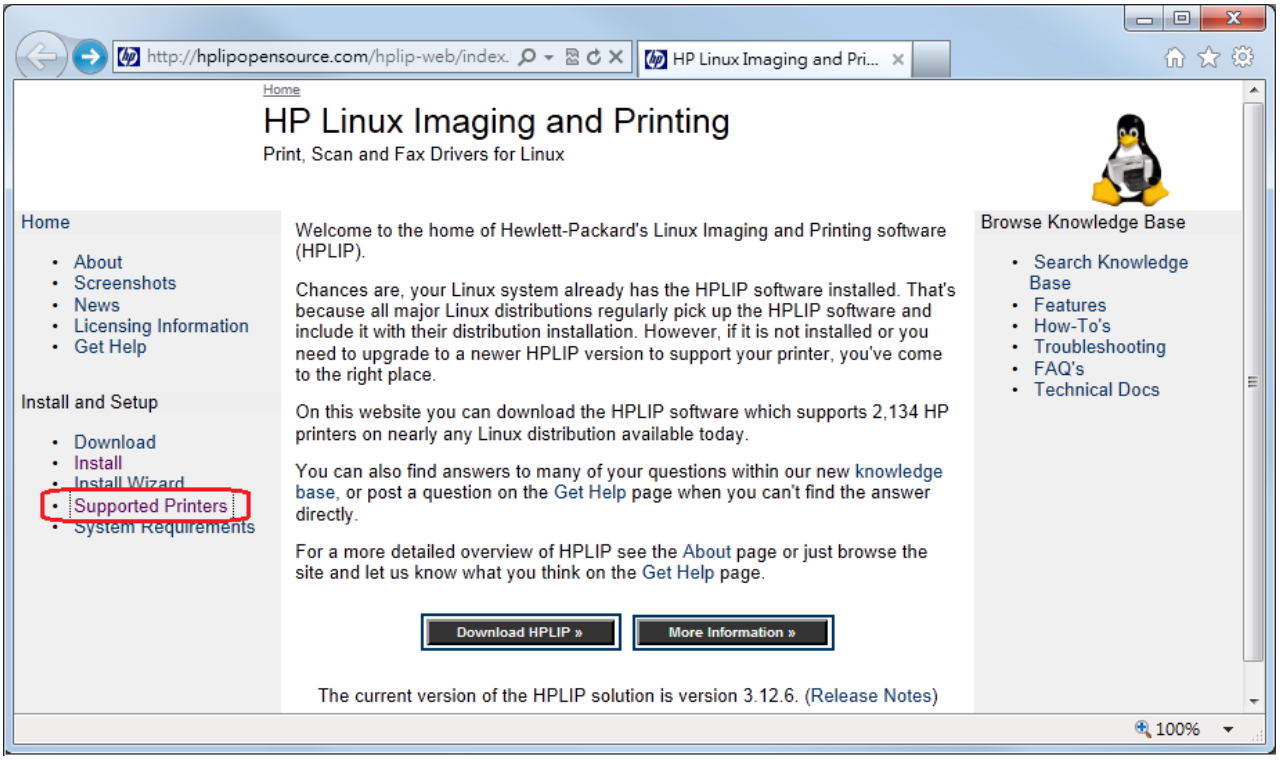
■ Remote Printer Server

Use EasyPrinter to start printing by the printers connected with PC via Ethernet. This works under MS Windows so the most printers on market are supported.

23.2 How to check the Supported HP Printer Types

1. Visit the website: http://hplipopensource.com/hplip-web/index.html

2. Click **[Install and Setup] » [Supported Printers]**.



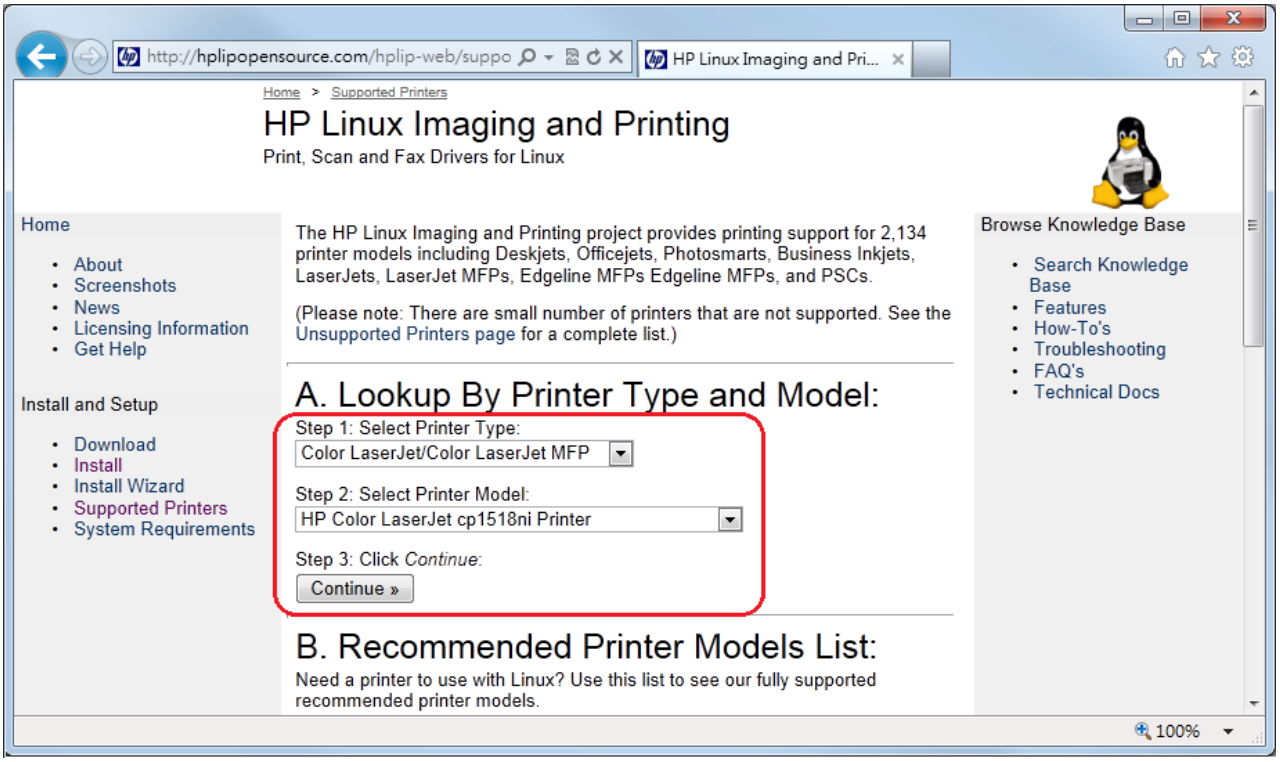
3. In “A. Lookup By Printer Type and Model”:

Select the printer type to be checked, ex: HP Color LaserJet cp1518ni Printer.

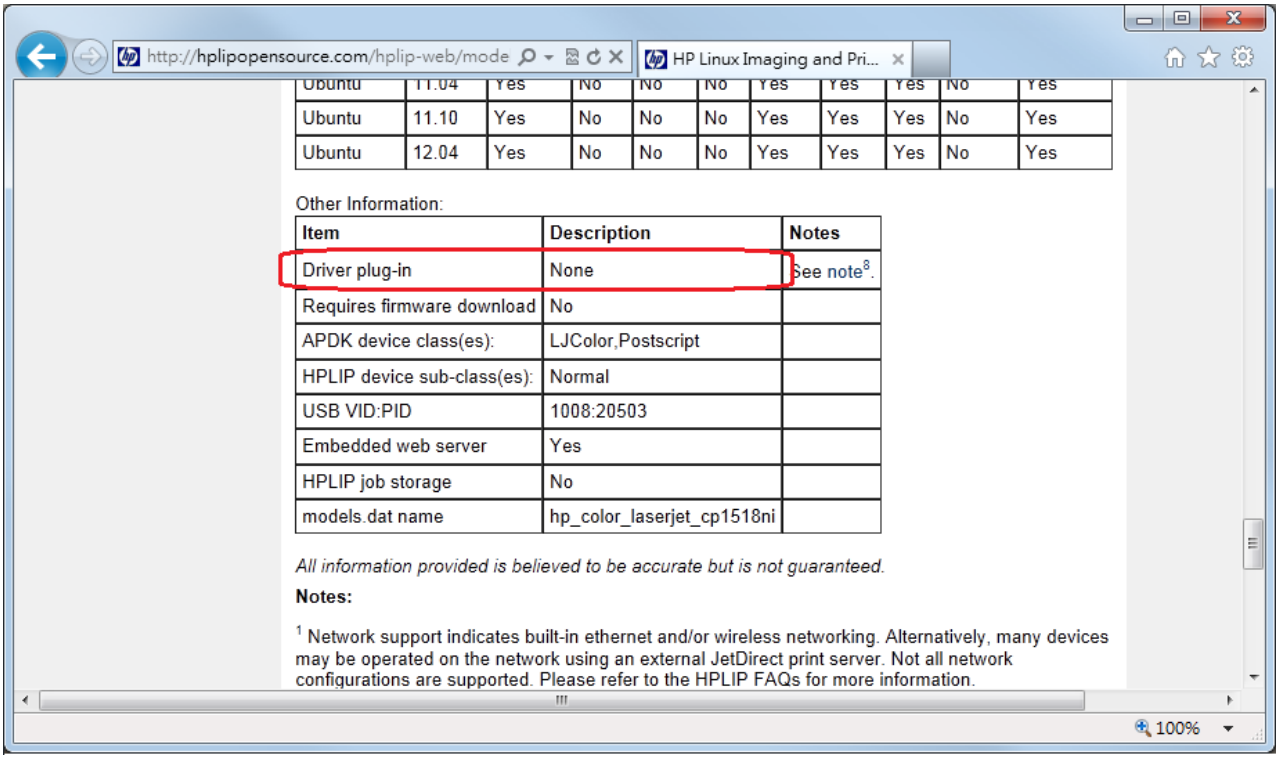
Step 1: Select Printer Type: Color LaserJet/Color LaserJet MFP.

Step 2: Select Printer Model: HP Color LaserJet cp1518ni Printer.

When finished, click **[Continue]**.



4. Find “Other Information”. The first item is “Driver plug-in”; its description must be **“None”,** which indicates that this HP printer is supported by HMI.



5. If Driver plug-in Description shows **"Required"** as below, this printer is not supported by HMI.

Other Information:

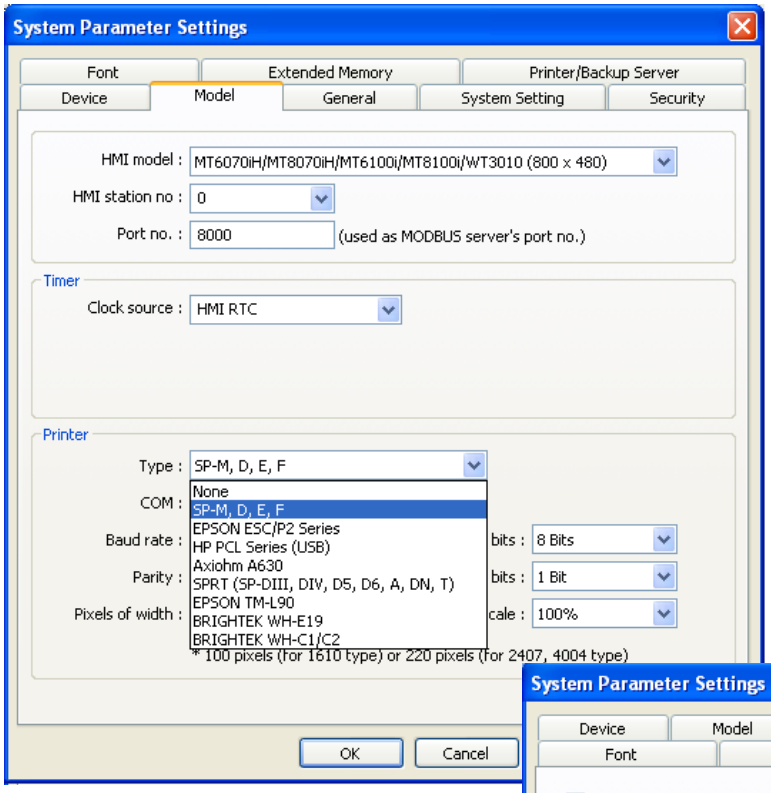
Item	Description	Notes
Driver plug-in	Required	See note[8].
Requires firmware download	No	
APDK device class(es):	LJColor,Postscript	
HPLIP device sub-class(es):	Normal	
USB VID:PID	1008:1834	
Embedded web server	No	
HPLIP job storage	No	
models.dat name	hp_laserjet_cm1415fn	

Information provided in this website is intended to be accurate and reliable. However, Weintek Labs., Inc. assumes no responsibility for its use.

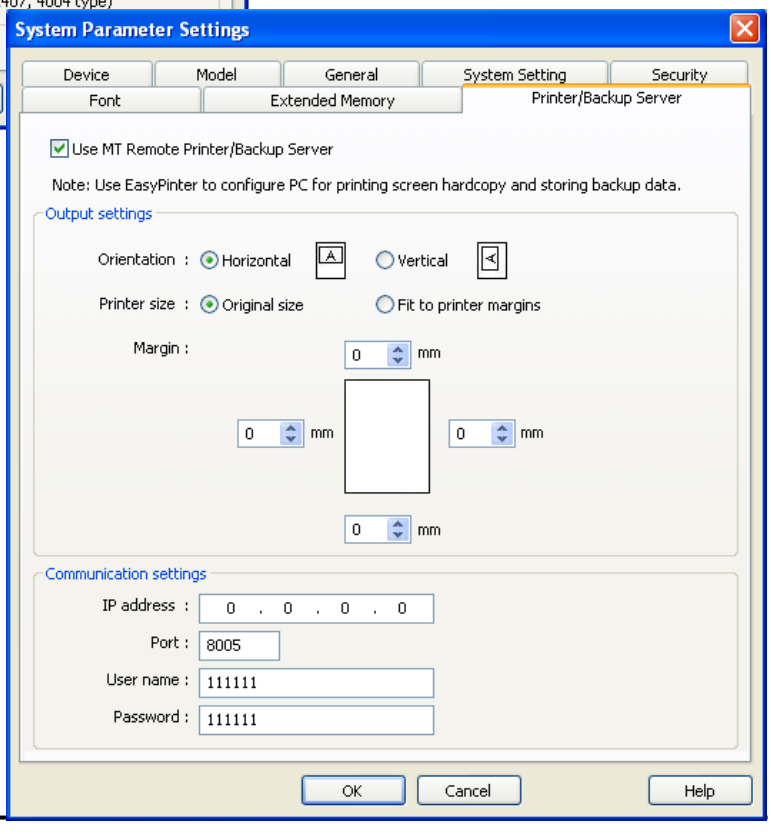
23.3 How to Add a New Printer and Start Printing

23.3.1 Add Printer Type

In **[System Parameter Settings] » [Model]** select the printer type and set the relevant parameters.

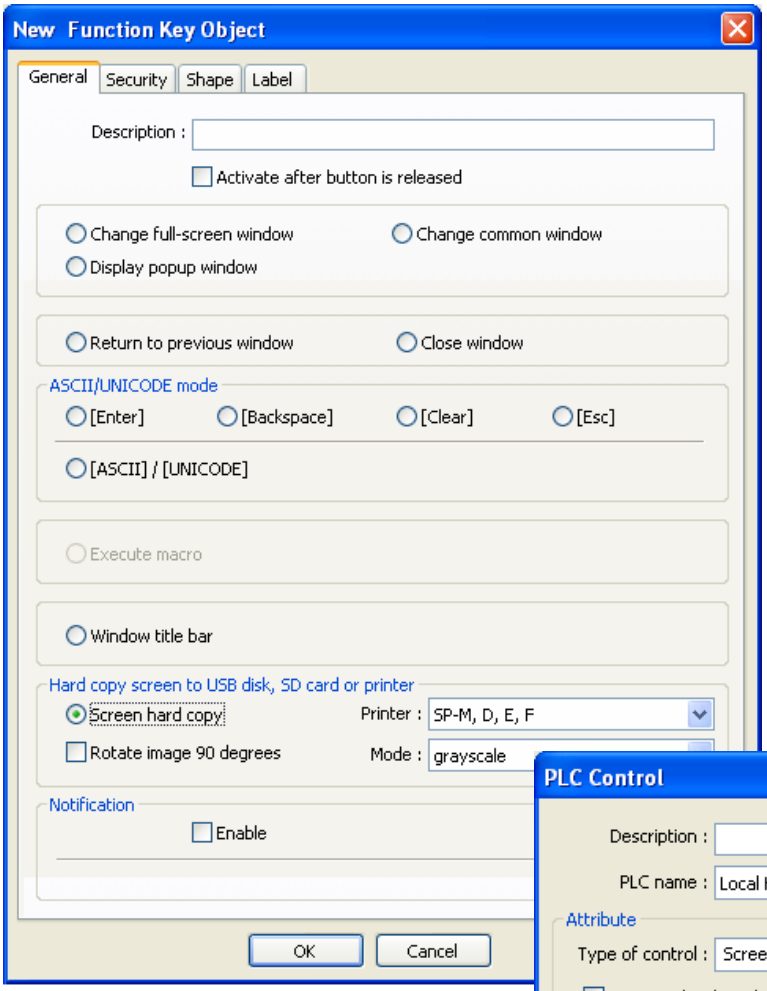


To connect Remote Printer Server, set the parameters in **[System Parameter Settings] » [Printer/Backup Server]**.

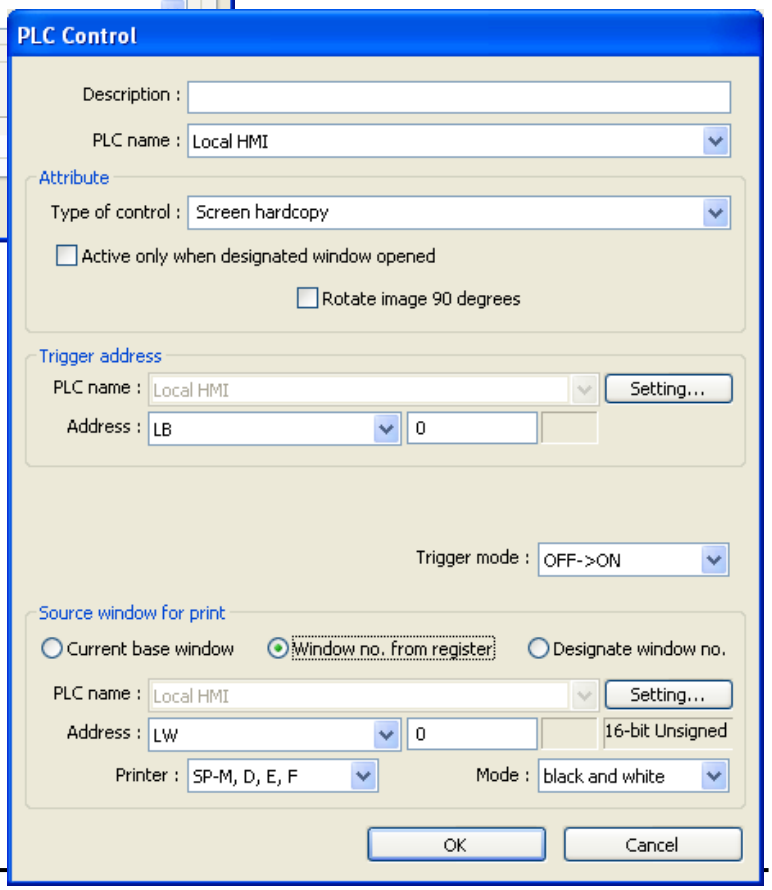


23.3.2 Start Printing

Start printing with Function Key.



Or, use PLC Control [Screen hardcopy] to start printing with a designated bit address.



Chapter 24 Recipe Editor

24.1 Introduction

Recipe Editor is used to create, view, and edit recipe data. Open Project Manager and click **[Recipe/Extended Memory Editor]** to start editing.

.

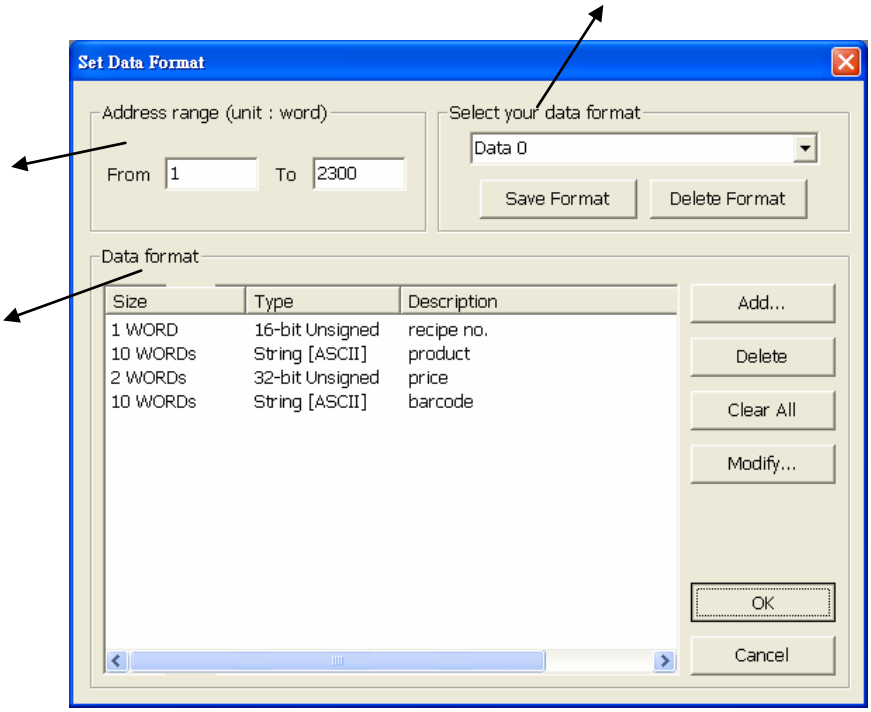
24.2 Recipe / Extended Memory Editor Setting

How to add new *.rcp / *.emi files?
Set Address Range » Select Data Format

[Select your data format]
Save the specified data format for loading next time. The saved file name: “dataEX.fmt” under EasyBuilder installation directory.

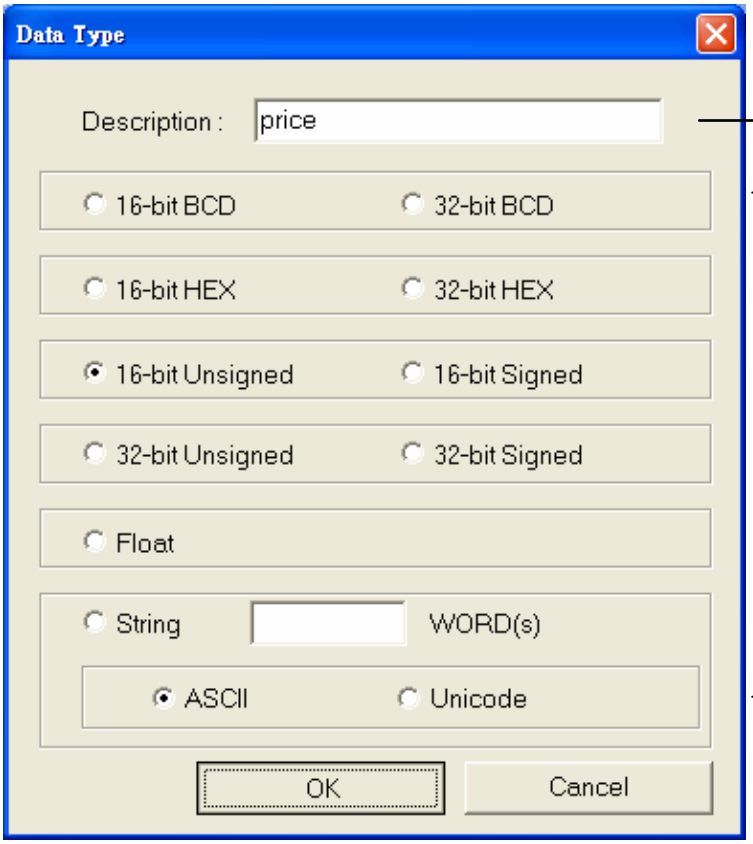
[Address range]
Fill in address range, the unit is word.

[Data format]
Edit new data format in this field.



Example 1

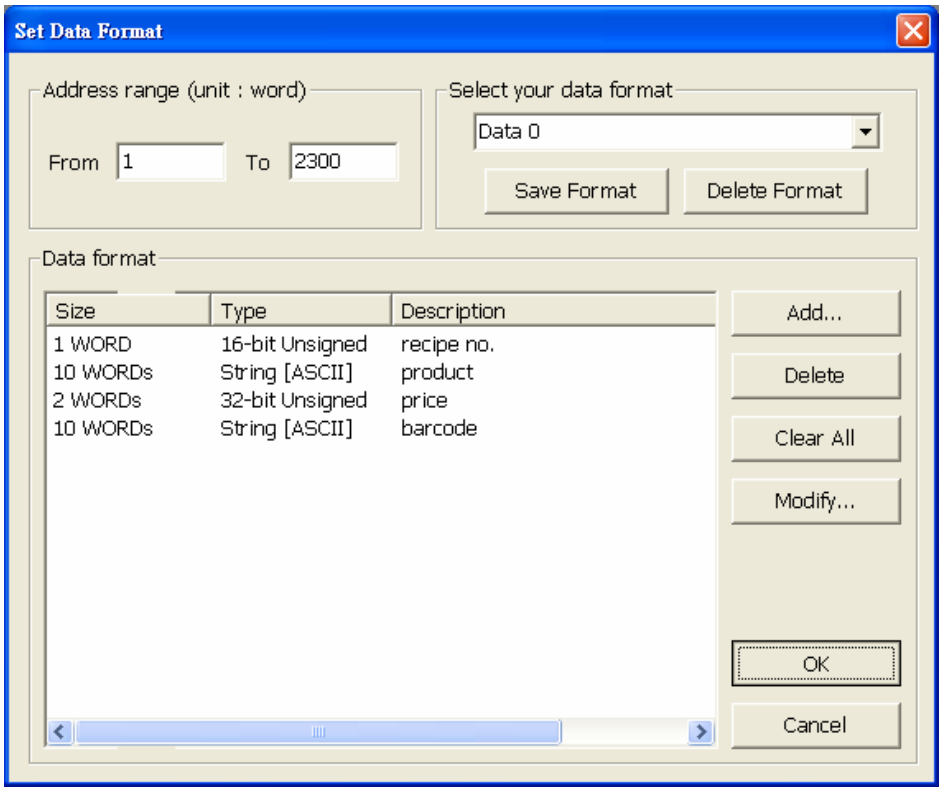
Step 1 Click **[Add]**, the popup dialog shows:

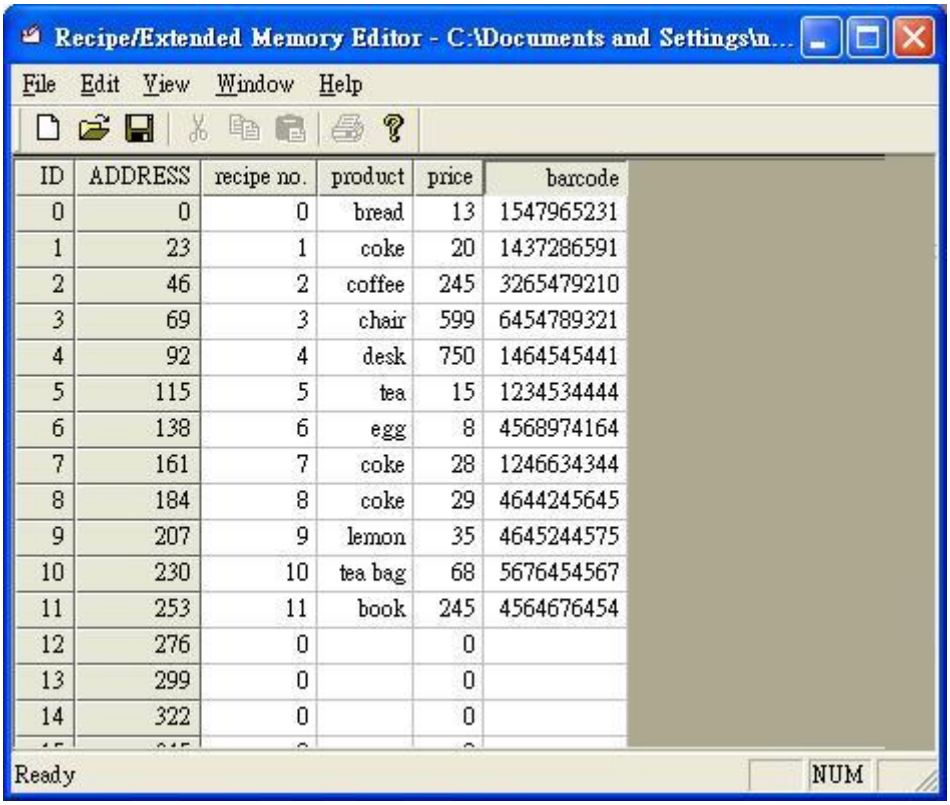


[Description]
Enter the name of the data type.

[Data format] Select data format. If select **[String]**, please enter the length (words) and select ASCII/Unicode.

Step 2 After setting, click **[OK]** to start editing recipe data.





ID	ADDRESS	recipe no.	product	price	barcode
0	0	0	bread	13	1547965231
1	23	1	coke	20	1437286591
2	46	2	coffee	245	3265479210
3	69	3	chair	599	6454789321
4	92	4	desk	750	1464545441
5	115	5	tea	15	1234534444
6	138	6	egg	8	4568974164
7	161	7	coke	28	1246634344
8	184	8	coke	29	4644245645
9	207	9	lemon	35	4645244575
10	230	10	tea bag	68	5676454567
11	253	11	book	245	4564676454
12	276	0		0	
13	299	0		0	
14	322	0		0	

Step 3

In this example, the total length of data format is 23 words. Each 23 words will be one set of recipe data.

The first set: “recipe no.” = address 0, “product” = address 1 ~ 10, “price” = address 11 ~ 12, “barcode” = address 13 ~ 22;

The second set: “recipe no.” = address 23, “product” = address 24 ~ 33, “price” = address 34 ~ 35, “barcode” = address 36 ~ 45…and so on.

- After editing recipe data, it can be saved as *.rcp, *.emi, or *.csv. The *.rcp files can be downloaded to HMI using Project Manager or external devices (USB disk or SD card). The *.emi files can be saved directly to the external device and insert the device to HMI to be the extended memory (EM).

Chapter 25 EasyConverter

EasyConverter reads the data sampling file (*.dtl) and event log file (*.evt) in HMI and convert the files to Excel format.

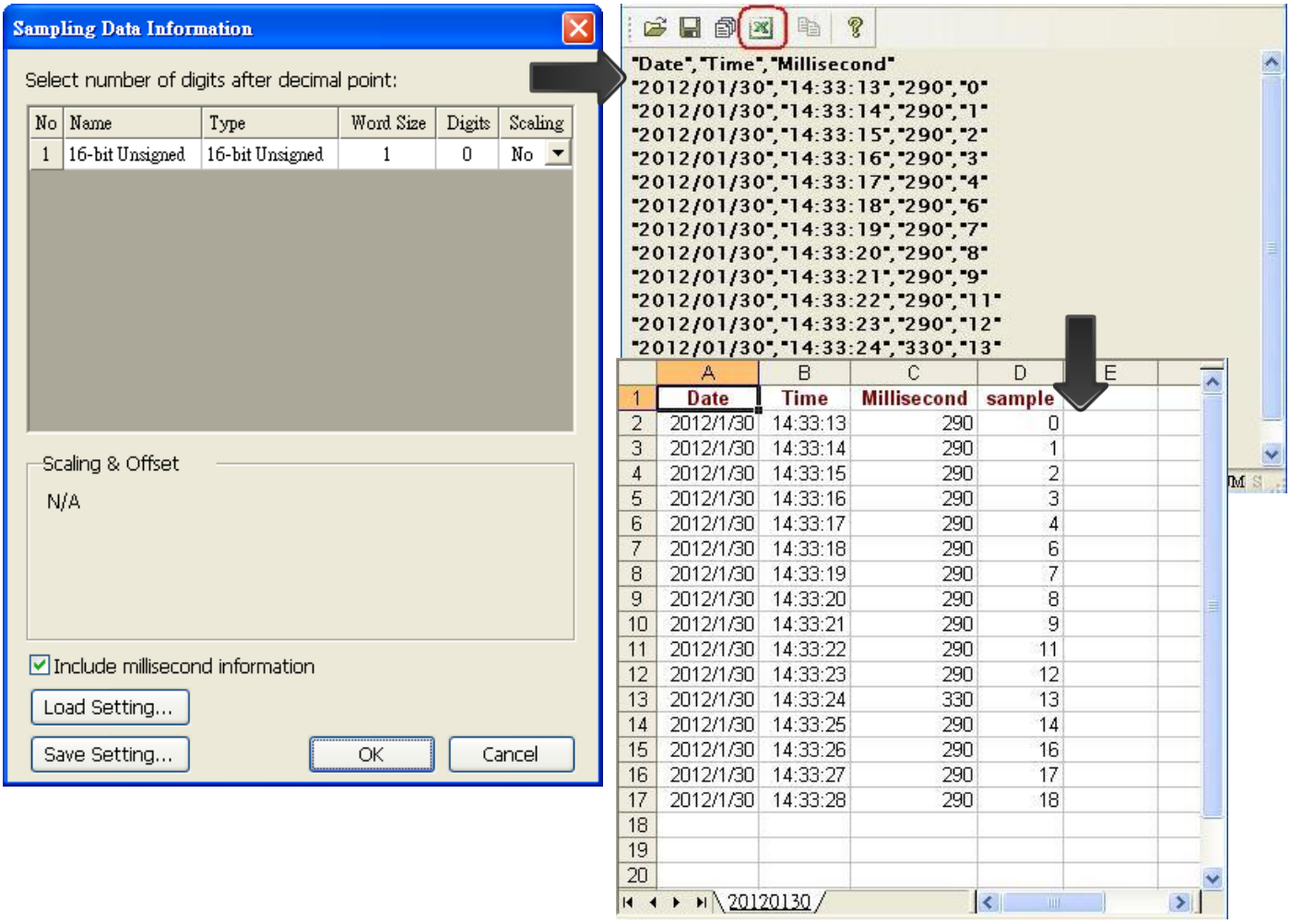
How to launch EasyConverter:

■ From Project Manager click EasyConverter.

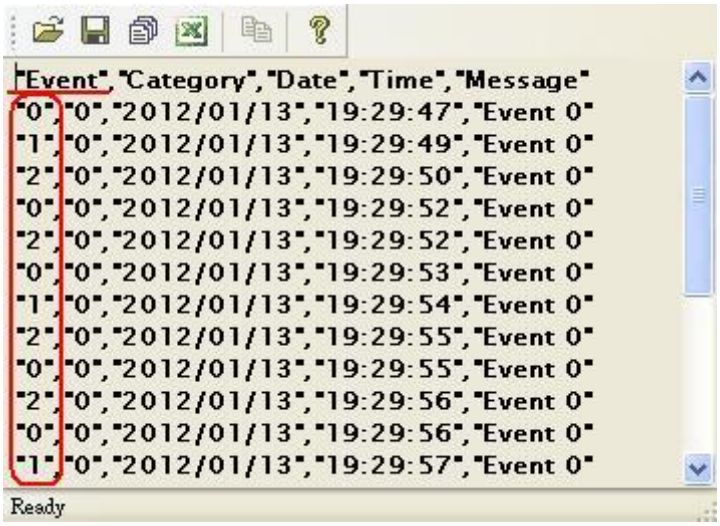
■ From EasyBuilder 8000 menu. Select **[Tool] » [Data/Event Log Converter].**

25.1 How to Export DTL or EVT file to Excel

1. When opening a data sampling file (*.dtl), a pop up window shows as below.
2. Click **[OK]**, then click **[Export to Microsoft Excel]**.



When opening an event log file, an **[Event]** column can be found as shown below.
0 → Event triggered; **1** → Event acknowledged; **2** → Event returns to normal.



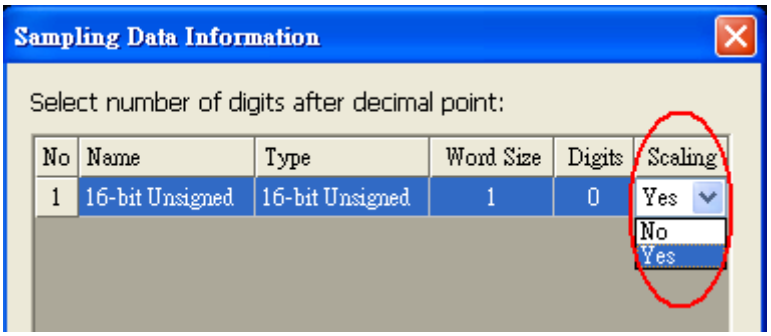
- Double click on a dtl or evt file will automatically generate an Excel file. However, if the strings in dtl file are in UNICODE, please open EasyConverter to convert the file manually.

25.2 Scaling Function

The equation of scaling:

new value = [(value + A) x B] + C, users can set the values of A, B, and C.

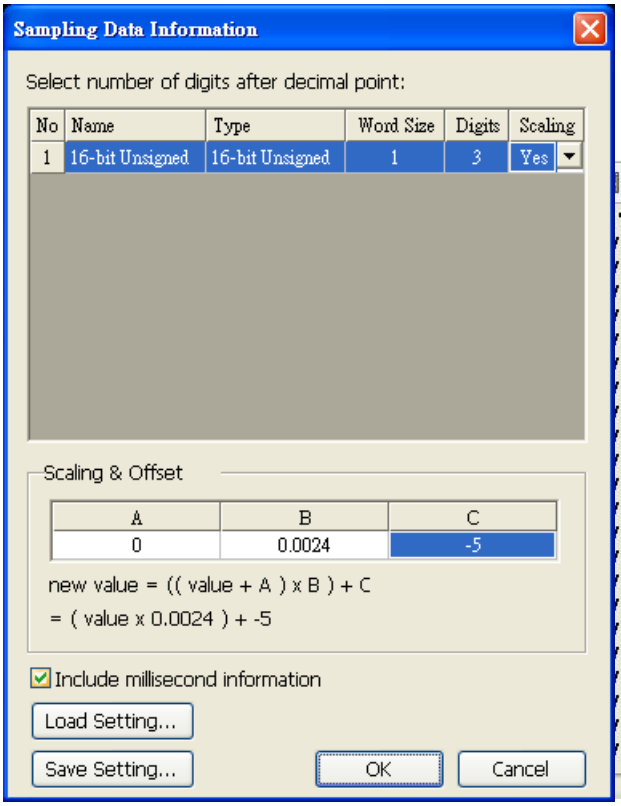
A: lower limit of the value; B: [(scaled max.) - (scaled min.)] / [(upper limit) - (lower limit)]; C: scaled min.



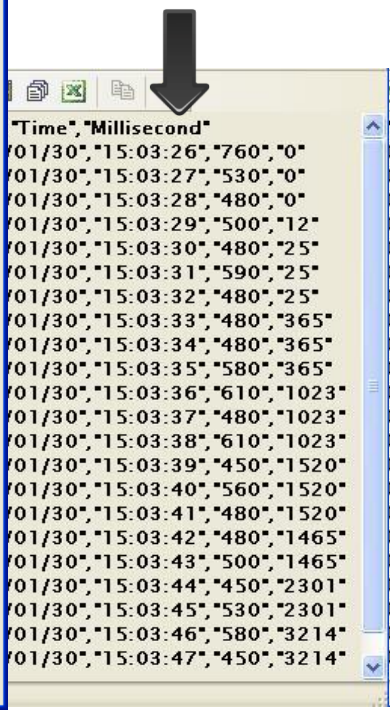
Example 1

For example, here is a voltage data, the format is 16-bit unsigned (range: 0 ~ 4096).

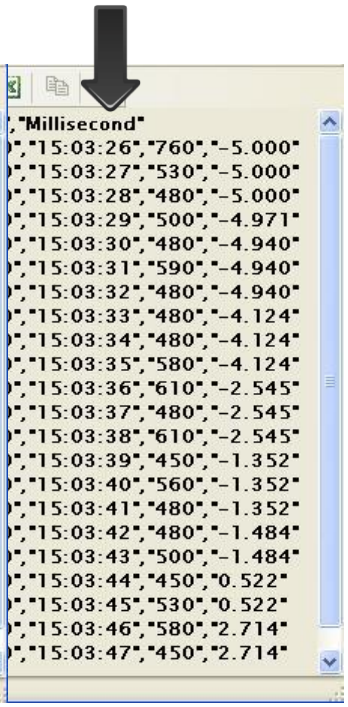
To convert the data to volt, range: -5V ~ +5V: New value = [(value + 0) x 0.0024] + (-5):



Before scaling



After scaling



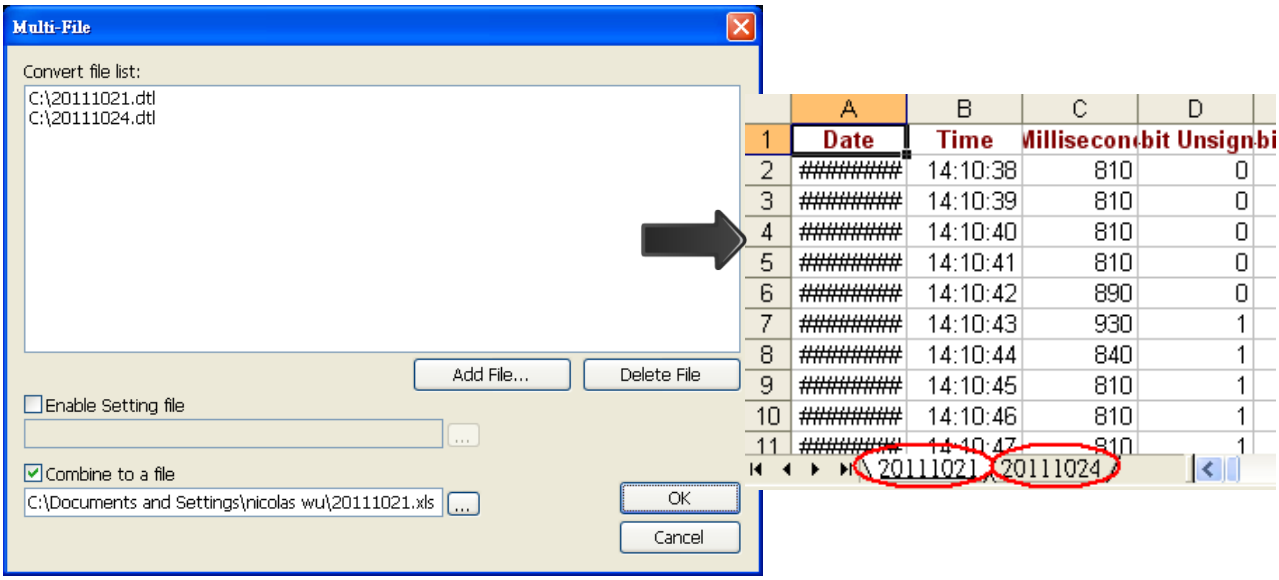
■ The settings above can be saved and loaded next time. The extension name of the setting file: *.lgs.

■ After setting the scaling values, click **[Save Setting]**. In the new Sampling Data Information dialog, click **[Load Setting]** to use the settings saved before.

25.3 How to Use Multi-File Conversion

Example 1

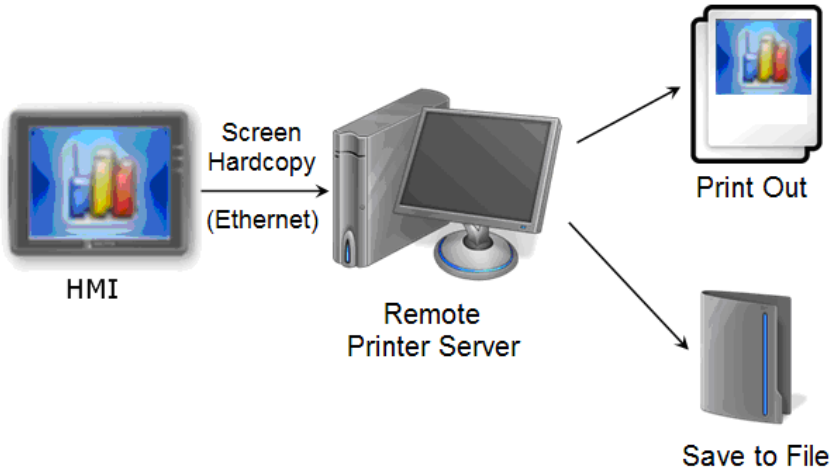
1. Click **[File] » [Multi-File] » [Add File]** to combine multiple files into one Excel (*.xls) file.
2. Click **[Combine to a file]**, files will be separated into sheets of one Excel (*.xls) file labeled with the dated it is created. If this check box is not selected, the files will be exported to Excel respectively.



The saved setting files can be loaded for combining:
Select both **[Enable Setting file]** and **[Combine to a file]** check boxes and select the files to be combined then click **[OK]**.

Chapter 26 EasyPrinter

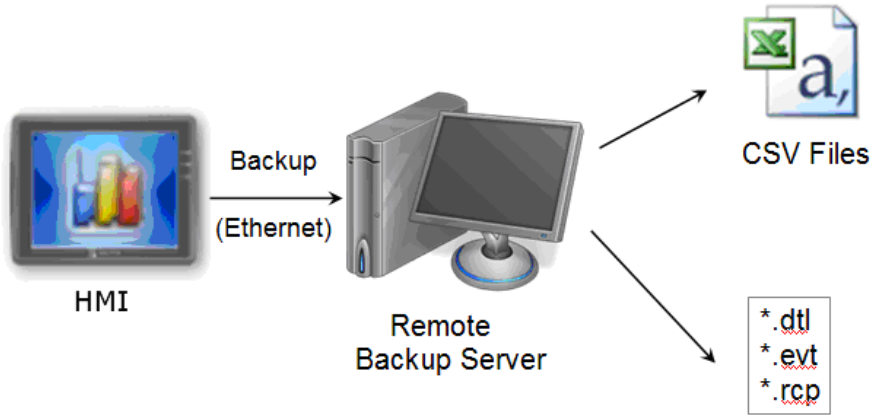
EasyPrinter is a Win32 application and can only run on MS Windows 2000 / XP / Vista / 7. It enables HMI to output screen hardcopies to a remote PC via Ethernet. The following explains how to use EasyPrinter.



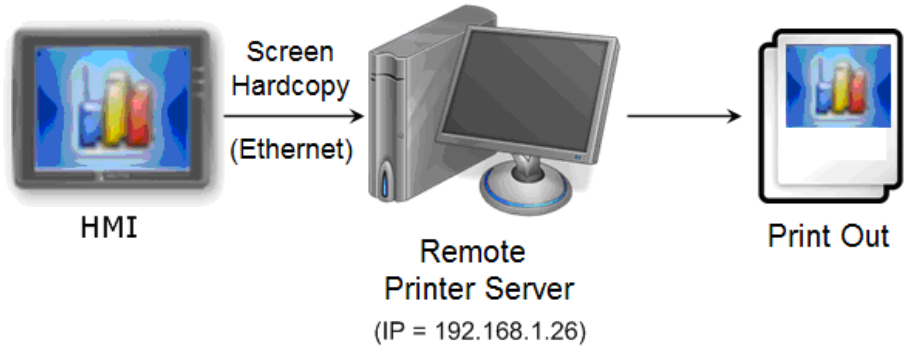
Here are some advantages of using EasyPrinter:

- EasyPrinter provides two modes of hardcopy output: [Print Out] and [Save to File]. Users can use either way or both ways.
- Since EasyPrinter runs on MS Windows system, it supports most of the printers available on the market.
- Multiple HMIs can share one printer so users don't have to prepare printers for each HMI.

Additionally, EasyPrinter can also be a backup server. Users can use **[Backup]** objects in HMI to copy history files such as Data Sampling and Event Log into a remote PC via Ethernet. Please see the following illustration:



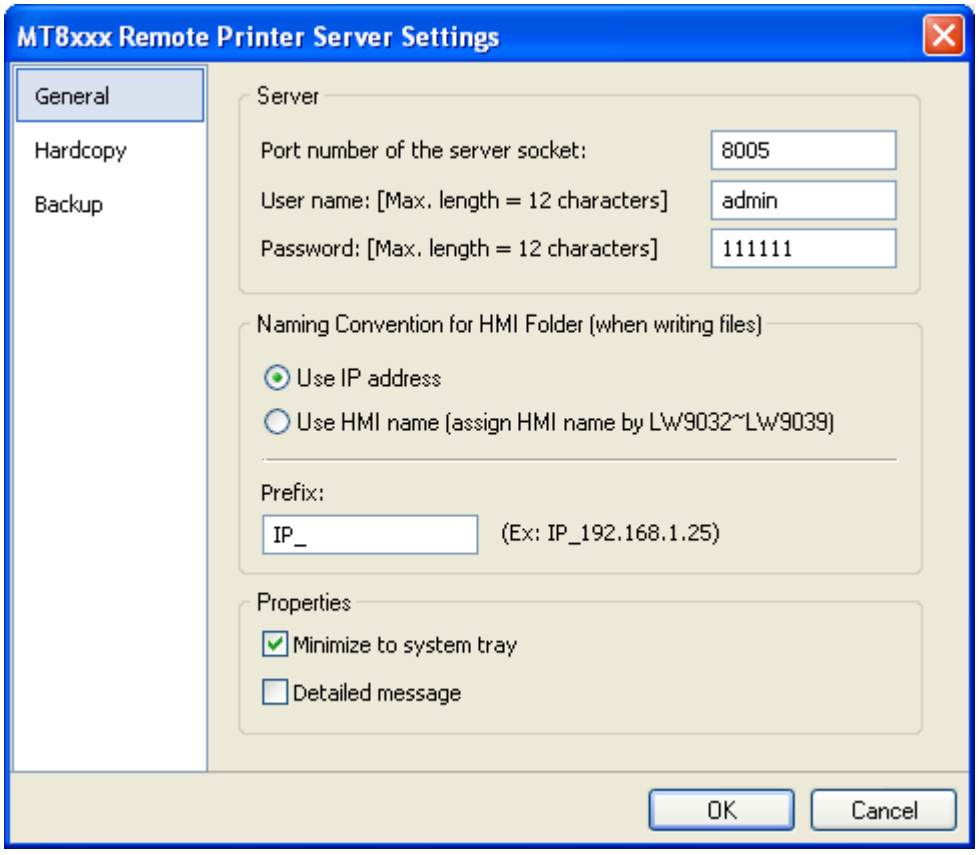
26.1 Using EasyPrinter as a Printer Server



Users can make screen hardcopies with a **[Function Key]** object. The hardcopies will be transferred to the Remote Printer Server via Ethernet and then printed out.

26.1.1 Setup Procedure in EasyPrinter

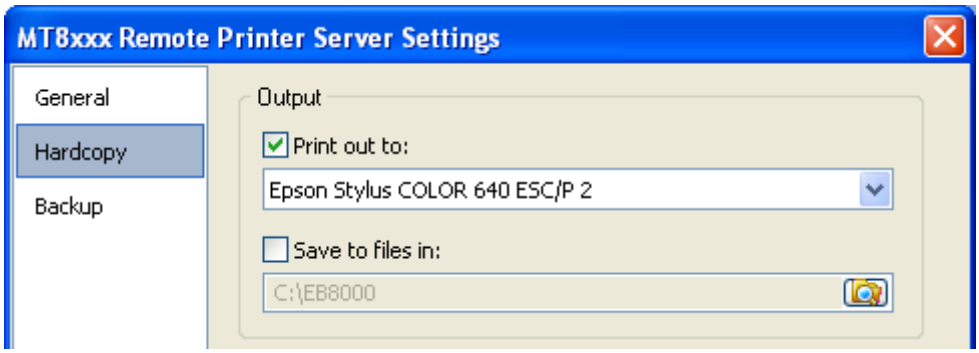
In EasyPrinter main menu select **[Options] » [Settings]** and the following dialog appears:



1. Select **[General]** on the left side.
2. In **[Server]**, set **[Port number of the server socket]** to “8005”, **[User name]** to “admin” and **[Password]** to “111111”. (Note: These are default values.)
3. In **[Naming Convention for HMI Folder]**, select **[Use IP address]** and enter “IP_” in the **[Prefix]** field.

4. In **[Properties]**, select **[Minimize to system tray]**.

Set the print out location:

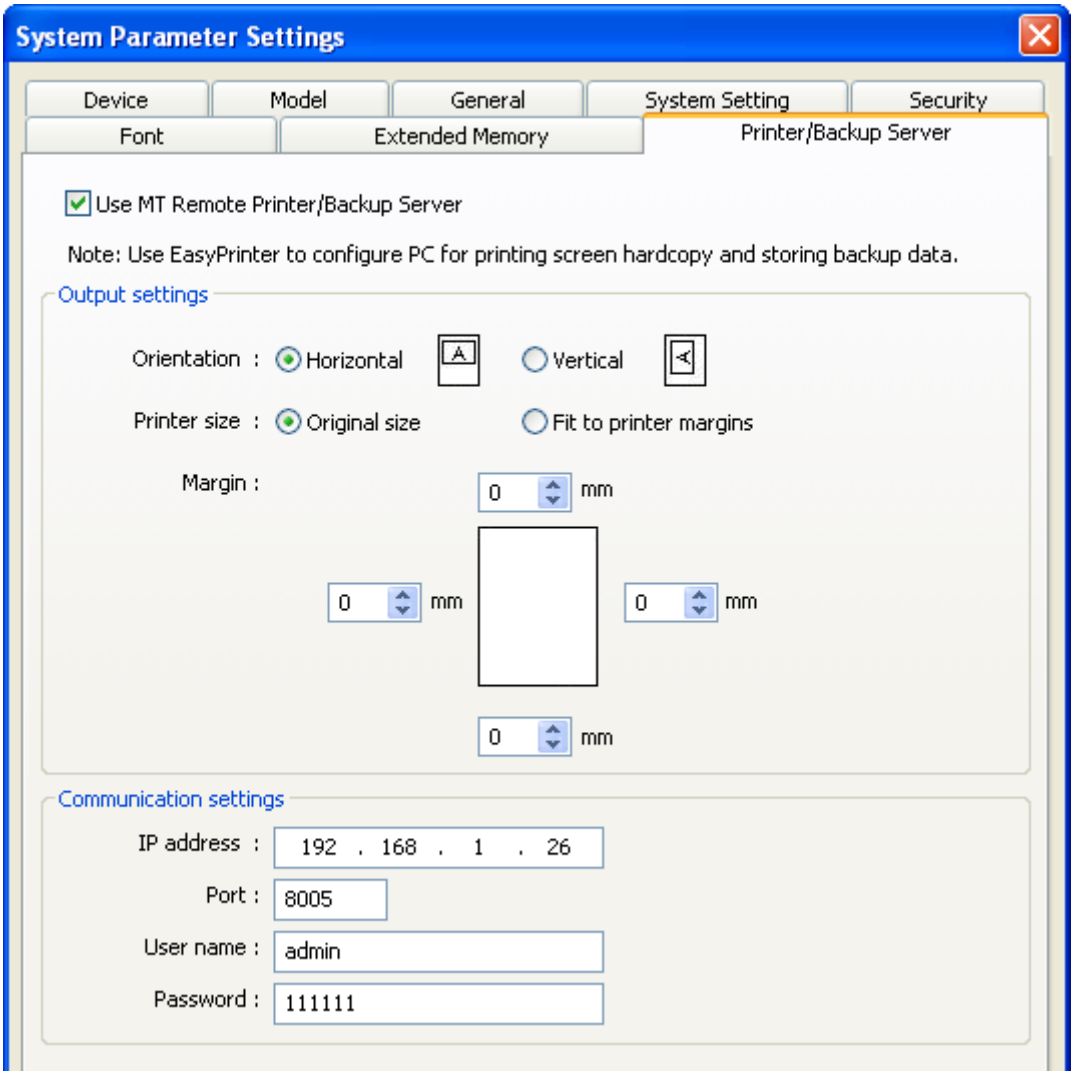


1. Select **[Hardcopy]** on the left side.
2. In **[Output]**, select **[Print out to]** and choose a printer as the output device for screen hardcopies. (Note: the printer shown in the image above is an example; please select an actual printer located in your private network environment.)
3. Click **[OK]** to confirm the settings.
4. In EasyPrinter main menu select **[File] » [Enable Output]** to output any incoming print request.

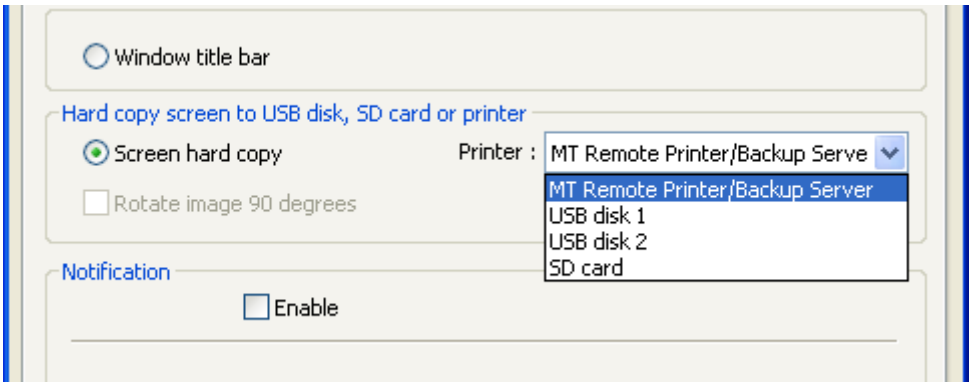
26.1.2 Setup Procedure in EasyBuilder

The setting procedure of EasyPrinter in EasyBuilder:

1. Open a new project or an existing project in EasyBuilder.
2. In EasyBuilder main menu select **[Edit] » [System Parameter Settings] » [Printer/Backup Server]** and select **[Use MT Remote Printer/Backup Server]** check box.



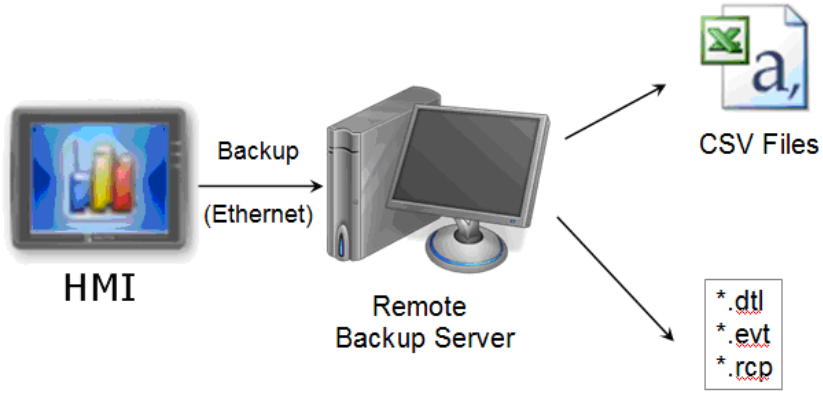
3. In **[Output settings]**, set appropriate values for left / top / right / bottom margins. (Note: The margins are all set to 15mm in the example.)
4. In **[Communication settings]**, fill in the **[IP address]** of the printer server following the settings in EasyPrinter. Set the **[port number]** to “8005”, **[User name]** to “admin” and **[Password]** to “111111”.
5. Click **[OK]**.
6. In EasyBuilder main menu select **[Objects] » [Button]**, select **[Function Key]**, select **[Screen hardcopy]** and set **[Printer]** to **[MT Remote Printer/Backup Server]**.



7. Place the **[Function Key]** object in the common window (window no. 4) for screen hardcopies anytime.
8. **[Compile]** and **[Download]** project to HMI. Press the **[Function Key]** object in the screen to make a screen hardcopy.

- A **[PLC Control]** object can also make screen hardcopies.
- Alarm information cannot be printed via EasyPrinter.
- EasyPrinter can only communicate with HMI via Ethernet, please check the HMI located in private network environment.

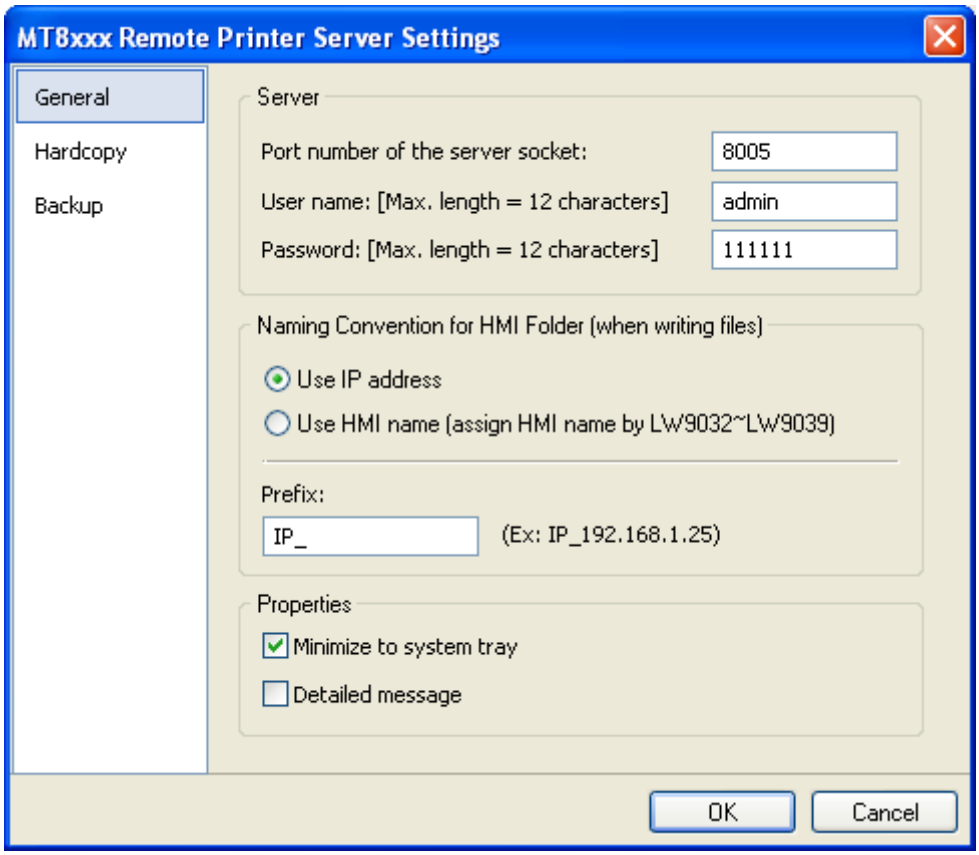
26.2 Using EasyPrinter as a Backup Server



[Backup] objects can upload historical data such as Data Sampling and Event Log history files to MT remote backup server.

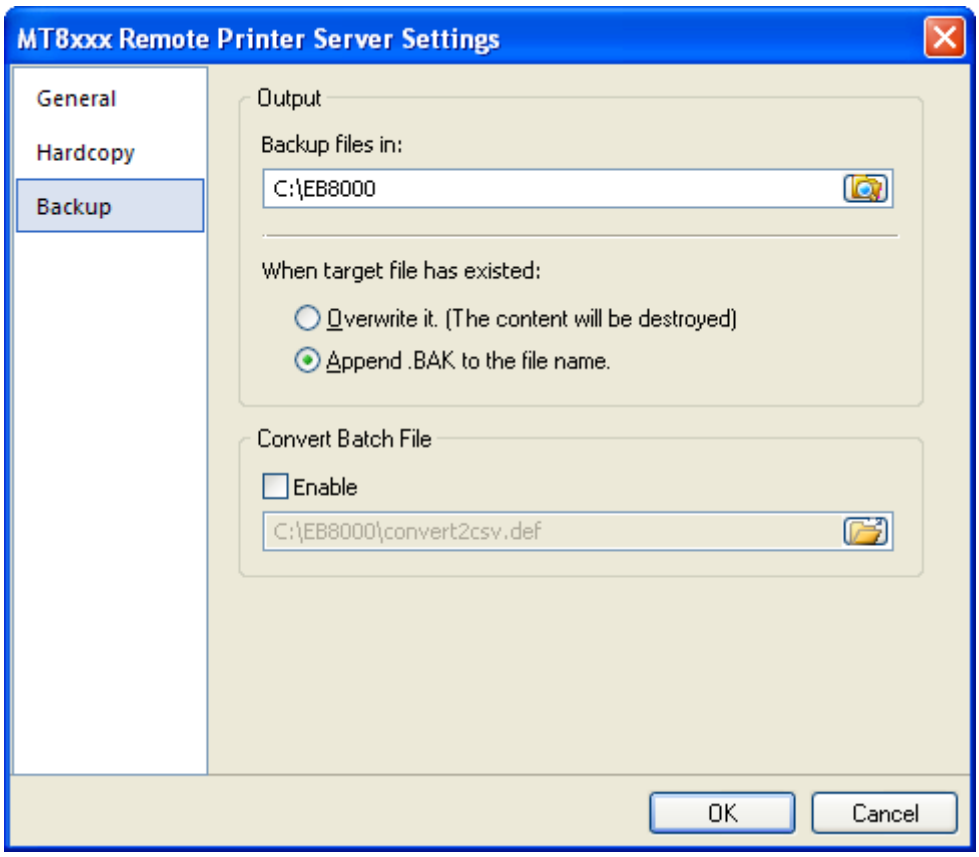
26.2.1 Setup Procedure in EasyPrinter

In EasyPrinter main menu select **[Objects] » [Settings]** and the following dialog appears:



1. Select **[General]** on the left side.
2. In **[Server]**, set **[Port number of the server socket]** to “8005”, **[User name]** to “admin” and **[Password]** to “111111”. (Note: These are default values.)
3. In **[Naming Convention for HMI Folder]**, select **[Use IP address]** and enter “IP_” in the **[Prefix]** field.
4. In **[Properties]**, select **[Minimize to system tray]**.

Set the backup location.

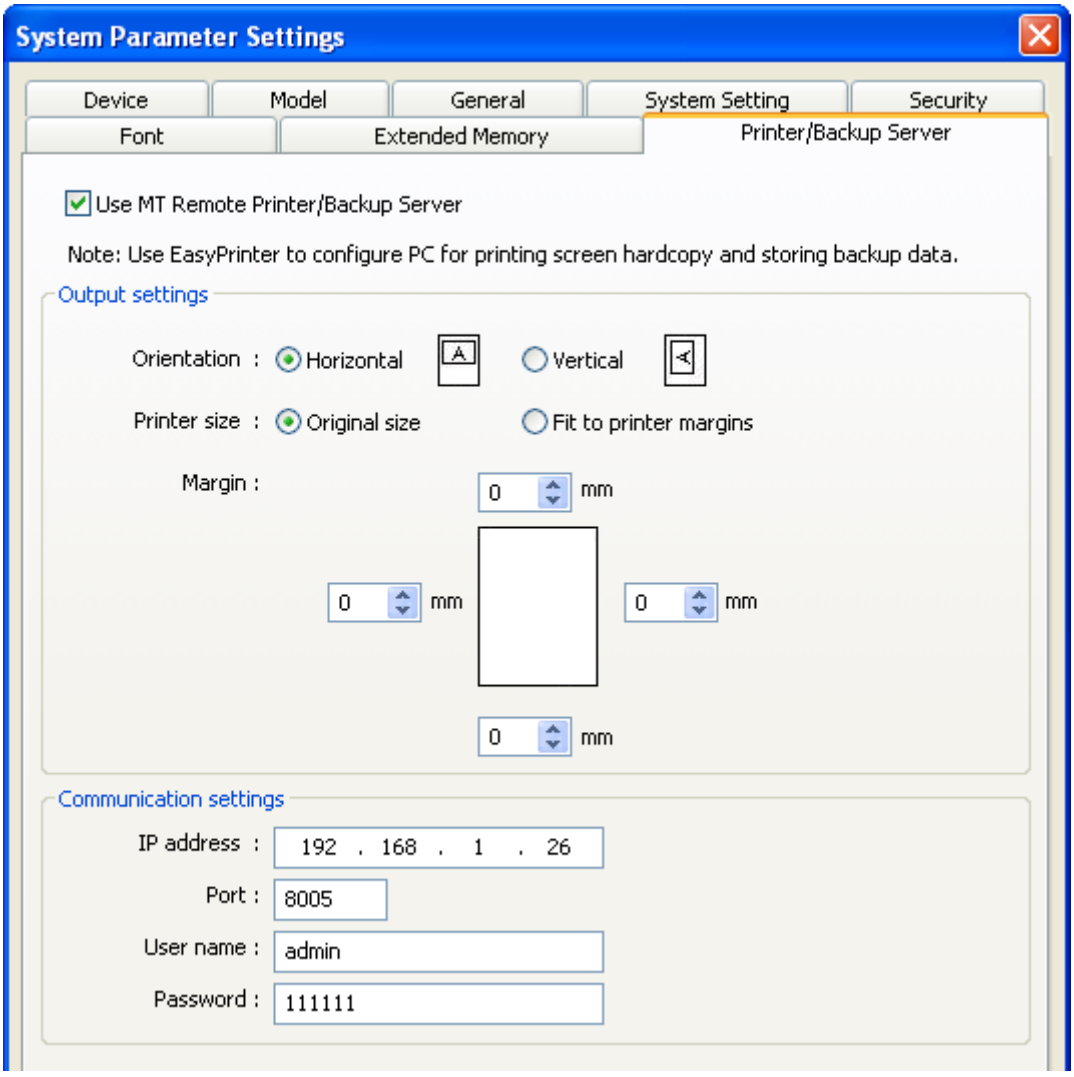


1. Select **[Backup]** on the left side.
2. In **[Output]**, click on button to browse and select a storage directory of the incoming history files.
3. Click **[OK]** to confirm the settings.
4. In EasyPrinter main menu select **[File] » [Enable Output]** to backup data in the selected directory.

26.2.2 Setup Procedure in EasyBuilder

The setting procedure of EasyPrinter :

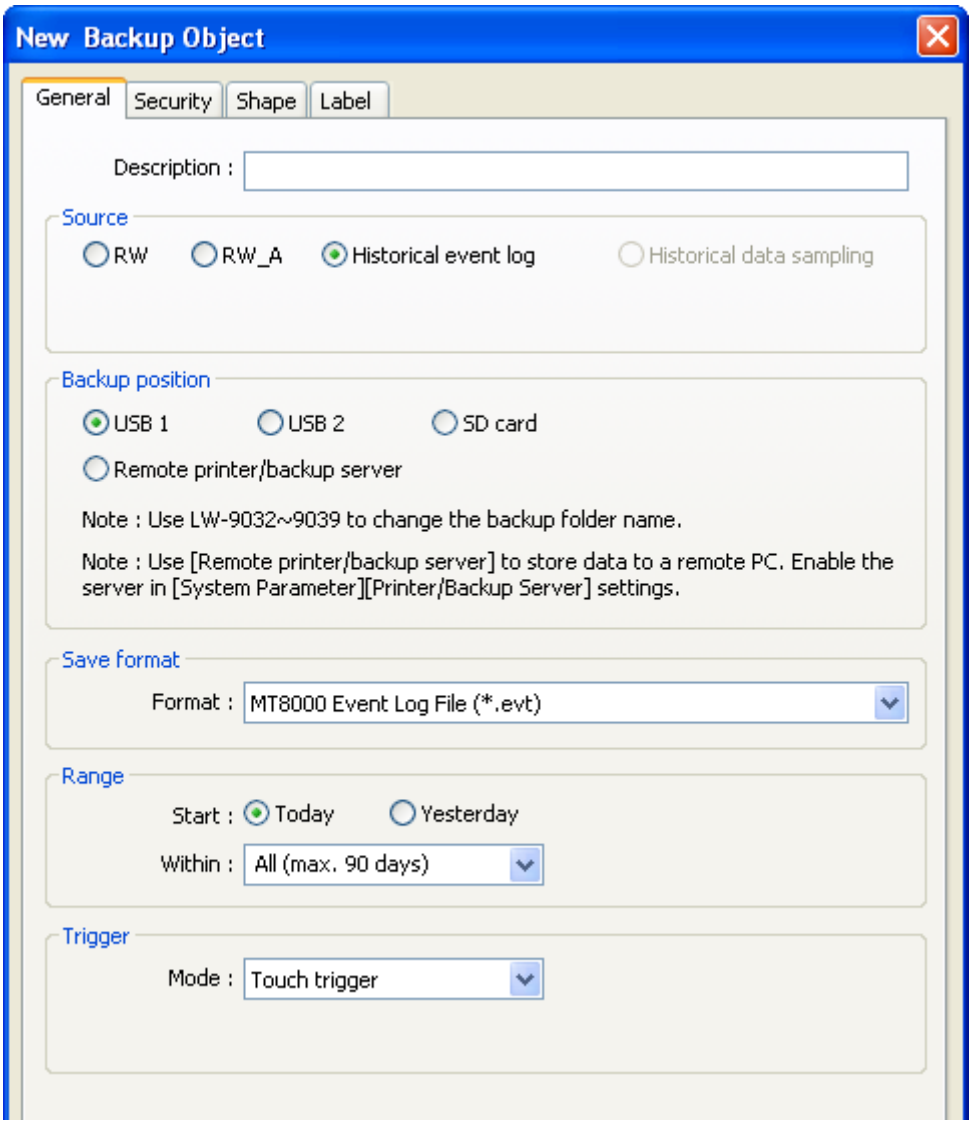
1. Open a new project or an existing project in EasyBuilder.
2. In EasyBuilder main menu select **[Edit] » [System Parameter Settings] » [Printer/Backup Server]** and select **[Use MT Remote Printer/Backup Server]** check box.



3. In **[Communication settings]**, fill in the **[IP address]** of the printer server following the settings in EasyPrinter. Set the **[port number]** to “8005”, **[User name]** to “admin” and **[Password]** to “111111”. (Note: These are default values.)
4. Click **[OK]**.

Create a Backup object.

1. In EasyBuilder main menu select **[Objects] » [Backup]** and the following dialog appears:



2. In **[Source]**, select **[Historical event log]** (or **[RW]**, **[RW_A]** if needed.)
3. In **[Backup position]**, select **[Remote printer/backup server]**.
4. In **[Range]**, select **[Today]** and **[All]** (or other options if needed.)
5. In **[Trigger]**, select **[Touch trigger]**.
6. Click **[OK]**.
7. Place the **[Backup]** object in the common window (window no. 4), and users will be able to make backups anytime needed.
8. **[Compile]** and **[Download]** project to HMI. Press the **[Backup]** object in the screen to make a backup of the history data.

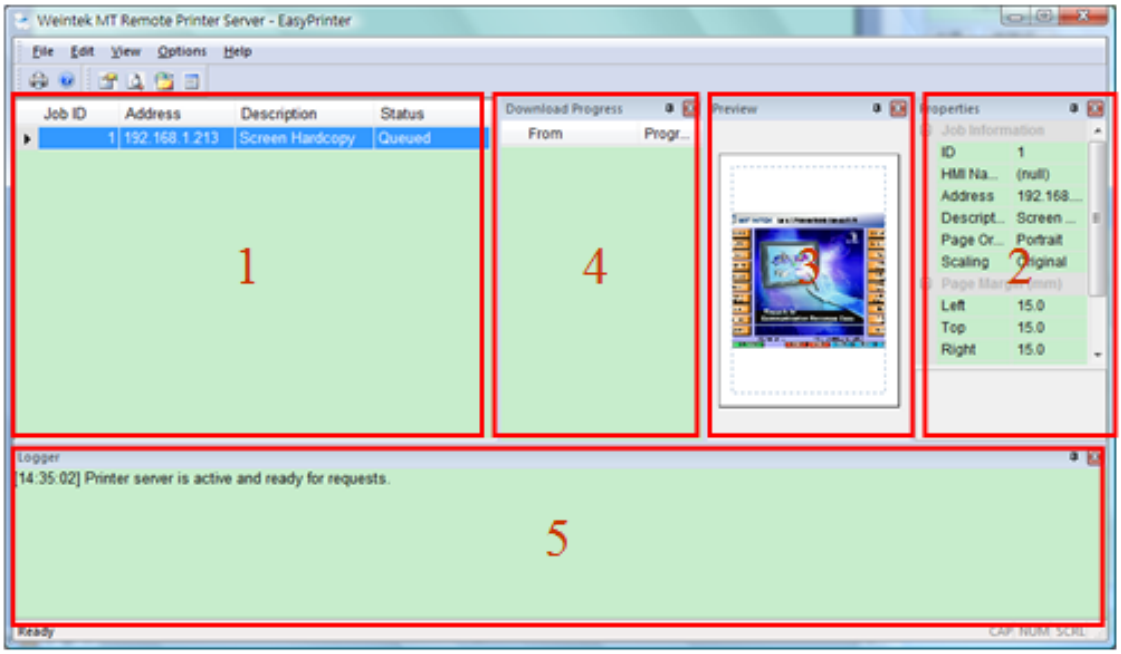
- The **[Backup]** object can be triggered via a bit address.
- Users can arrange a **[Scheduler]** object, which turns a bit ON at the end of the week, to trigger the **[Backup]** object to automatically back up all history data.

26.3 EasyPrinter Operation Guide

The following introduces the interface and operation of EasyPrinter.

26.3.1 Appearance

EasyPrinter main menu is divided into 5 parts as shown below:



The following introduces the functions:

Area	Name	Description
1	Job List	Lists all incoming tasks, i.e. screen hardcopy and backup requests.
2	Property Window	Shows the information about the task selected from [Job List].
3	Preview Window	Shows the preview image of the screen hardcopy task selected from [Job List].
4	Download Progress Window	Shows the download progress of incoming requests.
5	Message Window	Shows the time and message information of events such as incoming request, incorrect password, etc.

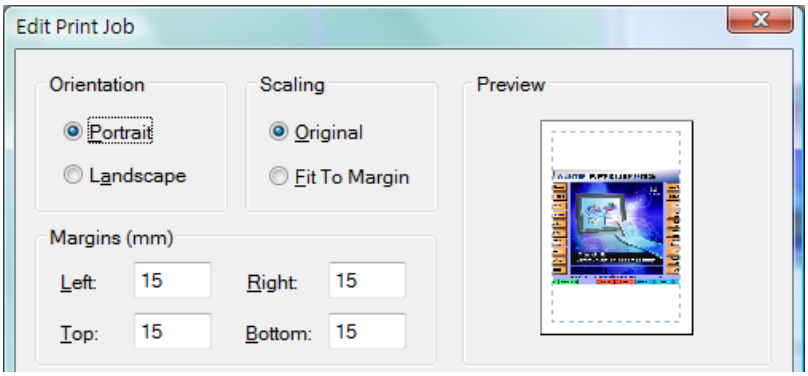
26.3.2 Operation Guide

The following describes the function of EasyPrinter menu items.

- **[File] » [Enable Output]**
  Selected: EasyPrinter processes the tasks one by one.
  Unselected: EasyPrinter stores the tasks in memory.

- EasyPrinter can only reserve up to 128 MB of task data in memory. If the memory is full, any request coming in afterwards will be rejected. Users must either operate [Enable Output] or delete some tasks to make room for new tasks.

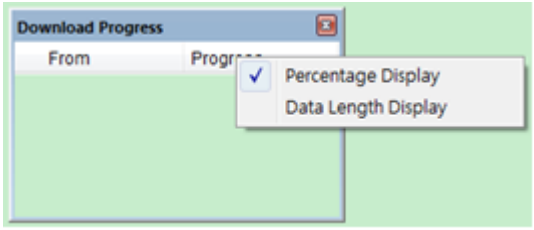
- **[Edit] » [Edit]**
  Edit screen hardcopy by setting **[Orientation]**, **[Scaling]** and **[Margins]**.



- **[Edit] » [Delete]**
  Delete the selected tasks permanently.
- **[Edit] » [Select All]**
  Select all tasks from [Job List].

- The backup task is not editable.
- [Edit] is available only when a task is selected.
- [Delete] is available when at least one task is selected.

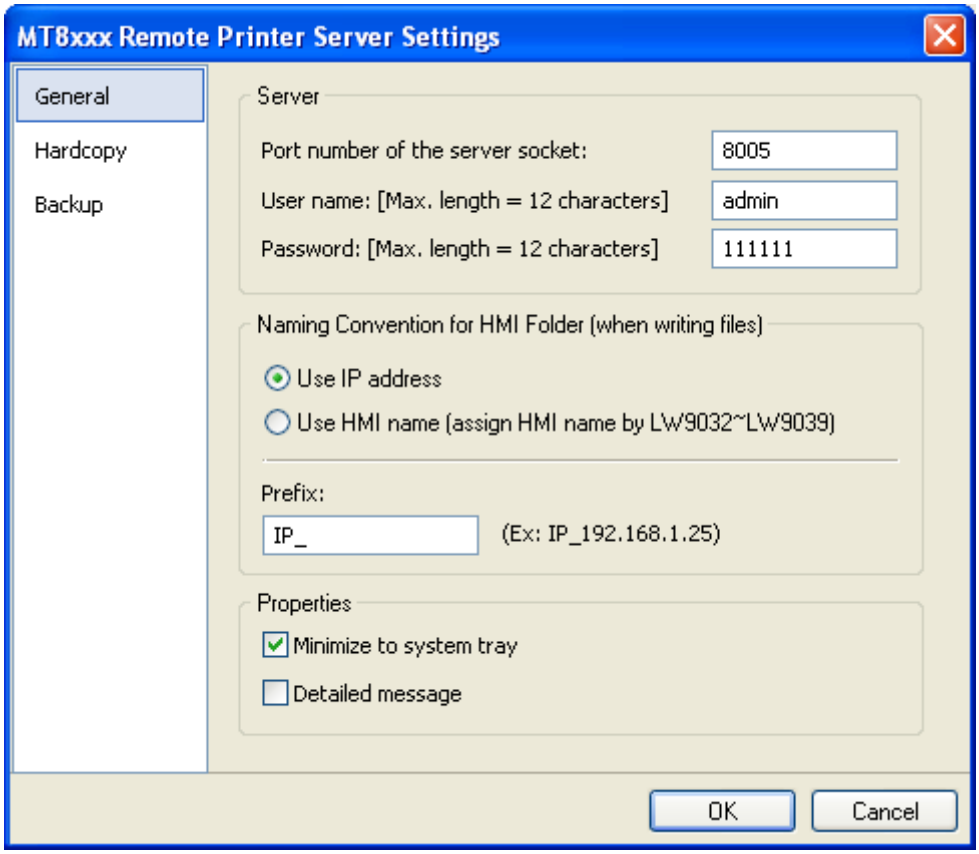
- **[View] » [Properties Bar]**
  Show or hide the Property Window.
- **[View] » [Preview Bar]**
  Show or hide the Preview Window.
- **[View] » [Download Bar]**
  In **[Download Progress]** Window, the mode to display download progress can be set by clicking the header of the **[progress]** column as shown below:



- **[View] » [Logger Bar]**
  EasyPrinter can reserve up to 10,000 messages in Message Window. If a new message comes in, the oldest message will be deleted.

The following is the detail for **[Options] » [Settings]**

1. [General]:



- **Server**

  [Port number of the server socket]

  Set the Ethernet port number to connect HMI. The range is 1 to 65535 and 8005 is the default value.

  [User name] & [Password]

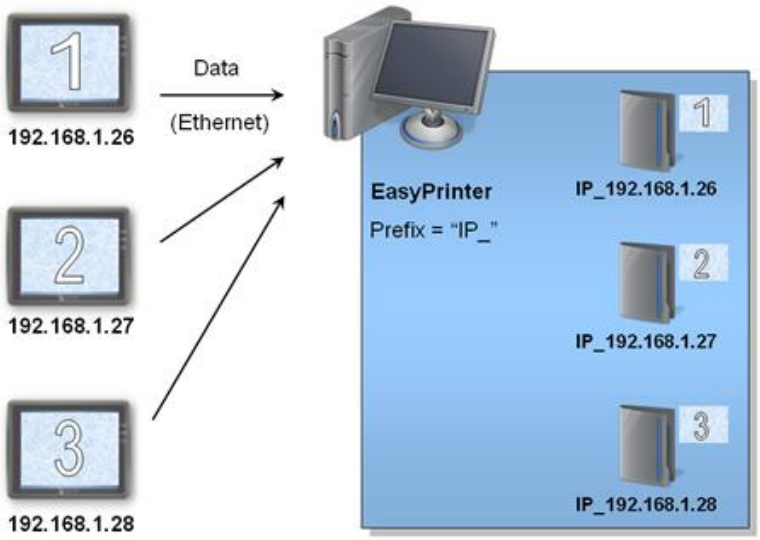
  Set the user name and password to restrict that only authorized HMI can send requests to EasyPrinter.

- **Naming Convention for HMI Folder**

  EasyPrinter use different folders to store files (e.g. hardcopy bitmap files, backup files) from different HMI. There are two ways to name the folders:

  [Use IP address]

  EasyPrinter names the folder in [Prefix] + [IP address] after the HMI in this IP address sends request.



  [Use HMI name]

  EasyPrinter names the folder in [Prefix] + [HMI name] after the HMI this name indicates sends request.

- **Properties**

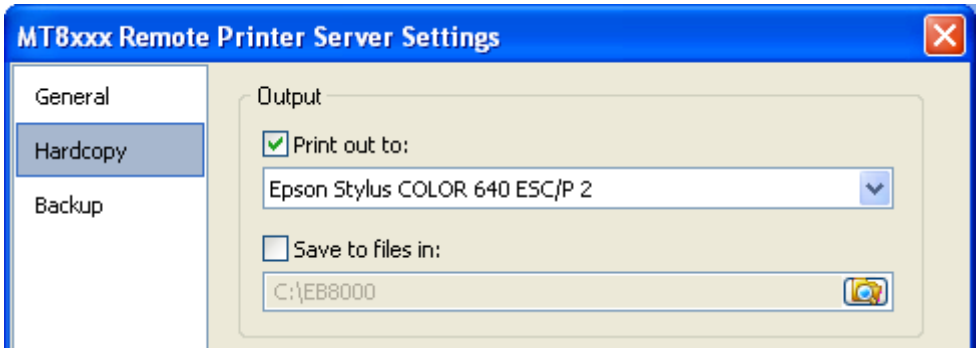
  [Minimize to system tray]

  If this check box is selected, the EasyPrinter shortcut icon will be placed in the system tray in PC. Double click the icon in system tray to open EasyPrinter.

  [Detailed message]

  Select this check box to display more detailed messages about events in the message window.

2. [Hardcopy]



EasyPrinter provides two modes to output hardcopy results:

- **Output**

  [Print out to]

  Inform EasyPrinter to print out the hardcopy result with the specified printers.

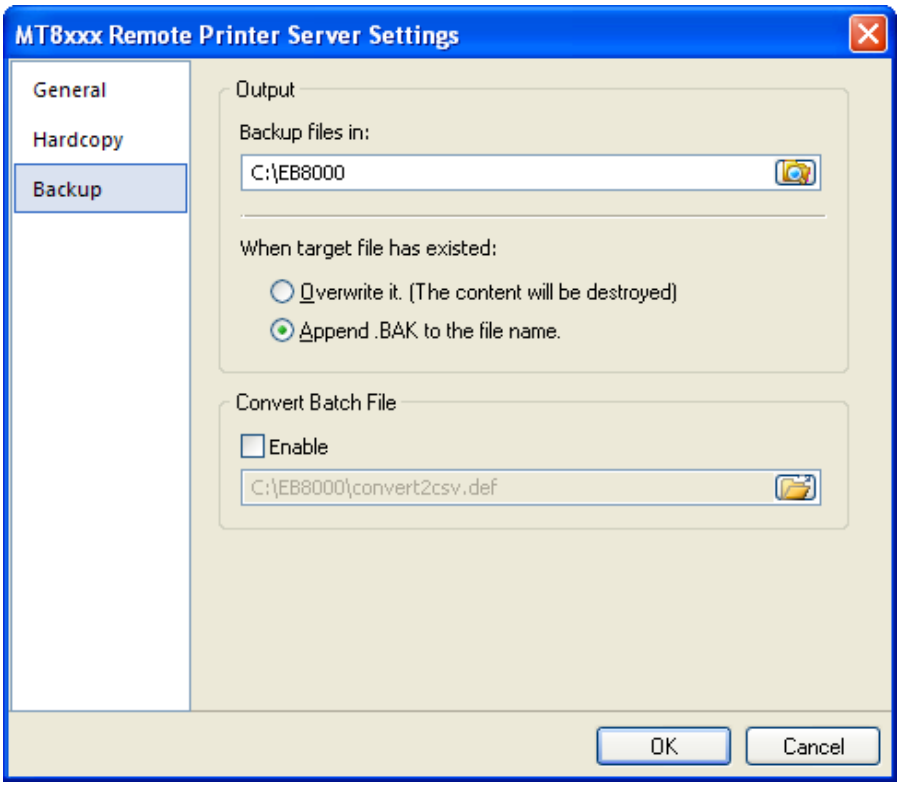
  [Save to files in]

  Inform EasyPrinter to convert the hardcopy result into a bitmap file and save it in the specified directory. Users can find the bitmap files at:

  [Specified Path] \ [HMI Folder] \ yymmdd_hhmm.bmp

  For example, when a hardcopy request is given at 17:35:00 12/Jan/2009, the bitmap file will be named “090112_1735.bmp”. And if there is another bitmap file generated in the same minute, it will be named “090112_1735_01.bmp” and so on.

3. [Backup]



- **Output**

EasyPrinter stores the backup files to the specified path.

For Event Log files:
[Specified Path] \ [HMI Folder] \ [eventlog] \ EL_yyyymmdd.evt

For Data Sampling files:
[Specified Path] \ [HMI Folder] \ [datalog] \ [Folder name of the Data Sampling] \ yyyymmdd.dtl

For Recipe files:
[Specified Path] \ [HMI Folder] \ [recipe] \ recipe.rcp or recipe_a.rcp

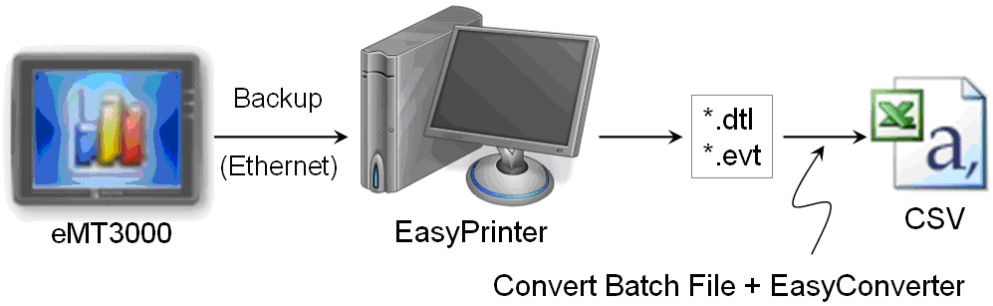
- **Convert Batch File**

  Select **[Enable]** to convert the selected history file to csv or xls (Excel) format of Convert Batch Files.

  - System registers LW9032 to LW9039 can be used to specify HMI name.
  - EasyPrinter names the folder using IP address if the HMI name is not set.

26.4 Convert Batch File

EasyPrinter provides a conversion tool to convert the uploaded Data Sampling and Event Log history files to csv files automatically. To do so, please prepare a Convert Batch File to inform EasyPrinter to convert the history files.



In the illustration above, the conversion is actually executed by EasyConverter. EasyPrinter simply follows the criteria in Convert Batch File and activates EasyConverter with proper arguments to achieve the conversion.

- EasyConverter is another Win32 application that converts history data into csv or MS Excel xls files. Users can find it in the EasyBuilder installation directory.
- Users requesting this function must ensure EasyPrinter and EasyConverter are placed in the same directory.

26.4.1 The Default Value of Convert Batch File

The following is the default Convert Batch File : convert2csv.def

Listing 1 Default Convert Batch File

```
1:  "dtl", "EasyConverter /c $( Pathname)"
2:  "evt", "EasyConverter /c $( Pathname)"
```

There are two lines in the file. Each line has two arguments separated by a comma and forms a criterion of how to process a specific type of files. The first argument stands for the extension name of the file type to be processed. The second argument stands for the command to be executed in console mode. Please note that “$(Pathname)” is a key word to inform EasyPrinter to replace it with the real name of the converted backup file. For example,

if a Data Sampling history file named 20090112.dtl is uploaded and stored, EasyPrinter will send out the following command to a console window:

```
1:  EasyConverter /c 20090112.dtl
```

A file named 20090112.csv is created.

The criteria of the default Convert Batch File:
1. Convert all Data Sampling history files (*.dtl) into csv files.
2. Convert all Event Log history files (*.evt) into csv files.

- "$(Pathname)" in the second argument stands for the full path name of the file. In the previous case, EasyPrinter replaces it with:
  [Specified Path] \ [HMI Folder] \ [datalog] \ [Folder name of the Data-Sampling object] \ 20090112.dtl
- EasyPrinter interprets the Convert Batch File in line basis, i.e. each line forms a criterion.
- Any two arguments should be separated by a comma.
- Every argument should be put in double quotes.
- Do not put any comma inside an argument.
- Please refer to "chapter25 Easy Converter" for more information.

26.4.2 Specialized Criteria

Sometime specialized criterion are needed when:
- Upload file to a specific HMI, see listing 2.
- Identify the HMI by HMI name, see listing 3.
- Process differently to different Data Sampling, see listing 4.
  (This can only be used for Data Sampling file with the file name "voltage".)
  The 3rd argument ("*") indicates this criterion accepts the Data Sampling files that meet the criterion from any HMI. Users can also change the 3rd argument to "192.168.1.26", "192.168.1.*", or HMI name, etc. for narrowing the range of the target HMI.

Listing 2 Specialized Criterion for the HMI IP = 192.168.1.26

```
1:  "dtl", "EasyConverter /c $(Pathname)", "192.168.1.26"
```

Listing 3 Specialized Criterion for HMI name = Weintek_01

```
1:  "dtl", "EasyConverter /c $(Pathname)", "Weintek_01"
```

Listing 4 Specialized Criterion for Data Sampling file name = Voltage

```
1:  "dtl", "EasyConverter /s Voltage.lgs $(Pathname)", "*", "Voltage"
```

26.4.3 The Format of a Convert Batch File

The following explains the arguments in a criterion.

File Type	Command (line)	HMI IP / Name	Condition 1	Condition 2

- File Type
  This argument specifies the extension name of the uploaded file in this criterion. (e.g. “dtl” for Data Sampling history files, “evt” for Event Log history files)
- Command (line)
  The command EasyPrinter sends to a console window if the uploaded file meets the criterion.
- HMI IP / Name
  This argument specifies the HMI that meets the criterion.
- Condition 1
  This argument specifies the folder name of the Data Sampling files that meet the criterion. This is not effective to other format of files.
- Condition 2
  Reserved

26.4.4 The Order of Examining Criterion

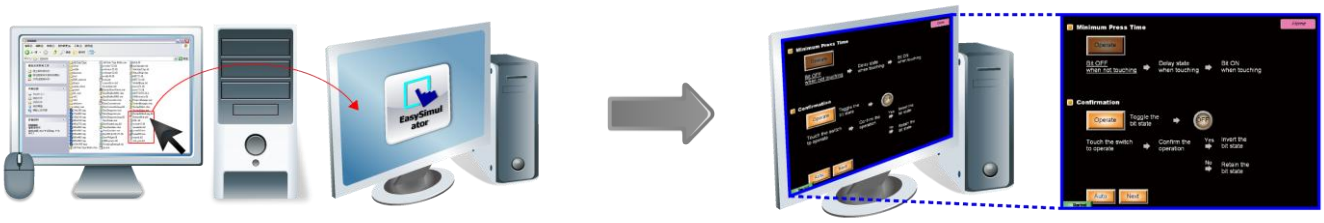
EasyPrinter examines criterion in descending order every time a file is uploaded. Once the file meets a criterion, it stops the examination and starts over for the next file. Therefore, users should place the criterion with a wider range downward in the Convert Batch File and place the more specific criteria upward. EX:

"evt", "EasyConverter /c $(Pathname)"
"dtl", "EasyConverter /c $(Pathname)"
"dtl", "EasyConverter /c $(Pathname)", "192.168.1.26"
"dtl", "EasyConverter /c $(Pathname)", "my_HMI_01"
"dtl", "EasyConverter /c $(Pathname)", "my_HMI_02"

"dtl", "EasyConverter /s Voltage.lgs $(Pathname)", "*", “Voltage"
The correct order of examination would be: (from button to top)
"dtl", "EasyConverter /s Voltage.lgs $(Pathname)", "*", “Voltage"
"dtl", "EasyConverter /c $(Pathname)", "my_HMI_02"
"dtl", "EasyConverter /c $(Pathname)", "my_HMI_01"
"dtl", "EasyConverter /c $(Pathname)", "192.168.1.26"
"dtl", "EasyConverter /c $(Pathname)"
"evt", "EasyConverter /c $(Pathname)"

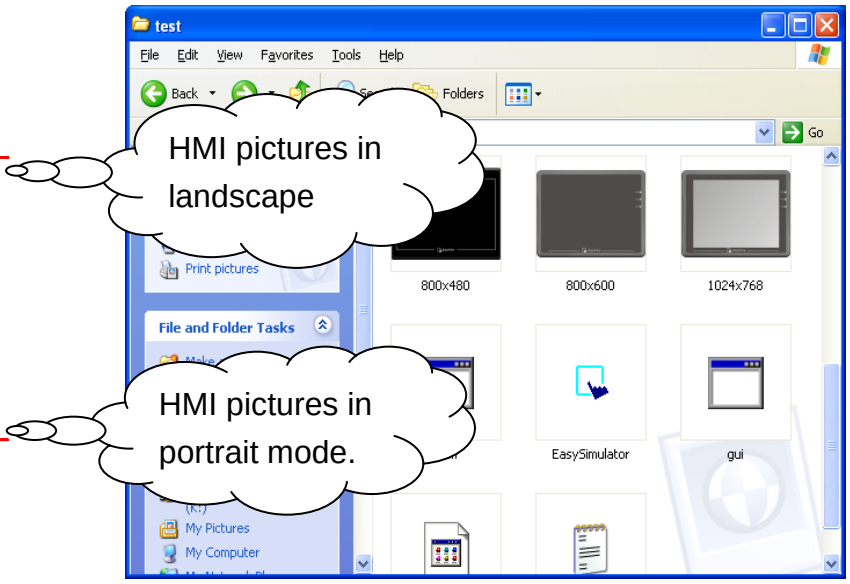
Chapter 27 EasySimulator

EasySimulator enables users to execute On-line / Off-line Simulation without installing EasyBuilder. To do this, please prepare the following files.



27.1 Prepare Needed Files

1. [driver] → [win32]
2. 320x234.bmp
3. 480x234.bmp
4. 480x272.bmp
5. 640x480.bmp
6. 800x480.bmp
7. 800x600.bmp
8. 1024x768.bmp
9. 234x320.bmp
10. 272x480.bmp
11. 480x800.bmp
12. 600x800.bmp
13. com.exe
14. EasySimulator.exe
15. gui.exe
16. xob_pos.def

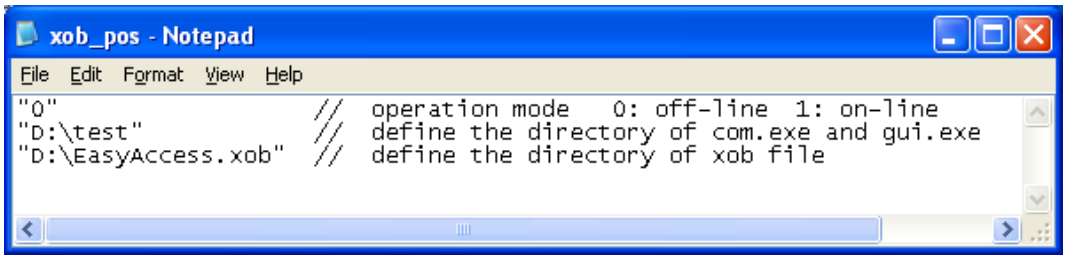


■ The files above can be found in EasyBuilder installation directory. Please install EasyBuilder on a PC first then copy the relevant files to the target PC.

27.2 Set the Content of “xob_pos.def”

Step 1

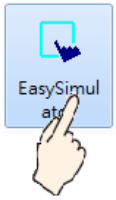
Open xob_pos.def using a text editing tool (e.g. Notepad) and set the relevant contents.



Line no.	Description
1	“0” execute Off-line Simulation; “1” execute On-line Simulation.
2	The directories of the relevant files. (e.g. com.exe, gui.exe, EasySimulator.exe…etc.)
3	The full path of the xob file.

Step 2

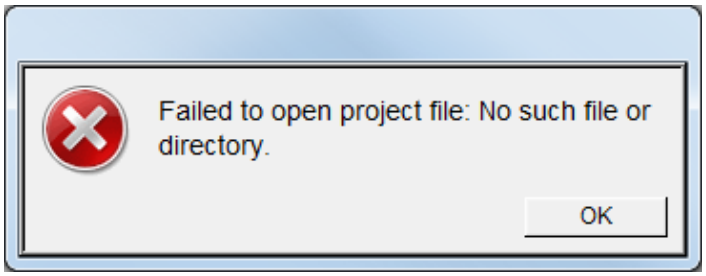
Double click on EasySimulator.exe to start simulation.



Step 3

 On-line /  Off-line Simulation is displayed on the screen.

- If EasySimulator.exe can’t be activated, please check if the relevant directories are correct.
- If the dialog below is shown, it indicates an error for xob file path, please check again.



Chapter 28 Multi-HMI Intercommunication (Master-Slave Mode)

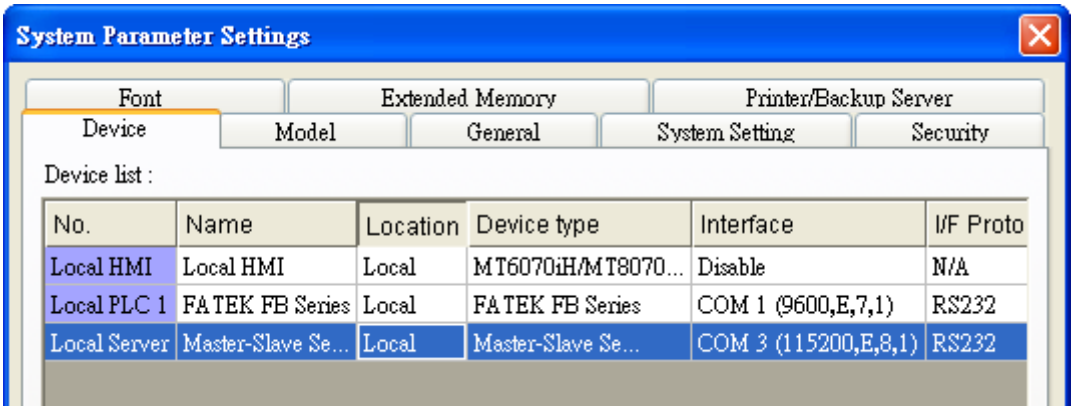
Multi-HMI intercommunication means that HMI uses COM port to connect with a remote HMI, and read/write data from/to PLC connected to remote HMI as below:

The above shows the PLC is connected with HMI 1, and HMI 1 is connected with HMI 2 via COM port, so that HMI 2 can control the PLC through HMI 1.

The following are examples of how to use EasyBuilder to create projects used on HMI 1(Master) and HMI 2 (Slave).

28.1 How to Create a Project of Master HMI

The following is the project content of HMI 1 in **[System Parameter Settings] / [Device].**

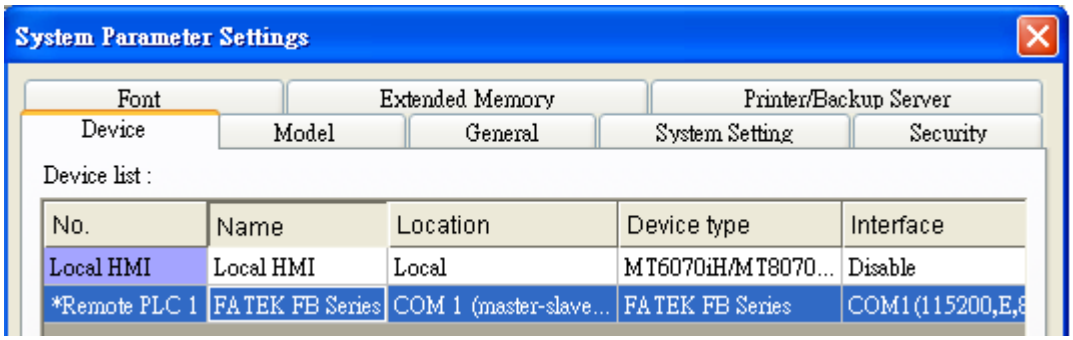


1. Due to COM 1 of HMI 1 connects PLC, the device list must include **[Local PLC 1]** in this case is “FATEK FB Series”. The communication parameters must be set correctly.

2. Due to COM 3 of HMI 1 is used to receive commands from HMI 2; a new device must be added– **[Master-Slave Server]** for setting communication properties of COM 3. The picture above shows the parameters of COM 3- “115200, E, 8, 1”, and uses RS232. These parameters are not required to be the same as PLC, but the **[Data bits]** must be set to **8**. In general, a higher baud rate for HMI 2 is recommended for a more efficient communication with PLC.

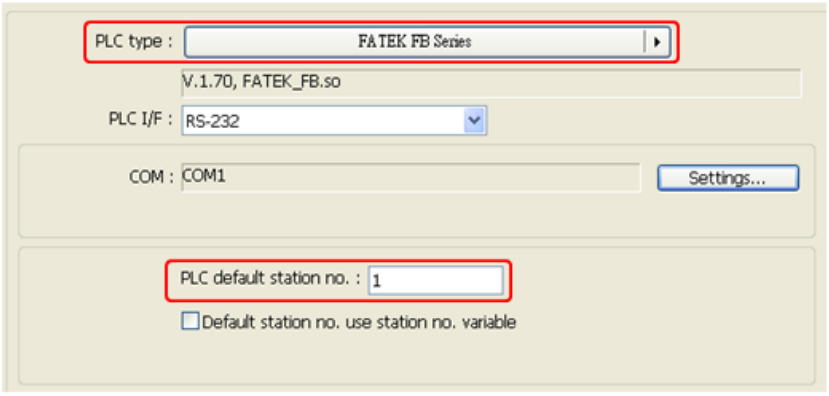
28.2 How to Create a Project of Slave HMI

The project content of HMI 2 in **[System Parameter Settings] / [Device].**

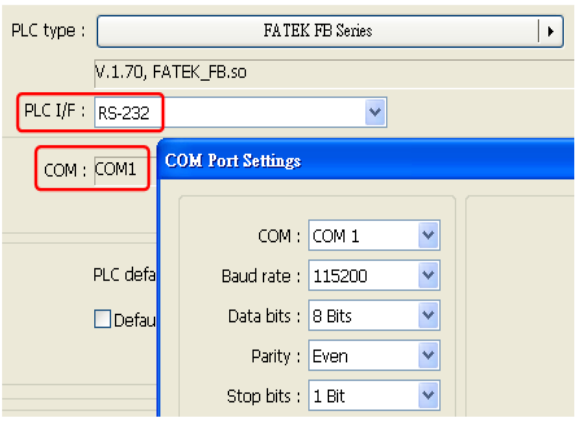


Due to the PLC that HMI 2 reads from is connected with HMI 1, thus HMI 2 views PLC as a remote device. Therefore, it is necessary to add a **[*Remote PLC 1]** into the device list and in this case is “FATEK FB Series”. The way to create [*Remote PLC 1] is described below:

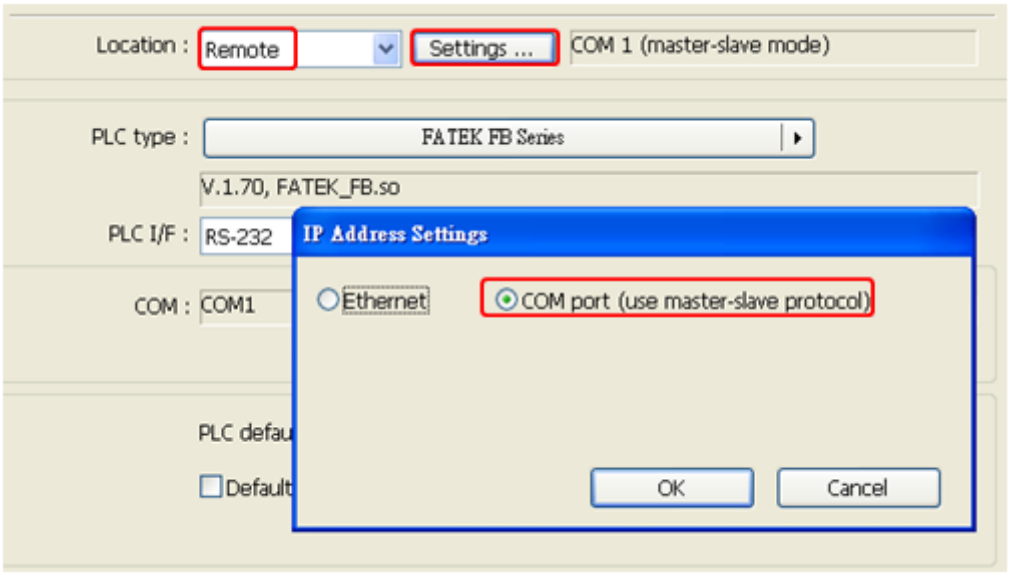
1. Create a new device”FATEK FB Series”. **[PLC default station no.]** must be the same as the connected PLC.

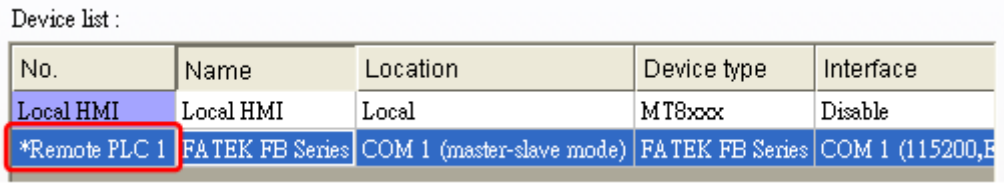


2. Correctly set the parameters. COM 1 of HMI 2 connects with COM 3 of HMI 1, so they both must have the same communication parameters and interfaces, ignoring the PLC parameters. As below, use RS232, parameters - [115200, E, 8, 1].



3. Since HMI 2 views PLC a remote device, here we change **[Location]** to **[Remote]**, and select **[COM port]** to connect remote HMI (HMI 1).





4. Upon completion of the settings, users can find a new device named **[*Remote PLC 1]** in the **[Device List]**. This device has a "*" symbol, which means, even if it contains "Remote" in the name, it actually gives commands and gets replies through a local COM port, and therefore the connection with PLC can be viewed form a local system reserved register, that is, [*Remote PLC 1], [*Remote PLC 2], [*Remote PLC 3] and [Local PLC 1], [Local PLC 2], [Local PLC 3] use the same system reserved register from the listed below:

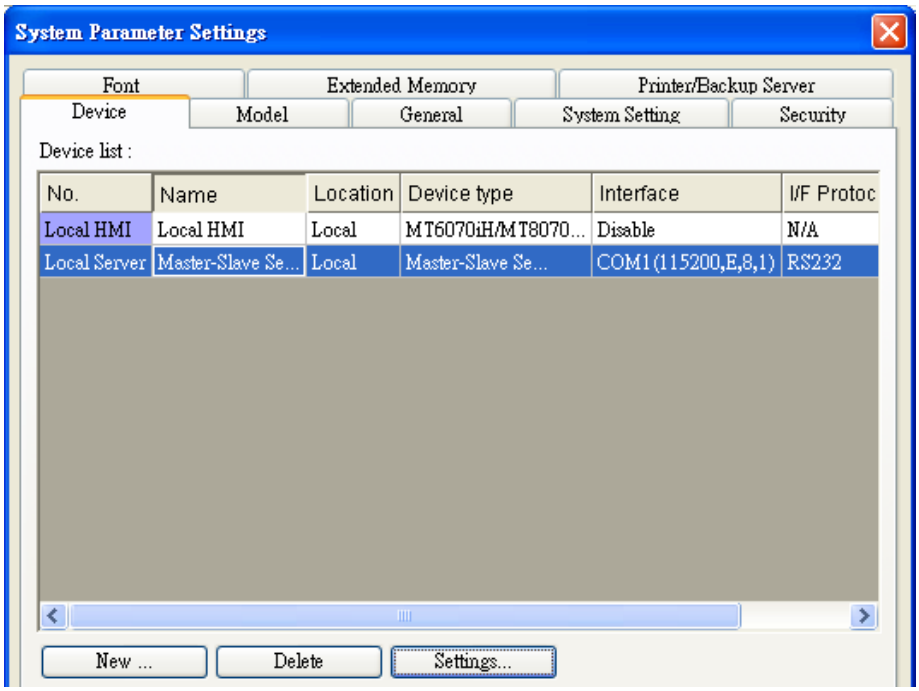
Tag	Description
LB-9150	When ON, auto. connection with PLC (COM 1) when disconnected. When OFF, ignore disconnection with PLC.
LB-9151	When ON, auto. connection with PLC (COM 2) when disconnected. When OFF, ignore disconnection with PLC.
LB-9152	When ON, auto. connection with PLC (COM 3) when disconnected. When OFF, ignore disconnection with PLC.
LB-9200~ LB-9455	These local registers indicate the connection states with PLC (through COM1). LB9200 indicates the connection state with PLC (station no. 0), and LB9201 indicates the connection state with PLC (station no. 1) and so on. When ON, indicates connection state is normal. When OFF, indicates disconnection with PLC. Set ON again, the system will then try to connect with PLC.
LB-9500~ LB-9755	These local registers indicate the connection states with PLC (through COM2). LB9500 indicates the connection state with PLC (station no. 0), and LB9501 indicates the connection state with PLC (station no. 1) and so on. When ON, indicates connection state is normal. When OFF, indicates disconnection with PLC. Set ON again, the system will then try to connect with PLC.
LB-9800~ LB-10055	These local registers indicate the connection states with PLC (through COM3). LB9800 indicates the connection state with PLC (station no. 0), and LB9801 indicates the connection state with PLC (station no. 1) and so on. When ON, indicates connection state is normal. When OFF, indicates disconnection with PLC. Set ON again, the system will then try to connect with PLC.

28.3 How to Connect with MT500 Project of Slave HMI

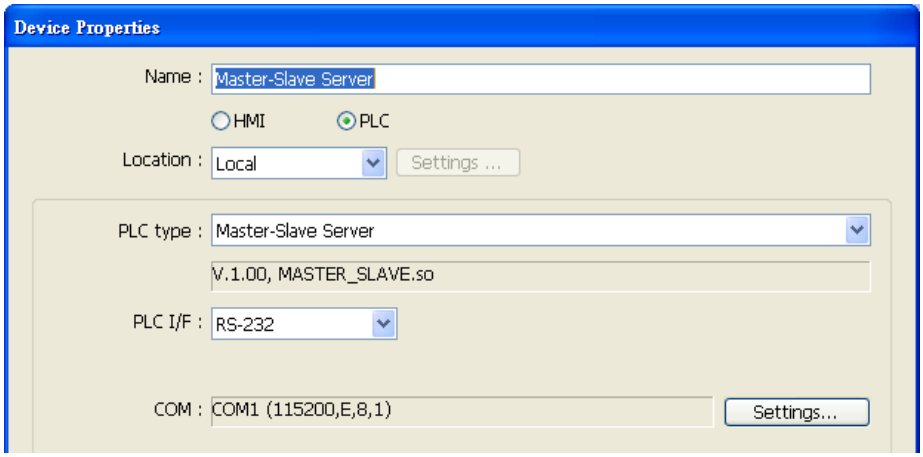
EasyBuilder Master-Slave Protocol enables MT500 to exchange data with MT8000 local data.via the connected PLC

■ EasyBuilder Settings

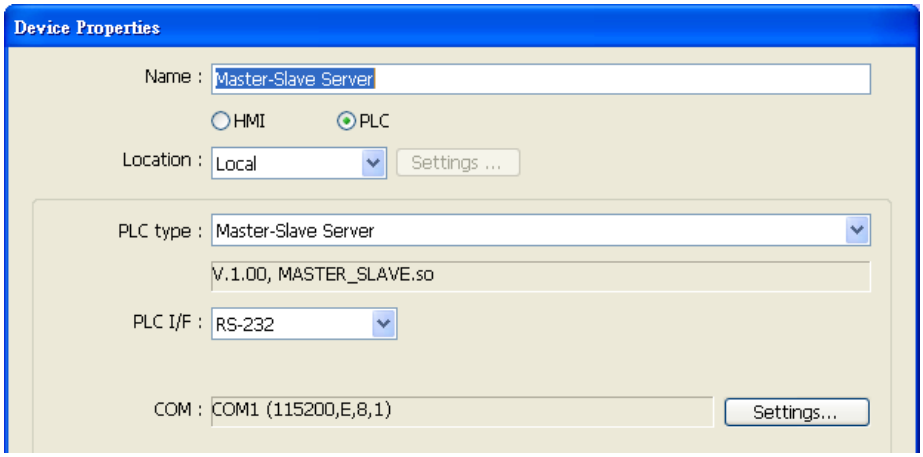
1. Select **[Master-Slave Server]** driver and click **[Settings]**. If a PLC is connected, follow the original settings.



2. Select **[RS-232]**, click **[Settings]**.

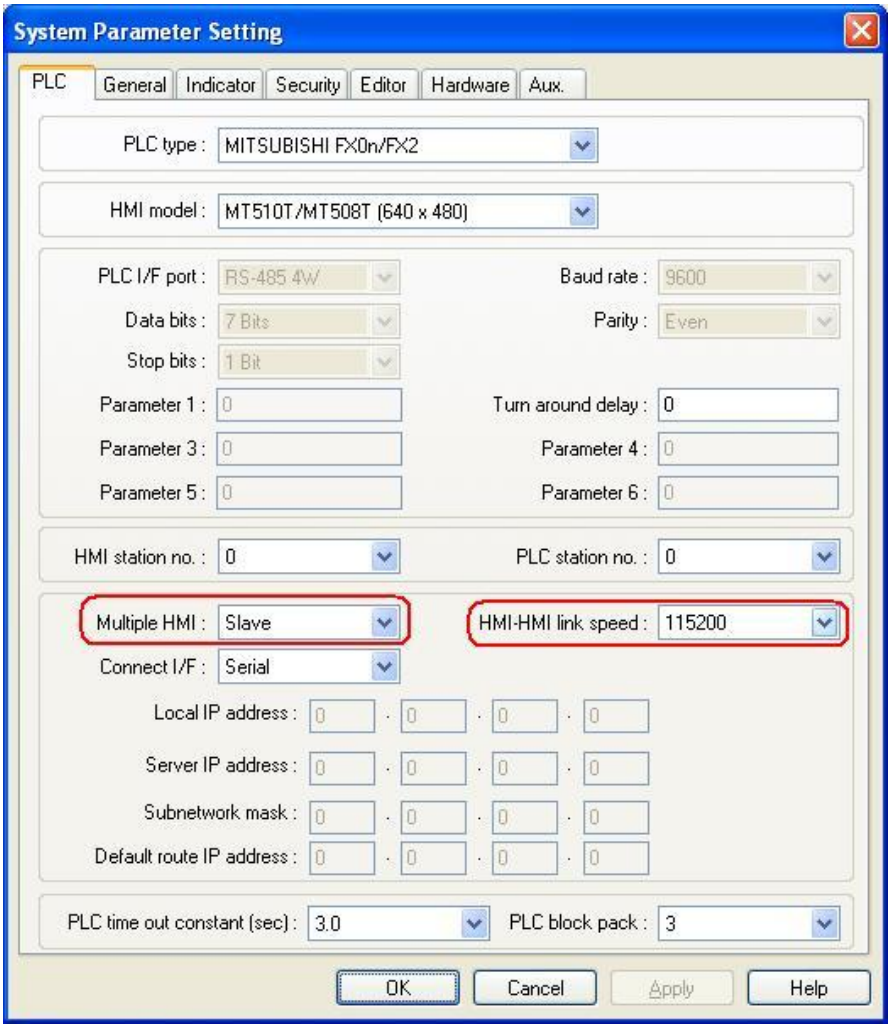


3. Fill in MT500 PLC ID No. in **[Parameter 1]** (Refer to MT500 settings).



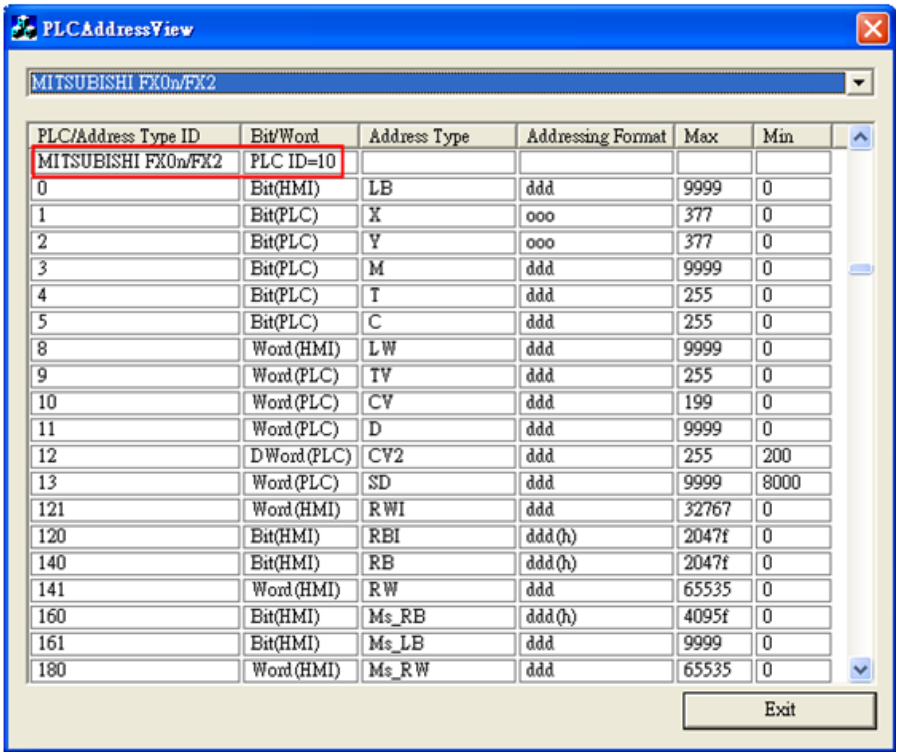
■ EB500 Settings

1. In EB500 System Parameter Settings, set Multiple HMI: Slave, HMI-HMI link speed: 115200



■ **[Baud rate]** must be identical in EB500 and EasyBuilder.

2. Double click on PLC Address View.exe to check PLC ID No. and fill in **[Parameter 1]** of EasyBuilder.



PLC/Address Type ID	Bit/Word	Address Type	Addressing Format	Max	Min
MITSUBISHI FX0n/FX2	PLC ID=10				
0	Bit(HMI)	LB	ddd	9999	0
1	Bit(PLC)	X	ooo	377	0
2	Bit(PLC)	Y	ooo	377	0
3	Bit(PLC)	M	ddd	9999	0
4	Bit(PLC)	T	ddd	255	0
5	Bit(PLC)	C	ddd	255	0
8	Word(HMI)	LW	ddd	9999	0
9	Word(PLC)	TV	ddd	255	0
10	Word(PLC)	CV	ddd	199	0
11	Word(PLC)	D	ddd	9999	0
12	DWord(PLC)	CV2	ddd	255	200
13	Word(PLC)	SD	ddd	9999	8000
121	Word(HMI)	RWI	ddd	32767	0
120	Bit(HMI)	RBI	ddd(h)	2047f	0
140	Bit(HMI)	RB	ddd(h)	2047f	0
141	Word(HMI)	RW	ddd	65535	0
160	Bit(HMI)	Ms_RB	ddd(h)	4095f	0
161	Bit(HMI)	Ms_LB	ddd	9999	0
180	Word(HMI)	Ms_RW	ddd	65535	0

3. Connect COM ports RS232 of each HMI, the communication is then enabled.

Device address:

Bit/Word	EB500	EasyBuilder	Range	Memo
B	Ms_RB	RW_Bit	dddd: 0~4095 (h): 0~f	
B	Ms_LB	LB	dddd: 0~9999	
W	Ms_RW	RW	ddddd: 0~65535	
W	Ms_LW	LW	dddd: 0~9999	

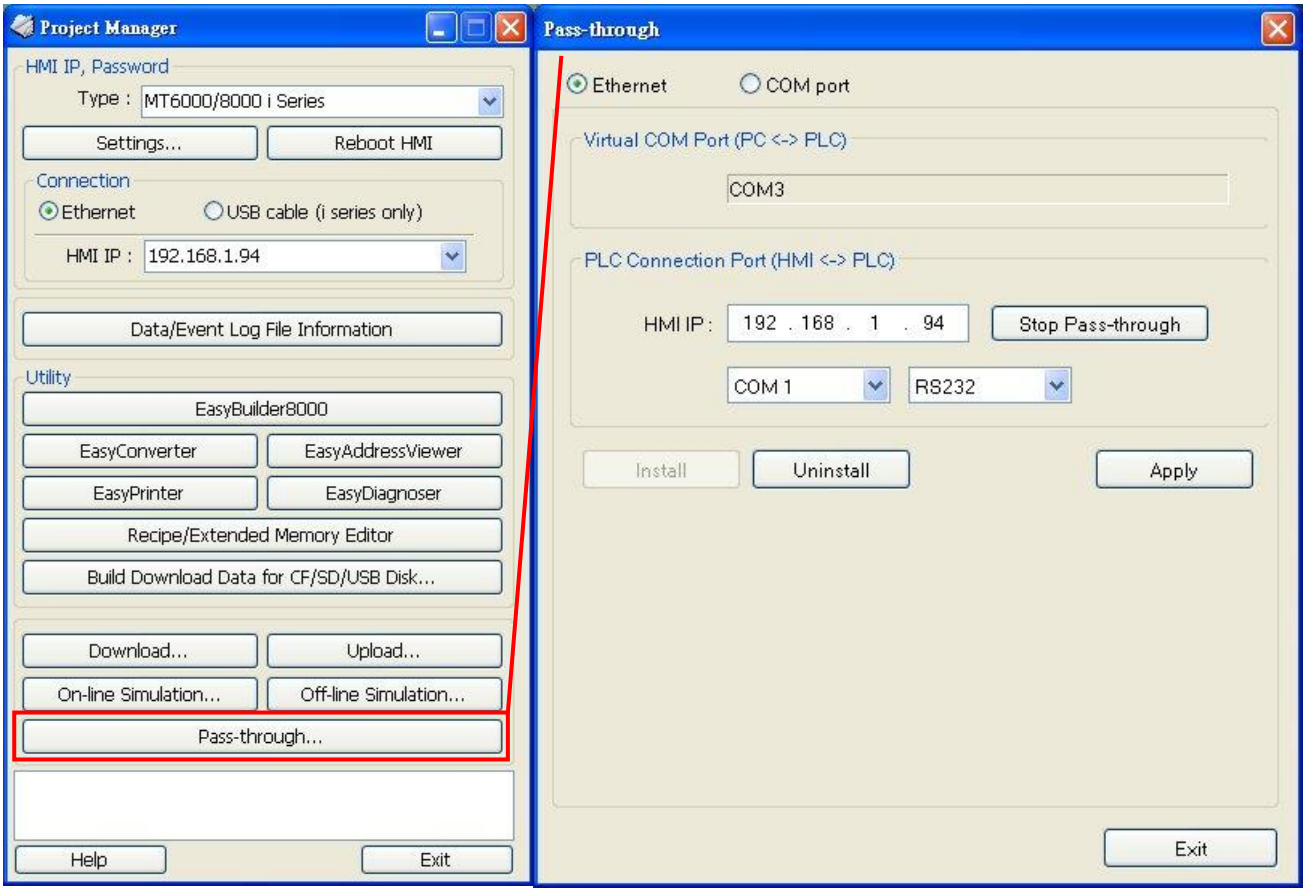
■ There will always be a PLC selected in EB500 system parameter settings, in this case, even to read/write MT8000 HMI Local Data only, the ID of the selected PLC in EB500 system parameters must also be filled in EasyBuilder parameter 1.

■ When using S7-200, S7-300 drivers, since in EB500 the high and low bytes are sent in reverse order, this will cause MT500 to misread MT8000 Local data.

Chapter 29 Pass-through Function

The Pass-through function allows the PC applications to control PLC via HMI. In this case the HMI is a converter.

The Pass-through function provides two modes: **[Ethernet]** and **[COM port]**. Click **[Pass-through]** button in Project Manager to open the setting dialog.

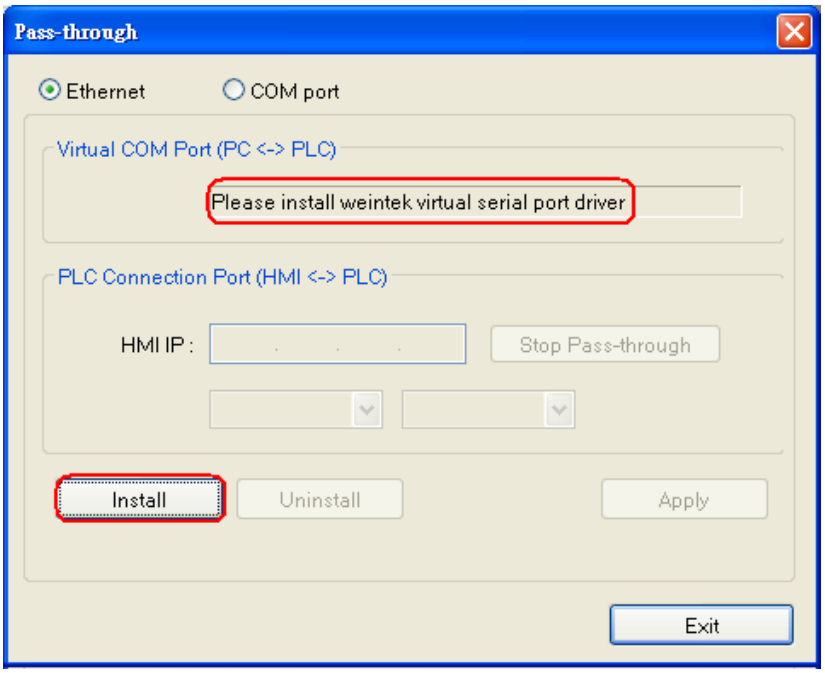


29.1 Ethernet Mode

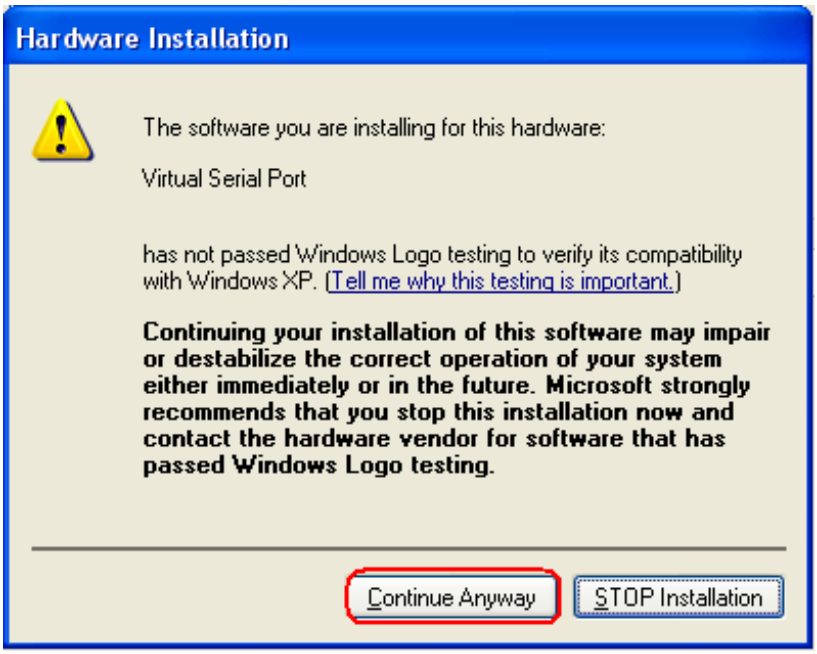
29.1.1 How to install virtual serial port driver

Before using [Ethernet] mode, please check if Weintek virtual serial port driver is installed as shown below:

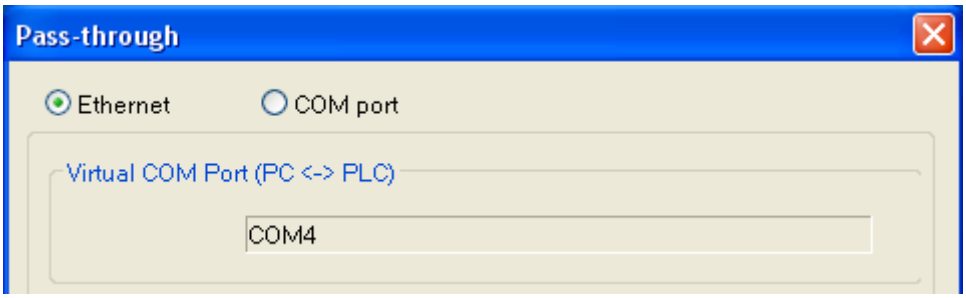
Open Project Manager to check if the driver is installed. If it shows **[Please install weintek virtual serial port driver]**, please click **[Install]**.



If the dialog below shows up during installation, please click **[Continue Anyway]**.



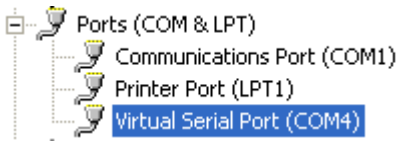
When finished, the [Virtual COM Port (PC <-> PLC)] field displays the virtual COM port used. In this example the virtual comport used is COM 4.



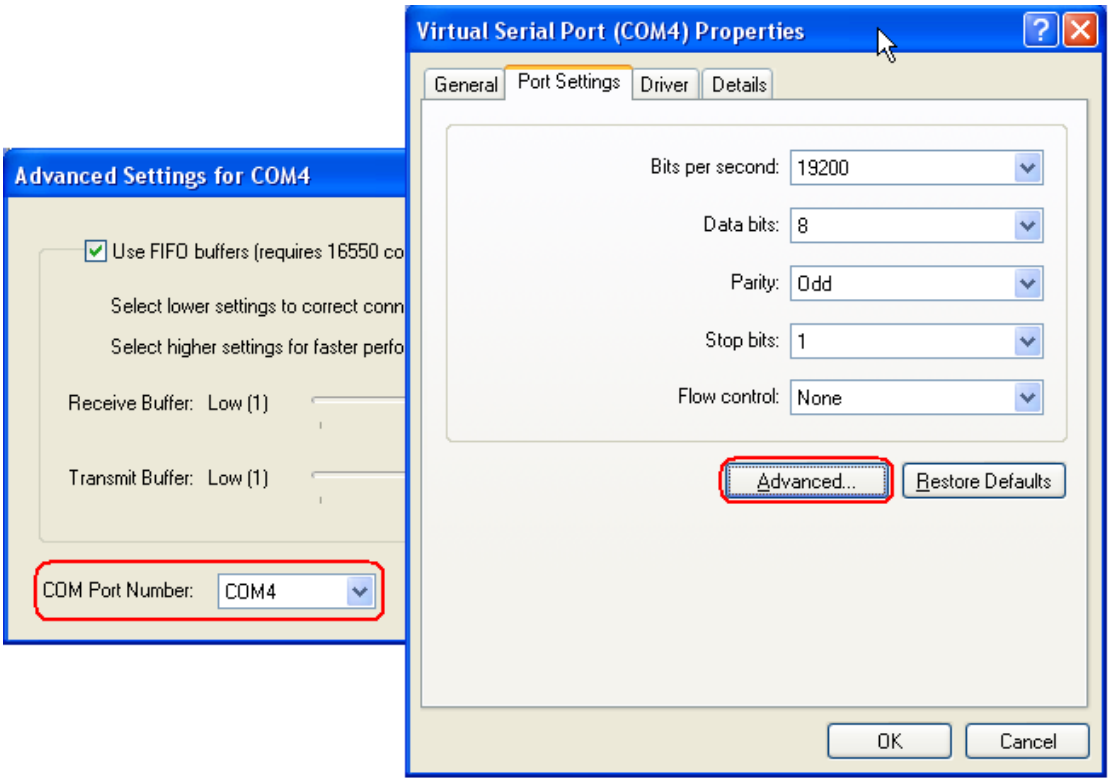
- This driver is not supported under Windows 7 64 Bit operation system.

29.1.2 How to Change the Virtual Serial Port

Open **[Device Manager]** to check the installed **[Virtual Serial Port]**.



To change the number of virtual serial port, please click **[Virtual Serial Port]** to open **[Port Settings] » [Advanced]** as follows:



29.1.3 How to Use Ethernet Mode

After installing virtual serial port driver, please follow the steps below to use Ethernet mode of pass-through function.

Step 1

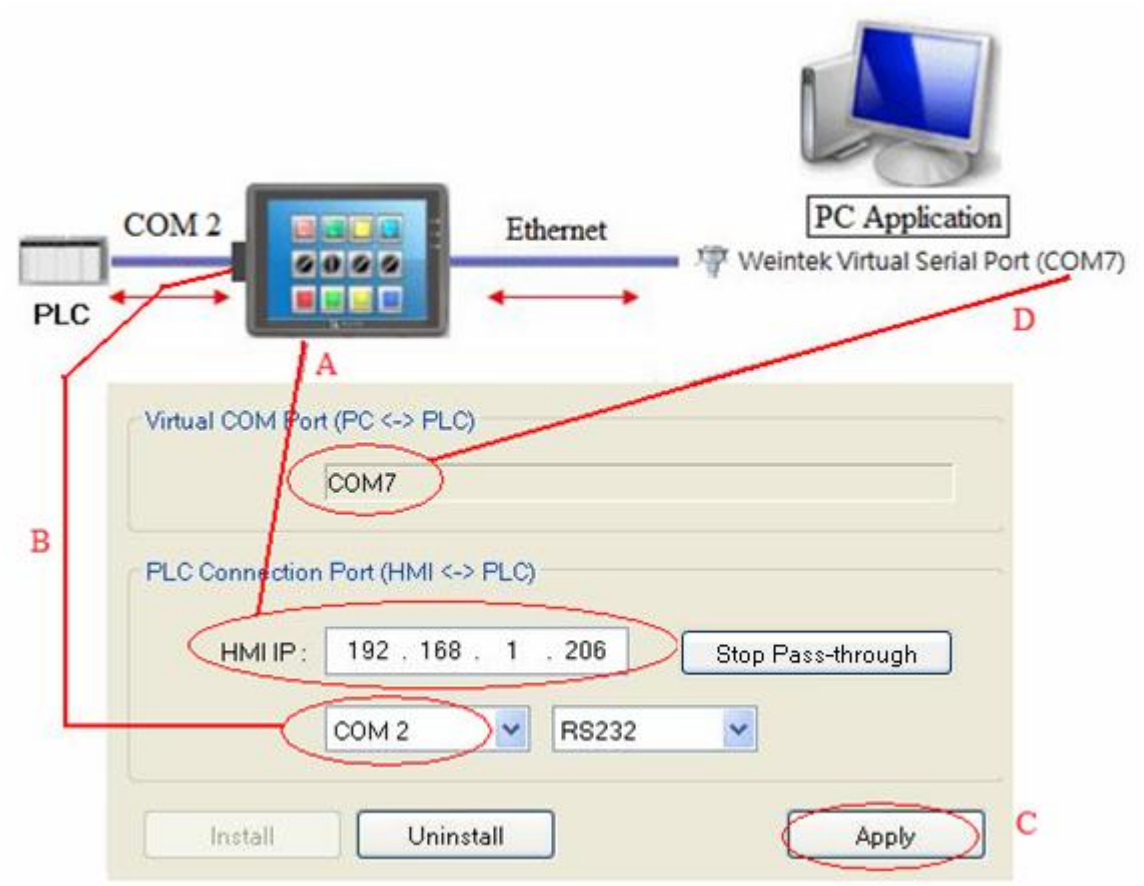
Set the IP address of the HMI connected with PLC. For example, HMI IP is 192.168.1.206

Step 2

Set the serial port that connects HMI with PLC. For example, COM 2 RS-232 is used to connect PLC.

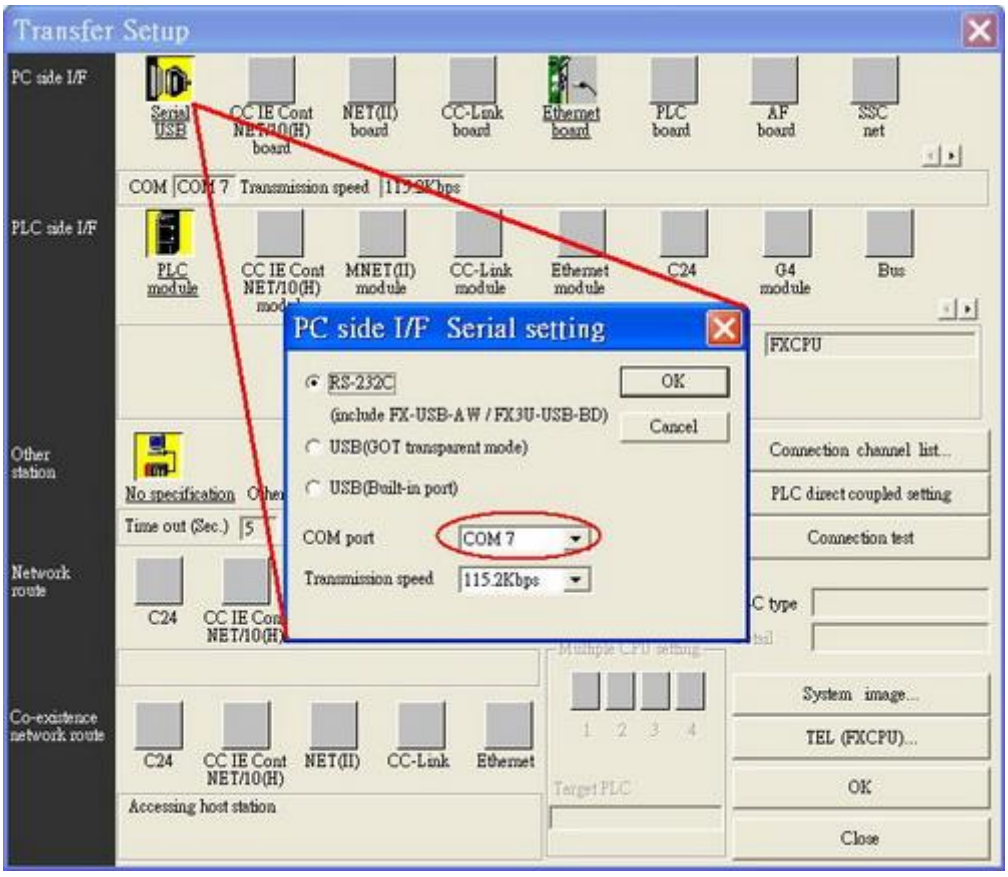
Step 3

Click **[Apply]**, to update the settings.

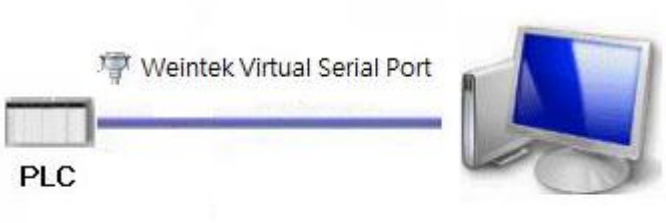


Step 4

When running PC application, set COM port to the used virtual serial port. For example, in Mitsubishi application, if the virtual serial port is COM 7, please set **[PC side I/F Serial setting] » [COM port]** to COM 7, as follows:



When finished, to execute PLC application on PC, HMI will be switched automatically to Pass-through mode (the communication between HMI and PLC will be suspended). The PLC is controlled via the virtual serial port by PC as shown below. Pass-through mode will be turned off if the application ends.



When select Ethernet under Pass-through mode, the communication between HMI and PLC can continue without stop, but not all drivers support this. System register
[LW-9903: pass-through control (0 : normal, 1 : pause, 2 : stop communications between HMI and PLC when executing pass-through)] controls the communication status under Ethernet Pass-through mode.

29.2 COM Port Mode

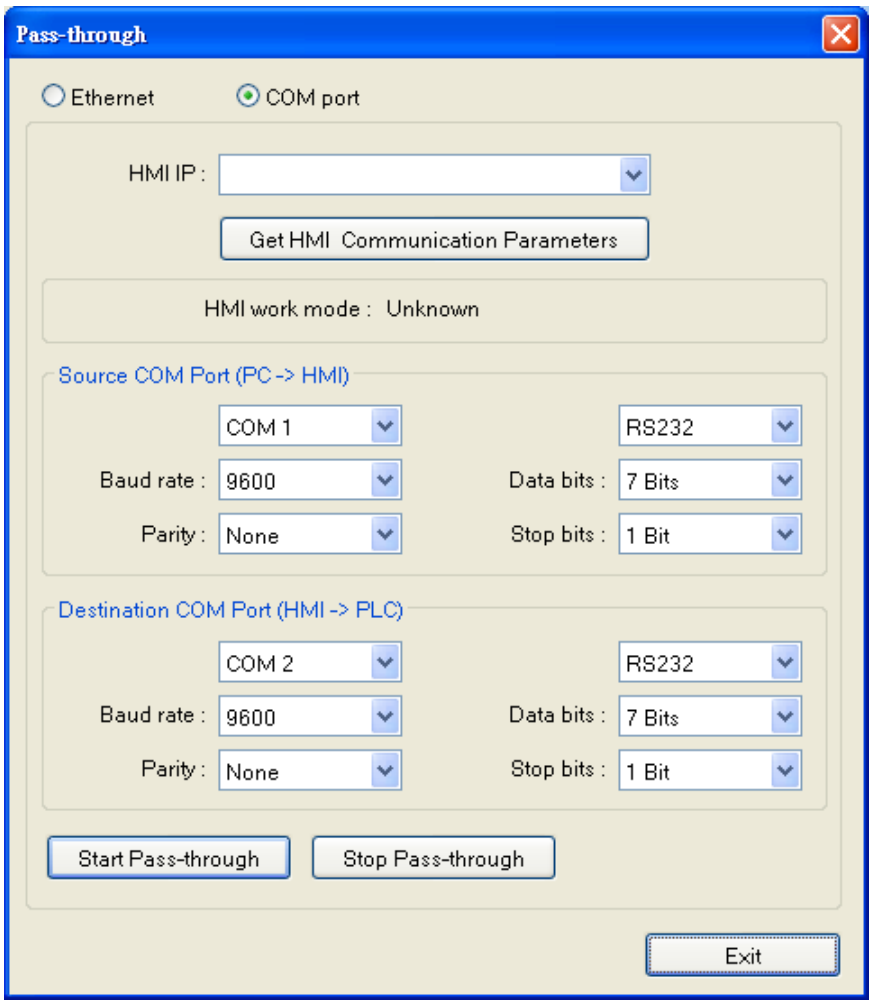
[Source COM Port] The port connects HMI with PC.
[Destination COM Port] The port connects HMI with PLC.

To use **[COM port]** mode of Pass-through, please set the properties of Source COM Port and Destination COM Port correctly.

29.2.1 Settings of COM Port Mode

There are two ways to enable [COM port] mode of Pass-through function.
(1) Project Manager.
(2) Use system registers.
LW-9901: pass-through source COM port (1 ~ 3: COM 1 ~ COM 3)
LW-9902: pass-through destination COM port (1 ~ 3: COM 1 ~ COM 3)

Start Pass-through Com Port Function in Project Manager
Click **[Pass-through]** button in Utility Manager / Project Manager to set the communication parameters as shown below:



[HMI IP]

Assign HMI IP address.

[Get HMI Communication Parameters]

Read the settings of Source and Destination COM port.

Click **[Get HMI Communication Parameters]** to update the communication parameters.

[Source COM Port]、**[Destination COM Port]**

The communication parameters of Source and Destination COM Port are displayed.

The settings will be used when **[Start Pass-through]** is clicked.

[Baud rate], **[Data bits]**, **[Parity]**, and **[Stop bits]** of Source and Destination COM Port should be set to the same. [Source COM Port] connects PC, so select RS-232 mode in most situations; [Destination COM Port] connects PLC, so the setting depends on the PLC type, RS-232, RS-485 2W, or RS-485 4W.

- When finish using pass-through (COM port) function, click **[Stop Pass-through]** to stop it. HMI will then restart to communicate with PLC.

29.2.2 HMI Work Mode

There are three work modes of HMI:

Mode	Description
Unknown	Before reading the settings of HMI, the work mode is “Unknown”.
Normal	After reading the settings of HMI, if the work mode is “Normal” the HMI does not accept any data form the Source COM Port.
Pass-through	If the work mode is “Pass-through”, the PC connected via Source COM Port can control the PLC connected via Destination COM Port.

29.3 Using System Registers to Enable Pass-Through

Another way to enable Pass-through is to use the system registers. When the values of LW-9901 (Source COM Port) and LW-9902 (Destination COM Port) match the conditions below, HMI will start Pass-through automatically:

a. The values of LW-9901 and LW-9902 must be 1 ~ 3
   (1 ~ 3: COM 1 ~ COM 3).
b. The values of LW-9901 and LW-9902 must be different.

To change the communication parameters, just change the value in the related registers and set ON [LB-9030: update COM 1 communication parameters], [LB-9031: update COM 2 communication parameters] and [LB-9032: update COM 3 communication parameters]. HMI will be forced to update the settings.

Address	Description
LB-9030	update COM 1 communication parameters (set ON)
LB-9031	update COM 2 communication parameters (set ON)
LB-9032	update COM 3 communication parameters (set ON)
LW-9550	(16bit) : COM 1 mode(0:RS232,1:RS485 2W,2:RS485 4W)
LW-9551	(16bit) : COM 1 baud rate(7:1200,8:2400,0:4800,1:9600,2:19200,3:38400,4:57600,..)
LW-9552	(16bit) : COM 1 databits (7 : 7 bits, 8 : 8 bits)
LW-9553	(16bit) : COM 1 parity (0:none, 1:even, 2:odd, 3:mark, 4:space)
LW-9554	(16bit) : COM 1 stop bits (1 : 1 bit, 2 : 2 bits)
LW-9555	(16bit) : COM 2 mode(0:RS232,1:RS485 2W,2:RS485 4W)
LW-9556	(16bit) : COM 2 baud rate(7:1200,8:2400,0:4800,1:9600,2:19200,3:38400,4:57600,..)
LW-9557	(16bit) : COM 2 databits (7 : 7 bits, 8 : 8 bits)
LW-9558	(16bit) : COM 2 parity (0:none, 1:even, 2:odd, 3:mark, 4:space)
LW-9559	(16bit) : COM 2 stop bits (1 : 1 bit, 2 : 2 bits)
LW-9560	(16bit) : COM 3 mode(0:RS232,1:RS485 2W)
LW-9561	(16bit) : COM 3 baud rate(7:1200,8:2400,0:4800,1:9600,2:19200,3:38400,4:57600,..)

LW-9562	(16bit) : COM 3 databits (7 : 7 bits, 8 : 8 bits)
LW-9563	(16bit) : COM 3 parity (0:none, 1:even, 2:odd, 3:mark, 4:space)
LW-9564	(16bit) : COM 3 stop bits (1 : 1 bit, 2 : 2 bits)

■ To stop Pass-through, change the values of LW-9901 and LW-9902 to 0.

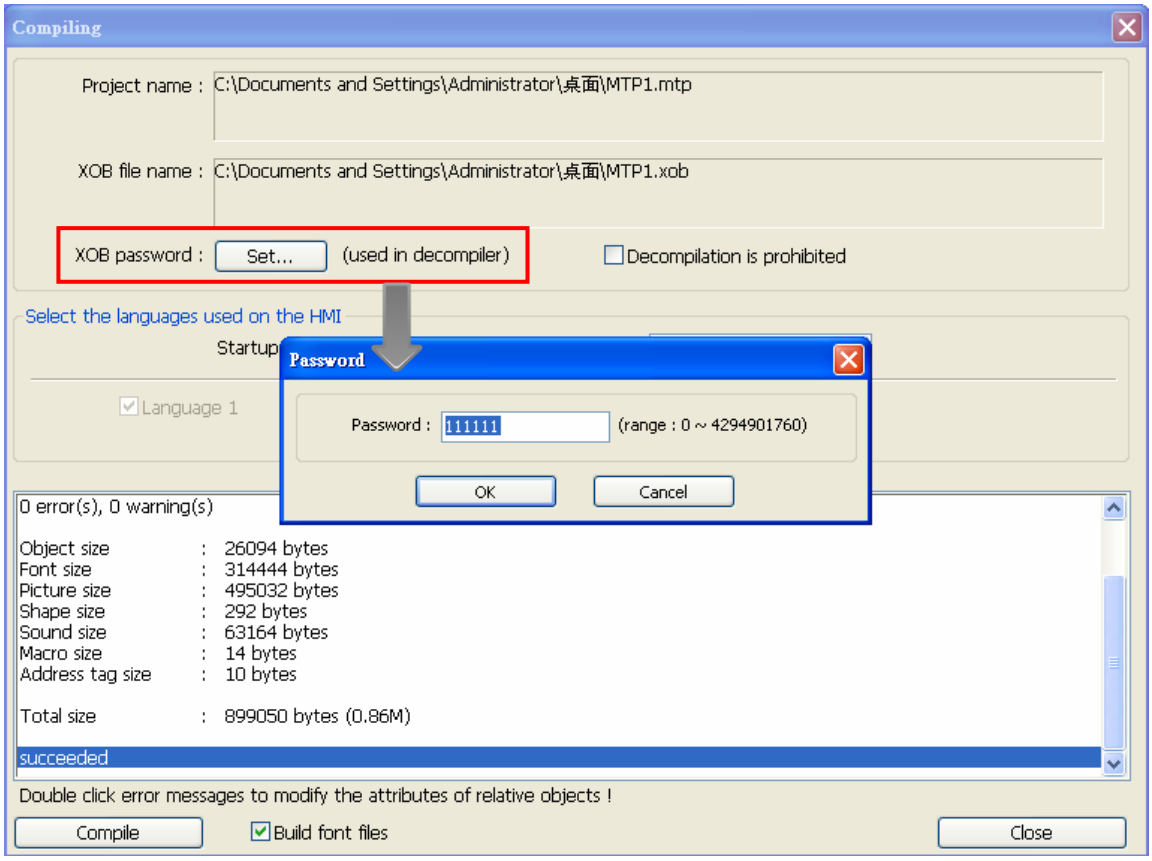
Chapter 30 Project Protection

The copyright of program design must be protected. This chapter discusses how to protect the projects by settings in EasyBuilder.

■ The protected projects **cannot** be decrypted by the factory since they are encrypted by users, therefore, please remember your password.

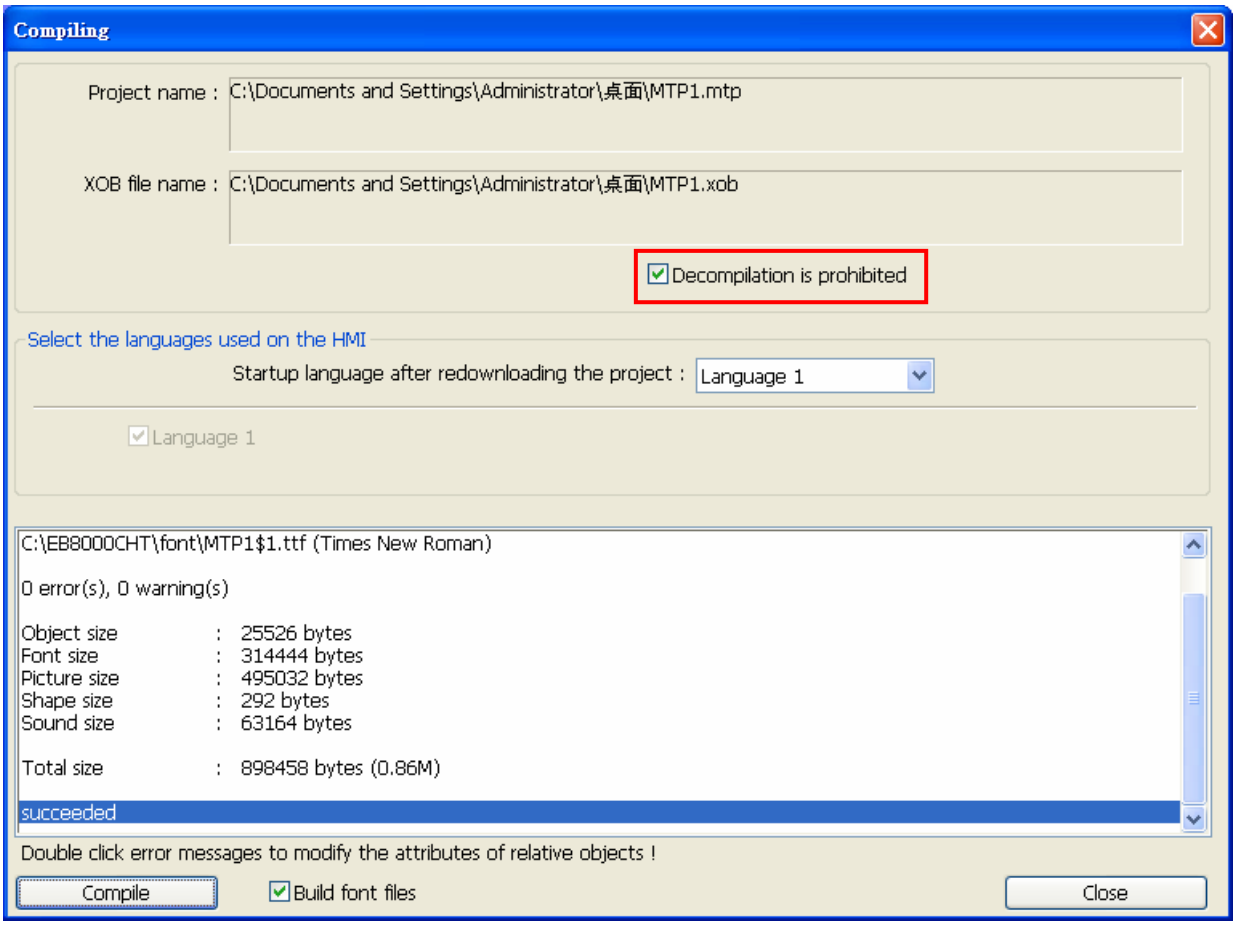
30.1 XOB Password

After a project (mtp) is done editing, users can compile the project to xob format. The xob file can be downloaded to HMI. Password can be set to protect the xob file in **[XOB password]** when compiling. (Password range: 0 ~ 4294967295)
A password will be required when attempting to decompile the xob file to mtp. If the password is entered incorrectly for three times, please restart EasyBuilder.



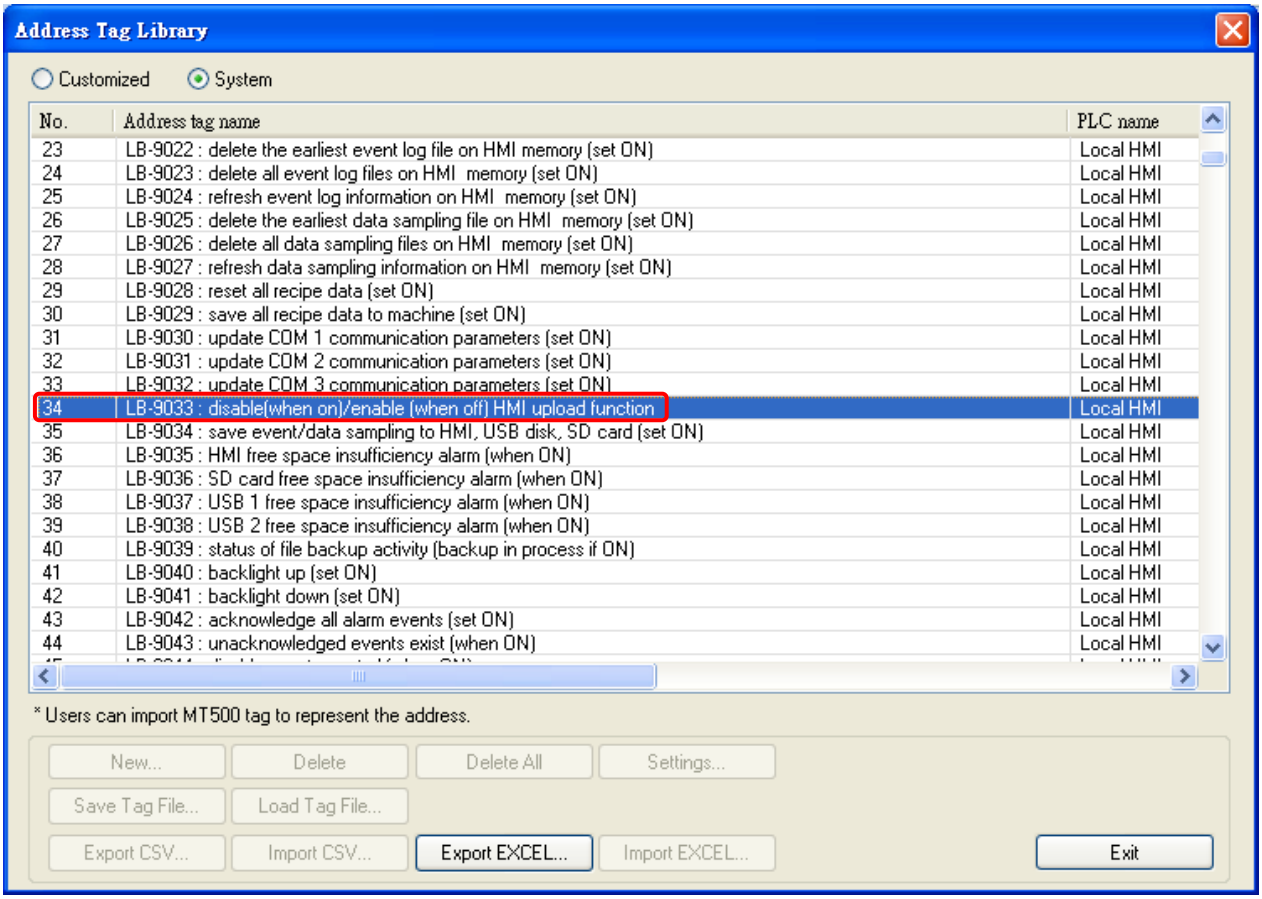
30.2 Decompilation is Prohibited

After a project (mtp) is done editing, users can compile the project to xob format. The xob file can be downloaded to HMI. Select the **[Decompilation is prohibited]** check box when compiling, the setting in **[XOB password]** will be ignored. Furthermore, the xob file cannot be decompiled to mtp file.



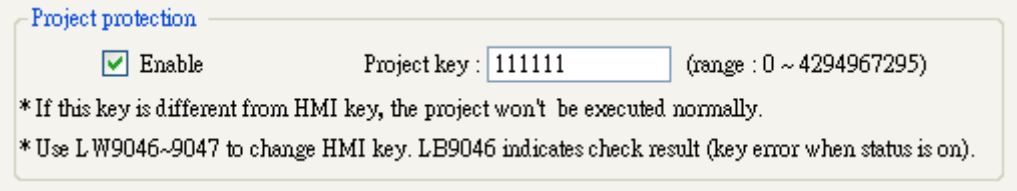
30.3 Disable XOB Upload Function

EasyBuilder provides a system reserved address [LB-9033]. When this address is set ON, xob file cannot be uploaded. To change the setting, please reboot HMI. Attempting to upload an xob file with this address set ON, the file obtained after uploading is 0 byte, and cannot be decompiled.



30.4 Project Key

Projects can be restricted to run on a specific HMI.
The setting is in **[System Parameters Settings] » [General] » [Project protection]**.



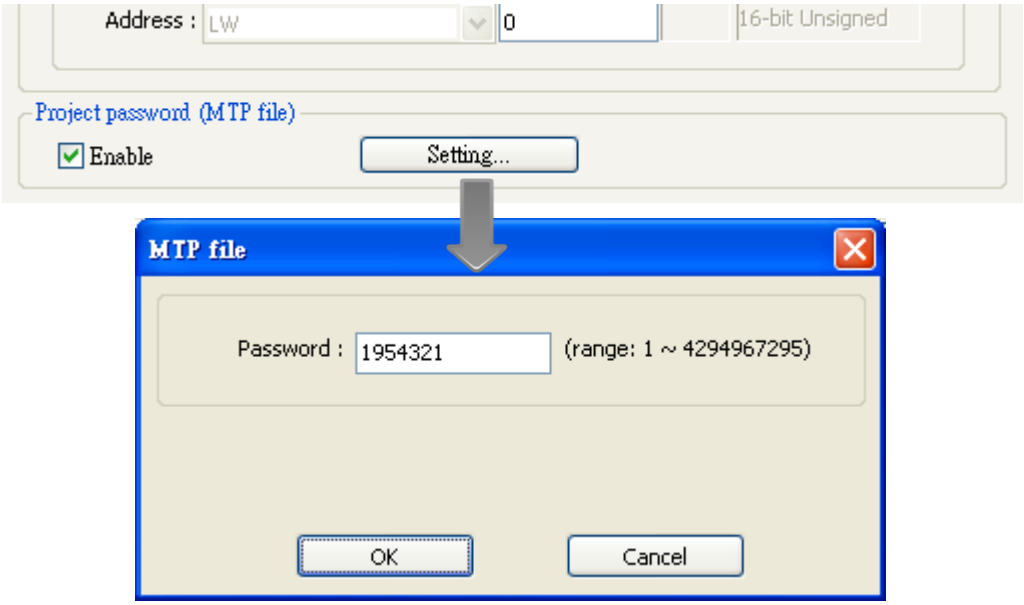
If the **[Enable]** check box is selected in **[Project protection]**, please set the **[Project key]** (password range: 0 ~ 4294967295). System registers LW-9046 ~ LW-9047 (32-bit) can be used to set the **[HMI key]** for HMI. The values in LW-9046 ~ LW-9047 cannot be read or written by a remote device. The xob file obtained after compiling can only be executed on HMI when **[HMI key]** and **[Project key]** match. If the keys don't match, LB-9046 is set ON. To change **[HMI key]**, please reboot HMI.

■ When **[HMI key]** and **[Project key]** don't match, HMI and PLC cannot communicate.

Please confirm your Internet connection before downloading the demo project.

30.5 MTP Password

After a project (mtp) is done editing, a password can be set to protect the mtp file. In **[System parameter] » [Security]** tab, select **[Enable]** check box in **[Project password]** and click **[Settings]** (password range: 1 ~ 4294967295).
The password will be required when attempting to open the mtp file.



■ When using [Window Copy] function, if the source file is protected by MTP password, please enter the correct password for the system to execute window copy.

Chapter 31 Memory Map Communication

31.1 Introduction

Memory Map communication protocol is similar to IBM 3764R, and it is used when the memory data transferred seldom between two devices. When setting the two devices, one is set as Master, and another is Slave. Generally, Master and Slave do not communicate unless the data in the assigned address has changed. Once the data is synchronized, the communication will stop. The purpose of Memory Map is to keep the consistency of the assigned part of data between two devices (Master and Slave).

The corresponding addresses of Master and Slave devices should have the same property as MW (MB) address type. The size of MW (MB) in HMI is 10,000 words.

MB and MW indicate the same area of memory, for example, MB0~MBf correspond to the bits of MW0, MB10~MB1f correspond to MW1, as shown below:

Device Type	Format	Range
MB	DDDDh	DDDD:0~4095 h:0~f(hex)
MW	DDDD	DDDD:0~9999

31.2 PIN Settings

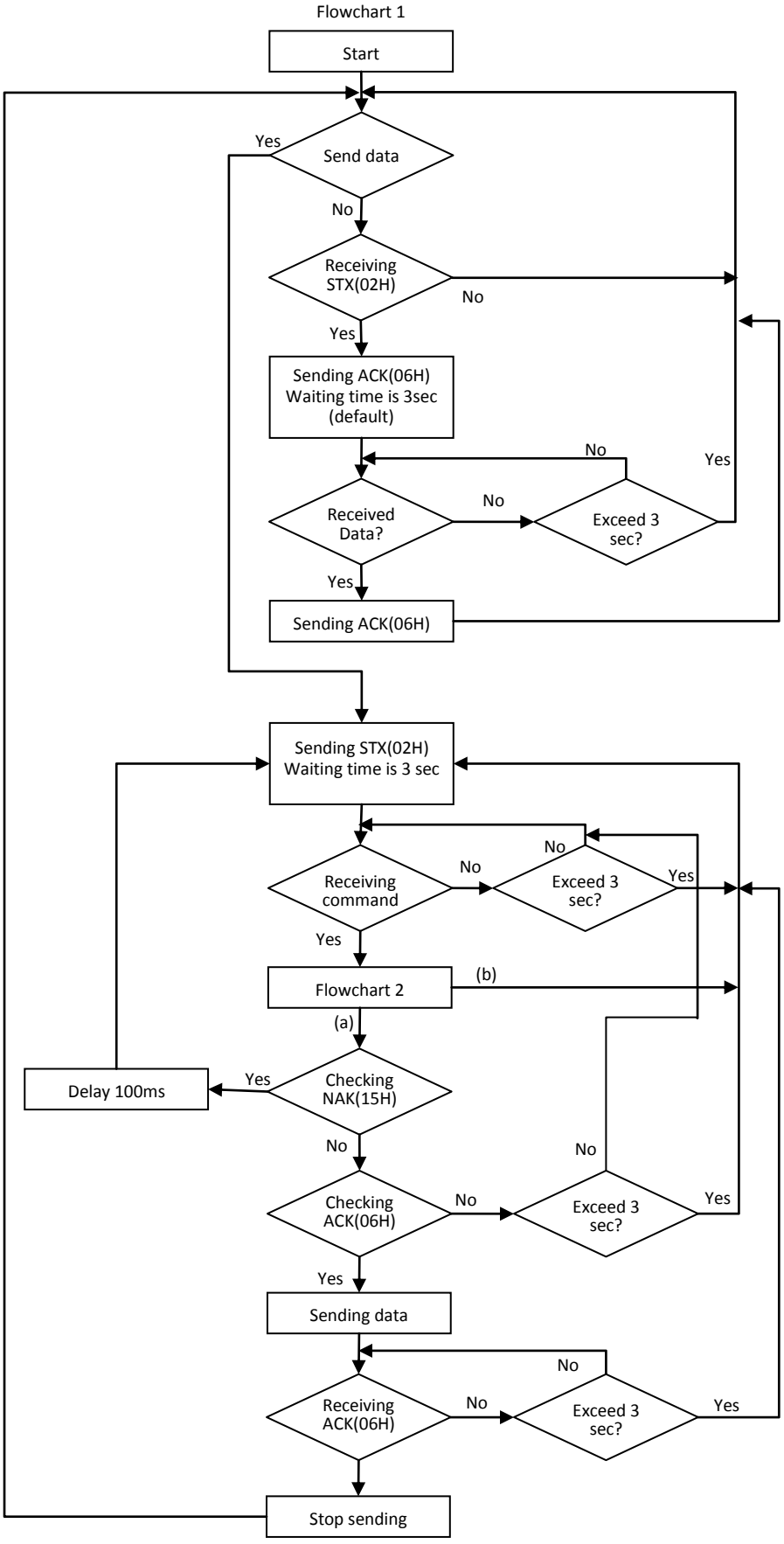
When using Memory Map communication protocol, the Master and Slave must have the same communication parameters. The wiring is shown below:

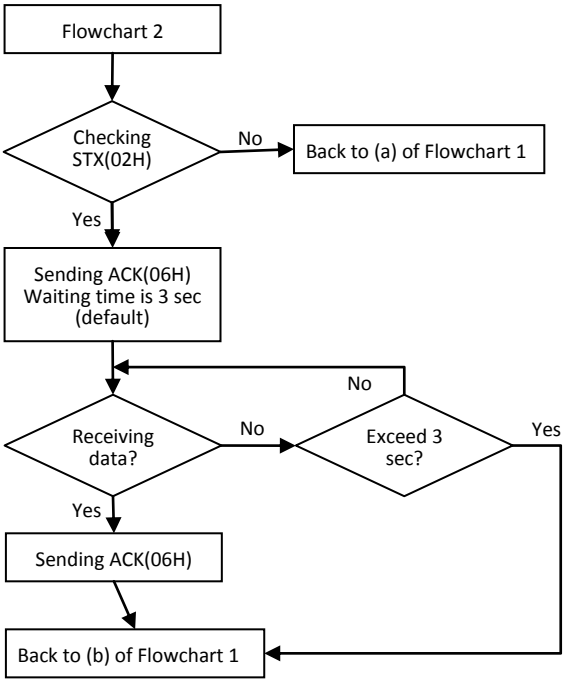
COM Port	RS232	
Device	Master	Slave
Pin Mapping	TX(#)	RX(#)
	RX(#)	TX(#)
	GND(#)	GND(#)

COM Port	RS485 (4W)	
Device	Master	Slave
Pin Mapping	TX+(#)	RX+(#)
	TX-(#)	RX-(#)
	RX+(#)	TX+(#)
	RX-(#)	TX-(#)
	GND(#)	GND(#)

Note: the # will be distinct depends on the type of PLC or controller.

31.3 Communication Flowchart





- Flowchart 2 works for Slave but not Master.
- STX: Start of Text, ACK: Acknowledge, NAK: Negative Acknowledge

31.4 Address Types

There are two address types, MB and MW.

The format of the commands that controls MB are listed below:

MB Commands		
Offset (byte)	Format	Description
0	0x02	The operating sign to MB
1 2	0x## 0x##	Address (Low byte) Bit Address (High byte) For example: MB-18 = 1*16 + 2 = 18 = 0x12 and 0x00
3	0x00 (or 0x01)	The data in MB address. (Bit type, must be 0 or 1)
4, 5	0x10, 0x03	Stop sign
6	0x##	The checksum. Calculate XOR from offset 0 to 5.

The format of the commands that controls MW are listed below:

MW commands		
Offset(byte)	Format	Description
0	0x01	The operating sign to MW
1 2	0x## 0x##	Address (Low byte) Bit Address (High byte) If the address includes 0x10, insert another 0x10 after it and all offsets after that are increased by 1. For example: 0x10, 0x04 will become 0x10,0x10,0x04
3	0x##	Number of sending bytes (To control a word, the number of bytes must be even). If the number of bytes is 0x10, insert another 0x10 after it and all offsets after that are increased by 1.
4 to 4+n-1	0x##(L),0x##(H) 0x##(L),0x##(H) …	The address that the first and second bytes correspond to is the initial address. “n” is the number of bytes. If the data includes 0x10, insert another 0x10 after it and the “Number of sending bytes” (offset 3) remains the same, but n = n + 1. Same thing applies to other 0x10 data.
4+n, 4+n+1	0x10 0x03	End sign
4+n+2	0x##	The checksum. Calculate XOR from all above.

31.4.1 Communication Examples

If Master sets the data of MW-3 to 0x0a, Master will build communication with Slave immediately due to the data changed, so Slave will update its MW-3 to 0x0a, the procedure is shown below:

1. Master sends STX(0x02h).
2. Slave receives STX(0x02h) from Master, and sends ACK(0x06h) to Master.
3. Master receives ACK(0x06h) from Slave.
4. Master sends 0x01,0x03,0x00,0x02,0x0a,0x00,0x10,0x03,0x19, as shown below:

Offset (byte)	Format	Description
0	0x01	The operating sign for MW
1 2	0x03 0x00	Address(Low byte) Bit Address (High byte)
3	0x02	The number of bytes sent (MW-3= two bytes).
4, 5	0x0a, 0x00	Data in MW-3 is 0x0a and 0x00
6, 7	0x10, 0x03	End sign
8	0x19	The checksum 0x01^0x03^0x00^0x02^0x0a^0x00^0x10^0x03=0x19

5. Slave receives data from Master and then sends ACK(0x06h).
6. Master receives ACK(0x06h) from Slave.

When finish communicating, Master sends the updated data in MW to Slave, and Slave synchronizes its MW data with Master.

Example 2

If the data includes 0x10; please notice the change in data format.
If set MW-10 of Slave to 0x10, Slave will build communication with Master immediately, and Master will update its MW-10 to 0x10, the procedure is shown below:

1. Slave sends STX(0x02h)
2. Master receives STX(0x02h) from Slave, and sends ACK(0x06h) to Slave.
3. Slave receives ACK(0x06h) from Master
4. Slave sends 0x01,0x10,0x10,0x00,0x02,0x10,0x10,0x00,0x10,0x03,0x10 as shown below:

Offset (byte)	Format	Description
0	0x01	The operating sign to MW
1	0x10	Address(Low byte)
2	0x10	Insert 0x10
3	0x00	Bit Address (High byte)
4	0x02	The number of bytes sent (MW-10= two bytes).
5	0x10	0x10 is the low byte in MW-10
6	0x10	Insert 0x10
7	0x00	0x00 is the high byte
8	0x10	End sign
9	0x03	
10	0x10	The checksum, 0x01^0x10^0x10^0x00^0x02^0x10^0x10^0x00^0x10^0x03=0x10

5. Master receives data from Slave and sends ACK(0x06h) to Slave.
6. Slave receives ACK(0x06h) from Master.

Slave sends the updated data in MW to Master, and Master synchronizes its MW data with Slave.

31.5 Settings

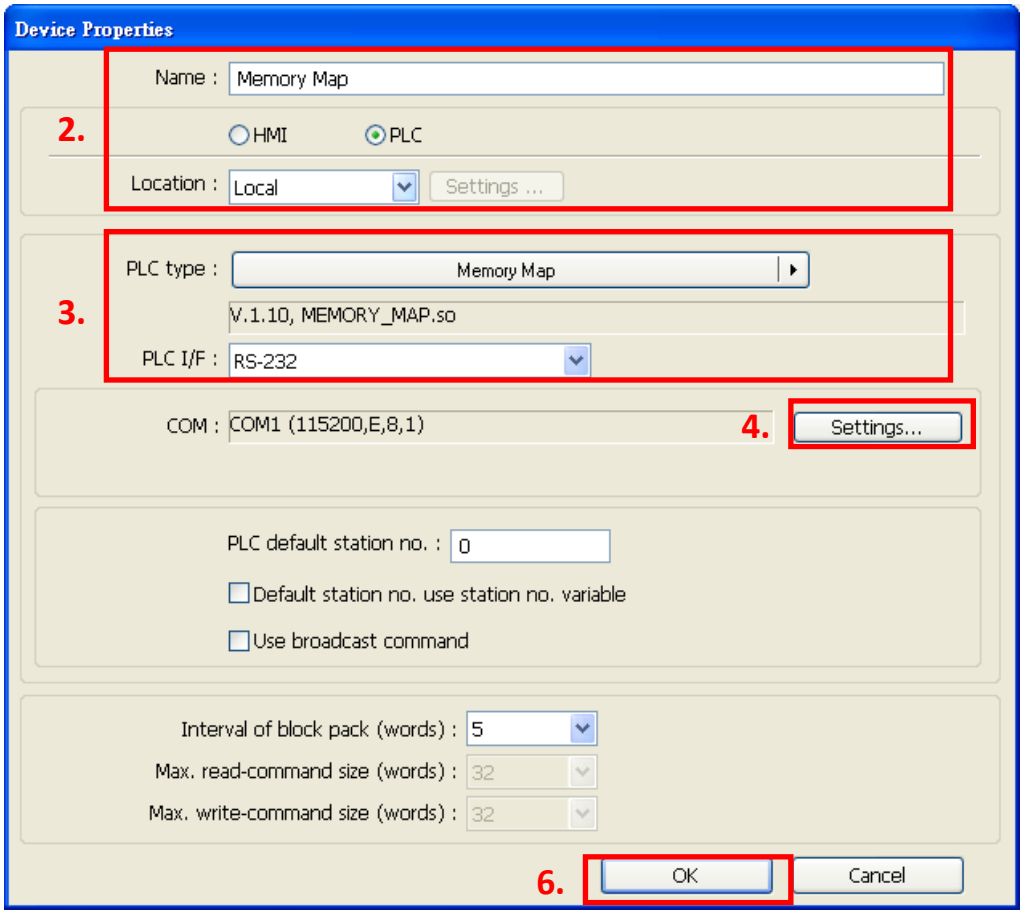
The following explains how to connect two HMIs using Memory Map protocol.

- If the type of these two HMIs are different, please create different project files, or, after setting the first HMI, directly change to the type of the second HMI in **[Edit] » [System Parameter Settings] » [Model]**, and then compile and download the project to the second HMI.

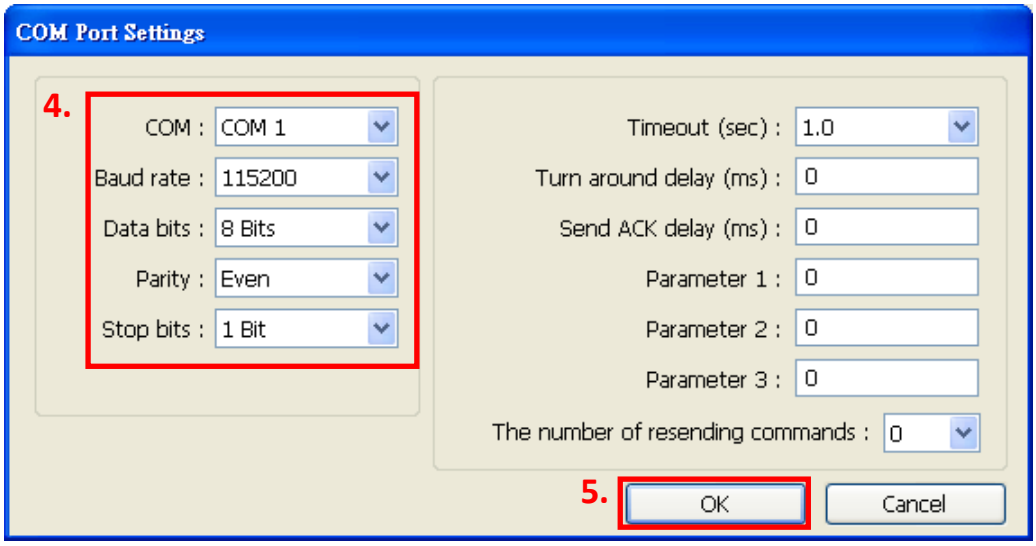
31.5.1 Add a Memory Map Device

Launch EasyBuilder, select **[New]**, and the model of HMI, as shown below:

1. Click **[Edit]** form the main menu, click **[System Parameter Settings]**, and select **[Device]** tab, then click **[New]** to add a new device.
2. In the **[Name]** field enter “Memory Map”, and then select **[PLC]**, set the **[Location]** to **[Local]**.
3. Set **[PLC type]** to **[Memory Map]**, and set **[PLC I/F]** to **[RS-232]**.



4. Click **[Settings]**, and the setting is shown below:
   A. COM: COM 1
   B. Baud rate: 115200
   C. Data bits: 8 Bits
   D. Parity: Even
   E. Stop bit: 1 Bit



5. After setting the COM port click **[OK]**.
6. Click **[OK]** to finish setting.

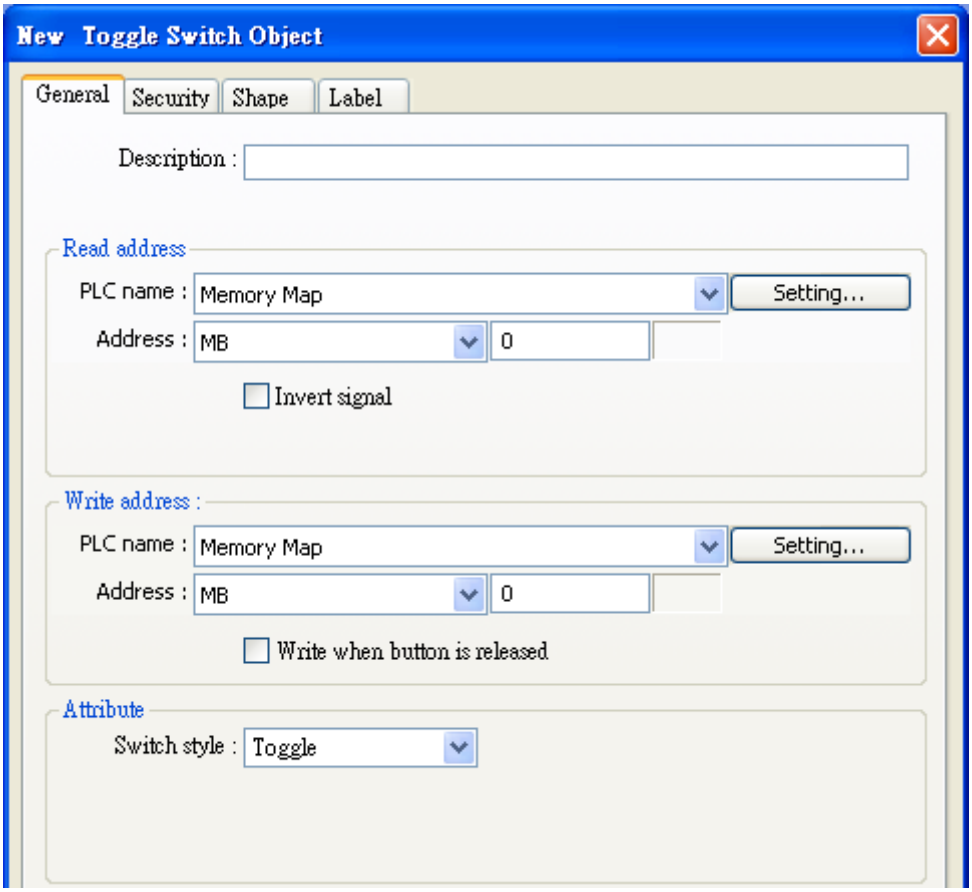
- Memory Map in MT500 is divided into [Memory Map_Master] and [MemoryMap_Slave]; please refer to the relevant manual.
  For eMT3000 and MT8000 Series, select [Memory Map] in the PLC type setting.
- **[Data bit]** must set to 8 bits.
- All the settings of the two HMIs must set to the same

31.5.2 Object Settings

Add two objects in window no. 10, a Toggle Switch and a Multi-state Switch:

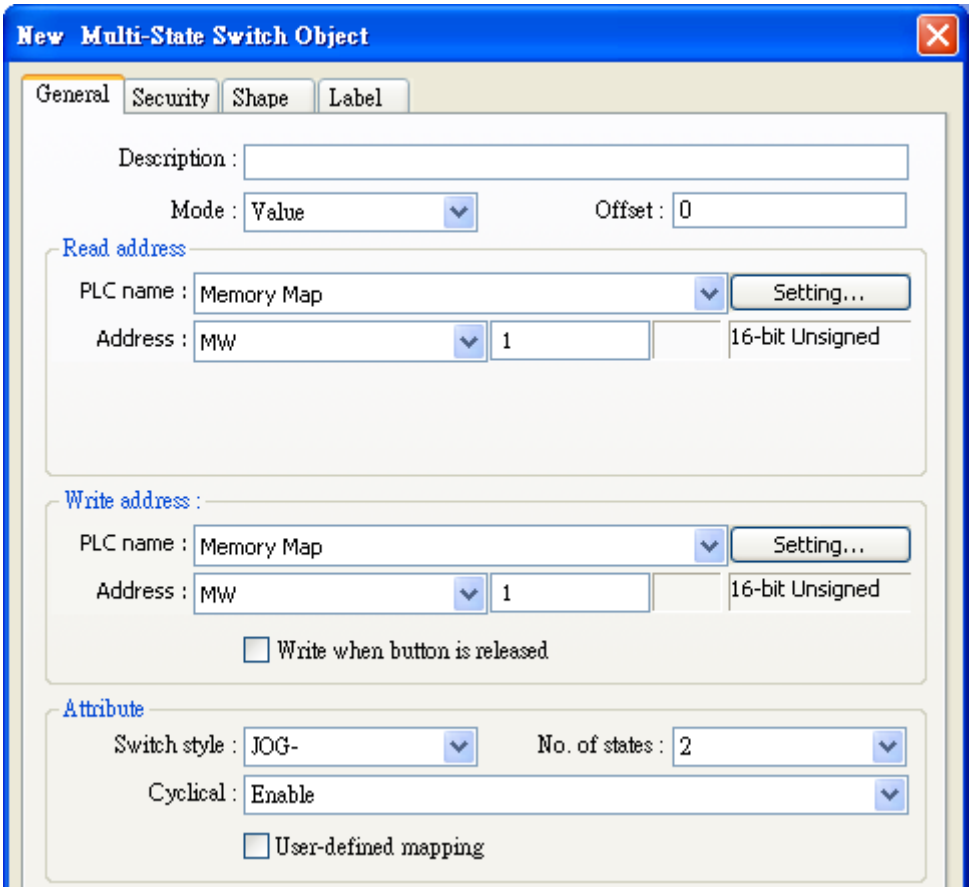
Create a Toggle Switch Object as shown below.

1. Set the **[PLC name]** of read address and write address to **[Memory Map]**.
2. Set **[Address]** to MB-0.
3. Set **[Switch style]** to **[Toggle]**. (The picture and label of the object can be select).



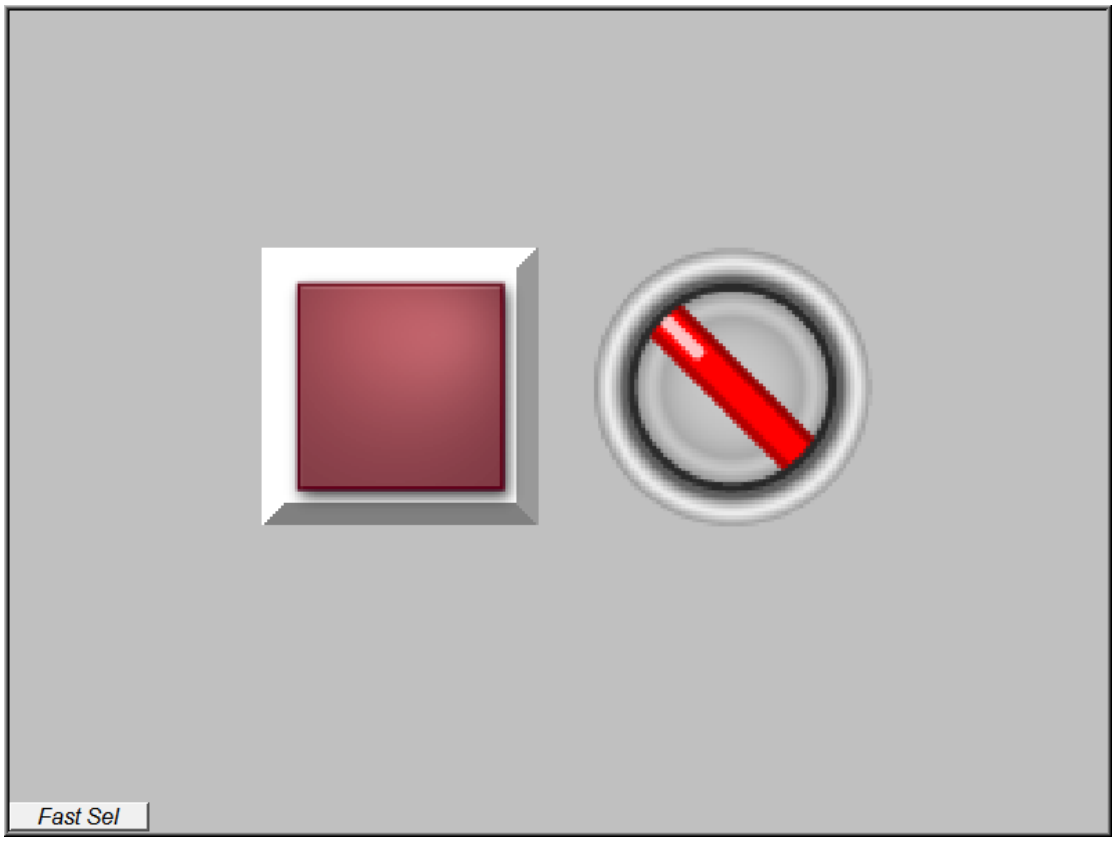
Create a Multi-state Object as shown below.

1. Set the **[PLC name]** of read address and write address to **[Memory Map]**.
2. Set **[Address]** to MW-1.
3. Set **[Cyclical]** to **[Enable]**. (The picture and label of the object can be select).



31.5.3 Execute the Settings

Compile and download the same project to HMI 1 and HMI 2..:

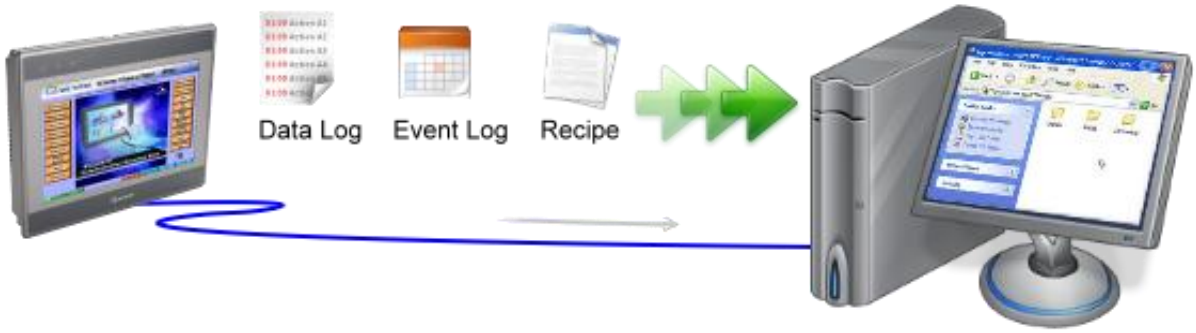


Press the button in one of the HMIs, the status of another one will also be changed.

The way to connect a HMI with a controller is similar to the example above Please make sure that the data in addresses of the two devices are kept identical.

Chapter 32 FTP Server Application

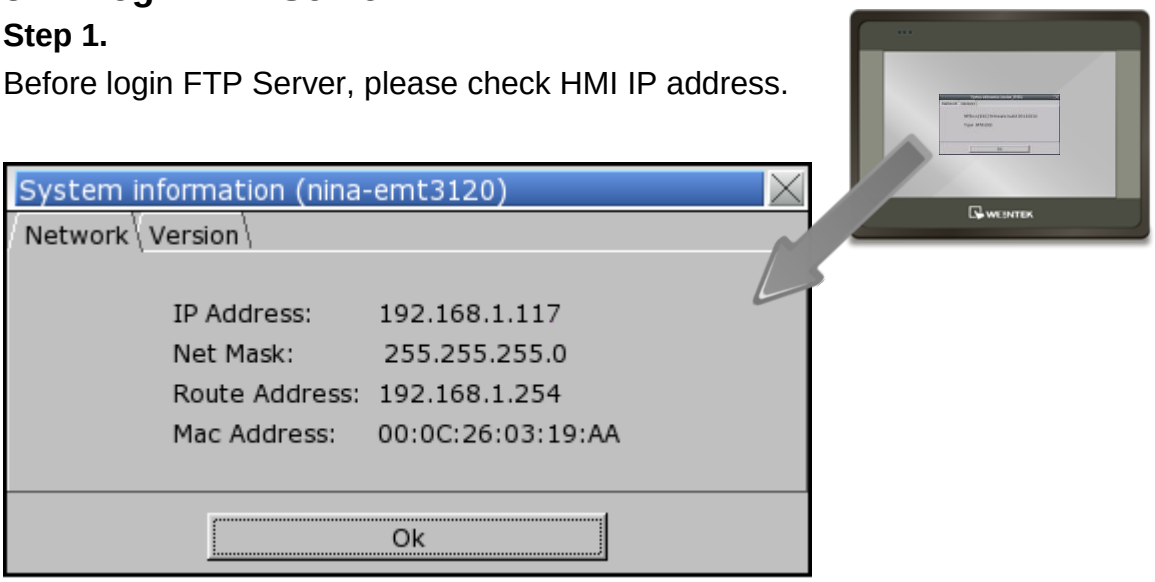
In addition to backup history data from HMI to PC by SD card, USB disk or EasyPrinter, FTP Server can also do this. After downloading project to HMI, FTP Server can be used to backup or update history data and recipe data. The files in FTP Server can't be deleted.



32.1 Login FTP Server

Step 1.

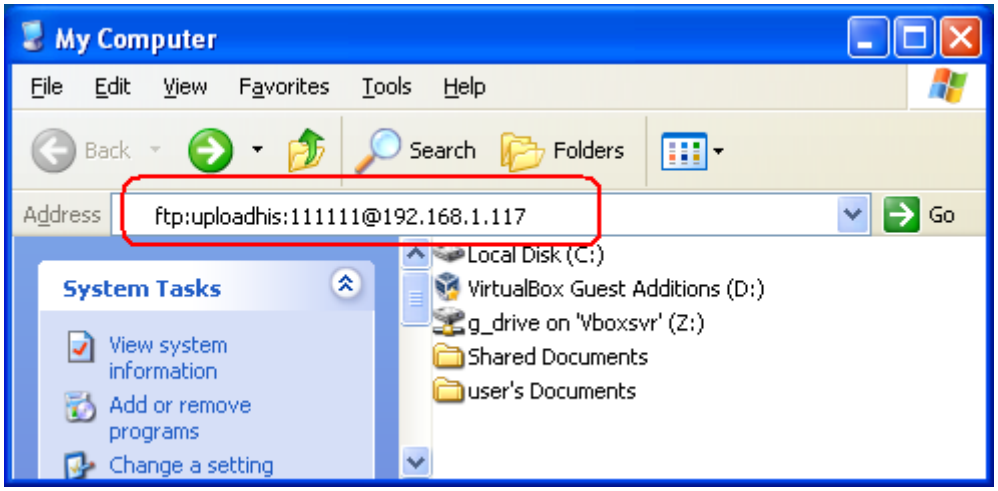
Before login FTP Server, please check HMI IP address.



Step 2.

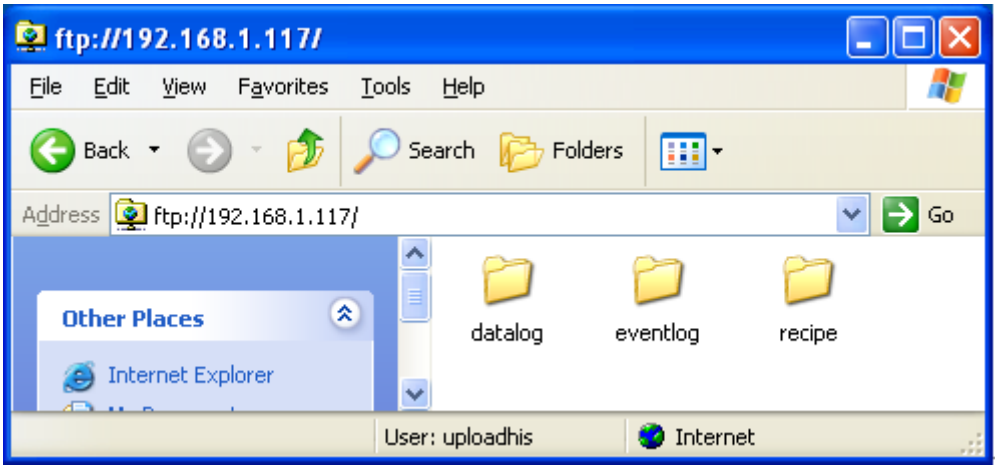
Enter HMI IP: ftp://192.168.1.117/ (example), and log in user name: uploadhis, and the HMI history upload password (if not changed, the default is 111111).

Or, directly enter ftp://uploadhis:111111@192.168.1.117/



Step 3.

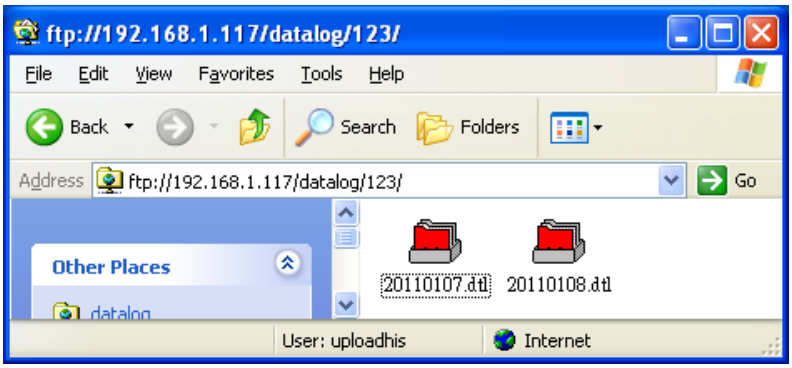
After entering IP, ftp://192.168.1.117/ is shown, and the “datalog”, “eventlog”, and “recipe” folders can be seen.



32.2 Backup History Data and Update Recipe Data

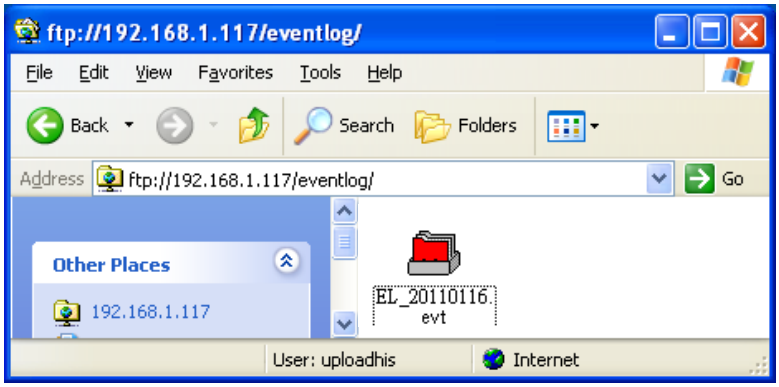
◆ To backup Data Sampling records

1. Click “datalog” folder to check the file names in EasyBuilder.
2. Click on file names to check the content.
3. Copy and paste to save the files to PC.



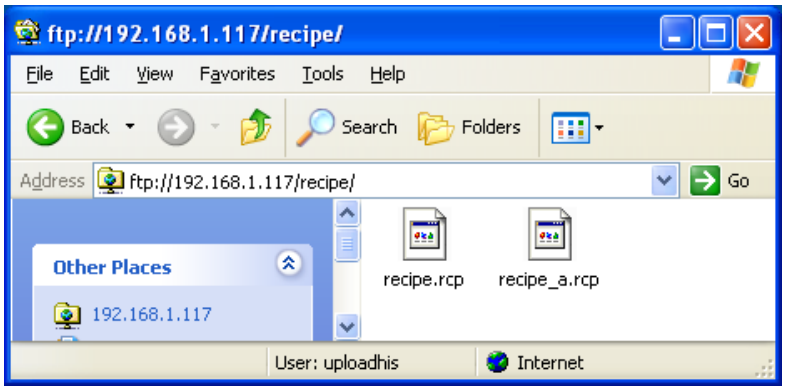
◆ To backup Event (Alarm) Log records

1. Click “eventlog” folder to check the files.
2. Copy and paste to save the files to PC.



◆ To backup or update Recipe records

1. Click “recipe” folder to check the files.
2. Copy and paste to save the files to PC.



- Since recipe data is automatically saved once every minute, after updating “recipe.rcp” or “recipe_a.rcp”, HMI must be restarted in one minute otherwise the new updated recipe data will be overwritten by the former data.
- Use system register [LB-9047] (reboot HMI) and [LB9048] (reboot HMI protection) to reboot HMI. Set [LB-9048] ON first, and then set [LB-9047] ON to restart HMI.

Chapter 33 EasyDiagnoser

33.1 Overview and Configuration

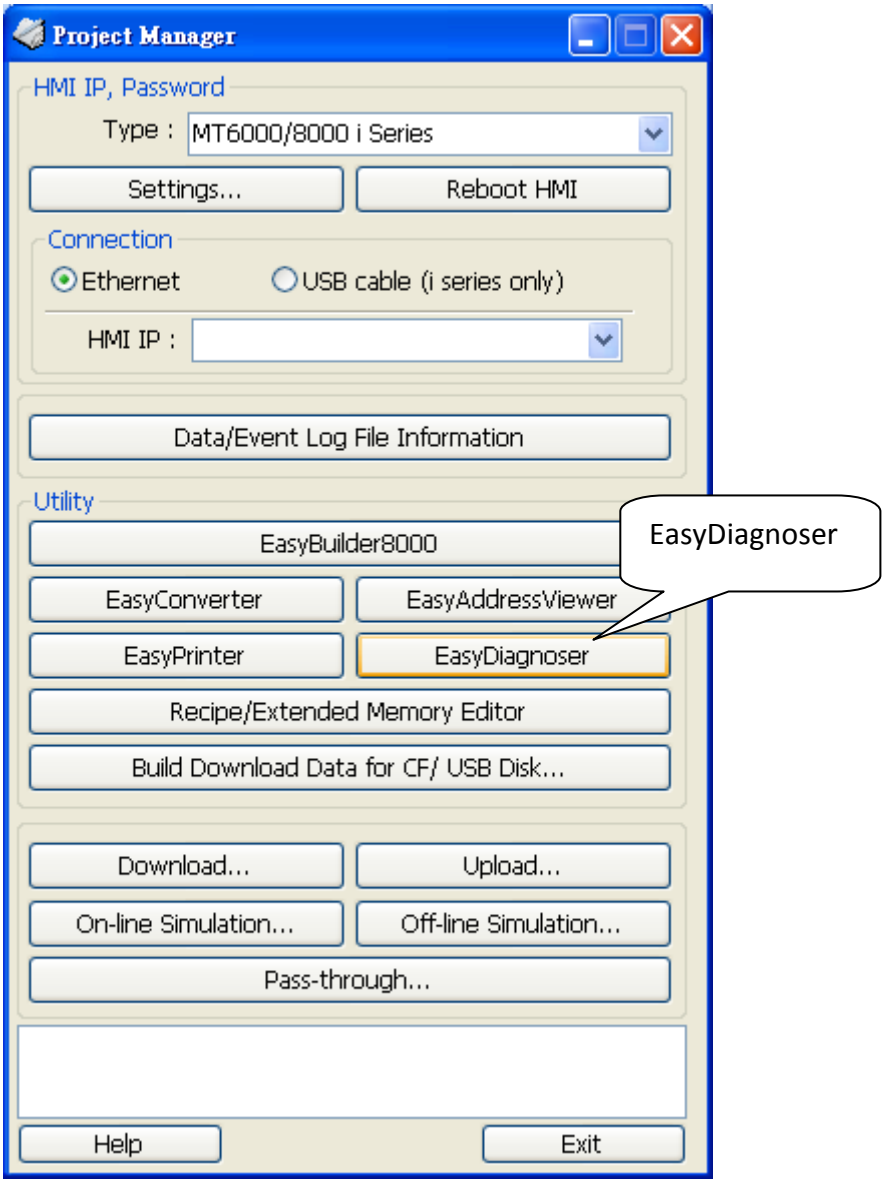
Overview

EasyDiagnoser is a tool for detecting the error occurs while HMI is communicating with PLC.

Configuration

Step 1.

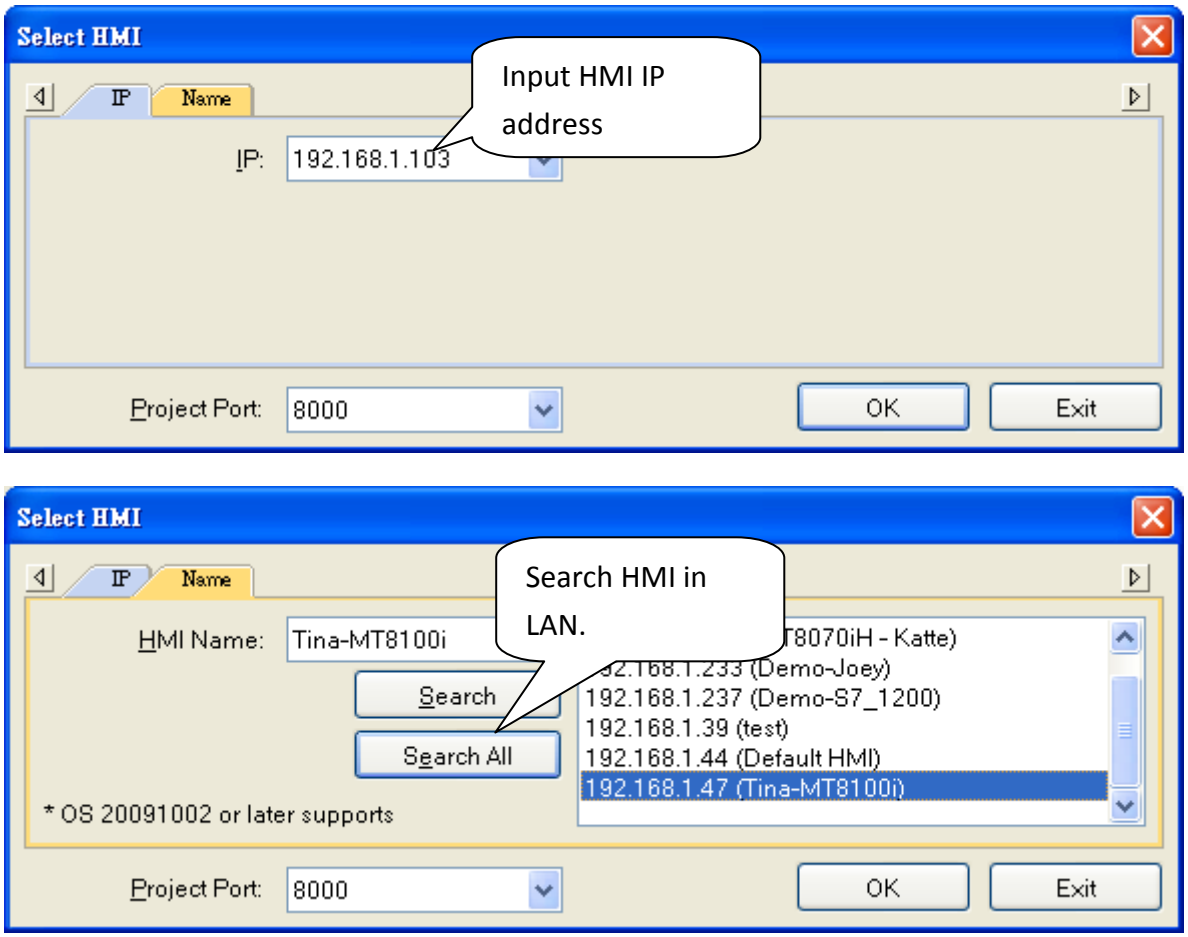
Open Project Manager and click EasyDiagnoser.



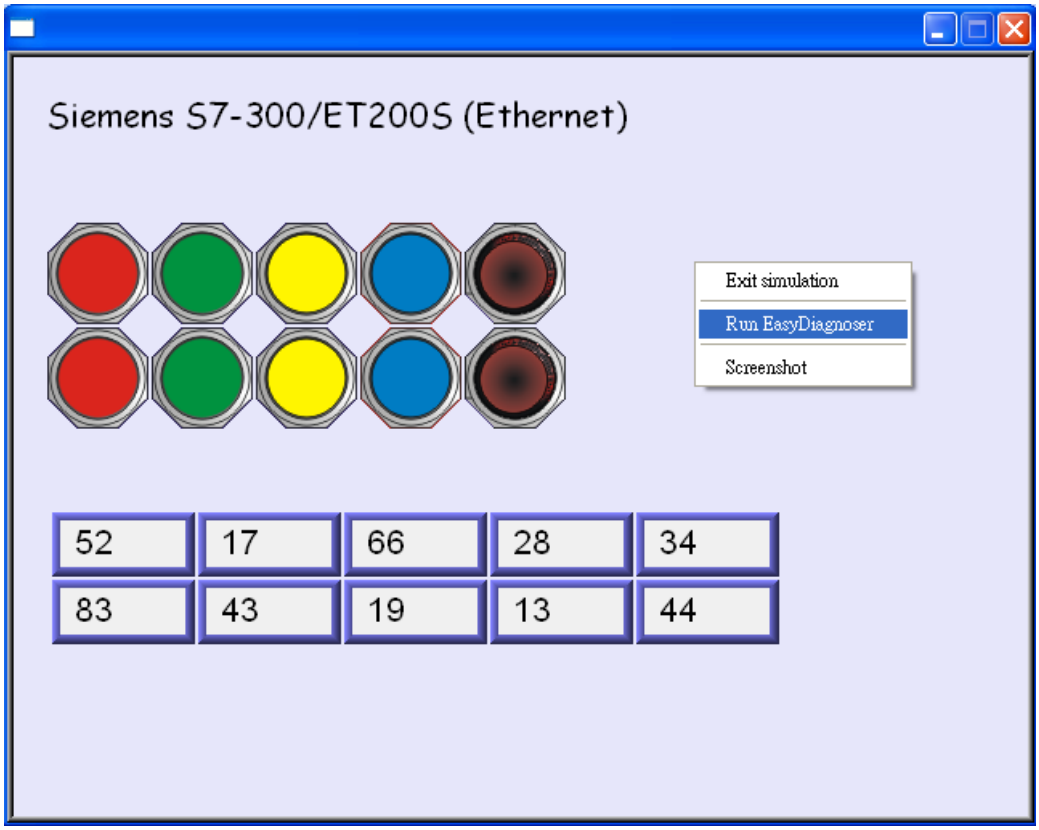
Step 2.

Set the IP address of the HMI to communicate with.

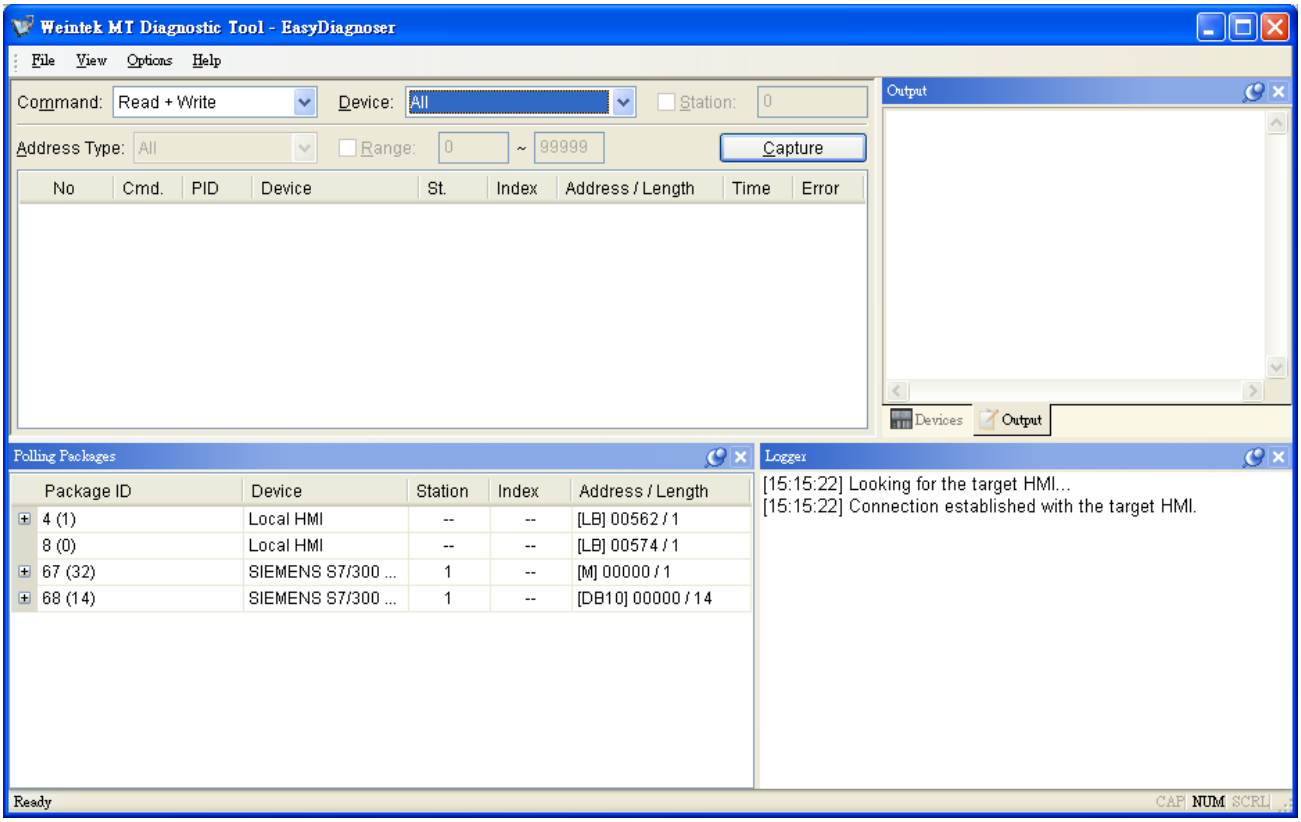
Users can input IP address manually or simply click [Search All]. Please input Project Port as well.



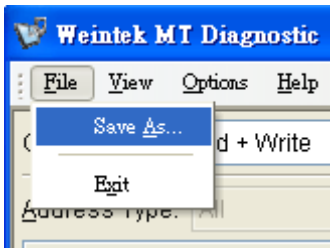
It is also available to right click and select “Run EasyDiagnoser” for entering the setting window when executing On-Line Simulation in EasyBuilder .

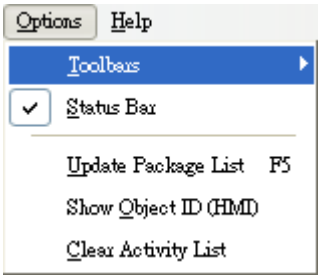
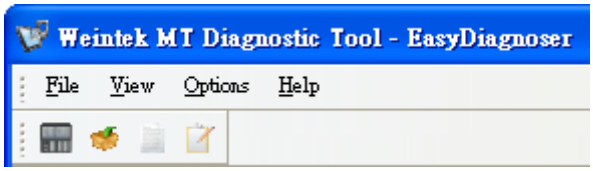
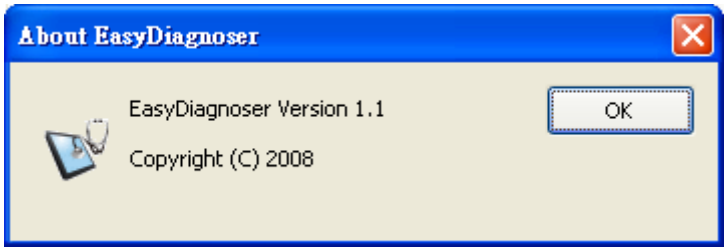


After setting completed, click OK, EasyDiagnoser operation window appears as below:



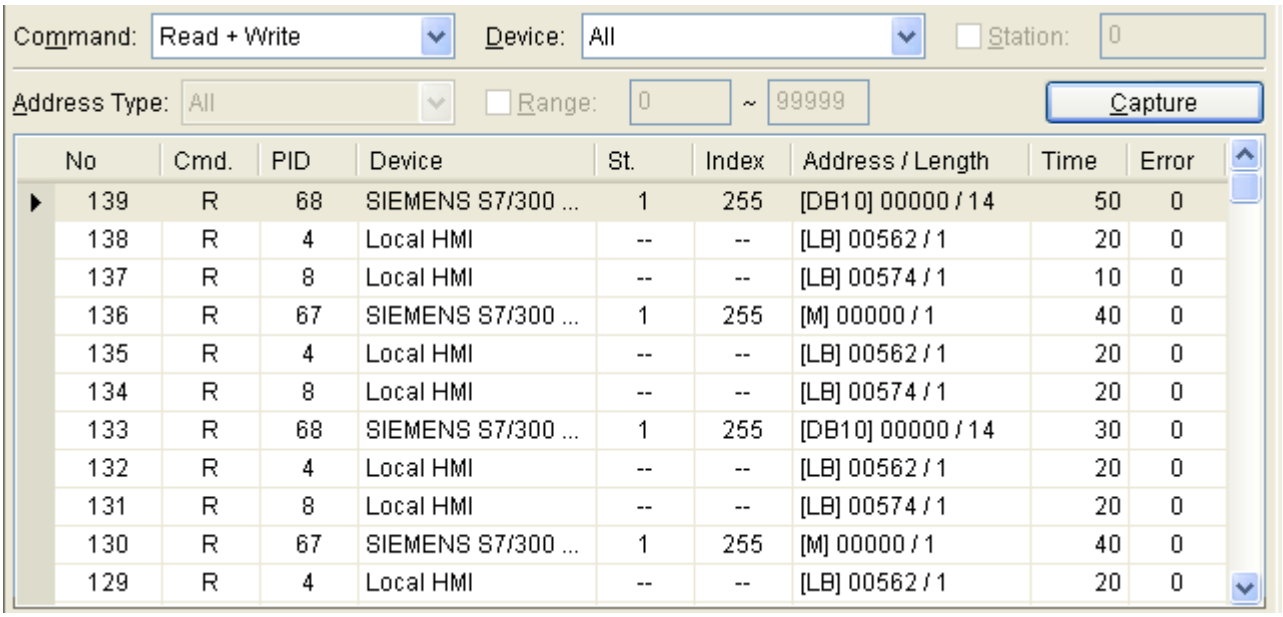
33.2 EasyDiagnoser Settings

Item	Description
File	**Save As** The captured information of Easy Diagnoser can be saved as *.xls which can be read in Excel. 
	Exit Exit current file.
View Device Bar Ctrl+Alt+D Package Bar Ctrl+Alt+P Logger Bar Ctrl+Alt+L Output Bar Ctrl+Alt+O	Click [Device Bar] to display Device window. Click [Package Bar] to display Package window. Click [Logger Bar] to display Logger window. Click [Output Bar] to display Output window.

<table>
<tr><td rowspan="4">Options
 
</td><td>Toolbars
Display toolbar icons of [Device Bar] [Package Bar] [Logger Bar] [Output Bar].
 
</td></tr>
<tr><td>Show Status Bar
At the bottom of EasyDiagnoser window, display information of CAP, NUM, and SCRL.
 
</td></tr>
<tr><td>Update Package List
When users change window on HMI, update the Polling Package information of current window with this list.</td></tr>
<tr><td>Show Object ID (HMI)
Show the ID of objects in HMI as shown below.
 
</td></tr>
<tr><td>Clear Activity List
Clear all information in activity area.</td></tr>
<tr><td>Help</td><td>Display EasyDiagnoser version information.
 
</td></tr>
</table>

- **Activity area**

In the activity area, users can observe the communication between HMI and PLC.



Item	Description
Command	**a. Read + Write** Display Read and Write commands in activity area.
	b. Read Display only Read commands in activity area.
	c. Write Display only Write commands in activity area.
Device	**a. All** Display information of Local HMI and PLC. It depends on the setting of command as following. • If command is set **Read + Write**, the Read and Write information of Local HMI and PLC will be displayed in activity area. • If command is set **Read**, the Read information of Local HMI and PLC will be displayed in activity area. • If command is set **Write**, the Write information of Local HMI and PLC will be displayed in activity area.

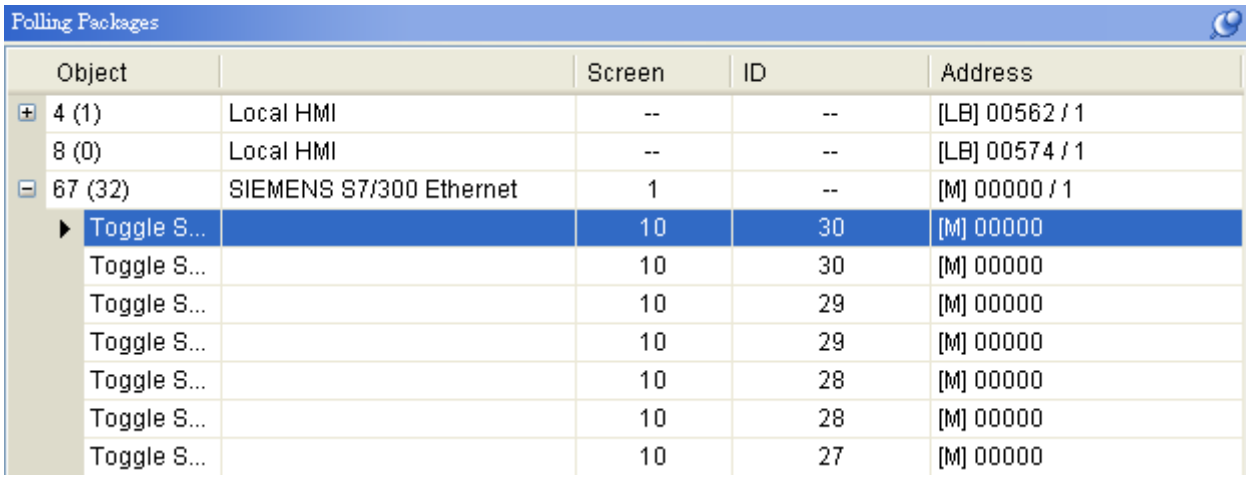
	b. Local HMI Display information of Local HMI, it depends on the setting of command as following. • If command is set **Read + Write**, the Read and Write information of Local HMI will be displayed in activity area. • If command is set **Read**, the Read information of Local HMI will be displayed in activity area. • If command is set **Write**, the Write information of Local HMI will be displayed in activity area.
	c. PLC Display information of PLC, it depends on the setting of command as following. • If command is set **Read + Write**, the Read and Write information of PLC will be displayed in activity area. • If command is set **Read**, the Read information of PLC will be displayed in activity area. • If command is set **Write**, the Write information of PLC will be displayed in activity area.
Station	Select specific Station for display on the screen. (This function will be disabled when selecting [All] in Device).
Address Type	Users can select all or a part of address types to be displayed on the screen. (This function will be disabled when selecting [All] in Device).
Range	Set the range of address types to be displayed. (This function will be disabled when selecting [All] in Address Type).
Capture	Click to start/stop capturing communication message.
Error	Please refer to the section coming later.

- **Polling Packages**

Polling Packages

Package ID	Device	Station	Index	Address / Length
4 (1)	Local HMI	--	--	[LB] 00562 / 1
8 (0)	Local HMI	--	--	[LB] 00574 / 1
67 (32)	SIEMENS S7/300 Ethernet	1	--	[M] 00000 / 1
68 (3)	SIEMENS S7/300 Ethernet	1	10	[DB10] 00000 / 3
69 (3)	SIEMENS S7/300 Ethernet	1	11	[DB10] 00003 / 3
70 (3)	SIEMENS S7/300 Ethernet	1	12	[DB10] 00006 / 3
71 (5)	SIEMENS S7/300 Ethernet	1	--	[DB10] 00009 / 5

Item	Description
Package ID	Use the information of package ID to check the PID in activity area for finding the problem.
Device	Displays HMI and PLC type.
Station	Displays PLC station number.
Index	Display objects-used index register numbers.
Address/Length	Displays device type address. Length-how many words of the Package.



Item	Description
Object	Package ID where this object is placed.
Screen	Window in the project where this object is placed.
ID	ID of the object.
Address	Address of the object.

Note:

a. Click **[Package ID]**, the device station number will be displayed in 3rd column.

Polling Packages

	Package ID	Device	Station	Index	Address / Length
⊞	4 (1)	Local HMI	--	--	[LB] 00562 / 1
	8 (0)	Local HMI	--	--	[LB] 00574 / 1
⊞	67 (32)	SIEMENS S7/300 Ethernet	1	--	[M] 00000 / 1
⊞	68 (3)	SIEMENS S7/300 Ethernet	1	10	[DB10] 00000 / 3

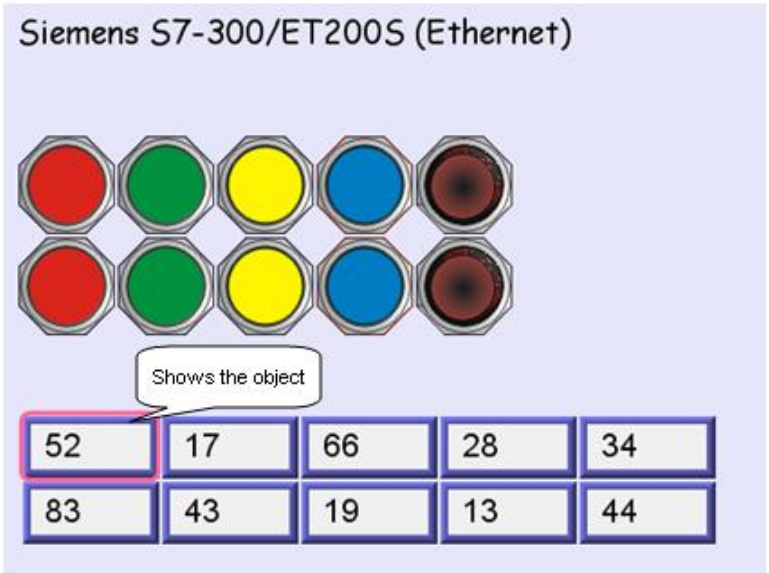
b. Double click **[Package ID]** then select **[object]**, the 1st column directs the object's position.

For example, select [Numeric Input] and the screen no. displays 10.

This shows that this object is in window no. 10 in the project and will be marked with pink frame in HMI as shown below.

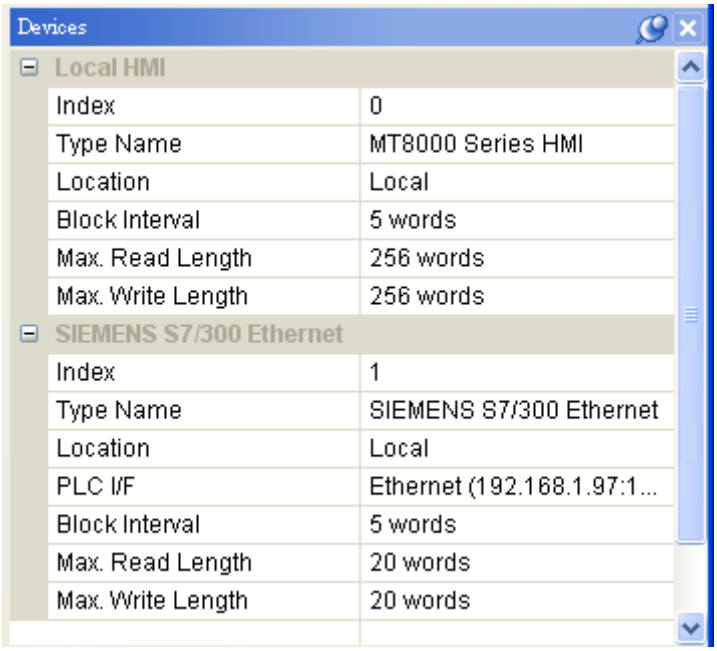
Polling Packages

	Object		Screen	ID	Address
⊞	4 (1)	Local HMI	--	--	[LB] 00562 / 1
	8 (0)	Local HMI	--	--	[LB] 00574 / 1
⊞	67 (32)	SIEMENS S7/300 Ethernet	1	--	[M] 00000 / 1
⊟	68 (3)	SIEMENS S7/300 Ethernet	1	10	[DB10] 00000 / 3
▶	Numeric I...		10	2	[DB10] 00000
	Numeric I...		10	3	[DB10] 00001
	Numeric I...		10	4	[DB10] 00002



- **Devices**

  Devices window displays information of HMI and PLC.



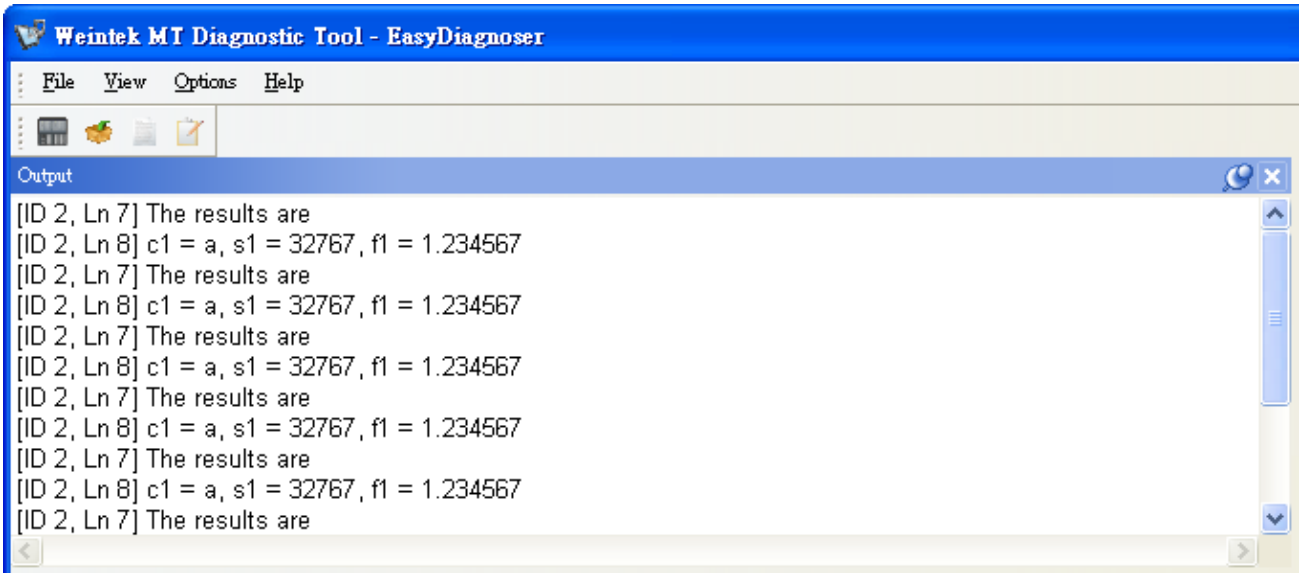
- **Output (Macro debug)**

With Trace function offered by Macro, the executing status of Macro can be seen. Please refer to EasyBuilder User's Manual *"Chapter 18 MACRO"* for more information.

In illustration below, for [ID 2, Ln 7] and [ID 2, Ln 8]

ID 2 represents Macro name.

Ln 7 and Ln 8 represent that they are in 7th and 8th lines of Macro.

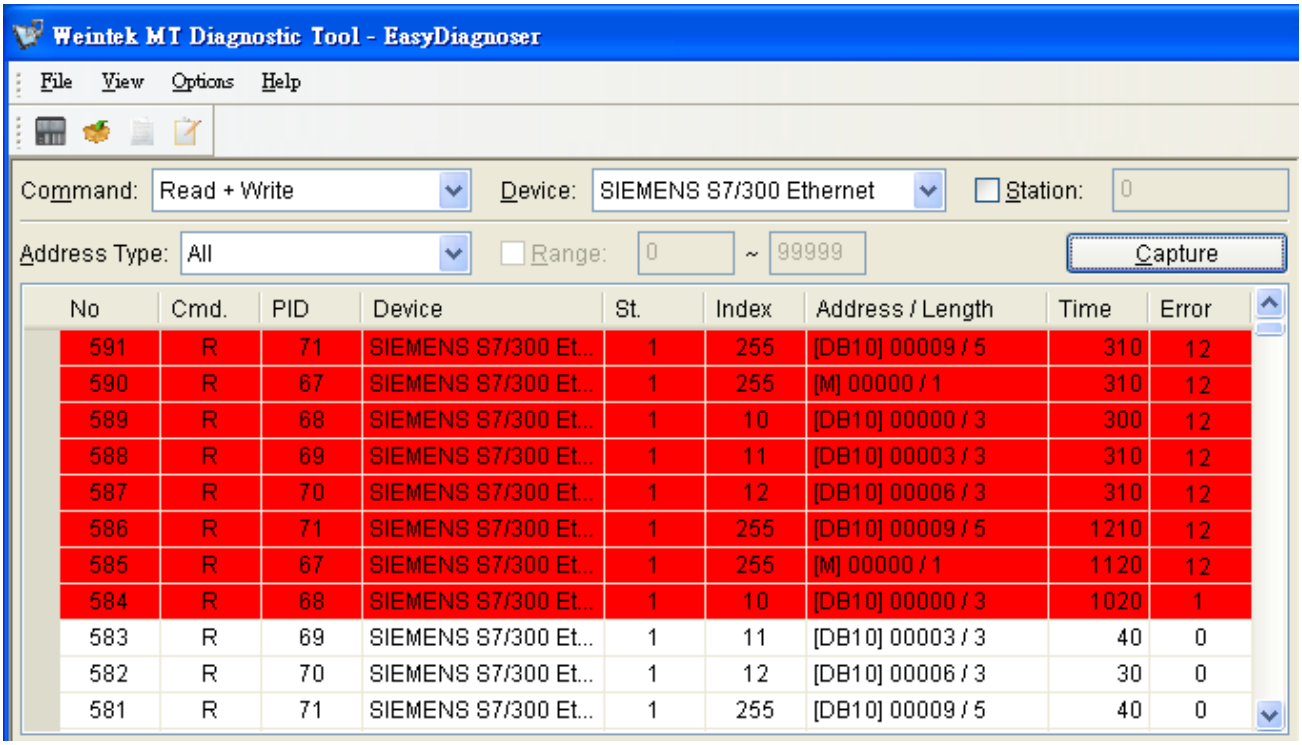


33.3 Error Code

In activity area, users can find the reason of error through error codes listed below.

0: Normal
1: Time out
2: Fail Error
12: Ignore

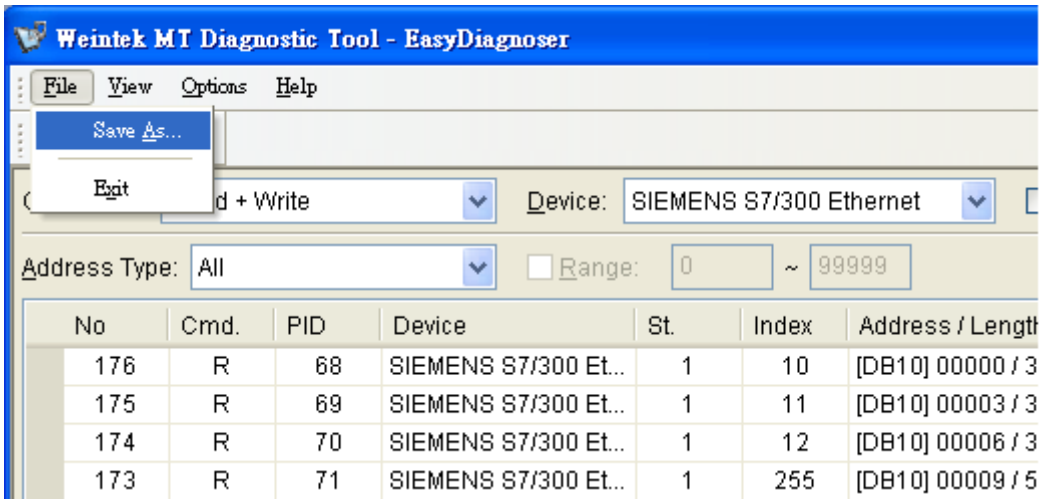
When error occurs, error message will be shaded red as shown below.
The error code is 1 since PLC is disconnected with HMI.
The error code is 12 since “PLC No Response” message window is shown.



No	Cmd.	PID	Device	St.	Index	Address / Length	Time	Error
591	R	71	SIEMENS S7/300 Et...	1	255	[DB10] 00009 / 5	310	12
590	R	67	SIEMENS S7/300 Et...	1	255	[M] 00000 / 1	310	12
589	R	68	SIEMENS S7/300 Et...	1	10	[DB10] 00000 / 3	300	12
588	R	69	SIEMENS S7/300 Et...	1	11	[DB10] 00003 / 3	310	12
587	R	70	SIEMENS S7/300 Et...	1	12	[DB10] 00006 / 3	310	12
586	R	71	SIEMENS S7/300 Et...	1	255	[DB10] 00009 / 5	1210	12
585	R	67	SIEMENS S7/300 Et...	1	255	[M] 00000 / 1	1120	12
584	R	68	SIEMENS S7/300 Et...	1	10	[DB10] 00000 / 3	1020	1
583	R	69	SIEMENS S7/300 Et...	1	11	[DB10] 00003 / 3	40	0
582	R	70	SIEMENS S7/300 Et...	1	12	[DB10] 00006 / 3	30	0
581	R	71	SIEMENS S7/300 Et...	1	255	[DB10] 00009 / 5	40	0

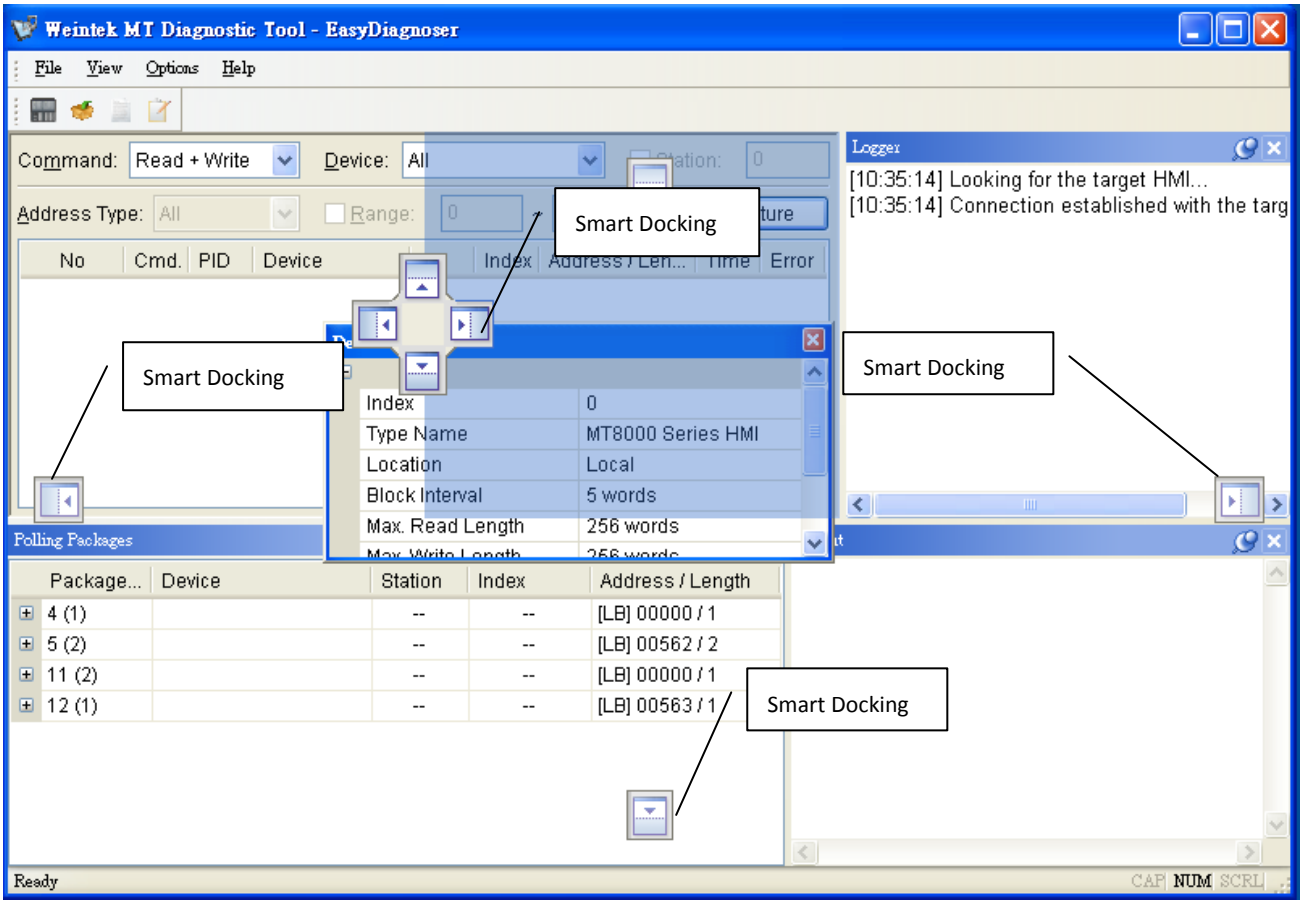
33.4 Save As

The captured information of Easy Diagnoser can be saved as *.xls which can be read in Excel.



33.5 Window Adjustment

Users can drag or use smart docking icons in editing window to place the windows to the desired position.

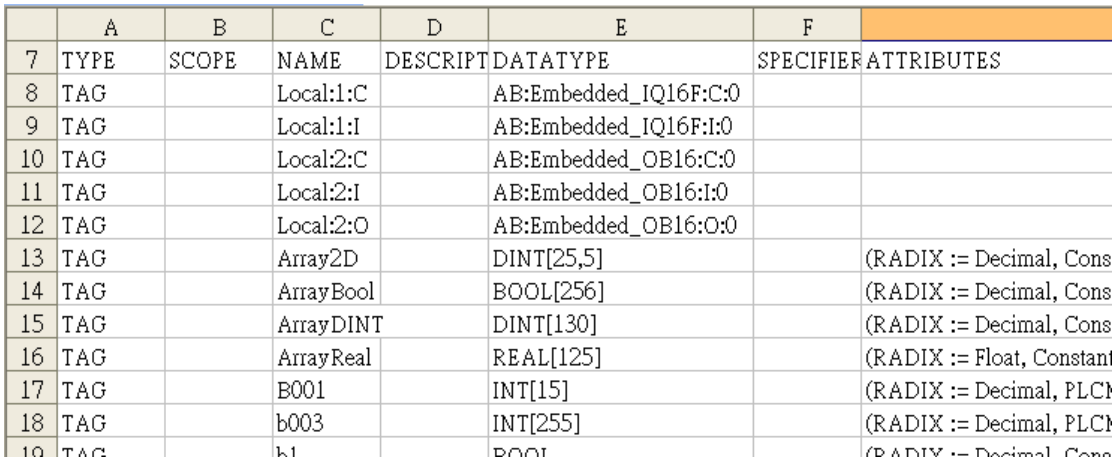


Note:

EasyDiagnoser doesn't support Siemens S7/1200 (Ethernet) and Allen-Bradley Ethernet/IP (CompactLogix/ControlLogix) – Free Tag Names since both of the PLC use tag.

Chapter 34 Rockwell EtherNet/IP Free Tag Names

When using the driver of Rockwell EtherNet/IP Free Tags (CompactLogix/ControlLogix), the User-defined tag in RSLogix5000 can be exported to csv file, and then import to EasyBuilder. However, the data types: User-Defined, Predefined and Module-Defined cannot be exported.



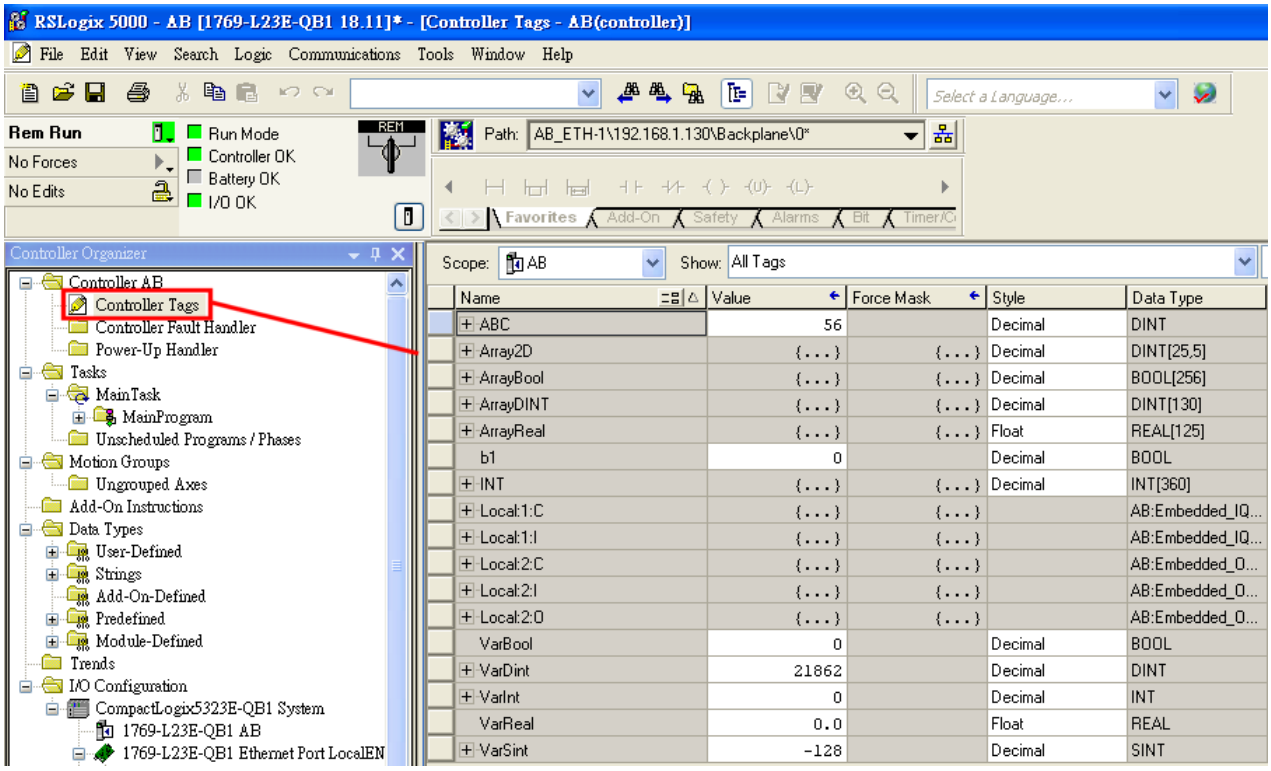
	A	B	C	D	E	F	
7	TYPE	SCOPE	NAME	DESCRIPT	DATATYPE	SPECIFIER	ATTRIBUTES
8	TAG		Local:1:C		AB:Embedded_IQ16F:C:0		
9	TAG		Local:1:I		AB:Embedded_IQ16F:I:0		
10	TAG		Local:2:C		AB:Embedded_OB16:C:0		
11	TAG		Local:2:I		AB:Embedded_OB16:I:0		
12	TAG		Local:2:O		AB:Embedded_OB16:O:0		
13	TAG		Array2D		DINT[25,5]		(RADIX := Decimal, Cons
14	TAG		ArrayBool		BOOL[256]		(RADIX := Decimal, Cons
15	TAG		ArrayDINT		DINT[130]		(RADIX := Decimal, Cons
16	TAG		ArrayReal		REAL[125]		(RADIX := Float, Constant
17	TAG		B001		INT[15]		(RADIX := Decimal, PLCI
18	TAG		b003		INT[255]		(RADIX := Decimal, PLCI
19	TAG		b1		BOOL		(RADIX := Decimal, Cons

Therefore, Structure Editor in EasyBuilder is for importing and editing User-Defined, Predefined and Module-Defined tags.

34.1 Import User-Defined AB Tag CSV File to EasyBuilder

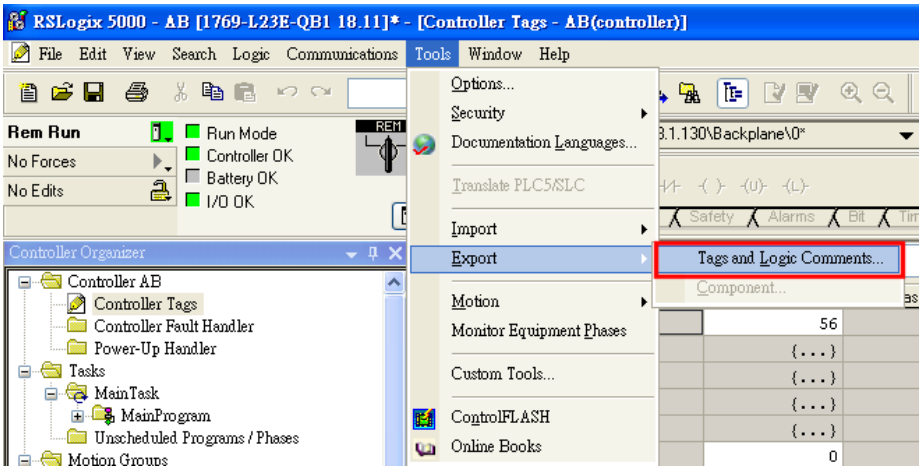
Step 1

Create Tags in RSLogix5000.



Step 2

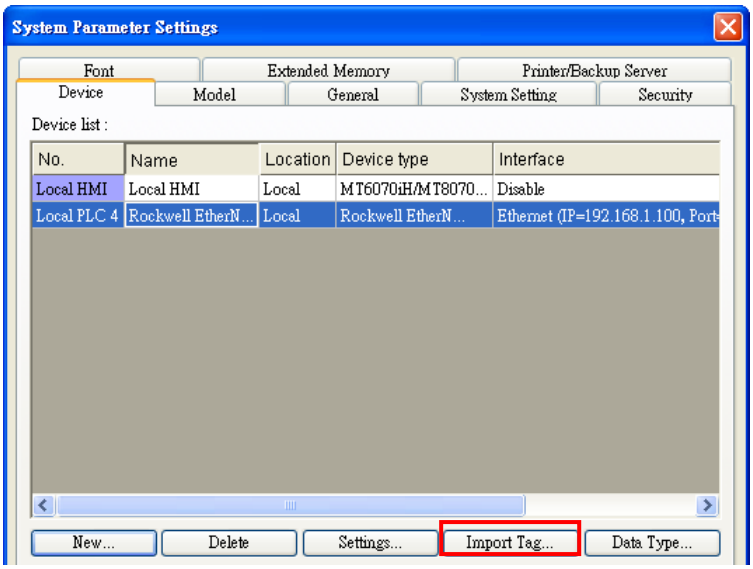
Export Tags to csv file.

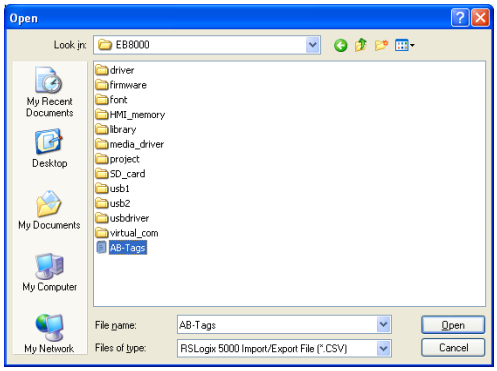


Step 3

In EasyBuilder, add Rockwell EtherNet/IP-Tag (CompactLogix/ControlLogix) driver.

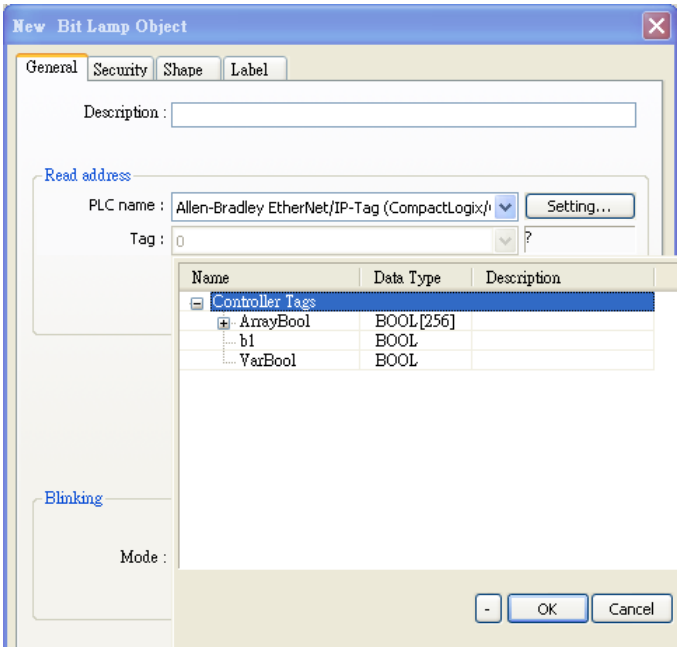
Enter PLC IP address and click [Import Tag].





Step 4

In the object setting dialog, select the PLC type, and select a controller tag.

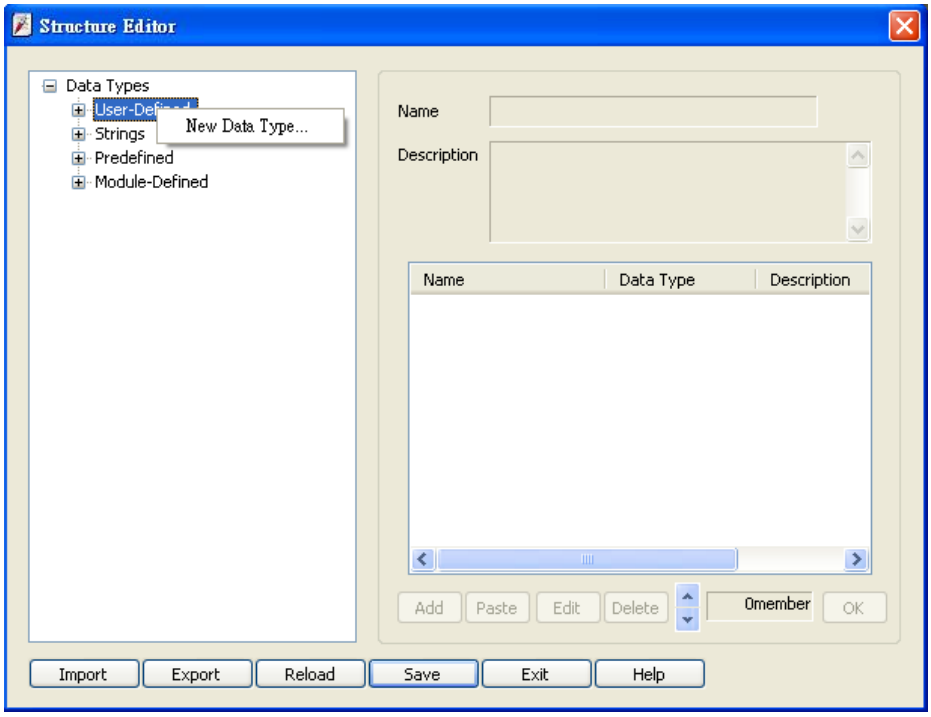


34.2 Adding a New Data Type

Structure Editor is located in the installation directory of EasyBuilder. Double-click Structure Editor.exe and the editor window will show as below.

Step 1

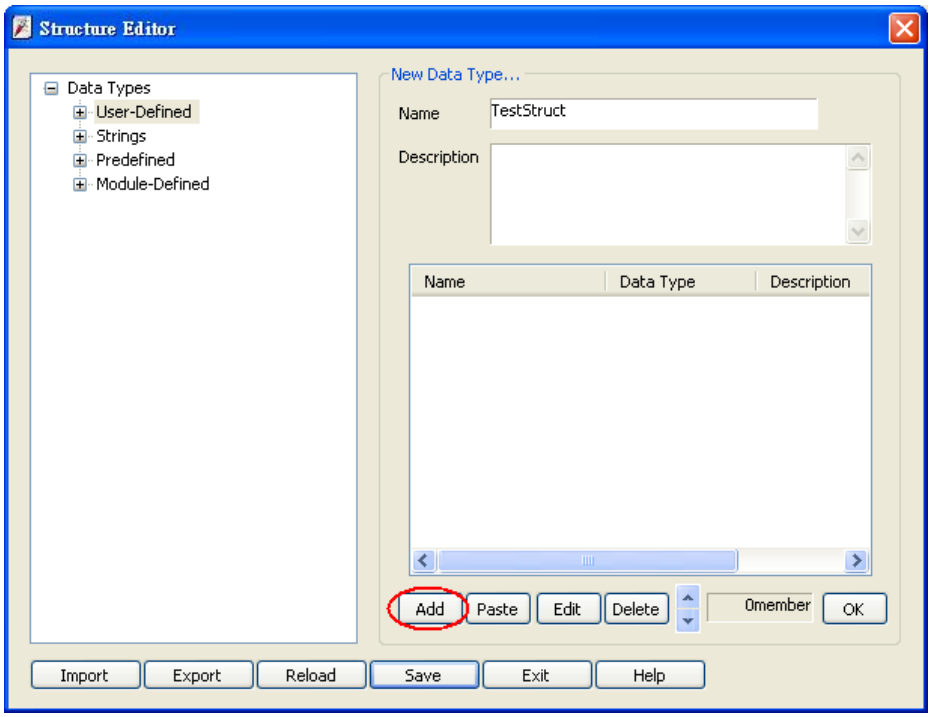
Right click on the assigned data type (usually labeled as User-Defined), then click **[New Data Type]** to start editing.



Step 2

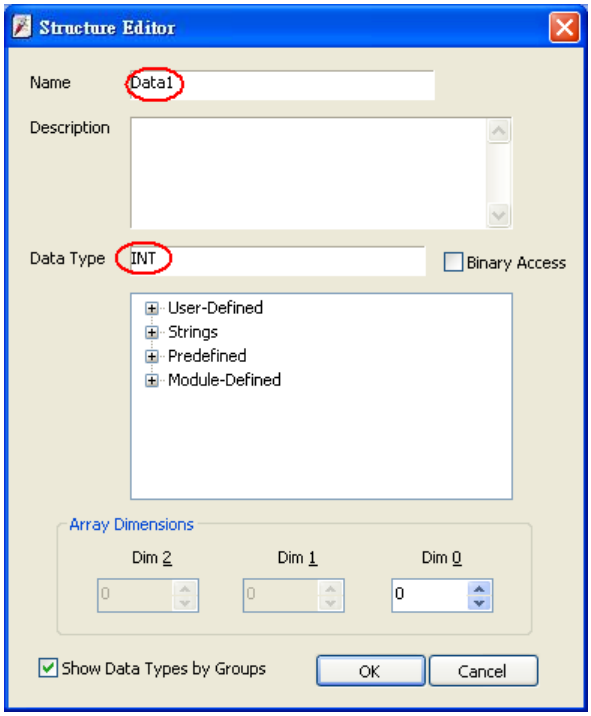
Enter the name of the data type. **[Description]** can be left blank.

To add a sub-item, click **[Add]**.



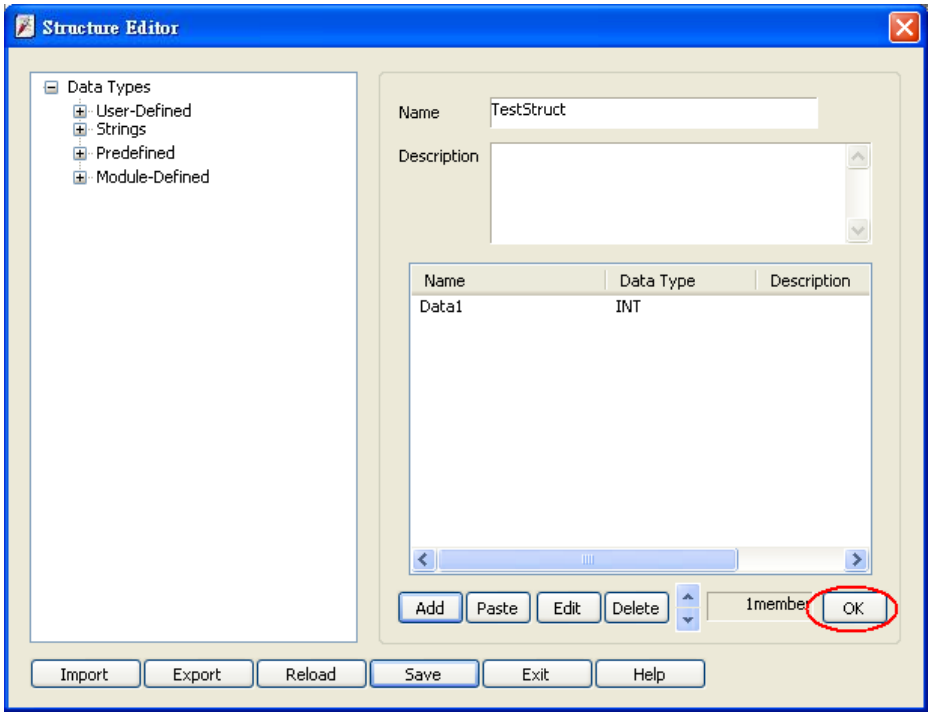
Step 3

Enter the name and the data type then click **[OK]**.



Step 4

After adding all sub-items, click **[OK]**. The built data type will be listed on the left side.

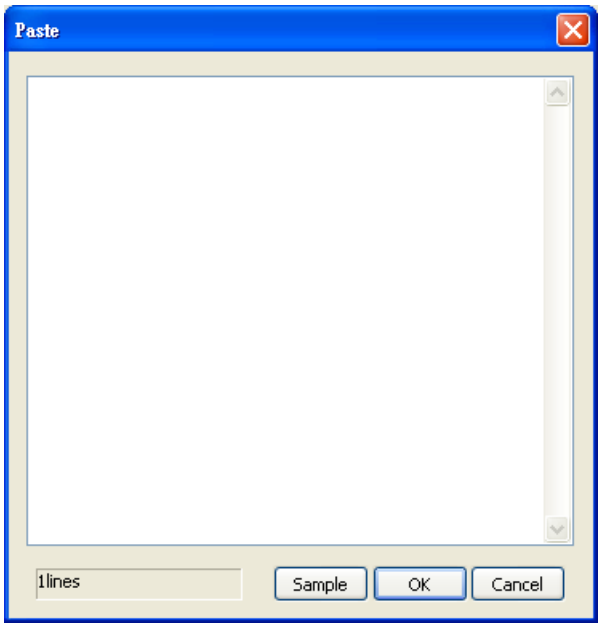


After changing the name or description of a data type, click **[OK]** to update.

34.3 Paste

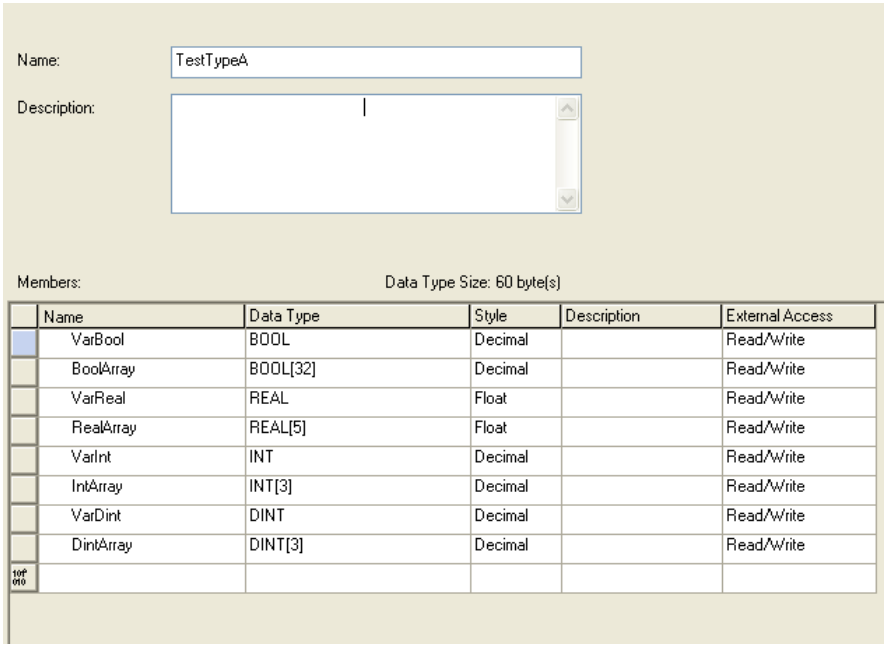
Step 1

When adding a patch of sub-items, this function allows users to add multiple data at one time. First, click **[Paste]** in the main menu.



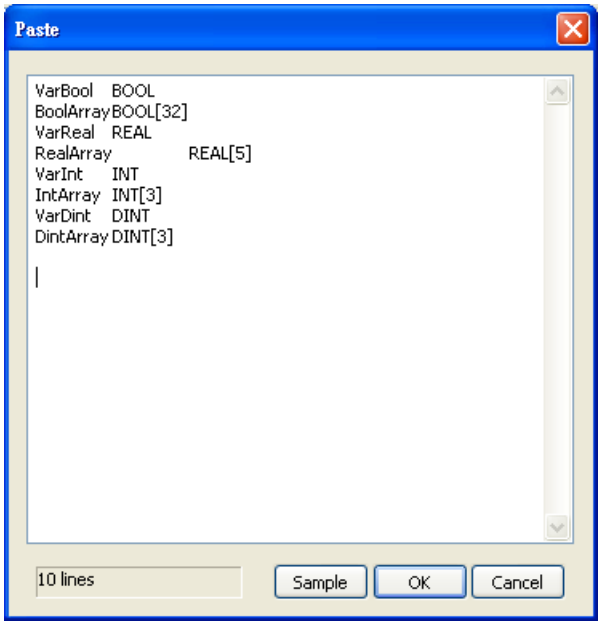
Step 2

Type in data name and data type in each line first. It is recommended to directly copy and paste from RSLogix5000 to avoid errors. Users can click **[Sample]** for reference.



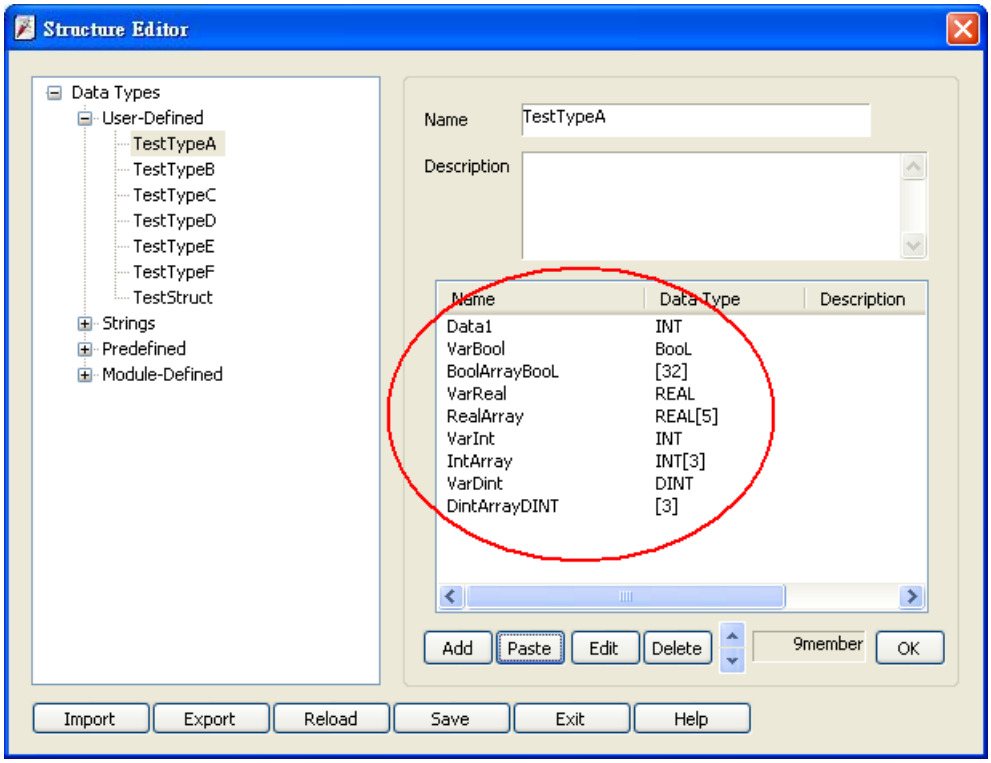
Step 3

Select needed Name and Data defined in RSLogix as the table above. To select all the items, press and hold on the first line, then slide down to the bottom until the scroll rolls to the end then stop holding. Press Ctrl-C to copy and Ctrl-V to paste in the editing window, as shown below.



Step 4

Click **[OK]** to finish setting then return to the main menu to check the added data.



34.4 Miscellaneous

- Revising sub-item data:

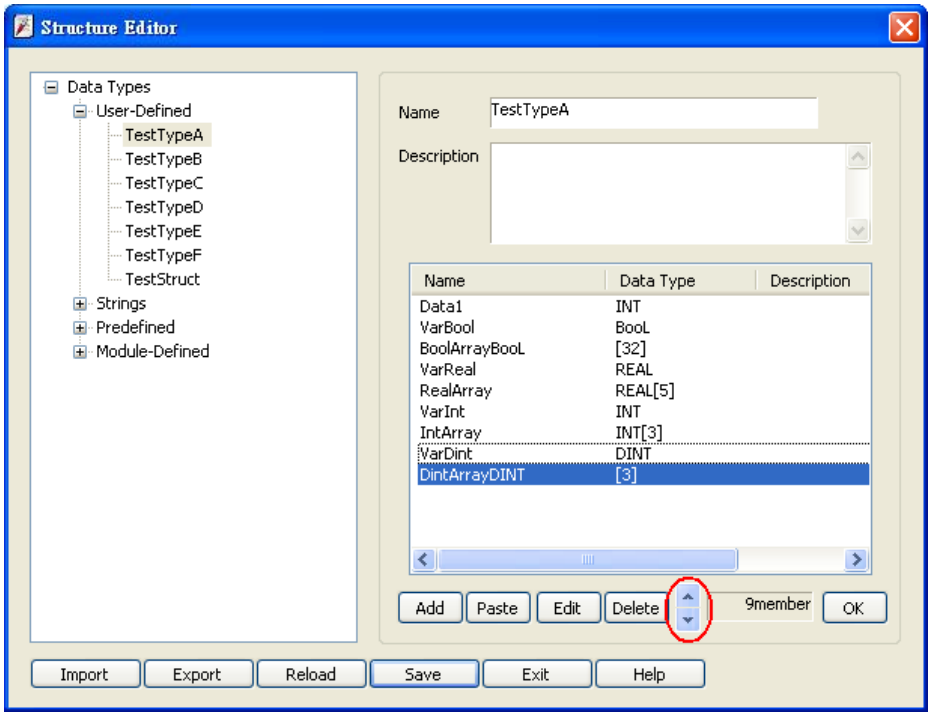
Double click on the sub-item to be revised in the main menu, or click on the sub-item then click **[Edit]**.

- Deleting sub-item data:

Select the data to be deleted then click **[Delete]**. To delete all sub-items, press and hold the Delete button on the keyboard then click the **[Delete]** button in the main menu.

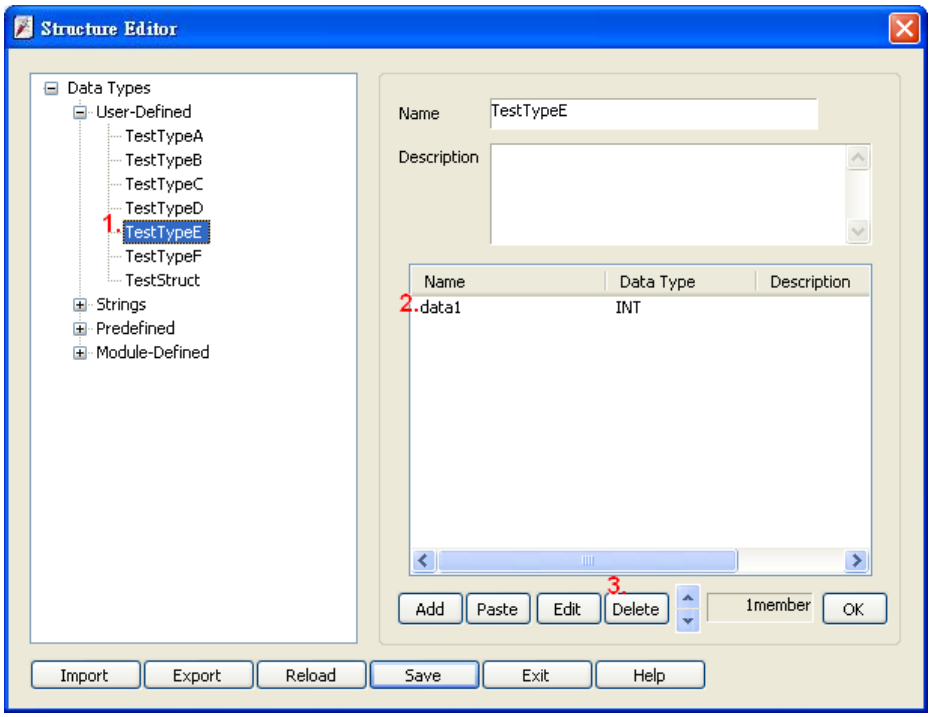
- Adjusting the order of sub-item data:

After selecting a single sub-item, use the move up and move down buttons in main menu to change the order. This makes selecting items in EasyBuilder easier.



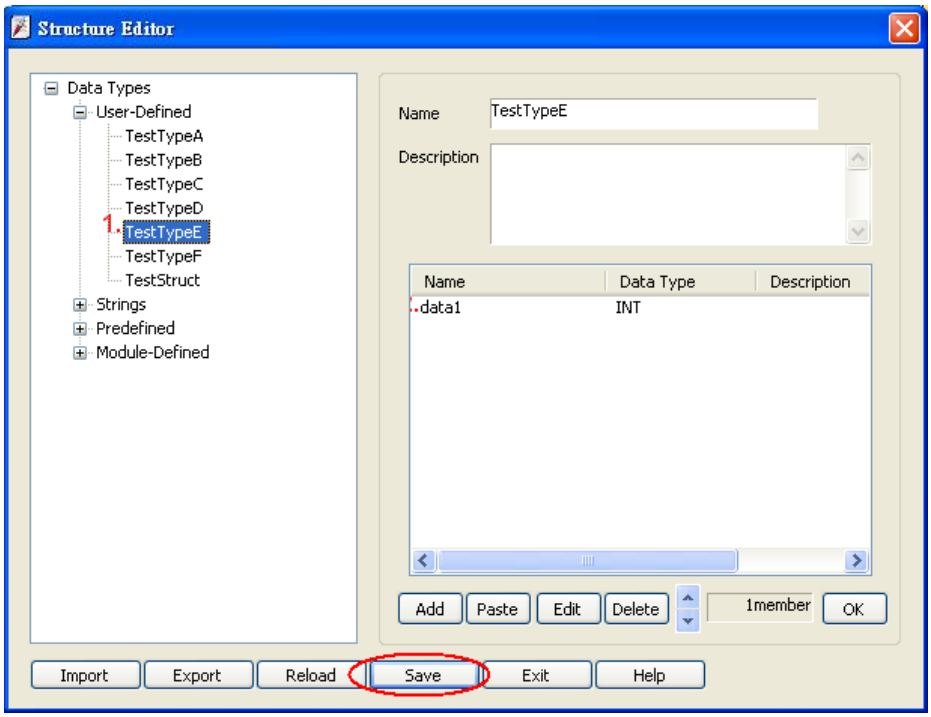
- Deleting data type:

Select from the list on the left side of the main menu and then select the data type to be deleted on the right side then press Delete on the keyboard. The data type can then be deleted.



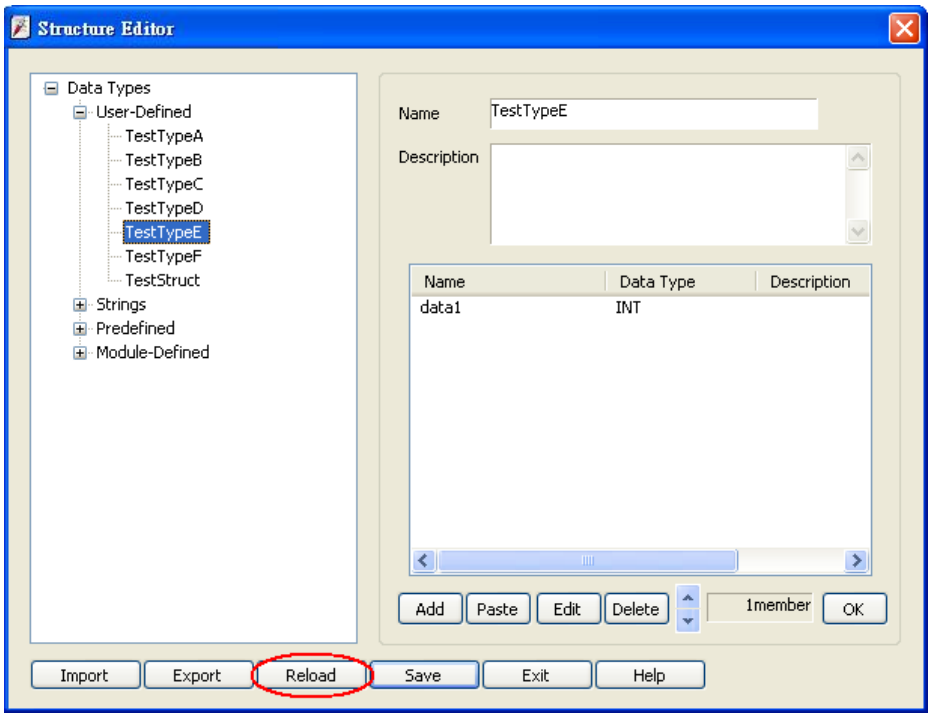
- Saving the revision:

After revising, click **[Save]** in the main menu and then restart EasyBuilder to check the result.



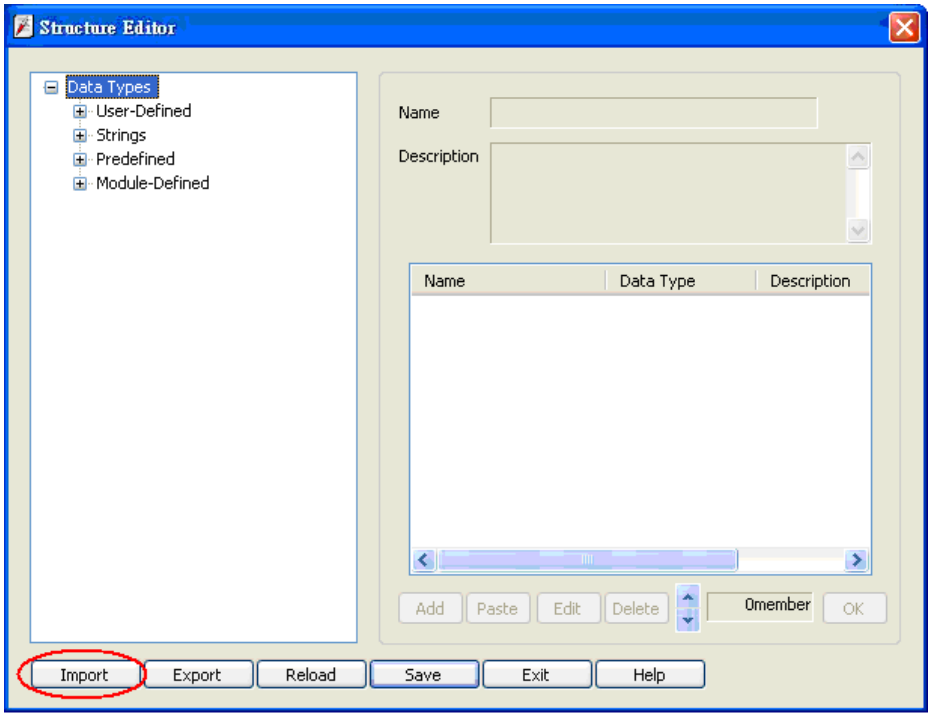
- Reload:

To abandon all the changes and re-edit, click **[Reload]** button in main menu.



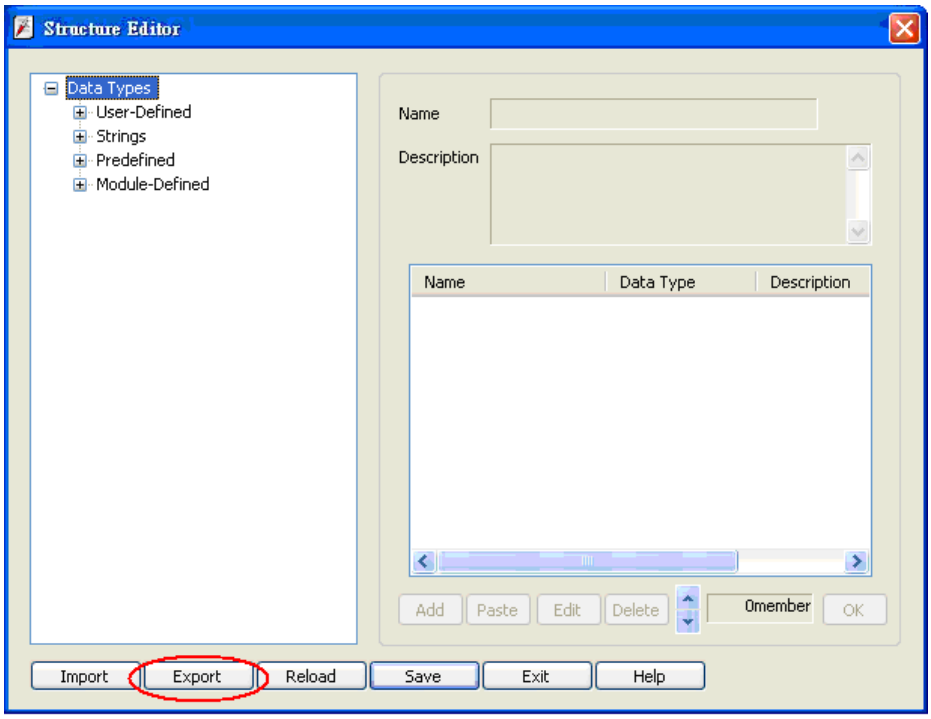
- Import:

Import TDF files.



- Export:

Export the edited data to *.tdf file, the exported file can be used in other PC or saved as a backup.

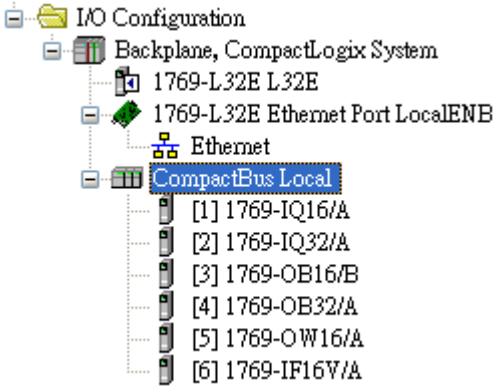


34.5 Module-Defined

Module-Defined is a default structure of a module.

Here is an example showing how to define the default structure of a module.

In RSLogix5000 [I/O Configuration], the I/O module is set.

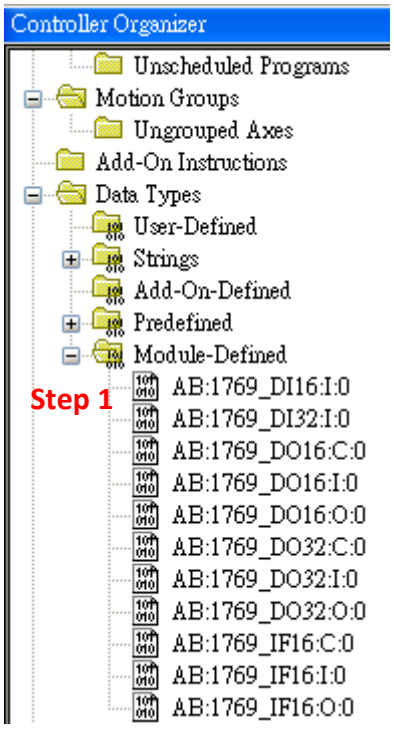


The Tags of these modules won't list the structure when exported to csv file. Therefore, users should define it first.

	A	B	C	D	E	F	G	H
7	TYPE	SCOPE	NAME	DESCRIPT	DATATYPE	SPECIFIER	ATTRIBUTES	
8	TAG		Local:1:I		AB:1769_DI16:I:0			
9	TAG		Local:2:I		AB:1769_DI32:I:0			
10	TAG		Local:3:C		AB:1769_DO16:C:0			
11	TAG		Local:3:I		AB:1769_DO16:I:0			
12	TAG		Local:3:O		AB:1769_DO16:O:0			
13	TAG		Local:4:C		AB:1769_DO32:C:0			
14	TAG		Local:4:I		AB:1769_DO32:I:0			
15	TAG		Local:4:O		AB:1769_DO32:O:0			
16	TAG		Local:5:C		AB:1769_DO16:C:0			
17	TAG		Local:5:I		AB:1769_DO16:I:0			
18	TAG		Local:5:O		AB:1769_DO16:O:0			
19	TAG		Local:6:C		AB:1769_IF16:C:0			
20	TAG		Local:6:I		AB:1769_IF16:I:0			
21	TAG		Local:6:O		AB:1769_IF16:O:0			

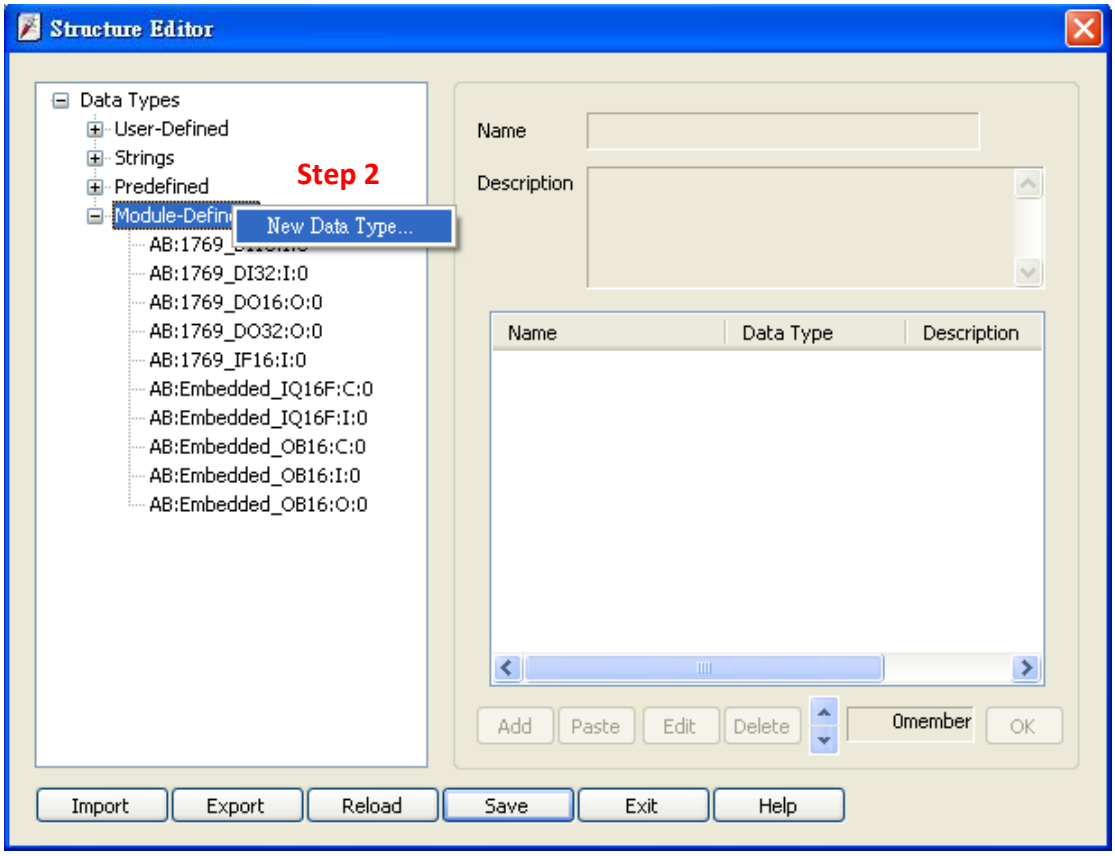
Step 1

In RSLogix5000 **[Controller Organizer] » [Data Types] » [Module-Defined]**, double click Data Type of the module. Sub-items of the module will be shown in a popup dialog. Copy the Name and Data Type of the members.



Step 2

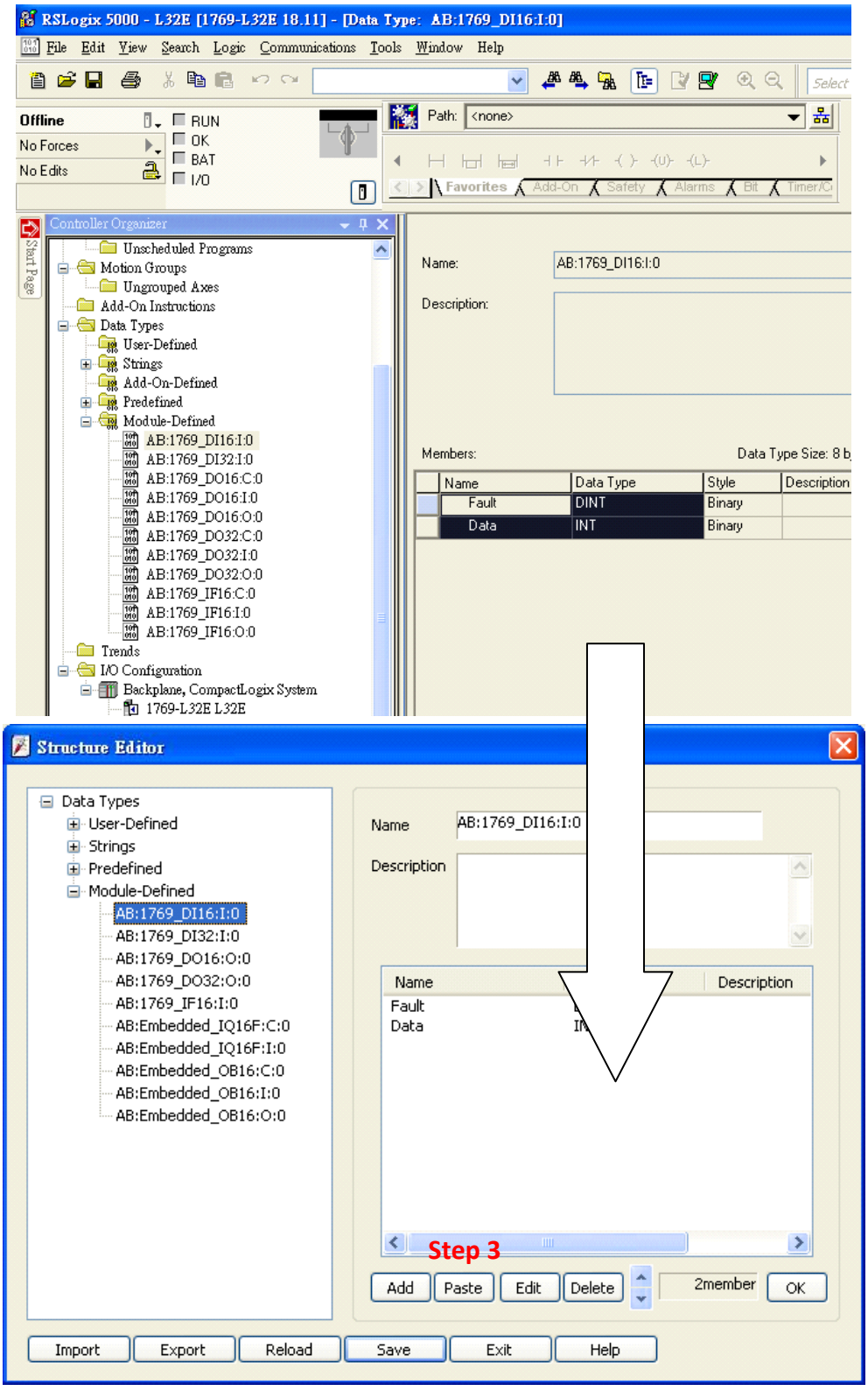
In Structure Editor.exe, right click on **[Module-Defined]**, and then click **[New Data Type].**



In **[New Data Type] » [Name]**, enter the Module-Defined name.

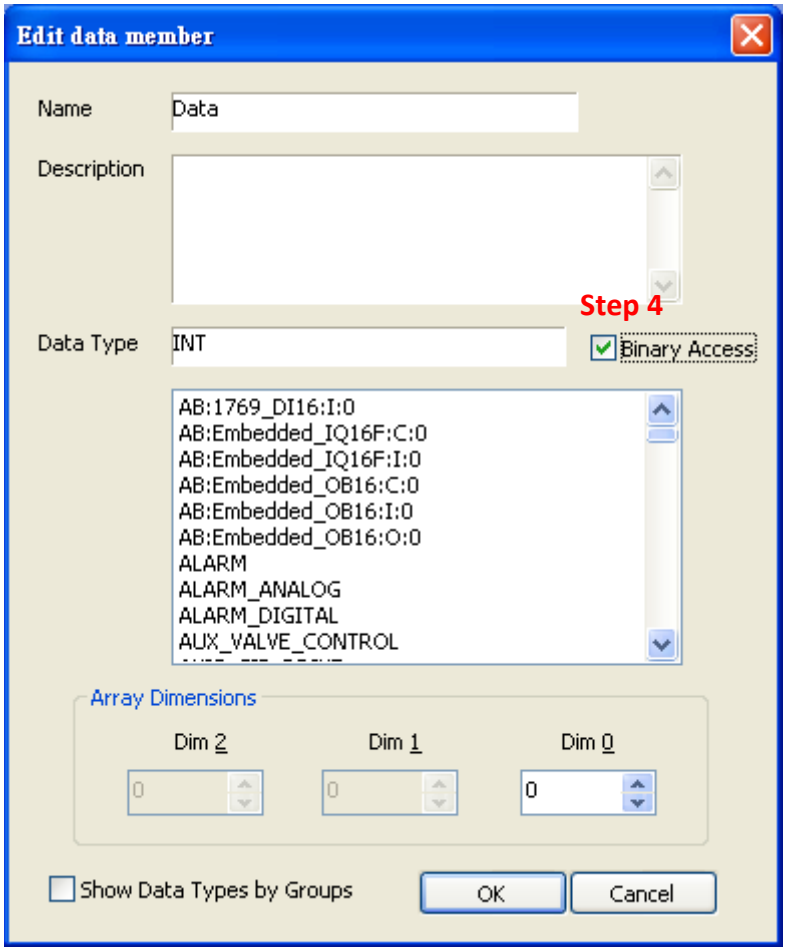
Step 3

Click **[Paste]**, in dialog box press Ctrl-V to paste Name and Data Type.



Step 4

Select data then click **[Edit]**, since the data of the modules can be operated by bit, **[Binary Access]** should be selected, then click **[OK]** to return to Structure Editor.

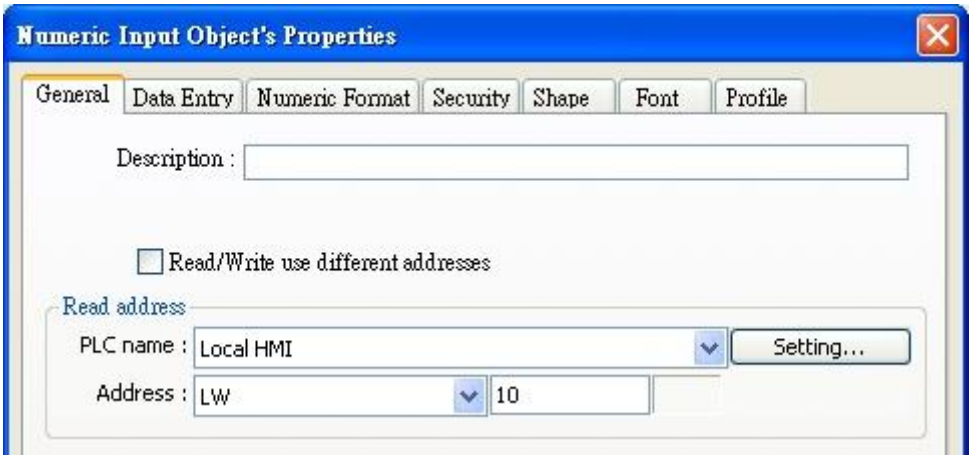


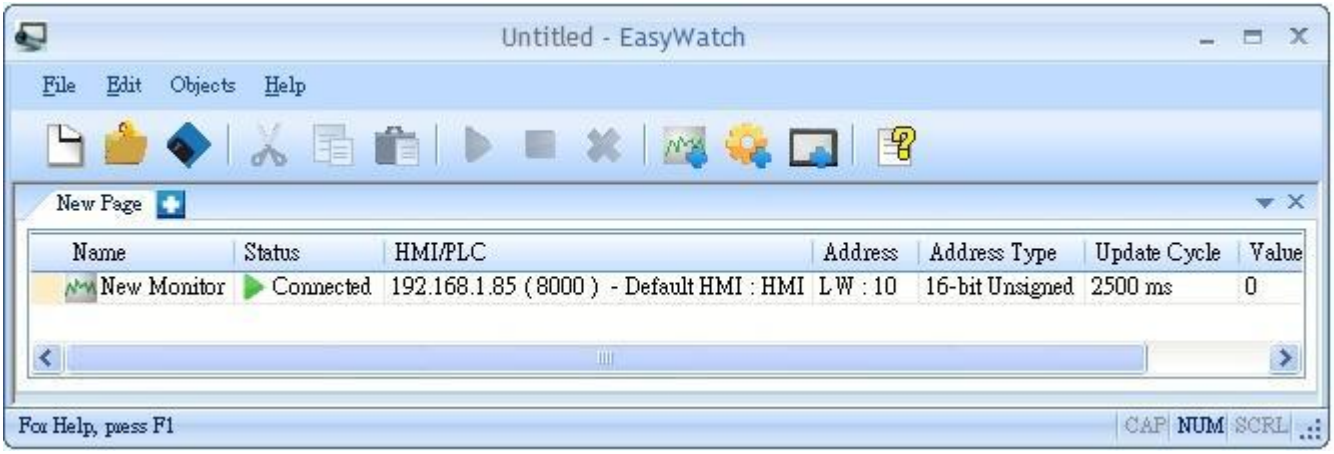
Click **[OK]** to finish setting.

Chapter 35 EasyWatch

35.1 Overview

EasyWatch allows users to monitor HMI or PLC address values via Ethernet on PC, or calling out Macro for easier debugging, remote monitoring, and controlling.
For example, In EasyBuilder create a Numeric Input Object, address set to LW-10, and set the same in EasyWatch. The value will be shown in EasyWatch when it is successfully connected..





When system register **[LB-9044 (disable remote control)]** or **[System Parameter Settings] » [System Setting] » [Prohibit remote HMI connecting to this machine]** is enabled, monitoring in EasyWatch is not available.

35.2 Basic Functions

35.2.1 Basic Functions

Item	Description
File	**New** Open a new EasyWatch file.
	Open Open an existing EasyWatch file.
	Save Save EasyWatch file.
	Save As Save EasyWatch file to ewt format.
	Exit Exit EasyWatch.
Edit	**Cut** Cut to relocate the selected items to the clipboard.
	Copy Copy the selected items to the clipboard.
	Paste Paste the items in the clipboard at the selected location.
Objects	**Add Object** Add new Monitor or Macro objects.
	Delete Objects Select the objects to be deleted, a dialog will be shown, click [Yes] to delete.
	Modify Object Change the settings of the selected object.
	HMI Manager Add, modify, or remove HMI settings.
	Run Execute the selected object.
	Stop Stop executing the selected object.
Help	**Help Topics** Reference of how to operate the basic functions.
	About EasyWatch EasyWatch version information.

35.2.2 Quick Selection Tools

 New: Open a new EasyWatch file.

 Open: Open an existing EasyWatch file.

 Save: Save EasyWatch file.

 Cut: Cut to relocate the selected items to the clipboard.

 Copy: Copy the selected items to the clipboard.

 Paste: Paste the items in the clipboard at the selected location.

 Run: Execute the selected object.

 Stop: Stop executing the selected object.

 Delete Objects: Delete the selected object.

 Monitor: Add a new Monitor object.

 Macro: Add a new Macro object.

 HMI Manager: Add, modify, or remove HMI settings.

 Help Topics: Reference of how to operate the basic functions.

35.3 Monitor Settings

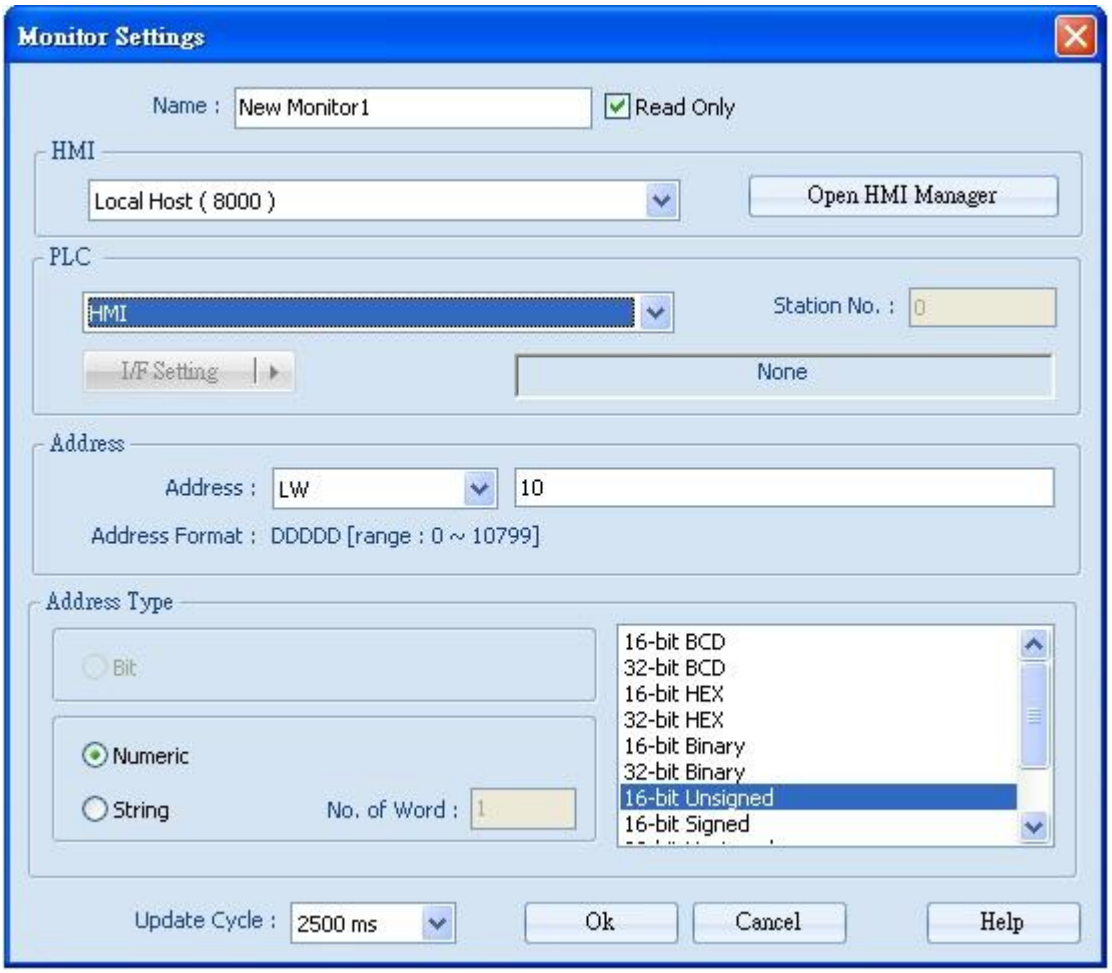
35.3.1 Add Monitor

There are two ways to create a Monitor Object：

1. Select from the toolbar: **[Objects] » [Add Object] » [Add Monitor]**.

2. Select from the quick selection tools: **[Add Monitor]**.

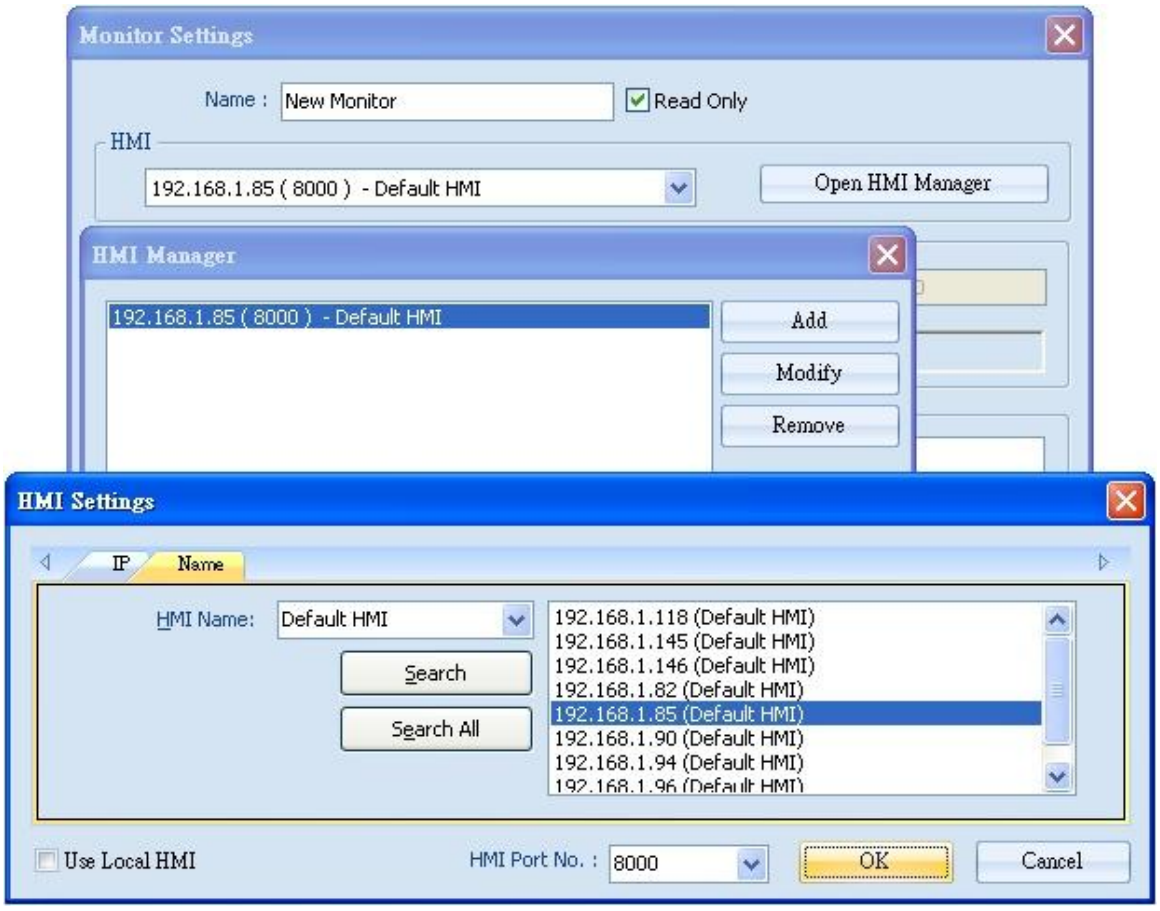
35.3.2 Monitor Settings



[Name]: Name the object and the name can't be repeated.
[Read Only]: If an object is set to read only, its address value can't be set.
[HMI]: Select the HMI to monitor.
[PLC]: Set the device, station number, and interface of the PLC to monitor.
[Address]: Set the address type and the address to monitor.
[Address Type]: When the address type is set, the available formats to display the address can be selected. When executing, the address will be calculated and displayed according to the selected format.
[Update Cycle]: Set the update interval of the monitor object. If many objects are executed simultaneously, error or delay may happen.

35.3.3 Add a New Device

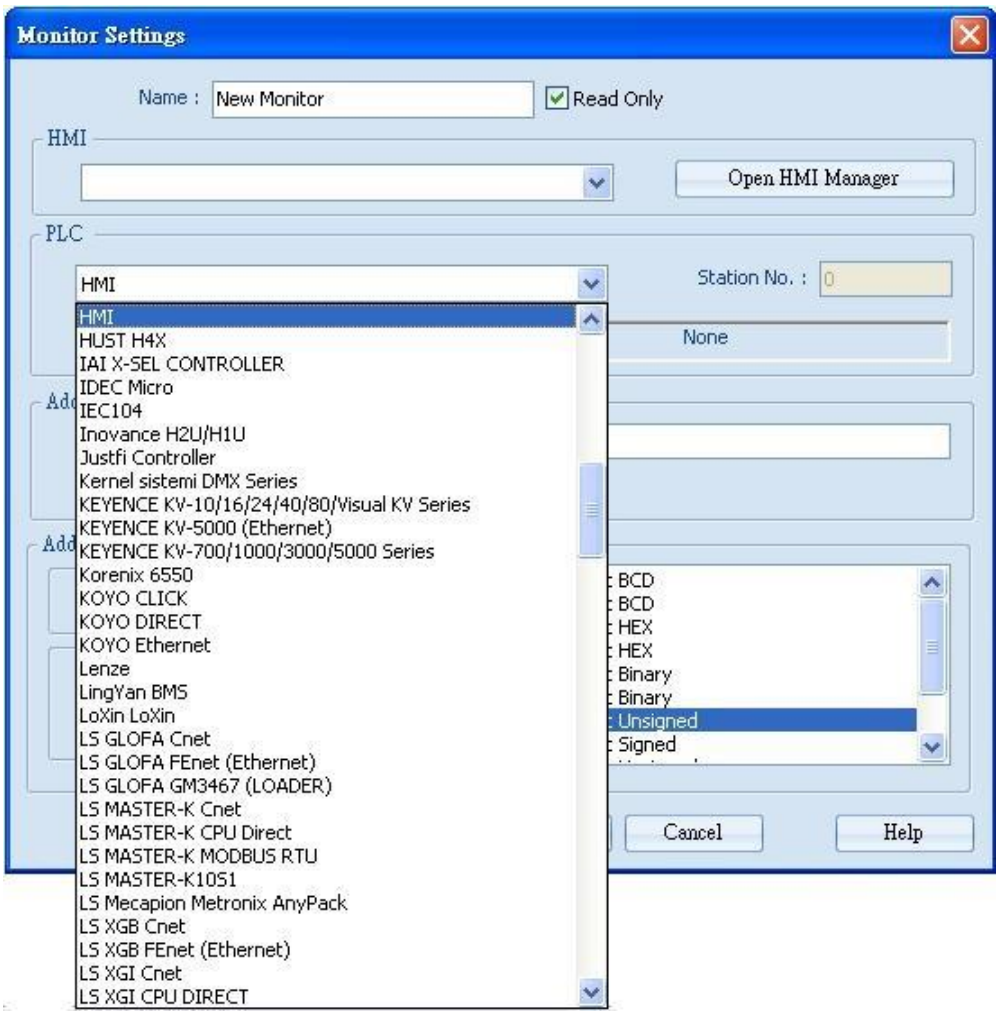
6. **Select HMI:** Select a target HMI. If the target HMI does not exist, follow the steps to add a new device:



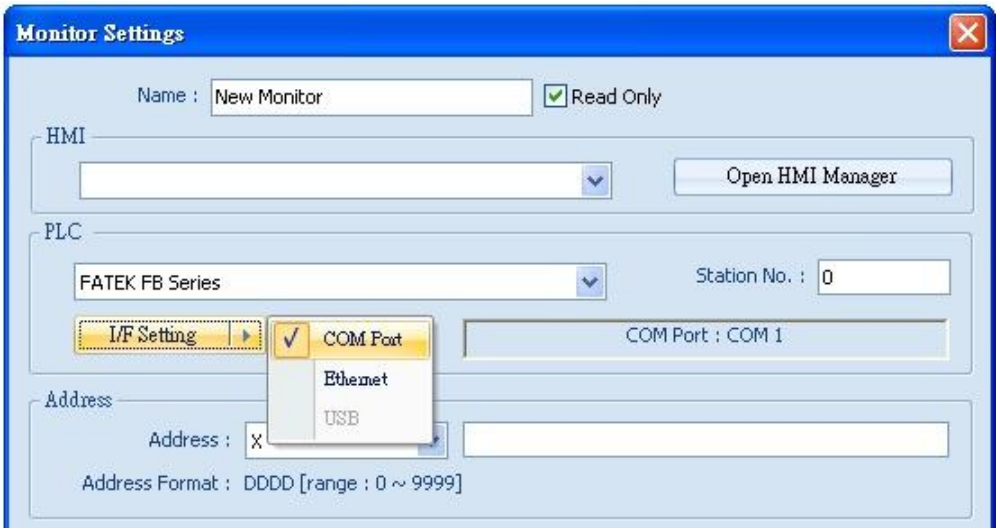
Or, select [Use Local HMI] check box to use the project simulated on PC to be the monitor device.

7. Select PLC: Select a target PLC.

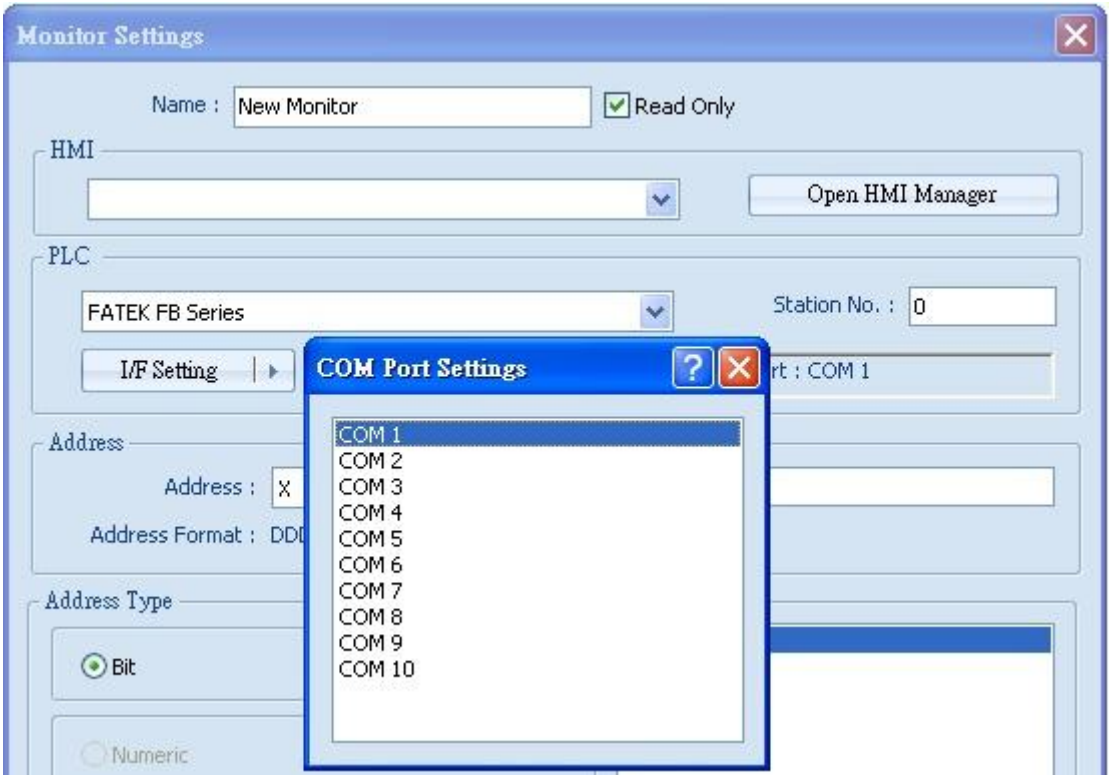
If HMI is selected, directly control the local HMI.



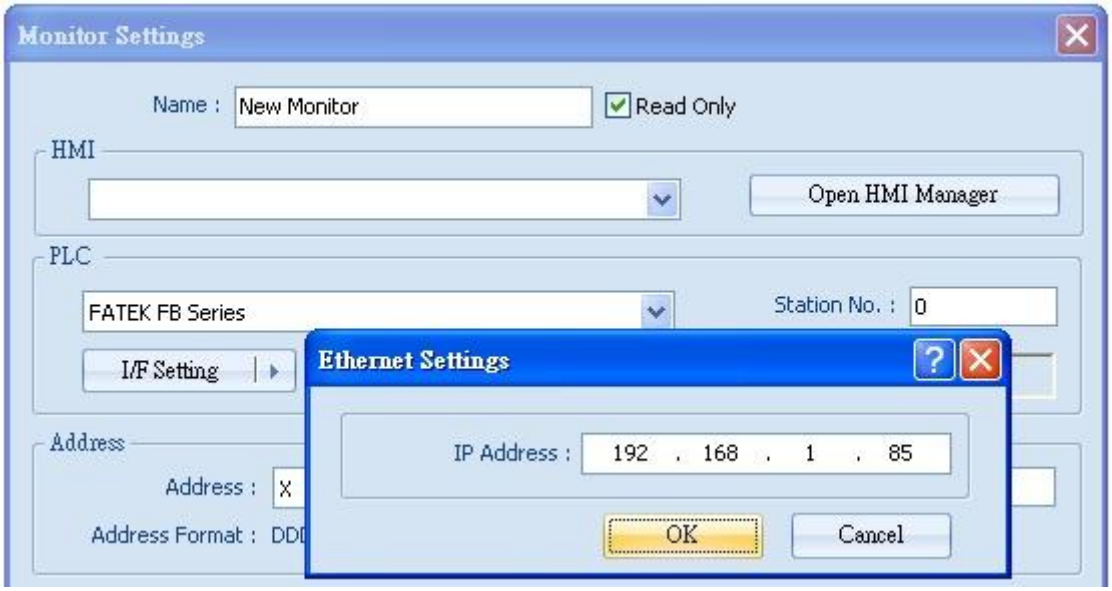
Please select **[I/F Setting]** to **[COM Port]** or **[Ethernet]** of the PLC.



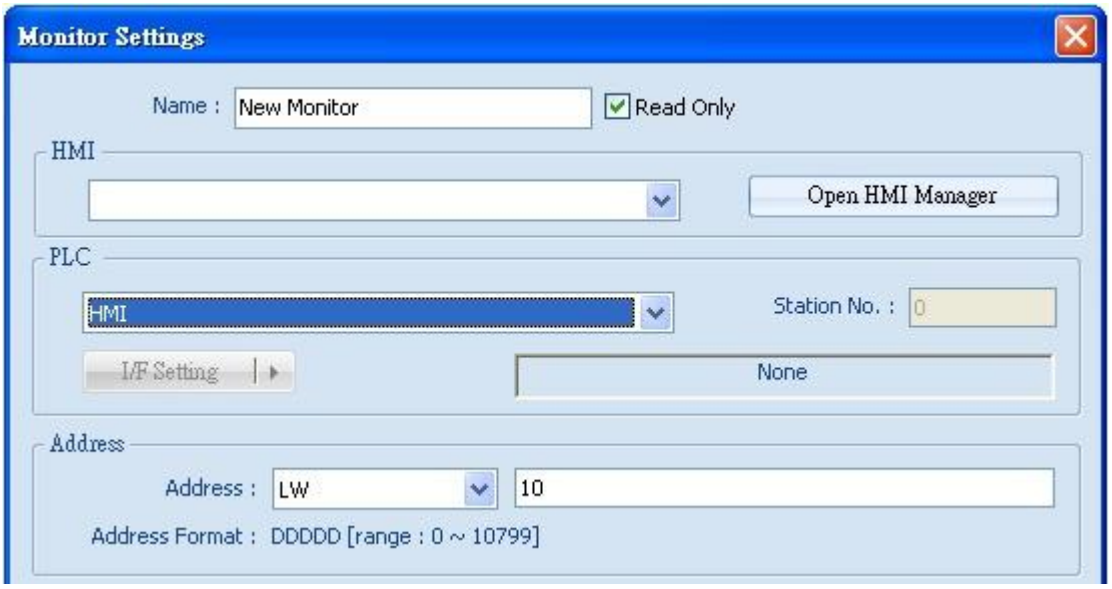
Select COM Port: Click [I/F Setting] to select a COM port.



Select Ethernet: Click [I/F Setting] to set IP Address.

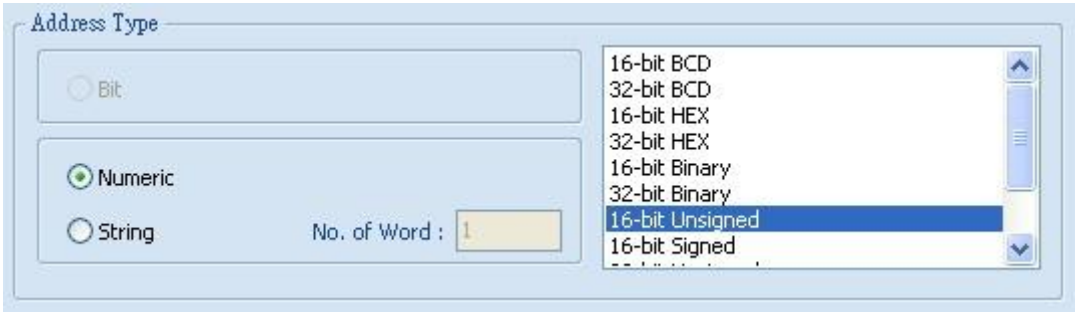


8. Set Address: Set the address type and the address to be monitored.

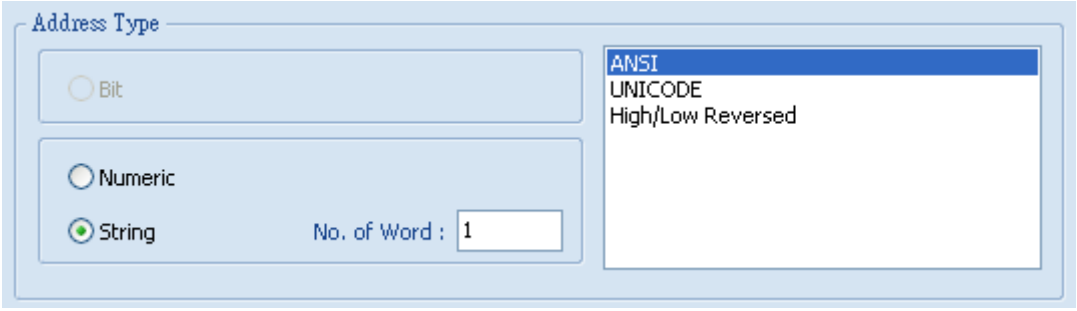


9. Set Address Type: When Word type is selected, set address type to [Numeric] or [String].

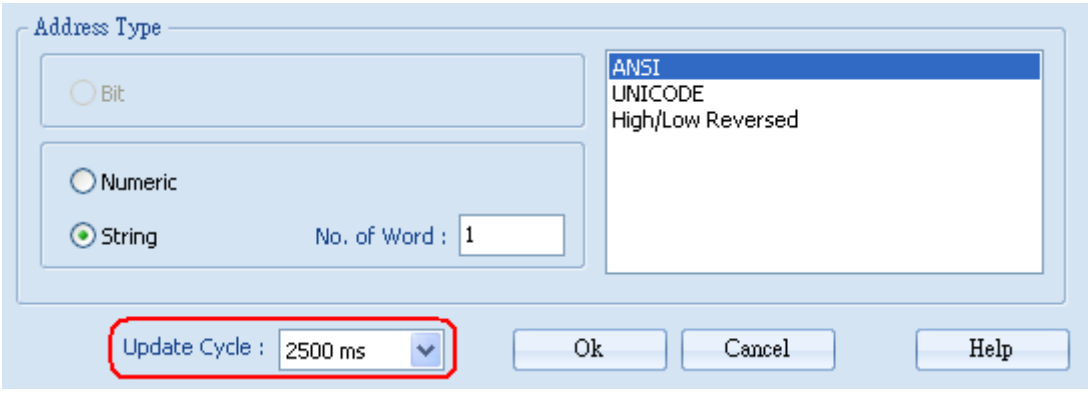
[Numeric]: Select the data format of the monitor address.



[String]: Select data format from [ANSI], [UNICODE], and [High/Reversed]. Set [No. of Word] to read.



10. Set Update Cycle: Set the update interval of the monitor object.
The range can be set from 500ms to 5000ms.

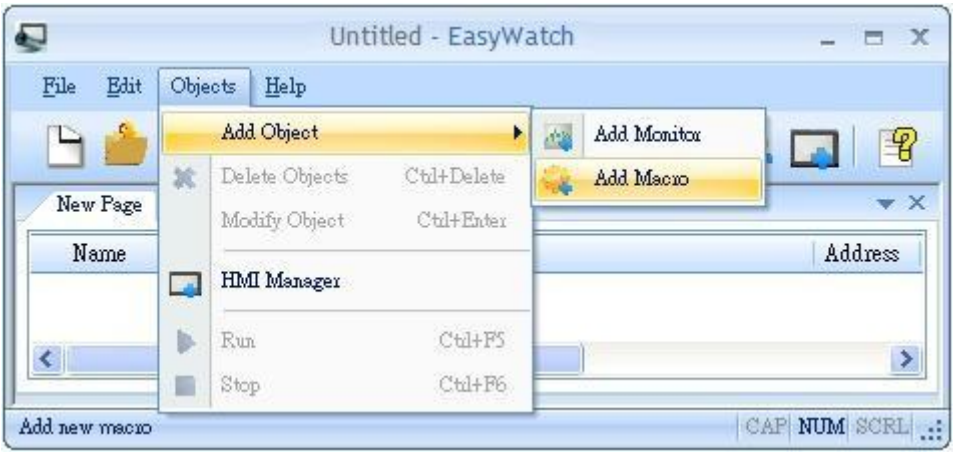


35.4 Macro Settings

35.4.1 Add Macro

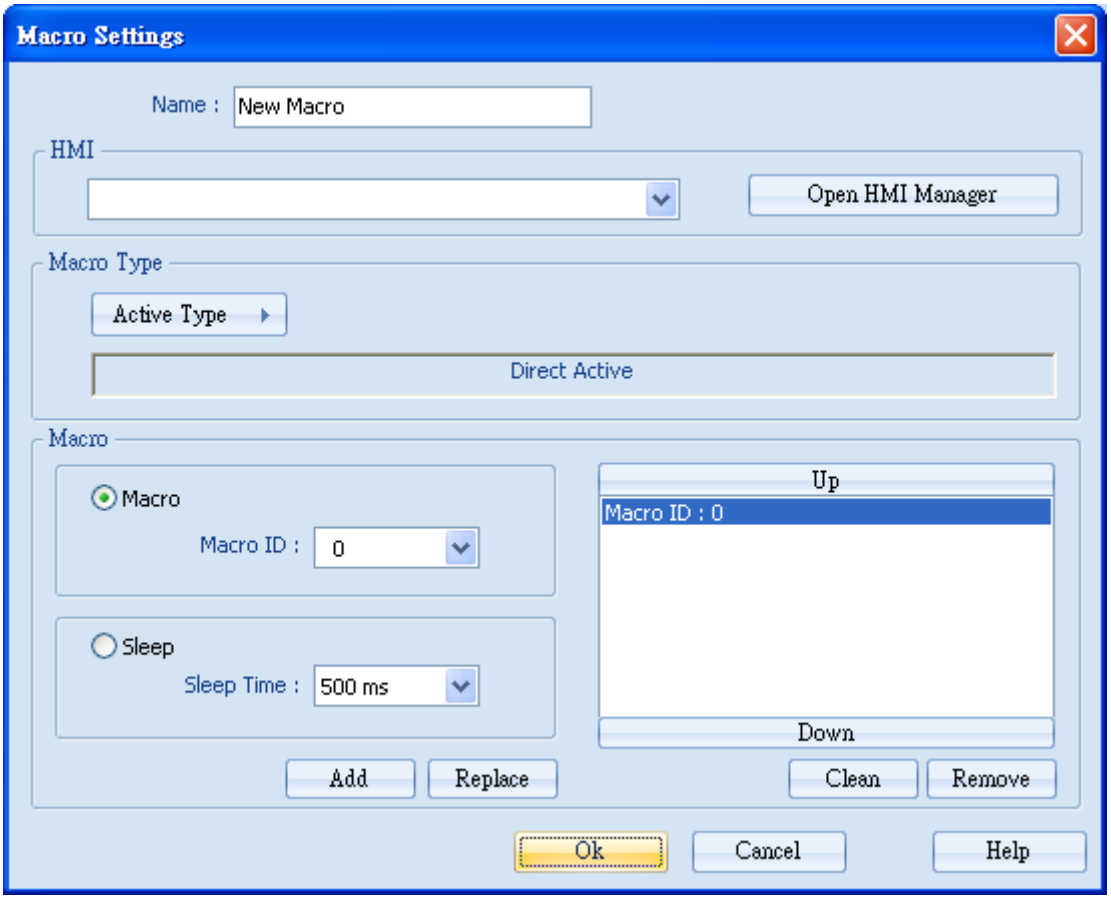
There are two ways to create a Macro object：

1. Select from the toolbar: **[Objects] » [Add Object] » [Add Macro]**.



2. Select from the quick selection tools: **[Add Macro]**.

35.4.2 Macro Settings



[Name]: Name the object and the name can't be repeated.

[HMI]: Select a HMI to monitor.

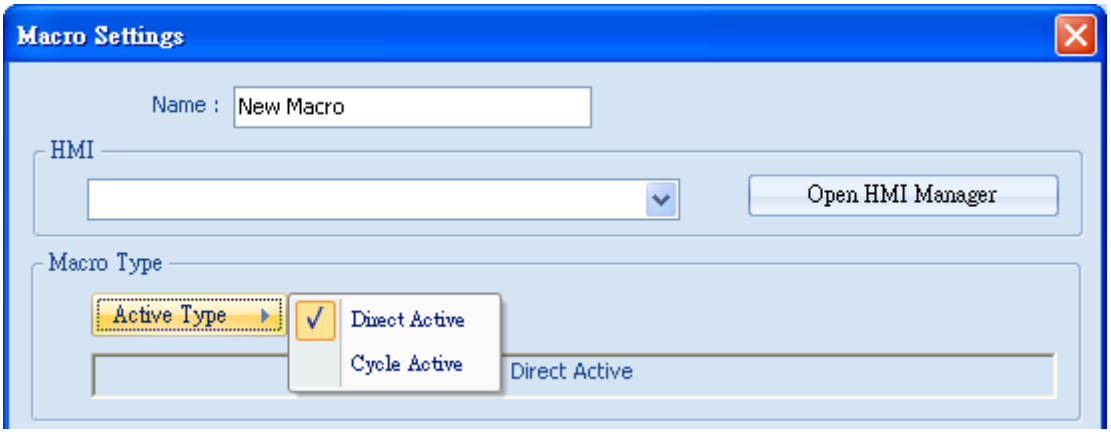
[Macro Type]: The ways to execute Macro include Direct Active or Cycle Active.

[Macro]: Each Macro Object can execute multiple macros. The time interval between the execution of two macros can be set.

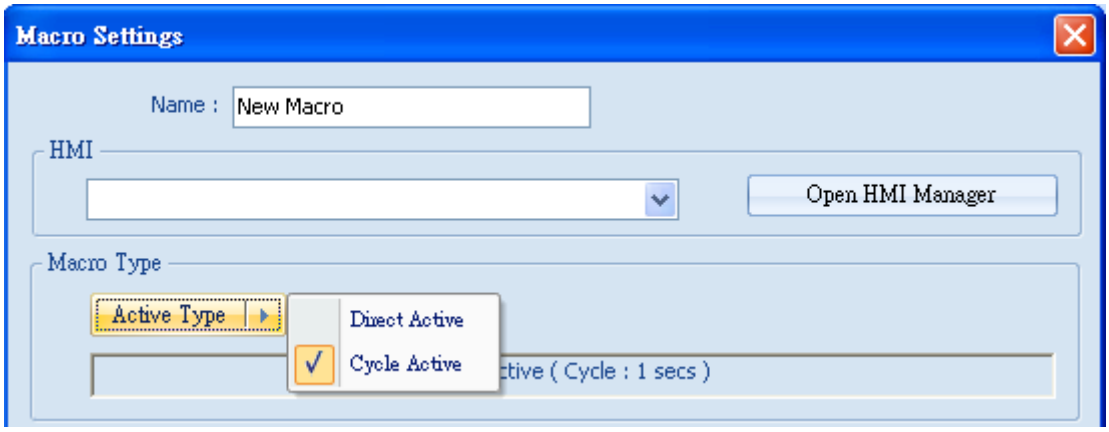
35.4.3 Add New Macro Settings

1. **HMI**: Please refer to “35.3.3 Add a New Device”.
2. **Macro Type**: Set to [Direct Active] or [Cycle Active].

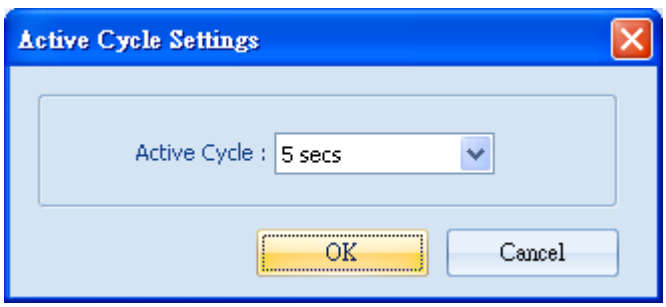
[Direct Active]: Directly execute Macro once.



[Cycle Active]: Set the interval of executing Macros.



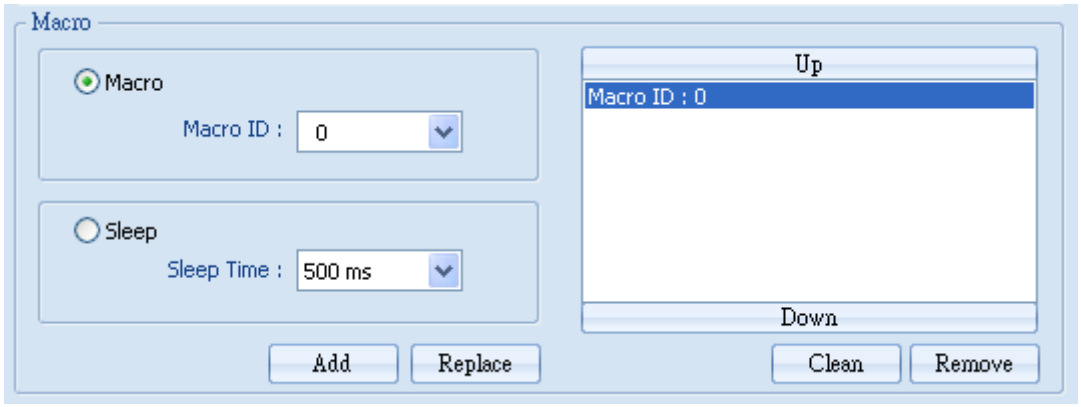
For example, if [Cycle Active] is set to 5 seconds, when start executing, the next time to execute the macro object will be 5 seconds later.



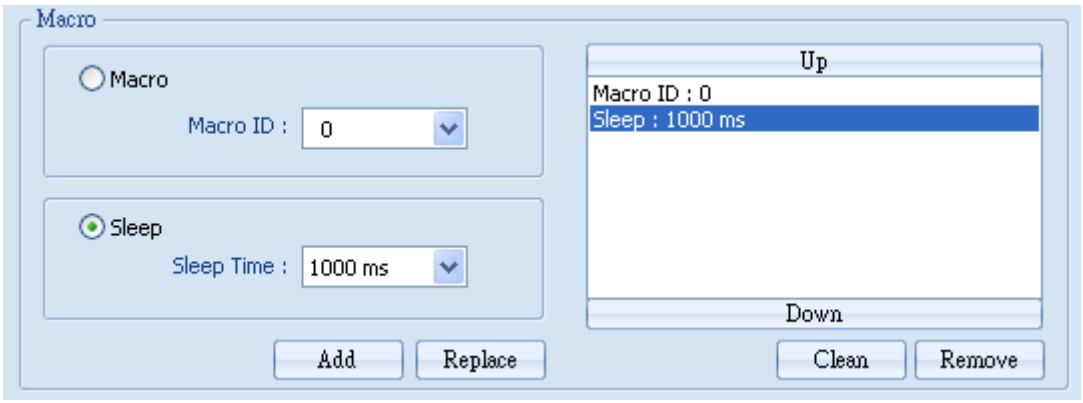
3. Macro

The setting includes **[Macro]** and **[Sleep]**.

[Macro]: Set the Macro ID to execute, and click [Add] to add the Macro to the list.



[Sleep]: Set the time interval between the execution of two Macros. Click [Add] or [Replace] to add or replace the Macros listed here.

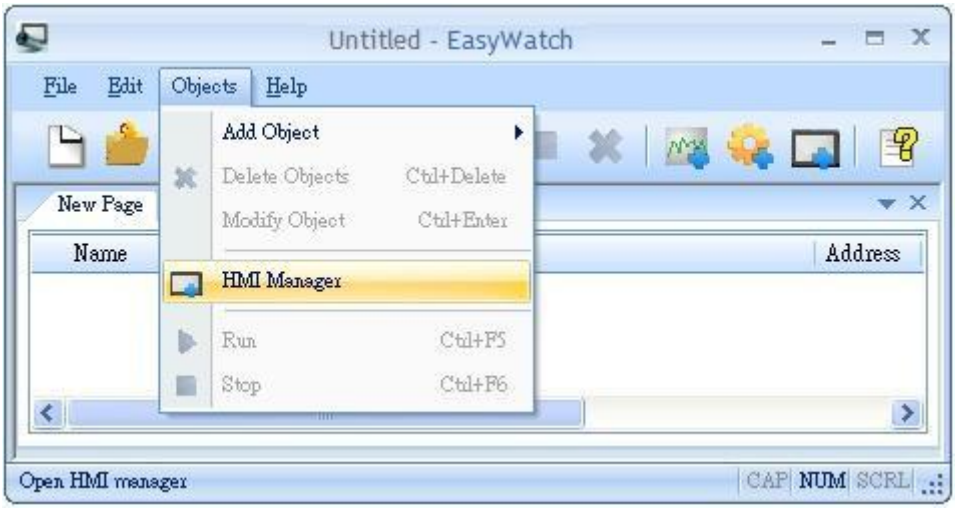


35.5 HMI Manager

35.5.1 HMI Settings

There are two ways to open HMI Settings：

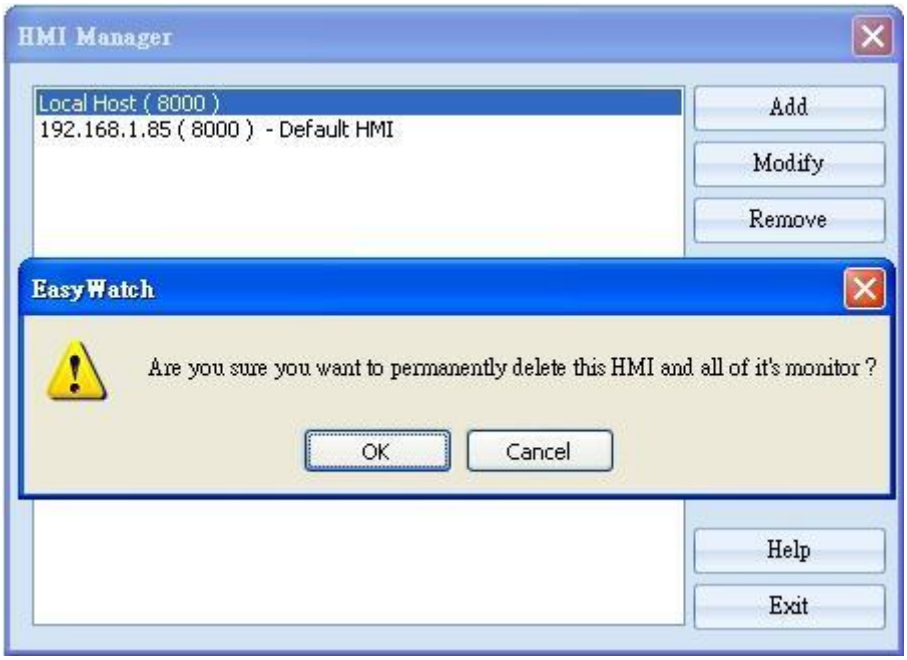
1. Select from the toolbar: **[Objects] » [HMI Manager]**.



2. Select from the quick selection tools: **[HMI Manager]**.

35.5.2 HMI Manager

EasyWatch allows monitoring addresses of multiple HMIs for easier management.
[Add]: Please refer to “35.3.3 Add a New Device”.
[Modify]: Select a HMI to modify the settings.
[Remove]: Remove HMI settings and click [OK].



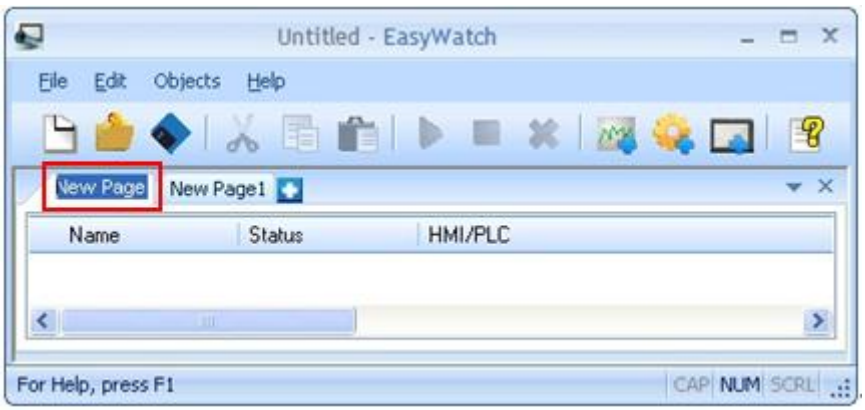
35.6 Object List

35.6.1 Page Settings

Add a new page: Click on “+” icon to add a new page.

Delete a page: Click on “X” icon and confirm the deletion.

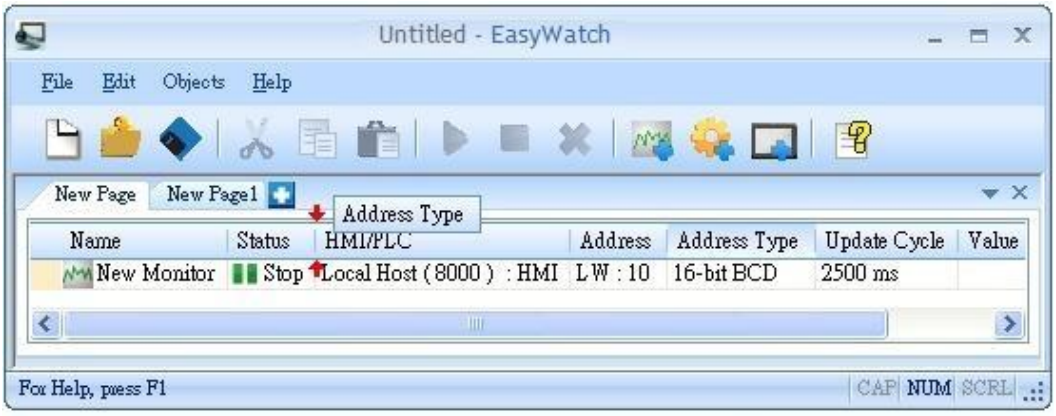
Rename the page: Double click on the page name and type in the new name.



35.6.2 Columns of Object List

[Name]: Display object names, the small icons beside the names are for users to identify the type of the objects.
[Status]: Display the status of the objects: [Connecting], [Connected], or [Stop]. If HMI is not connected or Port No. is incorrect, error message “HMI Not Found” will be shown. For Monitor Objects, if the address is incorrect, “Address Error” message will be shown.
[HMI/PLC]: Display the information of HMI / PLC that is currently operated by the objects.
[Address] / [Address Type]: For Monitor Objects, the relevant address settings will be displayed.
[Update Cycle]: Set the update interval of the monitor object.
[Value]: For Monitor Object, if the status shows [Connected], current HMI address value will be displayed. Modifying the value is also available when Read-Only checkbox is not ticked. For Macro Object, if set to [Direct Active], there will be an [Active] button in this column for clicking and directly execute Macro.

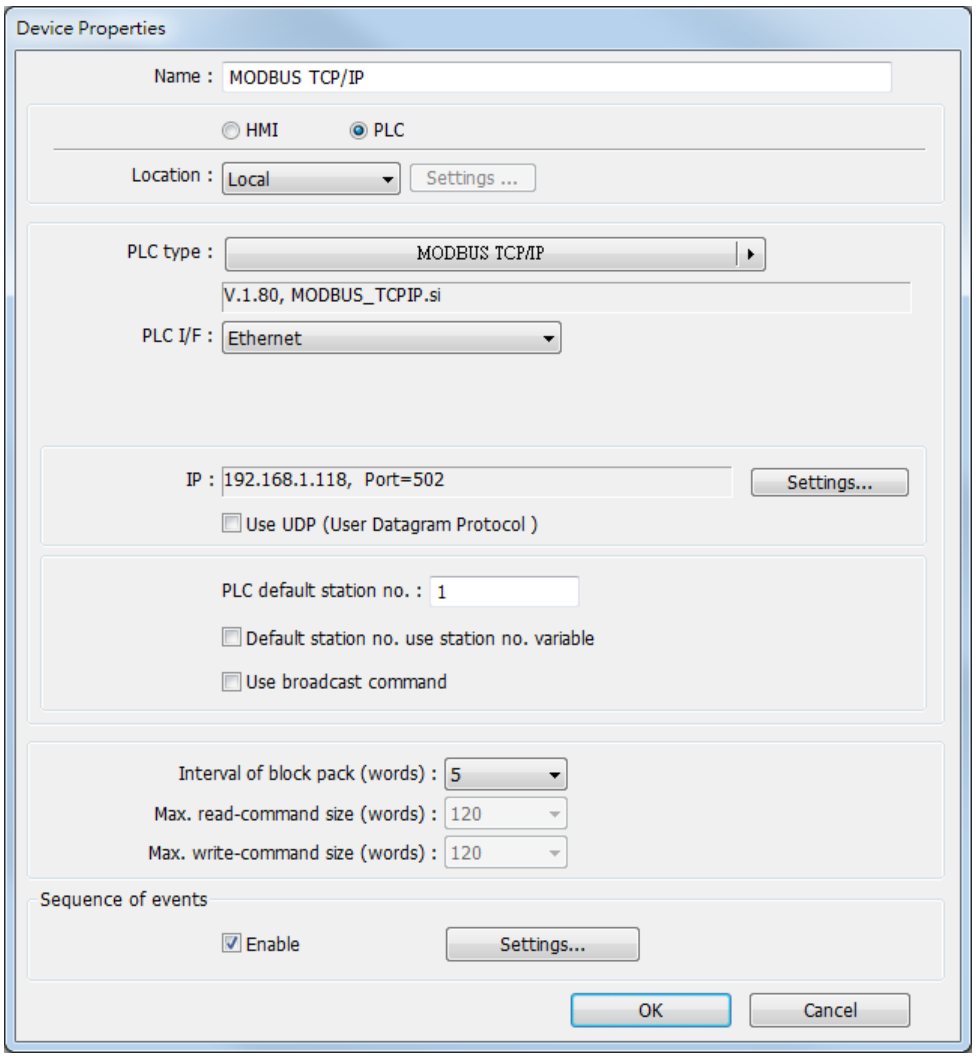
Drag and drop the column headers to the desired location.



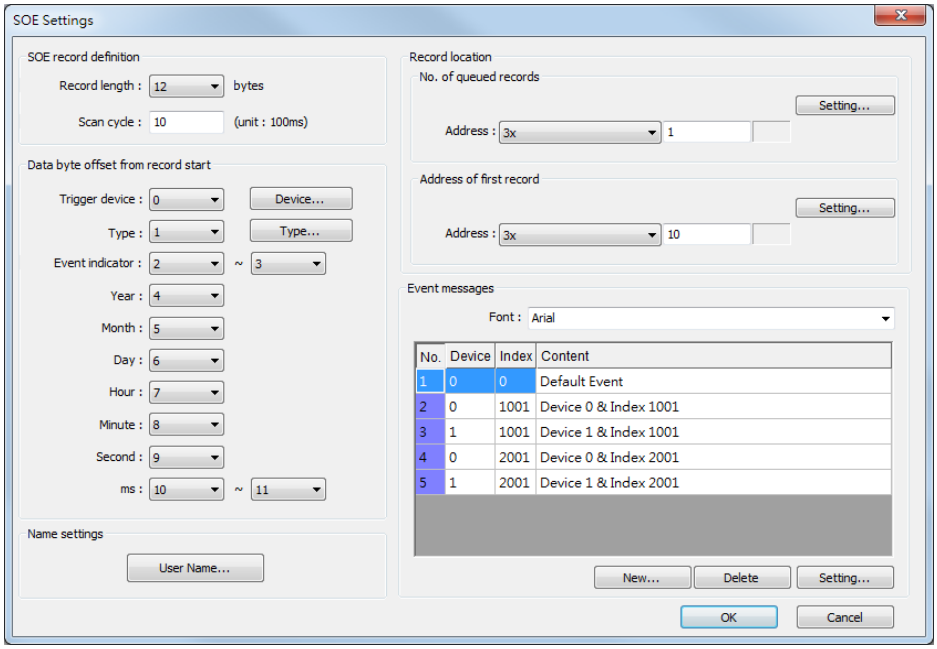
Chapter 36 Sequence of Events

36.1 Introduction

SOE, the abbreviation for Sequence Of Events, is a function that records the precise time of the events occur, and sorts the events by their time sequence. When an event occurs, PLC will store the data frame in [Address of first record], and writes the number of events in [No. of queued records]. HMI will scan [No. of queued records] in the frequency set in [Scan cycle]. If the value in [No. of queued records] is not 0, HMI reads the data frame in [Address of first record]. The content of data frame records the precise time and type of the event, etc. In EasyBuilder [System Parameter Settings] > [Device list] > [Settings] > [Device Properties], select [enable] check box under [Sequence of events]. Click [Settings] to specify the data format. SOE Display object can then display the event sequence for easier observation.



36.2 SOE Settings



Setting	Description
SOE record definition	
Record length	Sets the data length of SOE. The range is 10 to 128 bytes.
Scan cycle	Sets the frequency for the system polling the SOE data, the unit is 100 milliseconds, the range is 10 to 32767.
Data byte offset from record start	
The SOE data format of each brand of PLC is different. User should manually specify the format, the start offset and end offset of each data field, for the system to analyze data frame correctly. The format is defined as the following.	
Trigger device	Designates the name of device type and its index number. The range is 0 to 255.
Type	Designates the name of event type and its index number. The range is 0 to 255.
Event Indicator	The index value of the event.
Year	The date of the event.
Month	
Day	
Hour	The time of the event.
Minute	
Second	

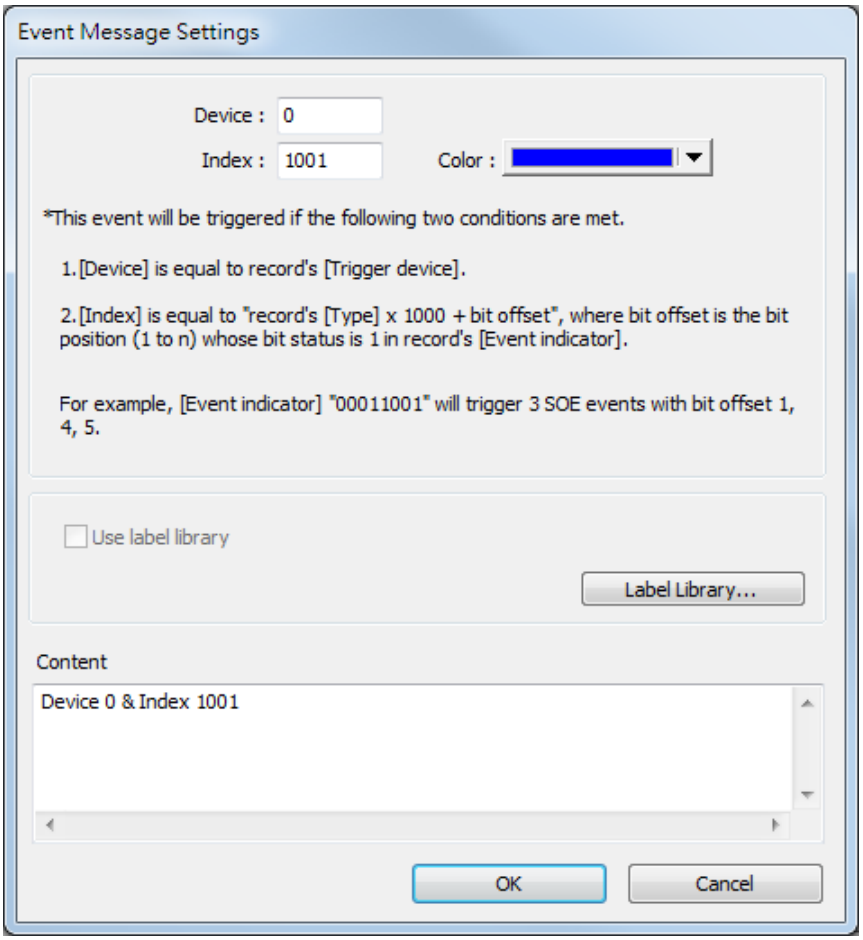
ms	The millisecond of the time of the event.
Name settings	
User Name	Designates the user name for login. The range is 1 to 12.
Record location	
No. of queued records	The number of SOE occur will be written in this register. For example, if set [Address of first record] to LW-0, data length 12 bytes (6 words), when there are two SOE occur, then LW-0 to LW-5 will be the first data while LW-6 to LW11 will be the second data, and so on. After the value is input, it returns to zero in the system.
Address of first record	Sets the start address to read SOE data.
Event Messages	
Font	Sets the font for displaying events.
Event message list	Sets the index number of the device and event and the corresponding message content to be displayed. When the system receives a SOE data frame, there are two conditions must be satisfied when comparing the following values in order to trigger the corresponding message. 1. The value in [Trigger device] matches the value in [Device] column. 2. The value gained by calculating **[Type] × 1000 + [Bit offset of the event indicator]** matches the value in [Index] column. If either of the [Device] or [Index] columns is not defined, the event message with index number 0 is displayed. For example, if the value of [Type] is 1, and the value of [Event indicator] is 5 (Binary 101), the event messages with index number 1001 and 1003 will be triggered. 1 × 1000 + Bit 1 => Event Index 1001 1 × 1000 + Bit 3 => Event Index 1003

The way of setting SOE is different comparing EasyBuilder8000 versions earlier than V4.65.04 with versions later than V4.65.05. The major difference is the way to calculate the event message index value, as compared in the following table.

Versions before and includes EB8000 V4.65.04	**[Type] × 16 + [Bit offset of the event index]** For example, if the value of [Type] is 1, and the value of [Event index] is 5 (Binary 101), the event messages with index number 17 and 19 will be triggered. 1 × 16 + Bit 1 => Event Index 17 1 × 16 + Bit 3 => Event Index 19
Versions after and includes EB8000 V4.65.05	**[Type] × 1000 + [Bit offset of the event indicator]** For example, if the value of [Type] is 1, and the value of [Event indicator] is 5 (Binary 101), the event messages with index number 1001 and 1003 will be triggered. 1 × 1000 + Bit 1 => Event Index 1001 1 × 1000 + Bit 3 => Event Index 1003

For Event Message Settings, please see the following page.

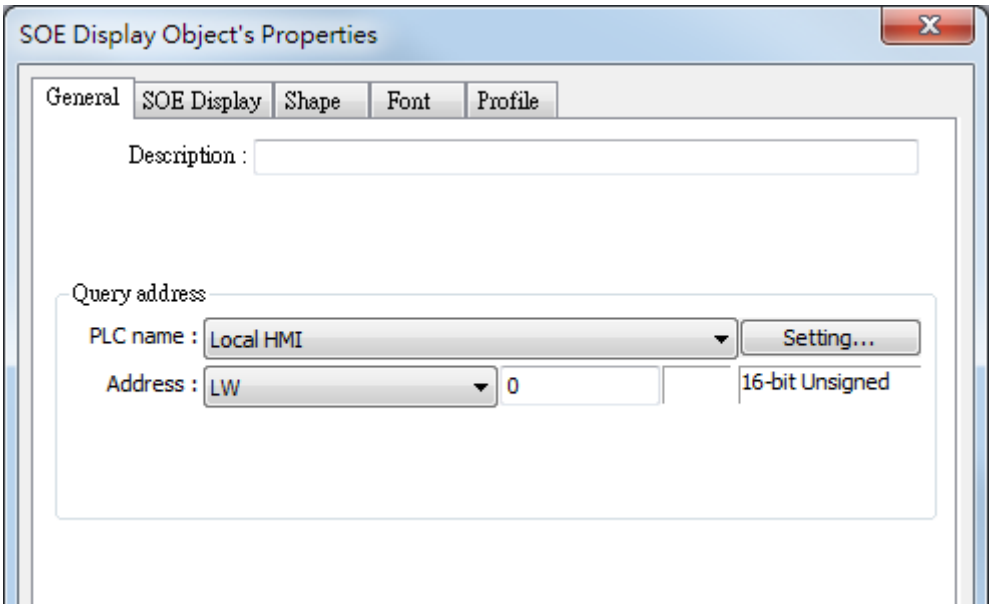
Event Message Settings



Setting	Description
Device	Gives the device type a specific index value, range: 0 to 255.
Index	Gives the event message a specific index value
Color	Sets the font color of the displayed event.
Use label library	If selected, the content is from Label Library.
Label Library	Opens Label Library to build, modify, or delete label tag.
Content	Enters the content to be displayed.

36.3 SOE Display

SOE Display object allows users to view SOE event messages. The setting of SOE Display is available after enabling [Sequence of events] in [Device list].
The General tab of SOE Display object is shown in the following figure.



[Query address] uses 19 word registers to select the events to be displayed.
(n is any number)

Name	Query Address	Description
Mode	n	Please see the table next page.
Status	n + 1	Please see the table next page.
Start date	n + 2 ~ 4	Year, Month, Day
Start time	n + 5 ~ 7	Hour, Minute, Second
Start time ms	n + 8	
Type	n + 9	Range: 0 to 9
Device	n + 10	Range: 0 to 99
User name	n + 11	Range: 1 to 12
End date	n + 12 ~ 14	Year, Month, Day
End time	n + 15 ~ 17	Hour, Minute, Second
End time ms	n + 18	

The value in [Mode] indicates:

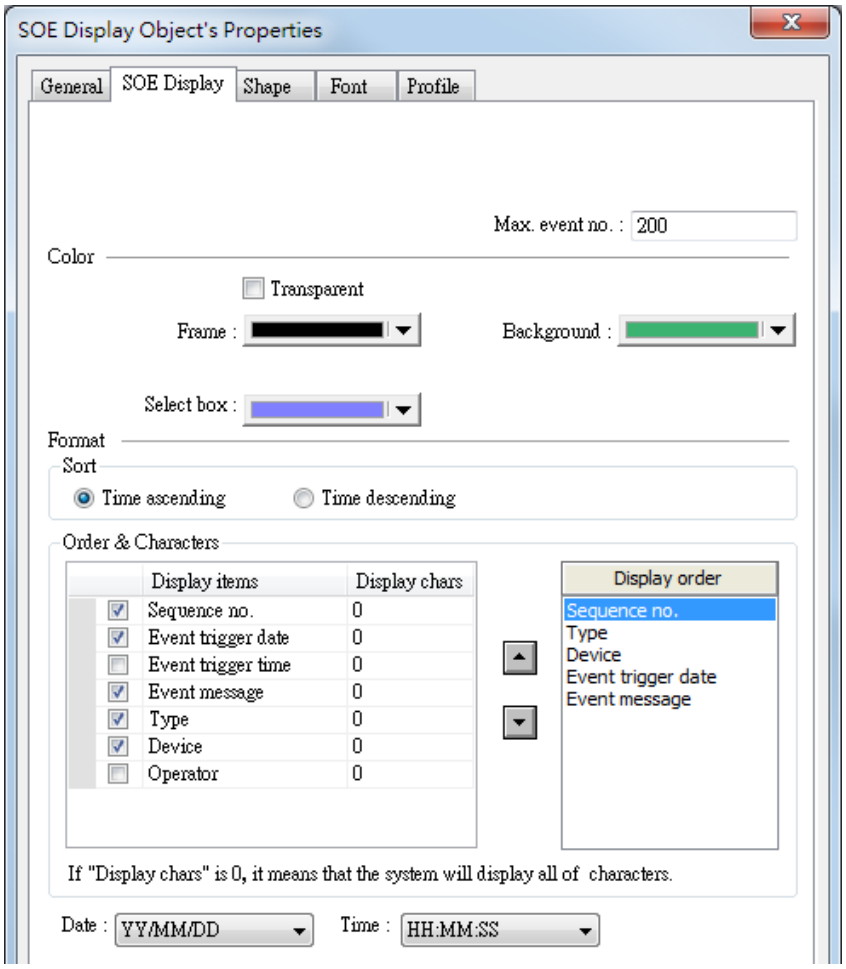
Value	Mode
1	Query [Start date] (values in query address n + 2 to n + 4 are used). When [Status] is 4, the value in [End date] (Query address n + 12 to n + 14) is needed to form a query range.
2	Query [Start time] (values in query address n + 5 to n + 7 are used). When [Status] is 4, the value in [End time] (Query address n + 15 to n + 17) is needed to form a query range.
4	Query [Start time ms] (value in query address n + 8 is used). When [Status] is 4, the value in [End time ms] (Query address n + 18) is needed to form a query range.
8	Query [Type] (value in query address n + 9 is used).
16	Query [Device] (value in query address n + 10 is used).
32	Query [User name] (value in query address n + 11 is used).

The value in [Status] indicates:

Value	Mode
0	Displays all events.
1	Selects the events where the value of the column assigned in Mode **equals to** the value read from a register.
2	Selects the events where the value of the column assigned in Mode is **greater than or equals to** the value read from a register.
3	Selects the events where the value of the column assigned in Mode is **less than or equals to** the value read from a register.
4	Selects the events where the value of the column assigned in Mode falls in the specified **range**.

- To query [Type], [Device], or [User name], the acceptable status is 0 or 1. That is, if the status is 0, all the events are displayed. Otherwise it selects the events only if the value of the column assigned in mode equals to the value in the query address of [Type], [Device], or [User name].
- Set both [Mode] and [Status] to start the query.

The SOE Display tab is shown in the following figure.



Setting	Description
Max. event no.	Enter the number of events to display. When the number of triggered events is greater than this number, the old events are overwritten by the new events.
Transparent	If selected, the colors of the object frame and background are not displayed.
Frame	Selects the color of the object frame.
Background	Selects the color of the object background.
Select box	Selects the color of the select box shown when an event is chosen.
Time ascending	The latest events are placed at the bottom.
Time descending	The latest events are placed at the top.
Display order	Selects the items to display when an event occurs. If [Display chars] is set to 0, displays the full content of the item.
Date	Selects the format of date.
Time	Selects the format of time.

Chapter 37 MODBUS TCP/IP Gateway

37.1 Overview

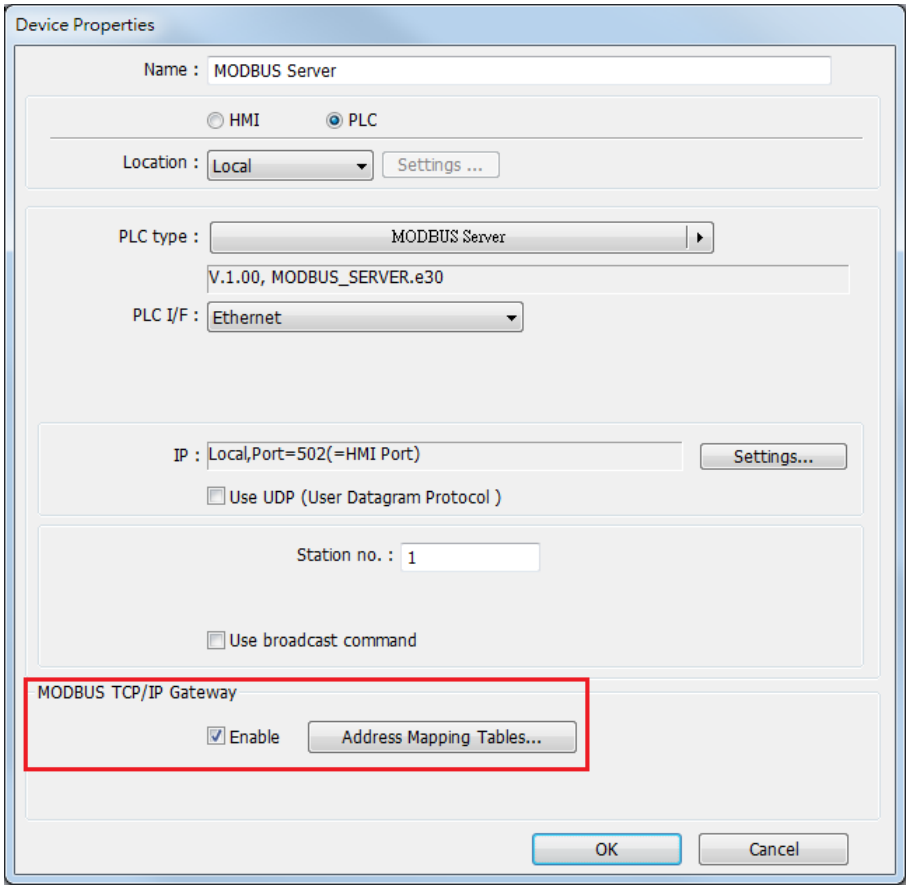
To access the data of the PLC connected to HMI with SCADA software (Supervisory Control and Data Acquisition), the former way was to transfer PLC data to the HMI's local address first, and then use MODBUS TCP/IP protocol on PC to read HMI local address to get PLC data. Now by using MODBUS TCP/IP Gateway provided by EasyBuilder, the mapping of MODBUS address to PLC address can be defined first, and then one can directly use MODBUS TCP/IP protocol to access PLC data.

37.2 Configuration

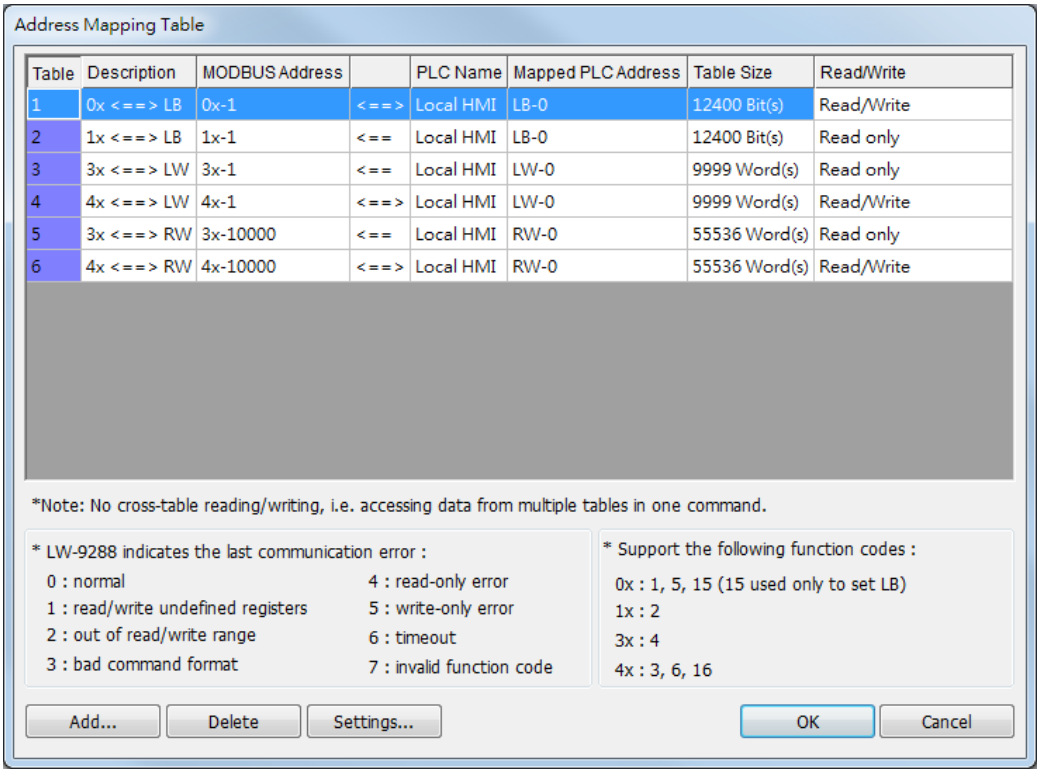
37.2.1 Steps to Create an Address Mapping Table

To create an Address Mapping Table, please follow the steps:

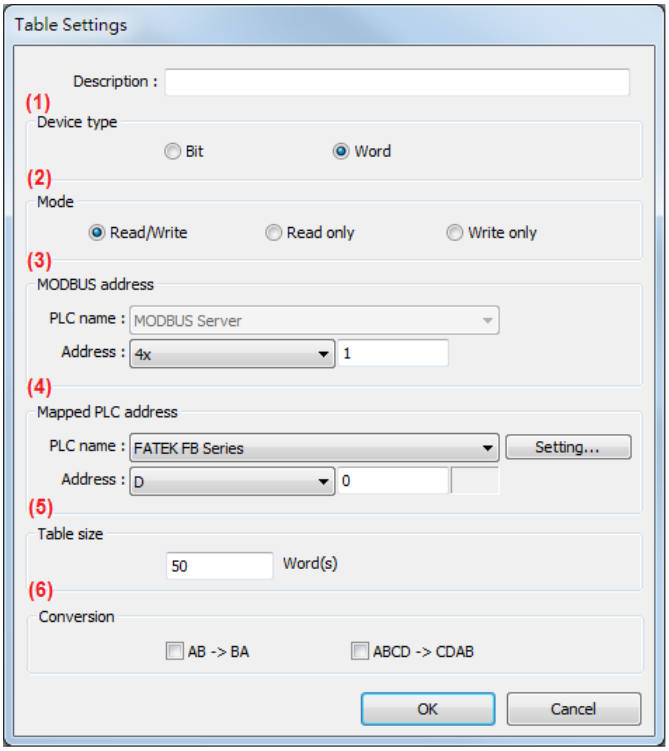
1. In [System Parameter Settings] » [Device] tab, add the PLC device. (In the example FATEK FB Series is used).
2. Add MODBUS Server (Ethernet), select [Enable] check box under [MODBUS TCP/IP Gateway] as shown in the following figure.



3. Click [Address Mapping Tables] button and the following default tables will be displayed. Modify the tables if needed or add new tables.



4. For example, to access the data in the 50 consecutive registers of FATEK FB Series PLC starting from register D-0, configure the settings as shown in the following figure.



(1) Select the device type of the registers to be mapped, in the example select [Word].
(2) Select the mode to access the data in the mapped register, in the example set to [Read/Write].
(3) Set the start address of MODBUS, in the example set to “4x-1”.
(4) Set the start address of the mapped PLC, in the example set to “D-0”.
(5) Set the range size of address mapping, in the example set to “50”.
(6) If needed, select high/low byte swap (AB->BA) or high/low word swap (ABCD->CDAB).

Table	Description	MODBUS Address		PLC Name	Mapped PLC Address	Table Size	Read/Write
1	Access D0 ~ D49	4x-1	<==>	FATEK FB Series	D-0	50 Word(s)	Read/Write

The above figure shows that MODBUS Server 4x-1 ~ 4x-50 registers are mapped to FATEK FB Series PLC D-0 ~ D-49 registers.

5. When finished, the data of FATEK FB Series PLC D-0 ~ D-49 registers are now accessible by using MODBUS TCP/IP protocol to send read / write command to 4x-1 ~ 4x-50 registers.

37.2.2 Notes about Configuring Address Mapping

- UDP is not supported when using the MODBUS TCP/IP Gateway function.
- This function is only supported by MODBUS Server (Ethernet) interface.
- System register LW-9288 is used to indicate if data transfer has been correctly executed.

The following error codes represent:

Value	Definition
0	Normal
1	Read or write the register that is not defined in the Address Mapping Table.
2	Read or write a range of registers that is not within the range defined in a single Address Mapping Table. (Or, read / write a register that is defined in other Address Mapping Table.)
3	The command format does not follow MODBUS TCP/IP protocol.
4	Modify a read-only register.
5	Read a write-only register.
6	Cannot get the correct reply from PLC within the specified time range.
7	Use a function code that is not supported by MODBUS Server.

- The defined register range must not overlap between different mapping tables.
- If [MODBUS TCP/IP Gateway] is enabled, EasyBuilder will cancel the original mapping between MODBUS Server and HMI register. That includes:
  (1) 0x, 1x mapped to LB
  (2) 3x, 4x mapped to LW, RW

Therefore, to access data in LB or LW register via 0x, 1x, 3x, 4x, configure the Address Mapping Table again. The following figure is an example.

Table	Description	MODBUS Address		PLC Name	Mapped PLC Address	Table Size	Read/Write
1	0x <==> LB	0x-1	<==>	Local HMI	LB-0	12400 Bit(s)	Read/Write
2	1x <==> LB	1x-1	<==	Local HMI	LB-0	12400 Bit(s)	Read only
3	3x <==> LW	3x-1	<==	Local HMI	LW-0	9999 Word(s)	Read only
4	4x <==> LW	4x-1	<==>	Local HMI	LW-0	9999 Word(s)	Read/Write
5	3x <==> RW	3x-10000	<==	Local HMI	RW-0	55536 Word(s)	Read only
6	4x <==> RW	4x-10000	<==>	Local HMI	RW-0	55536 Word(s)	Read/Write

- SCADA can only read / write the register defined in one Address Mapping Table at one time, that is, the same MODBUS command cannot access the data in the registers defined in different Address Mapping Tables.

Table	Description	MODBUS Address		PLC Name	Mapped PLC Address	Table Size	Read/Write
1	Access D200 ~ D298	4x-1	<==>	FATEK FB Series	D-200	99 Word(s)	Read/Write
2	Access R0 ~ R99	4x-100	<==>	FATEK FB Series	R-0	100 Word(s)	Read/Write

As shown in the above figure, in Mapping Table 1 set MODBUS 4x-1 to access register D-200, table size 99 words, and in Mapping Table 2 set MODBUS 4x-100 to access register R-0, table size 100 words. If using SCADA to send a command to read from 4x-1 to 4x-199, table size 199 words, since the range spans two different tables, the command will not be accepted by HMI. Instead, access the data with two separate commands (4x-1~4x-99 and 4x-100~4x-199), each reading only from one table as shown in the following figure.