HollySys

LM MICRO SERIES PLC

PowerPro PLC Programming Software User Manual

Revision 1.0 ©2008, MAN-PLC-LM-2008-08-20001
HOLLYSYS (ASIA PACIFIC) PTE LTD
200 Pandan Loop, #08-01,
Pantech 21, Singapore 128388
Phone +65 6777-09507 • Fax +65 6777-2730

Copyright Notice

Permitted Uses and Restrictions on Use

Disclaimer

The materials contain in this manual are provided for general information purpose only. Whilst every care has been taken to ensure the accuracy of the information provided, we can accept no responsibilities for loss or damage which may arise from reliance on the information contained in this manual.

The text and charts of this manual have been checked to be consistent with all the hardware equipment mentioned in the content, however, errors are still unavoidable and completed consistence may not be guaranteed. All materials and content is subjected to changes without prior notifications.

Specifications, charts, simple programs and application examples in this manual, which are only raised for illustration purpose, have been tested. However, because of the update of software version and other various unforeseeable changes, the companies shall not undertake any responsibilities of applications according to this manual.

Trademarks

HollySys and the logos, trade names, and product names are trademarks or registered trademarks of Beijing HollySys Co., Ltd and HollySys (Asia Pacific) Pte Ltd in PRC and other countries.

Microsoft, Windows and Windows NT are trademarks or registered trademarks of Microsoft in the United States and/or other countries branches.

Other trademarks or registered trademarks in this manual belong to their respective owners.

Contact Us

HollySys (Asia Pacific) Pte Ltd
Address: 200 Pandan Loop, #08-01 Pantech 21, Singapore 128388
Tel: (+65) 6777-0950
Fax: (+65) 6777-2730
Http: //www.hollysys.com.sg
Technical Support: PLC_support@hollysys.com.sg
Revision 1.0, Edition: September 2008

Preface

LM Micro series PLC, a new generation of smart PLC products by Hollysys Co., Ltd, consists of various kinds of CPU modules and expansion modules. Because of their advantages such as stable performance, reliable quality and affordable price, the LM Micro series PLC products have achieved wide applications in the vast range of all automation industry and earned good reputation from our clients.

The PowerPro is a Windows-based programming tool specially developed for the LM Micro series PLC by HollySys. It is the standard software package for LM Micro series PLC hardware configuration and software programming. PowerPro has the following features:

- Fully consistent with IEC61131-3 standard, support programming languages such as LD, IL, ST, FBD, SFC, CFC, etc.
- Supporting more than 400 instructions, various data types, flexible programming and high program execution efficiency.
- Supporting plentiful expansion instructions, support user-defined libraries to improve program reusability and function expansion.
- Strong mathematical calculation functions that support floating point calculation and multidimensional array.
- Powerful software stimulation, online debugging and program examination; Supporting stimulation with single step, single circle, breakpoint setting, force values, etc.
- Completed visualization, alarm and log functions that realize the visualization for control process.
- Strong password protection that allows the setting of 8 different levels of passwords and accessibilities.

Application Scope

The content of this manual is applicable to PowerPro Version 2.1.

How to Use

For first-time users, a thorough reading of the whole manual is necessary. Experienced users can use the Content and the Guide to find relevant information.

More References:

- LM Micro PLC Overview
- LM Micro PLC Selection Guide
- LM Micro PLC Hardware Manual
- LM Micro PLC Instruction Sets Reference Manual

Table of Contents

USERS' GUIDE

This manual is designed to help users writing PLC controlling programs with PowerPro software that it mainly discussed how to use PowerPro software and standard programming languages.

- **Chapter 1** An introduction of the installing, uninstalling and install target of PowerPro.
- **Chapter 2** An overview of PowerPro and its programming environment, including the main interfaces, menu bar, shortcut tools, objects organizer, etc. Users may refer to this chapter if they need to know the menu and shortcut options.
- **Chapter 3** A quick start guide taking a simple programming process as the example to show the basic steps and usages of the PowerPro software. First-time users are recommended to read through this chapter.
- **Chapter 4** An introduction of LM Micro series PLC storage assignment and the variable management in PowerPro that includes address and variable definitions, variable classifications. For any applications of addresses and variables, please refer to this chapter.
- **Chapter 5** A brief introduction of the POU management in PowerPro, such as POU creating, POU calling, etc. In this chapter users may find answers for questions in creating a new program.
- **Chapter 6** An introduction of PLC working mode, task management and configuration based on Chapter 5. For interrupt handling, please refer to this chapter.
- **Chapter 7** Based on the knowledge of address, variable and POU management of PowerPro, this chapter gives an introduction on how to write a program with PowerPro software. . In this chapter, a LD language programming process has been used as an example to illustrate steps of the creation and management of a project, such as creating new project, PLC configuration, programming, subprogram calling and comments adding. For complete project programming, please refer to this chapter.
- **Chapter 8** An introduction of steps after programming such compilation, download and debugging.. Users may refer to this chapter for commands and problems they might meet in the execution.
- **Chapter 9** An introduction of programming languages used in PowerPro, such as FBD, IL, ST and SFC. Please refer to this chapter for the usage of these languages.
- **Chapter 10** An introduction of special functions of LM PLC such as Modbus communication, interrupt, analog functions, DP communication and Ethernet communication. Refer to this chapter for usages of these functions.
- **Chapter 11** An introduction of PowerPro visualization, which is an advanced application of PowerPro software. Please refer to this chapter for debugging on a visual interface.

Chapter 1

Installation Guide

This chapter gives an introduction to the installation and un-installation of the PowerPro software application and a detailed description of its Install Target.

- **Section 1.1** introduces the installation process of the software
- **Section 1.2** deals with the un-installation. For users who are very familiar with Windows operation systems, these 2 sections may be skipped.
- **Section 1.3** gives detailed instructions on the definition of the Install Target which configures PowerPro, the PLC programming software, as the default programming software for LM Micro series PLCs. A thorough reading of this section is recommended to first-time users of PowerPro.

1.1 INSTALLING THE SOFTWARE

- Insert the installation CD of PowerPro software into the Cd drive of the windows operating system.
- Locate the SETUP.EXE program found on the CD and double click it to run and install the PowerPro software.
- Click the “Next” button on the installation interface, as shown in figure 1-1-1.

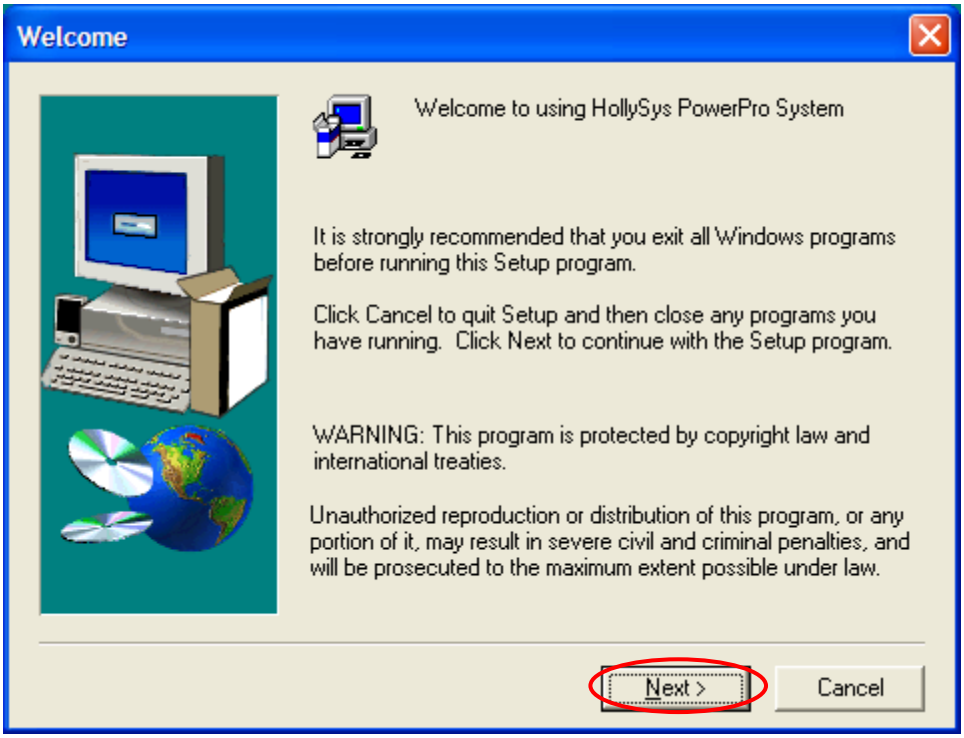


Figure 1-1-1 Installation Steps (1)

- Accept the license agreement and click “Yes”, as shown in figure 1-1-2.

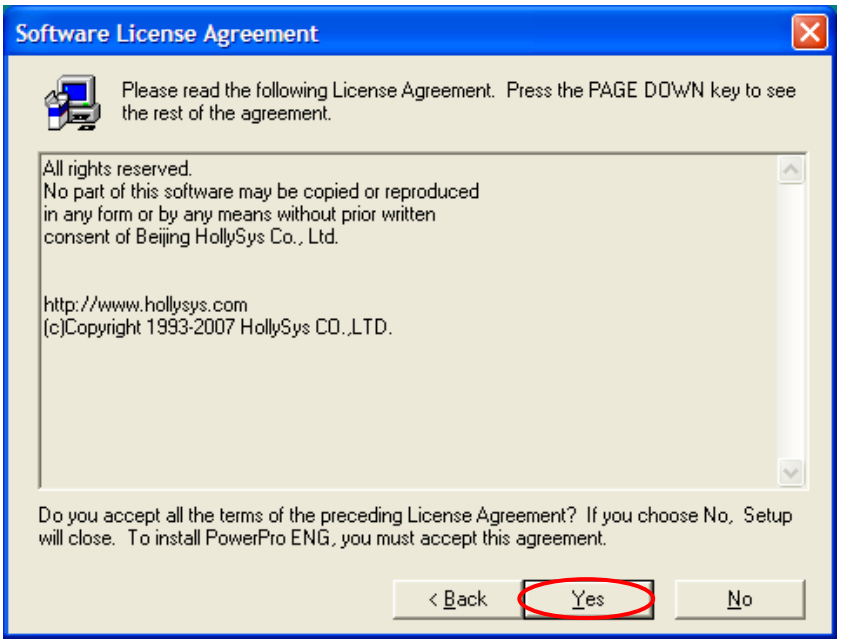


Figure 1-1-2 Installation Steps (2)

- Choose installation directory. The default directory is “D:\Hollysys\PowerPro” and it is recommended not to change the installation path. Click “Browse” to choose other directories. Click “Next” to start installation, as shown in figure 1-1-3.

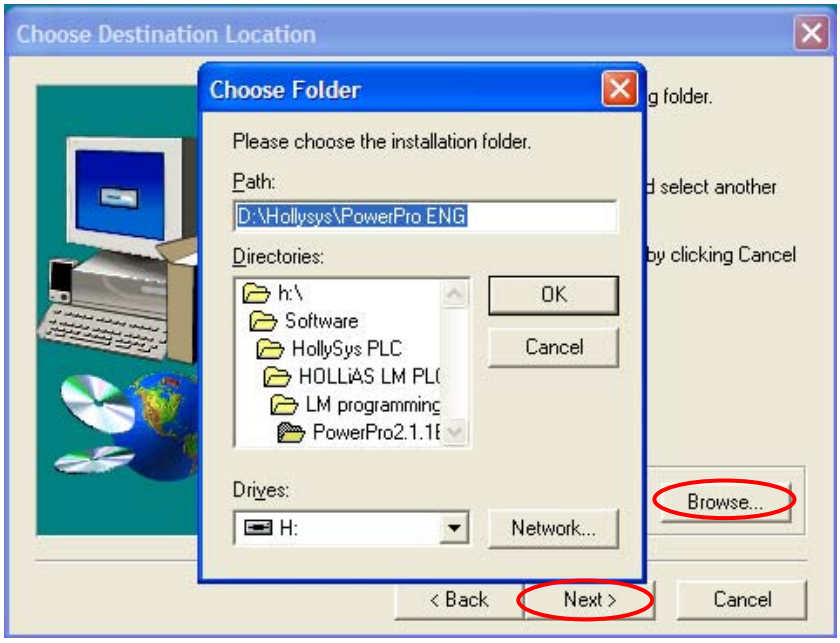


Figure 1-1-3 Installation Steps (3)

- After installation, an interface “About to launch the MSXML 3.0 setup” will appear. Then click “OK”, as shown in figure 1-1-4.

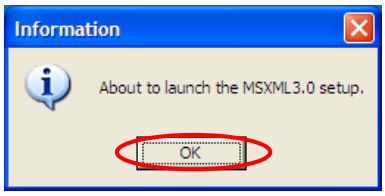


Figure 1-1-4 Installation Steps (4)

- Enter MSXML 3.0 setup wizard and click “Next”, as shown in figure 1-1-5.

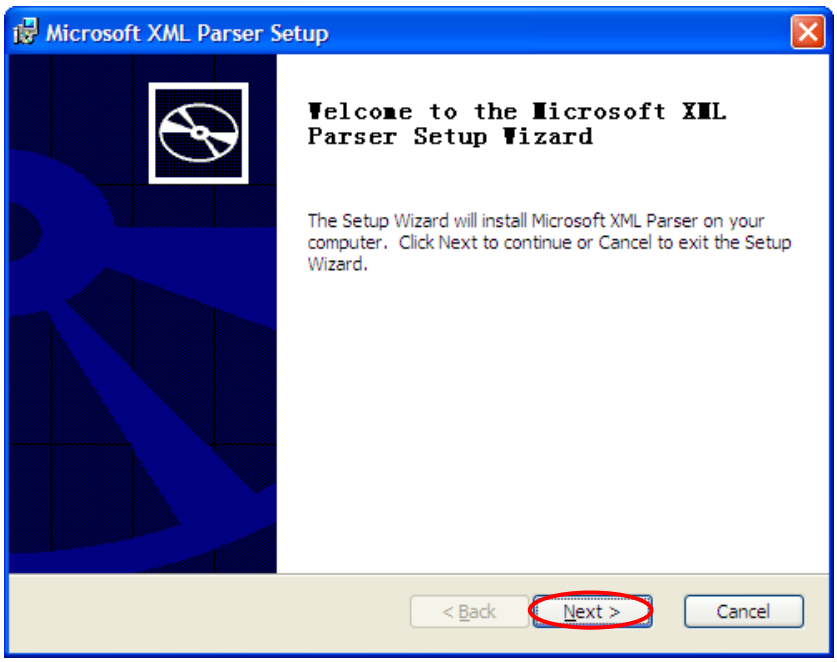


Figure 1-1-5 Installation Steps (5)

- Select “I accept the terms in the License Agreement” and click “Next” button, as shown in figure 1-1-6.

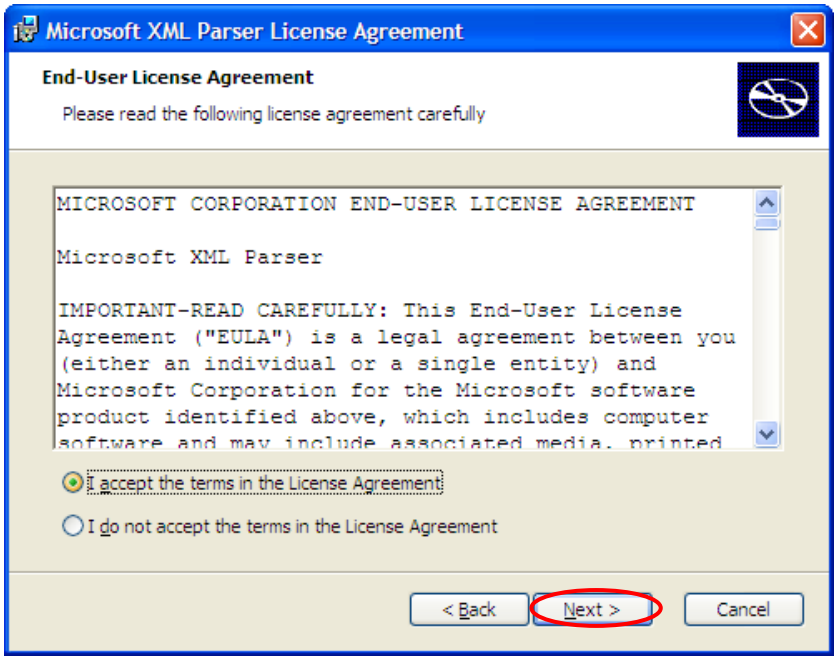


Figure 1-1-6 Installation Steps (6)

- Fill in User Name and Organization and click “Next” button, as shown in figure 1-1-7.

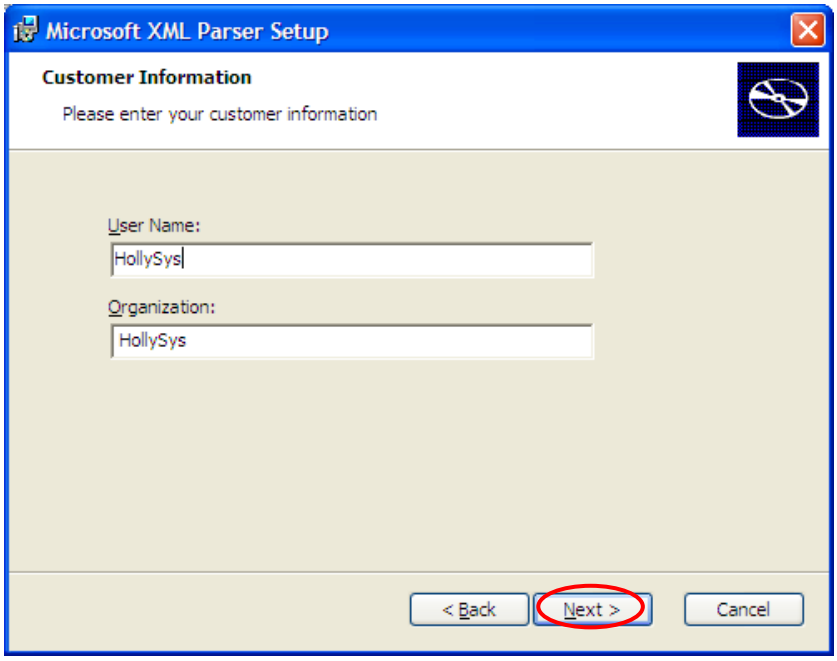


Figure 1-1-7 Installation Steps (7)

- Click “Install” button to begin the installation, as shown in figure 1-1-8.

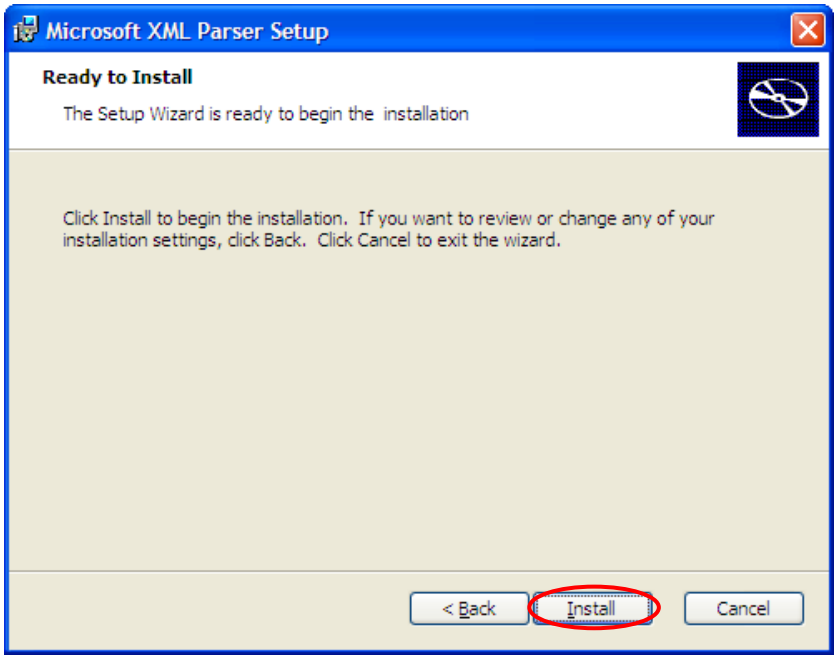


Figure 1-1-8 Installation Steps (8)

- After installation, click “Finish” button to exit the setup wizard MSXML 3.0, as shown in figure 1-1-9.

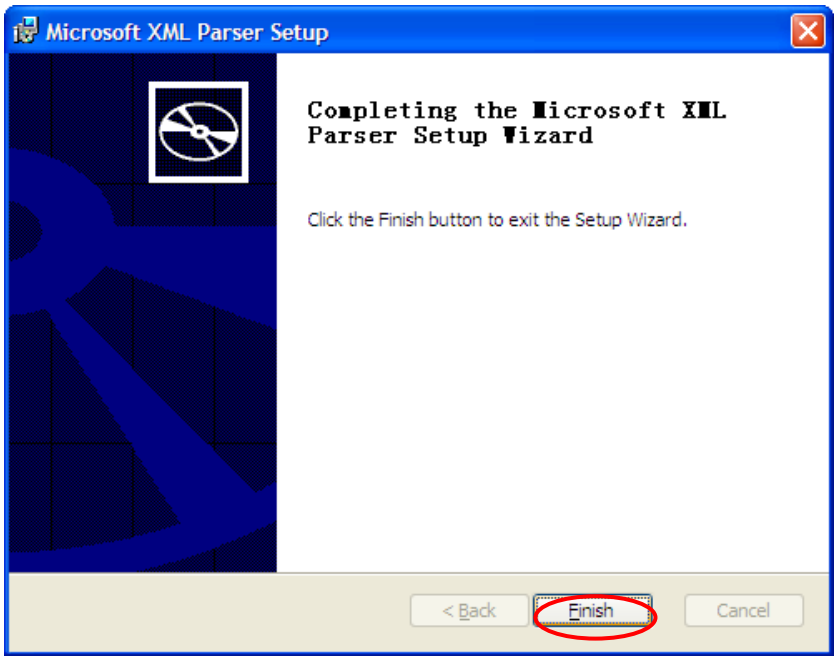


Figure 1-1-9 Installation Steps (9)

- Then an interface “About to launch the MSXML 3.0 SP4 setup” will appear, and then click “OK”, as shown in figure 1-1-10.

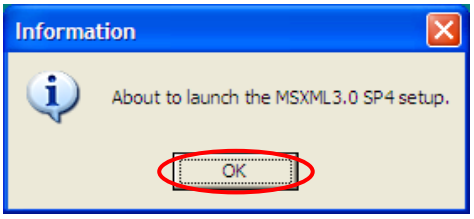


Figure 1-1-10 Installation Steps (10)

- Enter MSXML 3.0 SP4 setup wizard and click “Next”, as shown in figure 1-1-11.

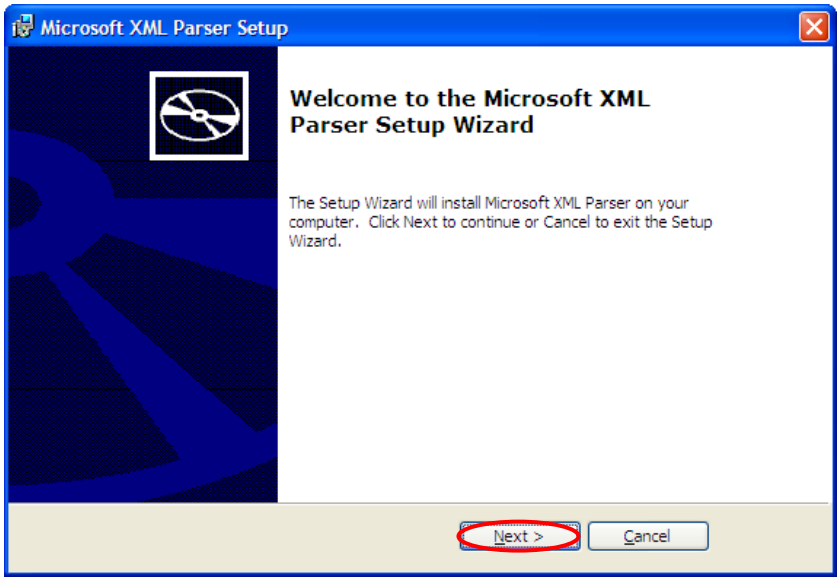


Figure 1-1-11 Installation Steps (11)

- Select “I accept the terms in the License Agreement” and click “Next” button, as shown in figure 1-1-12.

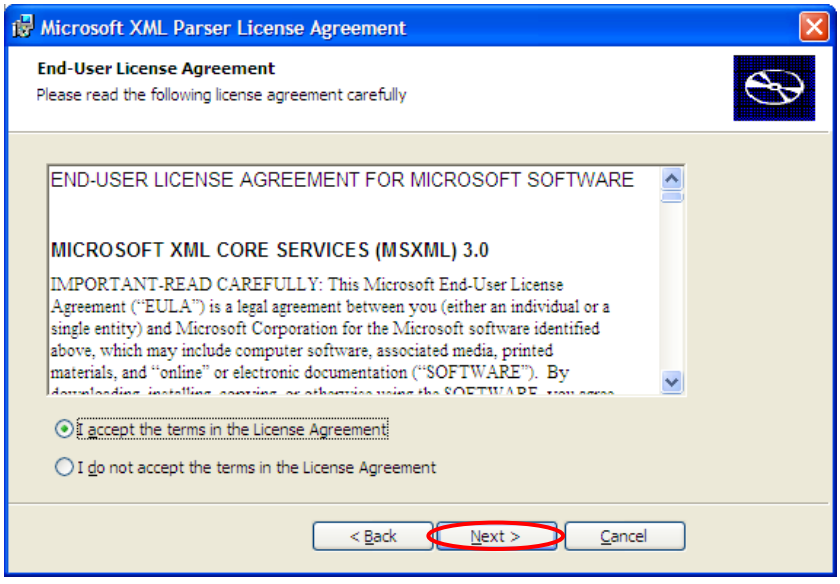


Figure 1-1-12 Installation Steps (12)

- Fill in User Name and Organization and click “Next” button, as shown in figure 1-1-13.

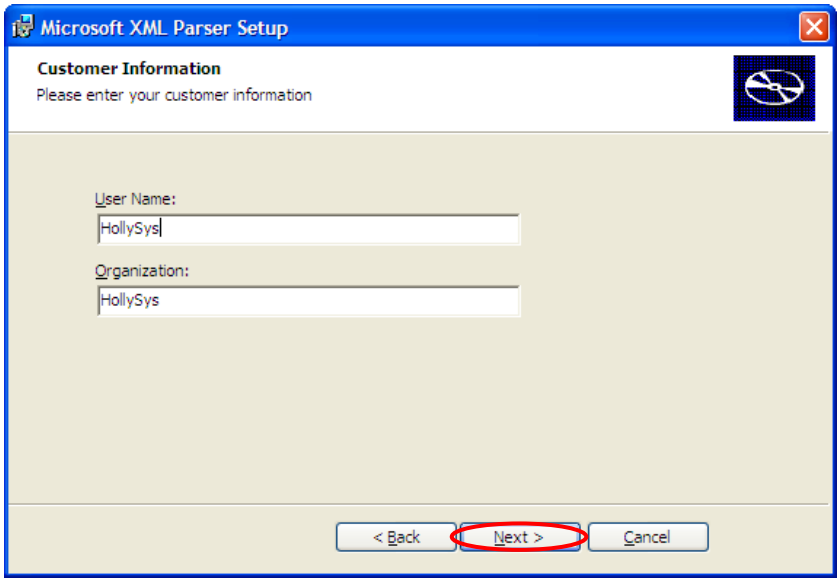


Figure 1-1-13 Installation Steps (13)

- Click “Install” button to begin the installation, as shown in figure 1-1-14.

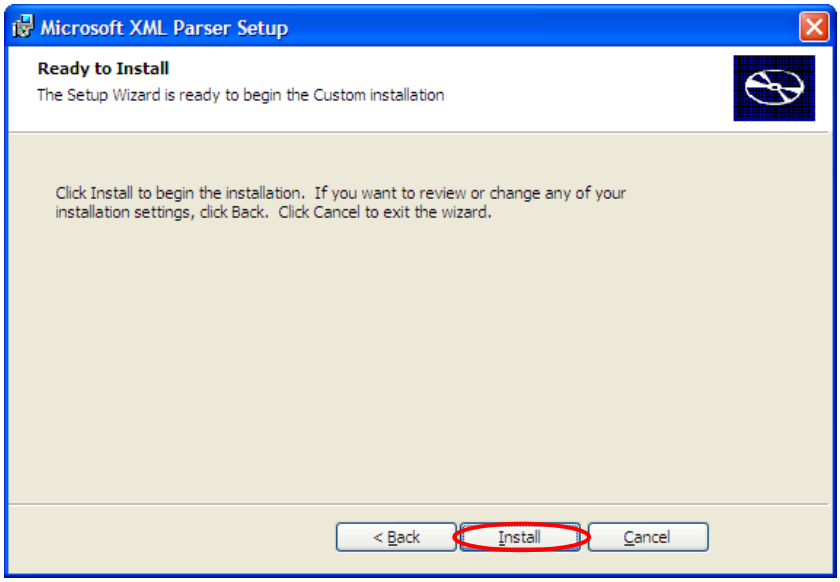


Figure 1-1-14 Installation Steps (14)

- After installation click “Finish” button to exit the setup wizard MSXML 3.0 SP4, as shown in figure 1-1-15.

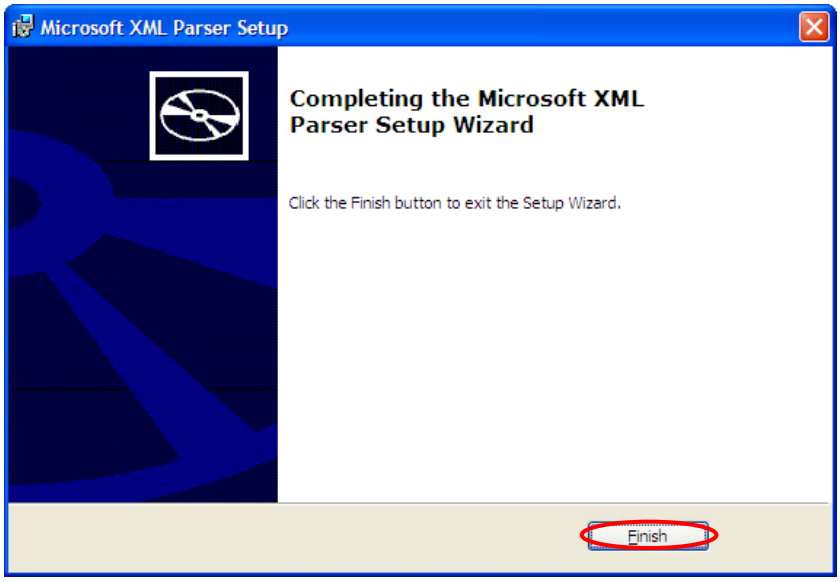


Figure 1-1-15 Installation Steps (15)

Click “Finish” button to finish the setup of PowerPro, as shown in figure 1-1-16.

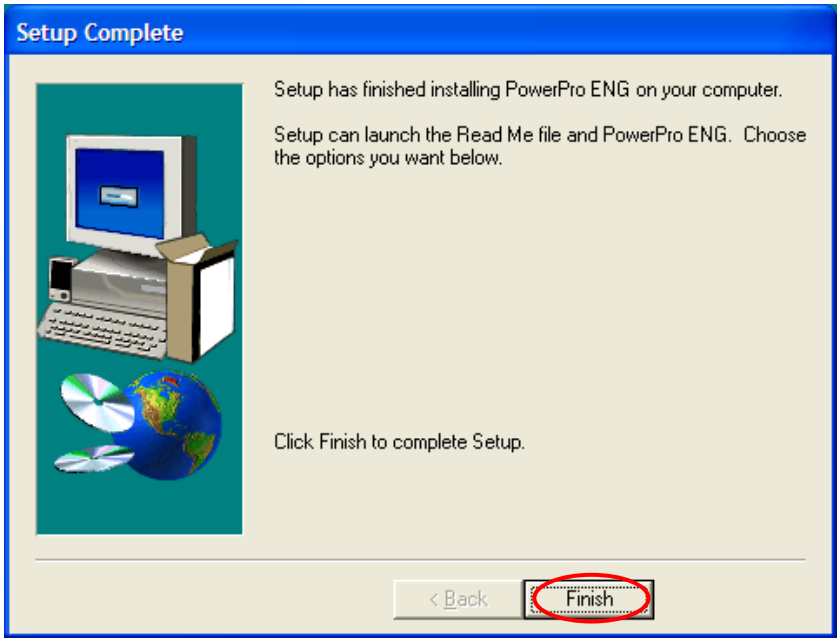


Figure 1-1-16 Installation Steps (16)

1.2 UNINSTALLING SOFTWARE

If a lower version of PowerPro has been installed on the computer, it must be uninstalled before the installation of a new version. Select PowerPro in “Control Panel”/“Add/Remove Programs” and click “Change/Remove” to uninstall the program, as shown in figure 1-2-1.

NOTE:
Please EXIT the Gateware.exe (if any) in the system tray on the lower right corner of the desktop before trying to start the un-installation of PowerPro Software.

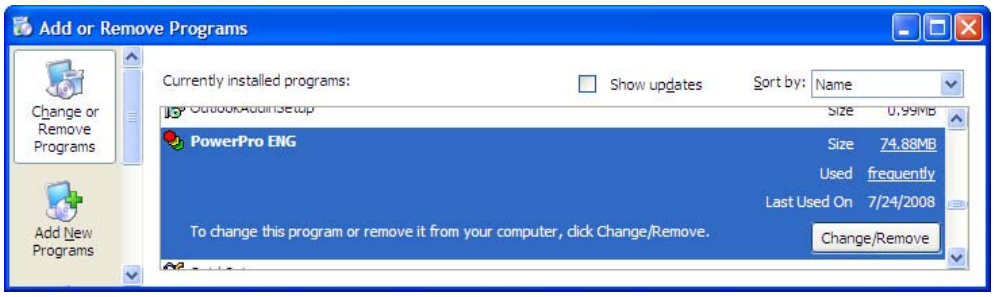


Figure 1-2-1 Software Uninstall

1.3 INSTALL TARGET

PowerPro is the development platform of PLC control solutions. Before using the PowerPro application, the "**Install Target**" must be configured first. Because the installation is a global setting for all projects, only one "**Install Target**" configuration is required before using the PowerPro application.

1.3.1 How to configure the "Install Target"

- Click "Start"/"Programs"/"Hollysys PowerPro ENG"/"Install Target" on the desktop, as shown in figure 1-3-1.

Figure 1-3-1 Install Target (1)

- The widow "**InstallTarget**" appears, as shown in figure 1-3-2. Click "**Open**." button, choose the file "**C16x_hollysys.tnf**" in the pop-up window and click "**Open**".

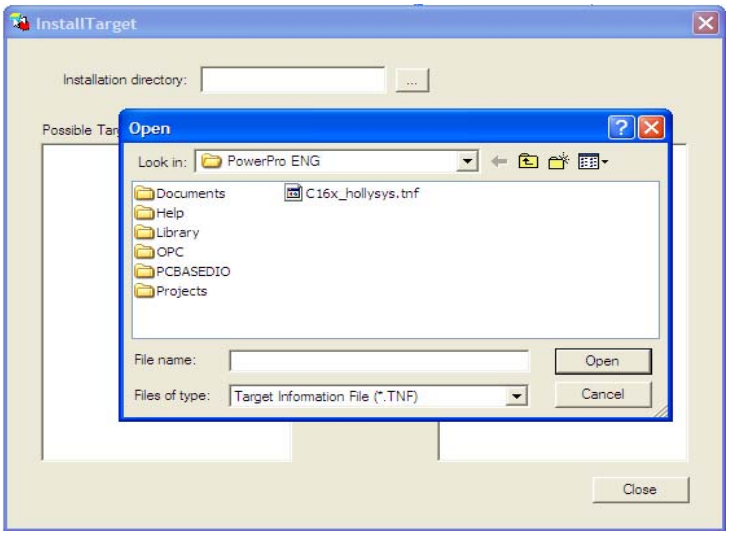


Figure 1-3-2 Install Target (2)

- Then a target "**Hollysys PLC**" is displayed in the window of "**Possible Targets**" on the left of "**InstallTarget**" window. Select the target "**HollySys PLC**" and click the button "**Install**", as shown in figure 1-3-3.

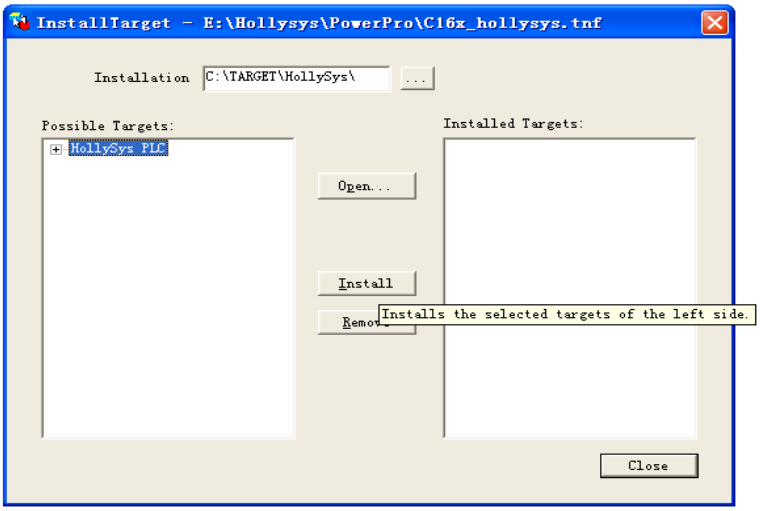


Figure 1-3-3 Install Target (3)

- Then the same target file "**Hollysys PLC**" is generated in the window of "**Installed Targets**" on the right of the "**InstallTarget**" window. Click "**Close**" button to end the installation, as shown in figure 1-3-4.

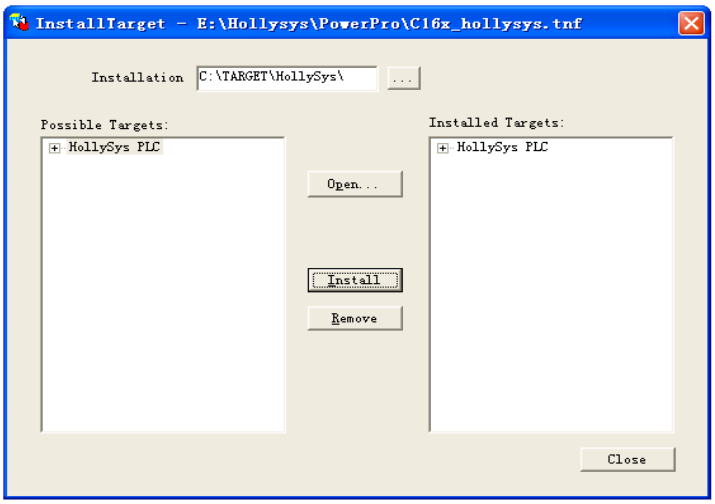


Figure 1-3-4 Install Target (4)

Chapter 2

An Overview of the PowerPro Software

This chapter gives an overview of the PowerPro programming environment including programming interfaces, menu commands, etc. so that you can understand and be familiar with the programming environment.

Start PowerPro to enter PowerPro programming environment.

2.1 A BRIEF INTRODUCTION OF POWERPRO

PowerPro is a Windows based programming tool which is developed for LM Micro series PLC by HollySys. PowerPro with its editing, simulation, debugging functions for PLC controlling solutions, has become a standard software package for hardware configuration and software programming of LM Micro series PLC.

Comparing with the traditional programming software, PowerPro has the following features and functions:

2.1.1 Standardized programming language

In the mid and late 1990's, IEC issued the international standards programming languages for the automation industry, first IEC1113-3 and later a modified IEC61131-3, to unify PLC, NC and DCS programming languages. Fully complied with IEC61131-3 standard, PowerPro programming software provides multi-programming language modes including IL, ST, SFC, FBD, LD and CFC.

2.1.2 Variable instead of component

Unlike other PLC products, LM Micro series PLC uses less inner components such as timers, counters and replaces them by variables. Variable is a special concept in PowerPro that is similar to an advanced programming language. Variables are declared according to the accurate usages. They may also be named according to their functions that made it much easier to identify them than only numbers. Variables may be declared into different categories such as global or local, input or output, retained or non-retained. At the same time, because of the powerful calculation function of PowerPro, a number of data types can be defined, including not only Boolean, Byte, Word, Double Word types but also single, pointer, enumerate, and multi-dimension array. ***For detailed parameter description, please refer to section 4.4 of this manual.***

2.1.3 Modularized program organization

The program organization of PowerPro is completely modularized. The concept of POU (Program Organization Unit) is promoted in PowerPro. The POU of PowerPro consists of program, function, and function blocks which compose a project. PowerPro achieves organization of a program by the main program calling other POUs. This does not only allow multi-programmers participation, convenient of reuse of codes, reading, debugging of the program, but also save memory space ensuring a safer operation of the program. Meanwhile, PowerPro is an open system that allows its users to develop their own instructions when necessary. *Also refer to chapter 5 for more details of POU.*

2.1.4 Module setting software

PowerPro is an open system that, on one hand allows its users to develop their own instructions when necessary and on the other hand opens up a large number of PLC parameters and module settings to users in the form of instructions. Therefore users may set parameters in programs according to their respective requirements, such as the setting of serial port communication parameter.

2.1.5 Integrated programming and monitoring

With unique visualization and alarm functions, PowerPro provide a visualized interface for program running and debugging. In addition, PowerPro also provides powerful simulation and debugging functions that ease the tests of the logical correctness of users programs. *For more information of simulation and debugging functions, please refer to section 8.4. For more information of visualization please see chapter 11.*

2.2 PROGRAMMING WINDOW

Start PowerPro and enters the main interface, as shown in figure 2-2-1.

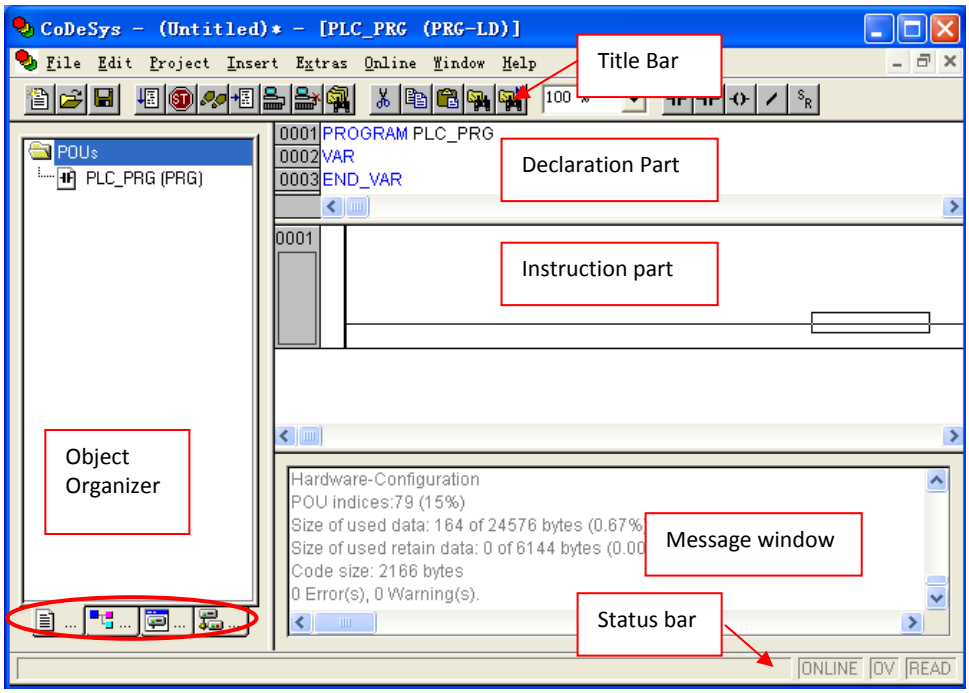


Figure 2-2-1 Main Window

The following elements are found in the main window of PowerPro:

- **Title Bar**: including Menu bar (which contains all menu commands) and Tool bar (optional), with some buttons for faster selection of menu commands.
- **Object Organizer**: including POUs, Data Types, Visualizations and Resources.
- **Variable Declaration Window**: displaying the variables declared or defined in program.
- **Instruction Window**: where one can edit and modify the programs.
- **Message Window**: Display the previous compilations, including basic message, errors and warnings.
- **Status Bar**: Display the information about the current project and about menu commands.

2.2.1 Title Bar:

Run PowerPro, the Title bar will be found located at the upper edge of the main window, as shown in figure 2-2-2. It consists of menu bar (including “File”, “Edit”, “Project, “Insert”, “Extra”, “Online”, “Window”, “Help” options) and Tool bar (optional) which provides some buttons for faster selection of menu commands.

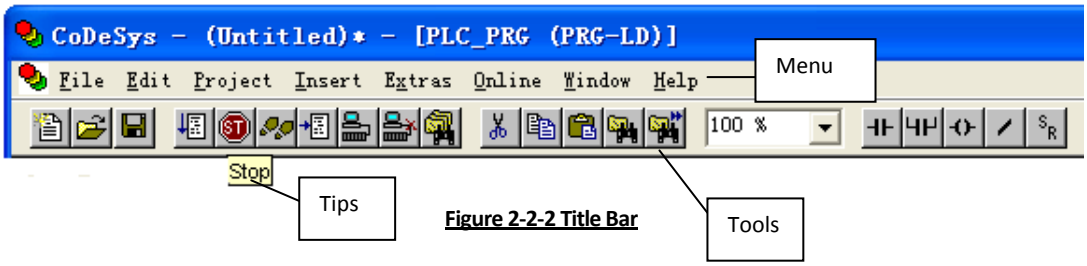


Figure 2-2-2 Title Bar

First let us take a look at the shortcut buttons on the tool bar. Hove over a button for a short while,, the name of the button will be shown in a tooltip, as shown in figure 2-2-2. A grey menu command or shortcut button indicates that the function is disabled in the current window.

 **New**: Create an empty project with the name “Untitled”.

 **Open**: Open an already existing project.

 **Save**: Save any changes in the project.

 **Run**: Start the program in the PLC or in simulation mode.

 **Stop**: Stop the execution of the program in the PLC or in simulation mode between two cycles.

 **Login**: Combine the programming system with the PLC (or start the simulation program) and changes into the online mode.

 **Logout**: The connection tp the PLC is broken, or the simulation mode program is ended and is shifted to the offline mode.

 **Cut**: Remove the current selection from the editor to the clipboard.

Copy: Copy the current selection from the editor to the clipboard.

Paste: Paste the content of the clipboard onto the current position in the editor window.

Find: Search for a certain text passage in the current editor window.

Find Next: Execute a search with the same parameters as with the most recent action 'Edit' 'Find'.

Refer to section 2.3 for menu commands.

2.2.3 Object Organizer

The Object Organizer is the vertical window located on the left side of the main interface. At its bottom there are four tabs for the four types of objects POUs, Data Types, Visualization and Resources, as shown in figure 2-2-3.

The POUs tab is used for POUs management, such as new subprogram and new interrupt service routine. The Data Type tab is used to manage user-defined data types which are supported by PowerPro. The Visualization tab is used to manage visualizations. The Resources tab is used for the functions such as PLC hardware configuration, adding instructions, workspace and interrupt setting.

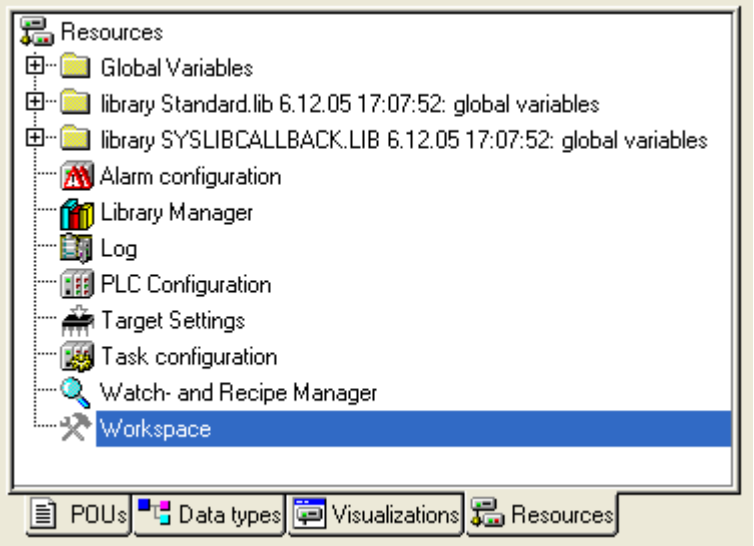


Figure 2-2-3 Object Organizer

2.2.4 Declaration Window

The declaration window is located on the upper right corner of object organizer. The data in PowerPro are categorized into addresses and variables. Variables may be assigned no specific address and identified directly by symbols, such as "start", "run", and the same symbol represents the same variable. The difference between variables and addresses is that variables must be defined when used while addresses can be referenced directly. The declaration window is used to display all the defined variables.

There are two ways to define variables. First, variables can be automatically defined while programming which are displayed in declaration window, as shown in figure 2-2-4; Second,

variables can be defined directly in the declaration window. ***Refer to section 4.4 for variable definitions.***

The declaration window can be displayed in text or table forms and the declaration window, in figure 2-2-5 is displayed in table form.

Note:
The same variable symbol cannot be defined as two different data types.
The address data will not be displayed in declaration window.
When a defined variable is deleted in programs, the variable declaration in declaration window will not be deleted automatically.

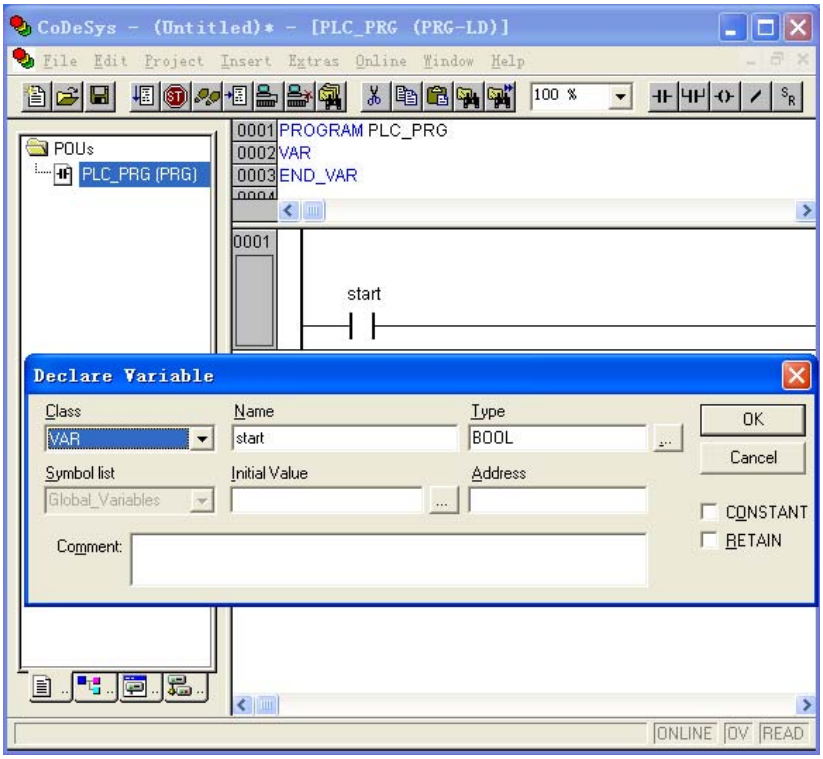


Figure 2-2-4 Auto-declaration

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	Name	Address	Type	Initial	Comment
0001	sw		BOOL		
0002	t1		TON		
0003	DES1		des		

Figure 2-2-5 Declaration in a Table Form

2.2.5 Instruction Window

The Instruction Window is located below the Declaration Window. On the “POUs” tab in the object organizer, the instruction window is the editor window of programs, functions and functional modules, and it is used to edit control algorithms. The programming environment varies with the selected programming language. According to the characteristics of different languages, the programming language is classified into textual languages and graphical

languages. The editor for LD, SFC, FBD and CFC languages are graphical while those for IL and ST languages are text editors that have the usual capabilities of Windows text editors.

For more information of operations in instruction window, please refer to section 7.4. For other programming languages please refer to chapter 9.

2.2.6 Message Window

Located underneath the instruction window on the main interface, the message window displays compilation, error, warning or comparison messages of the program in real-time, as shown in figure 2-2-6. To search for relevant information, double-click on a message in this window, the editor will open with the relevant line of the object. With the commands 'Edit' 'Next error' (F4) and 'Edit' 'Previous error' (Shift+F4 combination key), it is possible to quickly jump between the error messages. The display of the message window is optional. If the messages window is open, then a check (√) will appear in front of the command.

In addition, the message window can also display cross-reference list, such as unused variables or overlapping memory area, etc. ***Please refer to section 8.2.3 for more details.***

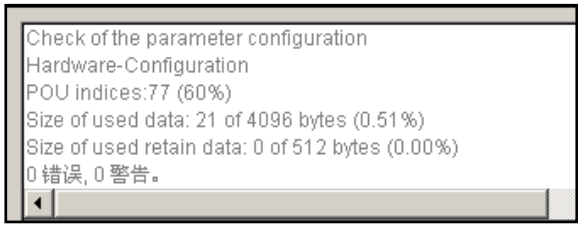


Figure 2-2-6 Message Window

2.2.7 Status Bar

The status bar that locates at the bottom of the main interface frame gives information of the current project and command. Once an item is selected, the relevant information will appear in black script on the status bar.

2.3 MENUS

2.3.1 File

File menu is shown in figure 2-3-1. Detailed functions are as follows:

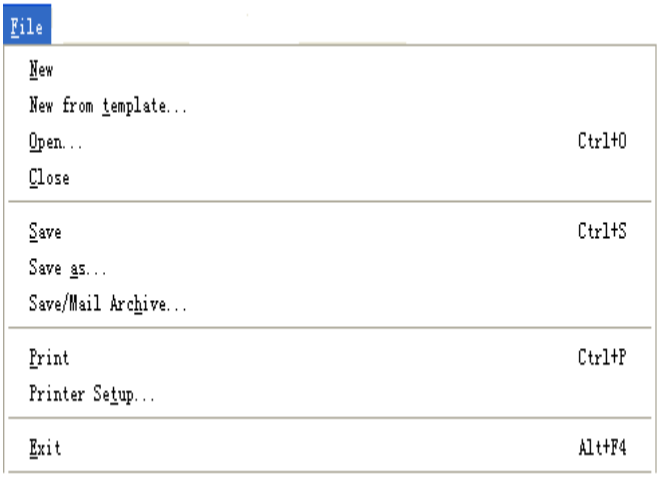


Figure 2-3-1 File Menu

- **File [F]/New [N]**: Create an empty project.
- **File [F]/New from template [T]**: Create a new project from an existed template. Click on this item to open a dialog, select an existed template, click on “Open”, then a new project will be created from this existed template. PLC does not support the two functions of “Open project from PLC” and “Open project from source code manager”.
- **File [F]/Open [O]**: Open an existed project.
- **File [F]/Close [C]**: Close the currently-open project.
- **File [F]/Save [S]**: Save changes in the project.
- **File [F]/Save as [A]**: Save the current project as a new file or into a new directory.
- **File [F]/Save /Mail Archive [H]**: Automatically create a project archive of all references, compress it to a zip file and email it.
- **File [F]/Print [P]**: Print the content of the current window.
- **File [F]/Printer Setup [T]**: Setup the printer properties. Click on this item to open a dialog where the printer may be selected and the page size, number of copies, printing directions, quality and layout can be set.
- File [F]/Exit [E]: Exit PowerPro.

2.3.2 Edit

Edit menu is shown in figure 2-3-2. Detailed functions are as follows:

Figure 2-3-2 Edit Menu

- **Edit [E]/Undo [U]**: Undo the previous action..
- **Edit [E]/Redo [E]**: Repeat previous action.
- **Edit [E]/Cut [T]**: Cut the selected content from the editor to the clipboard.
- **Edit [E]/Copy [C]**: Copy the selected content to the clipboard.
- **Edit [E]/Paste [P]**: Paste the content from the clipboard onto the current position .
- **Edit [E]/Delete [D]**: Delete the selected content.
- **Edit [E]/Find [F]**: Find certain text in the current editor window.
- **Edit [E]/Find next [N]**: Find the same text as the last search
- **Edit [E]/Replace [R]**: Find certain text and replace it with another text.
- **Edit [E]/Input Assistant [A]**: quick entry of relevant content. At the current cursor in the editor window, press F2, and items that can be inserted at the current location will pop out automatically, such as lists of operators, functions, modules and variable types.

Select the input type from the popup list on the left, and select the expected input from the list on the right, select the content that needs to be entered, and click "OK", then the selected content will be entered. *For details, refer to Section 7.4.3.*

- **Edit [E]/Auto Declare**: open a Variable declaration dialog.
- **Edit [E]/Next Error**: Find and display the next error or warning message in the message window.
- **Edit [E]/Previous Error**: Find and display the previous error or warning message in the message window..
- **Edit [E]/Macros**: this function is disabled for PLCs.

2.3.3 Project

Project menu is shown in figure 2-3-3. Detailed functions are as follows:

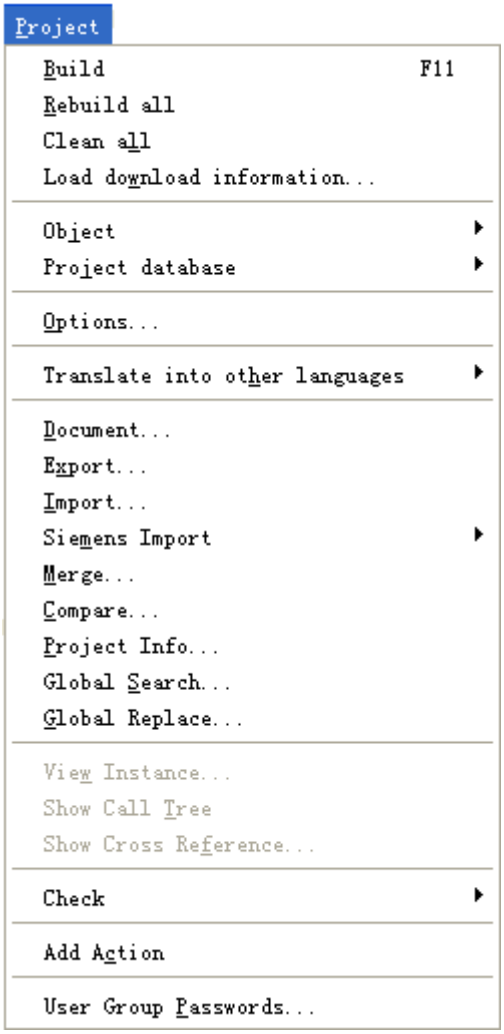


Figure 2-3-3 Project Menu

- **Project [P]/Build [B]**: Compile changes to POUs.
- Project [P]/Rebuild all [R]: Compile all the POUs .
- **Project [P]/Clean all [L]**: Delete all downloaded files from the previous compilations. To “Clean all” is to ensure the system creating new download files in new compilations, but this action has no influence on the user program in PLC. Perform “Build” or “Rebuild all” after “Clean all”, in the next program download, a message “the program has been changed! Download the new program?” will pop up, then finish download

according to the message. It is different from “Online/Clean All User Program” function that cleans all the programs in PLC and re-initializes PLC system.

- **Load download information**: This function is disabled in PLC.
- **Project [P]/Object [J]**: Delete, add, rename, covert, copy, edit or modify properties of the selected object, as shown in figure 2-3-4.

Figure 2-3-4 Object Sub-menu

- **Delete**: Delete the current object.
- **Add**: Add a new object in current POU, select programming language and name.
- **Rename**: Rename the current object.
- **Convert**: Convert current object to other programming languages. For example: convert current object language from LD to target language IL or FBD, as shown in 2-3-5.

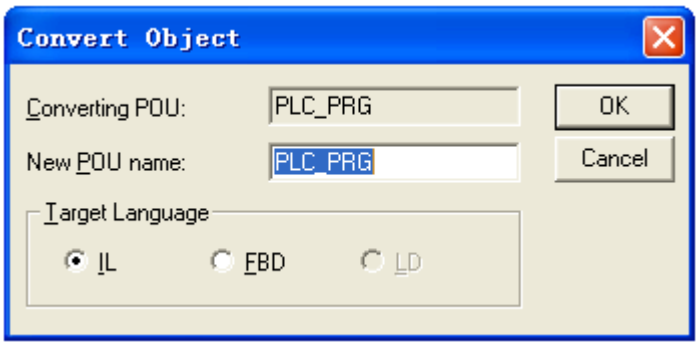


Figure 2-3-5 Convert Object

- **Copy**: Copy a selected object and save it under a new name.
- **Edit**: Open the editor window of the selected POUs. The editor window may also be opened by double-click on the POU name.
- **Properties**: Set user’s access rights to the current POU, as shown in figure 2-3-6.

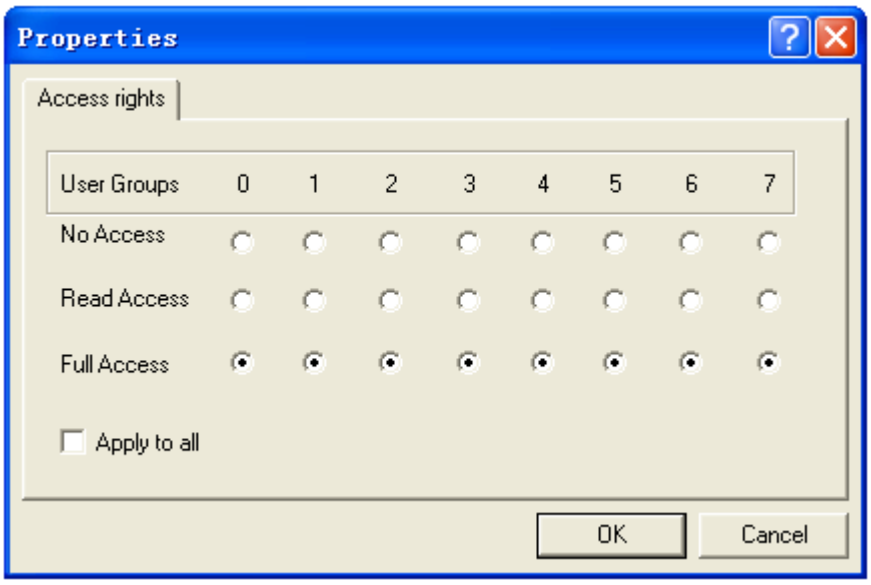


Figure 2-3-6 Properties Dialog

- **Project [P]/Project database [J]**: This function is disabled in PLC.
- **Project [P]/Options [O]**: Set the parameters of current project, such as Load&Save, Directories, Passwords, etc. ***Please refer to section 7.6 for more details.***
- **Project [P]/Translate into other languages [H]**: Translate the current project into other languages, as shown in figure 2-3-7.
- **Create translation file**: Create a translation file with the extension ".tlt" for the current project.
- **Translate this project**: Translate the current project to target language.
- **View translated project**: View the translated project files.
- **Toggle translation**: Once Toggle Translation is clicked, the "Translate this Project" option will be displayed in grey, and the project cannot be translated. The translation of the project is allowed only after Toggle Translation is clicked again.

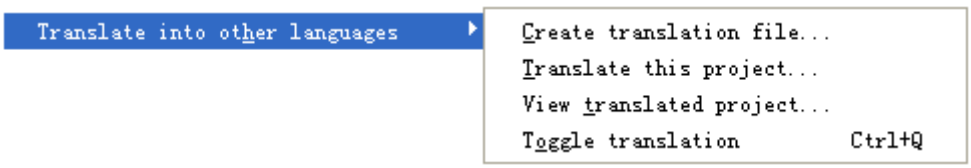


Figure 2-3-7 Language Dialog

- **Project [P]/Document [D]**: Print the entire or partial documentations of current
- **Project [P]/Export [E]**: Export the selected POUs of current project into a *.EXP file.
- **Project [P]/Import [I]**: Import a *.EXP file to current project. This option is usually used to the import of POUs between different projects. ***Refer to section 7.5.2 for more details.***
- **Project [P]/Siemens Import [M]**: this function is disabled in PLC.

Figure 2-3-8 Siemens Import

- **Project [P]/ Merge [M]**: merge the necessary content from other projects into current project.

- **Project [P]/Compare [C]**: Compare the “POUs”, “Data Types”, “Visualization”, “Resources” of the current project with those of other projects, and display the results in message window.
- **Project [P]/Project info [P]**: Show the information about the current project, such as file names and directories.
- **Project [P]/Global Search [S]**: Search for a certain text in some objects of the current project.
- **Project [P]/Global Replace [R]**: Search for a certain text in some objects of the current project and replace this text by another.
- **Project [P]/View Instance [W]**: Open instances of function blocks to be viewed. Login online with compiled and error-free programs before viewing any instance.
- **Project [P]/Show Call Tree [T]**: Open the Call Tree Window. The project must be completely compiled before viewing the Call Tree. ***Refer to section 8.2.1 for details.***
- **Project [P]/Show Cross Reference [F]**: Show the cross reference of variables and programs in current project. ***Refer to section 8.2.2 for details.***
- Project [P]/Check: View information of variables, as shown in figure 2-3-9. ***Refer to section 8.2.3 for details.***
- **Unused variables**: Automatically search for declared but unused variables in the project. If there are no unused variables, a message “No unused variables found” will be displayed in the message window.
- **Overlapping Memory Areas**: Automatically search for variables with overlapping memory areas in the project. If there are no variables with overlapping memory area, a message “No variables with overlapping memory area found” will be displayed in the message window.
- **Concurrent Access**: Automatically search in project memory areas for addresses that are concurrently accessed by more than one task. If there are no concurrent accesses, a message "No concurrent accesses found” will be displayed in the message window.
- **Multi Write Access on Output**: Search for memory areas to which a single project gains write access at more than one location. If there are no multi write access on output memory, a message “No outputs found which are written to at more than one location” will be displayed in the message window.

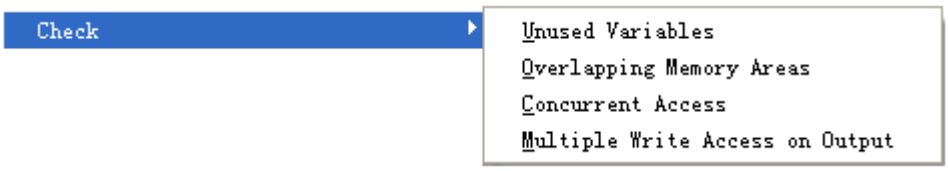


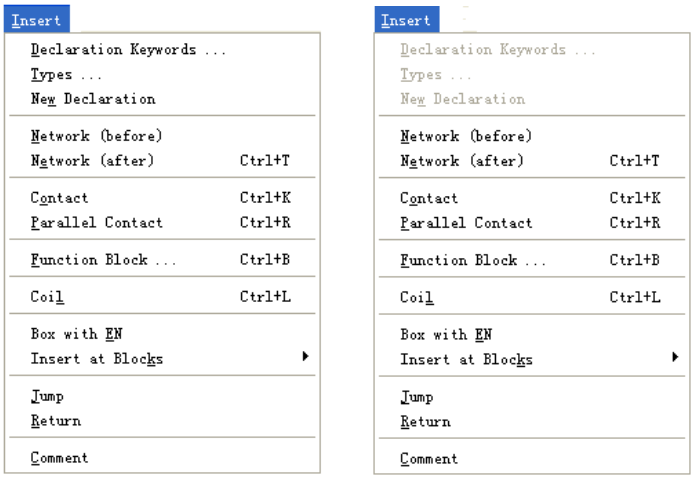
Figure 2-3-9 Check Sub-menu

- The options in “Project”/“Check” have the similar functions with those in “Workspace”/ “build”/ “Check automatically” of the “Resources” tab. The only difference is that the options in “Project / Check” are available only after the project has been fully compiled without any error and the checks are done one at a time while “Check automatically” allows the operation of more than one check during the compilation.
- **Project [P]/Add Action [A]**: Add a branch to the current program.
- **Project [P]/User Group Passwords [P]**: Assign passwords to user group. The password will be required when the project is opened next time. If the passwords match, operations will be allowed, otherwise operations are not allowed.

2.3.4 Insert

The insert menu provides options to insert new declaration and call function blocks, operators and functions. The insertion options on insert menu vary from different programming languages and cursor positions. We now take “LD” language for an example to introduce usages of the “Insert” menu in several different situations.

Firstly, the items displayed on the pull-down Insert Menu when the cursor is at declaration windows are different from those on the Insert Menu of the instruction window, as shown in figure 2-3-10. In this example you may insert Function block, Coil, Comment and Declaration Keywords, Types of LD language. ***Refer to section 7.4 for details of item insertions.***



a) cursor in declaration part **b) cursor in instruction part**

Figure 2-3-10 "Insert" Menu (1)

Secondly, the Insert menu of "PLC Configuration" on "Resources" tab,allows the insertion of modules. Users may insert the CPU module first and then insert the related expansion modules, as shown in figure 2-3-11. If the I/O channels of CPU module can meet the project requirement, then there is no need to add any expansion module.

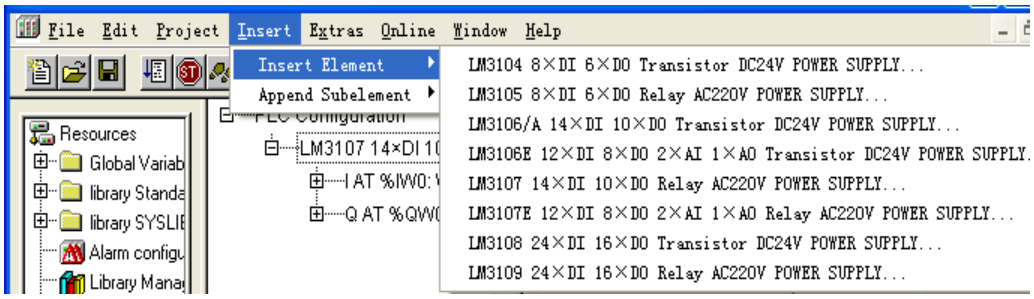


Figure 2-3-11 "Insert" Menu (2)

Thirdly, new watch lists can be inserted by using the the Insert menu under the "Watch- and Recipe Manager" of "Resources" tab, or right clicking on the left column of the "Watch- and Recipe Manager", choosing "Insert"/"New Watch List" from the context menu, and entering an appropriate name for the list as shown in figure 2-3-12.

Figure 2-3-12 "Insert" Menu (3)

2.3.5 Extras

The options under the “Extras” menu vary according to different programming languages or different cursor positions. In the following, we explain the “Extras” menu in different scenarios with the “LD” language as an example. .

First, in the “Programs” option of the “Resources” tab, “Extras” menu is as shown in figure 2-3-13.

Figure 2-3-13 “Extras” Menu (1)

- **Negate**: Use this command to negate a contact or a coil. If it is a coil, the negated value will be saved in the corresponding Boolean variable. If it is a contact, it will only be connected when the Boolean value is FALSE.
- **Set/Reset**: Coils can also be defined as set or reset coils. A set coil is represented by an “S” symbol. If a set coil is set to TRUE, it will remain so until it is reset. A reset coil is represented by an “R” symbol. If a coil is reset to FALSE, it will remain so until it is set.
- **Zoom**: Shortcut is “Alt+Enter”. In a Ladder Diagram, select a function block and press “Alt+Enter”, then the library manager or “help” files will pop up to display the usage of this block.
- **Options**: In a LD, use the “Options” menu to open a dialog “Function block and Ladder Diagram Options”, as shown in figure 2-3-14. In the dialog different options for LD POU may be set. ***Refer to section 7.4.9 for detailed options.***

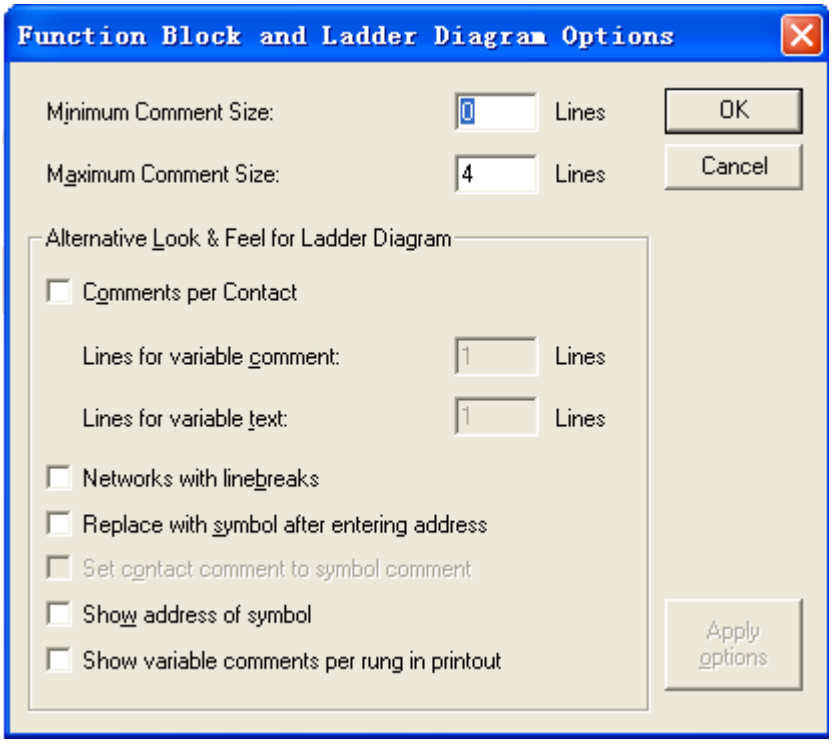


Figure 2-3-14 "Extras" Menu (2)

The "Extras" menu of "PLC Configuration" on the "Resources" tab is shown in figure 2-3-15.

Figure 2-3-15 "Extras" Menu (3)

- **Replace element**: Replace CPU Modules. This function is disabled in PLC.
- **Calculate addresses**: auto-calculation of module addresses.
- **Add configuration file**: No need to add any configuration file for PowerPro.
- **Standard configuration**: select this option to open a dialog as shown in figure 2-3-16, then click on "Yes" to restore default configuration or click on "No" to keep current configuration.

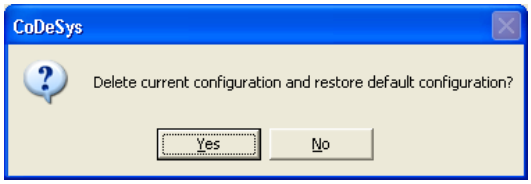


Figure 2-3-16 "Extras" Menu (4)

- **Properties**: If there is a "✓" in front of "Properties", base parameters, module parameters and communication parameters will pop up, otherwise no such information will be shown.

The “Extras” menu of “Watch- and Recipe Manager” on the “Resources” tab is shown in figure 2-3-17.

Figure 2-3-17 “Extras” Menu (5)

- **Monitoring active**: click the option, then a “✓” will appear in from of it, the watch list is activated to monitor variables in the list.
- **Write Recipe**: Write variable values.
- **Read Recipe**: Read current variable values.
- **Rename Watch List**: Change the name of watch list.
- **Save Watch List**: Save watch list with the extension*.wtc.
- **Load Watch List**: Load the renamed watch list with the extension*.wtc.

2.3.6 Online

"Online" menu contains a set of tools for program download and debug, as shown in figure 2-3-18.

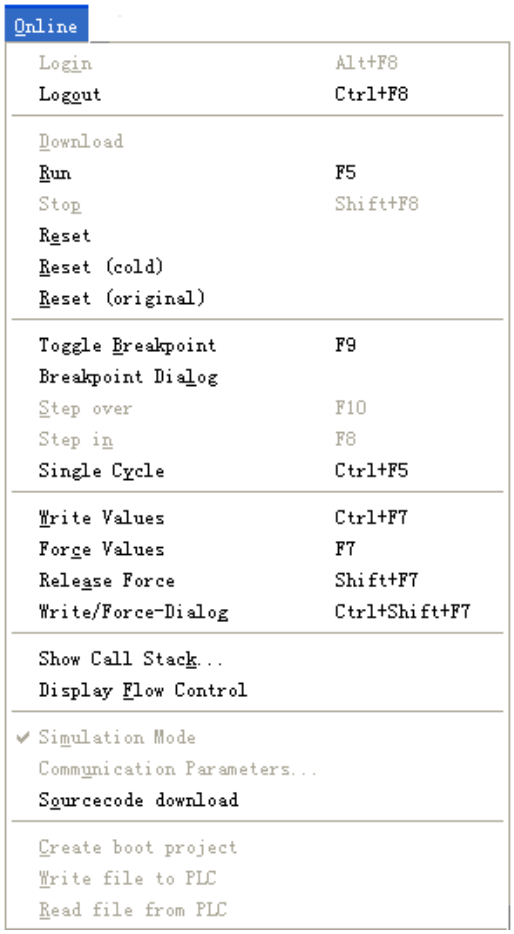


Figure 2-3-18 "Online" Menu

- **Online [O]/Login [I]**: Establish connection between PowerPro and PLC. When the program in PowerPro is consistent with that in PLC, the system enters Debug Mode automatically. If they are inconsistent, the user will be prompted to download the program. ***Refer to section 8.3.3 for details.***
- **Online [O]/Logout [X]**: Exit the Debug Mode and switch to Editing Mode.
- **Online [O]/Download [D]**: Load the compiled project in the PLC. It's valid only after the connection between PLC and PowerPro is established. ***Refer to section 8.3.3 for the differences between download and login.***
- **Online [O]/Run [R]**: Start the program and enter the Run Mode.
- **Online [O]/Stop [P]**: Stop the execution of the program.
- **Online [O]/Reset [E]**: Stop the execution of the program, and initialize the variables. Variables of type Retain keep their current values.
- **Online [O]/Reset (cold) [T]**: Stop the execution of the program, and initialize all the variables.
- **Online [O]/Reset (original) [O]**: Reset all variables to their initial values and erase the user program in PLC. Note that this is different from "Project"/"Clean all" which is to delete all the information from the last download and from the last compilation and re-build download files in the next compilation.
- **Online [O]/Toggle Breakpoint [B]**: Set or remove a breakpoint in present position. The program will stop at breakpoints, and display the code fragment in red background.

Use the commands “Online /Run”, “Online /Step over” or “Online /Step in” to continue running the program. ***Refer to section 8.4.6 for details.***

- **Online [O]/Breakpoint Dialog [L]**: Edit breakpoints throughout the entire project.
- **Online [O]/Step over S]**: This command causes a single step to be executed and program stops after the execution.
- **Online [O]/Step in [N]**: If the code at the present position is calling a function or a function block, the program will stop after the first instruction in the called POU. In other cases it is the same as “Online/ Step over”.
- Online [O]/Single Cycle [Y]: Execute a single PLC cycle and stop after this cycle. ***Refer to section 8.4.8 for details.***
- **Online [O]/Write Values [W]**: Set the variables to user defined values when debugging.
- **Online [O]/Force Values [C]**: Set the variables to user defined values. The variables being forced will be assigned forced values after each loop. This function remains active until the “Release Force” command is executed.
- **Online [O]/Release Force [A]**: End the forcing of variable values in the controller.
- **Online [O]/Write/Force-Dialog [G]**: One or more variables are set to user defined values and output to PLC. For “Online/ Write Values”, variables are set one time only and can be assigned values by other programs immediately. For “Online/ Force Values”, variables are set to forced values at the end of each loop until the “Release Force” is executed.
- **Online [O]/Show Call Stack [K]**: In the simulation mode, show a list of POUs in the call stack.
- Online [O]/Display Flow Control [F]: Toggle display of flow control.
- **Online [O]/Simulation Mode [M]**: If simulation mode is chosen, a check “√” will appear in front of the menu item. In the simulation mode, the user program runs on the local PC under the operating system. This mode is used to test the project. If the program is not in simulation mode, the program will run on the PLC.
- **Online [O]/Communication Parameters [U]**: Set the communication parameters between local PC and PLC modules.
- **Online [O]/Source code download [O]**: PLC does not support the function.
- **Online [O]/Create boot project [C]**: Set the project downloaded in PLC as a default project when starting the PLC. A boot project is a user program which is saved in flash of PLC and can be run after it is powered on. If the codes of boot project are different from the codes downloaded last time, a message as shown in figure 2-3-19 will appear: The current code is different from the code downloaded last time, continue? Generally Yes should be chosen to create a boot project. In offline mode a new *.ri file will be created. ***Refer to section 8.3.3 for the difference between “Online” “Create boot project”, “Login” and “Download”.***

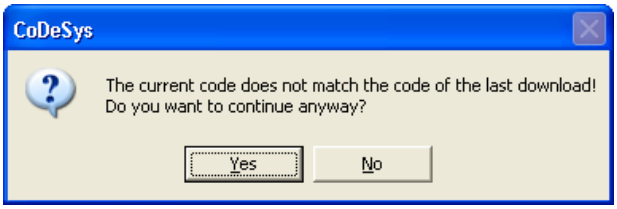


Figure 2-3-19 Create Boot Project

- **Online [O]/Write file to PLC [W]**: this function is disabled in PLC.
- **Online [O]/Read file from PLC [R]**: this function is disabled in PLC.

2.3.7 Window

"Window" menu is shown in figure 2-3-20. Its detailed functions are as follows:

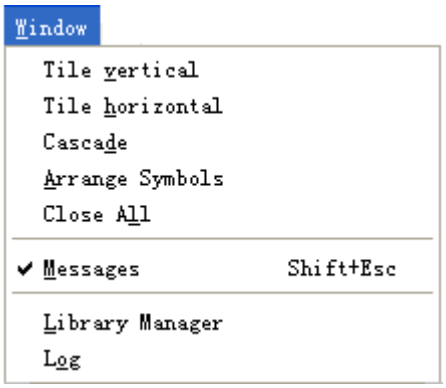


Figure 2-3-20 "Window" Menu

- **Window [W]/Tile Vertical [V]**: Arrange all the windows vertically in the work area so that they do not overlap and will fill the entire work area.
- **Window [W]/Tile Horizontal [H]**: Arrange all the windows horizontally in the work area so that they do not overlap and will fill the entire work area.
- **Window [W]/Cascade [D]**: Arrange all the windows in the work area in a cascading fashion, one behind another.
- **Window [W]/Arrange Symbols [A]**: Arrange all of the minimized windows in the work area into a row at the lower end of the work area.
- **Window [W]/Close All [L]**: Close all open windows in the work area.
- **Window [W]/Messages [M]**: Open or close the message window.
- **Window [W]/Library Manager [L]**: Open library manager window to add or delete library functions.
- **Window [W]/Log [O]**: Open log window.

2.3.8 Help

Help menu, as shown in figure 2-3-21, provides helps about the software. Its detailed functions are follows:

Figure 2-3-21 "Help" Menu

- **Help [H]/Contents [C]**: Open help topics and list the related items for a convenient and fast search.
- **Help [H]/Search [S]**: Open a dialog, find key words.
- **Help [H]/About [A]**: Information about software name and version.

2.4 SHORTCUT TOOL

Here is the introduction of shortcut buttons in the tool bar. Hove over a tool button for a short while and the name of the button will be shown as a tool-tip. When a command or a tool button turns in to gray, the function is disabled in the current window.

2.4.1 File Tool

 New: Create an empty project.

 Open: Open an existed project.

 Save: Save project.

2.4.2 Debug Tool

 Run: run a logged in program.

 Stop: Stop the program.

 Step Over: execute a single statement of the program and stop the program after the execution. When the “Step Over” command is applied on a function block or function, the program will then step over to execute the next statement.

 Breakpoint: Set or remove a breakpoint in present position.

 Login: enter the debug mode.

 Logout: logout debug mode and return to edit mode.

 Global Search: Search for a certain text in the entire project.

2.4.3 Edit Tool

 Cut: Remove the current selection from the editor to the clipboard.

 Copy: Copy the current selection from the editor to the clipboard.

 Paste: Paste the content of the clipboard onto the current position in the editor window.

 Find: Search for a certain text in the current editor window. Open a “Find” dialog,to enter a string in the field “Find what”. In addition, it also provides options such as “Match whole word only”, “Match case” and Search Up or Down from the current cursor position.

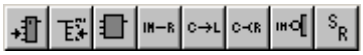
 Find Next: find the latest text of ‘Edit’ ‘Find’.

2.4.4 Programming Tool

The programming tools vary from different languages. According to different language types, the detailed functions of programming tool are as follows.

LD Language:  ,

Operators for Contact, Parallel Contact, Coil, Negate, Set/Reset from left to right.

FBD Language: 

Operators for Input, Output, Box, Assign, Jump, Return, Negate, Set/Reset from left to right.

SFC Language: 

Operators for Step-Transition (before), Step-Transition (after), Alternative Branch (right), Alternative Branch (left), Parallel Branch (right), Parallel Branch (left), Jump, Transition-Jump, Use IEC-Steps from left to right.

CFC Language: 

Operators for Input, Output, Box, Jump, Label, Return, Comment, Negation, Set/Reset, EN/ENO, Create micro, In pin, Out pin, Return to top level, Return to prior micro level, Jump into micro from left to right.

IL and ST languages do not provide programming tools like this. See chapter 9 for examples of programming tool applications.

2.5 OBJECT ORGANIZER

Object Organizer is the vertical window that locates on the left of the main window. It consists of four tabs “POUs”, “Data Types”, “Visualization” and “Resources” that contains all the base objects in a project.

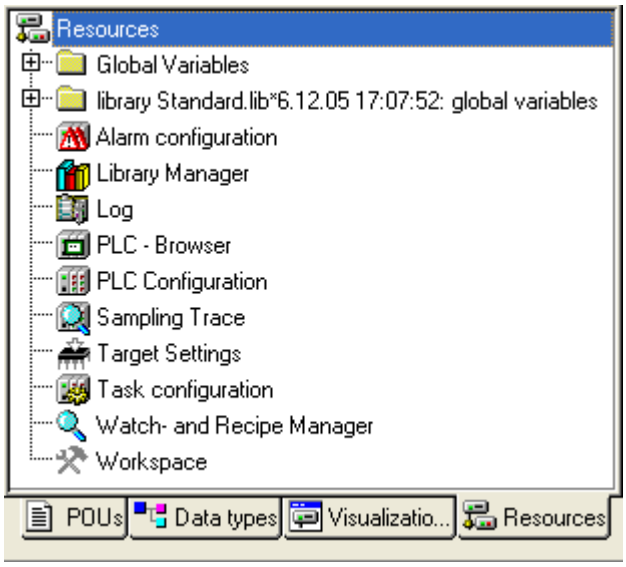


Figure 2-5-1 Object Organizer (1)

2.5.1 POUs

The function of “POUs” tab is to generate a POU (Program Organization Unit) in which a user program may be composed. Right click on the blank space of the “POUs” tab to open the menu shown in figure 2-5-2.

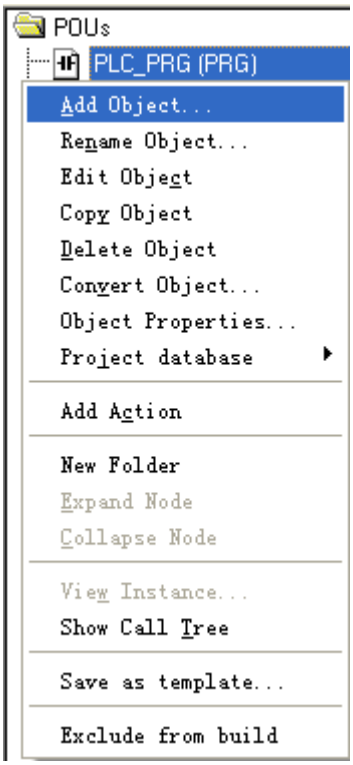


Figure 2-5-2 POUs

- **Add Object**: Create a new object in a POU, select the type of POU, language and name the new POU.
- **Rename Object**: Rename a selected object.
- **Edit Object**: Open the editor window of a selected object, or double click on the POUs name to open the editor window.
- **Copy Object**: copy a selected object and save it under a new name.
- **Delete Object**: Delete the selected POUs.
- **Convert Object**: Convert POUs from the languages SFC, CFC, ST, FBD, LD and IL into one of the three languages IL, FBD and LD.
- **Object Properties**: Assign access rights to different user groups.
- **Project database**: this function is disabled in PLC.
- **Add Action**: Generate an action in a selected program or function block. Enter the name of the new action and set the implementation language in the pop uo dialog. An Action represents a further implementation that may also be implemented using general function blocks written in other languages. An action belonging to a function block or program may be called like a function block. The syntax is: <Program_name>. <Action_name> or <Instance_name>. < Action_name >.
- **New Folder**: With the command a new folder is generated with the default name “New Folder”. If a folder has been selected, then the new one is created underneath it. In addition, you can give a new name to the folder through the command “Rename Object” in the context menu of a selected folder.
- **Expand Node**: With the “Expand Node”, expand “ ⊞” to “ ⊟”, and the subordinated objects appear.
- **Collapse Node**: With the “Collapse Node”, collapse “ ⊟” to “ ⊞”, and the folder is closed.
- **View Instance**: With this command it is possible to open and display the instance of the function block which is selected in the Object Organizer. You can also open the instance list through a double click on the function block or using the command “Project”/“View Instance”. Please note that if you want to view instances, you first have to log in. (The

project has been compiled with no errors and downloaded to the PLC with "Online"/"Login".)

- **Show Call Tree**: It's the same with "Project"/"Show Call Tree". With the command you open a window which shows the call tree of program, function, function block called in the object chosen. You can see the relationship between the object chosen and other objects clearly. Before this the project must have been compiled without any error.
- **Save as template**: The variables declared in current project and the changed settings in "Workspace" will be saved as a template automatically.
- **Exclude from build**: Activate the option and then the program, function block or function will be displayed in green color and not be included during compilation. If you want to cancel the option, select it again and the "√" in front of "Exclude from build" will be deleted and the program, function block or function changes from green to black.

2.5.2 Data Types

In "Data Types", along with standard data types, the users can define their own data types. Structures, enumeration types and references can be created. For example, users can add an object named "A" by right clicking "Data Types" tab, as shown in figure 2-5-3. ***Refer to section 4.6 for details.***

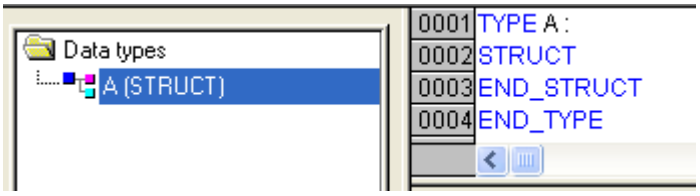


Figure 2-5-3 Data Types

2.5.3 Visualization

The "Visualization" option is used to display the project variables, as shown in figure 2-5-4. See chapter 11 for more details about "Visualization".

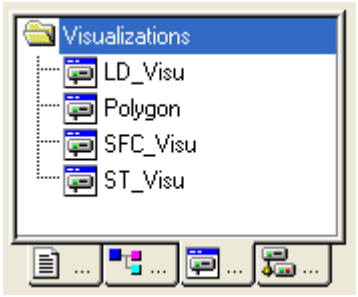


Figure 2-5-4 Visualization

2.5.4 Resources

In the “Resources” tab in the Object Organizer, there are Global Variables, PLC configuration and Workspace, as shown in figure 2-5-5.

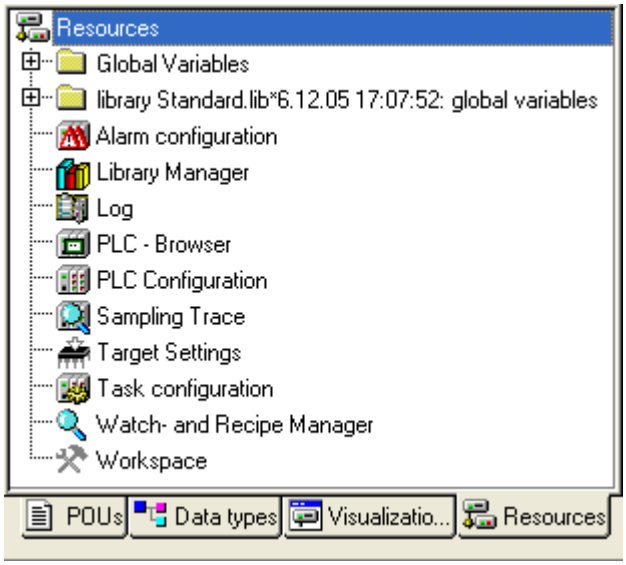


Figure 2-5-5 Resources

- **“Global Variables”**: Declare global variables used in the entire project.
- **“PLC Configuration”**: For configuring the hardware.
- **“Alarm configuration”**: Configure alarm parameters.
- **“Workspace”**: Configure the parameters of the current project, such as Load and Save, Directories, Passwords and so on.
- **“Watch- and Recipe Manager”**: In “Online Mode”, monitor the variable values of different POUs in a project. ***Refer to section 8.4.13 for detailed applications.***
- **“Library Manager”**: Consistent with the “Window” “Library Manager”, which handles all the libraries included in the project. ***Refer to section 7.4.4 for details.***
- **“Target Settings”**: Set the target platform and view memory allocation.
- **“Task configuration”**: Insert tasks and append program calls.

“Log”: Used to display project log information.

Chapter 3

Programming: A Quick Start

This chapter is to explain PowerPro programming by walking through sample programs, so that beginners will have a preliminary understanding of PowerPro programming and get familiar with the basic operations of PowerPro. It is suggested that this chapter should be read carefully if this is the first time you come across PowerPro.

The main function of the sample program in this chapter is to turn on and turn off a switch repeatedly in a certain time interval.

Before any programs can be written, the hardware configuration has to be determined. For this POU, not many I/O points are needed. Hence one CPU module will be sufficient. Here we have chosen the CPU module LM3107-CAR with 24 I/Os.

3.1 HARDWARE CONNECTION

3.1.1 Equipment needed

- A PC installed with PowerPro and RS232 serial port
- A CPU module LM3107
- A 220V AC power line
- A special programming cable for RS232 communication between PC and CPU module

3.1.2 Select the CPU Module

Select the LM3107 module and connect the power line as shown in figure 3-1-1. Note that when the power line is connected, put on the terminal cover to avoid unnecessary human injury or equipment damage.

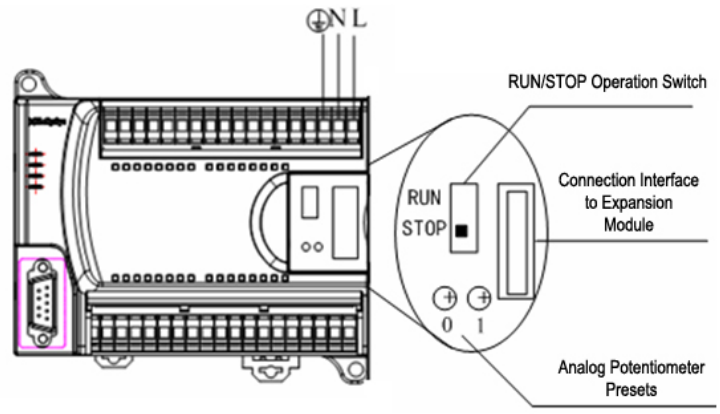


Figure 3-1-1 Connect Power Line

Do not switch on the power immediately after the power line is connected. The power should be switched on only after all the cables are confirmed to be connected correctly. Ensure that the RUN or STOP indicators on CPU board are on, so that the PLC can run reliably.

LM CPU modules have two running modes, set by the RUN/STOP Switch. See table 3-1-1 for details.

When programs cannot be downloaded to CPU module in RUN mode, the CPU has to be switched to STOP mode for the download to succeed.

Switch	Description
RUN	CPU is running and executing user program.
STOP	CPU is not executing user program and can download to PLC now.

Table 3-1-1 Setting description of "RUN/STOP Switch"

3.1.3 Setup PC Communication

Connect the CPU module to the PC through a programming cable to build a data transmission channel, as shown in figure 3-1-2.

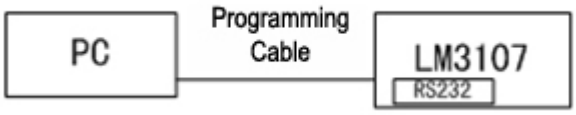


Figure 3-1-2 Connect Programming Cable

Note:
Please connect the communication programming cable before powering on the PLC to avoid damaging the PLC.

3.2 STARTING UP

Before running PowerPro for the first time, target installation has to be performed via the Install Target program. The installation steps have been mentioned in section 1.3.

In the "**Programs**" -> "**HollySys PowerPro ENG**" menu group from the Windows Start Menu, click "**PowerPro ENG**" to startup PowerPro, as shown in figure 3-2-1.

Figure 3-2-1 Startup PowerPro Software

The interface of PowerPro is as shown in figure 3-2-2.

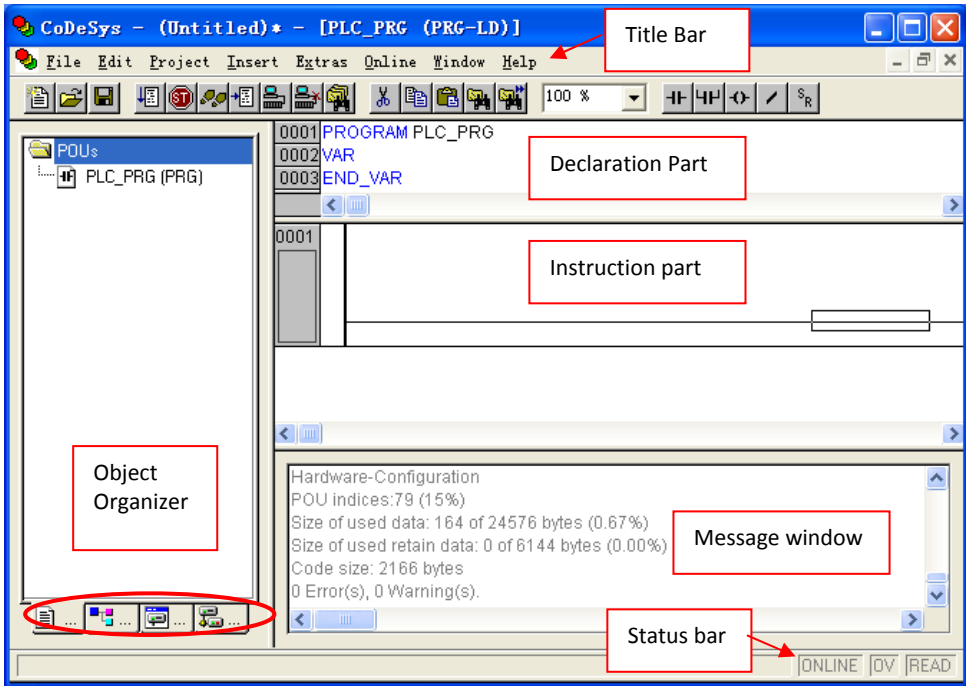


Figure 3-2-2 Interface of PowerPro

3.3 NEW POU

3.3.1 Target settings

- Click “File”/“New” in the main window or the button“ ” in tool bar, and a dialog box titled “Target Settings” will appear. “Target” means the memory space in PLC, and “Target Settings” means configurations according to the memory space of the selected PLC.
- Select “HOLLYSYS-LEC G3 CPU Extend” in the “Configuration” box. This target is for CPU modules with memory space of 120KB. Click “OK”, as shown in figure 3-3-1. Select “HOLLYSYS-LEC G3 CPU” if the memory space of the CPU module is 28KB. If you are not sure about the size of memory space, see appendix. Select “None” if you want to write new library instructions. ***Refer to section 7.4.5 for details about making libraries.***

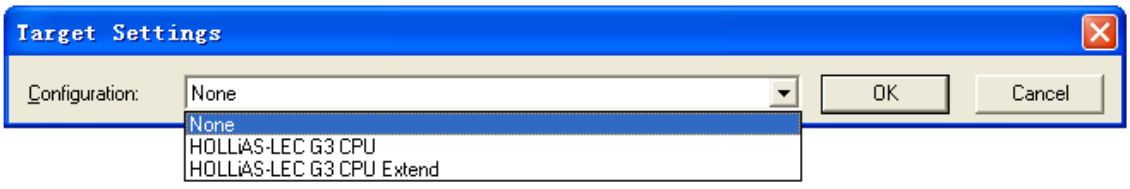


Figure 3-3-1 Target Settings

- After that, the window titled “Target Settings” will appear. The default settings will suffice for most application requirements. Click “OK”, as shown in figure 3-3-2.

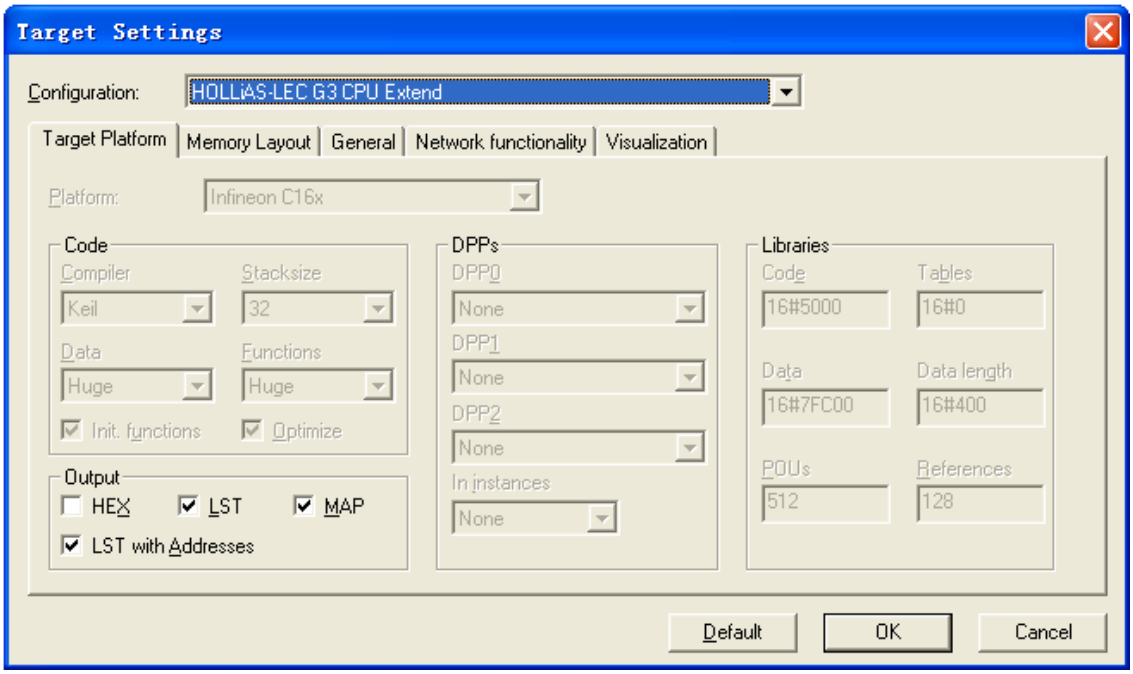


Figure 3-3-2 "Target Settings" Window

Refer to section 7.1 about Target Settings.

3.3.2 New POU

- POU stands for Program Organization Unit, and POUs are basic elements that form a project. See chapter 5 for details about POUs.
- Here the "LD" language is used as an example. Select "LD" in the "Language of the POU" option, as shown in figure 3-3-3. See chapter 9 for detailed explanations for other languages.

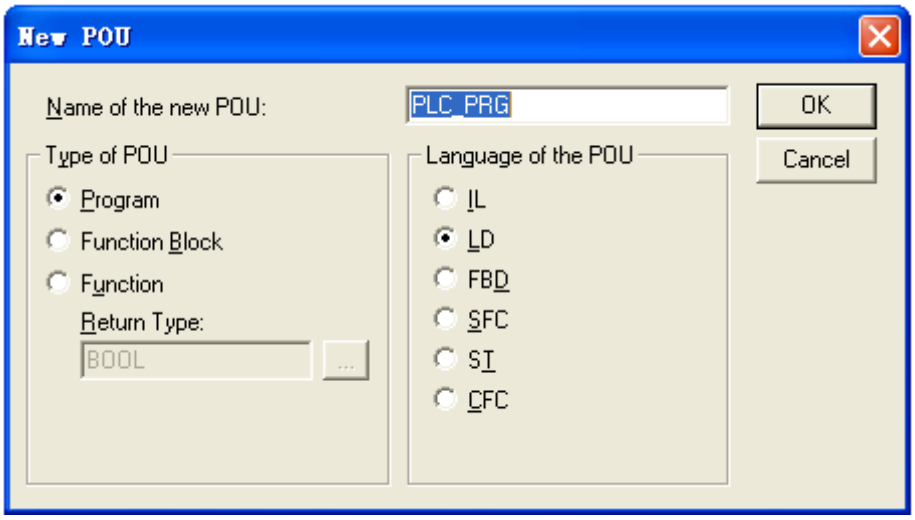


Figure 3-3-3 New POU

- Select Program in the "Type of POU" option, PLC_PRG is the default name of the new POU. *Refer to section 5.1.3 for details about PLC_PRG.*
- Click "OK" and the window as shown in figure 3-3-4 will appear.

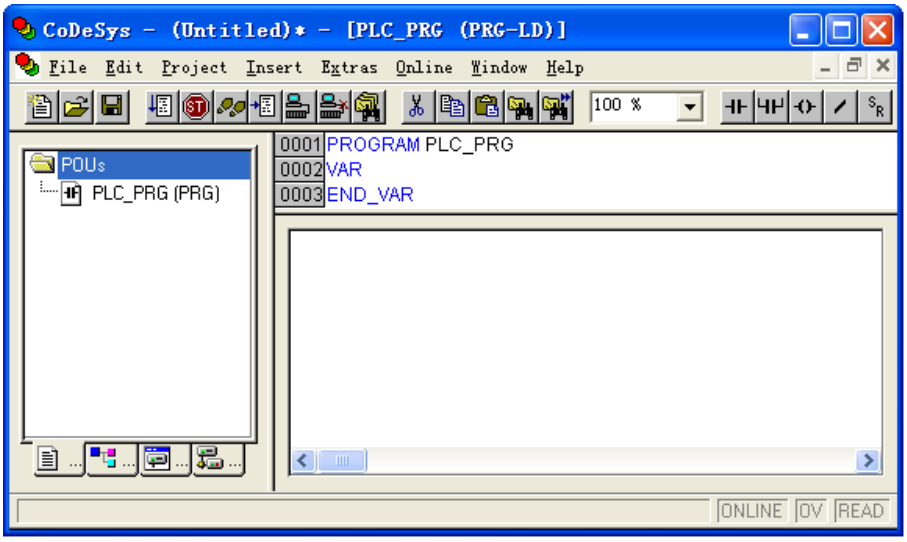


Figure 3-3-4 Workspace

3.4 PLC CONFIGURATION

PLC configuration is needed after building a new POU. It's suggested to complete PLC configuration before programming to avoid addressing errors.

Double-click "PLC Configuration" in the "Resources" tab, right click on "PLC Configuration", and select "LM3107" from "**Append Subelement**" menu item, as shown in figure 3-4-1.

Refer to section 7.3 for detailed descriptions about PLC configuration.

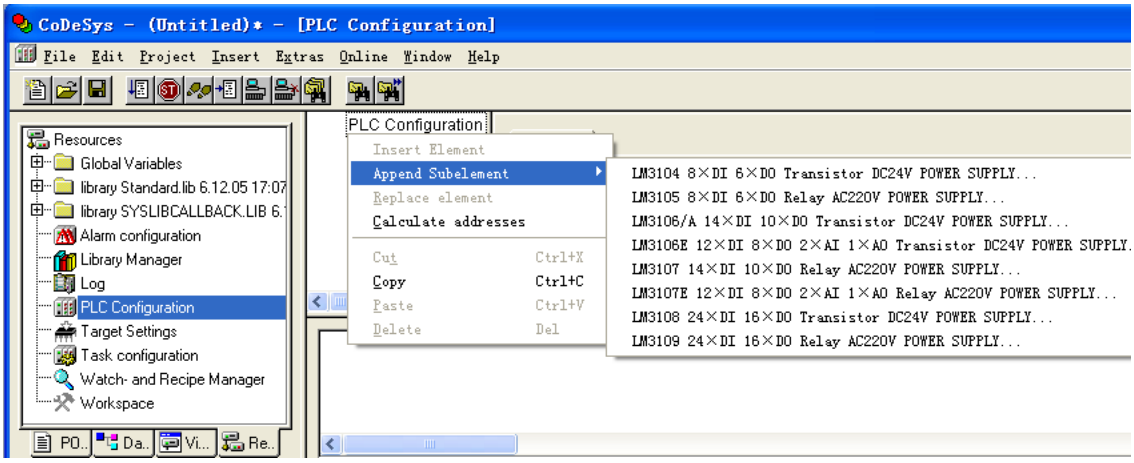


Figure 3-4-1 PLC Configuration

3.5 SET COMMUNICATION PARAMETERS

- Select "Communication Parameters" in the "Online" menu and a dialog box of communication parameters will appear, as shown in figure 3-5-1.

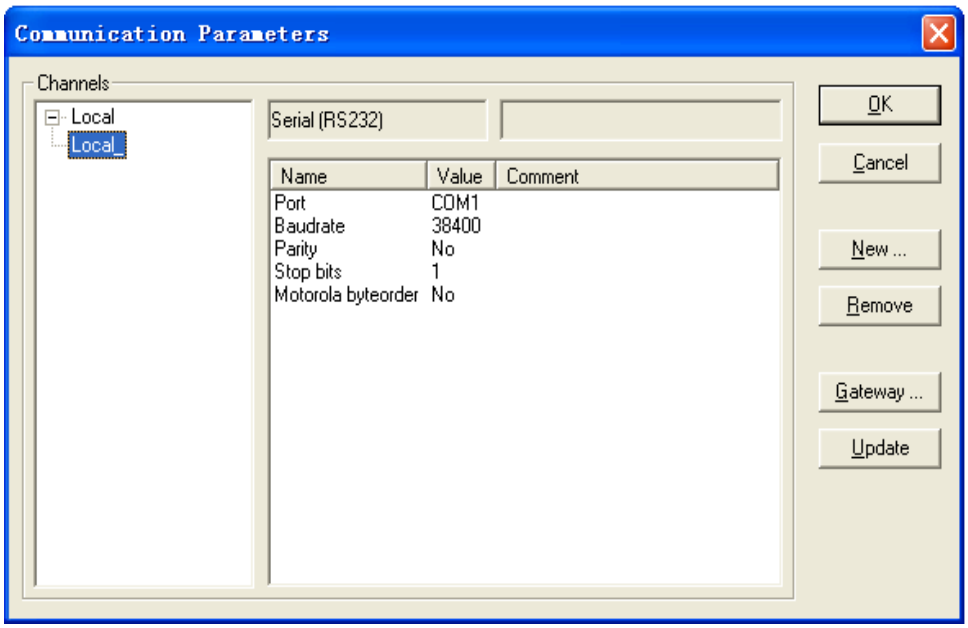


Figure 3-5-1 "Communication Parameters" Dialog

- Select "New" to add a new channel and a dialog will appear as shown in figure 3-5-2.

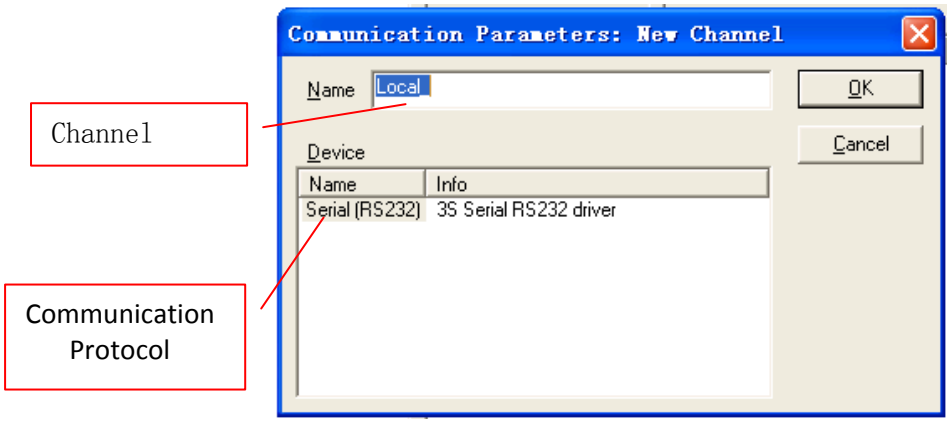


Figure 3-5-2 New Channel

- The default name of the channel is "Local_" and the default communication protocol is RS232. Return to dialog of communication parameters by clicking "OK", as shown in figure 3-5-3. Click "OK" and the connection between the PC and the CPU module is completed.

Note:
To change the port or baud-rate, double click on their field "Value".

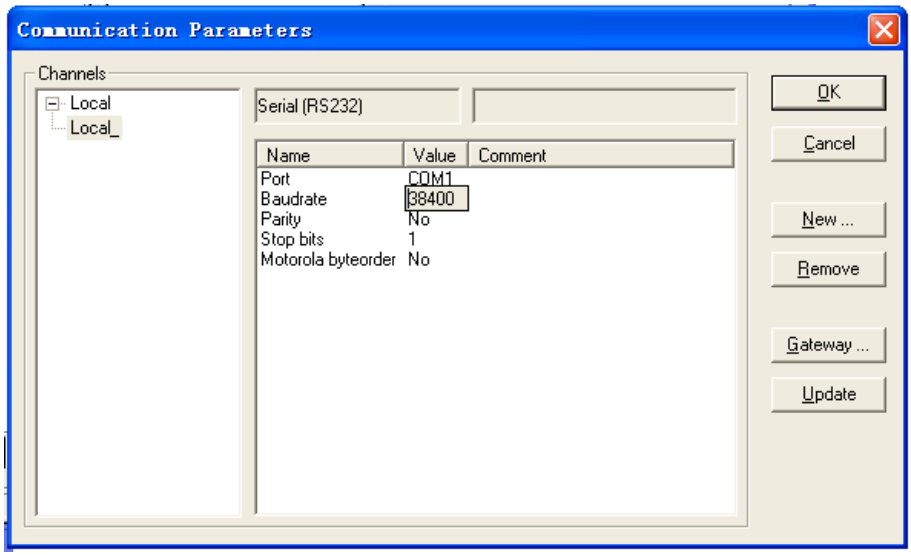


Figure 3-5-3 Baud-rate setting

3.6 PROGRAMMING

After PLC configuration, programming may be started. The following is a simple timer application to generate a “one second on and two seconds off" pulse signal.

- Click the “Contact” button “ ” on the tool bar, and then a contact “ ” will appear in the network 0001, as shown in the figure 3-6-1.

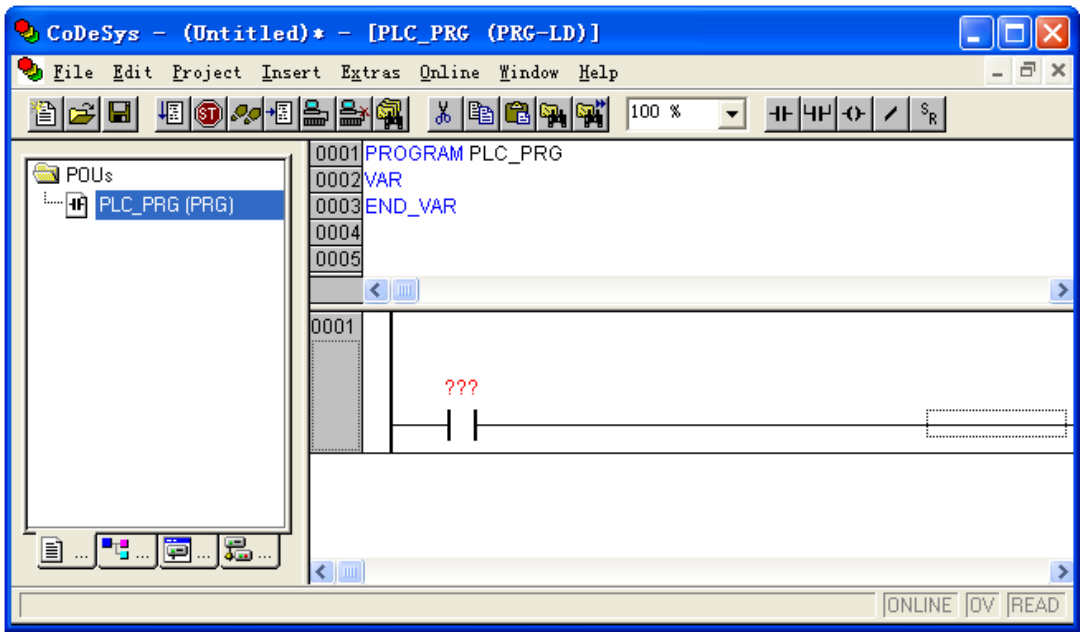


Figure 3-6-1 Programming Step (1)

- The default text of contact is “???”. Click the text and enter “%IX0.0”, as shown in figure 3-6-2. %IX0.0 indicates the first input of PLC. ***Refer to section 4.2 for data addresses.***

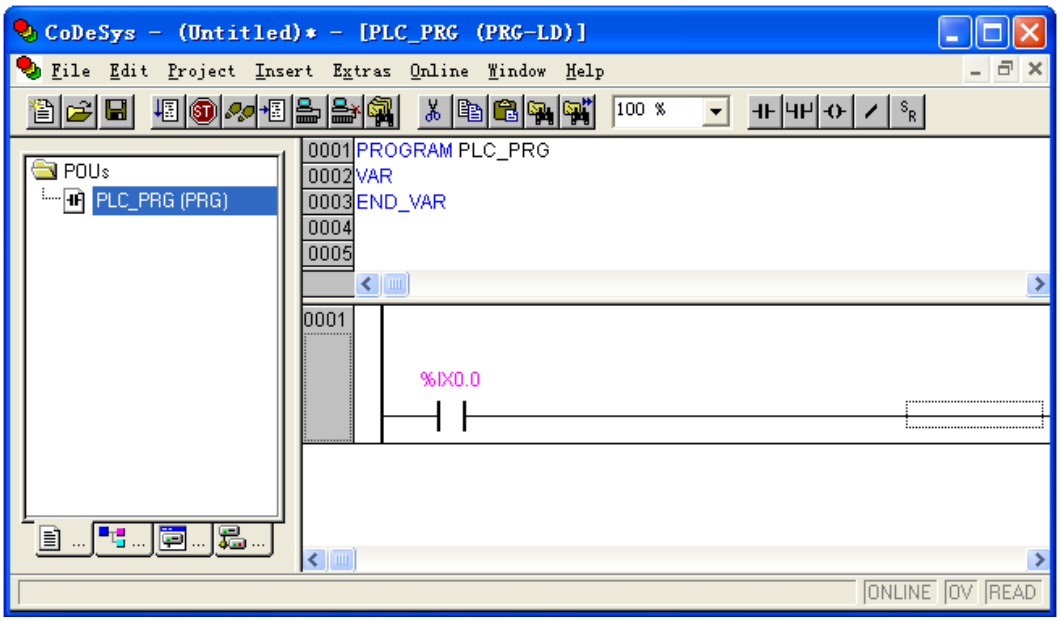


Figure 3-6-2 Programming Step (2)

- Right click after contact "%IX0.0", then select "Function block", as shown in figure 3-6-3.

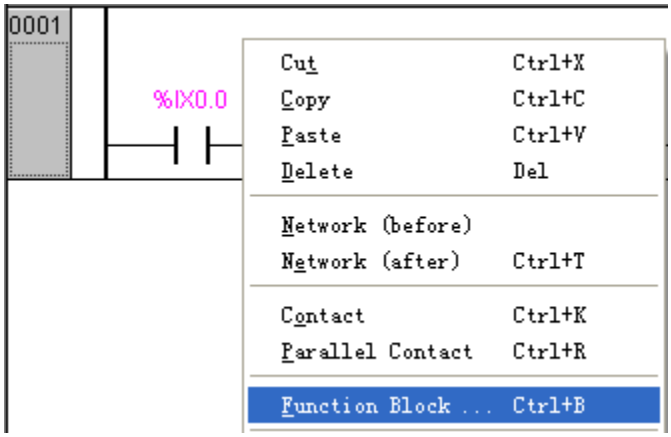


Figure 3-6-3 Programming Steps (3)

- Click on "Function block", a dialog will appear as shown in figure 3-6-4, then select "TON (FB) ", a command of timer connecting in power delay.

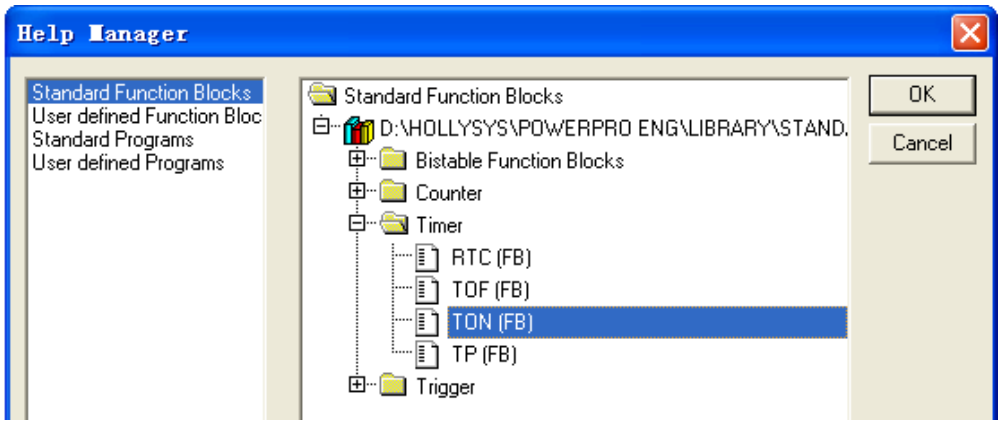
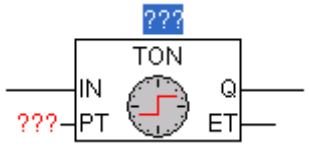


Figure 3-6-4 Programming Step (4)

- Double-click “TON (FB) ”, key in T1 at the cursor position of



” and Enter, a dialog as shown in figure 3-6-5 will Select the default type as “TON”, and then click “OK”. T1 is an identifier of TON. A function block must be identified by an identifier. ***Refer to section 5.3.2 and section 7.4.3 for details.***

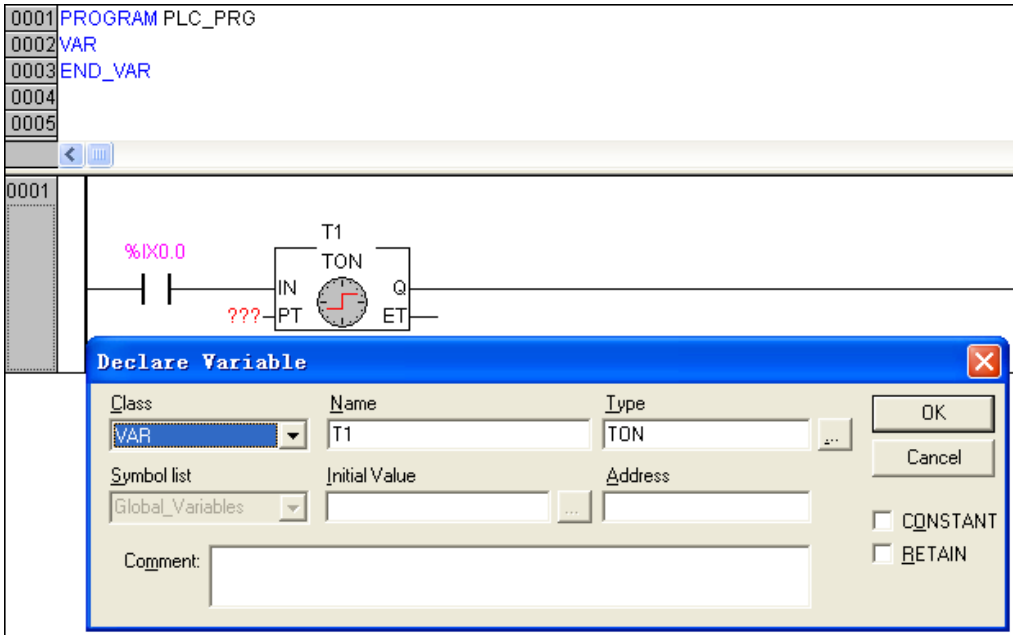


Figure 3-6-51 Programming Step (5)

- A power delay timer “TON” named T1 has been added to the Ladder Diagram. Enter “T#1S” in the field “???” to indicate 1s delay. Enter time variable “ET” in the field “ET”, shown in figure 3-6-6. Then click “ok”.
- PT is an input time parameter that can be set either as a time constant or a time variable. ***Refer to section 4.3 and section 4.4 for details.***
- ET indicates time passed after the timer setting, i.e. current time. Define a time variable in the field “ET” to observe the time passed.

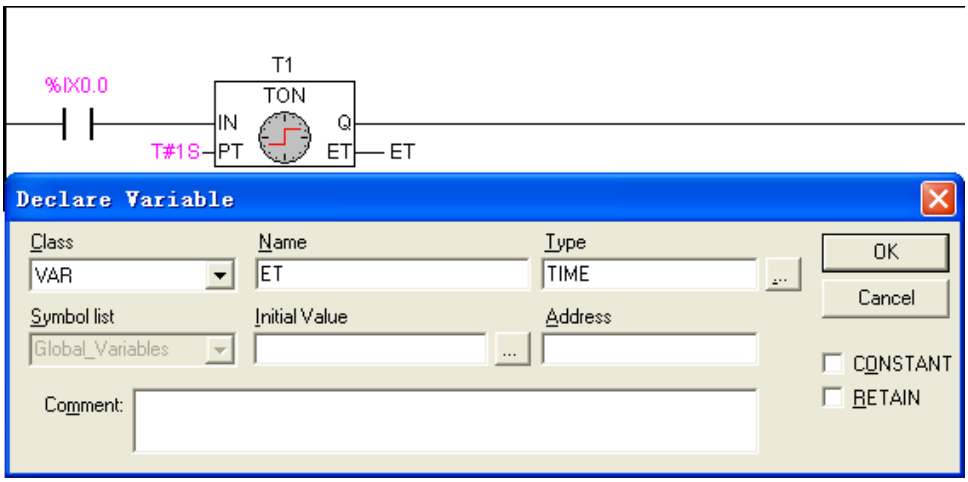


Figure 3-6-6 Programming Step (6)

- Put the cursor after “T1”, then click the coil button “  ” on tool bar, a coil “  ” will appear at the cursor position, as shown in figure 3-6-7.

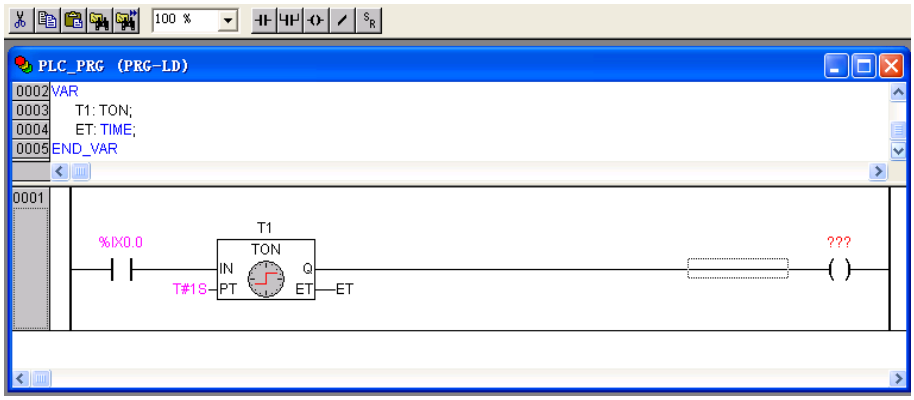


Figure 3-6-7 Programming Step (7)

- Enter “M” in field “???”, and select the “Type” as BOOL, as shown in figure 3-6-8, then click “ok”.
- M is a middle variable. ***Refer to section 4.4 for details.***

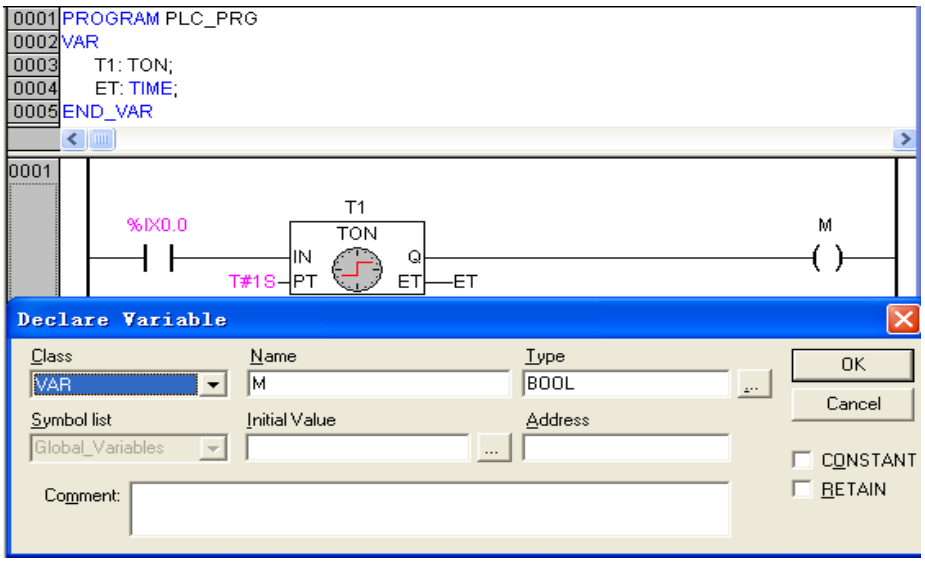


Figure 3-6-8 Programming Step (8)

- As shown in figure 3-6-9, right click on the workspace, then select “Network (after)” to start programming in network 0002.

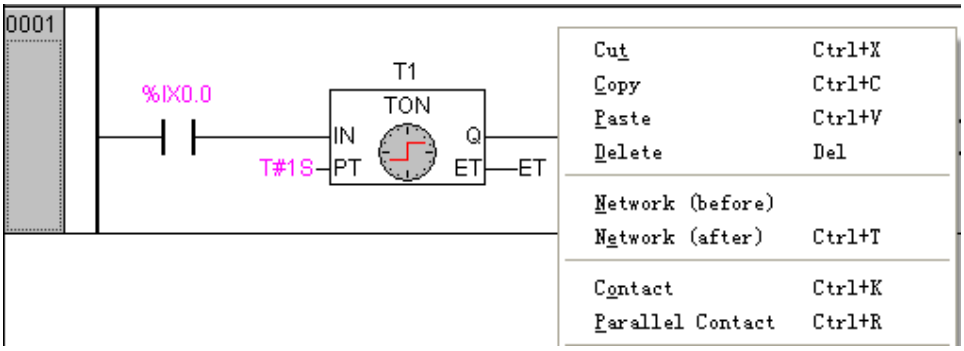


Figure 3-6-9 Programming Step (9)

- The ladder diagram in 0002 is shown in figure 3-6-10, and its programming steps are the same with 0001.

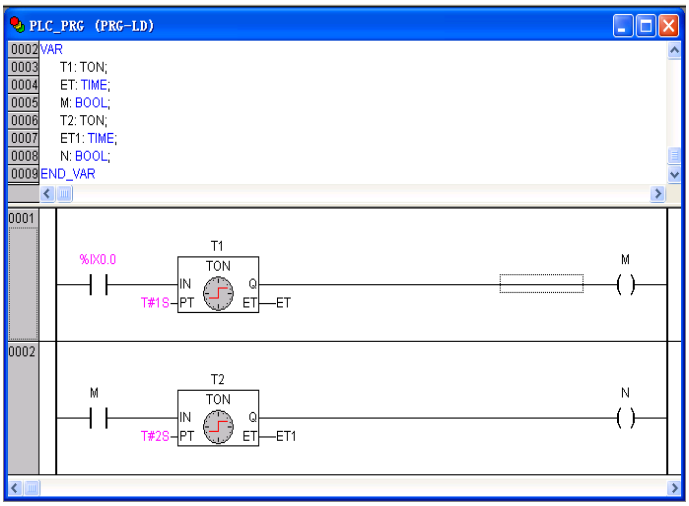


Figure 3-6-10 Programming Step (10)

- In order to achieve the regular on/off switch, a switch “N” shall be added before “T1”. Select “T1”, click the “Contact” button on the tool bar or right click to select “Contact”, as shown in figure 3-6-11.

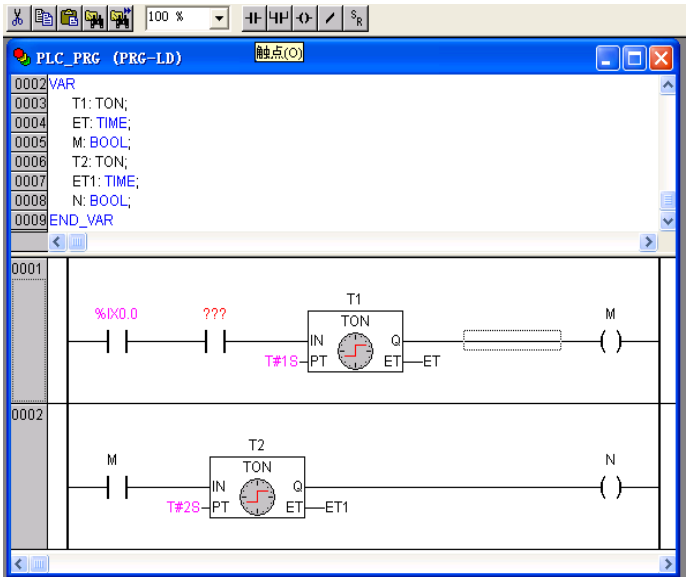


Figure 3-6-11 Programming Step (11)

- Enter “N” in the field “???”, and click “Negate” on the tool bar, as shown in figure 3-6-12.

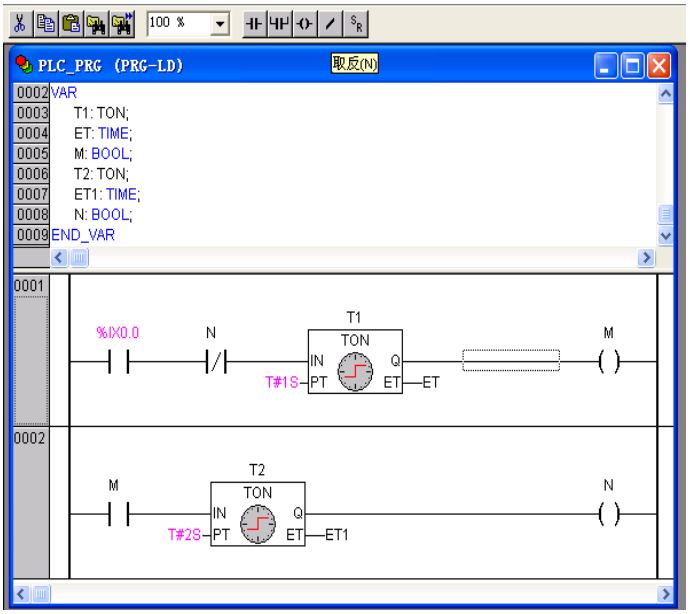


Figure 3-6-12 Programming Step (12)

- Add network 0003 to output the value of M from %QX0.0. Add a contact “M” in network 0003 and a coil after it. Click the “Coil” button on the tool bar and name the coil “%QX0.0”, as shown in figure 3-6-13.

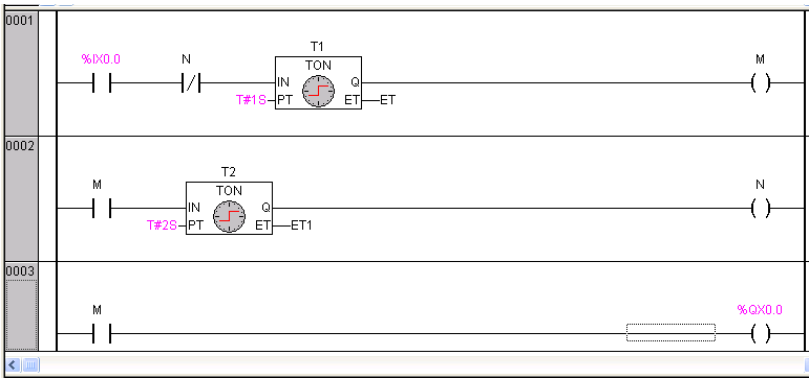


Figure 3-6-13 Programming Step (13)

- Comment may be added for a better understanding of the POUs. For example, right click on network 0001, and then select “Comment”, as shown in figure 3-6-14.

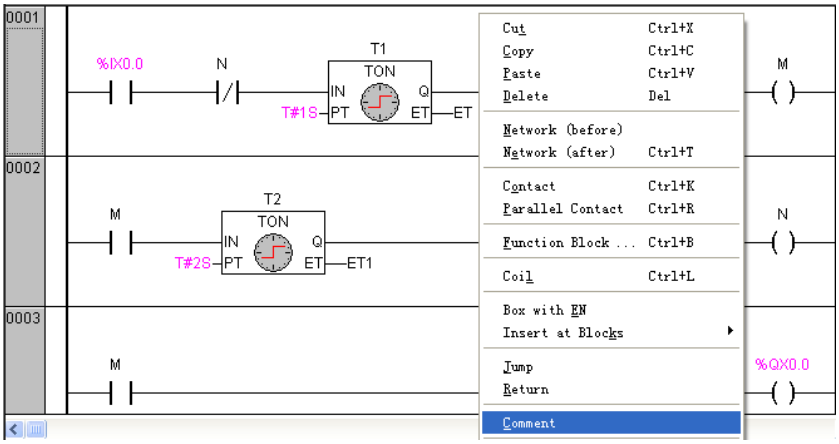


Figure 3-6-14 Programming Step (14)

- Double click “Comment” to enter the detailed comments, as shown in figure 3-6-15.

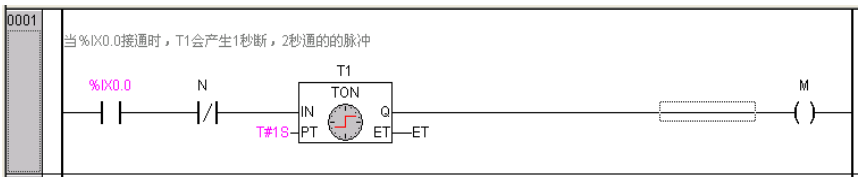


Figure 3-6-15 Programming Step (15)

- The comment is completed, as shown in figure 3-6-16.

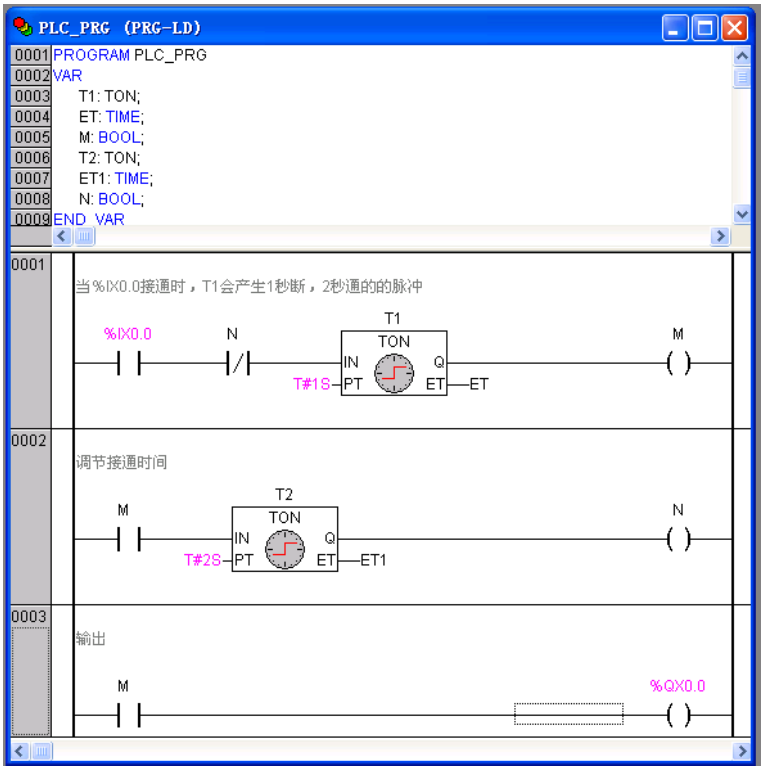


Figure 3-6-16 Programming Step (16)

3.7 BUILD

After programming, the project shall be built.. Click “Project” “Rebuild all” to build the program, as shown in figure 3-7-1.

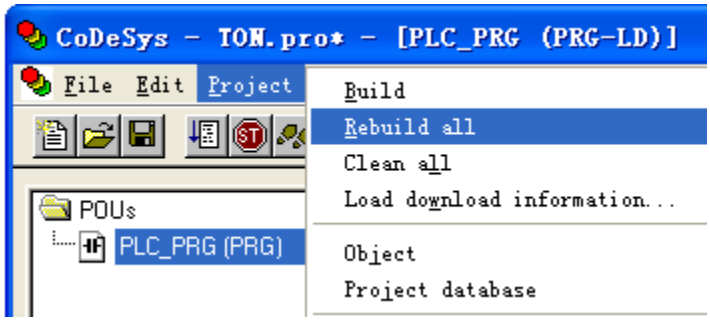


Figure 3-7-1 Build (1)

The information is displayed in Message Window, as shown in figure 3-7-2

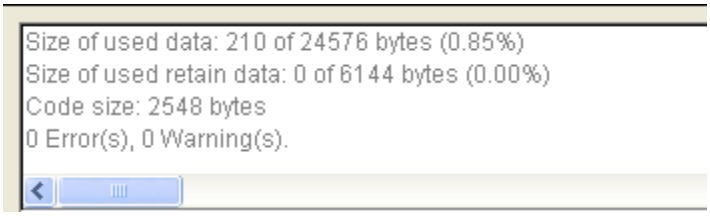


Figure 3-7-2 Build (2)

Click “Project”/ “Check”/“Unused Variables”, if there are no unused variables, a message will appear in Message Window: “No unused variables found”, as shown in figure 3-7-3.

Figure 3-7-3 Build (3)

If there is a unused variable “a” in the program, a message will appear in the Message Window: “PLC_PRG(9): a”, as shown in figure 3-7-4.

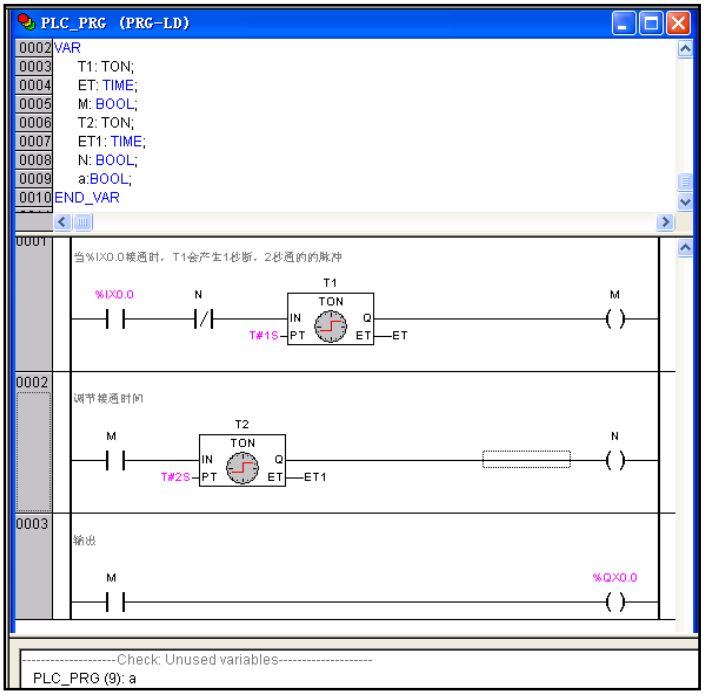


Figure 3-7-4 Build (4)

The use of is “Check”/“Unused Variables” command is recommended after every building of the program. Unused variables shall be deleted to improve the run of the program.

In addition, the unused variables can be checked automatically. Open “Project”/“Build”/“Auto Check” on the “Resources” tab, then select “Unused variables” to start an auto-check.

Refer to section 8.1 for details of building. Refer to section 8.2.3 for the checking of unused variables.

After the building is finished, execute “Online”/“Login” to log in debug mode. The debugging contains online debugging and simulation debugging. The program running under these tow debug mode are introduced in the following sections.

3.8 ONLINE DEBUGGING

Online debugging will download the program to CPU modules. Download all the compiled files into CPU modules and reset the CPU modules at the same time and all the variables will be restored to their initial status.

Select “Online”/“Login” to establish the connection between local PC and CPU modules, and a message will appear as shown in figure 3-8-1.

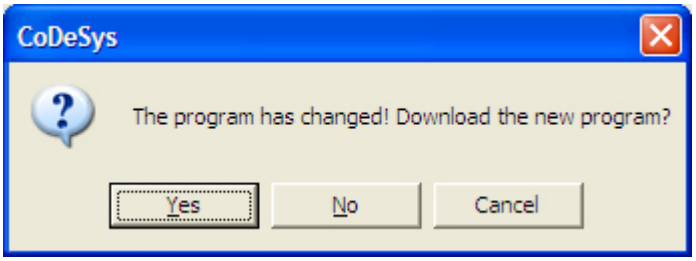


Figure 3-8-1 Download Message

Click “Yes” to download the program to CPU modules. When the message shown in figure 3-8-2 for creating boot project appears click “Yes” to complete the download. Restart the PLC to run the downloaded project.

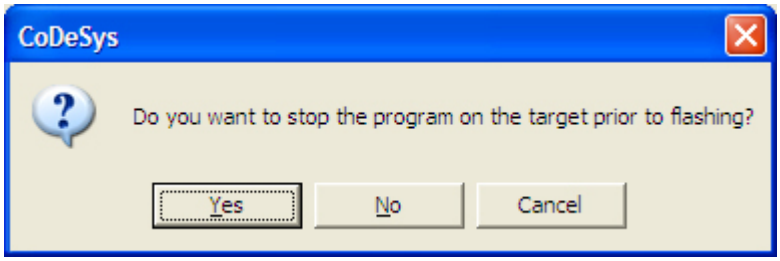


Figure 3-8-2 Message for Creating Boot Project

The program will not automatically run after download. The PLC shall be manually set before running of the program. Use the command “Online”/“Run” or simply press “F5” key to run a program, as shown in figure 3-8-3.

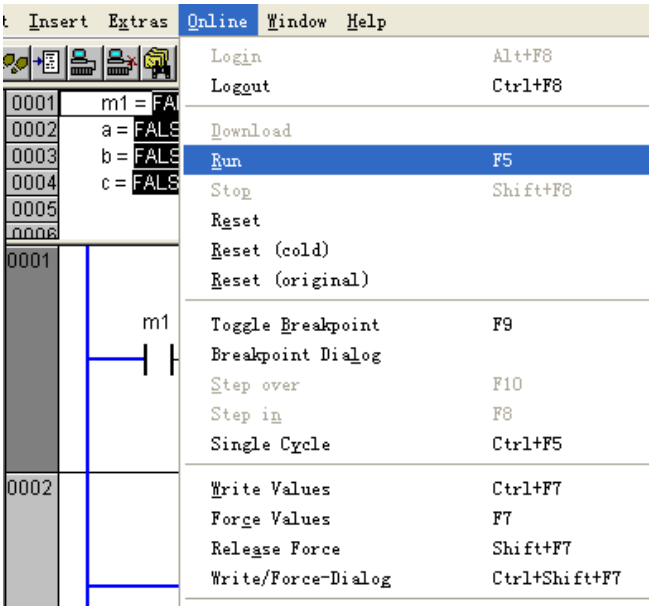


Figure 3-8-3 Run the Program

Double click “%IX0.0” and press the “F7” key to force values, then %IX0.0 is closed and the program starts to run, as shown in figure 3-8-4. The running result shows that the channel %QX0.0 outputs a “1s off 2s on” pulse signal.

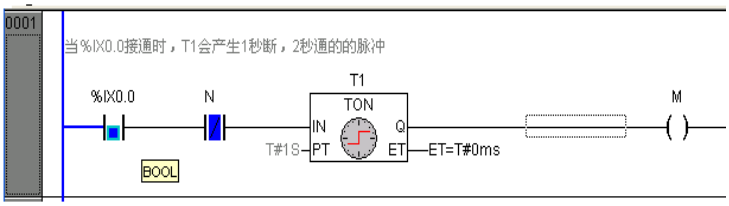


Figure 3-8-4 Program Running in Forced Values

The program running process is as follows. As shown in figure 3-8-5, when the program first started and its running time t<1s, %QX0.0 is not closed and the channel light %QX0.0 is off.,

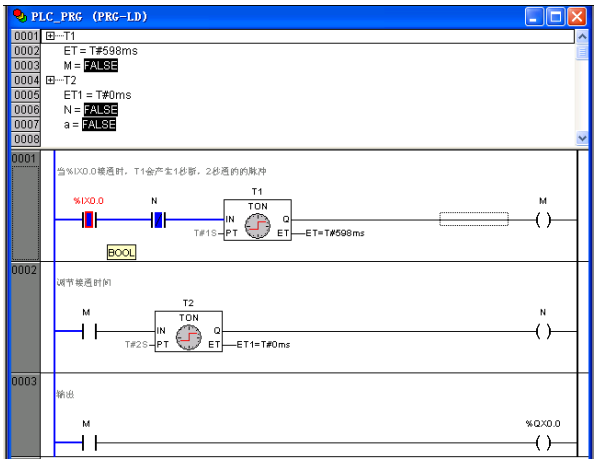


Figure 3-8-5 Program Running Process (1)

When the running time s<t<3s, %QX0.0 is closed and the first channel light %QX0.0 on the PLC is on, as shown in figure 3-8-6.

When the running time 3s<t<4s, contact “M” is off and the channel light %QX0.0 on PLC is off. Repeat like this, and the channel light %QX0.0 will be on at 1s, last for 2s and then off and on again after 1s.

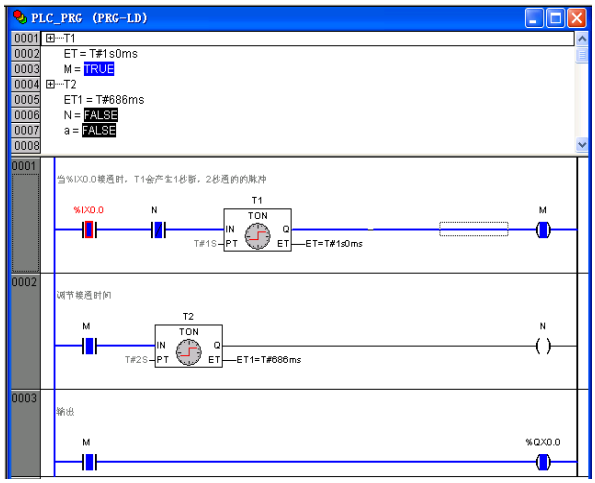


Figure 3-8-6 Program Running Process (2)

3.9 SIMULATION MODE

The last section shows how the program is performed in the online mode. When there is no CPU modules connected to PLC, the user program may run on the local PC in simulation. This is called the simulation mode. Open the “Online” menu from menu bar, select “Simulation Mode” to enter the simulation mode, as shown in figure 3-9-1. If the “Online”/“Simulation Mode” is selected (with a “√”), the program will run under the simulation mode after log in

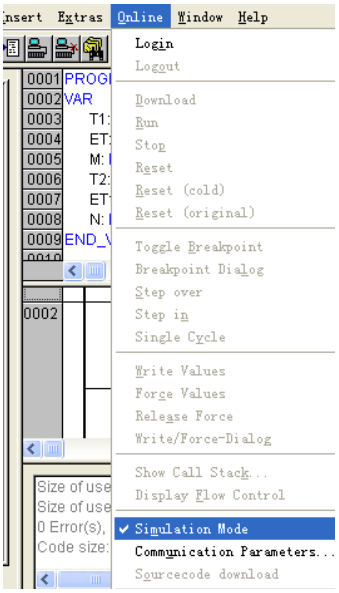


Figure 3-9-1 Simulation Mode

Double click “%IX0.0”, Press “CTRL+F7” to write values or press “F7” to force values, the “%IX0.0” is closed. Press “F5” key to run the program, as shown in figure 3-9-2. The program running is the same with that in online mode.

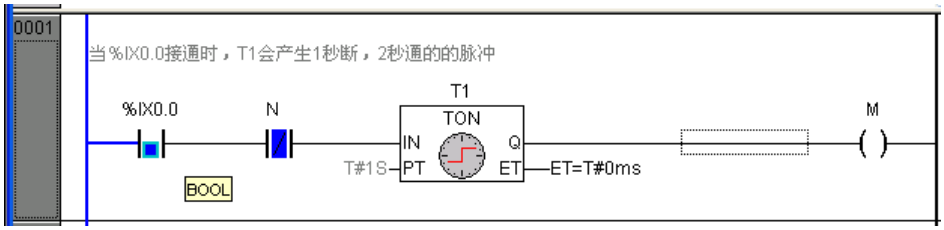


Figure 3-9-2 Program Running in Simulation Mode

Refer to section 8.4 for more details of online debugging and simulation debugging.

Chapter 4

Data Storage and Variables

Before programming, we need to understand the data management of LM Micro series PLCs. For better programming, this chapter gives an introduction on concepts such as storage assignment, data addressing and variable usage of LM Micro series PLCs.

There are five types of data storage in LM Micro series PLCs, namely,

- I (input)
- Q (output)
- M (memory)
- N
- R (retaining in power failure)

Section 4.1 is devoted on the features and usages of respective storage types.

Section 4.2 explains the addressing methods and storage rules.

The I (input), Q (output) and M (memory) data storage of the LM Micro series PLCs are addressable.

Similar to the advanced programming languages, the concepts of constants and variables are also applied in LM Micro series PLCs. The so-called constants are the variables of unchangeable values. ***Section 4.3 gives a detailed explanation of the classification and usages of constants.***

According to the definition promoted by IEC61131-3 standard, variables are changeable values in program running. They are used to initialize, store and handle user data. Each variable has a certain data type. The addresses of variable may either be assigned to I, Q or M storage by users or be automatically assigned by the system. ***Section 4.4 gives the detailed introduction of types and usages of the variables.***

LM-series PLC has a powerful data management functions. ***Section 4.5 and 4.6 explain how to handle arrays and how to define data types.***

4.1 STORAGE ASSIGNMENT

The storage of the LM Micro series PLCs, e.g. their CPU memory, is classified into different areas that have different features and usages.

The storage is classified into the following types:

4.1.1 Input Storage (I)

The mapping area of inputs: Digital and analogue inputs of CPU and expansion modules occupy the addresses of input storage. In addition, some special functions, such as Ethernet communication and DP communication, also occupy the addresses of input storage. I storage can store maximum 512 bytes.

Input storage can be accessed by addressing according to bit, byte, word (two bytes) and double word (four bytes). Refer to section 4.2 for detailed addressing method.

The data in input storage is read-only and has no retaining in power failure. In simulations, the addresses of input storage may be entered or forced. However, in online debugging, the addresses of input storage can only be forced. PowerPro provides two methods, 'Write Values' and 'Force Values', to change the variable values in debugging, please refer to section 8.4.9 and 8.4.10 for details.

4.1.2 Output Storage (Q)

The mapping area of output: Digital and analogue outputs of CPU and expansion modules occupy the addresses of output storage. In addition, some special functions such as Ethernet communication and DP communication, also occupy the addresses of output storage. Q storage can store maximum 512 bytes.

Output storage can be accessed by addressing according to bit, byte, word (two bytes) and double word (four bytes). Refer to section 4.2 for detailed addressing method.

The data in output storage may be read and written and has no retaining in power failure. In simulations or online debugging the addresses of output storage may be entered or forced.

4.1.3 M Storage

M storage is a middle register of PLC that is used to store and manage data or status generated in the middle processes. Both bit data and word data can be stored in M storage.

M storage can be accessed by addressing according to bit, byte, word (two bytes) and double word (four bytes). The LM-series PLCs adopt 8KB M storage that has a range of MB0~MB8191 in bytes. Refer to section 4.2 for detailed access method.

Same as those in Q storage, the data addresses in M storage can be read and written while their vales may be entered or forced.

Part of the M storage addresses (MB300~MB799) can be retained in power failure while other parts do not have this function.

Furthermore, the first 100 bytes in M storage, i.e. MB0~MB99, are used as a self-diagnosis area. The data in these addresses are readable but not writable, so that users are suggested to start programming from address MB100.

4.1.4 N Storage

N storage is also a middle register of PLC that is used to store and manage data or status generated in the middle processes. The difference between the M and N storage is that the N storage can only be accessed and called by variables. Although variables have been mentioned in former chapters, their detailed usages are explained in section 4.4.

The variable addresses in N storage are automatically assigned by system that cannot be defined by users. Besides data types of bit, byte, word (two bytes), double word (four

bytes), the variables of N storage also adopts other types such as REAL, TIME, INT, etc. In addition to the data variables, the defined function block variables are also stored in N storage. See section 5.1 for the concept of function blocks.

N storage provides 24KB space for variables. These variables stored in N can be read and written while their values may be entered or forced. The data in N storage cannot be retained in power failure.

4.1.5 R Storage

R storage is the retaining area in power failure that adopts the same calling method of variables access as the N storage. The addresses in R storage, as those of the N storage, are automatically assigned by the system and cannot be defined by users.

The size of R storage is 6KB. The variables stored in R can be read and written while their values may be entered or forced.

In the declaration of one variable, if the option “RETAIN” is not selected or the variable is directly declared between the keywords VAR and END_VAR, then the variable will be stored in the N storage; if the option “RETAIN” is selected or it is directly declared between the keywords VAR RETAIN and END_VAR, then it will be stored in R area and its last may be retained in power-failure. Refer to section 4.4.6 for the declaration of RETAIN variables.

Note:
LM Micro Series PLC provides two ways to retain data in case of power failures. One is the address method that selects the addresses between MB300 ~ MB799. The other, is the variable method that declares a RETAIN variable.

4.2 ADDRESSING

4.2.1 Address Mapping

I, Q and M storage of LM Micro series PLC are accessed by addressing. Each storage area has a unique and definite address. Users may read and set values in the storage by addressing, i.e. by direct calling of the storage address.

Before explaining the accessing rules, we need to understand the storage format. Since the storage formats of I, Q and M are the same, here we may just take M storage for an example, as shown in figure 4-2-1.

All the direct addressing storage areas in LM Micro series PLC store data in bytes.

Each byte contains 8 bits and each 8-bit form a byte. Each 2-byte forms a word and each 2-word forms a double-word.

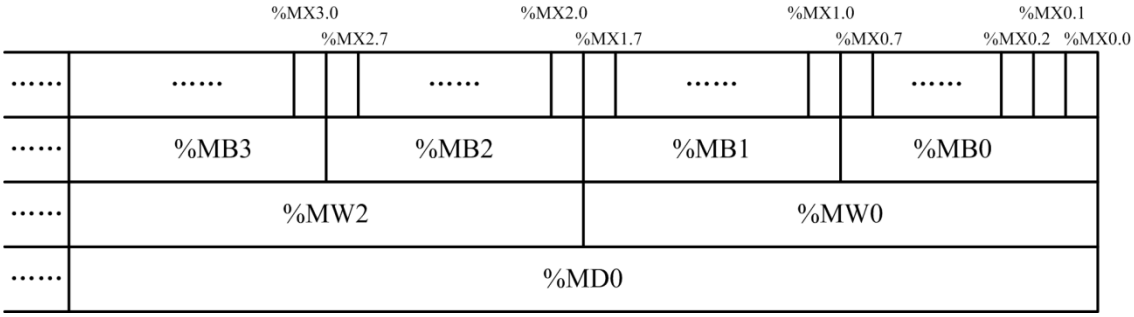


Figure 4-2-1 Storage Format in M

For example: %MX2.7 stands for the 8th bit of %MB2.

%MW12 is a word composed of %MB12 and %MB13.

Bit, byte, word or double-word addressing are possible with the following variable types: BOOL, BYTE, WORD and DWORD. See section 4.4.2 for the ranges of these data types. For the using of variable addressing to access INT, REAL or other data types, see 4.4 for details.

Users shall be aware of that the data storage area may be overlapped when being accessed according to different data types. For example, if the value stored in %MW0 is 3, then the value of %MB0 is 3, the values of %MX0.0 and %MX0.1 are also TRUE. Also, if the value of %MX0.0 is forced as TRUE, then the values of %MB0, %MW0 and %MD0 are all forced as 1 since %MX0.0 is the first bit of these storage addresses.

I and Q areas use the same storage method. Refer to section 7.3 for the corresponding relation between the addresses in I, Q and the actual DI/DO points.

Addressing Format

Note:
In word addressing, the number in the address must be even for a word is composed of two bytes. For example, %MW0 is composed of %MB0 and %MB1 so the next word is %MW2, not %MW1. Address %MW1 is invalid. Double-word addressing also abides by the rule.

According to IEC61131-3 standard, all the direct addresses start with "%". Addressing formats of M storage are explained for examples, as shown in table 4-2-1.

Bit addressing	Format	%MXm.n
	Description	X: bit addressing; m: byte number in M storage; n: bit number of m byte, range 0~7
	Data type	BOOL
	Example	%MX0.3, %MX100.0, %MX3212.7
Byte addressing	Format	%MBm
	Description	B: byte addressing m: byte number in M storage
	Data type	BYTE
	Example	%MB103, %MB2000
Word addressing	Format	%MWm
	Description	W: word addressing m: first byte number of the word in M storage, "m" must be even

	Data type	WORD
	Example	%MW150, %MW3000
Double word addressing	Format	%MDm
	Description	D double-word addressing m: first byte number of the double-word in M storage, “m” must be even
	Data type	DWORD
	Example	%MD300, %MD432

Table 4-2-1 Addressing Format

For addressing format of I and Q storage, change the “M” in the above table to “I” or “Q”. Table 4-2-2 shows the ranges of I, Q and M and any address exceeds the ranges is invalid.

Memory area	Range (bytes)
I (Input)	%IB0~%IB511 (maximum 512 bytes, the actual size determined by PLCs)
Q (Output)	%QB0~%QB511 (maximum 512 bytes, the actual size determined by PLCs)
M (Memory location)	%MB0~%MB8191

Table 4-2-2 Data Memory Area and Range

Please note that the size of M is 8KB with a range of %MB0~%MB8191, among which, %MB0~%MB99 are used as diagnostic addresses for PLCs so that they are suggested not to be used by users. %MB300~%MB799 can be retained in power failure while other parts do not have this function

4.3 CONSTANTS

In PLC programming, parameters of fixed values may be frequently used, for example, the time parameter of timers and the proportion parameter of calculations. These parameters with unchangeable values are called constants. LM Micro series PLC supports a large number of constants in different data types, and the commonly used include BOOL, TIME and numerical constants, as shown in table 4-3-1.

Type of Constants	Format	
BOOL	Description	There are only two BOOL constants: logic values TRUE and FALSE (also shown as 1 and 0), TRUE = 1, FALSE = 0
	Example	TRUE, 0
Integer	Description	Values of numerical constants may be binary, octal, decimal and hexadecimal If an integer value is not decimal, then a number indicating the base and the sign “#” shall be put in front of the value. The values of the decimal numbers 10-15 is represented as letters A-F in hexadecimal number system.
	Example	14 (*decimal number 14*) 2#1001_0011 (*binary number 1001_0011*) 8#67 (*octal number 67*) 16#AE (*hexadecimal number AE*)
REAL	Description	In standard scientific format, real number constants may be represented as decimal fractions and exponents. We use REAL as the data type of real number constants.
	Example	7.4 (*REAL constant 7.4*) 1.64e+009 (*REAL constant 1.64e+009*)
TIME	Description	Generally TIME constants are used to operate the timer. A

		TIME constant consists of a “t#” (or “T#”) and the actual time value that includes components such as days (“d”), hours (“h”), minutes (“m”), seconds (“s”) and milliseconds (“ms”). Please note that the correct order of these components in time entries shall be d, h, m, s, ms.
	Example	T#18ms (*18ms*) T#100s12ms (*100s12ms, the highest component may be allowed to exceed its limit*) t#12h34m15s (*12h34m15s*) the followings are incorrect TIME constants: t#5m68s (*lower component exceeds limit*) 15ms (*T# is missing*) t#4ms13d (*incorrect order of entries*)
TIME_OF_DAY	Description	This type of constants is used to store the times of the day. ATIME_OF_DAY constant consists of “TOD#” (“tod#”, “TIME_OF_DAY#” or “time_of_day#”) and “a time value” in the format: hour: minute: second (the component “second” may be entered as a real number).
	Example	TOD#00:00:00 (*0h0m0s*) TIME_OF_DAY#15:36:30.123 (*15h36m30.123s*)
DATE	Description	A DATE constant is composed of “D#” (“d#”, “DATE#” or “date#”) and “a date value” in the format: year-month-day.
	Example	DATE#2005-05-06 (*2005-5-6 *) d#1980-09-22 (*1980-9-22 *)
DATA_AND_TIME	Description	DATE constants and TIME_OF_DAY may also be combined to form DATE_AND_TIME constants. A DATE_AND_TIME constant is composed of “DT#” (“dt#”, “DATE_AND_TIME#” or “date_and_time#”) and “a date and time value”.
	Example	DT#1980-09-22-15:45:18 (*1980y9m22d15h45m18s*) date_and_time#2001-03-09-00:00:00 (*2001y3m9d0h0m0s*)
STRING	Description	STRING constants are enclosed with a pair of single quotation marks, which may include spaces and special characters
	Example	'Abby and Craig' (*string Abby and Craig *) ':-)' (*string :-)*)

Table 4-3-1 Types and Formats of Constants

Note:
PowerPro is not case sensitive, therefore, “T#3s” and t#3s” indicates the same constant, and both “TRUE” and “true are Boolean constants.

4.4 VARIABLES

According to the definition promoted by IEC61131-3 standard, variable is an identifier composed of characters, numbers and underlines. The identifier must be declared before use to describe the object it identifies. Each identifier corresponds to a certain data type.

In LM Micro series PLCs, declared identifiers may be used to indicate some parameters that change in real-time, and also some function block instructions. This chapter mainly deals with the identification of parameters with changing real-time values, or in other word the variables. ***For identifiers of function blocks, please refer to section 5.3.2.***

The use of variables may replace that of memory addressing in programs. It may also enhance the readability of programs by declaring identifiers according to actual definitions. Before any application of the variables, we need to know more about their detailed classifications. The classifications according to different standards are as followings.

According to different data types, variables can be classified into standard data types and user-defined data types, among which, standard data types contain BOOL, INT, REAL, STRING, TIME etc.

According to the validity scope (application range), variables can be classified into global variables and local variables. Local variables are valid in partial programs of the whole project and cannot be used by other programs. Global variables are valid in the whole project and can be used by any program of the whole project.

According to the properties,, variables can be classified into intermediate variables, input variables, output, input/output variables, etc.

According to the retain ability in power failure, variables can be classified into retainable variables and non-retainable variable.

Section 4.4.2 gives descriptions of all the data types. Section 4.4.4 gives descriptions of the declaration and usage of global variables and local variables are described. Section 4.4.5 gives descriptions of the declaration and usage of input variables, output variables and input-output variables, while section 4.4.6 explains how to declare a retain variable.

4.4.1 Naming Rules of Variables

Naming of variables must abide by the following rules:

The name must start with a character or an underline, following by a certain amount of characters, numbers or underlines.

Characters are case insensitive that “ABC” and “abc” will be recognized as the same variable.

The keywords cannot be use as variable names. PowerPro defined keywords are standard identifiers that their usage and naming have been defined automatically by the system. The keywords in PowerPro are shown in table 4-4-1.

ARRAY	FUNCTION	TYPE
AT	FUNCTION_BLOCK	VAR
CONSTANT	OF	VAR_ACCESS
END_FUNCTION	PERSISTENT	VAR_CONFIG
END_FUNCTION_BLOCK	PROGRAM	VAR_EXTERNAL
END_PROGRAM	READ_ONLY	VAR_GLOBAL
END_STRUCT	READ_WRITE	VAR_IN_OUT
END_TYPE	RETAIN	VAR_INPUT
END_VAR	STRUCT	VAR_OUTPUT

Table 4-4-1 List of Keywords

4.4.2 Data Types of Variables

PowerPro supports standard data types and user-defined data types of variable. See section 4.6 for user-defined data types. This section mainly describes the standard data types.

The standard data types and the range of the variables supported by PowerPro are shown in figure 4-4-2.

Data Type	Type Name	Upper Limit	Lower Limit	Storage Space	Comments
BOOL	bool	0	1	1bit	
BYTE	byte	0	255	8 Bit	
WORD	word	0	65535	16 Bit	
DWORD	double word	0	4294967295	32Bit	
SINT	short integer	-128	127	8 Bit	
USINT	unsigned short integer	0	255	8 Bit	
INT	integer	-32768	32767	16 Bit	
UINT	unsigned integer	0	65535	16 Bit	
DINT	long integer	-2147483648	2147483647	32 Bit	
UDINT	unsigned long integer	0	4294967295	32 Bit	
REAL	real	-3.402823E+38	3.402823E+38	32Bit	Single precision floating point
TIME	time			32Bit	Example: Time1 : TIME := t#3s;
TOD	time and date				Example: Tod1:TOD:= TOD#00:00:00;
DATA	date				Example: Data1:DATA:= D#2008-8-8;
DT	date and time				Example: DT1:DT:= dt#2008-08-08-20:08:08;
STRING	string				Example: Str:STRING(35):= 'hi'

Table4-4-2 Data types of Variables

4.4.3 Declaration of Variables

A variable must be declared before using. PowerPro provides different variable types for different variable functions. To declare a variable, users have to declare the data type as well as the variable type.

In PowerPro, variables may be defined into different types, such as global variables, local variables, input variables, output variables, etc. See table 4-4-3 for details.

Type of Variables	Data Type
VAR	Local variable can be used only in the current program. A variable with the same name can be declared in other programs and they are recognized as two variables.
VAR_INPUT	Input variable is used to transfer parameters in the calling of a POU. The parameters can be transferred to subprograms or other POUs through input variables. See section 5.3 for details.
VAR_OUTPUT	Output variable is used to transfer parameters in the calling of a POU. The parameters can be transferred from the called POU to the POU that calls it through output variables. See section 5.3 for details.
VAR_IN_OUTPUT	Input/output variable is the combination of VAR_INPUT and VAR_OUTPUT and is used to transfer parameters.
VAR_GLOBAL	Global variable can be used in any program of the current project. Meanwhile, a variable with the same name cannot be declared.

Table 4-4-3 Types of Variables

VAR, VAR_INPUT, VAR_OUTPUT, VAR_IN_OUTPUT, and VAR_GLOBAL are keywords to identify variables.

There are two ways to declare a variable: auto-declaration and manual declaration.

Auto-declaration

If the "**Autodeclaration**" option of the Editor Options is activated, whenever a new variable is input into the POU, a dialog will appear in the editor for the declaration of this variable, as shown in figure 4-4-1. The Class, Name and Type in this dialog are mandatory.

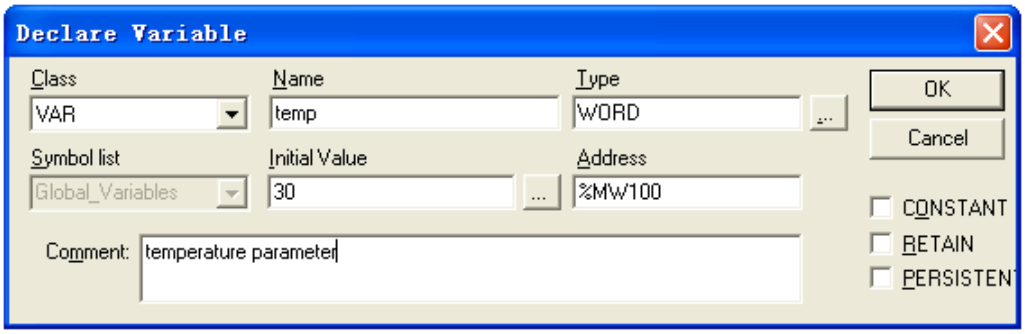


Figure 4-4-1 Auto-declaration

The options of the "Autodeclaration" dialog are as follows:

- **Class**: five classes, see table 4-4-3 for the differences.
- **Name**: variable name, i.e. identifier. See section 4.4.1 for variable naming rules.
- **Type**: selection of data type, data types may be entered manually into the "Type" field or selected from the Input Assistant dialog that are opened by click the button. See section 4.4.2 for the data types.

- **Symbol list**: the symbol list is only available when “VAR_GLOBAL” is selected in the “Type” field. The default setting of which is “Global_Variables”. In the declaration of a global variable, a “Global_Variables” will appear in the “Global Variables” folder of the “Resources” tab. Double click on the “Global_Variables”, then all declared global variables will be displayed, as shown in figure 4-4-2.
- **Initial value**: the initial value of the variable. To initialize the variable, a constant value that is corresponding to variable data type shall be entered here.
- **Address**: declare the address for the variable..
- **Comment**: comment for variable.

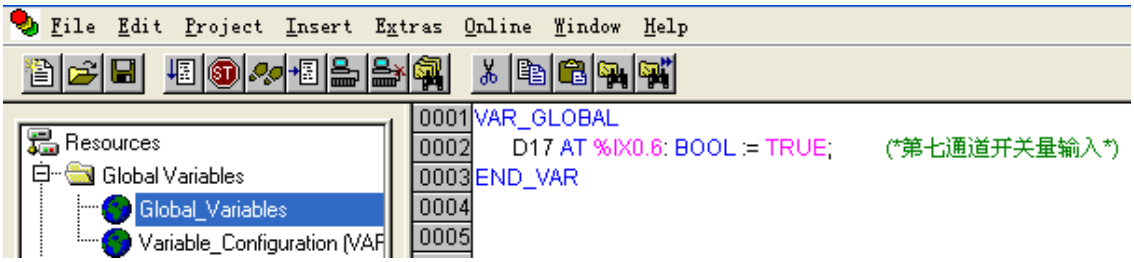


Figure 4-4-2 Code of variable declaration

In the auto-declaration of a variable, users shall pay attention to the follows:

A variable can be specified with an address and the address format must be consistent with that has been mentioned in section 4.2. When a variable is specified to an address, it will be stored in the data storage appointed by this address. For example, the variable “temp” declared as shown in figure 4-4-1 will be stored in M storage as its address “%MW100” indicates. In the program, if the value of %MW100 is changed through direct addressing, the value of the variable “temp” will also be changed. The variable which is not specified to an address will be stored in N storage.

In the declaration, a variable can be initialized with a constant value of which the type must be consistent with the variable data type. For example, the initial value of a TIME variable must be a TIME constant like t#5s and the initial value of the variable will be set as such after power on.

After auto-declaration, a statement of the declared variable will appear in the declaration window. For example, when the variable “temp” in figure 4-4-1 has been declared, the following message will appear:

```
PROGRAM PLC_PRG
VAR
Temp AT %MW100: WORD := 30;          (*temperature parameter*)
END_VAR
```

When a global variable has been declared, it will be displayed in the Global Variables of the Resources tab.

In auto-declaration, there are two options at the lower right corner of the dialog box: CONSTANT and RETAIN. If the CONSTANT is activated, the variable will be saved as a constant and its value cannot be changed. If the RETAIN is activated. the variable will be stored in R storage and its value can be retained after power-failure.

When a new variable is created, the system can auto-declare it. However, when this variable is deleted, the declaration statement in the editor will not be deleted automatically, so that a variable cannot be declared repeatedly. These unused declaration

statements may be find with the command “Project”/“Check”/“Unused Variables”. See section 8.2.3 for detailed usages.

Manual declaration

Besides the auto-declaration, a variable can also be declared manually in PowerPro. The manual declaration adds instructions directly in the declaration window without the help of the dialog box “declare variable”. A variable declaration adopts the following syntax:

<Identifier> {AT<Address>} : <Data Type> {: = <Initialization>};

The parts in the braces { } are optional.

Variables of different types shall be declared in different positions. For example, a local variable shall be declared between the keywords VAR and END_VAR while an input variable shall be declared between the keywords VAR INPUT and END_VAR.

The declaration window can be defined as a table. If the “Declaration as tables” option is activated in the dialog box of “Project”/“Options”/“Editor” or on the context menu in editor, the declaration editor looks like a table, as shown in figure 4-4-3.

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN | INFO

	名称	地址	类型	初始值	注释
0001	AO1		Analog_OUT		
0002	DP		DP_Slave		
0003	xunhuan		BOOL		
0004	TP1		TP		
0005	TP2		TP		
0006	TON_1		TON		
0007	TON_2		TON		
0008	TON_3		TON		
0009	TON_4		TON		
0010	TON_5		TON		

Figure 4-4-3 Declarations as Tables

A new variable can be brought into this declaration table by using the command “Insert”/“New Declaration”. By default, this new variable will be set as “Name” in the Name field, and “Bool” in Type field. These values can be changed according to the user. “Name” and “Type” are both necessary for a complete declaration of variables.

The declaration window can be switched back to normal format by deactivating “Declaration in table form” in the context menu.

Comparing with text declaration, declaration in table form is much more succinct. For beginners, auto-declaration or declaration in table form are recommended.

The difference between variable calling and addressing calling

The “variable+address” calling is quite similar with address calling with some minor differences. The data types that can be called through addressing are only BOOL, BYTE, WORD and DWORD while more data types can be called through “variable+address”.

For example: in case of declaring a REAL variable the address %MD100, Directly set the address as %MD100 will result in a the DWORD data type but not REAL. In this case the “variable+address” method shall be used to declare this variable (only the initial address needs to be specified, and the length of the data will be calculated automatically based on its type), which achieves the declaration of a REAL variable at the address %MD100. .

4.4.4 Global/Local Variables

A complicated project may consist of a number of POUs (Programs, Function blocks). Declare A variable that can be recognized throughout the project shall be declared as a global variable and a variable that can be only used in the current program or function block shall be declared as a lock variable.

In the auto-declaration, a global variable is declared as VAR_GLOBAL while a local variable is declared as VAR.

In a program, all the addresses in I , Q and M storage area can be called, read and written by all POU. So the global variables can replace the use of storage addresses in all POU.

4.4.5 Input/Output Variables

In programming, subprogram may be called. To transfer parameters in main program to subprogram and transfer the calculating result to main program, the input variables and output variables shall be employed. The input variables and output variables are declared in subprogram. Input variable is a variable that transfer from main program to subprogram while output variable is a variable that transfer from subprogram to main program. The input/output variable is the combination of input variable and output variable.

In the auto-declaration of a variable, an input variable is defined as VAR_INPUT, a output variable is defined as VAR_OUTPUT while an input/output variable is defined as VAR_IN_OUTPUT.

See section 5.3 for details of POU calling.

4.4.6 RETAIN Variables

Many projects may require some of the variables retain their value after a power failure. In the declaration, a variable can be directly defined as a Retain variable that saved in R storage which has the power-failure protection function.

In a variable declaration, select “RETAIN” at the lower right corner of the dialog box to save the variable as a Retain variable. A Retain variable can also be defined manually between the keywords VAR_RETAIN and END_VAR.

It has been mentioned in section 4.1 that partial addresses in M (%MB300~%MB799) also have the power-failure protection function. Declare a variable as a RETAIN variable has the same effect as stored a variable in addresses %MB300~%MB799.

4.4.7 Pointer Variables

Pointer variables are a special kind of variables. In LM Micro series PLCs, each storage memory occupies a CPU address that is called absolute address. That is to say, I, Q, M or N, R storage all has their own partial addresses of the CPU while the address in I, Q and M that is accessed by addressing, such as %MW100, can be considered as relative address. Pointer is an absolute address.

If two relative addresses are adjacent, their absolute addresses are adjacent too. The storage in LM Micro series PLCs is stored in bytes, so if the absolute address of %MW100 is pt, the absolute address of %MW102 will pt+2 and the absolute address of %MW200 is pt+100.

In programming, this pointer feature may be used to realize more complicated functions. A pointer variable shall be declared before using.

The declaration of Pointer variable is quite similar to other data types, only that the data type shall be set as POINTER TO <data type>.

Pointer declarations adopt the following syntax:

```
< Identifier > : POINTER TO < Datatype/Functionblock>;
Example:
pt:POINTER TO INT;   (*define a pointer of INT type pt*)
Var_int1:INT := 5;     (*define a INT variable Var_int1:=5*)
Var_int2:INT;        (*define a INT variable Var_int2*)
pt := ADR(Var_int1);   (*assign the address of Var_int1 to pt*)
Var_int2:= pt^;   (*assign the value of pt pointer to address to Var_int2, that is, Var_int2=5*)
```

Pointer Usage Example:

In case of moving 100 WORD variables that are stored in M storage addresses stating from %MW100 to the addresses starting from %MW300 to make these variables retainable after power-failure:

Since the data area is quite large, it will be convenient to move the data area by using pointer.

Declare two pointer variables pt1 and pt2, one points to %MW100 and the other points to %MW300. Address operator ADR and content operator ^ are used here, refer to <<Instruction Manual>> for more.

Variable declaration as follows:

```
VAR
        m: INT;
        pt1: POINTER TO WORD;
        pt2: POINTER TO WORD;
END_VAR
```

Where, m is used to transfer 100 bytes.

Here ST programming language is used for program writing; *see section 9.3 for more of ST language.*

In the FOR loop, at each loop the values of pt1 and pt2 increase by 2 (pointer of WORD type) and assign the value of pt1 to pt2.

The program as follows:

```
pt1:=ADR(%MW100);
pt2:=ADR(%MW300);
FOR m:=1 TO 100 BY 1 DO
 pt2^:=pt1^;
pt1:=pt1+2;
pt2:=pt2+2;
END_FOR
```

4.5 ARRAY

The data management functions of PowerPro are very powerful. It supports not only many different data types but also multi-dimensional data. One-, two-, and three-dimensional arrays can be defined according to elementary data types. Arrays can be defined automatically or manually in the declaration window of a POU.

The identifier of array is ARRAY and the syntax is:

<Arrayname>: ARRAY [<L1>..<U1>, <L2>..<U2>, <L3>..<U3>] OF <elementary data types>;

L1, L2 and L3 identify the minimum of the field range, U1, U2 and U3 the maximum. The field range must be integers. For one-dimensional array, only L1 and U1 shall be configured, For two-dimensional array, L1, U1, L2, U2 shall be configured while L1, U1, L2, U2, L3, U3 shall be configured for three-dimensional array.

A dialog box of the auto-declaration of array [1..10] is shown in the following figure:

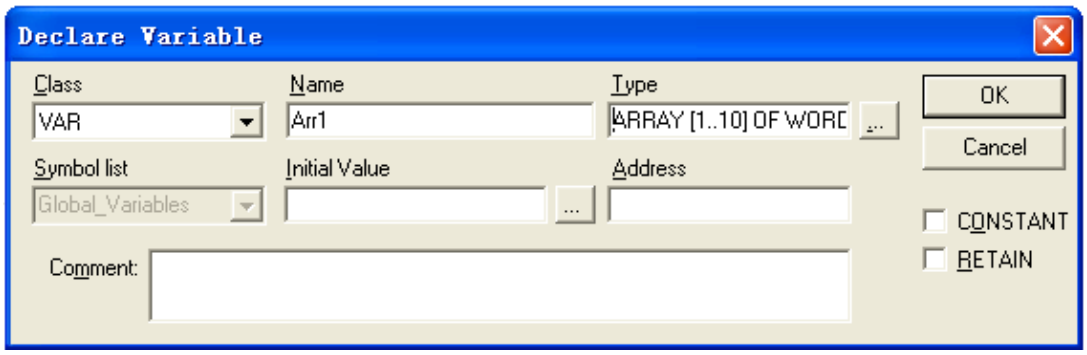


Figure 4-5-1 Auto Declaration of Array

In the declaration of arrays, the elements assignment is called the initialization of array. The complete elements assignment is optional in array declarations

Example for array declaration:

*Card_game: ARRAY [1..13, 1..4] OF INT; (*two-dimensional array Card_game*)*

Example for the complete initialization of an array

Arr1:ARRAY [1..5] OF BYTE:= 1,2,3,4,5;
*Arr2:ARRAY [1..2,3..4] OF INT := 1,3(7) ; (*short for 1,7,7,7 *)*
*Arr3:ARRAY [1..2,2..3,3..4] OF INT := 2(0),4(4),2,3; (*short for 0,0,4,4,4,4,2,3 *)*

Example of the partial initialization of an Array

Arr1:ARRAY [1..10] OF BYTE:= 1,2;

Elements that have no pre-assigned value are initialized according the default initial values of the elementary data types. In this example, the elements [3] to [10] are therefore initialized as 0.

4.6 USER-DEFINED DATA TYPES

In some practical applications, certain customized data may be necessary. Each group of these customized data may contain parameters of different types, such as REAL, WORD or TIME, therefore it is convenient to use user-defined data types to manage these data.

User-defined data types have already been mentioned in section 2.5.2. In the "Data Types" tab of Object Organizer, structures or data types can be created as objects and data types can be defined by users.

First right click to Add Object and name the new data type according to the variable naming rules. After this, the identifier can represent this data type as a structure.

Structures begin with the keywords TYPE and STRUCT, end with END_STRUCT and END_TYPE.

The syntax for structure declarations is as follows:

```
TYPE < Structurename >:
STRUCT
        <Declaration of Variables 1>
        < Declaration of Variables 2>
        ...
        < Declaration of Variables n>
END_STRUCT
END_TYPE
```

<Structurename> is a type that is recognized throughout the project and can be used like a standard data type. The only restriction is that variables may not be placed at addresses, i.e. the AT declaration is not allowed.

Example: declaring a structure named Polygonline

```
TYPE Polygonline:
STRUCT
        Start:ARRAY [1..2] OF INT;
        Point1:ARRAY [1..2] OF INT;
        Point2:ARRAY [1..2] OF INT;
        Point3:ARRAY [1..2] OF INT;
        Point4:ARRAY [1..2] OF INT;
        End:ARRAY [1..2] OF INT;

END_STRUCT
END_TYPE
```

Example: the initialization of a structure

```
P1:Polygonline:=(Start:=3,3,Point1=5,2,Point2:=7,3,Point3:=8,5,Point4:=5,7,End:=3,5);
```

Example: the initialization of a structure array

```
TYPE STRUCT1:
STRUCT
        p1:int;
        p2:int;
        p3:dword;
END_STRUCT
END_TYPE
A1[1..3] OF STRUCT1:=(p1:=1,p2:=10,p3:= 3),(p1:=2,p2:=0,p3:=2),(p1:=4,p2:=5,p3:=1);
```

Syntax for access on structure components:

<Structurename>.<Componentname>

Example

If there is a structure named “Week” that contains a component named “Monday”, the component can be accessed by the following syntax:

Week.Monday

Chapter 5

Program Organization Unit

This chapter introduces an important concept in PowerPro—program organization unit (POU). POU is the base structure of a project. Any complicated project is composed of POUs and POUs include programs, functions and function blocks.

The basic concept of POUs is given in section 5.1 to allow the users obtain basic knowledge about POUs.

In section 5.2, we explain how to create a POU, and in section 5.3 the POU calls are discussed. The key point of these two sections is the rules to use POUs. In section 5.4 we discuss how to manage POUs using PowerPro.

For beginners, if you encounter concepts or rules that are not easy to understand in the learning process, it is suggested to skip this chapter and learn through practices, which will help you understand the content easily. It is suggested to learn the following chapters in this way too.

5.1 BASIC CONCEPT OF POU

POU is short for Program Organization Unit. A POU can be a Function, a Function Block or a Program. Each POU consists of a declaration section and a body. The body is written in one of the IEC programming languages which include LD, FBD, IL, ST, SFC or CFC. See chapter 9 for more details about programming languages of POU.

5.1.1 Types of POUs

POUs include Programs, Function Blocks and Functions.

- A Program is a statement sequence written for a certain task and it is a set or a group of instructions. Programs are the only executable POUs and they are the subject of logic execution. Programs can be activated by task configurations or be called by other programs.
- Function Blocks are pre-written for a certain operation. The inputs of a function block can be one or more and the outputs can be one or more executing results. Function blocks are different from functions in that function blocks have no return values.
- Functions are also pre-written for a certain operation. In the implementation of a function, it will produce a result according to a series of special inputs and the result which is assigned to the function itself is called return value. A function can only be called by other POUs and cannot be executed alone.

5.1.2 Calling POUs

Two methods for calling POUs:

- Called by a POU which is called by another POU.
- Called by Task configuration which is limited to program calls. If you do not configure the Task configuration, the system will call main program PLC_PRG automatically.

Conform to the following rules in POU calls, as shown in figure 5-1-1.

- Programs can call functions, function blocks and other programs.
- Function blocks can call functions and other function blocks.
- Functions can call functions.

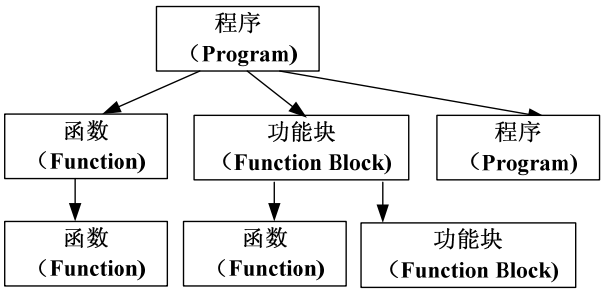


Figure 5-1-1 POU calls

5.1.3 Main Program PLC_PRG

PLC_PRG is the default name of main program and it is a special POU. Each project must include a PLC_PRG to run normally. This POU is called exactly once per control cycle by default and there is no need to configure Task configuration explicitly. So each project must contain the PLC_PRG as a main program to call other POUs.

5.2 CREATING NEW POUS

5.2.1 Creating New Programs

- Right click in the “POUs” tab in the object organizer, and select “Add Object”. The default name of a new POU is “PLC_PRG” if there are no POUs. In the dialog “New POU”, select “Program” in “Type of POU”, as shown in figure 5-2-1.

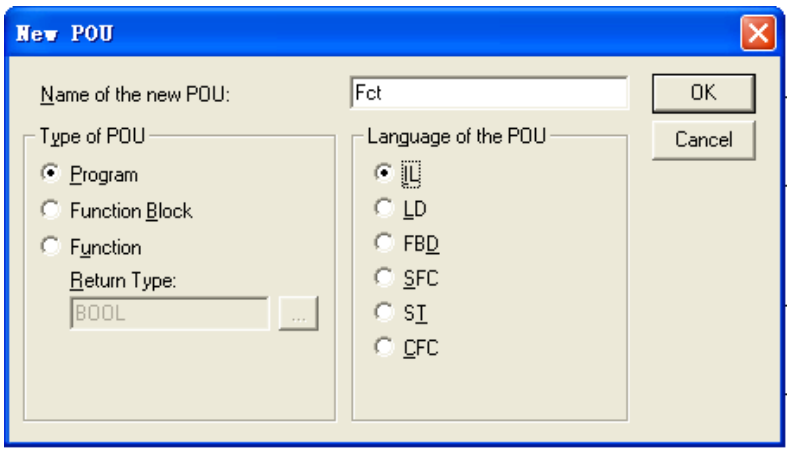


Figure 5-2-1 New Program

5.2.2 Creating New Function Blocks

Creating a new function block is similar to creating a new program. Right click in the “POUs” tab in the object organizer, and select “Add Object”. In the dialog “New POU”, select “Function Block” in “Type of POU” and enter the name in the field “Name of the new POU”.

5.2.3 Creating New Functions

The external structure of a function is similar to that of a function block. The only difference is that a function has only one output. We describe how to create a new function below.

Right click in the “POUs” tab in the object organizer, and select “Add Object”. In the dialog “New POU” select “Function” in the field “Type of POU”, and select or enter “Return Type”, “Language of the POU” and “Name of the new POU” fields, as shown in figure 5-2-2. If the name of the new POU is “Fct” the function name will be “Fct”. Generally meaningful names are used to make it easier to identify.

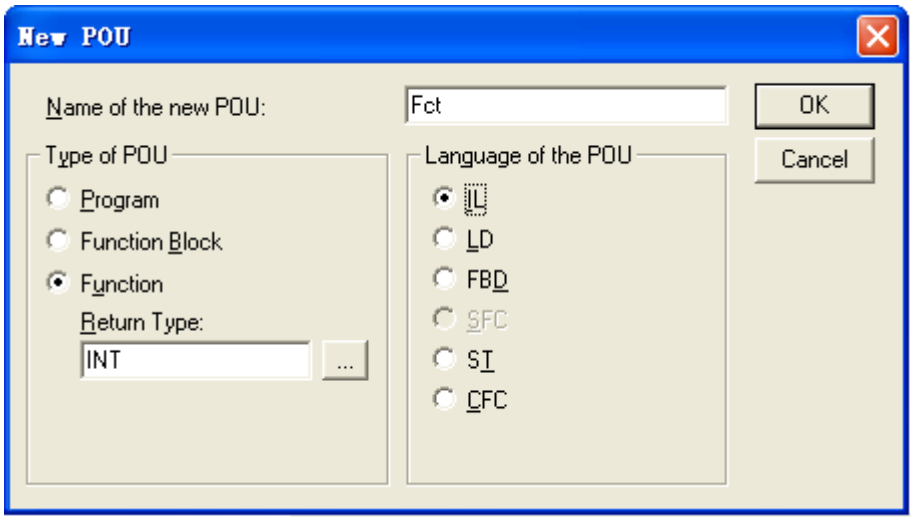


Figure 5-2-2 New Function (1)

After configuration is done, the function structure is generated automatically. You can add content to it as needed, as shown in figure 5-2-3.

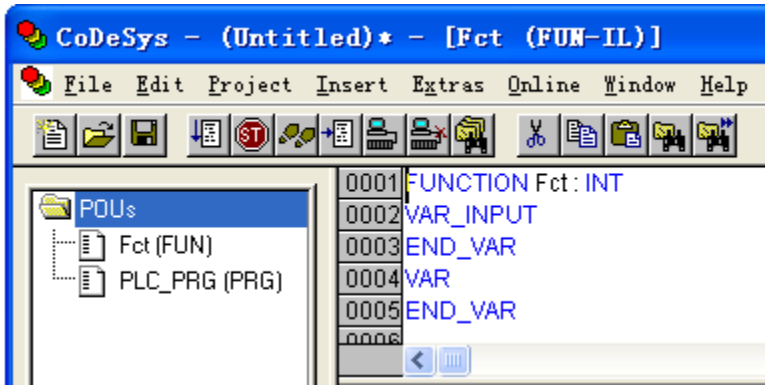


Figure 5-2-3 New Function (2)

Now we discuss how to write a function using a mathematical formula as an example, as shown in figure 5-2-4. Declare the input variables between VAR_INPUT and the first

END_VAR and declare local variables between VAR and the second END_VAR, and write the implementation part in the window above.

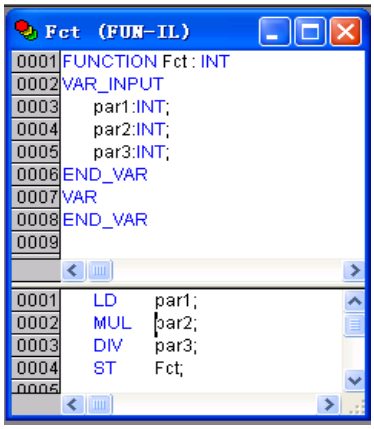


Figure 5-2-4 New Function (3)

Save it after you have finished it. You can call the function directly in other POUs after it has been compiled without errors, as shown in figure 5-2-5.

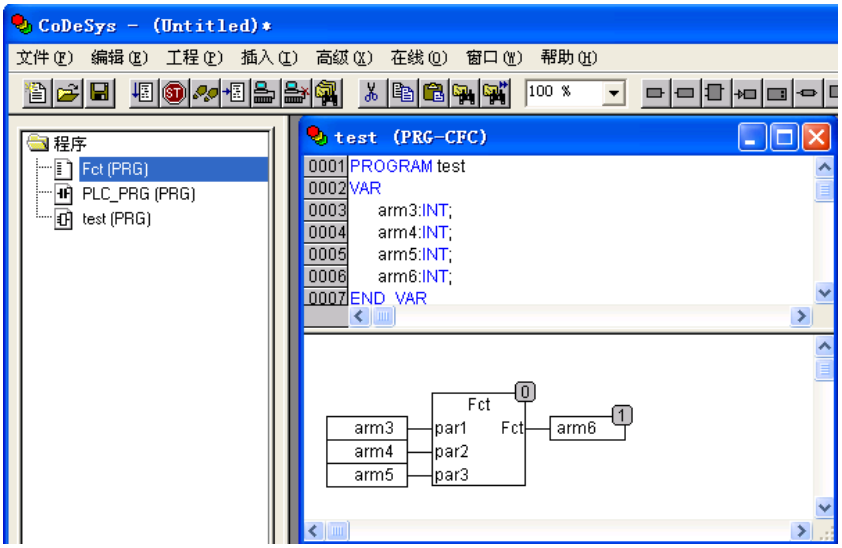


Figure 5-2-5 New Function (4)

5.3 CALLING A POU

Two methods for calling a POU:

- Called by a POU which is called by another POU.
- Called by Task configuration which is limited to program calls. If you do not configure the Task configuration, the system will call the main program PLC_PRG automatically.

Conform to the following rules in POU calls, as shown in figure 5-1-1.

- Programs can call functions, function blocks and other programs.
- Function blocks can call functions and other function blocks.
- Functions can call functions.

5.3.1 Calling a Program

Programs are the only POUs that can be executed. Programs can call function blocks and functions. A program declaration begins with the keyword PROGRAM, as shown in figure 5-3-1.

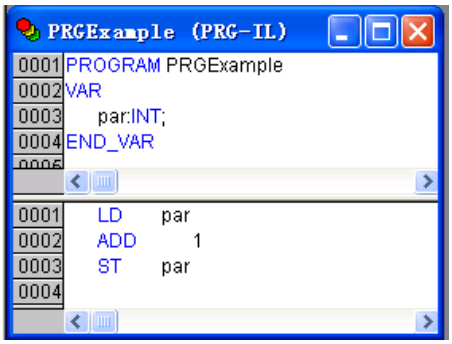


Figure 5-3-1 Program Example

Programs can be called by other programs but it cannot be called by functions.

The program "PRGExample" can be called in different programming languages. If a POU calls a program, and if the values of the program are changed, these changes will be retained until the next time the program is called.

Example for program calls in IL:

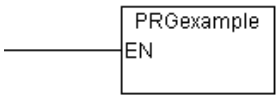
```
CAL PRGExample
LD PRGexample.PAR
ST ERG
```

Example for program calls in ST:

```
PRGExample;
Erg := PRGexample.PAR;
```

Example for program calls in FBD:

Example for program calls in LD:



5.3.2 Calling a Function Block

Function Blocks

A function block is a POU which provides one or more values and provides no return value during the procedure. A function block declaration begins with the keyword FUNCTION_BLOCK. In Figure 5-3-2, there is an example in IL of a function block with two input variables and two output variables. One output (MULERG) is the product of the two inputs, and the other (VERGL) is a comparison for equality.

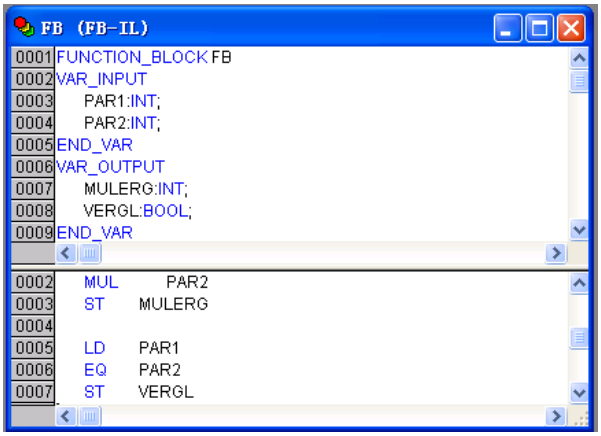


Figure 5-3-2 Example of Function Block

Declaration of Function Block Instances

If you want to call a function block, you have to declare a function block instance. For example, a FUB instance named INSTANZ can be declared as below:

INSTANZ: FUB;

Each function block instance is an independent active object which implements a special logic function. Different programs and tasks can also define and call function block instances and each function block instance has its own memory to keep its own logic states. Function blocks are always called through the instances.

Only the input and output parameters can be accessed from outside of a function block instance, not its internal variables. The instance name of a function block instance can be used as the input for a function or a function block.

Calling a Function Block

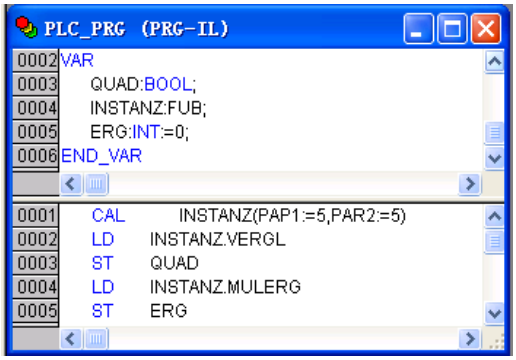
The input and output variables of a function block can be accessed from another POU by setting up an instance of the function block and specifying the desired variable using the syntax “instance name.variable name”.

You can set the input parameters in the text languages IL and ST by assigning values to the parameters after the instance name of the function block in parentheses. For input parameters this assignment takes place using “:=” just as with the initialization of variables during declaration.

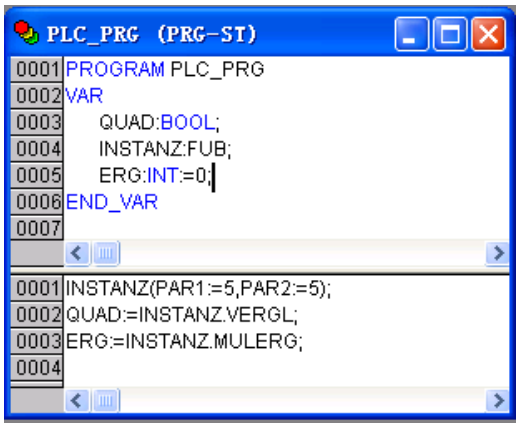
In SFC function block calls can only take place in steps.

All values are retained after processing a function block until the next time it is processed. Because of the internal variables, function block calls with the same inputs do not always return the same output values.

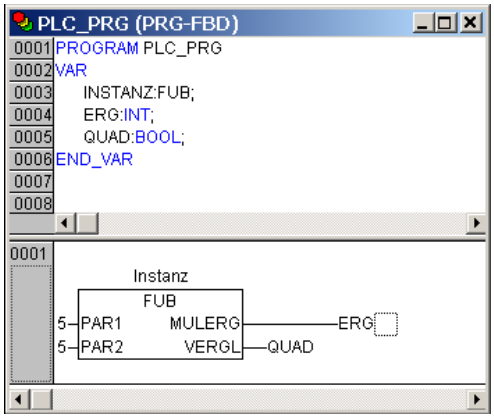
The function block FUB can be called in different languages, as shown in figure 5-3-3. The multiplication result is saved in the variable ERG, and the result of the comparison is saved in QUAD.



Calling FUB in IL



Calling FUB in ST (Declaration part as shown above for IL)



Calling FUB in FBD (Declaration part as shown above for IL)

Figure 5-3-3 Calling FUB

The difference between "Function Block" and "Box with EN"

Box with EN will now be discussed so that it can be distinguished from function blocks. The general difference is that the calling mode is different. For a function block, it will be executed when the program runs no matter it is enabled or not. But a Box with EN will be executed and called only when the EN is enabled. But the detailed application is not given here. We take a simple program written in LD as an example to examine the differences. This is shown in figure 5-3-4.

From figure 5-3-4 we can see that the IN of TON T1 is equal to the EN of "Box with EN", i.e. enabling port in itself. The program starts to run and when the delay is 1s, %QX0.0 is set to 1, and the related channel light Q0.0 in PLC is turned on. For ADD, when it is added the EN is

generated and only when EN is set 1 the “Box with EN” can be called. That means only when %IX0.0 is 1 when ADD will be executed.

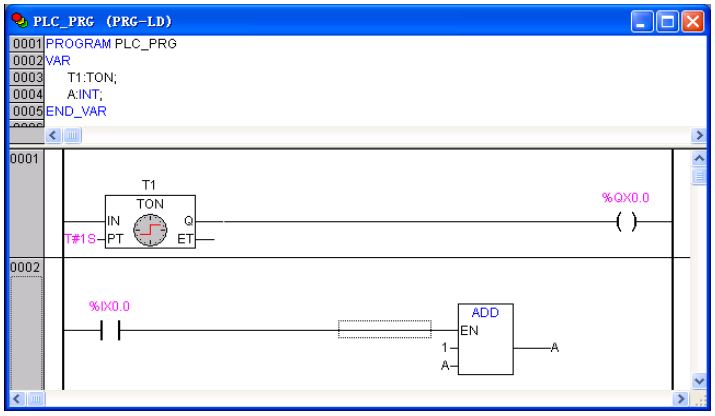


Figure 5-3-4 Function Block and Box with EN

5.3.3 Calling Functions

A function is a POU, which yields exactly one data element when it is processed with a series of specified inputs. Compared with function blocks, functions have only one output and do not have any internal conditions. That means calling a function with the same input parameters always produces the same value. The commonly used mathematical operators such as SIN(X) are typical function types. When declaring a function, keep in mind that the function must receive a data type as the type of the return value. The result is assigned to function itself, which means the output variable is the function name itself. A function declaration begins with the keyword FUNCTION.

In figure 5-3-5 there is a function Fct in IL, in which three input variables are declared. The first two input variables are multiplied and then divided by the third one. The function returns the result of this operation. A function call can be used as operand in an expression and can be assigned.

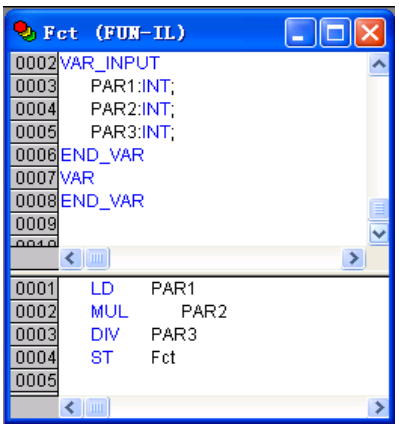


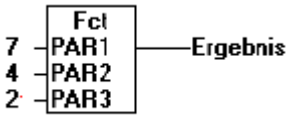
Figure 5-3-5 Example of Function

Example for calling the above described function Fct:

In IL:
LD 7
Fct 4,2
ST Result

In ST:
Result := Fct(7, 4, 2);

In FBD:



In SFC:

A function call can only take place within actions of a step or a transition.

Example

More examples for function calls are described below.

Example 1: if you define a function in your project with the name CheckBounds, you can use it to check range overflows, as shown in figure 5-3-6.

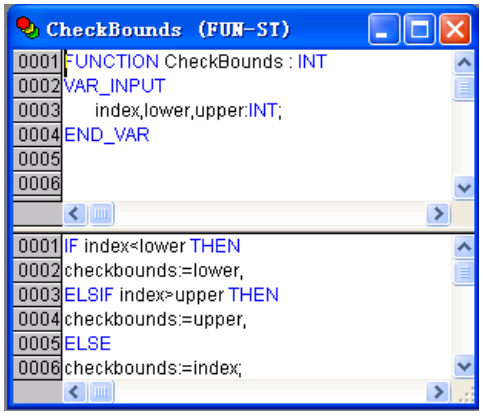


Figure 5-3-6 Example for Function Call (1)

Example 2: If you define functions in your project with the names CheckDivByte, CheckDivWord, CheckDivDWord and CheckDivReal, you can use them to check the value of the divisor. The function CheckDivReal is described in figure 5-3-7.

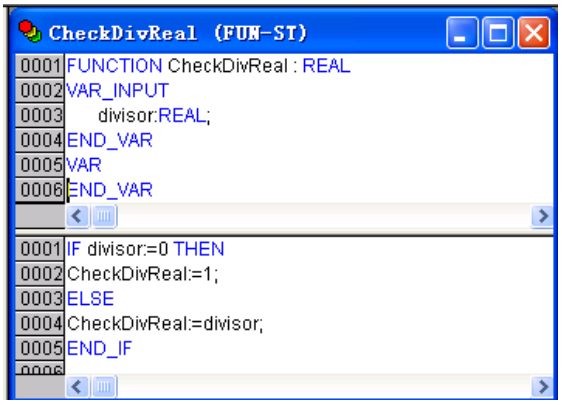


Figure 5-3-7 Example for Function Call (2)

Use the output of function CheckDivReal as the divisor of the operator DIV, as shown in figure 5-3-8, to avoid a division by 0 and the variable d is forced to 1 from 0 and the result is 799.

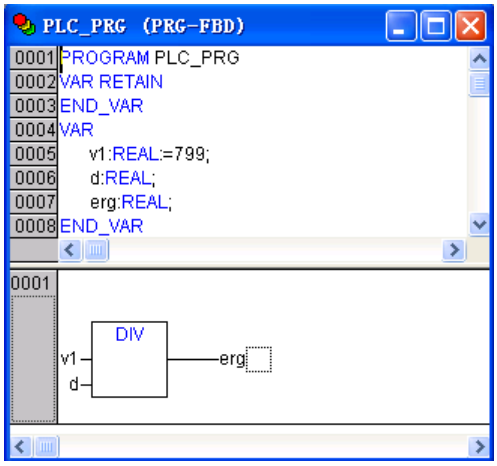


Figure 5-3-8 Example for Function Call (3)

5.4 MENU FOR MANAGING POUS

Right click on the selected POU in the “POUs” tab in object organizer in the main window of PowerPro, and the context menu is shown in figure 5-4-1.

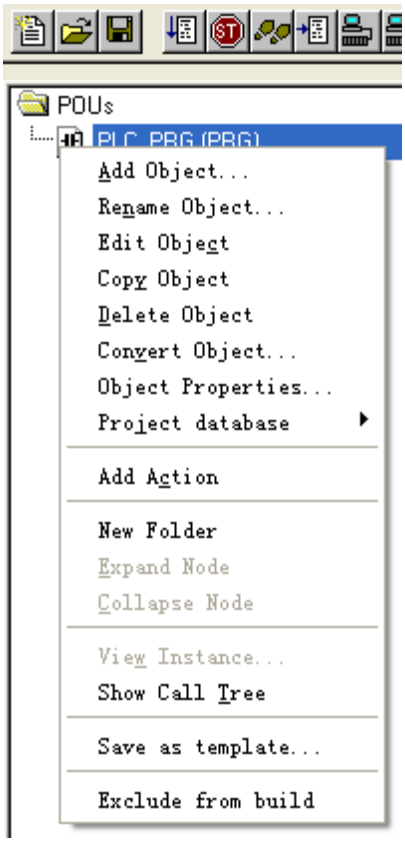


Figure 5-4-1 Menu for Managing POU

5.4.1 Creating New Folders

In order to manage and keep track of large projects you should group your POUs, data types and global variables systematically in folders. You can set up as many levels of folders as you want (a plus sign is in front of a collapsed folder icon). With Drag & Drop you can move the objects as well as the folders within their object type. To do this, select the object and drag it while holding down the left button to the desired position, as shown in figure 5-4-2. Folders have no effects on the programs, but rather serve only to structure your project clearly. You can create more folders with the command “New Folder”. If a folder has been selected, the new one is created underneath it. If a file has been selected, then the new folder is created on the same level.

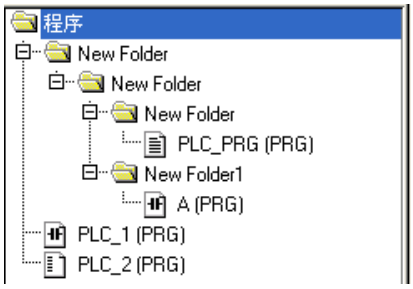


Figure 5-4-2 Folder

5.4.2 Converting Objects

With the command “Convert Object” you can convert POUs from the current programming language into one of the three languages IL, FBD or LD. The command can only be used with programs, functions and function blocks. Before doing this the project must be compiled. In “Target Language” choose the language into which you want to convert the POU, and give

the POU a new name. Remember that the name of the POU must not be already used. Next, press OK, and the new POU is added to your POU list.

Note:
In the conversion process, some useless statements could be generated. If the conversion in between languages occurs frequently, please delete the useless statements first before generating the conversion process to avoid even more unnecessary conversion and errors.

The conversion between different languages embodies the portability of programming languages. However pay attention to the following:

- It is difficult to convert multi-nested ST to LD. The statements written in IF, THEN, CASE, FOR, WHILE and REPEAT format cannot be converted into function block networks directly.
- It is very difficult to convert IL language to other languages unless the application scope and of the way IL operators are written are limited strictly. It's easy to convert the programs written in other languages to IL language.
- Most programs written in FBD language can be converted into IL language and LD language.
- LD language, FBD language and IL language can be converted reciprocally.
- Programs written in ST language can be converted into LD, FBD or IL language and Programs written in ST language can be converted into functions, function blocks and their related parameter values.

An example of converting a program from ST language to LD language is shown in figure 5-4-3.

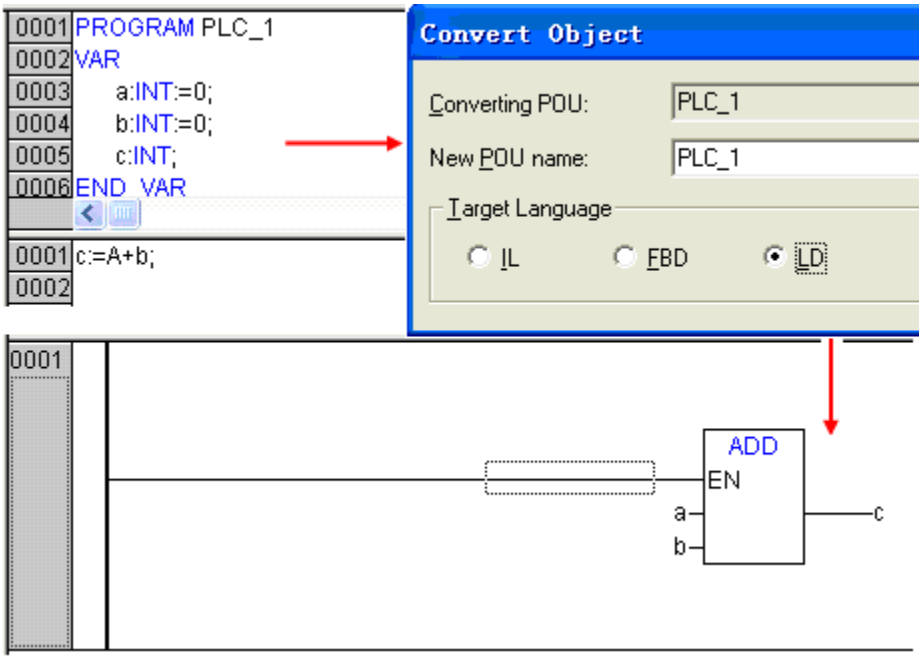


Figure 5-4-3 Convert Object

Chapter 6

PLC Working Mode

After you have learnt how to manage the data and POUs in PLC, you still need to know how PLC works. The PLC working mode is described in section 6.1. Please read it carefully to help you to better understand the program execution process.

The concept of task is proposed in LM Micro series PLC. A task is a piece of content going to be executed in PLC. Generally PLC scans the programs circularly, if you want to generate an interrupt or trigger other events you have to configure it in task configuration. Refer to section 6.2 if you want to know the details of the task configuration process.

6.1 PLC WORKING PROCESS

PLC works in san mode. A scan is the process that CPU executes user programs and tasks circularly. In every working cycle, PLC goes through three steps to run programs which include input sampling, executing users' program and output refurbishing, as shown in figure 6-1-1.

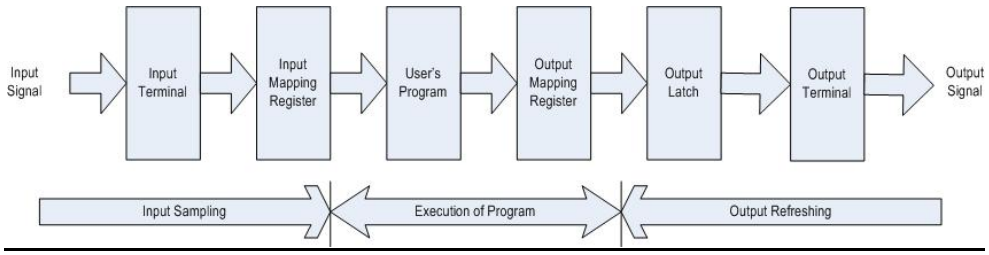


Figure 6-1-1 PLC Working Process

6.1.1 Input sampling

PLC works in scan mode. PLC reads all the signals in order and stores them in input mapping register which is used for storing input states. This process is called sampling. In this working cycle the sampling result cannot be changed and it will be used in the execution of users' program.

6.1.2 Executing users' program

PLC scans every instruction in the program from top to bottom and from left to right, and calculates and processes the data obtained from the input mapping register and output mapping register. Next, the calculating result is sent to the output mapping register which is used for storing the execution result. Note that before the execution is finished, the executing result will not be transferred to output port.

6.1.3 Output refurbishing

After the whole user program is executed, PLC outputs the content in the output mapping register to the output latch which is used for storing output states in order to drive the user's devices. This is called output refurbishing.

PLC repeats the three steps above. The time for executing the three steps is called a scan cycle. In every scan cycle, generally the input sampling time and output refurbishing time are less than 1ms and the time for executing users' program varies with the size of users' program. General PLC scan cycle is limited in tens of milliseconds.

A PLC working cycle mainly contains the three steps. But strictly the following three steps are included and they are performed after input sampling.

- System self-checking: check whether the program is executed correctly and the CPU stops working if time out.
- Exchange information with PowerPro: this is executed once when using programmer input and debugging.
- Network communication: when a communication module is configured, it is used for data exchange with communication object.

When PLC runs the three steps above, i.e., cycle scanning working process, will be executed repeatedly, as shown in figure 6-1-2. The main feature of PLC is that the input signals, execution process and output control are batch processed. The PLC "serial" working mode can avoid the contact competition between relay—contactor and sequential logic mismatch, that is one of the reasons why the PLC has high reliability. But, cycle scanning working process will lead to output lag compared with input and it's one of the disadvantages of PLC.

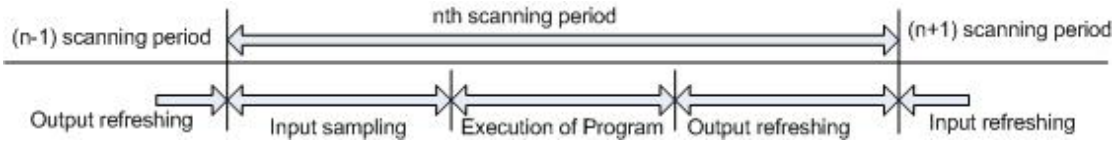


Figure 6-1-2 PLC Scanning Cycle

The state values used in PLC program execution are not from the actual input ports directly but from input mapping register and output mapping register. The state values in input mapping register are the data acquired from input terminal in previous period and they retain in the process of executing users' program. The state values in output mapping register are the executing results. The state values in output latch are from output mapping register in output refurbishing of previous scanning cycle.

In addition, in PLC a watchdog timer is usually used to monitor whether the actual working cycle exceeds the prefixed time to avoid infinite loop or execute not-prefixed program resulting in system paralysis.

Note:
By default, LM Micro Series PLC starts up the watch-dog function automatically. When the scanning time exceeds 500ms, the PLC will consider it as an infinite loop program and will restart the PLC. In this situation, the PLC ERR light will flash for six times at a slower speed before the program is then reset, restart, and execute again.

6.2 TASK CONFIGURATION

For a project, according to different requirements many tasks can be configured to call different programs. Generally it is suggested that a task should be configured to call the main program and the other programs are called indirectly via the main program . Only programs can use this kind of calling mode and function blocks and functions cannot be called in this way.

Strictly if the task configuration is not done, in single task environment the default main program is PLC_PRG and it is called automatically and uniquely and the other programs are called by the main program. If the task configuration is done, program calling depends on the task configuration. Generally, single task environment is enough for PLC control and there is no need to configure the task configuration.

6.2.1 Task Configuration

Double click the “Task Configuration” in the “Resources” tab and the dialog for setting the task configuration is opened. Right click on the “Task Configuration” and select the “Append Task”, as shown in figure 6-2-1.

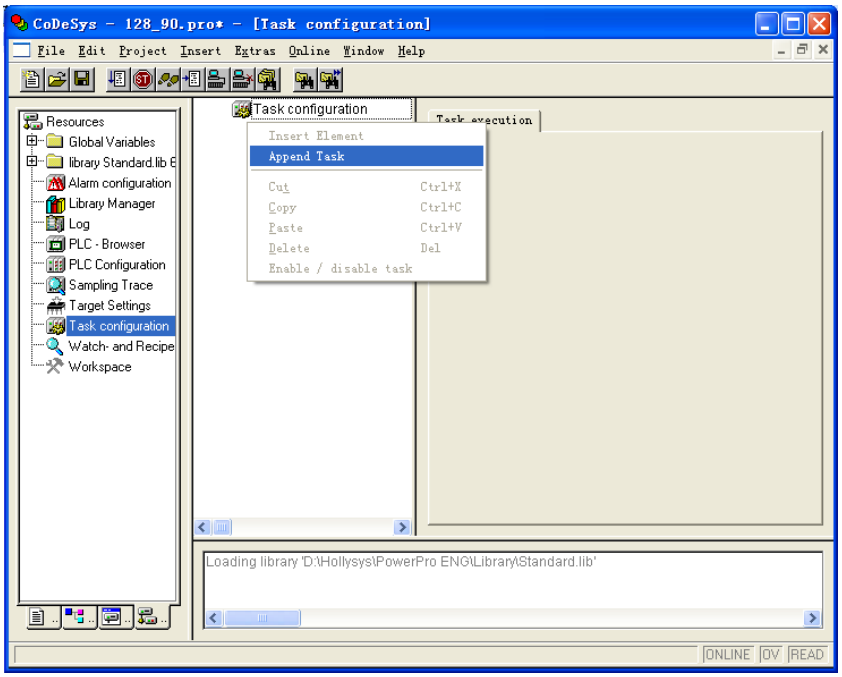


Figure 6-2-1 Task Configuration

The dialog for setting the “Taskattributes” will appear. Enter the information in the fields “Name”, “Priority”, and “Interval”, as shown in figure 6-2-2.

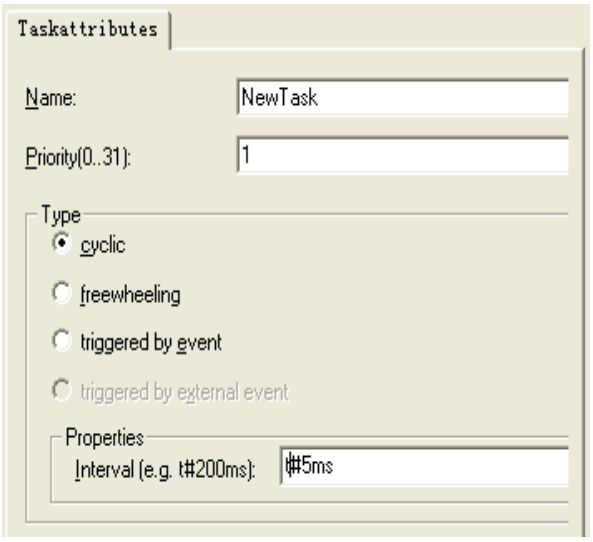


Figure 6-2-2 Task Attributes

Name

- A name for the task is represented with letters and numbers in any way.

Priority

- There is no priority for LM and first call the task configured first, so the default setting is adopted and there is no need to modify it.
- Type

Cyclic: The task will be processed cyclic according to the time definition given in the field 'Properties'/'Interval'. If "cyclic" is selected, then an icon " " will be displayed at the left of the "NewTask".

Freewheeling: The task will be processed as soon as the program is started and at the end of one run will be automatically restarted in a continuous loop. If "freewheeling" is selected, then an icon " " will be displayed at the left of the "NewTask".

Triggered by event: PLC does not support the function

Triggered by external event: PLC does not support the function

Properties

Interval: You should enter the period of time according to the control speed of the POUs included in the task if the "Type"/ "cyclic" is selected. Remember to plus the "t#" before the number (fixed format) and you can choose the desired unit: ms (milliseconds), s (second), m (minute), h (hour), d (day).

6.2.2 System Events

Instead of a "task", a "system event" can also be used to call a POU of your project, which calls the corresponding POU when the related event is triggered. For example, corresponding POUs will be called when T2, T3, T4 timers overflow and fast external interrupt runs.

Double click “Task Configuration” in “Resources” tab, and the window of task configuration will appear on the right of the main window. All the available system events are displayed on the right as soon as the “System events” is selected in the “Task Configuration” tree, as shown in figure 6-2-3. If you actually want the POU to be called by the event, activate the entry in the assignment table (activating is done by a click on the control box) and in the column “called POU”, enter the name of the project POU which should be called and processed as soon as the event occurs.

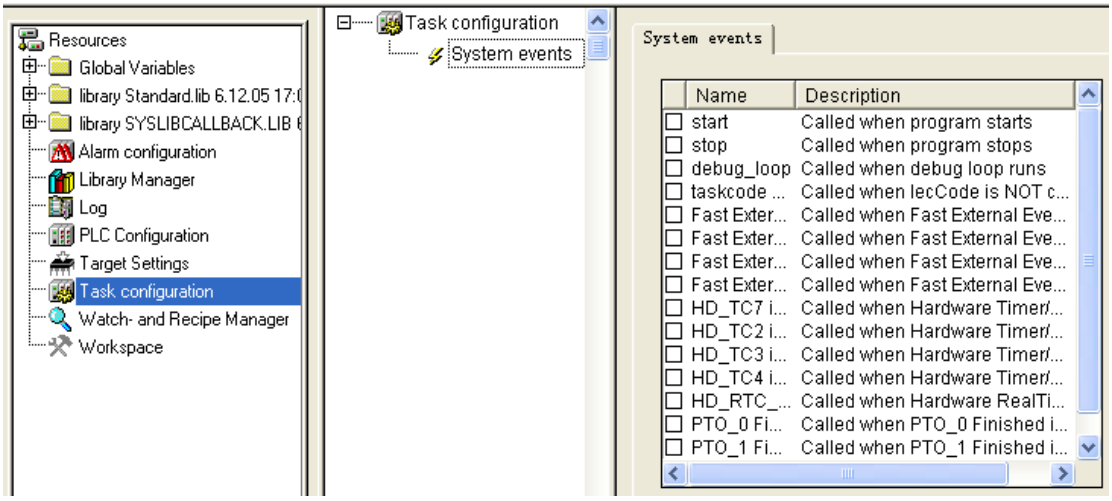


Figure 6-2-3 Dialog of System Events

Configure the system events when using interrupts and refer to section 10.2 for details.

Note
System events are not supported in simulation mode and it will be responded only when the program has been compiled without any error and the login is successful.

6.2.3 Program Calls

Right click on the “NewTask” and select “Append Program Call”, the dialog of “Program Call” will appear, as shown in figure 6-2-4.

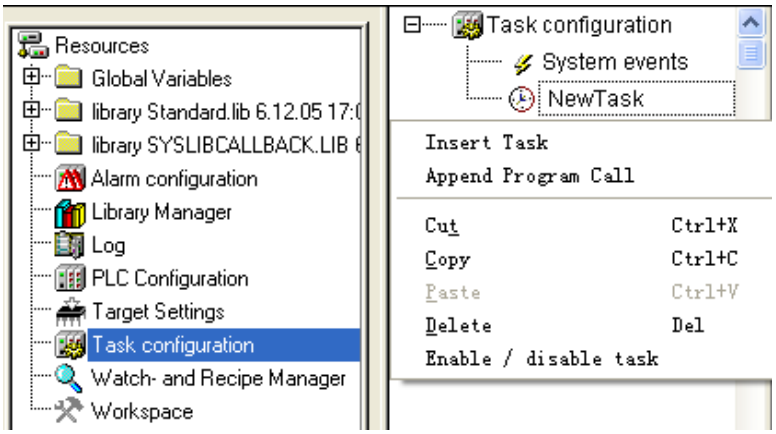


Figure 6-2-4 Append Program Call (1)

Click the ellipsis as shown in the figure 6-2-5 and select the program needed and return by clicking “OK” with the program called by the task.

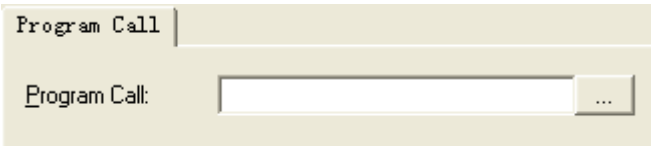


Figure 6-2-5 Append Program Call (2)

An example of program call is used to call the program of reset as shown in figure 6-2-6, where the name is “task”, priority is “1” and the interval is “5ms”.

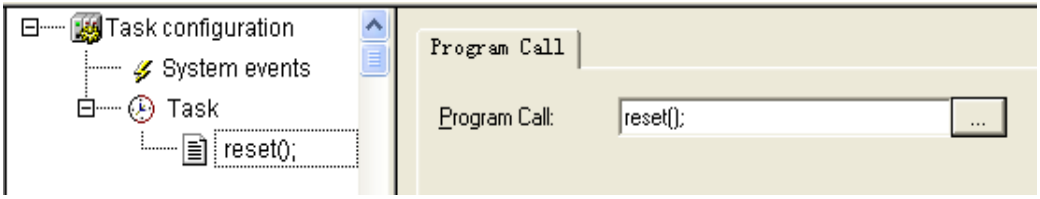


Figure 6-2-6 Example of Task Configuration

Chapter 7

Creating and Managing Projects

After explaining the programming environment in PowerPro, in this chapter we will explain how to create a project. A project contains all the objects, including POUs, data types, resources and arithmetic. The steps to create a new project are flexible. The basic steps include target settings, creating a main program, hardware configuration, and saving the created project.

7.1 TARGET SETTINGS

After using the menu command "File"/"New" in the main window or the button '  ', the dialog titled "Target Settings" will appear. The "Target" refers to the storage space of the PLC and "Target Settings" are configurations of the selected PLC according to storage space.

In "Configuration" option, select "**HOLLYSYS-LEC G3 CPU Extend**", where the target is 120KB. Click the "OK" button a dialog box will appear as shown in figure 7-1-1.

If the storage space of the module is 28KB, select "**HOLLYSYS-LEC G3 CPU**" instead. If you are not sure about the storage space, refer to the appendix for more information.

Choose "**None**" if you want to write a library instruction. ***Refer to section 7.4.5 about how to write a library instruction.***

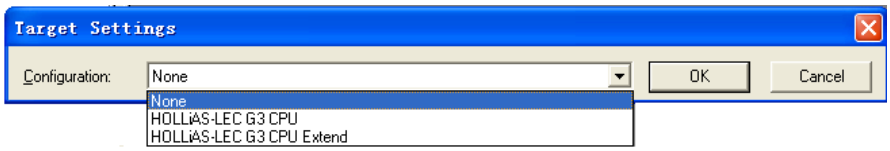


Figure 7-1-1 Select Target

Next, the window titled "Target Settings" will appear, where the default settings will suffice for most of the applications. Click "OK" to exit. This is as shown in figure 7-1-2.

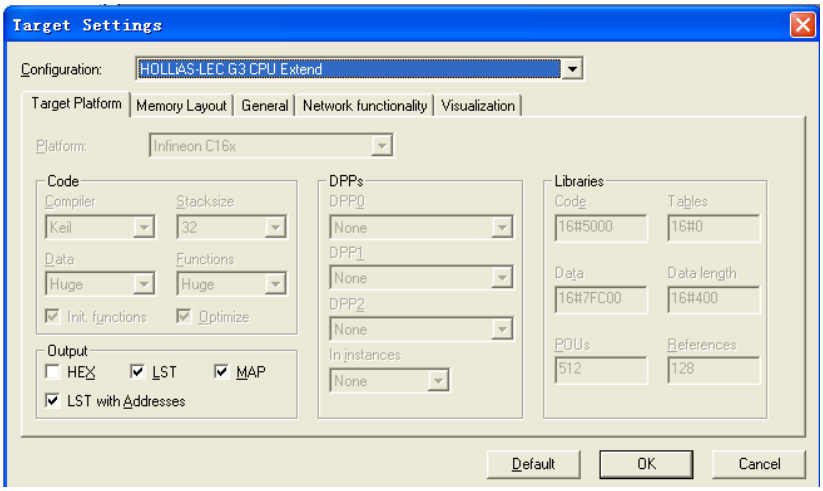


Figure 7-1-2 Target Settings

7.2 CREATING POU

Each project must contain a main program which can call other subprograms. When creating a project, the default “PLC_PRG” is specified as main program and you cannot change it. Otherwise the user program cannot run normally.

After the target settings are done, a dialog box titled “New POU” will appear as shown in figure 7-2-1. In the “Language of the POU” option, you can select any of the languages from IL, LD, FBD, SFC, ST and CFC. Here we select LD. The “Type of POU” is selected as “Program”, and the default main program name is “PLC_PRG”. Click “OK” to exit the window.

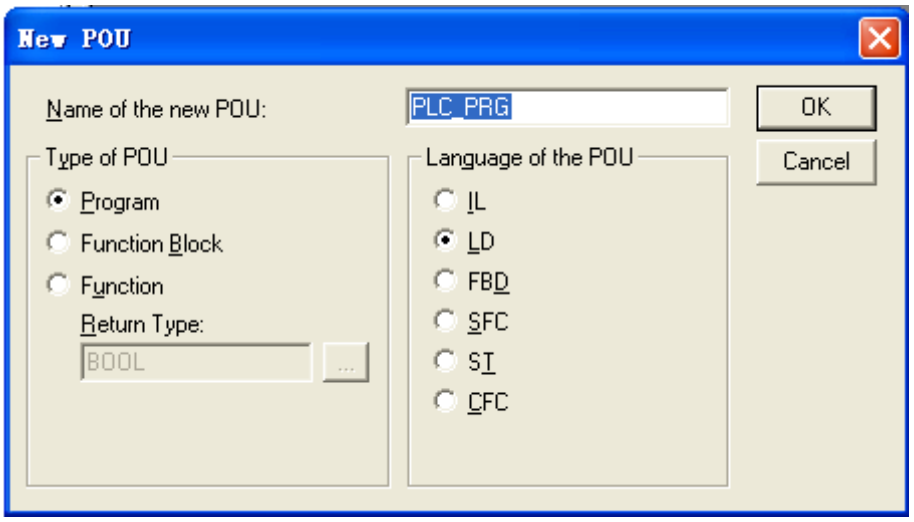


Figure 7-2-1 Creating Main Program “PLC_PRG”

7.3 HARDWARE CONFIGURATION

A PLC system acquires and processes data through hardware modules (including CPU modules and expansion modules). Input channels acquire data from field and output channels controlling electrical equipments in process. In order to complete the acquisition and control, configuration of the hardware modules must be done according to the actual project.

7.3.1 Configuration of CPU Modules

In “Resources” tab, select “PLC Configuration” and the configuration interface will appear. Right click in the PLC configuration window, select the command “Append Subelement”, and select the CPU type to build a “PLC Configuration” tree. Here, select CPU module LM3107 as shown in figure 7-3-1.

The I/Os of CPU modules have fixed I/O addresses. For example, a CPU module LM3107 has 14×DI and 10×DO, where:

The input in words starts from %IW0, and each input channel occupies one bit. The addresses are: %IX0.0,%IX0.1,.....,%IX0.7,%IX1.1,%IX1.2,.....,%IX1.7 and the first 14 addresses are valid.

The output in words starts from %QW0 and each output channel occupies one bit. The addresses are: %QX0.0,%QX0.1,.....,%QX0.7,%QX1.0,%QX1.1,.....,%QX1.7 and the first 10 addresses are valid.

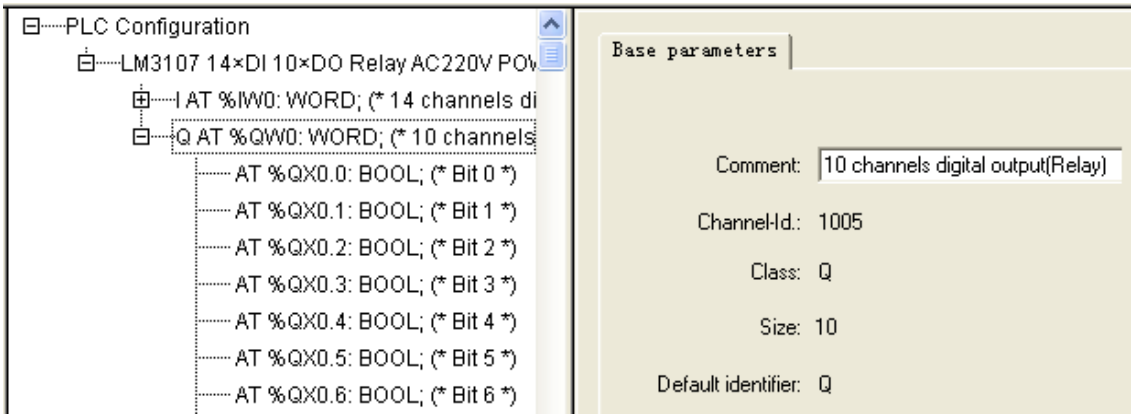


Figure 7-3-1 LM3107 Base Parameters

For CPU input channels, its Input filter needs to be configured. The input filter parameter setting is shown in figure 7-3-2 and the default value is “64”. Select the “Value” column if you want to change the parameter value.

Click the “Module parameters” option and you can set the channel filter parameter. The meaning of each field is as follows:

- Index: The Index is a sequence number (i), which enumerates all the way to the parameters of the module.
- Name: Name of the parameter.
- Value: Value of the parameter, editable.
- Default: Default value of the parameters, which is 64.

The definition of “filter parameter” is as follows: if the acquired data remains the same during 64 scanning cycles, the acquired data is valid, and the filtering is finished. Here the filters are digital, i.e. the number of filters is 64.

Filter parameters are invalid for high-speed input channels. There is no need to set the channel filter parameters where the high-speed input channels are used and even the settings are finished they are not used.

The parameter configurations of the other CPUs are similar. According to the actual control requirements, select an appropriate CPU and add the expansion modules.

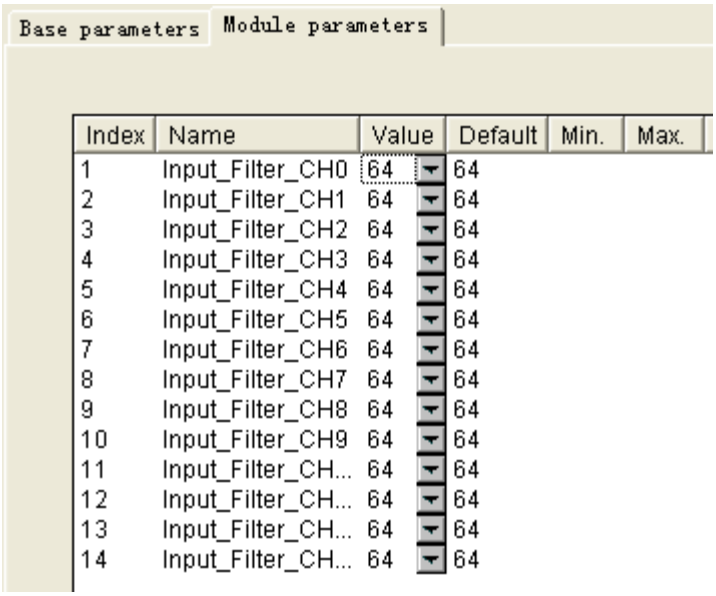


Figure 7-3-2 Parameter Settings of LM3107 Module

7.3.2 Configuration of Expansion Modules

Expansion modules are added when a CPU module is added. Right click on the CPU module and select "Append Subelement" to select the expansion modules needed in the list, as shown in figure 7-3-3. For example, if the "LM3230" module is selected, it will be displayed in the "PLC Configuration" tree, as shown in figure 7-3-4.

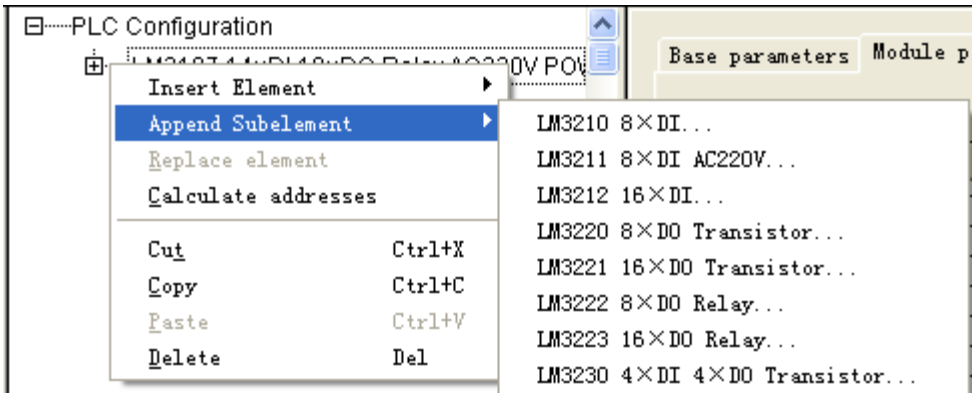


Figure 7-3-3 Append Subelement (1)

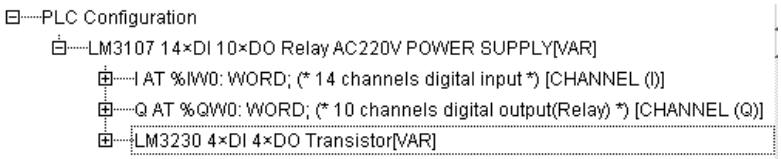


Figure 7-3-4 Append Subelement (2)

Each I/O channel of CPU modules and expansion modules corresponds to an actual physical device and the corresponding relationship is displayed with modules in the "Base parameters". "Base parameters" include Node ids, Input addresses, Output addresses, and Diagnostic addresses, as shown in figure 7-3-5. The "Node id" is defined by an entry in the configuration file by the position of the module in the configuration structure in an order of

"0","1","2" and the "Node id" cannot be changed by users freely. The "Input address" and the "Output address" are the initial addresses of the I/O storage area corresponding to the module channels.

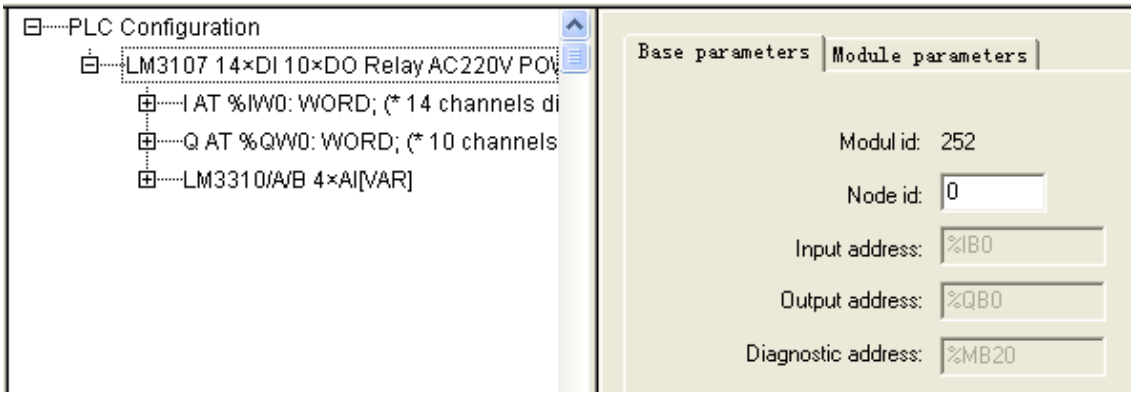


Figure 7-3-5 Base Parameters of Expansion Modules

The "Module ids" of expansion modules are determined by its "Node id", "Input address", "Output address", and module type. For example, if a 4×AI LM3310 is the first expansion module after CPU module LM3107, the channel addresses will be %IW2, %IW4, %IW6 and %IW8. Double click on the module or single click on "+", and you will see the module type and the detailed I/O addresses of each channel.

After a module has been configured, if necessary, you can define an I/O variable to access the module conveniently. Double click on "AT" to activate the input box of variable name and enter the variable name. The character "%" after "AT" stands for the address, which means that the characters after "%" is the address of the variable before "AT".

If necessary you can define a general variable for a number of channels. For example, if the default setting of "%IW0" of CPUs is "I", you can access the first input by "I.0" directly instead of "%IX0.0". In addition, you can define a variable name for each channel, such as defining a variable "start" for "%IX0.0" and a variable "temp1" for "%IW2", as shown in figure 7-3-6.

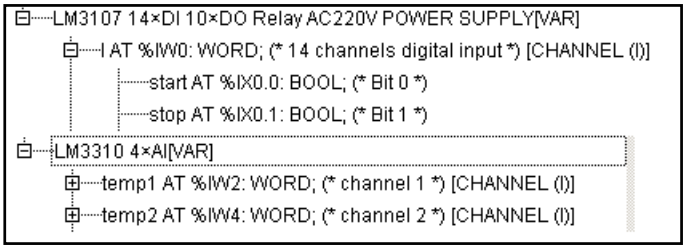


Figure 7-3-6 Definition of I/O Variable

When the cursor position is at "PLC Configuration", and the "Settings" tab is shown on the right of the window, including "Automatic calculation of addresses", "Check for overlapping addresses" and "Save configuration files in project", as shown in figure 7-3-7. The meanings are:

"Automatic calculation of addresses": The node id, input address, output address and diagnostic address in "Base parameters" can be calculated automatically according to the order in which the hardware modules are connected.

"Check for overlapping addresses": During compilation, the project will be checked for overlapping addresses and a corresponding message will be displayed.

"Save configuration files in project": The configuration file will be saved in the project.

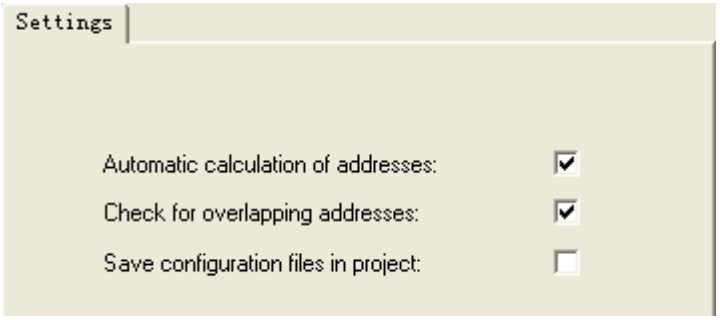


Figure 7-3-7 "Settings"

See<<Hardware Software>>for parameter settings of expansion modules, and see section 10.3 for application of analog modules.

7.4 PROGRAMMING

After the PLC configuration has finished, you can start to write a program. When creating a program, select the programming language from IL, LD, ST, SFC, FBD and CFC. Here, we describe the programming specification in PowerPro using the commonly used LD language as an example. The programming specifications for other languages are introduced in chapter 9.

LD (Ladder Diagram) is a graphics oriented programming language. It is convenient to construct logical operations using LD. The main components in LD include contact, coil, functional blocks and connection line. In LD a plane net is generated by vertical lines and horizontal lines. Generally the leftmost line is the "energy line" and it is always TRUE. Every programming element connects according to the rules to the "energy line" eventually to form a "Grid", "Segment" or "Network" to complete the specific logical operations. This is illustrated in figure 7-4-1.

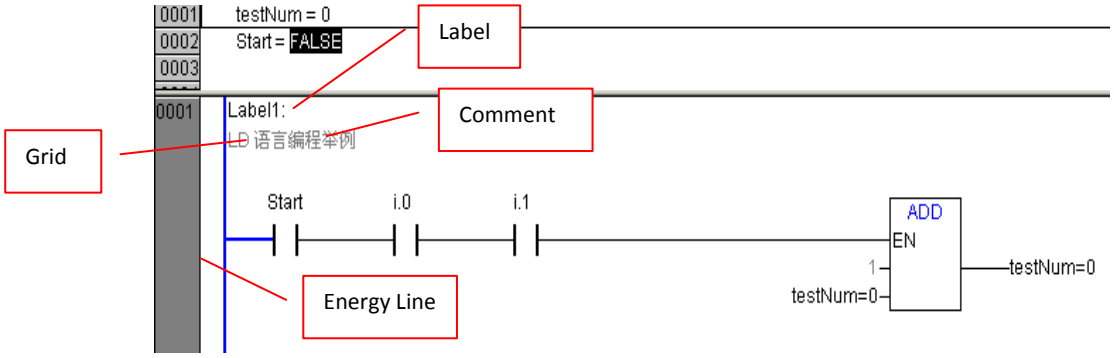


Figure 7-4-1 LD Editor

- Right click in the instruction in workspace and the commonly used commands are displayed in the context menu, as shown in figure 7-4-2.
- Network (before), Network (after): Insert a new network before or after the present network where the cursor is located.
- Contact, Parallel Contact: Insert a contact or parallel contact to the marked position in the network.
- Function Block: Insert a function block at the cursor position.
- Coil: Insert a coil at the cursor position.

- Box with EN: Insert an IEC operator, a function, a function block or a sub-program with EN input into a LD network.
- Jump: Jump to the indicated label if the condition is true.
- Return: When the return condition is true, return to the POU by which the current POU is called if the current POU is called by other POUs.
- Comment: Insert a comment at each network for better understanding of the program.

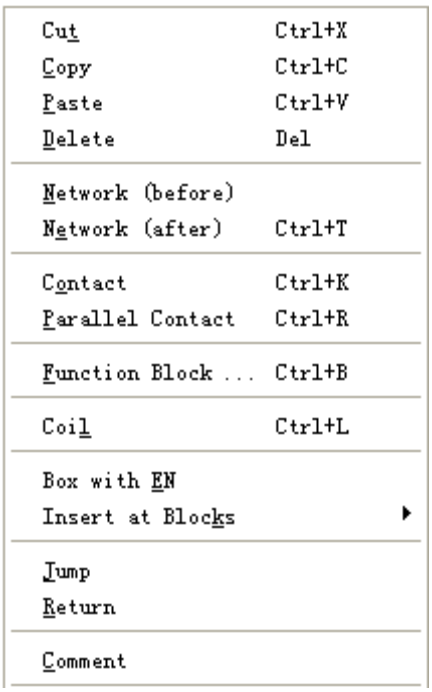


Figure 7-4-2 Context Menu

7.4.1 Network Operation

Network is an important concept in PowerPro which is a basic unit of POU and each POU is composed of networks.

Insert a network

Use the commands “Insert”/“Network (before)”, “Network (after)” to insert a new network in LD editor. It’s the same with the commands “Network (before)”, “Network (after)” in the context menu.

Insert network comment

Every network can be supplied with a multi-lined comment. Use the command “Insert”/“Comment” to insert a comment line and the default text is “Comment”. In “Extras”/“Options”/“Maximum Comment Size”, enter the maximum number of lines to be made available for a network comment (The default value here is 7). In “Minimum Comment Size”, enter the minimum number of lines to be made available for a network comment (The default value here is 0). This is shown in figure 7-4-3. If, for example, a number 2 is entered, at the start of each network there will be two empty lines after the label line. If the minimal comment size is greater than 0, in order to enter a comment you simply click in the comment line and enter the comment. In contrast to the program statements, comments are displayed in grey.

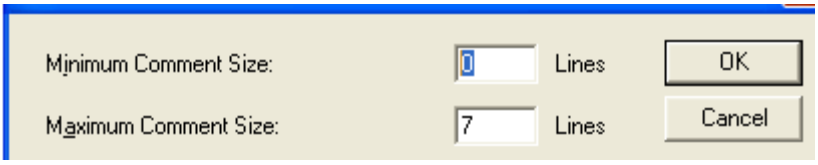


Figure 7-4-3 Settings of Comment

7.4.2 Inserting Contact and Coil

Inserting "Contact"

Shortcut:  Insert a contact in front of the marked location in the network.

If the marked position is a coil or the connection line between the contacts and the coils, the new contact will be connected serially to the previous contact connection.

The contact is preset with the text "???". You can click on this text and change it to the desired variable or the desired constant. For this you can also use the Input Assistant (F2) to select input directly from the variable list, as shown in figure 7-4-4.

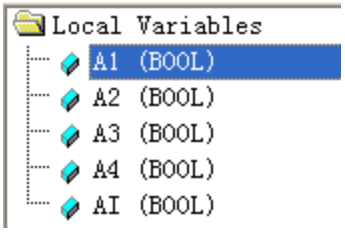


Figure 7-4-4 Variable List

Insert "Parallel Contact"

Shortcut  Insert a contact parallel to the marked position in the network.

If the marked position is a coil or the connection line between the contacts and the coils, the new contact will be connected in parallel to the previous contact connection.

The contact is preset with the text "???". You can click on this text and change it to the desired variable or the desired constant. For this you can also use the Input Assistant (F2) to select input directly from the variable list, as shown in figure 7-4-4.

Insert "Coil"

Shortcut  Insert a coil in parallel to the previous coils.

If the marked position is a connection between the contacts and the coils, then the new coil will be inserted as the last. If the marked position is a coil, then the new coil will be inserted directly above it.

The coil is preset with the text "???". You can click on this text and change it to the desired variable or the desired constant. For this you can also use the Input Assistant (F2) to select input directly from the variable list, as shown in figure 7-4-4.

Coils can only be in parallel. A coil transmits the value of the connections from left to right and copies it to an appropriate Boolean variable. At the entry line the value ON

(corresponds to the Boolean variable TRUE) or the value OFF (corresponding to FALSE) can be present.

"Negate" Operation

Shortcut  Negate contacts and coils.

If a coil is negated, it copies the negated value to an appropriate Boolean variable. If a contact is negated, it connects through only if the appropriate Boolean value is FALSE.

"Set/Reset" Operation

Shortcut:  Coils can be defined to be in Set or Reset state.

A Set Coil is represented by an "S" in the coil symbol. Once you have set the value of this variable to TRUE, it will always remain TRUE until it reset.

A Reset Coil is represented by an "R" in the coil symbol. Once you have reset the value of this variable to FALSE, it will always remain at FALSE until it is set.

7.4.3 Adding Instructions

The instructions in PowerPro can be called in two ways: Box with EN and Function Block. The methods to use them will be described below.

Calling Box with EN

In PowerPro instruction system, some standard instructions, such as Arithmetic Operators, Comparison Operators, Bit-Shift Operators, Assignment Operator, Type Conversion, and Logical Operators, are called in the form of Box with EN.

You can insert a Box with EN with the context menu/ Box with EN or the command "Insert"/"Box with EN" from the main menu.

When a Box with EN is inserted, an input terminal with a symbol EN is displayed. The type of EN input is BOOL. The operation will be executed only when the input EN is TRUE, as shown in figure 7-4-5.

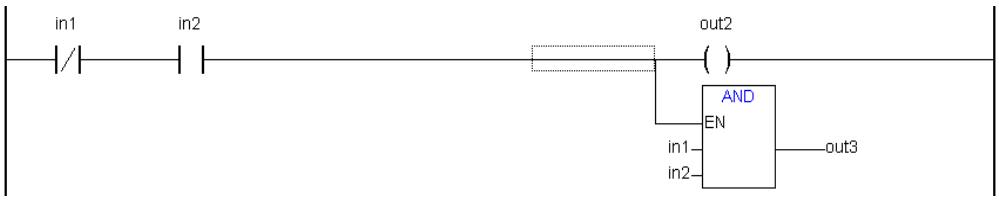


Figure 7-4-5 Insert Box with EN

The newly inserted block contains initially the label "AND". If you wish, you can change this label to another one, such as "MOVE". For this you can also use the Input Assistant. Select the operator keyword and press F2 or use the command "Edit"/"Input Assistant" to select the desired operator from the Help Manager, as shown in figure 7-4-6.

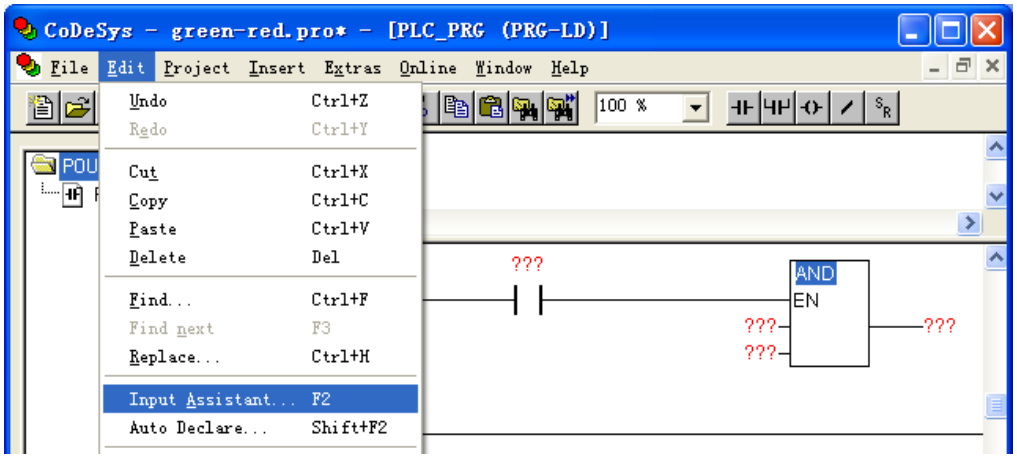


Figure 7-4-6 Input Assistant

Calling Function Block

The instructions including timer, counter, trigger, communication, and high-speed input/output, analog monitors are called in the form of function blocks.

Before calling an instruction in the form of a function block, first you need to understand what a library is. In PowerPro, the commonly used instructions are gathered to create specific libraries. If you want to use an instruction, first you have to add the library in which the instruction is contained. About the concept and usage of libraries, please refer to section 7.4.4.

An instruction can be called in the form of a function block if the corresponding library has been added to the project. With the command “Insert”/ “Function Block” or the “Function Block” in context menu a dialog box is displayed and you can select the instruction needed, as shown in figure 7-4-7.

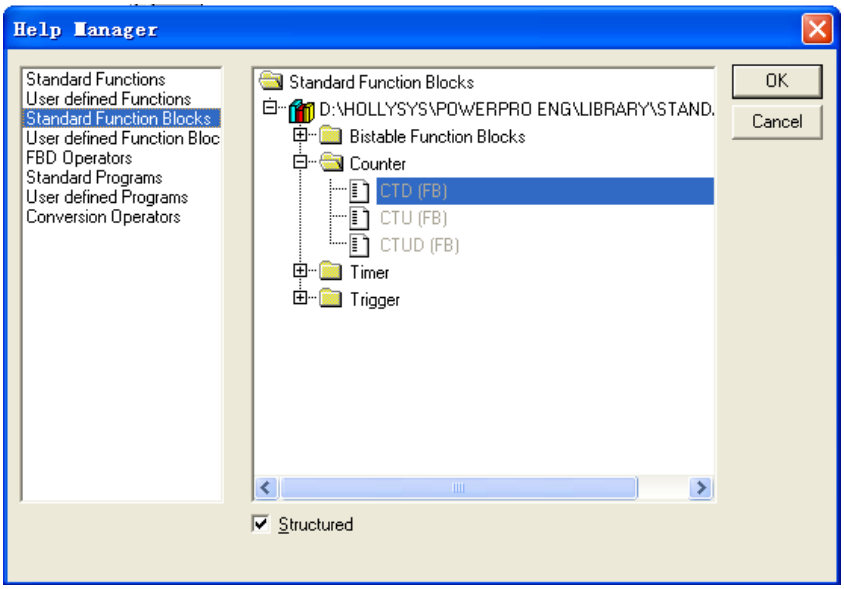


Figure 7-4-7 Insert Function Block

In applications the concepts of “Function Block” and “Box with EN” are often mixed up. Actually the distinction is obvious and their calling modes are different. For “Function Block” with EN, it will be executed when the program runs no matter whether it is enabled or not, while the “Box with EN” will be called only when the EN is enabled.

Note:
When using function blocks instructions, instances are declared as variables whereas the name of the function block is indicated by the type of an identifier. If the instruction is used many times in a program the variable declarations should be different.

7.4.4 Additional Libraries

In PLC programming some instructions or function blocks are frequently used, such as string functions, triggers, counters, PIDs, and so on. In PowerPro, commonly used instructions and function blocks are gathered and classified to create libraries.

A library is the collection of instructions and function blocks and each library contains a file “library name.lib” (including input and output code of instructions and function blocks) . If you want to call the instructions or function blocks, add the library file “library name .lib”.

Commonly used libraries include Standard.lib, Utility libraries (Util.lib, Util_no_Real.lib) , and System libraries (SysLibC16x.lib, SyslibCallBack.lib) . The standard library and the system libraries are added to the project automatically when creating a new project and can be called directly, whereas other libraries can be called only after they are added to the project manually. All the library files provided by the system are stored in the directory \Hollysys\PowerPro\Library in the form of “*.lib”.

Library Manager

Library Manager is used to manage library functions and function blocks and it contains all standard functions and function blocks. Library Manager is opened with the command “Window”/“Library Manager”, as shown in figure 7-4-8.

The library manager shows all libraries that are connected with the current project. The POUs, function blocks, functions, data types, and variables of the libraries can be used the same way as user-defined POUs, function blocks, functions, data types, and variables.

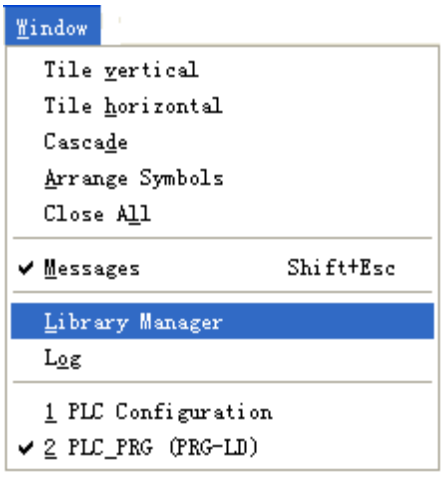


Figure 7-4-8 Open Library Manager

The window of the library manager is divided into four areas, as shown in figure 7-4-9. The libraries attached to the project are listed in the upper left area. In the area below that there is a listing of the POUs, Data types, Visualizations or Global variables of the library selected in the upper area. If it is a function or function block, the declaration will appear in the upper right area of the library manager and in the lower right there is the graphic display.

It is very important to know how to view function libraries. In the function libraries, some very important messages are displayed, such as the number of input variables and intermediate variables and their types, variable comments and the intermediate variables which must be initialized.

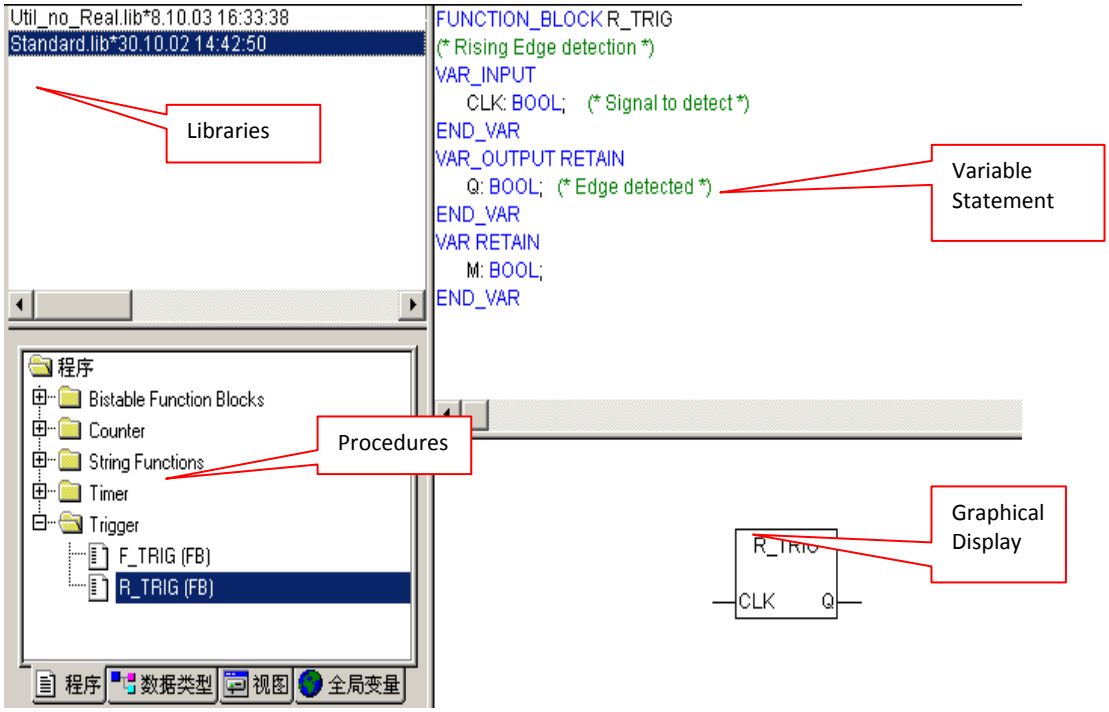


Figure 7-4-9 Window of Library Manager

Additional Libraries

Additional libraries are needed when the library files in library list cannot meet the programming requirements. Because there is a conflict in the data types in Util.lib and Util_no_Real.lib and errors will appear while compiling, so they are not allowed to be added at the same time. When using the library, ensure the library files are stored in the following directory: PowerPro installation directory\Library\. Use the command “Insert”/“Additional Library” or the “Additional Library” in the context menu in the library list of the library manager window to add an additional library to your project, as shown in figure 7-4-10.

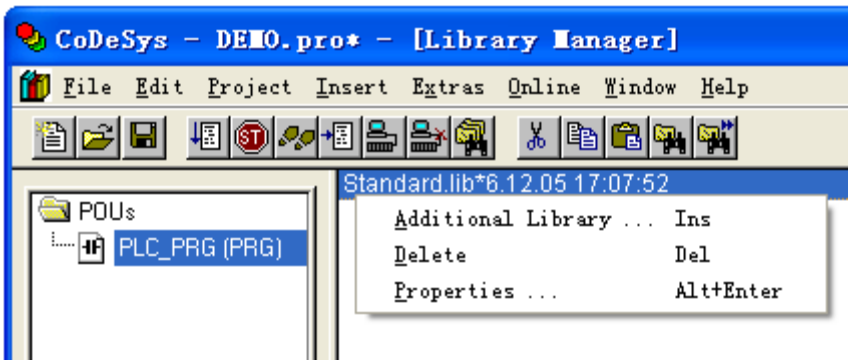


Figure 7-4-10 Additional Library

Select what you need and click “Open”. No matter what kind of library it is, only open the corresponding *.lib file, as shown in figure 7-4-11.

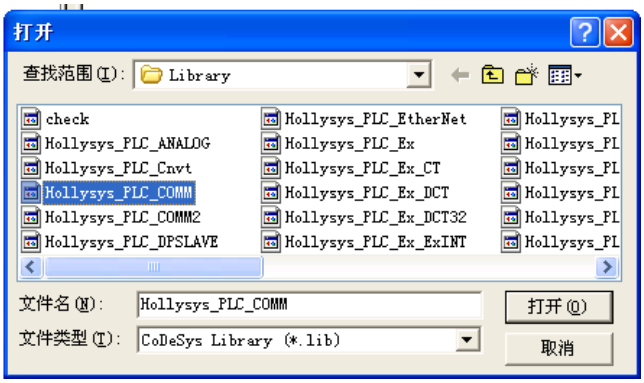


Figure 7-4-11 Open Library

The library selected above is added to the library list, as shown in figure 7-4-12. Note that all the libraries added to the library manager will occupy some space of the user program, so it is suggested to only add the library needed.

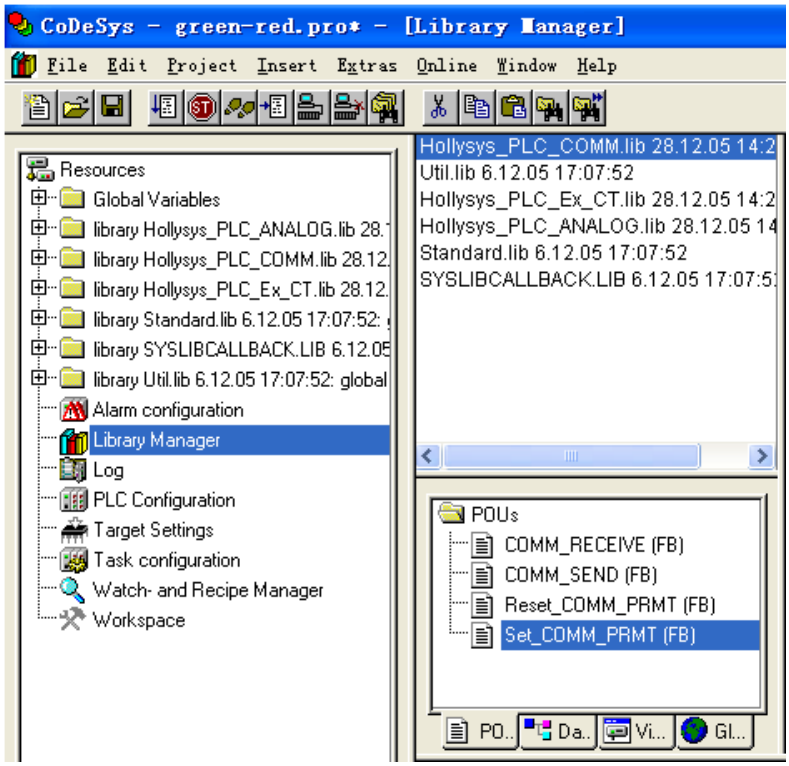


Figure 7-4-12 Display Library

Delete Libraries

With the command “Edit”/“Delete” or “Delete” in the context menu, you can delete existing libraries.

Note:
After creating a new project, the Standard.lib and SyslibCallBack.lib are added automatically, and other libraries need to be added manually.

7.4.5 Creating a Library

In project implementations, some algorithms or logics are used repeatedly and can be used in many other projects. These logics can be put in a library for the convenience of programming, engineering maintenance and technological resources sharing. Passwords are used to protect your algorithms and logics in a library against illegal usage or modifications. Here is how to create a library:

First, create a new project, and select “None” in “Target Settings”/“Configuration”, as shown in figure 7-4-13.

Figure 7-4-13 User-defined Library (1)

Click “OK”, and a “New POU” window appears, as shown in figure 7-4-14. Select “Function Block” in the field “Type of POU”. Enter the name in the field “Name of the new POU”, and generally the name can show what it used for. In figure 7-4-14, select “ST” as the “Language of the POU”, and enter the name “Generate_CRC” as the “Name of the new POU”.

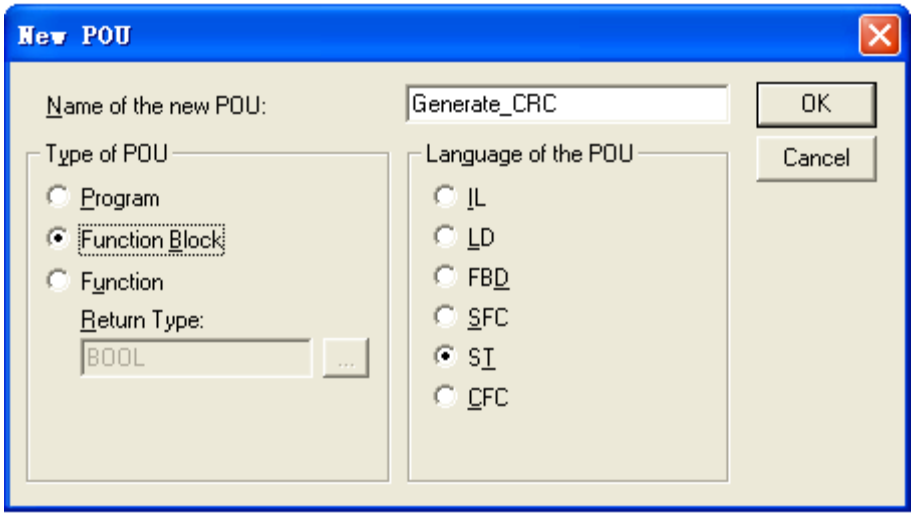


Figure 7-4-14 User-defined Library (2)

After the configuration of a function block structure is generated, save it as a library and add content needed to it. Remember to save it under the directory PowerPro\Library and give it a name according to requirements and save it as an “Internal Library” for algorithm implementations inside the library, as shown in figure 7-4-15. Another type is “External Library” of which the algorithms and logics are implemented in modules and only variable declarations are inside the library.

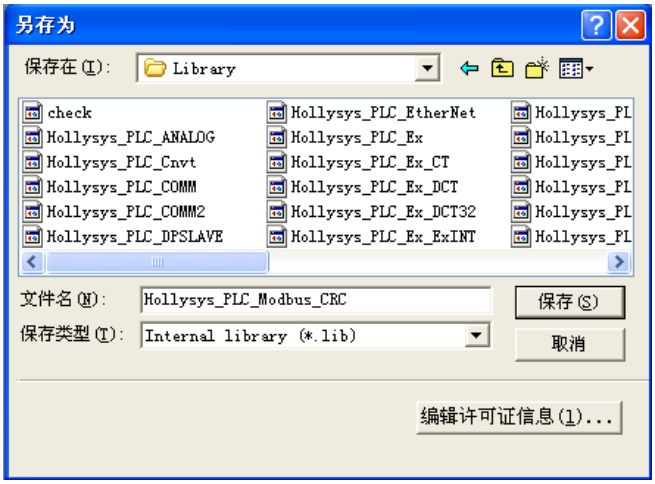


Figure 7-4-15 User-defined Library (3)

Example

The example is used to generate a library for CRC16 check written in ST language, as shown in figure 7-4-16.

After compilation the library can be used in other projects. Remember to add it to library manager before using.

If you want to use it on other computers, you have to copy and paste it to the software installation directory PowerPro\Library on other computers and add it to library manager.

A library can contain a number of function blocks or functions. If you want to add a new function block or function in the library, you have to open the library file and add new function blocks or functions in the program.

Regard that the function block in library can't be recursively called, in another words, it cannot be called by itself.

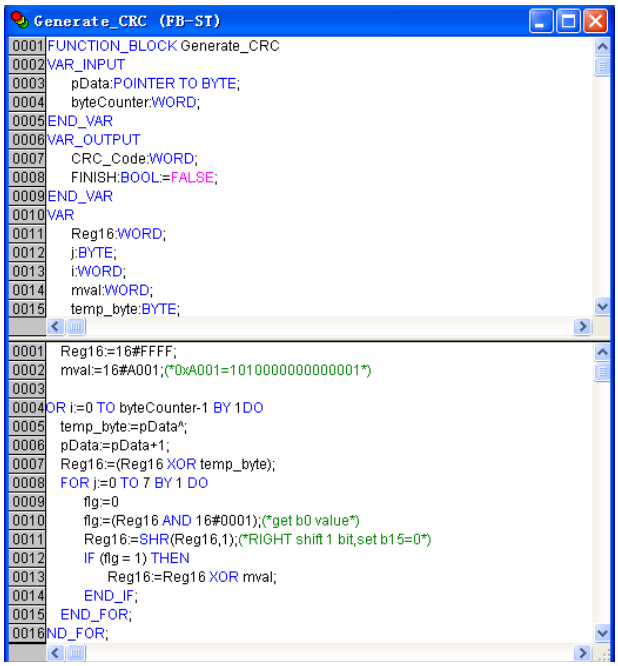


Figure 7-4-16 User-defined Library (4)

For user-defined library, it can be protected from being changed. The encryption principle is the same as that of programs. Select “Resources”/ “Workspace”/ “Passwords” and set the passwords to protect the library from being changed, as shown in figure 7-4-17.

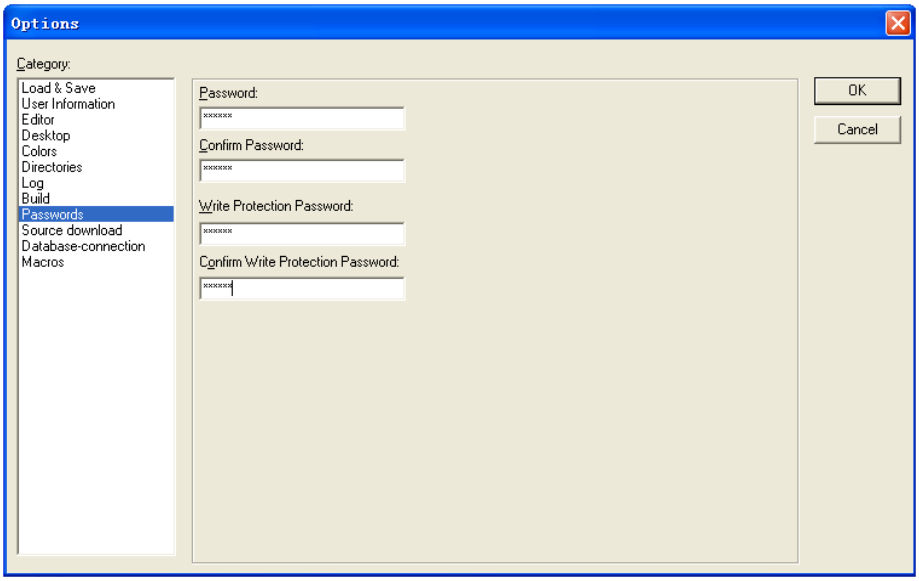


Figure 7-4-17 User-defined Library Encryption

7.4.6 Jump and Return

Jump and return will change the program scanning sequence and normally PLC will scan according to the network sequence in main program.

Jump: Jump to the assigned network when meeting the jump condition.

With the command “Insert”/“Jump” or “Jump” in the context menu insert a jump, as shown in figure 7-4-18.

For an inserted jump, enter a jump label (default is “Label”) to which it is to be assigned. Jump directly to network 3 without implementing network 2 when meeting the jump condition, as shown in figure 7-4-18.

Each network has a label that can optionally be left empty. This label is edited by clicking the first line of the network, directly next to the network number. Now you can enter a label.

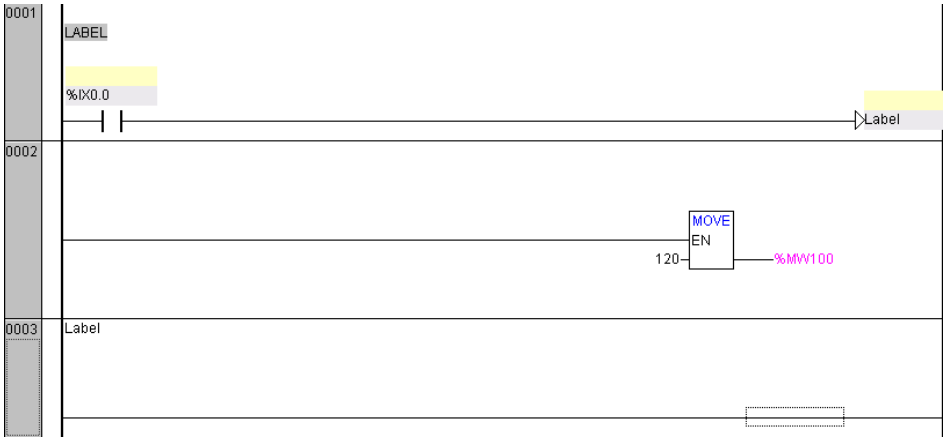


Figure 7-4-18 Jump

Return: when return condition arrives, the called POU will not be executed and return to the calling POU.

With the command “Insert”/“Return” or return in the context menu to insert a return, as shown in figure 7-4-19.

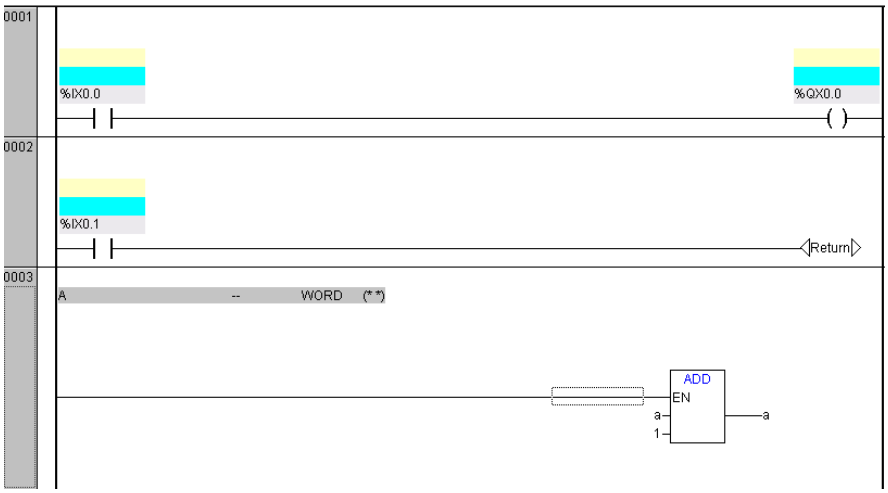


Figure 7-4-19 Return

7.4.7 Call Subprogram

When the program is complicated then a lot of subprograms are needed. We have introduced many times in the previous chapters that in PowerPro the default main program is “PLC_PRG” and the others are subprograms.

Before calling a subprogram you have to create a subprogram. Refer to 5.2 about how to create a subprogram. After the subprogram is created, in the main program call a “Box with EN” and change the keyword to the name of the subprogram, as shown in figure 7-4-20. Refer to section 7.4.3 about how to call a “Box with EN”.

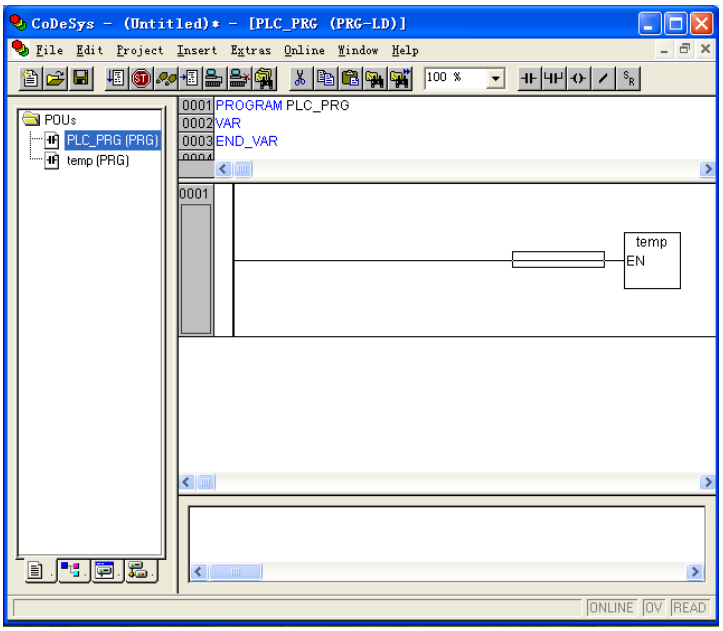


Figure 7-4-20 Call Subprogram

If you want to transfer parameters between the main program and subprogram, you have to declare some input variables, output variable or global variables. About variables, see section 4.4.

7.4.8 Adding Comments

In order to make it easier to read the programs, comments can be added to programs, networks or variables and addresses. There are a number of possibilities to add comments in PowerPro.

Comments for programs and networks

Comments for programs are the same as that for networks. In PowerPro, comments can be entered for each network, as shown in figure 7-4-21. Refer to section 7.4.1 for more details about comments.

Comments for variables

In PowerPro you can add comments for variables. At the declaration of a variable, you can add comments in the dialog box of variable declaration or in the declaration part in the editor, as shown in figure 7-4-21.

Comments for addresses

If the addresses in I, Q or M are used, you can add comments for these addresses. Select “Extras”/ “Options” and activate option “Comments per Contact” and click “Apply options” to exit the dialog box, then you can add comments for these addresses, as shown in figure 7-4-21. See section 7.4.9 about ladder diagram options.

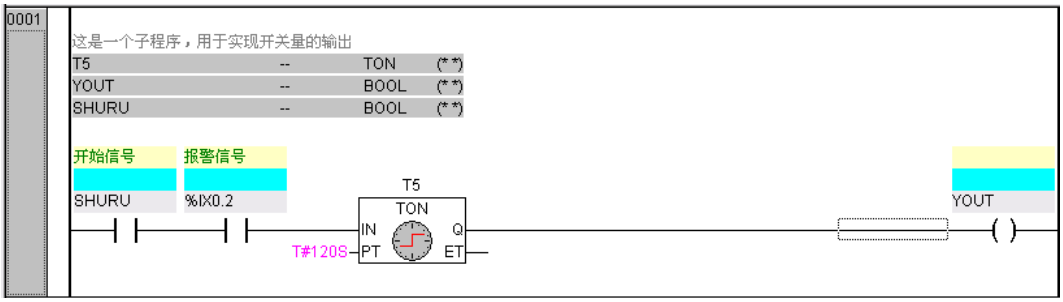


Figure 7-4-21 Comments

7.4.9 Ladder Diagram Options

Select “Extras”/“Options”, a dialog box will appear as shown in figure 7-4-22, and you can set different options for your ladder diagram.

Note:
Adding comments for networks is different from adding jump labels mentioned in section 7.4.6. If you want to add comments for networks, you must use the command “Insert”/ “Comments” or “Comments” in the context menu.

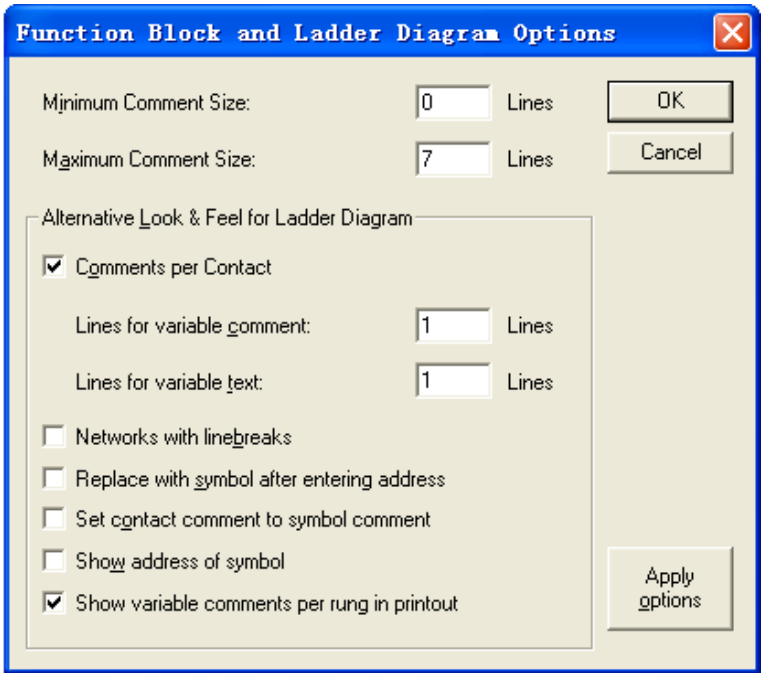


Figure 7-4-22 Ladder Diagram Options

Maximum Comment Size and Minimum Comment Size are used to assign the number of lines for a network comment.

Comments per Contact is used to assign Lines for variable comment and Lines for variable text.

Networks with linebreaks: Linebreaks will be forced in the networks as soon as the network length exceeds the given window size so that all contacts, coils and instructions can be displayed in one window size.

Replace with symbol after entering address: If this option is activated, you can enter an address at a box at a contact or coil and this address will be replaced immediately by the name of the variable assigned to the address.

Set contact comment to symbol comment: the comment of a contact will change into the comment of symbol comment.

Show address of symbol: If a variable is assigned to an address, the address will be displayed above the variable name when you enter the variable name.

Show variable comments per rung in printout: Each network for each variable used in that network there will be displayed a line showing the name, address, data type and comment for this variable, as defined in the variables declaration.

7.4.10 Save Files

Use the command “File”/“Save” in the main menu or the button “  ” in tool bar to save the current project.

Enter the name of the new project in the “File name” and it is suggested to use meaningful characters and numbers.

Select “*.pro” in “Save as type” field and the project file will be saved in the default directory \Hollysys\PowerPro\Projects, as shown in figure 7-4-23.

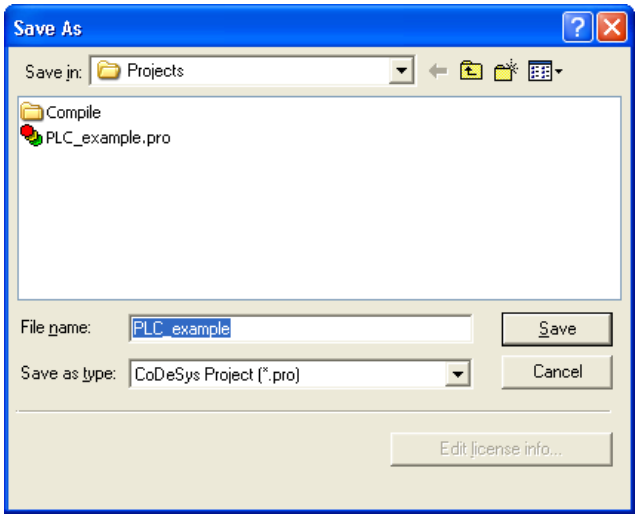


Figure 7-4-23 Save Project (1)

After the project is saved, the "(Untitled)*" in the upper left corner of the main window will change into the project file name, as shown in figure 7-4-24. During the process of creating a project, remember to save the project from time to time to avoid operations mistakes or data loss. When the project has been changed but not saved yet, a "*" will appear after the project name in the upper left corner and it will disappear when the project is saved.

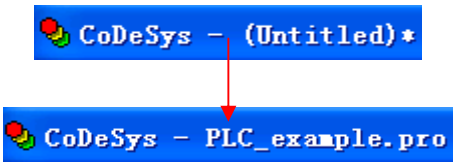


Figure 7-4-24 Save Project (2)

With the "Project"/"Options" menu or "Workspace" in "Resources" tab, you can set the system attributes. With the command "Project"/"Options" the dialog box for setting options is opened. Choose the desired category on the left side of the dialog box by means of a mouse click or using the arrow keys and change the options on the right side. The changes amongst other things serve to configure the view of the main window. They are saved in the file "CoDeSys.ini" and restored at the next startup.

7.5 MENU FOR MANAGING PROJECTS

In PowerPro the user programs are saved in the form of project file and all the information is saved in the project file with the extension "*.pro". In PowerPro the default software installation directory is shown in figure 7-5-1, where "Library" and "Projects" are used to store library file and project file respectively. A large number of operations of project are provided by system and the users can use them to manage the project better.

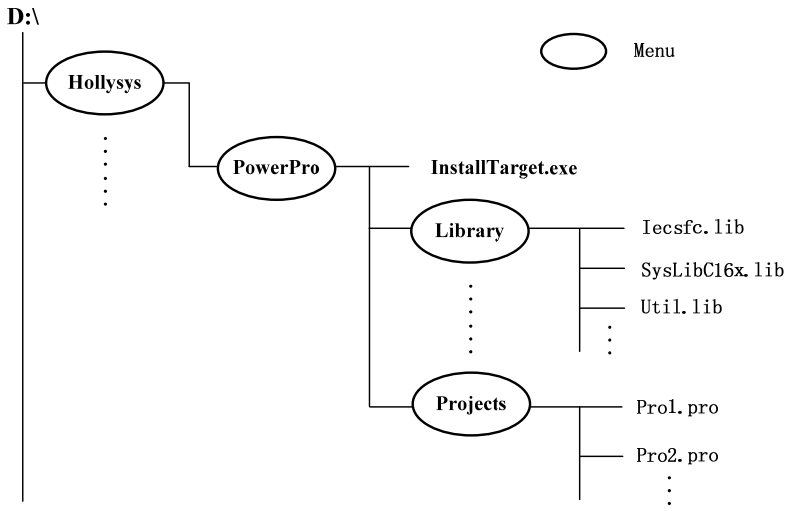


Figure 7-5-1 Software Directory Tree

Open the “Project” menu in the main window, as shown in figure 7-5-2.

Figure 7-5-2 Manage Project Menu

The commonly used commands for managing project are described below.

7.5.1 Printing Documentation of a Project

The command “Project”/“Document” lets you print the documentation of your entire project or part of it. Document consists of Project information, Contents of Documentation, POUs, and Resources, as shown in figure 7-5-3. Resources include Global Variables, PLC Configurations, Alarm configurations, Workspaces, Watch- and Recipe Manager, Task configurations and Parameter Managers.

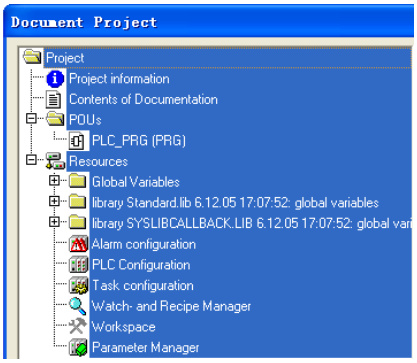


Figure 7-5-3 Print Document Project

Only those areas in the dialog box highlighted in blue are printed. If you want to select the entire project, select the name of your project in the first line. If you only want to select a single object, click on the corresponding object. Objects with a plus sign in front of their symbols are organization objects which contain other objects. With a click on a plus sign, the organization object will be expanded, and with a click on the minus sign that appears, it will be callapsed again. When you select an organization object, all relevant objects are also selected. By pressing the <Shift> key you can select a group of objects, and by pressing the <Ctrl> key you can select several individual objects. Once you have made your selection and then click on OK. The Print dialog box appears. You can determine the layout of the pages to be printed with “File”/“Printer Setup”, as shown in figure 7-5-4.

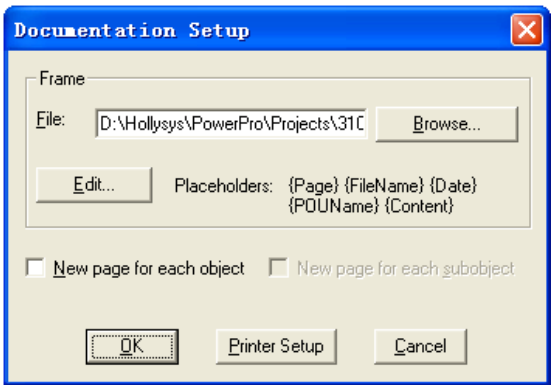


Figure 7-5-4 Documentation Setup Dialog Box

File

In the field File you can enter the name of the file with the extension “.dfr” in which the page layout should be saved. The default destination for the settings is the file DEFAULT.DFR.

Browse

If you would like to change an existing layout, then browse through the directory tree to find the desired file with the button Browse.

Edit

If you click on the “Edit” button, the frame for setting up the page layout will appear. With the command “Insert”/“Placeholder”, you can insert Pages, POU-Names, Filenames, Dates and Content by dragging a rectangle the layout.

New page for each object

If the option is activated, a new page will be started for each object.

New page for each sub-object

If the option is activated, a new page will be started for each sub-object.

Printer Setup

Use the “Printer Setup” button to open the printer configuration. With the button “Properties” you open the dialog box to set up the printer.

If you click on the Edit button, the frame for setting up the page layout will appear, as shown in figure 7-5-5. With the menu item “Insert” “Placeholder” and subsequent selection

among the five placeholders (Page, POU name, File name, Date, and Content), insert into the layout a so-called placeholder by dragging a rectangle the layout while pressing the left mouse button. In the printout they are replaced by the current page number, the current name of the POU, the name of the project, the current date, and the contents of the POU respectively. Click “OK” to exit the setup.

If the template was changed, the software will ask if these changes should be saved or not when the window is closed.

In order to be sure that the page format will be valid for printouts, define the layout as described above, and additionally activate the option “Show print area margins” in “Project” “Options” “Desktop”.

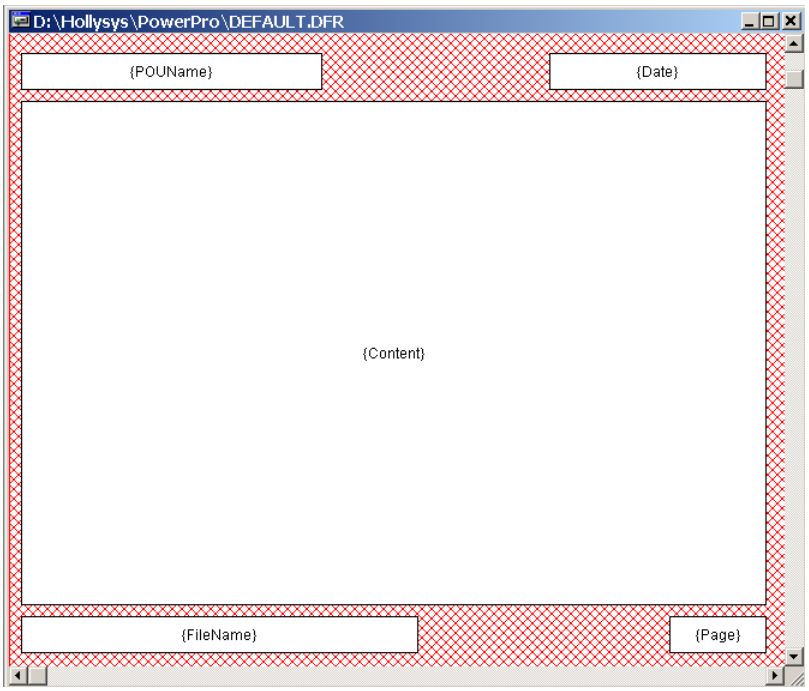


Figure 7-5-5 Page Layout

7.5.2 Importing and Exporting Projects

With the commands “Project”/“Import”, “Export” you can import or export project objects in order to exchange programs between different project files. When export, you can decide whether you want to export the selected parts to one file or to export in separate files, as shown in figure 7-5-6. Switch on or off the option “One file for each object” and then click on OK.

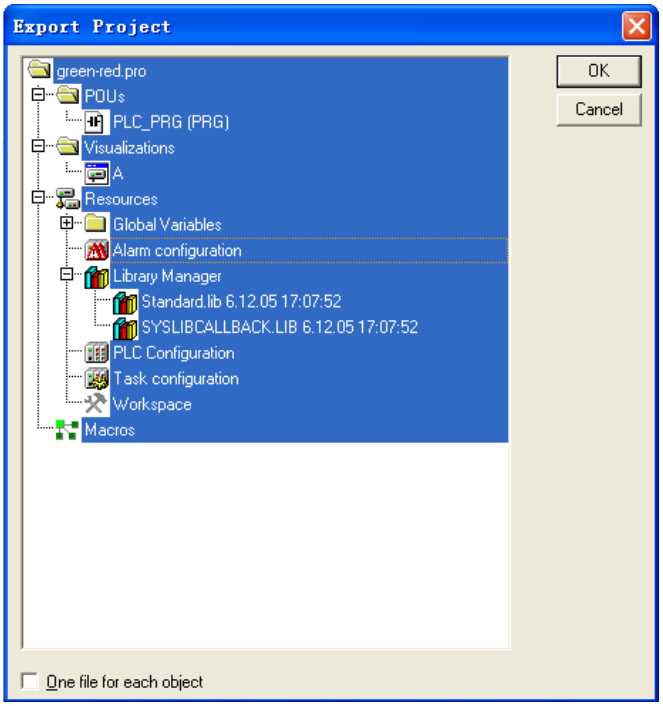


Figure 7-5-6 Export Project (1)

The dialog box for saving files will appear. Enter the name and the file to be saved in the directory with an extension ".exp", and a message window will appear to display related information.

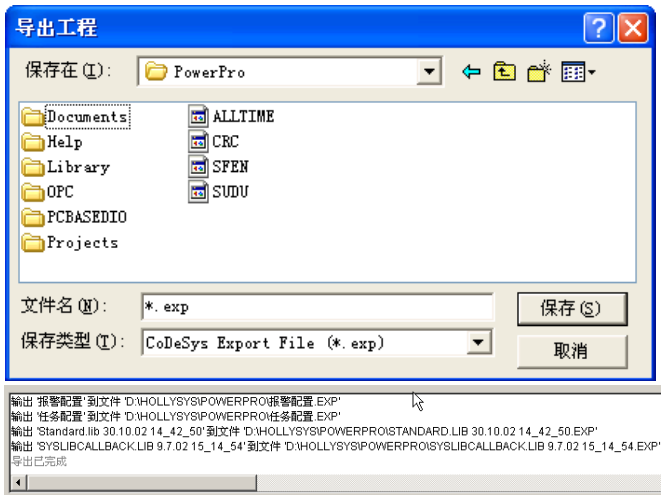


Figure 7-5-7 Export Project (2)

When importing, select the desired export file and the object will be imported into the current project. If an object with the same name already exists in the same project, a dialog box will appear with the question "The object already exists. Do you want to replace it? ". This is as shown in figure 7-5-8.

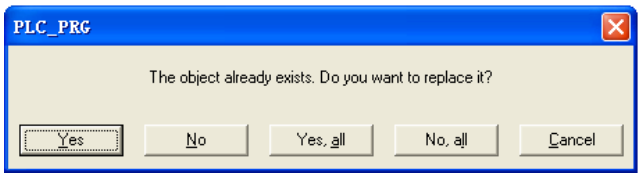


Figure 7-5-8 Tip for Replace

The imported objects and exported results are all saved in *.exp file in order to exchange all objects with different projects, as shown in figure 7-5-9.

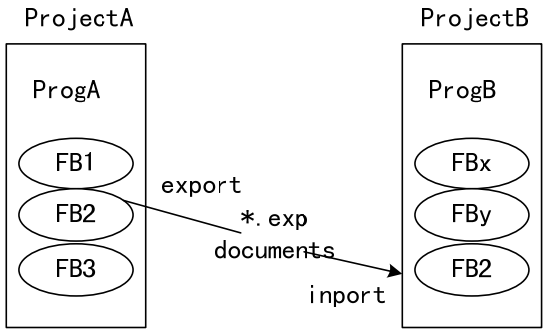


Figure 7-5-9 Schematic Diagram of Import and Export

7.5.3 Merging Projects

With the command “Project”/“Merge” you can merge objects from other projects into the current project. When the command has been given, first the standard dialog box for opening files appears. When you have chosen a file there, a dialog box appears in which you can choose the desired object, as shown in figure 7-5-10.

For the merge of library and resources, a dialog box appears with the question “The object already exists. Do you want to replace it? ”.

For the merge of POUs, the new POUs will be added to the list of original POUs. If an object with the same name already exists in the same project, then a dialog box appears with the question “The object already exists. Do you want to replace it? ”

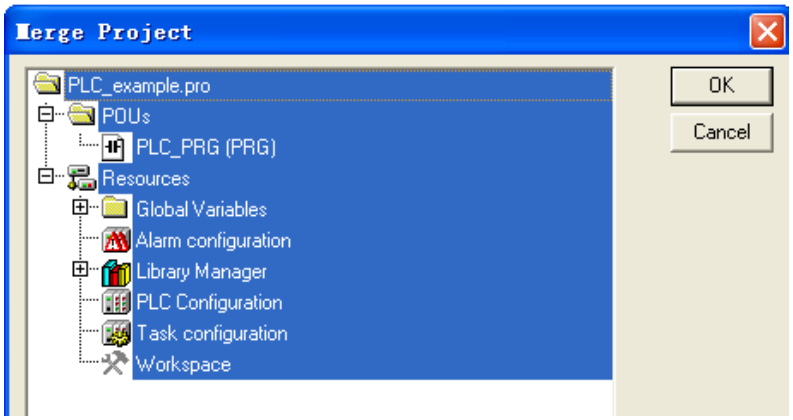


Figure 7-5-10 Merge Project

Note that the system events will not be merged when merge projects.

7.5.4 Comparing Projects

With the command “Project”/“Compare” you can compare two projects. Click the command and a dialog box of project comparison will appear, as in figure 7-5-11.

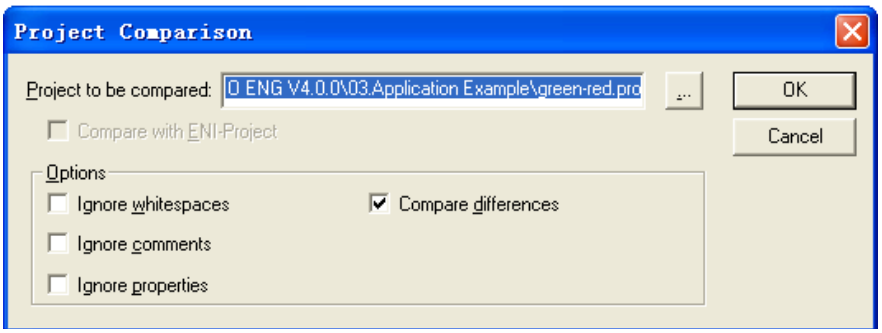


Figure 7-5-11 Compare Projects (1)

When the dialog “Project Comparison” is closed by pressing OK, the comparison will be executed according to the settings, and the results are represented in figure 7-5-12. The five colors stands for different comparison results:

Black: Unit for which no differences have been detected.

Red: Unit has been modified.

Blue: Unit only available in current project.

Green: Unit not available in current project.

Grey: Different objects in two projects and double-click them for details.

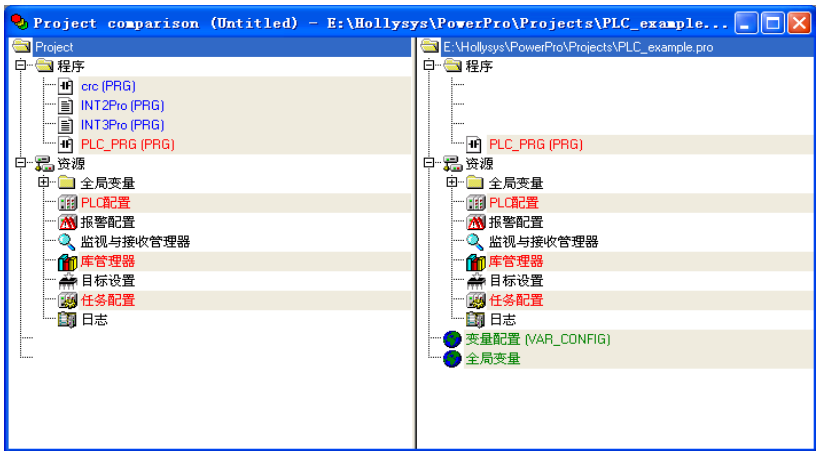


Figure 7-5-12 Compare Projects (2)

In addition, the command “Compare” can be used to compare the actual version of one project with another that was saved previously, as shown in figure 7-5-13. In the compare mode, objects cannot be edited and all operations are disabled until you close the window of project comparison.

However, the comparison of hardware configuration is not supported.

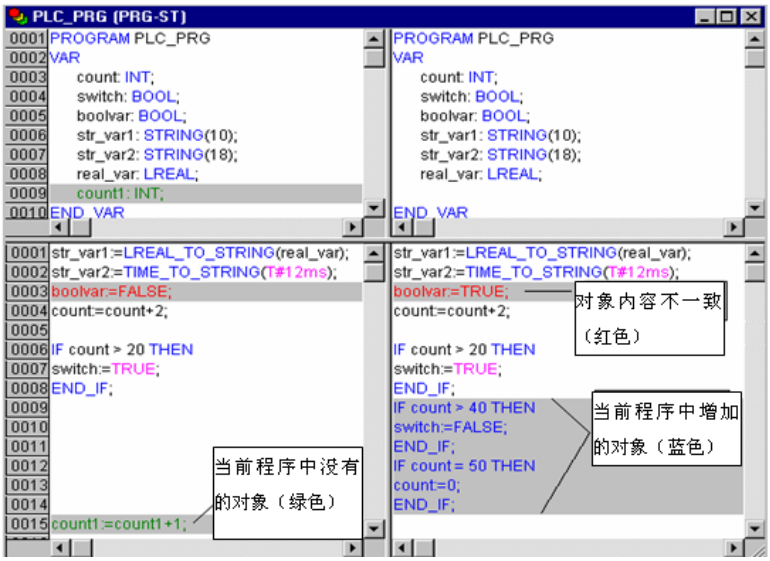


Figure 7-5-13 Compare Projects (3)

7.5.5 User Passwords

With the command “Project”/“Options”/“Passwords” to protect your files from unauthorized access and protect against your files being opened or changed, as shown in figure 7-5-14. You can enter “Password” to open a project, and “Write Protection Password” to change a project.

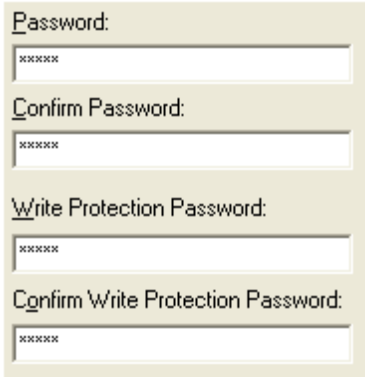


Figure 7-5-14 Options of Passwords

Enter the desired password in the field “Password”. For each character typed an asterisk (*) appears in the field. You must repeat the same password in the field “Confirm Password”. If you now save the file and then reopen it, you get a dialog box to enter the password. If a dialog box appears, as shown in figure 7-5-15, you have to re-enter the password.

Figure 7-5-15 Password Error

A write-protected project can be opened without a password. To do this, simply press the button “Cancel” when you are told to enter the write-protection password when opening a

file. Now you can compile the project, load it into the PLC, and simulate, etc., but you cannot change it.

In order to create differentiated access rights you can define user groups and passwords for user groups. In PowerPro up to eight user groups with different access rights to the project can be set up. The members of each group are recognized by passwords. With the command “Project”/“User Group Passwords” to assign access rights for different user groups. The user groups are numbered from 0 to 7, whereby the Group 0 has the administrator rights, i.e. only members of group 0 may determine passwords and access rights for all groups and objects, as shown in figure 7-5-16.

When a new project is launched, then all passwords are initially empty. Until a password has been set for the group 0, one enters the project automatically as a member of the group 0. In the left combo box “User Group” you can select the group and enter the desired password for the group in the field “Password”, as shown in figure 7-5-16.

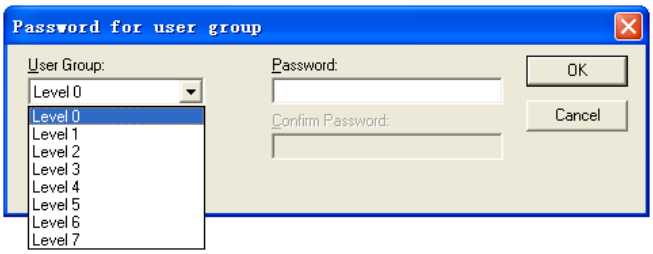


Figure 7-5-16 Password for User Group

With the command “Project”/“Object”/“Properties” you can open the dialog box for assigning access rights to the different user groups, as shown in figure 7-5-17. There are three possible settings:

- No Access: the object may not be opened by a member of the user group
- Read Access: the object can be opened for reading by a member of the user group but not changed.
- Full Access: the object can be opened and changed by a member of the user group.

The settings are effective for the currently-selected object (a POU, hardware configuration or global variable). If you should ever forget a password, then contact the manufacture of your PLC. The passwords are saved with the project.

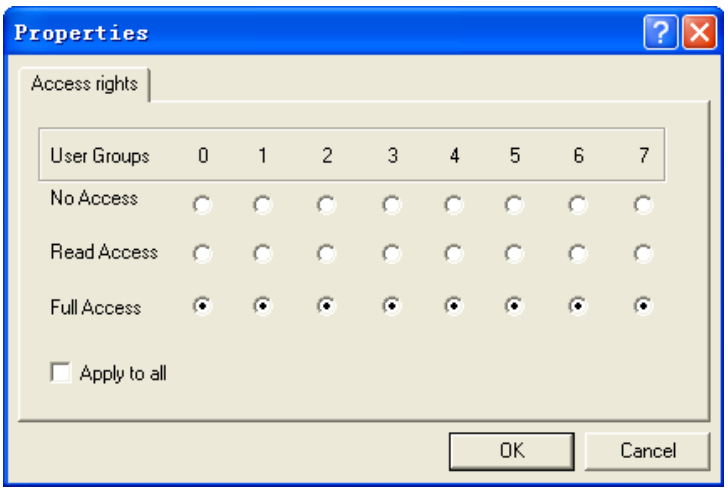


Figure 7-5-17 Properties

A simple example for object access rights is shown below.

With the command “Project”/“Object”/“Properties” open the dialog “Properties”, as shown in figure 7-5-18.

- Group 0 and group 1 have full access.
- Group 2 and group 3 have read access.
- Groups 4, 5, 6 and 7 have no access.
- Set the passwords for each user group after the settings of access rights are finished.

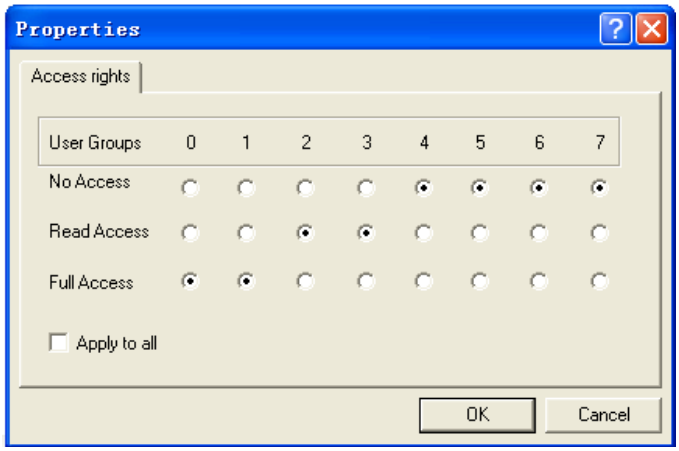


Figure 7-5-18 Access Rights

With the command “Project” “User group passwords” a dialog for password assignment for user groups is opened, as shown in figure 7-5-19.

In the left combo box User group select “Level 0” and enter “0” in the field “Password”. You must repeat the same password in the field Confirm password. Close the dialog box after each password entry with OK.

It is the same for Level1, Level2 and Level3. The relationships between user groups and passwords are shown in table 7-5-1.

There is no need to set passwords for user groups 4, 5, 6 and 7 because they have no access.

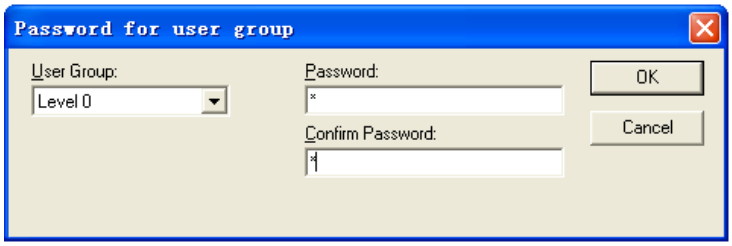


Figure 7-5-19 Password for user group

User Group	Passwords
Level 0	0
Level 1	1
Level 2	2
Level 3	3

Table 7-5-1 User Group and Passwords

If you now save the file and reopen it, you will get a dialog box in which you are requested to enter the password. Because the passwords are different for different user groups, one can enter the project in four different identities and have different operation rights. Now enter the project as “Level0” and enter the password, as shown in f igure 7-5-20.

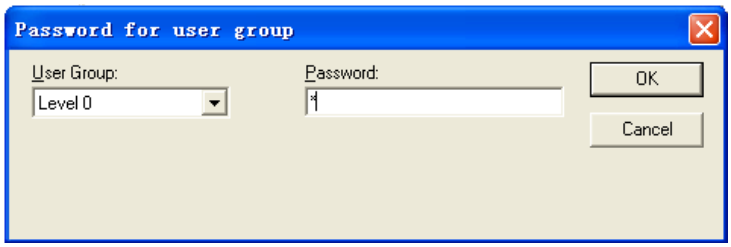


Figure 7-5-20 Enter the Password for user group

If the password does not agree with the saved password of Level 0, then the message appears, as shown in figure 7-5-21:

Figure 7-5-21 Password Error

The project can be opened only when you have entered the correct password of Level 0. For user groups Level4, Level5, Level6, Level7 and Level8 who have no access, the following dialog will appear, as shown in figure 7-5-22:

Figure 7-5-22 No Access to the Object

7.6 WORKSPACE SETTINGS

You can open a “Workspace” in two ways: the first is to click “Project”/“Options”; the second is to double click “Workspace” in the “Resources” tab, as shown in figure 7-6-1.

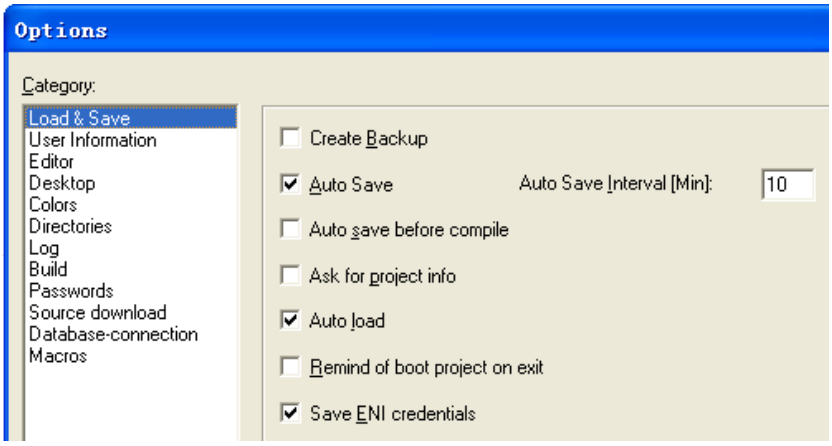


Figure 7-6-1 Workspace Options

Open the dialog box of “Workspace”. The options are divided into different categories. Choose the desired category on the left side of the dialog box by means clicking or using the arrow keys and change the options on the right side.

- Load & Save
- User Information
- Editor
- Desktop
- Colors
- Directories
- Log
- Build
- Passwords
- Source download
- Symbol configuration
- Database-connection
- Macros

PLC can’t support “Source download”, “Symbol configuration”, “Database-connection” and “Macros” and here they are ignored.

In the following we will introduce “Load&Save”, “User Information”, “Editor”, “Desktop”, “Color”, “Directories”, “Log”, “Build” and “Passwords”, and the rest keep default settings.

7.6.1 Load&Save

In “Workspace” select “Load&Save” and the options are displayed on the right of the window, as shown in figure 7-6-1. The options of “Load&Save” are:

“Create Backup”: PowerPro creates a backup file at every save with the extension “.bak”. Contrary to the *.asd-file (see below, “Auto Save”) this *.bak-file is kept after closing the project. So you can restore the version you had before the last project save.

"Auto Save": While you are working, your project is saved according to a defined time interval (Auto Save Interval [Min]) to a temporary file with the extension ".asd". This file is erased at a normal exit from the program. If for any reason PowerPro is not shut down "normally" (e.g. due to a power failure), the file will not get erased. When you open the file again, the following message will appear, as shown in figure 7-6-2. You can now decide whether you want to open the original file or the auto save file.

Figure 7-6-2 Auto Save Backup

"Auto save before compile": The project will be saved before each compilation. In doing so a file with the extension ".asd" will be created, which behaves like described above for the option "Auto Save".

"Ask for project info": When saving a new project or saving a project under a new name, the project info is automatically called. You can visualize the project info with the command "Project" "Project Info" and also process it. "Project Info" includes "File name", "Directory", "Author", and "Version", which are part of the project documentation and can be printed.

"Auto load": At the next start of PowerPro the last open project is automatically loaded.

"Remind of boot project on exit": If the project has been modified and downloaded without creating a new boot project since the last download of a boot project, then a dialog will advise the user before leaving the project: "No boot project created since last download. Exit anyway? ", as shown in figure 7-6-3. The so-called boot project is a user program saved in PLC flash and runs after electrify.

"Save ENI credentials": Save ENI credentials.

Figure 7-6-3 A Dialog Box of "No boot project created since last download. Exit anyway?"

7.6.2 User Information

In "Workspace", select "User Information", the options appear at the right of window. The options are shown in figure 7-6-4. Enter user name, initials and company in corresponding fields.

Figure 7-6-4 User Information

7.6.3 Editor

In “Workspace”, select “Editor”, the options will appear on the right of the window. The options of Editor include Autodeclaration, Autoformat, List components, Declarations as tables, Mark, Bit values and so on, as shown in figure 7-6-5.

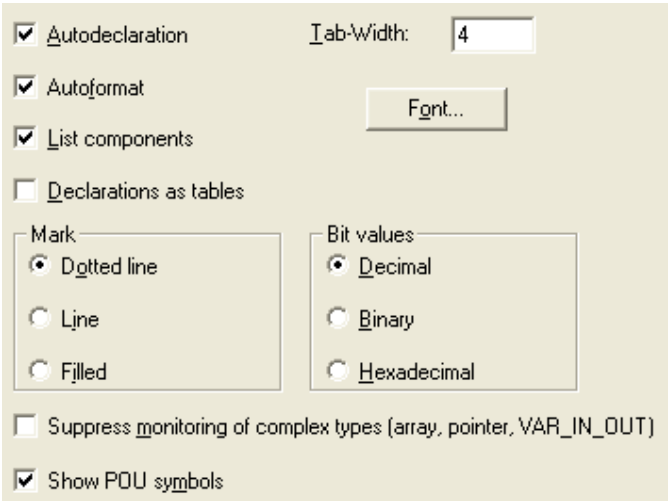


Figure 7-6-5 Editor Options

“Autodeclaration”: If this option is activated, then after the input of a not-yet-declared variable a dialog box will appear in all editors with which this variable can be declared.

“Autoformat”: If this option is activated, then PowerPro executes automatic formatting in the editors. When you have finished with a line, the following formatting is made: 1.Operator written in small letters is shown in capitals; ˜Tabs are inserted to that the columns are uniformly divided.

“Declarations as tables”: If this option is activated, you can edit variables in a table. This table is sorted like a card box, where you can find an INFO card and six variable cards including VAR, VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, CONSTANT and RETAIN. For each variable there are edit fields to insert Name, Address, Type, Initial and Comment. INFO card will display the type and name of POUs automatically.

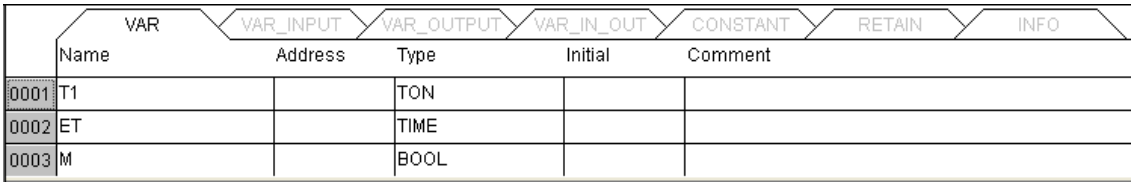


Figure 7-6-6 Declarations as Tables

"Tab-Width": In the field Tab-Width you can determine the width of a tab as shown in the editors. The default setting is four characters, whereby the character width depends on the font used.

"Font": By clicking on the button Font you can choose the font in all editors. The font size is the basic unit for all drawing operations. Choosing a larger font size enlarges the printout in every editor.

"Mark": You can choose whether the current selection in your graphic editors should be represented by a dotted rectangle (Dotted), a rectangle with continuous lines (Line) or by a filled-in rectangle (Filled).

"Bit Values": You can choose whether variables (type BYTE, WORD, DWORD) in online mode should be shown Decimal, Hexadecimal, or Binary.

Example

Decimal [D]	Binary [B]	Hexadecimal [H]
a=53	a=2#0000 0000 0011 0101	a=16#0035
b=57	b=2#0000 0000 0011 1001	b=16#0039

"Suppress monitoring of complex types (Array, Pointer, VAR_IN_OUT)": PLC cannot support this function.

"Show POU symbols": If this option is activated, in the module boxes which are inserted to a graphic editor, additionally symbols will get displayed, if those are available in the directory "PowerPro/Library". The name of the bitmap-file must be composed of the name of the module and the extension ".bmp".

7.6.4 Desktop

In "Workspace", select "Desktop", the options appear on the right of window, as shown in figure 7-6-7

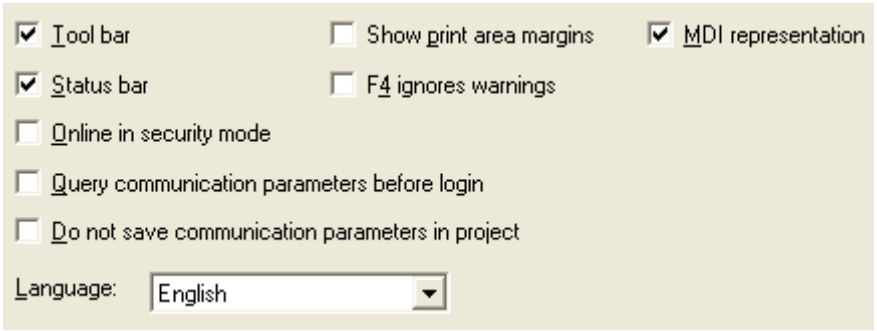


Figure 7-6-7 Desktop Options

"Tool bar": The tool bar with the buttons for faster selection of menu commands becomes visible underneath the menu bar.

"Status bar": The status bar is located at the lower edge of the main window when it becomes visible.

"Online in security mode": In Online mode with the commands 'Run' 'Stop' 'Reset' 'Toggle Breakpoint' 'Single cycle' 'Write values' 'Force values' and 'Release force', a dialog box

appears with the confirmation request whether the command should really be executed. Operating mistake can be avoided by doing this.

“Query communication parameters before login”: As soon as the command ‘Online’ 'Login' is executed, first the communication parameters dialog will open to check whether the parameters are set and set correctly. It is not needed to execute “Online”/“Communication Parameters”.

“Language”: Choose the language for the menus and dialog texts, and the default setting is English.

“Show print area margins”: In every editor window, the limits of the currently set printing range are marked with red dashed lines. Their sizes depend on the printer settings (paper size, orientation) and the size of the “Content” field in the printing layout (menu: “File” “Documentation Settings”), as shown in figure 7-6-8.

“F4 ignores warnings”: After compilation, when F4 is pressed in a message window, the focus jumps only to lines with error messages; warning messages are ignored.

Figure 7-6-8 Show Print Area Margins

7.6.5 Colors

In “Workspace”, select “Colors”, the options will appear on the right of the window, as shown in figure 7-6-9.

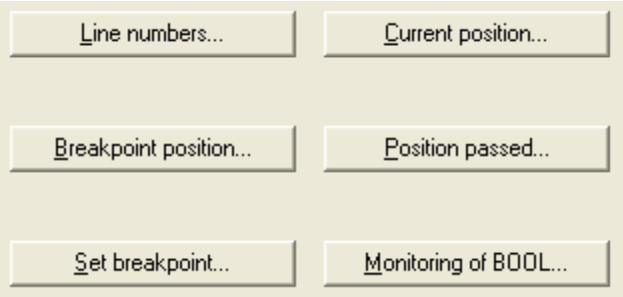


Figure 7-6-9 Colors Options

You can reset some specific display colors according the user’s requirement and customization. According to the colors you can observe the program running easily in online debugging. Click each button in the dialog box of “Colors Options” to open the color setting dialog and select the related color according to specific requirements. Generally the default settings are adopted.

“Line numbers”: default is light grey, background color of network numbers or line numbers in editors.

“Current position”: default is red, in online mode the background color of network numbers or line numbers where the program stops at a breakpoint.

"Breakpoint position": default is dark grey, the background color of network numbers or line numbers where you can set a breakpoint.

"Position passed": default is green, the background color of network numbers or line numbers which have been executed in flow control.

"Set breakpoint": default is light blue, the background color of network numbers or line numbers where you have set a breakpoint.

"Monitoring of BOOL": default is dark blue, in online mode the color of digital logic TRUE.

7.6.6 Directories

In "Workspace", select "Directories", the options will appear on the right of the window, as shown in figure 7-6-10. In "Project" the directories of 'Libraries', 'Compile files', 'Configuration files', 'Visualization files' can be set. In "Target", the directory of "Configuration files" is displayed, and the default directory of "Libraries" is none and is generated automatically and cannot be changed. The directories of automatically generated files when installing PowerPro are displayed in 'Libraries', 'Compile files', 'Upload files', 'Configuration files', 'Visualization files' in "General".

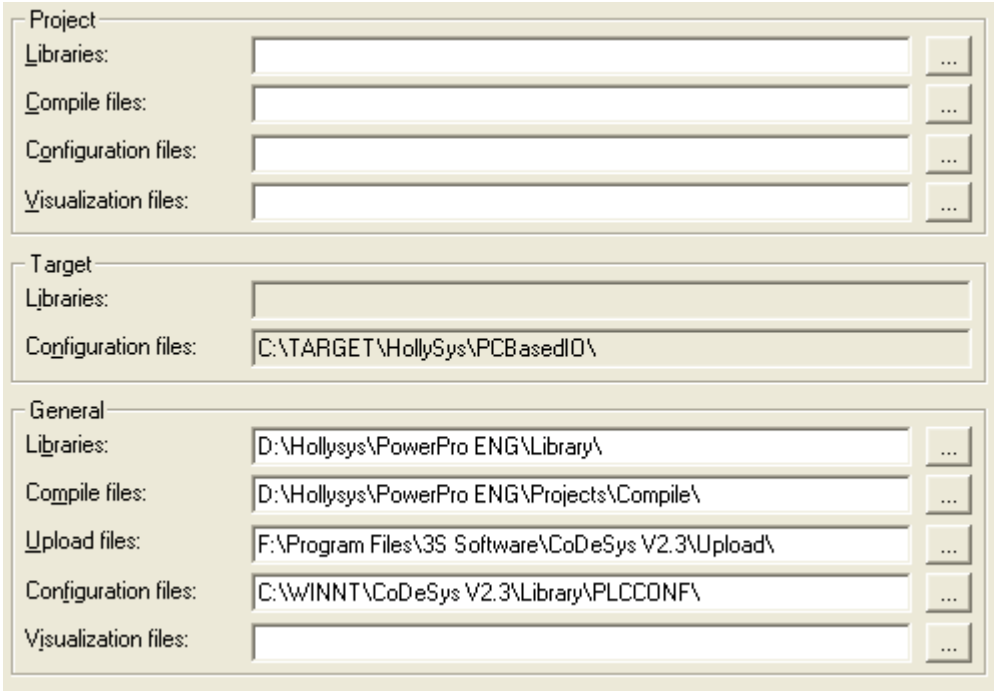


Figure 7-6-10 Directories

7.6.7 Log

The log chronologically records user actions. The recorded contents include Login, Running, Init debugging, Write value, Logout, Delete buffers and login failure. Choose the category "Log" in "Workspace", the options are displayed on the right of the window, as shown in figure 7-6-11.

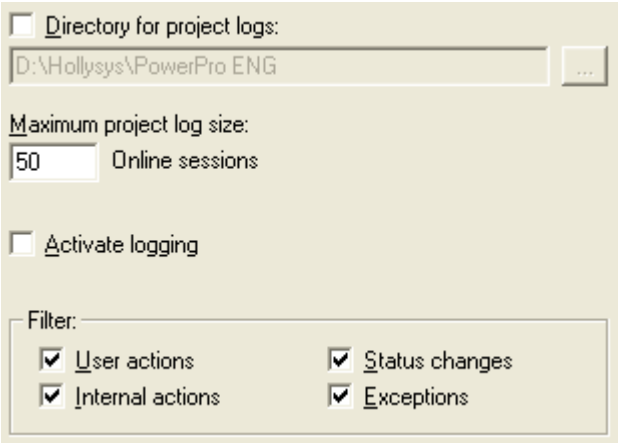


Figure 7-6-11 Log Options

"Directory for project logs": You can change the directory where the project logs are saved and the default setting is D:\Hollysys\PowerPro.

"Maximum project log size": The maximum number of 'Online sessions' as shown in log window.

"Activate logging": Activate log function to display log list, as shown in figure 7-6-12. A serial User actions, Status changes, Internal actions and Exceptions are recorded in the log, such as Login, Running, Init debugging, Write value, Logout, Delete buffers.

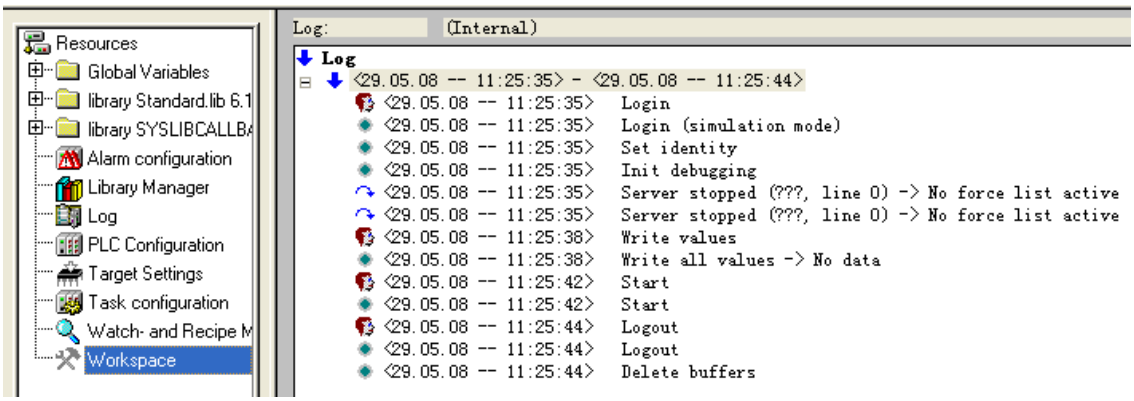


Figure 7-6-12 Log List

"Filter": You can select in the area which actions are to be recorded: User actions, Status changes, Internal actions and Exceptions.

"Log" can be used in either "offline" mode or "online" mode. There is a log list generated in simulation mode, as shown in figure 7-6-12.

7.6.8 Build

In "Workspace", select "Build", the options appear at the right of the window, as shown in figure 7-6-13.

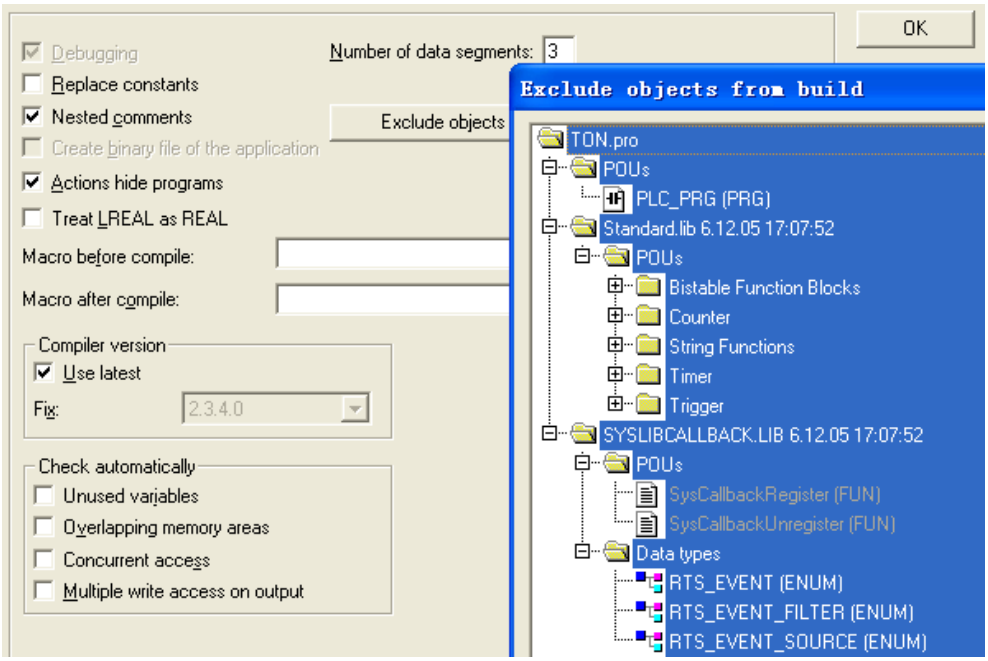


Figure 7-6-13 Build

"Debugging": If it is activated, additional debugging code is created, that is the code can become considerably larger. The debugging code is needed in order to make use of the debugging functions offered by PowerPro (e.g. breakpoints, single step). When you switch off this option, project processing becomes faster and the size of the code decreases.

"Replace constants": The value of each constant is loaded directly, and in online mode the constants are displayed in green. Forcing, writing and monitoring of a constant are no longer possible. If the option is deactivated, the value is loaded into a storage location via variable access.

"Nested comments": Comments can be placed within other comments.

Example: nested comments

```
(*
a:=inst.out;  (*to be checked*)
b:=b+1;
*)
```

"Number of data segments": Here you define how many memory segments should be allocated in the PLC for the project data.

"Compiler version": If you want to get the project compiled with the actual version in any case, activate the option Use latest. If the project should be compiled with a specific version, define this via the selection list at Fix.

"Check automatically":

"Unused variables": Search for variables that have been declared but not used in the program.

"Overlapping memory areas": Test whether during allocation of variables via the "AT" declaration overlaps have occurred at specific memory areas.

"Concurrent access": Search for memory areas of IEC addresses which are referenced in more than one task.

"Multiple write access on output": Search for memory areas to which a single project gains write access at more than one place.

"Check automatically" has the same function with "Project"/"Check". The difference is that "Project /Check" can be used only after compilation one by one, but "Check automatically" can check more than one item while compiling.

When the four items are activated, the check result created automatically while compiling displays in message window, as shown in figure 7-6-14.

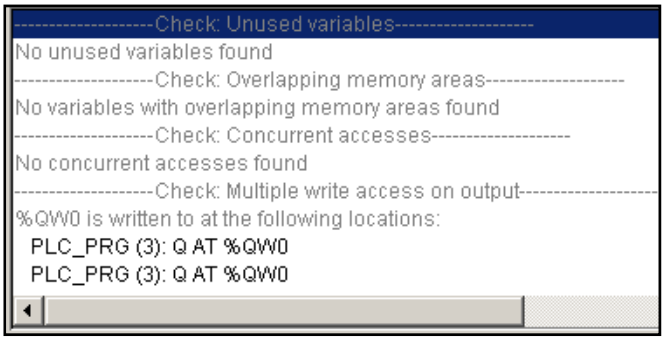


Figure 7-6-14 Compile Information

7.6.9 Passwords

If you want to protect the program from changing, please set passwords for it.

In "Workspace" select "Passwords" and the options are displayed on the right of the window and you can set passwords now. You can set the password whatever you want. It's suggested to set multi-password codes for higher security. In the example in figure 7-6-15 six pieces of code are set and how to use it will be introduced as follows.

Figure 7-6-15 Passwords

If you now save the file and then reopen it, then you get a dialog box in which you are requested to enter the password, as shown in figure 7-6-16.

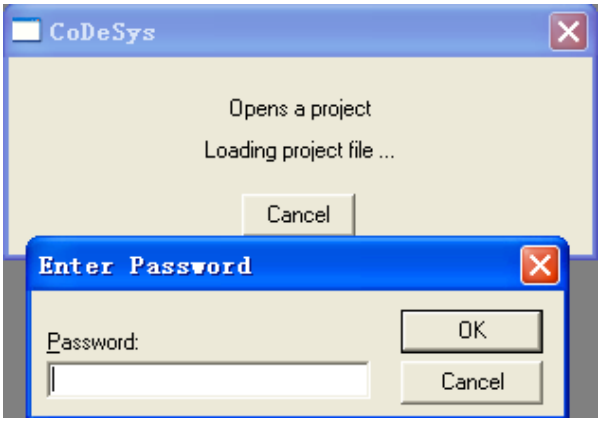


Figure 7-6-16 Enter Password

Click “OK”, a dialog box appears in which you are requested to enter the write protection password, as shown in figure 7-6-17. If the passwords entered are correct, the program is opened and you can edit it.

Figure 7-6-17 Write Protection Password

If you want to cancel the password for a program, you have to open “Workspace”/ “Passwords” to delete the original password and save it. After that, if you open the program again, the password protection will be gone.

Chapter 8

Compiling and Debugging

Compile the program when the program is finished and download it to PLC only when no compilation errors occurred. In this chapter, we will explain the details of the compilation and downloading.

8.1 PROJECT BUILDING

In PowerPro, the commands “Project”/ “Build” and “Rebuild all” are used to check for any syntax errors, as shown in figure 8-1-1.

- Build: Build the program which has changed and renew it to the original target file.
- Rebuild all: Unlike the command “Build”, the project is completely recompiled.
- Clear all: Clear previous compilation and download information.
- Load download information: PLC cannot support the function.

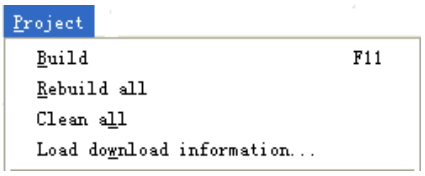


Figure 8-1-1 Project List

A program without syntactic error can be compiled to executable target files. The compilation results will be displayed in a message window, as shown in figure 8-1-2.

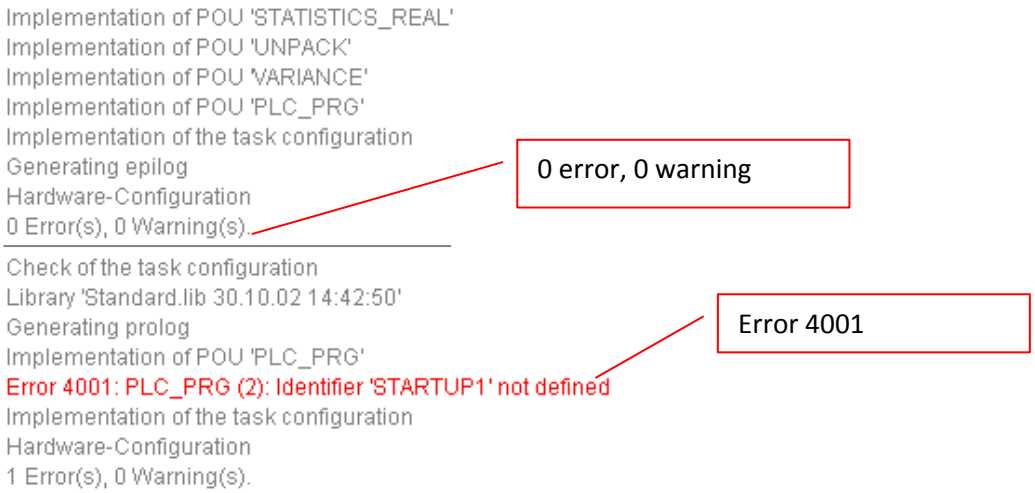


Figure 8-1-2 Compilation Message with errors

8.2 REFERENCES TO DATA TYPES

The “Project” menu in PowerPro provides some commands to show the references to data types. Before this the project must have been compiled without any error.

8.2.1 Displaying Call Tree

With the command “Project”/“Show Call Tree” to show the tree-structure of the POUs, functions and function blocks called by the current object in a new window and to show the calling relationship between the current POU and other POUs in this project, as shown in figure 8-2-1. Before this can be done, the project must have been compiled without any error.

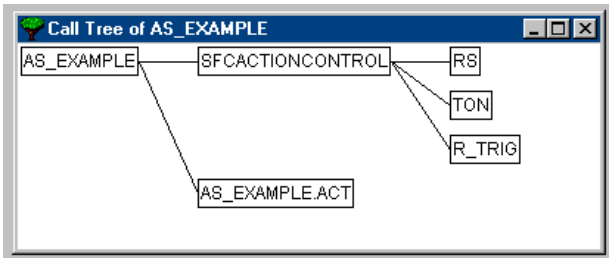


Figure 8-2-1 Show Call Tree

8.2.2 Displaying Cross References

With the command “Project”/“Show Cross References” you can view all application points, as shown in figure 8-2-2. A so-called “application point” is the position where a variable, address or a POU is located. To do this the project must be compiled.

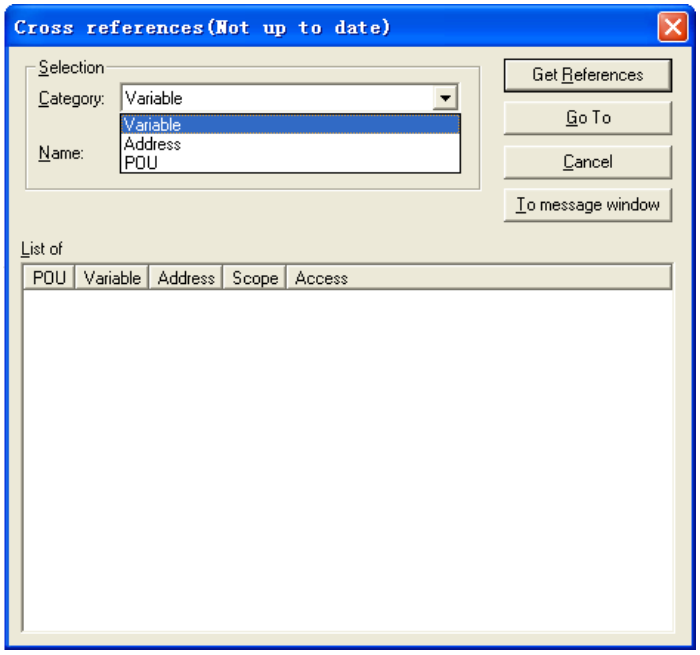


Figure 8-2-2 Show Cross References

Choose first the category ‘Variable’, ‘Address’, or ‘POU’ and then enter the name and click the button “Get References”, then you will get the list of all application points. You can get the information whether the variable is to be accessed for reading or writing, a local or a global variable, connected to hardware address or not.

When you select a line of the cross reference list and press the button Go To or double click on the line, then the POU is shown in its editor at the corresponding point. In this way you can jump to all application points without a time-consuming search. In order to make processing easier, you can use the Send to message window button to bring the current cross reference list into the message window and from there change to the respective POU.

8.2.3 Checking

"Check" can only be used in "Simulation Mode", as shown in figure 8-2-3.

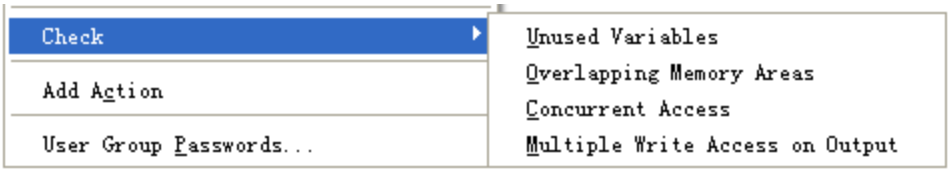


Figure 8-2-3 Check Menu

"Check"/"Unused Variables"

When writing a program, you usually delete a variable or rename it, and then the original variable declaration cannot be deleted automatically and remains in the editor window. So there are variables that have been declared but not used in the program.

The command "Project"/"Check"/"Unused Variables" is used to search for variables that have been declared but not used in the program. Before this can be done the project must have been compiled without any error. Results are displayed in the message window, as shown in figure 8-2-4.

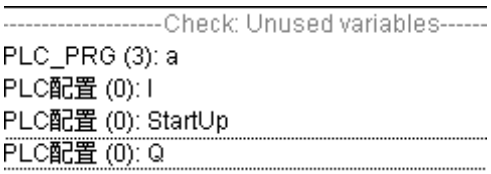


Figure 8-2-4 Check Unused Variables

"Check"/"Overlapping Memory Areas"

This function menu tests whether during allocations of variables via the "AT" declaration overlaps have occurred at specific memory areas. If there is no overlapping memory area, a message window appears with the message: No variables with overlapping memory area found.

"Check"/"Concurrent Access"

This function menu searches for memory areas of IEC addresses which are referenced in more than one task. If there are no concurrent accesses, a tip displays in the message window: No concurrent accesses found.

"Check"/"Multiple Write Access on Output"

This function menu searches for memory areas to which a single project gains write access at more than one place. If there is no multiple write access on output, a message displays in the message window: No outputs found which are written to at more than one location.

8.3 DOWNLOADING

8.3.1 Device Installation and Connection

Installing Devices

First, select appropriate CPU modules and expansion modules according to the actual project. Next, determine the installation mode according to field condition and determine PLC working mode. At last, plan and make a reasonable connection scheme to connect the sensor or actuator to PLC terminal.

Connecting Cables

According to CPU model number and type connect power line, as shown in figure 8-3-1. Don't connect the power supply when the power line is connected. First check whether the cable is connected correctly and then connect the system power supply and ensure that the RUN indicator light on CPU board is on to ensure the PLC can run reliably. Note that when the power line is connected, put on the terminal cover to avoid unnecessary personal injury and device damage.

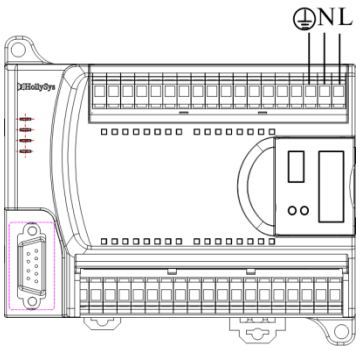


Figure 8-3-1 Connect Power Supply Line

8.3.2 Establishing PC Communication

Establish the communication channel through the programming cable connecting CPU module to RS232 serial port of local PC, as shown in figure 8-3-2. Because the CPU RS232 serial port is not isolated, connect the programming cable before PLC power-on.

Regard that LM3108 CPU module and LM3109 CPU module have 2 serial ports, and download the program to PLC by the left serial port PORT1.

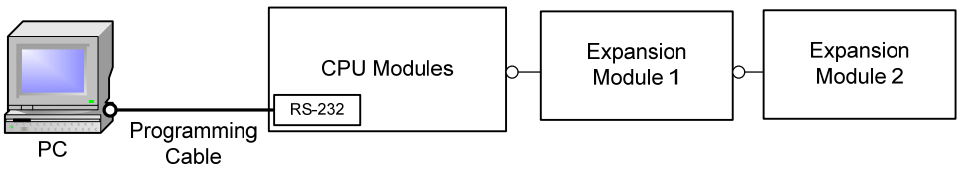


Figure 8-3-2 Connect Programming Cable

8.3.3 Establishing Communication Connection

Download the target file to CPU module, select and configure the channel to establish the communication connection between local PC and target module. The steps are as follows. Click "Online"/"Communication Parameters", and a dialog box of "Communication Parameters" will appear, as shown in figure 8-3-3.

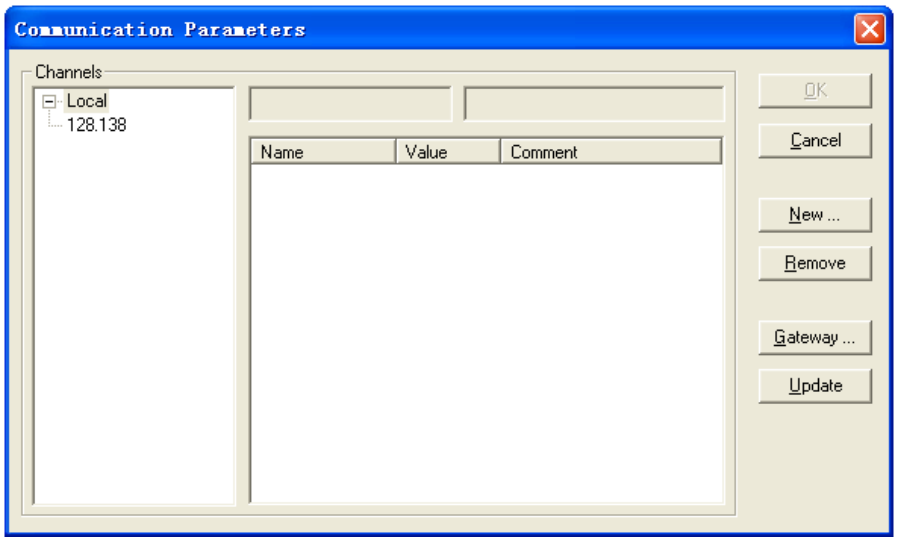


Figure 8-3-3 Dialog Box of Communication Parameters

Click “Gateway” to select “Local” in the field “Connection”, and click “OK”, as shown in figure 8-3-4.

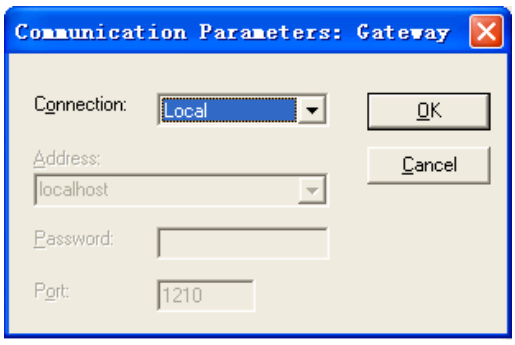


Figure 8-3-4 Communication Parameters: Gateway

Return to “Communication Parameters” dialog by clicking “OK”. Press the New button in the “Communication Parameters” dialog. The dialog Communication Parameters: New Channel comes up, as shown in figure 8-3-5. The default name is “Local_”, and the default communication protocol is RS232 and return to “Communication Parameters” dialog by clicking “OK”.

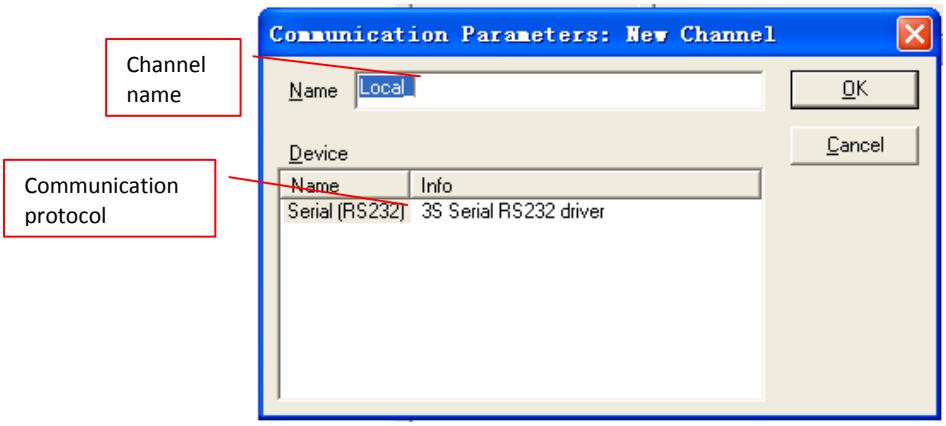


Figure 8-3-5 New Channel

Double click the “Value” in “Baudrate” to change the communication rate to “38400” and click “OK”, as shown in figure 8-3-6. Then the communication connection is set up between the local PC and CPU modules.

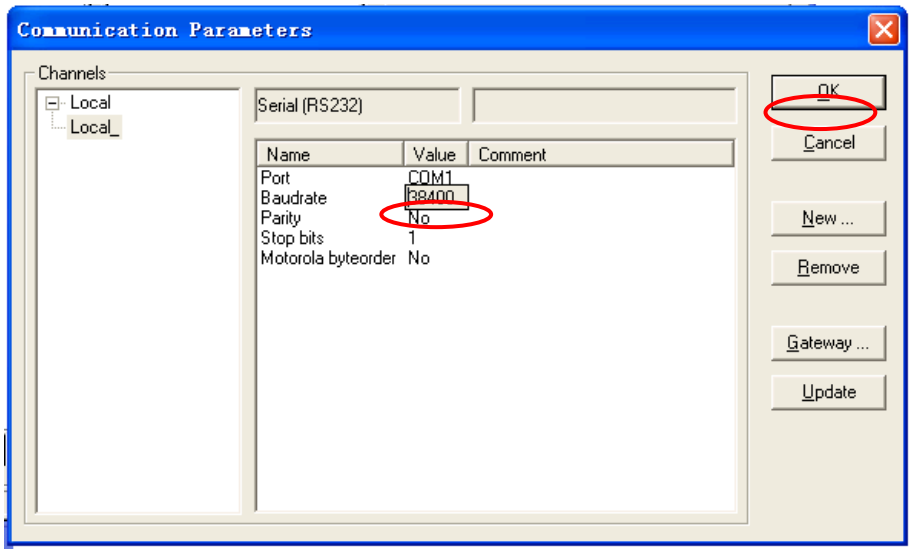


Figure 8-3-6 Setting of Baudrate

8.3.4 Program Downloading

Program Downloading

Use the command “Online”/ “Login” to download programs to PLC. All the target files are downloaded to the module when downloading the target file generated after compilation. At the same tome the module is reset and all variables are returned to initial status. With the command “Online”/ “Login” the connection can be established between local PC and CPU module and a message will appear, as shown in figure 8-3-7.

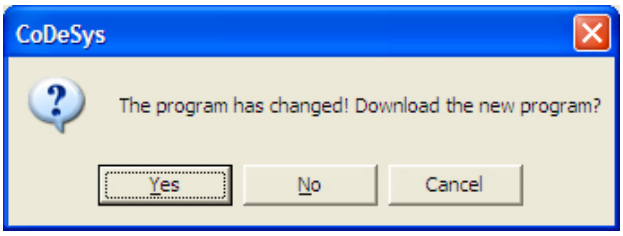


Figure 8-3-7 Download Information

When the downloaded PLC program is not the same as the PLC internal program, the dialog above will appear.

Click the “Yes” button to change the program and download the new program to CPU module and click “No” button to enter the original program and a connection is established only.

Click the “Yes” button to download the new program to CPU module then a dialog box of creating a boot project appears, as shown in figure 8-3-8. A boot project is a program downloaded to Flash. A boot project will be created in FLASH in order to protect the program from being lost during power cycles.

Click the “Yes” button in figure 8-3-8 to run the downloaded program when rebooting. Now the download is finished.

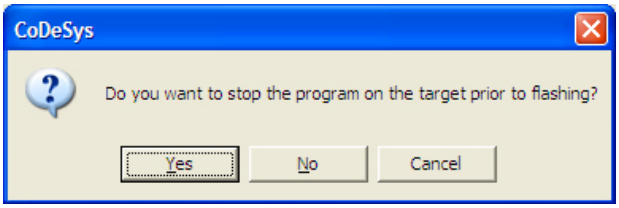


Figure 8-3-8 Create Boot Project Message

Entering monitoring status but do not download the program

If you do not want to download the program and want to enter the monitoring status only, click the “No” button in figure 8-3-7. And then the changed project will not be downloaded to CPU module and you can monitor the original program. Regard to that if you want to download the program when in monitoring status, download it with the command “Online”/ “Download”.

The Difference between Login, Download, Create boot project

To know the terms better you need to know how the program is loaded onto the controller. Before the compiled project in PowerPro is loaded onto the controller, first establish a connection between PowerPro and PLC. The “Login” command combines the programming system PowerPro with the PLC. After the connection is set up PowerPro will judge whether the program has been changed automatically. Change into the online mode if the program is not changed. If the program has been changed a dialog opens with the question “The program has been changed. Load changes?”, No results in a log-in without the changes made since the last download being loaded onto the controller and change into online mode; Then if you want to download the program use the command “Download” to do so. That is to say, the command “Download” is available after log-in. By answering “Yes” you confirm that, on log-in, the modified portions of the project are to be loaded onto the CPU. All the data will be cleared when CPU power-off, so save the program to FLASH after it has been downloaded to CPU to ensure the program still exists after PLC power-on again, the process is called “create a boot project”. When the download to CPU is finished, a dialog box of “Stop target program before creating a boot project” will appear, click “Yes” or “No” to create a boot project. Click “Cancel” button not to create a boot project and in this situation the new program will be not be saved and the original program is saved after power-off and power-on again. Now with the command “Online”/ “Create boot project” you can save the new program in FLASH.

8.4 DEBUGGING

All the debugging commands provided by system are under the “Online” menu and are available in debugging mode, as shown in figure 8-4-1. In debugging mode the different default colors stand for different status and operations, such as TRUE (blue) , FALSE (black), breakpoint (light blue) , flow control (green) and the colors can be set in “Project”/“Options”/“Color”. It’s useful to know the meanings of the colors for debugging and monitoring programs.

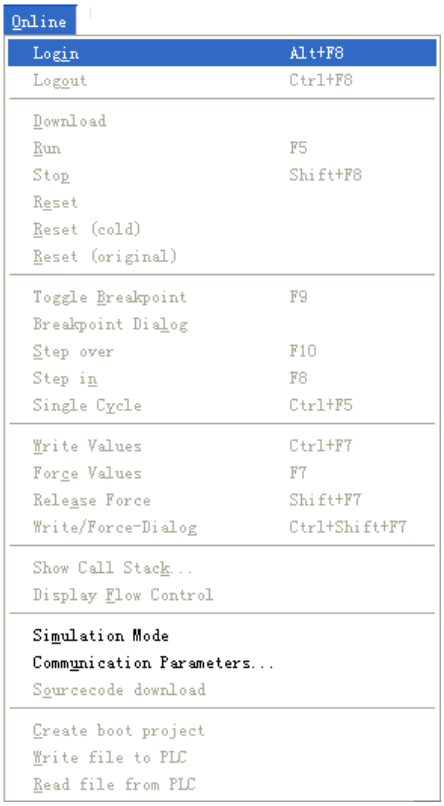


Figure 8-4-1 Online Menu

8.4.1 "Online"/"Login"

With the command "Online"/"Login" to change into the debugging status. Download the programs into CPU modules is called online mode. If the PC is not connected to hardware modules and the user program runs on the local PC is called simulation mode. If "Online"/"Simulation Mode" is chosen, a check ("v") will appear in front of the menu item, and you will be in simulation mode when logging in.

If the function blocks related to RTS hardware are used in a program, then in simulation mode that implementation of the function blocks will be executed. For example system events cannot be implemented in simulation mode. All the function blocks related to RTS hardware, including external expansion function blocks and external function blocks in customized library, cannot be used in simulation mode.

In addition, after setting the free port parameters if you want to obtain again the download and debugging functions in previous programming system, switch RUN/STOP to STOP to log in.

8.4.2 "Online"/"Logout"

With the command "Online"/"Logout" the connection to the PLC is broken, or the simulation mode program is ended and is switched to the offline mode.

8.4.3 "Online"/"Run"

The command "Online"/"Run" starts the program in the PLC or in simulation mode. The command can be used in the following situations:

- After the “Online”/“Login” command.
- After the user program in the PLC has been ended with the “Stop” command.
- When the user program is at a breakpoint.
- After the “Single cycle” command has been executed.

8.4.4 “Online”/“Stop”

With the command “Online”/“Stop”, stop the execution of the program in the PLC or in simulation mode and save the values of current variables. And now the program is still in debugging status, and you can restart the program with the command “Online”/“Run” at the position where it stopped last time.

8.4.5 Reset

If you have initialized the variables with specific values, then the command “Reset” will reset the variables to the initialized value with the exception of the retain variables. Press “F5” to restart the program.

The command “Reset (cold)” will reset the variables to the initialized value including retain variables but not the persistent variables. Press “F5” to restart the program.

The command “Reset (original)” resets all variables including retain variables and persistent variables to their initial values and erases the user program on the controller. The controller is returned to its original state.

A dialog box will appear to confirm the reset when executing the commands above, as shown in figure 8-4-2.

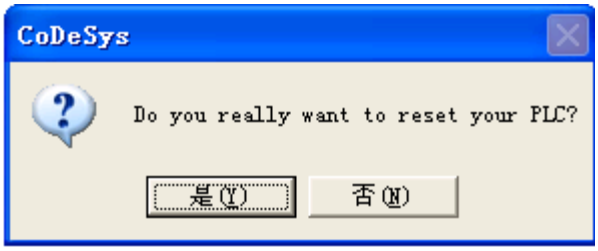


Figure 8-4-2 Reset Confirmation

8.4.6 Breakpoints

A breakpoint is a place in the program at which the processing is stopped. Thus it is possible to look at the values of variables at specific places within the program.

Setting a Breakpoint

With the command “Online”/“Toggle Breakpoint” to set or remove a breakpoint in the present position. The position at which a breakpoint can be set depends on the language in which the POU in the active window is written. You can set a breakpoint in the number field or the network number field with a dark background color, but not in a grey background color, as shown in figure 8-4-3.

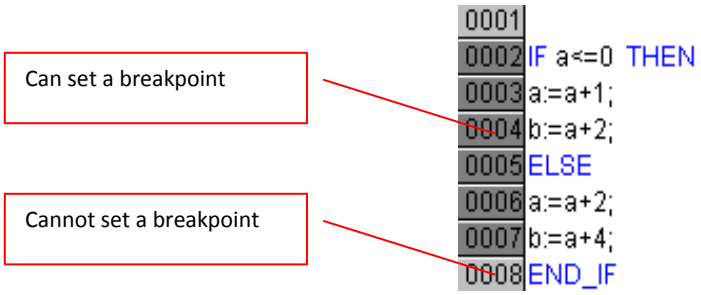


Figure 8-4-3 Breakpoint Setting (1)

In IL and ST the breakpoints is set at the line where the cursor is located. In FBD and LD the breakpoint is set at the currently selected network. In SFC the breakpoint is set at the currently selected step.

You can click on the line number or network number with a dark background color to set or remove a breakpoint. If the line is a breakpoint position, the line number field or the network number field or the step will be displayed with a light-blue background color.

If a breakpoint is reached while the program is running, the program will stop, and the corresponding field will be displayed in a red background color. In order to continue the program, use the “Online”/“Run”, “Step in” or “Step over” commands.

For example, set a breakpoint at line 0004 after logging in, as shown in figure 8-4-4.

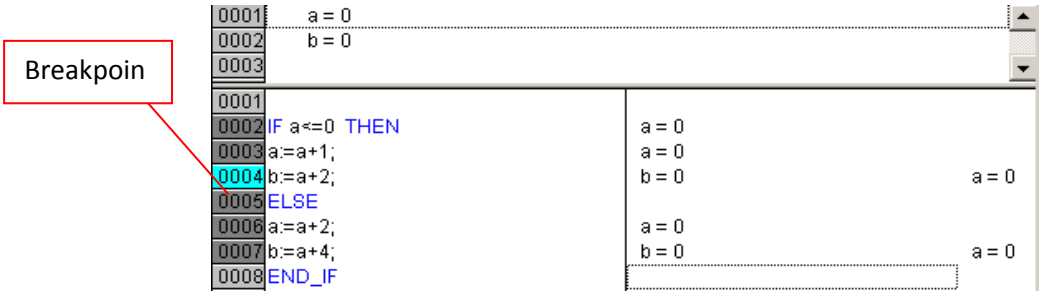


Figure 8-4-4 Breakpoint Setting (2)

Program stops at line 0004, b keeps the initial value, and a is re-assigned, as shown in figure 8-4-5.

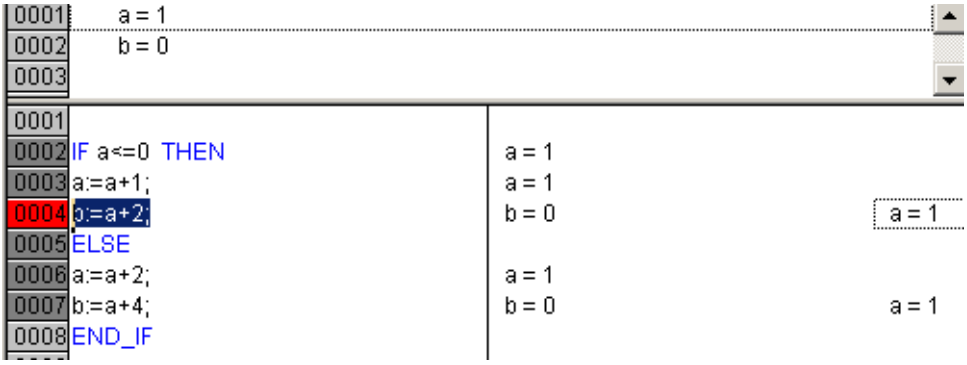


Figure 8-4-5 Breakpoint Setting (3)

Editing Breakpoints

With the command “Online”/“Breakpoint Dialog” open a dialog box to display and edit breakpoints throughout the entire project, as shown in figure 8-4-6.

- Set a breakpoint: choose a POU in the field “POU” and the line or the network in the field “Location” where you would like to set the breakpoint, and then press “Add”. The breakpoint will be added to the list.
- Delete a breakpoint: select a breakpoint and press “Delete” to delete the breakpoint.
- Check a breakpoint: select a breakpoint and press “Goto” to go to the location in the editor where a certain breakpoint was set.

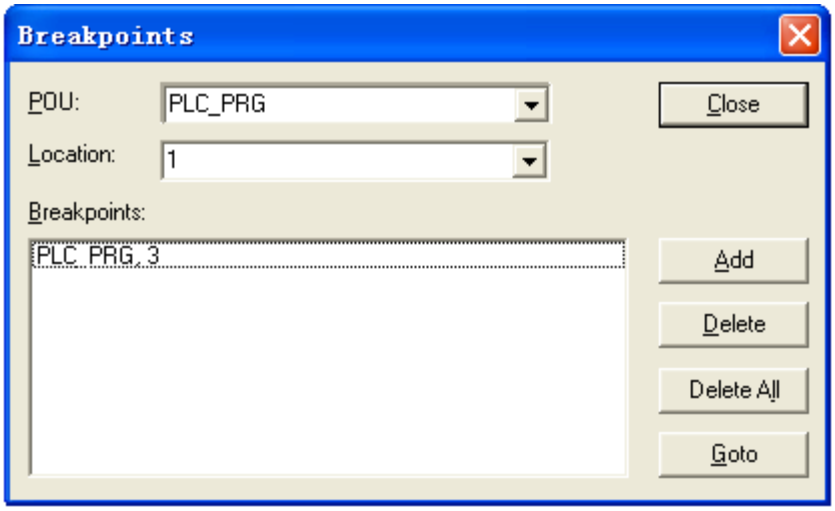


Figure 8-4-6 Breakpoint Setting (4)

All breakpoints are deleted after executing the command “Online”/“Logout”. Note that the number of breakpoints should be less than 100, or else an error will appear, as shown in figure 8-4-7.

Figure 8-4-7 Breakpoint Setting (5)

8.4.7 Single Step

By proceeding step by step you can check the logical correctness of your program. For different programming languages single step in active window has different meanings:

- In IL: Execute the program until the next CAL, LD or JMP command.
- In ST: Execute the next instruction.
- In LD, FBD: Execute the next network.
- In SFC: Continue the action until the next step.

With the commands “Online”/“Step over” or “Step in” execute a single step. When a breakpoint is reached the program will stop after the current instruction. When a function

or a function block is reached, use the command “Step over” to jump to the next instruction, or use the command “Step in” move to the first instruction of a called function or function block.

8.4.8 Single Cycle

With the command “Online”/“Single Cycle” execute a single PLC cycle and stops after this cycle. In essence, “Single Cycle” is like “Run” and both of them can execute the user program online. “Single Cycle” command executes a single PLC cycle and stops after this cycle while “Run” command executes the program continuously until the command “Stop” is used.

8.4.9 Writing Values

Setting new values

In debug mode, double click on the line in which a variable is declared, a dialog box will appear as shown in figure 8-4-8. Enter the new value in the field “New Value” and press “OK”, and then the value is displayed in brackets and in turquoise colour behind the former value of the variable. For Boolean variables, the value is toggled (switched between TRUE and FALSE, with no other value allowed) by double-clicking on the line in which the variable is declared and no dialog appears.

Activating new values

Use the command “Online”/“Write Values” or shortcut “Ctrl+F7” to active new values of variables and write them to modules. When the command “Write Values” is executed, all the values are written to the appropriate variables in the controller at one time.

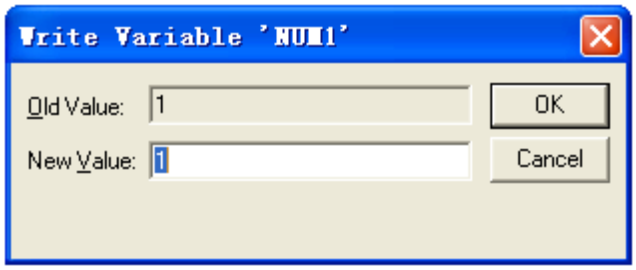


Figure 8-4-8 Write Variable

8.4.10 Forcing Values

Forcing Values

It is the same as “Write Values” for setting and activating new values. First, set new values and then execute “Online”/“Force Values” or shortcut “F7” to write the new values to the appropriate variables in the controller.

If the ‘Online’/‘Force values’ command is given, all variables will be set to the selected values after each cycle until the ‘Release force’ is given. With the command “Write Values”, the variables are set to the selected values once and can be assigned again by other programs.

When the digital input is a physical point, such as I0.0, the “Force Values” command must be used. But in simulation mode “Write Values” can be used. You are allowed to execute the command “Force Values” after multiple variables have been written values.

The forced values are display in red, as shown in figure 8-4-9.

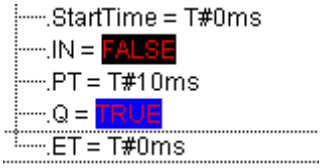
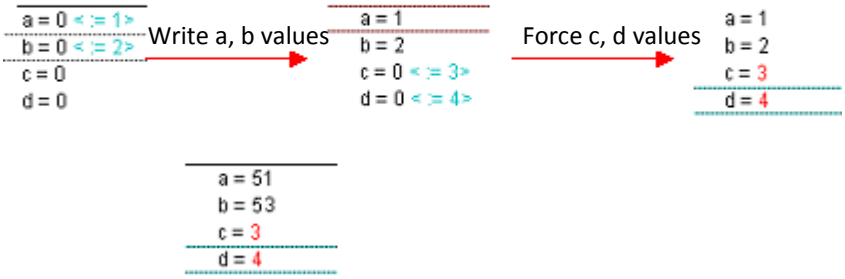


Figure 8-4-9 Force Values

We can see the difference between "Write Values" and "Force Values" through the following example.

Main program:

```
a:=a+1;
b:=a+2;
c:=a+3;
d:=a+4;
```



Display after running

The command "Write Value" only changes the current values of variables a and b, once the program is running, the variables a and b will increase automatically according to the program setting and are re-assigned at each execution. With the command "Force Values" variables c and d are assigned the forced values and retain the 3, 4 at each execution.

Releasing Force

The command "Online"/"Release Force" ends the forcing of variable values. The variable values change again in the normal way with black color.

Write/Force-Dialog

When the command "Online"/"Write/Force-Dialog" is used, a dialog box will appear, which displays in two tabs the current writelist and forcelist, as shown in figure 8-4-10

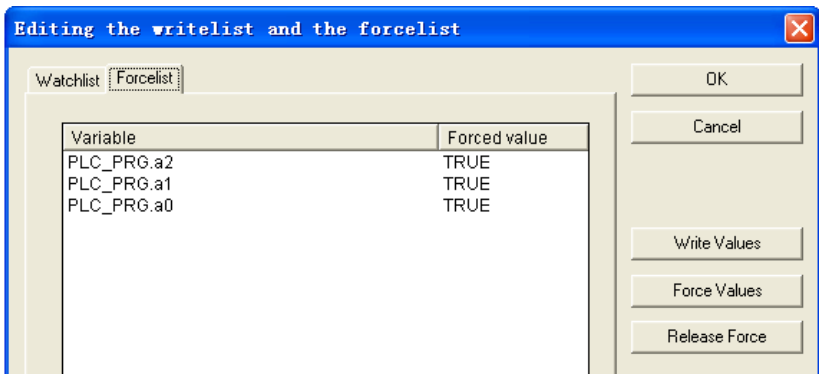


Figure 8-4-10 Write/Force-Dialog

8.4.11 Showing Call Stack

When the command “Online”/“Show Call Stack” is used, the program will stop at a breakpoint and a dialog box with a list of the POU call stack will appear, as shown in figure 8-4-11.

The first POU in the list is always the default PLC_PRG or the first called POU set in “Task configuration”. The last POU is always the POU being executed.

After you have selected a POU and have pressed the “Go To” button, the selected POU is loaded in its editor, and it will display the line or network being processed.

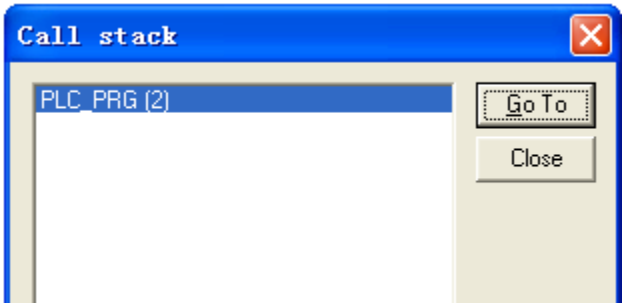


Figure 8-4-11 Call Stack

8.4.12 Displaying Flow Control

If “Online”/“Display Flow Control” is selected a check (“√”) will appear in front of the menu. Following this, every line or every network will be marked which was executed in current cycle. The line number field or the network number field of the lines or networks which just run will be displayed in green. In “Single Cycle” mode, you can see the current flow of POU clearly.

A simple example is shown in figure 8-4-12, a is an INT variable and is assigned 0. The variable increases by 1 when a<=0 and increases by 2 when a>0.

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     a: INT:=0;
0004 END_VAR
0005
```

```
0002 IF a<=0 THEN
0003 a:=a+1;
0004 ELSE
0005 a:=a+2;
0006 END_IF
```

Figure 8-4-12 Example (1)

After “Login”, select “Display Flow Control” and then select “Single Cycle”. Because the initial value of a is 0, and at the first running time a<=0 so a increases by 1 and changes into 1. With the command “Display Flow Control” the line number field (line 2 and 3) which just run in this cycle will be displayed in green, as shown in figure 8-4-13.

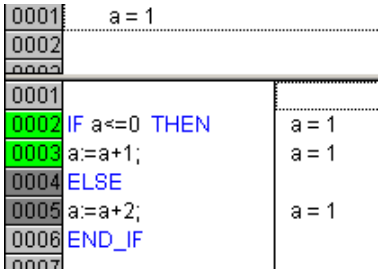


Figure 8-4-13 Example (2)

Select “Single Cycle” again, but now a is 1 and increases 2 and changes into 3. When the command “Display Flow Control” is used, the line number field (line 2 and 5) which just run in this cycle will be displayed in green, as shown in figure 8-4-14.

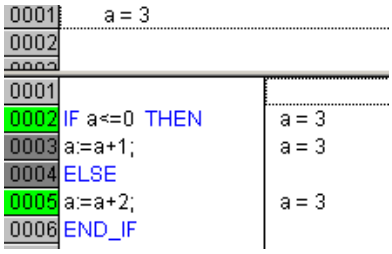


Figure 8-4-14 Example (3)

8.4.13 Watch- and Recipe Manager

In the “Resources” tab in the object organizer, double click “Watch- and Recipe Manager” and a window will appear, which is shown in figure 8-4-15. With the help of the “Watch- and Recipe Manager” you can view the values of selected variables. The “Watch- and Recipe Manager” also makes it possible to preset the variables with definite values and transfer them as a group to the PLC. In the same way, current PLC values can be read into and stored in the “Watch- and Recipe Manager”.

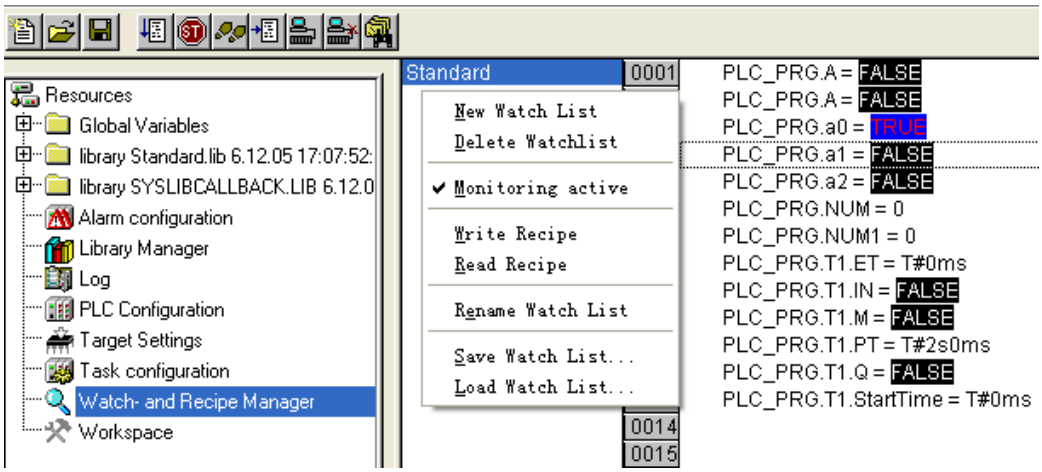


Figure 8-4-15 Watch- and Recipe Manager

New Watch List

Click the right mouse key at the left column of the “Watch- and Recipe Manager” and select “New Watch List” and enter the desired name. You can also new a watch list by clicking “Insert”/“New Watch List”.

Renaming a Watch List

With the command “Rename Watch List”, you can change the name of watch list.

Selecting Monitoring Variables

Select “Input Assistant” by clicking the right mouse in the “Watch- and Recipe Manager” or “Edit”, and then all the available variables in the project are listed. You can select the needed variables into the watch list.

Saving Watch List

With the command “Save Watch List” you can save a watch list. The file name is preset with the name of the watch list and is given the extension “*.wtc”.

Loading Watch List

With the command “Load Watch List” you can reload a saved watch list.

Monitoring active

If the display is active, a check (√) will appear in front of the menu item. All the functions about “New Watch List” are effective only before “Monitoring active” is activated.

Writing Recipe

With the command “Write Recipe” you can write the preset values into the variables.

Reading Recipe

With the command “Read Recipe”, you can replace the presetting of the variables with the present value of the variables.

Chapter 9

IEC Programming Fundamentals

PowerPro conforms to IEC61131-3 standard of IEC (International Electrotechnical Commission, IEC). Programming by LD has been introduced in section 7.4, and the other IEC standard programming languages such as FBD, IL, ST, SFC and CFC are described in this chapter.

9.1 FUNCTION BLOCK DIAGRAM (FBD)

FBD is a graphically oriented programming language, similar with LD. FBD works with a list of networks whereby each network contains a structure which represents either a logical or arithmetic expression, the call of a function block, a jump, or a return instruction. This is illustrated in figure 9-1-1.

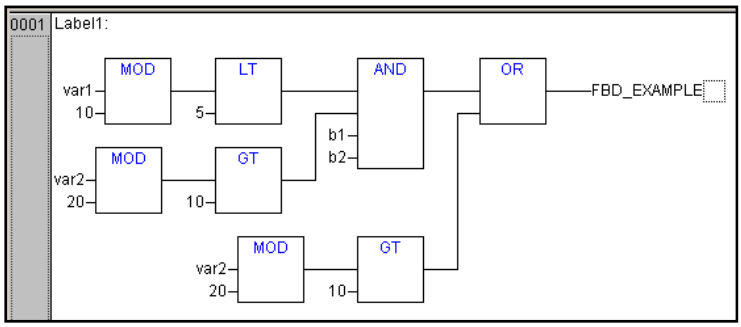
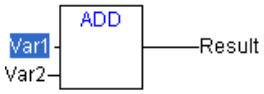


Figure 9-1-1 Language of FBD

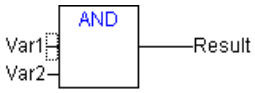
9.1.1 Cursor Position

You can recognize the present cursor position by a dotted rectangle. The following is a list of all possible cursor positions:

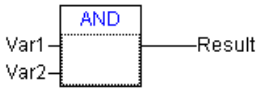
Text (Cursor Position 1) :



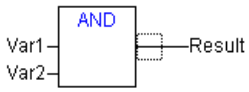
Input (Cursor Position 2) :



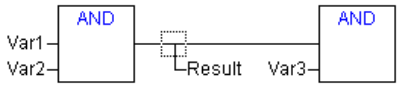
Operator, Function or Function Block (Cursor Position 3) :



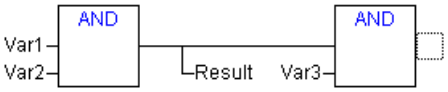
Output (Cursor Position 4, an assignment or a jump comes afterward) :



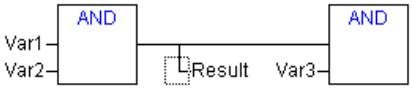
The lined cross above an assignment, a jump or a return instruction (Cursor Position 5) :



Behind the outermost object on the right of every network (Cursor Position 6) :



The lined cross directly in front of an assignment (Cursor Position 7) :



9.1.2 Operation Description

Insert "Input"

Shortcut:  Insert an input of an operator in current cursor position.

- For some operators the number of inputs may vary and sometimes it is necessary to extend such an operator by an input. For example, the inputs of ADD may be two or more constants (variables). If you select an input (Cursor Position 2) , then the new inserted input become the first input of the operator. You must select the operator itself (Cursor Position 3), if a lowest input is to be inserted. The inserted input is allocated with the text "???". This text must be clicked and changed into the desired constant or variable. For this you can also use the Input Assistant.
- Increasing the number of inputs can simplify programming, as shown in figure 9-1-2.

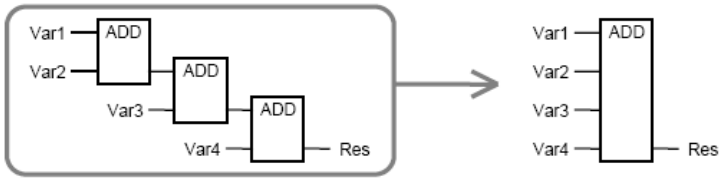


Figure 9-1-2 Extend Inputs of Operators

- How to switch inputs fast? The branch located on the right of the operator is now connected with the first input, but should be connected with the second input. You can now select the first input and perform the command "Edit"/"Cut". Following this, you can select the second input and perform the command "Edit"/ "Paste". This way, the branch is dependent on the second input.

Insert "Output"

Shortcut:  Insert an output of an operator in current cursor position.

- For some operators the number of outputs may vary and the command can be used to extend such an operator by an output.

Insert "Box"

Shortcut:  Insert an operator in current cursor position.

- If an input is selected (Cursor Position 2), the POU is inserted in front of this input. The first input of this POU is linked to the branch on the left of the selected input. The output of the new POU is linked to the selected input.
- If an output is selected (Cursor Position 4), the POU is inserted after this output. The first input of the POU is connected with the selected output. The output of the new POU is linked to the branch with which the selected output was linked.
- If a POU, a function, or a function block is selected (Cursor Position 3), the old element will be replaced by the new POU. As far as possible, the branches will be connected the same way as they were before the replacement. If the old element had more inputs than the new one, the unattachable branches will be deleted. The same holds true for the outputs.
- If a jump or a return is selected, the POU will be inserted before this jump or return. The first input of the POU is connected with the branch to the left of the selected element. The output of the POU is linked to the branch to the right of the selected element.
- If the last cursor position of a network is selected (Cursor Position 6), the POU will be inserted following the last element. The first input of the POU is linked to the branch to the left of the selected position.
- First of all, an "AND" operator is always inserted. This can be changed by Selection and Overwrite of the type text ("AND") into every other operator, every function, every function block and every program. You can select the desired POU by using Input Assistant. The new inserted operator is linked to the branch to the right of the selected element.

Insert "Assign"

Shortcut:  Insert an assignment and output the result.

- Depending on the selected position, insertion takes place directly in front of the selected input (Cursor Position 2), directly after the selected output (Cursor Position 4) or at the end of the network (Cursor Position 6).
- In order to insert an additional assignment into an existing assignment, use the "Insert" "Output" command.

Insert "Jump"

Shortcut:  Insert a jump and jump to the desired position if the condition is TRUE.

- Depending on the selected position, insertion takes place directly in front of the selected input (Cursor Position 2), directly after the selected output (Cursor Position 4) or at the end of the network (Cursor Position 6).
- For an inserted jump, the jump can be replaced by the label to which it is to be assigned

Insert "Return"

Shortcut:  Insert a return.

- When the current POU is called by other POU and the return condition is TRUE, return to the calling POU.
- Depending on the selected position, insertion takes place directly in front of the selected input (Cursor Position 2),directly after the selected output (Cursor Position 4), directly before the selected line cross (Cursor Position 5), or at the end of the network (Cursor Position 6).

"Negative"

Shortcut:  Negative.

- With this command you can negate the inputs, outputs, jumps, or RETURN instructions.
- The symbol for the negation is a small circle at a connection. If an input is selected (Cursor Position 2), this input will be negated. If an output is selected (Cursor Position 4), this output will be negated. If a jump or a return is marked, the input of this jump or return will be negated. A negation can be canceled through renewed negation.

"Set/Reset"

Shortcut:  Set/Reset

- With this command you can define outputs as Set or Reset Outputs.
- A grid with Set Output is displayed with [S], and a grid with Reset Output is displayed with [R], as shown in figure 9-1-3. An Output Set is set to TRUE, if the grid belonging to it returns TRUE. The output now maintains this value, even if the grid jumps back to FALSE. An Output Reset is set to FALSE, if the grid belonging to it returns FALSE. The output maintains its value, even if the grid jumps back to FALSE. With multiple executions of the command, the output will alternate between set, reset, and normal output.

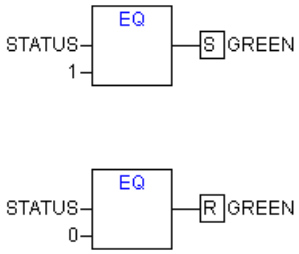


Figure 9-1-3 Set and Reset

- An application example in FBD language is shown in figure 9-1-4 to generate a pulse with "1s off 2s on".

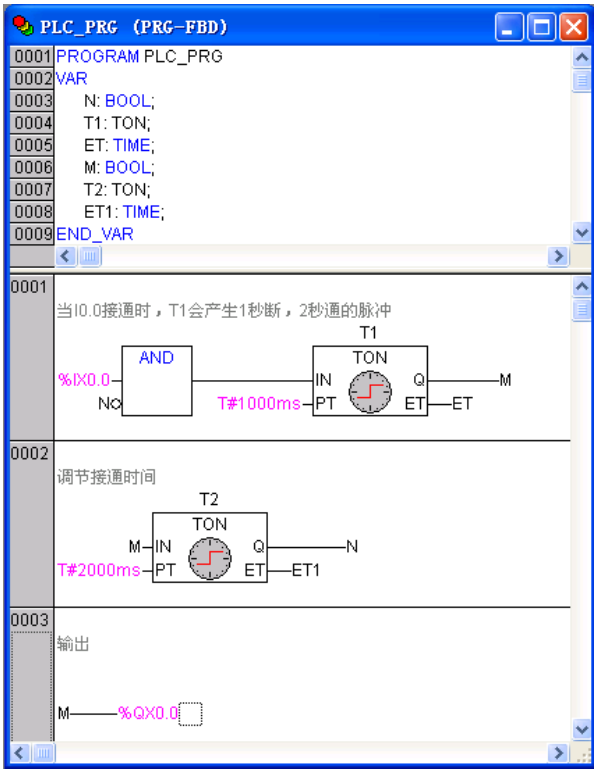


Figure 9-1-4 Example in FBD Language

9.2 INSTRUCTION LIST (IL)

Instruction List (IL) is a programming language with an assembly language style and is difficult to read, but it executes fastest. IL consists of a series of instructions. Each instruction begins with a new line and contains an operator and one or more operands separated by commas. In front of an instruction there can be a label followed by a colon. At the end of an instruction there can be a comment enclosed in "(* *)". Empty lines can be inserted between instructions. The Instruction List editor is a text editor with the usual capabilities of Windows text editors. The most important commands are found in the context menu or the menu bar.

9.2.1 Operation Description

Open "Insert", select what you want to insert, as shown in figure 9-2-1

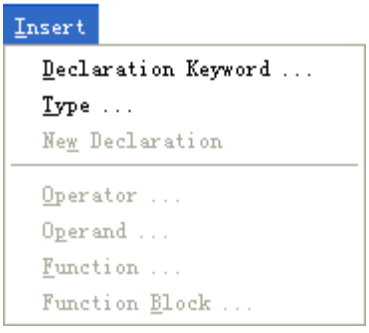


Figure 9-2-1 "Insert" Menu

"Insert"/"Operator"

Click the command at any position in programming area, and a dialog will appear. Select what you need, such as “ABS”.

“Insert”/“Operand”

Click the command at any position in programming area, and a dialog will appear. Select the operand, and click “ok”.

“Insert”/“Function”

Click the command at any position in programming area, and a dialog will appear. Select the function, and click “ok”.

“Insert”/“Function Block”

Click the command at any position in programming area, and a dialog will appear. Select the function block, and click “ok”.

9.2.2 Program Example

Example

The following is a simple program in the IL language.

```
Declaration:
VAR
        A: REAL;
        B: REAL;
        C: REAL;
END_VAR
Program:
LD          10      (*make the current result equal to 10*)
ADD   A     (*execute ADD with variable A *)
GE          B       (*judge whether the current value is >= variable B *)
JMPC Next1  (*if the result was TRUE then jump to the label “Next1”*)
LD          A       (* make the current result equal to A*)
ADD   B     (* execute ADD with variable B*)
ST          C       (*A+B in C*)
JMP Next2   (*jump to label Next2*)

Next1:      (*label*)
LD          A       (* make the current result equal to A*)
SUB         B       (*execute SUB with B*)
ST          C       (*A-B in C*)
Next2:      (*label*)
```

In IL there are two modifiers C and N. C: the instruction is only executed if the result of the preceding expression is TRUE. N with JMP, CAL, RET: the instruction is only executed if the result of the preceding expression is FALSE. Otherwise, N is the negation of the operand. All the operators and modifiers in IL are listed in table 5-4-1.

Operators	Modifiers	Meaning
LD	N	Make current result equal to the operand
ST	N	Save current result at the position of the operand
S		Then put the Boolean operand exactly at TRUE if the current result is TRUE

R		Then put the Boolean operand exactly at FALSE if the current result is TRUE
AND	N, (	Bitwise AND
OR	N, (	Bitwise OR
XOR	N, (	Bitwise exclusive OR
ADD	(	Addition
SUB	(	Subtraction
MUL	(	Multiplication
DIV	(	Division
GT	(	>
GE	(	>=
EQ	(	=
NE	(	<>
LE	(	<=
LT	(	<
JMP	C, N	Jump to the label
CAL	C, N	Call other POU
RET	C, N	Leave POU and return to caller
)		Evaluate deferred operation

Table 9-2-1 Operators and Modifiers in IL Language

Example

Here is the example using modifiers in IL:

```
LD    TRUE   (*make current result equal to TRUE *)
ANDN  BOOL1  (*execute AND with the negated value of BOOL1variable*)
JMPC  mark   (*if the result was TRUE, then jump to the label "mark" *)
LDN   BOOL2  (*save the negated value of *)
ST    ERG    (*BOOL2 in ERG*)
Mark:
LD    BOOL2  (*save the value of *)

ST    ERG    (*BOOL2 in ERG*)
```

It is also possible in IL to put parentheses after an operation. The value of the parenthesis is then considered as an operand.

For example:

```
LD   2
MUL  2
ADD  3
ST   ERG
```

Here the value of ERG is 7. However, if one puts parentheses:

```
LD   2
MUL  (2
ADD  3
)
```

ST ERG

The resulting value for ERG is 10, the operation MUL is only executed if you come to “)”, as operand for MUL 5 is then calculated.

Example

A simple application example in IL is shown in figure 9-2-2. The purpose is to generate a pulse signal with 1s off and 2s on.

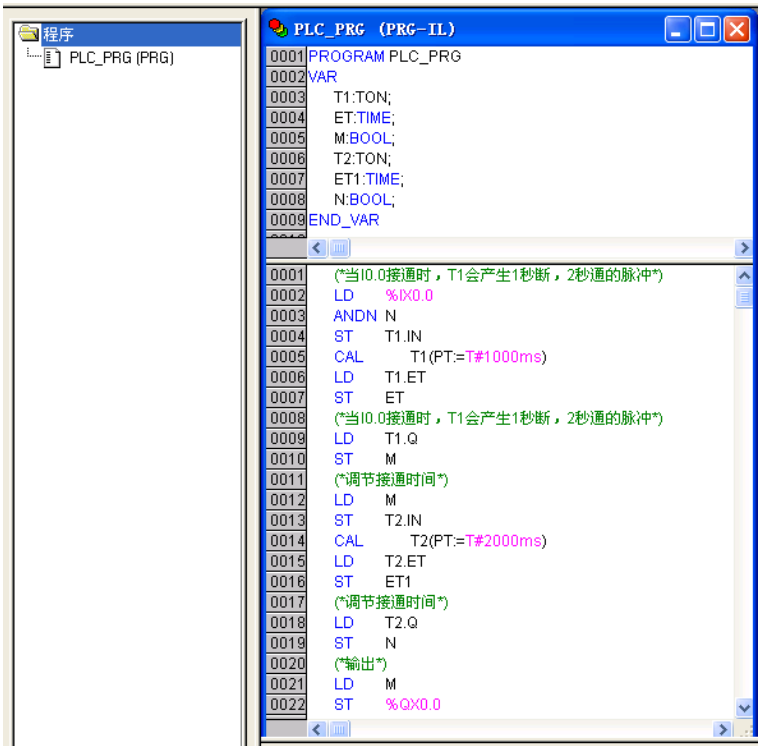


Figure 9-2-2 Example in IL Language

9.3 STRUCTURED TEXT (ST)

The Structured Text consists of a series of instructions such as (IF…THEN…ELSE) and (WHILE…DO), similar to high level languages PASCAL and BASIC.

9.3.1 ST Expression

Expressions in ST are composed of operators and operands. An operand can be a constant, a variable, a function call or another expression. The evaluation of expression takes place by means of processing the operators according to certain priority. The operator with the highest priority is processed first, then the operator with the next highest priority, until all operators have been processed. Operators with equal priority are processed from left to right. The operators in ST are listed in table 9-3-1.

Operation	Symbol	Pripory
Put in parentheses	(expression)	Strongest pripory
Function call	Function name (parameter list)	
Exponentiation	EXPT	
Negate	-	

Building of complements	NOT	
Multiply	*	
Divide	/	
Modulo	MOD	
Add	+	
Subtract	-	
Compare	<, >, <=, >=	
Equal to	=	
Not equal to	<>	
Boolean AND	AND	
Boolean XOR	XOR	
Boolean OR	OR	Weakest pripory

Table 9-3-1 ST Operators

9.3.2 ST Instruction

Instruction list in ST is shown in table 9-3-2.

Type of Instruction	Example
Assignment operator	A:=B; CV := CV + 1; C:=SIN(X);
Calling function block	TP(IN:= %IX0.5, PT:=t#30);
Output of function block	A:=TP.Q;
Return	RETURN;

IF	D:=B*B;
	IF D<0.0 THEN
	C:=A;
	ELSE IF D=0.0 THEN
	C:=B;
	ELSE
	C:=D;
	END_IF;

CASE	CASE INT1 OF 1: BOOL1 := TRUE; 2: BOOL2 := TRUE; ELSE BOOL1 := FALSE; BOOL2 := FALSE; END_CASE;

FOR loop	J:=101;
	FOR I:=1 TO 100 BY 2 DO
	IF ARR[I] = 70 THEN
	J:=I;
	EXIT;
	END_IF;
	END_FOR;

WHILE loop	J:=1; WHILE J<= 100 AND ARR[J] <> 70 DO J:=J+2;

	END_WHILE
REPEAT loop	J:=-1; REPEAT J:=J+2; UNTIL J= 101 OR ARR[J] = 70 END_REPEAT;
Exit instruction	EXIT;
Null instruction	;

Table 9-3-2 Instructions in ST

Assignment operator

- On the left side of an assignment there is an operand (variable, address) to which is assigned the value of the expression on the right side. For example: Var1 := Var2 * 10;

Calling function blocks

- A function block is called by writing the name of the instance of the function block and then assigning the values of the parameters in parentheses. For example:

Variable declaration:

```
TPInst: TP;
VarBOOL1: BOOL;
VarBOOL2: BOOL;
```

Program:

```
TPInst(IN:= VarBOOL1,PT:= T#5s); (*IN: trigger signal, PT: length of high voltage *)
VarBOOL2:=TPInst.Q; (*assign pulse output Q to variable VarBOOL2*)
```

RETURN instruction

- The instruction can be used to leave a POU depending on a condition.

IF instruction

- With the IF instruction you can check a condition and depending on the condition execute instructions.

Syntax:

```
IF <Boolean expression> THEN
<IF instructions >
{ELSIF < Boolean expression 1> THEN
<ELSE IF instructions 1>
ELSIF < Boolean expression n> THEN
<ELSE IF instructions n>
ELSE
<ELSE instructions >}
END_IF;
The part in {} is optional.
```

- If < Boolean expression > returns TRUE, only <IF instructions> are executed and none of the other instructions. Otherwise the Boolean expressions beginning with < Boolean expression 1> are evaluated one after the other until one of the expressions returns TRUE. After that, only those instructions after this Boolean expression and before the next ELSE or ELSE IF are evaluated.
- If none of the Boolean expressions produce TRUE, only the <ELSE instructions >are evaluated. For example:

```
IF temp<17
THEN heating_on := TRUE;
ELSE heating_on := FALSE;
END_IF;
```

- Here, the hearting is turned on when the temperature is below 17 degrees. Otherwise it remains off.

CASE instruction

- With the CASE instruction one can combine several conditioned instructions with the same condition variable in one construct.

Syntax:

```
CASE <Var1> OF
<Value1>: < instruction 1>
<Value2>: < instruction 2>
<Value3, Value4, Value5>: < instruction 3>
<Value6 .. Value10>: < instruction 4>
...
<Value n>: < instruction n>
ELSE <ELSE instruction >
END_CASE;
```

A CASE instruction is processed according to the following model:

- If the variable in <Var1> has the value < value i>, the the instruction < instruction i> is executed.
- If <Var1> has none of the indicated values, the <ELSE instruction >is executed.
- If the same instruction is to be executed for several values of the variables, one can write these values one after the other separated by commas, and thus condition the common execution.
- If the same instruction is to be executed for a variable range of a variable, one can write the initial value and the end value separately by two dots one after the other. So you can condition the common condition.

For example:

```
CASE INT1 OF
1, 5: BOOL1 := TRUE;
BOOL3 := FALSE;
2: BOOL2 := FALSE;
BOOL3 := TRUE;
10..20: BOOL1 := TRUE;
BOOL3:= TRUE;
ELSE
```

```
BOOL1 := NOT BOOL1;
BOOL2 := BOOL1 OR BOOL2;
END_CASE;
```

FOR loop

- With the FOR loop one can program repeated processes.

Syntax:

```
INT_Var :INT;
FOR <INT_Var> := <INIT_VALUE> TO <END_VALUE>
{BY <Step Size>} DO
<Instructions>
END_FOR;
```

The part in {} is optional.

- The <Instructions> are executed as long as the counter <INT_Var> is not greater than <END_VAULE>. This is checked before executing the <Instructions> so that the <Instructions> are never executed if <INIT_VALUE> is greater than <END_VALUE>.
- When <Instructions> are executed, <INT_Var> is always increased by <Step Size>. The step size can be any integer value. If it's missing the default is 1. The loop must also end since <INT_Var> is greater than <END_VALUE>.

For example:

```
FOR Counter:=1 TO 5 BY 1 DO
Var1:=Var1*2;
END_FOR;
Erg:=Var1;
```

Let us assume that the default setting for Var1 is 1, then Var1 will be 32 after the FOR loop.

Note: <END_VALUE> must not be equal to the limit value of the counter <INT_VAR>. For example, if the variable Counter is of type SINT and if <END_VALUE> is 127, you will get an endless loop.

WHILE loop

- The WHILE loop can be used like the FOR loop with the difference that the break-off condition can be any Boolean expression. This means that you indicate a condition and the loop will be executed if the condition is satisfied.

Syntax:

```
WHILE <Boolean expression>
<Instructions>
END_WHILE;
```

- <Instructions> are repeatedly executed as long as the <Boolean expression> returns TRUE. If the <Boolean expression> is already FALSE at the first evaluation, then the <Instructions> are never executed. If <Boolean expression> never assumes the value FALSE, the <Instructions> will be repeated endlessly which causes a relative time delay.

For example:

```
WHILE counter<>0 DO
Var1 := Var1*2;
Counter := Counter-1;
END_WHILE
```

- In a certain sense, the WHILE and REPEAT loops are more powerful than the FOR loop since one doesn't need to know the number of cycles before executing the loop. In some cases one will only be able to work with these two loop types. However, if the number of the loop cycles is clear, then a FOR loop is preferable since it allows no endless loops.

REPEAT loop

- REPEAT loop is different from WHILE loop because the break-off condition is checked only after the loop has been executed. This means that the loop will run through at least once, regardless of the break-off condition.

Syntax:

```
REPEAT
<Instructions>
UNTIL <Boolean expression>
END_REPEAT;
```

- The <Instructions> are carried out until the <Boolean expression> returns TRUE. If <Boolean expression> is produced already at the first TRUE evaluation, the <Instructions> are executed only once. If <Boolean expression> never assume the value TRUE, the <Instructions> will be repeated endlessly which causes a relative time delay.

For example:

```
REPEAT
Var1 := Var1*2;
Counter := Counter-1;
UNTIL
Counter=0
END_REPEAT;
```

EXIT instruction

- If the EXIT instruction appears in a FOR, WHILE or REPEAT loop, then the loop is ended regardless of the break-off condition.

Example, here is a simple program in ST language.

Variable declaration:

```
PROGRAM PLC
VAR
A: BOOL;
B: BOOL;
C: INT;
END_VAR
```

```
Program:
IF A=TRUE THEN
C:=20;
ELSE IF B=TRUE THEN
C:=30;
ELSE C:=50;
END_IF
```

Neat and legible ST language

- ST obviously means that it's structured programming. That is to say ST language provides a predefined structure for special common used structured programming. Example:
- Compare the program codes written in IL and ST in which the circle 2 exponentiation is implemented.

In IL language:

```
Loop:
LD Counter
NE 0
NOT
JMPC END_LOOP
LD Var1
MUL 2
ST Var1
LD Counter
SUB 1
ST Counter
JMP Loop
End_LOOP:
LD Var1
ST ERG
```

In ST language:

```
WHILE counter<>0 DO
Var1:=Var1*2;
Counter:=counter-1;
END_WHILE
Erg:=Var1;
```

- We can see that the program written in ST is neat and easy to read.

9.4 SEQUENTIAL FUNCTION CHART (SFC)

The Sequential Function Chart (SFC) is a graphically oriented language which makes it possible to describe the chronological order of different actions within a program. For this the actions are assigned to step elements and the sequence of processing is controlled by transition elements.

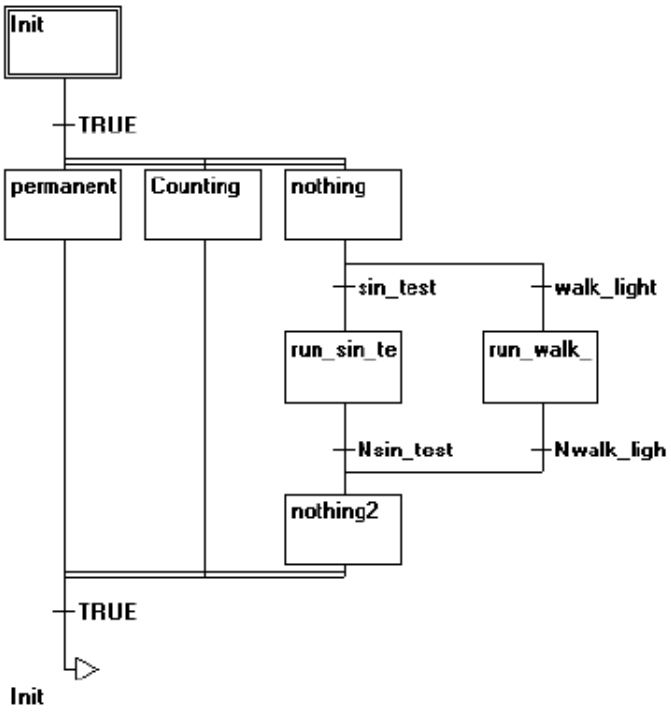


Figure 9-4-1 Language of Sequential Function Chart

9.4.1 Basic Concepts

Steps

- A POU written in SFC consists of a series of steps which are connected with each other through directed connections. There are two types of steps.
- The simplified type consists of an action and a flag which indicates if the step is active. If the action of a step is implemented, then a small triangle appears in upper right corner of the step.
- An IEC step consists of one or more actions and Boolean variables. Whether a newly created step is an IEC step or not depends on whether the command “Extras”/“Use IEC-Steps” is activated. In order to use IEC steps you must link the special SFC library Iecsfc.lib into your project.

Actions

- An action can contain a series of instructions in IL or in ST, a lot of networks in FBD or in LD or again in SFC.
- With the simplified steps an action is always connected to a step. In order to edit an action, click twice on the step to which the action belongs.
- New actions for IEC steps are created in the Object Organizer for an SFC POU with the ‘Add Action’ command in the context menu. Several actions can be assigned to an ICE step and the actions can be used repeatedly by several ICE steps.

- The actions associated with an IEC step are shown at the right of the step in a two-part box. The left field contains the qualifier, possibly with time constant, and the right field contains the action name respectively Boolean variable name.
- IEC actions are scattered in step. IEC actions can be reused many times within a POU to which the actions belong. Add actions using the command “Extras”/“Associate Action” and delete the added actions using the command “Extras”/“Clear Action/Transition”.
- An example of IEC step with two actions is shown in figure 9-4-2.

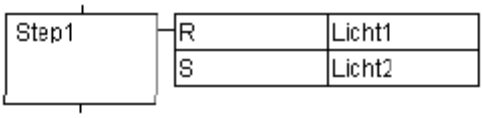


Figure 9-4-2 Example in SFC

Entry Actions and Exit Actions

- You can add an entry action and an exit action to a step by the commands “Add Entry-Action” and “Add Exit-Action” respectively.
- An entry action is executed only once right after the step has become active. An exit action is executed only once just before the step is deactivated. A step with an entry action is indicated by an “E” in the lower left corner, the exit action by an “X” in the lower right corner. The entry and exit action can be implemented in any language. In order to edit an entry or exit action, double click in the corresponding corner in the step.
- An example of a step with entry and exit actions is shown in figure 9-4-3.

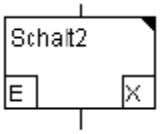


Figure 9-4-3 Example of a Step with Entry and Exit Action

Transitions/ Transition conditions

- Between the steps there are so-called transitions. When the step transition condition is TRUE, the transition is executed. When the previous step action is executed, if there is an exit action then execute it and if there is an entry action then execute it. Next, execute all the actions of the active step according to control cycle.
- A transition condition consists of a Boolean variable, a Boolean address, a Boolean constant or a Boolean result written in other programming languages.

Active steps

- After calling the SFC POU, the action (surrounded by a double border) belonging to the initial steps executed first. A step whose action is being executed is called active.
- The actions which belong to active step are executed at each cycle. In online mode active steps are shown in blue. In order to make it easier to follow the processes, all active actions in online mode are shown in blue like the active steps. After each cycle a check is made to see which actions are active. In a control cycle all actions are executed which belong to active steps. Thereafter the respective following steps of the active steps become active if the transition conditions of the following steps are TRUE. The currently active steps will be executed in the next cycle.

Qualifiers

- In order to associate the actions with IEC steps, qualifiers are needed. The use of IEC steps is shown in figure 9-4-4 and the meaning of qualifier is listed in table 9-4-1.

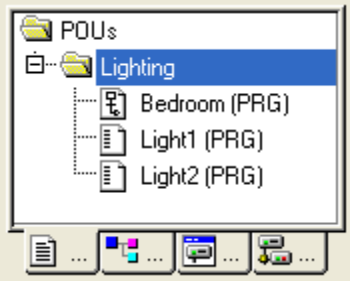


Figure 9-4-4 Use of IEC Steps

Qualifier	Name	Description	Comments
N	Non-stored	Normal type: the action is executed as long as the step is active and stops when the step exits.	
S	Set (Stored)	Set type: the action is activated and remains active until a Reset. Before a Reset any qualifiers even P cannot stop the action.	
R	Reset	Reset type: the action is deactivated as long as the step is activated.	
L	Time limited	Time limited: The action is activated for a certain time, maximum as long as the step is active. The action starts to be executed as long as the step is activated; At the same time timing starts and the action stops executing when the set time is arrived. If the set time is not arrived but the transition condition is TURE then the action stops executing too.	TIME constant (set value)
D	Time delayed	Time delayed: timing starts when the step is activated and the action starts to be executed when the set time is arrived. If the set time is not arrived but the transition condition is TURE in this situation the action will not be executed.	TIME constant (set value)
P	pulse	Pulse type: the action is executed just one time if the step is active.	
SD	Stored and time delayed	Stored and time delayed: timing starts when the step is activated and the action starts to be executed when the set time is arrived; When the step exits the action will continue to be executed until it's reset by R qualifier. Before the R qualifier no qualifier can stop the action (including P qualifier). If the set time is not arrived but the transition condition is TURE, in this situation timing continues until arriving the set value, then the action will be executed no matter it's a current step or not the original step.	TIME constant (set value)
DS	Delayed and stored	Delayed and stored: timing starts when the step is activated and the action starts to be executed when the set time is arrived; When the step exits, the action will continue to be executed until it is reset by R qualifier. Before the R qualifier no qualifier can stop the action (including P qualifier). If the set time is not arrived but the transition condition is TURE, in this situation the action will not be executed.	TIME constant (set value)
SL	Stored and time limited	Stored and time limited: The action is activated for a certain time. The action starts to be executed as long as the step is activated; At the same time timing starts and the action stops executing when the set time is arrived. If the set time is not arrived but the transition condition is TURE then the action will continue to be executed until the set time is arrived. The qualifiers L, D, SD, DS and SL need a time value in the TIME constant format.	TIME constant (set value)

Table 9-4-1 Qualifiers in SFC

Implicit Variables in SFC

There are implicitly declared variables in the SFC which can be used. A flag belongs to each step which stores the state of the step.

- The step flag is called <StepName> for the simplified steps or < StepName >.x for IEC steps.
- The step flag has the value TRUE when the associated step is active and FALSE when it's inactive.
- An enquiry with variable <ActionName>.x can be made to determine whether an IEC action is active or not. For IEC steps, the implicit variables <StepName>.t can be used to enquire about the active time of the steps

SFC flags

In SFC the active time of the steps depends on the time defined in the step attributes and sometimes a flag will be set. At the same time a variable can be set to control program flow in SFC. To read the flags you have to define appropriate global or local variables as inputs or outputs.

- SFCEnableLimit: The variable is of the type BOOL. When it has the value TRUE, the timeouts of the steps will be registered in SFCWError. Other timeouts will be ignored.
- SFCInit: The variable is of the type BOOL. When it has the value TRUE the sequential function chart is set back to the Init step. The other SFC flags are reset too (initialization). The Init step remains active, but is not executed, for as long as the variable has the value TRUE. It is only when SFCInit is again set to FALSE that the block can be processed normally.
- SFCQuitError: The variable is of the type BOOL. When it has the value TRUE the execution of the SFC diagram is stopped. A possible timeout in the variable SFCError is reset. All previous times in the active steps are reset when the variable again assumes the value FALSE.
- SFCPause: The variable is of the type BOOL. When it has the value TRUE the execution of the SFC diagram is stopped.
- SFCError: This Boolean variable is TRUE when a timeout has occurred in a SFC diagram. If another timeout occurs in a program after the first one, it will not be registered unless the variable SFCError is reset first.
- SFCTrans: This Boolean variable takes on the value TRUE when a transition is actuated.
- SFCErrorStep: This variable is of the type STRING. If SFCError registers a timeout, in this variable is stored the name of the step which has caused the timeout.
- SFCErrorPOU: This variable of the type STRING contains the name of the block in which a timeout has occurred.
- SFCCurrentStep: This variable is of the type STRING. The name of the step is stored in this variable which is active, independently of the time monitoring. In the case of simultaneous sequences the step is stored in the branch on the outer right.

9.4.2 Operation Descriptions

Selecting Elements

- You can select an element by pointing the mouse on the element and pressing the left mouse button, or you can use the arrow keys. Press <Shift>to mark a group of several elements. Please note that a step can only be deleted together with the preceding or the succeeding transition.

“Insert”/“Step-Transition (before)”:  , shortcut: <Ctrl>+<T>

- This command inserts a step in the SFC editor followed by a transition in front of the marked block.

“Insert”/“Step-Transition (after)”:  , shortcut: <Ctrl>+<E>

- This command inserts a step in the SFC editor followed by a transition after the first transition in the marked block.

“Insert”/“Alternative Branch (right)”:  , shortcut: <Ctrl>+<A>

- This command inserts an alternative branch in the SFC editor as a right branch of the marked block. For this the marked block must both begin and end with a transition. The new branch is then made up of one transition.

**“Insert”/ Alternative Branch (left)”:  **

- This command inserts an alternative branch in the SFC editor as the left branch of the marked block. For this the marked block must both begin and end with a transition. The new branch is then made up of one transition.

“Insert”/“'Parallel Branch (right)”:  , shortcut: <Ctrl>+<L>

- This command inserts a parallel branch in the SFC editor as the right branch of the marked block. For this the marked block must both begin and end with a step. The new branch is then made up of one step. To allow jumps to the parallel branches that have been created, these must be provided with a jump label.

**“Insert”/“Parallel Branch (left)”:  **

- This command inserts a parallel branch in the SFC editor as the left branch of the marked block. For this the marked block must both begin and end with a step. The new branch is then made up of one step. To allow jumps to the parallel branches that have been created, these must be provided with a jump label.

“Insert”/“Jump”:  , shortcut: <Ctrl>+<U>

- This command inserts a jump in the SFC editor at the end of the branch, to which the marked block belongs. To do this the branch must be an alternative branch. The inserted text string 'Step' in the inserted jump can then be selected and replaced by the step name or the jump label of a parallel branch to be jumped to.

**“Insert”/“Transition-Jump”:  **

- This command inserts a transition in the SFC editor, followed by a jump at the end of the selected branch. To do this the branch must be a parallel branch. The inserted text string 'Step' in the inserted jump can then be selected and replaced by the step name or the jump label of a parallel branch to be jumped to.

"Insert"/"Add Entry-Action"

- With this command you can add an entry action to a step. An entry-action is only executed once right after the step has become active. The entry-action can be implemented in a language of your choice. A step with an entry-action is designated by an "E" in the bottom left corner.

"Insert"/"Add Exit-Action"

- With this command you can add an exit action to a step. An exit-action is only executed once, before the step is deactivated. The exit-action can be implemented in a language of your choice. A step with an exit-action is designated by an "X" in the lower right corner.

"Extras"/"Paste Parallel Branch (right)"

- This command pastes the contents of the clipboard as a right parallel branch of the marked block. For this the marked block must both begin and end with a step. The contents of the clipboard must, likewise, be an SFC block that both begins and ends with a step.

"Extras"/"Add label to parallel branch"

- In order to provide a newly inserted parallel branch with a jump label, the transition occurring before the parallel branching must be marked and the command "Add label to parallel branch" must be executed. At that point, the parallel branch will be given a standard name consisting of "Parallel" and an appended serial number, which can be edited. In figure 9-4-5, "Parallel" is replaced by "abc". If you want to delete jump label, delete label name.

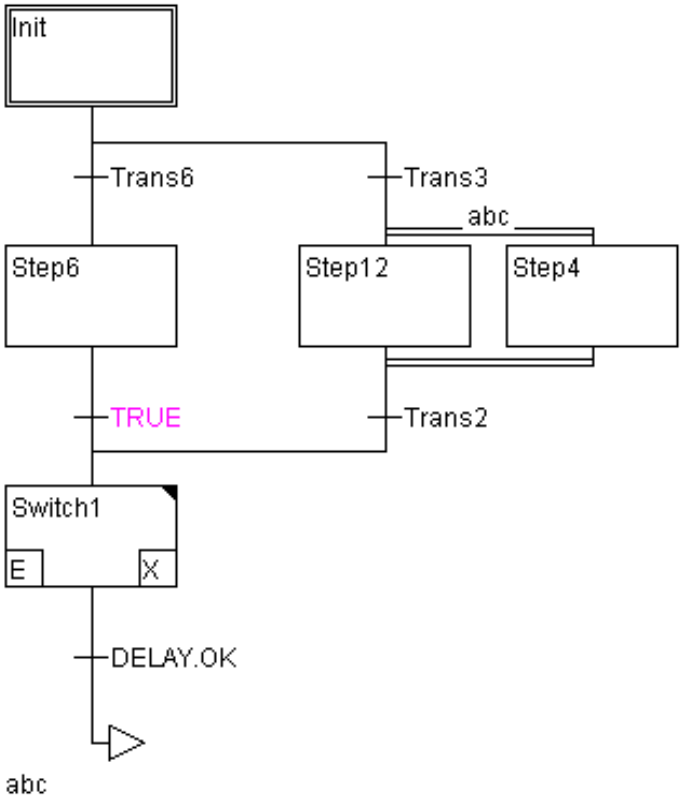


Figure 9-4-5 Add Label to Parallel Branch

"Extras"/"Paste after"

- This command pastes the SFC block on the clipboard after the first step or the first transition of the marked block. Normal copying pastes it in front of the marked block ("Edit"/"Paste").

"Extras"/"Zoom Action/Transition", shortcut: <Alt>+<Enter>

- The action of the first step of the marked block or the transition body of the first transition of the marked block is loaded into the editor in the respective language, in which it has been written. If the action or the transition body is empty, then the language must be selected, in which it has been written.
- Regard that the transition condition which is written within the editor window will take precedence over a condition which might be written directly at the transition mark. Example in figure 9-4-6: If here i>100, then the transition condition will be FALSE, although TRUE has been entered at the mark.

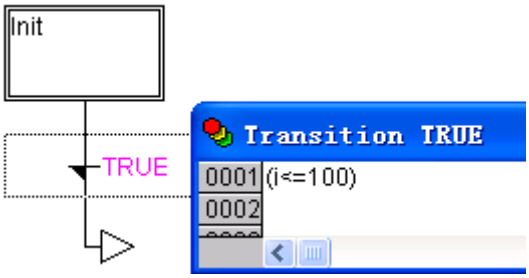


Figure 9-4-6 Priority of Transition Condition

"Extras"/"Clear Action/Transition"

- With this command you can delete the actions of the first step of the marked block or of the transitions body of the first transition.

Alternative Branch

- Two or more branches in SFC can be defined as alternative branches. Each alternative branch must begin and end with a transition. Alternative branches can contain parallel branches and other alternative branches. An alternative branch begins at a horizontal line (alternative beginning) and ends at a horizontal line (alternative end) or with a jump.
- If the step which precedes the alternative beginning line is active, then the first transition of each alternative branch is evaluated from left to right. The first transition from the left whose transition condition has the value TRUE is opened and the following steps are activated.

Parallel Branches

- Two or more branches in SFC can be defined as parallel branches. Each parallel branch must begin and end with a step. Parallel branches can contain alternative branches or other parallel branches. A parallel branch begins with a double line (parallel beginning) and ends with a double line (parallel end) or with a jump. It can be provided with a jump label.

- If the parallel beginning line of the previous step is active and the transition condition after this step has the value TRUE, the first steps of all parallel branches become active. These branches are now processed parallel to one another. The step after the parallel end line becomes active when all previous steps are active and the transition condition before this step produces the value TRUE.

Jumps

- A jump is a connection to the step whose name is indicated under the jump symbol. Jumps are required because it is not allowed to create connections which lead upward or cross each other
- Figure 9-4-7 illustrates an application example in SFC.

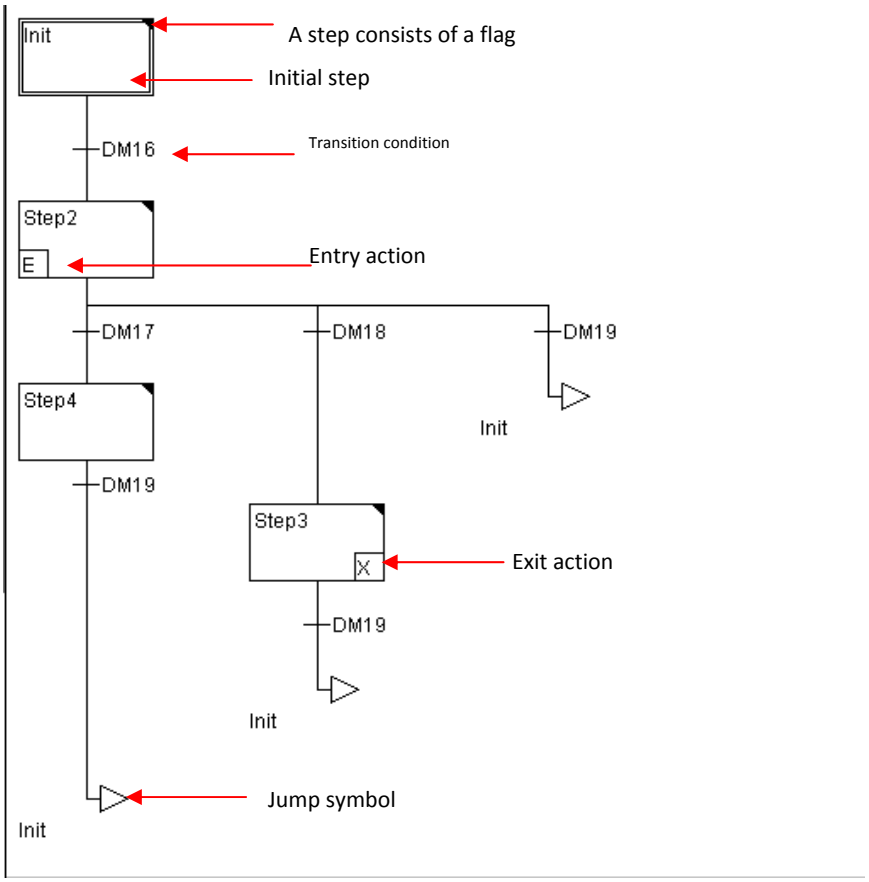


Figure 9-4-7 Explanation of SFC

9.5 CONTINUOUS FUNCTION CHART (CFC)

9.5.1 CFC Editor

CFC (Continuous Function Chart) is a graphically oriented language. CFC bases on the Function Block Diagram language. However it does not operate with networks, but rather with freely placeable elements.

CFC editor is a graphically oriented editor, as shown in figure 9-5-1 and you can edit through menu bar or the context menu. The elements can be placed anywhere. The inputs and outputs of these elements can be connected by dragging a connection with the mouse. The connecting line will be drawn automatically. The connecting lines are automatically adjusted when the elements are moved. If the case arises where a connecting line cannot be drawn simply because of lack of space, a red line will be shown between the input and the associated output instead. This line will be converted into a connecting line just as soon as space is available.

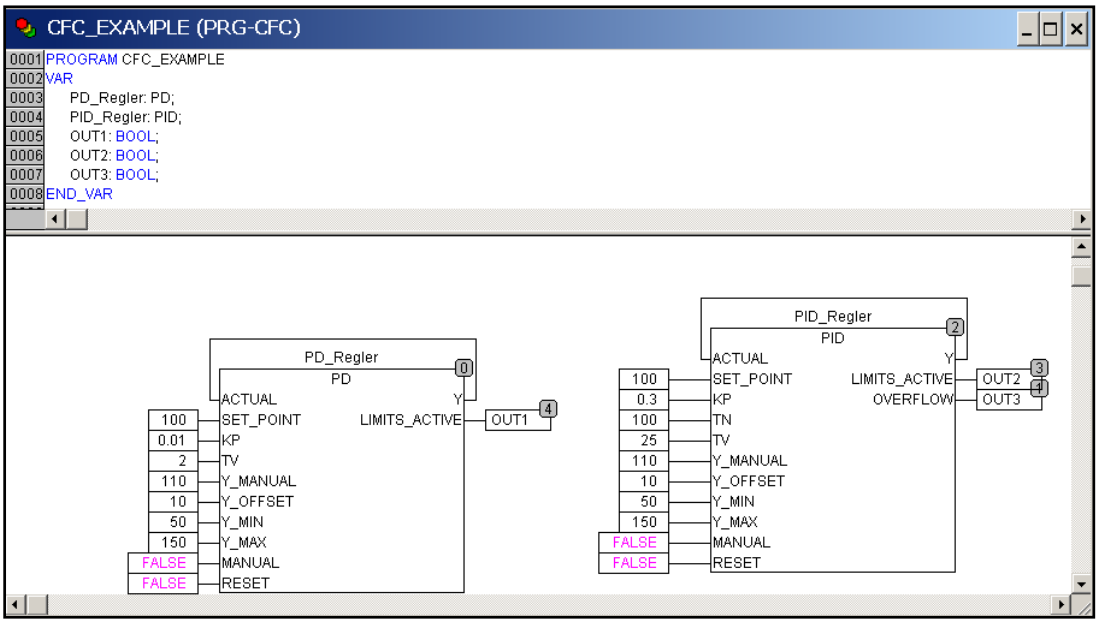
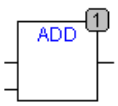
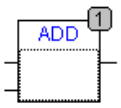
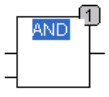
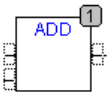
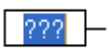
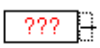


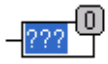
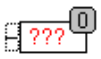
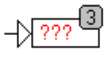
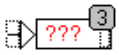
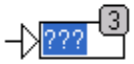
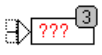
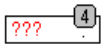
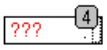
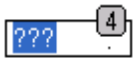
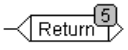
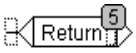
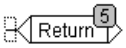
Figure 9-5-1 CFC Editor

9.5.2 Operation Descriptions

Elements of CFC include boxes, inputs, outputs, jumps, labels, returns and comments, where boxes include operators, functions, function blocks and programs.

Cursor Position

Name	Graph element	Selected element	Text fields	Inputs and outputs
Cursor position	Programming tool elements	Trunks of the elements	Shaded in blue or position of comments	Inputs and outputs
Box				
Input				

Output				
Jump				
Label				——
Return			——	
Comments				——

Selecting Elements

- Elements can be selected by clicking on their bodies. .
- More than one element can be selected by clicking in the elements required, one after the other, while holding down the <SHIFT> key, or by dragging over the elements to be marked. The command “Extras”/ “Select all” marks all elements at once.

Moving Elements

- One or more selected elements can be moved with the arrow keys while holding the <Shift> key. Another possibility is to drag elements while holding the left mouse button. These elements are placed by releasing the left button. If the elements cover other elements or exceed the foreseen size of the editor, the marked element jumps back to its initial position and moving fails.

Connecting Line

An input of an element can be precisely connected to the output of another element (the output of the element itself or other elements), an output of an element can be connected to the inputs of a number of other elements (the inputs of the element itself or other elements), as shown in figure 9-5-2. There are three possibilities to connect the input of an element E2 (ADD) with the output of an element E1 (a). Type testing is carried out during the connection. If the types of the two pins are not compatible, the cursor changes to “Forbidden” and the connection fails.

- Place the mouse on the output of element E1, click, hold the left button down and drag the mouse cursor onto the input of element E2 and let go.
- Place the mouse on the input of element E2, click, hold the left button down and drag the mouse cursor onto the output of element E1 and let go.
- Move one of the elements E1 or E2 and place it by letting go of the left button when the output of element E2 and the input of element E1 touch.

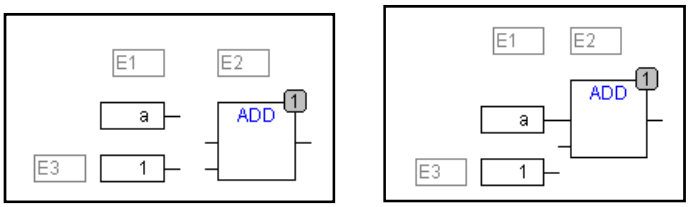


Figure 9-5-2 Connecting Line

Deleting Connecting Line

There are three possibilities for removing the connection between the output of an element E1 (a) and the input of an element E2 (ADD), as shown in figure 9-5-2. Select the output of element E1 and press the <Delete> key or execute the command “Edit”“Delete”. Several connections will be removed at the same if the output of E1 is connected to more than one of inputs. Select the input of element E2 and press the <Delete> key or execute the command “Edit”“Delete”. Select the input of E2 with the mouse, hold the left button depressed and drag the connection from the input to E2 away. The connection is removed when the left button is released in a free area of the screen.

“Insert”/“Box”:  , shortcut: <Ctrl>+<B>

- This command can be used to paste in operators, functions, function blocks and programs First of all, it is always inserted an “AND” operator. This can be converted by Selection and Overwrite of the text into every other operator, into every function, into every function block and every program. The input assistance serves to select the desired block from the list of supported blocks through selecting AND and pressing F2.

“Insert”/“Input”:  , shortcut: <Ctrl>+<I>

- This command is used to insert an input. The new inserted “Input” moves along with the mouse until clicking the left mouse after moving to an appropriate position. The text "???" can be selected and replaced by a variable or constant. The input assistance can also be used here.

**“Insert”/“Output”:  **

- This command is used to insert an output. The new inserted “Output” moves along with the mouse until clicking the left mouse after moving to an appropriate position. The text "???" can be selected and replaced by a variable. The input assistance can also be used here.

“Insert”/“Jump”:  , shortcut: <Ctrl>+<J>

- This command is used to insert a jump. The new inserted “Jump” moves along with the mouse until clicking the left mouse after moving to an appropriate position. The text "???" can be selected and replaced by the jump label to which the program should jump.

“Insert”/“Label”:  , shortcut: <Ctrl>+<L>

- This command is used to insert a label. The new inserted “Label” moves along with the mouse until clicking the left mouse after moving to an appropriate position. The text "???" can be selected and replaced by the jump label. In Online mode a RETURN label for marking the end of POU is automatically inserted.

“Insert”/“Return”:  , shortcut: <Ctrl>+<R>

- This command is used to insert a RETURN command. The new inserted “Return” moves along with the mouse until clicking the left mouse after moving to an appropriate

position. Note that the “Return” here is different with the RETURN label in online mode which is to execute next cycle automatically after execution leaves the POU.

“Insert”/“Comment”: , shortcut: <Ctrl>+<K>

- This command is used to insert a comment. You obtain a new line within the comment with <Ctrl> + <Enter>. The new inserted comment moves along with the mouse until clicking the left button after moving to an appropriate position.

“Extras”/“Order”/“Show Order”

- This command switches the display of the order of execution on and off. The default setting is to show it (recognised by a tick (√) in front of the menu point).

“Extras”/“Order”/“Order topologically”

- Elements are ordered in a topological sequence when the execution takes place from left to right and from above to below, that is the number increases from left to right and from above to below for topologically arranged elements. Regard that the number appears in the upper right corner.

“Extras”/“Order”/“Order everything according to data flow”

- With this command the order of execution is determined by the data flow of the elements and not by their position.

An application example in CFC is shown in figure 9-5-3.

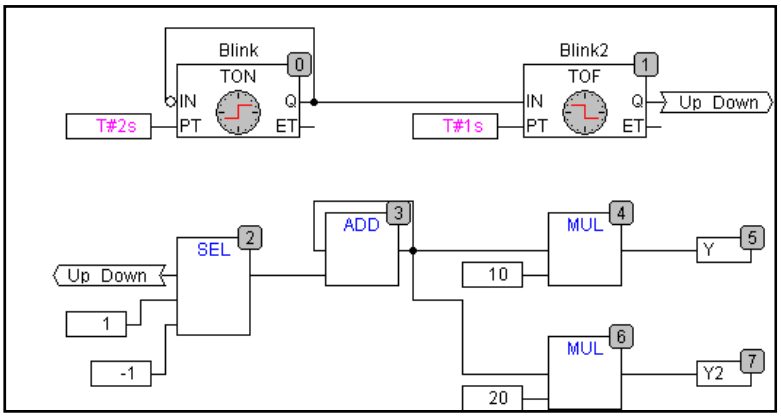


Figure 9-5-3 Example in CFC

Chapter 10

Special Functions

10.1 MODBUS COMMUNICATION

LM-serial PLC can communicate with third party equipment, such as touch screens and configuration software, through Modbus protocol. LM-serial PLC is considered as Modbus RTU slave in default.

RS232 port and RS485 port of LM-serial PLC both support Modbus RTU slave protocol.

10.1.1 Modbus Overview

Modbus protocol is a master/slave communication protocol that can be used on different physical layers (RS485 or RS232). Devices communicate on a Modbus serial line. Data rate can reach 115kbps and can connect one master and up to 247 slaves in theory. Maximum one master and 32 slaves are physical media and devices dependent.

Some of the features of Modbus protocol are fixed, such as frame format, frame sequence, communication error and processing of exception and perform functions, which can't be changed casually. The other features are optional for users, such as transmission media, baud rate, parity check, the number of stop-bit and so, transmission mode is RTU (Remote Terminal Unit). The selected parameters by users must be the same in each station and can't be changed when the system runs.

10.1.2 Modbus Communication Function

The communication function codes of Modbus RTU supported by LM Micro series PLC are listed in table 10-1-1.

Function code	Name	Meaning (for master)
01	Read coil status	Read a group of coils status
02	Read input status	Read a group of inputs status
03	Read holding register	Read a group of holding registers status
04	Read input register	Read a group of input registers status
05	Force single coil	Force single coil
06	Preset single register	Preset single register
15	Force multiple coils	Force multiple coils
16	Preset multiple registers	Preset multiple registers

Table 10-1-1 Modbus Function Code

Modbus RTU protocol can access the following data locations:

Input (I), Output (Q), Memory location (M)

The three data locations can be accessed through BOOL or WORD variables. The mapping between these data locations and Modbus protocol addresses is shown in table 10-1-2.

Data location		Type	Address range	Modbus address	Mapping formula	Modbus Data type
I location	%IX	BOOL	%IX0.0~%IX511.7	0~4095	IXm.n: m*8+n	1x
	%IW	WORD	%IW0~%IW510	0~255	IWm: m/2	3x
Q location	%QX	BOOL	%QX0.0~%QX511.7	0~4095	QXm.n: m*8+n	0x
	%QW	WORD	%QW0~%QW510	0~255	QWm: m/2	4x
M location	%MX	BOOL	%MX0.0~%MX7816.7	3000~65535	MXm.n: m*8+n+3000	0x
	%MW	WORD	%MW0~%MW8190	3000~7095	MWm: m/2+3000	4x

Table 10-1-2 The mapping between the data locations of LM Micro series PLC and Modbus protocol addresses

Some HMI data address starts with 1, if Modbus RTU is used to communicate with PLC plus1on the mapping formula when filling in the data address. For example the mapping address of %MX100.0 is 100*8+0+3000+1=3801. This type of HMI contains the touch screen of Eview, MCGS, Weinview and the configuration software of King View and Sunwayland Forcecontrol. However for some HMI data address there is no need to add 1 on the mapping formula, for example: Hitech.

Note:
1. The size of M is 8K and can accessed by BOOL variables from %MX0.0 to %MX8191.7, but the maximum range is 3000~65535 according to Modbus protocol, so the maximum that can be accessed is %MX7816.7.
2. The range of I and Q locations in the above list is the largest possible range and calculate the range according to the actual configuration. The datas of nonexistent I and Q locations can't communicate.

10.1.3 Application of Modbus Communication

RS232 and RS485 of LM Micro series PLC can be set as Modbus RTU slave protocols independently. Before communication in PLC port the slave address and communication parameters must be set. For LM Micro series PLC the default slave address is 51 and communication parameter is 38400, n, 8, 1.

Use the SET_LOCAL_ADDRESS instruction to set PLC slave address. Use the Reset_COMM_PRMT instruction to set the communication parameters of RS232 and use the Reset_COMM2_PRMT instruction to set the communication parameters of RS485. Refer to<<Instruction Manual>>for details.

An application of which slave address is 5, RS232 baudrate is 9600, data bit is 8, stop bit is 1 and no check is shown in figure 10-1-1.

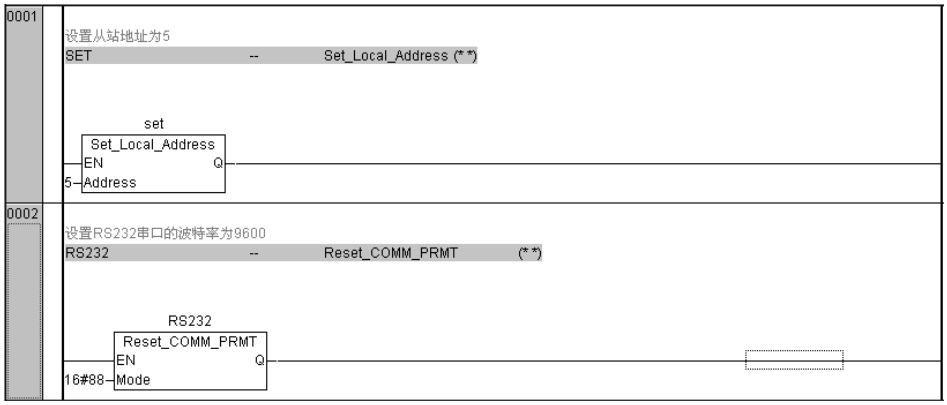


Figure 10-1-1

10.2 INTERRUPTS

10.2.1 Overview of Interrupts

When PLC receives a special signal from external or internal hardware, the CPU stops the current work and starts to process the event and returns to the previous work after the event has been processed. The entire process is called an interrupt. The signal which generates an interrupt is called an interrupt source.

LM serried PLC can handle a plenty of interrupt signals generated by interrupt sources. The interrupt sources have different priorities and the interrupt with higher priority will be executed when two interrupts arrive at the same time.

The interrupt sources that can be processed in LM Micro series PLC is listed in table 10-2-1.

Name	Description
Start	Called when program starts
Stop	Called when program stops
Debug_loop	Called when debug loop runs
Taskcode not called	Called when taskcode is not called
Fast External 0 interrupt	Called when fast external event 0 interrupt runs
Fast External 1 interrupt	Called when fast external event 1 interrupt runs
Fast External 2 interrupt	Called when fast external event 2 interrupt runs
Fast External 3 interrupt	Called when fast external event 3 interrupt runs
HD_TC7 interrupt	Called when Hardware Timer/Counter 7 interrupt runs
HD_TC2 interrupt	Called when Hardware Timer/Counter 2 interrupt runs
HD_TC3 interrupt	Called when Hardware Timer/Counter 3 interrupt runs
HD_TC4 interrupt	Called when Hardware Timer/Counter 4 interrupt runs
HD_RTC_ALM0 interrupt	Called when Hardware Real/Time Alarm 0 interrupt runs
PTO_0 Finished interrupt	Called when PTO_0 Finished interrupt runs
PTO_1 Finished interrupt	Called when PTO_1 Finished interrupt runs

Table 10-2-1 Interrupt Events

10.2.2 Applications of Interrupts

If you want to use an interrupt, there are two tasks that must be done. First, an interrupt service routine has to be written. The interrupt service routine is a program which will be executed when an interrupt is generated. Writing an interrupt service routine is similar to a subprogram. More details can be found in section 7.4.7. Second, after the interrupt service routine is done, the interrupt and the related interrupt service routine must be configured. Double click “Task configuration” in “Resources” tab and a dialog window is opened on the right, then click the “System events” in “Task configuration” tree and all the available system events are displayed, as shown in figure 10-2-1. If you actually want the POU to be called by the event, activate the entry in the assignment table (V) and in the column “called POU” can enter the name of the project POU which should be called and processed as soon as the event occurs.

The usage of interrupt is displayed in the following example.

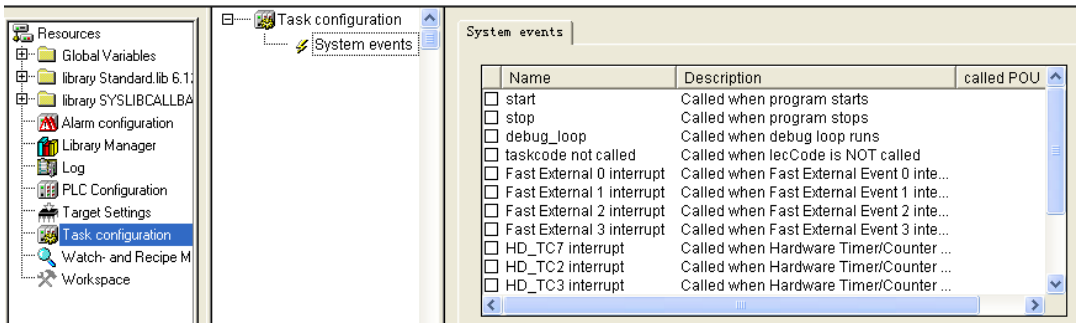


Figure 10-2-1 System Events

Example

Requirements

- Realize the following functions not influenced by PLC scanning period:
- At each rising edge of I0.6, PLC responds the pulse immediately and generates an interrupt and the value in %MW100 increases by 10.
- At each rising edge of I0.7, PLC responds the pulse immediately and generates an interrupt and the value in %MW102 increases by 10.

Program analysis

- According to the requirements above, select LM3106 CPU module and the following instructions are used in the software:
- Fast_ExINT_E (fast external interrupt)
- The program is divided into three parts:
- Main program—define the fast external interrupt mode of CPU module LM3106.
- INT3PRO—interrupt program which is executed at the rising edge of I0.6 and the value in %MW100 increases by 10.
- INT2PRO—interrupt program which is executed at the rising edge of I0.7 and the value in %MW102 increases 15.

Programming

Main program:

First add the Hollysys_PLC_Ex_ExINT.lib used in the program to library manager. According to the requirement, an interrupt is generated at each rising edge of I0.6 and I0.7, and I1.0 is not used, then the Fast_ExINT_E Mode=16#50, the variable declarations in main program and LD are shown in figure 10-2-2.

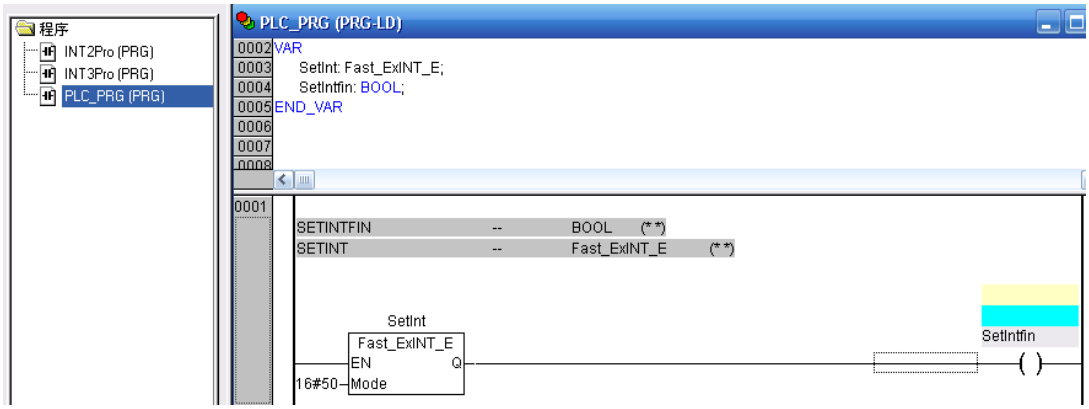


Figure 10-2-2 Variable Declarations and LD of Main Program

Because Fast External Event 3 (I0.6) and Fast External Event 2 (I0.7) are used, activate them by clicking on the control box, as shown in figure 10-2-3.

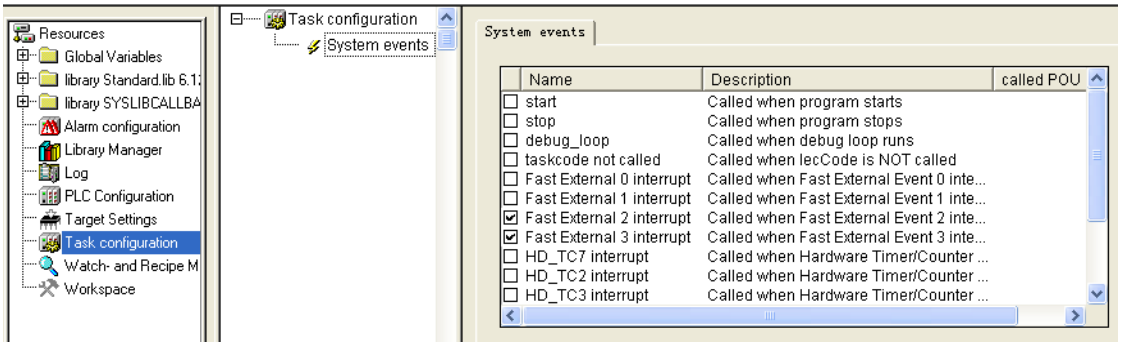


Figure 10-2-3 Choice of Fast External Event

Create subprograms INT2Pro and INT3ProFast after External 2 interrupt and Fast External 3 interrupt and click Create POU respectively, and the two interrupt programs are created successfully, as shown in figure 10-2-4.

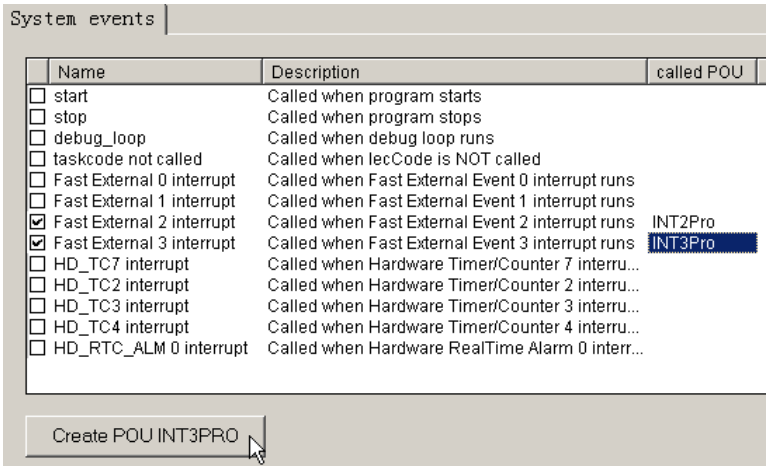


Figure 10-2-4 Create Interrupt Program

The default language of the created interrupt programs is ST, and you can convert it to LD. Before this the project must have been compiled without any error, as shown in figure 10-2-5.

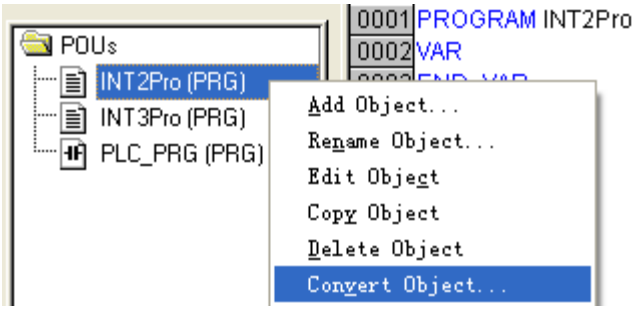


Figure 10-2-5 Convert ST to LD

The LD of INT3Pro is shown in figure 10-2-6.

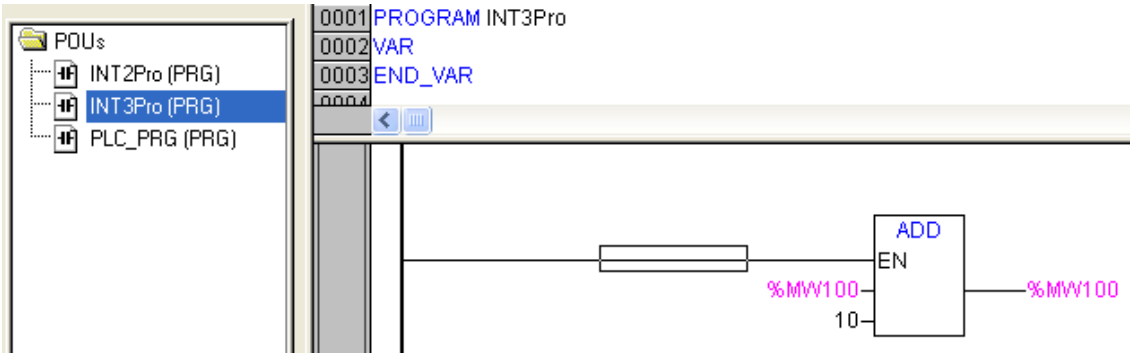


Figure 10-2-6 LD of INT3Pro

The LD of INT2Pro is shown in figure 10-2-7.

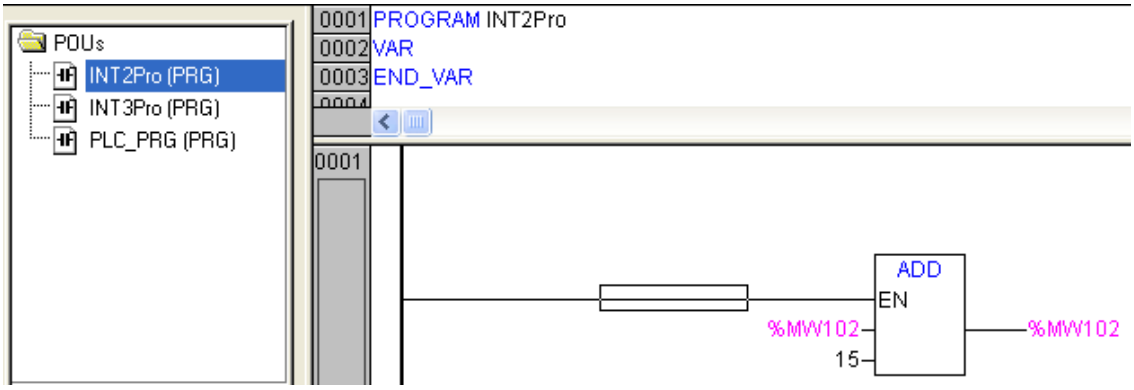


Figure 10-2-7 LD of INT2Pro

Next, you can write the program in INT2Pro and INT3Pro. Of course you can write the program in ST language not converting it to LD. Call the related program when the system event is triggered.

Note that system event is not supported in simulation mode and it will be responded only when the program has been compilation without any error and login. When multi-tasks have been configured at one time, you are requested to rebuild all and then save the file.

10.3 ANALOG FUNCTIONS

10.3.1 ANALOG MODULE ADDRESSING

When analog modules are used in LM Micro series PLC, first you need to know the address occupied by this module. No matter it is an analog input or analog output, it will occupy PLC input area or output area. In PLC configuration when an analog module is added, the data address of this module will be given automatically.

Take LM3310 for example. As shown in figure 10-3-1, you can configure a 4×AI module LM3310 after LM3107, and the addresses %IW2, %IW4, %IW6, %IW8 will be assigned automatically and each word stands for a channel acquisition data.

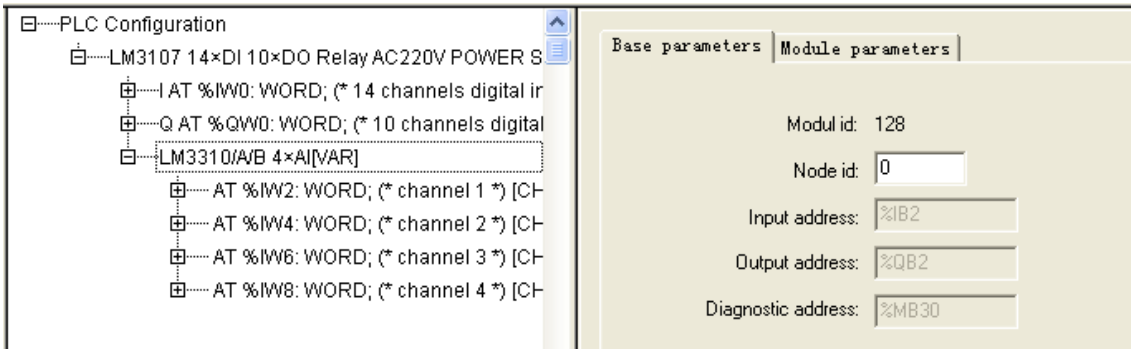


Figure 10-3-1 Configuration of AI Module

For AO modules, the address in Q will be assigned automatically. Such as %QW2, if you want to output an analog value, assign a value to %QW2.

In programs these addresses can denote channel values directly and the relationships between channel values and their ranges are shown in the table below.

Modules	Range	Scope
LM3107E	Input: 0~10V (voltage), 0~20mA (current)	0~10000
	Output: 0~10V (voltage), 0~20mA(current)	0~4095
LM3310/A/B	Input: 0~10V (voltage), 0~20mA(current), 4~20mA(current)	0~65535
LM3320	Output:0~10V (voltage), 0~20mA(current)	0~4095
LM3313	Input: -10~10V (voltage) , -20mA~20mA(current)	-32000~32000
LM3330	Input: 0~10V (voltage), 0~20mA(current), 4~20mA(current)	0~65535
	Output:0~10V (voltage), 0~20mA(current)	0~4095

Table10-3-1 Range and Scope of AI Module

10.3.2 USE OF ANALOG MODULES

- If analog input expansion module is used in PLC configuration, call the instruction ANALOG_IN. The input value on Address of the instruction must be the same with the node id of AI module. The input value on Address of LM3310 is set 0, as shown in figure 10-3-1.
- If you want to use several Analog_IN modules, it is needed to configure several Analog_IN instructions and the address of each Analog_IN must be consistent with the corresponding module node id.

- If analog output expansion module is used in PLC configuration, call the instruction ANALOG_OUT. The input value on Address of the instruction must be the same with the node id of AO module.
- If you want to use several Analog_OUT modules, it is needed to configure several Analog_OUT instructions and the address of each Analog_OUT must be consistent with the corresponding module node id.

Refer to<<Instruction Manual>>for detailed instruction usage.

Configure the module parameters after the analog modules have been configured. The module which is set correctly can be used correctly. The parameter settings include Filter_Factor, XFactor and Channel_Enable. Refer to<<Hardware Manual>>for detailed parameter setting. You can also refer to section 7.3.2 about analog module configuration.

Note:
1. For LM3107E there is no need to add ANALOG_IN and ANALOG_OUT when using analog processing.
2. For LM3330 add ANALOG_IN and ANALOG_OUT when using analog input and analog output at the same time

10.3.3 Application of Analog Modules

Take LM3330 for an example to describe the usage of analog module. LM3330 is a module with 4×AI and 1×AO. The first channel of LM3330 is used to acquire the value of pressure transmitter with the range 4~20mA and the output channel is used to control the valve opening with the range 0~10V. Select LM3107 as CPU module.

PLC configuration is shown in figure 10-3-2 and the Module id is 0.

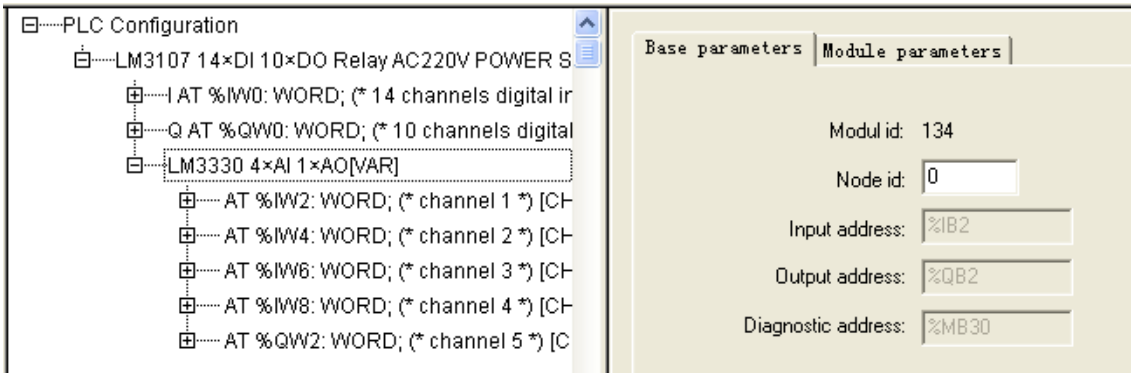


Figure 10-3-2 LM3330 Configuration

From the PLC configuration we can see that the input channels of LM3330 occupy the addresses %IW2, %IW4, %IW6, %IW8 and the output channel occupies the address %QW2. Configure the module parameters after LM3330 has been configured. Set the Filter_Factor 16 for steady acquisition data, as shown in figure 10-3-3.

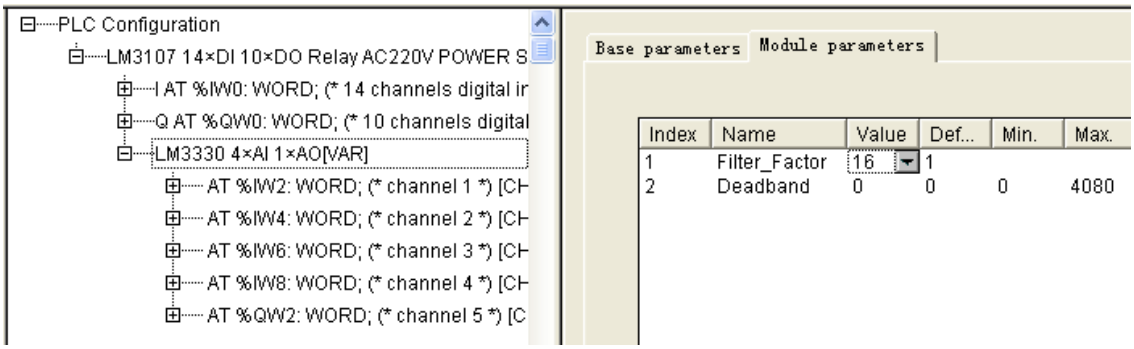


Figure 10-3-3 Set Filter Factor

Configure the range of input channel and output channel after setting the Filter_Factor, as shown in figures 10-3-4 and 10-3-5. Configure the the range of the first channel %IW2 4~20mA and output channel %QW2 0~10V.

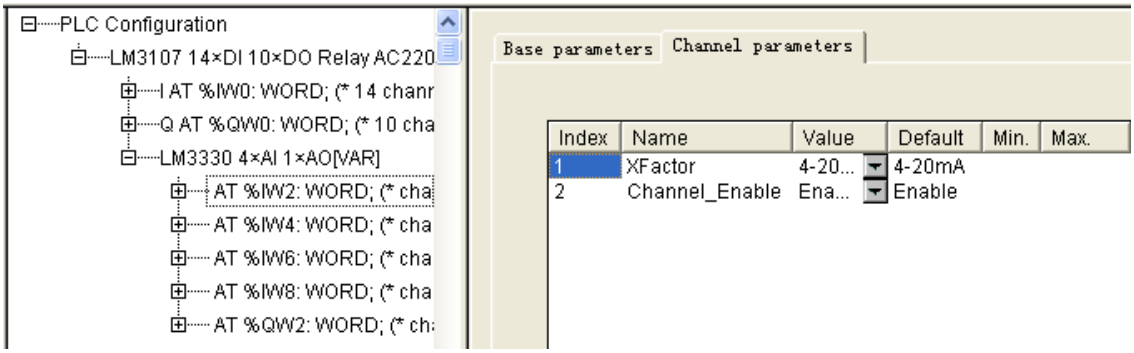


Figure 10-3-4 Set the Range of AI Channel

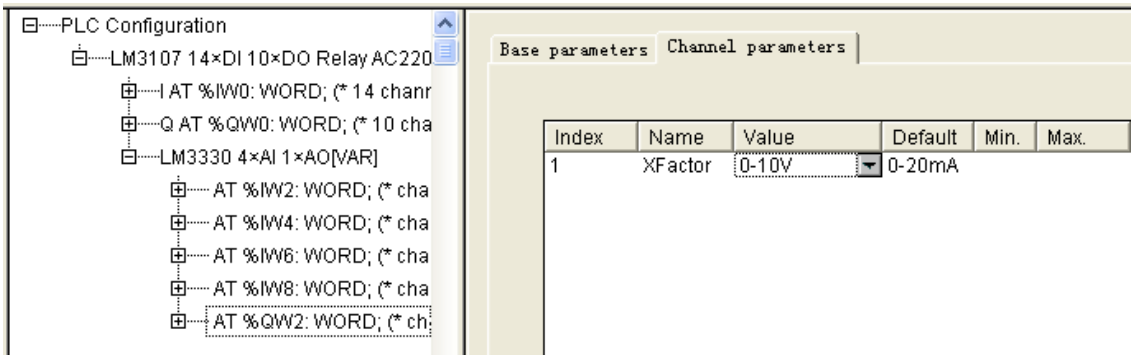


Figure 10-3-5 Set the Range of AO Channel

Start programming after the configuration is finished. First add the instructions ANALOG_IN and ANALOG_OUT and set the parameter Address 0. Before using the two instructions add the related library Hollysys_PLC_analog.lib. Refer to section 7.4.4 about library operations.

The range of AI channel is 4~20mA and the corresponding value is 0~65535 as shown in table 10-3-1, so the scope of %IW2 is 0~65535 in which 0 stands for the 4mA acquired current signal and 65535 stands for the 20mA acquired current signal.

The range of AO channel is 0~10V and the corresponding value is 0~4095 as shown in table 10-3-1, so the scope of %QW2 is 0~4095 in which 0 stands 0V voltage signal and 4095 stands for 10V voltage signal.

The program is shown in figure 10-3-6. Analog_IN is inserted in network 1 and Analog_OUT is inserted in network 2. The read and setting of analog input and analog output is displayed

in network 3. Here another instruction H_E is used to convert Hexadecimal got from %IW2 to EU and save the result in pressure variable. The scope of %IW2 is 0~65535 and the data type is INT, while the scope of pressure is 4~20 mA and the data type is REAL. Refer to<<Instruction Manual>>for the usage of H_E instruction.

Assign the variable output to %QW2 and LM3330 outputs 0~10V voltage when the scope of %QW2 is 0~4095.

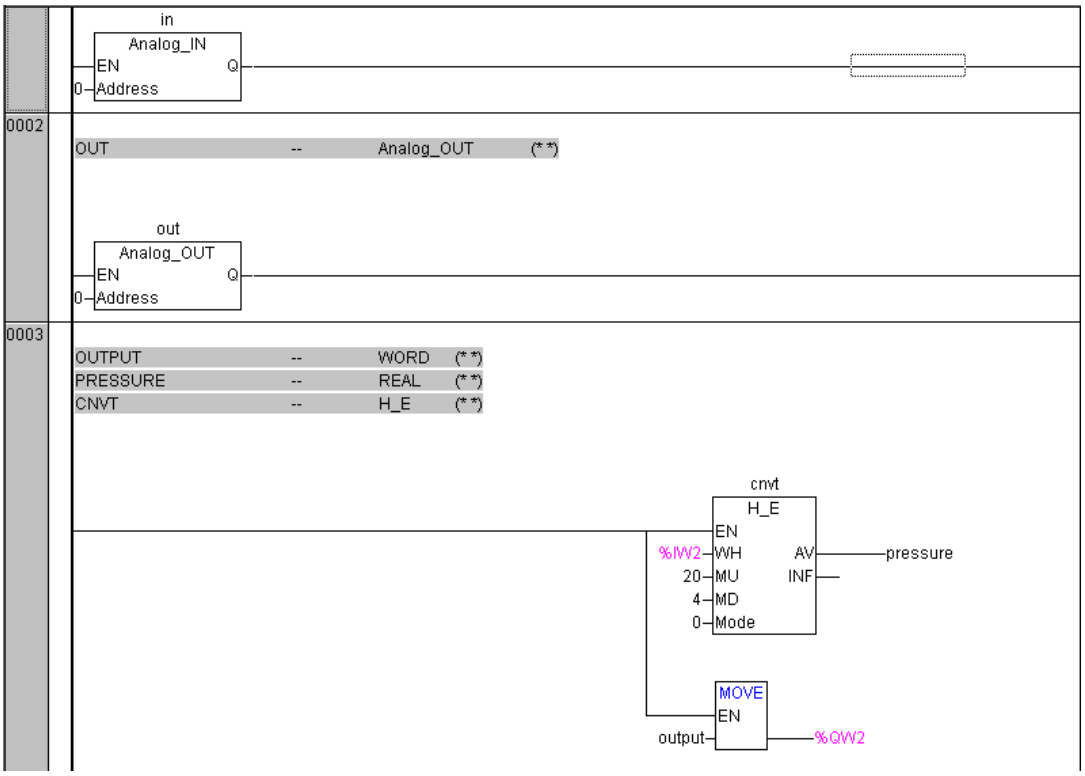


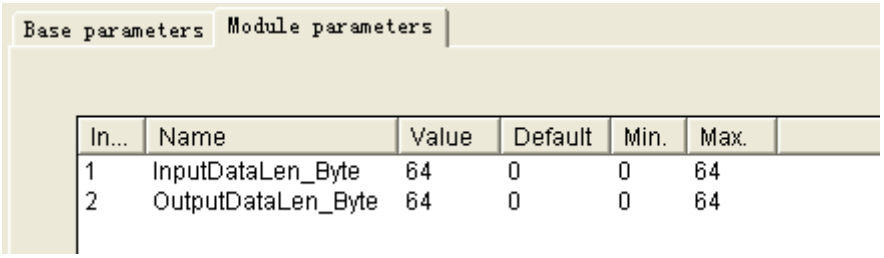
Figure 10-3-6 Program

10.4 DP COMMUNICATION

10.4.1 DP Communication Setting

LM Micro series PLC provides Profibus-DP communication processing. LM3401 is a DP slave module.

Configure the LM3401 parameters after the PLC configuration has been finished. The main parameters include InputDataLne_Byte and OutputDataLne_Byte. The parameter settings of LM3401 are shown in figure 10-4-1 and the Value ranges between 0-64. The Value is 64 in figure 10-4-1. Refer to<<Hardware Manual>>for detailed technical specifications of LM3401.



In...	Name	Value	Default	Min.	Max.
1	InputDataLen_Byte	64	0	0	64
2	OutputDataLen_Byte	64	0	0	64

Figure 10-4-1 LM3401 Module parameters

When using DP communication insert instruction DP_Slave, similar with analog processing. The program DP_Slave is displayed in figure 10-4-2, where the input value on Address is 0 which is consistent with the node id of LM3401 in PLC configuration list.

Scan DP slave when EN is set and cannot scan DP slave when EN is reset.

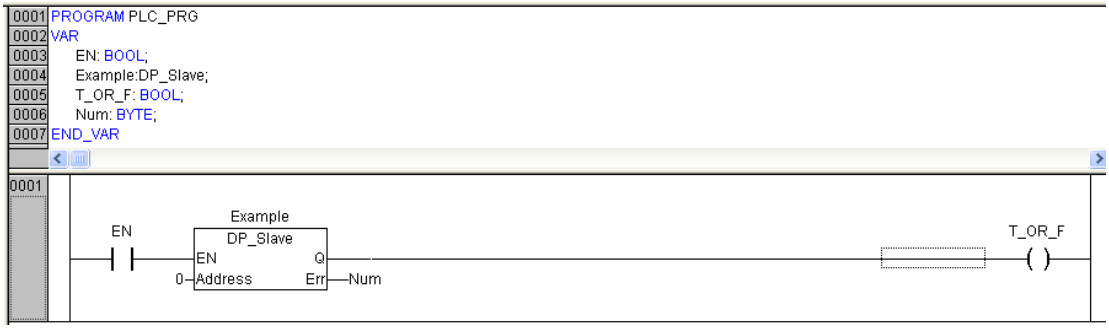


Figure 10-4-2 DP_Slave Function Block

System will assign input address and output address automatically after LM3401 has been configured, as shown in figure 10-4-3. The input address is 64 bytes starting from %IW2 and output address is 64 bytes starting from %QW2. The communication between the DP Master and LM Micro series PLC is completed through these inputs and outputs. Inputs are used to store the data from DP master and the outputs are used to store the data which is going to be sent to DP master from LM Micro series PLC.

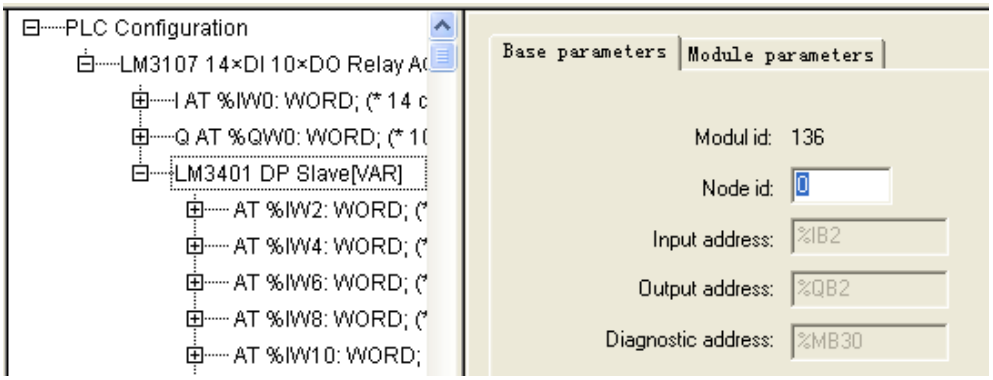


Figure 10-4-3 Configuration of DP_Slave

10.4.2 Example of DP Communication

A simple application example of Profibus-DP function block is as below.

Requirement: PLC sends out data of 8 bytes to DP master from DP slave and receives data of 8 bytes from DP master at the same time.

Variable Declarations:

```
PROGRAM PLC_PRG
VAR
    EN: BOOL;
    Example: DP_Slave;
    T_OR_F: BOOL;
    SendDataA: WORD; PLC sends data A to DP master
    SendDataB: WORD; PLC sends data B to DP master
    SendDataC: WORD; PLC sends data C to DP master
    SendDataD: WORD; PLC sends data D to DP master
```

RecDataA: WORD; DP master sends data A to PLC
RecDataB: WORD; DP master sends data B to PLC
RecDataC: WORD; DP master sends data C to PLC
RecDataD: WORD; DP master sends data D to PLC
END_VAR

Software Configuration:

The configuration of LM3401 is shown in figure 10-4-4.

- InputDataLen_Byte: the length of data sent to PLC by DP master, the number of bytes received is 8.
- OutputDataLen_Byte: the length of data sent to DP master by PLC, the number of bytes sent is 8.

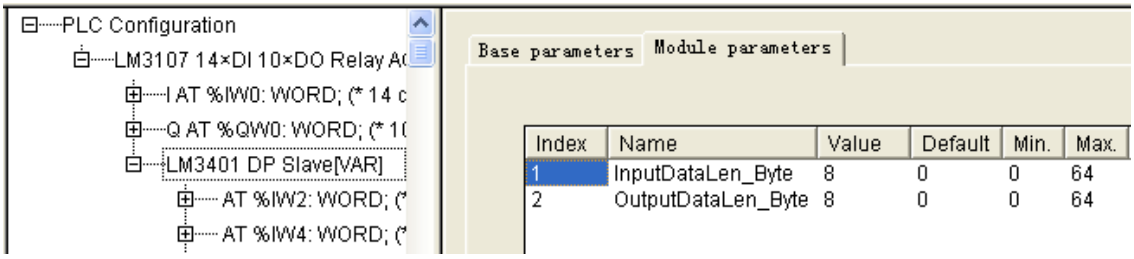


Figure 10-4-4 LM3401 Module parameters

The Address of DP_Slave should be consistent with node id in figure 10-4-5, and the data A, B, C, D sent to PLC by DP master are stored in %IW2, %IW4, %IW6 and %IW8.

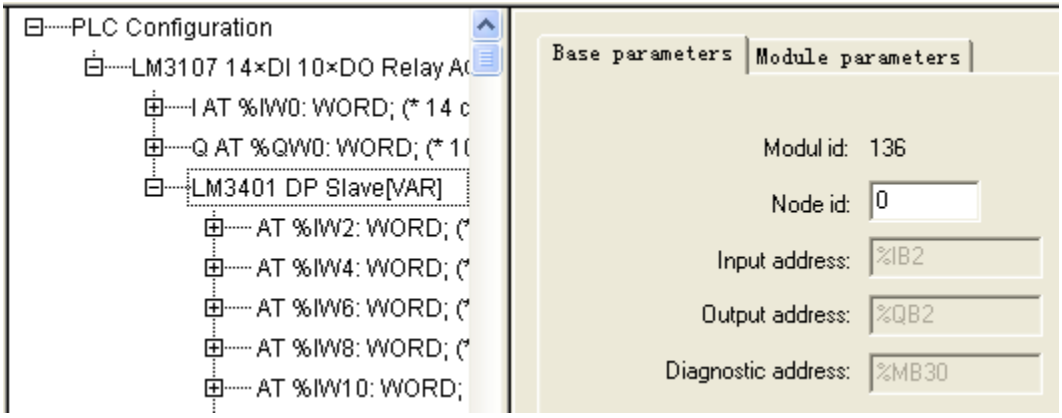


Figure 10-4-5 Configuration of LM3401

The data A, B, C, D sent to DP master by PLC is stored in %QW2, %QW4, %QW6, %QW8 as shown in figure 10-4-6.

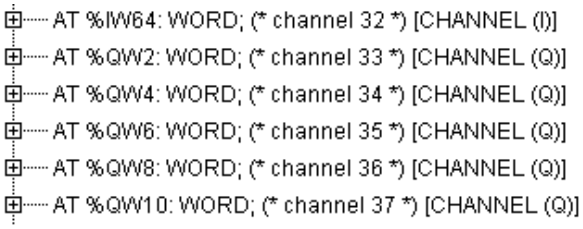


Figure 10-4-6 LM3401 Channels

Configure the receive area and send area of DP master so that the data in QW area of DP slave will send to the receive area of DP master and the data of DP master will send to IW area of DP slave automatically. The ladder diagram is shown in figure 10-4-7.

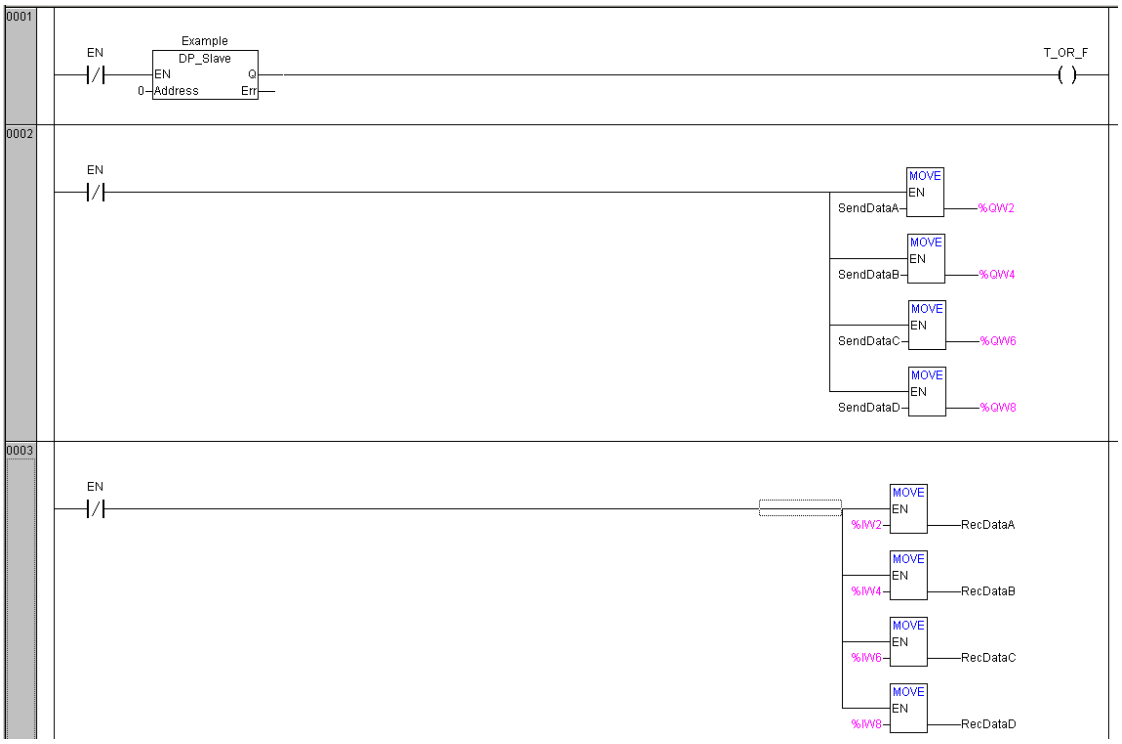


Figure 10-4-7 Ladder Diagram of LM3401

After electrify, program will copy the data of SendDataA, SendDataB, SendDataC, SendDataD to %QW2, %QW4, %QW6, %QW8 continually, and the data in %QW2, %QW4, %QW6, %QW8 will send to receive area of DP master automatically.

The data sent by DP master will be stored automatically in %IW2, %IW4, %IW6, %IW8, and the program will copy the data in %IW2, %IW4, %IW6, %IW8 continually to RecDataA, RecDataB, RecDataC, RecDataD .

In master configuration, the addresses are calculated according to address sequence of DP modules. If it's calculated by bit, the first bit address is 1 like %IX2.0 or %QX2.0, the second bit address is 2 like %IX2.1 or %QX2.1. If it is calculated in bytes, the first byte address is 1 like %IB2 or %QB2, and the second byte address is 2 like %IB3 or %QB3, and so on.

10.5 ETHERNET COMMUNICATION

10.5.1 Ethernet Communication Setting

LM Micro series PLC provides Ethernet communication processing. LM3403 is an Ethernet communication block and provides Modbus TCP slave communication function.

Configure the module parameters of LM3403 after the PLC configuration has been done. The main parameters include WriteDataLen_Byte and ReadDataLen_Byte. The module parameter settings of LM3403 are shown in figure 10-5-1. Configure IP_Address, Subnet_Mask, Gateway_Adress, WriteDataLen_Byte and ReadDataLen_Byte, MAC_Address and so on in figure 10-5-1. Refer to<<Hardware Manual>>for detailed technical specifications of LM3401.

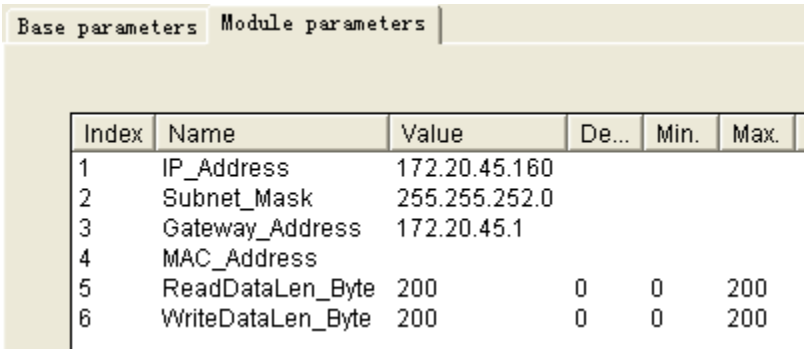


Figure 10-5-1 LM3403 Module Parameters

Add an instruction EtherNet_TCP when using Ethernet communication, similar with analog processing. The program of EtherNet_TCP is shown in figure 10-5-2 where the input value on Address 0 should be consistent with node id of EtherNet_TCP in PLC configuration list.

Scan Ethernet module when EN is set and cannot scan it when EN is reset.

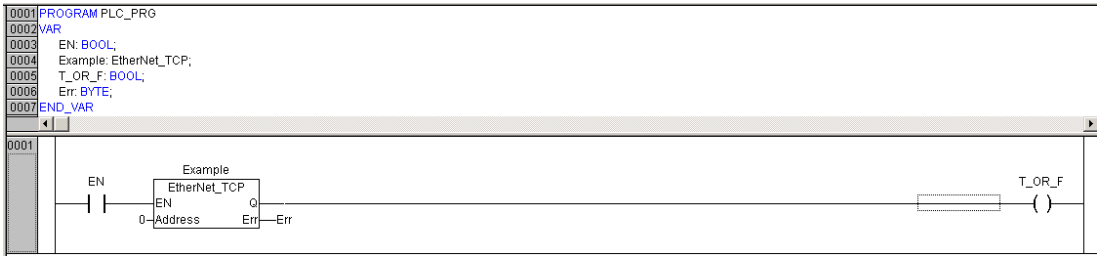


Figure 10-5-2 Ladder Diagram of EtherNet_TCP

System will assign input addresses and output addresses automatically after LM3403 has been configured, as shown in figure 10-5-3. The input addresses are 200 bytes starting from %IW2 and output addresses are 200 bytes starting from %QW2. The communication between the Modbus TCP Master and LM Micro series PLC is completed through these inputs and outputs. Inputs are used to store the data from DP master and the outputs are used to store the data which is going to be sent to Modbus TCP master from LM Micro series PLC.

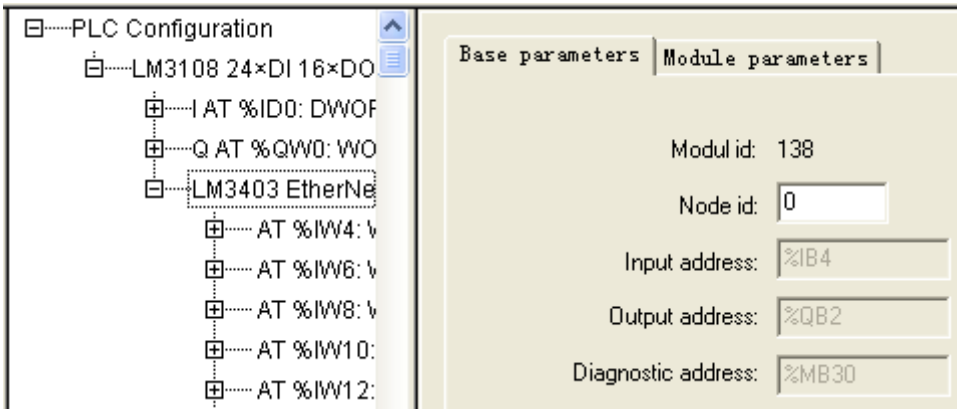


Figure 10-5-3 Configuration List of EtherNet_TCP

10.5.2 Example of Ethernet Communication

In this example PC (MODBUS/TCP master) sends data of 2 bytes to input area of PLC by Ethernet module, and receives data of 2 bytes from output area of PLC. At the same time PC sends 1bit to input area of PLC and receives 1bit from output area of PLC.

Variable declarations:

```
PROGRAM PLC_PRG
VAR
EN: BOOL;
        Example: EtherNet_TCP;
T_OR_F: BOOL;
SendDataA: WORD;   (*data A sent to MODBUS/TCP master by PLC *)
SendDataB: WORD;   (*data B sent to MODBUS/TCP master by PLC *)
SendBitC: BOOL;    (*data C sent to MODBUS/TCP master by PLC *)
RecDataA: WORD;   (*data A sent to PLC by MODBUS/TCP master *)
RecDataB: WORD;   (*data B sent to PLC by MODBUS/TCP master *)
RecBitC: BOOL;     (*data C sent to PLC by MODBUS/TCP master *)
END_VAR
```

Software configuration

Configure Ethernet module LM3403, as shown in figure 10-5-4.

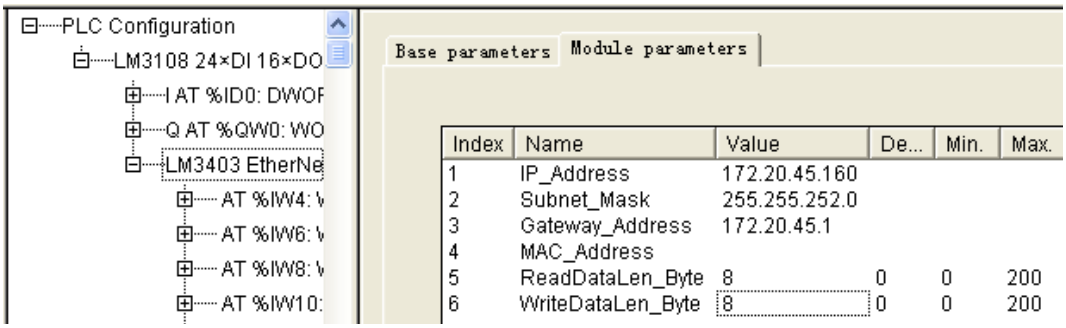


Figure 10-5-4 LM3403 Configuration

- IP_Address is the IP address of the Ethernet module (must in the same network segment with PC and no conflict with other IP addresses) .
- Subnet_Mask is the subnet mask, and is consistent with the subnet mask of the PC.
- GateWay_Addres is the address of the gateway.
- MAC_Address is null.
- ReadDataLen_Byte is the length of data sent to the PLC by the PC, and the number of received bytes is 8 (must be larger than actual length and the maximum is 200) .
- WriteDataLen_Byte is the length of data sent to the PC by the PLC, and the number of sent bytes is 8 (must be larger than actual length and the maximum is 200) .

The Address in an Ethernet module should be consistent with the node id in figure 10-5-5, the data A, B sent to PLC by PC are stored in %IW4, %IW6 and data C is stored in %IX8.0.

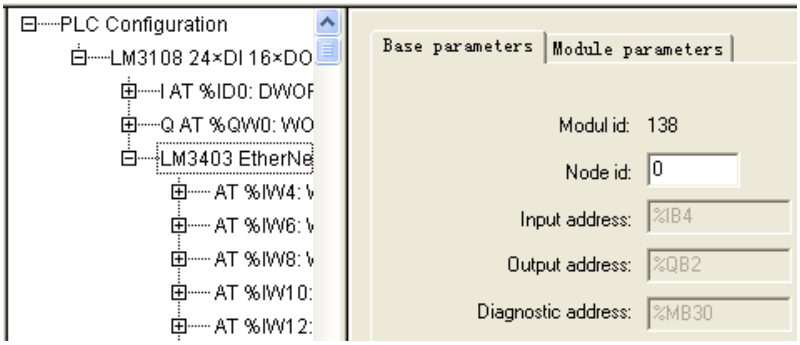


Figure 10-5-5

The data A, B sent to PC by PLC are stored in %QW2, %QW4 as shown in figure 10-5-6, and data C is stored in %QX6.0.

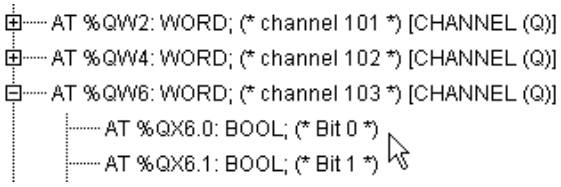


Figure 10-5-6 Address of Output Area

The program is shown in figure 10-5-7. In master configuration, the addresses are calculated according to address sequence of Ethernet modules. If it's calculated by bit, the first bit address is 1 like %IX2.1 or %QX2.1, and then the second bit address is 2 like %IX2.1 or %QX2.1. If it's calculated by byte, the first byte address is 1 like %IB2 or %QB2, and then the second byte address is 2 like %IB2 or %QB2, etc.

Figure 10-5-7 Program

Notes:
In Ethernet master configuration, I corresponds to bit address type 0x and word address type 4x and Q corresponds to bit address type 1x and word address type 3x.

Chapter 11

Visualization

Visualization is one of the components in PowerPro and is a graphical representation of the project variables and their changes to achieve the visualization of control processes. Hence visualization is a PLC HMI (Human Machine Interface, HMI) .

In PowerPro programming system there is an integrated visualization editor. In the process of developing applications of control systems, in PowerPro the users are allowed to develop visualization objects to watch and operate PLC data without other development tools.

11.1 CREATING NEW VISUALIZATION

Startup PowerPro, and build a new project1.pro. In object organizer click “Visualization”, as shown in figure 11-1-1.

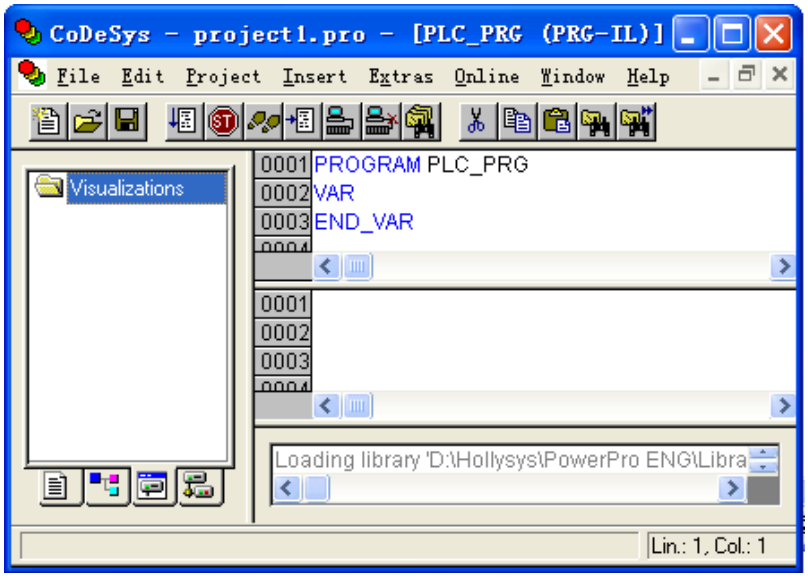


Figure 11-1-1 Visualization

In “Visualization” right click and select “Add Object”, as shown in figure 11-1-2.

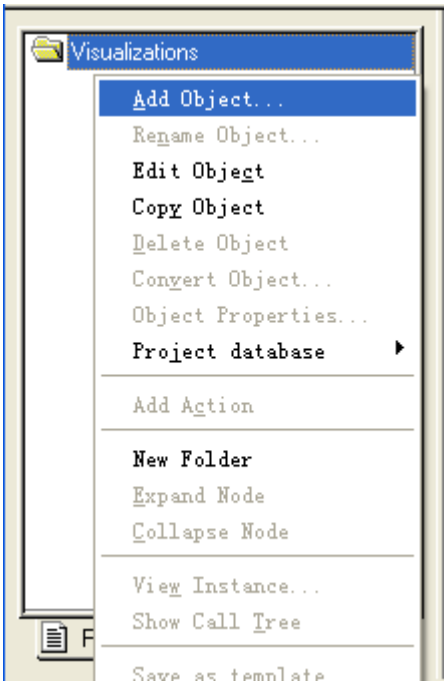


Figure 11-1-2 Add Visualization

The window titled “New Visualization” will appear, as shown in figure 11-1-3.

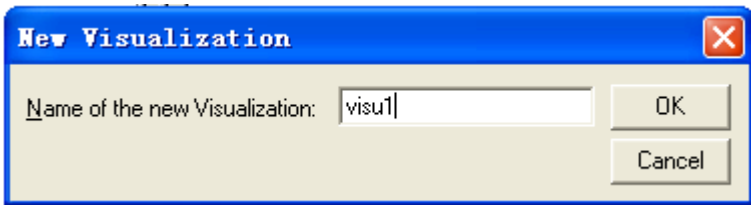


Figure 11-1-3 Name of the New Visualization

Enter the name in “Name of the new Visualization”, such as visu1, click “OK”, and a new visualization is built. On the right of the window there is an editor window, as shown in figure 11-1-4.

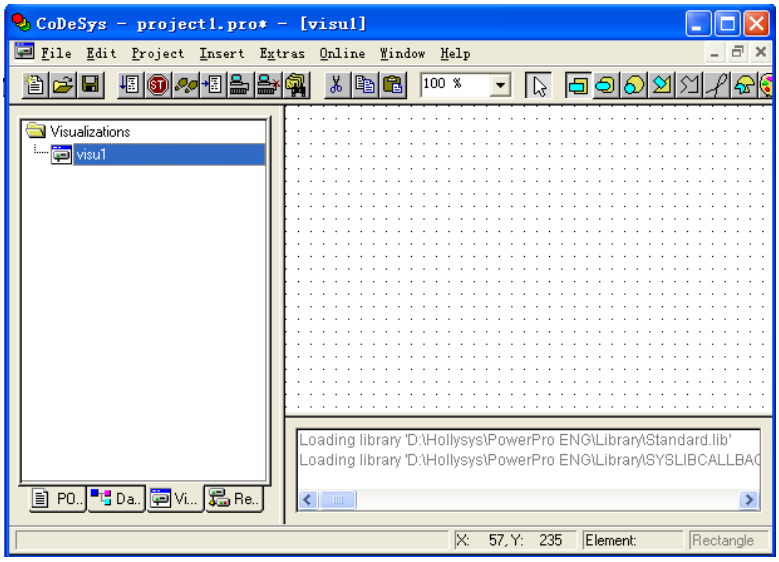


Figure 11-1-4 Build a visu1

11.2 VISUALIZATION ELEMENTS

The tool for editor is located on the top the main window, including pull-down menus and shortcut buttons.

Click the “Insert” menu in title bar, a pull-down menu of “Insert” appears, as shown in figure 11-2-1, and you can select what you need to add visualization which are the visualization elements.

Figure 11-2-1 “Insert” Menu

Click “Extras” menu in title bar, and a pull-down menu appears, as shown in figure 11-2-2, and you can make certain settings that affect the visualization. The menu can appear also by clicking the editor window.

Figure 11-2-2 “Extras” Menu

The shortcut buttons are located on the top of the main window, as shown in figure 11-2-3, completely the same with “Insert” menu. There are 15 different tools, and the functions are listed in table 11-2-1.

Figure 11-2-3 Shortcut Buttons

Name	Shortcut	Description
Rectangle		Insert rectangle in current visualization.
Rounded Rectangle		Insert rounded rectangle in current visualization.
Ellipse		Insert ellipse or circle in current visualization.
Polygon		Insert polygon in current visualization.
Polyline		Insert polyline in current visualization.
Curve		Insert Bezier curve in current visualization.
Pie		Insert pie in current visualization (wedge pattern of circle or ellipse).
Bitmap		Insert bitmap in current visualization.
Visualization		Insert a visualization that has been built in current visualization.
Button		Insert button in current visualization.
WMF file		Insert WMF in current visualization (Windows Metafile).
Table		Insert table in current visualization.
Trend		Insert trend in current visualization.
Alarm table		Insert alarm table in current visualization.
ActiveX element		Insert ActiveX element in current visualization.

Table 11-2-1 Edit Tool in Visualization

11.3 EDIT VISUALIZATION

11.3.1 Draw Visualization

If you hold down the mouse for a short time on the elements in tool bar, the name of the element is shown in a tooltip.

Create a rectangle, rounded rectangle, circle or ellipse

- Use the tool bar to select a visualization element by clicking the left mouse key on it and move to the editor window. Click on the desired starting point of your element and move the pointer with pressed left mouse key until the element has the desired dimensions.

Create a polygon or a line

- If you want to create a polygon or a line, first click with the mouse on the position of the first corner of the polygon or on the starting point of the line, and then click on the further desired corner points. By double clicking on the last corner point you will close the polygon and it will be completely drawn respectively the line will be completed.

Create a curve

- If you want to create a curve determine the initial, middle and end points with mouse clicks to define the circumscribing rectangle. An arc is drawn after the third mouse click. You can then change the position of the end point of the arc by moving the mouse and can then end the process with a double click or add another arc with additional mouse clicks.

Copy visualization elements

- You can copy one or more visualization elements with the command “Edit”/“Copy” or the <Ctrl>+<C> key combination. Another way is to select the elements and to again click in one of these elements with the key <Ctrl> held down. If you now hold the left mouse button down, you can separate the elements copied from the original.

Status bar in the visualization

- In a visualization the current X and Y position of the mouse cursor in pixels relative to the upper left corner of the image is displayed in the status bar. If the mouse pointer is located on an element, or if the element is being processed, then the number of the element will be displayed. If you have selected an element to insert, then the name of this element will also appear in the status bar.

11.3.2 Arrange Visualization

When you are drawing visualizations, it's necessary to modify and arrange them.

Select

- In order to select an element, click on the element.
- You can select the first element of the elements list by pressing the <Tab> key and jump to the next by each further keystroke. If you press the <Tab> key while pressing the <Shift> key, you jump backwards in the order of the elements list.
- In order to mark multiple elements, press and hold the <Shift> key and click the corresponding elements, one after another; or, while holding down the left mouse button, pull a window over the elements to be selected.

Select all

- With the command "Extras"/"Select all" you can select all visualization elements within the current visualization object.

Select mode

- You are in selection mode and you can select a graphic element if there is a "√" in front of "Extras"/"Select mode" menu item or the symbol  is pressed down. Or else you are in insert mode.

Changing the Selection and Insert Mode

- After the insertion of a visualization element, there is an automatic change back into the selection mode. If you want to insert an additional element the same way, you can once again select the corresponding command in the menu or the symbol  in the tool bar.
- You can also quickly change between the selection mode and the insert mode by pressing the <Ctrl>-key and the right mouse button simultaneously.
- In the insert mode, the corresponding symbol will also appear at the mouse pointer, and the name will also be indicated in black in the status bar.

Drag

- Select one or more visualization elements by clicking the left mouse key and drag them to desired position by pressing the left mouse key or direction keys.

Modify

- You can select an element which has already been inserted by a mouse click on the element or by pressing the <tab> key. A small black square will appear at each corner of each of the elements. You can change the size of the element by clicking on one of the black squares and, while keeping the left mouse button pressed, controlling the new outline.
- When an element is selected, its turning point is also displayed at the same time. The turning point is displayed as a small black circle with a white cross, and you can rotate the element around this point with a set angle. You can drag the turning point with a pressed left mouse button.

- When a polygon is selected, you can drag each individual corner using the same technique. While doing this, if you press the <Ctrl>-key then an additional corner point will be inserted at the corner point. By pressing the <Shift>+<Ctrl>-key, you can remove a corner point.
- Click the right mouse key in the visualization editor a menu appears, as shown in figure 11-3-1.

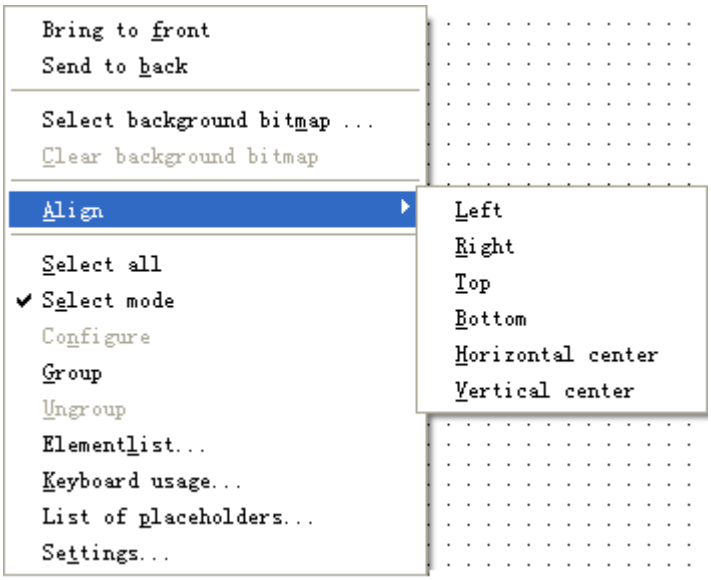


Figure 11-3-1 Context Menu of Visualization

The options are:

Bring to front

- Use this command “Extras”/“Bring to front” to bring selected visualization elements to the front.
- When the visualization elements are displayed in different element numbers, it indicates that elements are displayed in different layers. When several elements are located in the same position, the element with the largest number is in the front. Use the command to bring a certain visualization element to the front and the number is the largest.

Send to back

- Use this command “Extras”/“Send to back” to send selected visualization elements to the back.
- It’s the same with “Bring to front”, send selected visualization elements to the back and the number is 0.

Select background bitmap

- Use this command to open the dialog box for selecting files. Select a file with the extension “*.bmp” and click “Open”, the selected bitmap will then appear as the background in your visualization.

Clear background bitmap

- Use this command to remove the bitmap as the background for the current visualization.

Align

- Use the command “Extras”/“Align” to align selected visualization elements.

The following alignment options are available:

- “Left”: the left edge of each of the elements will be aligned with the element that is furthest to the left.
- “Right”: the right edge of each of the elements will be aligned with the element that is furthest to the right.
- “Top”: the top edge of each of the elements will be aligned with the element that is furthest to the top.
- “Bottom”: the bottom edge of each of the elements will be aligned with the element that is furthest to the bottom.
- “Horizontal center”: each of the elements will be aligned with the average horizontal center of all elements
- “Vertical center”: each of the elements will be aligned with the average vertical center of all elements.

Configure

- The options in “Category” vary according to different visualization elements. “Configure” is available only after one or more elements are selected, or else the menu item is in grey.

Group

- Elements can be grouped by selecting all desired elements and performing the command ‘Extras’ ‘Group’. The group will behave like a single element

Ungroup

- With the command “Extras”/“Ungroup” to resolve a group into individual elements.

In addition, the other four items including Elementlist, Keyboard usage, list of placeholders and settings are introduced below.

11.3.3 Elementlist

Click “Extras”/“Elementlist” to open a dialog box of element list, as shown in figure 11-3-2, including Number, Type and Position of elements. You can edit the elements using the buttons on the right. The elementlist will also be brought up by right clicking and selecting Elementlist.

- “OK”: Close the dialog box and confirm the changes.
- “To front”: Bring selected visualization elements to the front and the number is the largest.
- “To back”: Move selected visualization elements to the backand the number is the smallest.
- “One to front”: Bring selected visualization elements to an upper layer and the number increase by one.

- “One back”: Move selected visualization elements to a lower layer and the number decrease by one.
- “Delete”: Delete the selected visualization elements.
- “Undo”: Undo the last change.
- “Redo”: Restore the last change.
- “Edit”: Get the configuration dialog for the element.

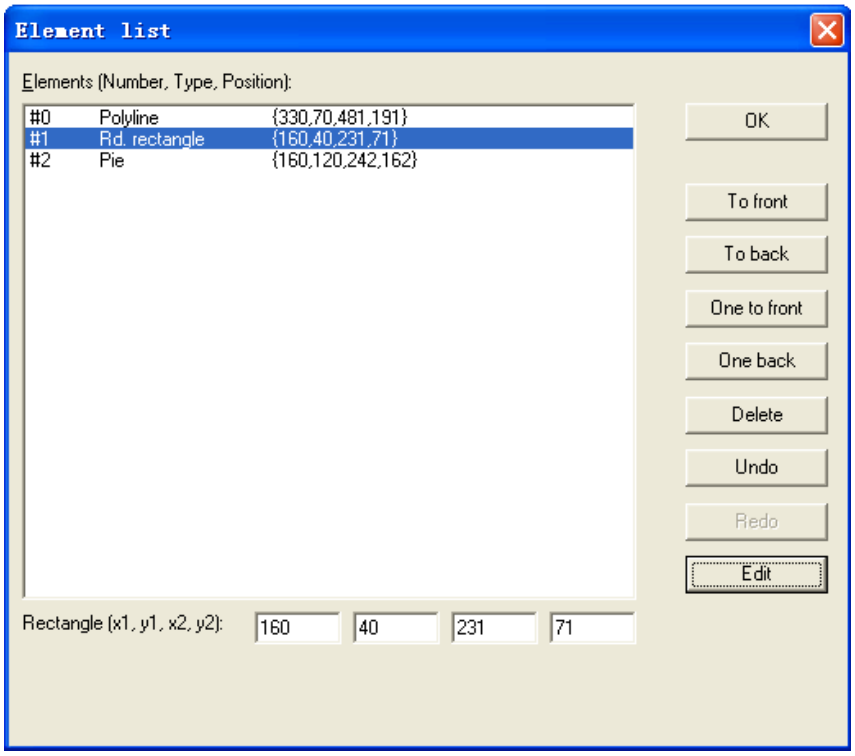


Figure 11-3-2 Elementlist

11.3.4 Keyboard usage

Select “Shift” and/or “Ctrl” and a selection list in column “Key” to assign an action.

“Expression” is used to express the action going to be achieved and the running result. The keyboard usage can be configured separately for each visualization object. Thus the same key (combination) can start different actions in different visualization.

Example

There are two different keyboard usages of VIS_1 in table 11-3-1 and VIS_2 in table 11-3-2.

Shift	Ctrl	Action	Key	Expression
x		Toggle	A	PLC_PRG.automatic
	x	Zoom	Z	VIS_2

Table 11-3-1 VIS_1 Keyboard Usage

Shift	Ctrl	Action	Key	Expression
x		Exec	E	INTERN LANGUAGE DEUTSCH
	x	Zoom	Z	VIS_1

Table 11-3-2 VIS_2 Keyboard Usage

If now you are in “Online Mode” and VIS_1 is the active window, then pressing “Shift+A” will cause that variable PLC_PRG.automatic will be toggled. “Ctrl+Z” will cause a jump from VIS_1 to VIS_2.

See table11-3-3 for the options of Action.

Action	Meaning
Toggle	Toggle variable
Tap true	Tap variable (set to TRUE)
Tap false	Tap variable (set to FALSE)
Zoom	Zoom to visualization
Exec	Execute program
Text	Text input of variable ‘Textdisplay’

Tale 11-3-3 Actions

In the columns Shift and Ctrl you can add the <Shift>- and/or the <Ctrl>-key to the already chosen key, so that a key combination will result.

11.3.5 List of Placeholders

The list of placeholder is shown in figure 11-3-3.

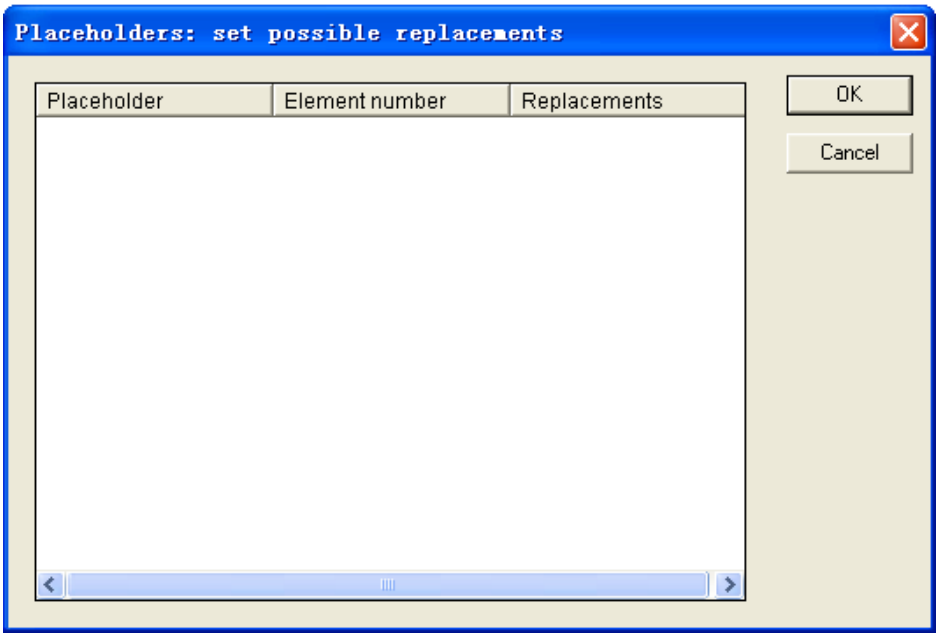


Figure 11-3-3 List of Placeholder

- Placeholder: List all placeholders.
- Element number: Show the elements which contain a placeholder.
- Replacements: Enter one or several strings (text, variable, expression).

11.3.6 Settings

The dialog box of visualization settings is shown in figure 11-3-4.

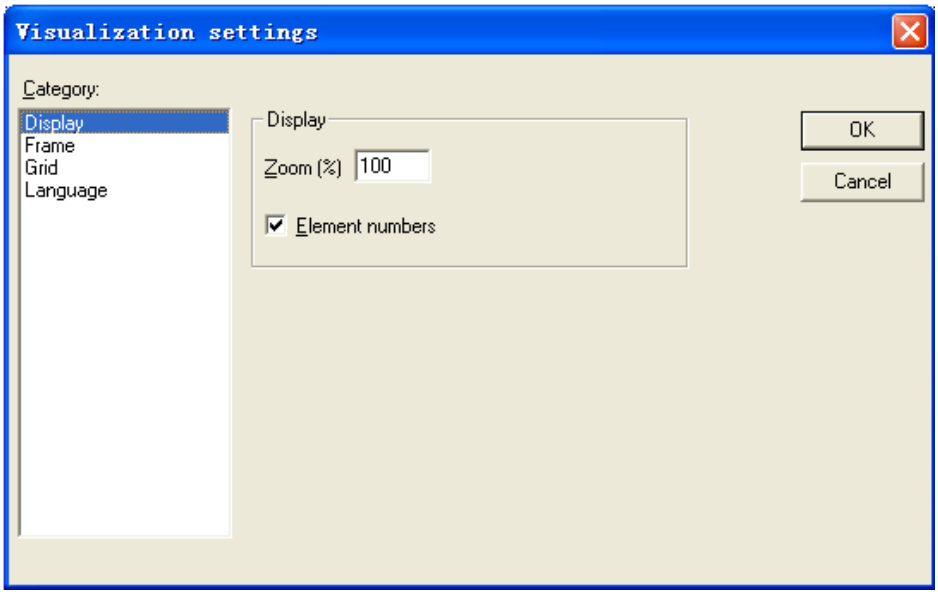


Figure 11-3-4 Visualization Settings

Display: Enter a zoom factor into the field Zoom (%) to increase or decrease the size of the visualization display. The Element numbers setting is optional, as shown in figure 11-3-4.

Frame: Frame setting, as shown in figure 11-3-5.

- If “Auto-Scrolling” is selected, the visible portion of the visualization window will move automatically when you reach the edge while drawing or moving a visualization element.
- If “Include background bitmap” is selected, the background bitmap will be fitted into the window as well, otherwise only the elements will be considered.
- If “Best fit in online mode” is selected, the entire visualization including all elements will be shown in the window in online mode regardless of the size of the window.

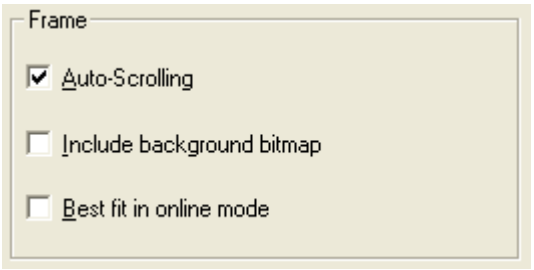


Figure 11-3-5 Frame Setting

Grid: Grid setting is shown in figure 11-3-6.

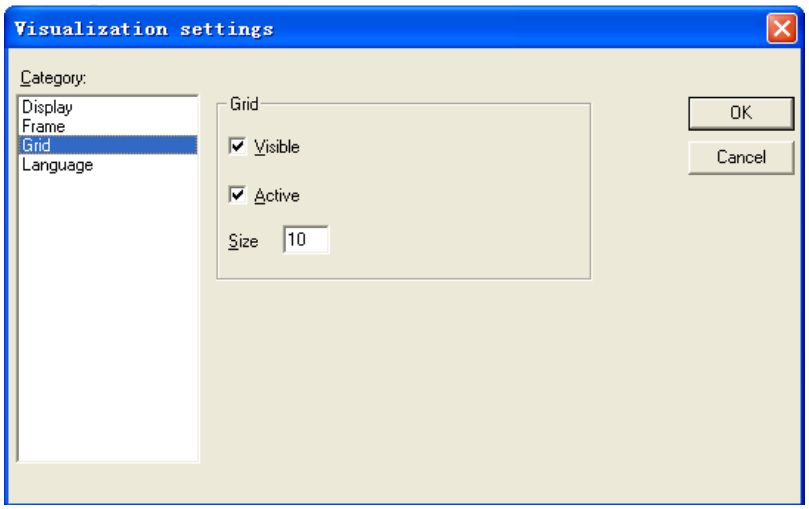


Figure 11-3-6 Grid Setting

- If “Visible” is selected, the grid points are visible. The spacing of the grid points is set in the field Size and is at the least 10.
- If “Active” is selected, the grid points only appear with a spacing which is a multiple of the entered size when they are moved, and otherwise can move at any spacing.
- Language: PLC does not support this function.

11.4 VISUALIZATION ELEMENT CONFIGURATION

11.4.1 Method for Visualization Element Configuration

In visualization editor select a visualization element and right click, or execute the command “Extras”, and choose “Configure”, as shown in figure 11-4-1. The “Configure” menu is available under the condition that a visualization element is selected, otherwise it will be displayed in grey.

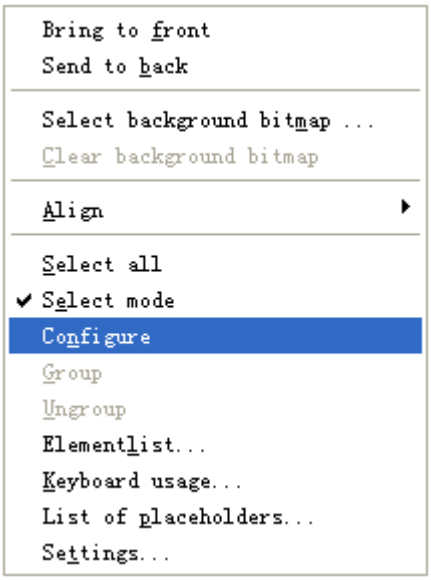


Figure 11-4-1 “Extras”/“Configure” Menu of Visualization

With the command “Configure”, a visualization element configuration dialog box appears, as shown in figure11-4-2 and you can configure the selected visualization element. The dialog can be opened also by double clicking the selected visualization element.

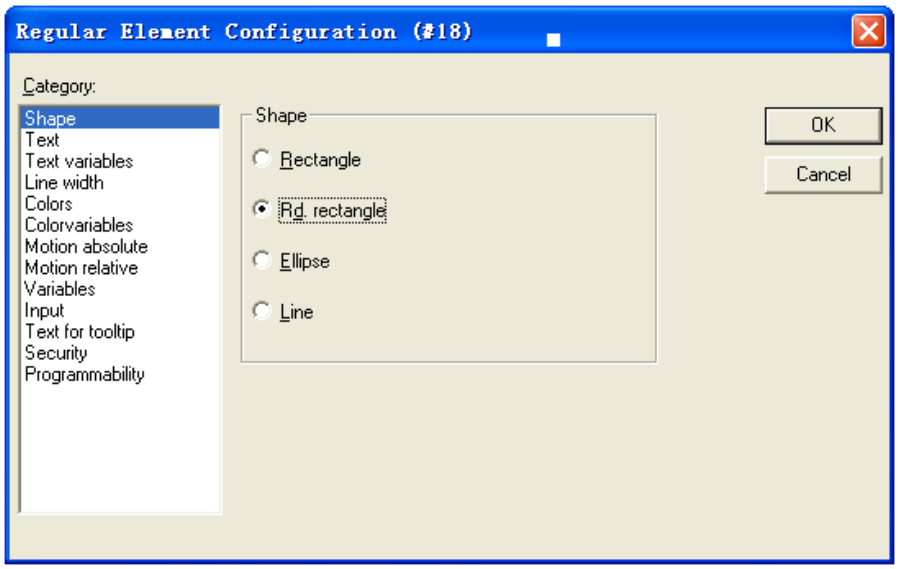


Figure 11-4-2 Visualization Element Configuration Dialog Box

11.4.2 Properties of Visualization Objects

For different visualization object, the category in the left area of the visualization element configuration dialog box and the corresponding parameters in the right area of the dialog box are different. One can set different variable parameters to define a visualization object and change its property. The relationships between visualization properties and visualization elements are shown in table 11-4-1, where the symbol "√" shows that there is a relationship.

Visualization Property		Visualization Elements														
Category	Name	Rectangle	Rounded Rectangle	Ellipse	Polygon	Polyline	Curve	Pie	Bitmap	Visualization	Button	WMF file	Table	Trend	Alarm	activeX element
Shape	Shape	√	√	√	√	√	√									
Text	Text	√	√	√	√	√	√	√	√	√	√	√				
Text variables	Text variables	√	√	√	√	√	√	√	√	√	√	√				
Line width	Line width	√	√	√	√	√	√	√	√	√		√				
Colors	Colors	√	√	√	√	√	√	√						√		
Color variables	Color variables	√	√	√	√	√	√	√	√	√		√				
Motion absolute	Motion absolute	√	√	√	√	√	√	√	√	√		√				
Motion relative	Motion relative	√	√	√					√	√		√				
Variables	Variables	√	√	√	√	√	√	√	√	√	√	√				
Input	Input	√	√	√	√	√	√	√	√	√	√	√				
Text for tooltip	Text for tooltip	√	√	√	√	√	√	√	√	√	√	√	√	√	√	
Security	Security	√	√	√	√	√	√	√	√	√	√	√	√	√	√	
Programmability	Programmability	√	√	√	√	√	√	√	√	√	√	√				
Angle	Angle							√								
Bitmap	Bitmap property								√		√					
Visualization	Visualization property									√						
Group	Group property											√				

Table	Table property												√			
Columns	Columns												√		√	
Rows	Rows												√			
Selection	Selection												√			
Trend	Trend													√		
Alarm table	Alarm table														√	
Settings for sorting	Settings for sorting														√	
Settings for alarm table	Settings for alarm table														√	
Control	Control type															√
Method calls	Method calls															√
Display	Display															√

Table1 1-4-1 the relationship between visualization property and visualization element

11.5 STATIC VISUALIZTION PROPERTIES

Static visualization properties are parameters that describe the basic shapes of visualization elements, including Shape, Text, Line width, and Colors.

11.5.1 Shape

The Shape property is used to define the shape of visualization elements. For regular elements you can select in the Shape category from among Rectangle, Rounded Rectangle, Line and Ellipse. For un-regular elements you can select in the Shape category from among Polygon, Polyline and Curve, as shown in figure 11-5-1. The changes of Shape attribute are valid only in the determined area.

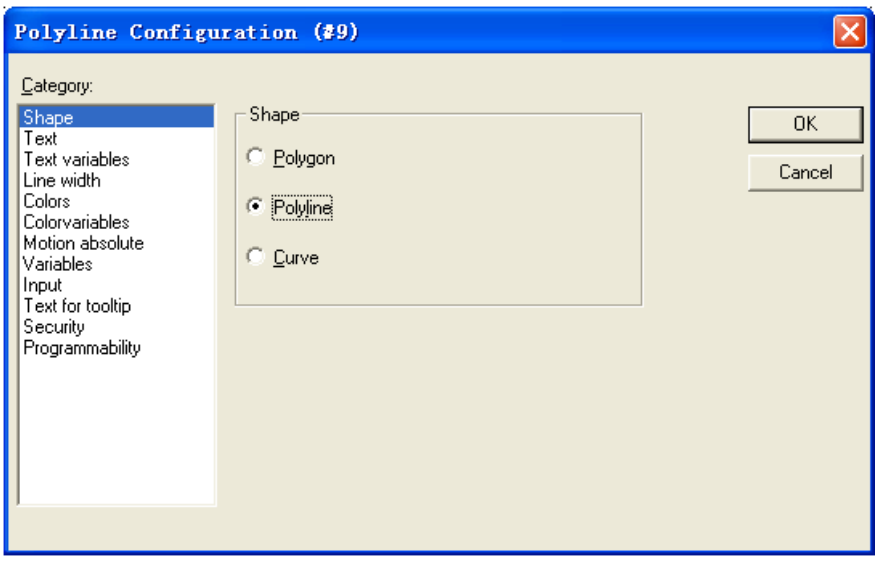


Figure 11-5-1 Dialog Box for Configuring Visualization Elements

11.5.2 Text

One can add text in visualization object using Text property, as shown in figure 11-5-2.

Content

- Enter the text in the Content field. With the key combination <Ctrl>+<Enter> you can insert line breaks.

Horizontal

- In the option “Horizontal” set text placement: Left, Center or Right.

Vertical

- In the option “Vertical” set text placement: Top, Center, or Bottom.
- Click “Font” or “Standard-Font” to set the font.

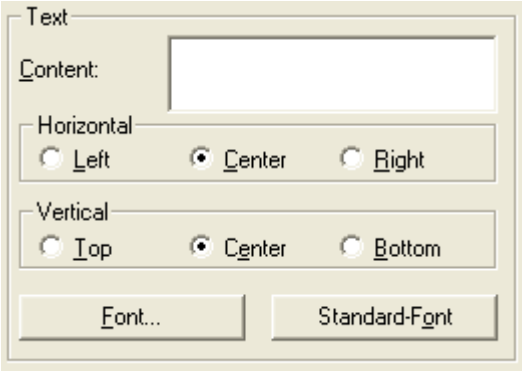


Figure 11-5-2 Text Setting

In standard C library the commonly used placeholders include %s (string format) , %f (floating-point format) , %d (decimal number) and %x (hexadecimal) and the meanings are shown in table 11-5-1.

Placeholder	Meaning
% a	Abbreviated weekday name
% A	Full weekday name
% b	Abbreviated month name
% B	Full month name
% c	Date and time representation appropriate for locale
% d	Day of month as decimal number (01-31)
% H	Hour in 24-hour format (00 – 23)
% I	Hour in 12-hour format (01-12)
% j	Day of year as decimal number (001 – 366)
%m	Month as decimal number (01-12)
%M	Minute as decimal number (00-59)
%p	Current locale’s A.M./P.M. indicator for 12-hour clock
%S	Second as decimal number (00-59)
%U	Week of year as decimal number, with Sunday as first day of week (00-53)

%w	Weekday as decimal number (0 – 6; Sunday is 0)
%W	Week of year as decimal number, with Monday as first day of week (00-53)
%x	Date representation for current locale
%X	Time representation for current locale
%y	Year without century, as decimal number (00-99)
%Y	Year with century, as decimal number
%z	Time-zone name, for: standard time
%Z	
%%	Percent sign

Table 11-5-1 Placeholder and Meaning

Example

- If you fill the text with %2.5f mm, then 32.8889 mm will be displayed in online mode.

Example

- Set the visualization in the format as shown in figure 11-5-3. The result in online mode is displayed in figure 11-5-4. Display current time format: time-zone name day-month-year hour：minute：second.

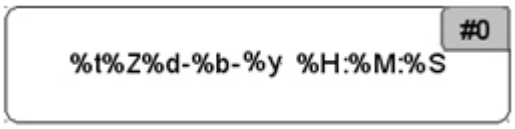


Figure 11-5-3 Time Placeholder Application

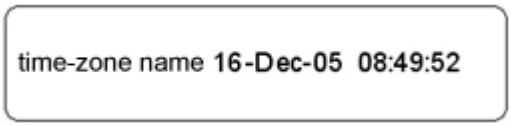


Figure 11-5-4 Display in Online Mode

11.5.3 Line width

The Line width property is used to define the line width of visualization elements. You can choose the line width from 1 to 5 pixels in the dialog box for line width configuration, as shown in figure 11-5-5. Additionally, another value can be entered manually in the field “Other”. A project variable can be defined dynamically in the field “Variable for line width” with the help of Input Assistant < F2>. The static setting will be overwritten by dynamic setting in online mode.

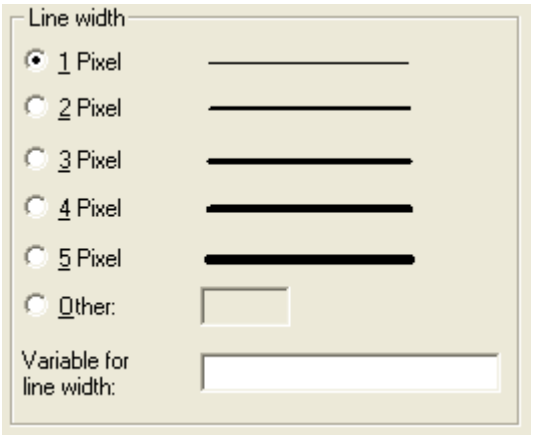


Figure 11-5-5Line Width Configuration

11.5.4 Colors

In the visualization element configuration dialog box, in the color category you can select primary colors and alarm colors for the inside area and for the frame of your element, as shown in figure 11-5-6. Choosing the options “No color inside” and “No frame color” you can create transparent elements. You can select alarm colors for the inside area and for the frame of your elements. As soon as the parameter is defined dynamically by a variable, the static setting will be overwritten in online mode.

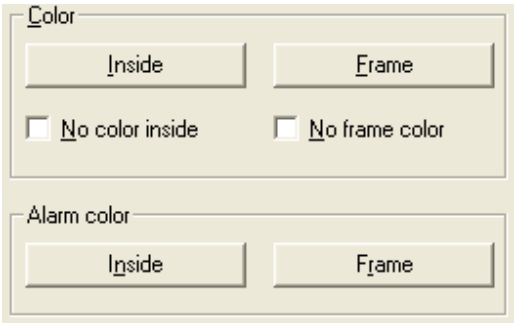


Figure 11-5-6 Colors Configuration

11.5.5 Text for tooltip

The dialog “Text for tooltip”, as shown in figure 11-5-7, offers an input field “Content” for text which will appear in a text field as soon as the mouse cursor is hovering over the object in offline or online mode. You can insert a line break by using the key combination <Ctrl>+<Enter>.

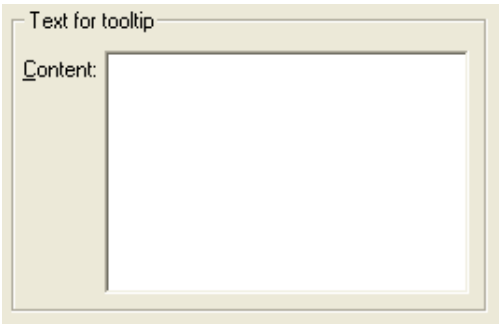


Figure 11-5-7 Text for tooltip

11.5.6 Security

With the “Security” you can assign different access rights (“No Access”, “Read Access” and “Full Access”) concerning particular visualization elements for the eight user groups, as shown in figure 11-5-8. In online mode user group with different access rights get different operating possibilities:

- “No Access”: Element will not be visible.
- “Read Access”: Element will be visible but not operatable (no inputs allowed)
- “Full Access”: Element is visible and operatable.

If you want to assign the access rights also to all other elements of the visualization object, activate option “Apply to all visual elements”.

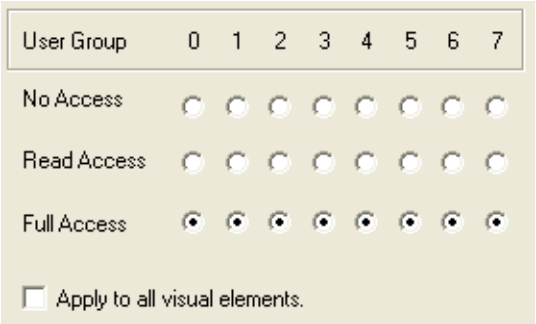


Figure 11-5-8 Access Setting

11.5.7 Bitmap Configuration

You can insert a Bitmap in visualization. You can enter the options for a bitmap in the Bitmap category within the visualization element configuration dialog box, as shown in figure 11-5-9.

Bitmap

- You can use the ... button after “Bitmap” to open the standard Windows Browse dialog box from which you can select the desired bitmap.
- You can create a new transparent visualization object by activating “Background transparent”.

Frame

In the “Frame” option you can set the frame property of the bitmap:

- “Anisotropic”: The bitmap remains the same size as the frame which allows you to change the height and width of the bitmap independently.
- “Isotropic”: The bitmap retains the same proportions even if the overall size is changed.
- “Fixed”: The original size of the bitmap will be maintained regardless of the size of the frame.

If the Draw option is selected together with the Fixed option, the frame of the visualization object will be displayed, otherwise it will not be displayed. If the Clip option is also selected, only the portion of the bitmap that is contained within the frame will be displayed. If you want to display the whole bitmap, drag the frame of the bitmap to a desired size.

The options “Color” and “Alarm color” can be used to configure frame color and alarm color.

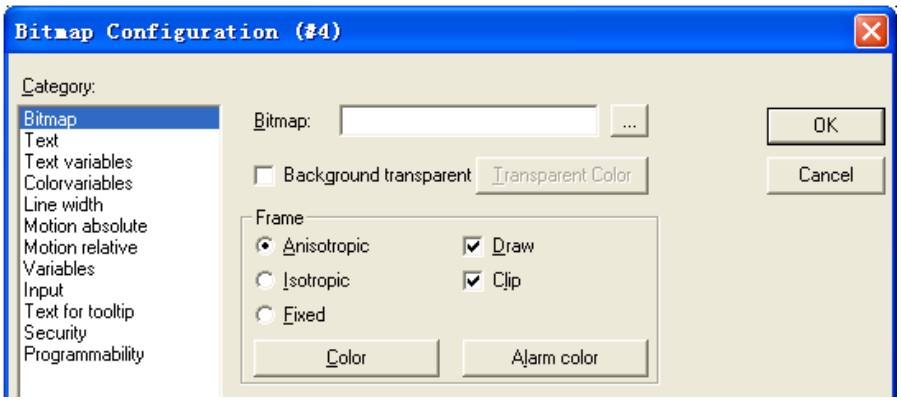


Figure 11-5-9 Bitmap Configuration

11.5.8 Visualization Configuration

You can insert an existing visualization as an element in the present visualization. You can enter the options for a visualization object within the visualization dialog box, as shown in figure 11-5-10.

Visualization

- Use the ... button after Visualization to open a dialog box containing the visualizations available in this project.
- The "Placeholder" button leads to the "Replace placeholder" dialog.

Frame

- In the "Frame" option you can set the frame property of the visualization:
- "Anisotropic": The bitmap remains the same size as the frame which allows you to change the height and width of the bitmap independently.
- "Isotropic": The bitmap retains the same proportions even if the overall size is changed.
- "Fixed": The original size of the bitmap will be maintained regardless of the size of the frame.
- If the Draw option is selected together with the Fixed option, the frame of the visualization object will be displayed, otherwise it will not be displayed. If the Clip option is also selected, only the portion of the bitmap that is contained within the frame will be displayed. If you want to display the whole bitmap, drag the frame of the bitmap to a desired size.
- The options "Color" and "Alarm color" can be used to configure frame color and alarm color.

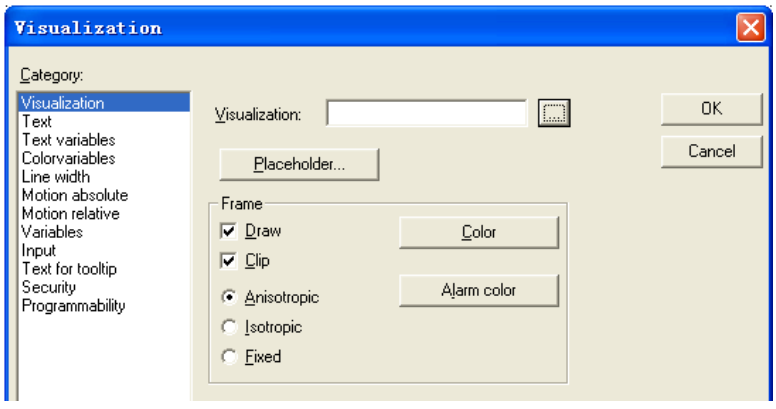


Figure 11-5-10 Visualization Configuration

11.5.9 Group Configuration

One can insert a Windows Metafile (WMF) into a visualization object. Group configuration dialog is used to set the parameters of WMF. Group configuration dialog of WMF is shown in figure 11-5-11.

In the “Frame” option you can set group frame property:

- “Draw”: The frame of the visualization object will be displayed.
- “Isotropic”: The bitmap retains the same proportions even if the overall size is changed.
- “Clip”: When the frame is smaller than the bitmap, only that portion of the bitmap that is contained within the frame will be displayed.

The options “Color” and “Alarm color” can be used to configure frame color and alarm color.

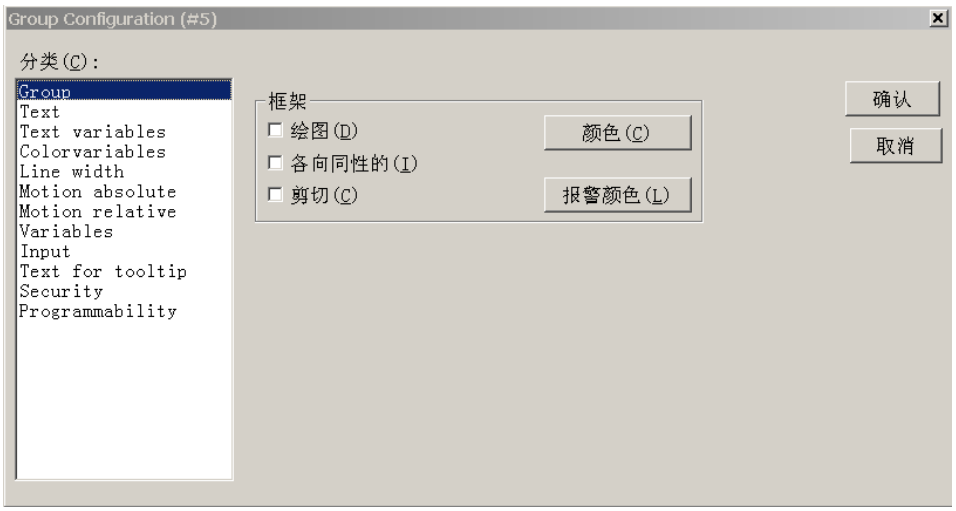


Figure 11-5-11 Group Configuration of WMF

11.5.10 Angle

The category “Angle” is used to define the two angles of the sector element in degrees. Double click the pie and the dialog of configuring a pie will appear, as shown in figure 11-5-12. Click the Angle and enter the start angle and the end angle of the sector element in degrees in the fields “Start angle” and “End angle”, the sector will be drawn clockwise from the start angle position to the end angle position. If “Show only segment” is selected, the segment will be displayed without angle.

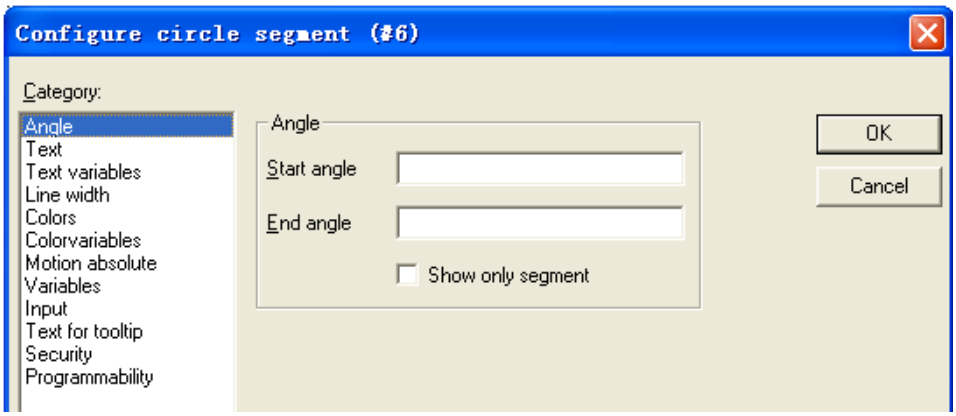


Figure 11-5-12 Dialog for Configuring a Pie

Example

Variable declaration:

```
PROGRAM PLC_PRG
VAR
    angle_start: REAL := 0;
    angle_end: REAL := 90;
END_VAR
```

Display in online mode is shown in figure 11-5-13.

Figure 11-5-13 Pie Example

11.6 VISUALIZATION PROGRAMMABILITY

11.6.1 Programmability

The properties of a visualization element not only can be defined using a static setting, but also can be defined using the components of a structure variable, which is exclusively used for programming visualization elements. For this purpose the structure VisualObjectType is available in the library SysLibVisu.lib. Its components can be used to define most of the element properties. In case of multiple definition of an element property the value of the "normal" project variables will overwrite that of the structure variable and both will overwrite a static definition.

In order to configure the element properties by using a structure variable, open the configuration dialog, and select category 'Programmability', as shown in figure 11-6-1. Active "Object name" and enter a variable name. The variable will be declared with type VisualObjectType automatically, which is a structure contained in the library SysLibVisu.Lib. The variable declared here is a global variable and the declaration is done implicitly and not visible by the user. Make sure that the library is included in the library manager.

After the next compilation, the newly assigned structure variable will be available in the project.

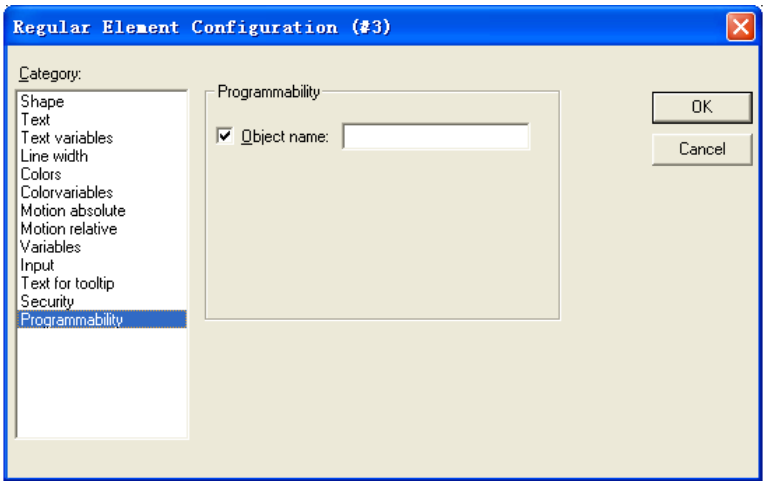


Figure 11-6-1 Programmability

11.6.2 Visual Library

The definition of VisualObjectType in SysLibVisu.lib is shown below.

```
TYPE VisualObjectType
STRUCT
    (* Absolute movement *)
    nXOffset:INT;
    nYOffset:INT;
    nScale:INT;
    nAngle:INT;
    (* Variables *)
    bInvisible:BOOL;
    stTextDisplay:STRING;
    bToggleColor:BOOL;
    bInputDisabled: BOOL;
    stTooltipDisplay:STRING;
    (* Text and font *)
    dwTextFlags:DWORD;
    dwTextColor:DWORD;
    nFontHeight:INT;
    dwFontFlags:DWORD;
    stFontName:STRING;
    (* Line *)
    nLineWidth:INT;
    (* Color *)
    dwFillColor: DWORD;
    dwFillColorAlarm: DWORD;
    dwFrameColor: DWORD;
    dwFrameColorAlarm: DWORD;
    dwFillFlags: DWORD;
    dwFrameFlags: DWORD;
    (* Relative movement *)
    nLeft:INT;
    nTop:INT;
    nRight:INT;
    nBottom:INT;
END_STRUCT
END_TYPE
```

Data type and effect of component of STRUCT VisualObjectType are shown in table 11-6-1. At the beginning of the component name the data type is integrated:

n: INT, dw: DWORD, ,b: BOOL, st: STRING.

Component and Data Type	Effect	Example (the Object Name “vis1” has been defined for the element)	corresponding entries in configuration dialog
nXOffset : INT;	Shift element in X-direction	vis1.nXOffset:=val1; (element is set to position X=val1)	Motionabsolute: X-Offset
nYOffset : INT;	Shift element in Y-direction	vis1.nYOffset:=val2; (element is set to position Y=val2)	Motionabsolute: Y-Offset
nScale : INT;	Change of the size	vis1.nScale:=plc_prg.scale_var; (element size changes linear with change of value of plc_prg.scale_var)	Motion absolute: Scale
nAngle : INT;	Rotating element around its center	vis1.anglevar:=15; (element rotates clockwise by 15)	Motion absolute: angle

bInvisible : BOOL;	Element is visible / invisible	vis1.visible:=TRUE; (element is invisible)	Color: No color inside+ No frame color Colorvariables: Fillcolor+Framecolor
stTextDisplay: STRING;	Text is displayed in element	vis1.TextDisplay:='ON / OFF'; (element is inscribed with this text)	Text: Content
bToggleColor : BOOL;	color change when toggling between TRUE and FALSE	vis1.bToggleColor:=alarm_var; (As soon as alarm_var gets TRUE, the element gets the color defined via the components dwFillColorAlarm and dwFrameColorAlarm)	Input: Toggle variable Variables: Change color
bInputDisabled : BOOL;	if FALSE: Inputs in category 'Input' are ignored	vis1.bInputDisabled:=FALSE; (no input is possible for this element)	Variables: Input Disable
stTooltipDisplay : STRING;	Text of the tooltip	vis1.stTooltipDisplay:='Switch for '; (show text of the tooltip 'Switch for')	Text for Tooltip: Content:
dwTextFlags : DWORD;	Text position: 1 left justified 2 right justified: 4 centered horizontally 8 top 10 bottom 20 centered vertically (addition of values)	vis1.dwTextFlags:=24; (Text will be placed in the center of the element (4+20))	Text: Horizontal and Vertical options Textvariables: Textflags
dwTextColor : DWORD;	Text color (definition of colors see subsequent to this table)	vis1.dwTextColor :=16#00FF0000; (Text is blue-colored)	Textvariables: Textcolor
nFontHeight : INT;	Font height in Pixel. should be in range 10-96	vis1.nFontHeight:=16; (Text height is 16 pt)	Textvariables: Font heigth
dwFontFlags : DWORD;	Font display. Available flags: 1 italic 2 fett 4 underlined 8 canceled (combinations by adding values)	vis1.dwFontFlags:=10; (Text is displayed blue and canceled, 2+8)	Textvariables: Fontflags
stFontName : STRING;	Change font	vis1.stFontName:='Arial'; (Arial is used)	Textvariables: Fontname
nLineWidth : INT;	Line width of the frame (pixels)	vis1.nLWidth:=3; (Frame width is 3 Pixels)	Line width
dwFillColor : DWORD;	Fill color (definition of colors see subsequent to this table)	vis1.dwFillColor":=16#00FF0000; (fill blue color)	Color: Color Inside Colorvariables: Fill color
dwFillColorAlarm : DWORD;	Fill color as soon as bToggleColor gets TRUE (definition of colors see subsequent to this	vis1.dwFillColorAlarm:=16#00808080; (as soon as Variable bToggleColor gets TRUE, the frame will be displayed grey-colored)	Color: Alarm color Inside Colorvariables: Fill color alarm

	table)		
dwFrameColor : DWORD;	Frame color (definition of colors see subsequent to this table)	vis1.dwFrameColor:=16#00FF0000; (Frameis blue-colored)	Color: Color Frame Colorvariables: Frame color
dwFrameColorAlarm : DWORD;	Fill color as soon as bFrameColor gets TRUE (definition of colors see subsequent to this table)	vis1.dwFrameColorAlarm:=16#00808080; (as soon as Variable bFrameColor gets TRUE, the frame will be displayed grey-colored)	Color: Alarm color Frame Colorvariables: Frame color alarm
dwFillFlags: DWORD;	Color, as defined by the color variables, can be displayed or ignored 0 = show color >0 = ignore setting	vis1.dwFillFlags:=1; (element gets invisible)	Color: No color inside + No frame color Colorvariables: Fillflags
dwFrameFlags : DWORD;	Display of frame: 0 full 1 dashed (---) 2 dotted ( ) 3 dash-point (_._._) 4 dash-point-point (_.._..) 8 blind out line	vis1.FrameFlags:=1; (Frame will be displayed as dashed line)	Colorvariables: Frameflags

Table 11-6-1 Data Type and Effect of Component

Defining color values:

```
vis1.dwFillColor := 16#00FF00FF;
```

A color is entered as a hex number which is composed of the Blue/Green/Red (RGB) components. The first two zeros after "16#" should be set to in each case, to fill the DWORD size. For each color value 256 (0-255) colors are available.

16#	00	FF	00	FF
keyword	null	blue	green	red

Example (defining color values)

```
PROGRAM PLC_PRG
VAR
    n:INT:=0;
    bMod:BOOL:=TRUE;
END_VAR
(* Blinking element *)
n:=n+1;
bMod:= (n MOD 20) > 10;
IF bMod THEN
    blinker.nFillColor := 16#00808080; (* grey *)
ELSE
    blinker.nFillColor := 16#00FF0000; (* blue *)
END_IF
```

11.7 DYNAMIC VISUALIZTION PROPERTIES

Dynamic visualization properties are used to define parameters dynamically by modifying project variables.

11.7.1 Text variables

Text variables are used to define dynamic text. Enter the values for the variables in the dialog of configuring text variables, as shown in figure 11-7-1. Enter the variable names with the aid of the input assistant (<F2>).

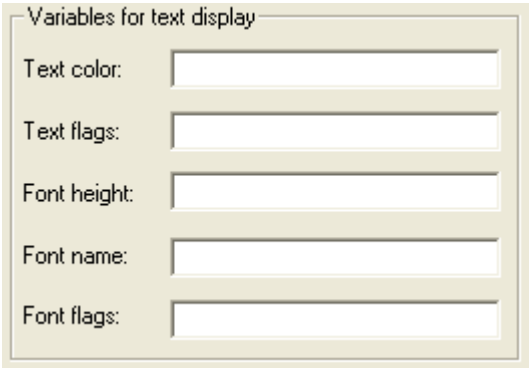


Figure 11-7-1Variables for text display

11.7.2 Color variables

"Color variables" are used to define dynamic attributes of colors. The dialog box titled "variables for colors" is shown in figure 11-7-2, which can be used to enter the values of relevant variable in text box to achieve animation effects. Enter the variable names with the aid of the input assistant (<F2>).

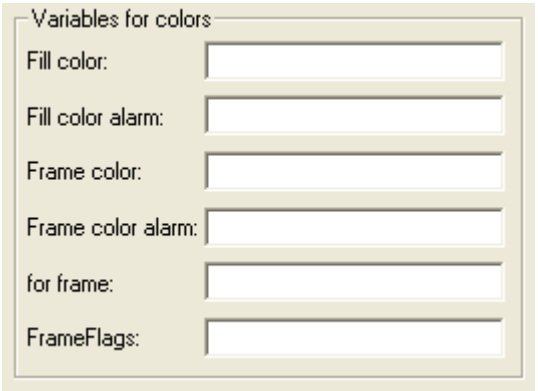


Figure 11-7-2 Setting of Color variables

11.7.3 Motion absolute

Motion absolute is used to define the dynamic attributes of motion absolute. The dialog box titled "motion absolute setting" is shown in figure 11-7-3, which can be used to enter the values of relevant variables in the text boxes to achieve animation effects. Enter the variable names with the aid of the input assistant (<F2>).

X-Offset: The variable which can shift the element in the X direction, depending on the respective variable value.

Y-Offset: The variable which can shift the element in the Y direction, depending on the respective variable value.

Scale: A variable in the Scale field will change the size of the element linear to its current value. This value, which is used as scaling factor, will be divided by 1000 implicitly. So the value is 1000 if you want to get the new visualization object one time of the original one and the value is 2000 if you want to get the new visualization object two times of the original one.

Angle: A variable in the Angle field causes the element to turn on its turning point, depending on the value of the variable (positive value = clockwise, negative value = counter-clockwise). For rectangle, it shifts a certain angle taking the basic point as a origin rather than rotate. With all other elements such as polygon and ellipse, the element rotates, in such a way, that the upper edge always remains on top. That means every point of such elements rotate.

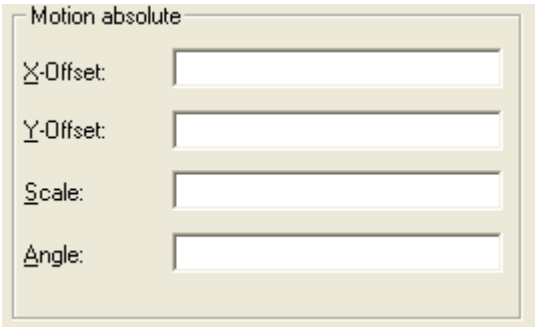


Figure 11-7-3 Configuration of Motion absolute

11.7.4 Motion relative

Motion relative is used to define the dynamic attributes of motion relative. The dialog box titled “motion relative setting” is shown in figure 11-7-4, which can be used to enter the values of relevant variables in the text boxes to achieve animation effects. The easiest way to enter variables into the fields is to use the Input Assistant (<F2>). For coordinate axis, the right of X direction and the top of Y direction are positive, that is, the edge moves towards the positive direction if the value is positive, otherwise towards the negative direction.

- Left edge: The variable which can shift the left edge, depending on the respective variable value.
- Top edge: The variable which can shift the top edge, depending on the respective variable value.
- Right edge: The variable which can shift the right edge, depending on the respective variable value.
- Bottom edge: The variable which can shift the bottom edge, depending on the respective variable value.

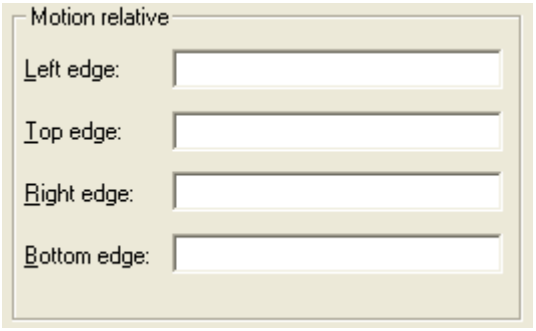


Figure 11-7-4 Setting of Motion Relative

11.7.5 Variables

"Variables" is used to define the other dynamic attributes of graphic object. The dialog box titled "Variables" is shown in figure 11-7-5, which can be used to enter the values of relevant variables in the text boxes to achieve animation effects. Enter the variable names with the aid of the input assistant (<F2>).

- "Invisible": You can enter Boolean variables in the Invisible field. The values in the fields determines the actions. If the variable of the Invisible field contains the value FALSE, the visualization element will be visible. If the variable contains the value TRUE, the element will be invisible.
- "Change color": You can enter Boolean variables in the Change color field to determine the element color. If the variable which is defined in this field, has the value FALSE, the visualization element will be displayed in its default color. If the variable is TRUE, the element will be displayed in its alarm color.
- "Text display": Enter the value of the textdisplay variable. If you have inserted a "%s" in the "Content" field of the Text category, then the value of the variable which is defined in "Textdisplay" will be displayed in online mode instead of "%s".

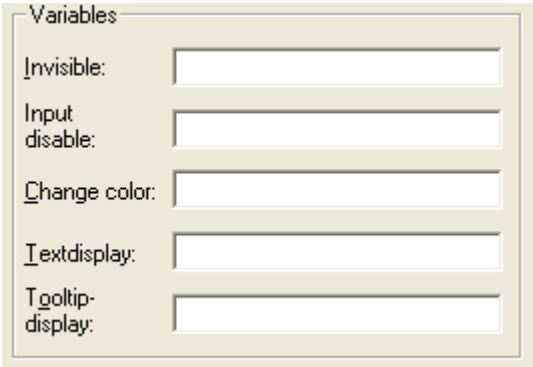


Figure 11-7-5Variable Setting

11.7.6 Input

Input is used to define input dynamic attributes of graphic object. The dialog box titled "input" is shown in figure 11-7-6, which can be used to enter the values of relevant variables in the text boxes to achieve animation effects. Enter the variable names with the aid of the input assistant (<F2>).

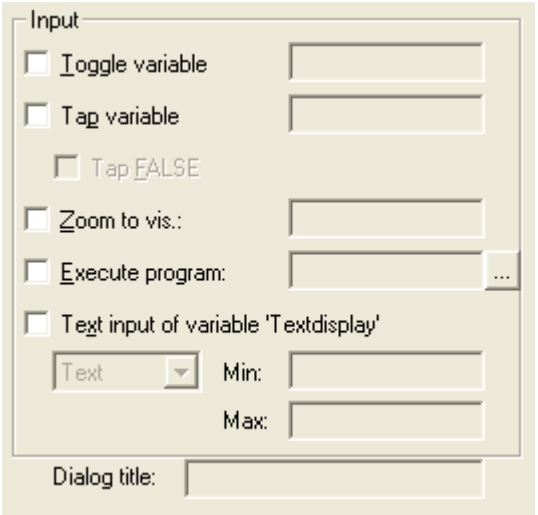


Figure 11-7-6 Input Setting

- Toggle variable: If this option is activated, in online mode you will toggle the value of the variables which are located in the input field by each mouse click on the visualization element. The value of the Boolean variable changes with each mouse click from TRUE to FALSE and then back to TRUE again at the next mouse click, etc.
- Tap variable: If this option is activated, in online mode you can switch the value of the Boolean variable which is located in the input field, between TRUE and FALSE. Place the mouse cursor on the element, press the mouse-key and hold it depressed. If option Tap FALSE is activated, the value is set to FALSE as soon as the mouse key is pressed, otherwise it is set to TRUE at this moment. The variable changes back to its initial value as soon as you release the mouse key.
- Zoom to vis.: If this option is activated, you can enter in the edit field the name of a visualization object of the same project to which you want to jump by a mouse-click on the element in online mode. If a program variable of the type STRING (e.g. PLC_PRG.xxx) has been entered instead of a visualization object, then this variable can be used to define the name of the visualization object (e.g. ‚visu1') which the system should change to when a mouse click occurs (e.g. xxx:= ‚visu1).
- Execute program: If this option is activated you can enter any program in the input field, which will be executed in online mode as soon as you perform a mouse-click on the element.
- Text input of variable ‘Text display’: If you select the Text input of variable 'Textdisplay', then in Online mode you will get the possibility to enter an value in this visualization element which will upon pressing <Enter> be written to the variable that appears in the Textdisplay field of the Variables category.

11.8 TABLES

You can insert a table in a visualization object, as shown in figure 11-8-1.

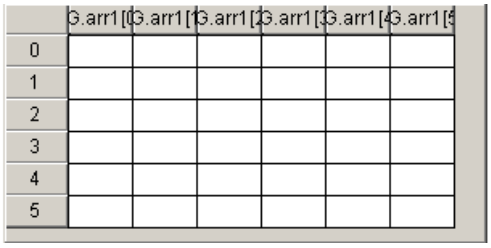


Figure 11-8-1 Insert Table

Double click the table and a dialog “Configure Table” will appear, as shown in figure 11-8-2, where you can set the options ‘Table’, ‘Columns’, ‘Rows’, ‘Selection’, and ‘Text for tooltip and Security’. Enter the variable names with the aid of the input assistant (<F2>).

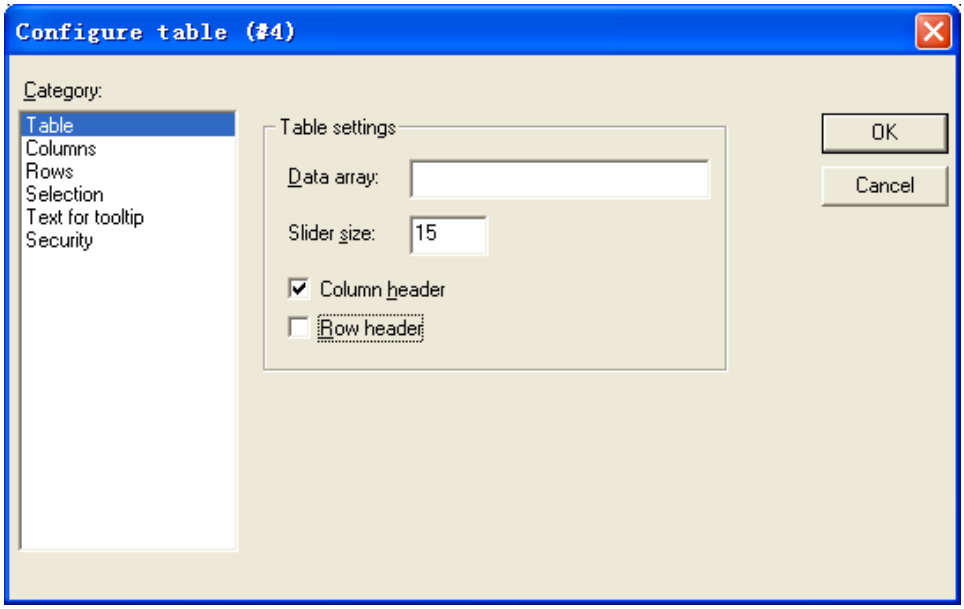


Figure 11-8-2 Configure table

11.9 TREND

You can insert a trend in a visualization object, as shown in figure 11-9-1.

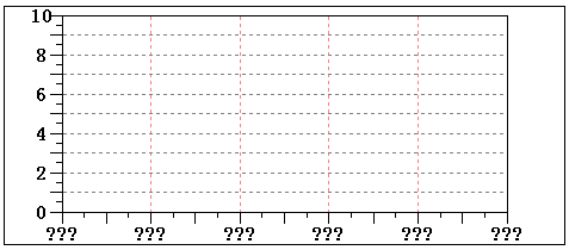


Figure 11-9-1 Insert Trend

Double click the trend and a dialog for configuration of a trend element will appear, as shown in figure 11-9-2, where you can set the options Trend, Colors, Text for tooltip and Security. Enter the variable names with the aid of the input assistant (<F2>).

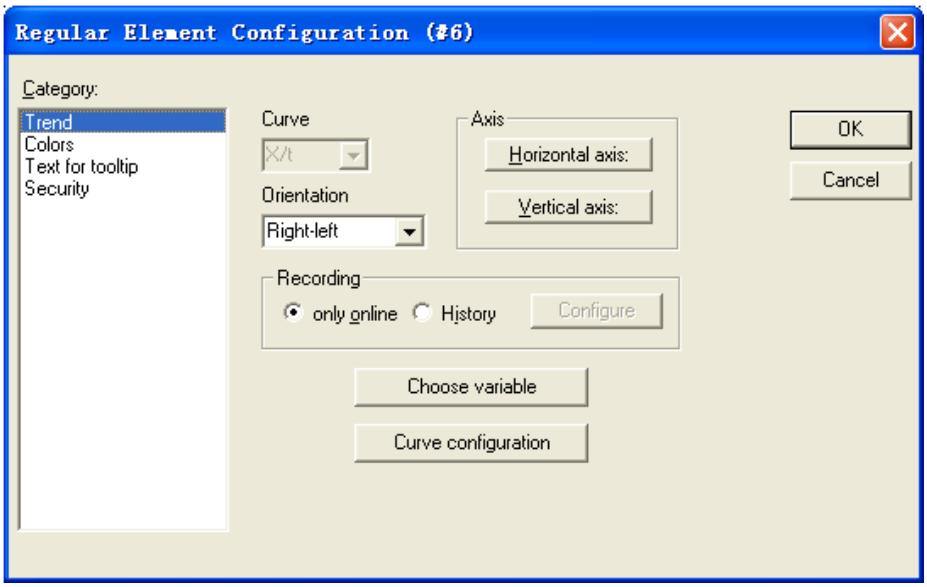


Figure 11-9-2 Trend Configuration

11.10 ALARM TABLES

You can insert an alarm table in a visualization object, as shown in figure 11-10-1.

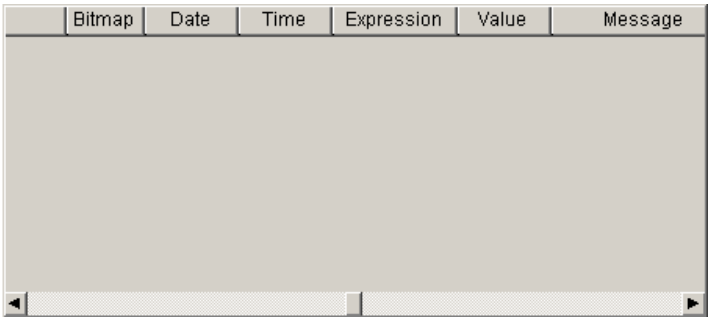


Figure 11-10-1 Insert Alarm Table

Double click the alarm table and a dialog for configuration of a alarm table will appear, as shown in figure 11-10-2, where you can set the options 'Alarm table', 'Columns', 'Settings for sorting', 'Selection settings', and 'Text for tooltip and Security'. Enter the variable names with the aid of the input assistant (<F2>).

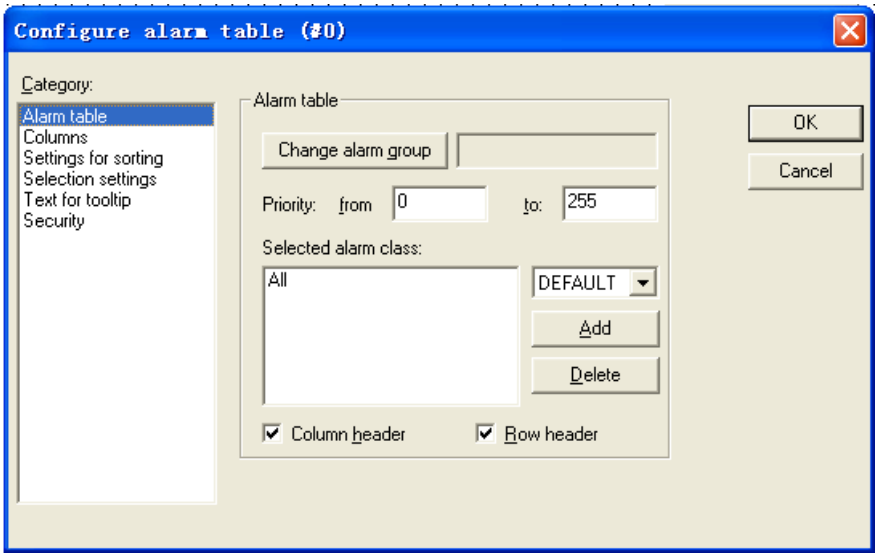


Figure 11-10-2 Configure Alarm Table

11.11 ACTIVEX ELEMENTS

You can insert an ActiveX element in a visualization object, as shown in figure 11-11-1.

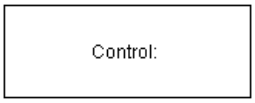


Figure 11-11-1 Insert ActiveX element

Double click the ActiveX element and a dialog “Configure ActiveX element” will appear, as shown in figure 11-11-2, where you can set the options ‘Control’, ‘Methodcalls’ and ‘Display’. Enter the variable names with the aid of the input assistant (<F2>).

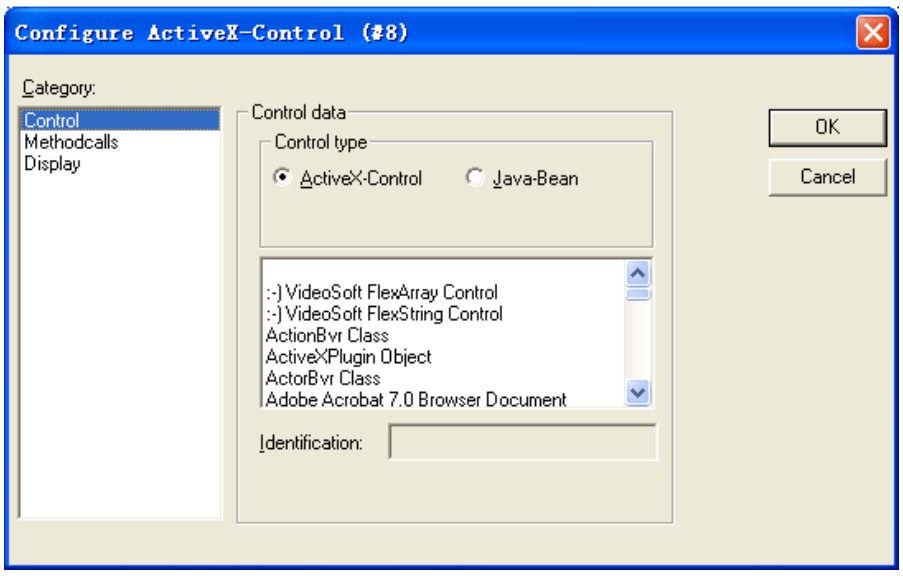


Figure 11-11-2 ActiveX-Control Configuration

11.12 VISUALIZATION APPLICATIONS

Now we discuss visualization applications with a small program as an example. In this program two signal lamps flash in turn. The program is displayed in figure 11-12-1.

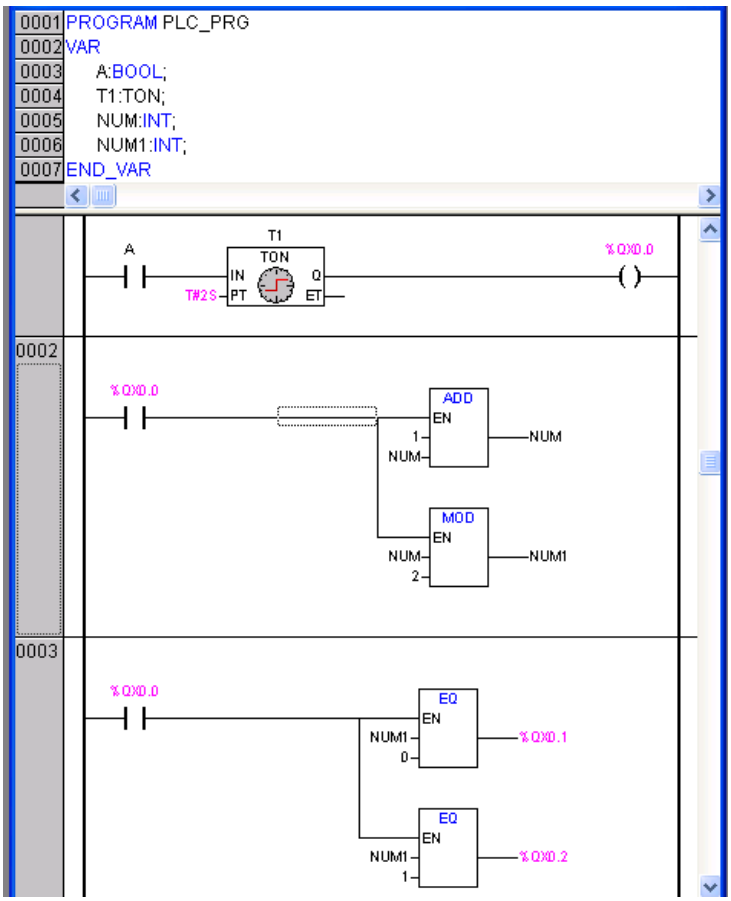


Figure 11-12-1 Program

In order to see the flash more clearly, we can define a visualization object in which the on and off signal lamps can be displayed in different colors. The on signal is displayed in green while off is displayed in red.

First, create a visualization A in which two circles stand for two signal lamps and two rectangles stand for the names of two signal lamps “Q.1” and “Q.2”, as in figure 11-12-2.

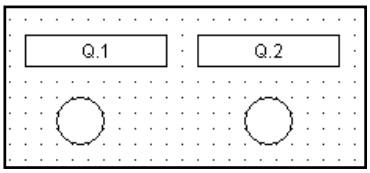


Figure 11-12-2 Visualization A

Select Q.1 and double click on it. A dialog box titled “Regular Element Configuration” will appear. Select “Colors”/ “Color”/ “Inside” and choose the green color and click the “OK” button, as shown in figure 11-12-3.

Select “Alarm color”/ “Inside” and choose the red color. The color setting of Q.2 is the same with Q.1.

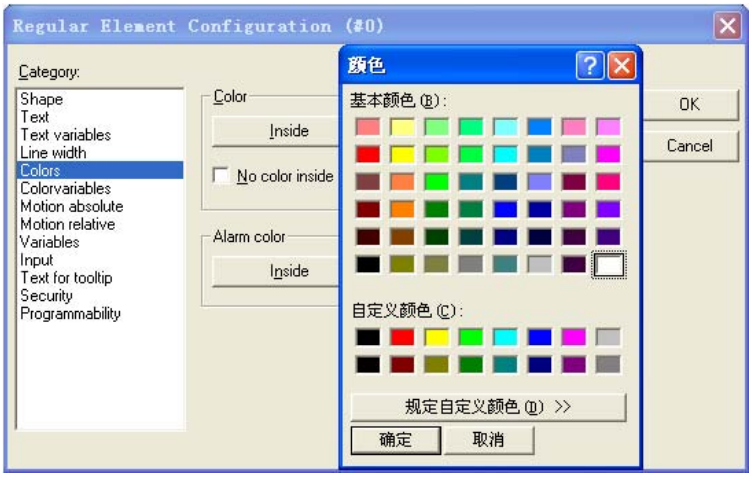


Figure 11-12-3 Color Setting

You can enter "Q.1" in the field "Category"/"Variables"/"Change color" and click the "OK" button, as shown in figure 11-12-4. In the same way, enter "Q.2" in the field "Change color" of "Q.2".

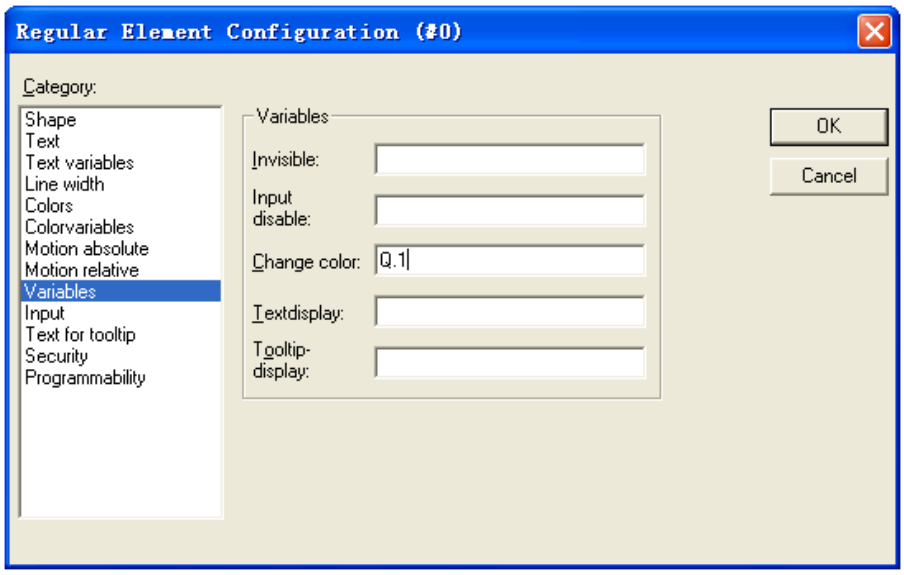


Figure11-12-4 Variable Setting

Now the configuration of visualization A is finished. The result in online mode is displayed in figure 11-12-5. Q.1 and Q.2 flash in turn.

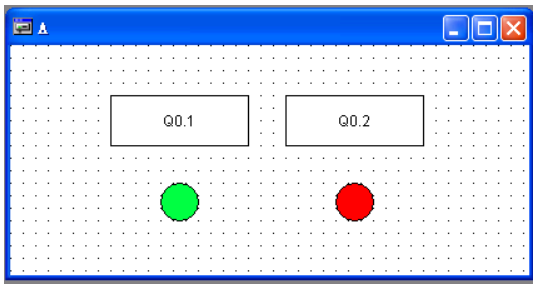


Figure 11-12-5 Result in Online Mode

Appendix

A

Appendix

12.1 MODULE MEMORY

The program memory of the modules in the following table is 28KB. In the dialog box titled "Target Settings", select "HOLLYSYS-LEC G3 CPU" for the "Configuration".

Module Type	Module Version
LM3104	A01, A02, A03
LM3105	A01, A02, A03
LM3106	A01, A05, B06, B07, B08
LM3106A	A01, A02, A03, A04
LM3107	A01, A05, B06, B07, B08

Besides, the program memory of other modules is 120KB. In the dialog box titled "Target Settings", select "HOLLYSYS-LEC G3 CPU Extend" for the "Configuration".

12.2 INPUT ASSISTANT IN POWERPRO

For users who are not familiar with standard programming languages, the command "Edit"/"Input Assistant" or <F2> provides a dialog box titled "Help Manager" for choosing possible inputs (Standard Functions, Function Blocks, Operators, Operands and Variables) at the current cursor position, as shown in figure appendix-1. The categories vary depending upon the current cursor position and the selected object.

- Standard Functions: standard function from standard library;
- User defined Functions: user defined functions written by users;
- Standard Function Blocks: standard function blocks from standard library;
- User defined Function Blocks: user defined function blocks written by users;
- FBD Operators: standard IEC operators supported by PowerPro;
- User defined Programs: user defined programs written by users;
- Conversion Operators: conversion operators for type conversion between variables of different types;

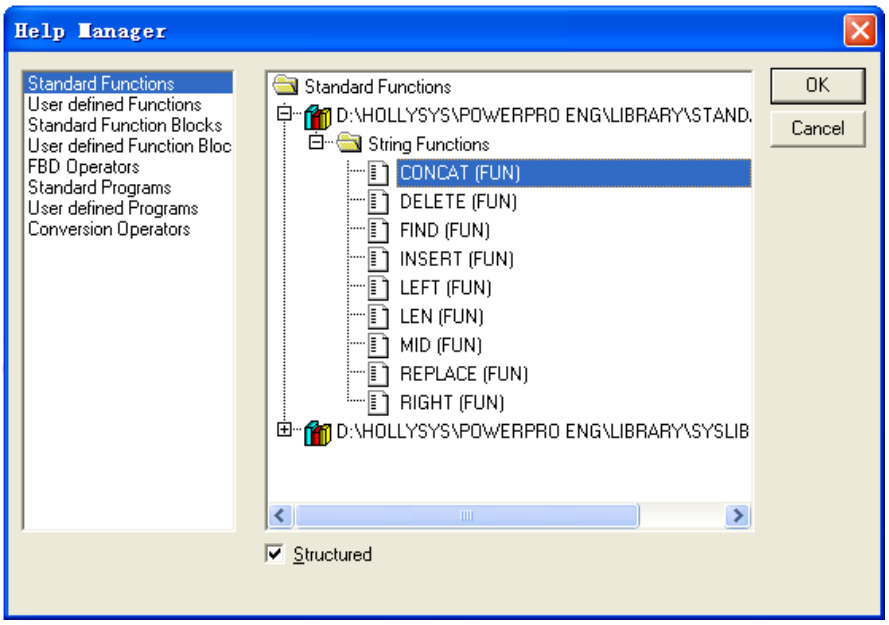


Figure Appendix-1 Help Manager

12.3 KEY COMBINATIONS IN POWERPRO

General Functions	General Functions
Move between the declaration part and the instruction part of a POU	Move between the declaration part and the instruction part of a POU
Move between the Object Organizer, the object and the message window	Move between the Object Organizer, the object and the message window
Move to the next open editor window	Move to the next open editor window
Move to the previous open editor window	Move to the previous open editor window
Right Key menu	Right Key menu
Shortcut mode for declarations	Shortcut mode for declarations
Move from a message in the Message window back to the original position in the editor	Move from a message in the Message window back to the original position in the editor
Open and close multi-layered variables	Open and close multi-layered variables
Open and close folders	Open and close folders
Switch register cards in the Object Organizer and the Library Manager	Switch register cards in the Object Organizer and the Library Manager
Move to the next field within a dialog box	Move to the next field within a dialog box
Help	Help
General Commands	General Commands
"File" /"Open"	"File" /"Open"
" File " /"Save"	" File " /"Save"
" File " /"Print"	" File " /"Print"
" File " /"Exit"	" File " /"Exit"
"Edit" /"Undo"	"Edit" /"Undo"
" Edit " /"Redo"	" Edit " /"Redo"
" Edit " /"Cut"	" Edit " /"Cut"
" Edit " /"Copy"	" Edit " /"Copy"

" Edit " /"Paste"	" Edit " /"Paste"
" Edit " /"Delete"	" Edit " /"Delete"
" Edit " /"Find next"	" Edit " /"Find next"
" Edit " /"Replace"	" Edit " /"Replace"
" Edit " /"Input Assistant"	" Edit " /"Input Assistant"
" Edit " /"Auto Declare"	" Edit " /"Auto Declare"
" Edit " /"Next Error"	" Edit " /"Next Error"
" Edit " /"Previous Error"	" Edit " /"Previous Error"
"Project" /"Rebuild all"	"Project" /"Rebuild all"
" Project " /"Delete Object"	" Project " /"Delete Object"
" Project " /"Add Object"	" Project " /"Add Object"
" Project " /"Rename Object "	" Project " /"Rename Object "
" Project " /"Open Object "	" Project " /"Open Object "
"Online" /"Login"	"Online" /"Login"
" Online " /"Logout"	" Online " /"Logout"
" Online " /"Run"	" Online " /"Run"
" Online " /"Stop"	<Shift>+<F8>
" Online " /"Breakpoint"	<F9>
" Online " /"Step over"	<F10>
" Online " /"Step in"	<F8>
" Online " /"Single Cycle"	<Ctrl>+<F5>
" Online "/"Write Values"	<Ctrl>+<F7>
" Online " /"Force Values "	<F7>
" Online " /"Release Force"	<Shift>+<F7>
" Online " /"Write/Force dialog"	<Ctrl>+<Shift>+<F7>
"Window" /"Messages"	<Shift>+<Esc>
LD Editor Commands	
“Insert”/Network (after)"	<Shift>+<T>
“Insert”/ "Contact"	<Ctrl>+<O>
“Insert”/ "Parallel Contact"	<Ctrl>+<R>
“Insert”/ "Function Block"	<Ctrl>+<B>
“Insert”/ "Coil"	<Ctrl>+<L>
"Extras" /"Paste below"	<Ctrl>+<U>
" Extras " /"Negate"	<Ctrl>+<N>
FBD Editor Commands	
“Insert”/ " Network (after)"	<Shift>+<T>
“Insert”/ "Input"	<Ctrl>+<U>
“Insert”/ "Function Block"	<Ctrl>+<B>
“Insert”/ "Assignment"	<Ctrl>+<A>
“Insert”/ "Jump"	<Ctrl>+<L>
“Insert”/ "Return"	<Ctrl>+<R>
" Extras " /" Negate "	<Ctrl>+<N>
" Extras " /"Zoom"	<Alt>+<Enter>

SFC Editor Commands	
“Insert”/ "Step-Transition (before)"	<Ctrl>+<T>
“Insert”/ " Step-Transition (after)"	<Ctrl>+<E>
“Insert”/ "Alternative Branch (right)"	<Ctrl>+<A>
“Insert”/ "Parallel Branch (right)"	<Ctrl>+<L>
“Insert”/ "Jump	<Ctrl>+<U>
" Insert " /"Zoom Action/Transition"	<Alt>+<Enter>