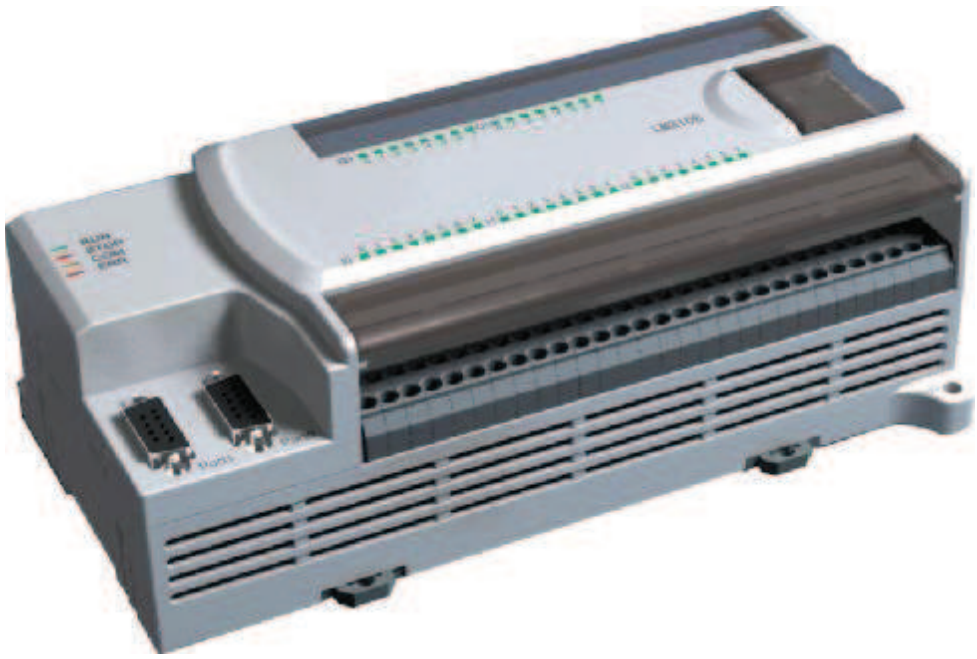
LM PLC
SOFTWARE
MANUAL



CONTENTS

READING GUIDE

The purpose of this software manual is to help you write a perfect PLC control program using PowerPro. The main contents include how to use PowerPro and how to write programs in standard programming languages.

Chapter 1 Introduce the inatallation, uninstallation and installation target of PowerPro.

Chapter 2 Overview of PowerPro and software programming environment are introduced, including the main window, menu bar, shortcut tool, object organizer and so on. Refer to chapter 2 if you want to know the software menu and shortcut.

Chapter 3 Quick introduction. Take a simple program for an example to introduce the basic steps and usage of PowerPro software. It 's suggested to learn this chapter carefully for new learners.

Chapter 4 Introduce LM series PLC storage assignment and how to manage the variables in PowerPro including address and variable declaration, variable classification. If you meet some questions in the use of address or variable, refer to this chapter.

Chapter 5 Mainly introduce the POU managing in PowerPro, such as POU creating, POU calling. In this chapter you can find the answers for some questions in programming.

Chapter 6 Based on chapter 5, PLC working mode and task management and configuration areintroduced in this chapter. The content is related to interrupt calling.

Chapter 7 After that you have known how to manage addresses, variables and POU in PowerPro, how to write a program is introduced in this chapter. Take a program written in LD language for an example to describe project creation and management including new, PLC configuration, writing a program, subprogram calling and adding comments. It's helpful for you to write a project completely.

Chapter 8 The compilation, download and debugging are the main content in this chapter. The commands you need to execute and the questions you meet are all included in this chapter.

Chapter 9 Introduce the programming languages in PowerPro including FBD, IL, ST and SFC.

Refer to this chapter carefully if you want to write a program in one of these languages.

Chapter 10 Introduce some special functions in PLC including Modbus communication, interrupt, analog functions, DP communication and Ethernet communication. Refer to this chapter when you use these functions.

Chapter 11 Introduce the PowerPro visualization. Visualization is the advanced application of PowerPro software. If you want to debug with a visual interface, please refer to this chapter.

CHAPTER 1 INSTALLATION

The installation and uninstallation of PowerPro software are introduced in this chapter, and especially the InstallTarget is described in details. The software installation is introduced in section 1.1 and the uninstallation is introduced in section 1.2. If you 're familiar with Windows operations you can skip over the two sections. How to install the InstallTarget is introduced in section 1.3. PowerPro is a powerful PLC programming software and the purpose of InstallTarget is to configure PowerPro as the programming software for LM series PLC. It 's suggested to read this chapter carefully if you are a new learner.

1.1 SOFTWARE INSTALLATION

In the computer with Chinese Windows operating system, insert an installing CD of PowerPro software into the drive and select " LM series PLC programming software PowerPro2.1.1" in the pop-up interface. You can also find "\\programming software PowerPro2.1.1ch\Setup.exe" in the CD and double click on it to install PowerPro software. In the installation interface click the "Next" button, shown in figure 1-1-1.

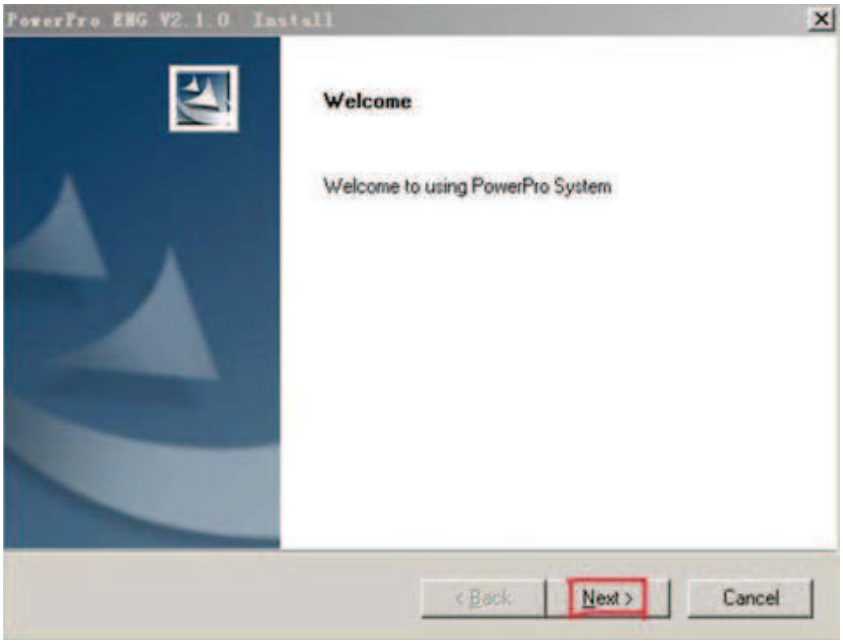


Figure 1-1-1 Installation Steps (1)

Accept the license agreement and click "Yes", shown in figure 1-1-2.

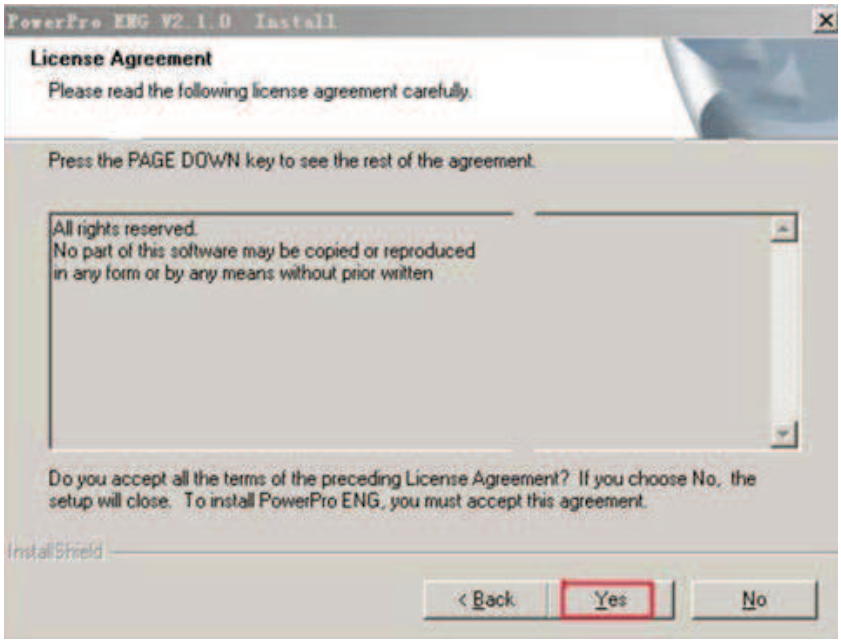


Figure 1-1-2 Installation Steps (2)

Choose installation directory. The default directory is D:\PowerPro\PowerPro ENG and it ’s suggested not to change it. Click “Browse” to choose other directories. Click “Next” to start installation, shown in figure 1-1-3.

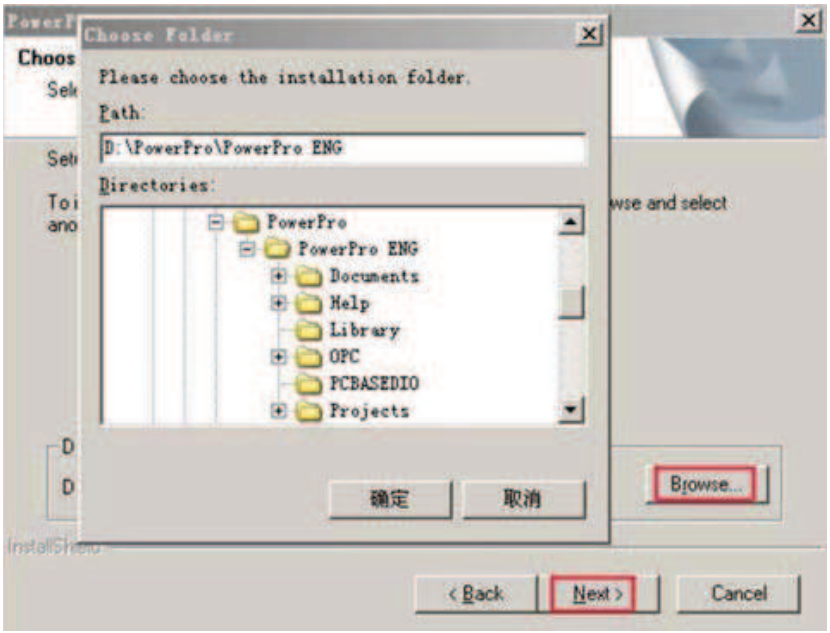
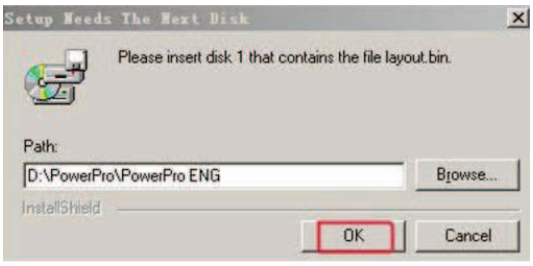


Figure 1-1-3 Installation Steps (3)

After installation an interface “About to lanch the MSXML 3.0 setup” appears and click “OK”, shown in figure 1-1-4.



Then an interface "About to lanch the MSXML 3.0 SP4 setup " appears and click "OK", shown in figure 1-1-10.

Figure 1-1-10 Installation Steps (10)

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

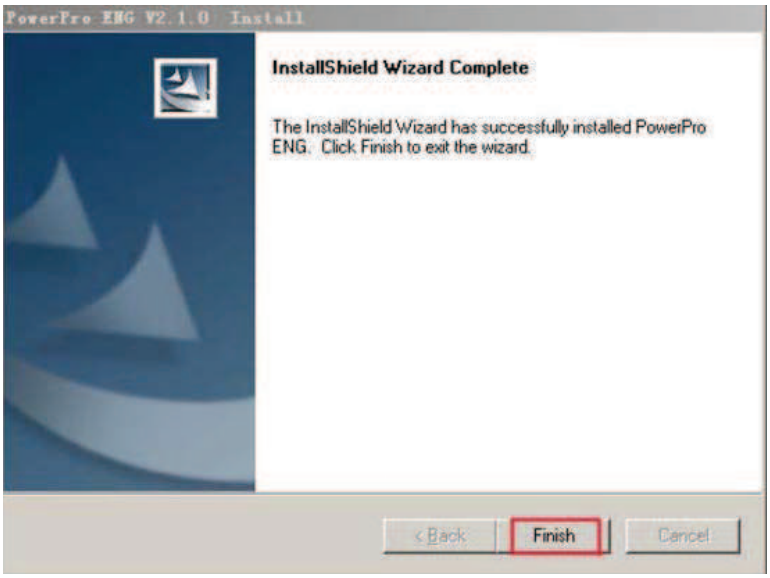


Figure 1-1-11 Installation Steps (11)

1.2 SOFTWARE UNINSTALLATION

If a lower PowerPro version has been installed, it must be uninstalled before installing the new version. Select PowerPro in "Control Panel"/"Add/Remove Programs" and click "Add/Remove" to uninstall the program, shown in figure 1-2-1.

Figure 1-2-1 Software Uninstall

Note:

Exit the Gateway.exe in system tray on the lower right corner of thedesk before uninstallation!

1.3 Install Target

PowerPro is the development platform of PLC control trategy. Before using PowerPro one must first do "Install Target" to choose the software running platform for PLC modules. Because the installed content is universal for all projects, so "Install Target" can be done only once before using PowerPro.

Introduce the concrete steps of"Install Target" as follows.

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

Figure 1-3-1 Install Target (1)

The widow "InstallTarget" appears, shown in figure 1-3-2. Click "Open..." button, choose the file "C16x_PowerPro.tnf" in the popup window and click "Open" to close the window.

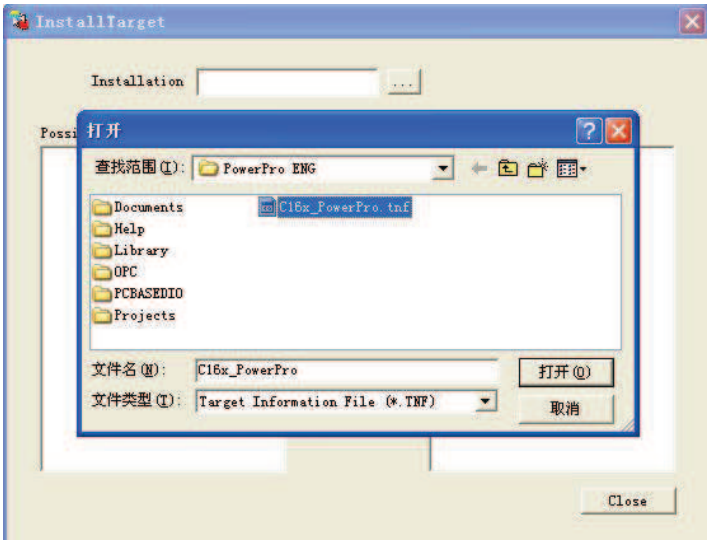


Figure 1-3-2 Install Target (2)

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

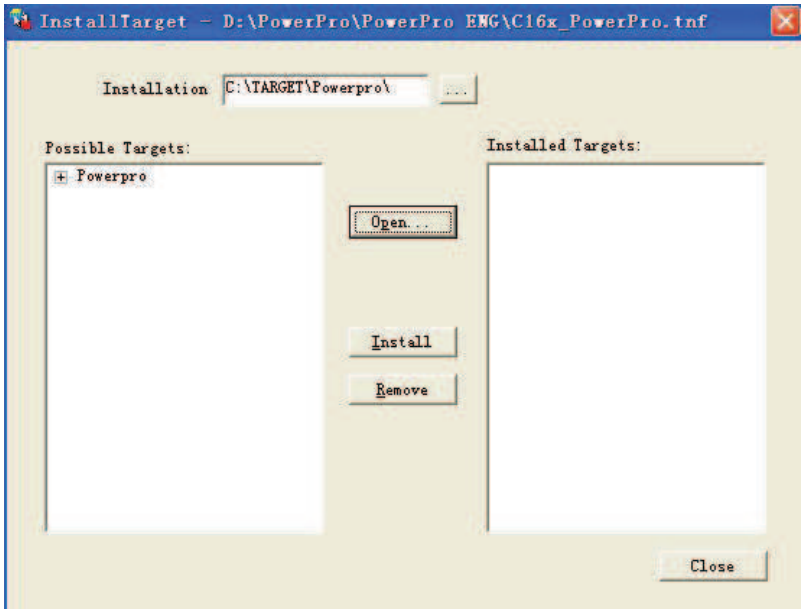


Figure 1-3-3 Install Target (3)

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

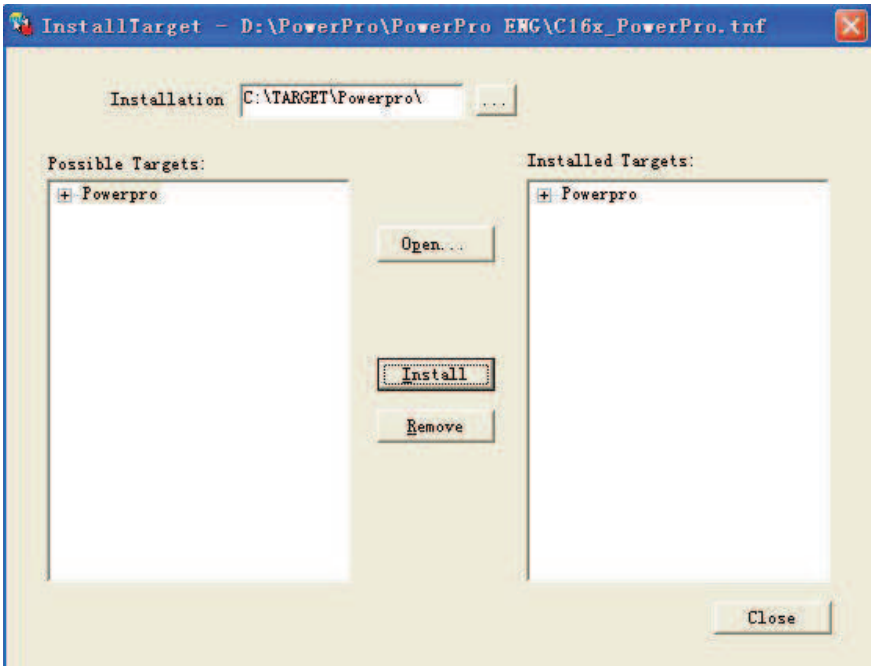


Figure 1-3-4 Install Target (4)

CHAPTER 2 PowerPro OVERVIEW

Start PowerPro to enter PowerPro programming environment.

The PowerPro programming environment including programming interface and menu commands are introduced in this chapter so that the users can know and be familiar with the programming environment. Refer to this chapter about menu items at any time needed.

2.1 BRIEF INTRODUCTION OF PowerPro

PowerPro is a programming tool based on Windows and is developped specially for LM by ANN Co., Ltd. PowerPro has the editing, simulation, debugging functions for control method, and is a standard package for hardware configuration and software programming of LM.

Compared with tranditional programming software, PowerPro has the following features and functions:

- **Programming language standardization**

In the Mid and Late 1990s in last century, IEC promoted international standards of programming languages in automation industry. First it's IEC1113-3 standard and then it's modified to IEC61131-3 standard to unify PLC, NC and DCS programming language to promote the programming technology. PowerPro programming software is accord with IEC61131-3 standard completely and the program can be written in one of the IEC programming languages which include IL, ST, SFC, FBD, LD or CFC.

- **Internal device**

LM series PLC has not so many internal devices as normal PLC, like timer, counter which isreplaced by variables. The concept of variable is special in PowerPro which is similar with the form of advanced language. The number of variable declarations depends on how many you need. Variable can be named by its function which is much easier to identify than device numbers. Variables can be classed into global and local, input and output, hold on power-off or not. At the same time, PowerPro has powerful calculation function and a number of data types can be defined, including not only bool, byte, word, doubleword but also pointer, enumerate, multi-dimension array, single precision floating point. For detailed parameter description see section 4.4.

- **Program modularization**

The program organization in PowerPro is modularized completely. The concept of POU (Program Organization Unit) is promoted in PowerPro. The POU of PowerPro includes program, function and function block of which a project is composed. Organizize a program in PowerPro is realized by calling other POUs by the main program. It's convenient for multi-person programming and also convenient for program reuse, read, debug and the momory is saved to ensure program safety. At the same time, PowerPro is an open system and the users can develop their own instructions when necessary. About POU see chapter 5.

- **Module setting software**

PowerPro is an open system. On one hand, the users can develop their own instructions when necessary; On the other hand, PowerPro opens up a large number of PLC parameters and module settings to users in the form of instructions, and depending on the user 's requirement the users can

set parameters in programs, like serial port communication parameter setting.

- **Programming and monitoring integration**

PowerPro has its special visualization and alarm functions and a visualized interface is provided when running or debugging a program. In addition, PowerPro has powerful simulation and debugging functions, you can test the logical correctness of your program easily. About simulation and debugging see section 8.4. About visualization see chapter 11.

2.2 PROGRAMMING WINDOW

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

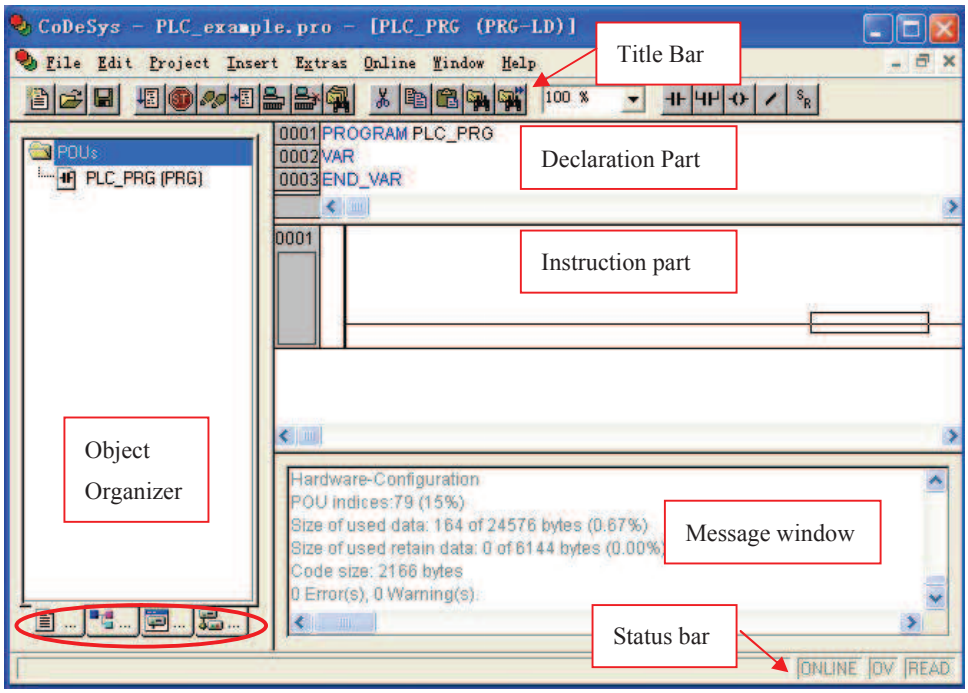


Figure 2-2-1 Main Window

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

- Title Bar: Include Menu bar (It contains all menu commands) and Tool bar (optional), with some buttons for faster selection of menu commands.
- Object Organizer: Include POUs, Data Types, Visualizations and Resources.
- Variable Declaration Part: Display the variables declared or defined in program.
- Program Part: 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

After PowerPro is running, the Title bar located at the upper edge of the main window is shown in figure 2-2-2, including ("File", "Edit", "Project, "Insert", "Extra", "Online", "Window", "Help" and Tool bar (optional), with some buttons for faster selection of menu commands.

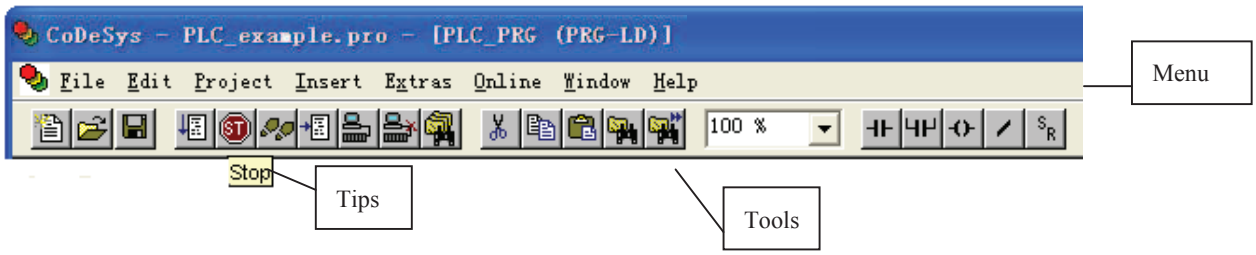


Figure 2-2-2 Title Bar

Introduce shortcut symbols in tool bar first. If you hold the mouse pointer for a short time on a symbol in the tool bar, then the name of the symbol is shown in a tool tip, shown in figure 2-2-2. The function is forbidden in current window if the menu commands and shortcut symbols turn into grey.

- 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'.

See section 2.3 for menu commands.

222 Object Organizer

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

The register card **POUs** is used for managing POUs, such as new subprogram and new interrupt service routine. The register card **Data Type** is used to manage user-defined data type which is supported by PowerPro. The register card **Visualization** is used to manage visualization. The register card **Resources** is used for the functions such as PLC hardware configuration, adding instructions, workspace and interrupt setting.

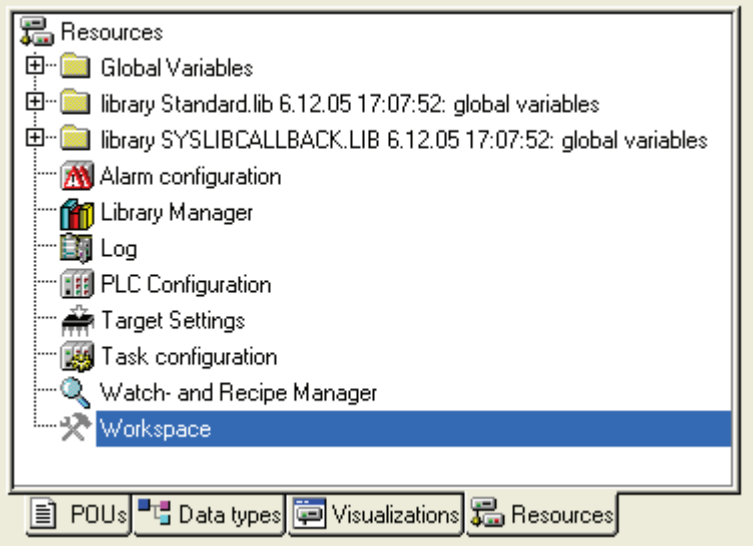


Figure 2-2-3 Object Organizer

223 Declaration Part

The declaration part is seated on the upper right corner of object organizer. The data in PowerPro are classed into address and variable. Variables can be not assigned specific address and identified by symbols, such as “start”, “run” and the same symbol stands for the same variable. The difference between variable and address is that variable must be defined when using while address can be used directly. The declaration part is used to display all the declared variables.

There are two ways for variable declarations. First, autodeclaration when programming and the variable is displayed in declaration part, shown in figure 2-2-4; Second, declare the variable at declaration part directly. Refer to section 4.4 for variable declaration.

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

Note:

1、 The same variable symbol can't be defined as two different data types.

2 The address data can't be displayed in declaration part.

3 When a defined variable is deleted in programs, the variable declaration in declaration part will not be deleted automatically.

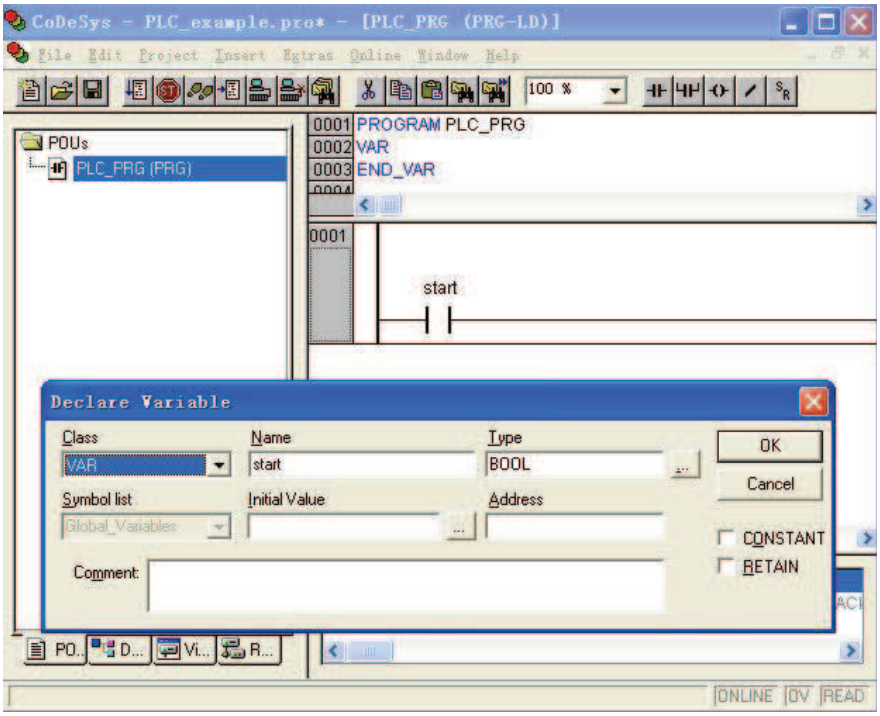


Figure 2-2-4 Autodeclaration

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

2.2.4 Instruction Part

Instruction Part is located under declaration part. In the register card "POUs" in object organizer, the instruction part is the editor window of program, function and function block and it's used to edit control algorithm. The programming environment varies from the selected programming language. According to the characteristic of different languages, the programming language is classed into textual languages and graphic languages. LD, SFC, FBD and CFC belong

to graphic languages and IL and ST belong to textual languages. The text editor is a text editor with the usual capabilities of Windows text editors.

Refer to section 7.4 for the operations in instruction part. Refer to chapter 9 for other programming languages.

225 Message Window

The message window is underneath the work space in the main window. It contains all messages from the previous compilation, errors, warnings or comparisions, shown in figure 2-2-6. If you doubleclick with the mouse in the message window on a message, the editor opens with the object, and the relevant line of the object is selected. With the commands ‘Edit’ ‘Next error’ and ‘Edit’ ‘Previous error ’ (Shift+F4 combination key) you can 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 and so on. See section 8.2.3 for details.

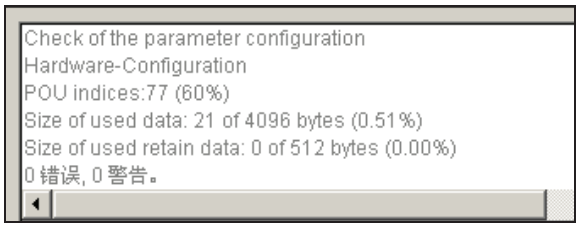


Figure 2-2-6 Messge Window

226 Status Bar

The status bar at the bottom of the window frame of the main window gives you information about the current project and about menu commands. If an item is relevant, then the concept appears on the the right side of the status bar in black script, others in gray script.

23 MENU

231 File

File menu is shown in figure 2-3-1, and detailed functions are introduced.

Figure 2-3-1 File Menu

- File [F]/New [N]: Create an empty project with the name "Untitled".
- File [F]/New from template [T]: Open any desired project as a "template" project. The dialog for opening a project file will be available and the selected project will be opened. PLC cannot support the two functions of "Open project from PLC " and "Open project from source code manager".
- File [F]/Open [O]: Open an already exiting project.
- File [F]/Close [C]: Close the currently-open project.
- File [F]/Save [S]: Save any changes in the project.
- File [F]/Save as [A]: Save the current project in a new file or a new directory.
- File [F]/Save /Mail Archive [H]: Create a project archive file. All files which are referenced by and used with a project can be packed in a compressed zip file. The zip file can be stored or directly can be sent in an email.
- File [F]/Print [P]: Print the content of the current window.
- File [F]/Printer Setup [T]: Setup the printer parameters. If you click on the Printer Setup button, then a dialog appears. Here you can determine the printer name, the page size, the number of copy, direction and so on. You can also setup the print quality and print layout.
- File [F]/Exit [E]: Exit from PowerPro.

232Edit

Edit menu is shown in figure 2-3-2, and detailed functions are introduced.

Figure 2-3-2 Edit Menu

- Edit [E]/Undo [U]: Undo the action which was most recently executed.
- Edit [E]/Redo [E]: Restore an action you have undone.
- Edit [E]/Cut [T]: Transfer the current selection from the editor to the clipboard. The selection is removed from the editor.
- Edit [E]/Copy [C]: Copy the current selection from the editor to the clipboard. This does not change the contents of the editor window.
- Edit [E]/Paste [P]: Paste the content of the clipboard onto the current position in the editor window.
- Edit [E]/Delete [D]: Delete the selected area from the editor window.
- Edit [E]/Find [F]: Search for a certain text passage in the current editor window.
- Edit [E]/Find next [N]: Execute a search with the same parameters as with the most recent action ‘Edit’ ‘Find’.
- Edit [E]/Replace [R]: Search for a certain passage just as with the command ‘Edit’ ‘Find’, and replace it with another.
- Edit [E]/Input Assistant [A]: Fast input. At the current cursor position in the editor window by pressing F2 a dialog box for chosing possible inputs like operators, functions, function blocks, variable types appears. In the left column choose the desired input category, select the desired entry in the right column, and confirm your choice with OK. This inserts your choice at this position. See section 7.4.3 for details.
- Edit [E]/Auto Declare: A dialog for Declare Variable appears.
- Edit [E]/Next Error: Show the next error or warning after the incorrect compilation of a project.
- Edit [E]/Previous Error: Show the previous error or warning after the incorrect compilation of a project.
- Edit [E]/Micros: PLC cannot support the function.

2.3.3 Project

Project menu is shown in figure 2-3-3, and detailed functions are introduced.

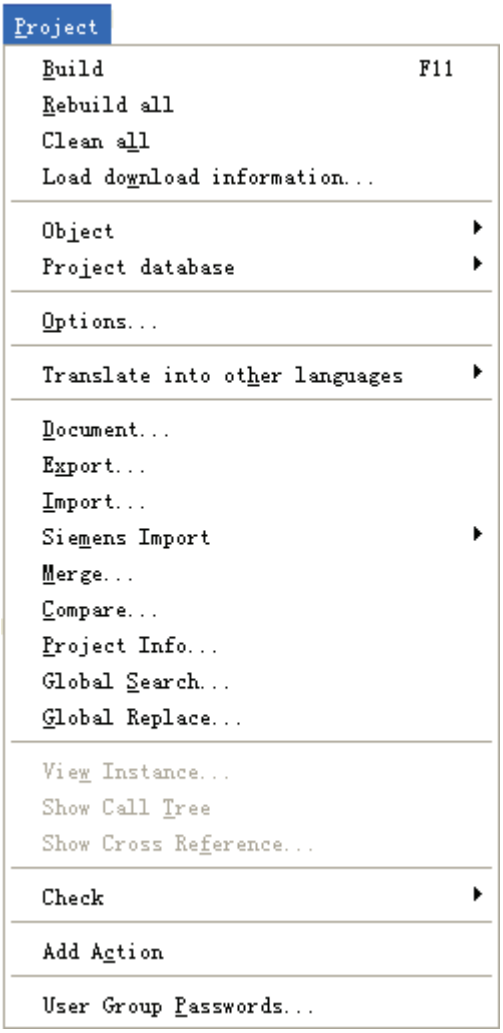


Figure 2-3-3 Project Menu

- Project [P]/Build [B]: Compile the changed POUs.
- Project [P]/Rebuild all [R]: Compile all the POUs completely.
- Project [P]/Clean all [L]: Delete all the information from the last download and from the last compilation. The purpose of "Clean all" is to re-build download file in the next compilation, but it does 't influence the user program in PLC. Perform "Build" or "Rebuild all" after "Clean all", no matter the program is changed or not, in the next download a tip appears: "The program has changed! Download the new program? ", and finish download according to the tips. It 's different with "Project/Clean all" which will clean the program in PLC and re-initialize PLC system.
- Load download information: PLC can't support this function.
- Project [P]/Object [J]: For the selected object do the operations of delete, add, rename, convert, copy, edit and properties, shown in figure 2-3-4.

Figure 2-3-4 Object Sub-menu

Delete: Delete the currently selected object.

Add: Create a new object.

Rename: Give a new name to the currently-selected object.

Convert: Convert POUs from the languages SFC, ST, FBD, LD and IL into one of the three languages IL, FBD and LD. For example: convert language LD to IL or FBD, shown in 2-3-5.

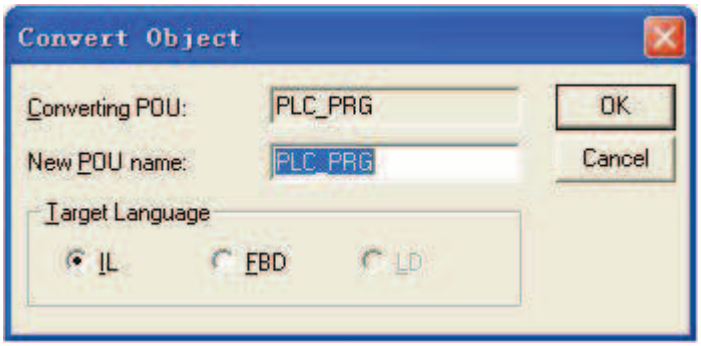


Figure 2-3-5 Convert Object

Copy: Copy a selected object and save it under a new name.

Edit: Open the editor window of the selected POUs. You can also open the editor window by double-click the name of POUs.

Properties: Set access rights to the current program, shown in figure 2-3-6.

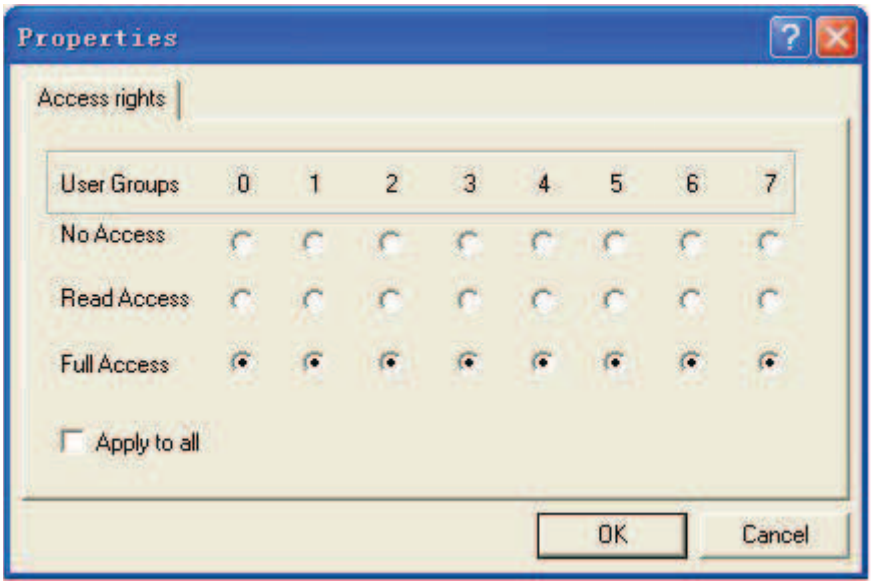


Figure 2-3-6 Properties Dialog

- Project [P]/Project database [J]: PLC can't support this function.

- Project [P]/Options [O]: Set the parameters of current project, for Load&Save, Directories, Passwords and so on. See section 7.6 for details.

- Project [P]/Translate into other languages [H]: Translate the current project file into another language, shown in figure 2-3-7.

Create translation file: Create a translation file for the current project with the extension".tlt".

Translate this project: Translate the current project to target language.

View translated project: View the translated project files.

Toggle translation: Translate this project will turn into gray once Toggle translation is selected and can't edit it. Translate this project is permited only after Toggle translation is selected again.

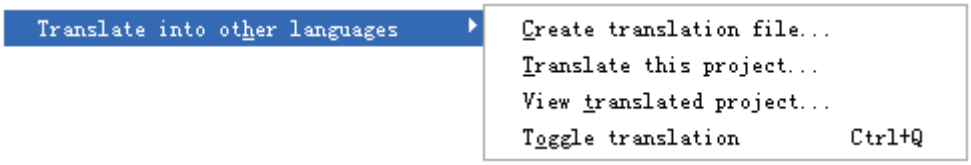


Figure 2-3-7 Language Dialog

- Project [P]/Document [D]: Print the documentation of your entire project with all files or partial files.
- Project [P]/Export [E]: Export the selected POUs in current project to a *.EXP file.
- Project [P]/Import [I]: Import a *.EXP file to current project, and generally it 's used between POUs in different projects. See section 7.5.2 for details.
- Project [P]/Siemens Import [M]: PLC cannot support the function.

Figurre 2-3-8 Siemens Import

- Project [P]/ Merge [M]: Can merge the needed objects from other projects into your own project.
- Project [P]/Compare [C]: Compare the "POUs", "Data Types ", "Visualizaation", "Resources" of the current project with that of other projects, and output the results in message window.
- Project [P]/Project info [P]: Show the information about the the current project, such as file name, directory path and so on.
- Project [P]/Global Search [S]: Search for the location of a text in some objects of the current project.
- Project [P]/Global Replace [R]: Search for the location of a text in some objects of the current project and replace this text by another.
- Project [P]/View Instance [W]: Open and display the instance of the function block which is selected in the Object Organizer. 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'.)
- Project [P]/Show Call Tree [T]: Show the call tree of the object chosen in the Object Organizer. Before this the project must have been compiled without any error (see 'Rebuild all'). See section 8.2.1 for details.

- Project [P]/Show Cross Reference [F]: Show the cross reference of variables and programs. See section 8.2.2 for details.
- Project [P]/Check: View the information about variables, shown in figure 2-3-9. See section 8.2.3 for details.

Unused variables: Search for the variables that have been declared but not used in the program.If there are no unused variables, the result is displayed in the message window: No unused variables found.

Overlapping Memory Areas: Test whether in allocation of variables via the "AT" declaration overlaps at specific memory areas. If there are no variables with overlapping memory area, the result is displayed in the message window: No variables with overlapping memory area found.

Concurrent Access: Search for memory areas of addresses which are referenced in more thanone task. If there are no concurrent accesses, the result is displayed in the message window: No concurrent accesses found.

Multi Write Access on Output: Search for memory areas to which a single project gains write acess at more than one location. If there are no multi write access on output, the result is displayed in the message window: No outputs found which are written to at more than one location.

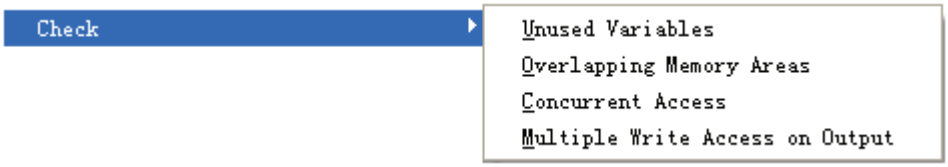


Figure 2-3-9 Check Sub-menu

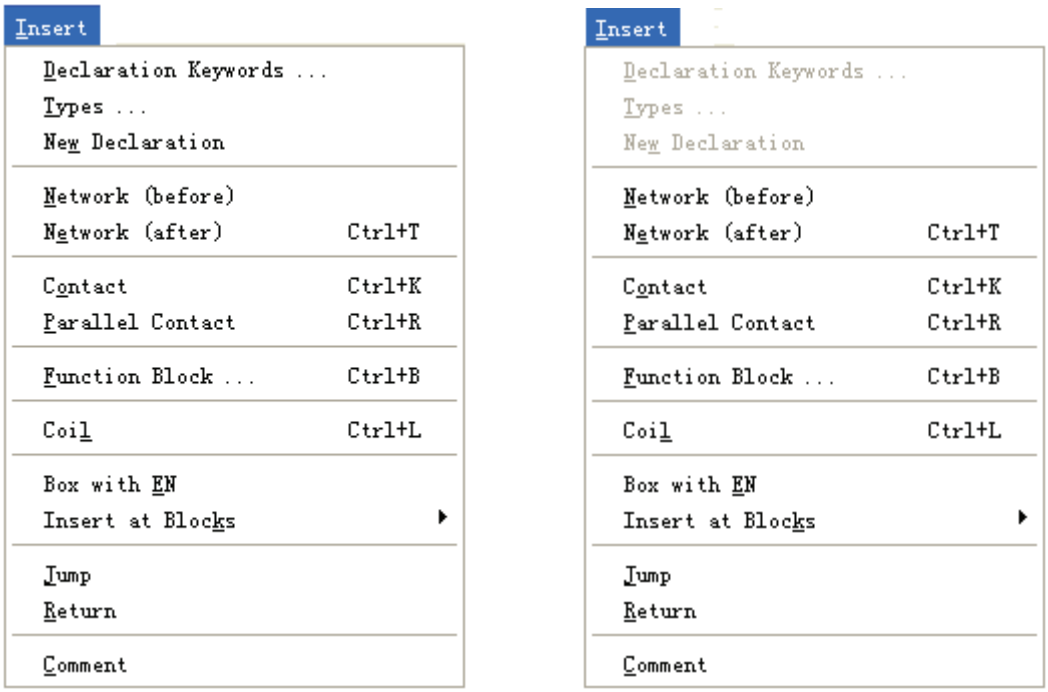
The options in "Project"/"Check" have the similar functions with that in "Workspace"/ "build"/ "Check automatically" in "Resources" register card. The only difference is that the options in "Project / Check" are available only after the project has been compiled without any error and can be check one by one. With "Check automatically" more than one option can be activated at each compilation of the project.

- Project [P]/Add Action [A]: Add an action allocated to the current program.
- Project [P]/User Group Passwords [P]: Assign passwords for user group. Enter the passwords when opening a project next time. You can operate it if the passwords are correct, or else it's forbidden to operate.

2.3.4 Insert

When programming you can insert New Declaration, Function Block, Box with EN and function. The insert items vary from different programming language and cursor positions. Take "LD" language for an example, the "Insert" menu is introduced in several situations.

First, the items when the cursor in declaration part and in instruction part are different, shown in figure 2-3-10. Here you can insert Function Block, Coil, Comment and Declaration Keywords, Types in LD language. How to insert each item is introduced in section 7.4.



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

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

Second, in "PLC Configuration" in "Resources" register card you can insert element or subelement. Insert the CPU module first and then insert the related expansion modules, 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 expension module.

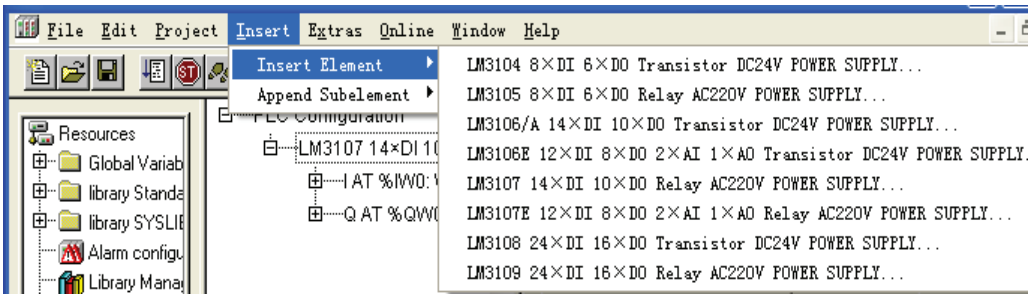


Figure 2-3-11 "Insert" Menu (2)

Last, in "Watch- and Recipe Manager" in "Resources" register card you can use the command "Insert"/"New Watch List " or the context menu in the left column of the "Watch- and Recipe Manager" to "New Watch List" and name it, shown in figure 2-3-12.

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

235Extras

The options of "Extras" menu vary according to different programming languages or different cursor positions. Take "LD" for an example, "Extras" menu is introduced in different situations.

First, in “Resources” register card, “Extras” menu is shown in figure 2-3-13.

Figure 2-3-13 “Extras” Menu (1)

- Negate: Use this command to negate a contact or a coil. The coil now writes the negated value of the input connection in the respective Boolean variable. Right at this moment, a negated contact switches the status of the input to the output, if the respective Boolean variable carries the value FALSE.
- Set/Reset: CoilS can also be defined as set or reset coils. One can recognize a set coil by the “S” in the coil symbol. If the variable has once been set at TRUE, then it remains so until it is reset again. One can recognize a reset coil by the “R” in the coil symbol. If the variable has been once set at FALSE, then it remains so until it is reset again.
- Zoom: Shortcut is “Alt+Enter”. In LD select a function block and press “Alt+Enter”, then the library manager is called up and the details are displayed.
- Options: In LD, with the command “Options” you open a dialog box “Function Block and Ladder Diagram Options ”, shown in figure 2-3-14, in which you can set different options for your LD POU. See section 7.4.9 about options.

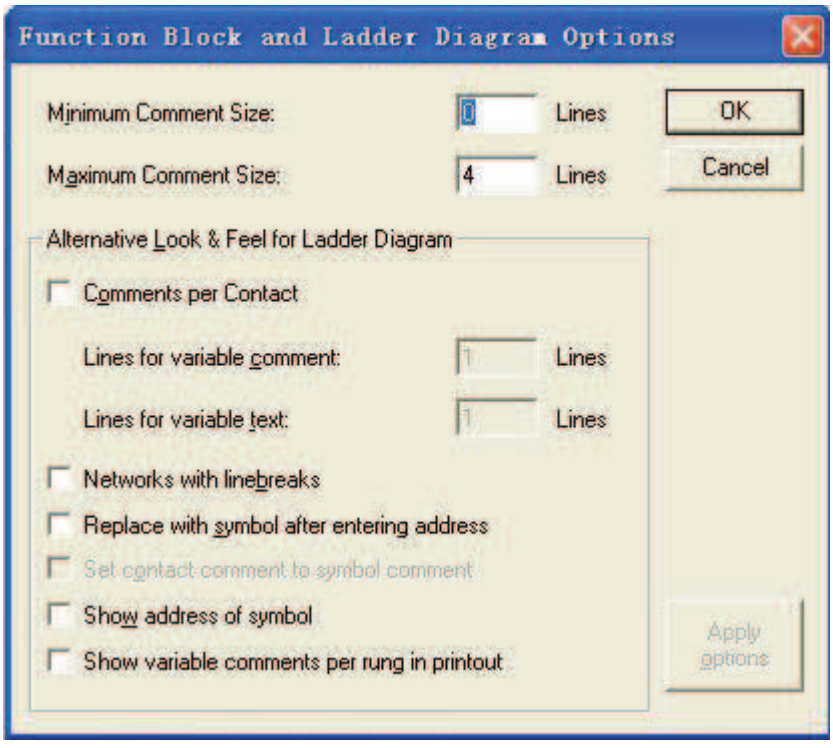


Figure 2-3-14 “Extras” Menu (2)

In “PLC Configuration” of the register card “Resources”, “Extras” menu is shown in figure 2-3-15.

Figure 2-3-15 “Extras” Menu (3)

- Replace element: Replace CPU Modules. PLC can’t support the function.
- Calculate addresses: Calculate module addresses automatically.
- Add configuration file: No need to add any configuration file for PowerPro.
- Standard configuration: A dialog shown in figure 2-3-16 appears, restore default configuration by clicking Y and keep current configuration by clicking N.
- Properties: If there is a “√” in front of “Properties”, base parameters, module parameters and communication parameters will display, or else they will not display.

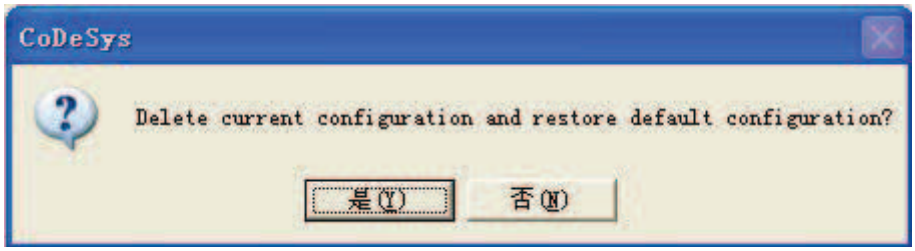


Figure 2-3-16 “Extras” Menu (4)

In “Watch- and Recipe Manager ” of the register card “Resources”, “Extras” menu is shown infigure 2-3-17.

Figure 2-3-17 “Extras” Menu (5)

- Monitoring active: Active watch list to monitor variables in a list if there is a “√” in front of the menu item.
- 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 in the extension*.wtc.
- Load Watch List: Load the renamed watch list in the extension*.wtc.

2.3.6 Online

"Online" menu is a tool used for program download and debug, shown in figure 2-3-18.

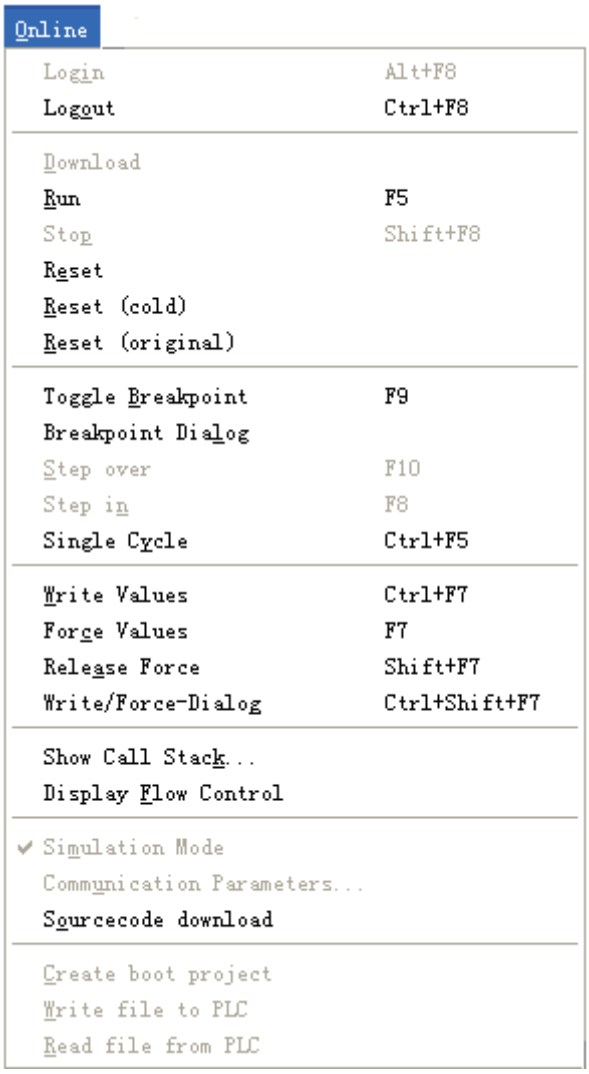


Figure 2-3-18 "Online" Menu

- Online [O]/Login [I]: Connect the PowerPro with the PLC. When the program in PowerPro is the same with that in PLC, the system change into debug mode. If they are different, a tip for program downlad appears. See section 8.3.3 for details.
- Online [O]/Logout [X]: The connection to the PLC is broken and is shifted to the offline mode.
- Online [O]/Download [D]: Load the compiled project in the PLC. It's valid only after the connection between PLC and PowerPro is build. See section 8.3.3 for the differences between download and login.
- Online [O]/Run [R]: Start the program and change into run mode.

- Online [O]/Stop [P]: Stop the execution of the program.
- Online [O]/Reset [E]: Stop running program, and initialize the variables with a specific value. Retain variables keep current values.
- Online [O]/Reset (cold) [T]: Stop running program, and initialize all the variables with a specific value.
- Online [O]/Reset (original) [O]: Reset all variables to their initialization values and erase the user program in PLC. It 's different with "Project" "Clean all" which is to delete all the information from the last download and from the last compilation and re-build download file in the next compilation.
- Online [O]/Toggle Breakpoint [B]: Set or remove a breakpoint in present position. The program will stop at breakpoint position, and display in red background. Use the commands "Online /Run", "Online /Step over" or "Online /Step in" to continue running the program. See 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 execute and stops after its execution.
- Online [O]/Step in [N]: If the present position is a call-up of a function or of a function block, then the command will proceed on to the first instruction in the called POU. In other situations it's the same with "Online/ Step over".
- Online [O]/Single Cycle [Y]: Execute a single PLC cycle and stop after this cycle. See section 8.4.8 for details.
- Online [O]/Write Values [W]: Set the variables to user defined values at the beginning of a cycle in debugging.
- Online [O]/Force Values [C]: Set the variables to user defined values. The setting occurs in the run-time system, both at the beginning and at the end of the cycle. The function remains active until it's explicitly suspended by the command "Online" "Release force".
- 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 by other programs at once. For "Online/ Force Values", variables are set at the end of the cycle until it 's explicitly suspended by the command "Online" "Release force".
- Online [O]/Show Call Stack [K]: In the simulation mode, show a dialog box with a list of the POU call stack.
- Online [O]/Display Flow Control [F]: Toggle display of flow control.
- Online [O]/Simulation Mode [M]: If simulation mode is chosen, then a check "√" will appear in front of the menu item. In the simulation mode, the user program runs on the local PC under Windows. This mode is used to test the project. If the program is not in simulation mode, then the program will run on the PLC.
- Online [O]/Communication Parameters [U]: Set the communication parameters between local PC and PLC modules.
- Online [O]/Sourcecode download [O]: PLC can't support the function.
- Online [O]/Create boot project [C]: Take the project downloaded in PLC as a default project when starting the PLC. A boot pproject is a user program which is saved in flash

of PLC and can run after electrify. If the codes of boot project are different from the codes downloaded last time, a tip shown in figure 2-3-19 appears: The current codes are different from the codes downloaded last time, continue? Generally click Y to create a boot project. In offline mode a new *.ri file might be created. See section 8.3.3 for the difference among “Online” “Create boot project”, “Login” and “Download”.

- Online [O]/Write file to PLC [W]: PLC can’t support the function.
- Online [O]/Read file from PLC [R]: PLC can’t support the function.

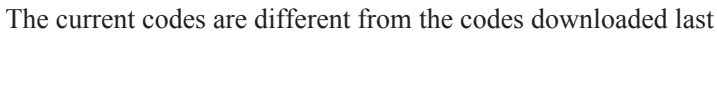


Figure 2-3-19 Create Boot Project

2.3.7 Window

“Window” menu is shown in figure 2-3-20 and detailed functions are introduced.

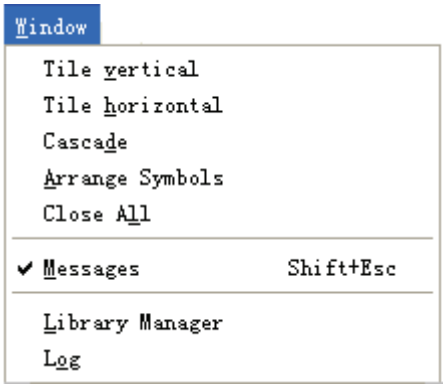


Figure 2-3-20 “Window” Menu

- Window [W]/Title 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 in 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.

238 Help

Help menu is shown in figure 2-3-21 to provide help about the software and the detailed functions are introduced.

Figure 2-3-21 "Help" Menu

- Help [H]/Contents [C]: Open help topics and list the related items for convient and fast find-out.
- Help [H]/Search [S]: Open a dialog box, and enter the text to find.
- Help [H]/About [A]: Information about software name and version.

24 SHORTCUT TOOL

Introduce shortcut symbols in tool bar first. If you hold the mouse pointer for a short time on a symbol in the tool bar, then the name of the symbol is shown in a tool tip. The function is forbidden in current window if the menu commands and shortcut symbols turn into grey.

241 File Tool

-  New: Create an empty project with the name "Untitled".
-  Open: Open an already existing project.
-  Save: Save any changes in the project.

242 Debug Tool

-  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.
-  Step Over: This command causes a single step to execute and stops after its execution. If a function block or function has been reached, then program will go to the next instruction in the POU.
-  Breakpoint: Set or remove a breakpoint in present position.

-  Login: Combine the programming system with the PLC (or start the simulation program) and changes into the online mode.
-  Logout: The connection to the PLC is broken, or the simulation mode program is ended and is shifted to the offline mode.
-  Global Search: Search for a certain text in the entire project.

243 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.

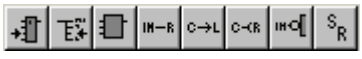
   Find: Search for a certain text passage in the current editor window. Open "Find" dialog and enter the series of characters you are looking for in the field "**Find what**". In addition, you can decide whether the text you are looking for **Match whole word only** or not, or also whether **Match case** is to be considered, and whether the search should proceed **Up** or **Down** starting from the current cursor position.
-  Find Next: Execute a search with the same parameters as with the most recent action 'Edit' 'Find'.

244 Programming Tool

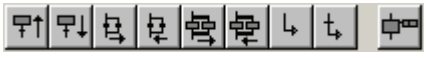
The programming tool varies from different languages. According to different kinds of languages, the detailed functions of programming tool are described as follows.

- LD Language:  ,

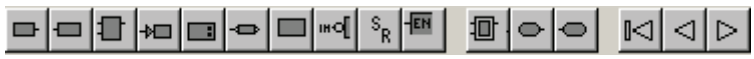
The operators are Contact, Parallel Contact, Coil, Negate, Set/Reset in turn.

- FBD Language: 

The operators are Input, Output, Box, Assign, Jump, Return, Negate, Set/Reset in turn.

- SFC Language: 

The operators are Step-Transition (before), Step-Transition (after), Alternative Branch (right), Alternative Branch (left), Parallel Branch (right), Parallel Branch (left), Jump, Transition-Jump, Use IEC-Steps in turn.

- CFC Language: 

The operators are Input, Output, Box, Jump, Lable, 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 in turn.

There are no programming tools like this in IL and ST languages. See chapter 9 for application examples.

2.5 OBJECT ORGANIZER

Object Organizer is located on the left of the main window. At the bottom there are four register cards with symbols for the four types of objects POUs, Data Types, Visualization and Resources which are the required objects for a project.

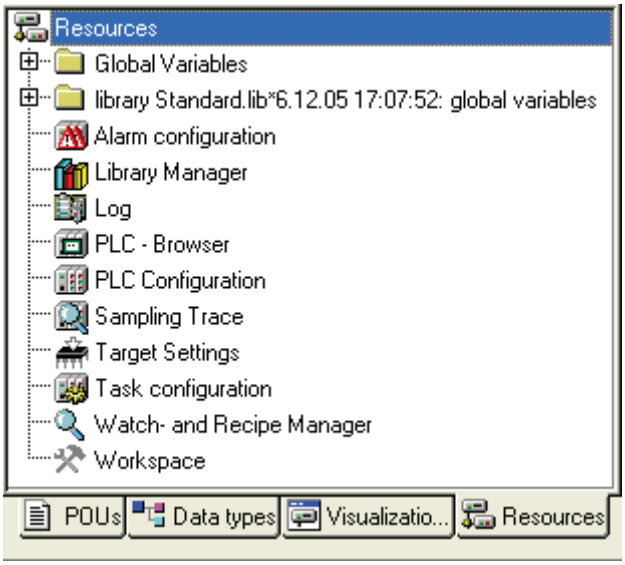


Figure 2-5-1 Object Organizer (1)

2.5.1 POUs

The function of “POUs” is to generate a POU (Program Organization Unit) in which you can write a program. Click the right mouse key in the blank space of the register card “POUs” in the object organizer, shown in figure 2-5-2.

Figure 2-5-2 POUs

- Add Object: Create a new object, select the type of POU, language of the POU and enter the name of the new POU.
- Rename Object: Give a new name to the currently-selected object.
- Edit Object: Open the editor window of a selected object and you can also do this by double click on the POUs name.
- Copy Object: With the command a selected object is copied and saved 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: Can assign access rights to the different user groups.
- Project database: PLC can't support the function.
- Add Action: Generate an action under a selected program or function block. You can enter the name of the action in the dialog which appears and also the language in which the action should be implemented. The action represents a further implementation which can be entirely created in another language as the "normal" implementation. An action belongs to a function block or program and it can be called like a function block. The syntax is: <Program_name>.<Action_name>or<Instance_naem>.< 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 regard: 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 relationgship 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 template automatically.
- Exclude from build: Active the option and then the program, function block or function will be displayed in green color and not be regared during compilation. If you want to cancle the option, select it again and the "√" in front of "Exclude from build" will be deleted and the program, function block or function changes into black color.

252 Data Types

In "Data Types", along with the standard data types the user can define his own data types. Structures, enumeration types and references can be created. For example, in "Data Types" add a object named "A" by clicking the right mouse, shown in figure 2-5-3. See section 4.6 for details.

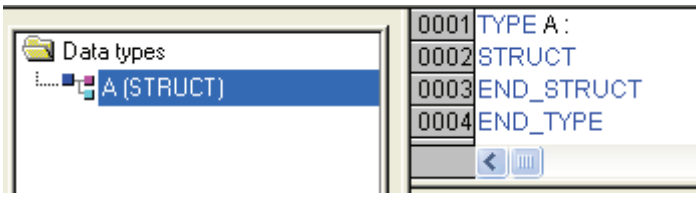


Figure 2-5-3 Data Types

253 Visualization

In "Visualization" add a visualization to display the project variables, shown in figure 2-5-4. See chapter 11 for detailed application about "Visualization".

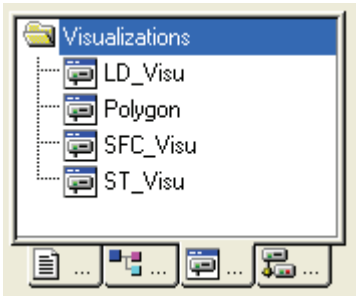


Figure 2-5-4 Visualization

2.5.4 Resources

In the "Resources" register card of the Object Organizer, there are Global Variables, PLC configuration and Workspace and so on, shown in figure 2-5-5.

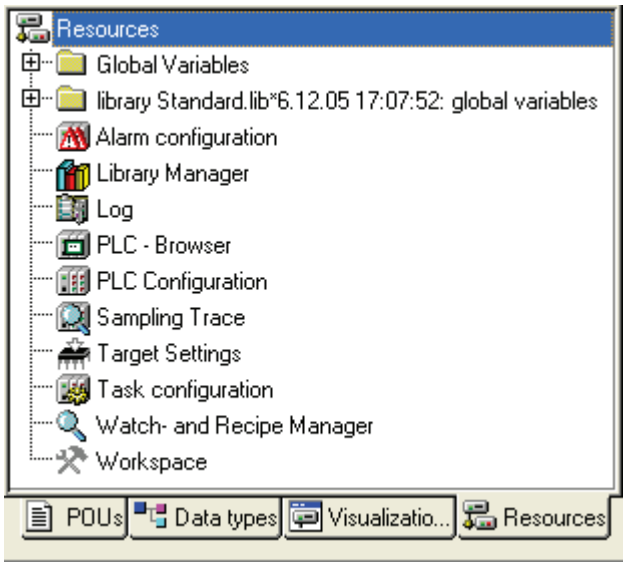


Figure 2-5-5 Resources

- "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 current project, such as Load&Save, Directories, Passwords and so on.
- "Watch- and Recipe Manager ": In "Online Mode ", monitor the variable values of different POUs in a project. See section 8.4.13 for detailed applications.
- "Library Manager": Consistent with the "Window" "Library Manager", and handle all the libraries included to the project. See section 7.4.4 for details.
- "Target Settings": Set the target platform, memory layout and so on.
- "Task configuration": Insert task and append program call.
- "Log": Record project information.

CHAPTER 3 QUICK INTRODUCTION

Take a simple program writing for an example to describe the programming mode of PowerPro so that the new learner will have a preliminary cognition for PowerPro programming and know the basic operations.It's suggested to read this chapter carefully if you are a new learner.

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

Determine the hardwre configuration before programming. For this POUs, not too much I/Os, one CPU module is ok, here we use CPU module LM3107 with 24 I/Os.

3.1 Hardware Connection

- 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

- Select CPU Module

Select LM3107 and connect power line as shown in figure 3-1-1. Regard that when the power line is connected put on the terminal corver to avoid unnecessary personal injury or equipment damage!

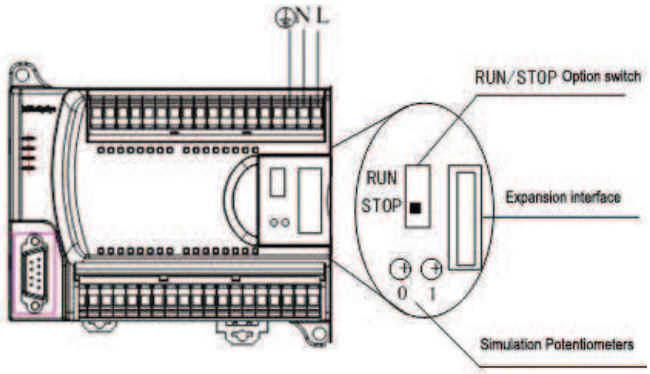


Figure 3-1-1 Connect Power Line

Not to connect the power supply when the power line is connected. After checking all the cable connections inerrably, connect the power supply and ensure the RUN or STOP indicator lights on CPU board normal so that the PLCcan run reliably.

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

Can't download to CPU module if in RUN mode until switch it to STOP position.

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

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

➢ Setup PC Communication

Through programming cable connect CPU module to PC to build a data transmition passage, shown in figure 3-1-2.

Figure 3-1-2 Connect Programming Cable

> Note:
>
> Connect communication cable before electrify, or else may damage the equipments.

3.2 STARTUP SOFTWARE

If you use PowerPro for the first time, remember to **Install Target**. It has been introduced in section 1.3.

In "Start"/"Program", click "HollySys PowerPro"/"PowerPro ENG" to startup PowerPro shown in figure 3-2-1.

Figure 3-2-1 Startup PowerPro Software

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

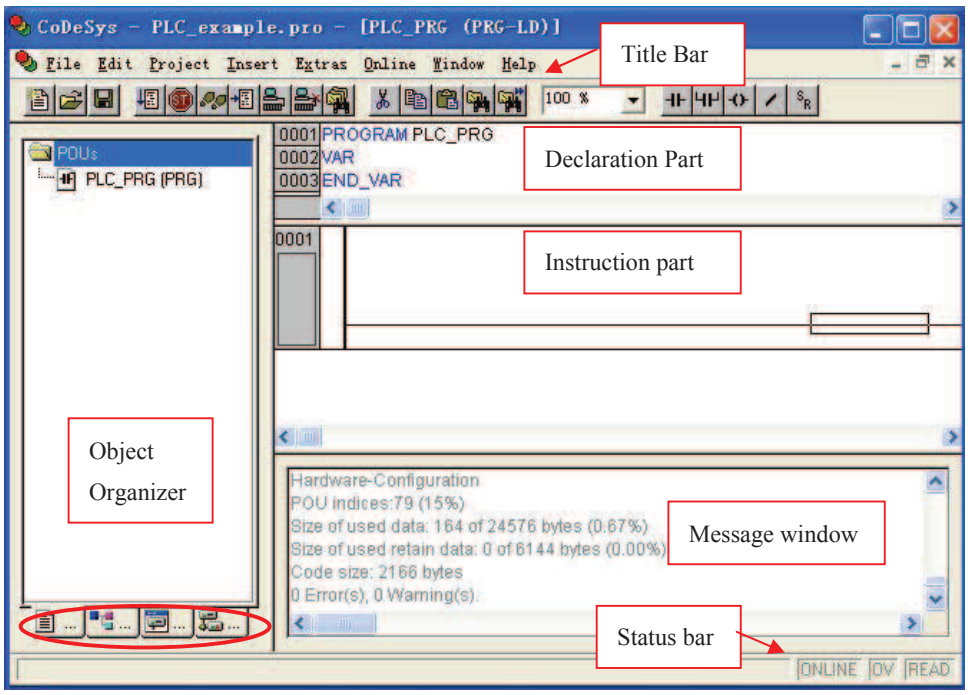


Figure 3-2-2 Interface of PowerPro

3.3 New POUs

- Target settings

Click “File”/“New” in the main window or the button“ ” in tool bar, and then a dialog box of “Target Settings” appears. “Target” means the memory space in PLC, and “Target Settings” maens to configure according to the selected memory space in PLC.

Select “PowerPro G3 CPU Extend ” in the field “Configuration” of which the memory space of CPU modules is 120KB, and then click “ok”, shown in figure 3-3-1. Select “PowerPro G3 CPU” if the memory space of the used CPU module is 28KB. If you are not sure about the size of memory space, see appendix. Select “None” if you want to new a library instruction. See section 7.4.5 for detailes.

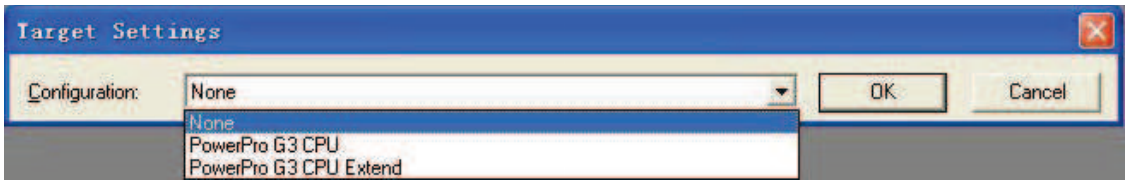


Figure 3-3-1 Target Settings

And then the window “Target Settings ” appear, the default settings can satisfy most application requirements and click “ok”, shown in figure 3-3-2.

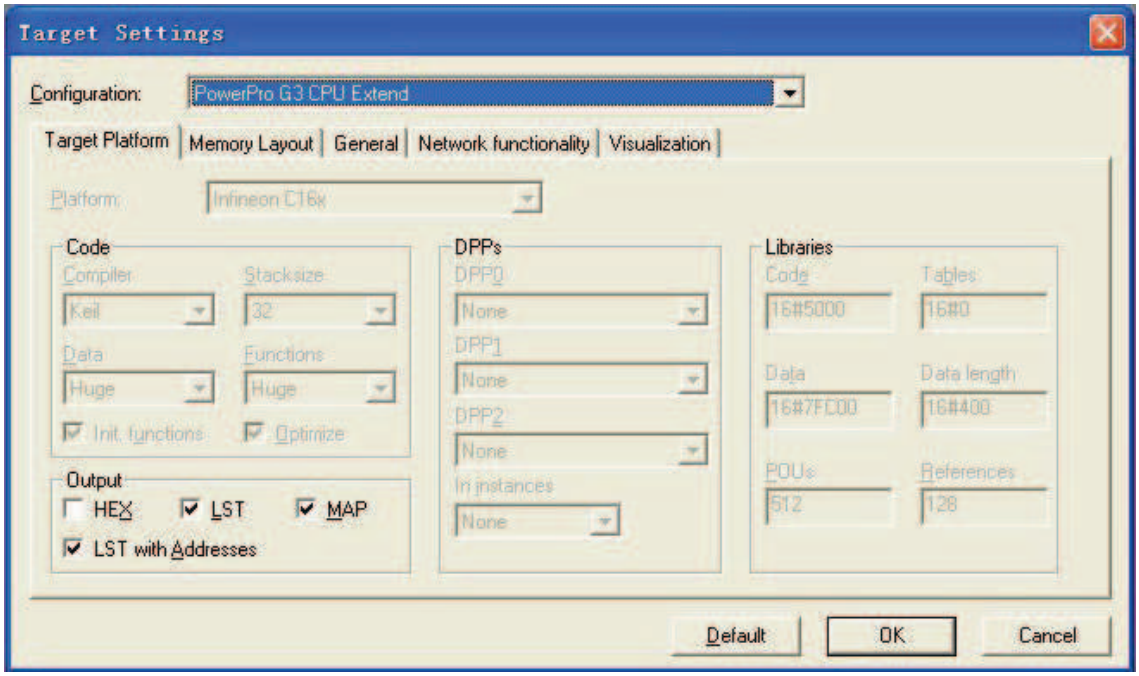


Figure 3-3-2 “Target Settings” Window

See section 7.1 about **Target Settings**.

➢ New POU

POU is program organization unit, and is the basic element for a project. See chapter 5 for details.

Select “LD” in the field “Language of the POU ”, shown in figure 3-3-3. See chapter 9 for other languages.

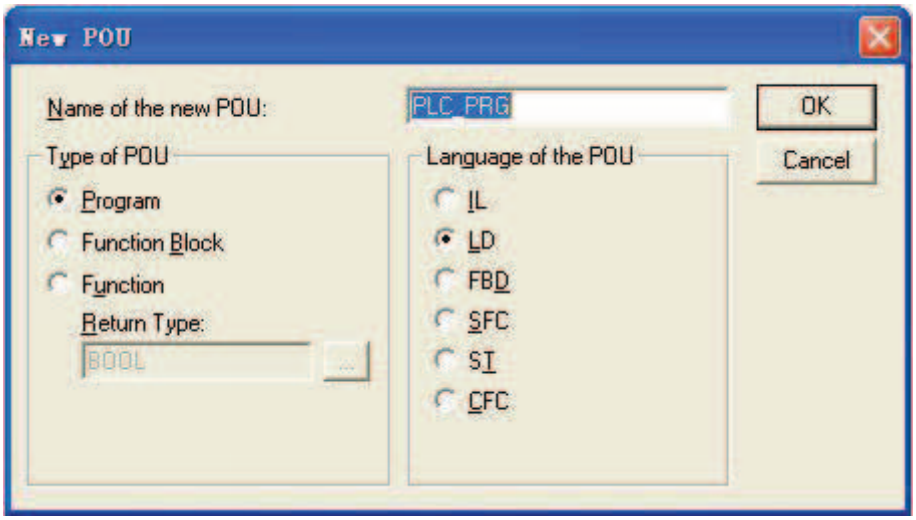


Figure 3-3-3 New POU

Select **Program** in the field “Type of POU”, PLC_PRG is the default name of the new POU. See section 5.1.3 for detailed PLC_PRG.

Click “ok” and then the window shown in figure 3-3-4 appears.

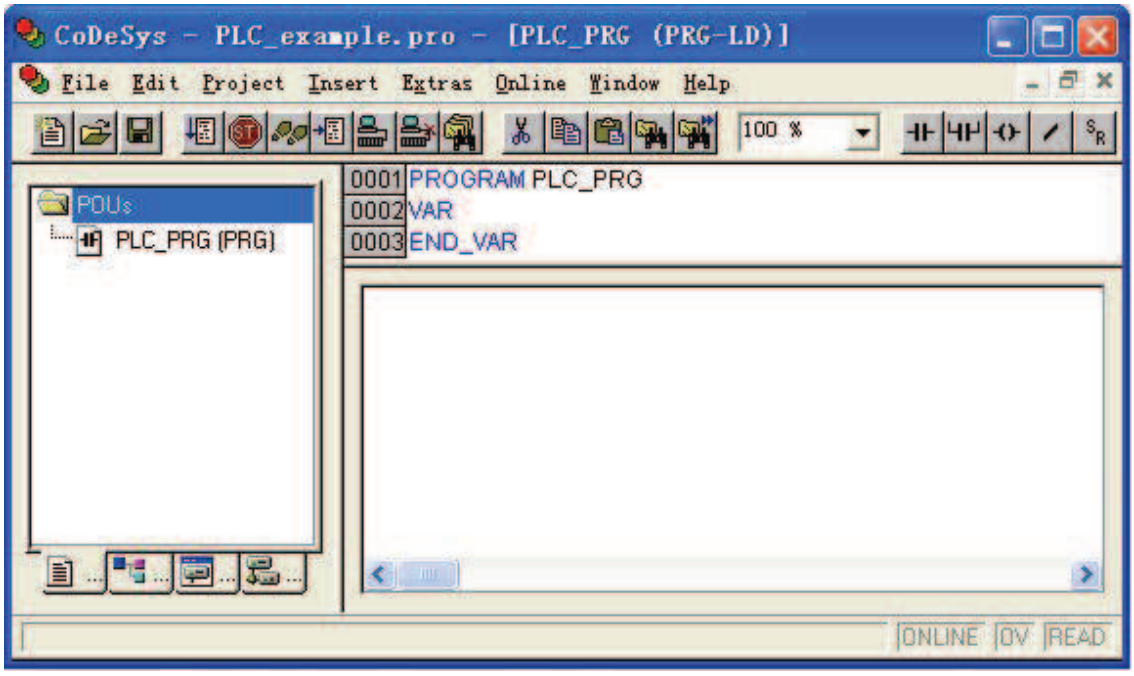


Figure 3-3-4 Workspace

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 register card "Resources", click the right mouse on "PLC Configuration", and select "LM3107" in the field "Append Subelement", shown in figure 3-4-1.

See section 7.3 for detailed description about PLC configuration.

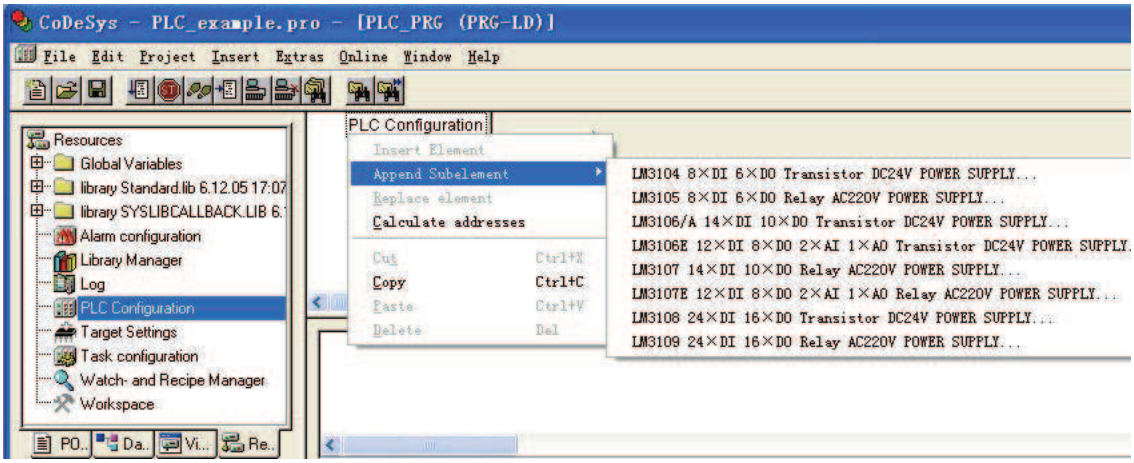


Figure 3-4-1 PLC Configuration

3.5 Set Communication Parameters

Select "Communication Parameters" in the field "Online" and a dialog box of communication parameters appears, shown in figure 3-5-1.

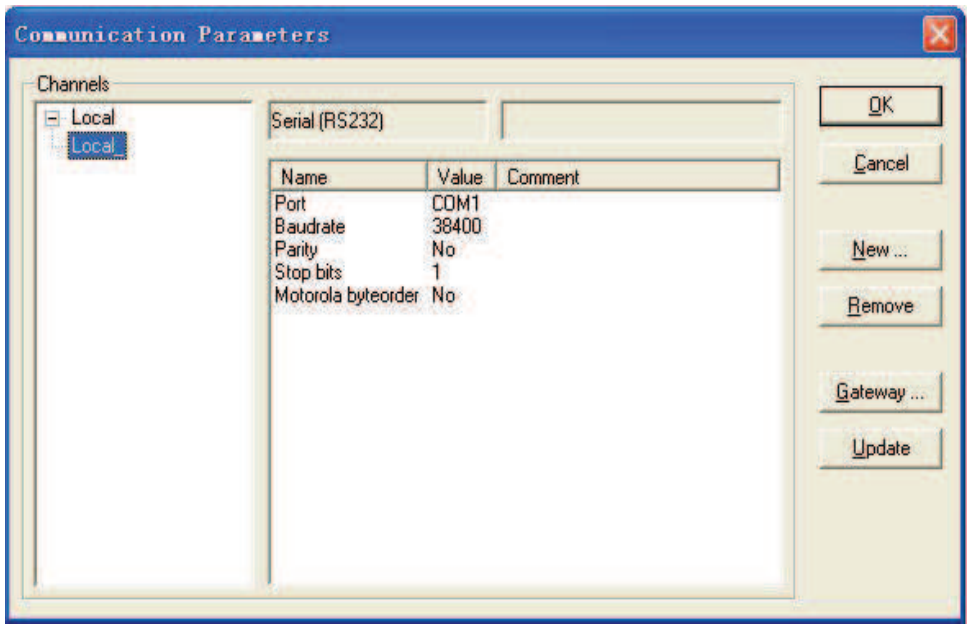


Figure 3-5-1 "Communication Parameters" Dialog

Select “New” to add a new channel and a dialog appears, shown in figure 3-5-2.

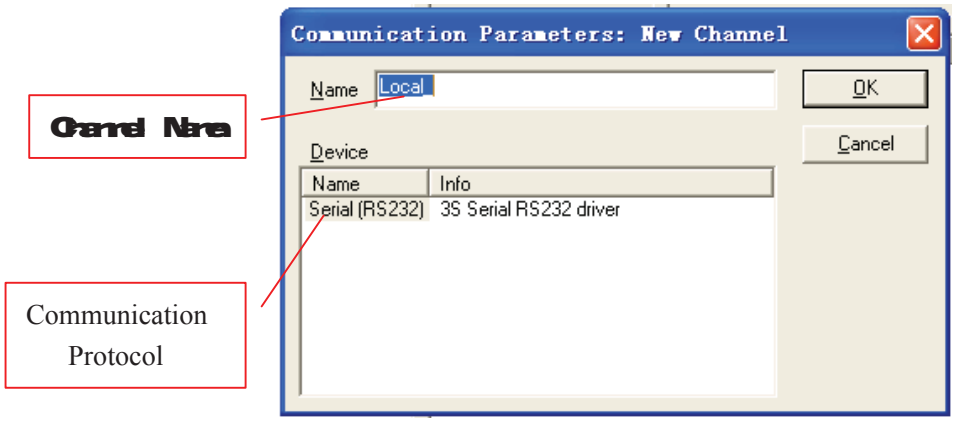


Figure 3-5-2 New Channel

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

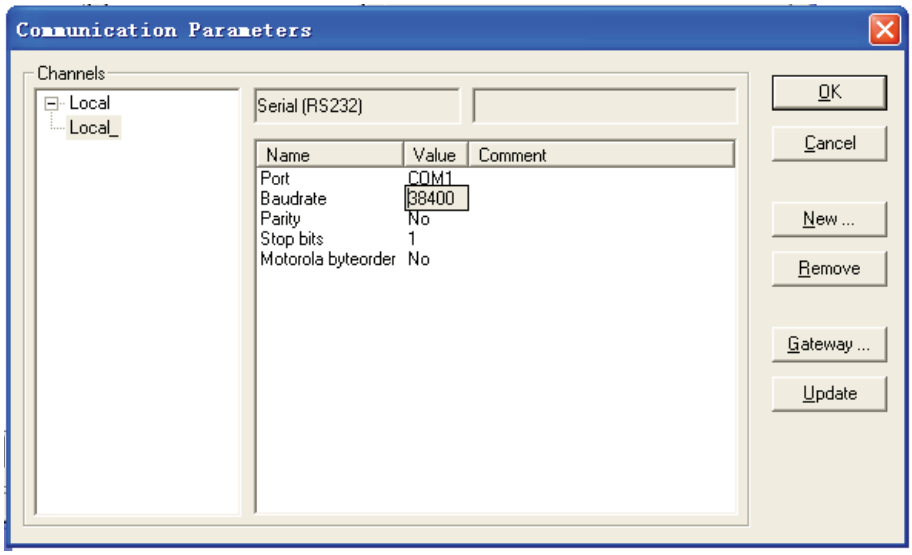


Figure 3-5-3 Setting of Baudrate

Note:

If you want to change Port or Baudrate, fast double click the left mouse in the field‘Value”.

3.6 PROGRAMMING

After PLC configuration, you can begin to write programs. The following is a simple application of timer to generate a pulse signal with 1s on and 2s off.

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

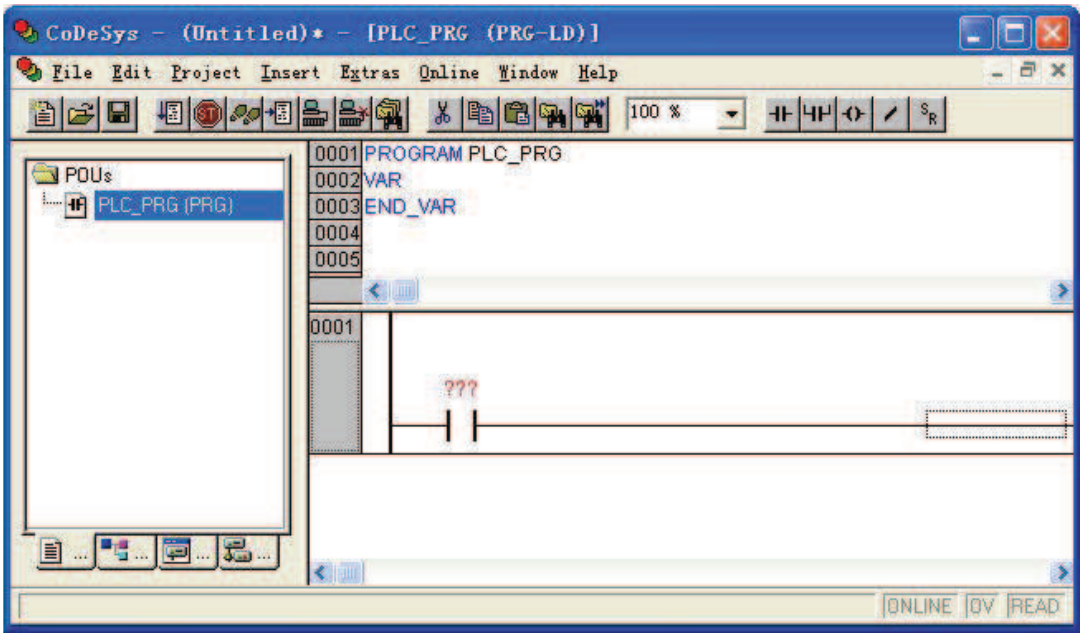


Figure 3-6-1 Programming Steps (1)

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

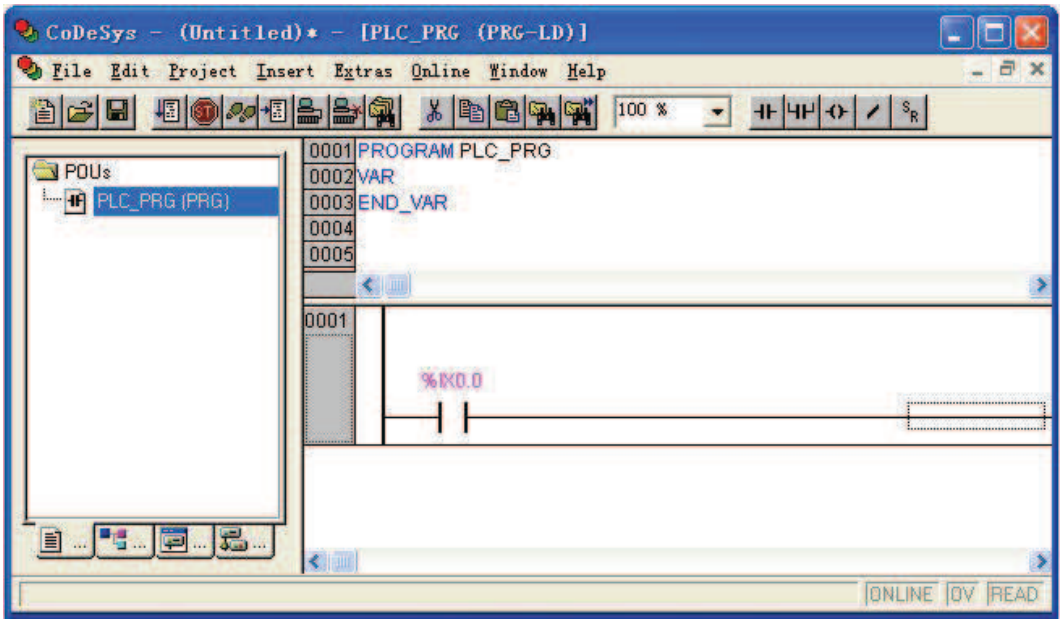


Figure 3-6-2 Programming Steps (2)

Click the right mouse after contact “%IX0.0” and select “Function Block”, shown in figure 3-6-3.

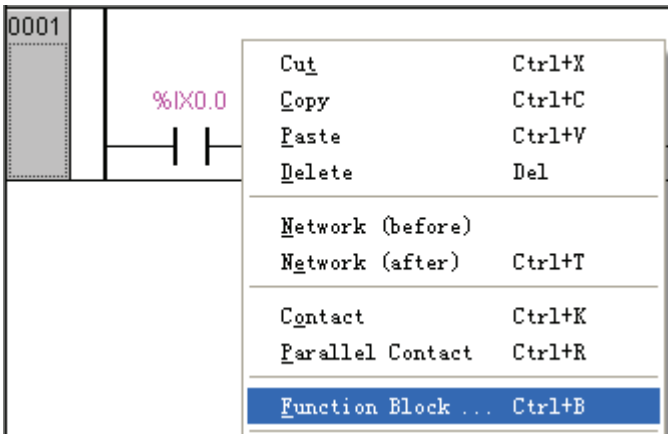


Figure 3-6-3 Programming Steps (3)

Click “Function Block”, and a dialog box appears shown in figure 3-6-4, select “TON (FB) ” which is a timer on delay.

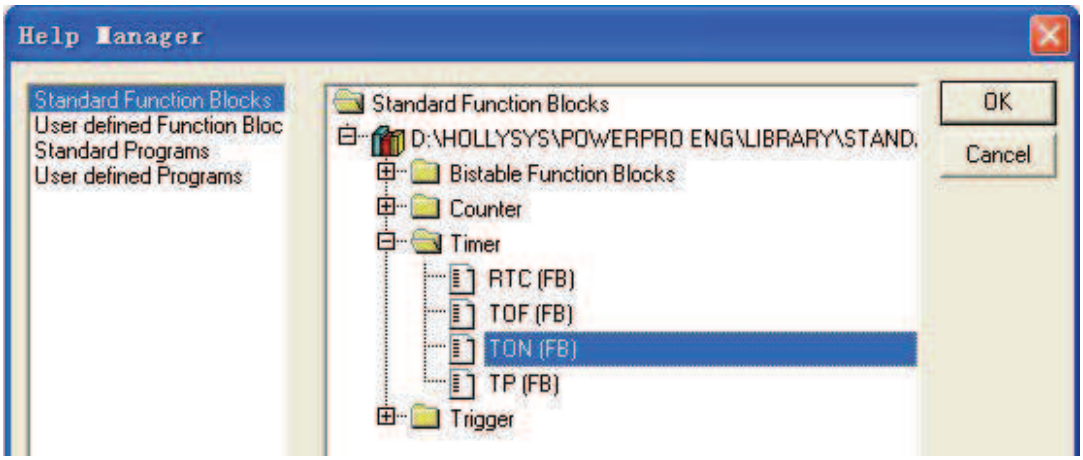
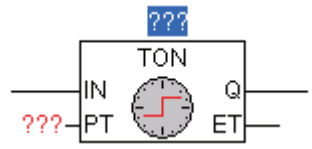


Figure 3-6-4 Programming Steps (4)

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

” and Enter, a dialog shown in figure 3-6-5 appears, select the default type TON AND CLICK “OK”. T1 is an identifier of TON. Any function block needs an identifier. See section 5.3.2 and section 7.4.3 for details.

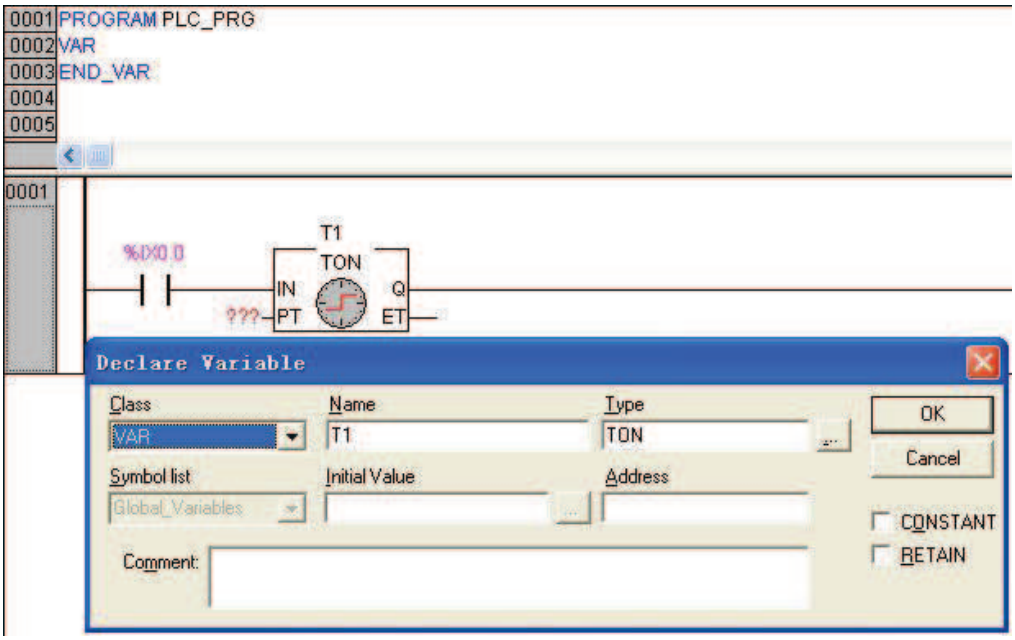


Figure 3-6-51 Programming Steps (5)

In the field "???" enter "T#1S" which stands for 1s delay. In the field "ET" enter "ET" which is a TIME variable, shown in figure 3-6-6 and click"ok".

PT is an input time parameter. Here it can be a time constant or a time variable. See section 4.3 and section 4.4 for details.

ET is the passed time, naming current time. Define a time variable in the field"ET" to observe the passed time.

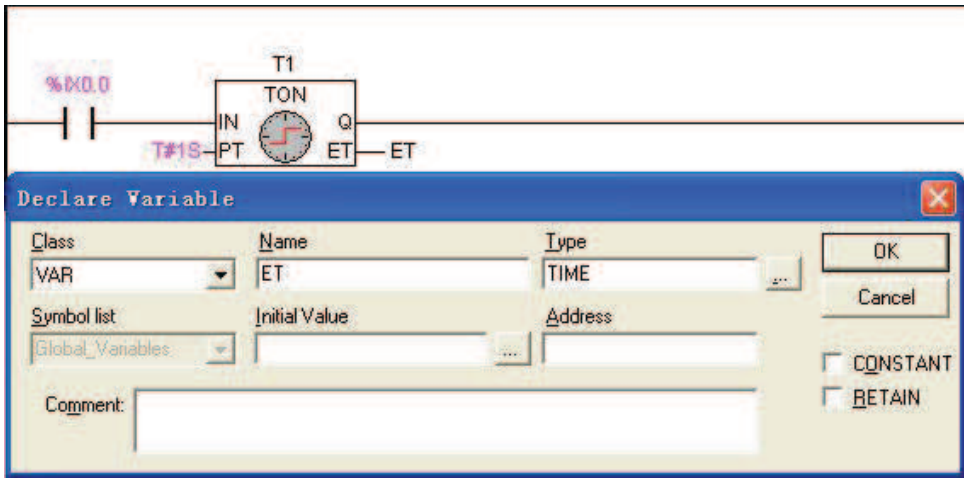


Figure 3-6-6 Programming Steps (6)

Click the coil button "  " when the cursor position is after "T1", and a coil "???" appearsatthe cursor position, shown in figure 3-6-7.

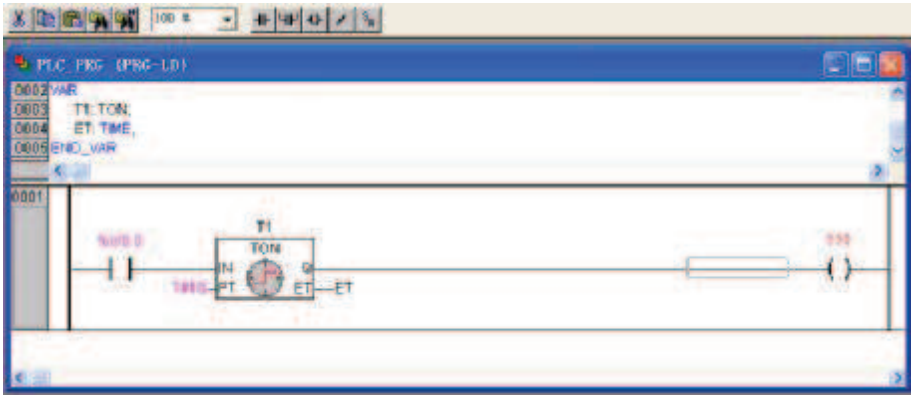


Figure 3-6-7 Programming Steps (7)

Enter “M” in “???”, and select BOOL in the field “Type”, show in figure 3-6-8, and click “ok”.

M is a middle variable, see section 4.4 for details.

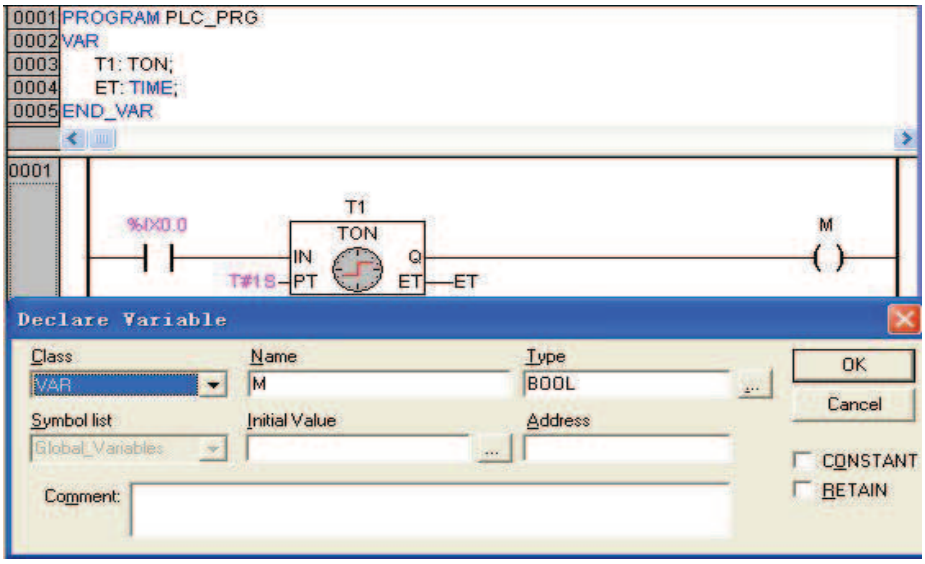


Figure 3-6-8 Programming Steps (8)

Click the right mouse in the workspace, and select“Network (after)”, shown in figure 3-6-9.

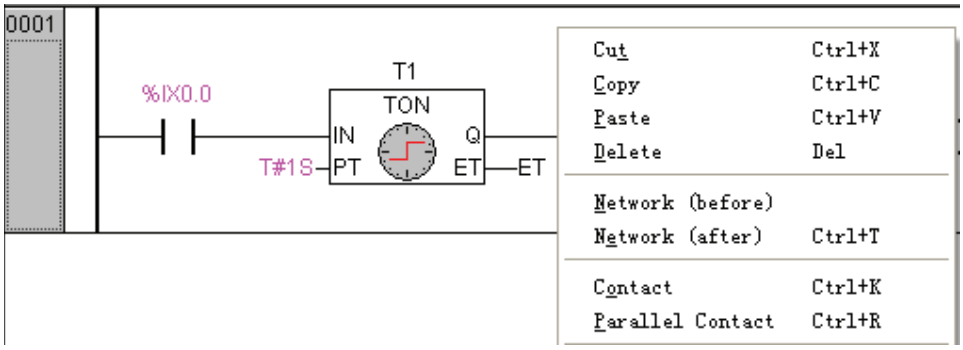


Figure 3-6-9 Programming Steps (9)

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

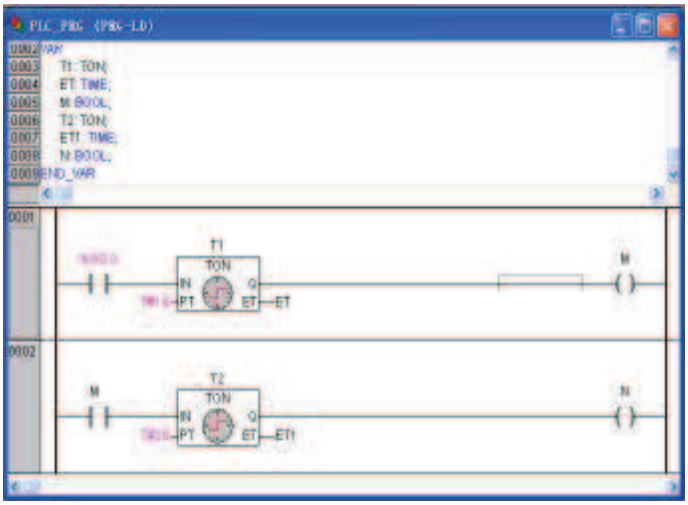


Figure 3-6-10 Programming Steps (10)

In order to achieve on-off switch, add a switch "N" before "T1". Select "T1", click the button "Contact" in tool bar or click the right mouse to select "Contact", shown in figure 3-6-11.

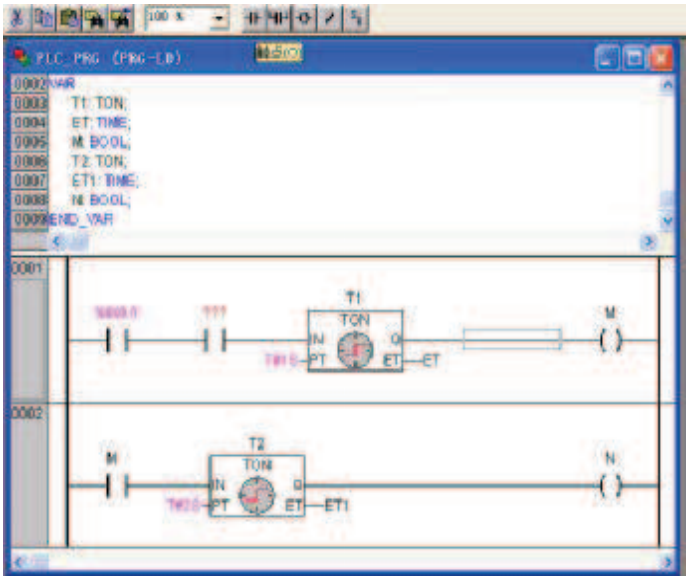


Figure 3-6-11 Programming Steps (11)

Enter "N" in the field "???", and click "Negate" after selecting ⊣ ⊢, shown in figure 3-6-12.

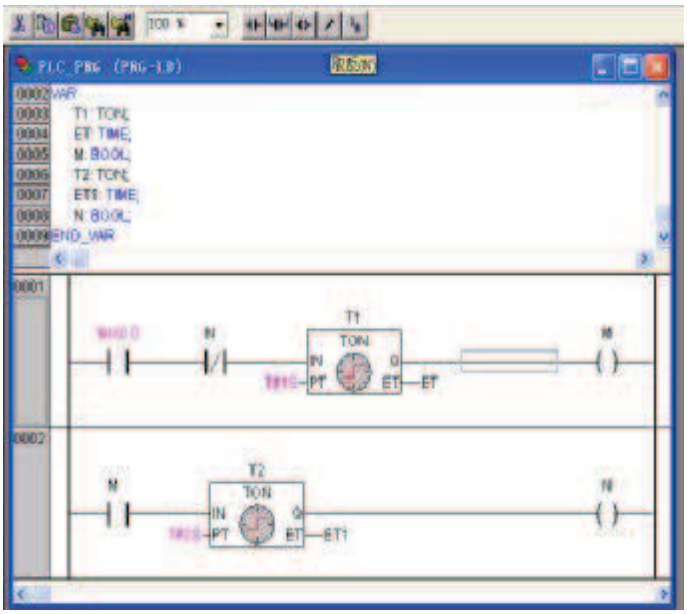


Figure 3-6-12 Programming Steps (12)

Add network 0003 to output the value of M from %QX0.0. In network 0003 add contact "M" and coil after it, and name the coil "%QX0.0", shown in figure 3-6-13.

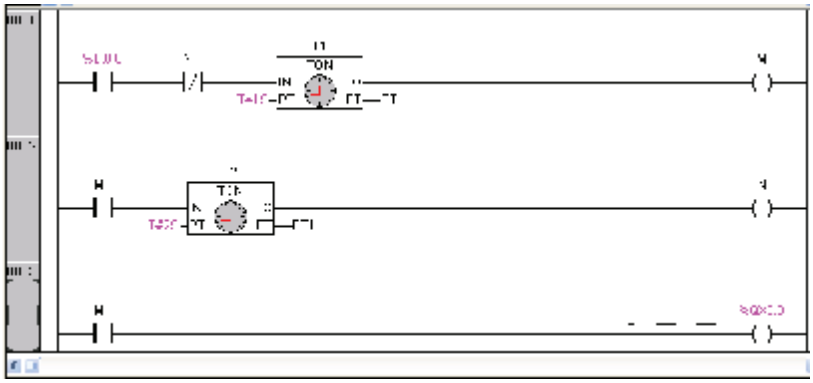


Figure 3-6-13 Programming Steps (13)

You can inset comment for better underatanding the POUs. For example in network 0001, click the right mouse and select "Comment", shown in figure 3-6-14.

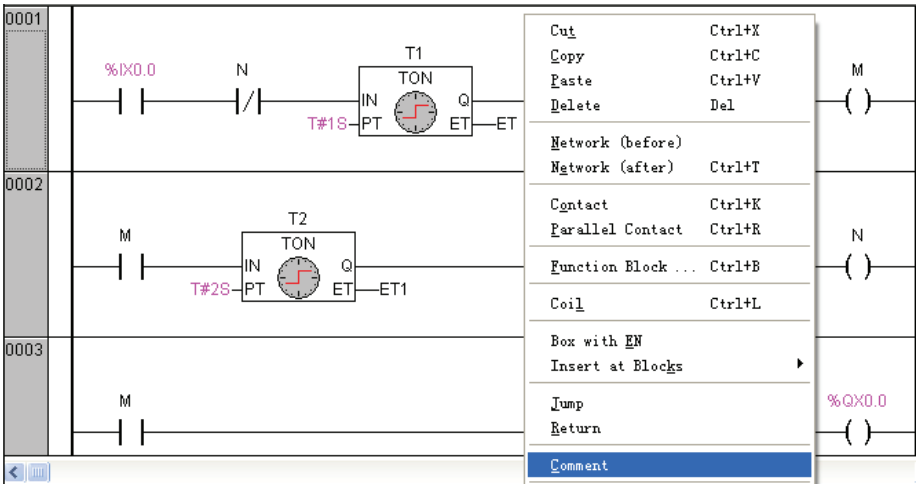


Figure 3-6-14 Programming Steps (14)

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

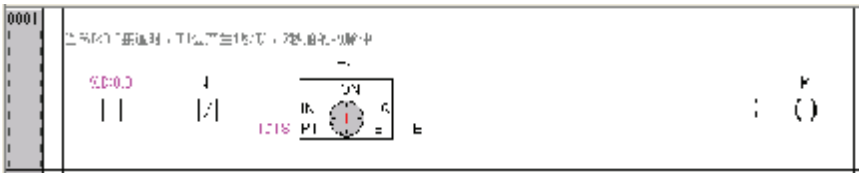


Figure 3-6-15 Programming Steps (15)

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

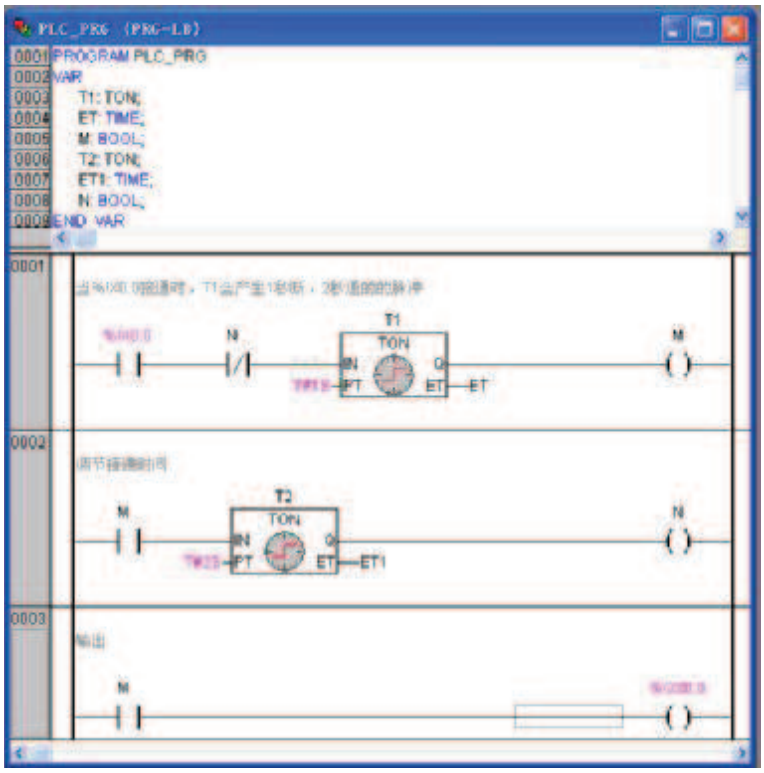


Figure 3-6-16 Programming Steps (16)

37BUILD

Build the program after the programming is finished. Click “Project” “Rebuild all” to build the program, shown in figure 3-7-1.

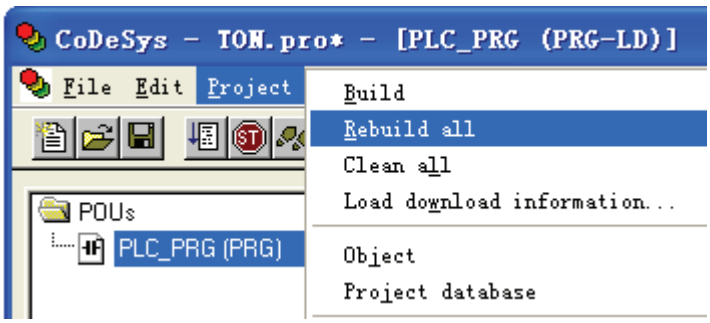


Figure 3-7-1 Build (1)

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

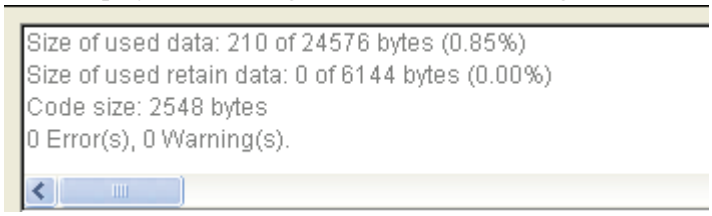


Figure 3-7-2 Build (2)

Click "Project"/ "Check"/"Unused Variables", and a message will appear in Message Window: "No unused variables found", shown in figure 3-7-3.

Figure 3-7-3 Build (3)

If there is a variable "a" not used in the program, a messafe will appear in the Message Window: "PLC_PRG(9): a", shown in figure 3-7-4.

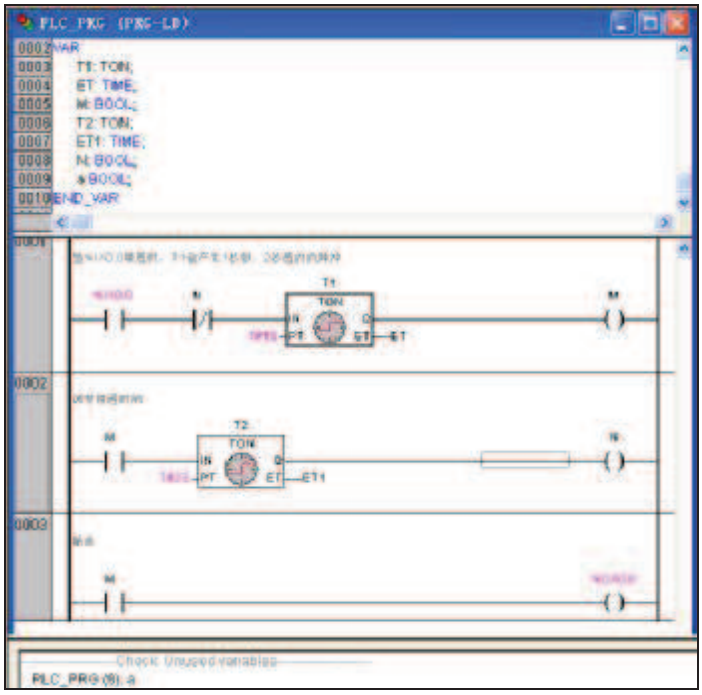


Figure 3-7-4 Build (4)

It's suggested to use the command "Check"/"Unused Variables" after building the program. You'd better delete the unused variables if there are unused variables.

In addition, the unused variables can be checked automatically by in "Resources" clicking "Workspace"/"Build"/"Check automatically",and selecting "Unused variables".

See section 8.1 for building. See section 8.2.3 for unused variables.

After the building is finished, execute "Online"/"Login". The debugging contains online debugging and simulation debugging. The program running will be introduced in the two debugging modes afterwards.

3.8 ONLINE DEBUGGING

Download the program to CPU modules is called online debugging. Download all the compiled files into CPU modules and reset the CPU modules at the same time and all the variablrs restore to initial status.

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

Figure 3-8-1 Download Message

Click the “Yes” button to download the program to CPU modules. When the message for creating boot project appears, shown in figure 3-8-2, complete the download by clicking “Yes” button to run the project after power failure and re-electrify.

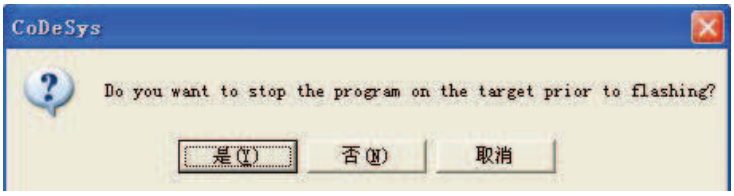


Figure 3-8-2 Message for Creating Boot Project

The program doesn ’t run after download, one need to run the program manually. Use the command “Online”/“Run” or “F5” to run a program, shown in figure 3-8-3.

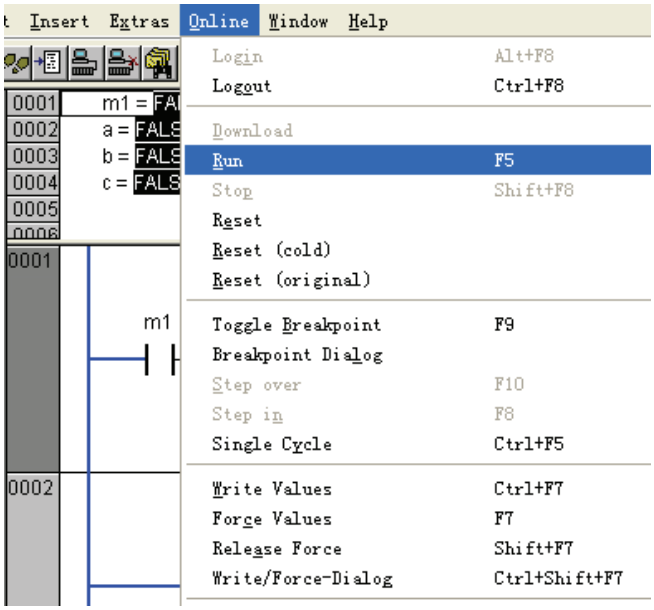


Figure 3-8-3 Run Program

Double click “%IX0.0” and press the keyword “F7” to force values, then %IX0.0 is closed and the program starts to run, shown in figure 3-8-4. From the running result we can see that the

channel %QX0.0 outputs a pulse signal of“1s off 2s on”.

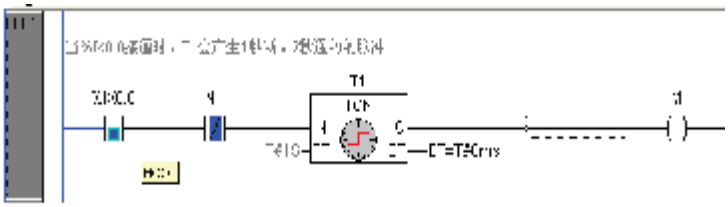


Figure 3-8-4 Program Running in Forced Values

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

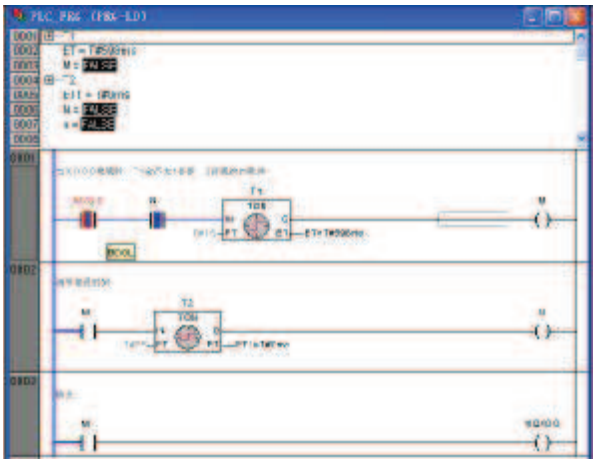


Figure 3-8-5 Process of Program Running (1)

When the running time s<t<3s, %QX0.0 is closed and the first channel light %QX0.0 on PLC is on, 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 display like “1s on and last 2s then off and on again after 1s”.

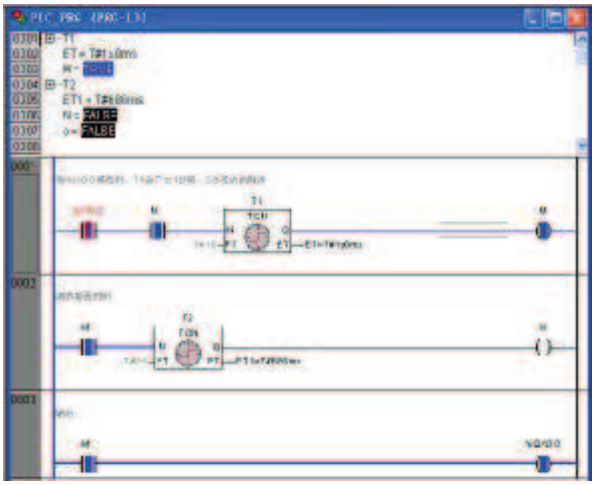


Figure 3-8-6 Process of Program Running (2)

3.9 SIMULATION MODE

The program above is performed in online mode. If there is no CPU module connected to PLC, the user program runs on the same PC under Windows, this is called simulation mode. You can enter the simulation mode by selecting "Online"/"Simulation Mode ". Select "Login" in "Simulation Mode" and the menu is shown in figure 3-9-1. If "Online"/"Simulation Mode" is selected (with a "√"), you will enter simulation mode by clicking"Login".

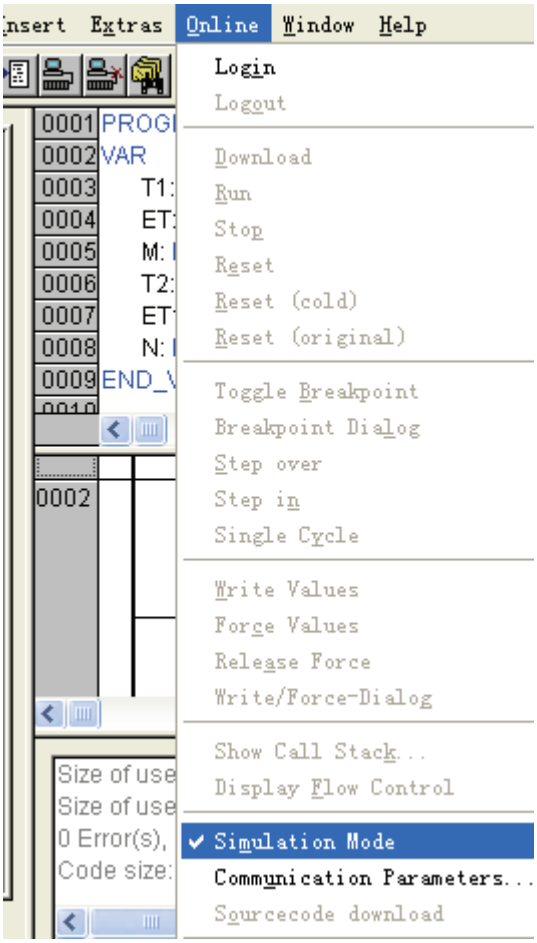


Figure 3-9-1Simulation Mode

Double click "%IX0.0", and close %IX0.0 by pressing"CTRL+F7" to write values or pressing "F7" to force values, and then press "F5" to run the program, shown in figure 3-9-2. The program running is the same with that in online mode.

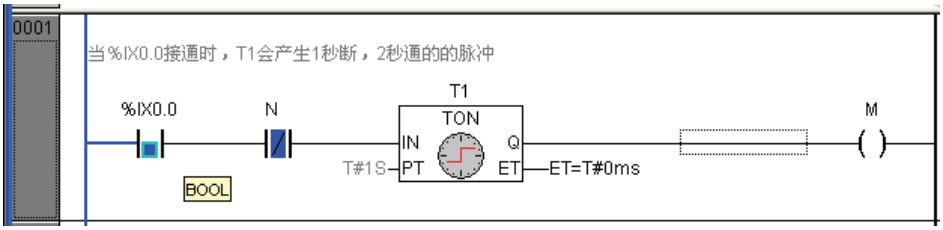


Figure 3-9-2 Program Running in Simulation Mode

See section 8.4 about online debugging and simulation debugging.

CHAPTER 4 STORAGE AREA AND VARIABLES

Before programming we need to know how to manage datas in LM. LM series PLC's memory assignment, data addressing and variable usage will be introduced in this chapter for better programming.

The data storage area in LM is classed into I (input area), Q (output area), M (memory), N and R (retain), and each has its feathers and using rules which will be introduced in section 4.1.

Addressing and storage rules are introduced in section 4.2. In LM I, Q and M are the data areas which are called by addressing.

Similar with the advanced languages, LM-series PLC also have the concepts of constant and variable. The so-called constant is the variable of which the value doesn't change. The classification and usage of constants will be introduced in section 4.3.

Variable concept is proposed in IEC61131-3 standard. When the program is running the variable value will change. Variable is used to initialize, store and handle user datas. Each variable has its own data type. The users can assign the address of I, Q or M as variable memory address. The address can also not be assigned by users but assigned by system automatically. In section 4.4 the type and usage of variables will be introduced.

LM-series PLC has powerful function for managing datas. In section 4.5 and 4.6 how to deal with array and how to declare user-defined data type will be introduced.

4.1 AllOCATE STORAGE

The storage area of LM-series PLC (CPU memory) is classed into different areas and and each has its feathers and using rules.

The classification of storage area:

Input Area (I)

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

Input storage can be accessed by addressing bit, byte, word and doubleword. Refer to section 4.2 for detailed addressing method.

The datas in input storage can be read only and has no power failure protection. In simulation mode the addresses of input area can be written or forced. But in online debugging the addresses of input area can be forced only. 'Write Values' and 'Force Values' are two ways to change the value of variavles while debugging in PowerPro and refer to section 8.4.9 and 8.4.10 for details.

Output Area (Q)

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

Output storage can be accessed by addressing bit, byte, word and doubleword. Refer to section 4.2 for detailed addressing method.

The datas in output storage can be read and written and has no power failure protection. In simulation mode or online debugging the addresses of output area can be written or forced.

M Area

M area is the PLC middle register, for the storage and management of the data or status generated in the middle process. No matter it's a bit data or word data, it can be stored in M area.

M area can be accessed by addressing bit, byte, word and doubleword. The size of M area of LM-series PLC is 8KB and the range is MB0~MB8191 in bytes. Refer to section 4.2 for detailed access method.

The data address in M area is the same with that in Q area and it can be read, written, "Write Values" or "Force Values".

Some addresses (MB300~MB799) in M area have the power-failure protection function. The other addresses don't have the power-failure protection function.

In addition, regard that the former 100 bytes in M area i.e. MB0~MB99 are used for diagnostic area. The data in these addresses can be read but can't be written. It's suggested to use the address starting from MB100 for users in programming.

N Area

N area is also the PLC middle register, for the storage and management of the data or status generated in the middle process. The difference between the M and N is that the N area can be accessed and called only by variable mode. The variables have been introduced in the former chapters and refer to section 4.4 for detailed variable usage.

The variable addreses in N area are assigned by system automatically and can't be assigned by users.The data type of variables stored in N area includes bit, byte, word, doubleword, REAL, TIME, INT and other data types. In addition, besides the data variable, the function block variables are also stored in N area. About the concept of function block refer to section 5.1.

The size of N area is 24KB, and that means 24KB variables can be stored in N area. The variable stored in N can be read and written, written values and forced values. The datas in N area can't retain its last value after power-failure.

R Area

R area belongs to power-failure protection area and its calling mode is the same with N area which can be accessed by variable mode and be assigned by system automatically and can't be assigned by users.

The size of R storage area is 6KB. The variable stored in R can be read and written, written values and forced values.

At the declaration of variable, if the option "RETAIN" is not activated or one declare it directly between the keywords VAR and END_VAR then the variable will be stored in N area, if the option "RETAIN" is activated or one declare it directly between the keywords VAR RETAIN and END_VAR then the variable will be stored in R area and can retain its last value after power-failure. Refer to section 4.4.6 about declaration of RETAIN variables.

Note:

In LM series PLC there are two ways for power-failure protection: first, address mode, select the addresses MB300~MB799; second, variable mode, declare a variable as a RETAIN type

4.2 ADDRESSING OF ADDRESS

4.2.1 Address Storage Mapping Relationship

The I area, Q area and M area in LM series PLC are accessed by addressing of address and each of the storage areas have unique and definite address. The users can read and set the value in this storage by addressing of address i.e. by using the storage address directly.

Before introducing the accessing rules you need to know the storage format. The storage format of I, Q and M is the same, here take M for an example, shown in figure 4-2-1.

All the direct addressing storage areas in LM series PLC are stored in bytes.

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

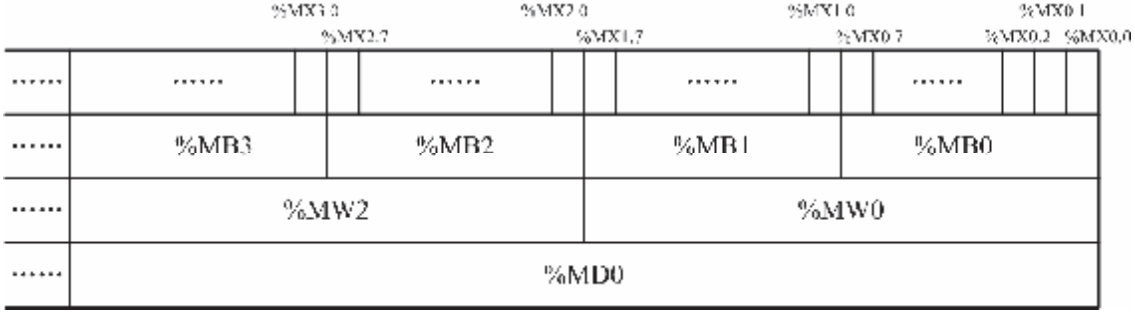


Figure 4-2-1 Storage Format in M

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

%MW12 is composed of %MB12 and %MB13.

Bit, byte, word or doubleword 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. Use variable mode to access INT, REAL or other data types. See 4.4 for details.

Regard that the data storage area may be overlapped when accessing in different data types. For example, if the value stored in %MW0 is 3, then the value in %MB0 is 3, the value in %MX0.0 and %MX0.1 is TRUE. One more example, if the forced value in %MX0.0 is TRUE, then the values in %MB0, %MW0 and %MD0 are forced 1 because %MX0.0 is the first bit of these storage area.

It's the same for I and Q. Refer to section 7.3 for the corresponding relation between the addresses in I, Q and the actual DI/DOs.

Note:

Because a word is composed of two bytes, so the number must be even in word addressing. For example, %MW0 is composed of %MB0 and %MB1 so the next word is %MW2 not %MW1. %MW1 is invalid. Doubleword addressing abides by the rule too.

422 Addressing Format

According to IEC61131-3 standrad, all the direct addresses start with "%". Take M for an example, shown in table 4-2-1.

Table 4-2-1 Addressing Format

Bit addressing	Format	%MXm.n
	Description	X: bit addressing; m: byte number in momory location; 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 momory location
	Data type	BYTE
	Example	%MB103, %MB2000
Word addressing	Format	%MWm
	Description	W: word addressing m: first byte number of the word in momory location, and m must be even
	Data type	WORD
	Example	%MW150, %MW3000
Doubleword addressing	Format	%MDm
	Description	D doubleword addressing m: first byte number of the doubleword in momory location, and m must be even
	Data type	DWORD
	Example	%MD300, %MD432

For I and Q, replace M in the above table by I or Q.

The range of I, Q and M is displayed in table 4-2-2 and the address which is over the range is invalid.

Table 4-2-2 Data Memory Area and Range

Memory area	Range (bytes)
I (Input)	%IB0~%IB511 (maximum 512 bytes, the size depends on PLC)
Q (Output)	%QB0~%QB511 (maximum 512 bytes, the size depends on PLC)
M (Memory location)	%MB0~%MB8191

Regard that the size of M is 8KB with the range %MB0~%MB8191. Where, %MB0~%MB99 are the diagnostic addresses and it 's suggested not to be used by users. %MB300~%MB799 have power failure protection function and the others don't have the function.

4.3 Constants

In PLC programming some parameters of which the values don 't change, such as time constant of timer and proportion parameter are called constant. LM series PLC supports a large number of constants in different data types and the common used constants include BOOL, TIME and numerial, shown in table 4-3-1.

Table 4-3-1 Class and Representation Method of Constants

Type of Constants	Representation Method	
BOOL	Description	BOOL type variables may be given the values TRUE (1) and FALSE (0).
	Example	TRUE, 0
Integer	Description	Number values can appear as binary numbers, octal numbers, decimal numbers and hexadecimal numbers. If an integer value is not a decimal number, you must write its base followed by the number sign (#) in front of the integer constant. The values for the numbers 10-15 in hexadecimal numbers will be represented as always by the letters A-F.
	Example	14 (*decimal number 14*) 2#1001_0011 (*binary number 1001_0011*) 8#67 (*octal number 67*) 16#AE (*hexadecimal number AE*)
REAL	Description	REAL constants can be given as decimal fractions and represented exponentially. Use the standard American format with the decimal point to do this.
	Example	7.4 (*REAL constant 7.4*) 1.64e+009 (*REAL constant 1.64e+009*)
TIME	Description	TIME constants are generally used to operate the timer. A TIME constant is always made up of a"t#" (or "T#") and the actual time declaration which can include days (identified by"d"), hours (identified by"h"), minutes (identified by"m"), seconds (identified by"s") and milliseconds (identified by"ms"). Please note that the time entries must be given in this order according to length (d

		before h before m before s before m before ms).
	Example	T#18ms (*18ms*) T#100s12ms (*100s12ms, the highest component may be allowed to exceed its limit*) t#12h34m15s (*12h34m15s*) the following would be incorrect: t#5m68s (*limit exceeded in a lower component*) 15ms (*T# is missing*) t#4ms13d (*incorrect order of entries*)
TIME_OF_DAY	Description	Use the type of constant to store times of the day. ATIME_OF_DAY constant is composed of "TOD#" ("tod#", "TIME_OF_DAY#" or "time_of_day#") and "a time" with the format: hour: minute: second (you can enter seconds as real numbers).
	Example	TOD#00:00:00 (*0h0m0s*) TIME_OF_DAY#15:36:30.123 (*15h36m30.123s*)
DATE	Description	A DATA constant is composed of "D#" ("d#", "DATE#" or "date#") and "a date" with 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 the time of day can also be combined to form so-called DATE_AND_TIMEconstants. A DATE_AND_TIME 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 preceded and followed by single quotation marks. You may also enter blank spaces and special characters.
	Example	'Abby and Craig' (*string Abby and Craig *) ':-)' (*string :-)*)

Note:

There is no case sensitive in PowerPro, T#3s and t#3s are the same constant, TRUE and true can both represent a Bool constant.

4.4 Variables

The concept of variable is proposed by IEC61131-3. The variable is an identifier composed of

characters, numbers and underlines. Before using the identifier must be declared to describe the object. Each identifier corresponds to a data type.

In LM series PLC, the declared identifier can be used to denote some time-changing parameters or some function block instructions. In this chapter how to identify time-changing parameters i.e. the variables in general meaning is introduced. About function block identifiers see section 5.3.2.

Variables can be used in programs instead of memory addressing mode. The identifier can be declared based on its actual meaning for better program readability. You had better to know the concrete classification before using. The following is the concrete classification at a different angle.

According to different data types, variables can be classed into standard data types and user-defined data types, where standard data types contain BOOL, INT, REAL, STRING, TIME and so on.

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

According to the property, variables can be classed into intermediate variable, input variable, output, input/output variable and so on.

According to power-off protection or not, variables can be classed into retain variable and non-retain variable.

In section 4.4.2 all the data types are described. In section 4.4.4 the declaration and usage of global variable and local variable are described. In section 4.4.5 the declaration and usage of input variable, output variable and input-output variable are described. In section 4.4.6 how to declare a retain variable is described.

4.4.1 Naming Rules of Variables

Naming of variables must follow the rules:

- The name must start with a character or an underline, following by a number of characters, numbers or underlines.
- Capitalization is not recognized which means that ABC and abc are the same variable.
- The variable identifier cannot be the same as any of the keywords. The keywords are standard indentifiers of which the usage and naming have been defined automatically. The keywords in PowerPro are shown in table 4-4-1.

Table 4-4-1 List of Keywords

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

442 Data Types of Variables

Data types of variables in PowerPro have standard data types and user-defined data types. See section 4.6 for user-defined data types. Here standard data types are introduced.

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

Standard data types and user-defined data types

Table4-4-2 Data types of Variables

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’

443 Variables Declaration

The variable must be declared before using. PowerPro provides different variable types for

different variable functions. At the declaration of a variable, you have to declare not only the data type but also the variable type.

In PowerPro, variables can be defined into different types, such as global variables, local variables, input variables, output variables and so on. See table 4-4-3 for details.

Table 4-4-3 Types of Variables

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 variavle is used to transfer parameter when calling a POU. The parameter can be transferred to subprogram or other POU through input variable. See 5.3 for details.
VAR_OUTPUT	Output variable is used to transfer parameter when calling a POU. The parameter can be transferred to the POU which is calling the current POU through output variable. See 5.3 for details.
VAR_IN_OUTPUT	Input/output variable is the combination of VAR_INPUT and VAR_OUTPUT and is used to transfer parameter.
VAR_GLOBAL	Global variable can be used in any program of the current project. At the same time the variable with the same name can't be declared.

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

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

Autodeclaration

If the **Autodeclaration** option has been chosen in the Editor category of the Options dialog box , then a dialog box will appear in all editors after the input of a variable that has not yet been declared, shown in figure 4-4-1. The **Class**, **Name** and **Type** are necessary.

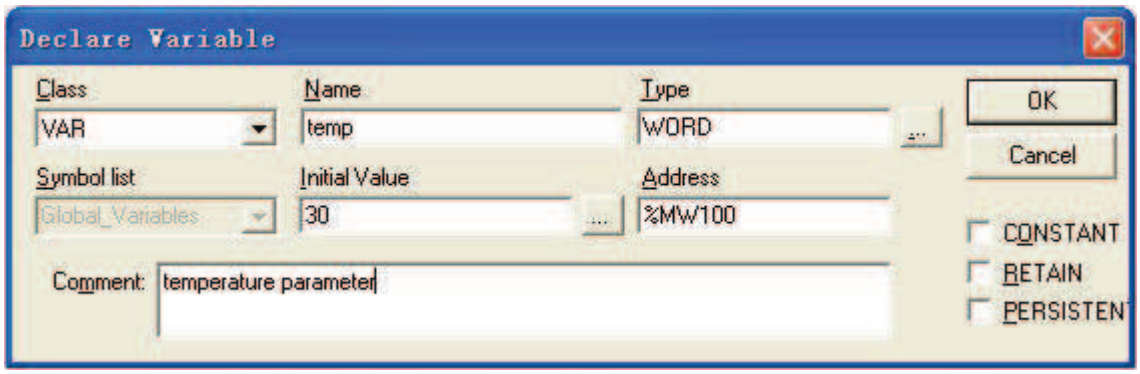


Figure 4-4-1 Autodeclaration

The options as follows:

- Class: five classes, see table 4-4-3 for the differences.
- Name: variable name ie identifier, see section 4.4.1 for variable naming rules.
- Type: selection of data type, one can enter data type in **Type** field or click the ...

button and select from all the possible data types in the **Input Assistant** dialog. Refer to section 4.4.2 about the data types.

- Symbol list: the symbol list is available only when selecting "VAR_GLOBAL" in "Type" field. The default setting is "Global_Variables". After a declaration of a global variable, you will find a "Global_Variables" in the "Resources" register card in the "Global Variables" folder and all the declared global variables will be displayed here by double clicking the "Global_Variables", shown in figure 4-4-2.
- Initial value: the initial value of the variable being declared, one can enter a constant corresponding to variable data type as the initial value.
- Address: In the **Address** field, you can bind the variable being declared to an address (AT declaration).
- Comment: comment for variable.

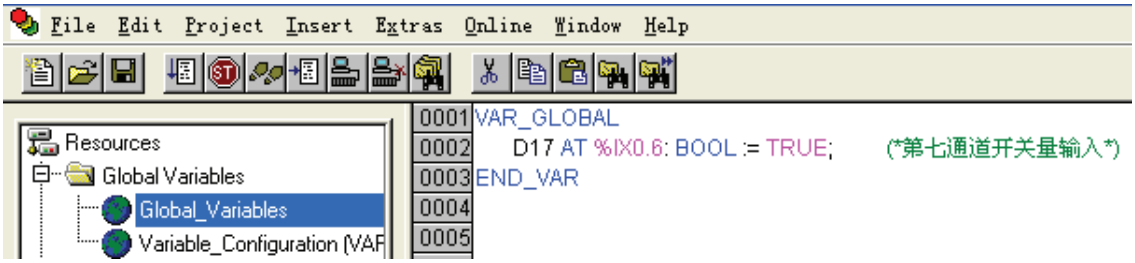


Figure 4-4-2 Code of variable declaration

At the autodeclaration of a variable pay attention to:

- A variable can be specified a address and the address format must be consistent with that described in 4.2. When a variable is specified an address then the variable will be stored in the data area appointed by the address. For example the variable temp declared in figure 4-4-1 will be stored in %MW100. In a program change the value of %MW100 through direct addressing then the value of variable temp will be changed. The variable which is not specified an address will be stored in N location.
- At the declaration of a variable, it can be initialized with a constant of which the type must be consistent with the variable type. For example the initial value of a TIME variable must be a TIME constant like t#5s. Then the variable will be set as the initial value at the moment of power-on.
- After autodeclaration the variable declaration will be displayed in declaration part automatically. If the variable temp in figure 4-4-1 has been declared then the following will display in declaration part:

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

If a global variable has been declared then it will be displayed in **Global Variables** in Resources tab.

- At autodeclaration there are two options at the lower right corner of the dialog box: CONSTANT and RETAIN. If the CONSTANT is activated then the variable will be saved as a constant and the value can 't be changed. If the RETAIN is activated then the

variable will be stored in R and can retain its value after power-failure.

- When new a variable it can be autodeclared. But when the variable is deleted, the declaration in declaration part will not be deleted automatically and it 's still in editor, so regard that the variable can 't be declared repeatedly. With the command "Project"/"Check"/"Unused Variables" to check the used variables. See section 8.2.3 for detailed usage.

Manual declaration

Besides autodeclaration a variable can be manually declared in PowerPro. The so-called manual declaration is to add instructions in declaration part directly without the help of the dialog box of "declare variable". A variable declaration has the following syntax:

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

The parts in the braces { } are optional.

Declare variables of different types in different positions, for example, decalare local variables between the keywords VAR and END_VAR and declare input variables between the keywords VAR INPUT and END_VAR.

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

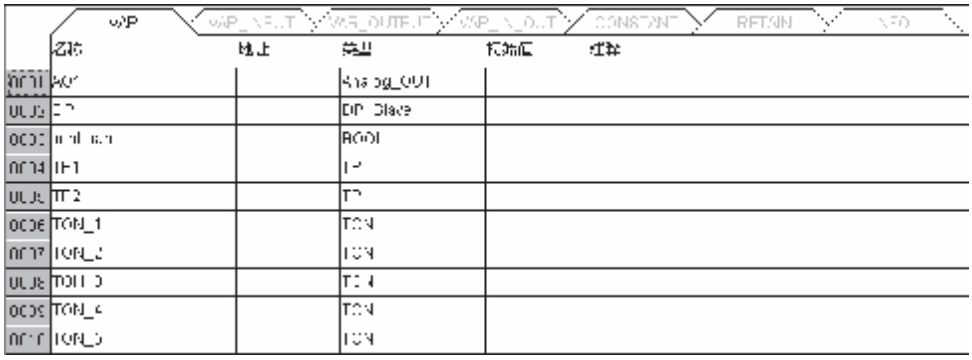


Figure 4-4-3 Declarations as Tables

With the command "Insert"/"New Declaration" you bring a new variable into the declaration table of the declaration editor. You will receive a variable that has "Name" located in the Name field, and "Bool" located in the Type field, as its default setting. You should change these values to the desired values. Name and type are all that is necessary for a complete declaration of variables.

Switch to normal by deactivating "Declaration in table form" in the context menu.

Declaration in table form is more succinct compared with declaration in text form, so it 's suggested to use autodeclaration or declaration in table form for new learners.

The difference between variable calling and addressing calling

The "variable+address" calling is similar with address calling, but there is still difference. The data type that can be called through address can be BOOL, BYTE, WORD and DWORD while the number of data type that can be called through "variable+address" is larger.

For example: Declare a REAL variable stored in %MD100. If you use address %MD100 only then the data type will be DWORD not REAL. Now you should declare a REAL variable stored in %MD100 in "variable+address" mode (enter the initial address and the length of the data type will

be calculated automatically).

4.4.4 Global/Local Variables

A complicated project can be composed of a number of POUs (Program, Function Block). Declare a global variable which can be recognized throughout the project and declare a local variable which can be only used in the current program or function block.

At the declaration of a variable, with the help of the **Class** combobox one can define a global variable (VAR_GLOBAL) or a local variable (VAR).

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

4.4.5 Input/Output Variables

In programming subprogram may be called to transfer the parameter in main program to subprogram and transfer the calculating result to main program, now input variable and output variable are needed. The input variable and output variable are declared in subprogram. Input variable is a variable that transfer from main program to subprogram and 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.

At the declaration of a variable, with the help of the **Class** combobox one can define an input variable (VAR_INPUT), output variable (VAR_OUTPUT) or input/output variable (VAR_IN_OUTPUT).

See section 5.3 about detailed POU calls.

4.4.6 RETAIN Variables

In many projects the variables which maintain their value even after an uncontrolled shutdownof the controller are needed. At the declaration of a variable, one can define a Retain variable directly saved in R memory area which has the power-off protection function.

At the declaration of a variable, select "RETAIN" at the lower right corner of the dialog box, then the variable will be saved as a Retain variable. You can also define a Retain variable manually between the keywords VAR_RETAIN and END_VAR.

It has been introduced in section 4.1 that partial address in M (%MB300~%MB799) has the power-failure protection function. A variable declared as a RETAIN variable or stored in %MB300~%MB799 has the same effort and has power-failure protection function.

4.4.7 Pointer Variables

Pointer variable is a special kind of variable. In LM series PLC all memory areas occupy CPU address which is called absolute address. No matter it 's I, Q, M or N, R it will occupy partial addresses of CPU. The address in I, Q and M which is accessed by addressing, such as %MW100, can be considered as relative address. Pointer is an absolute address.

If two relative addresses are adjacent then their absolute addresses are adjacent too. Because

the storage area in LM series PLC is stored in bytes, so if the absolute address of %MW100 is pt, then the absolute address of %MW102 is pt+2 and the absolute address of %MW200 is pt+100.

In programming this pointer feature can be used to realize more complicated functions. Remember to declare a pointer variable before using.

The declaration of Pointer variable is like other data types, and the difference is that the data type is POINTER TO <data type>.

Pointer declarations have 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*)
```

Example of usage of pointer:

Move 100 WORD variables stored in M stating from %MW100 to the addresses starting from %MW300 to enable the variables have power-failure protection function.

Because the data area is large so it's 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 to write programs, see section 9.3 about ST.

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 functions for managing datas in PowerPro is very powerful, both datas of different types and multi-dimensional datas can be spported. One-, two-, and three-dimensional arrays are supported as elementary data types. Arrays can be defined both in the declaration part of a POU and in the global variable lists.

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 lower limit of the field range, U1, U2 and U3 the upper limit. The limit values must be integers. Configure L1, U1 for one-dimensional array, L1, U1, L2, U2 for two-dimensional array, and L1, U1, L2, U2, L3, U3 for three-dimensional array.

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

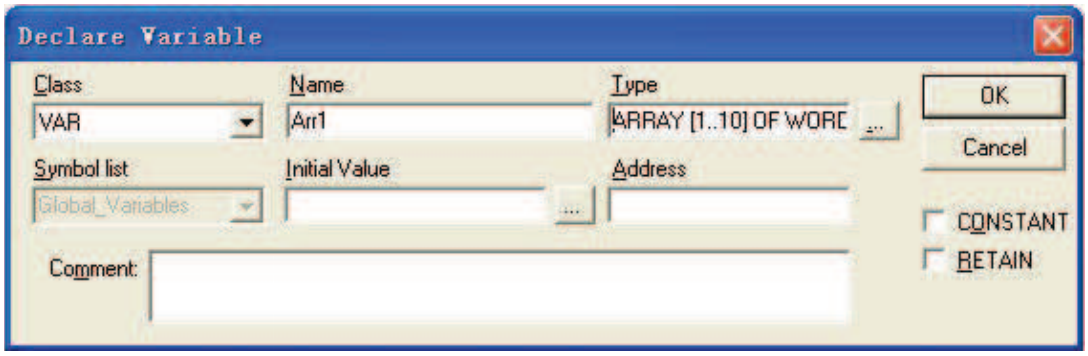


Figure 4-5-1 Auto Declaration of Array

At the declaration of arrays, the assignment of the elements is called initialization of array. At the declaration of arrays, all the elements can be assigned or not.

✧ Example for array definition

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

✧ Example for 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 to which no value is pre-assigned are initialized with the default initial value of the basic type. In the example above, the elements [3] to [10] are therefore initialized with 0.

4.6 User-defined Data Type

In some practical applications, it ’s needed to use some 配方 datas. Each 配方 contains a

number of parameters of different types, such as REAL, WORD or TIME, so it's convenient to use user-defined data type to manage these datas.

About user-defined data type has been introduced in section 2.5.2. Structures are created as objects in the Object Organizer under the register card **Data Types**.

First click the right mousekey/**Add Object** and enter the name of 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 and 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 (the AT declaration is not allowed.)

✧ Example for a structure definition 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 for 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 of the initialization of an array of a structure

```
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 the structure name is Week and the component name is Monday, you can access the component using the following syntax:

Week.Monday

CHAPTER 5 PROGRAM ORGANIZATION UNIT (POU)

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

The basic concepts of POU are introduced in section 5.1 so that the users have a basic knowledge for POU.

In section 5.2 how to create a POU is introduced and in section 5.3 the POU call is introduced. The key point of these two sections is the POU using rule. In 5.4 how to manage the POU using PowerPro software is introduced.

If you meet some concepts or rules not easy to understand in the learning process, it 's suggested to skip over this chapter for new learners and the mode "while leaning, while practicing" is adviced to help you understand the content easily. It 's suggested to learn the following chapters in this mode too.

5.1 BASIC CONCEPT OF POU

POU is short for Program Organization Unit. POU can be Function, Function Block or Program. Each POU consists of a declaration part 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 programming language of POU.

5.1.1 Type of POU

POU includes Program, Function Block and Function.

- Program is a statement sequence written for a certain task and it 's a set of a group of instructions. Program is the only executable POU and it 's the subject of logic execution. The program can be activated by task configuration or be called by other programs.
- Function Block is 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. The function block is different with functions that the function block has no return value.
- Function is 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 can't be executed alone.

5.1.2 Calling POU

Two methods for calling POU:

- Called by POU which is called by other POU.
- Called by **Task configuration** which is limited to program calls. If you don 't configure the **Task configuration**, the system will call main program PLC_PRG automatically.

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

- Program can call function, function block and other progrms.
- Function block can call function and other function blocks.
- Function can call functions.

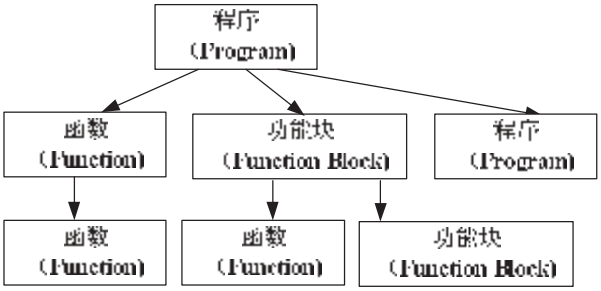


Figure 5-1-1 POU calls

513 Main Program PLC_PRG

PLC_PRG is the default name of main program and it 's 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** additionally. So each project must contain the PLC_PRG as a main program to call other POUs.

52 NEW POU

521 New Program

Click the right mouse in the register card "POUs" in the object organizer, and select "Add Object", the default name of the new POU is "PLC_PRG" if there is no POUs. In the dialog "New POU" select "Program" in "Type of POU", shown in figure 5-2-1.

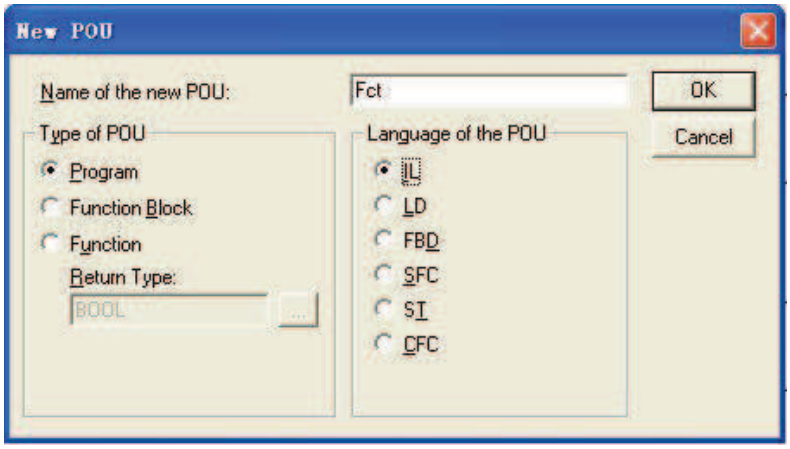


Figure 5-2-1 New Program

522 New Function Block

New function block is similar with new program. Click the right mouse in the register card "POUs" 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".

523 New Function

The external structure of function is similar with that of function block. The only difference is that function has only one output. How to new function is introduced below.

Click the right mouse in the register card "POUs" in the object organizer, and select "Add Object". In the dialog "New POU" select "Function" in the field "Type of POU",and according to requirement select or enter in the fields "Return Type", "Language of the POU" and "Name of the new POU", shown in figure 5-2-2. If the name of the new POU is "Fct" then the function name is "Fct". Generally the characters which have actual meaning are used for naming to make it easier to identify.

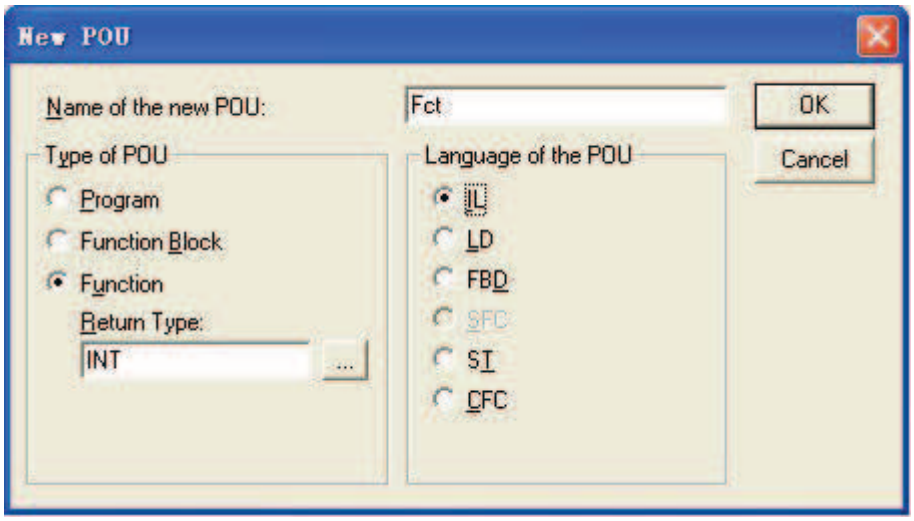


Figure 5-2-2 New Function (1)

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

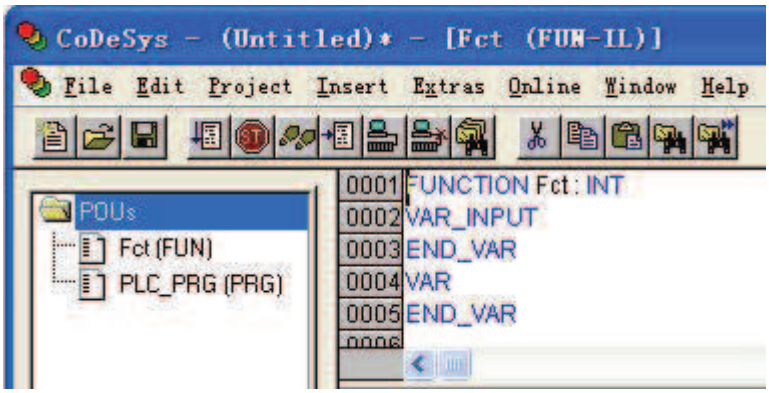


Figure 5-2-3 New Function (2)

Take a mathematical formula as an example to describe how to write a function, 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.

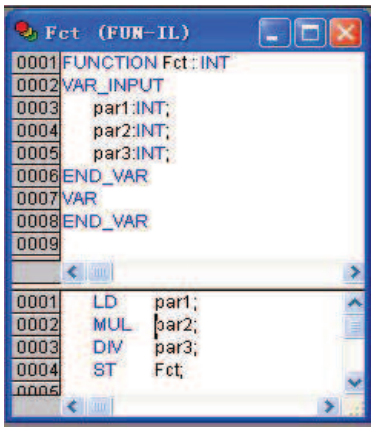


Figure 5-2-4 New Function (3)

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

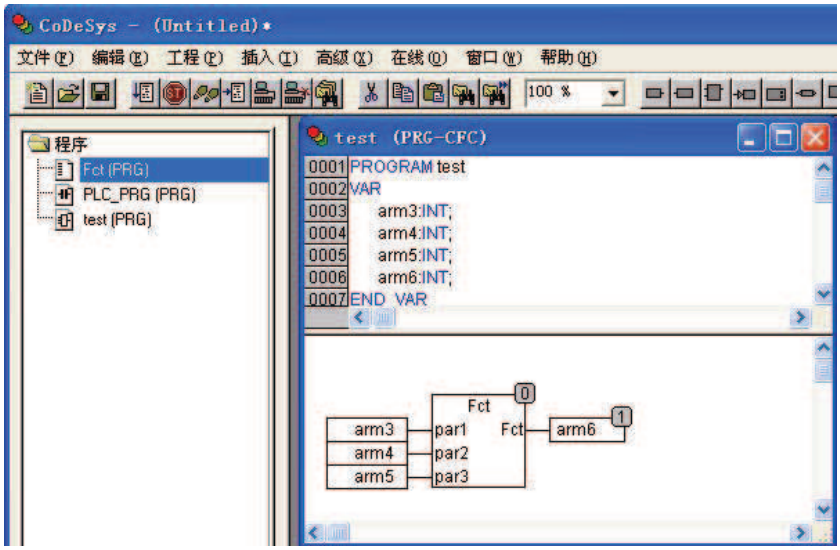


Figure 5-2-5 New Function (4)

5.3 CALLING A POU

Two methods for calling POU:

- Called by POU which is called by other POU.
- Called by **Task configuration** which is limited to program calls. If you don't configure the **Task configuration**, the system will call main program PLC_PRG automatically.

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

- Program can call function, function block and other progrms.

- Function block can call function and other function blocks.
- Function can call functions.

5.3.1 Calling a Program

Program is the only POU which can be implemented. Program can call up function blocks and functions. A program declaration begins with the keyword PROGRAM, shown in figure 5-3-1.

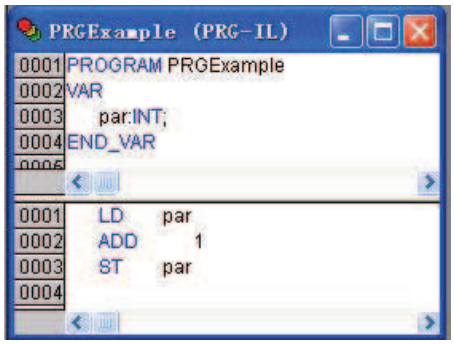


Figure 5-3-1 Program Example

Programs can be called up by other programs but it can not be called up by functions.

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

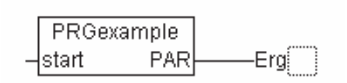
Example for program calls in IL:

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

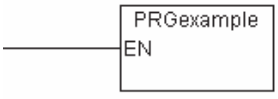
Example for program calls in ST:

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

Example for program calls in FBD:



Example for program calls in LD:



5.3.2 Calling a Function Block

➢ Function Block

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.

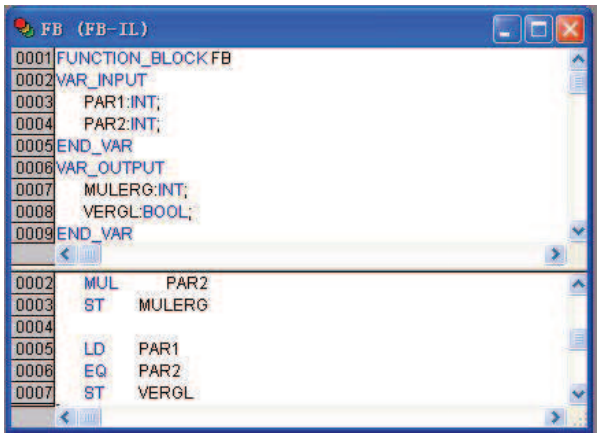


Figure 5-3-2 Example of Function Block

➢ Declaration of Function Block Instance

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

INSTANZ: FUB;

Each function block instance is an independent active object which can realize a special logic function. The 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 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".

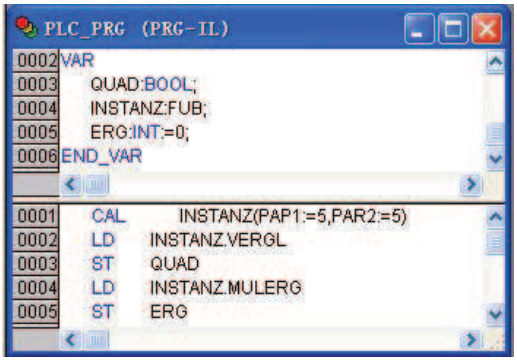
You can set the input parameters in the text languages IL and ST by assigning values to the parameters after the instance name of the function block in parentheses. For input parameters this assignment takes place using ":=" just as with the initialization of variables at the declaration position.

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

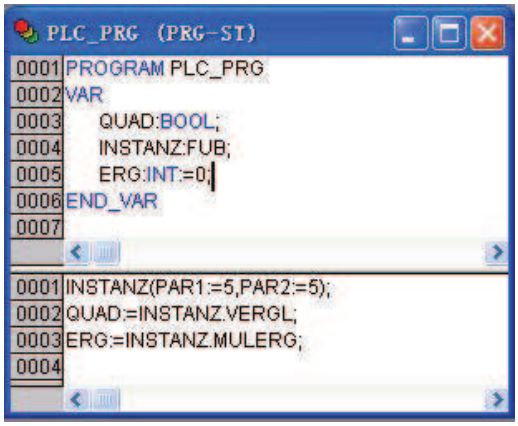
All values are retained after processing a function block until the next it is processed. Because of the internal variables, function block calls with the same inputs do not always return the same

output values.

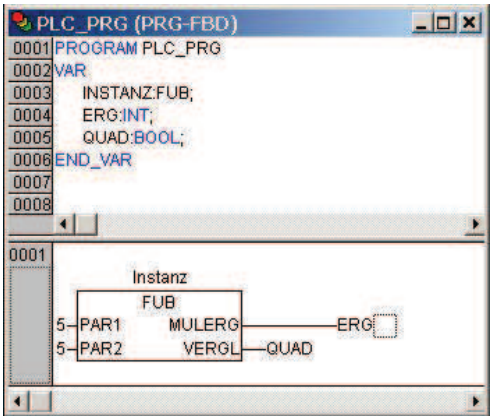
The function block FUB can be called in different languages, shown in figure 5-3-3. The multiplication result is saved in the variable ERG, and the result of the comparison is saved in QUAD.



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”

Now Box with EN will be introduced so that one can distinguish function block from Box with EN. The general difference is that the calling mode is different. For function block, it will be

executed when the program runs no matter it 's enabled or not. But Box with EN will be executed and called only when the **EN** is enabled. But the detailed application is not introduced, and here take a simple program written in LD for an example to introduce the differences, shown in figure 5-3-4.

From the 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 time is 1s, then %QX0.0 is set 1 and the related channel light Q0.0 in PLC turns on. For ADD, when it 's 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 then ADD will be executed.

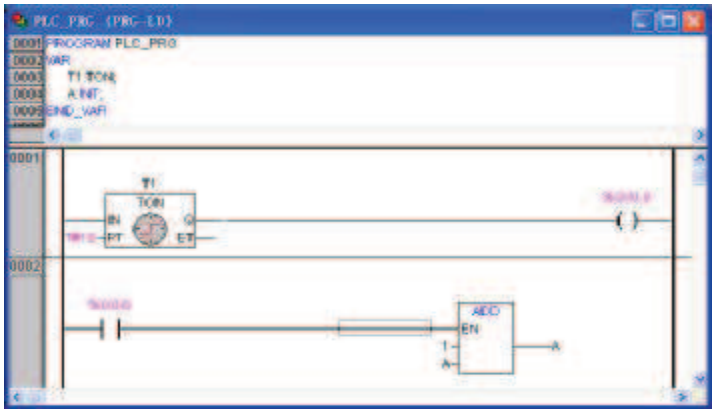


Figure 5-3-4 Function Block and Box with EN

5.3.3 Calling Functions

A function is a POU, which yields exactly one data element when it is processed for a series of specified inputs. Compared with function blocks, functions have only one output and do not have any internal conditions. That means that calling up a function with the same input parameters always produces the same value. The common used mathematical operators such as SIN(X) are typical function types. When declaring a function do not forget that the function must receive a data type as type of the return value. Assign the result to function itself, that 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 both input variables get multiplicated 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.

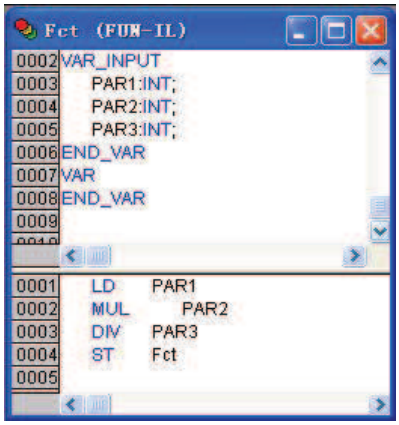


Figure 5-3-5 Example of Function

✧ Example for calling the above described function Fct:

In IL:

LD 7

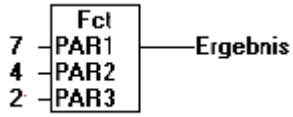
Fct 4,2

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 for range overflows, shown in figure 5-3-6.

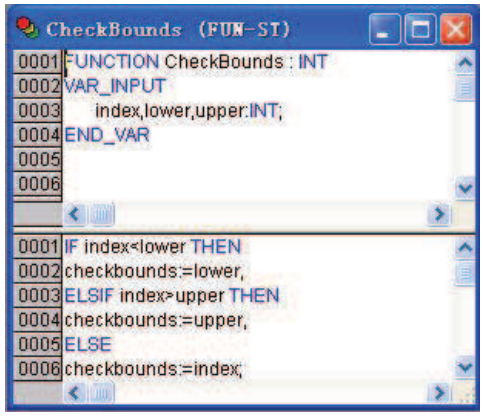


Figure 5-3-6 Example for Function Call (1)

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.

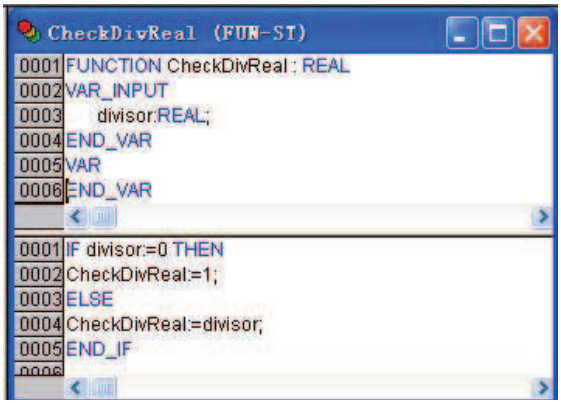


Figure 5-3-7 Example for Function Call (2)

Use the output of function CheckDivReal as the divisor of the operator DIV, 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.

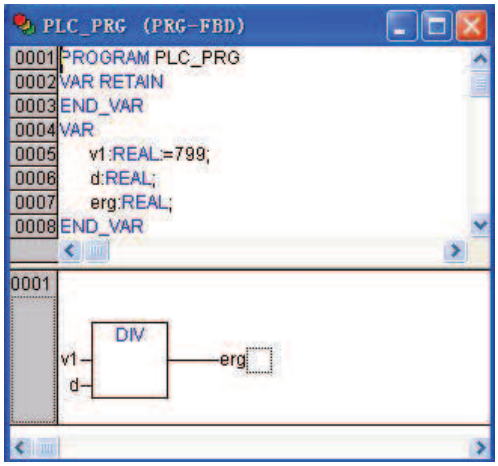


Figure 5-3-8 Example for Function Call (3)

5.4 MENU FOR MANAGING POU

Click the right mouse on the selected POU in the register card "POUs" in object organizer in the main window of PowerPro software and the menu is shown in figure 5-4-1.

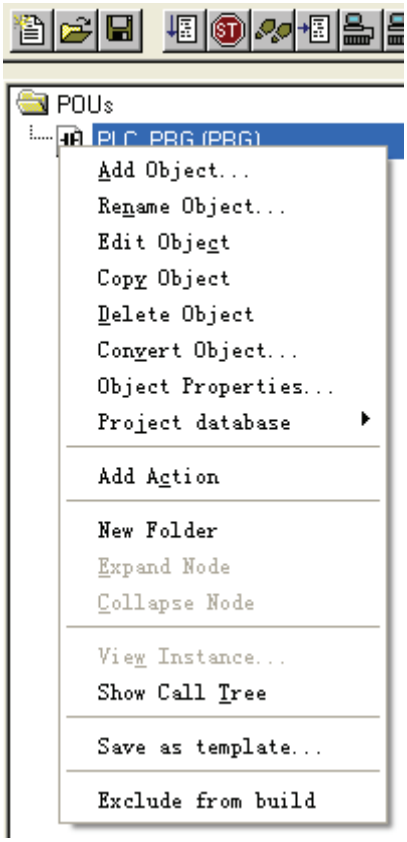


Figure 5-4-1 Menu for Managing POU

541 New Folder

In order to manage and keep track of larger 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 closed folder symbol) .With Drag&Drop you can move the objects as well as the folders within their object type. For this select the object and drag it with pressed left mouse button to the desired position, shown in figure 5-4-2. Folders have no influence on the program, 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, then the new one is created underneath it. If a file has been selected, then the new one is created on the same level.

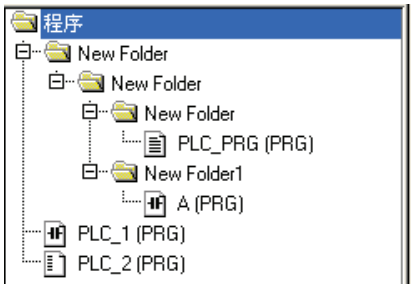


Figure 5-4-2 Folder

542 Convert Object

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 program, function and function block. Before this the project must be compiled. In "Target Language" choose the language into which you want to convert and give the POU a new name. Remember that the name of the POU may not have already been used. Then press **OK**, and the new POU is added to your POU list.

> Note:
>
> In the conversion process some usless sentences will be generated. If the conversion between different languages happens frequently, please first delete the usless sentences generated in the conversion process to avoid unnecessary troubles.

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

- It's difficult to convert multi-nested ST to LD. The sentences written in IF, THEN, CASE, FOR, WHILE and REPEAT format can't be converted into function block networks directly.
- It's very difficult to convert IL language to other languages unless the application scope and writing form of IL operators are limited strictly. It's easy to convert the program 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 block and its related parameter values.

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

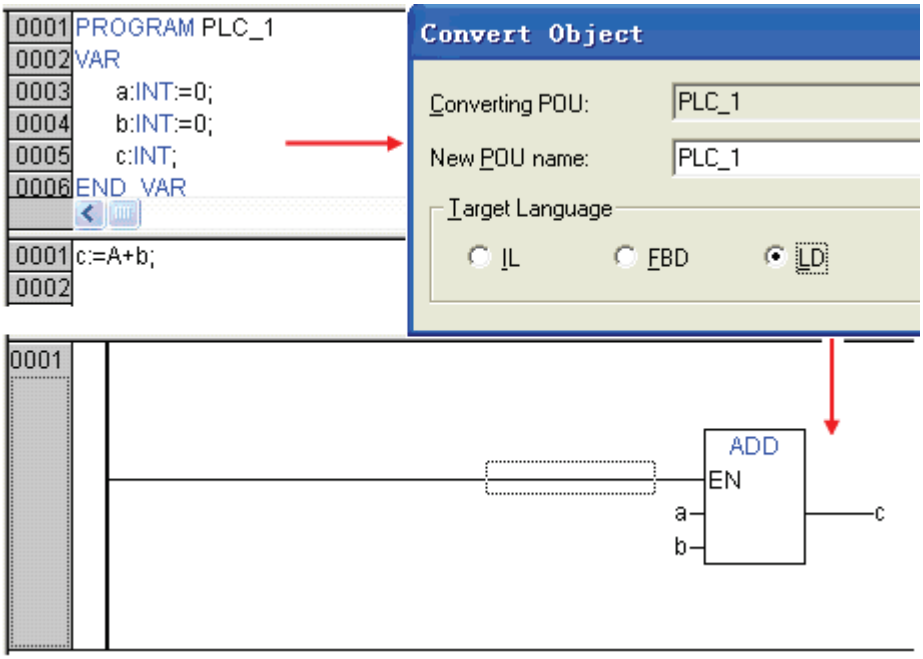


Figure 5-4-3 Convert Object

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 introduced 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 series PLC. The so-called task is a 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 task configuration process.

6.1 PLC WORKING PROCESS

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

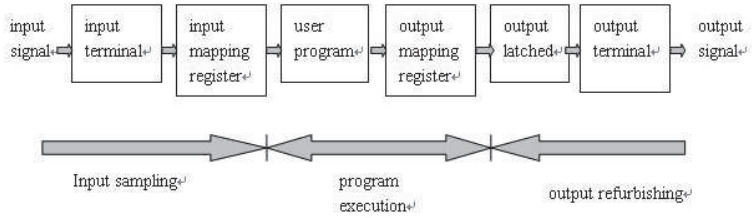


Figure 6-1-1 PLC Working Process

- 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 and the process is called sampling. In this working cycle the sampling result can't be changed and it will be used in the process of executing users' program.

- executing users' program

PLC scans every instruction in the program in order from up to down and from left to right and calculates and processes the data got from the input mapping register and output mapping register. Then output the calculating result to output mapping register which is used for storing executing result and save it. Regard that before the execution is finished, the executing result will not be transferred to output port.

- output refurbishing

After the whole user program is executed, PLC outputs the content in output mapping register to 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, general the input sampling time and output refurbishing time are less than 1ms and the time for executing users' program varies from 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.

1, system self-checking: check whether the program is executed correctly and the CPU stops working if time out.

2, exchange information with PowePro: this is executed once when using programmer input and debugging.

3, network communication: when a communication module is configured, it ’s used to data exchange with communication object.

When PLC runs the three steps above will be executed repeatedly i.e. cycle scanning working process, 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.

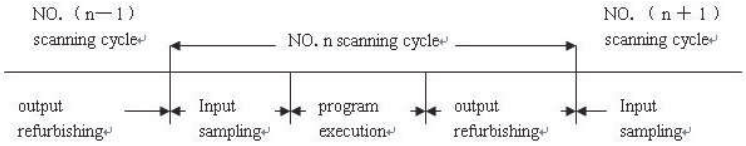


Figure 6-1-2 PLC Scanning Cycle

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 endless loop or execute not-prefixed program resulting in system paralysis.

Note:

In default LM series PLC starts up the watch-dog function automatically, when the scanning time exceeds 500ms then it’s considered that the program is in endless loop and it will be re-started. And now PLC ERR light flashes for six times in a lower speed and then the program is reset and starts to execute again.

6.2 TASK CONFIGURATION

For a project, according to different requirements many tasks can be configured to call

different programs. Generally it 's suggested to configure a task to call the main program and the other programs are called by the main program indirectly. Only programs can use this kind of calling mode and function blocks and functions can't be called like this.

Strictly if the task configuration is not configured, in single task environment the default main program is PLC_PRG and it's called automatically and uniquely and the other programs are called by the main program. If the the task configuration is configured, 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.

621 Task Configuration

Double click the "Task Configuration" in the register card "Resources" and the dialog for setting the task configuration is opened. Click the right mouse on the "Task Configuration" and select the "Append Task", shown in figure 6-2-1.

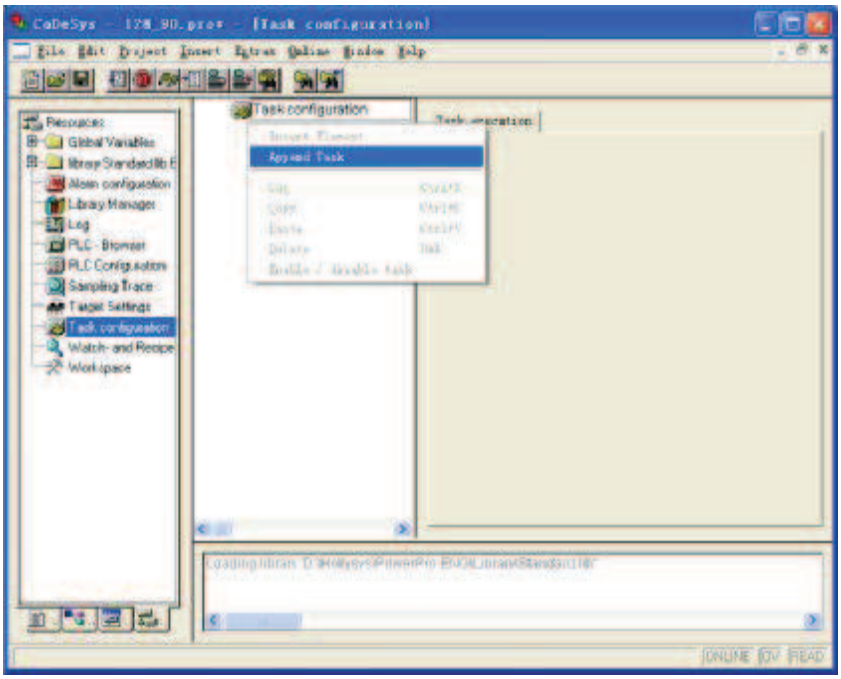


Figure 6-2-1 Task Configuration

The dialog for setting the "Taskattributes" will be opened and enter the information in the fields "Name", "Priority", "Interval" and so on, shown in figure 6-2-2.

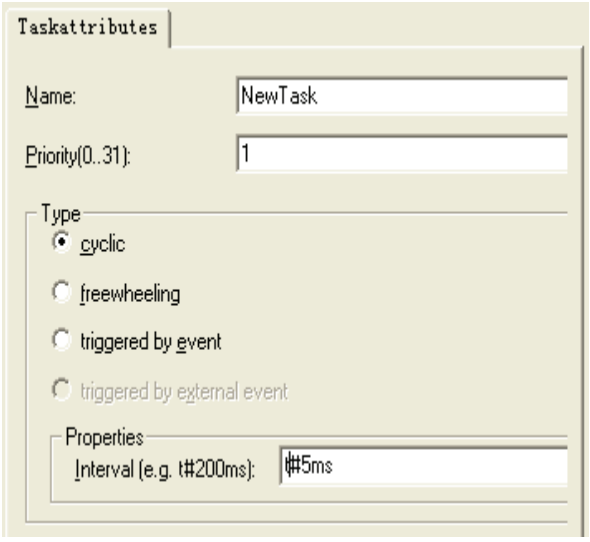


Figure 6-2-2 Task Attributes

- 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 can't support the function

Triggered by external event: PLC can't 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" also a "system event" can be used to call a POU of your project. Call the corresponding POU when the related event is triggered. For example, call the corresponding POU when T2, T3, T4 timers overflow and fast external interrupt runs.

Double click "Task Configuration" in "Resources" register card, and the window of task configuration is opened 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,

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 mouseclick on the control box) and then in the column "called POU" enter the name of the project POU which should be called and processed as soon as the event occurs.

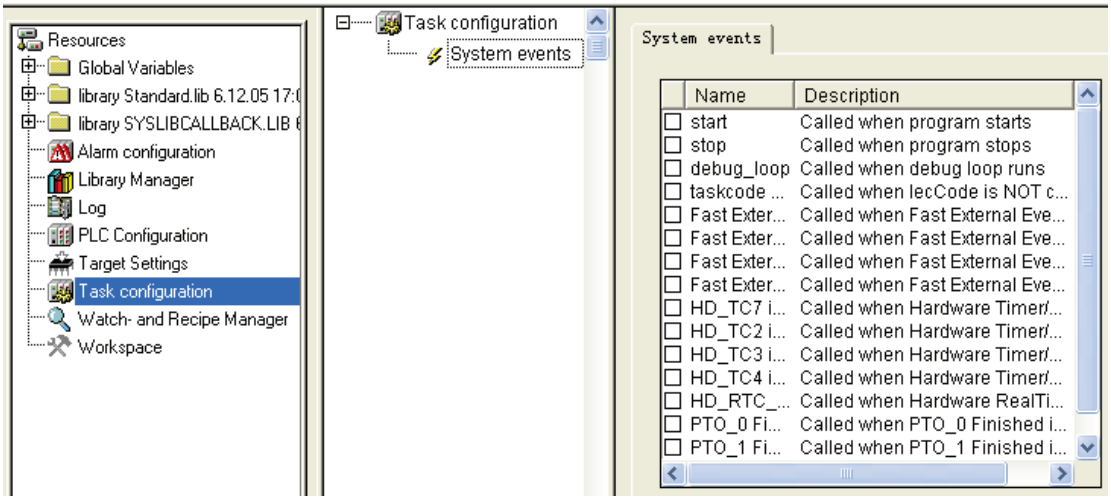


Figure 6-2-3 Dialog of System Events

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

> Note:
>
> 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.

623 Program Call

Click the right mouse on the "NewTask" and select "Append Program Call", then the dialog of "Program Call" is opened, shown in figure 6-2-4.

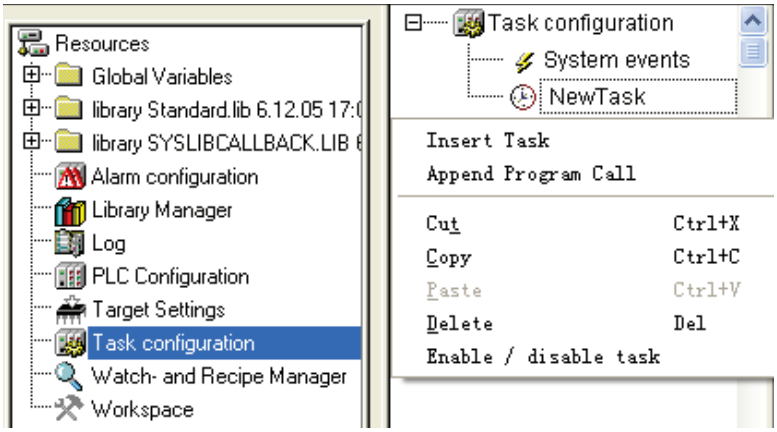


Figure 6-2-4 Append Program Call (1)

Click the ellipsis shown in the figure 6-2-5 and select the program needed and return by

clicking “OK” with the program called by the task.

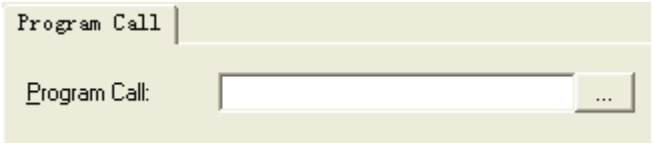


Figure 6-2-5 Append Program Call (2)

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

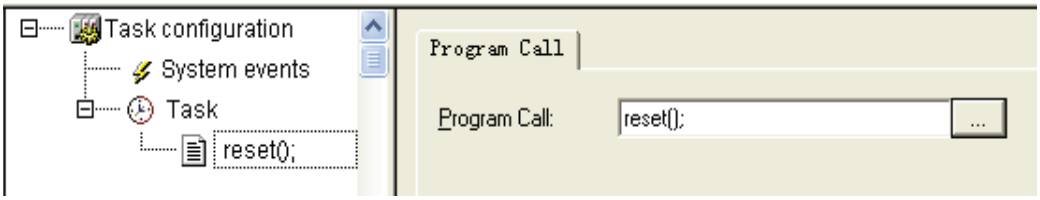


Figure 6-2-6 Example of Task Configuration

CHAPTER 7 NEW AND MANAGE A PROJECT

After introduing the programming environment in PowerPro, how to new a project will be introduced in this chapter. A project contains all the objects, including POU, data type, resources and arithmetic. The sequence of new a project is flexible, and the basic steps include target setting, new main program, and hardware configuration, save project.

7.1 TARGET SETTINGS

After the command "File"/"New" in the main window or the button "  " the dialog of "Target Setting" opens. "Target" is the storage space of PLC and "Target Setting" is to configure according to the selected PLC storage space.

In "Configuration" field select "HOLLiAS-LEC G3 CPU Extend " and the target is 120KB. Click the "OK" button a dialog box appears, shown in figure 7-1-1.

If the storage space of the used module is 28KB select "HOLLiAS-LEC G3 CPU". If you're not sure about the storage space refer to the appendix.

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

Then the window of "Target Settings" is opened and the default setting can satisfy most of the application requirements and click "OK" to exit, shown in figure 7-1-2.

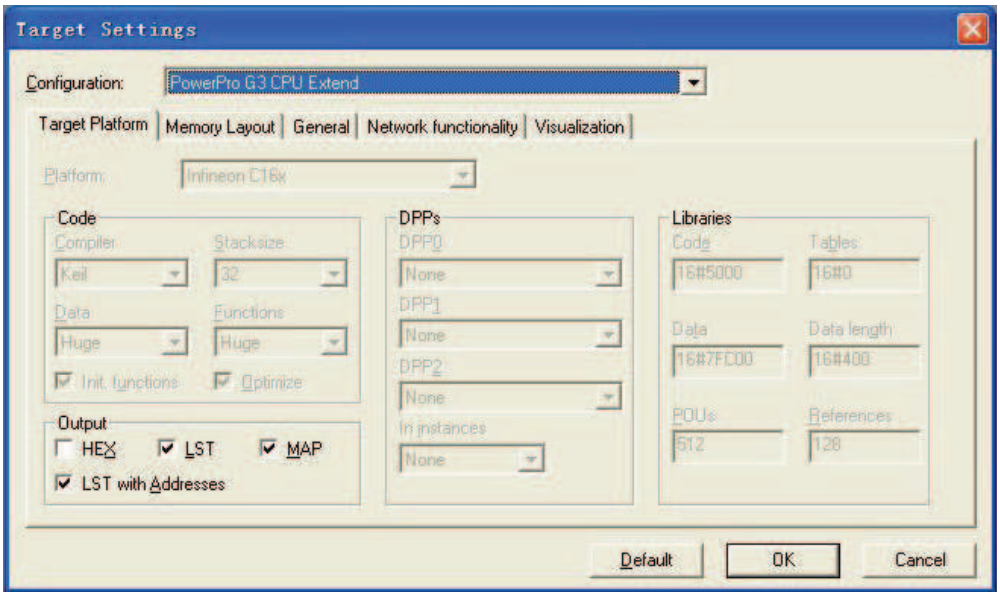


Figure 7-1-2 Target Settings

7.2 NEW POU

Each project must contain a main program which can call other subprograms. When new a project, the default "PLC_PRG" is specified as main program and you can't change it. Or else the user program can't run normally.

After the target setting is finished a dialog box of "New POU" appears, shown in figure 7-2-1. In the "Language of the POU" field, you can select any of the languages including IL, LD, FBD, SFC, ST and CFC. Here we select LD. In the "Type of POU" field select "Program", the default main program name is "PLC_PRG" and click "OK" button to exit the window.

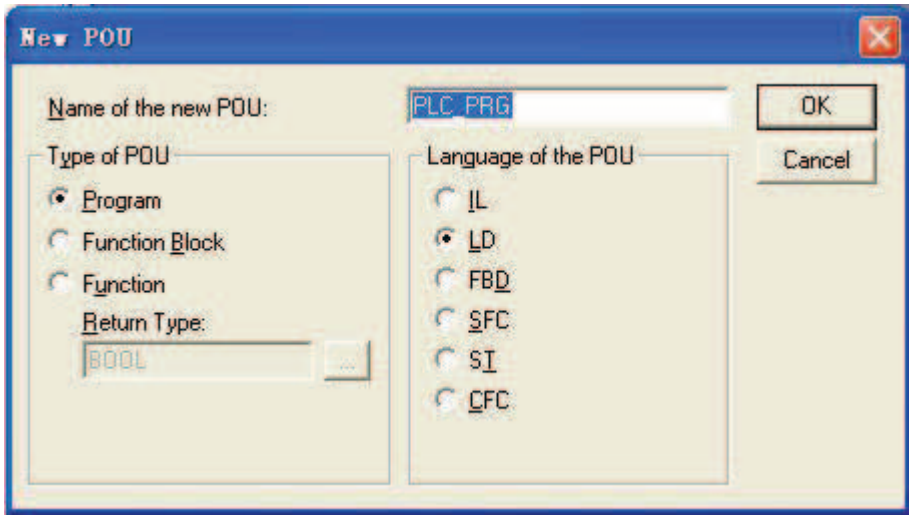


Figure 7-2-1 New Main Program "PLC_PRG"

7.3 HARDWARE CONFIGURATION

PLC system acquires and processes the data through hardware modules (including CPU modules and expansion modules). Input channels acquire the data from field and output channels control the electrical equipments in process. In order to complete the acquirement and control, configure the hardware modules according to the actual project.

7.3.1 Configuration of CPU Modules

In "Resources" register card select "PLC Configuration" and the configuration interface appears. Click in the PLC configuration window with the right mouse-button and select command "Append Subelement" and select the CPU type to establish a "PLC Configuration" tree. Here select CPU moduleLM3107, shown in figure 7-3-1.

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

The input in word unit starts from %IW0 and each input channel occupies a bit. The addresses are: %IX0.0,%IX0.1,.....,%IX0.7,%IX1.1,%IX1.2,.....,%IX1.7 and the former 14 addresses are

valid.

The output in word unit starts from %QW0 and each output channel occupies a bit. The addresses are: %QX0.0,%QX0.1,.....,%QX0.7,%QX1.0,%QX1.1,.....,%QX1.7 and the former 10 addresses are valid.

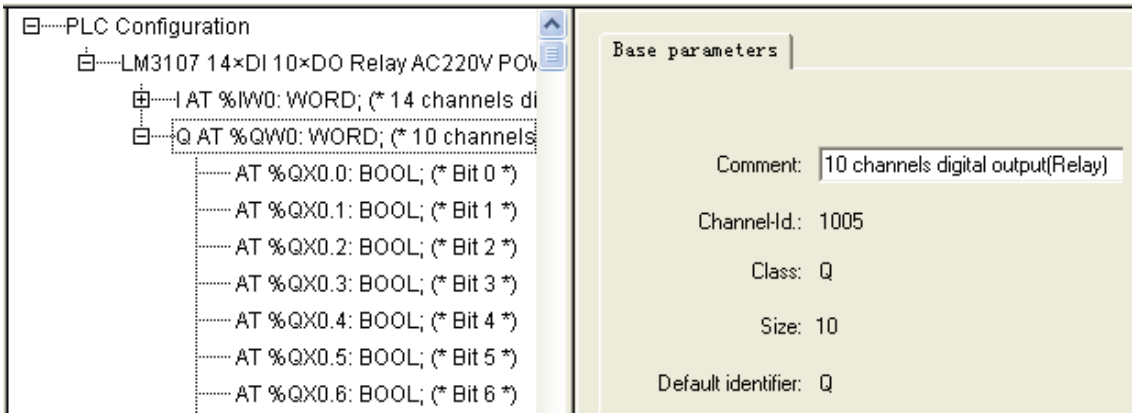


Figure 7-3-1 LM3107 Base Parameters

For CPU input channel configure its Input_Filter. The input filter parameter setting is shown in figure 7-3-2 and the default value is "64". Select in 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 consecutive digit (i), which numbers through all the way the parameters of the module.
- Name: Name of the parameter.
- Value: Value of the parameter, editable.
- Default: Default value 64 of the parameters.

The definition of "filter parameter" is as follows: if the acquired data maintains the same during 64 scanning cycles then the acquired data is valid and the filtering is finished. Here the filter is for digital, i.e. the number of filter is 64.

Filter parameter is invalid for high-speed input channels. There is no need to set the channel filter parameter where the high-speed input channel is used and even the settings are finished they are useless.

The parameter configuration of the other CPUs is similar. According to the actual control requirement, select an appropriate CPU and add the expansion modules.

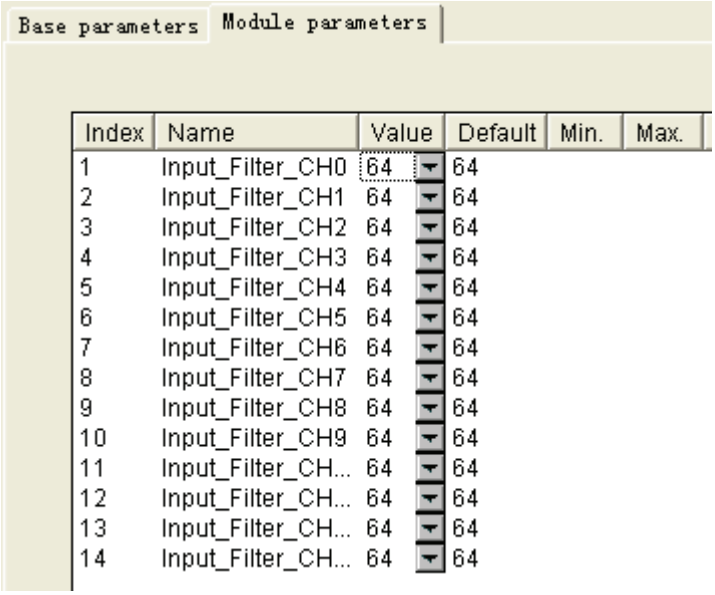


Figure 7-3-2 Parameter Settings of LM3107 Module

7.3.2 Configuration of Expansion Modules

Add the expansion modules when the CPU module is added. Click the right mouse on the CPU module and select "Append Subelement" to select the expansion module needed in the list, shown in figure 7-3-3. For example if the "LM3230" module is selected then it will be displayed in "PLC Configuration" tree, shown in figure 7-3-4.

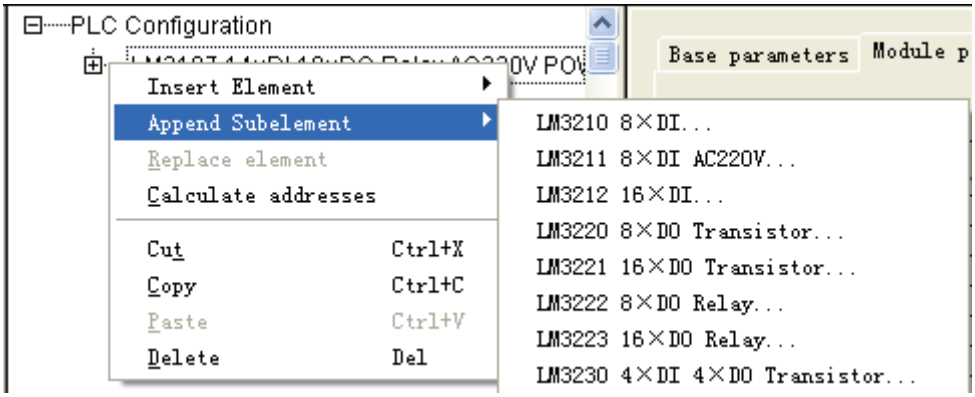


Figure 7-3-3 Append Subelement (1)

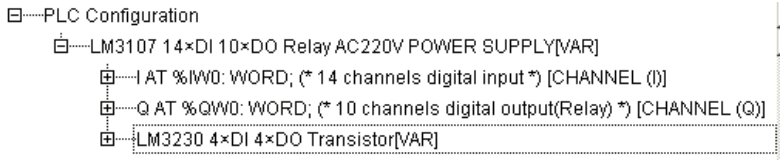


Figure 7-3-4 Append Subelement (2)

Each I/O channel of CPU modules and expansion modules corresponds to a actual physical device and the corresponding relationship is displayed with module in the "Base parameters". "Base parameters" includes Node id, Input address, Output address, Diagnostic address and so on,

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" can't be changed by users freely. "Input address" and "Output address" are the initial addresses of the I/O storage area corresponding to module channels.

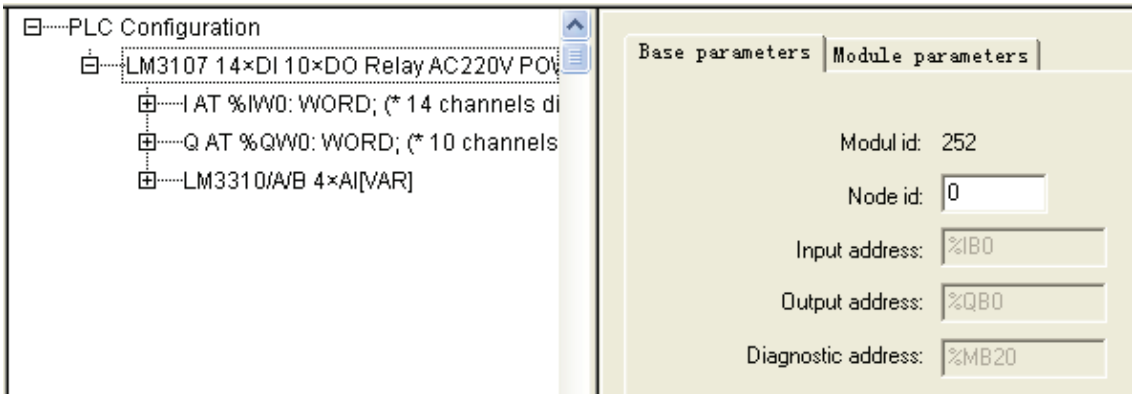


Figure 7-3-5 Base Parameters of Expansion Modules

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

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

If necessary you can define a general variable for a number of channels. For example the default setting of "%IW0" of CPU is "I", then 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", shown in figure 7-3-6.

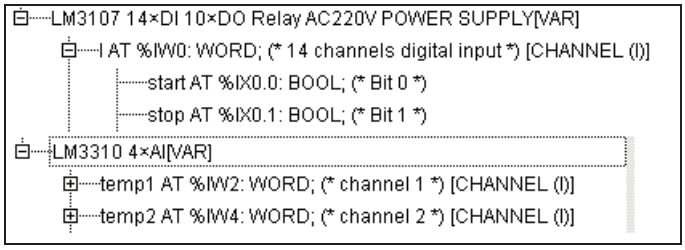


Figure 7-3-6 Definition of I/O Variable

When the cursor position is at "PLC Configuration", and the dialog of the option card "Settings" is shown at the right of the window, including "Automatic calculation of addresses", "Check for overlapping addresses" and "Save configuration files in project", 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 ”: At 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.

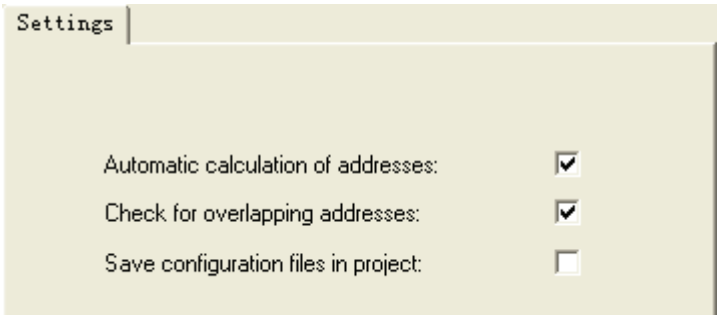


Figure 7-3-7 “Settings”

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

7.4PROGRAMMING

After the PLC configuration has been finished, you can start to write a program. When new a program, select the programming language including IL, LD, ST, SFC, FBD and CFC. Here take the common used LD language for an example to describe the programming specification in PowerPro. The programming specifications for other languages are introduced in chapter 9.

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

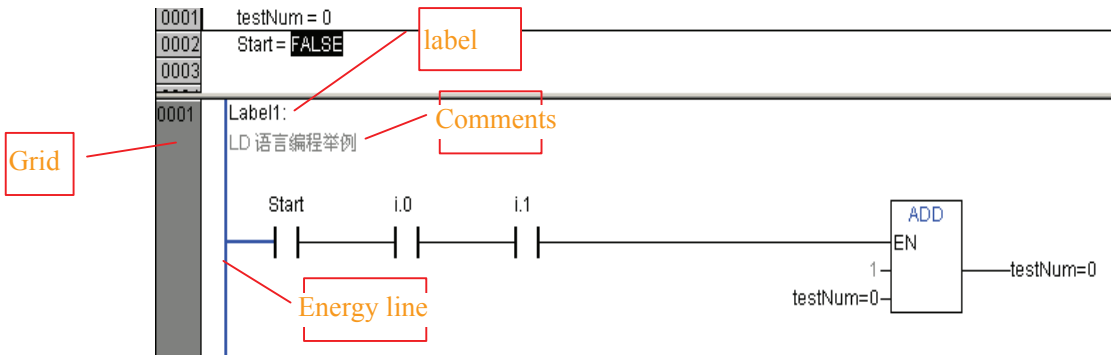


Figure 7-4-1 LD Editor

Click the right mouse in the instruction part in workspace and the common used commands are displayed in the context menu, 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.

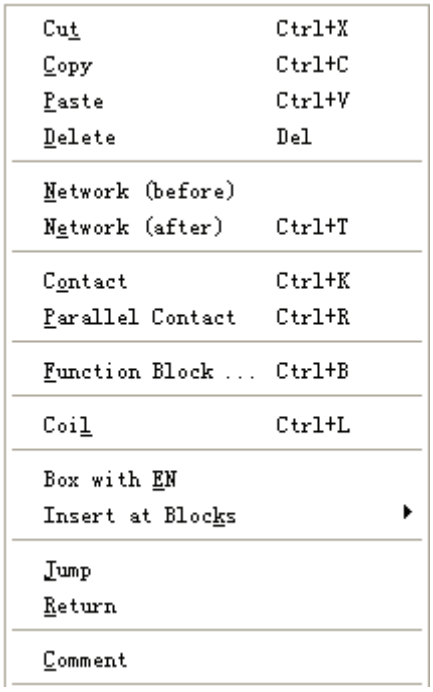


Figure 7-4-2 Context Menu

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), shown in figure 7-4-3. If, for example, the number 2 is entered, then, 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, then in order to enter a comment you simply click in the comment line and then enter the comment. In contrast to the program text, comments are displayed in grey.

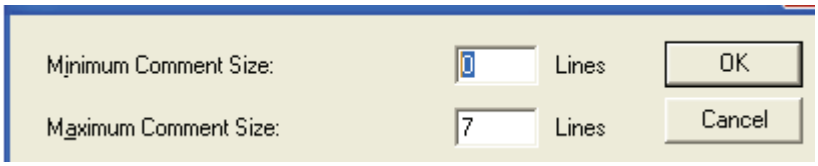


Figure 7-4-3 Settings of Comment

7.4.2 Insert Contact and Coil

- Insert "Contact"

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

If the marked position is a coil or the connecting line between the contacts and the coils, then 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, shown in figure 7-4-4.

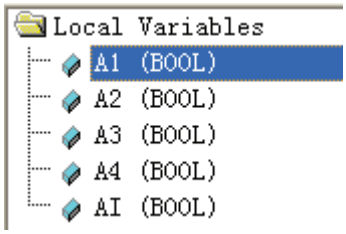


Figure 7-4-4 Variable List

- Insert "Parallel Contact"

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

If the marked position is a coil or the connecting line between the contacts and the coils, then 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, 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, 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 in 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, then it copies the negated value in the appropriate Boolean variable. If a contact is negated, then it connects through only if the appropriate Boolean value is FALSE.

- "Set/Reset" Operation

Shortcut:  Coils can be defined Set or Reset status.

A Set Coil is designated with an "S" in the coil symbol. Once you have set the value of this variable to TRUE, it will always remain at TRUE until it's reset again.

A Reset Coil is designated with an "R" in the coil symbol. Once you have set the value of this variable to FALSE, it will always remain at FALSE until it's set again.

7.4.3 Add Instructions

The instructions in PowerPro can be called in two ways: Box with EN and Function Block. The using method will be introduced below.

- Call Box with EN

In PowerPro instruction system, some standard instructions, such as Arithmetic Operators, Comparison Operators, Bit-Shift Operators, Assignment Operator, Type Conversion, Logical Operators and so on, 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" in 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, shown in figure 7-4-5.

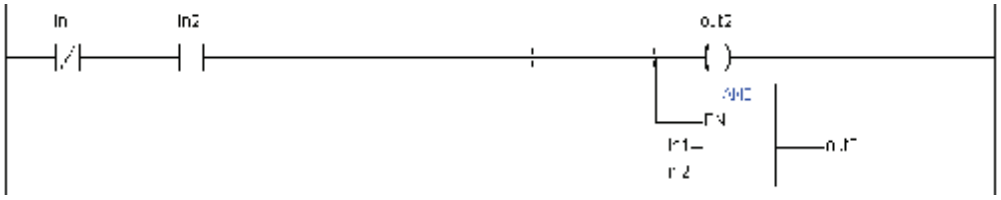


Figure 7-4-5 Insert Box with EN

The new inserted contains initially the designation "AND". If you wish, you can change this designation 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**, shown in figure 7-4-6.

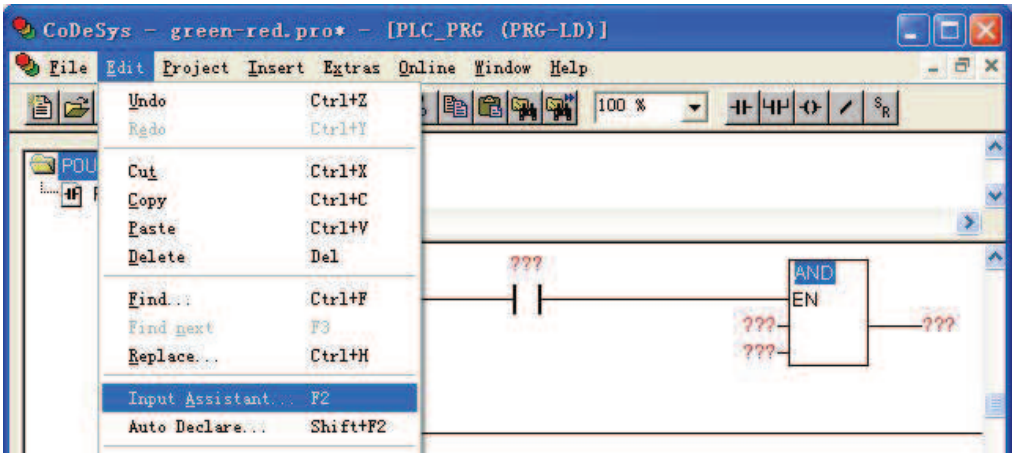


Figure 7-4-6 Input Assistant

➢ Call Function Block

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

Before calling the instruction in the form of function block, first you should what is a library. In PowerPro software, the common used instructions are gathered to create special 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 library please refer to section 7.4.4.

The instruction can be called in the form of 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 opened and select the instruction needed, shown in figure 7-4-7.

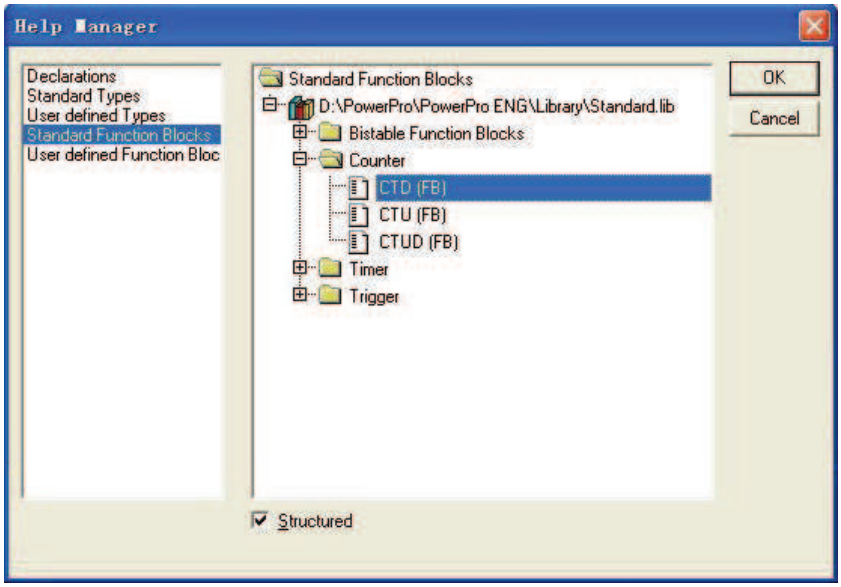


Figure 7-4-7 Insert Function Block

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

Note:

When using function block instruction, instances are declared as varibles whereas the name of the function block is indicated as 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 Library

In PLC programming some instructions or function blocks are often used, such as string functions, trigger, counter, PID and so on. In PowerPro, the common used instructions and function blocks are gathered and classified to create special libraries.

Library is the collection of instructions and function blocks and all the libraries contain a file "library name.lib" (including input and output codes of instructions and function blocks) . If you want to call the instruction or function block add the library file"library name .lib".

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

- Library Manager

Library Manager is used to manager library functions and function blocks and it contains all standard functions and function blocks. Library Manager is opened with the command "Window"/"Library Manager", 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.

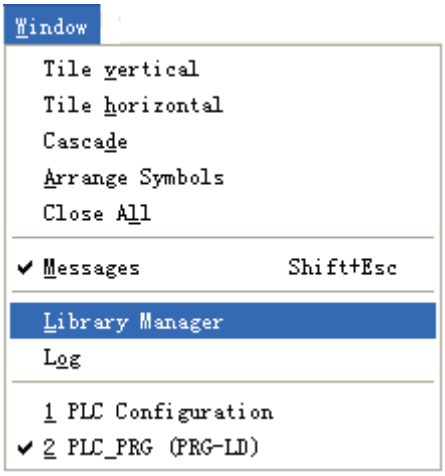


Figure 7-4-8 Open Library Manager

The window of the library manager is divided into four areas, 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 a function or function block then the declaration will appear in the upper right area of the library manager and in the lower right is the graphic display.

It's very important to learn how to view the function library. In the function library some very important messge is displayed, such as the number of input variables and intermediate variables and their data types, variable comments and the intermediate variables which must be initialized and so on.

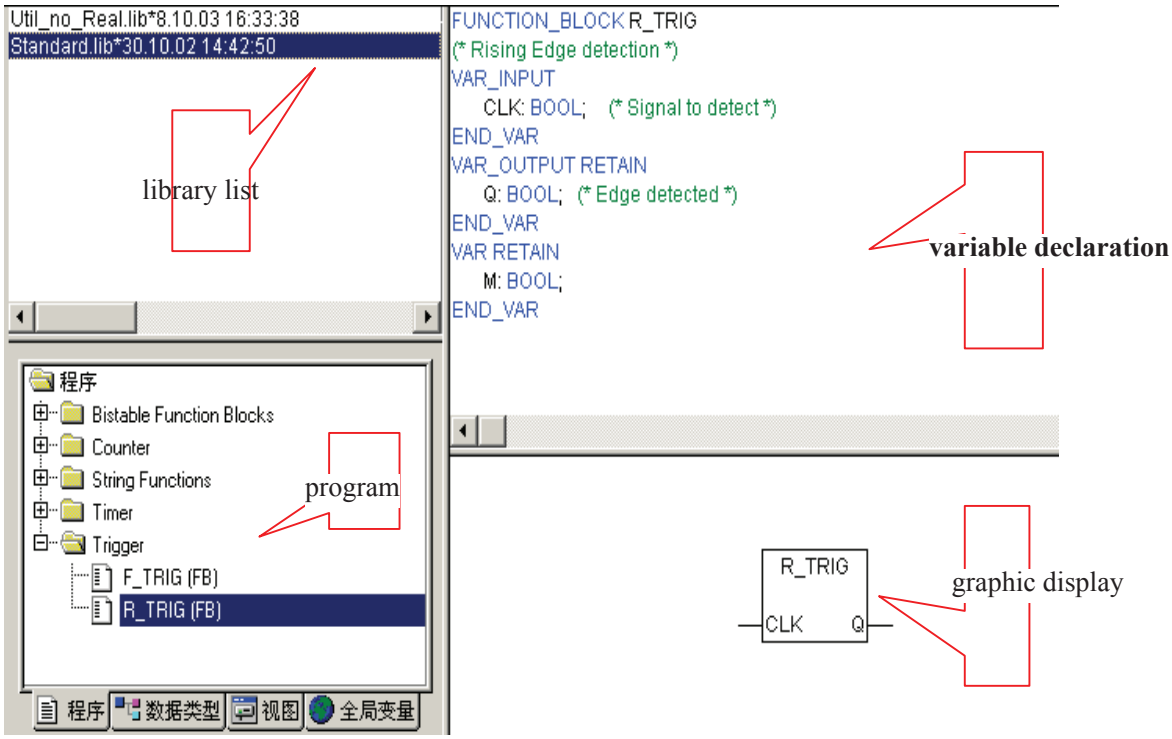


Figure 7-4-9 Window of Library Manager

➢ Additional Library

Additional libraries are needed when the library files in library list can ’t meet the programming requirements. Because there is a conflict in the data types in Util.lib and Util_no_Real.lib and errors appear while compiling, so they are not allowed to add 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, shown in figure 7-4-10.

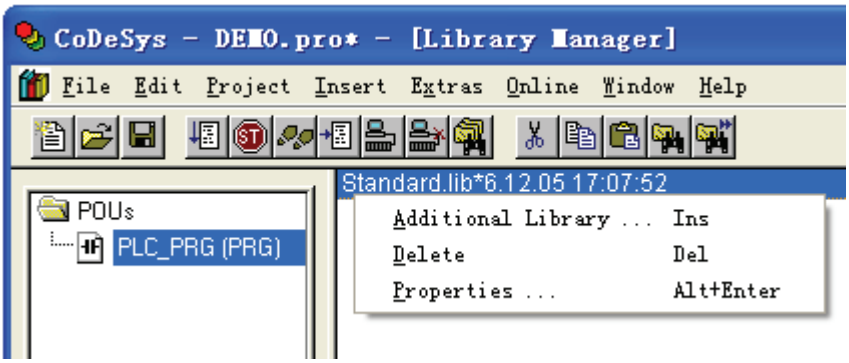


Figure 7-4-10 Additional Library

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

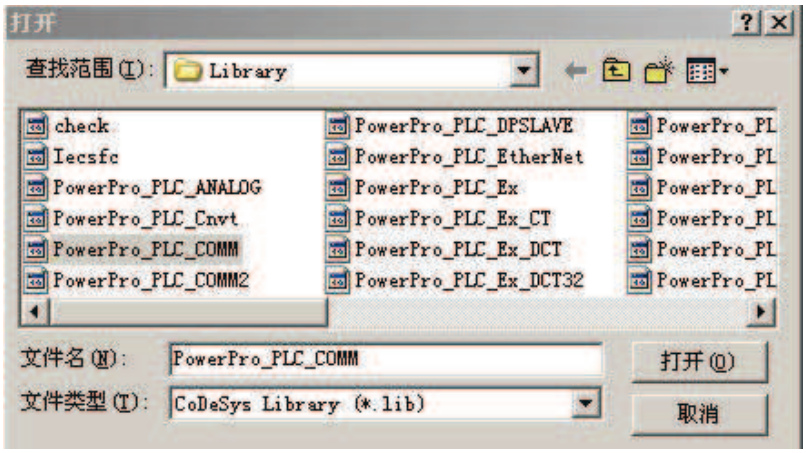


Figure 7-4-11 Open Library

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

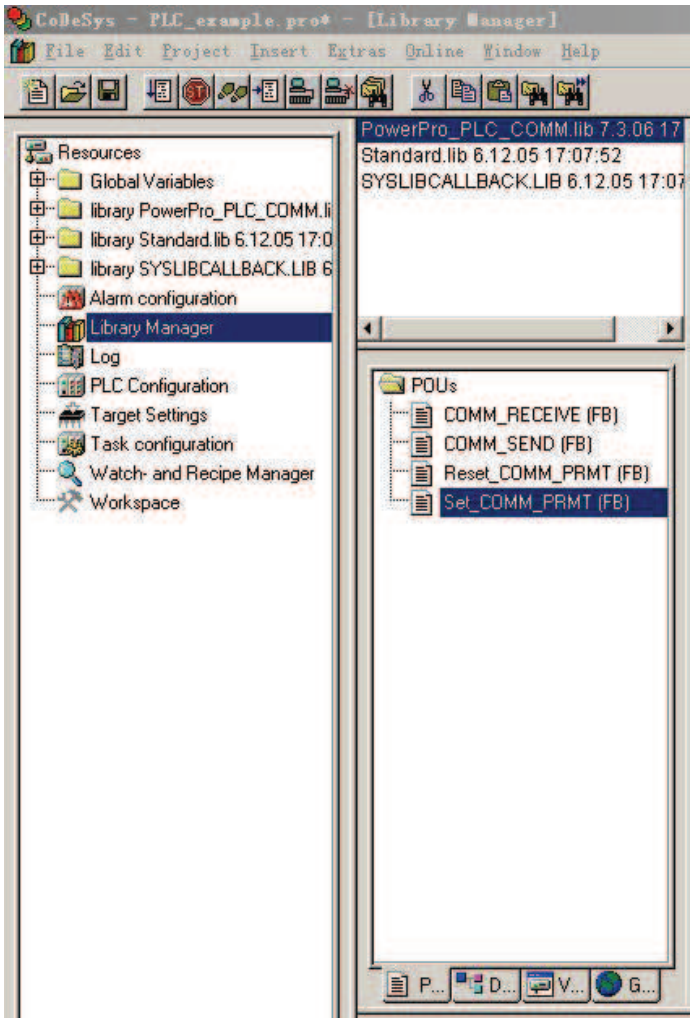


Figure 7-4-12 Display Library

➢ Delete Library

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

Note:

After new a project, the Standard.lib and SyslibCallBack.lib are added automatically, and the other libraries are needed to add manually.

7.45 Create a Library

In project implementation, some algorithm or logic are used repeatedly and can be used in many other projects then thses logics are included in a library convenient for programming, engineering maintenance and technological resources sharing. Passwords are used to protect your algorithm and logic in a library against being opened or changed. How to create a library is as below:

First new a project, and select "None" in the field "Target Settings"/"Configuration", shown in figure 7-4-13.

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

Click "OK", and a "New POU" window appears, 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 7-3-2, select "ST" in the field "Language of the POU", and enter the name "Generate_CRC" in the field "Name of the new POU".

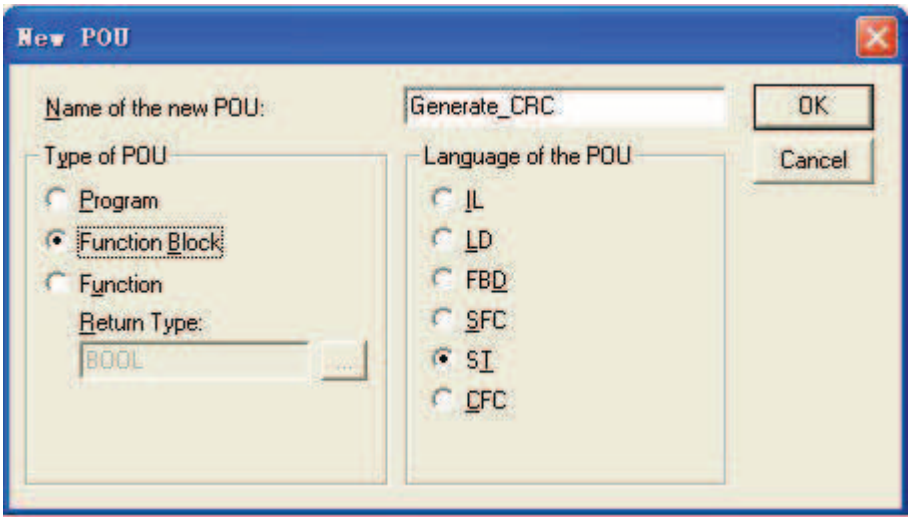


Figure 7-4-14 User-defined Library (2)

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

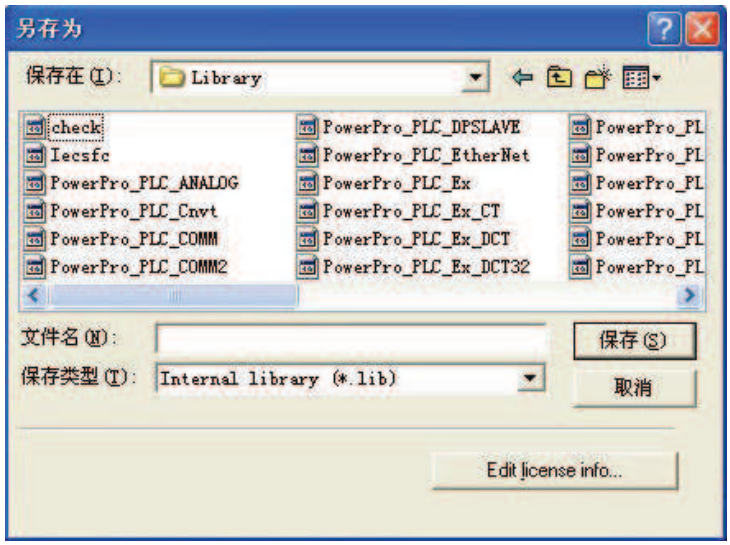


Figure 7-4-15 User-defined Library (3)

✧ Example

The example is used to generate a library for CRC16 check written in ST language, 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 in other computers, you have to copy it and paste it to the software installation directory PowerPro\Library in 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, i.e it can ’t be called by itself.

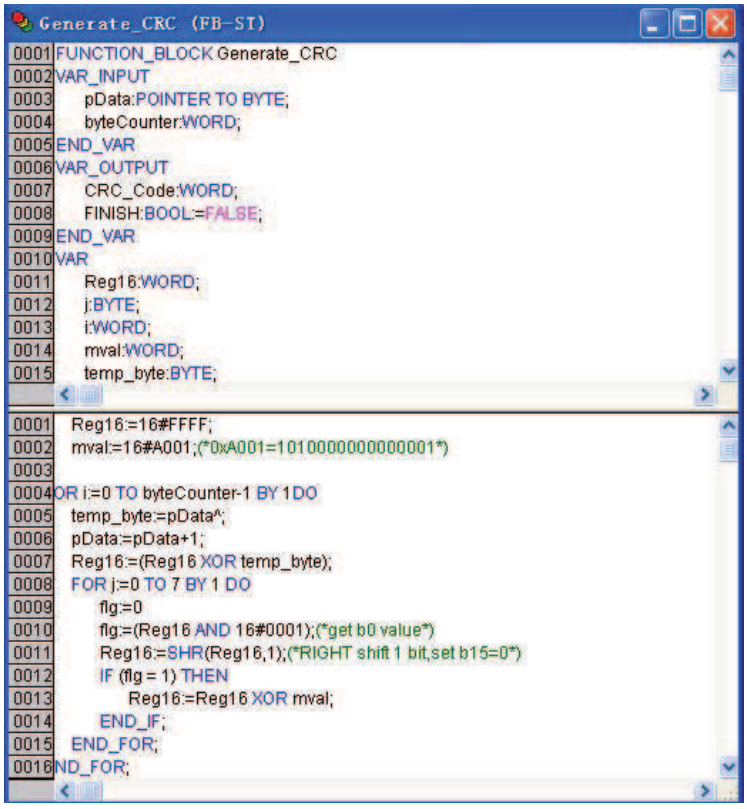


Figure 7-4-16 User-defined Library (4)

For used-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, shown in figure 7-4-17.

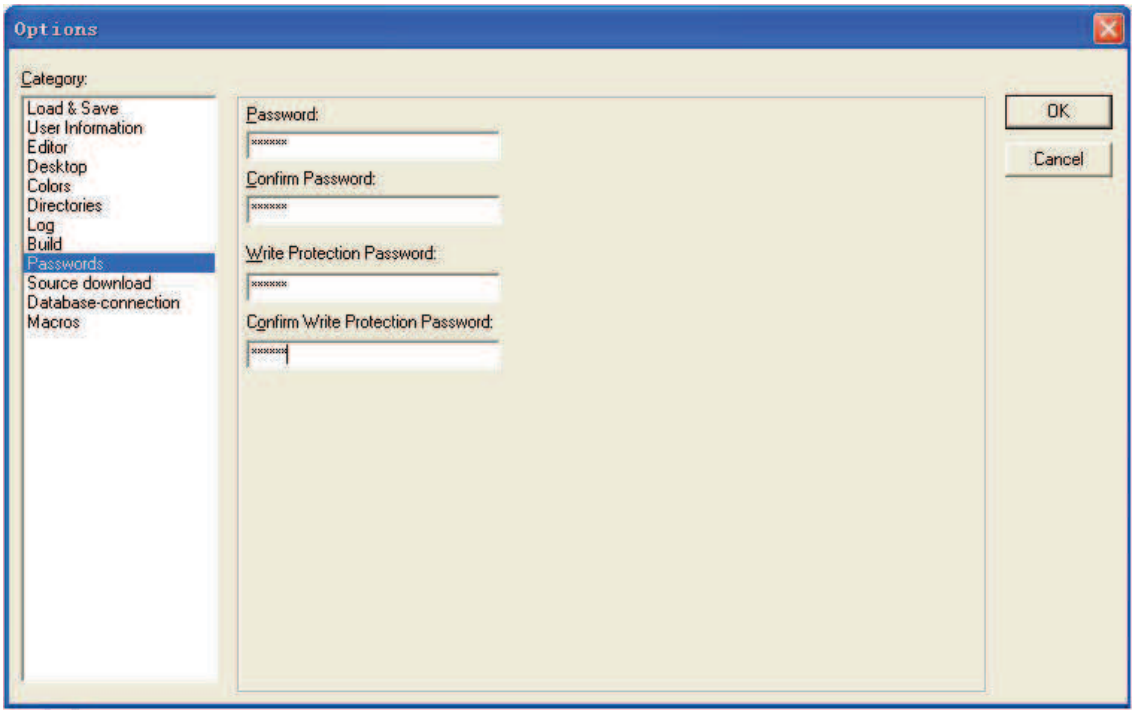


Figure 7-4-17 User-defined Library Encryption

7.46 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, 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, 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.

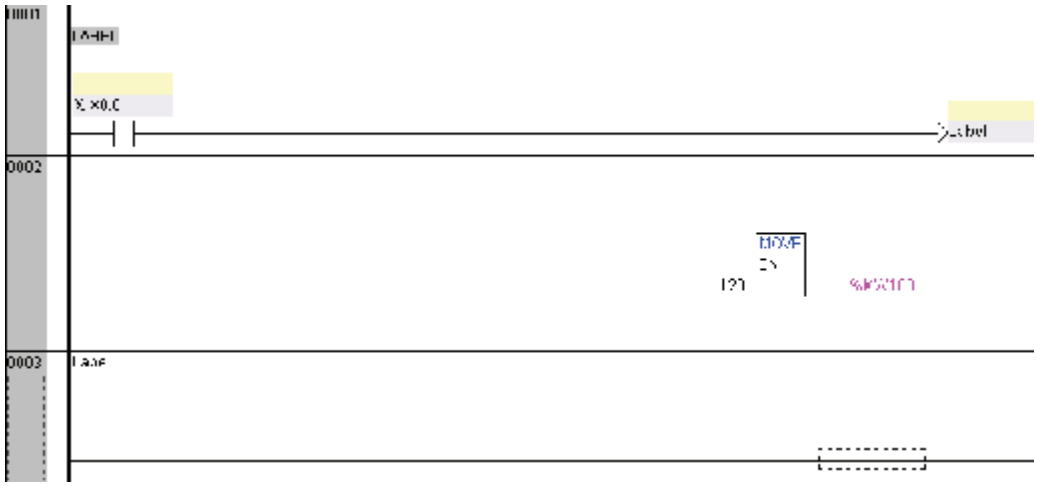


Figure 7-4-18 Jump

- 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, shown in figure 7-4-19.

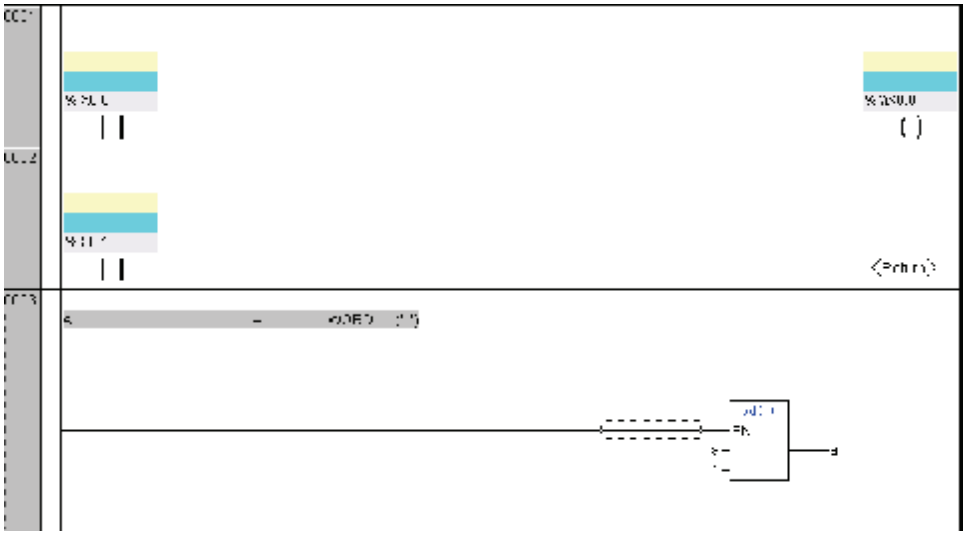


Figure 7-4-19 Return

7.47 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, shown in figure 7-4-20. Refer to section 7.4.3 about how to call a “Box with EN”.

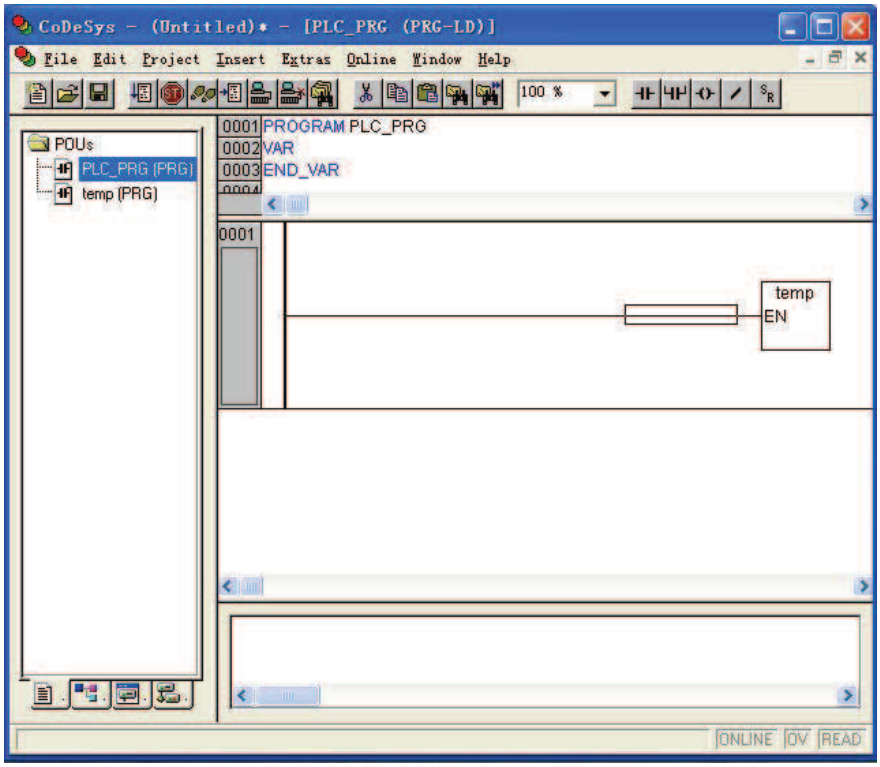


Figure 7-4-20 Call Subprogram

If you want to transfer parameters between 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 Add Comment

In order to make it easier to read the program, comments can be added for program, network or variable and address. There are a number of possibilities to add comments in PowerPro.

- Comments for program and network

The comment for program is the same with that for network. In PowerPro comments can be entered for each network, shown in figure 7-4-21. Refer to section 7.4.1 for comments.

- Comments for variables

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

- Comments for address

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, shown in figure 7-4-21. See section 7.4.9 about ladder diagram options.

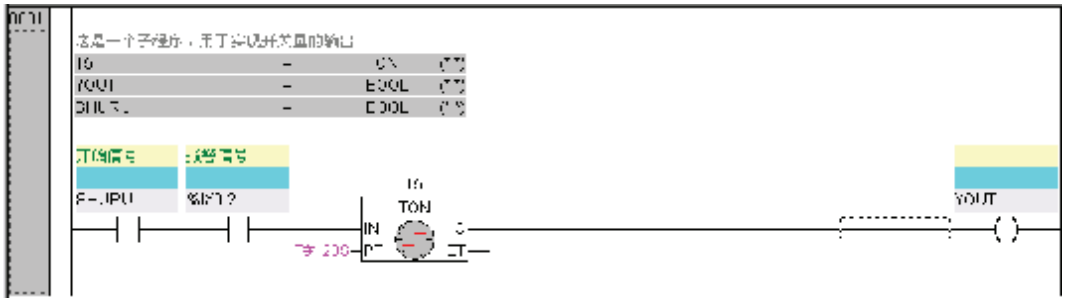


Figure 7-4-21 Comments

Note:

Adding comments for networks is different from adding jump label 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.

7.4.9 Ladder Diagram Options

Select "Extras"/"Options", a dialog box appears, shown in figure 7-4-22, and you can set different options for your ladder diagram.

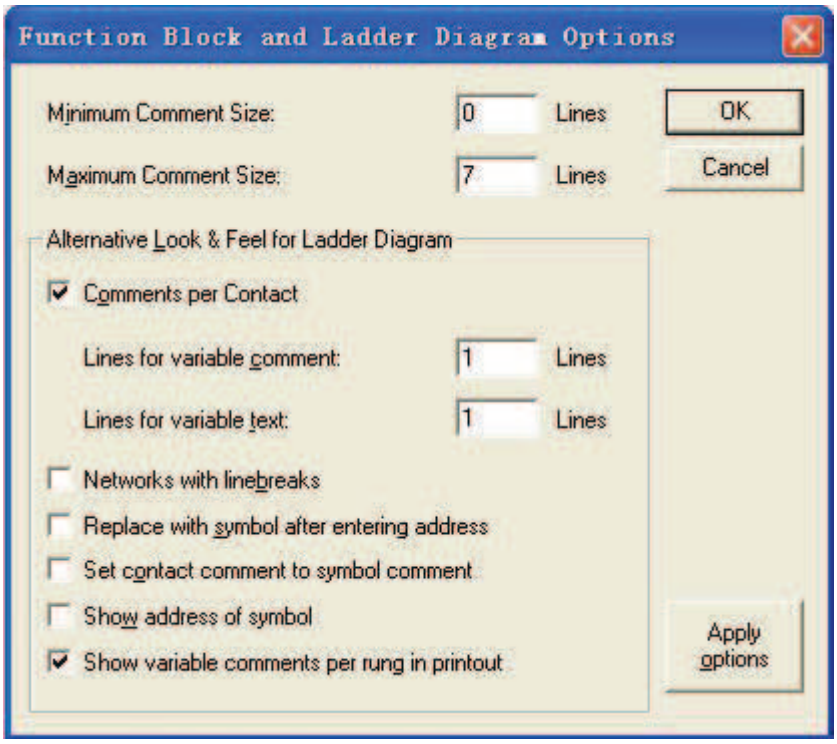


Figure 7-4-22 Ladder Diagram Options

- **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 resp. at a contact or coil and this address will be replaced immediately by the name of the variable which is 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 field "File name" and the meaningful characters and numbers are suggested to use.

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

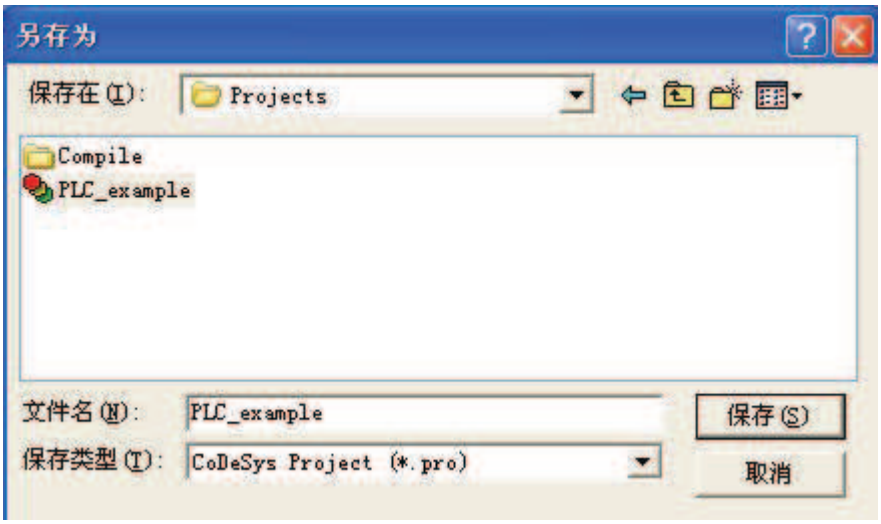


Figure 7-4-23 Save Project (1)

After the project is saved, the "(Untitled)*" in the upper left corner of the main window will change into the project file name, shown in figure 7-4-24. In the whole process of creating a

project remember to save the project momentarily to avoid misoperations resulting data loss. When the project has been changed but not saved yet, a "*" will appears after the project name in the upper left corner and it disappears when the project is saved.

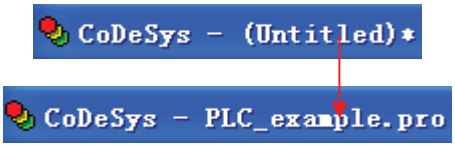


Figure 7-4-24 Save Project (2)

With the "Project"/"Options" menu or "Workspace" in "Resources" register card 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 softwate 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.

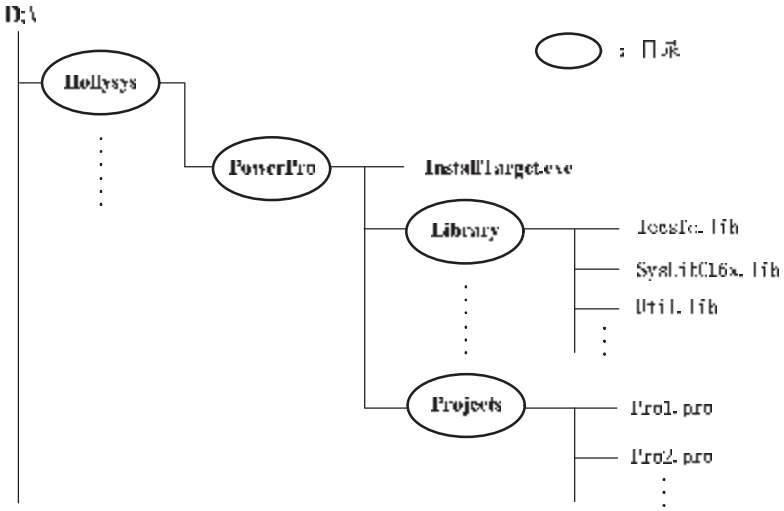


Figure 7-5-1 Software Directory Tree

Open the "Project" menu in the main window, shown in figure 7-5-2.

Figure 7-5-2 Manage Project Menu

The common used commands for managing project are introduced below.

7.5.1 Print 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, Resources and so on, shown in figure 7-5-3. Resources include Global Variables, PLC Configuration, Alarm configuration, Workspace, Watch- and Recipe Manager, Task configuration and Parameter Manager and so on.

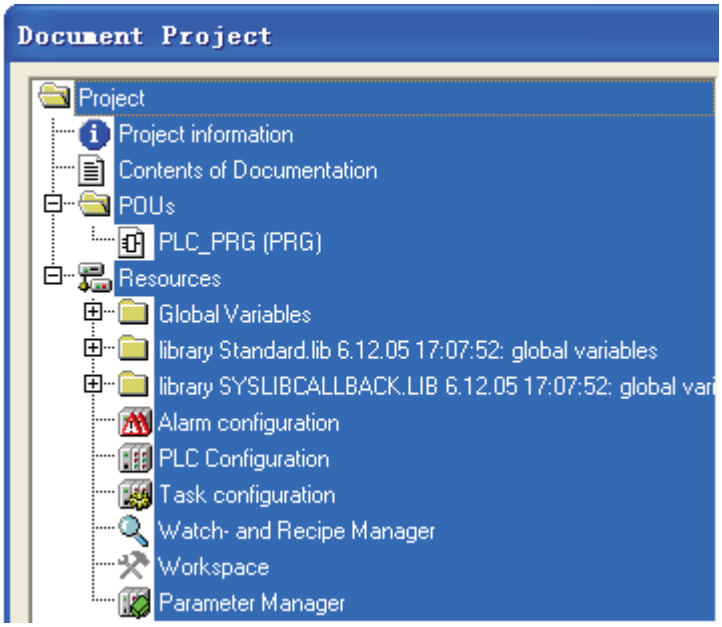


Figure 7-5-3 Print Document Project

Only those areas in the dialog box are printed which are highlighted in blue. If you want to select the entire project, then select the name of your project in the first line. If you only want to select a single object, then click on the corresponding object. Objects which have a plus sign in front of their symbols are organization objects which contain other objects. With a click on a plus sign organization object is expanded, and with a click on the resulting minus sign it can be closed

up again. When you select an organization object, then 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", shown in figure 7-5-4.

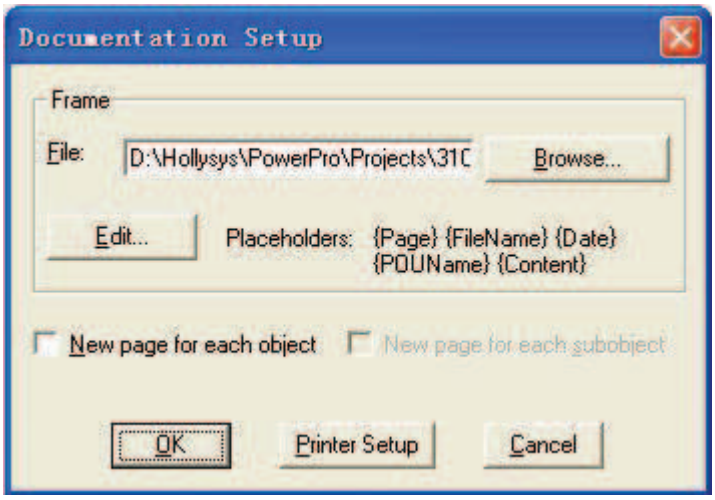


Figure 7-5-4 Documentation Setup Dialog Box

- 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, then the frame for setting up the page layout appears. With the command "Insert"/"Placeholder" can insert Page, POU-Name, Filename, Date and Content by dragging a rectangle the layout while pressing the left mouse button.

- New page for each object

If the option is activated begin a new page for each object.

- New page for each subobject

If the option is activated begin a new page for each subobject.

- 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, then the frame for setting up the page layout appears, 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, then CoDeSys asks when the window is closed if these changes should be saved or not.

In order to be aware of the page format which will be valid for printouts, define the layout as

described above and additionally activate option “Show print area margins” in “Project” “Options” “Desktop”.

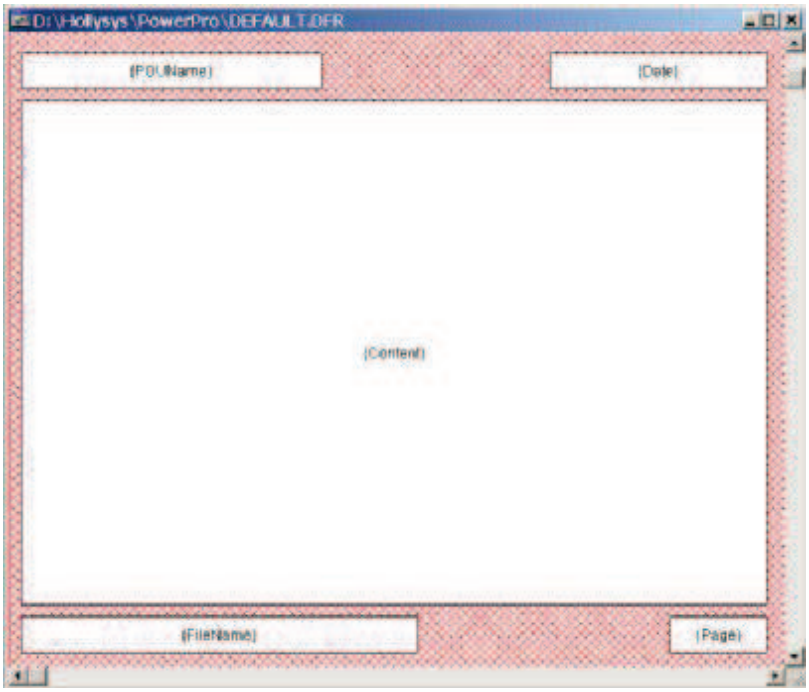


Figure 7-5-5 Page Layout

7.5.2 Import and Export 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, shown in figure 7-5-6. Switch on or off the option “One file for each object” and then click on OK.

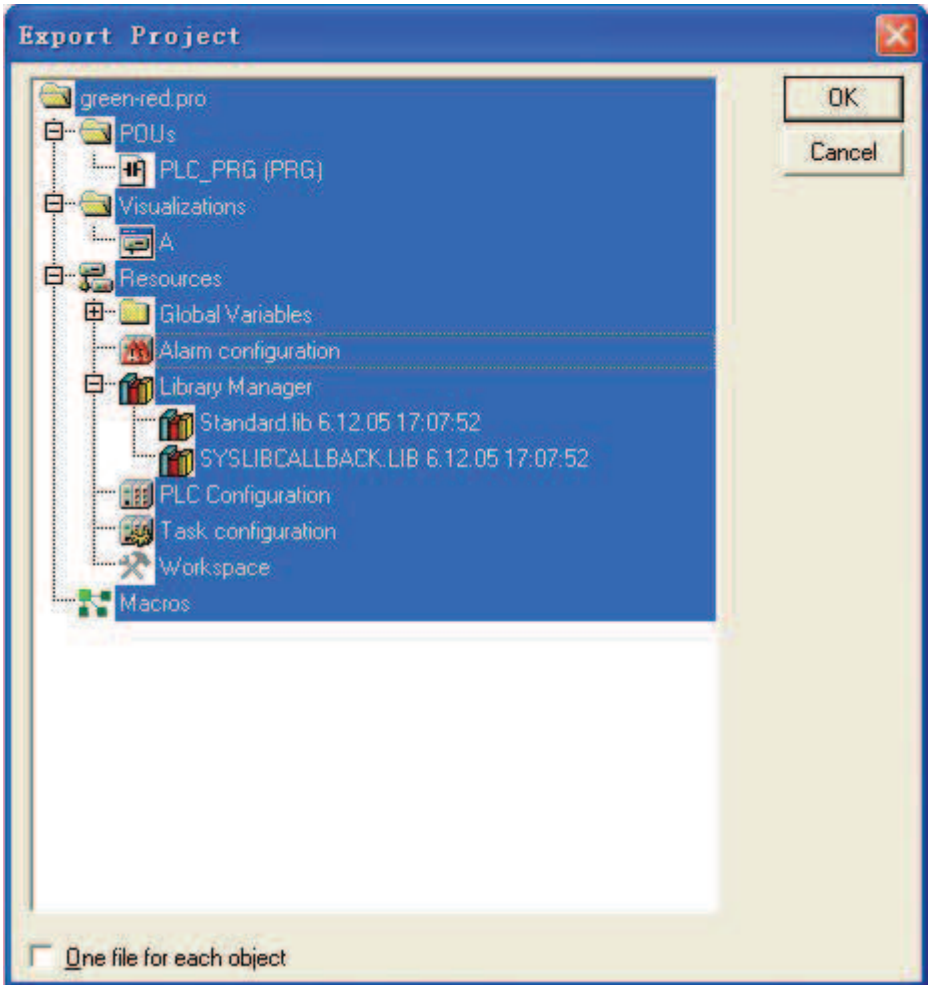
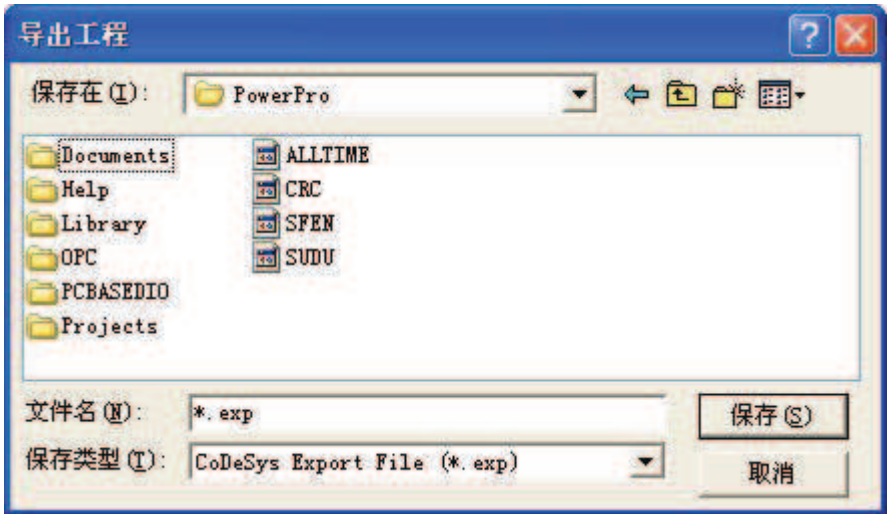


Figure 7-5-6 Export Project (1)

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



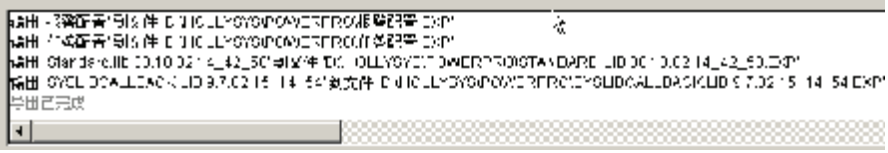


Figure 7-5-7 Export Project (2)

When import, select the desired export file and the object will imported into the current project. 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?”, shown in figure 7-5-8.

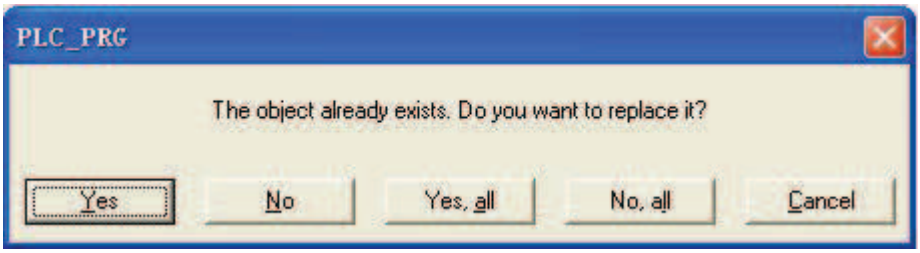


Figure 7-5-8 Tip for Replace

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

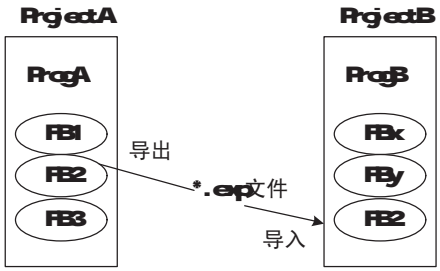


Figure 7-5-9 Schematic Diagram of Import and Export

7.5.3 Merge Projects

With this 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, 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?”

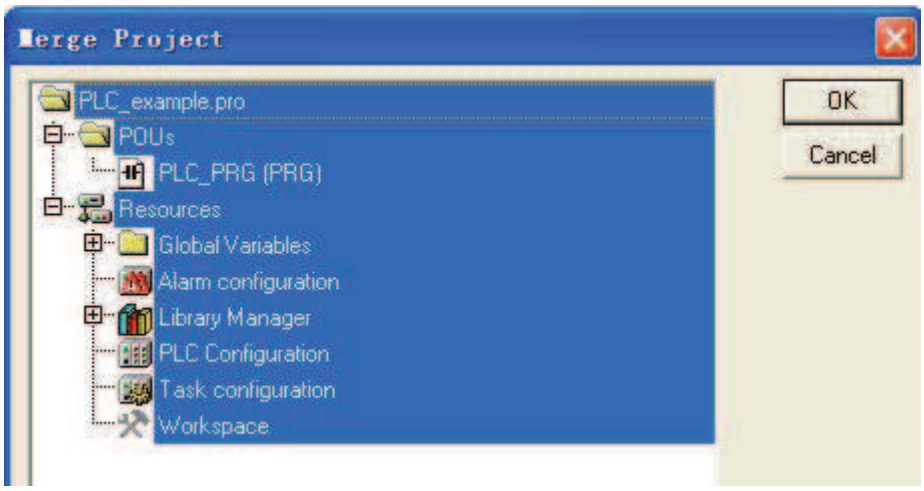


Figure 7-5-10 Merge Project

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

7.5.4 Compare Projects

With the command "Project"/"Compare" you can compare two projects. Click the command and a dialog box of project comparison appears, shown in figure 7-5-11.

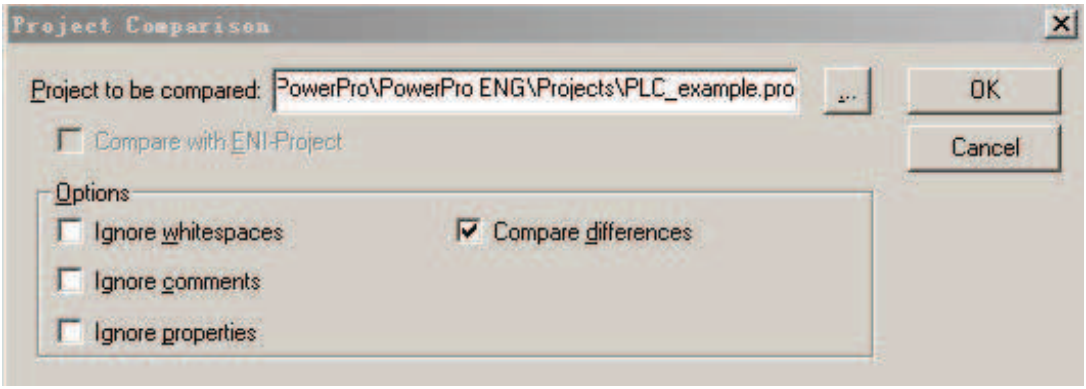


Figure 7-5-11 Compare Projects (1)

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

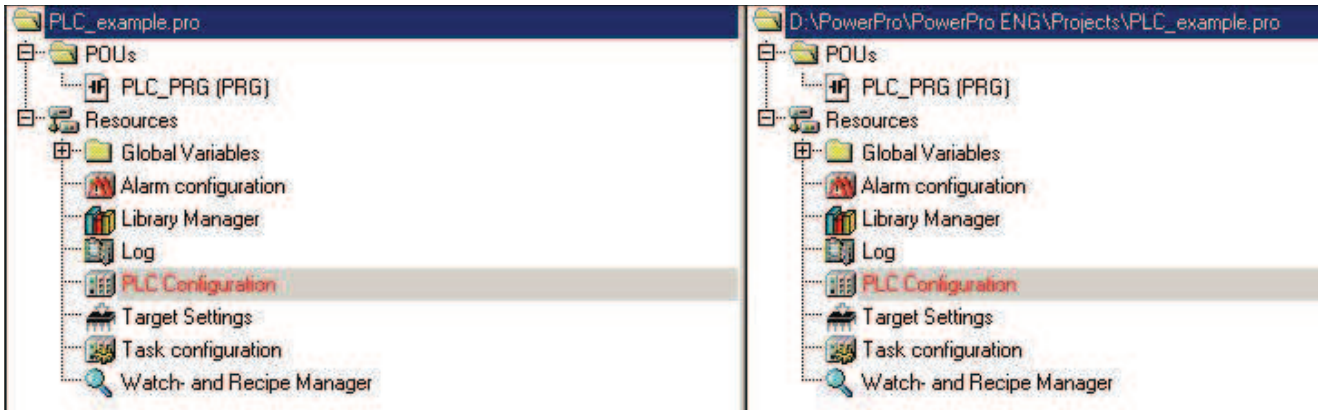


Figure 7-5-12 Compare Projects (2)

In addition, the command "Compare" can be used to compare the actual version of one project with that which was saved last, shown in figure 7-5-13. In the compare mode, objects can not be edited and all operations are forbidden until you close the window of project comparison.

However, the comparison of hardware configuration is not supported.

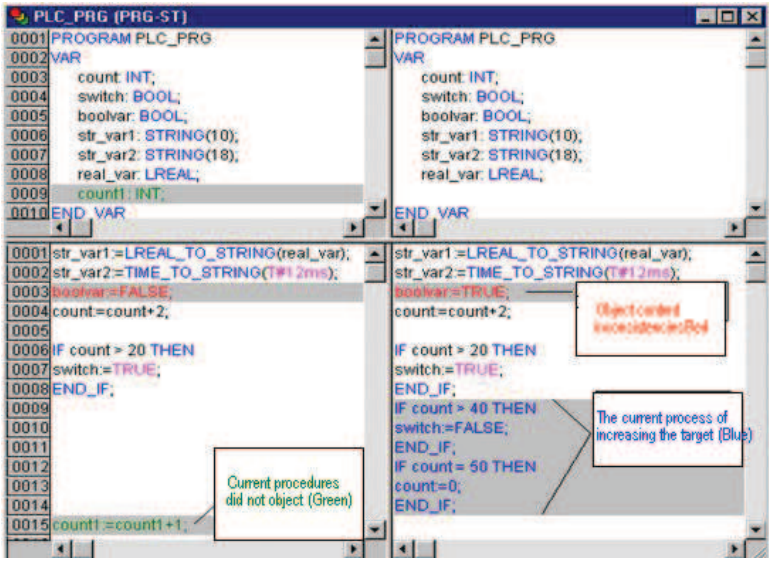


Figure 7-5-13 Cmpare Projects (3)

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, shown in figure 7-5-14. You can enter "Password" to open project, and "Write Protection Password" to change project.

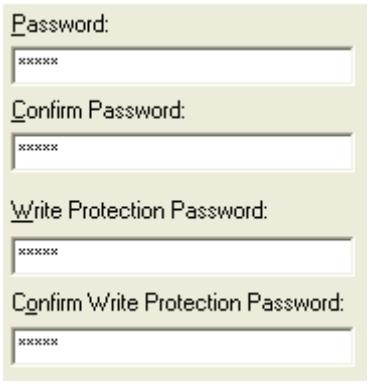


Figure 7-5-14 Options of Passwords

Enter the desired password in the field "Password". For each typed character an asterisk (*) appears in the field. You must repeat the same word in the field "Confirm Password". If you now save the file and then reopen it, then you get a dialog box to enter the password. If a dialog box appears, 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. For this simply press the button "Cancel", if 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 CoDeSys 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, 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 combobox "User Group" you can select the group and enter the desired password for the group in the field "Password", 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, 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.

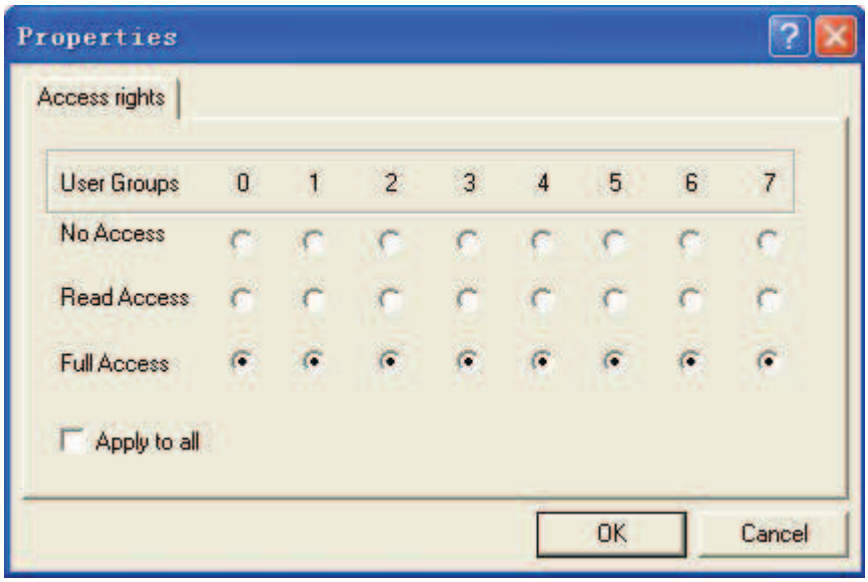


Figure 7-5-17 Properties

A simple example for object access rights is shown below.

With the command "Project"/"Object"/"Properties" open the dialog "Properties", 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.

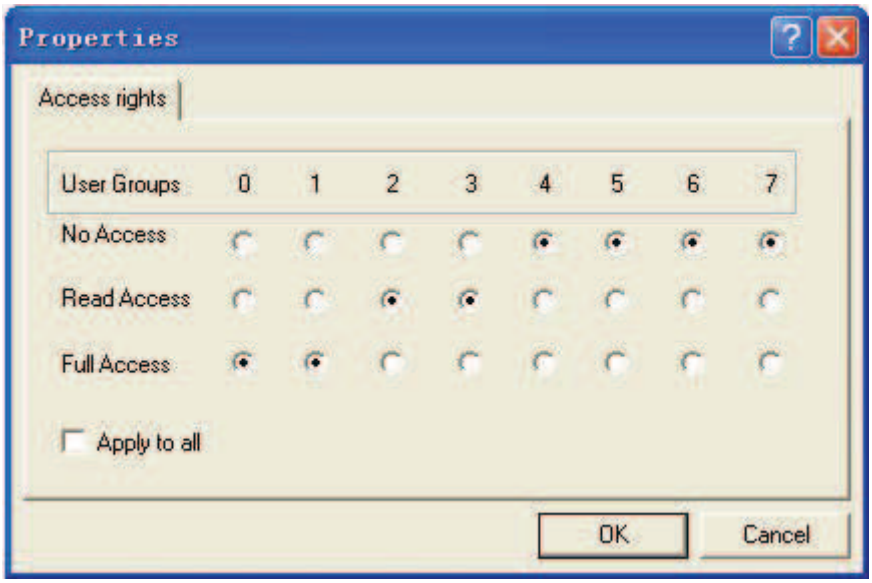


Figure 7-5-18 Access Rights

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

In the left combobox **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's the same for Level1, Level2 and Level3. The relationship between user group and password 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.

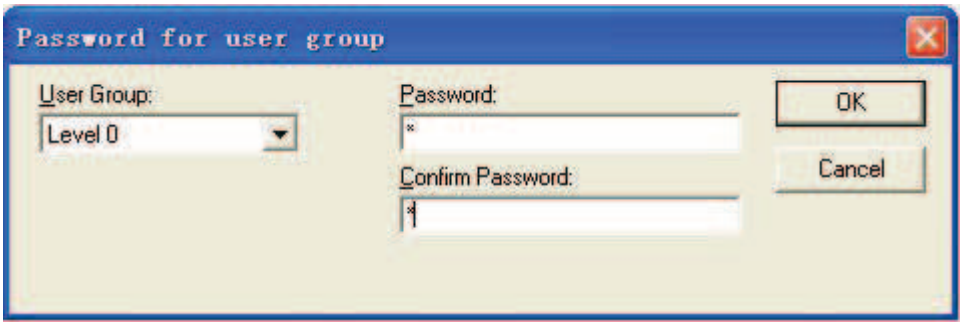


Figure 7-5-19 Password for user group

Table 7-5-1 User Group and Passwords

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

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. 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, shown in f igure 7-5-20.

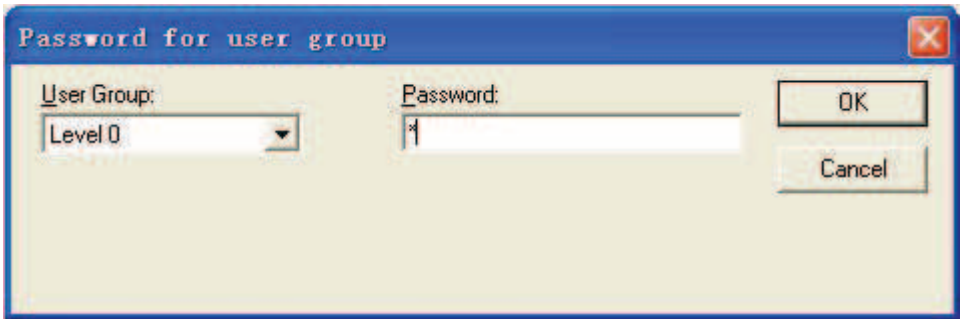


Figure 7-5-20 Enter the Password for user group

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

Figure 7-5-21 Password Error

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

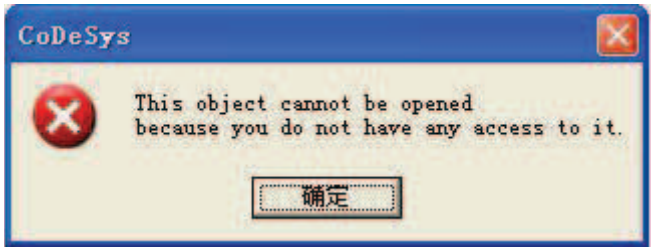


Figure 7-5-22 No Access to the Object

7.6 WORKSPACE SETTINGS

You can open “Workspace” in two ways: first, click “Project”/“Options”; second, double click “Workspace” in register card “Resources”, shown in figure 7-6-1.

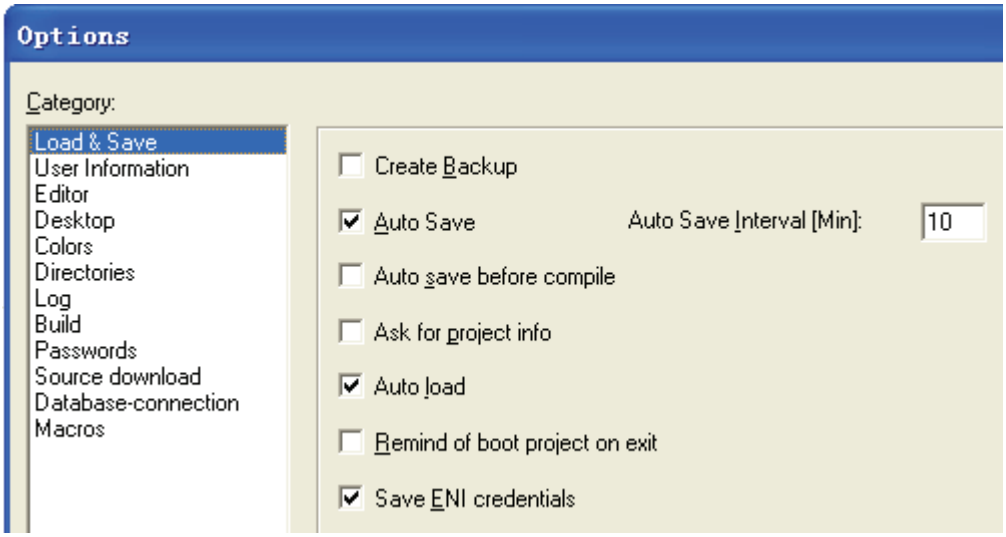


Figure 7-6-1 Workspace Options

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 of a mouse click or using the arrow keys and change the options on the right side.

- Load&Save
- User Infomation
- 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 Infomation", "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, 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 CoDeSys is not shut down "normally" (e.g. due to a power failure), then the file will not get erased. When you open the file again the following message appears, shown in figure 7-6-2. You can now decide whether you want to open the original file or the auto save file.

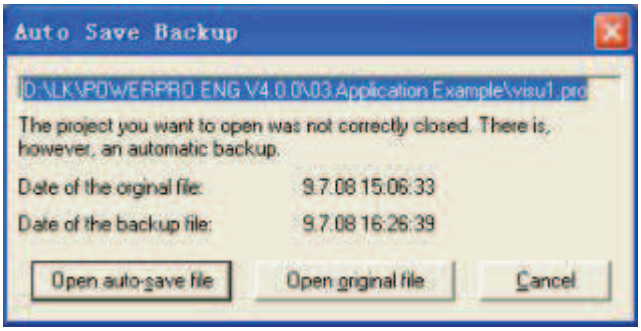


Figure 7-6-2 Auto Save Backup

- "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", "Version" and so on, which are part of 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? ", 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 appear at the right of window. The options of Editor include Autodeclaration, Autoformat, List components, Declarations as tables, Mark, Bit values and so on, shown in figure 7-6-5.

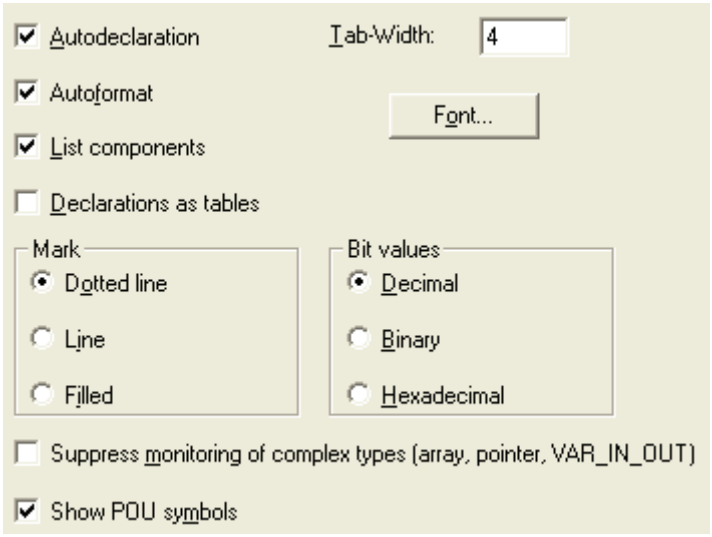


Figure 7-6-5 Editor Options

- "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; 2. Tabs are inserted to that the columns are uniformly divided.
- "Declarations as tables ": If this option is activated, then 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 type and name of POUs automatically.

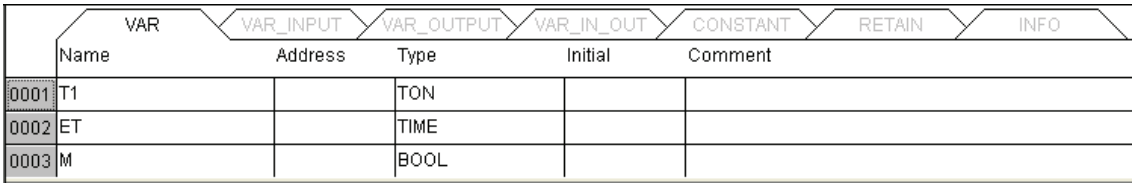


Figure 7-6-6 Declarations as Tables

- "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 upon the font which is chosen.
- "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. The choice of a larger font size thus enlarges the printout, even with each 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 can 't support the 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 at the right of window, shown in figure 7-6-7

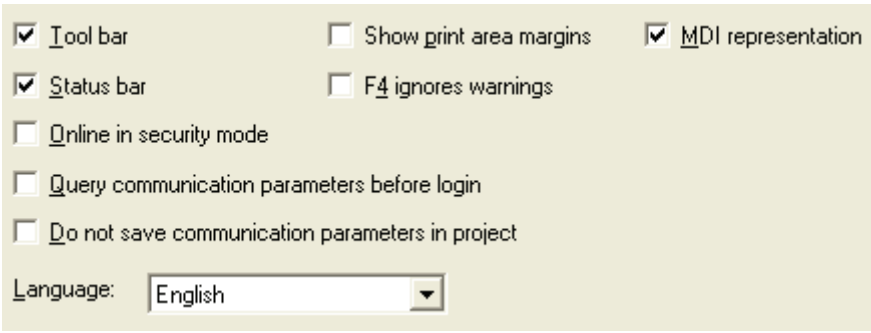


Figure 7-6-7 Desktop Options

- “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 at the lower edge of the main window 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. For ding this can avoid misoperation.
- “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. For doing this no need to execute “Online”/“Communication Parameters”.
- “Language”: Choose the language for the menu and dialog texts, and the default setting is English.
- “Show print area margins ”: In every editor window, the limits of the currently set print range are marked with red dashed lines. Their size depends on the printer characteristics (paper size, orientation) and on the size of the “Content” field of the set print layout (menu: “File” “Documentation Settings”), 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 appear at the right of the window, shown in figure 7-6-9.

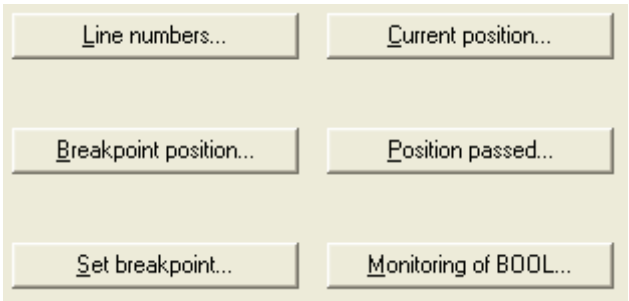


Figure 7-6-9 Colors Options

You can re-set some specific display colors according the user ’s requirement and custom.

According to the colors you can observe the program running situation 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 light gray, background color of network numbers or line numbers in editors.
- "Current position": default red, in online mode the background color of network numbers or line numbers where the program stops at a breakpoint.
- "Breakpoint position": default dark gray, the background color of network numbers or line numbers where you can set a breakpoint.
- "Position passed ": default green, the background color of network numbers or line numbers which have been executed in flow control.
- "Set breakpoint": default light blue, the background color of network numbers or line numbers where you have set a breakpoint.
- "Monitoring of BOOL ": default dark blue, in online mode the color of digital logic TRUE.

7.6.6 Directories

In "Workspace", select "Directories", the options appear at the right of the window, 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 can 't 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".

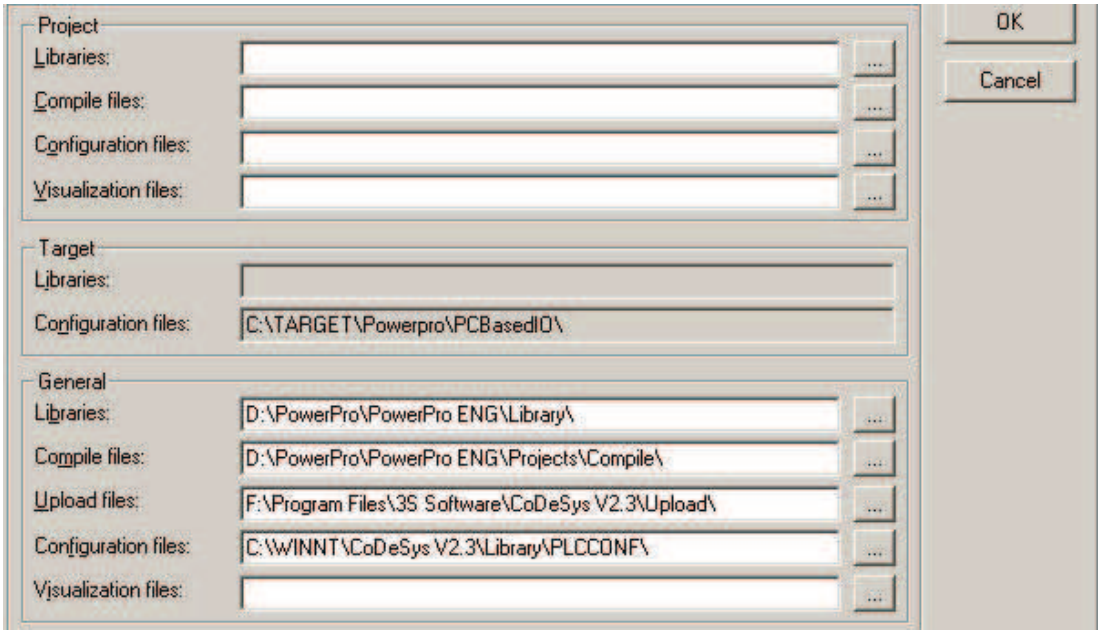


Figure 7-6-10 Directories

7.67 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, shown in figure 7-6-11.

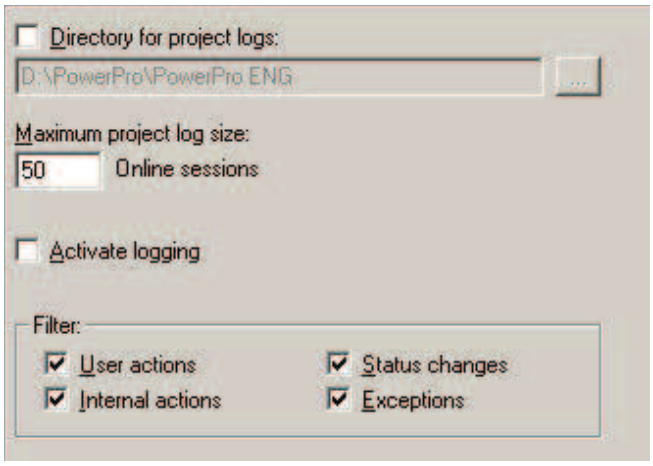


Figure 7-6-11 Log Options

- “Directory for project logs ”: You can change the directory where the project logs are saved and the default setting is D:\PowerPro\PowerPro ENG.
- “Maximum project log size ”: The maximum number of ‘Online sessions’ shown in log window.
- “Activate logging”: Activate log function to display log list, 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.

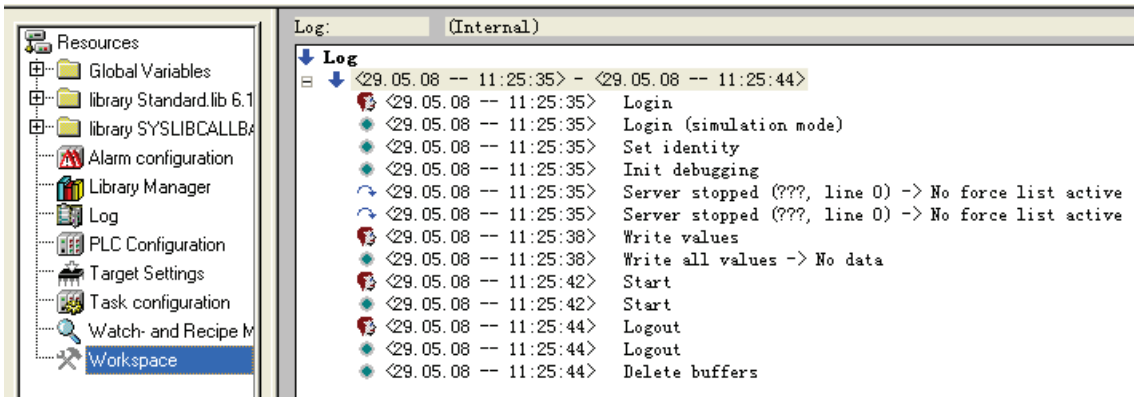


Figure 7-6-12 Log List

- “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, shown in figure 7-6-12.

7.6.8 Build

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

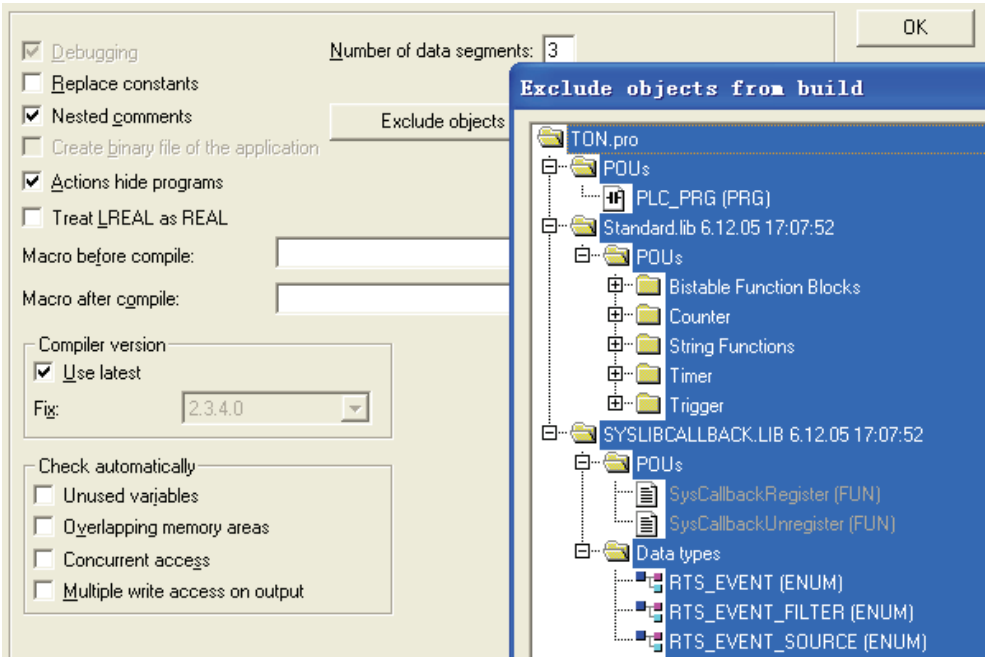


Figure 7-6-13 Build

- “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 is then 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 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 in allocation of variables via the "AT" declaration overlaps have arisen 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, shown in figure 7-6-14.

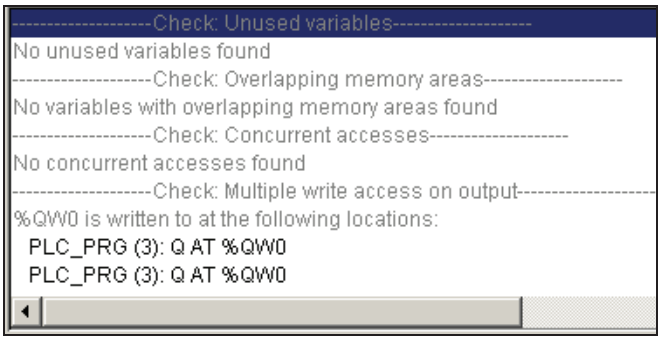


Figure 7-6-14 Compile Information

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 codes are set and how to use it will be introduced as follows.

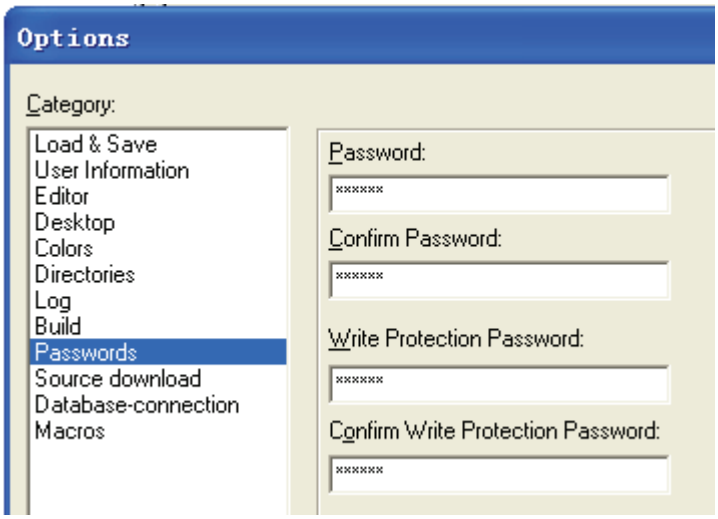


Figure 7-6-15 Passwords

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, shown in figure 7-6-16.

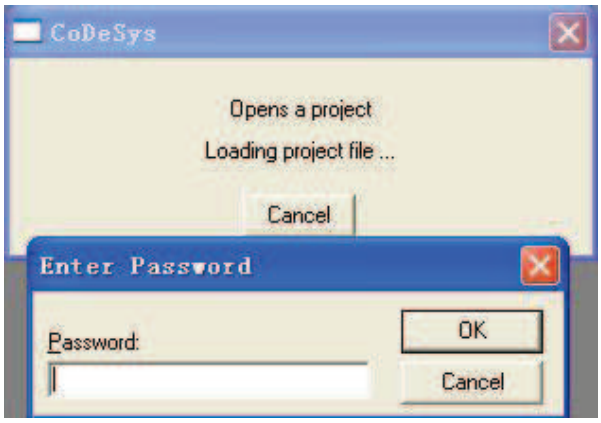


Figure 7-6-16 Enter Password

Click “OK”, a dialog box appears in which you are requested to enter the write protection password, 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. Then if you open the program again, the password protection is invalid.

CHAPTER 8 BUILD AND DEBUG

Compile the program when the program is finished and download it to PLC only when the program has been compiled without any error. In this chapter the compilation and downlad are introduced mainly.

8.1 BUILD

The commands “Project”/ “Build” and “Rebuild all” in PowerPro are used to check semantic correctness, shown in figure 8-1-1.

- Build: Build the program which has changed and renew it to thr 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 can’t support the function.

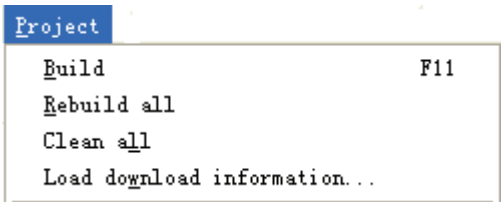


Figure 8-1-1 Project List

The program without syntactic error can generate executive target file. The compilation result will be displayed in message window, shown in figure 8-1-2.

The common compilation errors:

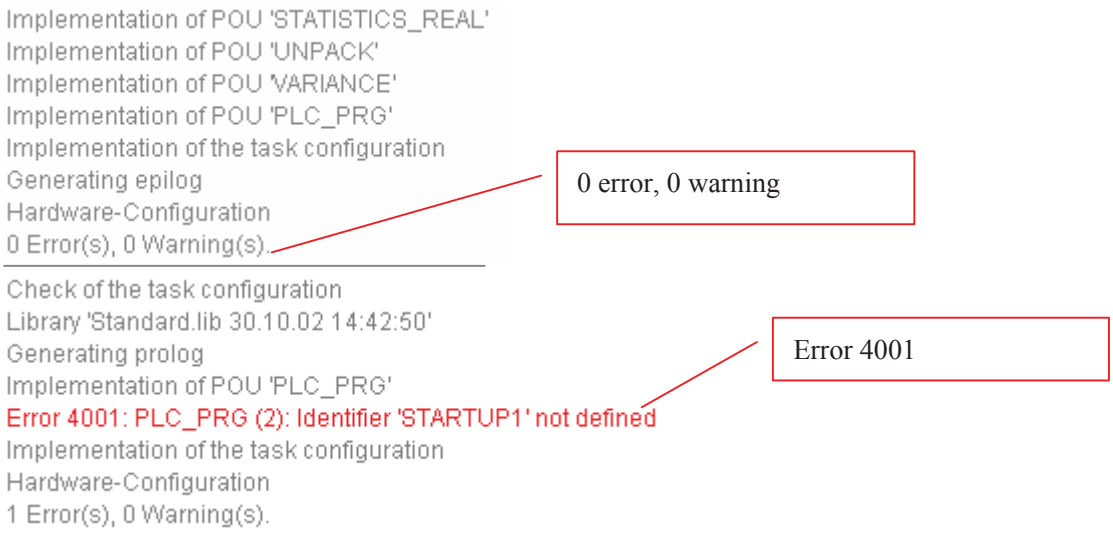


Figure 8-1-2 Compilation Message

8.2 SHOW 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 Show 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, shown in figure 8-2-1. Before this the project must have been compiled without any error.

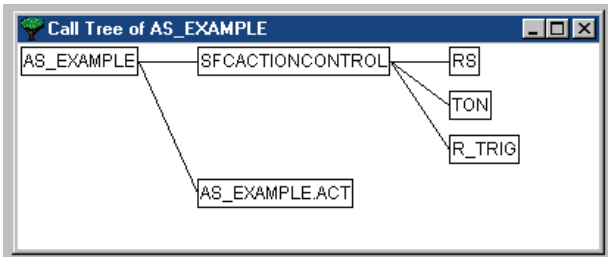


Figure 8-2-1 Show Call Tree

8.2.2 Show Cross References

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

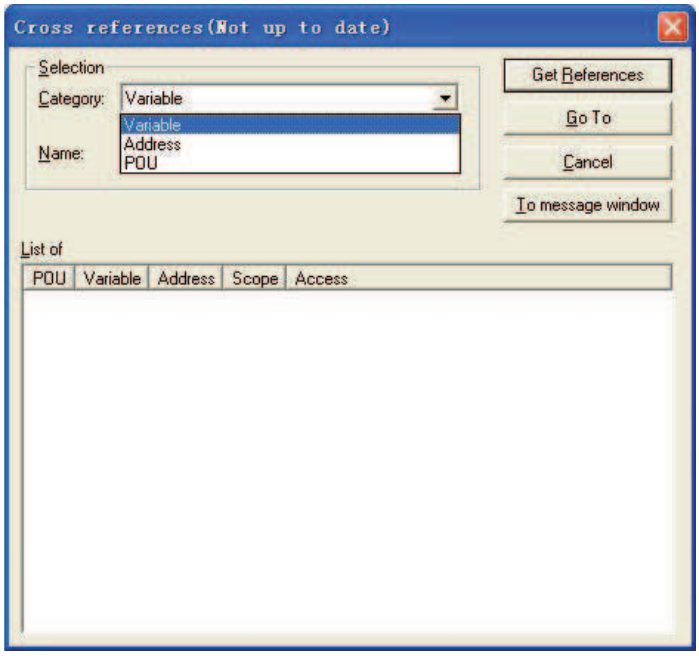


Figure 8-2-2 Show Cross References

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 doubleclick 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.

823Check

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

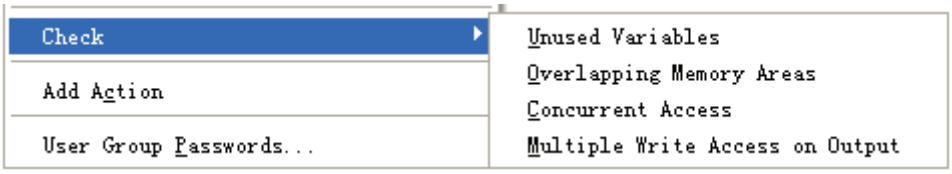


Figure 8-2-3 Check Menu

- "Check"/"Unused Variables"

When write a program, you usually delete a variable or rename it, and then the original variable declaration can 't be deleted automatically and remains in 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 the project must have been compiled without any error. Results are displayed in the message window, shown in figure 8-2-4.

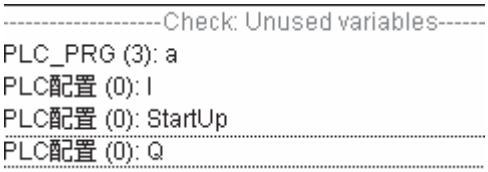


Figure 8-2-4 Check Unused Variables

- "Check"/"Overlapping Memory Areas"

This function menu tests whether in allocation of variables via the "AT" declaration overlaps have arisen at specific memory areas. If there is no overlapping memory area, a tip displays in the message window: 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 Acess 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 acess on output, a tip displays in the message window: No outputs found which are written to at more than one location.

8.3 DOWNLOAD

8.3.1 Device Installation and Connection

➢ Install Device

First according to the actual project select appropriate CPU modules and expansion modules. And then 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.

➢ Connect Cable

According to CPU model number and type connect power line, 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. Regard that when the power line is connected put on the terminal cprver to avoid unnecessary personal injury and device damage.

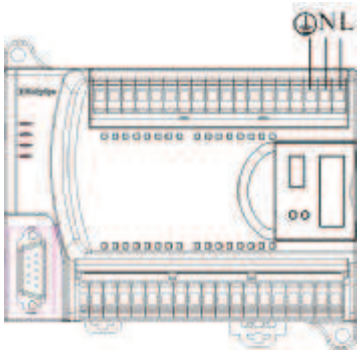


Figure 8-3-1 Connect Power Supply Line

➢ Establish PC Communication

Through the related programming cable connect CPU module to RS232 serial port of local PC to establish communication channel, shown in figure 8-3-2. Because CPU RS232 serial port is not isolated, so 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

832Establish 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" appears, shown in figure 8-3-3.

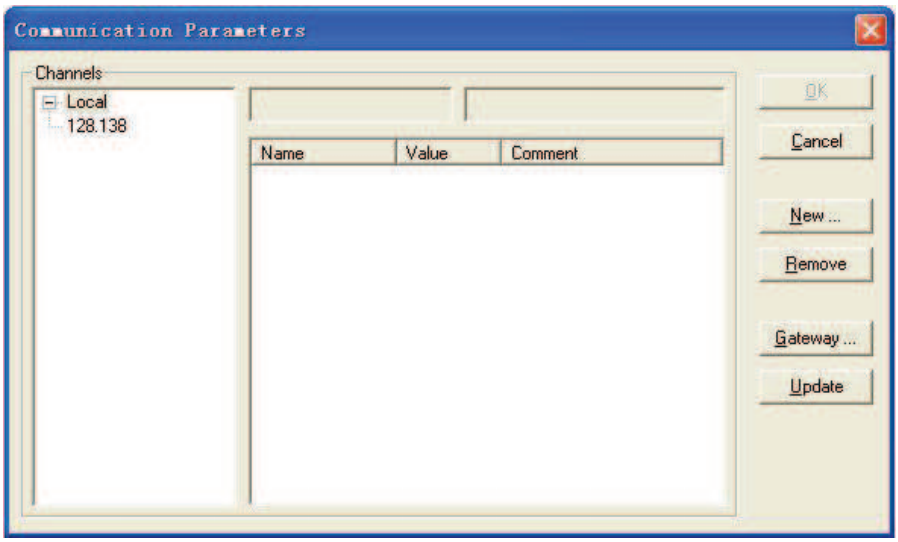


Figure 8-3-3 Dialog Box of Communication Parameters

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

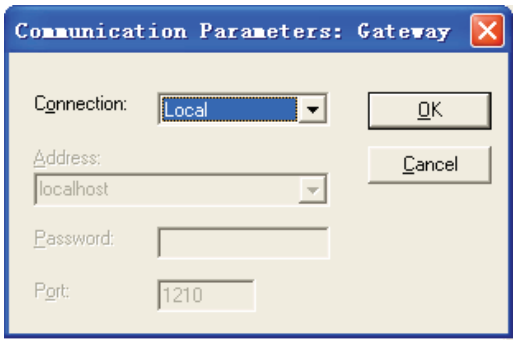


Figure 8-3-4 Communication Parameters: Gateway

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, 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".

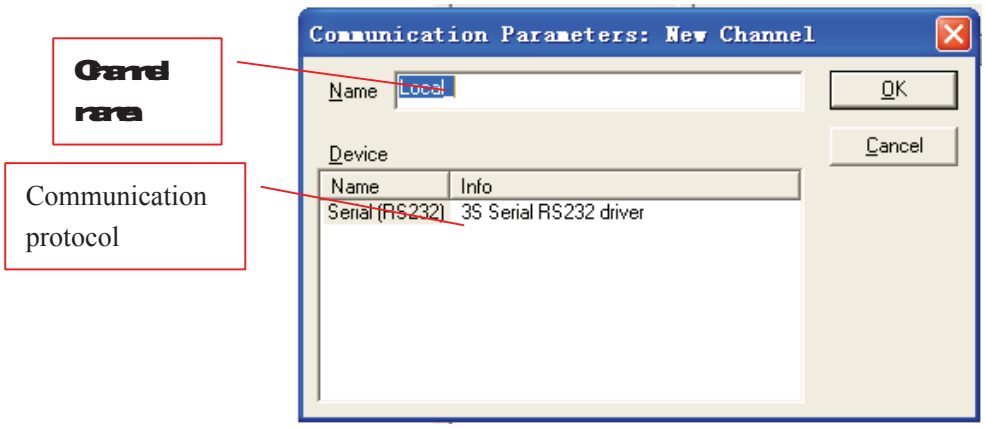


Figure 8-3-5 New Channel

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

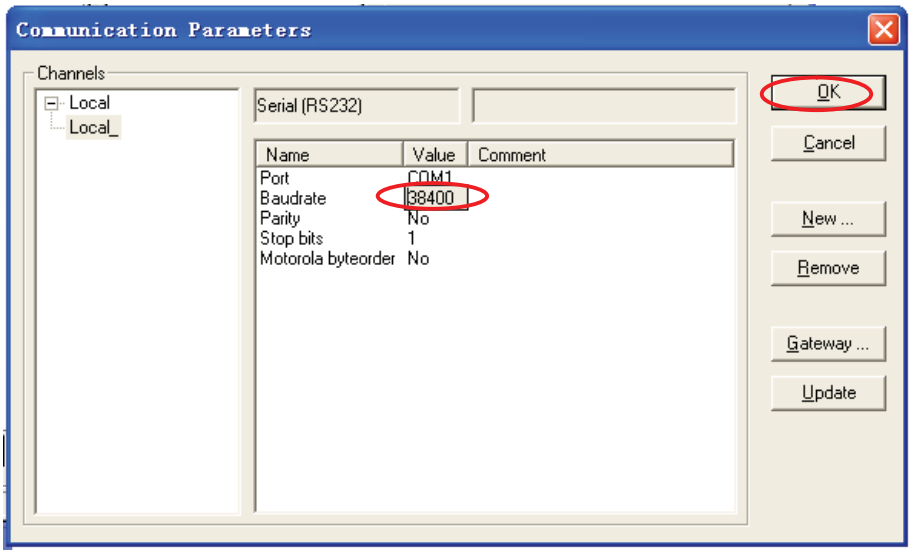


Figure 8-3-6 Setting of Baudrate

8.3.3 Program Download

- Program Download

With the command "Online"/ "Login" download the program to PLC. All the target files are downloaded to the module when download the target file generated while compilation. At the same tome the module is reset and all variables return to initial status. With the command "Online"/ "Login" establish the connection between local PC and CPU module and a tip message appears, shown in figure 8-3-7.

Figure 8-3-7 Download Information

When the download PLC program is not the same with the PLC internal program the dialog above appears.

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, shown in figure 8-3-8. A boot project is a program downloaded to Flash. A boot project will be created in FLAH in order to protect the program from losing when power-off and repower-on.

Click the "Yes" button in figure 8-3-8 to run the download program when power-off and repower-on. Now the download is finished.

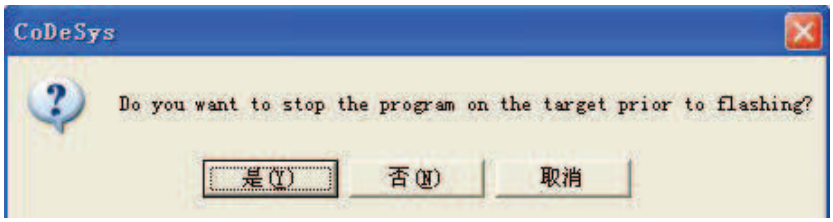


Figure 8-3-8 Create Boot Project Message

- Not download the program and enter monitoring status

If you don't 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 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 's 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 doanload to CPU is finished, a dialog box of "Stop target program before creating a boot project"appears, 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 DEBUG

All the debugging commands provided by system are under the "Online" menu and are available in debugging mode, shown in figure 8-4-1. In debugging mode the different detault 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 usful to know the meanings of the colors for debugging and monitoring programs.

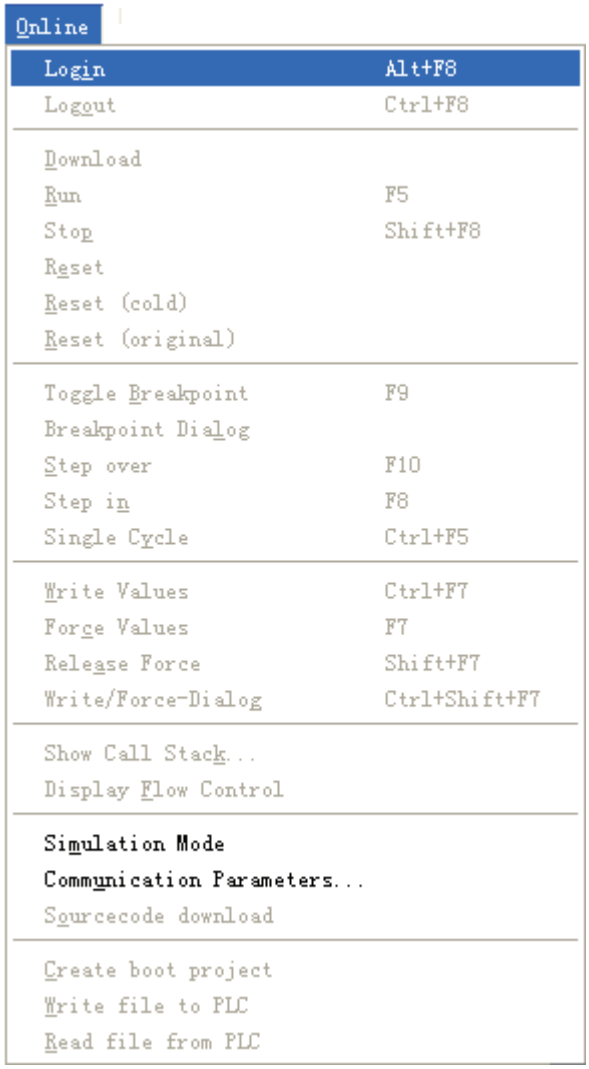


Figure 8-4-1 Online Menu

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 ("√") 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 can ’t 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, can ’t be used in simulation mode.

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

842 “Online”/“Logout”

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

843 “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.

844 “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 re-start the program with the command “Online”/“Run” at the position where it stops at the last time.

845 Reset

If you have initialized the variables with a specific value, 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 besides of the persistent variables. Press“F5” to restart the program.
- The command “Reset (original) ” resets all variables including retain variables and persistent variables to their initialization 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, shown infigure 8-4-2.

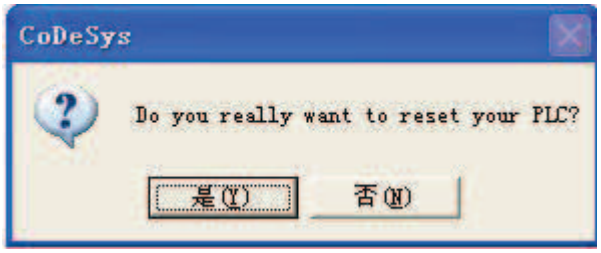


Figure 8-4-2 Reset Confirmation

8.4.6 Breakpoint

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.

- Set 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 gray background color, shown in figure 8-4-3.

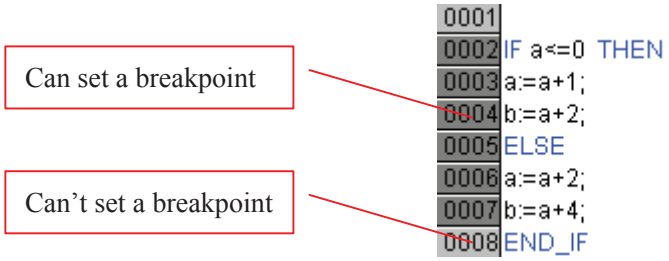


Figure 8-4-3 Breakpoint Setting (1)

In IL and ST the breakpoint 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, shown in figure 8-4-4.

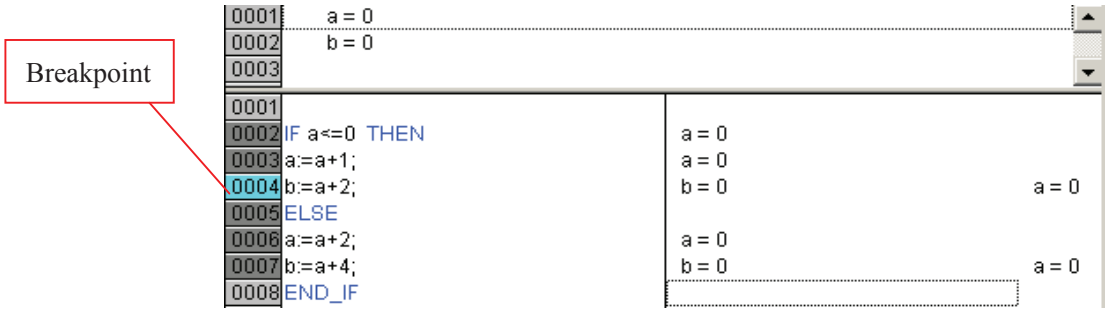


Figure 8-4-4 Breakpoint Setting (2)

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

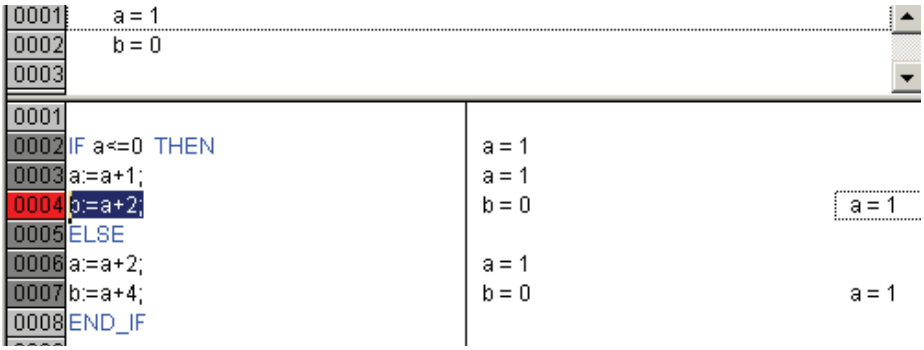


Figure 8-4-5 Breakpoint Setting (3)

➢ Edit Breakpoints

With the command "Online"/"Breakpoint Dialog" open a dialog box to display and edit breakpoints throughout the entire project, 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.

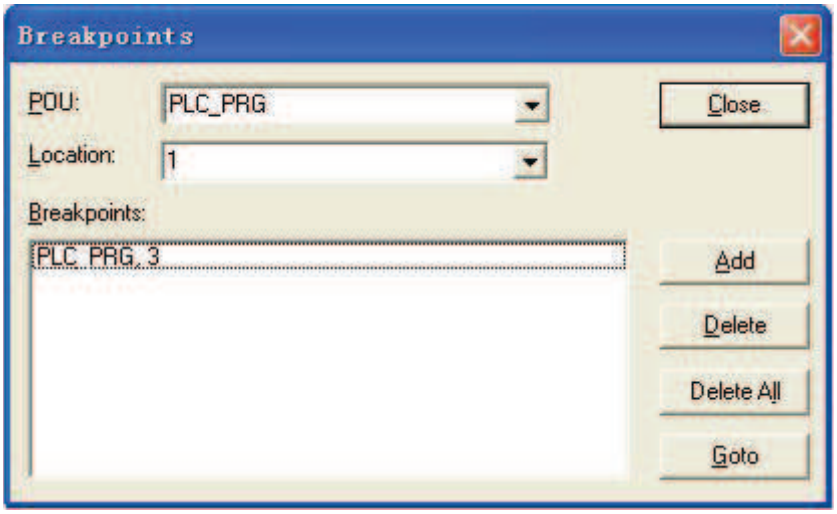


Figure 8-4-6 Breakpoint Setting (4)

All breakpoints are deleted after executing the command "Online"/"Logout". Regard that the number of breakpoints should be less than 100, or else an error appears, shown in figure 8-4-7.

Figure 8-4-7 Breakpoint Setting (5)

847 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 then the program stops after the implementation of the current instruction. When a function or a function block are reached, with the command "Step over" jump to the next instruction while with the command "Step in" move to the first instruction of a called function or function block.

848 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 circularly until the command"Stop".

8.4.9 Write Values

- Set new values

In debugging status double click on the line in which a variable is declared, a dialog box appears 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.

- Active new values

With the command "Online"/"Write Values" or shortcut "Ctrl+F7" 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 Force Values

- Force Values

It's the same with "Write Values"for setting and active 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, then all variables will be set to the selected values after each cycle until the 'Release force' is given. For the command "Write Values" the variables are set to the selected values once and can be assigned by other programs.

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

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

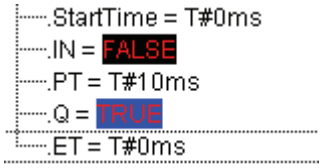
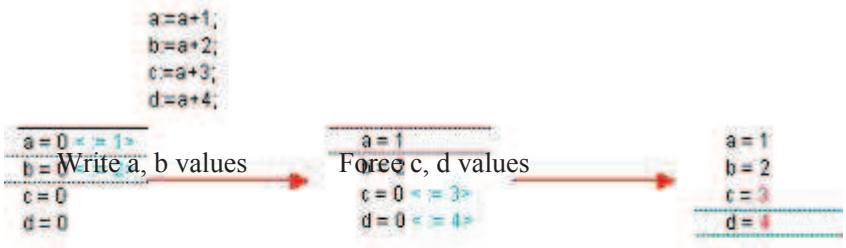


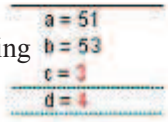
Figure 8-4-9 Force Values

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

Main program:



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.

- Release 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

With the command "Online"/"Write/Force-Dialog" a dialog box appearswhich displays in tworegisters the current writelist and forcelist, shown in figure 8-4-10

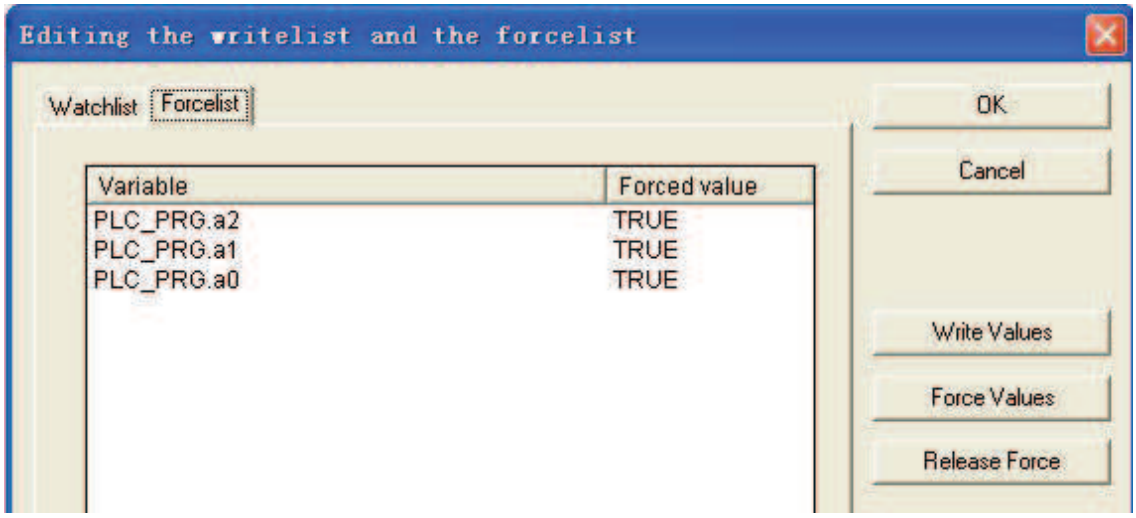


Figure 8-4-10 Write/Force-Dialog

8411 Show Call Stack

With the command "Online"/"Show Call Stack" the program will stop at a breakpoint and a dialog box with a list of the POU call stack appears, 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

8412 Display 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 curret flow of POU clearly.

A simple example is shown in figure 8-4-12, a is a 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, shown in figure 8-4-13.

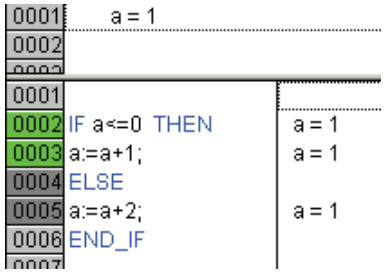


Figure 8-4-13 Example (2)

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

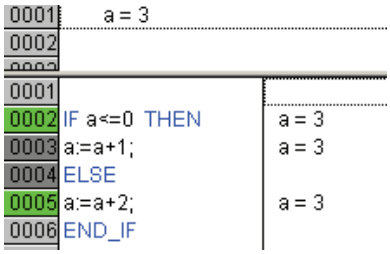


Figure 8-4-14 Example (3)

8413 Watch- and Recipe Manager

In the register card "Resources" in object organizer, double click "Watch- and Recipe Manager" and a window appears, and 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".

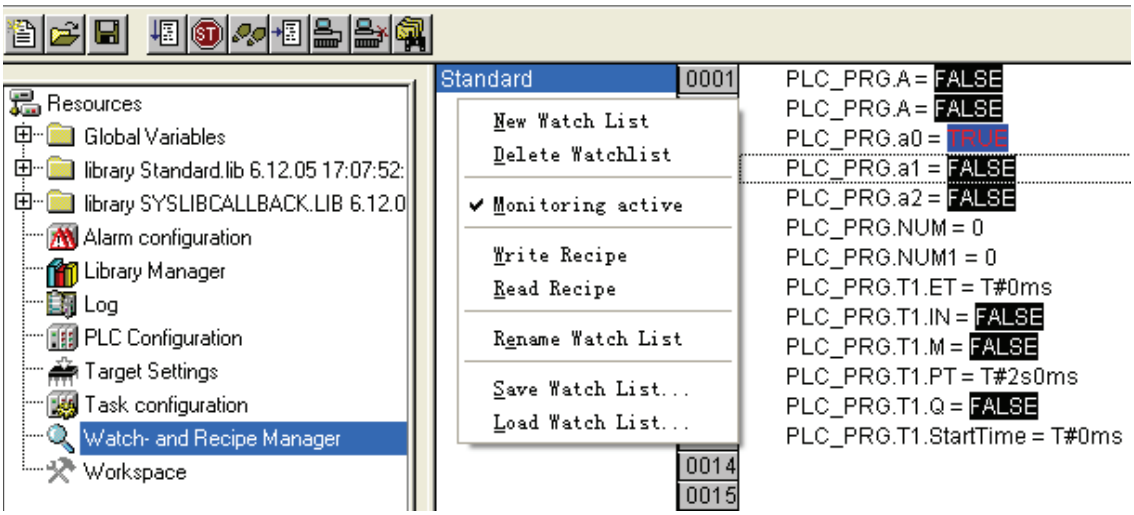


Figure 8-4-15 Watch- and Recipe Manager

- New Watch List

Click the right mouse key at the lelf 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".

- Rename Watch List

With the command, you can change the name of watch list.

- Select 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.

- Save Watch List

With this command you can save a watch list. The file name is preset with the name of the watch list and is given the extension "*.wtc".

- Load Watch List

With this command 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.

- Write Recipe

With this command you can write the preset values into the variables.

- Read Recipe

With the command, you can replace the presetting of the variables with the present value of the variables.

CHAPTER 9 BASIC KNOWLEDGE FOR IEC PROGRAMMING

PowerPro conform 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 introduced in this chapter.

9.1 FUNCTION BLOCK DIAGRAM (FBD)

FBD is a graphically oriented programming language, similar with LD. FBD works with a alist 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. As shown in figure 9-1-1.

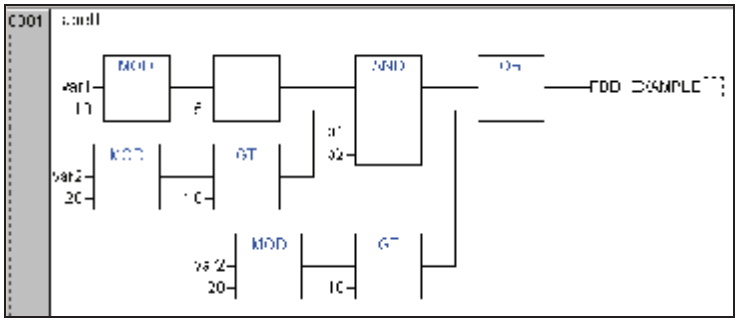
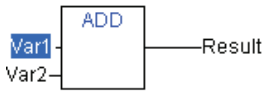


Figure 9-1-1 Language of FBD

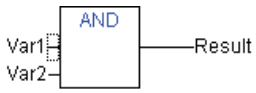
9.1.1 Cursor Position

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

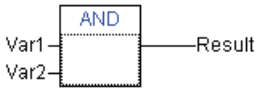
- Text (Cursor Position 1) :



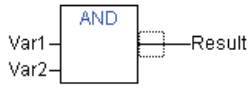
- Input (Cursor Position 2) :



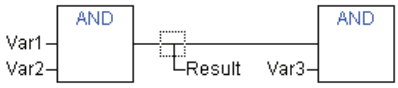
- Operator, Function or Function Block (Cursor Position 3) :



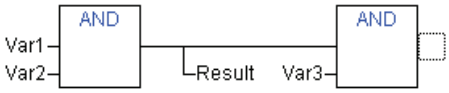
- Output (Cursor Position 4, an assignment or a jump comes afterward) :



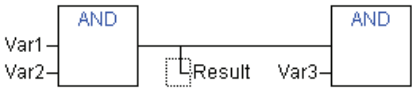
- The lined cross above an assignment, a jump or a return instruction (Cursor Position 5) :



- 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 's needed 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 the program, shown in figure 9-1-2.

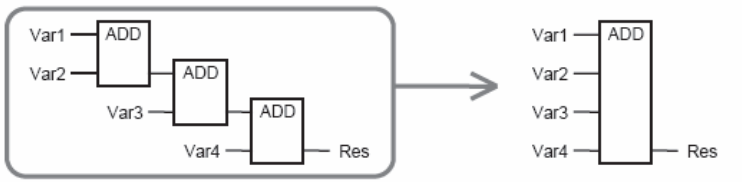


Figure 9-1-2 Extend Inputs of Operators

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), then 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), then 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, then the unattachable branches will be deleted. The same holds true for the outputs.

If a jump or a return is selected, then 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), then 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, it is always inserted an “AND” operator. This can be converted by Selection and Overwrite of the type text (“AND”) into every other operator, into every function, into 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 to 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

POU which is calling.

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:  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:  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], 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.

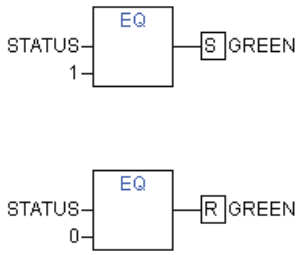


Figure 9-1-3 Set and Reset

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

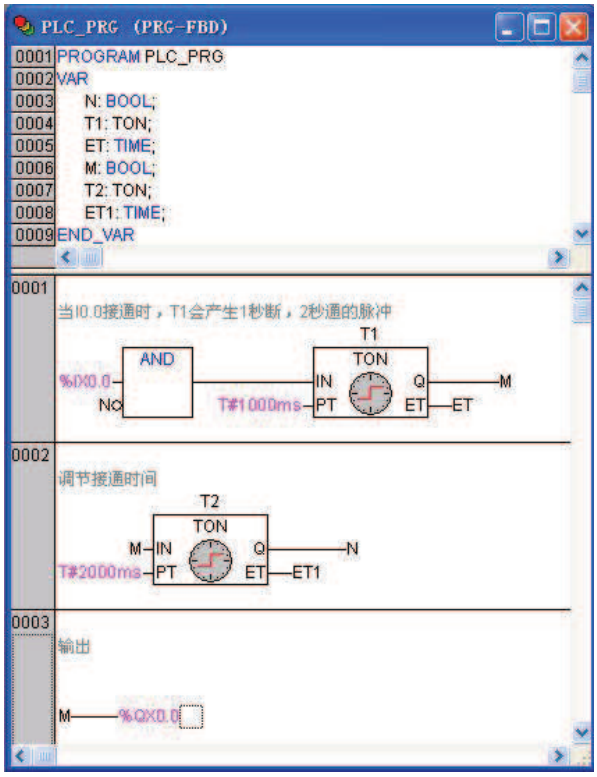


Figure 9-1-4 Example in FBD Language

92INSTRUCTION LIST (IL)

Instruction List (IL) is a programming language in the style of assembly language and difficult to read but it executes fastest. IL consists of a series of instructions. Each instruction begins in 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.

921 Operation Description

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

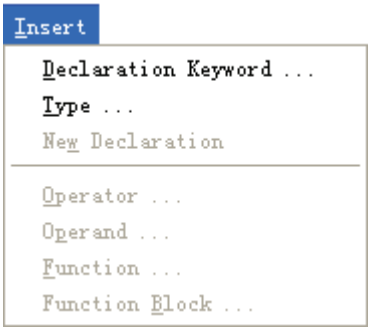


Figure 9-2-1 "Insert" Menu

➢ “Insert”/“Operator”

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

➢ “Insert”/“Operand”

Click the command at any position in programming area, a dialog appears and select the operand, and click “ok”.

➢ “Insert”/“Function”

Click the command at any position in programming area, a dialog appears and select the function, and click “ok”.

➢ “Insert”/“Function Block”

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

9.2.2 Example of Program

✧ Example

The following is a simple program in 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 then executed if the result of the preceding expression is TRUE. N with JMP, CAL, RET: the instruction is only then 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.

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

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

✧ 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 then exexuted 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.

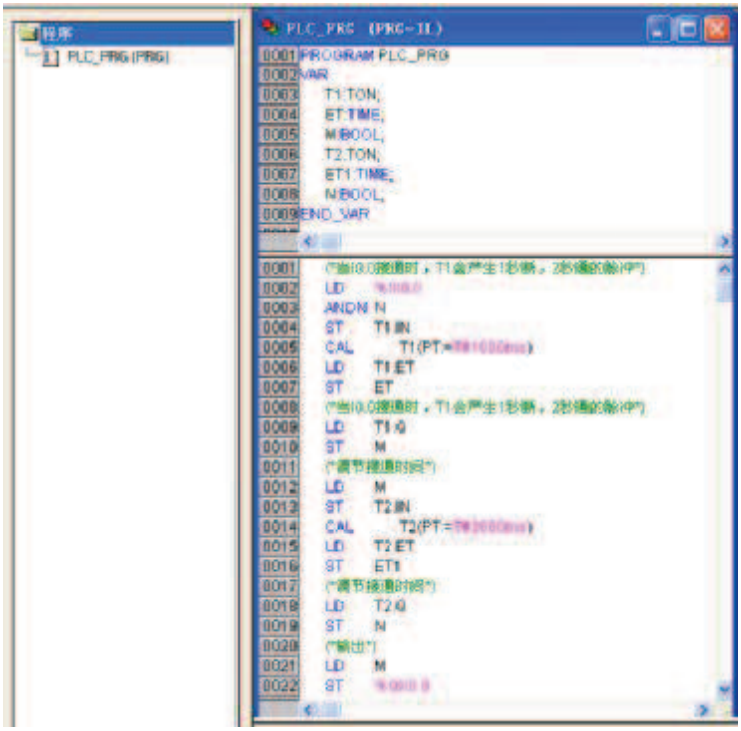


Figure 9-2-2 Example in IL Language

93 STRUCTURED TEXT (ST)

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

931 ST Expression

Expressions in ST are composed of operators and operands. An opernd 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 strongest pripory is processed first, then the operator with the next strongest pripory, etc., until all operators have been processed. Operators with equal pripory are processed from left to right. The operators in ST are listed in table 9-3-1.

Table 9-3-1 ST Operators

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

9.3.2 ST Instruction

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

Table 9-3-2 Instrcutions in ST

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	;

- 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, then 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. Then 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, then 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, then the <ELSE instruction >is executed.

If the same instruction is to be executed for several values of the variables, then 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 separated 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 stepsize 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 uou indicate a acondition which, when it's fulfilled, the loop will be executed.

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, then the <Instructions> are 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 onlyafter 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, then the <Instructions> are executed only once. If <Boolean expression> never assume the value TRUE, then the <Instructions> are 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 realized.

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 sequency of processing is controlled by transition

elements.

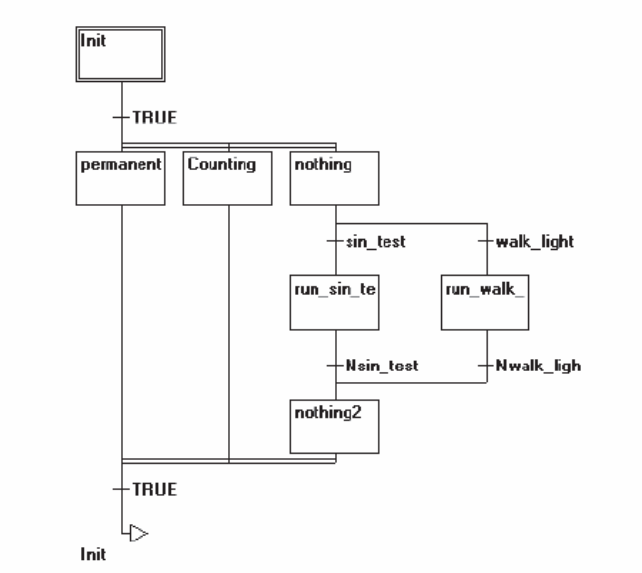


Figure 9-4-1 Language of Sequential Function Chart

941 Basic Concepts

- Step

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 shows 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. The new 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.

- Action

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 with the mouse 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.

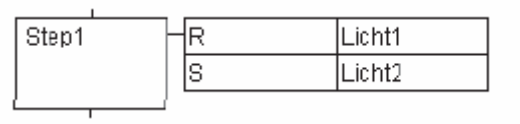


Figure 9-4-2 Example in SFC

- Entry Action and Exit Action

You can add an entery action and an exit action to a step by the commands "Add Entry-Action" and "Add Exit-Action".

An entry action is executed only once right after the step has become active. An exit action is executed only once before the step is deactivated. A step with 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 oe exit action, double click in the corresponding corner in the step with the mouse.

An example of a step with entry and exit action is shown in figure 9-4-3.

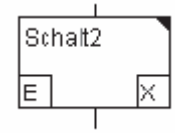


Figure 9-4-3 Example of a Step with Entry and Exit Action

- Transition/ Transition condition

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 a exit action then execute it and if there is a entry action then execute it. And then execute all the actions of the active step according to control cycle.

A transition condition can consist of a Boolean variable, a Boolean address, a Boolean constant or a Boolean result written in other programming languages.

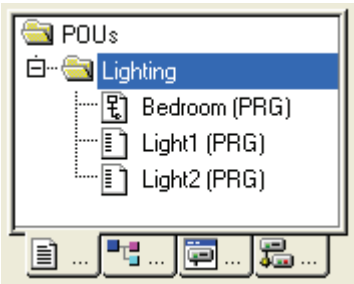
- Active step

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 cconditions of the following steps are TRUE. The currently active steps will be executed in the next cycle.

- Qualifier

In order to associate the actions with IEC steps, qulifiers 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.



elements.

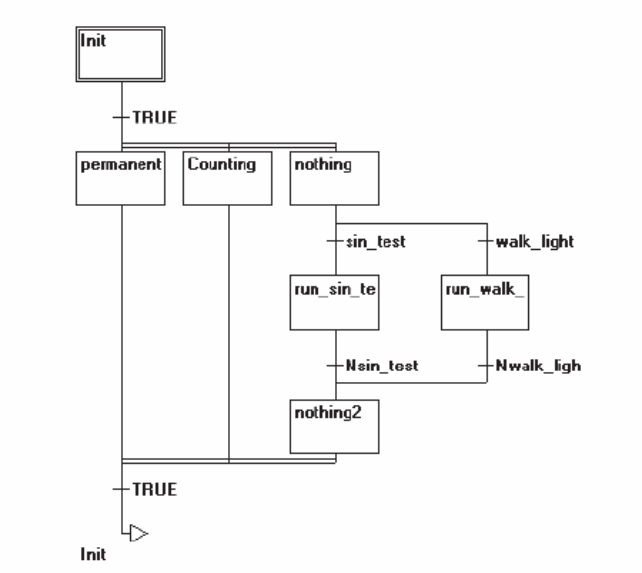


Figure 9-4-1 Language of Sequential Function Chart

941 Basic Concepts

- Step

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 shows 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. The new 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.

- Action

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 with the mouse 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.

		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.	

➢ 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.

One can make an enquiry with trhe variable <ActionName>.x as to whether an IEC action isactive 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 inpus or outputs.

SFCEnableLimit: The variable is of the type BOOL. When it has the value TRUE, the timeoutsof 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 Description

- Select Elements

You can select an element by pointing the mouse on this element and pressing the left mouse button, or you can use the arrow keys. Press <Shift>to mark a group of several elements. Please regard, that a step can only be deleted together with the preceeding 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. For 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. For 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.

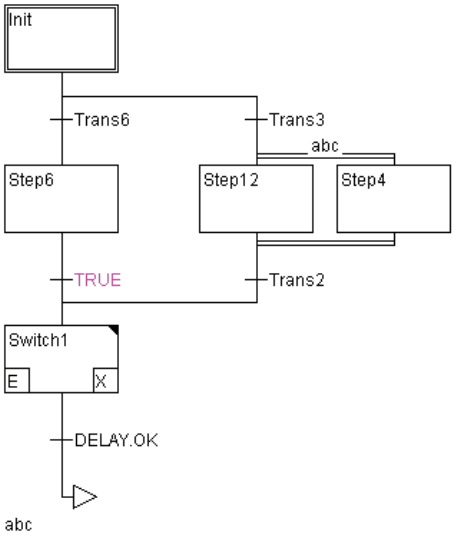


Figure 9-4-5 Add Label to Parallel Branch

- "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.

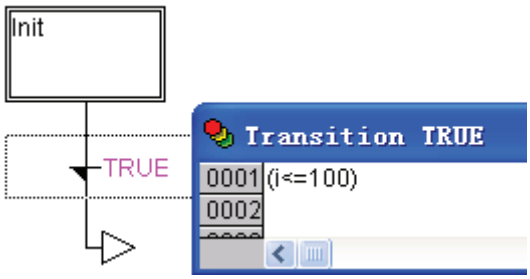


Figure 9-4-6 Priority of Transition Condition

- "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 Branch

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, then 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.

➢ Jump

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

In figure 9-4-7 there is an application example in SFC.

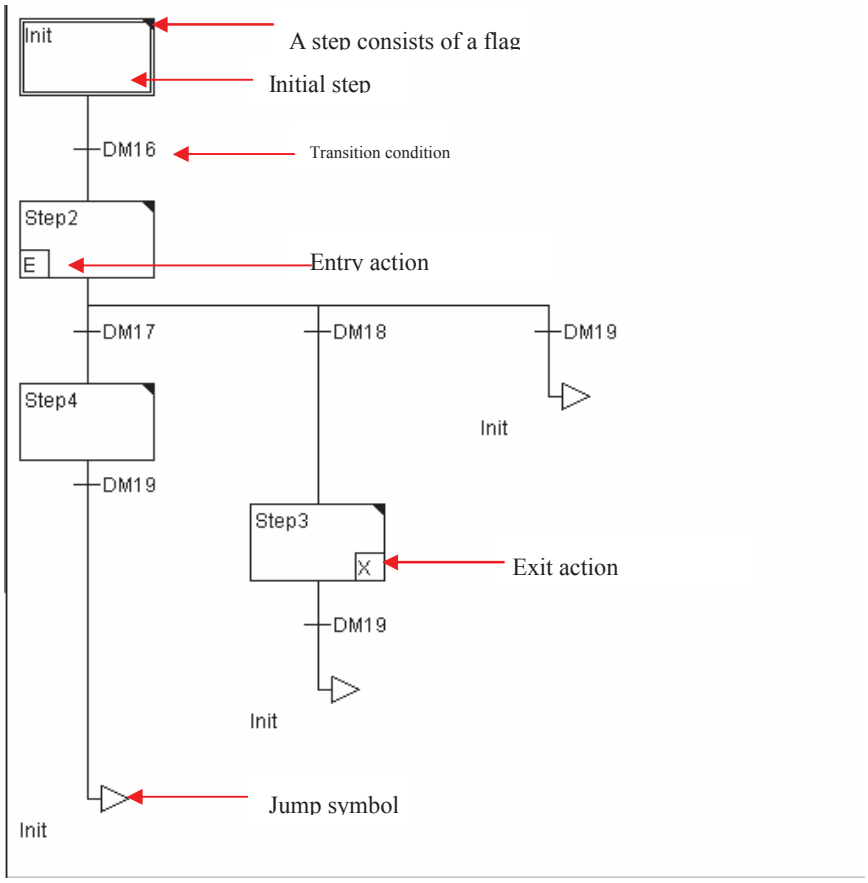


Figure 9-4-7 Explanation of SFC

95CONTINUOUS FUNCTION CHART (CFC)

951 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, shown in figure 9-5-1 and you can edit through menu bar or the contect 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.

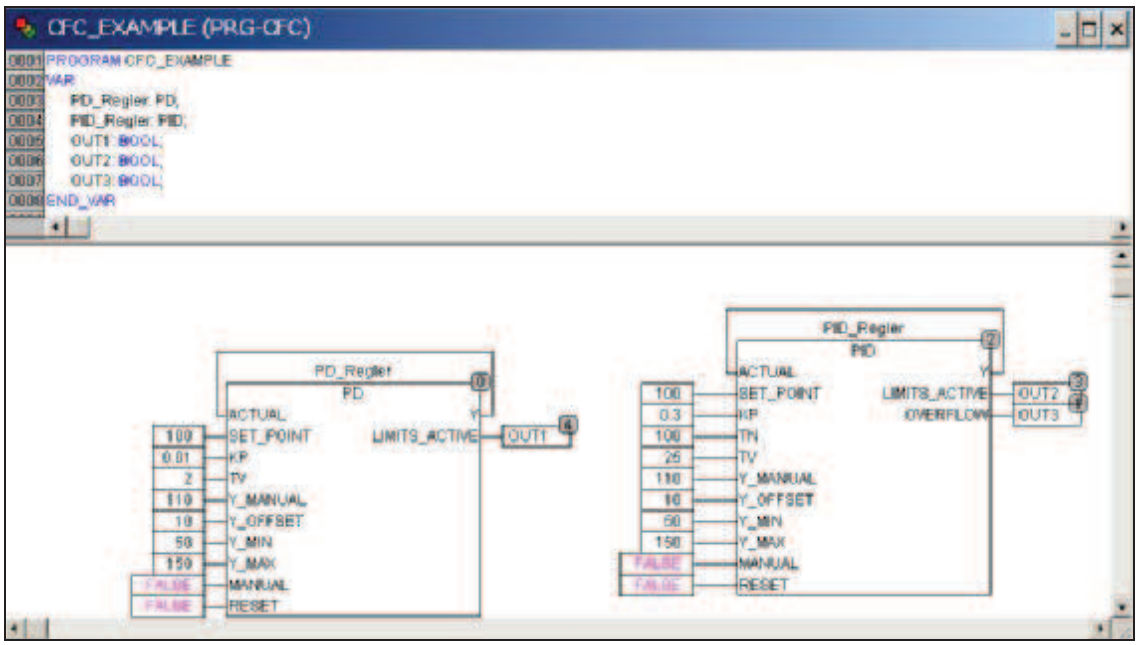


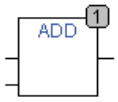
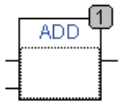
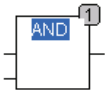
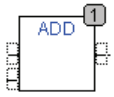
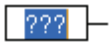
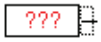
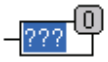
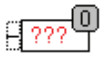
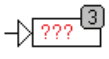
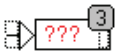
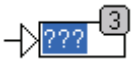
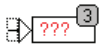
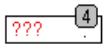
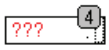
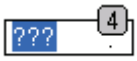
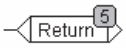
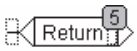
Figure 9-5-1 CFC Editor

952Operation Description

Elements of CFC include boxes, input, output, jump, label, return and comments, where boxes include operator, function, function block and program.

- 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				——

➢ Select Elements

One clicks on the trunk of the element to select it.

To mark more elements one presses the <Shift> key and clicks in the elements required, one after the other, or one drags the mouse with the left hand mousekey depressed over the elements to be marked. The command "Extras"/ "Select all" marks all elements at once.

➢ Move Elements

One or more selected elements can be moved with the arrow keys as one is pressing on the <Shift> key. Another possibility is to move elements using a depressed left mousekey. These elements are placed by releasing the left mousekey. If the elements cover other elements or exceed the foreseen size of the editor, the marked element jumps back to its initial position in such cases 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), 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 with the left mousekey, hold the left mousekey down and drag the mouse cursor onto the input of element E2 and let the left mousekey go.

Place the mouse on the input of element E2, click with the left mousekey, hold the left mousekey down and drag the mouse cursor onto the output of element E1 and let the left mousekey go.

Move one of the elements E1 or E2 and place it in such a way by letting go of the left mousekey that the output of element E2 and the input of element E1 touch.

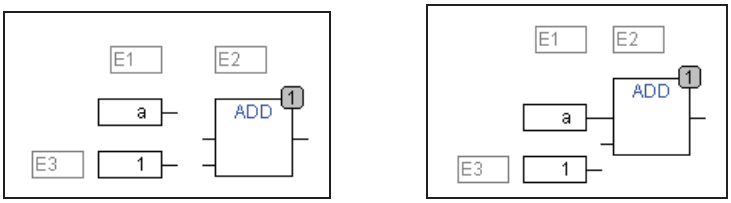


Figure 9-5-2 Connecting Line

- Delete 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), 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 mousekey depressed and drag the connection from the input to E2 away. The connection is removed when the left mousekey 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 offered "???" 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 offered "???" 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 offered "???" 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 offered "???" 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 mouse 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.

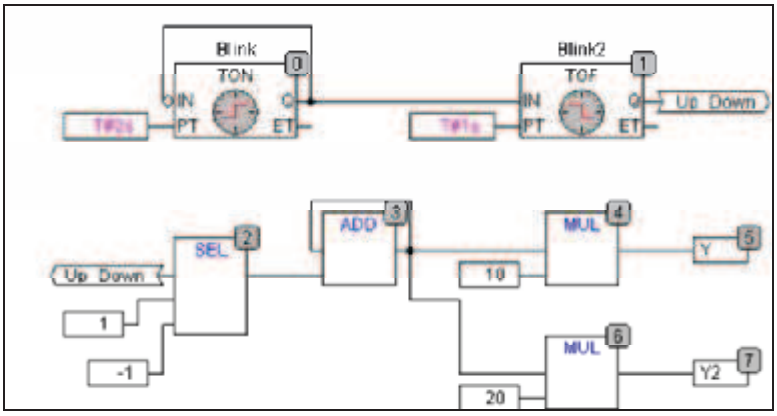


Figure 9-5-3 Example in CFC

CHAPTER 10 SPECIAL FUNCTIONS

10.1 Modbus COMMUNICATION

LM-serial PLC can communicate with the third party equipment, such as touch screen, 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 foramt, 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 astransmission media, baudrate, parity check, the nuber 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 series PLC are listed in table 10-1-1.

Table 10-1-1 Modbus Function Code

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

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 relationship between these data locations and Modbus protocol addresses is shown in table 10-1-2.

Table 10-1-2 The mapping relationship between the data locations of LM series PLC and Modbus protocol addresses:

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

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 plus 1 on the mapping formula, for: 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 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 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.

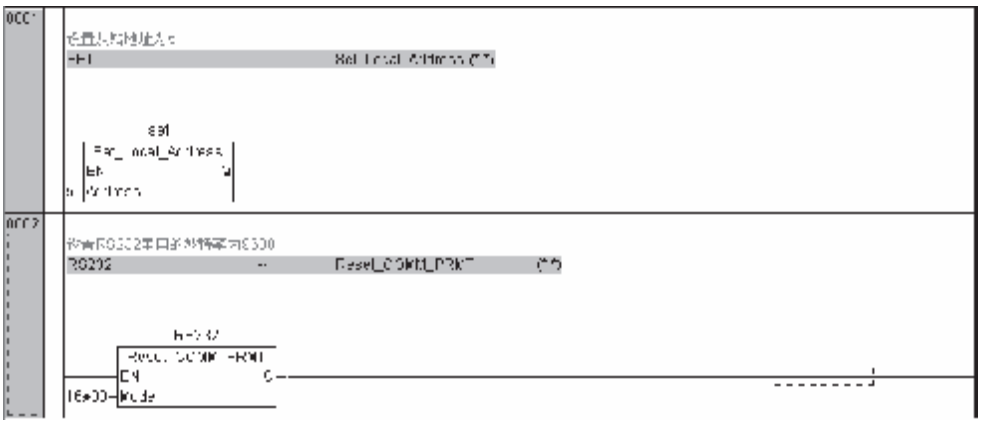


Figure 10-1-1

10.2 INTERRUPT

10.2.1 Interrupt Overview

When PLC receives a special signal from external hardware or internal, CPU stops the original work and starts to process the event and return to the original work after the event has been finished, the whole process is called interrupt. The signal which generates an interrupt is called interrupt source.

LM serried PLC can handle a plenty of interrupt signals generated by interrupt sources. The interrupt sources have different priorities and execute the higher priority interrupt when two interrupts come.

The interrupt sources that can be processed in LM series PLC is listed in table 10-2-1.

Table 10-2-1 Interrupt Events

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 interupt	Called when fast external event 0 interrupt runs
Fast External 1 interupt	Called when fast external event 1 interrupt runs
Fast External 2 interupt	Called when fast external event 2 interrupt runs
Fast External 3 interupt	Called when fast external event 3 interrupt runs
HD_TC7 interupt	Called when Hardware Timer/Counter 7 interrupt runs
HD_TC2 interupt	Called when Hardware Timer/Counter 2 interrupt runs
HD_TC3 interupt	Called when Hardware Timer/Counter 3 interrupt runs

HD_TC4 interupt	Called when Hardware Timer/Counter 4 interrupt runs
HD_RTC_ALM0 interupt	Called when Hardware Real/Time Alarm 0 interupt runs
PTO_0 Finished interupt	Called when PTO_0 Finished interrupt runs
PTO_1 Finished interupt	Called when PTO_1 Finished interrupt runs

1022 Interrupt Application

If you want to use an interrupt the two tasks must be finished. First, write an interrupt service routine. The interrupt service routine is a program which will be executed when an interrupt is generated. Writing an interrupt service routine is the similar with a subprogram and refer to section 7.4.7. Second, after the interrupt service routine has been finished configure the interrupt and the related interrupt service routine. 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, 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 (√) 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.

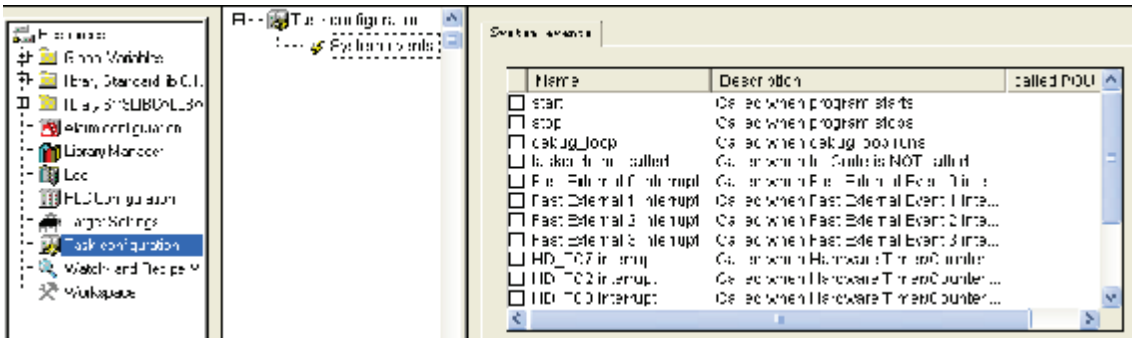


Figure 10-2-1 System Events

✧ Example

✓ Requirement

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 requirement 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 PowerPro_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.

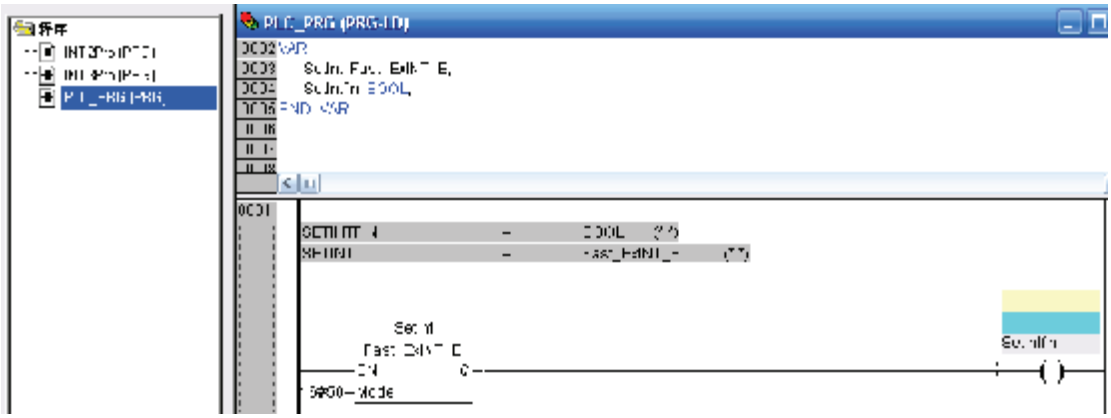


Figure 10-2-2 Variable Declarations and LD of Main Program

Because Fast External Event 3 (I0.6) and Fast External Event 2 (I0.7) are used, activate them by a mouseclick on the control box, shown in figure 10-2-3.

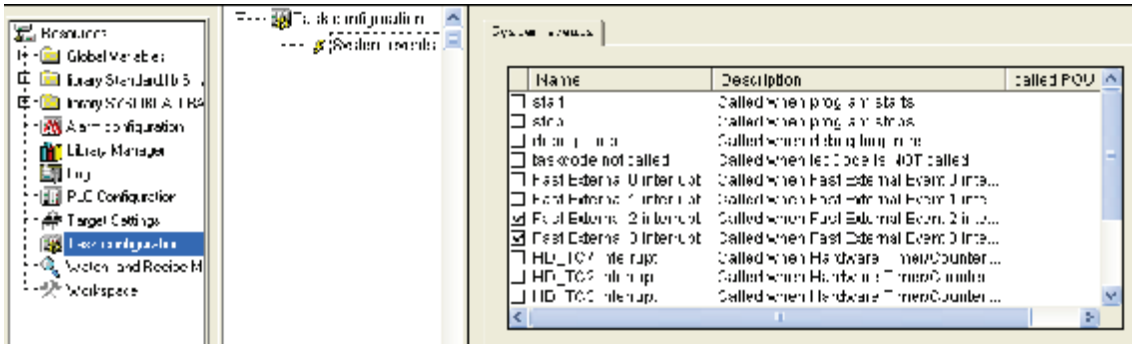


Figure 10-2-3 Choice of Fast External Event

Create subprograms INT2Pro and INT3ProFast after External 2 interrupt and Fast External 3 interrupt and click Create POU respectively, then the two interrupt programs are created successfully, shown in figure 10-2-4.

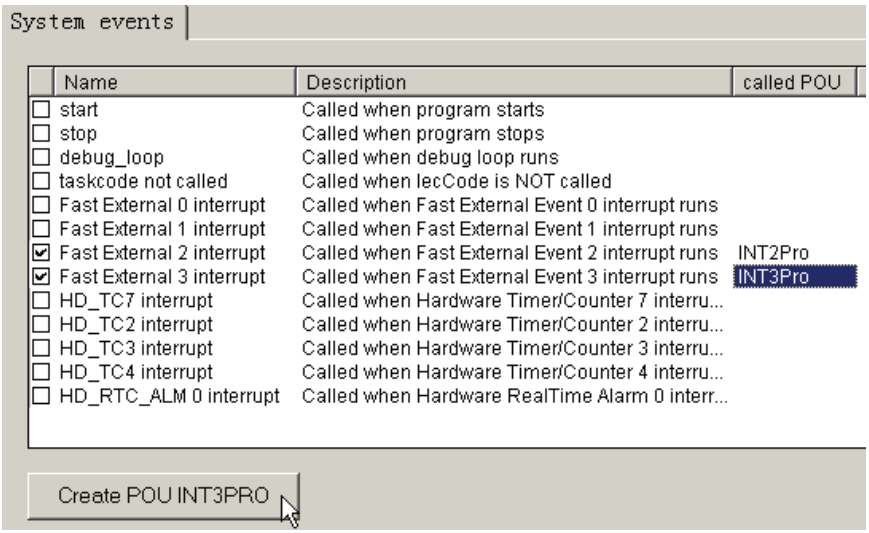


Figure 10-2-4 Create Interrupt Program

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, shown in figure 10-2-5.

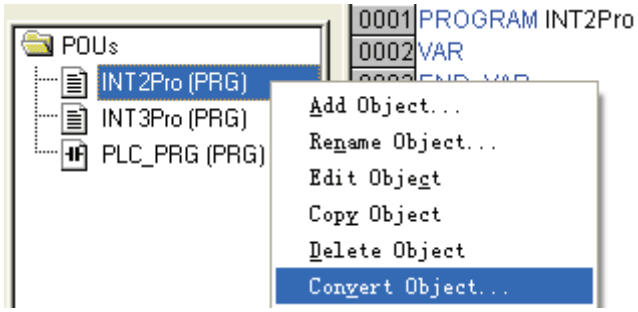


Figure 10-2-5 Convert ST to LD

The LD of INT3Pro is shown in figure 10-2-6.

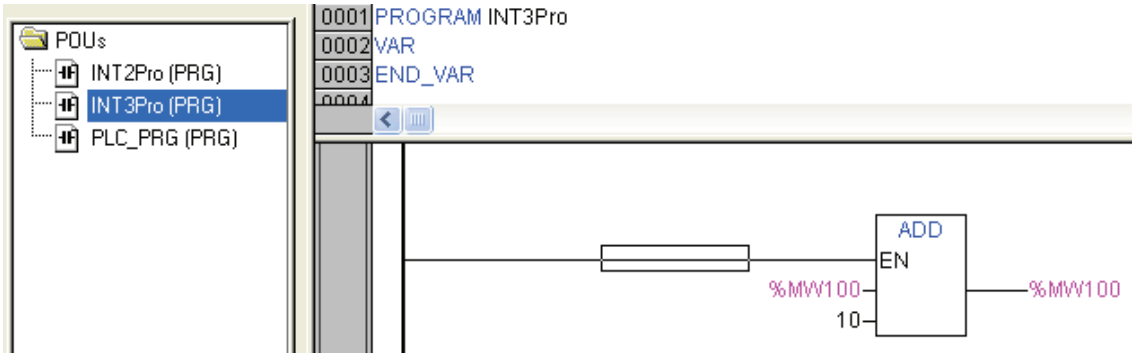


Figure 10-2-6 LD of INT3Pro

The LD of INT2Pro is shown in figure 10-2-7.

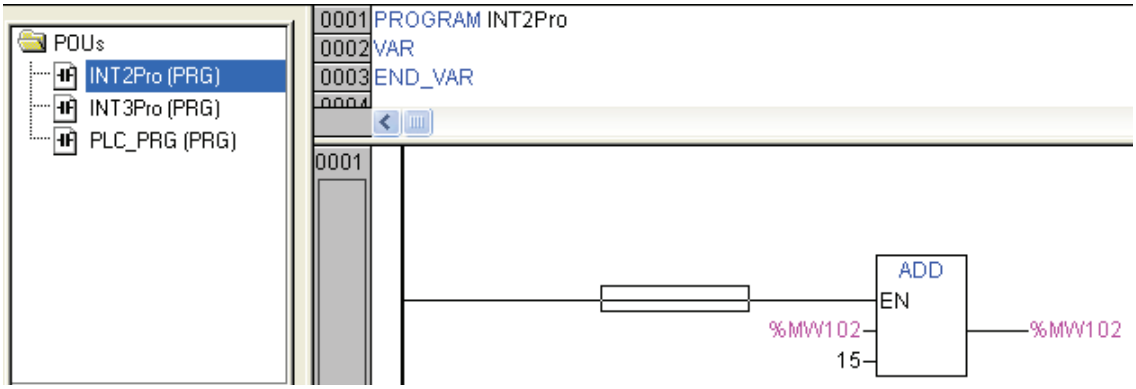


Figure 10-2-7 LD of INT2Pro

Then you can write the program in INT2Pro and INT3Pro. Of cource you can write the program in ST language not converting it to LD. Call the related program when the system event is triggered.

Regard 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 series PLC, first you should know the address occupied by this module. No matter it 's 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 an example. Shown in figure 10-3-1, configure a 4 ×AI module LM3310 after LM3107, then the addresses %IW2, %IW4, %IW6, %IW8 will be assigned automatically and each word stands for a channel acquisition data.

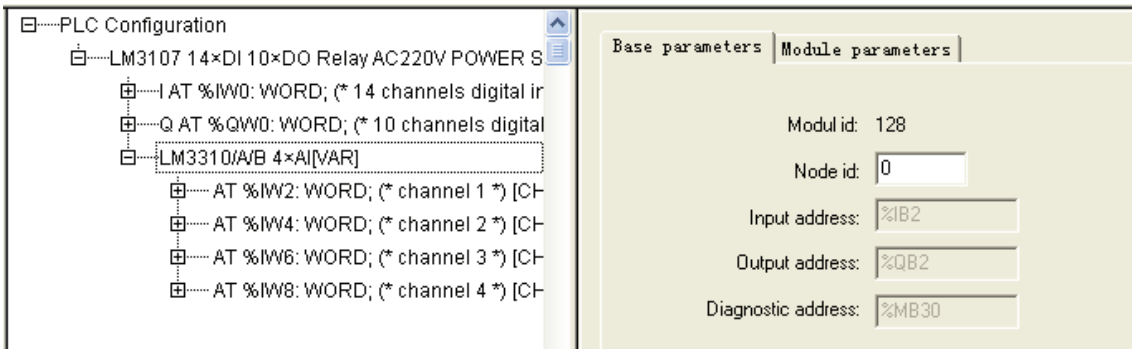


Figure 10-3-1 Configuration of AI Module

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 relationship between channel value and rang is shown in the table below.

Table10-3-1 Range and Scope of AI Module

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

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, 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 aquire 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.

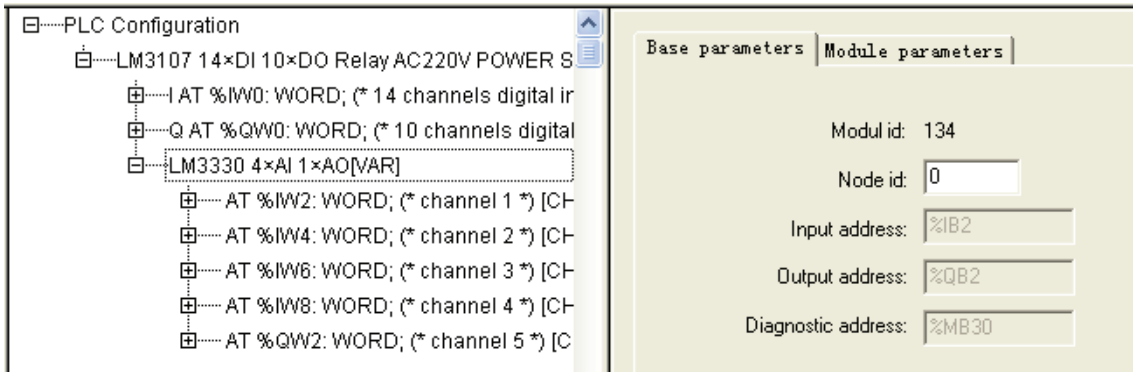


Figure 10-3-2 LM3330 Configuration

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, shown in figure 10-3-3.

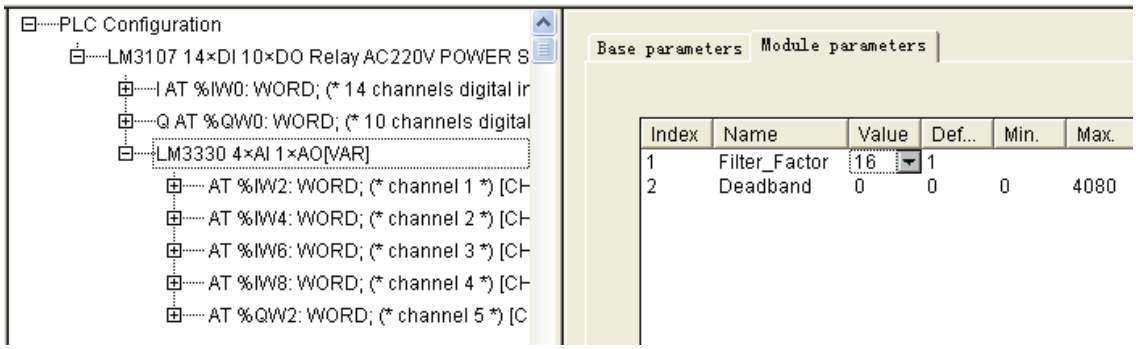


Figure 10-3-3 Set Filter Factor

Configure the range of input channel and output channel after setting the Filter_Factor, 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.

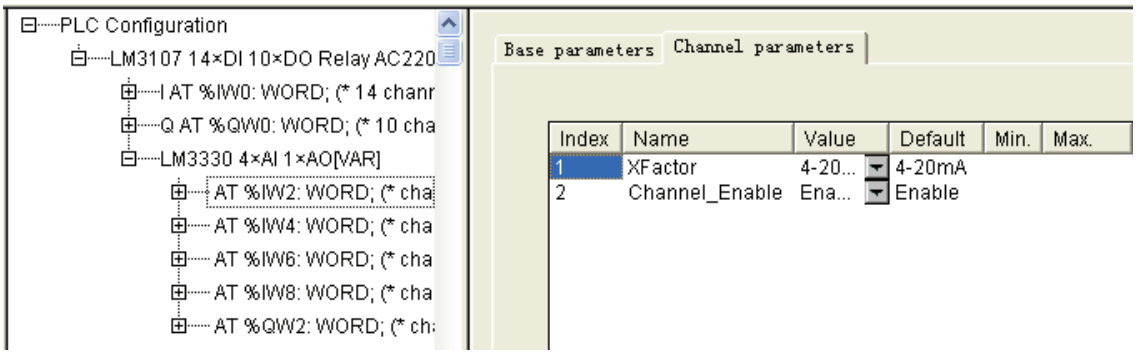


Figure 10-3-4 Set the Range of AI Channel

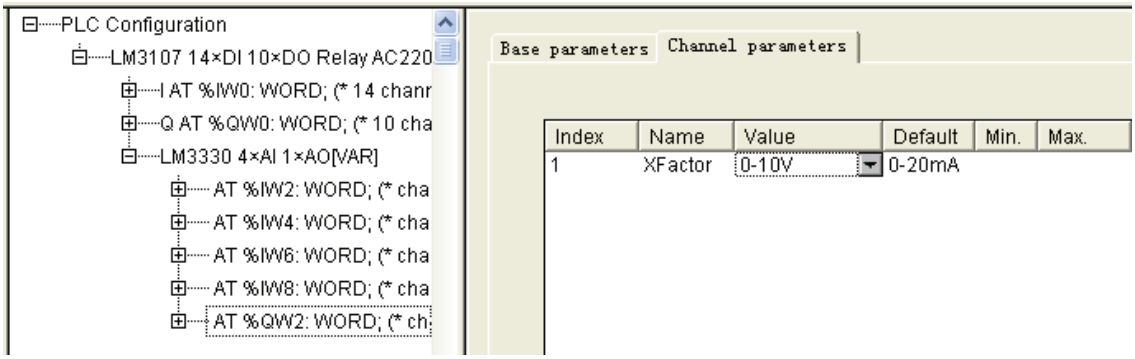


Figure 10-3-5 Set the Range of AO Channel

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 refered to table10-3-1, 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 refered to table10-3-1, 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_OUTis 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 isi 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.

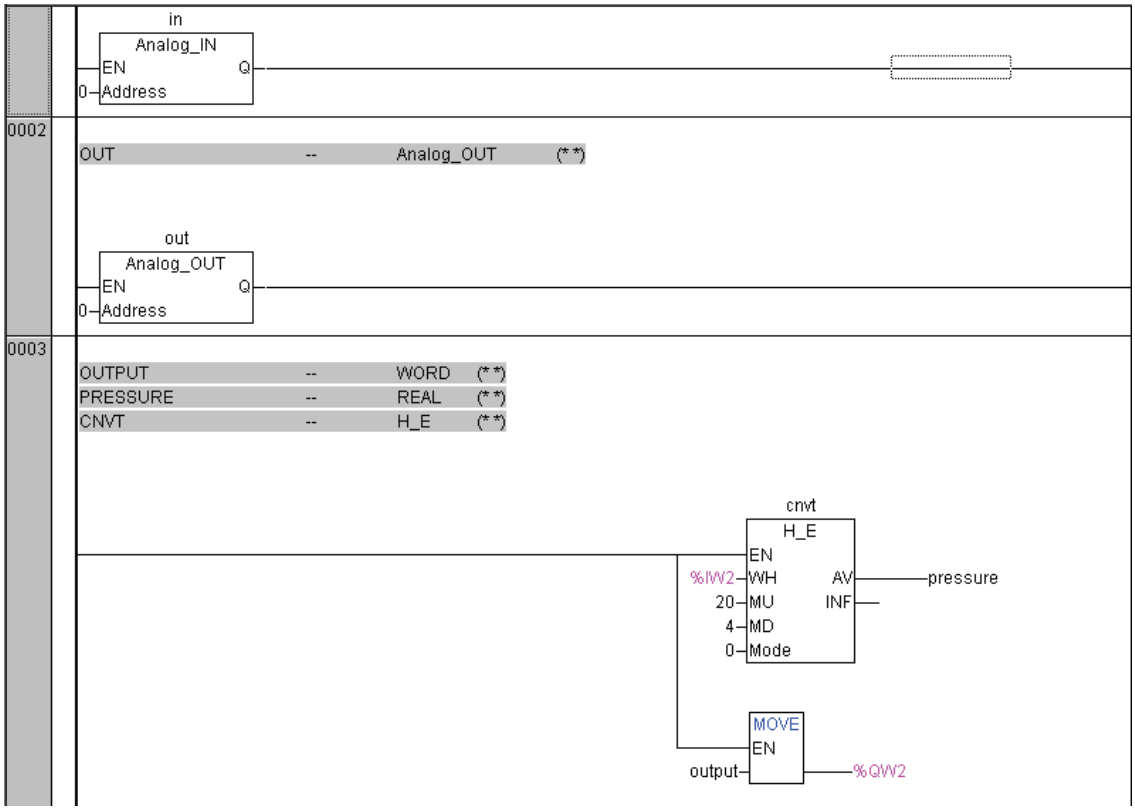


Figure 10-3-6 Program

10.4 DP COMMUNICATION

10.4.1 DP Communication Setting

LM 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 setting of LM3401 is shown in figure 10-4-1 and the **Value** is optional 0-64. The **Value** is 64 in figure 10-4-1. Refer to<<Hardware Manual>>for detailed technical specifications of LM3401.

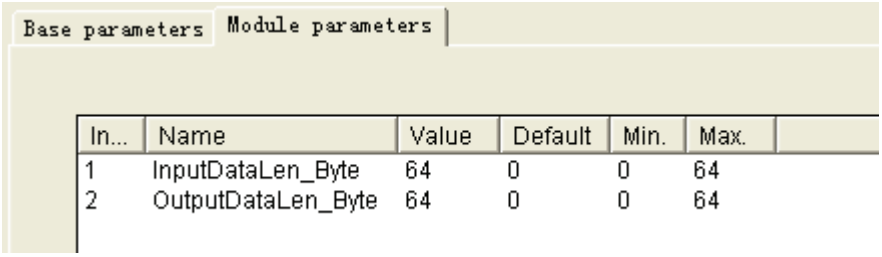


Figure 10-4-1 LM3401 Module parameters

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 can't 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, 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 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 series PLC.

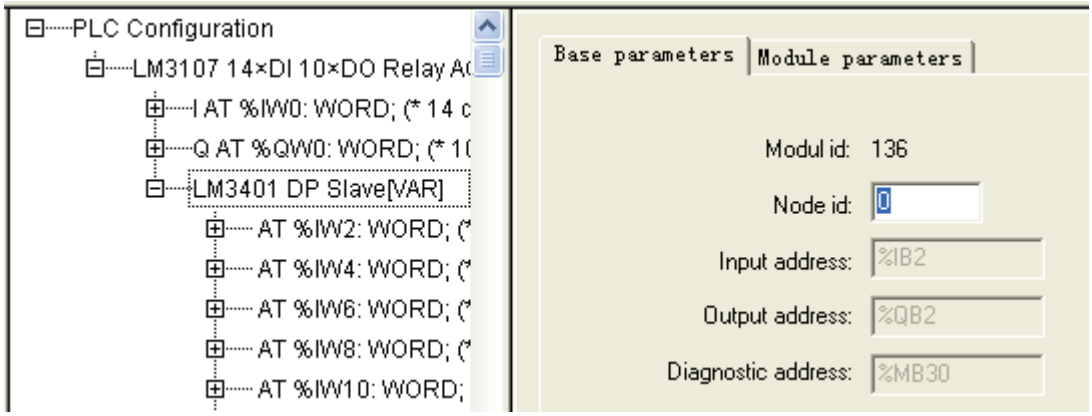


Figure 10-4-3 Configuration of DP_Slave

10.4.2 Example of DP Communication

A simple application example of Profibus-DP function block is introduced 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.

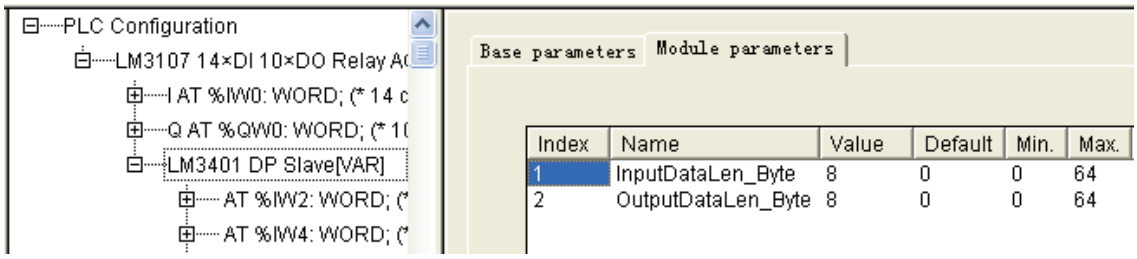


Figure 10-4-4 LM3401 Module parameters

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.

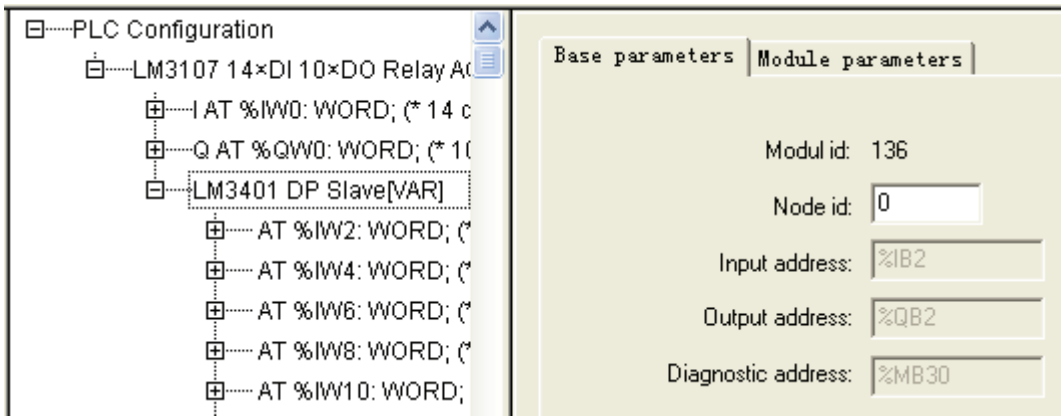


Figure 10-4-5 Configuration of LM3401

The data A, B, C, D sent to DP master by PLC is stored in %QW2, %QW4, %QW6, %QW8 shown in figure 10-4-6.

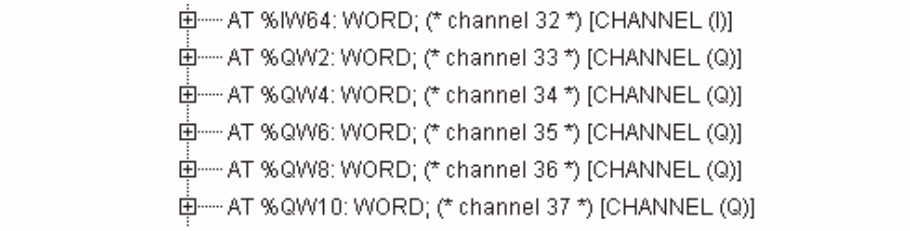


Figure 10-4-6 LM3401 Channels

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 hsown in figure 10-4-7.

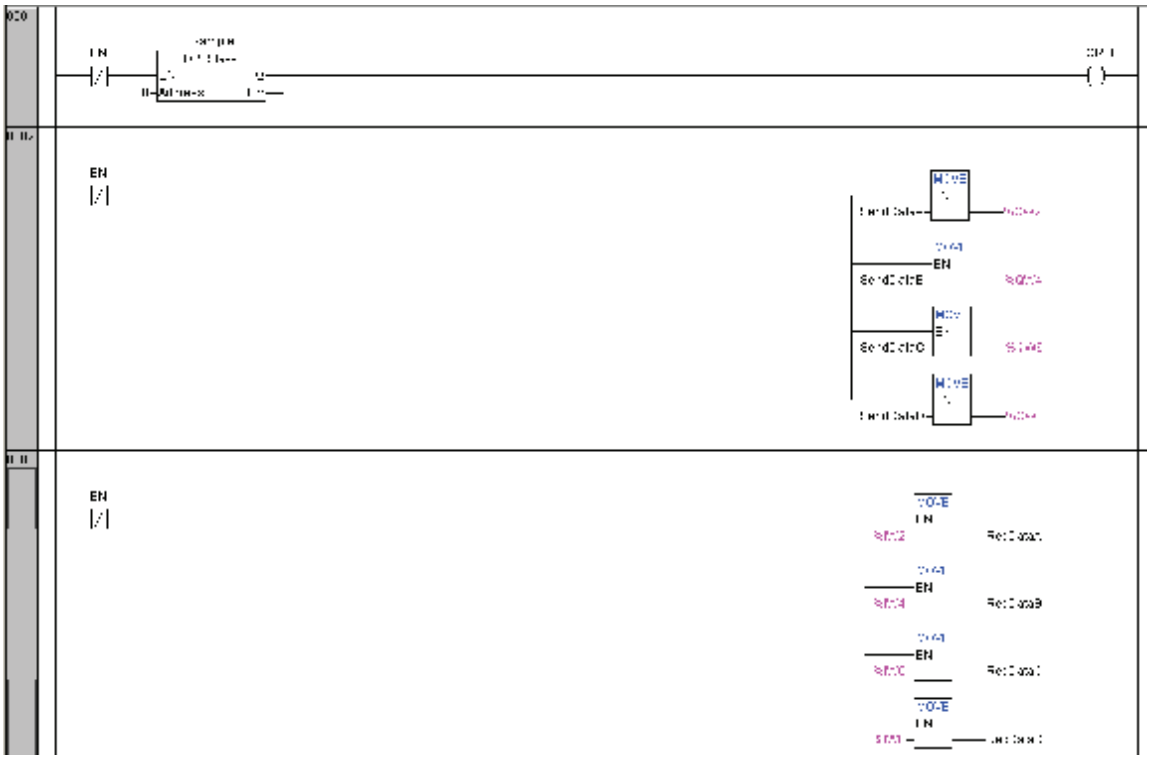


Figure 10-4-7 Ladder Diagram of LM3401

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, 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 %IB3 or %QB3, Etc.

10.5 ETHERNET COMMUNICATION

10.5.1 Ethernet Communication Setting

LM series PLC provides Ethernet communication processing. LM3403 is a Ethernet communication block and provides Modbus TCP slave communication function.

Configure the module parameters of LM3403 after the PLC configuration has been finished. The main parameters include WriteDataLen_Byte and ReadDataLen_Byte. The module parameter setting of LM3403 is 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.

Base parameters | Module parameters

Index	Name	Value	De...	Min.	Max.
1	IP_Address	172.20.45.160			
2	Subnet_Mask	255.255.252.0			
3	Gateway_Address	172.20.45.1			
4	MAC_Address				
5	ReadDataLen_Byte	200	0	0	200
6	WriteDataLen_Byte	200	0	0	200

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 can't scan it when EN is reset.

Figure 10-5-2 Ladder Diagram of EtherNet_TCP

System will assign input address and output address automatically after LM3403 has been configured, shown in figure 10-5-3. The input address is 200 bytes starting from %IW2 and output address is 200 bytes starting from %QW2. The communication between the Modbus TCP Master and LM 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 series PLC.

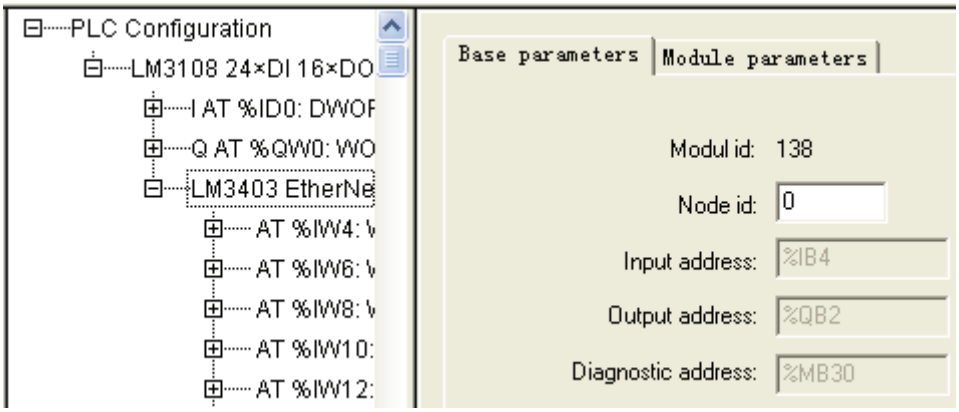


Figure 10-5-3 Configuration List of EtherNet_TCP

1052 Example of Ethernet Communication

In the 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, shown in figure 10-5-4.

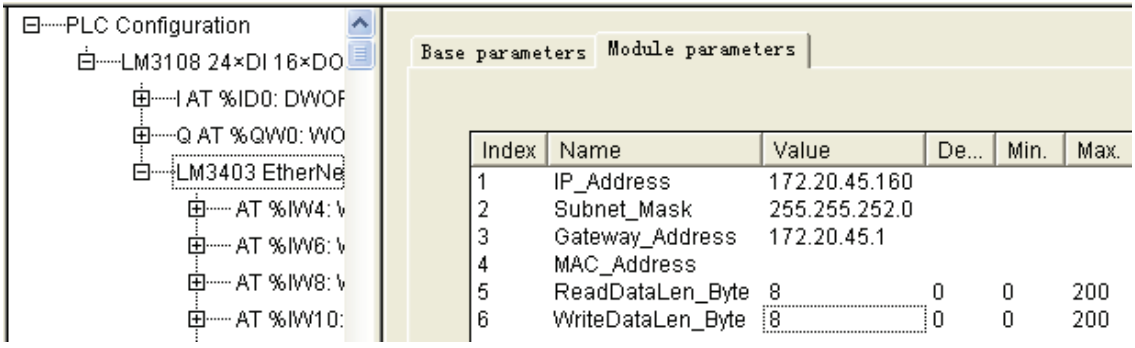


Figure 10-5-4 LM3403 Configuration

- ✓ IP_Address is the IP of Ethernet module (must in the same network segment with PC and no conflict with other IP) .
- ✓ Subnet_Mask is subnet mask, and is consistent with the subnet mask of PC.
- ✓ GateWay_Addres is the address of gateway.
- ✓ MAC_Address null.
- ✓ ReadDataLen_Byte is the length of data sent to PLC by PC, and the received bytes is 8 (must larger than actual length and the maximum is 200) .
- ✓ WriteDataLen_Byte is the length of data sent to PC by PLC, and the sent bytes is 8 (must larger than actual length and the maximum is 200) .

➢ The Address in Ethernet module should be consistent with 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.

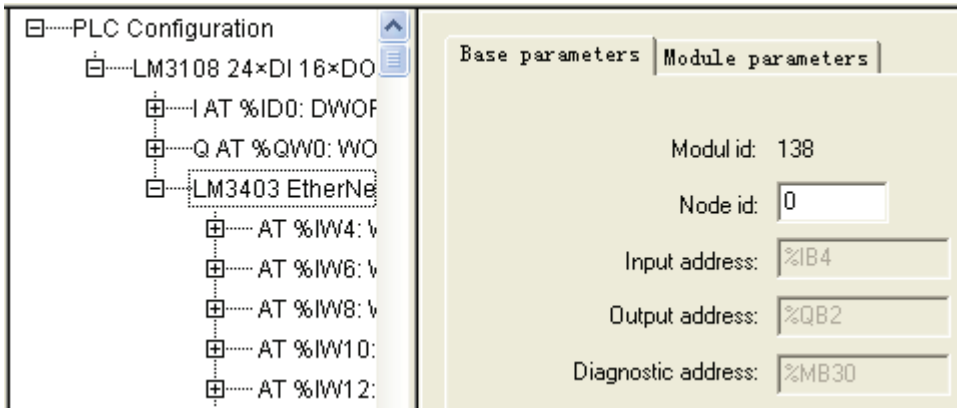


Figure 10-5-5

➢ The data A, B sent to PC by PLC are stored in %QW2, %QW4 shown in figure 10-5-6, and data C is stored in %QX6.0.

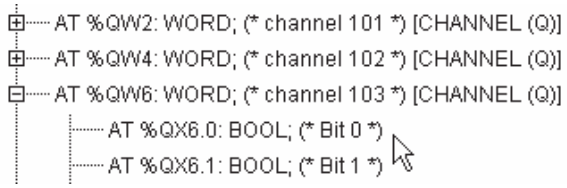


Figure 10-5-6 Address of Output Area

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.

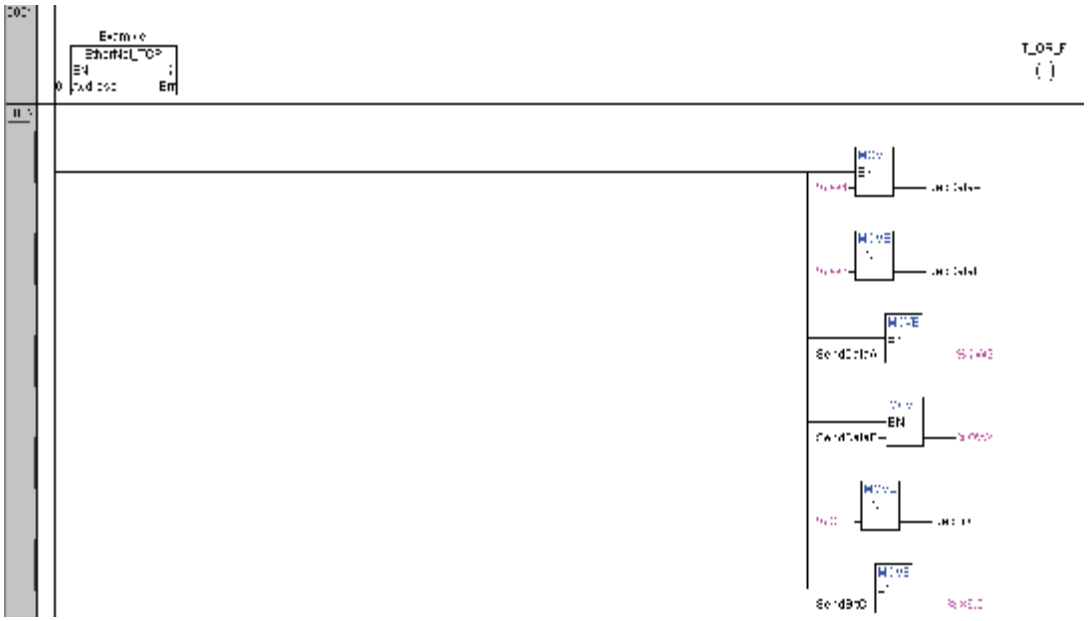


Figure 10-5-7 Program

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 realize the visualization of control process. So visualization is PLC HMI (Human Machine Interface, HMI) .

In PowerPro programming system there is an integrated visualization editor. In the process of developing applications of control system, in PowerPro the users are allowed to develop visualization object to watch and operate PLC datas without other development tools.

11.1 NEW VISUALIZATION

Startup PowerPro, and build a new project1.pro. In object organizer click "Visualization", shown in figure 11-1-1.

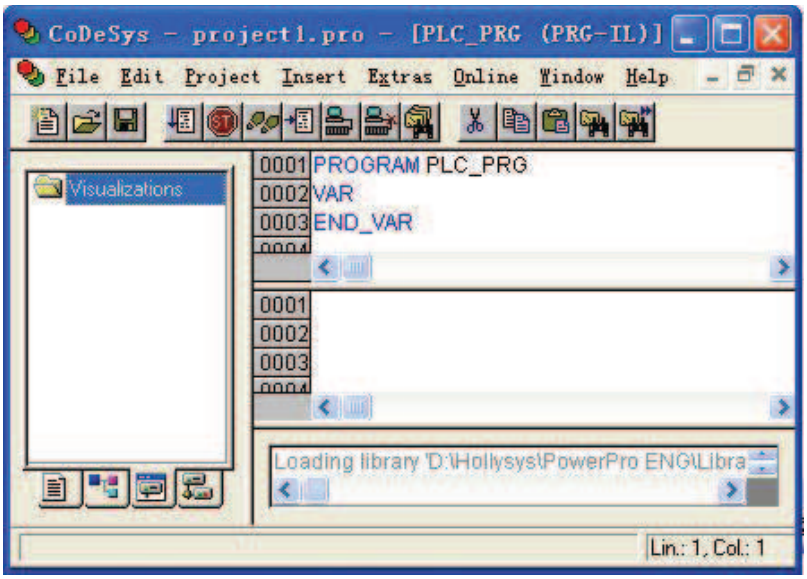


Figure 11-1-1 Visualization

In "Visualization" click the right mouse and select "Add Object", shown in figure 11-1-2.

Figure 11-1-2 Add Visualization

The window of "New Visualization" appears, shown in figure 11-1-3.

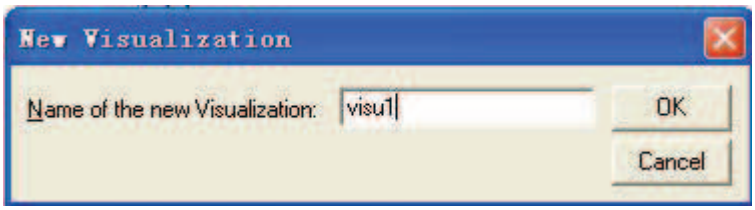


Figure 11-1-3 Name of the New Visualization

Enter the name in "Name of the new Visualization ", such as visu1, click "ok", and then a newvisualization is built. The right of the window is editor window, shown in figure 11-1-4.

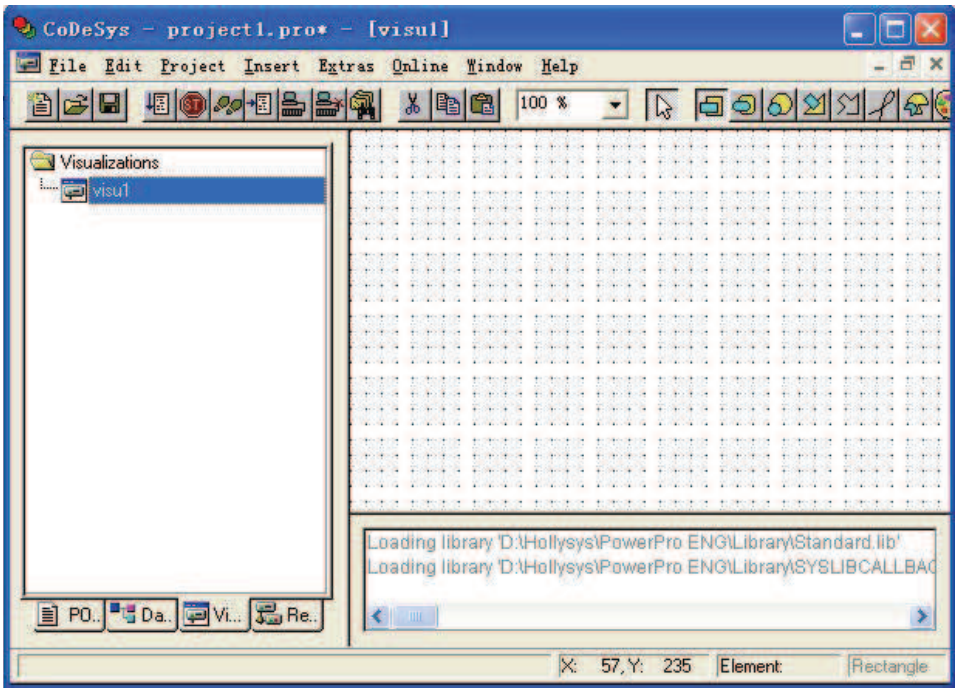


Figure 11-1-4 Build a visu1

11.2 VISUALIZATION ELEMENTS

The tool for editor is allocated 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, 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, shown in figure 11-2-2, and

you can make certain settings that affect the visualization. The menu can appear also by clicking the right mouse in editor window.

Figure 11-2-2 “Extras” Menu

The shortcut buttons are allocated on the top of the main window, 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

Table 11-2-1 Edit Tool in Visualization

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 ellips).
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.

11.3 EDIT VISUALIZATION

11.3.1 Draw Visualization

If you hold the mouse pointer for a short time on the elements in tool bar then the name of theelement 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 thenmove 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 doubleclicking 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 atatus bar.

11.3.2 Arrange Visualization

When you are drawing visualizations, it's necessary to modify and arrange them.

- Select

In order to select anelement, click with the mouse 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.

With a selected element, the 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.

With the selection of a polygon, 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, shown in figure 11-3-1.

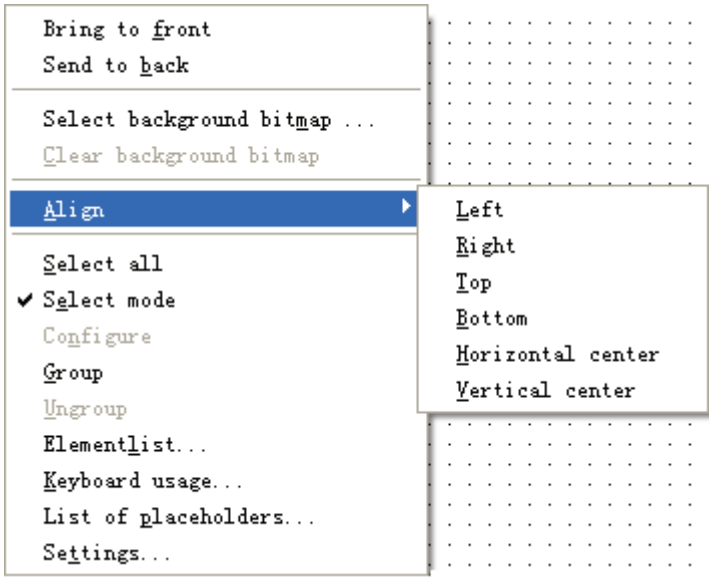


Figure 11-3-1 Context Menu of Visualization

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 to the element that is furthest to the left.

"Right": the right edge of each of the elements will be aligned to the element that is furthest to the right.

"Top": the top edge of each of the elements will be aligned to the element that is furthest to the top.

"Bottom": the bottom edge of each of the elements will be aligned to the element that is furthest to the bottom.

"Horizontal center": each of the elements will be aligned to the average horizontal center of all elements

"Vertical center": each of the elements will be aligned to the average vertical center of all elements.

- Configure

The options in "Category" vary accoding to different visualization elements. "Configure" is available only after one or more elements are selected, or else the menu item is in gray.

- 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, shown in figure 11-3-2, including **Number**, **Type** and **Position** of elements. You can edit the elements according to the buttons on the right. The elementlist also will appear by clicking the right mouse 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.

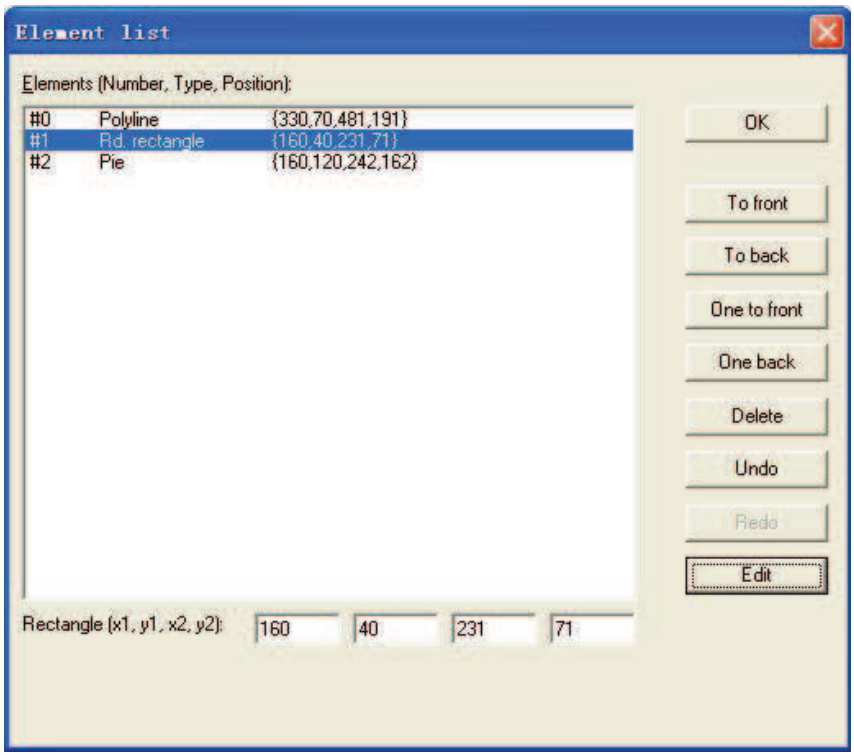


Figure 11-3-2 Elementlist

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.

Table 11-3-1 VIS_1 Keyboard Usage

Shift	Ctrl	Action	Key	Expression
x		Toggle	A	PLC_PRG.automatic
	x	Zoom	Z	VIS_2

Table 11-3-2 VIS_2 Keyboard Usage

Shift	Ctrl	Action	Key	Expression
x		Exec	E	INTERN LANGUAGE DEUTSCH
	x	Zoom	Z	VIS_1

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**.

Tale 11-3-3 Actions

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'

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.

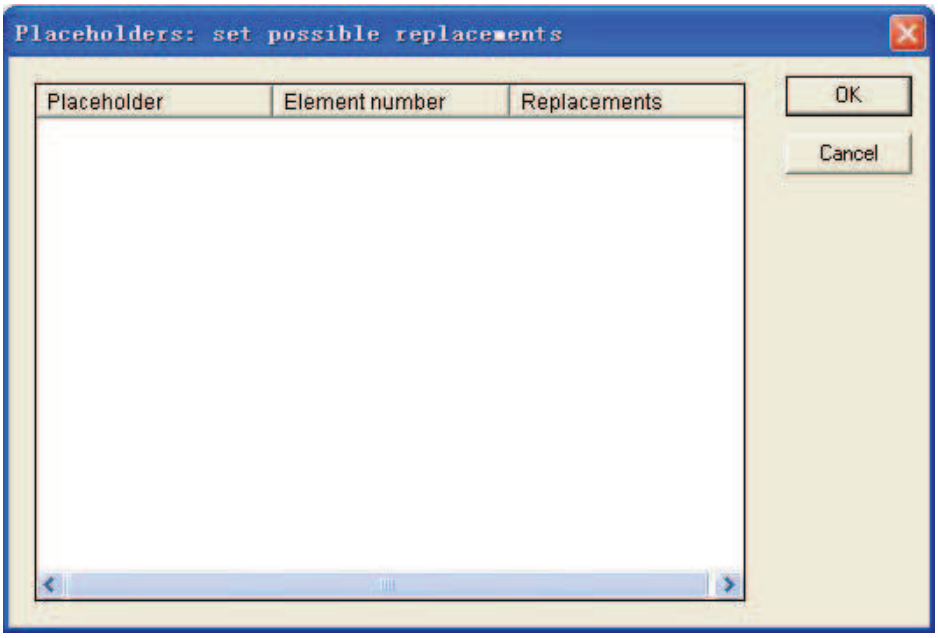


Figure 11-3-3 List of Placeholder

- 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.

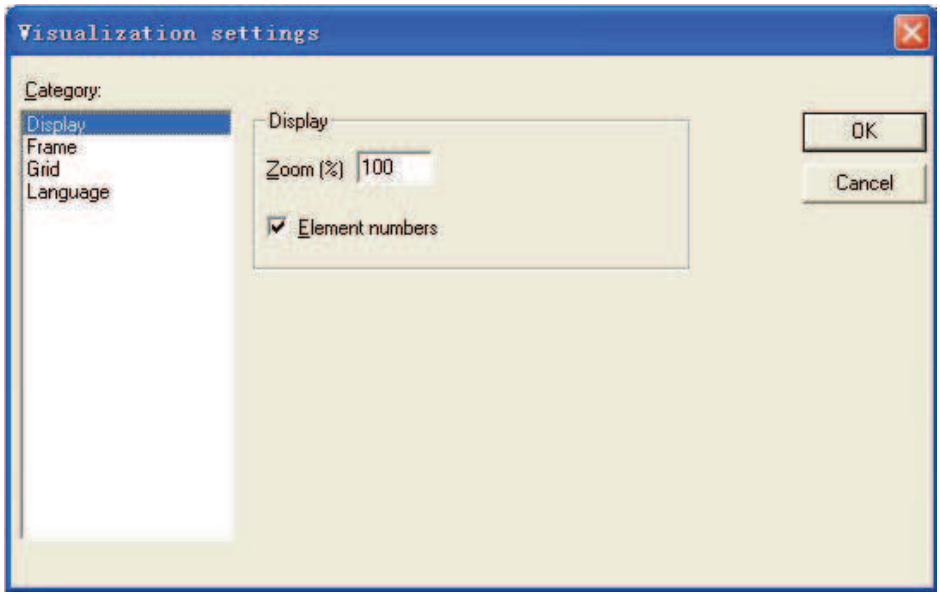


Figure 11-3-4 Visualization Settings

- Display: Enter a zoom facter into the field **Zoom (%)** to increase or decrease the size of the visualization display. The **Element numbers** is optional, shown in figure 11-3-4.
- Frame: Frame setting, 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.

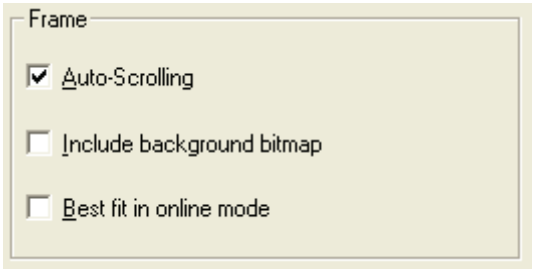


Figure 11-3-5 Frame Setting

- Grid: Grid setting is shown in figure 11-3-6.

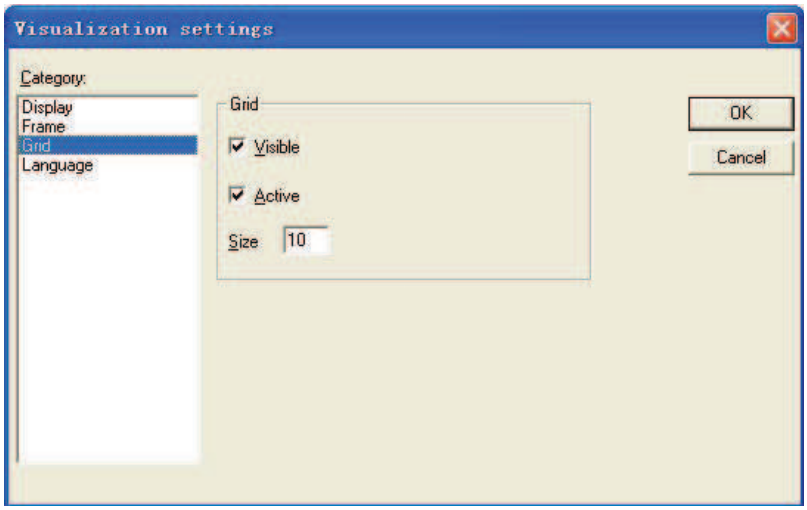


Figure 11-3-6 Grid Setting

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 can't support the function.

11.4 VISUALIZATION ELEMENT CONFIGURATION

11.4.1 Method for Visualization Element Configuration

In visualization editor select a visualization element and click the right mousekey or execute the command "Extras" and active "Configure", shown in figure 11-4-1. The "Configure" menu is available in the condition that a visualization element should be selected, or else it's in gray script.

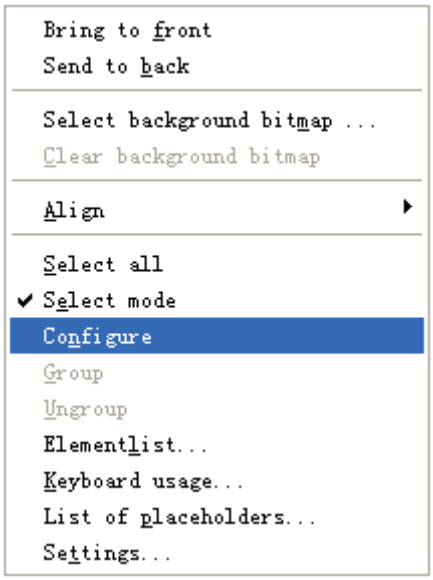


Figure 11-4-1 “Extras”/“Configure” Menu of Visualization

With the command “Configure”, a visualization element configuration dialog box appears, 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.

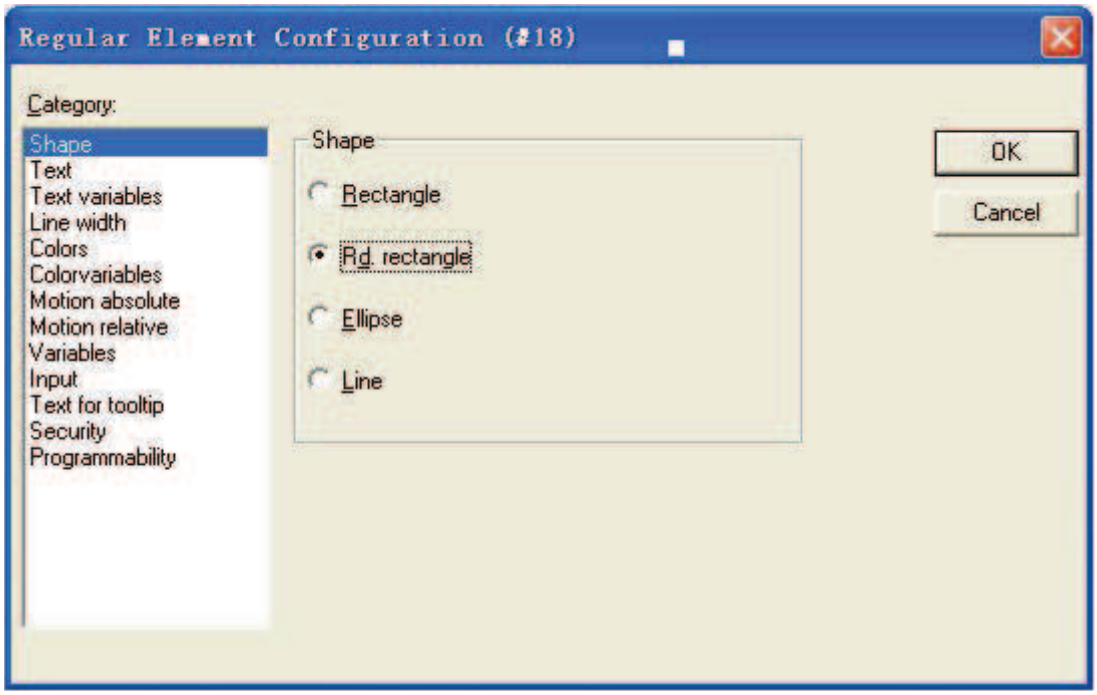


Figure 11-4-2 Visualization Element Configuration Dialog Box

11.4.2 Properties of Visualization Object

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 the visualization object and change its property. The relationship between visualization property and visualization element is shown in table11-4-1, where the symbol “√” shows that there is a relationship.

Table1 1-4-1 the relationship between visualization property and visualization element

visualization property		Visualization elements														
Category	Name	Rectangle	Rounded rectangle	Ellipse	Polygon	Polyline	Curve	Pie	Bitmap	Visualization	Button	WMF file	Table	Trend	Alarm	activeX element
ıape	Shape	√	√	√	√	√	√									
ext	Text	√	√	√	√	√	√	√	√	√	√	√				
ext variables	Text variables	√	√	√	√	√	√	√	√	√	√	√				
ne width	Line width	√	√	√	√	√	√	√	√	√		√				
olors	Colors	√	√	√	√	√	√	√						√		
olor 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															√

11.5 VISUALIZTION STATIC PROPERTY

The visualization static property is the parameter that describes the basic shape of visualization element, including Shape, Text, Line width, Colors and so on.

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**, shown in figure 11-5-1. The changes of Shape attribute are valid only in the determined area.

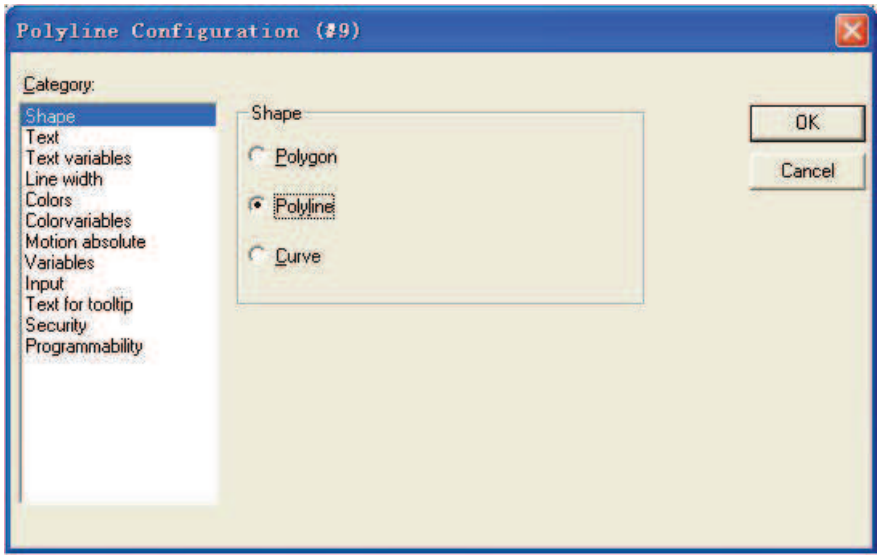


Figure 11-5-1 Dialog Box for Configuring Visualization Elements

11.5.2 Text

One can add text in visualization object using **Text** property, 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.

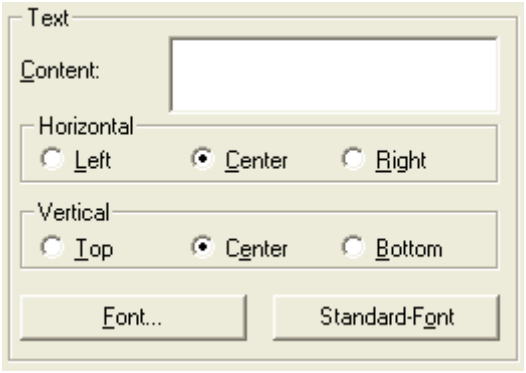


Figure 11-5-2 Text Setting

In standard C library the common 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.

Table 11-5-1 Placeholder and Meaning

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: china standard time
%Z	
%%	Percent sign

✧ 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 shown in figure 11-5-3. The result in online mode is diaplayed in figure 11-5-4. Display current time format: 中国标准时间-year-month-day hour：minute：second.

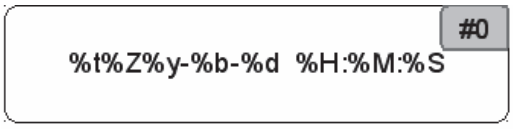


Figure 11-5-3 Time Placeholder Application

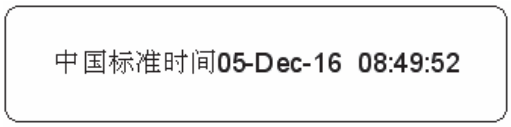


Figure 11-5-4 Display in Online Mode

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 pixel in dialog box for line width configuration, shown in figure 11-5-5, additionally an other 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.

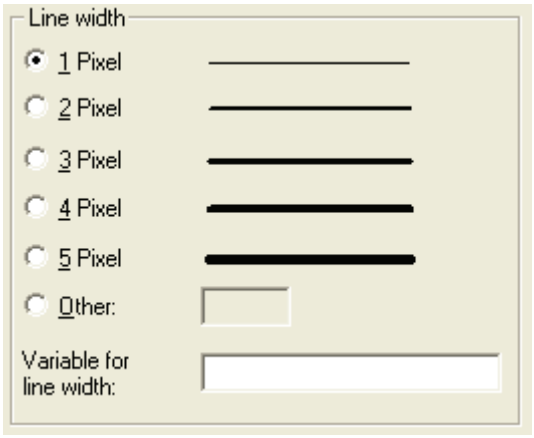


Figure 11-5-5Line Width Configuration

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, 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 element. As soon as the parameter is defined dynamically by a variable, the static setting will be

overwritten in online mode.

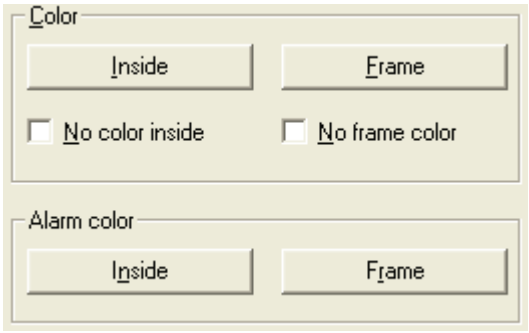


Figure 11-5-6 Colors Configuration

11.5.5 Text for tooltip

The dialog "Text for tooltip", shown in figure 11-5-7, offers an input field "Content"for text which appears in a text field as soon as the mouse cursor is passed over the object in offline or online mode. You can insert a line break by using the key combination <Ctrl>+<Enter>.

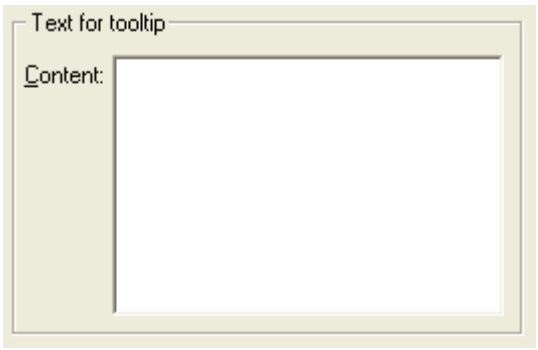


Figure 11-5-7 Text for tooltip

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, 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".

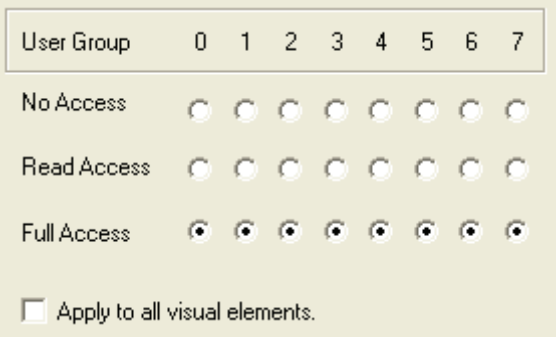


Figure 11-5-8 Access Setting

11.5.7 Bitmap Configuration

You can insert a **Bitmap** in a visualization.You can enter the options for a bitmap in the **Bitmap** category within the visualization element configuration dialog box, 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 new a 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, or else it will not. If the **Clip** option is also selected, only that 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.

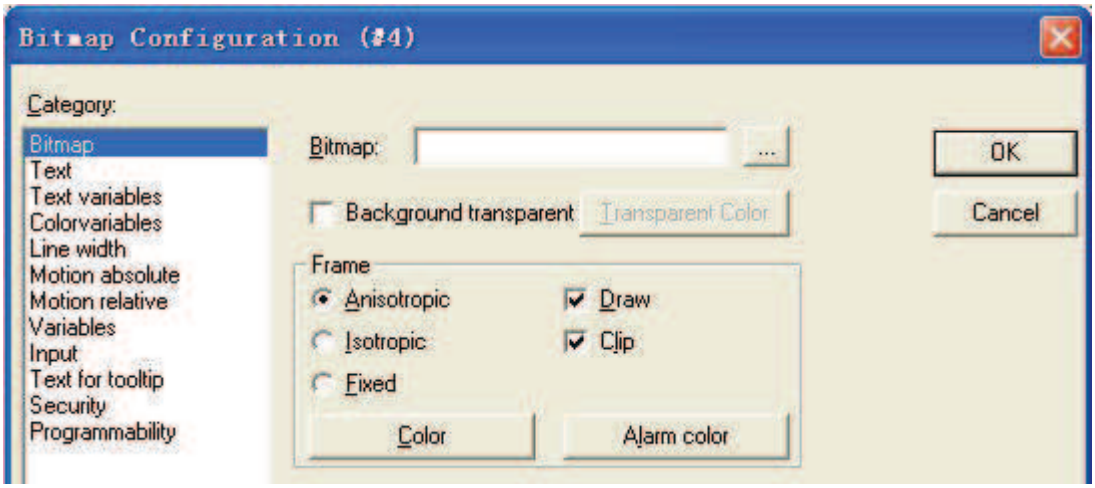


Figure 11-5-9 Bitmap Configuration

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, 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, or else it will not. If the **Clip** option is also selected, only that 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.

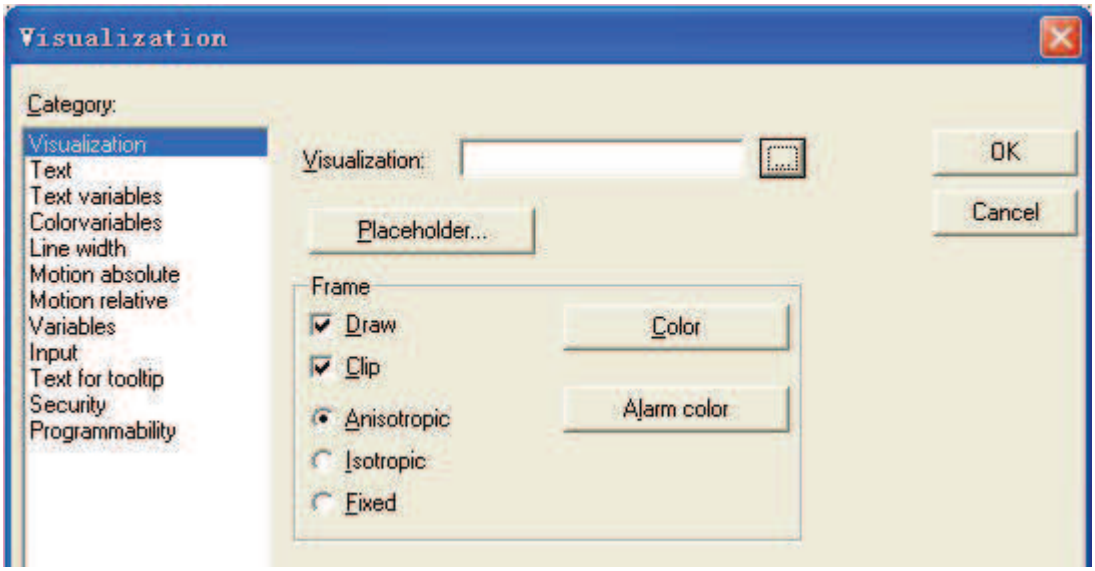


Figure 11-5-10 Visualization Configuration

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.

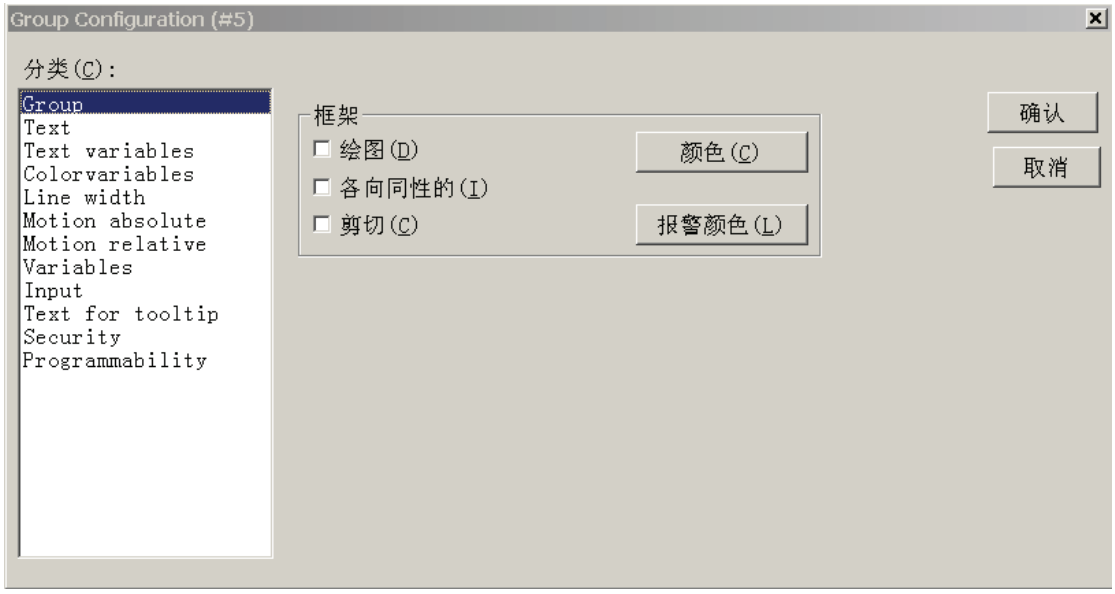


Figure 11-5-11 Group Configuration of WMF

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 is opened, 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, then segment is displayed only without angle.

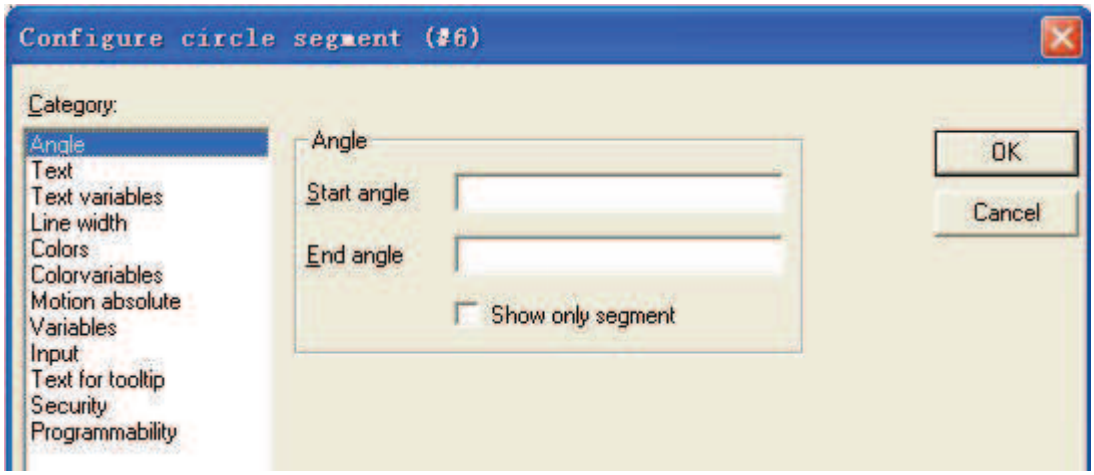


Figure 11-5-12 Dialog for Configuring a Pie

✧ 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 can not only be defined by a static setting, but also by 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', shown in figure 11-6-1. Active "Object name" and enter a variable name. The variable automatically will be declared with type **VisualObjectType**, a structure which is contained in the library **SysLibVisu.Lib**. Tthe variable

declared here is a global variable and the declaration is done implicitely and not visible for the user. Make sure that the library is included in the library manager.

After the next compile the newly assigned structure variable will be available in the project.

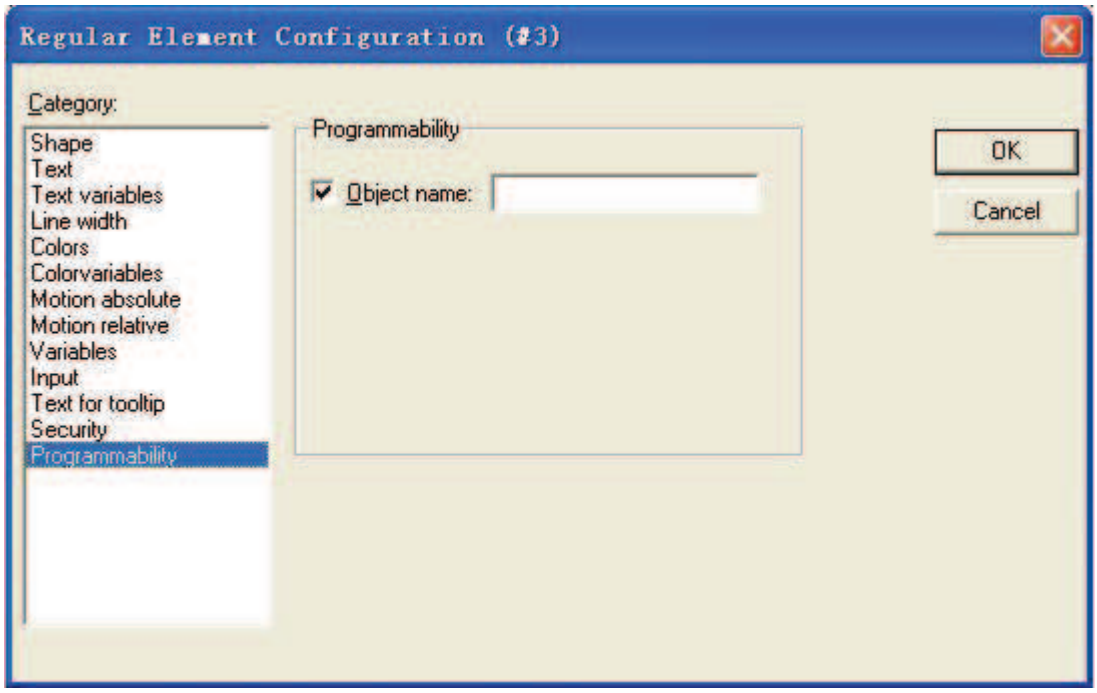


Figure 11-6-1 Programmability

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.

Table 11-6-1 Data Type and Effect of Component

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	vis1.dwFillColorAlarm:=16#00808080; (as soon as Variable bToggleColor	Color: Alarm color Inside Colorvariables:

	colors see subsequent to this table)	gets TRUE, the frame will be displayed grey-colored)	Fill color alarm
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

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 VISUALIZTION DYNAMIC PROPERTY

Visualiztion dynamic property is to define a dynamic parameterizing by entering project variables.

11.7.1 Text variables

Text variable is used to define dynamically text. Enter the variables in the dialog of configuraing text variables, shown in figure 11-7-1. Enter the variable name with the aid of the input assistant (<F2>).

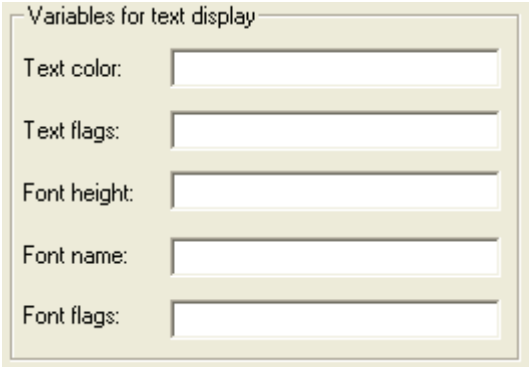


Figure 11-7-1Variables for text display

11.7.2 Color variables

“Color variables ” is used to define the dynamic attributes of colors. The dialog box of “variables for colors” is shown in figure 11-7-2 and one can enter the related variable in text box to realize the animation effect. Enter the variable name with the aid of the input assistant (<F2>).

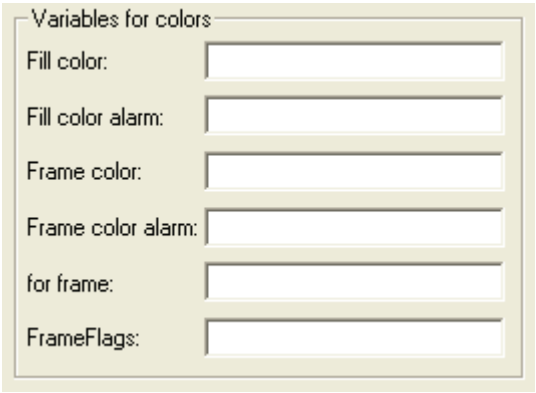


Figure 11-7-2 Setting of Color variables

11.7.3 Motion absolute

Motion relative is used to define the dynamic attributes of motion absolute. The dialog box of "motion absolute setting" is shown in figure 11-7-3 and one can enter the related variable in text box to realize the animation effect. Enter the variable name 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 implicitely. 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.

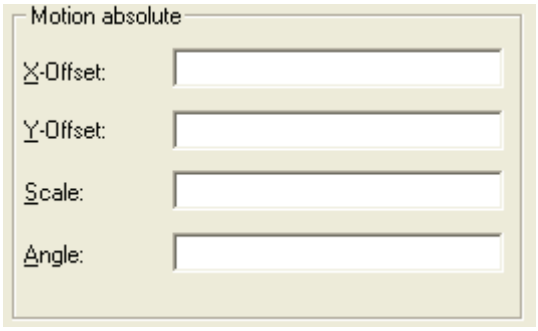


Figure 11-7-3 Configuration of Motion absolute

11.7.4 Motion relative

Motion relative is used to define the dynamic attributes of motion relative. The dialog box of "motion relative setting" is shown in figure 11-7-4 and one can enter the related variable in text box to realize the animation effect. 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.

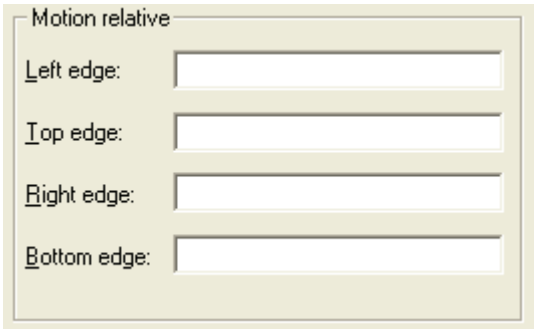


Figure 11-7-4 Setting of Motion Relative

11.7.5 Variables

"Variables" is used to define the other dynamic attributes of graphic object. The dialog box of "Variables" is shown in figure 11-7-5 and one can enter the related variable in text box to realize the animation effect. Enter the variable name with the aid of the input assistant (<F2>).

- "Invisible": You can enter Boolean variables in the **Invisible** field. The valuesin 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".

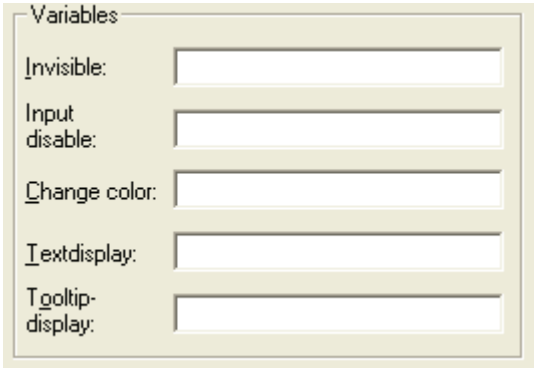


Figure 11-7-5Variable Setting

11.7.6 Input

Input is used to define input dynamic attributes of graphic object. The dialog box of "input setting" is shown in figure 11-7-6 and one can enter the related variable in text box to realize the animation effect. Enter the variable name with the aid of the input assistant (<F2>).

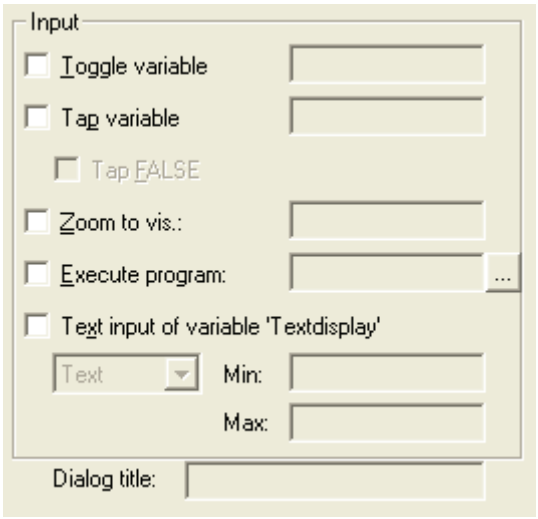


Figure 11-7-6 Input Setting

- 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 TABLE

You can insert a table in a visualization object, shown in figure 11-8-1.

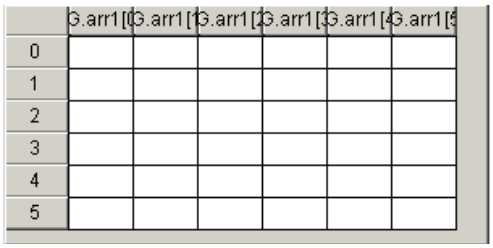


Figure 11-8-1 Insert Table

Double click the table and a dialog "Configure Table" is opened, shown in figure 11-8-2, and you set the options 'Table', 'Columns', 'Rows', 'Selection', 'Text for tooltip and Security'. Enter the variable name with the aid of the input assistant (<F2>).

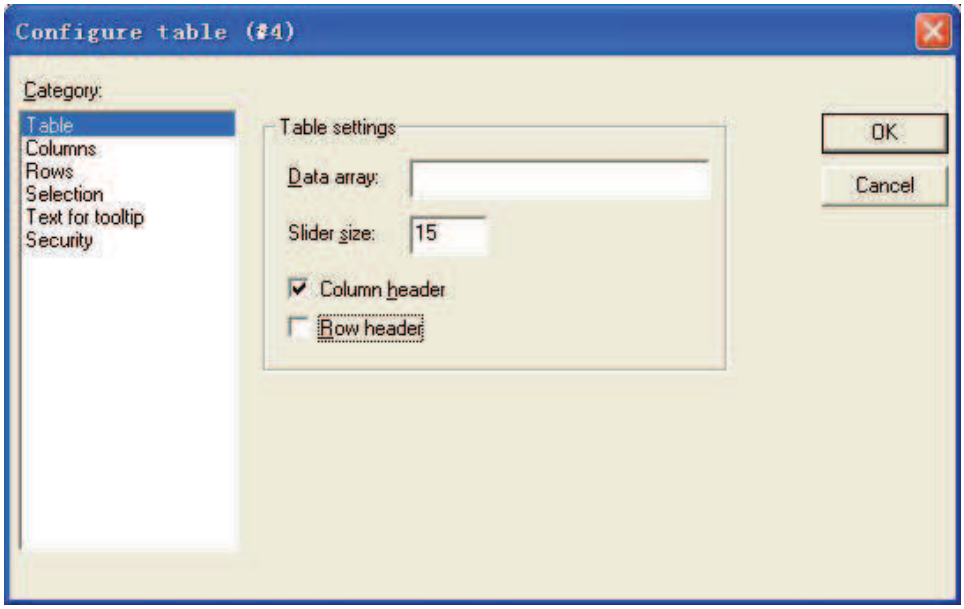


Figure 11-8-2 Configure table

11.9 TREND

You can insert a table in a visualization object, shown in figure 11-9-1.

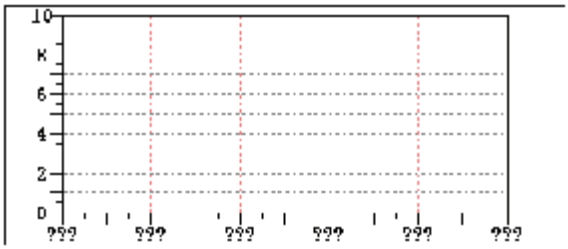


Figure 11-9-1 Insert Trend

Double click the trend and a dialog for configuration of a trend element is opened, shown in figure 11-9-2, and you can set the options Trend, Colors, Text for tooltip and Security. Enter the variable name with the aid of the input assistant (<F2>).

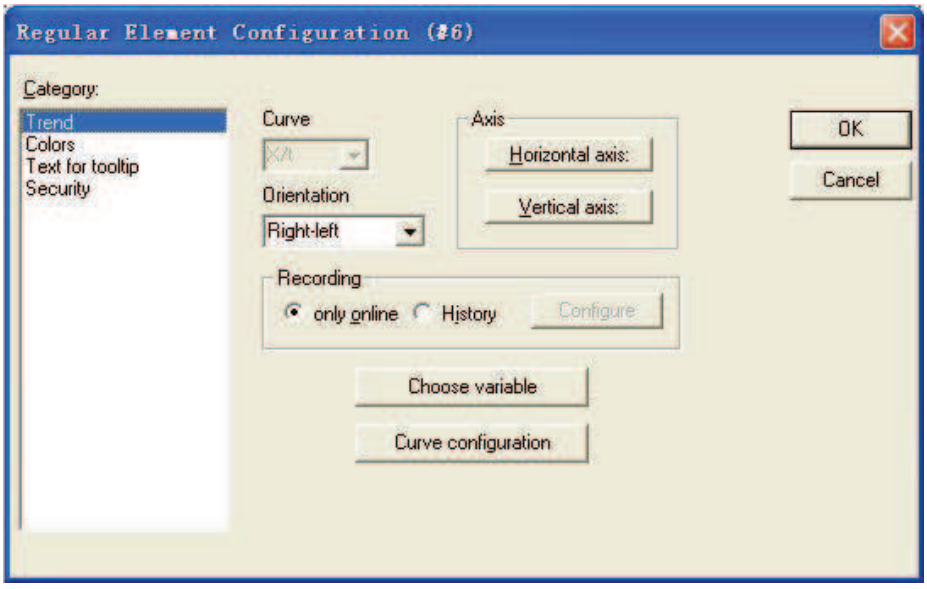


Figure 11-9-2 Trend Configuration

11.10 ALARM TABLE

You can insert an alarm table in a visualization object, shown in figure 11-10-1.

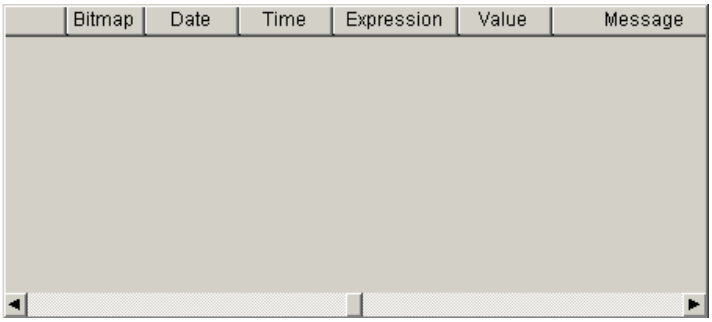


Figure 11-10-1 Insert Alarm Table

Double click the alarm table and a dialog for configuration of a alarm table is opened, shown in figure 11-10-2, and you can set the options ‘Alarm table’, ‘Columns’, ‘Settings for sorting ’, ‘Selection settings’, ‘Text for tooltip and Security ’. Enter the variable name with the aid of the input assistant (<F2>).

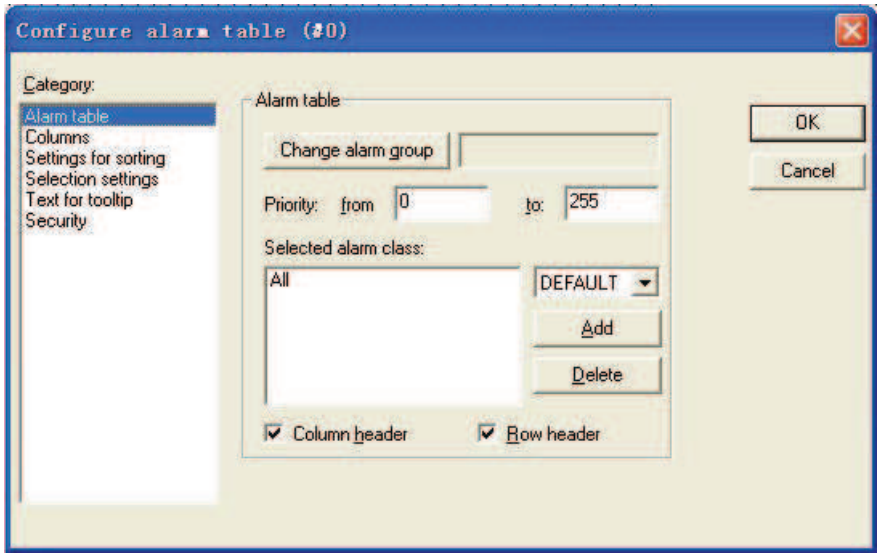


Figure 11-10-2 Configure Alarm Table

11.11ACTIVEX ELEMENT

You can insert an ActiveX element in a visualization object, shown in figure 11-11-1.

Figure 11-11-1 Insert ActiveX element

Double click the ActiveX element and a dialog "Configure ActiveX element " is opened, shown in figure 11-11-2, and you can set the options 'Control', 'Methodcalls' and 'Display'. Enter the variable name with the aid of the input assistant (<F2>).

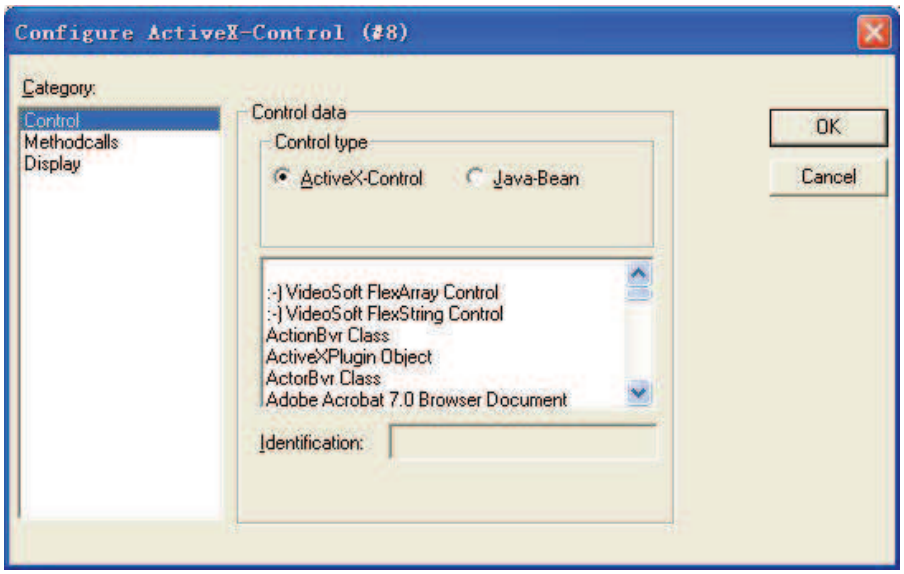


Figure 11-11-2 ActiveX-Control Configuration

11.12 VISUALIZATION APPLICATION

Take a little program for an example to describe the visualization application. In this program two signal lamps flash in turn. The program is displayed in figure 11-12-1.

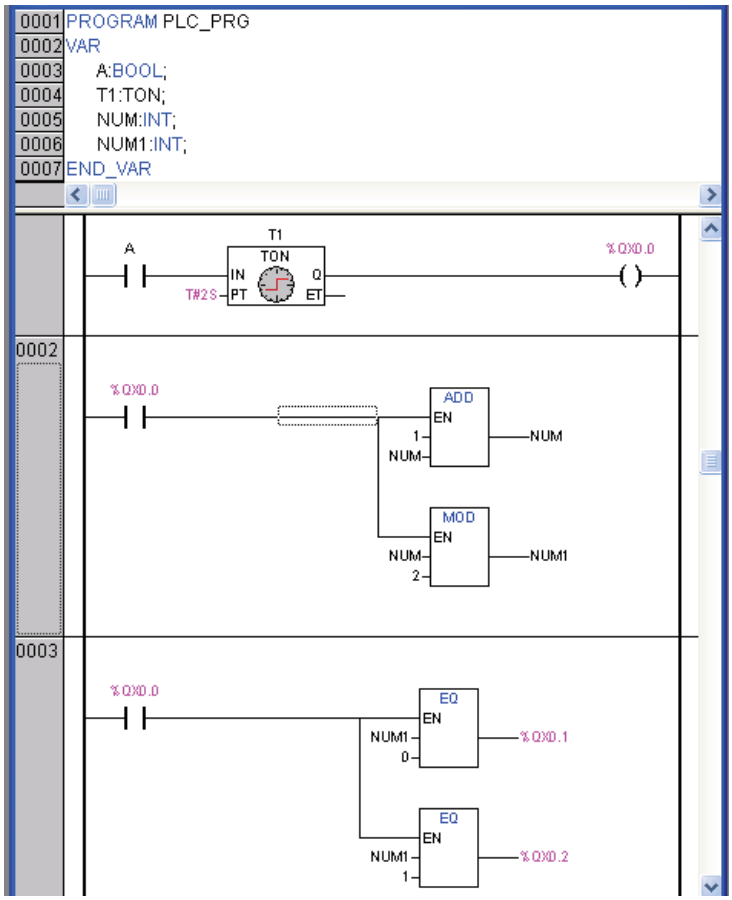


Figure 11-12-1 Program

In order to see the flash more clearly, one can define a visualization object in which the on and off of the signal lamp can be displayed in different colors. The on 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", shown in figure 11-12-2.

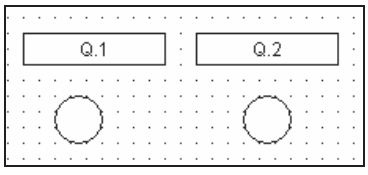


Figure 11-12-2 Visualization A

Select Q.1 and double click on it then a dialog box "Regular Element Configuration " is opened. Select "Colors"/ "Color"/ "Inside" and choose the green color and click the "OK" button, 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, shown in figure 11-12-4. In the same way, enter"Q.2" in the field"Change color" of"Q.2".

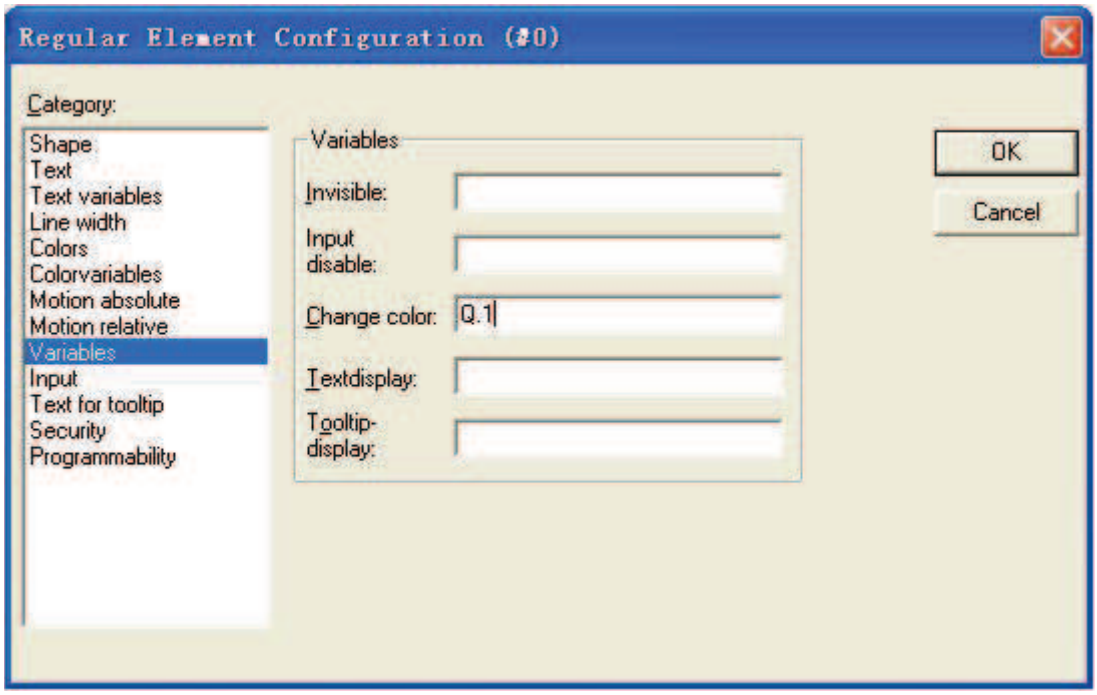


Figure11-12-4 Variable Setting

Now the configuration of visualization A is finished. The result in online ode is displayed infigure 11-12-5. Q.1 and Q.2 flash in turn.

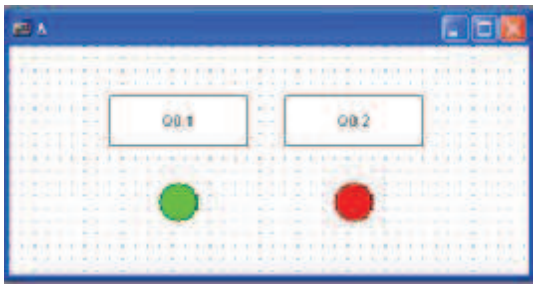


Figure 11-12-5 Result in Online Mode

CHAPTER 12 APPENDIX

Module Memory

The program memory of the moules in the following table is 28KB. In the dialog box "Target Settings" select "PowerPro LEC G3 CPU" in the field "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 moules is 120KB. In the dialog box "Target Settings" select "PowerPro LEC G3 CPU Extend" in the field "Configuration".

Input Assistant in PowerPro

For the users who are not familiar with standard programming language, the command "Edit"/"Input Assistant" or <F2> provides a dialog box "Help Manager" for choosing possible inputs (Standard Functions, Function Blocks, Operators, Operands and Variables) at the current cursor position, 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;

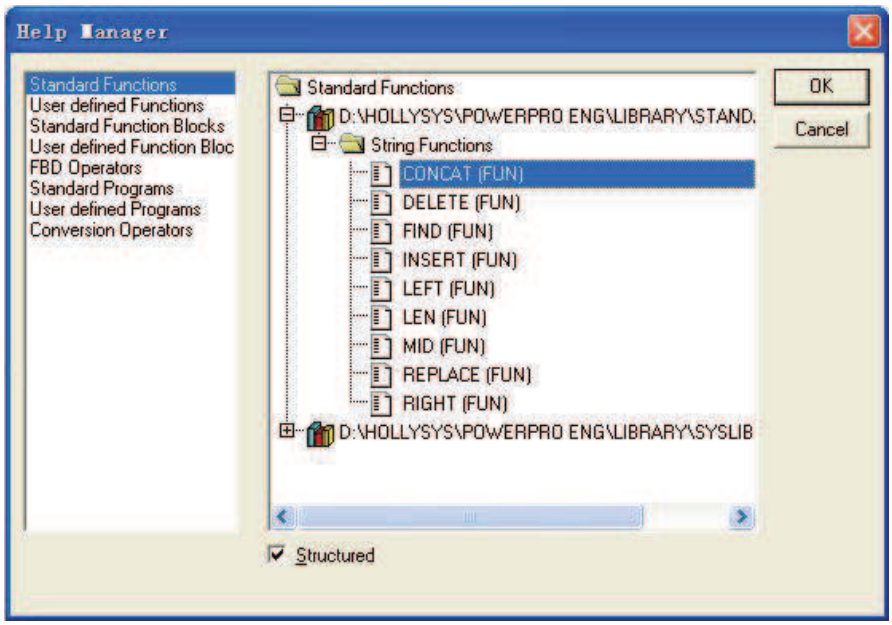


Figure Appendix-1 Help Manager

Key Combinations in PowerPro

General Functions	
Move between the declaration part and the instruction part of a POU	<F6>
Move between the Object Organizer, the object and the message window	<Alt>+<F6>
Move to the next open editor window	<Ctrl>+<F6>
Move to the previous open editor window	<Ctrl>+<Shift>+<F6>
Right Key menu	<Shift>+<F10>
Shortcut mode for declarations	<Ctrl>+<Enter>
Move from a message in the Message window backto the original position in the editor	<Enter>
Open and close multi-layered variables	<Enter>
Open and close folders	<Enter>
Switch register cards in the Object Organizer and the Library Manager	<Arrow Keys>
Move to the next field within a dialog box	<Tab>
Help	<F1>
General Commands	
"File" /"Open"	<Ctrl>+<O>
" File " /"Save"	<Ctrl>+<S>
" File " /"Print"	<Ctrl>+<P>
" File " /"Exit"	<Alt>+<F4>
"Edit" /"Undo"	<Ctrl>+<Z>
" Edit " /"Redo"	<Ctrl>+<Y>

" Edit " /"Cut"	<Ctrl>+<X>
" Edit " /"Copy"	<Ctrl>+<C>
" Edit " /"Paste"	<Ctrl>+<V>
" Edit " /"Delete"	<Del>
" Edit " /"Find next"	<F3>
" Edit " /"Replace"	<Ctrl>+<H>
" Edit " /"Input Assistant"	<F2>
" Edit " /"Auto Declare"	<Shift>+<F2>
" Edit " /"Next Error"	<F4>
" Edit " /"Previous Error"	<Shift>+<F4>
"Project" /"Rebuild all"	<F11>
" Project " /"Delete Object"	<Del>
" Project " /"Add Object"	<Ins>
" Project " /"Rename Object "	<Spacebar>
" Project " /"Open Object "	<Enter>
"Online" /"Login"	<Alt>+<F8>
" Online " /"Logout"	<Ctrl>+<F8>
" Online " /"Run"	<F5>

" 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>